ROSE: Cluster Scheduling through Efficient Resource Overselling

Xiaoyang Sun[1*], Chunming Hu[1], Renyu Yang[1*], Peter Garraghan[2], Chao Li[3]

Beihang University[1] Lancaster University[2] Alibaba Cloud Inc[3]

Motivations: A long-standing issue in cluster scheduling is the ability to effectively improve the resource utilization of heterogeneous resources. Specifically, there exists a substantial disparity between perceived and actual resource utilization. The scheduler considers a cluster fully utilized if a large resource request queue is present, even when in actuality the average actual resource utilization of cluster nodes is low. This disparity results in the formation of idle resources producing inefficient cluster resource usage, reduced system availability and incurring greater operational costs and an inability to provision service. Through profiling a production cluster, we have identified that the main causes for such idle resources are resultant of message feedback delay, resource over-estimation and fragmentation. An intuitive solution to this problem is to leverage the overselling technique or opportunistic execution. It is highly desirable to exploit idle resources to minimize the waiting time of submitted jobs, thereby shortening the end-to-end job timespan and increasing system resource efficiency.

Apollo[2] introduces *opportunistic scheduling* to take advantage of idle resources. However, randomly chosen tasks can only fill the spare space of compute slots and may lead to blind task dispatching. Mercury[3] adopts *hybrid scheduling* to enhance cluster throughput and reduce feedback delays. The method heavily relies on precise queue delay estimation, and thus is not widely applicable to systems where the execution time is challenging to estimate due to volatile workload manifestation. Sparrow[4] performs random-based probing to assign tasks. However, due to limited visibility of the entire cluster resources, it sacrifices scheduling quality for low-latency, and thus is only designed for short interactive tasks. It is unlikely to ascertain an appropriate destination server under high machine load. None of these approaches are able to effectively reuse idle resources, particularly during frequent resource requests.

Design: We design a cluster scheduling system ROSE that enhances the two-layer scheduler architecture[1] [5] to oversell idle resources to guarantee sufficient job execution whilst maintaining inter-job fairness and cluster resource efficiency. Specifically, when resource requests cannot be fully satisfied, the ROSE job attempts to select idle resources from compute nodes in a speculative manner instead of waiting for the official resource allocation provided by the centralized scheduler. The job intelligently requests to launch tasks within ranked machines that are most suited to oversell resources. To avoid inter-task performance interference, these oversold tasks run at lower priorities and are preemptable compared to currently executing tasks within the machine. This approach is complementary and compatible to existing protocols between the Application Master (AM) and Resource Manager (RM) within YARN[1].

Implementation: ROSE leverages a *multi-phase machine filtering mechanism* to select and rank candidates prior to overselling resources. The procedure considers machine load, correlative workload performance, and queue states into account for decision-making. To this end, a *monitor* is introduced into the Node Manager(NM)[1] and a *scorer* into the AM. The monitor component is responsible for collecting runtime local information (`CPU`, `memory`, `disk IO`, `network`, `queue length`, `running task number`, etc.) The scorer component is responsible for rating the penalty level of each machine by synthesizing the eviction, failure, straggler occurrences at task-level. We use CA (*Cluster Aggregator*) to converge the monitored statistic data and machine scores. Machines that are timing-out, overloaded or blacklisted will be eliminated. In the next phase, an election is conducted by measuring scores, the weighted loads of multi-dimension resources and queue states. Additionally, we leverage timestamp ordering and bit compression to incrementally maintain the consistency of candidate information whilst optimizing the transmission efficiency.

We extend NM by introducing a *threshold controller* and *queue management*. The controller manages the whole life-cycle of oversold tasks based on runtime system information and quota control: task enqueue permission, execution start time, resource allocation, preemption, and the suitable time for task placement. Multi-resource restrictions are imposed during the execution of oversold tasks. Herein, a quota is used to regulate the degree of resource overselling in the cluster to avoid using excessive resources. In this manner, the over-estimation and fragmented resources can be aggregated and fully reused. Furthermore, due to the significant variation of cluster states, the original decisions for task placement can become sub-optimal. In our design, once the job determines that oversold tasks are delayed within the NM's queue via *time-out detection*, it will autonomously adjust the task distribution to improve task placement and reduce straggler occurrence.

Evaluation: We extended Fuxi[5] to implement ROSE and evaluated its ability to improve cluster resource utilization and job makespan on a 210-machine cluster and submitted a synthetic workload with 60 IO-intensive jobs to emulate production jobs. Compared with the non-overselling approach, ROSE almost doubles the CPU utilization (from 25.63% to 52.9%) on average. Additionally, our system outperforms random-based, system-load-based and queue-length-based methods with at most 30.11% makespan reduction and 18.23% disk utilization improvement.

References.

[1] https://hadoop.apache.org/docs/r2.7.4/hadoop-yarn/

[2] Boutin E, et al. Apollo: scalable and coordinated scheduling for cloud-scale computing. USENIX OSDI 2014.

[3] Karanasos K,et al. Mercury: hybrid centralized and distributed scheduling in large shared clusters. USENIX ATC, 2015.

[4] Ousterhout K, et al. Sparrow: distributed, low latency scheduling. ACM SOSP 2013.

[5] Zhang Z, et al. Fuxi: a Fault-Tolerant Resource Management and Job Scheduling System at Internet Scale. VLDB 2014

** The student Xiaoyang Sun will present the poster. Dr. Renyu Yang is the corresponding author (renyu.yang@buaa.edu.cn).*