# Job-scheduling for Multi-user MapReduce clusters

# Outline

- Background
- Hadoop On Demand Issues
- FAIR Scheduler
- Obstacles to Fair Sharing

# Background

- Hadoop scheduling is FIFO, with 5 priority levels
- Due to poor response time between short/ long jobs, Hadoop introduced Hadoop On Demand (HOD)

# HOD Issues

- Poor locality
  - Since nodes have access to the entire HDFS, some map jobs have to work across the network
- Poor Utilization
  - Some nodes can be idle

# FAIR Scheduler

- Purpose: give all jobs slot-level granularity
  - Isolation: give each job the illusion of having their own cluster
  - Statistical Multiplexing: Redistribute unused capacity to other “pools”

# Pooling jobs

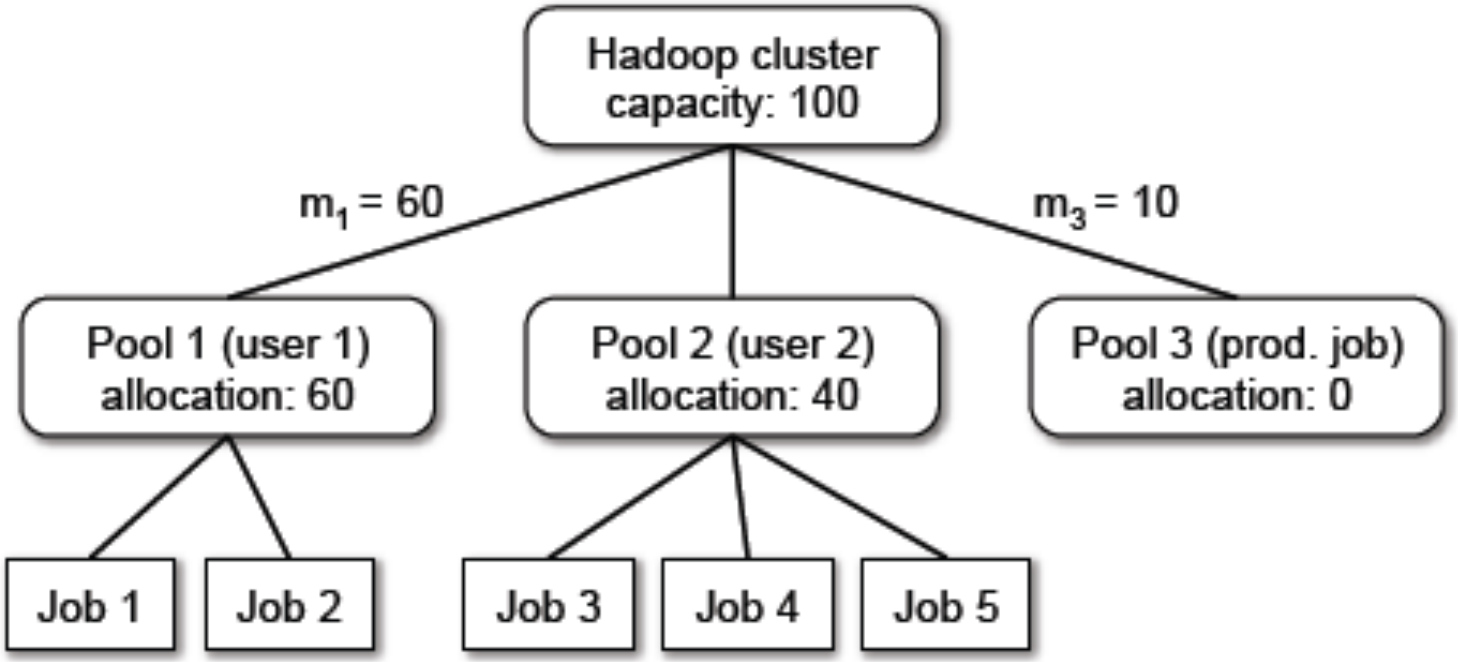
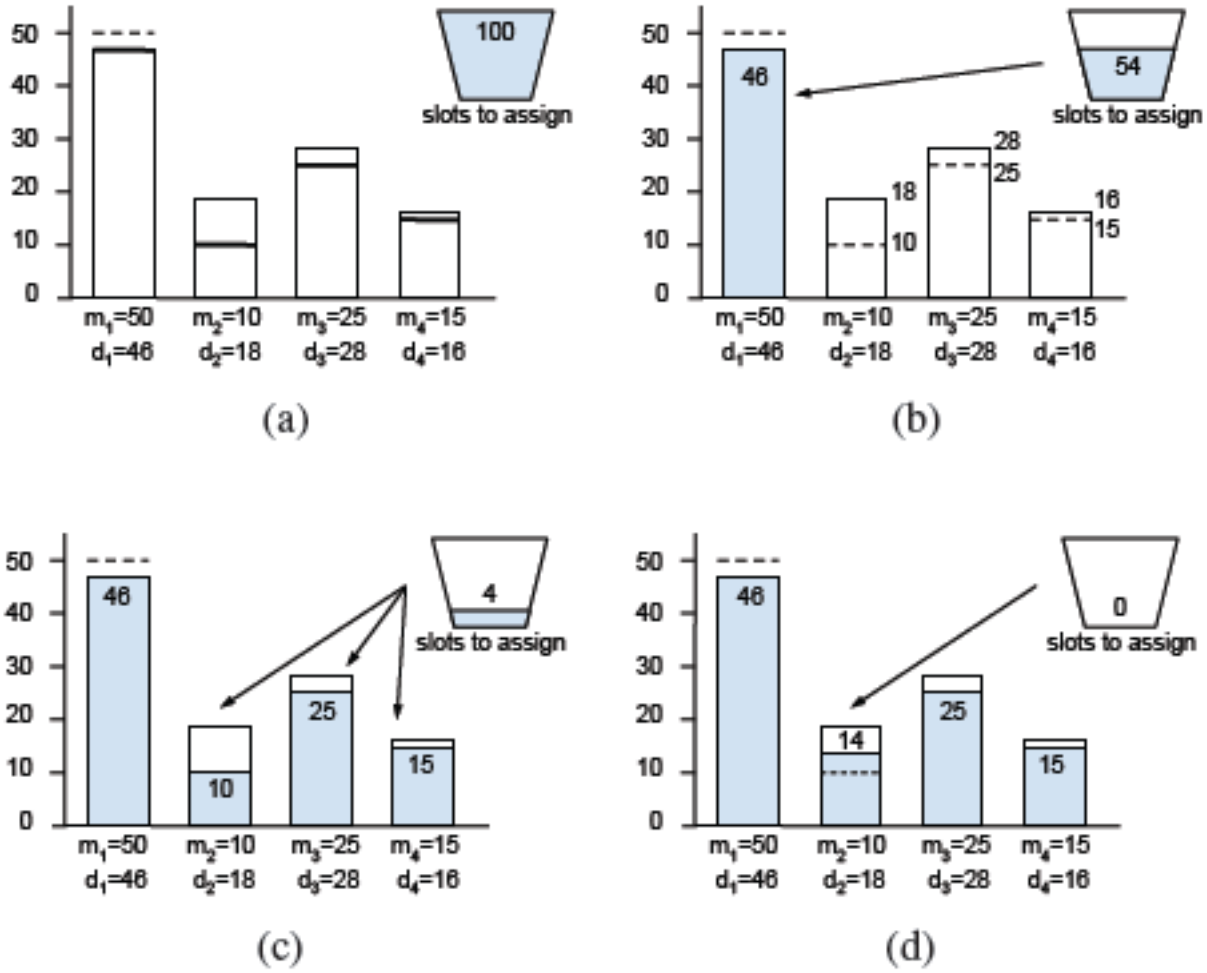



Figure 2: Example of allocations in our scheduler. Pools 1 and 3 have minimum shares of 60, and 10 slots, respectively. Because Pool 3 is not using its share, its slots are given to Pool 2.

# Redistribution of jobs



# Redistribution Explained

- $m_i$ = minimum number of shared needed to start the job
- $d_i$ = demand needed to complete the job
- Redistribution occurs by filling the min slots to complete as many tasks as possible
- $d_1$, $d_3$ ,$d_4$ should complete without needing to refill
- $d_2$ will require a refill (the last 4 slots to assign) to complete

# Obstacles to Fair Sharing

- Data Locality
  - Solution: Delay scheduling
    - Tasks are prioritized by locality
    - There are 2 wait times, one for the local pool wait, and one for the remote wait. The job will try to catch a local pool until the local wait time exceeds, then run on the next pool that's available.
    - There are 3 types of locality
      - Node local tasks
      - Rack-local tasks
      - Off-rack tasks

# Obstacles to FAIR Sharing

- Reduce/Map interdependence
  - "slot hoarding"
    - Long jobs hold reduce slots for a long time, starving short jobs
  - Solution: Copy-compute splitting
    - Split reduce jobs into two different jobs
      - Copy task (Network IO job)
        - » Fetches and merges map outputs
      - Compute task (Reduce job)
    - There is a controller CPAC which checks 2 fields
      - maxReducers
      - maxComputing
    - eg. 6 simultaneous reducers, but 2 able to compute