BoPF: Mitigating the Burstiness-Fairness Tradeoff in Multi-Resource Clusters

Tan N. Le[1,2], Xiao Sun[1], Mosharaf Chowdhury[3], and Zhenhua Liu[1]
[1]Stony Brook University, [2]SUNY Korea, [3] University of Michigan

1 MOTIVATION

Even though batch, interactive, and streaming applications all care about performance, their notions of performance are different. For instance, while the average completion time can sufficiently capture the performance of a throughout-sensitive batch-job queue (TQ) [5], interactive sessions and streaming applications form latency-sensitive queues (LQ): each LQ is a sequence of small jobs following an ON-OFF pattern. For these jobs [7], individual completion times or latencies are far more important than the average completion time or the throughput of the LQ.

Indeed, existing "fair" schedulers are inherently unfair to LQ jobs: when LQ jobs are present (ON state), they must share the resources equally with TQ jobs, but when they are absent (OFF state), batch jobs get all the resources. In the long run, TQs receive more resources than their fair shares because today's schedulers such as Dominant Resource Fairness [4] make *instantaneous* decisions.

Clearly, it is impossible to achieve the best response time for LQ jobs under *instantaneous* fairness. In other words, there is a hard tradeoff between providing instantaneous fairness for TQs and minimizing the response time of LQs. However, instantaneous fairness is not necessary for TQs because average-completion time over a relatively long time horizon is their most important metric. This sheds light on the following question: *how well can we simultaneously accommodate multiple classes of workloads with performance guarantees, in particular, isolation protection for TQs in terms of long-term fairness and low response times for LQs*?

This work serves as our first step in answering the question by designing BoPF: the first multi-resource scheduler that achieves both isolation protection for TQs and response time guarantees for LQs in a strategyproof way. The key idea is "bounded" priority for LQs: as long as the burst is not too large to hurt the long-term fair share of TQs and other LQs, they are given higher priority so jobs can be completed as quickly as possible.

2 BOPF: BOUNDED PRIORITY FAIRNESS

2.1 Modeling the Problem

We consider a system with K types of resources. The capacity of resource k is denoted by C^k. The system resource capacity is therefore a vector $\overrightarrow{C} = \langle C^1, C^2, ..., C^K \rangle$.

LQ-i's demand comes from a series N_i of bursts, each consisted of a number of jobs. We denote by $T_i(n)$ the arrival time of the n-th burst, which must be finished within $t_i(n)$. Therefore, its n-th burst needs to be completed by $T_i(n) + t_i(n)$ (i.e., deadline). Denote the demand of its n-th arrival by a vector $\overrightarrow{d_i}(n) = \langle d_i^1(n), d_i^2(n), ..., d_i^K(n) \rangle$, where $d_i^k(n)$ is the demand on resource-k. We assume users report their estimated demand for presentation simplicity, but uncertainty handling and other details can be found in our technical report [1].

Completion time: Let us denote by $R_i(n)$ the (last) completion time of jobs during LQ-i's n-th arrival. If LQ-i is admitted with *hard* guarantee, we ensure that a large fraction (α_i, e.g., 95% or 99% depending on the SLAs) of arrivals are completed before deadlines, i.e., $\sum_{n \in N_i} \mathbf{1}_{\{R_i(n) \le T_i(n) + t_i(n)\}} \ge \alpha_i |N_i|$, where $\mathbf{1}_{\{\cdot\}}$ is the indicator function.If LQ-i is admitted with only *soft/best-effort* guarantee, we want to maximize the fraction of arrivals completed on time.

Long-term fairness: Denote by $\overrightarrow{a_i}(t)$ and $\overrightarrow{e_j}(t)$ the resources allocated for LQ-i and TQ-j at time t, respectively. For a possibly long evaluation interval $[t, t+T]$ during which there is no new admission or exit, the average resource guarantees received are calculated as $\frac{1}{T}\int_t^{t+T} \overrightarrow{a_i}(\tau) \mathrm{d}\tau$ and $\frac{1}{T}\int_t^{t+T} \overrightarrow{e_j}(\tau) \mathrm{d}\tau$. We require the allocated dominant resource, i.e., the largest amount of resource allocated across all resource types, received by any TQ queue is no smaller than that received by an LQ for isolation protection.

2.2 Solution Approach

BoPF consists of the following three major components:

Admission control procedure: BoPF classifies admitted LQs and TQs into three classes: LQs admitted with hard resource guarantee ($\mathbb{H}$), LQs admitted with soft resource guarantee ($\mathbb{S}$), and elastic queues that can be either LQs or TQs ($\mathbb{E}$).

Before admitting LQ-i, BoPF checks if admitting it invalidates any resource guarantees committed for LQs in $\mathbb{H} \cup \mathbb{S}$, i.e., the *safety condition* $\overrightarrow{d_j}(n) \le \frac{\overrightarrow{C}(T_j(n+1) - T_j(n))}{|\mathbb{H}| + |\mathbb{S}| + |\mathbb{E}| + 1}, \forall n, \forall j \in \mathbb{H} \cup \mathbb{S}$. If it is not satisfied, LQ-i is rejected. Otherwise, it is safe to admit LQ-i. If its own total demand exceeds its long-term fair share (*fairness condition*), LQ-i is added to $\mathbb{E}$ for best-effort services. Otherwise if there are enough uncommitted resources, LQ-i is added to $\mathbb{H}$ with hard guarantee. If there are not enough resources left, it is added to $\mathbb{S}$. For TQ-j, BoPF simply checks the safety condition. If it is satisfied, TQ-j is added to $\mathbb{E}$. Otherwise TQ-j is rejected.

Guaranteed resource provisioning procedure: For each LQ-i in $\mathbb{H}$, during $[T_i(n), T_i(n) + t_i(n)]$, BoPF allocates constant resources to fulfill its demand $\overrightarrow{a_i}(t) = \frac{\overrightarrow{d_i}(n)}{t_i(n)}$. LQs in $\mathbb{S}$ shares the uncommitted resource $\overrightarrow{C} - \sum_{j \in \mathbb{H}} \overrightarrow{a_j}(t)$ based on SRPT [3] until each LQ-i's consumption reaches $\overrightarrow{d_i}(n)$ or the deadline arrives. Remaining resources are allocated to queues in $\mathbb{E}$ using DRF [4].

Spare resource allocation procedure: If some allocated resources are not used, they are further shared by TQs and LQs with unsatisfied demand. This maximizes system utilization.

2.3 Properties of BoPF

First, we briefly discuss how *BoPF ensures long-term fairness, burst guarantee, and Pareto efficiency.* The safety condition and fairness condition ensure the long-term fairness for all TQs. For LQs in $\mathbb{H}$,

SIGMETRICS'18 WIP, Irvine, California, USA
2018.

they have hard resource guarantee and therefore can meet their SLA. For LQs in $\mathbb{S}$, they have resource guarantee whenever possible, and only need to wait after LQs in $\mathbb{H}$ when there is a conflict. Therefore, their performance is much better than if they were under fair allocation policies. The addition of $\mathbb{S}$ allows more LQs to be admitted with resource guarantee, and therefore increases the system resources utilized by LQs. Finally, we fulfill spare resources with TQs, so system utilization is maximized, reaching Pareto efficiency.

In addition, we prove that *BoPF is strategyproof* in [1].

2.4 Enabling BoPF in Cluster Managers

We now describe how we have implemented BoPF on Apache YARN. We use standard techniques for demand estimation. A key benefit of BoPF is its simplicity of implementation: we have implemented it in YARN using only 600 lines of code. We made three main changes for user input, admission control, and resource scheduler – all in the *resource manager* (RM). We do not modify *node manager* (NM) or application master (AM). Figure 1 depicts our design. More detailed implementation and operational issues can be found in our full technical report [1].

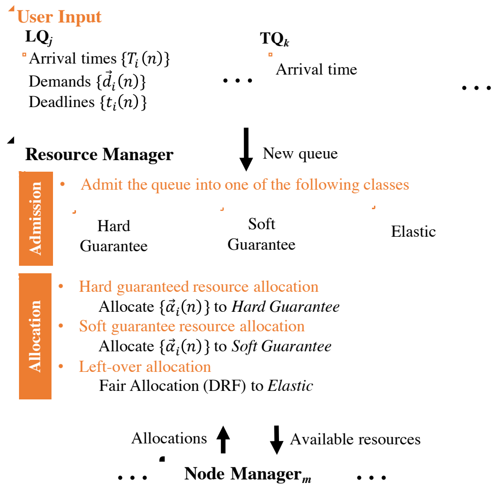
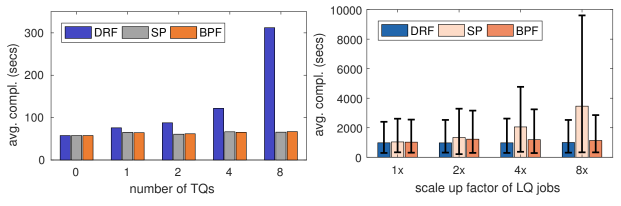

User Input
LQ$_j$
Arrival times $\{T_i(n)\}$
Demands $\{\vec{d}_i(n)\}$
Deadlines $\{t_i(n)\}$
...
TQ$_k$
Arrival time
...
Resource Manager
New queue
Admission
• Admit the queue into one of the following classes
Hard Guarantee
Soft Guarantee
Elastic
Allocation
• Hard guaranteed resource allocation
Allocate $\{\vec{a}_i(n)\}$ to *Hard Guarantee*
• Soft guarantee resource allocation
Allocate $\{\vec{a}_i(n)\}$ to *Soft Guarantee*
• Left-over allocation
Fair Allocation (DRF) to *Elastic*
Allocations
Available resources
...
Node Manager$_m$
...

Figure 1: Enabling bounded prioritization with long-term fairness in a multi-resource cluster.

3 EVALUATION

We evaluated BoPF using the big data benchmark – BigBench (BB) [2]. We ran experiments on a 40-server CloudLab cluster. We setup Tez atop YARN for the experiment.



(a) Avg. compl. of LQ jobs **(b) Avg. compl. of TQ jobs**

Figure 2: BoPF can closely approximate the LQ performance of Strict Priority and the long-term fairness for TQs of DRF.

We compare BoPF to two baseline policies: Strict Priority (SP) [6] and DRF [4]. While SP prioritizes LQ whenever possible, DRF ensures instantaneous fairness.

In Figure (2a), there are a single LQ and multiple TQs. When there are no TQs, the average completion times of LQ jobs across three schedulers are the same (57 seconds). As the number of TQs increases, the performance of DRF significantly degrades because DRF tends to allocate less resource to LQ jobs. In contrast, BoPF and SP give the highest priority to LQs and the average completion times of LQ jobs are almost unchanged (65 seconds).

When we have too many LQ jobs, Figure (2b) highlights that SP allocates too much resource to LQ jobs that significantly hurts TQ jobs, while DRF maintains similar average completion times of TQ jobs. BoPF performs closely to DRF.

In summary, BoPF speeds up latency-sensitive jobs by 4.66× compared to DRF, while still maintaining long-term fairness. In the meantime, BoPF improves the average completion times of throughput-sensitive jobs by up to 3.05× compared to Strict Priority.

Figure 3 illustrates the admission in the case of multiple LQs. LQ-0 arrives and receives the performance guarantee. LQ-1 conflicts with LQ-0 and is put into soft-guarantee. LQ-2 is treated like TQ-0 because it requires too much resource.

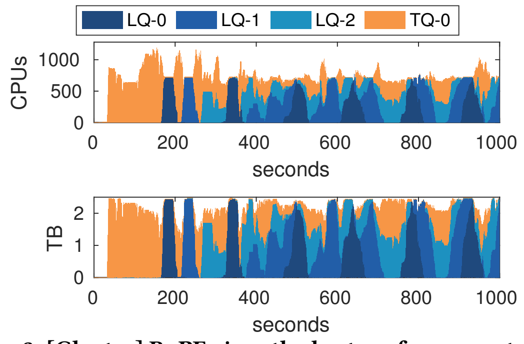


Figure 3: [Cluster] BoPF gives the best performance to LQ-0, near optimal performance for LQ-1, and maintains fairness among 4 queues.

4 FUTURE DIRECTIONS

For those cases where inter-arrival time is hard to predict, service curve can be leveraged to provide performance guarantee. However, admission control based on service curves can be conservative, resulting in few queues admitted. We are currently working on probabilistic service curve based allocation algorithms.

5 ACKNOWLEDGMENTS

This research is supported by NSF grants 1617698, 1717588, 1730128, 1563095, 1617773, 1629397, and was partially funded by MSIP, Korea, IITP-2017-R0346-16-1007.

REFERENCES

[1] BoPF: Mitigating the Burstiness-Fairness Tradeoff in Multi-Resource Clusters – Technical Report. https://bit.ly/2rBZDjc.
[2] Big-Data-Benchmark-for-Big-Bench. https://github.com/intel-hadoop/Big-Data-Benchmark-for-Big-Bench, 2016.
[3] N. Bansal and M. Harchol-Balter. *Analysis of SRPT scheduling: Investigating unfairness*, volume 29. ACM, 2001.
[4] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *NSDI*, 2011.
[5] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella. Multi-resource packing for cluster schedulers. In *SIGCOMM*, 2014.
[6] L. Kleinrock and R. Gail. *Queueing systems: Problems and Solutions.* Wiley, 1996.
[7] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica. Discretized streams: Fault-tolerant stream computation at scale. In *SOSP*, 2013.