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

Ioan Raicu
Distributed Systems Laboratory
Computational Institute
University of Chicago

Slides based on ACM HPDC 2009 Slides

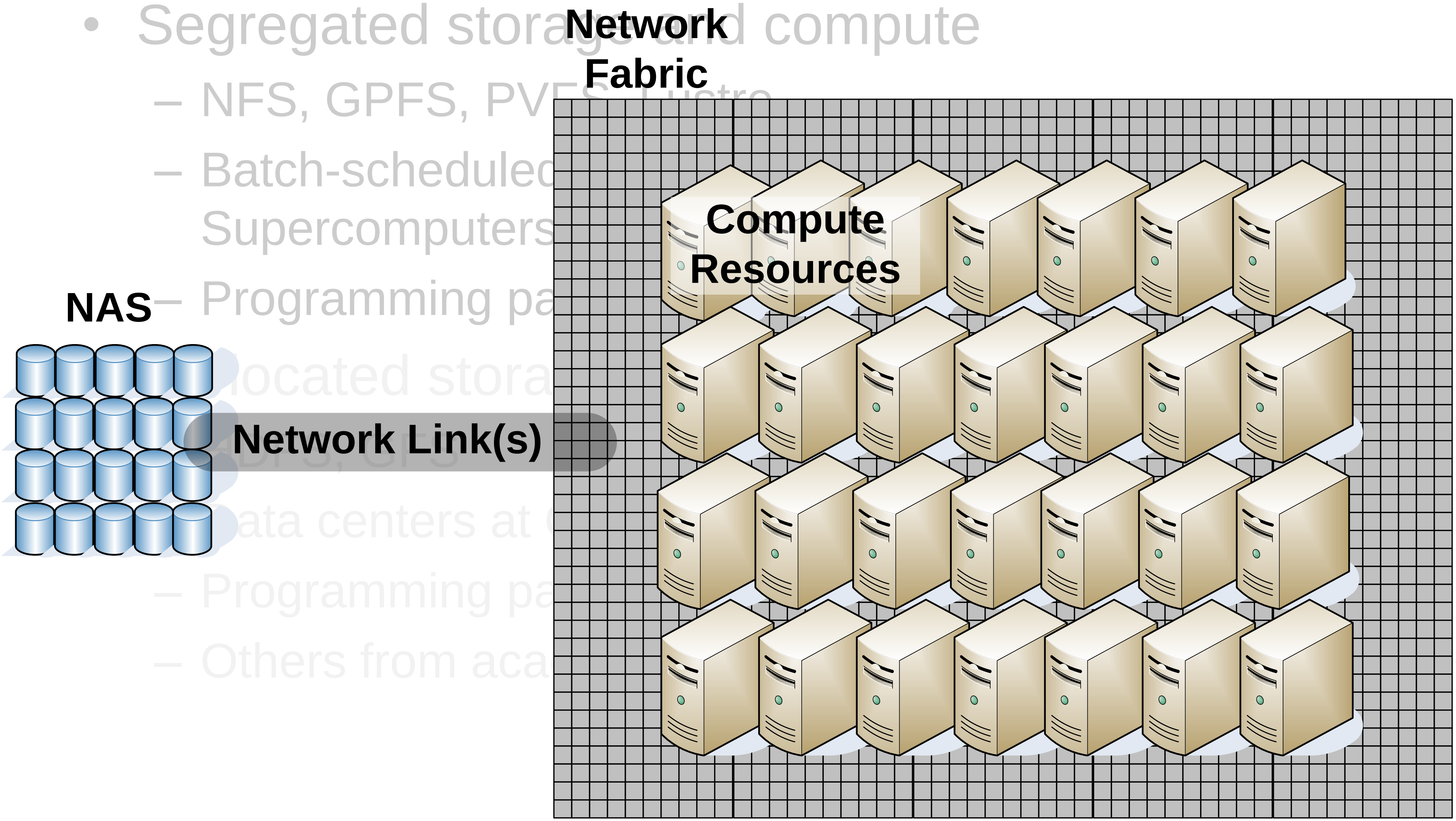
October 21st, 2013

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 storage and compute
  - NFS, GPFS, PVFS, Lustre
  - Batch-scheduled ... Supercomputers
  - Programming pa...
- ...located stora...
  - ...DFS, GFS
- ...ata centers at ...
  - Programming pa...
  - Others from aca...

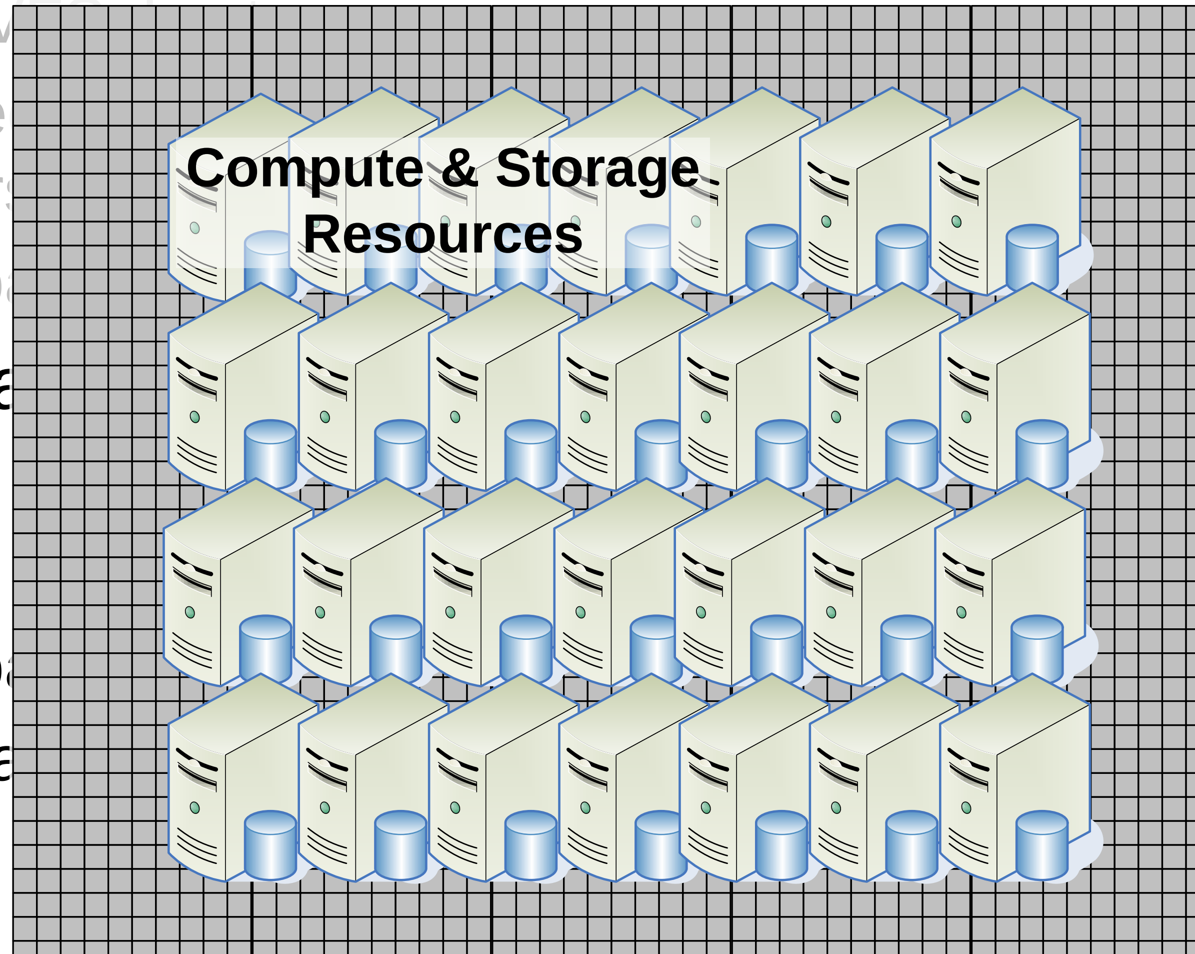


State of the Art: Storage Systems

- 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

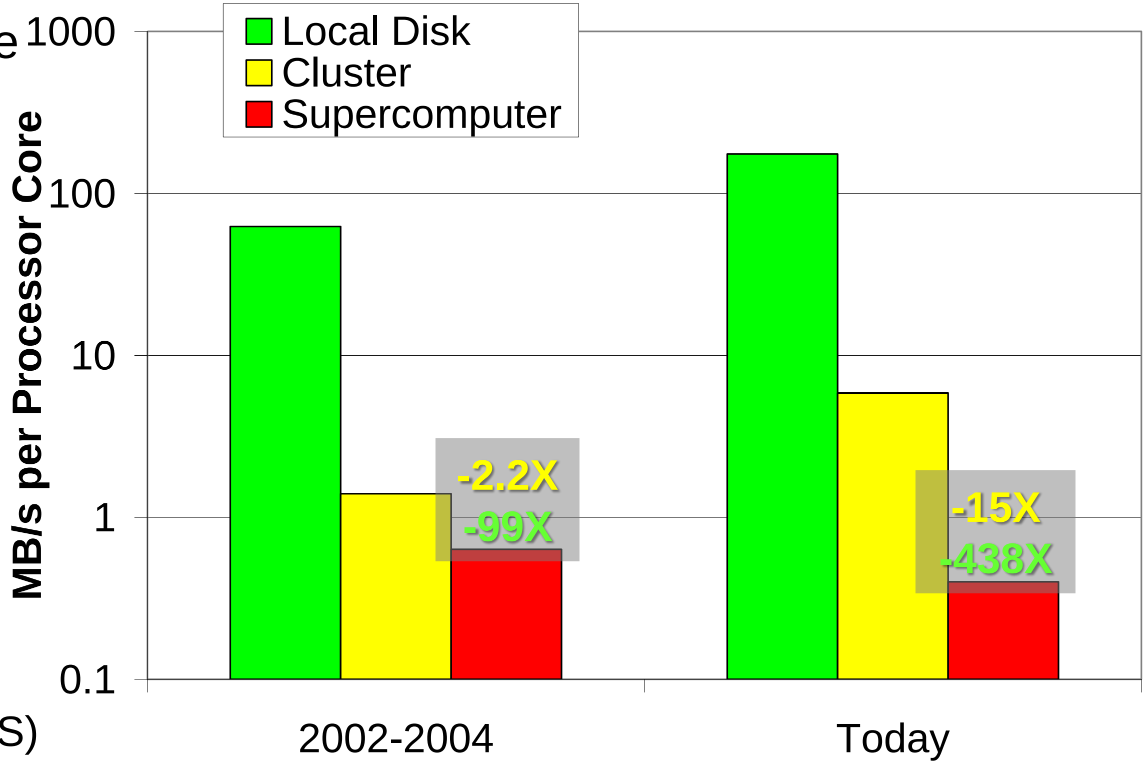
State of the Art: Storage Systems

- Segregated stor… d compute
  - 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

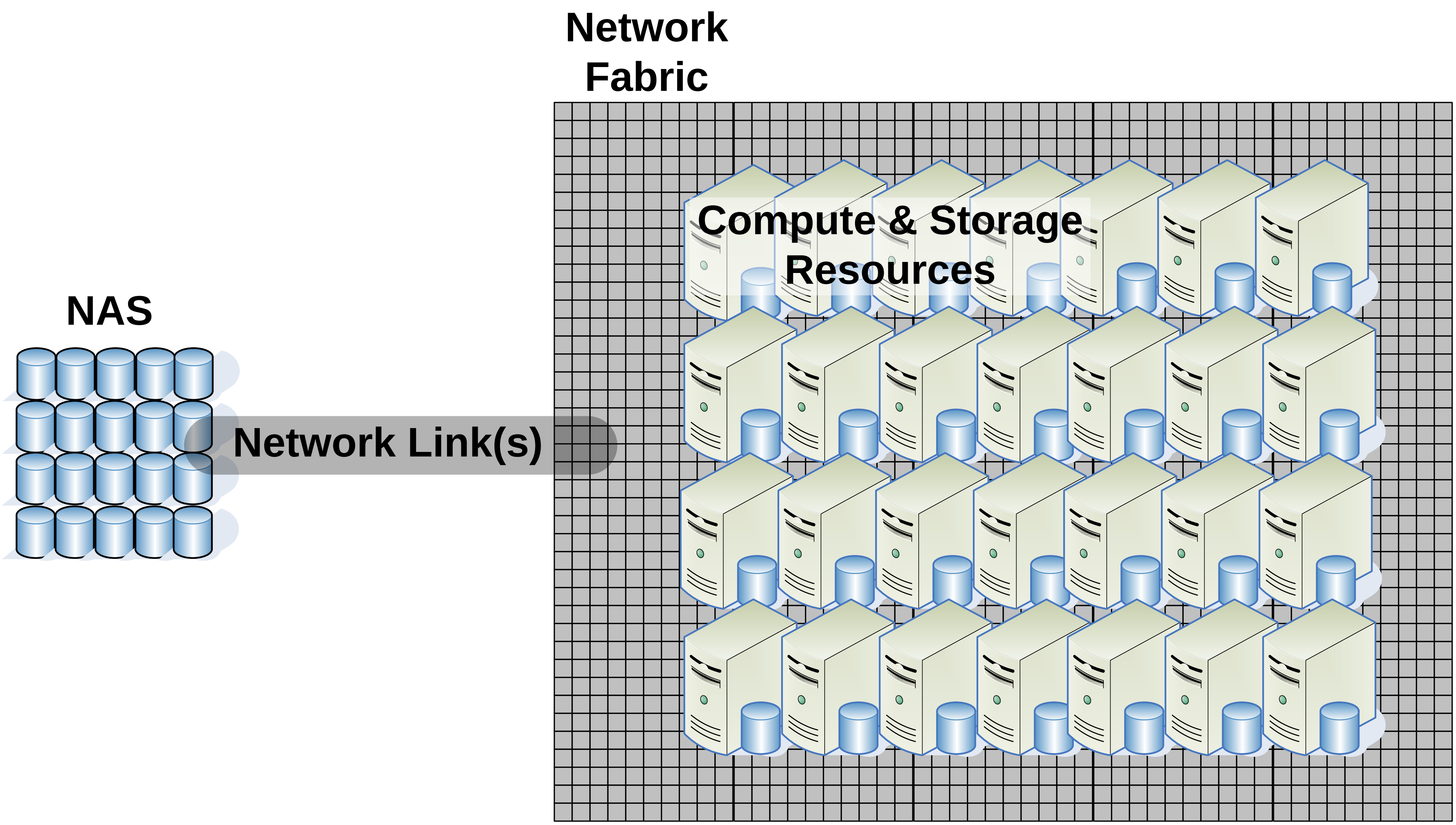
- Local Disk:
  - 2002-2004: ANL/UC TG Site (70GB SCSI)
  - Today: PADS (RAID-0, 6 drives 750GB SATA)
- Cluster:
  - 2002-2004: ANL/UC TG Site (GPFS, 8 servers, 1Gb/s each)
  - Today: PADS (GPFS, SAN)
- Supercomputer:
  - 2002-2004: IBM Blue Gene/L (GPFS)
  - Today: IBM Blue Gene/P (GPFS)



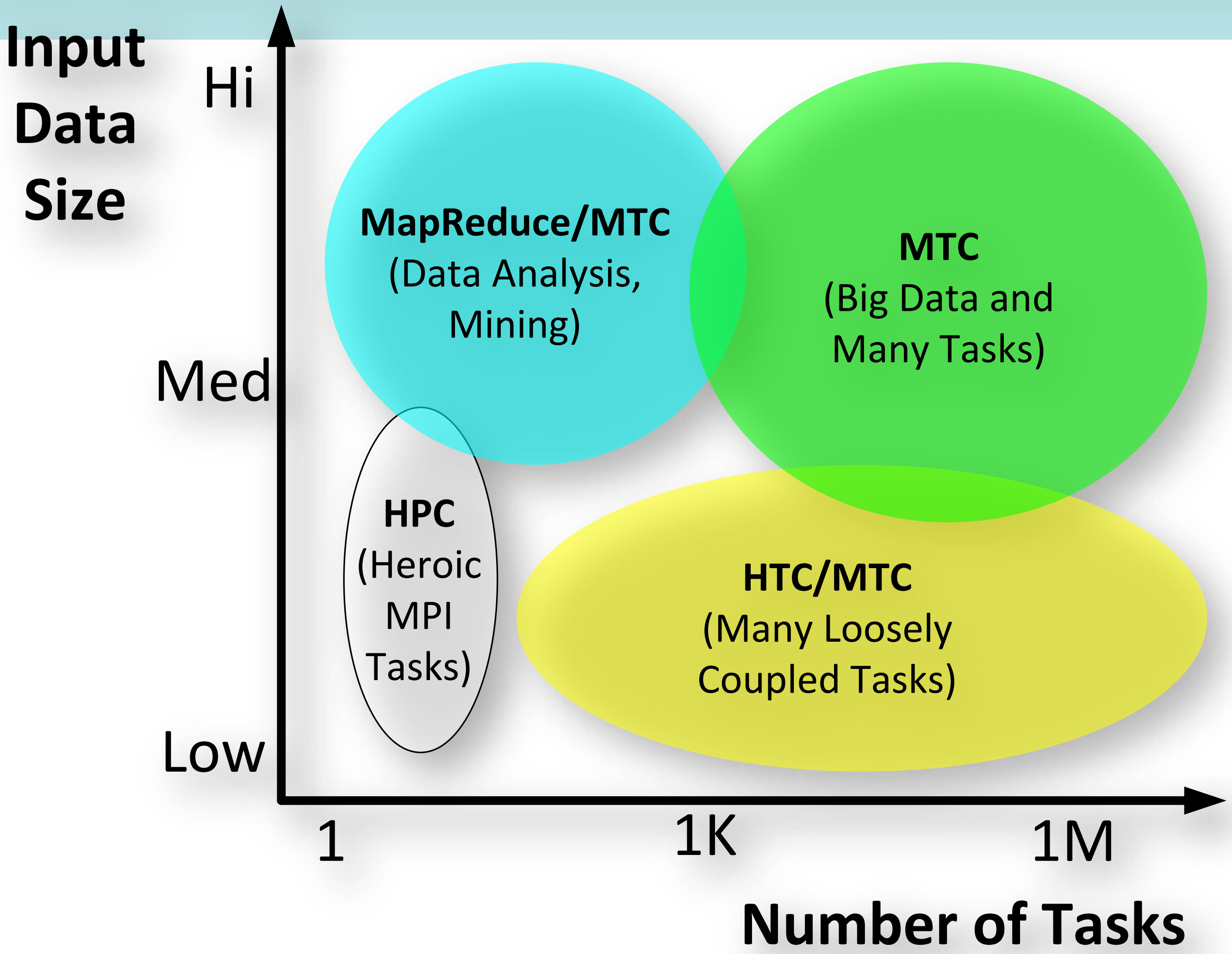
Question

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



[MTAGS08] "Many-Task Computing for Grids and Supercomputers"

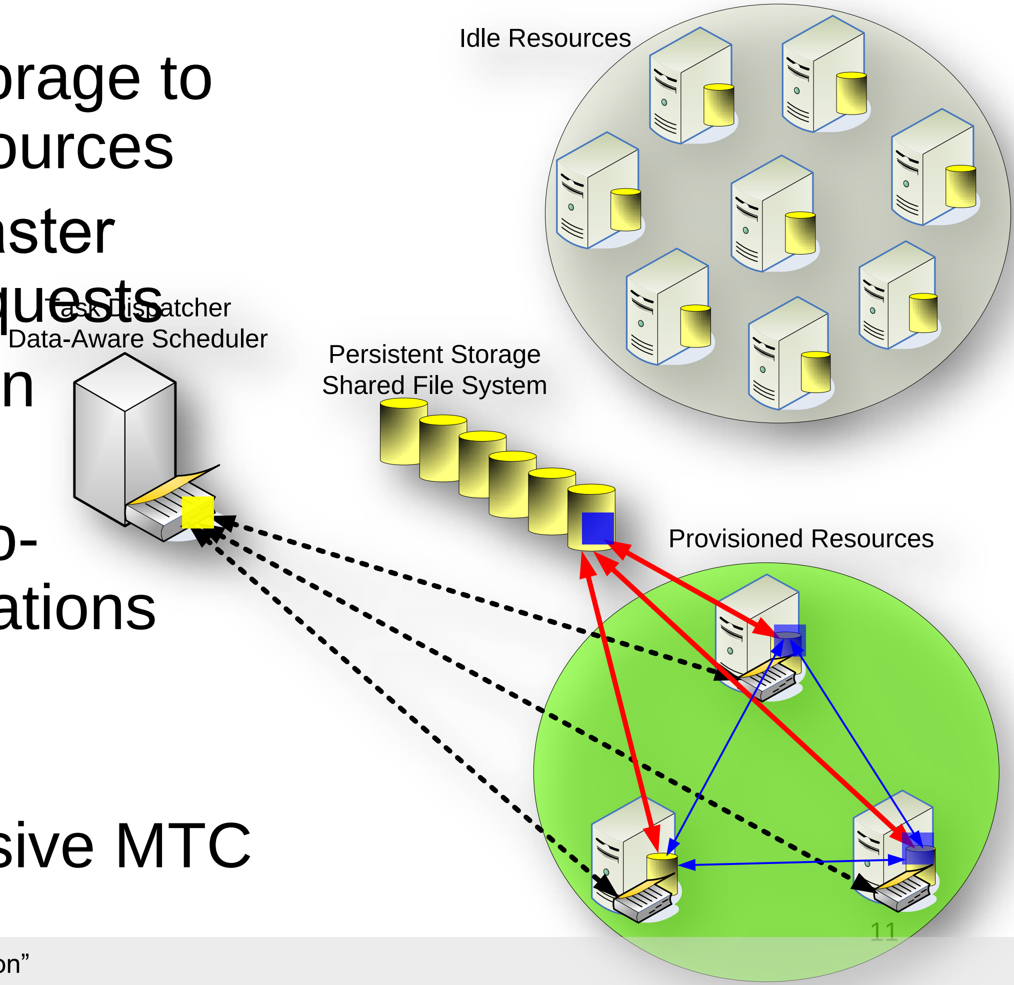
Hypothesis

"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

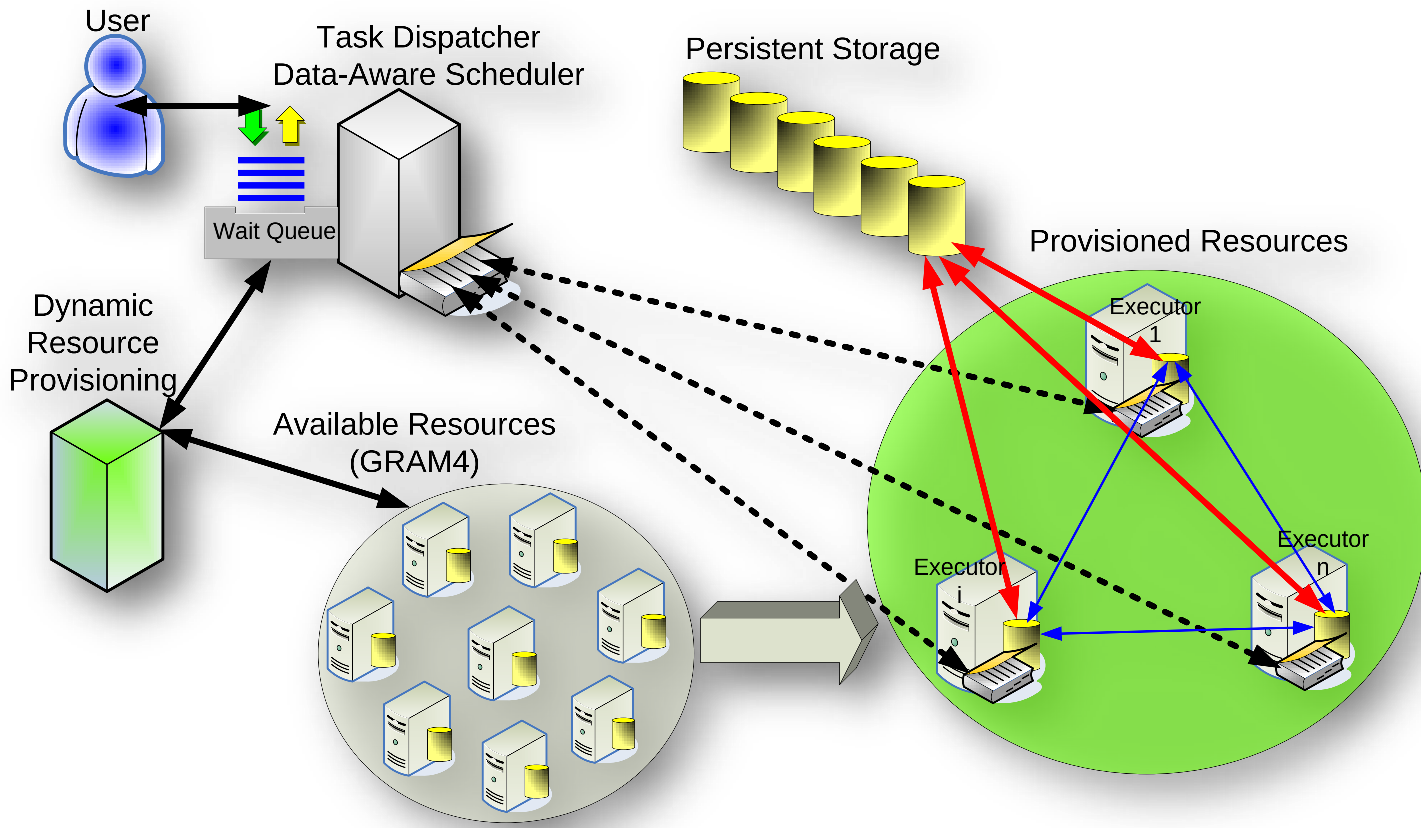
- Resource acquired in response to demand
- Data diffuse from archival storage to newly acquired transient resources
- Resource "caching" allows faster responses to subsequent requests
- Resources are released when demand drops
- Optimizes performance by co-scheduling data and computations
- Decrease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



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

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"

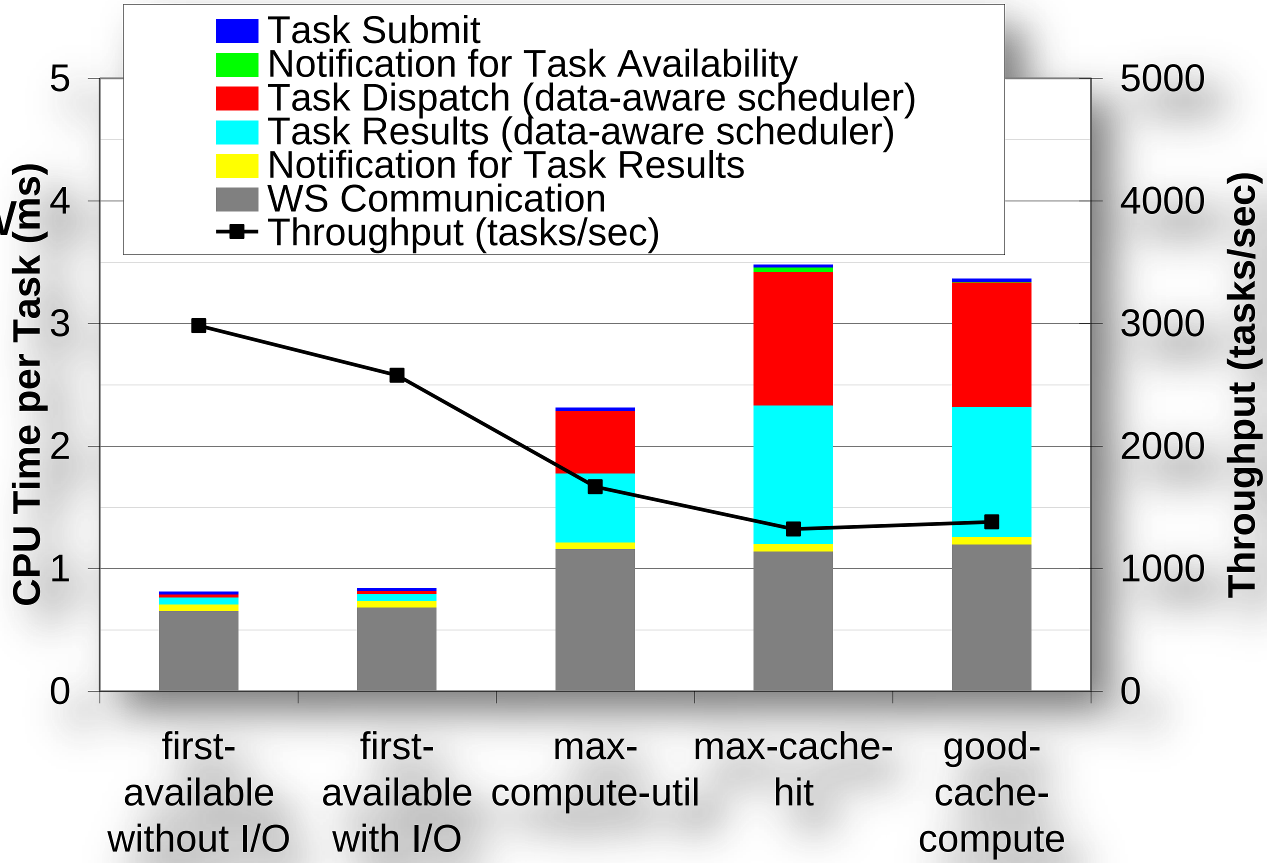
Scheduling Policies

- 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

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”

Data-Aware Scheduler Profiling

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



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

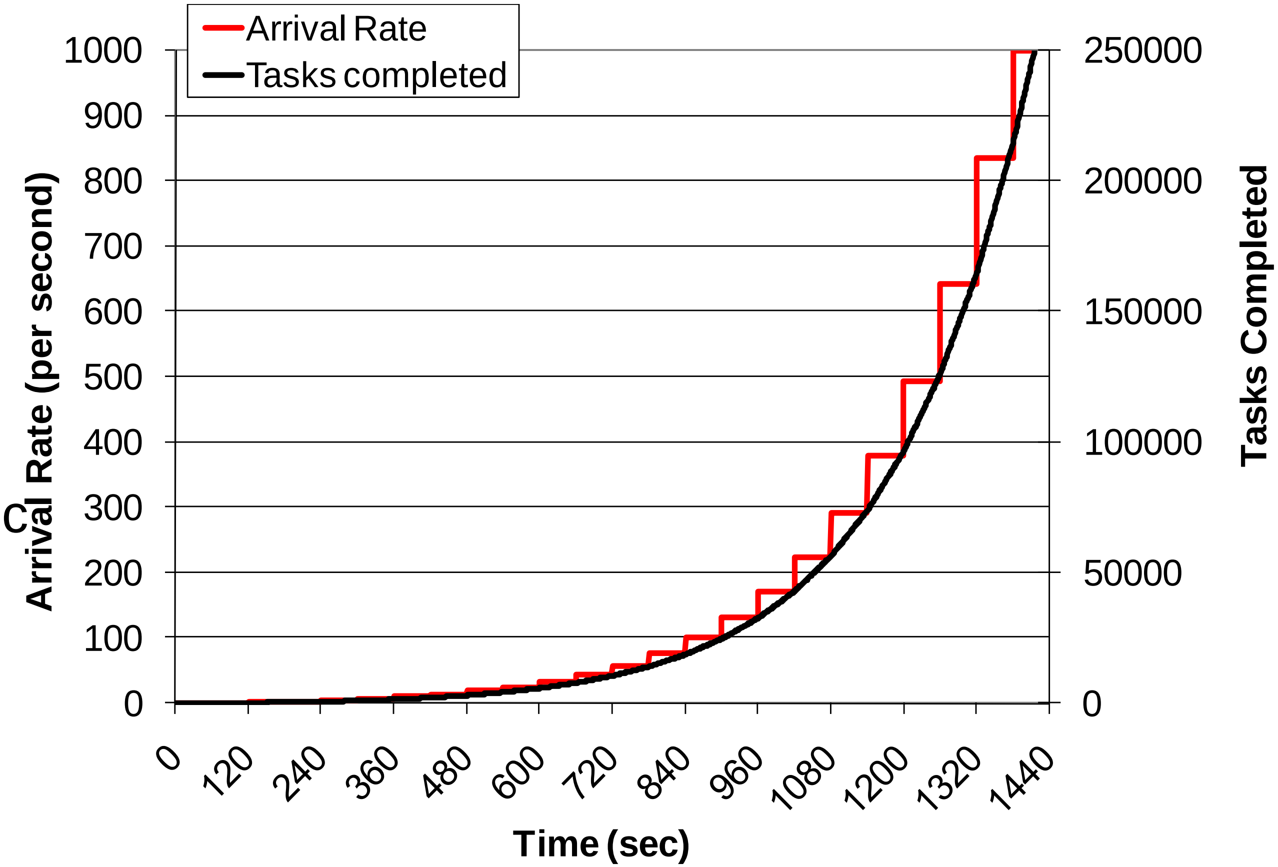
Workloads

- 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

[DADC08] “Accelerating Large-scale Data Exploration through Data Diffusion”

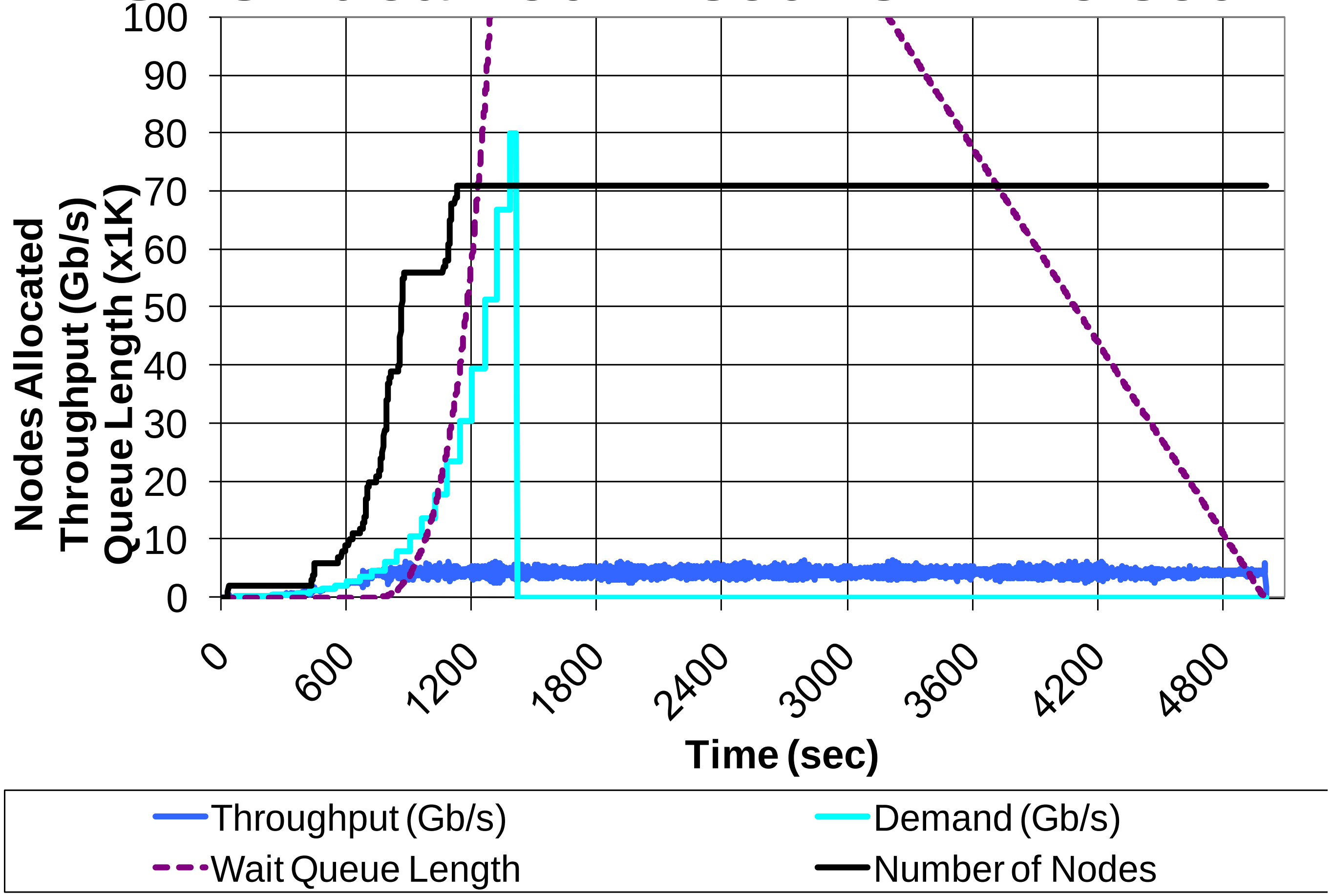
Monotonically Increasing Workload

- 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

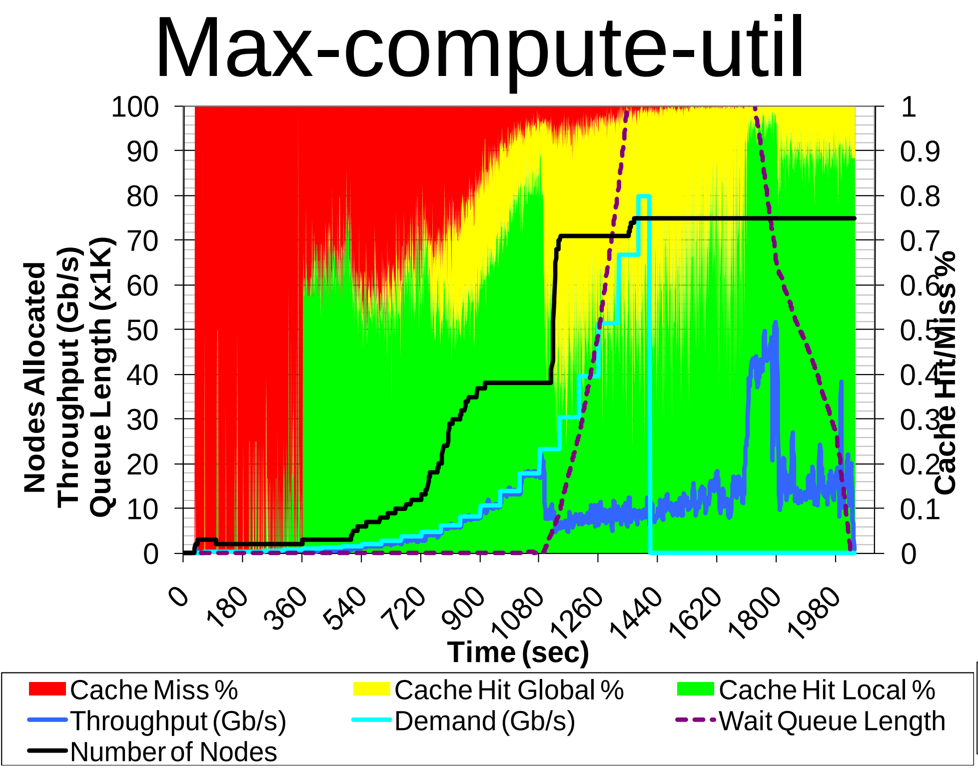


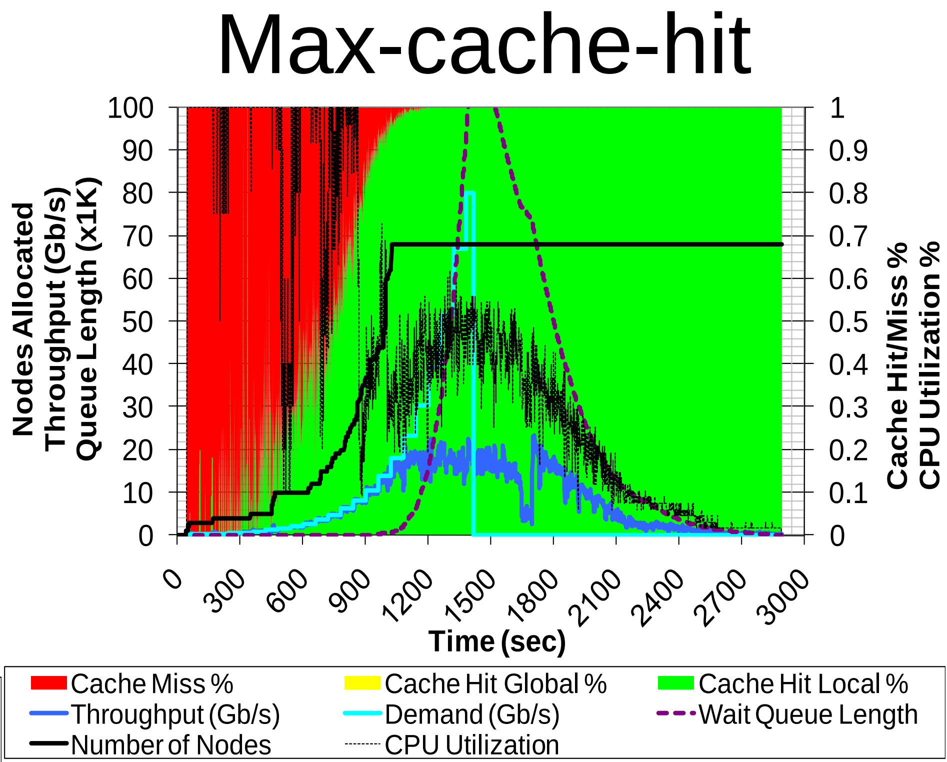
Monotonically Increasing Workload First-available (GPFS)

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

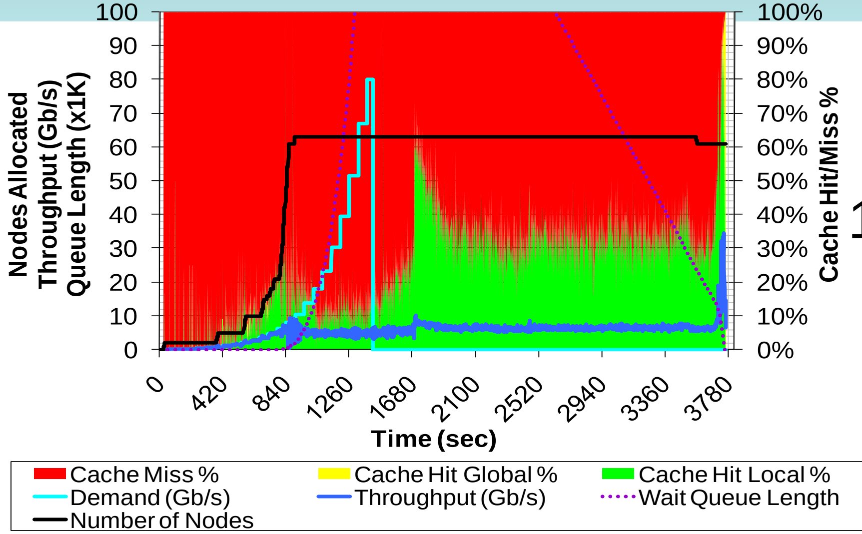


Monotonically Increasing Workload Max-compute-util & Max-cache-hit



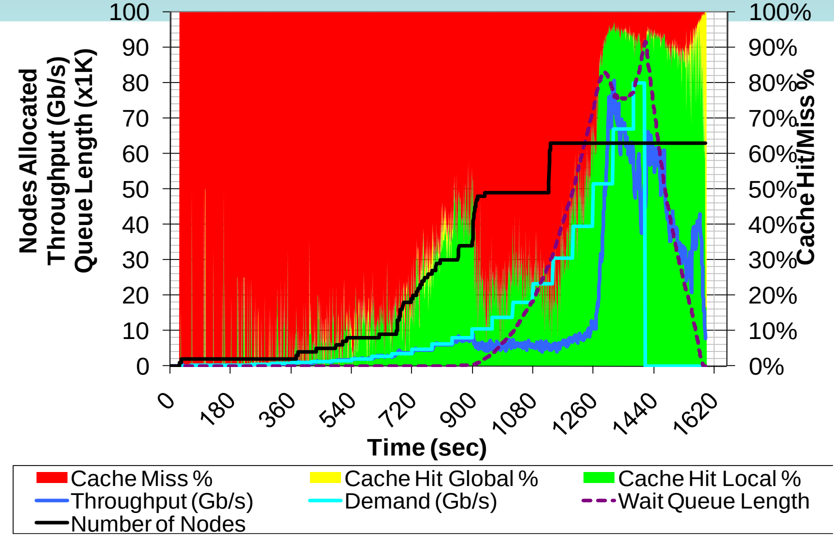


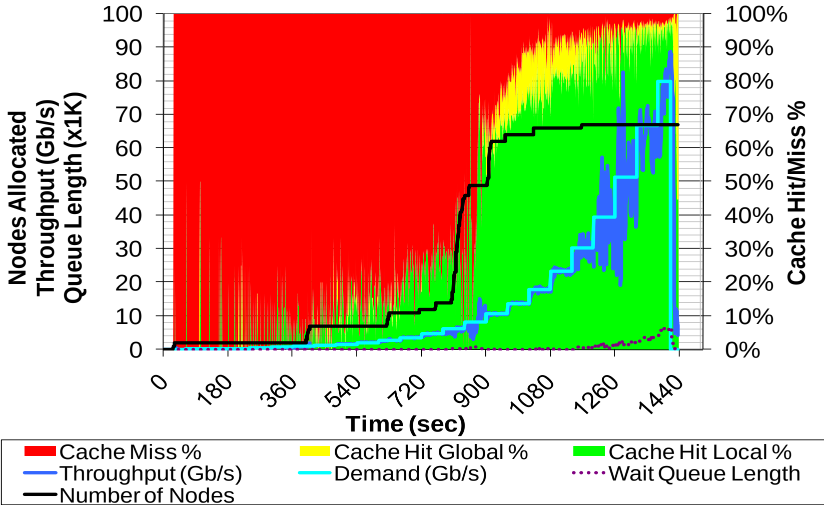
Monotonically Increasing Workload Good-cache-compute



← 1GB

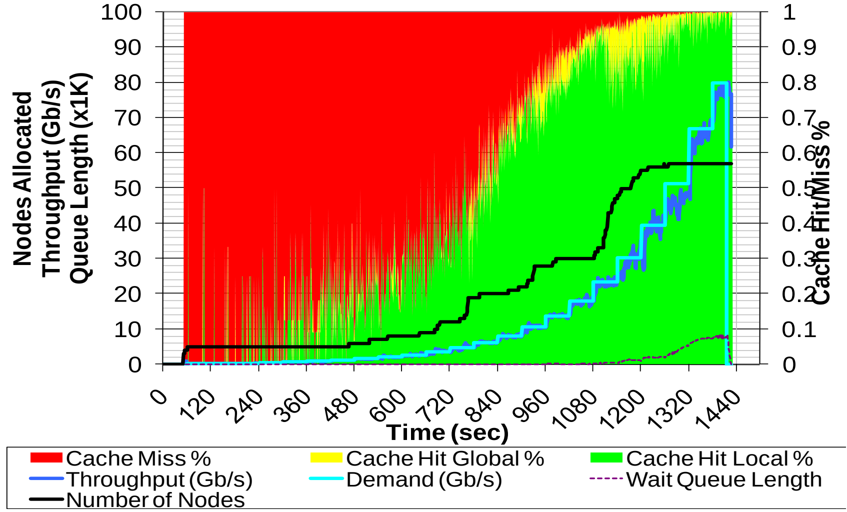
1.5GB →





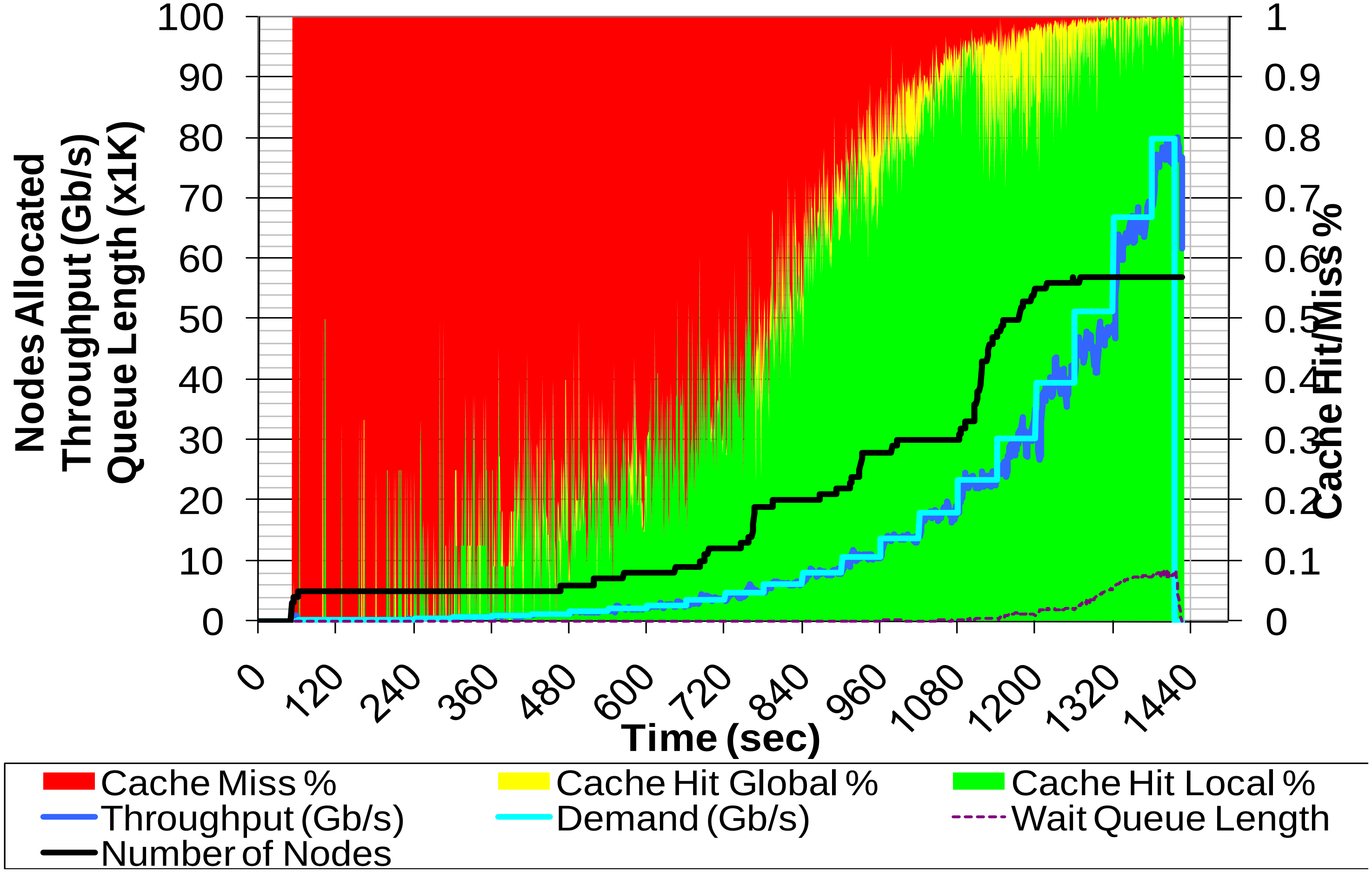
← 2GB

4GB →

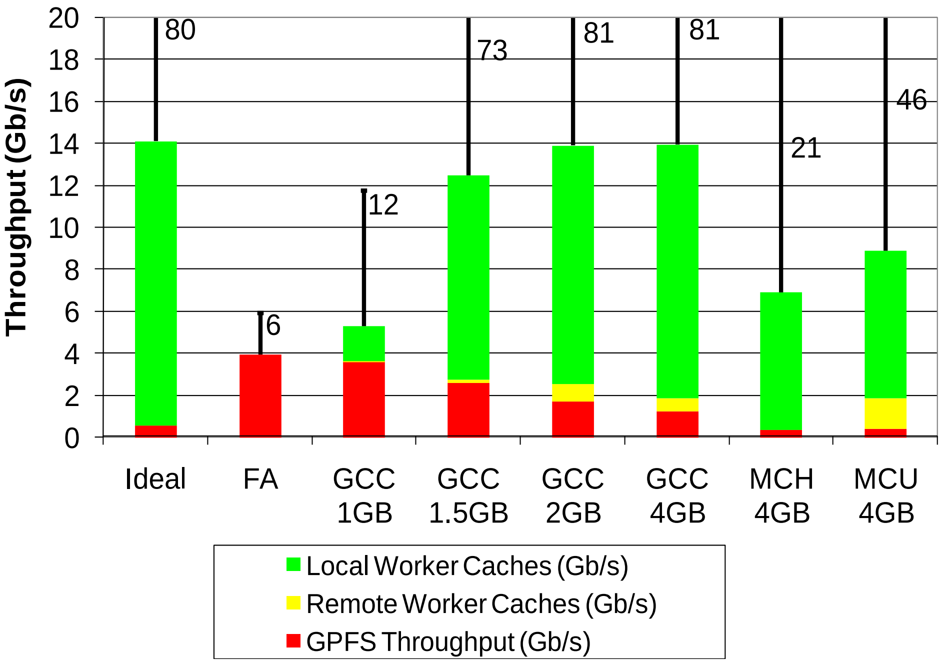


Monotonically Increasing Workload Good-cache-compute

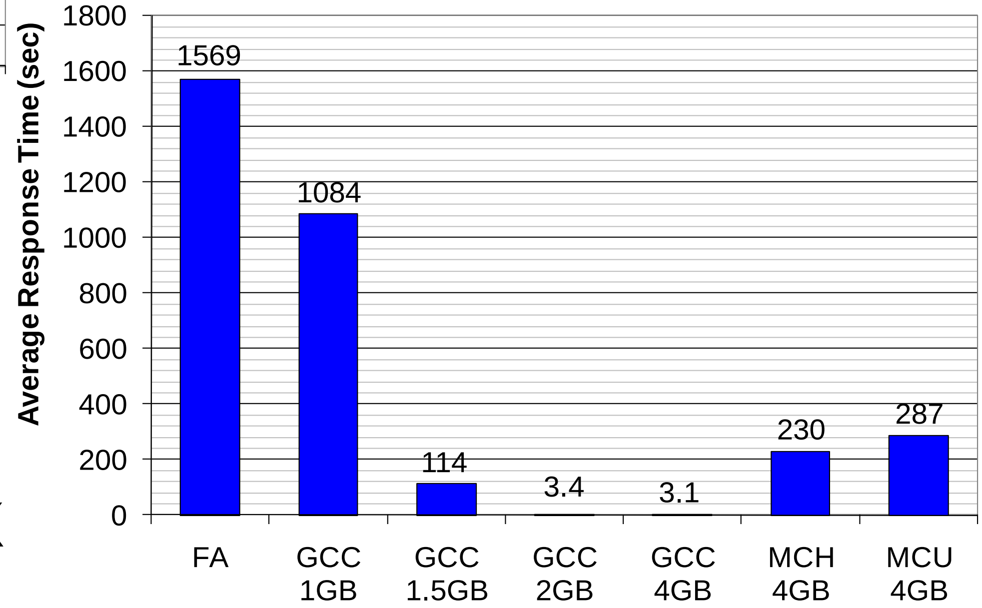
- **Data Diffusion vs. ideal**: 1436 sec vs 1415 sec



Monotonically Increasing Workload Throughput and Response Time



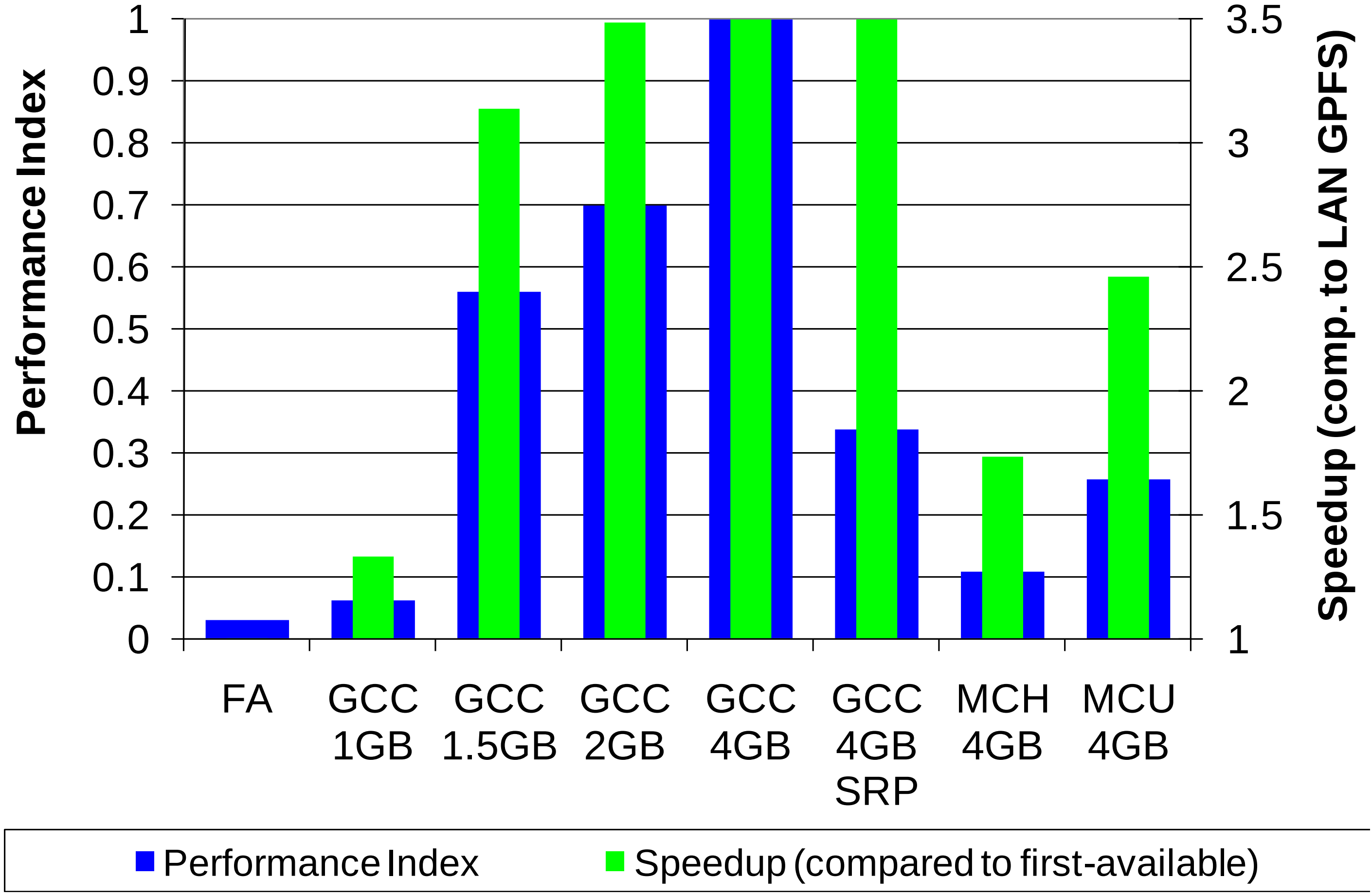
- ➔ Throughput:
  - Average: 14Gb/s vs 4Gb/s
  - Peak: 81Gb/s vs. 6Gb/s



- Response Time ➔
  - 3 sec vs 1569 sec ➔ 506X

Monotonically Increasing Workload Performance Index and Speedup

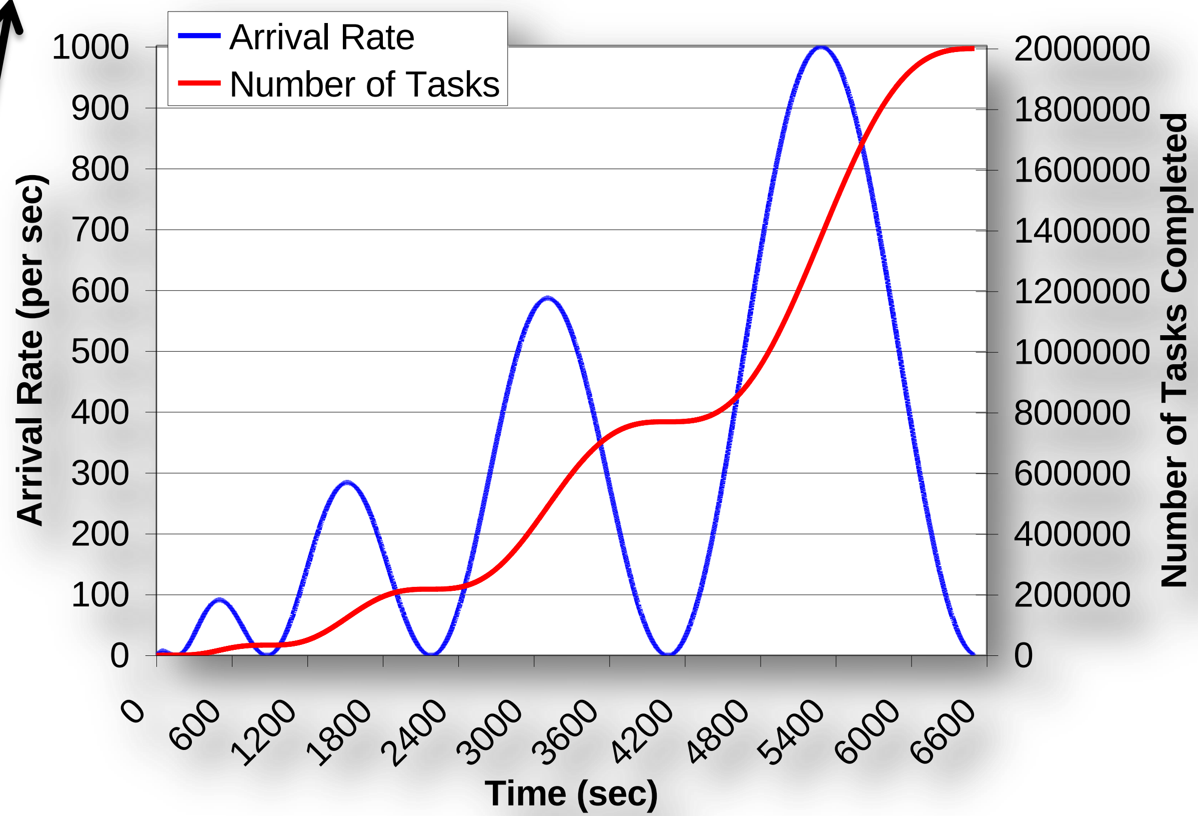
- Performance Index:
  - 34X higher
- Speedup
  - 3.5X faster than GPFS



Sine-Wave Workload

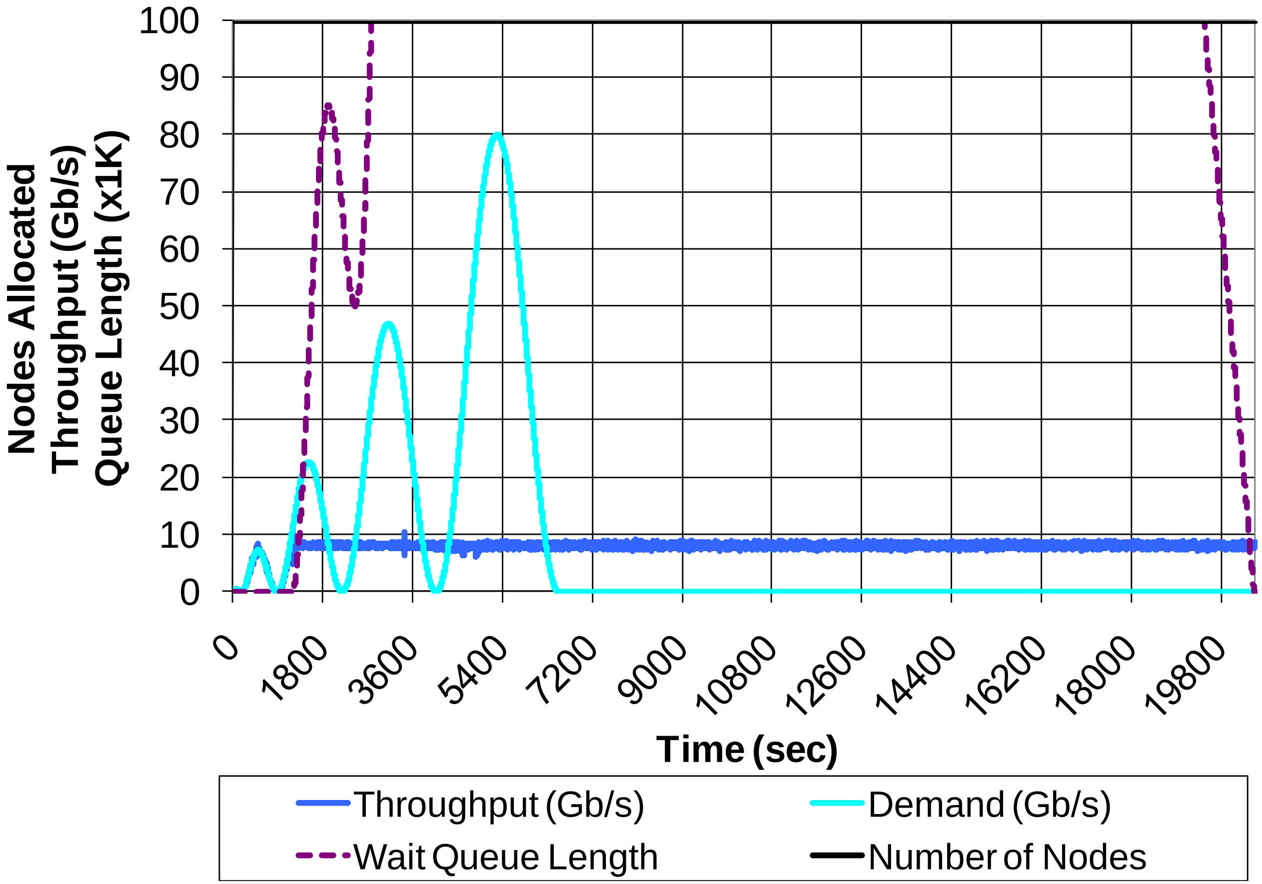
- 2M tasks
  - 10MB reads
  - 10ms compute
- Vary arrival rate:
  - Min: 1 task/sec
  - Arrival rate function:
  - Max: 1000 tasks/sec
- 200 processors
- Ideal case:
  - 6505 sec
  - 80Gb/s peak throughput

$$A = \left\lfloor (\sin(sqrt(time+0.11)*2.859678)+1)*(time+0.11)*5.705 \right\rfloor$$



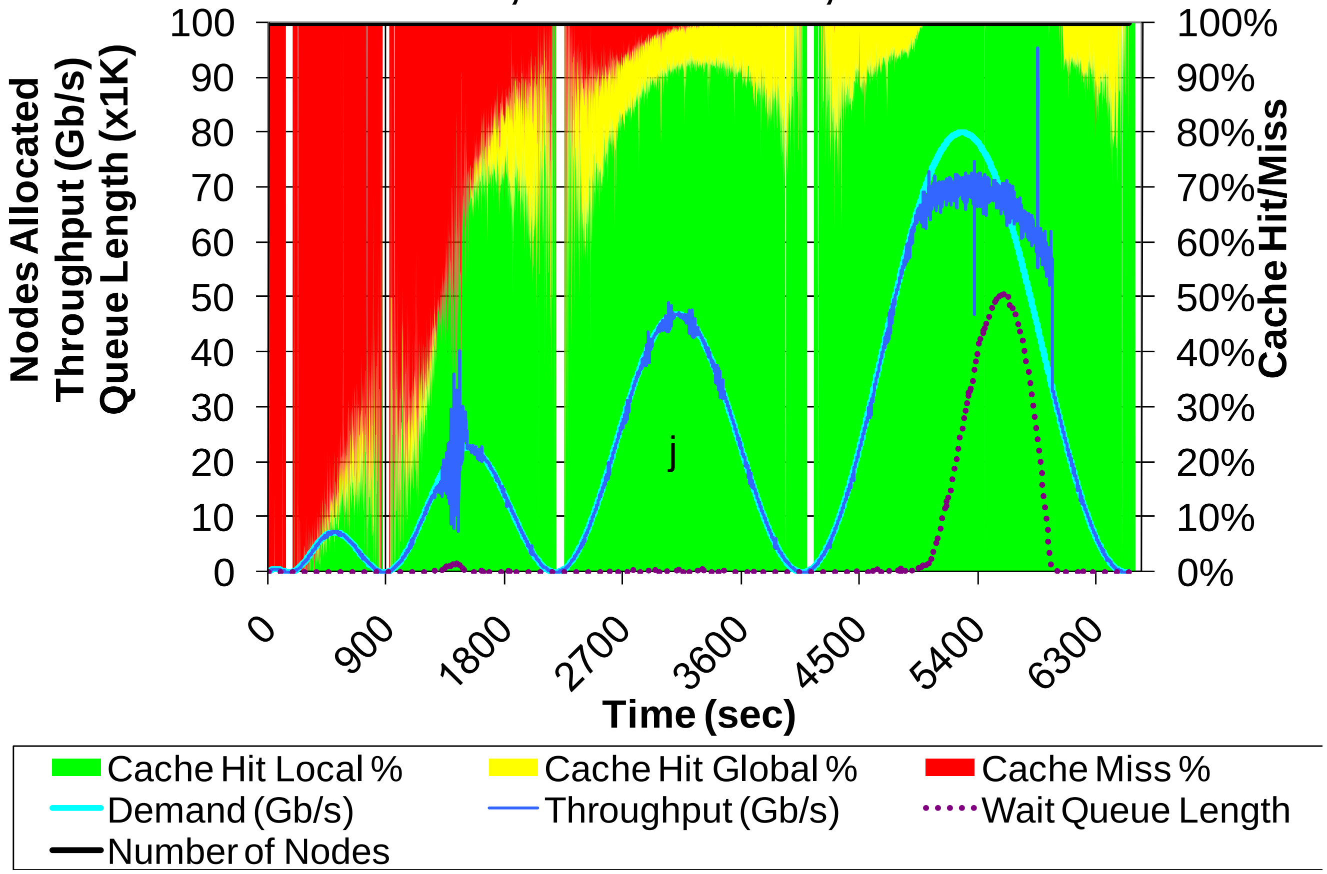
Sine-Wave Workload First-available (GPFS)

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



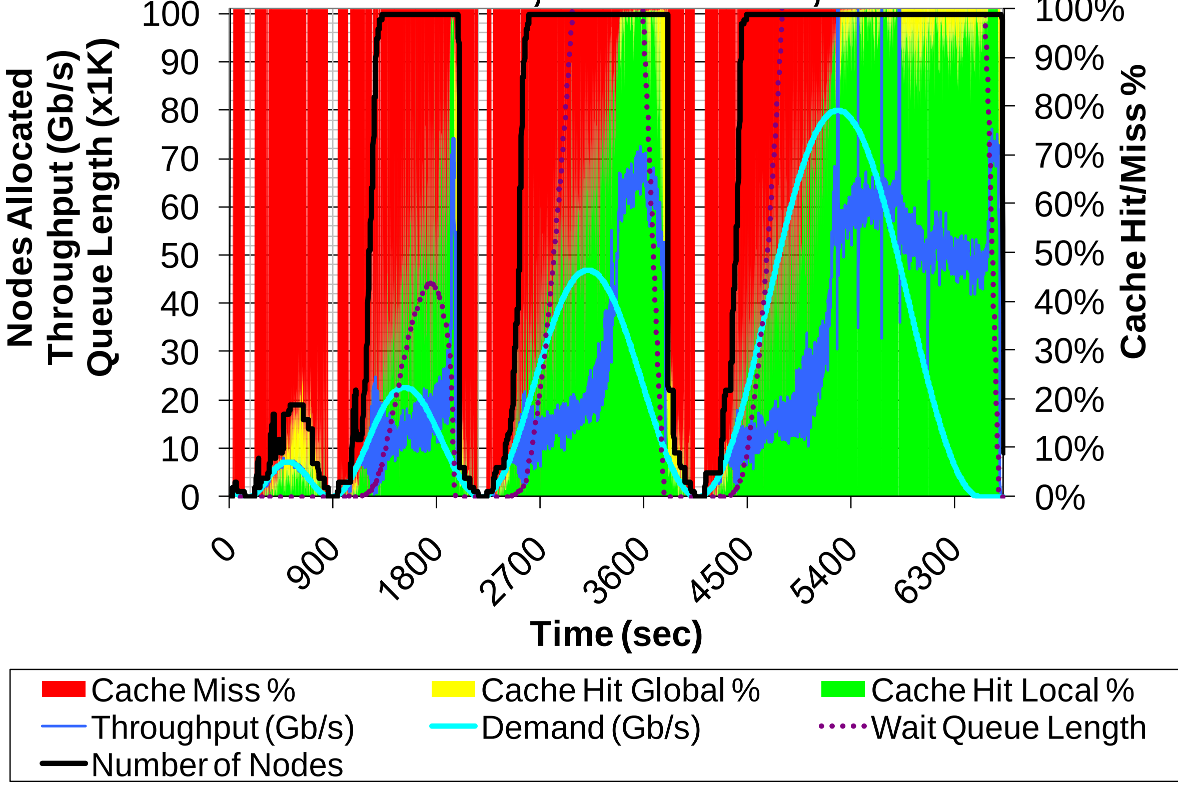
Sine-Wave Workload Good-cache-compute and SRP

- 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

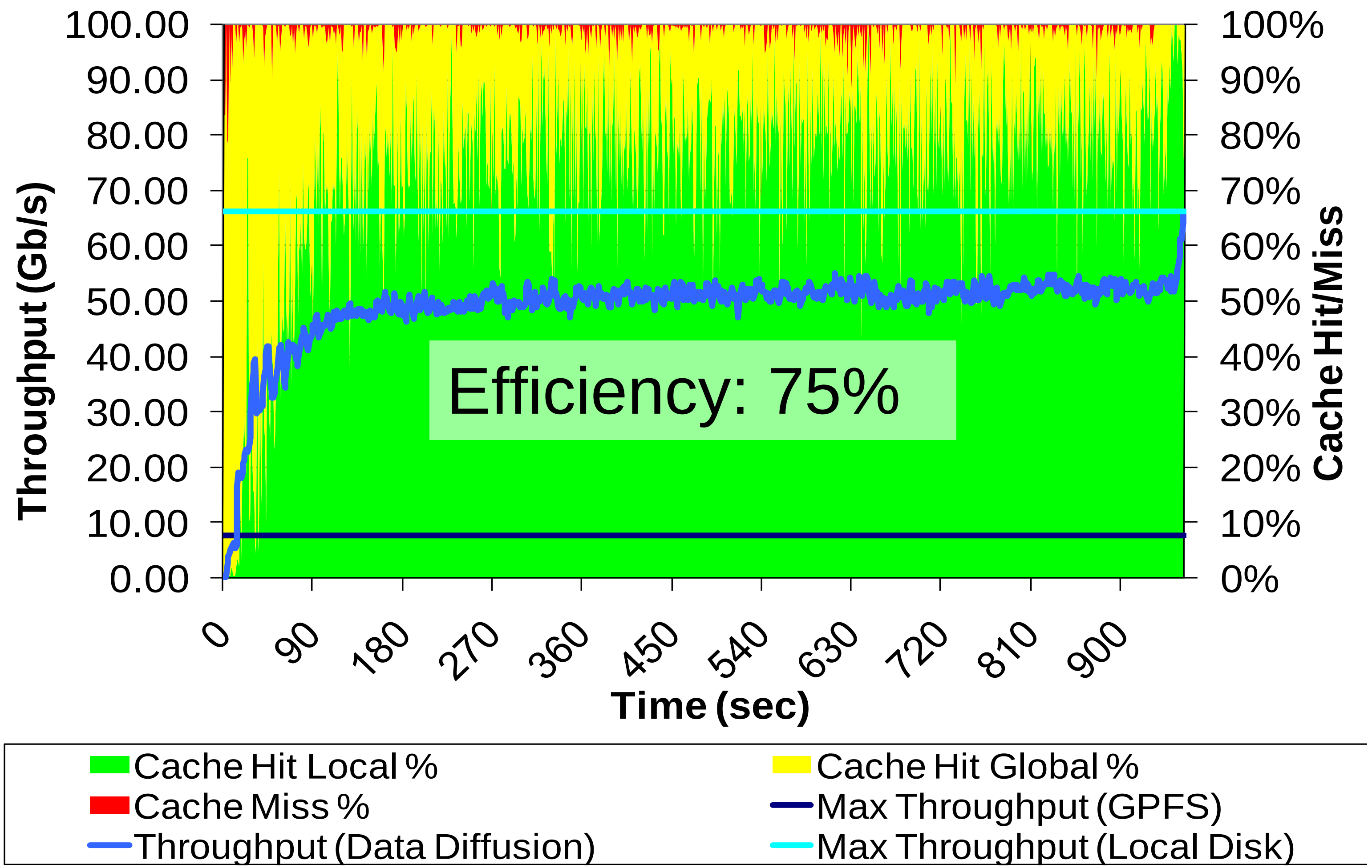
- All-Pairs(set A, set B, function F) returns matrix M:
- Compare all elements of set A to all elements of set B via function F, yielding matrix M, such that

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

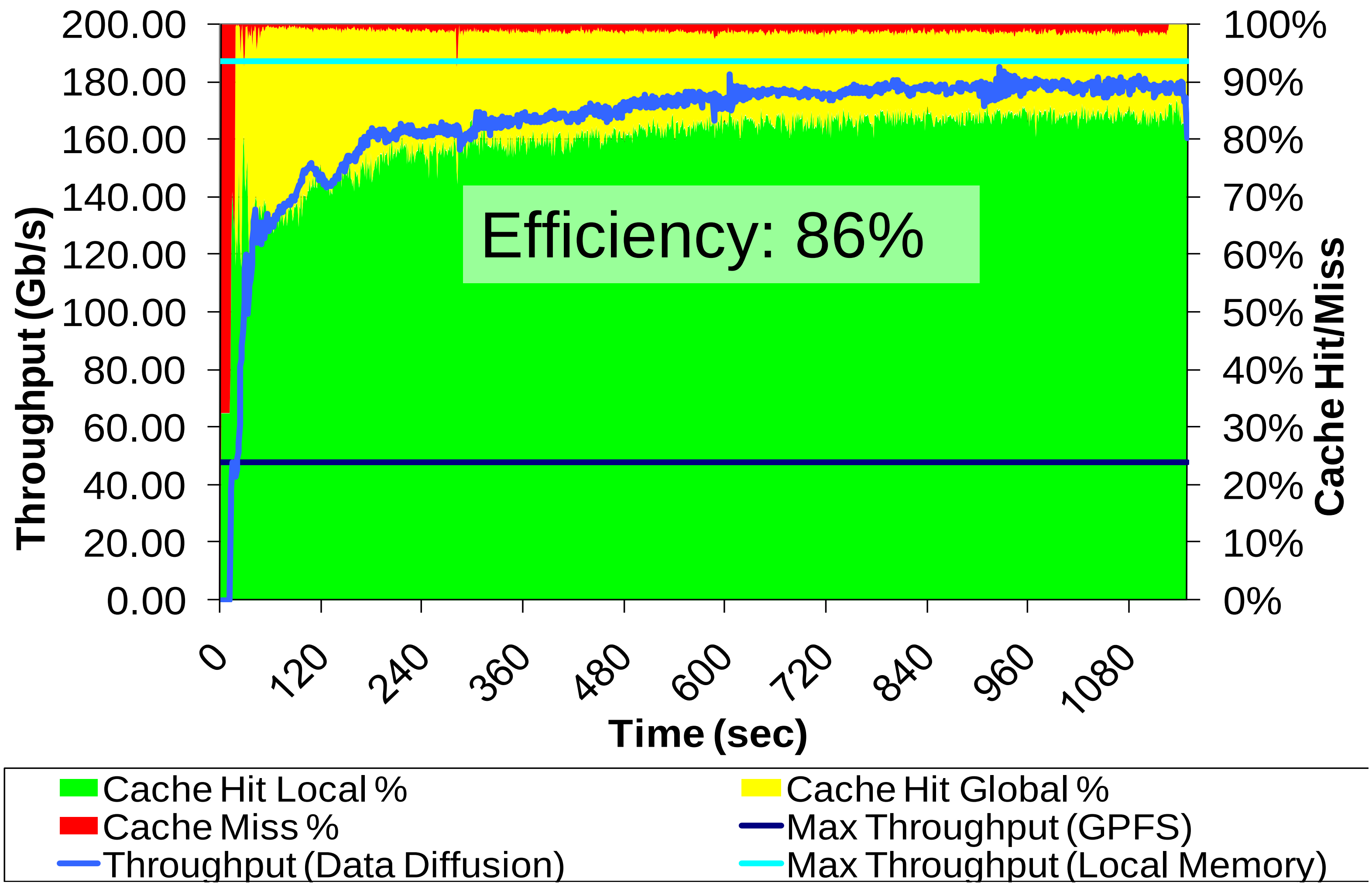
```
foreach $i in A
    foreach $j in B
        submit_job F $i $j
    end
end
```


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

All-Pairs Workload 500x500 on 200 CPUs



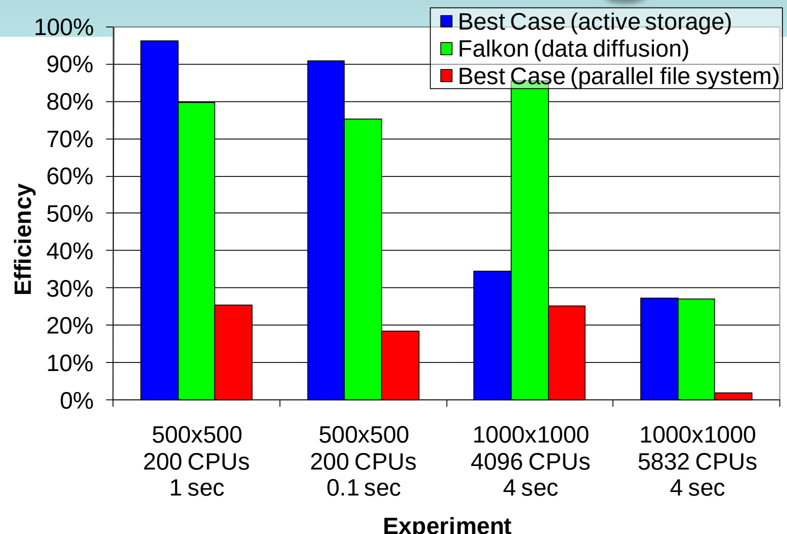
All-Pairs Workload 1000x1000 on 4K emulated CPUs



[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



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 4 sec	**Best Case (active storage)**	24000	12288	24
	Falkon (data diffusion)	24000	3867	906

Christopher Moretti, Douglas Thain, University of Notre Dame

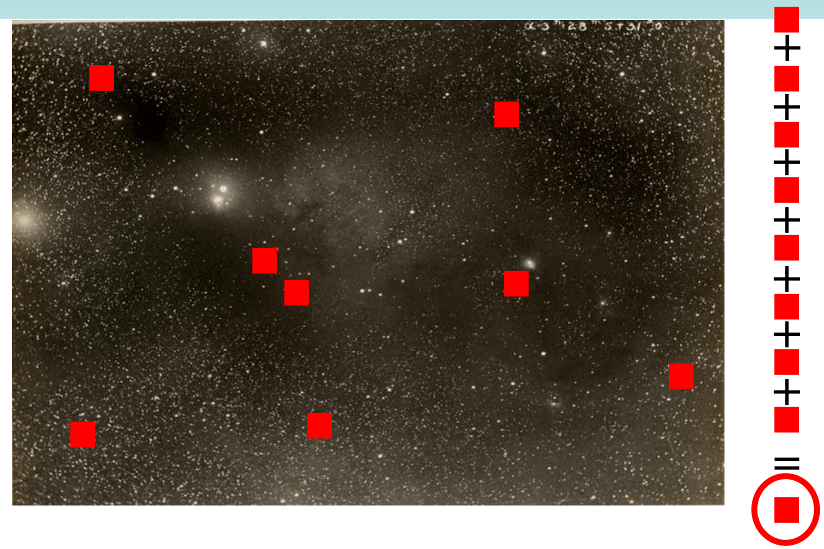
[DIDC09] "Towards Data Intensive Many-Task Computing", under review

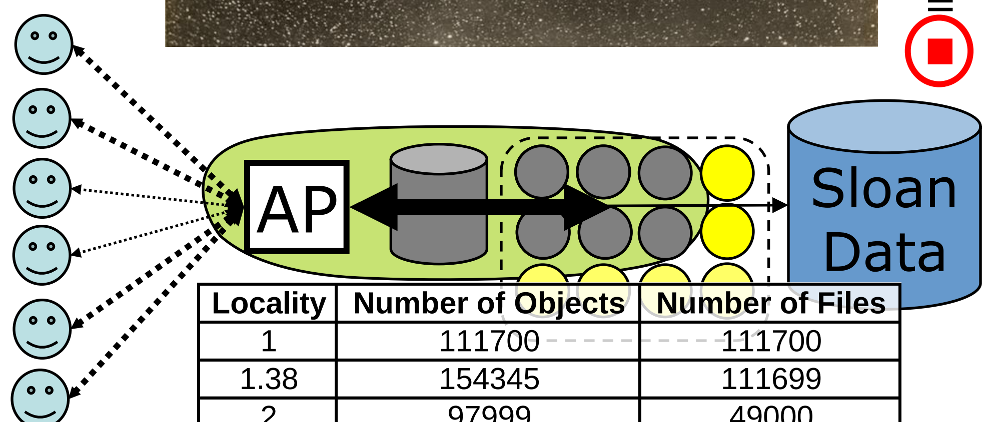
All-Pairs Workload
Data Diffusion vs. Active Storage

- 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

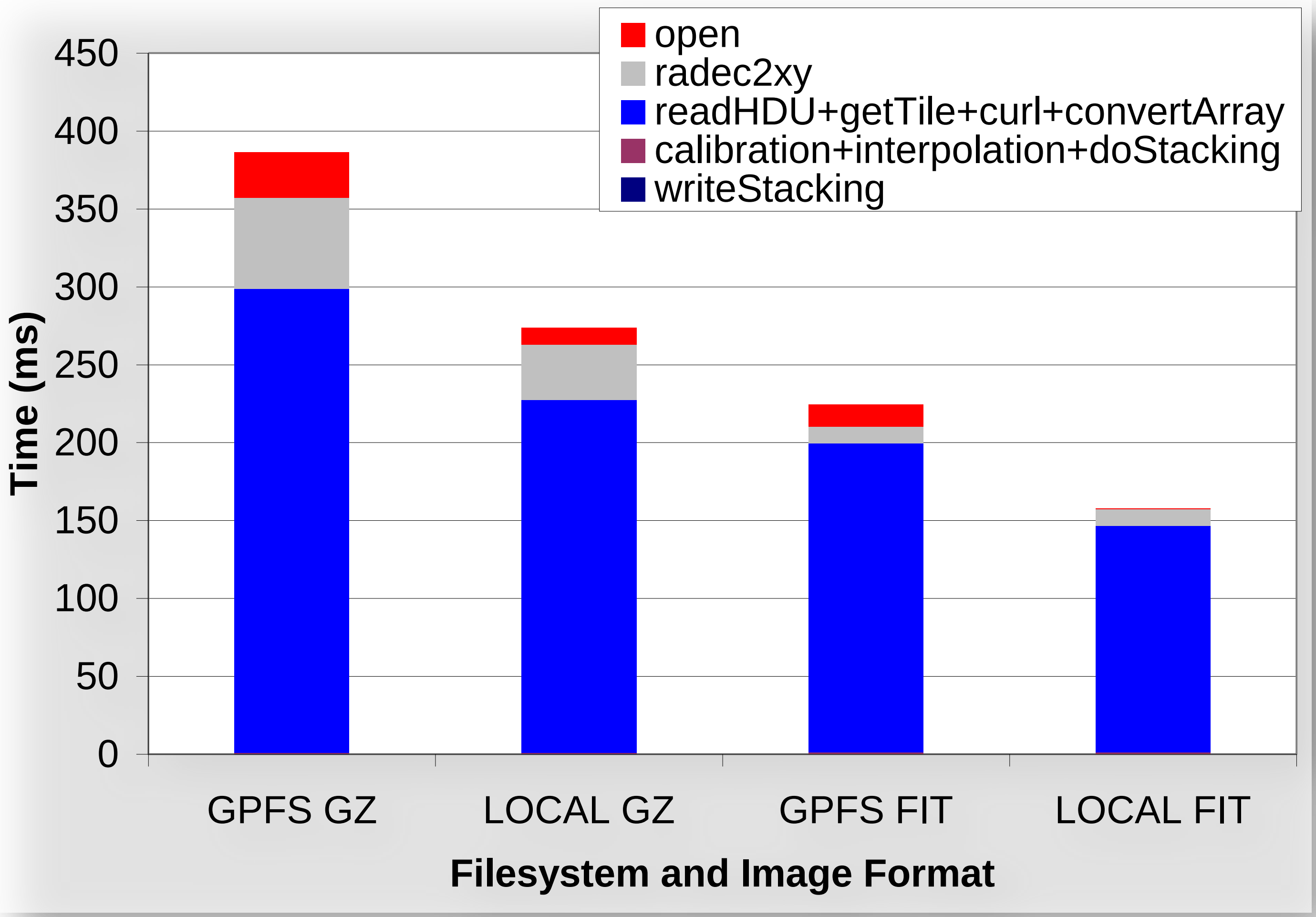




Locality	Number of Objects	Number of Files
1	111700	111700
1.38	154345	111699
2	97999	49000
3	88857	29620
4	76575	19145
5	60590	12120
10	46480	4650
20	40460	2025
30	23695	790

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

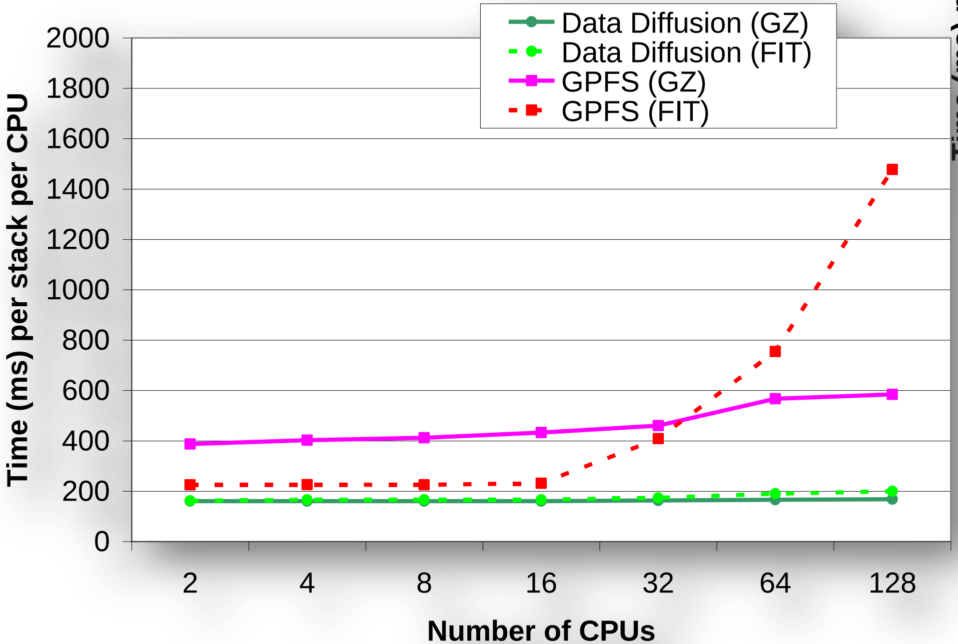
Image Stacking Workload Profiling

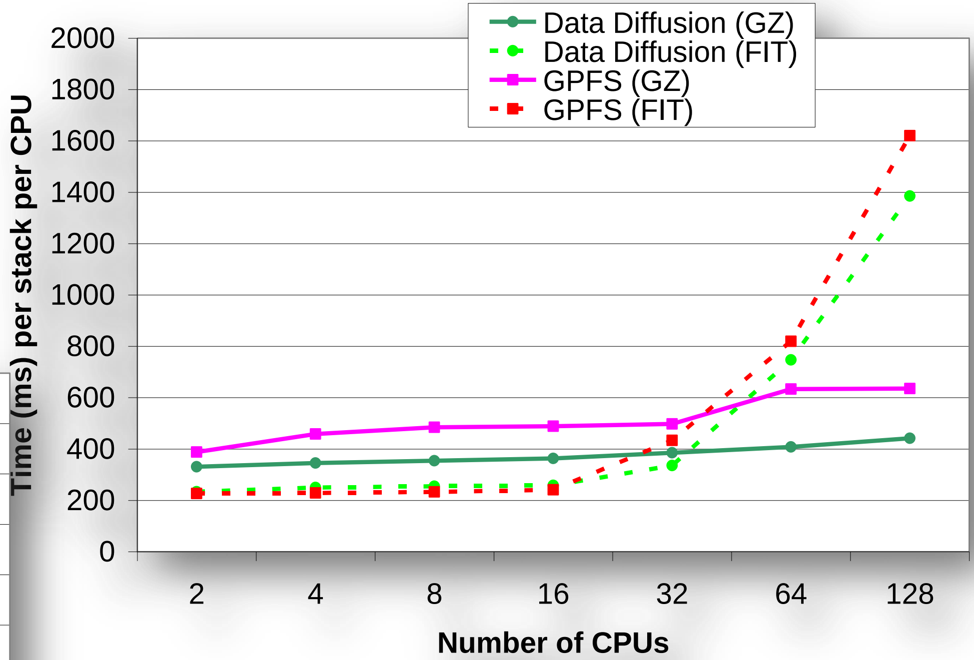


[DADC08] "Accelerating Large-scale Data Exploration through Data Diffusion"

Image Stacking Workload Varying Scale

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



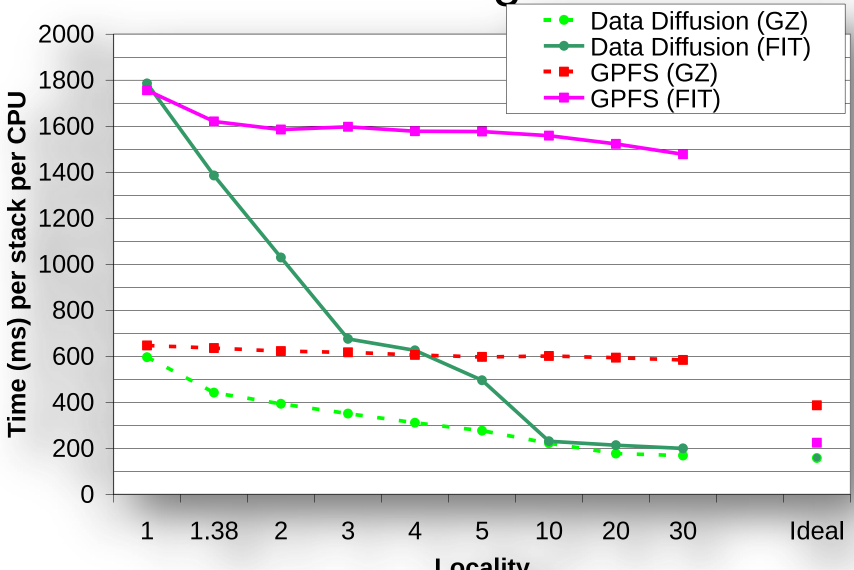


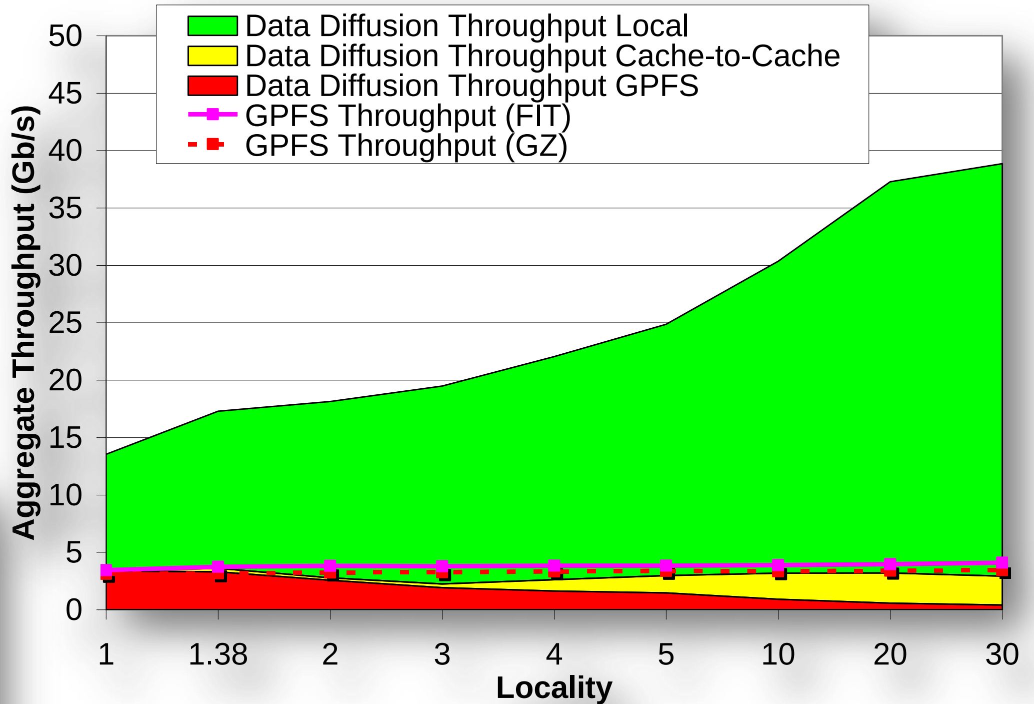
🡐High data locality
- Near perfect scalability

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

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

More Information

- More information:
  - Falkon: http://dev.globus.org/wiki/Incubator/Falkon
  - Swift: http://www.ci.uchicago.edu/swift/index.php