ABSTRACT
The PhD thesis presented in this paper aims for the development of novel methods that enable the realization of highly scalable, highly reliable distributed Complex Event Processing systems that are capable of processing huge amounts of incoming events within a guaranteed time.

Categories and Subject Descriptors
C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed applications; C.4 [Performance of Systems]: Fault tolerance

General Terms
Performance, Reliability

1. INTRODUCTION
For modern IT systems, the ability for the timely reaction to the occurrence of real-world situations in the system environment has become a fundamental requirement. This applies to many different applications, e.g., in smart energy grids, automatic financial trading, logistics and production control. For example, in a smart grid scenario, the timely detection of a divergence between the energy consumption and the energy production can enable the deployment of an intelligent Demand Response [3] system, adapting the energy demand of intelligent appliances to the energy production, which reduces the demand for operating reserve provided in expensive supplemental power plants. In such applications, incoming data streams of low level information arrive from heterogeneous sources at high rates and need to be processed in real-time in order to detect more complex situations. Those streams can be bursty, so that their rates fluctuate tremendously.

To tackle such scenarios, the paradigm of Complex Event Processing (CEP) has emerged as an appropriate approach.

CEP systems enable the integration and analysis of primary event streams captured by distributed heterogeneous sources, e.g., sensors. In doing so, events from the primary streams get correlated to complex events corresponding to knowledge of a higher level, which is then provided in outgoing event streams to consumers. As event sources are often inherently distributed and available in a large number, it is of great importance to design the CEP system as a distributed system in order to minimize the network load and the detection latencies. In distributed CEP systems, operators hosted on different nodes in the network are taking a share in the overall event processing by correlating events from incoming event streams into events of higher information value and that way producing outgoing event streams that can again serve other operators as input.

Fig. 1 shows an example of a distributed CEP system applied to control a smart grid. The operators $\omega_1$ and $\omega_2$ aggregate events emitted by power consumers and producers respectively that signal changes in the rate of power consumption and production. $\omega_3$ then combines these aggregated rates with other information about smart appliances in the system (not shown in the graph), producing events that switch them on and off accordingly to keep the grid in a stable state. This is a Big Data scenario, as incoming event rates of hundreds of thousands of events per second can be reached at an operator. In that scenario, every household has a smart meter installed that reports changes in the en-
ergy consumption with a high frequency, as the system must be able to react to a changing grid load within a second.

As such a system would be responsible for the stability of the energy grid, the reliability of event streams, especially those emitted by $\omega_3$, is of great importance. No false-positive or false-negative events should be emitted, which would lead to an accidental switch-on or switch-off of appliances, possibly violating constraints of those appliances and bringing the grid into a critical state. Because of the enormous data rates of primary streams, reliability-preserving methods must be designed in a way that induces as little runtime overhead as possible. However, because of real-time requirements of the application which must be guaranteed even in the case of the failure of operators, the recovery time must be guaranteed.

Furthermore, operators need to be scalable, so that requirements regarding the timeliness of event detections will not be violated despite of an increasing load. To allow for this, the scale-out of operators through the parallelization of their execution is an important approach, as scaled-out systems enable an elastic response to increasing and decreasing demands of computational power. In doing so, the parallelization methods should be easily applicable to existing operators by providing an easy execution model that does not require an operator to be adapted to highly specialized parallelization-specific programming models.

2. PROJECT VISION

In my PhD project, I will develop novel methods for reliability and scalability of distributed CEP systems, so that they are able to process huge data rates in real time despite of operator failures. In doing so, I want to overcome shortcomings of existing solutions as they are described in Section 3. To achieve that, I will investigate how CEP operators work on event streams and develop appropriate execution models. Basing on such models, new algorithms for reliability and parallelization can be developed. The execution models should cover a wide range of real-world CEP operators and allow for their easy integration.

After developing novel basic approaches in these areas, I plan to design an optimized highly scalable, highly reliable CEP middleware that automatically adapts the parameters of the basic algorithms to optimize the system behavior according to the user’s requirements with regard to Quality of Services.

3. IMPACT BEYOND STATE OF THE ART

3.1 Reliability

3.1.1 Prior State of the Art

The following excerpt from [7] summarizes the prior state of the art with regard of the reliability of distributed CEP systems:

“The efficiency of reliable event processing can be measured with respect to its runtime overhead in a failure-free execution as well as its recovery overhead in the presence of failures. Currently, dealing with reliability leaves two basic options for event processing systems, known as replication and rollback-recovery. While active replication [9] minimizes the time to deal with host and communication failures, it imposes high processor utilization on the hosts at run-time since the execution of every operator needs to be replicated. Replication also raises significantly the message overhead since event streams targeted to an operator must also be streamed to all of its replicas. Passive replication [2] has slightly different properties, sacrificing recovery-time in order to avoid run-time overhead, but the general problems remain the same. Rollback-recovery [4], on the other hand, requires in its classical form to store checkpoints at regular times to persistent storage. This adds additional run-time overhead regarding bandwidth that is needed to transfer (incremental) state information, which is a burden especially for high bandwidth streams. Furthermore, to ensure the atomic capturing of (incremental) checkpoints, the processing of operators needs to be interrupted inducing event detection latency. Given that an operator state even for simple processing such as aggregation may comprise several Gigabytes [10], minimizing the state for performing a recovery is one of the important research questions in providing large-scale event processing systems. A promising way towards avoiding the need for persistent checkpoints is to recover the state of an operator by replaying logs of incoming event streams [5, 6] (known as “upstream backup”). Yet, the approach is very restrictive regarding operators for which a consistent state can be guaranteed after a recovery.” [7]

3.1.2 Recent and Future Work

In recent work published at this conference [7], we have “proposed a novel method for rollback-recovery that allows for recovery from multiple simultaneous operator failures, but eliminates the need for persistent checkpoints. Thereby, the operator state is preserved in savepoints at points in time when its execution solely depends on the state of incoming event streams which are reproducible by predecessor operators. We proposed an expressive event processing model to determine savepoints and algorithms for their coordination in a distributed operator network. Evaluations showed that very low overhead at failure-free execution in comparison to other approaches is achieved.” [7]

Reliability-preserving methods often deal with a trade-off between recovery time and run-time overhead. With the savepoint recovery approach, we have achieved a rollback-recovery method for multiple simultaneous operator failures...
that has low runtime overhead and thereby is suitable for Big Data scenarios. However, the recovery time can be high, if multiple adjacent operators fail and have to be recovered in a sequence. To deal with that problem, I plan to develop methods to allow for real-time guarantees of the recovery-time, so that operator failures will not violate the real-time requirements of a CEP application. These methods will manage the times when a savepoint is captured and the place where savepoints and event logs are preserved in a way that allows for such guarantees.

3.2 Parallelization

3.2.1 Prior State of the Art

To allow for scalability, two different approaches can be followed: scale-up and scale-out. In the scale-up approach, each operator is deployed on a single host that provides sufficient resources for the highest possible load. However, this causes unnecessary costs by idle resources in times of lower load. Moreover, additional redundant resources must be provided to allow for failure-tolerance. In the scale-out approach, an operator is run on multiple (potentially weaker) hosts in parallel. This can allow the system to react to changing load in an elastic way, comparable to virtual resources in Cloud Computing systems, and enables the usage of commodity hardware instead of high-performance computers. Hence, redundant resources needed to achieve reliability can be kept smaller, too. To allow for the scale-out of operators, methods for their parallelization are needed.

In this respect, there have already been proposed several approaches. However, they either base on sophisticated programming models that require a suitable operator implementation, e.g., in StreamMapReduce [1], or they follow a rather simplistic programming model that does not provide enough expressiveness for more sophisticated operators, e.g., in Apache S4 [8]. There is still a lack of an approach specialized to stateful event processing operators that allows for their easy integration.

3.2.2 Recent and Future Work

One approach that I plan to follow is to split the incoming streams into parts that get processed by multiple operator instances in parallel (see Figure 3). The results then get merged, so that the parallelized processing results in the same outgoing event streams as the sequential processing would do. In doing so, I aim to provide a split function that is easy to implement and suitable for a big class of CEP operators relevant for real-life applications. The work on some basic ideas for this approach is in progress at the moment. Parallelization brings new challenges with regard to reliability, which shall also be tackled in the approach.

Other approaches will also be considered. For instance, it might be possible to split the operator functionality into several independent steps that can be processed in parallel on duplicated event streams. However, such an approach is operator-dependent, i.e., must be adapted to the specific operators. To facilitate the definition of the steps, an appropriate execution model can be defined that serves as basis of a parallelization framework.

4. REFERENCES


