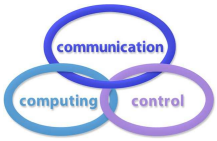
INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 16, Issue: 6, Month: December, Year: 2021
Article Number: 4236, https://doi.org/10.15837/ijccc.2021.6.4236



CCC Publications

Cloud Computing in Remote Sensing : High Performance Remote Sensing Data Processing in a Big data Environment

Y. Sabri, S. Aouad

Yassine SABRI
Laboratory of Innovation in Management and Engineering for Enterprise (LIMIE), ISGA Rabat, 27 Avenue Oqba, Agdal,Rabat,Morocco
yassine.sabri@isga.ma

Aouad Siham
RITM- ESTC/CED -ENSEM, University Hassan II Km7, El jadida Street, B.P. 8012, Oasis, Casablanca 8118
Maizate@outlook.com

Abstract

Multi-area and multi-faceted remote sensing (SAR) datasets are widely used due to the increasing demand for accurate and up-to-date information on resources and the environment for regional and global monitoring. In general, the processing of RS data involves a complex multi-step processing sequence that includes several independent processing steps depending on the type of RS application. The processing of RS data for regional disaster and environmental monitoring is recognized as computationally and data demanding.Recently, by combining cloud computing and HPC technology, we propose a method to efficiently solve these problems by searching for a large-scale RS data processing system suitable for various applications. Real-time on-demand service. The ubiquitous, elastic, and high-level transparency of the cloud computing model makes it possible to run massive RS data management and data processing monitoring dynamic environments in any cloud. via the web interface. Hilbert-based data indexing methods are used to optimally query and access RS images, RS data products, and intermediate data. The core of the cloud service provides a parallel file system of large RS data and an interface for accessing RS data from time to time to improve localization of the data. It collects data and optimizes I/O performance. Our experimental analysis demonstrated the effectiveness of our method platform

Keywords: Remote Sensing , Data integration , Cloud Computing , Big Data ;

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. The

latest-generation space-borne sensors are capable of generating continuous streams of observation data at a growing rate of several gigabytes per second ([1]) almost every hour, every day, every year. The global archived observation data probably exceed one exabyte according to the statistics of an OGC report ([2]). 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 ([3]), while, the high-resolution satellites, namely indicating 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 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 ([4]). 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 rising 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 ([5]). 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 ([6][7]). Some relevant examples are large debris flow investigation ([8], flood hazard management ([9]) and large ocean oil spills surveillance ([10][11])). Generally, these large-scale data processing problems in RS applications ([12][13][14]) with high QoS requirements 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 a cluster-based HPC (High-Performance Computing) paradigm in RS applications turns out to be the most widespread yet effective approach ([14][15][16][17][18]). Both NASA's NEX system ([14] for global processing and InforTerra's "Pixel Factory" ([19]) for massive imagery auto-processing adopt cluster-based platforms for QoS optimization.

Cloud computing ([20]) 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. The Cloud paradigm has also been widely adopted in largescale RS applications, such as the Matsu project ([21]) for cloud-based flood assessment. Currently, Clouds are rapidly joining HPC systems like clusters as variable scientific platforms ([22]). Scientists could easily customize their HPC environment and access huge computing infrastructures in the Cloud. However, compared to conventional HPC systems or even supercomputers, the Clouds are not QoS-optimized large-scale platforms. Moreover, differing from the 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 a Cloud computing paradigm with cluster-based HPC systems in an attempt 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 ([23]), 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 is also dynamically encapsulated and delivered as on-line services. Meanwhile, benefiting from a dynamic scientific workflow technique, pipsCloud offers the ability to easily customize collaborative processing workflows for large-scale RS applications.
The rest is organized as follows. Section 2 reviews some related works and discuss the challenges lying in the building and enabling a high performance cloud system for data-intensive RS data processing. Section 3 demonstrates the design and implementation of the pipsCloud from the system level point view. Then Section 4 discusses the experimental validation and analysis of the pipsCloud. Finally Section 5 concludes this paper.

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

Each solution has its pros and cons. 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, 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 advantage of a Cloud computing paradigm. As increasing numbers 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 ([14]) have also fostered the widespread applying of multi-core clusters. The first shot goes to the NEX system ([14]) for global RS applications built by NASA on a cluster platform with 16 computer in the middle of 1990s. The "Pixel Factory" system ([19]) of InforTerra employed a cluster-based HPC platform for massive RS data auto-processing, especially Ortho-rectification. These HPC platforms are also employed in the acceleration of hyperspectral imagery analysis ([24]). It is worth noting that the 10,240-CPU Columbia supercomputer equipped with InfiniBand network has 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, MPI is adopted within or across nodes, and the MPI+OpenMP hybrid paradigm ([25]) is employed for exploiting multilevels of parallelism. Recently, great efforts have been made in the incorporation of an MPI-enabled paradigm with remote sensing data processing in the large scale scenarios. Some related works with Plaza et al. presented parallel processing algorithms for hyperspectral imageries ([26]), Zhao et al. ([27]) implemented soil moisture estimation in parallel on a PC cluster, as well as MPI-enabled implementing of image mosaicking ([28], fusion ([29]) and band registration ([30]). Obviously, benefiting from the efforts and developments conducted in HPC platforms, plenty of RS applications have enhanced their computational performance in a significant way ([14]).
However, in spite of the elegant performance acceleration has achieved, it is still anything but easy for non-experts to employ the cluster-based HPC paradigm. Firstly, the programming, deploying as well as implementing of parallel RS algorithms on an MPI-enabled cluster are rather difficult and errorprone ([31]). 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 an HPC system would be rather challenging. Some emerging PGAS ([32]) typed approaches offer global but partitioned memory address spaces across nodes, like UPC ([33], Chapel ([34]) and X10 ([35]). The on-going DASH project 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 an easy and convenient way of utilizing high perfor-

mance data processing resources and applications, not to mention the on-demand customizing of computing resources and processing workflows.
The Cloud has emerged as a promising new approach for ad-hoc parallel processing, in the Big Data era[115]. 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, the cloud paradigm provides computing resources in a more easy-to-use and convenient way a service-oriented way.
The advent of the Cloud has also empowered remote sensing and relevant applications. Matsu ([21], the on-going research project of NASA for on-demand flood prediction and assessment with RS data adopts an Eucalyptus-based[37]) distributed cloud infrastructure with over 300 cores. GENESI-DEC , a project of the 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 the 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, a project to offer on-demand processing for Earth observation data was initially constructed with GRID. Subsequently, the Terradue cloud infrastructure was selected to enhance G-POD for resources provisioning ([24]).
Great efforts have been made in the employing of cloud computing in the context of remote sensing data processing, both in terms of programming models and resource provisioning.

Several optional distributed programming models[36] are prevalently employed for processing large data sets in the cloud environment, like MapReduce ([38]) and Dryad, 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 in Cloud computing, it has been greatly employed in RS data processing scenarios. Lin et al. ([36]) proposed a service integration model for GIS implemented with MapReduce, B. Li et al. ([38]) employed MapReduce for parallel ISODATA clustering. Based on the Hadoop MapReduce framework, Almeer ([37]) built an experimental 112-core high-performance cloud system at the University of Qatar for parallel RS data analysis.

Essentially, on-demand resource managing and provisioning are foremost in the 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 (OpenCloud, Eucalyptus ([37]) and OpenNebula ([38], 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, Kernel-based Virtual Machine (KVM) as well as VMWare . By management of Provisioning of Virtual Machines(VMs), hypervisors could easily scale up and down to provide a large-scale platform with a great number of VMs. Likewise, the network virtualization concept has also emerged. Yi-Man 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 an extra performance penalty. Recently, Varrette et. al. 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 the Cloud suit as a desirable HPC environment is still unclear.
The Cloud computing paradigm has empowered RS data processing and makes it more pos-

sible than ever . Unlike conventional ways of processing that are done by standalone server or software, the cloud-based RS data processing is enabled with a revolutionary promise of unlimited computing resources.

3 pipsCloud: High Performance Remote Sensing Clouds

To properly address the above issues, we propose pipsCloud, a highperformance RS data processing system for large-scale RS applications in the 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.

3.1 The system architecture of pipsCloud

As illustrated in Figure 1, pipsCloud adopts a multi-level system architecture. From bottom to top it is respectively physical resources, cloud framework, VE-RS, VS-RS, data management and cloud portal. The cloud framework manages physical resources to offer Iaas (Infrastructure as a Service) by virtue of OpenStack. Based on the cloud framework, the VE-RS offers a virtual HPC cluster environment as a service and VS-RS provides a 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 the excellent scalability, the performance penalty of virtualization is inevitable. To support the HPC cluster environment in the 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 an 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 a virtual HPC cluster with Torque task scheduler and Ganglia for monitoring on top of the cloud framework. Then varieties of RS softwares could be customized and automatically deployed on this virtual cluster with SlatStack. Moreover, a generic parallel skeleton 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 an MPI-enabled cluster.

VS-RS, a virtual processing system that is built on top of VE-RS is served as an on-line service especially for large-scale RS data processing. A VS-RS not only provides RS data products processing services, but also offers a VS-RS processing system as a service to provide processing workflow customization. By virtue of the Kepler scientific workflow engine, VS-RS could offer dynamic workflow processing and also on-demand workflow customization. The thing that is 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 the RS workflow depository are employed in VS-RS for workflow customization, interpreting and implementing. Besides, 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 is a novel and efficient way of managing and sharing RS big data on top of the cloud framework. It adopts unbounded cloud storage enabled by Swift to store these varieties of data and serve them in a RS data as a service manner. HPGFS the distributed file system for an RS data object is used for managing enormous unstructured RS

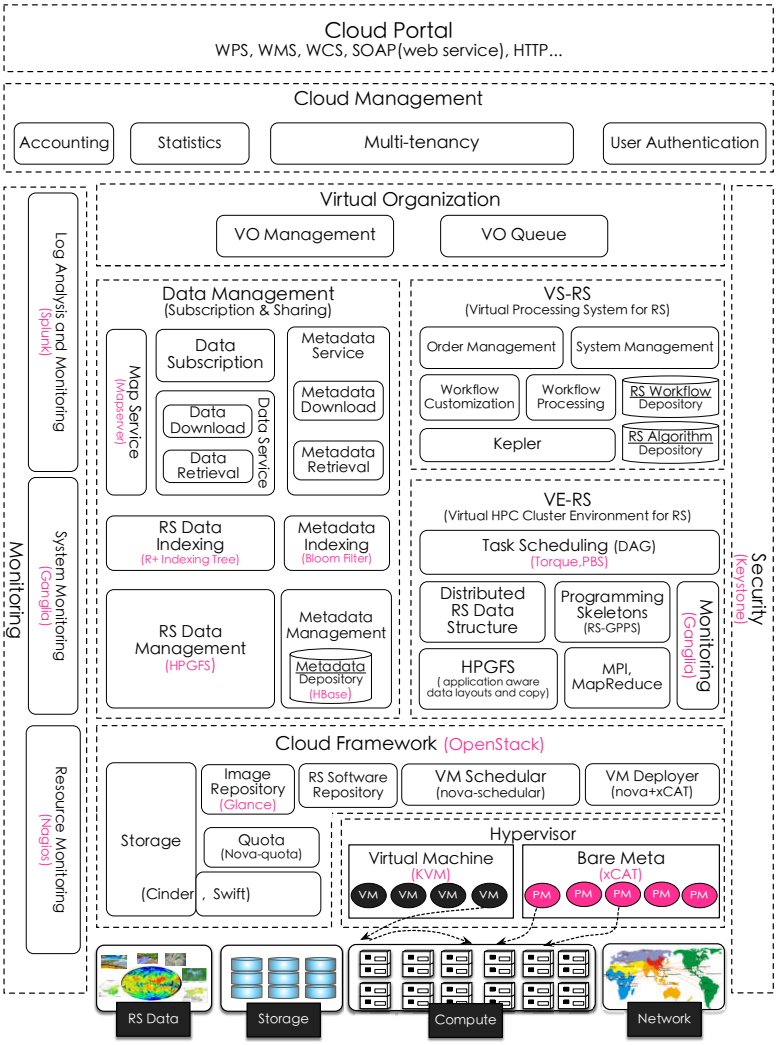


Figure 1: The system architecture of pipsCloud.

data, like RS imageries, RS data products and interim data, while the structured RS metadata are managed by a NoSQL database HBase[129]. For quick retrieval from varieties of massive RS data, a Hilbert R+ tree together with an 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 means of data sharing (Figure 5).

Cloud Management and Portal manages the whole pipsCloud platform, including system monitoring, user authentication, multi-tenancy management as well as statistics 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).

3.2 RS data management and sharing

Efficient RS data management and sharing are paramount especially in the context of large-scale RS data processing. The managing of RS big data is not only limited 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. These unstructured RS data are organized in a GeoSOT global subdivision grid, and each data block inside these data are arranged in Hilbert order and encoded with a synthetic GeoSOT-Hilbert code combining 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 GeoSOT10 [51] 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 into the Redis[52][53] in-memory database and offer hash table indexing. The most important thing that is worth mention is the easy but novel means of RS data sharing thesis-based RS data subscription and data push through virtual data catalog mounting as local.

The runtime implementing of RS data management and sharing is demonstrated in Figure 2. First, pipsCloud interprets the data requests, and checks the user authentication. Second, it conducts a quick search in the ‘’hot" data cache on the Redis in-memory database with the hash table index; if the cache hits then it returns data. Third, it searches the required data in the Hilbert R+ tree for the unique Hilbert-GeoSOT code. Fourth, it uses 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. Fifth, for a subscription request, it re-organizes these data entries to form a virtual mirror and mount it to user’s local mount point. Sixth, if an acquisition of data or metadata is needed, then it invokes GridFTP for downloading. Finally, Accounting is used for charging if the data is not free.

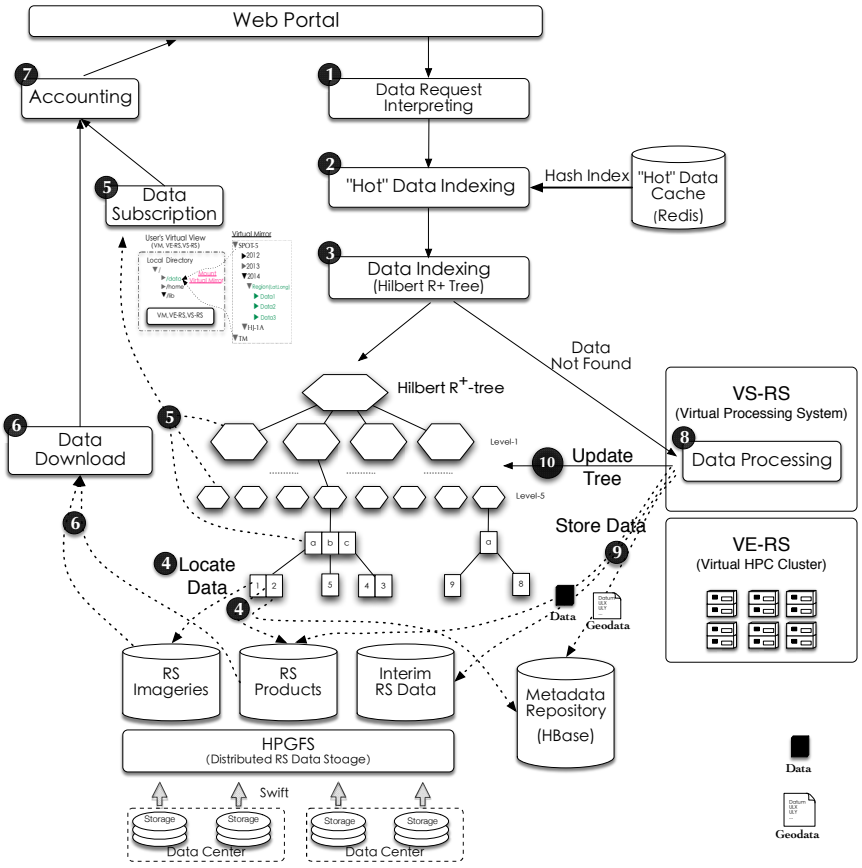


Figure 2: The runtime implementing of data management and sharing.

However, when the requested data products are not available then an 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 the interim and products repository based on

HPGFS, while the relevant metadata will be abstracted and inserted into the 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.

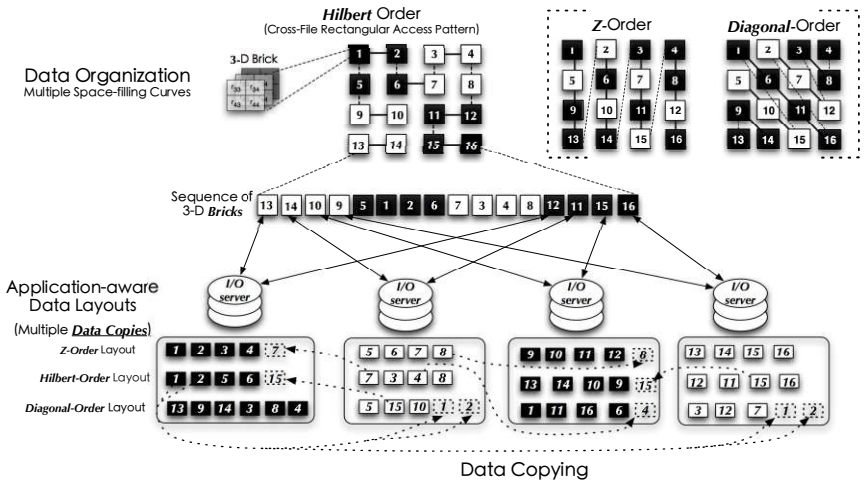


Figure 3: The application-aware data layout and data copies.

As is depicted in Figure 3, application-aware data layouts and data copy policies 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 the data striping method with fixed or variable stripe size, RS data are sliced into data bricks, which 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 patterns. Data copy organized in the Z-order curve is provided for the consecutive-lines/column access pattern, diagonal curve is for diagonal irregular data access pattern, while Hilbert curve is used for the 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 to 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 transferred to the target I/O nodes using a tree-based copying mechanism as in Figure 3 to form dynamical copies of data bricks.

Quick and efficient data retrieval among enormous distributed RS data has always been a considerably 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 the indexing tree greatly depends on the algorithms used to cluster the minimum bounding rectangles (MBRs) of the RS data on a node. Hilbert R+tree employs a Hilbert space-filling curve to arrange the data rectangles in a linear order and also group the neighboring rectangles together. To meet the requirements of realtime RS data updating, a dynamic Hilbert R+tree is normally more desirable. 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. A data scene may span several degrees of longitude in lat-long geo-

graphical coordination, like 5o for Modis data. Under this consideration, pipsCloud adopts a hybrid solution, which the combines the Hilbert R+tree with a GeoSOT global subdivision grid model. As is showed in Figure 4, the global RS data are first grouped through a deferred quad-subdivision of GeoSOT global subdivision grid model. Normally, through several levels of

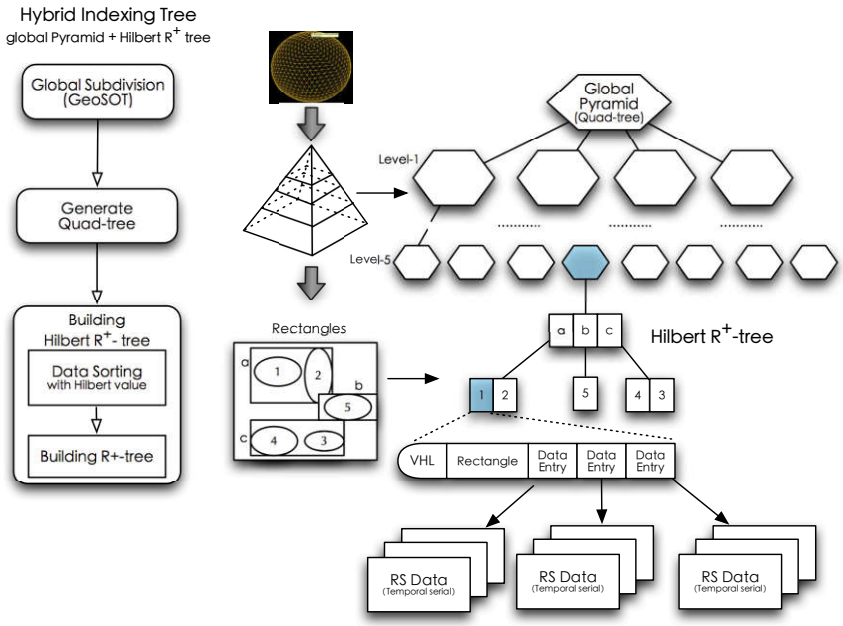


Figure 4: Optimal RS data indexing with Hilbert R+tree and global subdivision grid.

quad-subdivision (like 5 levels) a quad-tree could be constructed, while if the a RS dataset covers a really big region that much larger 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 the 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

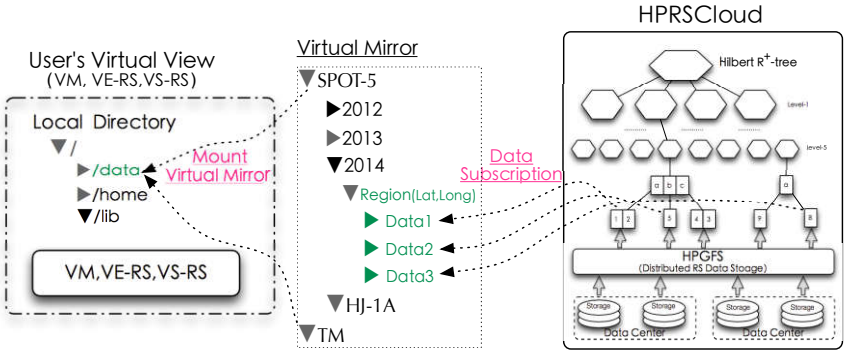


Figure 5: Data subscription and virtual catalog mirror.

this method, each RS dataset or data block would be encoded with a unique GeoSOT-Hilbert code which consists of both the 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. The non-leaf node contains LHVs (Largest Hilbert Value) and also the geographical region of the rectangles. Then by recursively sorting these new nodes by the Hilbert value of its rectangle and creating new nodes with a 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 informa-

tion (URL) of a temporal serial of RS data that is 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 the Hilbert R+tree, the GeoSOT-Hilbert code as well as MBR of the data itself 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, descend the quad-tree of the GeoSOT global grid model, and then visit 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 the URL.

3.3 VE-RS: RS-specific HPC environment as a service

(Figure 6) VE-RS offers an RS-specific cluster environment as a service on top of the OpenStack enabled cloud framework. Based on the VMs or BMs provided by the cloud framework, VE-RS could build a virtual HPC cluster through automatic deployment of cluster auto-configuration tools like AppScale. By means of SlatStack, VE-RS allows customized deployment of RS software 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 an 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 patterns, HPGFS adopts application aware data layout policy so as to exploit data locality and reduce data movements. Moreover, for easy but productive programming, VE-RS provides RS-GPPS, generic parallel skeletons for large-scale RS data processing applications on the 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 a local resource manager and scheduler in the virtual cluster, and ganglia for system monitoring.

In this study, we adopt xCAT to extend the dominant OpenStack platform for supporting bare-metal provisioning, and use the 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 shown in Figure 7, we use the KVM hypervisor for the creation and management of VMs from the pool of physical machines. When a virtual machine is requested, Nova-compute component 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-metal machines in OpenStack through xCAT, a bare-metal driver for xCAT should be integrated in the Nova-compute component ([23]). Normally, Nova-compute uses the 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, the 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 shown in Figure 7, when an HPC cluster environment for RS data processing with bare-metal machine is requested by a 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

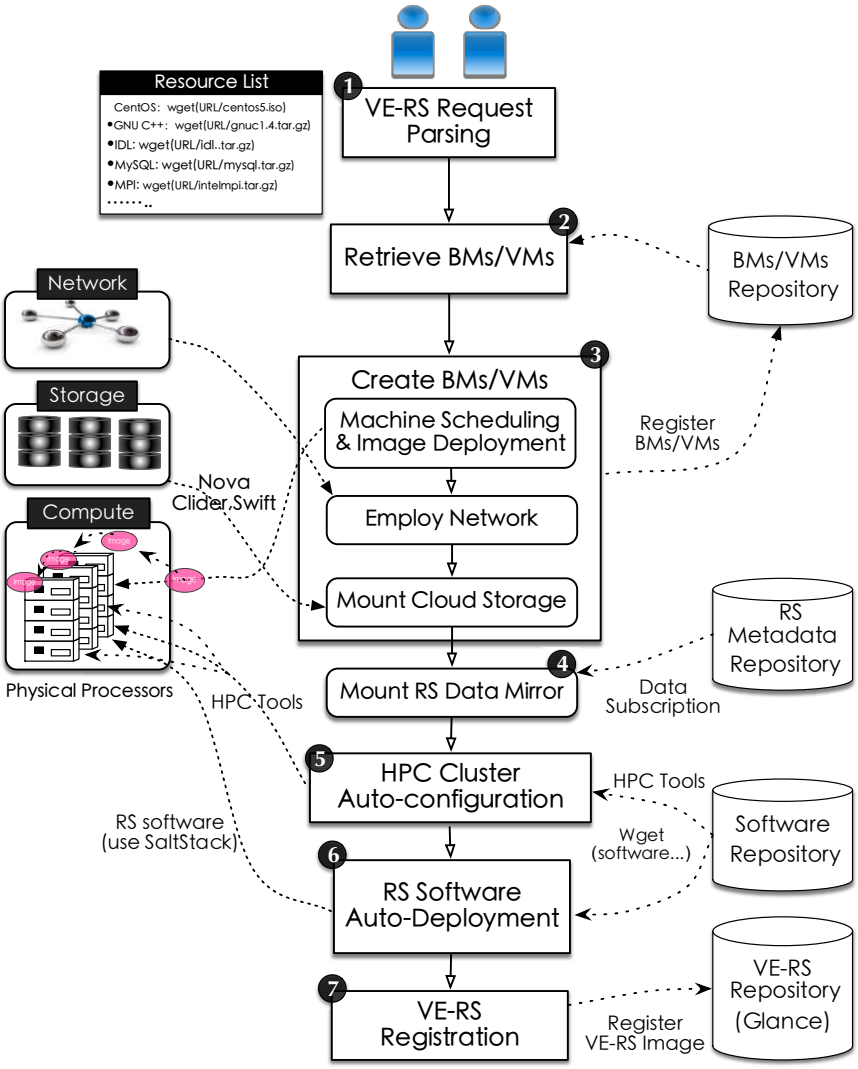


Figure 6: The generation and auto-deployment of VE-RS.

from BM/VMs database, and downloads the system images with OS and software needed for building the HPC cluster for RS. Then, xCAT activates the boot loader 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 the network and deploys it with the specified system image (OS and software). Finally, it registers the information of the bare-metal machine into the BM/VMs database. In case the baremetal machine is running, the xCAT is also responsible for managing and monitoring its status. Following this process, 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 the performance penalty.

Cluster-based HPC platforms are characterized by extreme scale and a multilevel hierarchical organization. Efficient and productive programming for these systems are 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 a template class mechanism, and working on top of MPI. Generic parallel algorithms are abstract and recurring patterns lifting from many concrete parallel programs and concealing 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 a high-level.

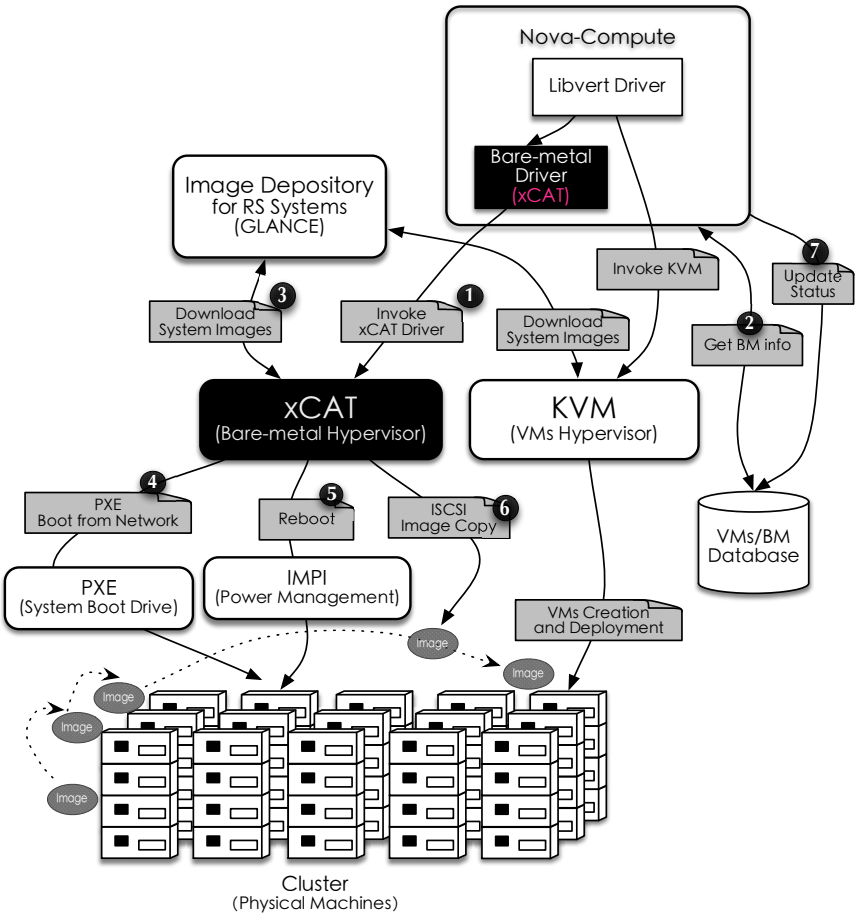


Figure 7: Bare-metal provisioning in Cloud with xCAT.

We focus on so-called class templates, which are parameterized computations patterns used to implement algorithm skeletons (Figure 8). The main contribution of RS-GPPS is that it provides both generic distributed RS data structure and generic parallel skeletons for RS algorithms.

3.4 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 the 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. 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 the local cluster in VERS or launches it to remote data centers, and finally coordinates the whole collaborative workflow processing procedure.

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

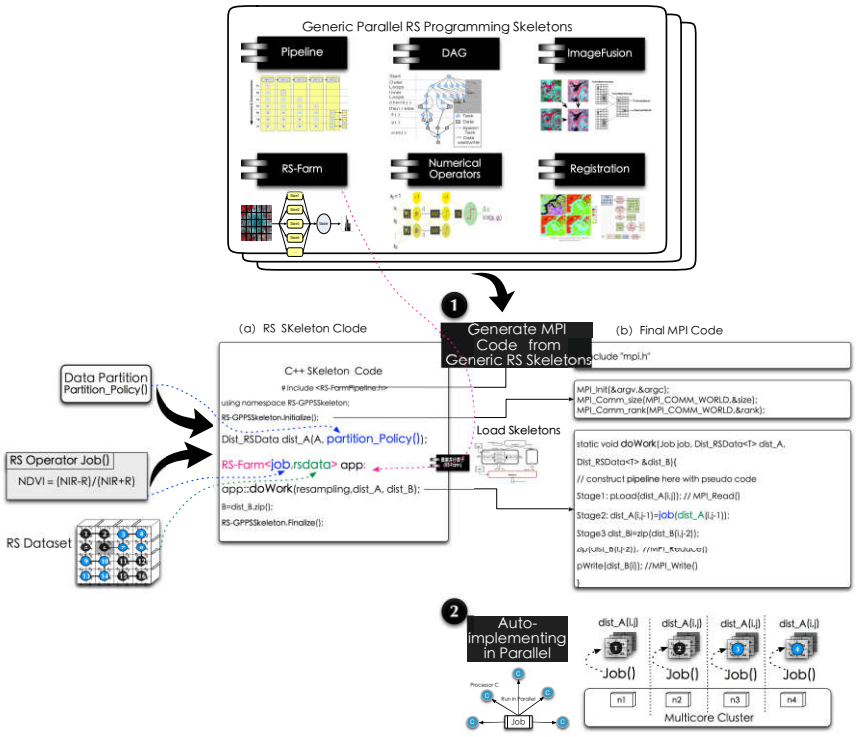


Figure 8: The skeletal parallel programming model for RS big data in pipsCloud.

- Firstly, Abstract workflow matching is responsible for interpreting the requested orders into abstract RS workflows 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 the cloud web portal, while, the abstract workflow interpreted only tells the blueprint of the data processing procedure, including the 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 resources and systems. The resources here refer 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 product knowledge represents in rules that 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 a condition statement for an RS data query. Accordingly, a list of RS imageries or auxiliary data could be drawn out from the data repository for further processing. After that goes the algorithm allocation, the candidate executable programs are deposited from the algorithm repository with the key word of the functional name of the algorithms.

- 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 the case when the processing step of workflow is a local actor, then a PBS/Torque task submission is triggered

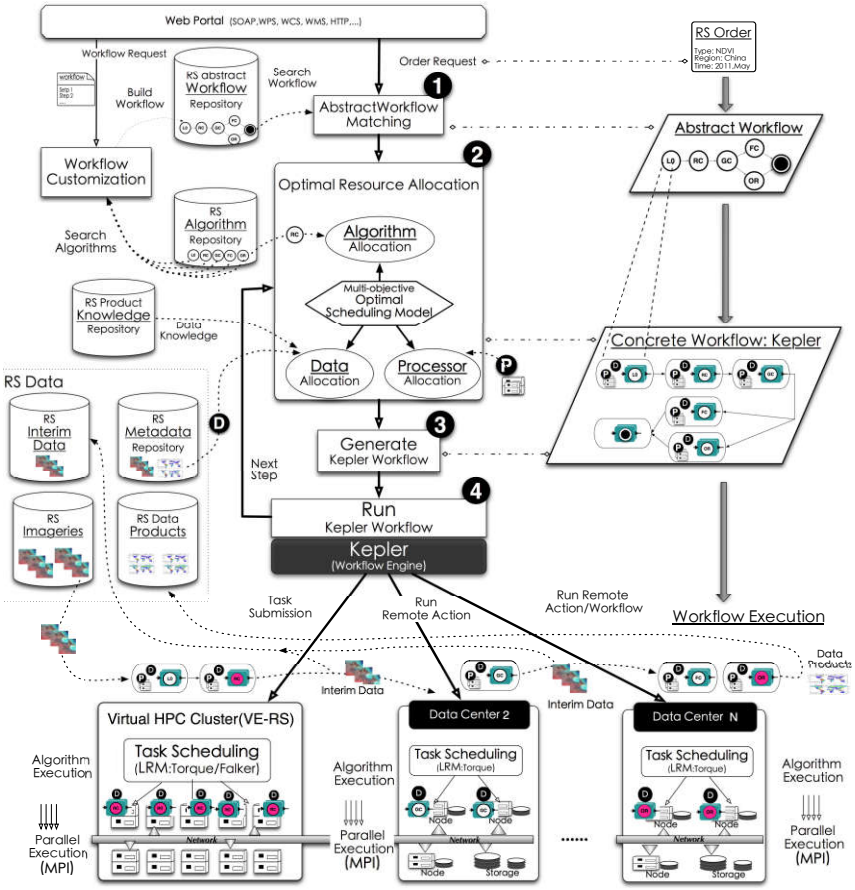


Figure 9: Dynamic and optimal workflow processing for large-scale RS applications with Kepler.

to the LRMs (Local Resource Manager). Then the LRMs launches the program of this workflow processing step onto the allocated processors (VMs or BMs) in the virtual HPC cluster environment in the Cloud and executes it in parallel, while, if the workflow step is "sub-workflow" expressed as a web service actor, Kepler would directly invokes 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.

- Finally, the workflow processing is continued recursively from optimal resource allocation, generating Kepler workflow to implementing workflow collaboratively until the end of the workflow procedure. Following this process, the entire complex processing workflow could be generated and implemented dynamically on the Kepler engine with a nearly optimal performance QoS of the whole processing procedure. When the workflow ends, the RS data products would be registered into the RS data product repository for downloading.

Consequently, with the logical control and data transferring among data centers, a distributed workflow among different data centers 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 occurs, then a re-allocation of the resource would be triggered for a re-build and re-run of the Kepler workflow.

4 Experiments and Discussion

The pipsCloud which offers a 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 can all be efficiently managed and accessed. Through the VE-

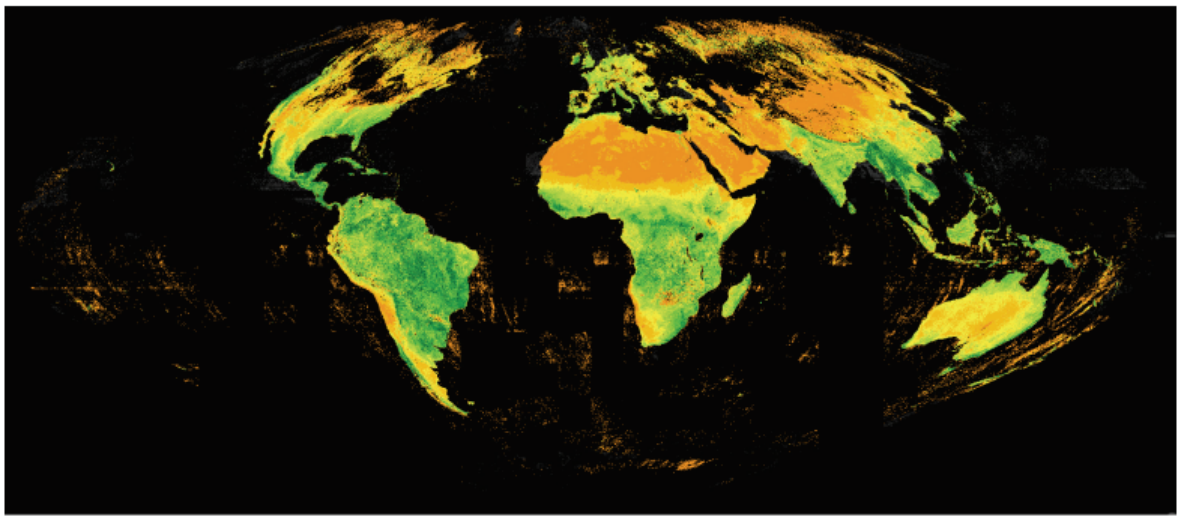
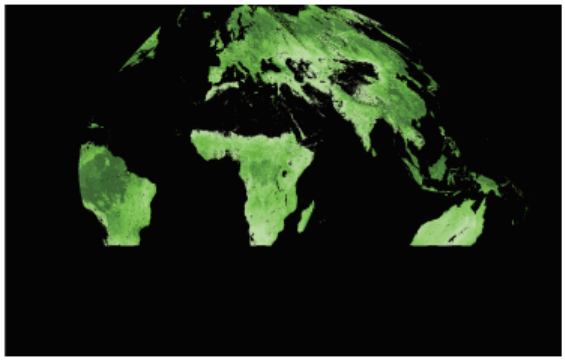
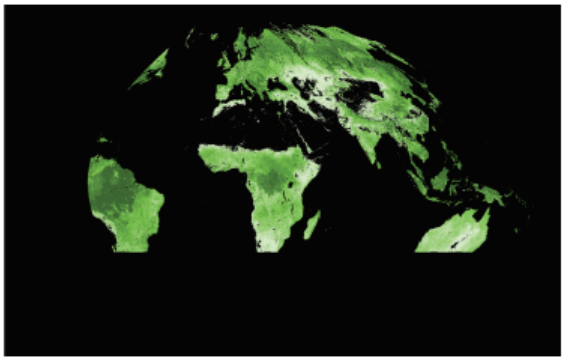


Figure 10: The 5-day synthetic global NDVI products in 2014.





(a) NPP products from day 211 to 215 (b) NPP products from day 221 to 225

Figure 11: The 5-day synthetic global NDVI products in 2014.

RS service in pipsCloud, a customized virtual HPC cluster environment is easily built and equipped with RS software, a parallel programming model and a largescale 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, a workflow engine and a 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 are implemented through collaboration a cross multiple data centers. This 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,

NSOAPS, NMSC. It offers online processing of regional to global climate change related quantitative RS data products with these multisource RS data across data centers. The RS data products generated by MDCPS include vegetation related parameters like NDVI 18 and NPP19, radiation and hydrothermal flux related parameters like AOD20 and SM21, 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 shown in Figure 10. The 5-day global synthetic NPP parameter products which were also produced with MODIS 1km data in day 211 to 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

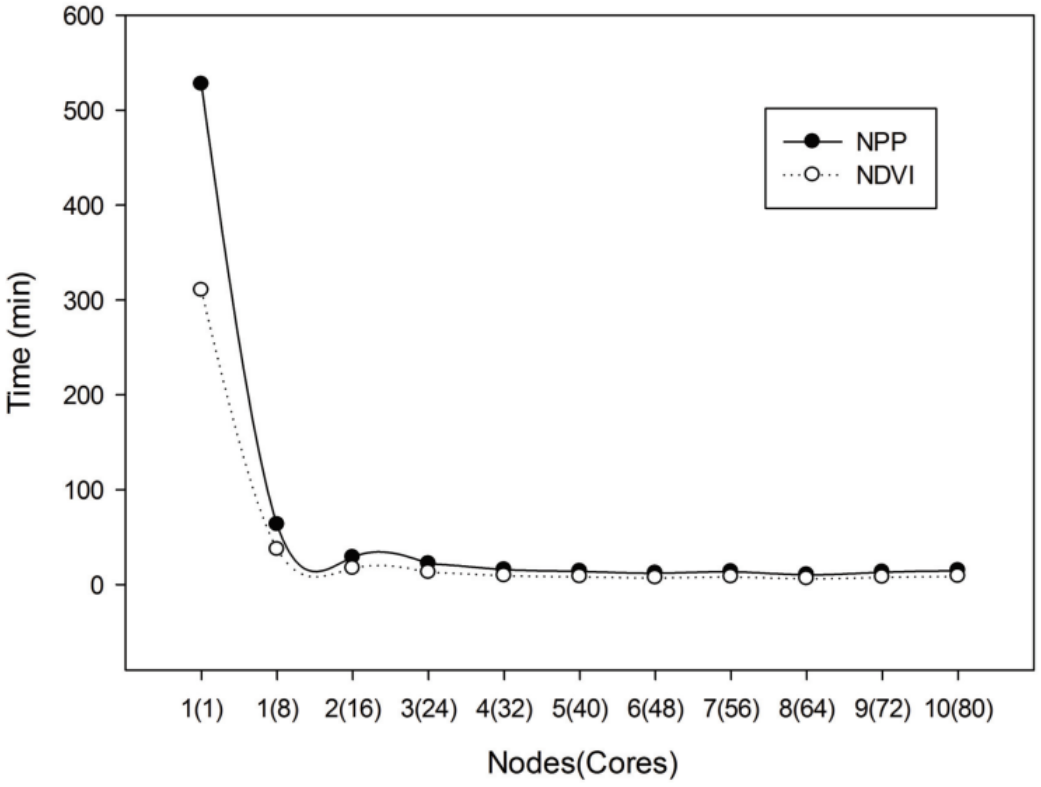


Figure 12: Run time of NPP and NDVI with scaling nodes.

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, the 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-metal provisioned processor with dual Intel (R) Quad core CPU (3.0 GHz) and 8 GB memory. The operating system was Cent OS5.0, the C++ compiler was a GNU C/C++ Compiler with optimizing level O3, and the MPI implementation was MPICH.

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 scaled to less than 4 processors (32 cores). However, the decrease rate is much slower when scaled 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 shown in sub figure (b) that the speedup metric curves of both two algorithms soar up linearly when scaling to 10 processors (80 cores). With the amount of RS data increasing from 0.5 gigabytes

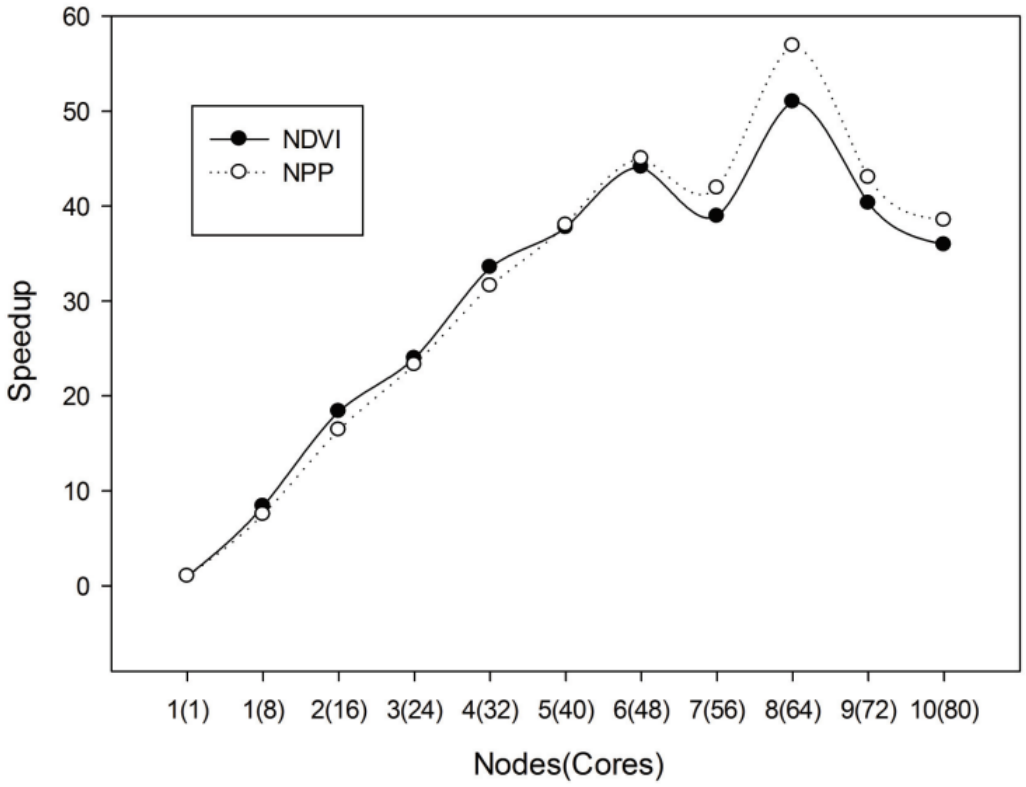


Figure 13: Speedup of NPP and NDVI with scaling nodes.

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.

5 Conclusions

The Cloud computing paradigm has been widely accepted in the IT industry with highly matured Cloud computing middleware technologies, business models, and well-cultivated ecosystems. Remote sensing is a typical information associated zone, where data management and processing play a key role. The advent of the high resolution earth observation era gave birth to the explosive growth of remote sensing (RS) data. The proliferation of data also gave 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 discussed 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 study brings a complete reference design and implementation of high performance Cloud computing for remote sensing. In 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 the cloud computing model, could manage and process the massive data, meeting the future applications' requirements.

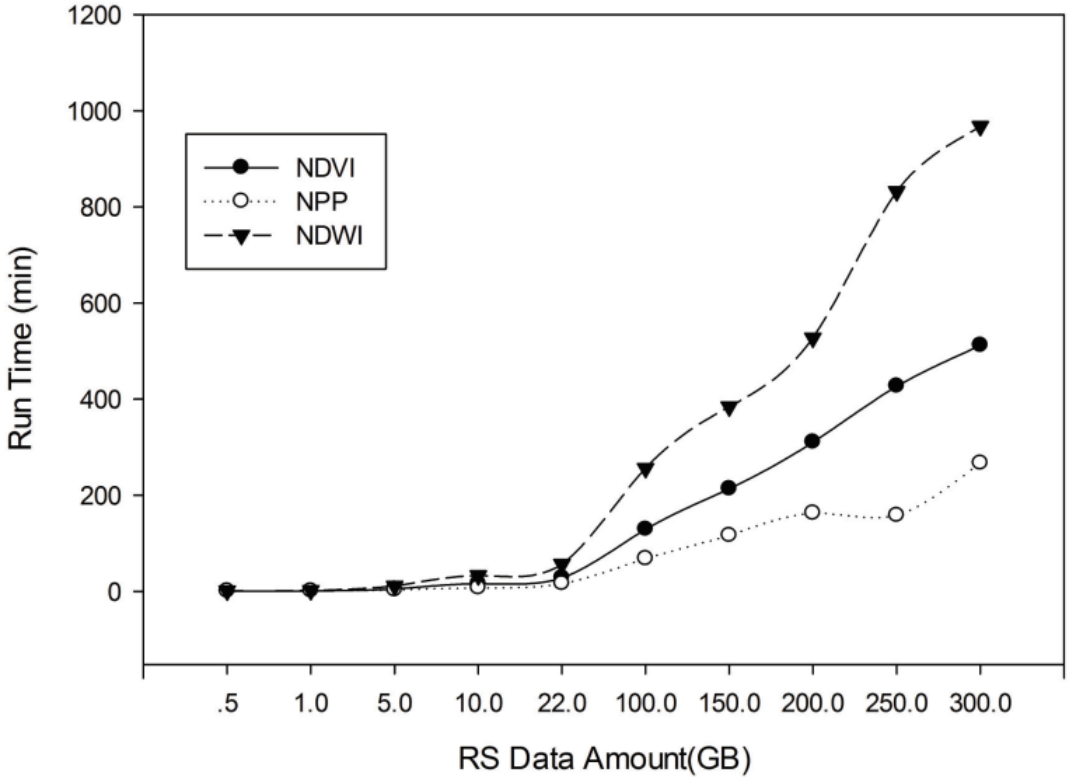


Figure 14: Run time of NPP and NDVI with increasing data amount.

References

[1] Huadong Guo, Lizhe Wang, Fang Chen, et al. Scientific big data and digital earth. Chinese Science Bulletin, 59(35):5066-5073, 2014.

[2] Rissouni Youssef, Moumen Aniss, and Chao Jamal. 2020. Machine Learning and Deep Learning in Remote Sensing and Urban Application: A Systematic Review and Meta-Analysis. In Proceedings of the 4th Edition of International Conference on Geo-IT and Water Resources 2020, Geo-IT and Water Resources 2020 (GEOIT4W-2020). Association for Computing Machinery, New York, NY, USA, Article 18, 1–5.

[3] Weijing Song, Lizhe Wang, Yang Xiang, et al. Geographic spatiotemporal big data correlation analysis via the Hilbert-Huang transformation. Journal of Computer and System Sciences, 89:130 (141), 2017.

[4] Panpan Zheng, Shuhan Yuan, and Xintao Wu. Using Dirichlet Marked Hawkes Processes for Insider Threat Detection. Digital Threats: Research and Practice 0, ja.

[5] Toshihiro Sakamoto, Nhan Van Nguyen, Akihiko Kotera, et al. Detecting temporal changes in the extent of annual ooding within the Cambodia and the Vietnamese Mekong Delta from modis time-series imagery. Remote Sensing of Environment, 109(3):295 - 313, 2007.

[6] Zhang L.F., Chen H., Sun X.J., et al. Designing spatial-temporal-spectral integrated storage structure of multi-dimensional remote sensing images. Journal of Remote Sensing, 21(1):62 73, 2017.

[7] Assis, Luiz Fernando, Gilberto Ribeiro, et al. Big data streaming for remote sensing time series analytics using map-reduce. In XVII Brazilian Symposium on GeoInformatics, 2016.

[8] Yongzhi Huang, Kaixin Chen, Yandao Huang, Lu Wang, and Kaishun Wu. 2021. Vi-liquid: unknown liquid identification with your smartphone vibration. In Proceedings of the 27th Annual

International Conference on Mobile Computing and Networking (MobiCom '21). Association for Computing Machinery, New York, NY, USA, 174–187.

[9] Chiwon Lee, Hyunjong Joo, and Soojin Jun. 2021. Social VR as the New Normal? Understanding User Interactions for the Business Arena. Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, Article 420, 1–5.

[10] Lizhe Wang, Weijing Song, and Peng Liu. Link the remote sensing big data to the image features via wavelet transformation. Cluster Computing, 19(2):793-810, Jun 2016.

[11] LizheWang, Jiabin Zhang, Peng Liu, et al. Spectral-spatial multi-featurebased deep learning for hyperspectral remote sensing image classification. Soft Computing, 21(1):213-221, Jan 2017.

[12] Weitao Chen, Xianju Li, Haixia He, et al. Assessing different feature sets' effects on land cover classification in complex surface-mined landscapes by Ziyuan-3 satellite imagery. Remote Sensing, 10(1), 2018.

[13] Myunghee Jung and Sang-Hoon Lee. 2018. Adaptive Reconstruction of Time Series Satellite Data Based on Multi-Periodic Harmonic Analysis. In Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence (MLMI2018). Association for Computing Machinery, New York, NY, USA, 8–12.

[14] Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In Kathryn Hu and James Bergstra, editors,Proceedings of the 14th Python in Science Conference, pages 130 - 136, 2015.

[15] Erik Thomsen. OLAP Solutions: Building Multidimensional Information Systems. John Wiley Sons, Inc., 2002.

[16] Z. Yijiang. The conceptual design on spatial data cube. In 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), pages 645-648, April 2012.

[17] Juglar Diaz. 2020. Spatio-temporal Conditioned Language Models. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. Association for Computing Machinery, New York, NY, USA, 2478.

[18] Tianqin Zhang, Ruimin Lyu, and Zhaolin Yuan. 2021. Visualization of imaginary stroke movement in painting and calligraphy. In ACM SIGGRAPH 2021 Art Gallery (SIGGRAPH '21). Association for Computing Machinery, New York, NY, USA, Article 14, 1–4.

[19] Junqing Fan, Jining Yan, Yan Ma, et al. Big data integration in remote sensing across a distributed metadata-based spatial infrastructure. Remote Sensing, 10(1), 2018.

[20] Jining Yan, Yan Ma, Lizhe Wang, et al. A cloud-based remote sensing data production system. Future Generation Computer Systems, 2017.

[21] Gilberto Camara, Luiz Fernando Assis, et al. Big earth observation data analytics: Matching requirements to system architectures. In Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, BigSpatial '16, pages 1-6, New York, NY, USA, 2016. ACM.

[22] SCIDB. A database management system designed for multidimensional data. http://scidb.sourceforge.net/project.html, 2017.

[23] Apache. Hadoop web site. http://hadoop.apache.org/, 2017.

[24] Chun-Yi Liu, Jagadish Kotra, Myoungsoo Jung, and Mahmut Taylan Kandemir. Centaur: A Novel Architecture for Reliable, Low-Wear, High-Density 3D NAND Storage. Proc. ACM Meas. Anal. Comput. Syst.

[25] odc. Open data cube. http://datacube-core.readthedocs.io/en/latest/index.html, 2017.

[26] OpenStreetMap. the project that creates and distributes free geographic data for the world. http://www.openstreetmap.org, 2017.

[27] UCAR. Netcdf le format and api. http://www.unidata.ucar.edu/software/netcdf/, 2017.

[28] Stephan Hoyer and Joseph J.Hamman. Xarray: N-d labeled arrays and datasets in python. Journal of Open Research Software, 5(3), 2017.

[29] Weitao Chen, Xianju Li, Haixia He, et al. A review of
ne-scale land use and land cover classi
cation in open-pit mining areas by remote sensing techniques. Remote Sensing, 10(1), 2018.

[30] Xianju Li, Weitao Chen, Xinwen Cheng, et al. Comparison and integration of feature reduction methods for land cover classi
cation with rapideye imagery. Multimedia Tools and Applications, 76(21):23041-23057, Nov 2017.

[31] Xianju Li, Gang Chen, Jingyi Liu, et al. Effects of rapideye imagery's red-edge band and vegetation indices on land cover classification in an arid region. Chinese Geographical Science, 27(5):827-835, Oct 2017.

[32] Jie Zhang, Jining Yan, Yan Ma, et al. Infrastructures and services for remote sensing data production management across multiple satellite data centers. Cluster Computing, 19(3):1-18, 2016.

[33] Ye Tian, Xiong Li, Arun Kumar Sangaiah, et al. Privacy-preserving scheme in social participatory sensing based on secure multi-party cooperation. Computer Communications, 119:167 178, 2018.

[34] Chen Chen, Xiaomin Liu, Tie Qiu, et al. Latency estimation based on traffic density for video streaming in the internet of vehicles. Computer Communications, 111:176 186, 2017.

[35] Castañeda Herrera, L. M.; Campo, W. Y.; Duque-Torres, A. (2021). Video Streaming ServiceIdentification on Software-Defined Networkin,*International Journal of Computers Communications & Control*, 16(5), 4258, 2021.

[36] Aloudat, M.; Alomari, S.; Qattous, H.; Azzeh, M.; AL-Munaizel, T. (2021). An Interactive Au-tomation for Human Biliary Tree Diagnosis Using Computer Vision,*International Journal of Com-puters Communications & Control*, 16(5), 4275, 2021.

[37] Daranda A.; Dzemyda G. (2021) Novel Machine Learning Approach for Self-Aware Prediction basedon the Contextual Reasoning, *International Journal of Computers Communications & Control*, 16(4),4345, 2021.

[38] Song Y.; Deng Y. (2021). Entropic Explanation of Power Set, *International Journal of Computers Communications & Control*, 16(4), 4413, 2021.

Journal's webpage: http://univagora.ro/jour/index.php/ijccc/

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/international-journal-computers-communications-and-control

Cite this paper as:
Sabri Y.; Aouad S. (2021). Cloud Computing in Remote Sensing : High Performance Remote Sensing Data Processing in a Big data Environment, *International Journal of Computers Communications & Control*, 16(6), 4236, 2021.
https://doi.org/10.15837/ijccc.2021.6.4236