CMSC714
Lecture 21
Job Scheduling

Alan Sussman
(mostly from Abhinav Bhatele)

UNIVERSITY OF
MARYLAND

Notes

- Midterm exam next Thursday, November 16, in class
  - Sample questions posted on Exams web page
  - Email me if you are out of town next week, to arrange a time to take the exam
- Interim report for group project due Monday, 6PM
- No class Tuesday

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a "job" scheduler

Job Queue

	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a "job" scheduler
- The scheduler decides:
  - what job to schedule next (based on an algorithm: FCFS, priority-based, ….)
  - what resources (compute nodes) to allocate to the ready job

Job Queue

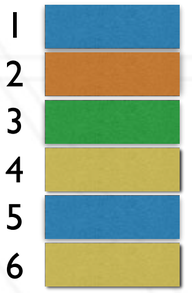
	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a "job" scheduler
- The scheduler decides:
  - what job to schedule next (based on an algorithm: FCFS, priority-based, ….)
  - what resources (compute nodes) to allocate to the ready job

- Compute nodes: dedicated to each job
- Network, filesystem(s): shared by all jobs

Job Queue



	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Job scheduling

- HPC systems use job or batch scheduling
- Each user submits their parallel programs for execution to a "job" scheduler
- The scheduler decides:
  - what job to schedule next (based on an algorithm: FCFS, priority-based, ….)
  - what resources (compute nodes) to allocate to the ready job

Concurrently running jobs can contend for shared resources: network, filesystem

- Compute nodes: dedicated to each job
- Network, filesystem: shared by all jobs

Job Queue

	#Nodes Requested	Time Requested
1	128	30 mins
2	64	24 hours
3	56	6 hours
4	192	12 hours
5	…	…
6	…	…

Two components of a scheduler

- Decide what job(s) to schedule next: scheduler
- Decide what nodes (and other resources) to allocate to them: resource manager

Scheduling policies

- First come first serve (FCFS)
- Priority-based
  - Depending on project name and remaining allocation
- Backfilling
  - Use idle nodes that are being reserved for the next large jobs
  - Aggressive (EAZY) backfill: run jobs as long as they don't delay the first job in the queue (could lead to unbounded delays)
  - Conservative backfill: runs jobs as long as they don't delay **any** future job

Resource management

- Most primitive: manage nodes
- Advanced management:
  - Node type aware (low vs. high memory, GPU nodes, etc.)
  - Network topology aware
  - Power aware

Space sharing and time sharing

- Space sharing: Exclusive access to a resource until job completion
- Time sharing: Interleaved access to the same resource
  - Co-scheduling
  - Gang scheduling

Quality of service metrics

- Job Wait Time: time between a job's submission and start

$$T_{wait} = T_{start} - T_{submit}$$

- Slowdown: incorporates running time of a job

$$\text{Slowdown} = \frac{T_{wait} + T_{running}}{T_{running}}$$

Quality of service metrics

- System Utilization: fraction of nodes allocated to running jobs at a given time

$$utilization_t = \frac{N_t}{N}$$

- Schedule Makespan: time between the first job's submission and last job's completion for a job trace (workload)

PBS paper - Takeaways

- Separating job scheduling policy from resource management makes it possible for sites to manage their own resources as they see fit – to optimize for throughput, give priority to specific groups of users (at certain times), or whatever the resource owner desires
  - The real power is in managing clusters to run parallel (e.g., MPI) jobs, not single machines as is mainly discussed in the paper
- PBS is the beginning of a lot of efforts at schedulers for clusters, including SLURM
  - Eventually 2 companies formed to support PBS, and later a derivative called Torque (PBS and Torque were both used on UMD clusters, before SLURM)
  - An open source version, OpenPBS, is still used, but at many sites has been supplanted by SLURM

Gang Scheduling/Backfilling paper

- A study to take a careful look at the benefits of the two scheduling methods, which are complementary
  - Conclusion is that backfilling is the big win, since it allows for utilizing resources that would otherwise go unused with a FCFS policy, or other standard policies
  - But gang scheduling helps by enabling multiple jobs to utilize the same nodes at the same time – time sharing in addition to space sharing
    - Gang scheduling ensures that all processes for the same job run at the same time (really important for MPI, and other parallel, jobs)
  - Multiprogramming level does not seem to matter all that much, once it is more than 1
    - And higher levels of over-estimation of job run times do not seem to hurt much, especially when using both gang scheduling and backfilling
    - For all the metrics, including responsiveness, slowdown, fairness, and utilization