OMICS A Journal of Integrative Biology
Volume 15, Number 4, 2011
© Mary Ann Liebert, Inc.
DOI: 10.1089/omi.2011.0013

Technology and Data-Intensive Science in the Beginning of the 21st Century

Philip A. Bernstein,[1] Dave Wecker,[1] Ashok Krishnamurthy,[2] Dinesh Manocha,[3] Jeffrey Gardner,[4] Natali Kolker,[5] Chance Reschke,[4] Jesse Stombaugh,[6] Pamela Vagata,[1] Elizabeth Stewart,[5] Dean Welch,[5] and Eugene Kolker[4,5]

Abstract

This article is a summary of the technology issues and challenges of data-intensive science and cloud computing as discussed in the Data-Intensive Science (DIS) workshop in Seattle, September 19–20, 2010.

Introduction

The recent trend in scientific and related applications has been toward using the cloud infrastructure to allow users to deploy their applications and data by providing the infrastructure, platform, and software as a service. Fundamentally speaking, cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources. This includes networks, servers, storage, applications, and services that can be rapidly provisioned and released with minimal management effort and service provider interaction. Even though there have been exciting developments in terms of large cloud infrastructures (e.g., Amazon S3/EC2, Microsoft Windows Azure), they may not offer enough flexibility to users in terms of application usability and direct user control. As a result, we need considerable research toward developing next-generation cloud-computing architectures.

The computing landscape has been constantly evolving over the last few decades. Current high-performance and data-intensive computing systems rely exclusively on commodity components largely developed for the PC and server markets that are able to capitalize on the enormous design and fabrication costs of these components.

One such commodity component is the graphics processor unit (GPU) (Fig. 1), developed to address the demand for high-fidelity interactive computer games as well as sophisticated user interfaces in computers and mobile devices. High-end graphics processing units (GPUs) now provide an order of magnitude more computing power than other commodity processors and offer a higher performance/price ratio. Moreover, this trend is expected to continue into the near future.

The use of GPUs or many-core processors as accelerators brings many challenges. A new set of parallel algorithms and programming tools are needed to exploit the data-parallelism and high number of light-weight cores on these processors. Much of the recent work has fallen into the areas of coarse-grain parallelization, that is, new programming models and different ways to exploit threads and data-level parallelism. In the future, we need better and automated ways to improve performance by utilizing fine-grained parallelism. The main difficulty in terms of data-intensive computing is the distribution of data memory accesses across the data caches of each core, especially when cache sizes in these many-core processors are highly variable. Poor choices in the placement of data accesses can lead to increased memory stalls and low resource utilization. Finally, we need novel and improved cloud computing architectures that include support for these accelerators and provide sufficient flexibility for applications to utilize them.

Computation

Current state

Distributed computing platforms constructed from commodity components have dominated high-end computing for more than a decade. These range from the 16-node laboratory cluster to the immense systems operated by the National Science Foundation (NSF) Teragrid and commercial cloud providers. Shared-memory supercomputers have also been used, although they have not been as cost effective for most

[1]Microsoft Research, Redmond, Washington.
[2]Ohio Supercomputer Center, Columbus, Ohio.
[3]University of North Carolina, Chapel Hill, North Carolina.
[4]University of Washington, Seattle, Washington.
[5]Seattle Children's Research Institute, Seattle, Washington.
[6]University of Colorado, Boulder, Colorado.

FIG. 1. GPU image.

applications. Consequently, scaling applications requires adopting a distributed computing model such as MPI or Map-Reduce. Moreover, the recent trend in processing architectures is to increase the number of cores or data parallelism, and not the clock speed of individual processors. As a result, even applications operating at modest scales must adopt parallel approaches.

Applications developed for one platform are often poorly suited for another. The trend toward packaging application stacks as virtual machines helps streamline deployment, but portability is often limited to individual service providers (e.g., Amazon Web Services).

Barriers

Programming models for distributed systems are hard to use. Exploiting chip-level parallelism is challenging, especially for data-intensive computation. With the exception of Map-Reduce style systems, such as Hadoop, existing programming models are insufficiently resilient to cope with anticipated component failure rates in very large systems. The requirement for CPU and data to come together adds to the challenge of adopting cloud platforms for widely distributed instruments. Commercial cloud and Teragrid systems lack the configuration flexibility expected by some users. Commercial cloud prices are often uncompetitive with large private cloud deployments.

Future/outlook

Funding barriers to choosing the most appropriate platform for a given task must be removed. Ideally, computational infrastructure should support the notion of "supercomputers as accelerators" in which applications launched from portable clients automatically execute on appropriate, elastic, remote platforms.

Systems should enable users to ask for answers that are "good enough" relative to prespecified constraints on the cost and time needed to stage data and complete the necessary calculations.

Innovators must accept that the components that define "commodity" systems tomorrow will differ from those of today. These trends include increasing chip level parallelism in CPUs and GPUs, increasing scale and bandwidth in shared memory support (e.g., Intel QPI, POWER7 MCM, AMD Hypertransport), and new layers in the memory and storage cache hierarchy (e.g., flash-based SSDs, phase-change memory). Success depends largely on the designer's ability to hide this complexity from users while preserving performance gains.

Storage

Current state

Storage support for data-intensive computing currently varies considerably, depending on the service provider and the scale at which it is deployed. At the high end, two models dominate. Among supercomputer deployments, shared storage pools based on enterprise hardware managed by cluster file systems are standard. A variety of approaches for data warehousing based on shared nothing architectures are used in cloud systems, constructed primarily from commodity components, where data is replicated for high availability. Some commercial "platform as a service" providers also rely on redundant commodity hardware; however, individual virtual machine instances access this storage as a shared pool. The cost per byte of the most widely implemented approaches remains much more costly than commodity disk drives (Fig. 2).

These approaches need host systems, controllers, interconnects, software, and data center infrastructure to achieve maximum functionality. Researchers are faced with preserving large and growing volumes of data, but economical options for archiving large data sets are uncommon today.

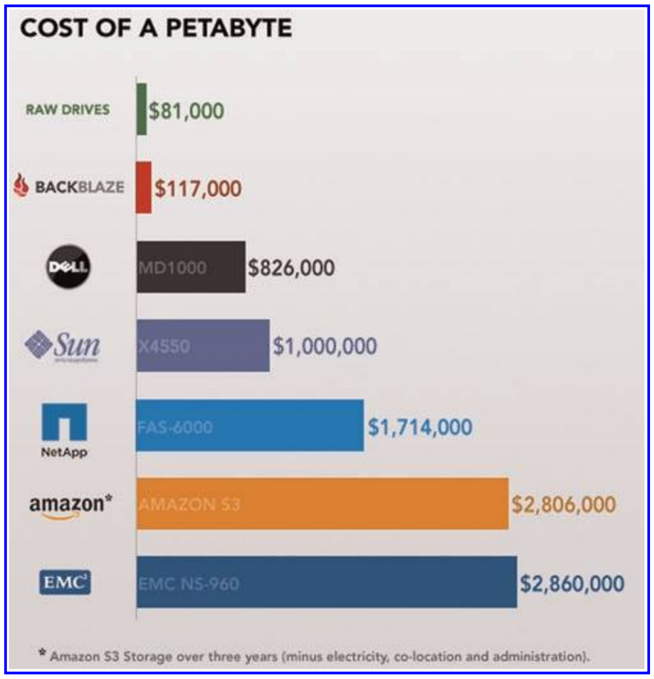

FIG. 2. Comparison of the cost of 1 petabyte for various solutions.

Barriers

Data center costs (such as power, infrastructure, and operations) dominate equipment costs when using commodity hardware. Higher density disks help, but maintaining redundant systems to address the reliability and performance limitations of inexpensive hardware adds complexity to solutions.

Shared storage pools allow users to interact with their data in a familiar manner. However, if storage is directly attached to servers (i.e., using shared nothing architecture), then applications that require frequent random access to their data must be partitioned to maximize local data accesses to the processor that holds the data. New applications should be designed with such partitioning in mind. However, this is a new kind of thinking for many developers of scientific applications.

Applications utilizing many files that must be opened and closed frequently cause an even more basic performance problem. This problem commonly occurs at supercomputer centers as users attempt to scale their difficult to support applications. Breaking these habits is simpler than refactoring for shared nothing, but still requires effort, effort that is repeated with every new user. Finally, offering services intended to operate over long periods, such as data archives, present funding challenges.

Future/outlook

To be most effective, either all data must be warehoused on the platform where it will be analyzed (shared nothing), or the performance of shared storage pools and storage interconnects must scale with the available CPU power. Ultimately, hybrid approaches in which node-local storage serves as a layer in the I/O hierarchy are likely to continue to play a role. Shared nothing and hybrid approaches will benefit from improvements in systems management and user interfaces that hide their complexity. Very fast networks connecting computers and instruments are essential for data provisioning. Ideally, archiving systems would allow users to pay once and store forever.

Network

Current state

Current campus networks are highly variable. Although the typical campus backbone is capable of 10 Gbps bandwidth, individual buildings may have 1 Gbps or less, and individual labs and desktops within these buildings typically have 1 Gbps or less.

Off-campus bandwidth availability is dependent on the nature of the network. Typically, Research 1 universities have dedicated bandwidth to Internet 2 and other national networks. Today, the intelligent filtering of data to remove bad data sets and reduce the size of data sets is done manually, with only a small amount of automation. Departmental networks must cope with firewalls, which are usually managed at the department or college level.

Barriers

Typical barriers to adoption of improved networks are the decentralized growth of DIS data sources and the acquisition of instruments not coupled with campus infrastructure improvements. Funding improved bandwidth to buildings and laboratories is difficult in the current financial environment. Acquisition of off-campus bandwidth is expensive, although somewhat aided by the recent Broadband Technology Opportunities Program awards as part of the American Recovery & Reinvestment Act. Intelligent data filtering is domain specific and requires time and effort to develop the required rules and software. Working around firewalls is difficult because network and system administrators are sometimes more concerned about ensuring security than enabling research.

Future/outlook

In the future, we need a campus data cloud that is designed and implemented to address DIS. An intelligent campus network needs a hierarchical hub arrangement. Instruments and other data sources should have sufficient bandwidth connections (10 Gbps now, 100 Gbps or greater in the future) to the hubs. The hubs should, in turn, be connected to data centers at higher bandwidths (100 + Gbps.) With sufficient off-campus bandwidth, we can have regional data centers that form a regional cloud, which in turn connects to the national cloud. Intelligent data filtering will ease the researchers' selection of data that needs to be moved to the campus/regional/national cloud.

Campus executives and Chief Information Officers need to understand DIS and the key role that campus infrastructure plays in effectively preparing for the future of DIS. Research needs to clearly articulate the expected return for these infrastructure investments.

Output and Visualization

Current state

If data and computation are housed in the cloud, then it becomes the responsibility of the cloud provider to develop the tools for visualizing and presenting the output. Today, most of the analytical software used to process the data assumes that the consumption of the output will be at the same location and is designed for a typical desktop-type environment. High-end visualization typically is centralized at a few locations on campus. Thus, the output that is created does not know or care about either the network (bandwidth and latency) over which the results are to be piped or the resolution of the device used for viewing the output. Finally, the analytical software is often limited in the type of device that can display its output.

Analytical software has to be made network and output device aware and device agnostic.

Future/outlook

It is clear that the devices over which users will be consuming the output of their computations will have varying resolution and interactivity (iPhone to iPad to Desktop to Cave) and will be connected over networks of different capacity (3G/Wimax, WiFi, 100 Mbps, 1 Gbps). Thin client output devices, in which only the pixels to be drawn are sent over the channel, will be increasingly important. Thus, the analytical software of the future should intelligently produce output that is network and device aware. Further, the software should dynamically change in the network.

Most campus administrators already see the proliferation of devices that connect to the campus network and are moving to provide support for these devices. They need to be made aware that DIS also has similar needs and should be supported along with the administrative functions.

Hardware and Datacenter Trends and Constraints

Current state

Large computing environments are trending toward movement from special purpose to commodity hardware. However, efficiencies that have been lost along the way are becoming apparent. Lots of tiny computing clusters at all levels of organizations are underutilized and not properly considered by accounting (e.g., power cost is hidden in overhead). Most software has been developed for sequential execution. Although some of today's software exploits parallelism, this has yet to become status quo. The move to a distributed model has been difficult if not impossible for nonexpert software developers. Current cloud vendors have chosen their main users from domains that do not overlap with data-intensive scientific investigation (DISC). Data gathering is currently done (mostly) in small, distributed increments. This incurs large overhead, low utilization, and fragmentation of data availability. Archiving the data, analytic software, and analysis results is difficult and expensive. Thus, result reproduction or regeneration in the future is almost impossible (Fig. 3).

Barriers

Hardware innovation is challenging because of the reliance on commodity solutions. It is difficult to utilize and depend upon shared resources. The researcher is forced to give up control and, simultaneously, is not held accountable for the hidden costs. Current programming models do not lend themselves to efficient, automatic parallelization and distribution and require both domain knowledge and technical skills. Cloud vendors do not see a business case for DISC. Current cloud models require homogeneous facilities, whereas domain-specific hardware solutions might be required for DISC. Generation of local data allows for control, filtering, and validation by the researcher, a convenience that few want to lose (no one wants to FedEx their samples). Further, prestige is based on local instrumentation (e.g., every hospital wants the latest MRI scanner). Today, research done in a way that is maintainable after the work is done is expensive.

FIG. 3. A large server farm.

Future/outlook

Specialized technologies need to overcome the reliance on current commodity hardware. The goal is to move successful, specialized technologies into future commodity solutions. We need to foster the move to greater use of shared resources by making it desirable, including maintenance, availability, support, programmability and flexibility. The rise of data-based analysis (nonprogramming) and automatic scalability (minimal programming) will become prevalent, along with deep control of execution for the technically skilled. The next large segment of users of the cloud will be DISC-based and the vendors, through their own self-interest, will be forced to support them. The heterogeneous nature of domain requirements will force private clouds that are more tailored to a given environment or hosted clouds for specific purposes. Due to the sheer volume of data, efficiencies and overhead costs, pooled sensor locations could become the norm. Further, they may even be colocated (or "near-located") to data centers (e.g., sequencing farms), solving many of the identified problems. By moving work to a cloud model (private/public), a virtual machine environment makes archiving and rerunning cheaper and easier to accomplish.

Representation and Engagement with Society

We need to provide more support for innovative datacenter designs. A change of mindset needs to be promoted for the benefits of leveraging shared resources. We need to teach new models for generating results without requiring esoteric programming skills. Funding DISC projects on clouds will show the limitations and point the direction to architecture changes. Funding of shared instrument facilities would help promote efficiencies. Requiring proper documentation and archiving to receive funding could motivate change.

Effective Use of Time and Resources

Current state

Scientists spend a lot of time preparing their data for computational analyses. Most of this time is spent cleaning, transforming, and validating the datasets. When cleaning the datasets, scientists utilize statistics to filter their data, identify and fix outliers or missing data, and rationalize irregularities in data formats (e.g., ad hoc spreadsheets). The transformations of datasets consist of changes in format, data type, units of measure, although validation can occur at the level of data and/or schema. Currently, there are some domain-specific agreed-upon standards, although these standards have not been widely adopted. Along with the preparation of datasets, scientists are developing the tools to move and visualize their data.

Barriers

There are not many general-purpose tools for automating data processing, nor is there sufficient funding to develop such tools. In some cases, the state of the art is not good enough to automate this work. Before development of tools, scientists in a domain must agree upon and standardize their formatting requirements, allowing for the development of software interfaces that currently do not exist.

Future/outlook

In general, scientists require more automation and increased use of existing automation. It would be useful to scientists if they had access to laboratory information systems, more standards, and open-source libraries of reusable tools. It would also benefit the scientist to develop tools that are easily adaptable when new formats are introduced.

Application Trends

Current state

Software used by research groups to analyze data is often written by members of the research team. One advantage of this approach is that the software is being written by domain experts, making it more likely that it correctly addresses the scientific needs. Also, because the applications are being developed for use by a small group of users to handle very specific tasks, basic needs of the users are considerably easier to meet.

However, there are disadvantages to this approach as well. One disadvantage is excessive duplicated effort (and, hence, extra expense). Multiple research groups write multiple software applications that essentially do the same thing, according to the NSF Office of Cyberinfrastructure Task Force on Data and Visualization (NSF_OCI_TFDV, in press). Further, people in research groups writing software typically have minimal software development expertise. As a result, although the software may work to meet the current needs of the group, when needs change it may be very difficult to modify the software. Depending on the changes in functionality it may even be easier to throw away the old software and rewrite new software from scratch. Also, if there is interest from other research groups in using this software, it may be difficult to modify the software to be usable by these other groups. In addition, the lack of standard data formats makes it difficult for software applications built by different research groups to interoperate. Interweaving software may require building custom adaption layers.

Another drawback of software developed in-house is that it will normally be geared towards nonscalable architectures, modifying it to work with scalable architectures may be prohibitively difficult.

Barriers

To improve the current situation, several barriers need to be overcome. More general-purpose software may not suit researchers' specific needs. In both reality and perception, researchers may resist using externally developed software for fear of not having control of the development of the software (e.g., not controlling the priority of new features). Writing more general-purpose software will require standard data formats. Getting a consensus on these formats will likely be difficult.

Understanding science and writing robust software are both challenging tasks. Writing scientific software requires the developers to have some understanding of the scientific domain; developers with this knowledge are not very common. Reciprocally, because the software development is driven by the needs of the research group, it is important for scientists to have some understanding of software design and development practices. Writing software to be more flexible and robust has immediate costs, but the benefits are typically only seen in the future. As a result, much software is fragile and not reusable. These challenges are not unique to scientific software, but it is a problem with software in general. Good software development requires a conscious effort to pay the short-term price for the long-term benefit.

Pure technical challenges also exist. Writing scalable software is difficult and, because most software developers do not work on these kinds of systems, it is a skill that most of them do not possess. Even if the developers are available, complex applications may make writing professional quality scalable software too expensive for an individual researcher. Also, software-licensing models are often incompatible with using shared or remote resources.

Finally, acquiring funding to create sustainable, sharable, scalable, and general-purpose software is currently difficult.

Future/outlook

The scale of science should be limited solely by the researcher's vision and not by the challenges presented by data size, data complexity, or computational demands. Scientists have valuable skills and should spend their time doing science, not programming or doing data management. Programming models that make the scale of computation transparent to developers would be a big step forward.

Conclusion

Encouraging broader collaboration will motivate the need for sharable software. Collaborations will be more successful, further motivating adoption of more efficient software development strategies. Research groups that develop software could get credit for advancing research if their software is used in discovery. Finally, funding agencies would encourage and fund development of sustainable, scalable software.

Acknowledgments

This policy report and DIS workshop were supported by SCRI and NSF Grant DBI-0969929 to E. Kolker (Principal investigator). The views expressed in this article are entirely personal opinions of the authors and do not necessarily represent positions of their affiliated institutions or NSF.

Author Disclosure Statement

The authors declare that no conflicting financial interests exist.

Reference

National Science Foundation, NSF (2011). Office of Cyberinfrastructure, Task Force on Data and Visualization (in press).

Address correspondence to:
Eugene Kolker, Ph.D.
Seattle Children's Research Institute
1900 Ninth Avenue
C9S-9
Seattle, WA 98101

E-mail: eugene.kolker@seattlechildrens.org