Managing and Executing Loosely-Coupled Large-Scale Applications on Clusters, Grids, and Supercomputers

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Collaborators:
Ian Foster (UC/CI/ANL), Yong Zhao (MS), Mike Wilde (CI/ANL), Zhao Zhang (CI), Alex Szalay (JHU), Jerry Yan (NASA/ARC), Catalin Dumitrescu (FANL), many others from Swift and Falkon teams

GlobusWorld08
May 14th, 2008
What will we do with 1+ Exaflops and 1M+ cores?
1) Tackle **Bigger and Bigger Problems**

Computational Scientist as Hero
2) Tackle **Increasingly Complex Problems**

Computational Scientist as Logistics Officer
Problem Types

- **Input data size**:
  - Hi
  - Med
  - Lo

- **Number of tasks**:
  - 1
  - 1K
  - 1M

- **Types**:
  - Data analysis, mining
  - Big data and many tasks
  - Many loosely coupled tasks
  - Heroic MPI tasks

5/19/2008  Managing and Executing Loosely-Coupled Large-Scale Applications on Clusters, Grids, and Supercomputers
Challenge #1: Long Queue Times

- Wait queue times are typically longer than the job duration times.

<table>
<thead>
<tr>
<th></th>
<th>Queue Wait Time (sec)</th>
<th>Job Run Time (sec)</th>
<th>Queue Wait Time %</th>
<th>Job Run Time %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Average</td>
<td>27,620</td>
<td>6,456</td>
<td>59%</td>
<td>41%</td>
</tr>
<tr>
<td>Median</td>
<td>597</td>
<td>607</td>
<td>73%</td>
<td>27%</td>
</tr>
<tr>
<td>Maximum</td>
<td>2,656,696</td>
<td>141,476</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>84,204</td>
<td>13,630</td>
<td>38%</td>
<td>38%</td>
</tr>
</tbody>
</table>
Challenge #2: Slow Job Dispatch Rates

- Production LRMs → ~1 job/sec dispatch rates
- What job durations are needed for 90% efficiency:
  - Production LRMs: 900 sec
  - Development LRMs: 100 sec
  - Experimental LRMs: 50 sec
  - 1~10 sec should be possible

<table>
<thead>
<tr>
<th>System</th>
<th>Comments</th>
<th>Throughput (tasks/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condor (v6.7.2) - Production</td>
<td>Dual Xeon 2.4GHz, 4GB</td>
<td>0.49</td>
</tr>
<tr>
<td>PBS (v2.1.8) - Production</td>
<td>Dual Xeon 2.4GHz, 4GB</td>
<td>0.45</td>
</tr>
<tr>
<td>Condor (v6.7.2) - Production</td>
<td>Quad Xeon 3 GHz, 4GB</td>
<td>2</td>
</tr>
<tr>
<td>Condor (v6.8.2) - Production</td>
<td></td>
<td>0.42</td>
</tr>
<tr>
<td>Condor (v6.9.3) - Development</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>Condor-J2 - Experimental</td>
<td>Quad Xeon 3 GHz, 4GB</td>
<td>22</td>
</tr>
</tbody>
</table>
Challenge #3: Poor Scalability of Shared File Systems

- **GPFS vs. LOCAL**
  - **Read Throughput**
    - 1 node: 0.48Gb/s vs. 1.03Gb/s \(\Rightarrow 2.15x\)
    - 160 nodes: 3.4Gb/s vs. 165Gb/s \(\Rightarrow 48x\)
  - **Read+Write Throughput**:
    - 1 node: 0.2Gb/s vs. 0.39Gb/s \(\Rightarrow 1.95x\)
    - 160 nodes: 1.1Gb/s vs. 62Gb/s \(\Rightarrow 55x\)
  - **Metadata (mkdir / rm -rf)**
    - 1 node: 151/sec vs. 199/sec \(\Rightarrow 1.3x\)
    - 160 nodes: 21/sec vs. 31840/sec \(\Rightarrow 1516x\)
Hypothesis

“Significant performance improvements can be obtained in the analysis of large dataset by leveraging information about data analysis workloads rather than individual data analysis tasks.”

• Important concepts related to the hypothesis
  – **Workload**: a complex query (or set of queries) decomposable into simpler tasks to answer broader analysis questions
  – **Data locality** is crucial to the efficient use of large scale distributed systems for scientific and data-intensive applications
  – Allocate computational and caching storage resources, **co-scheduled** to optimize workload performance
Falkon: a Fast and Light-weight task execution framework

- **Goal:** enable the *rapid and efficient* execution of many independent jobs on large compute clusters

- Combines three components:
  - a *streamlined task dispatcher* able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers ➔ Challenge #1
  - *resource provisioning* through multi-level scheduling techniques ➔ Challenge #2
  - *data diffusion* and data-aware scheduling to leverage the co-located computational and storage resources ➔ Challenge #3
Falkon Overview

[Diagram of Falkon system]

User -> Task Dispatcher
Data-Aware Scheduler
Persistent Storage
Wait Queue
Dynamic Resource Provisioning
Available Resources (GRAM4)
Provisioned Resources
Executor
Executor
Executor
• Fast:
  – Up to 3700 tasks/sec

• Scalable:
  – 54,000 processors
  – 1,500,000 tasks queued

• Efficient:
  – High efficiency with second long tasks on 1000s of processors

Dispatcher Throughput
Falkon Integration with Swift

<table>
<thead>
<tr>
<th>Application</th>
<th>#Tasks/workflow</th>
<th>#Stages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATLAS: High Energy Physics Event Simulation</td>
<td>500K</td>
<td>1</td>
</tr>
<tr>
<td>fMRI DBIC: AIRSN Image Processing</td>
<td>100s</td>
<td>12</td>
</tr>
<tr>
<td>FOAM: Ocean/Atmosphere Model</td>
<td>2000</td>
<td>3</td>
</tr>
<tr>
<td>GADU: Genomics</td>
<td>40K</td>
<td>4</td>
</tr>
<tr>
<td>HNL: fMRI Aphasia Study</td>
<td>500</td>
<td>4</td>
</tr>
<tr>
<td>NVO/NASA: Photorealistic Montage/Morphology</td>
<td>1000s</td>
<td>16</td>
</tr>
<tr>
<td>QuarkNet/I2U2: Physics Science Education</td>
<td>10s</td>
<td>3 – 6</td>
</tr>
<tr>
<td>RadCAD: Radiology Classifier Training</td>
<td>1000s</td>
<td>5</td>
</tr>
<tr>
<td>SIDGrid: EEG Wavelet Processing, Gaze Analysis</td>
<td>100s</td>
<td>20</td>
</tr>
<tr>
<td>SDSS: Coadd, Cluster Search</td>
<td>40K, 500K</td>
<td>2, 8</td>
</tr>
<tr>
<td>SDSS: Stacking, AstroPortal</td>
<td>10Ks ~ 100Ks</td>
<td>2 ~ 4</td>
</tr>
<tr>
<td>MolDyn: Molecular Dynamics</td>
<td>1Ks ~ 20Ks</td>
<td>8</td>
</tr>
<tr>
<td>MARS: Economic Modeling</td>
<td>1M~1B</td>
<td>1</td>
</tr>
<tr>
<td>DOCK: Molecular Dynamics</td>
<td>1B</td>
<td>1</td>
</tr>
</tbody>
</table>

Swift Architecture

Specifying

SwiftScript Compiler

Abstract computation

Execution Engine (Karajan w/ Swift Runtime)

Virtual Node(s)

Status reporting

Provenance collector

SwiftScript

Provenance data

file1

file2

file3

Scheduling

Execution

Provisioning

Falkon Resource Provisioner

Amazon EC2

Open Science Grid
Functional MRI (fMRI)

- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies
fMRI Application

- GRAM vs. Falkon: 85%~90% lower run time
- GRAM/Clustering vs. Falkon: 40%~74% lower run time
Managing and Executing Loosely-Coupled Large-Scale Applications on Clusters, Grids, and Supercomputers

5/19/2008

B. Berriman, J. Good (Caltech)
J. Jacob, D. Katz (JPL)
Montage Application

- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound
MolDyn Application

- 244 molecules → 20497 jobs
- 15091 seconds on 216 CPUs → 867.1 CPU hours
- Efficiency: 99.8%
- Speedup: $206.9x \rightarrow 8.2x$ faster than GRAM/PBS
- 50 molecules w/ GRAM (4201 jobs) → 25.3 speedup
MARS Economic Modeling on IBM BG/P

- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%
Many Many Tasks: Identifying Potential Drug Targets

200+ Protein target(s) x 5M+ ligands

(Mike Kubal, Benoit Roux, and others)
DOCK on SiCortex

- CPU cores: 5760
- Tasks: 92160
- Elapsed time: 12821 sec
- Compute time: 1.94 CPU years
- Average task time: 660.3 sec
- Speedup: 5650X (ideal 5760)
- Efficiency: 98.2%
Falkon: Data Diffusion

- Resource acquired in response to demand
- Data and applications diffuse from archival storage to newly acquired resources
- Resource “caching” allows faster responses to subsequent requests
  - Cache Eviction Strategies: RANDOM, FIFO, LRU, LFU
- Resources are released when demand drops
AstroPortal Stacking Service

- **Purpose**
  - On-demand “stacks” of random locations within ~10TB dataset

- **Challenge**
  - Rapid access to 10-10K “random” files
  - Time-varying load

- **Sample Workloads**

<table>
<thead>
<tr>
<th>Locality</th>
<th>Number of Objects</th>
<th>Number of Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>111700</td>
<td>111700</td>
</tr>
<tr>
<td>1.38</td>
<td>154345</td>
<td>111699</td>
</tr>
<tr>
<td>2</td>
<td>97999</td>
<td>49000</td>
</tr>
<tr>
<td>3</td>
<td>88857</td>
<td>29620</td>
</tr>
<tr>
<td>4</td>
<td>76575</td>
<td>19145</td>
</tr>
<tr>
<td>5</td>
<td>60590</td>
<td>12120</td>
</tr>
<tr>
<td>10</td>
<td>46480</td>
<td>4650</td>
</tr>
<tr>
<td>20</td>
<td>40460</td>
<td>2025</td>
</tr>
<tr>
<td>30</td>
<td>23695</td>
<td>790</td>
</tr>
</tbody>
</table>

5/19/2008  Managing and Executing Loosely-Coupled Large-Scale Applications on Clusters, Grids, and Supercomputers
AstroPortal Stacking Service with Data Diffusion

Low data locality ➔
- Similar (but better) performance to GPFS

High data locality ➙
- Near perfect scalability
AstroPortal Stacking Service with Data Diffusion

- **Aggregate throughput:**
  - 39Gb/s
  - 10X higher than GPFS
- **Reduced load on GPFS**
  - 0.49Gb/s
  - 1/10 of the original load

- **Big performance gains as locality increases**
Data Diffusion: Data-Intensive Workload

- 250K tasks on 128 processors
  - 10MB read, 10ms compute
- Comparing GPFS with data diffusion
Data Diffusion: Data-Intensive Workload

Throughput:
- Average: 14Gb/s vs 4Gb/s
- Peak: 100Gb/s vs. 6Gb/s

Response Time ➔
- 3 sec vs 1569 sec ➔ 506X

5/19/2008 Managing and Executing Loosely Coupled Large-Scale Applications on Clusters, Grids, and Supercomputers
Hadoop vs. Swift

- Classic benchmarks for MapReduce
  - Word Count
  - Sort
- Swift performs similar or better than Hadoop (on 32 processors)
Mythbusting

- **Embarrassingly** Happily parallel apps are trivial to run
  - Logistical problems can be tremendous
- Loosely coupled apps do not require “supercomputers”
  - Total computational requirements can be enormous
  - Individual tasks may be tightly coupled
  - Workloads frequently involve large amounts of I/O
- Loosely coupled apps do not require specialized system software
- Shared file systems are good all around solutions
  - They don’t scale proportionally with the compute resources
Solutions

- **Falkon**
  - A Fast and Light-weight task execution framework
  - Globus Incubator Project
  - [http://dev.globus.org/wiki/Incubator/Falkon](http://dev.globus.org/wiki/Incubator/Falkon)

- **Swift**
  - Parallel programming tool for rapid and reliable specification, execution, and management of large-scale science workflows
  - [http://www.ci.uchicago.edu/swift/index.php](http://www.ci.uchicago.edu/swift/index.php)

- **Environments:**
  - *Clusters:* TeraPort (TP)
  - *Grids:* Open Science Grid (OSG), TeraGrid (TG)
  - *Specialized large machines:* SiCortex 5732
  - *Supercomputers:* IBM BlueGene/P (BG/P)
More Information

- More information:
  - Personal research page: http://people.cs.uchicago.edu/~iraicu/
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - Swift: http://www.ci.uchicago.edu/swift/index.php

- Collaborators (relevant to this proposal):
  - Ian Foster, The University of Chicago & Argonne National Laboratory
  - Alex Szalay, The Johns Hopkins University
  - Yong Zhao, Microsoft
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
  - Catalin Dumitrescu, Fermi National Laboratory
  - Zhao Zhang, The University of Chicago
  - Jerry C. Yan, NASA, Ames Research Center

- Funding:
  - NASA: Ames Research Center, Graduate Student Research Program (GSRP)
  - NSF: TeraGrid