The Quest for Scalable Support of Data-Intensive Workloads in Distributed Systems

Ioan Raicu
Distributed Systems Laboratory
Computational Institute
University of Chicago

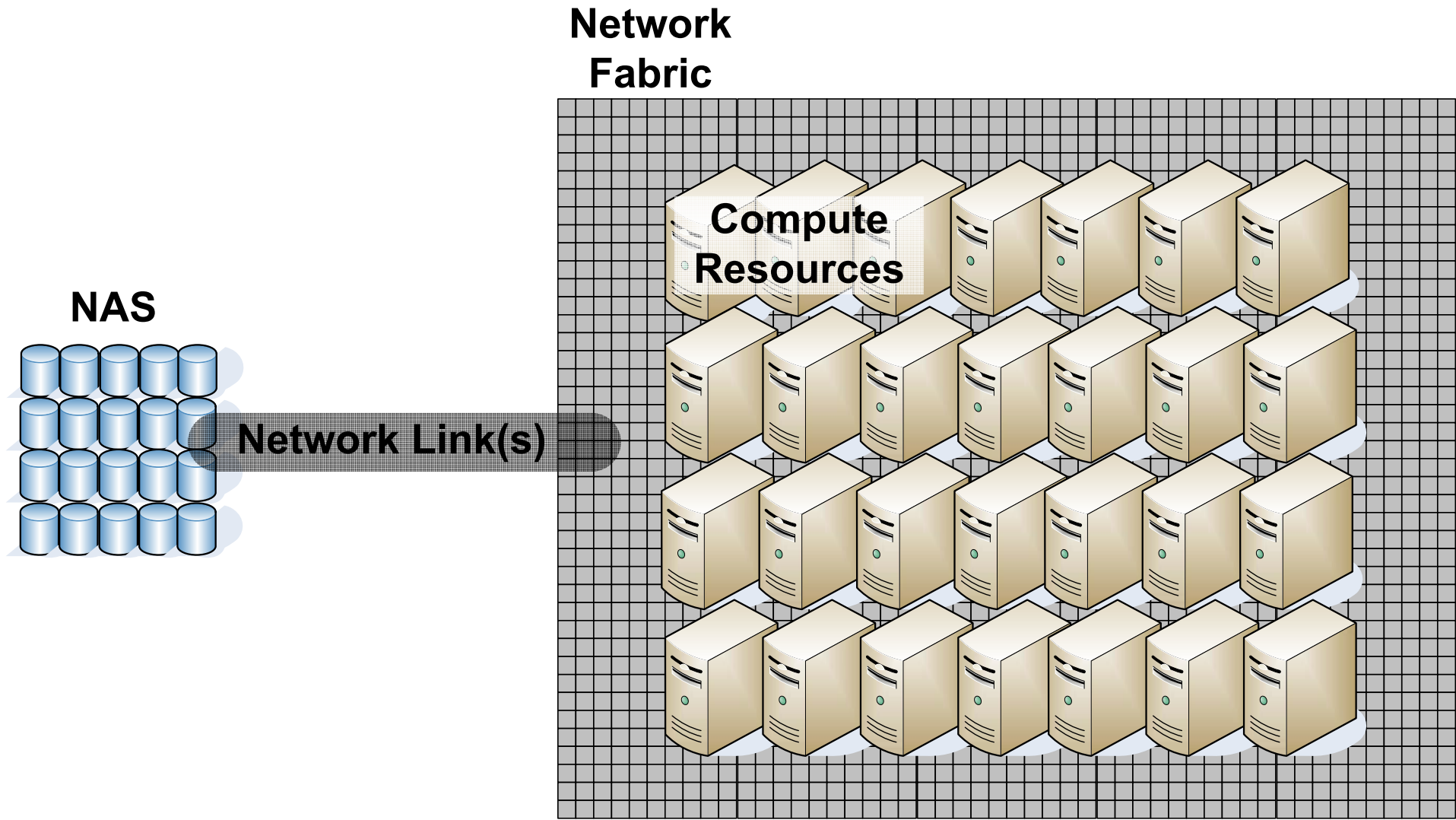
In Collaboration with:
Ian Foster, Yong Zhao, Douglas Thain, Amitabh Chaudhary, Philip Little, Christopher Moretti

ACM International Symposium on High Performance Distributed Computing (HPDC)
June 13th, 2009

State of the Art: Storage Systems

- Segregated storage and compute
  - NFS, GPFS, PVFS, Lustre
  - Batch-scheduled systems: Clusters, Grids, and Supercomputers
  - Programming paradigm: HPC, MTC, and HTC
- Co-located storage and compute
  - HDFS, GFS
  - Data centers at Google, Yahoo, and others
  - Programming paradigm: MapReduce
  - Others from academia: Sector, MosaStore, Chirp

State of the Art: Storage Systems

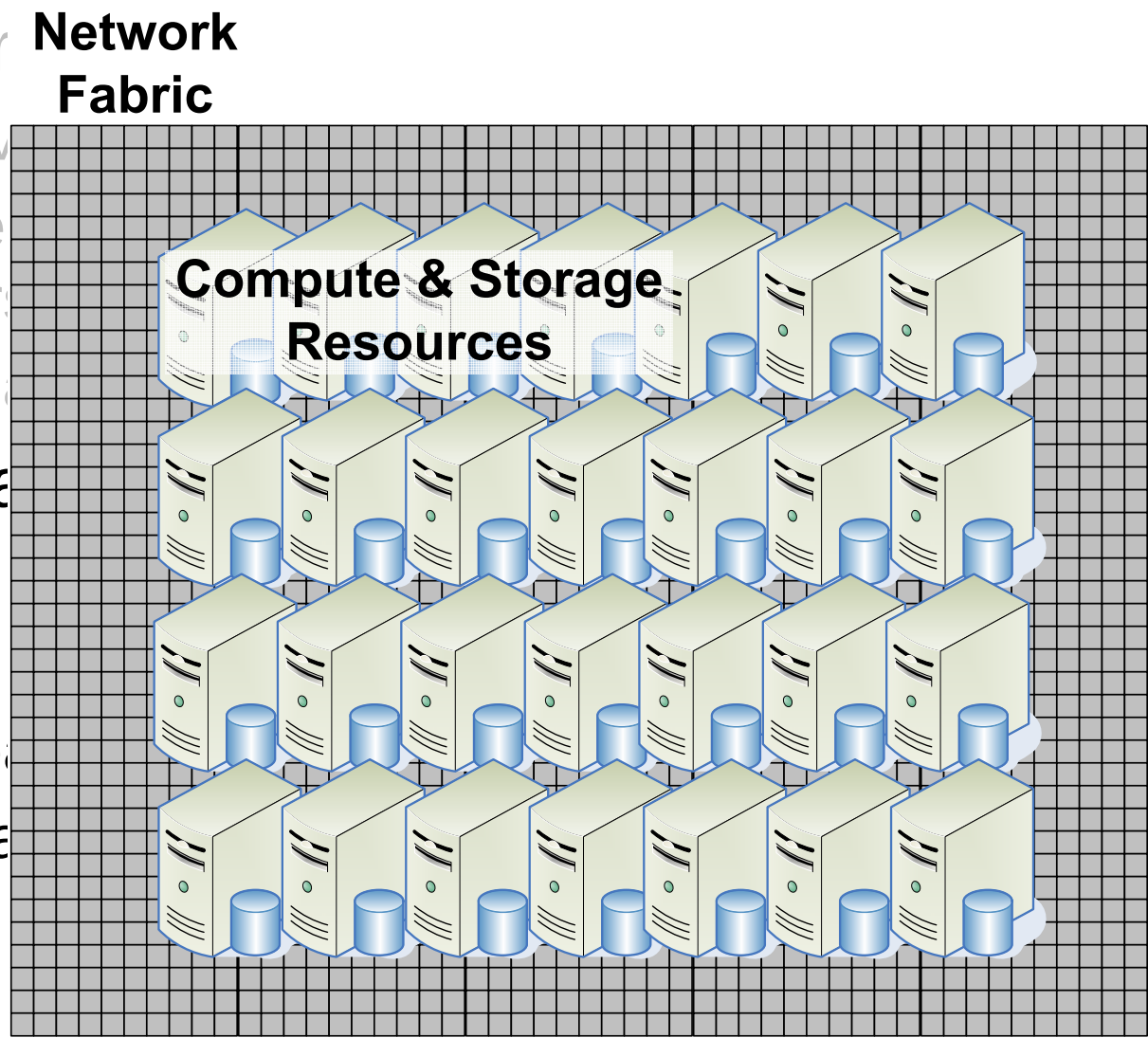


State of the Art: Storage Systems

- Segregated storage and compute
  - NFS, GPFS, PVFS, Lustre
  - Batch-scheduled systems: Clusters, Grids, and Supercomputers
  - Programming paradigm: HPC, MTC, and HTC
- Co-located storage and compute
  - HDFS, GFS
  - Data centers at Google, Yahoo, and others
  - Programming paradigm: MapReduce
  - Others from academia: Sector, MosaStore, Chirp

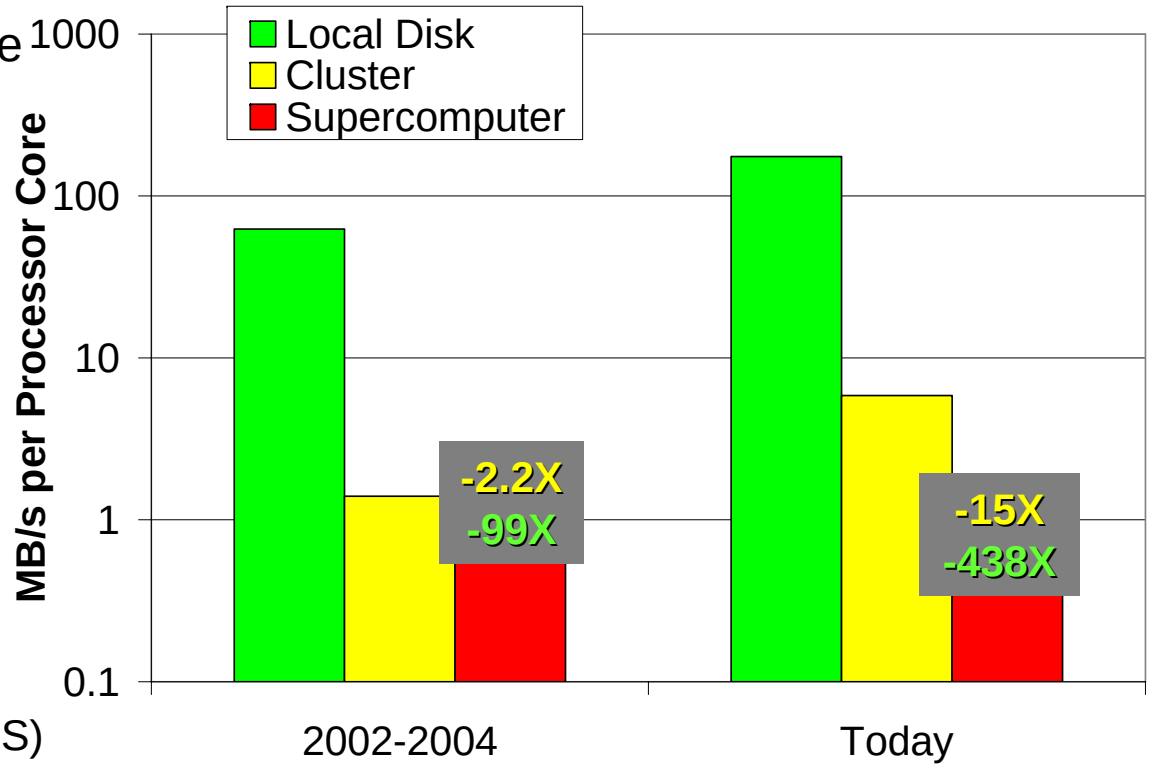
State of the Art: Storage Systems

- Segregated stor
  - NFS, GPFS, PV
  - Batch-schedule
    Supercomputer
  - Programming p
- Co-located stora
  - HDFS, GFS
  - Data centers at
  - Programming p
  - Others from aca



Growing Storage/Compute Gap

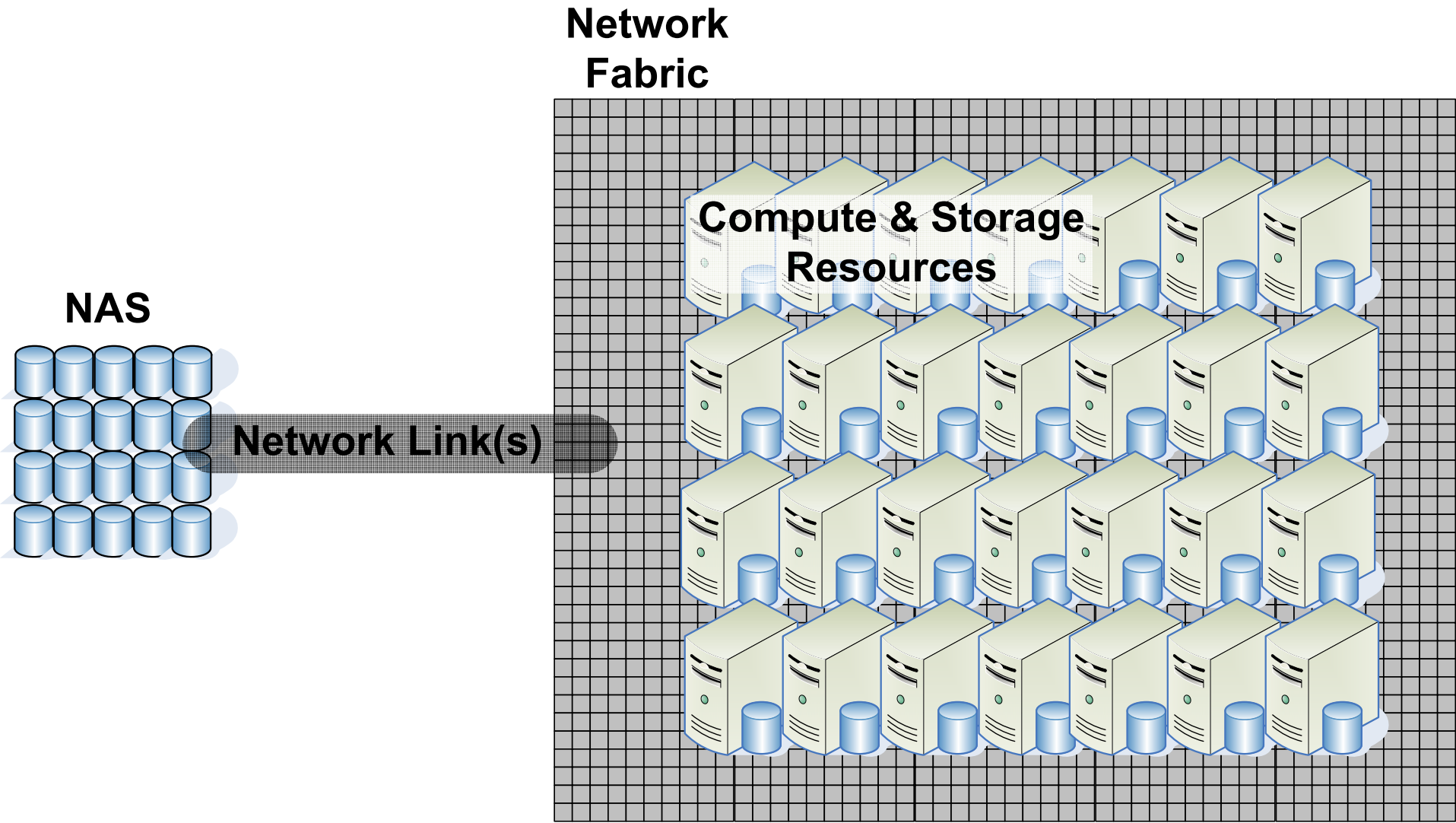
- Local Disk:
  - 2002-2004: ANL/UC TG Site (70GB SCSI)
  - Today: PADS (RAID-0, 6 drives 750GB SATA)
- Cluster:
  - 2002-2004: ANL/UC TG Site (GPFS, 8 servers, 1Gb/s each)
  - Today: PADS (GPFS, SAN)
- Supercomputer:
  - 2002-2004: IBM Blue Gene/L (GPFS)
  - Today: IBM Blue Gene/P (GPFS)



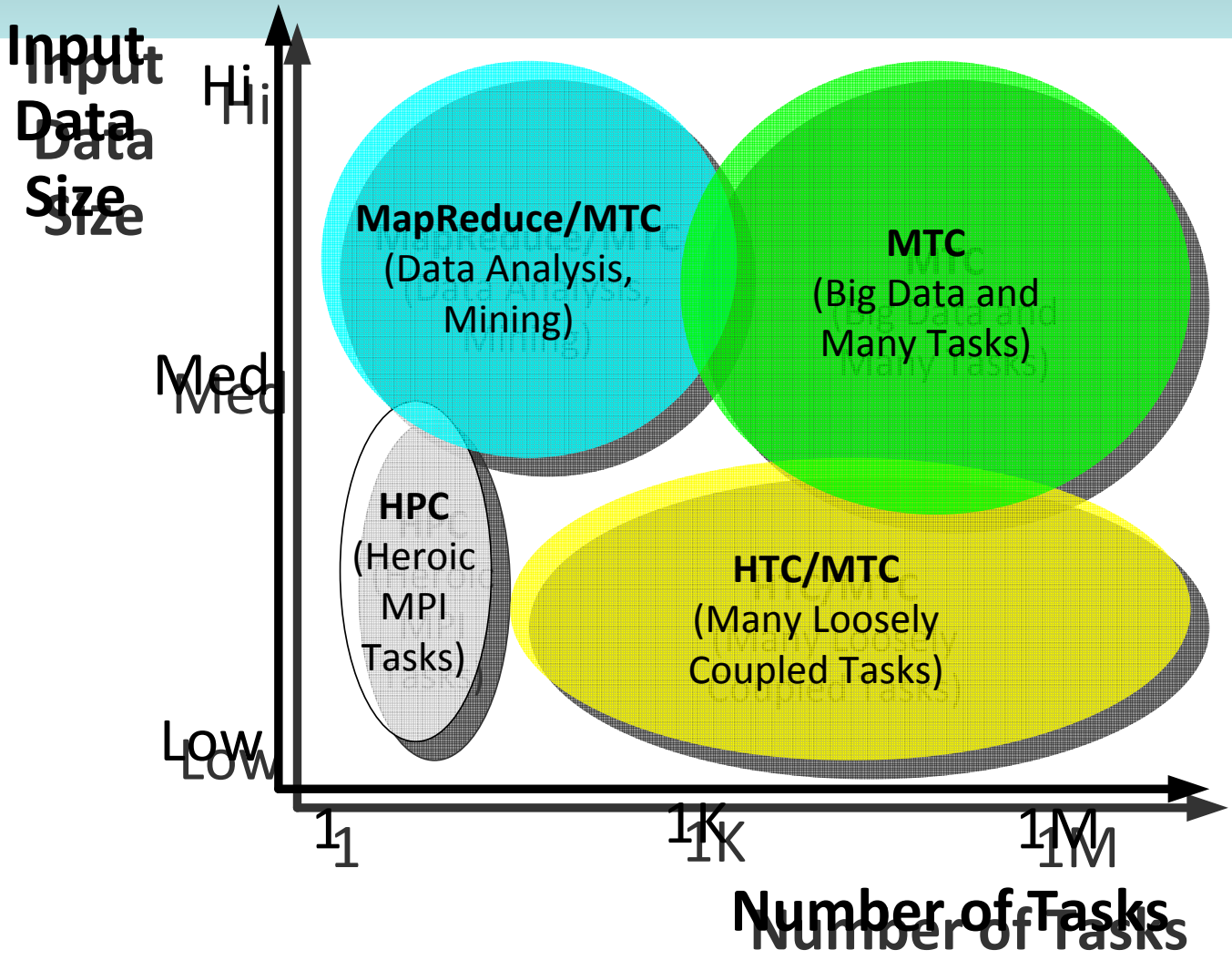
Question

What if we could combine the scientific community's existing programming paradigms, but yet still exploit the data locality that naturally occurs in scientific workloads?

Combine State of the Art Systems



Problem Space



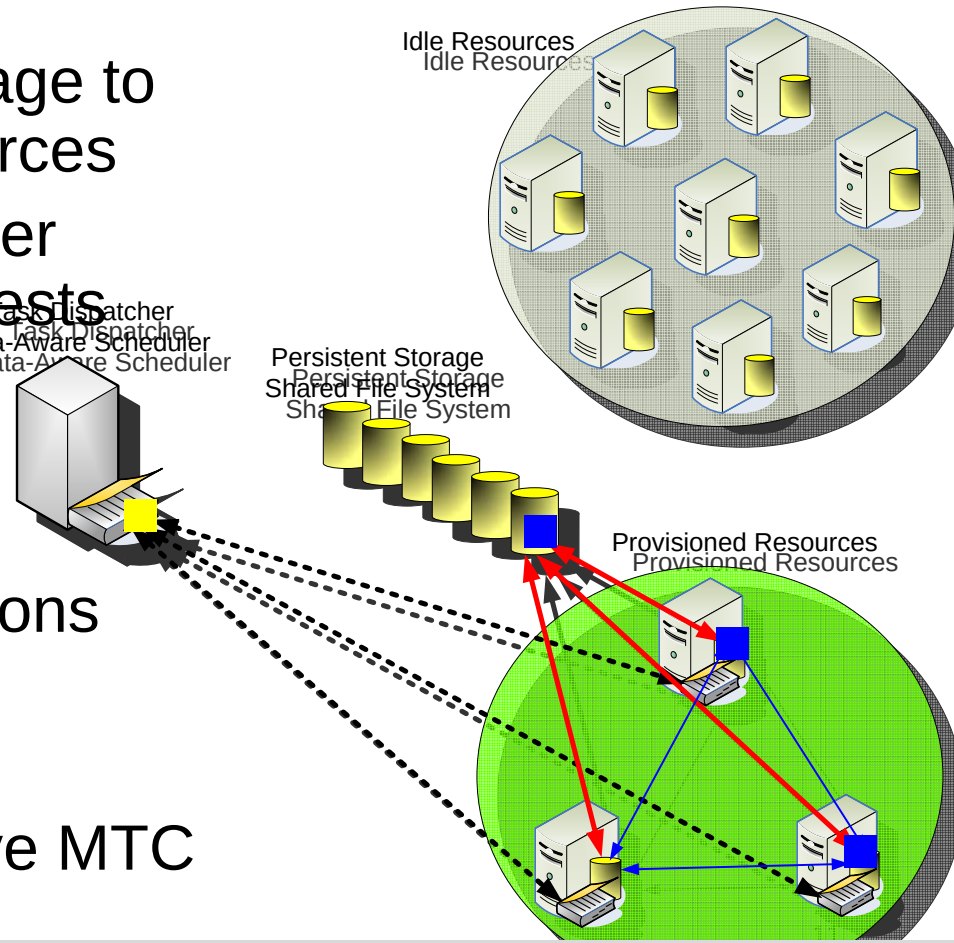
Hypothesis

"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

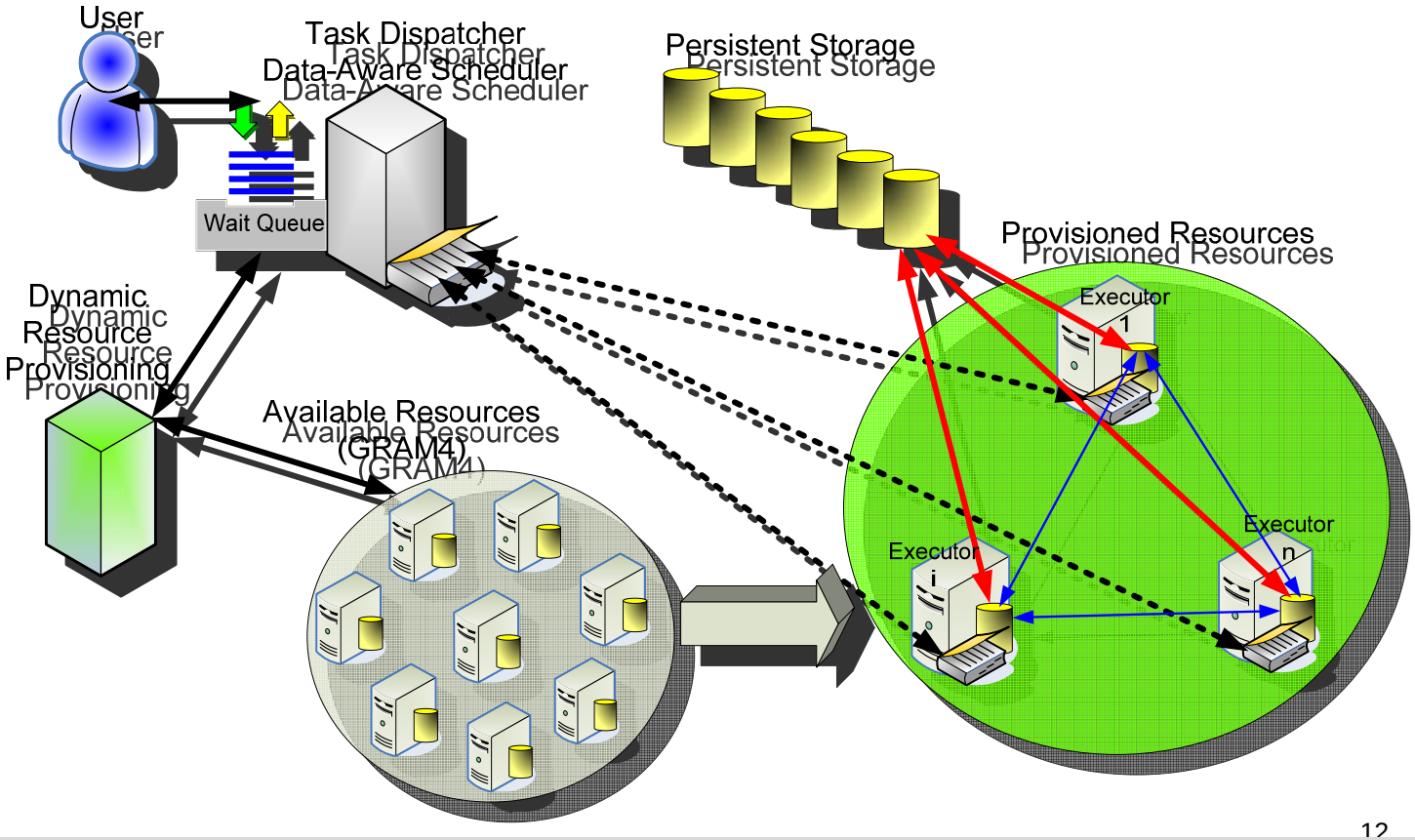
Proposed Solution: Data Diffusion

- Resource acquired in response to demand
- Data diffuse from archival storage to newly acquired transient resources
- Resource “caching” allows faster responses to subsequent requests
- Resources are released when demand drops
- Optimizes performance by co-scheduling data and computations
- Decrease dependency of a shared/parallel file systems
- Critical to support data intensive MTC



Data diffusion in Practice

- What would data diffusion look like in practice?
- Extend the Falkon framework



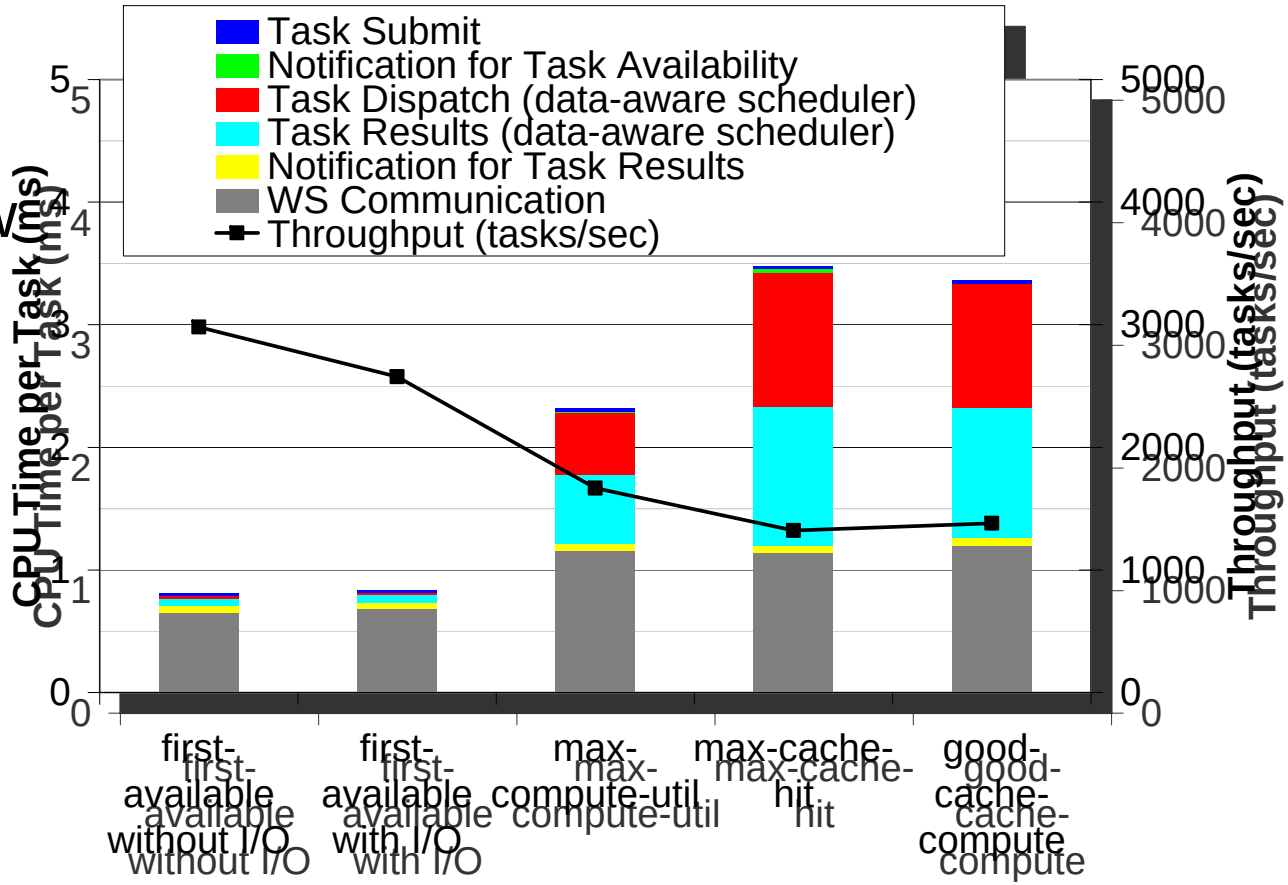
[SC07] “Falkon: a Fast and Light-weight tasK executiON framework”

Scheduling Policies

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

Data-Aware Scheduler Profiling

- 3GHz dual CPUs
- ANL/UC TG with 128 processors
- Scheduling windows 2500 tasks
- Dataset
  - 100K files
  - 1 byte each
- Tasks
  - Read 1 file
  - Write 1 file

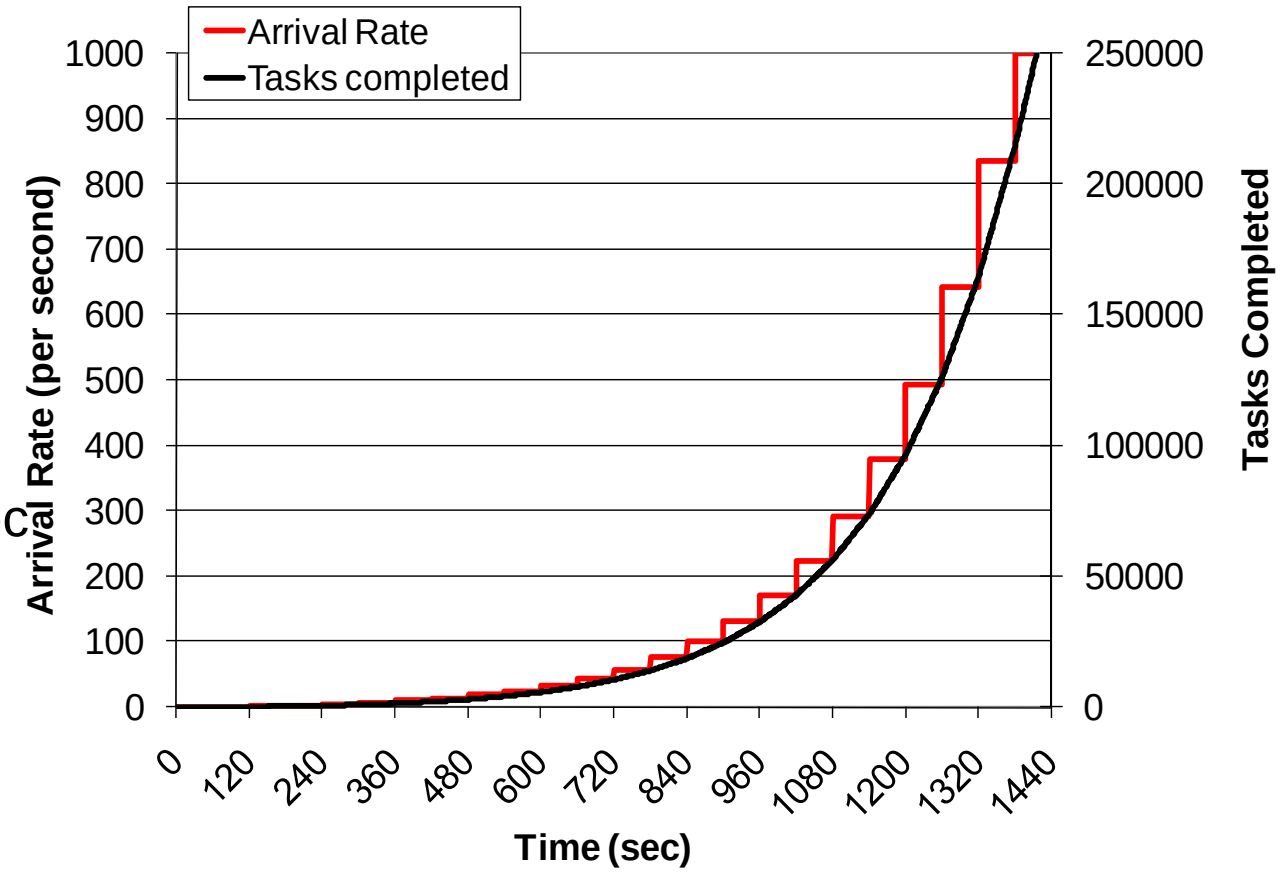


Workloads

- Monotonically Increasing Workload
  - Emphasizes increasing loads
- Sine-Wave Workload
  - Emphasizes varying loads
- All-Pairs Workload
  - Compare to best case model of active storage
- Image Stacking Workload (Astronomy)
  - Evaluate data diffusion on a real large-scale data-intensive application from astronomy domain

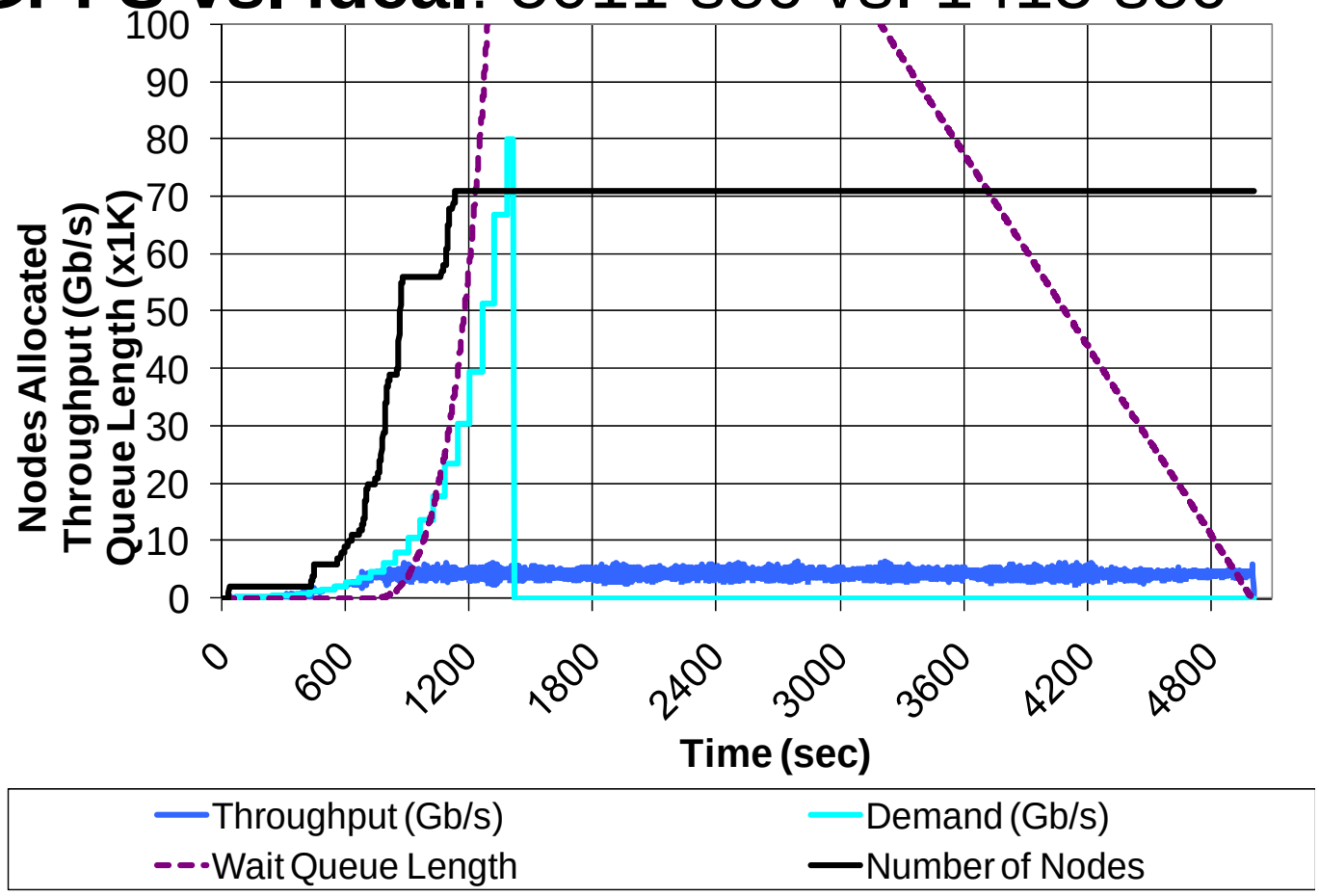
Monotonically Increasing Workload

- 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



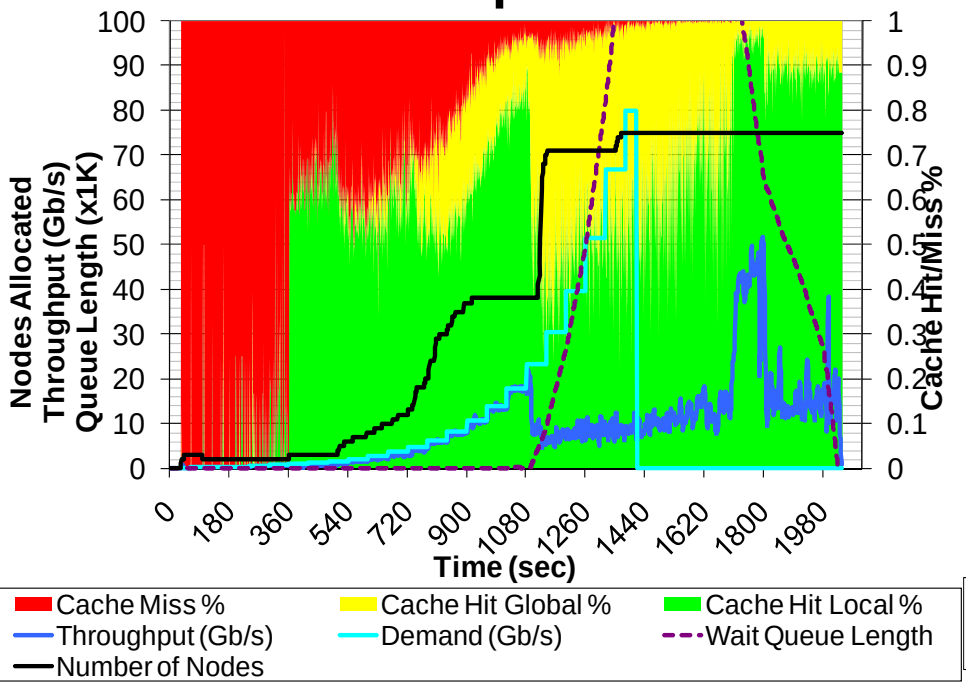
Monotonically Increasing Workload First-available (GPFS)

- **GPFS vs. ideal**: 5011 sec vs. 1415 sec

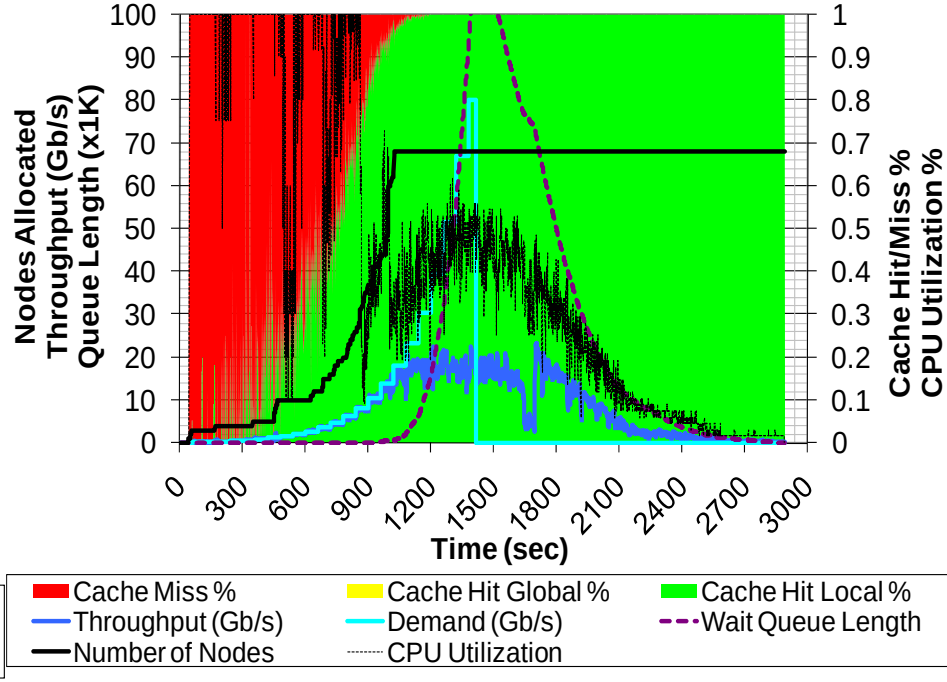


Monotonically Increasing Workload Max-compute-util & Max-cache-hit

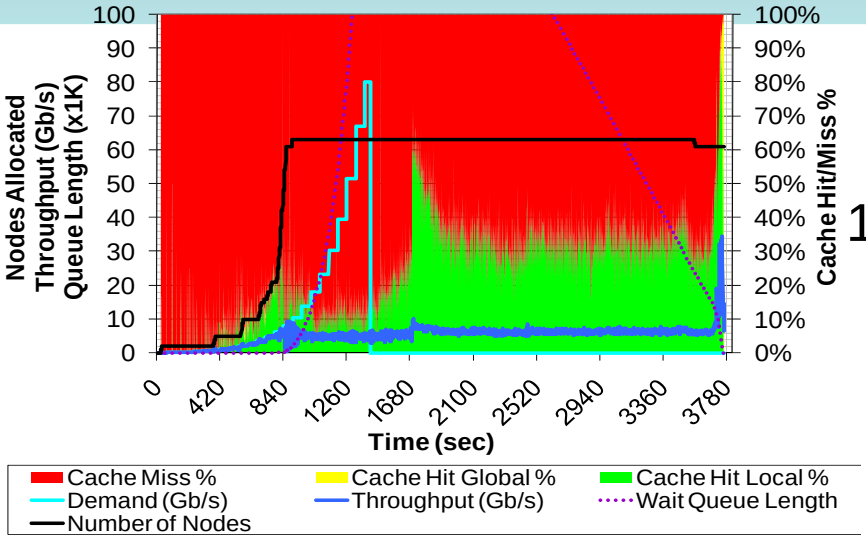
Max-compute-util



Max-cache-hit

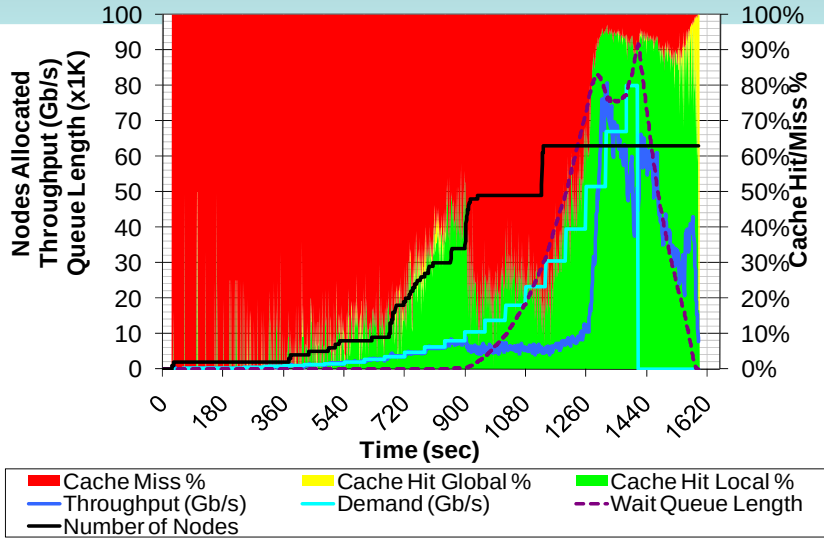


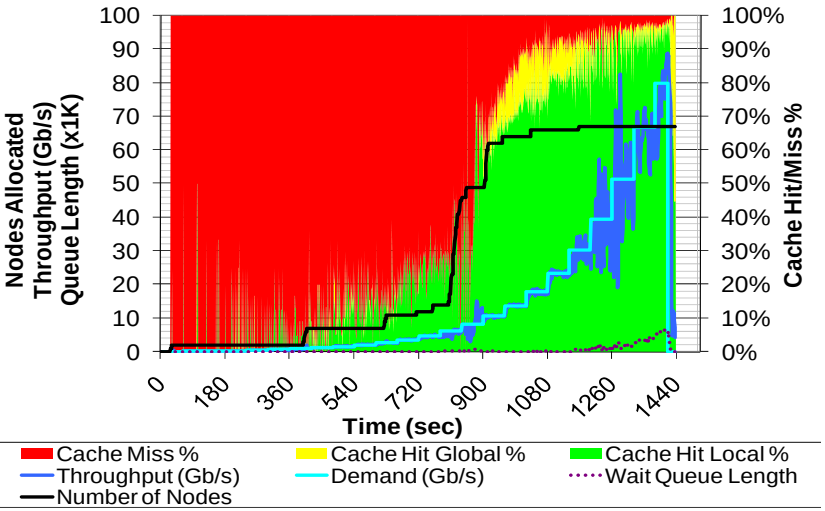
Monotonically Increasing Workload Good-cache-compute



← 1GB

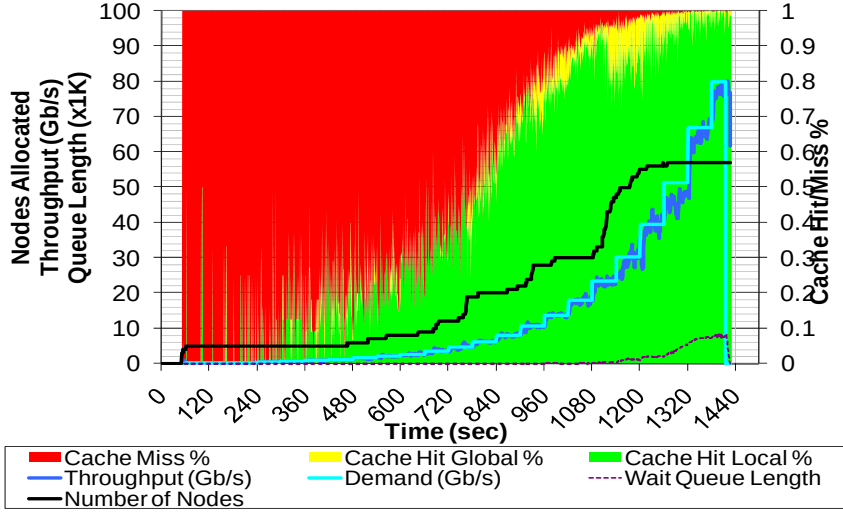
1.5GB →





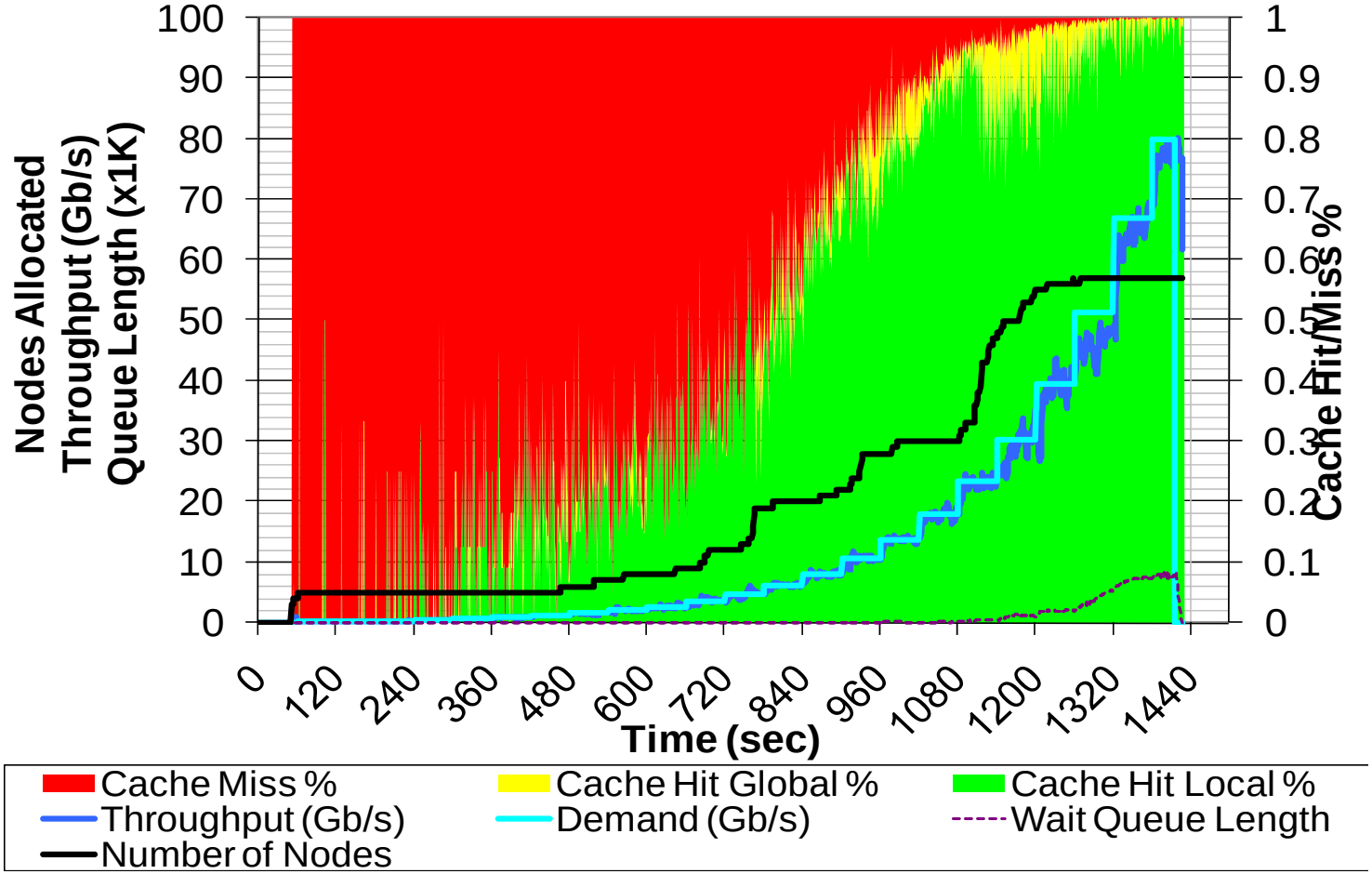
← 2GB

4GB →

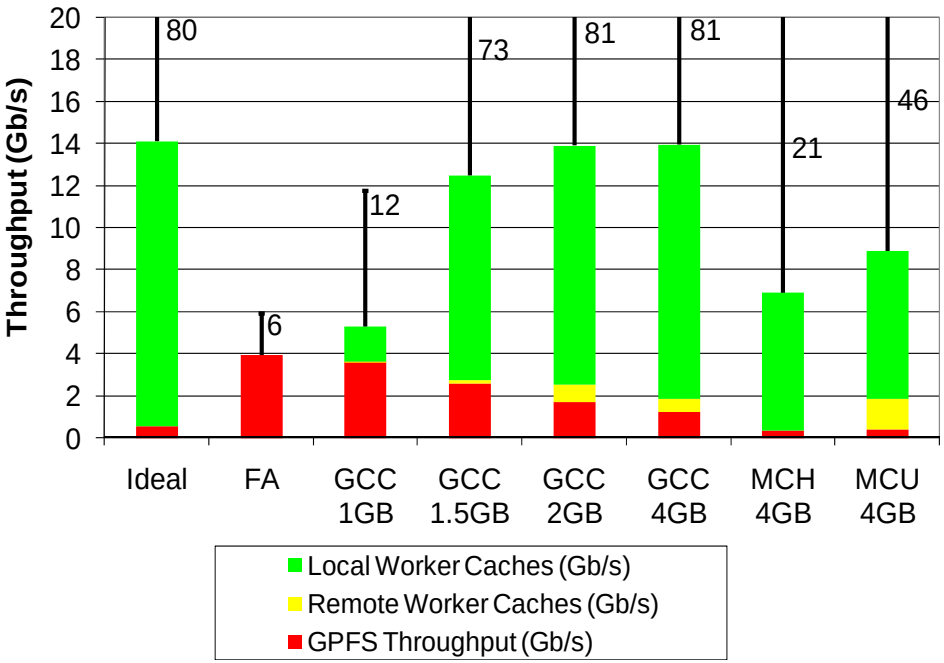


Monotonically Increasing Workload Good-cache-compute

- **Data Diffusion vs. ideal**: 1436 sec vs 1415 sec



Monotonically Increasing Workload Throughput and Response Time



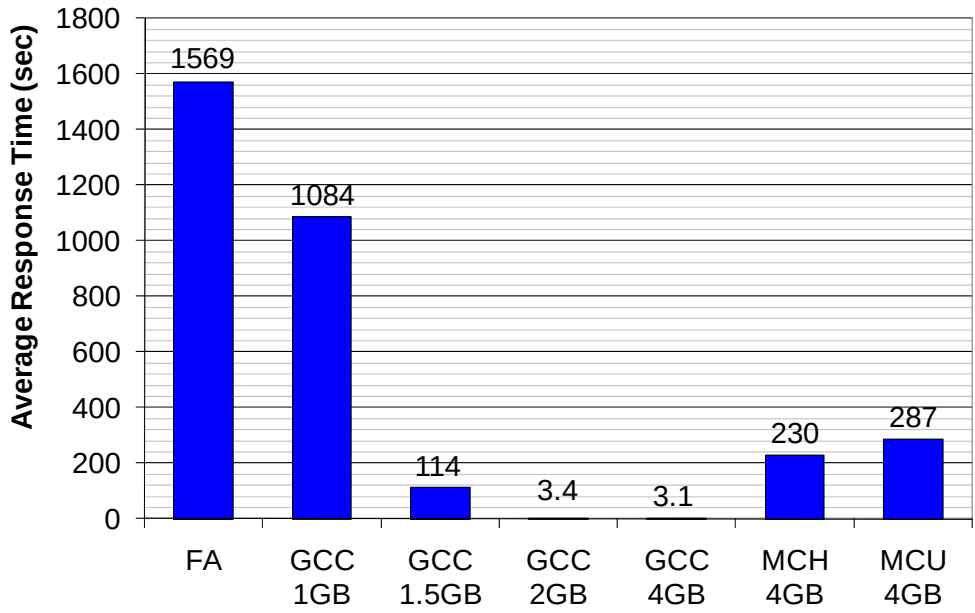
Response Time ➔

– 3 sec vs 1569 sec ➔ 506X

➔Throughput:

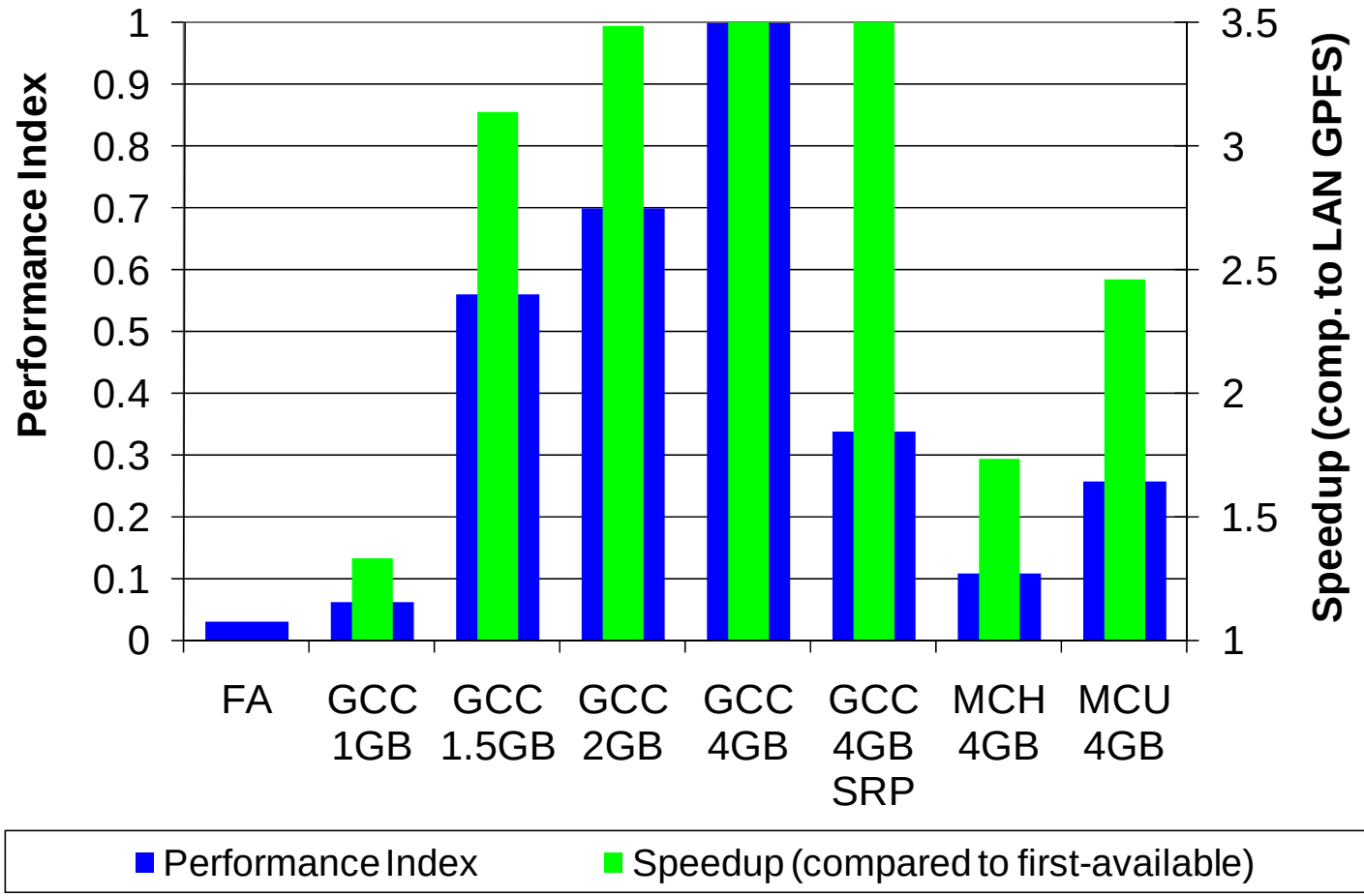
– Average: 14Gb/s vs 4Gb/s

– Peak: 81Gb/s vs. 6Gb/s



Monotonically Increasing Workload Performance Index and Speedup

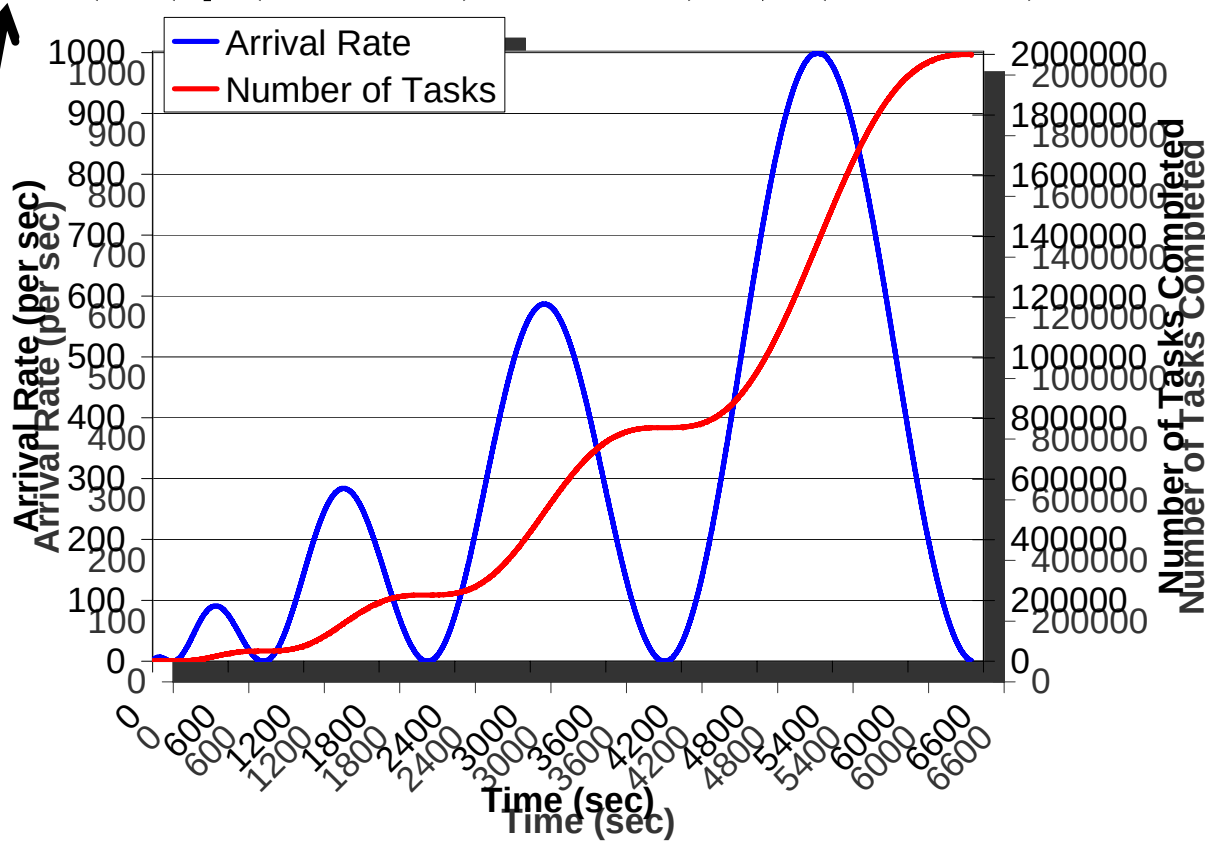
- Performance Index:
  - 34X higher
- Speedup
  - 3.5X faster than GPFS



Sine-Wave Workload

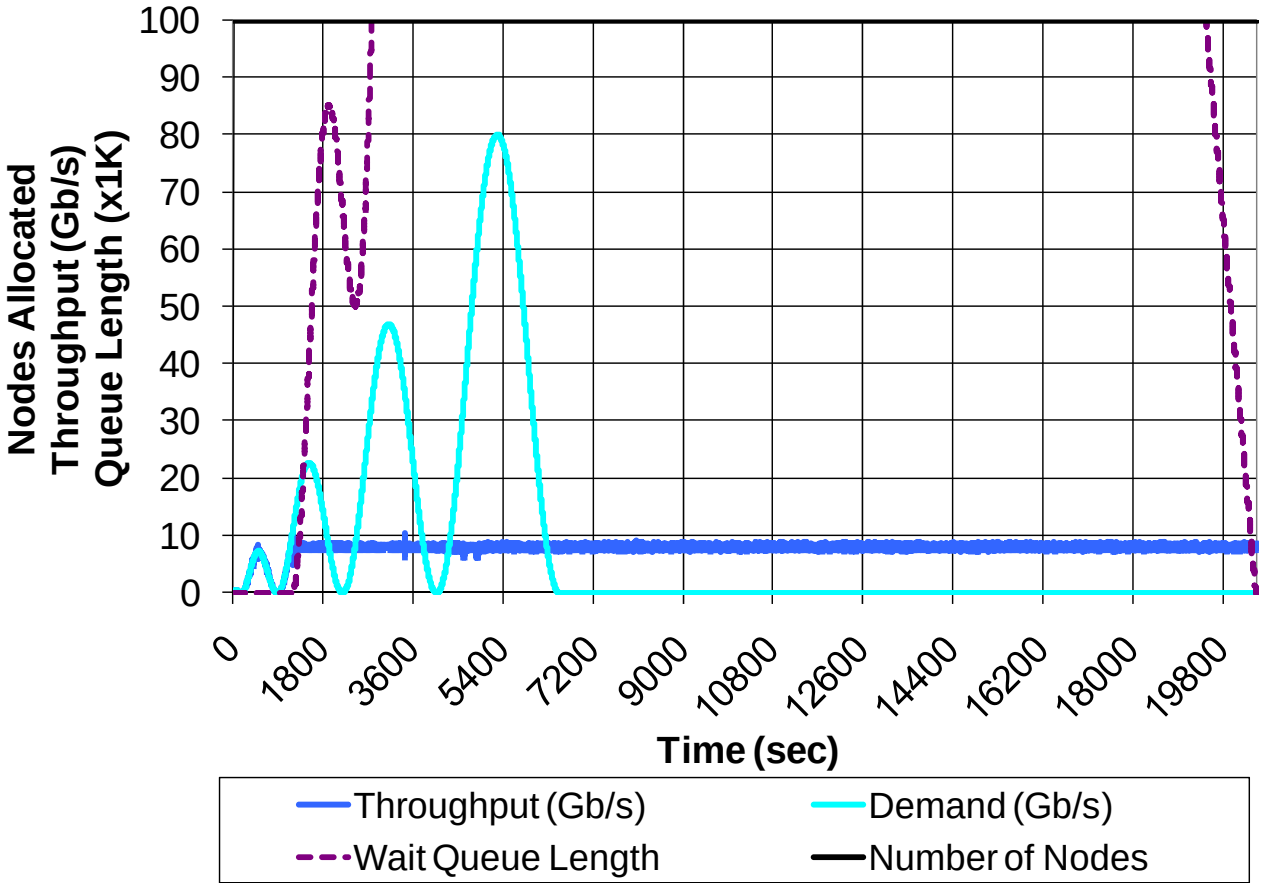
- 2M tasks
  - 10MB reads
  - 10ms compute
- Vary arrival rate:
  - Min: 1 task/sec
  - Arrival rate function:
  - Max: 1000 tasks/sec
- 200 processors
- Ideal case:
  - 6505 sec
  - 80Gb/s peak throughput

$$A \quad (\sin(sqrt(time \quad 0.11)*2.859678) \quad 1)*(time \quad 0.11)*5.705$$



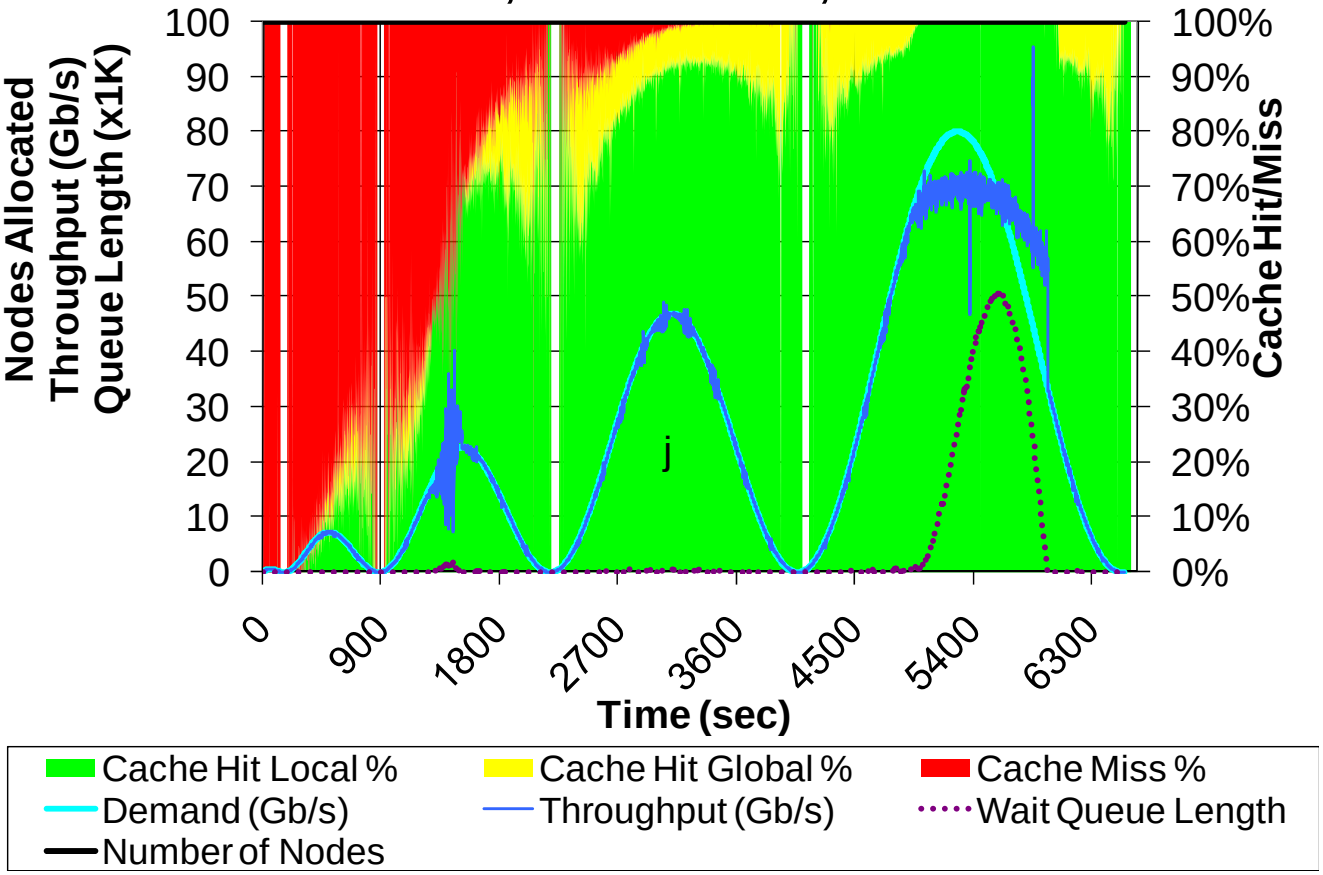
Sine-Wave Workload First-available (GPFS)

- GPFS ➔ 5.7 hrs, ~8Gb/s, 1138 CPU hrs



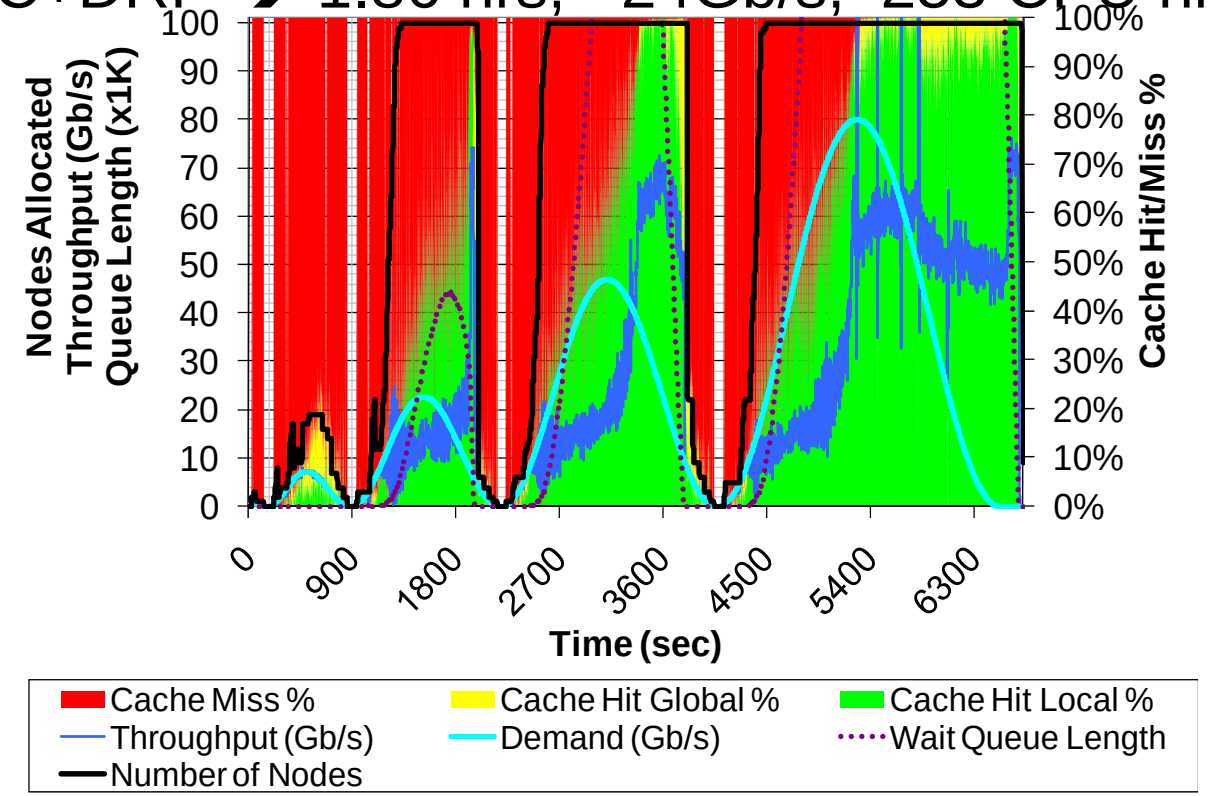
Sine-Wave Workload
Good-cache-compute and SRP

- GPFS ➔ 5.7 hrs, ~8Gb/s, 1138 CPU hrs
- GCC+SRP ➔ 1.8 hrs, ~25Gb/s, 361 CPU hrs



Sine-Wave Workload Good-cache-compute and DRP

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



All-Pairs Workload

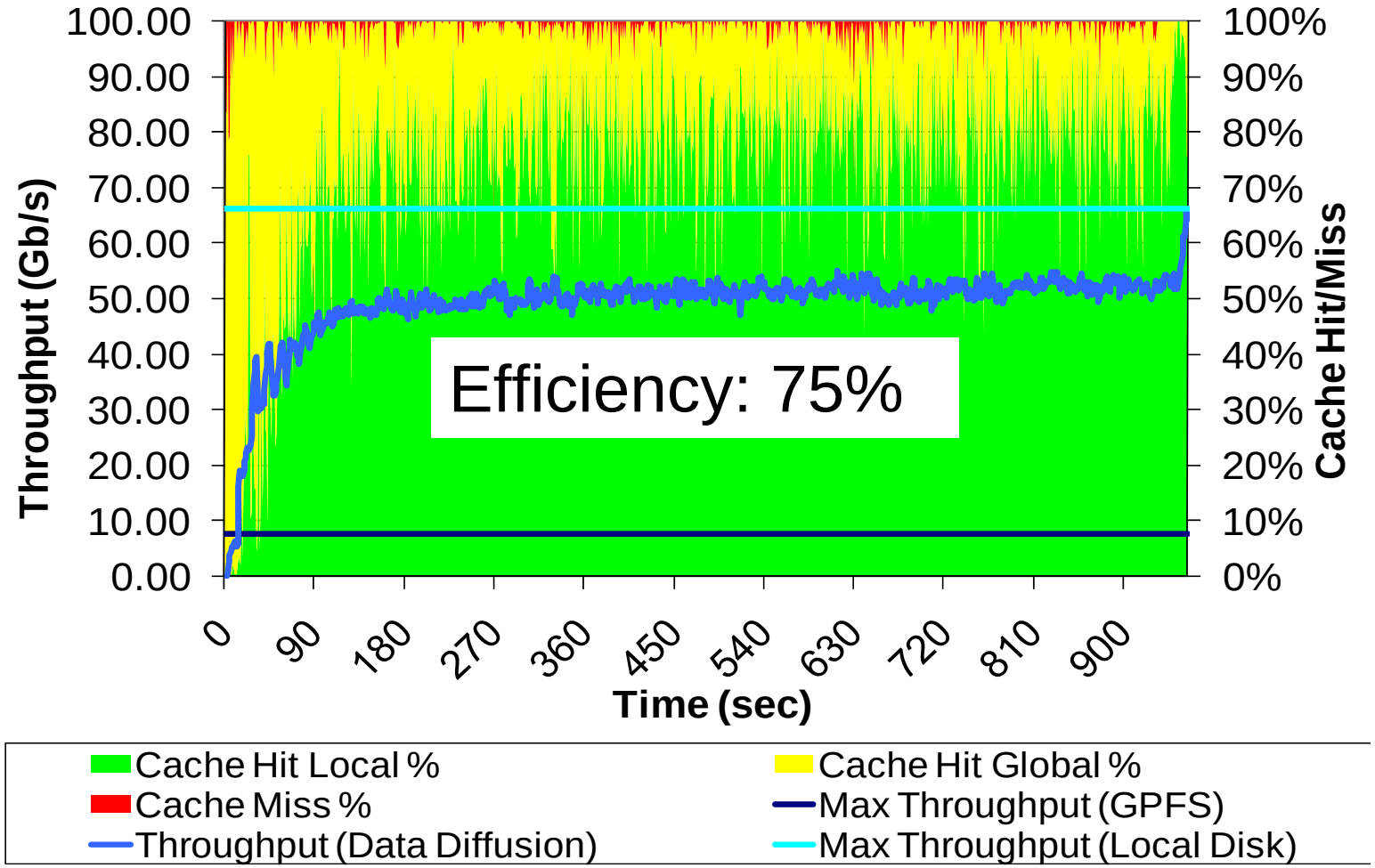
- 500x500
  - 250K tasks
  - 24MB reads
  - 100ms compute
  - 200 CPUs
- 1000x1000
  - 1M tasks
  - 24MB reads
  - 4sec compute
  - 4096 CPUs
- Ideal case:
  - 6505 sec
  - 80Gb/s peak throughput

- All-Pairs(set A, set B, function F) returns matrix M:
- Compare all elements of set A to all elements of set B via function F, yielding matrix M, such that

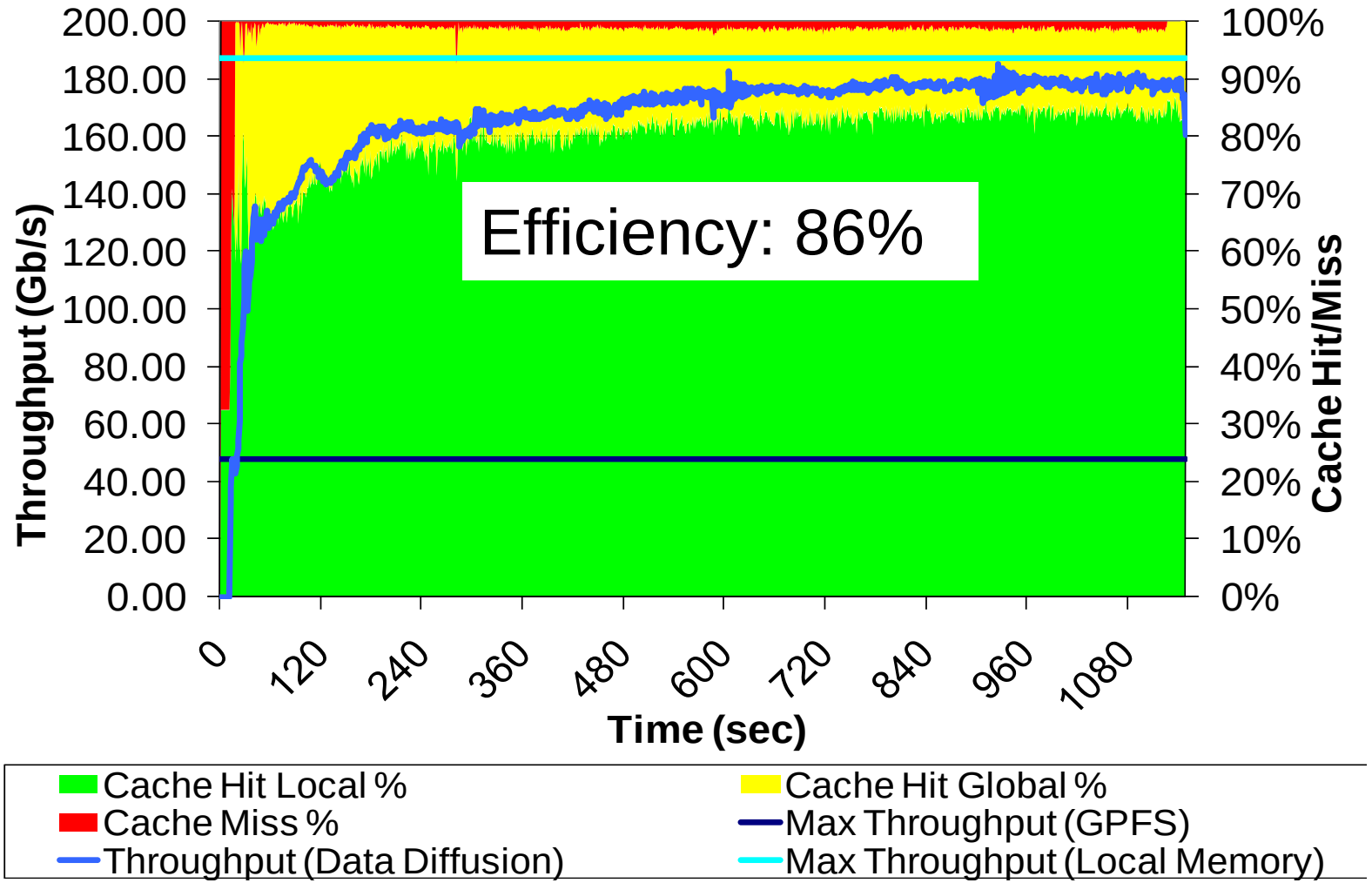
M[i,j] = F(A[i],B[j])

```
1 foreach $i in A
2       foreach $j in B
3               submit_job F $i $j
4       end
5 end
```

All-Pairs Workload 500x500 on 200 CPUs



All-Pairs Workload
1000x1000 on 4K emulated CPUs

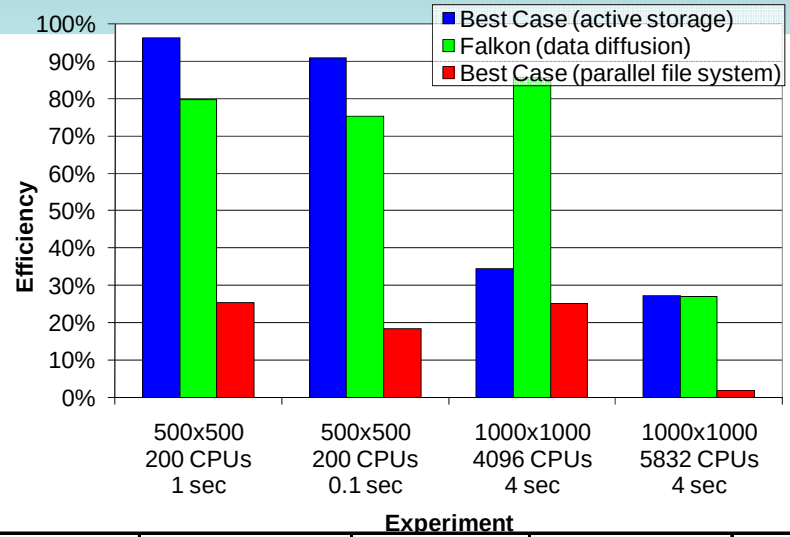


[DIDC09] "Towards Data Intensive Many-Task Computing", under review

All-Pairs Workload
Data Diffusion vs. Active Storage

- Pull vs. Push
  - Data Diffusion
    - Pulls ***task*** working set
    - Incremental spanning forest
  - Active Storage:
    - Pushes ***workload*** working set to all nodes
    - Static spanning tree

Christopher Moretti, Douglas Thain, University of Notre Dame



Experiment	Approach	Local Disk/Memory (GB)	Network (node-to-node) (GB)	Shared File System (GB)
500x500 200 CPUs 1 sec	Best Case (active storage)	6000	1536	12
	Falkon (data diffusion)	6000	1698	34
500x500 200 CPUs 0.1 sec	Best Case (active storage)	6000	1536	12
	Falkon (data diffusion)	6000	1528	62
1000x1000 4096 CPUs 4 sec	Best Case (active storage)	24000	12288	24
	Falkon (data diffusion)	24000	4676	384
1000x1000 5832 CPUs	Best Case (active storage)	24000	12288	24

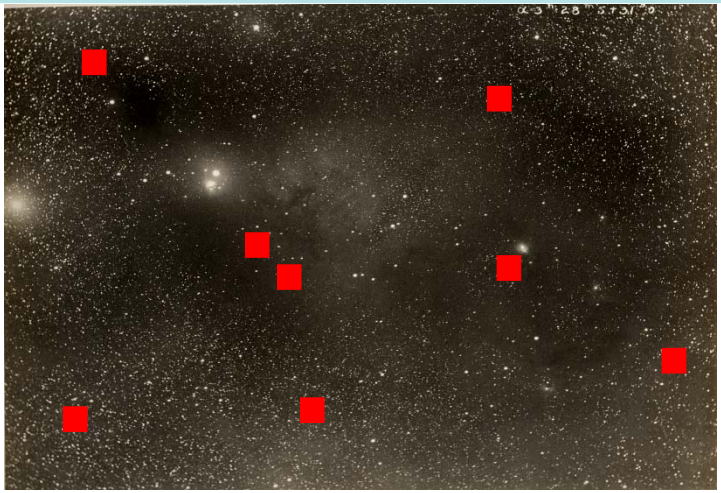
[DIDC09] "Towards Data Intensive Many-Task Computing", under review

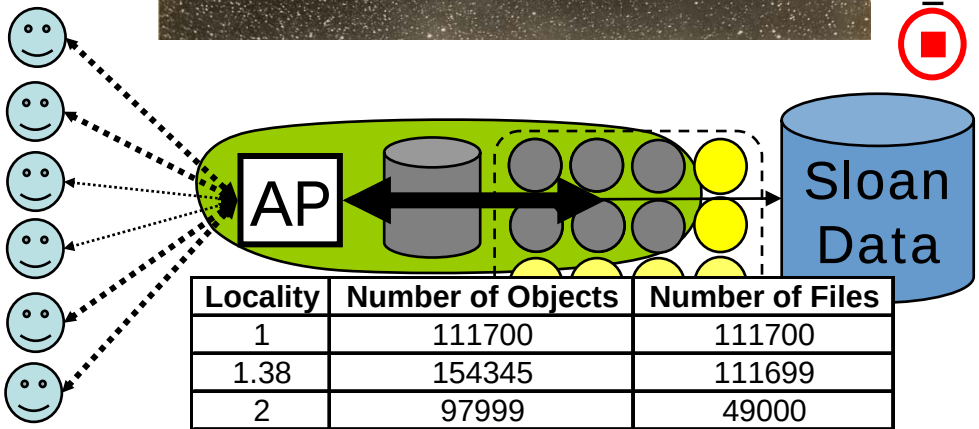
All-Pairs Workload
Data Diffusion vs. Active Storage

- Best to use active storage if
  - Slow data source
  - Workload working set fits on local node storage
- Best to use data diffusion if
  - Medium to fast data source
  - Task working set << workload working set
  - Task working set fits on local node storage
- If task working set does not fit on local node storage
  - Use parallel file system (i.e. GPFS, Lustre, PVFS, etc)

Image Stacking Workload Astronomy Application

- Purpose
  - On-demand “stacks” of random locations within ~10TB dataset
- Challenge
  - Processing Costs:
    - O(100ms) per object
  - Data Intensive:
    - 40MB:1sec
  - Rapid access to 10-10K “random” files
  - Time-varying load





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

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”
[TG06] “AstroPortal: A Science Gateway for Large-scale Astronomy Data Analysis”

Image Stacking Workload Profiling

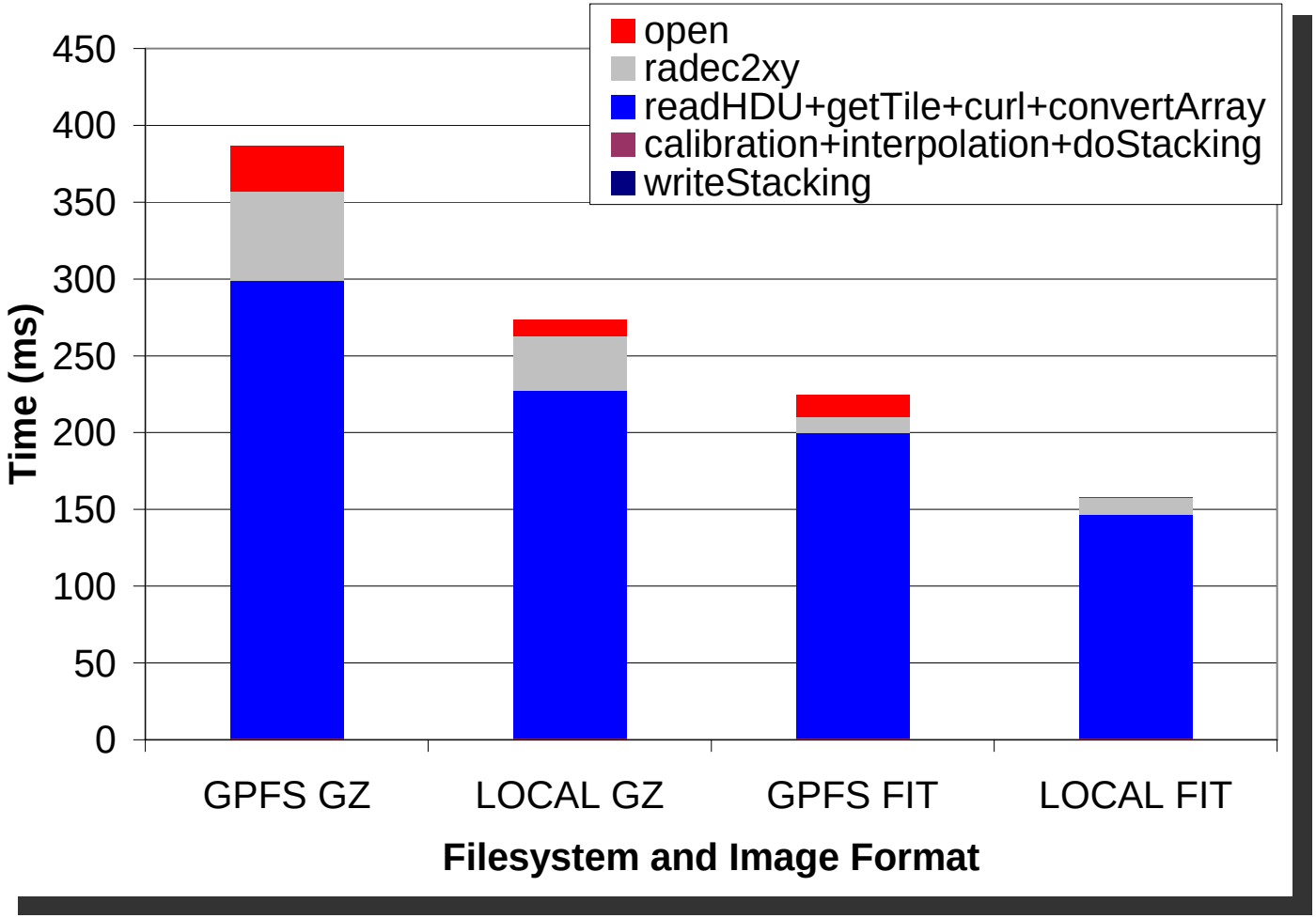
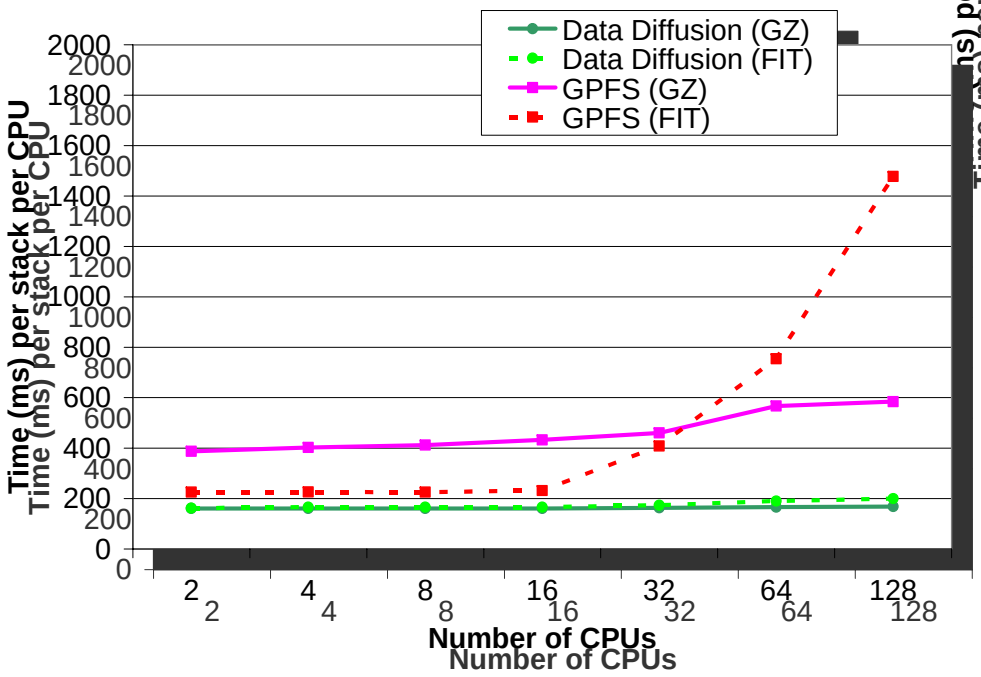
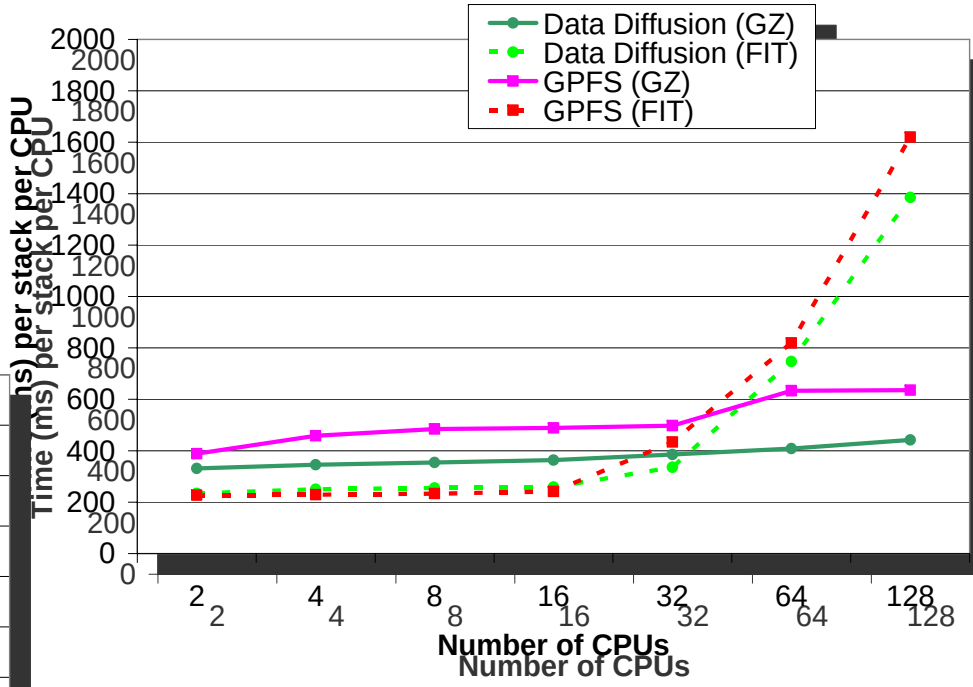


Image Stacking Workload Varying Scale

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

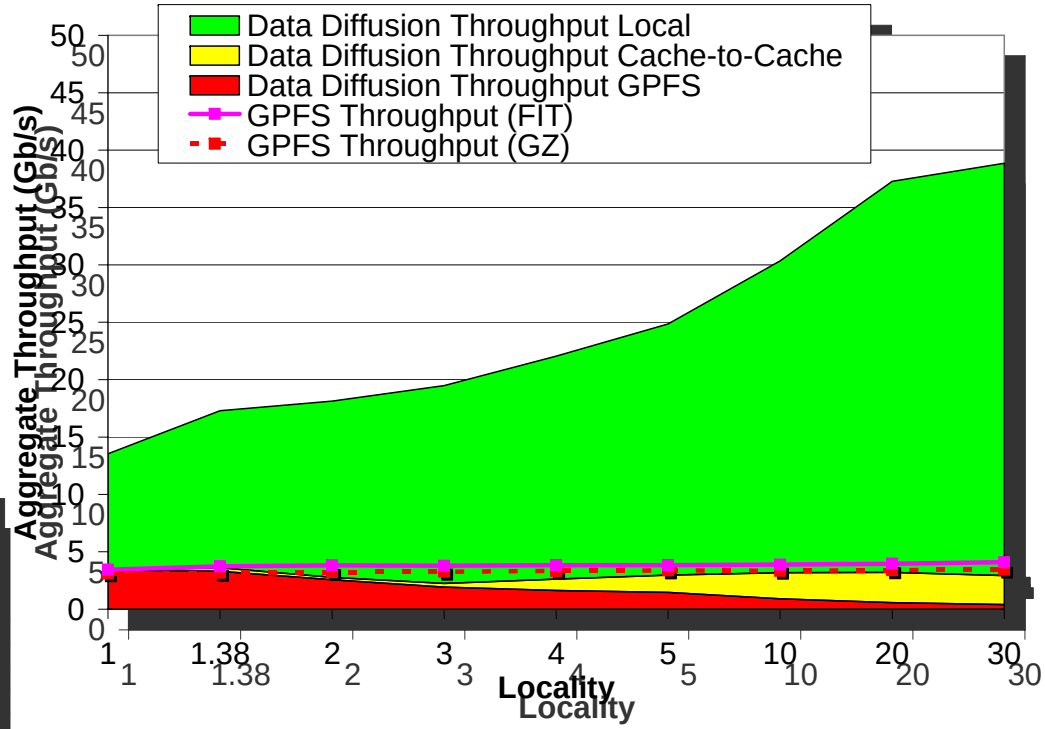


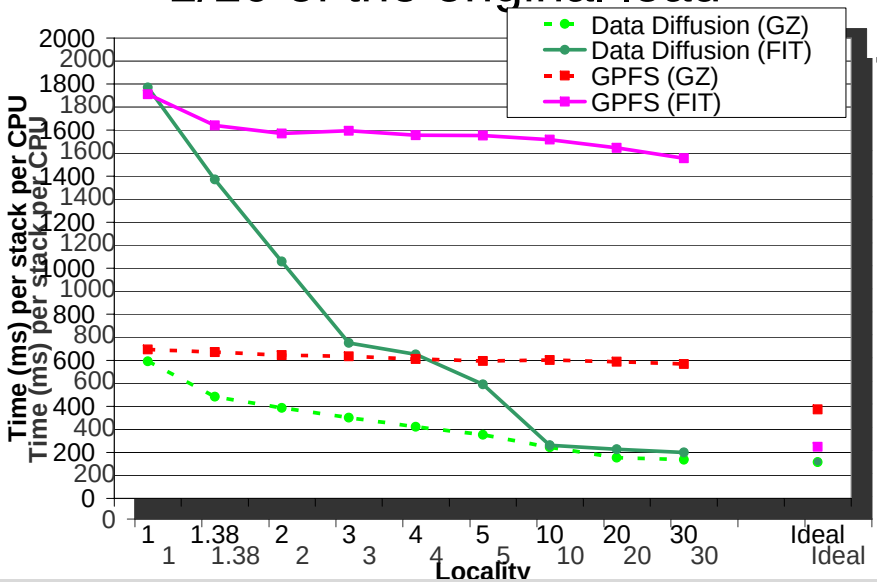


➔High data locality
- Near perfect scalability

Image Stacking Workload Varying Locality

- 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

[DADC08] "Accelerating Large-scale Data Exploration through Data Diffusion"

Limitations of Data Diffusion

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

Data Diffusion vs. Others

- [Ghemawat03,Dean04]: MapReduce+GFS
- [Bialecki05]: Hadoop+HDFS
- [Gu06]: Sphere+Sector
- [Tatebe04]: Gfarm
- [Chervenak04]: RLS, DRS
- [Kosar06]: Stork

- **Conclusions**
  - *None focused on the co-location of storage and generic black box computations with data-aware scheduling while operating in a dynamic elastic environment*
  - *Swift + Falkon + Data Diffusion is arguably a more generic and powerful solution than MapReduce*

Contributions

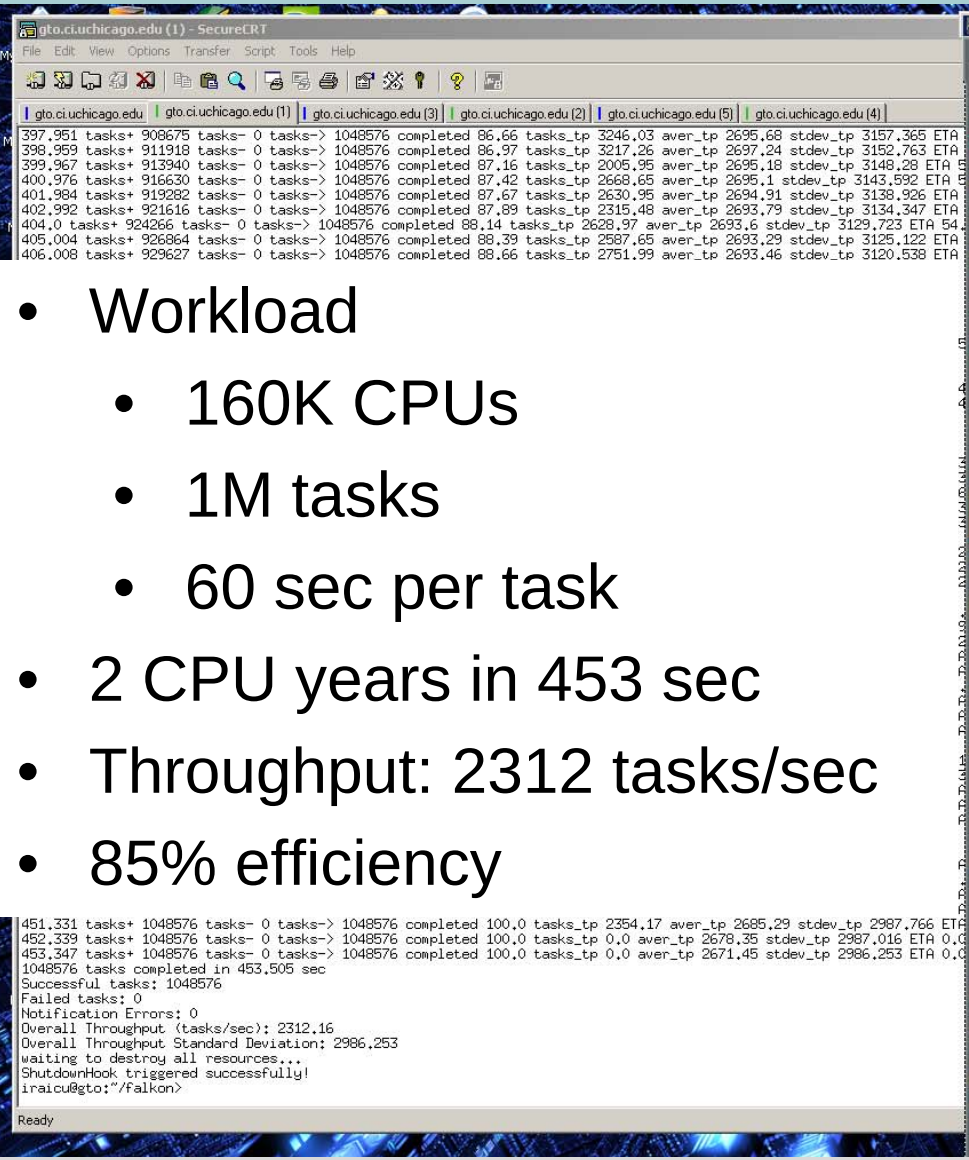
- Identified that data locality is crucial to the efficient use of large scale distributed systems for data-intensive applications ➔ Data Diffusion
  - Integrated streamlined task dispatching with data aware scheduling policies
  - Heuristics to maximize real world performance
  - Suitable for varying, data-intensive workloads
  - Proof of O(NM) Competitive Caching

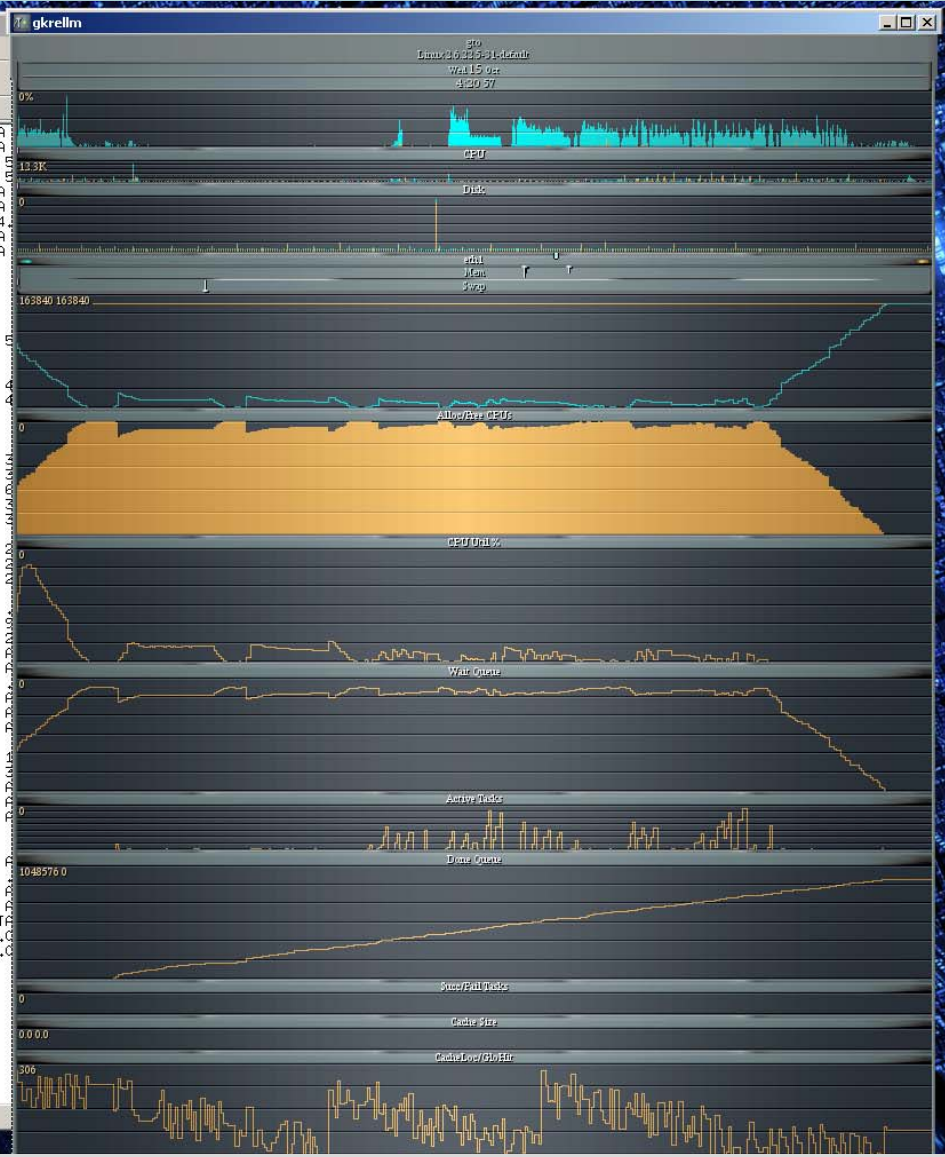
Falkon Project

- Falkon is a real system
  - Late 2005: Initial prototype, AstroPortal
  - January 2007: Falkon v0
  - November 2007: Globus incubator project v0.1
    - http://dev.globus.org/wiki/Incubator/Falkon
  - February 2009: Globus incubator project v0.9
- Implemented in Java (~20K lines of code) and C (~1K lines of code)
  - Open source: svn co https://svn.globus.org/repos/falkon
- Source code contributors (beside myself)
  - Yong Zhao, Zhao Zhang, Ben Clifford, Mihael Hategan

[Globus07] "Falkon: A Proposal for Project Globus Incubation"

Falkon Monitoring

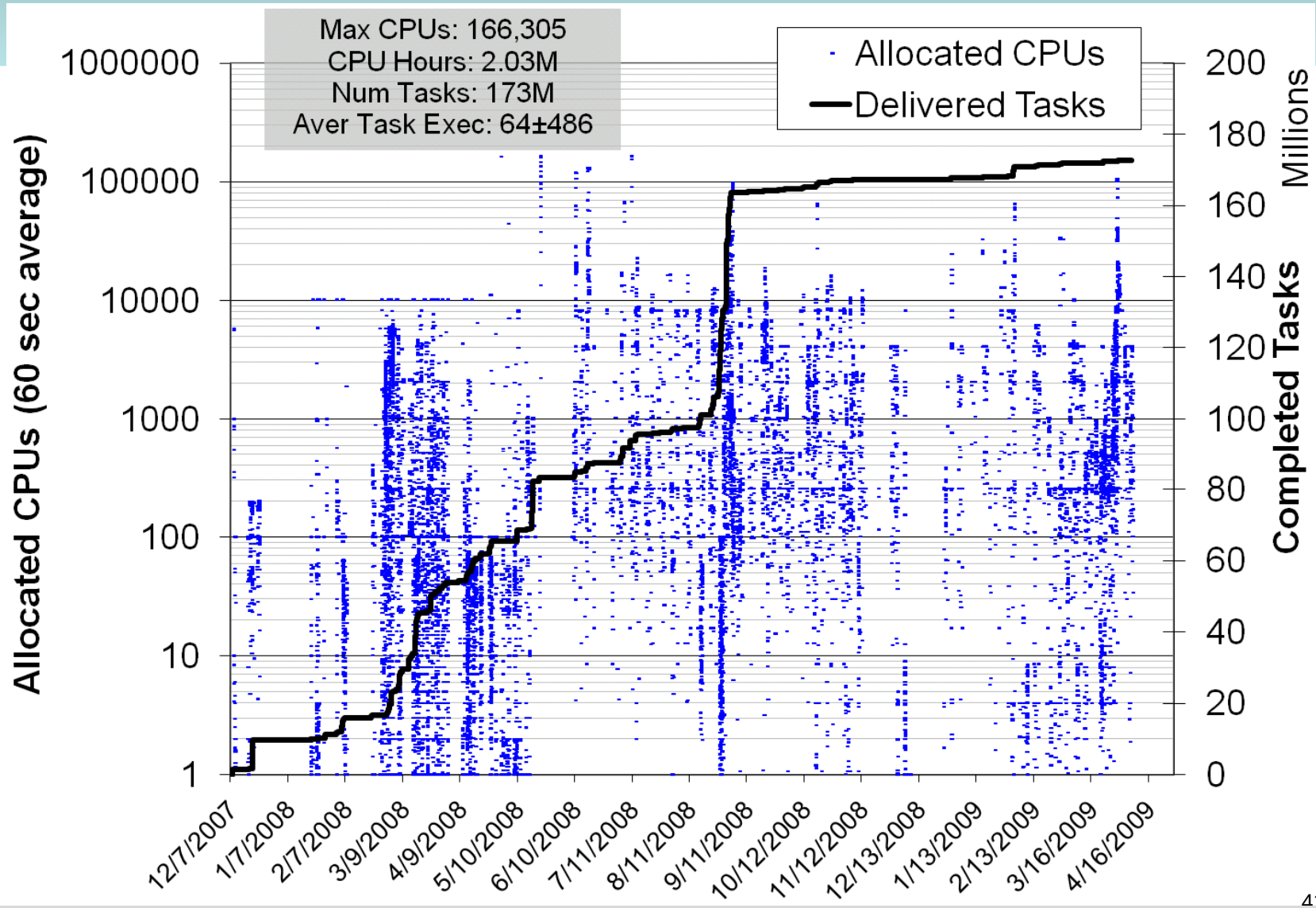




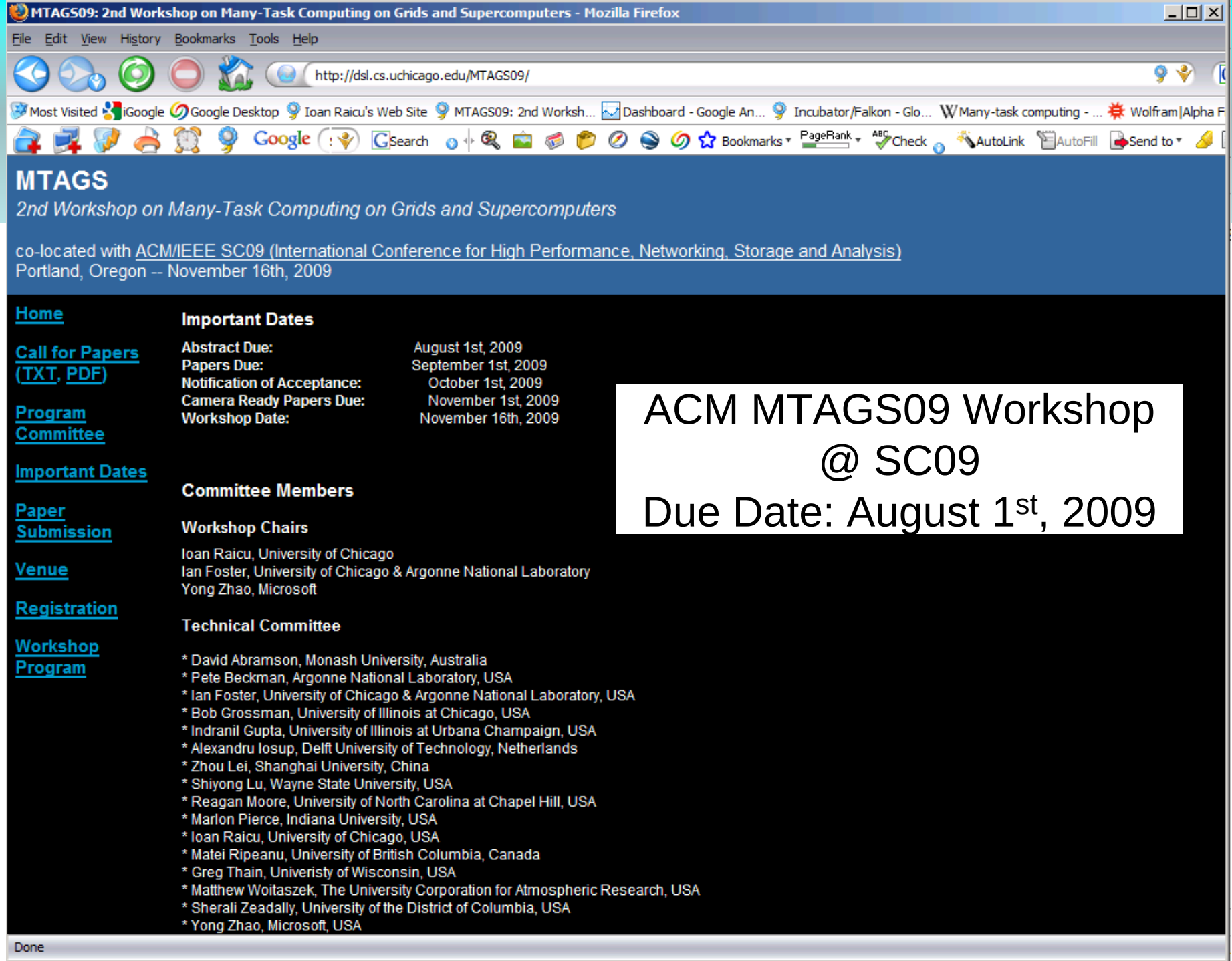
- Workload
  - 160K CPUs
  - 1M tasks
  - 60 sec per task
- 2 CPU years in 453 sec
- Throughput: 2312 tasks/sec
- 85% efficiency

[TPDS09] “Middleware Support for Many-Task Computing”, under preparation

Falkon Activity History (16 months)



[TPDS09] "Middleware Support for Many-Task Computing", under preparation



ACM MTAGS09 Workshop
@ SC09
Due Date: August 1st, 2009

IEEE Transactions on Parallel and Distributed Systems: Special Issue on Many-Task Computing - Mozilla Firefox

File Edit View History Bookmarks Tools Help

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

Most Visited iGoogle Google Desktop Ioan Raicu's Web Site MTAGS09: 2nd Worksh... Dashboard - Google An... Incubator/Falkon - Glo... Many-task computing - ... Wolfram|Alpha F

Google Search Bookmarks PageRank Check AutoLink AutoFill Send to

IEEE Transactions on Parallel and Distributed Systems

Special Issue on Many-Task Computing

Home

Call for Papers (TXT, PDF)

Important Dates

Paper Submission

Abstract Due:	December 1st, 2009
Papers Due:	December 21st, 2009
First Round Decisions:	February 22nd, 2010
Major Revisions if needed:	April 19th, 2010
Second Round Decisions:	May 24th, 2010
Minor Revisions if needed:	June 7th, 2010
Final Decision:	June 21st, 2010
Publication Date:	November, 2010

IEEE TPDS Journal
Special Issue on MTC
Due Date: December 1st, 2009

Special Issue Guest Editors

Ian Foster, University of Chicago & Argonne National Laboratory
Ioan Raicu, University of Chicago
Yong Zhao, Microsoft

Dr. Ian Foster is the Associate Division Director and a Senior Scientist in the Mathematics and Computer Science Division at Argonne Na Distributed Systems Laboratory, and he is an Arthur Holly Compton Professor in the Department of Computer Science at the University o Grid Forum and with the Globus Alliance as an open source strategist. In 2006, he was appointed director of the Computation Institute, a and Argonne. An earlier project, Strand, received the British Computer Society Award for technical innovation. His research resulted in the algorithms for high-performance distributed computing and parallel computing. As a result he is denoted as "the father of the Grid". Foste the I-WAY wide-area distributed computing experiment, which connected supercomputers, databases and other high-end resources at 1 labs, the Distributed Systems Laboratory is the nexus of the multi-institute Globus Project, a research and development effort that encour advances necessary for engineering, business and other fields. Furthermore the Computation Institute addresses many of the most cha problems facing Grid implementations today. In 2004, he founded Univa Corporation, which was merged with United Devices in 2007 an honors include the Lovelace Medal of the British Computer Society, the Gordon Bell Prize for high-performance computing (2001), as wel American Association for the Advancement of Science in 2003. Dr. Foster also serves as PI or Co-PI on projects connected to the DOE g Computational Science Alliance, the NASA Information Power Grid project, the NSF Grid Physics Network, GRIDS Center, and Internation other DOE and NSF programs. His research is supported by DOE, NSF, NASA, Microsoft, and IBM.

Dr. Ioan Raicu holds a Ph.D. in Computer Science from University of Chicago under the guidance of Dr. Ian Foster. He is a 3-year award win Research Center. His research work and interests are in the general area of distributed systems. His dissertation work focused on this rela (MTC), which aims to bridge the gap between two predominant paradigms from distributed systems, High-Throughput Computing (HTC) an for the last five years focused on defining and exploring both the theory and practical aspects of realizing MTC across a wide range of large-s interested in efficient task dispatch and execution systems, resource provisioning, data management, scheduling, and performance evaluat funded by the NASA Ames Research Center Graduate Student Research Program, as well as the DOE Office of Advanced Scientific Comput resource management in large scale distributed systems with a focus on many-task computing, data intensive computing, cloud computing

Done

More Information

- More information:
  - Other publications: http://people.cs.uchicago.edu/~iraicu/
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - Swift: http://www.ci.uchicago.edu/swift/index.php
- Funding:
  - **NASA**: Ames Research Center, GSRP
  - **DOE**: Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy
  - **NSF**: TeraGrid
- Relevant activities:
  - ACM MTAGS09 Workshop at Supercomputing 2009
    - http://dsl.cs.uchicago.edu/MTAGS09/
  - Special Issue on MTC in IEEE TPDS Journal
    - http://dsl.cs.uchicago.edu/TPDS_MTC/