Harnessing Grid Resources with Data-Centric Task Farms

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

In Collaboration with:
Ian Foster, University of Chicago and Argonne National Laboratory
Rick Stevens, University of Chicago and Argonne National Laboratory
Alex Szalay, The Johns Hopkins University
+24 more people, see “Most Recent Collaborators Relevant to Dissertation” slide at the end...

August 20th, 2008
What will we do with 1+ Exaflops and 1M+ cores?
Programming Model Issues

- Multicore 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 involving large amounts of data
Programming Model Issues

• Multicore 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 involving large amounts of data
Problem Types

Input Data Size

Hi

Med

Low

1

1K

1M

Number of Tasks

Data Analysis, Mining

Big Data and Many Tasks

Heroic MPI Tasks

Many Loosely Coupled Apps

8/20/2008
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

- Loosely coupled applications
  - High-performance computations comprising of multiple distinct activities, coupled via file system operations or message passing
  - Emphasis on using 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…
Motivating Example: AstroPortal Stacking Service

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

- **Challenge**
  - Processing Costs:
    - $O(100\text{ms})$ per object
  - Data Intensive:
    - 40MB:1sec
  - Rapid access to 10-10K “random” files
  - Time-varying load
Obstacles and Solutions

• Obstacles:
  1. Long queue times
  2. Slow job dispatch rates
  3. Poor shared file system scaling

...many many years of hard work...

• Solution ➔ Falkon: a Fast and Light-weight task execution framework
  1. Streamlined dispatching
  2. Multi-level scheduling
  3. Data diffusion
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
Abstract Model

• AMDASK: An Abstract Model for DAta-centric taSK farms
  – Task Farm: A common parallel pattern that drives independent computational tasks
• Models the efficiency of data analysis workloads for the split/merge class of applications
• Captures data diffusion properties
  – Resources are acquired in response to demand
  – Data and applications diffuse from archival storage to new resources
  – Resource “caching” allows faster responses to subsequent requests
  – Resources are released when demand drops
  – Considers both data and computations to optimize performance
AMDASK: Base Definitions

- **Data Stores**: Persistent & Transient
  - Store capacity, load, ideal bandwidth, available bandwidth
- **Data Objects**: 
  - Data object size, *data object’s storage location(s)*, copy time
- **Transient resources**: compute speed, resource state
- **Task**: application, input/output data
AMDASK: Execution Model Concepts

- Dispatch Policy
  - next-available, first-available, max-compute-util, max-cache-hit
- Caching Policy
  - random, FIFO, LRU, LFU
- Replay policy
- Data Fetch Policy
  - Just-in-Time, Spatial Locality
- Resource Acquisition Policy
  - one-at-a-time, additive, exponential, all-at-once, optimal
- Resource Release Policy
  - distributed, centralized
AMDASK: Performance Efficiency Model

• B: Average Task Execution Time:
  – K: Stream of tasks
  – \( \mu(k) \): Task k execution time
  \[ B = \frac{1}{|K|} \sum_{k \in K} \mu(k) \]

• Y: Average Task Execution Time with Overheads:
  – \( o(k) \): Dispatch overhead
  – \( \zeta(\delta, \tau) \): Time to get data
  \[ Y = \begin{cases} 
  \frac{1}{|K|} \sum_{k \in K} [\mu(k) + o(k)], & \delta \in \phi(\tau), \delta \in \Omega \\
  \frac{1}{|K|} \sum_{k \in K} [\mu(k) + o(k) + \zeta(\delta, \tau)], & \delta \notin \phi(\tau), \delta \in \Omega 
\end{cases} \]

• V: Workload Execution Time:
  – A: Arrival rate of tasks
  – T: Transient Resources
  \[ V = \max \left( \frac{B}{|T|}, \frac{1}{A} \right)^* |K| \]

• W: Workload Execution Time with Overheads
  \[ W = \max \left( \frac{Y}{|T|}, \frac{1}{A} \right)^* |K| \]
AMDASK: Performance Efficiency Model

- **Efficiency**
  
  \[ E = \frac{V}{W} \]

  \[ E = \begin{cases} 
  1, & \frac{Y}{|T|} \leq \frac{1}{A} \\
  \max \left( \frac{B}{Y}, \frac{|T|}{A*Y} \right), & \frac{Y}{|T|} > \frac{1}{A} 
  \end{cases} \]

- **Speedup**

  \[ S = E^* |T| \]

- **Optimizing Efficiency**
  
  - Easy to maximize either efficiency or speedup independently
  
  - Harder to maximize both at the same time

  - Find the smallest number of *transient resources* |T| while maximizing speedup*efficiency
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

• Integration into Swift to leverage many applications
  – Applications cover many domains: astronomy, astro-physics, medicine, chemistry, economics, climate modeling, etc
# Dispatch Throughput

<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>
• End-to-end execution time:
  – 1260 sec in ideal case
  – 4904 sec $\rightarrow$ 1276 sec
• Average task queue time:
  – 42.2 sec in ideal case
  – 611 sec $\rightarrow$ 43.5 sec
• Trade-off:
  – Resource Utilization for Execution Efficiency
Swift Architecture

Specification
- Abstract computation
- SwiftScript Compiler
- Virtual Data Catalog
- SwiftScript

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

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

Provisioning
- Falkon Resource Provisioner
- Amazon EC2

Scientific Workflow Systems for 21st Century, New Bottle or New Wine?
Functional MRI (fMRI)

• Wide range of analyses
  – Testing, interactive analysis, production runs
  – Data mining
  – Parameter studies
Completed Milestones: fMRI Application

- GRAM vs. Falkon: 85%~90% lower run time
- GRAM/Clustering vs. Falkon: 40%~74% lower run time
B. Berriman, J. Good (Caltech)
J. Jacob, D. Katz (JPL)
Completed Milestones: Montage Application

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

8/20/2008 Harnessing Grid Resources with Data-Centric Task Farms
Hadoop vs. Swift

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

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

- 244 molecules → 20497 jobs
- 15091 seconds on 216 CPUs → 867.1 CPU hours
- Efficiency: 99.8%
- Speedup: 206.9x → 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%
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 \times \text{target(s)} \quad \text{2M+ ligands}

Scientific Workflow Systems for 21st Century, New Bottle or New Wine?
(Mike Kubal, Benoit Roux, and others)
Many Many Tasks: Identifying Potential Drug Targets

- PDB protein descriptions (1MB)
- ZINC 3-D structures (6 GB)
- Protein (1 per protein: defines pocket to bind to)
- Dock6 rec file
- FRED rec file
- NAB script parameters (defines flexible residues, #MDsteps)

Start:

- FRED
  - ~4M x 60s x 1 cpu
  - ~60K cpu-hrs
- DOCK6
  - Select best ~5K
  - ~60K cpu-hrs
- Amber
  - Select best ~5K
  - ~10K x 20m x 1 cpu
  - ~3K cpu-hrs
- GCMC
  - Select best ~500
  - ~500 x 10hr x 100 cpu
  - ~500K cpu-hrs

End:

- Report
- Ligands
- Complexes

For 1 target:
- 4 million tasks
- 500,000 cpu-hrs (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%
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
Data Diffusion

• Considers both data and computations to optimize performance
  – Supports data-aware scheduling
  – Can optimize compute utilization, cache hit performance, or a mixture of the two

• Decrease dependency of a shared file system
  – Theoretical linear scalability with compute resources
  – Significantly increases meta-data creation and/or modification performance

• Central for “data-centric task farm” realization
Scheduling Policies

• first-available:
  – simple load balancing

• max-cache-hit
  – maximize cache hits

• max-compute-util
  – maximize processor utilization

• good-cache-compute
  – maximize both cache hit and processor utilization at the same time
Data-Aware Scheduler Profiling

![Graph showing CPU time per task and throughput (tasks/sec)]

- Task Submit
- Notification for Task Availability
- Task Dispatch (data-aware scheduler)
- Task Results (data-aware scheduler)
- Notification for Task Results
- WS Communication
- Throughput (tasks/sec)

8/20/2008 Harnessing Grid Resources with Data Data-Centric Task Farms
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>

8/20/2008 Harnessing Grid Resources with Data-Centric Task Farms
AstroPortal Stacking Service

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

- **Challenge**
  - Rapid access to 10-10K random files

  - open
  - radec2xy
  - readHDU+getTile+curl+convertArray
  - calibration+interpolation+doStacking
  - writeStacking

- **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>97999</td>
<td>49000</td>
</tr>
<tr>
<td>2</td>
<td>88857</td>
<td>296204</td>
</tr>
<tr>
<td>3</td>
<td>60590</td>
<td>12120</td>
</tr>
<tr>
<td>5</td>
<td>46480</td>
<td>465020</td>
</tr>
<tr>
<td>10</td>
<td>23695</td>
<td>790</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Filesystem and Image Format**

  - GPFS GZ
  - LOCAL GZ
  - GPFS FIT
  - LOCAL FIT

8/20/2008 Harnessing Grid Resources with Data-Centric Task Farms
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
AMDASK Model Validation

- Stacking service (large scale astronomy application)
- 92 experiments
- 558K files
  - Compressed: 2MB each → 1.1TB
  - Un-compressed: 6MB each → 3.3TB
Data Diffusion: Data-Intensive Workload

- 250K tasks
  - 10MB reads
  - 10ms compute
- Vary arrival rate:
  - Min: 1 task/sec
  - Increment function: CEILING(*1.3)
  - Max: 1000 tasks/sec
- 128 processors
- Ideal case:
  - 1415 sec
  - 80Gb/s peak throughput
Data Diffusion: First-available (GPFS)

- **GPFS vs. ideal**: 5011 sec vs. 1415 sec
Data Diffusion: Max-compute-util & max-cache-hit

Max-compute-util

Max-cache-hit

8/20/2008 Harnessing Grid Resources with Data Data-Centric Task Farms
Data Diffusion: Good-cache-compute

1GB ➡️ 1.5GB ➡️

2GB ➡️ 4GB ➡️
Data Diffusion: Throughput and Response Time

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

Response Time
- 3 sec vs 1569 sec → 506X

8/20/2008
Data Diffusion: Performance Index, Slowdown, and Speedup

• Performance Index:
  – 34X higher

• Speedup
  – 3.5X faster than GPFS

• Slowdown:
  – 18X slowdown for GPFS
  – Near ideal 1X slowdown for large enough caches
### All-Pairs Synthetic Workload

**Falkon and Data Diffusion**

500x500 ~ 250K tasks, 12MBx2 in, 8B out, 1 sec compute

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of CPUs</th>
<th>Time (sec)</th>
<th>Average Throughput (GB/s)</th>
<th>Average Throughput per CPU (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best Case Model</strong></td>
<td>&lt;500</td>
<td>~10800</td>
<td>~0.56</td>
<td>~1.11</td>
</tr>
<tr>
<td>Condor + Chirp (Active Storage)</td>
<td>&lt;500</td>
<td>~21600</td>
<td>~0.28</td>
<td>~0.56</td>
</tr>
<tr>
<td>Condor + Shared File System (Demand Paging)</td>
<td>&lt;500</td>
<td>~37800</td>
<td>~0.16</td>
<td>~0.32</td>
</tr>
<tr>
<td><strong>Idea Case (with I/O to Local Disk)</strong></td>
<td>128</td>
<td>1953</td>
<td>3.07</td>
<td>24.00</td>
</tr>
<tr>
<td>Falkon (Data Diffusion)</td>
<td>128</td>
<td>3056</td>
<td>1.96</td>
<td>15.34</td>
</tr>
<tr>
<td>Falkon (GPFS)</td>
<td>128</td>
<td>5438</td>
<td>1.10</td>
<td>8.62</td>
</tr>
</tbody>
</table>

500x500 ~ 250K tasks, 12MBx2 in, 8B out, 100ms compute

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of CPUs</th>
<th>Time (sec)</th>
<th>Average Throughput (GB/s)</th>
<th>Average Throughput per CPU (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Idea Case (with I/O to Local Disk)</strong></td>
<td>128</td>
<td>938</td>
<td>6.4</td>
<td>50.0</td>
</tr>
<tr>
<td>Falkon (Data Diffusion)</td>
<td>128</td>
<td>1033</td>
<td>5.8</td>
<td>45.4</td>
</tr>
<tr>
<td>Falkon (GPFS)</td>
<td>128</td>
<td>5416</td>
<td>1.1</td>
<td>8.7</td>
</tr>
</tbody>
</table>
Related Work: Task Farms

- [Casanova99]: Adaptive Scheduling for Task Farming with Grid Middleware
- [Heymann00]: Adaptive Scheduling for Master-Worker Applications on the Computational Grid
- [Danelutto04]: Adaptive Task Farm Implementation Strategies
- [González-Vélez05]: An Adaptive Skeletal Task Farm for Grids
- [Petrou05]: Scheduling Speculative Tasks in a Compute Farm
- [Reid06]: Task farming on Blue Gene

**Conclusion:** none addressed the proposed “data-centric” part of task farms, and the implementations were not as light-weight as ours
Related Work: Resource Provisioning

- [Appleby01]: Oceano - SLA Based Management of a Computing Utility
- [Frey02, Mehta06]: Condor glide-ins
- [Walker06]: MyCluster (based on Condor glide-ins)
- [Ramakrishnan06]: Grid Hosting with Adaptive Resource Control
- [Bresnahan06]: Provisioning of bandwidth
- [Singh06]: Simulations

**Conclusion:** None allows for dynamic resizing of resource pool (independent of application logic) based on system load
Related Work: Data Management

- [Beynon01]: DataCutter
- [Ranganathan03]: Simulations
- [Ghemawat03, Dean04, Chang06]: BigTable, GFS, MapReduce
- [Liu04]: GridDB
- [Chervenak04, Chervenak06]: RLS (Replica Location Service), DRS (Data Replication Service)
- [Tatebe04, Xiaohui05]: GFarm
- [Branco04, Adams06]: DIAL/ATLAS
- [Kosar06]: Stork
- [Thain08]: Chirp/Parrot

**Conclusion:** None focused on the co-location of storage and generic black box computations with data-aware scheduling while operating in a dynamic environment.
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 an abstract model for performance efficiency of data analysis workloads using data-centric task farms
• Provide a reference implementation (Falkon)
  – Use a streamlined dispatcher to increase task throughput by several orders of magnitude over traditional LRMIs
  – Use multi-level scheduling to reduce perceived wait queue time for tasks to execute on remote resources
  – Address data diffusion through co-scheduling of storage and computational resources to improve performance and scalability
  – Provide the benefits of dedicated hardware without the associated high cost
  – Show effectiveness on a real large-scale astronomy application
More Information

• More information: http://people.cs.uchicago.edu/~iraicu/
• Related Projects:
  – Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  – Swift: http://www.ci.uchicago.edu/swift/index.php
• Dissertation Committee:
  – Ian Foster, The University of Chicago & Argonne National Laboratory
  – Rick Stevens, The University of Chicago & Argonne National Laboratory
  – Alex Szalay, The Johns Hopkins University
• 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
Recent Collaborators
Relevant to Dissertation
(2006 – Present)

- **Ian Foster**, The University of Chicago & Argonne National Laboratory
- **Alex Szalay**, The Johns Hopkins University
- **Rick Stevens**, The University of Chicago & Argonne National Laboratory
- **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
- **Kamil Iskra**, Argonne National Laboratory
- **Pete Beckman**, Argonne National Laboratory
- **Mihae Hategan**, The University of Chicago
- **Ben Clifford**, The University of Chicago
- **Shiyong Lu**, Wayne State University
- **Veronika Nefedova**, Argonne National Laboratory
- **Tiberiu Stef-Praun**, The University of Chicago
- **Gabriela Turcu**, The University of Chicago
- **Atilla S. Balkir**, The University of Chicago
- **Jing Tie**, The University of Chicago
- **Quan T. Pham**, The University of Chicago
- **Sarah Kenny**, The University of Chicago
- **Dick Repasky**, Indiana University
- **Gregor von Laszewski**, Rochester Institute of Technology
- **Jim Gray**, Microsoft Research
- **Ruth Pordes**, Fermilab National Accelerator Laboratory
- **John McGee**, Renaissance Computing Institute
- **Julian Bunn**, California Institute of Technology
- **Marlon Pierce**, Indiana University
All Collaborators
(2000 – Present)

- Ian Foster
- Catalin Dumitrescu
- Yong Zhao
- Mike Wilde
- Alex Szalay
- Sherali Zeadally
- Matei Ripeanu
- Mihael Hategan
- Zhao Zhang
- Alexandru Iosup
- Ben Clifford
- Dick Epema
- Jerry Yan
- Loren Schwiebert
- John Bresnahan
- Kamil Iskra
- Nicolae Tapus
- Owen Richter
- Rick Stevens
- Sandeep Gupta
- Scott Fowler
- Ahmad Naveed
- Anne-Marie Bosneag
- Atilla Balkir
- Carl Kesselman
- Davis Ford
- Dick Repasky
- Douglas Comer
- Gabriela Turcu
- Gohar Margaryan
- Gregor von Laszewski
- H Mohamed
- Jan Dünnweber
- Jennifer M Schoph
- Jim Gray
- Jing Tie
- John McGee
- Julian Bunn
- Laura Pearlman
- Liqiang Zhang
- Marlon Pierce
- Michael Link
- Mike D’Arcy
- Mugurel Andreica
- Neill Miller
- Pete Beckman
- Pham Quan
- Philipp Lüdeking
- R Wasseem
- Rajkumar Kettimuthu
- Ruth Pordes
- Sarah Kenny
- Sergei Gorlatch
- Shiyong Lu
- Tiberiu Stef-Praun
- Veronika Nefedova
- William Allcock
Publications/Proposals
Central to Dissertation
(2005 – Present)

- Ioan Raicu, Ian Foster. “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets: Year 2 Status and Year 3 Proposal”, GSRP, Ames Research Center, NASA, March 2008 -- Award funded 10/1/08 - 9/30/09.
- Quan T. Pham, Atilla S. Balkir, Jing Tie, Ian Foster, Mike Wilde, Ioan Raicu. “Data Intensive Scalable Computing on TeraGrid: A Comparison of MapReduce and Swift”, Poster Presentation, TeraGrid Conference 2008.
- Ioan Raicu, Yong Zhao, Ian Foster, Mike Wilde, Zhao Zhang, Ben Clifford, Mihael Hategan, Sarah Kenny. “Managing and Executing Loosely Coupled Large Scale Applications on Clusters, Grids, and Supercomputers”, Extended Abstract, GlobusWorld08, part of Open Source Grid and Cluster Conference 2008.
- Ioan Raicu, Ian Foster. “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets: Year 1 Status and Year 2 Proposal”, GSRP, Ames Research Center, NASA, February 2007 -- Award funded 10/1/07 - 9/30/08.
- Ioan Raicu, Yong Zhao, Ian Foster, Alex Szalay. “A Data Diffusion Approach to Large Scale Scientific Exploration”, Microsoft Research eScience Workshop 2007.
- Ioan Raicu, Ian Foster. “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets”, GSRP, Ames Research Center, NASA, February 2006 -- Award funded 10/1/06 - 9/30/07.
Other Publications
(2002 – 2007)
Disjoint Set from Previous Slide


• Catalin Dumitrescu, Ioan Raicu, Ian Foster. “Performance Measurements in Running Workloads over a Grid”, The 4th International Conference on Grid and Cooperative Computing (GCC 2005); 11% acceptance rate.


Service
(2002 – Present)

- IEEE/ACM Workshop on Grid Computing Portals and Science Gateways (GCE08)
- IEEE International Conference on Internet and Web Applications and Services (ICIW 2009)
- TeraGrid Conference (TG09)
- IEEE International Conference on Networks (ICN 2009)
- IEEE International Conference on Networking and Services (ICNS 2009)
- Distributed Systems Laboratory Workshop (DSLW08)
- IEEE International Conference on Internet and Web Applications and Services (ICIW08)
- Sixth Annual Conference on Communication Networks and Services Research (CNSR08)
- TeraGrid Conference (TG08)
- ACM/IET/ICST International Workshop on Performance and Analysis of Wireless Networks (PAWN08)
- IEEE International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCMP08)
- IEEE International Conference on Systems and Networks Communications (ICNSC08)
- IEEE International Conference on Networking and Services (ICNS08)
- IEEE International Conference on Networking (ICN08)
- IEEE Internet Computing, Special Issue on Virtual Organizations, 2007
- IEEE/ACM Workshop on Grid Computing Portals and Science Gateways (GCE07)
- IEEE/ACM Grid Conference (SC07)
- Distributed Systems Laboratory Workshop (DSLW07)
- IEEE Internet Computing (IC07)
- The Handbook of Computer Networks (2007)
- IEEE/ACM SuperComputing (SC06)
- Distributed Systems Laboratory Workshop (DSLW06)
- IEEE Transactions on Computers (TC06)
- Journal of Concurrency and Computation: Practice and Experience 2006
- IEEE Communication Letters (CL05)
- High Performance Computing Symposium (HPCC05)
- IEEE Intelligent Sensing and Information Processing (ICISIP05)
- ARC Research Network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP05)
- IEEE International Conference on Computer Communications and Networks (IC3N02)
- IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS02)