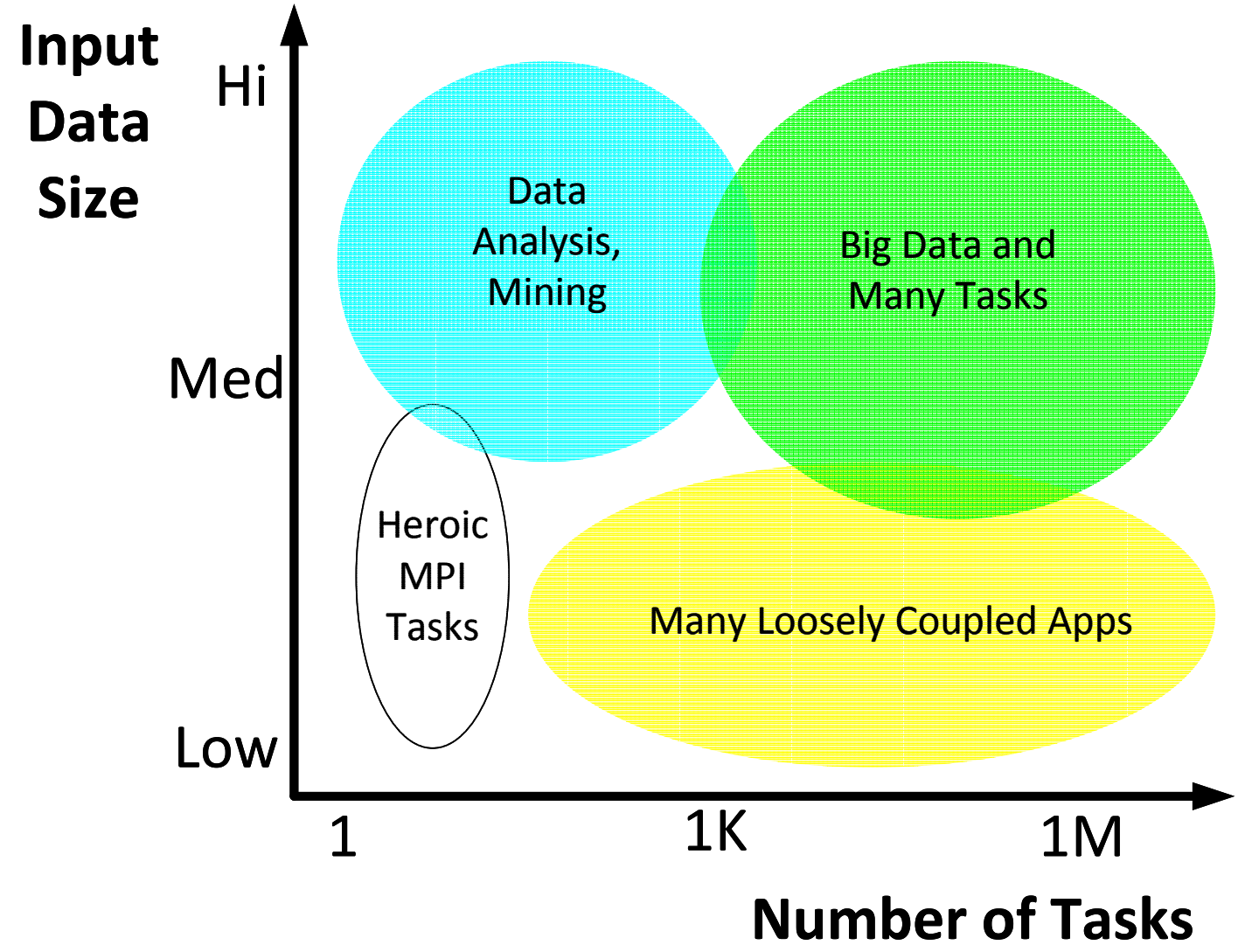
Falkon, a Fast and Light-weight tasK executiON framework for Clusters, Grids, and Supercomputers

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

In Collaboration with:
Ian Foster, Mike Wilde, Zhao Zhang, Catalin Dumitrescu, Yong Zhao, Pete Beckman, Kamil Iskra, Alex Szalay, Philip Little, Christopher Moretti, Amitabh Chaudhary, Douglas Thain, Ben Clifford, plus others…

IEEE/ACM Supercomputing 2008
Argonne National Laboratory Booth
November 19th, 2008

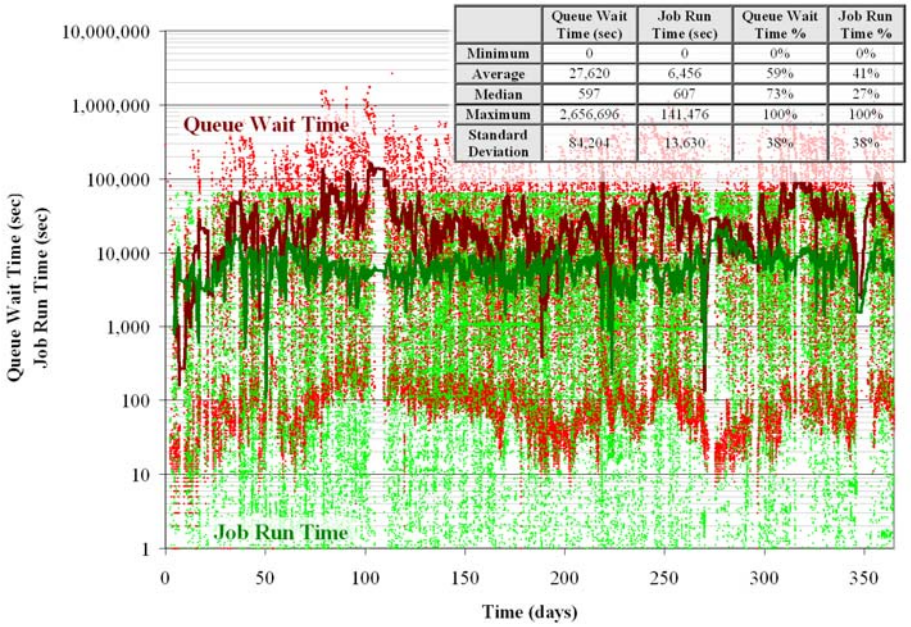
Problem Types



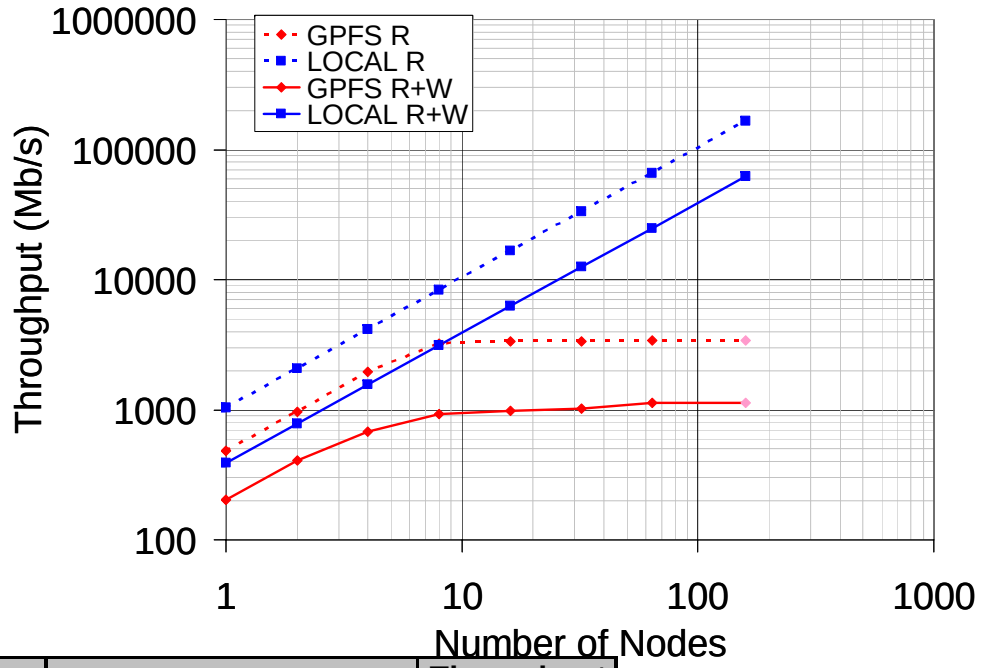
MTC: Many Task Computing

- Bridge the gap between HPC and HTC
- Loosely coupled applications with HPC orientations
- HPC comprising of multiple distinct activities, coupled via file system operations or message passing
- Emphasis on 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…

Obstacles running MTC apps in Clusters/Grids



	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%

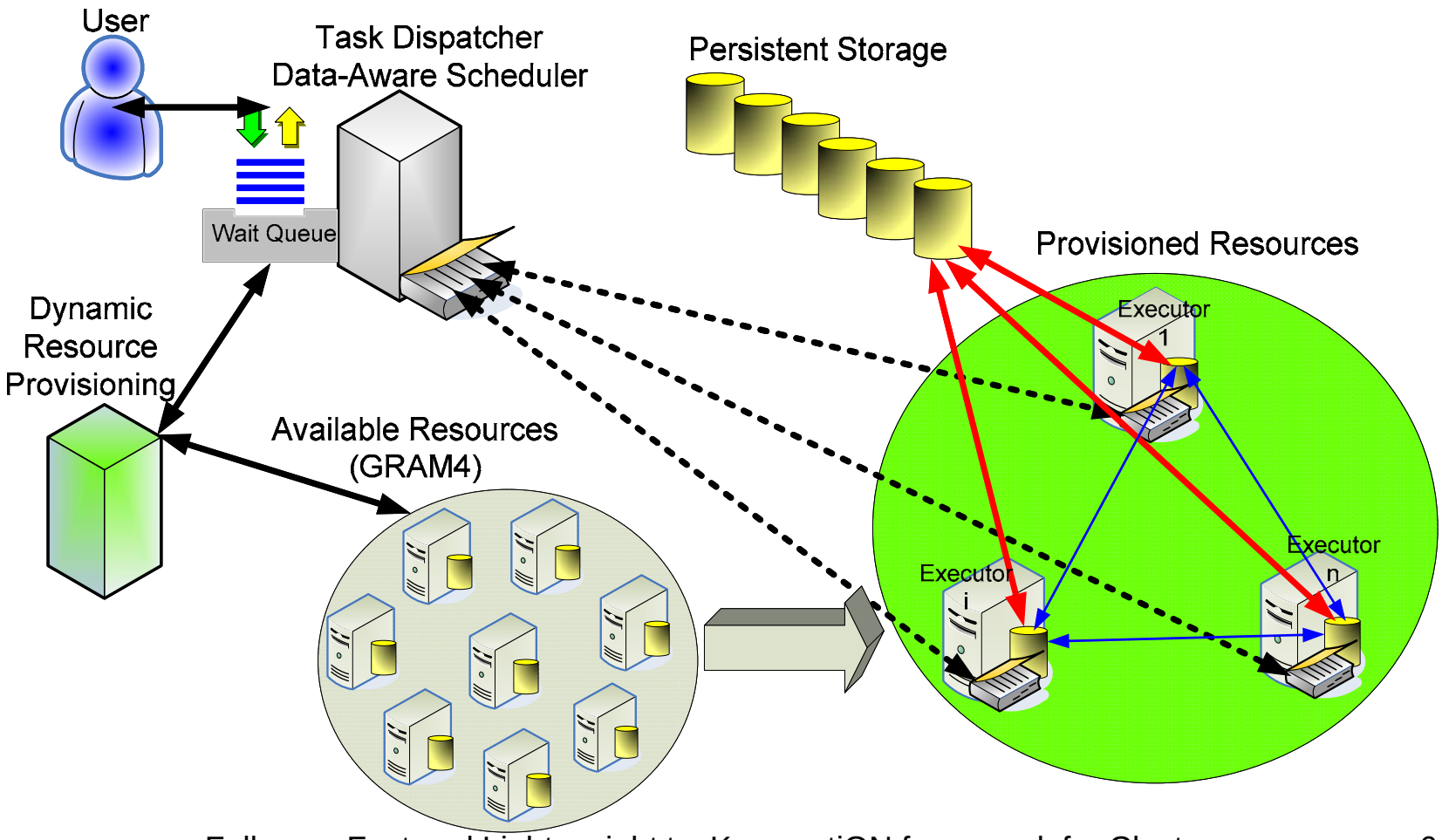


System	Comments	Throughput (tasks/sec)
Condor (v6.7.2) - Production	Dual Xeon 2.4GHz, 4GB	0.49
PBS (v2.1.8) - Production	Dual Xeon 2.4GHz, 4GB	0.45
Condor (v6.7.2) - Production	Quad Xeon 3 GHz, 4GB	2
Condor (v6.8.2) - Production		0.42
Condor (v6.9.3) - Development		11
Condor-J2 - Experimental	Quad Xeon 3 GHz, 4GB	22

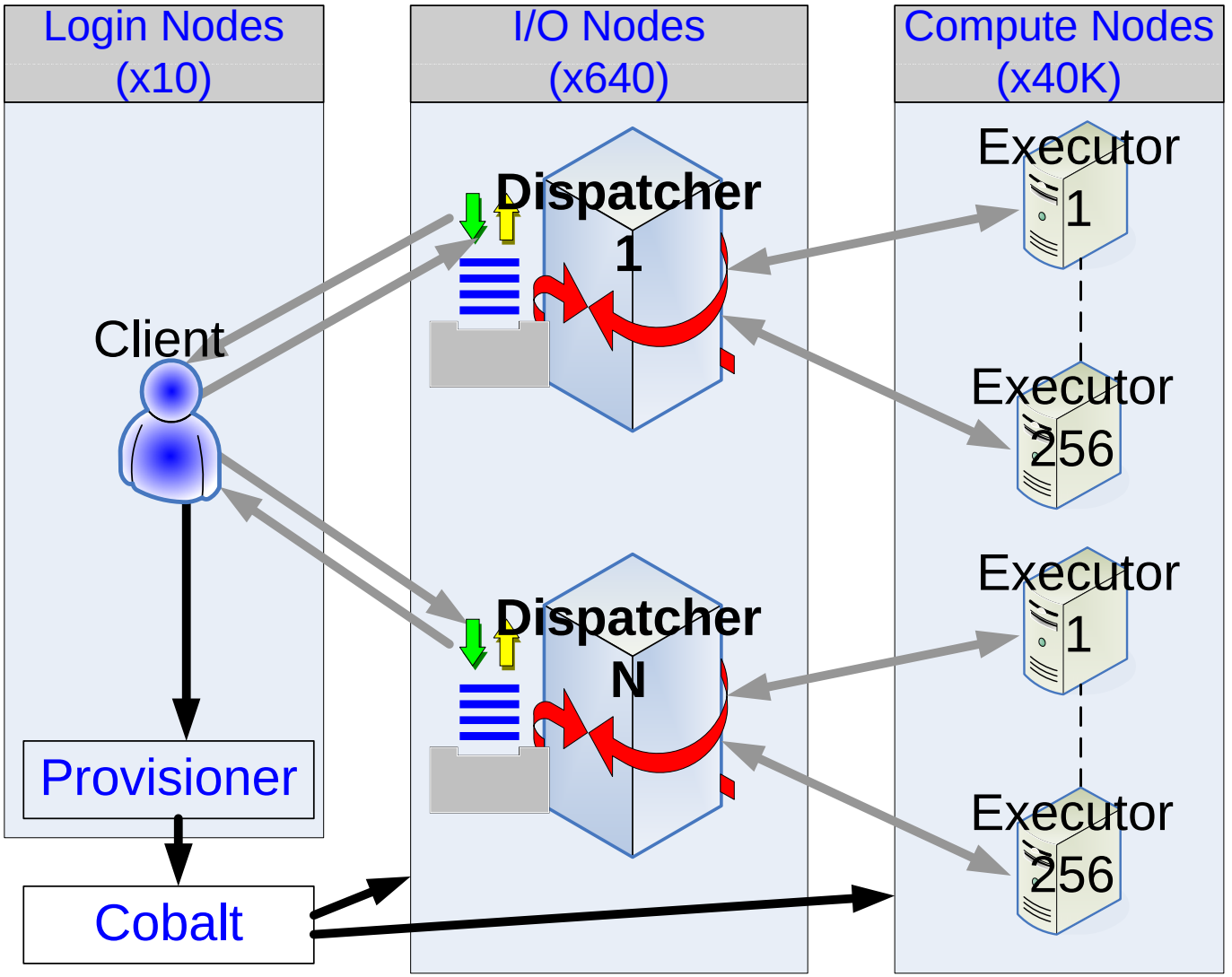
Solutions

- 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
- Swift: A parallel programming system for loosely coupled applications
  - Applications cover many domains: Astronomy, astro-physics, medicine, chemistry, economics, climate modeling, data analytics

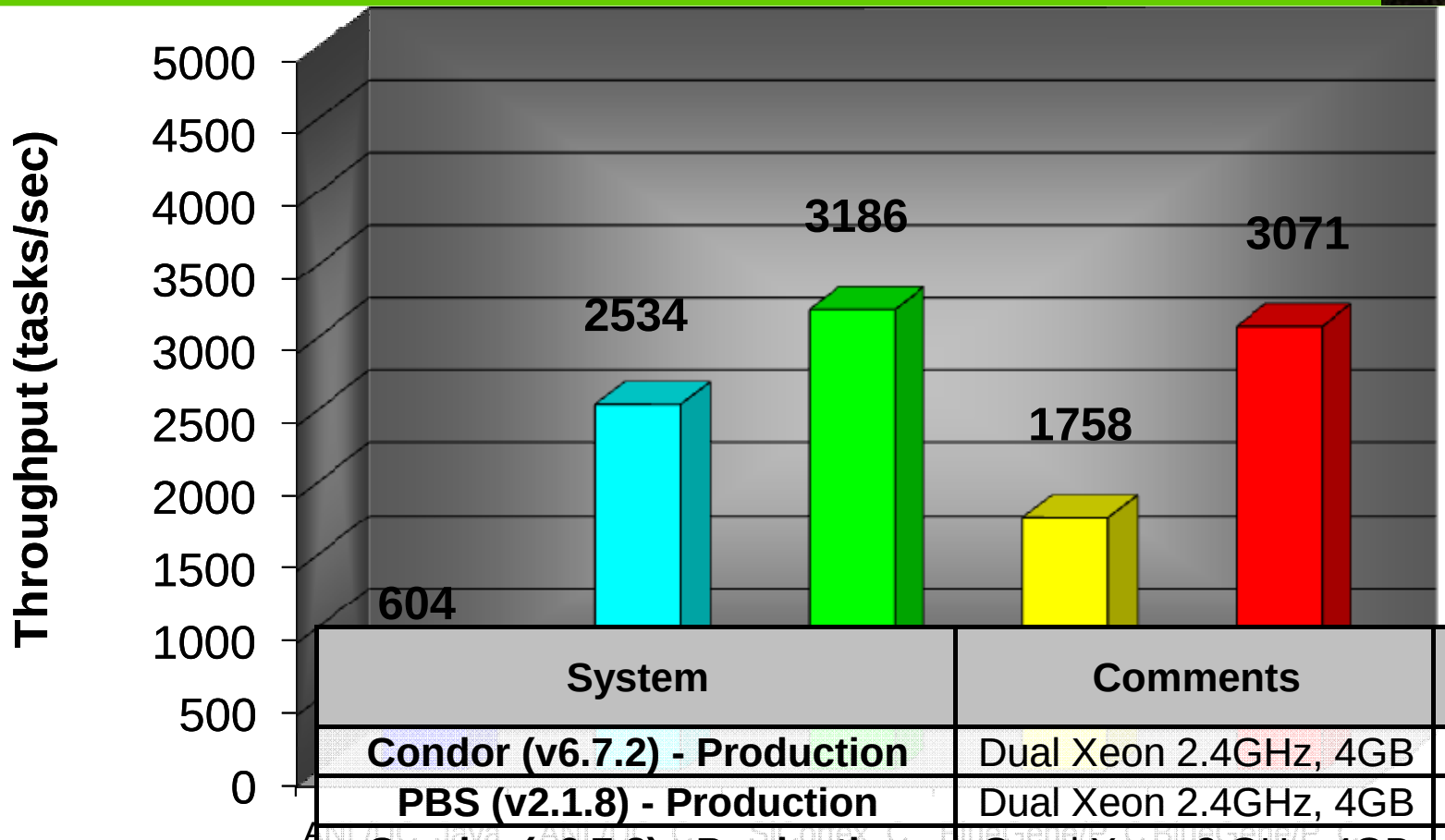
Falkon Overview



Distributed Falkon Architecture

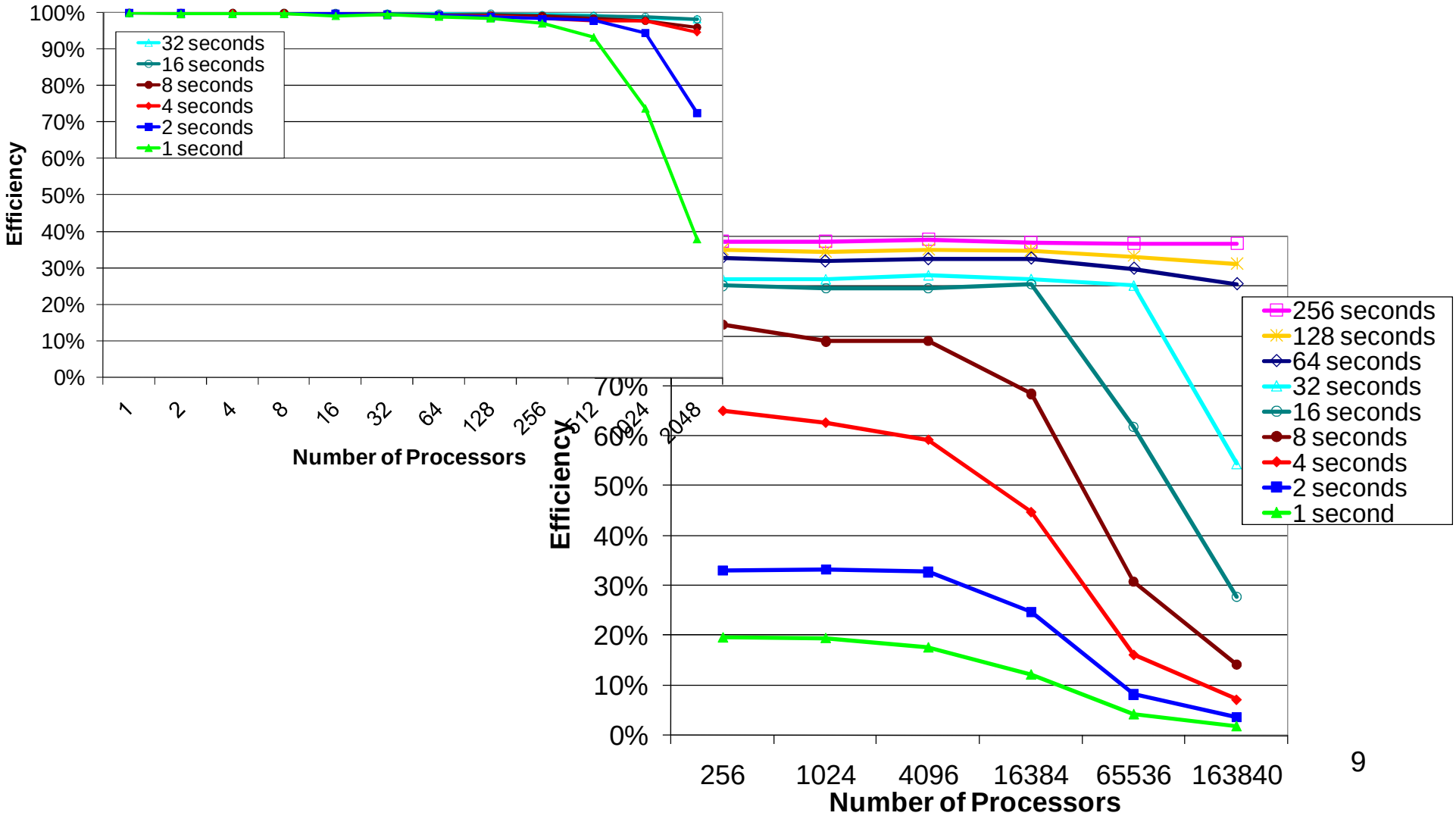


Dispatch Throughput

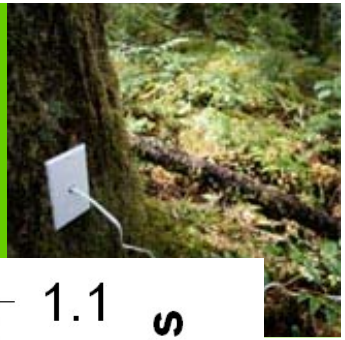


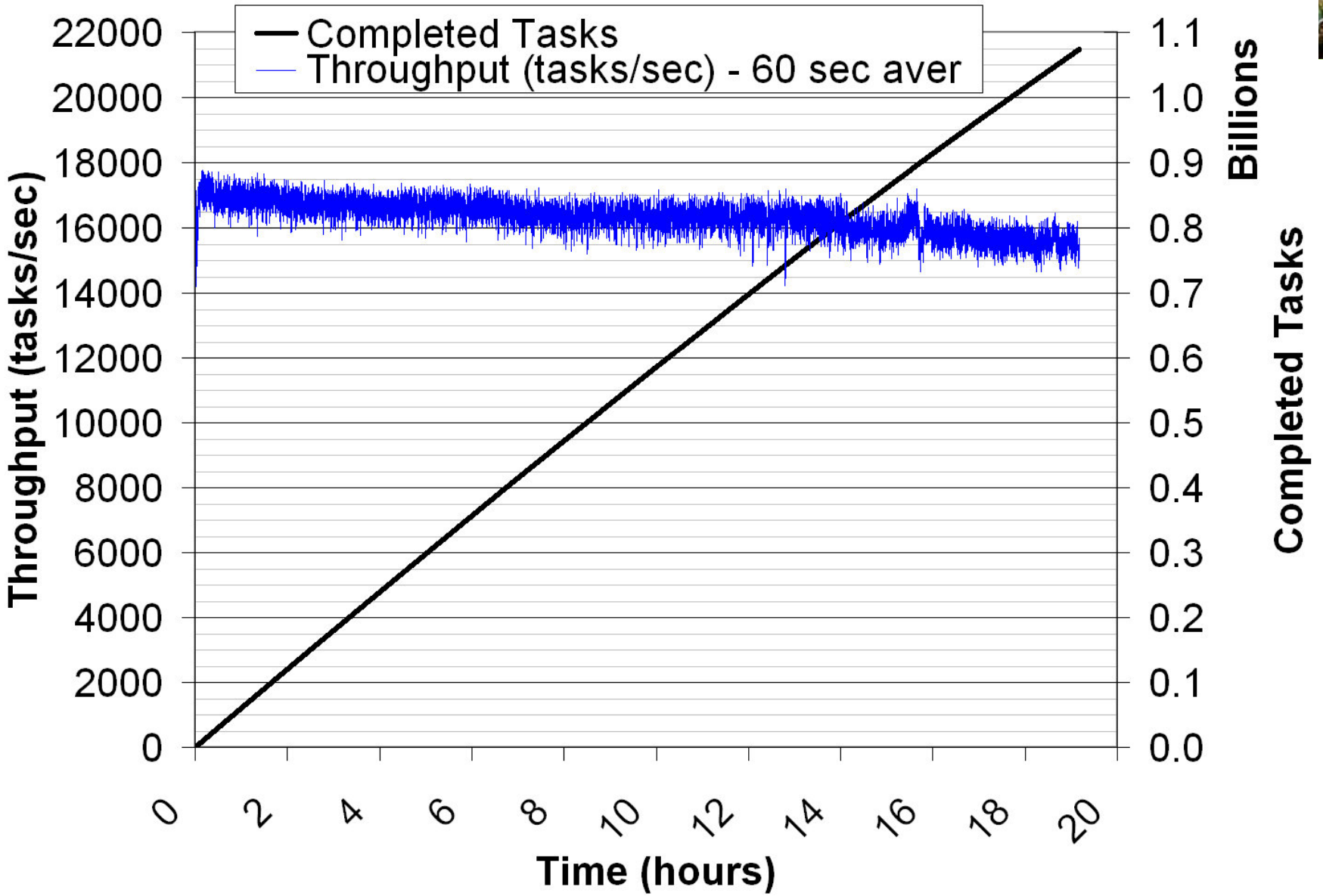
System	Comments	Throughput (tasks/sec)
Condor (v6.7.2) - Production	Dual Xeon 2.4GHz, 4GB	0.49
PBS (v2.1.8) - Production	Dual Xeon 2.4GHz, 4GB	0.45
Condor (v6.7.2) - Production	Quad Xeon 3 GHz, 4GB	2
Condor (v6.8.2) - Production		0.42
Condor (v6.9.3) - Development		11
Condor-J2 - Experimental	Quad Xeon 3 GHz, 4GB	22

Efficiency


100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Efficiency
32 seconds
16 seconds
8 seconds
4 seconds
2 seconds
1 second
1
2
4
8
16
32
64
128
256
512
1024
2048
Number of Processors
256 seconds
128 seconds
64 seconds
32 seconds
16 seconds
8 seconds
4 seconds
2 seconds
1 second
256
1024
4096
16384
65536
163840
Number of Processors

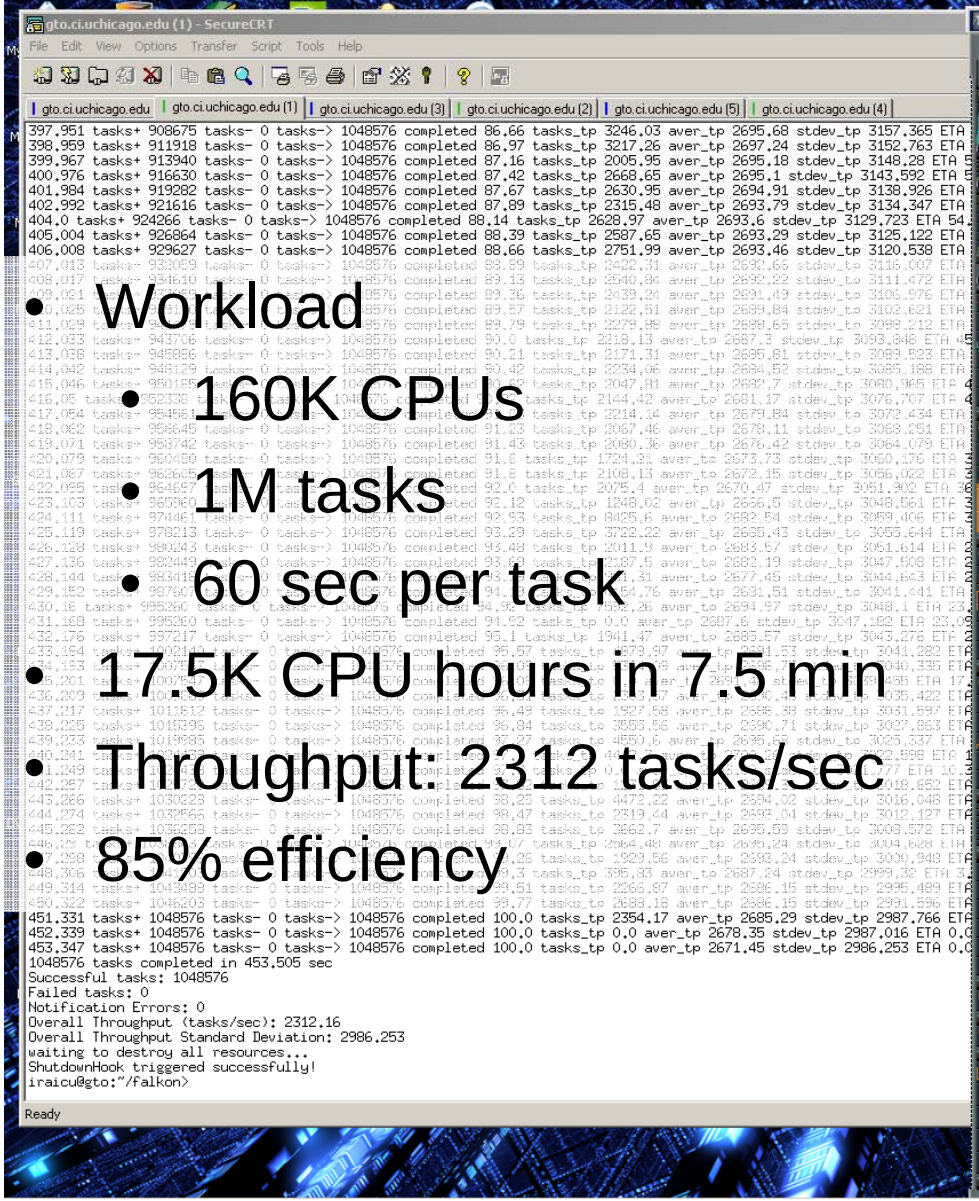
Falkon Endurance Test



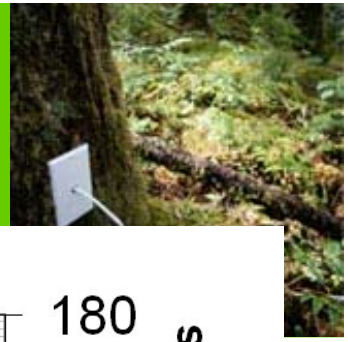


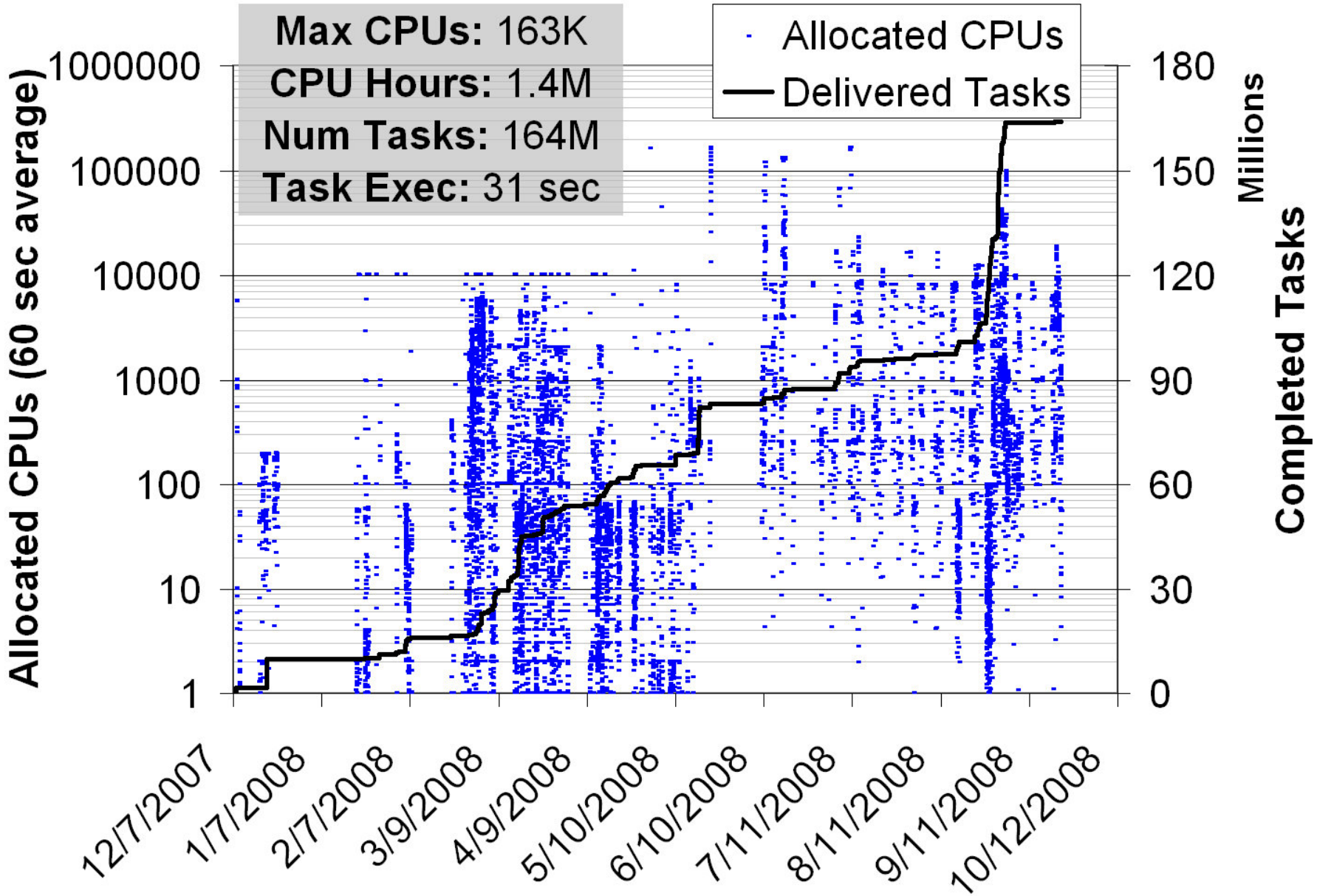
Falkon Monitoring

- Workload
  - 160K CPUs
  - 1M tasks
  - 60 sec per task
- 17.5K CPU hours in 7.5 min
- Throughput: 2312 tasks/sec
- 85% efficiency

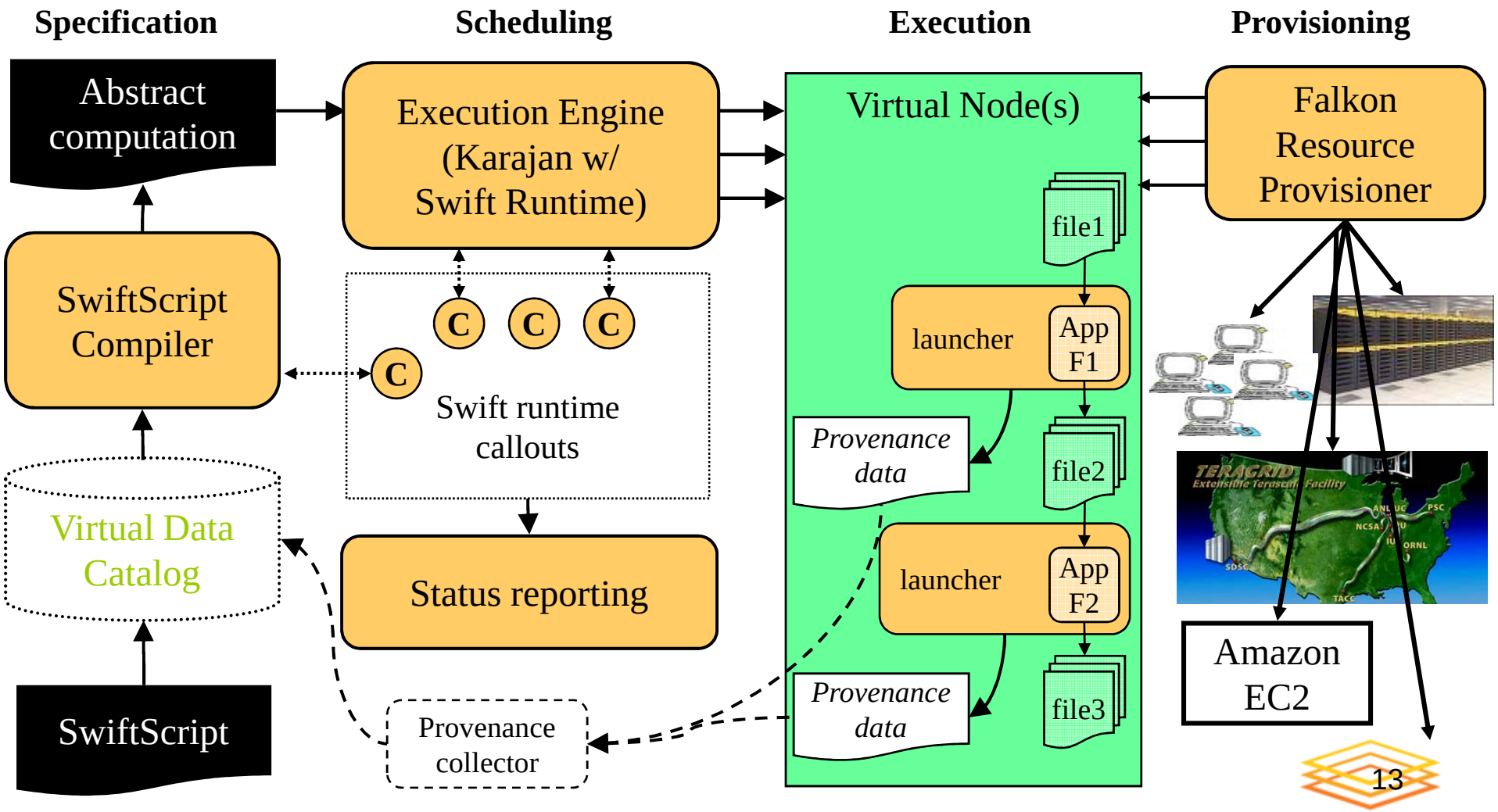


Falkon Activity History (10 months)



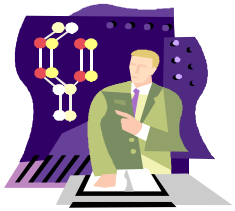


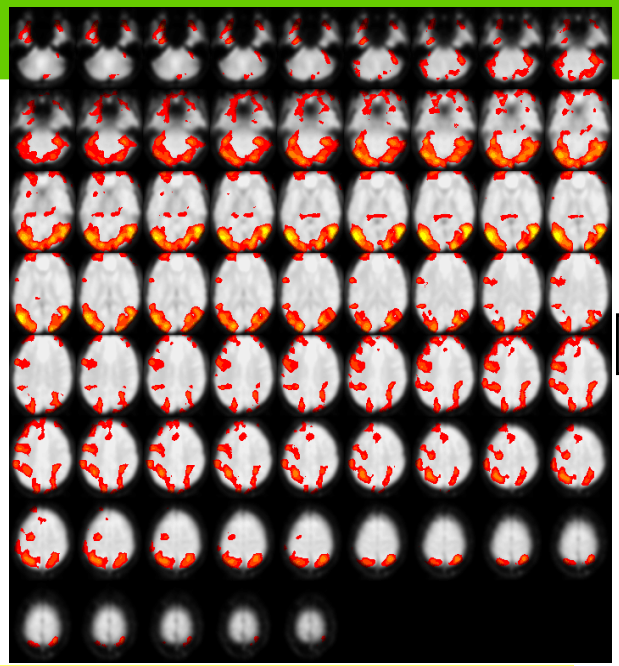
Swift Architecture

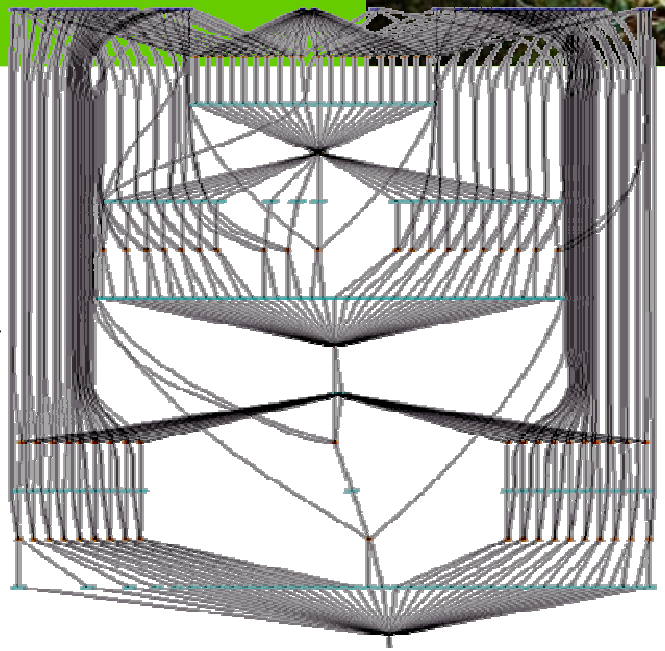


Open Science Grid

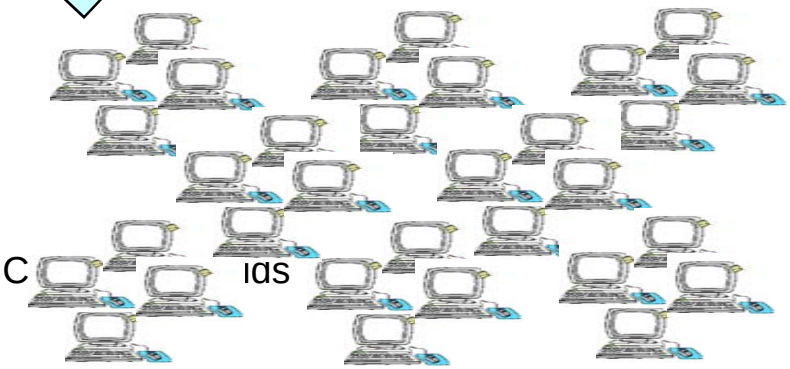
Functional MRI (fMRI)





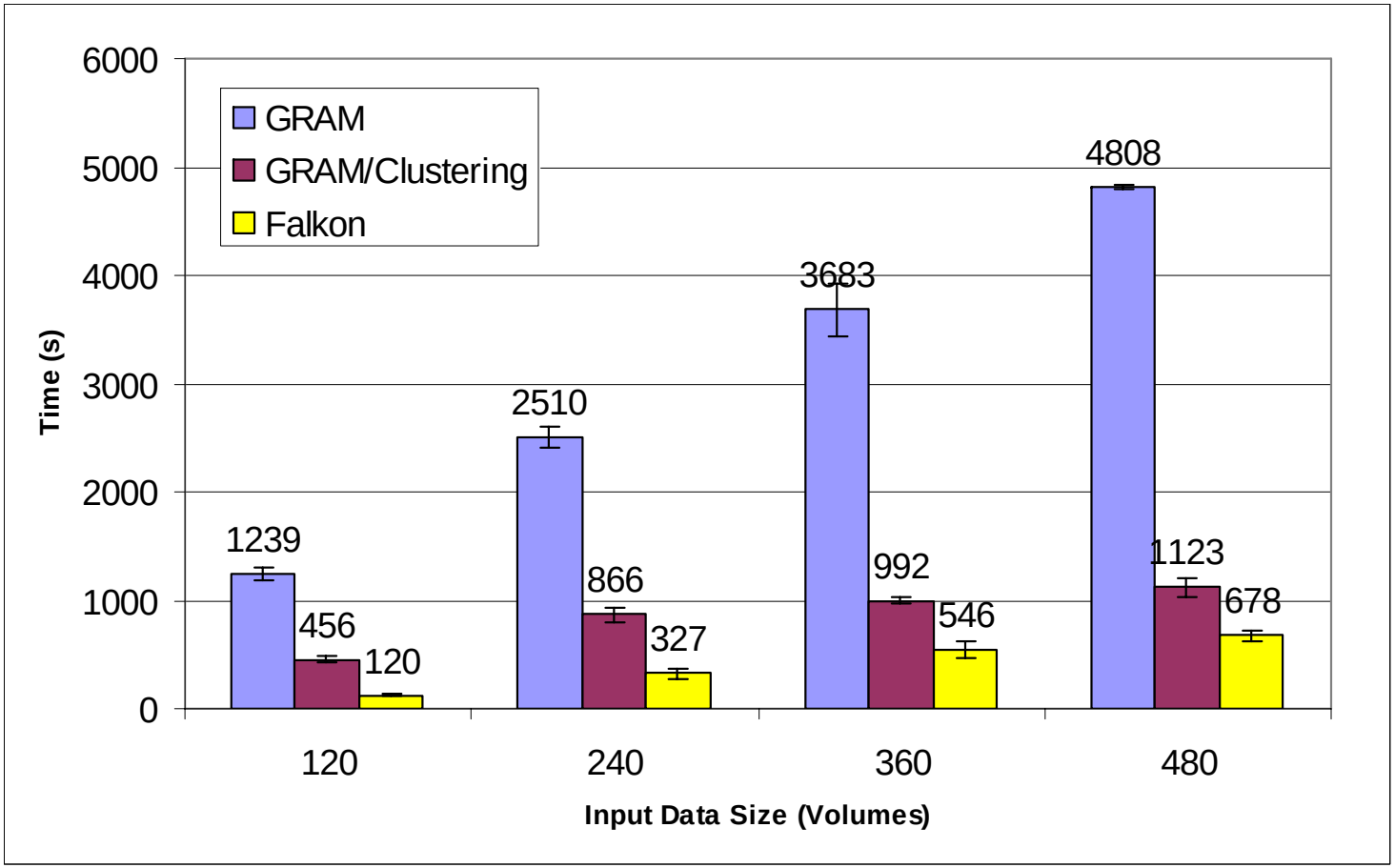


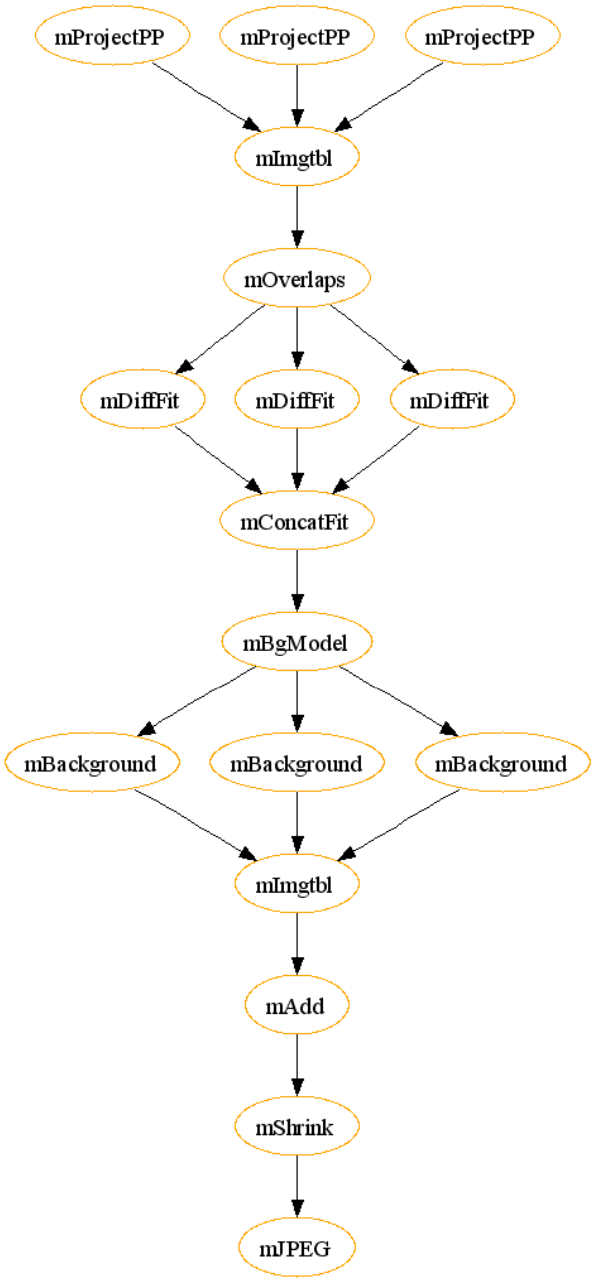
- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies



Completed Milestones: fMRI Application

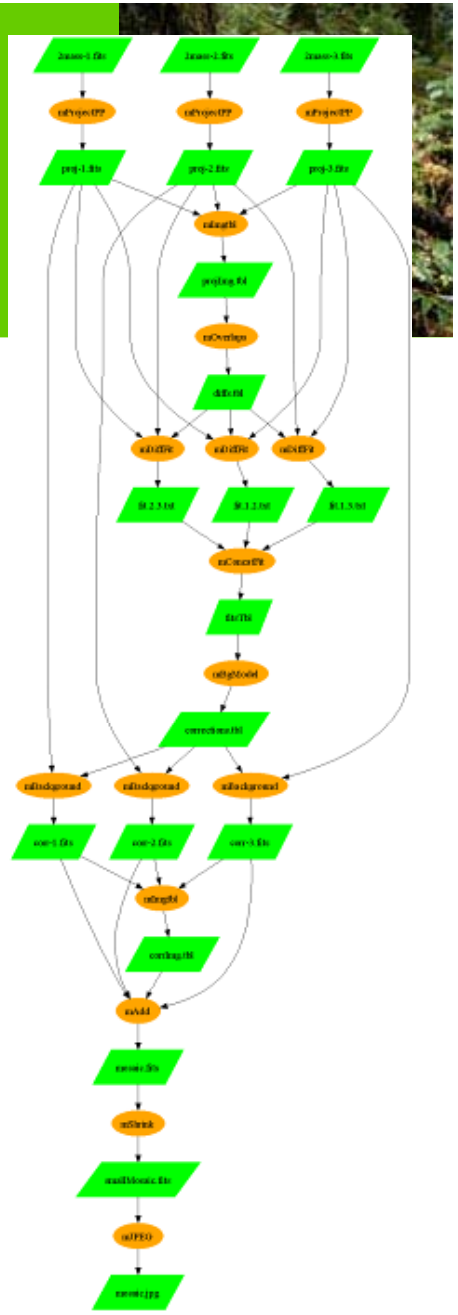
- GRAM vs. Falkon: 85%~90% lower run time
- GRAM/Clustering vs. Falkon: 40%~74% lower run time





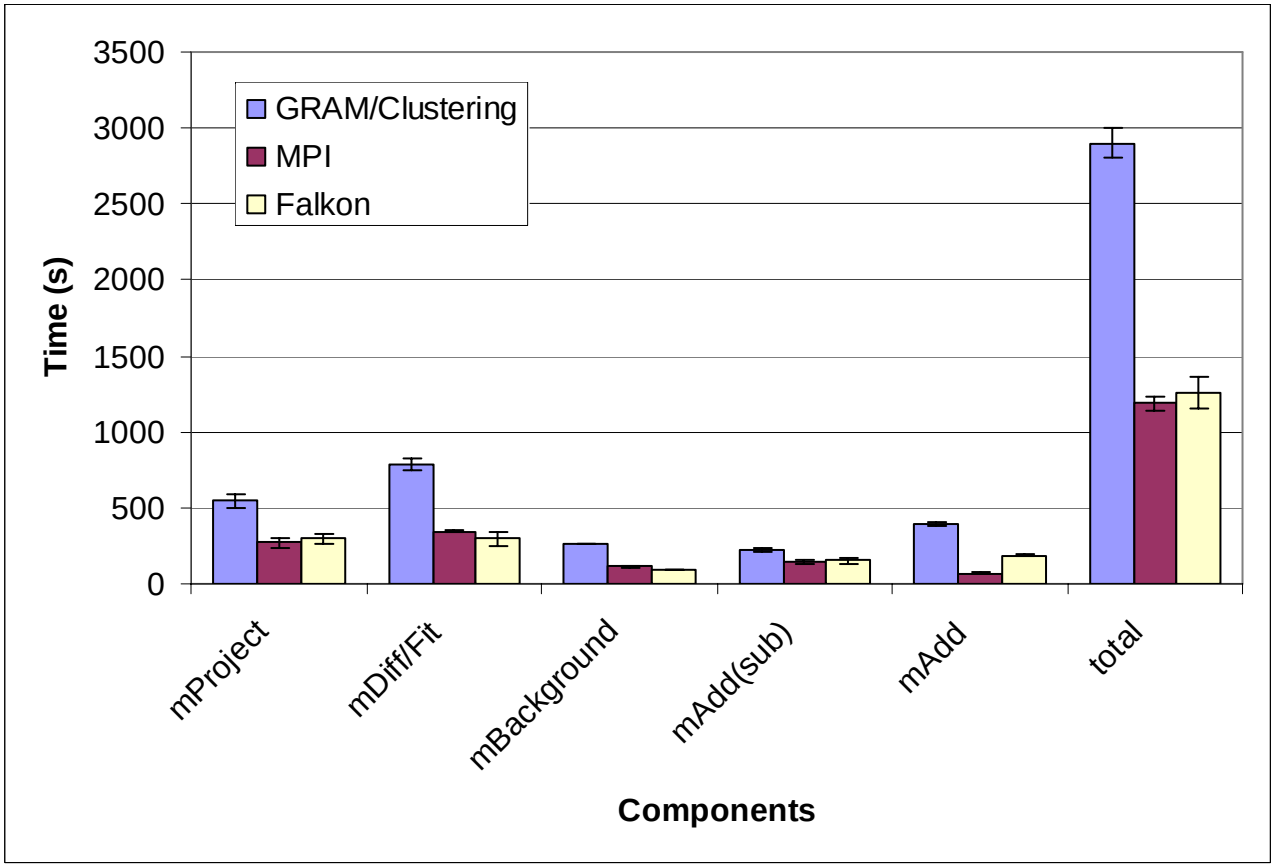
Montage

B. Berriman, J. Good (Caltech)
J. Jacob, D. Katz (JPL)



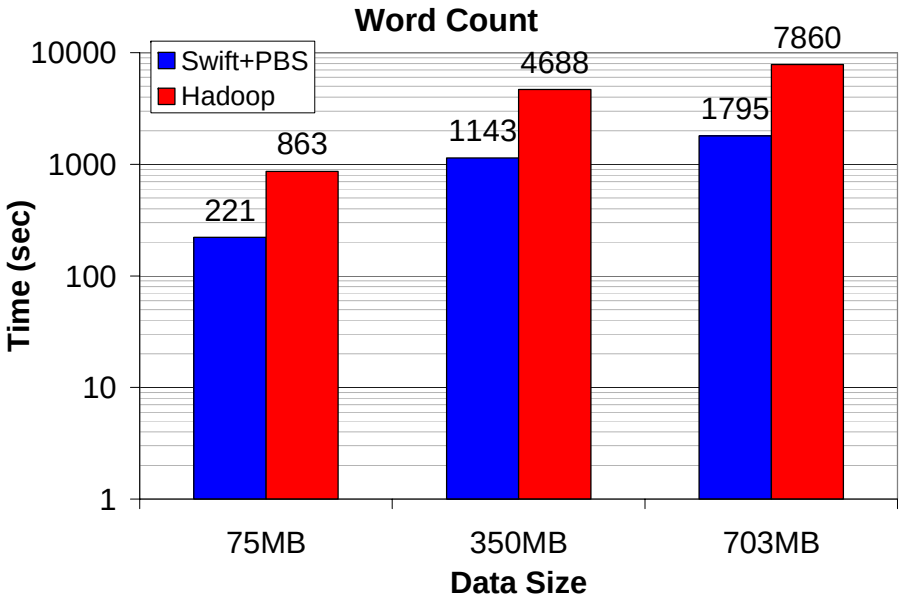
Completed Milestones: Montage Application

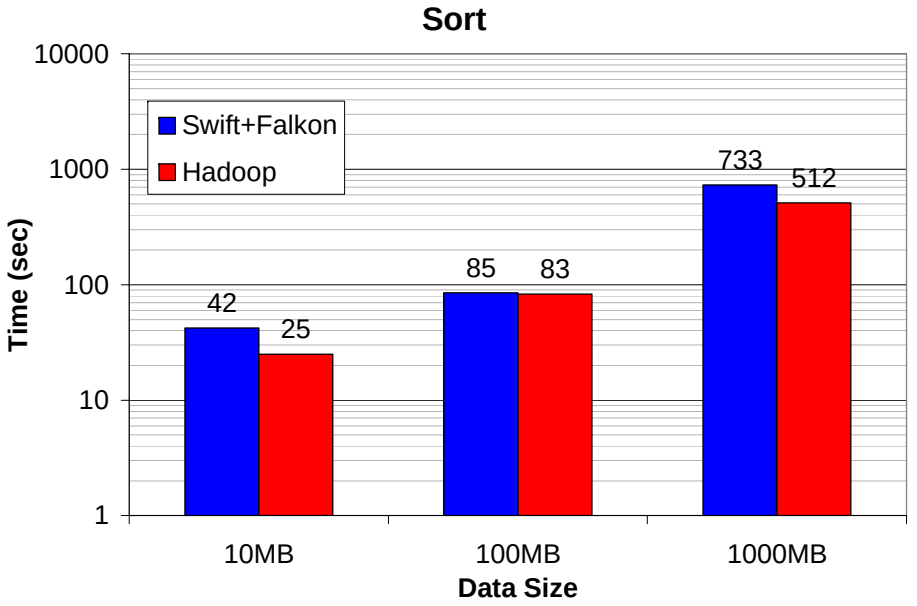
- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound



Hadoop vs. Swift

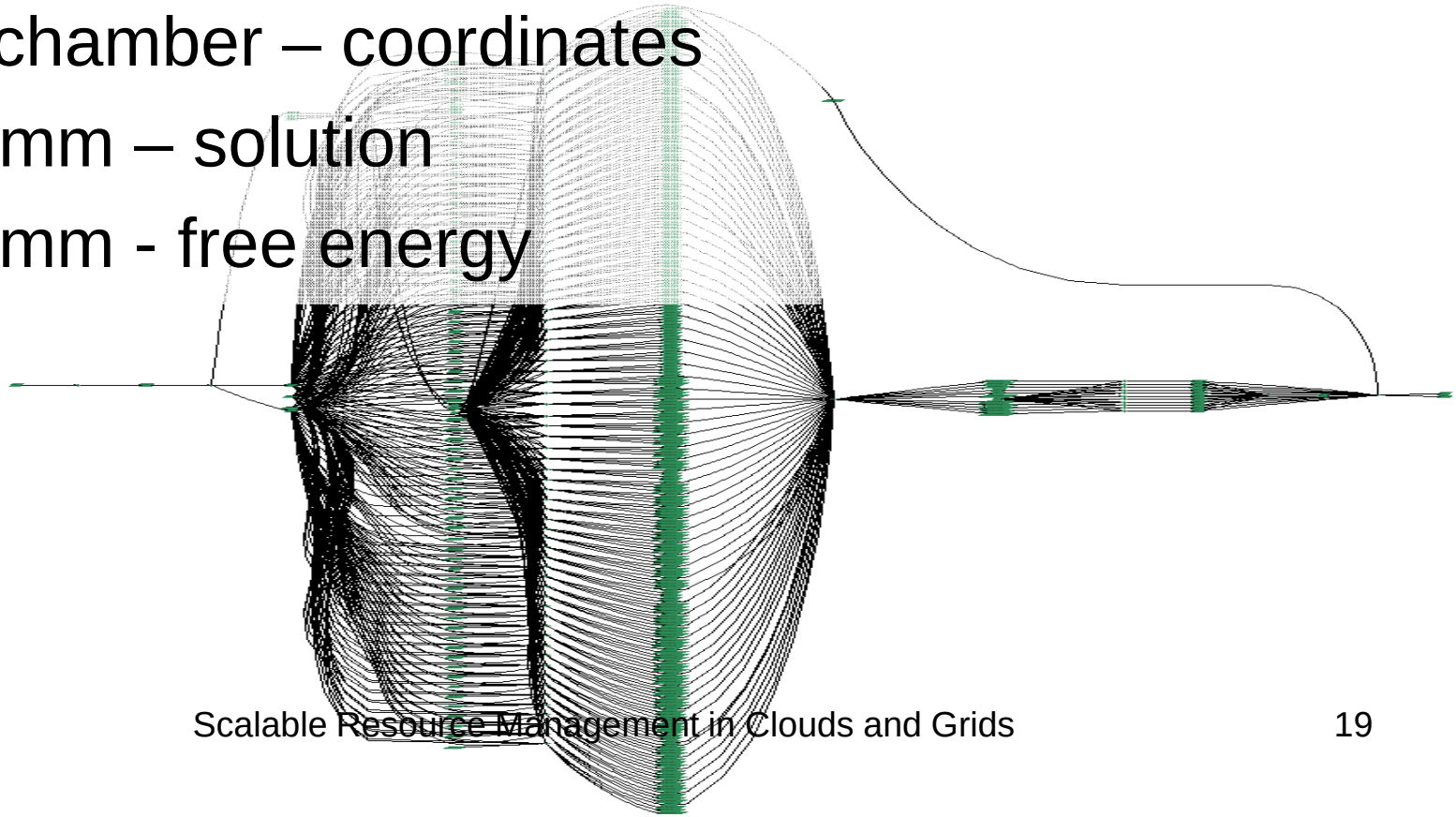
- Classic benchmarks for MapReduce
  - Word Count
  - Sort
- Swift performs similar or better than Hadoop (on 32 processors)





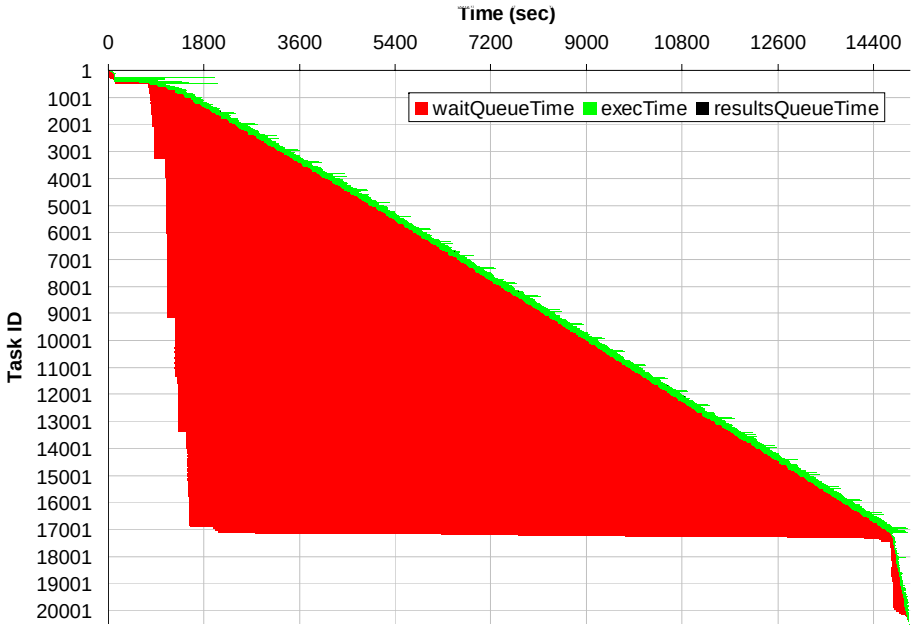
Molecular Dynamics

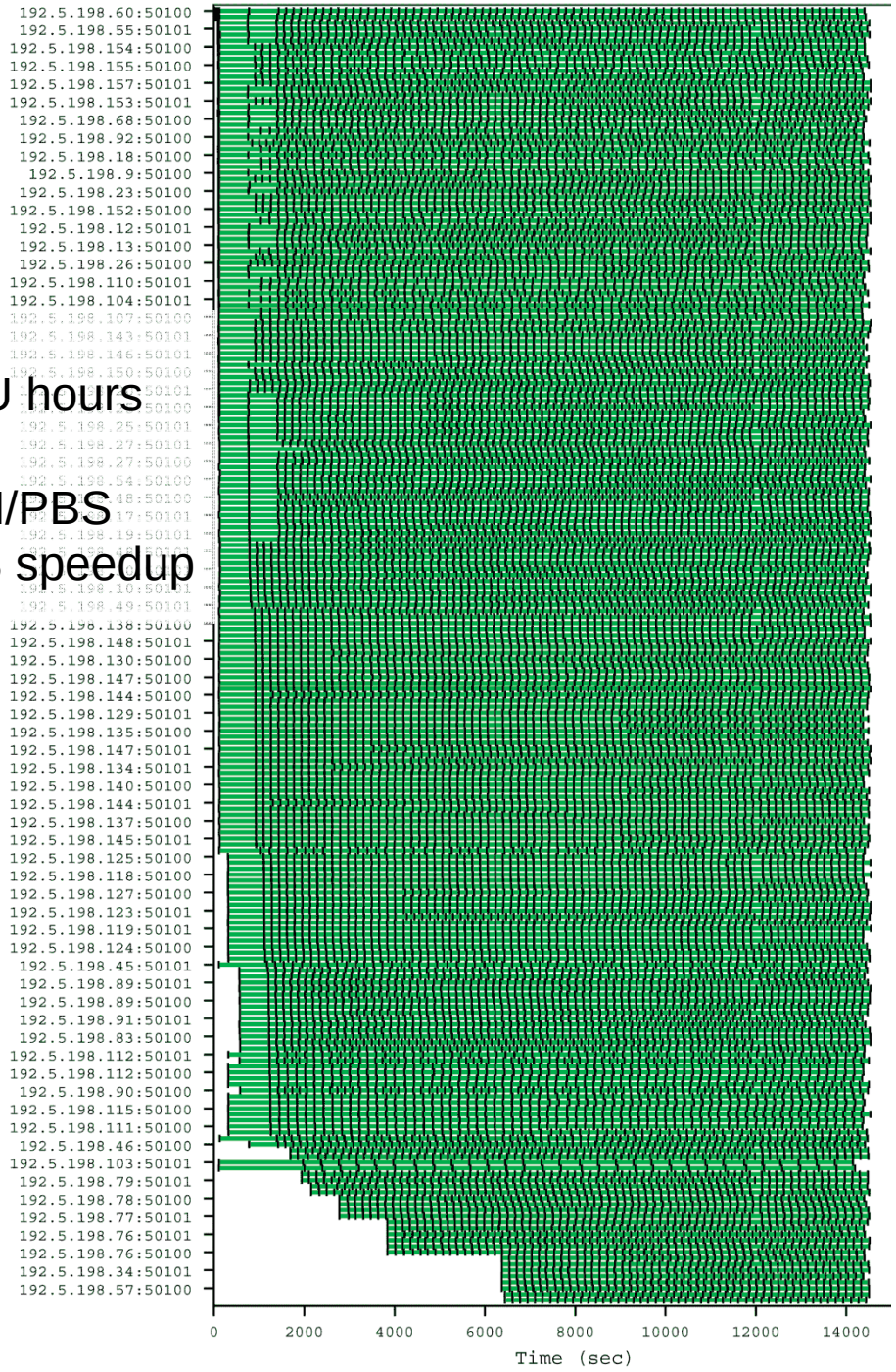
- Determination of free energies in aqueous solution
  - Antechamber – coordinates
  - Charmm – solution
  - Charmm - free energy



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



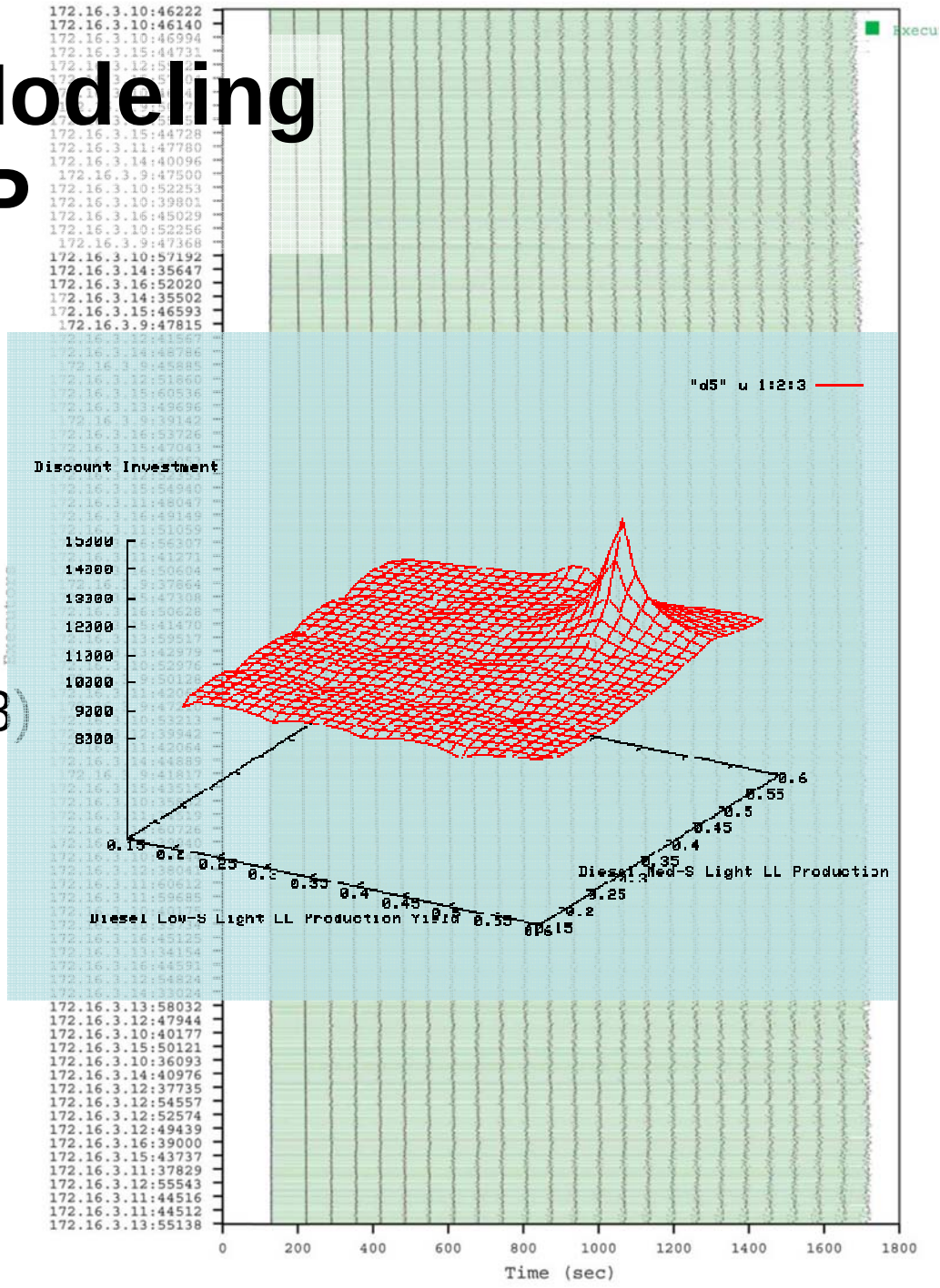


Scalable Resource Manage

MARS Economic Modeling on IBM BG/P

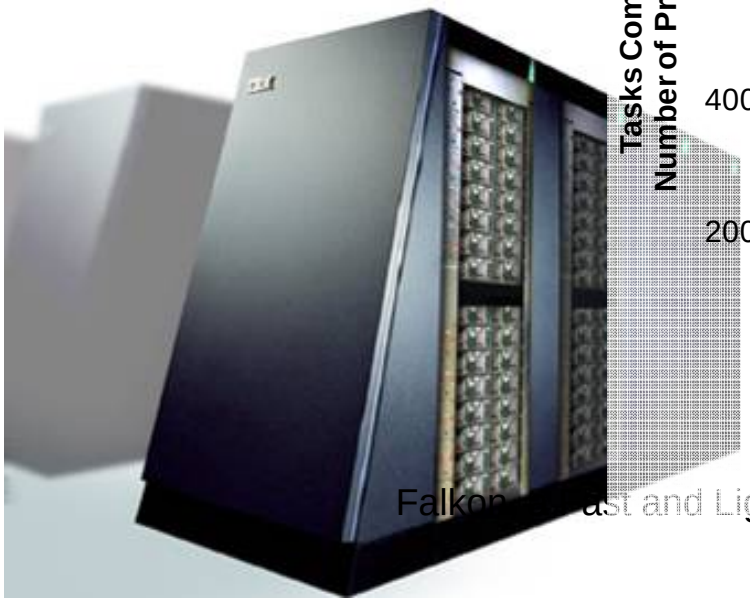
- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%

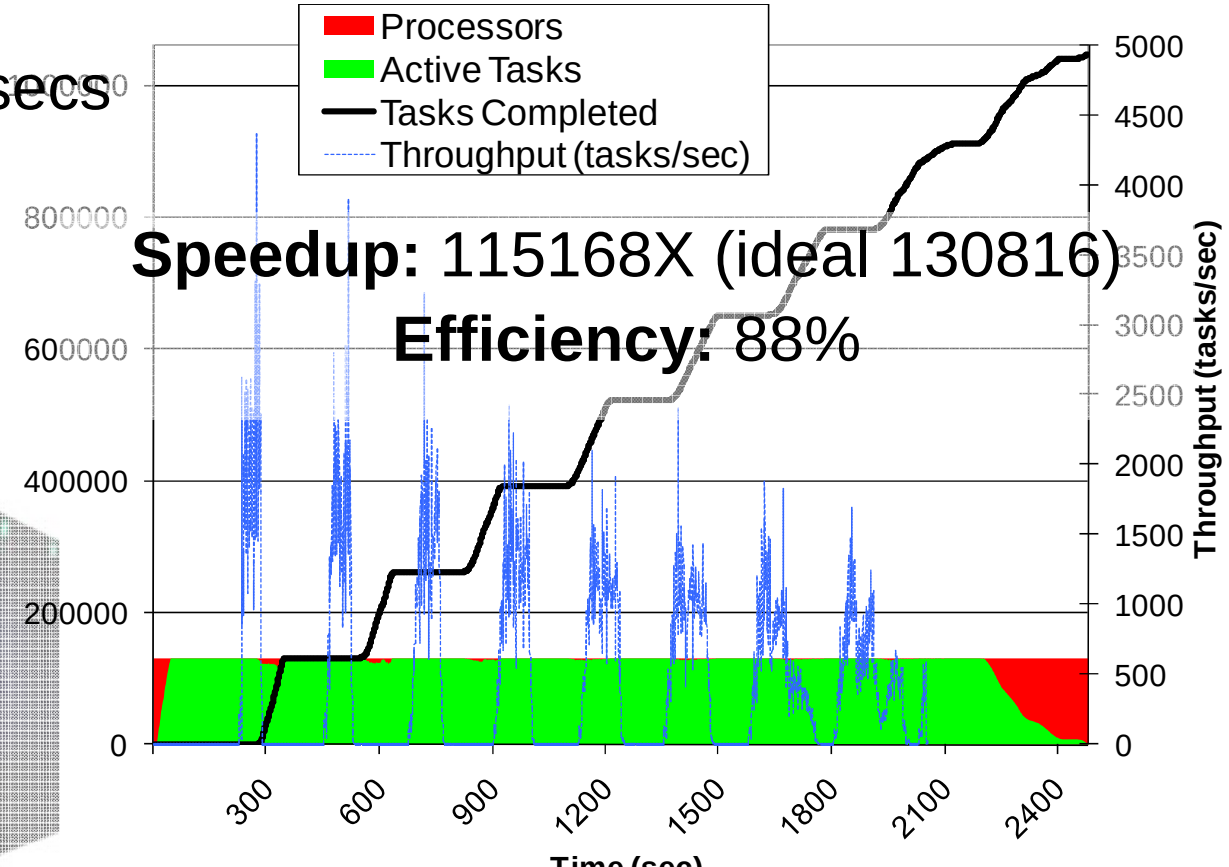
ght-weig
Grids, a

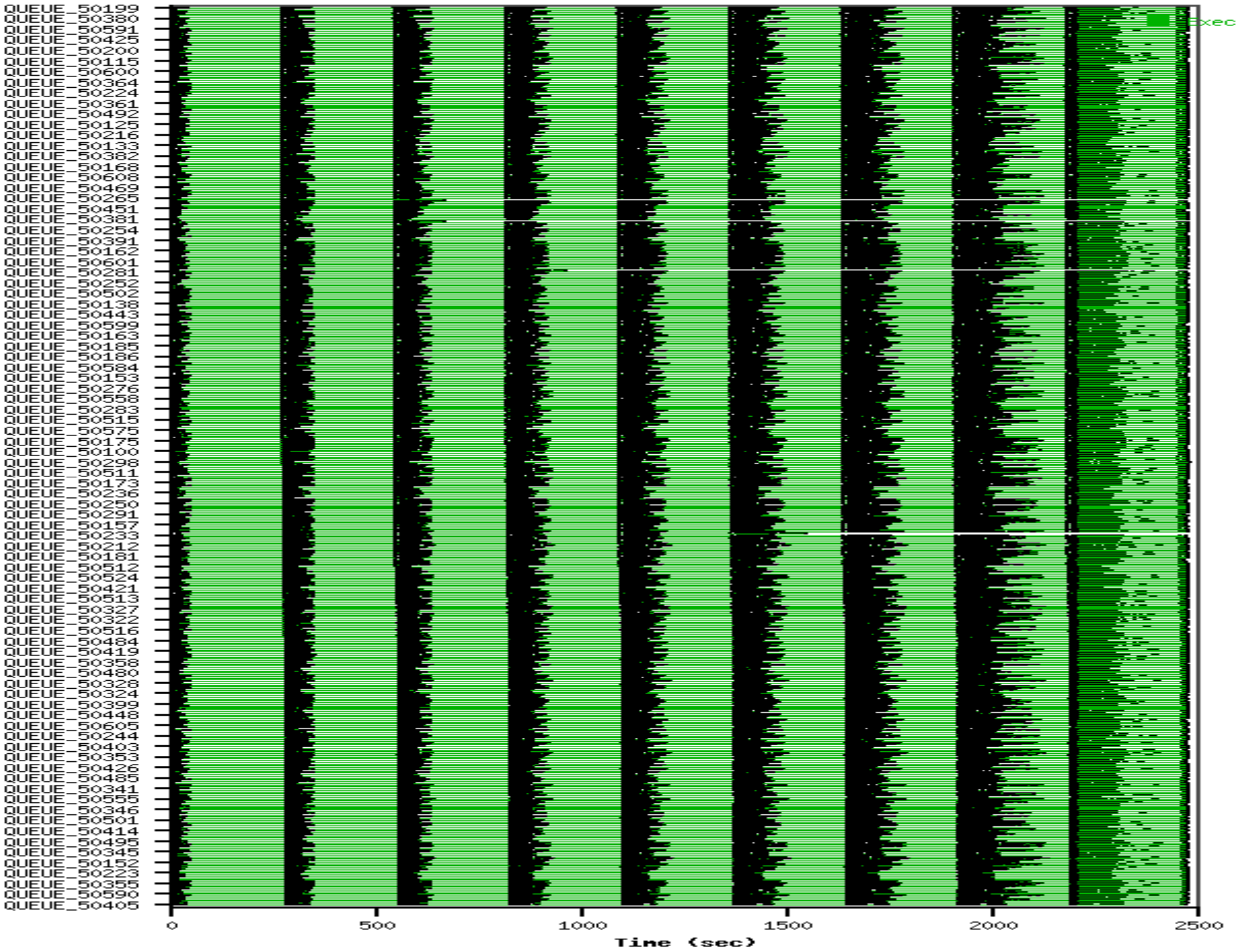


MARS Economic Modeling on IBM BG/P (128K CPUs)

- CPU Cores: 130816
- Tasks: 1048576
- Elapsed time: 2483 secs
- CPU Years: 9.3

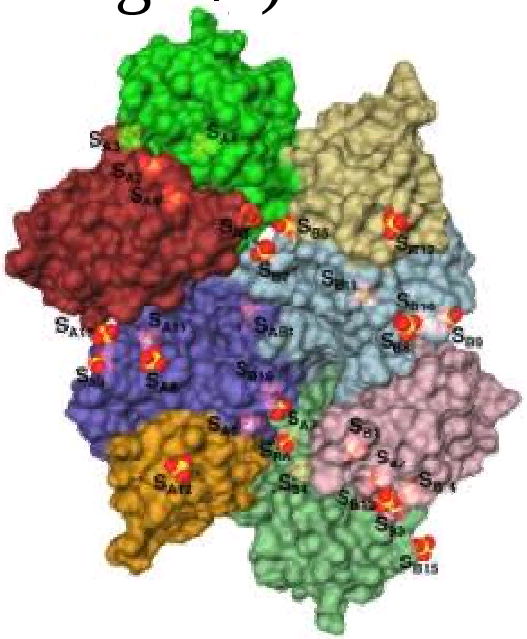


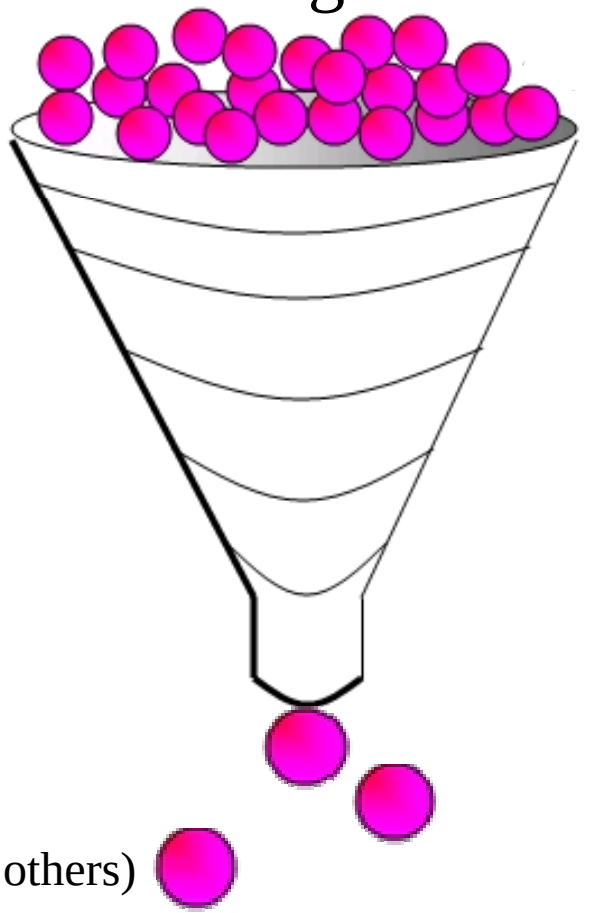



QUEUE_50199
QUEUE_50380
QUEUE_50591
QUEUE_50425
QUEUE_50200
QUEUE_50115
QUEUE_50600
QUEUE_50364
QUEUE_50224
QUEUE_50361
QUEUE_50492
QUEUE_50125
QUEUE_50216
QUEUE_50133
QUEUE_50382
QUEUE_50168
QUEUE_50608
QUEUE_50469
QUEUE_50265
QUEUE_50451
QUEUE_50381
QUEUE_50254
QUEUE_50391
QUEUE_50162
QUEUE_50601
QUEUE_50281
QUEUE_50252
QUEUE_50502
QUEUE_50138
QUEUE_50443
QUEUE_50599
QUEUE_50163
QUEUE_50185
QUEUE_50186
QUEUE_50584
QUEUE_50153
QUEUE_50276
QUEUE_50558
QUEUE_50283
QUEUE_50515
QUEUE_50575
QUEUE_50175
QUEUE_50100
QUEUE_50298
QUEUE_50511
QUEUE_50173
QUEUE_50236
QUEUE_50250
QUEUE_50291
QUEUE_50157
QUEUE_50233
QUEUE_50212
QUEUE_50181
QUEUE_50512
QUEUE_50524
QUEUE_50421
QUEUE_50513
QUEUE_50327
QUEUE_50322
QUEUE_50516
QUEUE_50484
QUEUE_50419
QUEUE_50358
QUEUE_50480
QUEUE_50328
QUEUE_50324
QUEUE_50399
QUEUE_50448
QUEUE_50605
QUEUE_50244
QUEUE_50403
QUEUE_50353
QUEUE_50426
QUEUE_50485
QUEUE_50341
QUEUE_50555
QUEUE_50346
QUEUE_50501
QUEUE_50414
QUEUE_50495
QUEUE_50345
QUEUE_50152
QUEUE_50223
QUEUE_50355
QUEUE_50590
QUEUE_50405
Exec
0
500
1000
1500
2000
2500
Time (sec)

Many Many Tasks: Identifying Potential Drug Targets

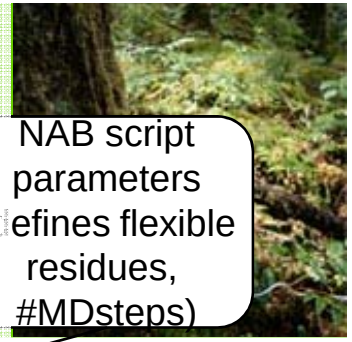
Protein target(s) x 2M+ ligands

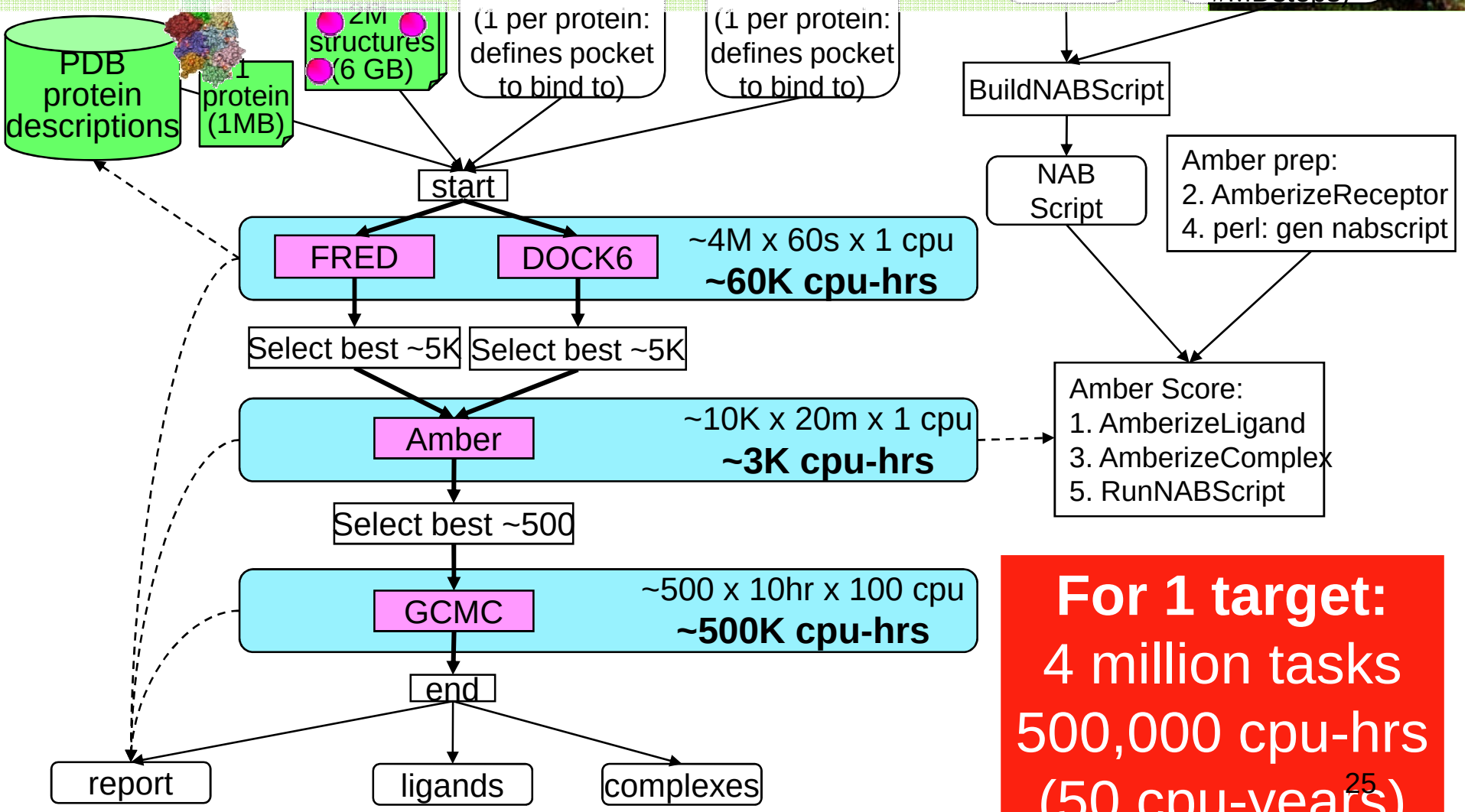




(Mike Kubal, Benoit Roux, and others)

Many Many Tasks: Identifying Potential Drug Targets

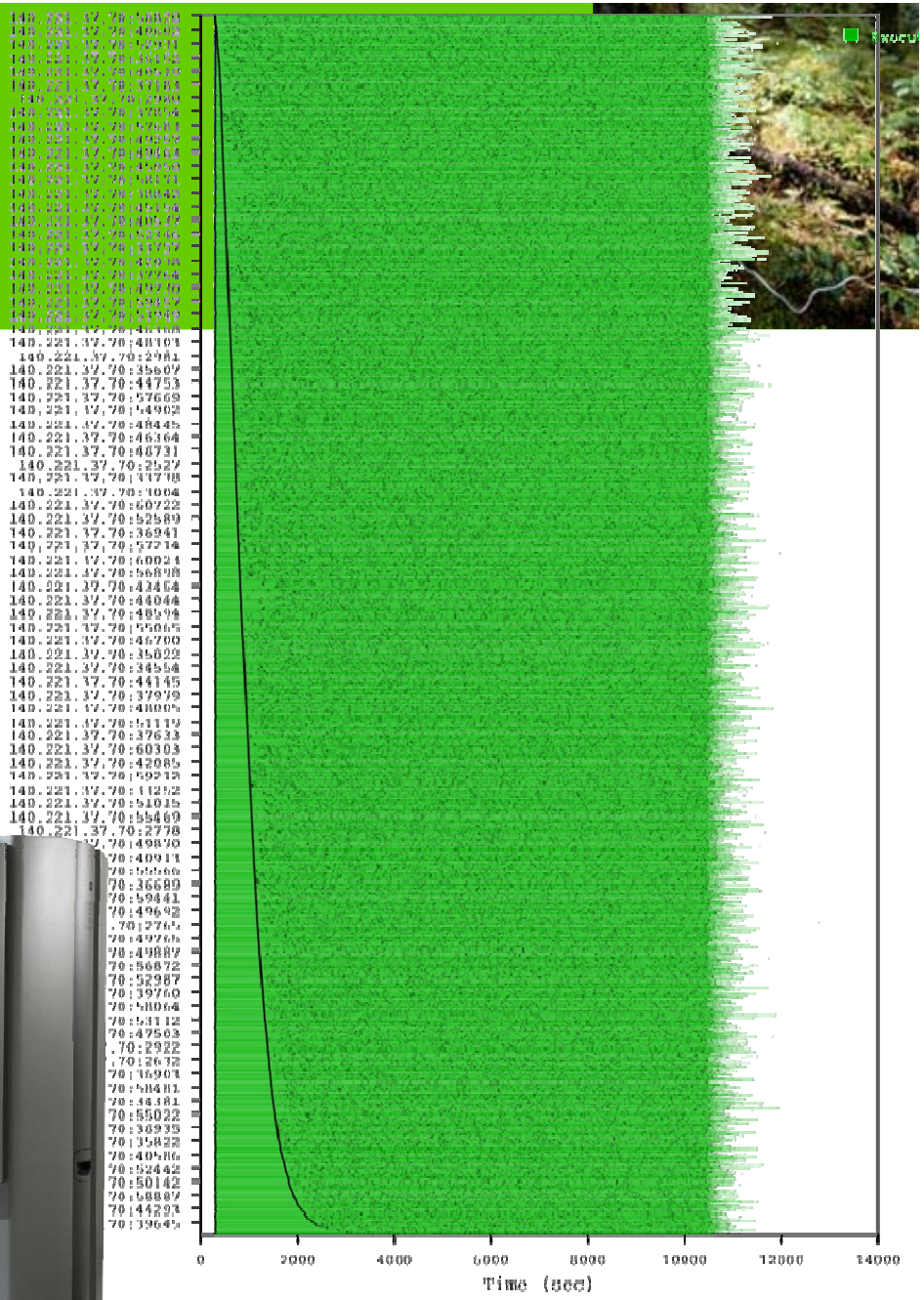




For 1 target:
4 million tasks
500,000 cpu-hrs
(50 cpu-years)

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

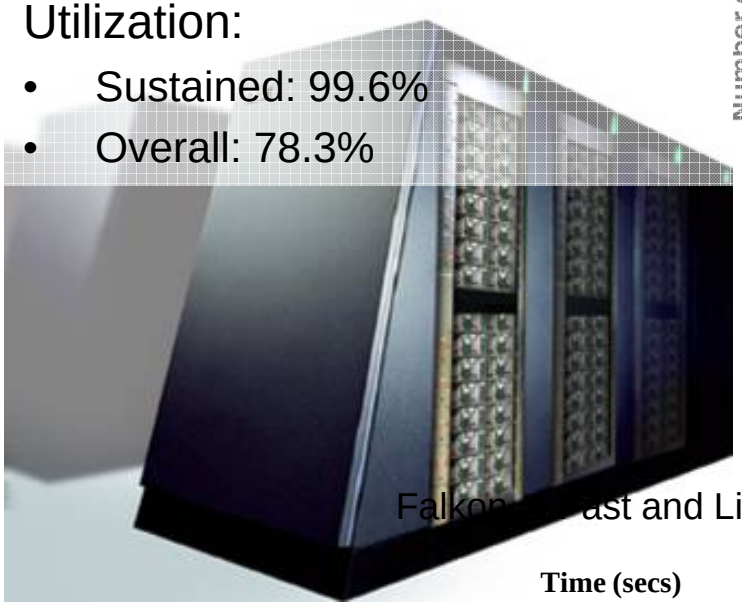
Average task time: 667 sec

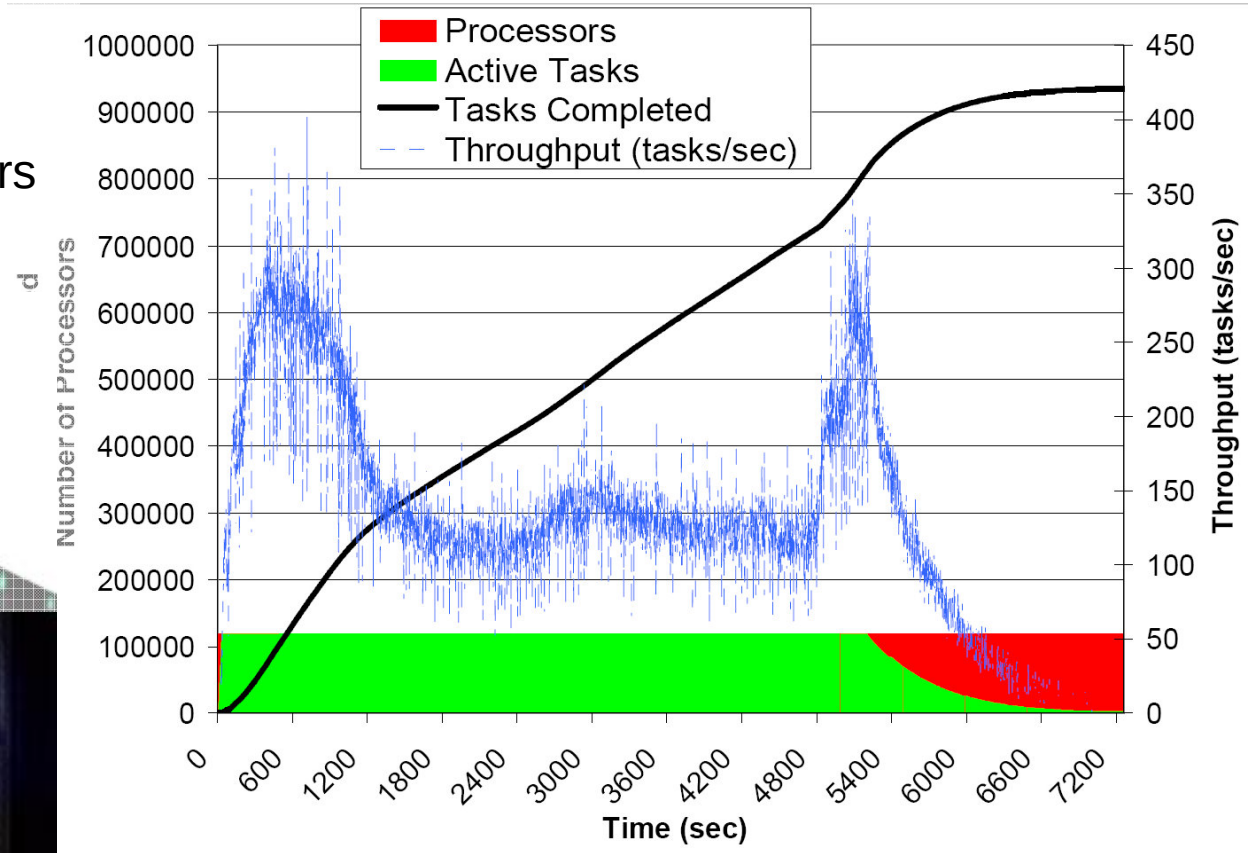
Relative Efficiency: 99.7%

(from 16 to 32 racks)

Utilization:

- Sustained: 99.6%
- Overall: 78.3%

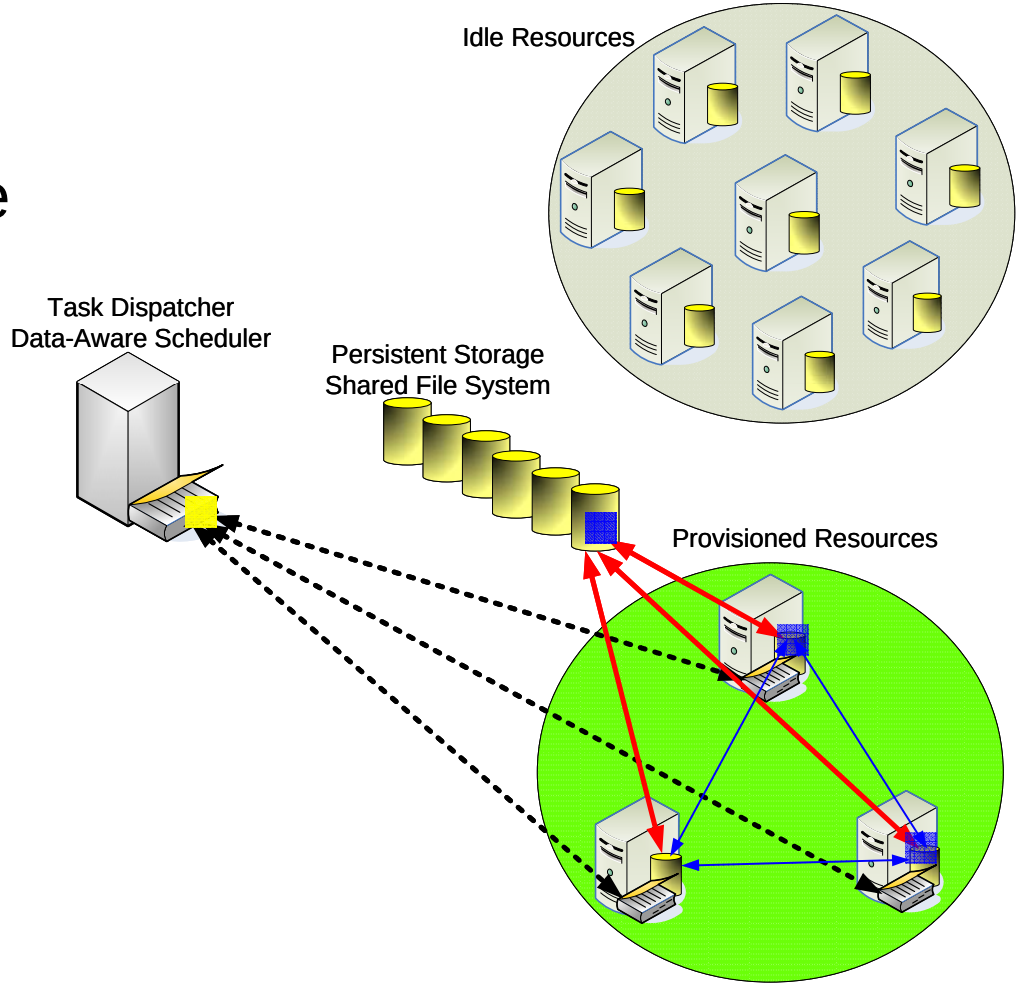




Time (secs)

Data Diffusion

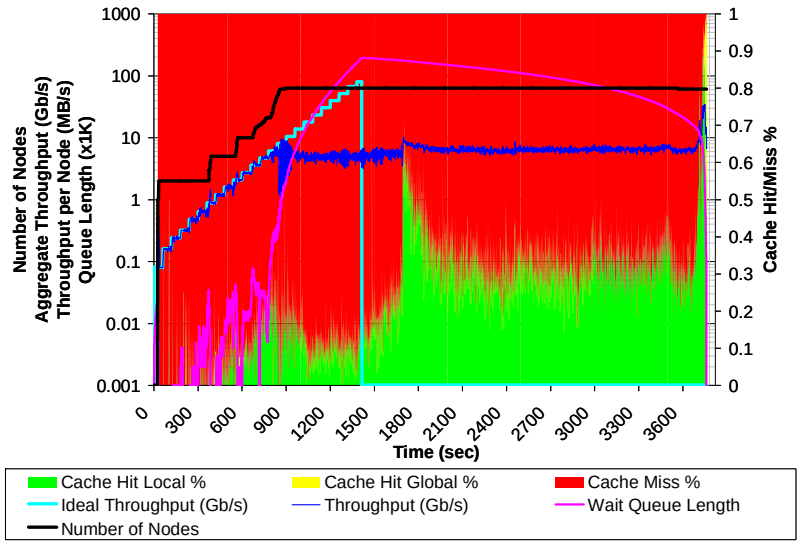
- Resource acquired in response to demand
- Data and applications diffuse from archival storage to newly acquired resources
- Resource “caching” allows faster responses to subsequent requests
  - Cache Eviction Strategies: RANDOM, FIFO, LRU, LFU
- Resources are released when demand drops



Data Diffusion

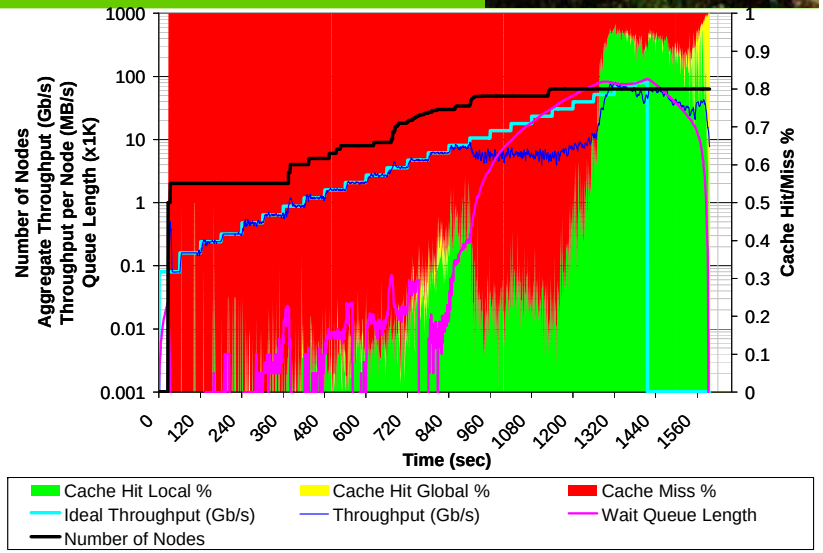
- Considers both data and computations to optimize performance
  - Supports data-aware scheduling
  - Can optimize compute utilization, cache hit performance, or a mixture of the two
- Decrease dependency of a shared file system
  - Theoretical linear scalability with compute resources
  - Significantly increases meta-data creation and/or modification performance
- Central for “data-centric task farm” realization

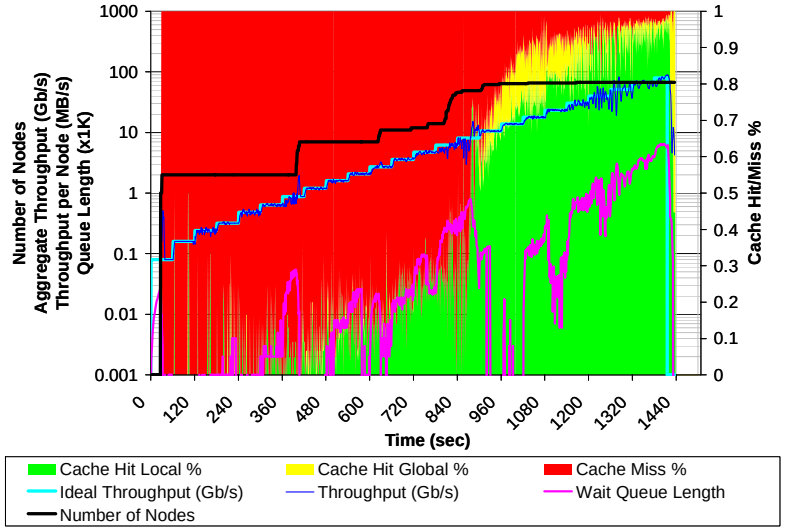
Data Diffusion: Good-cache-compute



← 1GB

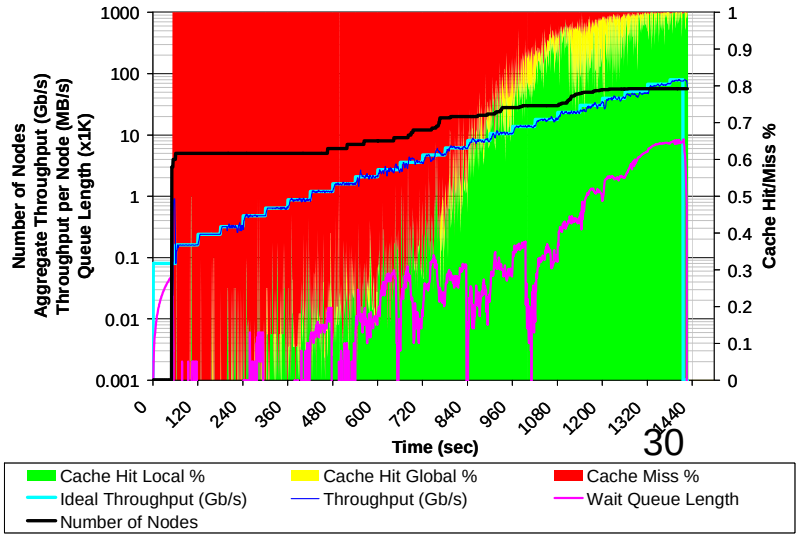
1.5GB →



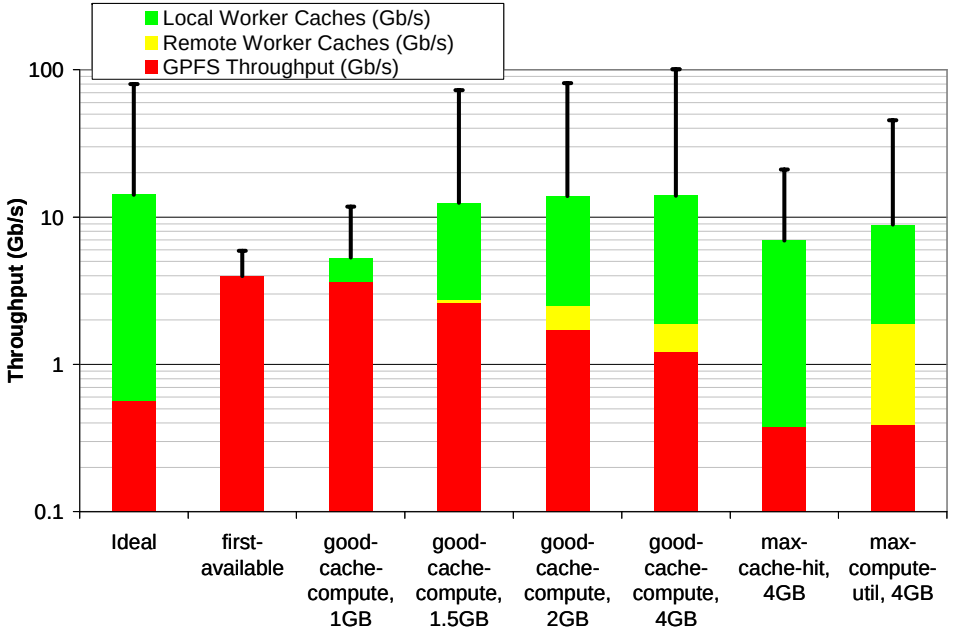


← 2GB

4GB →



Data Diffusion: Throughput and Response Time

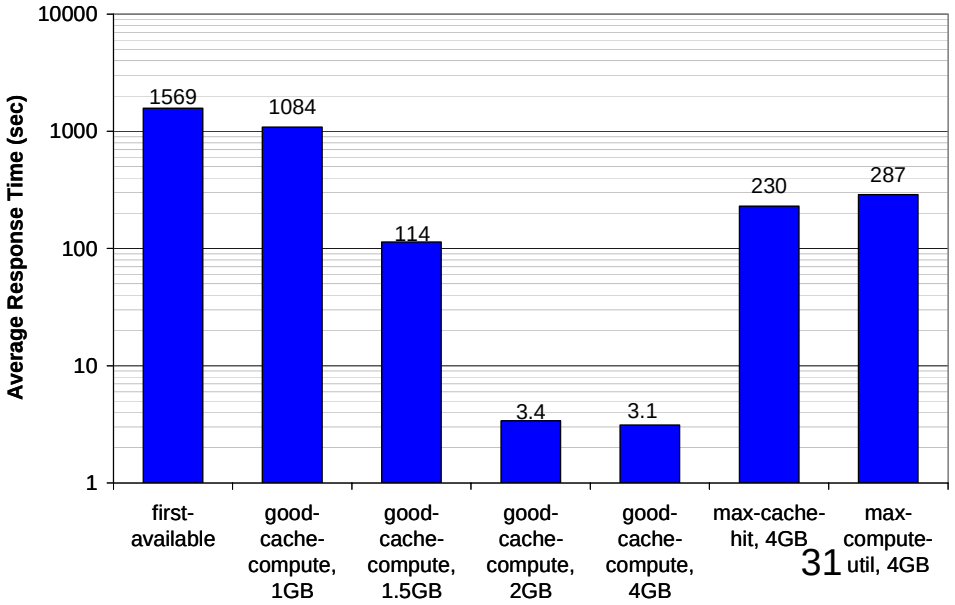


⬅Throughput:

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

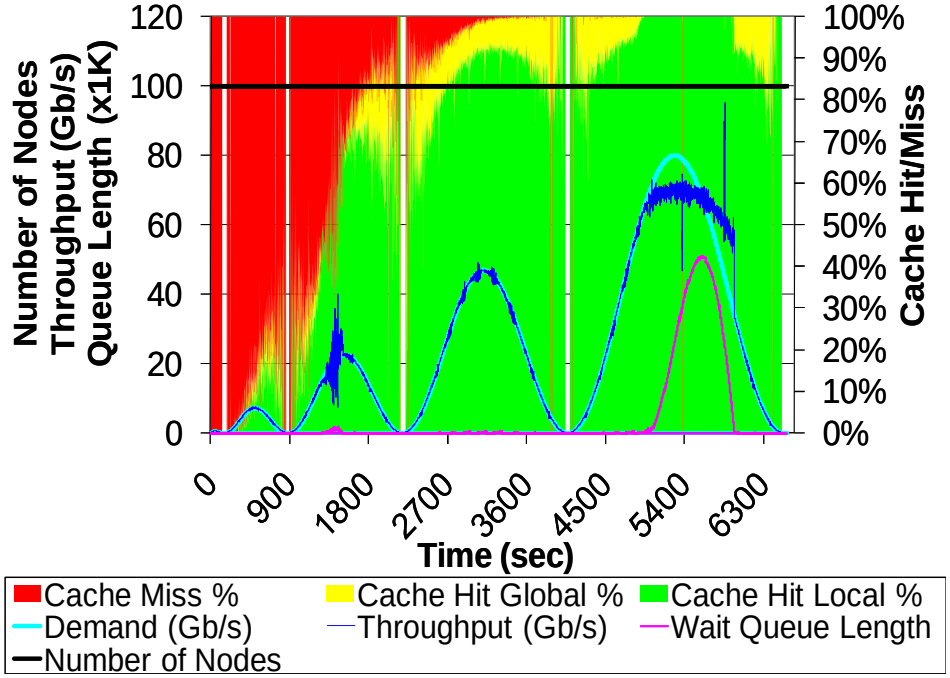
Response Time ➔

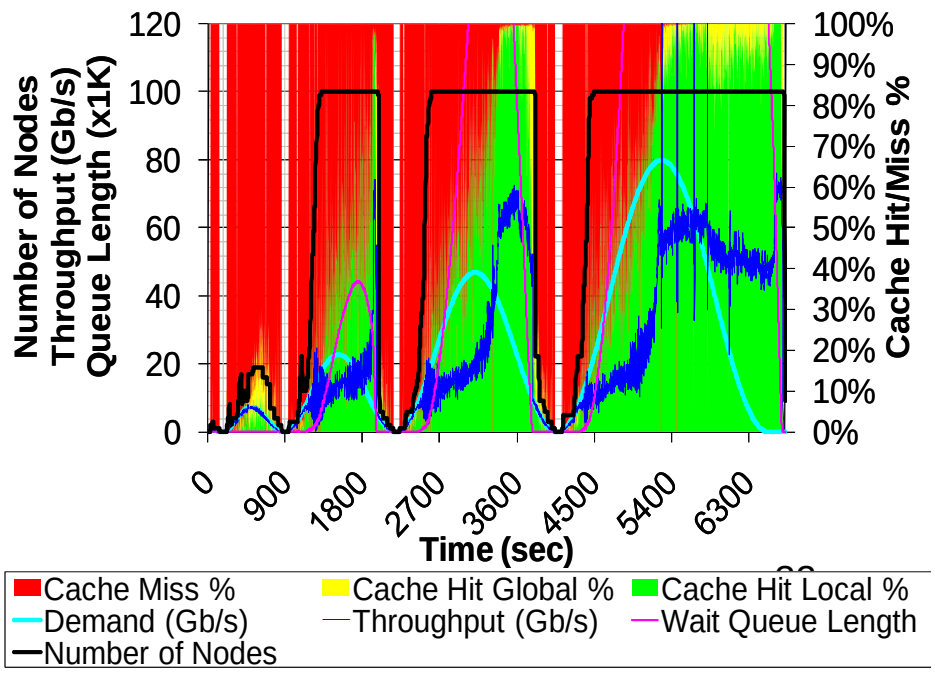
- 3 sec vs 1569 sec ➔ 506X



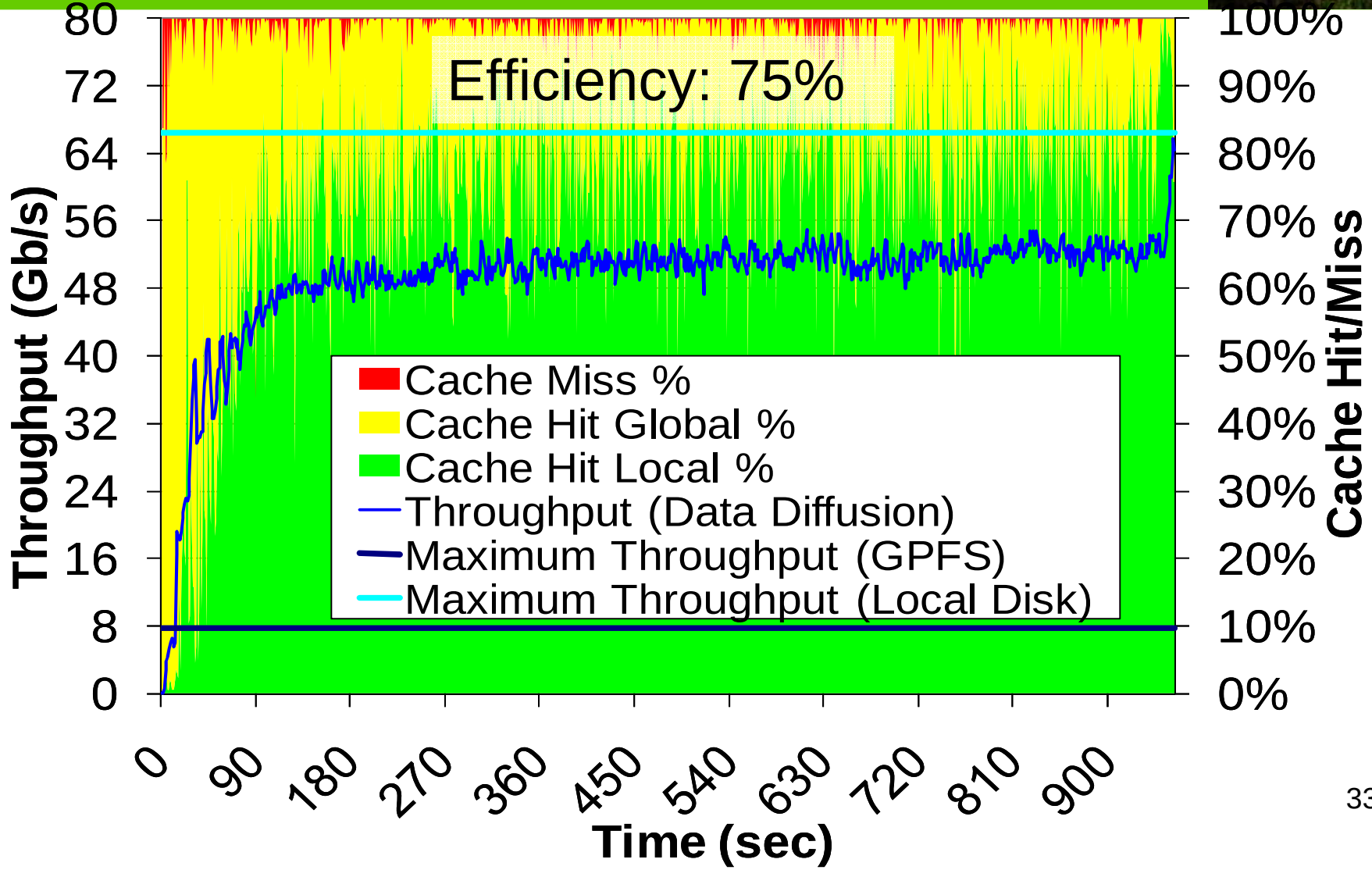
Sin-Wave Workload

- GPFS ➔ 5.7 hrs, ~8Gb/s, 1138 CPU hrs
- DF+SRP ➔ 1.8 hrs, ~25Gb/s, 361 CPU hrs
- DF+DRP ➔ 1.86 hrs, ~24Gb/s, 253 CPU hrs





All-Pairs Workload 500x500 on 200 CPUs



Limitations of Data Diffusion

- Needs Java 1.4+
- Needs IP connectivity between hosts
- Needs local storage (disk, memory, etc)
- Per task workings set must fit in local storage
- Task definition must include input/output files metadata
- Data access patterns: write once, read many

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

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
- 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