THE UNIVERSITY OF CHICAGO

Harnessing Grid Resources with Data-Centric Task Farms

Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

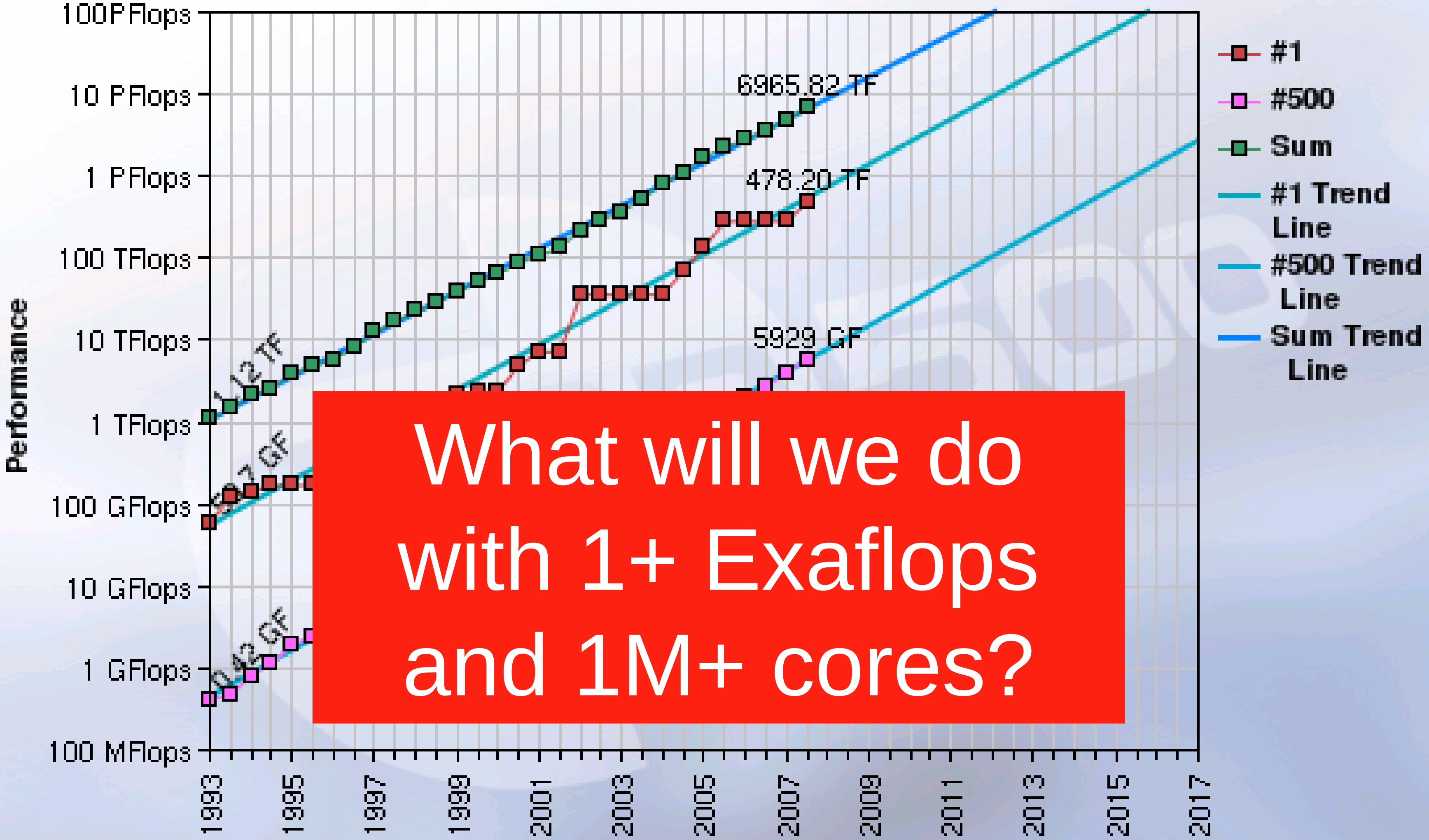
Collaborators:
Ian Foster (UC/CI/ANL), Yong Zhao (MS), Mike Wilde (CI/ANL), Zhao Zhang (CI), Rick Stevens (UC/CI/ANL), Alex Szalay (JHU), Jerry Yan (NASA/AMES), Catalin Dumitrescu (FANL)

NASA, Ames Research Center
GSRP Fellowship Talk
May 13th, 2008

Argonne NATIONAL LABORATORY

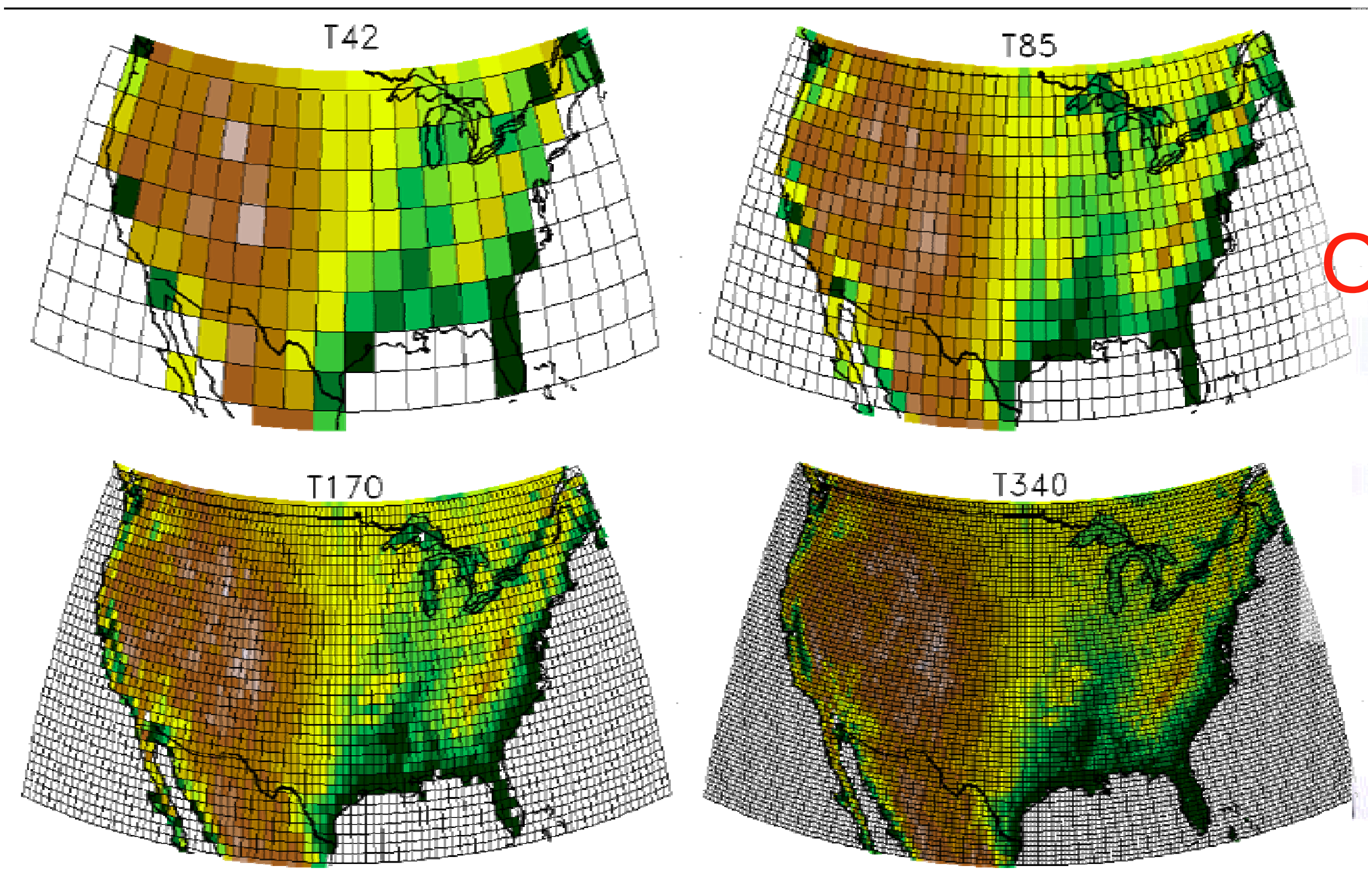
TOP500 SUPERCOMPUTER SITES

Projected Performance Development

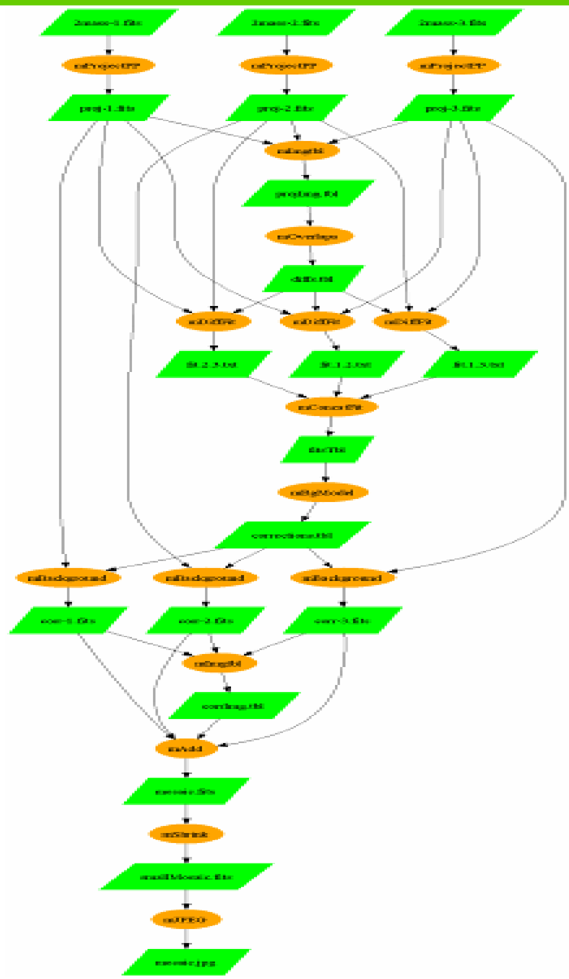


What will we do with 1+ Exaflops and 1M+ cores?

1) Tackle Bigger and Bigger Problems



2) Tackle Increasingly Complex Problems



Computational Scientist

as

Logistics Officer

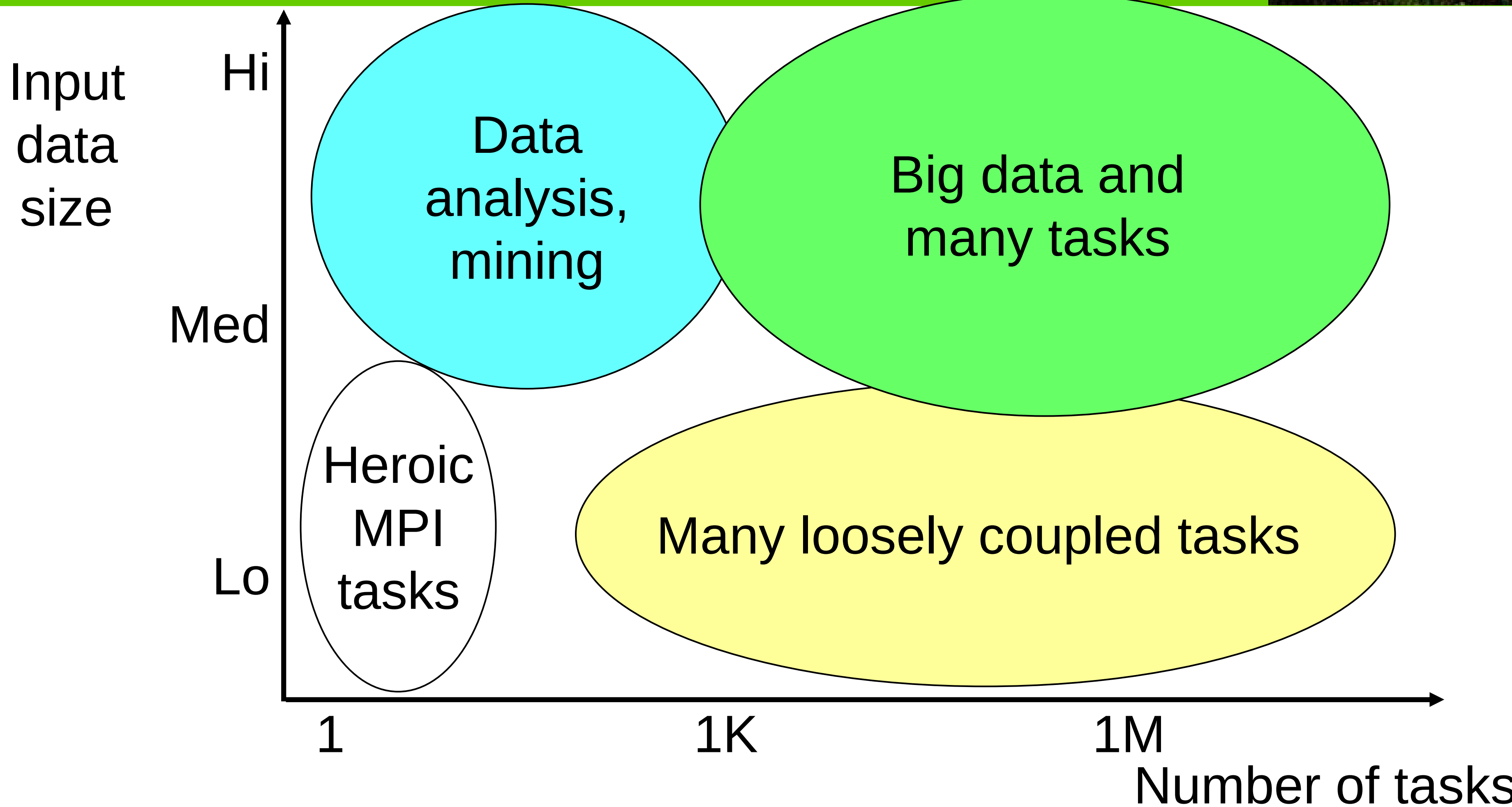
“More Complex Problems”

- Use ensemble runs to quantify **climate model uncertainty**
- Identify **potential drug targets** by screening a database of ligand structures against target proteins
- Study **economic model sensitivity** to key parameters
- Analyze **turbulence dataset** from multiple perspectives
- Perform **numerical optimization** to determine optimal resource assignment in energy problems

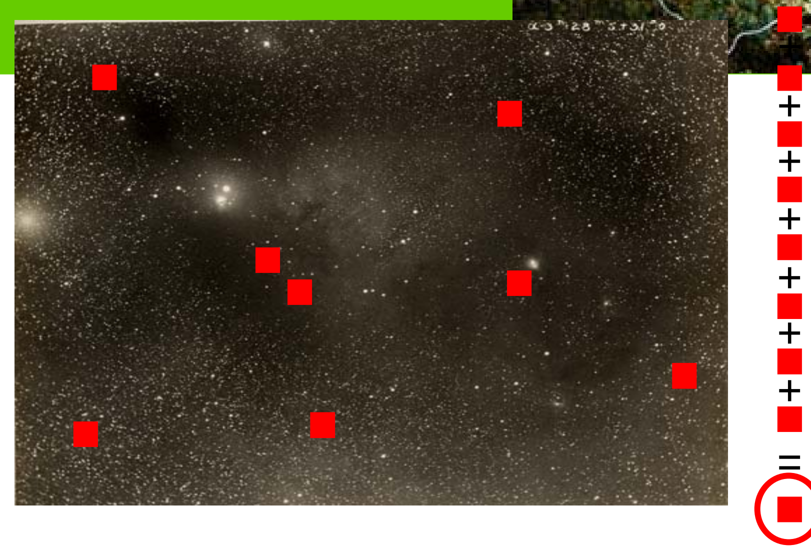
Programming Model Issues

- **Multicore** processors
- Massive **task parallelism**
- Massive **data parallelism**
- Integrating **black box applications**
- Complex **task dependencies** (task graphs)
- **Failure**, and other execution management issues
- **Data management**: input, intermediate, output
- **Dynamic task graphs**
- **Dynamic data access** involving large amounts of data
- Documenting **provenance** of data products

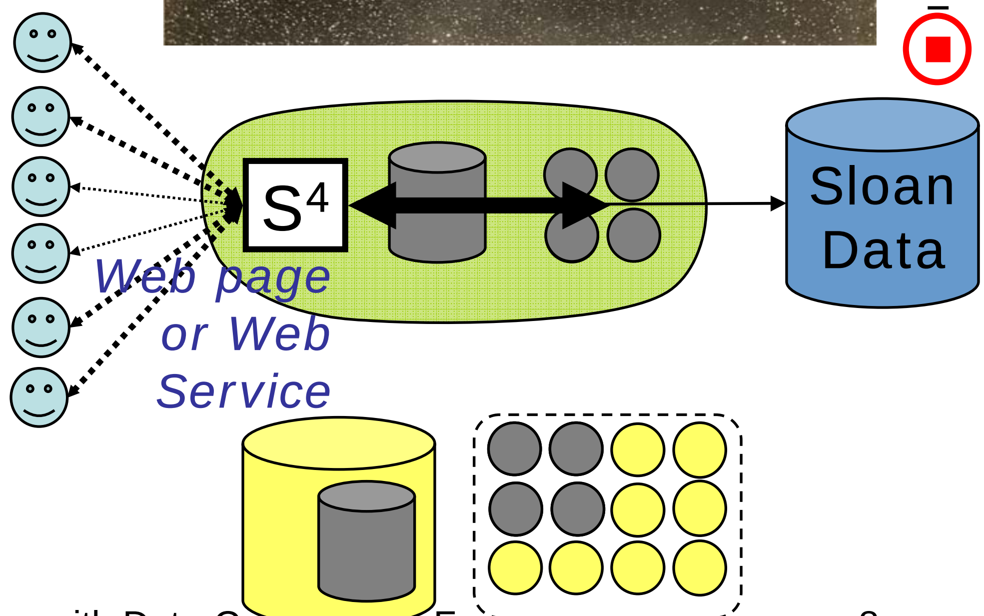
Problem Types



Motivating Example: AstroPortal Stacking Service



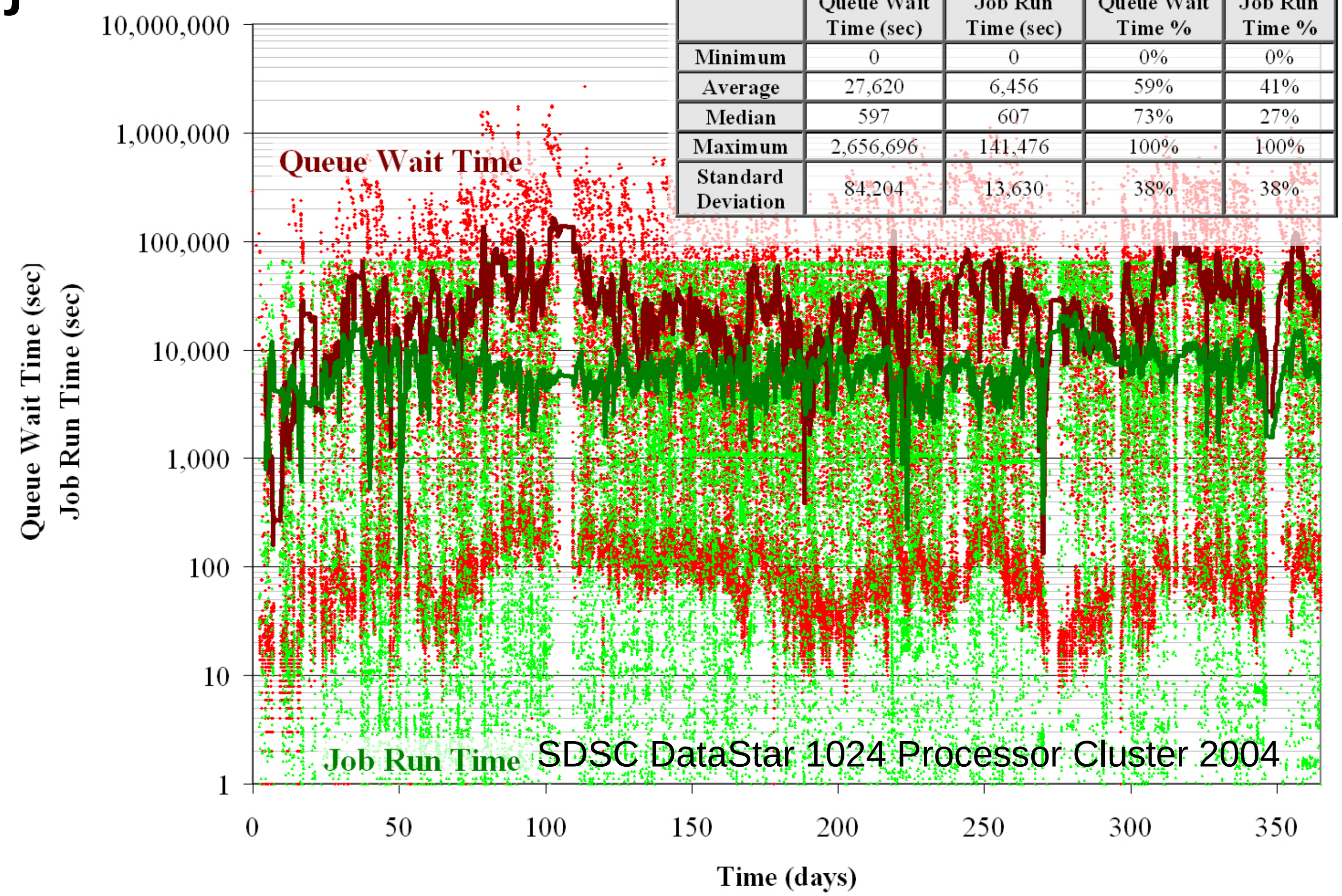
- Purpose
  - On-demand "stacks" of random locations within ~10TB dataset
- Challenge
  - Rapid access to 10-10K "random" files
  - Time-varying load
- Solution
  - Dynamic acquisition of compute, storage



Challenge #1:
Long Queue Times

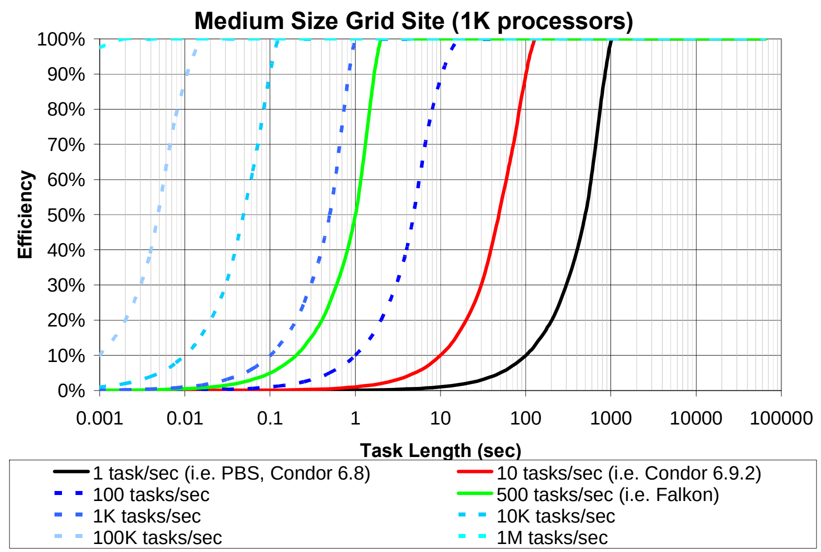
- Wait queue times are typically longer than the job duration times

	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%



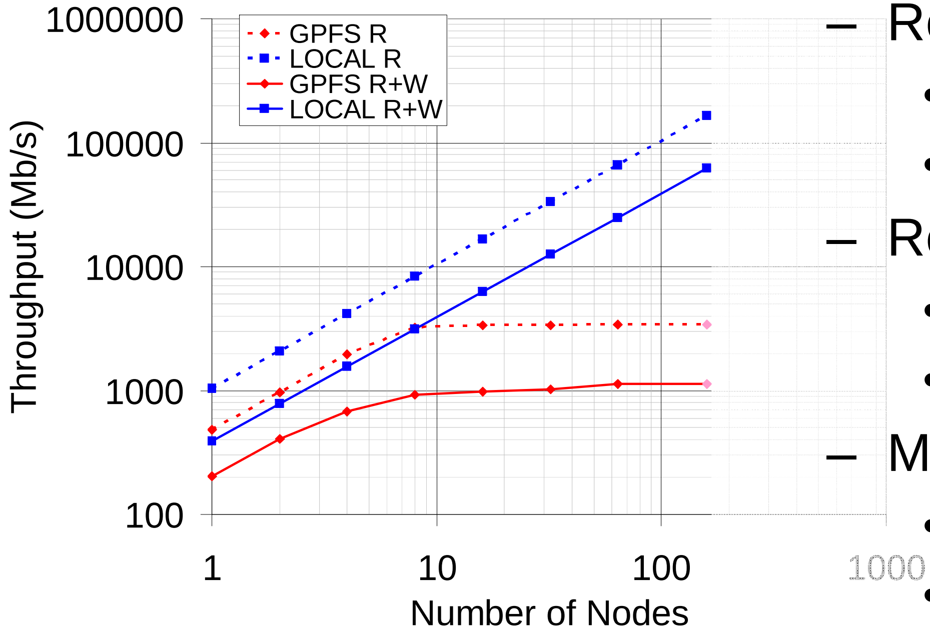
Challenge #2: Slow Job Dispatch Rates

- 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

Proposed Solution: Part 1
Abstract Model and Validation

- AMDASK:
  - An Abstract Model for DAta-centric taSK farms
    - Task Farm: A common parallel pattern that drives independent computational tasks
  - Models the efficiency of data analysis workloads for the split/merge class of applications
  - Captures the following data diffusion properties
    - Resources are acquired in response to demand
    - Data and applications diffuse from archival storage to new resources
    - Resource “caching” allows faster responses to subsequent requests
    - Resources are released when demand drops
    - Considers both data and computations to optimize performance
- Model Validation
  - Implement the abstract model in a discrete event simulation
  - Validate model with statistical methods (R^2 Statistic, Residual Analysis)

Proposed Solution: Part 2
Practical Realization

- Falkon: a Fast and Light-weight tasK executiON framework
  - Light-weight task dispatch mechanism
  - Dynamic resource provisioning to acquire and release resources
  - Data management capabilities including data-aware scheduling
  - Integration into Swift to leverage many Swift-based applications
    - Applications cover many domains: astronomy, astro-physics, medicine, chemistry, and economics

AMDASK: Performance Efficiency Model

- B: Average Task Execution Time:
  - K: Stream of tasks
  - μ(k): Task k execution time

$$B = \frac{1}{|K|}\sum_{k \in K} \mu(\kappa)$$

- Y: Average Task Execution Time with Overheads:
  - o(k): Dispatch overhead
  - ς(δ,τ): Time to get data

$$Y = \begin{cases} \frac{1}{|K|}\sum_{\kappa \in K}[\mu(\kappa) + o(\kappa)], & \delta \in \phi(\tau), \delta \in \Omega \\ \frac{1}{|K|}\sum_{\kappa \in K}[\mu(\kappa) + o(\kappa) + \zeta(\delta,\tau)], & \delta \notin \phi(\tau), \delta \in \Omega \end{cases}$$

- V: Workload Execution Time:
  - A: Arrival rate of tasks
  - T: Transient Resources

$$V = \max\left(\frac{B}{|T|}, \frac{1}{A}\right) * |K|$$

- W: Workload Execution Time with Overheads

$$W = \max\left(\frac{Y}{|T|}, \frac{1}{A}\right) * |K|$$

AMDASK: Performance Efficiency Model

- **Efficiency**

$$E = \frac{V}{W} \longrightarrow E = \begin{cases} 1, & \frac{Y}{|T|} \leq \frac{1}{A} \\ \max\left(\frac{B}{Y}, \frac{|\mathrm{T}|}{\mathrm{A} * Y}\right), & \frac{Y}{|T|} > \frac{1}{A} \end{cases}$$

- **Speedup**

$$S = E * |T|$$

- **Optimizing Efficiency**
  - Easy to maximize either efficiency or speedup independently
  - Harder to maximize both at the same time
    - Find the smallest number of *transient resources* |T| while maximizing speedup*efficiency

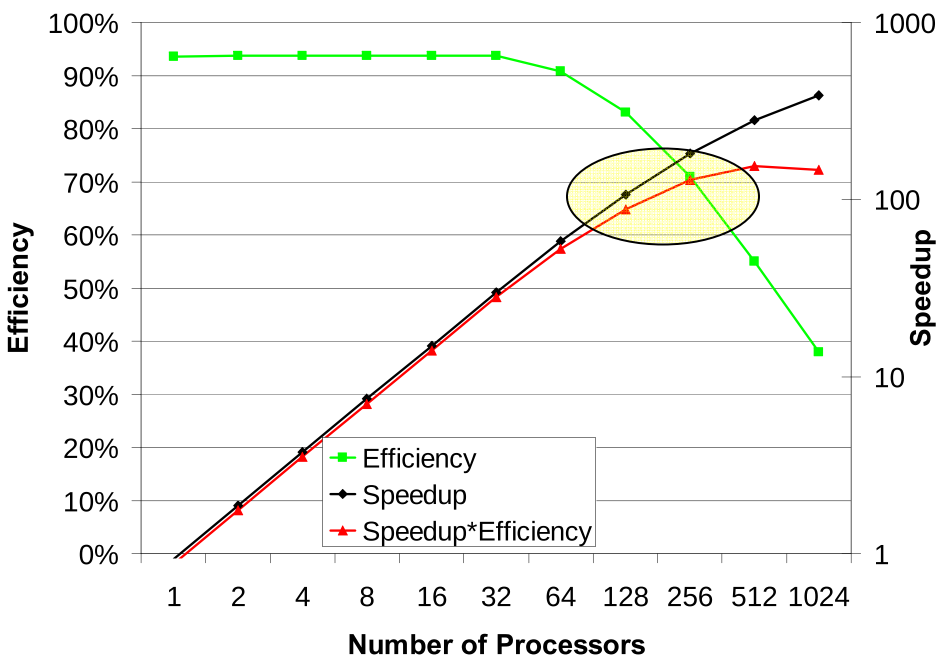
Performance Efficiency Model Example: 1K CPU Cluster

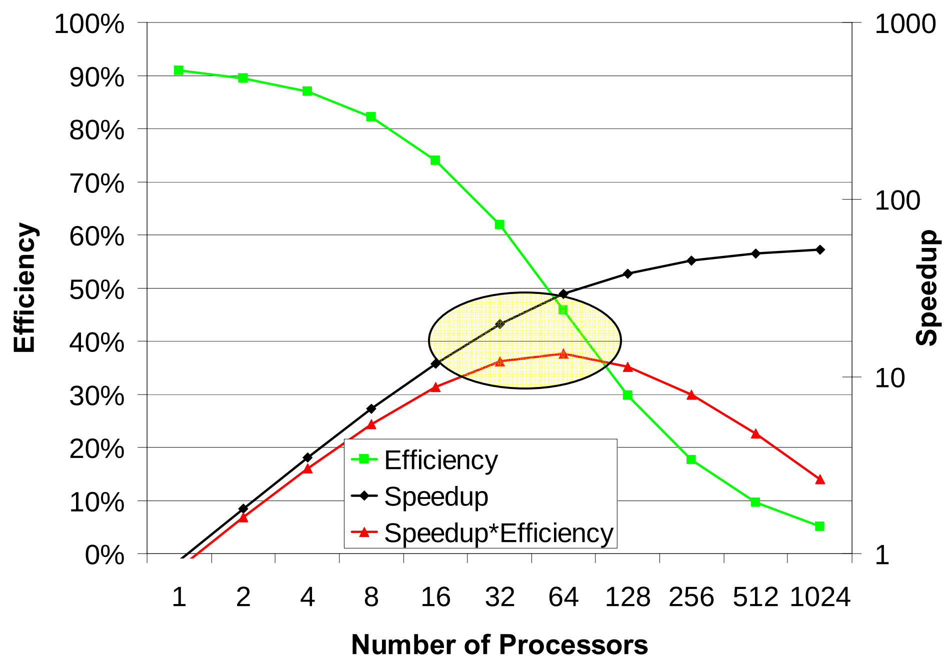
- Application: Angle - distributed data mining
- Testbed Characteristics:
  - Computational Resources: 1024
  - Transient Resource Bandwidth: 10MB/sec
  - Persistent Store Bandwidth: 426MB/sec
- Workload:
  - Number of Tasks: 128K
  - Arrival rate: 1000/sec
  - Average task execution time: 60 sec
  - Data Object Size: 40MB

Performance Efficiency Model Example: 1K CPU Cluster

Falkon on ANL/UC TG Site:
Peak *Dispatch Throughput*: 500/sec
Scalability: 50~500 CPUs
Peak *speedup*: 623x

PBS on ANL/UC TG Site:
Peak *Dispatch Throughput*: 1/sec
Scalability: <50 CPUs
Peak *speedup*: 54x





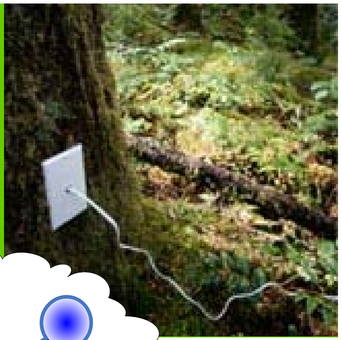
Model Validation: Simulations

- Implement the abstract model in a discrete event simulation
- Simulation parameters
  - number of storage and computational resources
  - communication costs
  - management overhead
  - workloads (inter-arrival rates, query complexity, data set properties, and data locality)
- Model Validation
  - R^2 Statistic
  - Residual analysis

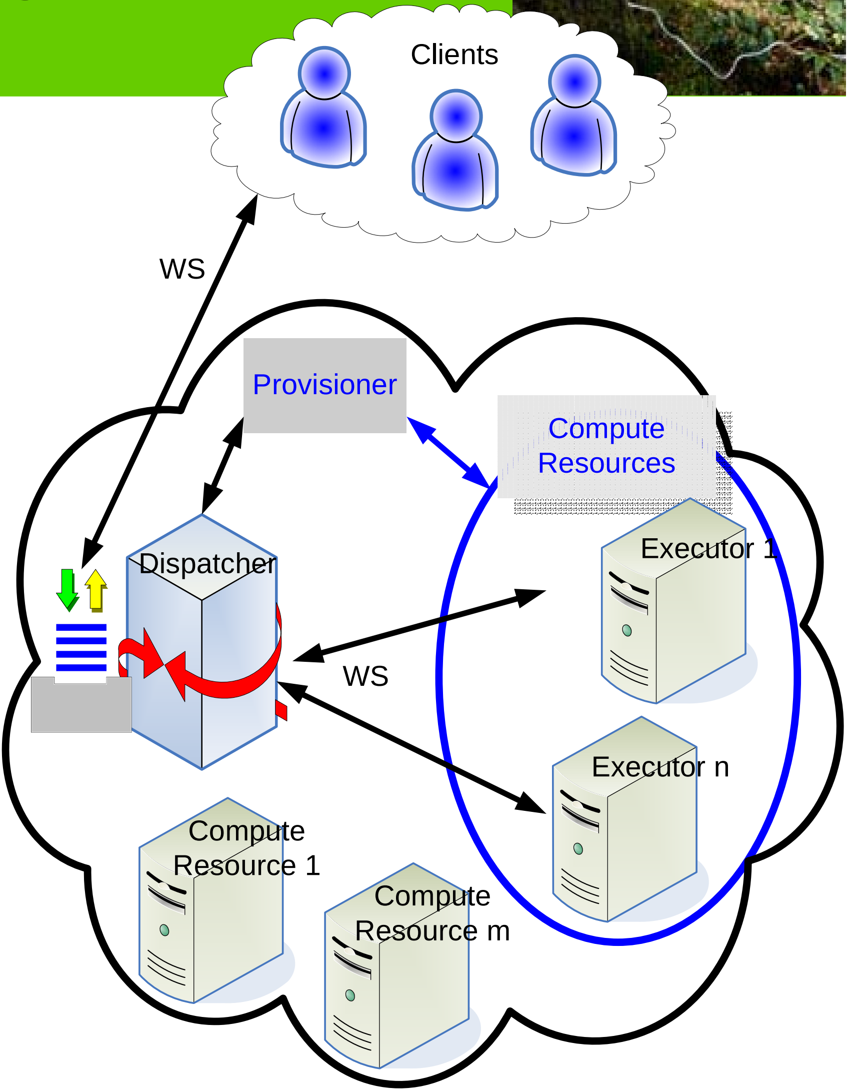
Falkon: a Fast and Light-weight tasK executiON framework

- ***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: The Streamlined Task Dispatcher

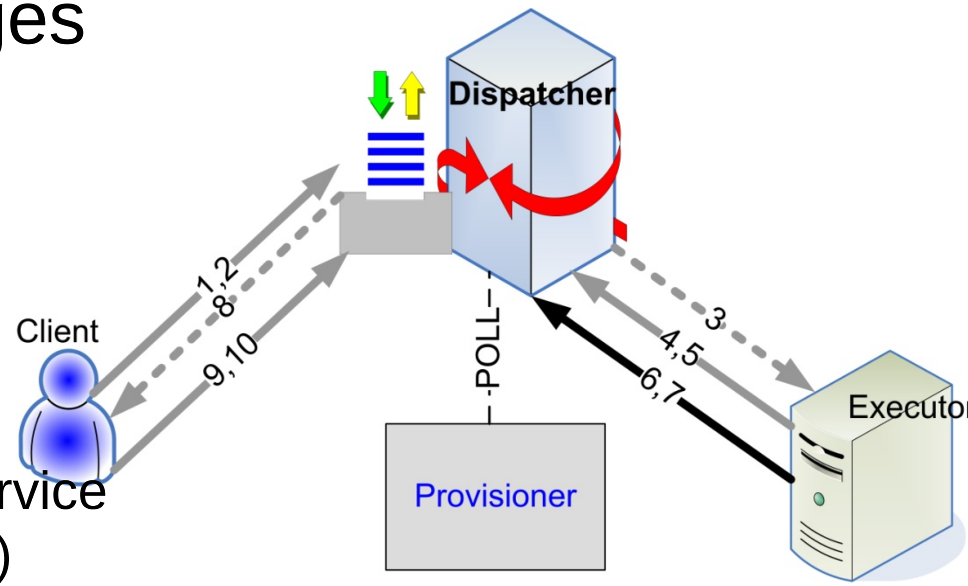


- Tier 1: Dispatcher
  - GT4 Web Service accepting task submissions from clients and sending them to available executors
- Tier 2: Executor
  - Run tasks on local resources
- Provisioner
  - Static and dynamic resource provisioning



Falkon: The Streamlined Task Dispatcher

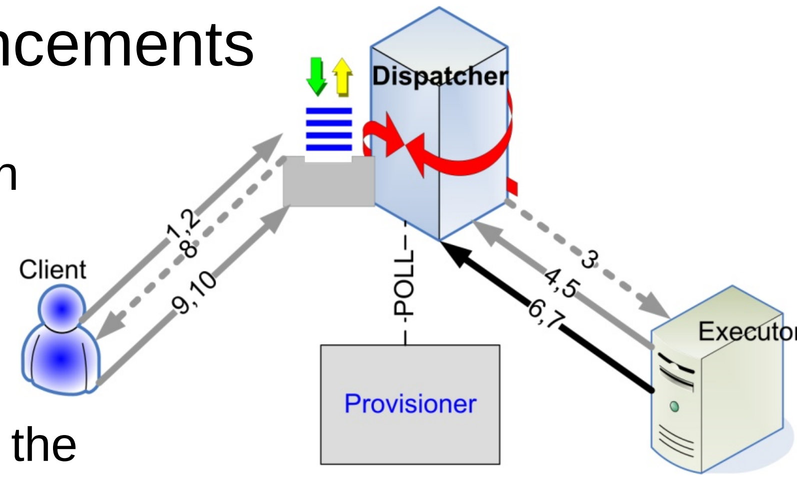
- Falkon Message Exchanges
  - Description:
    - {1}: task(s) submit
    - {2}: task(s) submit confirmation
    - {3}: notification for work
    - {4}: request for task(s)
    - {5 or 7}: dispatch task(s)
    - {6}: deliver task(s) results to service
    - {8}: notification for task result(s)
    - {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

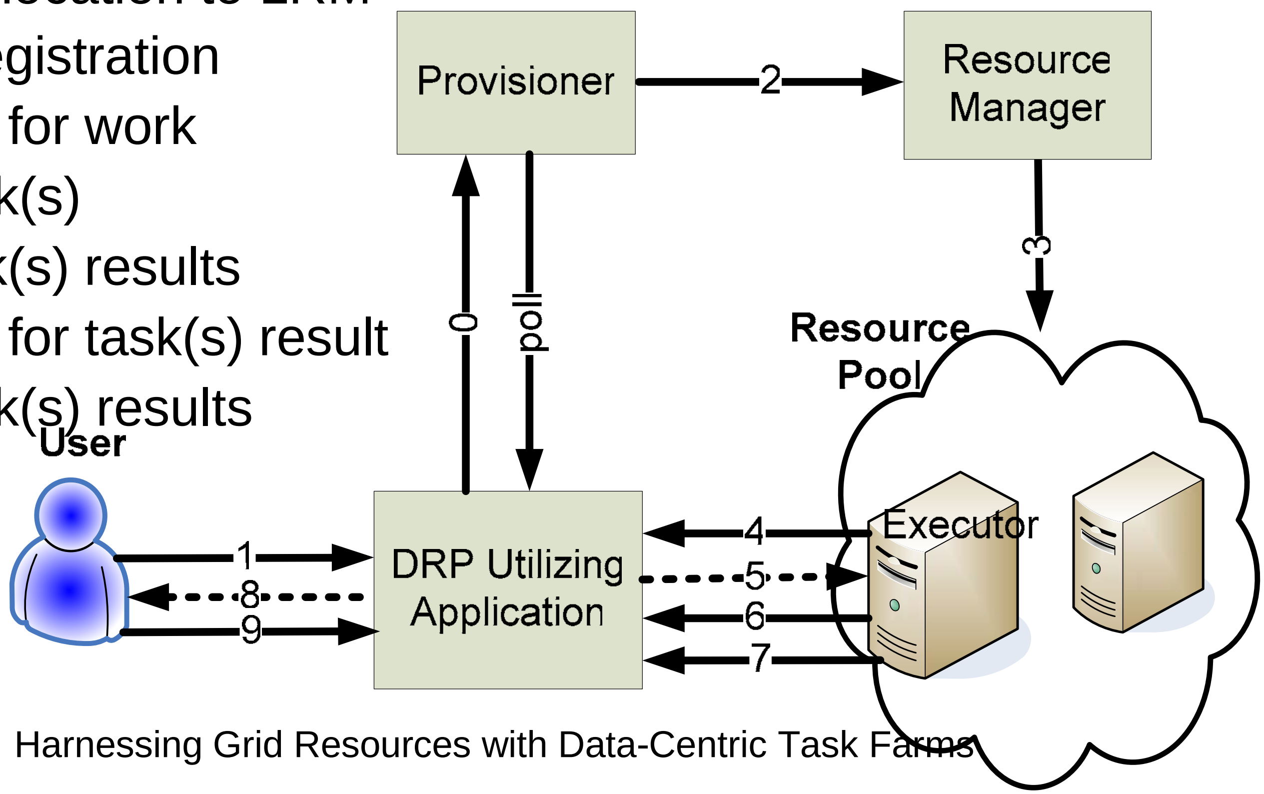
Falkon: The Streamlined Task Dispatcher

- Falkon Message Exchanges 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→2+c, where c is a small positive value
    - Application Specific Lower Bound: 10→0+c, where c is a small positive value



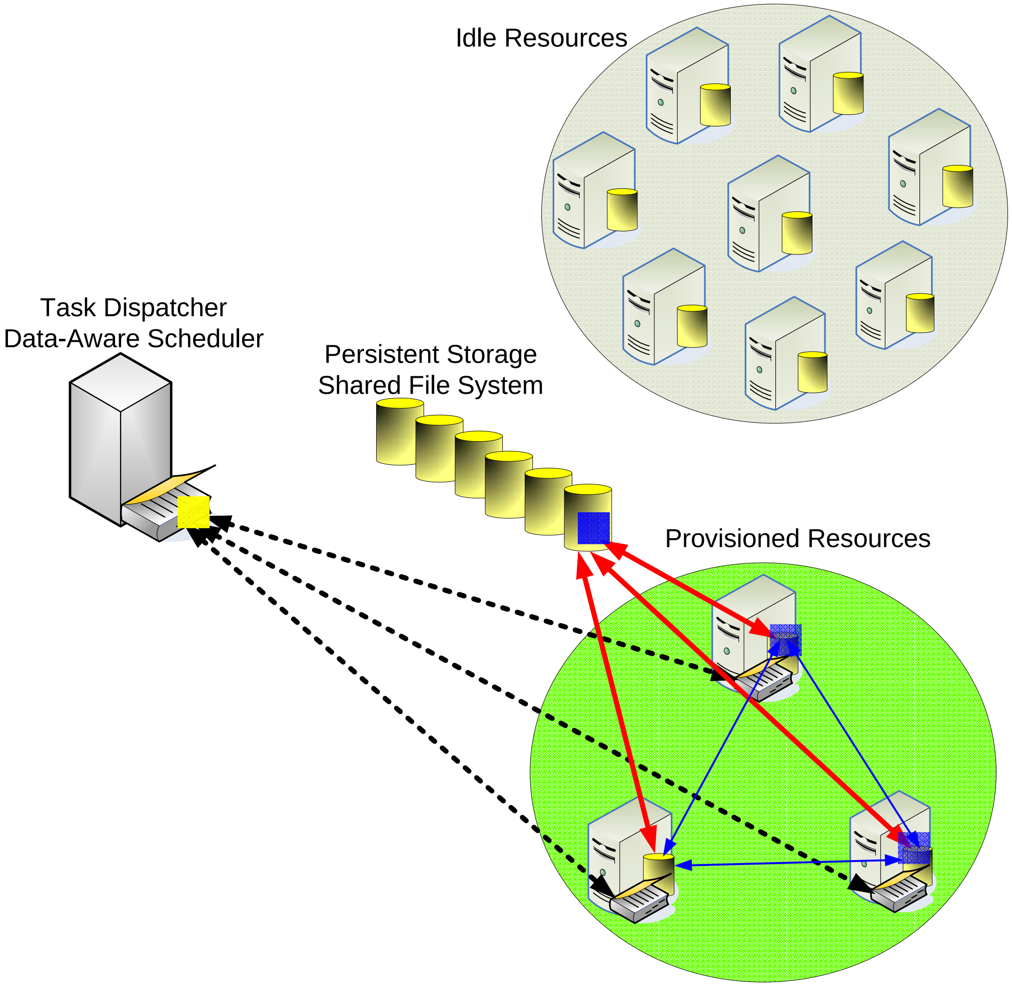
Falkon: Resource Provisioning

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(s) result
9. pick up task(s) results



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



Falkon: Data Diffusion

- Considers both data and computations to optimize performance
- Decrease dependency of a shared file system
  - Theoretical linear scalability with compute resources
  - Significantly increases meta-data creation and/or modification performance
- Completes the “data-centric task farm” realization

Related Work: Task Farms

- [*Casanova99*]: Adaptive Scheduling for Task Farming with Grid Middleware
- [*Heymann00*]: Adaptive Scheduling for Master-Worker Applications on the Computational Grid
- [*Danelutto04*]: Adaptive Task Farm Implementation Strategies
- [*González-Vélez05*]: An Adaptive Skeletal Task Farm for Grids
- [*Petrou05*]: Scheduling Speculative Tasks in a Compute Farm
- [*Reid06*]: Task farming on Blue Gene

Conclusion: none addressed the proposed “data-centric” part of task farms

Related Work: Task Dispatch

- [*Zhou92*]: **LSF** – Load Sharing Cluster Management
- [*Bode00*]: **PBS** – Portable Batch Scheduler and Maui Scheduler
- [*Anderson04*]: **BOINC** – Task Distribution for Volunteer Computing
- [*Thain05*]: **Condor**
- [*Robinson07*]: **Condor-J2** – Turning Cluster Management into Data Management

Conclusion: related work is several orders of magnitude slower

Related Work: Resource Provisioning

- [*Appleby01*]: **Oceano** - SLA Based Management of a Computing Utility
- [*Frey02, Mehta06*]: **Condor glide-ins**
- [*Walker06*]: **MyCluster** (based on Condor glide-ins)
- [*Ramakrishnan06*]: Grid Hosting with Adaptive Resource Control
- [*Bresnahan06*]: Provisioning of bandwidth
- [*Singh06*]: Simulations

Conclusion: Allows dynamic resizing of resource pool (independent of application logic) based on system load and makes use of light-weight task dispatch

Related Work: Data Management

- [*Beynon01*]: **DataCutter**
- [*Ranganathan03*]: **Simulations**
- [*Ghemawat03,Dean04,Chang06*]: **BigTable**, **GFS**, **MapReduce**
- [*Liu04*]: **GridDB**
- [*Chervenak04,Chervenak06*]: **RLS** (Replica Location Service), **DRS** (Data Replication Service)
- [*Tatebe04,Xiaohui05*]: **GFarm**
- [*Branco04,Adams06*]: **DIAL/ATLAS**

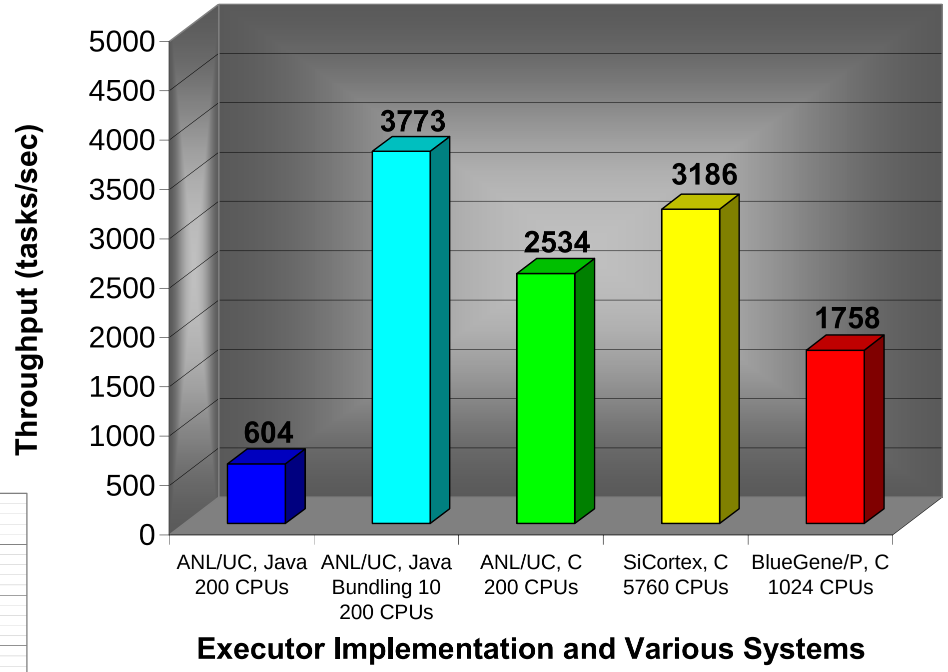
Conclusion: Our work focuses on the co-location of storage and computations close to each other (i.e. on the same physical resource) while operating in a dynamic environment.

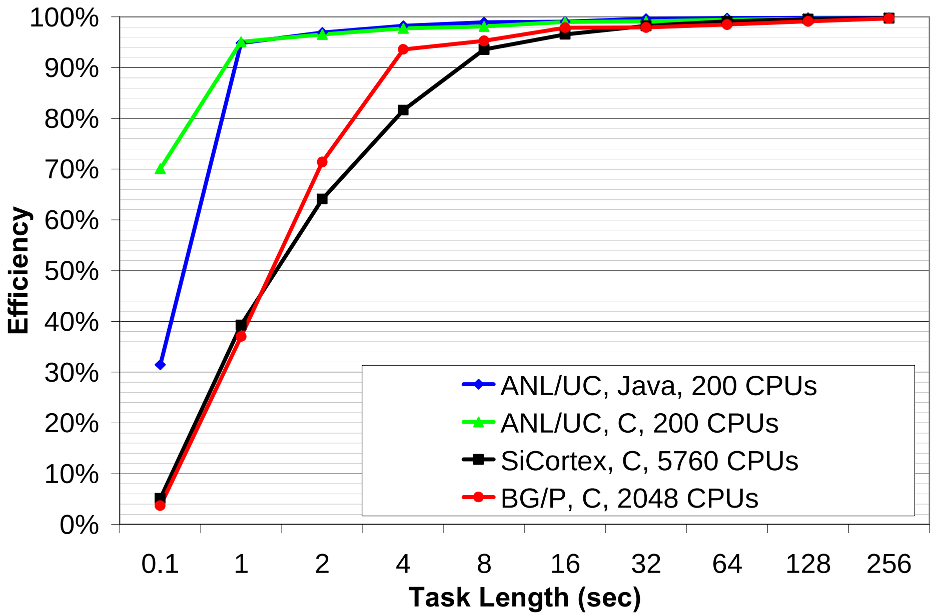
Results

- Abstract task farm model *[Dissertation Proposal 2007]*
- Practical Realization: Falkon
  – Task Dispatcher *[Globus Incubator 2007, SC07, SC08]*
  – Resource Provisioning *[SC07, TG07]*
  – Data Diffusion *[NSF06, MSES07, DADC08]*
  – Swift Integration *[SWF07, NOVA08, SWF08, GW08]*
- Applications *[NASA06, TG06, SC06, NASA07, SWF07, NOVA08, SC08]*
  – Astronomy, medical imaging, molecular dynamics (chemistry and pharmaceuticals), economic modeling

Dispatcher Throughput

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

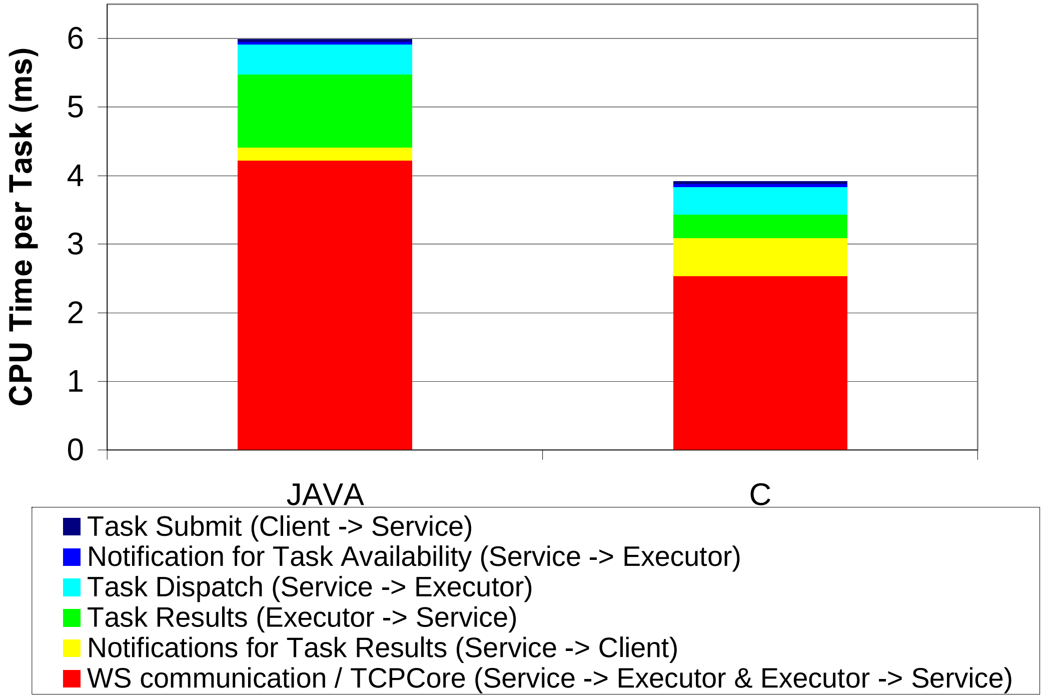




- Efficient:
  - High efficiency with second long tasks on 1000s of processors

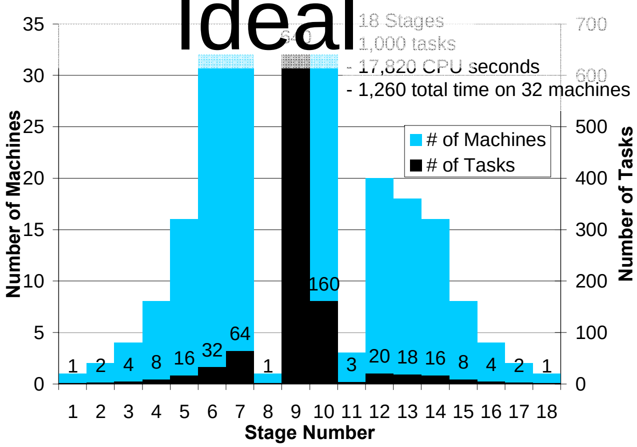
Dispatcher Performance Profiling

- **GT:** Java WS-Core 4.0.4
- **Java:** Sun JDK 1.6
- **Machine Hardware:** Dual Xeon 3GHz CPUs with HT
- **Machine OS:** Linux 2.6.13-15.16-smp
- **Executors Location:** ANL/UC TG Site, 100 dual CPU Xeon/Itanium nodes, ~2ms latency
- **Workload:** 10000 tasks, “/bin/sleep 0”

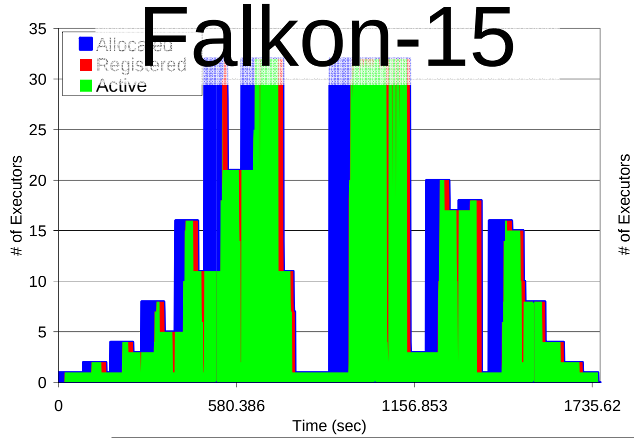


Resource Provisioning

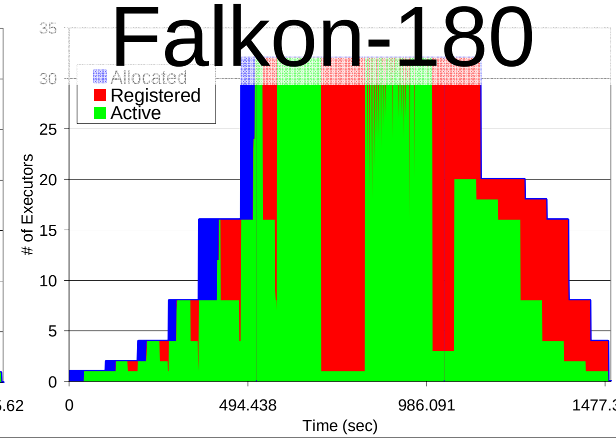
Ideal



Falkon-15



Falkon-180



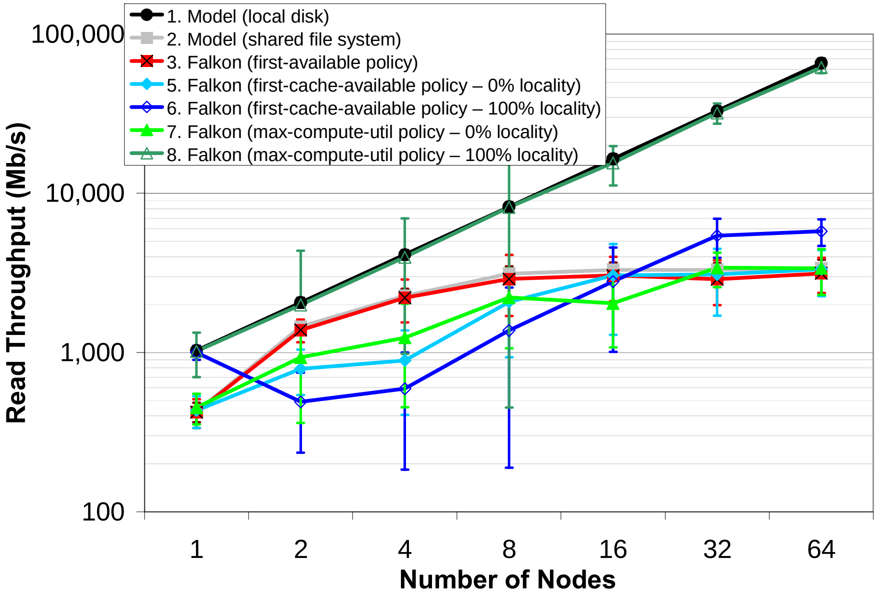
- End-to-end execution time:
  - 1260 sec in ideal case
  - 4904 sec → 1276 sec
- Average task queue time:
  - 42.2 sec in ideal case
  - 611 sec → 43.5 sec
- Trade-off:
  - Resource Utilization for Execution Efficiency

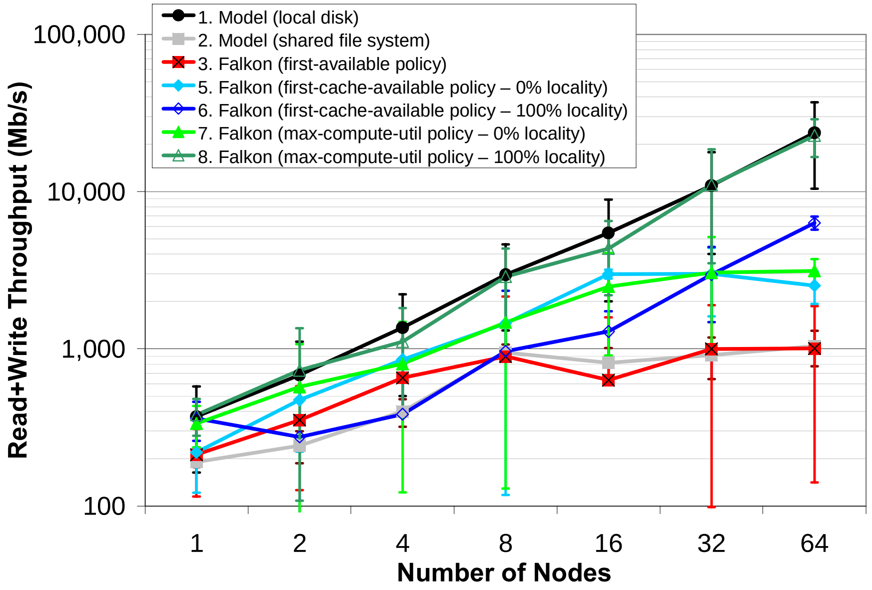
	GRAM +PBS	Falkon-15	Falkon-60	Falkon-120	Falkon-180	Falkon-∞	Ideal (32 nodes)
Queue Time (sec)	611.1	87.3	83.9	74.7	44.4	43.5	42.2
Execution Time (sec)	56.5	17.9	17.9	17.9	17.9	17.9	17.8
Execution Time %	8.5%	17.0%	17.6%	19.3%	28.7%	29.2%	29.7%

	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

Data Diffusion

- No Locality
  - Modest loss of read performance for small # of nodes (<8)
  - Comparable performance with large # of nodes
  - Modest gains in read+write performance
- Locality
  - Significant gains in performance beyond 8 nodes
  - Data-aware scheduler achieves near optimal performance and scalability

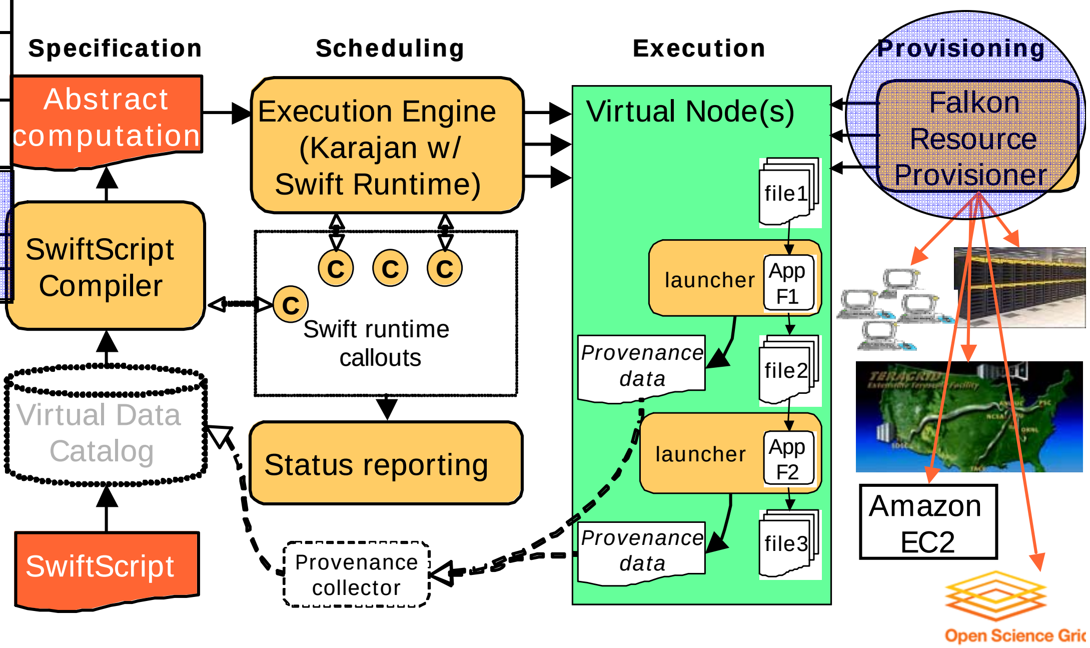




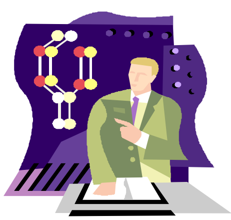
Falkon Integration with Swift

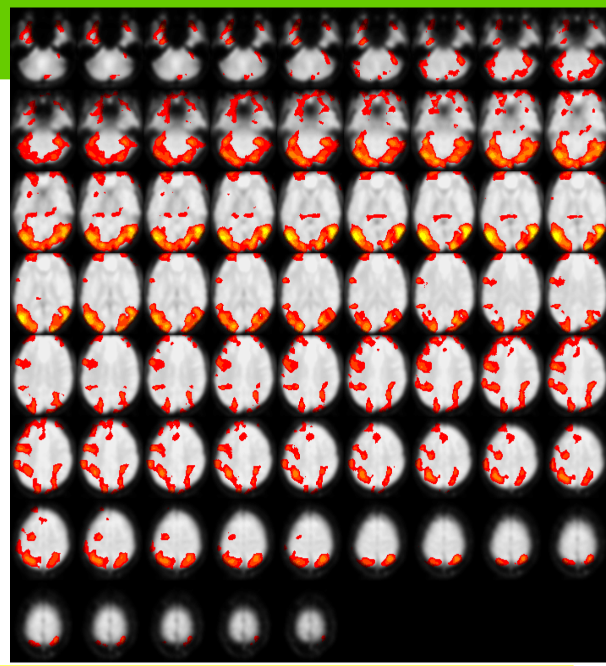
Application	#Tasks/workflow	#Stages
ATLAS: High Energy Physics Event Simulation	500K	1
fMRI DBIC: AIRSN Image Processing	100s	12
FOAM: Ocean/Atmosphere Model	2000	3
GADU: Genomics	40K	4
HNL: fMRI Aphasia Study	500	4
NVO/NASA: Photorealistic Montage/Morphology	1000s	16
QuarkNet/I2U2: Physics Science Education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

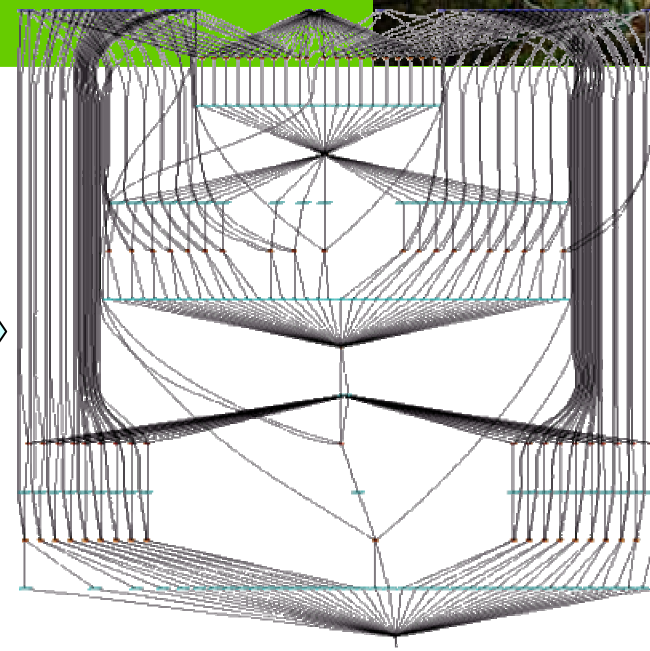
Swift Architecture

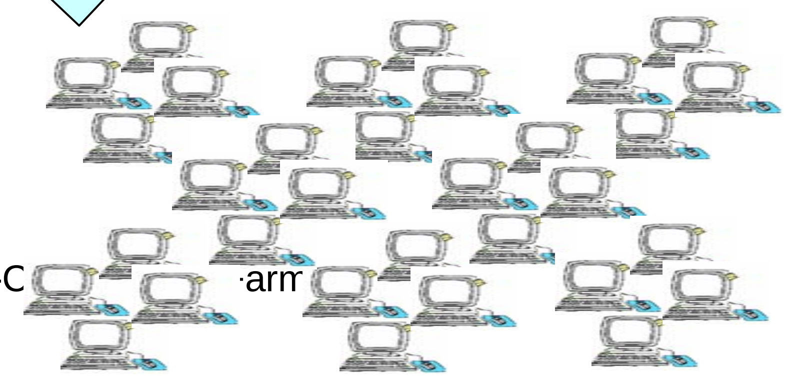


Functional MRI (fMRI)





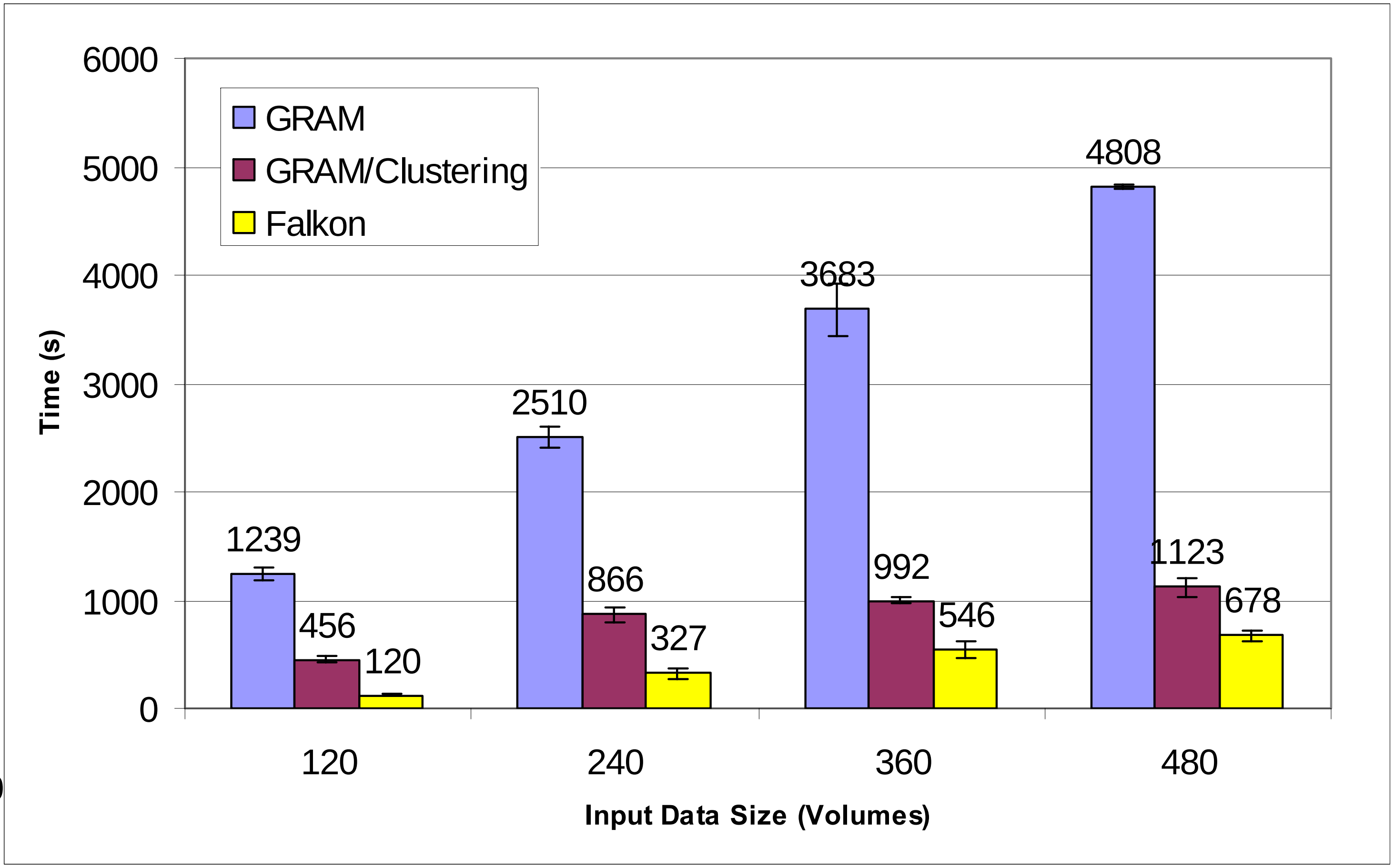


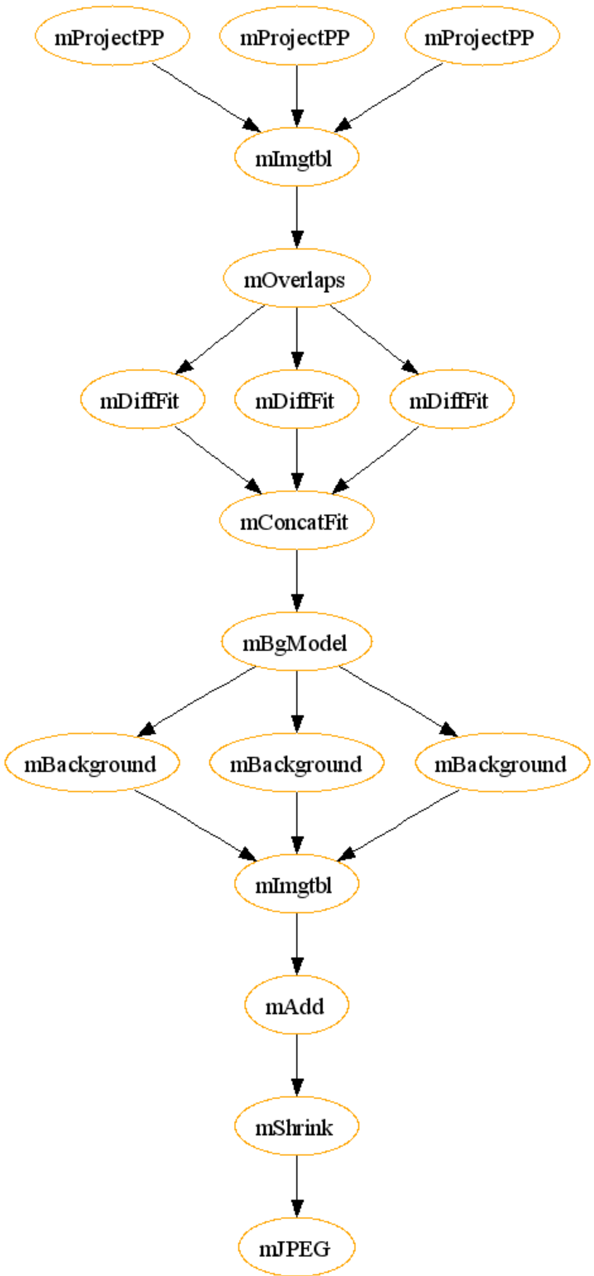


- Wide range of analyses
  - Testing, interactive analysis, production runs
  - Data mining
  - Parameter studies

fMRI Application

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



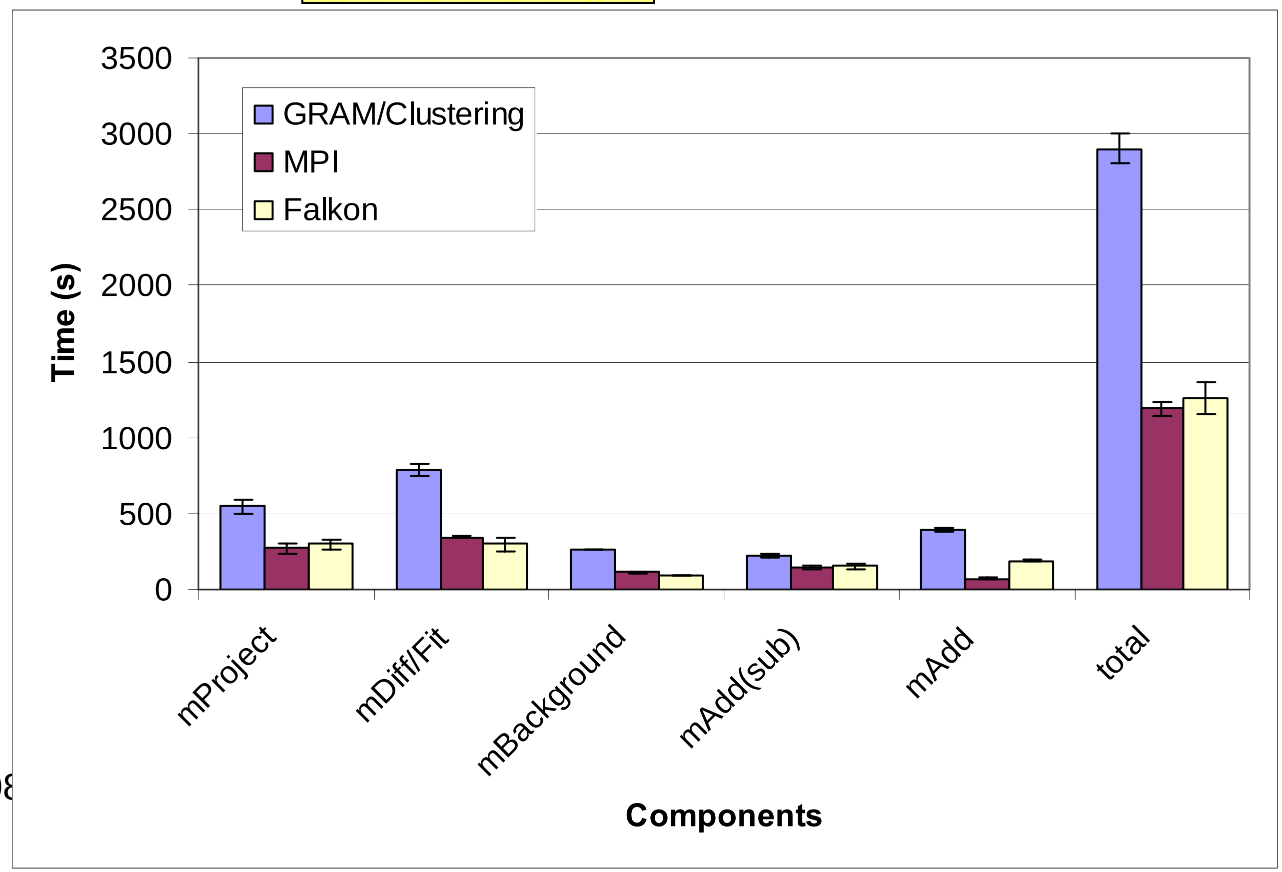


Montage

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

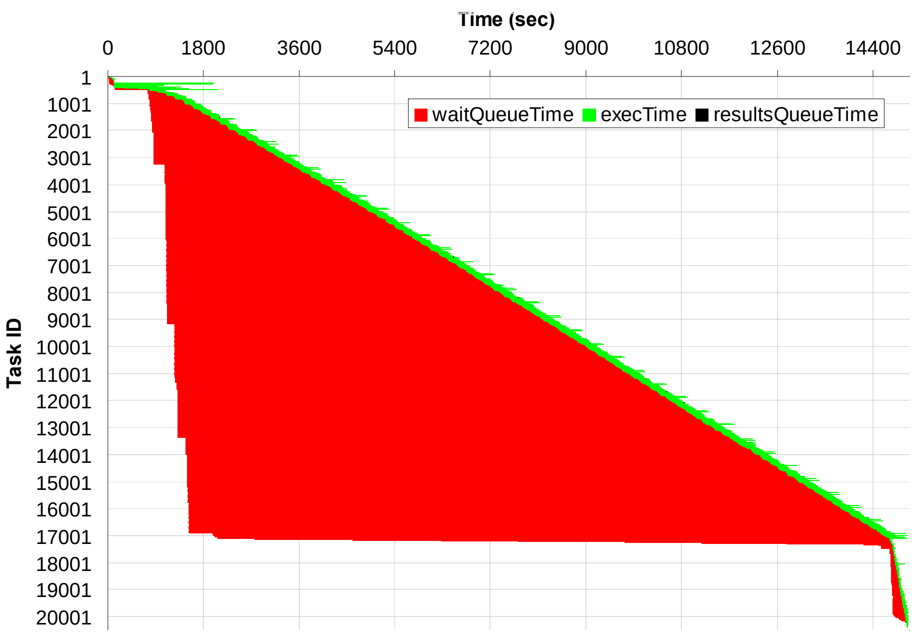
Montage Application

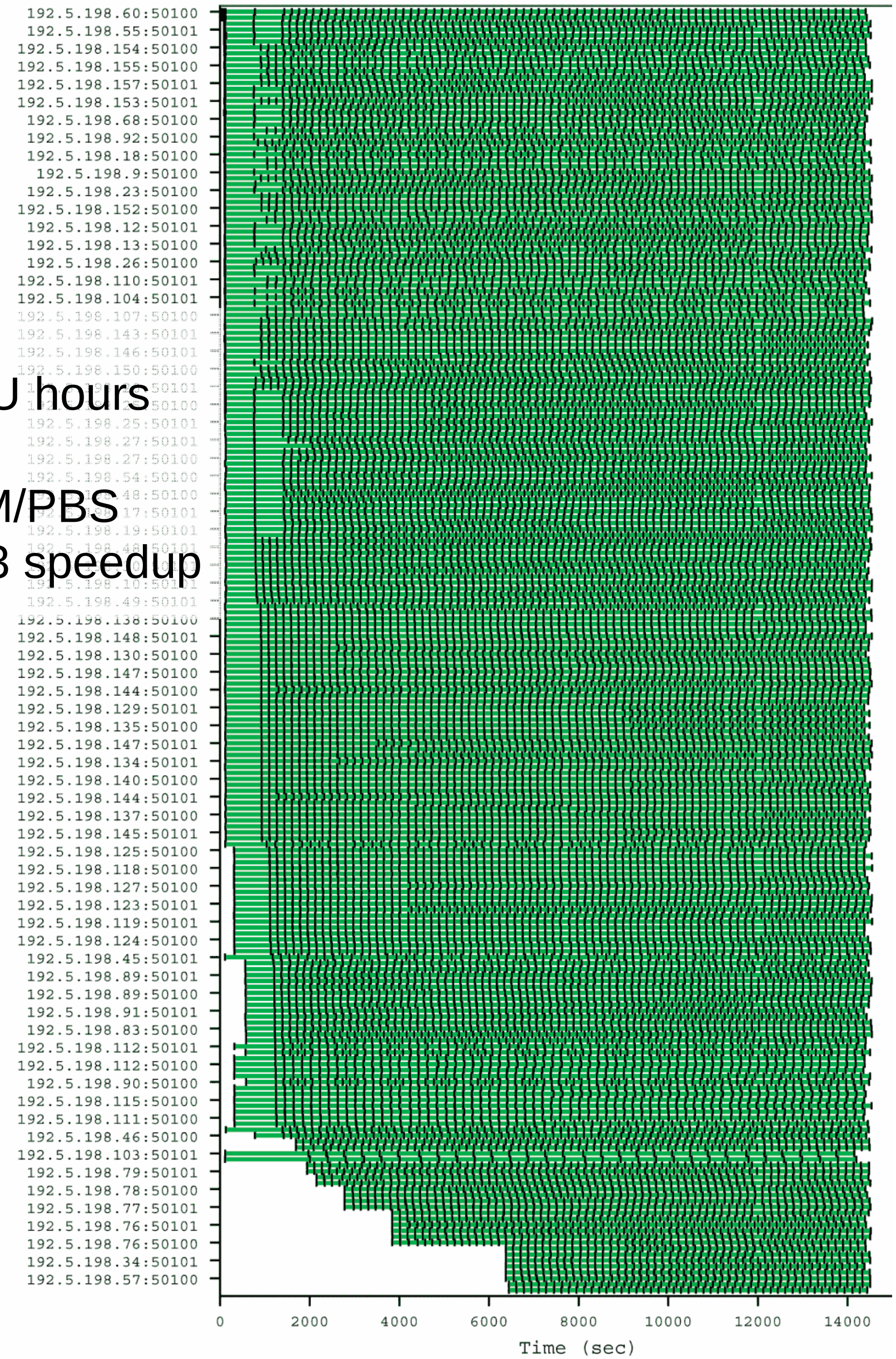
- GRAM/Clustering vs. Falkon: 57% lower application run time
- MPI* vs. Falkon: 4% higher application run time
- * MPI should be lower bound



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

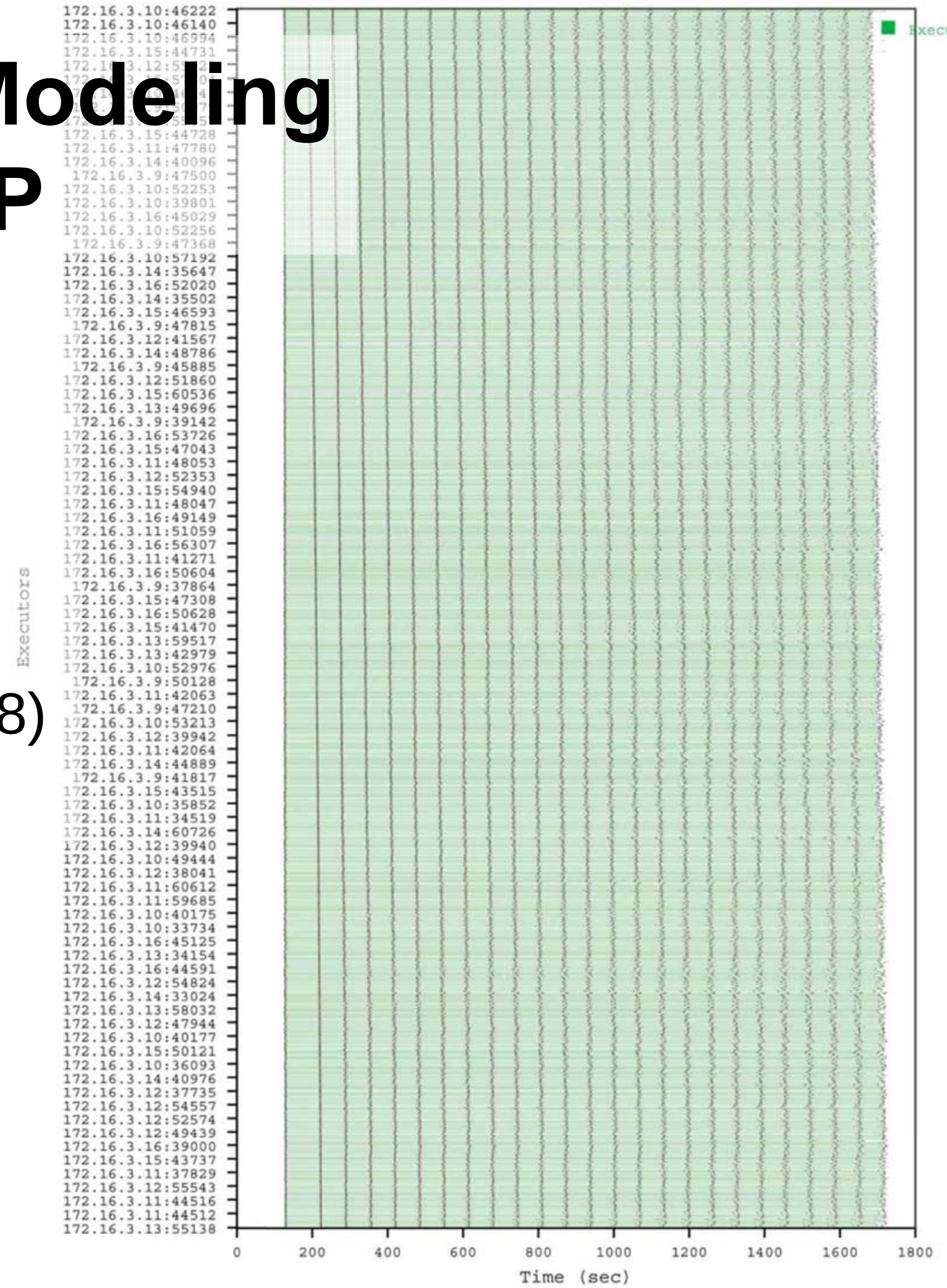




MARS Economic Modeling on IBM BG/P

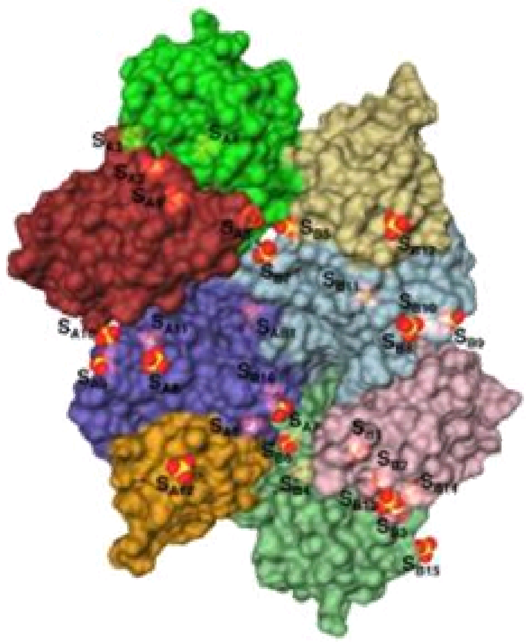
- CPU Cores: 2048
- Tasks: 49152
- Micro-tasks: 7077888
- Elapsed time: 1601 secs
- CPU Hours: 894
- Speedup: 1993X (ideal 2048)
- Efficiency: 97.3%

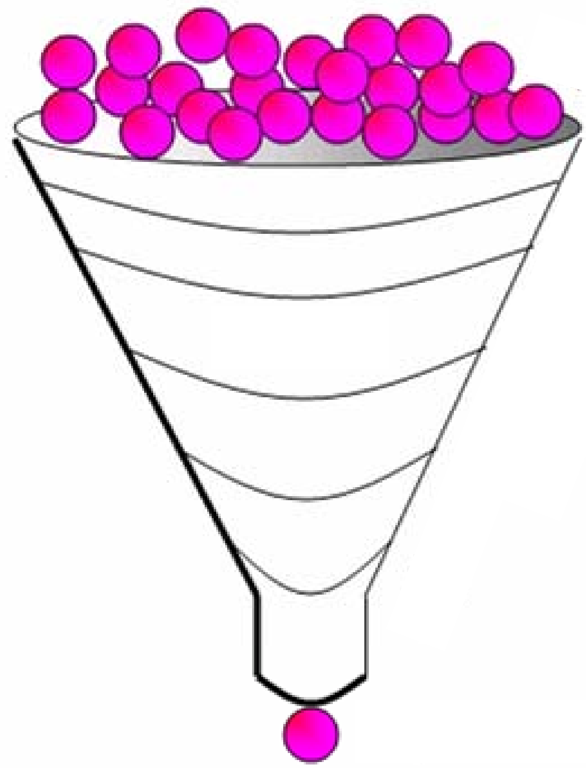
rid Reso



Many Many Tasks: Identifying Potential Drug Targets

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

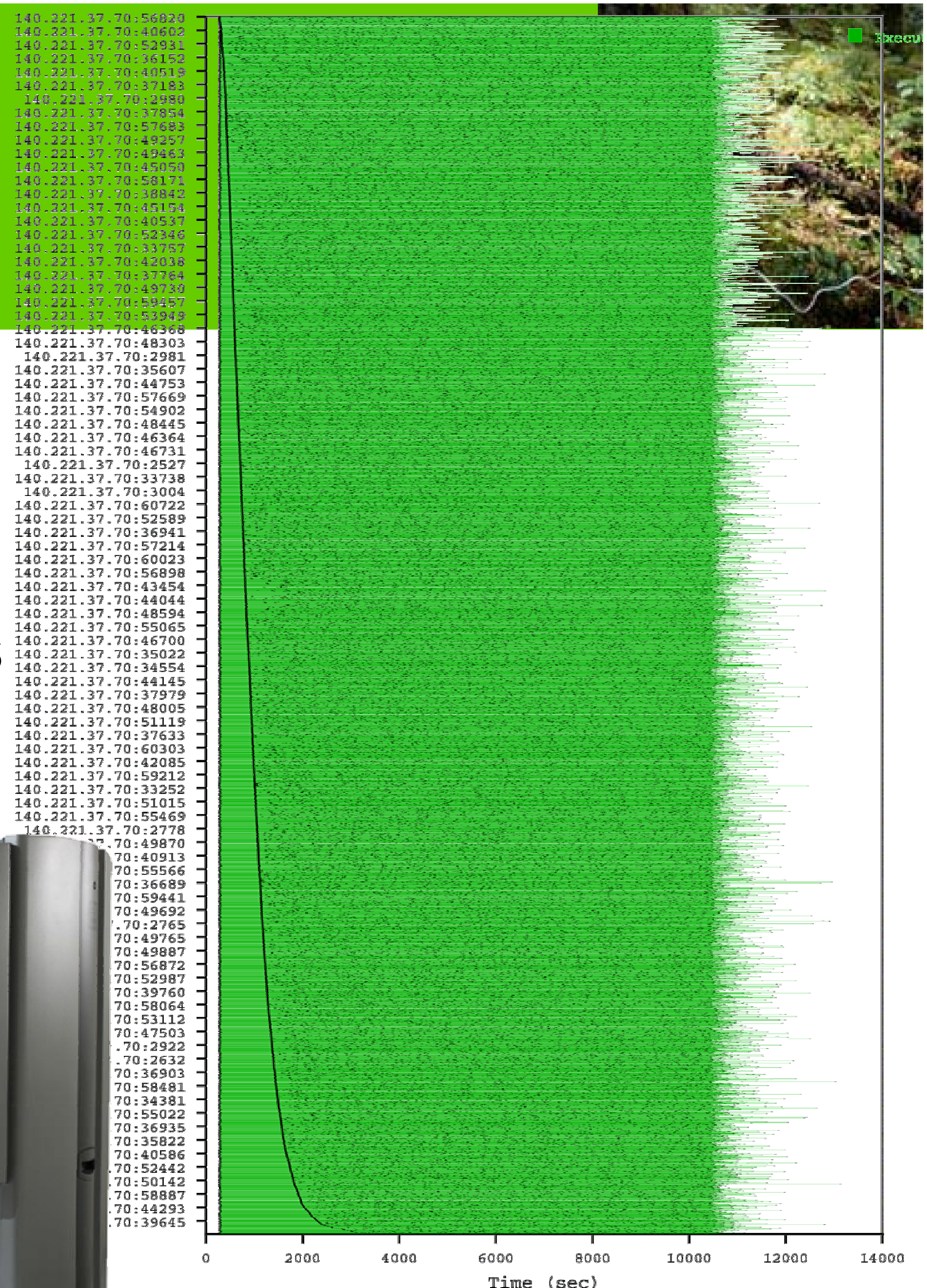




(Mike Kubal, Benoit Roux, and others)

DOCK on SiCortex

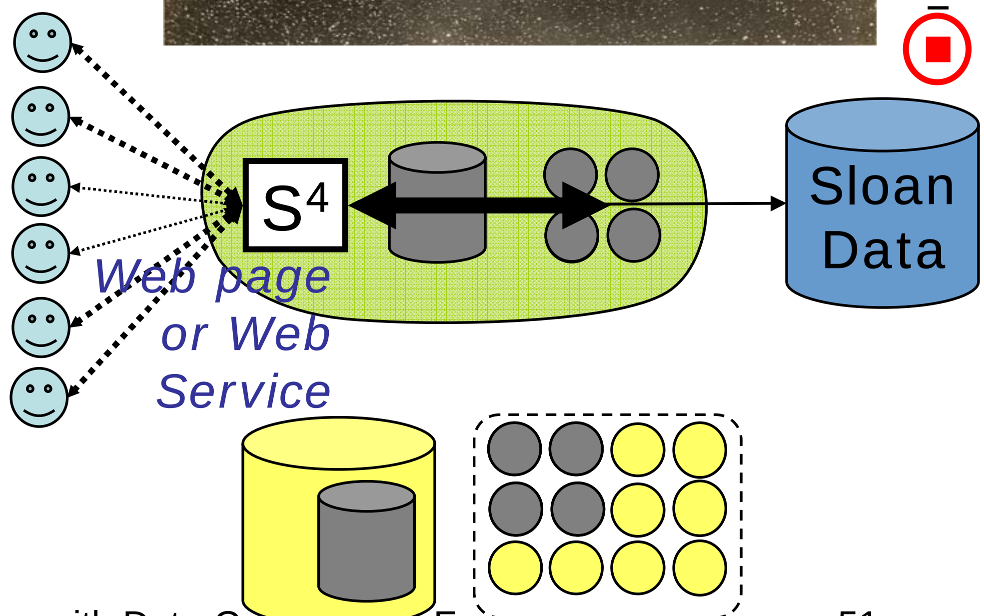
- CPU cores: 5760
- Tasks: 92160
- Elapsed time: 12821 sec
- Compute time: 1.94 CPU years
- Average task time: 660.3 sec
- Speedup: 5650X (ideal 5760)
- Efficiency: 98.2%



AstroPortal Stacking Service

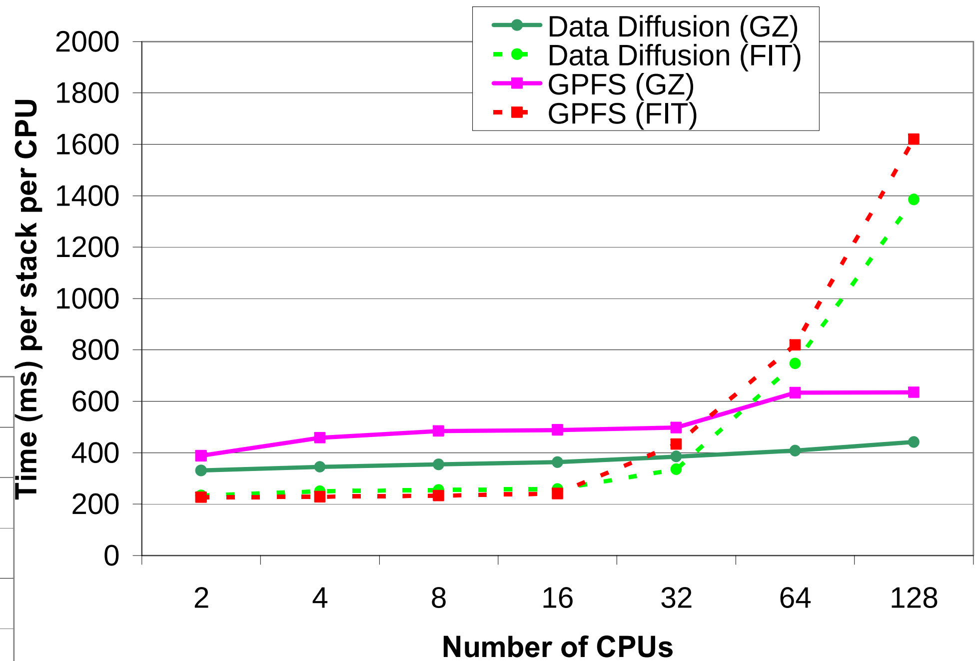
- Purpose
  - On-demand "stacks" of random locations within ~10TB dataset
- Challenge
  - Rapid access to 10-10K "random" files
  - Time-varying load
- Sample Workloads

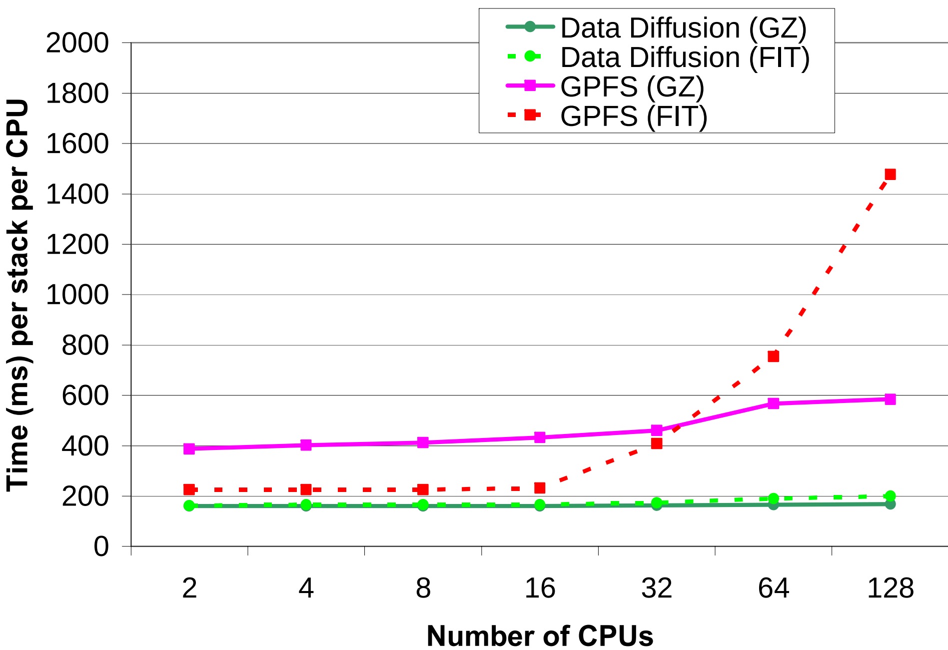
Locality	Number of Objects	Number of Files
1	111700	111700
1.38	154345	111699
2	97999	49000
3	88857	29620
4	76575	19145
5	60590	12120
10	46480	4650
20	40460	2025
30	23695	790



AstroPortal Stacking Service with Data Diffusion

Low data locality →



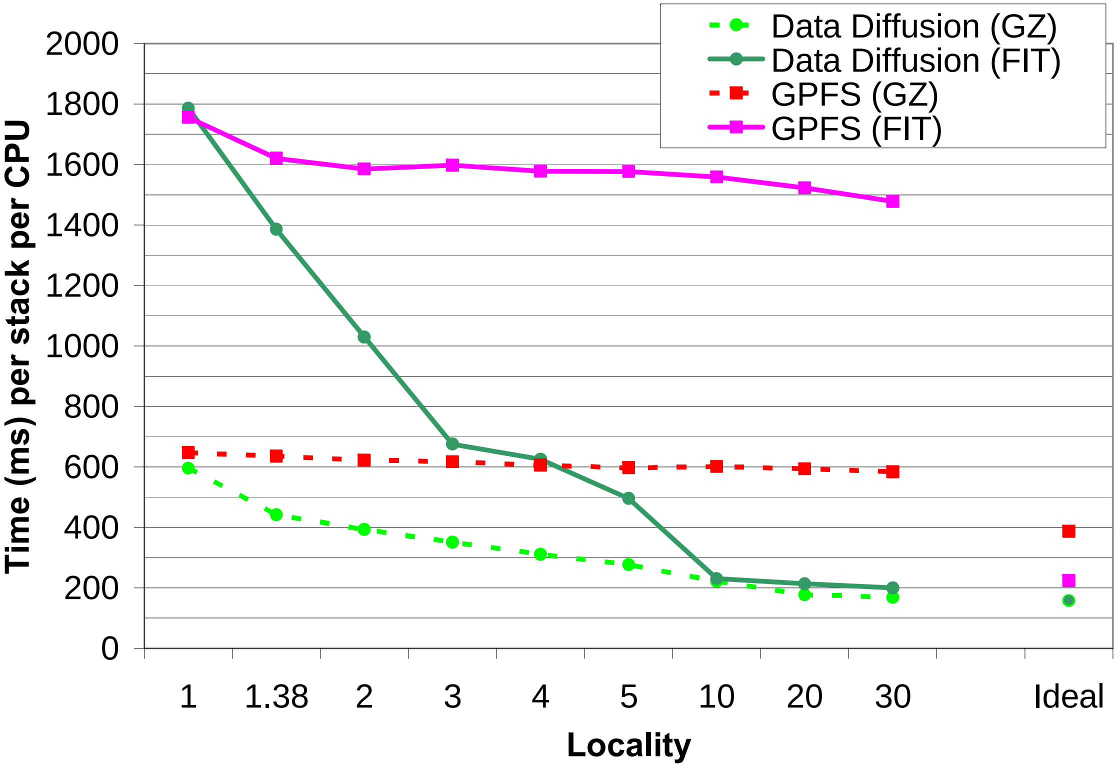


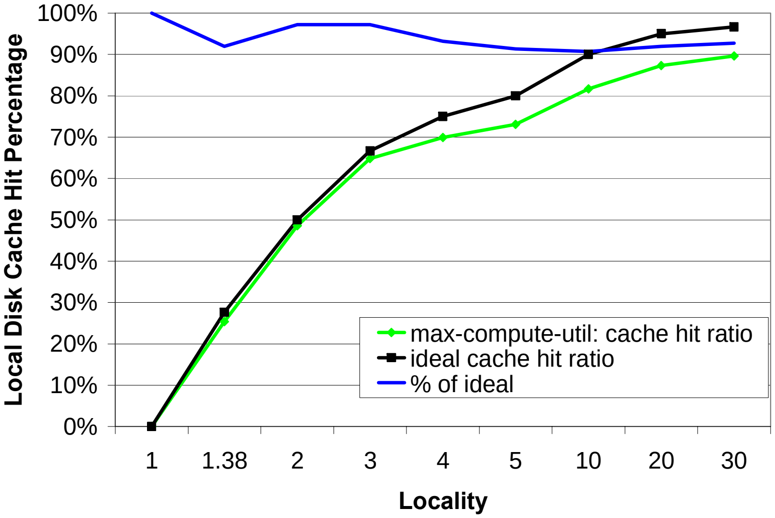
← High data locality

– Near perfect scalability

AstroPortal Stacking Service with Data Diffusion

- Big performance gains as locality increases

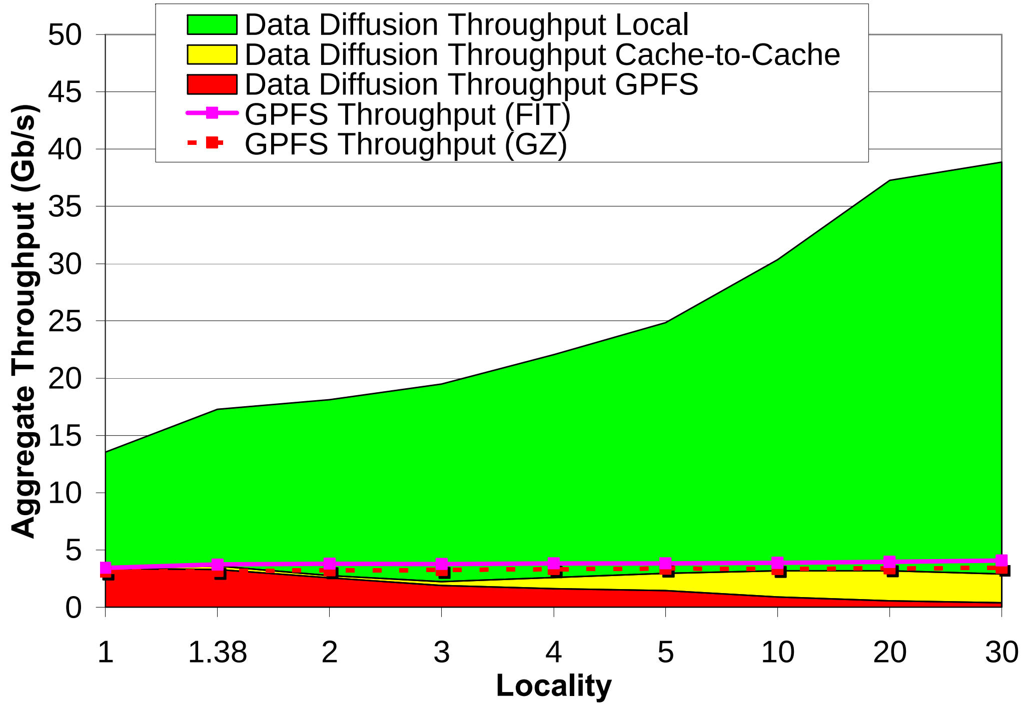


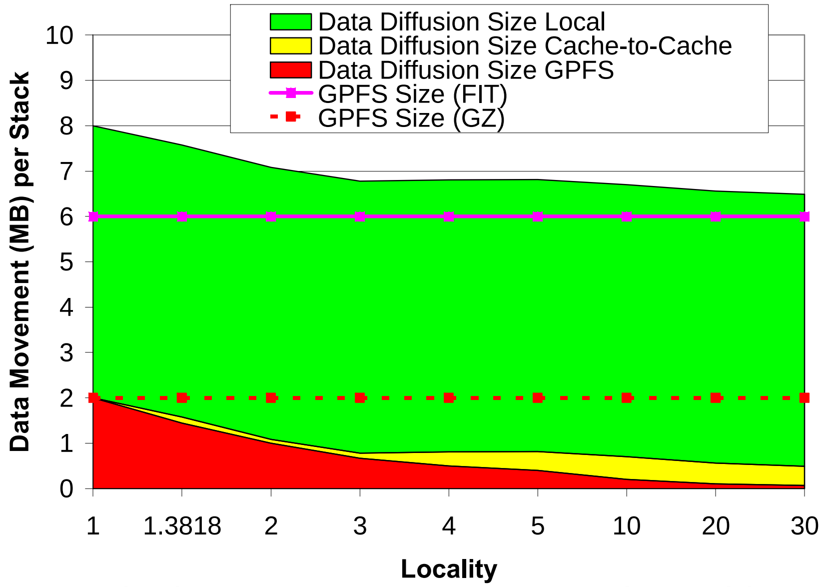


- 90%+ cache hit ratios

AstroPortal Stacking Service with Data Diffusion

- Aggregate throughput:
  - 39Gb/s
  - 10X higher than GPFS

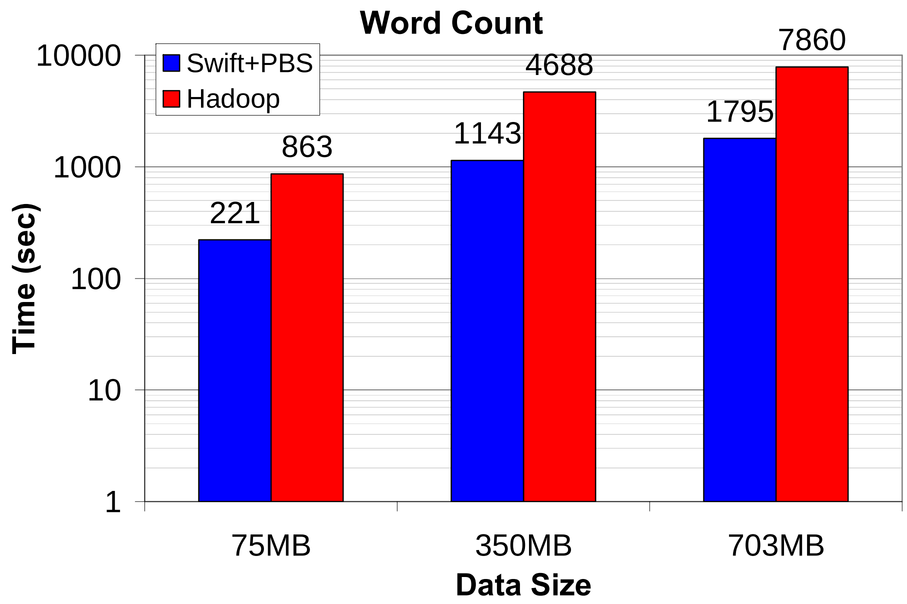


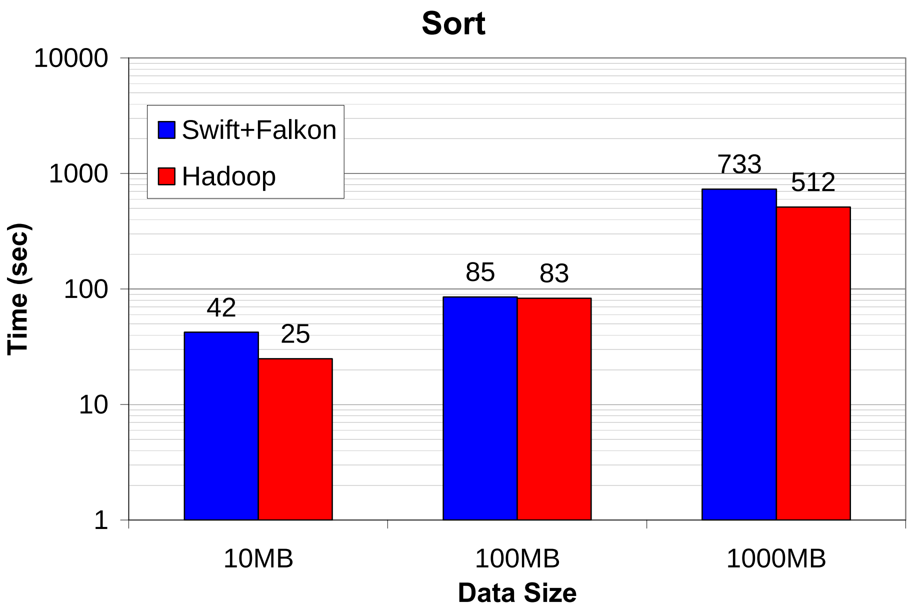


- Reduced load on GPFS
  - 0.49Gb/s
  - 1/10 of the original load

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

Conclusions & Contributions

- Defined an *abstract model for performance efficiency of data analysis workloads* using data-centric task farms
- Provide a reference implementation (Falkon)
  - Use a streamlined dispatcher to increase task throughput by several orders of magnitude over traditional LRMs
  - Use multi-level scheduling to reduce perceived wait queue time for tasks to execute on remote resources
  - Address data diffusion through co-scheduling of storage and computational resources to improve performance and scalability
  - Provide the benefits of dedicated hardware without the associated high cost
  - Show flexibility/effectiveness on real world applications
    - Astronomy, medical imaging, molecular dynamics (chemistry and pharmaceuticals), economic modeling
  - Runs on real systems with 1000s of processors:
    - TeraGrid, IBM BlueGene/P, SiCortex

More Information

- More information: http://people.cs.uchicago.edu/~iraicu/
- Related Projects:
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - AstroPortal: http://people.cs.uchicago.edu/~iraicu/projects/Falkon/astro_portal.htm
  - 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
  - Rick Stevens, The University of Chicago & Argonne National Laboratory
  - 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

Proposals / Journal Articles Book Chapters / Conference Workshop Articles (selected)

1. Yong Zhao, Ioan Raicu, Ian Foster. “Scientific Workflow Systems for 21st Century e-Science, New Bottle or New Wine?”, Invited Paper, to appear at IEEE Workshop on Scientific Workflows 2008.
2. Ioan Raicu, Yong Zhao, Ian Foster, Alex Szalay. "Accelerating Large-scale Data Exploration through Data Diffusion", to appear at International Workshop on Data-Aware Distributed Computing 2008.
3. Ioan Raicu, Yong Zhao, Ian Foster, Mike Wilde, Zhao Zhang, Ben Clifford, Mihael Hategan, Sarah Kenny. “Managing and Executing Loosely Coupled Large Scale Applications on Clusters, Grids, and Supercomputers”, to appear at GlobusWorld08, part of Open Source Grid and Cluster Conference 2008.
4. Yong Zhao, Ioan Raicu, Ian Foster, Mihael Hategan, Veronika Nefedova, Mike Wilde. “Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments”, to appear as a book chapter in Grid Computing Research Progress, ISBN: 978-1-60456-404-4, Nova Publisher 2008.
5. Ioan Raicu. “Harnessing Grid Resources with Data-Centric Task Farms”, University of Chicago, Computer Science Department, PhD Proposal, December 2007, Chicago, Illinois.
6. Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster and Mike Wilde. “Falkon: A Proposal for Project Globus Incubation”, Globus Incubation Management Project, 2007 – Proposal accepted 11/10/07.
7. Ioan Raicu, Yong Zhao, Ian Foster, Alex Szalay. “A Data Diffusion Approach to Large Scale Scientific Exploration”, to appear in the Microsoft Research eScience Workshop 2007.
8. Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. “Falkon: a Fast and Light-weight tasK executiON framework”, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SuperComputing/SC), 2007.
9. Ioan Raicu, Catalin Dumitrescu, Ian Foster. “Dynamic Resource Provisioning in Grid Environments”, TeraGrid Conference 2007.
10. 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”, IEEE Workshop on Scientific Workflows 2007.
11. I. Raicu, I. Foster. “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets”, NASA GSRP Proposal, Ames Research Center, NASA, February 2006, February 2007 -- Award funded 10/1/06 - 09/30/08.
12. Ioan Raicu, Ian Foster, Alex Szalay. “Harnessing Grid Resources to Enable the Dynamic Analysis of Large Astronomy Datasets”, poster presentation, IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SuperComputing/SC), 2006.
13. Ioan Raicu, Ian Foster, Alex Szalay, Gabriela Turcu. “AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis”, TeraGrid Conference 2006, June 2006.
14. Alex Szalay, Julian Bunn, Jim Gray, Ian Foster, Ioan Raicu. “The Importance of Data Locality in Distributed Computing Applications”, NSF Workflow Workshop 2006.