Many-Task Computing

Ioan Raicu
Computer Science Department
Illinois Institute of Technology

CS 595
Hot Topics in Distributed Systems: Data-Intensive Computing
September 8th, 2010
HTC: High-Throughput Computing

- Typically applied in clusters and grids
- Loosely-coupled applications with sequential jobs
- Large amounts of computing for long periods of times
- Measured in operations per month or years
• Bridge the gap between HPC and HTC  
• Applied in clusters, grids, and supercomputers  
• Loosely coupled apps with HPC orientations  
• Many activities coupled by file system ops  
• Many resources over short time periods  
  – Large number of tasks, large quantity of computing, and large volumes of data
Problem Space

Input Data Size

Hi

Med

Low

Number of Tasks

1

1K

1M

MapReduce/MTC
(Data Analysis, Mining)

MTC
(Big Data and Many Tasks)

HPC
(Heroic MPI Tasks)

HTC/MTC
(Many Loosely Coupled Tasks)

[MTAGS08] “Many-Task Computing for Grids and Supercomputers”
• **Goal**: enable the *rapid and efficient* execution of many independent jobs on large compute clusters.

• Combines three components:
  - a *streamlined task dispatcher*
  - *resource provisioning* through multi-level scheduling techniques
  - *data diffusion* and data-aware scheduling to leverage the co-located computational resources

• Integration into Swift:
  - Applications cover many domains: astronomy, astro-physics, medicine, chemistry, economics, climate modeling, etc.

[SciDAC09] “Extreme-scale scripting: Opportunities for large task-parallel applications on petascale systems”
[SC07] “Falkon: a Fast and Light-weight task executiON framework”
[SWF07] “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”
Falkon Project

- Falkon is a real system
  - Late 2005: Initial prototype, AstroPortal
  - January 2007: Falkon v0
  - November 2007: Globus incubator project v0.1
  - http://dev.globus.org/wiki/Incubator/Falkon
  - February 2009: Globus incubator project v0.9

- Implemented in Java (~20K lines of code) and C (~1K lines of code)
- Open source: svn co https://svn.globus.org/repos/falkon

- Source code contributors (beside myself)
  - Yong Zhao, Zhao Zhang, Ben Clifford, Mihael Hategan

Workload:
- 160K CPUs
- 1M tasks
- 60 sec per task
- 2 CPU years in 453 sec
- Throughput: 2312 tasks/sec
- 85% efficiency

References:
- [Globus07] “Falkon: A Proposal for Project Globus Incubation”
- [CLUSTER10] “Middleware Support for Many-Task Computing”
Falkon Activity History (16 months)

Max CPUs: 166,305
CPU Hours: 2.03M
Num Tasks: 173M
Aver Task Exec: 64 ± 486

Allocated CPUs
Delivered Tasks

Distributed Falkon Architecture

Login Nodes (x10)

I/O Nodes (x640)

Dispatcher 1

Dispatcher N

Compute Nodes (x40K)

Executor 1

Executor 256

Executor 256

Client

Provisioner

Cobalt

Managing 160K CPUs
IBM Blue Gene/P

ZeptOS

High-speed local disk

Slower distributed storage

Falkon
Applications

Medical Imaging: fMRI

• Wide range of analyses
  – Testing, interactive analysis, production runs
  – Data mining
  – Parameter studies

Improvement: up to 90% lower end-to-end run time

[SC07] “Falkon: a Fast and Light-weight task execution framework”
[SWF07] “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”
Applications
Astronomy: Montage

Improvement:
up to 57% lower end-to-end run time
Within 4% of MPI

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

[SC07] “Falkon: a Fast and Light-weight task execution framework”
[SWF07] “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”
Applications
Molecular Dynamics: MolDyn

• Determination of free energies in aqueous solution
  – Antechamber – coordinates
  – Charmm – solution
  – Charmm - free energy

Improvement:
up to 88% lower end-to-end run time
5X more scalable

[NOVA08] "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments"
Applications
Word Count and Sort

• Classic benchmarks for MapReduce
  – Word Count
  – Sort

• Swift and Falkon performs similar or better than Hadoop (on 32 processors)
Applications Economic Modeling: MARS

- CPU Cores: 130816
- Tasks: 1048576
- Elapsed time: 2483 secs
- CPU Years: 9.3

Applications Pharmaceuticals

PDB protein descriptions

ZINC 3-D structures

protein (1MB)

protein (6 GB)

FRED

DOCK6

~4M x 60s x 1 cpu
~60K cpu-hrs

Select best ~5K

Select best ~5K

Amber

~10K x 20m x 1 cpu
~3K cpu-hrs

Select best ~500

GCMC

~500 x 10hr x 100 cpu
~500K cpu-hrs

end

report

ligands

complexes

BuildNABScript

NAB Script

Amber prep:
1. AmberizeLigand
2. AmberizeReceptor
4. perl: gen nabscript

Amber Score:
1. AmberizeLigand
3. AmberizeComplex
5. RunNABScript

For 1 target:
4 million tasks
500,000 cpu-hrs
(50 cpu-years)

For 1 target:
4 million tasks
500,000 cpu-hrs
(50 cpu-years)

[SC08] "Towards Loosely-Coupled Programming on Petascale Systems"
Applications
Pharmaceuticals: DOCK

CPU cores: 118784
Tasks: 934803
Elapsed time: 2.01 hours
Compute time: 21.43 CPU years
Average task time: 667 sec
Relative Efficiency: 99.77%
(from 16 to 32 racks)
Utilization:
• Sustained: 99.6%
• Overall: 78.3%

Applications

Astronomy: AstroPortal

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

• Challenge
  – Processing Costs:
    • O(100ms) per object
  – Data Intensive:
    • 40MB:1sec
  – Rapid access to 10-10K “random” files
  – Time-varying load

<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>

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
[TG06] “AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis”
Applications
Astronomy: AstroPortal

- **AstroPortal**
  - Makes it really easy for astronomers to create stackings of objects from the Sloan Digital Sky Survey (SDSS) dataset.

- **Throughput**
  - 10X higher than GPFS

- **Reduced load**
  - 1/10 of the original GPFS load

- **Increased scalability**
  - 8X
Conclusions

- There is more to HPC than tightly coupled MPI, and more to HTC than embarrassingly parallel long jobs
- Data locality is critical at large-scale
• 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
  – Make use of idle resources from “supercomputers” via backfilling
  – Costs to run “supercomputers” per FLOP is among the best
• Loosely coupled apps do not require specialized system software
  – Their requirements on the job submission and storage systems can be extremely large
• Shared/parallel file systems are good for all applications
  – They don’t scale proportionally with the compute resources
  – Data intensive applications don’t perform and scale well
  – Growing compute/storage gap