Accelerating Large Scale Scientific Exploration with Falkon

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

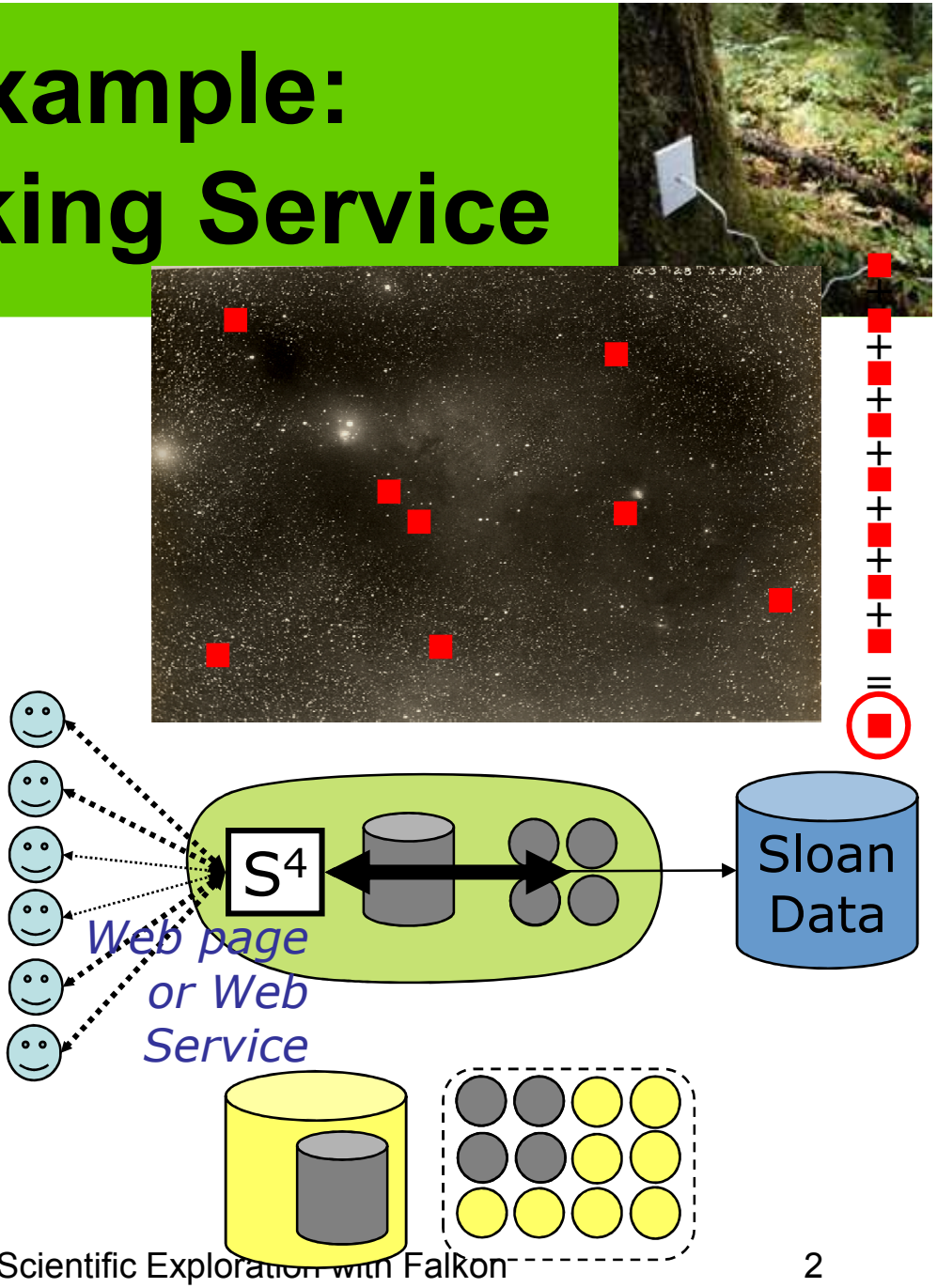
Joint work with:
Ian Foster: University of Chicago, Argonne National Laboratory
Yong Zhao: Microsoft
Alex Szalay: The Johns Hopkins University
Mike Wilde: University of Chicago
Catalin Dumitrescu: University of Chicago
Zhao Zhang: University of Chicago

Funded by:
NSF: TeraGrid
DOE: Advanced Scientific Computing Research
NASA: Ames Research Center

IEEE/ACM Supercomputing 2007
November 14th, 2007

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



Challenges

- Long wait queue times
  - Typically longer than the job duration times
- Slow job dispatch rates
  - ~1 job/sec
- Shared files systems don't scale with compute resources
  - Normally statically configured at 1:8 ~ 1:20 ratio of storage I/O servers to compute resources

Falkon: a Fast and Light-weight tasK executiON framework

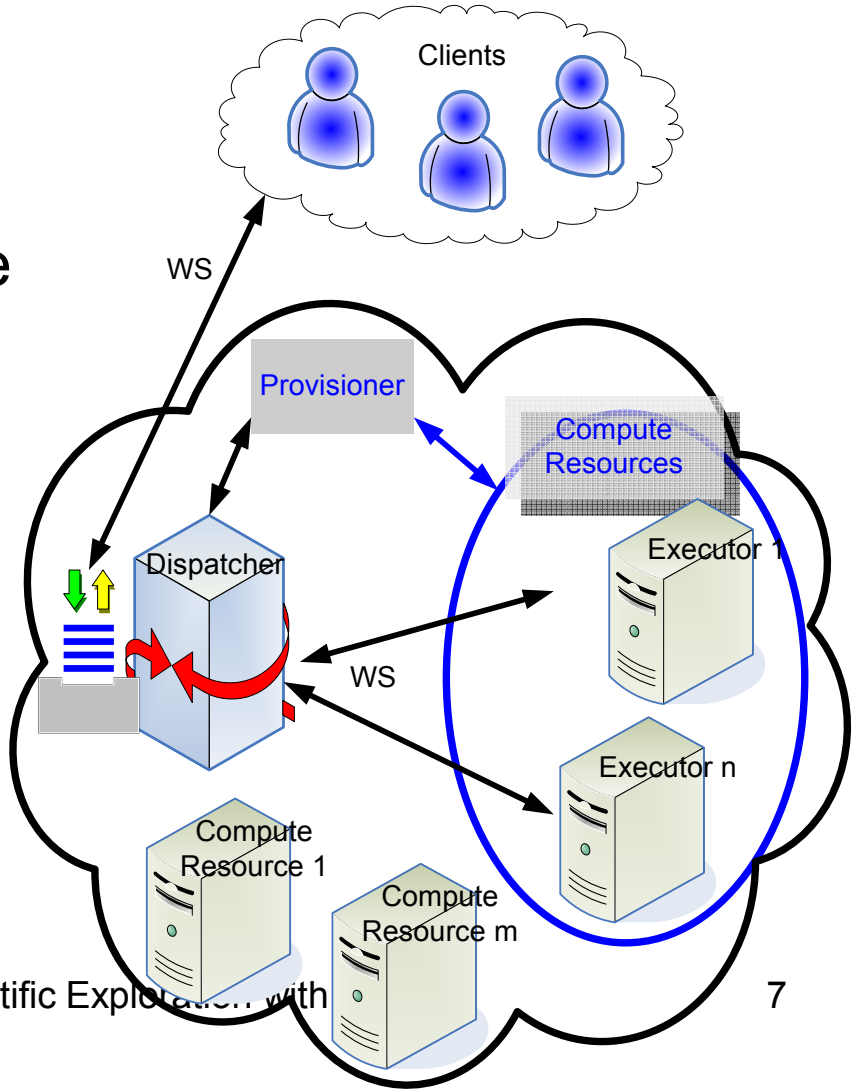
- ***Goal:*** enable the rapid and efficient execution of many independent jobs on large compute clusters
- Combines three techniques:
  - a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
  - ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
  - performs ***data diffusion*** and uses a data-aware scheduler to leverage the co-located computational and storage resources

Goals

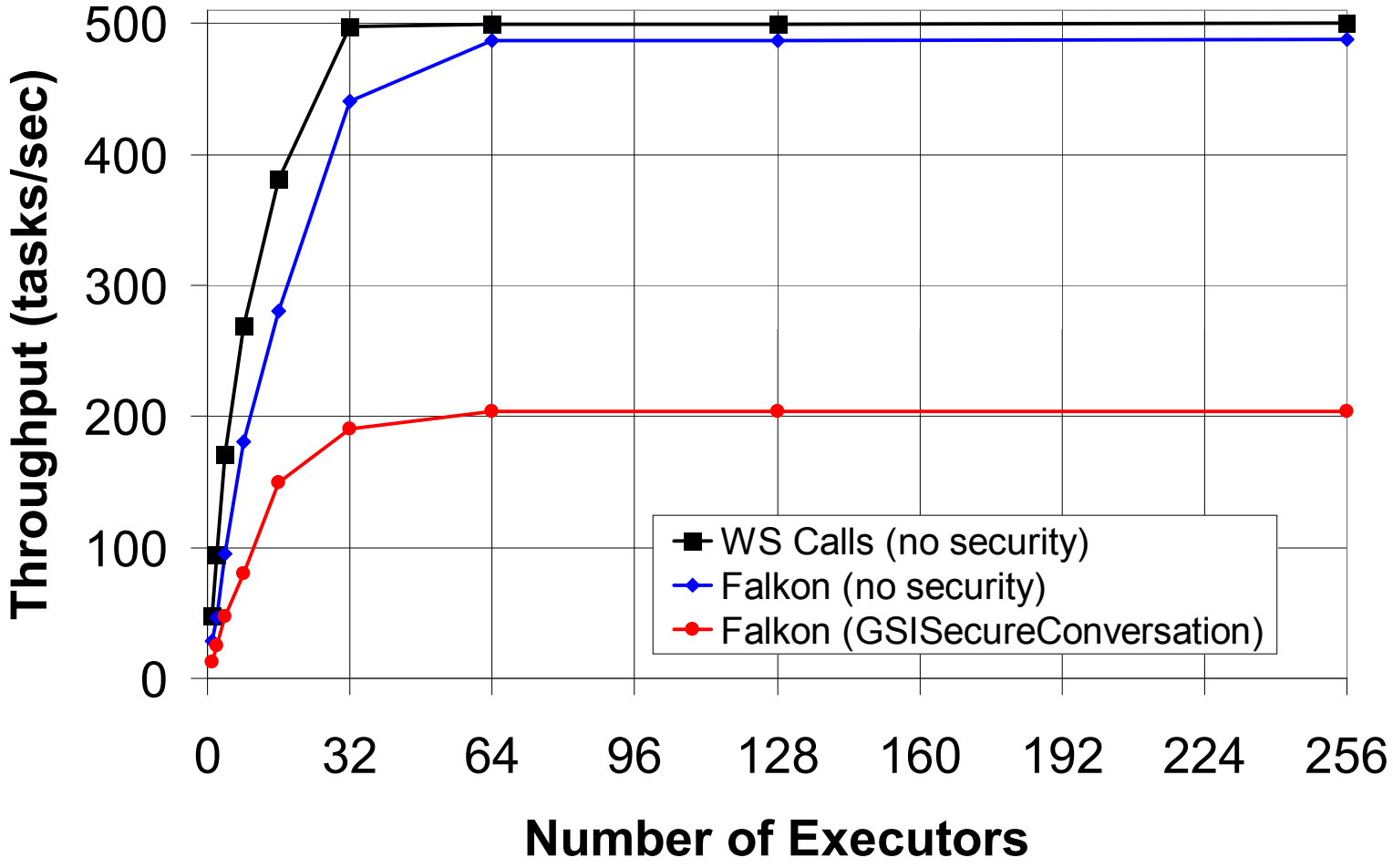
- a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
- ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
- performs ***data diffusion*** and uses a data-aware scheduler to leverage the co-located computational and storage resources

Falkon 2-Tier Architecture

- 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



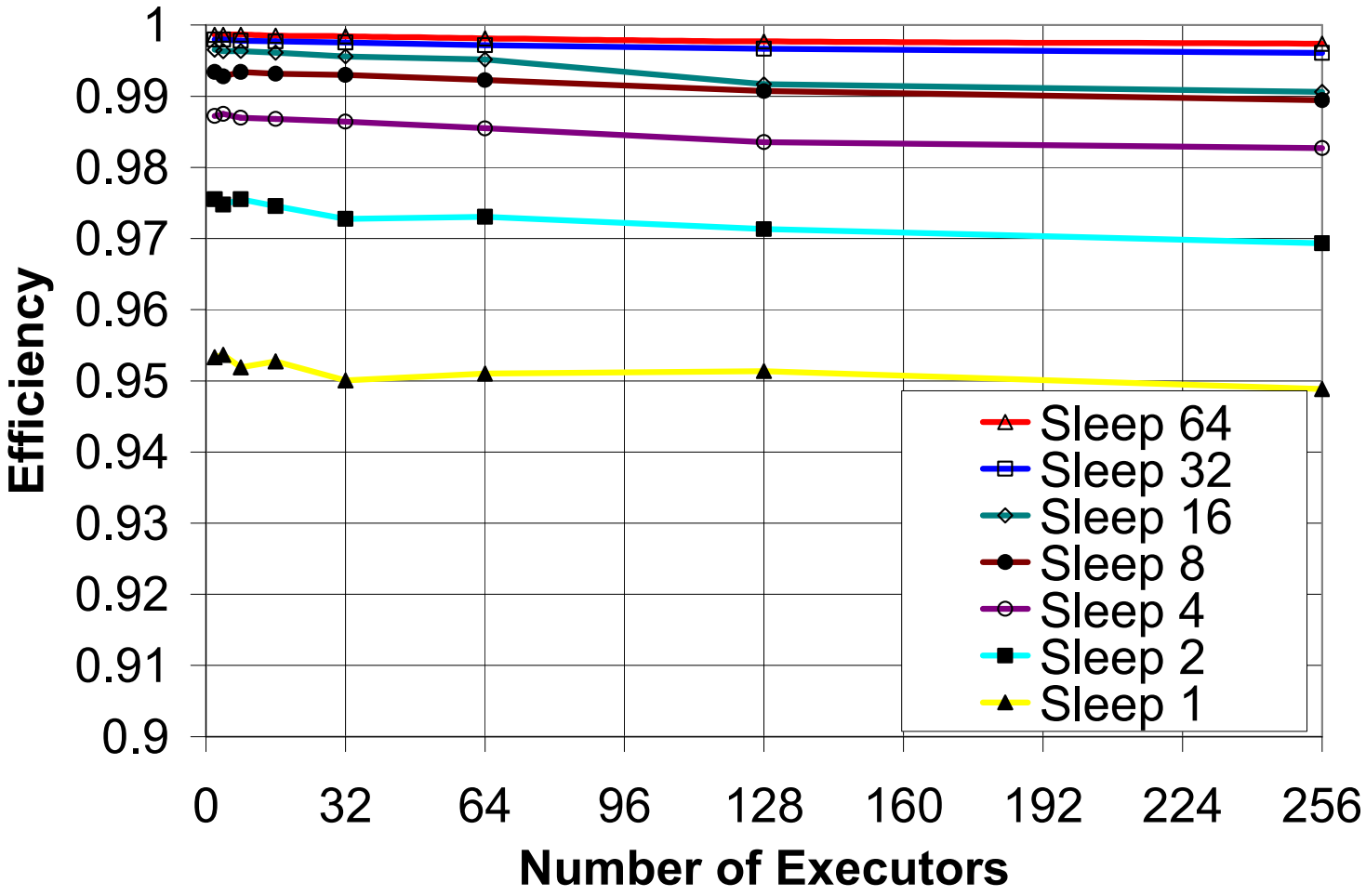
Dispatch Throughput



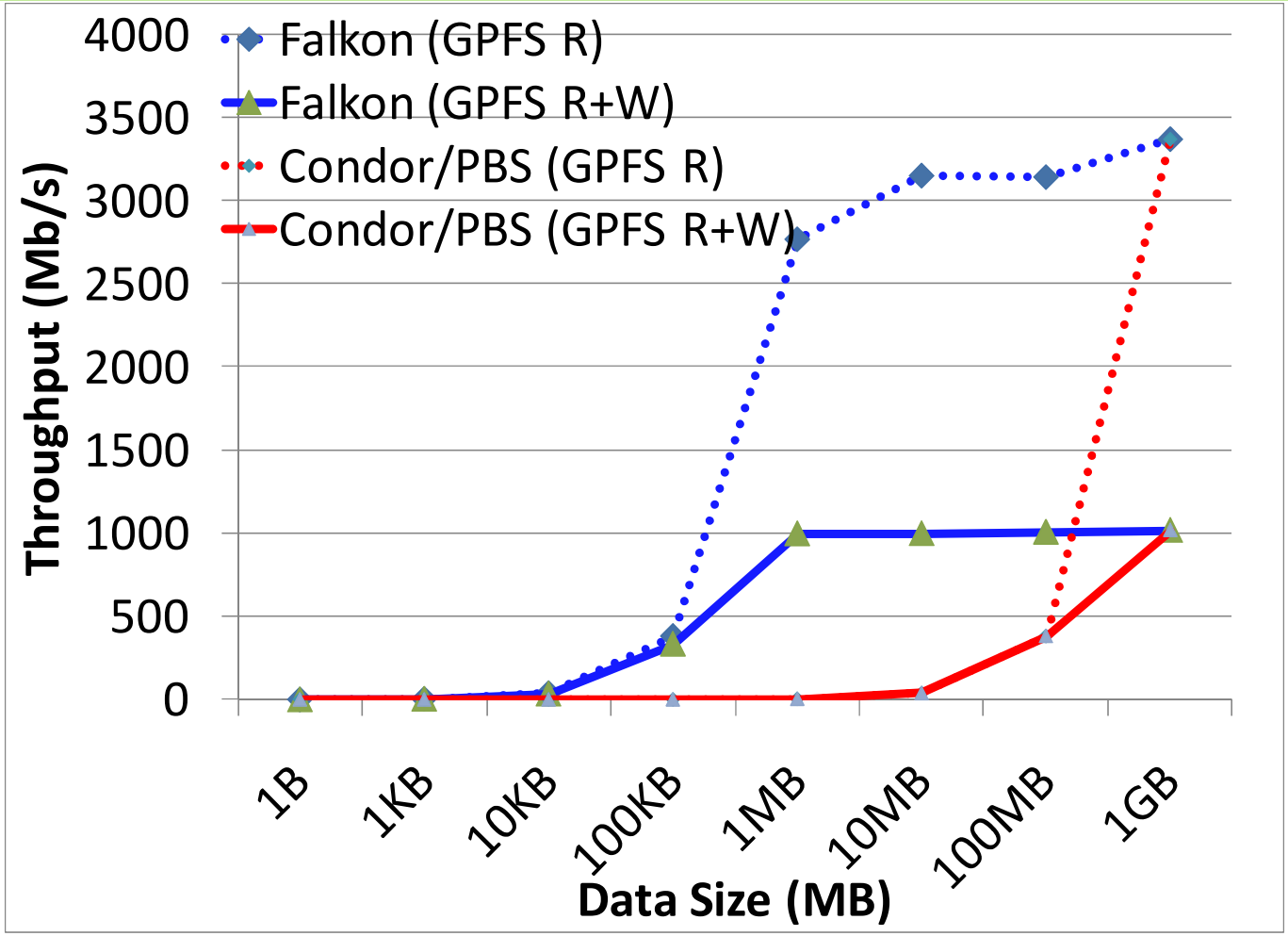
How does Falkon Compare?

System	Comments	Throughput (tasks/sec)
Falkon (no security)	Dual Xeon 3GHz w/ HT 2GB	487
Falkon (GSISecureConversation)	Dual Xeon 3GHz w/ HT 2GB	204
Condor (v6.7.2)	Dual Xeon 2.4GHz, 4GB	0.49
PBS (v2.1.8)	Dual Xeon 2.4GHz, 4GB	0.45
Condor (v6.7.2) [15]	Quad Xeon 3 GHz, 4GB	2
Condor (v6.8.2) [34]		0.42
Condor (v6.9.3) [34]		11
Condor-J2 [15]	Quad Xeon 3 GHz, 4GB	22
BOINC [19, 20]	Dual Xeon 2.4GHz, 2GB	93

Execution Efficiency



How does Dispatch Rates Affect Throughput I/O?

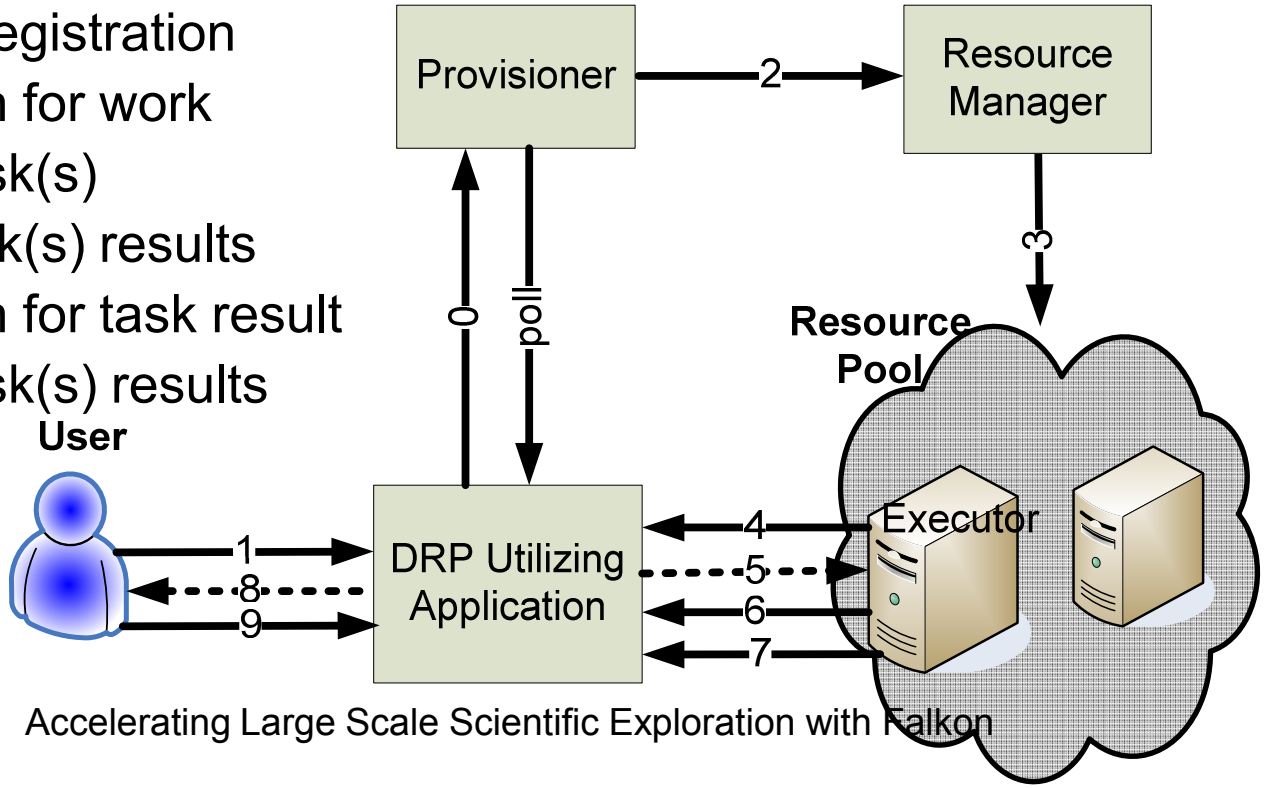


Goals

- a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
- ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
- performs ***data diffusion*** and uses a data-aware scheduler to leverage the co-located computational and storage resources

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 result
9. pick up task(s) results



Dynamic Resource Provisioning

- 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%

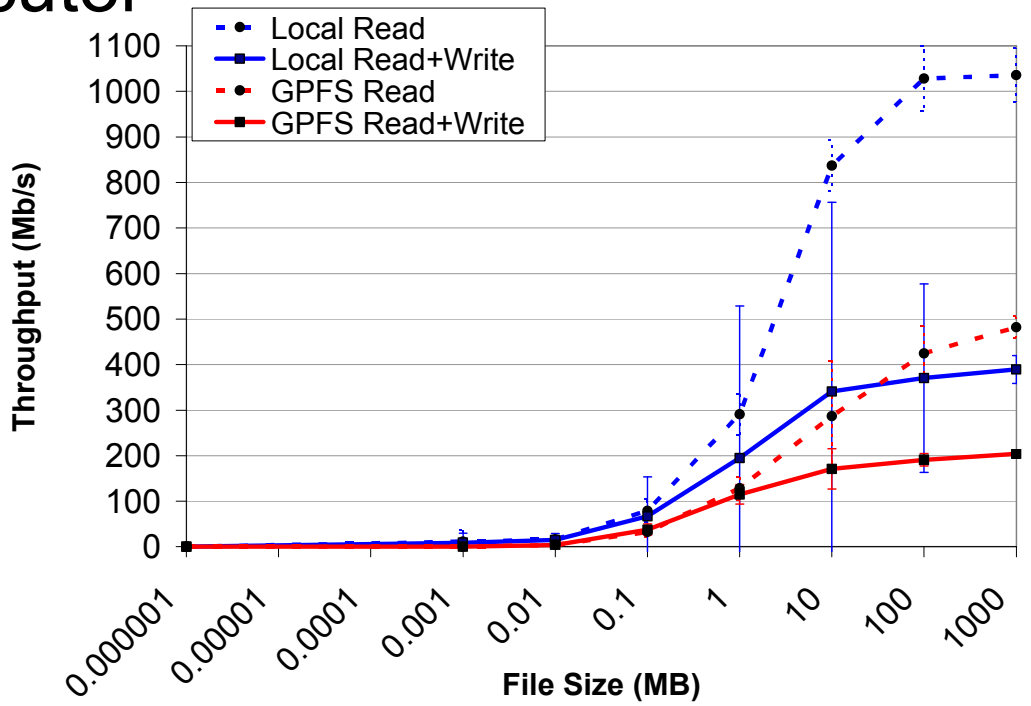
	GRAM +PBS	Falkon-15	Falkon-60	Falkon-120	Falkon-180	Falkon-∞	Ideal (32 nodes)
Time to complete (sec)	**4904**	**1754**	**1680**	**1507**	**1484**	**1276**	**1260**
Resouce Utilization	30%	89%	75%	65%	59%	44%	100%
Execution Efficiency	26%	72%	75%	84%	85%	99%	100%
Resource Allocations	1000	11	9	7	6	0	0

Goals

- a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
- ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
- performs ***data diffusion*** and uses a data-aware scheduler to leverage the co-located computational and storage resources

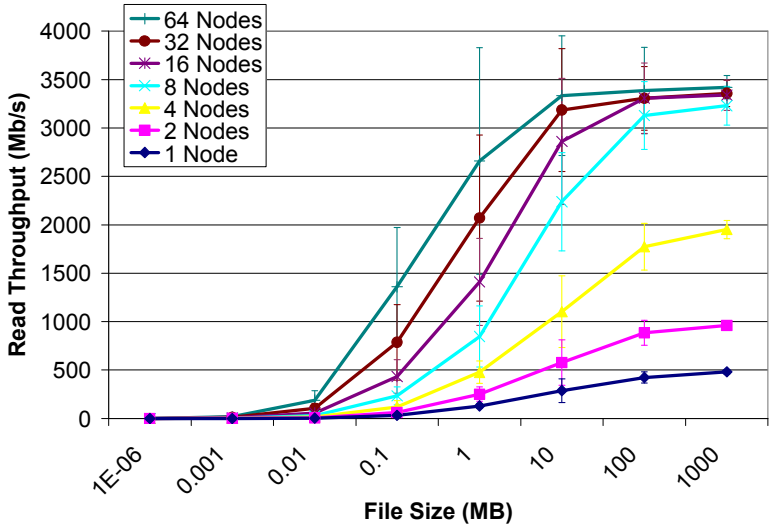
Data Diffusion

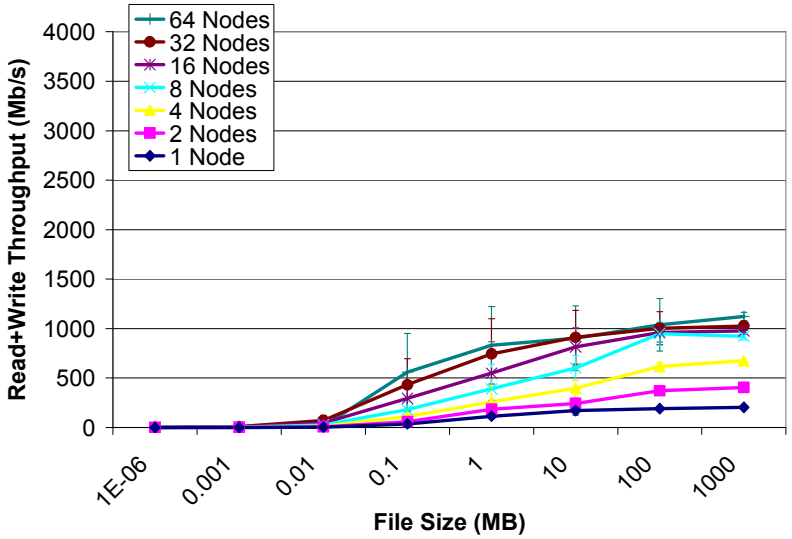
- Data caching at executor local disk
  - RANDOM
  - FIFO
  - LRU
  - LFU
- Avoid the need for a shared file system
- Theoretical linear scalability with compute resources



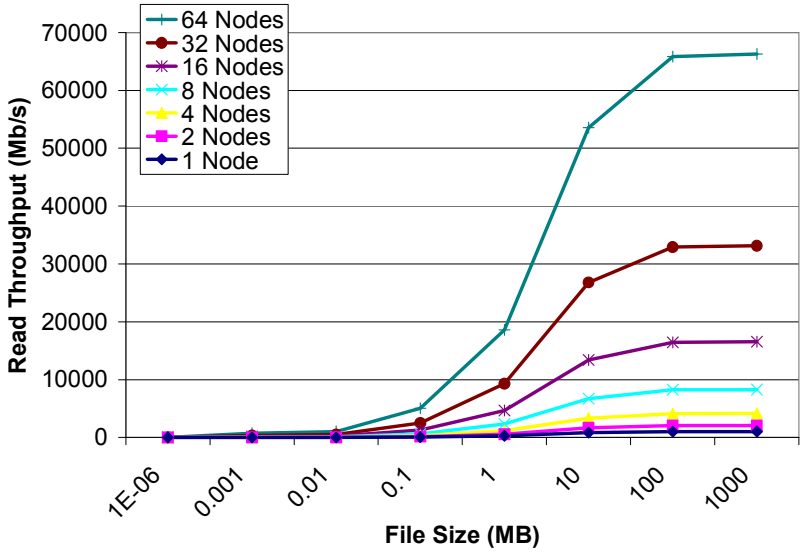
Shared vs. Local File System Performance

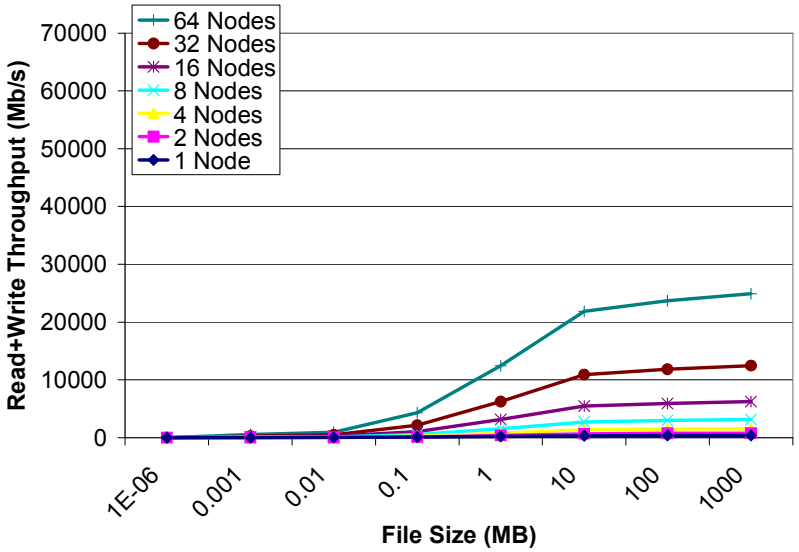
Shared File System





Local Disk

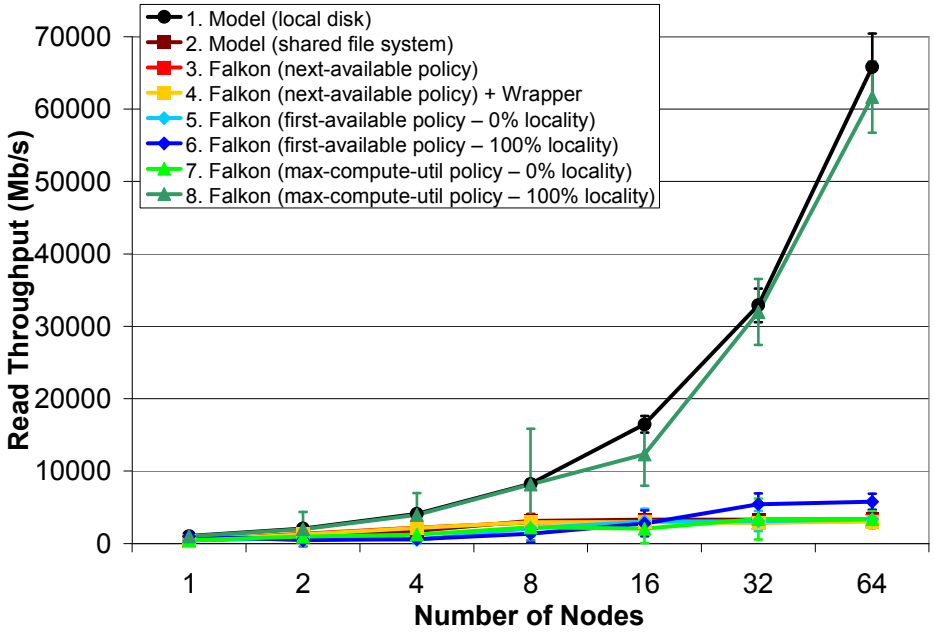


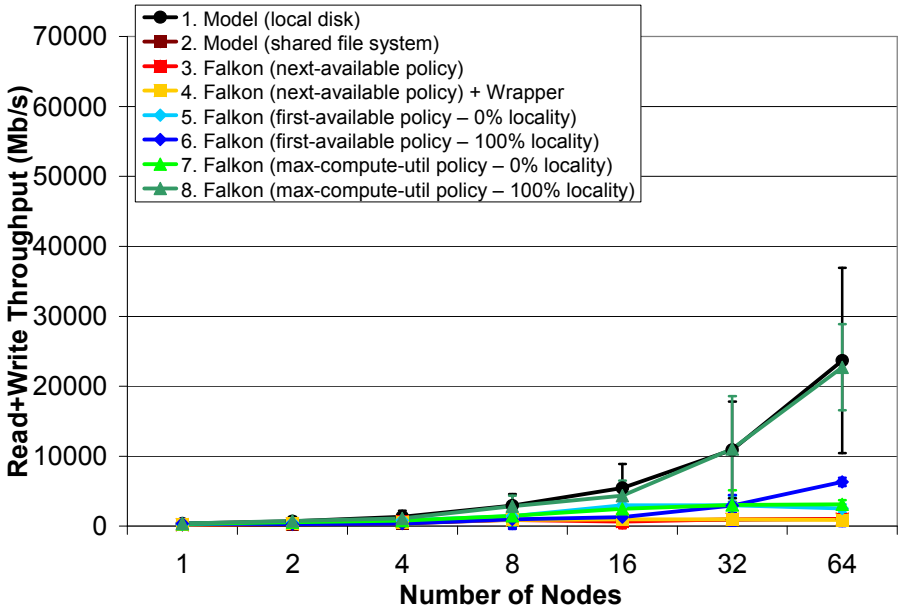


Data Diffusion: Micro-Benchmarks

1. **Model (local disk)**: local disk performance model
2. **Model (shared file system)**: shared file system (GPFS) performance model
3. **Falkon (next-available policy)**: load balancing across available executors, operates on shared file system
4. **Falkon (next-available policy) + Wrapper**: same as (3), but tasks execute through a wrapper that creates a temp scratch directory on the shared file system, makes symbolic links, executes task, and removes the temp scratch directory and symbolic links
5. **Falkon (first-available policy – 0% locality)**: load balancing across available executors with data caching; 0% data locality with data read from the shared file system
6. **Falkon (first-available policy – 100% locality)**: same as (5), but with the workload from (5) repeating four times; the caches are first populated (not as part of the timed experiment)
7. **Falkon (max-compute-util policy – 0% locality)**: same as (5), but uses the data-aware scheduler to send tasks to the executors that have the most data cached; the workload has 0% data locality
8. **Falkon (max-compute-util policy – 100% locality)**: same as (7), but with the workload repeated four times

Micro-Benchmarks Results: 1-64 Nodes

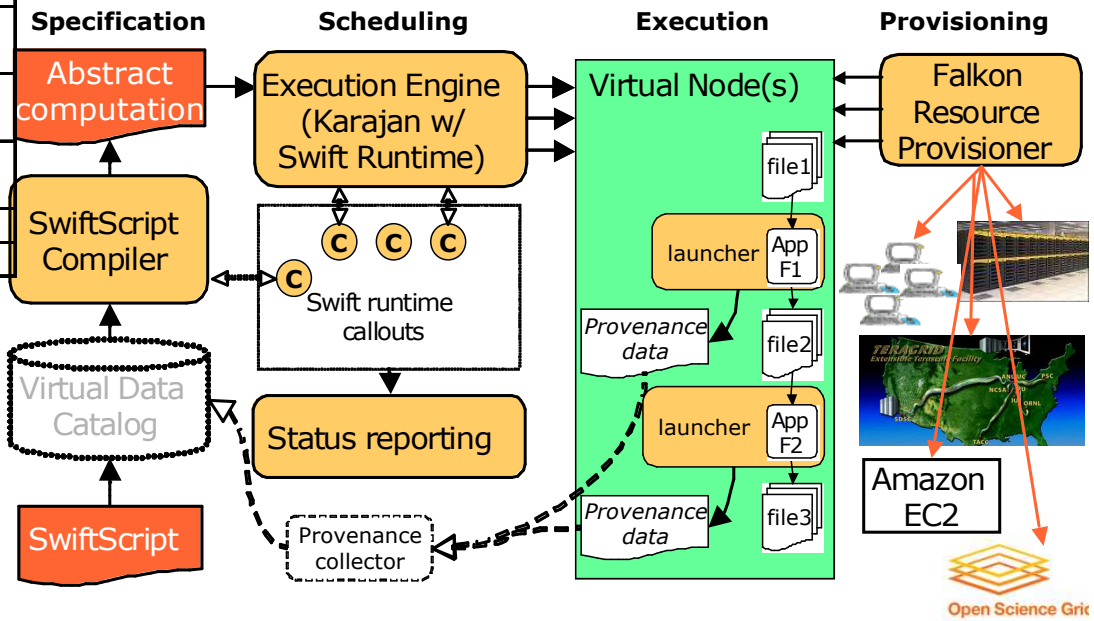




Applications via Swift

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

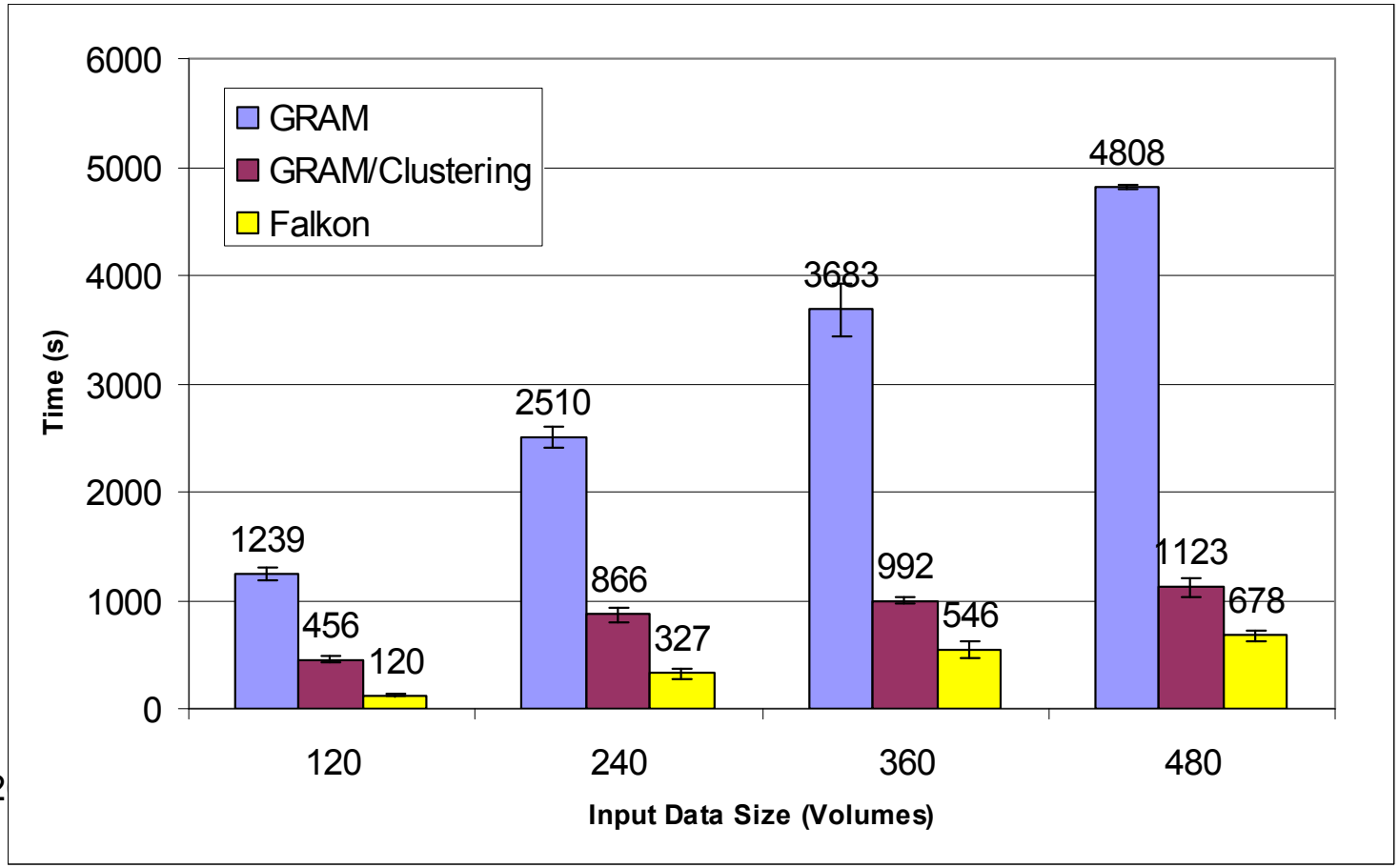


Applications

- Medicine:
  - fMRI (Falkon vs. GRAM)
- Astronomy:
  - Montage (Falkon vs. GRAM vs. MPI)
  - Stacking (Falkon with Data Diffusion vs. Falkon using shared file system)
- Chemistry
  - Molecular Dynamics Simulation

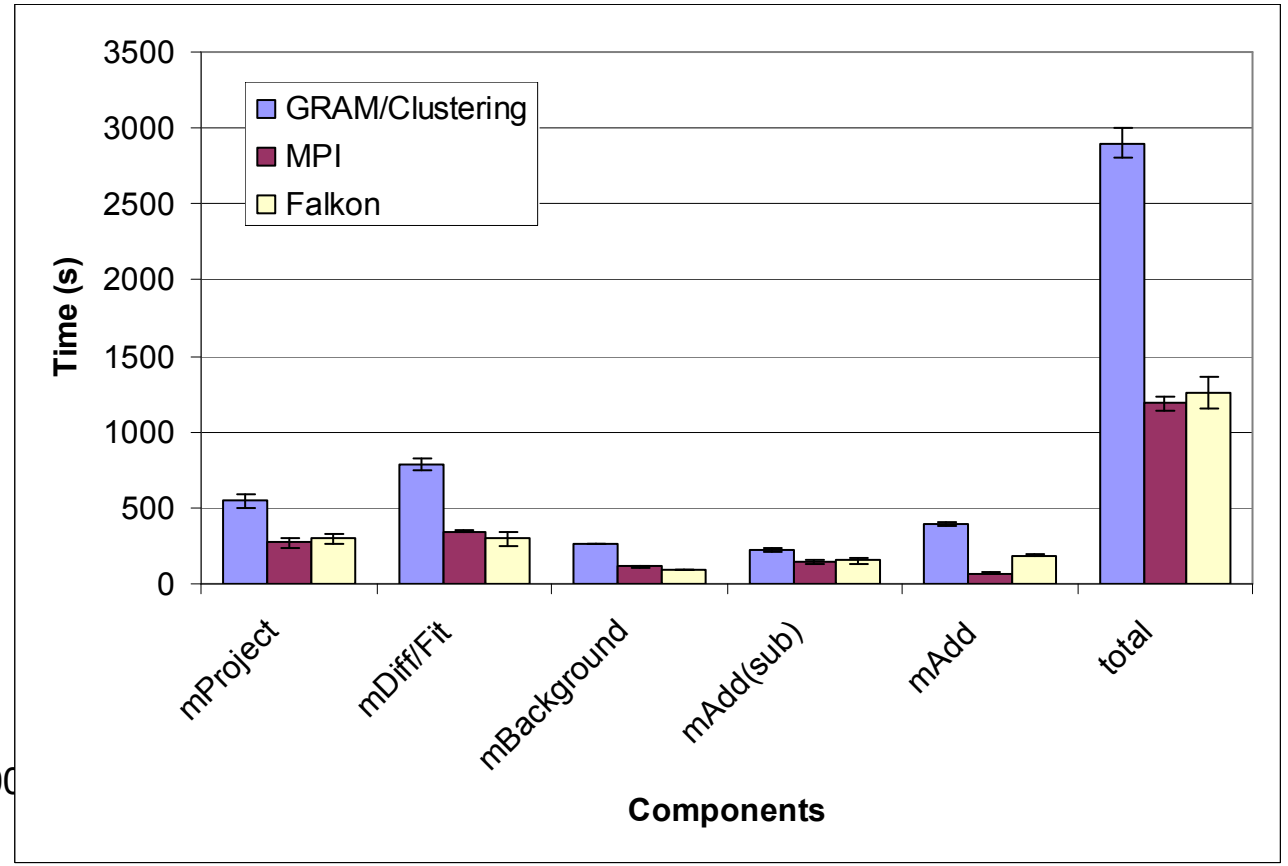
Applications: fMRI

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

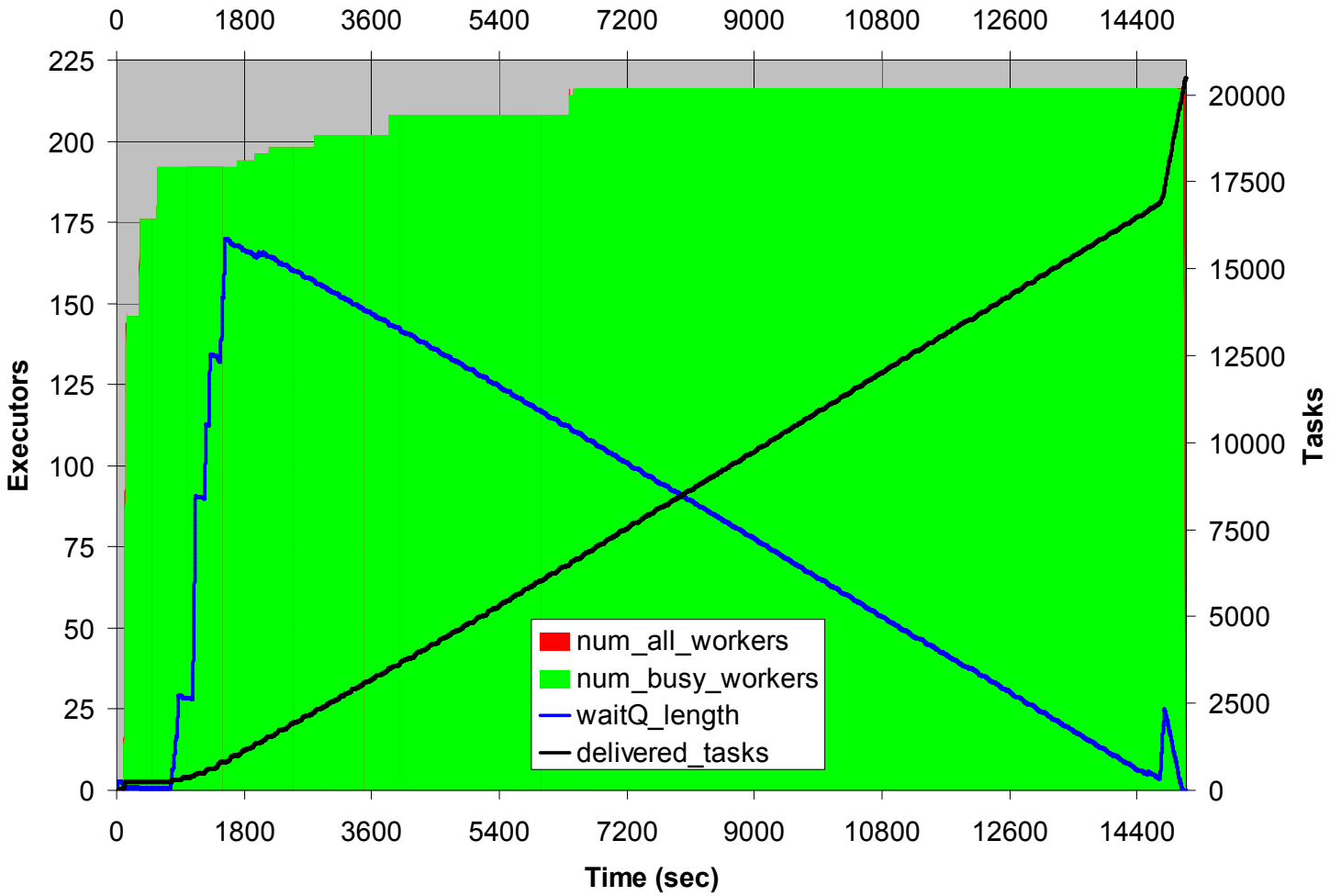


Applications: Montage

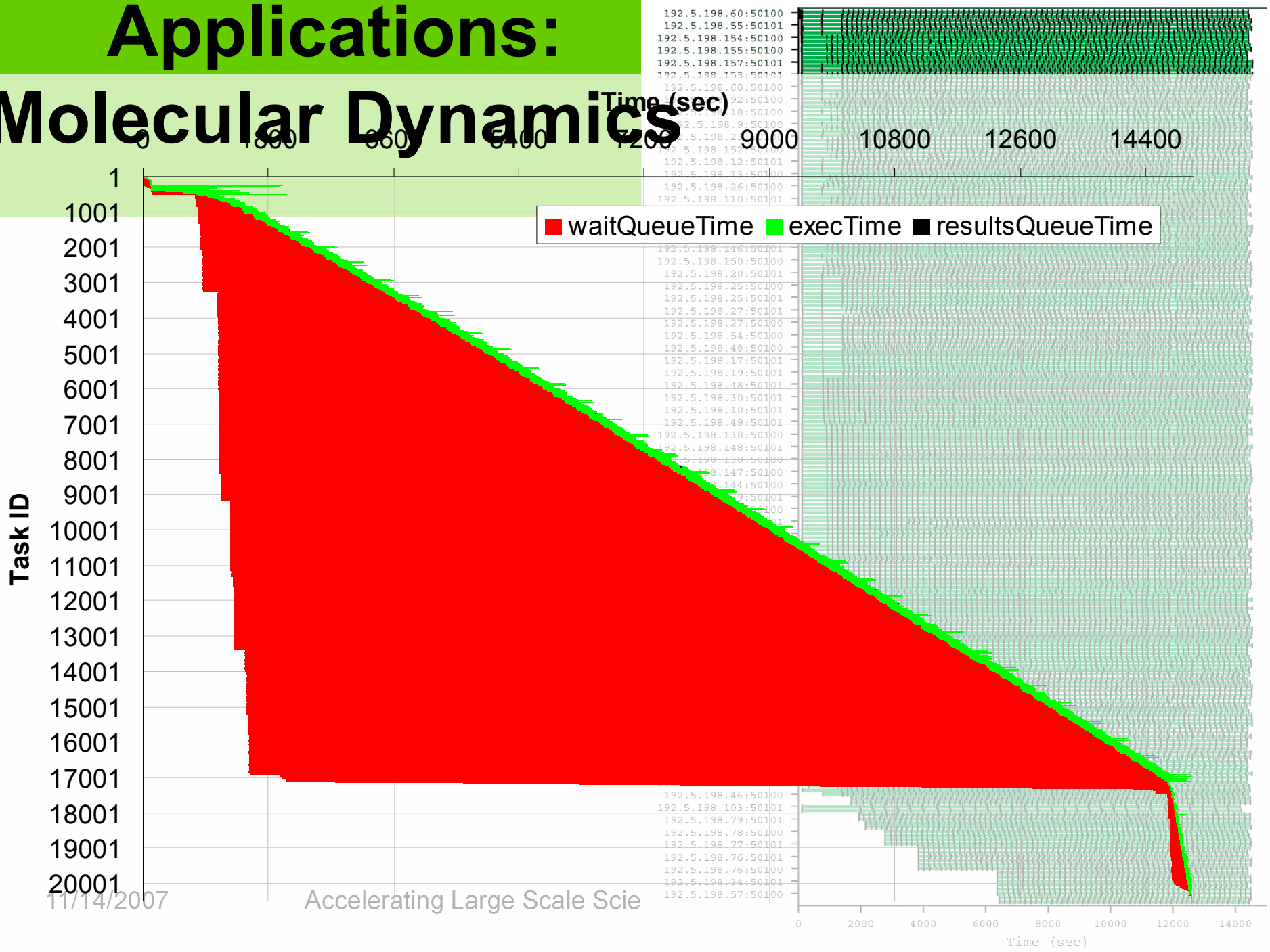
- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound



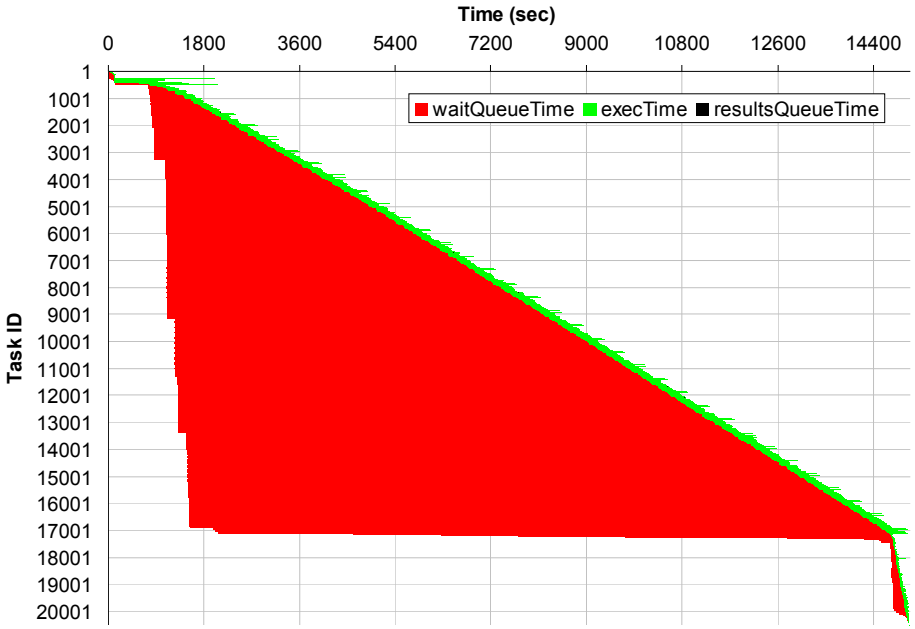
Applications: Molecular Dynamics

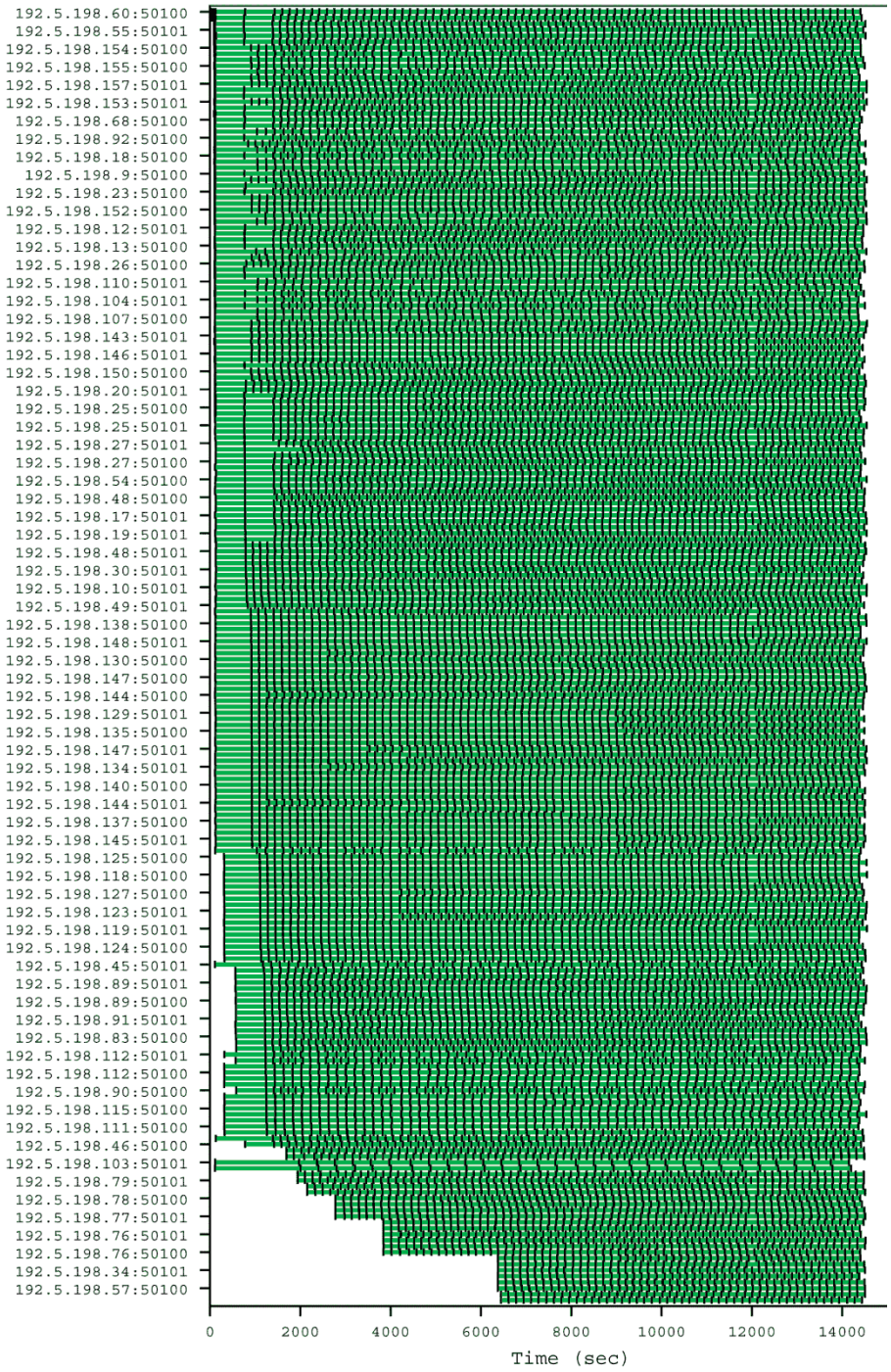


Applications:
Molecular Dynamics

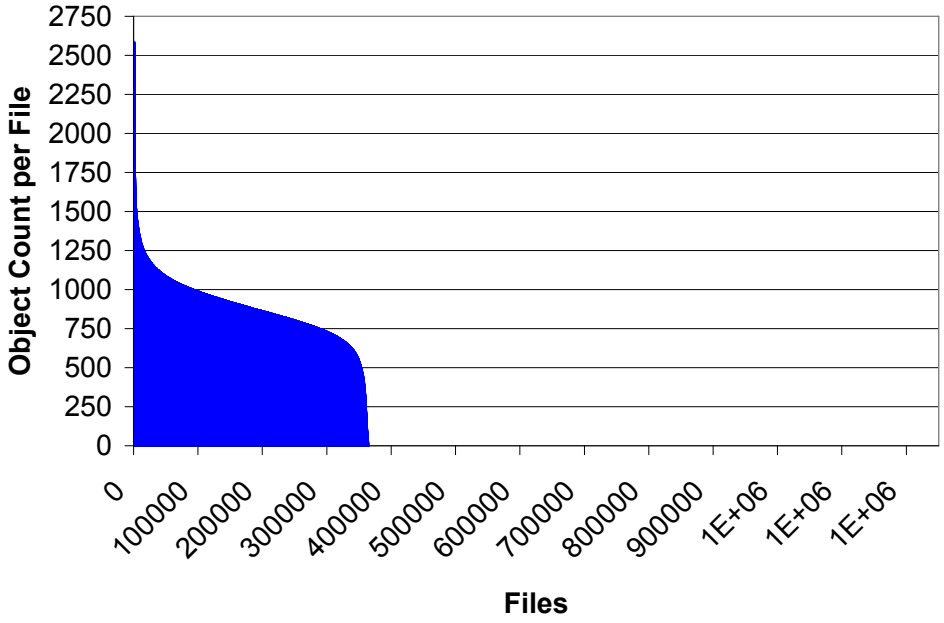


Applications: Molecular Dynamic



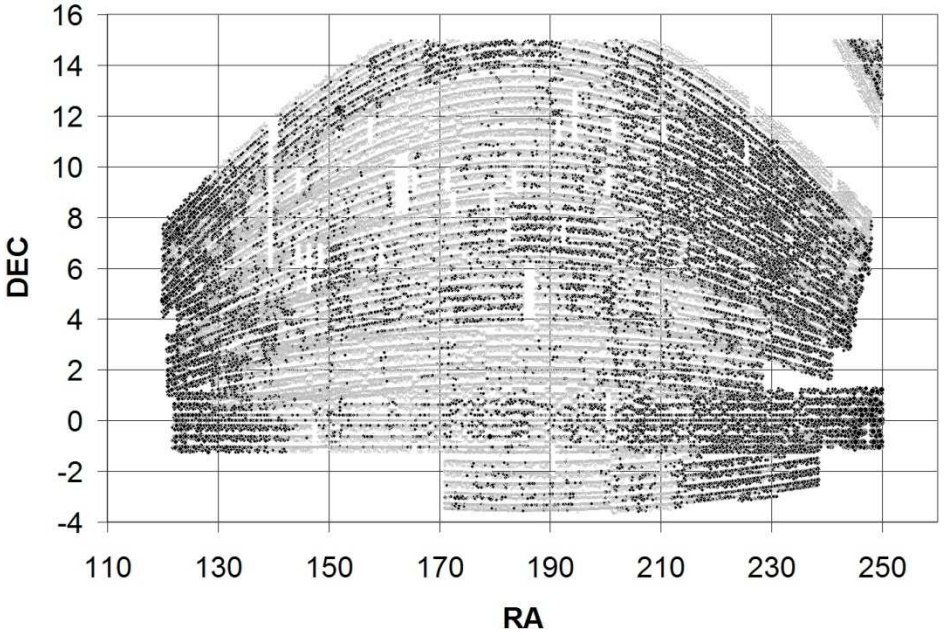


Applications: Stacking



- 320M+ objects
- 1.5M+ files
- 3.3 TB compressed / 9TB uncompressed

- SDSS DR5 Dataset

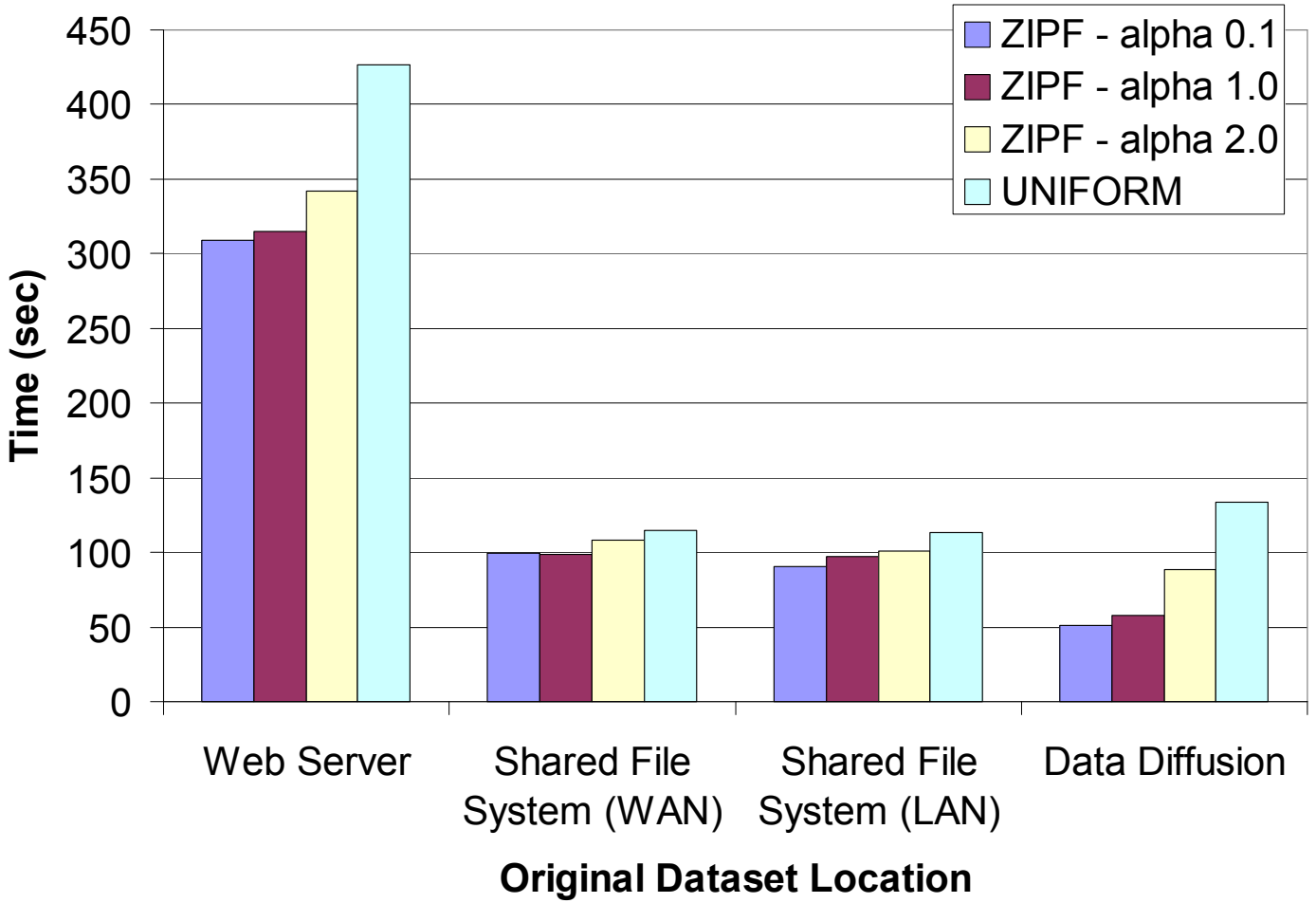


Stacking Workloads

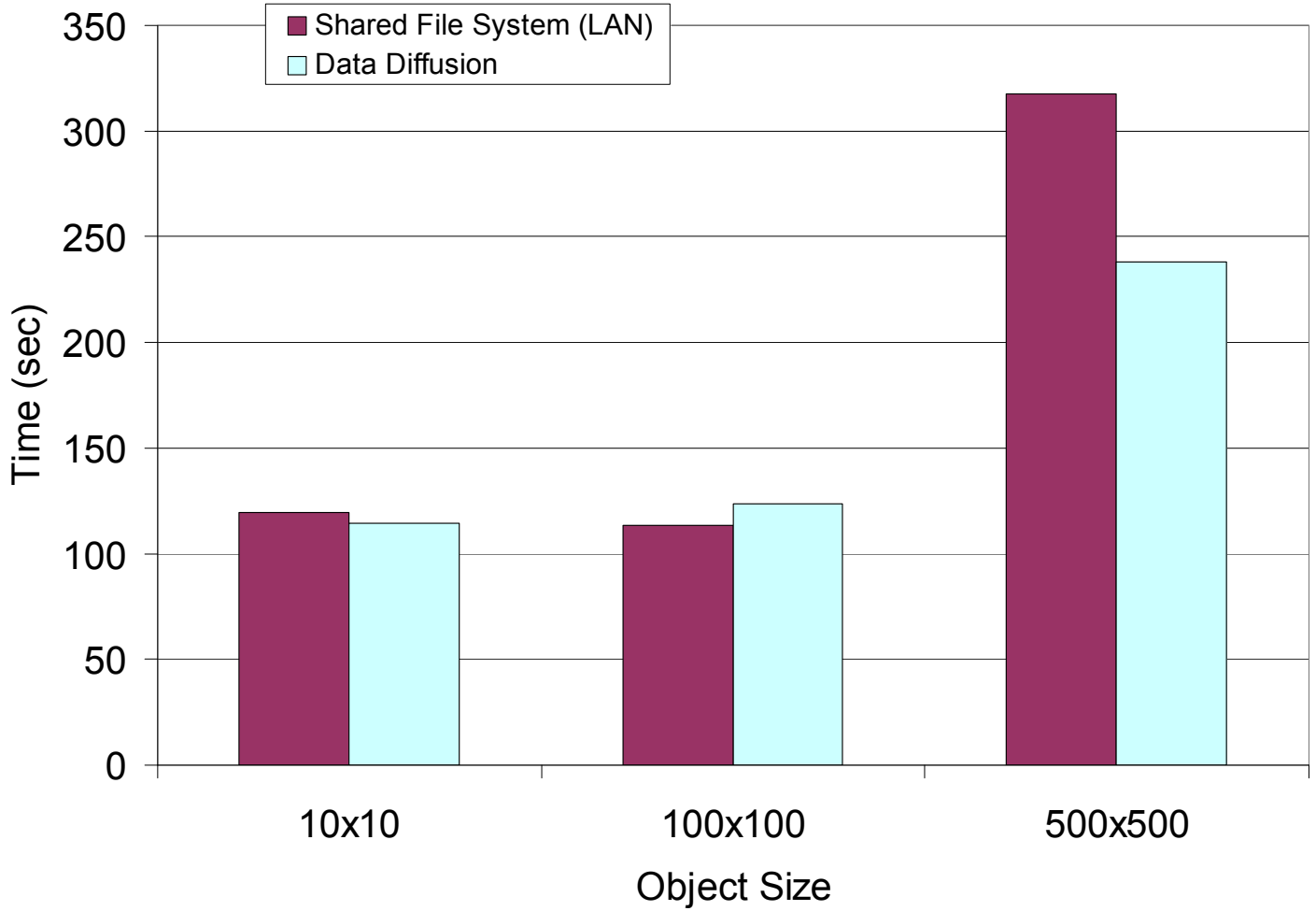
- Uniform sample from 320M objects
- ZIPF distribution of object popularity
  - alpha = 0.1, 1.0, 2.0

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

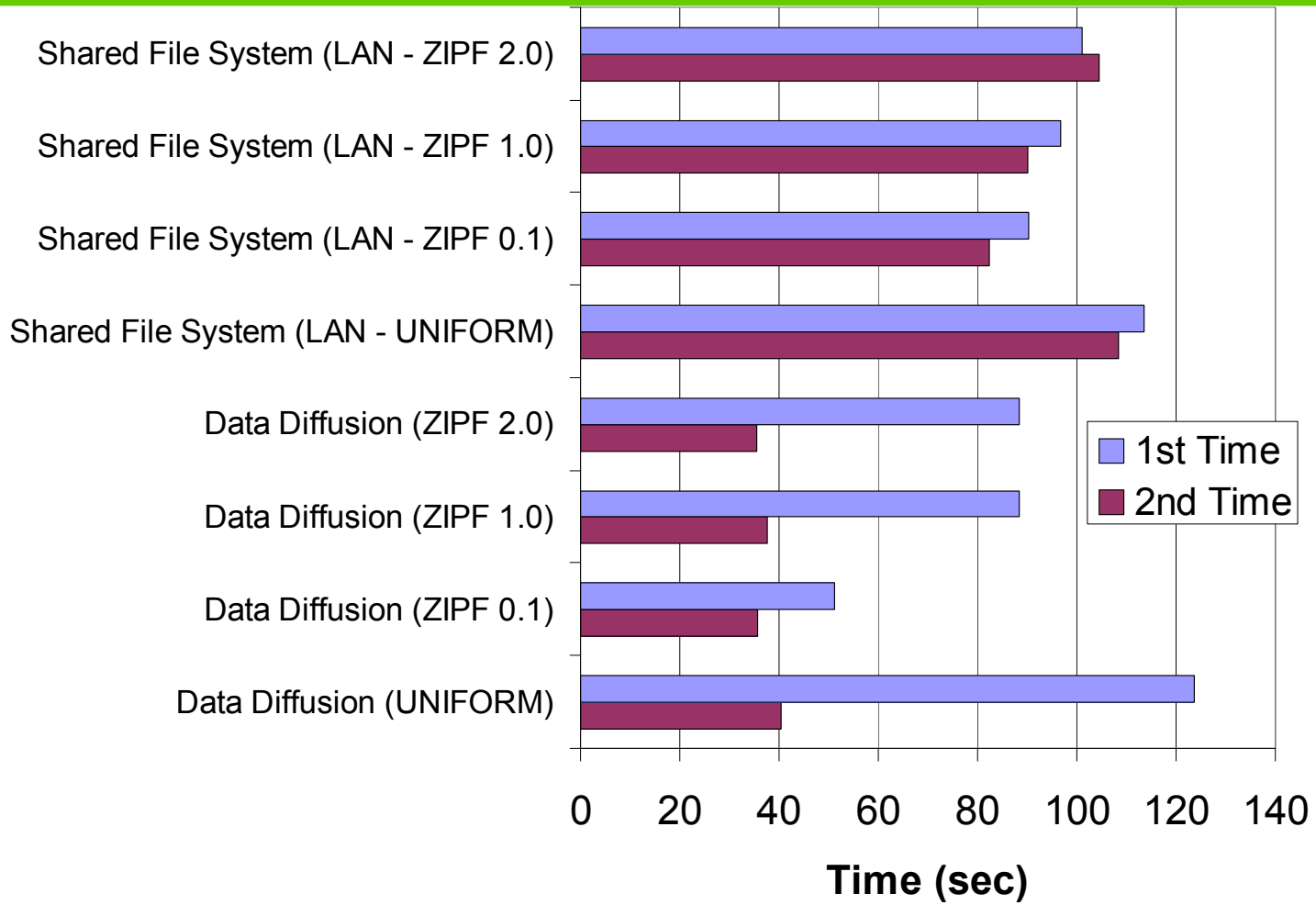
Stacking Performance Evaluation



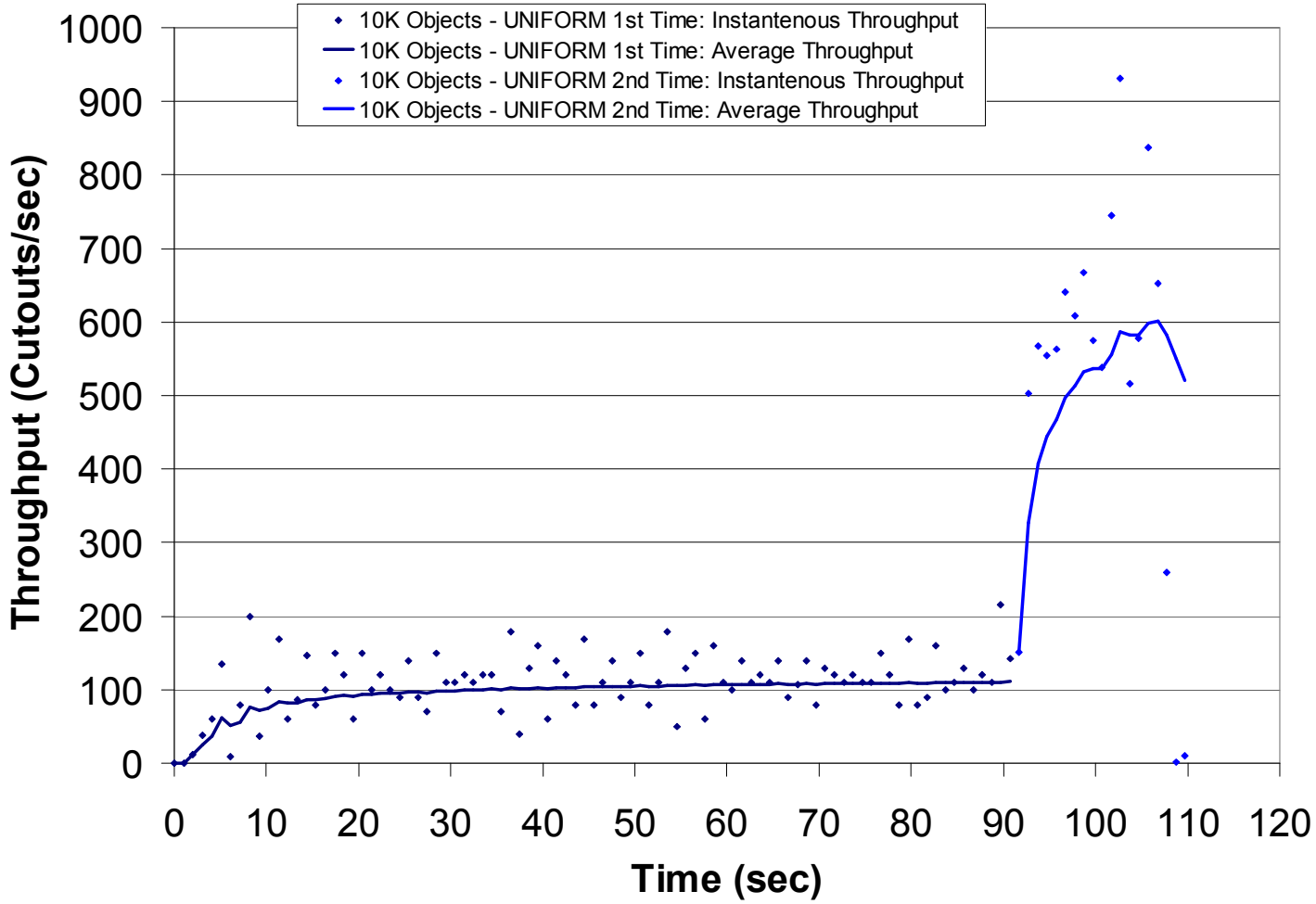
Object Size Effect



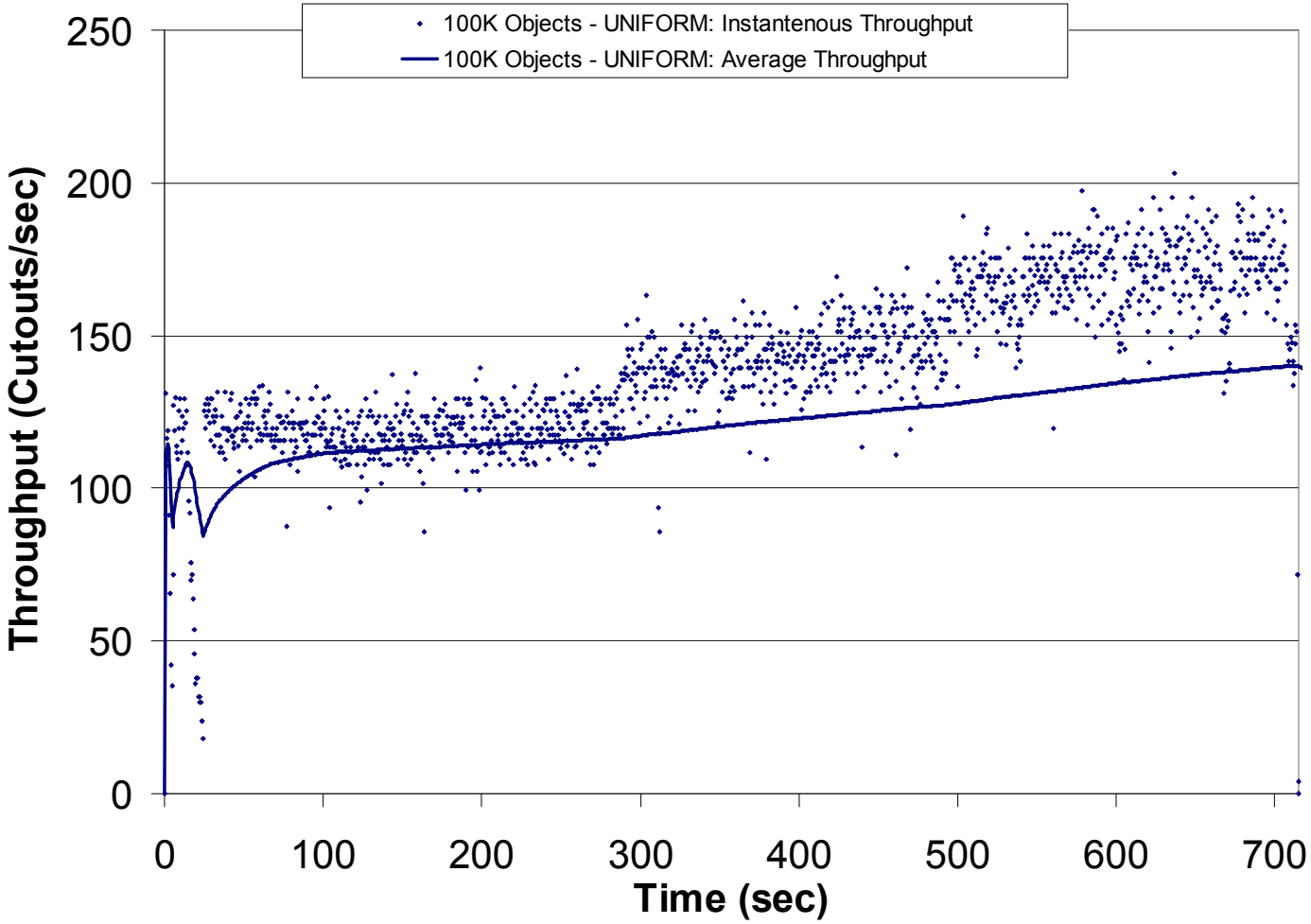
Data Diffusion Effect



10K*2 Stacking Workload



100K Stacking Workload



Future Work

- Explore data pre-fetching policies
- Update Swift to use data diffusion
- 3-Tier Architecture (essential on BG/P)
- Extend provisioner to support the Virtual Workspace Service (opens door to EC2)
- Globus Incubator Project
- Alternate languages and technologies

More Information

- Web: http://dev.globus.org/wiki/Incubator/Falkon
- Related Projects:
  - Swift: http://www.ci.uchicago.edu/swift/index.php
  - AstroPortal: http://people.cs.uchicago.edu/~iraicu/research/AstroPortal/index.htm
- Falkon Collaborators:
  - Ioan Raicu, The University of Chicago
  - Yong Zhao, Microsoft
  - Ian Foster, The University of Chicago & Argonne National Laboratory
  - Alex Szalay, The Johns Hopkins University
  - Catalin Dumitrescu, Computer Science Dept., The University of Chicago
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
  - Zhao Zhang, 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

More Information: Related Documents

- Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. “Falkon: a Fast and Light-weight tasK executiON framework”, to appear at IEEE/ACM SuperComputing 2007.
- Ioan Raicu, Catalin Dumitrescu, Ian Foster. Dynamic Resource Provisioning in Grid Environments, to appear TeraGrid Conference 2007.
- Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, Ioan Raicu, Tiberiu Stef-Praun, Mike Wilde. “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”, to appear at IEEE Workshop on Scientific Workflows 2007.
- Yong Zhao, Mihael Hategan, Ioan Raicu, Mike Wilde, Ian Foster. “Swift: a Parallel Programming Tool for Large Scale Scientific Computations”, under review at Scientific Programming Journal, Special Issue on Dynamic Computational Workflows: Discovery, Optimization, and Scheduling.
- Ioan Raicu, Ian Foster, Alex Szalay. “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets”, poster presentation, IEEE/ACM SuperComputing 2006.
- Ioan Raicu, Ian Foster, Alex Szalay, Gabriela Turcu. “AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis”, TeraGrid Conference 2006, June 2006.
- Alex Szalay, Julian Bunn, Jim Gray, Ian Foster, Ioan Raicu. “The Importance of Data Locality in Distributed Computing Applications”, NSF Workflow Workshop 2006.