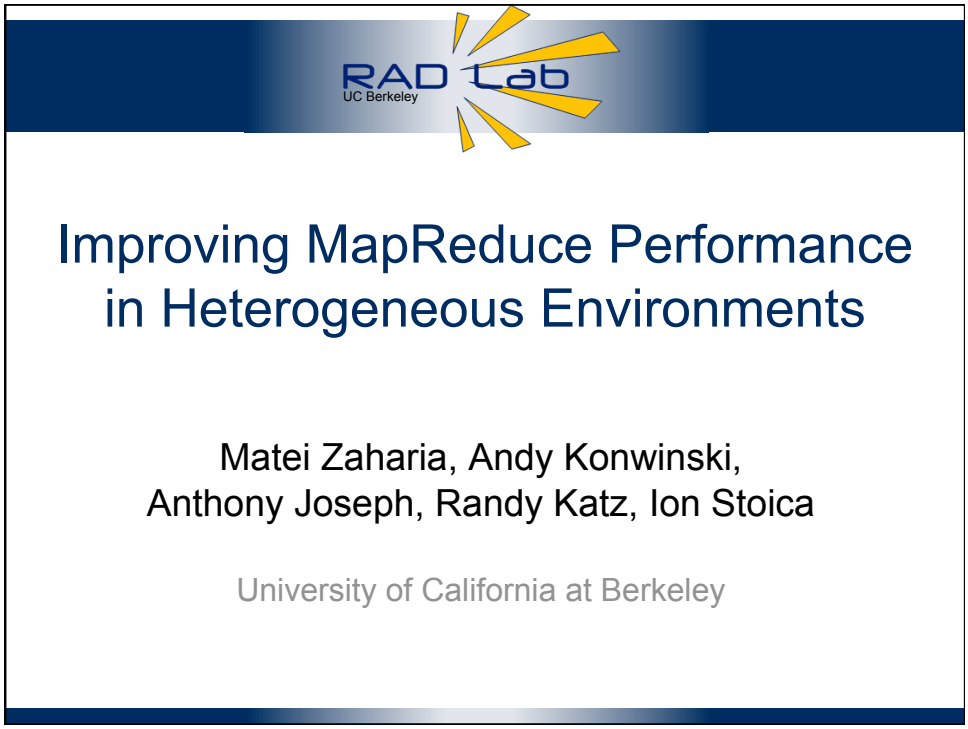
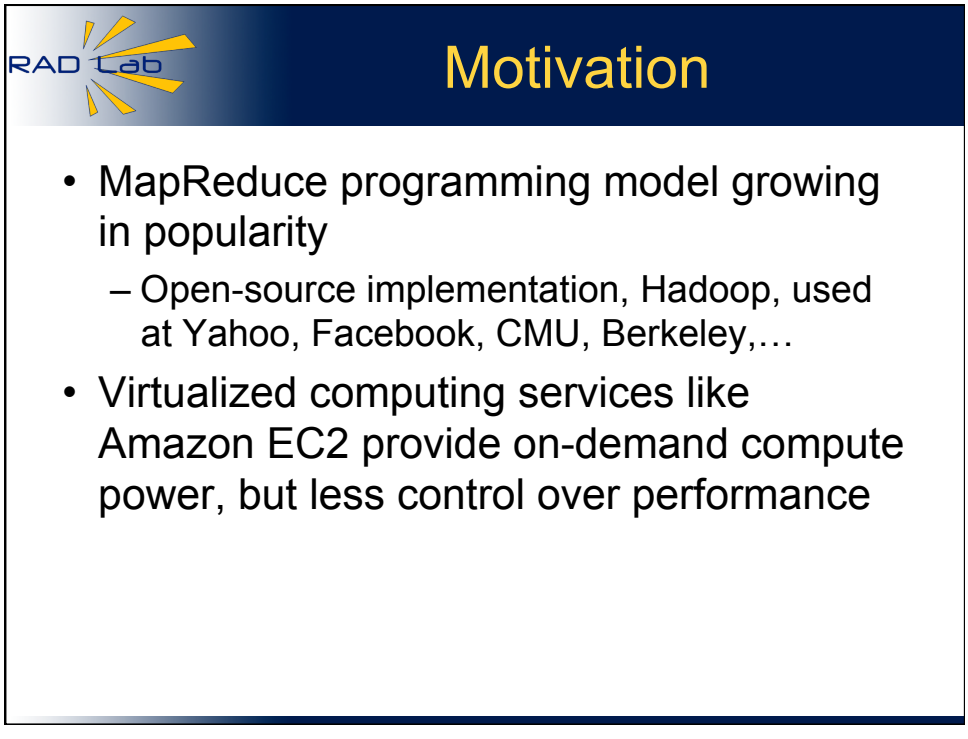


Improving MapReduce Performance in Heterogeneous Environments

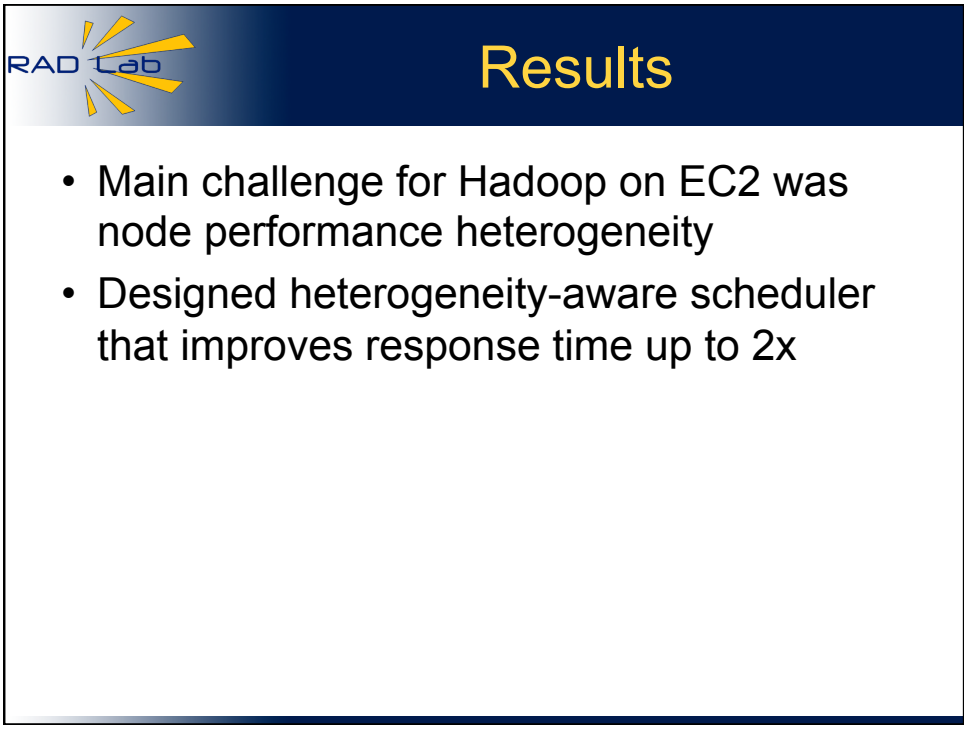
Matei Zaharia, Andy Konwinski,
Anthony Joseph, Randy Katz, Ion Stoica

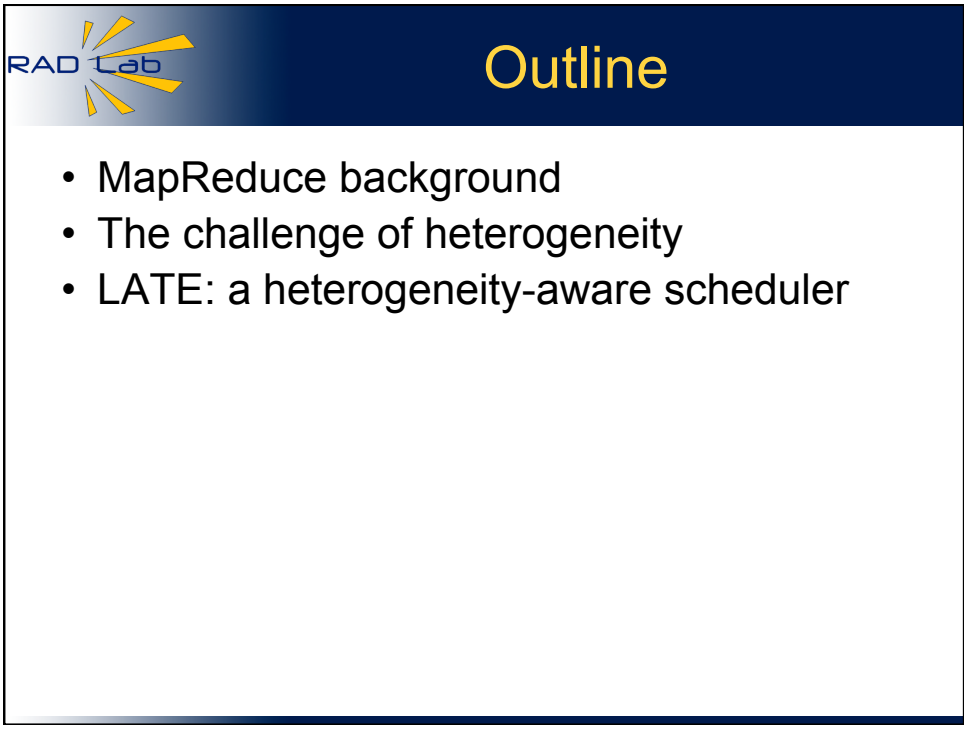
University of California at Berkeley



Motivation

- MapReduce programming model growing in popularity
  - Open-source implementation, Hadoop, used at Yahoo, Facebook, CMU, Berkeley,…
- Virtualized computing services like Amazon EC2 provide on-demand compute power, but less control over performance


RAD Lab
Results
• Main challenge for Hadoop on EC2 was node performance heterogeneity
• Designed heterogeneity-aware scheduler that improves response time up to 2x


RAD Lab
Outline
• MapReduce background
• The challenge of heterogeneity
• LATE: a heterogeneity-aware scheduler

What is MapReduce?

- Programming model to split computations into independent parallel tasks
  - Map tasks filter data set
  - Reduce tasks aggregate values by key
- Goal: hide the complexity of distributed programming and fault tolerance

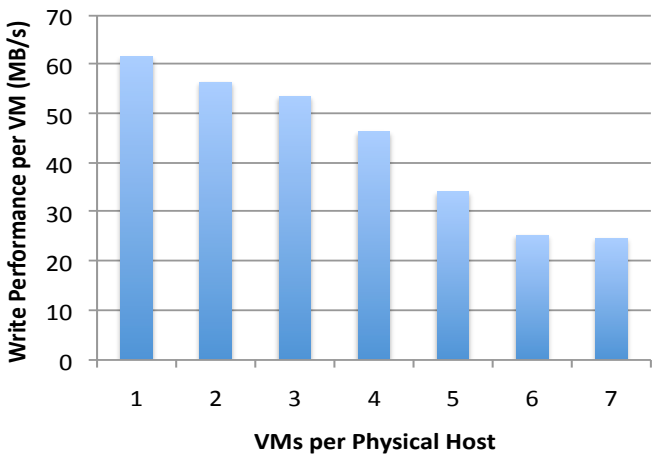
Fault Tolerance in MapReduce

- Nodes fail → re-run tasks
- Nodes very slow (stragglers) → launch backup copies of tasks
- How to do this in heterogeneous env.?

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
- 2.5x I/O performance difference on EC2

Disk Performance Heterogeneity Experiment



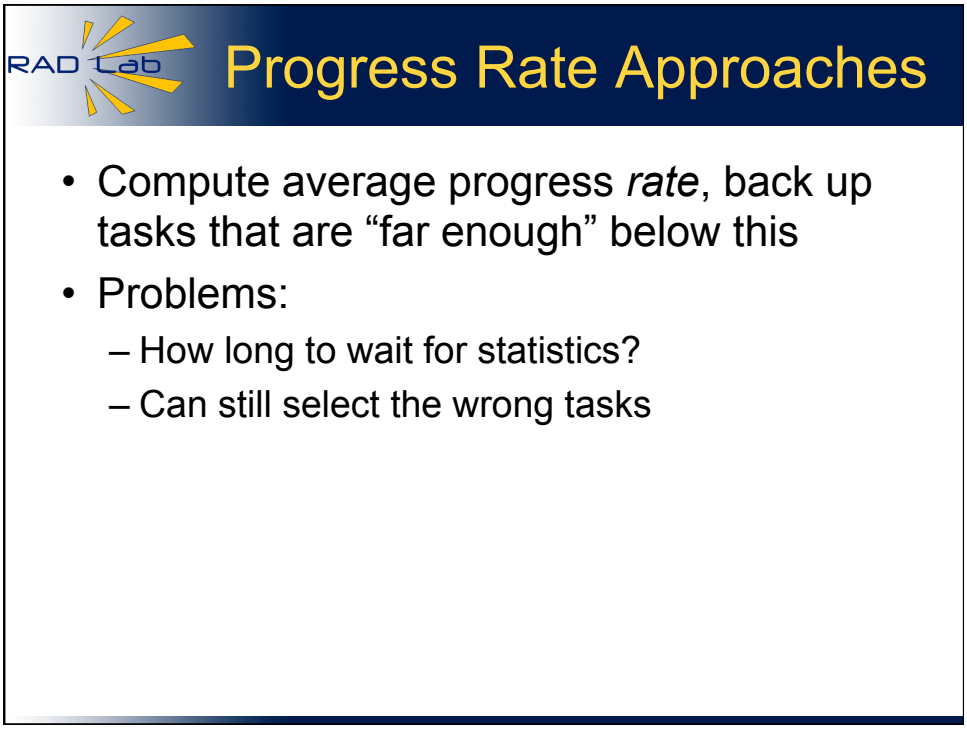
Backup Task Scheduling in Hadoop

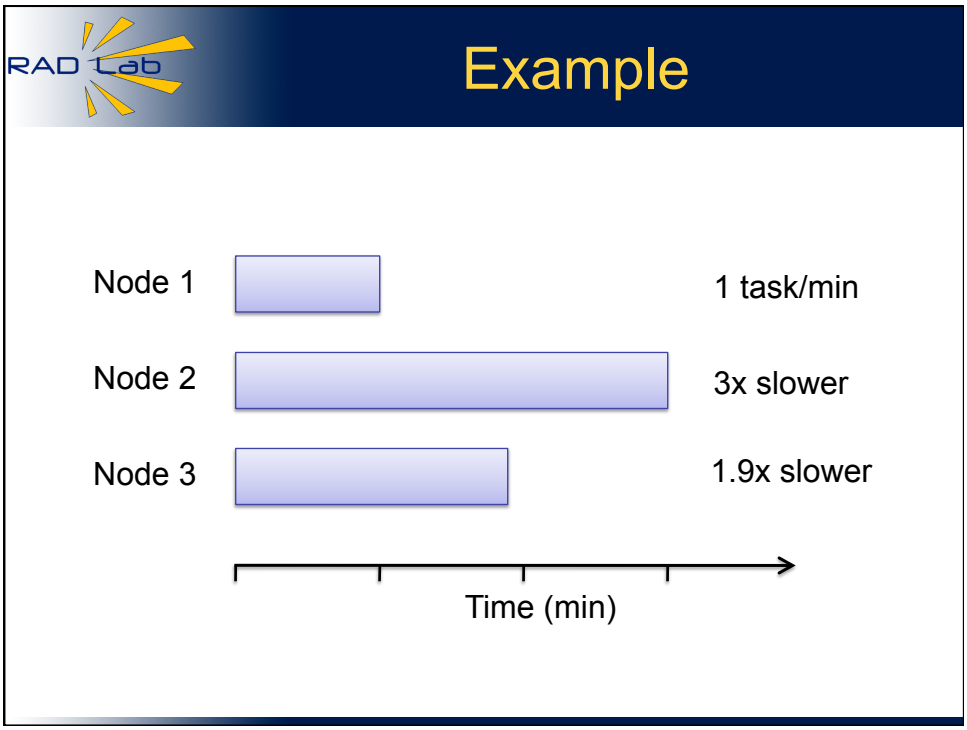
- Scheduler starts all primary tasks, then looks for tasks to back up
- Tasks report “progress score” from 0 to 1
- Backup launched if progress < avgProgress - 20%

Problems in Heterogeneous Environment

- *Too many* backups (trash shared resources)
- *Wrong* tasks may be backed up
- Backups may be placed on *slow nodes*
- *Tasks never backed up* if progress > 80%

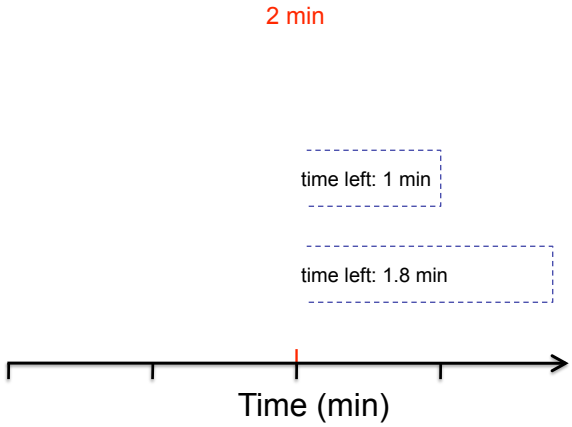
- Result: 80% of reduces backed up in some experiments, network overloaded


RAD Lab
Progress Rate Approaches
• Compute average progress *rate*, back up tasks that are "far enough" below this
• Problems:
– How long to wait for statistics?
– Can still select the wrong tasks


RAD Lab
Example
Node 1
1 task/min
Node 2
3x slower
Node 3
1.9x slower
Time (min)

Example

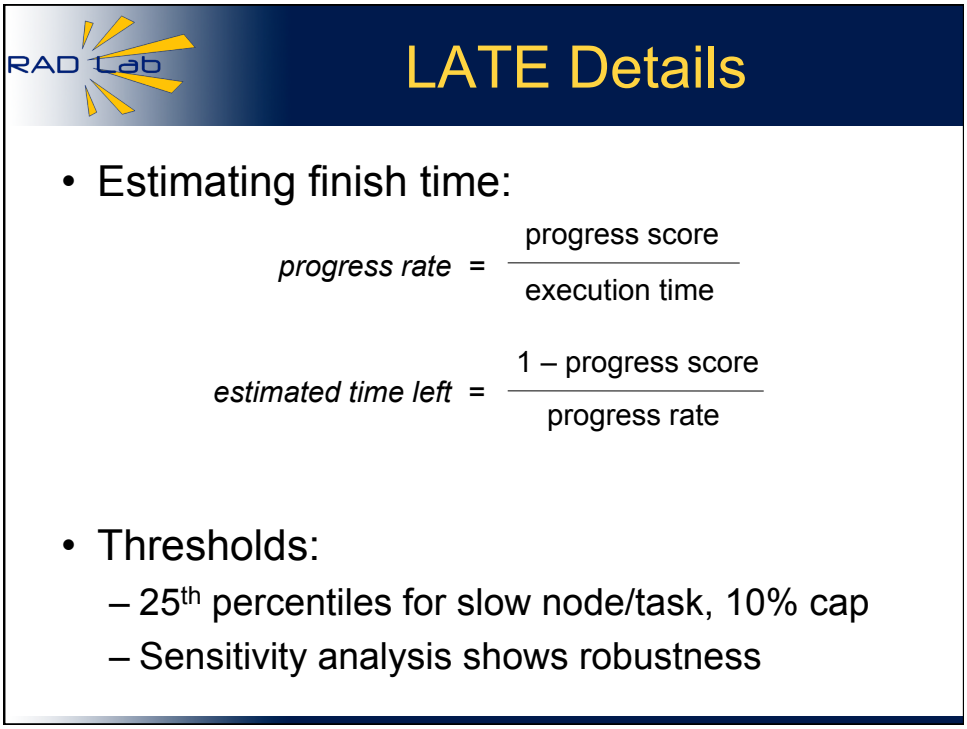
What if the job had 5 tasks?

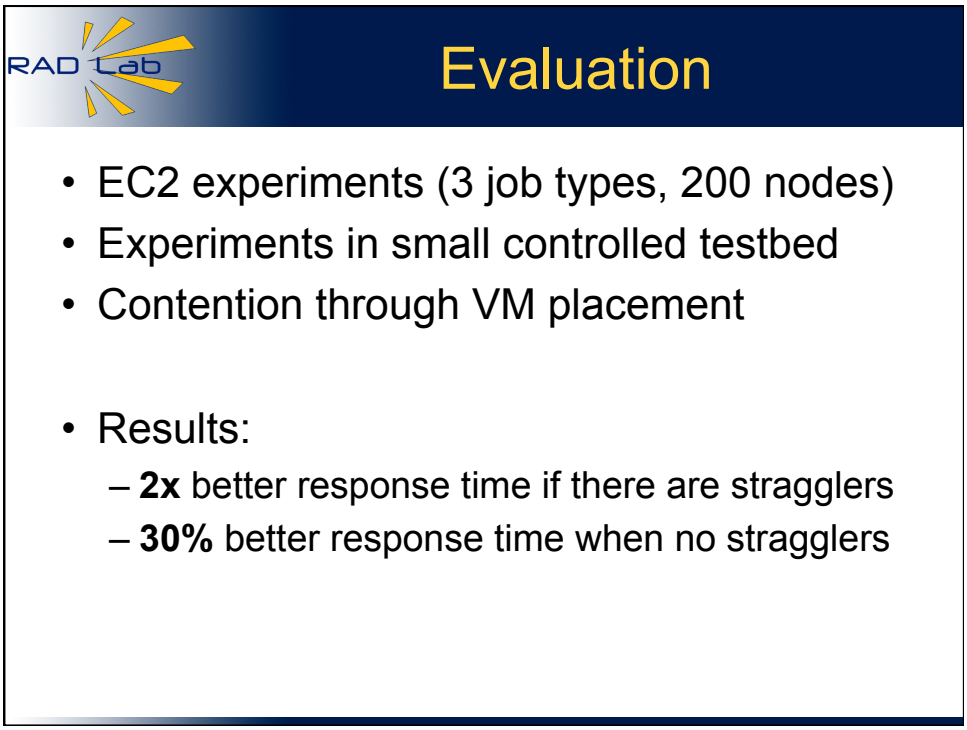


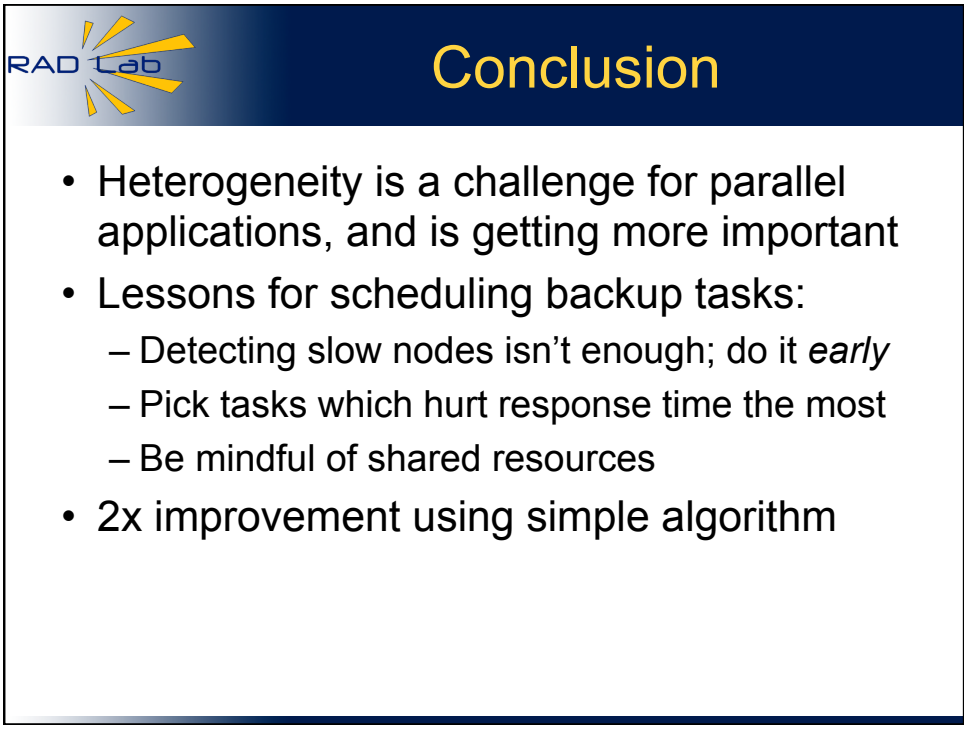
Should back up node 3's task

Our Scheduler: LATE

- Insight: back up the task with the largest *estimated finish time*
  - "Longest Approximate Time to End"
- Sanity thresholds:
  - *Cap* backup tasks to ~10%
  - Launch backups on *fast nodes*
  - Only back up tasks that are *sufficiently slow*


RAD Lab
LATE Details
• Estimating finish time:
progress rate = progress score / execution time
estimated time left = 1 – progress score / progress rate
• Thresholds:
– 25th percentiles for slow node/task, 10% cap
– Sensitivity analysis shows robustness


RAD Lab
Evaluation
• EC2 experiments (3 job types, 200 nodes)
• Experiments in small controlled testbed
• Contention through VM placement
• Results:
– 2x better response time if there are stragglers
– 30% better response time when no stragglers


RAD Lab
Conclusion
• Heterogeneity is a challenge for parallel applications, and is getting more important
• Lessons for scheduling backup tasks:
– Detecting slow nodes isn't enough; do it *early*
– Pick tasks which hurt response time the most
– Be mindful of shared resources
• 2x improvement using simple algorithm


RAD Lab
Questions?
Google
Microsoft
Sun microsystems
CISCO
FUJITSU
invent
IBM
NetApp
ORACLE
SIEMENS
vmware