Cloud Computing and Spatial Cyberinfrastructure

Marlon E. Pierce[1*], Geoffrey C. Fox[1,2], Yu Ma[1], Jun Wang[1],

[1]Community Grids Laboratory, Pervasive Technology Institute
Indiana University
501 North Morton Street
Bloomington, IN 47404

[2]School of Informatics
901 E. 10th St.
Bloomington, IN 47408

[*] Corresponding Author
Email: mpierce@cs.indiana.edu
Phone: 812-856-1212
Fax: 812-856-7972

Classification: Physical Sciences, Computer Sciences

Abstract: In this perspectives paper, we review the current state of Cyberinfrastructure and illustrate opportunities that we see if Cloud Computing strategies are adopted. In summary, Cloud Computing provides elastically provisioned computing, software, and service infrastructure, typically implemented on a foundation of virtual machine and virtual data storage technologies. This elasticity allows users to outsource their computing infrastructure, growing or shrinking it as necessary. Commercial investments in Cloud infrastructure make it likely that these systems will dominate large-scale computing hardware and software in the next decade. Furthermore, open source Cloud software makes it possible for universities and research laboratories to investigate and build open-architecture clouds for scientific computing and other uses. Given these general advantages, we consider the applicability of the approach to scientific computing generally and Spatial Cyberinfrastructure specifically through two case studies (flood modeling and radar image processing). We map these projects' requirements to both infrastructure and runtime capabilities typically provided by Clouds. Based on these case studies, we discuss gaps and research opportunities in Cloud Computing from the geospatial point of view. Our preliminary conclusion from this review is that Spatial Cyberinfrastructure's requirements are a good match for many common capabilities of Clouds, warranting a larger scale investigation and research by the community.

\body

Introduction

This perspectives piece summarizes our views on the next generation of Cyberinfrastructure (CI) generally and Spatial Cyberinfrastructure specifically. We base these views on experiences from number of relevant projects, including the NASA-funded QuakeSim project (1,2), the USGS-funded FloodGrid project (described here), and the NSF-funded PolarGrid project (www.polargrid.org). Our lab has developed Cyberinfrastructure software to support these distributed spatial applications, building on our general investigations of Cyberinfrastructure architectures (3). Previous applications include Geospatial Information System (GIS) Grid services based on Open Geospatial Consortium standards (4) and real-time streaming Global Positioning System processing infrastructure (5,6).

We take a broad view of the problems that Cyberinfrastructure (CI) must support. High performance computing and data storage are just two aspects; we also need to manage real-time data streams, integrate third party capabilities (such as geographic map and data providers), and build interactive user interfaces that act as Science Gateways (7). We believe the next generations of major Cyberinfrastructure deployments (such as the NSF TeraGrid (8)) need to provide a broader scope of infrastructure capabilities to their user communities. Cloud Computing approaches discussed here are good candidates for offering the infrastructure and services needed for both deep (computationally intense) science, such as is discussed by Wang in this special issue; and wide (non-traditional) usage, such the wide area GIS service networks discussed by Yang *et al.* and the GIS field worker case studies discussed by Poore, also both in this special issue. Spatial CI thus provides a subset of capabilities that spans many of the requirements of CI in general and so is a good test case for evaluating general CI architectures.

Cyberinfrastructure is the hardware, software, and networking that enables regionally, nationally, and globally scalable, sharable, distributed computing, data and information management, and collaboration. Grid computing is an important subset of Cyberinfrastructure. In the US, the NSF-funded TeraGrid and the NSF/DOE Open Science Grid (9) are examples of national-scale computing infrastructure. Internationally,

the European Grid Initiative (http://www.egi.eu/) is a prominent example, and the Open Grid Forum (http://ogf.org/) provides international community leadership and standards. An important characteristic of Grid deployments is that they provide middleware with network-accessible programming interfaces (such as Web services) that allow remote, programmatic access for executing science applications on large clusters and supercomputers, managing files and data archives, and getting information about the states of the system resources. Prominent examples of software (middleware) used to provide these capabilities include the Globus Toolkit (10), Condor (11), and gLite (glite.web.cern.ch). Higher-level capabilities can be built on these basic services. Examples include workflow composing tools (12,13), which compose basic services into higher order applications; and science gateways (7), which provide Web interfaces to services and workflows that are suitable for a broad range of users (researchers, students, and the general public). This service-oriented approach is generally compatible with, for example, the Open Geospatial Consortium's suite of service specifications, particularly the Web Feature Service and Web Map Service, as discussed by Yang *et al*. in this issue. Ideally, one may build higher-level applications out of a toolbox of third party services backed up by persistent Cyberinfrastructure; we formerly termed this the "Grid of Grids" approach (3).

The problem that we see is that there is no national scale infrastructure to provide the foundation for the comprehensive *cyber*infrastructure vision of the well-known Atkins report (14); that is, as we will elaborate, there is no "Infrastructure as a Service" in today's Cyberinfrastructure. The current flagship deployments of Cyberinfrastructure in the US are dominated by the requirements of closely coupled, high-end, high performance computing using batch queuing. This computing infrastructure is not well suited for many Spatial CI applications, which are dominated by database-driven applications, and is also ill suited for on-demand and real-time processing, such as is required in emergency response. Arguably the NSF DataNet funded projects such as DataONE (https://dataone.org/) may address the data-centric needs of Cyberinfrastructure that are crucial to much of Spatial CI, such as long-term storage and preservation of observational and experimental data and their processing pipelines, but this NSF program is in its earliest stages.

Cyberinfrastructure and Cloud Computing

There is an ongoing debate about the precise definitions of Cloud Computing and how (or if) it can be differentiated from Grid Computing. Following (15), clouds are notable for their elasticity (ability for users to scale resources up and down) and for new platform features such as distributed table data storage and the map-reduce programming model. These are not inconsistent with goals of Grid Computing. Some concepts, including service oriented architectures and workflows for scientific computing, were pioneered by Grids and are equally important for Clouds.

Academic surveys and initial investigations of clouds are available from (16,17,18), and Clouds from a Grid perspective are discussed in (19). A key distinguishing feature of Grids is the "virtual organization" (20). Grids are designed to support virtual organizations that federate multiple real, independent organizations with heterogeneous resources. In contrast, commercial Clouds are controlled by single entities (corporations such as Amazon, Google, or Microsoft), and so the virtual organization problem is not central. This may change as more resource-limited organizations (such as universities) stand up campus Clouds. In any case, Clouds expose a more user-centric view of their infrastructure: service agreements are between the user and the cloud provider, rather than between two resource providers attempting to federate themselves, as Grids attempt to do.

Ultimately, however, we believe the distinctions and similarities between Clouds, Grids, Service Oriented Architecture systems, and so on are best illustrated positively through case studies that illustrate the problems these systems attempt to solve, rather than through negative proofs or exclusionary arguments. With this in mind, we will focus on two specific aspects of the elastic capabilities of Cloud services: Infrastructure as a Service and runtime Software as a Service.

Infrastructure as a Service

At the lowest and simplest level, clouds are typically implemented using virtual machines and virtual storage devices deployed on large, centralized computing facilities and data centers. Users control the lifecycle of their virtual infrastructure through Web service-exposed programming APIs and Web user interfaces. A virtual machine is a software implementation of a computer than runs on a real computer; it can have a different operating system, software stack, and network address from its host. Clouds providers use vast collections of virtual machines to provide "Infrastructure as a Service". Through Web service and similar programming interfaces, users create and control their own virtual computing resources on remote cloud centers. A simple but powerful extension of this idea is for the virtual machines to come with software packages preinstalled and preconfigured. For example, a user may instantiate a virtual machine or cluster image that comes pre-configured with geospatial software (Web Map and Web Feature services, collections of data sets such as demographic and environmental data, and analysis software) needed for a particular investigation or to provide a particular service to a community.

Less well known than the virtual machine but at least as important for Spatial CI is the virtual block storage device. An example of this is Amazon's Elastic Block Store, which can be attached to a virtual machine to provide additional file space. These attached file systems do not need to be empty. As Amazon's public data sets illustrate (aws.amazon.com/publicdatasets/), data providers can create libraries of public and community data sets (both files and databases) that can be checked out from the Cloud by individual users. An open-architecture approach to virtual block stores is described in (44). The applicability of these services for hosting legacy (pre-cloud), distributed GIS data sets and services (see again for example Yang *et a*l in this issue) is apparent. Additionally, the major Cloud vendors all have very scalable (if non-relational) data management capabilities as part of their infrastructure. Examples include Google's BigTable, Microsoft Azure's Table Service, and Amazon's SimpleDB. These data management systems lack the full functionality of relational databases but work very well as extremely scalable spreadsheets. Google Maps and Google Earth are prominent GIS applications using BigTable, and Google Fusion Tables includes an interesting set of GIS capabilities.

Although we have focused on commercial cloud infrastructure above, it is possible to set up a cloud using Open Source software on existing server farms and clusters. Example software includes Eucalyptus (23), Nimbus (21), OpenStack, and OpenNebula. Academic cloud installations and test-beds base on these and related technologies are becoming available. The NanoHUB project at Purdue University, based on HUBzero middleware, is one of the most prominent (22). Examples of test-beds investigating Clouds include the NSF-funded FutureGrid, the NASA-funded Nebula, and the DOE-funded Magellan test-beds.

Virtualization does come with a price: virtual machines currently introduce significant communication overhead and do not support the fastest network connections such as Infiniband. Further, the virtualized networking currently used in the virtual machines in today's commercial clouds together with jitter from complex operating system functions increase synchronization/communication costs. This will effect closely coupled parallel

applications built with the Message Passing Interface (MPI), such as those commonly run on the NSF TeraGrid. This is especially serious in large scale parallel computing and leads to significant overheads in many MPI applications (52, 53). Indeed the usual (and attractive) fault tolerance model for clouds runs counter to the tight synchronization needed in most MPI applications. We review these overheads in (27). We expect that the largest, most closely coupled scientific parallel problems will continue to run on very large clusters built with advanced rather than commodity architectures (see for example the NSF funded Blue Waters supercomputer, http://www.ncsa.illinois.edu/BlueWaters/), but there remain many problems in scientific computing that are better suited for running on Cloud resources, as we discuss next. Finally, Clouds are not exclusively dependent on virtualization. Amazon's Cluster Computing Service provides access to real hardware. The FutureGrid project is also developing the infrastructure to provide real hardware as a service to support distributed computing research that is performance sensitive.

Infrastructure as a Service Case Study: Flood Grid

We first consider a somewhat typical Web service-based system built in a traditional (non-Cloud fashion) using a Service Oriented Architecture approach. We will use this to evaluate Cloud approaches by adapting it to use elastic Infrastructure as a Service approaches while retaining its overall architecture. The Flood Grid pilot study focuses on inundations of the White River at Ravenswood area in Indianapolis using the 2D hydraulic model, FaSTMECH (30), calibrated for the region. Real-time forecast data from the National Weather Service's Advanced Hydrologic Predication Service (http://water.weather.gov/ahps2/hydrograph.php?wfo=ind&gage=nori3) provide initial conditions of the simulation. The Computational Fluid Dynamics General Notation System (CGNS) (31) bridges the computation model and its environmental surface-water applications by providing a standard data format and the framework for exchanging data. Figure 1 outlines the service interactions of the system's workflow. We next review the steps in the workflow.

The river data monitoring service at the start of the workflow constantly monitors the NWS real-time forecast and starts recording both the flow gauge and the river stage data up to 6 days into the future once a pre-defined flood condition is met. During a flood study, the CGNS input process service provides initial conditions into the pre-calibrated regional model represented by a CGNS file. The updated CGNS file is then fed to the flood simulation service as its input to perform the FaSTMECH simulation, which stores computation results by updating the given CGNS file. The CGNS output process service parses the FaSTMECH simulation results and produces rectilinear flood depth grids using nearest neighbor clustering techniques. The loss calculation service overlays the generated flood grids with parcel property data and calculates percentage damages using Hazards U.S. Multi-Hazard (HAZUS-MH) (www.fema.gov/prevent/hazus) analysis tools. Finally the map tile cache service visualizes the study results in Google Maps.

The core flood simulation service wraps the FaSTMECH FORTRAN computation program using the Swarm job scheduling service framework (32). Swarm provides a set of Web Services for standard computation job management such as submission, status query, and output retrieval. The simulation service is deployed on the Gateway Hosting Service at Indiana University (33), a virtual machine-based hosting infrastructure service. Flood damage estimation and visualization services are developed with Visual Basic .NET, and deployed under Internet Information Services (IIS) by the Polis Center.

FloodGrid, as described above, is an example of a "Grid of Grids" federation of several services, rather than a Cloud. However, we will now use FloodGrid to illustrate the advantages of using Infrastructure as a Service. We map the Flood Grid infrastructure requirements to Cloud Computing infrastructure in Table 1. An important

requirement for FloodGrid's infrastructure is reliable service hosting to make sure that the services illustrated in Figure 1 are persistently available, with redundancy and load balancing. For related work, see (29).

Clouds would also be useful as providers of standard data libraries (CGNS files of hydrological models) through virtual block stores. For FloodGrid, the central piece is a validated CGNS input mesh that models a particular section of a river. Although only one such model was available for the study, one may envision a library of calibrated models for different geographic areas available for checkout from virtual block storage services. Similarly, standard GIS data sets (parcel and demographic information) can also be delivered in this fashion, coupled to a Web Feature Service that provides them. That is, one would not need to rely upon a third party Web service with its own reliability concerns. Instead, GIS data providers could provide virtual images of their data and software that can be instantiated by other developers on a Cloud as needed. Finally, we note that the system could use pre-configured virtual machines that include FaSTMECH, Swarm, and all supporting software to distribute the system to other groups wanting to run their own versions of the FloodGrid processing pipeline.

FloodGrid and Elastic Clouds: As described above, FloodGrid can support both automated and on-demand usage scenarios. Both are motivating case studies for elastic resources, since the infrastructure usage levels are very low on average but spike during flood events. The core flood simulation is the FaSTMECH computation program; it is the most computationally intensive and time-consuming part of FloodGrid project. When a flood is happening, FloodGrid will automatically start the flood simulation through its flood monitoring service for all regions of interest. Users responsible for disaster planning and emergency response will also place increased demands on the system to simulate different flood scenarios. Computing power demand thus peaks in a narrow period of the flood event. We illustrate how elastic resources may be used in this scenario.

Our simulation service is deployed as Xen virtual machines (VM) on an in-house cloud hosting testbed (see Figure 2). We estimate the computing power requirement for FaSTMECH by the simulation field size and length of flood input. Virtual machines are allocated by the service running on Xen master node accordingly. We use the Virtual Block Store (VBS) System (44), an open source version of Amazon's Elastic Block Store, to meet our storage requirements. VBS is accessed by the VMs as if it was a local disk. VBS is independent of the VM instances, so the simulation results can be accessed even after VM instances have been destroyed. In this "Infrastructure as a Service" configuration, the detail of FaSTMECH computing is invisible to FloodGird users; the workflow is always the same for flood simulation job.

To test this configuration, 10 flood scenarios with the different flood forecast parameters (based on the historical flood data) are added to the flood simulation queue. Four virtual machines with the FaSTMECH computing service are allocated automatically according to the computing time estimation. The estimated FaSTMECH running time ranges from 30 minutes to several hours. To provide rapid simulation result delivery with cloud computing resource cost in consideration, 4 virtual machines are allocated based on the estimation. 10 flood simulations are sent into the queue in random order. With 4 virtual machines, all the FaSTMECH jobs finished in 205 minutes. For comparison, 10 jobs take 739 minutes on a single virtual machine.

Software as a Service

Although one may want to use Cloud Computing to outsource infrastructure at the operating system level, it is also desirable to have higher-level tools and services available on top of the cloud infrastructure. For example, scientific computing needs to have tools that simplify running computing tasks on clouds, especially if these scale

extremely well. This is an example of what is commonly dubbed "Software as a Service". Apache Hadoop is relevant software. Hadoop is an implementation of two ideas promulgated by Google: the Google File System and MapReduce (25). Strictly speaking, Hadoop and its competitors do not need to run on elastic, virtual machine-based infrastructure, but the two are a good match (see for example Amazon's Elastic Map Reduce, aws.amazon.com/elasticmapreduce/).

MapReduce and its competitors (prominently, Microsoft's Dryad (26)) are designed to solve very large, data-file parallel information retrieval problems that arise in Internet-scale searching and indexing. MapReduce is designed to manage computing tasks in distributed environments for certain classes of parallel problems: those associated with fragmentable data sets. Although MapReduce can be applied to a wide range of problems (24), it generally is designed to support data-file parallelism; that is, we need to apply an operation or a sequence of operations to huge input files or file collections that can be split into smaller fragments on distributed file systems. The individual operations need little or no communication with each other. In contrast, traditional parallel programming, based around the Message Passing Interface (MPI), is better suited for tightly coupled applications with significant inter-process communication. The notion of file parallelism can be generalized to include memory, network, and other standard input/output mechanisms. Processing and mining sensor streams in a large sensor Web are obvious applications for stream data parallelism in Spatial CI. Although not supported by Hadoop, this is an intended feature of Dryad and has been explored by research groups (27, 28).

There are certain obvious advantages in applying MapReduce to spatial data processing. First, MapReduce scales well for appropriate problems, enabling data processing procedures to move smoothly from smaller test-beds during development and debugging to larger cluster and cloud environments. This scalable geospatial processing is crucial to spatial applications that deal with large volumes of spatial data. Second, the HDFS distributed file system makes it easy to handle a large volume of spatial data. Third, existing image processing binaries can be deployed on the cloud through Hadoop streaming. Finally, tiled spatial data system, such as Google Maps and Microsoft Bing Maps are well suited for MapReduce. We will examine these issues in our PolarGrid case study below.

The MapReduce programming model can be used in several different ways to support data-intensive spatial processing; there are three basic execution styles (38). *Map only applications:* only the mapper is present, it is suitable for one-step spatial data transformation operations, e.g. map projection and image filtering. *MapReduce applications:* both mapper and reducer are used. This can be used to build spatial index, performance querying and obtain statistical information (39,40,41). *Iterative MapReduce applications*: MapReduce steps run iteratively until certain criteria are met. It is suitable for spatial clustering and data mining algorithms, such as K-means clustering (42).

MapReduce as a programming model for clouds has received significant attention from the academic community. However, it is not a panacea, and it will not meet all the computing requirements of Spacial CI. MapReduce is a batch-processing approach for processing files registered with a fault-tolerant overlay file system (the Google File System or the Hadoop Distributed File System). In Spatial CI, problems such as large-scale image processing are the best fits for this programming style. However, relational database-centric geospatial problems are not well suited for MapReduce, and relational data processing on a cloud is an open problem. Instead of relational databases, cloud data management systems focus on highly scalable but not relational, "NoSQL" approaches that sacrifice strong consistency for scalability. In contrast, a great deal of

geospatial processing is associated with geospatial databases that extend classic relational database systems such as PostgreSQL and Oracle.

From the Spatial CI point of view, MapReduce's reliance on files as an implicit data model and its close integration with the file system are important limitations. Quadtree-modeled data (often used for indexing collections of two-dimensional images such as map tiles) can be adapted straightforwardly, but the more geospatial object-centric R-tree encoded data (such as geospatial features) are not a good match. As perhaps an intermediate step from the Spatial CI point of view, many commercial clouds are beginning to offer relational databases as services. Microsoft Azure's SQL Service (based on SQL Server) and Amazon RDS (based on MySQL) are two examples.

Research into extremely scalable relational database systems and intermediate systems between the relational and NoSQL extremes are very active areas, although to our knowledge no work specifically on the requirements of Spatial CI has taken place. HadoopDB (48), for example, is a hybrid between relational databases and MapReduce. Naturally supporting R-tree data models in MapReduce-style programming and more generally supporting relational databases in Cloud infrastructure are open academic problems (55). Google has recently published a description of its new, real-time indexing system, Percolator (49), which has replaced MapReduce as its internal mechanism for calculating search rankings. Unlike MapReduce, Percolator maintains state, avoiding the need to completely recalculate ranking indices. Related systems from Yahoo and Microsoft are described in (50) and (51), respectively. These systems are potentially interesting models for real-time geospatial processing.

Software as a Service Case Study: SAR Image Pipelines

In this case study, we examine the cloud computing requirements of a common Spatial CI problem: image processing. The sub-glacial terrain images acquired from Synthetic Aperture Radar (SAR) reveal ice sheet thickness and the details of internal ice layers over vast areas beneath the 3 KM-thick Greenland ice sheet (34, 35). Approximately 25 TB of raw SAR data are available for processing Polar Grid resources from the 2008-2009 campaigns (45). Batch processing on clusters is typically used to produce initial data products from raw data. These may require additional re-processing as the image processing algorithms are improved. Furthermore, it is desirable to create more specialized, higher-level data products, such as improving the SAR image qualities in post-processing and flight line calculations. We will examine both filtering and flight line calculations as MapReduce problems, although we note there is a larger problem to be addressed by Spatial CI pipelines.

For this test, we developed a sample Matlab application that implements the specially tailored Douglas-Peucker algorithm (47) to simplify flight line data and generate SAR images. We compiled it as a standalone executable. Again, the MCR allows us to port our compiled program to a cluster running our Hadoop installation. Flight line simplification is a data-parallel problem and well suited for MapReduce. Hadoop streaming is used to run MapReduce tasks. We had to develop a Python script to overcome the mismatch between Matlab's standard input/output mechanisms and the Hadoop Distributed File System. This script also acts as a Hadoop reader to read the flight line's native binary format; this eliminates the needs to convert the data into text format in advance.

Greenland flight path data, https://www.cresis.ku.edu/data/greenland, serve as our input. Figure 3 shows the overview of the 2007 flight lines. Flight lines are organized by the flight date; each flight line is broken into smaller sections, each section is distributed as a Matlab *mat* file, the file size is around 10M to 50M, and the total size of one day's flight line ranges from 500M to 1G. Flight path data are stored as (Latitude, Longitude)

pairs; the radar data that measured the underground ice structures are stored in the variable “Data” as a 2D array.

Data Processing: The main application is developed in Matlab, and the data is processed in the following procedure. Steps 1-4 are in the *map* stage; step 5 is the *reduce* step. 1) Simplify the flight lines using Douglas-Peucker algorithm; 2) export the simplified flight lines as Google KML; 3) generate radar images; 4) combine KML and radar images to the self-contained Google KMZ file; and 5) generate the overall simplified flight lines and radar images for the on-going visualization project (reduce stage). The sample output is shown in Figure 3. The left image shows the overall simplified flight line; the broken part in the middle is due to a missing data set. One sample radar image is shown in the right; it clearly indicates the underground ice bed exists around 2000m deep.

Testing Environment and Procedure: Software used includes Hadoop 0.20.2, Maltab 2009a, and Python 2.6. All computations were performed on IU’s Quarry cluster using *himem* (high memory) nodes obtained through the PBS scheduler. The testing is carried out by the combination of the number of the flight lines (1, 2, 3), and the number of computing nodes (3, 6, 9, 12, 15). We obtain one computing core for each node, since there is very limited multi-core support in Matlab Compiler Runtime. A Java program is used to generate the Hadoop configuration files on the fly. We then reformat the Hadoop namenode, copy data files from local files system to Hadoop HDFS. The compiled Matlab application is managed by Hadoop streaming. The processing time measured here is only the running time of Hadoop streaming jobs, rather than the whole testing process. It usually takes 2-3 minutes to start up Hadoop system on *himem* queue. The running time for individual tasks ranges from 60 to 90 seconds. Results are given in Table 2.

Three computing-node Hadoop runs are the smallest feasible size, so we calculate speedup and efficiency against these. We plot Table 2’s values in Figures 4 and 5. Figure 4 shows the overall processing time, which decreases for all flight lines with increasing nodes as expected. The efficiency and speedup measurements provide better measurements of relative performance. Figure 5 plots the speedups for increasing node sizes. All three flight line calculations collapse to the same line, indicating independence of problem size, as expected for a data parallel problem. Efficiency values from Table 2 also collapse to the same line.

MapReduce implementations such as Hadoop are subject to overheads, test-bed specific effects, and related implementation inefficiencies and artifacts that do not concern us here. We note that 60 seconds is, as a rule of thumb, a lower bound for efficient (or scalable) individual tasks run with Apache Hadoop. We observed great variability in Hadoop’s overheads (such as task scheduling), on the order of seconds. These results depend greatly on the testing environment and become negligible for longer running tasks. Other tests show parallel efficiencies over 95% are possible for larger data-parallel problems (54).

Conclusions

In this paper, we surveyed requirements that Spatial Cyberinfrastructure places on Cloud Computing. We focused on Cloud Computing's "Infrastructure as a Service" and "Software as a Service" models. We illustrated these requirements using two small projects initially developed in a pre-Cloud fashion: the Flood Grid and Polar Grid projects. Our key observation is that Clouds grant more control over the environment to developers through virtualization. This allows, for example, developers to install and control their own software without worrying about version conflicts with developers on unrelated projects. MapReduce, a common programming model for clouds, does provide

a powerful way to do some Spatial CI tasks (particularly image processing), but current implementations are a poor fit for geospatial problems that are not file-based, particularly those closely tied to geospatial database applications.

Spatial CI is an important subset of a more general CI, spanning the both "deep" (traditional, tightly coupled high performance computing) and "wide" (non-traditional or even disruptive) usage requirements. We have shown that a number of Spatial CI requirements, such as service hosting, elastic virtual clusters, and virtual data sets, map well to Cloud Computing's "Infrastructure as a Service" model. This important requirement (see for examples the service-oriented Spatial CI (Yang et al), human-centered CI (Siebar et al and Poore), and data management CI described by companion articles) is unmet by current CI deployments such as the TeraGrid. We also examined modeling and processing services with data-file parallelism (such as image processing pipelines), which are examples of common Cloud Computing "Software as a Service" models such as MapReduce. Cloud computing models still need to be applied to a broader class of Spatial CI problems, such as those discussed by Wang and by Helly et al in this special issue.

Large commercial vendors dominate Clouds, but there is a growing collection of open source software that can be used to build research clouds. A challenge for core Cyberinfrastructure research will be to investigate and document open architecture Cloud systems. Spatial CI can and should provide a wide range of important test cases.

Acknowledgments

The FloodGrid project is funded by the Federal Geographic Data Committee's National Spatial Data Infrastructure, Cooperative Agreements Program Category 2: Best Practices in Geospatial Service Oriented Architecture (SOA), Agreement #08HQAG0026. PolarGrid is funded by NSF through the award, "MRI: Acquisition of PolarGrid: Cyberinfrastructure for Polar Science", award # 0723054. We thank Thilina Gunarathne for assistance with Hadoop tests.

References

1. Atkas, M., et al. (2006), iSERVO: Implementing the International Solid Earth Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services, *Pure and Applied Geophysics*, Volume 163, Numbers 11-12, 2281-2296.
2. Donnellan, A., et al (2006) QuakeSim and the Solid Earth Research Virtual Observatory, *Pure and Applied Geophysics,* Volume 163, Numbers 11-12, 2263-2279.
3. Fox, G., Lim, S., Pallickara, S., Pierce, M. (2005) Message-based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services, *Journal of Future Generation Computer Systems*, 21(3), 401–415. (2005).
4. Aydin, G., et al., (2008) Building and applying geographical information system Grids. *Concurrency and Computation: Practice and Experience* 20(14): 1653-1695.
5. Aydin, G., Qi, Z., Pierce, M.E., Fox, G.C., and Bock, Y., Architecture, Performance, and Scalability of a Real-Time Global Positioning System Data, Grid 17 January 2007, Special issue on Computational Challenges in Geosciences in *PEPI* (Physics of the Earth and Planetary Interiors) 163: 347-359 (2007).
6. Granat, R., Aydin, A., Pierce, M.E., Qi, Z., and Bock, Y. (2007) Analysis of streaming GPS measurements of surface displacement through a web services environment, *CIDM:* 750-757 (2007).
7. Wilkins-Diehr, N., Gannon, D., Klimeck, G., Oster, S., Pamidighantam, S. (2008): TeraGrid Science Gateways and Their Impact on Science. *IEEE Computer* 41(11): 32-41.

8. Catlett, C., et al. (2004) TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications, *HPC and Grids in Action*, Ed. Lucio Grandinetti, IOS Press 'Advances in Parallel Computing' series, Amsterdam.
9. Foster, I. et al., (2004) The Grid2003 Production Grid: Principles and Practice, *HPDC*: 236-245.
10. Foster, I. (2006) Globus Toolkit Version 4: Software for Service-Oriented Systems. *J. Comput. Sci. Technol*. 21(4): 513-520.
11. Thain, D., Tannenbaum, T., Livny, M. (2005) Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience* 17(2-4): 323-356.
12. Gil, Y., et al (2007) Examining the Challenges of Scientific Workflows. *IEEE Computer* 40(12): 24-32.
13. Fox, G., Gannon, D. (2006) Special Issue: Workflow in Grid Systems. *Concurrency and Computation: Practice and Experience* 18(10): 1009-1019.
14. Atkins DE, et al. (2003) *Revolutionizing Science and Engineering through Cyber-infrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*, National Science Foundation Publication NSF0728 (National Science Foundation, Washington, DC), 84 pp.
15. Fox, Geoffrey (2010) Clouds and Map Reduce for Scientific Applications. Technical Report. Available from http://grids.ucs.indiana.edu/ptliupages/publications/CloudsandMR.pdf
16. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andy Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia: A view of cloud computing. Commun. ACM 53(4): 50-58 (2010)
17. Youseff, L.; Butrico, M.; Da Silva, D (2008) Toward a Unified Ontology of Cloud Computing. Page(s): 1-10 Digital Object Identifier 10.1109/GCE.2008.4738443.
18. Jha, S., Merzky, A., Fox, G (2009) Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurrency and Computation: Practice and Experience* 21(8): 1087-1108.
19. Foster, I. T., Zhao, Y., Raicu, I., Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared CoRR abs/0901.0131: (2009).
20. Ian T. Foster, Carl Kesselman, Steven Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. IJHPCA 15(3): 200-222 (2001)
21. Foster, I., et al. (2006) Virtual Clusters for Grid Communities, *CCGRID*: 513-520.
22. Klimeck, G., et al (2008), nanoHUB.org: Advancing Education and Research in Nanotechnology, *IEEE Computers in Engineering and Science (CISE),* Vol. 10, 17-23 (2008).
23. Nurmi, D., et al (2008) The Eucalyptus Open-source Cloud-computing System, in *Proceedings of Cloud Computing and Its Applications*, Chicago, IL (October 2008).
24. Chu C-T, et al (2006). Olukotun, Map-Reduce for Machine Learning on Multicore, *NIPS*: 281-288.
25. Dean, J., Ghemawat, S. (2008) MapReduce, Simplified Data Processing on Large Clusters. Commun, *ACM* 51(1): 107-113.
26. Isard, M., Budiu, M., Yu Y., Birrell, A., Fetterly, D. (2007) Dryad, Distributed Data-Parallel Programs from Sequential Building Blocks, *EuroSys*: 59-72.
27. Ekanayake, J.; Pallickara, S.; Fox, G. (2008) MapReduce for Data Intensive Scientific Analyses*. IEEE Fourth International Conference on eScience '08* 7-12 Dec. 2008 Page(s):277 - 284 Digital Object Identifier 10.1109/eScience.2008.59
28. Pallickara, S.; Ekanayake, J.; Fox, G. (2008) An Overview of the Granules Runtime for Cloud Computing. *IEEE Fourth International Conference on eScience '08*, 7-12 Dec. 2008 Page(s): 412 - 413 Digital Object Identifier 10.1109/eScience.2008.101.

29. Nadine Alameh: Chaining Geographic Information Web Services. IEEE Internet Computing 7(5): 22-29 (2003)
30. Nelson, J.M., Bennett, J.P., and Wiele, S.M., 2003, Flow and Sediment Transport Modeling, Chapter 18, p.539-576. In: *Tools in Geomorphology,* eds. M. Kondolph and H. Piegay, Wiley and Sons, Chichester, 688 pp.
31. CGNS: Legensky, S.M., Edwards, D.E., Bush, R.H., Poirier, D.M.A., Rumsey, C.L., Cosner, R.R., and Towne, C.E. (2002), CFD General Notation System (CGNS)—Status and future directions: *American Institute of Aeronautics and Astronautics* , 2002-0752.
32. Pallickara, S.L.; Pierce, M. (2008) SWARM: Scheduling Large-Scale Jobs over the Loosely-Coupled HPC Clusters. *IEEE Fourth International Conference on eScience* '08. 7-12 Dec. 2008 Page(s):285 - 292 Digital Object Identifier 10.1109/eScience.2008.64.
33. Lowe, J. M., et al (2009) Gateway Hosting at Indiana University. *Online Proceedings of TeraGrid 2009* June 22-25, Arlington, VA. Available from http://archive.teragrid.org/tg09/files/tg09_submission_47.pdf
34. Paden, J., et al (2010), Ice-Sheet Bed 3-D Tomography, *Journal of Glaciology,* Vol. 56, No. 195.
35. Allen, C., and J. Paden (2007), Synthetic-Aperture Radar Images Polar Ice-Sheet Bed, *SPIE Newsroom* [DOI: 10.1117/2.1200706.0780].
36. Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox Twister: A Runtime for Iterative MapReduce Proceedings of the First International Workshop on MapReduce and its Applications of ACM HPDC 2010 conference, Chicago, Illinois, June 20-25, 2010.
37. Judy Qiu, Thilina Gunarathne, Jaliya Ekanayake, Jong Youl Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Yang Ryan, Saliya Ekanayake, Tak-Lon Wu, Scott Beason, Adam Hughes, Geoffrey Fox Hybrid Cloud and Cluster Computing Paradigms for Life Science Applications Technical Report April 17 2010 submitted to the 11th Annual Bioinformatics Open Source Conference BOSC 2010.
38. J. Ekanayake, X. Qiu, T. Gunarathne, S. Beason, and G. Fox, (2010). "High Performance Parallel Computing with Cloud and Cloud Technologies," *Cloud Computing and Software Services: Theory and Techniques*, CRC Press (Taylor and Francis), pp. 1-39.
39. A. Cary, Z. Sun, V. Hristidis, and N. Rishe, (2009) "Experiences on Processing Spatial Data with Using MapReduce in Practice," Lecture Notes In Computer Science: Proceedings of the 21st International Conference on Scientific and Statistical Database Management, vol. 5566, pp. 302-319.
40. S.W. Schlosser, M.P. Ryan, R. Taborda, J. Lopez, D.R. O'Hallaron, and J. Bielak, (2008). "Materialized community ground models for large-scale earthquake simulation," 2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1-12.
41. S. Zhang, J. Han, Z. Liu, K. Wang, and Z. Xu, (2009). "SJMR: Parallelizing spatial join with MapReduce on clusters," *2009 IEEE International Conference on Cluster Computing and Workshops*, pp. 1-8.
42. J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox, (2010). "Twister : A Runtime for Iterative MapReduce," *The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) - HPDC2010*, pp. 1-8.
43. J. Dean and S. Ghemawat, (2004). "MapReduce: Simplified data processing on large clusters," OSDI'04: Sixth Symposium on Operating System Design and Implementation, pp. 1-13.

44. X. Gao, M. Lowe, Y. Ma, and M. Pierce, 2009. "Supporting cloud computing with the virtual block store system," 2009 5th IEEE International Conference on E-Science Workshops, pp. 71-78.
45. L. Hayden, J.H. Powell, and E. Akers, (2009). "Establishing field and base camp servers for remote sensing of ice sheets in ilulissat, Greenland," *2009 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, pp. 571-573.
46. MathWorks, (2010). MATLAB Distributed Computing Server 5.0, http://www.mathworks.com/products/distriben/
47. D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol. 10, 1973, pp. 112-122.
48. Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel J. Abadi, Alexander Rasin, Avi Silberschatz: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. PVLDB 2(1): 922-933 (2009)
49. Daniel Peng and Frank Dabek, "Large-scale Incremental Processing Using Distributed Transactions and Notifications", 9th USENIX Symposium on Operating Systems Design and Implementation, Vancouver, October 4-6, 2010.
50. D. Logothetis, C. Olston, B. Reed, K. C. Webb and K. Yocum. Stateful Bulk Processing for Incremental Algorithms. ACM Symposium on Cloud Computing (SOCC), Indianapolis, Indiana, June 2010.
51. Pradeep Kumar Gunda, Lenin Ravindranath, Chandramohan A. Thekkath, Yuan Yu, and Li Zhuang, "Nectar: Automatic Management of Data and Computation in Data Centers", Microsoft Research Technical Report, MSR-TR-2010-55, May 2010
52. Edward Walker, Benchmarking Amazon EC2 for High Performance Scientific Computing, USENIX ;login, vol. 33(5), Oct 2008 http://www.usenix.org/publications/login/2008-10/openpdfs/walker.pdf
53. Jaliya Ekanayake, Xiaohong Qiu, Thilina Gunarathne, Scott Beason, Geoffrey Fox High Performance Parallel Computing with Clouds and Cloud Technologies to appear as a book chapter to Cloud Computing and Software Services: Theory and Techniques, CRC Press (Taylor and Francis), ISBN-10: 1439803153. http://grids.ucs.indiana.edu/ptliupages/publications/cloud_handbook_final-with-diagrams.pdf
54. Thilina Gunarathne, Tak-Lon Wu, Jong Youl Choi, Seung-Hee Bae, Judy Qiu, “Cloud Computing Paradigms for Pleasingly Parallel Biomedical Applications”. To appear in Proceedings of CloudCom 2010, Indianapolis, IN Nov 30-Dec 3.
55. Afsin Akdogan, Ugur Demiryurek, Farnoush Banaei-Kashani and Cyrus Shahabi, “Voronoi-based Geospatial Query Processing with MapReduce”. To appear in Proceedings of CloudCom 2010, Indianapolis, IN Nov 30-Dec 3.

Figure Legends

Figure 1 Flood Grid Service Stack and Workflow. Each component (gray box) is a network accessible service with well-defined inputs and outputs expressed using the Web Service Description Language.

Figure 2 Virtualization is used to elastically increase the Flood Grid simulation service to handle greater demand during flood events.

Figure 3 Sample Output of 2009-9-17 Flight Line

Figure 4 Processing time versus number of computing nodes for different flight lines.

Figure 5 Speedup versus number of nodes for different flight lines

Table Legends

Table 1 Mapping Flood Grid infrastructure requirements to Cloud Computing.

Table 2 Haoop Test Results. P=Processing time (s). Relative Speedup S=T(3 nodes)/T (x nodes). E=Efficiency per node, S(nodes)*3/nodes

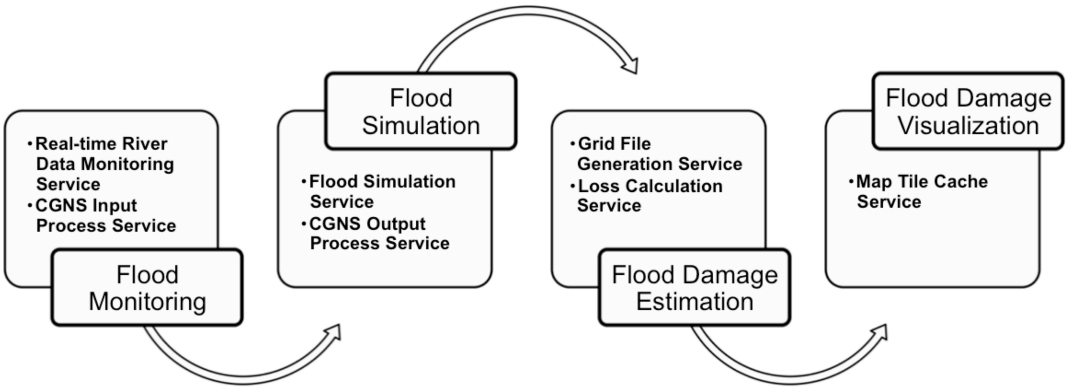

Figure 1 Flood Grid Service Stack and Workflow. Each component (gray box) is a network accessible service with well-defined inputs and outputs expressed using the Web Service Description Language.

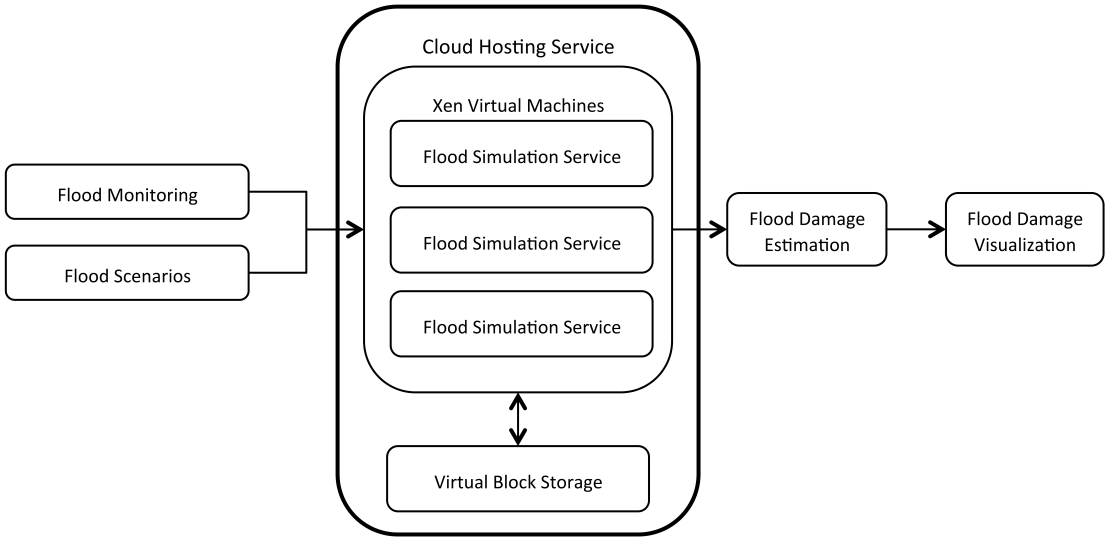

Figure 2 Virtualization is used to elastically increase the Flood Grid simulation service to handle greater demand during flood events.

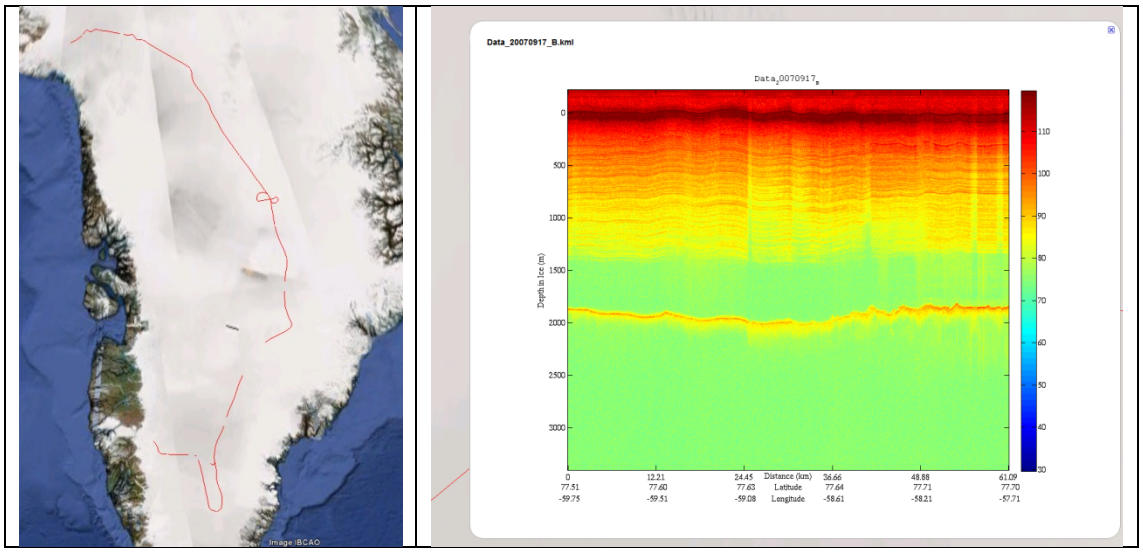

Figure 3 Sample Output of 2009-9-17 Flight Line

Table 1 Mapping Flood Grid infrastructure requirements to Cloud Computing.

Flood Grid Requirement	Cloud Computing Capability
Web Service hosting	Virtual machine infrastructure
CGNS mesh model data	Virtual block storage
GIS data (WFS parcel information, HAZUS-MH)	Virtual block storage
FaSTMECH Hosting	Virtual machine infrastructure; Map-Reduce style computation management (optional)

Table 2 Haoop Test Results. P=Processing time (s). Relative Speedup S=T(3 nodes)/T (x nodes). E=Efficiency per node, S(nodes)*3/nodes

		1 Flight Line			2 Flight Lines			3 Flight Lines		
		P	S	E	P	S	E	P	S	E
Computing nodes	3	1221	100.0%	1.00	2357	100.0%	1.00	3431	100.0%	1.00
	6	644	189.6%	0.95	1233	191.2%	0.96	1837	186.8%	0.93
	9	469	260.3%	0.87	874	269.7%	0.90	1260	272.3%	0.91
	12	364	335.4%	0.84	702	335.8%	0.84	1041	329.6%	0.82
	15	303	403.0%	0.81	590	399.5%	0.80	847	405.1%	0.81

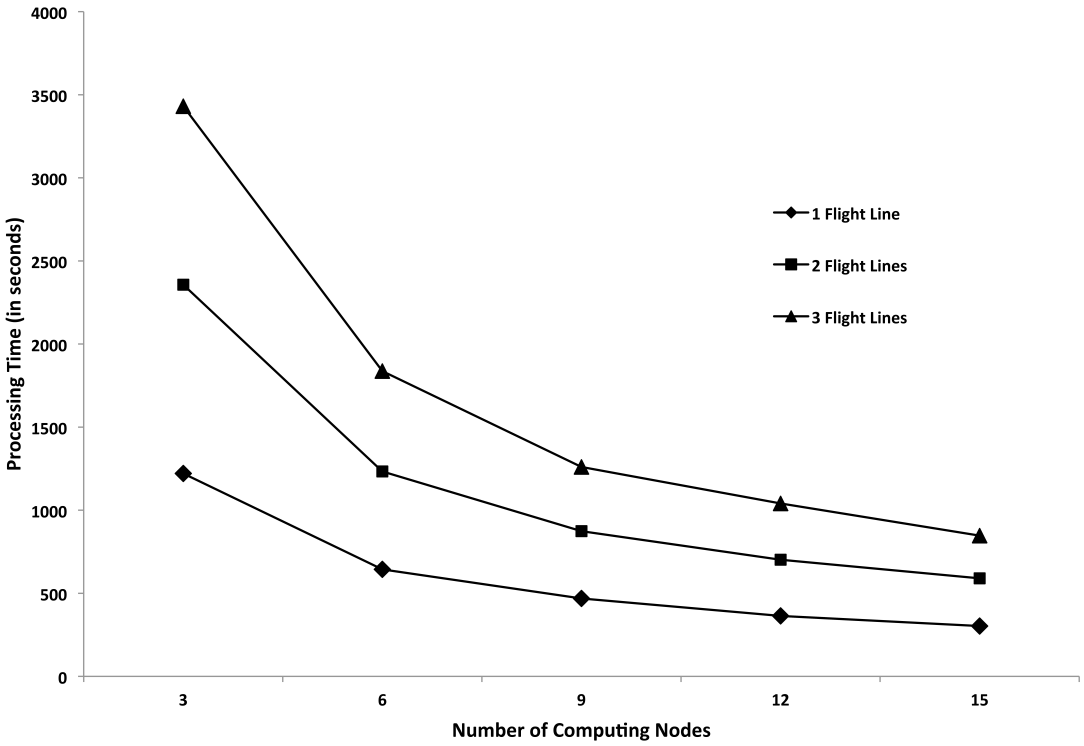

Figure 4 Processing time versus number of computing nodes for different flight lines.

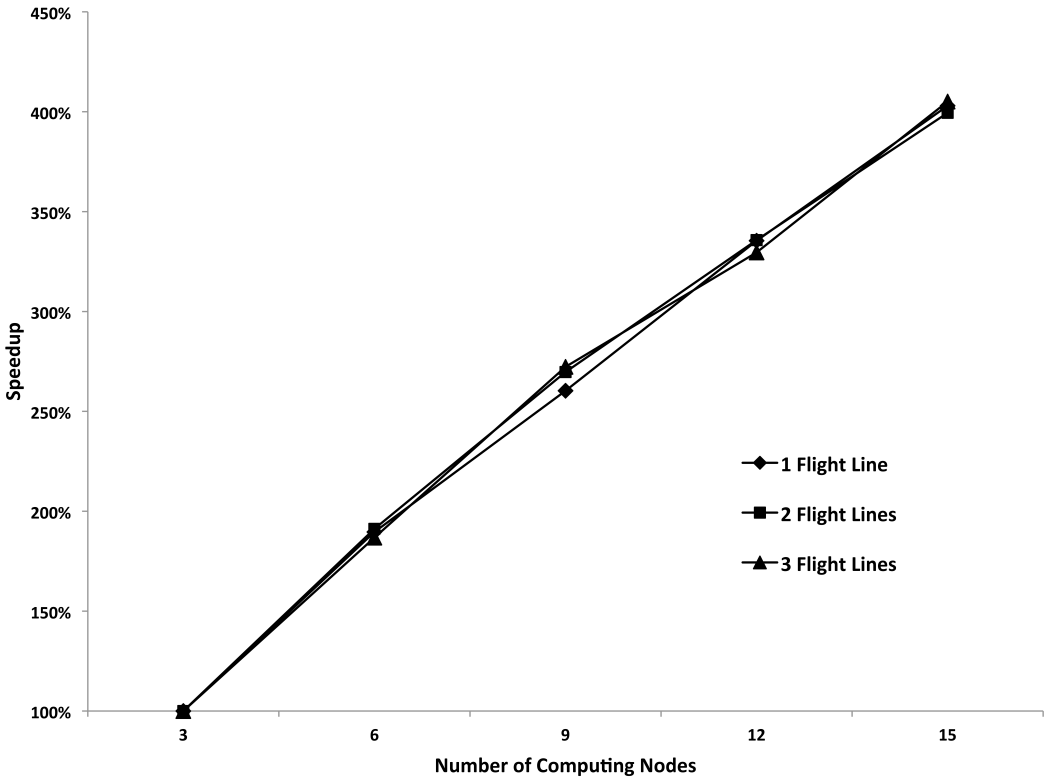

Figure 5 Speedup versus number of nodes for different flight lines.