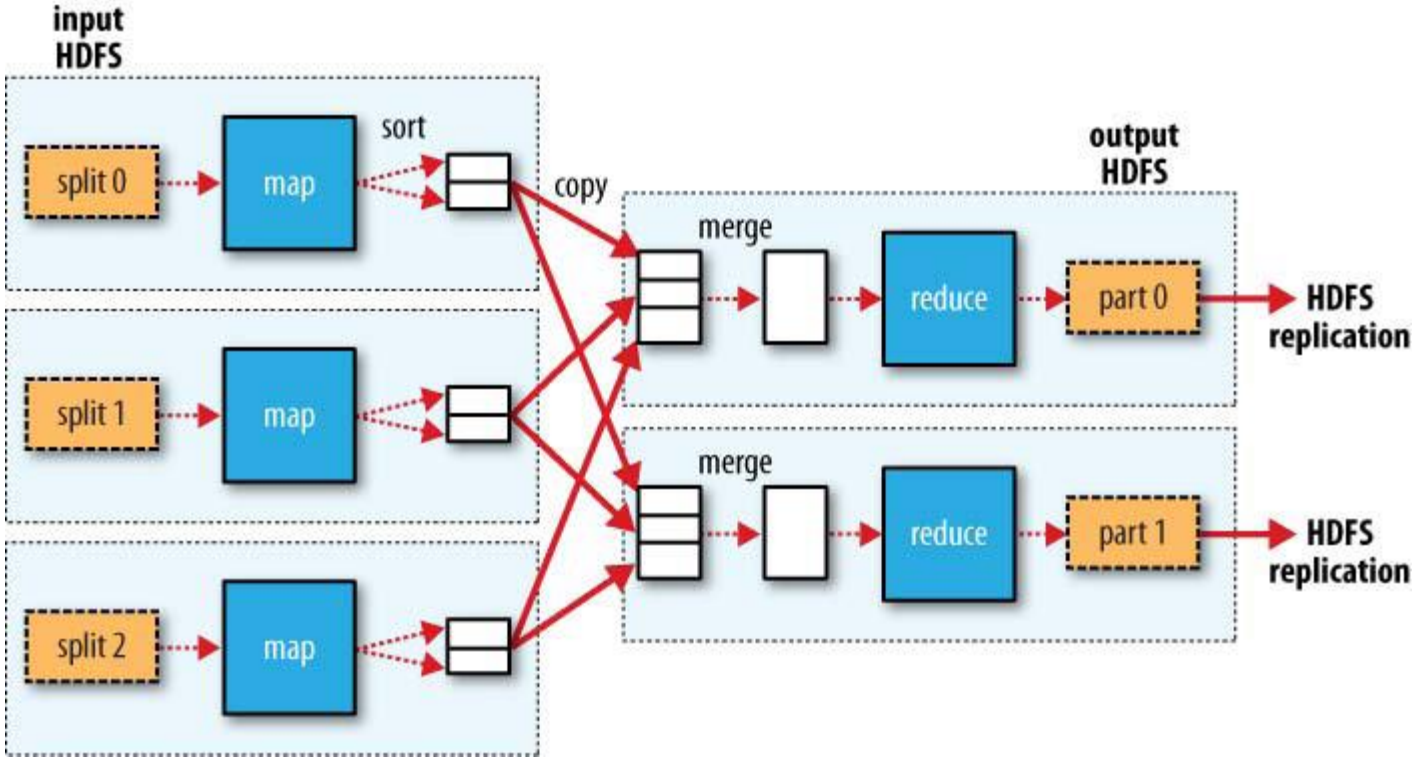
Big Data Research Lab
Computer Science Department
Colorado State University

Improving MapReduce Performance in Heterogeneous Environments

Feb 13, 2015

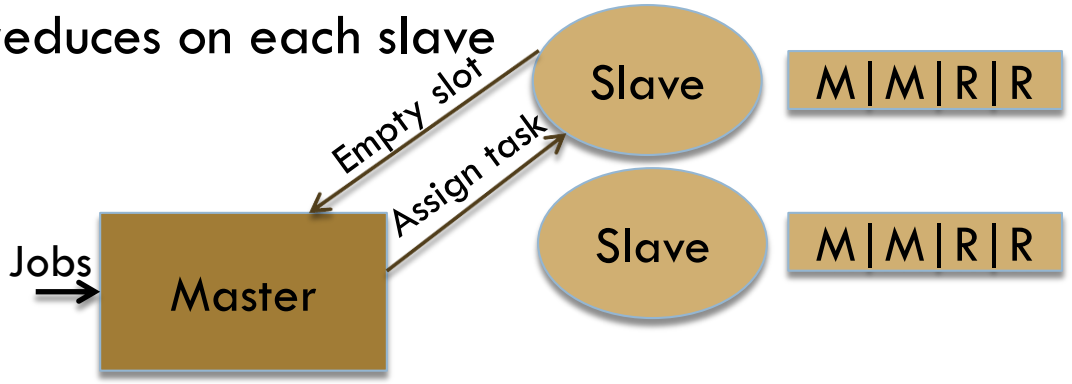
Tasks execution in hadoop

Map (k1, v1) → list (K2, [v2])



Hadoop runs several map and reduce tasks concurrently on each slave

- In hadoop there are one master and a number of slaves
- The cluster is divided into fixed number of slots
  - By default, 2 maps and 2 reduces on each slave



- Assignment order of tasks is as following:
  - Failed tasks have the highest priority
  - Non-running tasks (locality first)
  - Speculative tasks

Speculative task?

- Extra copy of the task running on straggler will be lunched on different machine
- Straggler is a poorly performing node
- The execution of speculative tasks reduces the job execution time
  - Google noticed 44% improvement of job response times

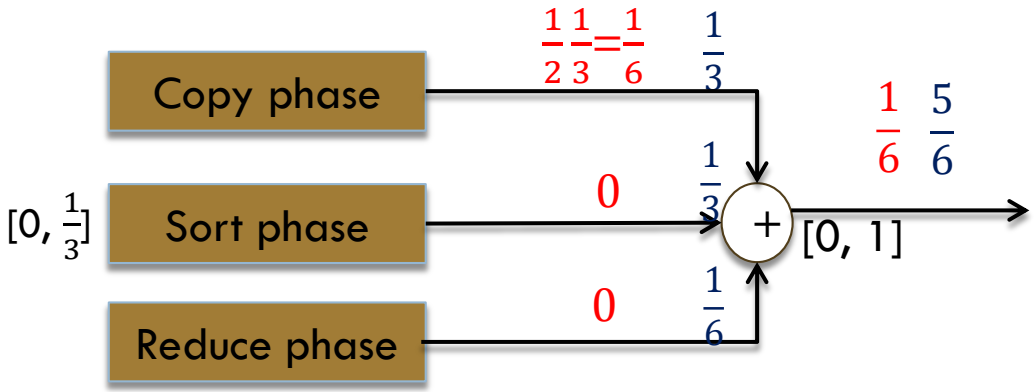
How does hadoop identify the speculative tasks?

- Hadoop uses progress scores to find the slow tasks
  - A value between 0 and 1
- For each task that is running for more than one minute
  - Compute the progress score s
  - If s < scoreAverage – 0.2 then
    - The task is a speculative task

scoreAverage is the average of progress scores of all tasks of of the same task's type (map, reduce)

How the progress scores are computed?

- Progress score of map task is a fraction of read input data
- Progress score of reduce task is the summation of progress scores of its 3 phases



Example of progress score:

- Task halfway through copy phase
- Task halfway through reduce phase

Progress score: the fraction of processed data

This works fine in homogeneous environments, but not in heterogeneous

- The assumptions that are made by hadoop and impact its performance in heterogeneous environment:
  - The progress rate at each node is the same
  - All tasks progress at constant rate
  - Launching speculative tasks on idle slots does not cause extra costs
  - The progress at different reduce phases is considered the same
  - Tasks of the same type require the same amount of work

Hadoop will perform poorly if the assumptions do not hold

- In some cases hadoop's performance will be better when the execution of speculative tasks is disabled
- For example:
  - Yahoo disables speculative execution on some jobs
  - Facebook disables speculation for reduce tasks

The proposed approach

- LATE (Longest Approximate Time to End) scheduler is proposed to schedule speculative tasks
  - To reduce the response time of a map-reduce job
- It schedules only the tasks that hurt the job's response time
- It limits the number of concurrent running speculative tasks
  - So that the execution of other tasks will not be affected

How does LATE find the speculative tasks?

- Tasks that have the highest impact on the job's response time will be scheduled first
  - That is why scheduling is done based on the remaining time

$$\text{remaining}Time = \frac{1 - progressScore}{progressRate},$$

$$progressRate = \frac{progressScore}{T}$$

Where T is the amount of time the task is running for

Speculative tasks spend the longest time to finish

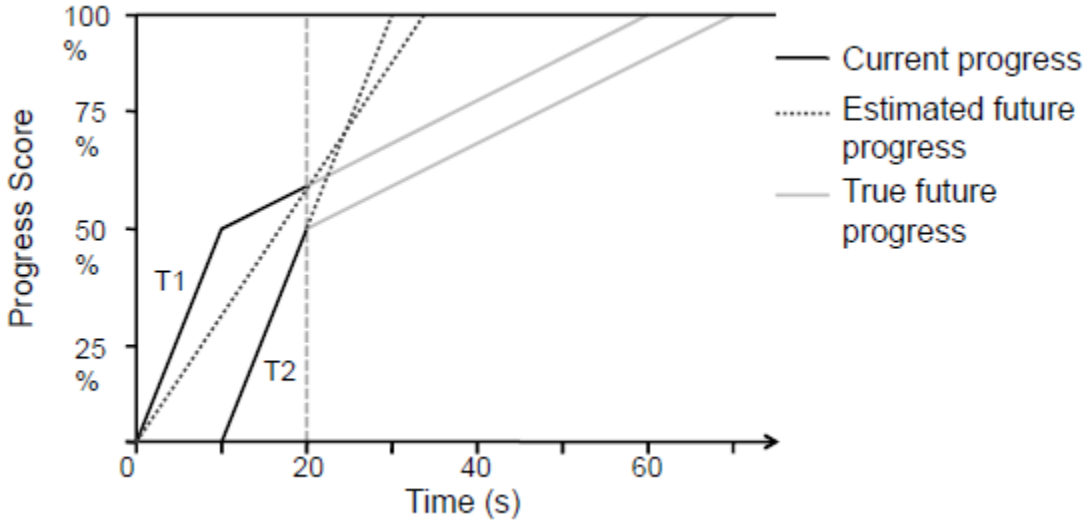
- All tasks will be sorted based on their remaining time
- Task having the longest remaining time will be executed first
  - Tasks that hurt the job's response time at most will be executed first

LATE launches only slow tasks on fast node

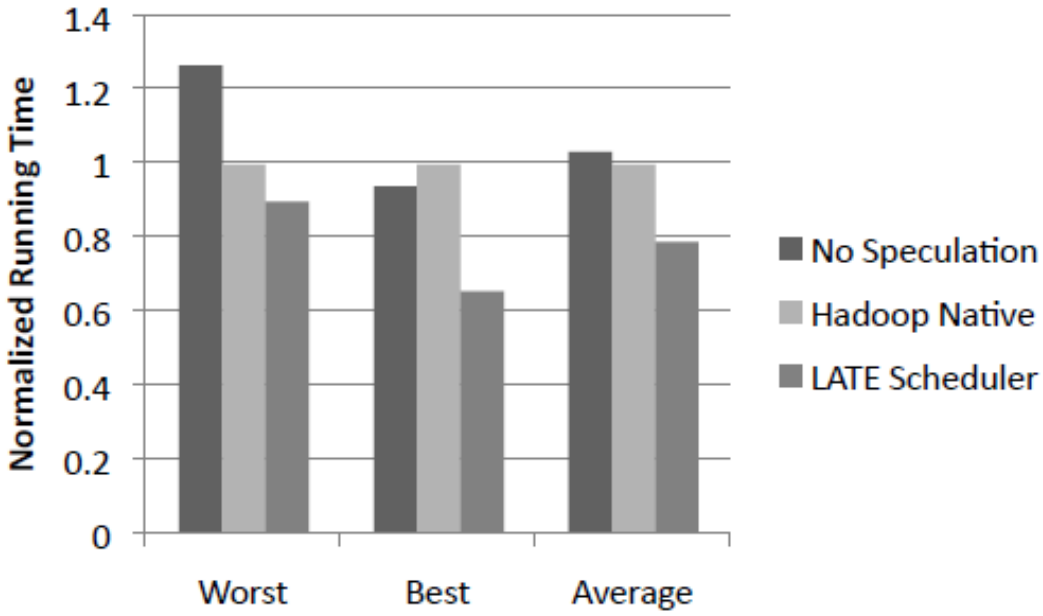
- Algorithm
  - When a node has idle task, it requests new to execute
  - LATE ignores requests coming from nodes whose total progress is below SlowNodeThreshold (25^{th} percentile of node progress)
  - LATE launches a copy of the task that has the longest remaining time and is slow
    - Its progress rate < slowTaskThreshold
- Here, data locality is not considered

The estimated finished time can be incorrect

- Because LATE assumes that tasks progress at constant rate, the estimated finished time can be incorrect



Running time of sort job on heterogeneous cluster



Conclusion

- LATE can reduce the job's response time by execute limited number of slow tasks on fast machines
- Assuming constant progress rate of the tasks can lead to incorrect estimate of finished time

Questions?