Whither 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

Manuscript information: 10 text pages, 4 figures, and 3 tables (approximately 3 manuscript pages)

Suggested Reviewers:
Sharon Kedar, NASA JPL, sharon.kedar@jpl.nasa.gov
Karen Moe, NASA GSFC, karen.moe@nasa.gov
Dennis Gannon, Microsoft, Dennis.Gannon@microsoft.com
Jim Myers, NCSA, jimmyers@ncsa.uiuc.edu
Shantenu Jha, LSU, sjha@cct.lsu.edu

Abstract: Cyberinfrastructure has closely tracked commercial best practices for over a decade. However, we believe there is still much to learn about correct strategies for building distributed systems for collaborating scientists and related communities. In this perspectives paper, we review the current state of Cyberinfrastructure and opportunities that we see if Cloud Computing strategies are adopted. In summary, Cloud Computing is the use of Web Services to control the life cycle of virtual machines and virtual data stores to create a flexible, user-controlled infrastructure. Huge commercial investments into Cloud infrastructure ensure that these systems will dominate large-scale computing hardware and software in the next decade. Furthermore, open source Cloud software is making it possible for organizations such as universities and research laboratories to build open-architecture clouds for scientific computing and other uses. We illustrate the applicability and potential advantages of Cloud Computing to Cyberinfrastructure through two sample projects.

Introduction

This perspectives piece summarizes our views on the next generation of Cyberinfrastructure generally and Spatial Cyberinfrastructure specifically. We have been involved in a 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). 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 very broad view of the problems that Cyberinfrastructure (CI) must support. Computing and data storage are just two aspects; we also need to manage real-time data streams, integrate third party capabilities (such as map and data providers), and build interactive user interfaces that act as Science Gateways (7). As we discuss in this paper, we believe the current CI deployments need to provide a broader scope of capabilities to their user community. We believe that Cloud Computing approaches (discussed below) can offer this infrastructure.

Other contributors to this special issue have offered definitions of Cyberinfrastructure and detailed examples, so we will only briefly summarize. Cyberinfrastructure (CI) is generally the hardware, software, and networking that enables regionally, nationally, and globally scalable distributed computing and collaboration. Such systems are also called Grids. In the US, the NSF-funded TeraGrid (8) and the NSF/DOE Open Science Grid (9) are examples of national-scale infrastructure. The important characteristic of these centers is that they provide Web Service interfaces that allow remote, programmatic access for running science applications on large clusters and supercomputers, managing files and archives, and getting information about the state of the system. These interfaces are typically built as Web services. Prominent examples of software used to provide these services 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 user interfaces to services 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. Ideally, one may build higher-level applications out of a toolbox of third party services backed up by persistent

Cyberinfrastructure; we have termed this the “Grid of Grids” approach (3). We take here a heterogeneous view of Cyberinfrastructure: it could include GIS services provided by state and local governments as well as Globus services on the TeraGrid.

Scientific research enabled by this infrastructure is called E-Science. We use E-Science broadly to include Economics, Social Sciences, Mathematics, Computer Science, and Engineering fields as well as Physics, Astronomy, Chemistry, and the Life Sciences. This still is actually a limited view, as many formerly non-computational academic fields (the arts and humanities, for example) have an increasing demand for digital data management, processing, and analysis, so we also have E-Humanities, and generally E-MoreOrLessAnything. Also, of course, synchronous and asynchronous collaboration are essential to any intellectual endeavor. An important desirable outcome of CI efforts is greater openness of data and reproducibility of science. This is the vision set out by the well-known Atkins report (14).

Unfortunately this comprehensive CI vision has not yet been realized. The current flagship deployments of Cyberinfrastructure in the US are dominated by the requirements of traditional high performance computing users. Arguably the NSF DataNet program will address the data-centric needs of Cyberinfrastructure, such as long-term storage and preservation of observational and experimental data and their pipelines, but this program is in its infancy. In this paper, we argue for the adoption of Cloud Computing approaches to CI, which we believe will offer a broader approach to infrastructure. We note that Cloud Computing-like infrastructure is of particular interest to Spatial CI applications, which provides important use cases that help clarify what capabilities an end-to-end CI deployment should provide. As discussed by Wright and Wang in their introduction to this special issue, spatial CI can be both computationally intensive and data-rich. Its applications are particularly visible to the general public through Web interfaces (gateways and services) and have a broad applicability to many domains (disaster and emergency management, environmental planning, social sciences, etc). We illustrate these requirements through two small projects, Flood Grid and the Polar Grid Portal. First, however, we will review Cloud Computing.

Cyberinfrastructure and Cloud Computing

Cloud Computing, as a marketing term, is usually left poorly defined. However, because of its potential value to research computing infrastructure, academic surveys and initial investigations exist (see for example 15, 16), which the reader should consult for more information. We will focus on two specific aspects: Cloud Computing to provide infrastructure and Cloud Computing to provide runtime management.

Infrastructure: At the lowest and simplest level, clouds may be defined as Web services that control the life cycles of virtual machines and virtual storage. The very well known Amazon and Microsoft Azure cloud systems fall in this category. Xen (www.xen.org) is a popular technology for virtualizing server farms and data centers based on Linux; Microsoft similarly has Hyper-V for Windows Server 2008-based systems. Through Web services and virtualization, users can create and control their own computing resources. These may be bare-minimum installations but more usefully the virtual machines can come with software packages preconfigured. For example, one may imagine checking out a virtual machine or cluster that comes pre-configured with geospatial software needed for a particular problem.

Less well known than the virtual machine but at least as important is the virtual block storage device. The best 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 don't need to be empty. As Amazon's public data sets illustrate (aws.amazon.com/publicdatasets/), we can create libraries of public and community data sets (files or databases) that can be checked out from the Cloud by individual users. Additionally, the major Cloud vendors all have very scalable but flat database technologies as part of their infrastructure. Examples include Google's BigTable, Microsoft Azure's Table Service, and Amazon's SimpleDB. These lack the full functionality of relational databases but work very well as Cloud spreadsheets.

Although we have focused on commercial cloud infrastructure, it is possible to set up a cloud using Open Source software on existing server farms and clusters. Example software includes Eucalyptus (19), Nimbus (17), and OpenNebula (www.opennebula.org). Production academic cloud installations based on these and related technologies are becoming available. The NanoHUB project at Purdue University is one of the most prominent (18).

Virtualization does come with a price: virtual machines (particularly those that use hypervisor-like approaches such as Xen) introduce significant communication overhead and cannot support the fastest network connections such as Infiniband. This will effect closely coupled parallel applications built with the Message Passing Interface (MPI), which commonly run on the NSF TeraGrid. We review these overheads in (24); more extensive investigations are currently submitted for review. Other virtualization approaches that do not use a hypervisor, such as OpenVZ (www.openvz.org), will have smaller overheads (20). In any case, the largest scientific parallel problems will continue to run on very large clusters built with custom architectures such as those funded by the NSF's Track 1 and Track 2 programs, but many other computations are better suited for running on Cloud resources, as we discuss next.

Runtime management: Although one may want to use a Cloud to outsource infrastructure at the operating system level, it is also desirable to have higher-level tools that can harness the computing power of entire Cloud installations. The idea is that the Cloud provides some specific suite of capabilities on top of its infrastructure, and the Cloud user does not drill down to the underlying operating system. Apache Hadoop is a relevant example of this. Hadoop is an implementation of two ideas promulgated by Google: the Google File System and Map-Reduce (22). Strictly speaking, Hadoop and its competitors don't need to run on Cloud infrastructure, but the two are a good match (see for example Amazon's Elastic Map Reduce, aws.amazon.com/elasticmapreduce/). Map-Reduce and its competitors (prominently, Microsoft's Dryad (23)) are designed to solve the world's largest data-file parallel problem: search. Map-reduce is essentially an approach for managing computing tasks in distributed environments that is works very well for certain classes of parallel problems: those associated with fragmentable data sets. Although it can be applied to a wide range of problems (21), 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 that can be split into smaller fragments. In contrast, traditional parallel programming, based around the Message Passing Interface (MPI), is memory parallel rather than file parallel.

The notion of file parallelism can also be extended to network streams 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 (24, 25).

The relevance of both Cloud infrastructure and runtimes to Spatial CI should be clear, and we will next look at relevant examples.

Case Study #1: Flood Grid

Floods are one of the most common and expensive natural hazards in the United States, affecting communities at various levels. To facilitate and improve flood planning, forecasting, damage assessments, and emergency responses, the USGS-funded FloodGrid project provides an integrated platform for inundation modeling, property loss estimation, and visualization. Rather than centralizing all capabilities onto a specific platform, we have developed this system following open service architecture principles, packaging functionalities as Web Services and pipelining them as an end-to-end workflow. Integration is achieved via a simple Web interface requiring minimal user interactions. This is an example of a relatively simple Science Gateway. As we review here, even this simple system combines real-time data services, computational services, GIS information and data services, and several data models. We build some of these services and leverage third party providers for others. We may consider this to be analogous to a Web 2.0 mash-up. FloodGrid is a collaboration between the Polis Center at IUPUI and the Community Grids Laboratory at IU.

The Multi-Dimensional Surface-Water Modeling System (MD_SWMS) (26) from the U.S. Geological Survey (USGS) provides simulations for a variety of environmental and hydraulic models. The Flood Grid pilot study focuses on inundations of the White River at Ravenswood area in Indianapolis, using the 2D hydraulic model, FaSTMECH (27), calibrated for the region. Real-time forecast data of the Nora station (http://www.crh.noaa.gov/ahps2/hydrograph.php?wfo=ind&gage=nori3) provided by the National Weather Service (NWS) Advanced Hydrologic Predication Service (AHPS) serve as initial conditions of the simulation. The Computational Fluid Dynamics General Notation System (CGNS) (28) bridges the computation model and its environmental surface-water applications by providing a standard data format and the framework for exchanging data in that format. Hence the main data stream of Flood Grid studies are in the format of CGNS files. A complete Flood Grid study consists of web services for flood monitoring, simulation, damage estimation, and visualization. Figure 1 outlines the service stack in such a workflow.

The real time river data monitoring service constantly monitors the NWS real-time forecast of the Nora station, 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. These pre-defined conditions are determined for a particular study area by model calibration. During a flood study, the CGNS input process service infuses such information as initial conditions into the pre-calibrated regional model represented by a CGNS file. The updated CGNS file is in turn fed to the flood simulation service as the input to perform the FaSTMECH simulation, which stores computation results by once again updating the given CGNS file. The CGNS output process service parses the FaSTMECH simulation results and generates curvilinear grids in ASCII files. The grid file generation service further consumes these files to produce rectilinear flood depth grids using nearest neighbor clustering techniques. The loss calculation service overlays the generated flood grids with parcel level property data and calculates percentage damages using the

Hazards U.S. Multi-Hazard (HAZUS-MH) (www.fema.gov/prevent/hazus) analysis from Federal Emergency Management Agency (FEMA). Finally the map tile cache service visualizes the study results in Google Maps.

The core flood simulation service wraps the FaSTMECH FORTRAN computation program under the Swarm job scheduling service framework (29). 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 TeraGrid Gateway Hosting Service at Indiana University Bloomington. Flood damage estimation and visualization services are developed with Visual Basic .NET, and deployed under Internet Information Services (IIS) by the Polis Center at Indiana University Purdue University at Indianapolis.

All Flood Grid services are orchestrated into asynchronous workflows under both the .NET framework and Business Process Execution Language (BPEL) modules. The distributed and complex system is however presented to users via a web interface. Figure 2 depicts the layout of this interface on the left, with the corresponding screenshot on the right. Upon registration, a user can request running a new study or review an earlier one in the flood studies control center. Execution statuses of each service in the study workflow are also displayed under this section. For a completed study, simulation results are visualized with Google Maps displaying flooded regions and damaged parcel properties that are obtained from regional Web Feature Services. Detailed analysis reports are available in the damage estimation section, calculated using HAZUS-MH. The map overlay section enables mash-ups with other online geospatial services such as county parcel maps and demographic maps from Social Assets and Vulnerabilities Indicators (SAVI) Community Information System (www.savi.org).

Flood modeling, simulation, forecasting, hazard analysis and disaster management require specialized knowledge, and each are usually well understood only among domain experts. As we have illustrated in this short summary, much of this information and expertise are distributed among several stakeholder groups. If each capability is taken separately, it is difficult to solve important problems such as determining the damage costs to property caused by a particular flood. However, through programmable service interfaces and an open-service architecture, we are able to integrate these into a single tool suitable for on-demand usage. The Flood Grid project brings together specialists such as hydrologists, environmentalists, and hazard analysts with a common platform to seek comprehensive solutions for general flood studies. Under this framework, discrete information pieces can be conveniently assembled into workflows, shared among communities, and made intuitive for the general public. The Flood Grid project demonstrates wide range collaborations from federal, state, and regional agencies, and delivers a Cyberinfrastructure over highly heterogeneous and distributed computation resources.

Although not explored in the current system, it is possible to greatly increase FloodGrid's computing requirements. FaSTMECH computations are on the order of minutes to hours for typical inputs but are limited to a particular area for which we have a model mesh. Increasing the number of models to cover more areas of interest is an obvious way to increase computational requirements. More interesting, perhaps, is partially automating model calibration. FaSTMECH has a large number of input parameters that must be narrowed down to reproduce known historical data. This is currently a manual process performed by a USGS hydrologist. Guided parameter space studies may help speed calibration by running many different scenarios and comparing them to historical data. There are well known parallel algorithms for doing this (30), but as we discussed previously, this is also an excellent use of Cloud Computing runtime environments.

We map the Flood Grid infrastructure requirements to Cloud Computing infrastructure in Table 1. An important requirement for Flood Grid's infrastructure is reliable service hosting to make sure that the services illustrated in Figure 1 are persistently available, with redundancy and load balancing. It is certainly possible to have these capabilities without using virtualization, but virtualization can be used to build redundancy into the fabric of the infrastructure rather than placing this burden on the developers. Clouds would also be useful as providers of standard data libraries through virtual block stores. For Flood Grid, the central piece is a validated CGNS input mesh that models a particular section of a river. Although only one such model was available to us for the study, one can envision a library of calibrated models for different geographic areas. Similarly, standard GIS data sets (parcel and demographic information) can also be delivered in this fashion, coupled to the Web Feature Service that provides them. That is, one does not need to rely upon a third party Web service with its own reliability concerns. Instead, local governments can 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 make use of pre-configured virtual machines that include FasTMECH, Swarm, and all supporting software.

Case Study #2: Polar Grid: Online SAR Image Post-Processing

In this case study, we examine a common Spatial CI problem: image processing. We are motivated by the extremely important problem of glacial melting and the need to determine the underlying rock bed beneath the Greenland and Antarctic glaciers. Detailed knowledge of the rock beds is needed to build more accurate models than are currently available for glacial motion. From the point of view of Spatial CI, these are examples also of data-parallel computing and are well suited for Cloud Computing runtimes discussed previously. The Polar Grid project described here supports the Center for Remote Sensing of Ice Sheets (CReSIS) at the University of Kansas.

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 (31). CReSIS collected and processed approximately 25 TB of raw data using Polar Grid resources during the 2008-2009 campaigns. Raw and initially processed data sets are equivalent to NASA CODMAC data products 0 and 1, and can be managed by a single group as a one-time exercise since there are generally no optional processing steps that need to be explored. However, higher-level data products require human interaction. Improving the SAR image qualities in post-processing is the essential step for any SAR image application. One main image quality issue is speckle noise inevitably generated by SAR. The speckle noise usually appears as random granular patterns, which can reduce the image resolution and give the image a fuzzy appearance. Applying proper filters will enhance the image quality and improve the interpretation of sub-glacial structures. SAR image processing is computationally intensive; it is necessary to develop a scalable parallel Cyberinfrastructure for SAR image post-processing. In this pilot project, we implement the Web Service that gives users the access to testing the three basic filters.

The system architecture is shown in Figure 3. Image processing is done by Matlab scripts, which are complied as a standalone executables by Matlab Compiler. The standalone executable uses the Matlab Compiler Runtime (MCR) engine and can be deployed royalty-free on servers such as TeraGrid clusters and Cloud Computing resources. The Computing Service Server contains multiple MCRs in a computing cluster.

The computing service is exposed as a Web Service to the web developer. The Web Service has the following parameters: dataset, filter, filter parameters, output image size and image name. The response from web service server returns an image-URL, which can be included in web interface. Users select the dataset, filter and filter parameters through a Web interface. The images with different parameters can be compared side by side, and also the ground track of SAR image is display on Google Map, in which user can trace and check the SAR image along the ground track (Figure 4).

The advantage of the system design relays on the separation of Matlab computing service and Web Service server. Depending on the complexity of computing task, computer server can have one or more MCRs, the job scheduler can assign the request to the available MCRs. The MCRs can be distributed dynamically on a virtual on-demanding cluster, yet exposed to the web developer using the same interface. Another advantage of this design is that it clearly separates the roles of the Matlab developer and the web developer, since they are from two different knowledge domains.

We summarize mappings of the Polar Grid project's prototype infrastructure to Cloud Computing requirements in Table 3. As before with FloodGrid, we need persistent, fault tolerant Web service hosting environments, which should be provided by the Cloud infrastructure providers rather than the Polar Grid developers. We likewise need to make standard SAR data sets available. The particular problem to note here is the size of the data: it is prohibitive (unlike FloodGrid) to move the SAR data for image processing on some remote cluster, so we must instead keep the computing power near the data. On the other hand, the SAR files are data-file parallel and so are good candidates for Cloud runtime tools. In the PolarGrid case, we need to extend the runtime engine (Hadoop, et al) to manage the filters shown in Table 3. These filters are compiled binaries that must be run on a compatible operating system (that is, one with a specific version of the Linux kernel), so virtualization can greatly expand the amount of resources available to us, compared to conventional systems. Virtualization also is useful for ensuring that the filter images have exactly the right dependencies (particularly the correct version of the MCR).

Conclusions

In this paper, we have discussed Cloud Computing, which we believe can provide a comprehensive approach to Cyberinfrastructure. Current Cyberinfrastructure deployments, such as the NSF TeraGrid, focus on the requirements of parallel computing. This provides valuable resources for many scientific problems, but it is not optimal for many other fields' computing, data, and hosting requirements. We have proposed that Spatial CI, when viewed in its broad context, provides many motivating scenarios for adopting Cloud Computing. We illustrated this through two small projects, Flood Grid and the Polar Grid portal. Our key point is that Clouds provide more control over the environment to developers through virtualization. This allows, for example, developers to install and control their own software libraries without worrying about version conflicts with developers on unrelated projects.

Computing on clouds is another important topic. Although performance of classic, closely coupled MPI programs degrades on virtual machines, there is a large number of what we dub data-file parallel applications (and generally, problems that have low communication-to-computation ratios) that are very well suited for Cloud-based systems. Finally, we note also the potential importance of data virtualization through virtual block storage systems such as Amazon's Elastic Block Store and its open architecture spinoffs. These can be used to disseminate public and community data sets directly to computing. Although there are many interesting provenance and other metadata and security problems that need to be explored, we believe this is an interesting alternative to simply putting data online or into an online data base.

Large commercial vendors dominate Clouds, but there is a growing collection of open source software that can be used to build research clouds. The challenge over the next several years for core Cyberinfrastructure research will be to investigate and document open architecture Cloud systems. By "open architecture", we mean providing documented, reproducible best practices for building and running academic Clouds to support research. Concurrently, much work needs to be done on providing production quality Cloud facilities to support Spatial CI and other fields that will benefit.

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.

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. 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.
16. 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.
17. Foster, I., et al. (2006) Virtual Clusters for Grid Communities, *CCGRID*: 513-520.
18. 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).
19. Nurmi, D., et al (2008) The Eucalyptus Open-source Cloud-computing System, in *Proceedings of Cloud Computing and Its Applications*, Chicago, IL (October 2008).
20. Kim, K. et al (2008) SQMD: Architecture for Scalable, Distributed Database System built on Virtual Private Servers. *Proceedings of E-Science for Cheminformatics and Drug Discovery workshop at E-Science 2008* , Indianapolis, IN Dec 12, 2008.
21. Chu C-T, et al (2006). Olukotun, Map-Reduce for Machine Learning on Multicore, *NIPS*: 281-288.
22. Dean, J., Ghemawat, S. (2008) MapReduce, Simplified Data Processing on Large Clusters. Commun, *ACM* 51(1): 107-113.
23. Isard, M., Budiu, M., Yu Y., Birrell, A., Fetterly, D. (2007) Dryad, Distributed Data-Parallel Programs from Sequential Building Blocks, *EuroSys*: 59-72.
24. 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
25. 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.
26. McDonald, R.R., Nelson, J.M., and Bennett, J.P., in press, Multi-dimensional surface-water modeling system user's guide: U.S. Geological Survey Techniques and Methods, 6-B2, 136 p.
27. 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.

28. 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.
29. 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.
30. Fox, G., Messina, P., Williams, R. (1994) *Parallel Computing Works! (*Morgan Kaufmann, San Mateo CA)
31. Allen, C., and J. Paden (2007), Synthetic-Aperture Radar Images Polar Ice-Sheet Bed, *SPIE Newsroom* [DOI: 10.1117/2.1200706.0780].