NRGM SCHEDULING

NGRM High Level Review

March 4, 2013

Goals of the presentation

- To present the terrain of the resource scheduling problem
- To introduce potential design directions

Job Scheduling

- Selecting and allocating computing resources to a job over the course of time.
- Selecting a set of resources on which to launch a job
  - Single task
  - Parallel (MPI) job.

Fundamental Scheduling Choice

- Schedule jobs to resources
  - The scheduler cycles through a prioritized list of jobs
  - Pick off each top priority job and find resources for it.
- Schedule resources to jobs
  - The scheduler cycles through a collection of resources and finds jobs which can run on them

Scheduling Activities

- Job Prioritization
  - Plugins that implement prioritization algorithms
    - Submission Order, Fair-share, Job Size, custom
- Resource Scheduling (aka, the Scheduling Loop)
  - Selects the resources for each job over the course of time
  - Plugins that implement resource selection algorithms
    - First Fit, Best Fit, Backfill, network topology, custom
  - Must support shared and exclusive resource allocations
- Job Preemption
  - Targeting running jobs for removal as part of the resource scheduling activities

NOTE 1: FIFO scheduling implies job prioritization based on submission order + first fit scheduling

NOTE 2: While a “simple” scheduler can have a simple job prioritization algorithm (submission order), the resource scheduling algorithm will rarely be “simple”.

Job Prioritization Factors

- Submission Order
- Job Size
- Job Time Limit
- (Size * Time Limit)
- Quality of Service
- Queue
- Fair-share
- A blend of all the above

Scheduling Plugins

- Support complex job dependencies, e.g. as in scientific workflows
- Backfill lower priority jobs
- Support for dynamic job growth and reduction
- Preempt running jobs to free up resources needed by more important jobs
- Calculate estimates of when each job will begin
- Provide scheduling answers to hypothetical scenarios
  - When would a job needing x resources run?
  - What if I asked for fewer resources for a longer time?
- Scheduling different resources to a job over time
- Gang scheduling

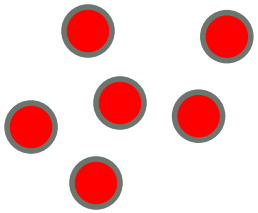
Resource Scheduling

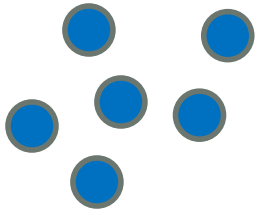
- The NGRM model calls for resource scheduling services to be associated with the job.
  - Allows for hierarchical scheduling decisions. E.g., job 0.1 runs on 16 nodes, but scheduling jobs 0.1.1 and 0.1.2 is an independent activity localized to job 0.1's instance.
- This model works to relieve the work of scheduling all jobs in the system by a single agent.

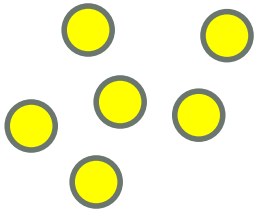
Worst Case Scheduling

- Even with hierarchical scheduling, scheduling lots of small (single core), short duration jobs submitted to job 0 presents a scaling challenge.
- The following slides review basic scheduling algorithms and end with suggestions for shortcuts.

Scheduling Marbles

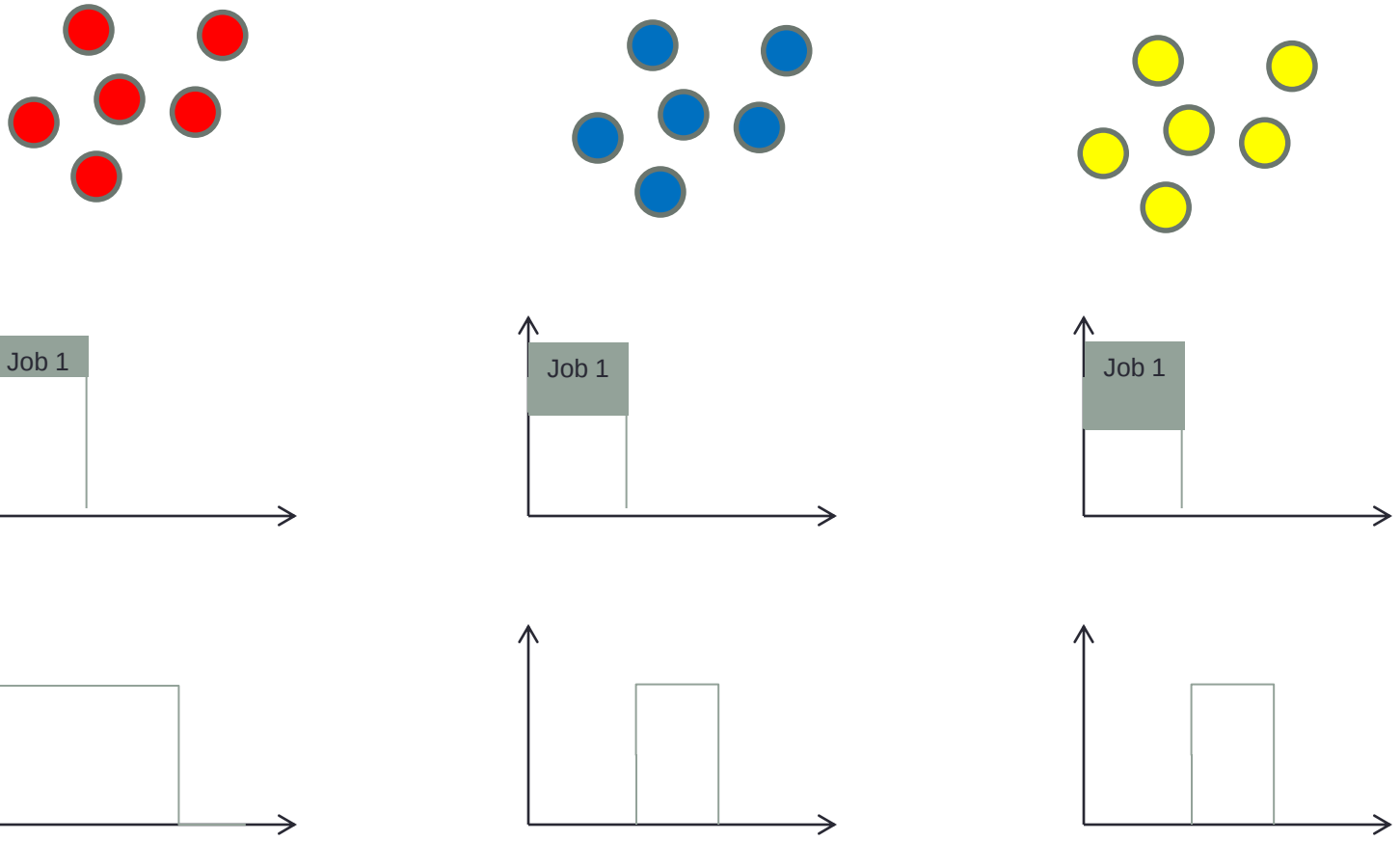




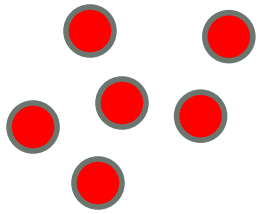


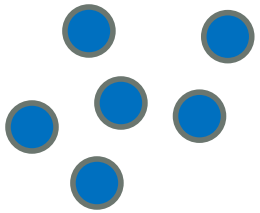
- Job 1 wants 1 red, 2 blue, and 3 yellow marbles for 10 minutes
- Job 2 wants 5 red, 5 blue, and 5 yellow marbles for 10 minutes

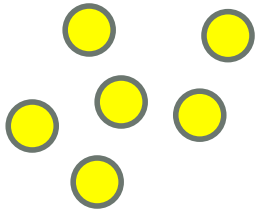
Scheduling Marbles



Job 2 is Scheduled

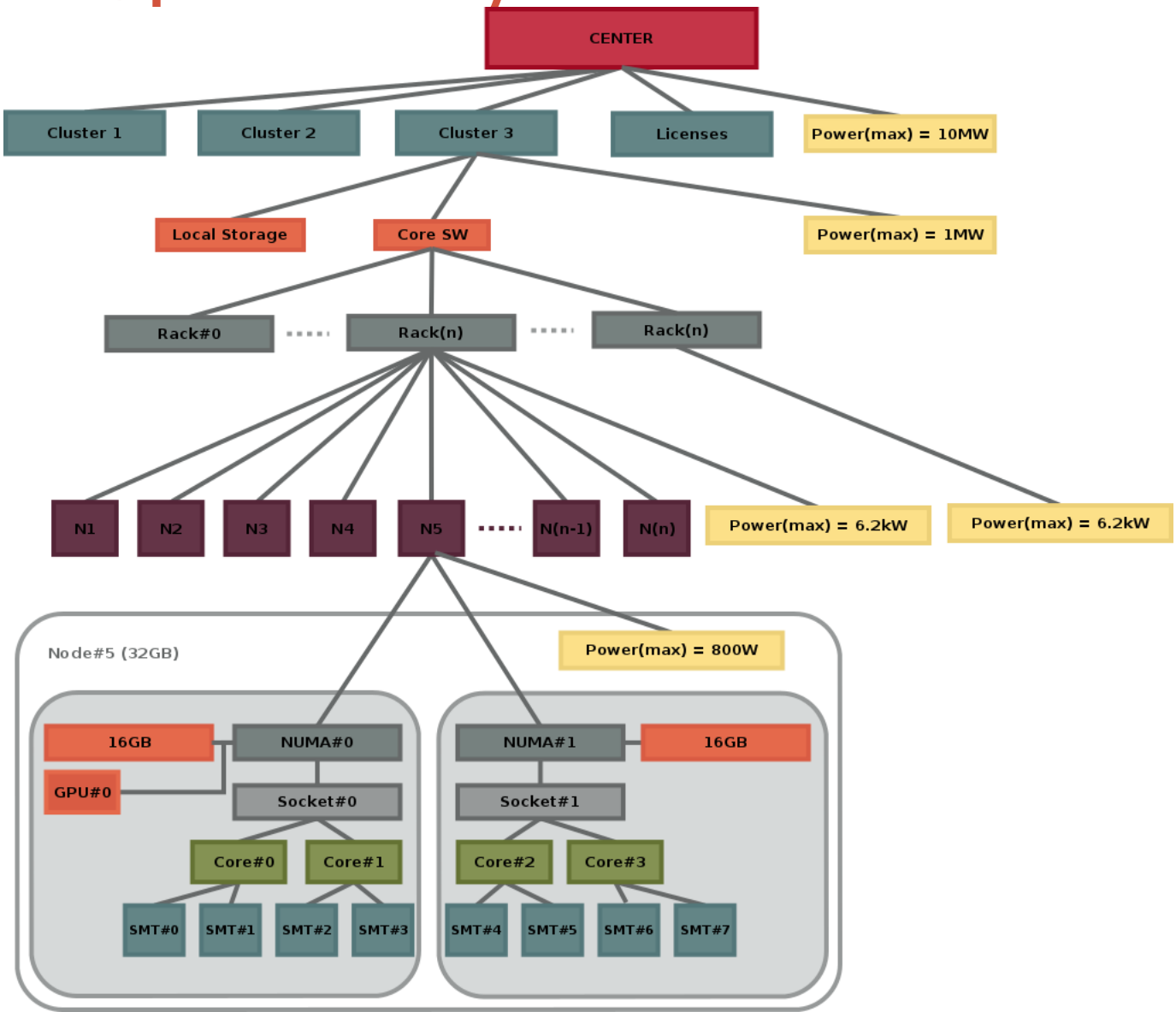






Scheduling Russian Dolls

More Specifically…



Scheduling - First Pass

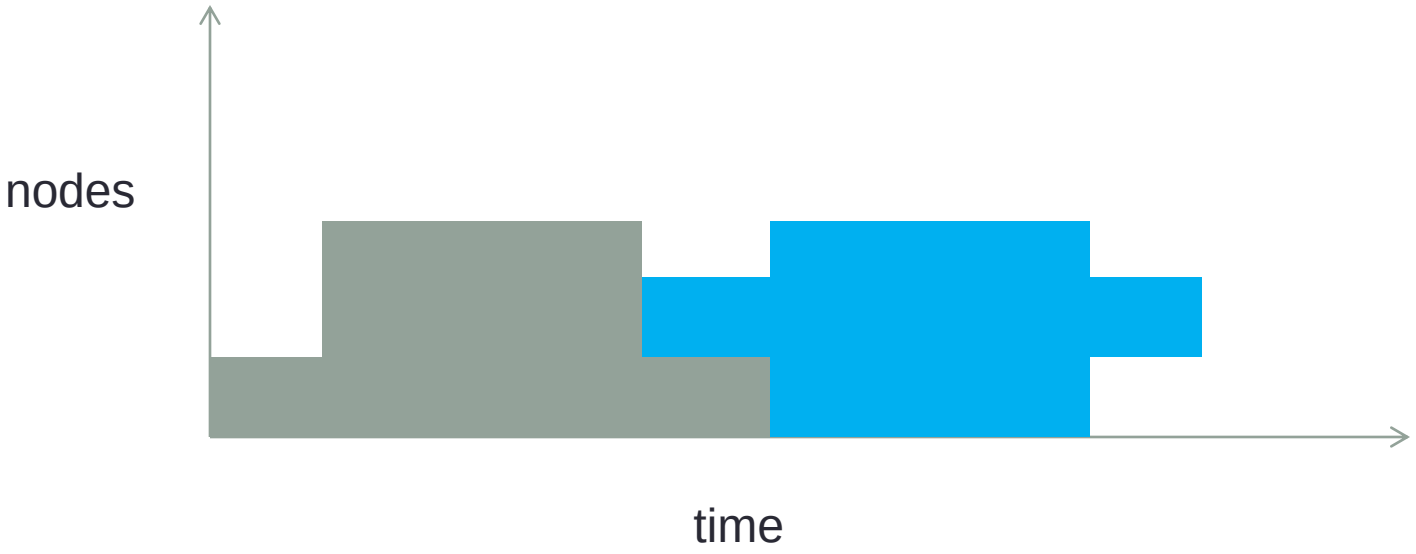
- The scheduler at the highest level must have visibility into all the resources down to the core level to schedule jobs submitted to job.0
- How all the center's resources are represented in memory (or the repository) is an implementation detail
- Job 0's scheduler could have a **big** task of scheduling short, single core jobs across every resource in the center (worst case scenario).

Scheduling Shortcuts to Better Performance

- Always push the scheduling decisions down to the child job whenever possible (e.g., DATs)
- Establish running job “queues”
  - A job for the traditional batch and debug partitions
  - A job for all the core-scheduled nodes (aztec)
- Commit to first scheduling decision and only occasionally (if ever) reconsider that decision
- Begin the scheduling loop with the highest priority *newly submitted* job

Scheduling Resources to a Job that Change over Time

- Scheduling a series of resource requirements as one job (without relying on a series of dependent jobs)
  - File staging - lots of I/O for 30 minutes, then
  - A bunch of compute nodes for 8 hours, then
  - File storage - I/O bandwidth for 30 minutes



Reactive vs. Proactive Scheduling

- All (known) batch schedulers to date can be considered reactive. They schedule in reaction to state changes: newly submitted jobs, newly completed jobs, etc.
- There could be a scheduler that is proactive. It monitors the status of all its jobs and makes adjustments dynamically to optimize performance.
  - Moving a task from a core that appears to be failing
  - Balancing load dynamically
  - Growing a job to make use of idle resources
  - Reducing monitoring/logging when bandwidth is suffering