Batch Job Run Time Prediction for Auto-Scaling in the Cloud

Minghua Ma [1] **Christopher Zheng** [2] **Junjie Chen** [3]
Yilin Li [1] **Dan Pei** [1]
[1]Tsinghua University, [2]McGill University, [3]Tianjin University

Abstract

Batch jobs are increasingly prevalent at cloud because of a growing demand of various industries. To swiftly manage batch jobs with limited resources in the cloud, we design Batman, a framework that learns from a baseline run time of batch jobs, makes predictions of online batch job run time, and auto-scales cloud resources according to online predictions. With Batman, batch jobs can be accelerated within the run time baseline.

Introduction

Keeping pace with the rapid development of cloud computing, a growing need of running batch jobs at cloud for many industries emerges. Batch jobs, such as big data analytics, usually run for a long time and are executed recurrently (*e.g.*, for nearly every day) (Chase et al. 2019). To improve resource utilization, it is a common strategy to deploy batch jobs and online services in a co-location mode. As an inevitable consequence, the lengths of run time of batch jobs executed on different days vary due to the jobs' complex environment. Operators always have an expectation of run time of batch jobs because the run time is critical to business operations. If the run time exceeds operators' expectation, they need to manually scale the current system up with more resources (CPU cores and/or memory) and resume the remaining part of batch jobs. This described procedure, however, is both labor-intensive and error-prone. Therefore, automatically managing batch jobs considering their run time is an important and meaningful task.

Previous work develops and focuses on a run time deadline driven strategy to schedule batch jobs (Ferguson et al. 2012), according to which operators need to manually specify a run time deadline and subsequently allocate the computing resources. Nevertheless, it is unfeasible to configure a fixed deadline for every batch job considering the overwhelmingly large number of batch jobs and their distinct workloads which unavoidably influence their run time.

By reviewing the current practice of a top-tier banking IT system in China, which houses thousands of batch jobs in its private cloud, We observe that run time of batch jobs does follow certain patterns, such as the periodicity and trend. Operators do not set specific deadlines for these jobs but they do care how long these jobs normally take to finish, which can be called run time baseline. Therefore, to be fully automatic, a batch job management system must have the ability of predicting the run time baseline of batch jobs, which also called offline prediction. Besides, if a batch job run time exceed the baseline, it cannot remedy. Thus, a batch job management system must accurately predict batch job run time in an online mode. We present some major challenges.

- **Accurate offline prediction:** It is intuitive that the workloads of batch jobs can significantly impact the jobs' run time given the same running environment. Besides, a batch job comprises a number of tasks, which may have complicated dependency relationships. The orchestration of these tasks changes over date. Therefore, the baseline run time may vary because of workloads and date, which need to be accurately predicted.
- **Agile online prediction:** It is challenging to precisely predict a batch job's run time due to the co-location of batch jobs and online services. Workload values of online services, which are given the first priority to ensure the quality of services, very often have spikes and fluctuations. Consequently, certain resources of running batch jobs may be occasionally taken by online services. Thus, batch jobs' run time can greatly vary. let alone there are thousands of batch jobs in our partnered banking IT system. Hence, we look forward to an agile online prediction in order to scale up resources when they are needed.
- **Auto-scaling strategy:** If a batch job's predicted run time exceeds the baseline, we must design an auto-scaling approach so that the job is guaranteed to finish on time. This approach should be robust enough to allocate minimum resources as well as finish batch jobs on time.

To address the aforementioned challenges, in this work, we design a framework *Batman* to predict batch jobs run time and auto-scale resources in the cloud. First, we characterize three years' records of batch jobs' run time. To accurately predict run time baseline, we summarize the pattern of critical paths which are series of tasks adding up to the longest overall duration for each run of a batch job. Then, we adopt a CNN-LSTM model to predict online batch jobs' run

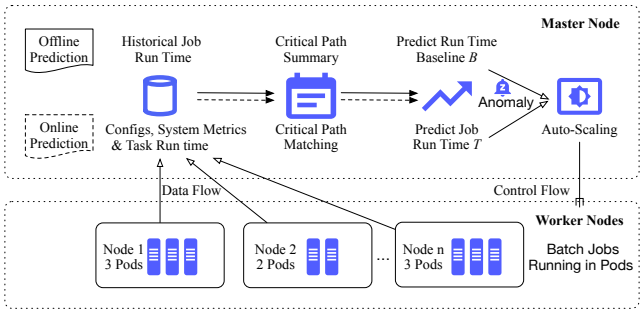

Figure 1: Framework of Batman.

time with the input of system metrics collected from batch job running environment. Finally, if online prediction time is more than baseline, an auto-scaling strategy is utilized to accelerate this job to finish on time with minimum resource allocation. This part of experiments are implemented in Spark and Kubernetes with the workload of TPC-DS (Nambiar and Poess 2006) to simulate the batch jobs.

Design

As Fig.1 shows, our framework consists of a master node and worker nodes. A node is a machine or virtual machine. A pod is the smallest creation and deployment unit comprising one or more containers that are tightly coupled and share resources. Batch jobs are running in pods of worker nodes.

Offline Prediction

Based on three years' batch job records from a banking system, we observe that the run time of a batch job highly depends on its component tasks. The dependency of these tasks are very complex. Specifically, one task may depend on another one or more task(s) to be executed or depend on a specific time to be executed. Moreover, some of the tasks are executed in a periodic standard. For instance, some tasks are run only on a particular date of a month and this can be automatically summarized with a calendar. Therefore, we generate a critical path of tasks, *i.e.*, a series which accumulates and represents the overall longest duration for each run of a batch job. The pattern of a critical path can be summarized by its period. With the input of historical job run time and critical path, Batman adopts Long Short-Term Memory (LSTM) and can predict run time baselines.

Online Prediction

In the online mode with a batch job, Batman can match a critical path according to its period. We observe that the system metrics collected from the batch job running environment can indicate a job's run time. However, the relationship between system metrics and job run time is sophisticated and non-linear. With the input of configurations of the batch job (*e.g.*, the number of nodes and pods), system metrics, run time of each task on the critical path and each task's execution time (*i.e.*, the day, week, month and holiday information), Batman utilizes Convolutional Neural Network (CNN)

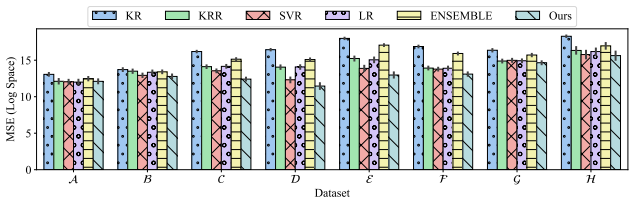

Figure 2: Batch job run time offline prediction result.

Table 1: Summary Coefficient of Variation (CoV) of batch jobs prediction dataset.

Dataset	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{H}$
CoV	0.07	0.13	0.29	0.44	0.81	0.98	1.43	3.93

layers for input feature extraction combined with LSTM to support sequence predictions in a constant time.

Auto-Scaling

For a running batch job, Batman predicts the offline baseline once, denoted as B. Batman also predicts the run time in an online mode and obtains T with the tasks in execution. If $T - B > \gamma$, where γ is a preset threshold, Batman will control the system to scale up pods or scale out nodes based on online prediction.

Preliminary Results

We choose eight datasets of batch jobs with three years' records from the bank sampled based on their coefficient of variation (CoV) distribution. As shown in Tab.1, the larger the values, the more fluctuations in the datasets which are thus more difficult to predict. We implement a prototype of Batman and evaluate the offline prediction model against five other baseline methods. We use the average logarithm of the Mean Squared Error (MSE) as the metric to measure the accuracy of these prediction model. The smaller the MSE, the higher the prediction accuracy. We use one-hour prediction horizon in this experiment. As Fig.2 shows, we compare Batman with five algorithms, *i.e.*, Kernel Regression (KR), Kernel Ridge Regression (KRR), Epsilon-Support Vector Regression (SVR), Linear Regression (LR) and Ensemble method (of KRR, SVR and LR). We note Batman is more accurate than other algorithms on all datasets. Using Batman, we can robustly predict the run time of batch jobs. More design and evaluation in terms of online prediction and auto-scaling are left for future work.

References

Chase, J.; Nguyen, D. T.; Sun, H.; and Lau, H. C. 2019. Improving law enforcement daily deployment through machine learning-informed optimization under uncertainty. In *Proceedings of the 28th AAAI*, 5815–5821. AAAI Press.

Ferguson, A. D.; Bodik, P.; Kandula, S.; Boutin, E.; and Fonseca, R. 2012. Jockey: guaranteed job latency in data parallel clusters. In *Proceedings of the 7th EuroSys*, 99–112. ACM.

Nambiar, R. O., and Poess, M. 2006. The making of tpc-ds. In *Proceedings of the 32nd VLDB*, 1049–1058. VLDB Endowment.