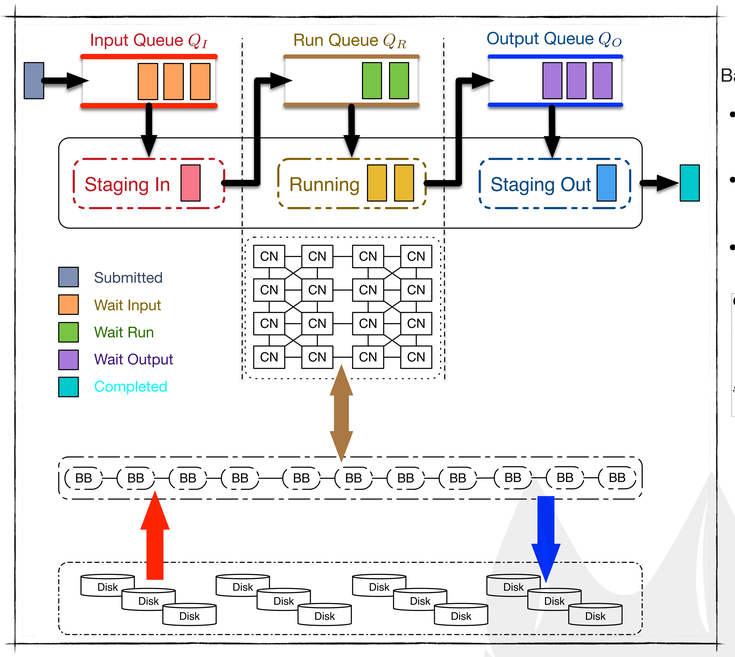
Cerberus: A Three-Phase Burst-Buffer-Aware Batch Scheduler for High Performance Computing

ILLINOIS INSTITUTE OF TECHNOLOGY

Jiaqi Yan, Xu Yang, Dong (Kevin) Jin, Zhiling Lan



Three Phase Job Model

Based on burst buffer's three major usage cases

- Jobs are divided into three phases: Stage-in phase, Running phases and Stage-out phase
- Cerberus manages jobs in three different phases with three queues: Q_I, Q_R, Q_O
- For each queue, Cerberus adopts different optimization-based scheduling algorithm

Optimize Input/Output Phase

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

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

Optimize Running Phase

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

$$s.t. \begin{cases} p_i = f(c_i, bb\_run_i, ert_i) \\ x_i \in \{0,1\}, \forall i \in J_{Q_R} \\ \sum_{i \in J_{Q_R}} c_i \cdot x_i \le CN_{available} \\ \sum_{i \in J_{Q_R}} bb\_run_i \cdot x_i \le BB_{available} \end{cases}$$

Table 1: Summary of Symbols

CN	number of total compute nodes
BB	amount of total burst buffer nodes
$CN_{available}$	number of available compute nodes
$BB_{available}$	amount of available burst buffer nodes
c_i	compute node demand of job_i
bb_in_i	burst buffer demand of job_i at *stage-in* phase
bb_run_i	burst buffer demand of job_i at *running* phase
bb_out_i	burst buffer demand of job_i at *stage-out* phase
rt_i	running time of job_i recorded in the log trace
ert_i	estimated or expected running time of job_i
Q_I, Q_R, Q_O	queues for *stage-in, running, stage-out* phases
$J_{Q_I}, J_{Q_R}, J_{Q_O}$	set of jobs in corresponding queues
$\mathcal{J}_{Q_I}, \mathcal{J}_{Q_R}, \mathcal{J}_{Q_O}$	selected jobs from the corresponding queues
p_i	profit of $job_i \in J_{Q_R}$
$dp(\cdot)$	memoization used in dynamic programming

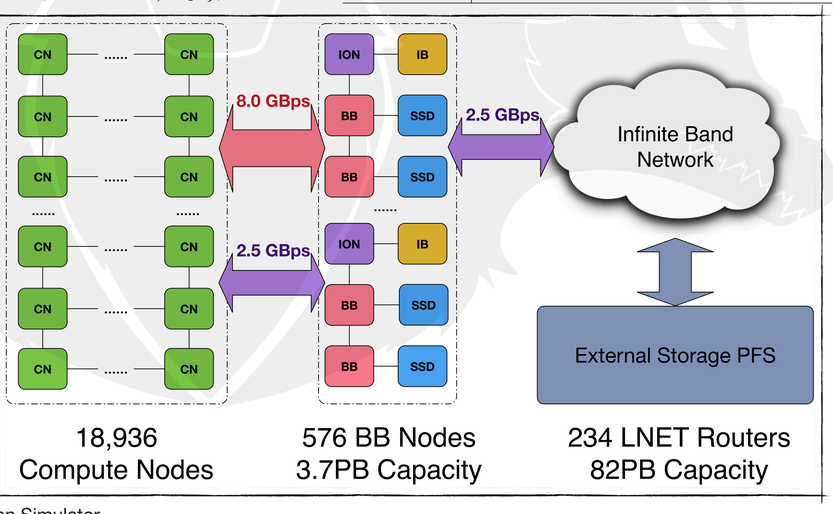
Design of Three-Phase Burst Buffer Aware Scheduler

$$dp(i,w) = \begin{cases} 0, \text{ if } i = 0 \\ dp(i-1,w), \text{ if } bb\_in_i > w \\ \max\{dp(i-1,w), dp(i-1,w-bb\_in_i) + bb\_in_i\}, \text{ if } bb\_in_i \le w \end{cases}$$

$$dp(i,c,w) = \begin{cases} 0, \text{ if } i = 0 \\ dp(i-1,c,w), \text{ if } c_i > c \text{ or } bb\_run_i > w \\ \max\{dp(i-1,c,w), dp(i-1,c-c_i,w-bb\_run_i) + p_i\}, \text{ otherwise} \end{cases}$$

Both two formularized problem are equivalent to the NP-complete 0-1 knapsack problem. We solve them in pseudo-polynomial time through dynamic programming, shown above, and backtracking.

Another thing to point out is that Cerberus is **NOT** unfair to non-I/O jobs. As for stage-in phase, a non-I/O job will always be selected by Cerberus. Since non-I/O jobs don't need burst buffer at this phase, scheduling it to will affect neither of the constraints. To avoid starving certain jobs demanding extremely large number of burst buffers and/or cores, Cerberus keeps tracking of their holding time. When the waiting time exceed certain threshold, Cerberus will stop the normal scheduling process to dispatch the large jobs.

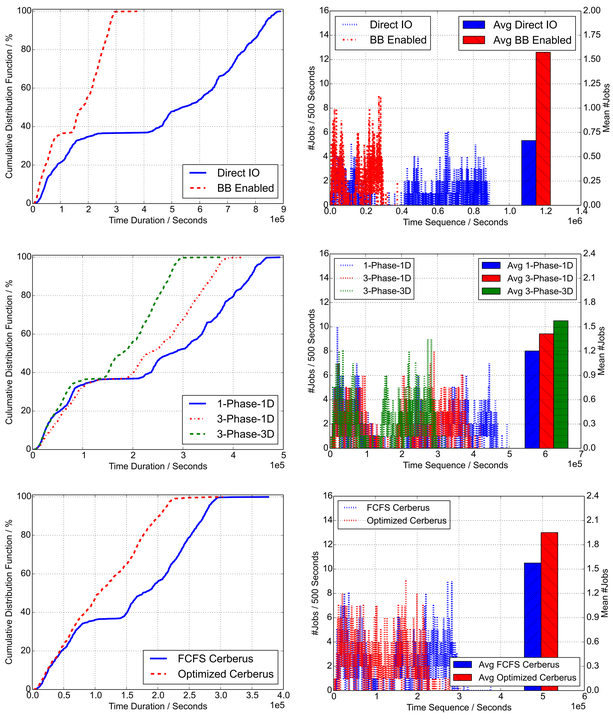


Evaluation with BBSim[2]

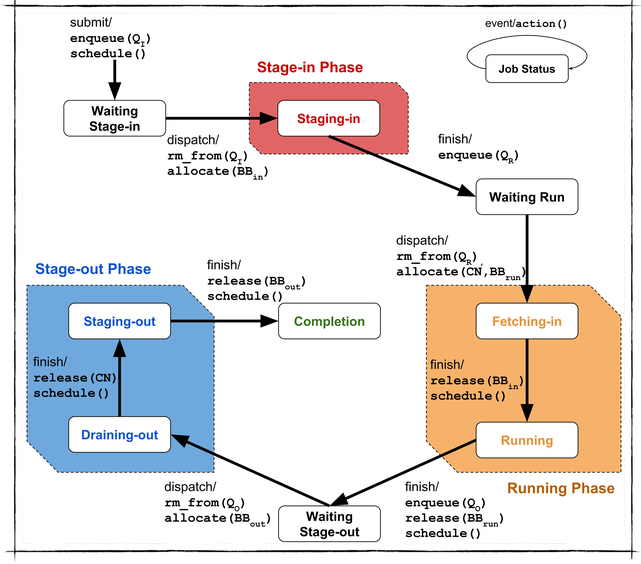
We evaluate Cerberus with **BBSim**, an event-driven Simulator

Simulating a Trinity Inspired HPC System[1]

- >80% jobs have less waiting time if they utilize burst buffer
- >80% of the three-phase-modeled jobs finish earlier than 1-phase-modeled jobs
- >60% of the jobs take less time if user specifies data usage demand at each phase
- Compared with the FCFS, Cerberus achieves about 23.87% higher average throughput (1.951 jobs / 500 seconds).



Job State Transition in Cerberus



[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. Yan, "BBsim: a batch scheduler simulator for HPC systems equipped with burst buffer," https://github.com/littlepretty/BBSimulator, 2016.