Cloud computing and MTC paradigm

Salman Toor

salman.toor@it.uu.se
PART-I
Computing model

- Most of the large scale applications both from academia and industry were designed for batch processing.

- Batch Processing:

  A complete set of batch or group of instructions together with the required input data to accomplish a given task (often known as job). No user interaction is possible during the execution.
Distributed Computing Infrastructures (DCI)

- Cluster Computing
  - Accessible via Local Area Network (LAN)
- Grid Computing
  - Based on Wide Area Network (WAN)
- Cloud Computing

- Desktop Computing
- Utility Computing
- P2P Computing
- Pervasive Computing
- Ubiquitous Computing
- Mobile Computing
Cluster computing

http://www.wikid.eu/index.php/Computer_Clustering
Cluster computing

- Known Softwares of Cluster computing:
  - HTCondor
  - Portable Batch System (PBS)
  - Load Sharing Facility (LSF)
  - Simple Linux Utility for Resource Management (SLRM)
  - Rocks
  - ....
  - ....
Cluster computing
Disadvantages

- Applications need to adopt the way underlying infrastructure is designed
- Cluster softwares are non-coherent
- Steep learning curve
- Less secure (improved significantly over the years)
- Tightly coupled with the underlying resources
- Difficult to port new applications
- Applications need to stick with the available tools and libraries
- Non standard interfaces
GRID COMPUTING

• Definition - 1 : (Computational Grid)

Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.

• Definition - 2 : (Computational Power Grids)

The computational power grid is analogous to electric power grid and it allows to couple geographically distributed resources and offer a consistent and inexpensive access to resources irrespective of their physical location or access point.

The anatomy of the grid: Enabling scalable virtual organizations
The Grid 2: Blueprint for a new computing infrastructure
http://www.gridcomputing.com/gridfaq.html
Grid computing

Current status

- Advanced Resource connector (ARC)
Grid Computing Disadvantages

- Complex system architecture
- Steep learning curve for the end user
- Only allow batch processing, zero level interactivity
- Difficult to attach a comprehensive economic model
- The sites are autonomous but the softwares are tightly connected with the underlying hardware
- Mostly available for academic and research activities
- Lack of standard interface
- Static availability of resources
Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Cloud offerings:

IaaS, PaaS, SaaS, NaaS …
IaaS

• Infrastructure-as-a-Service (IaaS)

• The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications.

• Aims:
  • Transparent access to the **resources**
  • Easy to access
  • Pay-as-you-go model

PaaS

• Plateform-as-a-Service (PaaS)

• The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.

• Aims:
  • Transparent access via IaaS
  • Easy to manage
  • Pay-as-you-go model

SaaS

• Software-as-a-Service (SaaS)

• The capability provided to the consumer is to use the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface.

• Aims:
  • Transparent access via PaaS
  • Easy to manage
  • Pay-as-you-go model

Computing model

- Together with batch processing, Cloud computing model provides interactive processing of complex applications

Wikipedia:

Interactive computing refers to software which accepts input from humans — for example, data or commands.

- Frameworks like; IPython or Jupyter notebooks extend web technologies for interactive computing
Cloud 1.0

Cloud Services

Compute Servers

Storage

Your Computer

Internet
Cloud 2.0
Cloud 3.0
Serverless architecture and smart services

- Amazon Lambda
- Google Cloud Functions
- Azure Functions
- OpenFaaS
PART-II
Virtualization
Basic illustration

Virtualization

• Virtualization Layer

Hypervisor or Virtual Machine Monitor (VMM) is a software that provides an interface between hardware and virtual operating systems.

• Types of Hypervisors

  • Bare-Metal
  • Hosted
Virtualization and Clouds

OpenStack

• Open source platform for build public and private Clouds

http://www.openstack.org/
DOES VIRTUALIZATION EFFECT THE SYSTEM PERFORMANCE?
Performance

- Yes, performance loss may occur but it is highly dependent on
  - Type of virtualization layer (Hypervisor)
  - Use case
- CPU bound application will perform differently than IO bound or network intensive applications
In comparison with the physical node:
- KVM perform 83.46%
- Xen perform 97.28%

Reason; Critical instruction test verses para-virtualization

Figure 1. LINPACK results of virtulized Fedora8 and XP in Xen, KVM and native Fedora8

In both cases, There is a performance different compare to physical machine.

Contextualization

In cloud computing contextualization means providing customized computing environment.

Or

Allows a virtual machine instance to learn about its cloud environment and user requirement (the ‘context’) and configure itself to run correctly.

http://www.journalofcloudcomputing.com/content/4/1/17
Orchestration

• Orchestration is a process of resource contextualization based on the automation available in the cloud systems.

• A process required for
  • rapid application deployment
  • scalability
  • management
  • high availability
  • Agility

• Essential for large complex applications

• A process at the level of Platform as a Service (PaaS)
HPC in Clouds

Helping engineers run their simulations on powerful cloud hardware.

Run your favorite simulation software

Get instant access to your favorite CAE software on powerful, secure industry leading cloud hardware.

- ANSYS, Siemens, CST, Abaqus, NUMECA, OpenFOAM and many more.
- Pick a license option that works for you: short-term, on-demand, or use your own.
- Nothing to install, nothing to setup – simple, ready to use, business as usual.

Learn More
PART-III
“I need my application to scale”

- What do we mean with a scalable CSE application?
Tightly coupled applications?

- Common in CSE, typical examples are Partial Differential Equation (PDE) solvers.
HPC programming models

• Message Passing Interface (Distributed Memory)
• OpenMP, Pthreads etc. (Shared Memory)
• CUDA, OpenCL (GPGPU)
• FGPA

- Low-latency interprocess communication is critical to application performance
- Specialized hardware/instruction sets.
- “Every cycle counts”, users of HPC become concerned about VM overhead
- Failed tasks causes major problems/performance losses
- Compute infrastructure organized accordingly
- ’High-Performance Computing, HPC cluster resources. Wait in a queue to obtain dedicated direct access to high-bandwidth hardware resources.
High-Throughput Computing (HTC)

While HPC is typically aiming to use a large amount of resources to solve few large, tightly coupled tasks in a short period of time on homogenous and co-localized resources, HTC is typically concerned with

- Solving a large number of (large) problems over long periods of time (days, months, years)
- on distributed, heterogenous resources
- Grid Computing developed largely for this application class
- Task throughput measured over months to years

http://research.cs.wisc.edu/htcondor/htc.html
Many-Task Computing (MTC)

Bridges the gap between HPC and HTC

- Solve a large number of loosely coupled tasks on a large number of resources in a fairly short time
- Can be a mix of independent and dependent tasks
- Throughput typically measured in tasks/s, MB/s etc.
- Often not so clear distinction between HTC and MTC
Cloud Computing

- Can be very good for MTC problems
- Capability to “burst” and dynamically provide large resources over relatively short periods of time, then scale down.
- It is questionable if (public) clouds would provide a economically feasible model for HTC. Combination of private and public can give best of two worlds.
- HPC (currently) benefits from specialized offerings
“Cloud computing doesn’t scale”

When you hear those types of statements, they typically refer to conceptions about traditional communication intensive HPC applications, typically implemented with MPI.

- Refers to low-performing network between instances in Public Clouds
- Does not typically account for tailored HPC cluster offerings and placement groups (available in e.g. EC2)

 Represents a very narrow view on what applications and programming models are admissible.

StarCluster deploys MPI-clusters over AWS:
http://star.mit.edu/cluster/
Vertical vs. Horizontal Scaling

**Vertical scaling ("scale-up"):**
Increase the capacity of your servers to meet application demand.
- Add more vCPUs
- Add more RAM/storage

**Horizontal scaling ("scale-out"):**
Meet increased application demand by adding more servers/storage/etc.
How to measure performance/efficiency?

• Traditional HPC: FLOPs, execution time
• Recent HPC/Multicore trend: Energy efficiency (green computing)
• Cloud computing: Same, but in addition cost due to the pay-as-you-go model.
  • “What does my computational experiment cost?”
  • Non-trivial to measure/project
  • Depends on CPU/hours, Number of API requests, Write/Read OPs on Volumes, GB traffic into and out of Object Store etc…
Strive for statelessness of task/component

Ideally you want your task to be able to execute anywhere, at any time.

- Robustness
- Can increase, decrease number of workers

Data, task input/output

"Growing is easier than shrinking"

But of course, all computations/applications need data and state. It is not so much about if you store it as it is about where you store it.
Is the controller machine designed for scalable storage?

Network a potential performance bottleneck

High pressure on critical component (robustness)
Use the object store when you can

Key-value storage, designed for horizontal scaling

Why even store task information on controller?
Performance (latency) is one good reason

Robustness/scalability through replication.

In this kind of master-slave setup the controller will still maintain a lot of information, route tasks etc.

S3, Swift
Use the object store when you can

Decouple the life-time of compute servers and data!
- Needed for elasticity
- Servers are cattle
- Data can be precious
Another option for robustness, store replicas of data over multiple hosts in a distributed filesystem.

- HDFS (Hadoop, Big Data)
- Does not decouple compute/storage lifetimes as easily
- Designed for extreme I/O throughput
AsynchronousTask

Aim to execute tasks in non-blocking mode, and have as few global synchronization points as possible.
QTL as a Service (QTLaaS), a cloud service for genetic analysis
QTL Analysis

• Abbreviation of “Quantitative Trait Loci”

• Understanding the relation between genes and traits is a fundamental problem in genetics.

• Such knowledge can eventually lead to e.g. the identification of possible:
  • drug targets,
  • treatment of heritable diseases, and
  • efficient designs for plant and animal breeding.

• QTL mapping is the way to locate such regions in genome.
A flexible computational framework using R and Map-Reduce for permutation tests of massive genetic analysis of complex traits

Citation information: DOI 10.1109/TCBB.2016.2527639, IEEE/ACM Transactions on Computational Biology and Bioinformatics

Authors: Behrang Mahjani, Salman Toor, Carl Nettelblad, Sverker Holmgren
Analysis Framework

Fig. 4: Distributed computing of the significance level using map-reduce setting; Details of block 4
Results

**TABLE 1:** $10^4$ permutations, PruneDIRECT is using the global minimum from true QTL

<table>
<thead>
<tr>
<th>Total time (hours)</th>
<th>Exhuastive</th>
<th>DIRECT</th>
<th>PruneDIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>70</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>Average of FE.*</td>
<td>531441</td>
<td>16867</td>
<td>63</td>
</tr>
<tr>
<td>Standard division of FE.*</td>
<td>0</td>
<td>2239</td>
<td>1555</td>
</tr>
</tbody>
</table>

*FE. is the number of function evaluations.

**TABLE 2:** $2 \times 10^5$ permutations, PruneDIRECT is using the global minimum from true QTL

<table>
<thead>
<tr>
<th>Total time (hours)</th>
<th>PruneDIRECT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td>Average of FE.*</td>
<td>62</td>
</tr>
<tr>
<td>Standard division of FE.*</td>
<td>1650</td>
</tr>
</tbody>
</table>

*FE. is the number of function evaluations.

**TABLE 3: 640 permutations to analyze the scalability of the proposed framework.**

<table>
<thead>
<tr>
<th>Number of virtual CPUs</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total time in hours</td>
<td>39.9</td>
<td>11.02</td>
<td>5.83</td>
<td>2.88</td>
<td>1.50</td>
</tr>
</tbody>
</table>