UNIVERSITY OF CAMBRIDGE

Cloud Computing
MapReduce in Heterogeneous Environments

Eva Kalyvianaki

ek264@cam.ac.uk

Contents

- Looking at MapReduce performance in heterogeneous clusters
- Material is from the paper:

  "Improving MapReduce Performance in Heterogeneous Environments",

  By Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz and Ion Stoica, published in Usenix OSDI conference, 2008
- and their presentation at OSDI

Motivation: MapReduce is becoming popular

- Open-source implementation, Hadoop, used by Yahoo!, Facebook, Last.fm, …
- Scale: 20 PB/day at Google, O(10,000) nodes at Yahoo, 3000 jobs/day at Facebook

Stragglers in MapReduce

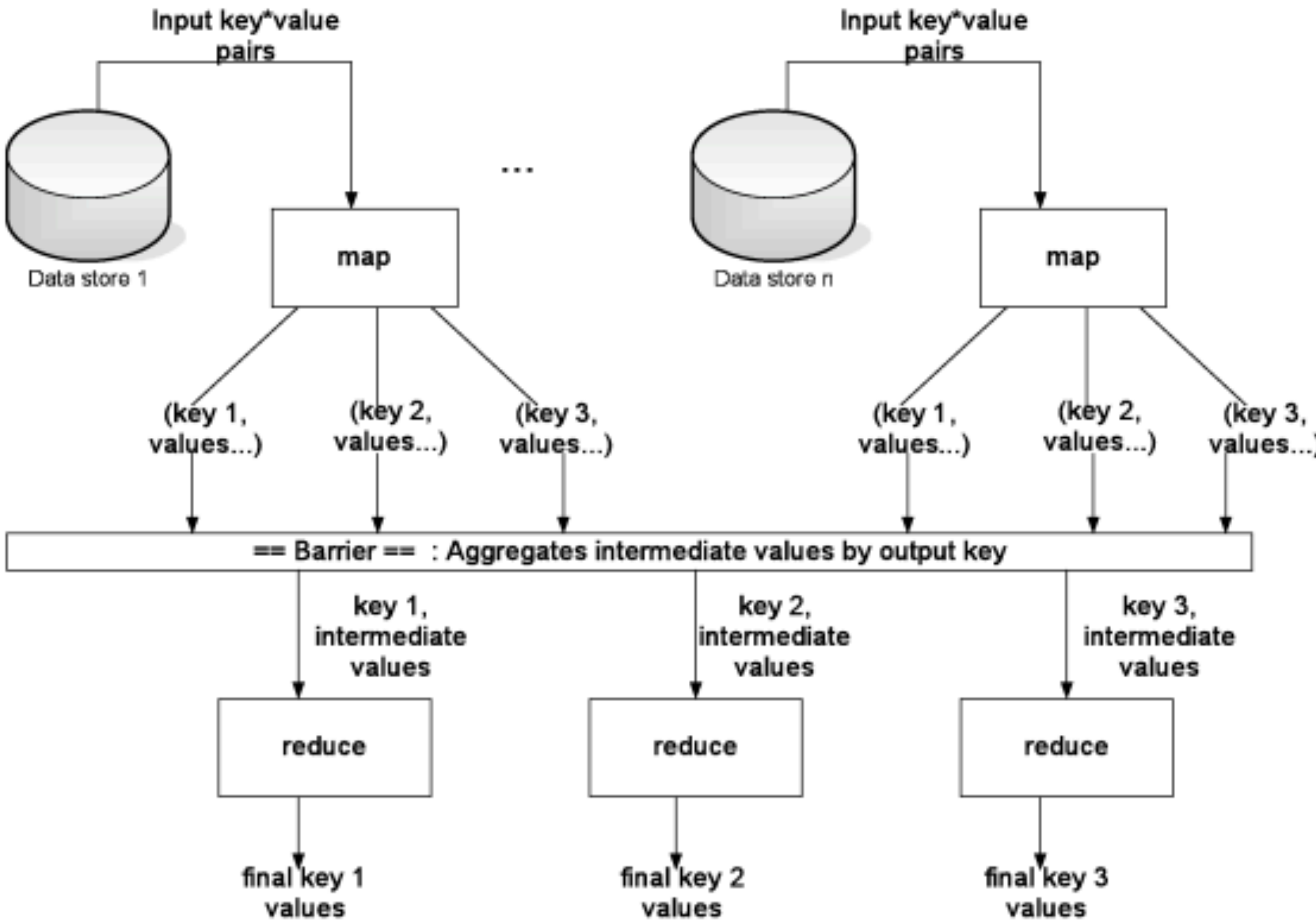
- Straggler is a node that performs poorly or not performing at all.
- Original MapReduce mitigation approach was:
  - To run a speculative copy (called a backup task)
  - Whichever copy or original would finish first would be included
- Without speculative execution, a job would be slow as the slowest sub-task
- Google notes that speculative execution can improve job response times by 44%
- Is this approach good enough for modern clusters?

Modern Clusters: Heterogeneity is the norm

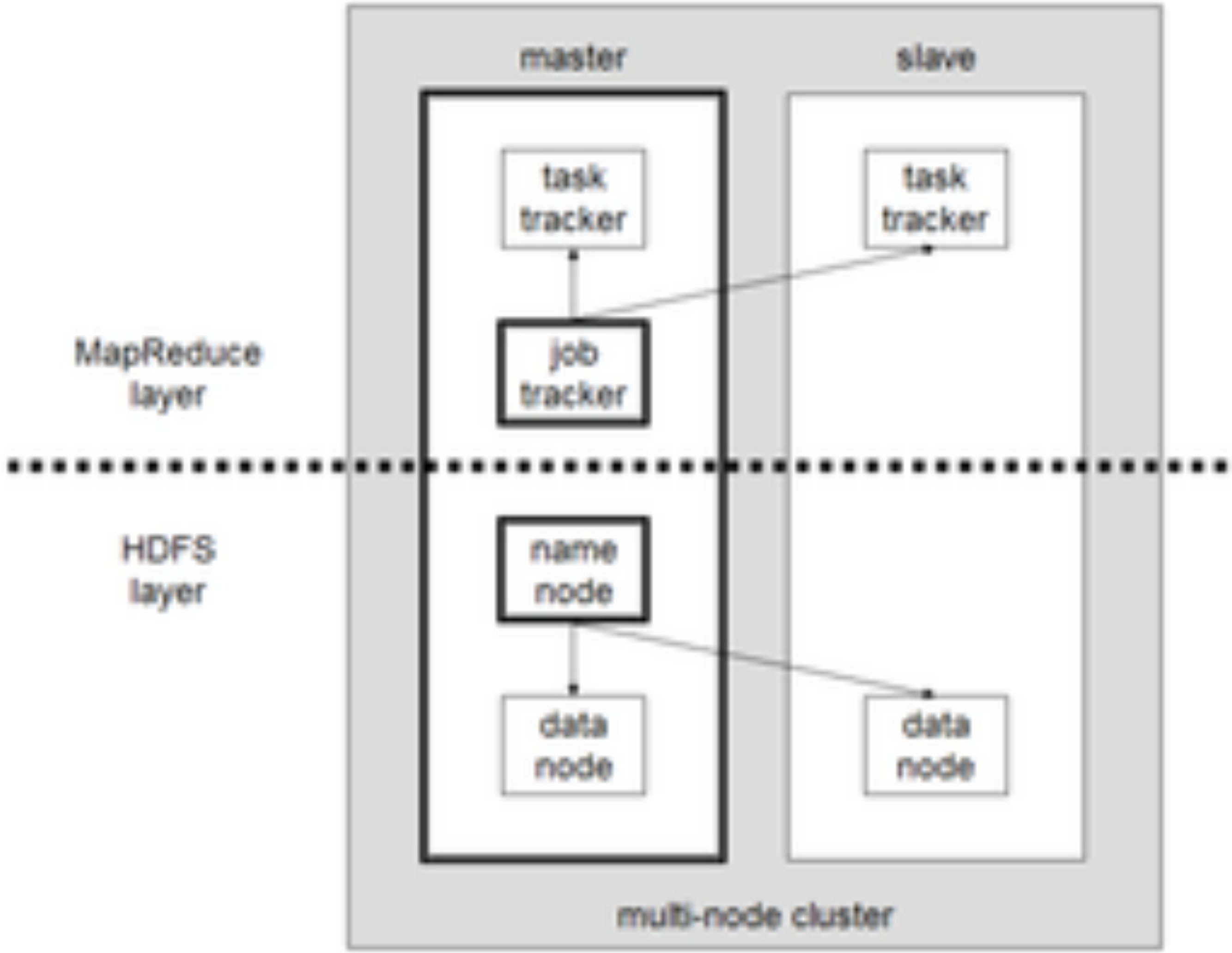
- Cloud computing providers like Amazon's Elastic Compute Cloud (EC2) provide cheap on-demand computing:
  - Price: 2 cents / VM / hour
  - Scale: thousands of VMs
  - Caveat: less control of performance

- Main challenge for Hadoop on EC2 is performance heterogeneity, which breaks task scheduler assumptions
- This lecture/paper is on a new LATE scheduler that can cut response time in half

MapReduce Revised



MapReduce Implementation, Hadoop



Scheduling in MapReduce

- When a node has an empty slot, Hadoop chooses one from the three categories in the following priority:
  1. A failed task is given higher priority
  2. Unscheduled tasks. For maps, tasks with local data to the node are chosen first.
  3. Looks to run a speculative task.

Deciding on Speculative Tasks

- Which task to execute speculatively?
- Hadoop monitors tasks progress using a *progress score*: a number from 0, …, 1
- For mappers: the score is the fraction of input data read
- For reducers: the execution is divided into three equal phases, 1/3 of the score each:
  - Copy phase: percent of maps that output has been copied from
  - Sort phase: map outputs are sorted by key: percent of data merged
  - Reduce phase: percent of data passed through the reduce function
- Example: a task halfway through the copy phase has progress score = 1/2*1/3 = 1/6.
- Example: a task halfway through the reduce phase has progress score = 1/3 + 1/3 + 1/2 * 1/3 = 5/6

Deciding on Speculative Tasks (con't)

- Hadoop looks at the average progress of each category of maps and reduces and defines a **threshold:**
- **When a task's progress is less than the average for its category minus 0.2, and the task has run at least one minute, it is marked as a straggler:**

threshold = avgProgress – 0.2

- All tasks with **progress score < threshold** are stragglers

- Ties are broken by data locality
- This approach works reasonably well in homogeneous clusters

Scheduler's Assumptions

1. Nodes can perform work at roughly the same rate
2. Tasks progress at constant rate all the time
3. There is no cost to starting a speculative task
4. A task's progress is roughly equal to the fraction of its total work
5. Tasks tend to finish in waves, so a task with a low progress score is likely a slow task
6. Different task of the same category (maps or reduces) take roughly the same amount of work

Revising Scheduler's Assumptions

(1) Nodes can perform work at roughly the same rate

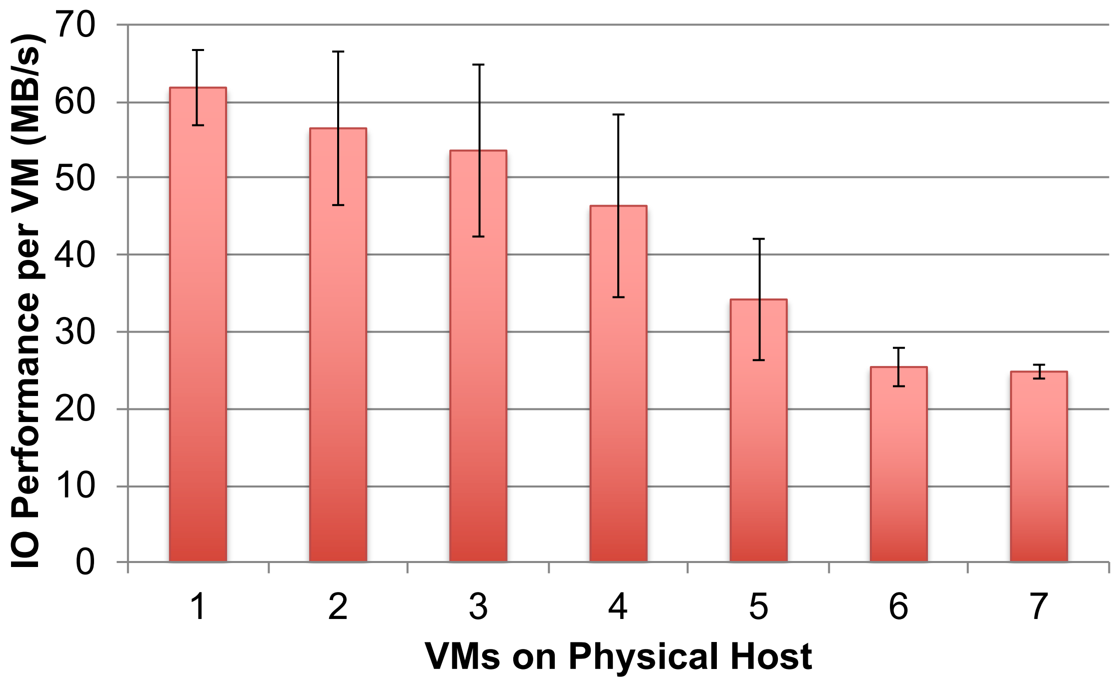
In heterogeneous clusters some nodes are slower (older) than others

(2) Tasks progress at constant rate all the time

Virtualized clusters "suffer" from co-location interference

Heterogeneity in Virtualized Environments

- VM technology isolates CPU and memory, but disk and network are shared
  - Full bandwidth when no contention
  - Equal shares when there is contention
- Timed a `dd` command that wrote 5000 MB of zeroes from `/dev/zero` to a file in parallel on 871 virtual machines in EC2's production cluster.
- **2.5x** performance difference



Revising Scheduler's Assumptions

(3) There is no cost to starting a speculative task

Too many speculative tasks can take away resources from other running tasks.

(4) A task's progress is roughly equal to the fraction of its total work

The copy phase of reducers is the slowest part, because it involves all-pairs communications. But this phase counts for 1/3 of the total reduce work.

(5) Tasks tend to finish in waves, so a task with a low progress score is likely a slow task

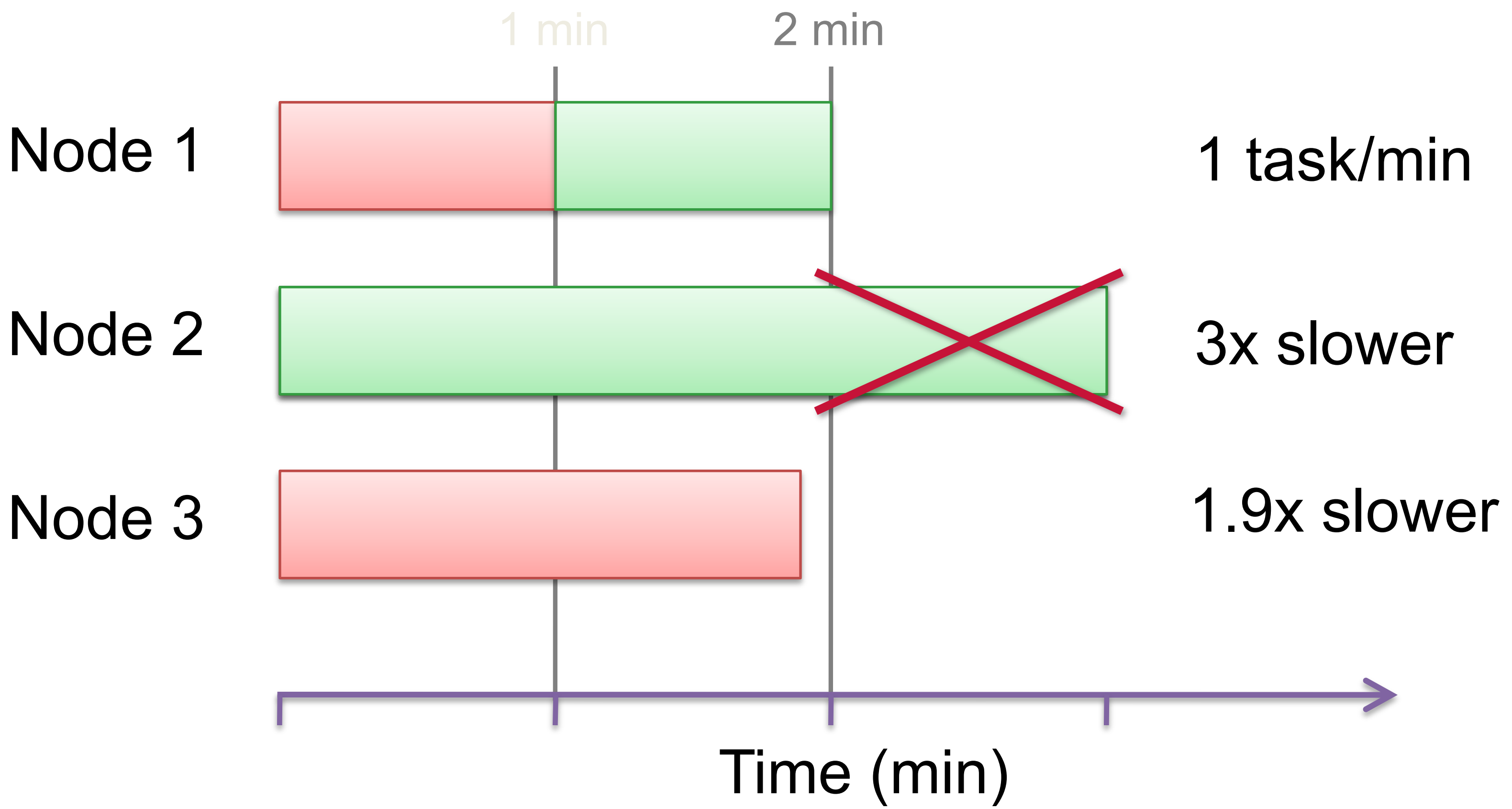
Tasks from different generations will be executed concurrently. So newer faster tasks are considered with older show tasks, avgProgress changes a lot.

Idea: Progress Rates

- Instead of using **progress score values**, compute **progress rates**
- Back up tasks with low **progress rate** that are "far enough" below the mean

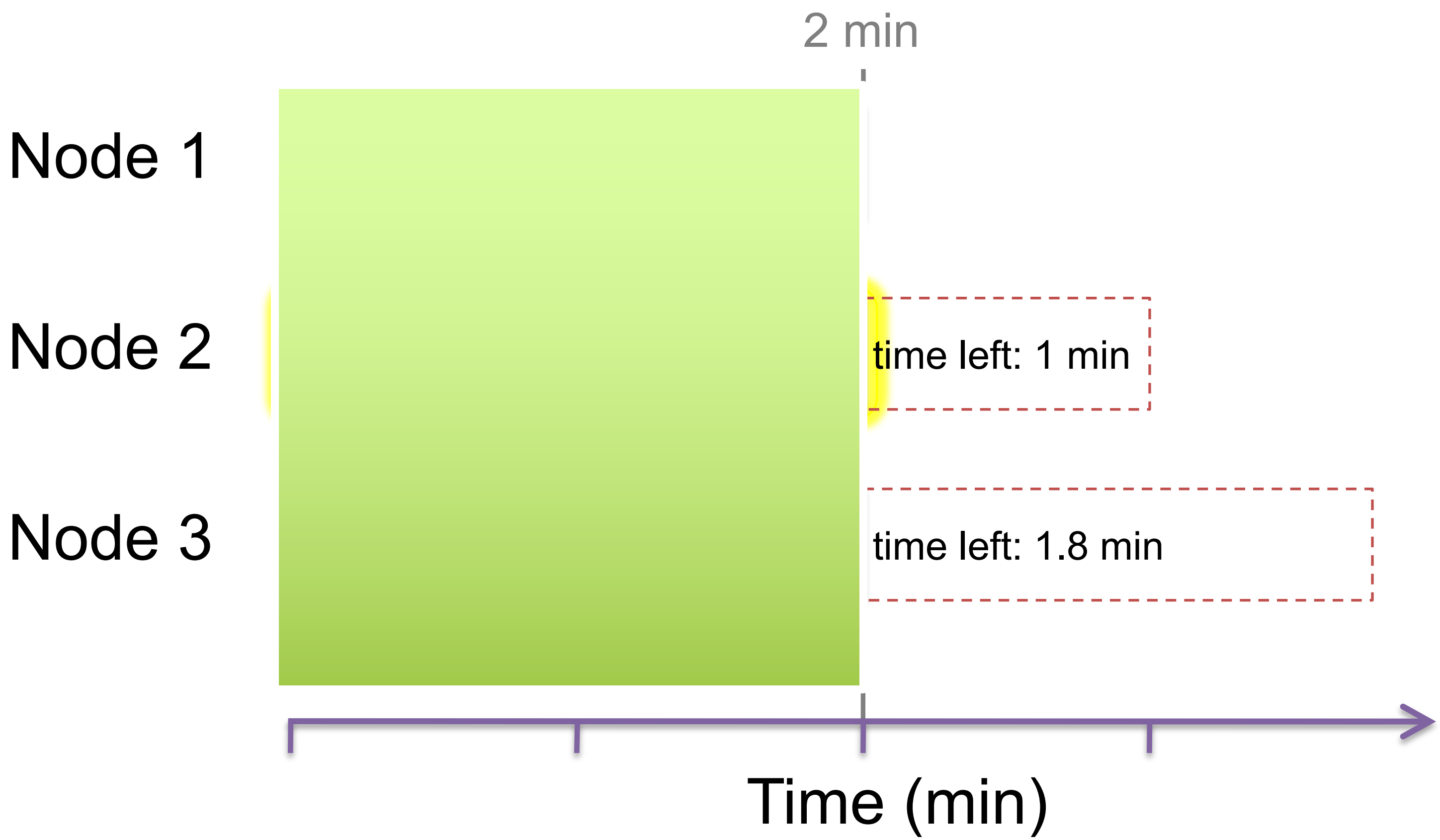
$$\textit{progress rate} = \frac{\text{progress score}}{\text{execution time}}$$

Progress Rate Example



Progress Rate Example

What if the job had 5 tasks?



Node 2 is slowest, but should back up Node 3's task!

LATE Details

- LATE: **Longest Approximate Time to End**
  - back up the task with the largest estimated finish time
  - look **forward** instead of looking **backward**

$$\textit{progress rate} = \frac{\text{progress score}}{\text{execution time}}$$

$$\textit{estimated time left} = \frac{1 - \text{progress score}}{\text{progress rate}}$$

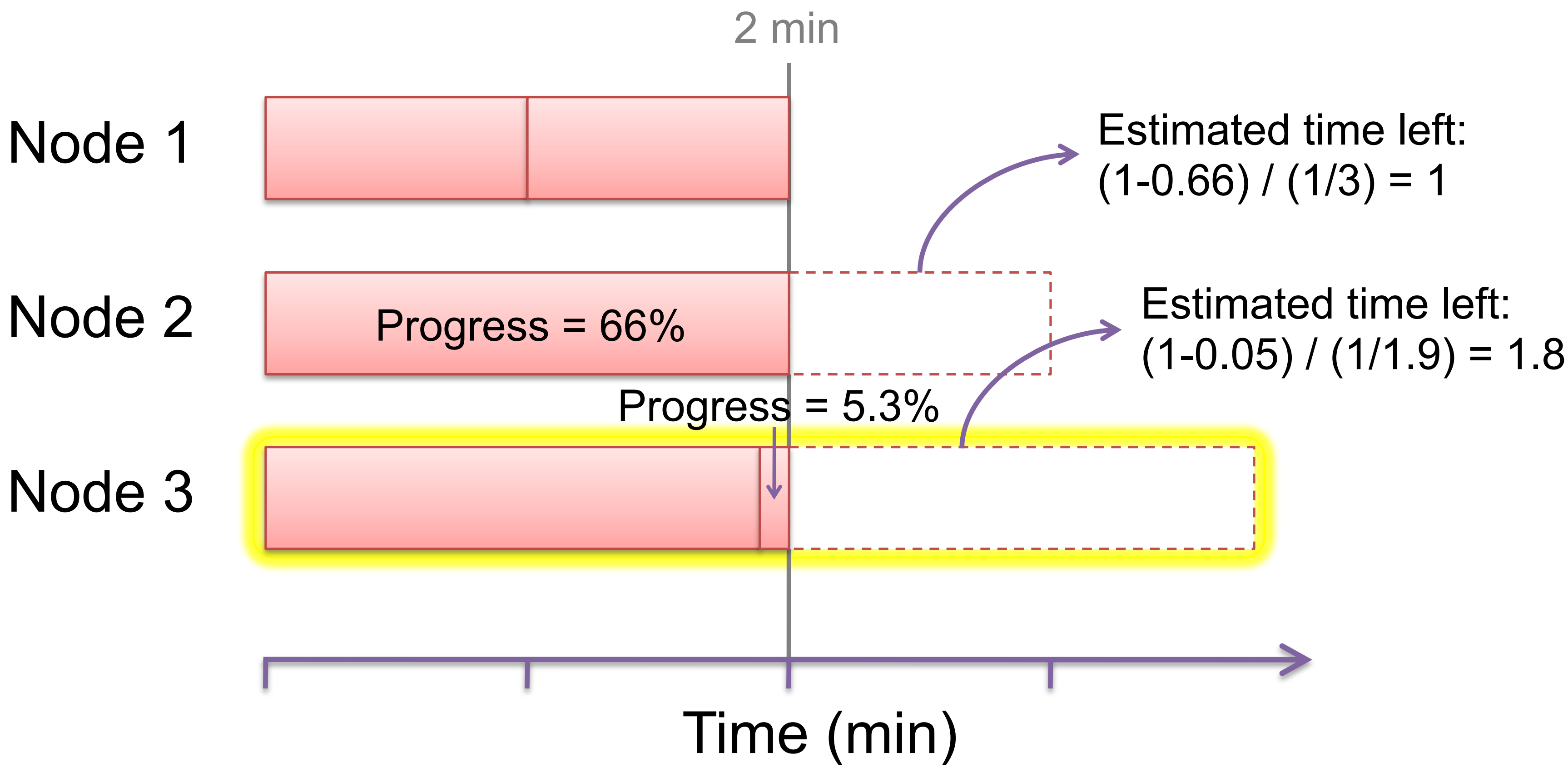
- Sanity thresholds:
  - Cap number of backup tasks
  - Launch backups on fast nodes
  - Only back up tasks that are sufficiently slow

LATE Scheduler

- If a task slot becomes available and there are less than *SpeculativeCap* tasks running, then:
  1. Ignore the request if the node's total progress is below *SlowNodeThreshold (=25th percentile)*
  2. Rank currently running, non-speculatively executed tasks by estimated time left
  3. Launch a copy of the highest-ranked task with progress rate below *SlowTaskThreshold (=25th percentile)*

- Threshold values:
  - 10% cap on backups, 25th percentiles for slow node/task
  - Validated by sensitivity analysis

LATE Example



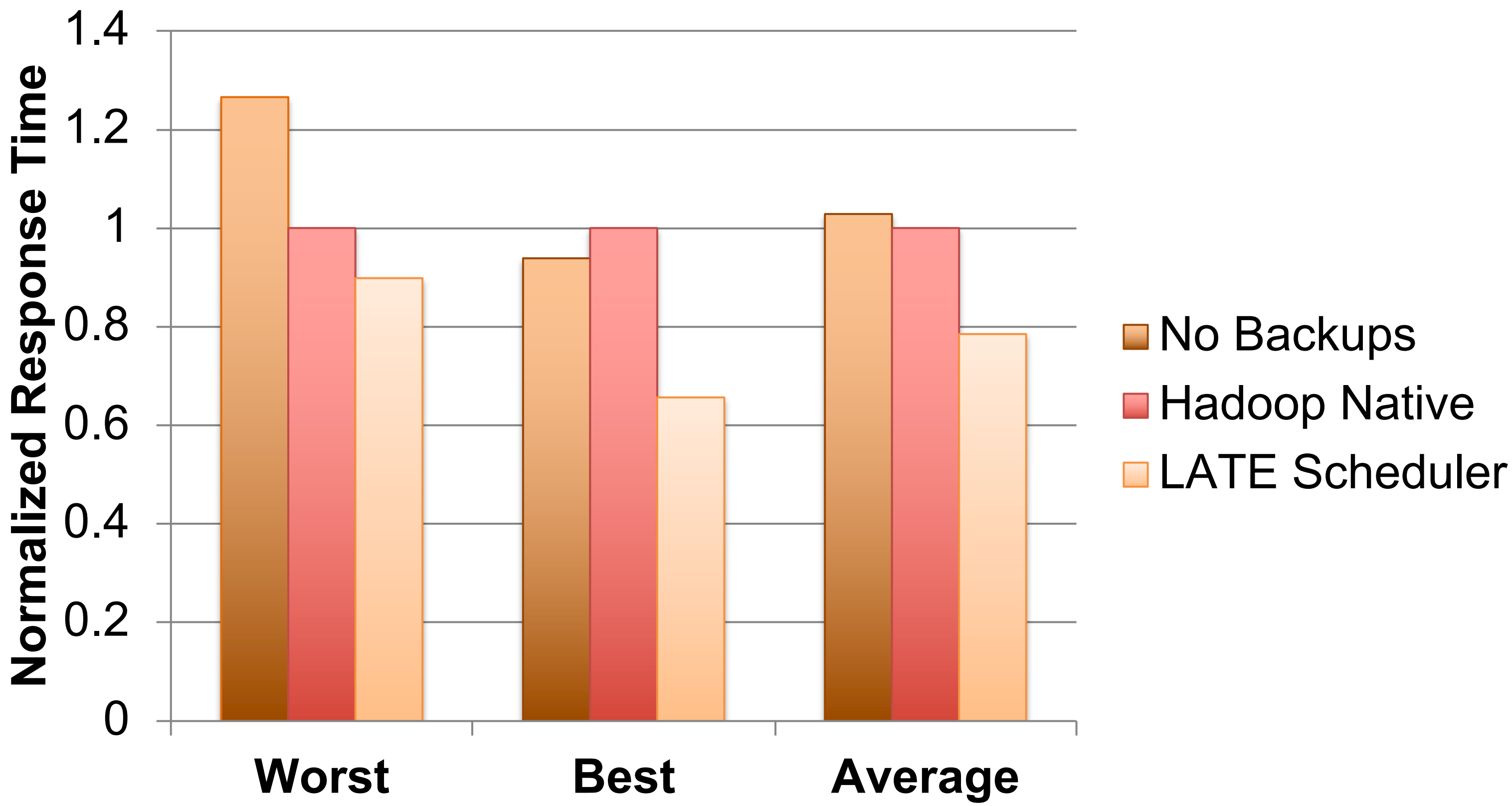
LATE correctly picks Node 3

Evaluation

- Environments:
  - EC2 (3 job types, 200-250 nodes)
  - Small local testbed
- Self-contention through VM placement
- Stragglers through background processes

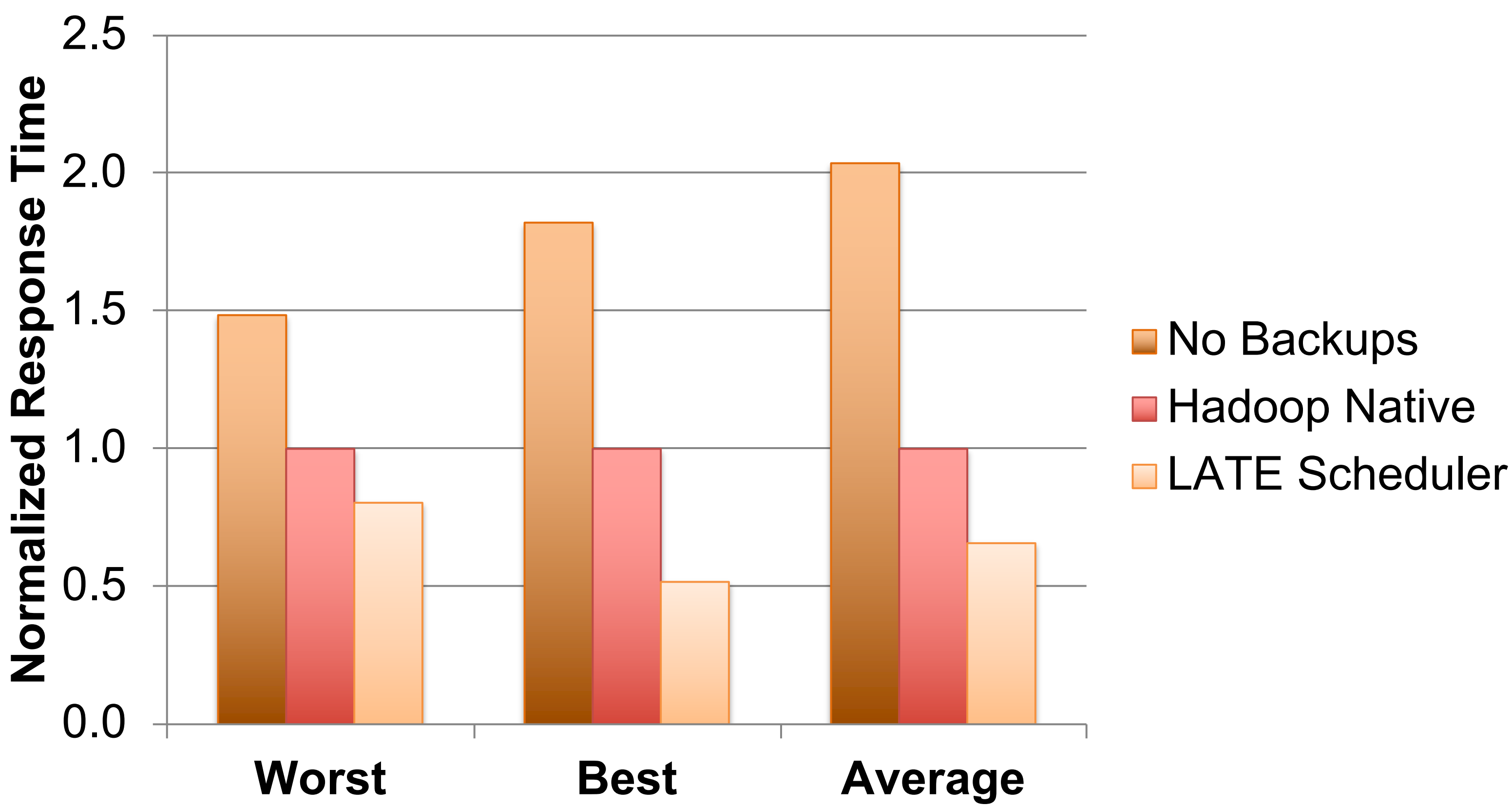
EC2 Sort without Stragglers (Sec 5.2.1)

- 106 machines , 7-8 VMs per machine → total of 243 VMs
- 128 MB data per host, 30 GB in total
- 486 map tasks and 437 reduce tasks
- average 27% speedup over native, 31% over no backups



EC2 Sort with Stragglers (Sec 5.2.2)

- 8 VMs are manually slowed down out of 100 VMs in total
- running background of CPU- and disk-intensive jobs
- average 58% speedup over native, 220% over no backups
- 93% max speedup over native



Conclusion

- Heterogeneity is a challenge for parallel apps, and is growing more important
- Lessons:
  - Back up tasks which hurt response time most
- 2x improvement using simple algorithm

Summary

- MapReduce is a very powerful and expressive model
- Performance depends a lot on implementation details

- Material is from the paper:

"Improving MapReduce Performance in Heterogeneous Environments",

By Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz and Ion Stoica, published in Usenix OSDI conference, 2008

- and their presentation at OSDI