Demo: Measuring and Estimating Monetary Cost for Cloud-based Data Stream Processing

Thomas Heinze[1] Patrick Meyer[1] Zbigniew Jerzak[1] Christof Fetzer[2]

[1]SAP AG
Dresden, Germany
firstname.lastname@sap.com

[2]System Engineering Group
Technische Universität Dresden, Germany
christof.fetzer@tu-dresden.de

ABSTRACT

In recent time due to the availability of cloud-based data streaming systems like Yahoo! S4 or Twitter Storm and virtually unlimited resources using a public cloud infrastructure it is possible to run stream processing tasks with a new dimension of computational complexity. However, the required resources in terms of CPU, memory, and network bandwidth differ depending on the use case and applied data streaming system. For the user of such a system this is directly visible in the monetary cost he has to spent for the used resources. Therefore, he would like to maximize the ratio between gained performance and his monetary cost.

In our demonstration we present an approach to measure and estimate the monetary cost for data streaming systems. We present a general scheme to model monetary cost for any combination of a cloud-based data streaming system and a major public cloud provider. Our model can be used as a starting point for optimizing the ratio between monetary cost and performance of streaming systems in general.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

Keywords

Monetary Cost; Cloud-based data stream processing

1. INTRODUCTION

In the past, the major metrics for evaluating a data streaming system have been throughput, latency, and computational accuracy. By the raise of cloud computing, offering an illusion of unlimited resources, and data streaming systems like Twitter Storm [1] and Yahoo! S4 [6], which allow to scale across a large number of hosts, such performance goals become more easily achievable. However, the required resources to process such tasks depend on many different factors like the streaming system and the use case [2]. In context of cloud computing a "pay per use" model is used. This causes that the different required resources are directly visible to the user by the bill he has to pay for his cloud setup. Therefore, a user is interested in improving the ratio between spent cost and provided quality of service. However, currently no tool to analyze the monetary cost for data streaming systems is available.

In this demonstration we present a model to measure and estimate the required cost for applications on top of cloud-based data streaming systems. Our model can be used with an arbitrary combination of a cloud provider and a data stream processing engine. It measures the monetary cost on a fine-granular level, allowing to find the most costly parts of a query. The demonstration comprises a graphical user interface showing detailed information on the spent monetary cost.

The problem of monetary cost as a new metric for data management systems has been outlined by Florescu et al. [3]. Kossmann [5] shows, that the monetary cost of executing a transactional database workload differs based on the used architecture as well as the cloud provider cost model. We are only aware of one work [4], which tried to optimize a data streaming system deployment towards minimizing the trade-off between cost and latency, however, it is specific to the used system and cloud environment.

2. MONETARY COST

The monetary cost to pay for a given setup depends on the cost model of the cloud provider, which differs between the various providers. For defining a generic cost model we studied the pricing schemes of the major public cloud vendors. All of them calculate the cost by the used RAM, CPU, and/or network capacity. We model the prices for storage as fixed cost for our system, because data streaming systems are using mostly in memory computing. Each provider uses individual intervals for charging the cost, varying between price per month to price per hour, and provides different hardware images. We scale the prices to the same time scale (1 hour) and resource size (1 core, 1 GB RAM, 1 GB network traffic). In summary, the following parameters need to be specified for a given cloud cost model: the price p_{CPU} for using one CPU core for one hour, price p_{RAM} for using a gigabyte of main memory for one hour and the prices $p_{NET_{\leftarrow}}, p_{NET_{\rightarrow}}$ per gigabyte for incoming or outgoing network traffic.[1]

[1]Due to space constraints we omit the details on how to model special offerings, different types of hosts (on demand, reserved instances) and fixed cost.

DEBS'13, June 29–July 3, 2013, Arlington, Texas, USA.

We assume that the data streaming system can be used by one or more users in parallel. The users deploy queries on top of the set of available hosts, where an individual host can be shared by several users. A query describes a certain computation task, which can be split into several atomic operations like filter, aggregation, or join modeled as operators.

Our model is able to calculate the cost to pay on three different levels: the total cost, the cost per query, and the cost per operator inside a query. The total cost p for the current setup can be given as $p = used_{CPU} \cdot p_{CPU} + used_{RAM} \cdot p_{RAM} + used_{NET_{\leftarrow}} \cdot p_{NET_{\leftarrow}} + used_{NET_{\rightarrow}} \cdot p_{NET_{\rightarrow}}$, where $used_{CPU}$, $used_{RAM}$, $used_{NET_{\leftarrow}}$ and $used_{NET_{\rightarrow}}$ represent the number of used CPU's, RAM, and network bandwidth respectively. To calculate the cost per operator, the cost for one host is split proportionally to the used RAM and CPU for all operators running on the host like shown in Figure 1. For the network consumption the used bandwidth is measured. The cost for a single query can be calculated by summing up the costs of all its operators. The different usage metrics are constantly measured and the cost values are updated accordingly. The calculated cost value expresses always the expected cost for the current cost interval (the current hour), the concrete cost is derived at the end of an interval and added to the total cost to pay.

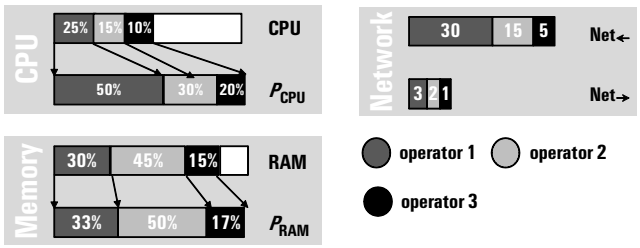
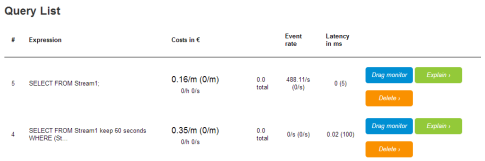


Figure 1: Calculating the cost per operator

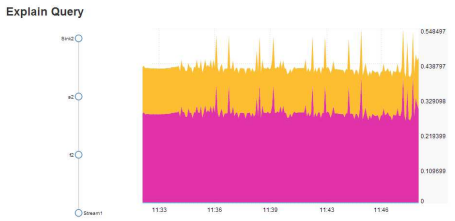
Using our model we are also able to estimate the cost for executing a query before deploying it on top of the system. Therefore, we use current system information about the performance of already running queries e.g. the required processing time for filtering or aggregating events. In addition, we use an approach similar to Viglas et al. [7] to estimate the event rates per operator. By combining these two measures the required resources are estimated from which the spent cost can be derived. The cost estimation allows a user to judge the cost of queries before actually deploying them and to rewrite a query if needed. In addition, a user can specify quality of service (QoS) constraints like maximal latency, minimal throughput or required availability.

3. OUR DEMONSTRATION

To illustrate our model, we built a demonstration system presented in Figure 2, where a user can visualize the monetary cost for a cloud-based data streaming system. A user-defined query can be added to a system with certain QoS constraints, and is executed on top of a commercial cloud-based streaming system. We run the system in a private cloud environment with up to 10 VM's and we are able to switch between different cost models within the demonstration. Based on constantly measured memory, network, and CPU consumption the cost values are updated to reflect e.g. a changing input rate of the query and can be compared with the estimated cost. In addition, a detail view similar to an explain functionality of a database is provided, which presents a break down of the cost of a query into cost per operator to allow to identify the most expensive parts of a query.



(a) List queries view



(b) Detailed cost view

Figure 2: Screenshots of our demonstration

4. CONCLUSION

In this demonstration we presented a generic cost model, which allows to measure and estimate the monetary cost for a cloud-based data streaming engine. It can be used as a starting point for improving the ratio between monetary cost and achieved performance.

5. REFERENCES

[1] Twitter Storm: http://storm-project.net/.

[2] M. Dayarathna, S. Takeno, and T. Suzumura. A performance study on operator-based stream processing systems. In *Workload Characterization (IISWC), 2011 IEEE International Symposium on*, pages 79–79, 2011.

[3] D. Florescu and D. Kossmann. Rethinking cost and performance of database systems. *ACM SIGMOD Record*, 38(1):43–48, 2009.

[4] A. Ishii and T. Suzumura. Elastic stream computing with clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 195–202, 2011.

[5] D. Kossmann, T. Kraska, and S. Loesing. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 2010 SIGMOD International Conference on Management of Data*, pages 579–590, 2010.

[6] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari. S4: Distributed stream computing platform. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 170–177, 2010.

[7] S. D. Viglas and J. F. Naughton. Rate-based query optimization for streaming information sources. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of data*, pages 37–48, 2002.