



THE UNIVERSITY OF
CHICAGO



Falkon: a Fast and Light-weight task executiON framework

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 15th, 2007

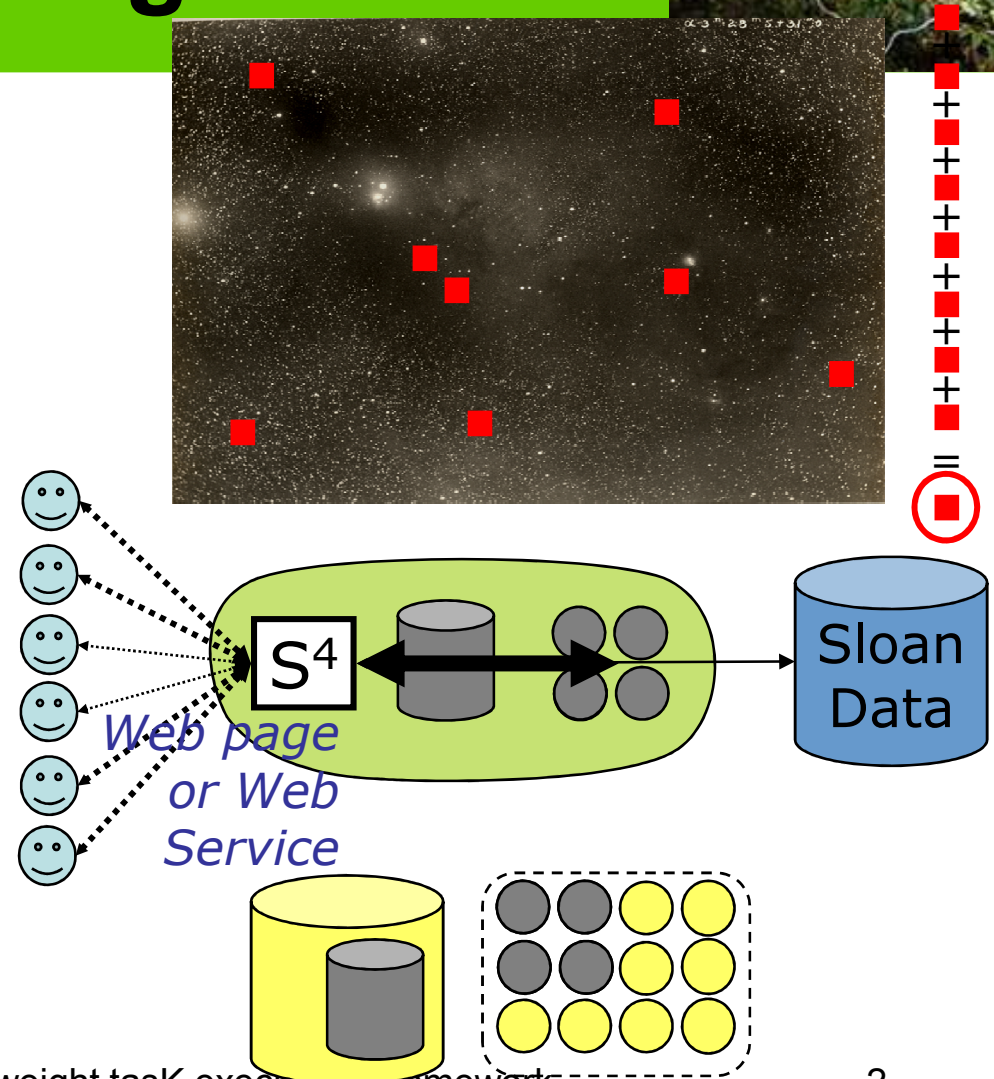


Argonne
NATIONAL LABORATORY

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

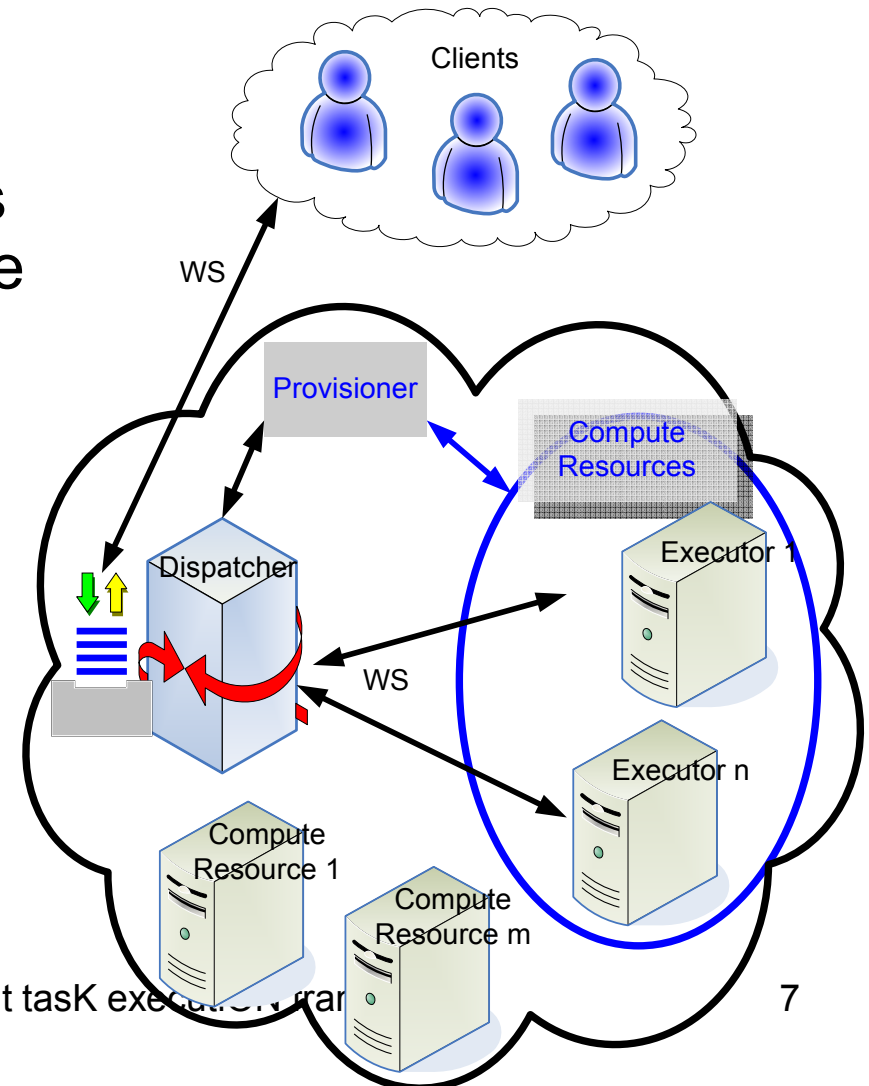


- a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
- ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
- performs ***data diffusion*** and uses a data-aware scheduler to leverage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



Falkon Message Exchanges



- Description:

- {1}: task(s) submit

- {2}: task(s) submit confirmation

- {3}: notification for work

- {4}: request for task(s)

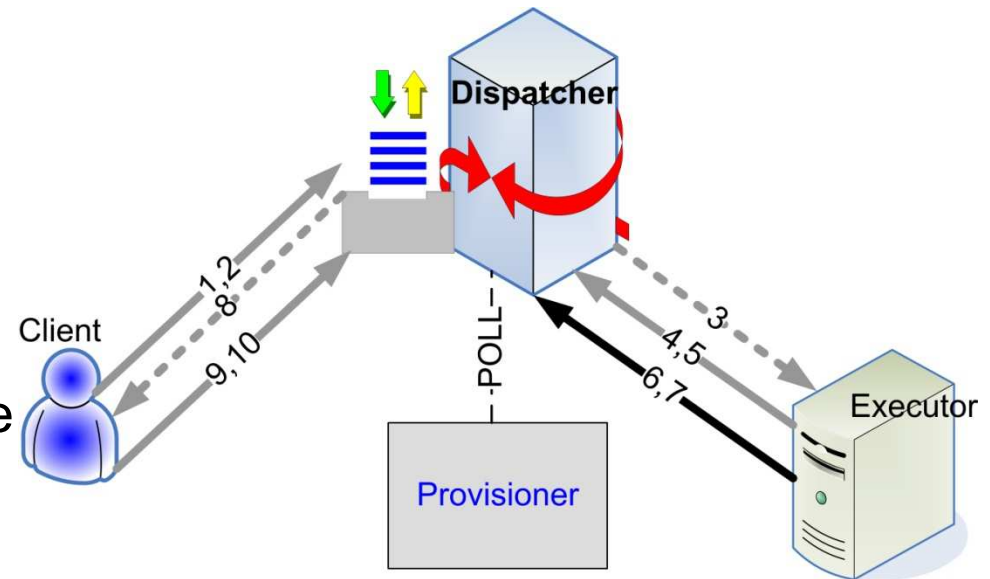
- {5 or 7}: dispatch task(s)

- {6}: deliver task(s) results to service

- {8}: notification for task result(s)

- {9}: request for task result(s)

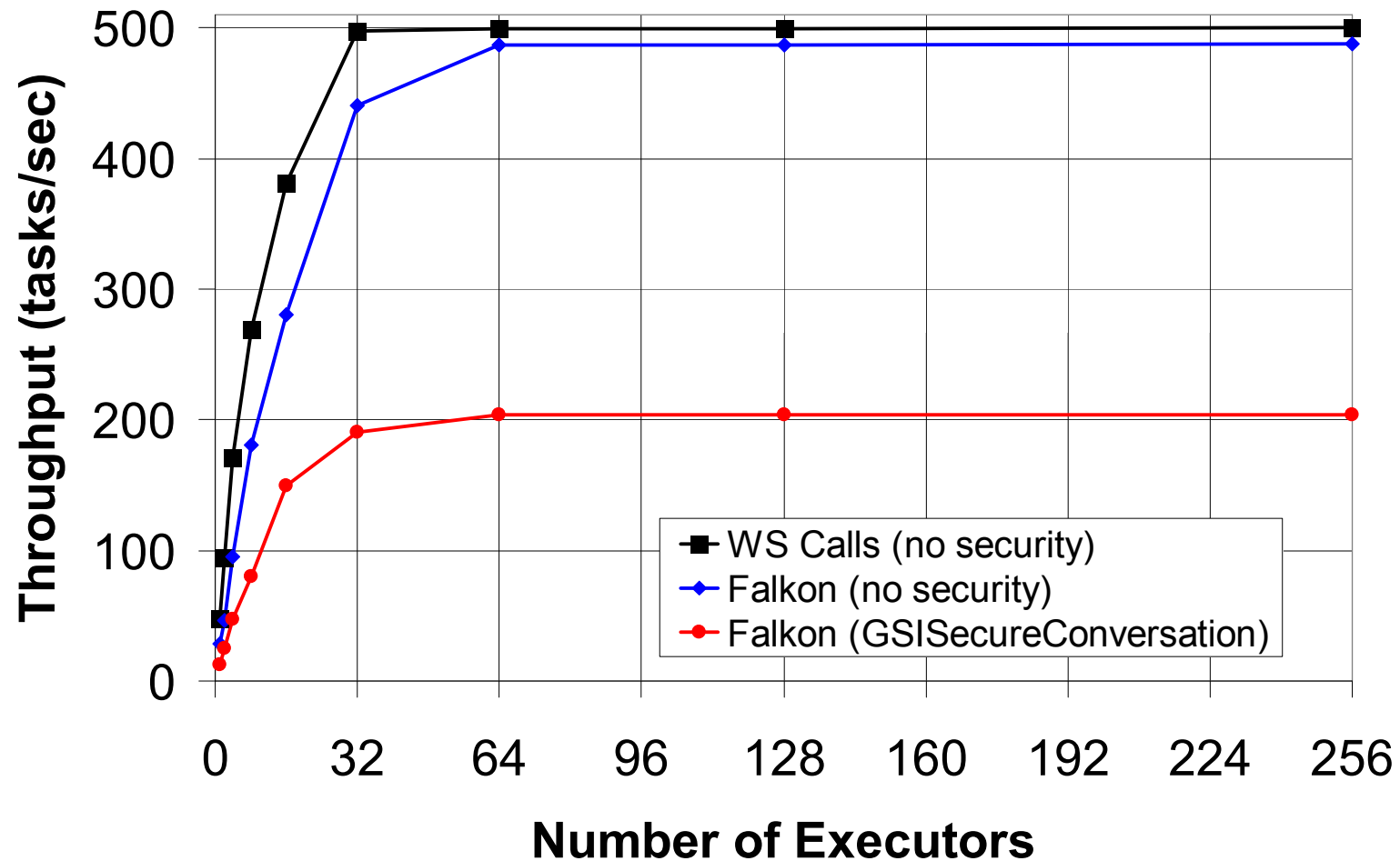
- {10}: deliver task(s) results to client



- 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

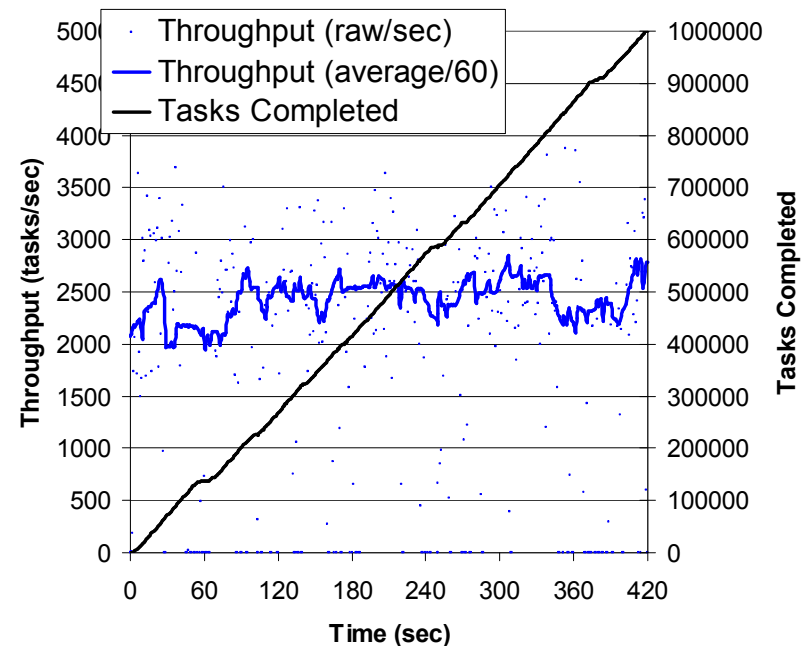
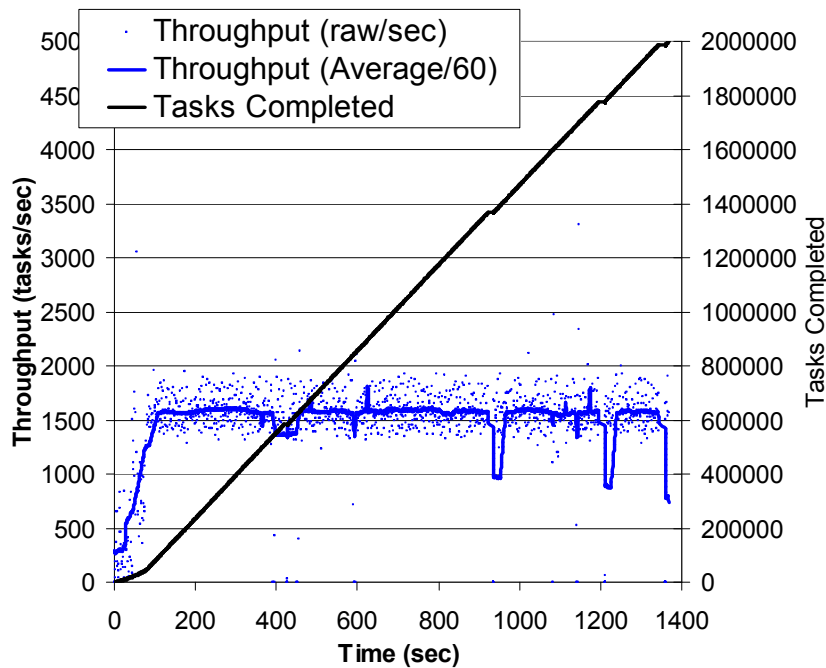
Dispatch Throughput



Dispatch Throughput: SiCortex Machine



Dispatch Throughput: Bundling Effect

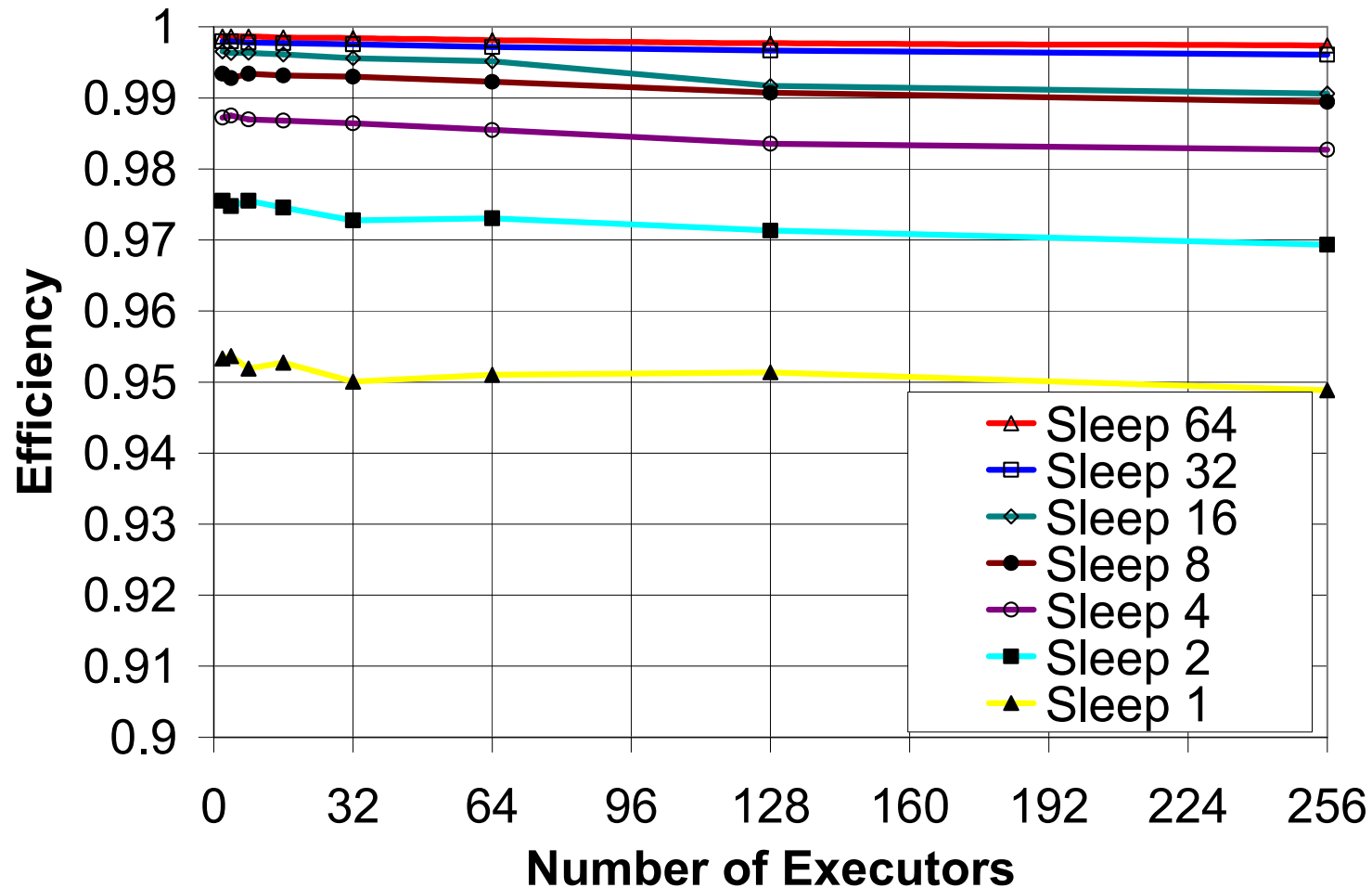


How does Falkon Compare?

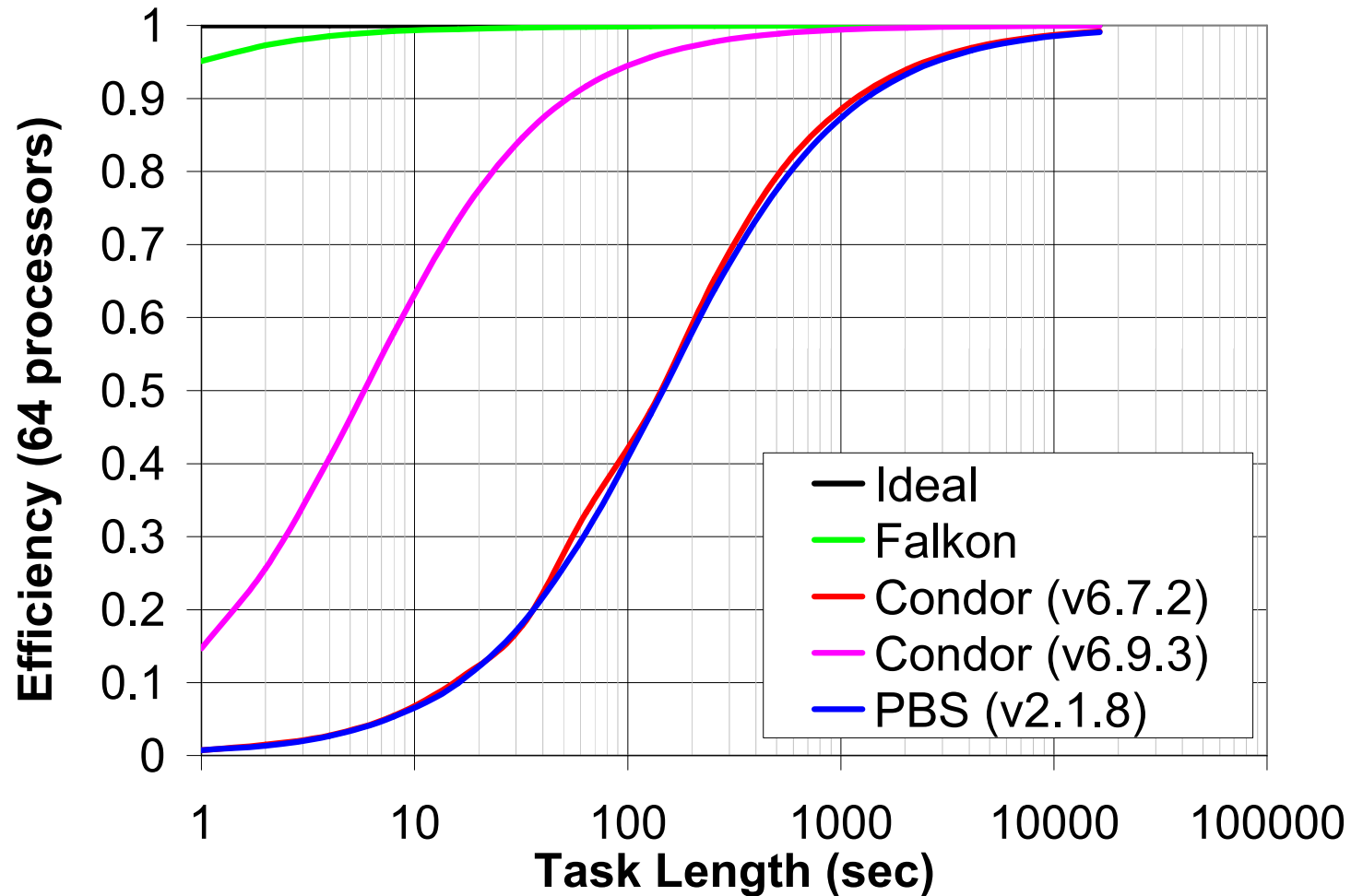


System	Comments	Throughput (tasks/sec)
Falkon (no security)	Dual Xeon 3GHz w/ HT 2GB	487
Falkon (GSISecureConversation)	Dual Xeon 3GHz w/ HT 2GB	204
Condor (v6.7.2)	Dual Xeon 2.4GHz, 4GB	0.49
PBS (v2.1.8)	Dual Xeon 2.4GHz, 4GB	0.45
Condor (v6.7.2) [15]	Quad Xeon 3 GHz, 4GB	2
Condor (v6.8.2) [34]		0.42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

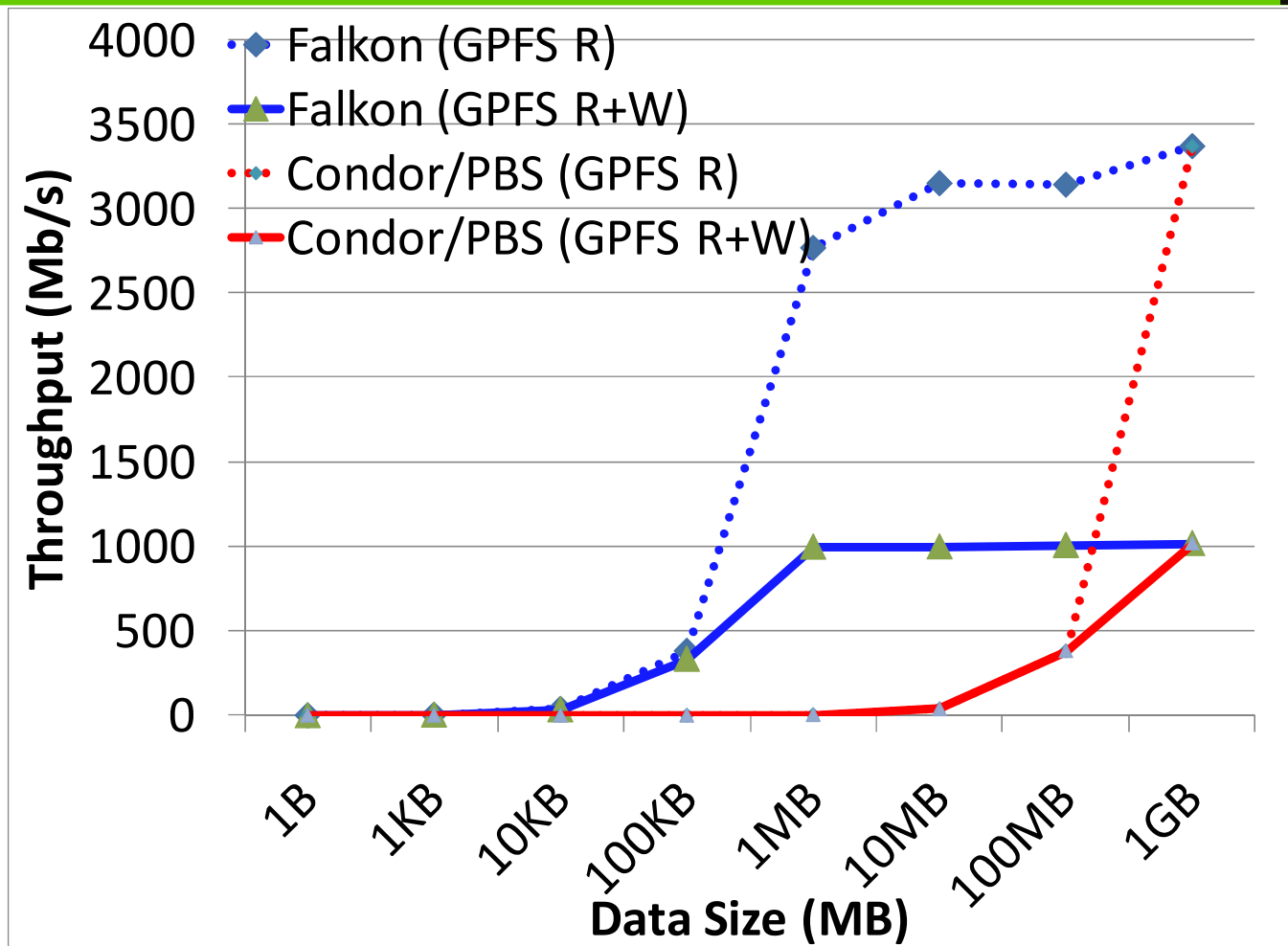
Execution Efficiency



Efficiency Comparison



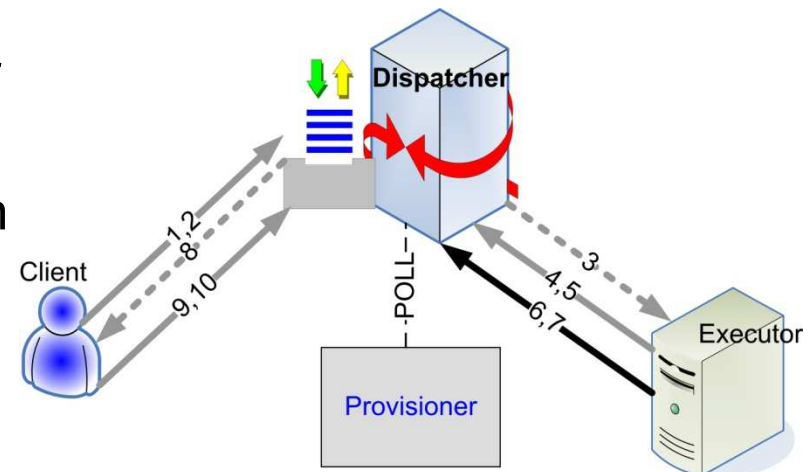
How does Dispatch Rates Affect Throughput I/O?



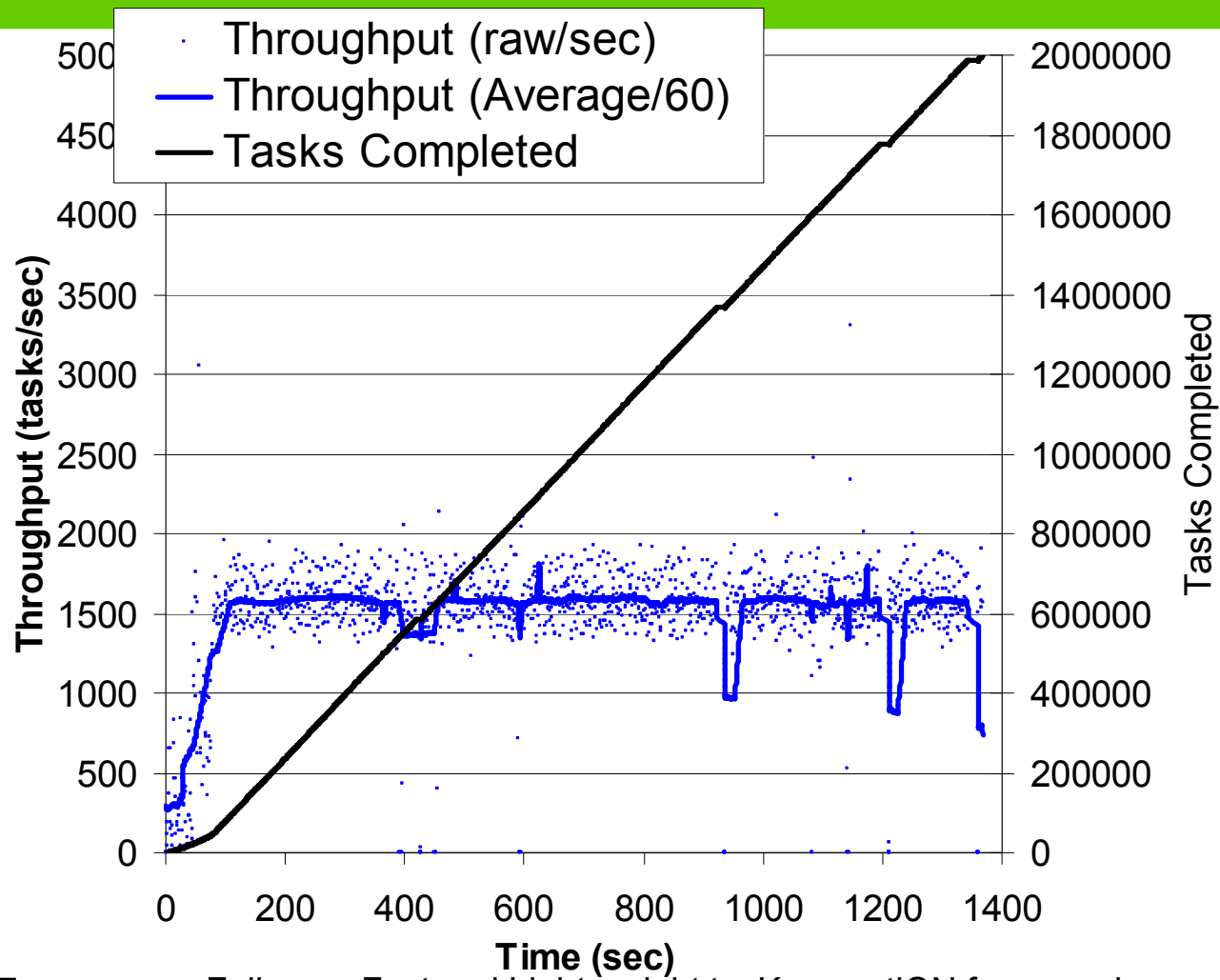
Enhancements



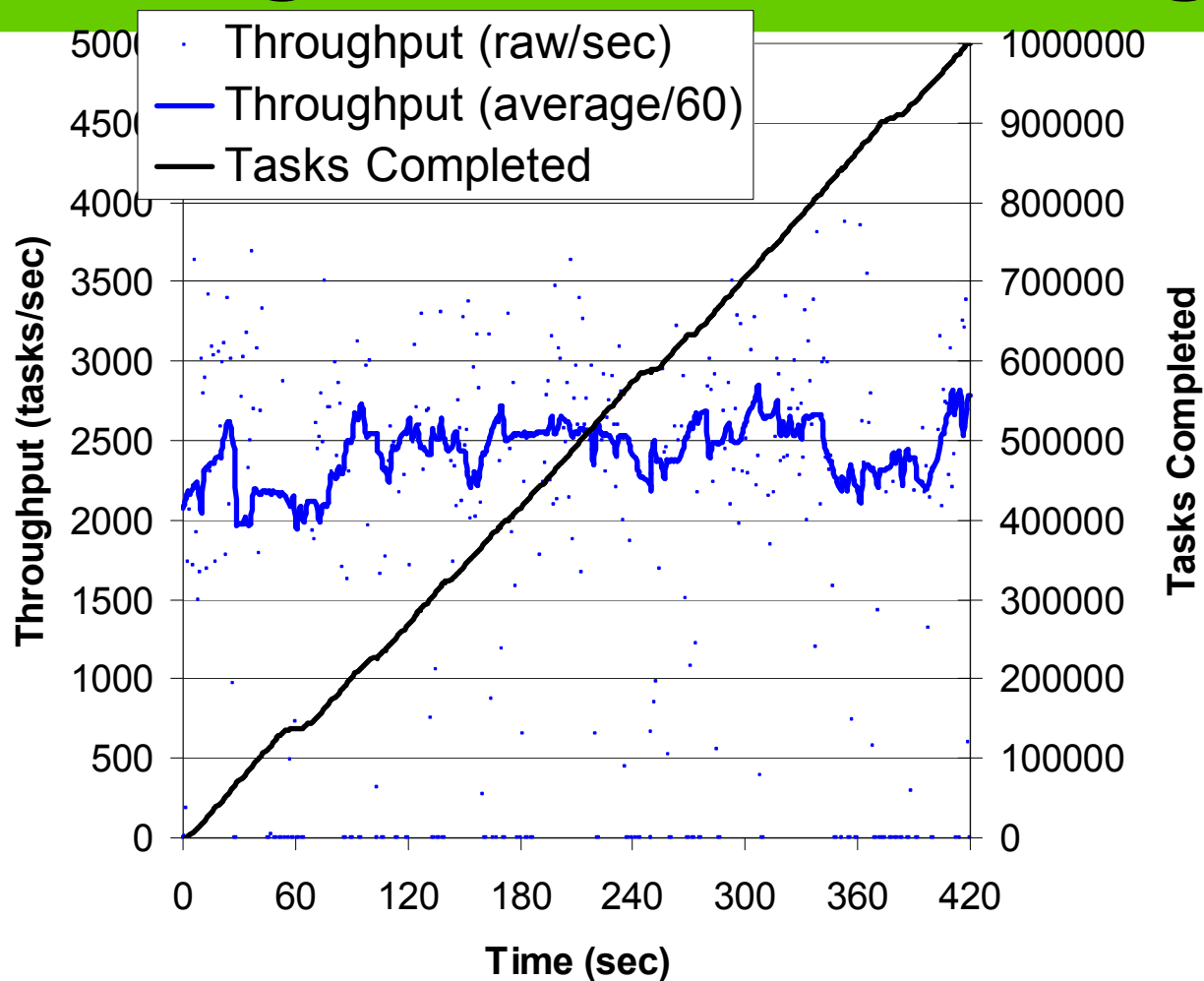
- Bundling
 - Include multiple tasks per communication message
- Piggy-Backing
 - Attach next task to acknowledgement of previous task
 - Include data management information in the task description and acknowledgement messages
- Message reduction:
 - General Lower Bound: $10 \rightarrow 2+c$, where c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



Falkon Scalability: Bundling and Throttling



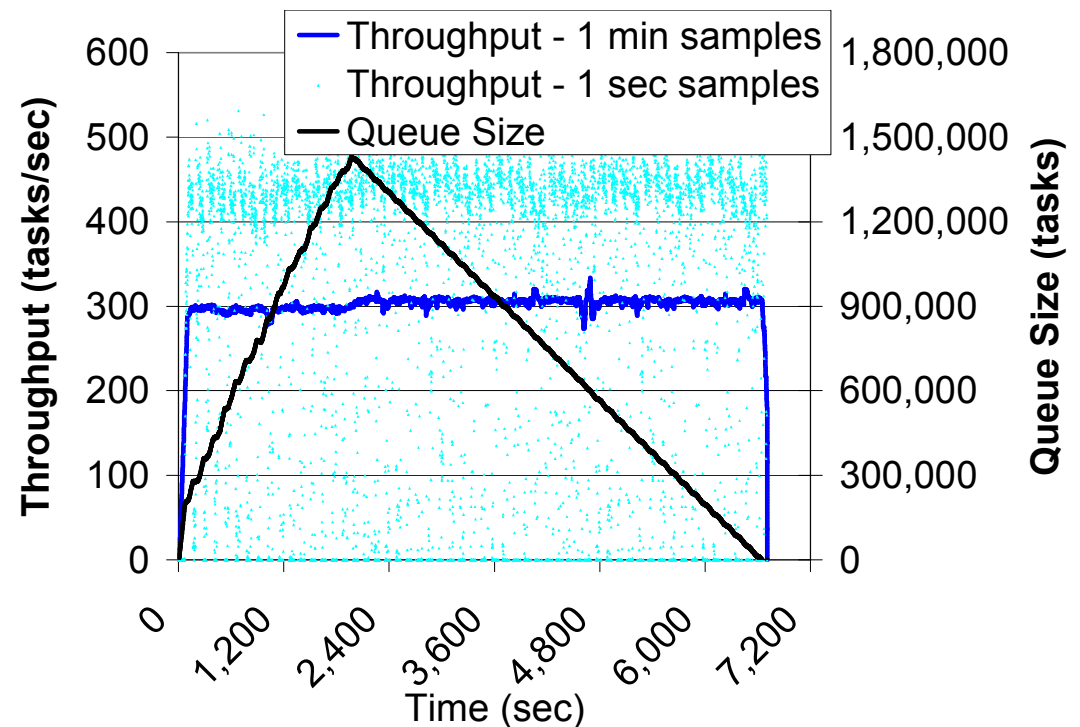
Falkon Scalability: Bundling and No Throttling



Falkon Scalability



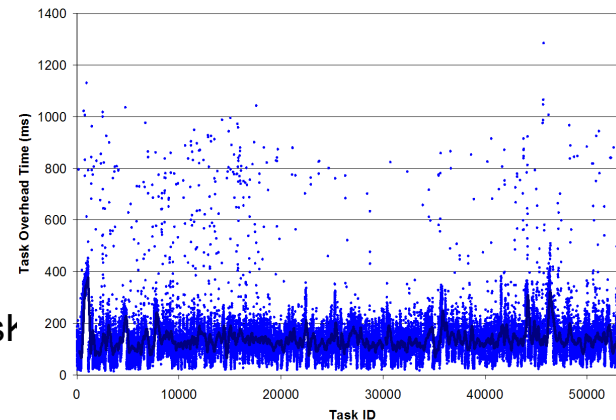
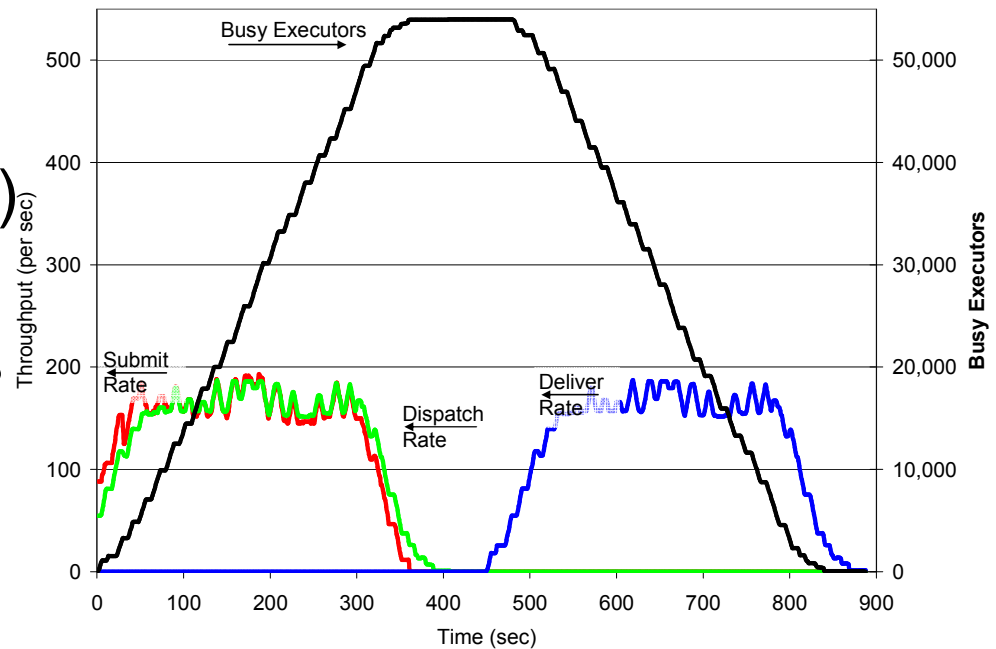
- 2M tasks processed (sleep 0) on 16 machines (32 executors)
- 6711 seconds ~ 298 tasks/sec
- Queue Length ~ 1.5M
- Sun JDK 1.4.2 with 1.5GB Heap
- Throughput variations → issues with GC



Falkon Scalability



- 54K tasks processed (sleep 480)
- 60 machines (54K executors)
- Per task overhead: 127 ms, standard deviation 76ms
- Sun JDK 1.4.2 with 1.5GB Heap



Goals

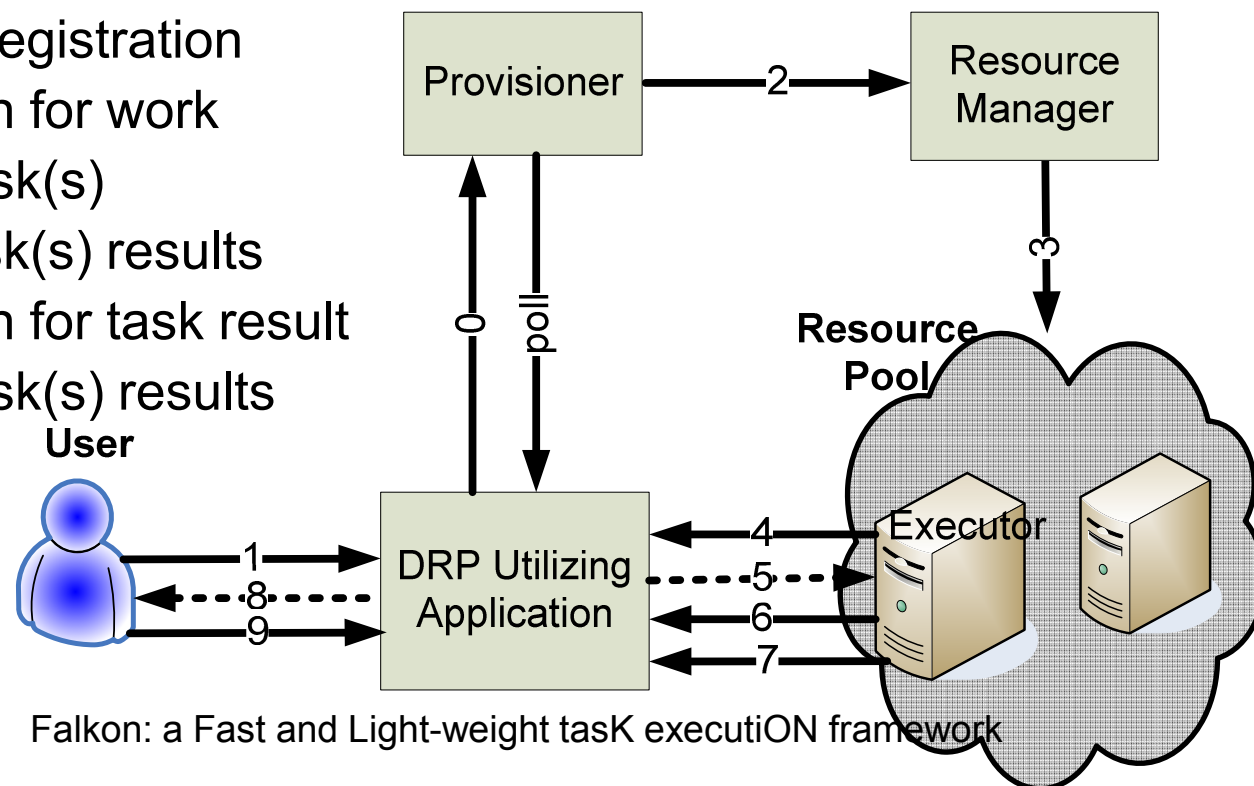


- a *streamlined task dispatcher* able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
- ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
- performs *data diffusion* and uses a data-aware scheduler to leverage the co-located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



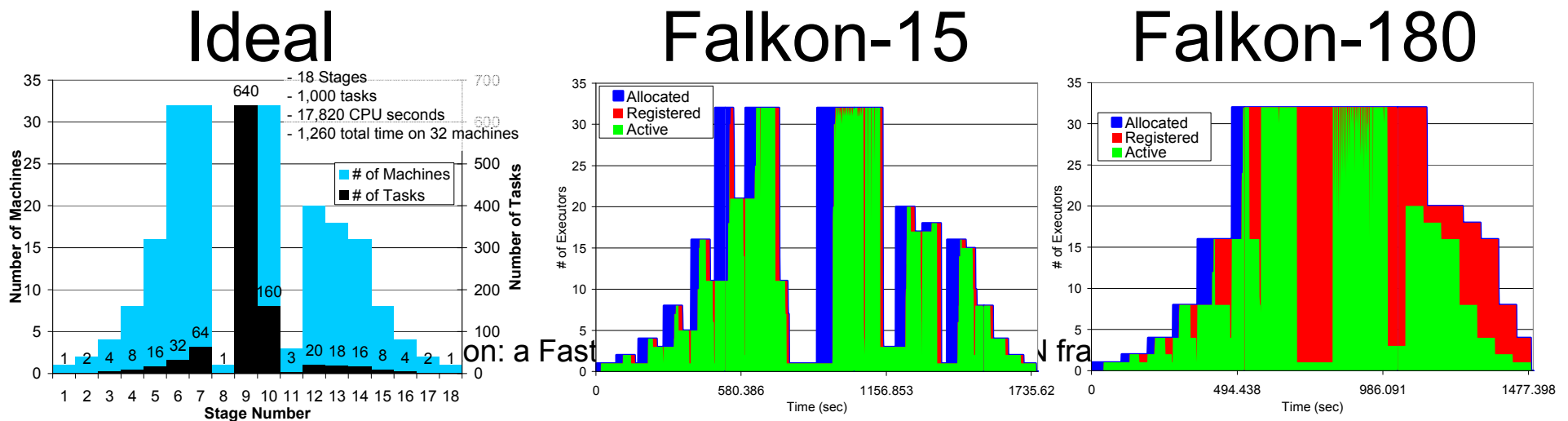
- Synthetic Workload
 - 18 stages
 - 1000 tasks
 - 17,820 CPU seconds
 - 1,260 seconds total time on 32 machines

Stage #	# of Tasks	Task Time (sec)	Stage Time (32 machines)	# Machines (32 max)
1	1	60	60	1
2	2	60	60	2
3	4	60	60	4
4	8	60	60	8
5	16	60	60	16
6	32	60	60	32
7	64	60	120	32
8	1	120	120	1
9	640	6	120	32
10	160	12	60	32
11	3	60	60	3
12	20	60	60	20
13	18	60	60	18
14	16	60	60	16
15	8	60	60	8
16	4	60	60	4
17	2	60	60	2
18	1	60	60	1

Dynamic Resource Provisioning



- Compared:
 - GRAM+PBS: 1 job per task
 - Falkon-15, -60, -120, -180: dynamic resource provisioning – 15, 60, 120, and 180 seconds idle time before resource de-allocation, and with 32 maximum machines
 - Falkon-∞: static resource provisioning with 32 maximum machines
 - Ideal with 32 maximum machines
- Falkon-15 / Falkon-180:
 - Allocated (blue): LRM wait queue
 - Registered (red): idle executors
 - Active (green): executors processing tasks



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
Resource 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

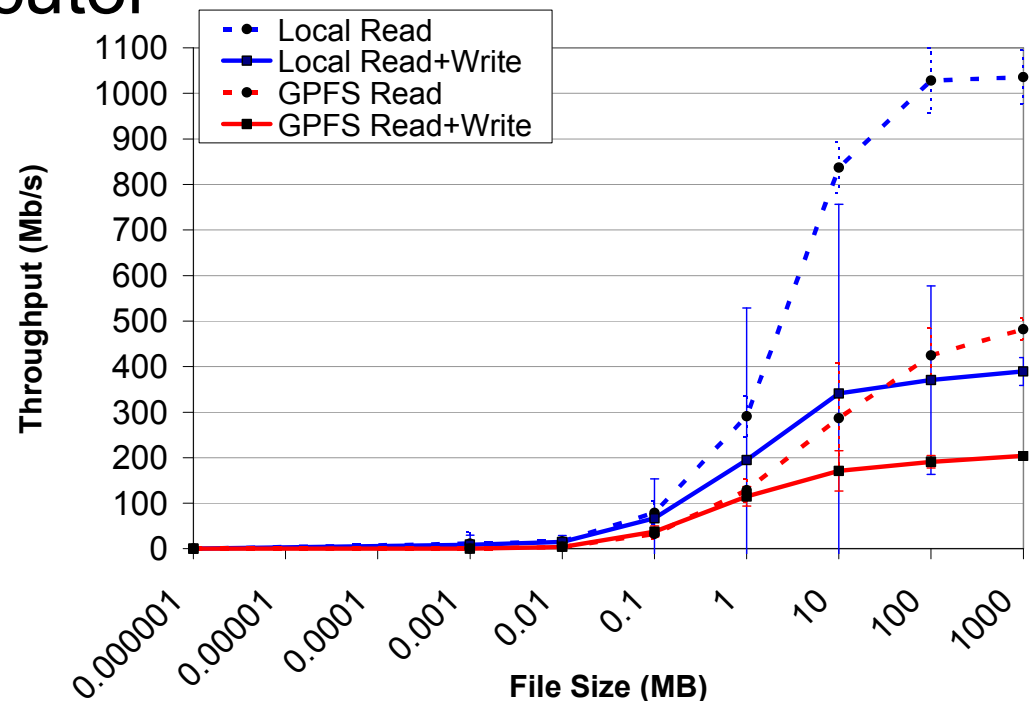


- a *streamlined task dispatcher* able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers
- *multi-level scheduling* techniques to enable separate treatments of resource provisioning
- performs ***data diffusion*** and uses a data-aware scheduler to leverage the co-located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

Data Diffusion



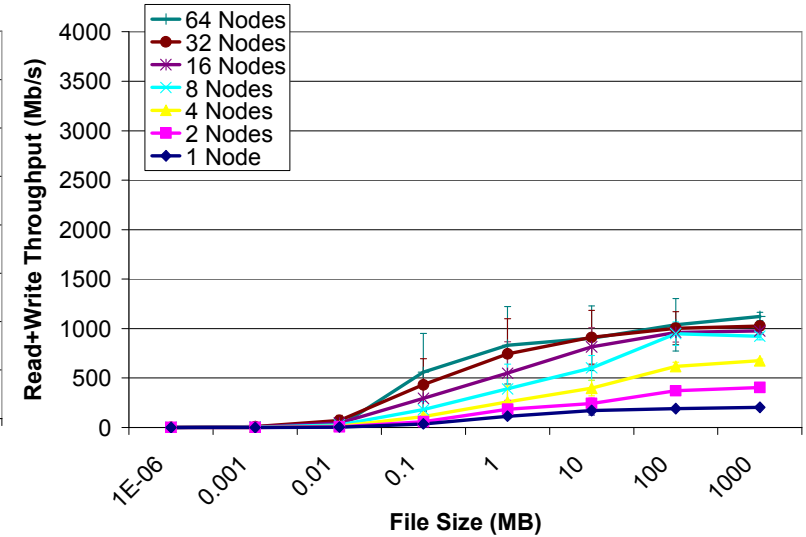
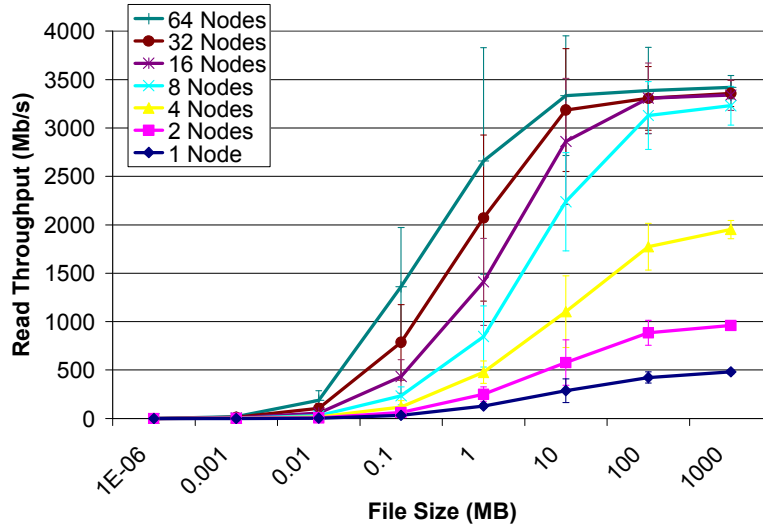
- Data caching at executor local disk
 - RANDOM
 - FIFO
 - LRU
 - LFU
- Avoid the need for a shared file system
- Theoretical linear scalability with compute resources



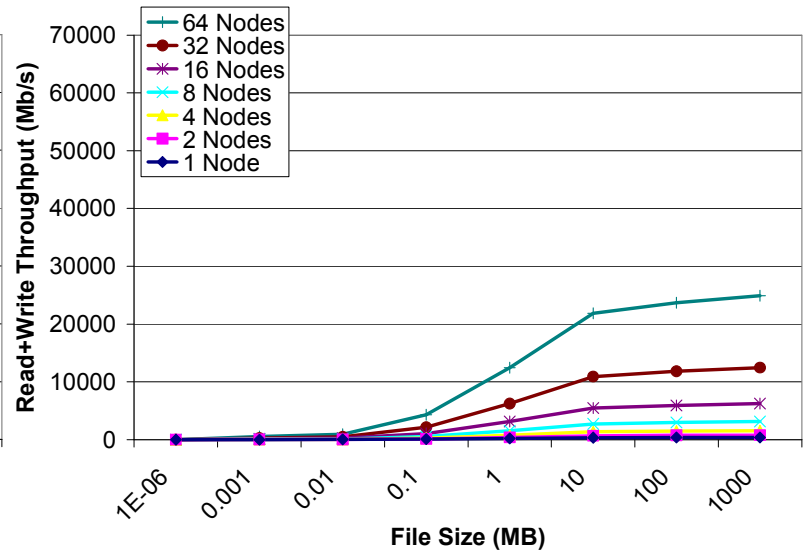
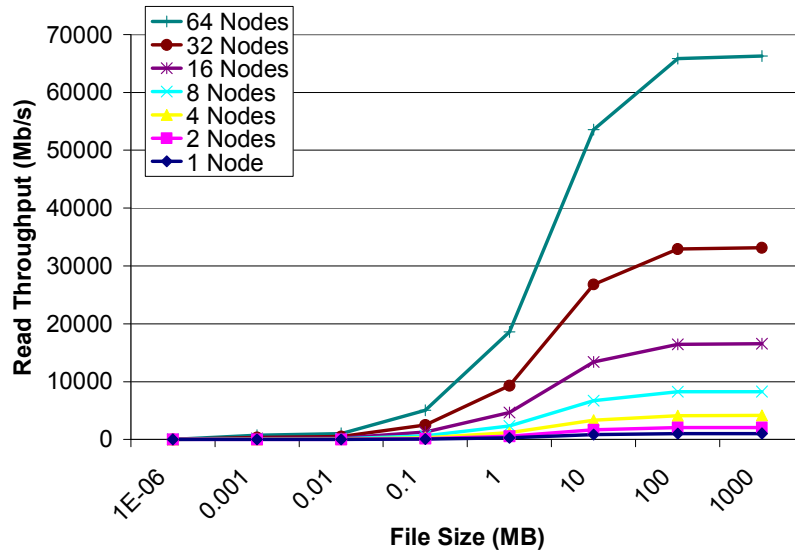
Shared vs. Local File System Performance



Shared File System



Local Disk

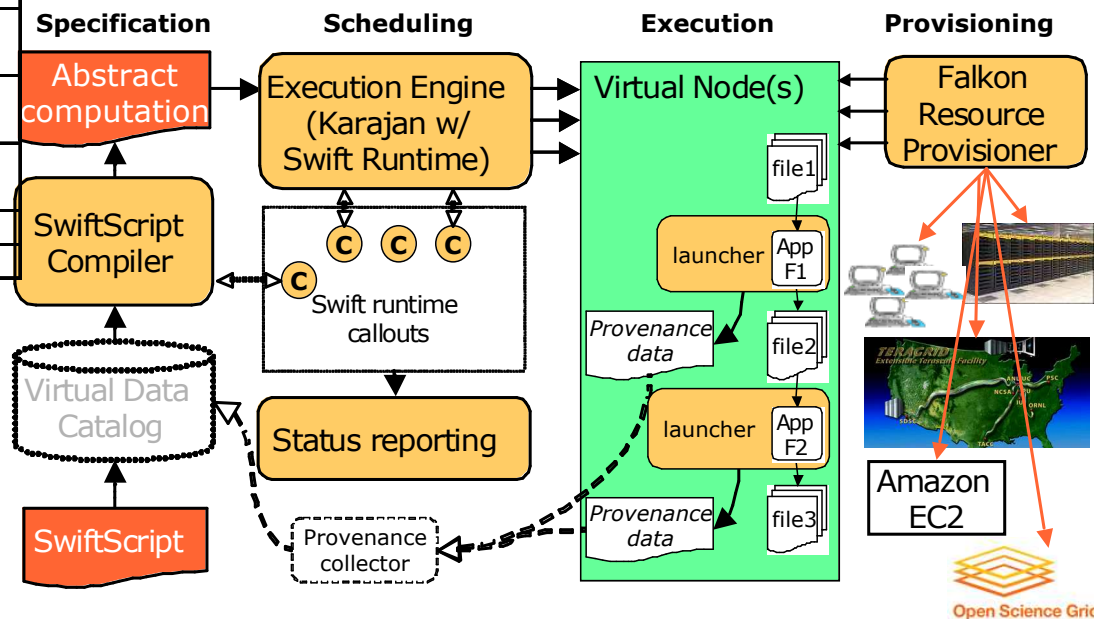


Applications via Swift



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



11/14/2007

Falcon: a Fast and Light-weight task executiON framework

29

Applications

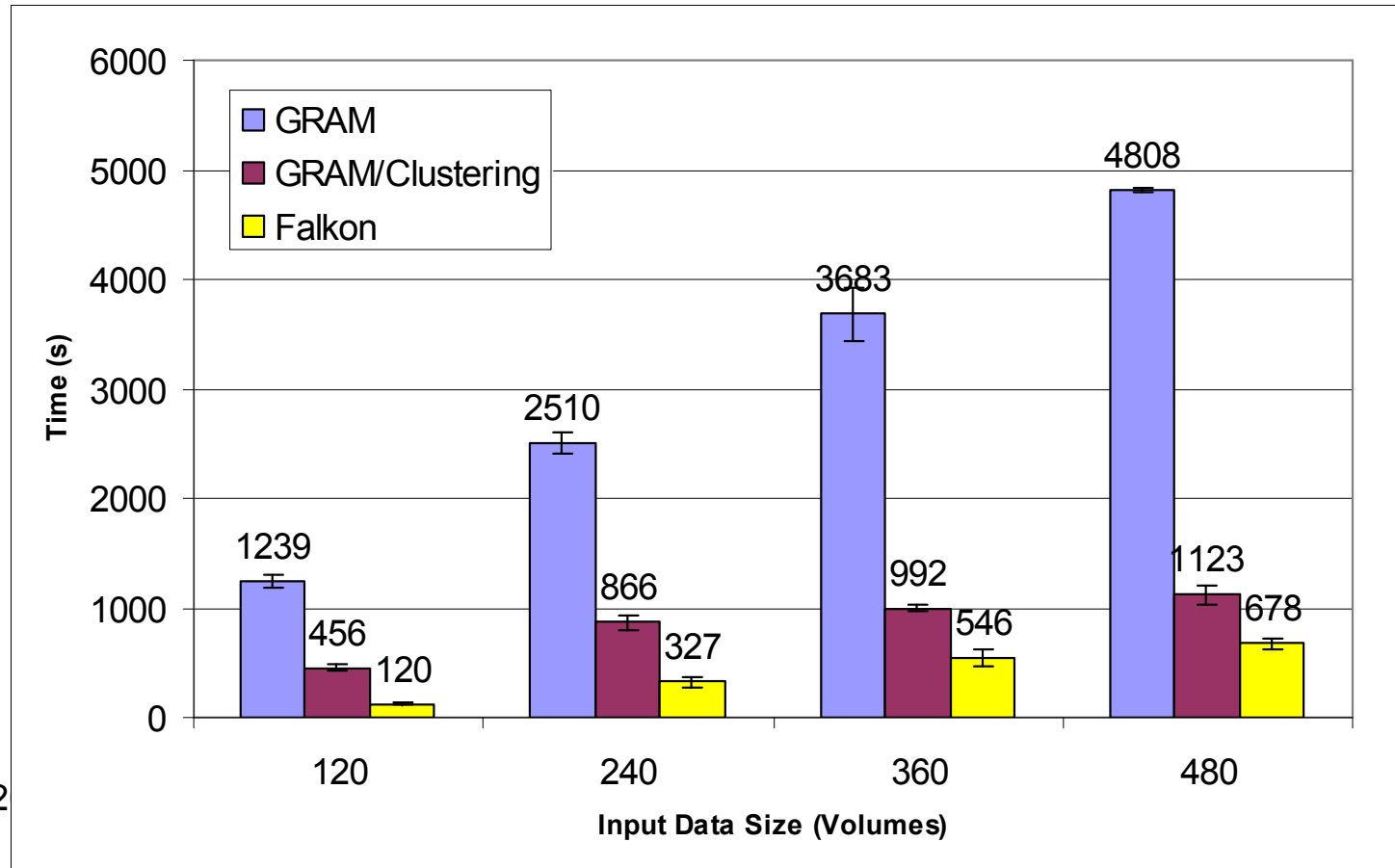


- Medicine:
 - fMRI (Falkon vs. GRAM)
- Astronomy:
 - Montage (Falkon vs. GRAM vs. MPI)
 - Stacking (Falkon with Data Diffusion vs. Falkon using shared file system)
- Chemistry
 - Molecular Dynamics Simulation

Applications: fMRI



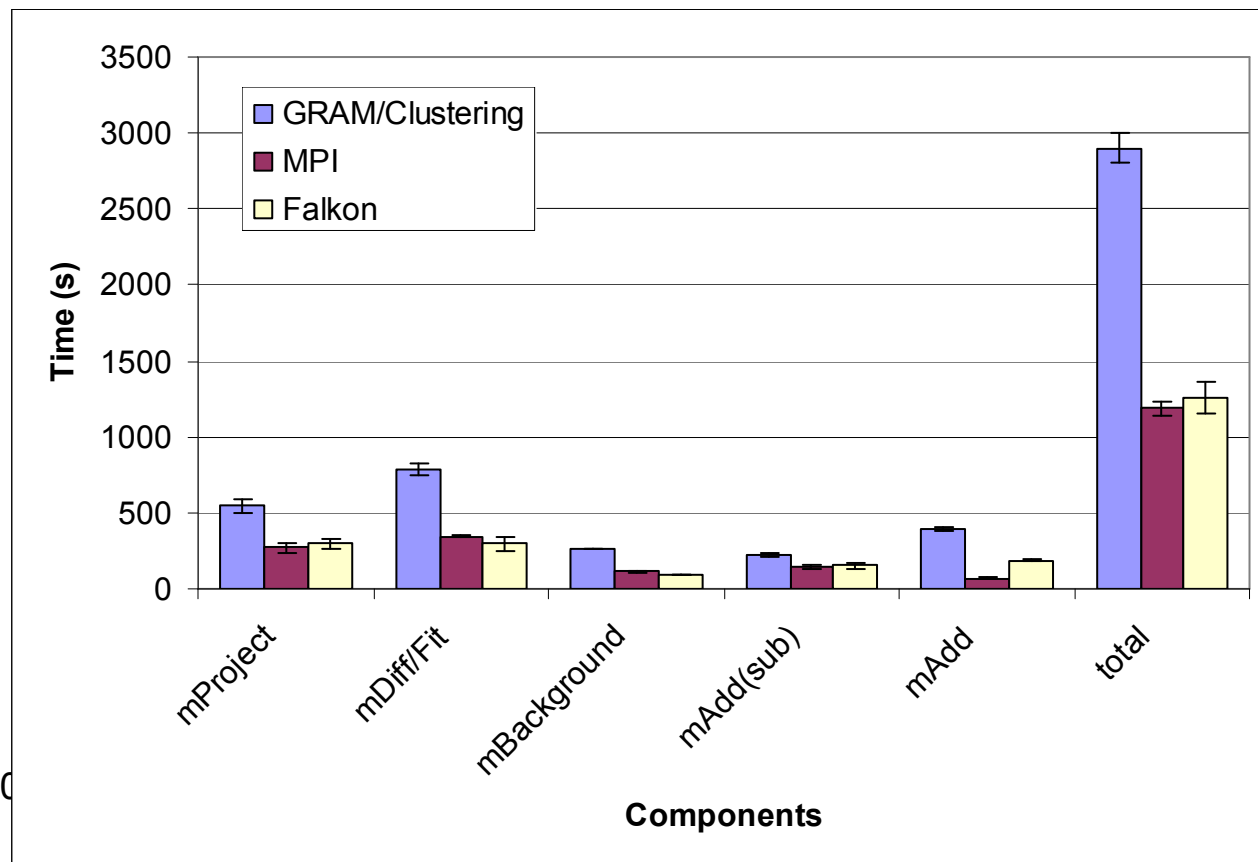
- GRAM vs. Falkon: 85%~90% lower run time
- GRAM/Clustering vs. Falkon: 40%~74% lower run time



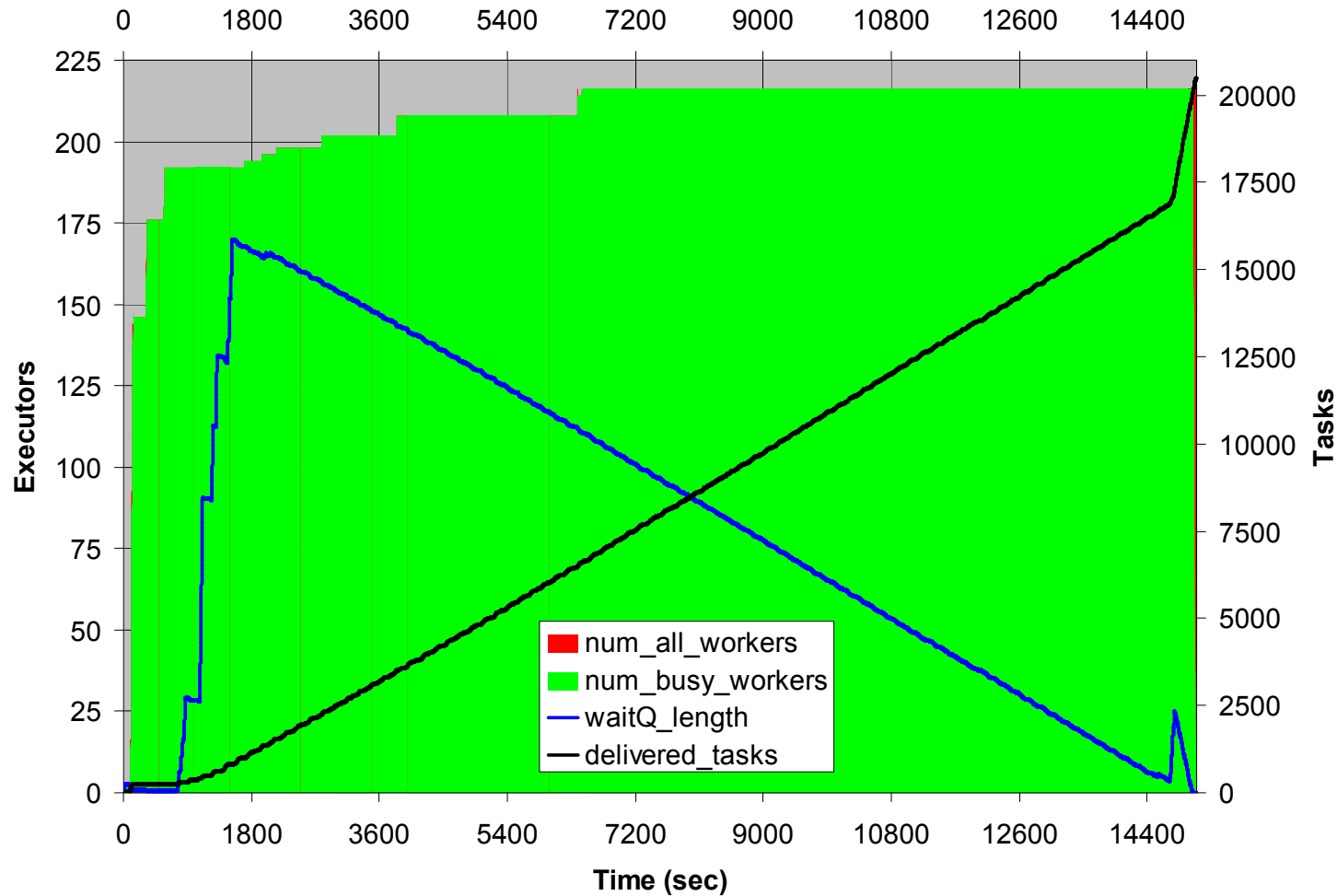
Applications: Montage



- 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

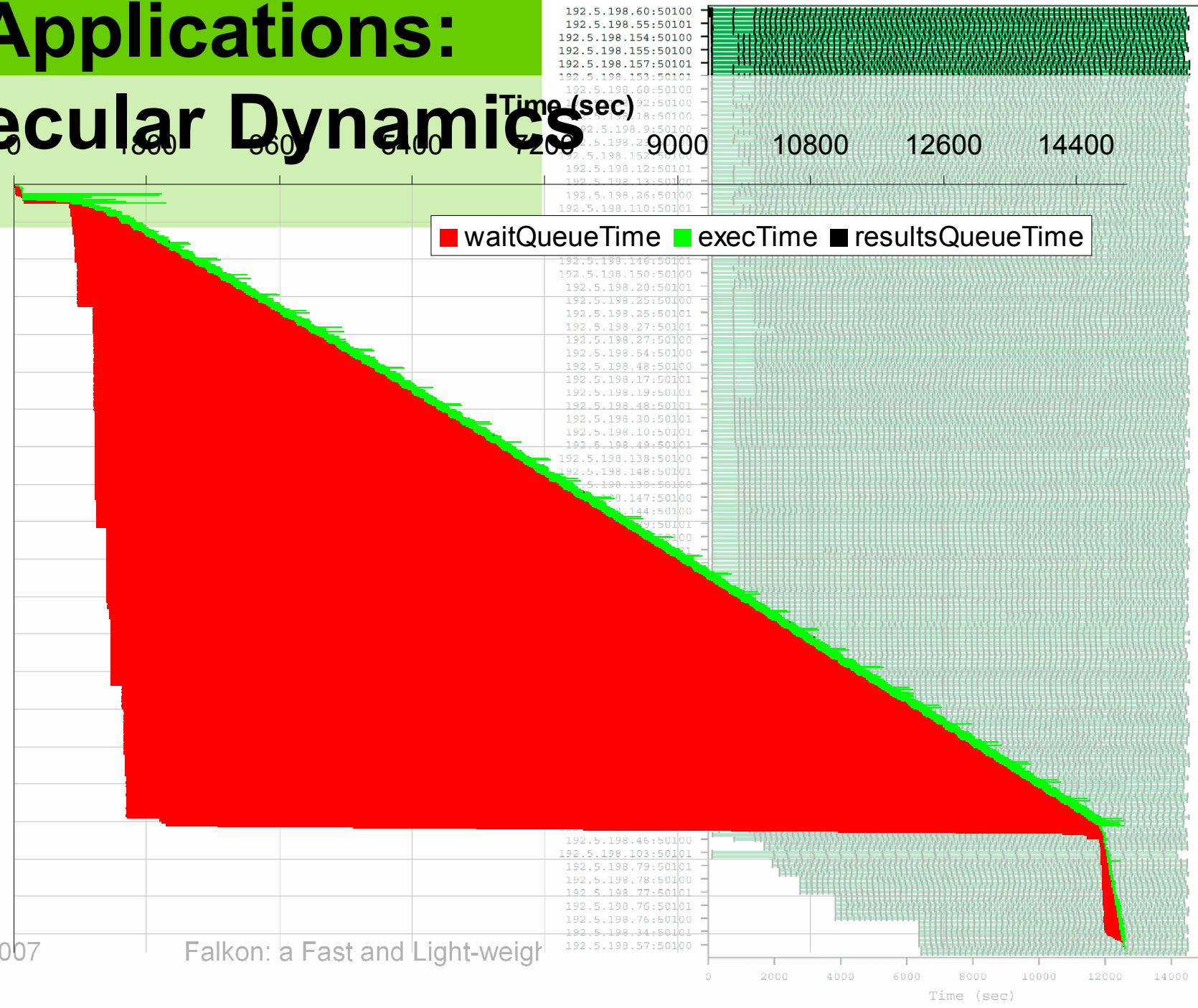


Applications: Molecular Dynamics

Task ID

1
1001
2001
3001
4001
5001
6001
7001
8001
9001
10001
11001
12001
13001
14001
15001
16001
17001
18001
19001
20001

■ waitQueueTime ■ execTime ■ resultsQueueTime

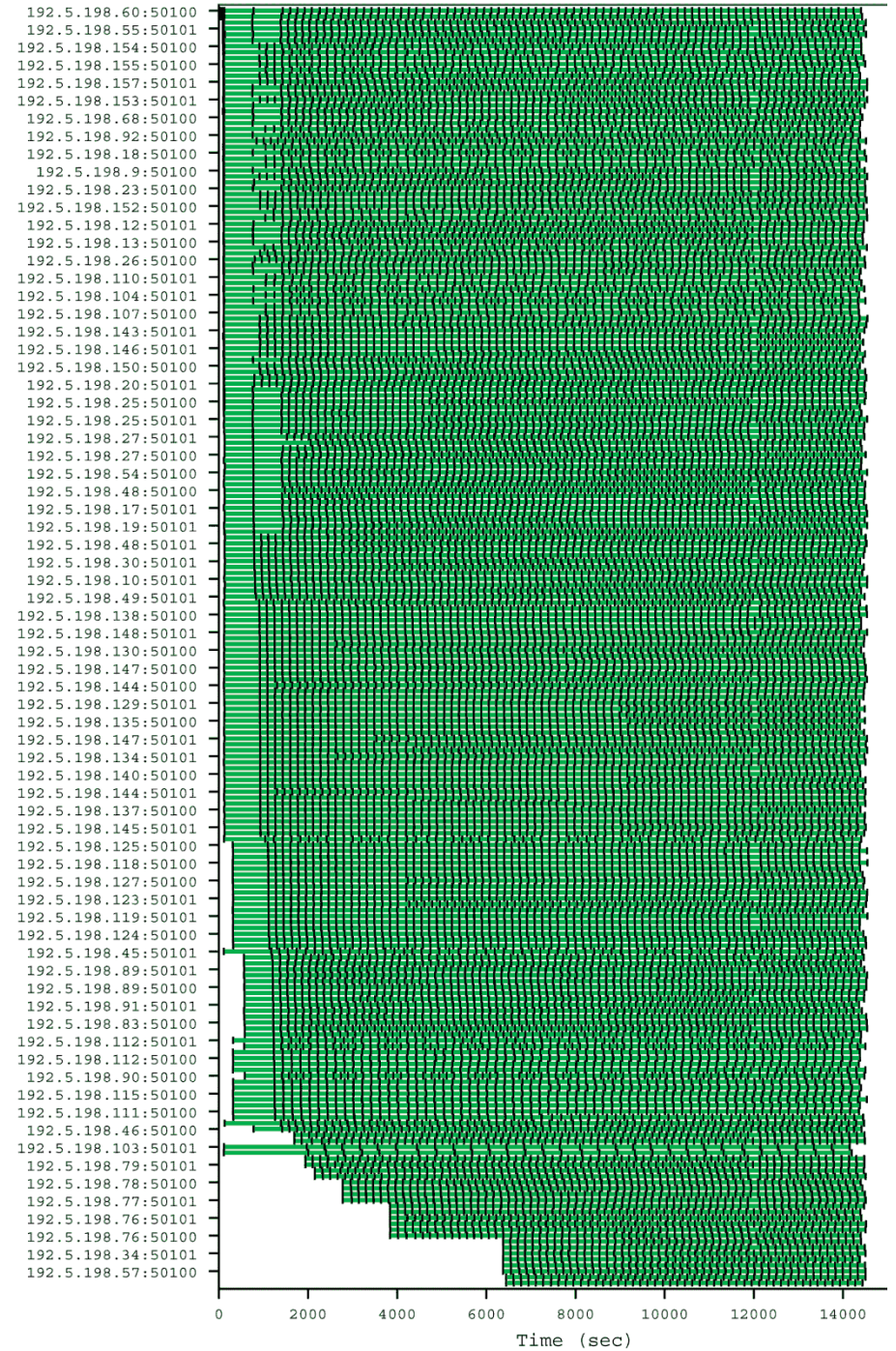
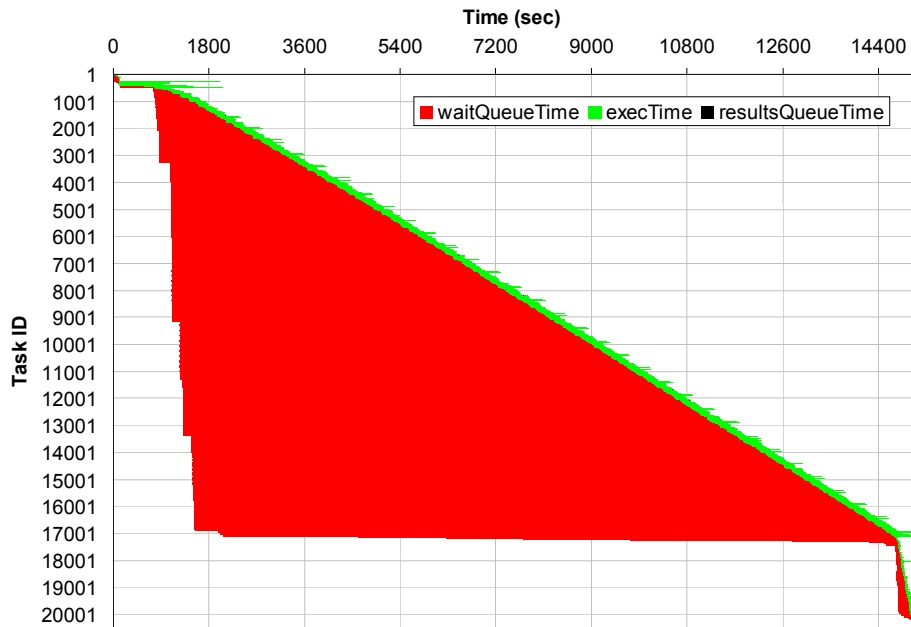


192.5.198.60:50100
192.5.198.55:50101
192.5.198.154:50100
192.5.198.155:50100
192.5.198.157:50101
192.5.198.153:50101
192.5.198.60:50100
192.5.198.92:50100
192.5.198.18:50100
192.5.198.9:50100
192.5.198.2:50100
192.5.198.15:50101
192.5.198.12:50101
192.5.198.13:50100
192.5.198.26:50100
192.5.198.110:50101
192.5.198.146:50101
192.5.198.150:50100
192.5.198.20:50101
192.5.198.25:50100
192.5.198.25:50101
192.5.198.27:50101
192.5.198.27:50100
192.5.198.54:50100
192.5.198.48:50100
192.5.198.17:50101
192.5.198.19:50101
192.5.198.48:50101
192.5.198.30:50101
192.5.198.10:50101
192.5.198.49:50101
192.5.198.138:50100
192.5.198.148:50101
192.5.198.130:50100
192.5.198.147:50100
192.5.198.144:50100
192.5.198.149:50101
192.5.198.46:50100
192.5.198.103:50101
192.5.198.79:50101
192.5.198.78:50100
192.5.198.77:50101
192.5.198.76:50101
192.5.198.76:50100
192.5.198.34:50101
192.5.198.57:50100

11/14/2007

Falkon: a Fast and Light-weight

Applications: Molecular Dynamic



11/14/2007

Falkon: a Fast and Light-weight

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.