



CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

SCHEDULING

Lecture B

HADOOP SCHEDULING

HADOOP SCHEDULING

- A Hadoop job consists of Map tasks and Reduce tasks
- Only one job in entire cluster => it occupies cluster
- Multiple customers with multiple jobs
 - Users/jobs = “tenants”
 - **Multi-tenant system**
- => Need a way to schedule all these jobs (and their constituent tasks)
- => Need to be *fair* across the different tenants
- Hadoop YARN has two popular schedulers
 - *Hadoop Capacity Scheduler*
 - *Hadoop Fair Scheduler*



HADOOP CAPACITY SCHEDULER

- Contains multiple queues
- Each queue contains multiple jobs
- Each queue guaranteed some portion of the cluster capacity
 - E.g.,
 - Queue 1 is given 80% of cluster
 - Queue 2 is given 20% of cluster
 - Higher-priority jobs go to Queue 1
- For jobs within same queue, FIFO typically used
- Administrators can configure queues

ELASTICITY IN HCS

- Administrators can configure each queue with limits
 - Soft limit: how much % of cluster is the queue guaranteed to occupy
 - (Optional) Hard limit: max % of cluster the queue is guaranteed
- Elasticity
 - A queue allowed to occupy more of cluster if resources free
 - But if other queues below their capacity limit, now get full, need to give these other queues resources
- Pre-emption not allowed!
 - Cannot stop a task part-way through
 - When reducing % cluster to a queue, wait until some tasks of that queue have finished

OTHER HCS FEATURES

- Queues can be hierarchical
 - May contain child sub-queues, which may contain child sub-queues, and so on
 - Child sub-queues can share resources equally
- Scheduling can take memory requirements into account (memory specified by user)



HADOOP FAIR SCHEDULER

- Goal: all jobs get equal share of resources
- When only one job present, occupies entire cluster
- As other jobs arrive, each job given equal % of cluster
 - E.g., Each job might be given equal number of cluster-wide YARN containers
 - Each container == 1 task of job

HADOOP FAIR SCHEDULER (2)

- Divides cluster into pools
 - Typically one pool per user
- Resources divided equally among pools
 - Gives each user fair share of cluster
- Within each pool, can use either
 - Fair share scheduling, or
 - FIFO/FCFS
 - (Configurable)

PRE-EMPTION IN HFS

- Some pools may have *minimum shares*
 - Minimum % of cluster that pool is guaranteed
- When minimum share not met in a pool, for a while
 - Take resources away from other pools
 - By pre-empting jobs in those other pools
 - By *killing* the currently-running tasks of those jobs
 - Tasks can be re-started later
 - Ok since tasks are idempotent!
 - To kill, scheduler picks most-recently-started tasks
 - Minimizes wasted work

OTHER HFS FEATURES

- Can also set limits on
 - Number of concurrent jobs per user
 - Number of concurrent jobs per pool
 - Number of concurrent tasks per pool
- Prevents cluster from being hogged by one user/job

ESTIMATING TASK LENGTHS

- HCS/HFS use FIFO
 - May not be optimal (as we know!)
 - Why not use shortest-task-first instead? It's optimal (as we know!)
- Challenge: Hard to know expected running time of task (before it's completed)
- Solution: Estimate length of task
- Some approaches
 - Within a job: Calculate running time of task as proportional to size of its input
 - Across jobs: Calculate running time of task in a given job as average of other tasks in that given job (weighted by input size)
- Lots of recent research results in this area!

SUMMARY

- Hadoop Scheduling in YARN
 - Hadoop Capacity Scheduler
 - Hadoop Fair Scheduler

- Yet, so far we've talked of only one kind of resource
 - Either processor, or memory
 - How about multi-resource requirements?
 - Next!