THE UNIVERSITY OF CHICAGO

Accelerating Large-Scale Data Exploration through Data Diffusion

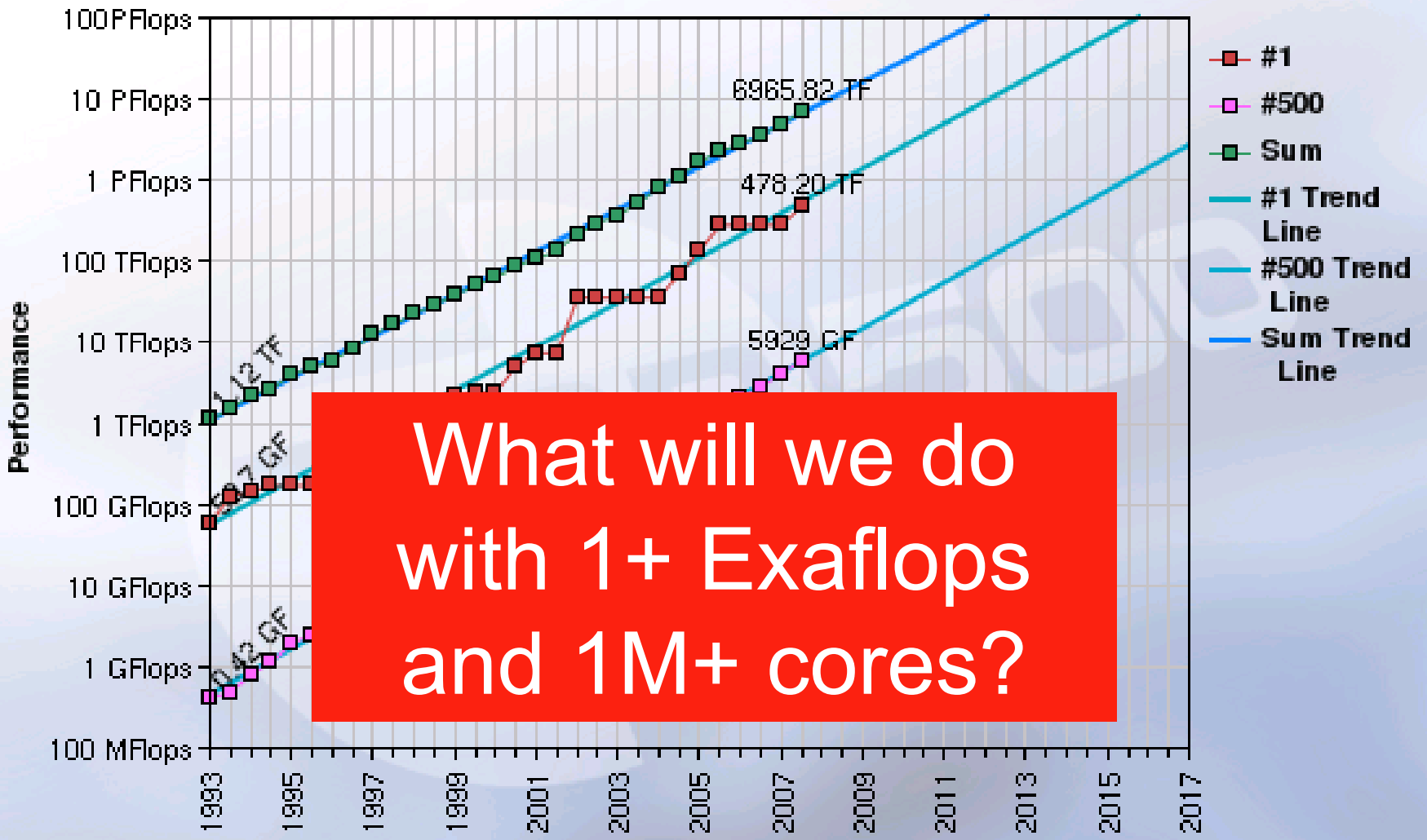
Ioan Raicu
Distributed Systems Laboratory
Computer Science Department
University of Chicago

In Collaboration with:
Yong Zhao, Microsoft Corporation
Ian Foster, University of Chicago and Argonne National Laboratory
Alex Szalay, The Johns Hopkins University

Argonne NATIONAL LABORATORY

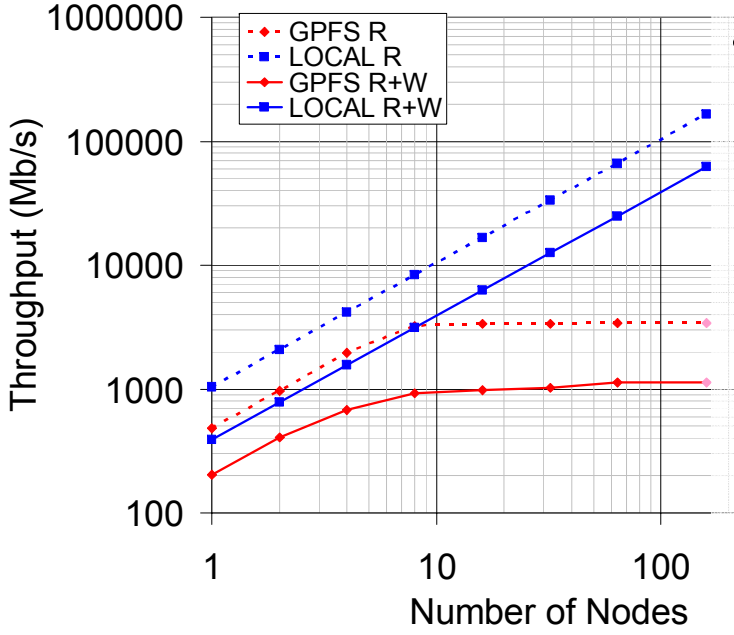
Workshop on Data-Aware Distributed Computing (DADC'08)
June 24th, 2008

Projected Performance Development



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

Storage Resource Scalability



- 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**
    - **11Mb/s per CPU vs. 515Mb/s per CPU**
  - 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**

- IBM BlueGene/P
  - 160K CPU cores
  - 78GB/s peak SAN read rates; GPFS 8GB/s sustained read rates (16 servers)
  - ~3.9Mb/s per CPU core peak; ~0.4Mb/s per CPU core sustained
  - Experiments on 4K CPU BG/P achieved 0.3Mb/s per CPU core
  - Experiments on 5.7K CPU SiCortex achieved 0.06Mb/s per CPU core

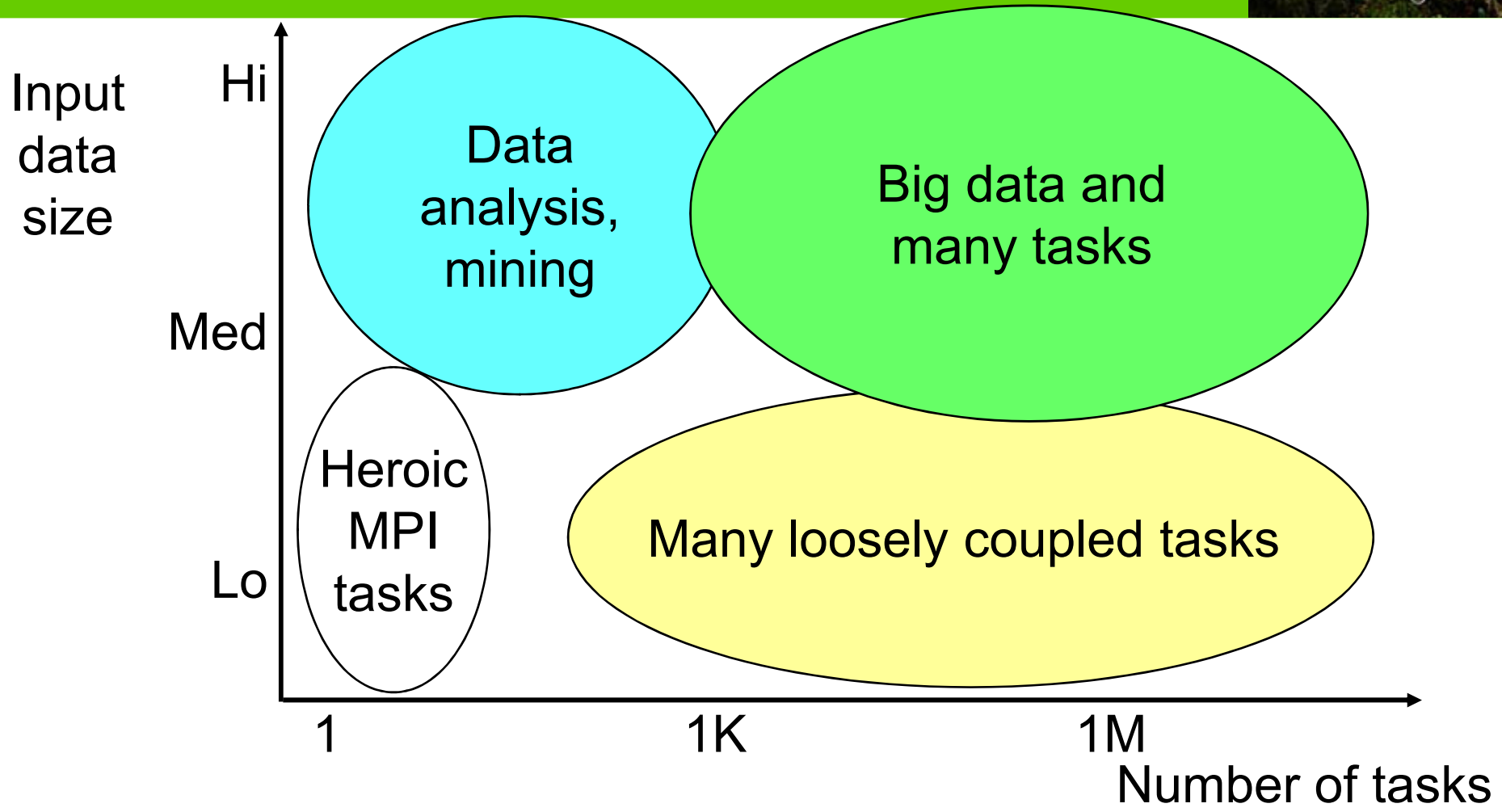
Programming Model Issues

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

Programming Model Issues

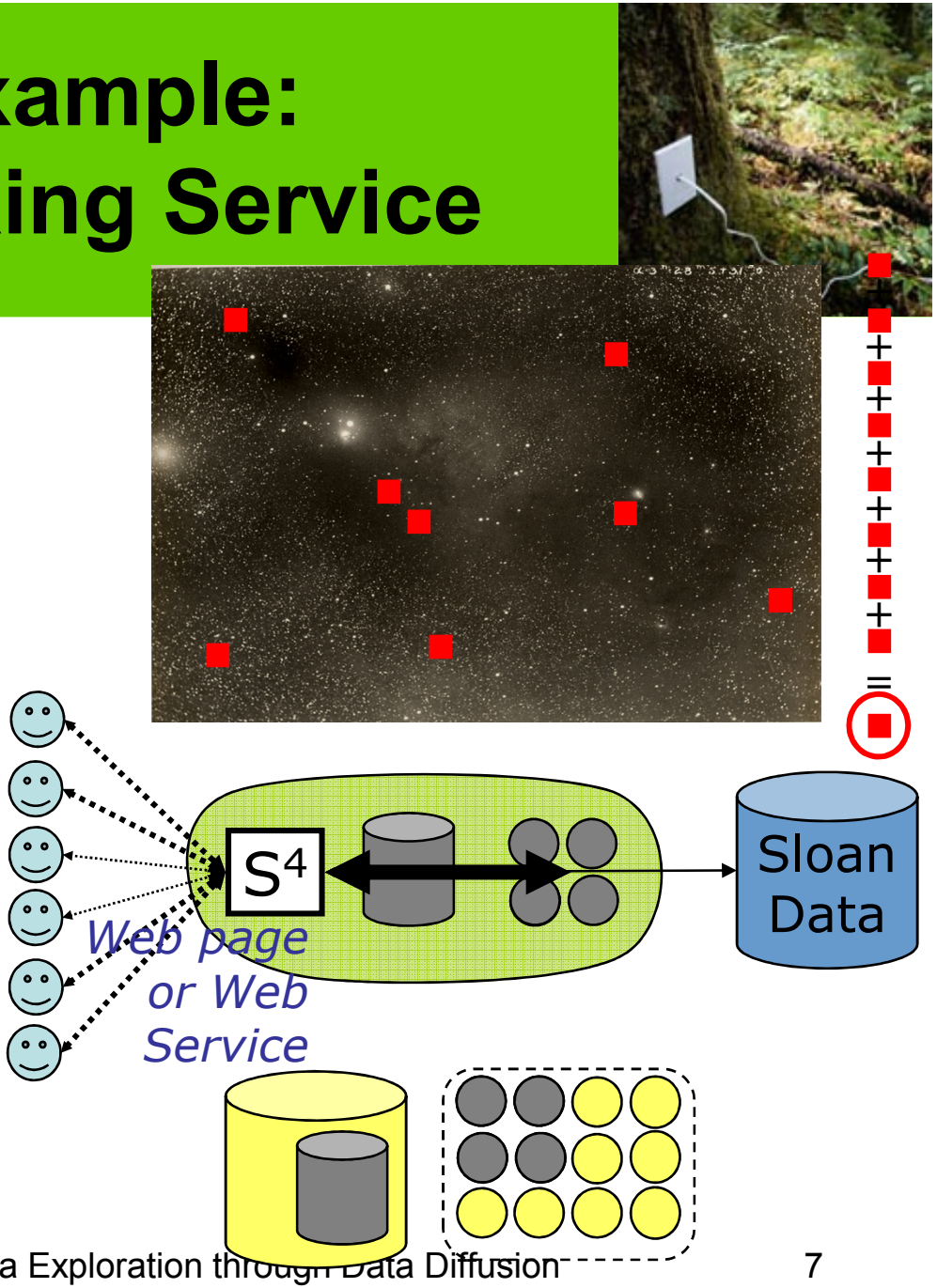
- **Multicore** processors
- Massive **task parallelism**
- Massive **data parallelism**
- Integrating **black box applications**
- Complex **task dependencies** (task graphs)
- **Failure**, and other execution management issues
- **Dynamic task graphs**
- Documenting **provenance** of data products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

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



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

Abstract Model

- 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

AMDASK: Base Definitions

- **Data Stores:** Persistent & Transient
  - Store capacity, load, ideal bandwidth, available bandwidth
- **Data Objects:**
  - Data object size, *data object's storage location(s),* copy time
- **Transient resources:** compute speed, resource state
- **Task:** application, input/output data

AMDASK: Execution Model Concepts

- Dispatch Policy
  - next-available, first-available, max-compute-util, max-cache-hit
- Caching Policy
  - random, FIFO, LRU, LFU
- Replay policy
- Data Fetch Policy
  - Just-in-Time, Spatial Locality
- Resource Acquisition Policy
  - one-at-a-time, additive, exponential, all-at-once, optimal
- Resource Release Policy
  - distributed, centralized

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**

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

- **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

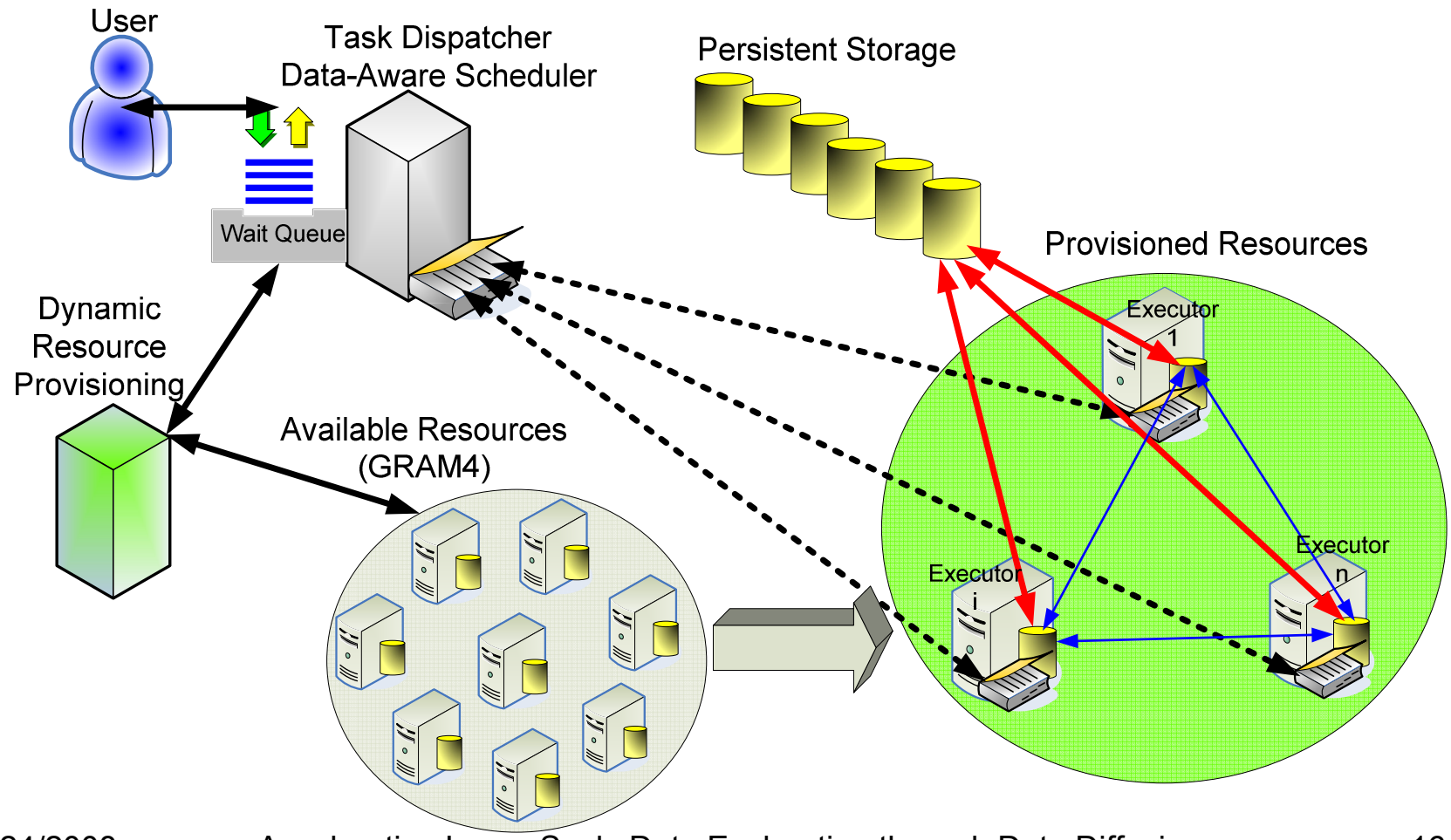
Falkon: a Fast and Light-weight tasK executiON framework

- ***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
- Integration into Swift to leverage many applications
  - Applications cover many domains: astronomy, astro-physics, medicine, chemistry, economics, climate modeling, etc

Falkon: a Fast and Light-weight tasK executiON framework

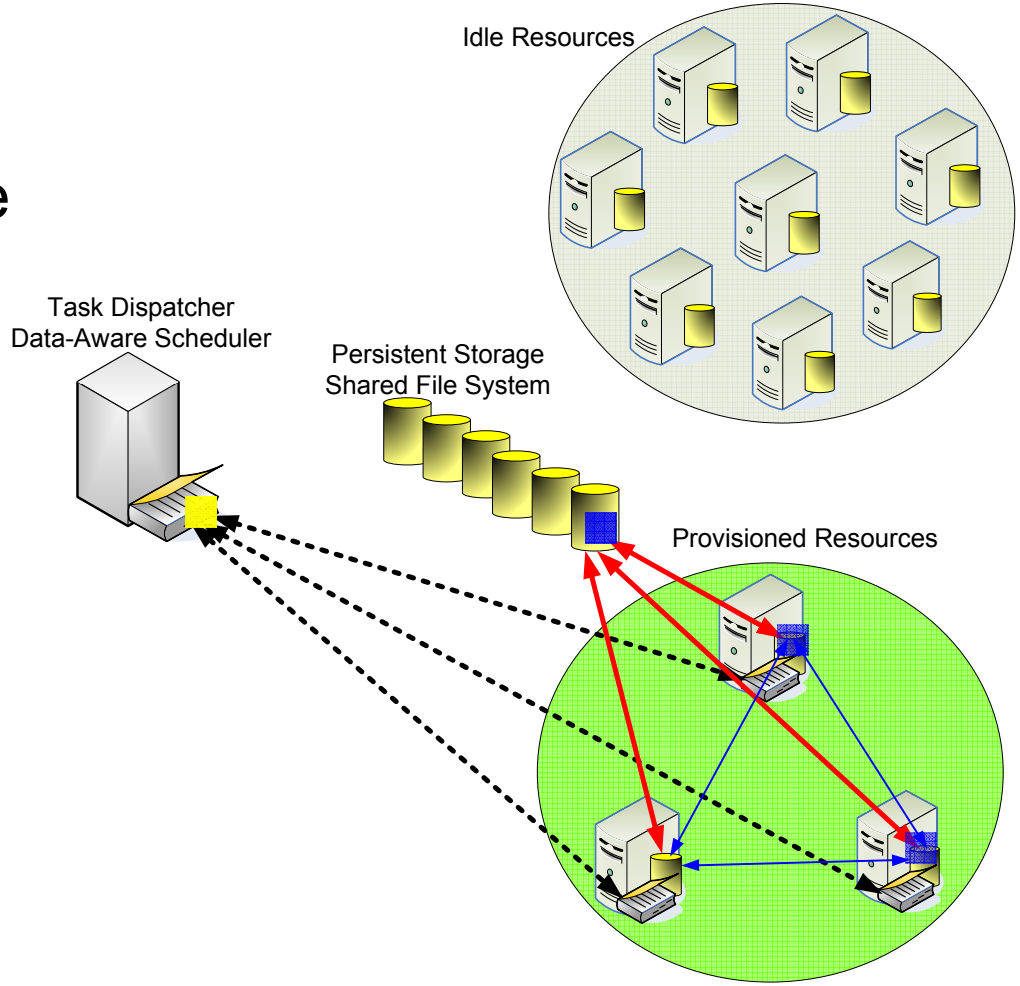
- ***Goal:*** enable the ***rapid and efficient*** execution of many independent jobs on large compute clusters
- Combines three components:
  - a ***streamlined task dispatcher***
  - ***resource provisioning*** through multi-level scheduling techniques
  - ***data diffusion*** and data-aware scheduling to leverage the co-located computational and storage resources
- Integration into Swift to leverage many applications
  - Applications cover many domains: astronomy, astro-physics, medicine, chemistry, economics, climate modeling, etc

Falkon Overview



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



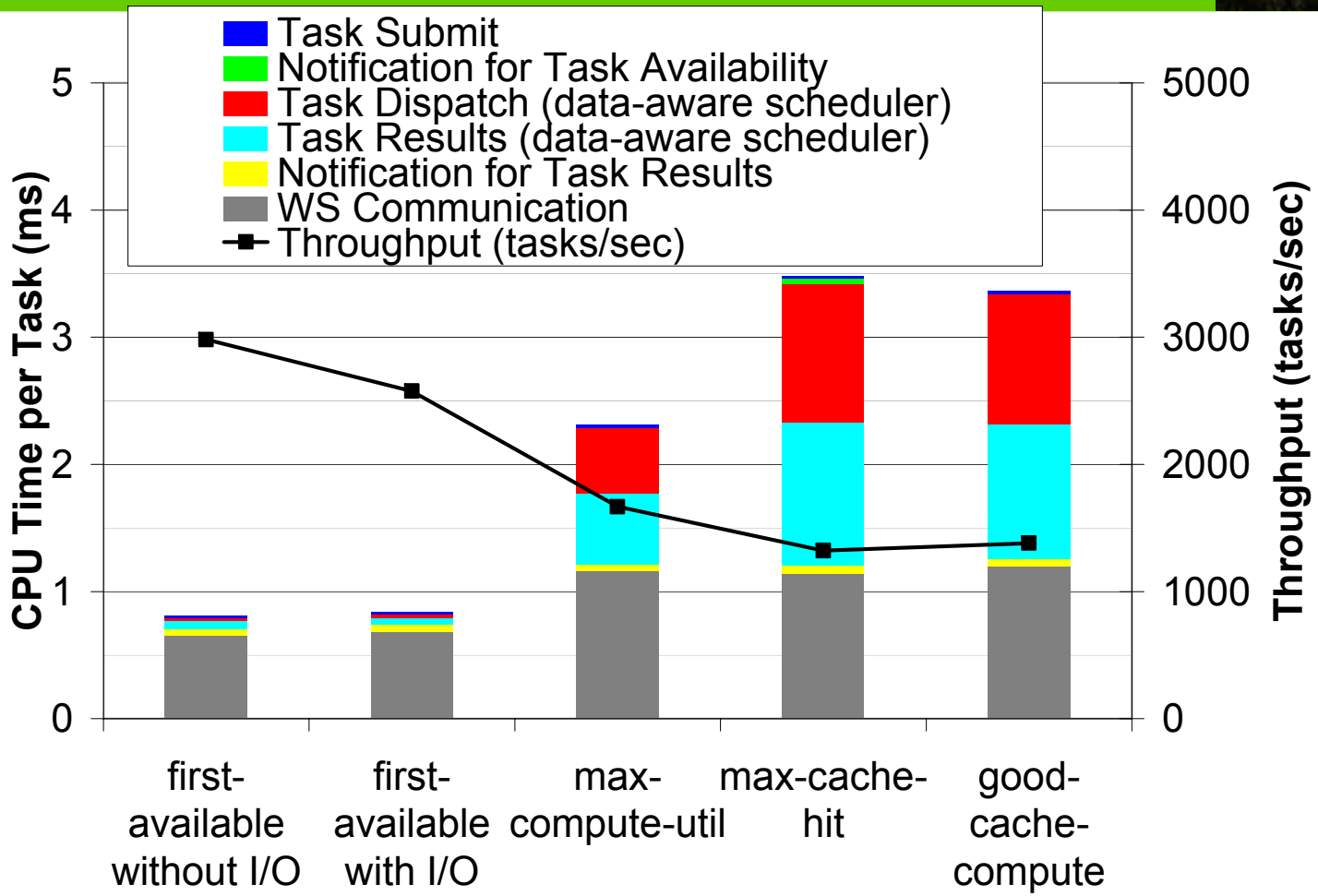
Data Diffusion

- Considers both data and computations to optimize performance
- 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

Scheduling Policies

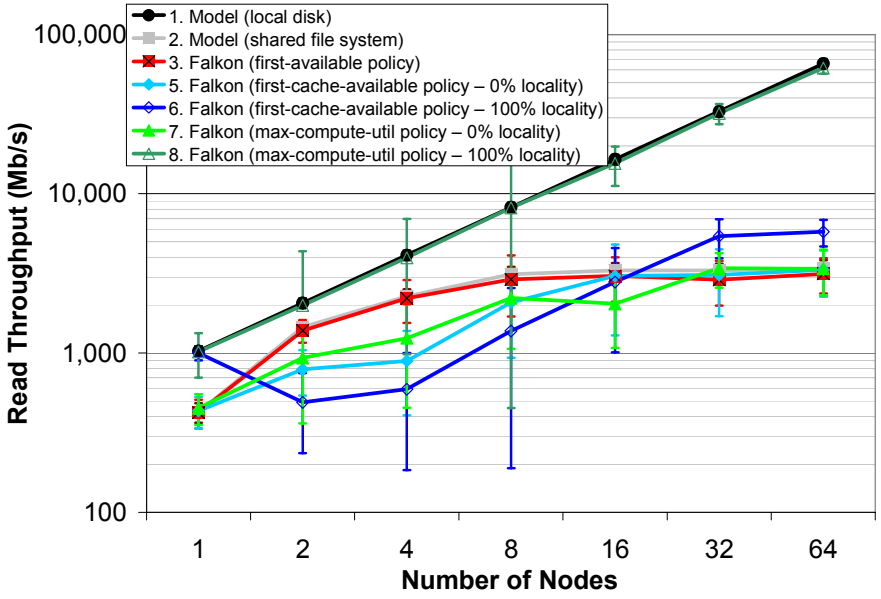
- first-available:
  - simple load balancing
- max-cache-hit
  - maximize cache hits
- max-compute-util
  - maximize processor utilization
- good-cache-compute
  - maximize both cache hit and processor utilization at the same time

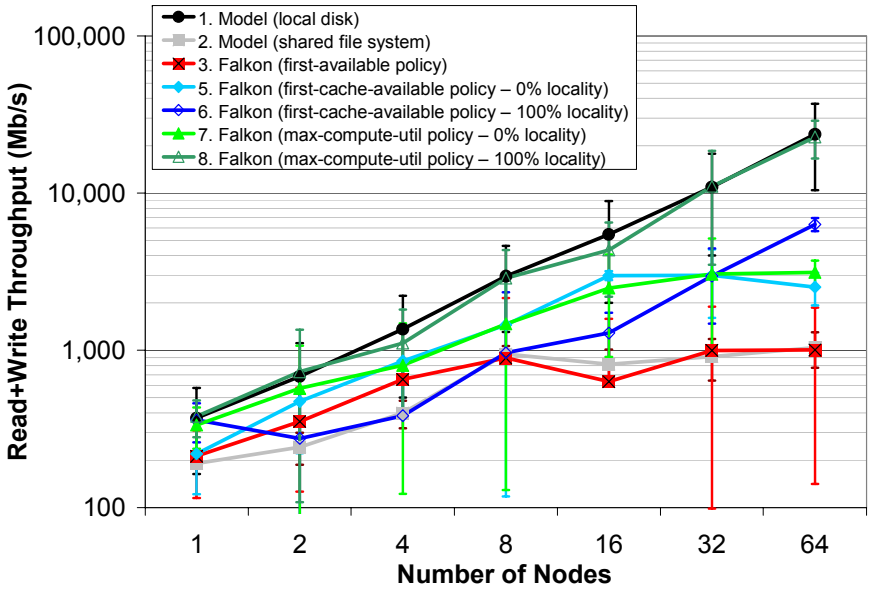
Data-Aware Scheduler Profiling



Data Diffusion: Micro-Benchmarks

- 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

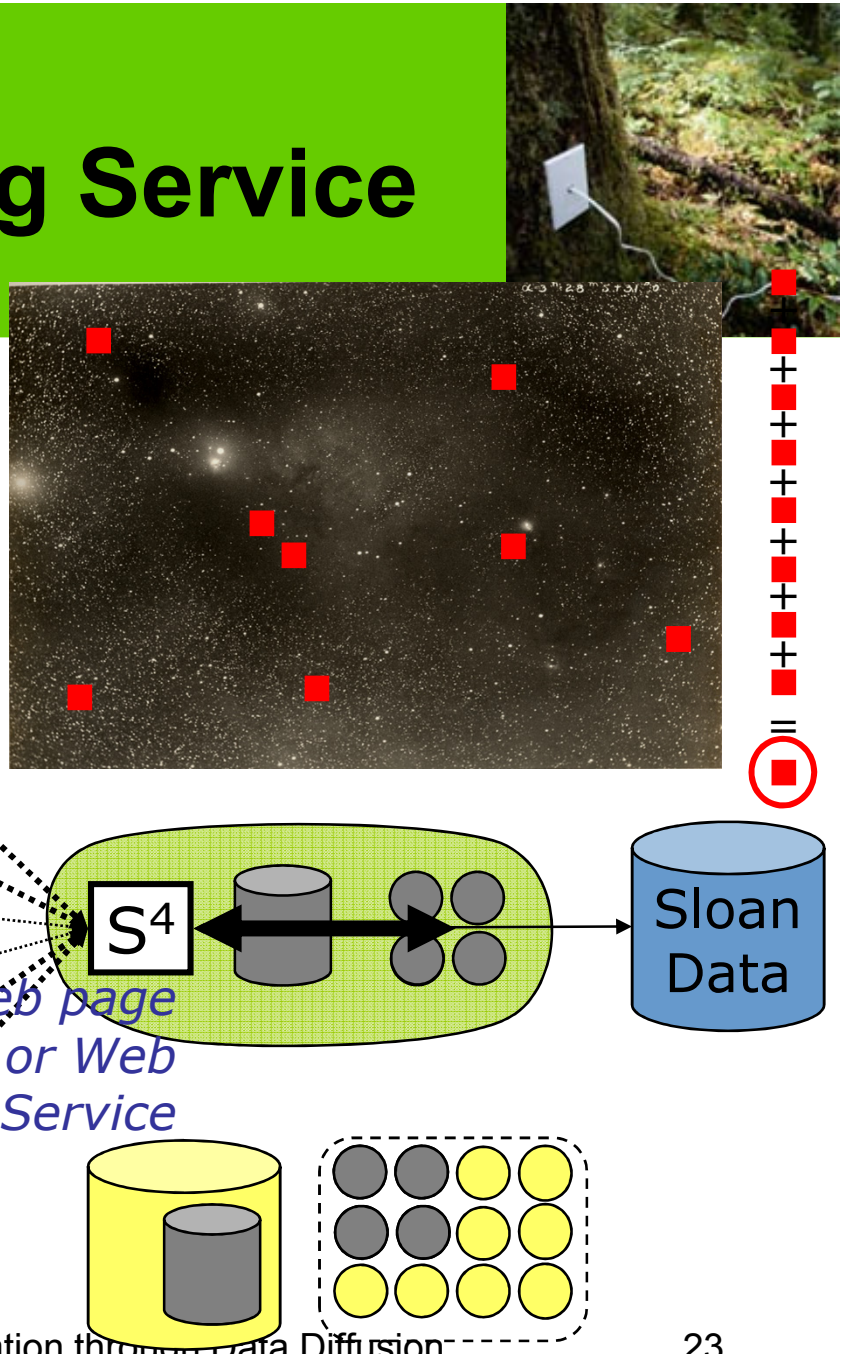




AstroPortal Stacking Service

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

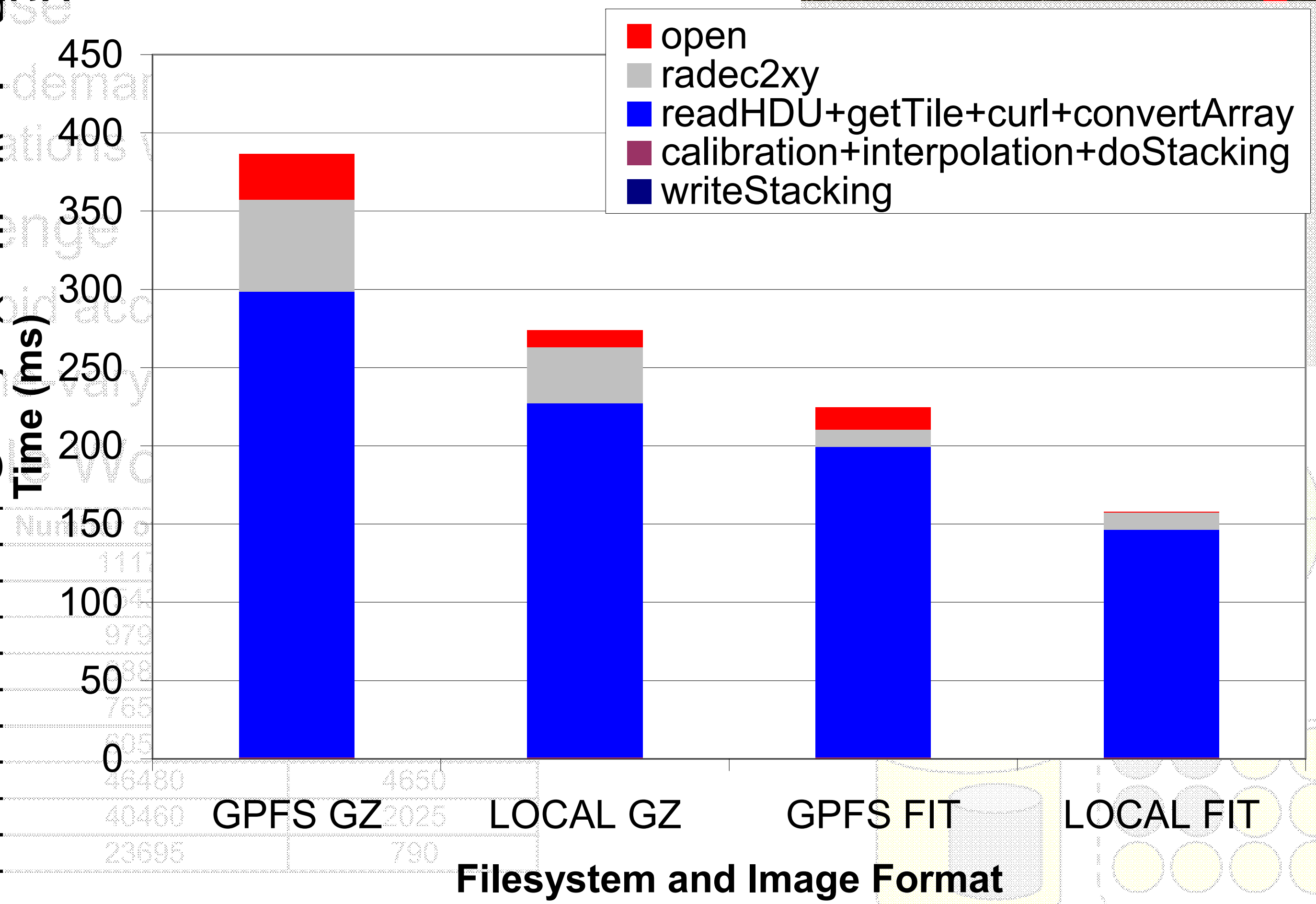
Locality	Number of Objects	Number of Files
1	111700	111700
1.38	154345	111699
2	97999	49000
3	88857	29620
4	76575	19145
5	60590	12120
10	46480	4650
20	40460	2025
30	23695	790

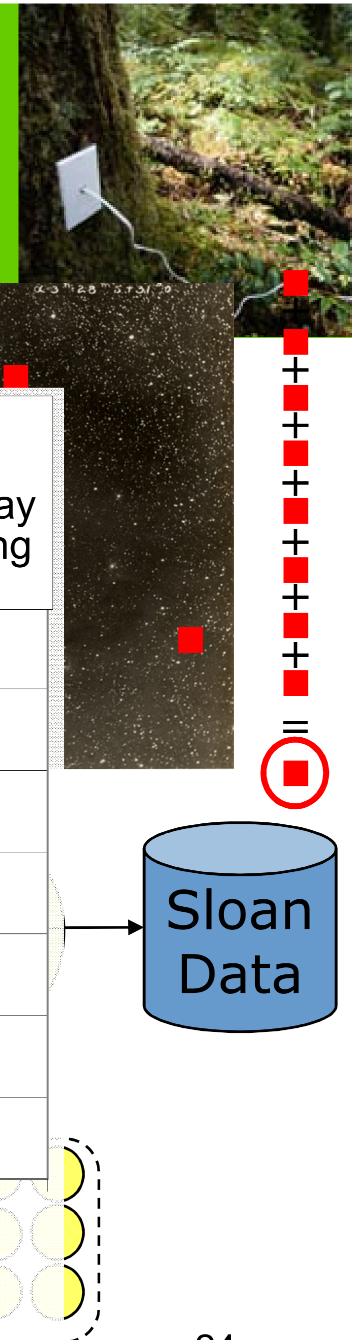


AstroPortal Stacking Service

- Purpo...
  - On-...
    loca...
- Challe...
  - Ra...
  - Tim...
- Samp...

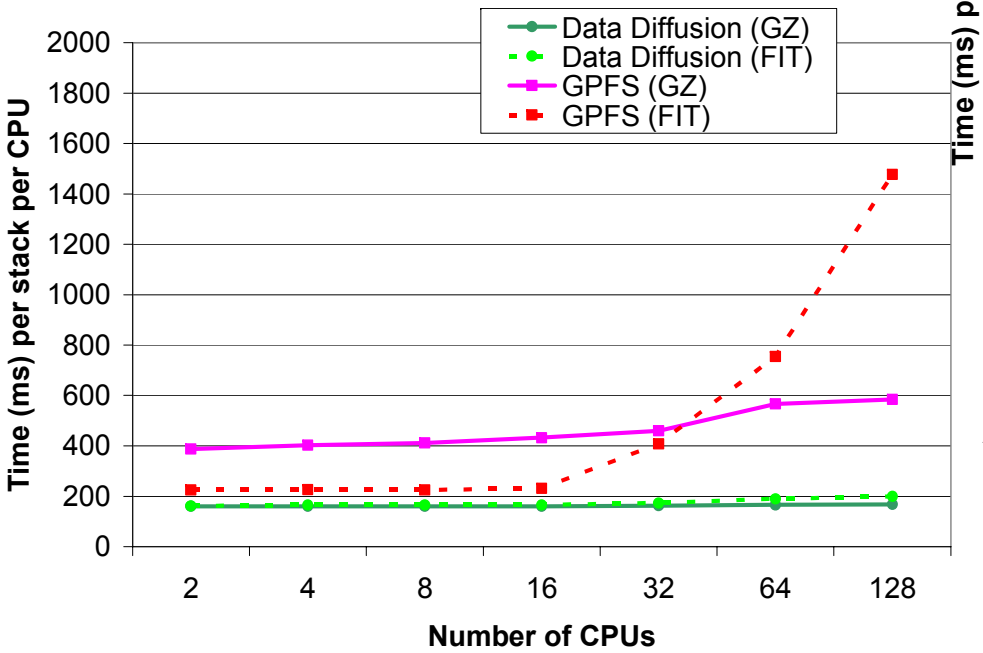
Locality	Num...	
1	[illegible]	
1.38	[illegible]	
2	[illegible]	
3	[illegible]	
4	[illegible]	
5	[illegible]	
10	46480	4650
20	40460	2025
30	23695	790

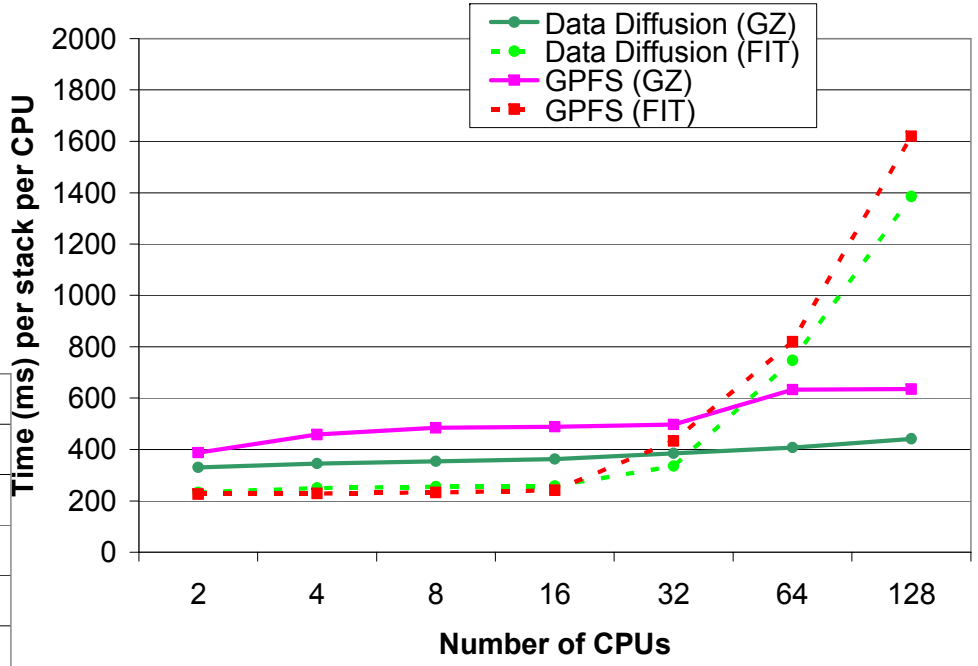




AstroPortal Stacking Service with Data Diffusion

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

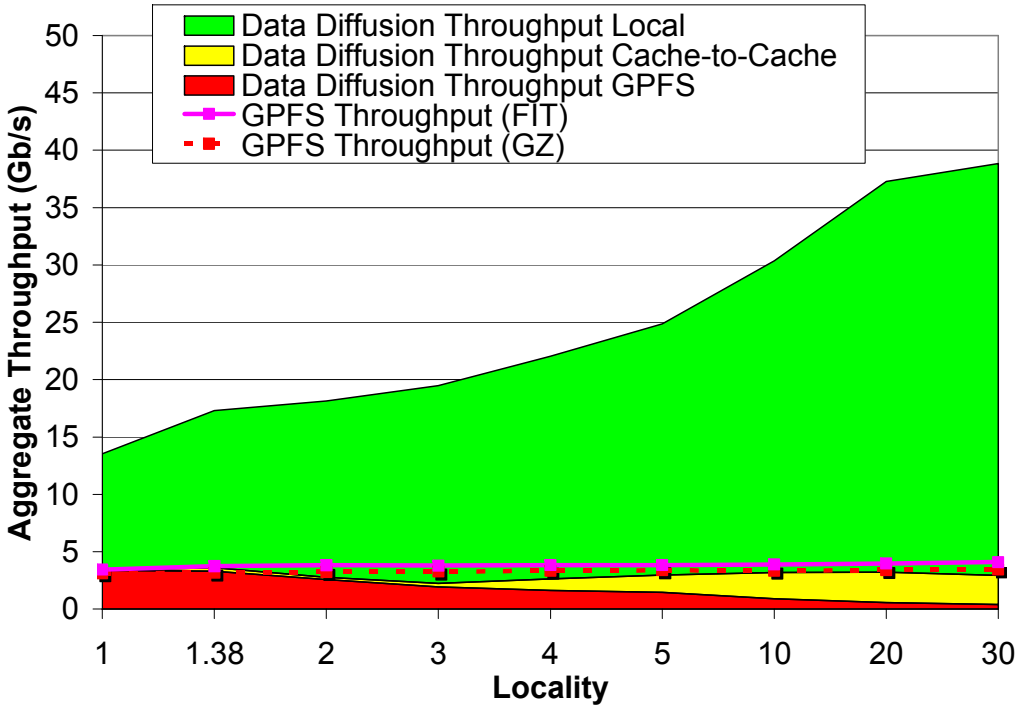


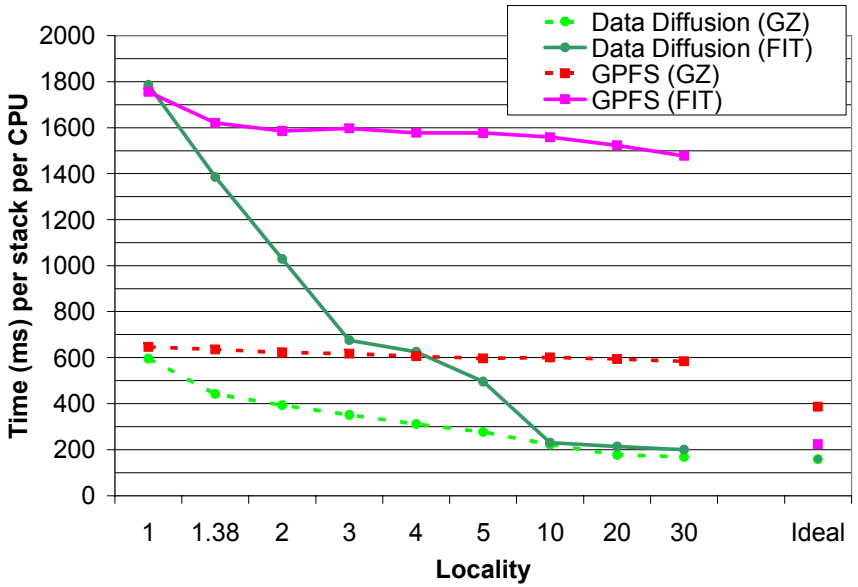


➔High data locality
- Near perfect scalability

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

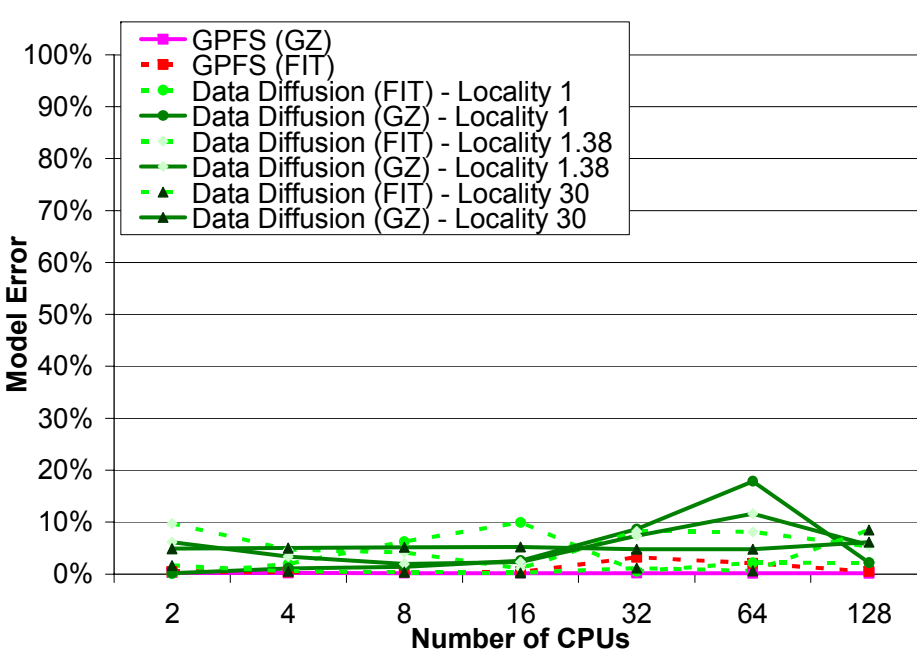


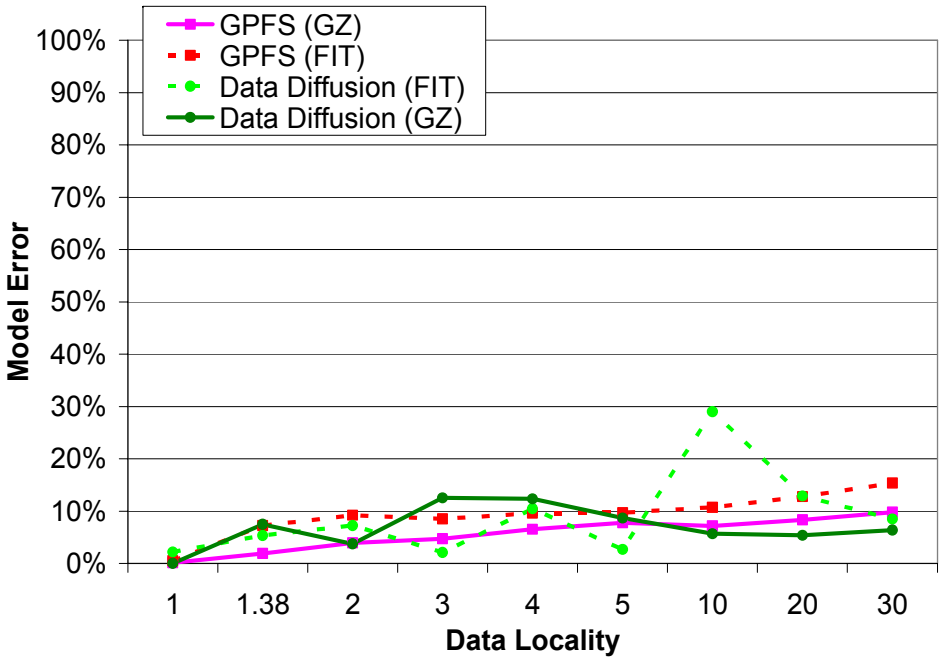


- Big performance gains as locality increases

AMDASK Model Validation

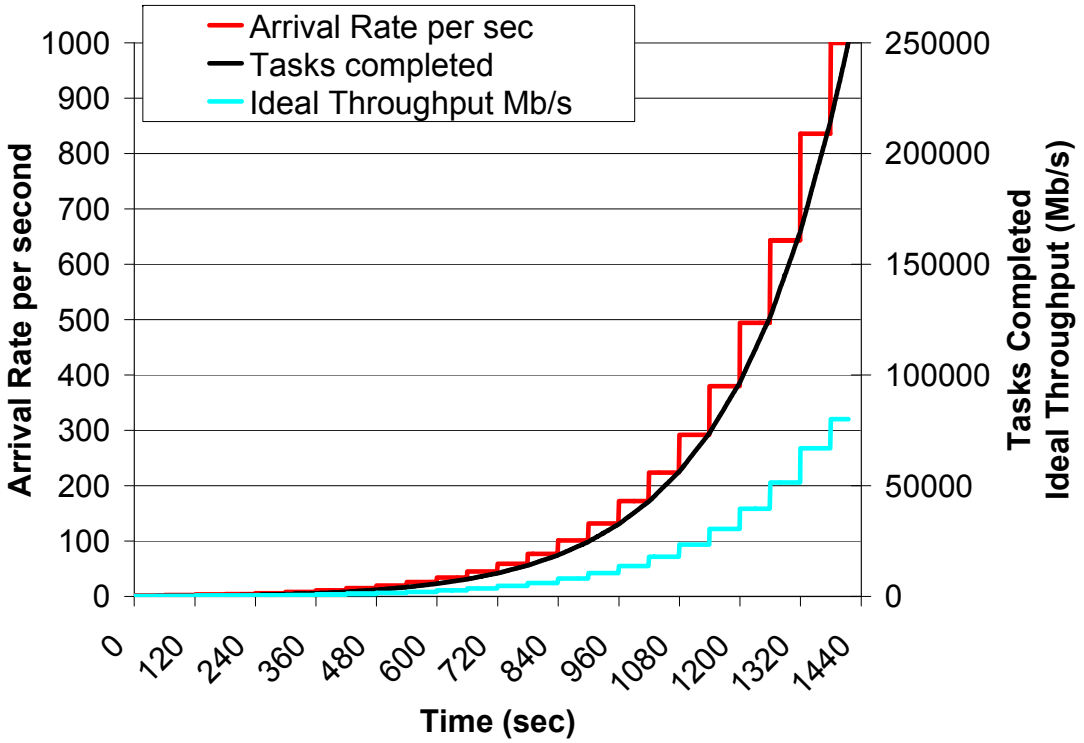
- Stacking service (large scale astronomy application)
- 92 experiments
- 558K files
  - Compressed: 2MB each ➔ 1.1TB
  - Un-compressed: 6MB each ➔ 3.3TB





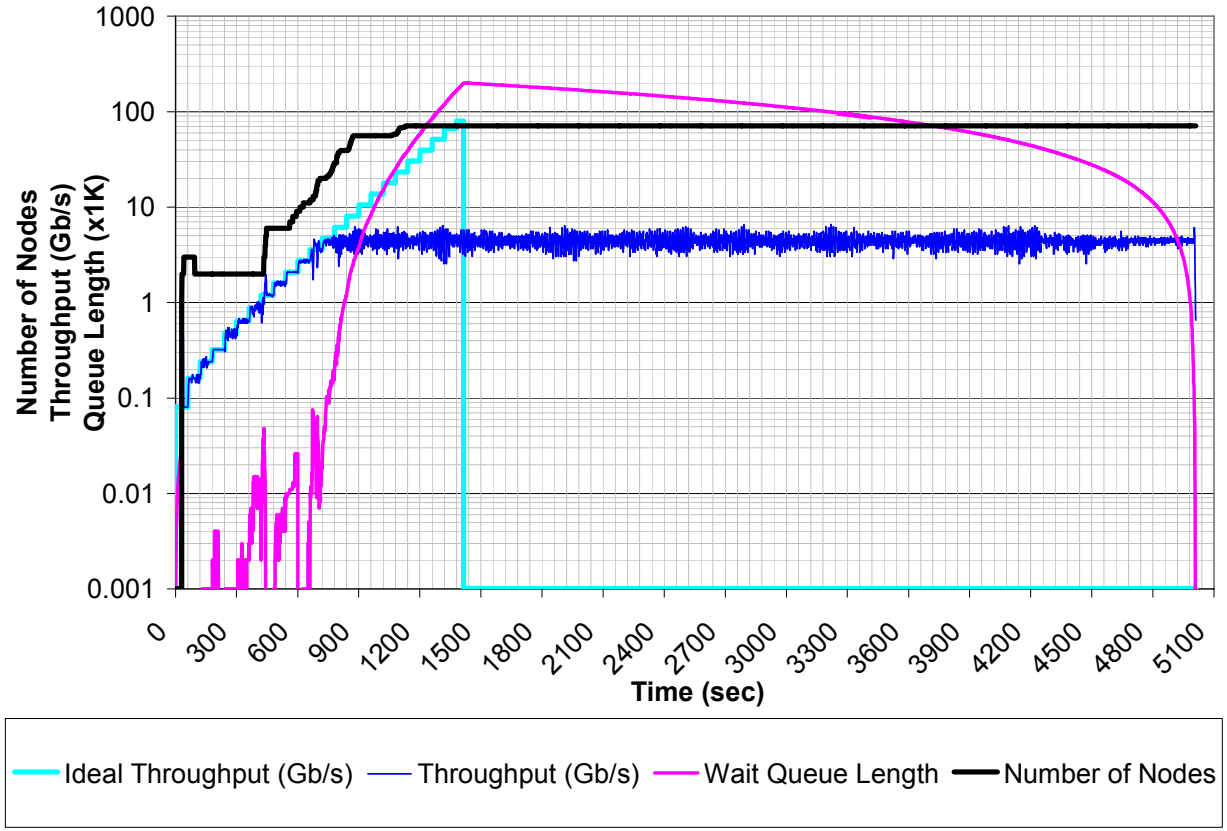
Data Diffusion: Data-Intensive Workload

- 250K tasks
  - 10MB reads
  - 10ms compute
- Vary arrival rate:
  - Min: 1 task/sec
  - Increment function: CEILING(*1.3)
  - Max: 1000 tasks/sec
- 128 processors
- Ideal case:
  - 1415 sec
  - 80Gb/s peak throughput



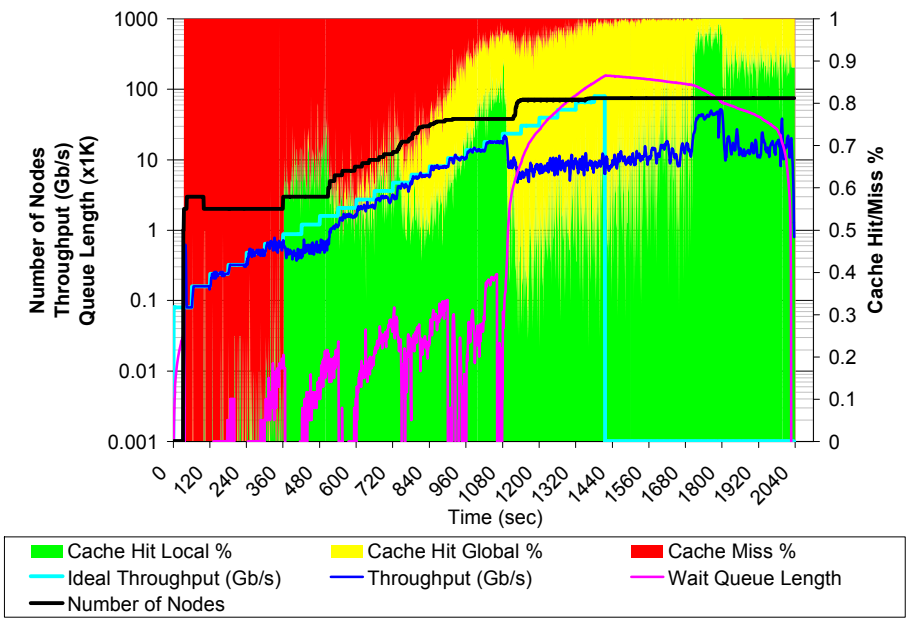
Data Diffusion: First-available (GPFS)

- Comparing GPFS with ideal
  - 5011 sec vs. 1415 sec

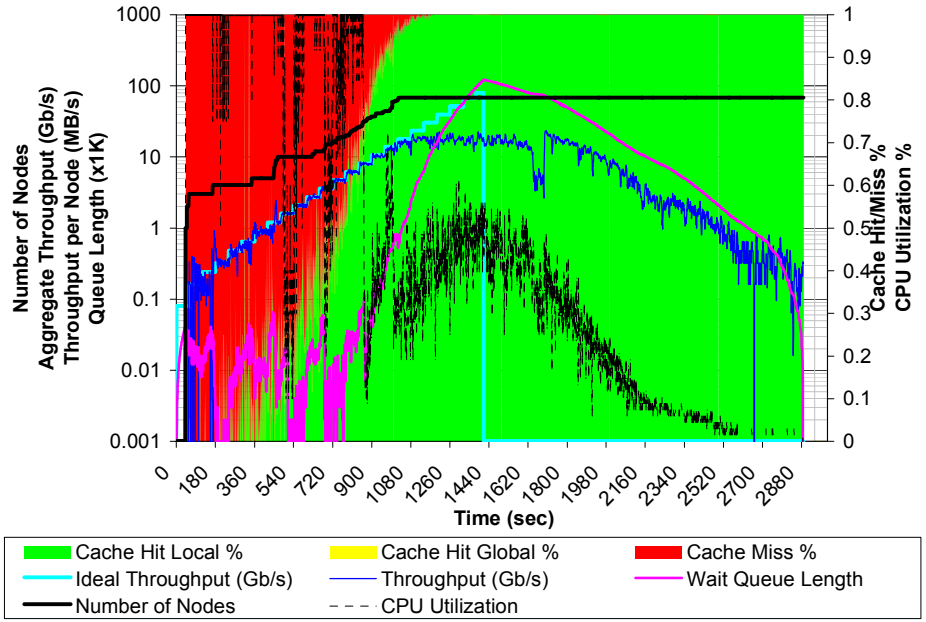


Data Diffusion: Max-compute-util & max-cache-hit

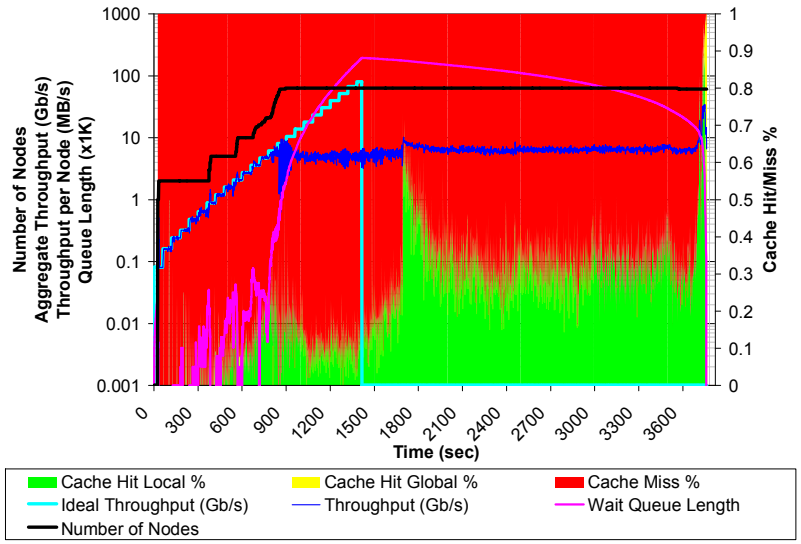
Max-compute-util



Max-cache-hit

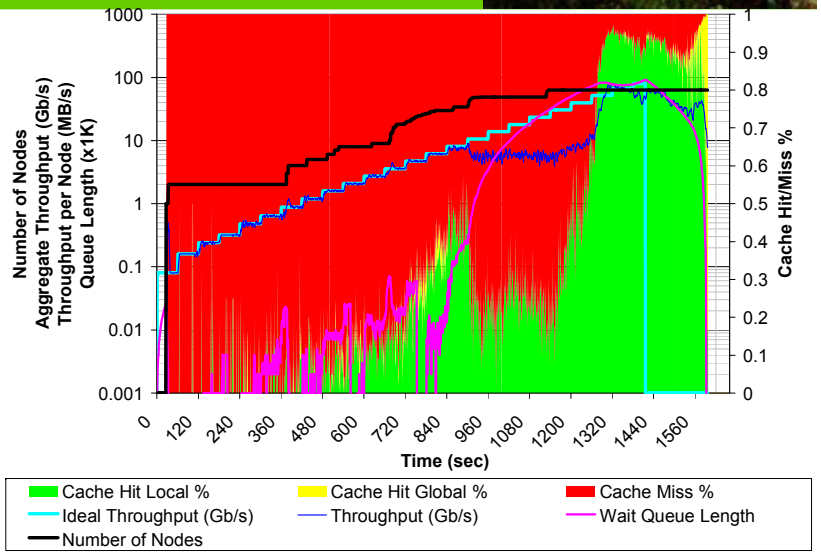


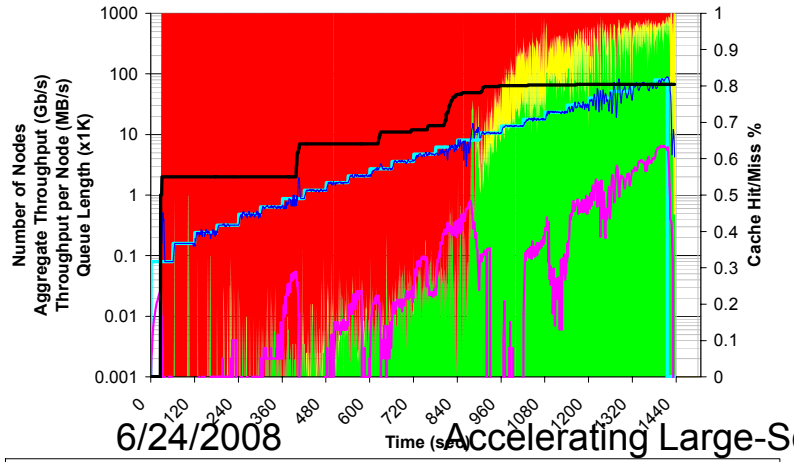
Data Diffusion: Good-cache-compute



← 1GB

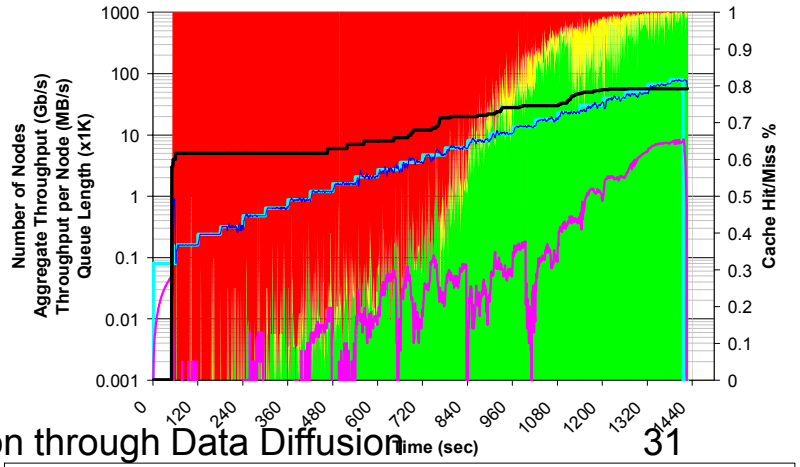
1.5GB →



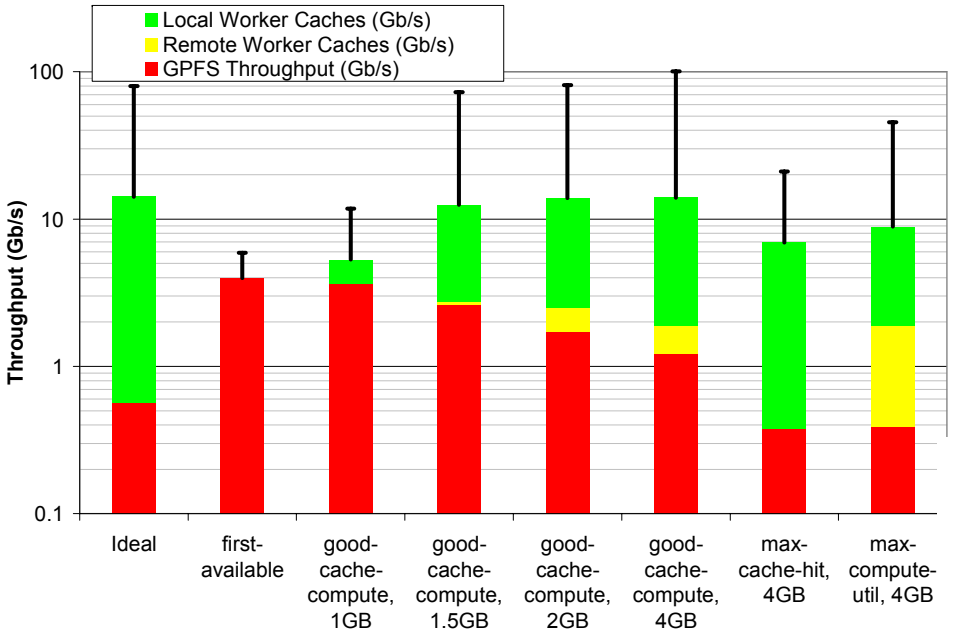


← 2GB

4GB →



Data Diffusion: Throughput and Response Time

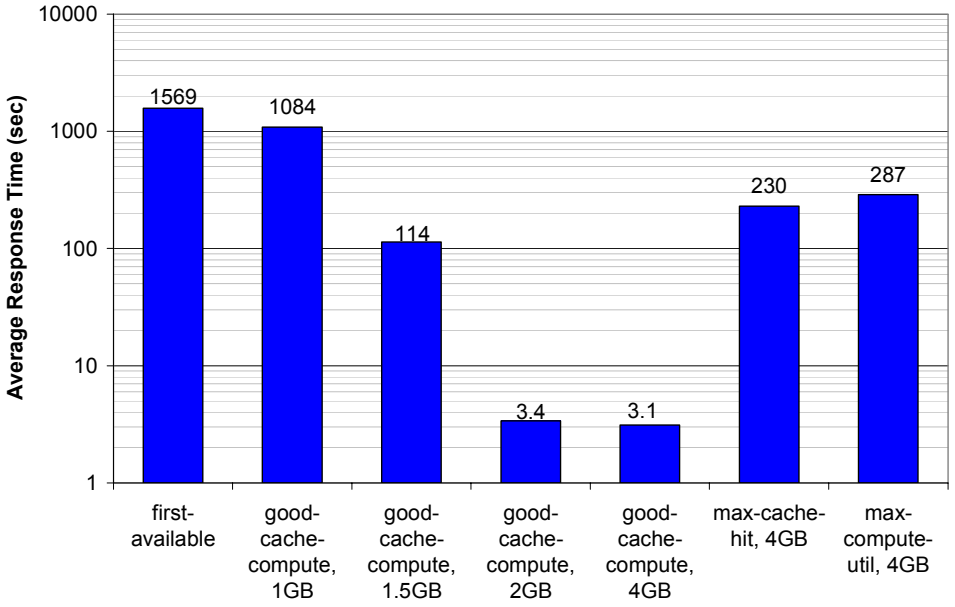


←Throughput:

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

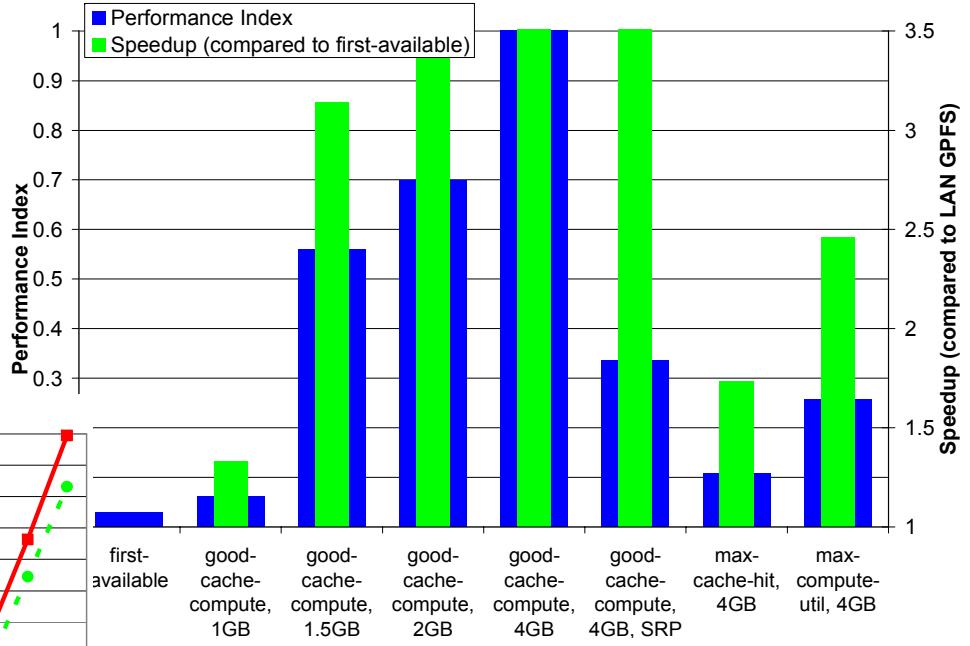
Response Time →

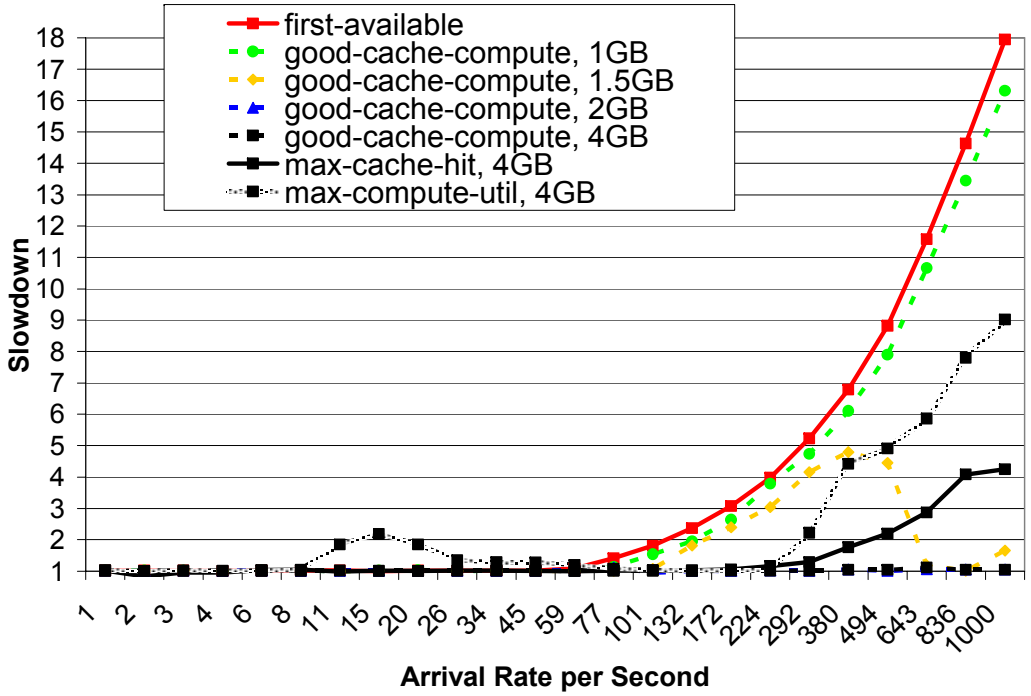
- 3 sec vs 1569 sec → 506X



Data Diffusion: Performance Index, Slowdown, and Speedup

- Performance Index:
  - 34X higher
- Speedup
  - 3.5X faster than GPFS





- Slowdown:
  - 18X slowdown for GPFS
  - Near ideal 1X slowdown for large enough caches

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: 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

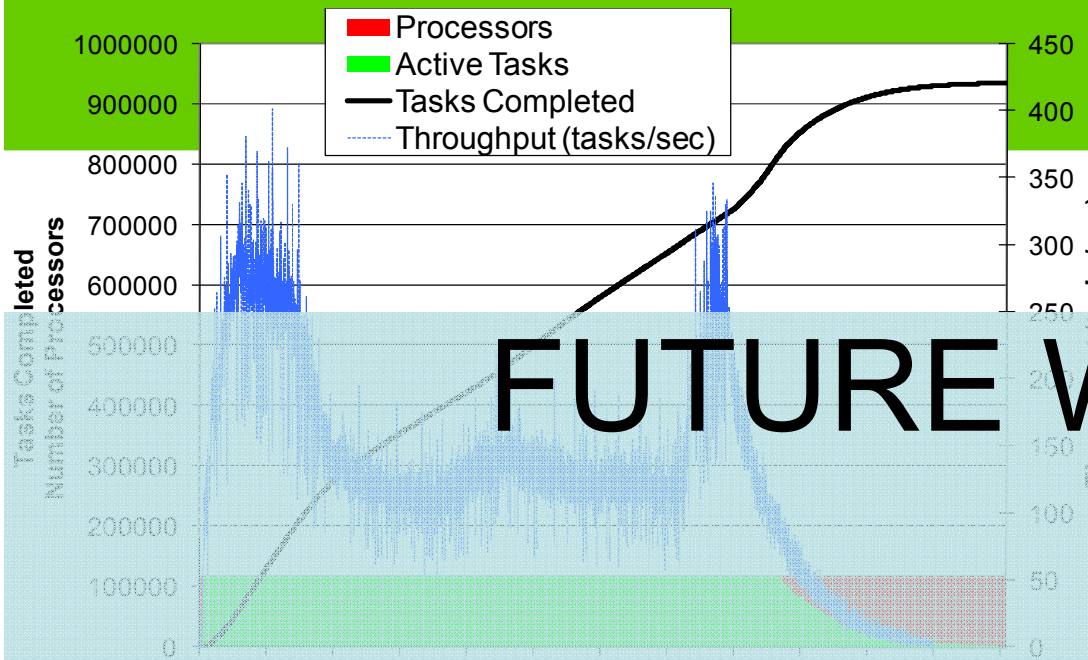
- [*Beynon01*]: **DataCutter**
- [*Ranganathan03*]: **Simulations**
- [*Ghemawat03,Dean04,Chang06*]: **BigTable**, **GFS**, **MapReduce**
- [*Liu04*]: **GridDB**
- [*Chervenak04,Chervenak06*]: **RLS** (Replica Location Service), **DRS** (Data Replication Service)
- [*Tatebe04,Xiaohui05*]: **GFarm**
- [*Branco04,Adams06*]: **DIAL/ATLAS**
- *[Kosar06]*: **Stork**

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.

Scaling from 1K to 100K CPUs with no Data Diffusion

- At 1K CPUs:
  - 1 Server to manage all 1K CPUs
  - Use shared file system extensively
    - Invoke application from shared file system
    - Read input data from shared file system
    - Write output data from shared file system
- At 100K CPUs:
  - N Servers to manage 100K CPUs (1:256 ratio)
  - Don’t trust the application I/O access patterns to behave optimally
    - Copy applications and input data to RAM
    - Read input data from RAM, compute, and write results to RAM
    - Archive all results in a single file in RAM
    - Copy 1 result file from RAM back to GPFS

DOCK: 1M Tasks on 100K+ CPUs



- CPU cores: 118784
- Tasks: 934803
- Elapsed time: 7257 sec
- CPU time: 21.43 CPU years
- Average task time: 667 sec
- Application Efficiency: 99.7%
- Utilization:
  - Sustained: 99.6%
  - Overall: 78.3%

FUTURE WORK

Enable Data Diffusion on the IBM BlueGene/P

- GPFS
  - 1 file read (~10KB)
  - 1 file write (~10KB)
- RAM (cached from GPFS on first task per node)
  - 1 binary (~7MB)
  - Static input data (~45MB)

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 effectiveness on a real large-scale astronomy application

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
  - Yong Zhao, Microsoft
  - Alex Szalay, The Johns Hopkins University
  - Mike Wilde, Computation Institute, University of Chicago & Argonne National Laboratory
  - Rick Stevens, The 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

Workshop on Many-Task Computing on Grids and Supercomputers

MTAGS08: Workshop on Many-Task Computing on Grids and Supercomputers - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://dsl.cs.uchicago.edu/MTAGS08/

MTAGS

Workshop on Many-Task Computing on Grids and Supercomputers

co-located with ACM/IEEE SC08 (Internal Conference for High Performance, Networking, Storage and Analysis)
Austin, Texas -- November 17th, 2008

Home
Call for Papers
Program Committee
Important Dates
Paper Submission
Venue
Registration
Workshop Program

MTAGS 2008 Workshop

The 1st workshop on Many-Task Computing on Grids and Supercomputers (MTAGS) will provide the scientific community a dedicated forum for presenting new research, development, and deployment efforts of loosely coupled large scale applications on large scale clusters, Grids, and/or Supercomputers. Many-task computing, the theme of the workshop encompasses loosely coupled applications, which are generally composed of many tasks (both independent and dependent tasks) to achieve some larger application goal. The workshop will focus on innovative uses of high performance systems (HPC) for many-task computing (MTC). We welcome paper submissions on all topics related to MTC on large scale systems. Papers will be peer-reviewed, and accepted papers will be published by IEEE/ACM through the SC08 proceedings (pending approval).

- In conjunction with IEEE/ACM SuperComputing 2008
- Location: Austin Texas
- Date: November 17th, 2008
- Papers Due: Early August

Scope

This workshop will focus on the ability to manage and execute large
Ranger), Grids (i.e. TeraGrid) with a dozen sites and 100K+ proc
they are efficient at executing tightly coupled parallel jobs within a
hand, Grids have been the preferred platform for more loosely co
new class of applications as what we call Many-Task Computing
administrative boundaries. MTC systems typically achieve this using
reminiscent to High Throughput Computing (HTC); however, MTC
seconds (e.g. FLOPS, tasks/sec, MB/s I/O rates). HTC on the other hand requires large amounts of computing for longer times (months and years, rather than hours and days, and are generally measured in operations per month).

Today's existing HPC systems are a viable platform to host MTC applications. However, some challenges arise in large scale applications when run on large scale systems, which can hamper the efficiency and utilization of these large scale systems. These challenges vary from local resource manager scalability and granularity, efficient utilization of the raw hardware, shared file system contention and scalability, reliability at scale, application scalability, and understanding the limitations of the HPC systems in order to identify good candidate MTC applications.

Slides related to the motivation of the MTAGS workshop:

- http://www.slideshare.net/ianfoster/many-task-applications-for-grids-and-supercomputers-481908/
- http://people.cs.uchicago.edu/~iraicu/presentations/2008_Falkon_Swift_GlobusWorld08_5-14-08.pdf

- Shared File System performance and scalability in large deployments
- Distributed file systems
- Data caching frameworks and techniques

Large-Scale Workflow Systems

Done