Comprehensive Burst Buffer Evaluation

Eugen Betke*, Julian Kunkel*
*Deutsches Klimarechenzentrum GmbH

***Abstract*—The German Climate Compute Center provides climate researchers with the necessary storage and compute capabilities to run large scale climate simulations, which tend to experience bursty I/O as snapshots are written. Burst buffers offer high throughput and low latencies, but how to best integrate into data centers is not fully understood. This work is studying the performance and application of different burst buffer solutions with a particular focus on climate applications. In our research we investigate characteristics of different burst buffer technologies and evaluate alternative ways of use.**

I. Challenges

A. Short-term campaign storage space

While burst buffers are mainly used for catching I/O peaks and may use different policies for de-staging data, the true benefit emerges when using them to manage and process data for random tasks in work flows like in-situ visualization and post-processing of known data products. Therefore, we believe the true benefit of utilizing fast storage comes when adapting work flows. For example, burst buffers (or other fast storage systems) can be used in an alternative way, e.g. ordinary scratch space. This is motivated by the fact that simulations often create data (e.g. intermediate results or snapshots) that losses its value after the simulation is finished and can be deleted. Additionally, such a system would catch random I/O to a shared file much better than main storage. Particularly, this is useful for typical climate data file formats such as NetCDF3 and 4. Typically, simulations produce only a small portion of data that needs to be stored on a long-term storage.

B. Memory vs. remote swap file system

In this study we investigate the system behavior which is equipped with a moderate amount of memory, but has access to large swap file systems deployed on fast storage device. In our workload manager logs we could see that the most jobs on our HPC are using only a fraction of the available memory, and only a few jobs had high memory requirements. Procuring a system with less, but sufficient memory for the most jobs, and providing additional memory for the few memory intensive jobs in form of remote swap file system, may significantly reduce the total system costs. The question is how much memory and swap space is optimal for compute nodes? We hope to find an answer to this question and create a cost model.

II. Approach

We first explore the benefit for climate data access patterns using the NetCDF API. NetCDF-Bench [1] is a native NetCDF parallel I/O benchmark that mimics I/O behavior of scientific applications. With IOR we measure the peak MPI-IO performance that can be achieved with individual and shared I/O.

Our cooperation partners provided us access to their burst buffer systems.

Kove XPD [2] is a in-memory storage that offers high throughput and low latency. The integrated UPS and multiple modes for de-staging data from volatile memory onto persistent media allows a usage as a persistent storage. The direct access over the proprietary KDSA interface is the fastest way to communicate with the XPDs. Based on this interface we implemented an MPI-IO driver and run IOR and NetCDF-Bench benchmarks. The work is mostly done.

DDN IME [3] is a SSD-based storage cache with high performance I/O characteristics and is our current object of investigation. On the test system, IMEs can be accessed through the IME-fuse file system. DDN also developed an experimental MPI-IO driver with direct access to IME. We will run similar benchmarks as we did on Kove XPDs to make both systems comparable.

Cray DataWarp [4] is another SSD-based burst buffer technology we will likely evaluate.

Along with the evaluation of the single burst buffer technologies we investigate the impact of memory reduction on compute nodes. For that purpose, we developed a Linux swap monitoring tool which uses the Linux's kprobe debugging mechanism for tracing calls to the swap in/out kernel functions. Firstly, we run a customized STREAM benchmark [5] on a test system with sufficient memory and no swap file system in order to obtain the best performance values. Then, we reduce the amount of memory by creating a large ram disk, create a swap file system on that ram disk, and repeat the benchmark. This configuration forces the operation system to swap memory when STREAM tries to allocate more memory than available. In this way we can see overhead of Linux swap and can observe the worst case behavior. This information is only a milestone on the way to the cost model.

In the next steps we are going to run benchmarks with a more realistic workloads and to create the swap file system on fast remote storage device.

References

[1] "Netcdf-bench," https://github.com/joobog/netcdf-bench, 2017-08-25.
[2] Kove, "Kove xpd," http://kove.net/downloads/Kove-XPD-L3-datasheet.pdf, 2017, 2017-08-24.
[3] D. Storage, *Burst buffer & beyond; I/O & Application Acceleration Technology*, https://www.ddn.com/download/resource_library/brochures/technology/IME_FAQ.pdf, DDN Storage, 9 2015.
[4] C. Inc., *Cray XC40 DataWarp's applications I/O accelerator*, http://www.cray.com/sites/default/files/resources/CrayXC40-DataWarp.pdf, Cray Inc., Cray Inc. 901 Fifth Avenue, Suite 1000 Seattle, WA 98164, 10 2015.
[5] "STREAM," https://github.com/jeffhammond/STREAM, 2017-08-25.