Scientific Computing Supported by Clouds, Grids and HPC(Exascale) Systems

June 24 2012
HPC 2012 Cetraro, Italy

Geoffrey Fox
gcf@indiana.edu
Informatics, Computing and Physics
Indiana University Bloomington
Science Computing Environments

• **Large Scale Supercomputers** – Multicore nodes linked by high performance low latency network
  – Increasingly with GPU enhancement
  – Suitable for highly parallel simulations

• **High Throughput Systems** such as European Grid Initiative EGI or Open Science Grid OSG typically aimed at pleasingly parallel jobs
  – Can use “cycle stealing”
  – Classic example is LHC data analysis

• **Grids** federate resources as in EGI/OSG or enable convenient access to multiple backend systems including supercomputers
  – **Portals** make access convenient and
  – **Workflow** integrates multiple processes into a single job

• Specialized **visualization, shared memory parallelization** etc. machines

https://portal.futuregrid.org
Some Observations

• **Classic HPC machines** as MPI engines offer highest possible performance on closely coupled problems
  – Not going to change soon (maybe delivered by Amazon)

• **Clouds** offer from different points of view
  • On-demand service (*elastic*)
  • **Economies of scale** from sharing
  • Powerful new **software models** such as MapReduce, which have **advantages** over classic HPC environments
  • Plenty of jobs making it attractive for students & curricula
  • Security challenges

• **HPC problems** running well on clouds have above advantages

• Note 100% utilization of Supercomputers makes elasticity moot for capability (very large) jobs and makes capacity (many modest) use not be on-demand

• Need **Cloud-HPC Interoperability**

[https://portal.futuregrid.org](https://portal.futuregrid.org)
Cloud Computing’s Role in Job Creation
Public and private cloud services will generate nearly 14 million jobs worldwide by 2015, according to new Microsoft research.

Cloud computing efficiencies allow organizations to invest more broadly and apply this innovation to hiring more sales, finance, production, marketing people and more.

14 million Cloud Jobs by 2015
Clouds and Grids/HPC

• Synchronization/communication Performance
  Grids > Clouds > Classic HPC Systems

• Clouds naturally execute effectively Grid workloads but
  are less clear for closely coupled HPC applications

• Service Oriented Architectures portals and workflow
  appear to work similarly in both grids and clouds

• May be for immediate future, science supported by a
  mixture of

  – **Clouds** – some practical differences between private and public
    clouds – size and software
  – **High Throughput Systems** (moving to clouds as convenient)
  – **Grids** for distributed data and access
  – **Supercomputers** ("MPI Engines") going to exascale

https://portal.futuregrid.org
What Applications work in Clouds
• Pleasingly parallel applications of all sorts with roughly independent data or spawning independent simulations
  – Long tail of science and integration of distributed sensors
• Commercial and Science Data analytics that can use MapReduce (some of such apps) or its iterative variants
  (most other data analytics apps)
• Which science applications are using clouds?
  – Many demonstrations –Conferences, OOI, HEP ....
  – Venus-C (Azure in Europe): 27 applications not using Scheduler, Workflow or MapReduce (except roll your own)
  – 50% of applications on FutureGrid are from Life Science but there is more computer science than total applications
  – Locally Lilly corporation is major commercial cloud user (for drug discovery) but Biology department is not
Parallelism over Users and Usages

• “Long tail of science” can be an 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 convenient resources for this important aspect of science.
• Can be map only use of MapReduce if different usages naturally linked e.g. exploring docking of multiple chemicals or alignment of multiple DNA sequences
  – Collecting together or summarizing multiple “maps” is a simple Reduction

https://portal.futuregrid.org
**Internet of Things and the Cloud**

- It is projected that there will be 24 billion devices on the Internet by 2020. 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 multitude of 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.

- Natural parallelism over “things”
Internet of Things: Sensor Grids
A pleasingly parallel example on Clouds

• A sensor (“Thing”) is any source or sink of time series
  – In the thin client era, smart phones, Kindles, tablets, Kinects, web-cams are sensors
  – Robots, distributed instruments such as environmental measures are sensors
  – Web pages, Googledocs, Office 365, WebEx are sensors
  – Ubiquitous Cities/Homes are full of sensors
• They have IP address on Internet
• Sensors – being intrinsically distributed are Grids
• However natural implementation uses clouds to consolidate and control and collaborate with sensors
• Sensors are typically “small” and have pleasingly parallel cloud implementations

https://portal.futuregrid.org
Sensors as a Service

Output Sensor

A larger sensor ..........

https://sites.google.com/site/opensourceiotcloud/ Open Source Sensor (IoT) Cloud
Classic Parallel Computing

- **HPC**: Typically SPMD (Single Program Multiple Data) “maps” typically processing particles or mesh points interspersed with multitude of low latency messages supported by specialized networks such as Infiniband and technologies like **MPI**
  - Often run large capability jobs with 100K (going to 1.5M) cores on same job
  - National DoE/NSF/NASA facilities run 100% utilization
  - Fault fragile and cannot tolerate “outlier maps” taking longer than others

- **Clouds**: MapReduce has asynchronous maps typically processing data points with results saved to disk. Final reduce phase integrates results from different maps
  - Fault tolerant and does not require map synchronization
  - **Map only** useful special case

- **HPC + Clouds**: Iterative MapReduce caches results between “MapReduce” steps and supports SPMD parallel computing with large messages as seen in parallel kernels (linear algebra) in clustering and other data mining
### 4 Forms of MapReduce

<table>
<thead>
<tr>
<th>(a) Map Only</th>
<th>(b) Classic MapReduce</th>
<th>(c) Iterative MapReduce</th>
<th>(d) Loosely Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td><strong>Input</strong></td>
<td><strong>Input</strong></td>
<td><strong>P_{ij}</strong></td>
</tr>
<tr>
<td><strong>map</strong></td>
<td><strong>map</strong></td>
<td><strong>map</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td><strong>reduce</strong></td>
<td><strong>reduce</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BLAST Analysis</th>
<th>High Energy Physics (HEP) Histograms</th>
<th>Expectation maximization Clustering e.g. Kmeans Linear Algebra, Page Rank</th>
<th>Classic MPI PDE Solvers and particle dynamics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parametric sweep</td>
<td>Distributed search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pleasingly Parallel</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Domain of MapReduce and Iterative Extensions**

[https://portal.futuregrid.org](https://portal.futuregrid.org)
Commercial “Web 2.0” Cloud Applications

• Internet search, Social networking, e-commerce, cloud storage

• These are **larger systems than used in HPC** with huge levels of parallelism coming from
  – Processing of **lots of users** or
  – An **intrinsically parallel** Tweet or Web search

• **Classic MapReduce is suitable** (although Page Rank component of search is parallel linear algebra)

• **Data Intensive**

• Do not need microsecond messaging latency

[https://portal.futuregrid.org]
Data Intensive Applications

- **Applications** tend to be **new** and so can consider **emerging technologies** such as clouds.
- Do not have lots of small **messages** but rather **large reduction** (aka Collective) operations
  - **New optimizations** e.g. for huge messages
  - e.g. **Expectation Maximization (EM)** has a few broadcasts but dominated by reductions
- Not clearly a single exascale job but rather **many smaller** (but not sequential) jobs e.g. to analyze groups of sequences
- Algorithms not clearly robust enough to analyze lots of data
  - Current standard **algorithms** such as **R not designed for big data**
- Multidimensional Scaling MDS is iterative **rectangular matrix-matrix multiplication** controlled by EM
- Look in detail at **Deterministically Annealed Pairwise Clustering** as an EM example

[https://portal.futuregrid.org](https://portal.futuregrid.org)
Cost per Megabase of DNA Sequence

Full Personal Genomics: 3 petabytes per day

National Human Genome Research Institute

genome.gov/sequencingcosts
Intermediate step in DA-PWC
With 6 clusters
MDS used to project from high dimensional to 3D space

Each of 100K points is a sequence.
Clusters are Fungi families. 140
Clusters at end of iteration
N=100K points is $10^5$ core hours

Scales between $O(N)$ and $O(N^2)$
DA-PWC EM Steps (Expectation E is red, Maximization M Black)

k runs over clusters; i,j points

1) \[ A(k) = -0.5 \sum_{i=1}^{N} \sum_{j=1}^{N} \delta(i, j) \langle M_i(k) \rangle \langle M_j(k) \rangle / \langle C(k) \rangle^2 \]

2) \[ B_i(k) = \sum_{j=1}^{N} \delta(i, j) \langle M_j(k) \rangle / \langle C(k) \rangle \]

3) \[ \varepsilon_i(k) = (B_i(k) + A(k)) \]

4) \[ \langle M_i(k) \rangle = \exp(-\varepsilon_i(k)/T) / \sum_{k=1}^{K} \exp(-\varepsilon_i(k)/T) \]

5) \[ C(k) = \sum_{i=1}^{N} \langle M_i(k) \rangle \]

- Parallelize by distributing points i across processes
- Clusters k in simplest case are parameters held by all tasks – fails when k reaches \( \sim 10,000 \). Real challenge to automatic parallelizer!
- Either Broadcasts of \( \langle M_i(k) \rangle \) and/or reductions

\( \delta(i, j) \)
distance between points i and j
(Iterative) MapReduce structure with Map-Collective is framework

Twister runs on Linux or Azure

Twister4Azure is built on top of Azure tables, queues, storage

Judy Qiu IU will talk Thursday
MDS Azure 128 cores

- Note fluctuations limit performance
- Each step is two (blue followed by red) rectangular matrix multiplications

https://portal.futuregrid.org
MDS Azure 128 cores

- Top is weak scaling
- Bottom 128 cores, vary data size
- Twister is on non virtualized Linux
- “Adjusted” takes out sequential performance difference
What to use in Clouds: Cloud PaaS

- HDFS style **file system** to collocate data and computing
- **Queues** to manage multiple tasks
- **Tables** to track job information
- **MapReduce** and **Iterative MapReduce** to support parallelism
- **Services** for everything
- **Portals** as User Interface
- **Appliances** and **Roles** as customized images
- Software tools like **Google App Engine, memcached**
- **Workflow** to link multiple services (functions)
- **Data Parallel Languages** like Pig; more successful than HPF?

https://portal.futuregrid.org
What to use in Grids and Supercomputers?

**HPC PaaS**

- **Services Portals** and **Workflow** as in clouds
- **MPI** and **GPU/multicore threaded** parallelism
- **GridFTP** and high speed networking
- **Wonderful libraries** supporting parallel linear algebra, particle evolution, partial differential equation solution
- **Globus, Condor, SAGA, Unicore, Genesis** for Grids
- **Parallel I/O** for high performance in an application
- **Wide area File System** (e.g. Lustre) supporting file sharing
- This is a rather different style of **PaaS** from clouds – should we unify?
Is PaaS a good idea?

• If you have **existing code**, PaaS may not be very relevant immediately
  – Just need **IaaS** to put code on clouds

• But surely it must be good to offer **high level tools**?

• For example, Twister4Azure is built on top of Azure **tables, queues, storage**

• Historically **HPCC** 1990-2000 built MPI, libraries, (parallel) compilers ..

• **Grids** 2000-2010 built federation, scheduling, portals and workflow

• **Clouds** 2010-.... have a fresh interest in powerful programming models
How to use Clouds I

1) **Build the application as a service.**  Because you are deploying one or more full virtual machines and because clouds are designed to host web services, you want your application to support multiple users or, at least, a sequence of multiple executions.

   • If you are not using the application, scale down the number of servers and scale up with demand.
   • Attempting to deploy 100 VMs to run a program that executes for 10 minutes is a waste of resources because the deployment may take more than 10 minutes.
   • To minimize start up time one needs to have services running continuously ready to process the incoming demand.

2) **Build on existing cloud deployments.**  For example use an existing MapReduce deployment such as Hadoop or existing Roles and Appliances (Images)

[https://portal.futuregrid.org](https://portal.futuregrid.org)
How to use Clouds II

3) Use PaaS if possible. For platform-as-a-service clouds like Azure use the tools that are provided such as queues, web and worker roles and blob, table and SQL storage.

3) Note HPC systems don’t offer much in PaaS area

4) Design for failure. Applications that are services that run forever will experience failures. The cloud has mechanisms that automatically recover lost resources, but the application needs to be designed to be fault tolerant.

- In particular, environments like MapReduce (Hadoop, Daytona, Twister4Azure) will automatically recover many explicit failures and adopt scheduling strategies that recover performance "failures" from for example delayed tasks.
- One expects an increasing number of such Platform features to be offered by clouds and users will still need to program in a fashion that allows task failures but be rewarded by environments that transparently cope with these failures. (Need to build more such robust environments)

https://portal.futuregrid.org
How to use Clouds III

5) **Use as a Service where possible.** Capabilities such as SQLaaS (database as a service or a database appliance) provide a friendlier approach than the traditional non-cloud approach exemplified by installing MySQL on the local disk.
   - Suggest that many prepackaged aaS capabilities such as *Workflow as a Service* for eScience will be developed and simplify the development of sophisticated applications.

6) **Moving Data is a challenge.** The general rule is that one should move computation to the data, but if the only computational resource available is a the cloud, you are stuck if the data is not also there.
   - Persuade Cloud Vendor to host your data free in cloud
   - Persuade Internet2 to provide good link to Cloud
   - Decide on Object Store v. HDFS style (or v. Lustre WAFS on HPC)
aaS versus Roles/Appliances

• If you package a capability X as XaaS, it runs on a separate VM and you interact with messages
  – SQLaaS offers databases via messages similar to old JDBC model
• If you build a role or appliance with X, then X built into VM and you just need to add your own code and run
  – Generalized worker role builds in I/O and scheduling
• Lets take all capabilities – MPI, MapReduce, Workflow .. – and offer as roles or aaS (or both)
• Perhaps workflow has a controller aaS with graphical design tool while runtime packaged in a role?
• Need to think through packaging of parallelism

https://portal.futuregrid.org
Private Clouds

• Define as non commercial cloud used to support science

• **What does it take to make private cloud platforms competitive with commercial systems?**

• Plenty of work at **VM management level** with Eucalyptus, Nimbus, OpenNebula, OpenStack
  – Only now maturing
  – Nimbus and OpenNebula pretty solid but not widely adopted in USA
  – OpenStack and Eucalyptus recent major improvements

• **Open source PaaS** tools like Hadoop, Hbase, Cassandra, Zookeeper but not integrated into platform

• **Need dynamic resource** management in a “not really elastic” environment as limited size

• **Federation** of distributed components (as in grids) to make a decent size system
Architecture of Data Repositories?

• Traditionally governments set up repositories for data associated with particular missions
  – For example EOSDIS (Earth Observation), GenBank (Genomics), NSIDC (Polar science), IPAC (Infrared astronomy)
  – LHC/OSG computing grids for particle physics

• This is complicated by volume of data deluge, distributed instruments as in gene sequencers (maybe centralize?) and need for intense computing like Blast
  – i.e. repositories need lots of computing?
Clouds as Support for Data Repositories?

• The **data deluge** needs cost effective computing
  – Clouds are by definition cheapest
  – Need data and computing co-located

• **Shared resources** essential (to be cost effective and large)
  – Can’t have every scientists downloading petabytes to personal cluster

• Need to reconcile **distributed** (initial source of ) **data** with shared analysis
  – Can move data to (discipline specific) clouds
  – How do you deal with multi-disciplinary studies

• **Data repositories of future will have cheap data and elastic cloud analysis support?**
  – Hosted free if data can be used commercially?
Using Science Clouds in a Nutshell

• High Throughput Computing; pleasingly parallel; grid applications
• Multiple users (long tail of science) and usages (parameter searches)
• Internet of Things (Sensor nets) as in cloud support of smart phones
• (Iterative) MapReduce including “most” data analysis
• Exploiting elasticity and platforms (HDFS, Object Stores, Queues ..)
• Use worker roles, services, portals (gateways) and workflow
• Good Strategies:
  – 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;
  – Address Challenge of Moving Data

https://portal.futuregrid.org
Cosmic Comments I

• Does Cloud + MPI Engine for computing + grids for data cover all?
  – Will current high throughput computing and cloud concepts merge?

• Need interoperable data analytics libraries for HPC and Clouds

• Can we characterize data analytics applications?
  – I said modest size and kernels need reduction operations and are often full matrix linear algebra (true?)

• Does a “modest-size private science cloud” make sense
  – Too small to be elastic?

• Should governments fund use of commercial clouds (or build their own)
  – Most science doesn’t have privacy issues motivating private clouds

• Most interest in clouds from “new” applications such as life sciences
Cosmic Comments II

• Recent **private cloud infrastructure** (Eucalyptus 3, OpenStack Essex in USA) much improved
  – Nimbus, OpenNebula still good
• But are they really competitive with **commercial cloud fabric runtime**?
• Should we **integrate HPC and Cloud Platforms**?
• More **employment opportunities** in clouds than HPC and Grids; so cloud related activities popular with students
• Science Cloud Summer School July 30-August 3
  – Part of virtual summer school in computational science and engineering and expect over 200 participants spread over 9 sites
• Science Cloud and MapReduce XSEDE Community groups

https://portal.futuregrid.org