Cost-Aware Query Optimization during Cloud-Based Complex Event Processing

Andreas Meister, Sebastian Breß, Gunter Saake
firstname.lastname@ovgu.de
University of Magdeburg, Germany

Abstract: *Complex Event Processing* describes the problem of timely and continuous processing of event streams. The load of Complex Event Processing systems can vary (e.g., event rates). Static resource provision leads to higher monetary costs because enough resources have to be provided to efficiently handle peak loads. Therefore, most of the time the resources will not be fully utilized.

One way to achieve scalable processing and elastical resource allocation fitting varying requirements is to use *Cloud Computing*. Properties of Cloud Computing are the pay-as-you-go-payment model and high availability. These properties can be used in Complex Event Processing systems to minimize the monetary costs of systems while satisfying Service Level Agreements. Complex Event Processing systems must continuously optimize the event processing to adapt to varying loads without violation of Service Level Agreements. To guarantee efficiency, the optimization cost must be considered, leading to cost savings without violating the Service Level Agreements.

In this work, we discuss factors, which should be considered during the optimization of cloud-based Complex Event Processing systems that use the pause-train-resume strategy to migrate operators. Furthermore, we propose heuristics to estimate the cost of these factors. In our experiments, the cost could be decreased by 15 % by using a cost-aware optimizer. This proofs that the costs of cloud-based Complex Event Processing systems can be further decreased if optimization is cost-aware.

1 Introduction

Complex Event Processing (CEP), the problem of timely and continuous processing of event streams, becomes increasingly important (e.g., in stock markets [BDG07] or financial applications [ScZ05]). The different application fields have in common that the system loads vary during runtime. Additionally, certain characteristics agreed in *Service Level Agreements* (SLAs) must be guaranteed (e.g., event processing time). CEP systems can use *Cloud Computing* to provide resources elastically (e.g., adding or removing additional nodes) to adapt to changing workloads and to save resources (and hence, money) in case of low system load. Based on the pay-as-you-go-payment of Cloud Computing, elastical resource provision leads to reduced costs, because only the needed amount of resources will be provided. To minimize system costs, a continuous optimization is needed to adapt the CEP system to the current requirements. During the optimization, it might be necessary to migrate operators form over- or underloaded nodes to other nodes of the system.

Problem statement. Operator migration can lead to additional costs. Therefore, the query optimizer should be cost-aware, meaning that the optimizer should consider the cost

and effects of operator migrations during optimization. In this paper we investigate the following research question:

> How much can we decrease the overall cost of a cloud-based CEP system using a cost-aware query optimizer considering the cost of the query optimization?

In this paper, we contribute an overview of effects of operator migrations and identify factors usable to calculate the effects. Furthermore, we present cost models for the effects of operator migrations in the used CEP systems. To evaluate the cost models, an existing optimization framework [Rö12] will be extended by integrating the cost models into the optimization process. Our experiments show that we can save up to 15% monetary costs while increasing robustness of CEP systems w.r.t. Service Level Agreements (SLAs).
The remainder of this work is structured as follows. In Section 2, we briefly describe the background information. We specify our cost models to estimate the effects of operator migrations in Section 3. In Section 4, we discuss our evaluation. We provide an overview of related work in Section 5. In Section 6, we present a conclusion and future work.

2 Background

In this section, we will provide an overview of cloud-based CEP, efficient CEP query optimization, and the process of an operator migration.

2.1 Cloud-based Complex Event Processing

CEP describes the problem of timely, continuous processing of event streams [Luc01]. Goal of CEP is to gain additional information based on single events by searching for known or unknown patterns in event streams. The gained information are used for decision making or starting of new processes (e.g., locking of a credit card by a detected fraud).
The processing of events is based on defined queries. In the context of CEP systems, a query processes events until it is stopped by the system. Therefore, the query execution time is potentially unbounded. Since queries in CEP systems are long running, the probability that the requirements change are high (e.g., event rate of streams, etc.) [MSHR02].
Systems that use a static resource provision cannot adapt to changes in the workload, which can lead to under- or overutilization of the system. *Overutilization* leads to higher query execution times and may lead to inaccurate results, if not all events of the event streams can be processed (load shedding). *Underutilization* leads to inefficient resource usage, because not all available resources are fully occupied, leading to an increased end user fee.
Typically an elastic resource provision (e.g., *Cloud Computing*) can be used to solve the problem of over- and underutilizations and to minimize system costs. Using Cloud Computing, resources can be automatically added or removed during the runtime of the system. Since in cloud-based CEP systems resources are charged according to the usage, the efficient resource usage can lead to lower overall monetary costs and hence, lower costs for the end user. The system has to adapt the usage of the resources according to the current requirements of the system to minimize the overall cost.
In order to minimize the cost of the system an efficient optimization is needed to adapt the resources to current requirements of the CEP system while avoiding SLAS violations.

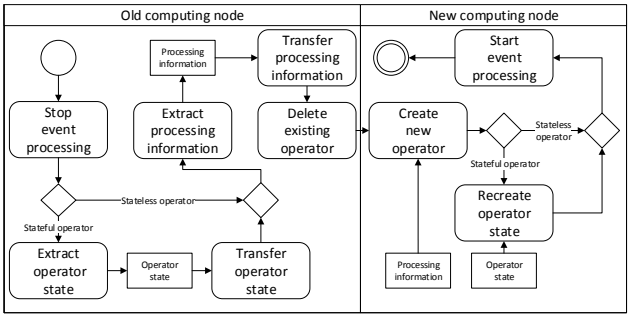

Figure 1: Migration process for stateful and stateless operators

2.2 Efficient query optimization for Complex Event Processing

The goal of query optimization is to ensure the efficiency of query processing. In cloud-based CEP systems, one essential part of the query optimization is the operator placement. The operator placement determines a assignment of existing operators to computing nodes based on the current placement and requirements of the operators (e.g., CPU load).
Since the effects of an optimization can lead to increased costs [BB05] (e.g., additional resource consumption or SLA violations), several aspects must be considered in order to provide an efficient CEP optimizer besides the efficient determination of a new placement: the estimated cost of an optimization, the point in time, when we start the optimization, and the time duration between optimizations.

Cost estimation. In order to provide a cost-aware optimization, an approach is needed that estimates the cost of an optimization. To guarantee the accuracy of the estimation, the cost calculation must consider the properties of the query processing and the operator migration of the used CEP system.

Optimization start time. The optimization causes additional resource consumption. Respectively, the optimization should only be triggered, if the system properties or workload have changed, and the query processing must be adapted to improve the overall efficiency of the system. An approach is needed to estimate, when the query optimization should be started.

Duration between optimizations. Since the optimization cost must be amortized during the runtime, the cost considerations of an optimization should include the duration of the optimized query execution. We cannot consider the remaining execution time, but the time to the next optimization, because the query execution time is potentially unbounded.

There are already suitable approaches to identify a point in time to start the optimization [KD98] and to determine the time duration between optimizations [JHJ+10]. Therefore, we focus on the cost estimation of operator migrations in the remainder of this paper.

2.3 Migration process

To remove underutilized nodes or use added nodes to release overloaded nodes, operators must be migrated between nodes. The decision which operators should be migrated will

be decided by the optimization framework. Hereby, the optimization framework has to determine three factors: (1) a node, from which an operator will be migrated, (2) the operator, which will be migrated, and (3) the target node, where the operator will be migrated to. The process of the operator migration depends on the used system. The considered cloud-based CEP system uses the pause-drain-resume strategy [ZRH04]. Based on the operator type, different steps must be performed to migrate an operator (see Figure 1). The operators can be divided into two groups: stateless and stateful operators. For stateless operators, the following steps must be performed: First, the event processing must be stopped. Second, the processing information has to be extracted and transferred. Third, the existing operator must be deleted on the old node and the new operator must be initialized on the new node. Finally, the event processing must be restarted.
For stateful operators, the state of an operator must be moved additionally. The state of an operator consists of one to n sliding windows based on the number of input streams. Since not all events of an unlimited event stream can be stored, only a selection of the event stream will be stored in a sliding window. The selection is based on a maximal storing time or a maximal number of events.
To migrate the state of an operator, the state must be extracted, transferred and recreated. We illustrate this in Figure 1. To extract the state, the events stored in sliding windows must be read and stored in a transfer format. The recreation of the state requires the processing of all events stored in the extracted sliding windows.
Notably, the described process is based on the used cloud-based CEP system and can vary in other systems. However, in all elastic CEP systems, an operator migration has to be performed, and hence, the basic problem remains the same.

3 Cost-estimation of operator migrations

To estimate the cost of operator migrations during query optimization, several different aspects must be considered. In this section, we describe what effects the operator migration has and specify corresponding cost-models.

3.1 Effects of operator migrations

In order to determine possible effects and costs of the query optimization, we conducted a literature review. To the best of our knowledge, there exists no work in the context of cloud-based CEP about an optimizer that is aware of the optimization cost itself. Since the application field of operator migration in cloud-based CEP systems is similar to virtual machine migration, we adapt existing approaches (see section 5) and apply them for cloud-based CEP systems. We identified three effects of an operator migration in the used CEP system:

Resource consumption. In cloud-based CEP systems, the cost calculation is based on the resource consumption. The operator migration consumes additional resources in order to transfer the operator information, leading to a higher fee for the end-user.

Lost revenue. During the operator migration the event processing must be stopped. Since stopping event processing means that no service can be provided to the costumer, potential revenues are decreased.

Penalty charge. In systems with SLAs, different quality features can be agreed on with the customer (e.g., event processing time). These quality features can be violated during an operator migration because event processing is paused. Therefore, the operator migration may cause additional penalty charges.

The operator migration also influences other aspects, such as the user satisfaction. Since these effects cannot be calculated, they are omitted. In the following sections, we will describe how the different effects of operator migrations can be estimated.

3.2 Resource consumption

The operator movement consumes different kinds of resources. Relevant resources in CEP systems are used CPU time, memory capacity, and network bandwidth capacity. To simplify the following considerations, we do not consider the consumed resources, but the resulting costs. Since the costs of different resources can vary, different constants are used to estimate the cost of the CPU time (mc_{cpu}), memory capacity (mc_{mem}) and network bandwidth capacity (mc_{net}). For stateful operators, the resource consumption for the migration of the state must also be considered. The resource consumption of a state migration is depending on the state size ws and differs from the resource consumption of the migration of the processing information. Therefore, other constants (sc_{cpu}, sc_{mem}, sc_{net}) must be used to estimate the cost for the resource consumption. Since more than one operator can be migrated in one optimization step, the overall cost for the resource consumption mc_d is calculated as sum of all operator migrations:

$$mc_d = \sum_{i=1}^{n} (mc_{cpu,i} + mc_{mem,i} + mc_{net,i}) + \sum_{j=1}^{m} [mc_{cpu,j} + mc_{mem,j} + mc_{net,j} + ws_j \cdot (sc_{cpu} + sc_{mem} + sc_{net})] \quad (1)$$

3.3 Duration of an operator migration

We consider mainly two aspects, to estimate the time of an operator migration mt_o. First, the time to migrate the processing information mt_{oc} (e.g., select or join conditions). Second, the time to migrate the operator state. Since the steps and therefore the time for transferring the processing information is identical for each operator type, we will use a time constant. The time to migrate the state is mainly based on the state size ws, the time to extract and recreate the state st, and the available network bandwidth bw.

$$mt_o = mt_{oc} + ws \cdot st + \frac{ws}{bw} \quad (2)$$

Since the operators cannot be migrated in parallel due to limitations of the used cloud-based CEP system, the time of migrating all operators of a query, is the sum of all migrated

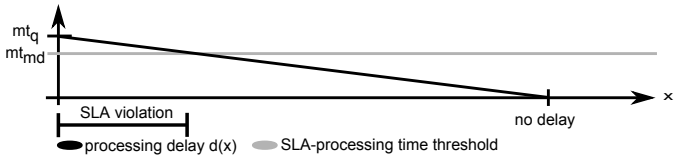

Figure 2: Processing delay $d(x)$ based on the x-th processed event and query migration time mt_q considering the SLAs mt_{md}.

operators ($mt_{o,1}, ..., mt_{o,n}$). Since the used system is a distributed system and only one operator at a time can be migrated, an additional delay mt_d should be considered.

$$mt_q = \sum_{i=1}^{n} (mt_{o,i} + mt_{d,i}) \tag{3}$$

The delay is the time of all operator migrations before the considered operator. Therefore, the delay depends on the order of the operator migration. Since the order cannot be determined in the real system, the worst case will be used for all operators. The worst case for an operator is to be migrated last. In this case, the delay is the sum of the migration time of all other operators of the same node.

$$mt_d = \sum_{i=1}^{k} mt_{oc,k} \tag{4}$$

3.4 Lost revenues and penalty charges

In contrast to a revenue loss, penalty charges can also occur without an optimization (e.g., if a host is overloaded). For this work, we consider only penalty charges caused by the optimization. Lost revenues and penalty charges are both caused by the stop of event processing during the operator migration.
The stop of event processing introduces an additional delay during event processing. The delay is caused by the events stored during the stop of event processing. The storing of events is necessary to guarantee an accurate and correct result of the query processing. In order to avoid the dropping of events (load shedding), enough capacity must be provided to store events. For simplicity, we assume that enough capacity will be provided, and no events must be dropped. Otherwise, the dropping of events must also be considered in the effects of an optimization (e.g., example through a penalty fee for every dropped event).
The number of stored events can be calculated by the event rate of the streams er and the time for the operator migration of the corresponding query mt_q. The behavior of the delay is shown in Figure 2. We assume that after the optimization the delay is decreasing, because enough resources can be provided in a cloud-based CEP system. If enough resources are provided, no node is overloaded. Therefore, more events can be processed than arriving in the system. The processing time of the stored events depends on the processing time of the operators. The delay depends on the order of arrival, because the events will be processed according to that order. The maximal delay is the query migration time mt_q. From this delay, the time of arrival must be subtracted by the event rate and the position of the input stream, $er \cdot x$. Since arriving events are processed in FIFO style, the processing time of

preceding events have to be added., $pt \cdot (x-1)$. The delay of the x-th event can therefore calculated by:

$$d(x) = mt_q - \frac{1}{er} \cdot x + pt \cdot (x-1) \tag{5}$$

The maximal delay mt_{md} of the events without penalty charges is the difference of the maximal processing time lt agreed with the customer and the current query latency ql.

$$mt_{md} = lt - ql \tag{6}$$

Combining equation 5 and 6 leads to:

$$mt_{md} = mt_q - \frac{1}{er} \cdot x + pt \cdot (x-1) \tag{7}$$

This equation can be rearranged to calculate the number of events which have to arrive in order to guarantee that the system response time is smaller than the maximal delay mt_{md}:

$$x = \frac{mt_{md} - mt_q + pt}{-\frac{1}{er} + pt} \tag{8}$$

The number of events can be used to calculate the SLAs-violation time mt_v based on the event rate:

$$mt_v = \frac{x}{er} \tag{9}$$

Inserting equation 8 and rearranging leads to the following equation, which computes the SLA violation time depending on all major factors:

$$mt_v = \frac{mt_{md} - mt_q + pt}{er \cdot \left(-\frac{1}{er} + pt\right)} \tag{10}$$

In general, the processing time between operators differ. A higher event processing time of operators will lead to a higher delay. Since only the maximal delay is of interest, the operator with the maximal processing time will be used to calculate mt_v.
The cost of lost revenues and penalties can be calculated as follows:

$$mc_i = \begin{cases} mt_v \cdot mc_p & \text{if } mt_v > mt_q \\ mt_q \cdot mc_p & \text{otherwise} \end{cases} \tag{11}$$

In case no latency threshold is defined or violated, only the lost revenues during the query migration is considered. Otherwise, the additional time of SLAs-violation is considered. Hereby, mc_p can be a fixed price or a percentage of the cost of the affected query.
The interesting effect is that the system can dynamically decide to violate SLAs or allocate additional resources depending on the caused costs. The overall migration cost considered by the optimization should include the cost for the consumed resources mc_d, the penalty charges, and the lost revenues. Since the cost should be amortized during runtime, the ratio between the estimated costs and periods between the optimizations should be considered during the optimization.

4 Evaluation

In order to answer our research question – *how much can we decrease the overall cost of a cloud-based CEP system using a cost-aware query optimizer considering the cost of the query optimization* – described in the previous sections, we conducted a series of experiments. Hereby, we examined different scenarios with different requirements (query patterns and event rates) based on a real application scenario.

4.1 Experiment Design

There are three different scenarios based on different combination of query patterns and event rates:

- **Scenario 1:** query pattern: linear, event rate: variable
- **Scenario 2:** query pattern: real, event rate: fixed
- **Scenario 3:** query pattern: real, event rate: variable

Since for all queries the same structure (event source, selection, aggregation, and event sink) is used, the load of the system is based on query count. In the linear query pattern, the query count is continuously increased from 1 to a maximum of 33 queries. After reaching the maximum of queries, the query count is constant. In contrast, the real query pattern dynamically changes the query count from 1 to 33 queries. Similar to the query pattern, the event rates can either be fixed ($500\frac{\text{events}}{s}$) or variable (100-1000 $\frac{\text{events}}{s}$). The used events, the real query pattern, and the variable event rates are based on information extracted from the Frankfurt stock exchange market representing real work loads.
In every scenario, five different optimization strategies are evaluated:

- **Baseline:** Optimum
  Considers migration cost: no
- **Variant 1:** Number of considered system states: one
  Considers migration cost: no
- **Variant 1+:** Number of considered system states: one
  Considers migration cost: yes
- **Variant 2:** Number of considered system states: many
  Considers migration cost: no
- **Variant 2+:** Number of considered system states: many
  Considers migration cost: yes

Since the optimization strategies are not the main focus of this work, we refer the interested reader to Rödiger et al. [Rö12].
To have a baseline of the optimization, the search space of the optimization is restricted to a small search space where an optimal solution can be efficiently determined. Hereby, the baseline solution just considers the cost of the current system and the optimized systems ignoring the migration cost. The difference between Baseline and Variant 1 is that Variant 1 uses Recursive Random Search [Rö12] to improve the efficiency of the system. In contrast, Variant 2 analyses several past system states. Variant 1 and Variant 2 were both adapted to consider the operator migration cost (+) using our described cost model (see Section 3).

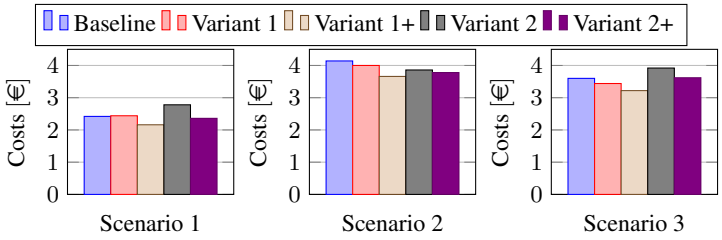

Figure 3: Experiment costs

4.2 Measurement

The experiments were performed on one computing node for measurement and optimization (Intel Core i7 870, 16 GB memory) and four virtual machines (AMD Opteron 6128 HE, 4 GB memory) for the event processing.
Since the load of the system cannot be recreated exactly, and the migration of operators is based on heuristic approaches, a variability of different runs with the same configuration (combination of scenario and optimization strategy) exists. Therefore, every configuration of each experiment is conducted five times. All results for the same configuration are aggregated using the average.
Hereby, the costs and time for the processing of a certain amount of events are measured. The costs are calculated based on the number of used nodes used in fixed time windows. The number of considered time windows n_{tw} is determined by the execution time of the experiment et and the duration of a time dtw

$$n_{tw} = \frac{et}{dtw} \tag{12}$$

For every time window the maximum number of used nodes mn will be determined and charged according to the used pricing model [Mey12] and costs for one node during the time window nc, independent of the real usage of the nodes in the time window.

$$costs = \sum_{i=0}^{n_{tw}} (nc_i \cdot mn_i) \tag{13}$$

The cost calculation is based on commercial cost models (e.g, Amazon E2[1]).

4.3 Evaluation Results

The costs (see Equation 13) for the different scenarios are displayed in Figure 3. In Scenario 1, compared to the baseline, the experiment costs can be reduced if the cost for the operator migration is considered during the optimization. The experiment cost were reduced by 2 % (Variant 2+) to 11 % (Variant 1+). When the different variants with and without the considerations of the migration costs are compared, the consideration of migration costs reduces the costs by 11 % (Variant 1) to 15 % (Variant 2).
In Scenario 2, compared to the baseline, the experiment cost can be reduced if the cost for the operator migration is considered during the optimization. The experiment cost

[1] http://aws.amazon.com/ec2

were reduced by 8 % (Variant 2+) to 11 % (Variant 1+). When the different variants with and without the considerations of the migration costs are compared, the consideration of migration costs reduces the costs by 2 % (Variant 2) to 8 % (Variant 1). Different to the other experiments, variant 2+ only achieves a similar result compared to the baseline in Scenario 3. For variant 1+ the experiment cost were reduced by 10 % compared to the baseline. When the different variants with and without the considerations of the migration costs are compared, the consideration of migration costs reduces the experiments costs by 2 % (Variant 2) to 8 % (Variant 1).

4.4 Discussion

The results of the different experiments show that the monetary costs of the system can be reduced up to 15% if the cost of the optimization is considered during the optimization. The main reason for this is that the number of operator migration is reduced. Hereby, operation migrations are prevented if the gain of efficiency is used up by the cost of the migration. An operator migration will therefore only be executed if it is necessary, for example if a host is overloaded, or if the gain in efficiency exceeds the migration costs. Since the operator migrations are restricted to beneficial migrations, the system needs less time to adapt the query processing. The faster adaption of the query processing leads to an improved efficiency of the overall system resulting in lower monetary cost. The costs of the baseline compared to Variant 1 is unexpected. Normally, the baseline should have better costs in all experiments. Since in two experiments, Variant 1 is better than the Baseline, it seems that the number of experiments is too low to void the effects of the described variety of workload. A gain of 15% is here substantial, because of the potential unbounded runtime of CEP queries. Also the consideration of SLAs during the optimization is leading to a robust optimization, which reduces the probability of SLA violations.

5 Related work

Migration Cost. In the field of virtual machine migration, several relevant effects of a migration were identified. To guarantee the efficiency of the system, the migration itself must be efficient [GSF11]. In the field of virtual machine migration for example the energy [Str12], bandwidth [ASR+10], and memory consumption [WZ11] are relevant. Furthermore, response time [QZW+12], migration costs [SSSS10], and lost revenues [ZZSB13] should be considered.
Since the requirements of virtual machine migration and operator movement in cloud-based CEP systems differs, the existing approaches must be adopted.
For example, the resource consumption differs. In the context of operator movements in CEP systems normally only CPU, main memory, and network traffic are relevant, while the used disk space is negligible. This does not hold in the context of virtual machine migration, where the migration of the disk is the main problem. Also the energy consumption is not relevant in this work, because cloud providers only charge for the used/ reserved amount of CPU, memory, and disk capacity and not for the consumed energy.
Response time, migration cost, and lost revenues are examples for aspects, which hold for both application fields, operator movement in CEP systems and virtual machine migration.

Elastic CEP. Several different approaches exists to provide elasticity for CEP. Hereby, different levels of elasticity can be differentiate. Elasticity can be provided on operator, query, or engine level [Hei11]. By providing elasticity on operator level, instances of an operator process can be added or removed [SAG+09]. If this approach is transferred to the query level, whole queries or subqueries can be replicated and performed in parallel [GJPPMV10]. If statefull operators are processed in parallel, the different instances have to synchronize the global state of the operator leading to an communication overhead. This communication overhead is restricting the system parallelism. Stateless operators can be easily parallized, since no synchronization is needed. If elasticity is provided on engine level, new computing nodes can be added or removed during runtime to migrate existing or execute new operator processes [SHLD11]. A combination of the different levels of elasticity is possible. Nevertheless, no approach is known to estimate operator migration costs or to include operator migration costs of CEP systems into the query optimization.

Operator migration. Our work considers operator migration processes based on the pause-drain-resume-strategy [ZRH04]. There exists also other strategies to migrate operators. For example, new operator processes can be initiated, and executed in parallel to the old operator processes until all needed information (e.g., operator state) from the new and the old operator process match [WB10]. Although the stated considerations must be adapted if another migration strategy is used, the general problem that operator migrations create costs and influence the efficiency of the system still exists. Therefore, the operator migration costs should considered during optimization whatever migration strategy is used.

6 Conclusion

In this paper, we investigated the research problem of cost-aware optimization in cloud-based CEP-systems. We identified factors (e.g., resource consumption) that need to be considered in the optimization process. Based on a cloud-based CEP prototype, cost models are presented estimating the identified factors. To provide a proof of concept, three experiments were conducted based on workloads of a real application scenario of a stock exchange. The results of the experiment show that the monetary costs of the system can be decreased by up to 15 %, when migration costs are considered during optimization.
In future work, a comprehensive statistical analysis is needed to assess the presented cost models conclusively. Since operator migrations do not only influence the efficiency of the system but also other factors (e.g., average query latency), further examinations are needed studying the influence of a cost-aware optimization of the query processing.

References

[ASR+10] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore, and Andy Hopper. Predicting the Performance of Virtual Machine Migration. MASCOTS, pages 37–46. IEEE, 2010.

[BB05] Shivnath Babu and Pedro Bizarro. Adaptive Query Processing in the Looking Glass. CIDR, pages 238–249, 2005.

[BDG07] Alistair Barros, Gero Decker, and Alexander Grosskopf. Complex Events in Business Processes. In *BIS*, volume 4439 of *LNCS*, pages 29–40. Springer, 2007.

[GJPPMV10] V. Gulisano, R. Jimenez-Peris, M. Patino-Martinez, and P. Valduriez. StreamCloud: A Large Scale Data Streaming System. ICDCS, pages 126–137. IEEE, 2010.

[GSF11] Pablo Graubner, Matthias Schmidt, and Bernd Freisleben. Energy-Efficient Management of Virtual Machines in Eucalyptus. IEEE CLOUD, pages 243–250, 2011.

[Hei11] Thomas Heinze. Elastic Complex Event Processing. MDS, pages 4:1–4:6. ACM, 2011.

[JHJ+10] Gueyoung Jung, Matti A. Hiltunen, Kaustubh R. Joshi, Richard D. Schlichting, and Calton Pu. Mistral: Dynamically Managing Power, Performance, and Adaptation Cost in Cloud Infrastructures. ICDCS, pages 62–73. IEEE, 2010.

[KD98] Navin Kabra and David J. DeWitt. Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. *SIGMOD Rec.*, 27(2):106–117, 1998.

[Luc01] David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2001.

[Mey12] Patrick Meyer. Spezifizierung, Visualisierung und Bewertung von Eigenschaften monetärer Kosten für Complex Event Processing in der Cloud. Master thesis, Technical University Munich, 2012.

[MSHR02] Samuel Madden, Mehul Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously Adaptive Continuous Queries over Streams. SIGMOD, pages 49–60. ACM, 2002.

[QZW+12] Xiulei Qin, Wenbo Zhang, Wei Wang, Jun Wei, Xin Zhao, and Tao Huang. Optimizing Data Migration for Cloud-Based Key-Value Stores. CIKM, pages 2204–2208. ACM, 2012.

[Rö12] Lars Rödiger. Kosteneffizientes Cloud-basiertes Complex Event Processing. Diploma thesis, Technical University Dresden, 2012.

[SAG+09] S. Schneider, H. Andrade, B. Gedik, A. Biem, and Kun-Lung Wu. Elastic Scaling of Data Parallel Operators in Stream Processing. IPDPS, pages 1–12. IEEE, 2009.

[ScZ05] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 Requirements of Real-Time Stream Processing. *SIGMOD Rec.*, 34(4):42–47, 2005.

[SHLD11] B. Satzger, W. Hummer, P. Leitner, and S. Dustdar. Esc: Towards an Elastic Stream Computing Platform for the Cloud. IEEE CLOUD, pages 348–355, 2011.

[SSSS10] Upendra Sharma, Prashant Shenoy, Sambit Sahu, and Anees Shaikh. Kingfisher: A System for Elastic Cost-aware Provisioning in the Cloud. Technical Report UM-CS-2010-005, University of Massachusetts Amherst, 2010.

[Str12] Anja Strunk. Costs of Virtual Machine Live Migration: A Survey. IEEE SERVICES, pages 323–329, 2012.

[WB10] B. Wolf and I. Behrens. On-the-Fly Adaptation of Data Stream Queries. ISORC, pages 175–179. IEEE, 2010.

[WZ11] Yangyang Wu and Ming Zhao. Performance Modeling of Virtual Machine Live Migration. IEEE CLOUD, pages 492–499, 2011.

[ZRH04] Yali Zhu, Elke A. Rundensteiner, and George T. Heineman. Dynamic Plan Migration for Continuous Queries over Data Streams. SIGMOD, pages 431–442. ACM, 2004.

[ZZSB13] Mohamed Faten Zhani, Qi Zhang, Gwendal Simon, and Raouf Boutaba. Dynamic Migration-Aware Virtual Data Center Embedding for Clouds. IEEE IM, 2013.