Preemptive ReduceTask Scheduling for Fair and Fast Job Completion

Yandong Wang, Jian Tan, Weikuan Yu,

Li Zhang, Xiaoqiao Meng

AUBURN UNIVERSITY

IBM Research

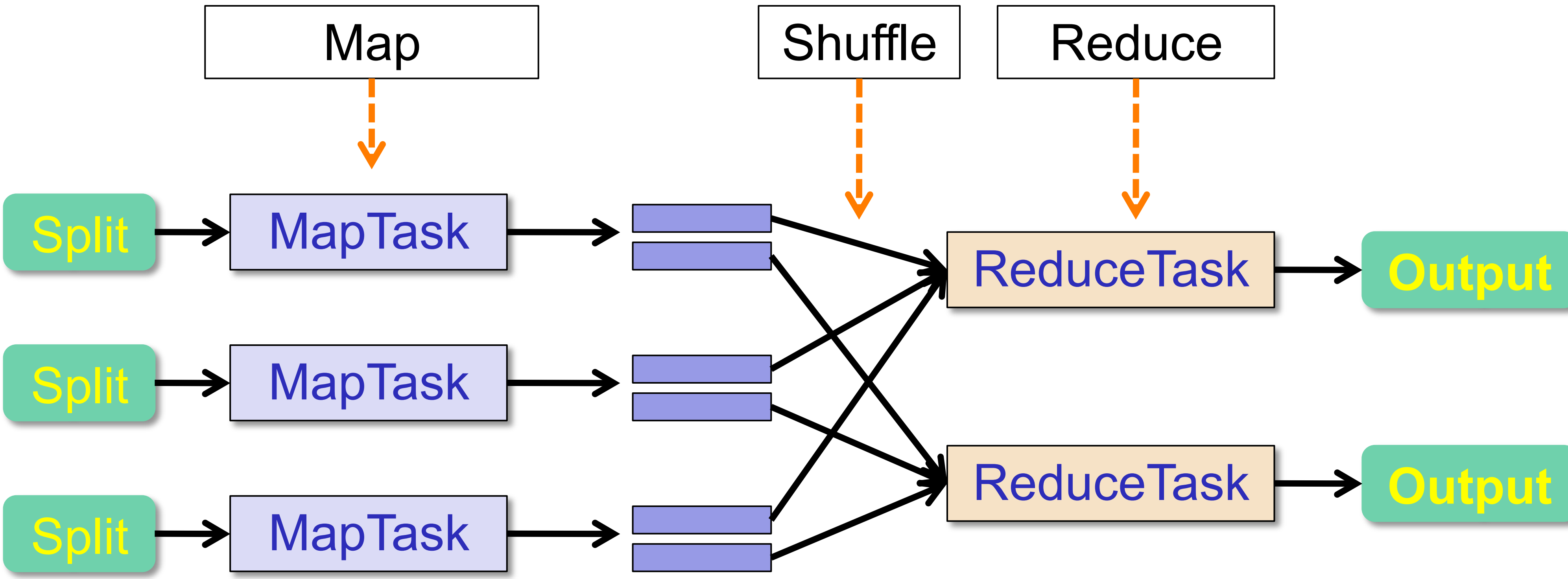
Outline

- Background & Motivation
- Issues in Hadoop Scheduler
- Preemptive ReduceTask
- Fair Completion Scheduler
- Performance Evaluation
- Conclusion and Future Work

Overview

- MapReduce is a programming model for processing massive-scale data.
  - Hadoop: Open-source implementation of MapReduce
- Hadoop has been widely adopted by leading companies.
  - Providing high scalability and strong fault tolerance.
- Data consolidation can be highly beneficial.
  - Co-location of disparate data sets and avoiding data replication cost.
- Mixed workloads of long batch jobs and small interactive queries.
  - Interactive queries are expected to return quickly.
  - Hadoop Fair Scheduler was introduced to allow fair sharing among concurrent jobs.

High-Level Hadoop Overview



- ***Hadoop schedulers strive to overlap the map and shuffle phases to accelerate data processing pipeline.***

Hadoop Fair Scheduler

- A widely used Hadoop scheduler for sharing a Hadoop cluster.
- Providing fairness among concurrently running jobs via max-min fair sharing.
  - Delay scheduling policy are used to provide data locality awareness.
- Tasks occupy slots until successful completion or failure.

AUBURN UNIVERSITY

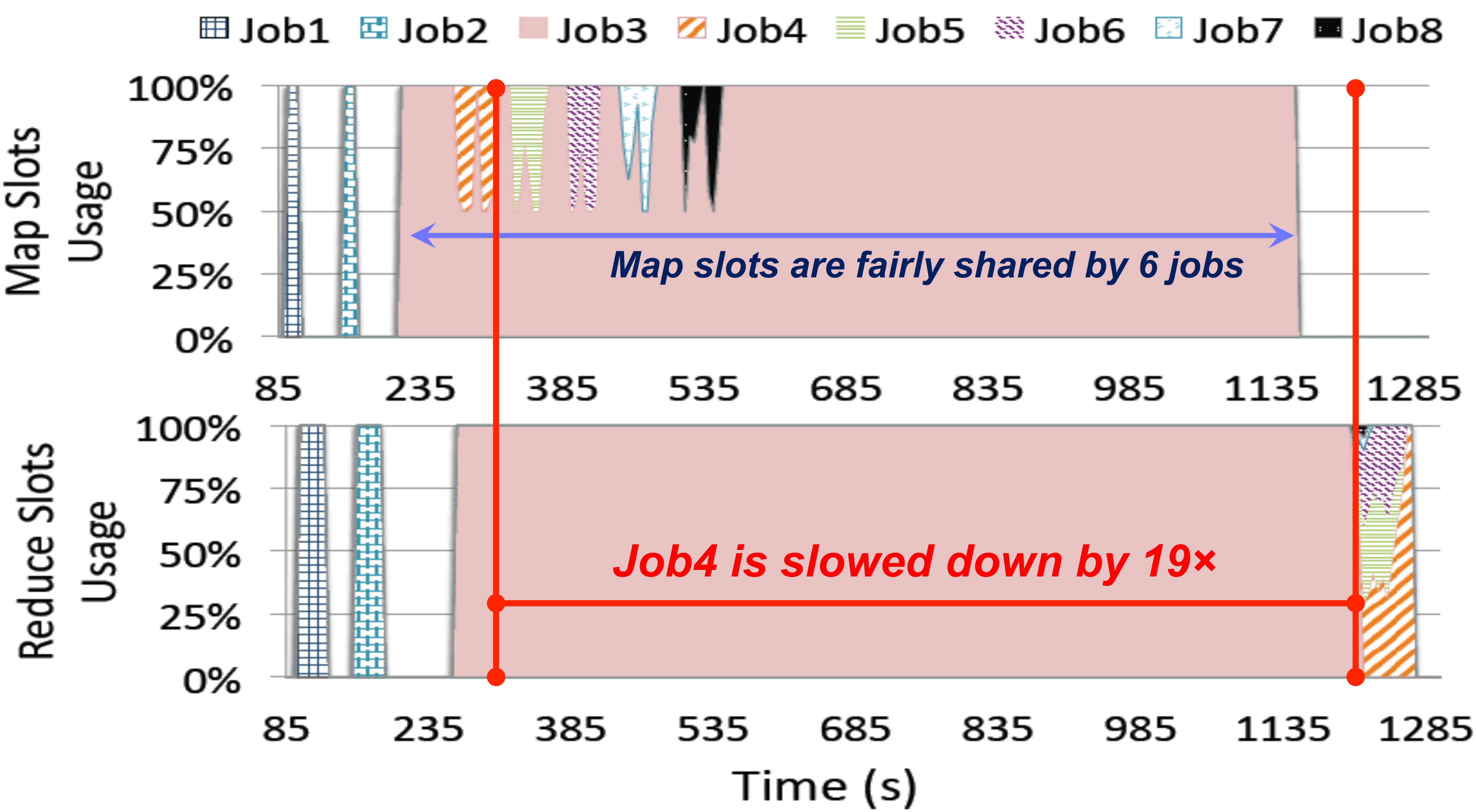
IBM Research

Outline

- Background & Motivation
- Issues in Hadoop Scheduler
- Preemptive ReduceTask
- Fair Completion Scheduler
- Performance Evaluation
- Conclusion and Future Work

Unfair Reduce Slots Allocation

- Monopolizing behavior of long ReduceTasks from the large job (Job3).
- On average, last 5 small jobs are severely slowed down by 15×.



AUBURN UNIVERSITY

IBM Research

Distinct Execution Pattern between Map and Reduce Tasks

- Current Hadoop schedulers treat map and reduce tasks similarly.

Distinctions	MapTask	ReduceTask
Execution Time	*Short-lived*	*Long-lived*
Execution Phase	*Single-phase*	*Multi-phase*
Execution Dependency	*None*	*Map phase*

Distinct Execution Pattern between Map and Reduce Tasks

- Current Hadoop schedulers treat map and reduce tasks similarly.

Distinctions	MapTask	ReduceTask
[illegible]	[illegible]	[illegible]
Execution Phase	Single phase	Multi-phase
Execution Dependency	***None***	***Map phase***

It is critical for Hadoop schedulers to be aware of these different patterns.

Existing Efforts

- Hadoop introduces slow start [1]
  - Mitigating the starvation but at the cost of slowing down the data processing pipeline.
  - Impacting the execution time of small jobs.
- Coupling scheduling policy from IBM[2]
  - Similar to slow start which let monopolization progressively happen
- Copy-Compute Splitting[3]
  - Performance is unknown, no results was reported.

[1]: "mapred.reduce.slowstart.completed.maps" .

[2]: Jian Tan, Xiaoqiao Meng, Li Zhang, "Coupling scheduler for MapReduce/Hadoop", HPDC'12.

[3]: "Job Scheduling for Multi-User MapReduce Cluster", Berkeley, Technical Report *UCB/EECS-2009-55.*

Fundamental Solutions

*How to achieve both high **Efficiency** and **Fairness** ?*

- How to tackle monopolizing behavior of long running ReduceTasks ?
  - Existing schedulers ignore long-lasting ReduceTasks, once they are launched, they occupy resource until completion or failure.
  - Introducing a new mechanism: Preemptive ReduceTask.

- How to coordinate two-phase job scheduling ?
  - MapReduce adopts two-phase scheme (map and reduce) to schedule tasks. However less contemplation has been given to coordinate them.
  - A new scheduler: Fair Completion Scheduler.

Outline

- Background & Motivation
- Issues in Hadoop Scheduler
- Preemptive ReduceTask
- Fair Completion Scheduler
- Performance Evaluation
- Conclusion and Future Work

Preemptive ReduceTask

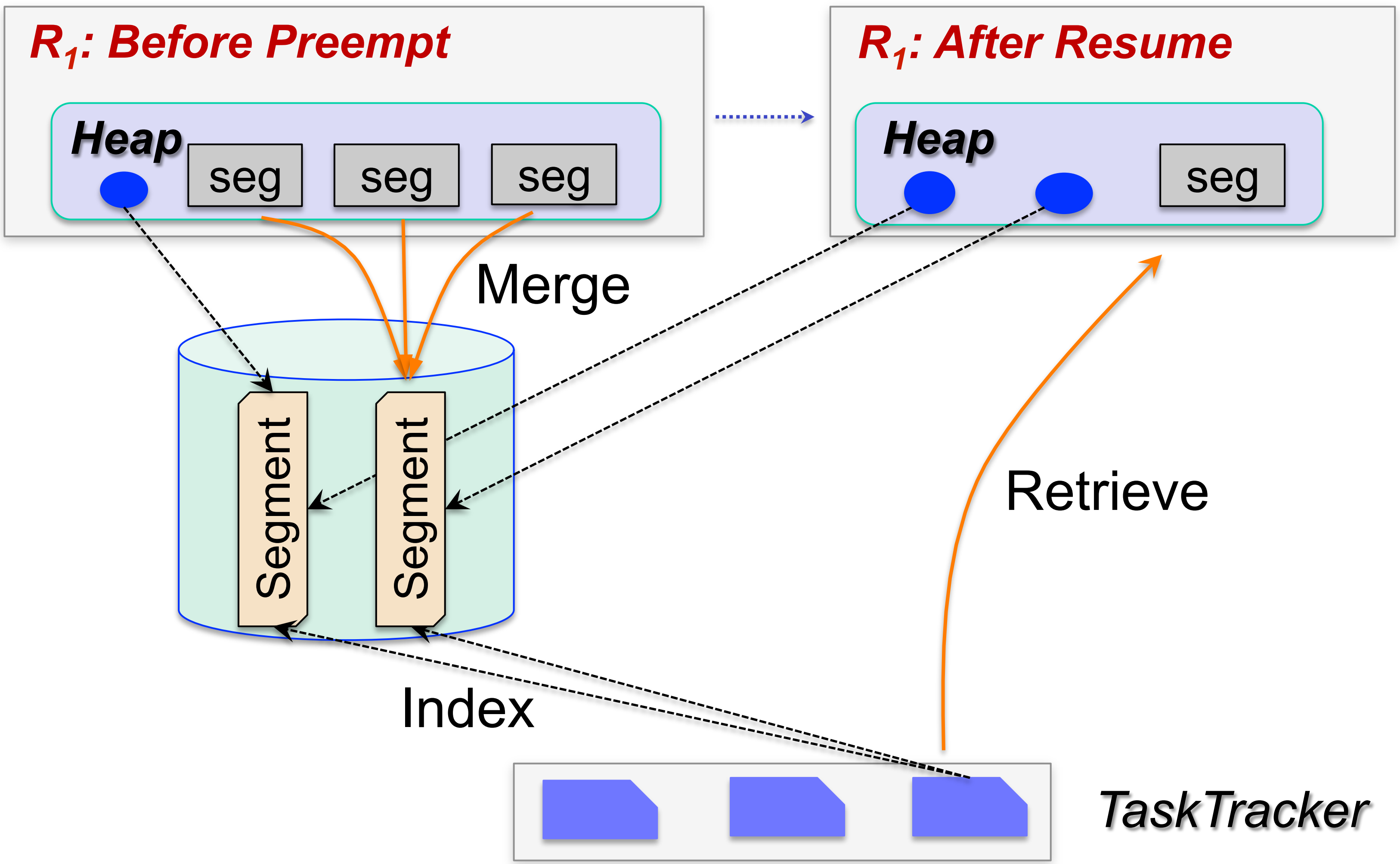
- Lightweight work-conserving preemption mechanism.
  - Preserving previous computation and I/O.
  - Providing lightweight preemption with no noticeable performance impact.

- Different from Linux process suspend commend (“*Kill -STOP $PID*”).
  - Preemptive ReduceTask releases the reduce slot.

- Superior to current killing preemption mechanism.
  - Killing can lead to significant waste of computation and I/O.

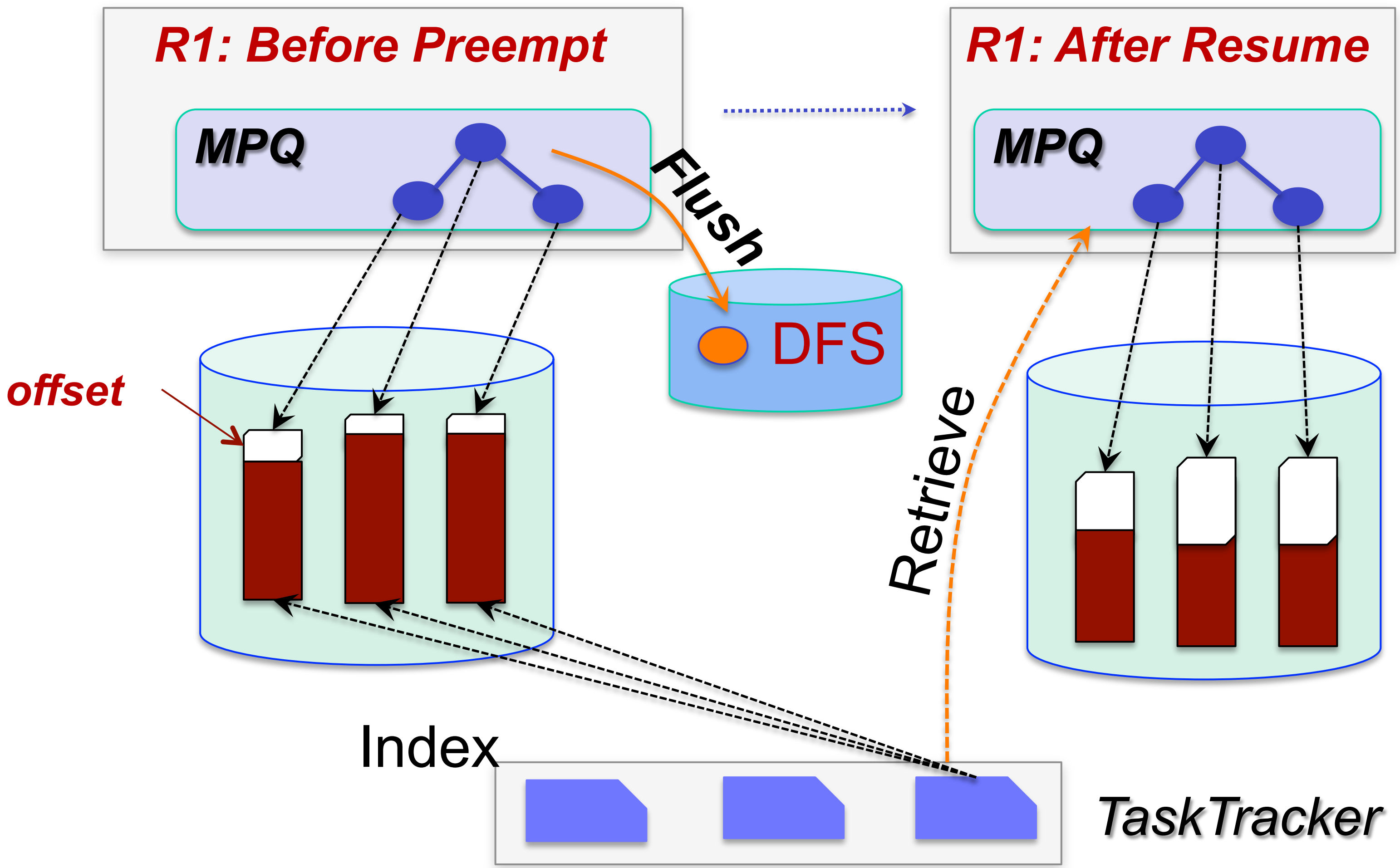
Preemption During Shuffle Phase

- Only merging the in-memory intermediate data, while maintaining on-disk intermediate data untouched.



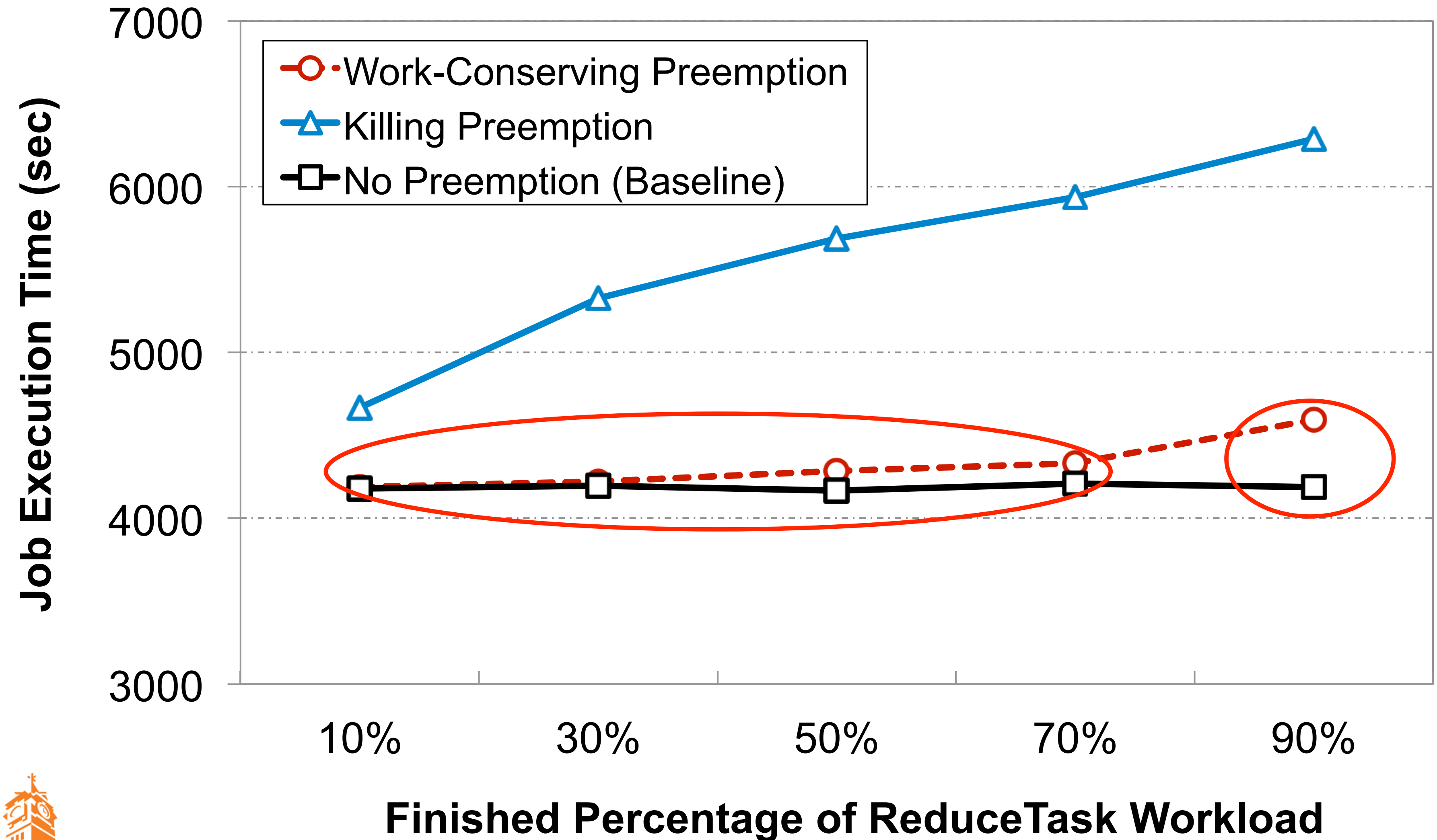
Preemption During Reduce Phase

- Recording the current offset of each segment and minimum priority queue
- Preemption occurs at the boundary of intermediate <key,value> pairs.



Evaluation of Preemptive ReduceTask

- Terasort benchmark with 512GB input data on a cluster of 20 worker nodes.



Outline

- Background & Motivation
- Issues in Hadoop Scheduler
- Preemptive ReduceTask
- Fair Completion Scheduler
- Performance Evaluation
- Conclusion and Future Work

Fair Completion Scheduler

- Prioritizing ReduceTasks from jobs with the shortest remaining map phases.
  - Allowing small jobs to preempt long-running ReduceTasks from large jobs.
  - MapTask scheduling follows max-min fair sharing policy.

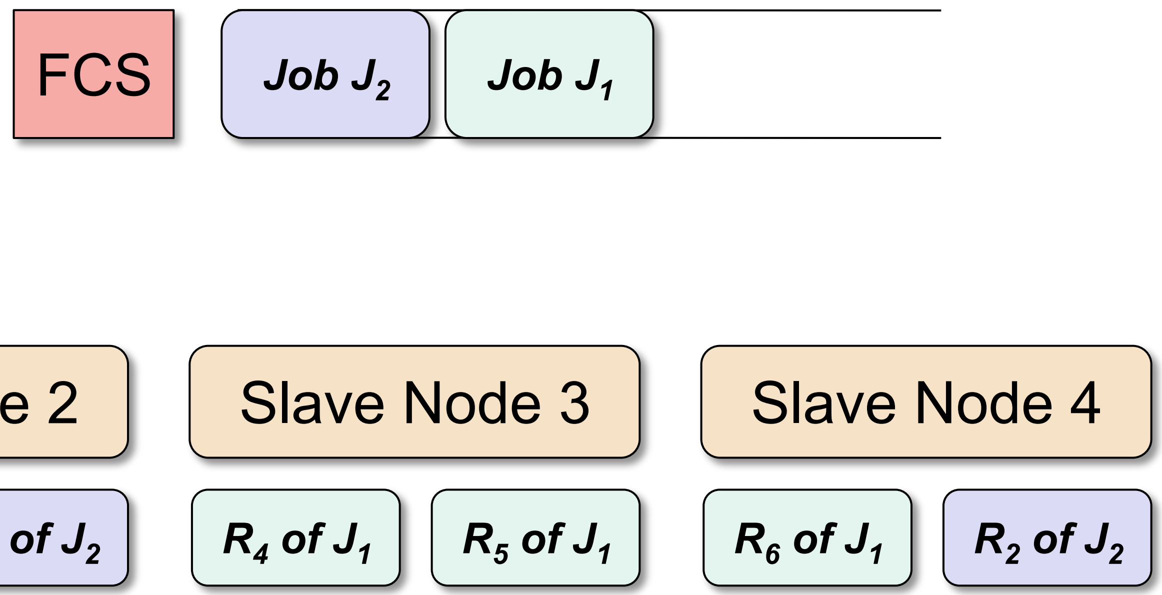
- When remaining map phases are equal, prioritizing ReduceTasks from jobs with least remaining reduce data.

- Detecting the job execution slowdown caused by preemptions.
  - Preventing ReduceTasks of large jobs from being preempted for too long and too many times.

Fair Completion Scheduling Details

	Remaining Map Phase	Remaining Reduce Data	Reduce
J_1	1000 s	100GB	6
J_2	200 s	10GB	2

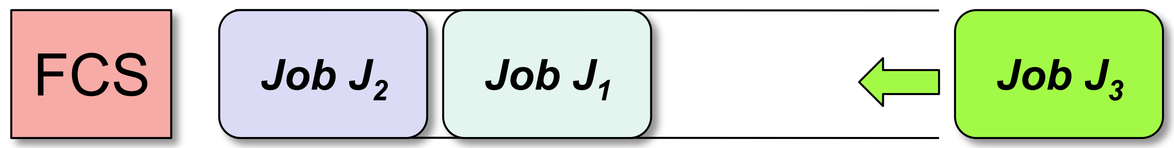
Sort Running Jobs:
(1): According to remaining map time
(2): According to remaining reduce data

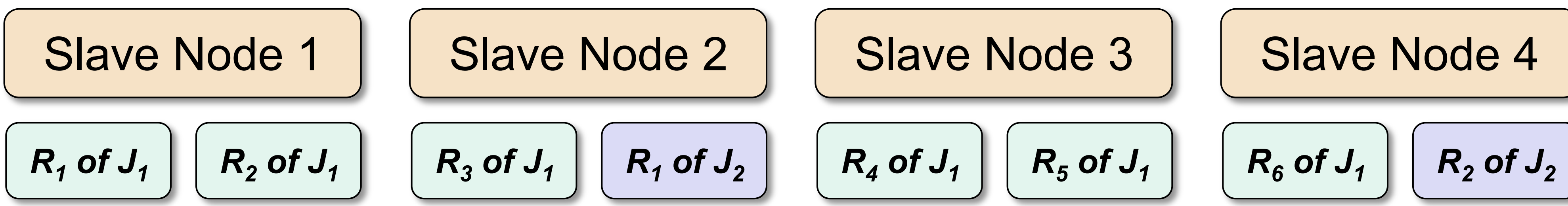


Fair Completion Scheduling Details

	Remaining Map Phase	Remaining Reduce Data	Reduce
J_1	1000 s	100GB	6
J_2	200 s	10GB	2
J_3	80 s	8GB	4

Sort Running Jobs:
(1): According to remaining map time
(2): According to remaining reduce data

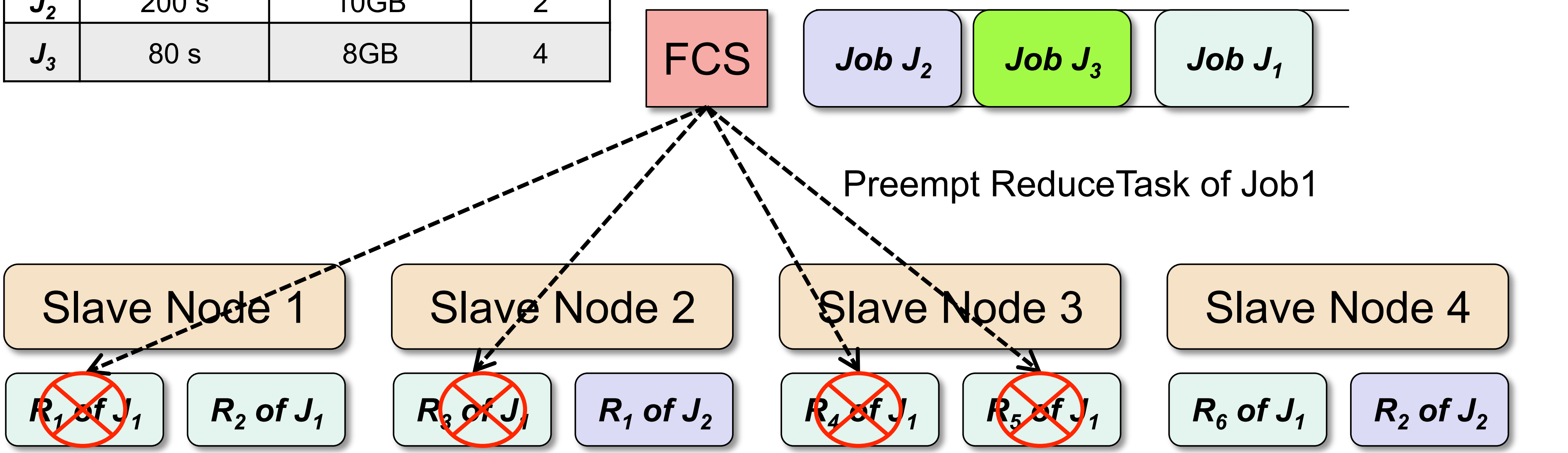




Fair Completion Scheduling Details

	Remaining Map Phase	Remaining Reduce Data	Reduce
J_1	1000 s	100GB	6
J_2	200 s	10GB	2
J_3	80 s	8GB	4

Sort Running Jobs:
(1): According to remaining map time
(2): According to remaining reduce data



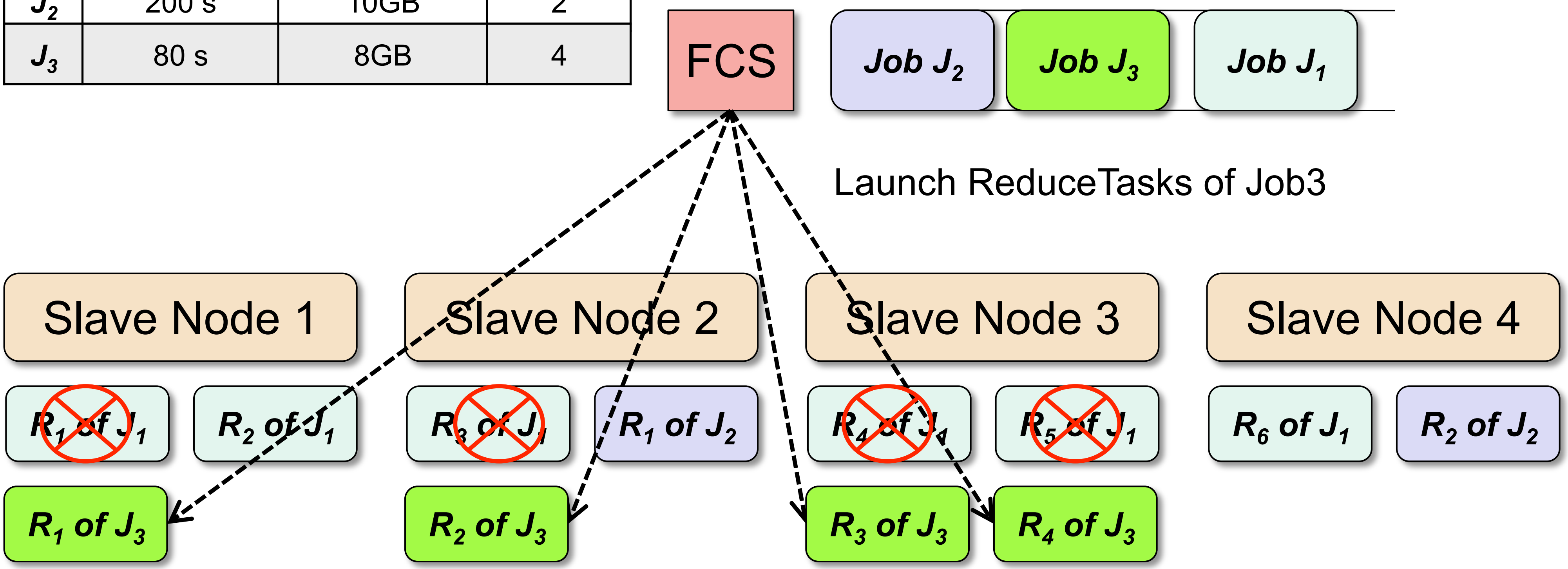
AUBURN UNIVERSITY

IBM Research

Fair Completion Scheduling Details

	Remaining Map Phase	Remaining Reduce Data	Reduce
J_1	1000 s	100GB	6
J_2	200 s	10GB	2
J_3	80 s	8GB	4

Sort Running Jobs:
(1): According to remaining map time
(2): According to remaining reduce data

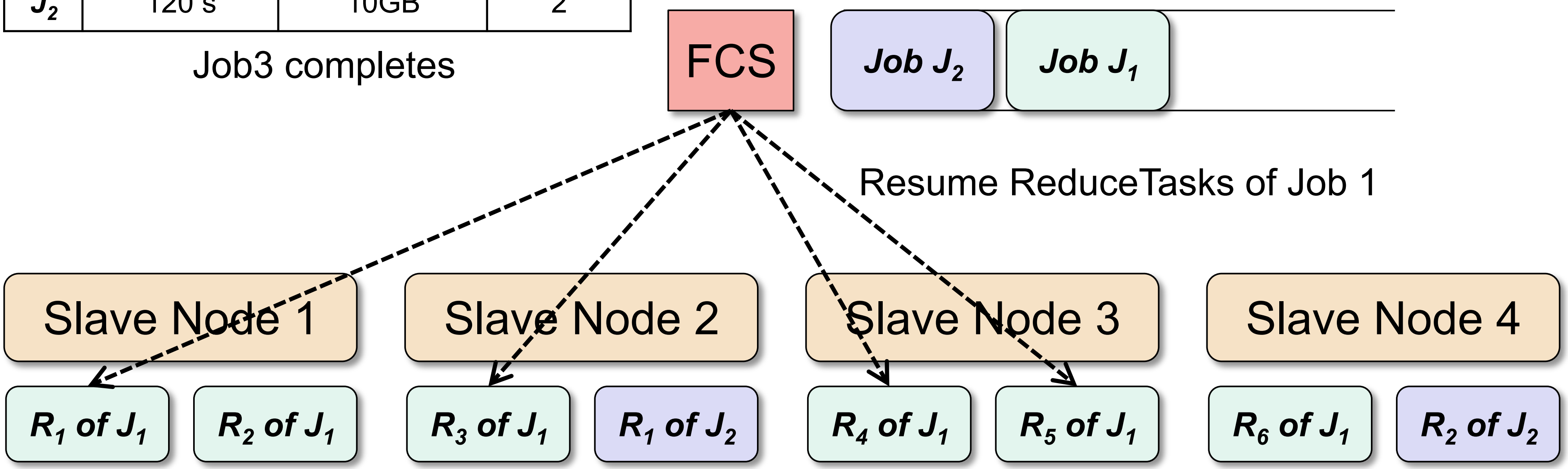


Fair Completion Scheduling Details

	Remaining Map Phase	Remaining Reduce Data	Reduce
J_1	800 s	100GB	6
J_2	120 s	10GB	2

Sort Running Jobs:
(1): According to remaining map time
(2): According to remaining reduce data

Job3 completes



Outline

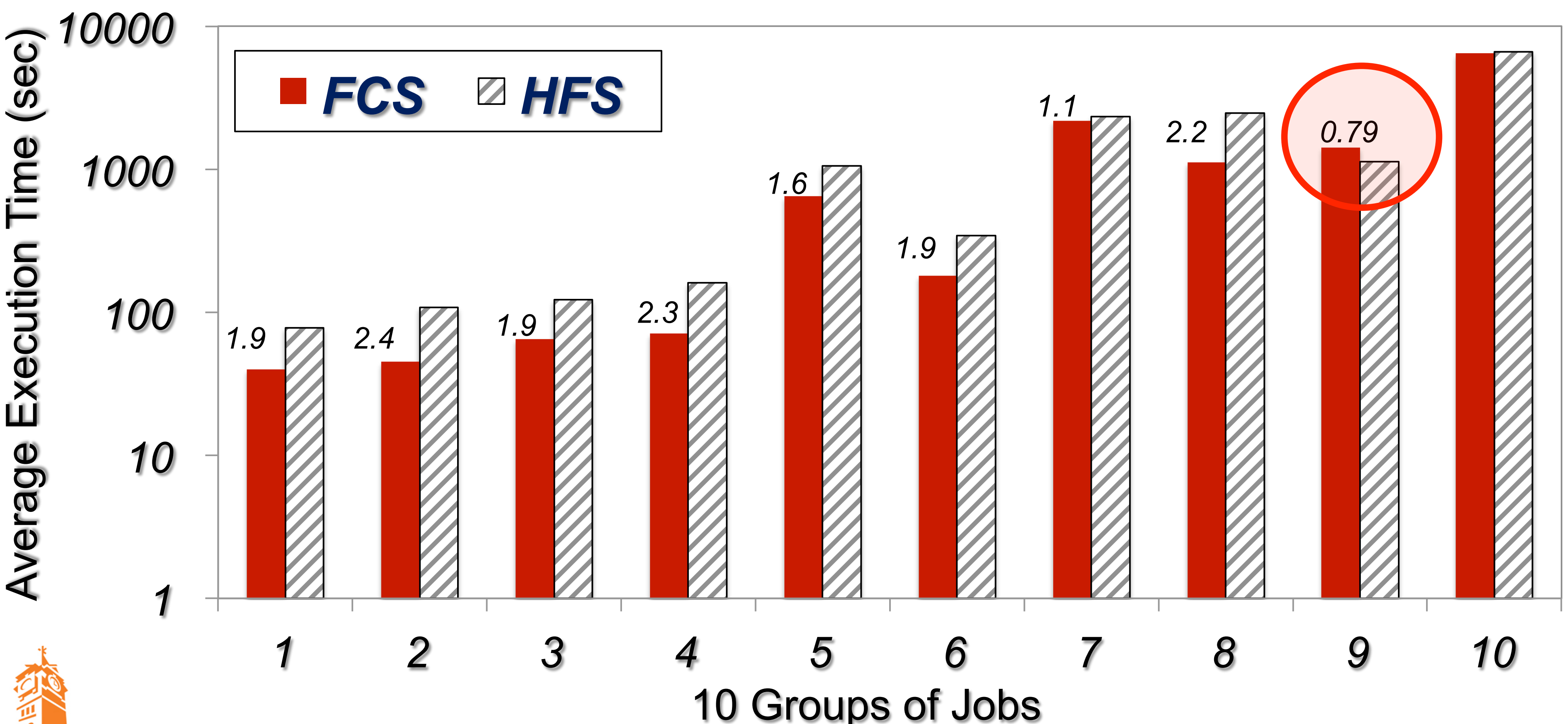
- Background & Motivation
- Issues in Hadoop Scheduler
- Preemptive ReduceTask
- Fair Completion Scheduler
- Performance Evaluation
- Conclusion and Future Work

Testbed and Benchmarks/Metrics

- Hardware configuration
  - A cluster of 46 nodes. 4 2.67GHz hex-core Intel Xeon CPUs, 24GB memory and two hard disks.
- Software configuration:
  - Hadoop 1.0.0 and its Fair Scheduler. 8 map slots and 4 reduce slots on each nodes.
- Gridmix2 and Tarazu benchmarks:
  - Map-heavy workload
  - Reduce-heavy workload
  - Scalability evaluation

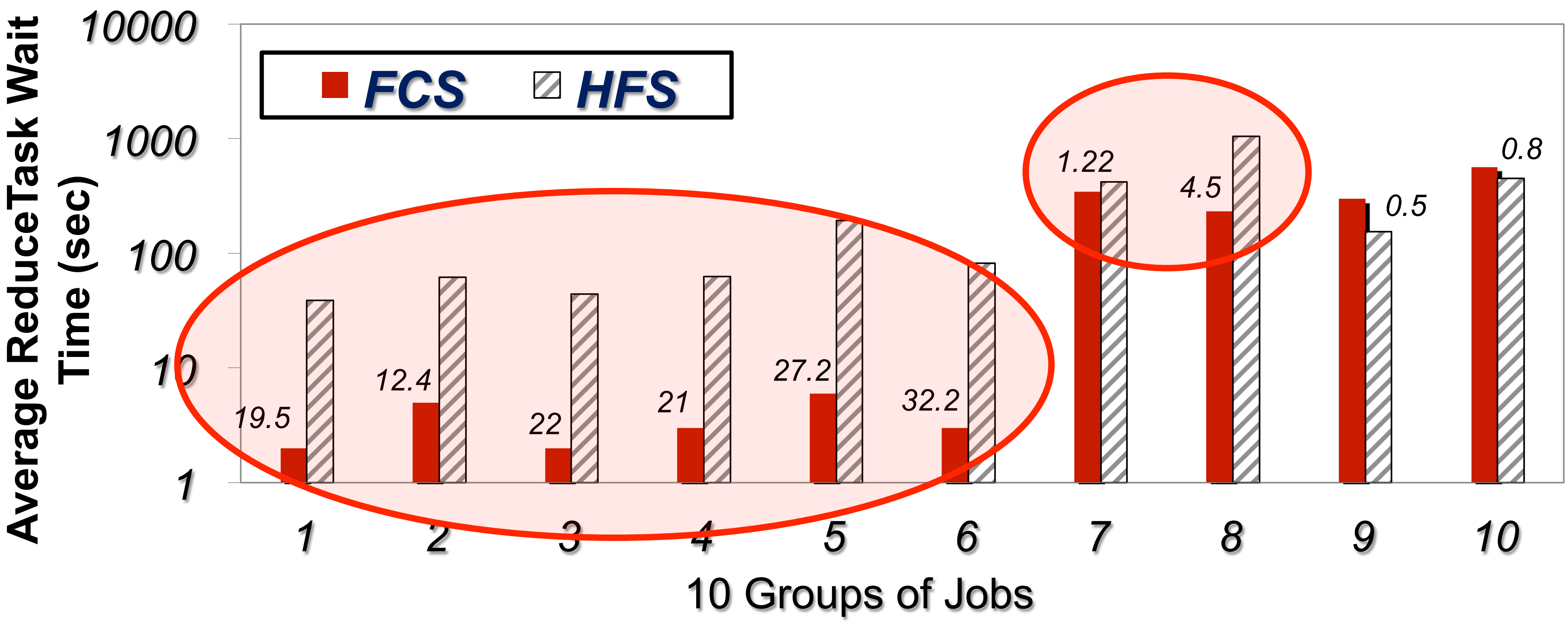
Results for Map-heavy Workload

- FCS reduces average execution time by 31% (171 jobs).
- Significantly speeds up small jobs, slightly slow down large jobs.



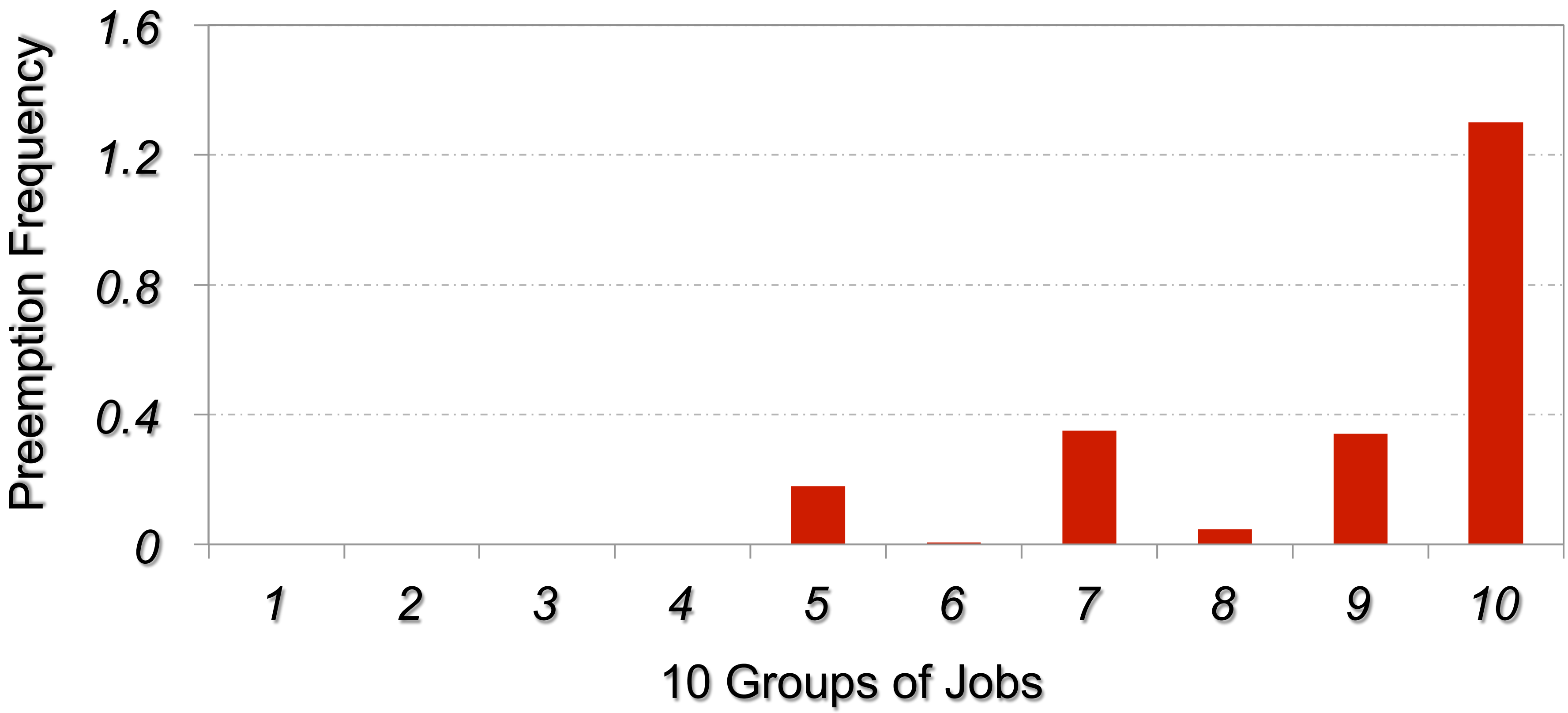
Average ReduceTask Wait Time

- Small jobs are benefited from significantly shortened reduce wait time.
- Waiting time are reduced by 22× for the jobs in the first 6 groups.



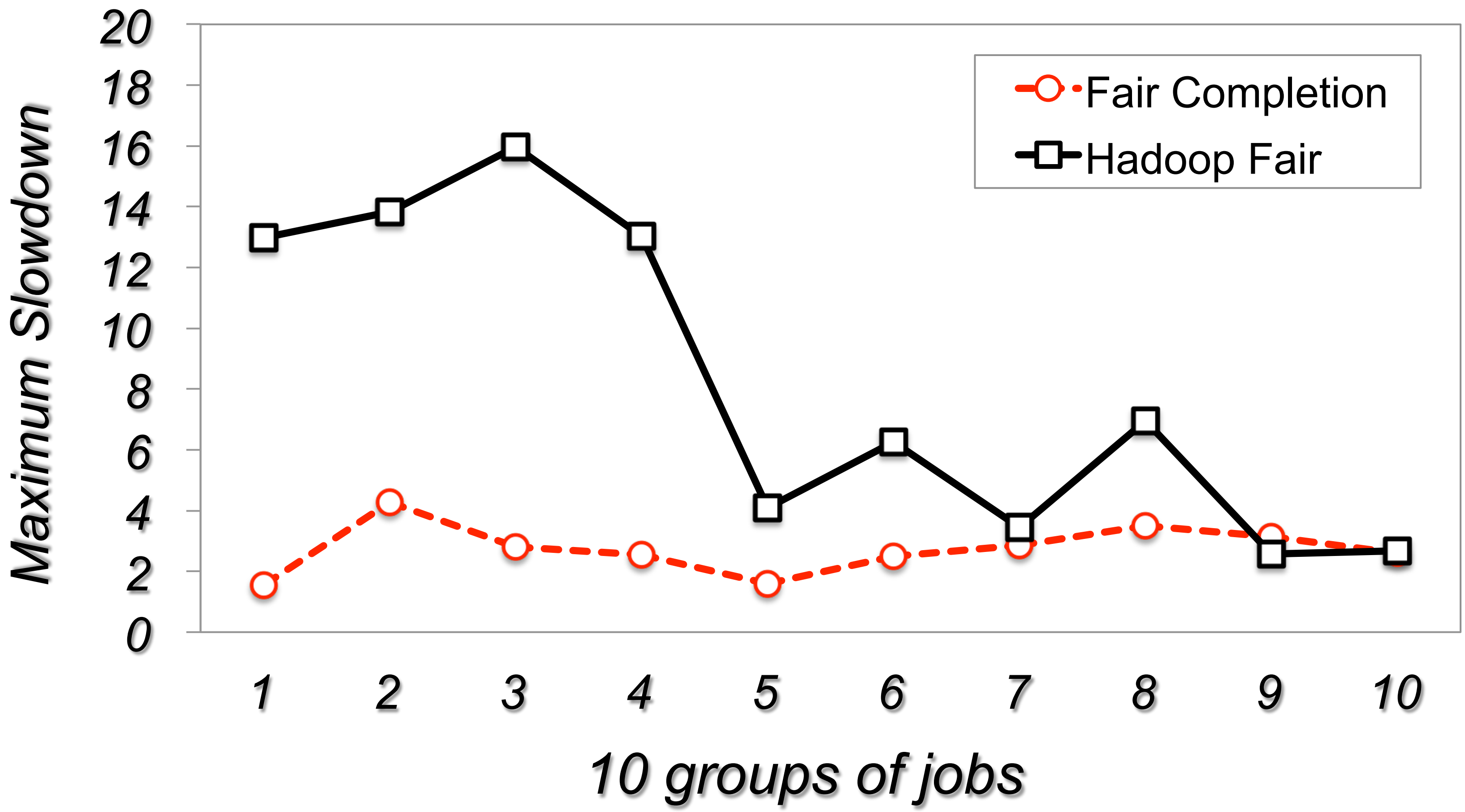
Preemption Frequency

- FCS controls the preemption frequency to avoid excessive preemptions.



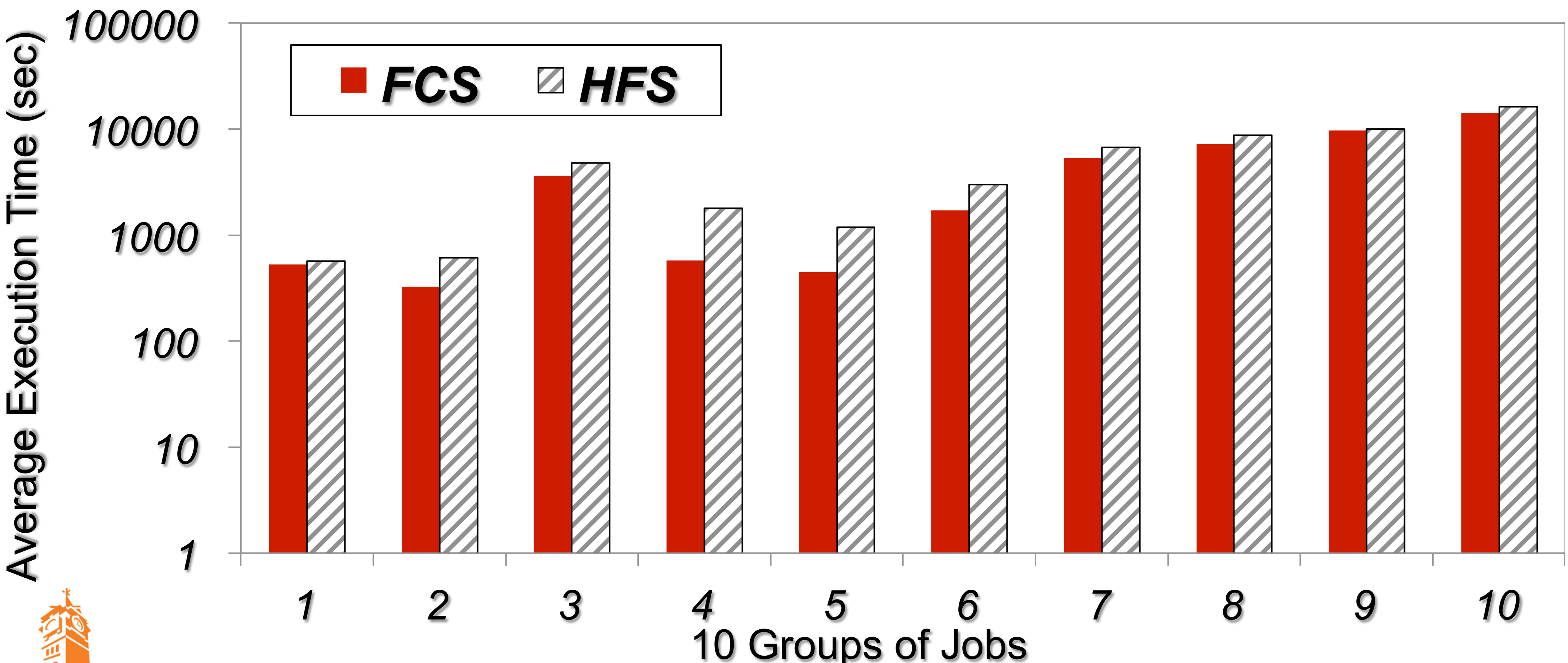
Fairness Evaluation: Maximum Slowdown

- FCS improves the fairness by 66.7% on average.
- Achieving nearly uniform maximum slowdown for all groups of jobs.



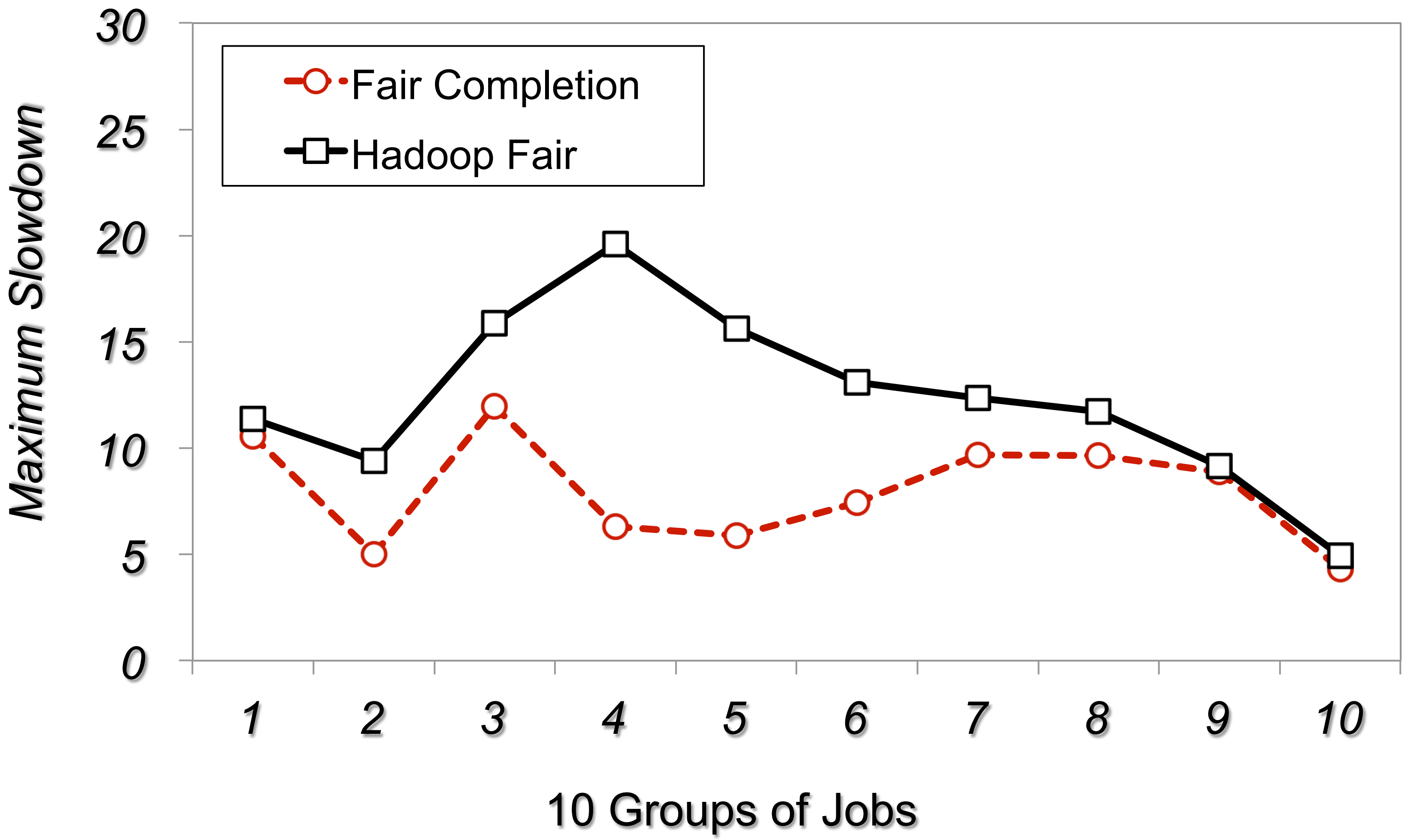
Results for Reduce-heavy Workload

- FCS reduces average execution time by 28% (171 jobs).
- FCS accelerates all types of jobs in the reduce-heavy workload.
  - Impact of preemption on large job is not heavy due to they are still in map phases.



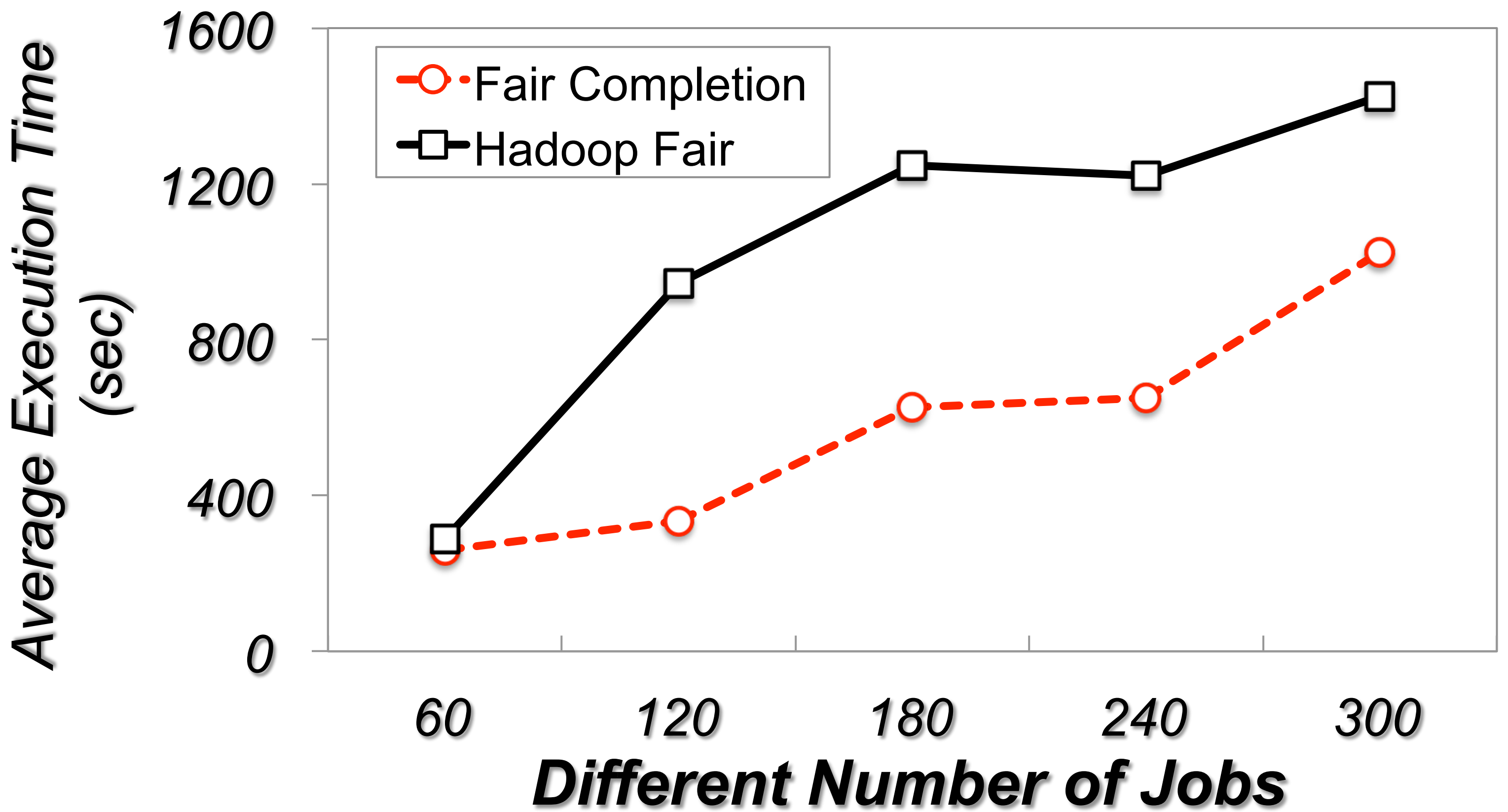
Fairness of Reduce-heavy Workload

- FCS improves the fairness by 35.2% on average.



Scalability Evaluation with GridMix-2

- FCS reduces the average execution time by 39.7%.
- Small improvement at 60 due to dominant number of small jobs.



Outline

- Background & Motivation
- Issues in Hadoop Scheduler
- Preemptive ReduceTask
- Fair Completion Scheduler
- System Evaluation
- Conclusion and Future Work

Conclusion and Future Work

- Identify the inefficiencies in existing Hadoop schedulers.
- Preemptive ReduceTask provides an efficient preemption approach.
- Fair Completion Scheduler is introduced to improve the efficiency and fairness of the concurrently running jobs.
- Preemptive ReduceTask provides opportunities to improve the fault tolerance mechanism.
- More preemptive scheduling policy can be implemented based on Preemptive ReduceTask.

Sponsors of Our Research

NSF

NASA

U.S. DEPARTMENT OF ENERGY

Lawrence Livermore National Laboratory

Mellanox TECHNOLOGIES

AUBURN UNIVERSITY

OAK RIDGE National Laboratory

NVIDIA

HPC ADVISORY COUNCIL NETWORK OF EXPERTISE

TeraGrid

Thank You and Questions ?