Cambridge Data Accelerator

Alasdair King {ajk203@cam.ac.uk}, Dr Paul Calleja {pjc82@cam.ac.uk}
University of Cambridge, Research Computing Services

***Abstract*— An overview of the Cambridge Data Accelerator, a open-source project providing on-demand file-systems, building on existing work in the Slurm workload scheduler, and is built using Dell EMC and Intel hardware. The hardware has been in testing since June 2018, and has allowed the team at the University of Cambridge to detail some of the problems and solutions in building complex flash storage. The system has been tested and produces the 500 GiB/s reads, and close to 300 GiB/s writes. Continuing work at the University of Cambridge on stabilising network faults that limit performance and file-system integrity; as well as patches to the prospective file-systems tested - Lustre and BeeGFS currently the main targets. Systems in the UK and in Australia have made use of this work as the bases for there own implementations. This software, that will be used in production at Cambridge, and its results will be made open to all. Follow up work will include the impact of this system on the I/O performance of UK science applications. This work has also helped in the use of SSDs in the Square Kilometer Array compute components.**

I. INTRODUCTION

With the commercialisation of burst buffer systems by Cray Inc. (DataWarp) and other storage providers, there is a requirement for a flexible, open solution for all. While this solution facilitates a similar function to these proprietary systems, this work is not a direct comparison and does not include, or leverage, any of the former's technology; with the exception of the Slurm Cray Burst Buffer plugin which is open-source under GPL2. The Data Accelerator project provides ephemeral per job file-systems. This is accomplished by a special program that orchestrates the resources. This program is controlled by the Slurm plugin and builds the chosen parallel file system over the nodes with NVMe devices. Files can then be stage in from any other source and copied into the device. However this is not the only way to make use of the system, as the system supports multiple workflows than a simple check-pointing and restarting.

Burst Buffers were developed by Los Alamos National Lab (LANL) and the San Diego Supercomputing Center (SDSC) for check-pointing and restarting in their large simulations. For these applications, one is check pointing multiple Terabytes of state to a parallel file system. At LANL the Parallel Log File System (PLFS) was used as the underlying file system for their implementation on fast disk or commodity SSD disks mounted on the nodes, or directly connected to the network. This allowed them to greatly enhance the time to solution by reducing the amount of time a simulation spent in I/O. To provide this level of improvement using standard scratch file systems such as Lustre, massive arrays of disks are used and can be x10 more in procurement costs than a SSD based solution for a similar performance.

II. SEMANTICS AND WORK FLOWS FOR BURST BUFFERS

A. Semantics

Using the example of LANL's Burst Buffer, the system is designed to provide a destination for user code to place checkpoints of memory state in a temporary caches that is several times larger than the memory in use by the program. In the case of petascale supercomputing, this can be several Terabytes of required space, which necessitates improved performance that the existing pool of parallel storage is often unable to provide. As this project builds on this work and extends the functionality, the term 'Burst Buffer' no longer seems adequate to describe the system. The term Burst Buffer is renamed Data Accelerator or DAC for short.

B. DAC Workflows

Glen Lockwood from NERSC at Berkeley National Lab, describes in a blog post the taxonomy for Data Accelerators based on his experiences with DataWarp on Corri. [1]

- **Stage In and Out**: This is the primary use case for users of the DAC. Data will be copied from a second source to the DAC. This could be a home directory or a parallel file system.
- **Transparent Caching**: An advanced feature allowing the fast tier to cache files and transparently drain back to the main parallel file system
- **Checkpoint Restart**: A single large namespace to place checkpoints from programs.
- **Journaling**: Programs can use the fast tier as a place to dump deltas of program output that can be replayed at a later date, reducing the amount of time a compute nodes spends in I/O.
- **Swap**: Mounted as NVMe over Fabric the device can be used to provide a memory extension for non deterministic memory requirements.

This project focuses on, stage in/out, checkpoint and swap workflows with other possible usage to be investigated. Staging in and out as well as checkpointing are seen as the most common use cases for applications that wish to make use of the Data Accelerators improved performance. Should the Data Accelerator make the Swap workflow achievable, this would allow shared memory programs to be used to grater affect at Cambridge.

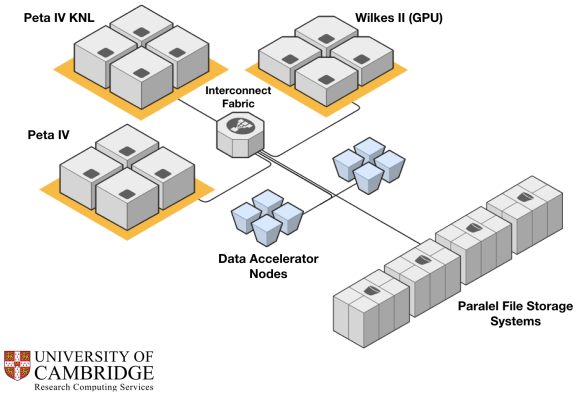

Fig. 1. Network Attached Data Accelerator

III. Architecture

Figure 1 shows a network attached architecture for the DAC. Other options include installing SSDs on IO forwarding nodes or inside the node, as an SSD or NV-DIMM. The system is built using Dell EMC R740XD. Each node has 12 NVMe's, 6 per NUMA domain dew to the number of PCI lanes available from the CPU. Each node has two fabric links to ensure the majority of the perforce is extracted over the network. Each node take a 2U space in each rack of the super computing resource of Cambridge. Each node is in 1:1 relation with its up links to the fat tree used as topology. This was dew to issues found early on with multiple Data Accelerator nodes being located on the same switch. The layout of file-system services varies between wish file-system is in used. It can be simplified to any NVMe being a member of N metadata or storage service targets per name space.

IV. Slurm Integration

The project makes use of the Cray Burst Buffer plugin open sourced as part of the Slurm workload scheduler. Our solution is designed in such a way that we can make use of it without modifying Slurm source code. Figure 2 shows the progression of a job when it requests a DAC file-system. Jobs will - if requested - stage in data and run the job when the resources has been allocated and data copied. The job then runs as normal but with the added traceable resource of an on-demand file system. When job is completed the data will then be staged back to a second file system, which can be storage location that can be mounted or accessed by a DAC node.

Figure 3 shows an example form the Slurm documentation of a users job submission script. In this example, the job is using the DAC to stage in and out data. The user specifies the location of their files. The parallel copy tool, supplied as part of the project, migrates the files from a parallel file system or home directory to the DAC.

V. Conclusion

This project provides the software platform required to manage and stage data between two tiers of storage through a workload scheduler. So far this has only been achieved in

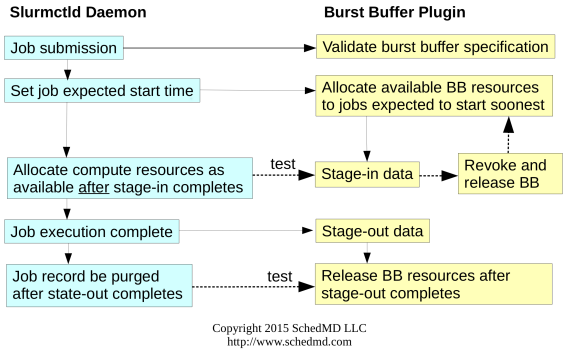

Fig. 2. Slurm Burst Buffer WorkFlow [2]

```
#!/bin/bash
#BB create_persistent name=alpha capacity=32GB access=striped type=scratch
#DW jobdw type=scratch capacity=1GB access_mode=striped
#DW stage_in  type=file source=/home/alan/data.in  destination=$DW_JOB_STRIPED/data
#DW stage_out type=file destination=/home/alan/data.out source=$DW_JOB_STRIPED/data
/home/alan/a.out
```

Fig. 3. Example Submission Script from Slurm Documentation [3]

closed environments or with limited integration to HPC job schedulers. Having this software and a validated hardware infrastructure put into use by the University of Cambridge shows the potential for these systems in research computing. Its flexibility allows the system to keep up the pace with the changing landscape of HPC computing in research and industry. The Data Accelerator will provide users with the ability to significantly improve their I/O performance. In the example of the University of Cambridge, the traditional Lustre scratch storage would be x10 the cost to achieve the same performance as the DAC. This system also allows for the preparation and experimentation required for any further procurement of SSD based storage systems.

References

[1] G. K. Lockwood, Reviewing the state of the art of burst buffers. https://glennklockwood.blogspot.com/2017/03/, March 2017.
[2] M Jette, T Wickberg, "Slurm Burst Buffer Support." https://slurm.schedmd.com/SLUG15/Burst_buffer.pdf, 2015.
[3] Slurm, "Slurm Burst Buffer Guide." https://slurm.schedmd.com/burst_buffer.html, April 2018.