Toward Loosely Coupled Programming on Petascale Systems

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

In Collaboration with:
Zhao Zhang & Ben Clifford, Univ. of Chicago
Mike Wilde & Ian Foster, Univ. of Chicago and Argonne National Lab.
Pete Beckman & Kamil Iskra, Argonne National Lab.

IEEE/ACM Supercomputing 2008
November 18th, 2008
PART I

Motivation
Many-Core Growth Rates

Pat Helland, Microsoft, The Irresistible Forces Meet the Movable Objects, November 9th, 2007
What will we do with 1+ Exaflops and 1M+ cores?
Programming Model Issues

- Multicore/Manycore processors
- Massive task parallelism
- Massive data parallelism
- Integrating black box applications
- Complex task dependencies (task graphs)
- Failure, and other execution management issues
- Dynamic task graphs
- Documenting provenance of data products
- Data management: input, intermediate, output
- Dynamic data access over large amounts of data
Problem Types

- **Input Data Size**
  - **Hi**
  - **Med**
  - **Low**

- **Number of Tasks**
  - **1**
  - **1K**
  - **1M**

- **Data Analysis, Mining**
- **Big Data and Many Tasks**
- **Heroic MPI Tasks**
- **Many Loosely Coupled Apps**

Toward Loosely Coupled Programming on Petascale Systems
An Incomplete and Simplistic View of Programming Models and Tools

- Single task, modest data
  - MPI, etc...
- Many Tasks
  - DAGMan+Pegasus
  - Karajan+Swift+Falkon
- Much Data
  - MapReduce/Hadoop
  - Dryad
- Complex Tasks, Much Data
  - Dryad, Pig, Sawzall
  - Swift+Falkon (using data diffusion)
MTC: Many Task Computing

- Bridge the gap between HPC and HTC
- Loosely coupled applications with HPC orientations
- HPC comprising of multiple distinct activities, coupled via file system operations or message passing
- Emphasis on many resources over short time periods
- Tasks can be:
  - small or large, independent and dependent, uniprocessor or multiprocessor, compute-intensive or data-intensive, static or dynamic, homogeneous or heterogeneous, loosely or tightly coupled, large number of tasks, large quantity of computing, and large volumes of data...
**MTAGS08: Workshop on Many-Task Computing on Grids and Supercomputers**

**Workshop Program**

The MTAGS08 will be on Monday, November 17th, 2008, in room 114B, from 8:30AM to 12PM. Please check in early to get your badge from the registration booth, as lines might be long. See below for the 7 talks that are part of the workshop, their abstracts, papers, and slides.

| Time | Description                                                                 | Authors                                                                 | Links |
|------|-----------------------------------------------------------------------------|------------------------------------------------------------------------|-------|
| 8:30AM | **Keynote Talk:** Many-task applications in use today with a look toward the future | Akio Ono, IBM Fellow and Blue Gene chief architect, IBM Research | Abstract | Paper | Slides |
| 9:05AM | **Session 1 (Rengan Moise, Session Chair)**                                  |                                                                        |       |
| 10:30AM | **A lightweight execution framework for massive independent tasks**          | Lli Hui, Peking University, China; Yu Hua, Peking University, China   | Abstract | Paper | Slides |
| 11:10AM | **ViG: A Grid Simulation and Monitoring Tool for Grid Workflows**             | A.T. Ther, Univ. of Texas at Arlington; O. V. Sebba, Univ. of Texas at Arlington | Abstract | Paper | Slides |
| 11:35AM | **Emarrassingly Parallel jobs are not Emarrassingly easy to Schedule on the Grid** | Eens Afzal, Univ. of Alabama at Birmingham; Farshid Esfahani Fard, Univ. of Alabama at Birmingham | Abstract | Paper | Slides |

**Closing Statements**
Growing Interest on enabling HTC/MTC on Supercomputers

- Project Kittyhawk
  - IBM Research

- HTC-mode in Cobalt/BG
  - IBM

- Condor on BG
  - University of Wisconsin at Madison, IBM

- Grid Enabling the BG
  - University of Colorado, National Center for Atmospheric Research

- Plan 9
  - Bell Labs, IBM Research, Sandia National Labs

- Falkon/Swift on BG/P and Sun Constellation
  - University of Chicago, Argonne National Laboratory
Many Large Systems available for Open Science Research

- Jaguar (#2) [to be announced in 90 minutes]
  - DOE, Oak Ridge National Laboratory
- Intrepid (#5)
  - DOE, Argonne National Laboratory
- Ranger (#6)
  - University of Texas / NFS TeraGrid
Why Petascale Systems for MTC Applications?

1. The I/O subsystem of petascale systems offers unique capabilities needed by MTC applications
2. The cost to manage and run on petascale systems is less than that of conventional clusters or Grids
3. Large-scale systems that favor large jobs have utilization issues
4. Some problems are intractable without petascale systems
PART II

Some context on systems we used as building blocks
Obstacles running MTC apps in Clusters/Grids

| System                  | Comments                        | Throughput (tasks/sec) |
|-------------------------|---------------------------------|------------------------|
| Condor (v6.7.2) - Production | Dual Xeon 2.4GHz, 4GB           | 0.49                   |
| PBS (v2.1.8) - Production | Dual Xeon 2.4GHz, 4GB           | 0.45                   |
| Condor (v6.7.2) - Production | Quad Xeon 3 GHz, 4GB           | 2                      |
| Condor (v6.8.2) - Production |                                |                        |
| Condor (v6.9.3) - Development |                                | 0.42                   |
| Condor-J2 - Experimental | Quad Xeon 3 GHz, 4GB           | 22                     |

Toward Loosely Coupled Programming on Petascale Systems
Solutions

• 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
    • Resource provisioning through multi-level scheduling techniques
    • Data diffusion and data-aware scheduling to leverage the co-located computational and storage resources

• Swift: A parallel programming system for loosely coupled applications
  – Applications cover many domains: Astronomy, astro-physics, medicine, chemistry, economics, climate modeling, data analytics
Falkon Overview

Toward Loosely Coupled Programming on Petascale Systems
Dispatch Throughput

| System                          | Comments                        | Throughput (tasks/sec) |
|--------------------------------|---------------------------------|------------------------|
| Condor (v6.7.2) - Production   | Dual Xeon 2.4GHz, 4GB           | 0.49                   |
| PBS (v2.1.8) - Production       | Dual Xeon 2.4GHz, 4GB           | 0.45                   |
| Condor (v6.7.2) - Production   | Quad Xeon 3 GHz, 4GB            | 2                      |
| Condor (v6.8.2) - Production   |                                 | 0.42                   |
| Condor (v6.9.3) - Development   |                                 | 11                     |
| Condor-J2 - Experimental        | Quad Xeon 3 GHz, 4GB            | 22                     |
Efficiency

Number of Processors

Efficiency

- 32 seconds
- 16 seconds
- 8 seconds
- 4 seconds
- 2 seconds
- 1 second
Falkon Endurance Test

- Completed Tasks
- Throughput (tasks/sec) - 60 sec aver

The graph shows the throughput of tasks per second over time, with the x-axis representing time in hours and the y-axis representing throughput in billions of tasks per second. The line indicates the trend of completed tasks over the 20-hour period.
Swift Architecture

Specification
- Abstract computation
- SwiftScript Compiler
- Virtual Data Catalog

Scheduling
- Execution Engine (Karajan w/ Swift Runtime)
  - Swift runtime callouts

Status reporting

Execution
- Virtual Node(s)
  - launcher App F1
  - Provenance data
  - file1

Provisioning
- Falkon Resource Provisioner
- Amazon EC2

Toward Loosely Coupled Programming on Petascale Systems
PART III

Contributions:
Proposed Changes & Results
Scaling from 1K to 100K CPUs

- At 1K CPUs:
  - 1 Server to manage all 1K CPUs
  - Use shared file system extensively
    - Invoke application from shared file system
    - Read/write data from/to shared file system

- At 100K CPUs:
  - N Servers to manage 100K CPUs (1:256 ratio)
  - Don’t trust the application I/O access patterns to behave optimally
    - Copy applications and input data to RAM
    - Read input data from RAM, compute, and write results to RAM
    - Archive all results in a single file in RAM
    - Copy 1 result file from RAM back to GPFS
  - Use collective I/O primitives to make app logic simpler
  - Leverage all networks (Ethernet, Tree, and Torus) for high aggregate bandwidth
Distributed Falkon Architecture

Login Nodes (x10)

I/O Nodes (x640)

Compute Nodes (x40K)

Client

Provisioner

Dispatcher 1

Dispatcher N

Executor 1

Executor 256

Executor 256

Executor 256

Cobalt
Managing 160K CPUs

Falkon

Slower shared storage

High-speed local disk

swift
Falkon Monitoring

- Workload
  - 160K CPUs
  - 1M tasks
  - 60 sec per task
  - 17.5K CPU hours in 7.5 min
  - Throughput: 2312 tasks/sec
  - 85% efficiency
Dispatch Throughput

Executor Implementation and Various Systems

Throughput (tasks/sec)

ANL/UC, Java
200 CPUs
1 service

ANL/UC, C
200 CPUs
1 service

SiCortex, C
5760 CPUs
1 service

BlueGene/P, C
4096 CPUs
1 service

BlueGene/P, C
163840 CPUs
640 services
Efficiency

![Graph showing efficiency versus number of processors for different time intervals. The graph includes lines for 256 seconds, 128 seconds, 64 seconds, 32 seconds, 16 seconds, 8 seconds, 4 seconds, 2 seconds, and 1 second. Each line represents the percentage of efficiency for a given number of processors.](image-url)
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%
MARS Economic Modeling on IBM BG/P (128K CPUs)

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

Speedup: 115168X (ideal 130816)
Efficiency: 88%
Many Many Tasks: Identifying Potential Drug Targets

Protein target(s) x 2M+ ligands

(Mike Kubal, Benoit Roux, and others)
Many Many Tasks: Identifying Potential Drug Targets

- PDB protein descriptions
- DOCK6 Receptor structures (1 per protein: defines pocket to bind to)
- FRED Receptor structures (1 per protein: defines pocket to bind to)
- NAB script parameters (defines flexible residues, #MDsteps)

**Start**

- **FRED**
  - ~4M x 60s x 1 cpu
  - ~60K cpu-hrs
- **DOCK6**
  - 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

**Report**

For 1 target:
- 4 million tasks
- 500,000 cpu-hours (50 cpu-years)
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%
DOCK on the BG/P

CPU cores: 118784
Tasks: 934803
Elapsed time: 2.01 hours
Compute time: 21.43 CPU years
Average task time: 667 sec
Relative Efficiency: 99.7% (from 16 to 32 racks)
Utilization:
  • Sustained: 99.6%
  • Overall: 78.3%

Toward Loosely Coupled Programming on Petascale Systems
 Costs to interact with GPFS

- Directory Create (single dir)
- File Create (single dir)
- Directory Create (across many dirs)
- File Create (across many dirs)
- Script Invocation
- Falkon Overhead (i.e. sleep 0)

Time per Operation (sec)

Number of Processors

256 4096 8192 16384
LCP Collective IO Model

Application Script

Global FS

ZOI0D IFS for staging
ZOI0D on IO node

<-- Torus & Tree Interconnects -->

CN-striped IFS for Data

Large Input Dataset

IFS seg

IFS Compute node

IFS seg

IFS Compute node

LFS Compute node (local datasets)

LFS Compute node (local datasets)
Read performance from IFS

Throughput (MB/s)

Degree of Striping over Multiple Nodes

158  187  315  532  746  831
Write Performance
CIO vs. GFS efficiency

Efficiency vs. Number of Processors

- 32sec+GPFS(1KB)
- 32sec+GPFS(16KB)
- 32sec+GPFS(128KB)
- 32sec+GPFS(1MB)
- 32sec+CIO(1KB)
- 32sec+CIO(16KB)
- 32sec+CIO(128KB)
- 32sec+CIO(1MB)
Falkon Activity History
(10 months)

Max CPUs: 163K
CPU Hours: 1.4M
Num Tasks: 164M
Task Exec: 31 sec

Allocated CPUs
Delivered Tasks

Allocated CPUs (60 sec average)

Completed Tasks

12/7/2007 1/7/2008 2/7/2008 3/9/2008 4/9/2008 5/10/2008 6/10/2008 7/11/2008 8/11/2008 9/11/2008 10/12/2008
Conclusions and Future Work
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
  – Make use of idle resources from “supercomputers” via backfilling
  – Costs to run “supercomputers” per FLOP is among the best
    • BG/P: 0.35 gigaflops/watt (**higher is better**)
    • SiCortex: 0.32 gigaflops/watt
    • BG/L: 0.23 gigaflops/watt
    • x86-based HPC systems: an order of magnitude lower
• **Loosely coupled apps do not require specialized system software**
• **Shared 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
Conclusions & Contributions

• Defined a new class of applications: MTC
• Proved that MTC applications can be executed efficiently on supercomputers at full scale
• Extended Falkon by distributing the dispatcher/scheduler
• Falkon installed and configured on the BG/P for anyone to use
Future Work: Other Supercomputers

- **Ranger: Sun Constellation**
  - Basic mechanisms in place, and have started testing
- **Jaguar: Cray**
  - Plan to get accounts on machine as soon as its online
- **Future Blue Gene machines (Q?)**
  - Discussions underway between IBM, ANL and UChicago
Future Work: 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
All-Pairs Workload
1000x1000 on 4K emulated CPUs

Efficiency: 86%
More Information

- More information: [http://people.cs.uchicago.edu/~iraicu/](http://people.cs.uchicago.edu/~iraicu/)
- Related Projects:
  - Falkon: [http://dev.globus.org/wiki/Incubator/Falkon](http://dev.globus.org/wiki/Incubator/Falkon)
  - Swift: [http://www.ci.uchicago.edu/swift/index.php](http://www.ci.uchicago.edu/swift/index.php)
- Funding:
  - **NASA**: Ames Research Center, Graduate Student Research Program
    - Jerry C. Yan, NASA GSRP Research Advisor
  - **DOE**: Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy
  - **NSF**: TeraGrid
Main Page

Megajobs: How to Run One Million Jobs

- What: Birds-of-a-Feather Session at Supercomputing 2008, Austin Texas
- Date: Tuesday, November 18th, 2008
- Time: 06:30PM - 07:00PM
- Location: Room 13A/13B
- Primary Session Leader:
  - Marion Pierce (Indiana University)
- Secondary Session Leader:
  - Ion Raicu (University of Chicago)
  - Ruth Porches (Fermi National Laboratory)
  - John McGee (Renaissance Computing Institute)
  - Dick Rapaport (Indiana University)

As large systems surpass 200K CPU cores and as applications increase in complexity, more scientists need to run thousands to millions of closely related jobs that are associated with individual projects. Scientists seek convenient means to specify and manage many jobs, arranging inputs, aggregating outputs, identifying successful and failed jobs and repairing failures. System administrators seek methods to process extraordinary numbers of jobs for multiple users without overwhelming queuing systems or disrupting fair-share usage policies. Under development are a new generation of queuing and scheduling systems and multi-level schedulers for use with existing queuing and scheduling systems, schedulers designed to handle millions of jobs. This Birds-of-feather session provides a venue for the exchange of information about processing large numbers of jobs. Short presentations of an invited sample of projects will be followed by discussion.

We are currently soliciting participation in the “Megajobs” BOF. We are looking for short, piquant presentations (6-10 minutes) from people who have worked on this problem or have a problem like this that needs to be worked on. If you are interested, please send a brief title and abstract (250 words) to Marion Pierce by October 27th, 2008. Please feel free to contact us if you have questions.

For the latest information hosted by SC08, see http://sc08way.nas.nasa.gov/conference/view/bof/118. The Megajobs BOF handout can also be found here.

Related activities at SC08, that might be of interest to BOF attendees are:
- Grid Computing Environments (GCE)
- Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS)