When you have a hammer, everything looks like a nail

Checkpoint/restart in Slurm

Manuel Rodríguez, J.A. Moríñigo, R. Mayo-García

CIEMAT. Madrid, Spain

Outline

- What have we done?
- How have we done it?
- New cluster capabilities through Checkpoint/ Restart

What have we done?

- Provide support for DMTCP checkpoint library in Slurm
- Explore the functionalities of Slurm that get a boost with this new capabilities
- Create some new cool tools

Background: checkpoint/restart

- Checkpoint/restart: being able to save the status of a running job and restart it somewhere else
- Why is it useful/important?
  - Straightforward answer: provide fault tolerance
  - Not so straightforward usage: take scheduling decisions on running jobs.
    - Interrupt a running job and continue it in the future
    - Move a job from a particular resource to another

Software Stack

- Resource Manager: Slurm
  - (hopefully no need to describe)

- Checkpoint library: DMTCP
  - No need to recompile: useful for legacy & proprietary applications
  - User-level
  - Support for MPI and OpenMP
  - Robust and reliable

- MPI: tested with mvapich2 and OpenMPI
  - Any should work

How have we done it?

- Create a DMTCP driver
- Modify Slurm to provide some required new functionalities related to access to information
- Explore how to use C/R on different ways
  - Existing scheduling algorithms
  - Design of new ones
  - Implementation of new tools

DMTCP Driver

- Analog to the existing ones, such as BLCR
- Only checkpoints sbatch
- Add “--with-dmtcp=<path>” flags on compilation, “--no-dmtcp” to sbatch
- Some (questionable) decisions
  - Checkpoint images are deleted after successful execution
  - Saves all content of TMPDIR owned by job user
  - DMTCP support is enabled by default.

Slurm modifications

- Spank
  - New parameter S_JOB_CHECKPOINT_DIR so spank_get_item returns checkpoint folder
  - Creation of spank_set_item to modify parameters of the job.
    - At this moment, only “S_JOB_ARGV”
- Preemption (node_scheduler.c):
  - correction of a couple of bugs related to job preemption with checkpoint

New cluster capabilities through Checkpoint/ Restart

- Job Preemption
- Eternal jobs
- New Scheduling algorithm
- Job migration for cluster maintenance

Job Preemption

- Basic idea: when a job with high priority comes in, any job with lower priority gets temporarily out of the way
- Already existed, but was pretty limited as there was no universal checkpoint/restart library
- A simple approach:
  - 2 job queues in Slurm: high priority and low priority
  - Configure Slurm to use CHECKPOINT to remove jobs
  - Jobs are restarted in any resource, not on the one they were originally running

Use case: multi-tenant cluster

- My group owns a small cluster for scientific experiments
- Users groups with time-sensitive tasks prefer to own their own hardware
- Problems: administration issues, low usage
- Solution we are starting to implement
  - instead of buying machines, they give us the money :)
  - we get a set of nodes for our cluster that they “own”
  - their jobs have high priority + preemption on their nodes, low priority in the rest of the cluster
  - they have access to a wider set of resources with no drawbacks

Eternal jobs

- Born from a discussion with Berkeley Lab. Credit goes to Douglas Jacobsen
- Idea: very low priority jobs with a really long execution time
- Process
  - When the cluster is not fully used, their execution is started
  - When a job with normal priority comes, they are checkpointed and put back in the queue
- Pros: full usage of the cluster, satisfy demands of intensive users
- Cons: their execution can take very long. Only suitable for certain situations

New scheduling algorithms using C/R

- We are now able to move running tasks to different places.
- “Dummy” prototype already implemented: all the technical problems are solved
- Now we are exploring how to use this to increase performance and/or reduce energy consumption
  - Looks promising, but no results to be presented yet

Tool for system management: smigrate

- Scenario: you have to update whatever in a node, but there is a job running there. Even marking the node as “DRAINING”, it might take hours or days before the job ends.

- Our tool: mark node as “DRAINING”, checkpoint running jobs, move them somewhere else
  - flags for some decisions: what should we do with parallel jobs? and with non-checkpointable ones?

- Already implemented and working fine :)

Discussion, issues, and random thoughts

- Worst case scenario on preemption: low priority jobs spending their entire time being restarted/preempted because there are higher priority jobs entering the queue at an unfavorable interval
    - Minimal time that a job can run without being preempted again?
- TMPDIR
  - Saving it takes too long, not saving it leads to information loss
    - Approach 1: TMPDIR for every job, with prolog/epilog scripts
    - Approach 2: only save information of job owner

Discussion, issues, and random thoughts (2)

- Containers and statically linked applications are not supported by DMTCP, but CRIU does support them
  - We created a plugin to support CRIU in Slurm
  - We created another one that uses CRIU or DMTCP depending on the kind of job,
  - This all looks like over-engineering
  - Current solution is just waiting for DMTCP team to add support

- In preemption, jobs with checkpoint=disabled just get cancelled.
  - Do we want that, or to leave them running and just preempt those with checkpoint support?

Take a look!

https://github.com/supermanue/slurm

Thanks & Acknowledgements

- DMTCP team worked a lot on this integration
- Slurm mailing list
- Ulf and Maik from Technische Universität Dresden
- Funding:
  - COST Action NESUS (IC1305)
  - CODEC2 (TIN2015-63562-R)
  - EU H2020 project HPC4E (grant agreement n 689772)

Questions?