SENSOR WEB AND GEOSPATIAL CLOUD COMPUTING MODELING AND ITS APPLICATION IN REAL-TIME COLLABORATIVE EARTH OBSERVATION DATA PROCESSING

XIAO FEI

Ph.D

The Hong Kong Polytechnic University

2016

The Hong Kong Polytechnic University

Department of Land Surveying and Geo-Informatics

Sensor Web and Geospatial Cloud Computing Modeling and its Application in Real-time Collaborative Earth Observation Data Processing

XIAO Fei

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

December 2015

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

_____Fei Xiao (肖斐)_____(Name of student)

ABSTRACT

Sensor Web and Geospatial Cloud Computing Modeling and its Application in Real-time Collaborative Earth Observation Data Processing

The geospatial science is one of data-intensive domains, where research and development typically produce and analyze large volumes of distributed heterogeneous geospatial data sets. The recent advancements of sensor network and computing technologies have resulted in an explosion of geospatial data. In addition, scientific workflows and Web Services have been widely employed in geospatial data infrastructures. These technologies allow distributed data and model resources to be accessed and chained together to achieve complex scientific problems. The emergence of cloud computing provides a new way for processing big geoscience data by dynamically scheduling computing and storage resources over the Internet. Although the geospatial community tends to deploy the Earth Observation and geospatial model resources onto the cloud, there are still some challenges on effectively applying cloud computing paradigm to manage and analyze big geoscience data. First, the service-orientated cloud computing paradigm is transforming traditional geoscientific workflow management system from a close and centralized control system into a worldwide dynamic business process, which always consists of complex interactions among a large set of geographically distributed processing resources deployed and maintained by various organizations. Out of the necessity, these complex applications need to make use of large volumes of heterogeneous data and be executed in distributed computing environments. Furthermore, Current web-based GIS or RS applications generally rely on centralized structure, which has inherent drawbacks such as single points of failure, network congestion, and data inconsistency. The inherent disadvantages of traditional GISs need to be solved for new applications on Internet or Web.

To address these challenges, this research presents the Hypercube Geospatial Service Framework (HyperCGSF), an agent-based framework comprising a scalable architecture and a set of distributed algorithms for decentralized enactment of construction and execution of geospatial processing workflows in the cloud computing environment. Using the Integrated Dust storm Detection Model (IDDM) as a case study, this research investigates how geospatial cloud computing and Earth Observation Sensor Web technologies can be utilized to realize standard-compliant geospatial web services, service composition, model input integration, and output utilization. Additionally, this research will explore how to apply a scalable hypercube Peer-to-Peer (P2P) topology to organize an arbitrary number of geospatial service agents, which can then collaborate in the decentralized execution and monitoring of geospatial workflows. Contrary to traditional centralized approaches (e.g. BPEL), each service agent does not fully take charge of executing the whole workflow and all of the processes in a workflow are evenly distributed among the participating nodes in a fine-grained manner. An experimental evolution of HyperCGSF and a comparison with traditional centralized BPEL engine architecture demonstrate that the proposed HyperCGSF can dramatically decrease the execution times of complex workflow and increase the stability of the whole systems.

PUBLICATIONS ARISING FROM THIS RESEARCH

Journals

1. **Fei Xiao**, Geoffrey Y. K. Shea, Jiannong Cao, Man Sing Wong, Zheng Wu. 2016. Decentralized orchestration of composite geospatial processing services based on OGC web processing service standard. *Computers & Geosciences.* (Submitted).

2. **Fei Xiao**, Man Sing Wong, Kwon Ho Lee, James R Campbell, Geoffrey Y. K. Shea, 2015. Retrieval of dust storm aerosols using an integrated Neural Network model, *Computers & Geosciences, 102(85):104-114.*

3. Wong M. S., **Xiao F.**, Nichol J. E., Shea, G.Y. K., Fung J., Kim J., Campbell J., Chan P. W., 2015, A Multi-Scale hybrid neural network retrieval model for dust storm detection, a study in Asia, *Atmospheric Research*, 158, 89-106.

Conference Papers

1. **Fei Xiao**, Geoffrey Y. K. Shea, Man Sing Wong, James Campbell, 2014. An Automated and Integrated Framework for Dust Storm Detection Based on OGC Web Processing Services. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 151-156

2. **Fei Xiao**, Geoffrey Y. K. Shea, Jiannong Cao, 2015. Decentralized Orchestration of Composite OGC Web Processing Services in the Cloud, First International Conference on Smart Data and Smart Cities September 7-9, 2016, University of Split, Croatia (accepted).

ACKNOWLEDGEMENTS

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First I must give a special thanks to my parents. Their unconditional love and support are the reasons that I am where I am today. Words cannot express how much I love you both.

Then I wish to thank Dr. Geoffrey Y. K. Shea, who has been the thesis supervisor. His wise advice, insightful criticisms, and patient encouragement aided the writing of this thesis in innumerable ways. I extend my gratefulness and highest reverence to him for his tolerance, restless academic and personal support. His astuteness, knowledge and commitment to the highest standards inspired and motivated me.

I also thank Prof. Jiannong Cao, who has been my co-supervisor. His vast knowledge in computing science and his dedication to science also served to establish excellent examples for me. I am fortunate to have him as my co-supervisor.

I also thank the department of LSGI for their support and assistance since the start of my postgraduate work in 2012, especially the head of department, Prof. Wenzhong Shi, who inspires me all along since I was enrolled in LSGI in 2009 and offers me a lot of help. Special thanks to Dr. Man Sing Wong (Charles) for his valuable comments and support during my studies. His attitudes towards researching have been inspiring me all along and his tips have been very useful to handle my PhD studies.

I would also to thank the staff of the department of LSGI, especially Vaness, Ziki, and Justin for their enthusiastic assistant. I also would like acknowledge the

financial, academic and technical support of The Hong Kong Polytechnic University and its staff, particularly in the award of a Postgraduate Research Studentship that provided the necessary financial support for this research.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AOT	Aerosol Optical Thickness
API	Application Programming Interface
BDI	Belief-Desire-Intension
BTD	Brightness Temperature Difference
CDM	Common Data Model
DE	Digital Earth
DGSPA	Distributed Geospatial Service Planning Algorithm
DTDA	Dynamic Task Dispatching Algorithm
EO	Earth Observation
EWRD	Embedded WPS Request Document
GCI	Geospatial Cyber Infrastructure
GDAL	Geospatial Data Abstraction Library
GeoSPA	Geospatial Service Provider Agent
GeoSPI	Geospatial Service Programming Interface
GIS	Geographic Information System
G-FSM	GeoSPA Finite State Machine
GP	Geospatial Process
GPW	Geoprocessing Web
HyperCGSF	Hypercube Geospatial Service Framework
HBase	Hadoop Database
HDF	Hierarchical Data Format
HDFS	Hadoop Distributed File System
IDDI	Infrared Dust Difference Index
IDDM	Integrated Dust storm Detection Model
IOE	Integrated Operating Environment
LST	Land Surface Temperature

MaaS	Model as a Service
MAS	Multiagent System
MIME	Multipurpose Internet Mail Extensions
MTSAT	Multi-functional Transport SATellite
NcML	NetCDF Markup Language
NCSS	NetCDF Subset Service
NetCDF	Network Common Data Form
OGC	Open Geospatial Consortium
OpenMI	Open Modeling Interface
OWS	OGC Web Services
P2P	Peer-to-Peer
PDG	Program Dependence Graph
RAT	Reverse Absorption Technology
RDBMS	Relational Databases Management Systems
Reff	Effective Radius
REST	Representational State Transfer
RS	Remote Sensing
SBDART	Santa Barbara DISORT Atmospheric Radiative Transfer
SDI	Spatial Data Infrastructure
SOA	Service-oriented Architecture
SQL	Structured Query Language
SWE	Senor Web Enablement
TDS	THREDDS Data Server
THREDDS	Thematic Real-time Environmental Distributed Data Services
TVAP	Three-bands Volcanic Ash Product
WCS	Web Coverage Service
WMS	Web Map Service
WMTS	Web Map Tile Service
WPS	Web Processing Service

Chapter 1: Introduction

1.1 Research Background

Geoscience theories and technologies have developed dramatically over the last three decades and the computational environment for Geographic Information Science (GIS) and Remote Sensing (RS) have evolved from traditional desktop to a web-based and service-oriented architecture (SOA) (David, 2005). As one essential technology of SOA, the Web Service has been widely used by the government, companies, and research organizations for building core enterprise systems because of some important features of Web Service such as flexibility, reusability and platform independence (Shen et al., 2007). Geospatial science can prominently benefit from Web Service technologies, which enable users to deploy, discover, and access geospatial resources faster and more efficiently by separating the geospatial service description from its implementation. It is believed that web-based distributed geospatial services and large-scale of collaborating applications are the next development trends of geoscience (Kiehle et al., 2007). In order to make geoscience data and models more accessible and foster the interoperability of geospatial resources, many research works have been conducted focusing on building geospatial warning and decision support systems through Web Service techniques, such as the Model Web (Geller and Turner, 2007; Granell et al., 2010; Thiebes et al., 2013; Nativi et al., 2013), Geoprocessing Web (Chen et al., 2010 and Zhao et al., 2012), CyberGIS (Yang et al., 2010 and Wang et al., 2013), and Model as a Service (MaaS) (Geller and Melton, 2008; Roman et al., 2009; Yang, Xu, and Nebert, 2013; Li et al., 2014).

In addition, to better understand, protect and improve our living environment, a variety of sensors have been developed and deployed for monitoring the Earth and accumulating valuable records. The advancements of sensing technologies have

dramatically improved the accuracy and spatiotemporal scope of the record. In recent years, the realization and development of Earth Observation (EO) Sensor Web (Di et al., 2010) is one of the most important achievements in geospatial science. The EO Sensor Web refers to the realization and development of a continuous, distributed, and cooperative EO data service system that aims at integrating and coordinating the multiple heterogeneous Earth monitoring platforms to achieve complex EO tasks (Chen et al., 2013). The advancements of sensing technologies dramatically increase people's capabilities in acquiring geospatial data for building the Digital Earth (DE) (Craglia et al. 2012) and Spatial Data Infrastructure (SDI) (Masser et al., 2005). Based on Yang et al. (2011), massive amounts of multi-dimensional data recording various physical phenomena are taken by the sensors across the globe, and these sensing data are collected rapidly with a daily increase rate of terabytes to petabytes. This increase is dramatically enhanced by novel crowd sourcing in situ ground-based sensor networks as well as the deployment of satellite systems which generates EO data with very high resolution (Zhao et al., 2012). For example, the National Aeronautics and Space Administration (NASA)'s Earth Observation System (EOS) satellites collect alone 1000 terabytes annually (Clery and Voss, 2005), and The geostationary satellite Himawari-9 for meteorological observation tasks generates more than three terabytes (TB) of data per day (Minchin, 2014).

Because of the instantaneity and flexibility, the EO Sensor Web technologies have been widely used in many geoscience projects including disaster monitoring and assessment, climate change, ecosystem dynamics, and atmospheric pollution monitoring, which produce massive volumes of geospatial data, or big geoscience data (Li et al., 2015). The geoscience is typically data-intensive domain, which always involves large volumes of heterogeneous geospatial data. The needs for high performance, big data analysis for modeling and simulation of geospatially enabled content are greater than ever. In addition, many geoscience problems are also experiencing an increased demand for computing resources, quantification of information, and making large spatial data available over the web (Vaccari et al.,

2009). The geoscience community has recognized that it is critical to leverage current advanced information and network technologies to share EO resources and relevant geospatial processing services to effectively achieve the global challenges, e.g. climate change, atmospheric pollution, and earthquake prediction (Gerard et al., 2013). Yang et al. (2011) proposed several great challenges in the geoscience community to achieve the goals of DE and SDI, which can be categorized into four aspects: data intensity, computing intensity, concurrent intensity, and spatiotemporal intensity. Recent advancements of cloud computing technology provide potential solutions to address these grand challenges, which is the motivation of this research.

In recent years, the SOA paradigm has been replaced by cloud computing in the software industry on a broader scope. The cloud computing can provide the computing resource, storage space, web services, and other software in a dynamic and scalable manner via the Internet, which enables service consumers to rent computing resources on demand (Zhang et al., 2010). The cloud computing is also a new generation computing paradigm to handle the dynamic demands on computing resources for processing Big Data. The recent advancements of cloud computing offers new approaches for dynamically scale compute-intensive tasks or web applications in the presence of temporarily large number of concurrent access (Laniak et al., 2013). Furthermore, the increasing tendency of network service users to use cloud computing encourages Web Service providers to develop and deploy more and more geospatial services with various functionalities to the service pool, which is a key characteristic of cloud computing. The cloud computing technology has been utilized by different geospatial sciences and applications including storing and acquiring EO data, extracting parameters, configuring and running models, obtaining knowledge, making decisions, and collecting users' feedback.

Regardless of the type and content, each piece of information in the cloud can be described as 'resource', and each resource has a set of properties as well as the relationships with other resources. The geospatial resources are 'born distributed',

where geospatial data are collected, stored, and processed at different locations across the Internet (Berners-Lee et al., 2001). Generally, the geospatial resources distributed over the cloud computing environment can be divided into three categories: (i) computational resources, (ii) data resources and (iii) model resources, which are shown in Figure 1.1.

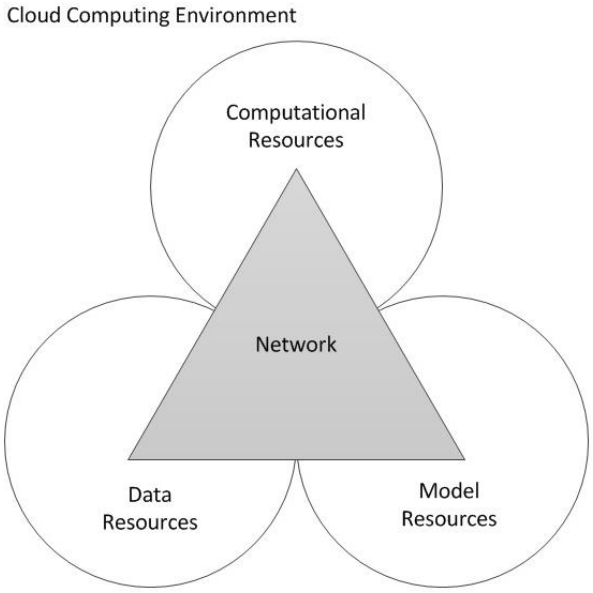


Figure 1.1. Illustration of diverse geospatial cloud computing resources distributed over the Internet

Scientific workflows have been widely used to combine these diverse resources to achieve more complex tasks. The term 'service chain' is an implementation of a workflow under the SOA (Friis-Christensen et al., 2009). It is significant to chain interoperable geospatial data and processing resources to achieve more complex tasks than the individual model alone (Dubois et al., 2013). Moreover, the emergence and development of DE and SDI are transforming the globe into an interconnected and mutual influenced organic unity which requires efficiently integration of independently developed GIS platforms to share knowledge and collaborate among diverse organizations worldwide.

1.2 Problem Statements

Although the geospatial community tends to deploy the Earth observation and geospatial model resources onto the cloud computing environment, there are still some challenges on effectively applying cloud computing paradigm to manage and analyze big geoscience data, which are introduced below.

1.2.1 Complex Interaction between Geospatial Processing Services

Geoscience problems are intrinsically complicated to analyze and model because of the complex and dynamical characteristics of the Earth system (Yang et al., 2011a). Solving comprehensive geographic problem (e.g. environmental disaster monitoring and evaluation) often requires the integration of multi-disciplinary geospatial models and massive observational data. Interactions among geospatial processes and datasets in the spatial or temporal dimensions are intrinsically complicated (Donner et al., 2009). The Web Service technologies have been intensively researched for decades to address this problem. However, web-based geospatial services are slightly different from traditional web service technologies because the geospatial data on which the geospatial processing services operate is always diverse, huge, and complex (Granell et al., 2007). Furthermore, the heterogeneity of existing geospatial models, data formats, semantics, as well as the complex spatial relationships dramatically complicate the integration of geospatial resources, which in practice are the most prominent limiting factors for the achievement of geospatial interoperability (Granell et al., 2010).

Moreover, geospatial processing workflow always consists of a large amount of geographically distributed geospatial services which are independently developed, maintained, and published by different organizations. The interoperability of the distributed geospatial services still remains a grant challenge for geospatial science disciplines (Nativi et al., 2013). It is complicated and expensive to make the independently developed geospatial service resources interoperable because of the

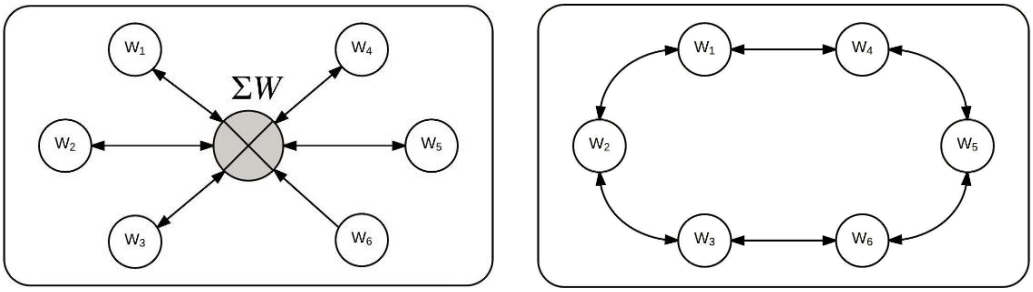
complex dependencies. The manner of handling the dynamic and complex interaction between these geospatial model resources for achieving complex geoscience task is a great challenge.

1.2.2 Geospatial Big Data and Data Intensity

With the rapid development of EO technologies, large amounts of geospatial data with multiple layers and dimensions are collected from heterogeneous sensing platforms including satellite imageries, airborne sensing images, and in-situ observations. These multiple data sources make the volume of EO data grow in a geometric progression. Yet model simulation result and geospatial dataset always have more than two dimensions with corresponding geospatial elements such as geographic coordinate systems, projections, and time series. These data is published and maintained by globally distributed-organizations over the entire Earth (Li et al., 2010). However, due to the limitation of the processing capability of a single server, traditional centralized geospatial service framework is hard to support the efficient storage and processing of EO data through Internet (Li et al., 2008). Besides, geospatial data analysis tasks always need to deal with large volumes of heterogeneous data with different data formats (e.g. text, raster-based data, and binary file), and share the analysis result over the Internet. Effectively managing, analyzing and storing these big geoscience dataset are grand challenges in geospatial science (Cui et al., 2010; Liu et al., 2009).

1.2.3 Centralized and Decentralized Web Service Composition

Web Service composition is the technology of combining a set of interconnected Web Services to create a more complex, cross-organizational, and value-added business process. Generally, the description and execution of Web Service composition can be divided into two categories: service orchestration and service choreography (Tong et al., 2011), which are shown in Figure 1.2.



(a) (b)

Figure 1.2. (a) service orchestration, and (b) service choreography

Service orchestration can be described as a service composition and all of the participating services are coordinated by a central orchestrator, i.e. $\sum W$. As shown in Figure 1.2(a), there are n participating services in the service orchestration model and each of which needs to communicate with $\sum W$ by message exchanges. On the other hand, service choreography has no central controller and all of the participated service nodes can collaborate with each other directly to achieve a common goal. As illustrated in Figure 1.2(b), the service choreography approach tracks the sequence of messages among n independently autonomous services directly, rather than a specific business process that a single party executes.

Given the distributed nature of the cloud computing and relevant technologies involved, geospatial workflow applications are inherently distributed (Lee et al., 2008). Currently, most of the geospatial service composition technologies or frameworks are built based on the centralized manner, which relies on one central orchestrator for the execution of the overall geospatial services (Zhao et al., 2012b; Pantazoglou et al., 2014). However, these centralized approaches for compositing geospatial processing services always suffer from performance bottleneck for the reasons analyzed below.

First, as introduced in section 1.2.2, many web-based geoscience applications always need to collect and integrate data from multiple locations, and a geospatial processing service can output a large amount of data which may be irrelevant to other

processing service in the service chain. However, these data are transferred to the central controller where it is discarded, which causes unnecessary communication load on the network. Hence, huge volume of data may be transferred over the Internet, which makes it a bottleneck for geoscience applications.

Second, in the cloud computing environment, web services are distributed across physical and geospatial boundaries, which are constantly removed and updated (Tong et al., 2011). The geospatial processing services become more fragile when they are deployed on the Internet due to network connection failure or server downtime. Currently, the main Web Service composition technologies are mainly based on a centralized manner, which may suffer from performance bottleneck and the failure of single node. In addition, when the scale of geoscience application network keeps increasing, it will be very difficult and expensive to discover the required services among the thousands of distributed geospatial service nodes that offer services (Tan et al., 2014). Finally, an ever-changing runtime environment may result in generating lots of various decentralized version of geospatial processing services at runtime. Considering such dynamicity in workflow execution raises the problem that there is lack of suitable software architectures to support the execution of different decentralized version of geospatial process (Safi Esfahani et al., 2011). Based on the analysis above, it can be concluded that developing a more fine-grained geospatial service planning and allocation approach is urgent for ensure the scalability of the geospatial processing workflow execution at lower cost.

Finally, the development of distributed geospatial processing services and the popularization of web and wireless devices enabled massive numbers of end users to access geospatial systems at the same time (Goodchild 2007). The real-time processing of EO data requires geospatial processing services own the capability to handle the pressure of high concurrency caused by sudden increment of the number of users and give fast response (Bodk et al., 2010). New software paradigms offered by cloud such as software, platform, and infrastructure as services always receive a

large number of requests. Particularly, in the case of geospatial workflow engine as a service, a large number of workflow instances are requested from different users all around the world. Consequently, it results in creating thousands of concurrent executing instances of geospatial processing services. So there it is urgent to investigate how to leverage cloud computing technologies to improve the performance of geospatial service compositions, enable the computability of concurrent-access-intensive geoscience problems, and hide the complexity of the computing infrastructure so that scientists can focus on resolving scientific problems (Yang et al., 2011b).

In summary, it is hard and expensive for the traditional centralized approach to expand in the presence of a potentially large number of simultaneous, long running geospatial processing service instances that produce and consume voluminous data. The decentralized service choreography can address these problems efficiently for inherent structural features (Gutierrez-Garcia et al., 2013; Tan et al., 2015). Based on the Figure 1.2, the service choreography model is more collaborative than service orchestration model because it can achieve efficient interactions and collaborations among multiple services (Nanda et al., 2004). However, the way of achieving complex and dynamic coordination and cooperation between the distributed geospatial service providers without the centralized control is still a great challenge.

1.3 Goals and Contributions

The major objective of this research is to propose a cloud-based geospatial service framework, called Hypercube Geospatial Service Framework (HyperCGSF), to address the challenges introduced above. The HyperCGSF consists of a multifunctional geospatial service provider agent model, an underlying networking topology called ‘hypercube’, and a set of distributed algorithms to support:

- Efficient publishing, sharing, managing, and accessing the geospatial service resources (data and processes) distributed over the cloud (Chapter 3).

- Automatic discovery and composition of geospatial services to achieve complex geoscience tasks (Chapter 4.2).
- Orchestration of geospatial processing services in a decentralized manner with the features of security, load balancing, and fault tolerance (Chapter 4.3 and 4.4).

The HyperCGSF focuses on the automatic geospatial service sharing, discovery and composition, as well as improvement of the average geospatial process execution time in the presence of multiple, concurrent and long running geospatial processing instances through a set of fully decentralized algorithms. In summary, the main contributions of this research were shown in Figure 1.3.

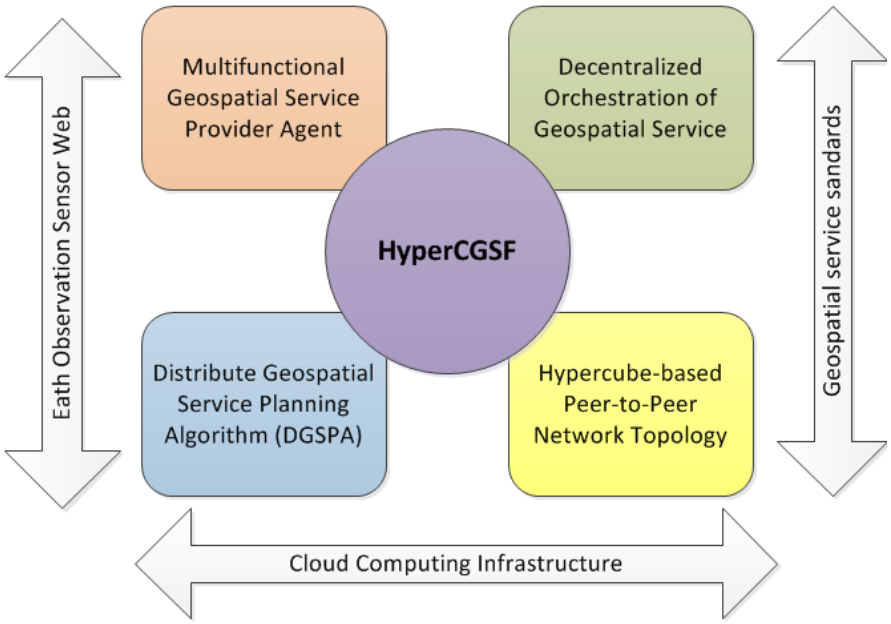


Figure 1.3. Contributions of the HyperCGSF

As shown in Figure 1.3, the contributions of this research consists of four aspects: a multifunctional Geospatial Service Provider Agent (GeoSPA) for offering web-based geospatial services, a hypercube-based Peer-to-Peer (P2P) network topology for organizing distributed GeoSPAs, a distributed geospatial service planning algorithm for service discovery and composition, and a fully decentralized approach to orchestration of geospatial processing services. Developed based on the

SOA and open geospatial service standards (e.g. OGC Web Service), the HyperCGSF can be conveniently deployed onto commercial cloud computing infrastructures (e.g. Amazon Elastic Compute Cloud, Google Computing Engine, and Windows Azure) and seamlessly integrated with the EO Sensor Web to supply RESTful-based geospatial services. The details of these four contributions were introduced further in detail below.

- **GeoSPA**

The GeoSPA is designed as a multifunctional service hub for managing the distributed geospatial service resources in cloud. Different geospatial service resources can be registered and managed by GeoSPA in a configurable manner. Three service models were predefined for GeoSPA, which are EO data service model, processing service model, and computing service model. These three service models make GeoSPA as a one-stop solution for building SDI in cloud. To ensure the interoperability and collaboration among service agents in HyperCGSF, all GeoSPA service models are fully compliant with the OGC Web Service specifications.

In addition, this research proposed a Geospatial Service Programming Interface (GeoSPI), which can significantly facilitate the model developers to deploy geospatial model entity onto GeoSPA in the form of web-based geospatial processing service fulfilled with OGC Web Service (OWS) standard specification. When the service provider registers a geospatial processing service onto GeoSPA through GeoSPI, the knowledge-base embedded in GeoSPA can automatically update its knowledge according to the description of geospatial process and dependent relations with other service agents. Through this way, GeoSPA can not only work independently to offer geospatial services, but also cooperate with other GeoSPAs to achieve more complex task.

- **A Hypercube-based P2P Network Topology**

Current web-based GIS applications generally rely on centralized structure, which has inherent drawbacks such as single point of failure, network congestion, and data inconsistency, etc. These inherent disadvantages of traditional GISs need to be solved for new applications on Internet or Web. To overcome these problems, a scalable P2P network topology called hypercube is applied to link and coordinate multiple GeoSPAs to resolve complex scientific problems. By utilizing the hypercube topology, the HyperCGSF can fully exploit the elasticity capabilities of cloud computing platforms by dynamically increasing the dimensions of hypercube on demand. Several algorithms were also proposed for automatically adding or removing the service agents at runtime, which makes HyperCGSF able to timely and effectively react to the changing workload in the high dynamic environment of cloud computing without affecting the performance of the whole system.

- **Distributed Geospatial Service Planning Algorithm**

Based on the GeoSPA service models and the Hypercube-based network topology, a Distributed Geospatial Service Planning Algorithm (DGSPA) is proposed for automatic geospatial processing service composition in the dynamic and complex distributed computing environment. Based on the distributed decision making of the GeoSPAs, DGSPA has better scalability and addresses the distributed nature of service composition in the cloud computing environment. The testing result of DGSPA using real-world study case indicates that DGSPA is effective and flexible for achieving complex geoscience tasks by composing distributed geospatial processing services.

- **Fully Decentralized Approach to Orchestration of Geospatial Services**

Decentralized orchestration offers performance improvements in terms of increased throughput and scalability and lower response time. The model developer

can deploy geospatial model onto GeoSPA as a geospatial process, which is then exposed as web-based geospatial processing service. Every GeoSPA is responsible for managing a set of geospatial processes. The GeoSPA is not fully responsible for managing the execution of the whole workflow; rather, it contributes by executing one or more processes in the workflow and maintains the result generated by these processes. Thus, the geospatial processing service composition and execution tasks do not need any central controller. All GeoSPAs in a HyperCGSF system can automatically negotiate with each other to exchange instantiate process instances and coordinate the execution of various task instances in a decentralized manner. Additionally, the HyperCGSF achieves a real parallelism by allocating the geospatial processing tasks to different GeoSPA for execution. The feature is particularly useful for the data-intensive application which always involves large amounts of data exchanging. Instead of transferring a large volume of data through the central controller, the geospatial process can migrate to target location for execution, which minimizes the low efficiency of large-volume transfer of spatial data on a cloud computing environment and ensure data integrity.

1.4 Thesis Structure

The thesis consists of seven chapters. This chapter of an introductory nature discusses the study by introducing research background and analyzing challenges, potential solutions, and the contributions of this thesis. Chapter 2 reviews related work, including spatial cloud computing, Web Service composition, comparison of current EO systems, geospatial service frameworks, and P2P technologies as well as their potential contributions in geoscience. Chapter 3 presents the structure of geospatial service provider agent and geospatial service programming interface (GeoSPI) which can be utilized by mode developers to expose geospatial process as standard-compliant Web Service. Chapter 4 investigates how to apply a hypercube-based P2P topology to manage the service agents and achieve the decentralized orchestration of geospatial services. Several widely used approaches

for dust storm detection through remote sensing technology are introduced and analyzed and the integrated dust storm detection model (IDDM) was proposed as the study case to evaluate the proposed HyperCGSF in Chapter 5. Chapter 6 brings the system evaluations and discussions. Chapter 7 summarizes the research and discusses future work.

Chapter 2: Literature Review

Over the past half century human's capability to explore the Earth system has been enhanced with the emergence of new computing, sensor and information technologies (Yang et al., 2008). While the technological advancements accelerate collecting, simulating and sharing geoscience data, they also produce Big Data for geosciences. In recent years, the importance of affordable access to reliable high-performance hardware and software resources and avoiding maintenance costs and security concerns has encouraged large institution managers and stakeholders of information technology companies to migrate to cloud computing. The birth of giant trustworthy clouds has led to a dramatic reduction in apprehension toward such an approach. The unprecedented flexibility and scalability provided by cloud bring new approaches for geoscientists to process large volumes of big geoscience data.

This section provides a brief overview of cloud computing and EO Sensor Web technologies as well as their applications in geoscience. Because the investigated subject is very extensive, it is impossible to include all relevant topics. Hence, some related subjects do not fall within the research scope of this review but will be briefly mentioned. First, the recent development of the cloud computing technologies was reviewed as well as their application in processing Big Data. Second, the EO Sensor Web technologies and their applications were discussed, and a new generation of EO Sensor Web system was analyzed, indicating that make full use of current advancement of cloud computing technologies to enhance the capability of traditional EO Sensor Web system is the next development direction. Then, some research hotpots of managing and analyzing geoscience Big Data, as well as web-based geospatial service composition technologies in the cloud computing environment were introduced. Finally, the P2P technologies were introduced as well as their application in geoscience.

2.1 Cloud Computing and Geospatial Cloud Computing

Cloud computing has become popular in recent years for its flexibility and scalability. Cloud computing is a new generation of computing paradigm for sharing and pooling computing resources to handle the dynamic demands on computing resources posed by many 21st century challenges. The idea of cloud computing is initiated from the networked computing or internet computing, and the term "cloud computing" comes from the general usage of the cloud symbol in the system diagrams of communication systems and networked computing (Yang et al., 2013). Cloud computing expands CPU and bandwidth sharing in the modern IT environment in order to share computing resources more effectively through hardware virtualization and delivering computing resources as a type of service on the Internet (Yang et al., 2013). Cloud computing connects networks of computing resources on the internet through a communication infrastructure.

There have been various definitions for cloud computing. In this study, the definition provided by the National Institute of Standards and Technology (NIST) (Mell, 2011) was applied, which is "…cloud computing is a model for enabling ubiquitous, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models." Figure 2.1 illustrates the cloud computing characteristics, service, and deployment models defined by NIST.

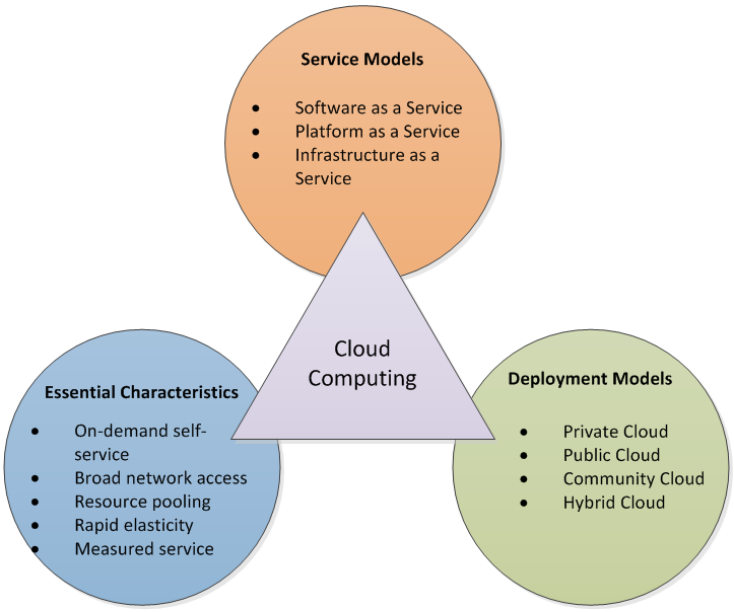


Figure 2.1. Cloud computing characteristics, service, and deployment models defined by NIST

As shown in Figure 2.1, the cloud computing concept defined by NIST consists of three aspects: essential characteristics, service models, and deployment models, which were introduced in further detail below.

2.1.1 Cloud Computing Characteristics

NIST denotes five characteristics of cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service. These five characteristics differentiate cloud computing from other distributed computing paradigms, such as grid computing. The details of these five characteristics were introduced as below:

- *On-demand self-service*. Cloud computing offers computing resources on demand through a "pay-and-go" method. Self-service refers to that the cloud service consumers perform all the actions through web-based service interfaces directly. And the user-specified service request is automatically processed by the cloud infrastructure, without human intervention through an professional IT department.

- *Broad network access*. The service and data resources that are distributed in different provider areas in the cloud computing platforms can be accessed from a wide range of locations and provisioned through standards approaches. Similar terms, e.g. “easy-to access standardized mechanism” (Hamdaqa and Tahvildari, 2012) and “global reach capability” (Yakimenko et al., 2009) are also used to refer to this characteristic.

- *Resource pooling*. Based on Wischik et al. (2008), a resource pool provides a collection of resources simulating the behavior of a single blended resource. This method enables service provider to supply various real or virtual computing resources in a dynamic manner.

- *Rapid elasticity*. Elasticity, or scalability, is the ability to scale up (or scale down) the resources as they need at any time. Elasticity takes advantage of the concept of virtual servers, which are installed using predefined images by removing any manual labor required to extend or reduce computing capacity. Everything is under the control of triggers provided by the system monitoring tooling. If more computing capacity is required, it is immediately initialized and configured within minutes. Yet the released free computing capacity is reduced instantly by system monitoring tooling.

- *Measured service*. The cloud computing resource can be measured and the status report will be returned to user in a real-time manner for optimization.

2.1.2 Cloud Computing Service Model

Based on NIST, the cloud service models are grouped into three different forms, which are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The first form concerns hardware resources, the second form is about runtime environments, and the third area describes the provisioning of software and services. These three forms were introduced as below:

- *IaaS* is the most widely used service model of cloud computing. IaaS supplies physical machines, storage and system software, networks, and other computer infrastructures in the form of virtualized computing resources over Internet. In addition, IaaS allows users to configure, deploy, and run operating systems (OS) and relevant application by themselves. IaaS user doesn't need to care about the infrastructure of application provider and has limited authority to configure relevant setting.

- *PaaS*. Different with IaaS, PaaS is a much higher level service and supplies a platform service for application developers. In addition, PaaS provides a layer of cloud-based software and APIs to computing platforms, which can be applied to develop higher-level web services. PaaS supplies all the toolkits for developing and publishing SOA-based applications and services through the Internet. Google App Engine and Microsoft Azure (www.microsoft.com/windowsazure) are the most popular examples of PaaS.

- *SaaS* is the most widely applied cloud computing service type and supplies various capabilities of applications which are traditionally supplied through the Web browser to end users. Notable examples are Google's Gmail and Salesforce.com. Another example of SaaS is the ArcGIS Online, which is an ArcGIS implementation on the cloud computing environment.

To facilitate data discoverability, accessibility, and processing in geospatial sciences, the Data as a Service (DaaS) and Model as a Service (MaaS) has also been studied in recent years.

- *DaaS* is the least well defined of the four types of cloud services. DaaS facilitates users to discover, share, utilize geospatial data resources distributed on the cloud, and delivers required data and computing resources to end users regardless of physical location of cloud clients and

servers (Olson, 2010). DaaS facilitates geospatial data discovery and utilization on the fly to achieve complex geoscience problems by integrating a layer called middleware layer that collocates and manages data and processing resources aims at optimizing cloud operations (Jiang 2011),

- *MaaS* is a cloud-enabled modeling infrastructure proposed by Li et al. (2014) to capture the technology advancements for dealing with geoscience modeling by: (1) publishing geoscience models as Web Services to hide the complexity of model setup; (2) providing an on-demand ready-to-go model environment, including hardware and software resources; (3) automatically provisioning computing resources to execute multiple model runs in parallel to support many-model-run scenarios and concurrent user accesses; and (4) effectively handling model output for online visualization.

2.1.3 Cloud computing deployment models

Fast development in the utilization of cloud computing leads to publishing more cloud services on the worldwide service pool. Because of the presence of complex and diverse services, a single simple service cannot satisfy the existing functional requirements for many real-world cases. To complete a complex service, it is essential to have a batch of atomic simple services that work with each other. Therefore, there is a strong need to define different cloud computing deployment models, which are introduced below:

- *Public cloud*. This model is the major one of cloud computing deployment model. In a public cloud, based on predefined rules, policies, and pricing model, the cloud provider provides services in the vast majority of cases on the Internet. Handling a large number of wide-spread computing resources enables cloud service providers to supply a consumer various choices to choose the most appropriate resources while considering the Quality of Service (QoS) at the same time.

- *Private cloud*. This approach is designed and established to prepare most of the benefits of a public cloud exclusively for an organization or institute. Considering the wide utilization of corporate firewalls, setting up such a private cloud can lead to decreased security concerns because all of the infrastructures are located inside the organization.

- *Community cloud*. This model is built by the organizations with similar requirements, concerns, and policies in form of a community where member share the cloud computing resources. A third-party service provider or some community members can be responsible for supplying the needed infrastructure of the cloud computing.

- *Hybrid cloud*. Hybrid cloud refers to the creation of a compound cloud model through a combination of two or more various public, private, or community clouds. In hybrid cloud system, each constitutive keeps their specific properties. Yet some standardized or agreed components are also required to enable the communications with each other with respect to portability and interoperability on applications and data.

2.1.4 Spatial cloud computing (SCC)

Geoscience applications have special requirements that cannot be automatically supported by generic cloud computing platforms, because most geospatial algorithms and applications are not designed to leverage multiple CPUs and be delivered through the Internet as a service (Huang et al., 2013). Geoscience phenomena are complex processes and geoscience applications often take a variety of data as input with a long and complex workflow. It becomes then a critical challenge to deliver such complex applications to cloud as a transparent service to support massive numbers of users. Most importantly, both the geoscience and the cloud computing environments are spatiotemporal intensive. However, the middleware used to schedule computing tasks on a cloud computing platform is mostly not developed for

Earth science applications and does not take the spatiotemporal principles and patterns into consideration. Such middleware should be reengineered to support spatiotemporal processing. The SCC seeks to optimize the data, model, and computing resources distributed on the cloud based on the spatiotemporal principles through middleware technologies (Yang et al. 2011a).

Yang et al., (2011a) defined the SCC as "…Spatial cloud computing refers to the cloud computing paradigm that is driven by geospatial sciences, and optimized by spatiotemporal principles for enabling geospatial science discoveries and cloud computing within distributed computing environment". Based on Yang, Spatiotemporal principles should be particularly considered in algorithms, methodologies and phenomena simulations. For example, in atmospheric sciences, the actual number of grid points selected for buffering would greatly impact both computation and forecasting accuracy. In addition, when forecasting dust storm as a weather component, we will consider the time and space interaction, i.e., how time changes impact the space distribution of dust in the atmosphere.

For Earth science models, multiple inputs with strict format are required to execute the models (Xie et al., 2010). Although the required datasets for a model are actually provided online directly, or indirectly, conversion and transformation processes are required. It is very different and time-consuming to obtain such datasets and greater effort has to be made on data processes before datasets can be assimilated by the models. However, no systematic study has been done on how to integrate widely distributed data resources to enable the executions of Earth Science models. Moreover, there are situations where different models must work together to tackle complex problems. These problems cannot be resolved efficiently and accurately by only one model without major modifications to the original models (Zhou et al., 2007).

Based on the features provided by current widely used cloud computing platform, a large number of core GIS operations, such as projection and spatial

analysis, can be implemented as cloud-based web services. To access these geospatial services, the cloud computing users can use a spatial cloud portal which is an integrated management interfaces used in Internet browsers (i.e. IE, Chrome, and Firefox), and local users can use the cloud servers directly through the management user interface provided by the middleware layers. Further research is required for integration of PaaS, IaaS, SaaS, DaaS, and MaaS to achieve the bidirectional enablement between geoscience and cloud computing (Yang et al., 2011a).

2.2 EO Sensor Web and Sensor Web Enablement (SWE)

Understanding the Earth system, its climate and weather, and natural environment and human-induced disasters, is crucial to human health, safety and sustainable development. One of the major achievements in Earth observation is the development and implementation of the EO Sensor Web (Zhang et al., 2012). In this section, the concept of EO Sensor Web and relevant technologies were reviewed.

2.2.1 Introduction of EO Sensor Web

The concept of EO Sensor Web has emerged due the fast development of Earth sensing, communication, and information technologies. EO Sensor Web is proposed to meet the requirements of geoscientists for the timely and pertinent geospatial data and information, which are used for supporting applications in the societal benefit area of EO. The EO data is most often referring to satellite imagery or satellite remote sensing, which utilized upon atmosphere, land and ocean. The obtained data of Earth Observation, which is named EO data, is widely used in the fields of scientific research such as climate, weather, environment, ecosystem, biodiversity, hydrology and natural disaster migration, forecasting or reduction. The capabilities of Sensor Web include that it can perform intelligent autonomous operations in uncertain environments, respond to changing environmental conditions, and carry out automated diagnosis and recovery (Delin et al., 2001).

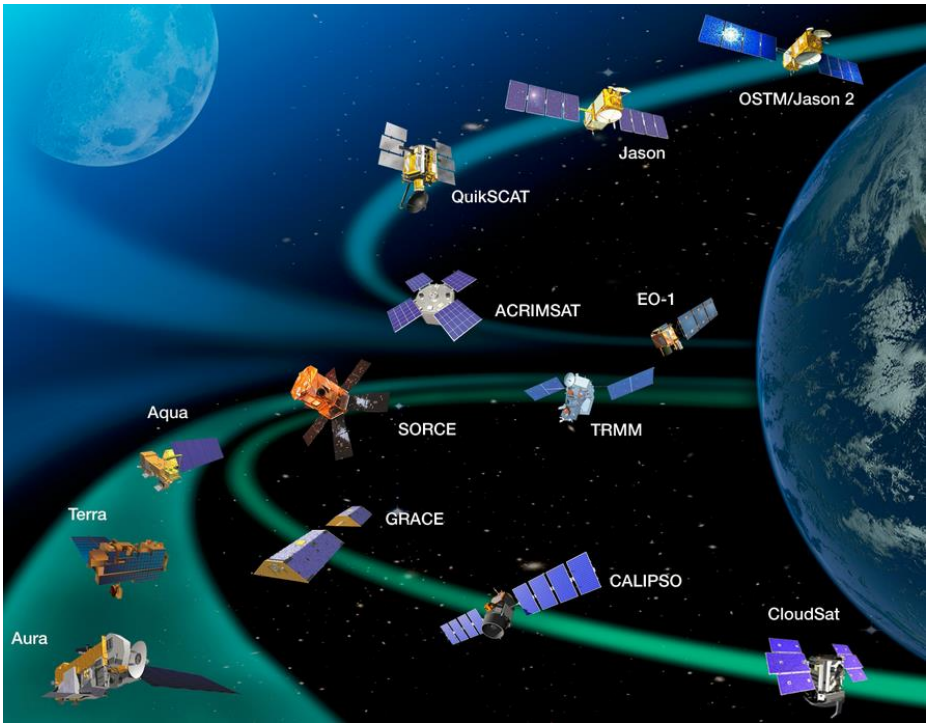


Figure 2.2. The A-Train satellite constellation (http://atrain.nasa.gov/historical_graphics.php)

In recent years, the EO Sensor Web has been an active research topic in geoscience. A lot of international EO programs and space agencies have ongoing research projects and application that contributes to EO Sensor Web. The National Aeronautics and Space Administration (NASA) of the United States sponsored more than 30 research projects in 2005 to evolve and develop the sensor web technology through its Advanced Information System Technology (AIST) program (Di et al., 2010). Furthermore, Dozens of EO applications have been developed based on those research projects in various social benefit areas, such as ecosystem dynamics, land-use change, disaster monitoring, disaster assessment, sustainability and agricultural production, climate change, biodiversity, and public health (NASA/ESTO, 2008).

In recent years, there have been several initiatives at various administrative levels (e.g., national, regional, and international) for organizing the geospatial service resources in the field of EO Sensor Web. As one of the most famous worldwide initiatives in this direction, The Global Earth Observation System of Systems

(GEOSS) has been developed for years with the goal to build a network of EO sensors by integrating a wide range of heterogeneous EO platforms. By sharing all international sensor resources, the GEOSS can provide a real-time image as the snapshot of the whole planet. the Global Monitoring for Environment and Security (GMES) initiative, Infrastructure for Spatial Information in the European Community (INSPIRE), and Shared Environmental Information System (SEIS) can be seen as part of the European contribution to GEOSS (Chen et al., 2013). Mandl et al. (2008) presents an ambitious space sensor web for disaster management with the objective of facilitating the United States contribution to the GEOSS. Another famous initiative of EO system is the Intelligent Sensor web for Integrated Earth Sensing (ISIES) undertaken by a team of Canadian industry. ISIES integrates the in-situ sensor web data with remote sensing data and vegetation models automatically to provide maps of leaf area index, soil moisture and biomass, as well as improved predictions of crop and rangeland yield.

2.2.2 OGC Sensor Web Enablement (SWE) Framework

The SWE initiative of OGC is a collection of various standard specifications which aims at increasing the interoperable usage of heterogeneous sensors or sensor system by enabling their discovery, planning, interacting and event processing (Broring et al., 2011). Utilizing the Web2.0 technology and Sensor Web Enablement (SWE) Web Service standards to enable access to Earth Observation (EO) data is an emerging mega-trend which will dramatically decrease the cost of the producing customized science by an order of magnitude (Daniel Mandl et al., 2010). OGC SWE has defined a suite of open standards for the sensor web (Botts et al. 2007), including specifications for data models, data encodings, and Web Service interfaces. Although these OGC SWE standards are not as popular as the WWW standards, the development and adaption of sensor web open standards is one of the necessary steps to realize the sensor web vision. OGC SWE components include models and XML schemas (SWE Common, O&M 2.0, SensorML 2.0, EML) and Web Service

interfaces ((SOS 2.0, SPS 2.0, SES, SIR, SOR), which are described as follows:

SWE components include models and XML schemas (SWE Common, O&M 2.0, SensorML 2.0, EML) and Web Service interfaces ((SOS 2.0, SPS 2.0, SES, SIR, SOR), which are briefly described below:

- SWE Common Data Model: a common data model defining some common and basic data types used throughout the SWE framework. The SWE Common Data Model is an encoding standard for exchanging sensing data between sensor nodes in the OGC SWE framework. These models enable SWE applications and servers to encode and transmit sensor datasets in a semantically enabled and self-describing way.

  The model enables the user and/or the server to organize, encode and transfer sensor data-sets through self-describing and semantic activation (Robin, 2011).

- SensorML, Sensor Model Language: SensorML specifies a model to encode the description of all kinds of sensors or sensor platforms, as well as related processes using XML Schema. It provides a functional description of detectors, actuators, filters, operators, and other sensor systems, which are treated as instances of process models (Botts, 2007).

- O&M, Observations & Measurements: O&M standard is a domain independent, XML-based conceptual representation of both spatiotemporal measurement data. This standard defines some terms as well as their relations relevant to observation and measurement (Cox, 2011).

- EML, Event Patterns Markup Language: EML is a new specification of SWE 2.0 which is developed for representing and processing of complex events, event flows and event cloud (Everding and Echterhoff, 2008).

- SOS, Sensor Observation Service: The SOS offers a series of web-based interfaces to facilitate the users to discover and access observations and sensor metadata generated by Sensor Web (Na and Priest, 2007). SOS enables client applications to discover various types of observation through standardized operations and filters and retrieve the observations in a common format specified by the standard.

- SPS, Sensor Planning Service: SPS can be used to define tasks for the collection of observations and the scheduling of requests. SPS performs as a middleware to support the complex interactions between users and sensors (Simonis and Echterhoff, 2011).

- SES, Sensor Event Service: SES provides a series of APIs for managing event subscription and message sending. SES evolves from the Sensor Alert Service and offers more powerful functionalities to process event-based observation tasks. Through SES, the sensing platform can automatically detect the changing of geospatial conditions, trigger predefined mechanism to process observed data, and sends messages to users based on the specifications of the user-specified subscription (Na and Priest, 2007).

- WNS, Web Notification Service, a Web Service which enables the asynchronous interchange of message between a client and one or more services (e.g., SES and SPS) (Simonis and Echterhoff, 2003).

2.2.3 Common Data Model (CDM) and NetCDF Markup Language (NcML)

The NETwork Common Data Form (NetCDF, Domenico, 2011) has become one of the most widely-used file formats in geoscience and environmental science applications. NetCDF is commonly used by integrating the Climate and Forecast (CF) metadata conventions (Domenico and Nativi, 2012) which provide semantic meaning

and georeferencing information. Figure 2.3 illustrates the UML diagram of the NetCDF information model. The Network Common Data Form (NetCDF) has been widely applied all over the world as a efficient data model for storing and sharing scientific data, especially geospatial datasets with multiple dimensions (Nativi et al., 2005). Some other file formats like the Hierarchical Data Format (HDF) and GRIdded Binary (GRIB) are also in wide use in geoscience. The Unidata Common Data Model (CDM) offers an abstraction layer to harmonize and unifies these formats. The CDM is implemented as Java library that can read all of these formats (and more) using a common Application Programming Interface (API).

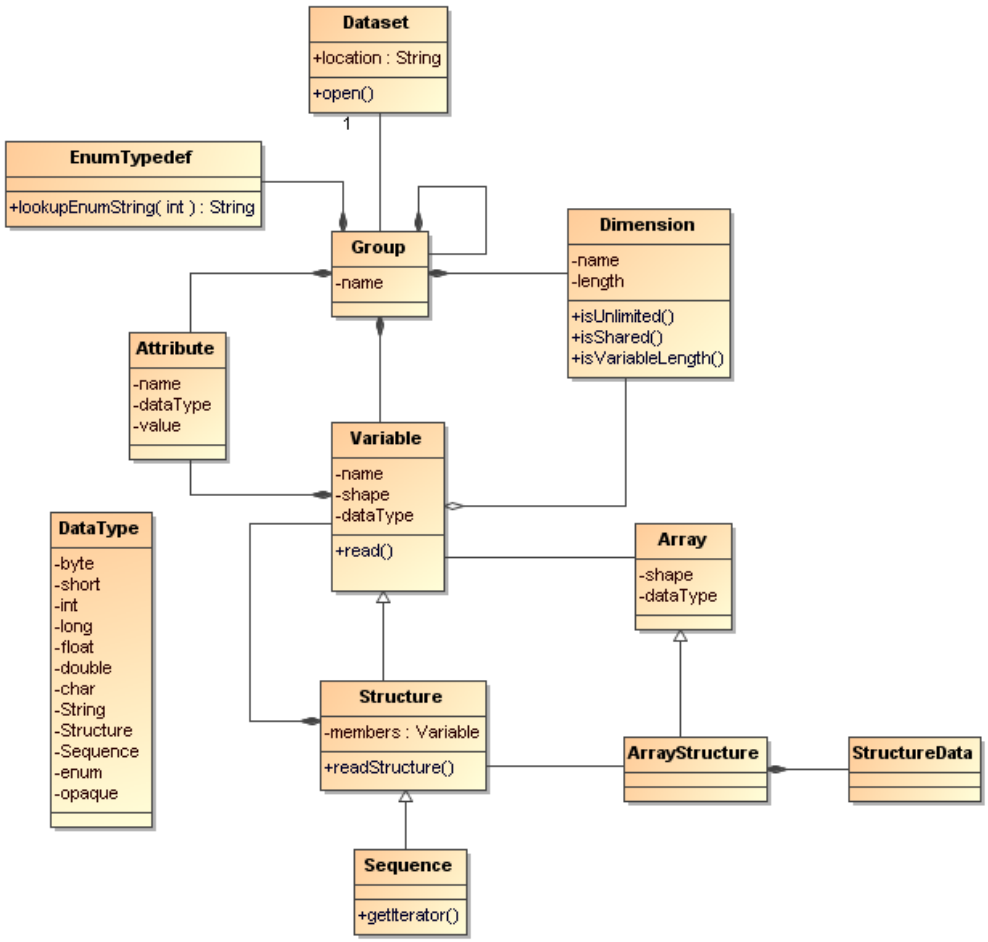


Figure 2.3. The UML class diagram of NetCDF information model (http://www.unidata.ucar.edu/software/thredds/current/netcdf-java/CDM)

The NetCDF Markup Language (NcML) was developed as a natural augmentation of the NetCDF with extensions encapsulating descriptions of the structure and content of NetCDF objects in an XML (Nativi et al., 2005). An NcML

document is an XML document describing the content and structure of the data stored in a NetCDF file and represents a generic NetCDF dataset. NcML describes the metadata of the NetCDF data and does not encode the data. NcML enables users to append additional attributes (e.g. CF convention attributes) to it instead of rewriting the original file. Furthermore, NetCDF dataset can be used to create the "virtual NetCDF" files that presenting existing NetCDF files or other gridded data files as a single dataset. By using NcML to encode part of dataset semantics, generate virtual datasets, and introduce GIS community semantics, the files do not required to be rewritten or the program to be remodified (Nativi et al., 2005). The purpose of NcML is to define and redefine NetCDF file. The NcML has the function as follows:

- Metadata to be added, deleted, and changed.
- Variables to be renamed, added, deleted, and restructured.
- Aggregated data from multiple CDM files.

The aggregation function of the NcML is useful for time series data combinations. Multiple time series NetCDF data can be aggregated into a single, logical dataset with several types of aggregation including *Union*, *JoinExisting*, and *JoinNew*. Figure 2.4 illustrates a demo of NcML aggregation. The attribute 'dimName' of 'aggregation' indicate the dimension on which the aggregation bases, and the attribute 'type' indicates the aggregation type. When the data server (THREDDS in this study) is started, the server will read this configuration file and aggregate relevant files into a single virtual dataset which is updated every 30 minutes.

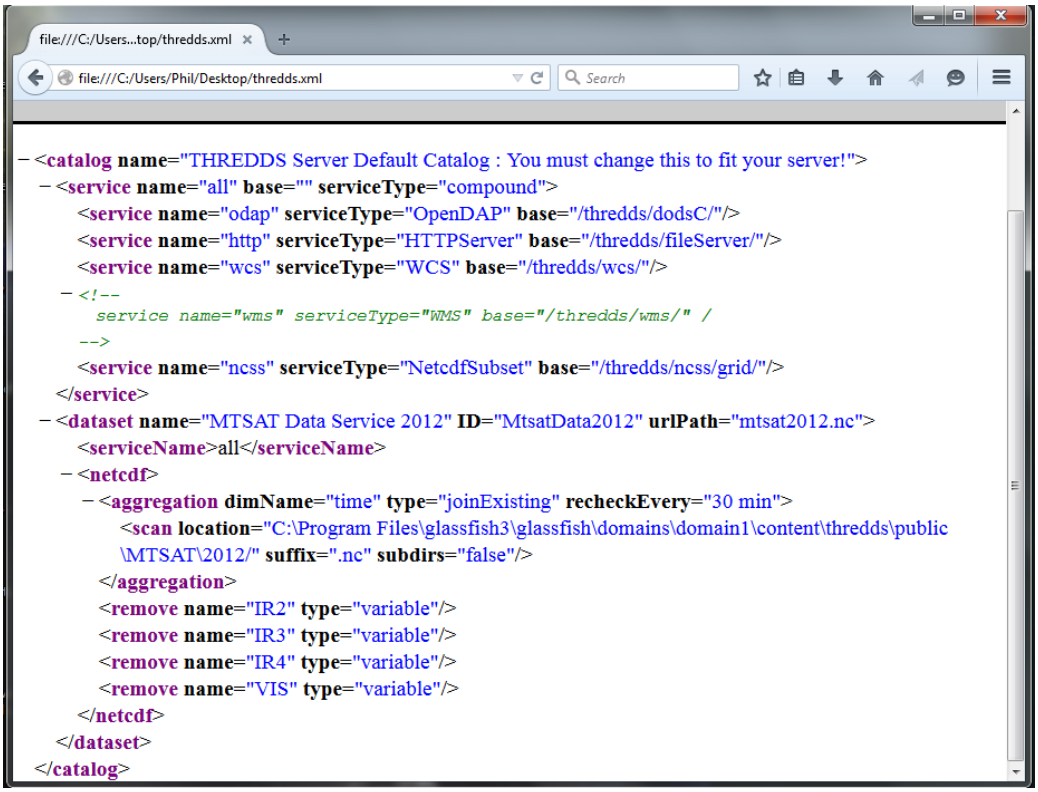


Figure 2.4. The aggregation function of the NcML

THREDDS (Thematic Real-time Environmental Distributed Data Services) catalogs provide information about which datasets are available via which services/protocols (Domenico et al., 2002). The three main client/server (as opposed to full-file transfer with FTP or GridFTP) protocols for remote data access in use in the community are OPeNDAP (Open-source Project for a Network Data Access Protocol), ADDE (Abstract Data Distribution Environment), and NetCDF access via HTTP protocol. In many cases, the EO data service systems can be augmented by integrating with THREDDS catalog services which has inventory list and metadata access. Through the THREDDS catalog services, client applications can understand the service capabilities of the EO data server, e.g. all available dataset on the server, via the THREDDS interface. In this way, the client applications can retrieve required subset of raw EO data via ADDE, OPeNDAP, or NetCDF/HTTP protocols.

The THREDDS Data Server also provides a Web Service called NetCDF Subset Service (NCSS) for retrieving the subset of CDM scientific datasets. The subset of a data is specified using earth coordinates, e.g. latitude and longitude, bounding boxes,

and date ranges, rather than index ranges referring to the underlying data arrays. The NcML and NCSS were intensively applied in this study for grid-based data transmission. Compared with traditional raster-based data service, e.g. WCS, the NCSS is more simple and effective because it is developed directly based on the NetCDF-java library, and can seamlessly integrated with NcML for data service.

2.2.4 New generation of EO Sensor Web

With the continuous development of EO Sensor Web theories and technologies, the traditional Sensor Web applications and service patterns for observing the Earth are not able to provide enough powers to handle the changeable and complex EO tasks, which includes the amount of multiscale observation dataset, distributed heterogeneous sensing systems, and dynamic and complex controlled network environments (Chen et al., 2014). It is extremely urgent to research and develop the new generation of EO Sensor Web system architectures and service patterns. The major challenge is to develop a robust approach to fast and effectively discover and integrate these various, heterogeneous sensors within an application of disaster emergency in a simultaneous manner (Hu et al., 2011). Moreover, how to better understand the capabilities of an EO sensing platform, and then process their observation data with disregard to the heterogeneous data formats is another grant challenge. Finally, how to achieve better performance in serving EO data on-demand in a virtual sensor web environment and how can the relevant emergency services including sensor planning, data accessing and relevant processing, etc., to be on-demand timely chained and composed have become the bottleneck of geospatial Sensor Web.

One of the research directions for EO Sensor Web is to develop a scalable and cooperative work mechanism, as well as the integration of multiple heterogeneous EO platforms called Integrated EO Sensor Web system (IEOSWS). First, the IEOSWS can realize dynamic assignment of ground-based, airborne, and aerospace observational resources based on user's requirement. Second, the IEOSWS can

effectively discover and utilize observational resources, fulfil increasingly various observation requirements, and enable the customizable, transparent, and efficient application of EO resources. Thus, the IEOSWS can truly realize the dynamic resource management, intelligent event perception, information fusion, on-demand observation, data assimilation, and smart service on multiple heterogeneous sensing platforms.

On the other hand, the EO processing challenges drive the evolution of distributed computing paradigms from cluster computing, grid computing, to cloud computing, which can provide more powerful and scalable computing capabilities to enable the computability of geoscience applications (Yang et al., 2011a). As introduced in section 2.1, the cloud computing offers computing resources (processing capability, storage, and network) in a scalable, dynamic, and virtualized way in the form of web services. It is believed that this new type of computing resource distribution and organization paradigm can supply more benefits for new generation of EO Sensor Web system, such as improvement of utilization ratio, intelligent sensing resource allocation, realization of green computing , energy saving and provision of a new sensor service mode (Chen et al., 2014). In addition, the performance of computing could be further improved and optimized through utilizing spatiotemporal patterns of phenomena, data, services, models, and computing resources.

Finally, Chen et al. (2009) and Chen et al. (2011) proposed the concept of virtualization sensor data service, which refers to the registry, deployment, discovery, planning, collaboration, combination, and fusion of EO resources under the Internet environment. The provision of creating highly virtualized sensing resources dynamically is one of the most important technologies utilized by cloud computing in EO Sensor Web system. The sensing resources commonly include the sensing platforms distributed in the air-space-earth, storage devices, networking resources, computing resources, and processing model resources. It is believed that the

virtualization service technologies of cloud computing is the development trend for the Sensor Web and can dramatically enhance the sensing resources utilization and the performance of complicated, multiple observation tasks,

2.3 Technologies for Processing Big Geoscience Data

One significant feature of modern distributed systems is the large volume of data that they are expected to handle. Amazon and eBay routinely deal with enormous volumes of data from all over the world. The research engines Google, Yahoo, and Bing routinely crunch enormous amount of data to process user queries from all over the world. With geoscience analytic are becoming more and more computation- intensive and data-intensive, massively parallel data processing engines have become for generating prompt responses for processing big geoscience data. To crunch massive volumes of data, Google invented MapReduce (Yang et al., 2008) and Google File System (Dean et al., 2003).

2.3.1 MapReduce Framework and Distributed File System

MapReduce refers to a programming model and an associated implementation for processing and generating large volumes of datasets. MapReduce was designed to solve the problem of processing in excess of terabytes of data in a scalable way. The key idea of MapReduce originates from functional programming (Alexandrov et al., 2011), which is based two second-order methods: *Map* and *Reduce*. Each function has two input parameters, input data set (a set of key/value pairs) and a user function (user-defined first-order function). To design MapReduce-based algorithms, users must implement a *Map* method for processing the key/value pair and producing a set of intermediate key/value pairs, as well as a *Reduce* function for combining all intermediate values produced by Map functions based on intermediate keys.

GFS is a Distributed File System (DFS, Ghemawat et al., 2003), which is widely used in Web Services, and the leading examples include Amazon Simple Storage Service (S3) and Apache Hadoop Distributed File System (HDFS). GFS provides fault tolerance, while running on inexpensive commodity hardware and also serving large number of clients with high aggregate performance. Even though the GFS shares many similar goals with previous distributed file systems, the design has been driven by Google's unique workload and environment. Google had to rethink the file system to serve their "very large scale" applications, using inexpensive commodity hardware.

The Apache Hadoop is a commonly used open-source implementation of MapReduce and GFS, which has been widely used to create a cloud computing environment for large amounts of data storage and linearly scalable processing on clusters. Hadoop consists of two main components, Hadoop MapReduce and HDFS. HDFS is the answer of storage industry for unstructured and huge amount of data which incurs huge amount of cost and fault tolerance. It is a fault tolerant file system designed to store data in a reliable manner even if failures like *NameNode*, *DataNode* and network occur. It works on a master slave architecture wherein a master server manages access to files and slave for storing user data via data nodes. An advantage of using HDFS is data awareness between the *TaskTrackers* and *JobTrackers*. Hadoop is also optimized for minimal network I/O by allocating the computation tasks as close as possible to the data. In general, the DataNodes of HDFS and TaskTrackers of MapReduce are placed on the same server that allows the map and reduce processes to run on the same physical site where the data is located. By using this approach, Hadoop gets ride of transferring the data over the network.

Hadoop has been widely used in the geospatial science research community. To enable Hadoop to process multi-dimensional and array-based geoscience data, Zhao et al. (2010) applied the text-based CDL file, which is transformed by the NetCDF data, in order to achieve the parallel processing of the NetCDF data with large file

size using MapReduce technologies. Duffy et al. (2012) utilized the Hadoop framework to analyze the meteorological dataset by transferring rater-based climate data into Sequence Files, which is widely used to store binary file in Hadoop software. Li et al. (2014) proposed a "Tile Cube", which is a Map-Reduce-enabled Spatial On-Line Analytical Processing (SOLAP), for aggregating large-scale of remote sensing data. The "Tile Cube" improves the scalability and throughput of satellite image aggregation through the implementation of Roll-Up/Drill-Across operations in the SOLAP environment, which makes the wide-range, long time-series, and multi-view analysis of remote sensing data can be processed in real-time manner. Malik et al. (2010) developed the Cassandra, which is a distributed storage system for managing large amounts of structured remote sensing data. Cassandra aims to run on top of an cluster of cheap commodity servers or desktops, which are possibly distributed across various data servers and these servers fail continuously. Cassandra can offer highly available and stable geospatial services, handle high I/O throughput while not sacrificing read efficiency. Rizvandi et al. (2011) gave an overview of the Prestack Kirchhoff Time Migration (PKTM) algorithm, which is one of the widely-used seismic imaging algorithms, and proposed an approach to deploy and run this algorithm on the MapReduce framework.

The MapReduce algorithms efficiently harness the built-in parallelism exhibited by many large-scale or data-intensive problems. MapReduce supplies a way to build a system that increases in performance linearly with the number of physical machines added. Following a divide-and-conquer approach by splitting the data located on a distributed filesystem so that the servers (or rather CPUs, or more modern "cores") available can access these chunks of data and process them as fast as they can. However, it is important to realize that although MapReduce can be directly used for a large class of problems that exhibit embarrassingly parallel feature, many algorithms cannot be easily expressed as a single MapReduce job. Complex algorithms have to be decomposed into a sequence of jobs, and data routing has to be orchestrated so that the output of one job becomes the input to another job.

2.3.2 NoSQL Database and HBase

More than thirty years, the relational databases management systems (RDBMS) (e.g. SQL Server, Oracle, and DB2) have become the prior solution for data storage in many geoscience projects and applications. RDBMSs have been widely utilized for the storage and management of a variety of geoscientific data for decades and currently the mainstream technology of GIS data storage is still using RDBMS. The traditional way for managing geospatial dataset is to store the metadata in a relational database while store the actual geospatial data in local file systems. The geospatial data can be retrieved by using the file location found by spatial query over the database. Furthermore, RDBMS can effectively support constructing GIS workflows because of the transaction and locking features and supply reliable backend for enterprise GIS systems. Geospatial data usually have a fixed schema and they are not applied independently in most cases. A combination of two or more heterogeneous datasets and exchanging data through geospatial operations is necessary in most GIS workflows.

Although the evolution of traditional RDBMS has achieved better scalability by using parallelization to deal with geographically dispersed data, these systems may not be able to offer the required effectiveness under some situations, especially with physical environment provided by cloud computing facilities consisting of relatively low-end hardware (Chen et al., 2014). One of these situations is when there is a need to publish a standard geospatial resource that deals with big geospatial data. At the moment a few terabytes of data can be considered as big data. Also in some situations (such as disaster management) when there is a need to use various geospatial data from different sources for fast decision making, the performance of the system (in terms of response time) is very important. Furthermore, the cloud computing provides an environment with high-concurrent and large-scale data accessing as well as massive data processing, which makes RDBMS inadequate to meet continuously increasing demands on big data storage and query.

One potential solution to these problems of traditional RDBMS is the NoSQL databases technologies (Stonebraker, 2010). NoSQL commonly stands for the "non SQL" or non-relational database management system. Distinct from traditional RDBMS (Relational Database Management System), NoSQL-based database does not support query based on the relationship of entities; rather, the rows in NoSQL database can only be retrieved through the row-key, which is a unique and used to identify the records in NoSQL database (Zhang et al, 2014). In recent years, the NoSQL database technology has been intensively researched and applied because it is highly scalable to achieve the grant challenges proposed by recent development of Web-based applications, such as concurrent read and write on the database, and access to a large volumes of data. The NoSQL databases are more palpable in advantage than the traditional RDBMS for processing a large mass of data. Furthermore, NoSQL databases are more suitable for storing and managing semi-structured or unstructured data such as text, images, and videos.

In addition, traditional RDBMS applies tables, row-keys, and relationships of entities for data storage, Structured Query Language (SQL) for performing all sorts of functions with data, and relational algebra and relational calculus as their theoretical foundation for managing data. These features make traditional RDBMS suitable for managing structured data. However, this approach faces challenges when dealing with unstructured data. Recent trends begin to focus on how to apply unstructured or semi-structured data model to represent geospatial dataset. In the geoscience domain, there are three examples of semi-structured and unstructured data which are geospatial data generated by Location Based Services (LBS), observation and measurements from sensor webs, and social networks. Furthermore, there are also some approaches that produce large volumes of geospatial data, e.g. the remote sensing and laser scanning. The traditional RDBMSs provide very expensive cost for storing and managing such semi-structured data in some situations when high scalability and availability are needed, even if it has a fixed schema.

HBase, which is the abbreviation of Hadoop Database, is a non-relational, open source, and distributed database systems. The HBase runs on top of the HDFS. Developed based on Google's BigTable (Chang et al., 2006) framework, HBase supplies high reliability and scalability storage abilities by managing data located on a cluster of commodity computer nodes with the capability of automatic recovery from node failure. All data stored in HBase is organized through tables. Figure 2.5 illustrates the architecture of HBase system.

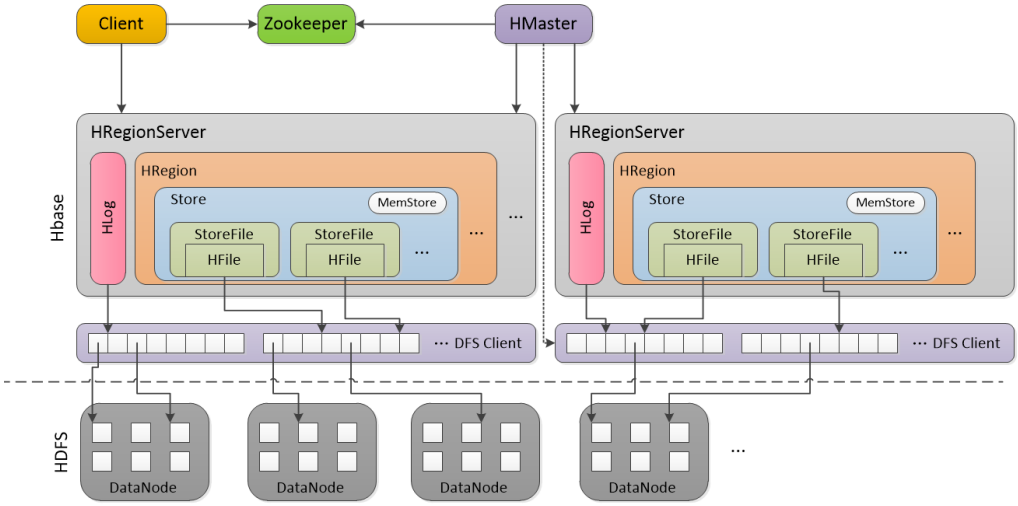


Figure 2.5. The overview of HBase architecture

As shown in Figure 2.5, the data in a table is stored in the form of *Row*, which is identified through a unique *row-key*. The row-key is similar to traditional RDBMS, yet the difference is that each row in HBase table is able to contain an arbitrary number of *Columns*. Furthermore, the HBase table is column oriented, which means that the tables are actually stored by column on file systems. Each column family is stored in one storage unit called *HStore*. This characteristic indicates that it is efficient to retrieve a column of a table rather than a row in HBase system. Finally, a table in HBase is divided into several *Regions* logically, which are then physically stored and managed by different *RegionServer*s. Each RegionServer locates on a single computer in the HBase cluster. A Region can be merged or re-divided when the number of rows in that Region reaches to a user-specified threshold.

Several studies have been conducted to investigate the capabilities of using HBase to store and manage large volumes of geospatial data. Liu et al. (2013) developed a way to store large volumes of raster-based remote sensing data in HBase by proposing tow tables, which are “HRasterTable” and “HRasterDataTable”. Chen et al. (2014) proposes an efficient mechanism for searching and managing satellite imageries stored in HBase. Li et al. (2015) proposed a decomposition mechanism to manage multidimensional geospatial data in a cloud computing environment. Based on Li, the remote sensing data can be divided into a set of tiles indexed by the row number, column number, and levels.

2.3.3 REST Architecture and RESTful Web Services

REST (Representational State Transfer) is an alternative software architectural development style for developing distributed network systems. REST was first proposed by Rod Thomas Melding (Fielding, 2000) in 2000 in his PhD thesis as a set of constraints used for the communication between clients and servers. REST offers software architects for engineers to develop concrete distributed systems and applications. In the REST architecture, all interactions with the services are actually stateless, and the information transferred between various service resources is the representations of these resources. All the architectures which are satisfied with the REST constraints are called RESTful architectures.

There are many different ways for a web application to follow the RESTful architecture. That is because for a REST-based application, only the Uniform Resource Indicator (URL) is mandatory. As the most widely used application-level protocol, HTTP has been one of the typical implementations of RESTful architecture (Mazzetti et al., 2009). Based on the HTTP protocol, four operations, GET, PUT, POST, and DELETE, were defined to represent the operations that can be executed on the target web resource. The CRUD pattern was applied to describe these basic operations. Based on CRUD pattern, the POST operation means creating a new resource, the GET operation means retrieving a resource, the PUT operation means

updating, and the DELETE operation means deleting a resource. A resource representation as well as its links corresponds to a snapshot of states about the web applications. Each interaction can trigger an updated application state. However For security consideration, the state of resources can only be modified in the presence of the PUT, POST or DELETE operations. In addition, the interactions between client and server are idempotent, which means that duplicate user-specified requests of creating, updating, or deleting a resource executed only once (Muracevic et al., 2009). The CRUD pattern has been proved to be effective in computer systems. RESTful is less complicated than SOAP in the concept level. Considering that the only web protocol needed by RESTful service is the HTTP, the RESTful-based Web Service is able to go through firewalls of operating systems without special security configuration. Furthermore, RESTful-based web service is easier to develop because the HTTP protocol is much easier than the protocol used by SOAP.

Geo-scientists are not profound IT experts and they are not able to rely on full-time technical staff often to maintain a complex IT infrastructure. On the contrary, they should concentrate on their own research topics. Typically, geoscientists usually require to access and deploy geospatial datasets in an easy way. So the emergence of REST technology is particularly attractive for geoscientists by avoiding the trouble of bored programming technologies and maintaining complex networking. Recently, the development and designing of RESTful Web Services and resource-oriented architectures are gaining much attention. Most of research works have been conducted on transferring data models defined by OGC specifications to RESTful-based web services in the SDI context (Mazzetti et al., 2009; Finney and Watts, 2011; Foerster et al., 2011; Janowicz et al., 2012).

A RESTful-based workflow interoperation method was discussed by Chen et al. (2009) to integrate heterogeneous geospatial workflow instances into an interoperable system and a study case about simulating nitrogen dioxide (NO_2) from a volcanic eruption was conducted to evaluate the efficiency of the proposed method.

Muracevic et al. (2009) proposed how to expose geospatial data and processing services as web services over the Internet based on REST. Muracevic gave a RESTful implementation of RESTful-based geospatial Web Services, which provides open and simple access to geospatial data over the Internet using standard web protocols. Granell et al., (2012) have conducted some proof-of-concept experiment of using RESTful interfaces for building geospatial processing services and evaluate the feasibility. The RESTful architecture was applied in this research to build geospatial data services. The detail implementation was given in section 3.2.

2.4 Geospatial Service Composition in Cloud

It is necessary to research how to encode geospatial data and transfer them between the servers and clients when exposing geospatial models in the form of web services. The Web Processing Service (WPS, Schut, 2007) standard proposed by OGC has become the de facto standard which is used to build web-based geospatial processing. A large number of GIS libraries and software are based on the WPS standard (Brauner et al., 2009). In order to improve the compatibility, WPS only defines the rules of exchanging information between WPS servers and WPS clients. Furthermore, WPS allows exchanging various types of data (i.e. bounding box, literal, complex, sub-process) which are widely used in geospatial analysis and processing (Schaeffer, 2008). The OGC WPS standard standardizes the implementation of web resources through a specific way. For example, a WPS-based web service must accept receiving input data and apply them to execute a geospatial process, and output the final result in the form of XML-encoded document which is sent back to the service consumer. Michaelis and Ames (2009) presented an evaluation and implementation of WPS in order to test the algorithms for raster manipulation and watershed delineation.

2.4.1 OGC WPS-Compliant Web Service composition

There have been several popular WPS server products including 52-North WPS, PyWPS, and Zoo. The 52-North is an organization aims at offering open-source software based on many OGC standards. The 52-North WPS component is developed using the Java programming language which offers various types of service interfaces such as raw data, HTTP, and SOAP. 52-North WPS enables users to connect to ArcGIS and GRASS GIS component for geospatial processes. 52-North WPS offers a component called WPS4R through which can be used to expose the R scripts as WPS service instances. PyWPS is an open source software written in Python aiming at implementing the geospatial processes provided by GRASS GIS system as Web Services. PyWPS provides a convenient Apache module called '*Mod_python*'. Through applying Mod_python to integrate a Python interpreter into the Apache server directly, the PyWPS guarantees a fifty times faster request processing capability than other WPS implementation (Fenoy et al., 2012). Finally, the Zoo project (Fenoy et al., 2012) is an open source WPS framework written in C. Zoo support online processing of both vector-based and raster-based dataset, and creating and chaining existing WPS processes. Contrary to other similar WPS implementations, Zoo enables model developers to apply various programming languages to develop and deploy new WPS instances. The Zoo-Kernel component embedded in Zoo can be used to effectively communicate to GRASS GIS tools through and generate a full-featured WPS service.

Many attempts have been proposed within the geoscience and environmental science domains to apply OGC WPS services in service-based geospatial workflows, which demonstrate the flexibility and reliability of OGC WPS interfaces in geospatial workflows. Chen (2010) proposed a general Sensor Web data service framework for Geo-processing Workflow (GPW) which integrates OGC Sensor Web Enablement (SWE) and OWS to achieve interoperability, flexibility, and reusability. Yue et al. (2010) utilized Ontology Web Language for Services (OWL-S) as the

underlying semantic technology for the semantic description of geospatial services. Cannata et al (2012) use WPS for shallow landslide assessment by linking two landslide models in a real-time application. Dubois et al. (2013) present the e-Habitat application, which implemented as WPS and combined with climate change scenarios to allow evaluating future conditions in ecosystems. Castranova et al. (2013) proposed an approach to integrate WPS implementation and the OpenMI standard for service oriented environmental modeling. Based on Castranova, the OGC WPS standard performs as a mechanism for exposing geospatial models as Web Services while the OpenMI standard provides the interfaces for consuming them. Thiebes et al. (2013) provide a model for landslide analysis as WPS, aiming to integrate this model in landslide early warning systems. Nativi et al. (2013) described how geospatial models can be deployed the cloud and applied WPS specification to improve access to available models and interoperability between models and modelers.

However, the current OGC OWS-based service composition technologies are mostly based on a centralized manner, which means both the data flow and control flow are controlled by a centralized workflow engine. The advantage of the centralized manner is that it is easy to implement and manage. However, the serious problem is that this type of service architecture is susceptive to so-called single point of failure. Furthermore, the centralized architecture has a bottleneck and it is hard to extend for the presence of high concurrency accessing number.

2.4.2 The Group on Earth Observation (GEO) Model Web

Some researchers have tried to expose geoscience and environmental models as Web Services. However, the interdependencies between the complex geospatial processes further complicated the integration and orchestration of integrating geospatial processing services. This problem is compounded by the limited access that decision makers have to the earth models, forecasting products, and related professional knowledges that do exist (Nativi et al., 2013). To solve these problems,

Geller and Turner (2007) introduced a concept of Model Web as an open-ended system of interoperable computer models and datasets based on the SOA architecture. The Model Web is a generic concept for increasing accessibility to models and their output, and to facilitate greater model-model interaction, resulting in a chain of interacting models, databases, and websites. The Group on EO Model Web initiative applies the web technologies to expose geospatial models as standards-compliant Web Service aims at increasing model sharing and access.

There are several applications in the climate dynamic community based on the Model Web architectures (e.g., the METAFOR project, Nativi et al., 2013). Furthermore, based on the concept of Model Web, Nativi et al. (2013) proposed the GEO Model Web initiative to increase the environmental model accessibility and sharing by defining model web conceptual framework, resource data model, and metadata framework. The GEO Model Web has now been widely used in geoscience community as a useful tool to build geoscience models, combining individual components in complex workflows (Bastin et al., 2013). However, chaining and integrating existing and independent models is still challenging due to the complex dependencies, e.g. different platform, heterogeneous data structure, and interfaces (Nativi et al., 2013).

2.4.3 Model as a Service (MaaS)

The geospatial models are inherently distributed, which means the model descriptions, and related resources can be deployed, in principle, anywhere on the Internet. Current development of cloud computing technologies enables service consumers to rent required cloud computing resources, opening new doors for developing web-based geoscience applications and achieve complex geoscience tasks which are always dynamically-scaling and computing-intensive (Gerard et al., 2013). The concept of model exposed in the form of Web Services, called “Model as a Service” (MaaS), has been promoted for several years and intensively studies (Geller and Turner, 2007; Geller and Melton, 2008; Roman et al., 2009; Yang, Xu,

and Nebert, 2013; Yang et al., 2014).

However, more research works are needed in order to investigate the way to effectively leverage these methods for specific applications within Model Web, for example, how to optimize data transmission and operations between traditional desktop environments and remote servers on the web. Several attempts have been made to integrate OGC services into service-based geospatial workflows within the geospatial and environmental domains (Granell et al., 2010; Goodall et al., 2011; De Jesus et al., 2012; Mullerm et al., 2013; Wang et al., 2013). The flexibility and reliability of using OGC WPS interface services in operational geospatial workflows can be enhanced. Cannata et al. (2012) used WPS for landslide assessment by linking landslide models in a real-time application. Thiebes et al. (2013) developed a model for landslide analysis based on WPS, and also integration of this model in landslide early warning systems. Dubois et al. (2013) presented the e-Habitat application, which made use of WPS and climatic models to give prediction in ecosystem environment. Yue et al. (2009) utilized OWL-S (Semantic Markup for Web Services) as the underlying semantic Web Service technology for the semantic description of OWSs. Castranova et al. (2013) proposed a WPS implementation framework for disseminating models as Web Services and adopting OpenMI standards as service oriented environmental modeling. Nativi et al. (2013) described how models can be set up and integrated in the cloud computing environment in order to improve interoperability between models.

2.4.4 Geospatial Cyberinfrastructure (GCI)

GCI was first proposed by Yang et al. (2010) as the cyberinfrastructure that utilizes geospatial information technologies to transform how research, education, and development are conducted within and across science domains, such as the environmental, climate, and geospatial sciences. Based on Yang, a Cyberinfrastructure (CI) is a combination of data storage systems, computing platforms, computational services, network protocols, and visualization

environments that integrates people, data and information, and computational resources together to perform science or other data-rich applications in this information-driven world.

GCI is based on recent advancements in GIS, IT, computer networks, sensor web, and cloud computing technologies. GCI offers an integrated architecture based on existing SDI to share geospatial data resources, computing resources, model resources, and knowledge in targeted domains, such as hydrology, social, and ecology sciences. To present of GCI, Yang et al. (2010) applied a GCI framework cube based on five aspects: enabling technologies, logical frameworks, desired future research, geospatial functions, and domain applications. GCI has been widely adopted in environmental projects (Pezzoli, Marciano, & Robertus, 2006; Rich et al., 2005; Kido et al., 2008; Mahinthakumar et al., 2006; Sucaet et al., 2008; Keating et al., 2009).

Besides the GCI, Wang et al. (2013) proposed the concept of Cyberinfrastructure-based GIS (CyberGIS) as a fundamentally new GIS modality comprising a seamless integration of CI, GIS, and geospatial analysis and modeling capabilities. Based on Wang, the CyberGIS has the following six core characteristics: open and distributed, high-performance and scalable, service-oriented, collaborative, community-driven, and user-centric.

2.4.5 Geoprocessing Web

As SOA has been developed to be one of the basic technologies for developing distributed and interoperable framework, more and more geospatial service resources and applications have been developed and deployed as web services over the Internet. However, it is a grand challenge in geoscience to develop an efficient geopolitical data processing method for extracting information and discovering knowledge over the web. The concept of Geoprocessing Web was first proposed by Zhao et al. (2012) aims to support the distributed, interoperable and collaborative processing of

geospatial data for information extraction and knowledge discovery. The Geoprocessing Web consists of crowd-sourcing capability, light-weight protocols, and the capability to process real-time geospatial data sources provided by sensors. Zhao introduced the Geoprocessing Web as a three-layer architecture which covers the conceptual, methodological, technical, and managerial aspects to facilitate distributed and collaborative geoprocessing over the Web. Interoperability, light-weight protocols, collaboration, distribution of resources, real-time and separation of concerns are six characteristics of Geoprocessing Web. Chen et al. (2010) also proposed a general Sensor Web data service framework for Geo-processing Workflow which integrates OGC Sensor Web Enablement (SWE) and WPS to achieve interoperability, flexibility, and reusability.

2.4.6 Open Model Interface (OpenMI)

The OpenMI was initiated within the HarmonIT project (Moore, 2001) in 2001, shortly after the adoption of the ambitious Water Framework Directive 2000/60/EC (WFD) by the European Parliament and Council. The primary goal of the OpenMI is to improve the interoperability between independently developed hydrologic models (Gregersen et al., 2007). The OpenMI is first proposed by Moore et al. (2005) as a component interface standard developed through the Water Framework Directive. The goal of OpenMI is to specify the communication standard for model interoperability, especially in hydrological science (Castronova et al., 2010). By designing a set of interoperation schemas and programming interfaces, OpenMI supplies a series of functions to facilitate the development of component-based modeling in a loosely-coupled way.

Based on Gregersen et al. (2005) and Goodall et al. (2008), the geoscience programs always consist of millions of code lines, and these programs are developed using various programming languages, e.g. C/C++, Java, IDL, Matlab, and Python. Model developers always have to constantly re-code popular programs in order to link them. In addition, many of these programs utilize a variety of techniques for

presentation of the result. The incompatibilities between programming languages and various visualization technologies make the geoscience models complicated to modify. The emergence of OpenMI addresses these problems. Instead of applying the common approaches of developing complex integrated models which always need constantly upgrading or customizing, OpenMI aims at combining existing geoscience models including both academic and commercial models or toolkits (Makropoulos et al., 2009) with slightly modified. By specifying some approaches of exchanging information in real-time and the way models can be linked to each other (Gregersen et al., 2007), an OpenMI-compliant model can be linked directly to other OpenMI-compliant models without applying any external configuration files. The OpenMI standards offer a way for integrating independently developed heterogeneous models. Furthermore, OpenMI provides a plug-and-play way for geospatial models to be linked, through which it is possible to replace one model in a workflow with another OpenMI-compliant model which that has more advanced functions and improved capability of simulating geospatial processes (Argent, 2005).

However, OpenMI was primarily designed for wrapping existing legacy model programs (Moore and Tindall, 2005), and it is hard for model developers to effectively develop new components in the process-level in a straight forward way. In addition, migrating local model to OpenMI interfaces is time-consuming since the technological details and knowledge needed is a grant challenge for many geoscience scientists (Granell et al., 2013).

2.4.7 The Infrastructure for Spatial Information in the European Community (INSPIRE)

Current research trends for discovery and access of large-scale distributed geospatial data and processing resources are being addressed by a European Union project called INSPIRE (INSPIRE, 2007). INSPIRE is designed for enhancing the interoperability of geospatial data and processing service deployed on geospatial infrastructures at the European member state level. The technical architecture of

INSPIRE consists of metadata of data and models, geospatial datasets, the Presentation layer, network services within a layered architecture that differentiates the presentation layer, the Service layer, and the Data Source layer. The implementation rules of INSPIRE directive propose a set of geospatial services classified in groups according to their functionality in order to embrace all required geospatial or GIS-based functionalities.

2.4.8 The Earth System Modeling Framework (ESMF)

The ESMF aims to provide shareable software component for climate, weather, and related projects by building flexible, scalable, and high-performance computing infrastructure that increase ease of use, interoperability, and performance portability in numerical weather products. ESMF is one of the most popular paradigms of modeling framework for combining earth model components and couplers of different Earth subsystem model through a common interface (Turuncoglu et al., 2013). The EMSF consists of a structure for combining components of earth system applications in a standardized approach, an infrastructure of automatic, high-performance utilities, and data structures which ensure consistent component behavior (Hill et al., 2004, 2006; Collins et al., 2005). In order to improve the interoperability of climate models, a series of standards were initiated by ESMF, which consists of a set of specified interfaces agreed by multi-agency consortium for developing ESMF. EMSF also supplies the ability to store and export component, grid, and field-level metadata as eXtensible Markup Language (XML) and other document.

2.5 P2P Technology and Its Application in Geoscience

With the fast development and widespread of GIS and RS theories, a large number of research works have been conducted to integrate how to integrate

wide-spread heterogeneous and autonomous GIS applications into a cooperative environment for constructing a new generation of GIS featuring in open architecture, distributed computing capability, cooperativity and extensibility. However, current web-based GIS or RS applications generally rely on centralized structure, where the geospatial data is stored on one single server. To get the required geospatial information, it is necessary to collect data and processing resources from multiple service nodes spreading over the network, composite these services as a workflow, and execute the workflow on a centralized controller. This approach has inherent drawbacks such as single points of failure, network congestion, and data inconsistency, etc. The inherent disadvantages of traditional GISs need to be solved for new applications on Internet or Web.

P2P networking is a paradigm where a set of user machines at the edge of the Internet communicates with one another to share resources without the help of any central authority (Sukumar, 2014). For a P2P network, the geographical boundaries become irrelevant, and the failure of any central authority promises spontaneous growth, as well as freedom form censorship. Peers include friends, collaborators and competitors, and the resource sharing has to be implemented through decentralized protocols. Scalability is an integral part of this concept, which means that no P2P system is worth looking at unless it scales to millions of machines around the globe. Regardless of the legal ramifications of ethical issues, P2P has led to users to a new form of freedom in collaborative resource sharing. One typical application of P2P network is the generation of genomic data about newly discovered proteins by collaborating hundreds of small laboratories all over the world. In addition, Facebook and Twitter also started using P2P technologies for content distribution,

Typically, a P2P system is composed of a large number of nodes with a host of sharable resources that encompass data/content, services, computing power, network bandwidth, etc. Each peer in a P2P network, which may take the roles of both a consumer and a provider of data and/or services, may join and depart the P2P

network at any time, resulting in a truly dynamic and ad-hoc environment. In addition, the distributed nature of such a design can eliminate the need for costly infrastructure by enabling direct communication among clients, and enable resource aggregation, thus provide promising opportunities for novel applications to be developed. By fitting Web services and P2P technologies into GIS, more flexibility and autonomy are added to GIS Web services, and the inherent limitations of centralized systems are alleviated to some degree.

Several studies have been conducted to apply P2P technologies to construct distributed GIS and RS systems. Guan et al. (2004) explored the techniques of enabling GIS services in a P2P environment to overcome the inherent shortcomings of current GISs and presented an implementation called BP-GService. Puppin et al. (2005) applied Globus package to develop a grid information service based on P2P network, which offers a fast propagation of information and has high scalability and reliability following the OGSA standard. Lee et al. (2006) proposed a method of applying P2P network to collaborative GIS environment, particularly targeting exploratory spatial data analysis for small-group brainstorming. Gianluigi et al. (2010) proposed a grid portal for solving geoscience problems using distributed knowledge discovery services by integrating workflow technologies with data mining resources and a portal framework in unique work environment called MOSÈ.

2.6 Concluding Summary

In this chapter, a literature review was conducted in five aspects. First, the recent development of the cloud computing technologies was reviewed as well as their application in processing Big Data. The conclusion is that the cloud computing technologies is the development trend for geoscience applications in the future. Second, the EO Sensor Web technologies and their applications were discussed, and a new generation of EO Sensor Web system was analyzed, indicating that make full

use of current advancement of cloud computing technologies to enhance the capability of traditional EO Sensor Web system is the next development direction. Then, some research hotpots of managing and analyzing geoscience Big Data, as well as web-based geospatial service composition technologies in the cloud computing environment were introduced. Finally, the P2P technologies were introduced as well as their application in geoscience.

Chapter 3: The GeoSPA Model

The GeoSPA was designed as a geospatial service hub through which the geospatial model developers and data producers can deploy their standard-based geospatial services in cloud. The GeoSPA supplies a set of algorithms for managing and discovering geospatial services, as well as orchestrating the service composition execution. To achieve these functionalities, three GeoSPA service models, which are EO data service model, processing service model, and computing service model, were defined and introduced as below:

- *EO Data Service Model* is proposed to facilitate the storage and management of large volumes of EO data by introducing a tile-based storage scheme and RESTful-based map service structure. Furthermore, The GeoSPA can be seamlessly integrated with HBase system to manage and process big geoscience data in the distributed file system, which can dramatically improve the processing efficiency.

- *Processing Service Model* enables the GeoSPA to expose geospatial models as standard-based geospatial web processing services, which can be combined as service chain to solve more complex geoscience problems. A knowledge model, which applies the notions of Belief, Process, Task and Plan to represent the geospatial processes as well as their composition, is designed for automatically discovering and compositing geospatial processing services.

- *Computing Service Model* is used to supply a distributed parallel execution environment for geospatial processes maintained by different GeoSPAs. Any registered geospatial process can migrate from one GeoSPA to another for execution. A Finite State Machine (FSM) is also deigned in this model for managing the states generated by The GeoSPA during the execution of geospatial workflows.

These three models make the GeoSPA as a one-stop solution for building SDI in cloud computing environment. To ensure different services can be interacted and cooperated with each other in a unified fashion, all of the GeoSPA service models are fully compliant to the OGC Web Service (OWS) specifications. The details of the internal structure and the three GeoSPA service models were introduced in next sections.

3.1 Internal Structure of GeoSPA

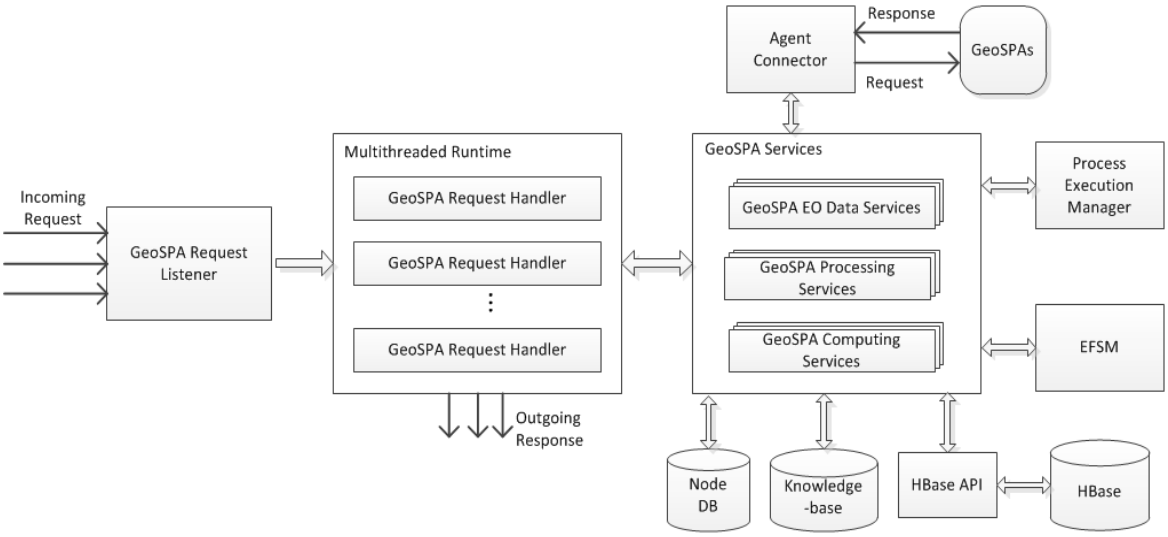


Figure 3.1. Internal structure of the GeoSPA

As shown in Figure 3.1, several functional components are embedded in the GeoSPA, which are introduced below. The *GeoSPA Request Handler* performs as the entry point of the GeoSPA for processing the incoming requests sent by service consumers or other service agents in a simultaneous manner. Each GeoSPA is equipped with a *knowledge-base*, through which the GeoSPA can determine which kind of service model needs to be used to handle the incoming request. Considering the geoscience problems are complex and several geospatial services always need to be cooperated and coordinated to achieve complex tasks, each GeoSPA was equipped with an *Agent Connector*, which is responsible for communicating with other GeoSPAs to exchange data. The *Process Execution Manager* (*PEM*) is responsible for managing the execution of geospatial processes in a multi-threaded manner, yet the *Extended Finite State Machine (EFSM)* is used to manage and handle the state transitions during the process execution lifecycle. Furthermore, each GeoSPA holds a *Node Database* which is responsible for storing all information needed for service agents communications and geospatial workflow execution. Finally, the *HBase API* component is utilized to interact with HBase system for storing and managing big geoscience data across a distributed file system.

3.2 GeoSPA EO Data Service Model

The GeoSPA EO data service model was proposed to facilitate the storage and management of large volumes of EO data. With the development of EO technology, the amount of grid-based EO data increases exponentially, reaching the scale of TB level and even PB level. This causes a grant challenge in storage and management of these data. The multi-resolution pyramid model based on the image block technology is an effective method for grid-based EO data organization. In order to store large

volumes of EO data rapidly and efficiently, this research proposed a tile-based scalable EO data storage scheme and introduced how to apply the NoSQL-based database called HBase to manage these data.

3.2.1 Tile-based Storage Scheme for EO Data

Tile-based data model uses a logical tile scheme that maps positions on the Earth to a two-dimensional surface and divides that surface into a series of regularly spaced blocks (Sample et al., 2010). As the essential component of the tile-based GIS, the logical tile-based storage scheme defines how to generate and locate the map tiles in multiple zoom levels, as well as the translation approach between the tile indexes and the geospatial coordinate system which is continuous. The tile-based storage scheme for EO data is typically based on map projection of two dimensions, and the indexing scheme enables a tile to be located using row and column numbers directly with discrete coordinates in the form of $(level, tile\_x, tile\_y)$, where $level$ is the layer number, $tile\_x$ is the the column number, and $tile\_y$ is the row number. Figure 3.2 illustrates the tile-based storage scheme used in this research for storing and managing large volumes of EO dataset.

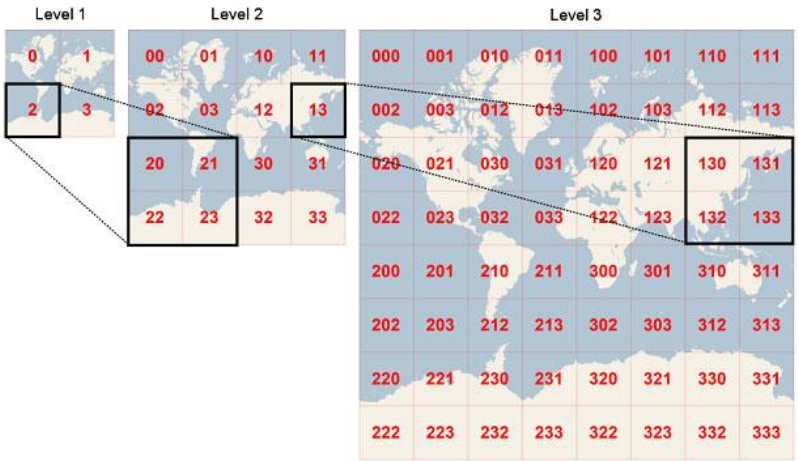


Figure 3.2. The tile-based storage scheme for EO data

As shown in Figure 3.2, the tile-based storage scheme used in this research is based on the Spherical Mercator projection, which is also applied by Microsoft Bing Maps, Google Maps, and Yahoo! Maps. Take the Bing Maps as an example, each tile can be divided into 4 sub-tiles to generate a zoom levels with higher resolution. The coordinate pairs are used to indicate the location of tiles in different zoom levels. The origin is located in the top-left. The tile size is defined as 256 × 256 by default. Assuming there are row rows and col columns after blocking, then there should be $(row \times col)$ tiles in this level.

Normally, EO data are array-based, which can be represented in five dimensions: space (latitude, longitude, and altitude), time and variable (band) (Li et al., 2015). The array-based data model is defined as Equation 1:

$$f(D) = DS(tile\_x, tile\_y, \text{level}, band, t) \quad (1)$$

Where D is the identifier of dataset, DS is the dataset, $tile\_x$ and $tile\_y$ represents the tile column number and tile row number respectively, $level$ is the level of map view, $band$ is a set of bands in this dataset, and t is the timestamp. Based on the raster-based data model defined by Equation 1, any EO data can be decomposed hierarchically as shown in Figure 3.3.

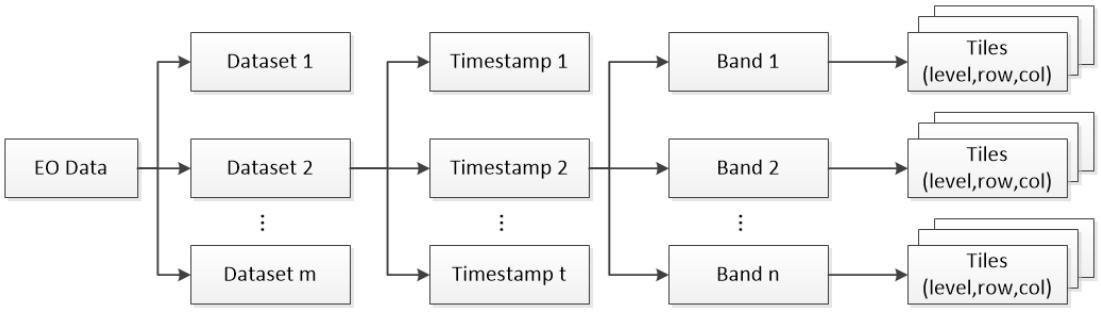


Figure 3.3. Hierarchical structure of the multi-dimensional EO data

Base on Figure 3.3, first, the EO data can be categorized as various dataset according to the sensing platform (e.g. Terra and Aqua), and then each dataset is tiered according to its observing date in the form of timestamp. For each timestamp, an original image can be tiered to some bands, and every band is a composition of a set of pyramid layers. Each layer in the pyramid model consists of multiple tiles (blocks), and each block is identified by a three-tuple in the form of $(level, tile\_x, tile\_y)$. Every type of EO data uploaded by the data producer is automatically processed and managed following the tile-based storage scheme by the GeoSPA and exposed as standard-based web services.

3.2.2 Storage of EO Data Tiles in HBase

HBase was utilized in this research to store the EO data tiles in a distributed file system. At the conceptual level, HBase stores data in *table* and each *row* in the table is identified and sorted through a unique *row-key*. The row-key is represented in the form of byte array, which means theoretically any data type can be used as a row-key. Each row has several *column*s, which are grouped into *column families*. All column family members have a common prefix. A column name can be expressed in the form of "$column\_family:label$", where $label$ indicates the specific column in a column family with the name of $column\_family$. The basic storage unit in HBase is *cell*, which is indexed through the intersection of the row and column coordinates and versioned by a *timestamp*. The content of the table cell is an uninterrupted array of bytes.

To store and manage EO data through HBase, the GeoSPA first reads the original EO data through the GDAL (Geospatial Data Abstraction Library) toolkit and then generates an EO dataset with specific timestamp according to the image

pyramid model introduced in section 3.2.1. And then the dataset is split into some different zoom layers based on the resolution and then each layer is split into same-scale tiles based on the tile-based storage scheme. The original image data is put as the bottom of pyramid model, and a series of image layers (band) is generated with the same scope but different resolutions by resampling. Finally each tile is assigned a unique identifier and stored as one row in HBase table. Figure 3.4 illustrates the procedure of storing EO data coverage in the HBase.

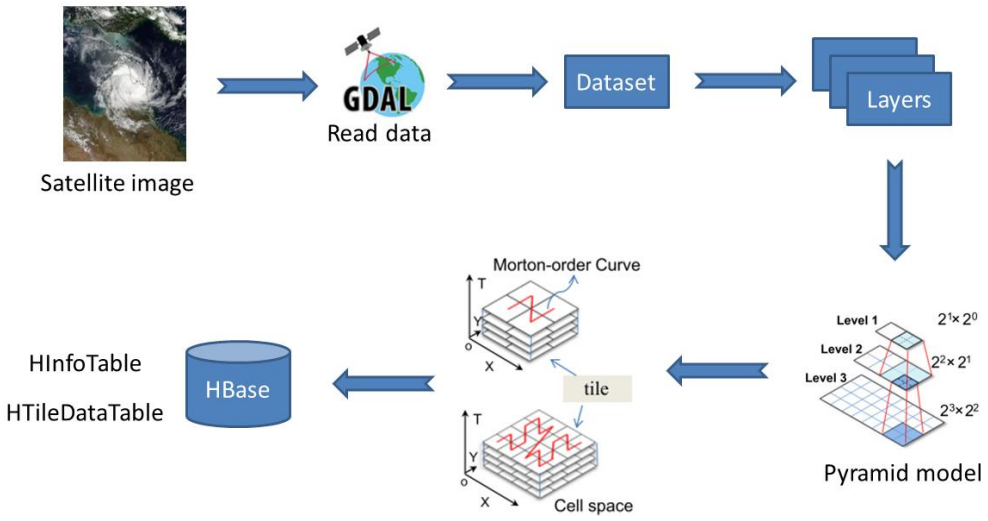


Figure 3.4. The procedure of storing EO data coverage in the HBase

As introduced in section 2.3.2, the row-key is the only way through which the user can query required records in the HBase tables. An effective row-key design pattern should be investigated first in order to improve the querying efficiency. The Geohash (Balkic et al., 2012) used by the GeoSPA makes a great choice for the row-key design because it's inexpensive to calculate and the prefix makes it easy to find nearest neighbors. Geohash is a latitude/longitude geocode system to encode and decode the geospatial coordinates in the form of latitude-longitude pair into a compact form. A Geohash code is actually a character string representing a fixed

geographic bounding box. Based on Balkic, the Geohash algorithm divides spatial regions into a hierarchical structure by interleaving bits generated from latitude-longitude pairs and then transfers these bits into a character string using the Base32 character alphabet. Figure 3.5 illustrates how to transfer geospatial coordinates to Geohash-based string using the latitude and longitude coordinates of (40.78° N, 73.97° W) as an example.

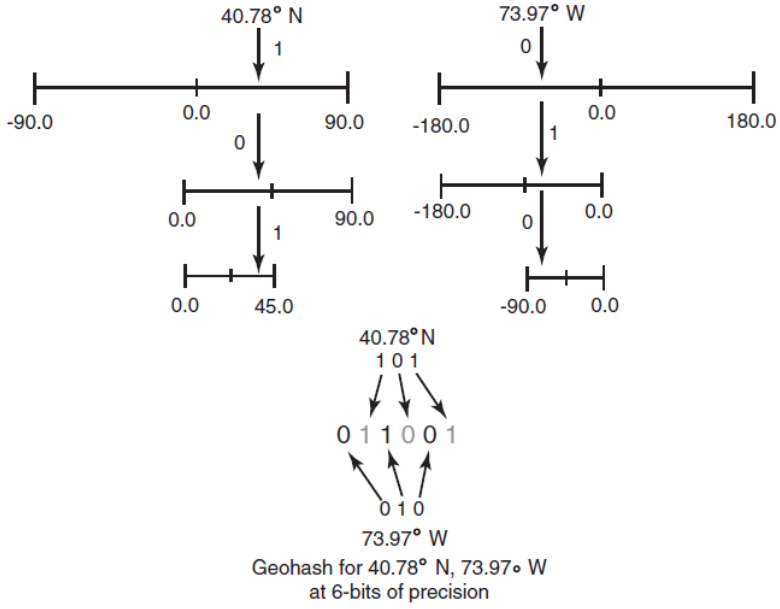


Figure 3.5. Illustration of how to transfer the latitude and longitude coordinates to Geohash code (Dimiduk et al., 2013)

As shown in Figure 3.5, the Geohash code is actually a sequence of bits which reflects the increasingly precise division of latitude and longitude. And these bits are all represented using character strings based on the Base32 encoding alphabet. For example, the latitude of 40.78° N falls in the upper half of the [-90.0, 90.0], which indicates the first Geohash bit should be 1. The second bit becomes 0 because after the first decision, the new range becomes to [0.0, 90.0] and 40.78 locates on the lower half of the new range. Similar operation is performed and the third bit is 1 because 40.78 locates in the upper half of new range [0.0, 45.0]. The same

calculation is performed for each dimension of the coordinates by halving the range values and determining which half the point locates in. If the coordinate is less than the value of the midpoint of the current range, it's a 0-bit. Otherwise, it's a 1-bit. This process is repeated until a predefined precision (which is usually the total length of the Geohash code) is reached. Different with the other methods which apply the bit sequence from each dimension independently, the Geohash method weaves the bits of all dimensions together to generate the hash code. This spatial partitioning approach makes Geohash own the capability to reflect the spatial locality property.

All levels of EO data tiles are stored in single table. Based on the row-keys, the HBase can automatically partition the big table into multiple storage blocks which are stored and managed on distributed storage regions. The coding mechanism introduced above ensures all the cell-compatible tiles to be stored in the same or closed regions, which can get rid of transferring massive data when user needs to do some spectral or spatial analysis within the same geographic domain. A row-key coding mechanism based on the Geohash is developed in this research to effectively identify and distinguish the tile. The goal of the coding mechanism is to create row-keys which make the close tiles have similar row-keys. The best advantage of this approach is to improve the efficiency of deriving multiple tiles at one time. As shown in Figure 3.6, the tile key has 32 bits, among which 8 bits in the beginning represent the Geohash codes, the next two bits represent level number of the pyramid model, 10-13 is the column index, 14-17 is the row index of the block, 18-20 is the band identifier. Finally, 21-28 is the timestamp represent by a long integer type and 29-31 is sensor identifier.

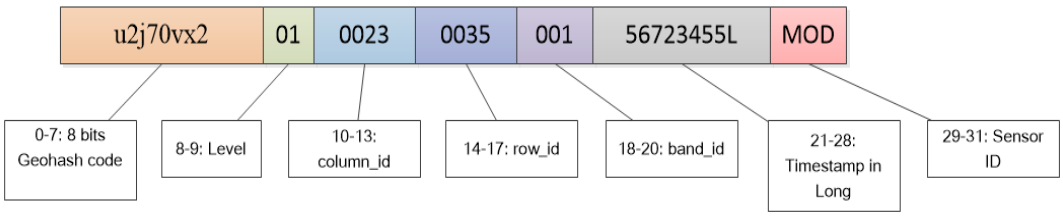


Figure 3.6. Illustrations of the row-key design schema

Two tables, named HInfoTable (Figure 3.7(a)) and HTileDataTable (Figure 3.7(b)), were designed to store the metadata of EO dataset and actual tile data respectively. When a new dataset is generated by sensing platforms and uploaded to the GeoSPA, a new record is generated to store the metadata in HInfoTable. And then the GeoSPA splits the image into tiles based on the tile-based storage scheme introduced above, generate the row-key for each tile, and write the image into the HTileDataTable. After storing the tiles into HBase, the GeoSPA can access these data through the standard HBase API.

rowkey	Timestamp	meta							
		Sensor	boundary	Xsize	Ysize	BandNun	LevelNum	Owner	…
0000…01	T1	MTSAT	min_x,min_y max_x,max_y						…
…	…	…	…						

(a)

rowkey	Timestamp	meta							data
		Raster_id	boundary	level	row	col	band	sensor	
u2j7…123MTS	T1	0000…01	min_x,min_y max_x,max_y	7	4	5	VIS	MTSAT	byte[]
xg5h…523MTS	T1	0000…01	min_x,min_y max_x,max_y	7	12	8	VIS	MTSAT	byte[]
…	…	…	…	…	…	…	…	…	…
bc4a…343MTS	T1	0000…01	min_x,min_y max_x,max_y	8	34	21	IR1	MTSAT	byte[]

(b)

Figure 3.7. Illustrations of the table design schema of (a) HInfoTable, and (b) HTileDataTable

3.2.3 RESTful-based Map Service

Many studies have been conducted on standardizing the storage scheme of map tiles and the ways to offer tiled-based geospatial data services. OGC released the Web Map Tile Service (WMTS) standard (Maso et al., 2005), which complements the existing OGC Web Map Service standard to support efficient tile-based map services. WMTS provides an open-source alternative to proprietary web mapping services, such as Google Maps and Microsoft Bing Maps. Furthermore, WMTS defines standard approaches to define the properties about tile storage scheme, projection, resolutions and so on (Sample et al., 2010). Based on WMTS standard specification, three service operations are defined: *GetCapabilities, GetTile,* and *GetFeatureInfo*.

WMTS uses a tiling model to describe the predefined images, which divides the space into a fixed tile matrix. To facilitate users to access the EO data stored on the GeoSPA and improve the performance of data processing, the GeoSPA offers an implementation of OGC WMTS standard for serving tile-based EO data service. The WMTS service implementation offered by the GeoSPA is utilized the RESTful programming structure in order to allow it to be easily integrated into a wide array of Web 2.0 applications. Although any reasonable tiling strategy can be used, the Google's approach was applied in this research in order to enable the seamless integration with the Google Map API on the client side. Figure 3.8 illustrates the architecture of WMTS implementation in the GeoSPA.

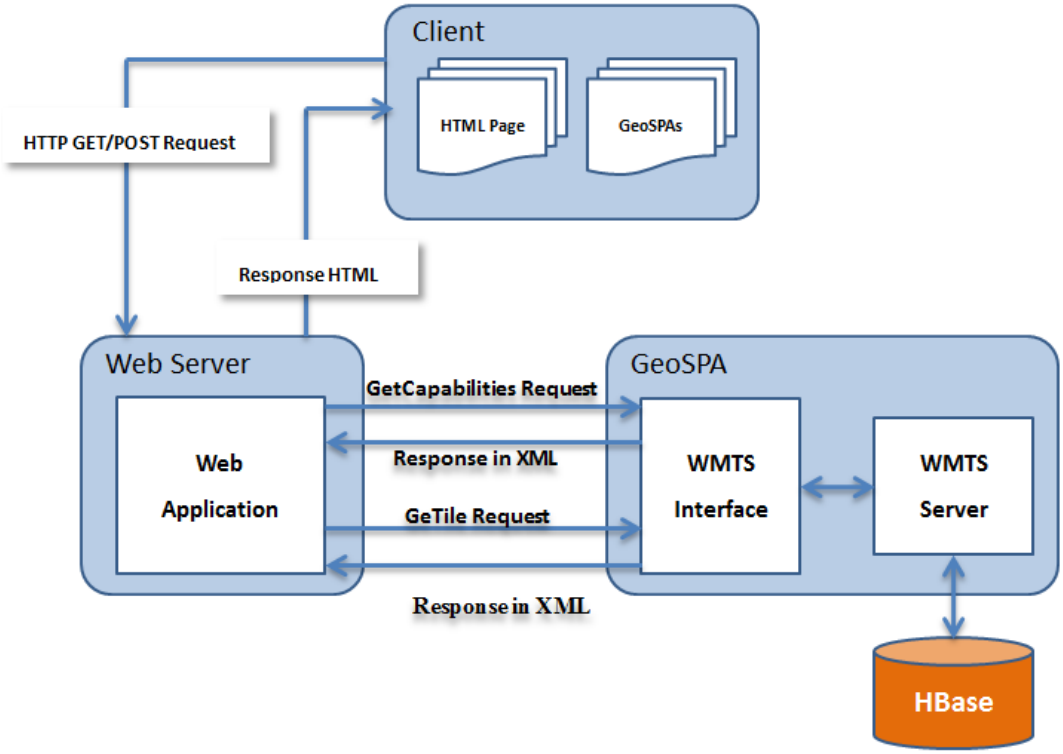


Figure 3.8. Structure of WMTS implementation of the GeoSPA

As shown in Figure 3.8, to visit tiled map service offered by the GeoSPA WMTS server, the first step is to select a zoom level and bounding box, which are used to determine the tiles for prefetching. The next step is to iterate through all the tiles at all zoom levels and retrieve map tiles from the HBase through the Geohash-based row-keys. The retrieved tiles will be cached into RAM (Random Access Memory) by the GeoSPA in order to improve the serving efficiency. The GeoSPA WMTS server has a built-in tile caching system which is used to cache image tiles. As new tiles are requested, the GeoSPA WMTS server intercepts these requests and returns corresponding tiles (PNG or JPEG) as necessary to client users. At the same time, these tiles are cached by the tile caching system. Once the request for the same tiles is received, the GeoSPA directly picks corresponding tile images from the tile caching system and returns them to client, which making for a more seamless user experience by increasing the speed of map rendering many times.

As introduced above, the WMTS tile resource represents a single cached tile, which is a fragment of an image in the context of WMTS specification. Users can get the WMTS tiles based on the RESTful syntax as defined by OGC WMTS specification. In response to a valid request for a tile representation from a client, a WMTS server shall send either an image representation of the tile or a reference to an image. An image is the most typical representation but representations in other formats are also allowed. The RESTful-based request pattern can be described as below:

http://<wmts-url>/<layer>/<style>/<tilematrixset>/<tilematrix>/<tilerow>/<tilecol>.<format>

Table 3.1 shows the description of the URL tokens defined in the RESTful syntax.

Table 3.1: The description of parameters of RESTful syntax

URL Token	Optional/Required	Description
layer	Required	Valid layer identifier advertised in WMTS service metadata
style	Optional	valid style identifier advertised in WMTS service metadata
tilematrixset	Required	TileMatrixSet is a concept in OGC WMTS specification which is similar to Tiling Schema. The identifier of one of the TileMatrixSet advertised in WMTS service metadata, which includes well-known TileMatrixSet like Google Maps Online, or a customized TileMatrixSet defined by service publisher.
tilematrix	Required	TileMatrix is a concept in OGC WMTS specification which is a collection of tiles for a fixed scale. The identifier of one of the TileMatrix defined in a particular TileMatrixSet

tilerow	Required	Row index of a tile matrix
tilecol	Required	Column index of a tile matrix
format	Optional	the suffix of one of the supported formats advertised in WMTS service metadata

3.3 GeoSPA Processing Service Model

3.3.1 Model Description

The GeoSPA processing service model provides a knowledge model to describe and manage the deployed geospatial processes as well as their compositions. The proposed knowledge model consists of five fundamental elements: Belief, Goal, Process, Task, and Plan. Belief represents the current states of the agent's internal and external worlds. Goal is the set of goals that the service agent wants to achieve. Process represents the action that the service agent can perform. Task is the basic execution unit which contains three components: the behaving agent, the behaved process, and the specified output parameters. Plan is composed of a set of tasks as well as their execution sequence for achieving a certain goal. The details of these five notions were introduced below.

- Belief Model

  The GeoSPA model applies Belief to represent the knowledge about itself and its environment. The knowledge of the GeoSPA can be classified into two categories, which are social knowledge and basic knowledge. The social knowledge indicates the relationships among service agents as well as other service agents' information such as their capabilities and addresses. The basic knowledge is the fact that the service agent knows about itself such as the states of a service. The basic knowledge is denoted as tuple: $\langle n, t, u, v \rangle$, where n is

the name , t is type , u is the unit measure for the basic knowledge, and v is the value. Social knowledge is represented by the tuple: $\langle AID, A_c \rangle$, where AID is the neighbor's identifier composed of agent name and agent address, A_c is the capabilities of neighbor. Through the social knowledge, the dependence relations among distributed service agents can be constructed.

- Process Model

Three types of Processes were defined by the GeoSPA: internal process, communicating process, and geospatial process (GP). The GP can be performed by an agent by defining the input parameters, output parameters, and executable model entity. The service providers can deploy their geospatial model as an instance of GP onto the GeoSPA, which can be exposed as the standard-compliant web-based geospatial processing service. Figure 3.9 illustrates the UML class diagram of Process object.

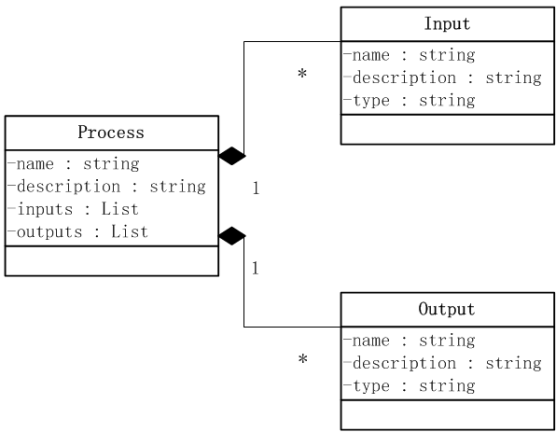


Figure 3.9. UML class diagram of the Process object.

As shown in Figure 3.9, each Process has a 'name' attribute and a 'description' attribute, which are used to describe the basic information of a process. Furthermore, each Process object owns two List objects for Processes composition, which are 'inputs' and 'outputs', respectively. Each 'inputs' has several 'input' objects, each of which represents an input parameter containing three attributes: name, description, and type.

The ‘type’ attributed is used to connect two processes by comparing its value between the output of process and the input of another process.

- Task Model

The Task model is the basic execution unit of the GeoSPA, which can be represented using an agent-process-parameters: $t(spa, p, v)$, where spa is the name of the GeoSPA that performs this task, p is the GP to be performed, and v is the user-specified output variables.

- Goal Model

The Goal model is the business goal that the agent achieves, which is denoted as tuple: $\langle inputs, outputs \rangle$, where the inputs and outputs denote the input parameters and output parameters, respectively.

- Plan Model

The Plan model encapsulates the business logics of how to use a set of $Tasks$ to achieve a specific Goal by defining the execution sequence of these Tasks.

Figure 3.10 illustrates the relations of Belief, Process, Task, Gola, and Plan model using the UML class diagram. A more detailed introduction about these five models was given in section 4.1. These five models compose the knowledge-base of the GeoSPA. When the service provider deploys GP onto the GeoSPA, the embedded knowledge-base can automatically update itself by transferring the user-specified metadata of GP into these five models.

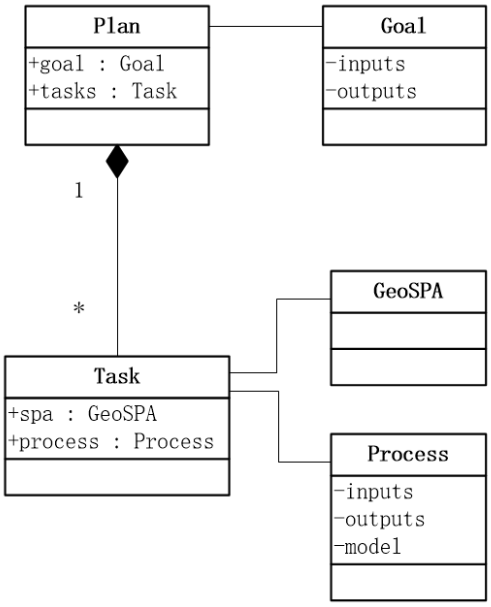


Figure 3.10. UML class diagram of the GeoSPA processing service model

3.3.2 Geospatial Service Programming Interface (GeoSPI)

To facilitate model developers to deploy GPs as the GeoSPA processing services, the GeoSPI was proposed in this section. The GeoSPI was designed to regulate the behavior of each geospatial processing service by defining a series of programming rules (programming interfaces and annotations for labelling metadata). Each GP should follow the rules to expose their metadata and services to client and other processes for interoperability. Many Object-oriented Programming (OOP) languages (e.g. Java, C++, and C#) supply a special file type called 'interface', which can be used as a contract between the classes that implement the interface and the outside world. Through GeoSPI, the service user can access the geospatial processing service through the standardized interfaces without caring the detailed implementation of the process. This feature is extremely useful in the high-level distributed computing environment (i.e., cloud computing), where many geospatial models and data resources were developed and maintained by different organizations and agencies.

The GeoSPI defines the ways for model developer to specify the metadata of a GP and for the GeoSPA to read these metadata. The traditional way of representing the metadata about a geospatial process is to store these data in a single file, which is separated from the executable model program codes (Matott et al., 2009). This approach is not able to provide the model users legible descriptions about the geospatial processes they need. Furthermore, separating the model metadata with executable model program codes makes it error-prone because the model developers have to spend extra time to guarantee the synchronization between the metadata files and model modifications. Combining the metadata directly to model program codes, which is applied by GeoSPI, can get rid of these problems and facilitate model clients to retrieve the metadata about the input and output parameters of a GP. It can also enable model developer to maintain the GPs more efficiently. The 'annotations' mechanism provided by Java programming language supplies a convenient approach to combine the metadata information with the model modules. Three annotations were defined by GeoSPI to support the combination of model metadata with model program codes:

1) *@DescribeProcess* labels the basic information of the GP.
2) *@DescribeOutput* labels the metadata about the output parameters of a GP.
3) *@DescribeInput* labels the metadata about the input parameters of a GP.

To design geospatial processing services that can be deployed and managed by the GeoSPA, the geospatial model developers only need to implement their own algorithms in the '*execute*' function defined in the GeoSPI and specify the input and output parameters, leaving the complex control algorithms to the GeoSPA. The code below demonstrates how to implement the GeoSPI on the IDDI model, which is a widely used dust storm detection model introduced in Chapter 5.

```
@DescribeProcess (name= " MTS_SBDART", description = "……")
public   class   MTS_SBDART implements GeoProcess
{
      @DescribeOutput (name="aot550nm" description = "…" type="xs:AOT550nm")
      public String   aot550;
      @DescribeOutput (name="height" description = "…" type="xs:DustHeight")
      public String   height;
      @DescribeOutput (name="reff" description = "…" type="xs:Reff")
      public String   reff;
      public NetcdfDataset execute(
            @DescribeInput (name = "dust_region", description = "… ", type="xs:Dust") NetcdfDataset   dust,
            @DescribeInput (name = "dust_region_name", description = "… ", type="xs:String") String band1,
            @DescribeInput (name = "mtsat", description = "… ", type="xs:MTSAT") NetcdfDataset mtsat,
            @DescribeInput (name = "input1", description = "… ", type="xs:String") String band1,
             … )
             {
                    // Implementation codes of geospatial algorithms
              }
}
```

As shown in the code above, the model class (InfraredDifferenceDustIndex) implements its own algorithms in the 'execute' functions defined in the *GeoProcess* Java interface. In addition, three GeoSPI annotations introduced above were labeled on different part of this class to describe the metadata information (i.e. model description, input, and output parameters) of the algorithm.

To deploy the implementation of GeoSPI as the GeoSPA processing service, the program codes of GP needs to be compiled by the Java Virtual Machine (JVM) into a JAR package file together with its dependent libraries. Then the JAR package is submitted into the directory of the GeoSPA which has been specified as a geospatial process deployment directory. After deploying the GPs onto the GeoSPA, model clients can then call these GPs following the WPS operations across the Internet. Before calling the required GP, client should identify and understand all the available

GPs on the GeoSPA, which can be retrieved through parsing the metadata return from the *GetCapabilities* operation. Although the metadata returned from the GetCapabilities operation can help clients understand the GPs, the information is insufficient for accessing the GP practically. Detailed process metadata can be obtained from the *DescribeProcess* operation. Figure 3.11 illustrates the geospatial processing service metadata for the IDDI model. The GeoSPA processing services allow clients to run a given GP on the GeoSPA by calling the *Execute* operation. Because the GeoSPA processing services are shared using open standards (e.g., OGC WPS, GML), any tool or system that complies with those standards can be used to access these services.

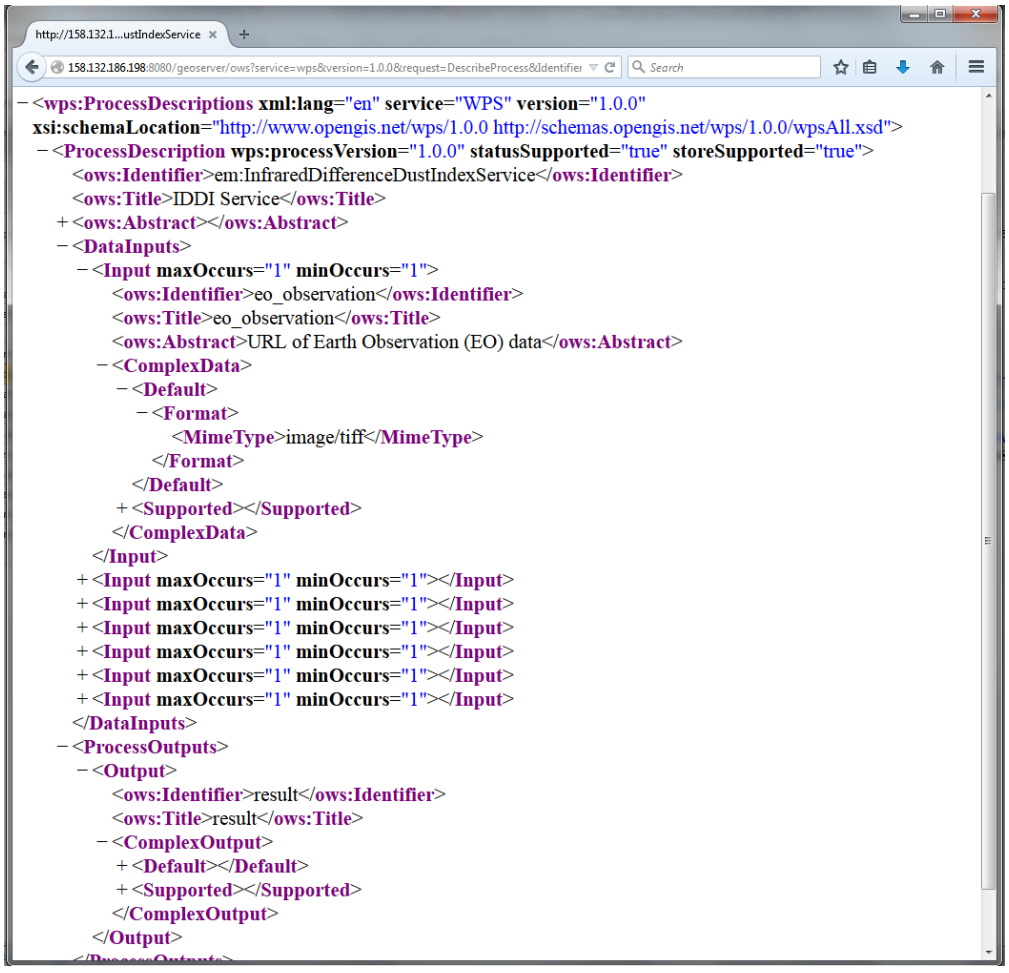


Figure 3.11. The GeoSPA processing service metadata returned from calling the DescribeProcess operation using a Web browser

Upon receipt of the compiled JAR package, the GeoSPA reads the metadata from the annotations defined by GeoSPI, and then the metadata will be imported to the local knowledge-base based on the processing service model introduced above. Figure 3.12 illustrates the Goal object of the 'InfraredDifferenceDustIndex' model in the form of XML document.

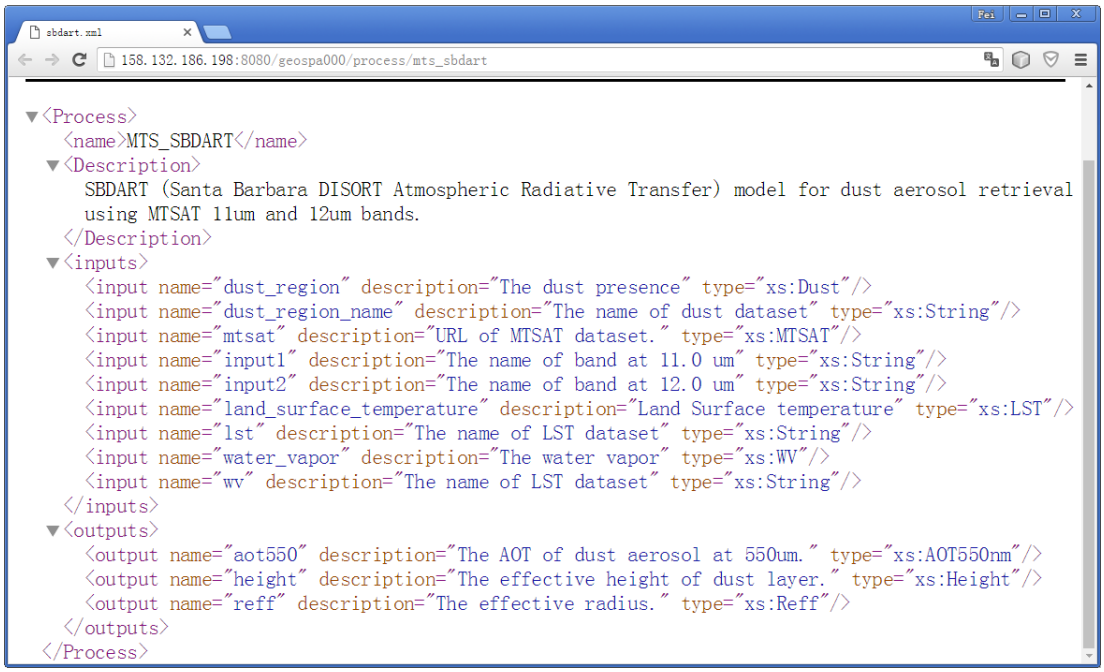


Figure 3.12. The Goal model of RAT model represented by XML

3.3.3 Implementation Strategies of GeoSPI

As introduced above, to deploy the GPs to the GeoSPA as geospatial processing services, model developer must implement the 'execute' method defined in the *GeoProcess* interface using their model program codes, and then specify the metadata information through the annotations defined in GeoSPI. Considering that multiple types of programming languages (e.g., C/C++, Java, FORTRAN, Python, Matlab, and IDL) have been used for developing geospatial models and each of them needs a special execution environment, two implementation strategies of GeoSPI

were introduced in this section. The details of these two strategies were introduced below:

- Direct Implementation.

Figure 3.13 illustrates the direct implementation of GeoSPI. Using this way, the model developers directly implement the GeoSPI using Java programming language, which needs the same execution environment as the GeoSPA does. Then the GeoSPI-based GP can be directly deployed onto the GeoSPA after compiling for offering geospatial processing service. Compared with the adapter implementation way introduced below, the direct implementation approach is much stable and can supply better performance. However, the disadvantage of this approach is that the modification of GP is difficult because the model developers have to access to the source code of GP for editing and the new GP has to be re-deployed onto the GeoSPA in order to expose the new version of geospatial processing service. In addition, the model developers must be familiar with Java programming because all of the model modules must be implemented using Java.

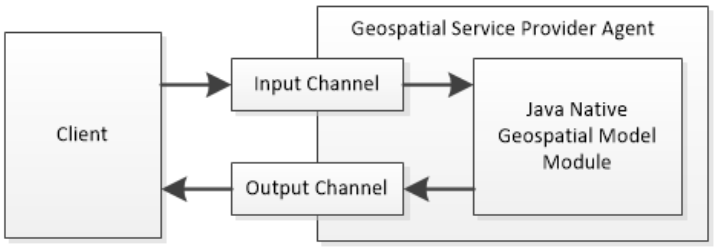


Figure 3.13. Direct implementation of GeoSPI

- Adapter Implementation

Figure 3.14 illustrates the adapter implementation of GeoSPI. Using this approach, an adapter module is developed to implement the GeoSPI for the

geospatial model module and connect the model module to the GeoSPA. The adapter module translates model calls and inputs/outputs from the GeoSPA form to the form supported by model, and it avoids modifications to the model. This approach is particularly suitable for integrating geospatial models developed using various programming language with the GeoSPA. However, the adapter implementation usually requires transforming input and output data back and forth between the model process and the process where its adapter module is running, which slows down the performance.

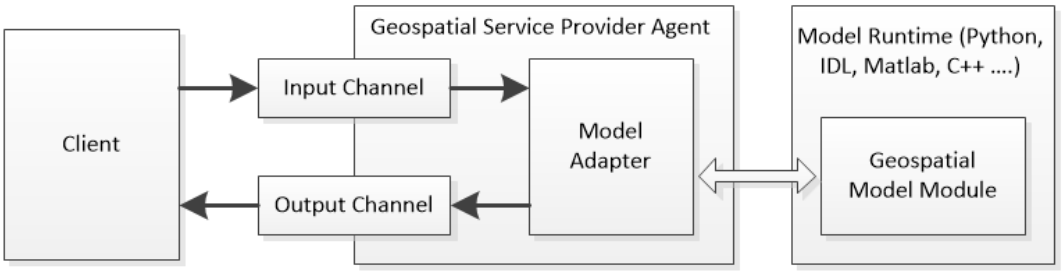


Figure 3.14. Adapter implementation of GeoSPI

Table 3.2 gives a comparison of these tow implantation strategies.

Table 3.2. Comparison of tow implantation strategies

Strategy	Advantages	Disadvantages
Direct implementation	● Better performance. ● Easy to deploy and manage. ● The model can be deployed on any GeoSPA node distributed on the internet.	● The model developer must be familiar with Java programming. ● Hard to utilized existing geoscience model resources
Adapter Implementation	● Make full use of existing models that implemented by other programming languages.	● The transformation of input and output data back and forth slows down the performance. ● The execution of model module need special runtime environment, which

		hinders the distributed execution of models.

3.4 GeoSPA Computing Service Model

The GeoSPA computing service model was designed for a) performing the GPs, and b) managing and processing the states generated during the GP execution. The GeoSPA offers a runtime environment for performing GPs, which may come from either local GeoSPA node or remote GeoSPA nodes. In addition, the execution of GP can generate a set of states, which should be captured and handled correctly and efficiently in order to guarantee the service consumer to get the correct result. To address this problem, the GeoSPA computing service model is first introduced in this section, and then the GeoSPA Finite State Machine (G-FSM) is proposed for state management.

3.4.1 Model Description

Figure 3.15 depicts how the GeoSPA computing service model works.

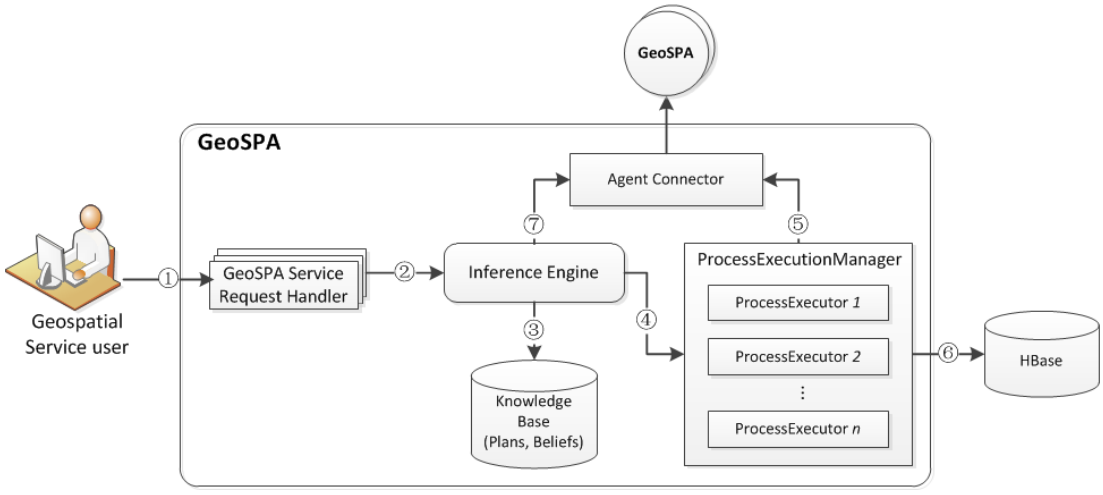


Figure 3.15. Life cycle of the GeoSPA computing service procedure

The symbol ⓝ (n is a positive integer) in Figure 3.15 indicates the position of

a step in the overall life cycle. In step ①, the service user submits a XML-encoded request document based on OGC WPS standard to the GeoSPA. Once the user submits the service requirement, the GeoSPA first creates a *GeoSPA Service Request Handler* object to process the request. A service requirement object denoted as $\langle r_i, r_o \rangle$ which is generated and delivered to the embedded *Inference Engine* for next operation (step ② in Figure 3.15). The Inference Engine is responsible for analyzing the service requirement and checking if there is any GP on local *knowledge-base* which can meet this service requirement (step ③ in Figure 3.15). If it is true, then the GeoSPA initiates a *ProcessExecutor* object, which is actually a Java object that can be executed in parallel to perform this GP. And this ProcessExecutor object will be sent to the *ProcessExecutionManager* for execution (step ④ in Figure 3.15). Upon receipt of a new ProcessExecutor object, the ProcessExecutionManager first checks if there are input parameters referring to the output of ProcessExecutors persisted by other GeoSPAs. If it is true, then the ProcessExecutionManager postpones the ProcessExecutor and sends a request to get the result generated by other ProcessExecutors through *Agent Connector* component (Step ⑤ in Figure 3.15). Once the GP finishes successfully, the result is written to local HBase system and an XML-encoded document describing the result is sent back to service user. Finally, if there is no GP to meet the user-specified service requirement in local knowledge-base, the GeoSPA forwards the service requirement to its neighbor GeoSPAs for further processing (Step ⑦ in Figure 3.15).

3.4.2 GeoSPA Finite State Machine (G-FSM)

The G-FSM used in this research can be described by a 6-tuple: $(S, s_0, \Sigma_e, \Sigma_o, T, V)$, where S is a finite set of states. Let $S = \{s_i | 0 \leq i \leq n\}$, then s_0 is called the initial state of the G-FSM. Σ_e is a finite set of events, yet Σ_o is a finites set of operations. Each operation $op \in \Sigma_o$ can be triggered by the event e $\in \Sigma_o$. V is a set of global variables and each variable $x \in V$ can be used by every state $s_i \in S$. T is a finite set of transitions and each $t \in T$ can be represented by a 5-tuple: $t = (s_s, e, op, e', s_e)$, where s_s is the start state of t, op is a sequential operation (e.g. assignment statement), e is the event triggers this op, e' is the generated event, and s_e is the end state of t. Figure 3.16 illustrates the state transition diagram of G-FSM model.

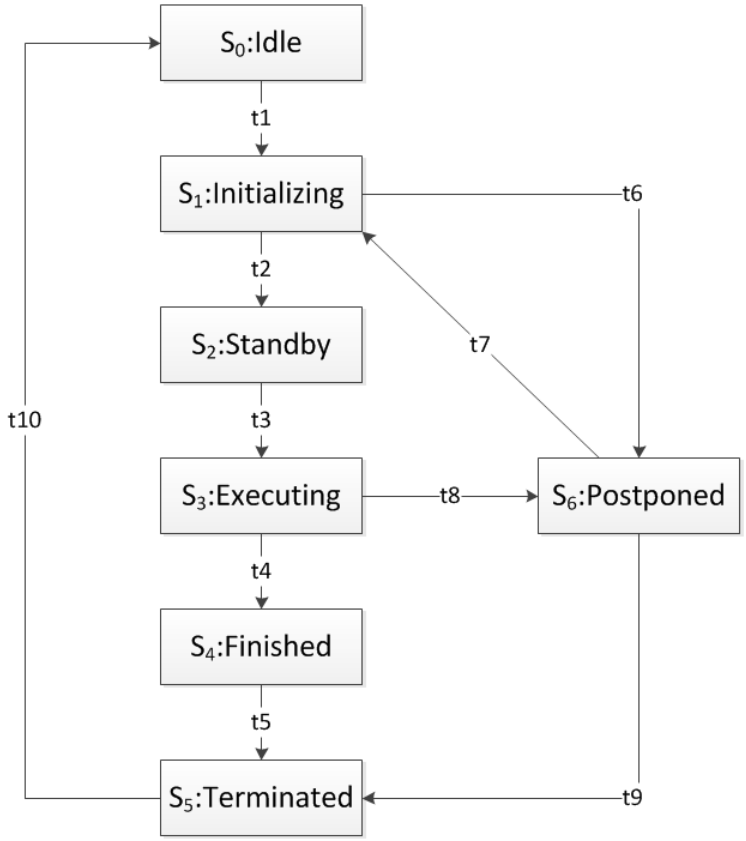


Figure 3.16. Transport protocol of the G-FSM model

The G-FSM totally defines seven runtime states (Table 3.3) and ten state transitions (Table 3.4). As shown in Table 3.3, there is a set of variables which in particular are used to represent the machine state and are called state or major state. A state transition occurs when one of the machine's state changes to another one. As shown in table 3.4, each transition has two major states: start state (s_s) and end state (s_e). A transition t may have one or more atomic operations (op) to be executed when t is taken. The Σ_1 and Σ_2 is based on the FIPA Communicative Act Library Specification (FIPA, 2002), which defines a series of primitives to standardize the communicative acts between various service agents.

Table 3.3. States and descriptions of the G-FSM model

State	Name	Description
S_0	Idle	The GeoSPA is available and waiting for new incoming request.
S_1	Initializing	The GeoSPA receives incoming request and begins to i) analyze the request content, ii) initialize the available GP and load it into local memory system, and iii) read required input parameters.
S_2	Standby	The target GP is ready. The GeoSPA waits for further operations from users.
S_3	Executing	The GP is being executed.
S_4	Postponed	The geospatial process is postponed due to some exceptions. The GeoSPA waits for further operations from service consumer.
S_5	Fished	The GP is completed successfully.
S_6	Terminated	The GP is terminated and occupied computing resources will be released.

Table 3.4. Core transition in the transport protocol of EFSM model

$T: S_e \rightarrow S_s$	e	e'	Operations
t_1: $S_0 \rightarrow S_1$	$\mathrm{CFP}(r_i, r_o)$	$\mathrm{CFP}(Goal)$	Upon receipt new user-specified service requirement in the form of input and output pair: (r_i, r_o), the GeoSPA launches a new thread to initialize the requested processing services as well as user-specified input parameters. The XML-encoded request will be transferred into $Goal$ model, which can be understood by the GeoSPA for further analysis.
t_2: $S_1 \rightarrow S_2$	$\mathrm{CFP}(Goal)$	$PROPOSE(GPs)$	The GeoSPA begins to analyze the $Goal$, and retrieves all GPs in local knowledge-base to determine if there is any GP that can achieve the Goal. The fitting GP's detail information is returned to users through the $PROPOSE$ message. Finally, GeoSPA waits for further operations.
t_3: $S_2 \rightarrow S_3$	$\mathrm{ACCEPT(GP)}$	$\mathrm{CONFIRM(GP)}$	If the potential GP is approved, the user should send a ACCEPT message to rotifer the GeoSPA to start the GP execution. The GeoSPA initializes a new Thread object called 'ProcessExecutor' with the input parameters. Then this thread object is delivered to the local Process Execution Manager for execution, and a $\mathrm{CONFIRM(GP)}$ message is returned to service consumer for accessing the result of GP.
T_4: $S_3 \rightarrow S_4$	NULL	$\mathrm{INFORM(GP, result)}$	Once the GP finishes without failures, the final result of GP execution is written into HBase system by the GeoSPA, and meanwhile an $\mathrm{INFORM(GP, result)}$ message is generated and returned to service consumer. Finally t_5 is triggered.
t_5: $S_4 \rightarrow S_5$	NULL	NULL	The GeoSPA releases the occupied computing resources.

t_6: $S_6 \rightarrow S_1$	INFORM(r_i, r_o)	NULL	The GeoSPA receives new user-specified service requirement through an INFORM(r_i, r_o) message, and the state of GeoSPA changes to S_1.
t_7: $S_1 \rightarrow S_6$	FAILURE(exception)	NULL	The GeoSPA catches the GP run exceptions, returns them to service consumer, and waits for further operations.
t_8: $S_3 \rightarrow S_6$	FAILURE(exception)	NULL	The GeoSPA catches the GP run exceptions, returns them to service consumer, and waits for further operations.
t_9: $S_6 \rightarrow S_5$	CANCEL(GP)	NULL	Upon receipt the CANCEL(GP) message from service consumer or the setting timeout is reached, the GeoSPA triggers the process of terminating the GP.
t_{10}: $S_5 \rightarrow S_0$	NULL	NULL	The GeoSPA changes its status to 'Idle' and waits for new incoming request.

3.5 Concluding Summary

This Chapter introduced a Geospatial Service Provider Agent (GeoSPA) structure which could be used as one-stop geospatial service solution for SDI. The overall structure of the GeoSPA was first introduced in section 3.1 as well as three service models: EO data service, processing service, and computing service. These services are fully compliant to the OGC Web Service specifications to ensure that different services can be interacted with each other in a unified fashion. Then these three service models were introduced in details respectively in section 3.2, 3.3 and 3.4. The GeoSPA is the core component for supporting geospatial services in the cloud. In next Chapter, a P2P-based HyperCGSF framework was proposed to manage and orchestrate multiple GeoSPAs in a decentralized manner.

Chapter 4: The HyperCGSF

As introduced in Chapter 3, the GeoSPA is designed as the single infrastructure node of SDI based on SOA. In geoscience, the SDI is actually a network of inter-linked infrastructure nodes and each single node in SDI maintains a set of geospatial services following the SDI principles grouped by geographic criteria commonly. In such a context, individual infrastructure node can be seen as the service providers which can be incorporated into common solutions for complex geoscience problems. However, it is complicated for non-SDI experts to discover and utilize required geospatial resources over the large amounts of SDI nodes due to the lack of proper communication scheme for connecting and coordinate heterogeneous and distributed SDI nodes (Granell et al., 2013).

In this chapter, a P2P-based networking structure called Hypercube Geospatial Service Framework (HyperCGSF) and a set of distributed algorithms are introduced which can be exploited to carry out efficient searching and broadcasting for cooperating distributed GeoSPAs to achieve complex geoscience problems. The P2P technologies and systems have been proven effective for constructing distributed systems with large-scale. Unlike the commonly used centralized structure in geoscience, e.g. BPEL, the P2P system applies the mutual cooperation pattern through which each peer can dynamically utilize other peers' resources (e.g., CPU, storage, bandwidth, etc.). The P2P-based service architecture is able to process large volumes of geospatial dataset, while preventing bottleneck in system performance and eliminating the possibility of single-point failure.

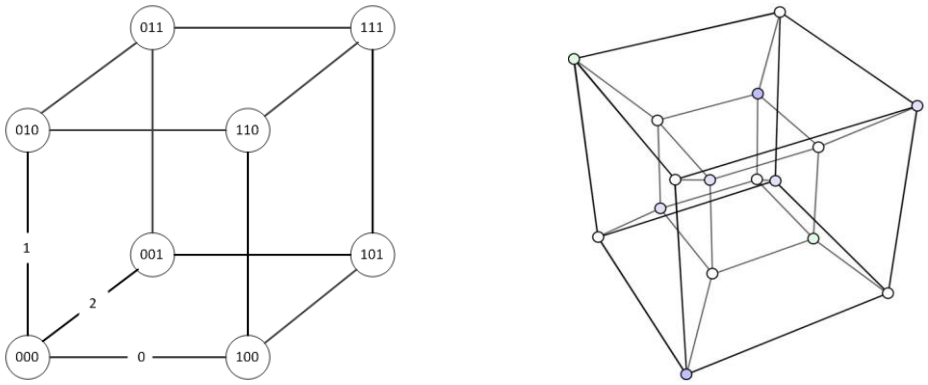
4.1 Hypercube Network Topology

4.1.1 Introduction

In order to build a scalable distributed system and avoid the worst case of

network diameter, several factors need to be considered. First, the degree of P2P nodes should be limited, and the number of networking links that one node needs to maintain should be as few as possible. Second, the networking topology should support redundancy and fault tolerance, which means that the single-node failure will not lead to the breakdown of the whole distributed system or hampering search and broadcast tasks severely. Lastly, the number of communication messages during the broadcast and search operations should be evenly allocated among all peers in the P2P network.

To meet the requirements introduced above, Schlosser et al (2002) propose a network topology called 'hypercube' to manage the peers in a P2P network. Figure 4.1 depicts the two examples of hypercube topology drawn in 3D with the base b=2 and b=3, respectively. Based on Schlosser, essentially every node can perform as the root node of a tree which spans all nodes in the hypercube. In P2P-based distributed systems, the network diameter, denoted as Δ, refers to the shortest path between most distant nodes. Δ is a crucial parameters to reflect the efficiency of a P2P network for search and broadcast. The worst case of Δ is $O(n)$, where n is the number of peers in a P2P network. A complete hypercube topology has $N = b^{L_{max}+1}$ nodes and has a Δ equals to $\log_b N$, where $L_{max}+1$ is the number of dimensions spanned by the cube. There are $(b-1)\cdot(L_{max}+1)$ neighbors for each node in the hypercube. Based on Figure 4.1, the hypercube topology is symmetric because there is no node has the more prominent position than other nodes. The most important feature is that every node in a hypercube can perform the source of a broadcasting task, yet the load always is always equally shared. This advantage is crucial for the load balancing in a P2P network.



(a) (b)

Figure 4.1. Illustration of Hypercube topology with the base: (a) b=2 and (b) b=3

Some definitions were stated in this study to describe the topology of a P2P network. A P2P network can be expressed by $G = (V, E)$, where E is the set of edges and V is the set of vertexes. Given a binary based Hypercube, The edge in E is labeled: Node X is denoted as the i-neighbor of node Y or $X = iN(Y)$ iff node X is Y's neighbor in ith dimension. There may be extended neighbors represented by $X = N(Y) = \{y_0, y_1, \dots\}(Y)$ for each node, where N represents the neighbor link set which indicates the sequence of i-neighbors one would have to follow in the complete hypercube graph to reach node X from node Y and vice versa. A hypercube node connects with other nodes as its neighbors with a link set using a transport network address. The dimension label starts at i=0 and the maximum dimension label of a node is L_{max}.

4.1.2 Broadcast Scheme in Hypercube

The hypercube topology can achieve more efficient broadcast tasks than traditional network topology. Traditional network topology such as the tree structure could not offer an effective and reliable broadcast scheme due to several factors listed below. First if the broadcast is performed by a non-root node, the tree must be reconstructed completely by setting the new sender node as the root node. Once the tree is reconstructed, there is an overhead associated with building new connections between nodes in order to create the tree with a new root node. Second, the reconstruction may result the tree to be unbalanced and suffer from poor performance

measures such as poor load-balancing across nodes and a long average path length. On the contrary, the hypercube topology offers a more effective and reliable broadcast scheme than the tree structure. As shown in Figure 4.2, a tree structure can be superimposed based on the hypercube topology structure easily. For hypercube is relative symmetry, the root node of the superimposed tree can be any node in the hypercube.

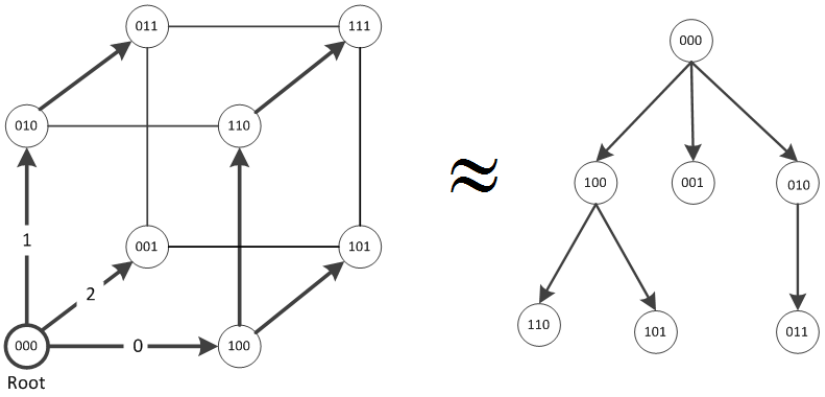


Figure 4.2. Illustration of broadcast operation conducted by node 000

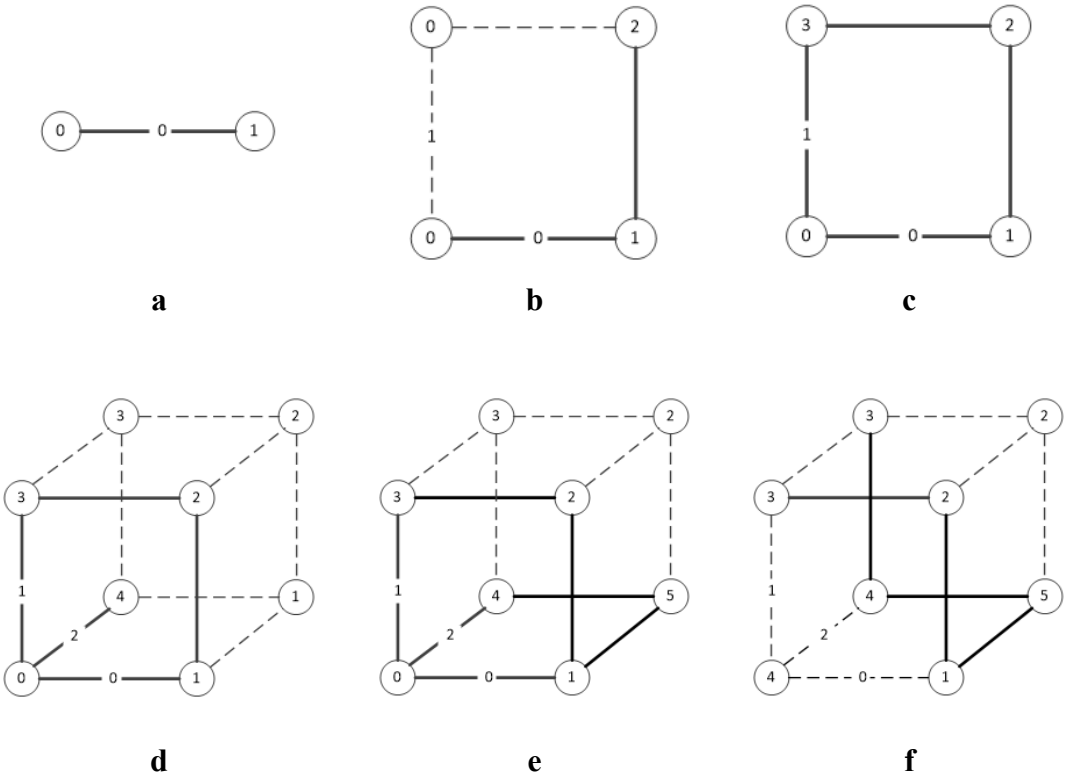
A broadcast scheme is proposed in this research which guarantees each node in the hypercube receives a message exactly only once. As shown in Figure 4.2, it can be seen that totally $N-1$ messages are needed for a broadcast operation to notify all nodes in the hypercube. Furthermore, there are $log_b N$ forwarding steps when the last node is reached, and every node can perform as the origin that starts a broadcast task in the hypercube and satisfy the crucial requirement. The hypercube-based broadcast scheme works as follows: A node initiates a broadcast task by sending the broadcast message to its neighbors. The message contains the edge label. Upon receiving the broadcast message, hypercube nodes can read the edge label and restrict the forwarding of the message to those links tagged with higher edge labels. As shown in Figure 4.2, Node 000 initiates a broadcast task and sends the broadcast message to all of its neighbors, which are nodes 100, 001, and 010. When node 100 receives the message, it checks the tagged dimension number on the link, which is 0. Then the node 100 forwards the message to its neighbors with higher dimensions, which are node 110 in dimension 1 and 101 in dimension 2. In addition, node 010

receives the message on dimension 1 and forwards the message to neighbor node 011 in dimension 2. Finally, node 110 receives the message at dimension 1 and forwards it to neighbor node 111 in dimension 2. The path length in this scheme can be calculated using the equation below:

$$\mathrm{L} = \frac{1}{L-1} \cdot \sum_{i=1}^{log_b N} \frac{(b-1)^{log_b N - i + 1}}{(log_b N - i)!} \cdot \prod_{j=0}^{log_b N - i} (i + j) \tag{2}$$

4.1.3 Building and Maintaining a Hypercube-based Network Topology

In this section, the algorithm for building a hypercube topology is introduced. Fig 4.3 illustrates the basic idea of the algorithm for building and maintaining a hypercube topology with nice nodes and three dimensions.



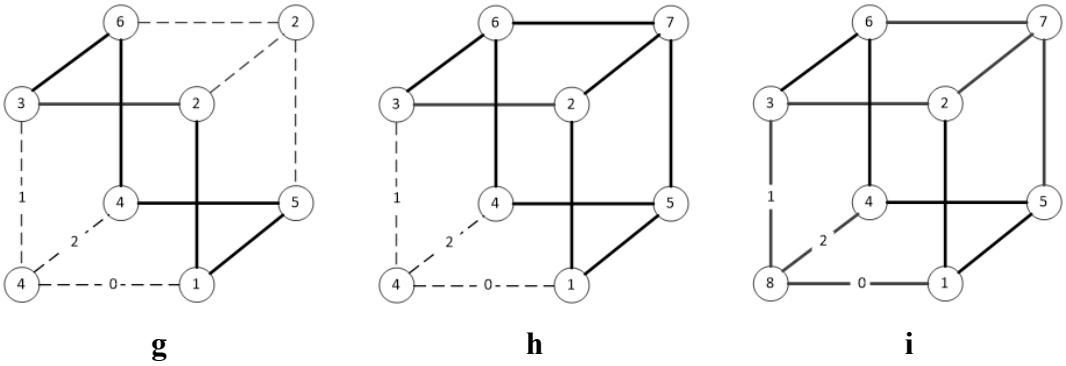


Figure 4.3. Illustration of building and maintaining a hypercube topology with 9 nodes and three dimensions

The basic principles of this algorithm are two-fold: first, the complete hypercube topology updates in the presence of arriving new node; second, when a node is removed, other nodes cover the positions of the leaving node, and prepare to give up these positions again when a new node joins. The detailed procedure is listed below:

Figure 4.3 (a): Node 1 joins the network and node 0 is performed as the proxy for adding node 1 into the hypercube topology. For node 0 has no neighbor at this stage, peer 1 is added as the 0-neighbor of node 0.

Figure 4.3 (b): Node 2 contacts peer 1 to join the network. At this time node 1 becomes the control node for adding node 2. Considering node 0 has been the 0-neighbor of node 1, node 1 opens up a new dimension for the integration of node 2 into the network, which is dimension 1. At the same, a vacant dimension is generated by node 0 on dimension 1 and node 0 will add itself as the 1- neighbor.

Figure 4.3 (c): Node 3 contacts node 0 to join the network. At this time, node 0 adds the node 3 in its first vacant dimension, which is 1, since node 0 has a 0-neighbor but no 1-neighbor. The node 3 is put on the temporary position which is used by node 0 to maintain in the hypercube.

Figure 4.3 (d): Node 4 contacts node 0 to join the network. Node 0 opens up a third dimension to add node 4 as its 2-neighbor. When node 4 is joined, it requires 3 neighbors at three dimensions. Considering that neither node 0's 1-neighbor, node 3, nor node 0's 0-neighbor, node 1, has 2-neighbor which can be add to node 4 as the neighbor node, node 3 acts as temporary 1-neighbor for node 4 and node 1 acts as temporary 0-neighbor for node 4.

Figure 4.3 (e): Node 1 is contacted to integrate the newly arriving node 5. Node 1 is still lacking a 2-neighbor, thus node 5 will be integrated on this position.

Figure 4.3 (f): Node 0 suddenly leaves. In this case, node 4 takes over node 0's position and establishes tow temporary links to the neighbors of node 0, which are node 1 and node 3.

Figure 4.3 (g): Node 4 is contacted by node 6 to integrate it as its new 1-neighbor, which is currently covered by node 3. Hence node 4 forwards the joining control to node 3. All temporary links, which are originally belong to the new position of node 6 while currently owned by node 3, are restored and passed to node 6. Additionally, node 3 adds node 6 as its new 2-neighbor.

Figure 4.3 (h): Node 7 contacts node 6 for joining the network. Node 6 add node 7 as its new 0-neighbor.

Figure 4.3 (i): Node 8 contacts node 4 for joining the network. At this stage, node 4 follows the general rule to add node 8 in its first vacant dimension, which is 2. Now node 8 covers the temporary position which is maintained by node 4 in the hypercube.

As shown in Figure 4.3, the hypercube topology is implicitly preserved in the presence of node addition or removal. This feature makes the broadcast and search algorithms do not need to be changed and every node in the hypercube still receives a broadcast message exactly once during a broadcast activity.

4.2 The Architecture of HyperCGSF

Based on the hypercube network topology introduced above and the GeoSPA model introduced in chapter 3, the HyperCGSF is designed in this study as a scalable geospatial service framework to enable the efficient discovery, composition and execution of geospatial processes persisted by multiple GeoSPAs. By utilizing the hypercube structure as the underlying network topology, the HyperCGSF possesses many advantages over existing distributed computing architectures as introduced below:

First, all GeoSPA nodes in the HyperCGSF are equivalent and no service agent owns a more prominent position than the other nodes. Consequently, any service agent in the HyperCGSF can become the entry point for the deployment and execution of a geospatial processing workflow.

Second, when a GeoSPA starts a broadcast task, the value of Δ is guaranteed to be exactly N-1 to ensure the message can be received by all N nodes in the hypercube network, regardless of the broadcasting source. This feature is of extremely critical to improve the performance because the broadcast scheme is extensively used in this research for the deployment of geospatial processing workflow.

Third, it is possible that the GeoSPA node may be deployed in less controlled public cloud computing environments. The GeoSPA node deployed on the cloud is assumed to be unreliable. It is always possible for the HyperCGSF to recover from sudden node losses due to utilizing the hypercube topology.

Finally, the HyperCGSF guarantees a complexity of $log_2 N$ for joining or removing a GeoSPA node in the network. Hence, the process of the joining and removing tasks does not inflict the overall performance of the HyperCGSF. Figure 4.4 illustrates the overall architecture of HyperCGSF.

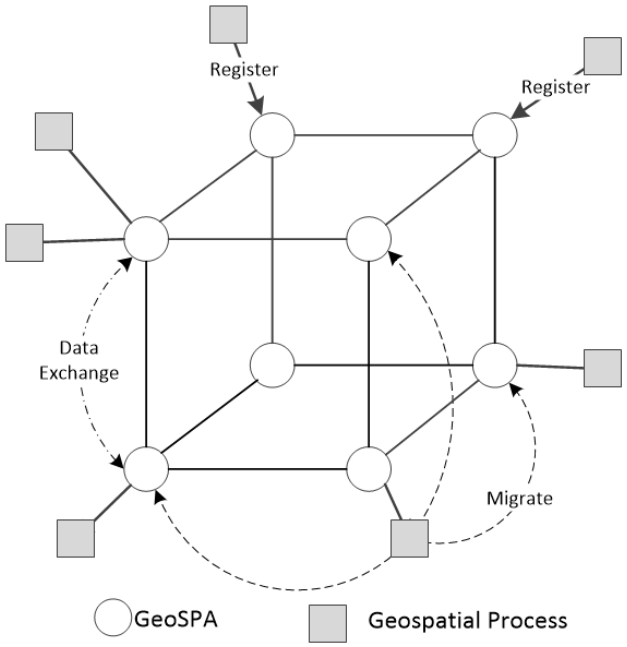


Figure 4.4. The overall architecture of the HyperCGSF

As shown in Figure 4.4, central to this framework is the binary hypercube topology for organizing an arbitrary number of available GeoSPA nodes (represented by circle in Figure 4.4). Based on the hypercube network topology introduced above, there is no centralized controller in the HyperCGSF and every GeoSPA can communicate with each other directly for data exchange. In addition, the geospatial processes can migrate among the HyperCGSF nodes for execution. One of the most important features of the HyperCGSF is to map the static job workflow specification to the dynamic cloud computing resources on the fly and coordinate multiple GeoSPAs to achieve complex geospatial processing tasks. Figure 4.5 illustrates the life cycle of the HyperCGSF-based geospatial processing service composition:

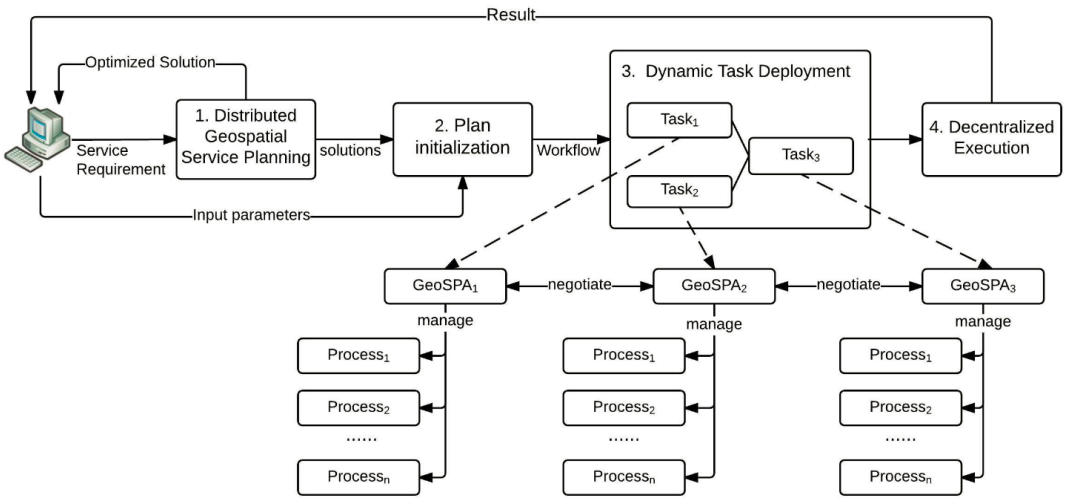


Figure 4.5. Life cycle of the HyperCGSF-based geospatial processing service composition

As shown in Figure 4.5, the life cycle of the HyperCGSF-based geospatial processing service composition can be divided into four steps: distributed geospatial service planning, plan initialization, dynamic task deployment, and decentralized execution. The details of these four steps were introduced below.

4.3 Distributed Geospatial Service Planning Algorithm (DGSPA)

Based on the GeoSPA processing service model introduced in Chapter 3, a distributed geospatial service planning algorithm called DGSPA is proposed in this section to support efficient discovery and composition of geospatial processing services distributed over the cloud. The essential criterion of the DGSPA is the solution with the smallest length has the highest priority among all of the potential solutions. The DGSPA formalizes geospatial processing service compositions into a search problem in graphic theories according to the dependence relationships among geospatial processes. The characteristics of the DGSPA are two-fold. First, the DGSPA has better scalability. This algorithm is fully distributed based on autonomous and intelligent service agents, which can find the best solution among the distributed geospatial processes. Second, DGSPA can produce solution with high quality at a low cost of communication, because the plan generated by the DGSPA has the smallest length.

4.3.1 Problem Formulation

Several notation concepts were previously defined to clearly introduce the DGSPA. For expression convenience, we use notation '$*.\#$' to express '$\#$ is a part of $*$' through the remaining of this chapter. For example, $p.DS$ represents that the DS is a part of p.

Definition 1: Service requirement

The *service requirement* can be represented using a tuple $\langle r_i, r_o \rangle$, where $\mathrm{r_i}$ is the set of input parameters that the user provides, $\mathrm{r_o}$ is the set of output parameters that the user desires.

Definition 2: Geospatial Process

The *geospatial process* (GP) is defined as tuple: $p = \langle id_p, pIn, pOut, C_p, QoS \rangle$, where id_p defines a unique ID of p, pIn and $pOut$ define the input and output parameters used by p, C_p is the identifier of an executable model, and QoS defines the quality of the geospatial service.

Definition 3: Task

A *task* is an atomic execution unit that can be performed by the GeoSPA. The task is defined as an agent-process-parameters: $t = \langle spa, p, v \rangle$, where spa is the name of the GeoSPA that performs this task, p is the GP to be performed, and v is required output parameters specifying $v \subseteq \mathrm{GetOutputs}(p)$.

Definition 4: Dependence

A GP is said to be dependent on another if the latter can help it to achieve one of its goals. As no GeoSPA has a privileged role than the other, the dependence relation between GPs enables the GeoSPA to adapt to a changing environment by taking into consideration of the other GeoSPAs in the HyperCGSF. The dependence relation among GeoSPAs is defined below.

Given two GeoSPAs denoted as $spa_i = (\text{Beliefs}_\text{i}, \text{Processes}_\text{i}, \text{Goals}_\text{i}, \text{Plans}_\text{i})$ and $spa_j = (\text{Beliefs}_\text{j}, \text{Processes}_\text{j}, \text{Goals}_\text{j}, \text{Plans}_\text{j})$, respectively. if there exists two processes $p \in Processes_\text{i}$ and $p' \in Processes_\text{j}$ satisfying $\text{GetOutputs}(p') \cap \text{GetInputs}\ (p) \neq \emptyset$, then the p of spa_i is said to be *dependent* on the p' of spa_j, which is denoted as $Dep(spa_i, p, spa_j, p', v)$, where $v = \text{GetOutputs}(p') \cap \text{GetInputs}\ (p)$.

Definition 5: Dependence Solution

Given a GeoSPA denoted as spa which has a set of dependence relations which can be expressed as $DS_p = \{Dep(spa, p, spa_i', p_i', v_i) |\ i = 1,2, \dots, n\}$ satisfying $\text{GetInputs}(p) \subseteq (r_o \cup v_1 \cup v_2 \cup \dots \cup v_n)$, then DS_p is the *dependence solution* of geospatial process p. For each element $\text{e} \in DS_p$, if $DS_p - e$ is not a dependence solution, then DS_p is called the *minimal dependence solution*, denoted as MDS_p, of geospatial process p. All MDS_p are alternative to solutions for dully matching the input parameters of the process.

Definition 6: Solution

A *solution* of the user-specified service requirement $\langle r_i, r_o \rangle$ is a set of tasks denoted as $S = \{t_i(spa, p, v) | i = 1,2, \dots, n\}$ satisfying $r_o \subseteq (t_1.v \cup t_2.v \cup \dots \cup t_n.v)$. For any task $t \in S$, if $S - t$ is not the solution for the service requirement, then S is called the *minimal cover solution*.

Definition 7: Plan`

The *plan* satisfying the user-specified service requirement can be graphically represented as $plan = (id_{plan}, Goal, S, L, t_m)$, where id_{plan} is the unique identifier of the plan, $Goal$ is the business goal that the plan achieves denoted as tuple $\langle inputs, outputs \rangle$, S is a set of minimal cover solution, and L is a set of relations among the processes in S represented as $L = \{(t_i, t_j) | t_i, t_j \in S\}$. If there is an edge connecting t_i and t_j $((t_i, t_j) \in L)$ and $t_j < t_i$, then t_j is called t_i's successor and t_i is called t_j's predecessor. The set of predecessors of t_i is denoted as $L_p(\mathrm{t_i})$, and the set of successors of t_i is denoted as $L_s(\mathrm{t_i})$. t_m is the unique identifier of the main task, i.e. the one that is always executed first.

Given a user-specified service requirement $\langle r_i, r_o \rangle$, the objective of DGSPA is to find the plan denoted as $plan$, which consists of a set of geospatial processes that distributed over the HyperCGSF as well as their execution sequence. The desired $plan$ should satisfy:

1. $GetGoalInputs(plan) \subseteq r_i$ and $r_o \subseteq GetGoalOutputs(plan)$.

2. For any geospatial process p_i without predecessor, there is $GetInputs(p_i) \subseteq GetGoalInputs(plan)$ Where the function of $GetInputs(p_i)$ returns the input parameters of the geospatial processing p_i and $\mathrm{GetGoalInputs}(plan)$ returns the input parameters defined in the goal of a $plan$.

3. For any operation $\mathrm{p_j}$ with predecessors, there is $\mathrm{re(p_j)} \subseteq (\mathrm{GetGoalInputs(plan)} \cup (\cup_{\mathrm{p_k} \in \mathrm{Pre(p_j)}} \mathrm{GetOutputs(p_k)}))$, Where the functions of $\mathrm{Pre(p_j)}$ and $\mathrm{GetOutputs(p_k)}$ returns the predecessors of the operation p_j and the output parameters of the operation template p_k, respectively.

4. $(\cup_{p_k \in O_s} GetOutputs(p_k)) \supseteq GetGoalOutputs(plan)$, Where the functions of $\mathrm{GetGoalOutputs}(plan)$ return the output parameters of the plan.

In the following paragraphs, we introduced how to apply the DGSPA to get the desired solutions for the user-specified service requirement in a distributed computing environment.

4.3.2 DGSPA Steps

The GeoSPA performing on behalf of the service consumer to communicate with the HyperCGSF is called the *manager agent* denoted as spa_m, while the agent which is responsible for processing the requests sent by spa_m is called the *worker agent* denoted as spa_i where $i = 1,2, \dots, n$ and n is the number of GeoSPA nodes in HyperCGSF. Upon receipt of a service requirement $\langle r_i, r_o \rangle$, the spa_m first broadcasts a $CFP(spa_m, r_i, r_o)$ message in the HyperCGSF to notify other worker agents of the incoming service requirement. When spa_i receives the CFP message, Algorithm 1 is executed as represented below:

Algorithm 1:

Input: $CFP(spa_m, r_i, r_o)$

```
1 begin
2   C_p ← GetProcesses(spa_i);
3   foreach process p ∈ C_p do
4     if GetOutputs(p) ∩ r_o ≠ ∅ then
5       send PROPOSE(spa_i, p, GetOutputs(p) ∩ r_o) to spa_m;
6     end
7   end
8 end
```

The function $GetProcesses(spa_i)$ returns a collection of geospatial processes (C_p) persisted by spa_i (Line 2, Algorithm 1). Then spa_i checks if there is a geospatial process p whose output parameters match r_0 (Line 3-8, Algorithm 1). If it is true, then spa_i replies a message $PROPOSE(spa_i, p, GetOutputs(p) \cap r_0)$ to spa_m. Once spa_m receives all proposed geospatial processes returned from other worker agents in the HyperCGSF, it generates a minimal cover solution denoted as MCS based on the proposed geospatial processes, and Algorithm 2 is executed as demonstrated below.

Algorithm 2:

```
1 begin
2   if MCS ≠ ∅ then
3     foreach t_i(spa, p, v) ∈ MCS do
4       send REQUEST(spa_m, null, spa, p, v, r_i) to spa
        S_req = S_req ∪ REQUEST(spa_m, null, spa, p, v, r_i);
5     end
6   else
7     Notify the service consumer that the service requirement can not be
      achieved.
8   end
9 end
```

If the MCS is not empty, the spa_m begins to retrieve every task t_i in MCS and send $REQUEST(spa_m, null, spa, p, v, r_i)$ message to $t_i.spa$ to query if the geospatial process p generating r_i is feasible; otherwise, the spa_m notifies the service consumer that the service requirement cannot be achieved. When $t_i.spa$ receives the $REQUEST(spa_m, null, spa, p, v, r_i)$ message, Algorithm 3 is executed as demonstrated below.

Algorithm 3:

Input: $REQUEST(spa', p', spa, p, v, r_i)$

```
1  begin
2      if p.status = "unexplored" then
3          CDS_p ←choose a cover dependence solution of geospatial process p;
4          if GetInputs(p) ⊈ r_i and CDS_p = ∅ then
5              p.explored ← "expolored";
6              p.feasible ← false;
7              send CONFIRM(spa', p', spa, p, v, false, null) to spa';
8          else if GetInputs(p) ⊆ r_i then
9              p.explored ← "expolored";
10             p.feasible ← true;
11             create a new task: t(spa, p, v);
12             Send CONFIRM(spa', p', t(spa, p, v), true, null) to spa';
13         else
14             p.explored ← "exploring";
15             foreach Dep(spa, p, spa'', p'', u) ∈ CDS_p do
16                 send REQUEST(spa, p, spa'', p'', u, r_i) to spa'';
17                 S_req_p = S_req_p ∪ REQUEST(spa, p, spa'', p'', u, r_i);
18             end
19         end
20     else
21         if p.feasible = true then
22             L ← get the realtion set of geospatial process p;
23             t(spa, p, v) ← create a new task;
24             send CONFIRM(spa', p', t(spa, p, v), true, L) to spa';
25         else
26             send CONFIRM(spa', p', t(spa, p, v), false, null) to spa';
27         end
28     end
29 end
```

As shown in Algorithm 3, the worker agent first checks if the requested geospatial process p has been checked before (Line 2, Algorithm 1). The "unexplored" of p represents that the worker agent never receives such a $REQUEST$ message about this geospatial process and the search for p has not been carried out before. While the "explored" means that the search for the process p has been performed before, and a $CONFIRM(spa', p', t(spa, p, v), flag, L)$ message is returned to spa', where $flag$ indicates if the process is feasible, and L is the relation set of p.

If the status of the process p is 'unexplored', three conditions are considered:

1. The input parameters of process p cannot be fully matched by r_i and the set of minimal cover dependence solutions is empty, which indicates that the process p can't be performed using the user-specified input parameters. Then the algorithm is end and the worker agent sends a message $CONFIRM\ (spa', p', t(spa, p, v), false, null)$ to spa' (Line 4-7 in Algorithm 3).

2. The user-specified input data in r_i, can be fully matched by the input parameters of the process p. In this situation, as the input data of the geospatial process p is directly satisfied by user-specified input data in r_i, which mean the geospatial process can meet the service requirement. Then the algorithm is end and the service agent sends the message $CONFIRM(spa', p', t(spa, p, v), true, tp)$ to spa, where tp refers to the sequence of processes (Line 14-12 in Algorithm 3).

3. The input parameters of the process p cannot be fully matched by user-specified input data in r_i, but minimal cover dependence solutions set is not empty. Under this circumstance, the worker agent first set the status of process p to 'explored' which means the processes has been visited. Then a minimal cover dependence solution set which has the minimal length is chosen by this service agent. Finally, every element in the minimal cover dependence solution set is searched by sending them a $REQUEST$ message to repeat the planning algorithm (Line 14-18 in Algorithm 3).

Upon receipt of the $CONFIRM(spa', p', t(spa, p, v), flag, tp)$ message, the GeoSPA executes Algorithm 4 shown below:

Algorithm 4:

Input: $CONFIRM(spa, p, t(spa', p', v), flag, L)$

begin
  $S_{resp_p} \leftarrow$ get the set of responses about geospatial process p;
  $S_{req_p} \leftarrow$ get the set of requests about geospatial process p;
  $S_{resp_p} = S_{resp_p} \cup CONFIRM(spa, p, spa', p', v, flag, L)$
  if $flag = true$ **then**
    $L = L \cup < t(spa', p', v), t(spa, p, v) >$;
  end
  if $|S_{resp_p}| = |S_{req_p}|$ **then**
    if *all flag bits of the CONFIRM messages in* S_{resp_p} *are true* **then**
      $p.status \leftarrow$"explored";
      $p.feasible \leftarrow$"true";
      foreach $REQUEST(spa'', p'', spa, p, v') \in S_{req_{p''}}$ **do**
        create a new task: $t(spa, p, v')$;
        send $CONFIRM(spa'', p'', t(spa, p, v'), true, L)$ to spa''
      end
    else if $DS_p \neq \emptyset$ **then**
      choose a minimal cover dependence solution D_s with the minimum length from S;
      $S_{req_p} = \emptyset$;
      $S_{resp_p} = \emptyset$;
      foreach $Dep(spa, p, spa''', p''', u) \in D_s$ **do**
        send $REQUEST(spa, p, spa''', p''', u, r_i)$ to spa''';
        $S_{req_p} = S_{req_p} \cup REQUEST(spa', p', spa''', p''', u, r_i)$;
      end
    else
      $p.status \leftarrow$"explored";
      $p.feasible \leftarrow$"false";
      foreach $REQUEST(spa'', p'', spa, p, v, r_i) \in S_{req_p}$ **do**
        send $CONFIRM(spa'', p'', t(spa, p, v), false, null)$ to spa'';
      end
    end
  end
end

When the spa_m receives the $CONFIRM(spa_m, null, t(spa', p', v), flag, L)$ message, Algorithm 5 shown below is executed.

Algorithm 5:

Input: $CONFIRM(spa_m, null, t(spa, p, v), flag, L)$

```
1  begin
2      S_req ← get the set of request;
3      S_resp ← get the set of response;
4      S_resp = S_resp ∪ CONFIRM(spa', p', t(spa, p, v), flag, L if flag = true then
5          plan.S = plan.S ∪ t(spa, p, v);
6          plan.L = plan.L ∪ L;
7      end
8      if |S_resp| = |S_req| then
9          if the currently searching minimal cover solution is feasible then
10             notify the service consumer that the plan has been found.
11         else
12             notify the serivce consumer that service requirement can not be
               achieved.
13         end
14     end
15 end
```

One scenario of DGSPA is illustrated below to describe how the algorithm works. As shown in Table 4.1, five GeoSPAs (denoted from spa_1 to spa_5) participate as the service provider. Each GeoSPA maintains one geospatial process described as $Process$: $input \longrightarrow output$, and a dependence solutions (DS_p). The variables with bold character in Table 4.1 represent the sharing variables between two processes. Another GeoSPA denoted as spa_m participates on behalf of the service consumer to communicate with the other five agents.

Table 4.1: The process and corresponding dependence solution for each GeoSPA

Agent	$\boldsymbol{Process}$: $\boldsymbol{input \longrightarrow output}$	$\boldsymbol{DS_p}$
spa_1	$p_1: (x_{11}, x_{12}, x_{13}) \longrightarrow (\boldsymbol{y_{11}}, y_{12})$	$\emptyset$
spa_2	$p_2: (x_{21}, x_{22}) \longrightarrow (\boldsymbol{y_{21}})$	$\emptyset$
spa_3	$p_3: (x_{31}, \boldsymbol{y_{11}}, \boldsymbol{y_{21}}) \longrightarrow (y_{31}, \boldsymbol{y_{32}})$	$Dep(spa_3, p_3, spa_1, p_1, (y_{11}))$ $Dep(spa_3, p_3, spa_2, p_2, (y_{21}))$
spa_4	$p_4: (x_{41}, x_{42}, \boldsymbol{y_{32}}) \longrightarrow (\boldsymbol{y_{41}})$	$Dep(spa_4, p_4, spa_3, p_3, (y_{32}))$
spa_5	$p_5: (x_{51}, \boldsymbol{y_{41}}) \longrightarrow (y_{51})$	$Dep(spa_5, p_5, spa_4, p_4, (y_{41}))$

Based on Table 4.1, given the user-specified service requirement denoted as $\langle(x_{51}, x_{41}, x_{31}, x_{21}, x_{11}, x_{12}), (y_{51})\rangle$, the objective of the DGSPA is to generate a $plan$ that can fulfill this service requirement by outputting y_{51}. One possible $plan$, which consists of five tasks, was shown in Figure 4.6. As the start tasks of this $plan$, the t_1 and t_2 has no predecessor and can be performed directly and generate y_{11} and y_{21}, respectively. y_{11} and y_{21} can then be used by t_3 as the input parameters to generate y_{32}, which then can be transmitted to t_4 as one input parameter. Finally, t_4 generates y_{41} and sends it to t_5, which is the end task generating the desired output y_{51}.to the service consumer.

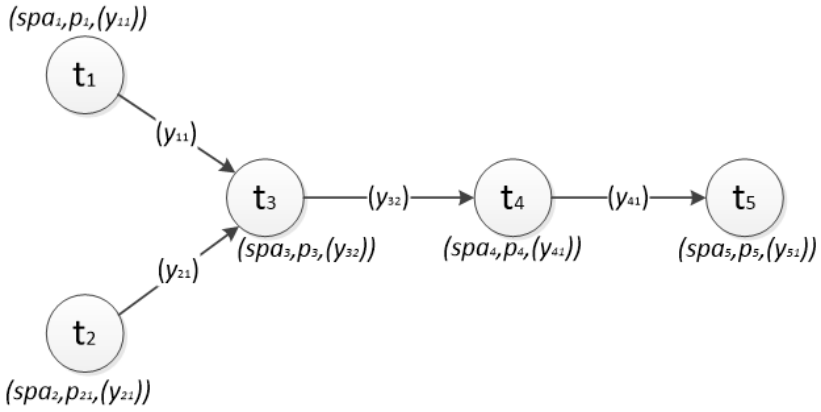


Figure 4.6. The desired plan for user-specified service requirement

Figure 4.7 depicts the UML sequence diagram of using DGSPA to derive the plan shown in Figure 4.6 above. The numbers in Figure 4.7 indicates the step number followed by the message type. For example, at the first step of DGSPA, the spa_m broadcasts a CFP message to every worker node in the HyperCGSF for the proposal of service requirement, and only spa_5 gives a response to this request. The interaction between the GeoSPAs is conducted through a generic request-response structure, which is briefly presented in following eight steps:

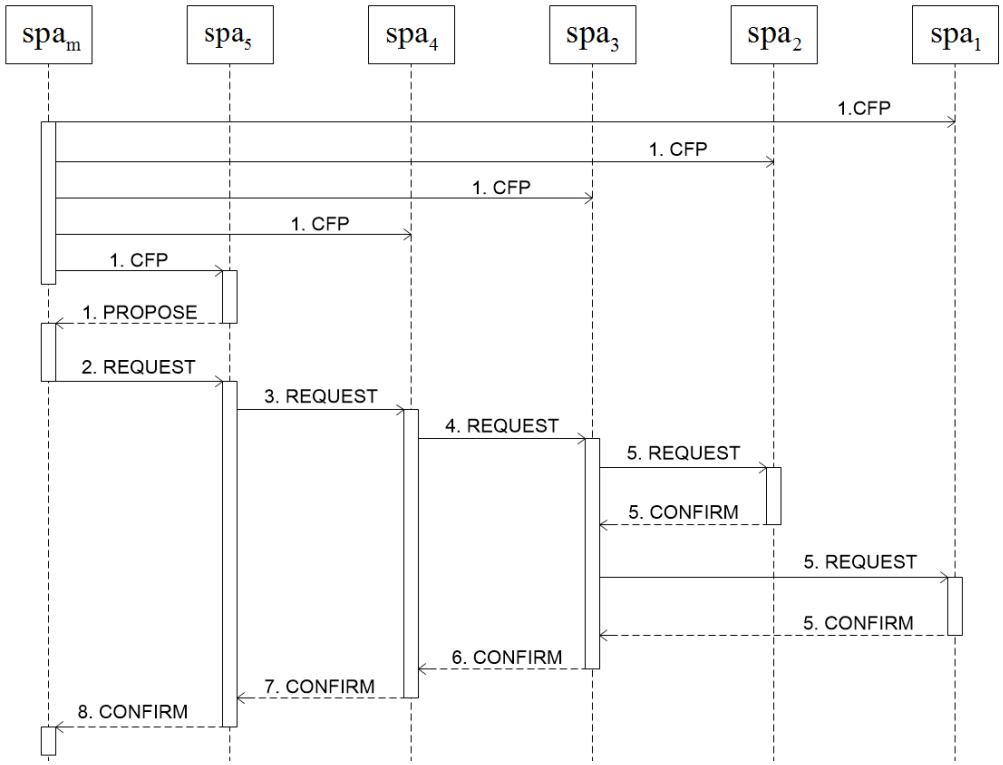


Figure 4.7. The sequence diagram of entire lifecycle of DGSPA

Setp1: Upon receipt of a service requirement, the spa_m on behalf of the service consumer broadcasts a $CFP(spa_m, r_i, r_o)$ to notify all GeoSPAs for proposal. The spa_5 checks its processes and gets the process p_5 whose output parameters has intersection with r_o. Then spa_5 returns a message $PROPOSE(spa_5, p_5, \langle y_{51} \rangle)$ to spa_m.

Step2: Upon receipt of the $PROPOSE$ messages, spa_m sends the message $REQUEST(spa_m, null, spa_5, p_5, \langle y_{51} \rangle)$ to spa_5 for the plan which contains the completed workflow.

Step3: When spa_5 receives the $REQUEST(spa_m, null, spa_5, p_5, \langle y_{51} \rangle)$ message, it first checks if the r_i contains all the needed input parameters of p_5. If it is false, spa_5 derives the dependence solution of p_5 based on Table 4.2 ($Dep(spa_5, p_5, spa_4, p_4, \langle y_{41} \rangle)$) and sends $REQUEST(spa_5, p_5, spa_4, p_4, \langle y_{41} \rangle)$ to spa_4 to check if it is feasible. This similar operation is performed in step 4.

Step4: A similar operation carried out in Step 3 is performed.

Step 5: When spa_1 receives $REQUEST(spa_3, p_3, spa_1, p_1, \langle y_{12} \rangle)$, it finds that there is no dependence solution of p_1, but an intersection between r_i and the input set of p_1. Then spa_1 will create a task denoted as $t(spa_1, p_1, \langle y_{12} \rangle)$ and send the $CONFIRM(spa_3, p_3, t(spa_1, p_1, \langle y_{12} \rangle), true, null)$ to spa_3 to notify that the process p_1 can be performed by spa_1. The similar operation is conducted on spa_2.

Step 6: Upon receipt of the $CONFIRM$ message, spa_3 first creates a task pair collection denoted as L which consists of $\langle \mathrm{t}(spa_1, p_1, y_{12}), t(spa_3, p_3, y_{32}) \rangle$ and $\langle \mathrm{t}(spa_2, p_2, y_{21}), t(spa_3, p_3, y_{32}) \rangle$. Then a new task denoted as $t(spa_3, p_3, \langle y_{32} \rangle)$ is created. Finally, spa_3 send the $CONFIRM(spa_4, p_4, t(spa_3, p_3, \langle y_{32} \rangle), true, L)$ message to spa_4..

Step 7: The similar operation carried out in Step 6 is performed.

Step 8: When spa_5 receives the $CONFIRM$ message, it will check if the flag is true. If it is true, the spa_5 will return the $plan = (id_{plan}, Goal, T, L)$ to spa_{m}.

4.3.3 DGSPA Complexity Analysis

As an important part of a computational complexity theory, the algorithm analysis was conducted in this section to evaluate the efficiency of DGSPA. The amount of messages required to run the DGSPA is used here to evaluate the algorithm complexity of the DGSPA. For the HyperCGSF system with n service agents and r dependence relations, the number of messages required to run the DGSPA for one time can be estimated by the following two steps:

1. There are totally n CFP message. Generally, since the number of geospatial processes whose output parameters can partially or fully math the user-specified output parameters in the service requirement could be very small, it can be estimated that the amount of $PROPOSE$ messages is far

less than that of CFP messages. Therefore, the amount of the messages needed in this stage is about n.

2. In the worst case, all the alternative solutions need to be searched to resolve a complex geoscience problem, which means or dependence relations will be checked to get the needs geospatial processes. Therefore, the amount of $REQUEST$ messages can be estimated as r and the number of corresponding $CONFIRM$ messages is r too.

Based on the analysis above, the total number of messages required for on the DGSPA task is about $n + 2r$ in the worst case.

4.4 Decentralized Orchestration of Plan

4.4.1 Plan Initialization

Each DGSPA-derived plan needs to be initialized for execution. The plan generated through DGSPA is called the abstract plan, which is not able be executed directly by the HyperCGSF because the processes in the abstract plan do not refer to any GeoSPA for execution. In this case, the service consumer should specify required input parameters in order to make it executable. In this research, the Embedded WPS Request Document (EWRD) was designed to facilitate service consumer to construct the executable plan. Much like how functions can call other functions, the WPS standard has the native ability to chain geospatial processes, which means that one process can be directly used as the input parameter of another process. Through EWRD, many complex functions can thus be combined into a single powerful request. Figure 4.8 demonstrates an example EWRD which describes a widely used raster-based overlap operation named ‘ras:Overlap’. Based on Figure 4.8, this operation contains two embedded WPS processes named ‘geoms:CominedRAT’ and ‘geoms:InfraredDifferenceDustIndex’, respectively.

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:Execute version="1.0.0" service="WPS">
    <ows:Identifier>geoms:Overlap</ows:Identifier>
    <wps:DataInputs>
        <wps:Input>
            <ows:Identifier>coverage1</ows:Identifier>
            <wps:Reference mimeType="application/netcdf"
                xlink:href="http://geoserver/wps" method="POST">
                <wps:Body>
                    <wps:Execute version="1.0.0" service="WPS">
                        <ows:Identifier>geoms:CombinedRAT</ows:Identifier>
                        <wps:DataInputs>
                        <wps:ResponseForm>
                    </wps:Execute>
                </wps:Body>
            </wps:Reference>                                  Embedded WPS1
        </wps:Input>
        <wps:Input>
        <wps:Input>
            <ows:Identifier>coverage2</ows:Identifier>
            <wps:Reference mimeType="application/netcdf"
                xlink:href="http://geoserver/wps" method="POST">
                <wps:Body>
                    <wps:Execute version="1.0.0" service="WPS">
                        <ows:Identifier>geoms:InfraredDifferenceDustIndex</ows:Identifier>
                        <wps:DataInputs>
                        <wps:ResponseForm>
                    </wps:Execute>
                </wps:Body>
            </wps:Reference>                                  Embedded WPS2
        </wps:Input>
        <wps:Input>
        <wps:Input>
    </wps:DataInputs>
    <wps:ResponseForm>
</wps:Execute>
```

Figure 4.8. Embedded WPS request demo of the 'Overlap' operation

Compared with the traditional BPEL-based service composition approaches, the most advantage of using the EWRD is that a geospatial process produces some output which will become the input of the next process, resulting in a processing pipeline that can solve complex spatial analysis with a single HTTP request. Furthermore, because the EWRD is fully WPS-based and all of the WPS processes are involved and connected in a single WPS document, the geospatial processing service providers no longer need a central controller (e.g. BEPL engine) for workflow execution. However, the disadvantage of using the EWRD is that users are not able to set complex operations such as using the 'if' statement to control the sequence of process execution.

Upon receipt of the EWRD, the GeoSPA first performs a syntactic validation of the document. If the format and structure is good, then the GeoSPA decomposes the EWRD into its constituent geospatial process p including its input/output variables. Each process variable v_i takes the form of 4-tuple:

$$\langle id_{v_i}, v, T, MIME \rangle$$

Where id_{v_i} is the unique identifier of the variable, v is the holder of the variable value, T is the type of v indicating how to derive the variables and expressed by a set: $T \in \{Literal, Text, Reference, Subprocess\}$. The $Literal$ type can be any character string such as float, date, etc., normally described as primitive datatype in the W3C XML Schema standard (Biron and Malhotra, 2004); the $Text$ type is a structured document which needs to be parsed based on some rules, e.g. GML and KML; the $Reference$ type is always described as a Universal Resource Locator (URL) of the resource, which indicates the address of the input parameter; the $Subprocess$ means that the input parameter comes from its embedded geospatial process. The $MIME$ represents the Multipurpose Internet Mail Extensions (MIME) type of v.

4.4.2 Dynamic Task Dispatching Algorithm (DTDA)

After initializing the plan through the EWRD, each of its tasks needs to be dispatched to one available the GeoSPA, called worker agent, for performance. It is possible that one worker agent in the HyperCGSF can perform more than one task at a time. This study proposed the DTDA for dispatching geospatial processes of a plan to various the GeoSPA for performance. The main goal of the DTDA is to optimize the allocation of the task instances of a plan in HyperCGSF system and finally improve the performance of plan execution. To balance the workload, the Least Recently Used (LRU; Pantazoglou et al., 2014) algorithm was used in the DTDA. Based on the LRU algorithm, each GeoSPA node in HyperCGSF maintains a record of the recent visited time by other agents. When the GeoSPA needs to deliver the dispatching request to its neighbors, it first picks up the latest visited GeoSPA, and then sends the dispatching request to it for further dispatching operations. The detail of DTDA is introduced below.

First, before dispatching task to worker agent, the manager agent spa_m first initializes the *plan execution session* (PES) for each plan, which can be expressed by the 4-tuple:

$$\langle id_{session}, id_{plan}, E_p, spa_m \rangle$$

Each PES instance is distinguished by the unique identifier denoted as $id_{session}$, while it is associated with the plan to be executed through the plan identifier id_{plan}. In addition, it includes a table denoted as $E_{plan} = \{(task_i, spa_i)\}_{i=1}^{|plan.S|}$, which maps the task of a plan to the endpoint addresses of worker agent.

In order to properly fill table E_p, a dispatching work is carried out based on Algorithm 6 given next.

Algorithm 6:

Input: $REQUEST(dispatch(id_{session}, id_{plan}, E_p, spa_m))$

```
1  begin
2      dispatch_complete=true
3      spa_L ← get local endpoint address
4      foreach (task, spa) ∈ E_p do
5          if spa = null then
6              if task.p can be performed by spa_L then
7                  spa ← spa_L
8                  Update E_p
9                  N ← get all hypercube neighbors
10                 Broadcast timestamp of last use to all nodes ∈ N
11                 dispatch_complete=false
12                 break
13             end
14         end
15     end
16     if dispatch_complete=false then
17         w ← get LRU neighbor in lowest dimension
18         spa' ← get endpoint address from w
19         Send REQUEST(dispatch(id_session, id_plan, E_p, spa_m)) to spa'
20     else
21         Send CONFIRM(dispatch(id_session, id_plan, E_p, spa_m)) notification to
           spa_m
22     end
23 end
```

The spa_m first selects its LRU hypercube neighbor spa_i in the lowest possible dimension, and then sends a dispatching request containing the updated plan execution session to the spa_i. Upon receipt of the $REQUEST$ message, the spa_i traverses every task in the plan to determine which process can be performed locally and update the E_p (Line 4-15, Algorithm 6). Finally, the spa_i checks if all geospatial processes of a plan have been assigned to an available GeoSPA for performance. If it is true, then the spa_i sends a message $CONFIRM$ $\left(dispatch\left(id_{session,}id_{plan,}E_{p,}spa_m\ \right)\ \right)$ to spa_m . Otherwise, the same set of steps are performed each time a node is visited during the dispatching procedure, until all entries in E_p are properly set.

One possible scenario is introduced below to clearly describe how the DTDA works. As shown in Table 4.2, the spa_{101} was configured as the geospatial processing service provider agent, which maintains five geospatial processes (denoted from p_1 to p_5) similar to the one in section 4.2.2. The variables with bold character in Table 4.1 represent the sharing variables between two processes. Figure 4.9 illustrates the DGSPA-derived abstract plan indicating the tasks as well as their relations.

Table 4.2: The Processes and corresponding dependence solution for each GeoSPA

Service Provider	$\boldsymbol{Process}$: $\boldsymbol{input}$ $\rightarrow$ $\boldsymbol{output}$
spa_{101}	p_1: $(x_{11}, x_{12}, x_{13}) \rightarrow (\boldsymbol{y_{11}}, y_{12})$
spa_{101}	p_2: $(x_{21}, x_{22}) \rightarrow (\boldsymbol{y_{21}})$
spa_{101}	p_3: $(x_{31}, \boldsymbol{y_{11}}, \boldsymbol{y_{21}}) \rightarrow (y_{31}, \boldsymbol{y_{32}})$
spa_{101}	p_4: $(x_{41}, x_{42}, \boldsymbol{y_{32}}) \rightarrow (\boldsymbol{y_{41}})$
spa_{101}	p_5: $(x_{51}, \boldsymbol{y_{41}}) \rightarrow (y_{51})$

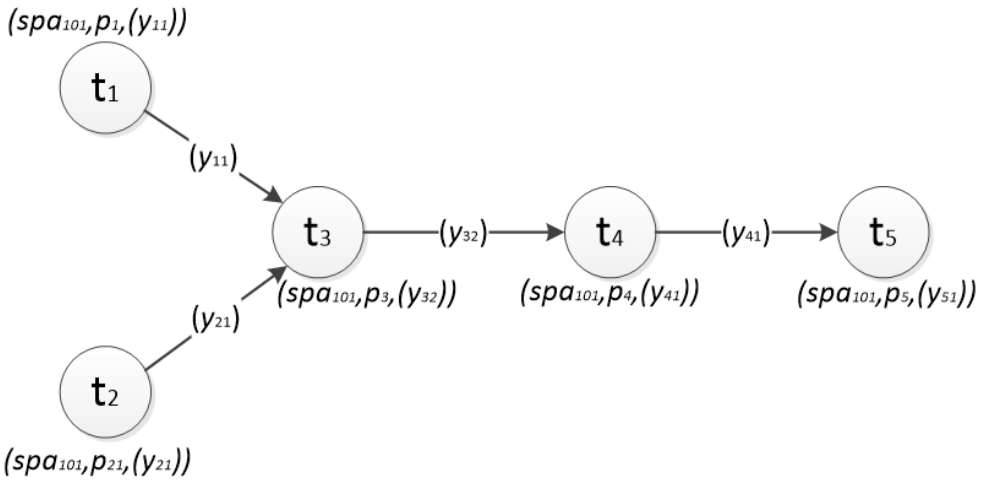
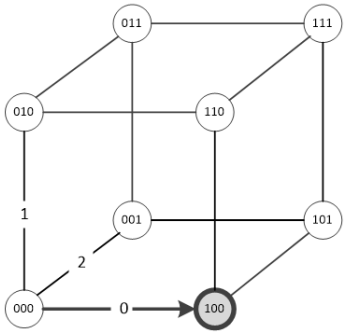


Figure 4.9. The DGSPA-derived abstract plan for user-specified service requirement

Once the abstract plan shown in Figure 4.9 was ready, the DTDA was applied to dispatch each geospatial processes to distributed GeoSPAs over the HyperCGSF for execution. Figure 4.10, read from top to bottom, demonstrates the sequence in which the GeoSPA is visited upon receipt of the dispatching request sent by spa_{000} during the execution of DTDA. As shown in Figure 4.10, the tables on the right side show the value of E_p at current stage. As it can be seen, the DTDA can evenly map each task to all available GeoSPAs as the worker agent while taking into account their sequence of use upon distribution of workload. At each step, one GeoSPA node was visited based on the LRU algorithm. If there was a geospatial process that the visited GeoSPA was able to perform, then a new record was added to the table E_p. For example, at the first step (Figure 4.10(a)), the spa_{100} was visited and the geospatial process p_1 was dispatched to it for execution. At the same time, a new record was added to table E_p. Then the dispatching request was forwarded to next GeoSPA together with the E_p. The similar operation was performed until all geospatial processes in the abstract plan were dispatched to corresponding GeoSPA for execution.

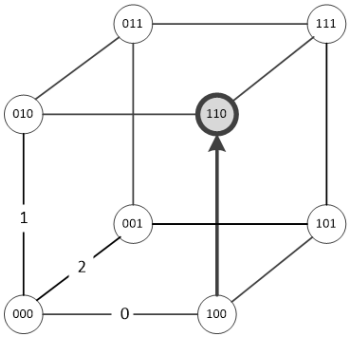
(a)



E_p

Provider	Process	Output	Executor
spa_{101}	p_1	y_{11}	spa_{100}

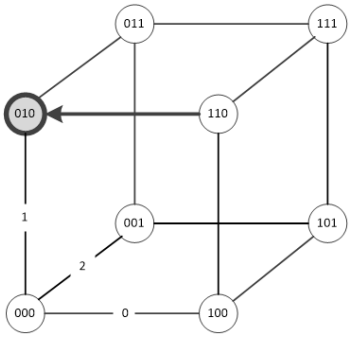
(b)



E_p

Provider	Process	Output	Executor
spa_{101}	p_1	y_{11}	spa_{100}
$\boldsymbol{spa_{101}}$	$\boldsymbol{p_2}$	$\boldsymbol{y_{21}}$	$\boldsymbol{spa_{110}}$

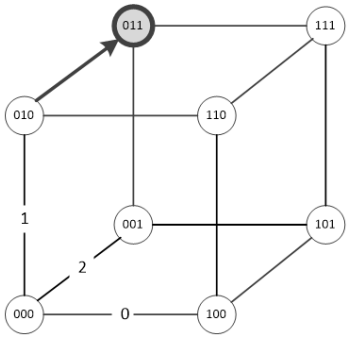
(c)



E_p

Provider	Process	Output	Executor
spa_{101}	p_1	y_{11}	spa_{100}
spa_{101}	p_2	y_{21}	spa_{110}
$\boldsymbol{spa_{101}}$	$\boldsymbol{p_3}$	$\boldsymbol{y_{32}}$	$\boldsymbol{spa_{010}}$

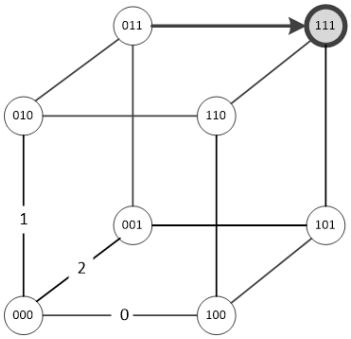
(d)



E_p

Provider	Process	Output	Executor
spa_{101}	p_1	y_{11}	spa_{100}
spa_{101}	p_2	y_{21}	spa_{110}
spa_{101}	p_3	y_{32}	spa_{010}
$\boldsymbol{spa_{101}}$	$\boldsymbol{p_4}$	$\boldsymbol{y_{41}}$	$\boldsymbol{spa_{011}}$

(e)



E_p

Provider	Process	Output	Executor
spa_{101}	p_1	y_{11}	spa_{100}
spa_{101}	p_2	y_{21}	spa_{110}
spa_{101}	p_3	y_{32}	spa_{010}
spa_{101}	p_4	y_{41}	spa_{011}
$\boldsymbol{spa_{101}}$	$\boldsymbol{p_5}$	$\boldsymbol{y_{51}}$	$\boldsymbol{spa_{111}}$

Figure 4.10. Dispatching sequence and results of task actors for the execution of plan

4.4.3 Decentralized Execution of Plan

As soon as the stage of dynamic task dispatching is finished, the manager agent retrieves from table E_p the endpoint address of the worker agents and sends to these agents the request containing the processing identifier and the plan execution session tuple to start the execution. During the execution, the geospatial process can migrate from one worker agent to another agent and executes in a decentralized manner. This paradigm has many advantages. First, the concurrent tasks can be forwarded to different sites for execution, which achieves real parallelism. Second, the geospatial tasks can be evenly distributed over the worker agents, which increases the flexibility and performance of the system. Lastly, if a site needs to leave, it can transfer its tasks to other sites to keep the system stable. Algorithm 7, which is presented below, shows the detailed procedure of plan execution.

Algorithm 7:

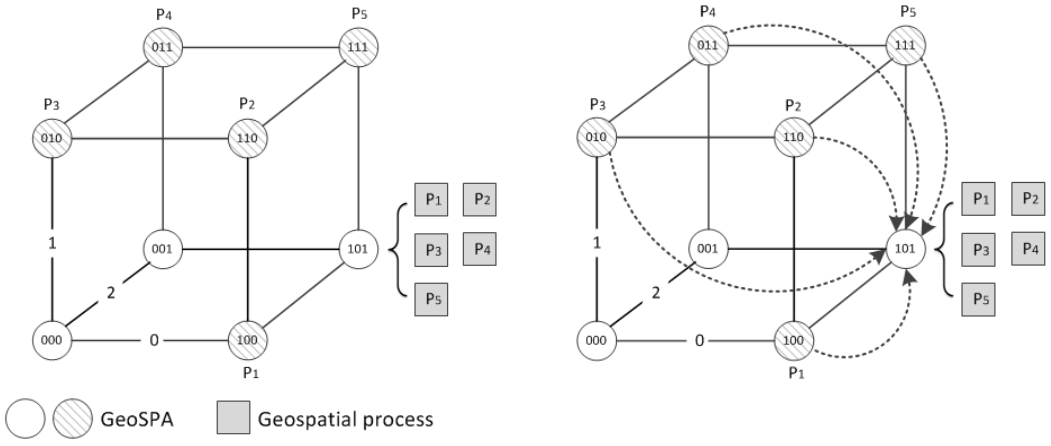
Input: $REQUEST(Execute(id_{session}, id_{plan}, E_p, spa_m)$

```
1  begin
2      plan ←Get plan tuple based on id_plan
3      task(spa, p, v) ←Get the task to be performed by this site based on E_p
4      p_i ← null
5      if spa = spa_i then
6          p_i ← task.p
7      else
8          Send REQUEST(Migrate(spa, p, spa_i)) to spa for process migration.
9          p_i ← p returned by spa
10     end
11     L_p ← get the set of task's predecessor from plan.L
12     L_s ← get the set of task's successor from plan.L
13     S_input ← create the collection of input parameters of p_i
14     foreach id_v ∈ p_i.pIn do
15         T ← get variable type of id_v
16         MIME ← get MIME type of id_v
17         if T = subprocess then
18             task'(spa, p, v) ← get the predecessor of task that offer v from L_p
19             spa' ← get endpoint address of GeoSPA from E_p for task'
20             v ← read task'.v from spa'
21         else
22             v ← read value directly
23         end
24         Process v based on MIME type
25         add v into S_input
26     end
27     Execute process p with S_input
28     v_i=get the subset of output generated by p_i based on task.v
29     if L_s ≠ null then
30         foreach task''(spa, p, v) ∈ L_s do
31             spa'' ← get endpoint address of GeoSPA from E_p for task''
32             write v_i to spa''
33         end
34     else
35         Send CONFIRM(id_session, id_plan, v_i) to spa_m
36     end
37 end
```

When spa_i receives the $REQUEST(Execute(id_{session}, id_{plan}, E_p, spa_m))$ sent from the manager agent spa_m, it first traverses each row in E_p to pick up the task(spa, p, v) dispatched to it (Algorithm 7, Line 2-3). If the endpoint address of task.spa is the same as spa_i, spa_i creates a geospatial process p_i from its

knowledge-base, otherwise, spa_i sends a request to task.spa for process migration (Algorithm 7, Line 5-10). Once the geospatial process p_i is ready, the spa_i retrieves two task sets, the predecessor set (L_p) and the successor set (L_s), from the dispatched task, as well as an input parameter set (S_{input}) of p_i (Algorithm 7, Line 11-13). Then spa_i begins to read the input parameters based on the input parameter type and the MIME type. If the parameter type is '*subprocess*', then spa_i retrieves the endpoint address of the node that is responsible to generate this parameter from L_p and waits for its response (Algorithm 7, Line 14-26). When all of the required input parameters are ready, spa_i begins to run the process and to write the value of all variables (v_i) in task.v to the nodes in charge of all tasks in L_s. If there is no process in L_s, which means the process is the last process of the plan, then spa_i sends a $CONFIRM(id_{session}, id_{plan}, v_i)$ message to the manager agent to notify the plan execution has finished successfully (Algorithm 7, Line 22-29).

Figure 4.11 illustrates the procedure of the decentralized execution of plan based on Algorithm 7. The $GeoSPA_{101}$ plays the role of geospatial process provider agent, while $GeoSPA_{100}$, $GeoSPA_{110}$, $GeoSPA_{010}$, $GeoSPA_{011}$, and $GeoSPA_{111}$ play the role of process worker agents (Figure 4.11(a)).



(a) (b)

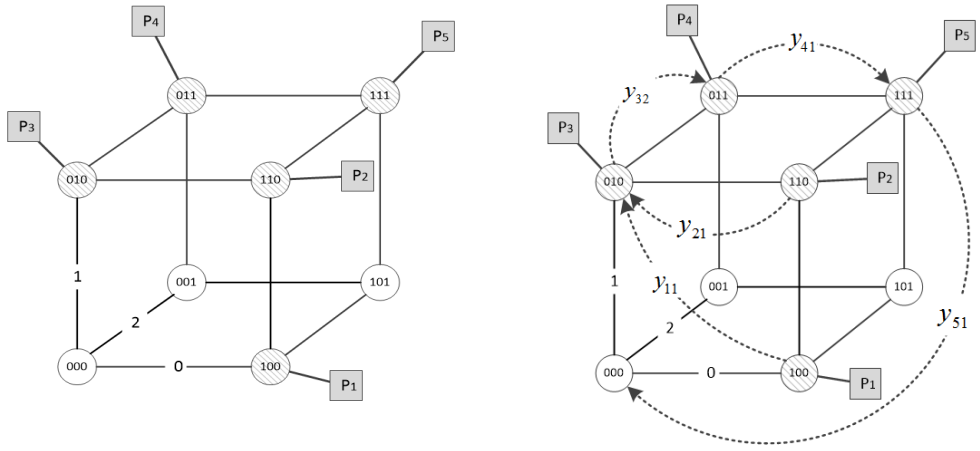


Figure 4.11. Illustration of decentralized plan execution

Based on the dynamic task dispatching result shown in Figure 4.11, at the beginning of plan execution, every worker agent requests the migration of geospatial process which has been assigned to it from $GeoSPA_{101}$ (Figure 4.11(b)). Upon receipt of the geospatial processing migration request, the geospatial processing service provider agent creates a new instance of requested geospatial processes, which then migrates to requesting a worker agent (Figure 4.11(c)). Then, every worker agent executes the process and cooperates with other worker agents to achieve the execution of the plan in a decentralized manner based on Algorithm 7. When a process execution completes successfully, the worker agent transmits the desired output variable to the process's immediate successor (Figure 4.11(d)). For example, in Figure 4.11(d), the $GeoSPA_{100}$ performs p_1 and returns the desired output parameter y_{11} to its immediate successor $GeoSPA_{010}$, where p_3 is executed with y_{11} as one of its input parameters.

4.5 Monitoring Process Execution and Fault Tolerance

4.5.1 Monitoring Process Execution

As the execution of geospatial process is always time-consuming, the ability to monitor the status of deployed geospatial process is a critical user requirement. The

HyperCGSF offered mechanism for users to keep tracking of the execution progress during the whole lifecycle of a plan. The service consumer can send a '*GetStatus'* request to the manager node to get the current status of plan execution. In doing so, the manager node checks all of the endpoint addresses from the table E_p in the plan execution session and sends to them a request containing the identifier of plan execution session $id_{session}$ and process identifier id_p. When the worker node receives the request, it checks current status of the processes allocated to it and returns the result to manager node.

4.5.2 Resilience to Network Failures

In the presence of node departure (spa) in the HyperCGSF, one of departing node's neighbor, denoted as spa', takes over spa's position. Then spa' inherits all the HyperCGSF-specific data from spa and takes its responsibility. After that, spa' broadcasts a notification inside the HyperCGSF to update the processes allocation data in all affected session tuples. Algorithm 8 illustrates the detailed procedure.

Algorithm 8:

```
1  begin
2      Set status to 'REMOVING'
3      spa' ← get LRU neighbor in lowest dimension
4      S_pes ← Get all plan execution sesssion tuples from local Node DB
5      foreach < id_session, id_plan, E_p, spa_m > ∈ S_pes do
6          Get task tuple from E_p based on id_plan
7          Update the session and E_p table in spa' for id_plan
8      end
9      S_task ← get all executing tasks in spa
10     foreach task_i ∈ S_task do
11         Migrate task_i to spa'
12     end
13     Broadcast INFORM(Repalce(spa', spa))
14     Broadcast INFORM(Depart(spa))
15 end
```

Algorithm 8 can be divided into four stages. First, the spa needs to set its status to "*REMOVING*", which means it stops receiving new tasks (Algorithm 8, line

2). Second, the spa communicates the replacing node (spa') for updating its hypercube-specific and the HyperCGSF-specific information (Algorithm 8, Line 4-8). Then, the GeoSPA moves all of its working tasks to the replacing node spa' so that the tasks can resume their execution (Algorithm 8, Line 9-12). Finally, the departing GeoSPA node broadcasts the message of $INFORM(Replace(spa', spa))$ to other worker agents to notify the replacement of spa with spa' and the departure of spa (Algorithm 8, Line 13-14).

4.6 Concluding Summary

In this chapter, a P2P-based geospatial service framework, called the HyperCGSF, was proposed to coordinate multiple GeoSPAs for achieving complex geoscience task. Given the user-specified service requirement, the HyperCGSF can automatically discover and composite geospatial service and orchestrate the execution of service composition in a decentralized manner. To describe HyperCGSF, this chapter is divided into four sections. First, a most widely used P2P network topology called Hypercube was analyzed and used for organizing the GeoSPA service nodes. Second, a problem formulation about service planning was given and the Distributed Geospatial Service Planning Algorithm (DGSPA) was introduced for automatically discovering and composing geospatial processing services. Third, we introduced the decentralized orchestration of geoprocessing workflow generated by the DGSPA. Finally, some important issues about workflow execution such as execution monitoring and fault tolerance technologies were discussed. In the next chapters, we applied the dust storm detection as a study case to test the efficiency of the HyperCGSF on processing of EO data.

Chapter 5: The Integrated Dust Storm Detection Model

To evaluate the performance of HyperCGSF, the Integrated Dust storm Detection Model (IDDM) was developed as a study case by combining five models for dust storm detection, aerosol optical thickness (AOT) retrieval, and dust trajectories simulation. The IDDM is developed based on the integrated environmental modelling (IEM) paradigm, which has been applied for addressing scientific challenges, such as monitoring the environmental change detection and forecasting environmental problems (Granell et al., 2013). IEM focuses on the issues of resource integration, model sharing and reusing, and decision making through model integration (Bulatewicz et al., 2013).

5.1 Review of Methodologies for Dust Storm Detection

There is an increasing concern for the health impacts of dust storms on urban populations. Every year approximately 800 terra grams of dusts are released from arid and semiarid regions of the northwestern China (Zhang et al., 1997). Asian dust is usually associated with frontal systems and/or cyclones (Tsai, 2008)). Dust particles can be transported by northwesterly winds at surface-level under Asian winter monsoon and by westerly winds in the troposphere layer from the eastern Asian continent to the Pacific Ocean (Zhao et al., 2006). It is also known that dust aerosols can have serious aviation and human health impacts (Chan et al, 2007). The chemical dioxins, anthropogenic inorganic pollutants, and trace metals, as well as the associated pathogenic fungi and bacteria may attach to dust particles when dust storms are passed through the urbanized and industrialized regions (Garrison et al., 2003). In addition, the atmospheric impacts from Asian dusts have also been extensively studied (Husar et al., 2001; McKendry et al., 2001; Zhao et al., 2008). The atmospheric mineral-dust loadings can affect the earth's radiation budget and

atmospheric, which can cause a reduction of 30-40% in solar radiation, as well as reduce visibility and promote the formation of severe haze (Husar, 2001).

5.1.1 Satellite-based detection methods

Satellite remote sensing is advantageous in monitoring the spatial and temporal variations of dust events (Chiapello et al., 1999). The application of satellite imaging in dust storm detection has been extensively studied in the last two decades. The ultraviolet measurements of the Nimbus 7 Total Ozone Mapping Spectrometer (TOMS) and the Ozone Monitoring Instrument (OMI) have been used to detect mineral dust (Chiapello et al., 1999; Torres et al., 2007), and the observations in the visible bands of the SeaWiFS and MODIS have been used to characterize the properties of dust aerosols (Yao et al., 2012).

In the visible wavelengths, Rao et al. (1989) proposed the first AVHRR algorithm used radiances in AVHRR channel 1 (630 nm) for AOT retrieval. Pavolonis et al. (2006) analyzed the limitations of "reverse absorption" technique and proposed another method that applies the ratio of reflectance at3.75 μm and 0.65 μm (hereafter RAT (3.7 μm, 0.65μm)), as a complement for automated dust aerosols detection. Qu et al. (2006) used Normalized Difference Dust Index (NDDI), which expressed as a normalized ratio of 2.1μm band and blue band, to detect dust storms and monitor the moisture change of dust. Lee et al. (2012) retrieved AOT from the Geostationary Ocean Color Imager (GOCI) on board the Communication, Ocean, and Meteorological Satellites (COMS) applying six visible bands (412, 443, 490, 555,660 and 680 nm) and two near-infrared (NIR) wavelengths (745 nm and 865 nm). However, these algorithms, which are based on visible and ultraviolet channels, are applicable only during the daytime.

Some successes in detecting dust form satellite-based Infrared (IR) measurements have been reported (Ackerman, 1997; Legrand et al., 2001; Li et al., 2007). Furthermore, quantitative physical parameters could be more beneficial than

an ambiguous dust index if measurements were possible during the nighttime. Then, using the Mie calculation for the Asian dust (Han et al., 2013), it may be possible to covert IR aerosol optical thickness (AOTs) into more familiar visible AOTs, if IR AOTs can be retrieved form IR measurements during the nighttime. Satellite radiometer measurements in thermal infrared channels own the capability of detecting dust storms during both the daytime and the nighttime (Ackerman, 1997; Wald et al., 1998). The Reverse Absorption Technique (RAT), which uses the Brightness Temperature Difference (BTD) of two or more thermal infrared bands, is the most commonly used satellite dust detection technique. BTD (11μm - 12 μm) can be applied to distinguish dust aerosols from clouds since dust particles absorb more infrared radiation at shorter wavelength while ice or liquid water particles exhibit higher absorption in longer wavelengths (Prata, 1989; Legrand et al., 2001; Ellrod et al., 2003; Zhao et al., 2010). Ackerman (1997) found that measurements of the dust refractive index show large discrepancies in the 3.7 μm and 11 μm wavelengths, thus proposed a method to locate and track the dust outbreaks (e.g., BTD 3.7 μm - 11 μm). Legrand (2001) developed the Infrared Difference Dust Index (IDDI) to detect the presence of desert dusts over Africa. The rationale of IDDI is based on observing thermal radiation (10 μm -12 μm) emitted by the same scene over the course of several days, where distinct changes are evaluated for potential dust presence. Ellrod et al. (2003) demonstrated a Three band Volcanic Ash Product (TVAP) using three bands from Geostationary Satellite System (GOES) centered at 3.75 μm, 11 μm and 12.0 μm wavelength. In contrast to BTD, TVAP generates better ash retrieval results due to its high sensitive to thin ash.

5.1.2 Radiative Transfer Model (RTM)

In order to comprehensively analyze the effort of dust storm presence and meteorological conditions on satellite observed brightness temperature at different wavelengths, the RTM was used in this chapter to simulate the radiances of different bands of MTSAT with various dust aerosols or atmospheric parameters, such as AOT,

effective radius, and water vapor. The influence of atmospheric conditions on the relationships of the TOA BTs at thermal infrared bands (in this study, the 3.75, 11, and 12 μm) were simulated by the Santa Barbara DISORT (DIScreet Ordinate Radiative Transfer) Atmospheric Radiative Transfer (SBDART) model choosing standard middle latitude winter mode as the atmospheric profile. The surface type is set to bare soil. The dust optical properties, including the single-scattering albedo (SSA) and asymmetry parameter, were defined based on tabulated values for mineral nucleation mode dust from OPAC (Optical Properties of Aerosol and Cloud) (Hess et al., 1998). The surface emissive of bare soil at the three channels was set according to the seeBor database.

5.2 Data Used and Study Area

5.2.1 Study Area

The study area covers most of the East Asia from 20°N to 55°N latitude and from 95°E to 135°E longitude, which includes the main source regions for Asian dusts: the Gobi and the Taklimakan deserts (Figure 5.1) (Sun et al., 2001; Shao et al., 2006). Approximately 240 Tg of dusts are re-deposited in Chinese deserts each year (Zhang et al., 1997), and 140 Tg of dusts falls off during their transportations in China (Zhang et al., 1997). Climatologically, dust storms in east Asia are reported dominantly in winter-spring season, and the highest frequency is observed in April. Approximately one-third to one-half of yearly dust storms occurs in April (Natsagdorj et al. 2003; Zhou and Zhang, 2003). During dust peak season, the estimated dust loads reach to 1.7×103 kg/km^2 (Shao et al., 2006). Tan et al. (2012) analyzed the transport pathways of dust storms from two stations (Sunitezuoqi (41.37°N, 102.37°E) and Guaizohu (43.87°N, 113.63°E)) of main dust resources. They have concluded that the pathways are normally transported from Inner Mongolia deserts via the Loess Plateau to the North China Plain, and then entered into the East and South China Sea.

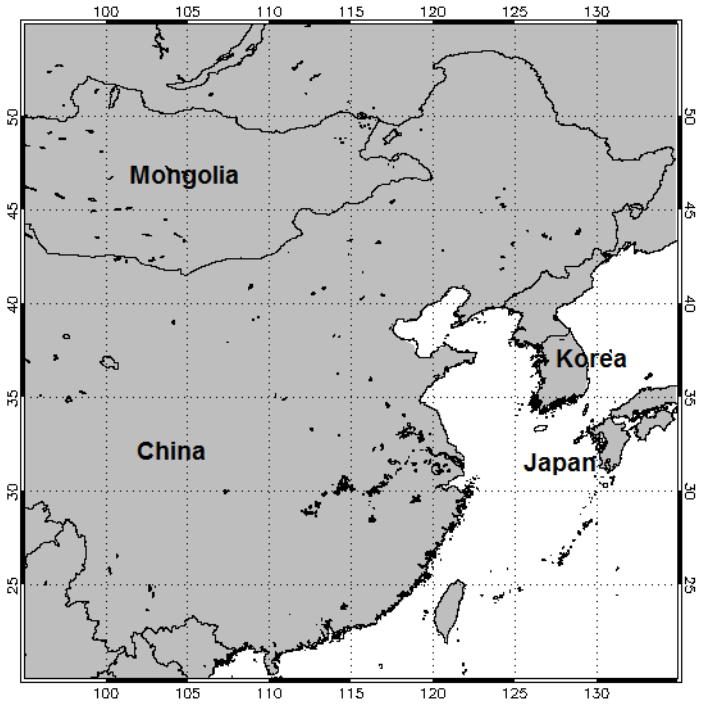


Figure 5.1. Domain of this study

5.2.2 EO Data Used

Multi-functional Transport Satellite (MTSAT-2)

MTSAT-2 is a Japanese geostationary earth-orbiting satellite. In contrast to low-earth orbiting satellites, geostationary satellites can profile atmospheric aerosols at higher temporal resolution. Hourly Brightness Temperature (hereafter BT) images derived from four infrared channels (mid-infrared: IR3 and IR4; thermal-infrared: IR1 and IR2), with a spatial resolution of 4 km, and one visible channel (VIS), with a resampled spatial resolution of 4 km (original data resolution is 1 km), were used. A summary of all MTSAT channels is given in Table 5.1.

Table 5.1: Different Band wavelengths and corresponding resolution of MTSAT

Wavelength/ Band	Spectral Range (μm)	Spatial Resolution (km)
IR1	10.5-11.5	4
IR2	11.5-12.5	4
IR3	6.5-7.5	4
IR4	3.5-4	4
VIS	0.55-0.9	1 (resampled to 4km)

MODerate resolution Imaging Spectroradiometer (MODIS)

MODIS is a passive multi-spectral imager deployed aboard the two NASA Earth Observing System (EOS) satellites, TERRA and AQUA, which respectively crosses the equator at 10:30 a.m. and 1:30 p.m. (UTC). Each provides global daily detection of the earth-atmosphere system in 36 spectral bands ranging in wavelength from 0.4 μm to 14.4 μm, with a ±55-degree scanning pattern at the EOS orbit of 705 km and a 2330-km imaging swath, and provides global coverage every 1-2 days. The MODIS project provides many standard products for the scientific community, and the morphemic profile products represent one of them, denoted as MOD_07 or MYD_07 if the Terra or Aqua platforms are used, respectively. In this study, we only use the MOD_07 Collection 5 products collected form the Terra platform. The MOD_07 product consists of several parameters, which include the total ozone burden, the atmospheric stability, the temperature and moisture profiles, and the atmospheric water vapor. The pixel resolution is 5 km×5 km.

NCEP FNL Final (FNL) Operational Global Analysis data (NCEP/FNL)

The NCEP final operational global analysis data in the GRIdded Binary (GRIB1) format is used in this study as atmospheric profiles data. The NCEP FNL (Final) Operational Global Analysis data are on 1-degree by 1-degree grids prepared operationally every six hours. This product is generated from the Global Data Assimilation System (GDAS), which continuously collects observational data from the Global Telecommunications System (GTS), and other sources, for numerous analyses. The data are on a 1°×1° longitude/latitude grid and generated globally every 6 hours (0:00, 06:00. 12:00, 18:00 UTC). The extracted atmospheric profiles of FNL have 26 mandatory (and other pressure) levels from 1000 to 10 hPa, in the surface boundary layer and at some sigma layers. Other vertical atmospheric parameters include the geopotential height, the air temperature and the relative humidity.

The advanced research ARW Weather Research and Forecasting (WRF) model is used to generate the 1-hour, 10×10 km atmospheric profiles. The 6-hour Final (FNL) Operational Global Analysis data with 1°×1° resolution from the NCEP were used for meteorological initial and boundary conditions of WRF. The Yonsi University (YSU) planetary boundary layer (PBL) (Hong et al., 2006), the NOAH land-surface model (Chen and Dudhia, 2001) and the Monin-Obukhov surface layer scheme (Obukhov, 1971) were used in the simulation. Grell 3D (Grell, 1993) scheme was used as the cumulus parameterization. The rapid radiative transfer model (RRTM) scheme (Mlawer et al., 1997) was used for both short-wave and long-wave radiation including only direct radiative effect of aerosols.

5.3 IDDM Description

Figure 5.2 illustrates the workflow of IDDM. The rectangle represents the model input and output item, which the round rectangle represents the geospatial model. The detailed introduction of each model was given below.

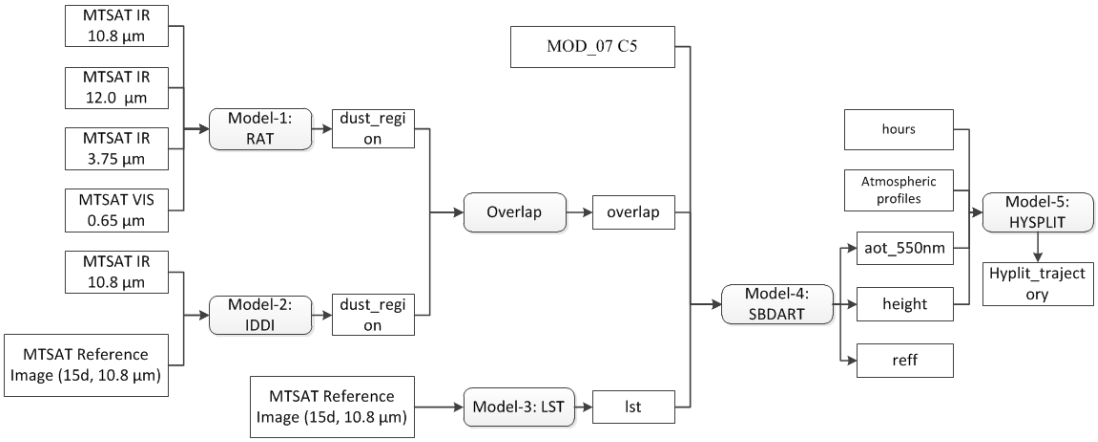


Figure 5.2. Workflow of the IDDM

5.3.1 Model-1: Combined reverse absorption technique (RAT) Model

The Combined RAT is one of the most widely used satellite-based techniques for dust storm detection (Gu et al., 2003; Hu et al., 2008), which exploits selective absorption in the thermal infrared wavelengths (e.g. 11 μm and 12 μm) (equation 1),

where dust aerosols absorb more infrared radiation at shorter wavelength and absorb less at longer wavelength (Prata, 1989; Ackerman, 1997). The presence of dust storm can be identified and distinguished with cloud pixels by BTD threshold tests of RAT. A negative difference (11 μm - 12 μm) of < -1.0 °K is normally used as the threshold to indicate the presence of dust (Ellord, 2003).

$$BTD_{11-12} = BT_{11\mu m} - BT_{12\mu m} < Threshol \quad (3)$$

However, the two bands RAT may generate false detection in some situations such as high water vapor concentrations, which are common in tropical areas (Ellrod et al., 2003; Pavolonis et al., 2006). Ellrod et al. (2003) demonstrated a Three-band Volcanic Ash Product (TVAP) using Geostationary Satellite System (GOES) data centered at 3.75 μm, 11 μm and 12.0 μm (Equation 2). TVAP generates better ash-retrieval results than BTD due to its high sensitivity to thin ash. The threshold value of TVAP can be set between 60 and 100 during nighttime for very thin ash and 200 and 255 for relatively thick ash during daytime (Ellrod et al., 2003). TVAP is also valid for detecting dust storms. In Figure 5.3(b), the TVAP values for dust storms over land or sea are both higher than those of the land/sea background. Clouds impartially have TVAP values between 150 and 200 and lower IR1 values. Therefore, by using TVAP and IR1, dust storms over land and sea can be discriminated from both clouds and land/sea background. The output of TVAP is shown in Figure 5.3(c).

$$TVAP = 60 + 10\left(BT_{12\mu m} - BT_{11\mu m}\right) + 3\left(BT_{3.75\mu m} - BT_{11\mu m}\right) < Threshold \quad (4)$$

5.3.2 Model-2: The Infrared Difference Dust Index (IDDI) model

The Infrared Difference Dust Index (IDDI) was first proposed by Legrand (2001) to detect the presence of desert dusts over Africa using thermal band (11 μm) of midday Meteosat-IR imagery. Based on Legrand, the surface thermal radiance does not change over relatively short time periods (e.g. 15 days) on clears days. However,

in the presence of dust, outgoing thermal radiation will be attenuated along its path through the atmosphere, resulting in a reduction in the apparent radiance at the top of atmosphere when compared to a clear day. Therefore increased contrast between observations can be obtained during clear and dust conditions. The thermal contrast can in turn be an indicator of dust pixels. In this study, the IDDI is used as an indicator for dust detection where dusts have a distinct range of IDDI as:

$$IDDI = T_{ri} - T_{oi} \tag{5}$$

Where T_{oi} is the BT of each pixel in original image and T_{ri} is the BT of each pixel in the reference image. The reference image can be derived by calculating the maximum pixel value of each pixel over a period, such as 15 days in this study. As shown in Figure 5.3(d), clouds, dust storms and land/sea background can be readily distinguished.

A significant dust storm occurred in the northern China and Inner Mongolia on April 27, 2012 and was selected for a sensitivity study using MTSAT imagery. Five sample regions (represented by different color squares with the size of 20 × 20 pixels) were selected from a MODIS RGB composite image (Figure 5.3 (a)), respectively for 5 different classes (e.g. land background (LB), sea background (SB), dust storm over Land (LD), dust storm over sea (SD) and cloud (CL)) to examine response at 3.75 μm, 11 μm and 12 μm wavelengths. Figure 5.3 (b) to (d) illustrate scatter plots for each of the five classes of dust detection applied to MTSAT. Based on Figure 5.3, the dust storms pixels can be clearly discriminated from cloud and from the land/sea background using proper BTD (11 μm - 12 μm), TVAP and IDDI threshold.

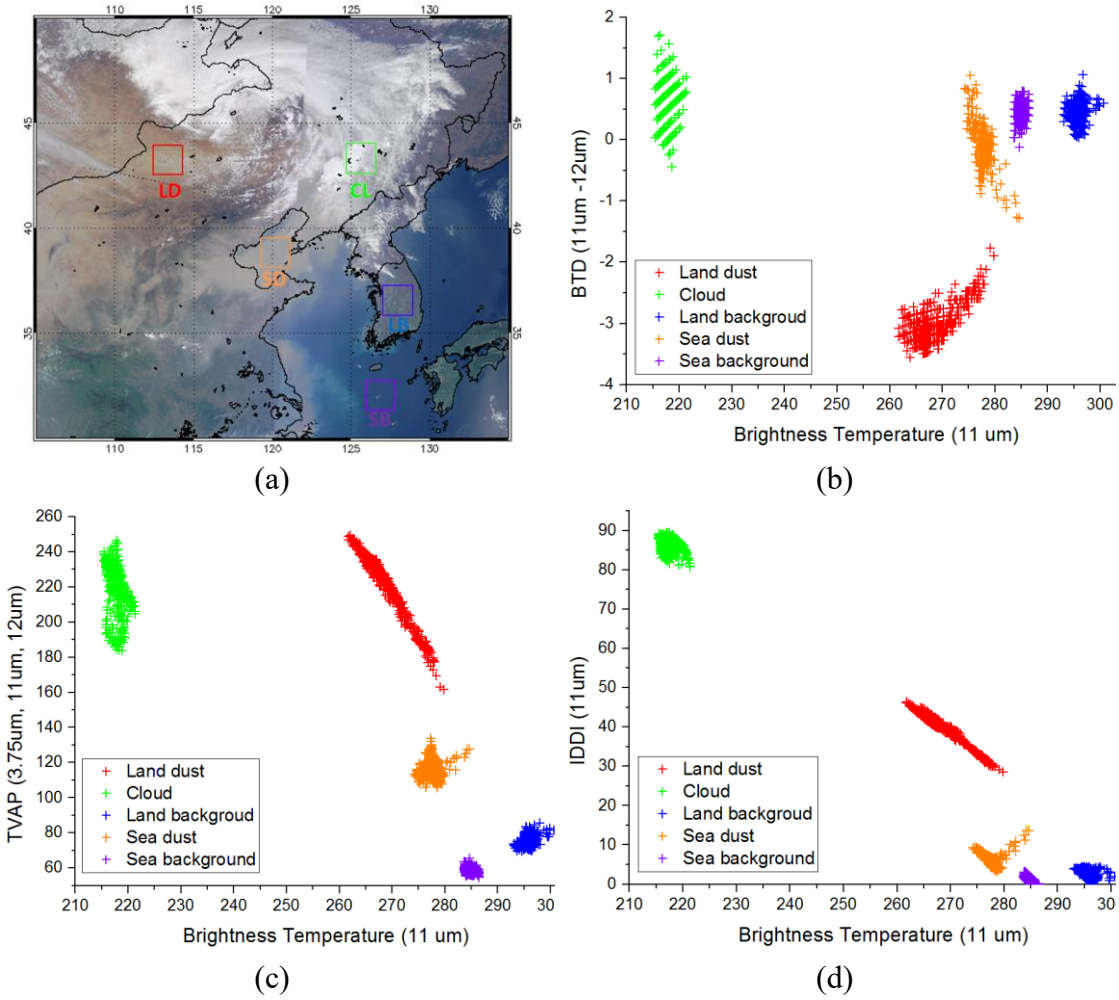


Figure 5.3. (a) RGB composition image from MODIS channel 1 (645 nm), 4 (555 nm) and 3 (469 nm) of a serious dust storm on April 27, 2012 and scatterplot of different class (e.g. land, cover, dust) of (b) BTD, (c) TVAP; (d) IDDI

5.3.3 Model-3: Land Surface Temperature (LST) Model

Based on the radiative transfer theory, the land surface temperature (LST) strongly affects the calculation of atmospheric aerosol properties (Prata et al., 1989). In this study, the LST was taken based on a reference $BT_{11\mu m}$. The reference $BT_{11\mu m}$ synthesized from two previous week's clear sky $BT_{11\mu m}$ value in the same study area. Considering the dust storm event is usually accompanied with the decreased temperature, 5K was subtracted form the background value (Zhang et al., 2006).

5.3.4 Model-4: SBDART Model

Table 5.2 illustrates the parameters for the simulations of infrared radiance on different MTSAT bands with SBDART. The relationships between BT11 and BT12 were simulated with various dust layer heights with atmospheric profiles of MLW. Figure 5.4 presents the relationship between BT11 and BTD11-12, with a water vapor amount of 0 g/cm2 and 3 g/cm2 respectively. For each figure, the retrieval net is labeled for specific effective radius and optical depth values. Solid lines in Figure 5.4 represent different effective radii, and the dashed lines represent optical depth at 550 nm.

Table 5.2: Configuration of SBDART in this study

Parameter	**Value**
Atmospheric Model	Midlatitude Winter (MLW)
Integrated water vapor amount (g/cm^2)	0,1,2,3
Solar zenith angle	40
View zenith angle	30
user azimuth angles	50
Surface temperature (K)	290, 300
Aerosol type	Mineral dust
Altitude of aerosol layer	2km
Aerosol optical depth	0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5

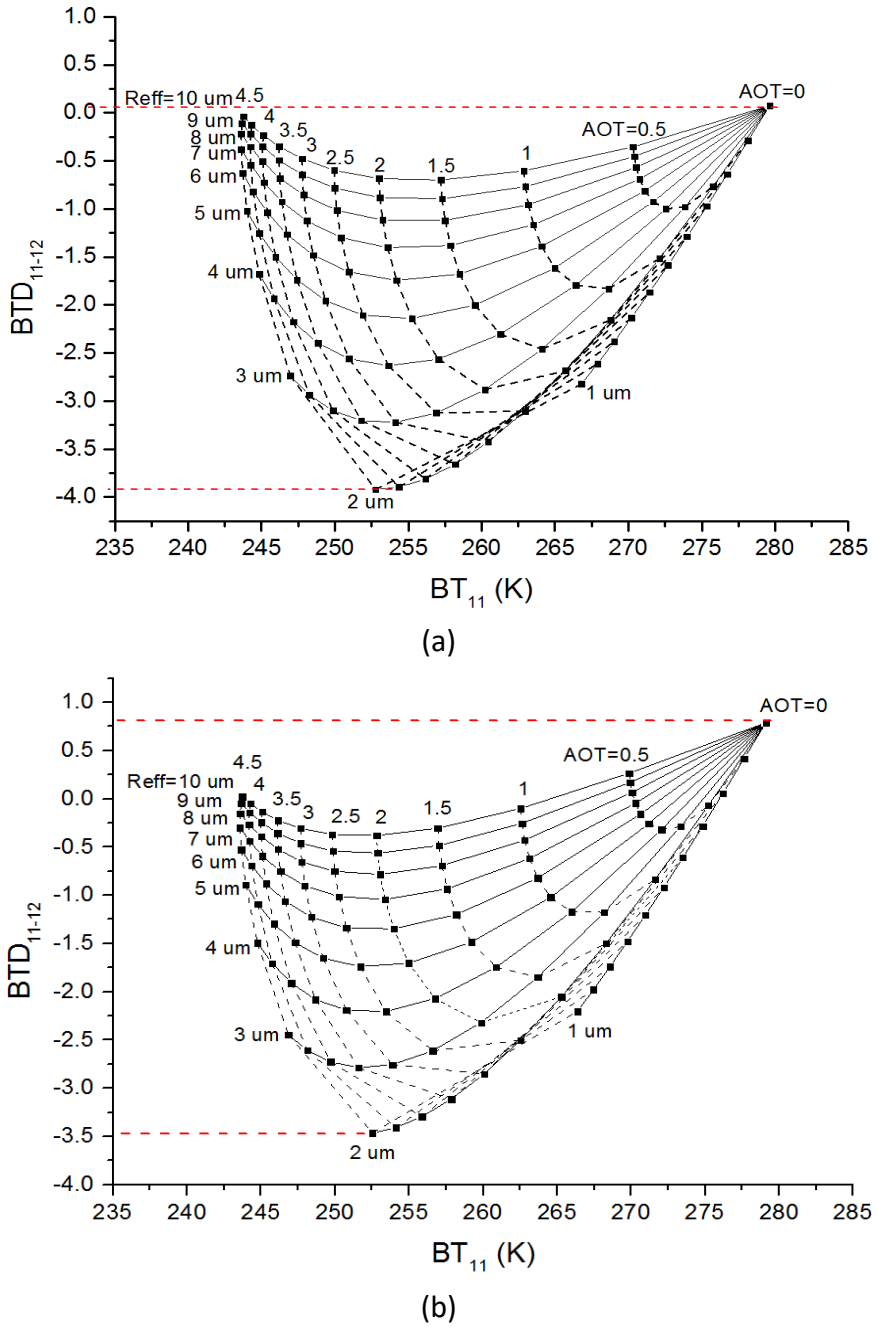


Figure 5.4. Simulation of the relationships between BT_{11} and $BTD_{11\text{-}12}$ for bare soil in various dust layer heights with an atmospheric profile of MLW: (a) BTD versus BT_{11} with a water vapor amount of 0 g/cm^2, (b) BTD versus BT_{11} with a water vapor amount of 3 g/cm^2. The altitude of dust layer is 2km and the surface temperature is set to 290K

Based on Figure 5.4, several conclusions can be drawn: first, the presence of dust aerosol displays a significant negative $BTD_{11\text{-}12}$ signal when the optical depth is greater than 0.5. Second, it can be seen that BT_{11} decrease with increasing optical

thickness while $BTD_{11\text{-}12}$ decreases with the increase of particle effective radius, which indicate that it is possible to retrieve the optical thickness and particle size of dust from the combination of BT_{11} and $BTD_{11\text{-}12}$ information. Finally, since absorption by water vapor in the infrared window is serious, the magnitude of the positive $BTD_{11\text{-}12}$ with a higher water vapor amount is greater than the $BTD_{11\text{-}12}$ with lower water vapor amount. This can explain why the BTD methods are always failed for dust storm detection in moist district, such as the South-East China.

IR radiances of the identified dust storm pixels are used for the dust storm properties retrieval by using the lookup-table (LUT) based inversion technique. Basically, the LUTs describe the relationships between satellite receiving radiance and the AOT for given atmospheric and surface conditions. For the construction of the LUTs, the SBDART radiative transfer code was run using the OPAC (Hess, 1998) dust model. Similar to Lee et al., (2013), for each dust storm presence and each of the two selected channels (11μm and 12μm), the LUTs contain radiances computed for 11 dust storm loadings (AOT = 0.0, 0.2, 0.4, 0.8, 1.0, 1.4, 1.8, 2.2, 3.0, 4.0, and 5.0) and 10 altitudes of the layer from 1 km to 10 km. LUTs were calculated using the SBDART code for the retrieval, which can calculate solar flux as well as radiance for the solar illumination and satellite observation geometry based on searching for the closest value in the LUT, according to the root–mean–square deviation (RMSD):

$$RMSD = \frac{1}{N_i}\sqrt{\frac{(BT_i{}^{calc} - BT_i{}^{obs})^2}{BT_i{}^{obs}}} + \frac{1}{N_i}\sqrt{\frac{(BTD_i{}^{calc} - BTD_i{}^{obs})^2}{BTD_i{}^{obs}}} \quad (6)$$

Where $BT_i{}^{calc}$ and $BT_i{}^{obs}$ are the calculated and observed BTs, respectively, of channel i. BTD is the BT difference between two IR channels. To be selected as closest to the observations, a set of BTs within the LUT should have values similar to the observations for each channel, and differences between the BTs should be

sufficiently close to the differences between the observed BTs. Before running the SBDART model, the input water vapor parameter of MOD07 needed to be interpolated in order to match the resolution of MTSAT, which is 4 kilometers.

5.3.5 Model-5: HYSPLIT Model

To understand the impacts of dust storms form different source areas on the China, a 72-hour forward trajectory analysis was performed using the NOAA HYSPLIT model with inputs from the National Centers for Environmental Prediction/the National Center for Atmospheric Research (NECP/NCAR) global reanalysis meteorological data. This public domain model is available at the NOAA/ARL's (U.S. National Oceanic and Air Administration/Air Resources Laboratory) web server (https://ready.arl.noaa.gov/HYSPLIT.php).

5.4 Implementation and Deployment of the IDDM

The previous section described the IDDM for real-time dust storm detection. This section focuses on an implementation of the IDDM based on the HyperCGSF as a specific case study.

5.4.1 Setting Up the Model as a Geospatial Processing Service

Every sub-model in the IDDM is implemented in Java programming language based on GeoSPI and packaged in the form of standard Java Archive (JAR) file. Then the JAR package was deployed as a GeoSPA processing service by copying the single JAR package into a folder that has been specified by GeoSPA as the model deployment folder. Meanwhile, the GeoSPA can automatically parse the JAR file and conduct two operations. First, the GeoSPA transfers the model metadata to its knowledge-base in the form of geospatial process objects, and the dependency solution is generated too. Second, the geospatial process is exposed based on the OGC WPS standard, which allows model users to retrieve detailed model process metadata through '*DescribeProcess'* operation and run a given model by calling the

Execute operation. Once the resources have been developed and deployed on the GeoSPA, they can be discovered through DGSPA and composed into a plan. Table 5.3 shows the geospatial processes of the IDDM and their input/output parameters.

Table 5.3. The geospatial processes of IDDM and their input/output parameters

Model	Geospatial Process Name	Input parameters	Output parameters
Model-1	sds: CombinedRAT	MTSAT-2 (IR1, IR2, IR3, IR4, and VIS)	dust_region
Model-2	sds: IDDI	MTSAT-2 (IR1,VIS), MTSAT-2 IR1 reference image	dust_region
Model-3	sds: LST	MTSAT-2 IR1 reference image	lst
Model-4	sds: SBDART	MTSAT-2 (IR1, IR2), land_surface_temperature, MOD07_5	aot_550nm, reff, dust_height
Model-5	sds: HYSPLIT	dust_region, dust_height, hours	forward_trajectory

* The geospatial process name consists of two parts connected by a colon: the left part is the workspace name and the right part is the actual process name.

Furthermore, several EO datasets must be first collected and uploaded on the HyperCGSF for model run. First, the MTSAT-2 data was automatically downloaded from the Hong Kong Observatory per hour and feed into HyperCGSF through the GeoSPA EO data service. Second, the MOD07 data was downloaded from official website of MODIS to offer the water vaper data. All of the EO datasets is processed based on the tile-based storage scheme and can be accessed by users and GeoSPAs through the OGC WMTS operations.

5.4.2 Running the Model in a Workflow Composition

Once the geospatial data and processing resources has been developed and

deployed on HyperCGSF as standard-based geospatial services, all sub models of the IDDM can be discovered and composed as plan object through DGSPA. By using the DGSPA, the service consumer needs to specified a service requirement object denoted as $\langle r_i, r_o \rangle$, for example where $r_i = (mtsat2, mod\_07\_c5)$ and $r_o = (aot\_550nm, forward\_\text{trajectory})$, to a random GeoSPA as the manager agent on behalf of the service consumer to interact with the HyperCGSF. The manager agent generates an abstract plan through DGSPA and returns the abstract plan to service consumer for further operation. If the service consumer agrees the proposed abstract plan, the plan can be initialized and submitted for execution by specifying all mandatory input parameters by the service consumer.

To complete the plan initialization, links should be established between sub models of IDDM to define how data will flow during plan execution. Based on the workflow of IDDM shown in Figure 5.2, an '*ras*:*overlap*' process is linked to '*sds:CombinedRAT*' and '*sds:IDDI*' respectively to supply the '*dust_region*' output which represent the region of dust storm presence. Similarly, a link is established to send the output variable '*lst'*, which represents land surface temperature, from the '*sds:LST*' to the '*sds:SBDART*' for dust storm physical parameters (AOT at 550nm, dust layer height, and effective radius) retrieval. Finally, a link is added between the '*sds:SBDART*' and '*sds:HYSPLIT*' to supply the forward trajectories of dust particles. Collectively, these links define how data will flow during IDDM execution.

5.5 Illustration of Final Result

Two dust storms events (i.e., April 27-30, 2009, March 20-22, 2010) were used as case study to evaluate the performance of the model. Figure 5.5 shows the IDDM-derived dust storm pixels (column 2) and spatial distribution of dust AOT at 550 nm for the three dust storm cases. The IDDM-derived AOT in Figure 5.5 (column 3) works on the dust pixels and presents promising results when compared MYD04 products (column 4).

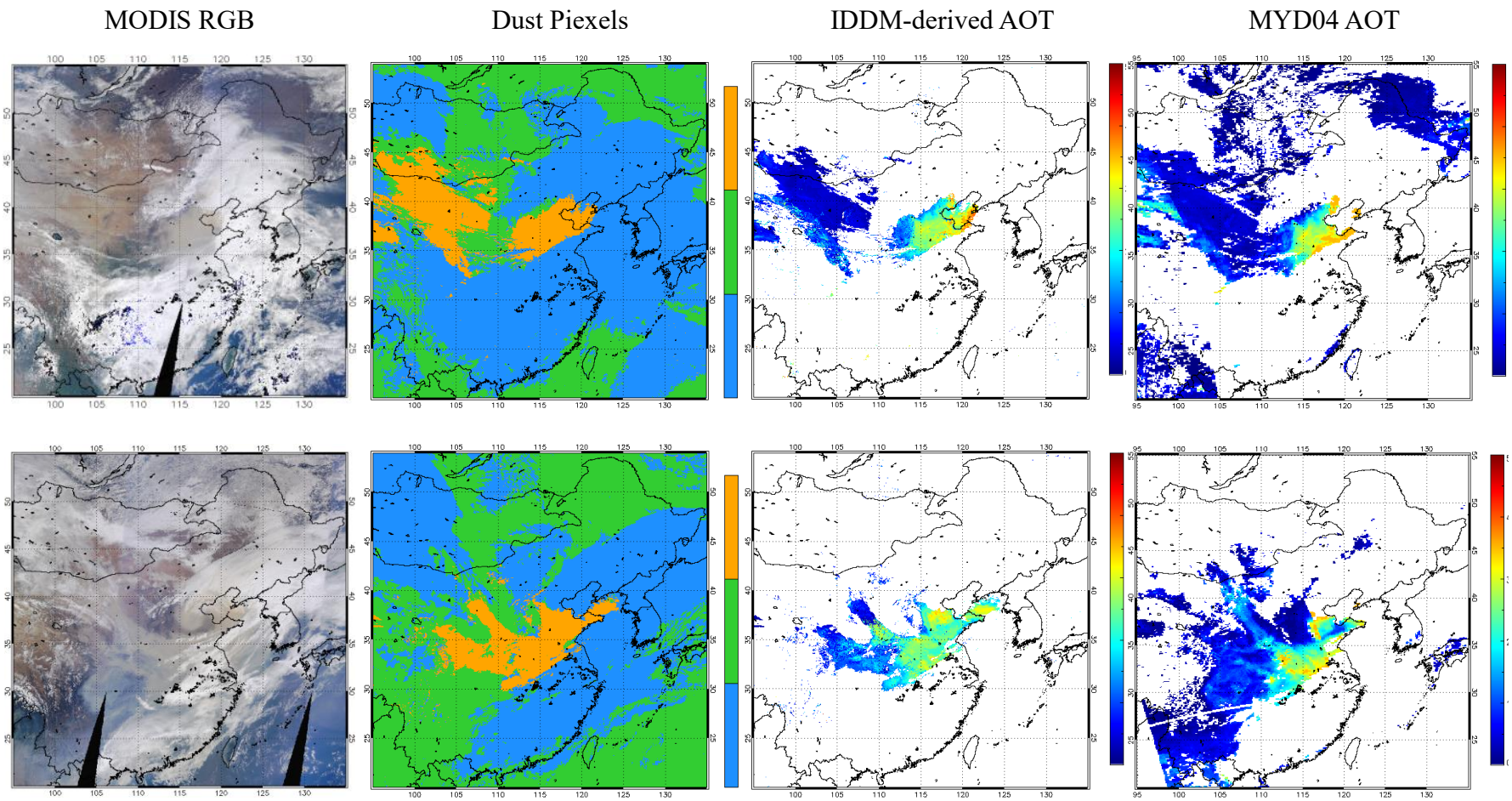
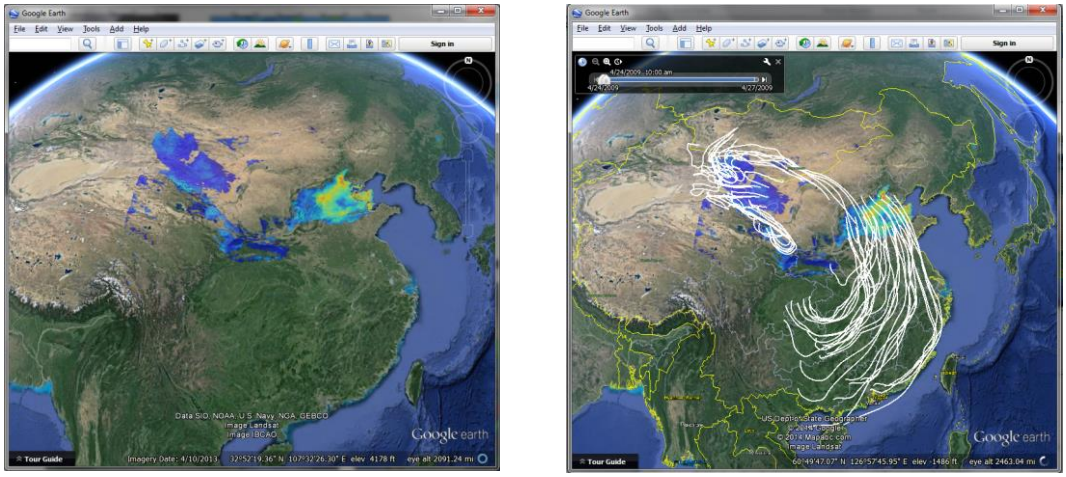


Figure 5.5. MODIS RGB color composite image, IDDM-derived dust presence, IDDM-derive AOT at 550nm, and MYD04-based AOT at 550nm from upper to lower image for (upper) a dust storm case on 24 April 2009 and (lower) a dust storm case in 20 March 2010

As shown in Figure 5.5, it can be seen that the IDDM can effectively detect the dust storm presence based on the EO dataset, and the IDDM-derived AOTs are statistically comparable to the MODIS AOT products (MYD04). This newly automated IDDM can be used to give advance near real-time warning of dust storms, for both environmental authorities and public. It is also benefit from early warning of adverse air quality conditions, and prediction of low visibility associated with dust storm events for port and airport authorities.

Figure 5.6 shows the dust storm case initiated over the northwestern China and Mongolia on April 24, 2009.



(a) (b)

Figure 5.6. Screen capture of displaying the (a) NN-derived dust AOT at 550 nm and (b) simulated dust storm transportation paths generated by HYSPLIT model, in Google Earth

As shown in Figure 5.6(a), the dust storm travelled from the Inner Mongolia deserts, over the Loess Plateau and North China Plain, and then entered to the East China Sea through Shandong, Jiangsu, Zhejiang and Fujian provinces, eventually reaching Taiwan and Hong Kong. This dust storm thus affected large areas, and high values of aerosol concentrations were observed in Beijing, Korea, Taiwan and Hong Kong. The white lines in Figure 5.6(b) indicate the simulated dust storm transportation paths generated by HYSPLIT model.

5.6 Concluding Summary

In this chapter, several widely used dust storm detection as well as aerosol optical thickness retrieval approaches were reviewed first and the dataset used and study area in this research were illustrated. Then based on the dust storm detection approaches, the Integrated Dust storm Detection Model (IDDM) was proposed as the real-world study case to test the HyperCGSF developed in this research. The IDDM consists of five models (CombinedRAT, IDDI, LST, SBDART and HYSPLIT) and one raster-based process (Overlap), which were developed in Java programming language and deployed onto the GeoSPA as web-based geospatial processing services. Finally, two dust storm cases were used to evaluate the accuracy of IDDM. In next chapter, some experiments were conducted and analyzed to evaluate the performance of the HyperCGSF.

Chapter 6: Evaluation and Discussion

In this Chapter, several experiments were designed and conducted to test the efficiency of the proposed HyperCGSF and relevant algorithms using the IDDM introduced in Chapter 5 as the study case. Before discussion about the experiment result, the experiment environment building on top of the commercial clouding computing platform was first introduced. Then the GeoSPA service models were tested and discussed. Finally, a condition of node departure was simulated and tested to evaluate the stability of the HyperCGSF.

6.1 Experiment Environment

Provided as part of the Google Cloud Platform, the Google Compute Engine (GCE) is an infrastructure service which is made up of three major components: virtual machines, persistent disks, and networks. GCE is available at several Google data centers worldwide and is provided exclusively on an on-demand basis. GCE provides worldwide Cloud services, such as IaaS, PaaS, and SaaS. Figure 6.1 illustrates the GCE console for managing the computing resources used in this study.

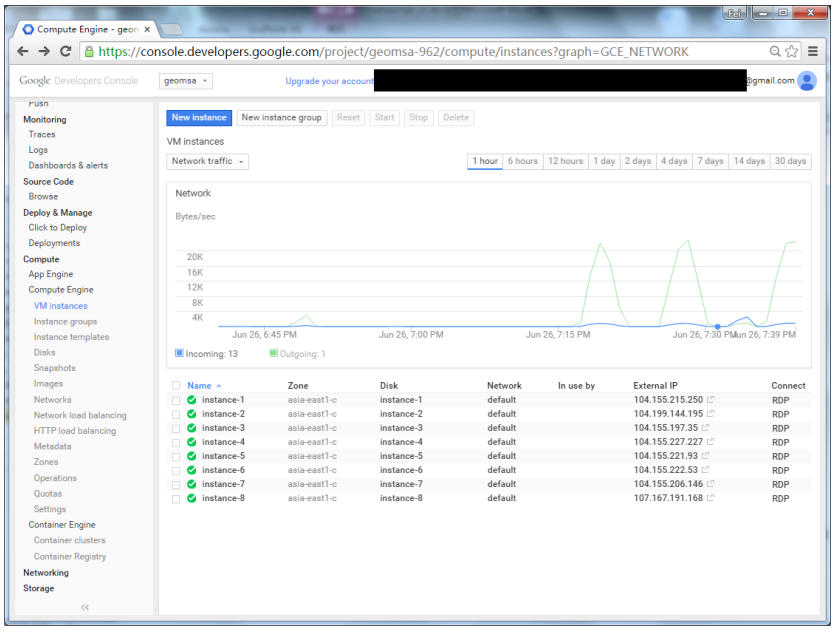


Figure 6.1. Google Compute Engine and VM instances used in this study

As shown in Figure 6.1, eight GCE virtual machine (VM) instances (instance-1 to 8) with CentOS 6 operating system were purchased in this research and each VM has one virtual CPU of 2.50 GHz, with 3.75 GB of RAM, a 50-GB disk, and bandwidth of 4 Gb/s. Performance tests were conducted to evaluate the potential computational costs introduced by the HyperCGSF. A prototype system of HyperCGSF was implemented and deployed onto the GCE platform.

6.2 Prototype System of HyperCGSF

A user-friendly integrated operating environment (IOE) based upon web technologies was developed to help clients access the services of HyperCGSF. Figure 6.2 shows the layout of the IOE interface, which is composed of three components: toolbar panel (number 1 in Figure 6.2), layer management panel (number 2 in Figure 6.2), and workspace panel (number 3 in Figure 6.2).

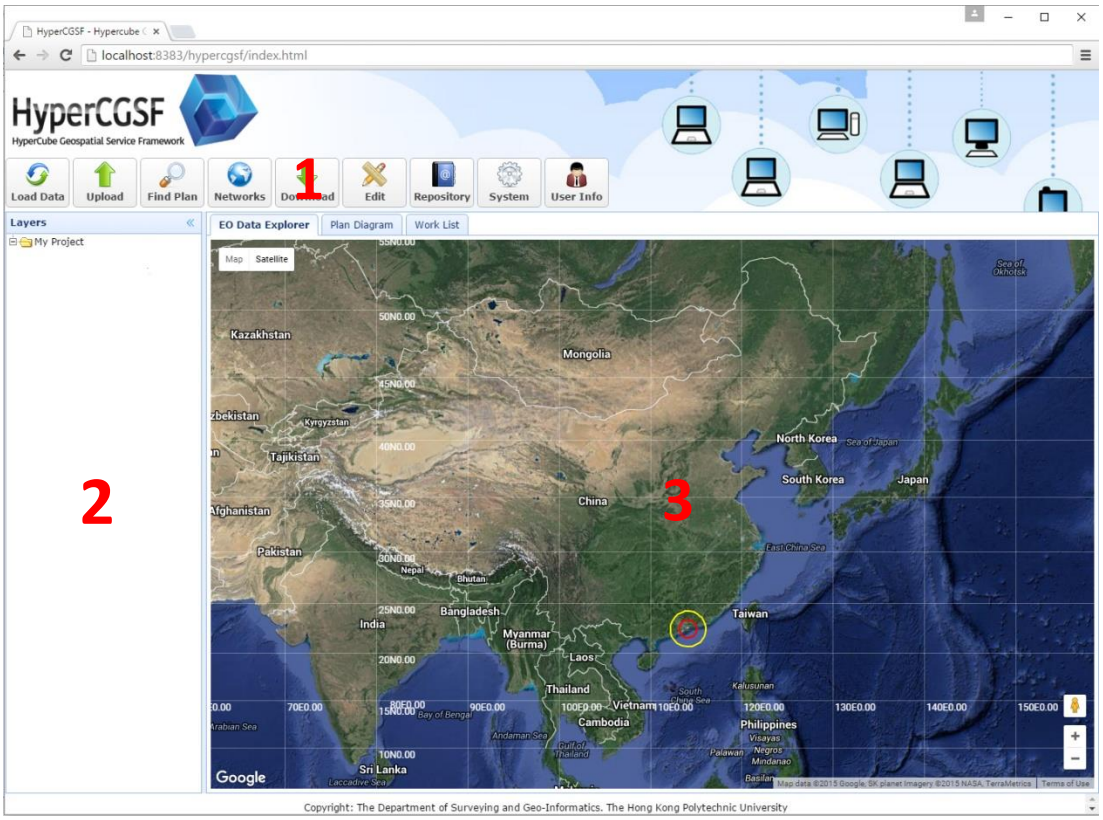
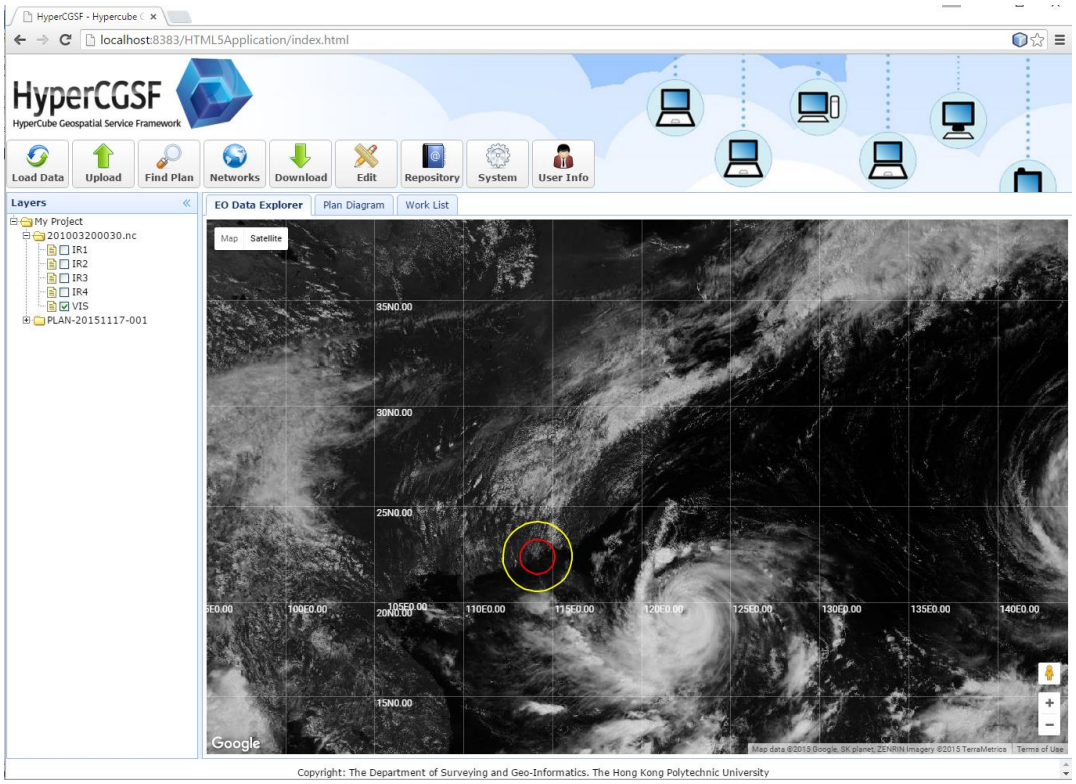
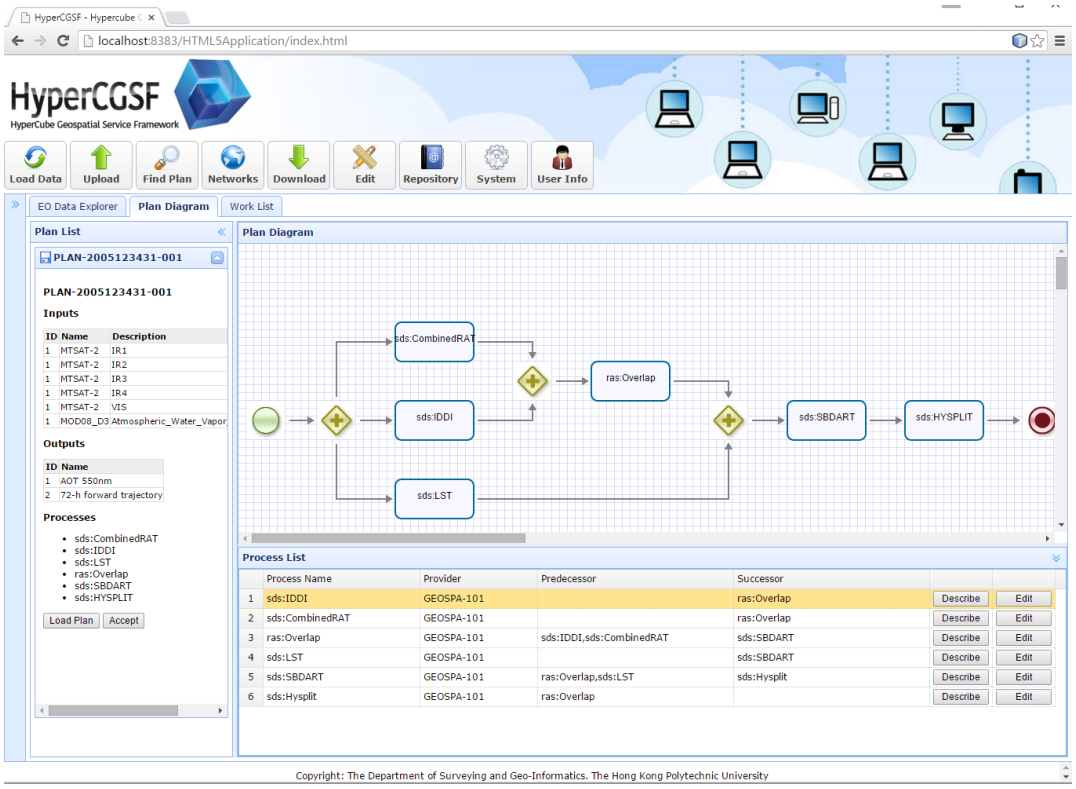


Figure 6.2. Integrated operating environment of HyperCGSF

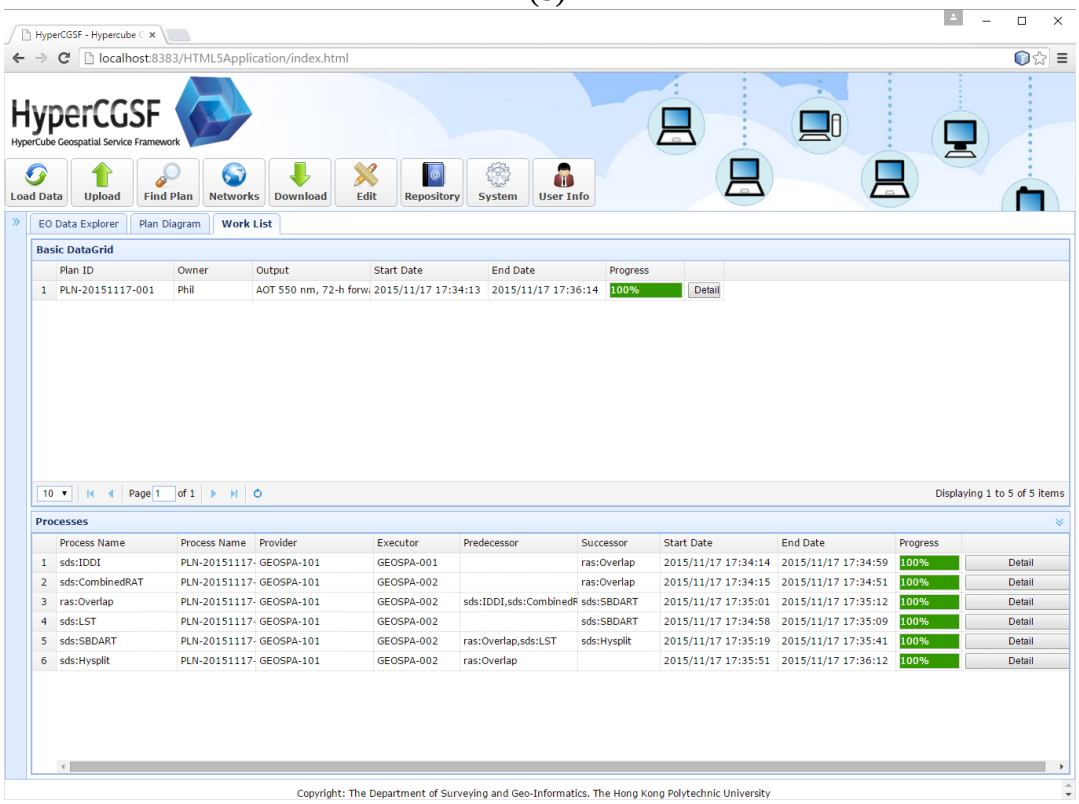
As shown in Figure 6.2, the IOE is composed of three components: toolbar panel (number 1 in Figure 6.2), layer management panel (number 2 in Figure 6.2), and workspace panel (number 3 in Figure 6.2). The toolbar panel consists of a set of buttons which can trigger various functionalities provided by GeoSPA (e.g., loading and editing EO data, find plan, and checking system information). The layer management panel locates on the left side, which has a 'tree' control for displaying and managing the EO data layers and the processing result generated by GeoSPA. The workspace panel has a control called 'tab' which contains other three panels: the 'EO Data Explorer' panel (Figure 6.3(a)), the 'Plan Diagram' panel (Figure 6.3(b)), and the 'Work List' panel (6.3 (c)).



(a)



(b)



(c)

Figure 6.3. (a) The EO Data Explorer displaying the MTSAT-2 VIS layer. (b) The Plan Diagram displaying the DGSPA-derived plan of the IDDM and (c) The Work List panel

When platform users load EO data into IOE using the 'Load Data' button, the name of this dataset together with band information are added to the layer tree. Then the user can select the desired layers and display them in the EO Data Explorer panel (Figure 6.3 (a)). In addition, when the user submits a service requirement through the 'Find Plan' button, the GeoSPA on behalf of the user as the manager node executes the DGSPA and obtains an abstract plan. The abstract plan will be displayed as a workflow diagram which is displayed in the Plan Diagram panel. Figure 6.3(b) illustrates a workflow diagram of the IDDM model. As shown in Figure 6.3(b), the Plan Diagram panel consists of two sub-panels, a diagram panel on the top and an information panel on the bottom. Then the service consumer can edit the abstract plan on the Plan Diagram panel for supplying required data and parameters to enable the abstract plan executable. Once the plan is submitted for execution, the plan status will be displayed on the Work List panel (Figure 6.3(c)). If the plan executes successfully, the final result is added and displayed in the layer management panel and the EO Data Explorer panel for the user to check. Figure 6.4 illustrates the output parameters of the IDDM.

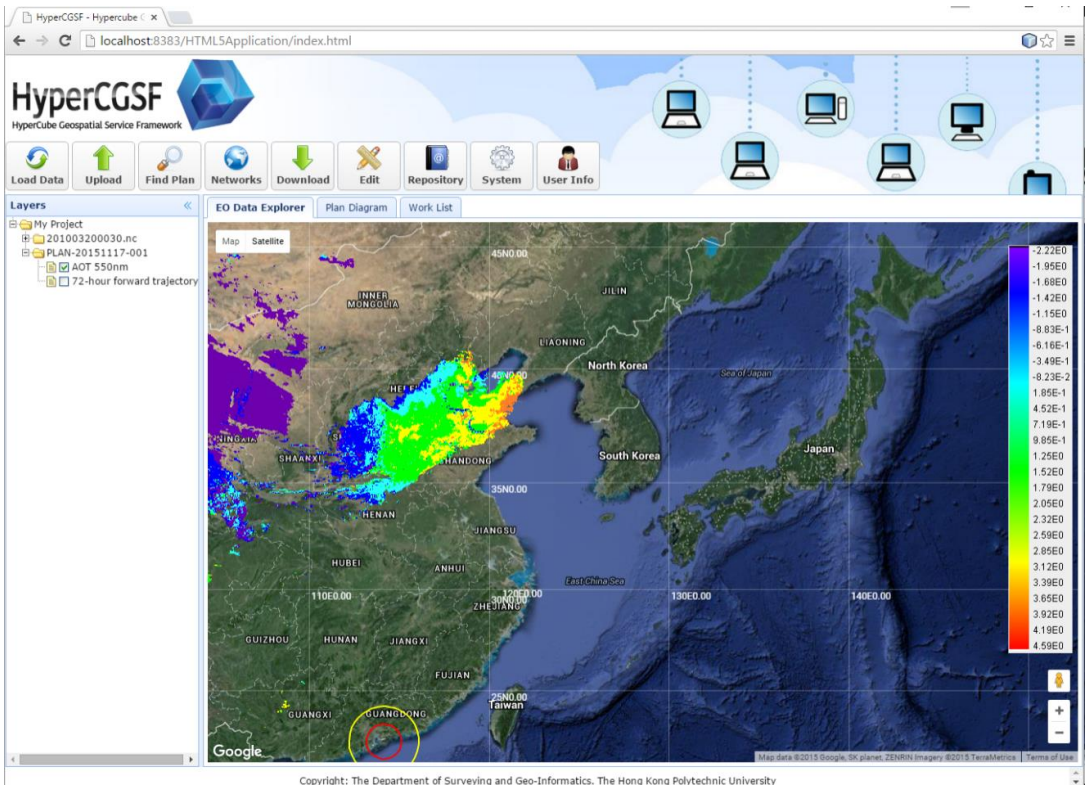


Figure 6.4. Displaying output of IDDM on the EO Data Explorer

6.3 Evaluation of GeoSPA EO Data Service

To evaluate the efficiency of GeoSPA EO data service, two platforms were built and tested respectively with a changing request rate. Each platform contains three GCE VMs. Figure 6.5 illustrates the structure of these two platforms.

Figure 6.5. Two platforms for GeoSPA EO data service evaluation: (a) THREDDS-based GeoSPA EO data service testing environment, and (b) WMTS-based GeoSPA EO data service testing environment

For the first platform in Figure 6.5(a) (hereafter platform-1), three VMs equipped with TDS were combined as a cluster system for offering a geospatial data service. The TDS combines THREDDS catalog services with integrated data-serving capabilities, including OPeNDAP and OGC WCS (Web Coverage Service), and automatic catalog generation. While for the second platform in Figure 6.5(b) (hereafter platform-2), three VMs equipped with GeoSPA nodes and HBase were connected to form a HyperCGSF system. Both of them applied the Nginx as the load balancing and proxy server for simulating a high concurrency environment on the cloud.

Figure 6.6 and Figure 6.7 shows the testing results of these two platforms using different request rates from 30 to 150, over geographical scope 10°×10° and 20°×20° degree with 4km spatial resolution, respectively.

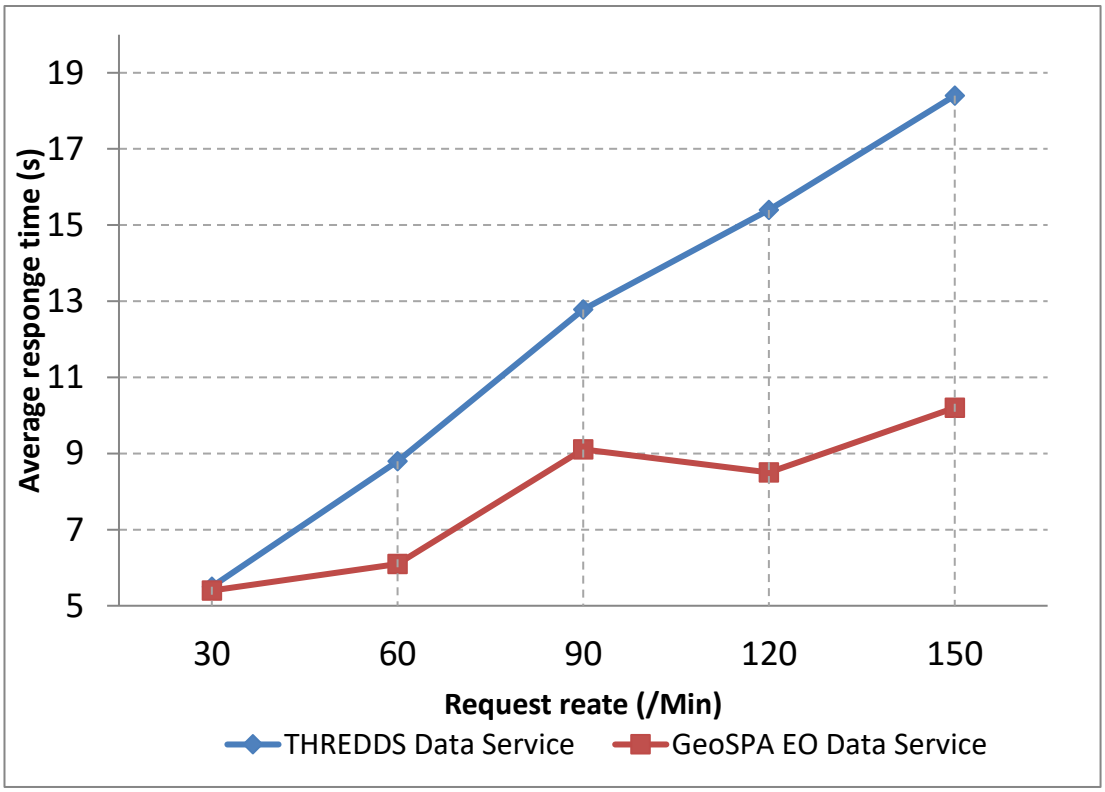


Figure 6.6: Comparison of response time using different data service (requesting domain size: 10°×10°)

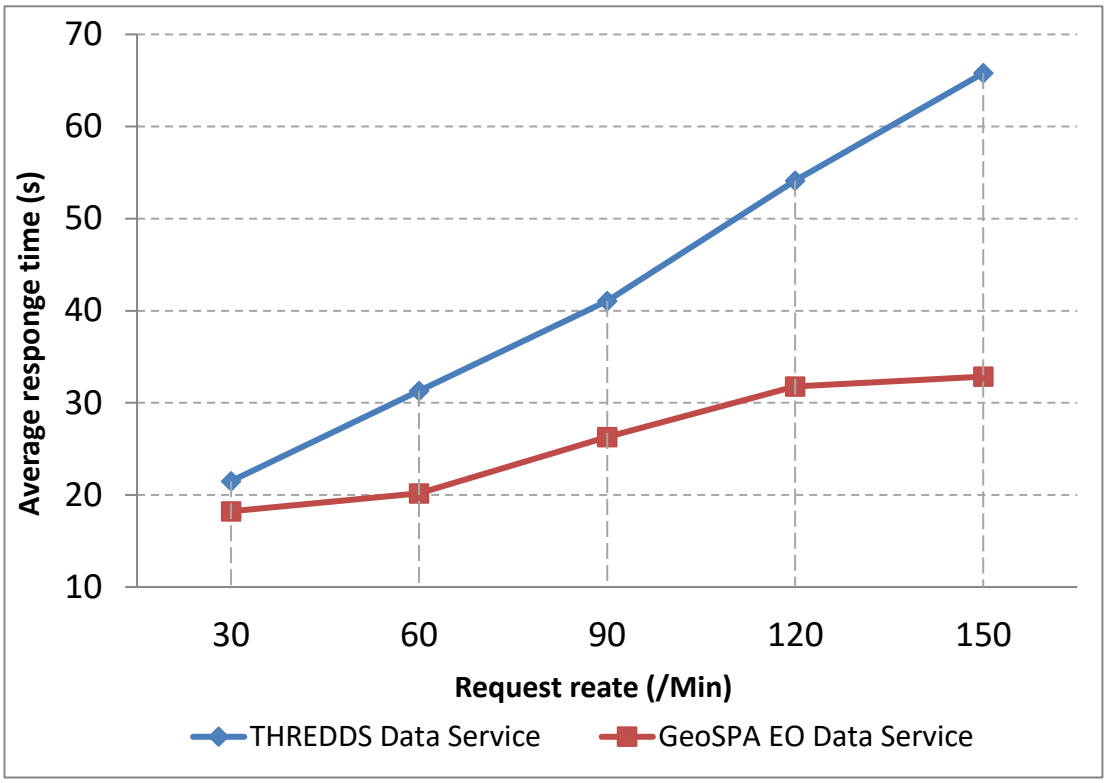


Figure 6.7. Comparison of response time using different data service (requesting domain size: 20°×20°)

Several conclusions can be drawn based on Figure 6.6 and Figure 6.7. First, the response time for both TDS and GeoSPA increase with the number of current requests. For example, as shown in Figure 6.7, the response time for these two methods increases from about several seconds to approximately one minute when the number of requests per minute increases from 30 to 150. This is because the processing capabilities of both platforms are limited. With the increase of request numbers, both platforms need to spend more time to generate the required dataset and send them to the service consumers.

Second, the response time of platform-2 is less than the platform-1 for every request rate, and the increasing rate of average response time for platform-2 is also lower than platform-1. This is because upon receipt of the incoming request, the platform-1 needs to operate on the original NetCDF file to obtain the required dataset. Loading the complete dataset is time-consuming and unnecessary. On the contrary, the platform-2 supplies tile-based data service, which only loads the tiles that fulfills

the user's request, and these tiles are stored in the local caching system. If the same tile is requested, the platform-2 picks it up directly from the caching system and sent back to the service consumer rather than retrieve it from HBase again via the spatial query, which dramatically improves the efficiency of the EO data service.

6.4 Evaluation of GeoSPA Processing and Computing Services

In the second part of our experiment, the performance of HyperCGSF was evaluated by comparing it with the traditional BEPL-based WPS service composition (BPEL-WPS) (Yu et al., 2012) approach in experimental tests. Figure 6.8 illustrates the structure of these two systems.

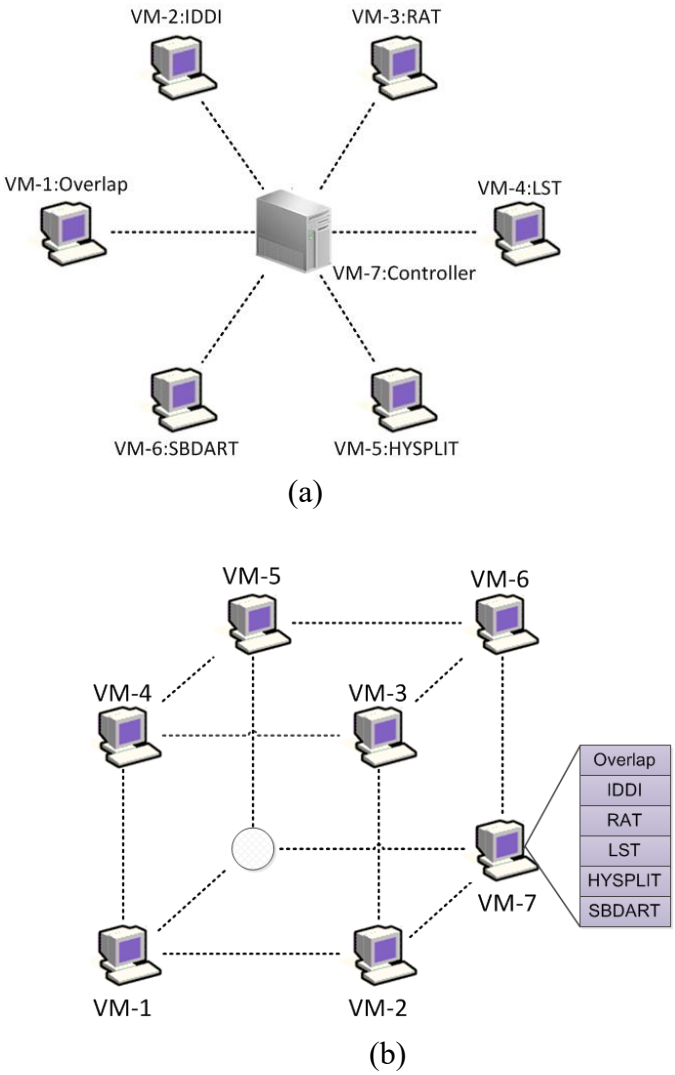


Figure 6.8. Structure of (a) centralized BPEL and (b) decentralized HyperCGSF

Figure 6.8(a) illustrates the structure of the centralized workflow management system. In this paradigm, one GCE VM instance is configured as the central workflow server taking charge as the orchestrator of the overall process, while six GCE VM instances, equipped with the PyWPS to offer web-based geospatial processing services based on the OGC WPS specification, were configured as the service provider. The potential user needs to write a BPEL script describing the workflow and submit it to the BPEL Engine for execution. Then the BPEL Engine starts to execute this workflow by calling corresponding services distributed over the six service providers and coordinate their execution. The workflow engine must communicate with each service provider, deliver the necessary information and retrieve the outcome of each task.

Figure 6.8(b) shows the structure of presented HyperCGSF with seven GeoSPA nodes (one node as geospatial processing service provider and the other six as both the computing service provider and data service provider) in this study. Distinguished with Figure 6.8(a), this paradigm supports the execution of the plan in a decentralized manner, where there is no central orchestrator, and all the nodes can interact with each other to exchange data directly.

6.4.1 Percentage of Response Time for Single Request Processing

The consumed time at different stages of the life cycle of the HyperCGSF-based geospatial processing service composition was recorded and analyzed. The response time of the life cycle for a processing service composition T can be described as:

$$T = T_{parse} + T_{dispatch} + \sum_{i=1}^{n} \left(T_{init_i} + T_{execute_i} \right) \quad (7)$$

where T_{parse} represents time of parsing embedded the XML-encoded WPS request document, $T_{dispatch}$ represents the time of dynamic task dispatching task,

T_{init} represent the time of initializing geospatial processes, and $T_{execute}$ represents the time of executing the whole workflow. The timing of T_{parse} and $T_{execute}$ is relatively stable in that all of the processes were performed on local machines. However, the times of $T_{dispatch}$ and T_{init} are unstable because of the uncontrolled network conditions, especially the T_{init} procedure, which includes the time of migrating geospatial processes among distributed GeoSPAs and preparing the input parameters of each process. The results were converted into percentages of the processing time and shown in Figure 6.9.

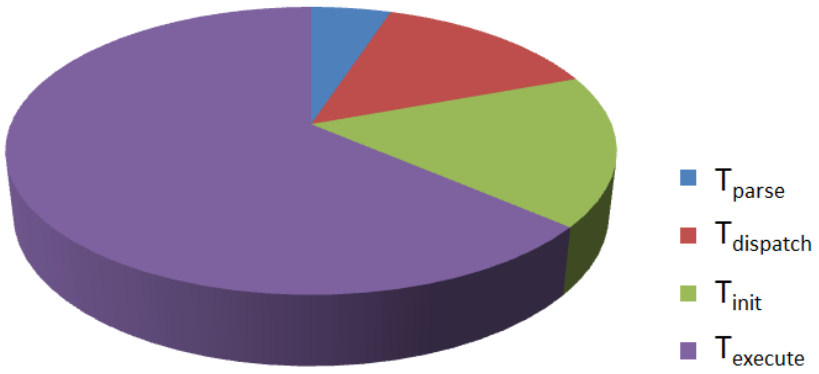


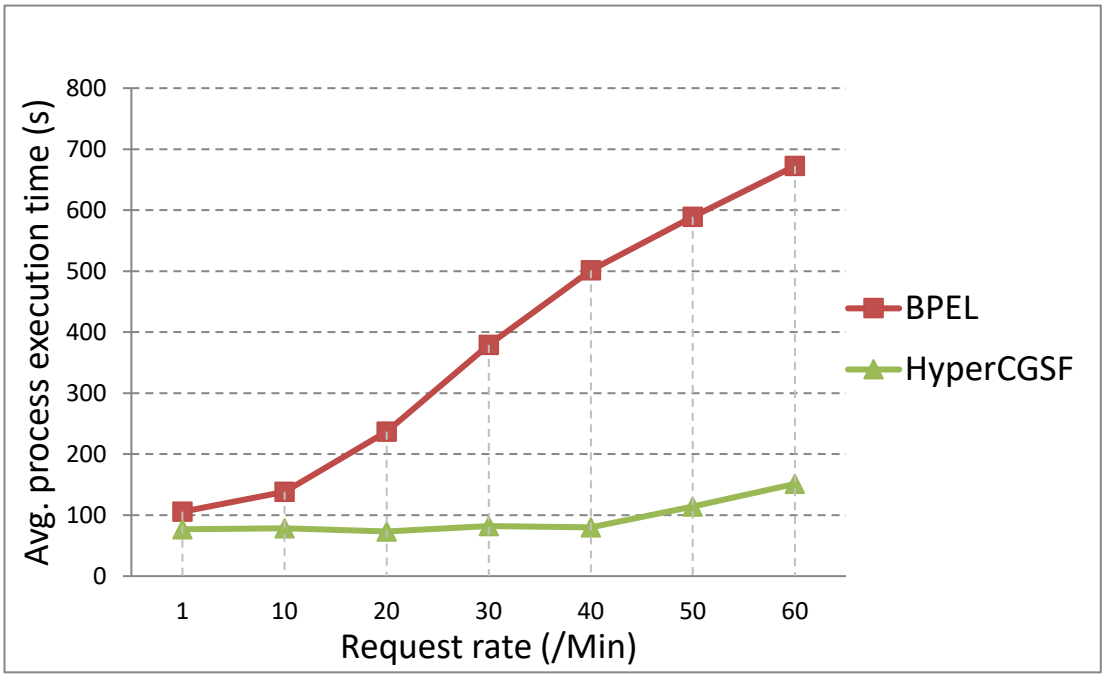
Figure 6.9. Percentage of response time in stages of IDDM workflow

As shown in Figure 6.9, for each HyperCGSF-based geospatial processing service composition lifecycle, the average time of T_{parse} is about 4.3s and occupies about 5% of the time, the average time of $T_{dispatch}$ is about 12.2s occupies about 14.3%, the average time of T_{init} is about 14.4s and occupies about 16.9%, and the average time $T_{execute}$ is about 54.2s and occupies about 63.8%. It can be concluded that the time spent on process execution occupies the largest part of the whole processing time (about 63.8%). This result is reasonable for that the execution of each geospatial models and the data interchange are time-consuming and computing-intensive.

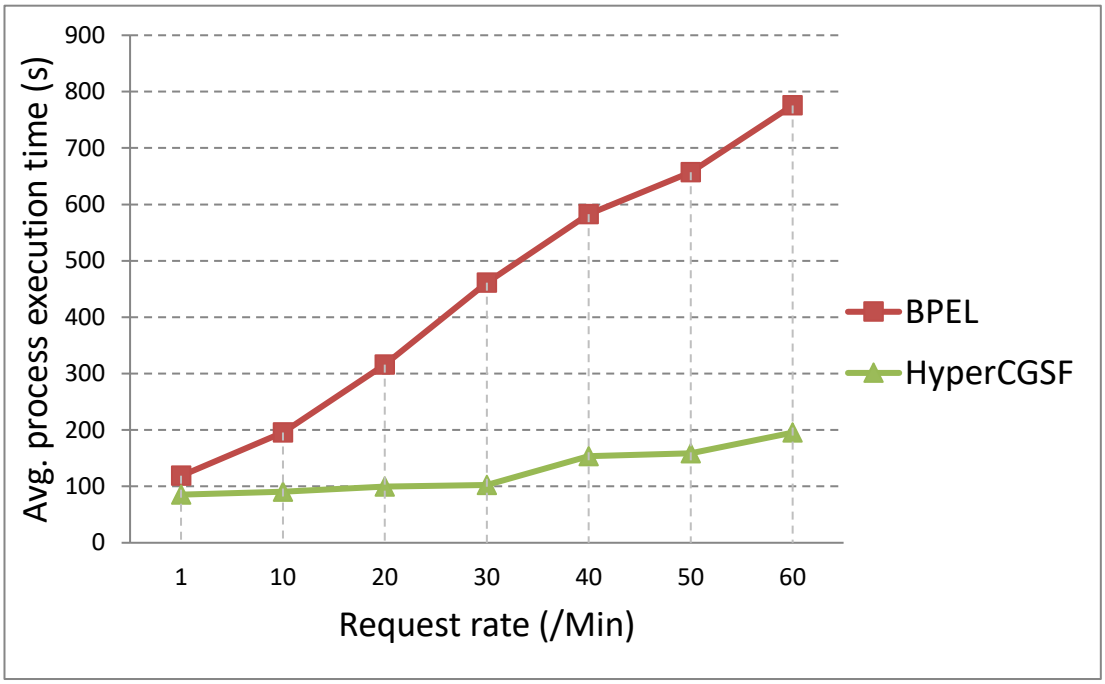
6.4.2 Varying Request Rate and Domain Size

To evaluate the efficiency of the algorithms in the presence of many

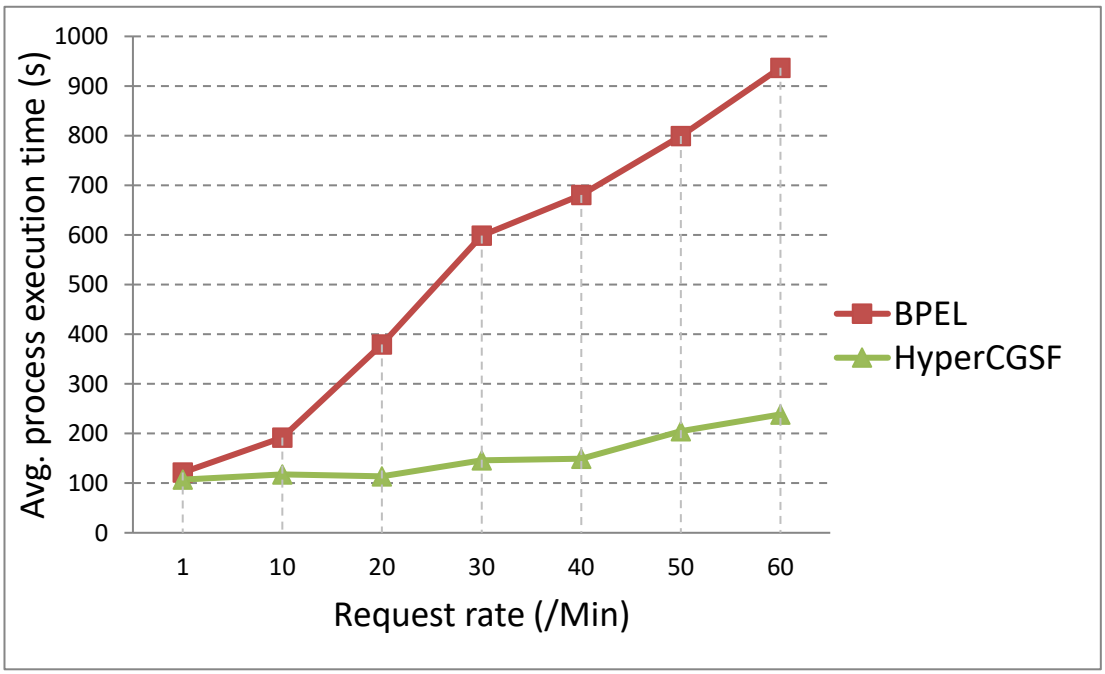
simultaneous accesses, the average execution time of IDDM workflow was recorded and compared using both the HyperCGSF and the traditional BEPL-based WPS service composition (BPEL-WPS) approach (Meng et al., 2009). The objective of this experiment was to evaluate how the average execution time varies with the increment of domain size and request rate. The request rate is the number of incoming service composition requests every minute. Figure 6.10 shows the experiment results using different request rates from 1 to 60, over geographical scope of 10°×10° (Figure 6.10(a)), 20°×20° (Figure 6.10(b)), and 30°×30° (Figure 6.10(c)) with 4km spatial resolution.



(a) Domain size: 10°×10°



(b) Domain size: 20°×20°



(c) Domain size: 30°×30°

Figure 6.10. Comparison of average process execution time using different service composition methods for the IDDM

Several conclusions can be drawn from Figure 6.10. First, the average execution time of BPEL-WPS and HyperCGSF increases dramatically with the number of current requests. For example, the response time for these two methods increases

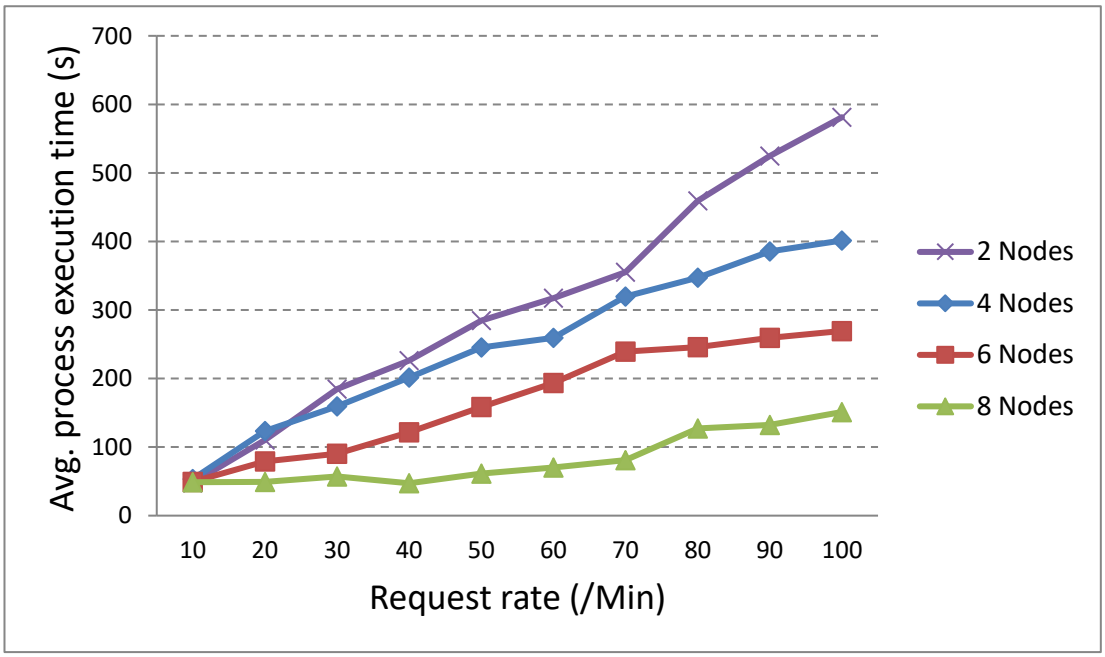
from about several minutes to approximately one hour when the number of requests per minute increases from 1 to 60. That is because before model execution, the service agent needs to read geospatial data with large volume from remote sites as the input parameters.

Second, the response time of HyperCGSF is less than the traditional BPEL-WPS approach for every request number, and the increasing rate of execution time for HyperCGSF is also lower than BPEL-WPS. The test result is reasonable. The BPEL-WPS approach applies the centralized manner that the interaction and data exchange movements are conducted through the orchestrator, or workflow execution engine. The geospatial processes can generate a lot of data that is irrelevant to the composite service, yet this data will be transferred to the coordinator node where it is discarded, thereby putting an unnecessary load on the network. Different from BPEL-WPS, the HyperCGSF applies the decentralized architecture in the way the service agents can communicate directly with each other to exchange processing results on demand.

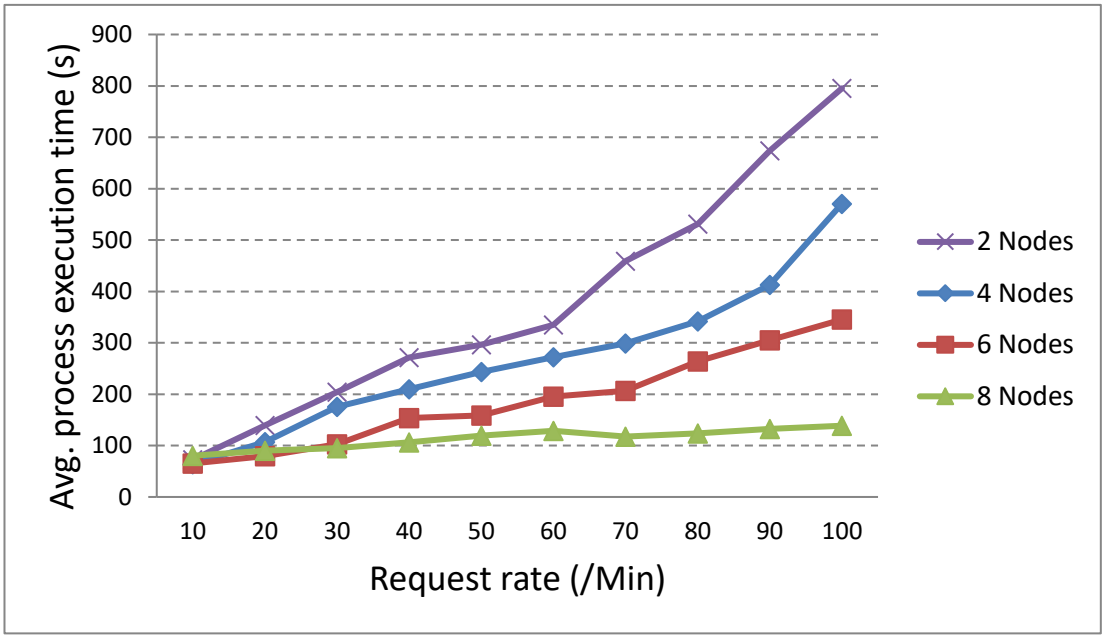
One of the most advanced features of HyperCGSF is that it supports the migration of geospatial processes among various GeoSPAs. This feature is extremely useful for geoscience because the geospatial data is always Big Data and distributed on remote sites. Considering that the geoscience applications always need to process large volumes of geospatial data, transferring the geospatial processes rather than the geospatial data over a cloud computing environment is significant in that it can dramatically decrease the volume of data transmission and increase the computing efficiency. Several studies have shown the advantages of applying the migration of the service agent in geospatial model services (Tan et al., 2015). However, some security issues must be taken into consideration before migrating a geospatial process from one GeoSPA node to another.

6.4.3 Varying HyperCGSF Nodes Number

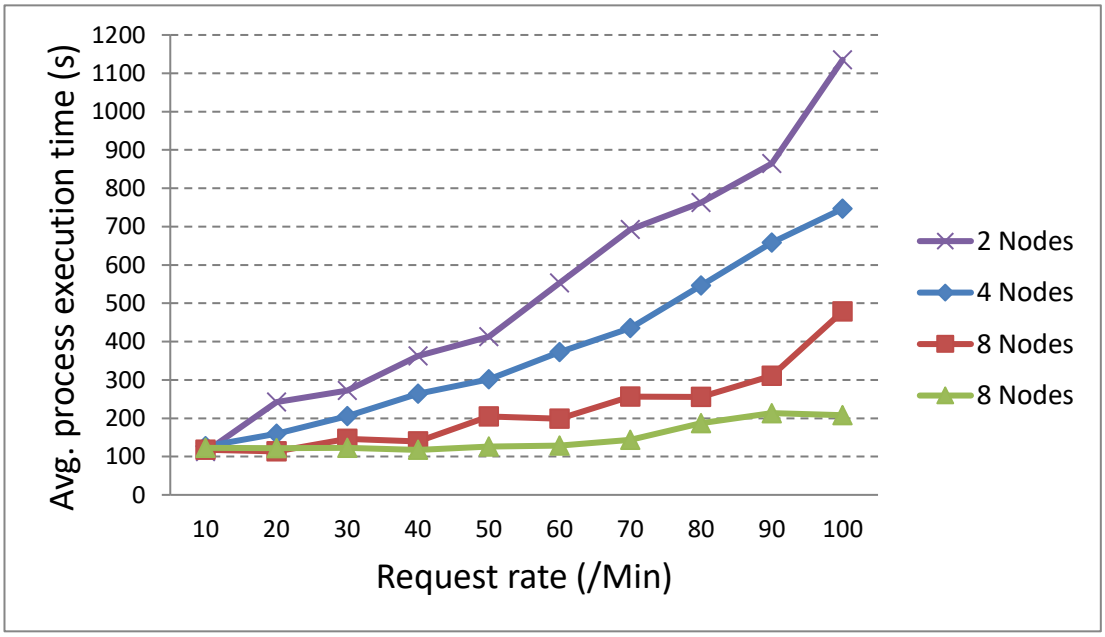
In this section, the efficiency of HyperCGSF was further evaluated by analyzing how the average process time varies with the number of nodes at different request rates. Four HyperCGSF (*N*=2, 5, 6, and 8) systems were constructed and evaluated respectively in this test. Figure 6.11 shows the experiment results using different request rates of 10 to 100, over geographical scope 10×10, 20×20, and 30×30 degree with 4km spatial resolution.



(a) Domain size: 10°×10°



(b) Domain size: 20°×20°



(c) Domain size: 30°×30°

Figure 6.11. Comparison of average process execution time with the increasing of HyperCGSF nodes for the IDDM

As the result in Figure 6.11 suggests, all four HyperCGSF nodes exhibited similar process times for rates up to 20 requests per minute, which is around 150 seconds. However, the performance of HyperCGSF with two nodes only was seriously degraded for larger request rates, and the processing time reached more

than 2000 seconds at the rate of 100 requests per minute, which was about 20 times longer than that of 10 requests per minute. In the case of 110 requests per minute, the 2-nodes HyperCGSF broke down. However, with the increment of HyperCGSF nodes, the performance improved dramatically, and the 8-nodes HyperCGSF managed a relatively steady average process execution time regardless of the request rate with a slight increase (29s) on the average process execution times from 10 requests per minutes to 120.

6.4.4 Test of Workload Distribution

The workload distribution was also monitored by counting the total number of processes that are assigned to each GeoSPA node over time. Figure 6.12 illustrates the distribution of geospatial processes across the nodes of the three dimensional HyperCGSF system with eight nodes for a 30-minute round of workflow execution. The request rate was set at 60/min.

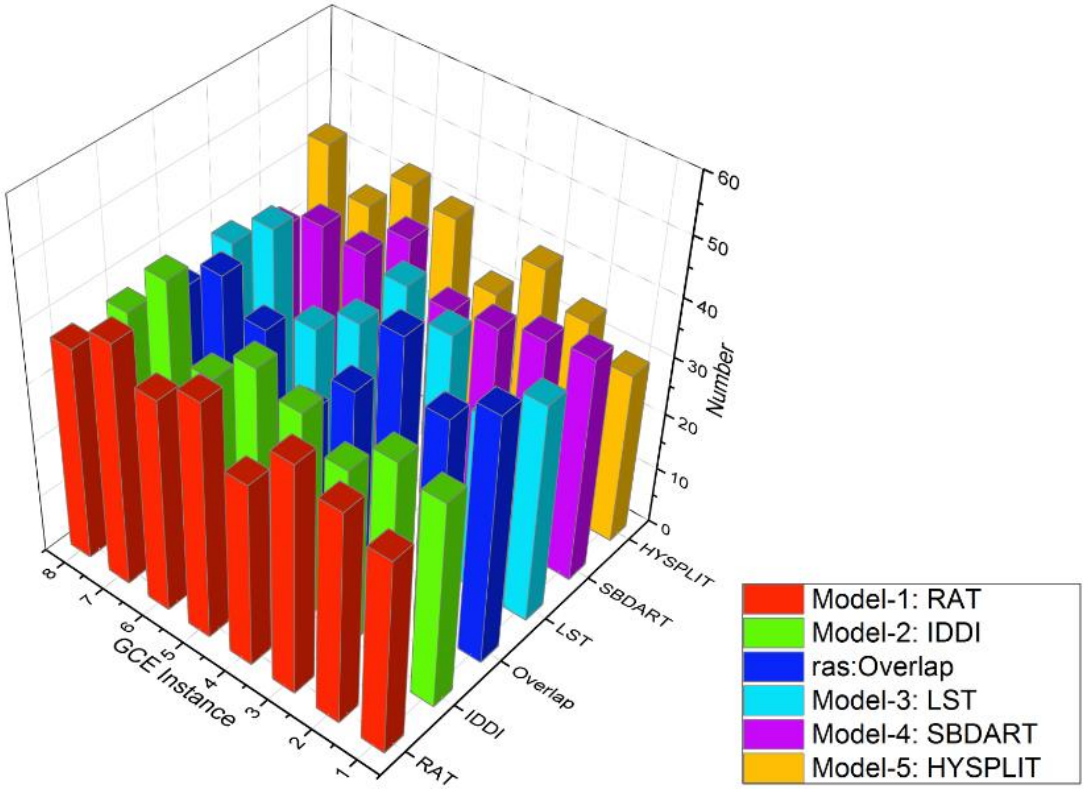


Figure 6.12. Distribution of geospatial processes across the nodes of the three dimensional HyperCGSF system with 8 nodes for the IDDM (duration=30 min, request rate=60/min)

As shown in Figure 6.12, our recruitment algorithm achieved a remarkably even distribution of the geospatial processes, exploiting all the available GeoSPA nodes and considering at the same time their frequency of use. Although the measurements revealed some deviation regarding the process number for each node, such a circumstance was anticipated because the GeoSPA node was randomly selected as the manager of the geospatial processes assignment for each request, and the node number (which is eight) is more than the process number (which is six), so we could not guarantee all of the nodes were assigned an equal number of processes.

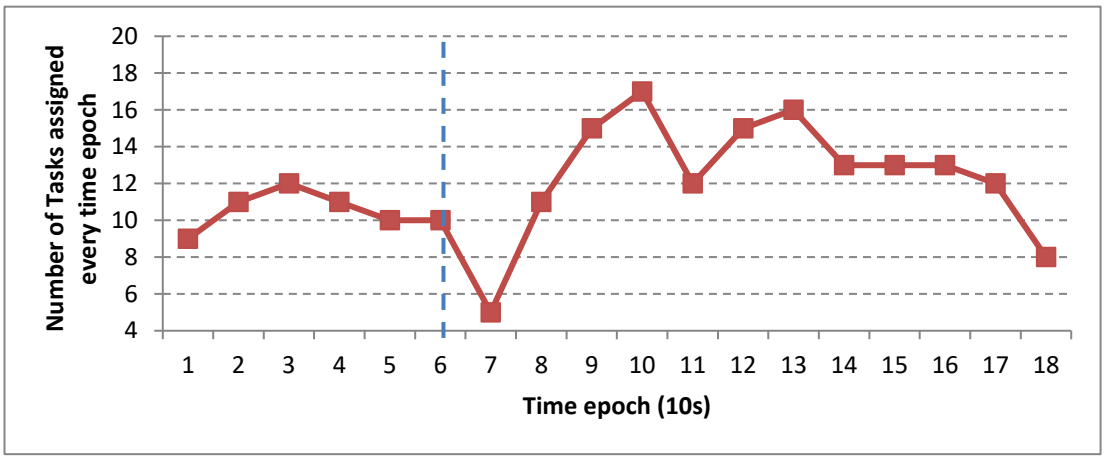
An import advantage of load balance in the HyperCGSF is that the computing capability of each GeoSPA node can be fully used. The geospatial processes are heterogeneous because every process requires various computing and storage resources. In traditional centralized structure, each server is responsible for processing certain types of geospatial process, which is not reasonable for the heterogeneity of the requirement of computing resources for each process. Distinguished by a centralized structure, the worker nodes in HyperCGSF have the equal role which means that each kind of geospatial process can be deployed on the GeoSPA node for execution. Through this approach, every GeoSPA can reach the highest working efficiency.

6.5 Test of Node Departure

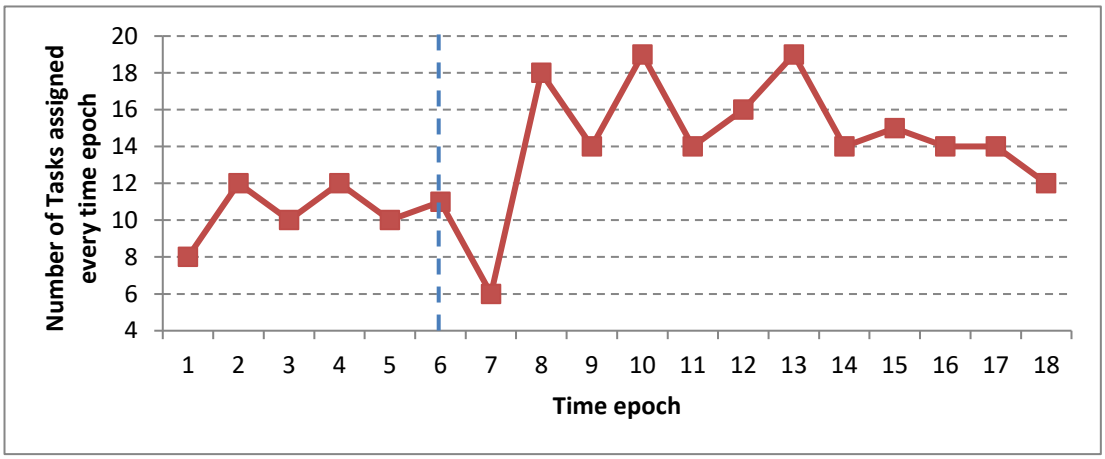
Finally, the scalability and stability of HyperCGSF were evaluated by simulating the departure of the GeoSPA node during the workflow execution. As introduced in Chapter 1, the web services are distributed across physical and geospatial boundaries in the cloud computing environment, and constantly removed and updated due to unpredicted factors such as network connection failure, server downtime, or hardware maintenance. As geospatial processes begin to rely on remote resources for their computation, they become more fragile and generating the desirable result at a given time could not be guaranteed. The objective of this

test is to illustrate that our system keeps running smoothly for the presence of node departure.

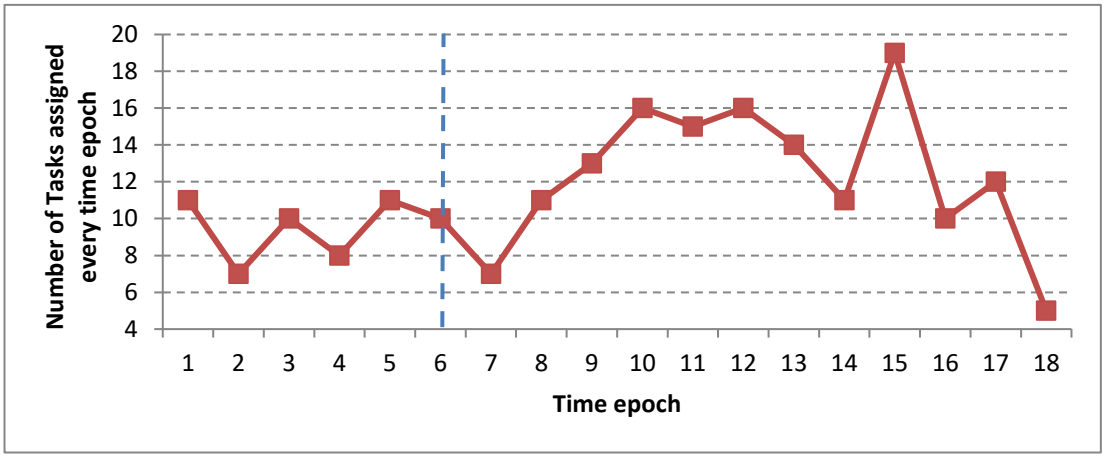
In this test, a HyperCGSF with six GeoSPA nodes was constructed for testing. The node of GCE-5 was simulated to depart after one-minute of running. The blue dotted line indicates the time of node departure. The temporal variation of workload at each GCE VM is recorded every 10 seconds, as depicted in Figure 6.13.



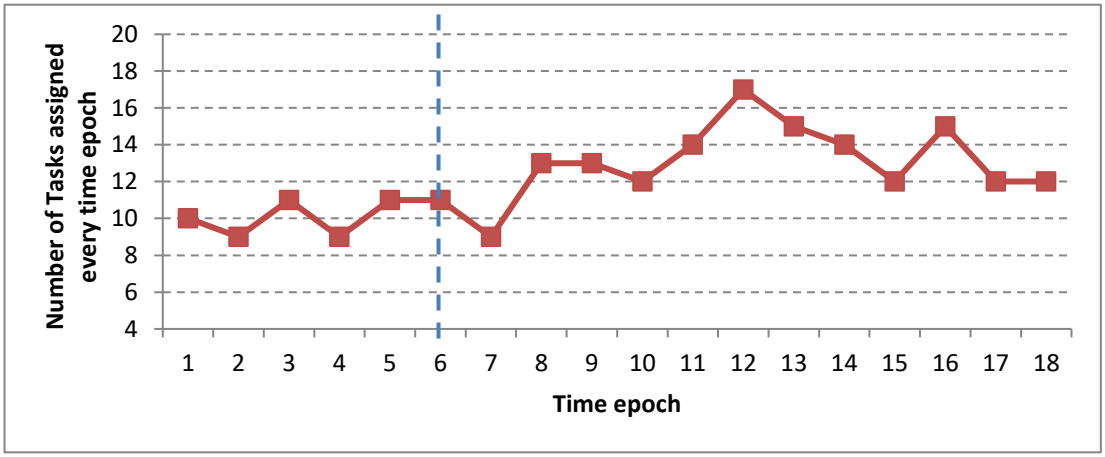
(a) Temporal variation of workload at GCE VM-1.



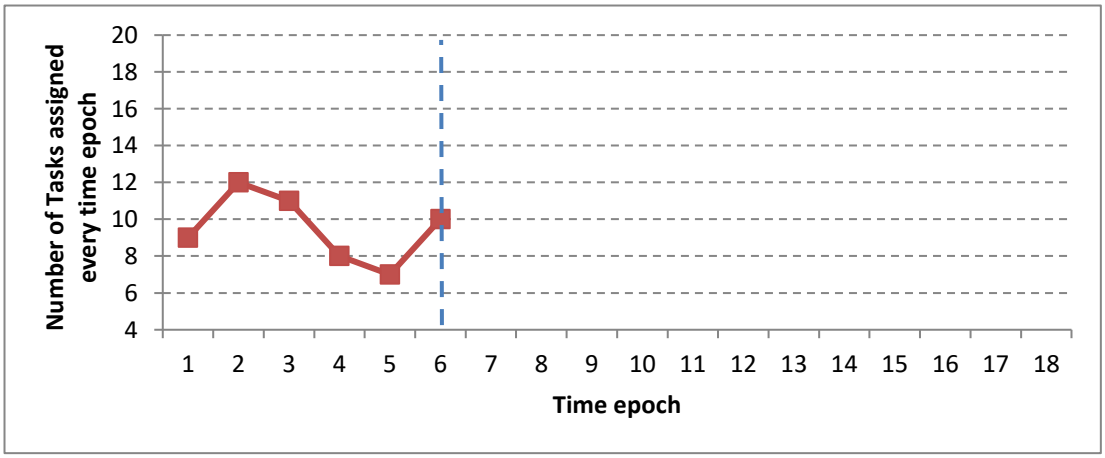
(b) Temporal variation of workload at GCE VM-2



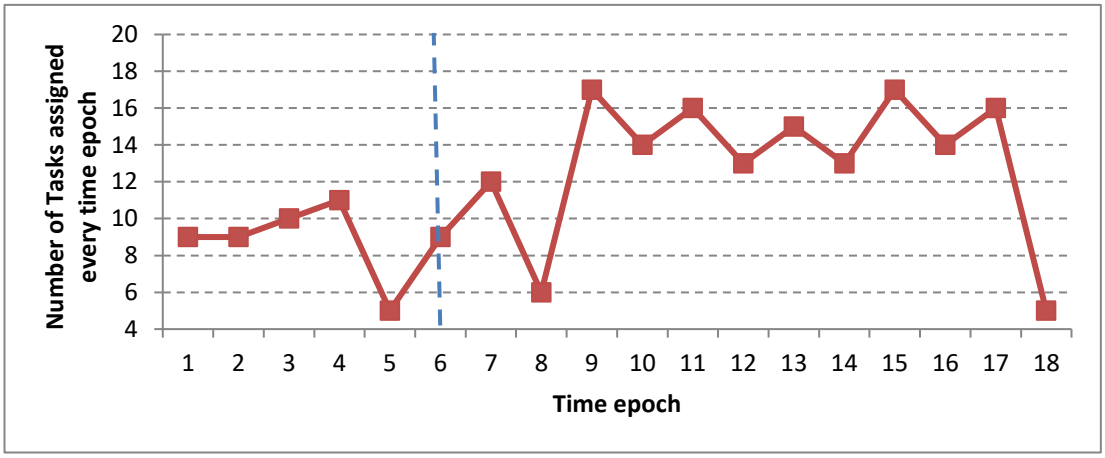
(c) Temporal variation of workload at GCE VM-3



(d) Temporal variation of workload at GCE VM-4



(e) Temporal variation of workload at GCE VM-5



(f) Temporal variation of workload at GCE VM-6

Figure 6.13. Illustration of the workload changing rate at each node for the presence of node departure

As shown in Figure 6.13, before the GCE-5 departed, the average number of processing tasks every 10 seconds is about 10, which indicates the processes were evenly distributed across all of the nodes. When GCE-5 departed, the average number of designed tasks of each node begins to increase (to about 14), which indicates the remaining nodes begin to take on more processing tasks due to the departure of GCE-5. It should be noted that at the time epoch of seven, the average number of processing tasks shows a relatively low value than at the other time for all of the five remaining nodes. This is because when the GCE-5 departed, the HyperCGSF began to reconstruct its topology and every node needs to spend extra computing resources to update its neighbor set and rebuild relations with them.

Another case is the unexpected node departure. If the HyperCGSF has been deployed to an open environment with a high churn rate, it is always possible that nodes go unexpectedly offline, without performing the HyperCGSF leave protocol. Under this circumstance, the other nodes will take the place of the vacant position eventually, leading to a stable and consistent topology. However, the HyperCGSF-specific data of the departed node including the information of processes deployed on this node will be missed. The only way to preserve the

HyperCGSF-specific data of the departed node is to enable replication. This feature will be developed in future work, as it comes with the price of increased network traffic and resources consumption.

Chapter 7: Conclusions and Future Work

7.1 Conclusions

Geoscience observations are generating vast amounts of multi-dimensional data. Effectively analyzing these data is a great challenge for geo-scientists. Cloud computing provides enabling capabilities for geosciences and Digital Earth in the twenty-first century. Studies for adopting cloud computing to enable or solve the geoscience problems and Digital Earth challenges have attracted many computational and geo-scientists to investigate the readiness of cloud computing. This study proposed the HyperCGSF, which is a decentralized geospatial processing service framework based on P2P technologies aims at managing geospatial processing services and service compositions in the cloud computing environment.

HyperCGSF is composed of GeoSPA, which was designed as a geospatial services hub through which the geospatial model developer can deploy standard-based geospatial services onto cloud system. GeoSPA supplies a series of algorithms for managing and discovering the geospatial services, as well as orchestrating the service composition execution. Based on P2P technologies and OGC Web Service specification, the HyperCGSF supports the distributed service discovery, automatic service composition and decentralized enactment of geospatial processing executions in the cloud computing environment. The HyperCGSF particularly focuses on the improvement of the average process execution times in the presence of multiple, concurrent and long-running process instances.

In order to provide a comprehensive overview of the HyperCGSF, some challenges of current distributed geospatial service technologies were first introduced in Chapter 1 and a literature review about cloud computing, EO sensor web, and cloud-based geospatial processing frameworks were given in Chapter 2. Then the HyperCGSF was introduced through two chapters. Chapter 3 introduced a

multifunctional geospatial server called GeoSPA, which has three service models: Earth observation (EO) data service model, geospatial processing service model, and computing service model. These three models make GeoSPA as a one-stop solution for building SDI in cloud computing environment. Chapter 4 introduced a P2P-based network topology called hypercube, which is used to organize multiple GeoSPAs into a decentralized and distributed geospatial data and processing service framework. Finally, the Integrated Dust Storm Detection Model was introduced in Chapter 5 as a study case to evaluate the efficiency of HyperCGSF.

HyperCGSF was evaluated through a series of experiments, and compared with traditional centralized architectures in term of performance in Chapter 6. The retrieved measurement indicate that our framework is more suitable for the execution of long-running and data intensive processes, while it is able to accommodate more concurrent clients than traditional centralized approach. Moreover, thanks to the even distribution of workload, our approach copes with large data in a more efficient manner.

To sum up, the major contributions of this thesis are as follows:

a) Based on the challenges and existing approaches in a general-purpose distributed geospatial service architecture were summarized, we identified two problems need to be resolved.

b) A multifunctional geospatial server called GeoSPA was developed, which has three configurable service models: EO data service model, geospatial processing service model, and computing service model. This structure makes GeoSPA scalable and easy to be deployed on the cloud computing environment as SDI service node.

c) The HyperCGSF was designed to organize multiple GeoSPAs aiming at achieving a more reliable and efficient decentralized geospatial processing

service framework. HyperCGSF applies a P2P-based network topology called hypercube, which possesses many scalability advantages over existing distributed computing architectures.

d) The presented distributed geospatial service planning algorithm (DGSPA) supports automatically build a geospatial processing service chain based on user-defined service requirement and the workflow execution is performed in a completely decentralized manner without the existence of a central coordinator.

7.2 Future Work

For future directions, it is believed that each of the identified challenges (in Chapter 1.2) is an interesting and important research question that is worth further investigation. In addition, as mentioned in Chapter 6, the quality optimization for composite Web Service and exception handling mechanism for the decentralized execution of the composite Web Service has not been sufficient studies. These topics are very important for that the cloud computing environment is very dynamic environment where the computational resources keep changing over time and have lower reliability. Furthermore, a more complex geospatial processing workflow needs to be designed to evaluate the proposed frameworks. And further research needs to be conducted to enhance our research to process dynamic partition and QoS constrained planning approach.

Another research direction is to integrate dynamic streaming data with my system. Streaming data is an important data source for Earth observation. The OGC Sensor Web Enablement (SWE) architecture could describe, discover and invoke services from different kinds of heterogeneous platforms by using SOAP and XML standards. This architecture has implemented the discovery, access, utilization and control of sensor resources via the web. So SWE could be an important dataset for GeoSPA EO data service.

In general, despite the fact that there are still many important topics and directions to be investigated, as one of the first distributed geospatial processing service frameworks based on hypercube topology, the proposed solutions and HyperCGSF architecture serve as a promising initiative to address the unique distributed geospatial processing challenges and consequently allow us to harvest the full potential of the cloud computing and EO sensor web.

References

Ackerman, S.A., 1997. Remote sensing aerosols using satellite infrared observations. Journal of Geophysical Research 102(D14), 17069–17079.

Akbar, M., Aliabadi, S., Patel, R., Marvin, W., 2013. Fully automated and integrated multi-scale forecasting scheme for emergency preparedness. Environmental Modelling& Software 39, 24-38.

Alexandrov, A., Ewen, S., Heimel, M., Hueske, F., Kao, O., Markl, V., Nijkamp, E., Warneke, D., 2011. MapReduce and PACT-comparing data parallel programming models. In Proceedings of the Conference Datenbanksysteme in Büro, Technik und Wissenschaft, BTW, GI, Bonn, Germany, pp. 25–44.

Ames, D.P., Michaelis, C., Anselmo, A., Chen, L., Dunsford, H., 2008. Map Window GIS. In Shekhar, S., Xiong, H. (Eds.), Encyclopedia of GIS. Springer US, Boston, MA, pp. 633-634.

Anselmi, J., Ardagna, D., and Cremonesi, P., 2007. A QoS-based selection approach of autonomic grid services. In Proceedings of the 2007 workshop on service-oriented computing performance. Aspects, issues, and approaches. Monterey, California, USA, pp. 1–8

Argent, R.M., 2005. A case study of environmental modelling and simulation using transplantable components. Environmental Modelling& Software 20 (12), 1514-1523.

Bao, H. H., and Dou, W. C., 2012. A QoS-aware service selection method for cloud service composition. In 2012 IEEE 26th international parallel and distributed processing symposium workshops and PhD Forum. New York: IEEE. pp. 2254–2261.

Barnes, N., 2010. Publish your computer code: it is good enough. Nature, 467, pp.753.

Barzegar, S., Davoudpour, M., Meybodi, M. R., Sadeghian, A., and Tirandazian, M., 2011. Formalized learning automata with adaptive fuzzy coloured petri net: an application specific to managing traffic signals. Scientia Iranica 18, 554–565.

Bastin, L., Cornford, D., Jones, R., Heuvelink, G.B.M., Pebesma, E., Stasch, C., Nativi, S., Mazzetti, P., Williams, A., 2013. Managing uncertainty in integrated environmental

modelling: the UncertWeb framework. Environmental Modelling& Software 39, 116–139.

Berners-Lee, T., Hall, W., Hendler, J., Shadbolt, N., Weitzner, D.J., 2006. Creating a science of the web. Science 313, 769–771.

Bodk, P., et al., 2010. Characterizing, modeling, and generating workload spikes for stateful services. In Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC'10, 6-11 June 2010, UC Berkeley, Berkeley, CA, USA, pp. 241-252.

Botts, M., Robin, A., 2007. Open Geospatial Consortium (OGC) document number:07-000.

Bratman, M. E., 1987. Intentions, Plans, and Practical Reason. Havard University Press, Cambridge, MA, USA.

Braun P. and Rossak W., 2005, Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit, Morgan Kaufmann Publıshers.

Brauner, J., Foerster, T., Schaeffer, B., Baranski, B., 2009. Towards a research agenda for geoprocessing services. In 12th AGILE International Conference on Geographic Information Science 2009 Leibniz University, Hannover, Germany, pp. 1-12.

Broring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., Liang, S., Lemmens, R., 2011. New generation sensor web enablement. Sensors 11, 2652-2699.

Bulatewicz, T., Andresen, D., Auvenshine, S., Peterson, J., Steward, D.R., 2013. A distributed data component for the Open Modeling Interface. Environmental Modelling& Software 57, 138-151.

Cannata, M., Molinari, M. E., Luan T. X., and Long N. H., 2012. Web processing services for shallow landslide. International Journal of Geo-informatics, 8 (1), 25–34.

Cao, J. et al., 2005. A multi-agent negotiation based service composition method for on-demand service. In Proceedings of International Conference of Services Computing (SCC) 1, 329-332.

Cao, J., Das, S. K., 2012. Mobile agents in networking and distributed computing. Wiley Series in Agent Technology, John Wiley & Sons, Inc., USA, pp. 134.

Castronova, A. M. and Goodall, J. L., 2010. A generic approach for developing process-level hydrologic modeling components. Environmental Modelling& Software, 25(7), 819-825.

Castronova, A.M., Goodall, J.L., 2013. Simulating watersheds using loosely integrated model components: evaluation of computational scaling using OpenMI. Environmental Modelling& Software 39, 304-313.

Chan, C. C., Chuang, K.J., Chen, W.J., Lee, C.T. and Peng, C.M., 2007. Increasing cardiopulmonary emergency visits by long range transported Asian dust storms in Taiwan. Environmental Research, 106, 393-400.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. 2006. Bigtable: A distributed storage system for structured data. In Proceedings of the 7th Conference on USENIX Symposium on Operating Systems Design and Implementation - Volume 7 (Seattle, WA, November 06 - 08, 2006). USENIX Association, Berkeley, CA, pp. 15-15.

Chen, D., et al., 2009. Regional CO pollution and export in China simulated by the high-resolution nested-grid GEOS-Chem model, Atmospheric Chemistry and Physics 9, 3825–3839.

Chen, F. and Dudhia, J., 2001. Coupling an advanced land surface hydrology model with the Penn State-NCAR MM5 modeling system. Part I: model implementation and sensitivity. Monthly Weather Review 129(4), 569-585.

Chen, N., Di, L., Yu, G., Gong, J., 2010. Geo-processing workflow driven wildfire hot pixel detection under sensor web environment. Computers & Geosciences 36, 362–372.

Chen, N., X. Chen, K. Wang and X. Niu , 2014. Progress and challenges in the architecture and service pattern of Earth Observation Sensor Web for Digital Earth. International Journal of Digital Earth 7(12), 935-951.

Chen, N., Zhen, Z., Di, L., Yu, G., Zhao, P., 2009. Resource oriented architecture for heterogeneous geo-processing workflow integration. In Proceedings of 17th International Conference on Geo-informatics. Fairfax, VA. pp. 1-5.

Chen, X. and Yang, C.W., 2014. Introduction to big geospatial data research. Annals of GIS 20(4), 227-232.

Chiapello, I., Prospero, J. M., Herman, J., and Hsu, C., 1999. Detection of mineral dust over the North Atlantic Ocean and Africa with the Nimbus 7 TOMS, Journal of Geophysical Research 104, 9277–9291.

Christian, E. 2005. Planning for the Global Earth Observation System of Systems (GEOSS). Space Policy 21 (2), 105-109.

Clery, D., Voss, D., 2005. All for one and one for all. Science 308 (5723), 809.

Collins, N., Theurich, G., Deluca, C., Suarez, M., Trayanov, A., Balaji, V., Li, P., Yang,W., Hill, C., and Da Silva, A., 2005. Design and implementation of components in the Earth System Modeling Framework. International Journal of High Performance Computing Applications 19 (3), 341-350.

Craglia, M., Bie, K. de, Jackson, D., Pesaresi, M., Remetey-Fülöpp, G., Wang, C., Annoni, A., Bian, L., F. Campbell, M. Ehlers, J. van Genderen, Goodchild, M., Guo, H., Lewis, A., Simpson, R., Skidmore, A., and Woodgate, P., 2011. Digital Earth 2020: towards the vision for the next decade. International Journal of Digital Earth 5(1), 4-21

David, J.M., 2005. Towards a GIS platform for spatial analysis and modeling. In Maguire, D.J., Batty, M., Goodchild, M.F. (Eds.), GIS, Spatial Analysis, and Modeling. ESRI Press, Redlands, C.A., U.S., pp. 19-39.

Deelman, E., Gannon, D., Shields , M., Taylor, I., 2010. Workflows and e-science: An overview of workflow system features and capabilities, Future Generation Computer Systems 25, 528–540.

Delin, K.A., and Jackson, S.P., 2000. Sensor web for in situ exploration of gaseous bio signatures. In Proceedings of the 2010 IEEE Aerospace Conference 7, 465-472.

Demir, I., Krajewski, W.F., 2013. Towards an integrated flood information system: centralized data access, analysis, and visualization. Environmental Modelling & Software 50, 77-84.

De Jesus, J., Walker, P., Grant, M., and Groom, S., 2012. WPS orchestration using the Taverna workbench: The eScience approach. Computers & Geosciences 47(0), 75-86

Di, L., K. Moe, and Van Zyl, T. L., 2010. Earth Observation Sensor Web: An overview. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 3 (4), 415-417.

Dijkstra, J., Timmermans, H.J.P., and Jessurun, A. J., 2001. A multi-agent cellular automata system for visualizing simulated pedestrian activity. Theory and Practical Issues on Cellular Automata. Springer London, 29-36.

Dimiduk, N., Khurana, A., 2013. HBase in Action, O'Reilly Media, Sebastopol, Calif, USA, pp. 254.

Domenico, B., Caron, J., Davis, E., Kambic, R., and Nativi,S., 2002. Thematic Real-time Environmental Distributed Data Service (THREDDS): incorporating interactive analysis tools into NSDL, Journal of Digital Information Management, 2(4), 29.

Donner, R., et al., 2009. Understanding the Earth as a complex system recent advances in data analysis and modelling in Earth sciences. European Physical Journal Special Topics 174, 19.

Dubois, G., Schulz, M., Skøien, J., Bastin, L., Peedell, S., 2013. eHabitat, a multipurpose Web Processing Service for ecological modeling. Environmental Modelling & Software 41 (3), 123-133.

Ellrod, G.P., 2001. Loss of the 12 mm "split window" band on GEOS-M: impacts on volcanic ash detection. Paper Presented at 11th Conference on Satellite Meteorology and Oceanography. American Meteorological Society, Madison, Wisc., USA, pp. 15-18.

Erl, T., 2005. Service-Oriented Architecture:Concepts, Technology, and Design. The Prentice Hall Service-Oriented Computing Series from Thomas Erl. Prentice Hall, pp.205.

Feng, M., Liu, S., Euliss Jr, N. H., Young, C., and Mushet, D. M., 2011. Prototyping an online wetland ecosystem services model using open model sharing standards. Environmental Modelling & Software 26(4), 458-468

Fenoy, G., Bozon, N., Raghavan, V., Jan. 2012. ZOO-Project: the open WPS platform. Applied Geomatics 5 (1), 19-24.

Fielding R.T., 2000. Architectural styles and the design of network-based software architectures. Ph.D. Thesis, University of California. Irvine, USA.

Finney, K.T., Watts, D., 2011. REST-based semantic feature catalogue services. International Journal of Geographical Information Science 25 (9), 1507-1524.

Foerster, T., Brühl, A., Schäffer, B., 2011. RESTful web processing Service. In Proceedings of the AGILE 2011 Conference. Utrecht, The Netherlands, pp. 8.

Folino, G., Forestiero, A., G. Papuzzo, G. Spezzano, A Grid portal for solving geoscience problems using distributed knowledge discovery services. Future Generation Computer Systems 26 (2010), 87–96.

Friis-Christensen, A., Lucchi, R., Lutz, M. and Ostländer, N., 2009. Service chaining architectures for applications implementing distributed geographic information processing. International Journal of Geographical Information Science 23(5), 561-580.

Gaber, N., Laniak, G., Linker, L., 2008. Integrated modeling for integrated environmental decision making. EPA White paper. 100/R, pp. 8-10.

Garrison, V.H., Shinn, E.A., Foreman, W.t., Griffin, D.W., Holmes, C.W., Kellogg, C.A., Majewski, M.S., Richardson, L.L., Ritchie, K.B., Smith, G.W., 2003. African and Asian dust: from desert soil to coral reefs. BioScience 53, 469-480.

Geller, G.N, Turner, W., 2007. The model web: A concept for ecological forecasting, Proceedings of the Geoscience and Remote Sensing Symposium (IGARSS 2007). IEEE International, pp. 2469-2472.

Geller, G. N., and Melton, F., 2008. Looking forward: Applying an ecological model web to assess impacts of climate change. Biodiversity 9(3–4), 79–83.

Gesell, G., 1989. An algorithm for snow and ice detection using AVHRR data: An extension to the APOLLO software package. International Journal of Remote Sensing 10, 897–905.

Ghemawat, S., Gobioff, H., and Leung, S.-T. 2003. The Google file system. In 19th Symposium on Operating Systems Principles. Lake George, NY, USA, pp. 29-43.

Goodall, J.L., Robinson, B.F., Shatnawi, F.M., Castronova, A.M., 2008. Linking hydrologic models and data: The OpenMI approach. American Geophysical Union, San Francisco, US, pp. 15-19.

Goodall, J.L., Robinson, B.F., Castronova, A.M., 2011. Modeling water resource systems using a service-oriented computing paradigm. Environmental Modelling& Software 26 (5), 573-582.

Goodchild, M., 2007. Citizens as sensors: the world of volunteered geography. GeoJournal 69(4), 211-221.

Grell, G. A., 1993. Prognostic evaluation of assumptions used by cumulus parameterizations. Monthly Weather Review 121(3), 764-787.

Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: Open Modeling Interface. Journal of Hydroinformatics 9(3), 175-191.

Granell, C., Diaz, L., Gould, M., 2010. Service-oriented applications for environmental models: reusable geospatial services. Environmental Modelling & Software 25(2), 182-198.

Group on Earth Observations, 2005. The Global Earth Observation System of Systems (GEOSS) 10-Year Implementation Plan. http://www.earthobservations.org/documents/10-Year_Implementation_Plan.pdf

Gu, Y.X., Rose, W.I., Bluth, G.J.S., 2003. Retrieval of mass and sizes of particles in sandstorms using two MODIS IR bands: A case study of April 7, 2001 sandstorm in China. Geophysical Research Letters 30, 1805.

Guan, J., Wang, L., Zhou, S., 2004. Enabling GIS services in a P2P environment. In Das, G., Gulati, V.P. (eds.) CIT 2004. LNCS, Springer, Heidelberg 3356, 776–781.

Gutierrez-Garcia, J. O., and Sim, K. M., 2010. Agent-based service composition in cloud computing. In T. H. Kim, S. S. Yau, O. Gervasi, B. H. Kang, A. Stoica, and Slezak, D. (Eds.). Grid and Distributed Computing, Control and Automation. Berlin: Springer-Verlag Berlin, pp, 1–10.

Gutierrez-Garcia, J. O., and Sim, K., 2013. Agent-based cloud service composition. Applied Intelligence 38, 436–464.

Gruber, T.R., 1993. A translation approach to portable ontology specification. Knowledge Acquisition 5 (2), 199–220.

Han, H.-J., Sohn, B.J., 2013. Retrieving Asian dust AOT and height from hyperspectral sounder measurements: An artificial neural network approach. Journal of Geophysical Research: Atmospheres 118, 837–845.

Hamdaqa, M., and Tahvildari, L., 2012. Cloud computing uncovered: a research landscape. in H. Ali & M. Atif (Eds.), Advances in Computers 86, 41–85.

Hess, M., Koepke, P., and Schult, I., 1998. Optical properties of aerosols and clouds: the software package OPAC, Bull. American Meteorological Society 79(5), 831–844.

Hill, C., DeLuca, C., Balaji, V., Suarez, M., Da Silva, A., 2004. The architecture of the earth system modeling framework. Computing in Science and Engineering 6 (1), 18-28.

Hill, C., DeLuca, C., Balaji, V., Suarez, M., Da Silva, A., Sawyer, W., Cruz, C., Trayanov, A., Zaslavsky, L., Hallberg, R., Boville, B.A., Craig, A., Collins, N., Kluzek, E., Michalakes, J., Neckels, D., Schwab, E., Smithline, S., Wolfe, J., Iredell, M., Yang, W., Jacob, L.R., Larson, J.W., 2006. Implementing applications with the Earth System Modeling Framework. Lecture Notes in Computer Science 3732, 563-572.

Hofer, B., 2014. Uses of online geoprocessing technology in analyses and case studies: a systematic analysis of literature. International Journal of Digital Earth 8(11), 901-917.

Hong, S. Y., Noh, Y. and Dudhia, J., 2006. A new vertical diffusion package with an explicit treatment of entrainment processes. Monthly Weather Review, 134(9), 2318-2341.

Hu, C.L., and Chen, N., 2011. Geospatial sensor web for smart disaster emergency processing. In Proceedings of the 19th International Conference on Geo-informatics, Shanghai, China, pp. 1-5.

Hu, X.Q., Lu, N.M., Niu, T., Zhang, P., 2008. Operational retrieval of Asian dust storm from FY-2C Geostationary Meteorological Satellite and its application to real time forecast in Asia. Atmospheric Chemistry and Physics 8, 1649-1659.

Huang, Q., Yang, C., Benedict, K., Chen, S., Rezgui, A., Xie, J., 2012. Utilize cloud computing to support dust storm forecasting. International Journal of Digital Earth 6 (4), 338–355.

Husar, R.B., Tratt, D.M., Schichtel, B.A., Falke, S.R., Li, F., Jaffe, D., Gasso, S., Gill, T., Laulainen, N.S., Lu, F., Reheis, M.C., Chun, Y., Westphal, D., Holben, B.N., Gueymard, C., McKendry, I., Kuring, N., Feldman, G.C., McClain, C., Frouin, R.J., Merrill, J., Dubois, D., Vignola, F., Murayama, T., Nickovic, S., Wilson, W.E., Sassen, K., Sugimoto, N., Malm, W.C., 2001. Asian dust events of April 1998. Journal of Geophysical Research 106(D16), 18317–18330.

INSPIRE, 2007. INSPIRE EU Directive, Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE). Official Journal of the European Union, L 108/1 50.

Jaber, A., Guarnieri, F., and Wybo, J., 2001. Intelligent software agents for forest fire prevention and fighting. Safety Science 39(1), 3–17.

Janowicz, K., Broering, A., Stasch, C., Schade, S., Everding, T., Llaves, A., 2012. A RESTful proxy and data model for linked sensor data. International Journal of Digital Earth 6(3), 233-254.

Jiang, J.R., 2011. Nondominated local coteries for resource allocation in grids and clouds. Information Processing Letters 111, 379-384.

Jiang, H., Kwong, C. K., Chen, Z., and Ysim, Y. C., 2012. Chaos particle swarm optimization and T-S fuzzy modeling approaches to constrained predictive control. Expert Systems with Applications 39, 194–201.

John C. Schaake, Thomas M. Hamill, Roberto Buizza, and Martyn Clark, 2007: HEPEX: The Hydrological Ensemble Prediction Experiment. Bulletin of the American Meteorological Society 88,1541–1547.

Jula, A., Sundararajan, E. and Othman, Z., 2014. Cloud computing service composition: A systematic literature review. Expert Systems with Applications, 41(8), 3809-3824.

Keating, T., 2009. Cyberinfrastructure for air quality management, EPA internal document, pp. 10.

Kido, M. H., Mundt, C. W., Montgomery, K. N., Asquith, A., Goodale, D. W., Kaneshiro, K. Y., 2008. Integration of wireless sensor networks into cyberinfrastructure for monitoring Hawaiian "mountain-to-sea" environments. Environmental Management 42(4), 658–666.

Kiehle, C., Greve, K., Heier, C., 2007. Requirements for next generation spatial Data Infrastructures - standardized Web based geoprocessing and Web Service orchestration. Transactions in GIS 11(6), 819–834.

Kofler, K., ul Haq, I., and Schikuta, E., 2009. A parallel branch and bound algorithm for workflow QoS optimization. In Parallel Processing 2009, ICPP International Conference, 478–485.

Kofler, K., Haq, I. U., and Schikuta, E., 2010. User-centric, heuristic optimization of service composition in clouds. LNCS 6271, 405–417.

Korte, B., and Vygen, J., 2012. Linear Programming (21). Berlin Heidelberg: Combinatorial Optimization Springer, pp. 51-71.

Koshy, T., 2004. Chapter 11 – formal languages and finite-state machines. In Discrete Mathematics With Applications. Burlington: Academic Press, pp. 733–802.

Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., G. Geller, Quinn, N., M. Blind, Peckham, S., Reaney, S., Gaber, N., Kennedy, R. and Hughes, A., 2013. Integrated environmental modeling: A vision and roadmap for the future. Environmental Modelling & Software 39(0), 3-23.

Lee, C., Percivall, G., 2008. Standards-based computing capabilities for distributed geospatial applications. Computer 41 (11), 50-57.

Lee, S.-S., and B. J. Sohn, 2012. Nighttime AOT retrieval for Asian dusts from MODIS IR measurements: An artificial neural network approach. Journal of the Meteorological Society of Japan 90, 163–177.

Lee, Y-W., Park, H.-H., Shibasaki, R., 2006. Collaborative GIS environment for exploratory spatial data analysis based on hybrid P2P network. Z. Pan et al. (Eds.) Edutainment 2006, LNCS 3942, 330-333.

Legrand, M., Plana-Fattori, A., and N'doumé, C., 2001. Satellite detection of dust using the IR imagery of Meteosat: 1. Infrared difference dust index. Journal of Geophysical Research 106(D16), 18251–18274.

Li, J., Zhang, P., Schmit, T. J., Schmetz, J., and Menzel, W. P., 2007. Technical note: Quantitative monitoring of a Saharan dust event with SEVIRI on Meteosat-8, International Journal of Remote Sensing 28(10), 2181–2186.

Li, Z., Yang, C., Huang, Q., Liu, K., Sun, M. and Xia, J., 2014. Building model as a service to support geosciences. Computers, Environment and Urban Systems (In press)

Li Z, Yang C, Jin B, Yu M, Liu K, Sun M, 2015. Enabling big geoscience data analytics with a cloud-based, MapReduce-enabled and service-oriented workflow framework. PLoS ONE 10(3), e0116781.

Liu, M., Wang, M. R., Shen, W. M., Luo, N., and Yan, J. W., 2012. A quality of service (QoS)-aware execution plan selection approach for a service composition process. Future Generation Computer Systems-the International Journal of Grid Computing and E-Science 28, 1080–1089.

Maamar Z, Moste´faoui SK, Yahyaoui H., 2005. Toward an agent-based and context-oriented approach for Web Services composition. IEEE Transactions on Knowledge & Data Engineering 17(5), 686–97.

Mahinthakumar, K., von Laszewski, G., Ranjithan, R., Brill, D., Uber, J., Harrison, K., Sreepathi, S., Zechman, E., 2006. An adaptive cyberinfrastructure for threat management in urban water distribution systems. In Proceedings of computational science-ICCS 2006. Part III (lecture notes in computer science, vol. 3993), May 2006, Reading, pp. 401–408.

Makropoulos, C., Safiolea, E., Efstratiades, A., Oikonomidou, E., Kaffes, V., Papathanasiou, C., Mimikou, M., 2009. Multi-Reservoir management with OpenMI, 11th International Conference on Environmental Science and Technology, Chania, Crete, pp. 788-795.

Malik, A. and Lakshman, P., 2010. Cassandra: a decentralized structured storage system. SIGOPS Operating System Review 44(2), 35-40.

Malucelli, A., Palzer, D., and Oliveira, E., 2006. Ontology-based services to help solving the heterogeneity problem in e-commerce negotiations. Electronic Commerce Research and Applications 5(1), 29–43.

Mandl, D., Sohlberg, R., Justice, C., Ungar, S., Ames, T., Frye, S., Chien, S., Tran, D., Cappelaere, P., Sullivan, D., and Ambrosia, V., 2008. A space-based sensor web for disaster management. In Geoscience and Remote Sensing Symposium. IGARSS 2008. IEEE International.

Matott, L.S., Babendreier, J.E., Purucker, S.T., 2009. Evaluating uncertainty in integrated environmental models: a review of concepts and tools. Water Resources Research 45, 1-14.

Mazzetti, P., Nativi, S., Caron, J., 2009. RESTful implementation of geospatial services for Earth and Space Science applications. International Journal of Digital Earth 2(1), 40-61.

McKendry, I.G., Hacker, J.P., Stull, R., 2001. Long-range transport of Asian dust to the Lower Fraser Valley, British Columbia, Canada. Journal of Geophysical Research 106(D16), 18361–18370.

Mell P, Grance T, 2011. The NIST definition of cloud computing (draft), NIST, http:// csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

McIlraith, S., and Son, T.C., 2002. Adapting Golog for composition of semantic Web Services. Proc 8th International Conference on Principles of Knowledge Representation and Reasoning, pp. 482–493.

Mlawer, E. J., Taubman, S. J., Brown, P. D., Iacono, M. J. and Clough, S. A., 1997. Radiative transfer for inhomogeneous atmospheres: RRTM, a validated correlated-k model for the longwave. Journal of Geophysical Research 102(D14), 16663-16682.

Moore, R.V., 2001. Description of work for the HarmonIT PROJECT agreed during contract negotiation, IT Frameworks (HarmonIT), Proposal Number: EVK1 2001-00037. www.HarmonIT.com.

Moore, R.V., Tindall, C.I., 2005. An overview of the open modelling interface and environment (the OpenMI). Environmental Science and Policy 8 (3), 279-286.

Muller, I., Kowalczyk, R., and Braun, P., 2006. Towards agent-based coalition formation for service composition. In Proceedings of of IEEE/WIC/ACM international conference on intelligent agent technology, Omaha, USA, pp. 73–80.

Muracevic, D., Kurtagic, H., 2009. Geospatial SOA using RESTful web services. In Proceedings of the 31st International Conference on Information Technology Interfaces, IEEE Piscataway, Dubrovnik, pp. 199–204.

Nanda, M.G., Chandra, S., Sarkar, V., 2004. Decentralizing execution of composite Web Services. In Proceedings of Conference on Object Oriented Programming Systems, Languages, and Applications, New York, USA, pp. 170-187.

Nativi, S., Caron, J., Davis, E., Domenico, B., 2005. Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-G(ML)). Computers & Geosciences 31, 1104–1118.

Natsagdorj, L., Jugder, D., Chung, Y.S., 2003. Analysis of dust storms observed in Mongolia during 1937-1999. Atmospheric Environment 37, 1401-1411.

NASA/ESTO. 2008. Report from the Earth Science Technology Office (ESTO) Advanced Information System Technology (AIST) Sensor Web technology meeting. Orlando, FL: NASA Earth Science Technology office.

Nativi, S., Caron, J., Davis, E. and Domenico, B., 2005. Design and implementation of netCDF markup language (NcML) and its GML-based extension (NcML-GML). Computers & Geosciences 31(9), 1104-1118

Nativi, S., Mazzetti, P., Geller, G.N., 2013. Environmental model access and interoperability: the GEO model web initiative. Environmental modelling & Software 39, 214-228.

Obukhov, A.M., 1971. Turbulence in an atmosphere with a non-uniform temperature. Boundary-Layer Meteorology 2, 7-29.

Olson, A.J., 2010. Data as a service: Are we in the clouds? Journal of Map & Geography Libraries 6 (1), 76-78.

Pantazoglou, M., Pogkas, I. and Tsalgatidou, A., 2014. Decentralized enactment of BPEL processes. IEEE Transactions on Services Computing 7(2), 184-197.

Parkera, D.C., S. M. Mansonb, M. A. Janssenc, M. J. Hoffmannd, and P. Deadmane, 2003. Multi-agent systems for the simulation of land-use and land-cover change: A review. Annals of the Association of American Geographers 93(2), 314–337.

Pavolonis, M.J., Feltz, W.F., Heidinger, A.K., Gallina, G.M., 2006. A daytime complement to the reverse absorption technique for improved automated detection of volcanic ash. Journal of Atmospheric and Oceanic Technology 23, 1422–1444.

Percivall, G. (Ed.), 2002. The OpenGIS abstract specification, topic 12: OpenGIS service architecture. Version 4.3. OGC 02-112. Open Geospatial Consortium, Inc., pp. 78.

Pezzoli, K., Marciano, R., Robertus, J., 2006. Regionalizing integrated watershed management: A strategic vision. In ACM international conference proceeding series – proceedings of the 7th annual international conference on digital government research, May 21–24, 2006, San Diego, CA, pp. 444–445.

Pham, T. V., Jamjoom, H., Jordan, K., and Shae, Z.-Y., 2010. A service composition framework for market-oriented high performance computing cloud. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, Chicago, Illinois: ACM. pp. 284–287.

Prata, A.J., 1989. Infrared radiative transfer calculations for volcanic ash clouds. Geophysical Research Letters 16, 1293–1296.

Puppin, D., Moncelli, S., R. Baraglia, N. Tonelotto, Silvestri, F., 2005. A Grid information service based on Peer-to-Peer. In Proceedings of 11th Euro-Par Conference, Euro-Par 2005, In LNCS, vol. 3648, Springer, 454–464.

Qu, J. J., Hao, X. J., Kafatos, M., and Wang, L. L., 2006. Asian dust storm monitoring combining Terra and Aqua MODIS SRB measurements, IEEE Geosci. Remote Sensing Letters 3(4), 484-486.

Rao, A. S. and Georgeff, M. P., 1991. Modeling rational agents within a BDI architecture. In Allen, J., Fikes, R., and Sandewall, E., editors, Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann. pp. 473-484.

Rich, P. M., Weintraub, L. H. Z., Ewers, M. E., Riggs, T. L., Wilson, C. J., 2005. Decision support for water planning: The ZeroNet water-energy initiative. In Proceedings of the 2005 World Water and Environmental Resources Congress, Anchorage, AK, pp. 468.

Rizvandi, N. B., Zomaya, A. Y., A. J. Boloori, and J. Taheri, 2011. On modeling dependency between MapReduce configuration parameters and total execution time, CoRR.

Roman, D., Schade, S., Berre, A. J., Bodsberg, N. R., and Langlois, J., 2009. Model as a service (MaaS). In AGILE workshop: Grid technologies for geospatial applications, Hannover, Germany,

Safi Esfahani, F., Azmi Murad, M. A.,Sulaiman, M. N. B. and Udzir, N. I., 2011. Adaptable decentralized service oriented architecture. Journal of Systems and Software 84(10), 1591-1617.

Sample J, Ioup E., 2010, Logical tile schemes. In Tile-based geospatial information systems. US: Springer. ISBN 978-1-4419-7630-7; 2010. pp. 5–15.

Schaeffer, B., 2008. Towards a transactional web processing service. In Proceedings of the Sixth Geographic Information Days, Münster.

Schlosser, M., Sintek, M., Decker, S., Nejdl, W., 2002. A scalable and ontology-based p2p infrastructure for semantic Web Services, in Proceedings of the Second International Conference on Peer-to-Peer Computing (P2P'02), Linkoping, Sweden, pp. 104–111.

Schut, P., 2007. OpenGIS Web Processing Service. Open Geospatial Consortium, pp. 1-87.

Searle, J., 1970. Speech Acts: An Essay in the Philosophy of language. Cambridge Univ. Press, pp. 142.

Sim K.M., 2009. Agent-based cloud commerce. In Proc IEEE international conference on industrial engineering and engineering management, Hong Kong, pp. 717–721

Sim K.M., 2011. Agent-based cloud computing. IEEE Transactions on Services Computing 5(4), 564 – 577.

Sim K.M., 2012. Complex and concurrent negotiations for multiple interrelated e-markets. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 43(1), 230-245.

Shao Y. and Dong C. H., 2006. A review on East Asian dust storm climate, modelling and monitoring. Global and Planetary Change 52, 1–22.

Sheth, A.P., 1999. Changing focus on interoperability in information systems: from system, syntax, structure to semantics. In Goodchild, M., Egenhofer, M., Fegeas, R., Kottman, C. (Eds.), Interoperating Geographic Information Systems. Kluwer Publisher, Norwell, pp. 5–30.

Shen, W., et al., 2007. An agent-based service-oriented integration architecture for collaborative intelligent manufacturing. Robotics and Computer Integrated Manufacturing 23, 315–325.

Shneiderman, B., 2007. Web science: a provocative invitation to computer science. Communication of ACM 50 (6), 25–27.

Sinnema, M., and Deelstra, S., 2007. Classifying variability modeling techniques. Information and Software Technology 49, 717–739.

Smith, R. G., 1981. Correction to the contract net protocol: high-level communication and control in a distributed problem solver. IEEE Transactions on Computers, 30, pp. 372.

Stonebraker, M., 2010. SQL databases vs NoSQL databases, Communications of the ACM 53 (4), 10–11.

Su, H., Houser, P.R., Tian, Y., Geiger, J.V., Kumar, S.V., Belvedere, D.R., 2008. A Land Information Sensor Web (LISW) study in support of land surface studies. International Geoscience & Remote Sensing Symposium (IGARSS), Boston, MA.

Sucaet, Y., Van Hemert, J., Tucker, B., and Bartholomay, L., 2008. A web-based relational database for monitoring and analyzing mosquito population dynamics. Journal of Medical Entomology 45(4), 775–784.

Sun, J., Zhang, M., and Liu, T., 2001. Spatial and temporal characteristics of dust storms in China and its surrounding regions, 1960 – 1999: Relations to sources area and climate. Journal of Geophysical Research 106(10), 325 – 10,333.

Tang, W., Wang, S., Bennett, D.A., and Liu, Y., 2011. Agent-based modeling within a cyberinfrastructure environment: A service-oriented computing approach. International Journal of Geographical Information Science 25(9), 1323–1346.

Talia D., 2011. Cloud computing and software agents: towards cloud intelligent services, in WOA, ser. CEUR Workshop Proceedings, G. Fortino, A. Garro, L. Palopoli, W. Russo, and G. Spezzano, Eds., vol. 741. CEUR-WS.org 2011, pp. 2–6.

Tan, S.C., Shi, G.Y., Wang, H., 2012. Long-range transport of spring dust storms in inner Mongolia and impact on the China seas. Atmospheric Environment 46, 299-308.

Tan, X.C., et al., 2015. Cloud- and agent-based geospatial service chain: A case study of submerged crops analysis during flooding of the Yangtze River basin. IEEE Journal on Selected Topics of Applied Remote Sensing 8(3), 1359-1370.

Thiebes, B., R. B. Bell, T. Glade, S. Jäger, M. D. Anderson, and L. Holcombe. 2013. A WebGIS decision-support system for slope stability based on limit-equilibrium modelling. Engineering Geology 158, 109–118.

Tong, H., et al., 2009. A distributed agent coalition algorithm for Web Service composition, in Proceedings of 2015 IEEE 10th World Congress on Services, pp. 62–69.

Tong, H., Cao, J., Zhang, S., Li, M., 2011. A distributed algorithm for Web Service composition based on service agent model. IEEE Transactions on Parallel and Distributed Systems 22(12), 2008–2021.

Torres, O., Tanskanen, A., Veihelmann, B., Ahn, C., R. Braak, Bhartia, P. K., Veefkind, P., and Levelt, P., 2007. Aerosols and surface UV products from Ozone Monitoring Instrument observations: An overview. Journal of Geophysical Research 112, 24-47.

Tsai, F., G. T.-J. Chen, T.-H. Liu, W.-D. Lin, and Tu J.-Y., 2008. Characterizing the transport pathways of Asian dust. Journal of Geophysical Research 113, D17311

Turuncoglu, U. U., Dalfes, N., Murphy, S. and DeLuca, C., 2013. Toward self-describing and workflow integrated Earth system models: a coupled atmosphere-ocean modeling system application. Environmental Modelling & Software 39, 247-262

Vaccari, L., Shvaiko, P., and Marchese M., 2009. A geo-service semantic integration in Spatial Data Infrastructures. International Journal of Spatial Data Infrastructures Research 4, 24–51.

Wald, A. E., Kaufman, Y. J., Tanré, D., and Gao, B. C., 1998. Daytime and nighttime detection of mineral dust over desert using infrared spectral contrast, Journal of Geophysical Research 103(D24), 32307–32313.

Wang, J.-Z., Wang, J.-J., Zhang, Z.-G., and Guo, S.-P., 2011. Forecasting stock indices with back propagation neural network. Expert Systems with Applications, 38,14346–14355.

Wang, Q. and Wang J., 2009. Intelligent web map service aggregation. In Proceedings of International Conference of Computing, Intell. Nat. Comput 2, 229–231.

Wang, S.Y., Shen, W.M., and Hao, Q., 2006. An agent-based Web Service workflow model for inter-enterprise collaboration. Expert Systems with Applications 31 (4), 787–799.

Wang, S., Anselin, L., Bhaduri, B., Crosby, C., Goodchild, M. F., Liu, Y., et al., 2013. CyberGIS software: A synthetic review and integration roadmap. International Journal of Geographical Information Science 27, 2122–2145.

Weiss G., 2013. Multiagent systems: a modern approach to distributed artificial intelligence, 2nd edition. MIT Press, Cambridge, pp. 197.

Wischik, D., Handley, M., and Braun, M. B., 2008. The resource pooling principle. SIGCOMM Computer Communication Review 38, 47–52.

Wittern, E., Kuhlenkamp, J., and Menzel, M., 2012. Cloud service selection based on variability modeling. LNCS 7636, 127–141.

Worm, D. et al., 2012. Revenue maximization with quality assurance for composite Web Services. In Service-oriented Computing and Applications (SOCA), 2012 5th IEEE International Conference. pp. 1–9.

Wooldridge, M., Jennings, N. R., and Kinny, D., 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems 3(3), 285-312.

Wu, Q., Zhang, M., Zheng, R., Lou, Y., and Wei, W., 2013. A QoS-satisfied prediction model for cloud-service composition based on a hidden Markov model. Mathematical Problems in Engineering, pp. 7.

Xie, J., et al., 2010. High performance computing for the simulation of dust storms. Computers, Environment, and Urban Systems 34 (4), 278-290.

Yakimenko, O. A., Slegers, N. J., Bourakov, E. A., Hewgley, C. W., Bordetsky, A. B., Jensen, R. P., Robinson, A. B., Malone, J. R., and Heidt, P. E., 2009. Mobile system for precise aero delivery with global reach network capability. In Control and Automation, 2009. ICCA 2009, IEEE International Conference, pp. 1394–1398.

Yang, C., Li, W., Xie, J., and Zhou, B., 2008. Distributed geospatial information processing: Sharing earth science information to support Digital Earth. International Journal of Digital Earth 1(3), 259–278.

Yang, C., and Raskin, R., 2009. Introduction to distributed geographic information processing research. International Journal of Geographic Information Science 23(5), 553–560.

Yang, C., Raskin, R., Goodchild, M., and Gahegan, M., 2010. Geospatial Cyberinfrastructure: Past, present and future. Computers, Environment and Urban Systems 34(4), 264-277.

Yang, C., M. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacus and D. Fay, 2011a. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing?, International Journal of Digital Earth 4(4), 305-329.

Yang C., et al., 2011b. Using spatial principles to optimize distributed computing for enabling physical science discoveries. Proceedings of National Academy of Sciences 106 (14), 5498-5503.

Yang C., Chen, N.C., and Di, L.P., 2012b. Restful based heterogeneous geoprocessing workflow interoperation for sensor Web Service. Computers & Geosciences 47, 102-110.

Yang, C., Xu, Y., and Nebert, D., 2013. Redefining the possibility of digital earth and geosciences with spatial cloud computing. International Journal of Digital Earth 6(4), 297-312.

Yao, Z., Li, J., Han, H.-J., Huang, A., Sohn, B. J., and Zhang, P., 2012. Asian dust height and infrared optical depth retrievals over land from hyperspectral longwave infrared radiances, Journal of Geophysical Research 117, D19202

Yu, G., Zhao, P., Di, L., Chen, A., Deng, M., Bai, Y., 2012. BPELPower – a BPEL execution engine for geospatial web services. Computer & Geosciences 47 (10), 87–101.

Yue, P., Gong, J., and Di, L., 2010. Augmenting geospatial data provenance through metadata tracking in geospatial service chaining. Computers & Geosciences 36(3), 270–281.

Zeng, C., Guo, X. A., Ou, W. J., and Han, D., 2009. Cloud computing service composition and search based on semantic. In M. G. Jaatun, G. Zhao, & C. Rong (Eds.). Cloud Computing, Proceedings. Berlin: Springer-Verlag Berlin, pp. 290–300.

Zhang, X.Y., Arimoto, R., An, Z.S., 1997. Dust emission from Chinese desert sources linked to variations in atmospheric circulation. Journal of Geophysical Research 102, 28041–28047.

Zhang, P., Lu, N.M., Hu, X.Q., Dong, C.H., 2006. Identification and physical retrieval of dust storm using three MODIS thermal IR channel. Global Planet. Change 52, 197–206.

Zhang, Q., Cheng, L., Boutaba, R., 2010. Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications 1(1), 7–18.

Zhang B., Di L.P., Yu G.N., Han W.G., Wang H.L., 2012 Towards data and sensor planning service for coupling earth science models and earth observations. IEEE Journal of selected topics in applied Earth Observations and Remote Sensing 5(6), 1939-1404,

Zhang, M., Ranjan, R., Nepal, S., Menzel, M., and Haller, A., 2012. A declarative recommender system for cloud infrastructure services selection. In Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services, Berlin, Germany: Springer-Verlag. pp. 102–113.

Zhang, X. M., Song, W., Liu, L. M., 2014. An implementation approach to store GIS spatial data on NoSQL database. In Geoinformatics, 22nd International Conference, pp. 25-27.

Zhao, T. L., Gong, S. L., Zhang, X. Y., Blanchet, J-P., McKendry, I. G., and Zhou, Z. J., 2006. A simulated climatology of Asian dust aerosol and its trans-pacific transport. Part I: Mean Climate and Validation. Journal of Climate 19, 88–103.

Zhao, T.L., Gong, S.L., Zhang, X.Y., Jaffe, D.A., 2008. Asian dust storm influence on North American ambient PM levels: observational evidence and controlling factors. Atmospheric Chemistry and Physics 8, 2717-2728.

Zhao, P., Foerster, T. and Yue, P., 2012a. The geoprocessing web. Computers & Geosciences 47, 3-12.

Zhao, P., Di, L. and Yu, G., 2012b. Building asynchronous geospatial processing workflows with Web Services. Computers & Geosciences 39(0), 34-41.

Zhao, X., Wen, Z., and Li, X., 2013. QoS-aware Web Service selection with negative selection algorithm. Knowledge and Information Systems, pp. 1–25.

Zhou, S., Balaji, V., Cruz, C., da Silva, A., Hill, C., Kluzek, E., Smithline, S., Trayanov, A., and Yang, W., 2007. Cross-organization interoperability experiments of weather and climate models with the Earth System Modeling Framework: Research Articles. Concurrency and Computation: Practice and Experience 19(5), 583-592.

Zhou, X., and Mao, F., 2012. A semantics web service composition approach based on cloud computing. In Computational and Information Sciences (ICCIS), 2012 Fourth International Conference, pp. 807–810.

Zhou, Z.J., Zhang, G.C., 2003. Typical severe dust storms in northern China during 1954-2002. Chinese Science Bulletin 48 (21), 2366-2370.

Zhu, Y., Li, W., Luo, J., & Zheng, X. (2012). A novel two-phase approach for QoS-aware service composition based on history records. In 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–8.