

# Cloud Scheduling

Improving MapReduce Performance in Heterogeneous Environments, M. Zaharia et al, OSDI 2008

Quincy: Fair Scheduling for Distributed Computing Clusters, M. Isard et al, SOSP 2009

CA-NFS: A Congestion-Aware Network File System, A. Batsakis et al, FAST 2009

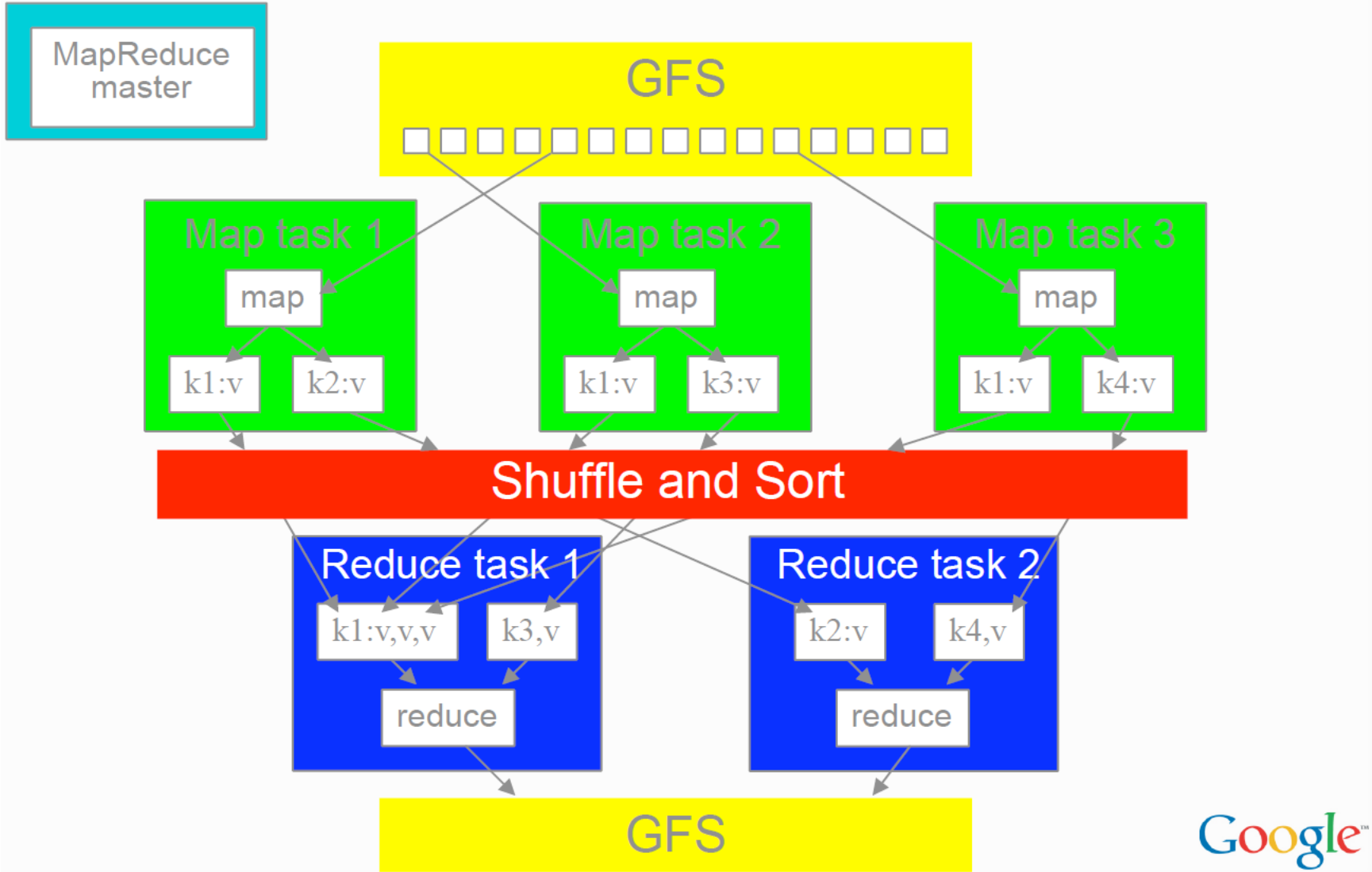
**Ghazale Hosseinabadi**  
**Wucherl Yoo**

# Improving MapReduce Performance in Heterogeneous Environments

Matei Zaharia, Andy Konwinski, Anthony D. Joseph,  
Randy Katz, Ion Stoica

OSDI 2008

# Motivation - MapReduce



# Motivation – Hadoop Scheduling

- **Straggler Task**

- Poorly perform due to faulty hardware or mis-configuration

- **Speculative Execution**

- Minimize job's response time
- Speculation copy (backup task) for straggler

- **Assumptions**

- **Homogeneous** cluster nodes
- Constant progress rate of task
- Copy, sort, and reduce phases take the same amount (1/3) work of reduce task
- Tasks finish in waves - low progress score represents a straggler

# Problem - Broken Assumptions

- **Heterogeneous Cluster Nodes**
  - Multiple VMs on a same physical host
  - Multiple HW generations
- **Non-Linear Progress of Tasks**
  - Copy phase of reduce task is slowest due to network communication
  - Tasks from different generations run concurrently
- **Problems**
  - Too many speculative tasks can run (80% of reducers)
  - Wrong (fast and new) task can be selected as straggler
  - Speculative task can be assigned to slow node

# LATE Scheduler

- **Longest Approximate Time to End**
  - Speculate (backup) the task with the largest estimated time left

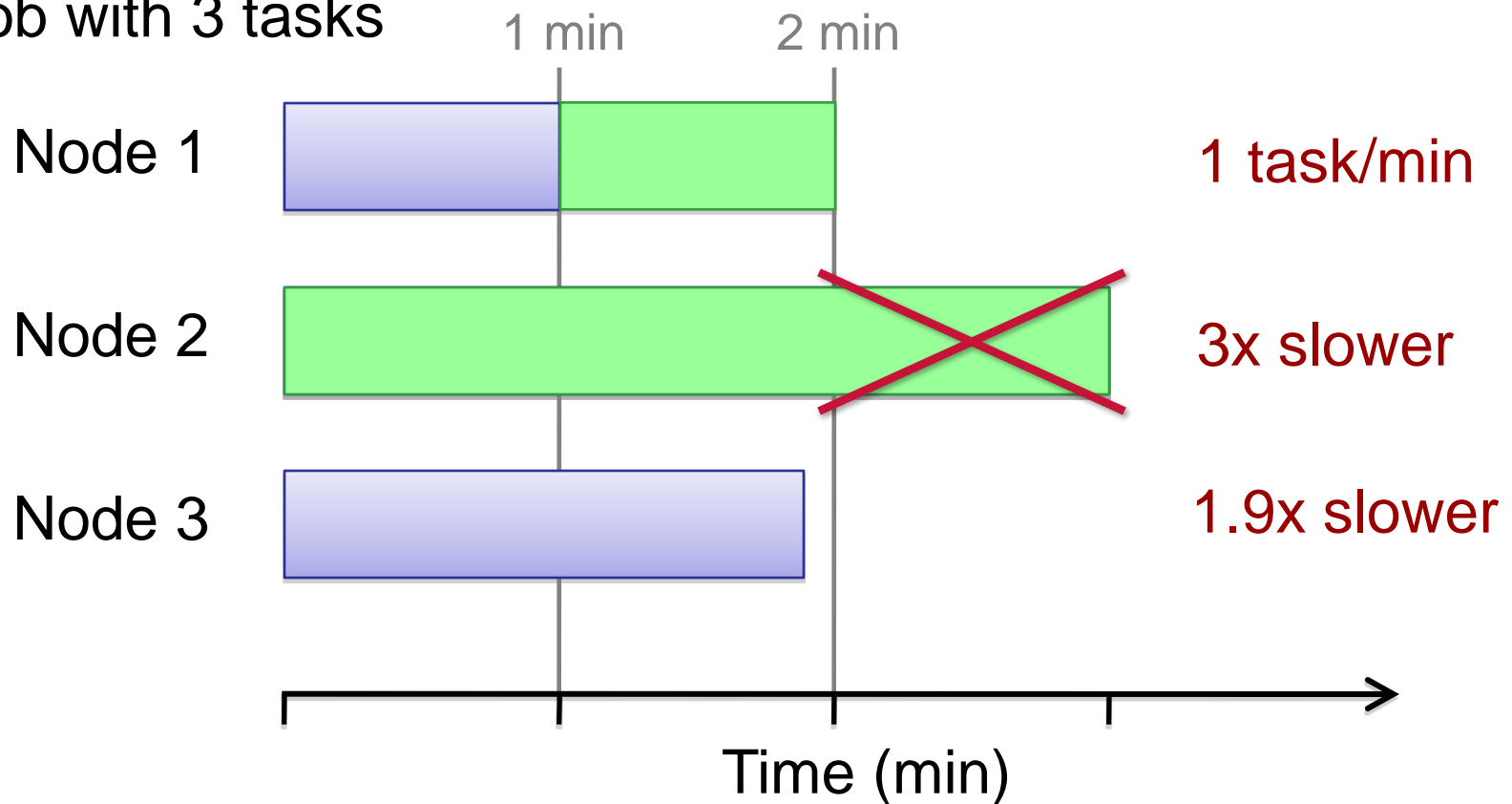
$$\textit{Progress Rate} = \frac{\textit{Progress Score}}{\textit{Execution Time}}$$

$$\textit{Estimated Time Left} = \frac{1 - \textit{Progress Score}}{\textit{Progress Rate}}$$

# Progress Rate Example

Speculate a task far below the average progress rate

A job with 3 tasks



# Progress Rate Example

A job with 5 tasks

2 min

Node 1



Node 2



Time left: 1 min, Progress Rate = 0.33

Node 3



Time left: 1.8 min, Progress Rate = 0.53

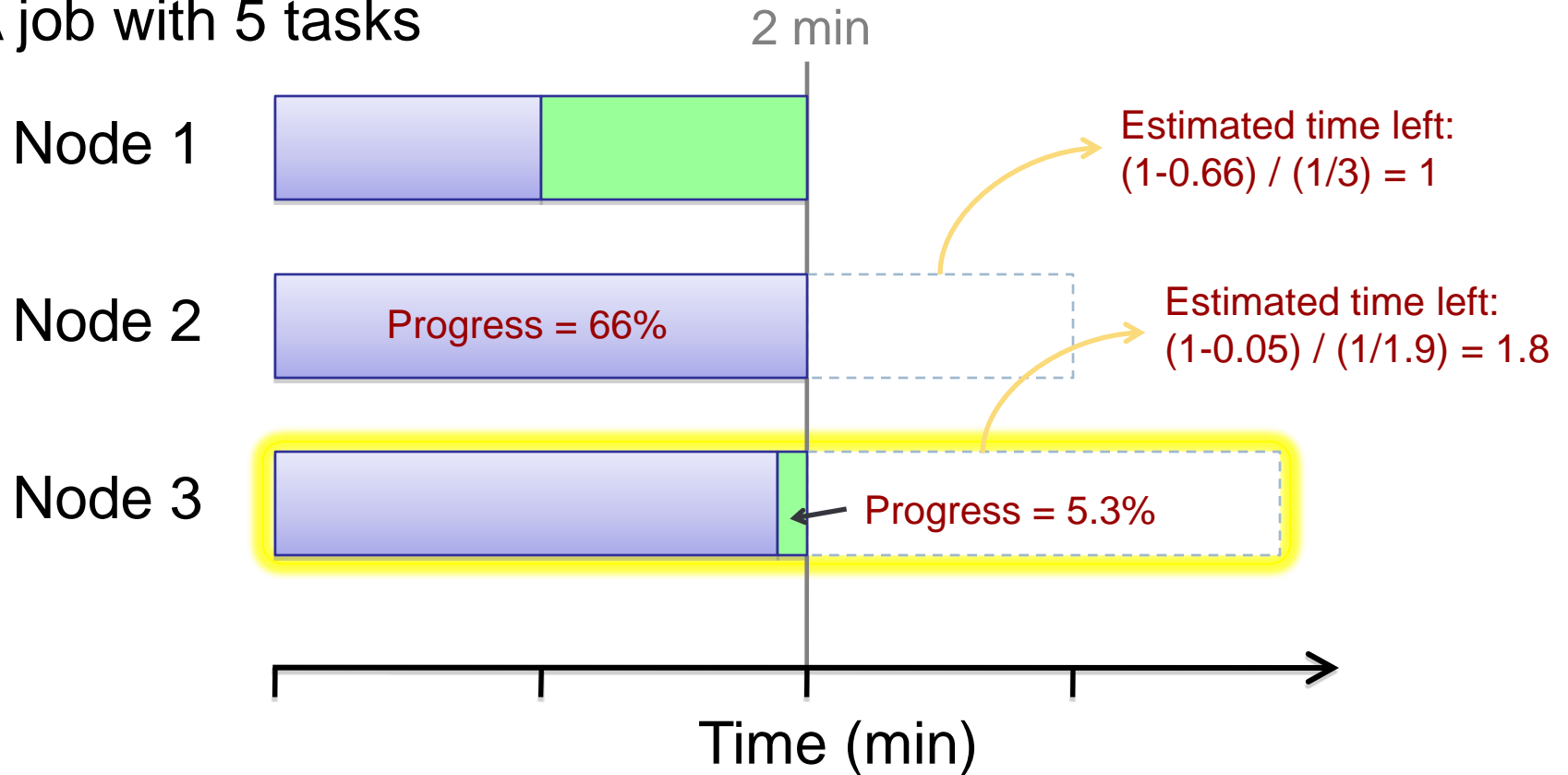
Time (min)

**Node 2 is slowest but picked  
It should have speculated Node 3's**



# LATE Example

A job with 5 tasks



**LATE picks Node 3**

# Rationales in LATE Scheduler

- **The Past Progress of a Task Represents the Future Progress**
  - Assuming constant progress rate – may be incorrect
  - Heterogeneity impacts appear in the past progress
- **Looking Forward**
  - Try to Speculate Tasks that Improve Response Time the Most

# Details on LATE Scheduler

- **Prioritize Tasks to Speculate**
  - Based on how much the tasks hurt the response time
  - *SlowTaskThreshold* – 25<sup>th</sup> percentile
- **Select Fast Node to Run on**
  - Based on total work performed
  - *SlowNodeThreshold* – 25<sup>th</sup> percentile
- **Cap to the Number of Speculative Tasks**
  - *SpeculativeCap* - 20%
  - Avoid unnecessary speculations
  - Limit contention and hurting throughput
- **Chosen Thresholds are Not Sensitive**

# Experimental Environments

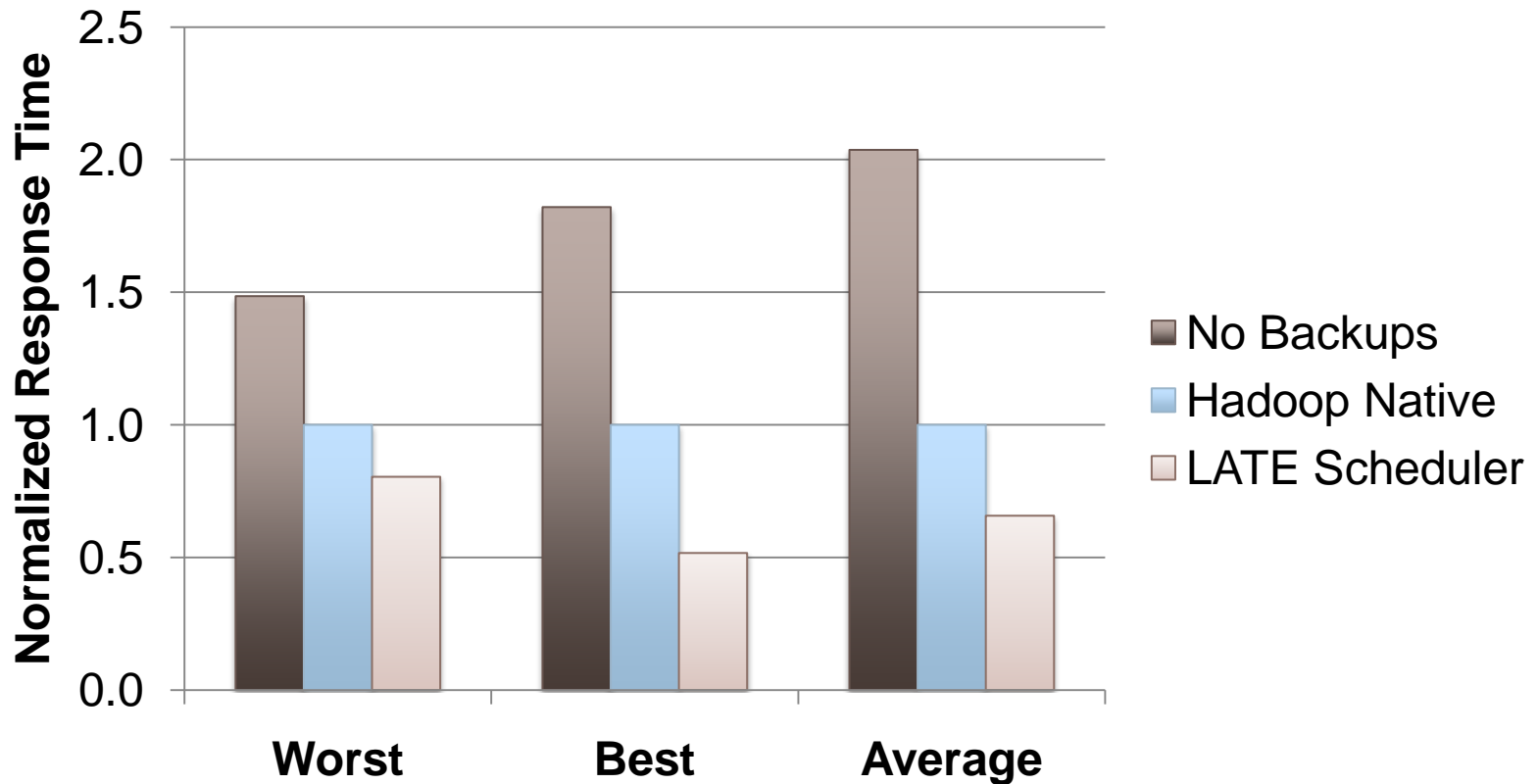
- **Environments**

- Amazon EC2 (200-250 nodes)
- 2 replicas of each chunk on Hadoop Distributed File System
- Up to 2 mappers and 2 reducers (Hadoop default)

- **Heterogeneity Setup**

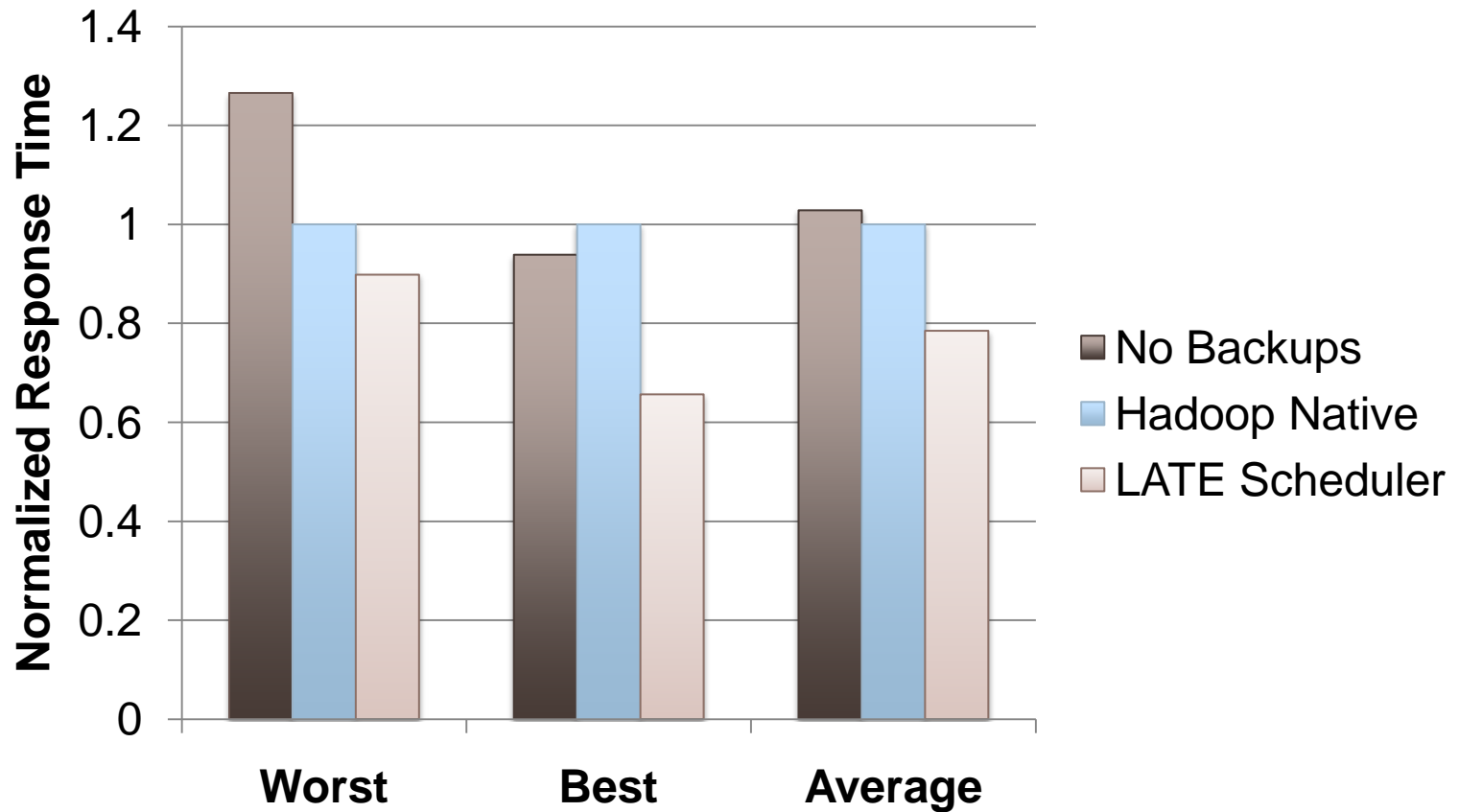
- Assigning a varying number of VMs to each node
  - Contention on resources
- Background job intentionally makes stragglers

# EC2 Sort with Stragglers



- **Average 58% speedup over native, 220% over no backups**

# EC2 Sort without Stragglers



- **Average 27% speedup over native, 31% over no backups**

# Remarks

- **Contributions**

- Analysis about heterogeneity that makes speculation of Hadoop worse than native setting
- LATE speculatively executes the tasks that hurts the response time the most on fast nodes
  - Considering heterogeneity

- **Limitations**

- Lack of considerations about data locality and fairness
- Tasks may require different amount of computation
- Speculation reduces the throughput of cloud

# Discussion Points

- **Workload is chosen in favor of LATE**
  - LATE doesn't always improve the performance
- **Would the preemption of tasks help?**
  - How many tasks are assigned to a node?



# **Quincy: Fair Scheduling for Distributed Computing Clusters**

**Michael Isard  
Vijayan Prabhakaran  
Jon Currey  
Udi Wieder  
Kunal Talwar  
Andrew Goldberg**

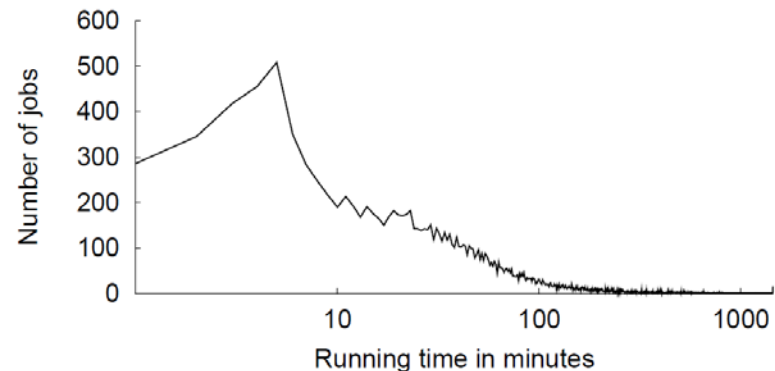
**Microsoft Research**

# Outline

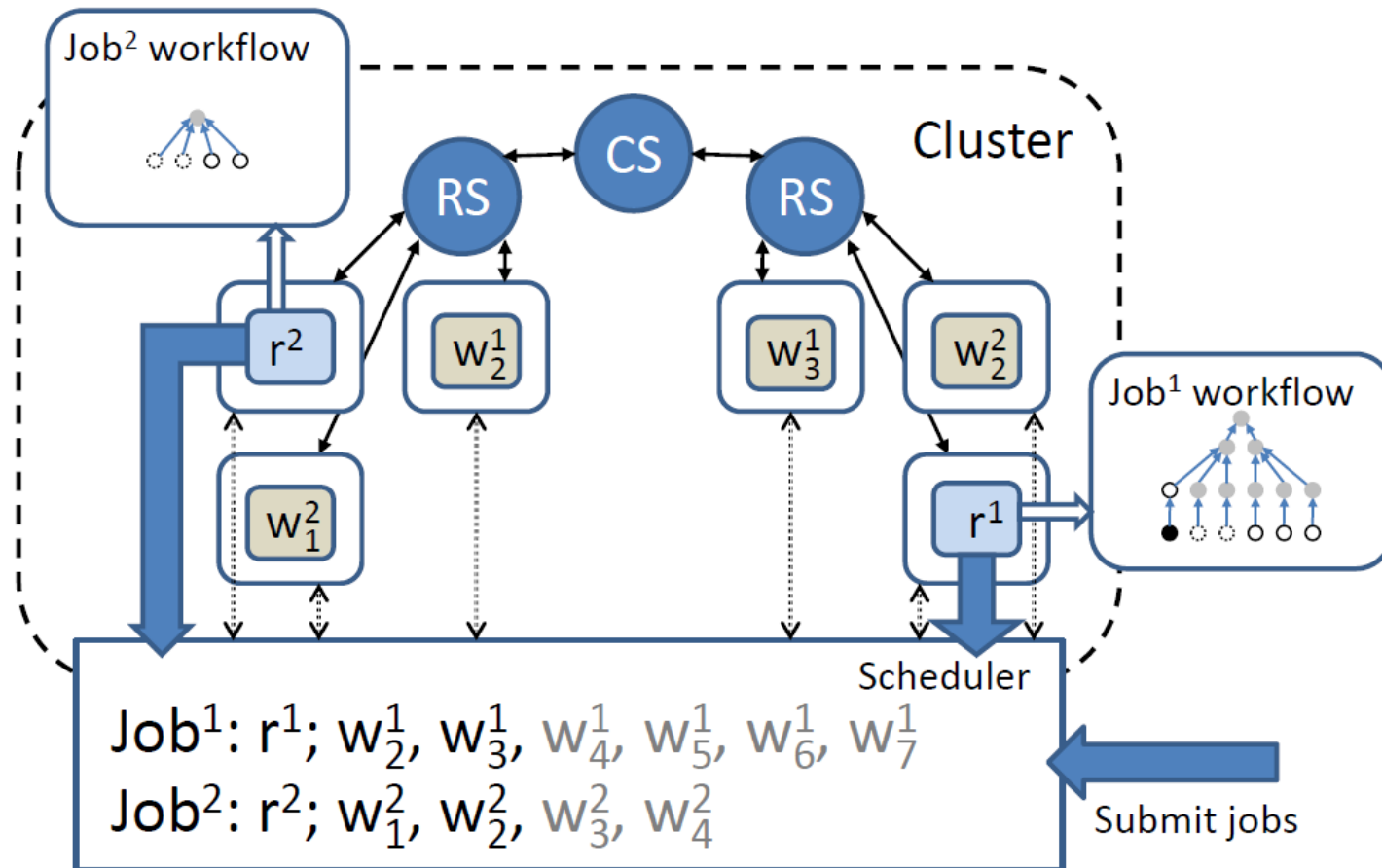
- **Cluster architecture**
- **Queue-based scheduling**
- **Min-cost flow**
- **Flow-based scheduling**
  - Fairness and locality constraints
- **Evaluation**

# Motivation

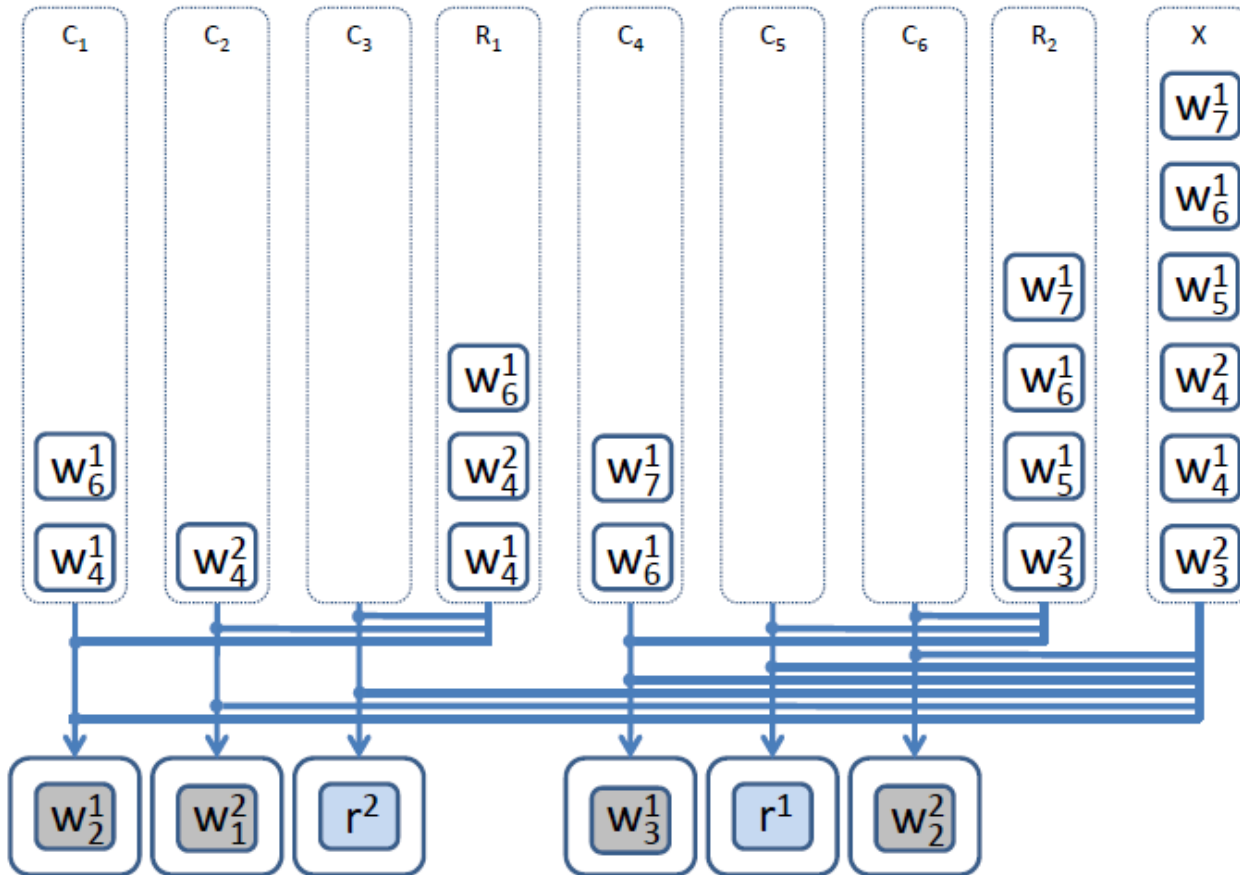
- **Scheduling concurrent jobs on clusters**
- **Sharing of the cluster among short jobs**
  - Microsoft cluster  $\approx$  250 computers
  - Academic cluster
- **Fair sharing**
- **Application data is stored on computers**
  - High bandwidth between computers: expensive
  - Computations are placed close to the input data
- **Fairness and locality conflict**



# Cluster Architecture



# Queue-based scheduler



# Min-cost flow

- **Flow network:**
  - Directed graph
  - $y_e$ : capacity of  $e$
  - $p_e$ : cost of  $e$
  - $\epsilon_v$ : integer “supply” of  $v$
  - $\sum_v \epsilon_v = 0$ .
- **Feasible flow:** assigns a non-negative integer flow  $f_e \leq y_e$ , such that for every node  $v$ ,

$$\epsilon_v + \sum_{e \in \mathcal{I}_v} f_e = \sum_{e \in \mathcal{O}_v} f_e$$

- **A min-cost feasible flow is a feasible flow that minimizes  $\sum_e f_e p_e$**

# Data Locality

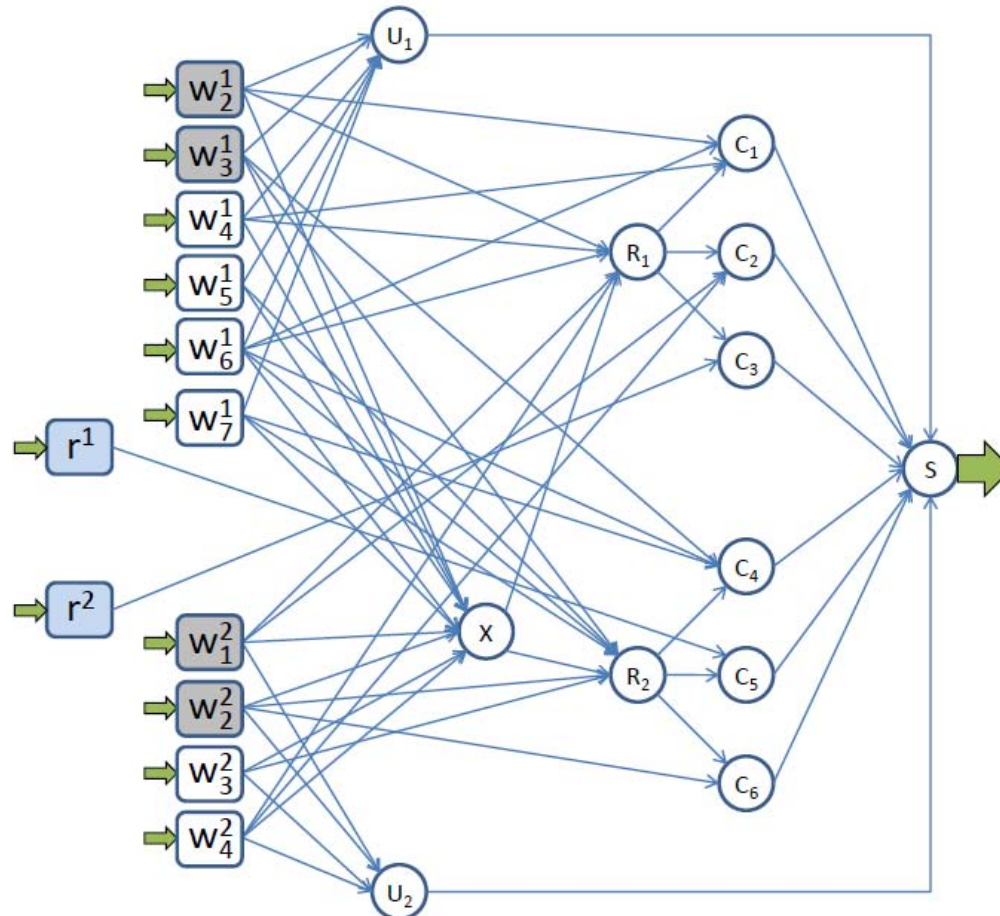
- **Application data is stored on the computing nodes.**
- **Scheduling computations close to their data is crucial for performance.**
- **Hadoop:**
  - Computer storing one of the replica
  - On the same rack
  - Random computer

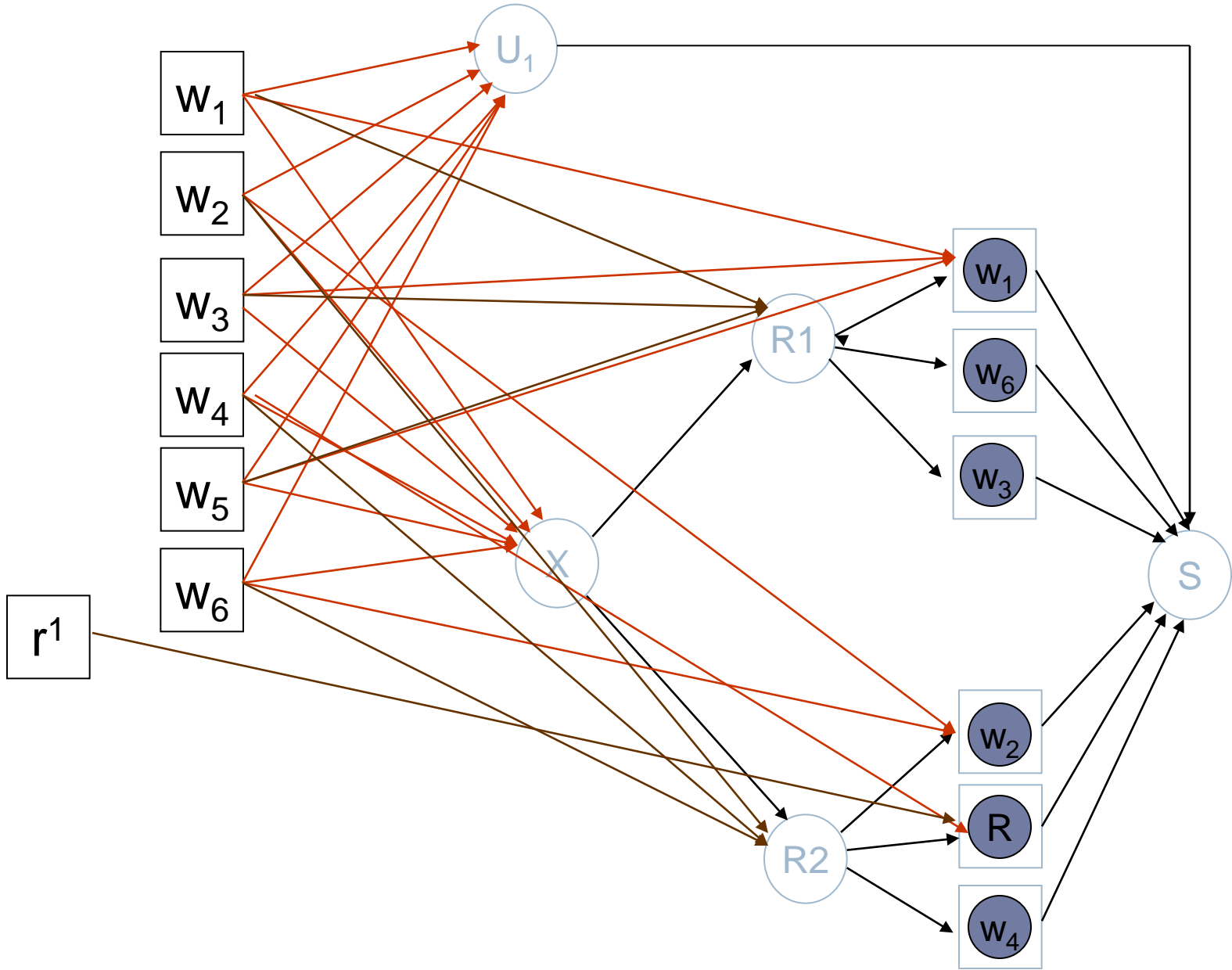
# Fairness in a Shared Cluster

- **N computers, J jobs**
- **Each job gets at least  $N/J$  computers**
- **Fine-grain sharing:**
  - Multiplex all computers in cluster between jobs.
  - When a task completes computer may be assigned to another job.
  - Job uses  $N/J$  computers at a time but set in use varies over lifetime.
- **Why not static allocation?**
  - Varying workload
  - Not able to adjust to workload changes
  - Low system throughput and resource utilization under non-uniform workloads



# Encoding scheduling as a flow network





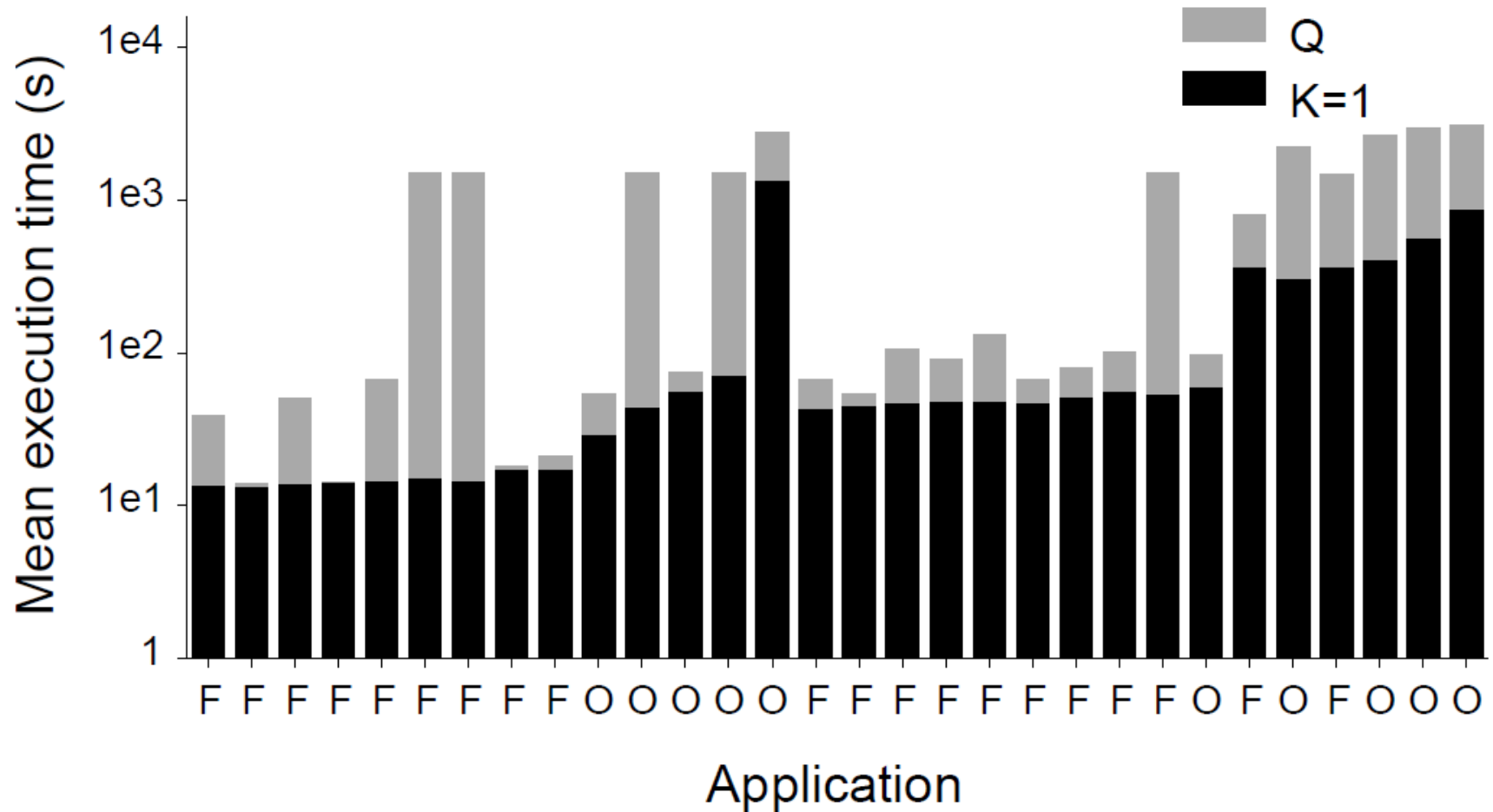
# Fairness Policies

- **Q: Quincy, Unfair sharing without Preemption**
- **QF: Quincy with Fairness, without Preemption**
- **QP: Quincy with Preemption, Unfair sharing**
- **QFP: Quincy with Fairness and Preemption**

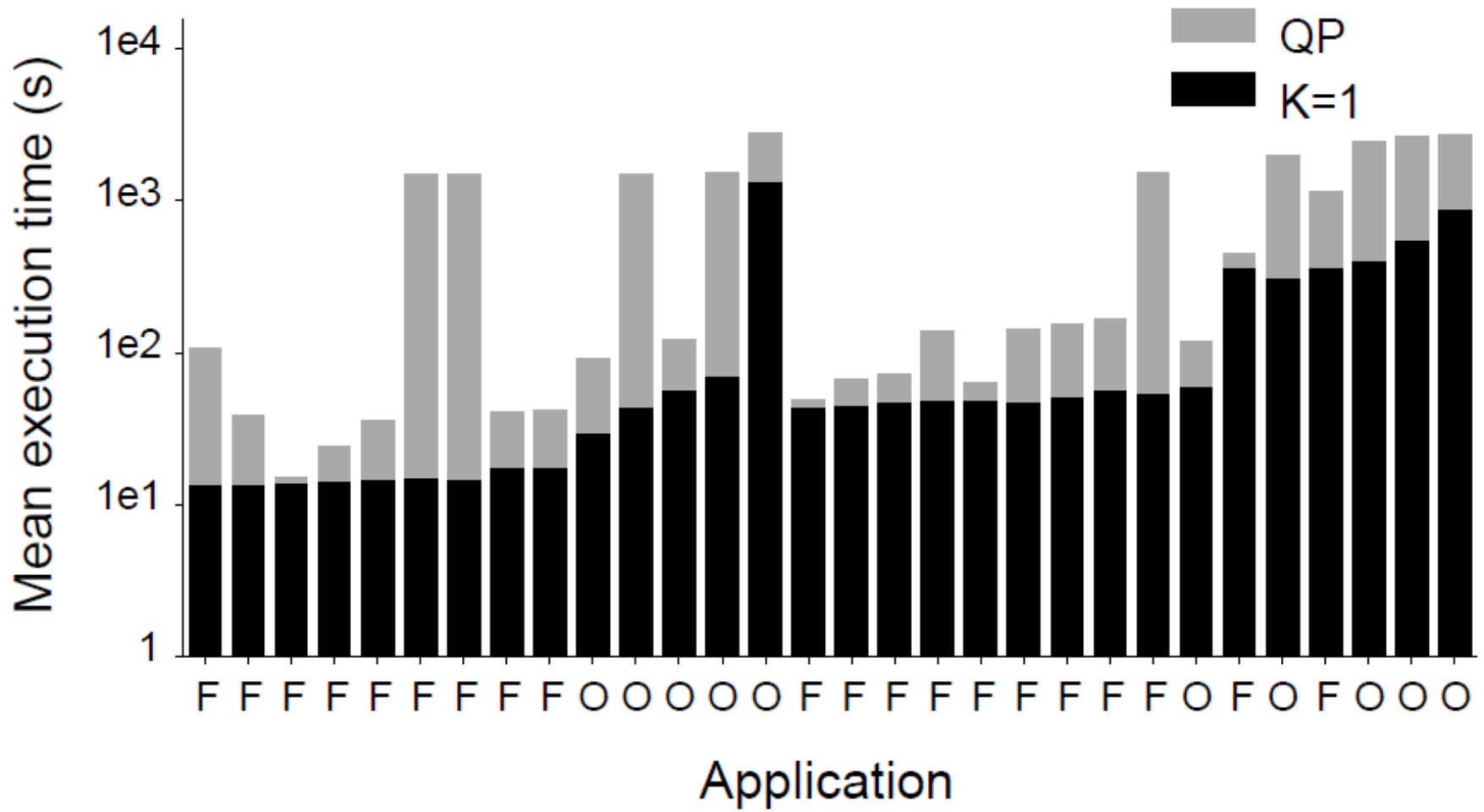
# Evaluation

- **Cluster of 240 computers**
- **Cluster runs the Dryad distributed execution engine**
- **Applications:**
  - Sort
  - DatabaseJoin
  - PageRank
  - WordCount
  - Prime

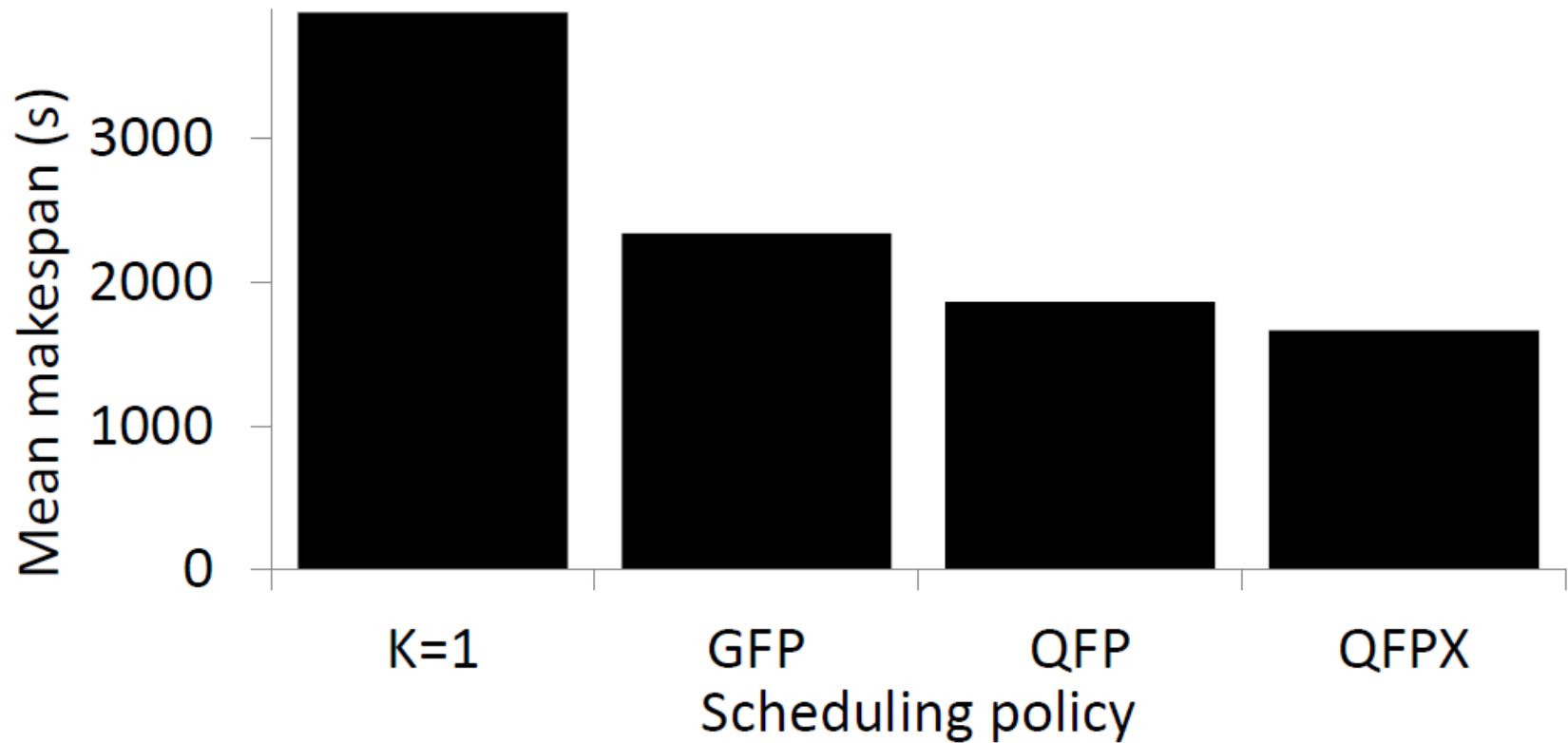
# Running Times



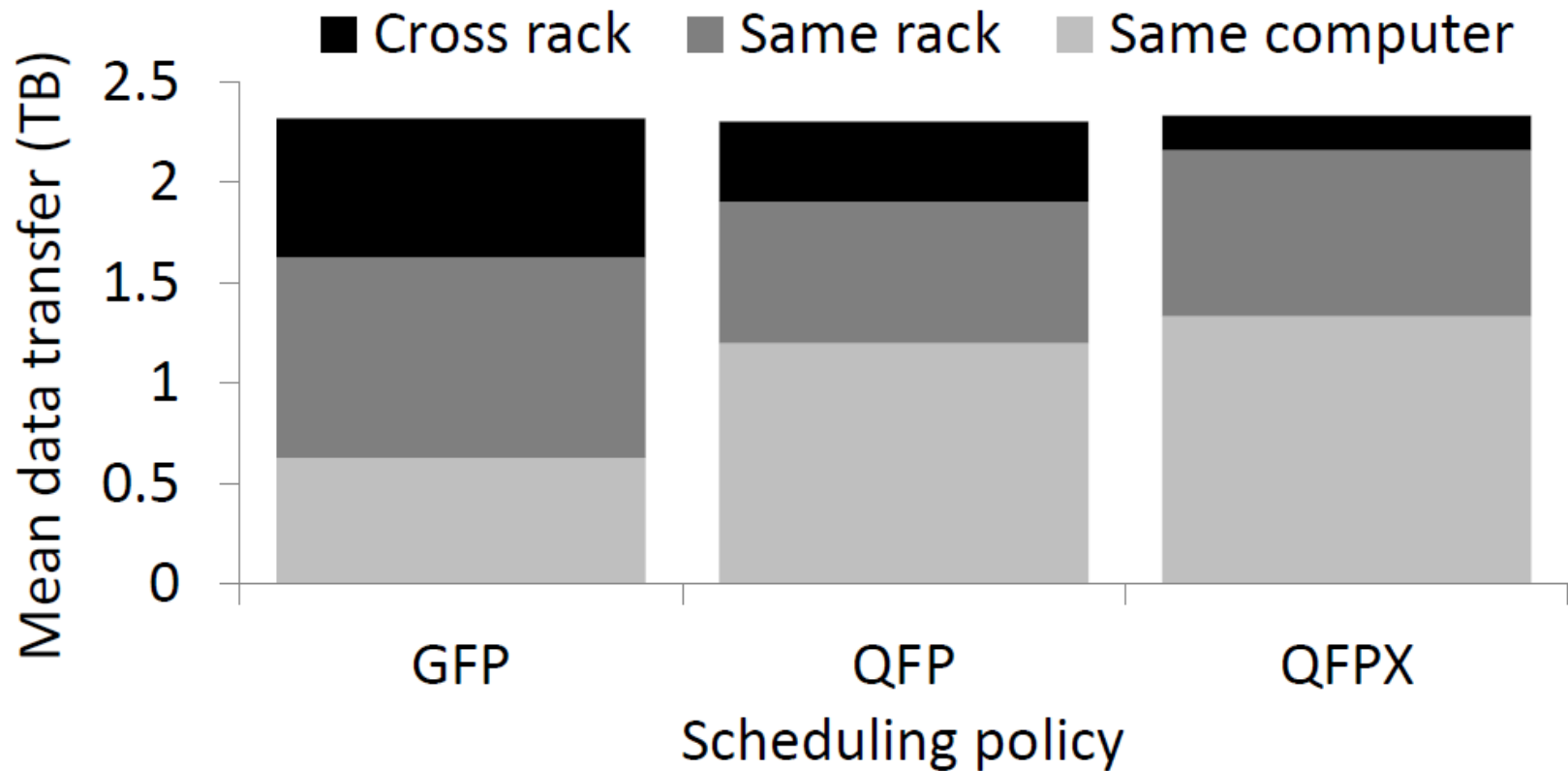
# Running Times



Makespan: total time taken by an experiment until the last job completes.



# Data transfer





# Discussion

- **Correlated constraints**
- **Multi-dimensional capacities:**
  - CPU, disk IO, memory
- **Fair sharing of the network or other resources**

# CA-NFS: A Congestion-Aware Network File System

Alexandros Batsakis, NetApp and Johns Hopkins  
University; Randal Burns, Johns Hopkins  
University; Arkady Kanevsky, James Lentini,  
Thomas Talpey

FAST 2009

# Problems in NFS

- **Congestion of Resources**
  - Selfish clients want to maximize throughput
  - Difficult to represent the congestion of multiple resources as a unified metric
- **False Assumptions**
  - Requests of clients have same priority
  - The benefit of clients increases by maximizing throughput even in congestion

# CA-NFS Overview

- **Congestion-Aware NFS**
  - Measure Congestion for Multiple Resources
    - Usages of resources are monitored as price
  - Schedule Client Operations to React Congestion
    - Asynchronous operations can be deferred depending on server and client states (depending on the price)
    - Try asynchronous operations not to interfere with on-demand synchronous operations

# Pricing Mechanism

$$P_i(u_i) = P_{max} \frac{\{k_i^{u_i} - 1\}}{\{k_i - 1\}}$$

- **Congestion Price**

- $P_i$  – price of resource  $i$ ,
- $P_{max}$  - max price, represents bottleneck
- $u_i$  – utilization of resource  $i$  ( $0 < u_i < 1$ )
- $k_i$  – performance degradation parameter due to congested resource

# Scheduling

- **Pricing**

- Price is increased or decreased corresponding to resource usages of client and server
- Increased price represents congestion of a resource

- **Client reacts to price**

- By comparing advertised server price with local price, client schedules asynchronous operations
  - Accelerate or defer asynchronous writes
  - Issue read-ahead aggressively or prudently

# Pricing Example (1)

Client 1

memory: 80%

RA eff: 10%

hit rate: 40%

price	Writes	Reads
	85	4

Client 2

memory: 10%

RA eff: 85%

hit rate: 90%

price	Writes	Reads
	20	17

Server

memory: 65%

network: 20%

disk: 90%

hit rate: 40%

price	Writes	Reads
	40	12

# Pricing Example (2)

**Client 1**

memory: 50%  
RA eff: 10%  
hit rate: 40%

price	Writes	Reads
	64	4

**Client 2**

memory: 50%  
RA eff: 85%  
hit rate: 90%

price	Writes	Reads
	60	17

**Server**

memory: 65%  
network: 20%  
disk: 60%  
hit rate: 40%

price	Writes	Reads
	55	12



# Scheduling – Asynchronous Writes

- **Server Price**
  - Based on server memory, disk, and network utilization
- **Client Price**
  - Based on client memory
- **Write Acceleration**
  - Clients flushes writes immediately when server load is low
  - Save client memory – more cache hit
  - Reduce write latency
- **Write Deferral**
  - Clients keep writes in local memory when server load is high
  - Save server memory and I/O (disk and memory)
  - Increase write latency and client memory usage

# Examples of Resource Monitoring

- **CPU**

- Utilization at a given time

- **Client and Server Network**

- Average bandwidth over hundreds milliseconds

- **Server Disk**

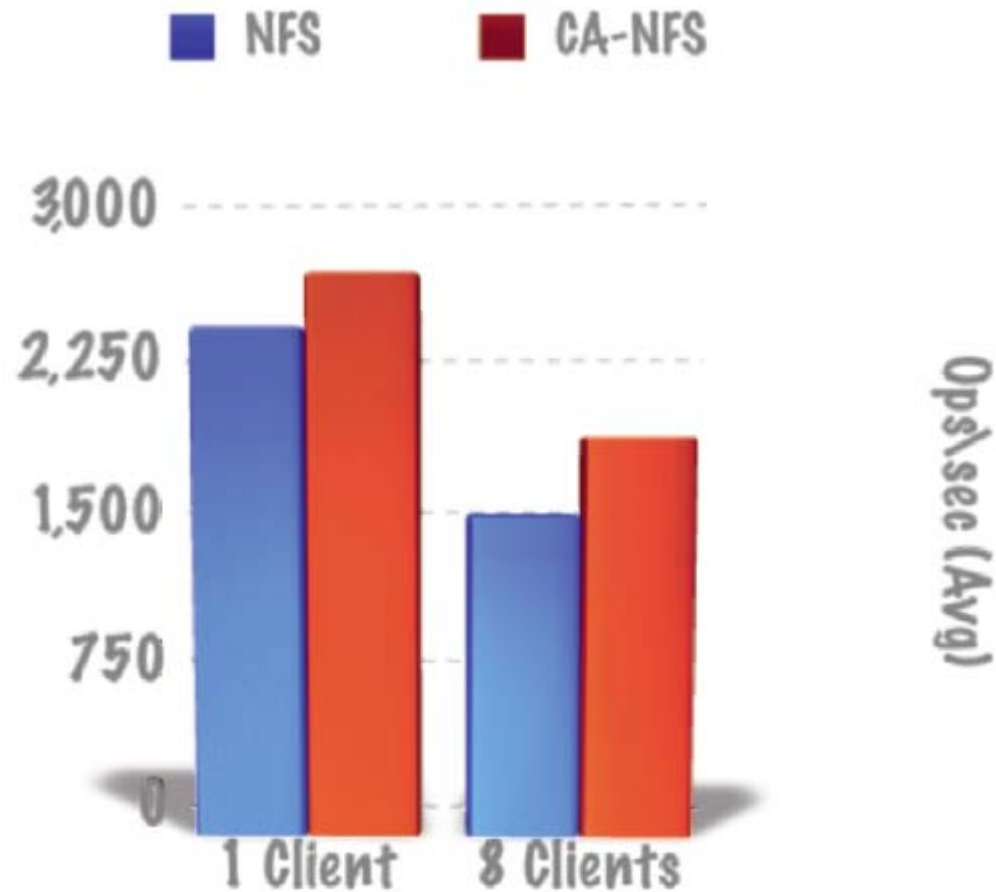
- Sampling the length of device's dispatch queue at regular small time interval

- **Client and Server Memory**

- Calculate the projected cache hit rates using the distribution of read requests

# Experimental Results – File Server

Average client throughput of NFS and CA-NFS for the fileserver workload



# Discussion Points

- **Limitations**

- Not Scalable - works only for small number of clients
- Price can be fluctuated

- **Would the CA-NFS apply to utility computing on cloud?**

- Is pricing sufficient to unify heterogeneous resources?
  - Multiple clients on separate VMs can run on same node
- How to extend single server topology for scalability?
  - Multiple servers run on separate nodes with replicated data

- **How to provide fairness or prioritized services?**