Programming Paradigms for Technical Computing on Clouds and Supercomputers

Geoffrey Fox, Indiana University
Dennis Gannon, Microsoft

In the past four years cloud computing has emerged as an alternative platform for high performance computing but there is still confusion about the cloud model and its advantages and disadvantages over tradition supercomputing based problem solving methods [1, 2]. We characterize the ways in which cloud computing can be used productively in scientific and technical applications. As we shall see there is a large set of application that can run on a cloud and a supercomputer equally well. There are also applications that are better suited to the cloud and there are applications where a cloud is a very poor replacement for a supercomputer. Our goal is to illustrate where cloud computing can complement the capabilities of a contemporary massively parallel supercomputer. Several Azure applications are described in detail in [2].

Defining the Cloud as Public and Private, Commercial and Academic

The term cloud is being in many ways so let's first define a public data center model that describes the major offerings of Microsoft, Amazon and Google. Their data centers are composed of containers of racks of servers which number between 10,000 and a million. Each server has 8 or more cpu cores and around 64GB of shared memory and one or more terabyte local disk drives. GPUs or other accelerators are not common. There is a network that allows messages to be routed between any two servers, but the bisection bandwidth of the network is very low and the network protocols implement the full TCP/IP stack so that every server can be a full Internet host with optimized traffic between users on the Internet and the servers in the cloud. In contrast supercomputer networks minimize interprocessor latency and maximize bisection bandwidth. Application data communications on a supercomputer generally take place over specialized physical and data link layers of the network and interoperation with the Internet is usually very limited.

Each server in the data center is host to one or more virtual machines and the cloud runs a "fabric controller" which manages large sets of VMs fort scheduling and fault tolerance across the servers and acts as the operating system for the data center. An application running on the data center consists of one or more complete VM instances that implement a web service. The basic unit of scheduling involves the deployment of one or more entire operating systems, which is much slower than installing and starting an application on a running OS. Most large scale cloud services are intended to run 24x7, so this long start-up time is negligiblen although running a "batch" application on a large number of servers can be very inefficient because of the long time it may take to deploy all the needed VMs. Data in a data center is stored and distributed over many spinning disks in the cloud servers. This is a very different model than found in a large supercomputer, where data is stored in network attached storage. Local disks on the servers of supercomputers are not frequently used for data storage.

There are more types of clouds than is described by this public data center model. For example, to address a technical computing market, Amazon has introduced a specialized HPC cloud that uses a network with full bisection bandwidth and supports GPGPUs. The major commercial clouds offer higher level capabilities -- commonly termed Platform as a Service PaaS – built on a basic scalable IaaS Infrastructure as a Service. For technical computing, important platform components include tables, queues, database, monitoring, roles (Azure), and the cloud characteristic of elasticity (automatic scaling). MapReduce, which is discussed below, is another major platform service offered by these clouds. Currently the different clouds have different platforms although the Azure and Amazon platforms have many similarities. The Google Platform is targeted at scalable web applications and not as broadly used in technical computing community as Amazon or Azure, but it has been used on some very impressive projects. We expect more academic interest in PaaS as the value of platform capabilities become clearer.

“Private clouds” are small dedicated data centers that have various combinations of the properties above and typically use one of the four major open source (academic) cloud environments Eucalyptus, Nimbus, OpenStack and OpenNebula (Europe) which focus at the IaaS level with interfaces similar to Amazon. FutureGrid is an NSF research testbed for cloud technologies and it operates a grid of cloud deployments running on modest sized server clusters with support for all four academic IaaS. Private clouds do not fully support the interesting platform features of commercial clouds. Open source Hadoop and Twister offer MapReduce features similar to those on commercial cloud platforms and there are open source possibilities for platform features like queues (RabbitMQ, ActiveMQ) and distributed data management system (Apache Cassandra). However, there is no complete packaging of PaaS features available today for academic or private clouds. Thus interoperability between private and commercial clouds is currently only at IaaS level where it is possible to reconfigure images between the different virtualization choices and there is an active cloud standards activity. The major commercial virtualization products such as VMware and Hyper-V are also important for private clouds but also do not have built-in PaaS capabilities.

Parallel Programming Models

Scientific Applications that are run on massively parallel supercomputers follow strict programming models that are designed to deliver optimal performance and scalability. Typically these programs are designed using a bulk synchronous model and the Message Passing Interface (MPI) communication library, where each task iterates a set of compute –communicate phases. Because the machine has been designed to execute MPI very efficiently, the relative time spent in communicating data can be made small compared to the time doing actual computation and the application scales very well to large numbers of processors.

The typical cloud data center does not have a low latency high bandwidth network needed to run communication intensive MPI programs. However, there is a subclass of traditional parallel programs called MapReduce computations that can do well on clouds architectures. MapReduce computations have two parts. In the first part, you “map” a computation to each member of an input data set. Each of these computations has an output. In the second part you reduce the outputs to a single output. There are several variants of MapReduce. In one case there is no real reduce step. Instead you just apply the operation to each input in parallel. This is often called an “embarrassingly parallel” computation. At the other extreme there are computations that use MapReduce inside an iterative loop. A large number of linear algebra computations fit this pattern. Clustering and other machine learning computations also take this form. Figure 1 below illustrates these different forms of computation. Cloud computing works well for any of these MapReduce variations. In fact a small industry has been built around doing analysis (typically called data analytics) on large data collections using MapReduce on cloud platforms.

Note that the synchronization costs of clouds lie in between those of grids and supercomputers (HPC clusters) and applications suitable for clouds include all computational tasks appropriate for grids and HPC applications without stringent synchronization. Synchronization in clouds includes both effects of commodity networks but also overheads due to shared systems and virtualized (software) system. The latter is illustrated from Azure application reported by Indiana University using Twister4Azure an early iterative MapReduce environment. One finds the typical parallel computing structure with concurrent tasks divided into execution (map) and communication stages. The measurements in [3] show a familiar structure but with increased fluctuation in task execution time that degrade performance at communication synchronization points.

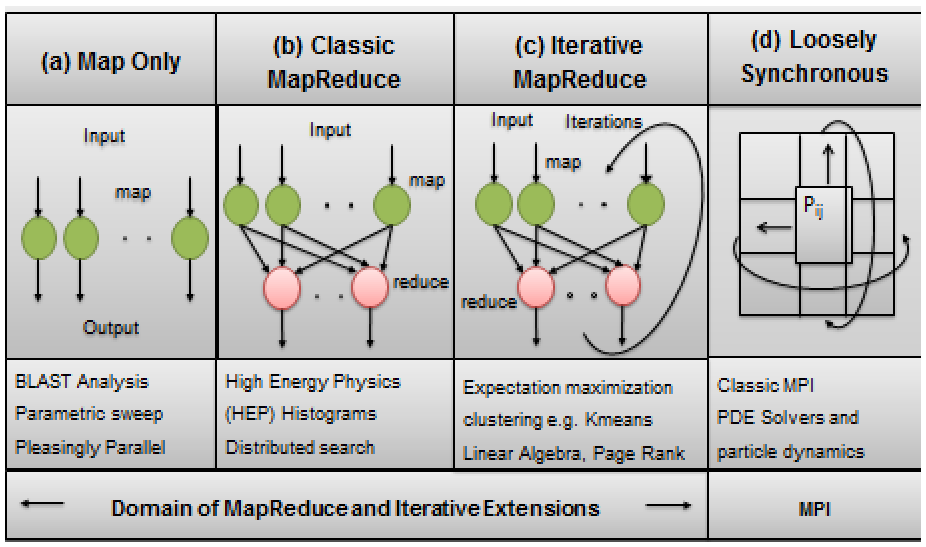

Fig 1: Forms of Parallelism and their application on Clouds and Supercomputers

Classes of Cloud Applications

Limiting the applications of the cloud to classical scientific computation would miss the main reason that the cloud exists. Large scale data center are the backbone of many ubiquitous cloud services we use every day. Internet search, mobile maps, email, photo sharing, messaging, social networks are all dependent upon large data center clouds. These applications all depend upon the massive scale and bandwidth to the Internet that the cloud provides. Note clouds are executing jobs at scales much larger than those on supercomputers but that scale does not come from tightly coupled MPI. Rather scale can come from two aspects: Scale (or parallelism) from a multitude of users and scale from execution of an intrinsically parallel application within a single job (invoked by a single user).

Clouds naturally exploit parallelism from multiple users or usages. The Internet of things will drive many applications of the cloud. It is projected that there will soon be 50 billion devices on the Internet. Most will be small sensors that send streams of information into the cloud where it will be processed and integrated with other streams and turned into knowledge that will help our lives in a million small and big ways. It is not unreasonable for us to believe that we will each have our own cloud-based personal agent that monitors all of the data about our life and anticipates our needs 24x7. The cloud will become increasing important as a controller of and resource provider for the Internet of Things. As well as today’s use for smart phone and gaming console support, “smart homes” and “ubiquitous cities” build on this vision and we could expect a growth in cloud supported/controlled robotics.

Beyond the Internet of things, we have the commodity applications such as social networking, internet search and e-commerce. Finally we mention the “long tail of science” [4, 5] as an expected important usage mode of clouds. In some areas like particle physics and astronomy, i.e. “big science”, there are just a few major instruments generating now petascale data driving discovery in a coordinated fashion. In other areas such as genomics and environmental science, there are many “individual” researchers with distributed collection and analysis of data whose total data and processing needs can match the size of big science. Clouds can provide scaling use for this important aspect of science.

Large scale Clouds also support parallelism within a single job with an obvious example of Internet search which has both “usage parallelism” listed above but also parallelism within each search as the summary of the web is stored on multiple disks on multiple computers and searched independently for each query. MapReduce was introduced to support successfully this type of parallelism. Today probably the “Map only” or Pleasingly Parallel mode of fig. 1 is most common where a single job invokes multiple independent components. This is very similar to the parallelism from multiple users and [2] gives examples where the different “map tasks” from a “single application” are generated outside the cloud which sees these tasks as separate usages. Basic data analysis can often be formulated as a simple MapReduce problem where independent event selection and processing (the maps) is followed by a reduction forming global statistics such as histograms. The final class of parallel applications explored so far is Iterative MapReduce implemented on Azure as Daytona from Microsoft or Twister4Azure from Indiana University. Here Page Rank is a well-known component of Internet search technology that corresponds to finding eigenvector of largest eigenvalue for the sparse matrix of internet links. This algorithm illustrates those that fit well Iterative MapReduce and other algorithms with parallel linear algebra at their core have been explored on Azure.

As well as data parallelism discussed above, clouds also support the functional parallelism seen where a given application breaks into multiple components which typically supported by workflow technology [6, 7]. Workflow is an old idea and was developed extensively as part of Grid research but its use can be taken over directly by clouds without conceptual change. In fact the importance of Software as a Service for commercial clouds illustrates that another concept Services developed for science by the grid community is a successful key feature of cloud applications. Workflow as the technology to orchestrate or control clouds will continue to be a critical building block for cloud applications.

Important access models for clouds include portals, which are often termed Science Gateways and their cloud use is similar to that in grids. Another interesting access choice seen in some cases is that of the queue either implemented using conventional publish-subscribe technology (such as JMS) or the built in queues of the Azure and Amazon platforms. Applications can use the advanced platform features of clouds (queues, tables, blobs, SQL as a service for Azure) to build advanced capabilities more easily than on traditional (HPC) environments. Of course the pervasive “on demand” nature of cloud computing emphasizes the critical importance of task scheduling where either the built-in cloud facilities are used or alternatively there is some exploration of technologies like Condor developed for grids and clusters.

The nature of the use of data [8] is another interesting aspect of cloud applications that currently is still in its infancy but is expected to become important as for example future large data repositories will need cloud computing facilities. A key challenge as the data deluge grows is how we avoid unnecessary data transport and if possible bring the computing to the data [9, 10]. We need to understand the tradeoffs between traditional wide area systems like Lustre, Object stores which the heart of Amazon, Azure and OpenStack storage today and the "data-parallel" file systems popularized by HDFS, the Hadoop File System. We expect this to be a growing focus of future cloud applications.

Many attempts to directly port a conventional HPC application to a cloud platform fail[1]. It is not unlike early attempts to move "vectorized" HPC application to massively parallel non-shared memory message passing clusters. The challenge is to think differently and rewrite the application to support the new computational and programming models. In the case of clouds we suggest six principles: Build the application as a service; Build on existing cloud deployments such as Hadoop; Use PaaS if possible; Design for failure; Use as a Service (e.g. SQLaaS) where possible; Moving Data is a challenge.

References

1. Katherine Yelick, Susan Coghlan, Brent Draney, and Richard Shane Canon, *The Magellan Report on Cloud Computing for Science*. December 2011. http://www.nersc.gov/assets/StaffPublications/2012/MagellanFinalReport.pdf.
2. Geoffrey Fox and Dennis Gannon. *Cloud Programming Paradigms for Technical Computing Applications*. 2012 January 17 Available from: http://grids.ucs.indiana.edu/ptliupages/publications/Cloud%20Programming%20Paradigms.pdf.
3. Thilina Gunarathne, Bingjing Zhang, Tak-Lon Wu, and Judy Qiu, *Scalable Parallel Scientific Computing Using Twister4Azure*. February 11, 2012. http://grids.ucs.indiana.edu/ptliupages/publications/scientific_applications_of_twister4azure_journal_submit4.pdf.
4. P. Bryan Heidorn, *Shedding Light on the Dark Data in the Long Tail of Science.* Library Trends, 2008. 57(2, Fall 2008): p. 280-299. DOI:10.1353/lib.0.0036
5. Peter Murray-Rust. *Big Science and Long-tail Science*. 2008 January 29 [accessed 2011 November 3]; Available from: http://blogs.ch.cam.ac.uk/pmr/2008/01/29/big-science-and-long-tail-science/.
6. Deelman, E., D. Gannon, M. Shields, and I. Taylor, *Workflows and e-Science: An overview of workflow system features and capabilities.* Future Generation Computer Systems, May 2009, 2009. 25(5): p. 528-540. DOI:http://dx.doi.org/10.1016/j.future.2008.06.012
7. Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers, *Examining the Challenges of Scientific Workflows.* Computer, 2007. 40(12): p. 24-32. DOI:10.1109/mc.2007.421
8. Geoffrey Fox, Tony Hey, and Anne Trefethen, *Where does all the data come from?* , Chapter in *Data Intensive Science* Terence Critchlow and Kerstin Kleese Van Dam, Editors. 2011. http://grids.ucs.indiana.edu/ptliupages/publications/Where%20does%20all%20the%20data%20come%20from%20v7.pdf.
9. Faris, J., E. Kolker, A. Szalay, L. Bradlow, E. Deelman, W. Feng, J. Qiu, D. Russell, E. Stewart, and E. Kolker, *Communication and data-intensive science in the beginning of the 21st century.* Omics: A Journal of Integrative Biology, 2011. 15(4): p. 213-215. http://www.ncbi.nlm.nih.gov.offcampus.lib.washington.edu/pubmed/21476843
10. Jim Gray, Tony Hey, Stewart Tansley, and Kristin Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. 2010 [accessed 2010 October 21]; Available from: http://research.microsoft.com/en-us/collaboration/fourthparadigm/.

[1] The exception to this is the Amazon HPC Cloud which seems to perform very well.