Scheduling Beyond CPUs for HPC

Fan Y, Lan Z, Rich P, et al.
HPDC 19

Zhouxin Xue, Xuanhang Diao

School of Biomedical Engineering,
ShanghaiTech University

14, April

Pareto set

A **Pareto set** is a set of optimal solutions, where no objective can be improved without worsening another objective.

Burst Buffer

A **Burst Buffer** is an intermediate storage layer positioned between compute nodes and parallel file systems (PFS) in high-performance computing (HPC) systems.

- Absorb the bursty I/O data generated by data-intensive applications.
- Built from solid-state drives (SSDs).
- Can be either attached to compute nodes as local resources or configured as global resources shared by compute nodes.

Multi-Resource Scheduling

- HPC systems are equipped with diverse global and local resources.
- HPC job scheduler plays a crucial role in efficient use of resources.

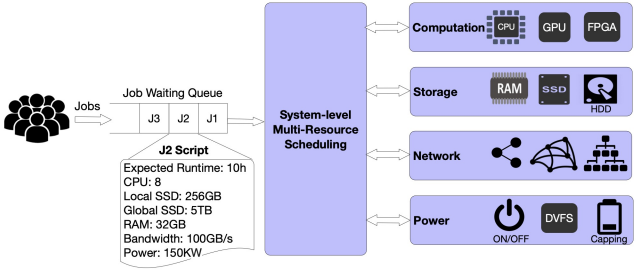

Figure 1: HPC job scheduling problem involves in multiple resources.

Limitations of Existing Scheduling Methods

Existing methods often overlook alternative solutions or optimal resource combinations, leading to under-utilization of resources or poor application performance.

- Naive method
- Constrained method
- Weighted method
- Bin packing method

Goal

The Motivation of the this paper is to improve overall resource utilization and reduce job wait time in HPC systems.

- providing rapid scheduling decisions
- minimizing the impact on site policies
- ensuring extensibility to accommodate emerging resources

Window-based Scheduling

The idea of window-based scheduling as shown in Fig 2 is that the first w jobs in the job waiting queue get copied into the window.
This allows our BBsched to consider the site policies and keep the order of the base scheduler as similar as possible.

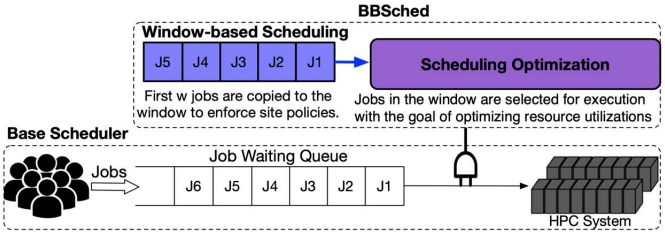

Figure 2: The overview of BBsched.

MOO Solver

Suppose a system has N nodes and burst buffers of total B GB. The amounts of nodes and burst buffers being used are N_{used} and B_{used} respectively. Suppose $J = \{J_1, ..., J_w\}$is a set of w jobs in the scheduling window. Job J_i requiring n_i nodes and b_iGB of burst buffers.

The scheduling problem can be transformed into the following MOO: to determine a finite set of Pareto solutions X; each Pareto solution $x \in X$ is represented by a binary vector $x = [x_1, ..., x_w]$, such that $x_i = 1$ if J_i is selected to execute and $x_i = 0$ otherwise.

Pareto solution optimizes the following two objectives:
(1) maximize node utilization: $f_1(\boldsymbol{x}) = \sum_{i=1}^{w} n_i \times x_w$
(2) maximize burst buffer utilization: $f_2(\boldsymbol{x}) = \sum_{i=1}^{w} b_i \times x_i$
Formally, the problem can be formulated as:

$$\begin{array}{ll} \max & (f_1(\boldsymbol{x}), f_2(\boldsymbol{x})) \\ \text{s.t.} & \sum_{i=1}^{w} n_i \times x_i \leq N - N_{\text{used}}\ , \quad x_i \in \{0,1\} \\ & \sum_{i=1}^{w} b_i \times x_i \leq B - B_{\text{used}}\ , \quad x_i \in \{0,1\} \end{array}$$

MOO Solver

Because this MOO problem is NP-hard, this paper uses a genetic algorithm to approximate the Pareto set (optimal solutions).

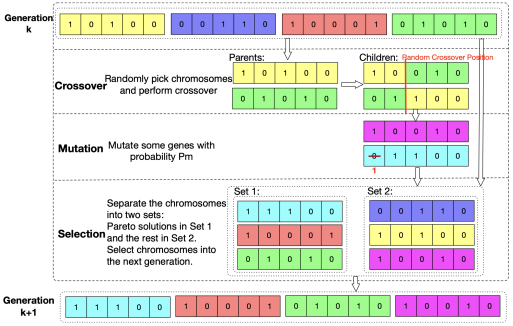

Figure 3: MOO solver maintains a population of candidate solutions (4 chromosomes). A chromosome consists of 5 genes, where each gene represents the selection of the job at a specific location in the window and encodes as a binary number: 1 (selected) or 0 (not selected).

Decision making

The output of the solver is a Pareto set, and a decision maker needs to select one preferred solution.

Different HPC facilities may have different site policies and scheduling priorities.

System managers may use a site-specific metric for selecting a preferred solution out of the Pareto set.