Node-local IO on Aurora - CPPR

Christopher Holguin | contributors: Kalyana Chadalavada, Jeffrey Olivier, John Carrier | Intel Federal, LLC. | Special thanks to Eric Barton for his contributions.

Problem/Opportunity

Aurora will have >7 PB of DRAM and persistent memory, a burst buffer, and a Lustre filesystem. How can Intel and Argonne help application developers take best advantage of this unique IO stack?

Solution

- Asynchronous file movement services, utilities, and a local namespace for applications running on the compute nodes
- Checkpointing and restart services optimized for Aurora
- Customized and scalable shared file loading

Project Overview

Common **P**ersistent-memory **P**OSIX **R**untime (CPPR) enables applications to leverage the fast persistent memory available on Aurora compute nodes (CNs) for I/O which enables more efficient use of compute cores. CPPR comprises 3 components - 1) async API for developers to utilize CPPR services 2) Compute node session services (CNSS) daemon 3) file movement utilities. Using CPPR, applications can 1) take advantage of the node-local (NL) filesystem backed by persistent memory hardware to reduce interaction with the global filesystem, 2) move files into and out of the NL filesystem asynchronously, and 3) create fault-tolerant checkpoint data with SCR and FTI, which utilize CPPR services.

A. CPPR Components

Async APIs (pseudo code)

- cppr_mv(handle, src, dst, args)
- cppr_grp_create(handle, cookie, args)
- cppr_grp_create(handle, flagpath, num)
- cppr_grp_commit(handle)
- cppr_wait(handle)
- cppr_test(handle)

CNSS plugin

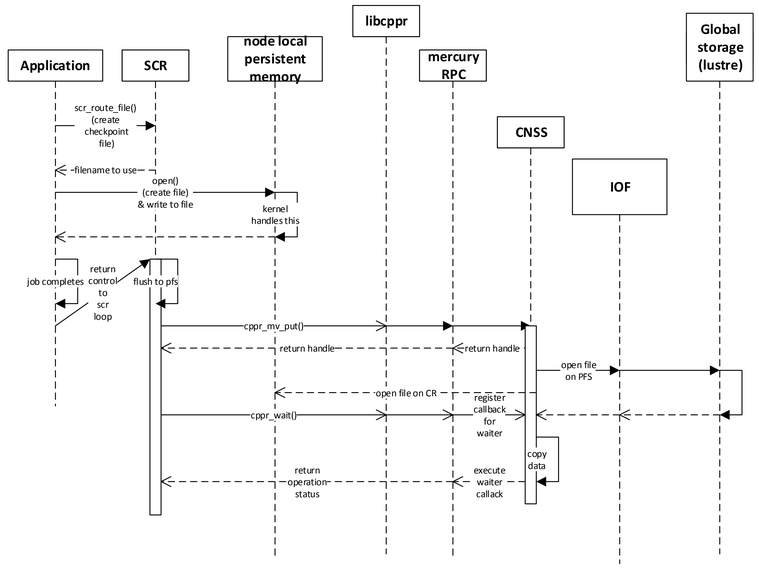
- libcppr API calls are function shipped using Mercury RPCs to this daemon, which enables the IO to be asynchronous
- Implements all the file movement services
- Runs as a part of a daemon process which contains other compute node services

File movement tools and utilities

- Python API
- Command line mover utility program
- A node local POSIX namespace

B.Checkpoint, Restart, and Redundancy

- SCR and FTI are libraries which make it easy to manage checkpoint data, and ensure it is resilient to node local storage failure/data loss. Both will be available on Aurora.
- CPPR enables SCR and FTI to offload IO to the CNSS with no need for their own mover services



C. Scalable File Copying

- Common use case: loading shared library files at the beginning of a session
- Contains both and API and a FUSE mount
  - "Cooperative Cache" namespace: allows access to this feature with no code modifications; just prepend PATH and/or LD_LIBRARY_PATH with the path to the CC mount
  - API – allows programmatic access
- Expands upon ideas from SPINDLE

Step 1

{ api }

User calls cppr_mv_cget

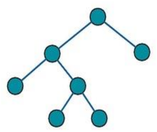
Args: handle, local_path, global_path

Step 2

lustre File System

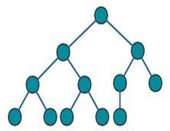
Nodes are mapped to a tree; File is fetched from Lustre by root node

Step 3



File is pulled down the tree from the node-local cache as each node requests it

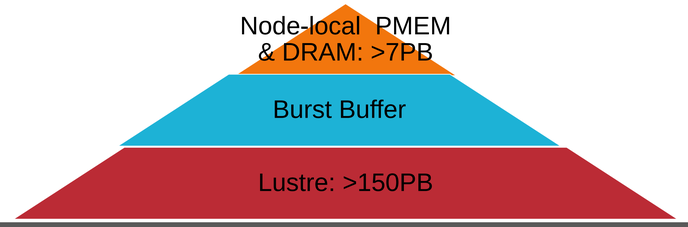
Step 4



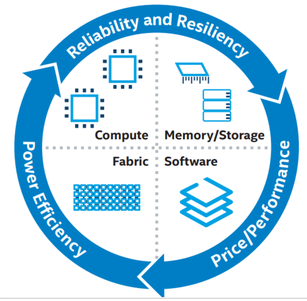
Each compute node contains the file; **only one request went to Lustre**

D. Memory Hierarchy

- Storage tiers on Aurora



Beyond Aurora



- Intel Scalable System Framework (SSF) – Tightly integrated components for HPC and other workloads
- CPPR will become a part of Intel SSF to enable future supercomputers have the same capability

Conclusion

- CPPR will enable users to come up with their own use cases for persistent memory.
- CPPR is providing a common framework to facilitate use of persistent memory by simplifying the transfer of data between node local mem. and the PFS
- CPPR to be delivered in December 2017, with functional demos in December 2016. As such, results are not yet available.
- CPPR will be open source, and changes made to SCR and FTI will be upstreamed to their public repos
  - CPPR sources available http://git.whamcloud.com/coral/cppr.git

Works Cited

- http://www.intel.com/content/www/us/en/high-performance-computing/holistic-solution-for-hpc-infrastructure-solution-brief.html
- http://www.intel.com/content/www/us/en/high-performance-computing/path-to-aurora.html
- http://aurora.alcf.anl.gov/
- https://github.com/hpc/scr
- https://github.com/leobago/fti
- https://mercury-hpc.github.io
- D. Ahn, M. LeGendre, T. Gamblin, B. de Suspinski, F. Wolf and W. Frings, "Massively Parallel Loading," in *27th International Conference on Supercomputing*, Eugene, OR, USA, June 2013.