THE UNIVERSITY OF CHICAGO

Argonne NATIONAL LABORATORY

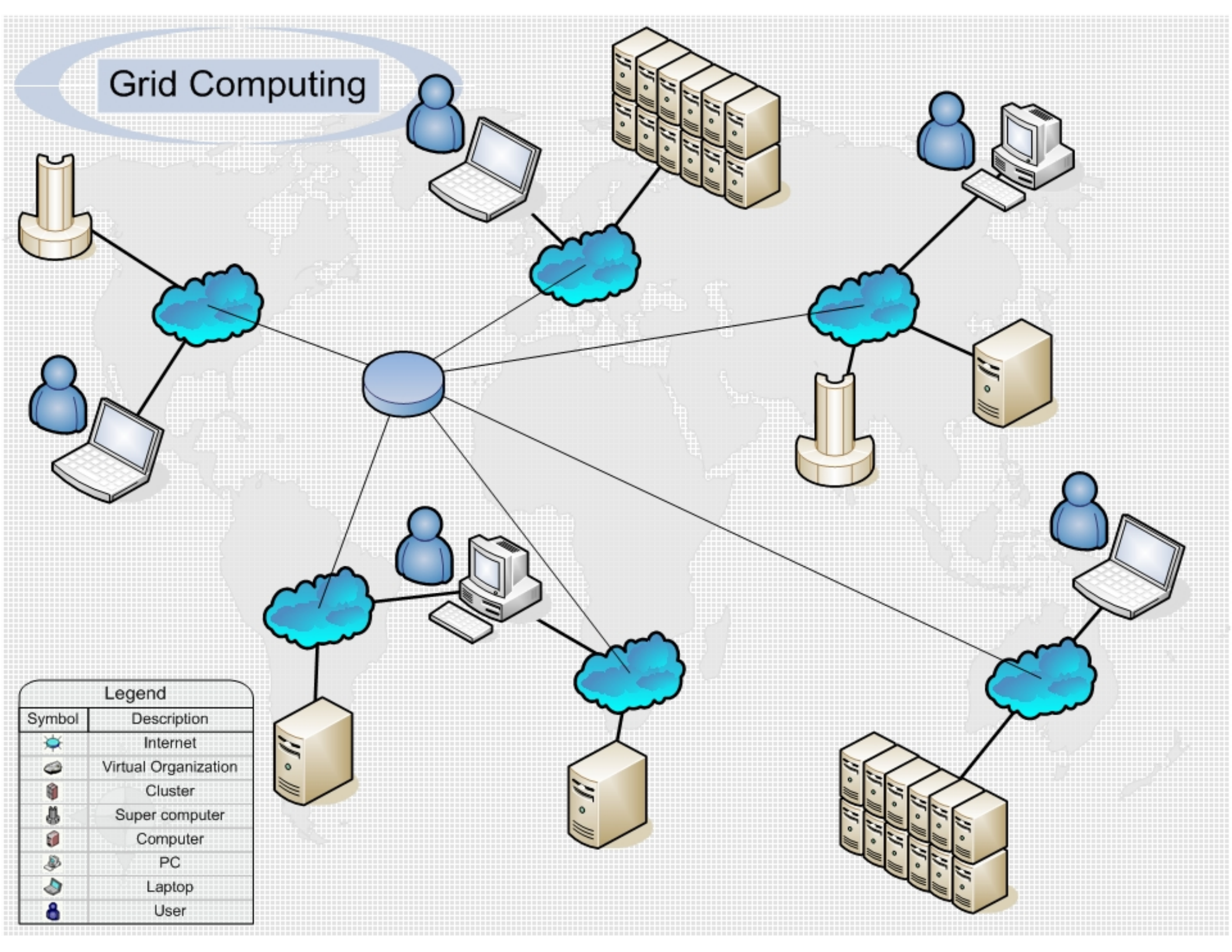
Falkon: a Fast and Light-weight tasK executiON framework for Grid Environments

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Joint work with:
Yong Zhao: Univ. of Chicago, CS
Catalin Dumitrescu: Univ. of Chicago, CS
Ian Foster: Univ. of Chicago, CS & CI, Argonne National Laboratory, MCS
Mike Wilde: University of Chicago, CI, Argonne National Laboratory, MCS

Funded by:
NSF TeraGrid: June 2005 – September 2006
NASA Ames Research Center GSRP: October 2006 – September 2007

CS Seminar
University of Chicago, Department of Computer Science
April 30th, 2007


Grid Computing
Legend
Symbol
Description
Internet
Virtual Organization
Cluster
Super computer
Computer
PC
Laptop
User

Grid Computing

- Grid Computing's focus:
  - **large-scale resource sharing:** direct access to computers, software, data
  - innovative applications
  - high-performance orientation
- The 'Grid problem':
  - **Definition:** flexible, secure, and ***coordinated resource sharing among dynamic collections of individuals, institutions, and resources***
  - **Challenges:** Security (Authentication, Authorization), ***resource management (resource access, resource discovery, scheduling, data management)***

Falkon: a Fast and Light-weight tasK executiON framework

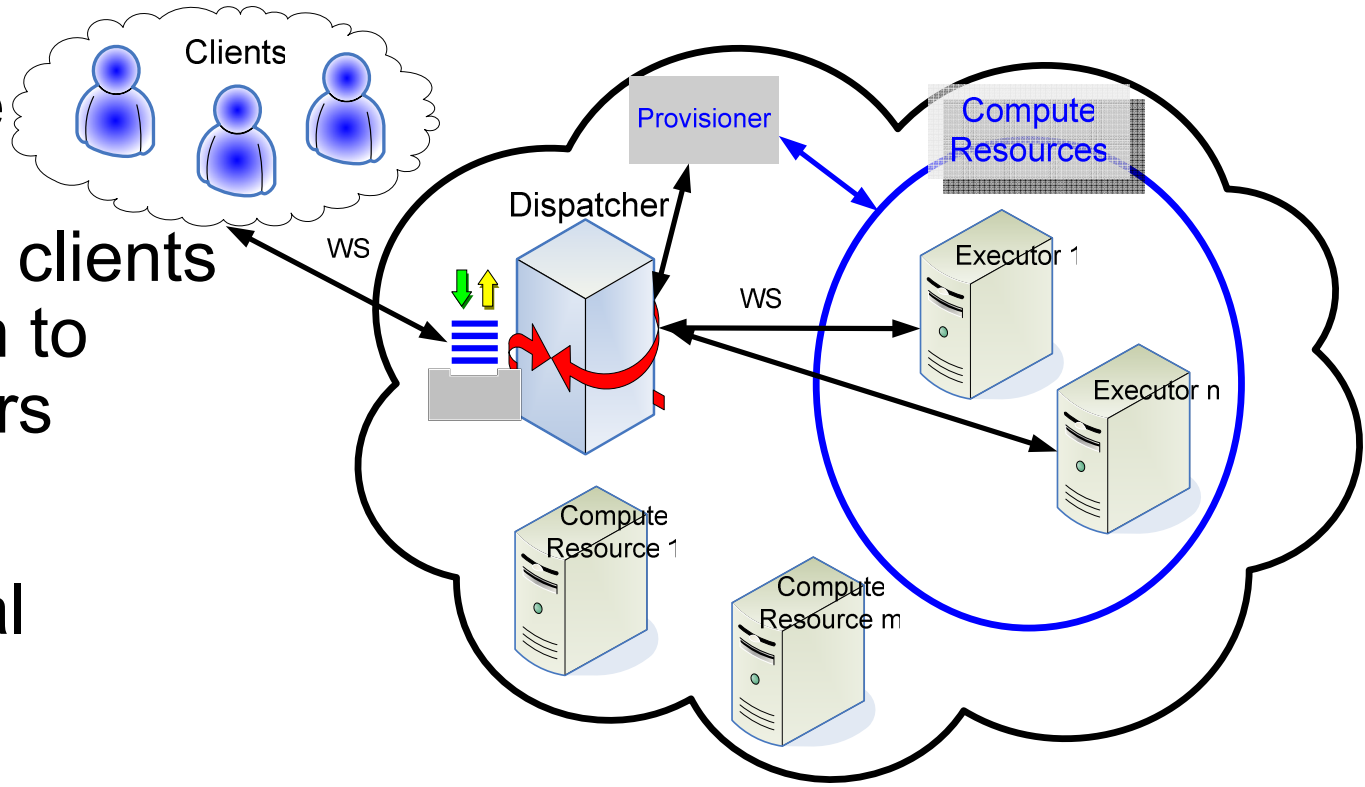
- ***Goal:*** enable the rapid and efficient execution of many independent jobs on large compute clusters
- Combines two techniques:
  - ***multi-level scheduling*** techniques to enable separate treatments of resource provisioning
  - a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers

Execution Model

- **Dispatch policy**
  - Round-robin
- **Replay policy**
- **Resource Acquisition Policy**
  - One-at-a-time
  - Additive
  - Exponential
  - All-at-once
  - Optimal
- **Resource Release Policy**
  - Centralized
  - Distributed

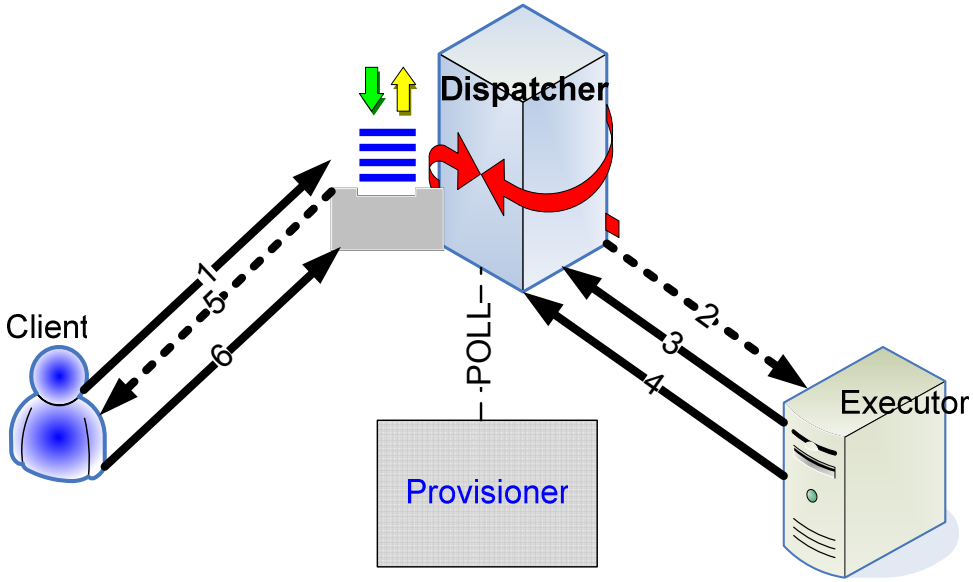
Falkon 2-Tier Architecture

- 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

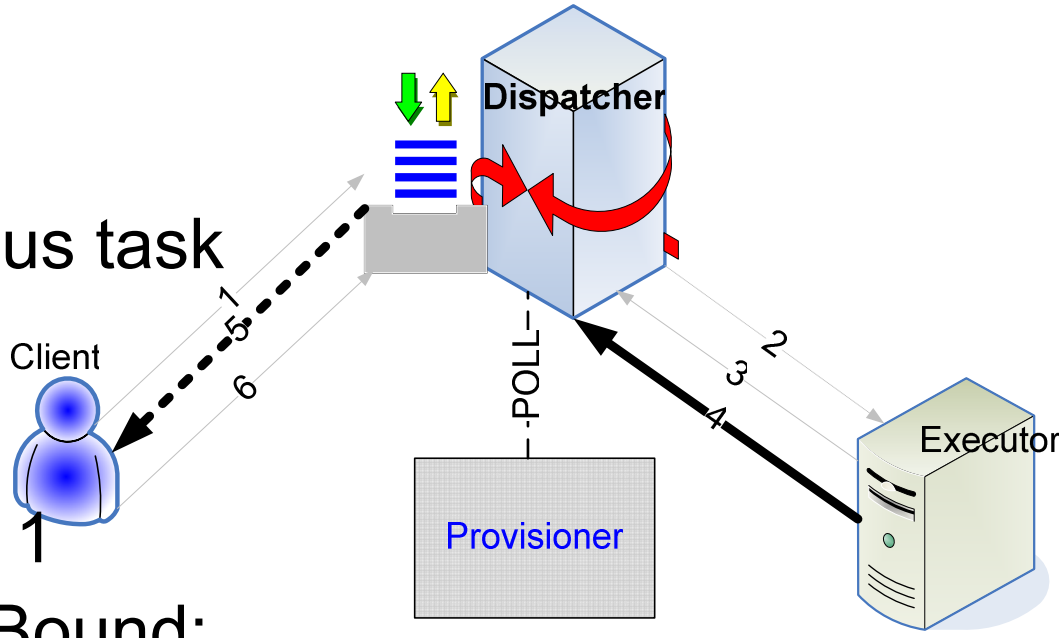
- Description:
  1. task(s) submit
  2. notification for work
  3. pick up task(s)
  4. deliver task(s) results
  5. notification for task result
  6. pick up task(s) results



- Worst case:
  - 4 WS messages (1, 3, 4, 6) and 2 notifications (2, 5) per task
- Best case:
  - 1 WS message (4) and 1 notification (5) message per task

Enhancements

- Bundling
  - Include multiple tasks per communication message
- Piggy-Backing
  - Attach next task to acknowledgement of previous task
- Message reduction:
  - Now: 6 → 2
  - General Lower Bound: 6 → 1
  - Application Specific Lower Bound: 6 → c, where c is a small positive value close to 0



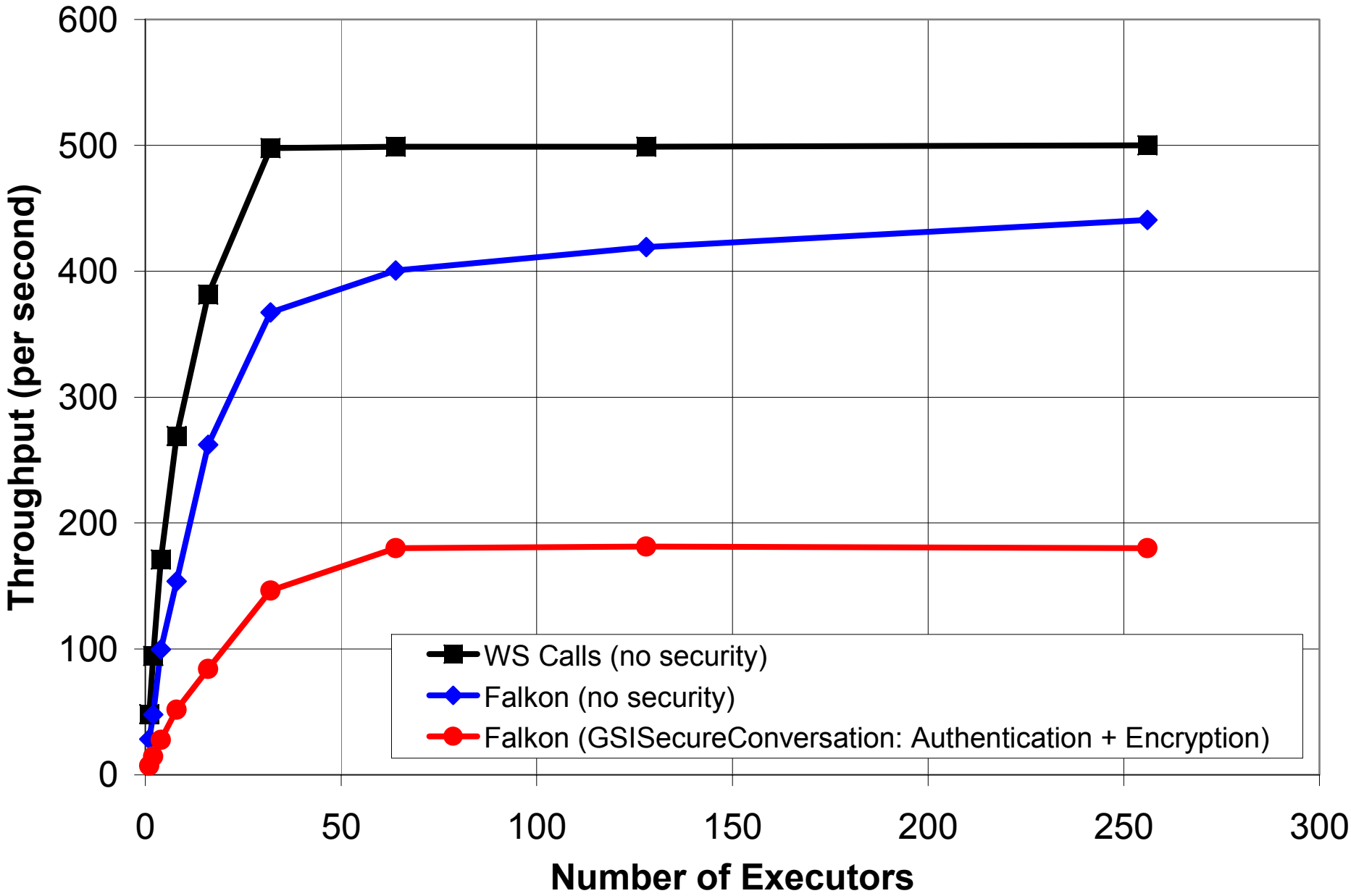
Testbeds

Name	# of Nodes	Processors	Memory	Network
TG_ANL_IA32_CLUSTER	98	Dual Xeon 2.4GHz	4GB	1Gb/s
TG_ANL_IA64_CLUSTER	64	Dual Itanium 1.5GHz	4GB	1Gb/s
TP_UC_x64_CLUSTER	122	Dual Opteron 2.2GHz	4GB	1Gb/s
UC_x64	1	Dual Xeon 3GHz w/ HT	2GB	100 Mb/s
UC_IA32	1	Intel P4 2.4GHz	1GB	100 Mb/s

Performance Metrics

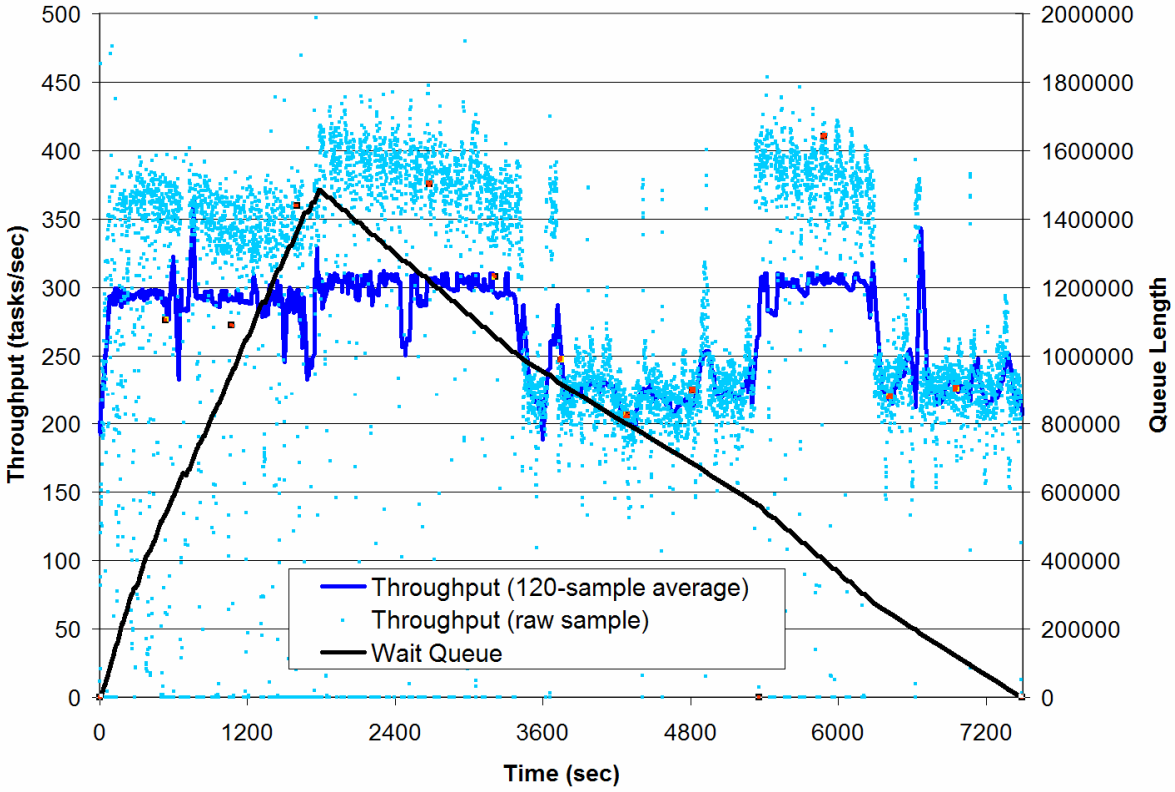
- Throughput (tasks/sec)
- Time to execute per task (ms)
- Execution time for a given set of tasks
- Speedup
- Efficiency

Throughput


Throughput (per second)
600
500
400
300
200
100
0
0
50
100
150
200
250
300
Number of Executors
WS Calls (no security)
Falkon (no security)
Falkon (GSISecureConversation: Authentication + Encryption)

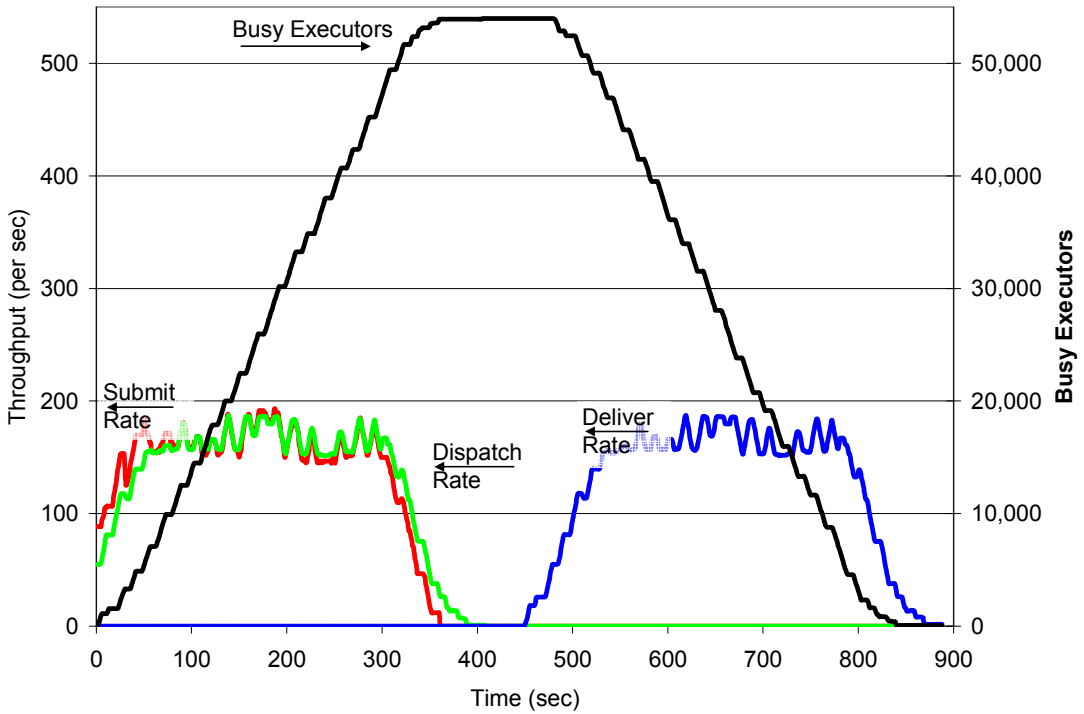
Falkon Scalability

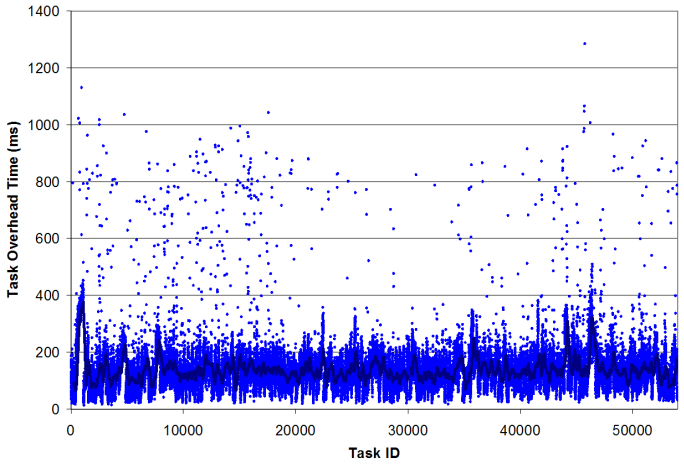
- 2M tasks processed (sleep 0) on 16 machines (32 executors)
- 7490 seconds ~ 267 tasks/sec
- Queue Length ~ 1.5M
- Sun JDK 1.4.2 with 1.5GB Heap
- Throughput variations → issues with GC



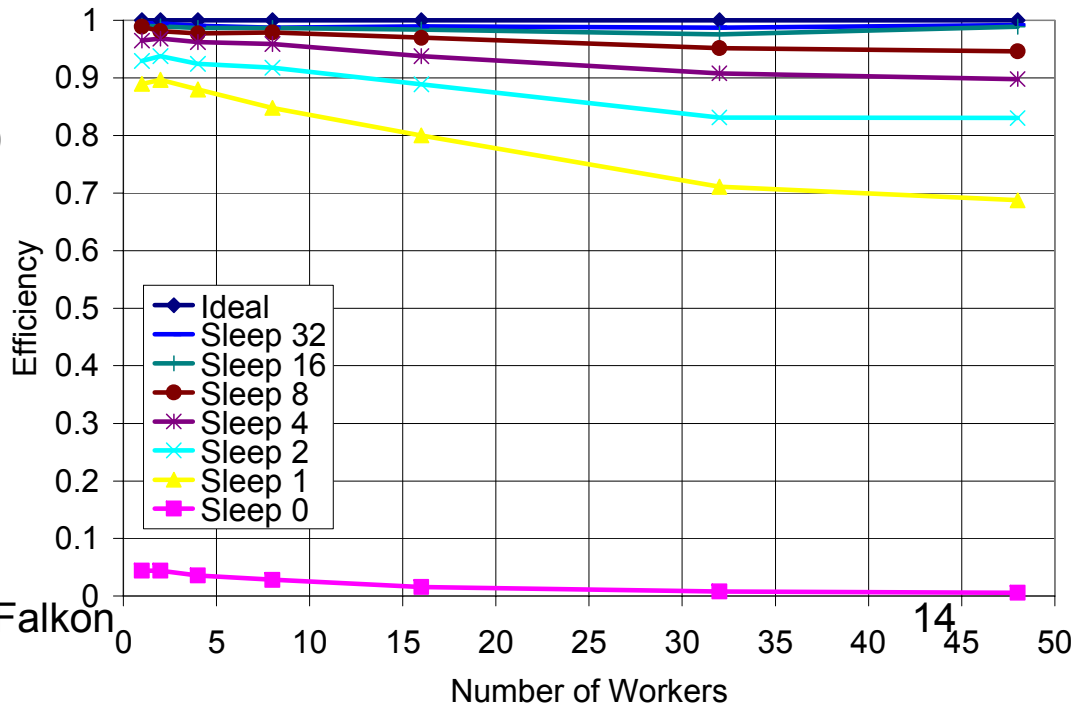
Falkon Scalability

- 54K tasks processed (sleep 480) on 60 machines (54K executors)
- 888 seconds ~ 61 tasks/sec
- Per task overhead: 127 ms, standard deviation 76ms
- Sun JDK 1.4.2 with 1.5GB Heap





Speedup & Efficiency



Security Overhead

- Experiment:
  - 1 client, 1 dispatcher, 1 executor
  - 30 tasks of sleep 60
  - Security:
    - None
    - GSITransport with authentication and encryption
    - GSISecureConversation with authentication and encryption
- MyCluster Comparison
  - Condor: 5%
  - SGE: 25%

	Exec Time (sec)	Exec Overhead %
Ideal Tasks Execution	1800.00	0.00%
No Security	1803.46	0.19%
GSI Transport (Authentication + Encryption)	1817.37	0.96%
GSI Secure Conversation (Authentication + Encryption)	1815.58	0.87%

Dynamic Resource Provisioning

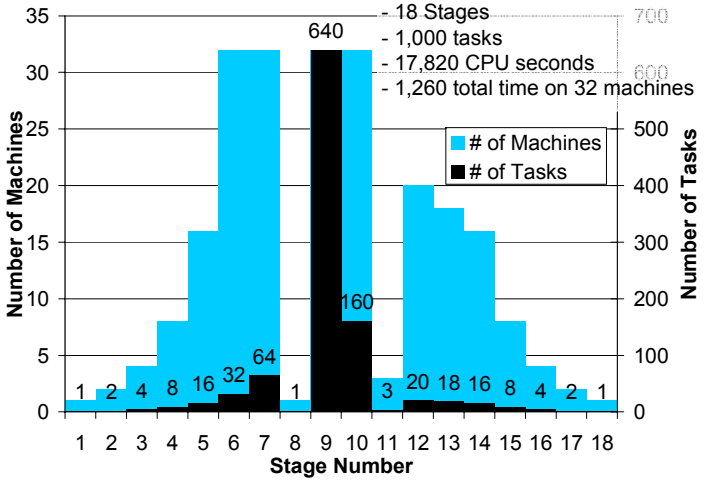
- Synthetic Workload
  - 18 stages
  - 1000 tasks
  - 17,820 CPU seconds
  - 1,260 seconds total time on 32 machines

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

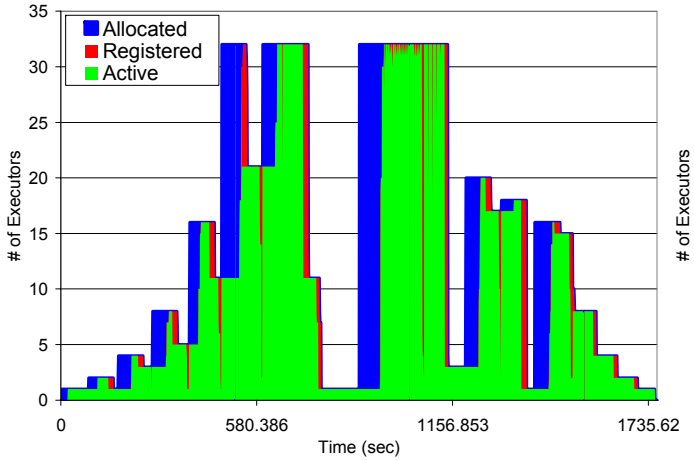
Dynamic Resource Provisioning

- 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

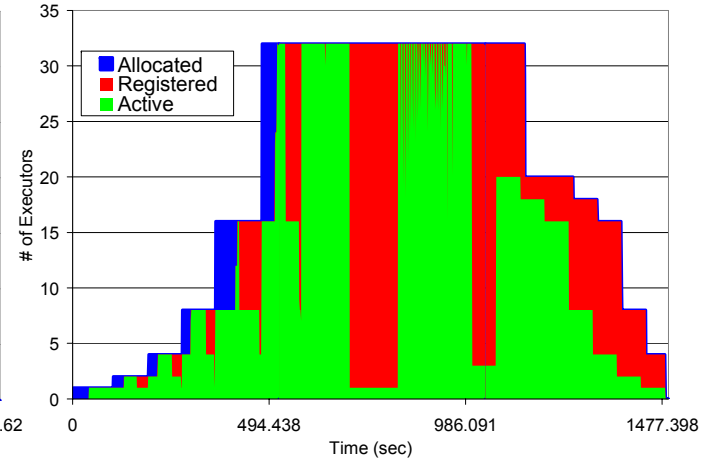
Ideal



Falkon-15



Falkon-180



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

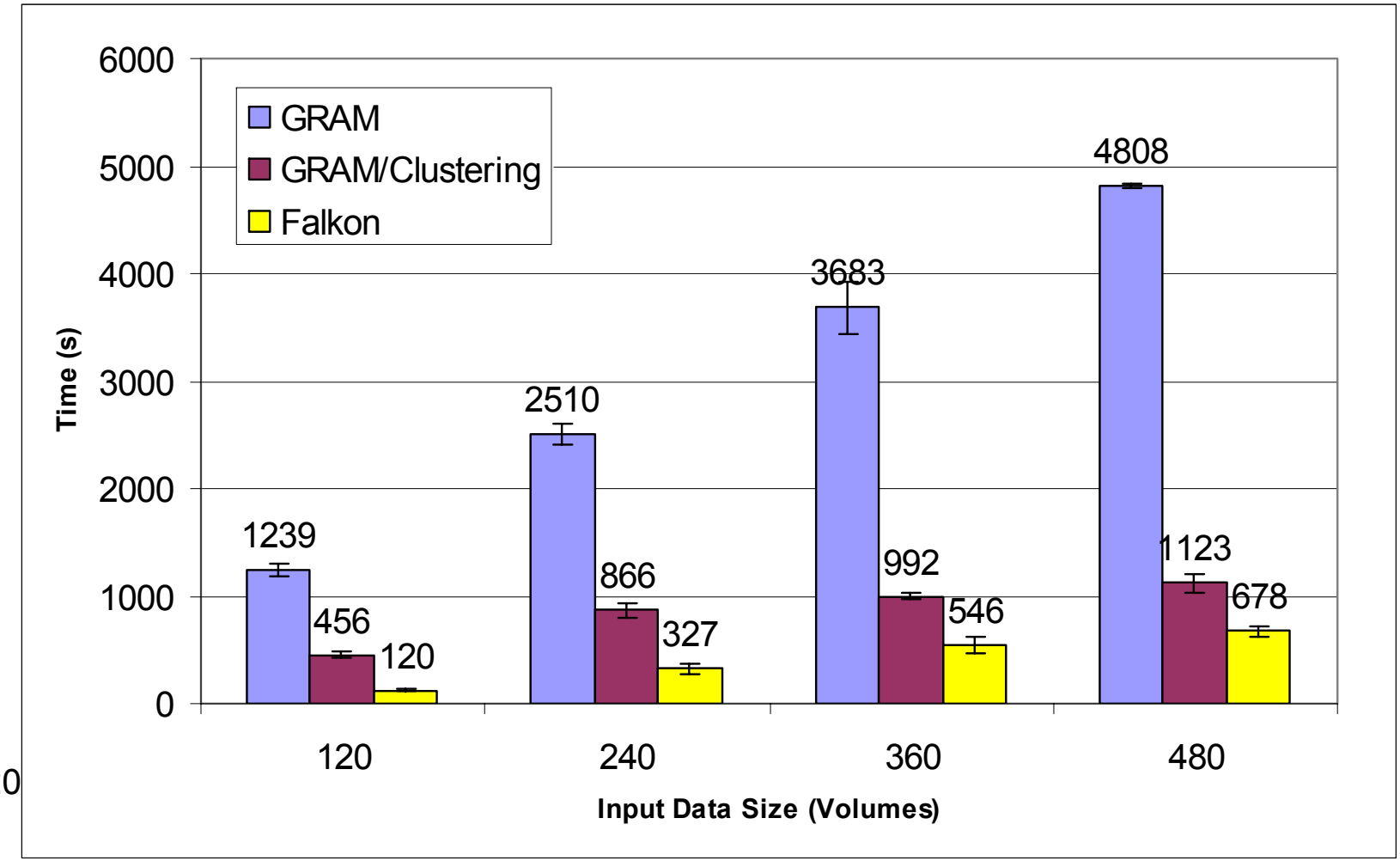
Applications

Application	#Jobs/workflow	#Levels
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

- Applications tested:
  - Medicine: fMRI
  - Astronomy: Montage
- Compared:
  - GRAM: Swift submitting each task as a GRAM job
  - GRAM/Clustering: Swift submitting clusters of tasks as GRAM jobs
  - MPI: Executing the entire application as an MPI program
  - Falkon: Swift submitting each task as a task to Falkon

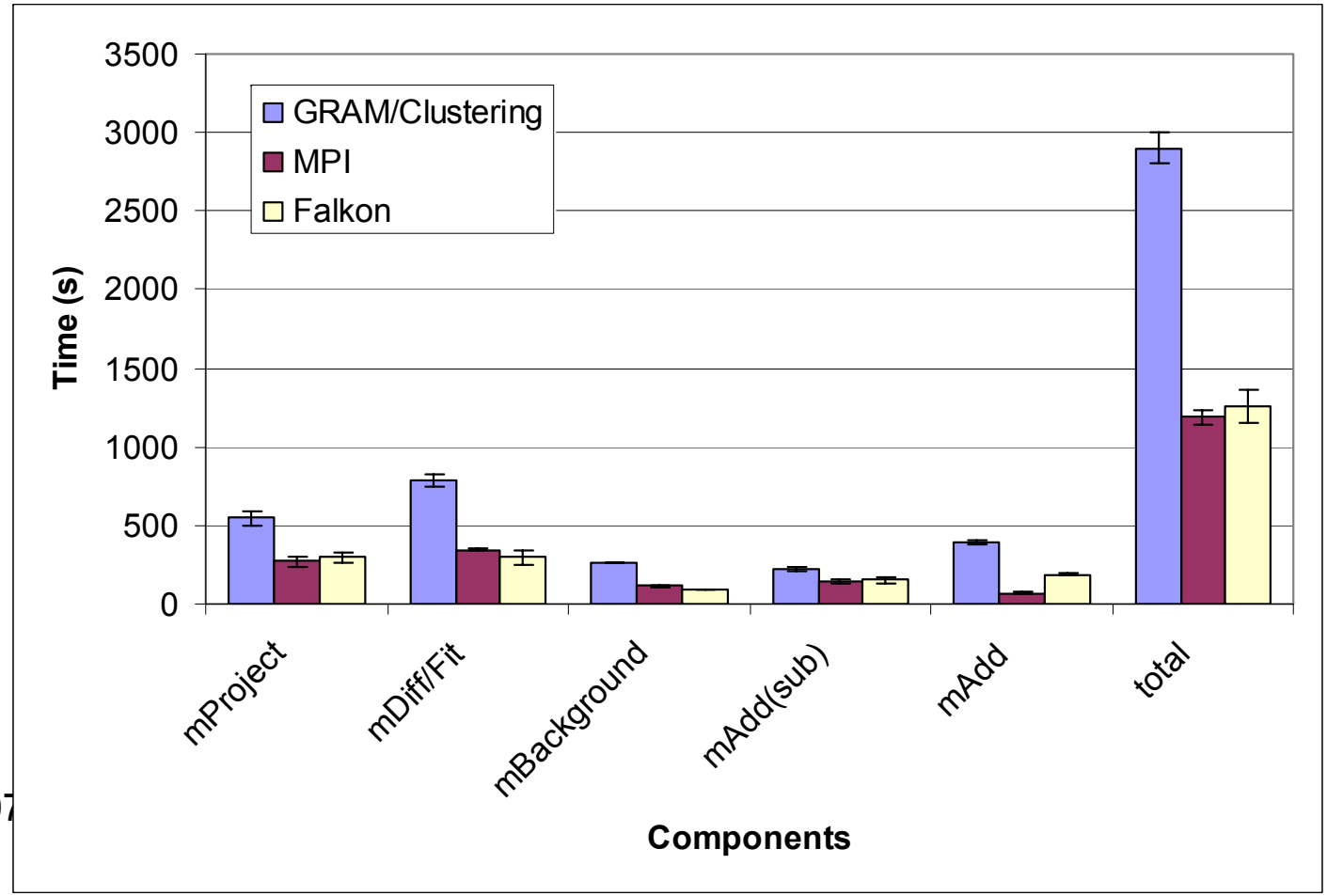
Applications: fMRI

- 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



Future Work

- Task pre-fetching
  - Overlap communication with computation
- Languages and Technologies
  - Move from Java to C
  - Move from Web Services to proprietary TCP-based protocol on the internals of Falkon
- Data Management
  - Data caching at executor local disk
    - Caching strategies: LRU, FIFO, popularity, …
  - Data replication at Grid site shared file systems
    - Replication strategies to meet a desired QoS

Future Work

- 3-Tier Architecture
  - Overview
    - Tier 1: Forwarder
      - Accept task submission from client(s) and forward them to appropriate dispatcher(s)
    - Tier 2: Dispatcher
      - Same as current Tier 1, but every Grid site (cluster) would have a separate dispatcher
    - Tier 3: Executor
      - Same as current Tier 2, and every executor would only communicate with respective local site dispatcher
  - Increases performance and scalability
  - Ensures that Falkon works with local access policies (firewalls, private IP spaces, etc)

More Information

- Collaborators:
  - Ioan Raicu, Computer Science Dept., The University of Chicago
  - Yong Zhao, Computer Science Dept., The University of Chicago
  - Catalin Dumitrescu, Computer Science Dept., The University of Chicago
  - Ian Foster, Math and Computer Science Div., Argonne National Laboratory & Computer Science Dept., The University of Chicago
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
- Contact information:
  - Falkon specific: iraicu@cs.uchicago.edu
  - Swift Specific: yongzh@cs.uchicago.edu
- Web: http://people.cs.uchicago.edu/~iraicu/research/Falkon/
- Source Code: http://people.cs.uchicago.edu/~iraicu/research/Falkon/Falkon_v0.8.tgz
- Related Projects:
  - Swift: http://www.ci.uchicago.edu/swift/index.php
  - AstroPortal: http://people.cs.uchicago.edu/~iraicu/research/AstroPortal/index.htm
- Documents:
  - Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. “Falkon: a Fast and Light-weight tasK executiON framework”, under review 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.