Three Pieces of the MapReduce Workload Management Puzzle

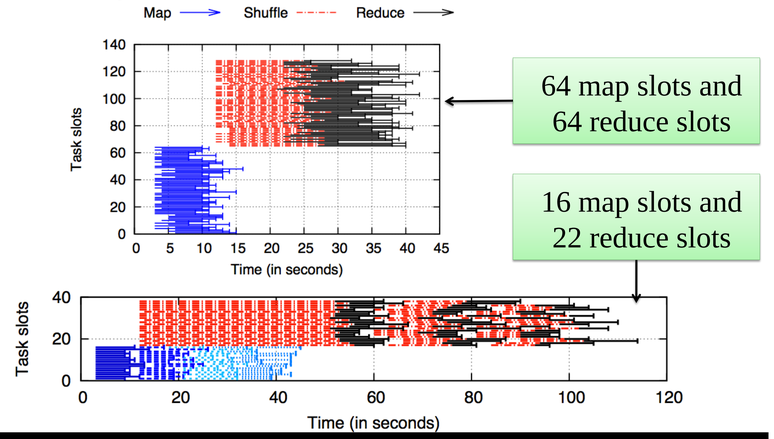
Abhishek Verma*, Ludmila Cherkasova#, Vijay S. Kumar#, Roy H. Campbell*
*{verma7, rhc}@illinois.edu University of Illinois at Urbana-Champaign, #{lucy.cherkasova, vijay.s.kumar}@hp.com HP Labs, Palo Alto

Problem

- Often MapReduce applications are a part of critical business pipelines and require job completion time guarantees (SLOs)
- **Problem:** Existing job schedulers do not support SLOs
- **Goal:** Design a workload management framework for efficient processing of MapReduce jobs with completion time goals
- Controlling ***tailored*** allocation and ***efficient use*** of resources in **shared** MapReduce environments is a key challenge

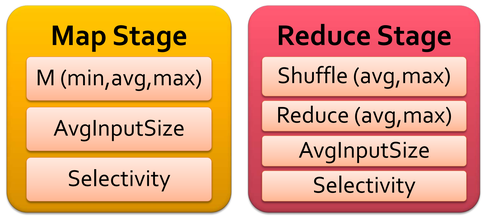
Job Execution with Different Resources

- Different amounts of resources can lead to drastically different job executions and different completion times



Job Profiles and MapReduce Performance Model

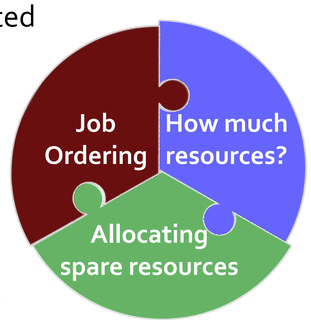
- Job Profiles compactly summarize performance metrics of different job stages collected from logs



- Automatic Resource Inference and Allocation (ARIA) with ***novel performance models*** that:
  - Can predict job completion time as a function of resources
  - Given a deadline, compute minimum resources to allocate to the job, so that it finishes within deadline

Three Pieces of the Puzzle

1. **Job Ordering**
   - Which order should the jobs be allocated resources?
2. **Tailoring amount of resources**
   - How many slots should be allocated to the chosen job?
3. **Allocating spare resources**
   - How to allocate the spare resources in the system and de-allocate them in case of a new urgent job?



Job Scheduling using Different Mechanisms

1. **Earliest Deadline First**
   - Allocate all the resources in the system to the job with the earliest deadline
2. **Min-EDF**
   - Compute the minimum resources to allocate to the job with the earliest deadline
3. **Min-EDF-WC**
   - Allocate any spare resources among running jobs
   - When new job arrives, compute if enough slots will be released in the future to satisfy the job
   - If not, cancel spare tasks of the currently running jobs

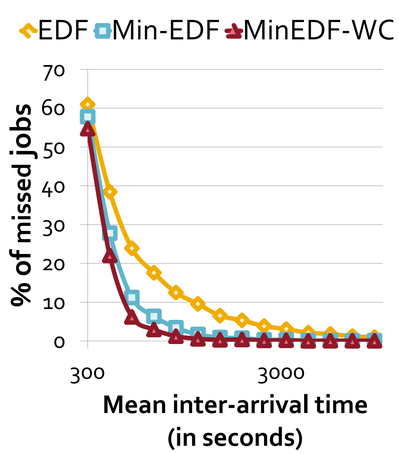
Evaluation Setup and Workloads

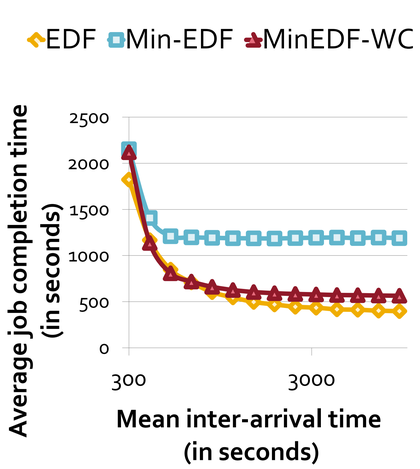
- **Testbed Setup**
  - 66 HP DL145 machines: 2 masters + 64 slaves
  - Four 2.39 GHz cores, 8 GB RAM, 2 x 160 GB hard disks
  - Two racks, Gigabit Ethernet
- **Workloads**
  - ***Real testbed trace*** of 1000 jobs with combinations of: Wordcount, Sort, Bayesian classification, TF-IDF, WikiTrends, Twitter on 3 different datasets
  - ***Synthetic Facebook trace***: generated using LogNormal distribution fit to 6 months of jobs

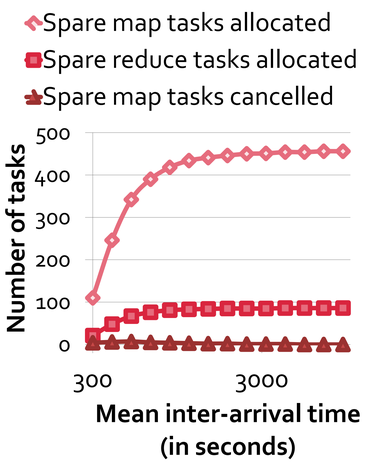
Simulator and Metrics

- **Replay traces using the *simulator SimMR***
  - Discrete event simulator replays job traces at task-level
  - Accuracy > 95%
  - Can replay two weeks workload in 2 seconds
- **Comparison metrics**
  1. % of missed jobs
  2. Average job completion time
  3. Number of spare slot allocations and cancellations

Evaluation







The simulation results with the Facebook workload are similar and reflect the same conclusions.

Future Work

- All three mechanisms are required for deadline-based workload management
- Incorporate these mechanisms in existing schedulers
- Scale smaller datasets to simulate larger ones
- **Dynamic resource adjustment**
  - Compare expected behavior against observed behavior and adjust
  - Deal with stragglers, input data skew