Comparing Decoupled I/O Kernels versus Real Traces in the I/O Analysis of the HACC Scientific Application on Large-Scale Systems

Student: Sean McDaniel, University of Delaware

Advisors: Hai Ah Nam, Oak Ridge National Laboratory

Synthetic benchmarks have been found inadequate in comparison to real-world applications as large scale end-to-end system stressors. HPC I/O benchmarking solutions IOR and IOZone have NOT provided the end-to-end stressors that are needed to test the file system stability, particularly at large scale. Using real-world applications for testing and tuning is not always feasible because of their long runtimes and the short testing windows on production systems. The lack in adequate solutions for stressing systems has increased efforts to extract I/O kernels from scientific applications. However, there does not exist a successful methodology for extracting I/O and communication codes from scientific applications .

GOAL

- Validate the decoupled I/O kernel of the Hardware Accelerated Cosmology Code application (HACC-IO) to determine whether it reliably mimics the I/O patterns of the real HACC application
- HACC-IO patterns can facilitate and support future I/O kernel extraction and tuning

Hardware Accelerated Cosmology Code (HACC)

HACC uses N-body methods to simulate the formation of structures under the influence of gravity in an expanding universe. HACC and HACC-IO (a decoupled I/O kernel) are provided as CORAL benchmark test applications . The CORAL benchmarks are used to guide the procurement for pre-exascale computing resources.

HACC vs. HACC-IO

HACC Time Step Work Flow

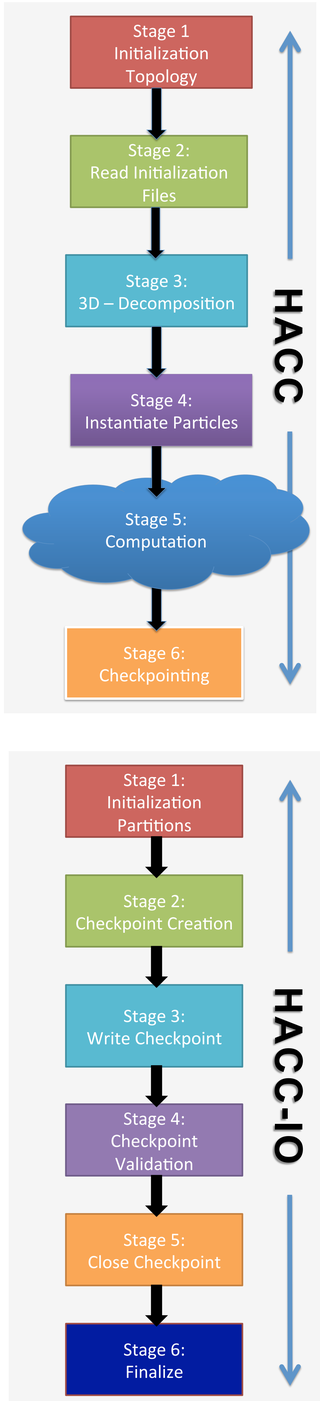
- Stage 1: Initialize Topology
  - Duplicates communicator
  - Create Partitions
- Stage 2: Read input data file
  - Specifies initial parameters
- Stage 3: Initialize geometry
  - 3D decomposition of Box
- Stage 4: Instantiate Particles
- Stage 5: Computation
  - Host side Long range FFT calculations
  - Architecture specific short range calculations run on GPU
- Stage 6: Write restart file

The purpose of HACC-IO is to evaluate the performance of the I/O system separately from HACC. It is intended to captures the I/O pattern of HACC. HACC-IO shows

- A known data pattern is written out as a checkpoint
- The checkpoint data of the known pattern is read back in and verified with the known pattern

HACC-IO Time Step Work Flow

- Stage 1: Initialize
  - Duplicates communication
  - Create Partitions
- Stage 2: Create Checkpoint
  - MPI Allreduce on particles
  - Stage 3: Write Checkpoint
  - Write using POSIX I/O
- Stage 4: Read Checkpoint
  - Read back to verify content
- Stage 5: Close Checkpoint
- Stage 6: Finalize
  - Destroy partitions
  - Destroy global communicator



METHODOLOGY

- Use Darshan I/O tool to study the I/O pattern and file access for both HACC and HACC-IO
  - Darshan instrumentation is inserted at build time and captures the behavior of both POSIX I/O and MPI-IO methods
- Use Vampire to analyze MPI communication in HACC and HACC-IO when running on up to 2048 nodes of Titan

Darshan provides a big picture of application I/O

VampirTrace analyzes network activity closer at a more granular level

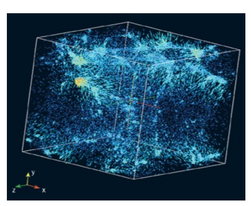
Example of VampirTrace Function Time – HACC-IO

Metrics

- Metrics of interest with Darshan:
  - Checkpoint write time: time spent writing restart files during the life time of our HACC run
  - Average Checkpointing Output: data size in relation to number of processes
  - Writes, Opens, and Seeks Times: number of writes open and seeks performed by HACC and HACC-I/O when writing restart files
  - Seq., Consec., and Total Writes: number of sequential, consecutive and total writes performed by HACC and HACC-I/O
- Metrics of interest with VampirTrace:
  - Accumulated exclusive time per function
  - Communication matrix of process message passing

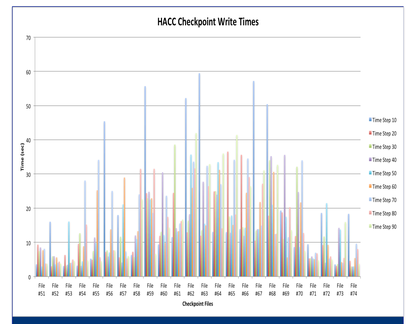
Tests and Platform Setup

- We use the Gordon Bell Prized scaling test to analyze HACC and HACC-I/O patterns on:
  - 128 Nodes - 4.096 x 10^9 Particles
  - 256 Nodes - 8.590 x 10^9 Particles
  - 512 Nodes - 1.6778 x 10^{10} Particles
  - 1024 Nodes - 3.2768 x 10^{10} Particles
  - 2048 Nodes - 6.8719 x 10^{10} Particles

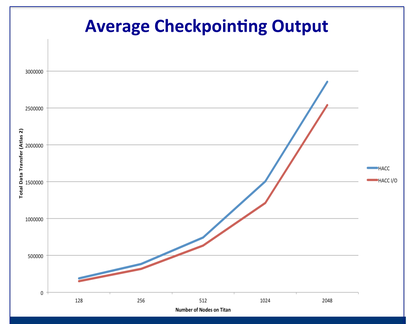


- We run the tests on Titan with this simulation setting:
  - Number of cores: 128 up to 2048
  - 1 MPI processes per node and 16 OpenMP threads per process

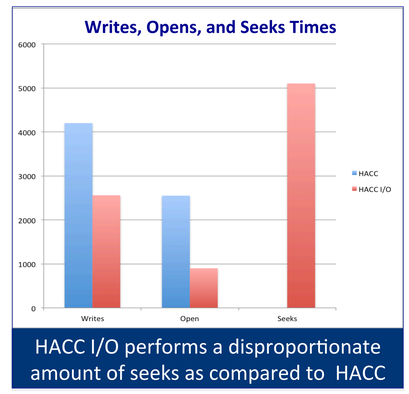
RESULTS



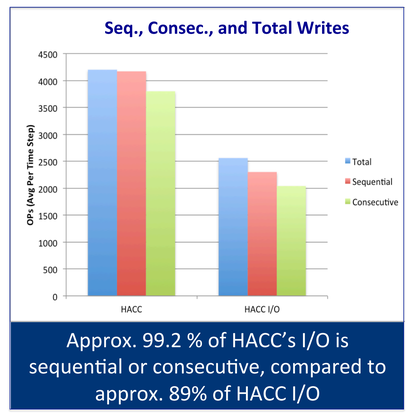
Number of tasks accessing checkpoint varies by time steps
Different checkpoint times across checkpoint files



HACC I/O average output data per checkpoint has similar HACC's trends



HACC I/O performs a disproportionate amount of seeks as compared to HACC



Approx. 99.2 % of HACC's I/O is sequential or consecutive, compared to approx. 89% of HACC I/O

Analysis

- Similarities between HACC and HACC-IO:
  - HACC-IO and HACC show comparable average checkpoint data output
- Discrepancies between HACC and HACC-IO:
  - HACC-IO lacks in sequential and consecutive I/O, causing it to perform more seeks than HACC
  - HACC-IO average output file size is not reflective of HACC output files
- Reasoning for discrepancies:
  - HACC-IO lacks an effective partitioning scheme and does not include aggregation
  - Adding improved partitioning scheme is work in progress

CONCLUSION AND FUTURE WORK

- We critically compare and contrast HACC-IO versus HACC and show similarities and discrepancies in their I/O patterns
- HACC-IO falls short in mimicking data output per file and I/O patterns
- The discrepancies provide new research opportunities:
  - Correct HACC-I/O to reliably mimicking HACC
  - Study I/O aspects of HACC (e.g., aggregation) by using HACC-I/O

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific computing Research and performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility. I would like to lastly thank Dr. Michela Taufer for all her guidance and mentorship.