Harnessing Grid Resources with Data-Centric Task Farms

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Committee Members:
Ian Foster: University of Chicago, Argonne National Laboratory
Rick Stevens: University of Chicago, Argonne National Laboratory
Alex Szalay: The Johns Hopkins University

Candidacy Exam
December 12th, 2007

Outline

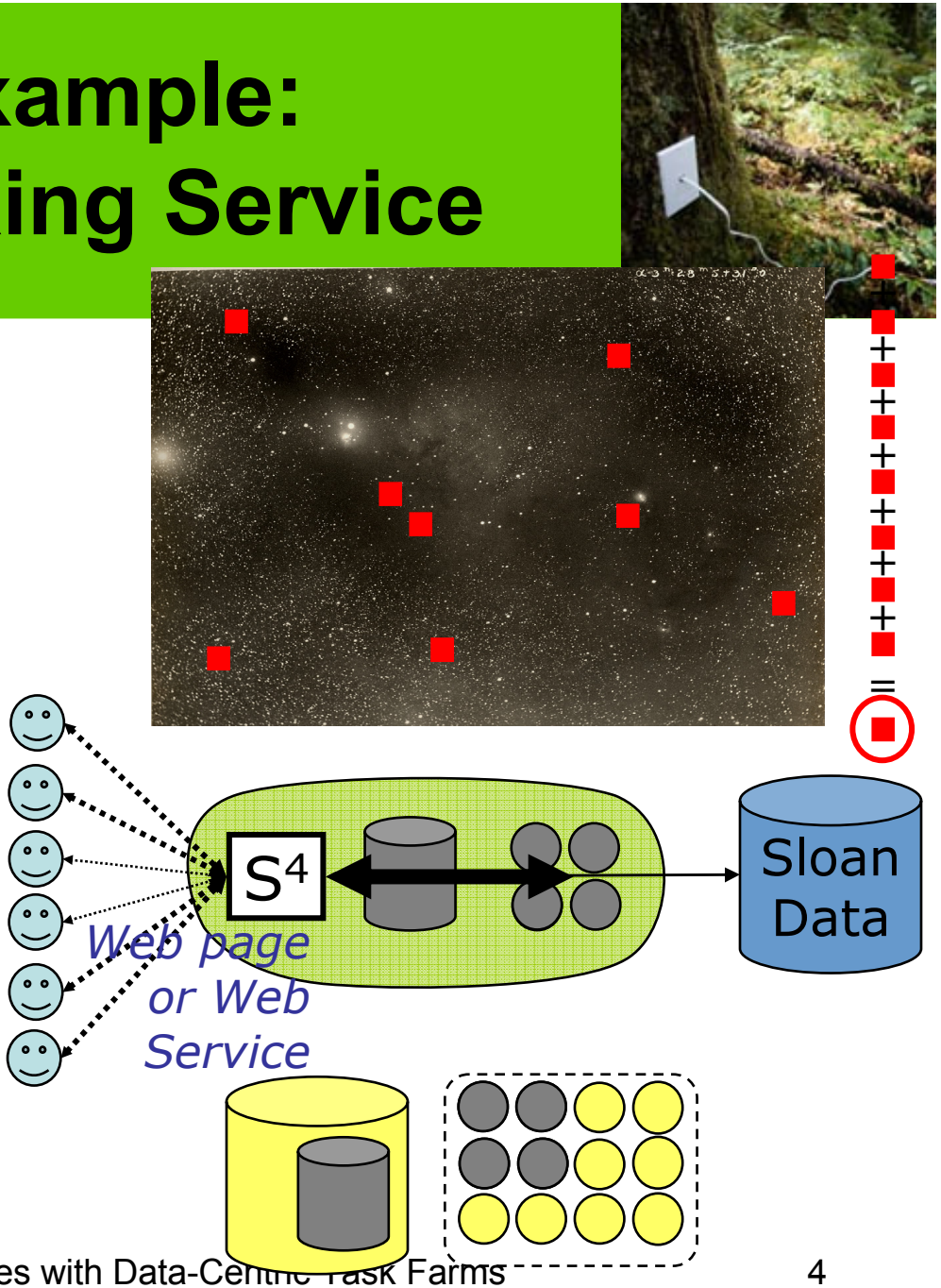
1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

Outline

1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

Motivating Example: AstroPortal Stacking Service

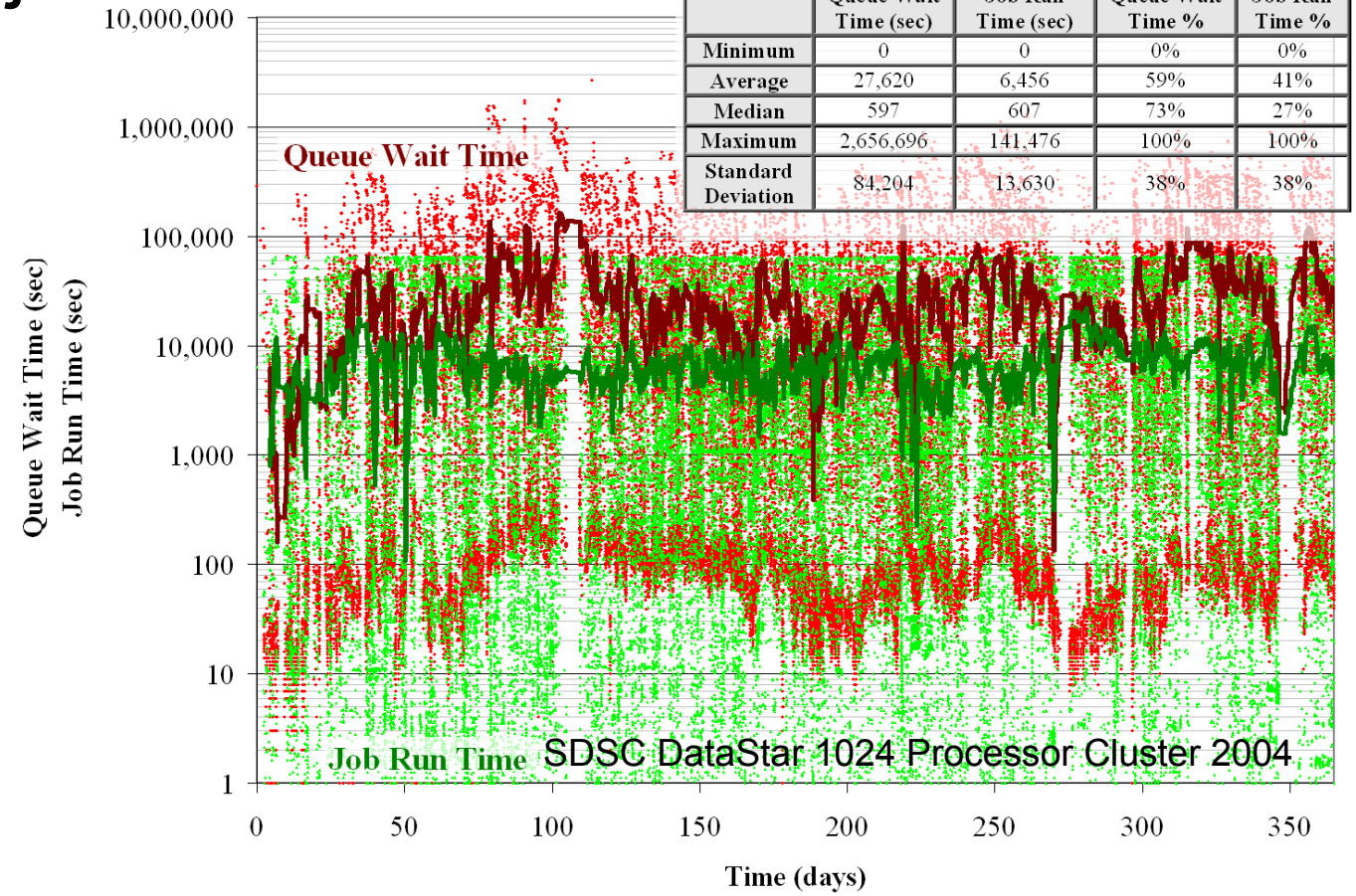
- Purpose
  - On-demand "stacks" of random locations within ~10TB dataset
- Challenge
  - Rapid access to 10-10K "random" files
  - Time-varying load
- Solution
  - Dynamic acquisition of compute, storage



Challenge #1: Long Queue Times

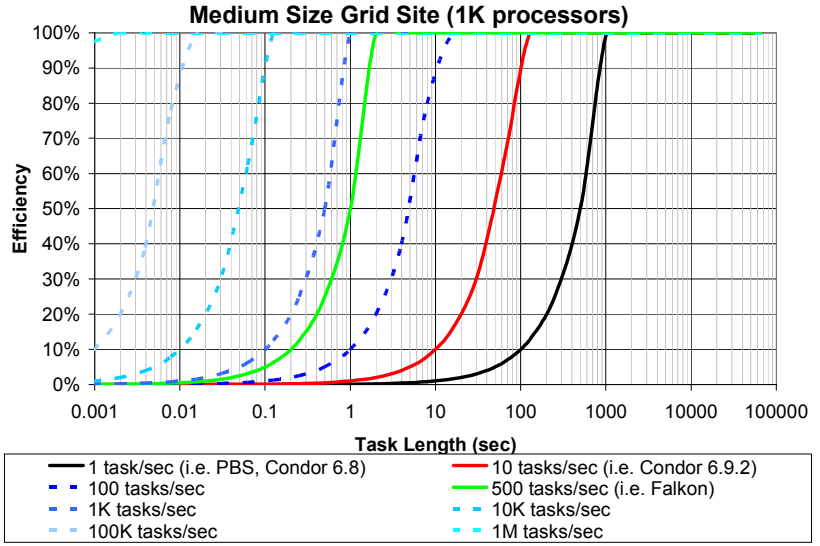
- Wait queue times are typically longer than the job duration times

	Queue Wait Time (sec)	Job Run Time (sec)	Queue Wait Time %	Job Run Time %
Minimum	0	0	0%	0%
Average	27,620	6,456	59%	41%
Median	597	607	73%	27%
Maximum	2,656,696	141,476	100%	100%
Standard Deviation	84,204	13,630	38%	38%



Challenge #2: Slow Job Dispatch Rates

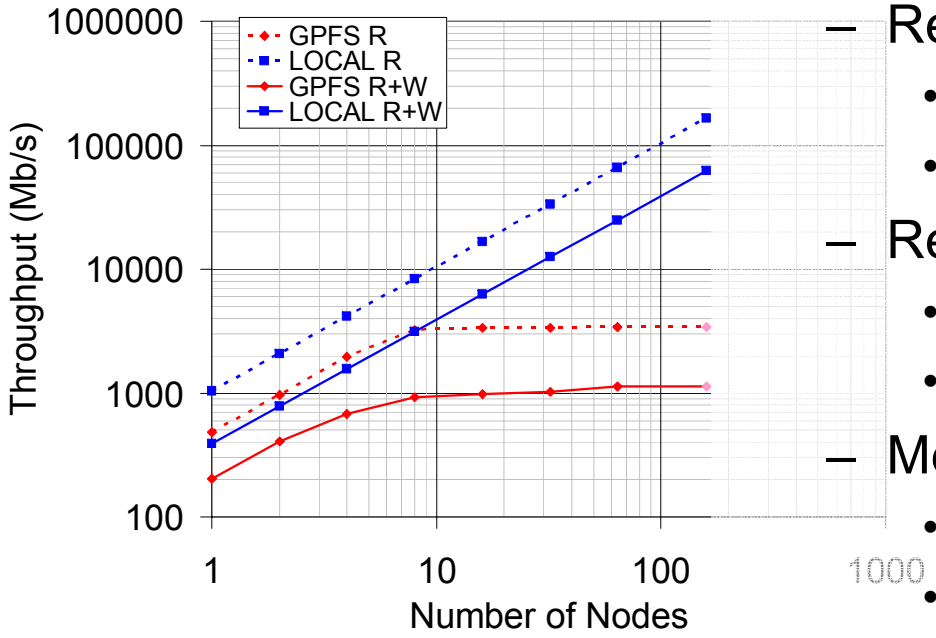
- Production LRMs → ~1 job/sec dispatch rates
- What job durations are needed for 90% efficiency:
  - Production LRMs: **900** sec
  - Development LRMs: **100** sec
  - Experimental LRMs: **50** sec
  - **1~10 sec** should be possible



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

Challenge #3: Poor Scalability of Shared File Systems

- GPFS vs. LOCAL



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

Outline

1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

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

Proposed Solution: Part 1 Abstract Model and Validation

- 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 the following 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
- Model Validation
  - Implement the abstract model in a discrete event simulation
  - Validate model with statistical methods (R^2 Statistic, Residual Analysis)

Proposed Solution: Part 2 Practical Realization

- Falkon: a Fast and Light-weight tasK executiON framework
  - Light-weight task dispatch mechanism
  - Dynamic resource provisioning to acquire and release resources
  - Data management capabilities including data-aware scheduling
  - Integration into Swift to leverage many Swift-based applications
    - Applications cover many domains: astronomy, astro-physics, medicine, chemistry, and economics

Outline

1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

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
  - μ(k): Task k execution time

$$B = \frac{1}{|K|}\sum_{k \in K} \mu(\kappa)$$

- Y: Average Task Execution Time with Overheads:
  - o(k): Dispatch overhead
  - ς(δ,τ): Time to get data

$$Y = \begin{cases} \frac{1}{|K|}\sum_{\kappa \in K}[\mu(\kappa) + o(\kappa)], & \delta \in \phi(\tau), \delta \in \Omega \\ \frac{1}{|K|}\sum_{\kappa \in K}[\mu(\kappa) + o(\kappa) + \zeta(\delta, \tau)], & \delta \notin \phi(\tau), \delta \in \Omega \end{cases}$$

- 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**

$$\mathrm{E} = \frac{V}{W} \longrightarrow E = \begin{cases} 1, & \frac{Y}{|T|} \leq \frac{1}{A} \\ \max\left(\frac{B}{Y}, \frac{|\mathrm{T}|}{\mathrm{A} * Y}\right), & \frac{Y}{|T|} > \frac{1}{A} \end{cases}$$

- **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

Performance Efficiency Model Example: 1K CPU Cluster

- Application: Angle - distributed data mining
- Testbed Characteristics:
  - Computational Resources: 1024
  - Transient Resource Bandwidth: 10MB/sec
  - Persistent Store Bandwidth: 426MB/sec
- Workload:
  - Number of Tasks: 128K
  - Arrival rate: 1000/sec
  - Average task execution time: 60 sec
  - Data Object Size: 40MB

Performance Efficiency Model Example: 1K CPU Cluster

Falkon on ANL/UC TG Site:

Peak *Dispatch Throughput*: 500/sec

Scalability: 50~500 CPUs

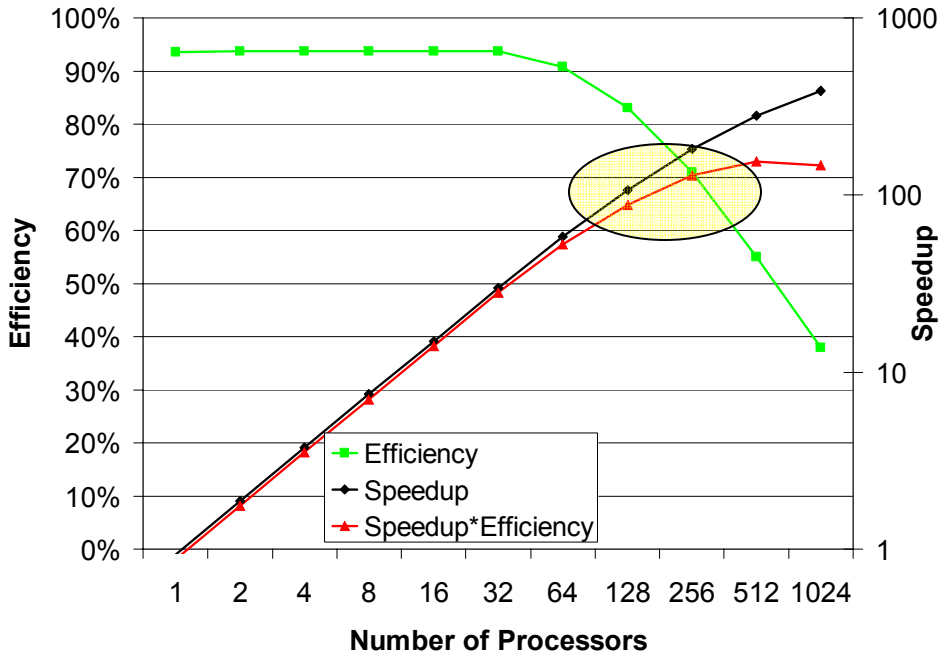
Peak *speedup*: 623x

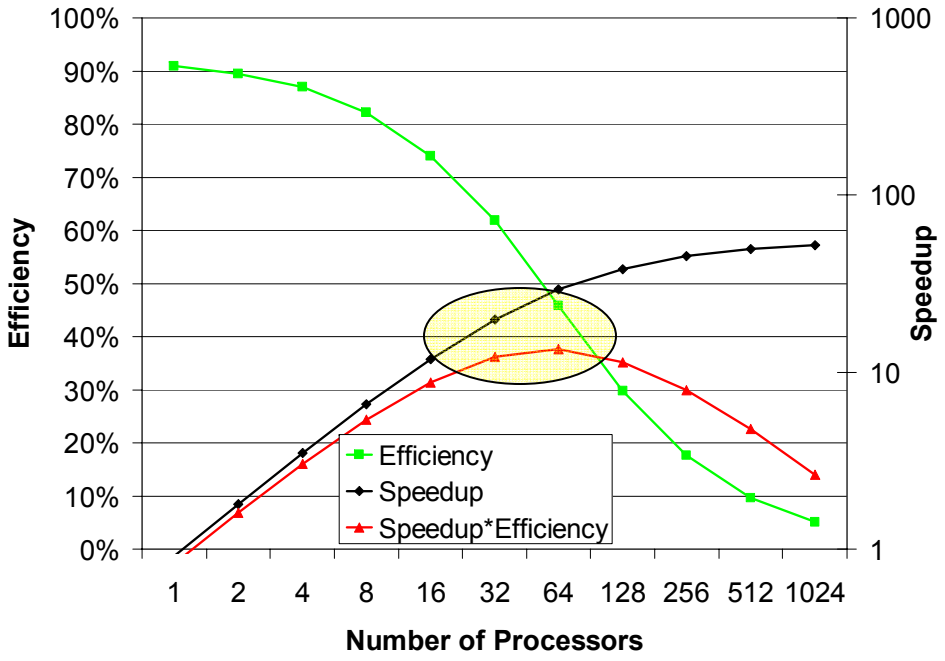
PBS on ANL/UC TG Site:

Peak *Dispatch Throughput*: 1/sec

Scalability: <50 CPUs

Peak *speedup*: 54x





Performance Efficiency Model Example: 128K CPU BG/P

- Application: Angle - distributed data mining
- Testbed Characteristics:
  - Computational Resources: 128K
  - Transient Resource Bandwidth: 10MB/sec
  - Persistent Store Bandwidth: 10000MB/sec
- Workload:
  - Number of Tasks: 128K
  - Arrival rate: 10000/sec
  - Average task execution time: 60 sec
  - Data Object Size: 40MB

Performance Efficiency Model Example: 128K CPU BG/P

Falkon on BG/P:

Peak *Dispatch Throughput* per I/O Node: 40/s

Scalability: 1000~10000 CPUs

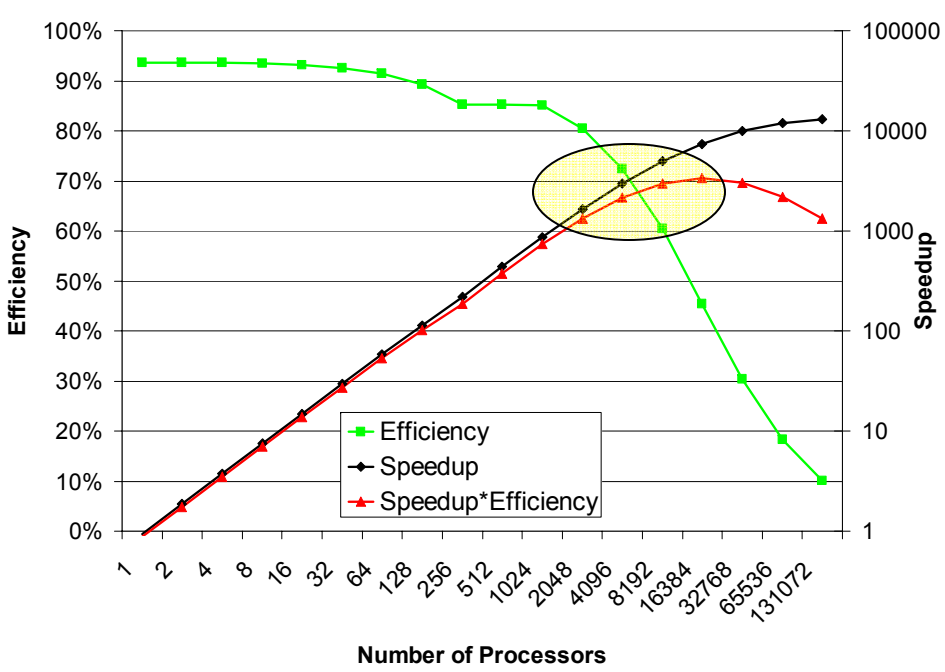
Peak *speedup*: 13153x

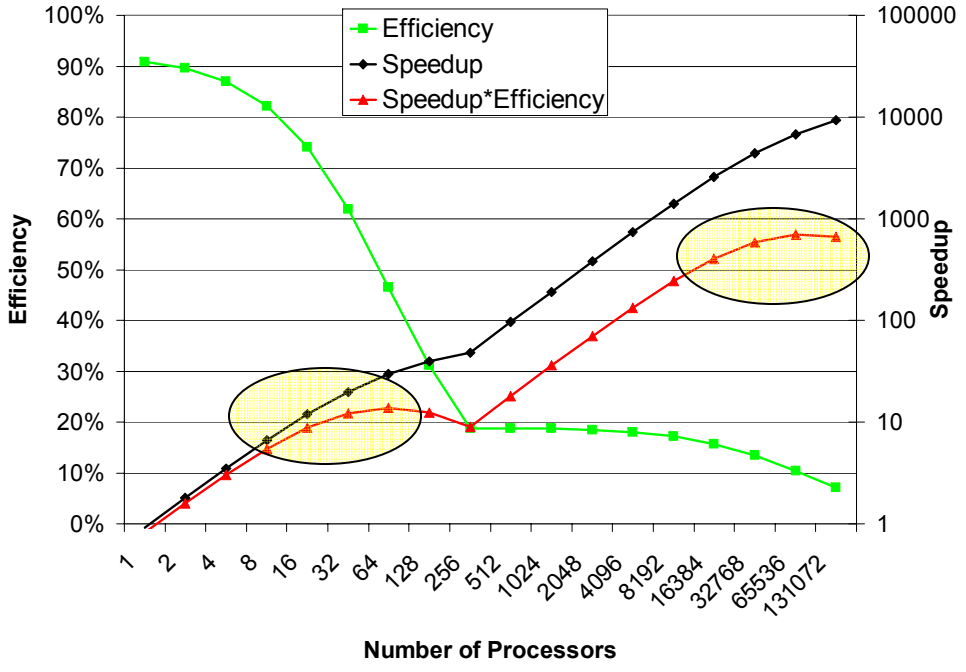
Condor on BG/P:

Peak *Dispatch Throughput* per I/O Node: 1/s

Scalability: ?

Peak *speedup*: 9353x





Model Validation: Simulations

- Implement the abstract model in a discrete event simulation
- Simulation parameters
  - number of storage and computational resources
  - communication costs
  - management overhead
  - workloads (inter-arrival rates, query complexity, data set properties, and data locality)
- Model Validation
  - R^2 Statistic
  - Residual analysis

Outline

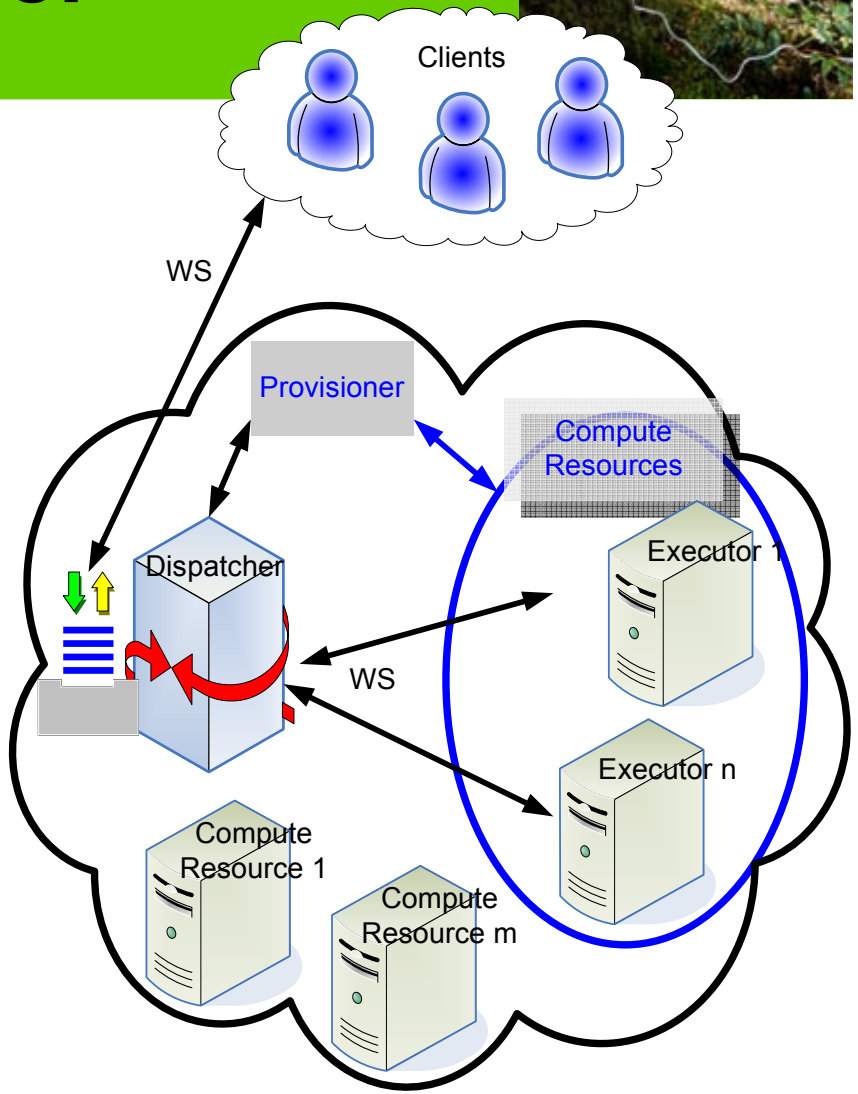
1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

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

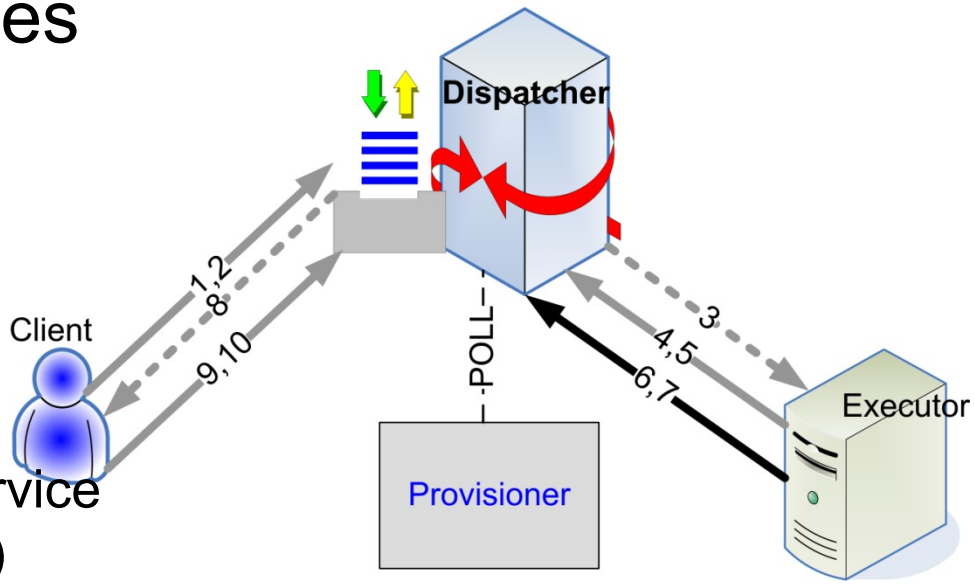
Falkon: The Streamlined Task Dispatcher

- Tier 1: Dispatcher
  - GT4 Web Service accepting task submissions from clients and sending them to available executors
- Tier 2: Executor
  - Run tasks on local resources
- Provisioner
  - Static and dynamic resource provisioning



Falkon: The Streamlined Task Dispatcher

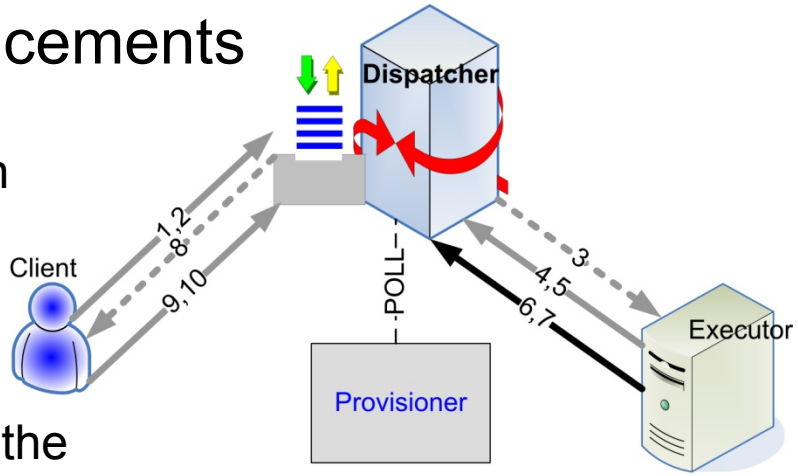
- Falkon Message Exchanges
  - Description:
    - {1}: task(s) submit
    - {2}: task(s) submit confirmation
    - {3}: notification for work
    - {4}: request for task(s)
    - {5 or 7}: dispatch task(s)
    - {6}: deliver task(s) results to service
    - {8}: notification for task result(s)
    - {9}: request for task result(s)
    - {10}: deliver task(s) results to client



  - Worst case (process tasks individually, no optimizations):
    - 4 WS messages ({1,2}, {4,5}, {6,7}, {9,10}) and 2 notifications ({3}, {8}) per task

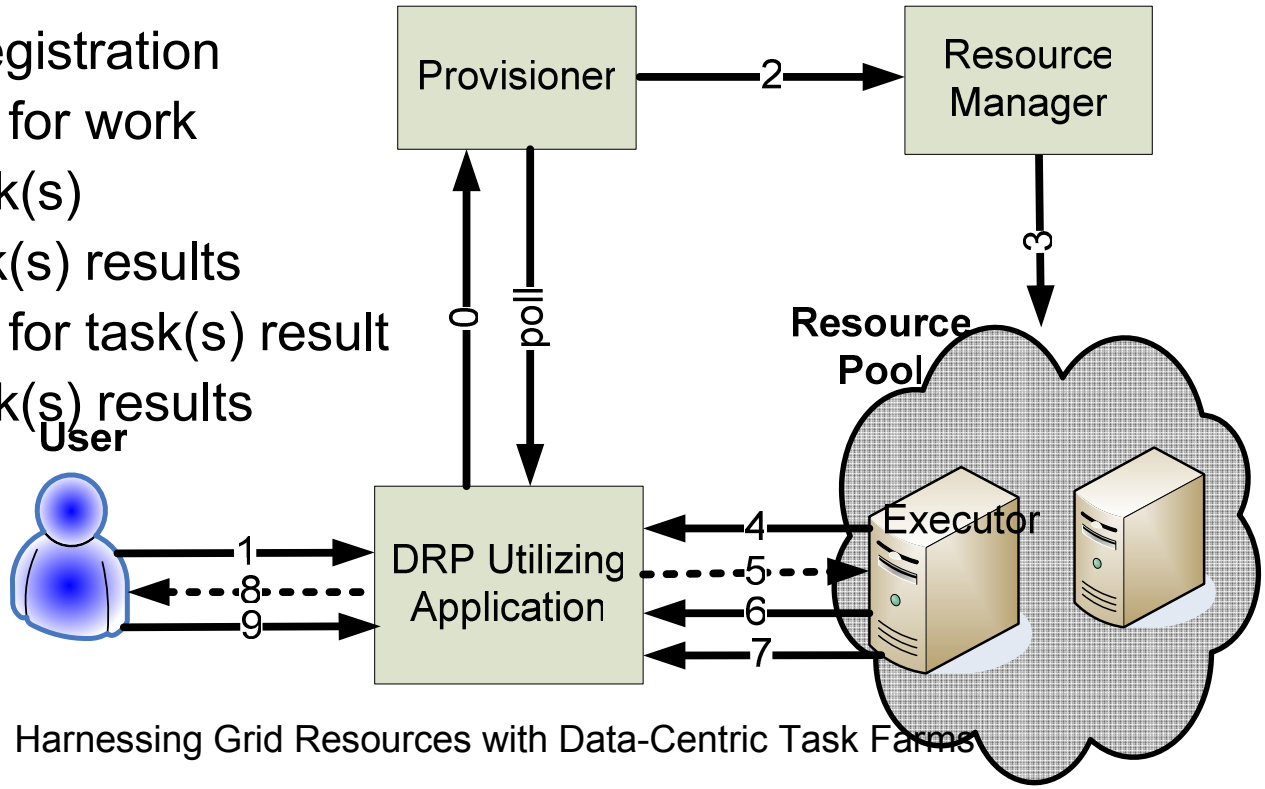
Falkon: The Streamlined Task Dispatcher

- Falkon Message Exchanges Enhancements
  - Bundling
    - Include multiple tasks per communication message
  - Piggy-Backing
    - Attach next task to acknowledgement of previous task
    - Include data management information in the task description and acknowledgement messages
  - Message reduction:
    - General Lower Bound: 10→2+c, where c is a small positive value
    - Application Specific Lower Bound: 10→0+c, where c is a small positive value



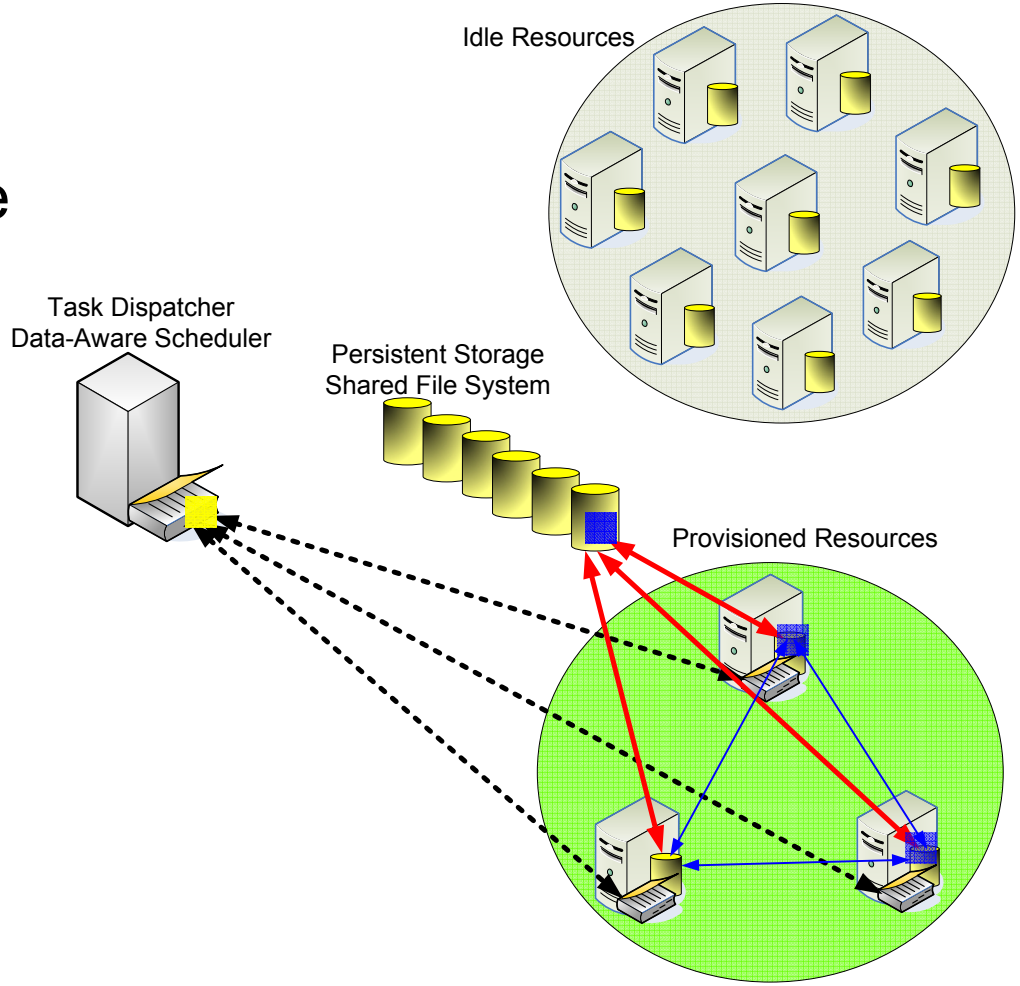
Falkon: Resource Provisioning

0. provisioner registration
1. task(s) submit
2. resource allocation to GRAM
3. resource allocation to LRM
4. executor registration
5. notification for work
6. pick up task(s)
7. deliver task(s) results
8. notification for task(s) result
9. pick up task(s) results



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



Falkon: Data Diffusion

- Considers both data and computations to optimize performance
- Decrease dependency of a shared file system
  - Theoretical linear scalability with compute resources
  - Significantly increases meta-data creation and/or modification performance
- Completes the “data-centric task farm” realization

Outline

1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

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

Related Work: Task Dispatch

- [*Zhou92*]: **LSF** – Load Sharing Cluster Management
- [*Bode00*]: **PBS** – Portable Batch Scheduler and Maui Scheduler
- [*Anderson04*]: **BOINC** – Task Distribution for Volunteer Computing
- [*Thain05*]: **Condor**
- [*Robinson07*]: **Condor-J2** – Turning Cluster Management into Data Management

Conclusion: related work is several orders of magnitude slower

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: Allows dynamic resizing of resource pool (independent of application logic) based on system load and makes use of light-weight task dispatch

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**

Conclusion: Our work focuses on the co-location of storage and computations close to each other (i.e. on the same physical resource) while operating in a dynamic environment.

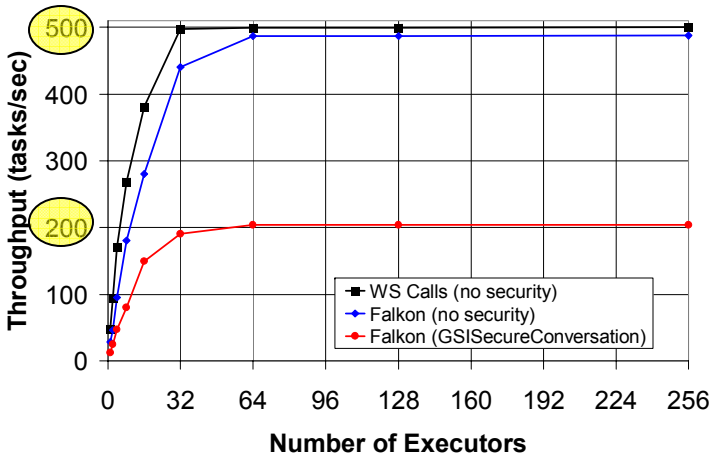
Outline

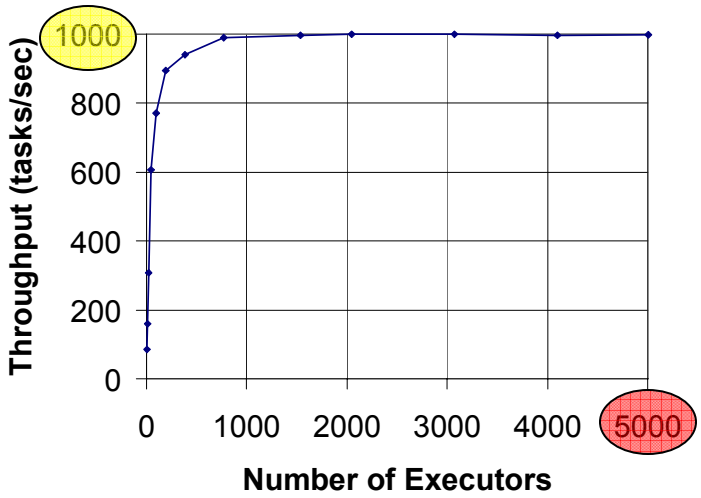
1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. Conclusion & Contributions

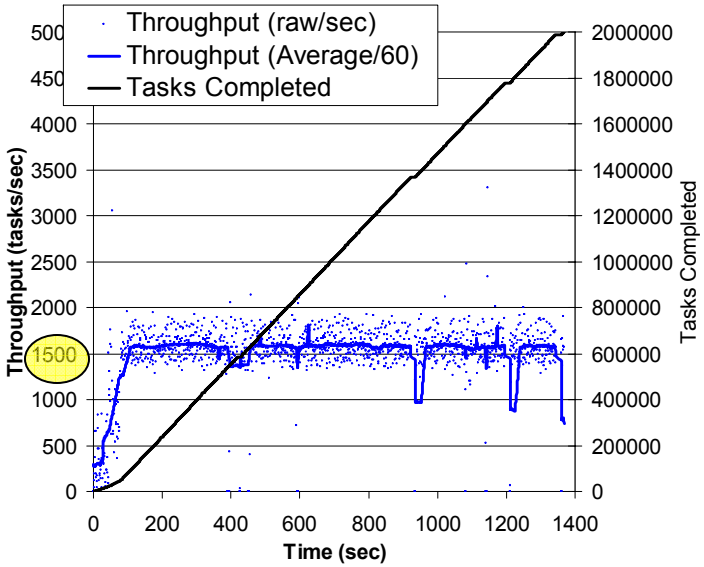
Completed Milestones

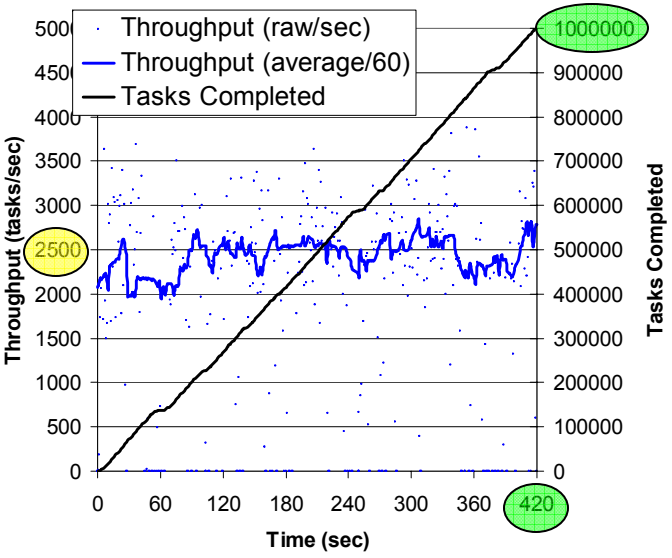
- Abstract task farm model *[Dissertation Proposal 2007]*
- Practical Realization: Falkon
  - Task Dispatcher *[Globus Incubator 2007, SC07]*
  - Resource Provisioning *[SC07, TG07]*
  - Data Diffusion *[NSF06, MSES07]*
  - Swift Integration *[SWF07, NOVA08]*
- Applications *[NASA06, TG06, SC06, NASA07, SWF07, NOVA08]*
  - AstroPortal, Montage, fMRI, MolDyn, Econ

Completed Milestones: Dispatcher Throughput



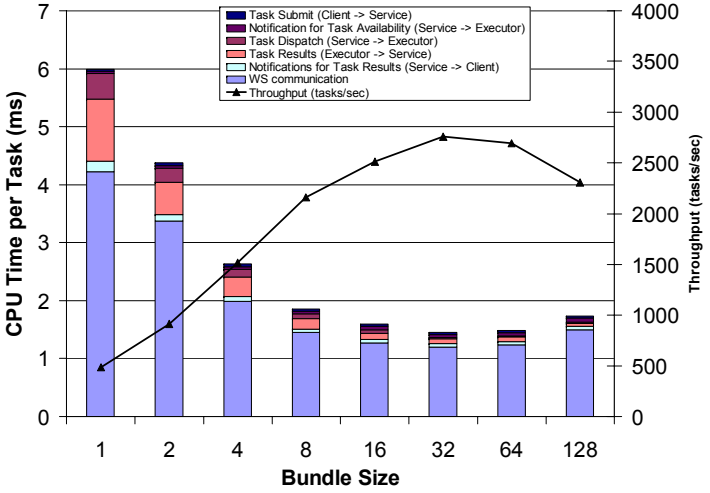


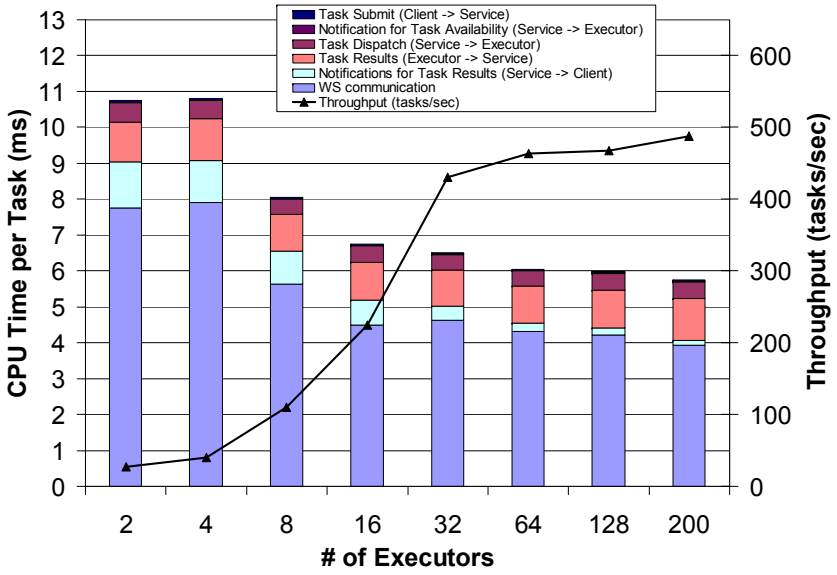


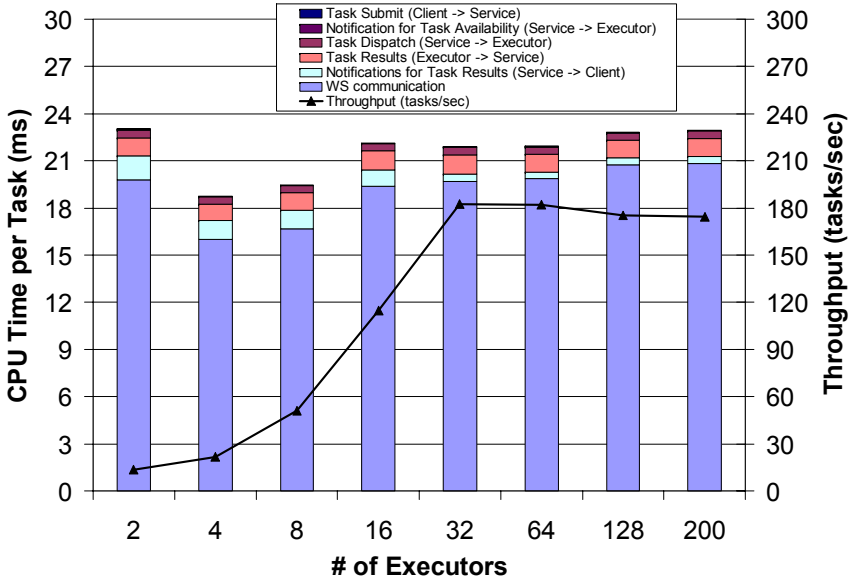


Completed Milestones: Dispatcher Performance Profiling

- **GT:** Java WS-Core 4.0.4
- **Java:** Sun JDK 1.6
- **Machine Hardware:** Dual Xeon 3GHz CPUs with HT
- **Machine OS:** Linux 2.6.13-15.16-smp
- **Executors Location:** ANL/UC TG Site, 100 dual Xeon 2.4GHz CPU nodes, ~2ms latency
- **Workload:** 10000 tasks, "/bin/sleep 0"

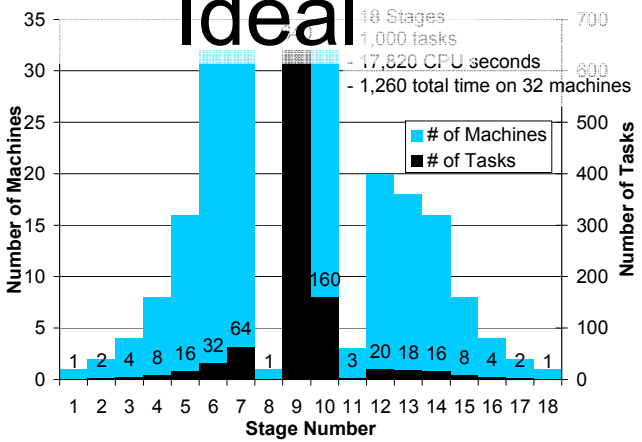




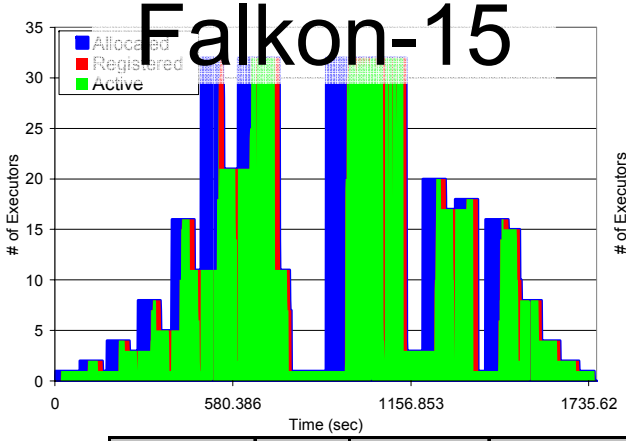


Completed Milestones: Resource Provisioning

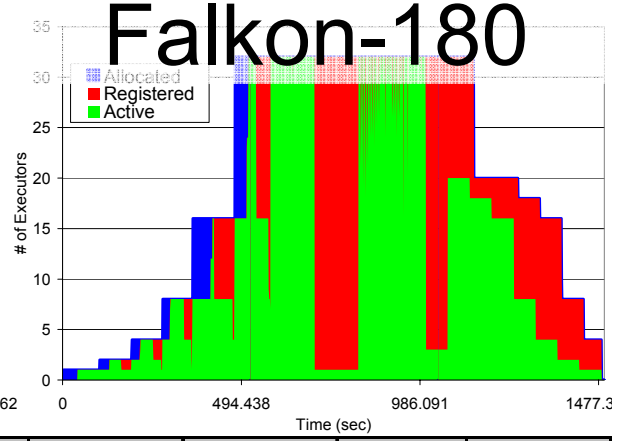
Ideal



Falkon-15



Falkon-180

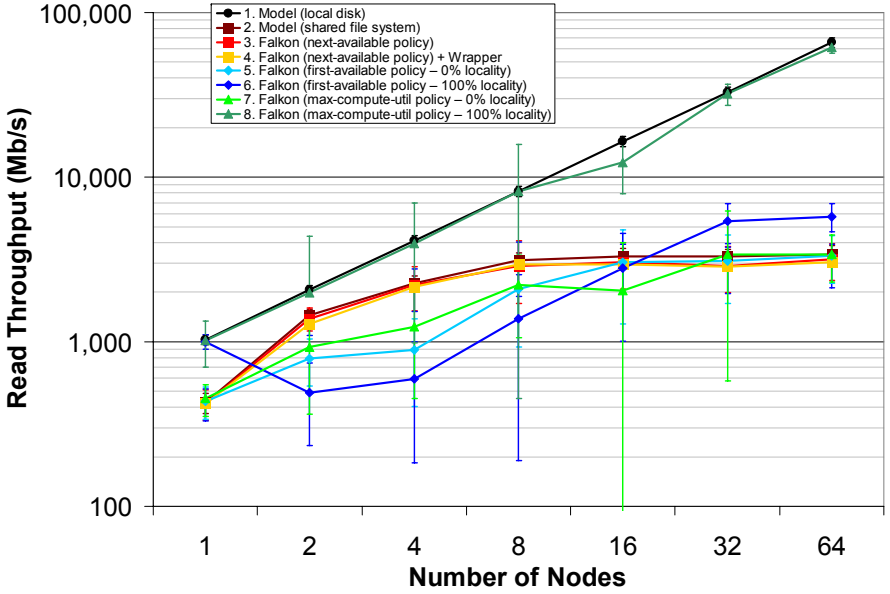


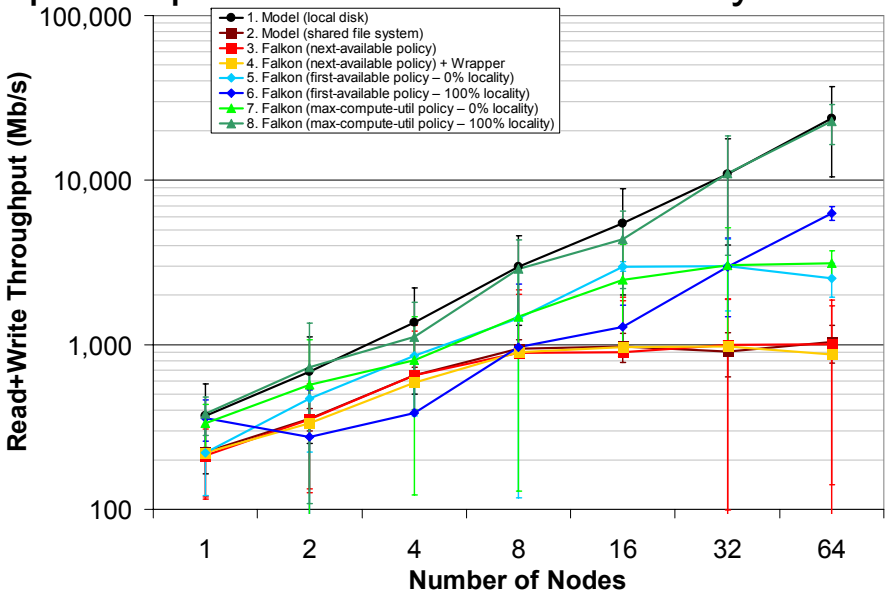
- End-to-end execution time:
  - 1260 sec in ideal case
  - 4904 sec → 1276 sec
- Average task queue time:
  - 42.2 sec in ideal case
  - 611 sec → 43.5 sec
- Trade-off:
  - Resource Utilization for Execution Efficiency

	GRAM +PBS	Falkon-15	Falkon-60	Falkon-120	Falkon-180	Falkon-∞	Ideal (32 nodes)
Queue Time (sec)	611.1	87.3	83.9	74.7	44.4	43.5	42.2
Execution Time (sec)	56.5	17.9	17.9	17.9	17.9	17.9	17.8
Execution Time %	8.5%	17.0%	17.6%	19.3%	28.7%	29.2%	29.7%
	GRAM +PBS	**Falkon-15**	**Falkon-60**	**Falkon-120**	**Falkon-180**	**Falkon-∞**	**Ideal (32 nodes)**
Time to complete (sec)	**4904**	**1754**	**1680**	**1507**	**1484**	**1276**	**1260**
Resouce Utilization	30%	89%	75%	65%	59%	44%	100%
Execution Efficiency	26%	72%	75%	84%	85%	99%	100%
Resource Allocations	1000	11	9	7	6	0	0

Completed Milestones: Data Diffusion

- No Locality
  - Modest loss of read performance for small # of nodes (<8)
  - Comparable performance with large # of nodes
  - Modest gains in read+write performance
- Locality
  - Significant gains in performance beyond 8 nodes
  - Data-aware scheduler achieves near optimal performance and scalability

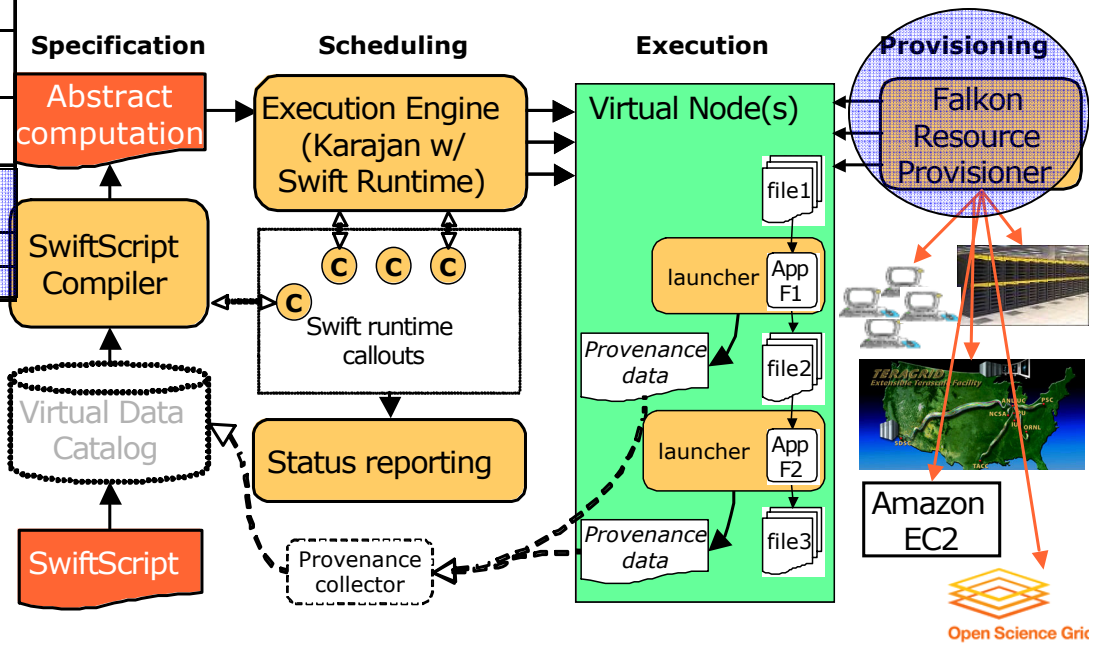




Completed Milestones: Falkon Integration with Swift

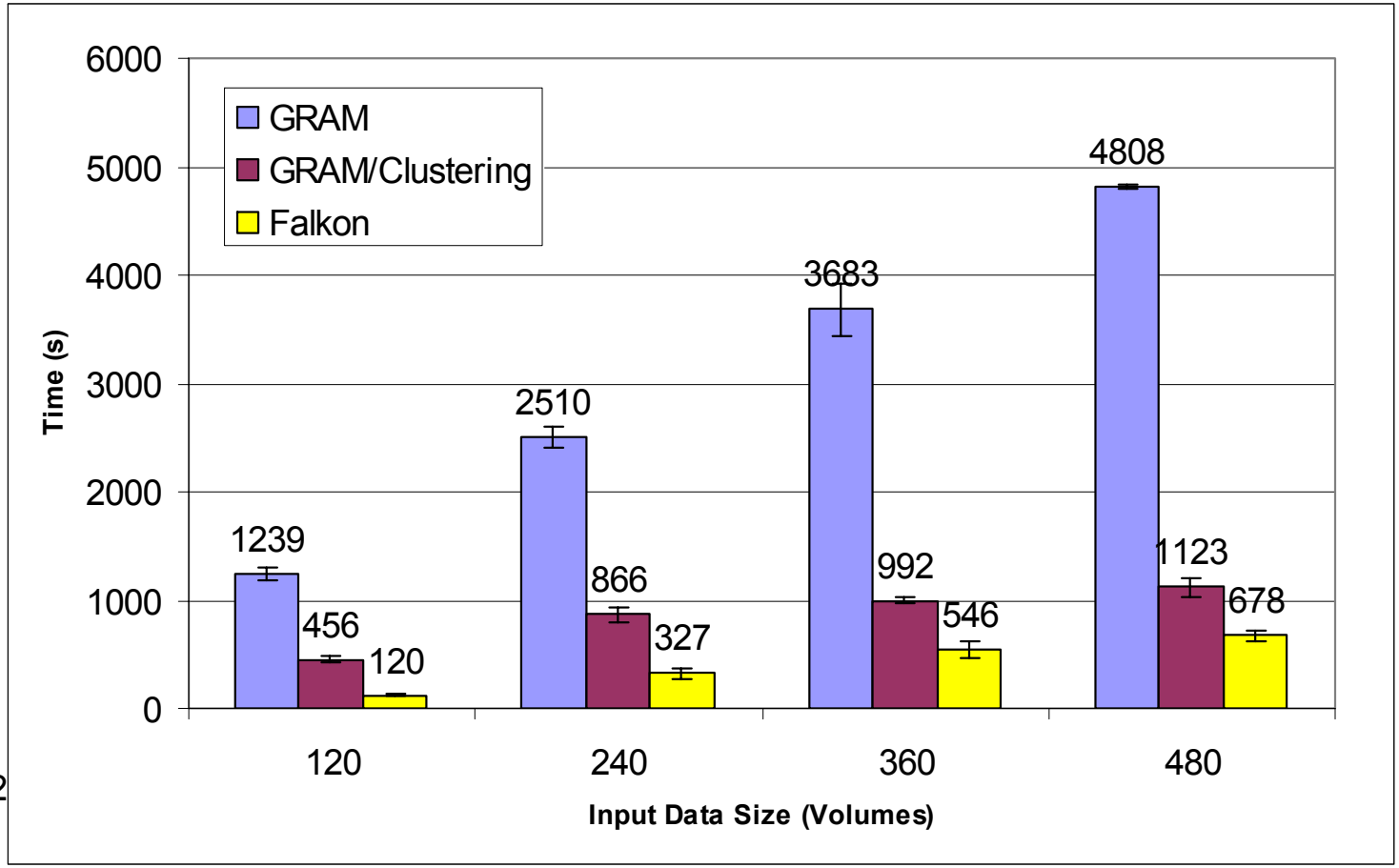
Application	#Tasks/workflow	#Stages
ATLAS: High Energy Physics Event Simulation	500K	1
fMRI DBIC: AIRSN Image Processing	100s	12
FOAM: Ocean/Atmosphere Model	2000	3
GADU: Genomics	40K	4
HNL: fMRI Aphasia Study	500	4
NVO/NASA: Photorealistic Montage/Morphology	1000s	16
QuarkNet/I2U2: Physics Science Education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

Swift Architecture



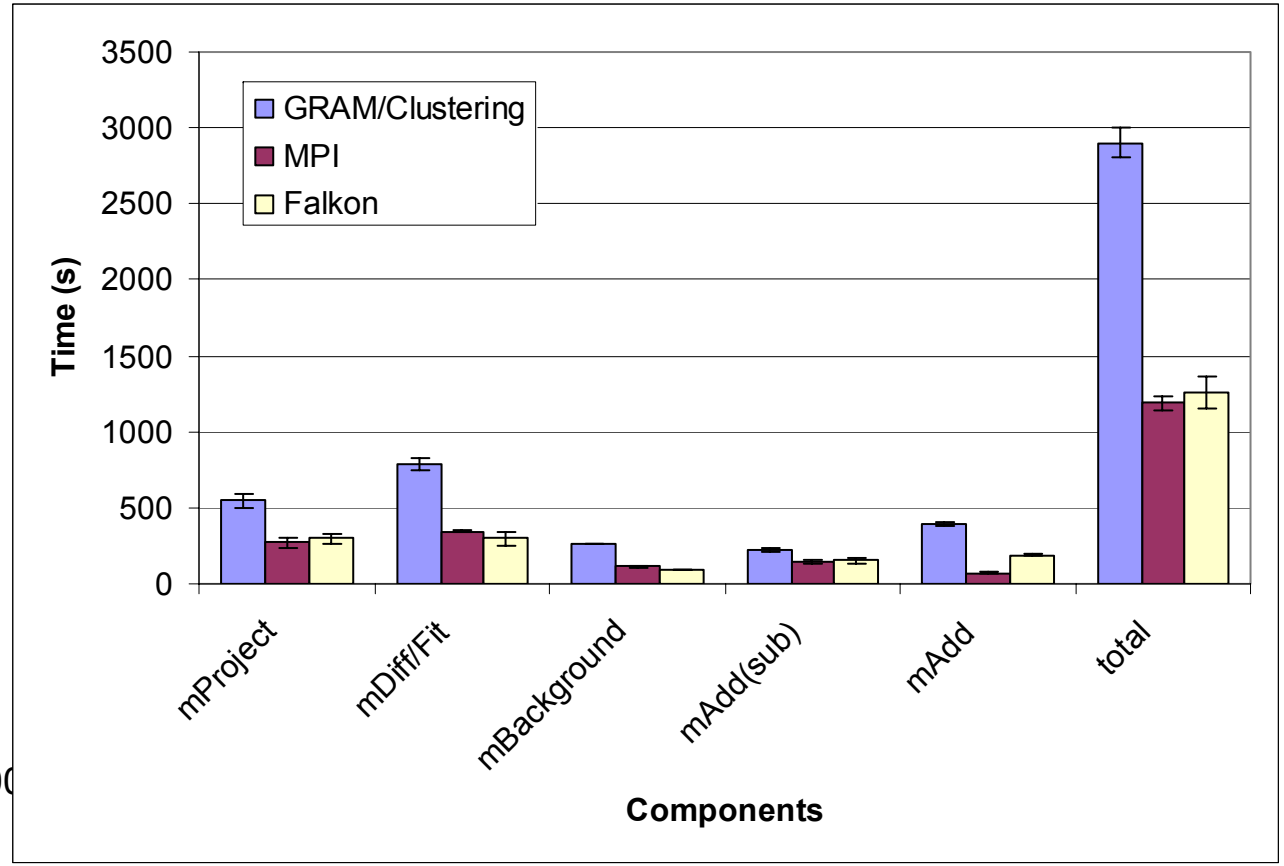
Completed Milestones: fMRI Application

- GRAM vs. Falkon: 85%~90% lower run time
- GRAM/Clustering vs. Falkon: 40%~74% lower run time



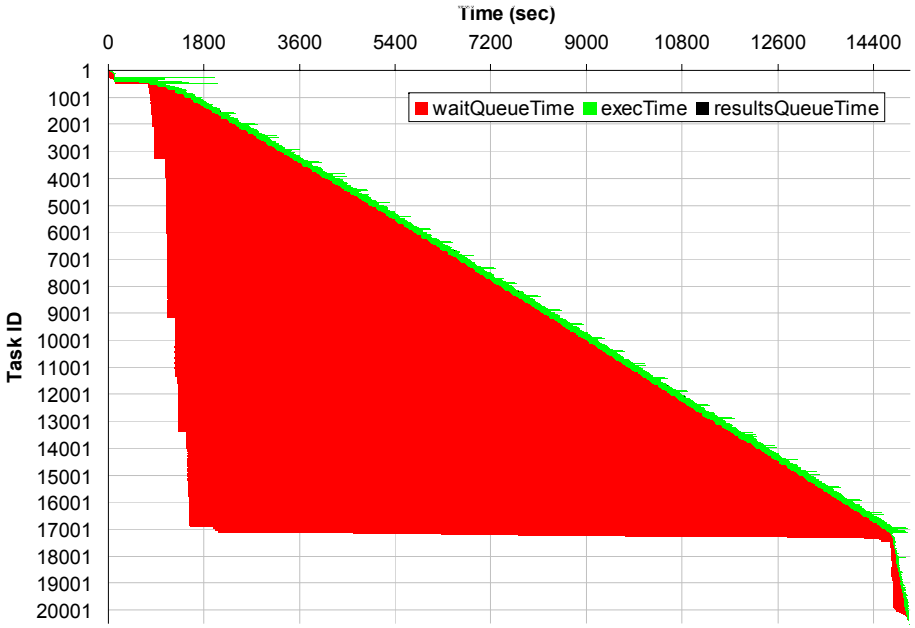
Completed Milestones: Montage Application

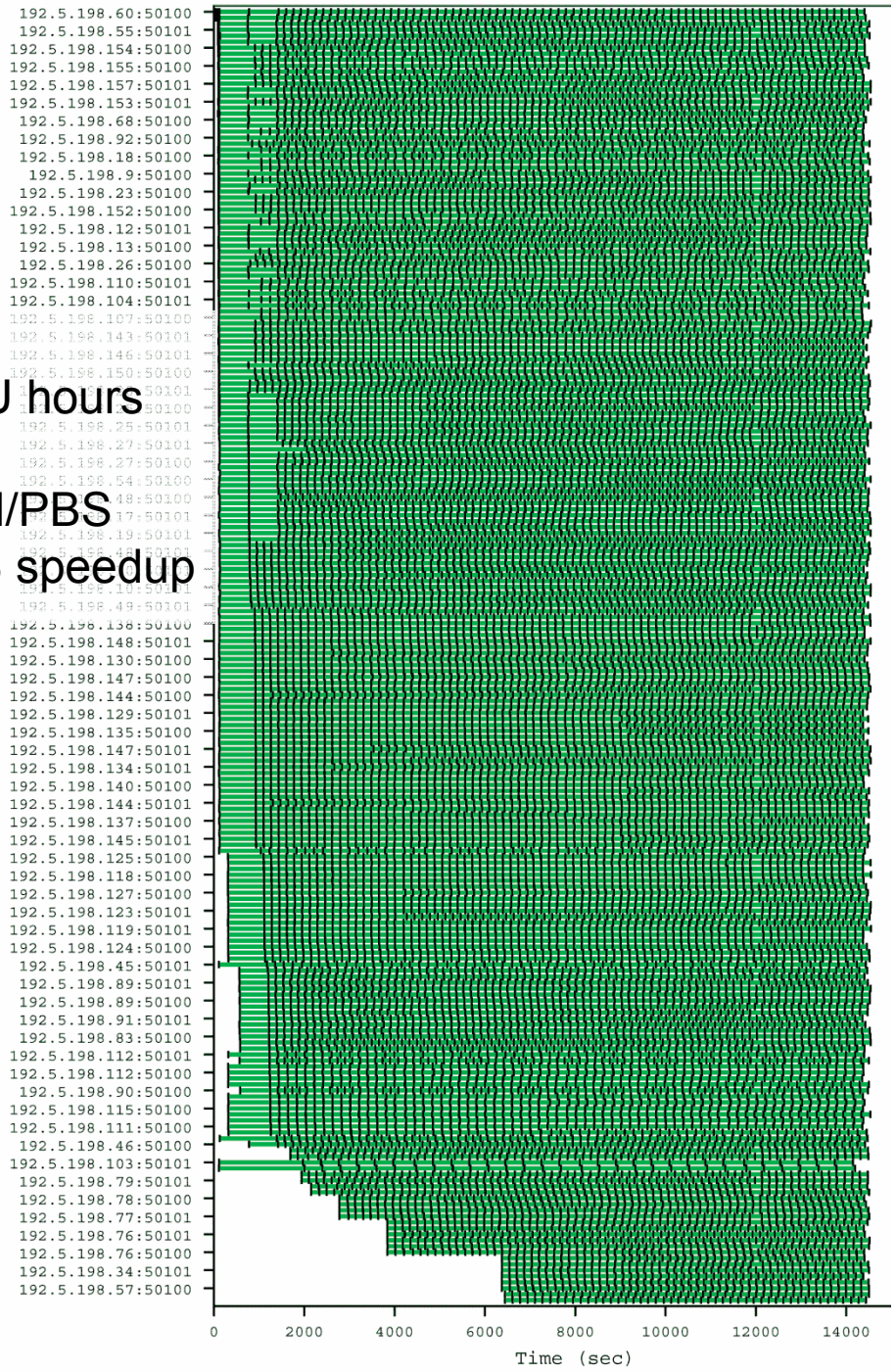
- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound



Completed Milestones: 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

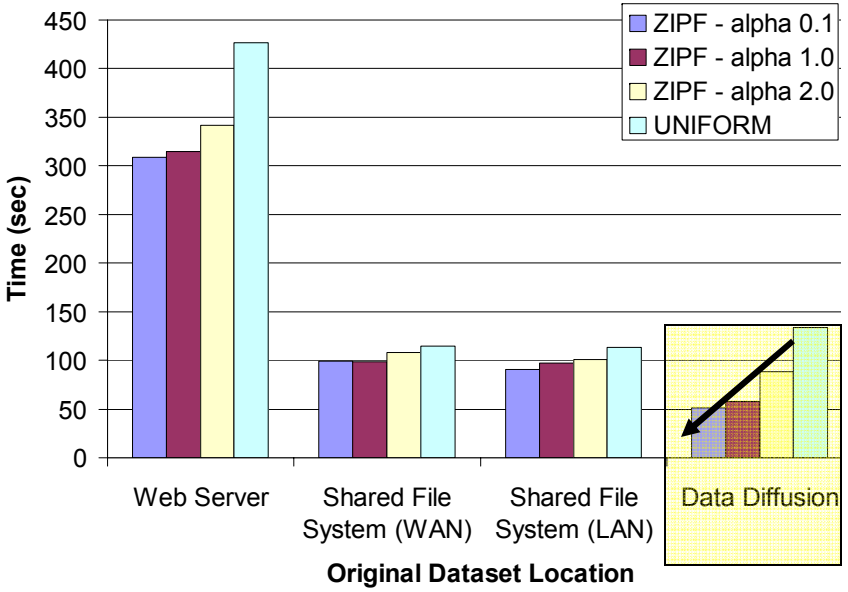


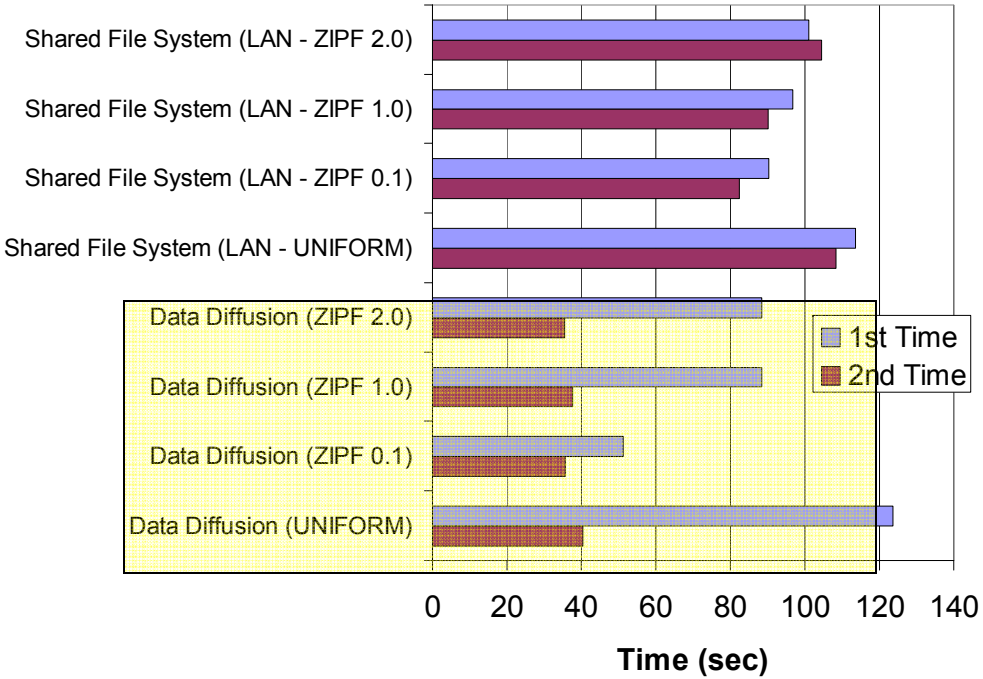


Completed Milestones: AstroPortal Application with Data Diffusion

- No data locality results in 10% lower performance
- Data Locality offers significant (up to 300%) performance improvement with data diffusion

Stacking Size (# of objects)	Object Distribution	Working Set Size (# of files)	Working Set Size (# of objects)	Locality (accesses per file)
10000	ZIPF - alpha 0.1	1915	1924	5.22
10000	ZIPF - alpha 1.0	2755	2819	3.63
10000	ZIPF - alpha 2.0	6110	6259	1.64
10000	UNIFORM	9771	10000	1.02





Outline

1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. **Work in Progress**
6. Conclusion & Contributions

Work in Progress

- Model Validation via Simulations
- Practical Realization
  - Task Dispatcher
  - Resource Provisioning
  - Data Diffusion
  - Performance Evaluation
- Applications

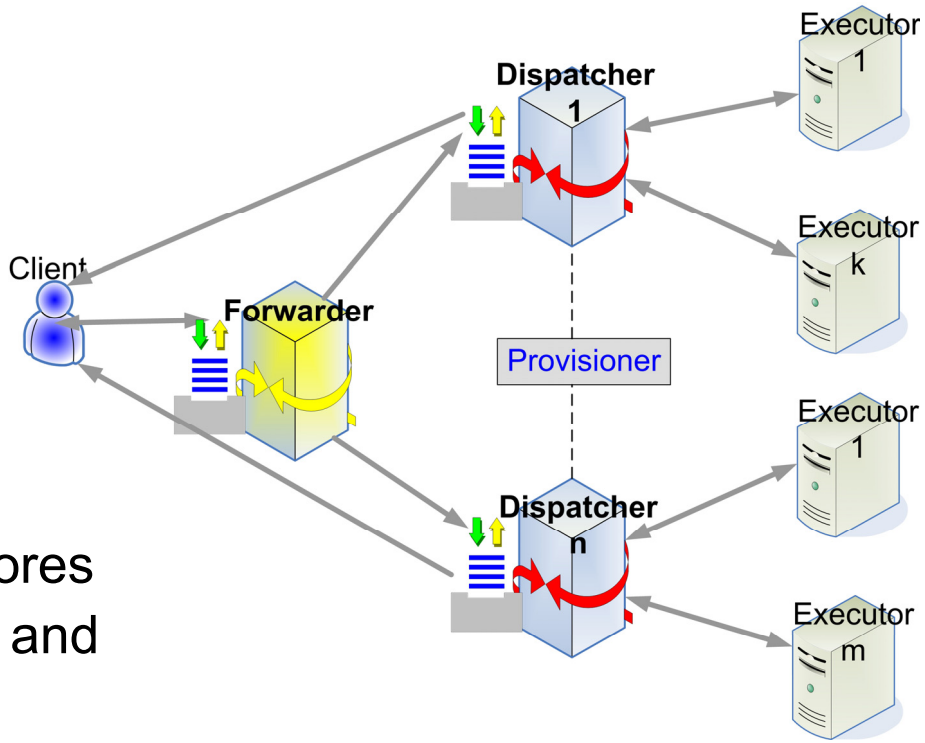
Work in Progress: Simulations

Can data-centric task farms offer good scalability and efficiency?

- Implement the abstract model in a discrete event simulation
  - GridSim simulator from GridBus project
- Model Validation
  - R^2 Statistic
  - Residual analysis

Future Work: 3-Tier Architecture

- Overview
  - Tier 1: Forwarder
  - Tier 2: Dispatcher
  - Tier 3: Executor
- Ensures that Falkon works with local access policies (firewalls, private IP spaces, etc)
- Increases performance and scalability
  - IBM BlueGene/P with 128K CPU cores
  - large multi-site Grids such as OSG and TeraGrid



Work in Progress: Provisioning

- Extend provisioner to support:
  - Virtual Workspace Service (opens door to EC2)
  - Cobalt LRM on the BG/P
  - Multiple sites and make use of wait queue prediction mechanisms
- Establish a permanent Planetlab testbed

Work in Progress: Data Diffusion

- Update Swift to use data diffusion
- Cache eviction policies
- Data-aware schedulers
- Hybrid vs. distributed data management
- Data cache migration on resource de-allocation
- Explore data pre-fetching policies

Work in Progress: Performance Evaluation Metrics

- ***Dispatch*** (Task throughput, Scalability, Resource efficiency)
- ***Provisioning*** (Queue wait time, Dynamic resource provisioning latency, Resource wastage)
- ***Data Management*** (Data caching: cache hits vs. cache misses, Scheduling overheads, Data management overheads)
- ***Applications*** (Execution time, Speedups)
- ***Performance Profiling*** (Quantifying communication overhead)

Work in Progress: Applications & New Science

- Applications (*via Swift)
  - Astronomy: AstroPortal, Montage*
  - Medicine: fMRI*
  - Chemistry: MolDyn*
  - Economics: Econ*
- New Science
  - AstroPortal (stacking service) used to find faint objects in SDSS DR5 dataset

Outline

1. Motivation and Challenges
2. Hypothesis & Proposed Solution
   - Abstract Model
   - Practical Realization
3. Related Work
4. Completed Milestones
5. Work in Progress
6. **Conclusion & Contributions**

Conclusions & Contributions

- Define 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 LRMs
  - 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 flexibility/effectiveness on real world applications
    - astronomy, chemistry, medicine, and economics

More Information

- Related Projects:
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - AstroPortal: http://people.cs.uchicago.edu/~iraicu/research/AstroPortal/index.htm
  - 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
  - Rick Stevens, The University of Chicago & Argonne National Laboratory
  - Yong Zhao, Microsoft
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
  - Catalin Dumitrescu, The University of Chicago
  - Zhao Zhang, The University of Chicago
- Funding:
  - NASA: Ames Research Center, Graduate Student Research Program (GSRP)
  - 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

Publications

1. Y. Zhao, **I. Raicu**, M. Hategan, M. Wilde, I. Foster. "Swift: Realizing Fast, Reliable, Large Scale Scientific Computation", under review at Journal of Future Generation Computer Systems.
2. Y. Zhao, **I. Raicu**, I. Foster, M. Hategan, V. Nefedova, M. Wilde. "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments", to appear as a book chapter in Grid Computing Research Progress, Nova Publisher 2008.
3. **I. Raicu**, Y. Zhao, I. Foster, A. Szalay. "A Data Diffusion Approach to Large Scale Scientific Exploration," Microsoft eScience Workshop at RENCI 2007.
4. **I. Raicu**, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. "Falkon: a Fast and Light-weight tasK executiON framework", IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis (SC07), 2007.
5. Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, **I. Raicu**, T. Stef-Praun, M. Wilde. "Swift: Fast, Reliable, Loosely Coupled Parallel Computation", IEEE Workshop on Scientific Workflows 2007.
6. **I. Raicu**, C. Dumitrescu, I. Foster. "Dynamic Resource Provisioning in Grid Environments", TeraGrid Conference 2007.
7. **I. Raicu**, I. Foster. "Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets: Year 1 Status and Year 2 Proposal", NASA GSRP Year 1 Progress Report and Year 2 Proposal, Ames Research Center, NASA, February 2007.
8. **I. Raicu**, I. Foster. "Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets", NASA GSRP Proposal, Ames Research Center, NASA, February 2006.
9. **I. Raicu**, I. Foster, A. Szalay. "Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets", IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis (SC06), 2006.
10. **I. Raicu**, I. Foster, A. Szalay, G. Turcu. "AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis", TeraGrid Conference 2006.
11. A. Szalay, J. Bunn, J. Gray, I. Foster, **I. Raicu**. "The Importance of Data Locality in Distributed Computing Applications", NSF Workflow Workshop 2006.

Reports

1. **I. Raicu**, I. Foster, Z. Zhang. “Enabling Serial Job Execution on the BlueGene Supercomputer with Falkon,” work in progress, http://www.ci.uchicago.edu/wiki/bin/view/VDS/DslCS/Falkon_BG, 2007.
2. **I. Raicu**, C. Dumitrescu, I. Foster. “Provisioning EC2 Resources,” work in progress, http://www.ci.uchicago.edu/wiki/bin/view/VDS/DslCS/Falkon_EC2, 2007.
3. **I. Raicu**, Y. Zhao, C. Dumitrescu, I. Foster and M. Wilde. “Falkon: A Proposal for Project Globus Incubation”, Globus Incubation Management Project, 2007, http://people.cs.uchicago.edu/~iraicu/research/reports/up/Falkon-GlobusIncubatorProposal_v3.pdf.
4. **I. Raicu**, Y. Zhao, I. Foster, A. Szalay. “Accelerating Large Scale Scientific Exploration through Data Diffusion,” Technical Report, University of Chicago, 2007, http://people.cs.uchicago.edu/~iraicu/research/reports/up/falkon_data-diffusion_v04.pdf.
5. **I. Raicu**, Y. Zhao, C. Dumitrescu, I. Foster, M. Wilde. “Dynamic Resource Provisioning in Grid Environments”, Technical Report, University of Chicago, 2007, http://people.cs.uchicago.edu/~iraicu/research/reports/up/DRP_v01.pdf.
6. **I. Raicu**, I. Foster, A. Szalay. “3DcacheGrid: Dynamic Distributed Data Cache Grid Engine”, Technical Report, University of Chicago, 2006, http://people.cs.uchicago.edu/~iraicu/research/reports/up/HPC_SC_2006_v09.pdf.
7. **I. Raicu**, I. Foster. “Storage and Compute Resource Management via DYRE, 3DcacheGrid, and CompuStore”, Technical Report, University of Chicago, 2006, http://people.cs.uchicago.edu/~iraicu/research/reports/up/Storage_Compute_RM_Performance_06.pdf.
8. **I. Raicu**, I. Foster. “SkyServer Web Service”, Technical Report, University of Chicago, 2006, http://people.cs.uchicago.edu/~iraicu/research/reports/up/SkyServerWS_06.pdf.
9. **I. Raicu**, , I. Foster. “Characterizing the SDSS DR4 Dataset and the SkyServer Workloads,” Technical Report, University of Chicago, 2006, http://people.cs.uchicago.edu/~iraicu/research/reports/up/SkyServer_characterization_2006.pdf
10. **I. Raicu**, , I. Foster. “Characterizing Storage Resources Performance in Accessing the SDSS Dataset,” Technical Report, University of Chicago, 2005, http://people.cs.uchicago.edu/~iraicu/research/reports/up/astro_portal_report_v1.5.pdf.