Designing Provably Efficient Data-Center Schedulers: A Probabilistic Approach

Ness Shroff
Depts. of ECE and CSE
The Ohio State University
E-mail: shroff@ece.osu.edu

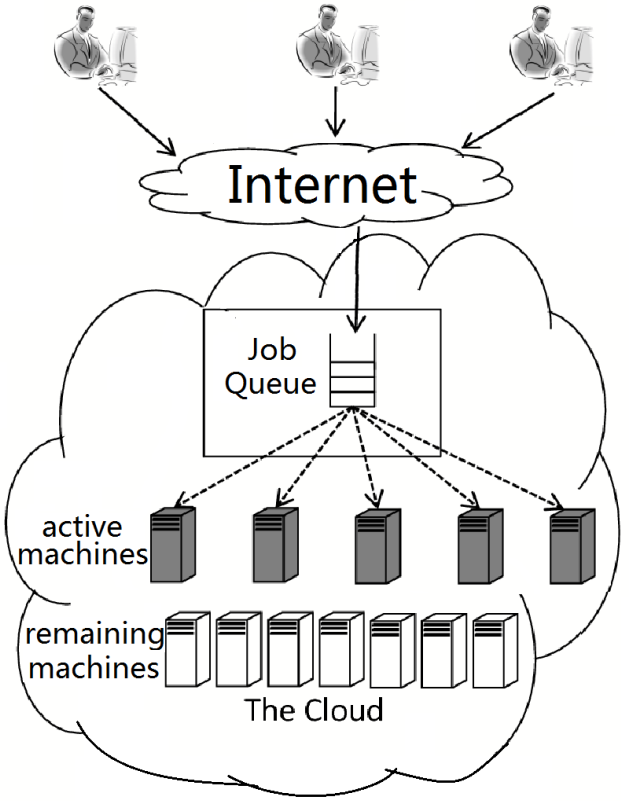
Joint work with **Yousi Zheng**
Prasun Sinha

February 11, 2014

Part of this work was published in IEEE INFOCOM' 2013

Data Centers

- Data Centers
  - Facility containing massive numbers of machines
    - Has roots in huge computer rooms of the early ages!
  - Services: IaaS, PaaS, SaaS
  - They process very large datasets
    - Use a (variation of) programming model called Map Reduce
      - Developed (popularized) by Google
      - Nearly ubiquitous (Google, IBM, Facebook, Microsoft, Yahoo, Amazon, eBay, twitter…)

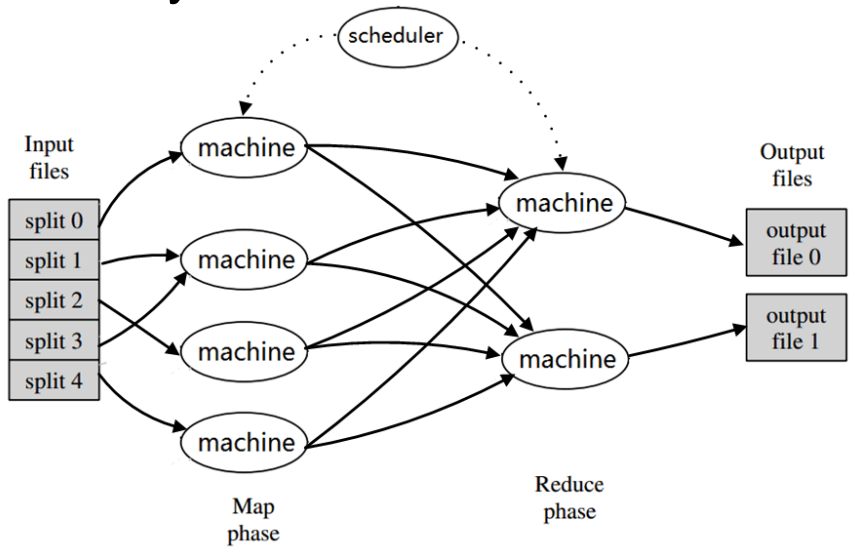


MapReduce

- Framework for processing highly parallel problems
- Inspired by Map and Reduce phases commonly used in functional programming
  - Their purpose is not the same as their original forms.
- Easy to use and make scalable
  - Don't need extensive database training to be used.
  - Nearly ubiquitous (Google, IBM, Facebook, Microsoft, Yahoo, Amazon, eBay, twitter…)
- Used in a variety of different applications
  - Distributed Grep
  - Distributed Sort
  - AI, scientific computation
  - Large scale pdf generation
  - Geographical data
  - Image processing…

MapReduce (cont'd)

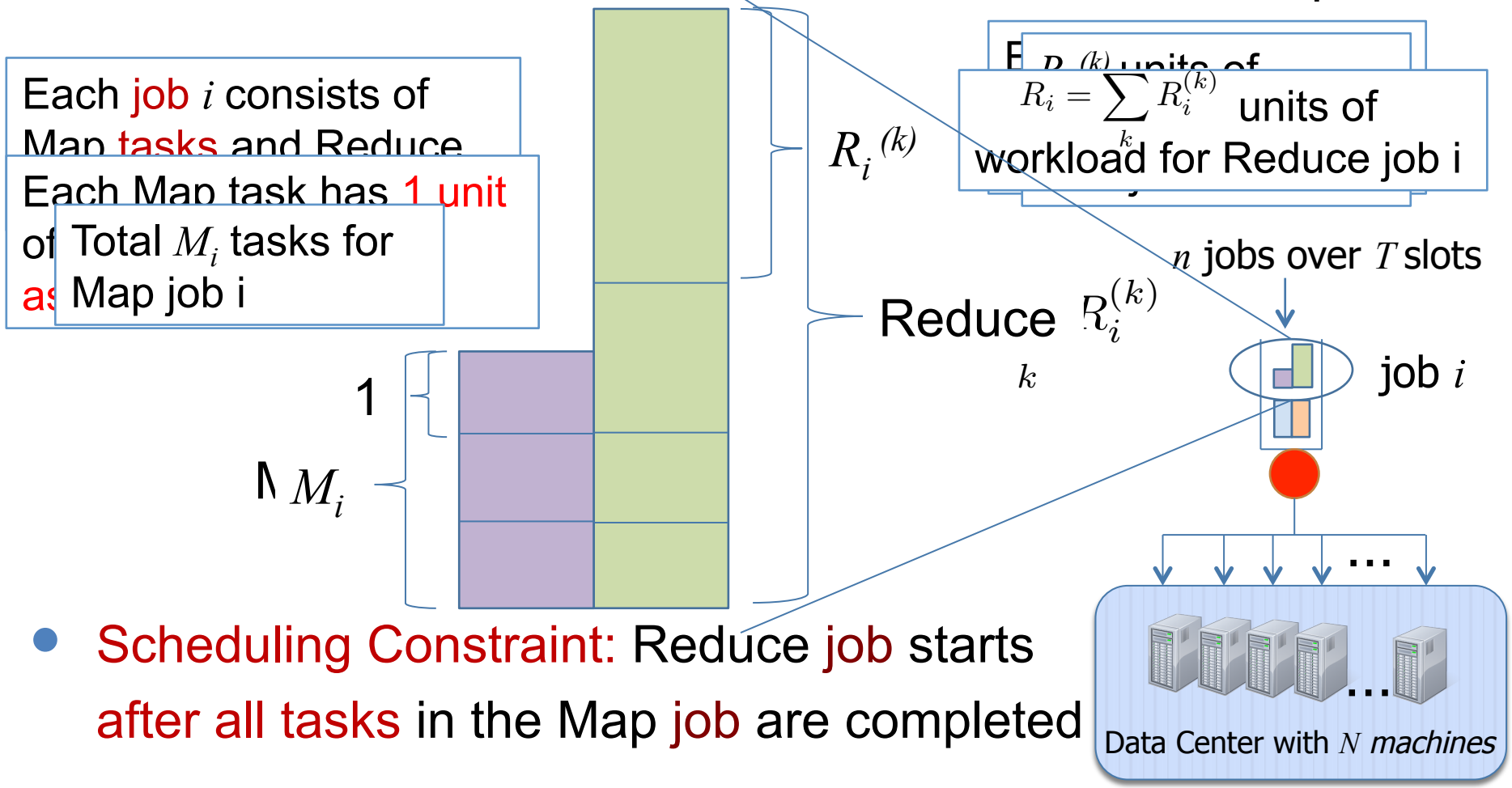
- Consists of two elemental processes
  - Each arriving job consists of a set of Map Tasks and Reduce Tasks
- Map phase:
  - Takes an input and divides into many small sub-problems
  - Operations can run in parallel on potentially different machines
- Reduce phase
  - Combines the output of Map
  - Usually occurs after the Map phase is completed
  - Operations can run on parallel machines…



Goal: Schedule these Map and Reduce tasks in order to minimize the total flow time in the system

Model

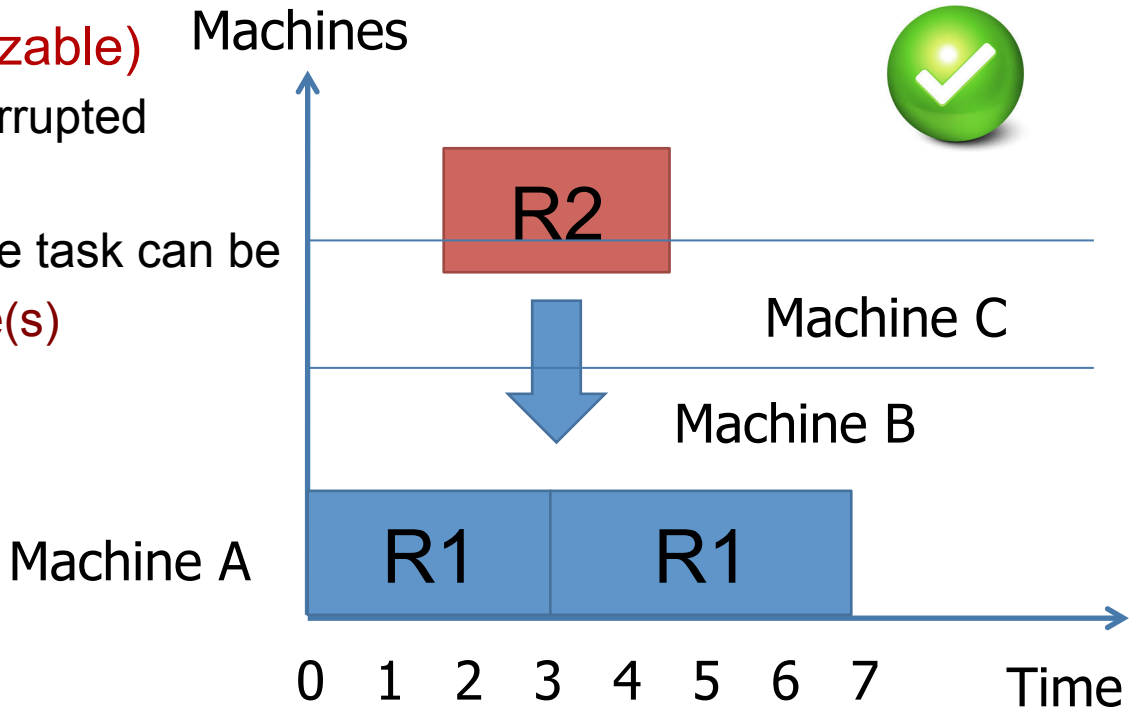
- Time is slotted
- *N* "Machines": Each machine runs 1 unit of workload per slot



- Scheduling Constraint: Reduce job starts after all tasks in the Map job are completed

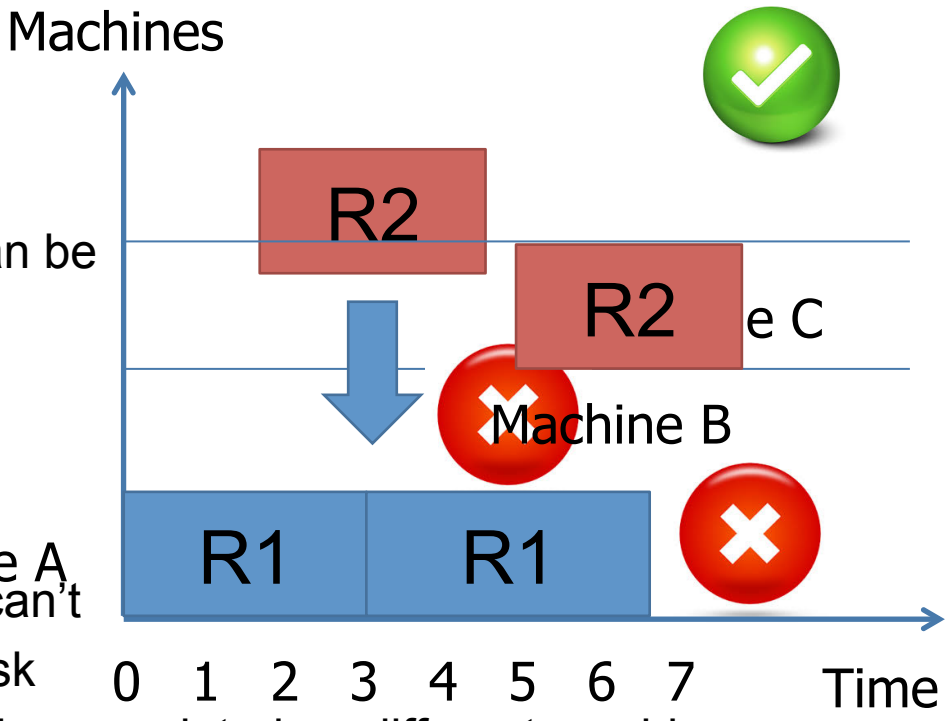
Types of Schedulers: Treatment of Reduce Tasks

- Preemptive (also parallelizable)
  - Reduce tasks can be interrupted at the end of a slot
  - Remaining workload in the task can be executed on any machine(s)
  - Reasonable if overhead of data-migration is small
- Non-preemptive

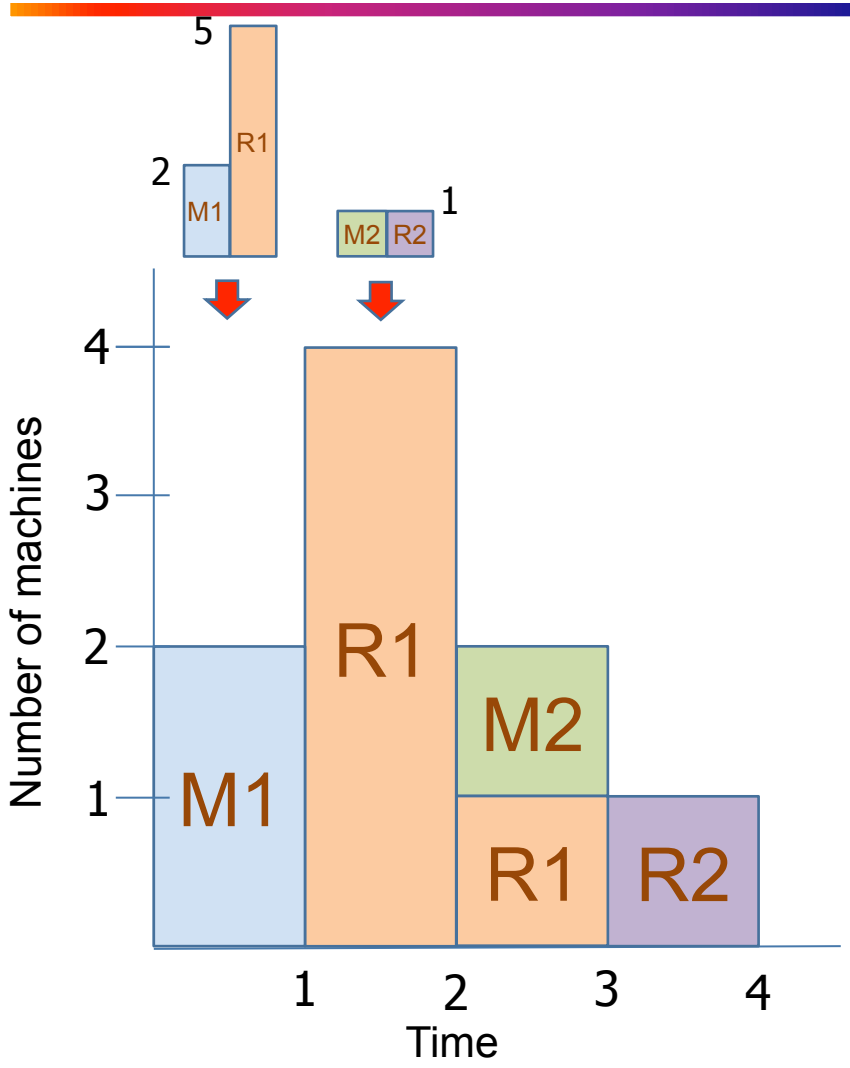


Types of Schedulers: Treatment of Reduce Tasks

- Preemptive (also parallelizable)
  - Reduce tasks can be interrupted at the end of a slot
  - Remaining workload in the task can be executed on any machine(s)
  - Reasonable if overhead of data-migration is small
- Non-preemptive
  - Once a Reduce task is started, it can't be interrupted till the end of the task
  - An individual Reduce task cannot be completed on different machines
  - But different tasks from same job can be assigned to different machines
- Note: Since Map tasks have unit workload, they cannot be interrupted.
  - In practice Map tasks may be > 1 unit, but small



MapReduce: FCFS scheduler



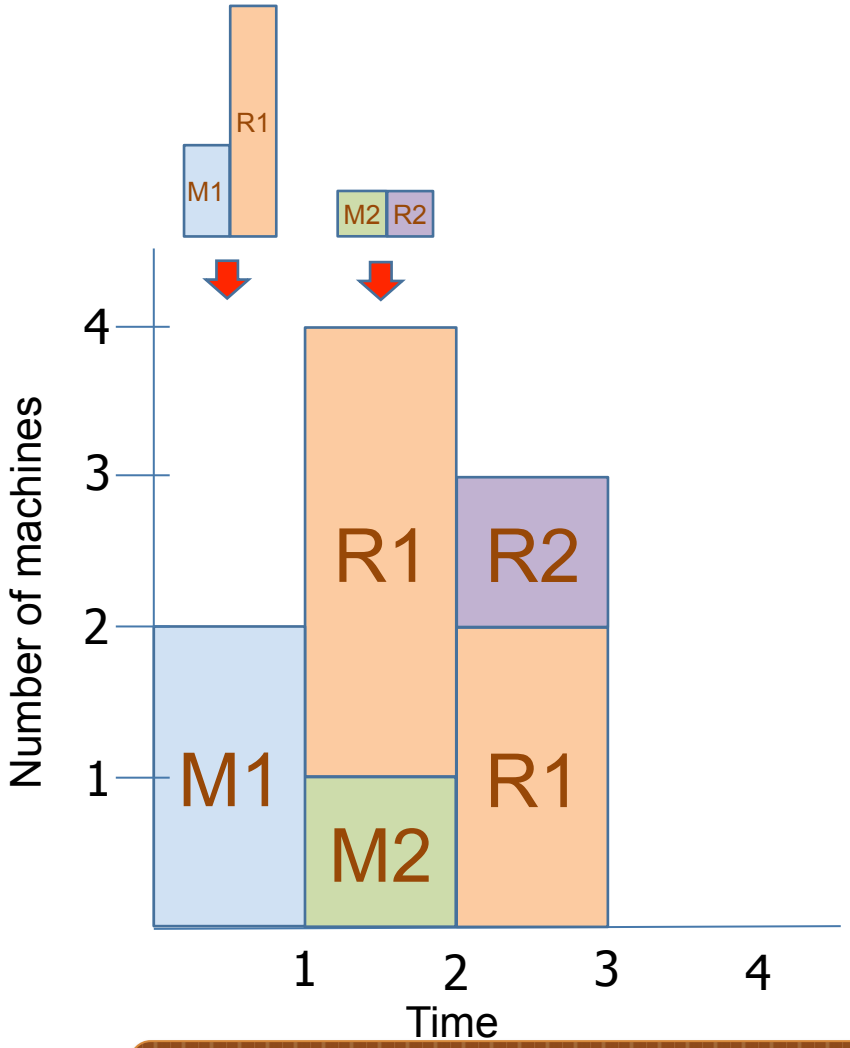
FCFS scheduler with
4 machines, 2 jobs

Map must finish before
Reduce can start

Flow-time of a job: time spent by job in the system

FT of Job 1: 3 time slots
FT of Job 2: 3 time slots
Total FT : 6 time slots

MapReduce: Smarter Scheduler



"Smarter Scheduler"

Flow-time: time in the system

FT of Job 1: 3 time slots

FT of Job 2: 2 time slots

Total FT: 5 time slots

Goal: Minimize total flow-time

Flow-Time Minimization Problem: Preemptive Scenario

$$\min_{m_{i,t}, r_{i,t}} \quad \sum_{i=1}^{n} \left(f_i^{(r)} - a_i + 1 \right)$$

Minimize flow-time

$$s.t. \quad \sum_{i=1}^{n} (m_{i,t} + r_{i,t}) \leq N, \ \ \forall t,$$

Total # machines is N

$$\sum_{t=a_i}^{f_i^{(m)}} m_{i,t} = M_i, \ \ m_{i,t} \geq 0, \ \ \forall i, \forall t,$$

Total map workload is M_i for job i

$$\sum_{t=f_i^{(m)}+1}^{f_i^{(r)}} r_{i,t} = R_i, \ \ r_{i,t} \geq 0, \forall i, \forall t.$$

Total reduce workload is R_i for job i

Flow-Time Minimization Problem
Non-Preemptive Scenario

$$\min_{m_{i,t}, r_{i,t}^{(k)}} \sum_{i=1}^{n} \left(f_i^{(r)} - a_i + 1 \right)$$

Minimize flow-time

$$s.t. \quad \sum_{i=1}^{n} (m_{i,t} + \sum_{k=1}^{K_i} r_{i,t}^{(k)}) \leq N, \ \forall t,$$

Total # machines is N

$$\sum_{t=a_i}^{f_i^{(m)}} m_{i,t} = M_i, \ m_{i,t} \geq 0, \ \forall i,$$

Total map workload is M_i for job i

$$\sum_{t=f_i^{(m)}+1}^{f_i^{(r)}} r_{i,t}^{(k)} = R_i^{(k)}, \ \forall i, \forall k \in \{1, ..., K_i\},$$

Total reduce workload is $R_i^{(k)}$ for task k from K_i tasks in job i

$$r_{i,t}^{(k)} = 0 \text{ or } 1, \ r_{i,t}^{(k)} = 1 \text{ if } 0 < \sum_{s=0}^{t-1} r_{i,s}^{(k)} < R_i^{(k)}.$$

Non-preemptive

Both Problems are NP-hard in the strong sense

Evaluation Metrics

- Θ : set of all possible schedulers (including non-causal)
- F^{ζ} : Total Flow time of scheduler ζ (sum of FT of all jobs)
- Let $$F^* = \min_{\zeta \in \Theta} F^{\zeta}$$
- Competitive ratio: A given online policy *S* is said to have a competitive ratio of *c* if for any arrival pattern

$$\frac{F^S}{F^*} \leq c$$

- Efficiency ratio: A given online policy *S* is said to have an efficiency ratio of γ if for a class of random arrival patterns (over an infinite time horizon)

$$\limsup_{T \to \infty} \frac{F^S}{F^*} \leq \gamma \text{ with probability } 1$$

Competitive Ratio in Non-Preemptive Scenario

THE OHIO STATE UNIVERSITY

Theorem

For any constant c_0 and any online scheduler *S*, there are sequences of arrivals and workloads in the non-preemptive case, such that the competitive ratio c is greater than c_0

➔ No policy gives a bounded (constant) competitive ratio

Proof idea

- In non-preemptive scenario, tasks cannot be interrupted.
- All the future arrivals are delayed by the running tasks.
- Delays are cumulative.

Finding optimal competitive ratio under preemptive scenario remains an open problem

Efficiency Ratio Analysis

- Assumptions:
  - Job arrivals: i.i.d. (rate λ in each slot)
    - Generalization: Job arrivals follow Markov Process
  - Total workload (summed over all jobs) at each time slot must have finite moments
  - Traffic intensity $\rho < 1$

Key Results:

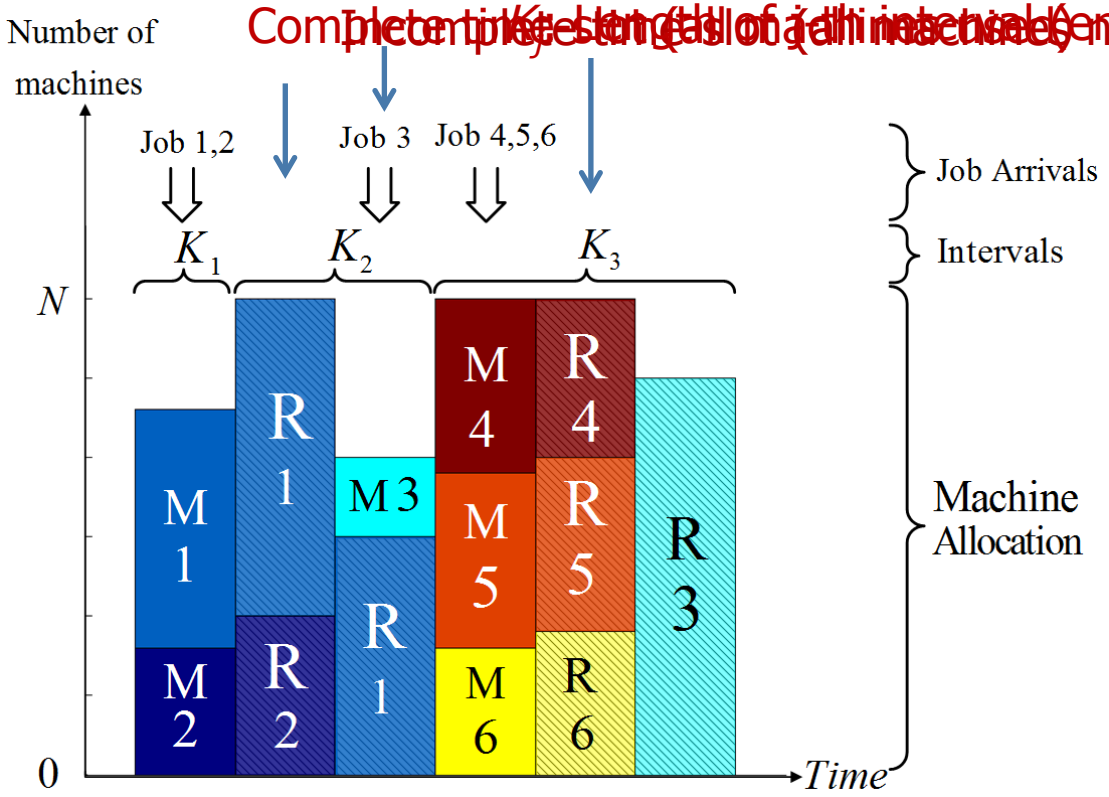
- Efficiency Ratio is bounded by a constant for any work-conserving scheduler for both preemptive and non-preemptive scenarios
- Developed a specific work-conserving algorithm (ASRPT) that has an efficiency ratio of 2 for preemptive scenario.

Analysis Outline (Bounded efficiency ratio)

- Divide up time into repeating intervals
  - Interval corresponds to the consecutive times slots until a time-slot occurs with an idle machine(s)
- Bound the moments of these intervals with constants
- Bound the efficiency ratio using the moments

- Analysis applies to:
  - Preemptive and non-preemptive scenarios
  - Allows jobs to have infinite but light-tailed support

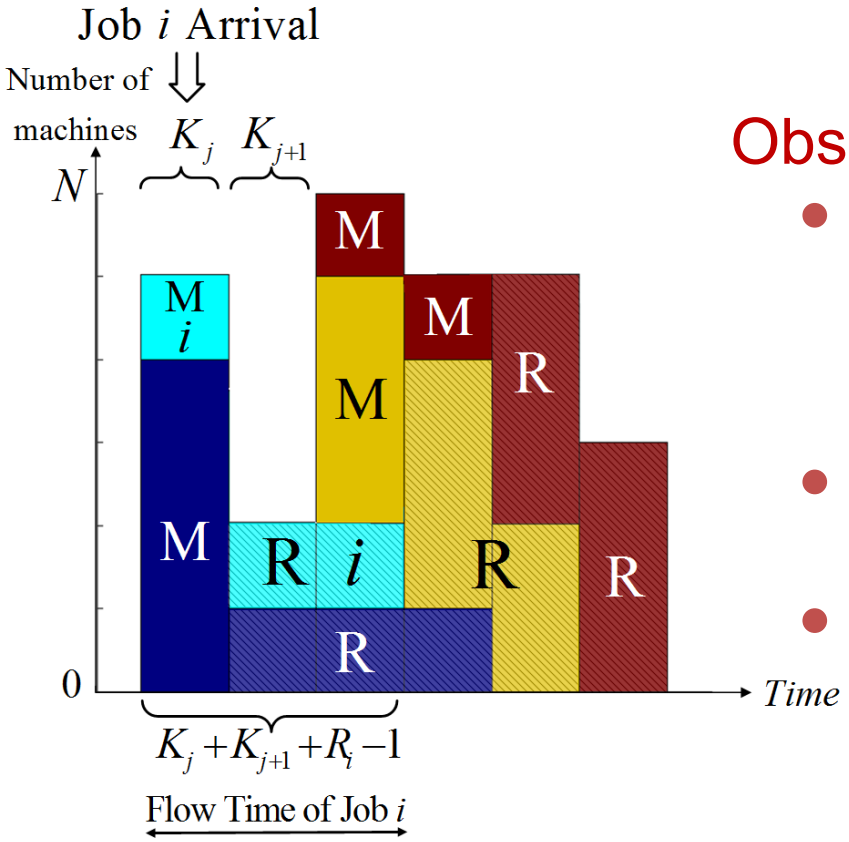
Preemptive Scenario: An Example



Observations for a job arriving in interval *j* (work-conserving)

- Map task finishes in interval *j* (otherwise last time-slot would be complete)
- Reduce tasks finish in interval *j* or interval *j+1*
- ➔ Flow time per job is bounded, wp1, if the interval sizes (moments) are bounded
- ➔ γ is bounded wp1.

Non-preemptive Scenario: An Example



Observation for reduce tasks:

- Unlike preemptive scenario, in the non-preemptive scenario, each job arriving in interval j will **start** (not finish) all its reduce tasks in interval j or interval $j+1$
- Reduce tasks must be finished by the end of interval $j+1$ plus time $R_i - 1$.
- Hence, if (all moments) of the interval are bounded then the flow times (per job) are bounded wp1 ➔ γ is bounded wp1.

Bounds on the Moments of the Interval

Lemma: If the scheduler is work-conserving, then for any given number H , there exists a constant B_H, such that $E\left[K_j^H\right] < B_H$, for all *j*.

$$B_1 \triangleq E[K_j] = \min_{\epsilon \in (0, N-\lambda(\overline{M}+\overline{R}))} \left\{ 1 + \lceil \frac{(N-1)R_{max}}{\epsilon} \rceil + \frac{2e^{l(N-\epsilon)} - 1}{e^{l(N-\epsilon)}(e^{l(N-\epsilon)} - 1)^2} \right\}$$

$$B_2 \triangleq E[K_j^2] = \min_{\epsilon \in (0, N-\lambda(\overline{M}+\overline{R}))} \left\{ \left(1 + \lceil \frac{(N-1)R_{max}}{\epsilon} \rceil \right)^2 + \frac{4e^{2l(N-\epsilon)} - 3e^{l(N-\epsilon)} + 1}{e^{l(N-\epsilon)}(e^{l(N-\epsilon)} - 1)^3} \right\}$$

where $l(a) = \sup_{\theta \geq 0} \left(\theta a - \log(E[e^{\theta W_s}]) \right)$.

Rough idea:

- Since the traffic intensity ρ <1, and the arriving workload in each slot is *i.i.d* & light-tailed, the number of consecutive complete time slots will decay exponentially (Chernoff's Theorem)
- ➔ The moments of interval K are bounded

Efficiency Ratio: Preemptive

For any work-conserving scheduling policy:

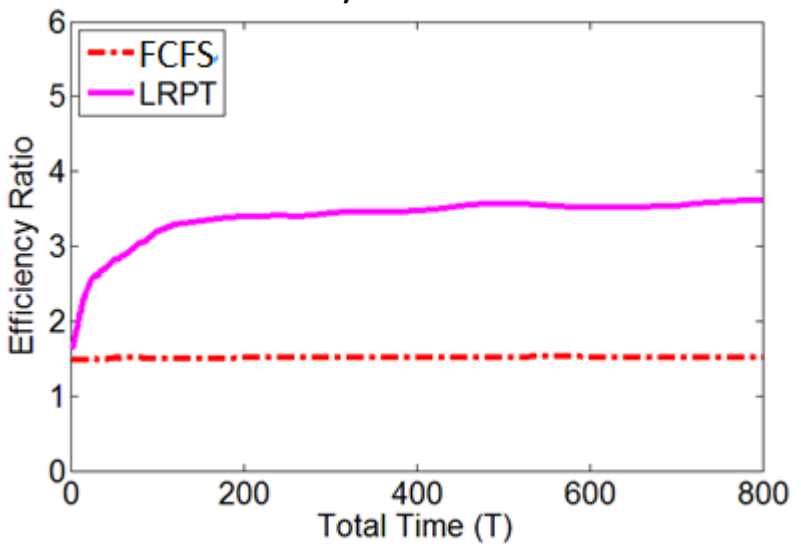
$$\gamma = \frac{B_2 + B_1^2}{\max\left\{2, \frac{1-p_0}{\lambda}, \frac{\rho}{\lambda}\right\} \max\left\{1, \frac{1}{N(1-\rho)}\right\}}$$

p_0 = Prob (a slot has no job arrivals)

$$Map : U[10, 50]$$

$$Reduce : U[10, 20]$$

N = 100 machines, total time slots T = 800



LRPT: Longest Remaining Processing Time

FCFS: First Come First Serve

Efficiency Ratio: Non-preemptive

For any work-conserving scheduling policy,

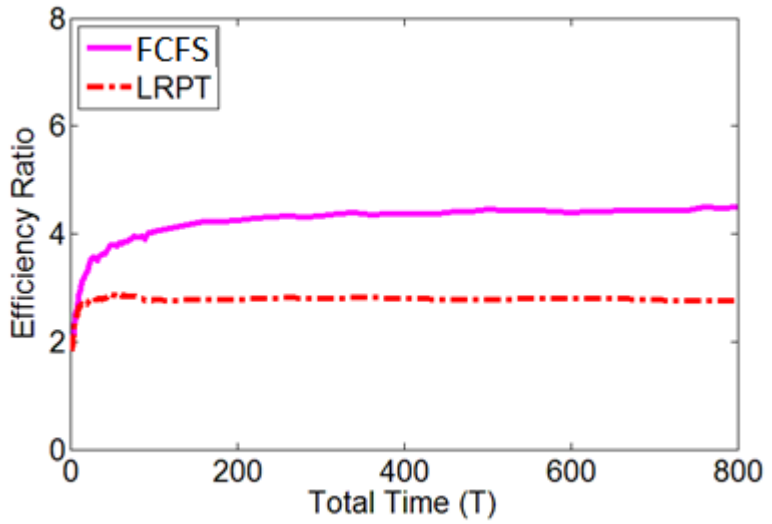
$$\gamma = \frac{B_2 + B_1^2 + B_1(\overline{R}-1)}{\max\left\{2, \frac{1-p_0}{\lambda}, \frac{\rho}{\lambda}\right\} \max\left\{1, \frac{1}{N(1-\rho)}\right\}}$$

p_0 = Prob (a slot has no job arrivals)

$Map : U[10, 50]$

$Reduce : U[10, 20]$

N = 100 machines, total time slots T = 800



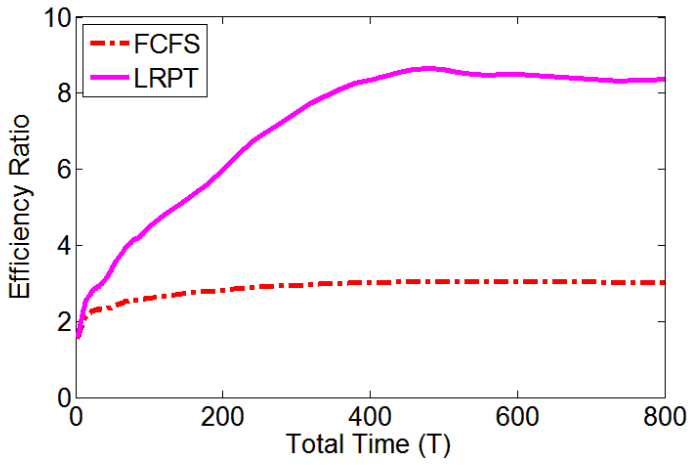
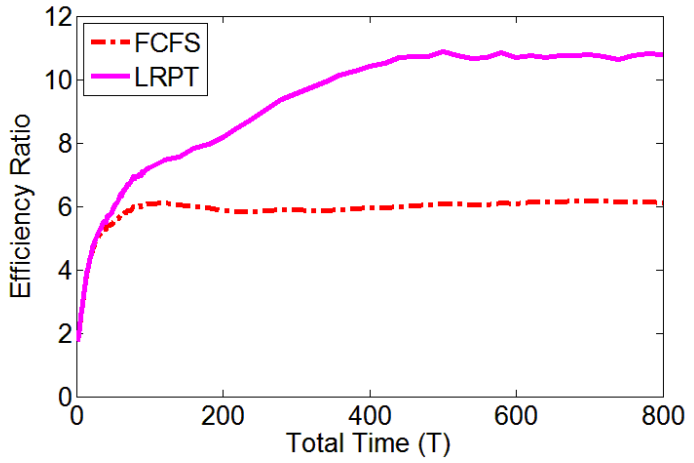
Light-tailed Distributed Workload of Reduce Phase

- $\exists r_0$ such that

$$P(R \geq r) \leq \alpha \exp(-\beta r), \ \forall r \geq r_0,$$

where α and β are constants.

- Similar result holds but with different constants (B_1' and B_2' instead of B_1 and B_2)

	Preemptive	Non-preemptive
$Map : Exp(5)$ $Reduce : Exp(20)$		

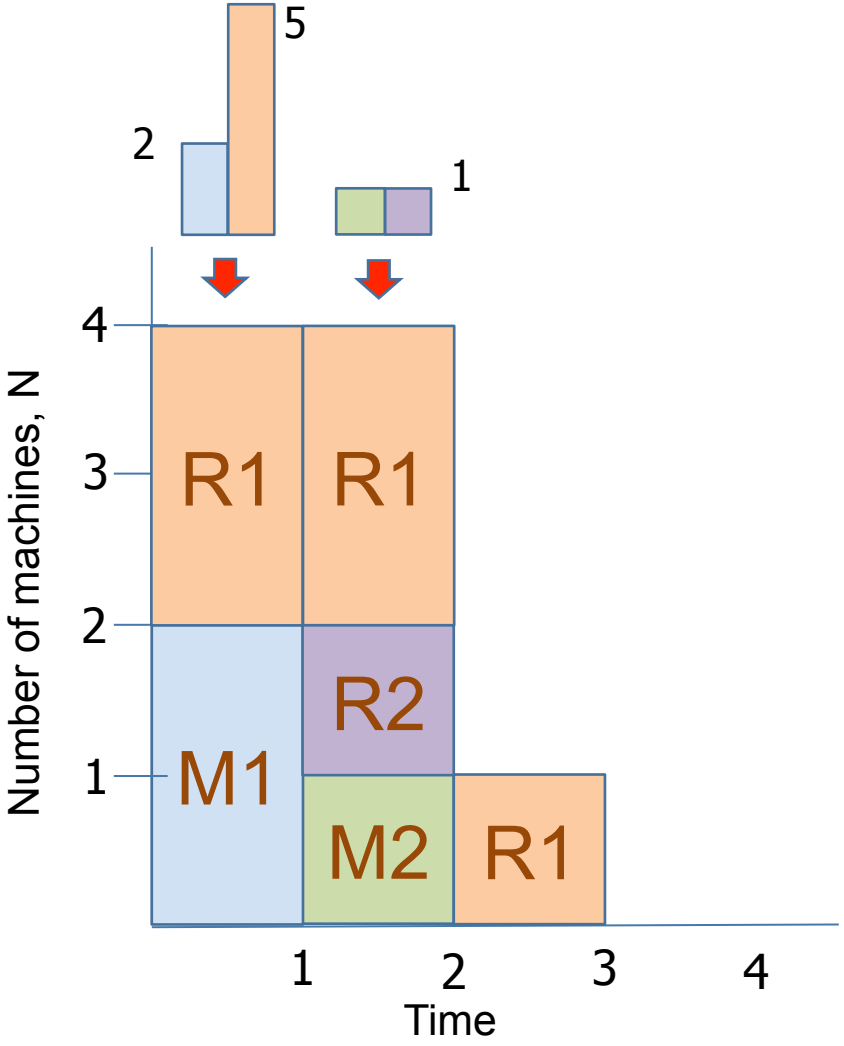
Thus Far

- There exist no schedulers that have a bounded competitive ratio in non-preemptive scenarios
- All work conserving schedulers have a finite efficiency ratio (γ) for both preemptive & non-preemptive scenarios
- But the performance of these schedulers could still be quite poor (γ could be very large)
- Key Question: Can we design simple provably efficient schedulers in the Map-Reduce paradigm?

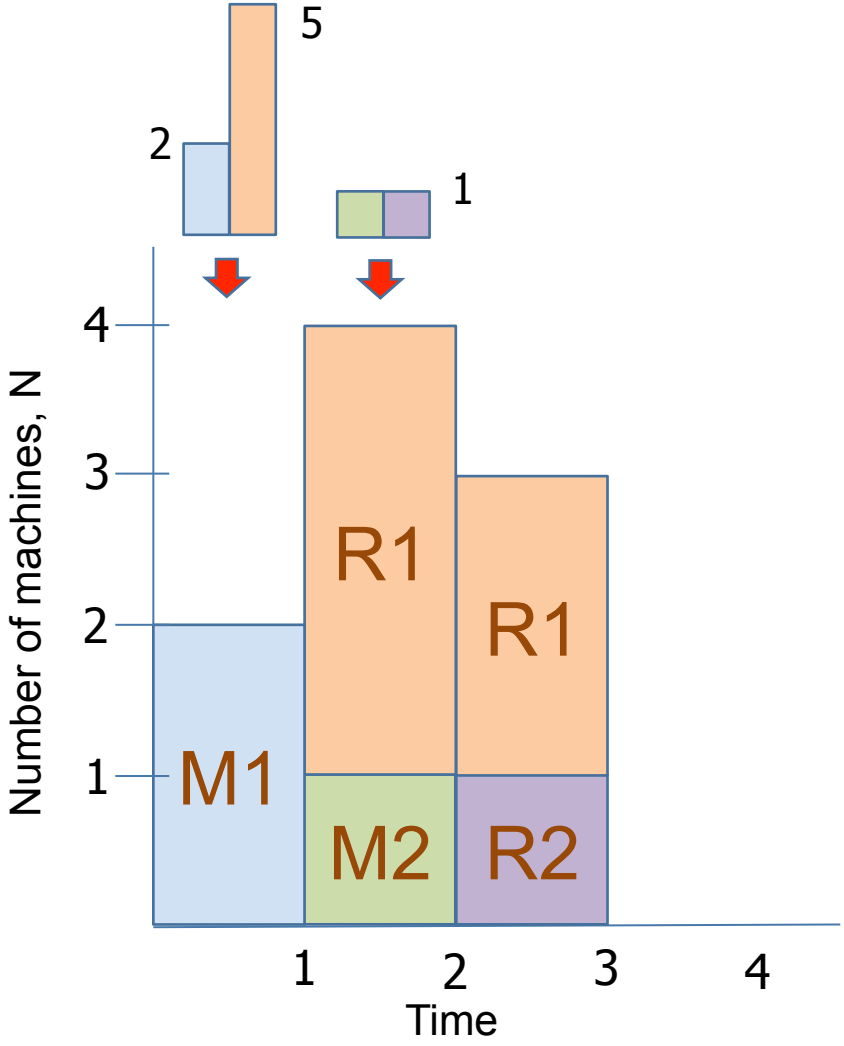
ASRPT: Available Shortest Remaining Processing Time [Quick Summary]

- ASRPT – A few key ideas
  - SRPT: An infeasible scheduler
    - Priority given to jobs with smallest remaining workload
    - In SRPT, Map and Reduce for a given job can be scheduled in same time-slot (infeasible in reality)
    - SRPT results in a lower flow time than any feasible (including non-causal) scheduler
  - ASRPT
    - Map tasks are scheduled according to SRPT (or sooner)
      - By running SRPT in a virtual manner
    - Reduce tasks greedily fill up the remaining machines
      - In the order determined by the shortest remaining processing time

ASRPT: Available Shortest Remaining Processing Time [Example]



SRPT: Total Flow-time 4 units



ASRPT: Total Flow-time 5 units

ASRPT: Available Shortest Remaining Processing Time [Main Result]

THE OHIO STATE UNIVERSITY

Theorem:

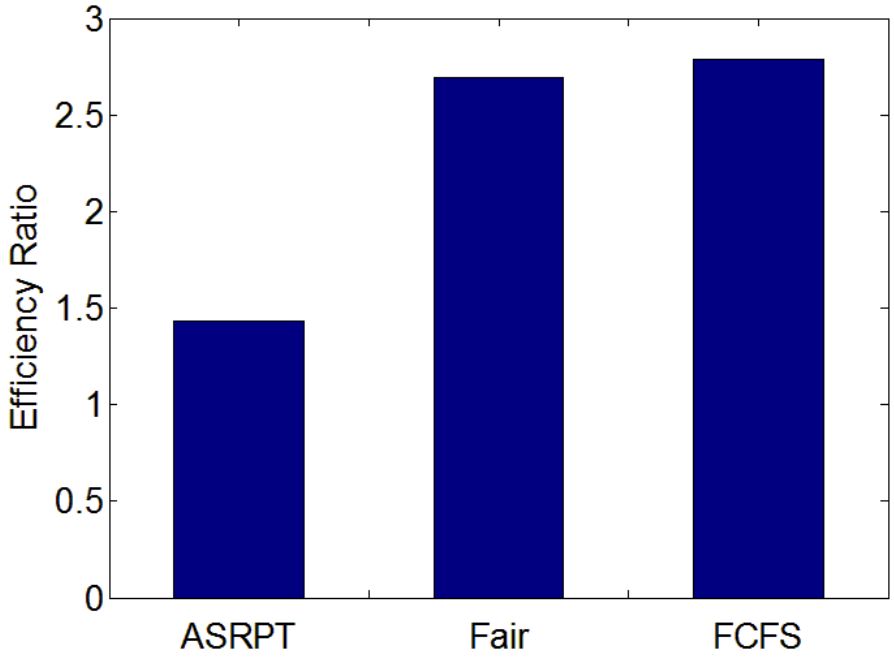
If the job arrivals and workload are i.i.d., then ASRPT has an efficiency ratio of 2 in the preemptive scenario.

Proof idea:

- SRPT is Infeasible in the MapReduce framework
- SRPT gives a lower bound on total flow time over all schedulers
- Since ASRPT is constructed based on SRPT, the efficiency ratio is obtained by comparing the performance of ASRPT to SRPT (the lower bound).

Simulation

- N = 100 machines, total time slots T = 800
- # Reduce tasks in each job is 10
- Job arrival process: Poisson; λ = 2 jobs per time slot.
- Preemptive Scenario $Map : Exp(5)$ $Reduce : Exp(20)$
- Schedulers
  - ASRPT
  - Fair (Facebook)
  - FCFS (Default in Hadoop)



Conclusion

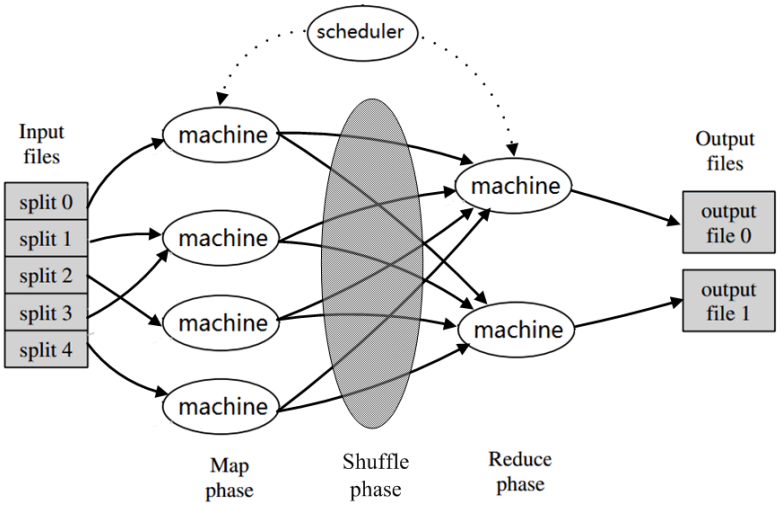
- Investigated the flow time minimization problem in MapReduce schedulers
- No online policy has a bounded competitive ratio in non-preemptive scenarios
- All work-conserving schedulers have a constant efficiency ratio in both preemptive and non-preemptive scenarios.
  - Bounded reduce workload
  - Light-tailed distributed reduce workload
- A specific algorithm (ASRPT) can guarantee an efficiency ratio of 2 in the preemptive scenario.
- Efficiency Ratio based analysis provides worse case (w.p.1) guarantees but gives more design flexibility
  - Relatively unexplored metric
  - Has the potential to be applied to other problems/domains

Many unanswered questions

	Competitive Ratio	Efficiency Ratio
Preemptive	?	Constant for any work-conserving scheduler
		2 for ASRPT
Non-preemptive	No constant for any scheduler	Constant for any work-conserving scheduler
		? for ASRPT

Ongoing/Future Work

- Consider cost of data migration
  - Combine preemptive and non-preemptive scenarios
- Consider Shuffle Phase
  - Introduce more information (e.g., dependency Graph) to the scheduler
  - So that all reduce tasks don't wait until Map tasks are completed



- Consider multiple phases
  - With phase precedence (for complex processing)
- Design simpler workload agnostic schedulers
  - Without detailed knowledge of Map or Reduce workload

Y. Zheng, P. Sinha, and N. B. Shroff, "A New Analytical Technique for Designing Provably Efficient MapReduce Schedulers," **IEEE INFOCOM'13**, April 2013, Turin, Italy.

Thank You!