Cerberus: A 3-Phase Burst Buffer Aware Batch Scheduler for HPC Systems

Jiaqi Yan and Xu Yang
Department of Computer Science
Illinois Institute of Technology
10 West 31st Street
Chicago, IL, United States
Email: {jyan31, xyang56}@hawk.iit.edu

Dong Jin and Zhiling Lan
Department of Computer Science
Illinois Institute of Technology
10 West 31st Street
Chicago, IL, United States
Email: {dong.jin, lan}@iit.edu

***Abstract*—Burst buffer drastically improves the performance of computational applications by providing high perceived IO bandwidth. However, traditional batch schedulers have barely embraced the full potential of burst buffer technology in practice. In this paper, we model the execution of scientific applications that generate tens of TB of data on a supercomputer equipped with burst buffer. We characterize the lifetime of generic applications into three phases: *stage-in*, *running*, and *stage-out*. We develop a novel burst-buffer-aware batch scheduler *Cerberus* to manage resource allocation in different phases. In both stage-in and stage-out phases, Cerberus allocates the burst buffer resources to achieve the maximum data transfer throughput between burst buffer and the external storage system. In the running phase, Cerberus maximizes the predefined objectives of job scheduling.**

1. Introduction

In the field of high performance computing (HPC), system performance is no longer throttled only by computation capability, but also by the ever-increasing I/O gap between computational resources and disk-based storage technologies. For example, on the early-state Trinity system [1], the I/O nodes between the compute nodes and the external storage can deliver transfer speed at 810 GB/s, which is still far from the target of 60 TB/s for the exascale computers [2]. The gap is critical because scientific applications typically exhibit "bursty" I/O patterns, resulting from application's defensive I/O strategy and the need for subsequent data processing [3], [4].

Extensive research have been conducted to improve I/O performance on HPC systems from both hardware and software layers. Recently, burst buffer (i.e., a solid state disk based or flash based cache tier) is proposed to absorb and spread out the "bursty" application I/O patterns [5]. Studies have demonstrated that the perceived I/O bandwidth can be significantly improved on burst-buffer-enabled systems [6]. As such, HPC users are encouraged to explicitly request burst buffer resources at job submission on burst-buffer-enabled systems.

We propose *Cerberus*[1], a novel batch scheduler for HPC systems equipped with burst buffer. A three-phase job model divides the lifetime of a user job into three phases: *stage in*, *running*, and *stage-out*. Burst buffer is used for different purposes in these phases. Unlike existing batch schedulers that make scheduling decision for each job upon its submission, Cerberus participates in all of the three phases. In each of the three phases, Cerberus adopts different optimization strategies for making scheduling decision based on job requirement at each phase. As shown later on, the preliminary simulation results demonstrate that the three-phase design of Cerberus not only boosts system responsiveness, but also improves job performance. Put together, we make three key contributions in this work:

1) We propose a three-phase job model to describe user jobs and further to facilitate job scheduling in a fine granularity.
2) We present Cerberus, a three-phase burst-buffer-aware scheduler for HPC systems equipped with burst buffer. Our design consists of several key optimization strategies for improving the performance of user jobs throughout different job phases.

2. Three-Phase Job Model

Stage-In: After submitted to the HPC system, jobs need to prefetch data, such as configuration files and input data, from PFS to the burst buffer. We define this phase as *stage-in*. The only resource needed by the job in this phase is the burst buffer. Users are encouraged to specify the amount of burst buffer required in the *stage-in* phase.

Running: The scheduler allocates the job with the required number of compute nodes and the amount of burst buffer. It then dispatches the job to the system for running. This is defined as *running* phase. During the running phase, there could be frequent data exchange between the compute nodes and the burst buffer. These interactions, like checkpointing, are mainly for achieving system fault tolerance.

1. In Greek mythology, Cerberus is a monstrous three-headed dog (three-phase scheduler), who guards the gate of the underworld to the earth (HPC system), preventing the dead (jobs) from leaving (running).

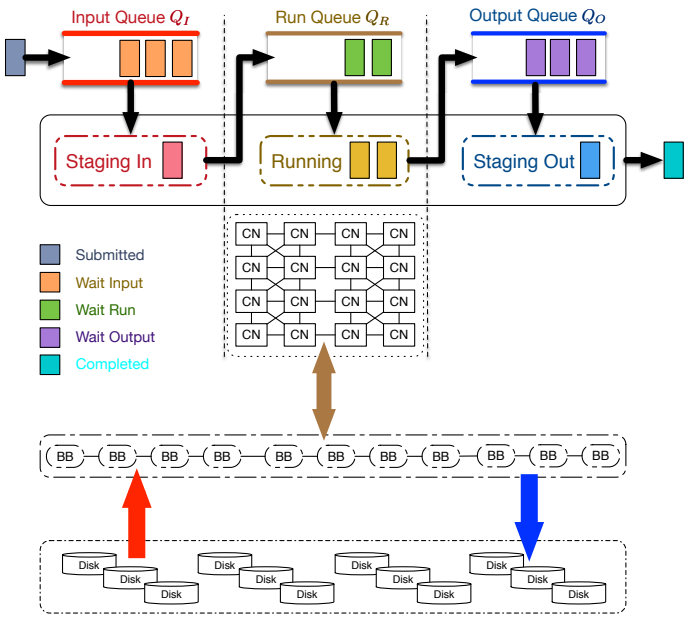

Figure 1. The design of Three-Phase burst buffer aware scheduling. The black arrows indicate the jobs changing of status; the red, brown and blue arrows in the bottom indicate the data flow in different phases.

Stage-out: When the job finishes computation and exits the *running* phase, its output data need to be drained out to the external storage from the on-node memory. We define this as *stage-out* phase. The job can release its compute nodes when the output data are staged out into the burst buffer at high speed.

3. Design of Cerberus

3.1. Three-Phase Job Scheduling

Cerberus makes scheduling decisions based on the most common use cases of burst buffer: data stage-in, checkpointing, and data stage-out. It allocates burst buffer to each job by maintaining three distinct queues, as shown in Figure 1. The input queue Q_I contains all the jobs that need to prefetch data from the external storage to the burst buffer. The running queue Q_R holds the jobs that are waiting for the compute nodes and the burst buffer used for checkpointing. Jobs waiting to be drained out to the external permanent storage are in the output queue Q_O. The layered hierarchy in Figure 1 also indicates that burst buffer can fill in the memory gap between the memory on compute nodes and the hard disk storage.

3.2. Scheduling in the Stage-in/out Phase

In the stage-in phase, jobs require only the burst buffer. Cerberus schedules the jobs in Q_I based on their burst buffer demands bb_in. To maximize burst buffer's utilization, Cerberus schedules the jobs according to the following optimization:

Problem 1

$$\max \sum_{i \in J_{Q_I}} bb\_in_i \cdot x_i$$

$$s.t. \begin{cases} x_i \in \{0,1\}, \forall i \in J_{Q_I} \\ \sum_{i \in J_{Q_I}} bb\_in_i \cdot x_i \leq BB_{available} \end{cases} \quad (1)$$

The stage-out phase scheduling is designed based on the value of bb_out of the jobs in Q_O. The optimization formula for the stage-out phase is the same as Problem 1.

3.3. Scheduling in the Running Phase

Running jobs not only require the compute nodes for computation, but also utilize the burst buffer to enhance the I/O performance. Here we take a general strategy by defining the **profit** of job_i in Q_R as it uses c_i compute nodes and bb_run_i amount of the burst buffer

$$p_i = f(c_i, bb\_run_i, ert_i) \quad (2)$$

where ert_i is the expected running time of job_i.

Cerberus's objective is to maximize the total profit of jobs in the running queue, formularized as

Problem 2

$$\max \sum_{i \in J_{Q_R}} p_i \cdot x_i$$

$$s.t. \begin{cases} x_i \in \{0,1\}, \forall i \in J_{Q_R} \\ \sum_{i \in J_{Q_R}} c_i \cdot x_i \leq CN_{available} \\ \sum_{i \in J_{Q_R}} bb\_run_i \cdot x_i \leq BB_{available} \end{cases} \quad (3)$$

References

[1] ACES Team, "Trinity Platform Introduction and Usage Model," http://www.lanl.gov/projects/trinity/_assets/docs/trinity-usage-model-presentation.pdf, Tech. Rep., August 2015.

[2] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale Computing Technology Challenges," in *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, ser. VECPAR'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–25.

[3] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, "Understanding and improving computational science storage access through continuous characterization," in *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on*, May 2011, pp. 1–14.

[4] Y. Kim, R. Gunasekaran, G. M. Shipman, D. A. Dillow, Z. Zhang, and B. W. Settlemyer, "Workload characterization of a leadership class storage cluster," in *Petascale Data Storage Workshop (PDSW), 2010 5th*, Nov 2010, pp. 1–5.

[5] J. Bent and G. Grider, "Usability at los alamos national lab," in *The 5th DOE Workshop on HPC Best Practices: File Systems and Archives*, Sept 2011.

[6] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *2012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, April 2012, pp. 1–11.