Analysis of Co-locating Batch and Latency-critical Workloads in Single Cluster

CS 744 Spring 24 Project Proposal

Mondo Jiang, Zihao Zhu

Introduction

In this project, we plan to analyze different schedulers' behavior when scheduling low priority workloads (e.g. machine learning training jobs, daily conclution data analysis jobs, etc.) and latency-critic workloads (e.g. web servers, machine learning inference and real-time interactive data analysis jobs) together in single cluster.

In previous works, most schedulers only schedule single type of workloads: In Mesos, each workload has its own scheduler [2], and INFaaS [3] only considers machine learning inference workloads, etc. But we might achieve better resource utilization by having a centralized scheduler to manage various types of workloads. This is especially true in the case of webservices, because they always have idle resources during the day, as we can see in Figure 1.

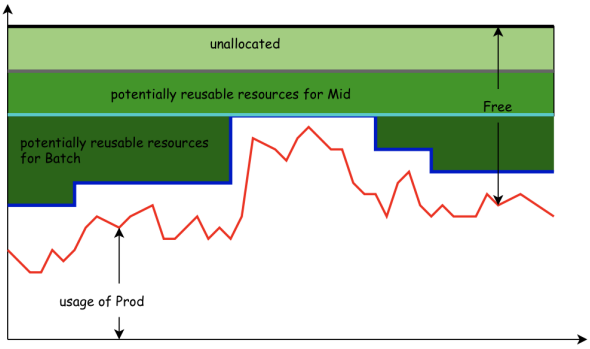

Figure 1: A Typical Day of a Web Service [4]

By co-locating batch jobs and real-time data analysis, we may further increase data locality among different jobs which manipute the same partition of data, and reduce the cost of data transfer. This is especially useful in cloud data center senerio, because it can significantly reduce the overall cost for running a data center.

We plan to use Koordinator [4] as our scheduler, and compare its performance with default K8s scheduler and other schedulers like Kube-Batch [**?**]. We will also analyze the tide effect of different jobs (GPU and other resources), and how to give wiggle room for online jobs. Specifically, we will consider following use cases:

1. co-locate webservices and batch jobs together in single cluster.

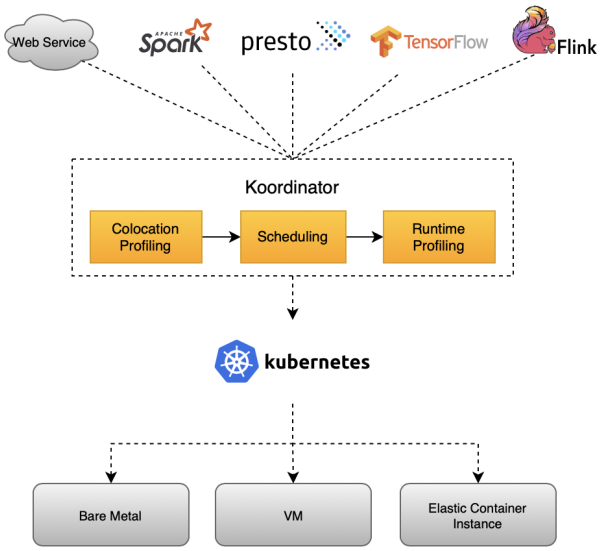

Figure 2: Mixing Workloads in Single Cluster [4]

2. running response time sensitive online ml models (gpt, search by image, receipt analysis) with batch processing and webservices.

3. running real-time analytics on the same data that is used for batch processing to analyze locality traits.

Methodology

1. Build a method to simulate different types of workloads in K8s (Web Services and its tide effect, Machine Learning Training and Inference Workloads, Hadoop Data Analysis Batch Jobs, Real-time Data Analysis).

2. Compare between differently jobs performance in K8s using current schedulers, form the metrics (e.g. SLO for ML Inference and web server requests, total execution time for training and batch jobs, overall resource usage, etc.) and a baseline.

3. Analysis more aspects of this topis, such as

   (a) Decisions made by current schedulers and their effects on cluster performance.

   (b) The tide effect of different jobs (GPU and other resources), how can we archive a overall better resource usage while not breaking SLO requirements.

   (c) How to give wiggle room for online jobs to handle fluctuation loads.

4. If time permits, we will also try to implement a new scheduler based on Kubernetes's default scheduler, and compare its performance with other schedulers.

Timeline

Related work

Kube-Scheduler default scheduler for Kubernetes, supports scheduling based on resource requirements, Pod and Job correspond to different types of workloads.

Koordinator QoS-based scheduling for efficient orchestration of microservices, AI, and big data workloads on Kubernetes[4]

Kube-Batch a batch scheduler for Kubernetes, providing mechanisms for applications which would like to run batch jobs leveraging Kubernetes.

Bytedance Co-locate Practice with Intel Performance Evaluation and Optimization for Workload Colocation[1]

References

[1] I. ByteDance. Bytedance: Performance evaluation and optimization for workload colocation. `https://www.intel.com.br/content/dam/www/public/us/en/documents/white-papers/Intel_ByteDance_WhitePaper.pdf`.

[2] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica. Mesos: A platform for {Fine-Grained} resource sharing in the data center. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

[3] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 397–411, 2021.

[4] K. Team. Koordinator, qos-based scheduling for efficient orchestration of microservices, ai, and big data workloads on kubernetes. `https://koordinator.sh`.