Multicore and Cloud Futures

CCGSC
September 15 2008

Geoffrey Fox
Community Grids Laboratory, School of informatics
Indiana University

gcf@indiana.edu, http://www.infomall.org
Gartner 2008 Technology Hype Curve

Clouds, Microblogs and Green IT appear
Basic Web Services, Wikis and SOA becoming mainstream

MPI way out on plateau?

Grids?

Source: Gartner (July 2008)
Gartner 2006 Technology Hype Curve

Grids did exist
Grids become Clouds

- Grids solve problem of \textit{too little computing}: We need to harness all the world’s computers to do Science.

- Clouds solve the problem of \textit{too much computing}: with \textit{multicore} we have so much power that we need to make usage much easier.

- Key technology: \textit{Virtual Machines} (dynamic deployment) enable more dynamic flexible environments
  - Is \textit{Virtual Cluster} or \textit{Virtual Machine} correct primitive?

- \textit{Data Grids} seem fine as data naturally distributed.

- GGF/EGA false assumption: \textit{Web 2.0} not Enterprise defined commercial software stack
  - Some Web 2.0 applications (\textit{MapReduce}) not so different from data-deluged eScience.

- \textit{Citizen Science} requires light weight friendly Cyberinfrastructure.
MPI on Nimbus for clustering

- Note fluctuations in runtime but performance OK for large enough problems
- 8 Nodes

Figure 3. Histogram of Kmeans clustering time for 100 iterations (X=100) of the figure 2

Table 3. MPI Time for 100 iterations (X = 100) of the figure 2. Calculated using the following formula:

\[
\text{MPI Time} = \frac{T(100) - T(1)}{99}
\]

<table>
<thead>
<tr>
<th>Graph</th>
<th>MPI Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM MIN</td>
<td>0.040</td>
</tr>
<tr>
<td>VM Average</td>
<td>0.234</td>
</tr>
<tr>
<td>VM MAX</td>
<td>0.112</td>
</tr>
<tr>
<td>Direct Average</td>
<td>0.014</td>
</tr>
</tbody>
</table>
Plans for QuakeSpace

- QuakeSim supports Earthquake Scientists who want some features of their kid’s (under 40) world
- Rebuild QuakeSim using Web 2.0 and Cloud Technology
- Applications, Sensors, Data Repositories as Services
- Computing via Clouds
- Portals as Gadgets
- Metadata by tagging
- Data sharing as in YouTube
- Alerts by RSS
- Virtual Organizations via Social Networking
- Workflow by Mashups
- Performance by multicore
- Interfaces via iPhone, Android etc.
Top 50 tags for Workflows

- abstracts
- affymetrix
- AIDA
- BioAID
- bioinformatics
- biorange_nl
- BLAST
- cel
- clone
- count
- data-driven
dbg
- disease
tenrez
- file
- gene
- genotype
- iteration
- kegg
- list
- mesh
- microarray
- mining
- mouse
- pathway
- pathway-driven
- pathways
- phenotype
- protein
- pubmed
- qtl
- rank
- scaffold
- SEG
- sequence
- shim
- similarity
- simplifier
- synonyms
- terms
- text mining
text mining network
- ugi
- uniprot
- unitary matrix
- up-and-down
- utility
- VL-e
- weighting

Most Recent  Last Updated  Most Viewed  Most Downloaded
<table>
<thead>
<tr>
<th>Enterprise Approach</th>
<th>Web 2.0 Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSR 168 Portlets</td>
<td>Gadgets, Widgets</td>
</tr>
<tr>
<td>Server-side integration and processing</td>
<td>AJAX, client-side integration and processing, JavaScript</td>
</tr>
<tr>
<td>SOAP</td>
<td>RSS, Atom, JSON</td>
</tr>
<tr>
<td>WSDL</td>
<td>REST (GET, PUT, DELETE, POST)</td>
</tr>
<tr>
<td>Portlet Containers</td>
<td>Open Social Containers (Orkut, LinkedIn, Shindig); Facebook; StartPages</td>
</tr>
<tr>
<td>User Centric Gateways</td>
<td>Social Networking Portals</td>
</tr>
<tr>
<td>Workflow managers (Taverna, Kepler, etc)</td>
<td>Mash-ups</td>
</tr>
<tr>
<td>Grid computing: Globus, Condor, etc</td>
<td>Cloud computing: Amazon WS Suite, Xen Virtualization, still Condor!</td>
</tr>
</tbody>
</table>
Different Programming Models

- (Web) services, "farm" computations, Workflow (including AVS, HeNCE from past), Mashups, MPI, MapReduce run **functionally or data decomposed execution units** with a wide variety of front ends

- **Front-end**: Language+communication library, Scripting, Visual, Functional, XML, PGAS, HPCS Parallel Languages, Templates, OpenMP

- Synchronize/Communicate with some variant of messaging (zero size for locks) with performance, flexibility, fault-tolerance, dynamism trade-offs

- **Synchronization**: Locks Threads Processes CCR CCI SOAP REST MPI Hadoop; not much difference for user?
MPI becomes Ghetto MPI

- **Multicore** best practice not **messaging** will drive synchronization/communication primitives

- Party Line Programming Model: **Workflow (parallel--distributed)** controlling optimized library calls
  - Core parallel implementations no easier than before; deployment is easier

- MPI is wonderful; it will be ignored in real world unless simplified

- CCI notes MPI is **HPCC Ghetto**

- CCI is high performance distributed message passing ghetto?

- **CCR** from Microsoft – only ~7 primitives – is one possible commodity multicore driver
  - It is roughly active messages
  - Will run MPI style codes fine on multicore
Deterministic Annealing Clustering
Scaled Speedup Tests on 4 8-core Systems

1,600,000 points per C# thread

Parallel Overhead = \( \frac{PT(P)}{T(1)} - 1 \)
On P processors = \( \frac{1}{\text{efficiency}} - 1 \)

Nodes

| 1 | 2 | 1 | 1 | 4 | 2 | 1 | 2 | 1 | 1 | 4 | 2 | 1 | 2 | 1 | 1 | 4 | 2 | 1 | 2 | 1 | 1 | 4 | 2 | 4 | 2 | 4 | 2 | 4 | 2 | 4 | 2 | 4 | 4 | 4 | 4 | 4 |

MPI Processes per Node

| 1 | 1 | 2 | 1 | 1 | 2 | 4 | 1 | 2 | 1 | 2 | 4 | 8 | 1 | 2 | 4 | 1 | 2 | 1 | 4 | 8 | 2 | 4 | 1 | 2 | 1 | 8 | 4 | 2 | 1 |

CCR Threads per Process

| 1 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 4 | 1 | 1 | 1 | 2 | 2 | 2 | 4 | 4 | 4 | 8 | 1 | 1 | 2 | 2 | 4 | 4 | 4 | 8 | 1 | 2 | 4 | 8 |
MPI, MapReduce, Java Threads for Clustering

- Overhead PT(P)/T(1)-1 of the messaging runtime for the different data sizes
- All perform well for large enough datasets
Hadoop v MPI and faster MapReduce for Clustering

Factor of 10^3

Factor of 30

Total Kmeans Time (seconds)

Number of Data Points

HADOOP

In memory MapReduce

MPI

Cluster of 5 machines with Dual Intel Clovertown Processors (2.3GHz)
Total of 40 processor cores
8 GB of memory in each machine

MPI 40 Processes
CGL MapReduce 40 map tasks
Hadoop 40 map tasks
N=3000 sequences each length ~1000 features
Only use pairwise distances
will repeat with 0.1 to `0.5 million sequences with a larger machine
C# with CCR and MPI