Managing and Executing Loosely-Coupled Large-Scale Applications on Clusters, Grids, and Supercomputers

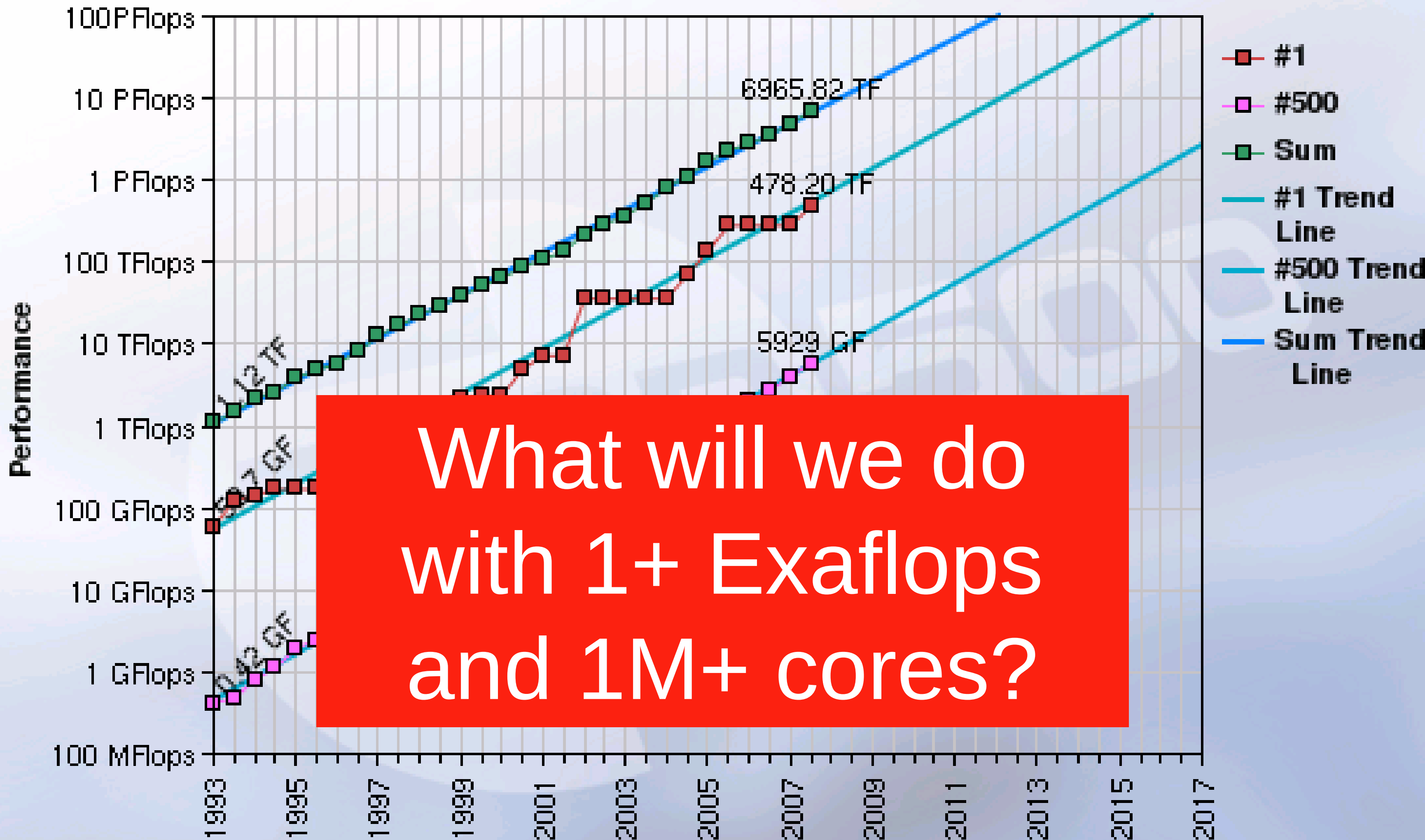
Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

Collaborators:
Ian Foster (UC/CI/ANL), Yong Zhao (MS), Mike Wilde (CI/ANL), Zhao Zhang (CI), Alex Szalay (JHU), Jerry Yan (NASA/ARC), Catalin Dumitrescu (FANL), many others from Swift and Falkon teams

GlobusWorld08
May 14th, 2008

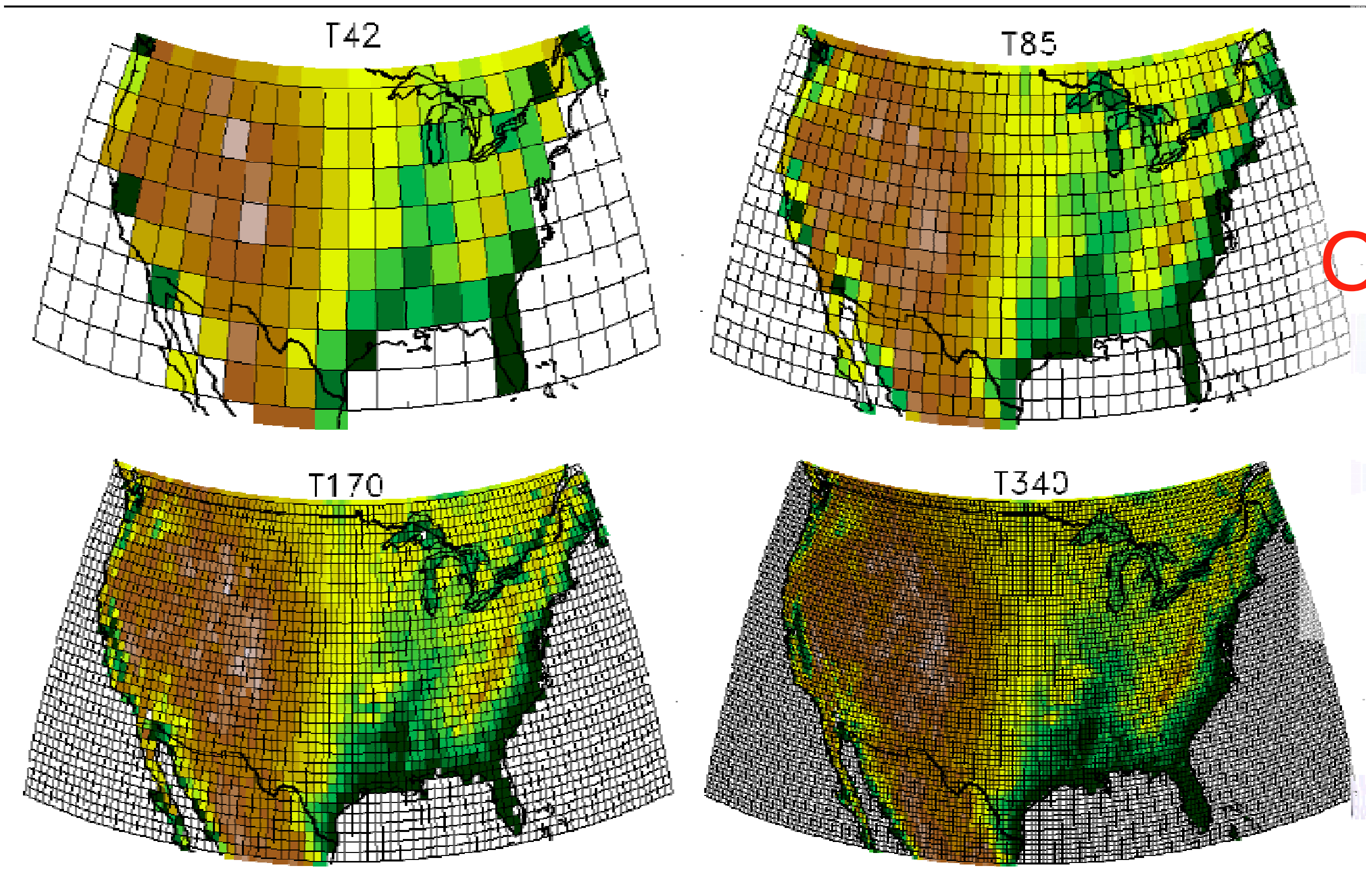
TOP500 SUPERCOMPUTER SITES

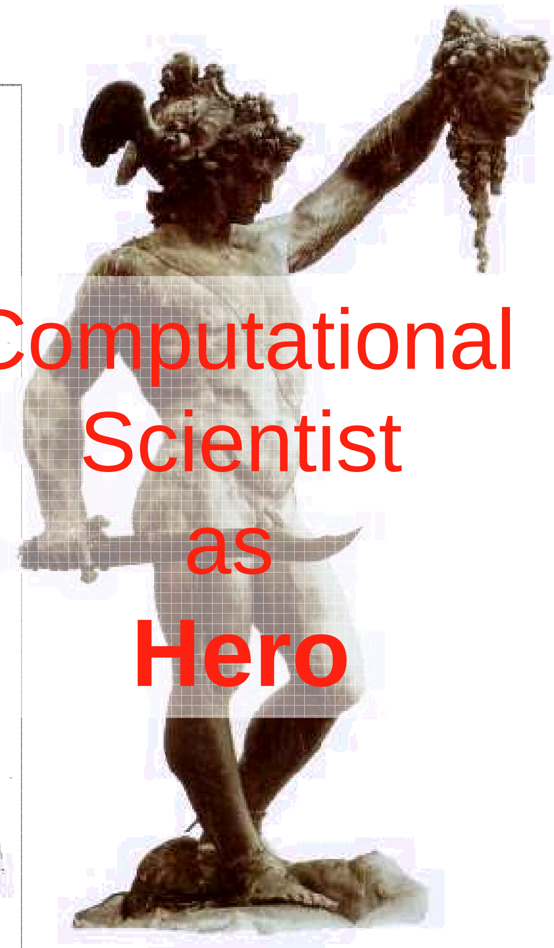
Projected Performance Development



What will we do with 1+ Exaflops and 1M+ cores?

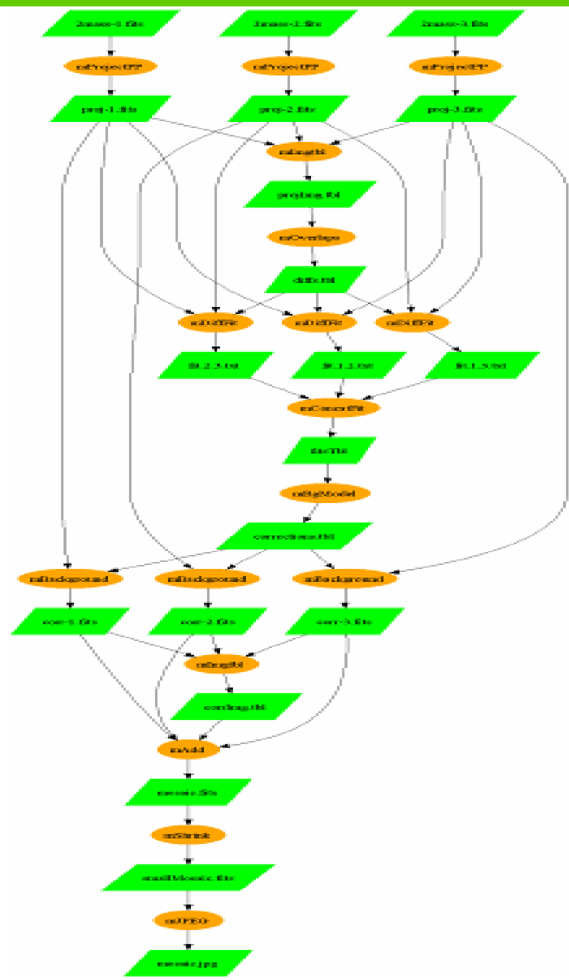
1) Tackle Bigger and Bigger Problems





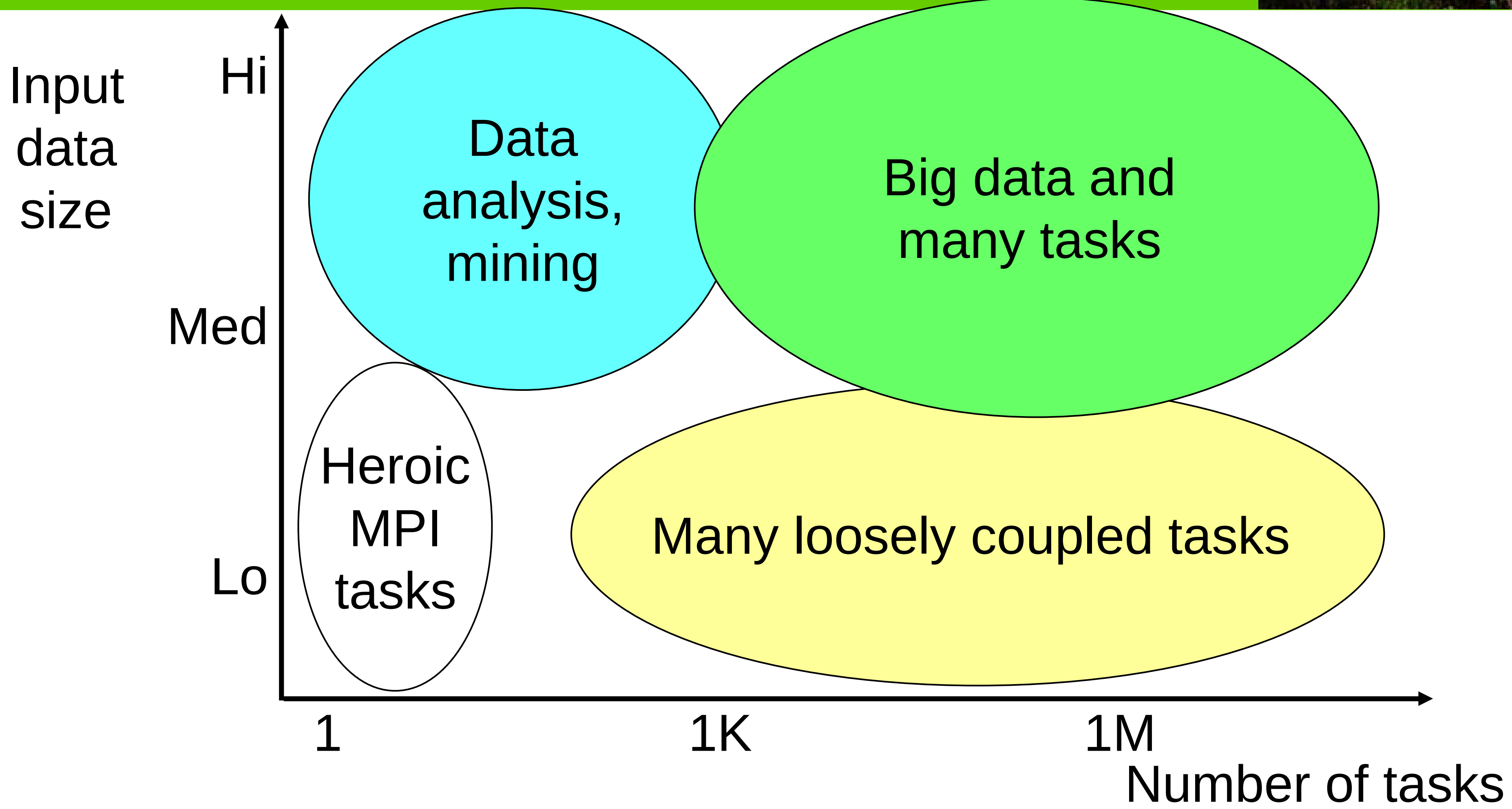
Computational
Scientist
as
Hero

2) Tackle Increasingly Complex Problems



Computational Scientist as **Logistics Officer**

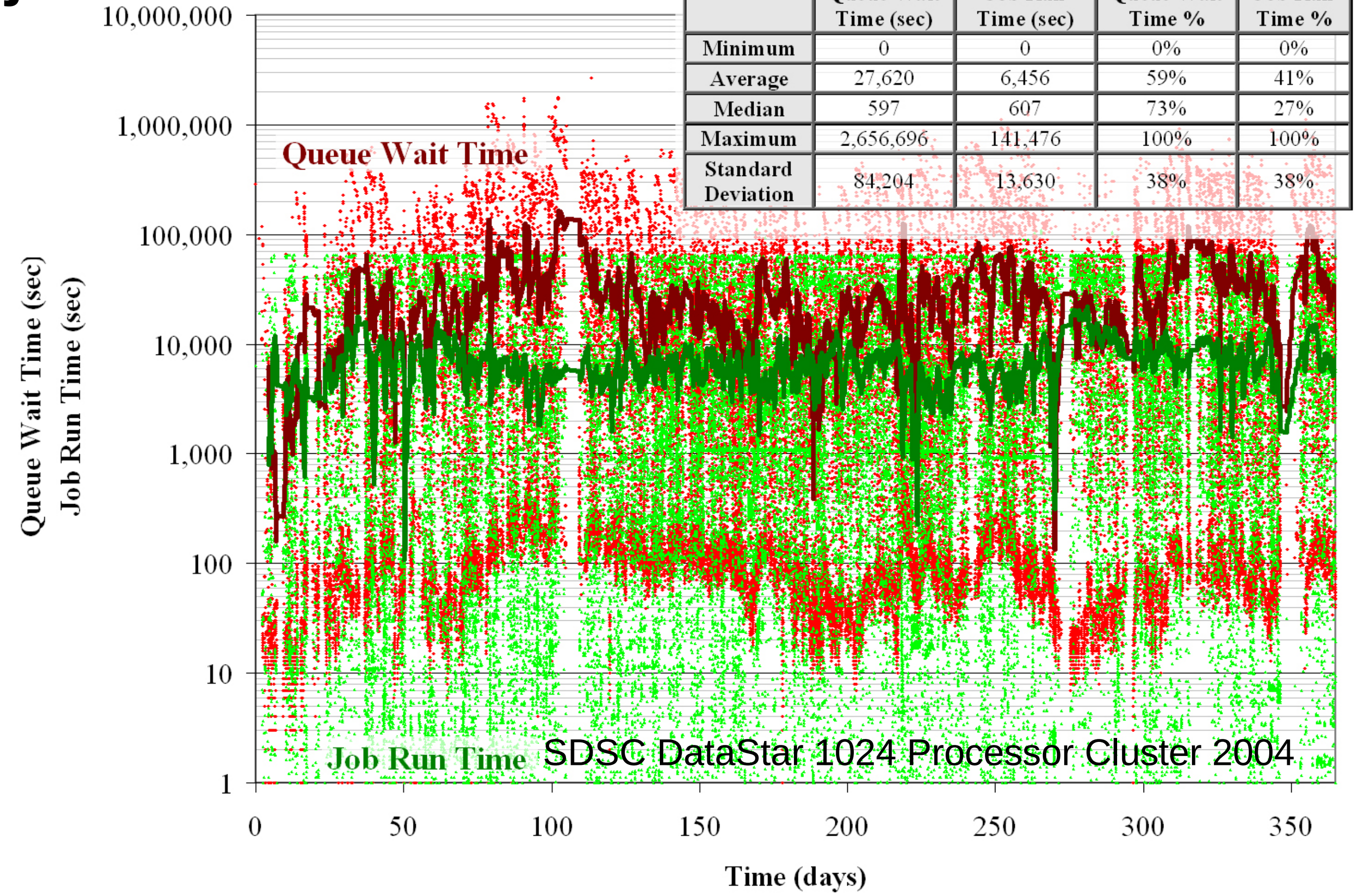
Problem Types



Challenge #1: Long Queue Times

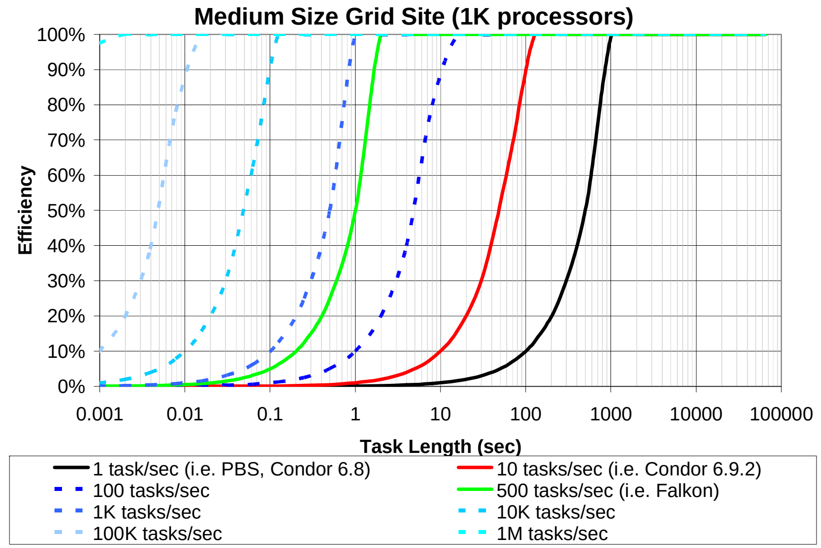
- Wait queue times are typically longer than the job duration times

	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%



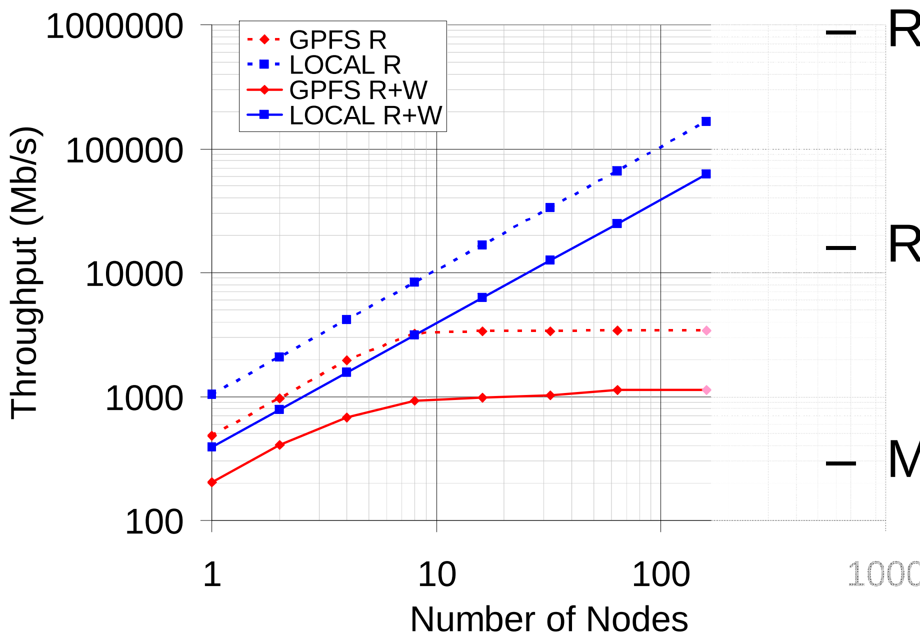
Challenge #2: Slow Job Dispatch Rates

- 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**

Hypothesis

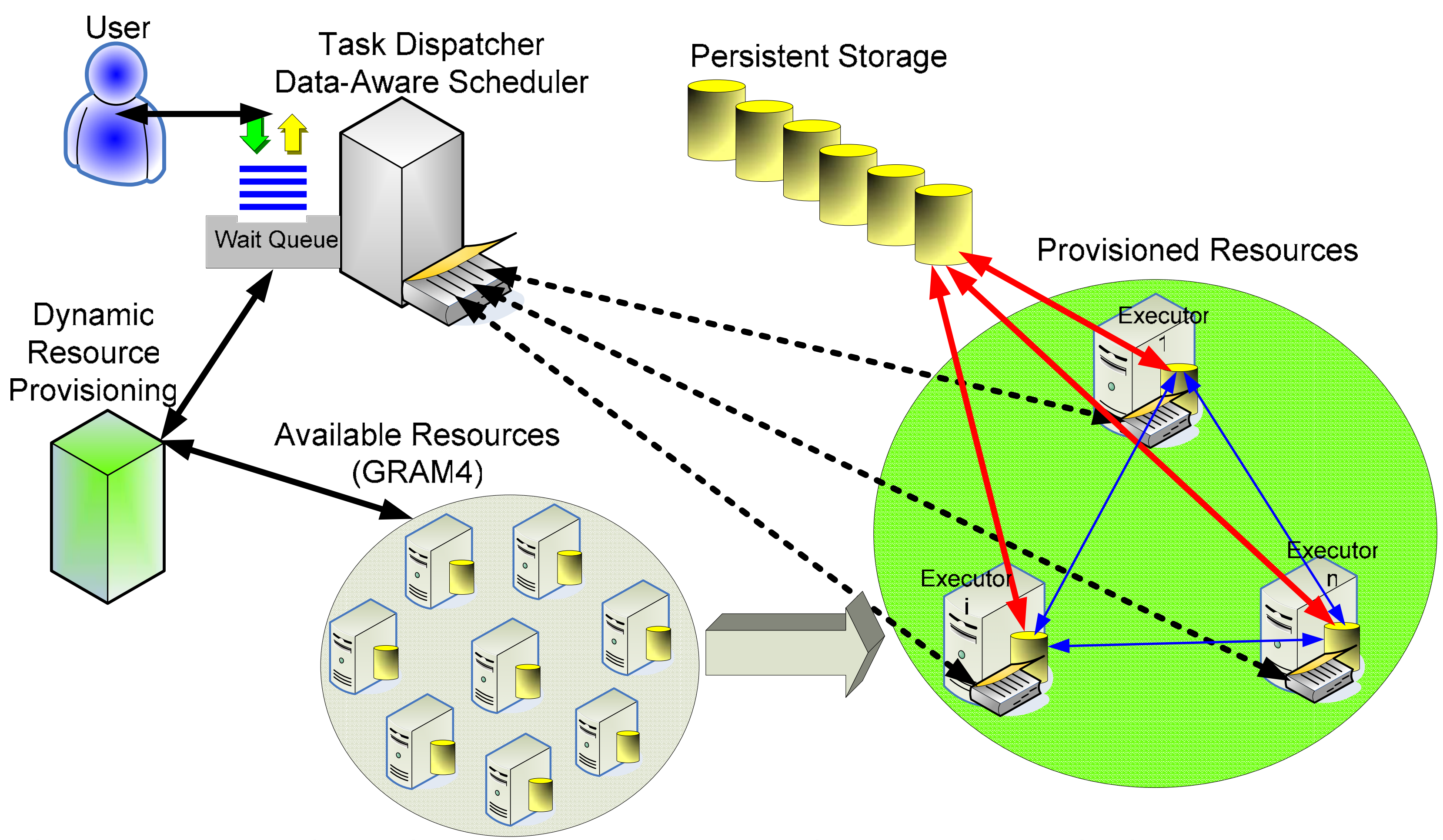
"Significant performance improvements can be obtained in the analysis of large dataset by leveraging information about data analysis workloads rather than individual data analysis tasks."

- **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

Falkon: a Fast and Light-weight tasK executiON framework

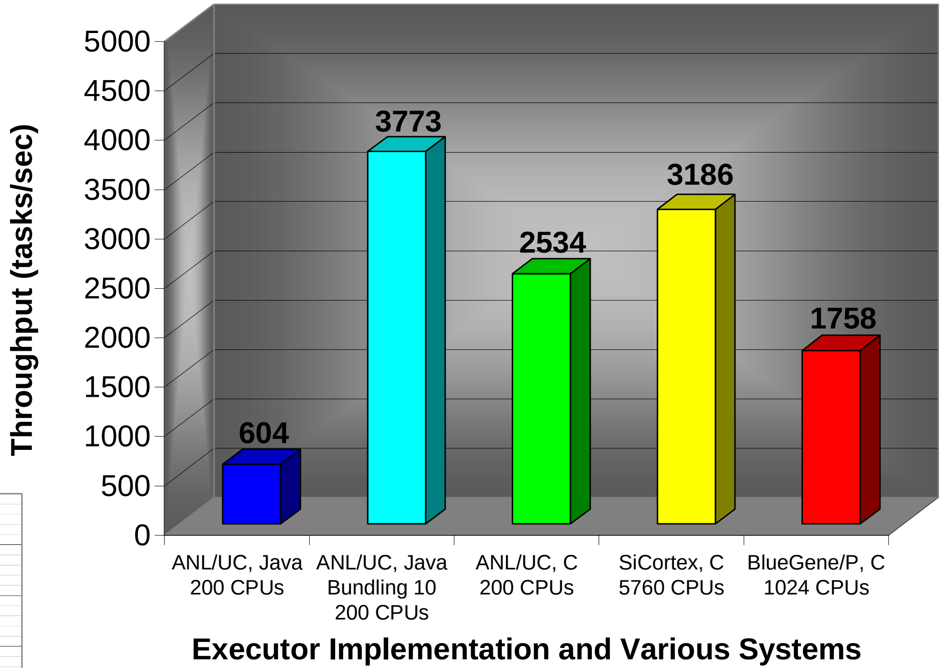
- ***Goal:*** enable the ***rapid and efficient*** execution of many independent jobs on large compute clusters
- Combines three components:
  - a ***streamlined task dispatcher*** able to achieve order-of-magnitude higher task dispatch rates than conventional schedulers ➔ Challenge #1
  - ***resource 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

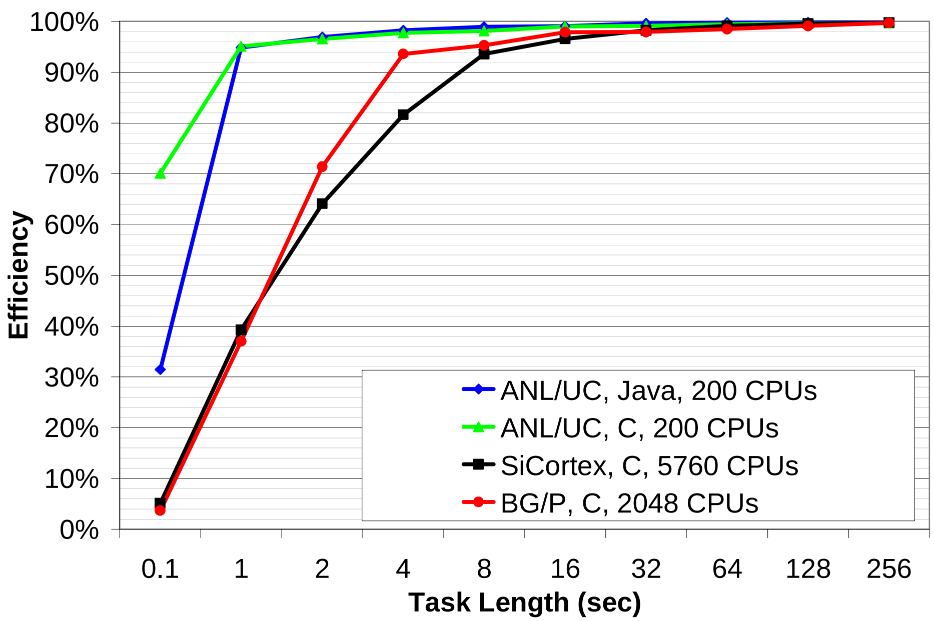
Falkon Overview



Dispatcher Throughput

- Fast:
  - Up to 3700 tasks/sec
- Scalable:
  - 54,000 processors
  - 1,500,000 tasks queued



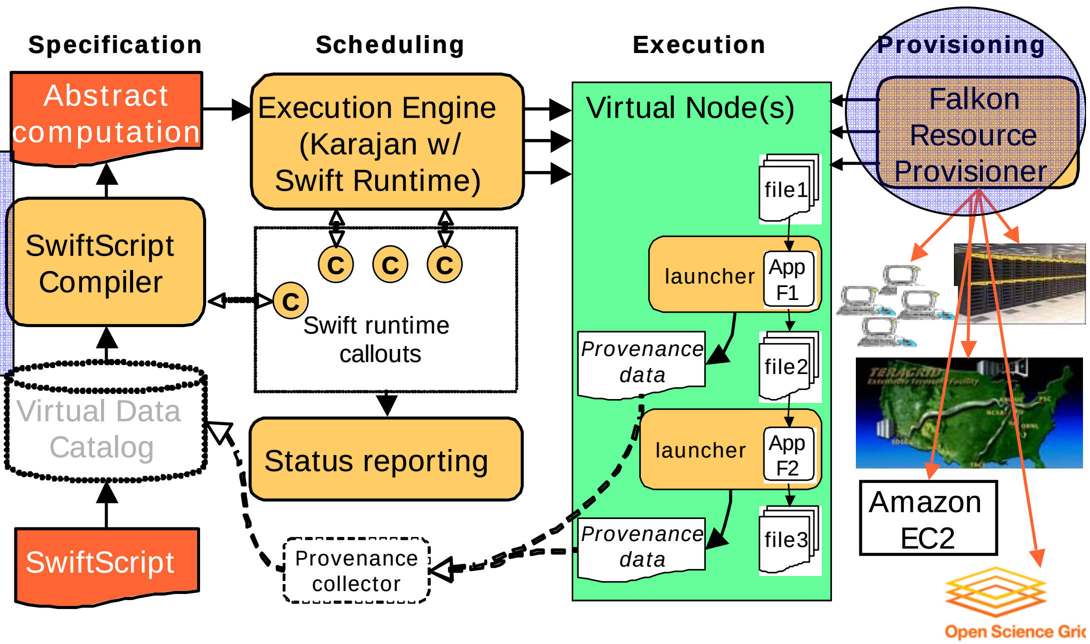


- Efficient:
  - High efficiency with second long tasks on 1000s of processors

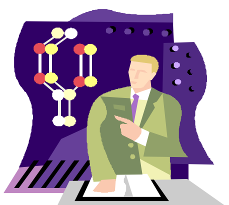
Falkon Integration with Swift

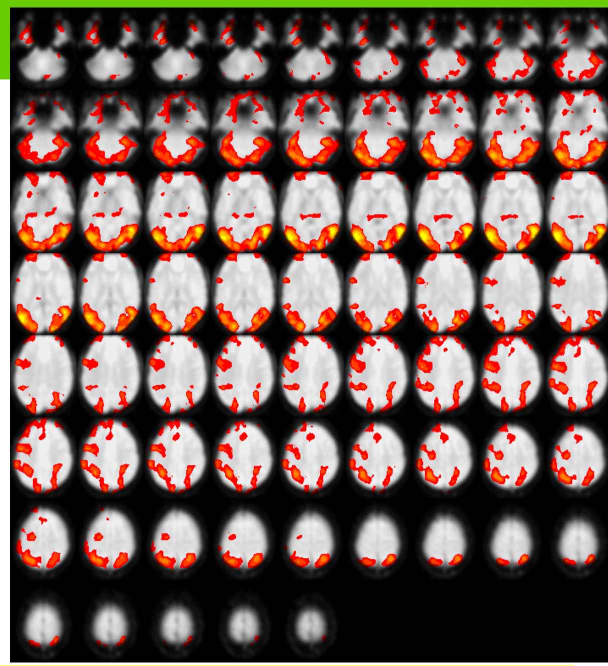
Application	#Tasks/workflow	#Stages
ATLAS: High Energy Physics Event Simulation	500K	1
fMRI DBIC: AIRSN Image Processing	100s	12
FOAM: Ocean/Atmosphere Model	2000	3
GADU: Genomics	40K	4
HNL: fMRI Aphasia Study	500	4
NVO/NASA: Photorealistic Montage/Morphology	1000s	16
QuarkNet/I2U2: Physics Science Education	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
MARS: Economic Modeling	1M~1B	1
DOCK: : Molecular Dynamics	1B	1

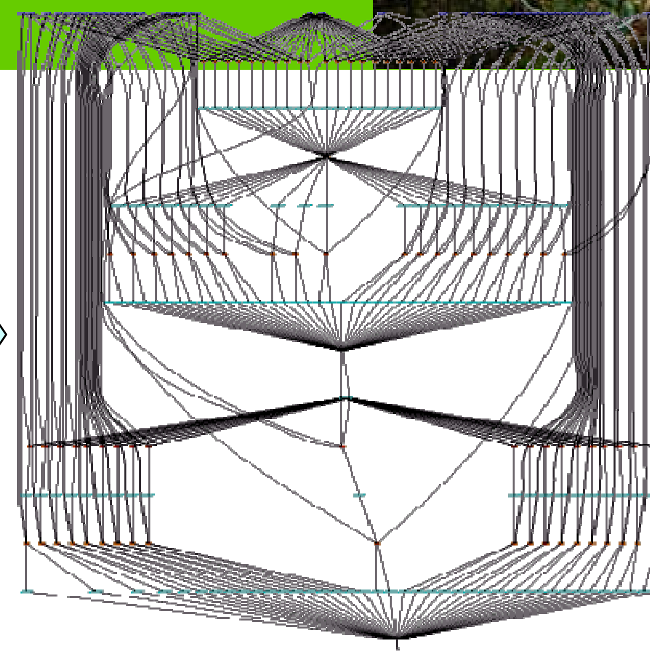
Swift Architecture



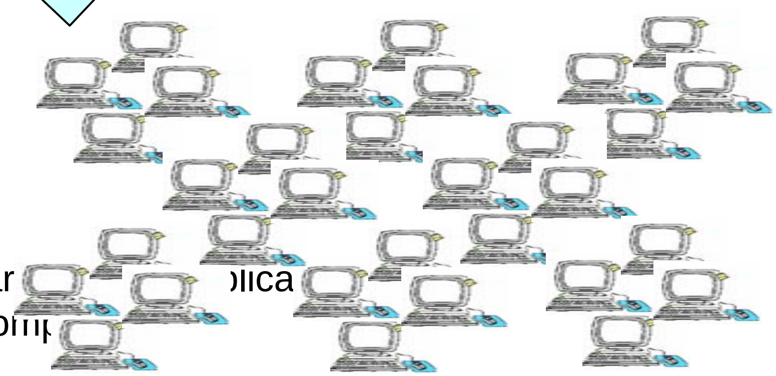
Functional MRI (fMRI)





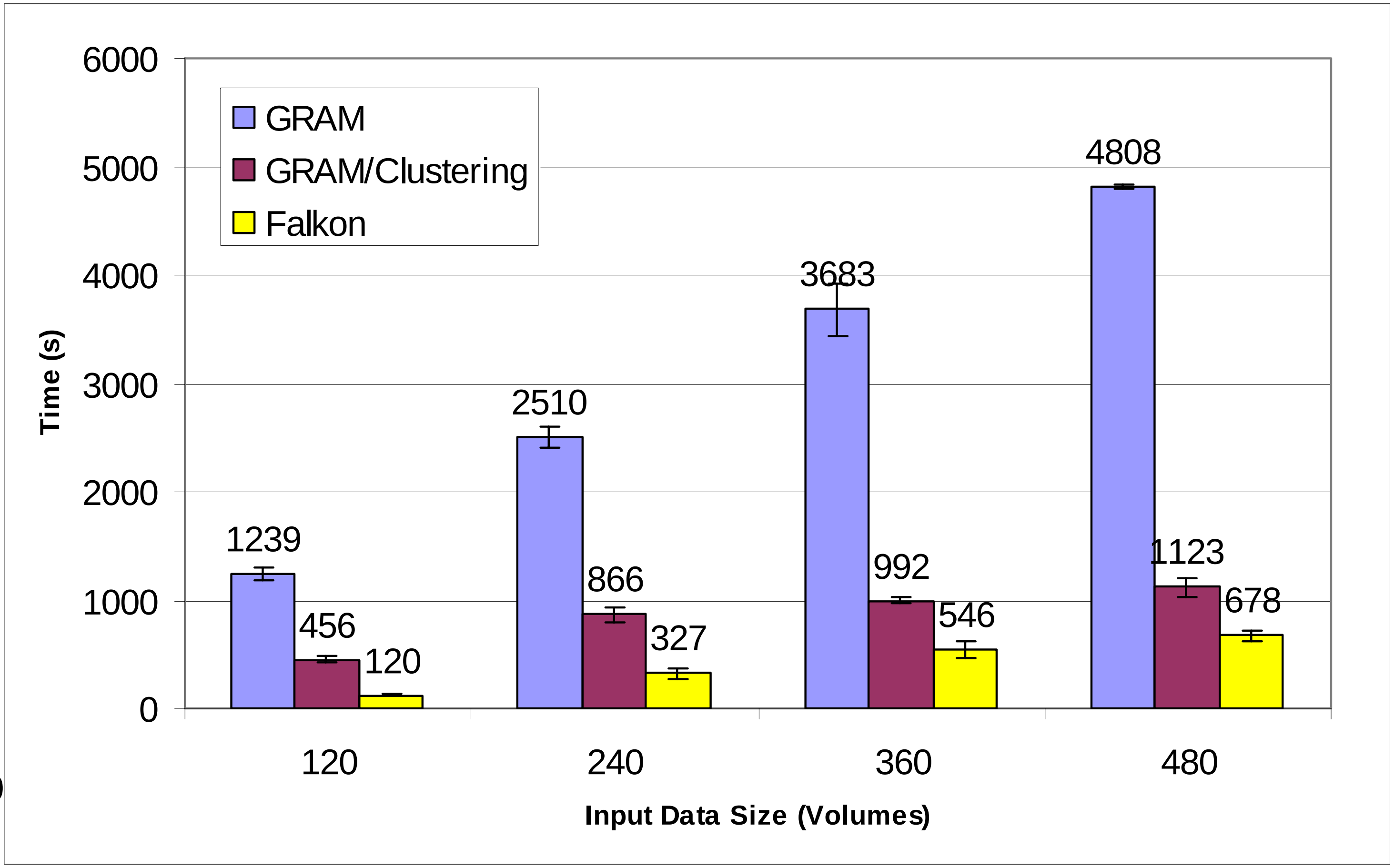


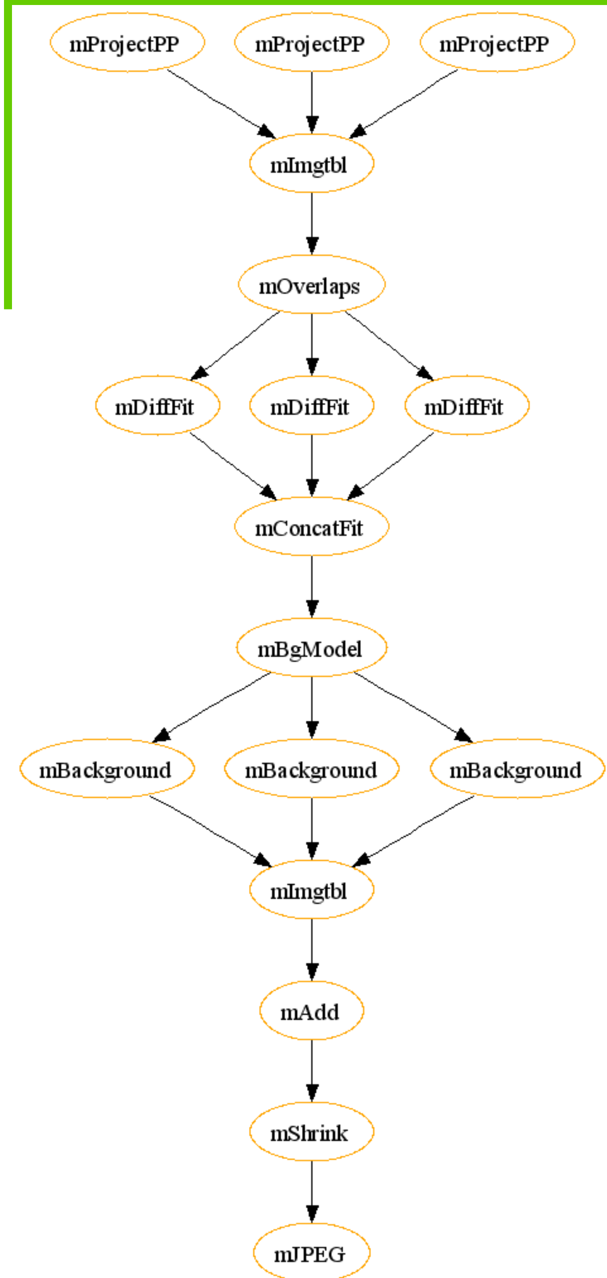
- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies



fMRI Application

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



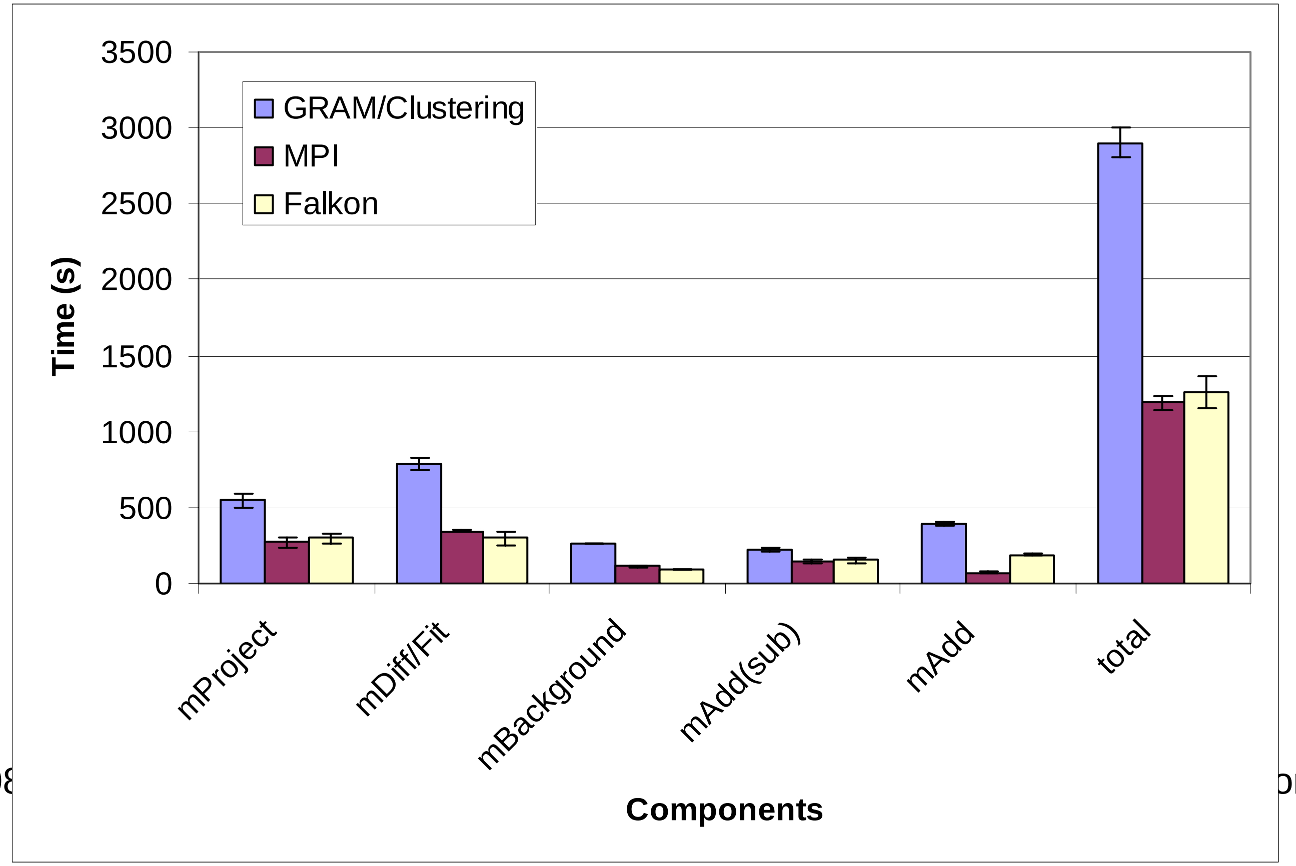


Montage

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

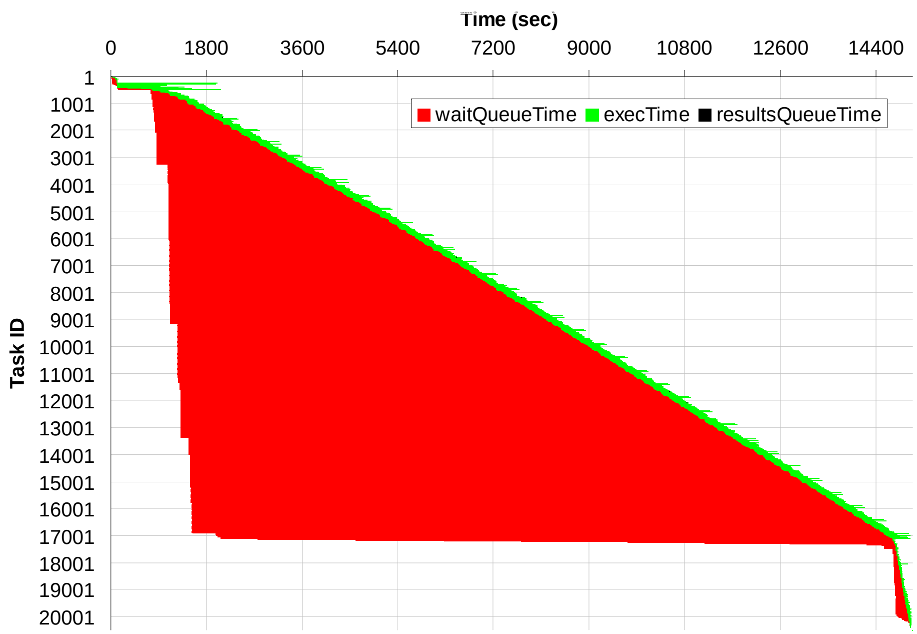
Montage Application

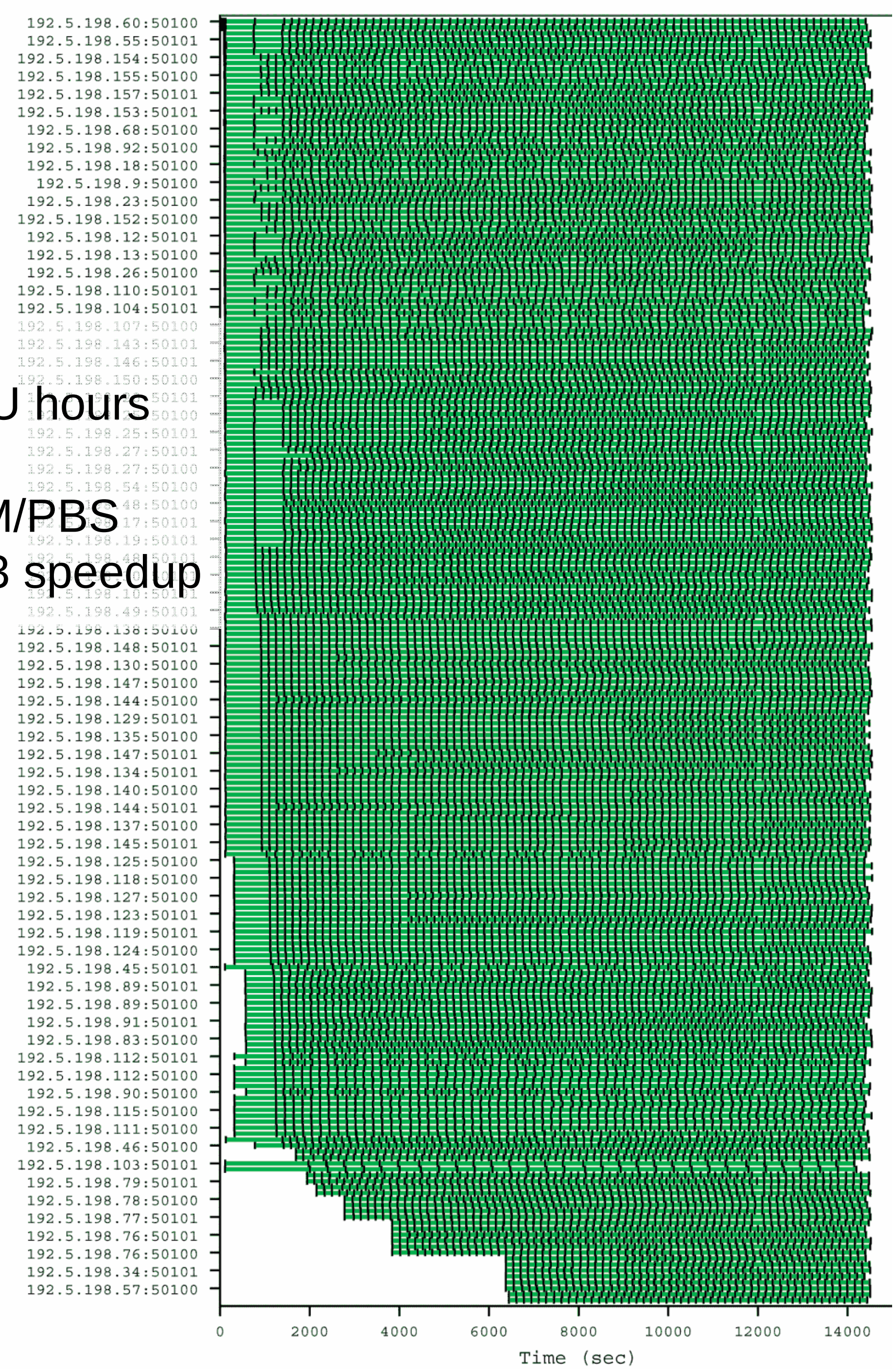
- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound



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

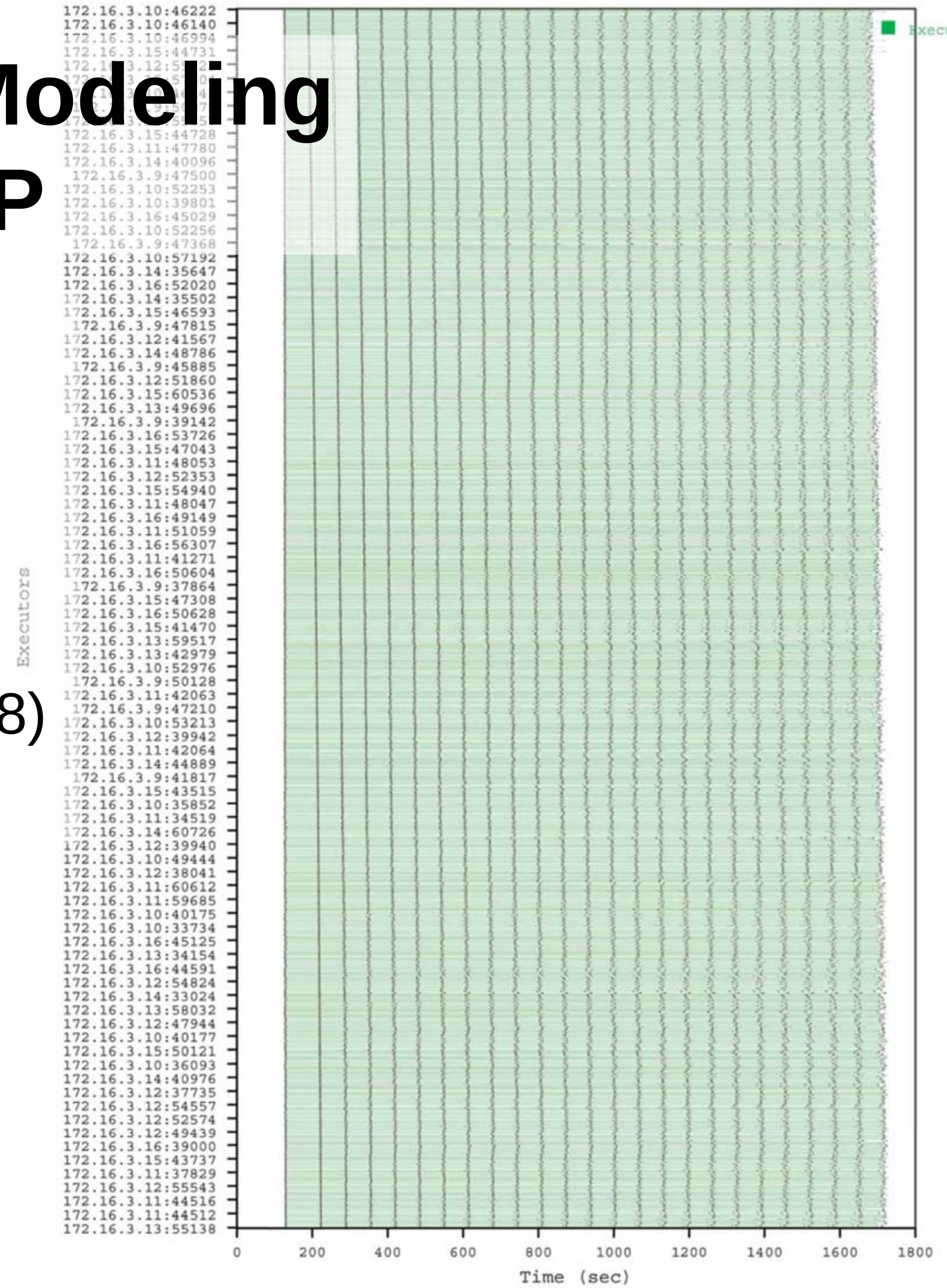




MARS Economic Modeling on IBM BG/P

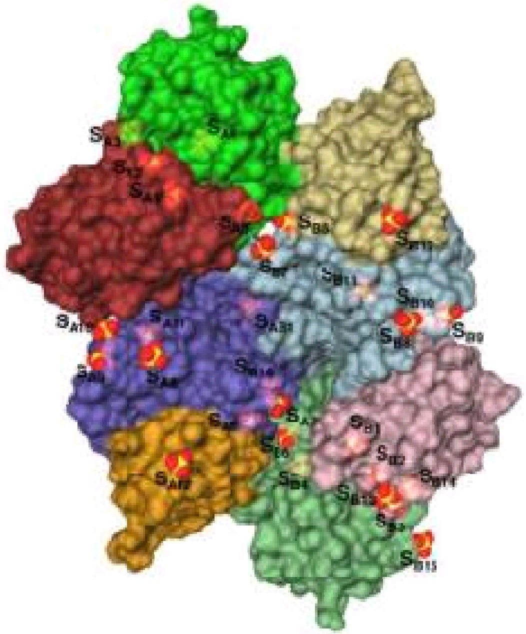
- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%

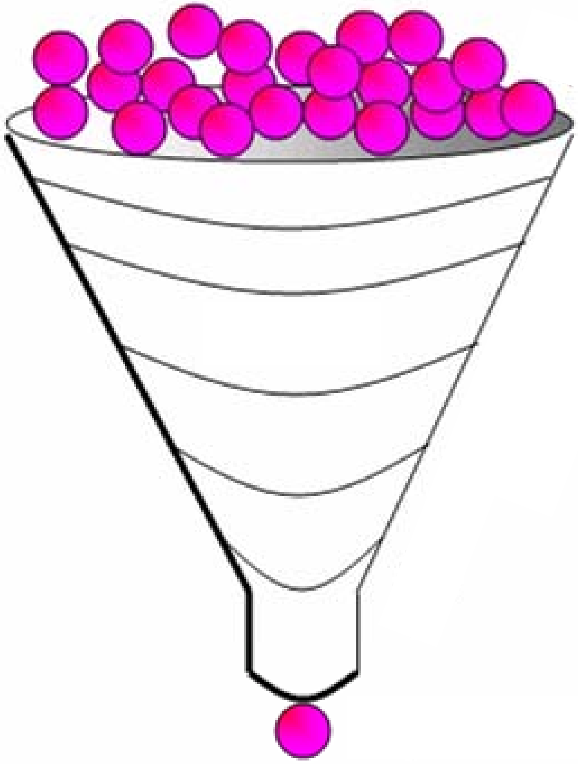
ting Loc
sters, Gr



Many Many Tasks: Identifying Potential Drug Targets

200+ Protein target(s) x 5M+ ligands

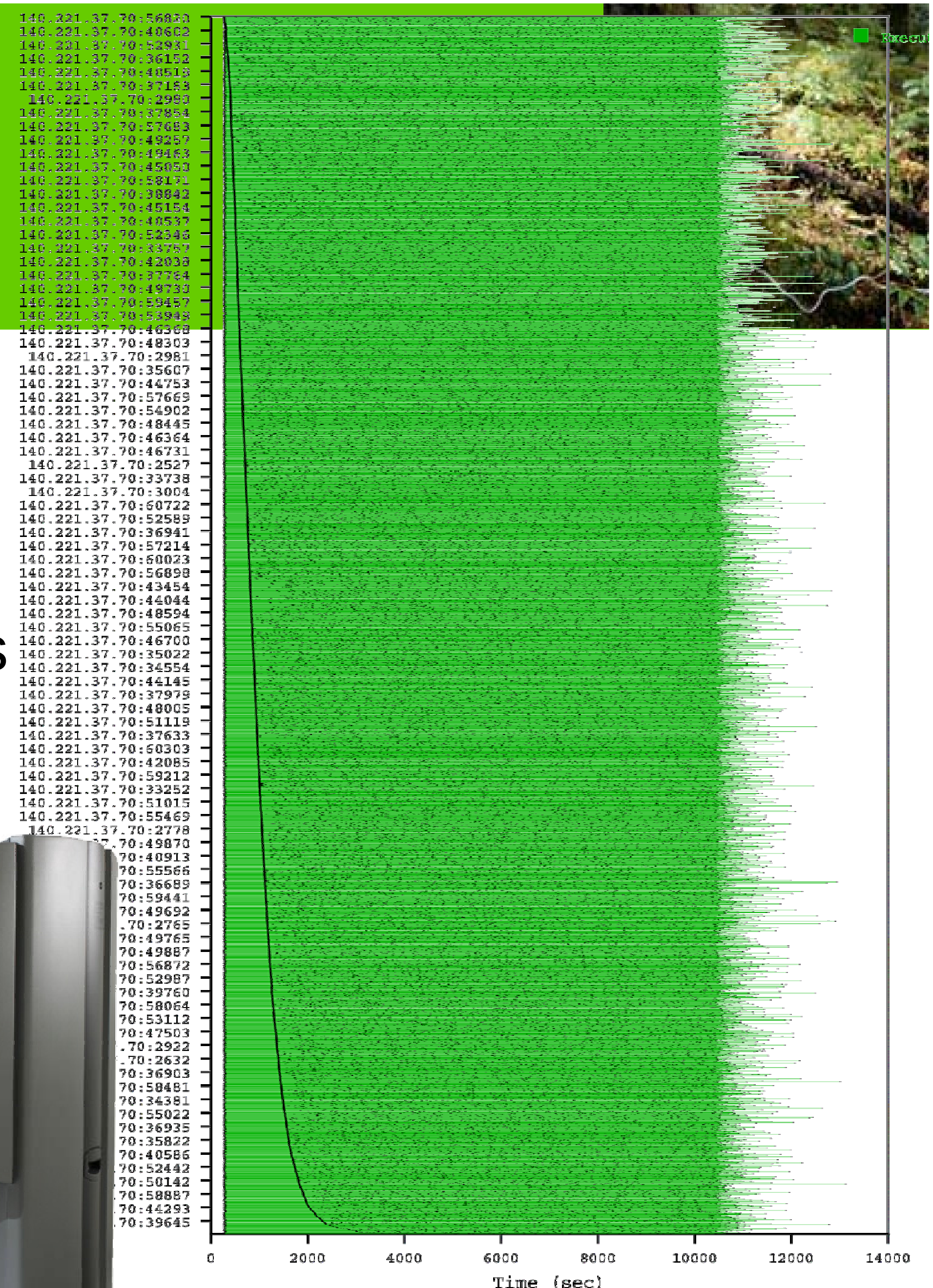




(Mike Kubal, Benoit Roux, and others)

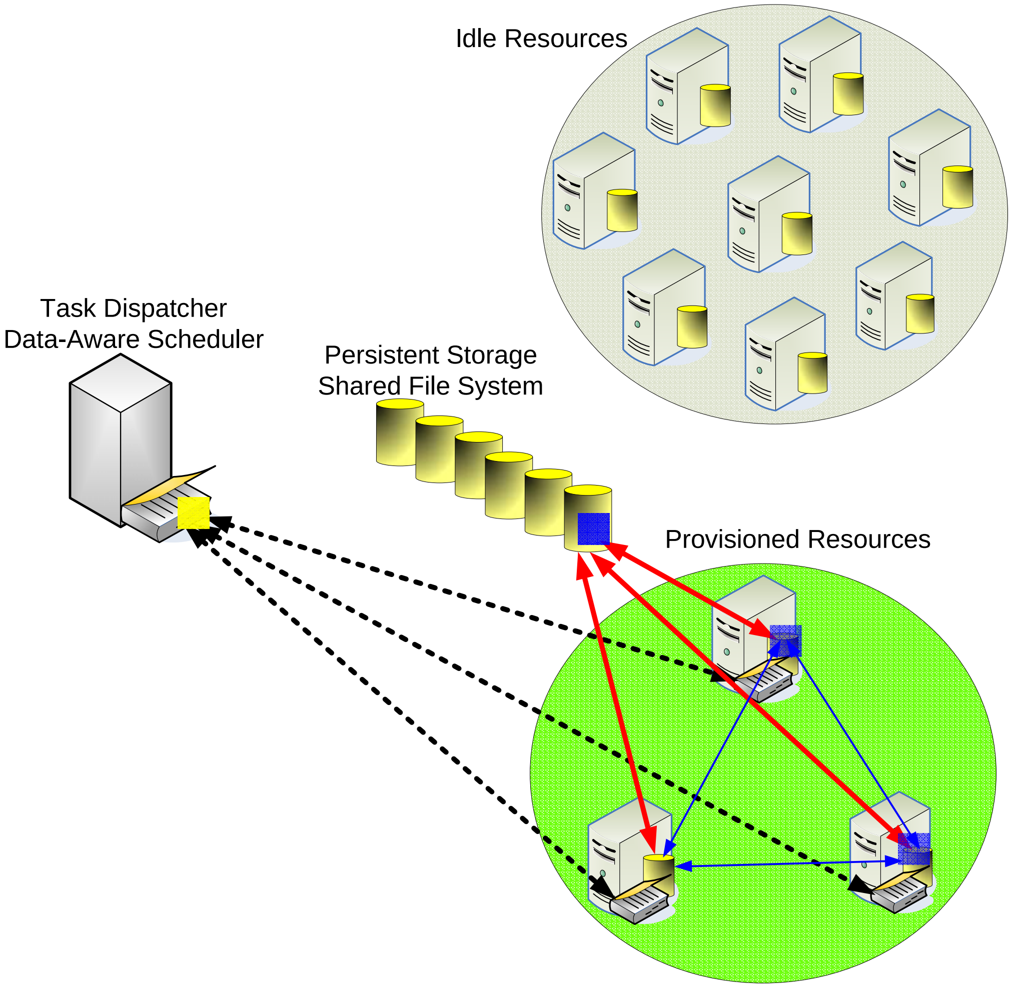
DOCK on SiCortex

- 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%



Falkon: Data Diffusion

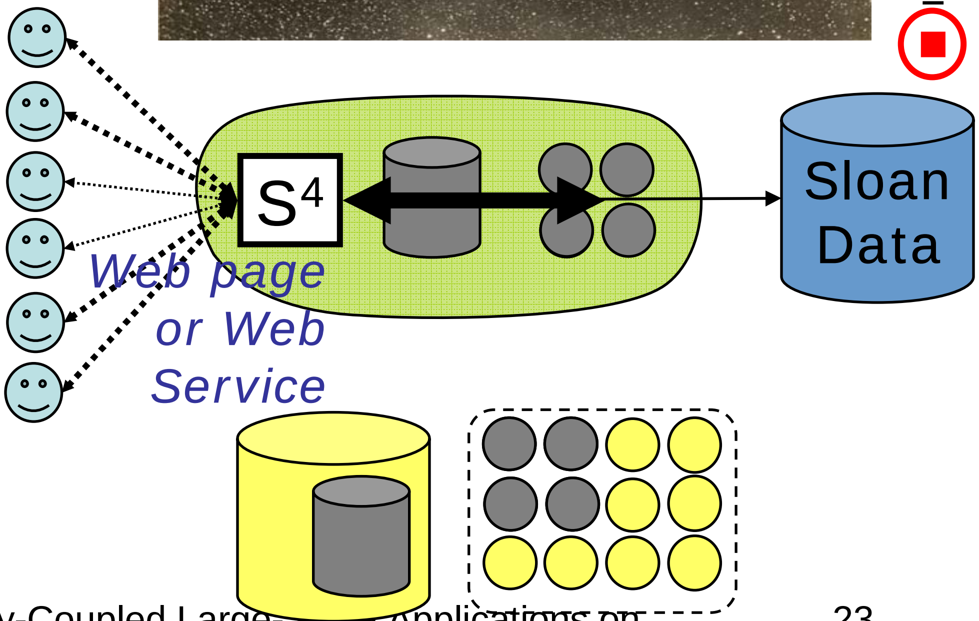
- Resource acquired in response to demand
- Data and applications diffuse from archival storage to newly acquired resources
- Resource "caching" allows faster responses to subsequent requests
  - Cache Eviction Strategies: RANDOM, FIFO, LRU, LFU
- Resources are released when demand drops



AstroPortal Stacking Service

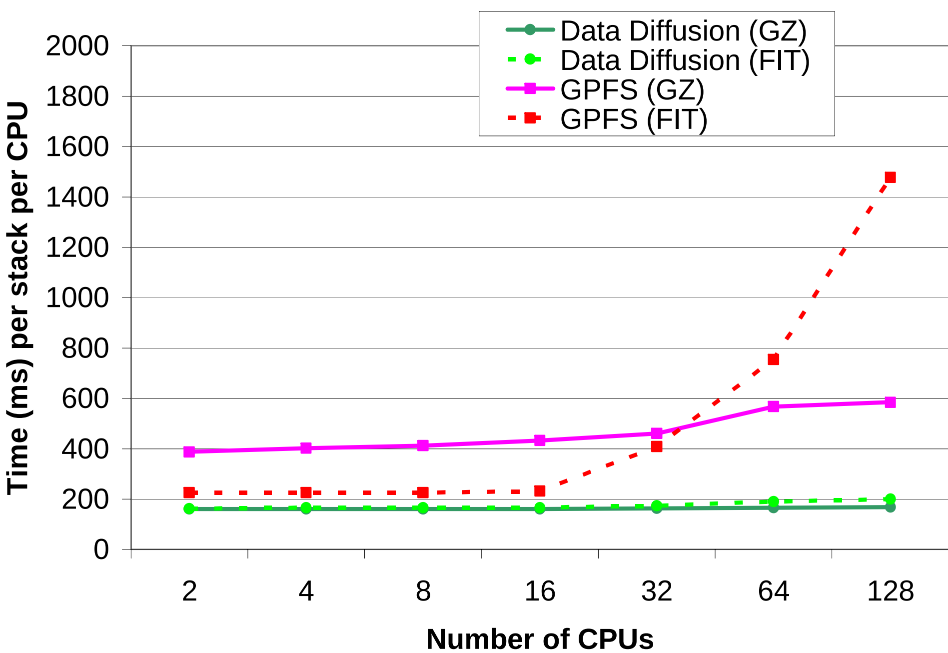
- Purpose
  - On-demand “stacks” of random locations within ~10TB dataset
- Challenge
  - Rapid access to 10-10K “random” files
  - Time-varying load
- Sample Workloads

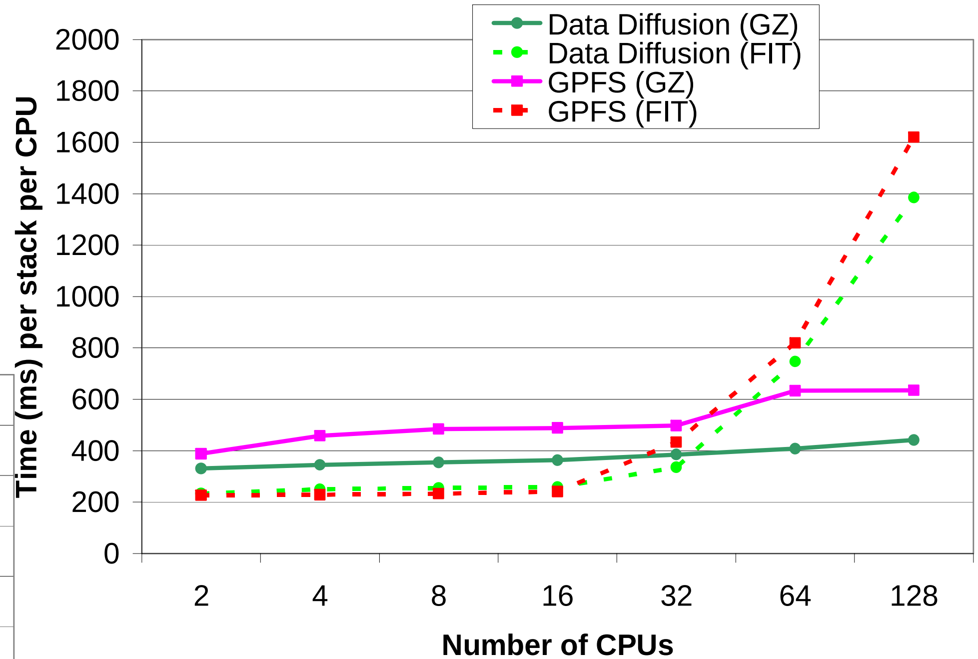
Locality	Number of Objects	Number of Files
1	111700	111700
1.38	154345	111699
2	97999	49000
3	88857	29620
4	76575	19145
5	60590	12120
10	46480	4650
20	40460	2025
30	23695	790



AstroPortal Stacking Service with Data Diffusion

- Low data locality ➔
  - Similar (but better) performance to GPFS

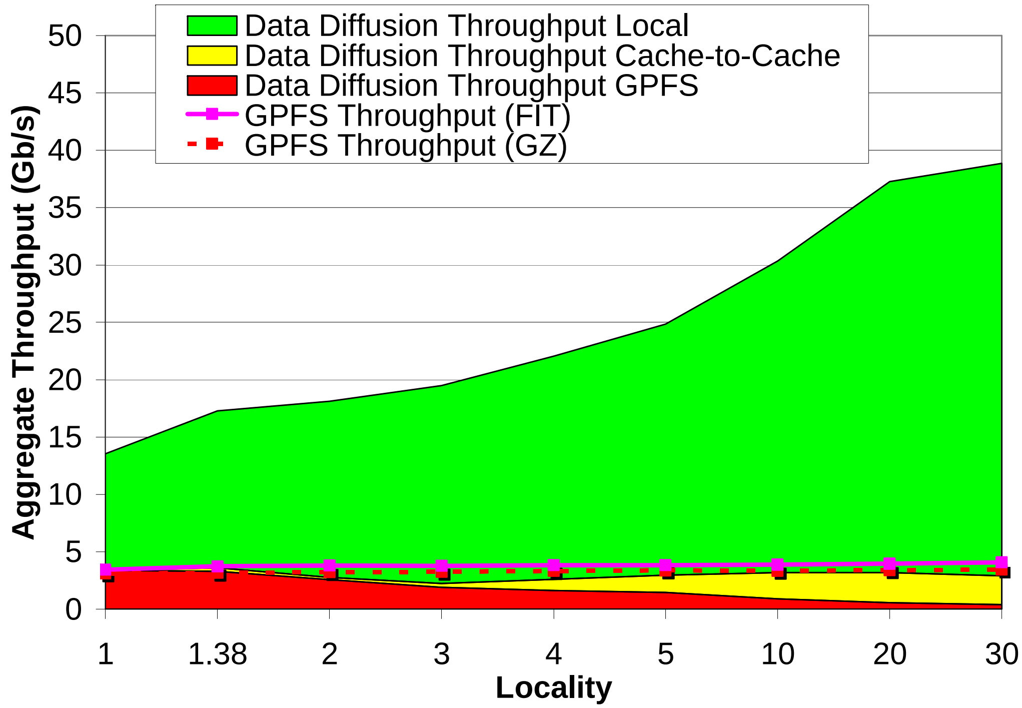


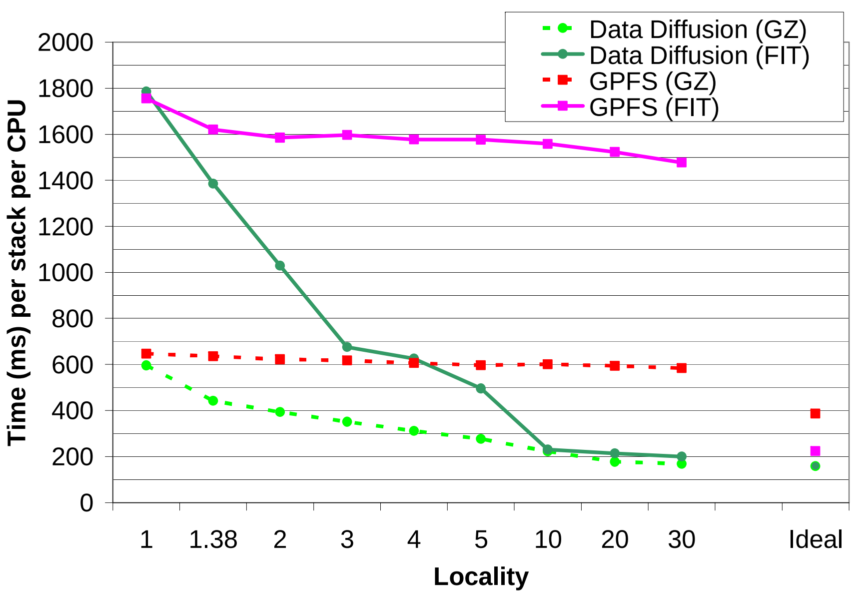


- 🡐High data locality
  - Near perfect scalability

AstroPortal Stacking Service with Data Diffusion

- Aggregate throughput:
  - 39Gb/s
  - 10X higher than GPFS
- Reduced load on GPFS
  - 0.49Gb/s
  - 1/10 of the original load

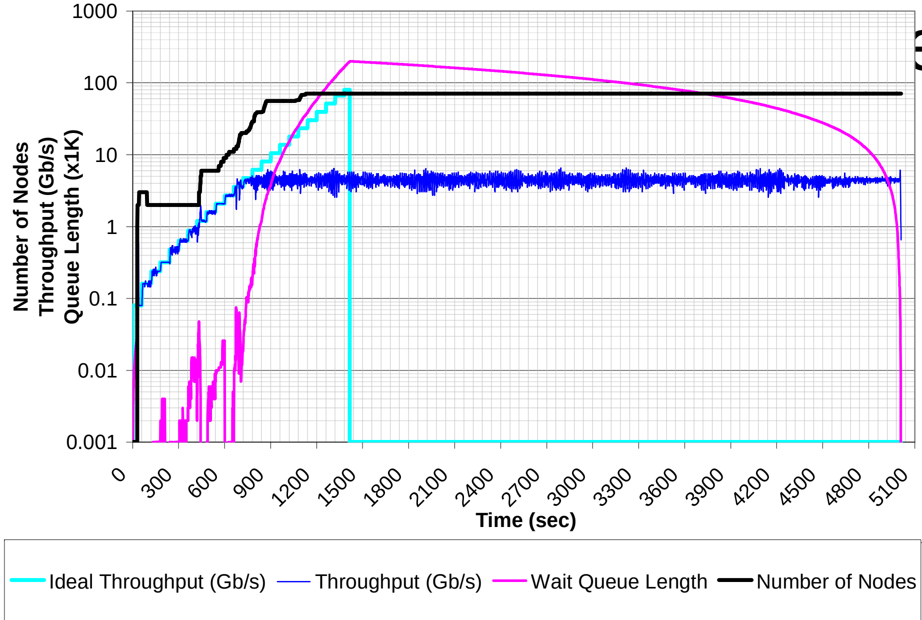


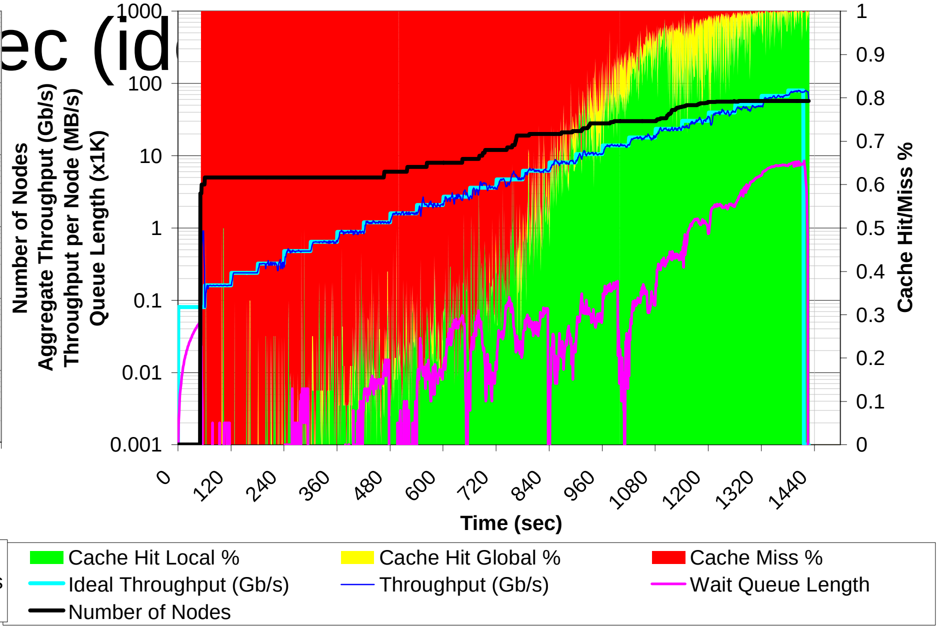


- Big performance gains as locality increases

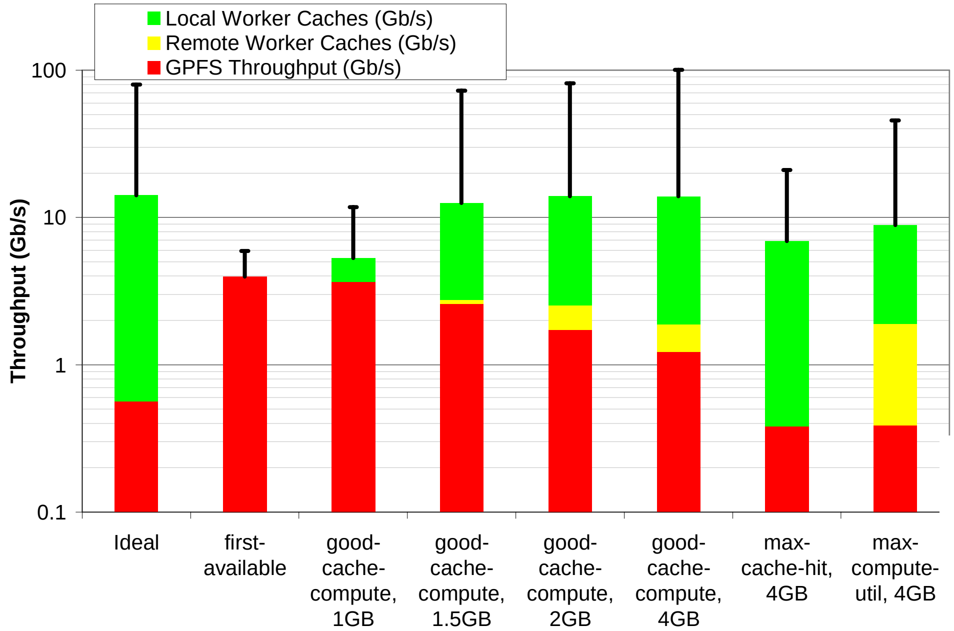
Data Diffusion: Data-Intensive Workload

- 250K tasks on 128 processors
  - 10MB read, 10ms compute
- Comparing GPFS with data diffusion





Data Diffusion: Data-Intensive Workload

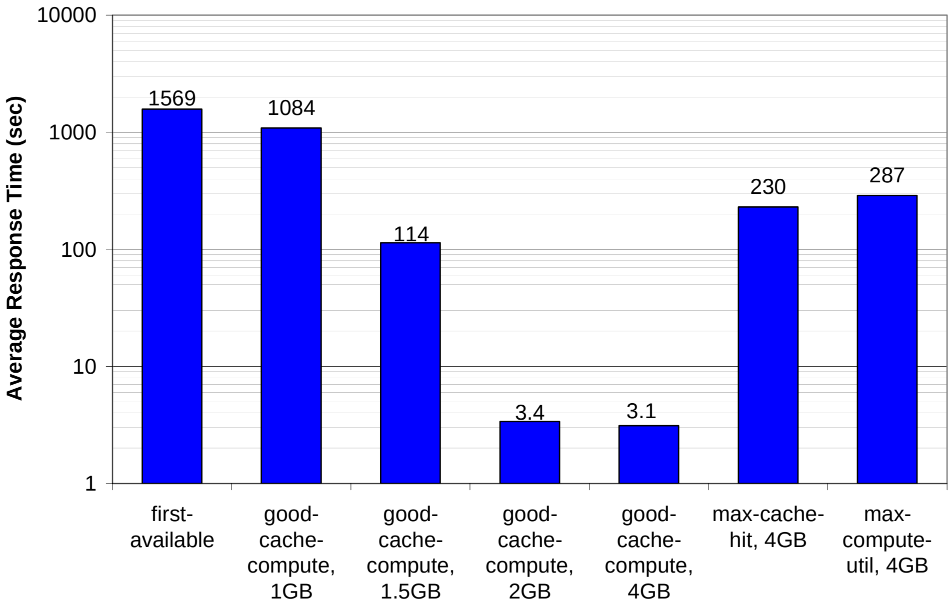


← Throughput:

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

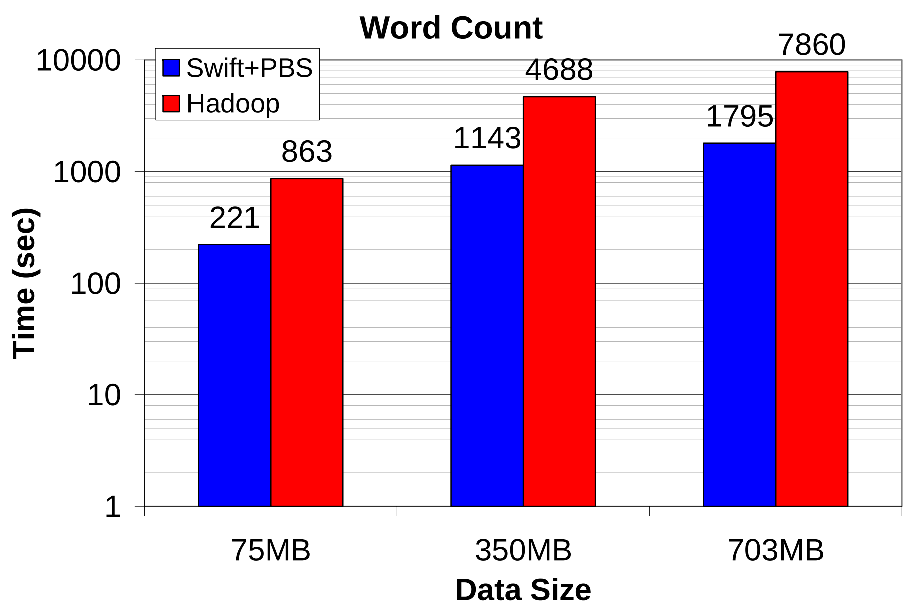
Response Time →

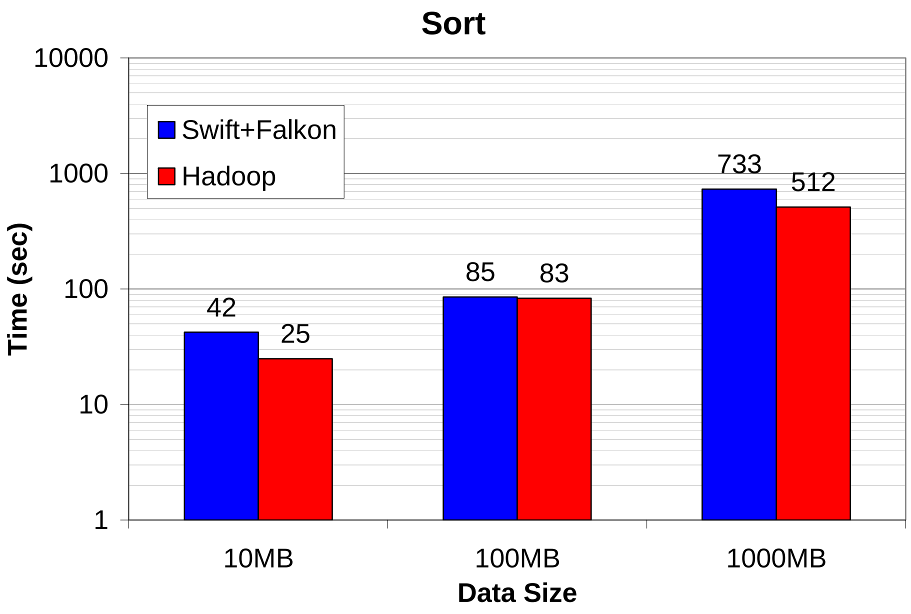
- 3 sec vs 1569 sec → 506X



Hadoop vs. Swift

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





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
- Loosely coupled apps do not require specialized system software
- Shared file systems are good all around solutions
  - They don't scale proportionally with the compute resources

Solutions

- Falkon
  - A Fast and Light-weight tasK executiON framework
  - Globus Incubator Project
  - http://dev.globus.org/wiki/Incubator/Falkon
- Swift
  - Parallel programming tool for rapid and reliable specification, execution, and management of large-scale science workflows
  - http://www.ci.uchicago.edu/swift/index.php
- Environments:
  - *Clusters*: TeraPort (TP)
  - *Grids*: Open Science Grid (OSG), TeraGrid (TG)
  - *Specialized large machines*: SiCortex 5732
  - *Supercomputers*: IBM BlueGene/P (BG/P)

More Information

- More information:
  - Personal research page: http://people.cs.uchicago.edu/~iraicu/
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - 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
  - Yong Zhao, Microsoft
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
  - Catalin Dumitrescu, Fermi National Laboratory
  - Zhao Zhang, The University of Chicago
  - Jerry C. Yan, NASA, Ames Research Center
- 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