A New Analytical Technique for Designing Provably Efficient MapReduce Schedulers

Authors: Yousi Zheng, Ness B. Shroff, Prasun Sinha

To appear in INFOCOM2013

Problem

- Minimize the total flow-time of a sequence of jobs in the MapReduce framework:
  - Map tasks generate intermediate data.
  - Reduce tasks operate on intermediate data to generate final results.
  - Make decisions on which task will be executed at what time and on which machine.

System Model

- N: machines in a data center; n: jobs in the system;
- M_i: workload for map tasks of job i; $\bar{M}$: Mi's expectation;
- R_i: workload for reduce tasks of job i; $\bar{R}$: Ri's expectation;
- $R_i^{(k)}$: workload of reduce task k of job i;
- K_i : number of job i's reduce tasks;
- $m_{i,t}$: machines allocated for job i's map tasks in time t;
- $r_{i,t}$: machines allocated for job i's reduce tasks in time t;
- $r_{i,t}^{(k)}$: machines allocated for reduce task k in time slot;
- a_i: arrival time of job i;
- $f_i^{(m)}$: the finishing time slot for the last map task of job i.
- $f_i^{(r)}$: the finishing time slot for all reduce tasks of job i.

Each machine run one unit of workload in a time slot.

Assumptions

- Each machine can only process one job at a time, and run one unit of workload in a time slot.
- The scheduler periodically collects the information on the state of job running on the machines.
- Traffic intensity $\rho < 1$;
- The distribution of job arrivals in each time slot is i.i.d., and the arrival rate is λ.
- $\{M_i\}$ and $\{R_i\}$ are i.i.d..

Flow-time minimization problem definition

- For preemptive scheduler:

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

The sum of flow time for each job.

$$s.t. \quad \sum_{i=1}^{n} (m_{i,t} + r_{i,t}) \leq N, \; r_{i,t} \geq 0, \; m_{i,t} \geq 0, \; \forall t,$$

The number of assigned machines must be less than or equal to the total number of machines.

$$\sum_{t=a_i}^{f_i^{(m)}} m_{i,t} = M_i, \quad \sum_{t=f_i^{(m)}+1}^{f_i^{(r)}} r_{i,t} = R_i, \; \forall i \in \{1, ..., n\}.$$

The workload of job i's map tasks should be processed by the assigned number of machines between time slot ai and fi(m).

The workload of job i's reduce tasks should be processed by the assigned number of machines between time slot fi(m) +1 and fi(r).

Flow-time minimization problem definition

- For non-preemptive scheduler:

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

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

$$\sum_{t=a_i}^{f_i^{(m)}} m_{i,t} = M_i, \ m_{i,t} \geq 0, \ \forall i \in \{1, ..., n\},$$

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

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

Additional constraints representing the non-preemptive nature.

Problem Complexity

- The scheduling problem (both preemptive and non-preemptive) is NP-complete in the strong sense.
- The proof of the NP-completeness follows a fairly standard reduction from 3-Partition. The detail is skipped here.

Problem Complexity

No constant competitive ratio for online algorithms:

- Scheduling algorithm S has a competitive ratio c if:

$$\frac{F^S(T, n, \{a_i, M_i, R_i; i = 1...n\})}{F^*(T, n, \{a_i, M_i, R_i; i = 1...n\})} \leq c$$

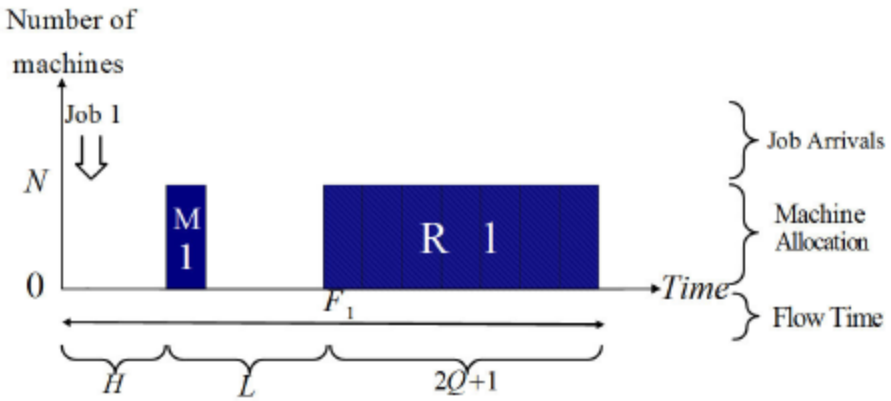
The proof of no constant competitive ratio:

- Given any scheduling algorithm S, any constant c0, we can construct an arrival pattern accordingly that makes the competitive ratio of S under the arrival pattern, c, is larger than c0.

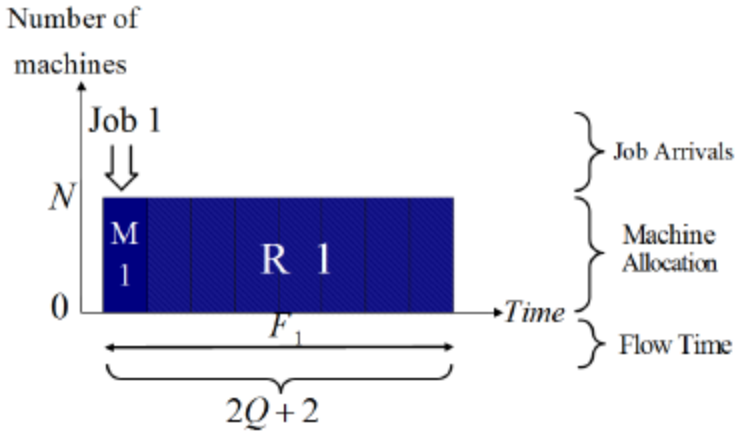
Proof of No Constant Competitive Ratio

- Example:
  - The data center has 1 machine.
  - Two types of arrival patterns:
    - a job with 1 map task and 1 reduce task, workload of Map and reduce tasks are 1 and 2Q+1, arriving in time slot 1.
    - a job as the first type, and Q additional jobs arrive in time slots H+L+2, H+L+4,…,H+L+2Q. Suppose Map task is scheduled in time slot H+1, Reduce task is scheduled in time slot H+L+1.
  - Divide scheduling algorithm S into two possibilities according how S schedules the first type of job arrival patterns.
    - Case 1: $H + L > 2(c_0 - 1)(Q + 1) + 1$
    - Case 2: $H + L \leq 2(c_0 - 1)(Q + 1) + 1$

Case 1: $H + L > 2(c_0 - 1)(Q + 1) + 1$



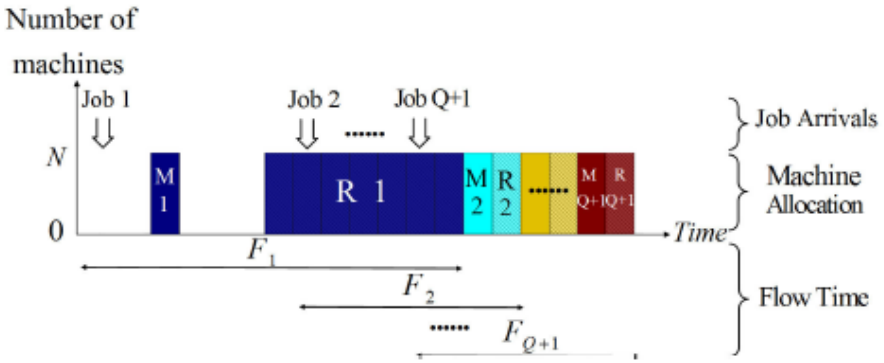
(a) Case I: Scheduler S



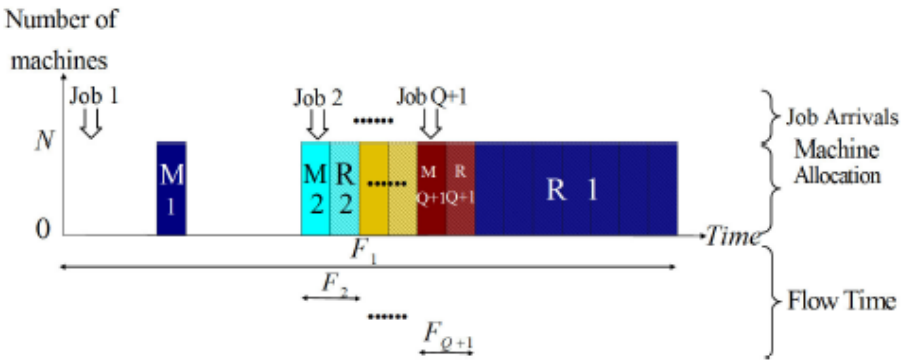
(b) Case I: Scheduler S_1

- Consider the scheduling of the first arrival pattern.
- The flow time of S is H+L+2Q+1
- Construct a new scheduling S1, the flow time of S1 is 2Q+2.
- The competitive ratio $c \geq \frac{H+L+2Q+1}{2Q+2} > c_0$.

Case 2: $H + L \leq 2(c_0 - 1)(Q + 1) + 1$



(c) Case II: Scheduler S



(d) Case II: Scheduler S_2

- Consider the scheduling of the second arrival pattern.
- The flow time of S is H+L+2Q+1+(2Q+1)Q; the flow time of S2 is 2Q+(H+L+1+2Q+2Q+1).
- The competitive ratio $c \geq \frac{H+L+2Q+1+(2Q+1)Q}{2Q+(H+L+1+2Q+2Q+1)} > \frac{Q}{c_0+2}$,by selecting $Q > c_0^2 + 2c_0, c > c_0$.

Efficiency Ratio Definition

Definition 2. *We say that the scheduling algorithm S has an efficiency ratio γ, if the total flow-time $F^S(T, n, \{a_i, M_i, R_i; i = 1...n\})$ of scheduling algorithm S satisfies the following:*

$$\lim_{T \to \infty} \frac{F^S(T, n, \{a_i, M_i, R_i; i = 1...n\})}{F^*(T, n, \{a_i, M_i, R_i; i = 1...n\})} \leq \gamma, \text{ with probability 1.}$$

Available Shortest-Remaining Processing Time(ASRPT) Algorithm and Analysis

- SRPT: Shortest Remaining Processing Time
  - SRPT picks up the job with the minimum total remaining workload to schedule.
  - SRPT assumes that Map and Reduce tasks from the same job can be scheduled simultaneously in the same slot.
  - SRPT may come up with an infeasible solution. It is used to obtain a lower bound on the total flow-time of MapReduce framework. And ASRPT is constructed based on SRPT.

Proof of SRPT achieving lower bound

- The flow-time minimization problem in the preemptive scenario under SRPT is:

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

$$s.t. \sum_{i=1}^{n} (m_{i,t} + r_{i,t}) \leq N,\ r_{i,t} \geq 0,\ m_{i,t} \geq 0,\ \forall t,$$

$$\sum_{t=a_i}^{f_i^{(r)}} (m_{i,t} + r_{i,t}) = M_i + R_i,\ \forall i \in \{1, ..., n\}.$$

This constraint is weaker.

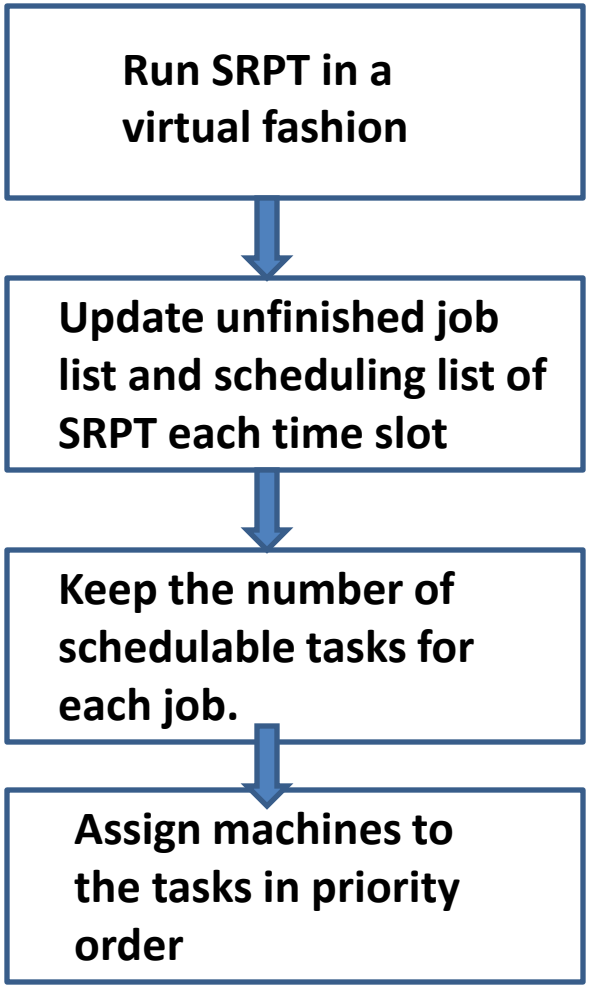
- The problem under requirement that reduce tasks can only be processed after map task:

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

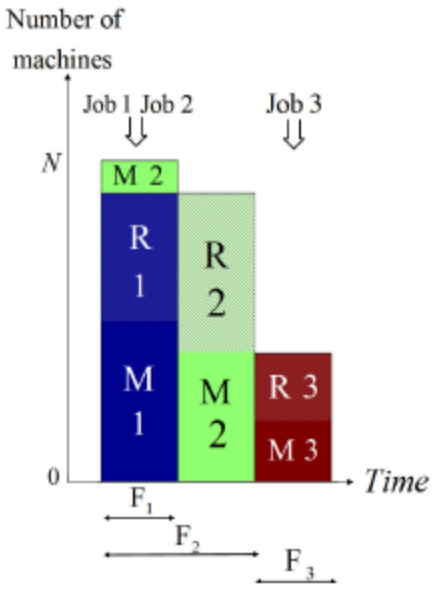
$$s.t. \sum_{i=1}^{n} (m_{i,t} + r_{i,t}) \leq N,\ r_{i,t} \geq 0,\ m_{i,t} \geq 0,\ \forall t,$$

$$\sum_{t=a_i}^{f_i^{(m)}} m_{i,t} = M_i,\ \sum_{t=f_i^{(m)}+1}^{f_i^{(r)}} r_{i,t} = R_i,\ \forall i \in \{1, ..., n\}.$$

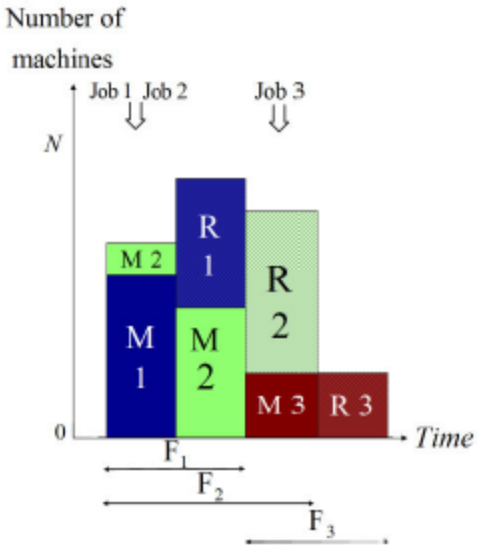
This constraint is stronger.



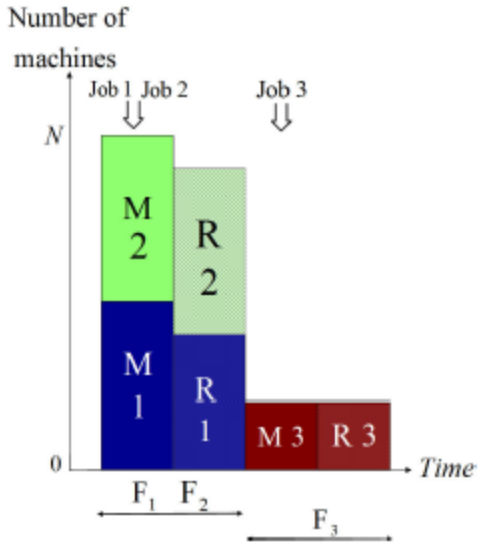
- Priority order:
  - Previously scheduled Reduce tasks which are not finished yet(only in the non-preemptive scenario)
  - Map tasks which are scheduled in the list S
  - available Reduce tasks
  - available Map tasks



(a) SRPT



(b) DSRPT



(c) ASRPT

- DSRTP (Delayed-Shortest-Remaining-Processing-Time):
  - Keep the scheduling method for the Map tasks exactly same as in SRPT.
  - Reduce tasks are scheduled in the same order from the next time slot compared to SRPT scheduling.

Efficiency Ratio Analysis of ASRPT Algorithm

- Theorem: In the preemptive scenario, DSRPT has an efficiency ratio 2.
- ASRPT is an improvement of DSRPT. As for each time slot, all the Map tasks finished by DSRPT are also finished by ASRPT. For the Reduce tasks of each job, the jobs with smaller remaining workload will be finished earlier than the jobs with larger remaining workload.
- Hence, the efficiency ratio of DSRPT also holds for ASRPT.

Efficiency Ratio Analysis of DSRPT Algorithm

- The difference among flow time between SRPT and DSRPT is due to the delay of reduce tasks in DSRPT.
- The reduce tasks in DSRPT can be modeled as a queueing system:
  - arrival in time slot $t \geq 2$, R_{t-1}^{S},
  - Service capacity of the reduce tasks: $N - M_t^S$.
  - The largest delay time for reduce tasks which are scheduled in time slot t-1 in SRPT: D_t.
  - Calculate the expectation of D_t: $E[D_t] \leq 2$.

- Then, we have

$$\lim_{T\to\infty}\frac{F_D}{F_S} \le \frac{\lim_{n\to\infty}\frac{F_S}{n} + E[\mathbb{D}]}{\lim_{n\to\infty}\frac{F_S}{n}} \text{ w.p.1}$$

$$=1+\frac{E[\mathbb{D}]}{\lim_{n\to\infty}\frac{F_S}{n}} \le 1+E[\mathbb{D}] \le 3.$$

and

$$\lim_{T\to\infty}\frac{F_D}{F} \le \frac{\lim_{n\to\infty}\frac{F_S}{n} + E[\mathbb{D}]}{\lim_{n\to\infty}\frac{F}{n}} \text{ w.p.1}$$

$$\le 1+\frac{E[\mathbb{D}]}{\lim_{n\to\infty}\frac{F}{n}} \le 1+\frac{E[\mathbb{D}]}{2} \le 2.$$

Summary

- The paper defines a new performance metric for the online algorithms, using this new metric to evaluate its scheduling algorithm.
- The scheduling algorithm is designed based on a popular algorithm: SRPT.

Construct ASRPT based on SRPT

1. Run SRPT in a virtual fashion and keeps track of how SRPT have scheduled jobs in any given slot.
2. Update J(list of unfinished jobs), S(scheduling list of SRPT) in the beginning of each slot.
3. Keep the number of schedulable tasks in current time slot.

   (1) a job that has unscheduled Map tasks:

   Schedulable tasks = unfinished Map tasks.

   available workload = units of unfinished workload of both Map and Reduce workload;

   (2) A job has no unscheduled Map tasks:

   Schedulable tasks = unfinished workload of the Reduce tasks(preemptive scenario) or the number of unfinished Reduce tasks(non-preemptive scenario)

   Available workload = unfinished reduce workload;

4. The algorithm assigns machines to the tasks in the priority order:
   1. Previously scheduled Reduce tasks which are not finished yet(only in the non-preemptive scenario) > Map tasks which are scheduled in the list S > available Reduce tasks > available Map tasks.