COBALT V2 (CV2):

WHAT WOULD YOU DO WITH A BLANK SHEET OF PAPER?

WILLIAM (BILL) ALLCOCK (ALLCOCK@ANL.GOV)

Aug 5th, 2019 @ The International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems

OUTLINE

- Brief Introduction / Background Material
- Looking Ahead
- Design / Technology Ideas
- What will it take to get there
- Summary

BACKGROUND

- The United States Department of Energy (https://www.energy.gov/)
  - Funds a significant amount of research in the US including the national laboratory system, which Argonne is a part of.
  - https://www.energy.gov/maps/doe-national-laboratories
- Argonne National Laboratory – located outside of Chicago (https://www.anl.gov/)
- Argonne Leadership Computing Facility (ALCF)) https://www.alcf.anl.gov/)
  - Who I work for; We are a "User Facility" and run supercomputers for scientists from around the world to use.
- "Leadership Computing"
  - We want the biggest, baddest, computational science problems that just can't be solved anywhere else. This influences our scheduling policies.
- INCITE, ALCC, DD – These are the programs that allocate time on our systems
  - https://www.doeleadershipcomputing.org/ (60%)
  - https://science.osti.gov/ascr/Facilities/Accessing-ASCR-Facilities/ALCC (30%)
  - https://www.alcf.anl.gov/dd-program (10%)

WHY COBALT?

- The ALCF had two requirements that lead us to Cobalt
  - Needed to work on Blue Gene
  - Needed to be able to boot alternate operating systems on the Blue Gene
- There was no scheduler in the world that could do both of those things.
- The Argonne Math and Computer Science division was already running a small Blue Gene/L and they were using Cobalt.
- We had to modify something, so we chose Cobalt.
  - It already worked on a Blue Gene
  - The developers were right down the hall from us
- At this point, we have scheduling algorithms that work well for our workload and we have built a lot of custom tooling around it.
- We have an incredible opportunity. The ability to have a blank sheet of paper to architect from comes once or twice in a lifetime. We want to get it right…

USE CASES

How are the workloads changing now? 20 years from now?

- Traditional HPC
  - MPI jobs, multi-hour
- High Throughput Jobs
  - ”embarrassing parallel”, single node or core, minutes or seconds
- On-demand Jobs
  - Driven by Instruments such as the Advanced Photon Source
  - Move data, analyze it, and get the results back to adjust the next run
- Malleable Jobs
  - Job wants to change its resource set during the job
  - Machine Learning and AI jobs often have this trait
- Cloud, external services, VMs, containers

SCALE

One thing we can count on is that scale is going to increase

- What scale targets are we designing for?
  - Schedulable entities (nodes, cores, GPUs, network links, etc.)
    - Billions
  - Simultaneous Executions
    - One per “execution unit” (node, core, GPU, etc)
  - Submission rate
    - Per user – their last mile bandwidth should be the bottleneck
    - Aggregate – linearly, horizontally scalable; It should just be a question of resources, not scheduler limitations.
  - Job start rate
    - Again, it needs to be linearly, horizontally scalable to support the above
  - Queue depths
    - Unlimited for users, policy driven for what is eligible to be scheduled.

WHAT WOULD I DO WITH A CLEAN SHEET?

I wouldn’t use a clean sheet of paper

- There is something intoxicating about the idea of writing every line of code yourself, but it isn’t practical.
- The real question when you have a “clean sheet of paper” is what approaches, tools, libraries, services, etc. will you build on and how will you build around them and what “special sauce” will you add of your own?
- For the remainder of this talk I will be discussing our ideas on just that.

If you would like to help us do this, we would welcome it.

I WOULDN'T USE A BLANK PIECE OF PAPER

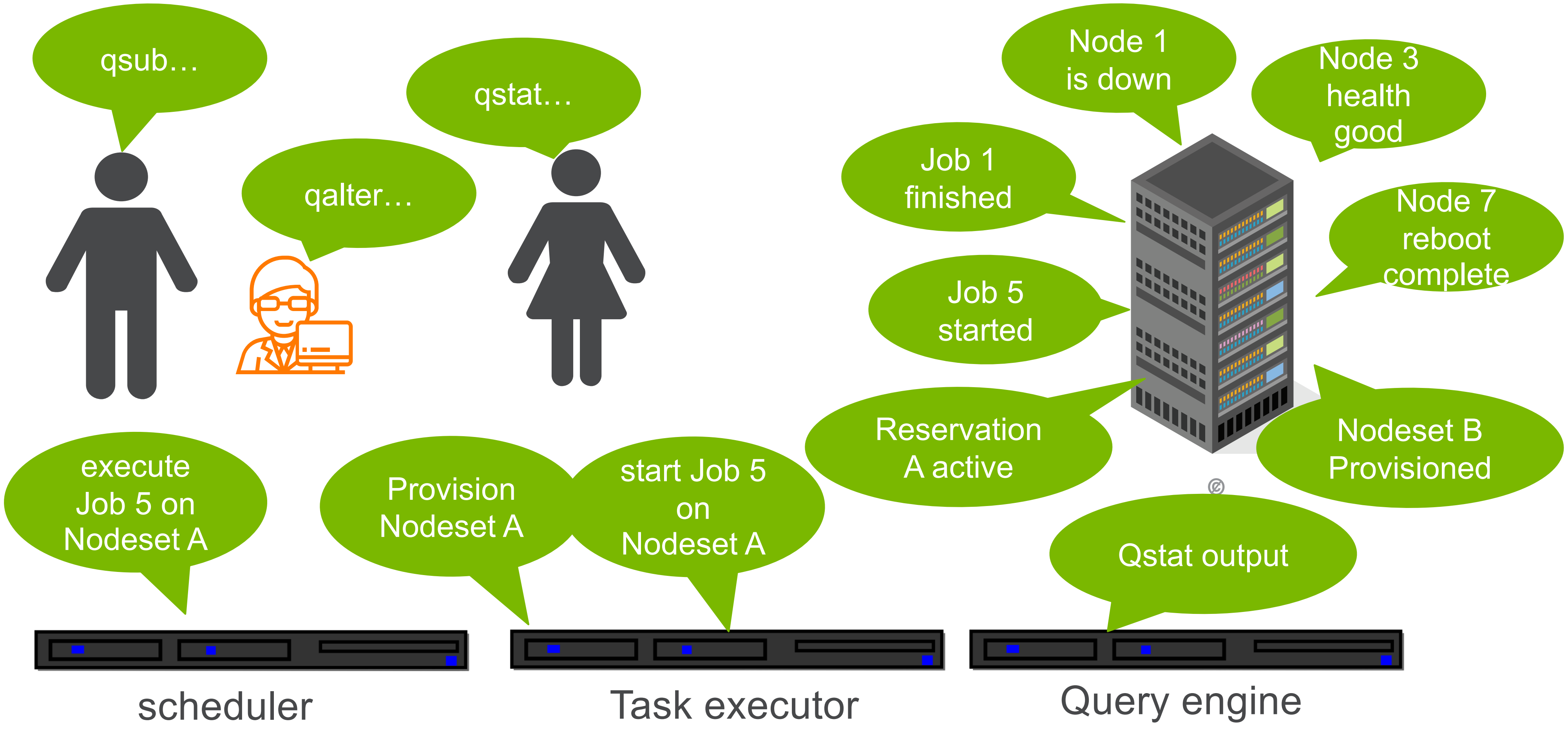
We plan on collaborating with the PBS Pro Open Source Community

- We initially approached Altair Engineering (the primary commercial entity behind PBS Pro) about extracting their resource management components.
- We met with Altair to discuss this. The meeting was so successful that my team came back and unanimously suggested that rather than extract the resource manager, we just use PBS Pro as the base and work from there. Why?
  - We had many of the same ideas for the future
  - The teams meshed really well
  - Many of the weak points of Cobalt immediately improve (documentation, support, the resource manager, etc.)
  - Risk mitigation: We could meet our mission goals with very few changes.
  - The one down side: They have an established code and user base, so it will make some of the changes we want to make more difficult.
  - Overall, we both believe it will be a huge win for everyone and are committed to making it work.

TECHNOLOGY CHOICES

- Event Sourcing
  - Everything is driven by events.
  - Recovery is by a combination of checkpoints and event replay
- Command Query Responsibility Separation (CQRS)
  - Separate the command code/storage model from Query code/storage model
  - I.e. optimize the command event processors for write and the query for read
- Massive Concurrency
  - Actor Model – Actors are relatively simple and single threaded, but allow massive concurrency
  - Akka – Implementation of the Actor Model
    - Used by twitter, Netflix and several other large scale web companies
    - By far the most mature and full featured Actor implementation out there.
    - Resilience is built into the model; Can cluster across machines or data centers
    - Has support for Event Sourcing, event persistence, and CQRS available
    - A sort of "micro services" model

HOW A SCHEDULER LOOKS LIKE TWITTER



JOBS WILL BE REPRESENTED AS GRAPHS

Nothing new here… move along… move along

- Workflow engines and such have done this for a long time, but we decided to integrate the concept into CV2
- Facilitates the "global facility view"
  - Your job can include nodes on different resources
  - Support for parallel services that need to be spun up
- We plan to include the support/mom/script nodes as well
  - The script nodes in CV1 are the "wild west" with no management at all
  - User scripts can do crazy things and significantly impact overall operation.
- Makes the "hooks" more explicit
  - User pre/post scripts or other ancillary processing can be added anywhere
  - CV2 will inject system pre/post scripts and similar into the job graph.

BILLION WAY SCHEDULING DECISIONS

How do we quickly and efficiently make billion way scheduling decisions?

- Bitmaps with time ranges
  - We have been using this internally on a smaller scale for a few years.
  - We use it to correlate events. Every component has its view of an event, we "or" the bitmaps that represent the affected resources and the time frames to capture the complete event.
- For scheduling
  - Imagine a set of bitmaps where each bitmap represents an attribute (up/down, available/in-use, cache-quad memory mode, etc) and each bit in the bitmap represents a unique resources (node, core, etc.)
  - Doing an "and" of all the required attributes returns a bitmap of available nodes.
  - Non-unique resource comparisons are done after the bitmap calculations reduce the search space.

COBALT INTERNAL SYSTEM REPRESENTATION

The CISR (“scissor”)

- At this point more of a concept. It may be a single data structure, a set of distributed data structures, etc..
- Based on the bitmaps with time ranges described previously. Every time an event alters state, a “copy” (likely copy on write deltas or similar) is made, the end time of the existing one is updated, and a new state with an unspecified end time is in place.
- What would be the benefits:
  - Can replay the history of the machine at will.
  - Metrics like availability, usage, and utilization are trivial and exact.
- What are the issues:
  - It takes a lot of space in memory
  - Writes could be an issue, CQRS helps, probably log based.

QUEUES AND TAGS AND FILTERS OH MY!

- We want the scheduler to be aware of every task; Rather than a script running 100 tasks, we would like those 100 tasks explicitly in the queue.
  - We know the machine better and we can be more efficient than they can
- Given the above, and the scale we expect, that means we need to support many millions of jobs in the queues.
- We plan to use a tag system similar to Gmail
  - There wont really be queues, just appropriate tags.
  - shardable / scalable
- What queue representations (tags) will there be?
  - User - A filter based on the user tag; Unlimited depth.
  - Project - A filter based on the project tag; Unlimited depth.
  - Resource – A filter based on the resource tag; Policy sets the depth
    - This is what we would think of today when we do a qstat.
  - other tags – User/scheduler can add an arbitrary set of additional tags

SPLIT SCHEDULING

- We want to enable the separation of the job description from the resource request
  - Put a million single node/core jobs in with user tags
  - Request 8K cores, but rather than a script, you specify a filter.
    - qsub –n 8192 –t 600 filter=[foo|bar|baz]
  - Mesos has a similar concept "You rented the space, do what you want"
- Why do this?
  - Scalability; Divide the machine and run many schedulers.
  - At the ALCF, larger requests get higher priority
  - We also limit the smallest allowed request size
- Extend the concept of the CISR
  - include a scheduler instance in the CISR and a filter
  - As far as the CISR is concerned its resources are the whole machine
  - You have 142 nodes that are going to sit idle for 27 minutes waiting to run a larger job; Create a CISR, assign a scheduler and optionally a filter and let it run.
  - Carry it to the logical extreme: Every Job has a CISR with a queue depth of 1

MULTI-OBJECTIVE (CO) SCHEDULING

We aren’t in “only cores matter” anymore Toto…

- Traditionally, the number of compute resources (nodes, cores, whatever level you were scheduling at) and wall time were the primary considerations in scheduling.
- The list of things to consider continues to grow: power, burst buffers, network links, remote RAM (ask me about this one offline if you are interested)
- We have been working with Dr. Zhiling Lan at IIT on this. Some reasonably good results with some genetic algorithms.
  - Scheduling includes policy and sociology. This one basically applies existing policy and then just tried to optimize the top N jobs for the other resources. It gives some improvements without totally tossing out existing policy.
- Another CISR idea: Arbitrarily divide the machine up into pieces to scale the scheduling and then co-schedule between them for larger jobs…

APIS AND SECURITY

- *ALL* functionality of CV2 will be accessible via APIs
  - Likely REST unless something changes our mind.
  - We would love for there to be a standard or at least defacto set of APIs, but people have tried that many times and all failed…
- You will be able to interact with CV2 via these APIs from outside the facility
  - Workflow engines, projects with a central work queue, etc..
- Obviously, that requires strong security
  - Likely a combination of OAuth2, OIDC, and some model for delegation for external services like Globus Transfer.
  - At ANL, unless policy changes, we will require you to use 2 Factor Authentication from our site.
  - We will design such that the security mechanism is pluggable and configurable, so a given site can use whatever meets their needs.

EVERYTHING IS DATA

- We ingest just about every piece of data our systems put out and then use them in our reporting systems.
  - Formats change, contents change, syslog loses messages, messages disappear or are combined, timestamps aren't to maximum resolution, they don't include timezones and on and on.
- We intend to treat our logs, messages, and similar the same way we treat the code.
  - They will be versioned, there will be tests written that parse them, etc.
- We intend to have a user accessible message bus so that other services can subscribe to events / messages.
- Every event / message must have a: unique identifier, version, source (what component produced it), timestamp in ISO format with TZ info to maximum system resolution, fully tested serialize and deserialize function, and a function to convert to a human readable format (think __str__)

RANDOM OTHER THOUGHTS

- What can we do to minimize the impact of future architectures (quantum, neuromorphic, things we haven't even thought of yet) on the scheduler?
  - Resource Independent Representation: Can we find an abstraction that lets us minimize/isolate the impact of the architecture on the scheduler, while still getting acceptable benefit from the new architecture?
  - The CISR concept is a first crack at this. It is how we hide the differences between the Blue Gene and the Cray from our reporting systems.
  - How do you minimize the impact on the users? (Blue Gene blocks vs nodes)
- Traditional HPC vs On-Demand Computing
  - Always keep some nodes available (the "full machine" isn't really the full machine).
  - Be able to do "context switches" with acceptable overhead
    - This becomes an I/O storage problem (have to put the state somewhere)
    - DMTCP project https://github.com/dmtcp/dmtcp
    - VeloC project https://github.com/ECP-VeloC/VELOC

RANDOM OTHER THOUGHTS

- How do we support malleable jobs?
  - Easy enough to imagine ways to do it
  - How do we avoid the “format wars” (every scheduler doing it differently)?
  - This is a variation of on-demand computing when requesting additional resources.
- Need support from the resource vendors
  - We are going to make sure *Cobalt* is not the bottleneck in our scalability goals, but if the vendor job launch is slow or falls over under load, it puts us between the Scylla and Charybdis:
    - Scylla: We don’t get the performance we want
    - Charybdis: We have to write the support ourselves which is a significant effort, could cause support problems, etc..

SIMULATION

This is a stretch goal, but will be very cool if we can pull it off

- Discrete Event Simulation… Hang On…
  - We said everything is an event…
  - Recovery is by event replay…
  - How is that different than simulation?
- It should be possible to replay a set of events with a different scheduling algorithm and see how they compare
  - We have to “speed up time”
  - Some events are input (user entering a job, nodes failing), some are output (a job can’t finish before it starts) so we will need to classify the events and treat them appropriately.
  - Guaranteed that the simulator is up to date since the scheduler is the simulator.
  - Any report you generate in production you can generate for comparison from the simulation.

TECHNOLOGY VS SOCIOLOGY AND POLITICS

- Scheduling is about implementing policy.
- Where there is policy, there is sociology and politics
- For instance
  - Leadership computing gives priority to big jobs, in particular full machine jobs that run for many hours
  - On-demand jobs need immediate access to resources
  - Q: Which one is more important? A: “Yes”
- There are a number of potential solutions to the above conflict, but everyone of them will require policy adjustments and expectation management as well as technology solutions.
- We also need the vendors to buy in or we will be limited by their SW stack.

WE NEED TO WORK JUST AS HARD ON THE POLICY AND VENDORS AS WE DO THE TECHNOLOGY, MAYBE HARDER.

SUMMARY

- We have some daunting, but very exciting challenges ahead of us in the scheduling and resource management arena.
- ALCF has some ideas about how to address some of those challenges and plan to do so via the open source process in collaboration with the PBS community and hopefully some of you.
- To be successful it will require
  - Technology – I believe the community can rise to that challenge
  - Policy and Funding support – We all need to be engaging with our respective policy makers and funders to be sure that the targets and timelines make sense. Policy and funding shifts require years of lead time.
  - Vendor support – Some of these challenges are dependent on the vendor software stack, which again can require years of lead time to impact.

THANK YOU!

QUESTIONS?

PROJECTS THIS BIG REQUIRE A LOT OF HELP. MY THANKS TO: LISA CHILDERS, ZACH NAULT, ERIC PERSHEY, PAUL RICH, GEORGE ROJAS, BRIAN TOONEN, MICHAEL ZHANG, AND MANY, MANY OTHERS.