pipsCloud: High performance cloud computing for remote sensing big data management and processing

Lizhe Wang, Yan Ma, Jining Yan, Victor Chang, Albert Y. Zomaya

PII: S0167-739X(16)30192-3
DOI: http://dx.doi.org/10.1016/j.future.2016.06.009
Reference: FUTURE 3075

To appear in: *Future Generation Computer Systems*

Received date: 29 February 2016
Revised date: 25 May 2016
Accepted date: 12 June 2016

pipsCloud: High Performance Cloud Computing for Remote Sensing Big Data Management and Processing

Lizhe Wang[1,2], Yan Ma[2,†], Jining Yan[2] and Victor Chang[3] and Albert Y. Zomaya [4]

1 School of Computer Science, China University of Geoscience, Wuhan 430074, P. R. China
2 Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, P. R. China
3 School of Computing, Creative Technologies and Engineering, Leeds Beckett University, United Kingdom
4 School of Information Technologies, University of Sydney, Australia
† corresponding author: mayan@radi.ac.cn

Abstract

With the increasing requirement of accurate and up-to-date resource & environmental information for regional and global monitoring, large-region covered multi-temporal, multi-spectral massive remote sensing (RS) datasets are exploited for processing. The remote sensing data processing generally follows a complex multi-stage processing chain, which consists of several independent processing steps subject to types of RS applications. In general the RS data processing for regional environmental and disaster monitoring are recognized as typical both compute-intensive and data-intensive applications.

To solve the aforementioned issues efficiently, we propose pipsCloud which combine recent Cloud computing and HPC techniques to enable large-scale RS data processing system as on-demand real-time services. Benefiting from the ubiquity, elasticity and high-level of transparency of Cloud computing model, the massive RS data managing and data processing for dynamic environmental monitoring are all encapsulate as Cloud with Web interfaces. Where, a Hilbert-R^+ based data indexing mechanism is employed for optimal query and access of RS imageries, RS data products as well as interim data. In the core platform beneath the Cloud services, we provide a parallel file system for massive high-dimensional RS data and offers interfaces for intensive irregular RS data accessing so as to provide improved data locality and optimized I/O performance. Moreover, we adopt an adaptive RS data analysis workflow manage system for on-demand workflow construction and collaborative execution of distributed complex chain of RS data processing, such as forest fire detection, mineral resources and coastline monitoring. Through the experimental analysis we have show the efficiency of the pipsCloud platform.

Keywords: High Performance Computing, Cloud computing, data-intensive computing, Big Data, Remote Sensing

1. Introduction

With the remarkable advances in high-resolution Earth Observation (EO), we are witnessing an explosive growth in the volume and also velocity of Remote Sensing (RS) data ([1]). The latest-generation space-borne sensors are capable of generating continuous streams of observation data at a growing rate of several gigabytes per second ([2]) almost every hour, every day, every year. The global archived observation data are probably exceed one Exabyte according to the statistics of OGC report ([3]). The volume of RS data acquired by a regular satellite data center is dramatically increasing by several terabytes per day, especially for the high-resolution missions ([4]). While, the high-resolution satellites, namely indicates higher spatial, higher spectral and higher temporal resolution of data, which would inevitably give rise to the higher dimensionality nature of pixels. Coupled with the diversity in the present and upcoming sensors, RS data are commonly regarded as *"Big RS Data"* or *"Big Earth Observation Data"*,not merely in the data volume, but also in terms of the complexity of data.

The proliferation of "RS Big Data" is revolutionizing the way RS data are processed, analyzed and interpreted as knowledge ([5]). In large-scale RS applications,regional or even global covered multi-spectral and multi-temporal RS datasets are exploited for processing, so as to meet the arising demands for more accurate and up-to-date information. A continent-scale forest mapping normally involves processing terabytes of multi-dimensional RS datasets for available forest information ([6]). Moreover, large-scale applications are also exploiting multi-source RS datasets for processing so as to compensate for the limitation of a single sensor. Accordingly, not only the significant data volume, but the increasing complexity of data has also become the vital issue. Particularly, many time-critical RS applications even demand real-time or near real-time processing capacities ([7][8]). Some relevant examples are large debris flow investigation ([9], flood hazard management ([10]) and large ocean oil spills surveillance ([11][12]). Generally, these large-scale data processing problems in RS applications ([4][13][14]) with high QoS requirement are typically regarded as both compute-intensive and data-intensive. Likewise, the innovative analyses and high QoS (Quality of Service) requirements are driving the renewal of traditional RS data processing systems. The timely processing of tremendous multi-dimensional RS data has introduced unprecedented computational requirements, which is far beyond the capability that conventional instruments could satisfy. Employing cluster-based HPC (High-Performance Computing) paradigm in RS applications turn out to be the most widespread yet effective approach ([15][16][17][18][19]). Both NASA's NEX system ([5] for global processing and InforTerra's "Pixel Factory" ([20]) for massive imagery auto-processing adopt cluster-based platforms for QoS optimization.

However, despite the enormous computational capacities, cluster platforms that not data-intensive optimized are still challenged with huge data analysis and intensive data I/O. The mainstream multi-core clusters are characterized with multi-level hierarchy and increasing scale. These HPC systems are almost out of reach for non-experts of HPC, since the easy programming on MPI-enabled (Message Massing Interface) HPC platforms is anythings but easy. Moreover, the prevalent on-line processing needs are seldom met. In that, there lacks easy-to-use way to serve end users the massive RS data processing capabilities in large-scale HPC environment ubiquitously. Whereas, diverse RS data processing typically follows a multi-stage on-the-flow processing. The on-demand processing also means the ability to customize and serve dynamic processing workflows, instead of the predefined static ones. While, on-demand provision of resources will result in unpredictable and volatile requirements of large-scale computing resources. A substantial investment for sustaining system upgrade and scale-out would be essential. In addition, the build and maintenance of such platforms is remarkably complicated and expensive.

Cloud computing ([21];) provides scientists with a revolutionary paradigm of utilizing computing infrastructure and applications. By virtue of virtualization, the computing resources and various algorithms could be accommodated and delivered as ubiquitous services on-demand according to the application requirements. Cloud paradigm has also been widely adopt in large-scale RS applications, such as Matsu project ([22]) for cloud-based flood assessment. Currently, Clouds are rapidly joining HPC systems like clusters as variable scientific platforms ([23]). Scientists could easily customize their HPC environment and access huge computing infrastructures in Cloud. However, compared to conventional HPC systems or even supercomputers, the Clouds are not QoS-optimized large-scale platforms. Moreover, differs from traditional Cloud, these Datacenter Clouds deployed with data-intensive RS applications should facilitate massive RS data processing and intensive data I/O.

To efficiently address the aforementioned issues, we propose *pipsCloud* a cloud-enabled High-Performance RS data processing system for large-scale RS applications. The main contribution of it is that, it incorporates Cloud computing paradigm with cluster-based HPC systems in the attempting to address the issues from a system architecture point of view. Firstly, by adopting application-aware data layout optimized data management and Hilbert R^+ tree based data indexing, the RS big data including imageries, interim data and products could be efficiently managed and accessed by users. By means of virtualization and bare-metal (BM) provisioning ([24]), not only virtual machines, but also bare-metal machines with less performance penalty are deployed on-demand for easy scale up and out. Moreover, the generic parallel programing skeletons are also employed for easy programming of efficient MPI-enabled RS applications. Following this way, the cloud-enabled virtual HPC environment for RS big data processing are also dynamically encapsulated and delivered as on-line services. Meanwhile, benefiting from dynamic scientific workflow technique, *pipsCloud* offers the ability to easily customize collaborative processing workflows for large-scale RS applications.

The rest of this paper is organized as follows. The Section 2 reviews some related works, and section 3 discuss the challenges lie in the building and enabling a high performance cloud system for data-intensive RS data processing. Section 4 demonstrates the design and implementation of the *pipsCloud* from a point view of system level. Then section 5 discusses the experimental validation and analysis of the *pipsCloud*.Finally Section 6 concludes this paper.

2. High Performance Computing for RS Big Data: State of the Art

Each solution has its cons and pros. In this section, we comparatively review current dominant system architectures regularly adopted in the context of RS data processing, both cluster-based HPC platforms and Clouds. Firstly,in section 2.1, we go deep into the incorporation of multi-core cluster HPC structure with RS data processing systems and applications. Then, in section 2.2, we introduce some new attempt to enable large-scale RS applications by taking advantages of Cloud computing paradigm.

2.1. Cluster Computing for RS Data Processing

As increasing amount of improved sensor instruments are incorporated with satellites for Earth Observation, we have been encountering an era of "RS Big Data". Meanwhile, the urgent demands for large-scale remote sensing problems with boosted computation requirements ([5]) have also fostered the widespread applying of multi-core clusters. The first shot goes to the NEX system ([5]) for global RS applications built by NASA on a cluster platform with 16 computer in the middle of 1990s. "Pixel Factory" system ([20]) of InforTerra have employed cluster-based HPC platform for massive RS data auto-processing, especially Ortho-rectification. These HPC platforms are also employed in the acceleration of hyperspcetral imagery analysis ([25]). It is worth noting that 10,240-CPU Columbia supercomputer[1] equipped with InfiniBand network have been exploited for remote sensing applications by NASA.

Several traditional parallel paradigms are commonly accepted for these multi-level hierarchy featured cluster systems. OpenMPparadigm is designed for shared-memory, MPIis adopt within or across nodes, and MPI+OpenMP hybrid

[1]Columbia Supercomputer at NASA Ames Research Center, http://www.nas.nasa.gov/Resources/Systems/columbia.html

paradigm ([26]) is employed for exploiting multilevel of parallelism. Recently, great efforts have been laid on the incorporation of MPI-enabled paradigm with remote sensing data processing in the large scale scenarios. Some related works go with Plaza et al. presented parallel processing algorithms for hyperspectral imageries ([27]), Zhao et al. ([28]) implemented soil moisture estimation in parallel on PC cluster, as well as MPI-enabled implementing of image mosaicking ([29], fusion ([30]) and band registration ([31]). Obviously, benefiting from the efforts and developments conducted in HPC platforms, plenty of RS applications have enhanced their computational performance in a significant way ([5]).

However, in spite of the elegant performance acceleration has achieved, it is still anything but easy for non-experts to employ cluster-based HPC paradigm. Firstly, the programming, deploying as well as implementing of parallel RS algorithms on an MPI-enabled cluster is rather difficult and error-prone ([32]). Secondly, HPC systems are not optimized for data-intensive computing especially. The loading, managing and communication of massive multi-dimensional RS data on the distributed multilevel memory hierarchy of HPC system would be rather challenging. Some emerging PGAS ([33]) typed approaches offer global but partitioned memory address spaces across nodes, like UPC ([34], Chapel ([35]) and X10 ([36]). The on-going DASH project[2] is developing Hierarchical Arrays (HA) for hierarchical locality. Thirdly, the relatively limited resources in HPC systems could not be easily scaled to meet the on-demand resource needs of diverse RS applications. For affordable large-scale computing resources, substantial upfront investment and sustaining scaling up would be inevitable but also rather expensive. In addition, cluster-base HPC systems lack easy and convenient way of utilizing high performance data processing resources and applications, even not mention the on-demand customizing of computing resources and processing workflows.

2.2. *Cloud Computing for Remote Sensing Data Processing*

Cloud has emerged as a promising new approach for ad-hoc parallel processing, in the Big Data era[37]. It is capable of accommodating variable large-scale platforms for different research disciplines with elastic system scaling. Benefiting from virtualization, not only computing resources, but also software could be dynamically provisioned as ubiquitous services best suited to the needs of the given applications. Compared to the MPI-enabled cluster systems, cloud paradigm provides computing resources in a more easy-to-use and convenient way – a service-oriented way.

The advent of Cloud has also empowered remote sensing and relevant applications. Matsu ([22], the on-going research project of NASA for on-demand flood prediction and assessment with RS data adopts an Eucalyptus-based[38]) distributed cloud infrastructure with over 300cores. GENESI-DEC[3], a project of Ground European Network for Earth Science Interoperations – Digital Earth Communities. It employs a large and distributed cloud infrastructure to allow worldwide data access, produce and share services seamlessly. With virtual organization approach, the Digital Earth Communities could lay their joint effort for addressing global challenges, such as biodiversity, climate change and pollution.The ESA (European Space Agency) G-POD[4], a project to offer on-demand processing for Earth observation data was initially constructed with GRID. Subsequently, Terraducloud infrastructure was selected to enhance G-POD for resources provisioning ([5]).

Great efforts have been laid on the employing of cloud computing in the context of remote sensing data processing, both in terms of programming models and resource provisioning.

2.2.1. *Programming Models for Big Data*

Several optional distributed programming models[37] are prevalently employed for processing large data sets in the cloud environment, like MapReduce ([39]) and Dryad[5]. Where, MapReduce is the most widely accepted model for distributed computing in Cloud environment. By using “Map” and “Reduce” operations, some applications could be easily implemented in parallel without concerning data splitting and any other system related details. With the growing interest towards Cloud computing, it has been greatly employed in RS data processing scenarios. Lin et al. ([40]) proposed service integration model for GIS implemented with MapReduce, B. Li et al. ([41]) employed MapReduce for parallel ISODATA clustering. Based on Hadoop MapReduce framework, Almeer ([42]) built an experimental 112-core high-performance cloud system at University of Qatar for parallel RS data analysis.

Despite of the simple but elegant programming feature, MapReduce is not a fits-all model. In the context of large-scale RS applications, we have data with higher dimensionality, algorithms with higher complexity as well as specific dependences between computation and data. In these scenarios, the simple data partition of MapReduce with no idea of actual applications would not always work for acceleration, even not mention the applications with numerical computing issues.

2.2.2. *Resource Management and Provisioning*

Essentially, on-demand resource managing and provisioning is foremost in cloud computing environment. Several choices of open-source cloud solutions are available to accommodate computing infrastructure as a service for viable computing. Among several solutions, such as OpenStack ([43], OpenCloud[6], Eucalyptus ([38]) and OpenNebula ([44], OpenStack is the most widely accepted and promising one. Basically, in recent Clouds, on-demand resource allocation and flexible management are built on the basis of virtualization. Many available choices of hypervisors for Server virtualization in current open cloud platforms are Xen hypervisor[45], Kernel-based Virtual Machine (KVM) ([46]) as well as VMWare ([47]). By management of poos of Virtual Machines(VMs), hypervisors could easily scale up and down to provide large-scale platform with great

[2]DASH project: under DFG programme “Software for Exascale Computing – SPPEXA”, http://www.dash-project.org

[3]GENESI-DEC, http://www.genesi-dec.eu

[4]Grid Processing on Demand, https://gpod.eo.esa.int

[5]Dryad, http://research.microsoft.com/en-us/projects/dryad/

[6]OpenCloud, https://www.opencloud.com

number of VMs. Likewise, the network virtualization concept has also emerged. Yi-Man ([48]) proposed Virt-IB for InfiniBand virtualization on KVM for higher bandwidth and lower latency.

The virtualization approaches normally deploy multiple VMs instances on a single physical machine (PM) for better resource utilization. However, virtualization and hypervisor middleware would inevitably introduce extra performance penalty. Recently, Varrette et. al. ([49]) has demonstrated the substantial performance impact and even a poor power efficiency when facing HPC-type applications, especially large-scale RS applications. As a result, whether the VMs in cloud suit as a desirable HPC environment is still unclear.

For performance sake, native hypervisors (compared to hosted hypervisors) that capable of bare-metal provisioning is another option to extend current cloud platforms. This kind of hypervisors implement directly on the hardware of hosts and take control of them. There are several bare-metal provisioning hypervisors, examples are xCAT[7], Perceus[8]). Xie ([24]) has extended OpenStack to support bare-metal provisioning using xCAT, so as to serve both VMs and bare-metal machines in cloud platform.

3. Requirements and Challenges: RSCloud for RS Big Data

Cloud computing paradigm has empowered RS data processing and makes it possible than ever ([50]). Unlike conventional ways of processing that done by standalone server or software, the cloud-based RS data processing is enabled with revolutionary promise of unlimited computing resources.

Despite the advantages we could explore in cloud computing paradigm, there remains some obstacles to cloud adoption in remote sensing. As we look into the future needs of large-scale RS exploration and applications, it is clear that the distributed "data deluge" and computing scales will continue to explode and even drive the evolution of processing platforms. Naturally, the tremendous RS data and RS algorithms may actually distributed across multiple data centers located in different geographic places crossing organizational or even national boundaries. In this scenario, the efficient storing, managing as well as sharing and accessing of these distributed RS data at such an extreme volume and data complexity would be anything but easy. Likewise, large-scale RS applications and environmental researches might always in demand for collaborative workflow processing across data centers in a distributed environment by network. To allow pervasive and convenient service for domain users on-line in such a heterogeneous distributed environment, one-stop service oriented user interface could be of vita importance. Moreover, different types of domain users as well as varieties of RS applications with different processing scales would normally give rise to diverse requirements of cloud service model. The examples of these service models are RS data subscription and virtual data storage service, virtual cluster-base HPC platform service or even virtual processing system service for RS.

To meet the above challenging demands, a could-enabled HPC framework especially designed for remote sensing era namely RSCloud is quite critical. Firstly, it should be capable of managing and sharing massive distributed RS data. According, RS big data across data centers can be easily accessed and on-demand subscribed as virtualized data catalog or storage for global sharing. Secondly, it should also be able to offer RS-specific HPC environment with not only elastic computing and storage capacities as well as RS-oriented programming and runtime environment so as to meet the on-demand need of applications. Thirdly, the dynamic workflow processing is also essential for addressing global RS problems which require the collaboration of multiple scientists or organizations. In addition, one-stop service of cloud portal is also important, where a wide variety of resources, and even on-line RS data processing could be easily accommodated through one-stop accessing. Last but not least, different service models of multi-tenant environment, where different types of virtual HPC environment and RS processing systems can be abstracted and served on-demand. Loosely speaking, RSCloud is a cloud-assisted platform for RS, which not only facilitate elastic provisioning of computing and storage resources, but also allow multi-tenant users to on-demand access, share and collaborative process of massive distributed RS data in different service models. .

4. *pipsCloud*: High Performance Remote Sensing Clouds

To properly address the above issues, we propose *pipsCloud*, a high-performance RS data processing system for large-scale RS applications in cloud platform. It provides a more efficient and easy-to-use approach to serve high-performance RS data processing capacity on-demand, and also QoS optimization for the data-intensive issues.

4.1. The System Architecture of pipsCloud

As illustrated in figure 1, *pipsCloud* adopts a multi-level system architecture. From bottom to up is respectively physical resources, cloud framework, VE-RS, VS-RS, data management and cloud portal. Wherein, the cloud framework manages physical resources to offer Iaas (Infrastructure as a Service) by virtue of OpenStack. Based on cloud framework, the VE-RS offers virtual HPC cluster environment as a service and VS-RS provides cloud-enabled virtual RS big data processing system for on-line large-scale RS data processing. While the management, indexing and sharing of RS big data are also served as Daas (Data as a service).

Cloud Framework employs the most popular but successful open source project OpenStack to form the basic cloud architecture. However, OpenStack mostly only offers virtual machines (VMs) through virtualization technologies. These VMs are run and managed by hypervisors, such as KVM or Xen. Despite of the excellent scalability, the performance penalty of virtualization is inevitable. To support the HPC cluster environment

[7]xCAT (Extreme Cloud Administration Toolkit), http://en.wikipedia.org/wiki/XCAT

[8]perceus, http://www.perceus.org

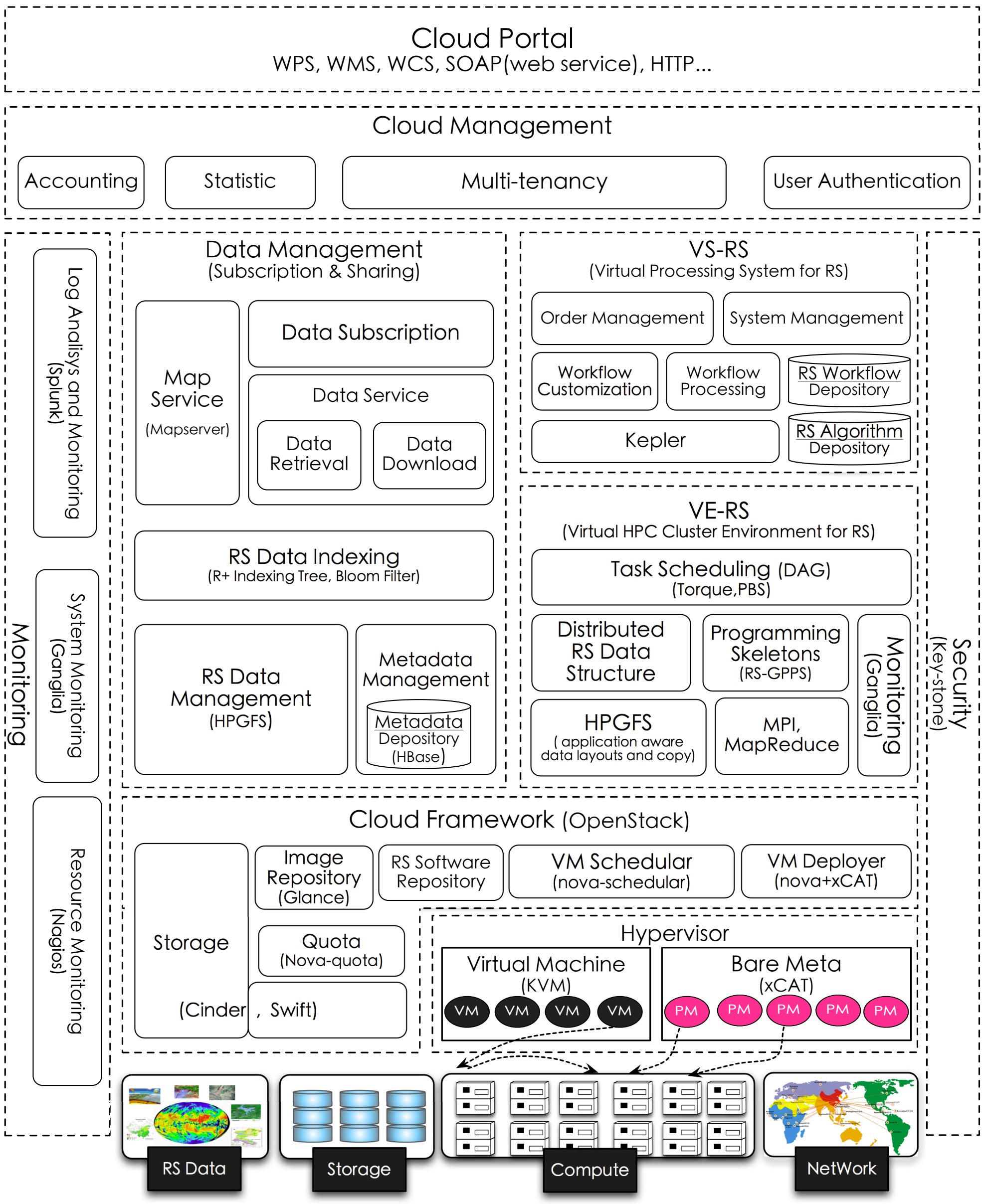

Cloud Portal
WPS, WMS, WCS, SOAP(web service), HTTP...
Cloud Management
Accounting
Statistic
Multi-tenancy
User Authentication
Monitoring
Log Analisys and Monitoring (Splunk)
System Monitoring (Ganglia)
Resource Monitoring (Nagios)
Data Management (Subscription & Sharing)
Map Service (Mapserver)
Data Subscription
Data Service
Data Retrieval
Data Download
RS Data Indexing (R+ Indexing Tree, Bloom Filter)
RS Data Management (HPGFS)
Metadata Management
Metadata Depository (HBase)
VS-RS (Virtual Processing System for RS)
Order Management
System Management
Workflow Customization
Workflow Processing
RS Workflow Depository
Kepler
RS Algorithm Depository
VE-RS (Virtual HPC Cluster Environment for RS)
Task Scheduling (DAG) (Torque,PBS)
Distributed RS Data Structure
Programming Skeletons (RS-GPPS)
Monitoring (Ganglia)
HPGFS (application aware data layouts and copy)
MPI, MapReduce
Security (Key-stone)
Cloud Framework (OpenStack)
Storage
(Cinder , Swift)
Image Repository (Glance)
RS Software Repository
VM Schedular (nova-schedular)
VM Deployer (nova+xCAT)
Quota (Nova-quota)
Hypervisor
Virtual Machine (KVM)
VM
Bare Meta (xCAT)
PM
RS Data
Storage
Compute
NetWork

Figure 1: The System Architecture of *pipsCloud*

in cloud, pipsCloud adopts a bare-metal machine provisioning approach which extends OpenStack with a bare-metal hypervisor named xCAT. Following this way, both VMs and bare-metal machines could be scheduled by nova-scheduler and accommodated to users subject to application needs.

VE-RS, namely a RS-specific cluster environment with data-intensive optimization, which also provided as a VE-RS service based on the OpenStack enabled cloud framework. By means of auto-configuration tools like AppScale, VE-RS could build virtual HPC cluster with Torque task scheduler and Ganglia for monitoring on top of cloud framework . Then varieties of RS softwares could be customized and automatically deployed on this virtual cluster with SlatStack[9]. Moreover, a generic parallel skeletons together with a distributed RS data structure with fine-designed data layout control are offered for easy but productive programming of large-scale RS applications on MPI-enabled cluster.

VS-RS, a virtual processing system that built on top of VE-RS are served as a on-line service especially for large-scale RS data processing.A VS-RS not only provides RS data products processing service, but also offers VS-RS processing system as a service and provide processing workflow customization. By virtue of Kepler scientific workflow engine, VS-RS could offer dynamic workflow processing and also on-demand workflow customization. The thing that worth noting is that enabled by Kelper, the complex workflow could also be built among clouds or different data centers with web services. Moreover, the RS algorithm depository together with RS workflow depository are

[9]saltstack: https://docs.saltstack.com/en/latest/

employed in VS-RS for workflow customization, interpreting and implementing. Besides, the order management as well as system monitoring and management are also equipped in VS-RS to enable on-line processing and management.

RS Data Management, a novel and efficient way of managing and sharing RS big data on top of cloud framework. It adopts unbounded cloud storage enabled by swift to store these varieties of data and served them in a RS data as a service manner. Wherein, HPGFS the distributed file system for RS data object is used for managing enormous unstructured RS data, like RS imageries, RS data products and interim data. While the structured RS metadata are managed by a NoSQL database – HBase[51]. For a quick retrieval from varieties of massive RS data, a Hilbert R^+ tree together with in-memory hot data cache policy are used for indexing acceleration. Last but not least, the thesis-based data subscription with virtual data catalog and thesis-based data push are also put forward as a novel way of data sharing.

Cloud Management and Portal manages the whole *pipsCloud* platform, including system monitoring, user Authentication, Multi-tenancy management as well as statics and accounting. While in the web portal of *pipsCloud*, the RS data management, RS data processing capabilities as well as on-demand RS workflow processing are all encapsulated as OGS web services interface standards, such as WPS (Web Proccessing Service), WCS (Web Coverage Service) and WMS (Web Map Service).

4.2. RS Data Management and Sharing

Efficient RS data management and sharing is paramount especially in the contest of large-scale RS data processing. The managing of RS big data not only limits to unstructured multi-source RS imageries, but also varieties of RS data products, interim data generated during processing, as well as structured metadata. As is mentioned above, HPGFS with application-aware data layout optimization is adopted for managing unstructured RS data, while the RS metadata are stored in HBase for query. Wherein, these unstructured RS data are organized in GeoSOT global subdivision grid, and each data block inside these data are arranged in Hilbert order and encoded with a synthetic GeoSOT-Hilbert code combing GeoSOT code with Hilbert value. The GeoSOT-Hilbert together with the info of data blocks are stored in the column family of the metadata in HBase for indexing. For a quick retrieval from varieties of massive RS data, a Hilbert R^+ tree with GeoSOT[10] [52] global subdivision is employed for indexing optimization. For further indexing acceleration, a hot-data cache policy is adopted to cache ''hot" RS imageries and metadata will cached into Redis[53][54] in-memory database and offer hashe table indexing. The most important thing that worth mention is the easy but novel way of RS data sharing – thesis-based RS data subscription and data push through virtual data catalog mounting as local .

[10]GeoSOT: Geographical Coordinate Subdividing Grid with One Dimension Integer Coding Tree

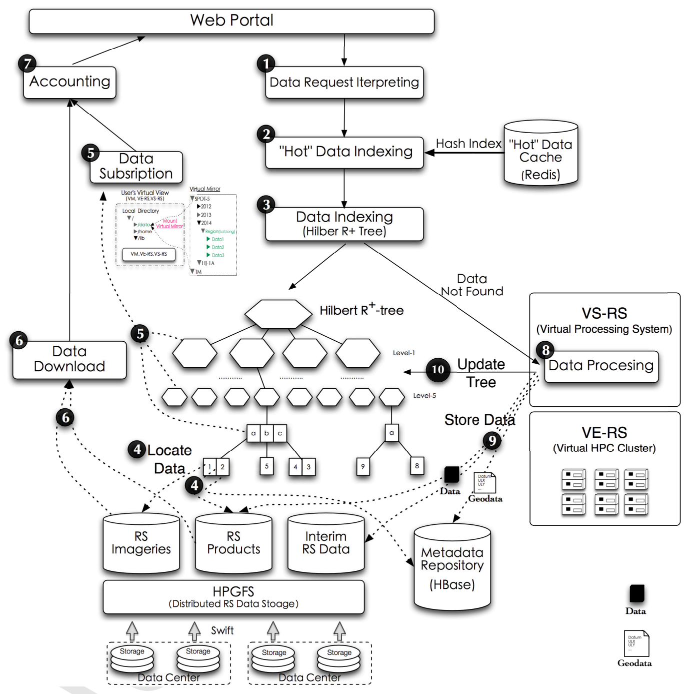


Figure 2: The Runtime Implementing of Data Management and Sharing

The runtime implementing of RS data management and sharing is demonstrated as figure 2. Firstly, pipsCloud interprets the data requests, and check the user authentication. Secondly, it will conducts a quick search in the ''hot" data cache on Redis in-memory database with hash table index, if cache hit then return data. Thirdly, search the required data in the Hilbert R^+ tree for the unique Hilbert-GeoSOT code. Fourthly, if used the Hilbert-GeoSOT code of the found data to locate the metadata entry in HBase, or locate the physical URL of the data for accessing. Fifthly, for a subscription request, re-organize these data entries to form a virtual mirror and mount it to users local mount point. Sixthly, if a acquisition of data or metadata is needed, then invoke GridFTP for downloading. Finally, Accounting is used for charging if the data is not free.

However, when the request data products are not available then a RS data product processing could be requested to VS-RS. The interim RS data and final RS data products generated during processing would be stored to interim and products repository based on HPGFS, while the relevant metadata will be abstracted and insert into metadata repository based on HBase. Meanwhile, the Hilbert R^+ tree should also be updated for further indexing, and the access RS data or metadata entry would automatically cache into Redis as "hot" data.

4.2.1. HPGFS: Distributed RS Data Storage with Application-Aware Data Layouts and Copies

The intensive irregular RS data access patterns of RS applications which always perform non-contiguous I/O across bunches of image data files would inevitably result in low I/O efficiency. While, HPGFS which extends the prototype of OrangeFS offers a an efficient and easy of use solution from server side to natively support the direct distributed storing and concurrent accessing of massive RS data with different irregular I/O patterns. With the interfaces for data layout policy customization, distributed metadata management, OrangeFS is highly configurable. In HPGFS, a logical distributed RS data object model is adopted to organize the RS image datasets with complex data structure. It is worth mention that HPGFS offers I/O interfaces with RS data operation semantics together with application-aware data layout and copy policies associated with expected data access patterns of applications.

HPGFS adopts a logical distributed RS data object model to describe the multi-band RS image, geographical metadata as well as relevant basic RS data operations. Where, the light-weighted Berkeley DB in distributed metadata servers is adopted for storing, managing and retrieval of the complex structured geographical metadata in a key-value fashion. While the multi-dimensional (normally 3-D) images are sliced and mapped into a 1-D data array using multiple space-filling curves simultaneously. Then the arrange data array scattered over a number of I/O servers with application-aware data layout policies. Withal, the basic RS data operation includes some metadata inquiry interfaces and geographical operations like projection reform and resampling.

As is depicted in figure 3, application-aware data layouts and data copy polices consistent with expected RS data access patterns are adopted for optimal data layout and exploiting data locality. It is worth noting that multiple redundant data copies with different application-aware data layouts are all simultaneously pre-created for each individual RS data. Instead of data striping method with fixed or variable stripe size, RS data are sliced into data bricks, which is are also multi-dimensional non-contiguous data. By awareness of the expected I/O patterns of RS applications, the 3-D data bricks in each copy of data are mapping and organized using a Space-filling Curve that best fits some certain data access pattern. Wherein, data copy organized in Z-order curve is provided for consecutive-lines/column access pattern, diagonal curve is for diagonal irregular data access pattern, while Hilbert curve is used for rectangular-block access pattern. With the knowledge of the I/O patterns, the requested RS data region distributed across different datasets or even data centers can be organized and accessed locally in one single logical I/O.

As is showed in figure 3, the hot data bricks would be dynamically copied and scheduled across I/O nodes adhere to the statistics of actual data accessing. During the idle time of the I/O nodes, the data bricks together with the list of the target I/O nodes would be packaged as a "brick copy task". Then, the data brick in the copy task would be copied and transfered to the target I/O nodes using a tree-based copying mechanism as in figure 3 to form dynamical copies of data bricks.

4.2.2. *RS Metadata Management with NoSQL Database*

Metadata management and indexing is also an importance part of RS data management and service. Recently, most of the cloud framework adopts a key-value NoSQL database based on distributed file system for the storage and random, real-time retrieval of massive structured data [55]. Therefore, NoSQL approach is employed in the management of enormous RS metadata, which is optimized with bloom filter accelerated indexing. By virtue of HBase database, the metadata of the RS data like data types, data products, geographical metadata are organized and stored in the column family. Wherein, the thumbnails of RS data for quick view is also stored in HBase in forms of several compressed tiles that sliced with fixed size. While, The actual data of the HBase database are stored in HDFS distributed file system in cloud. Following this approach, the geographically distributed metadata along with unstructured figures could also be managed by HBase for on-line metadata retrieval and quick view.

With the proliferation of data, the amount of metadata and catalogs would be bound to sour up. Not surprisingly, there would be millions of data entries in the metadata table of HBase. Actually, these enormous data entries in key-value fashion are normally serialized into thousands of files. Therefore, the on-line retrieval of metadata at extreme volume would definitely be a disaster ever. Actually, most of the key-value database are indexed by keyword of the data entry. There are several common indexing data structures, include Hash-based indexing, R-tree indexing, B-tree indexing and so on. Wherein, the indexing trees are normally used for local indexing inside a key-value storage, while the Hash mechanism is used for locating the data nodes. However, in this scenario, a global indexing tree would not be a wise chose, since the cost of building and maintenance it could be unprecedented huge.

In the distributed scenario of *pipsCloud*, a hybrid solution of combing both R-tree indexing and bloom filter is adopted instead of a global data indexing . Actually, we build a local R-tree indexing only for the metadata entries serialized in a group of data files or located in a single data center, instead of a global one. Meanwhile, bloom filter is a space-efficient probabilistic data structure that employed for global Hash-indexing, examples are Google BigTable and Apache Cassandra. Here, it is used to test whether the requested data is belonging to specified group of files or located in a certain data center. When in case of metadata retrieval, the first step is to determine which group (m groups) of entries it belongs to by conducting multiple bloom filter indexing of each group concurrently in parallel. Then follows a local searching along a R-tree of the selected group of metadata entries. Eventually, the faster decision of bloom filter as well as the k independent hash lookups in parallel, would give rise to an accelerated metadata query through HBase and reduced costs of global indexing.

4.2.3. *RS Data Index with Hilbert R^+tree*

Quick and efficient data retrieval among enormous distributed RS data has alway been a considerable challenging issue. Historically, indexing data structures like R-tree or B-tree are normally used to improve the speed of global RS data retrieval and sharing. Actually, the performance of indexing tree greatly depends on the algorithms used to clustering the minimum bounding rectangles (MBRs) of the RS data on a node. Hilbert R^+*tree* employ a Hilbert space-filling curve to arrange the data rectangles in a linear order and also group the neighbor rectangles together. To meet the requirements of real-time RS data updating, a dynamic Hilbert R^+*tree* is normally more de-

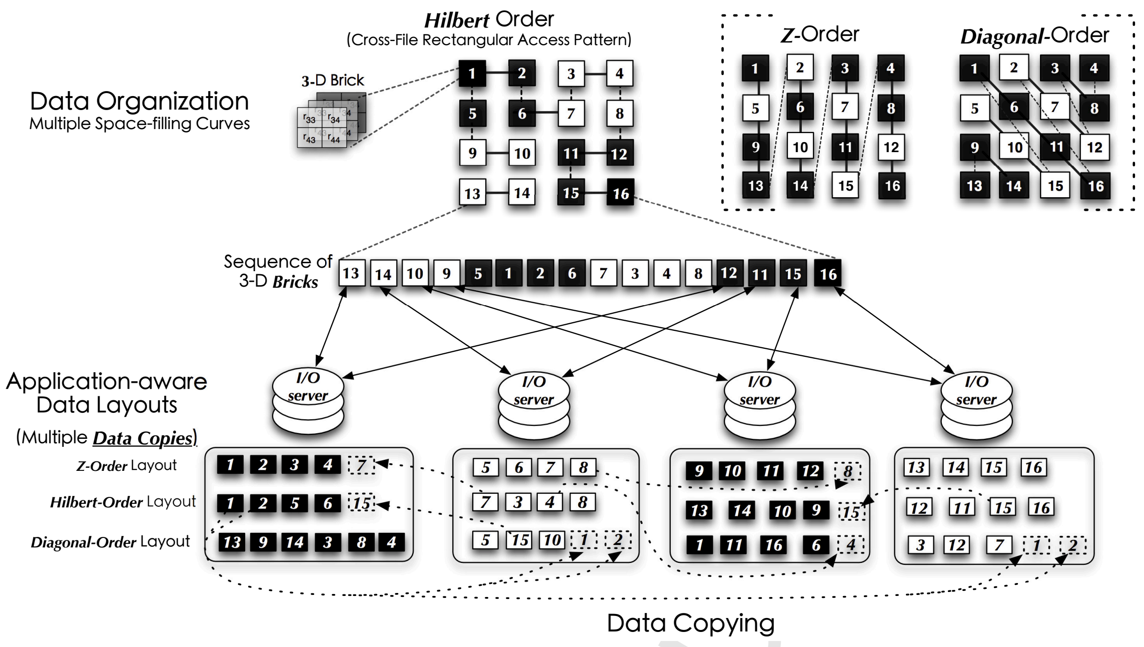


Figure 3: The Application-aware Data Layout and Data Copies

sirable. But compared to a static tree, it is also relatively more time-consuming and complicated.

Normally, RS data products are subdivided into standard scenes according to some global subdivision grid model. Where, a data scene may spans several degrees of longitude in lat-long geographical coordination, like 5^o for Modis data. Under this consideration, *pipsCloud* adopts a hybrid solution, which combines Hilbert R^+tree with GeoSOT global subdivision grid model. As is showed in figure 4, the global RS data are

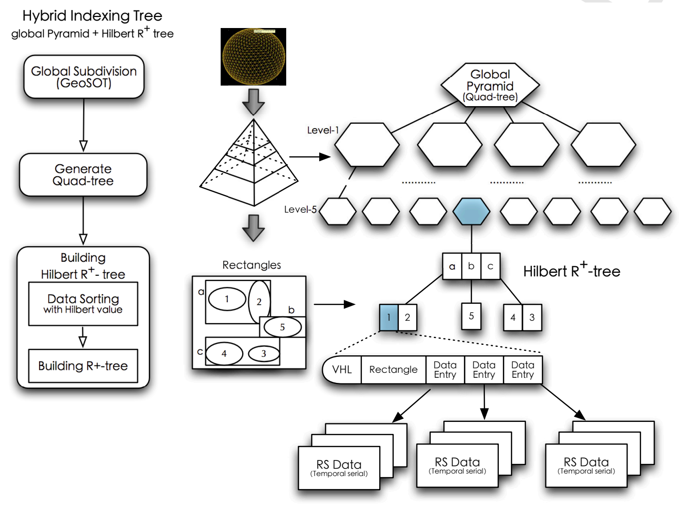

Hybrid Indexing Tree
global Pyramid + Hilbert R+ tree
Global Subdivision (GeoSOT)
Generate Quad-tree
Building Hilbert R+- tree
Data Sorting with Hilbert value
Building R+-tree
Global Pyramid (Quad-tree)
Level-1
Level-5
Rectangles
Hilbert R+-tree
VHL
Rectangle
Data Entry
RS Data (Temporal serial)

Figure 4: Optimal RS Data Indexing with Hilbert R^+tree and Global Subdivision Grid

firstly grouped through a deferred quad-subdivision of GeoSOT global subdivision grid model. Normally, through several levels of quad-subdivision (like 5 level) a quad-tree could be constructed. While if the a RS dataset covers a really big region that large than the GeoSOT grid, then this dataset would be logically further divided into data blocks. Then the RS datasets or data blocks inside the geographical region of each leaf node of quad-tree would be re-arranged according the Hilbert value of the center of the rectangles (i.e., MBR of the RS data or blocks). Following this way, each RS dataset or data blocks would be encoded with a unique GeoSOT-Hilbert code which consists of both GeoSOT code and Hilbert value. Given the Hilbert ordering, we generate new tree nodes and assign rectangles of RS data to these tree nodes sequentially. Wherein, the non-leaf node contains LHVs (Largest Hilbert Value) and also geographical region of the rectangles. Then by recursively sorting these new nodes by the Hilbert value of the rectangle of it and creating new nodes with higher level, a dynamic R^+tree could be generated. The leaf node of this hybrid Hilbert R^+tree is a data entry node, which contains the information (URL) of a temporal serial of RS data that inside the rectangle of node.

With the unique GeoSOT-Hilbert code, the Hilbert R^+tree and the RS metadata repository could be easily connected. For each RS dataset indexed in the leaf node of Hilbert R^+tree, the GeoSOT-Hilbert code as well as MBR of the data its-self and the data blocks inside this data would all be stored together with metadata in HBase. Accordingly, the searching of a given data region would be started from the root, it descends the quad-tree of GeoSOT global grid model, and then visits the nodes in the Hilbert R^+tree that intersect the desired rectangle to get the GeoSOT-Hilbert code of the data so as to access metadata in HBase and acquire imageries with URL.

4.2.4. RS Data Subscription and Distribution

Data sharing is paramount especially in the scenario where enormous RS data are geographically scattered across data centers. *pipsCloud* provides on-demand data sharing to a wide range of users in a data subscription manner. Through data subscription, the required RS data retrieved by search condition can be re-arranged and virtually mounted to the local storage of user's virtual machine or virtual environment as local catalogs (figure 5). Unlike conventional ways of data sharing, the subscribed RS data could be accessed and shared in a more easy-of-use and intuitive way through virtual data catalog mounted locally. Once when there is an up-to-date distribution of the subscripted data, data users may be informed and choose to update the virtual data catalog mirror without extra data retrieval or downloading.

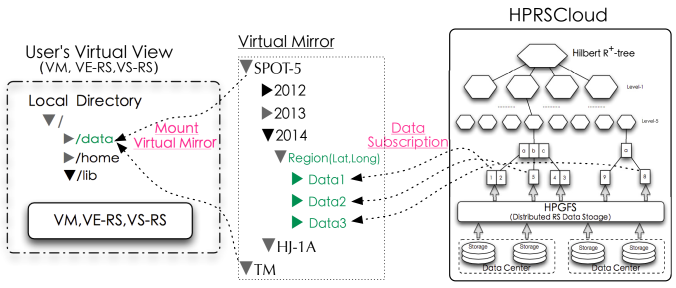


Figure 5: Data Subscription and Virtual Catalog Mirror

The flow of RS data subscription is as follows:

- *Image Data Retrieval*: Users can get the list of the request RS data through data retrieval. Normally different ways of data retrieval are also offered, including inquiry condition and visible data selection through map. The inquiry condition could be the synthesis of satellite, sensor, data product type, resolution, spatial region of data and time span.
- *Constructing Virtual Data Catalog Mirror*: Re-organize the retrieved RS data as in user or application customized catalog fashion, then link these data to form a virtual data catalog mirror.
- *Mount Virtual Data Catalog Mirror*: Take this catalog mirror as a shared network disk space and virtually mount it to the local disk of the user's VM or VE. Accordingly, data users could access any of the RS data inside the virtual data catalogs without extra data downloading operations.

Quick and efficient RS data distribution and browsing of geographically distributed massive multi-resolution RS image data is another challenging issue in a remote sensing cloud. *pipsCloud* offers GeoSOT global subdivision grid (GSG) model together with Hilbert Space-filling curve and code to create a Hilbert global image pyramid. Based on global image pyramid, the MapServer[11] is employed as the major platform for on-line RS data distribution and map browsing. For performance efficiency, TileCache[12] is used for caching the requested hot image tiles. Finally, the image data are published as WMS service for map displaying and retrieval by web-based map client like OpenLayers[13] or Ka-Map[14]).

4.3. VE-RS: RS-specific HPC Environment as A Service

VE-RS offers a RS-specific cluster environment as a service on top of OpenStack enabled cloud framework. Based on the VMs or BMs provided by cloud framework, VE-RS could build virtual HPC cluster through automatic deployment of cluster auto-configuration tools like AppScale. By means of SlatStack, VE-RS allows customized deployment of RS softwares such as ENVI, GDAL and ERDAS, together with HPC tools like MPI, MapReduce, Torque and Ganglia on VMs or PMs in the virtual cluster. Furthermore, *pipsCloud* also provides easy-to-use interfaces for auto-deployment of RS-specific HPC cluster with the APIs of ApppScale and SlatStack.

For efficient managing of RS big data, VE-RS adopts a parallel file system named HPGFS especially for RS imageries. To solve the poor I/O performance introduced by intensive irregular I/O pattern, HPGFS adopts application aware data layout policy so as to exploiting data locality and reduce data movements. Moreover, for easy but productive programming, VE-RS provides RS-GPPS, a generic parallel skeletons for large-scale RS data processing applications on MPI-enabled cluster environment. It also adopts a distributed RS data structure with fine-designed data layout control across distributed memories for efficient loading and communicating of RS big data. In addition, VE-RS adopts Torque scheduler as local resource manager and scheduler in virtual cluster, and ganglia for system monitoring.

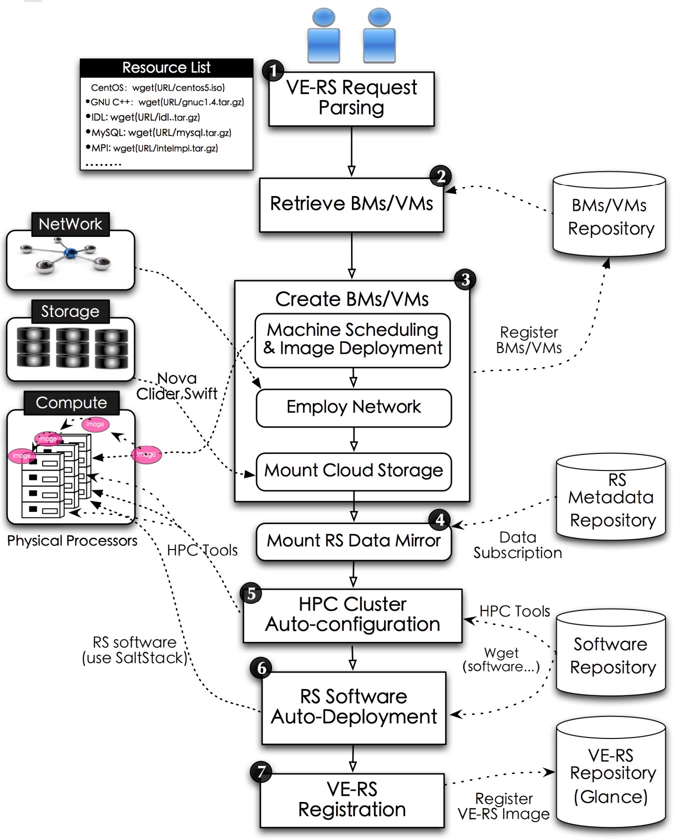


Figure 6: The Generation and Auto-deployment of VE-RS

When a Virtual HPC cluster is requested, the generation and

[11]MapServer, http://www.mapserver.org
[12]TileCache, http://tilecache.org
[13]OpenLayers, http://en.wikipedia.org/wiki/OpenLayers
[14]Ka-Map, http://ka-map.maptools.org/index.phtml?page=home.html

auto-deployment of VE-RS is invoked as showed in figure 6. Firstly, the VE-RS customization request is parsed into detailed resource requirement, such as the number or type of CPU, Memory, network as well as cloud storage. Secondly, retrieve available BMs or VMs in the BMs/VMs repository. Thirdly, if the existing BMs/VMs could be fit the requirement then create it according to user needs, including physical machine scheduling by nova-scheduling, BM/VM image deploying, employing network by nova-network, mount cloud storage from Cinder or Swift and register it to BMs/VMs repository. Fouthly, employ cluster auto-configuration tools like AppScale to automatically build HPC cluster with HPC tools like MPI/MapReduce, Torque and Ganglia. Fifthly, use SlatStack to automatically deploy programming model for RS, and other RS software like ENVI, GDAL and ERDAS. Following this way, a RS-specific cloud-enabled cluster environment would be automatically generated and registered into VE-RS repository for later use.

4.3.1. *On-demand HPC Cluster Platforms with Bare Meta Provisioning*

In this paper, we adopt xCAT to extend the dominant OpenStack platform for supporting bare-metal provisioning, and use KVM hypervisor for VMs provisioning. OpenStack consists of a collection of software components, including Nova for computing resource (VMs/BMs) management, Glance for image management and Swift for building cloud storage. Normally, OpenStack basically offers virtual machines and the resource visualization is enabled by some hypervisors like Xen or KVM. As is showed in figure 7, we use KVM hypervisor for the creation and management of VMs from the pool of physical machines. When a virtual machine is requested, Nova-compute of OpenStack will invoke the API of KVM for the creation and deployment of VMs. While, the virtualization and deployment of network resources is conducted by nova-network.

Actually, bare-metal provisioning is not directly supported in OpenStack. Hypervisor xCAT as a scalable distributed resource-provisioning tool, provides unified interfaces for discovery and software deployment of physical machines. However, to enable bare meta machines in OpenStack through xCAT, a bare-metal driver for xCAT should be integrated in Nova-compute component ([24]). Normally, Nova-compute uses libvirt library to manage different hypervisors for diverse virtualization approaches. In this context, the bare-metal driver for xCAT is needed as an alternative to the libvirt driver. On one hand, xCAT driver deals with the bare-metal (BM) machine requests from Nova-compute, and on the other hand it communicates with xCAT to complete resource provisioning.

As is showed in figure 7, when a HPC cluster environment for RS data processing with bare-metal machine is requested by user, the nova scheduler will choose a nova-compute node and pass the request to the Libvert driver of it. Then, the Libvert driver would invoke the so implemented bare-metal driver of xCAT and transfer the request to xCAT. Consequently, the xCAT will take charge of everything. It gets the information of bare-metal machines from BM/VMs database, and downloads the system images with OS and software needed for building HPC cluster for RS. Then, xCAT activate the boot loader

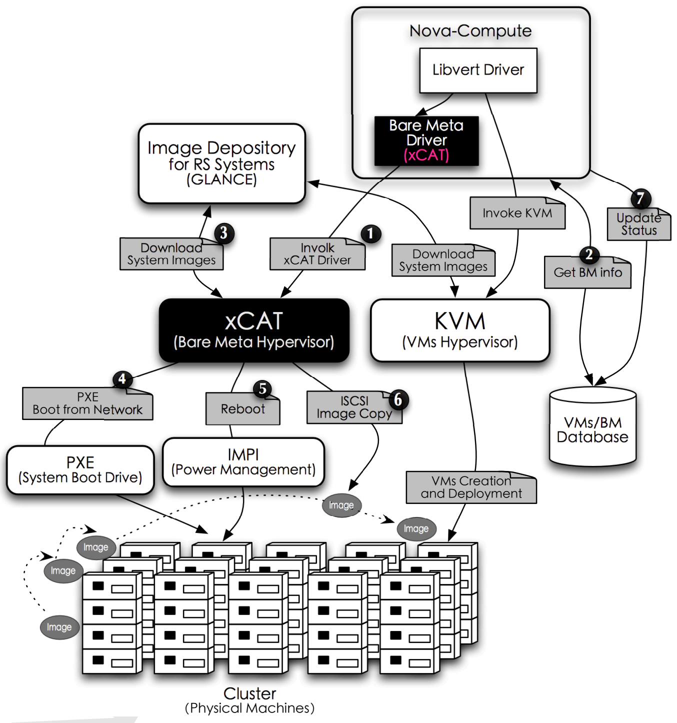


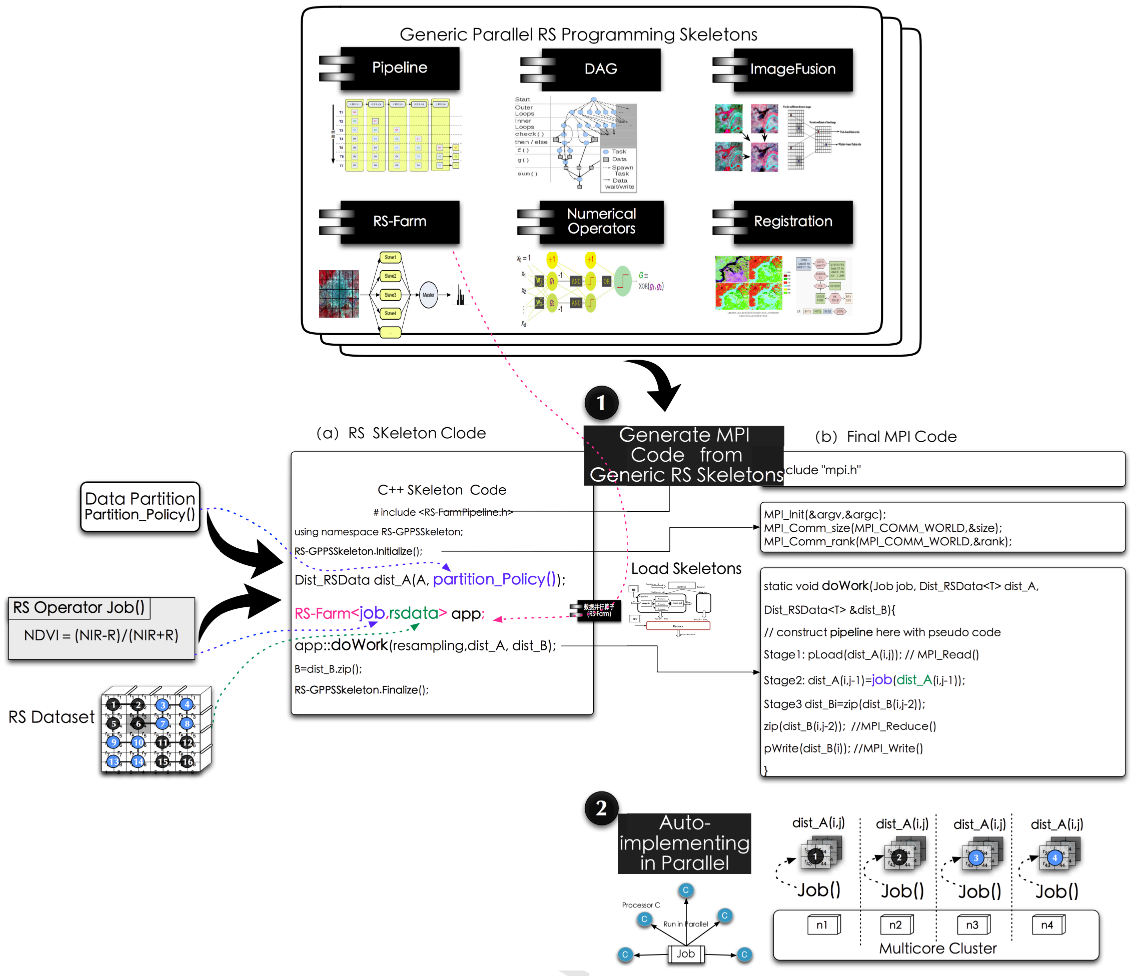
Figure 7: Bare Meta Provisioning in Cloud with xCAT

of the physical machine using PXE (Preboot Execute Environment), and power on the machine with power management driver IMPI. After that, xCAT boots the physical machine from network and deploys it with specified system image (OS and software). Finally, the register the information of bare-meta machine into the BM/VMs database. In case that the bare-metal machine is running, the xCAT is also responsible for the managing and monitoring the status of it. Following this way, not only VMs but also BMs could be accommodated for on-demand needs of variable HPC cluster platform for RS applications, so as to decrease performance penalty.

4.3.2. *Skeletal Programming for RS Big Data Processing*

Cluster-based HPC platforms will be characterized by extreme scale and a multilevel hierarchical organization. Efficient and productive programming for these systems will be a challenge, especially in the context of data-intensive RS data processing applications.

To properly solve the aforementioned problems, we propose RS-GPPS, Generic Parallel Programming Skeletons for massive remote sensing data processing applications enabled by template class mechanism, and work on top of MPI. Generic parallel algorithms are abstract and recurring patterns lifting from many concrete parallel programs and conceal parallel details as skeletons. This approach relies on type genericity to resolve polymorphism at compile time, so as to reduce the runtime overhead while providing readability of high-level. We focus on so-called class templates, which are parameterized computations patterns and used to implement algorithm skeletons (figure 8). The main contribution of RS-GPPS is that it provides



```
C++ SKeleton Code
# include <RS-FarmPipeline.h>
using namespace RS-GPPSSkeleton;
RS-GPPSSkeleton.Initialize();
Dist_RSData dist_A(A, partition_Policy());
RS-Farm<job,rsdata> app;
app::doWork(resampling,dist_A, dist_B);
B=dist_B.zip();
RS-GPPSSkeleton.Finalize();
```

```
clude "mpi.h"
```

```
MPI_Init(&argv,&argc);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
```

```
static void doWork(Job job, Dist_RSData<T> dist_A,
Dist_RSData<T> &dist_B){
// construct pipeline here with pseudo code
Stage1: pLoad(dist_A(i,j)); // MPI_Read()
Stage2: dist_A(i,j-1)=job(dist_A(i,j-1));
Stage3 dist_Bi=zip(dist_B(i,j-2));
zip(dist_B(i,j-2)); //MPI_Reduce()
pWrite(dist_B(i)); //MPI_Write()
}
```

Figure 8: The Skeletal Parallel Programming Model for RS Big Data in pipsCloud

both generic distributed RS data structure and generic parallel skeletons for RS algorithms.

Generic distributed RS data structure is a MPI-enabled data structure that allows the distribution of RS data across computing nodes in cluster. The massive RS data object with multi-dimensional image data and complex metadata, whose data are slicing into blocks and distributed among nodes is abstracted and wrapped as generic distributed RSData class template. Also the MPI one-sided messaging primitives and serialization of complex data structure are be used in data structure template, to offer the simple data accessing and residing of whole massive RS data in distributed memory space among nodes like local ones.

Generic parallel skeletons for RS algorithms that perform computations on distributed RS data structures could be used for RS algorithms with different computation modes. These skeletons express parameterization of parallelism without concern for implementation details like data distribution and task partition, complicated access modes of RS data, and all low-level architecture dependent parallel behaviors. When a generic parallel skeleton is instantiated and declared, the computations on distributed remote sensing data objects are performed. Firstly, the task would be divided into subtasks by two-stage task partition strategy, first nodes then intra-nodes, which consistent with the data partition strategy. Then to actually load the data blocks owned by each node concurrently through the parallel I/O operations enabled by parallel file system. Finally, the user defined remote sensing sequential code encapsulated in job class would be implemented in parallel by each process. In this situation, the easy of parallel programming could be offered with a minimum concern for architecture-specific parallel implementation behaviors.

4.4. VS-RS: Cloud-enabled RS Data Processing System

VS-RS offers on-demand workflow customization and dynamic processing for various large-scale RS applications in cloud as on-line services on top of a cloud-enabled HPC cluster environment VE-RS. It consists of order manager, resource scheduler, runtime for collaborative workflow processing, and data or algorithm repositories. Wherein, the order manager is responsible for parsing the requested RS data processing orders into abstract collaborative workflows according to the workflow repositories. While the resource scheduler adopts an optimal scheduling strategy to conduct an optimized resource mapping for the abstract workflow to form a concrete one, including data, algorithm and computing resources. Actually, these concrete workflows are constructed dynamically through dynamic optimal resource allocation during runtime according to the monitored status of resources and system. Meanwhile, the kepler-enabled workflow processing runtime dynamically implements each step of the workflow with allocated resources on Local cluster in VE-RS or launched it to remote data centers, and finally coordinates the whole collaborative workflow processing procedure.

4.4.1. Dynamic Workflow Processing for RS Applications in Cloud

The RS data processing applications are typically some what on-the-flow processing. The whole processing procedure are consists of several processing stages. Take a typical multi-stage pre-processing for instance, it includes L0 processing, radiometric correction (RC), geometric rectification (GR) and followed by fine rectification (FR) or ortho-rectification (OR). Each of the processing step produces corresponding RS data products. This kind of RS data processing workflow generates a data-driven processing flow, each step depends on the output data of the preceding step as input data.

In a traditional RS data processing system, the workflows for various RS applications are always predefined as static ones. But many RS applications normally demand for on-demand workflow processing capability. In this scenario, the dynamic customization of application-specific workflows according to the variable needs is essential. Moreover, the RS observation data generally are scattered among different satellite data centers geographically. So a large-scale RS applications like regional to global drought monitoring ([56]), normally need a collaboration of several data centers. In this sense, not only the data, but also the processing workflows from other data centers or scientist need to be shared and cooperated. To complicated the situation that the unstable computing environment across data centers will inevitably lead to the failure of whole processing. Therefore, a dynamic resource allocation and scheduling is essential for a distributed workflow collaboration across data centers.

To solve the dynamic RS workflow processing issue, we put forward a two-level worklow processing scheme with both the abstract workflow and the concrete one. The abstract workflow is used to logically represent the complex processing procedure customized by domain scientists. Each processing step in the abstract workflow is a logical function rather than an actual processing program, whose actual algorithm and data resources are not decided yet. When an abstract workflow is customized, it would be expressed in XML format and stored in the RS workflow depository. While, the concrete workflow is not built at once but dynamically constructed and implemented by an efficient workflow engine through dynamic resource mapping during runtime. In case when one step of the abstract workflow is allocated with required algorithm, data and computing resources then it will be launched to designated nodes or remote data center for parallel implementing and collaboration by workflow engine. Following this way, the large-scale RS application could be dynamically implemented on HPC cluster or across data centers for global workflow collaboration with optimal runtime resource allocation according to resource status.

For dynamic workflow processing with high efficiency, a proper workflow engine is also of vital importance. Currently, scientific workflows ([57]) are gradually employed to formalize and enable distributed scientific processing for in various disciplines, such as Physics and Earth Science. Compared to the traditional control flow oriented workflow system, scientific workflow management system (SWFMSs) like VIEW ([57], Kepler ([58] and Pegasus ([59]) are typically data flow oriented. In this paper, we adopt Kepler engine that are most widely used as the main scientific workflow runtime and management system.

On-demand workflow customization serviced by VS-RS in *pipsCloud* is demonstrated in figure 9. When a RS processing workflow creation or customization is requested through simple drag-and-drop of algorithms in graphical user portal, a searching operation is triggered in RS algorithm repository for a name catalog of various registered RS algorithms. Then follows the composition of user selected algorithms together with the control logics among these algorithms to form an abstract workflow. Here the algorithms which form workflows only refer to the functional names of them rather than the real algorithm resources with executable program. The customized abstract workflows expressed in XML format are then registered into RS abstract workflow repository for further processing.

Dynamic RS workflow runtime enabled by Kepler workflow engine and a two-level workflow scheme is illustrated in figure 9 for collaborative large-scale RS workflow processing across data centers or clouds. In case when a RS data products processing order is requested by user through portal, then a RS data processing procedure would be launched on the RS workflow runtime for processing.

The dynamic processing of large-scale collaborative workflows on Kepler-enabled runtime goes as follows:

– Firstly, *Abstract workflow matching* is a responsible for interpreting the requested orders into abstract RS worflows without allocation. With the key word "Product Type", runtime searches for the corresponding abstract workflow in the workflow repository for each RS order requested through cloud web portal. While, the abstract workflow interpreted only tells the blueprint of the data processing procedure, includes functional name of each step as well as the control logic among them. But the actual processing program or data needed for processing in each workflow step is not decided yet.
– Secondly, *Optimal Resource Allocation* continues to conduct optimal resource mapping for each workflow step according to the current status of various resource and system. Wherein, the resources here refers to three main categories of resources including algorithm resources with actual programs, various RS data required for processing and also processing resources like processors and network which are needed for execution. Initially, a knowledge query from product knowledge repository is invoked for acquiring the knowledge rules for this designated RS data product. The

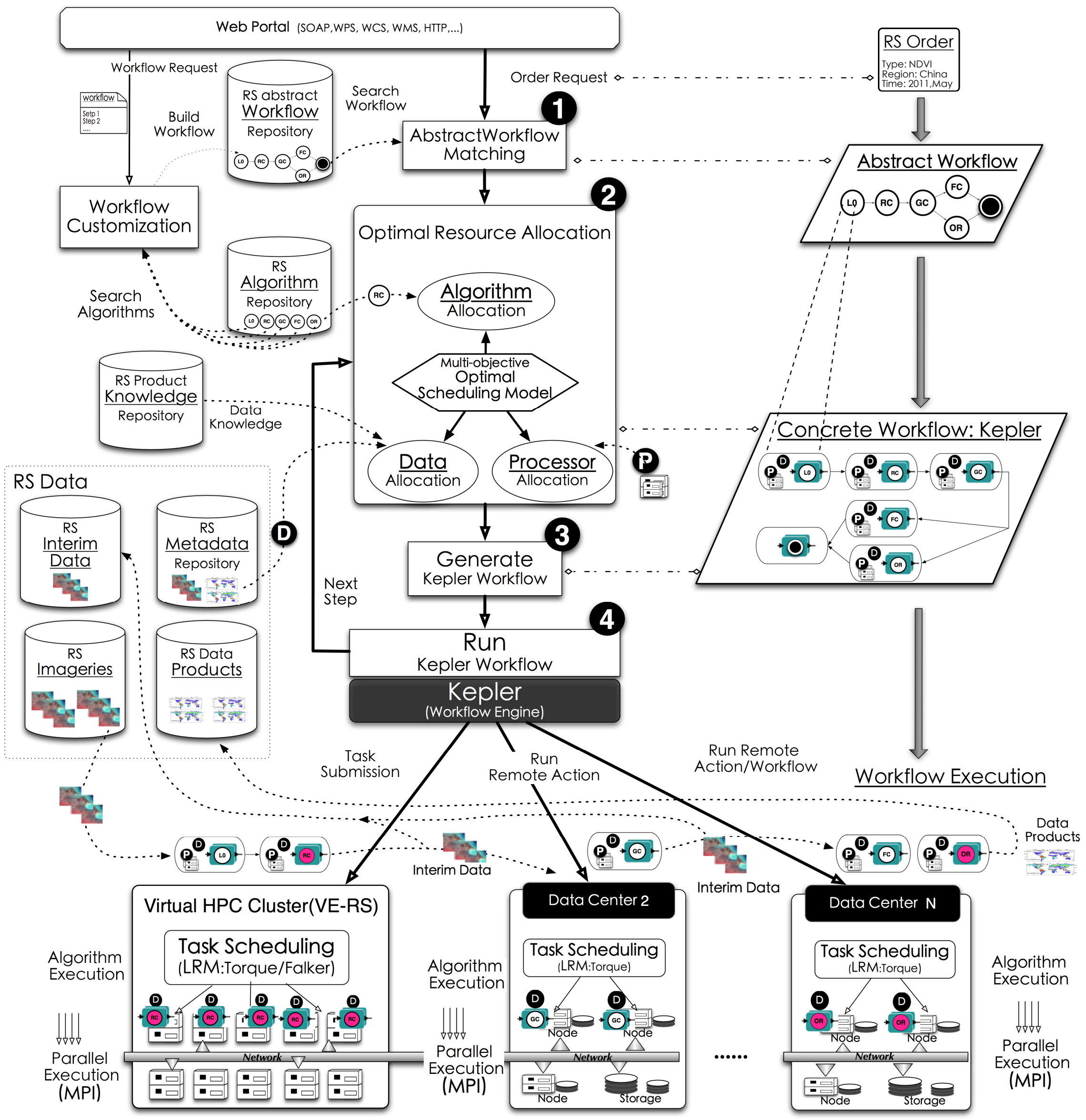


Figure 9: Dynamic and optimal Workflow processing for Large-scale RS Applications with Kepler

product knowledge represents in rules mainly indicate the requirement of the RS data, such as the resolution or sensor of the RS imageries, auxiliary data as well as some parameter data or file needed. Then with the knowledge rules of data products, there follows the generating of condition statement for RS data query. Accordingly, a list of RS imageries or auxiliary data could be draw out from the data repository for further processing. After that goes the algorithm allocation, the candidate executable programs are deposit from the algorithm repository with the key word of functional name of algorithms. In addition, the available computing resources like processor and network are also allocated for processing according to the current status through monitoring.

However, the problem worth noticing is that how to choose the resources that fit the data processing procedure best so as to achieve an optimal performance QoS target. The main reason for that would be the plenty of candidate program of algorithms, RS data as well as computing resources meeting the requirement of these specified processing steps in the workflow of data products processing. Nevertheless the location of RS data, network bandwidth and usage as well as the CPU capacity and usage of processors are all the factors that affect the total performance QoS target of the whole workflow.

To achieve optimal resources for a certain step of workflow, the workflow runtime employs an optimal resource scheduling model with performance QoS target function and user-customized scheduling rules. Wherein the QoS target function is a target function which take data amount, network width, program performance and processor performance into considerations each factor of which a assigned with a empirical weight value. Moreover, the scheduling rule could also be customized and stored into rule repository in a key-condition-threshold style. Some basic scheduling rules like near-data computing for reducing data movement are also provided in the workflow scheduling for optimization. According to the performance QoS target function and scheduling rules, we could select the best-fit RS data, algorithm programs and computing resources from candidates with an optimal final performance QoS. Then these resources are allocated to this certain processing step of wokflow.

– Thirdly, *Partly generating concrete Kepler workflow* from abstract workflow with allocated resources. Here runtime only generates part of the Kepler workflow for certain processing steps with allocated resources. Each step of the Kepler workflow is then represented as an executable Kepler "actor".
– Fourthly, *Run Kepler workflow* on Kepler workflow engine. In case when the processing step of workflow is a local actor, then a PBS/Torque task submission is triggered to the LRMs (Local Resource Manager). Then the LRMs would launch the program of this workflow processing step onto the allocated processors (VMs or BMs) in virtual HPC cluster environment in cloud and executed it in parallel. While, if the workflow step is "sub-workflow" expressed as a web service actor, Kepler would directly invoke the web service interfaces for execution. If the processing step is a remote job execution, then a remote action is invoked with a remote job submission. After receiving the job submission, the LRMs of the remote data center would soon run the program on processors and final feedback with interim data. The interim data would be cached into the interim data repository so as to preparing for data transferring to next data center or cloud for further processing of workflow. As is showed in figure 9, in the processing workflow of producing global NDVI data products, runtime firstly executes two initial steps of L0 processing and geometric correction (GC) on the virtual HPC cluster of cloud, then pass the interim data products to Data Center 2 and launch a remote execution of radiometric correction (RC), when RC is finished, then Kepler triggers a remote job submission to the LRMs of data center N for the parallel implementing of the last two programs.
– Finally, continue the workflow processing recursively from optimal resource allocation, generating Kepler workflow to implementing workflow collaboratively until end of the workflow procedure. Following this way, the entire complex processing workflow could be generated and implemented dynamically on Kepler engine with an nearly optimal performance QoS of the whole processing procedure. When the workflow ends, the RS data products would be registered into RS data product repository for downloading.

Consequently, with the logical control and data transferring among data centers, a distributed workflow among different data center or cloud systems could be collaboratively implemented. Each step of the workflow is implemented with the optimal allocated resources according to current system status. Even when a failure of the allocated resources are occurred, then a re-allocation of the resource would be triggered for the a re-build and re-run of the Kepler workflow.

5. Experiments and Discussion

The *pipsCloud* which offers high-performance cloud environment for RS big data has been successfully adopted to build the Multi-data-center Collaborative Process System (MDCPS). By virtue of the data management service in pipsCloud, the multi-source raw RS data, interim data and also data products could all be efficiently managed and accessed. Through the VE-RS service in pipsCloud, a customized virtual HPC cluster environment is easily built and equipped with RS softwares, parallel programming model and large-scale task scheduling especially for RS applications. By employing the VS-RS service offered in pipsCloud, MDCP are well constructed and equipped upon the VE-RS cluster environment with order management, workflow engine and depository for RS algorithms and workflows. Furthermore, enabled by the Kepler workflow engine, the complex processing procedures for global RS data products are customized as dynamic workflows that implemented through the collaboration cross multiple data centers. These processing is dynamic since the concrete workflows are not predefined but dynamically formed through runtime resource mapping from abstract workflows to data centers.

Actually, MDCPS connects several national satellite data centers in China, such as CCRSD[15], NSOAPS[16], NMSC[17]. It offers on-line processing of regional to global climate change related quantitative RS data products with these multi-source RS data across data centers. The RS data products generated by MDCPS include vegetation related parameters like NDVI [18] and NPP[19], radiation and hydrothermal flux related parameters like AOD[20] and SM[21], as well as global ice change and mineral related parameters. The 5-day global synthetic NDVI parameter product in 2014 generated using MODIS 1km data is showed in figure 10. The 5-day global synthetic NPP parameter products which also produced with MODIS 1km data in day 211to 215 and day 221 to 225 in 2014 are relatively demonstrated in sub figure (a) and (b) in figure 11.

The performance experiments on typical RS algorithms with both increasing processors and data amounts are carried out for the validation of the scalability of the pipsCloud platform. In this experiment, two MPI-enabled RS algorithms are chosen for implementing, including NDVI and NPP. Meanwhile, pipsCloud platform offers a virtual multi-core cluster with 10 nodes connected by a 20 gigabyte Infiniband network using RDMA(Remote Direct Memory Access) protocol. Each node is a bare-meta provisioned processor with dual Intel (R) Quad core CPU (3.0 GHz) and 8 GB memory. The operating system

[15]CCRSD: China Centre for Resources Satellite Data
[16]NSOAPS: National Satellite Ocean Application Service
[17]NSMC: National Satellite Meteorological Centre
[18]NDVI: Normalized Differential Vegetation Index
[19]NPP: Net Primary Productivity
[20]AOD: Aerosol optical depth
[21]SM: Soil Moisture

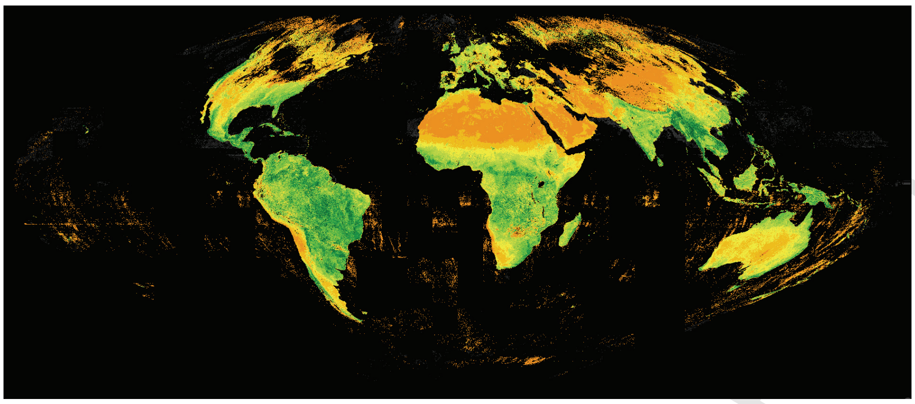
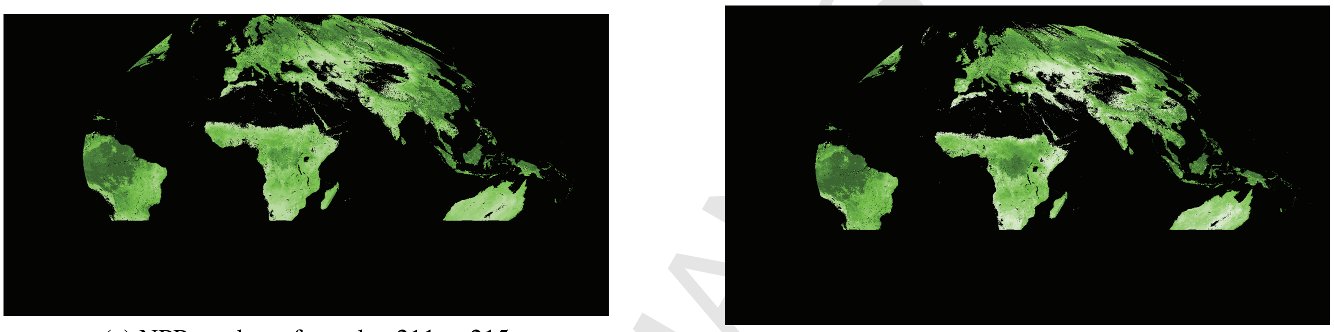


Figure 10: The 5-day synthetic Global NDVI products in 2014



(a) NPP products from day 211 to 215

(b) NPP products from day 221 to 225

Figure 11: The 5-day synthetic Global NPP products in 2014

was Cent OS5.0, the C++ compiler was GNU C/C++ Compiler with optimizing level O3, and the MPI implementation was MPICH.

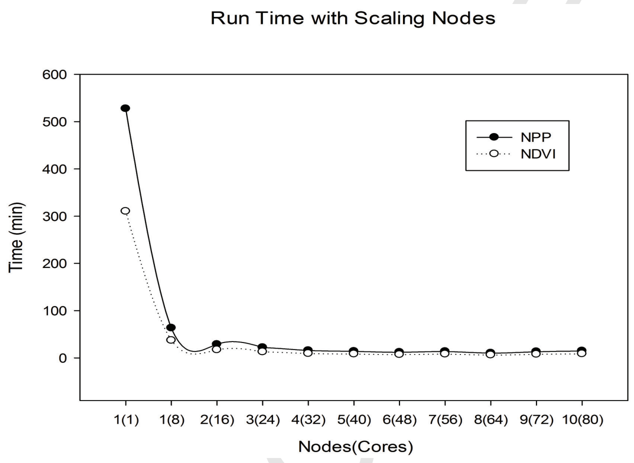


Figure 12: Run Time of NPP and NDVI with Scaling nodes

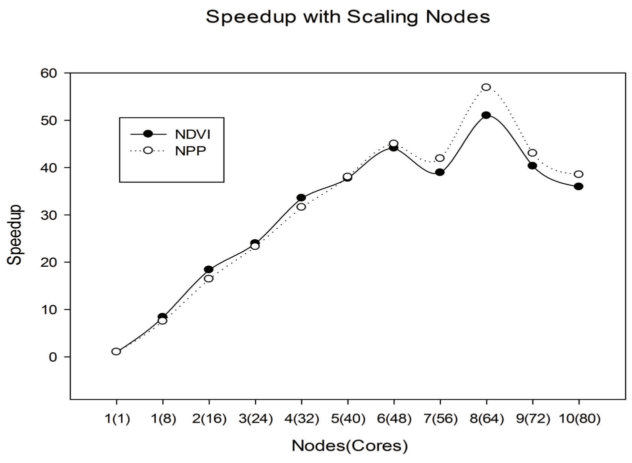


Figure 13: Speedup of NPP and NDVI with Scaling nodes

The runtime and speedup performance merit of both NPP and NDVI with increasing numbers of processors are illustrated relatively in figure 12 and figure 13. As is demonstrated in sub figure (a), the run time merit curves of these two algorithms decrease almost linearly especially when scale to less than 4 processors (32 cores). However, the decrease rate is mush slower when scale from 5 processors (40 cores) to 10 processors (80 cores). The main reason for that would be the total run time which is relatively small makes the speedup not that obvious, since the system overhead could not be omitted. The same trend is also showed in sub figure (b) that the speedup metric curves of both two algorithms soar up linearly when scaling to 10 processor (80 cores) .

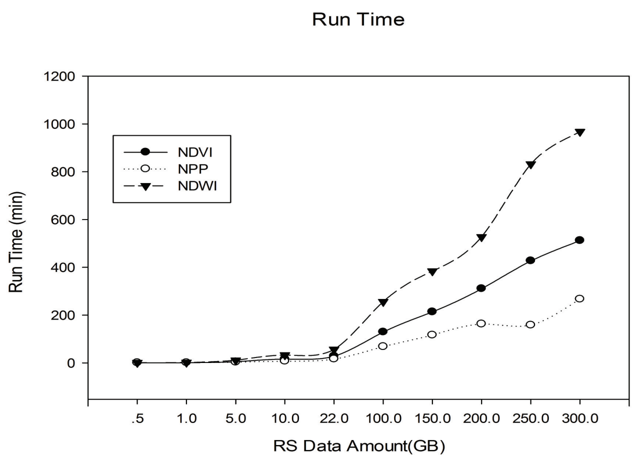


Figure 14: Run Time of NPP and NDVI with Increasing Data Amount

With the amount of RS data increasing from 0.5 gigabytes to about 300 gigabytes, the experimental result is depicted in figure 14. Judging from the performance curves demonstrated, the MPI-enabled NPP and NDVI algorithms implemented on pipsCloud both show their excellent scalability in terms of data.

6. Conclusion

The Cloud computing paradigm has been widely accepted in the IT industry with highly matured Cloud computing middleware technologies, business models, well-cultivated ecosystems. Remote sensing is a typical information associated zone, where data management and processing play a key role. With the advent of high resolution earth observation era give birth to the explosive growth of remote sensing (RS) data. The proliferation of data also give rise to the increasing complexity of RS data, like the diversity and higher dimensionality characteristic of the data. RS data are regarded as RS "Big Data".

In this paper we discusses how to bring the cloud computing methodologies into the remote sensing domain. We focus the overall high performance Cloud computing concepts, technologies and software systems to solve the problems of TS big data. *pipsCloud*, a prototype software system for high performance Cloud computing for RS is proposed and discussed with in-deep discussion of technology and implementation. As a contribution, this paper bring a complete reference design and implementation of high performance Cloud computing for remote sensing.

In the future applications, such as smart cities and disaster management, the great challenges will arise due to fusion of huge remote sensing data with other IoT data. The *pipsCloud*, benefiting from the ubiquity, elasticity and high-level of transparency of cloud computing model, could be able to manage and process the massive data, meeting the future applications requirements.

7. Acknowledgement

This research was supported by the National High Technology Research and Development Program of China ("863" Program) (No. 2013AA12A301) , National Natural Science Foundation of China (No. 41401512) and Youth Innovation Promotion Association CAS.

References

[1] Th. Udelhoven. Big data in environmental remote sensing: Challenges and chances, 12 2013.

[2] N Skytland. Big data: What is nasa doing with big data today. *Open. Gov open access article*, 2012.

[3] OGC-OpenGIS Consortium et al. The opengis abstract specification-topic 7: The earth imagery case, 1999.

[4] P. Gamba, Peijun Du, C. Juergens, and D. Maktav. Foreword to the special issue on "human settlements: A global remote sensing challenge". *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 4(1):5–7, March 2011.

[5] C.A. Lee, S.D. Gasster, A. Plaza, Chein-I Chang, and Bormin Huang. Recent developments in high performance computing for remote sensing: A review. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 4(3):508–527, Sept 2011.

[6] A Rosenqvist, M Shimada, B Chapman, A Freeman, G De Grandi, S Saatchi, and Y Rauste. The global rain forest mapping project-a review. *International Journal of Remote Sensing*, 21(6-7):1375–1387, 2000.

[7] Mathieu Fauvel, Jon Atli Benediktsson, John Boardman, John Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Paolo Gamba, A Gualtieri, M Marconcini, et al. Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, pages 1–45, 2007.

[8] Antonio J. Plaza. Special issue on architectures and techniques for real-time processing of remotely sensed images. *J. Real-Time Image Processing*, 4(3):191–193, 2009.

[9] Dongjian Xue, Zhengwei He, and Zhiheng Wang. Zhouqu county 8.8 extra-large-scale debris flow characters of remote sensing image analysis. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 597–600, Sept 2011.

[10] G. Schumann, R. Hostache, C. Puech, L. Hoffmann, P. Matgen, F. Pappenberger, and L. Pfister. High-resolution 3-d flood information from radar imagery for flood hazard management. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(6):1715–1725, June 2007.

[11] A.H.S. Solberg. Remote sensing of ocean oil-spill pollution. *Proceedings of the IEEE*, 100(10):2931–2945, Oct 2012.

[12] Bingxin Liu, Ying Li, Peng Chen, Yongyi Guan, and Junsong Han. Large oil spill surveillance with the use of modis and avhrr images. In *Remote Sensing, Environment and Transportation Engineering (RSETE), 2011 International Conference on*, pages 1317–1320, June 2011.

[13] A. Rosenqvist, M. Shimada, B. Chapman, K. McDonald, G. De Grandi, H. Jonsson, C. Williams, Y. Rauste, M. Nilsson, D. Sango, and M. Matsumoto. An overview of the jers-1 sar global boreal forest mapping (gbfm) project. In *Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International*, volume 2, pages 1033–1036 vol.2, Sept 2004.

[14] G. De Grandi, P. Mayaux, Y. Rauste, A. Rosenqvist, M. Simard, and S.S. Saatchi. The global rain forest mapping project jers-1 radar mosaic of tropical africa: development and product characterization aspects. *Geoscience and Remote Sensing, IEEE Transactions on*, 38(5):2218–2233, Sep 2000.

[15] Antonio J. Plaza and Chein-I Chang. *High Performance Computing in Remote Sensing*. Chapman & Hall/CRC, 2007.

[16] Yan Ma, Lizhe Wang, Albert Y. Zomaya, Dan Chen, and Rajiv Ranjan. Task-tree based large-scale mosaicking for remote sensed imageries with dynamic dag scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 99(PrePrints):1, 2013.

[17] Xinyuan Qu, Jiacun Li, Wenji Zhao, Xiaoli Zhao, and Cheng Yan. Research on critical techniques of disaster-oriented remote sensing quick mapping. In *Multimedia Technology (ICMT), 2010 International Conference on*, pages 1–4, Oct 2010.

[18] Yuehu Liu, Bin Chen, Hao Yu, Yong Zhao, Zhou Huang, and Yu Fang. Applying gpu and posix thread technologies in massive remote sensing image data processing. In *Geoinformatics, 2011 19th International Conference on*, pages 1–6, June 2011.

[19] Yan Ma, Lingjun Zhao, and Dingsheng Liu. An asynchronous parallelized and scalable image resampling algorithm with parallel i/o. In *Computational Science-ICCS 2009*, volume 5545 of *Lecture Notes in Computer Science*, pages 357–366. Springer Berlin Heidelberg, 2009.

[20] Min Cao and Zhao-liang Shi. Primary study of massive imaging auto-processing system pixel factory. *Bulletin of Surveying and Mapping*, 10:55–58, 2006.

[21] S. Pandey, A. Barker, K.K. ", and R. Buyya. Minimizing execution costs when using globally distributed cloud services. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 222–229, April 2010.

[22] Daniel Mandl. Matsu: An elastic cloud connected to a sensorweb for disaster response. pages 1–22, 2011.

[23] K. Keahey and M. Parashar. Enabling on-demand science via cloud computing. *Cloud Computing, IEEE*, 1(1):21–27, May 2014.

[24] Jun Xie, Yujie Su, Zhaowen Lin, Yan Ma, and Junxue Liang. Bare metal provisioning to openstack using xcat. *Journal of Computers(JCP)*, 8(7):1691–1695, 2013.

[25] A. Remon, S. Sanchez, A. Paz, E.S. Quintana-Orti, and A. Plaza. Real-time endmember extraction on multicore processors. *Geoscience and Remote Sensing Letters, IEEE*, 8(5):924–928, Sept 2011.

[26] R. Rabenseifner, G. Hager, and G. Jost. Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 427–436, Feb 2009.

[27] Plaza A., Qian Du, Yang-Lang Chang, and King R.L. High performance computing for hyperspectral remote sensing. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of*, 4(3):528–544, Sept 2011.

[28] Yinghui Zhao. Remote sensing based soil moisture estimation on high performance pc server. In *Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on*, volume 1, pages 64–69, July 2010.

[29] Yanying Wang, Yan Ma, Peng Liu, Dingsheng Liu, and Jibo Xie. An optimized image mosaic algorithm with parallel io and dynamic grouped parallel strategy based on minimal spanning tree. In *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*, pages 501–506, Nov 2010.

[30] Xue Xiaorong, Guo Lei, Wang Hongfu, and Xiang Fang. A parallel fusion method of remote sensing image based on ihs transformation. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 3, pages 1600–1603, Oct 2011.

[31] Taeyoung Kim, Myungjin Choi, and Tae-Byeong Chae. Parallel processing with mpi for inter-band registration in remote sensing. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 1021–1025, Dec 2011.

[32] Yan Ma, Lizhe Wang, Dingsheng Liu, Peng Liu, Jun Wang, and Jie Tao. Generic parallel programming for massive remote sensing data processing. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 420–428, Sept 2012.

[33] Filip Blagojevi, Paul Hargrove, Costin Iancu, and Katherine Yelick. Hybrid pgas runtime support for multicore nodes. In *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model*, PGAS '10, pages 3:1–3:10, New York, NY, USA, 2010. ACM.

[34] Wei-Yu Chen, C. Iancu, and K. Yelick. Communication optimizations for fine-grained upc applications. In *Parallel Architectures and Compilation Techniques, 2005. PACT 2005. 14th International Conference on*, pages 267–278, Sept 2005.

[35] Nan Dun and K. Taura. An empirical performance study of chapel programming language. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 497–506, May 2012.

[36] J. Milthorpe, V. Ganesh, AP. Rendell, and D. Grove. X10 as a parallel language for scientific computation: Practice and experience. In *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 1080–1088, May 2011.

[37] Ciprian Dobre and Fatos Xhafa. Parallel programming paradigms and frameworks in big data era. *International Journal of Parallel Programming*, 42(5):710–738, 2014.

[38] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The eucalyptus open-source cloud-computing system. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 124–131, May 2009.

[39] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[40] Feng-Cheng Lin, Lan-Kun Chung, Wen-Yuan Ku, Lin-Ru Chu, and Tien-Yin Chou. Service component architecture for geographic information system in cloud computing infrastructure. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 368–373, March 2013.

[41] Bo Li, Hui Zhao, and Zhenhua Lv. Parallel isodata clustering of remote sensing images based on mapreduce. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010 International Conference on*, pages 380–383, Oct 2010.

[42] Mohamed H. Almeer. Cloud hadoop map reduce for remote sensing image analysis. *Journal of Emerging Trends in Computing and Information Sciences*, 3(4):637–644, April 2012.

[43] R. Nasim and A.J. Kassler. Deploying openstack: Virtual infrastructure or dedicated hardware. In *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*, pages 84–89, July 2014.

[44] A.B.M. Moniruzzaman, K.W. Nafi, and S.A. Hossain. An experimental study of load balancing of opennebula open-source cloud computing platform. In *Informatics, Electronics Vision (ICIEV), 2014 International Conference on*, pages 1–6, May 2014.

[45] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[46] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. Kvm: the linux virtual machine monitor. In *OLS '09: Ottawa Linux Symposium 2009*, pages 225–230, Jul 2007.

[47] Qasim Ali, Vladimir Kiriansky, Josh Simons, and Puneet Zaroo. Performance evaluation of hpc benchmarks on vmwares esxi server. In Michael Alexander and P Da Ambra, editors, *Euro-Par 2011: Parallel Processing Workshops*, volume 7155 of *Lecture Notes in Computer Science*, pages 213–222. Springer Berlin Heidelberg, 2012.

[48] Yi-Man Ma, Che-Rung Lee, and Yeh-Ching Chung. Infiniband virtualization on kvm. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 777–781, Dec 2012.

[49] S. Varrette, M. Guzek, V. Plugaru, X. Besseron, and P. Bouvry. Hpc performance and energy-efficiency of xen, kvm and vmware hypervisors. In *Computer Architecture and High Performance Computing (SBAC-PAD), 2013 25th International Symposium on*, pages 89–96, Oct 2013.

[50] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes. Enabling smart cloud services through remote sensing: An internet of everything enabler. *Internet of Things Journal, IEEE*, 1(3):276–288, June 2014.

[51] Mehul Nalin Vora. Hadoop-hbase for large-scale data. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 601–605, Dec 2011.

[52] Nan Lu, Chengqi Cheng, An Jin, and Haijian Ma. An index and retrieval method of spatial data based on geosot global discrete grid system. In *Geoscience and Remote Sensing Symposium (IGARSS), 2013 IEEE International*, pages 4519–4522, July 2013.

[53] Jeremy Zawodny. Redis: Lightweight key/value store that goes the extra mile. *Linux Magazine*, 79, 2009.

[54] QI Jianghui, ZHANG Feng, DU Zhenhong, and LIU Renyi. Research of the landuse vector data storage and spatial index based on the main memory database. *Journal of Zhejiang University(Science Edition)*,

13(3):365–370, 2015.

[55] Cloud computing adoption framework: A security framework for business clouds. *Future Generation Computer Systems*, 57:24 – 41, 2016.

[56] Meixia Deng, Liping Di, Genong Yu, A. Yagci, Chunming Peng, Bei Zhang, and Dayong Shen. Building an on-demand web service system for global agricultural drought monitoring and forecasting. In *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, pages 958–961, July 2012.

[57] Cui Lin, Shiyong Lu, Xubo Fei, A. Chebotko, Darshan Pai, Zhaoqiang Lai, F. Fotouhi, and Jing Hua. A reference architecture for scientific workflow management systems and the view soa solution. *Services Computing, IEEE Transactions on*, 2(1):79–92, Jan 2009.

[58] Bertram Ludscher, Bertram, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[59] Ewa Deelman, Gurmeet Singh, Mei hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *SCIENTIFIC PROGRAMMING JOURNAL*, 13:219–237, 2005.

***Biographies (Text)**

Dr. Lizhe Wang is a “ChuTian” Chair Professor at School of Computer Science, China Univ. of Geosciences (CUG), and a Professor at Inst. of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS). Prof. Wang received B.E. & M.E from Tsinghua Univ. and Doctor of Eng. from Univ. Karlsruhe (Magna Cum Laude), Germany. Prof. Wang is a Fellow of IET, Fellow of British Computer Society. Prof. Wang serves as an Associate Editor of IEEE T. Computers, IEEE T. on Cloud Computing, IEEE T. on Sustainable Computing. His main research interests include HPC, e-Science, and remote sensing image processing.

Yan Ma received the Doctor of Engineering degree from Chinese Academy of Sciences, in 2013. She is an Associate Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS), Beijing, China. Her research interests include high performance geo-computing and parallel remote sensing image processing. She is a member of the IEEE.

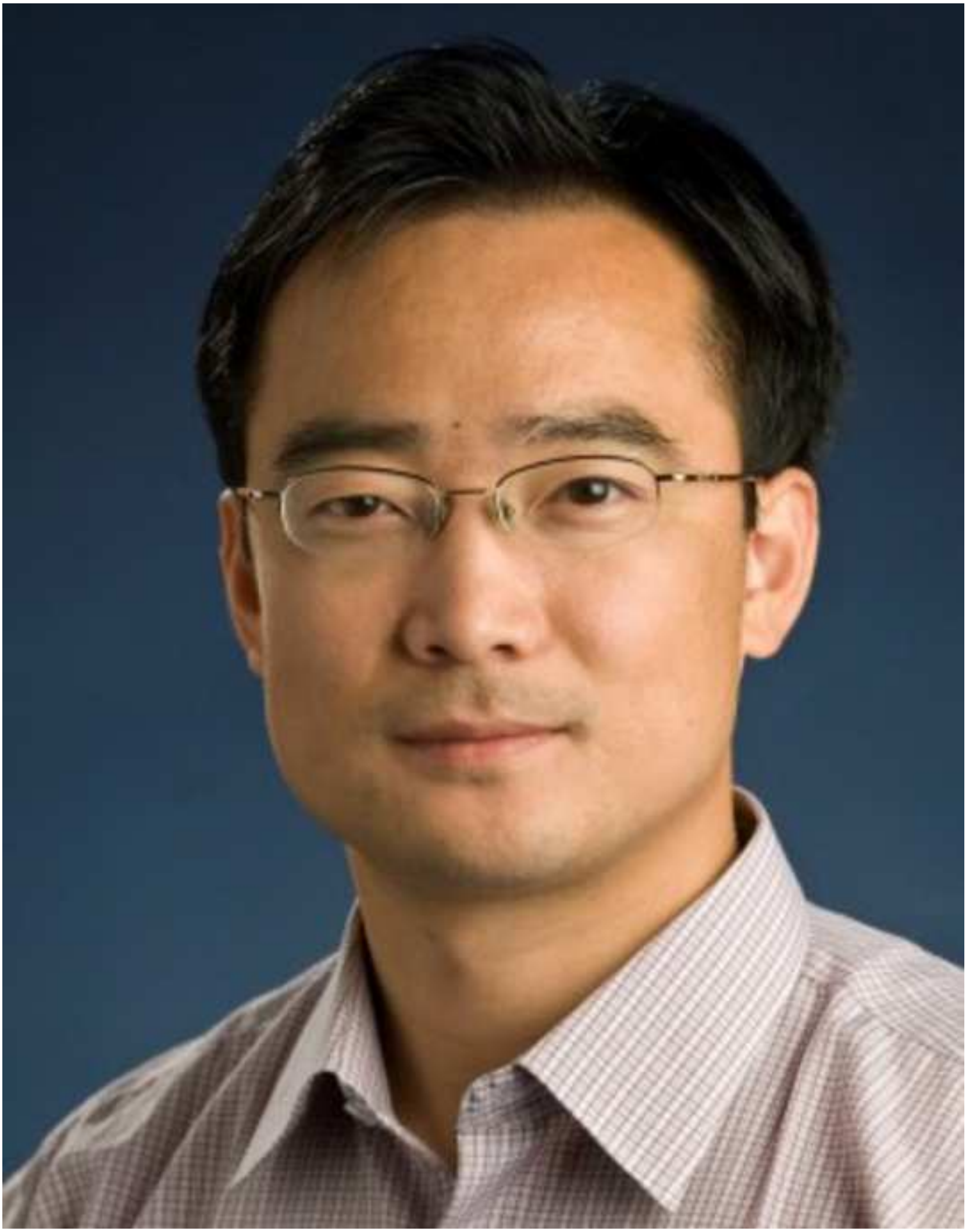
Jining Yan is a Ph.D. student in the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences. His main research interests include remote sensing data processing and information service, cloud computing in remote sensing.

Dr. Victor Chang is a Senior Lecturer at Leeds Beckett University since September 2012. Within four years, he completed PhD (CS, Southampton) and PGCert (Higher Education, Fellow) part-time. He helps organizations in achieving good Cloud design, deployment and services. He won a European Award on Cloud Migration in 2011, best papers in 2012 and 2015, and numerous awards since 2012. He is one of the most active practitioners and researchers in Cloud Computing, Big Data and Internet of Things in the UK. He is an Editor-in-Chief of IJOCI & OJBD journals, Editor of FGCS, founding chair of two international workshops and founding Conference Chair of IoTBD 2016 www.iotbd.org and COMPLEXIS 2016 www.complexis.org.

*Biographies (Photograph)

*Biographies (Photograph)

*Biographies (Photograph)



*Biographies (Photograph)

1. A high performance Cloud computing system architecture for remote sensing big data processing is proposed.
2. The system level implementation techniques are discussed for HPC Cloud computing.
3. The HPC Cloud computing application is discussed, remote sensing cloud computing, which enables regional environmental and disaster monitoring system as on-demand real-time services.