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

Sean McDaniel
University of Delaware
seanmcd@udel.edu

ABSTRACT

Synthetic benchmarks have been found inadequate in comparison to real-world applications as large-scale end-to-end system stressors. However running these real-world applications on machines like Titan require increased resources and time.

In this work we take two applications, the first being HACC I/O, a benchmark whose purpose is to evaluate the performance of the I/O system for the Hardware Accelerated Cosmology Code (HACC) simulation. The second being HACC itself. We use tools Darshan and VampirTrace to characterize the I/O and network activity of these applications. We show that though HACC I/O does compare in certain areas to HACC it falls short in terms of I/O pattern and actual data output.

Results presented in the poster show our findings from characterizing the I/O of HACC and HACC I/O.

1. MOTIVATION AND GOALS:

As supercomputers move closer to exascale, the complexities of these systems increase. Synthetic benchmarks are used to test and diagnose bottlenecks and areas of improvement in these systems. But, synthetic benchmarks fail to serve as end-to-end system stressors. Accurately stressing a system can only be done under the strain of real-world applications. However these real-world applications take increased time and resources. We propose that a better solution to synthetic benchmarks would be to extract the I/O and communication codes from scientific applications and use those as end-to-end system stressors.

Our goal is to extract the I/O and communication from HACC and use that to build a stand-alone binary to be used a benchmark. This poster shows the work we have done in first verifying and validating HACC I/O to HACC but also characterizing the I/O activity of HACC.

2. BACKGROUND

The HACC (Hybrid/Hardware Accelerated Cosmology Code) framework is used to simulate large bodies of particles to study dark matter and dark energy. It's a CORAL application and HACC was a 2012 finalist for the IEEE/ACM Gordon Bell prize in scientific high-performance computing.

Darshan is a runtime library used for characterization of application. Darshan instrumentation is inserted at build time, and captures POSIX I/O, MPI-IO, and limited HDF5 and PNetCDF functions. VampirTrace is an open source library that logs detailed program execution of parallel applications using message passing (MPI) and threads (OpenMP, Pthreads). VampirTrace is also capable of tracing GPU accelerated applications and generates time stamps for all GPU related events.

3. METHODOLOGY

The contributions of this poster are an increase in knowledge of the I/O and network activity of HACC. This knowledge will lay the groundwork for a future goal of attempting to extract the I/O and communication from HACC.

3.1 Darshan

We first use Darshan to get an accurate picture of the I/O behavior of HACC and HACC I/O; we looked at multiple areas of I/O activity. Areas of interest consisted of average I/O cost per process, I/O operation counts, the overall I/O pattern, and timespan of write access on independent files. We then use VampirTrace on areas where we identify as interesting.

3.2 VampirTrace

We use VampirTrace to analyze the network activity of HACC and HACC IO. VampirTrace allows us to instrument and record the behavior of sequential, MPI-parallel and hybrid parallel applications. We look at the timeline of execution of HACC and HACC I/O at statistics such as time consumed, number of invocations, and message statistics.

4. RESULTS

4.1 Tests and Platform Setup

We compared and contrasted the I/O of both HACC and HACC I/O on different scales. Our experiments were conducted on Titan at ORNL using a Lustre file system, Spider. Each simulation had 1 MPI processes per node with 16 OpenMP threads per MPI process. Due to the overhead of VampirTrace, we ran on a smaller number of cores, 128 up to 2048.

Our simulations were weak scaling. The input particles sizes that we used were the same ones used by HACC in the Gordon Bell Prize competition. These consisted of 4.096 x 10 ^ 9 on 128 nodes to 6.872 x 10 ^ 10 running on 2048 nodes.

4.2 Benchmark Results

HACC and HACC I/O both write out similar amounts of checkpoint data. At 128 nodes HACC and HACC I/O writes out 187 GB and 151 GB of data respectively, whereas at 2048 we see that HACC writes out 2.8 TB and HACC I/O writes out 2.5 TB.

5. CONCLUSION AND FUTURE WORK

In this poster, we've shown how HACC I/O compares to HACC in regards to I/O behavior. We have seen that HACC I/O does compare accurately to HACC in average checkpoint size. However it differs in I/O pattern. Through our traces we have seen that HACC writes are roughly 99% consecutive or sequential where as HACC I/O's writes are only 89%. This leads HACC I/O to perform more seeks when writing out checkpoint files.

We also found the number of checkpoint files of particular interest. For 256 nodes and 8.589x10^9 particles 23 unique restart files were created. Over those files we saw a normal distribution among writers.

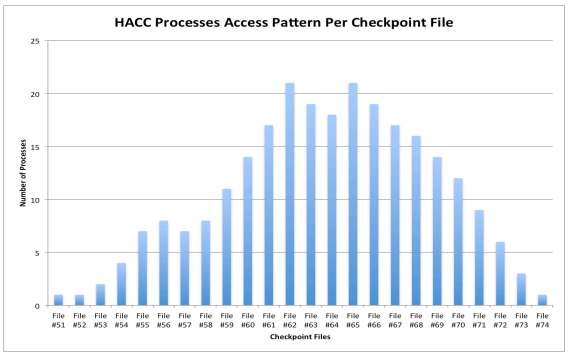

Figure 1. Distribution of writers across checkpoint files with 256 processes and 8.589 x 10^10 particles

Future work consists of further characterization of HACC in terms of I/O and communication using Darshan and VampirTrace. This data will be used to created a kernel of HACC in a standalone binary form.

6. ACKNOWLEDGMENTS

Thanks to ORNL, HERE, and OLCF for the resources needed to perform this research. Many thanks to Dr. Oral Sarp, Dr. Hai ah Nam, and Katrin Heitman for their help in integrating and testing HACC and HACC I/O.

7. REFERENCES

[1] Finkel et al., Cosmic Structure Probes of the Dark Universe (Porting and Tuning HACC on Mira) *ALCF-2 Early Science Program Technical Report* 2013

[2] Habib et al., HACC: Extreme Scaling and Performance Across Diverse Architectures Paper presented at SC13, Denver, Colorado