Online Parameter Optimization for Elastic Data Stream Processing

Thomas Heinze[1], Lars Roediger[1], Andreas Meister[2], Yuanzhen Ji[1], Zbigniew Jerzak[1], Christof Fetzer[3]
[1]SAP SE {firstname.lastname}@sap.com
[2]University of Magdeburg andreas.meister@iti.cs.uni-magdeburg.de
[3]TU Dresden christof.fetzer@tu-dresden.de

A major problem of today's cloud infrastructure is the low utilization of the overall system [3], which results in both high cost for the user and low energy efficiency of the cloud itself. This is caused by the fact, that a user typically needs to define a scaling strategy by configuring the number of used machines, a set of thresholds, or a controller to decide when the system should scale in or out. However, often a user has a limited understanding of the used system and the workload. Therefore, he chooses the scaling scheme conservatively to be able to handle peak loads and as a result achieves a very low system utilization.

This observation also holds true for data stream processing systems, which continuously produce output for a set of standing queries and a potentially infinite input stream. Many real-world workloads for data stream processing systems have a high variability, which means the data rate and selectivities of the streams are frequently changing in an unpredictable way. Several authors proposed data stream processing prototypes, which automatically scale in or out based on workload characteristics, to handle these dynamic workloads. Such systems are called elastic and allow to increase the system utilization by only using the minimal required number of hosts. However, in all of these prototypes the user needs to manually specify a scaling strategy. In our previous work [2], we illustrated that an adaptive auto-scaling technique is able to improve the utilization of the system, but degrades the quality of service. Each reconfiguration decision in a data stream processing system interferes with the data processing and as a result has a high impact on major quality of service metrics like the end to end latency. Therefore, this characteristic needs to be reflected in the scaling strategy to achieve a good trade-off between the monetary cost spent and the achieved quality of service.

In this paper we address the problem of choosing the scaling strategy for a data stream processing system by introducing a novel approach based on online parameter optimization. We make the following contributions:

1. We present an online parameter optimization approach, which automatically (1) chooses the scaling strategy to minimize the number of hosts used for the current workload characteristics, (2) detects changes of the workload pattern, and (3) adapts the scaling policy accordingly. Our system removes the burden from the user to manually configure the scaling policy.
2. We show how this parameter optimization problem can be enhanced with a given quality of service constraint for the maximum end to end latency. This constraint is used during the optimization to improve the trade-off between monetary cost and quality of service.
3. We evaluate our prototype based on three real-world use cases. We show that we reduce the average cost by 19% compared to a naive scaling scheme and by 10% compared to a manually tuned scaling strategy with a comparable or even better quality of service. We also show that our solution outperforms a state of the art adaptive auto-scaling technique based on Reinforcement Learning due to the more precise modeling of the scaling behavior of a data stream processing system.

The solution presented in this paper consists of three major components: an elastic scaling data stream processing engine, an online profiler, and a parameter optimization component.

We extend an existing elastic data stream processing engine [2], which scales automatically in and out based on a user-defined scaling policy and the current workload characteristics. The scaling policy uses threshold-based rules for the maximum and minimum utilization of a host or the system respectively. These thresholds and some additional parameters typically have to be chosen manually by the user to fit the specific use case characteristics.

Our optimization component finds automatically a good parameter configuration based on a short-term utilization history during the system runtime and reoptimizes it, if required. We define a cost function as well as a parameter search space, and solve a parameter optimization problem using an improved random search algorithm [4].

We also introduce an online profiler, which determines the frequency of triggering the parameter optimization. It monitors changes of the workload pattern based on the overall CPU load using an adaptive window [1]. If a change of the workload pattern is detected, the optimization component is triggered using an up to date short-term history of current load characteristics.

References

[1] A. Bifet and R. Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, SDM 2007, pages 443–448, 2007.

[2] T. Heinze, V. Pappalardo, Z. Jerzak, and C. Fetzer. Auto-scaling techniques for elastic data stream processing. In *Workshops Proceedings of the 30th International Conference on Data Engineering Workshops*, ICDEW 2014, pages 296–302. IEEE, 2014.

[3] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC 2012, page 7. ACM, 2012.

[4] T. Ye and S. Kalyanaraman. A recursive random search algorithm for large-scale network parameter configuration. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS 2003, pages 196–205. ACM, 2003.

Online Parameter Optimization for Elastic Data Stream Processing

Thomas Heinze, Lars Roediger, Andreas Meister, Yuanzhen Ji, Zbigniew Jerzak, Christof Fetzer

Goal: **Elastic Scalability**

Low system **utilization** → Need to scale to **varying number of hosts** → Automatically define parameters for a **scaling policy**

Solution Outline

Search best parameter configuration online based on short-term utilization history.

(1) Detect workload change | **(2)** Search best scaling policy | **(3)** Adapt parameters during runtime

Cost Function

Simulate behavior of scaling policy for **short-term utilization history**

INPUT

utils

	t_0	t_1	t_2
S1	0.4	0.5	0.4
A1	0.3	0.5	0.4
D1	0.3	0.4	0.4

assign(t_0)

	ops
H1	{ S1, A1}
H2	{D1}

*p**

thres↓	0.3	1 *d*↓
thres↑	0.8	1 *d*↑
g	1	FF bin

CALCULATION

t_0

	ops	*∑util*
H1	{ S1, A1}	0.7
H2	{D1}	0.3

t_1

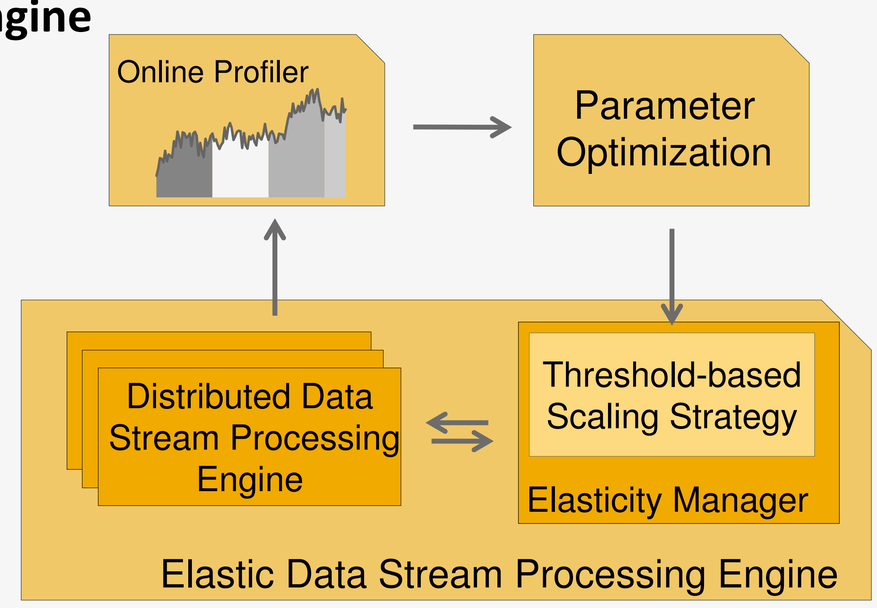
	ops	*∑util*
H1	{ S1}	0.5
H2	{D1}	0.4
H3	{A1}	0.5

t_2

	ops	*∑util*
H1	{ S1}	0.4
H2	{D1}	0.4
H3	{A1}	0.4

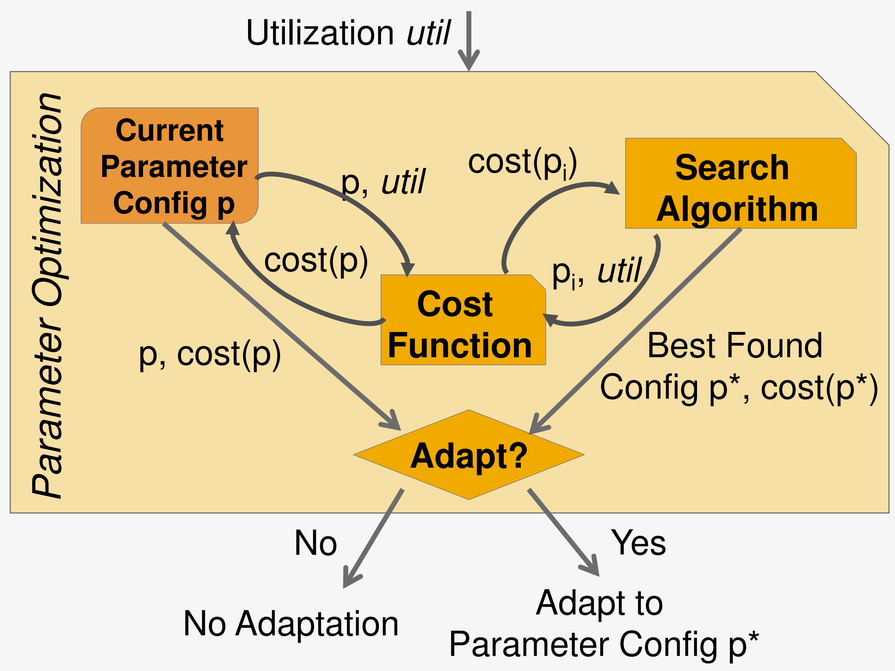
Architecture

Extends an **existing elastic data stream processing engine**



Parameter Optimization

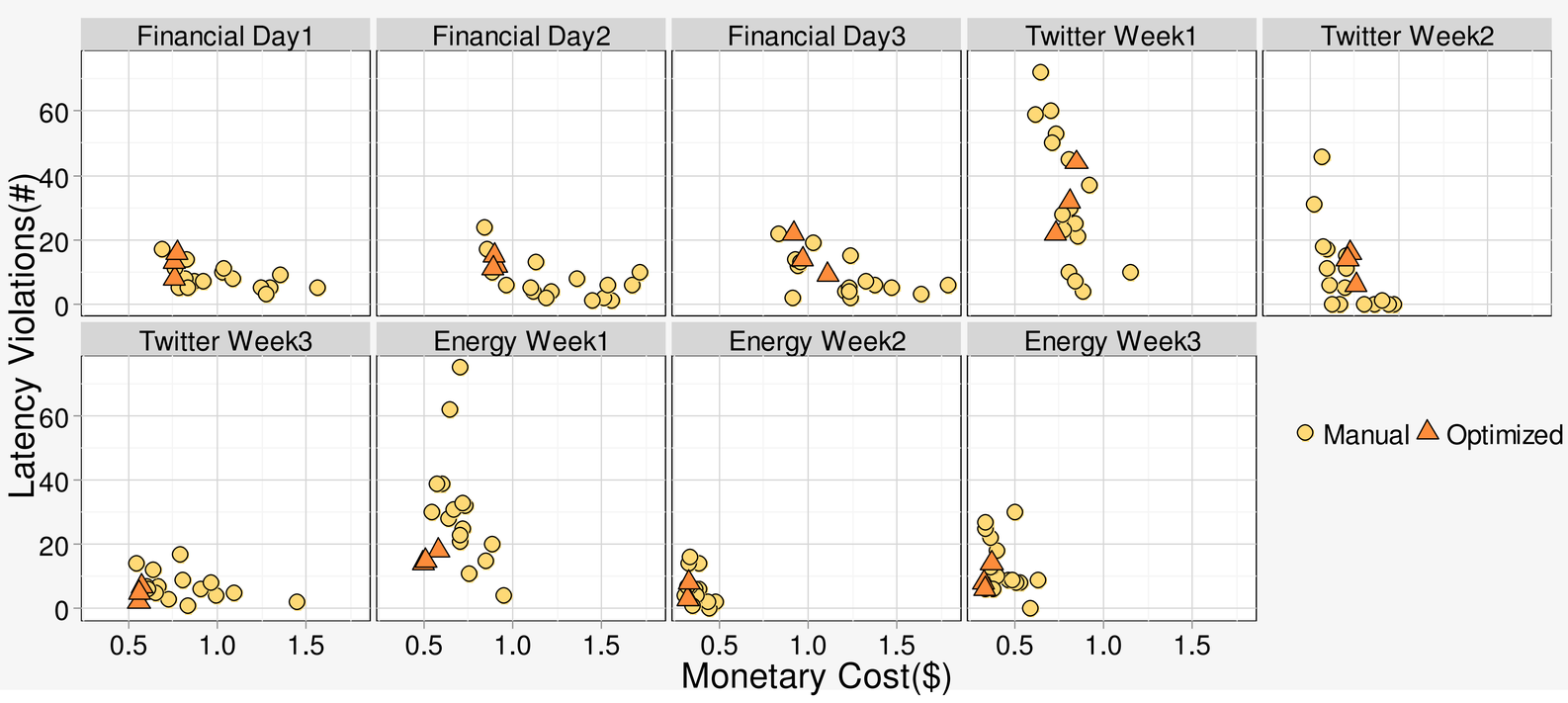
Search best parameters based on **Random Search**



Evaluation

Comparison with **manual-tuned thresholds** for **three real-world Scenarios.**

Improve cost by **19%/10%** compared to naïve/best configuration.



SoCC 2015, Hawaii

TECHNISCHE UNIVERSITÄT DRESDEN