Towards automatic configuration of big data processing systems

Keywords : Automatic Configuration; Optimization; Big data; Performance.

Muhammad Bilal, Marco Canini

Abstract – The goal of this research is to develop a generic framework for automatic configuration optimization of big data processing systems. The framework will allow users to specify their performance goals as well as their own metrics. We plan on utilizing system developer provided hints for searching suitable configurations for big data application deployments.

Introduction

Big data processing systems have dozens of available configuration parameters and system performance crucially depends on tuning several of these parameters. Moreover, the suitable configurations vary depending upon the available resources, workloads and applications for which the system is being used. Today, optimization of these configurations is done manually, possibly requiring several hours of investigation and testing by performance engineers. Performance engineers also need to have detailed knowledge about the big data system under consideration. This makes configuration optimization a tedious and time consuming task. Most of existing works present solutions specific to MapReduce jobs [2,3]. However, to the best of our knowledge the broader research challenge remains open.

Preliminary Analysis

To understand the variation in performance due to system configurations, we conducted preliminary tests on a multi-node Apache Storm cluster. The setup includes 3 nodes, with the capability to run 96 threads simultaneously. We use the Rolling word count topology of the Intel Storm benchmark [1], with count window size set to 10s and count emit frequency set to 1s.

Configuration	Possible values
Number of workers and acker threads	3-12 (each)
Spout, Split Bolt and Count bolt threads	3-64 (each)

Table 1 - Selected configurations and their values.

Following an Experimental Design approach, we use a space filling algorithm to cover the configuration space. Table 1 shows the space of possible configurations considered for each experiment. We have fixed the total number of threads in the topology to 96. We vary the number of threads assigned to each of the topology component while keeping the total number of threads constant. We used the WSP space filling algorithm [4] to generate 77 configurations for our experiments.

Figure 1 shows the variation of different performance metrics across different configurations. The per-tuple latency varies between 3ms to 108ms at the median and from 3ms to 577ms at the 99th percentile. In addition, there is a substantial variation in throughput from 10k tuples/s to 117k tuples/s. Thus we observed a variation as high as 2 orders of magnitude in latency and 1 order of magnitude in throughput, given a fixed resource budget. The interplay between parallelism of different Storm components has lead to this drastic variance in performance. The amount of work done by each component of the topology and its processing latency, the number of executor for each bolt/spout and their placement, all play a part in the resulting overall performance.

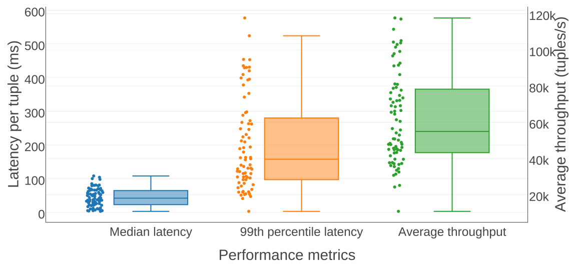

Figure 1: Performance results for selected configurations.

Research Directions

In the current form our tool is able to provide recommendations for bolt parallelism configurations in Apache Storm, according to a performance metric. Two types of recommendations are made. First, a configuration that achieves the best observed performance. Secondly, a configuration that achieves performance within the tolerance level (specified by the user) of the first configuration while using less resources.

Beyond Storm, our goal is to design a generic framework that can be used and extended to provide automatic configuration for a large variety of big data frameworks. To achieve this, there are two main interface that need to defined and refined. An interface offered to the users of the framework and an interface for the system developers of the big data processing frameworks. The user interface should provide ways for the user to provide their own performance metrics as well as performance goals that can be transformed into a fitness function for the evaluation of different configurations. On the other hand, the interface for the system developer should provide information about the key configuration parameters, system-specific metrics APIs and performance metrics. We will use this information to automatically improve the application-specific configurations, instead of forcing users to understand the fine details about a big data system. Lastly, we plan on expanding the available search algorithms to incorporate evolutionary algorithms and other mechanism to give user a wide range of choices.

References

[1] Intel storm benchmark. https://github.com/ intel-hadoop/storm-benchmark. Accessed: 2016-02-01.

[2] S. Babu. "Towards automatic optimization of MapReduce programs". In SoCC, pages 137-142. ACM, 2010.

[3] H. Herodotou and S. Babu. "Profiling, what-if analysis, and cost-based optimization of MapReduce programs". VLDB Endowment, 4(11):1111-1122, 2011.

[4] J. Santiago, M. Claeys-Bruno, and M. Sergent. "Construction of space-filling designs using wsp algorithm for high dimensional spaces". Chemometrics and Intelligent Laboratory Systems, 113:26-31, 2012.