pawsey

Full-stack Approach to HPC Testing

Dr. Pascal Jahan Elahi & Craig Meyer
Pawsey Supercomputing Research Centre
CUG 24

Australian Government
NCRIS National Research Infrastructure for Australia
CSIRO
Curtin University
Murdoch University
Government of Western Australia
ECU
The University of Western Australia

Pawsey Supercomputing Research Centre

- Headquarters located in Perth, Western Australia
- Offers critical support to radioastronomy research around the Square Kilometre Array (SKA).
- Support uses in a large number of different of science domains with a wide variety of workflows.

Pawsey Supercomputing Research Centre

Setonix: HPE Cray EX

- Setonix is the scientific name for the friendly Quokka.
- 1500 AMD CPU (Milan) + 192 AMD GPU (Trento+MI250x).
  - 180 CPU nodes dedicated to radio astronomy operational processing for the ASKAP radio telescope.
- Slingshot system currently running Shasta.
- CPE 23.05 with ROCm 5.2.3 on production nodes.

Setonix (& Quokka)

Motivation

The scope of our tests are such as to

- Cover lower-level (less user facing) software infrastructure, like MPI, GPU
- Cover more user facing software infrastructure, like SLURM and Pawsey-deployed software that span the diversity of user workflows at Pawsey.
- Standard acceptance tests often failed to trigger issues encountered by users or did not provide diagnostic information to correct the issue.
  Examples
  - OSU Microbenchmarks does not cover the range of MPI communication patterns used by codes and bugs in libfabric went unnoticed till users ran production codes.
  - Shared node access by users running job packing on both CPU and GPU bugs exposed a number of SLURM bugs.

Design

- Use ReFrame to automate all these tests.
- Broader range of tests than in the initial acceptance scope.
- Diversity of tests to cover workflow diversity.
  - Ex: shared-node job packing, use of external job orchestration tools, IO heavy workloads
- Tests check sanity, benchmark and also provide extensive logging to diagnose issues.
  - Ex: MPI tests log node health before and after running MPI job by looking at kernel messages, available memory of node, etc.
- Expanded software stack testing integrated into our software deployment process.

MPI

- Critical tests of sanity and performance covering communication patterns used

SLURM

- Number of bugs and workarounds identified

GPU

- SLURM issues, ROCm issues, software compilation issues

Software Deployment

- Integrated testing of deployed software stack

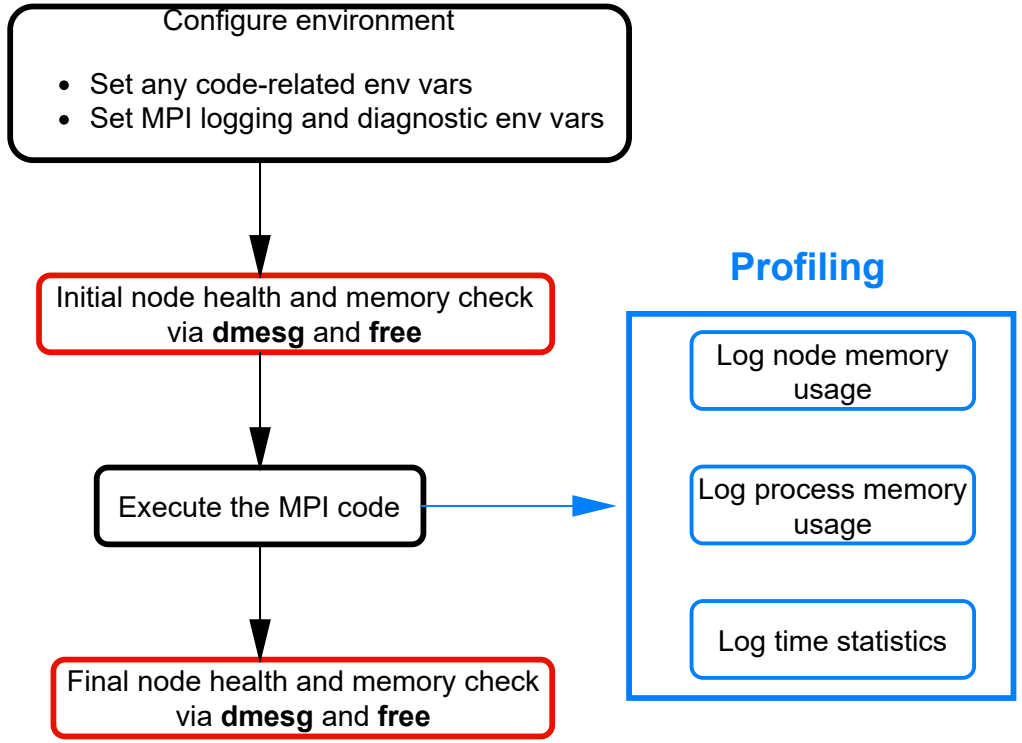
MPI

Initial Phase-1 Setonix users encountered numerous issues even though system was passing acceptance tests using HPL and some MPI-focused OSU Micro Benchmarks. Further testing indicated they occurred with MPI-enabled codes running at production scale, but error messages proved unhelpful.

Test Design

MPI errors were unusual enough and not limited to specific codes but rather any MPI job of a particular scale. To investigate, our tests critically

- Look at node health and provide diagnostic information before running MPI-enabled codes.
- Use complex communication patterns and check information sent is correct.



MPI

Issues

Workarounds and solutions

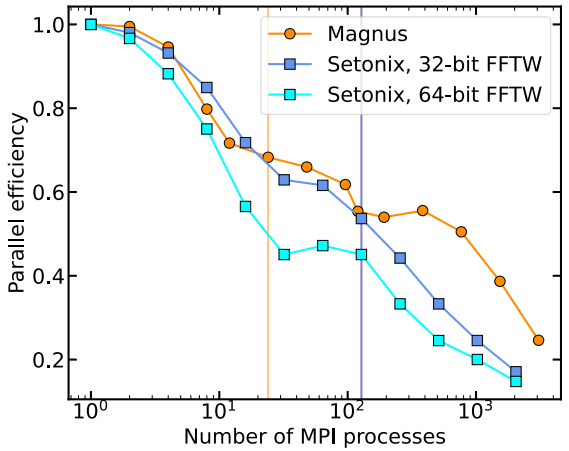
Memory

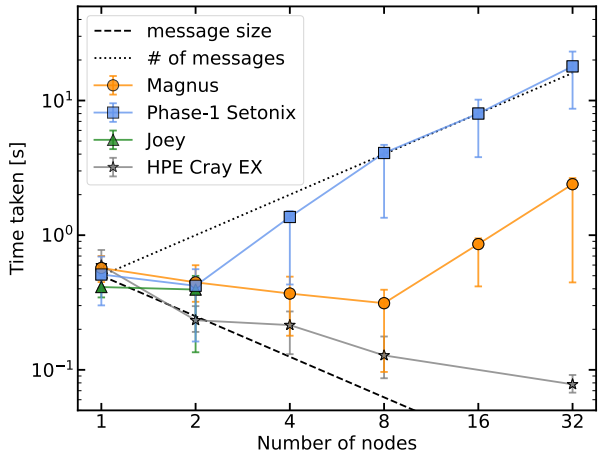
- Multi-node >64 rank MPI jobs with large messages (sent across nodes) were crashing with: bus errors; OOM; errors referencing xpmem library or OFI library.
- Nodes would slowly have decreasing amounts of available memory after running MPI jobs

- Workaround was to run codes with fewer MPI ranks (silly and not feasible in some cases)
- Solution was upgrading `libfabric` from 1.12 to 1.15.x which fixed memory leaks in this library. Regression test in place in case bug reoccurs.

Performance

- Several codes scaled poorly. Tests showed point-to-point communication scaling badly (worse than XC system)





- Workaround was to not use on-demand point-to-point communication by setting `MPICH_OFI_STARTUP_CONNECT=1`
- Solution was upgrading `libfabric` from 1.12 to 1.15.x

SLURM

Our SLURM deployment and tests we developed to either track or aid in diagnosing/- fixing issues encountered with vendor-provided version of SLURM but are useful for general regression testing.

We run wide variety of tests: multi-node, single node, exclusive, shared, job packing, job arrays, MPI, threads, memory-focused resource requests, CPU-focused resource requests, GPU node tests, GPU-CPU affinity tests.

Issues

Memory Calculations

- Some SLURM jobs failed, reporting *invalid node specification*, despite requesting valid resources.
- Testing showed the automatic memory calculation was incorrect when using `DefMemPerCPU` if no explicit memory was set and enough MPI processes per node were requested.

Billing

- Accounts billed incorrectly for requested resources. Billing based on CPUs requested or equivalent number of CPUS if there is explicit memory request using `DefMemPerCPU`.
- Tests showed TRES incorrect when jobs used hyperthreading and where CPUs from explicit memory request > than the explicit CPU request. Account billed the smaller amount.

SLURM

Issues

Affinity

- CPU affinity impacts performance. Benchmarks showed odd performance, seemingly related to affinity.
- Unit test reporting thread-level affinity (e.g. OMP) and task-level affinity (e.g. binding MPI processes to CPUs) showed consistent optimal affinity when using exclusive node access. Shared access generated unusual changes in binding, generally not optimal (hopping across sockets, filling up an L3 cache in each socket before moving onto the next L3 cache in each socket.)

Configuration

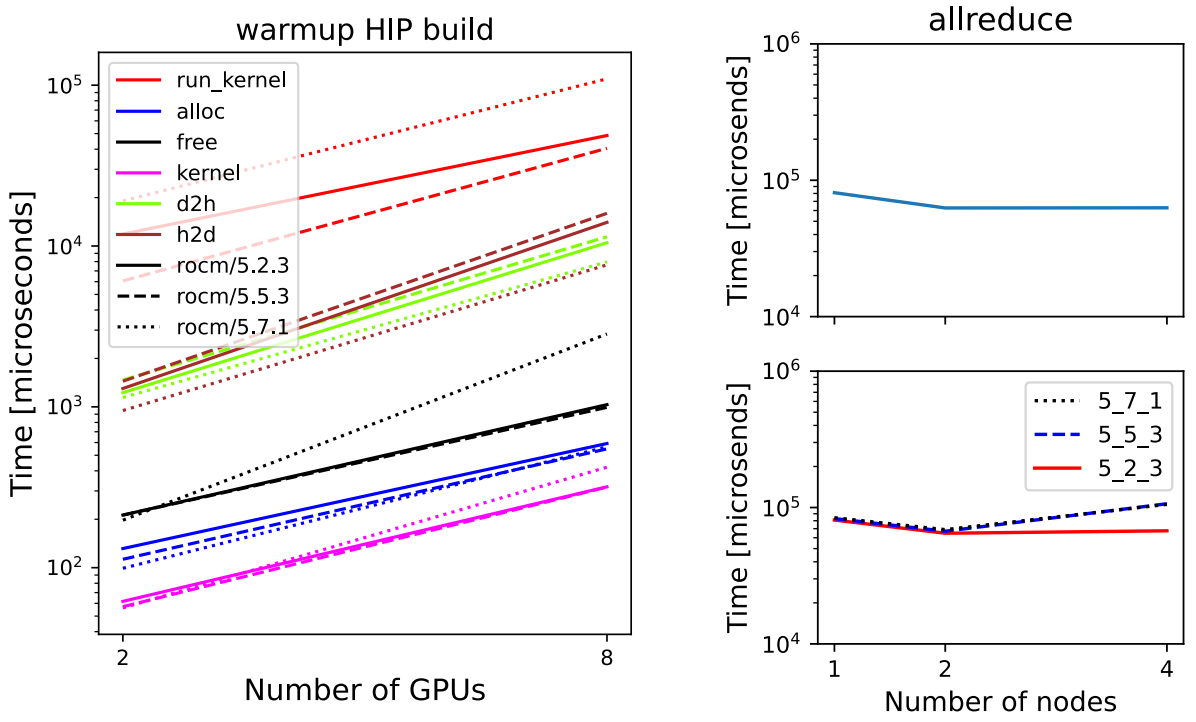
- Encountered issue where job-packing on GPU-nodes would suddenly fail (to run concurrently).
- Testing of job-packing and reporting of SLURM configuration as part of diagnostic showed SLURM configuration reverting to a different value without notice or apparent cause on active nodes running jobs.
- *Very worrying* given no notifications and impacts on jobs.
- Currently, when seen, *we manually reset the configuration*.

GPU

GPU tests (HIP/CUDA, OpenMP, OpenACC) which check functionality and performance of common GPU operations (memory allocation/ deallocation, host-to-device & device-to-host data transfers) GPU-direct MPI communication.

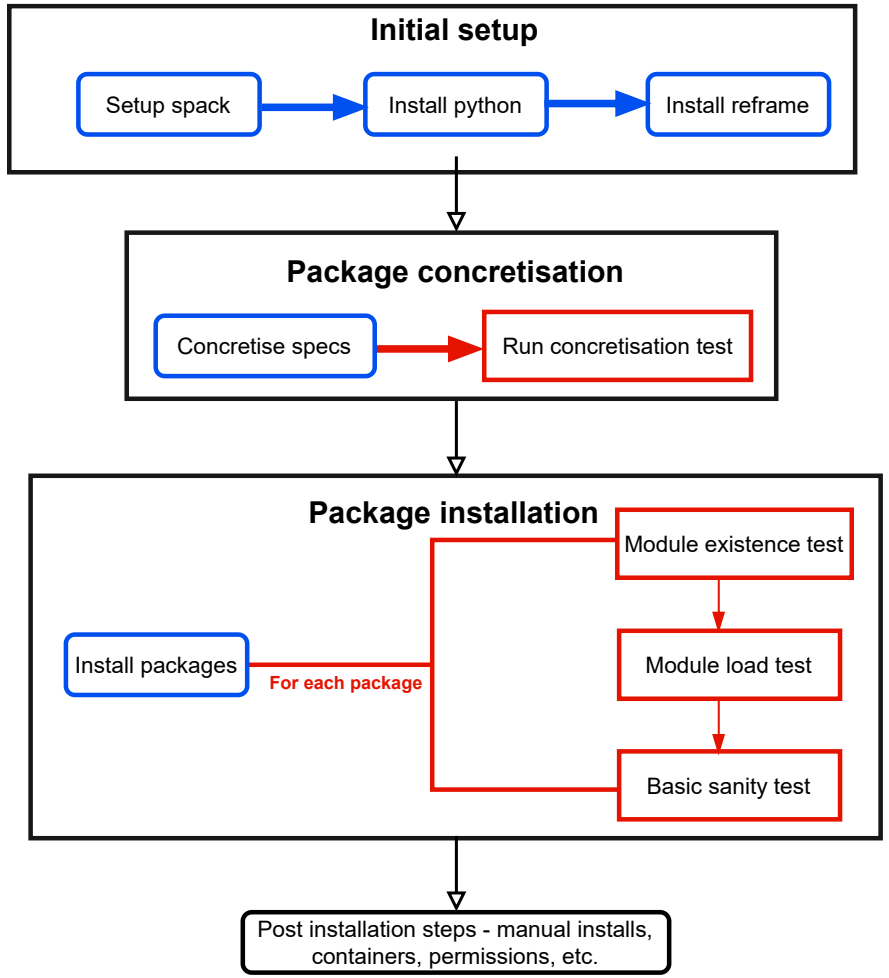
They serve as long-term performance monitoring, although GPU-MPI is an additional MPI test.

- Using tests, we can check performance of ROCm versions on operations.
- These can be ROCm in a container.
- Increasing ROCm versions do not give steady performance improvements.
- Newer ROCm versions require newer Cray-MPICH and so tests of >5.5 were run on node with CPE 23.09



Software Stack Tests

- Pawsey uses `Lmod` modules, `Spack` package manager, SHPC container manager and custom scripts to deploy software stacks to our users.
- Our tests focus on checking that software has been built (by `Spack`), an appropriate module has been constructed, and that the library or executable function at the most fundamental level.
  - Tests use `ldd` to check paths, run executable with `--help`/`--version` or equivalent and some executables have specific tests running the software at scale on compute nodes.
- Tests integrated into our software stack installation pipeline such that they are automatically run for all software and libraries at appropriate stages and results can be quickly viewed once installation has completed.



Software Stack Tests

Issues

Workarounds and solutions

Missing modules

- Occasionally Spack did not generate module files for dependencies of a requested installation.
- Due to the use of `rpaths` and the lack of explicit LOAD statement in modules for many executables, no errors are produced, and the basic sanity check test and any dedicated software tests would pass.
- Does impact Python modules due to PYTHONPATH not being updated, resulting in basic sanity test failures.

- Workaround was to explicitly build module.
- There is currently no solution as this odd error has been with several different versions of spack and diagnostics provide no clue why some modules are not produced.

"Slow" GPU Performance

- Some GPU-enabled codes surprisingly slow.
- By looking at the GPU activity when running code, we found codes were not actually offloading to GPU despite Spack recipe attempting to enable offloading.

- No workaround.
- Solution was to investigate recipe, updating recipe or creating custom script for build that would enable GPU offloading.

Current State of Affairs

The current state of our tests are:

- MPI, GPU, SLURM tests integrated into our broader acceptance test suite
- Some compilation tests separate and used in targeted investigative cases
- Software stack tests integrated into our software stack installation pipeline
- MPI, GPU, SLURM, compilation, and software stack tests available in a public github repo
  - Available at https://github.com/PawseySC/Reframe-MPI-Stress-Tests
  - Separate from our internal versions of tests designed specifically for our system(s)
  - Test structure and logic modified to be run more readily on external systems

Future work

MPI

- Add more GPU MPI tests, Remote Memory Access.

SLURM

- Add more tests for complex workflows in conjunction with job orchestration tools (e.g., Prefect, Nextflow, Airflow).

GPU

- Add more tests of reference kernels to check for performance/compilation regressions.
- Add the ROCm unit tests (those that work).
- Add more pragma offloading tests.

Software Deployment

- Improve integration and generalization

Filesystem for workflows

- Add test covering workflows have complex IO footprint (large + small files, less than optimal read/write patterns)

Containers

- Add tests to check functionality of MPI and ROCm in a container.
- Add tests to cover varied containers/squashfs usage and multiple Singularity modules.

Questions?

Full-stack HPC tests