2017 IEEE 33rd International Conference on Data Engineering

AdaStorm: Resource Efficient Storm with Adaptive Configuration

Zujian Weng*, Qi Guo†, Chunkai Wang*, Xiaofeng Meng* and Bingsheng He‡
*School of Information
Renmin University of China, Beijing, China
†Institute of Computing Technology
Chinese Academy of Sciences, Beijing, China
‡National University of Singapore

***Abstract*—Storm is a popular real-time processing system. However, our earlier experiment shows that the fixed configuration of Storm would lead to either significant resource waste or limited processing throughput. In this demonstration, we present AdaStorm, a system to dynamically adjust the Storm configuration according to current data stream properties. AdaStorm is designed to minimize the resource usage while still ensuring the same or even better real-time response. We will demonstrate that AdaStorm can achieve resource efficiency as well as data rate tolerance, compared to Storm system with fixed configuration. Video: https://youtu.be/YFPBFNdMbXM.**

I. Introduction

Storm [1] has been widely adopted by many companies in their business scenarios [2]. In current Storm, the topology configuration (e.g., the number of workers) is predetermined and the setting may not be aware of data stream properties (e.g., arrival rate). Hence, for continuous data flow with fluctuant stream properties, it may result in significant resource waste. Our earlier experiment shows that about 10% CPU and 50% memory resources are wasted when fixing the topology configuration [3]. Taking the Amazon EC2 as an example, the above resource waste will result in additional $300 per month per machine. Moreover, fixed topology configuration might also constrain the processing throughput if an overwhelming data flow exceeds the system capability.

In this demonstration, we present AdaStorm, a machine learning based resource-efficient system to dynamically adjust the Storm topology configuration according to the current data flow rate. Compared to the resource-aware scheduler (i.e., R-Storm [4]) adopted by Storm 2.0, which increase the overall throughput by *maximizing* the resource usage, AdaStorm is designed to *minimize* the resource usage while still satisfying the real-time constraint. This is achieved by using a two-phase mechanism. First, multiple resource usage models are trained incrementally with accumulated historical data. Then, whenever necessary, these models can be used for deriving the most resource-efficient topology configuration to be deployed. We will demonstrate two scenarios on the effectiveness of AdaStorm, that is, resource efficiency and data rate tolerance, with the GeoLife GPS Trajectories dataset [5].

II. AdaStorm: Architecture and Implementation

Figure 1 illustrates the overall architecture of AdaStorm. The key is the *Configuration Module*, consisting of four components. The first *Behavior Process* module treats streaming

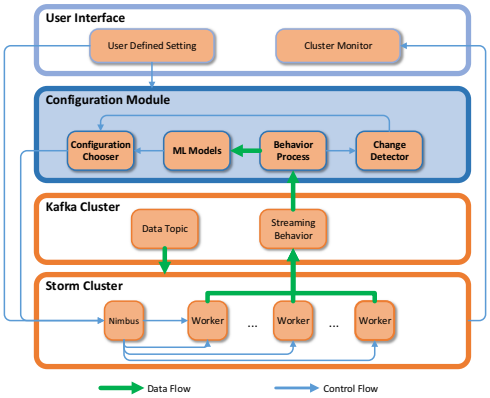

Fig. 1: AdaStorm Architecture

behaviors such as average data producing and consuming rate during the topology execution as new samples to build/update *machine learning (ML) Models*. Then, the models are used by the *Configuration Chooser* module for predicting the optimal configurations. Whether to deploy the configurations is determined by the *Change Detector* module based on collected system behaviors. Through the *User Interface*, users can configure both the Storm cluster and configuration management.

A. Model Construction

We decide to construct models for CPU usage, memory usage, processing throughput and processing latency since they are key metrics for evaluating the execution efficiency of Storm. According to a series of experiments and specific domain knowledge, we choose features from the following four aspects: cluster feature, task feature, topology feature and data feature. The learning models can be trained/updated in two stages. In the first *training* stage, AdaStorm randomly adjusts features in the above categories at a fixed time interval, in order to collect diverse samples. In the second *updating* stage, when running the actual topologies, AdaStorm continuously collects new samples to update the initial models for generating more accurate ones.

The constructed models fall into two categories, i.e., *regression* and *classification* models. The regression models are constructed for predicting the CPU and memory usage. Based on our experiments, the models built with about 300 samples can

2375-026X/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDE.2017.178

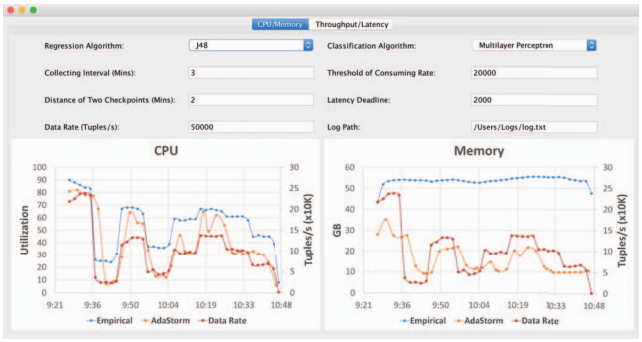

Fig. 2: Comparison of AdaStorm and the original Storm with empirical configuration.

achieve around 90% accuracy by using Multilayer Perceptron. For predicting the processing throughput and latency, we build binary classification models to get whether the throughput or latency can meet the system specification (e.g., real-time deadline). Similarly, with 300 samples, the built models can achieve about 90% classification accuracy by using the J48 algorithm.

B. Model Deployment

During model deployment, the constructed models are used for determining the optimal topology configuration. When topologies are running, the "Change Detector" (see Figure 1) will analyse data from "Behavior Process" and send request to "Configuration Chooser" if (1) the current data consuming rate significantly differs from previous data consuming rate or (2) the data producing rate is higher than the consuming rate to a certain extent. Receiving request, the "Parameter Chooser" exhaustively predicts the CPU usage, memory usage, processing latency and throughput for all possible configurations. Based on the predicted results, the optimal configuration is defined as the one using minimal CPU and memory resource, while the throughput and the latency requirements are still guaranteed for more formal information). Finally, the old topology will be deactivated and then switch to a new topology with the optimal configuration.

III. Demonstration

We will use a subset of the GeoLife GPS Trajectories dataset [5], which contains the real-time GPS data collected from taxis and buses, to simulate the traffic monitoring (TM) system. We will use a 5-node computing cluster of Storm, each node has 24 CPU cores and 32 GB memory. In addition, another 5-node cluster served as the Kafka and Zookeeper cluster. To demonstrate the effectiveness of AdaStorm, we will show two scenarios on a cluster with real workloads.

Training initial models. We will firstly show how AdaStorm trains the initial ML models offline with randomly generated samples. To help participants understand how to use AdaStorm in practice, we will showcase how to implement the AdaStorm interface, upload JAR packages and run topologies.

Scenario 1: Resource efficiency. This scenario is to show the resource efficiency of AdaStorm. Given the same cluster configuration and data producing rate, we will show how much resource can be saved by AdaStorm while still ensuring real-time response, in contrast to the original Storm with the fixed configuration according to rules of thumb (e.g., setting the worker number as the number of CPU cores), aiming to *offer high throughput*. The data producing rate is simulated under three circumstances: (1) data rate changes sharply, (2) fluctuates and (3) specified by audience. Through our user interface we will show the comparison of CPU/memory usage, throughput and latency. We will observe how storm's configuration changes and the reason why it changed. Besides, we will change some setting (e.g. threshold of consuming rate) through user interface and see how the AdaStorm will behave.

Figure 2 shows the resource comparison between AdaStorm and the original Storm, given fluctuant data rate. It shows that AdaStorm reduces the CPU and memory usage by about 15% and 60% respectively.

Scenario 2: Data rate tolerance. In this scenario, we compare AdaStorm against the original Storm with fixed configuration, aiming to *save resource usage*. The participants will see when the data rate *increases* to some extent, the original Storm cannot process data given a stringent deadline. On the contrary, AdaStorm can adapt to increasing data producing rate through adaptive configuration.

IV. Conclusion

In this demonstration, we present AdaStorm, a resource-efficient Storm system with adaptive topology configuration, so as to adapt to fluctuant arrival rate of data. Through our demonstration of AdaStorm with two scenarios, we will illustrate that: (1) AdaStorm can significantly reduce hardware resource usage and (2) more adaptive to fluctuate data. Hence, it is promising to deploy AdaStorm in production to significantly increase users' ROI.

Acknowledgment

This research was partially supported by the National Key Research and Development Program of China (No. 2016YFB1000602, 2016YFB1000603); The grants from the Natural Science Foundation of China (No. 61532016, 61532010, 61379050, 91646203); The 2017 Opening and Cooperation Project of Science and Technology in Henan (No. 172106000077); Specialized Research Fund for the Doctoral Program of Higher Education (No. 20130004130001), and the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University (No. 11XNL010).

References

[1] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham *et al.*, "Storm@ twitter," in *SIGMOD*, 2014, pp. 147–156.

[2] http://storm.apache.org/documentation/Powered-By.html.

[3] C. Wang, X. Meng, Q. Guo, Z. Weng, and C. Yang, "Orientstream: A framework for dynamic resource allocation in distributed data stream management systems," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 2281–2286.

[4] B. Peng, M. Hosseini, Z. Hong, R. Farivar, and R. H. Campbell, "R-storm: Resource-aware scheduling in storm," in *Middleware*, 2015, pp. 149–161.

[5] Y. Zheng, X. Xie, and W.-Y. Ma, "Geolife: A collaborative social networking service among user, location and trajectory." *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.