Moderated Resource Elasticity for Stream Processing Applications

Michael Borkowski✉, Christoph Hochreiner, and Stefan Schulte

Distributed Systems Group
TU Wien, Vienna, Austria
{m.borkowski, c.hochreiner, s.schulte}@infosys.tuwien.ac.at

Abstract. In stream processing, elasticity is often realized by adapting the system scale and topology according to the volume of input data. However, this volume is often fluctuating, with a high degree of noise, which can trigger a high amount of scaling operations. Since these scaling operations introduce additional overhead and cost, systems employing such approaches are at risk of spending a significant amount of time scaling up and down, nullifying the positive effects of scalability.
To overcome this, we propose an approach for moderating the scaling behavior of stream processing applications by reducing the number of scaling operations, while still providing quick responses to changes in input data volume. Contrary to existing approaches, instead of using linear smoothing techniques, we show how to employ non-linear filtering techniques from the field of signal processing to pre-process the raw volume measurements, mitigating superfluous scaling operations, and effectively reducing the number of such operations by up to 94%.

Keywords: Stream Processing, Elasticity, TVD, EKF

1 Introduction

A major aspect of modern stream processing systems is elasticity [11], a feature well-established in cloud computing [6]. In short, an elastic system is capable of scaling up during times of increased load, and scaling down during times of reduced load, instead of constantly over- or under-provisioning computational resources. This allows the system to adapt to new situations, reducing cost while maintaining Quality of Service (QoS) [10]. A system with less capacity than the volume is said to be under-provisioned, whereas on the other hand, a system with more capacity than is needed is called over-provisioned [14]. Scaling is not limited to cloud computing, but has also been applied in stream processing [12].

In order to make scaling decisions, certain properties of the system are observed. On the one hand, these properties may be intrinsic to the system, i.e., its CPU utilization [10], memory usage [5], network traffic [26], or its performance [3]. On the other hand, the observed properties may be extrinsic to the system, for instance, the amount of incoming data to be processed [25, 12], as observed in our work. Generally, every scaling operation requires resources

by itself, i.e., it incurs a delay, consumes energy without creating revenue, and leads to computational overhead [9, 18], and therefore additional cost. This is especially the case for scaling up, since additional operators on corresponding resources must be activated. Therefore, scaling operations should be kept at a minimum [5, 18].

In cloud computing, current approaches assume thresholds of utilization between which an operator must be [2]. In stream processing, this translates to the notion that an operator can only handle a certain amount of input data volume [13]. For any amount of data exceeding this volume, an additional operator is instantiated. However, using this threshold-based scaling in a simple way results in relatively frequent scaling operations, which causes an overhead of resource usage and cost, as discussed before [5, 18, 9]. In certain cases, this cost is necessary in order to benefit from the additional computing power made available by scaling up, avoiding under-provisioning, or saving power by scaling down, but on a large scale, excessively frequent scaling operations increase the risk of losing too much cost on the overhead of scaling.

We consider the volume of incoming data as a time series, and argue that both long-term trends in volumes, as well as short-term variances (spikes and valleys) are observable. The long-term trend, for instance, can be the development of input data depending on the time of day, time of year etc., while short-term spikes rather represent spontaneous and short-lived events, i.e., noise that we aim to ignore for scaling decisions.

Following this, we propose to improve classic threshold-based scaling by changing the way scaling mechanisms react to changes of the input volume. Instead of using the raw input value of the measured input volume, or using simple smoothing techniques, we employ advanced, non-linear noise reduction techniques from the field of signal processing. We apply these techniques to the raw input values, creating a filter. Using this approach, we aim to separate the actual data to be used for scaling (the long-term trend) from noise (the short-term variance), and focus on scaling only based on the long-term trend. The intuition is that this reduces the frequency of scaling decision while still being adaptive to the fluctuations in input data volume.

To this end, the remainder of this paper is structured as follows: In Section 2, we discuss work found in literature related to the topic of scaling in stream processing. In Section 3, we present in detail our approach of minimizing the number of scaling operations in stream processing systems, followed by a detailed description of our implementation in Section 4. We evaluate the approach and its implementation in Section 5. Finally, we conclude and give an overview of possible future work in Section 6.

2 Related Work

A fundamental assumption in our work is the claim that computational overhead caused by scaling, as explained in Section 1, causes significant cost. The general impact of overhead introduced by frequent scaling of cloud resources has been

studied by Corradi et al. [5] (in the context of overhead within cloud data centers) and by Mao et al. [18] (in the context of auto-scaling in cloud workflows) and the common result is that indeed, such overhead has significant impact and should be kept to a minimum. Other work in this field has been presented by Gong et al. [9], where the impact of scaling overhead is quantified by showing that the CPU consumption using shorter scaling intervals is up to four times as high, compared to longer intervals.

Scaling in stream processing systems has been thoroughly considered and surveyed in the literature [1, 12]. Abadi et al. [1] present the Borealis stream processing engine, along with a flexible and QoS-based optimization model. However, the scaling mechanisms presented do not take into account the volume of input data. No detailed information is given about whether any pre-processing of recorded data (e.g., denoising) is used. Hochreiner et al. [12] present a model for elastic stream processing, and discuss the methodologies, advantages and drawbacks of scaling within stream processing systems.

Mencagli et al. [19] use the Model-based Predictive Control (MPC) technique to create a trade-off between reconfiguration stability and amplitude. While the context (streaming application) is the same, and the aim (reduction of reconfiguration overhead) is similar to ours (reduction of the amount of scaling operations), the authors focus on the use of a distributed and cooperative approach, while we focus on the noise reduction in the input signal.

The usage of input data volume for scaling decisions has repeatedly been considered in literature [12, 25], as was using threshold-based systems to deduce concrete scaling decisions [4, 13]. All of those approaches, however, suffer from the same overhead problem as described before.

Some work has been done specifically to tackle this problem of overhead due to fluctuating input. A general recommendation seems to be the usage of low-pass filters [5], with a concrete instance of such a filter proposed by Shen et al. [23]. In this work, the authors employ a moving-average filter, similar to linear smoothing (LS). However, the authors do not use advanced non-linear approaches, like Total Variation Denoising (TVD) or Extended Kalman Filters (EKF).

Another example of linear filters is found in the work by Gong et al. [9], where scaling decisions are based on a Fast Fourier Transform (FFT) and pattern recognition. To avoid overhead, the authors use a delayed scaling mechanism, i.e., hysteresis. We argue that this is a rather basic approach in the context of signal filtering, and has the disadvantage of a fixed delay with which even the most extreme changes in input data volume are processed to scaling decisions. In contrast, the TVD approach presented in the work at hand reacts quickly to clear edges in the input data volume signal.

To the best of our knowledge, the only approach explicitly using a non-linear approach is presented by Khan et al. [16], where workload time series processing using clustering is proposed. Variations of workload patterns are predicted using hidden Markov models. Nevertheless, the authors do not take into account any normalization methods for processing the time series.

3 Approach

As stated in Section 1, the goal of our work is to minimize the amount of scaling operations performed, based on the volume of incoming data, using methods from the field of signal processing referred to as noise reduction or regularization.

We consider a stream processing system, which is receiving incoming data, e.g., from a message queue, processing it using an arbitrary amount of operators, and forwarding the resulting data as output. As stated in Section 1, we observe the volume of incoming data. This is done at the operators initially ingesting the data, either explicitly by measuring the incoming data, or by utilizing already-available data, for instance, statistics stemming from the incoming message queue.

The primary input for our approach is the time series of recorded measurements of input data volume. We denote a volume measurement at a time t as v_t. Figure 1 presents the intuition behind our approach. The dashed line represents the trend of the volume of input data of a stream processing system. However, due to temporally local variance and fluctuation, the measured amount varies, as denoted by the solid line. It is visible that while the recorded data generally follows the long-term trend, there is a substantial amount of noise overlapping the signal.

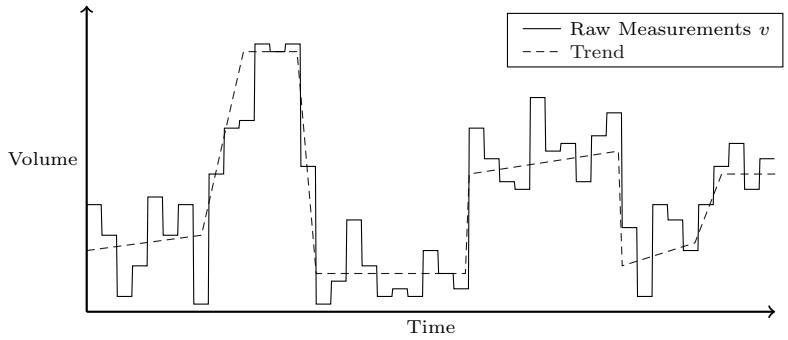

Fig. 1. Long-term volume trend (dashed) and actual, measured values (solid).

Naturally, if a stream processing system bases scaling decisions purely on the raw data, an excessive amount of scaling operations occurs [5, 18, 9]. In Figure 2, this is shown in the lower graph. Our approach applies filters to this process to reduce the number of scaling operations, i.e., reduce the number of steps in the *operators* line in Figure 2.

Therefore, we formally define our approach as follows. We regard a history of raw volume measurements, V, at various points in time t out of all measured times T, where v_t, as mentioned above, is the measured volume at time t:

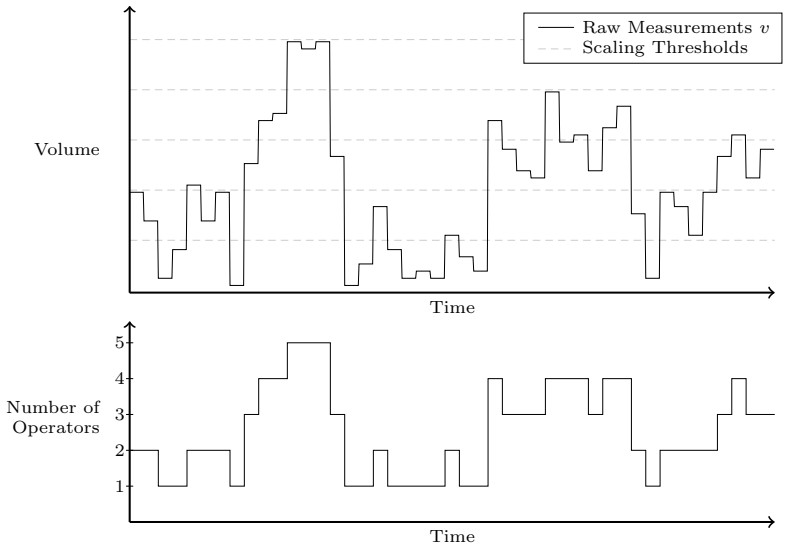

Fig. 2. Scaling of operator count according to thresholds of the actual volume, resulting in a high amount of scaling operations.

$$T = \{t_0, t_1, \ldots, t_n\} \tag{1}$$

$$V = \bigcup_{t \in T} v_t = \{v_{t_0}, v_{t_1}, \ldots, v_{t_n}\} \tag{2}$$

Based on the raw measurements $v \in V$, we define a filter f, which we apply to each value. The filter is applied at a given measurement time t and has access to all other measurement values in V, with the practical limitation that it can only access past measurements. We therefore define $f_V(t)$ as the filtered value for the time t, given all other values $v_i \in V$ where $i \leq t$. The concrete definition of f is not fixed, i.e., f is a parameter of our approach. Various concrete filters are described in the following section.

We then define the set of filtered measurements $\overline{V}$:

$$\forall v_t \in V : \bar{v}_t = f_V(t) \tag{3}$$

$$\overline{V} = \bigcup_{t \in T} \bar{v}_t = \{\bar{v}_{t_0}, \bar{v}_{t_1}, \ldots, \bar{v}_{t_n}\} \tag{4}$$

Figure 3 shows a possible resulting graph of the same volume measurement data, using a filter, along with the resulting scaling behavior of the system. When compared to Figure 2, it becomes clear that the amount of scaling operations has decreased. Note that this approach does not guarantee that the volume is met with correct scaling at each point in time. There is the possibility of under-provisioning for short periods in time, depending on the used filter.

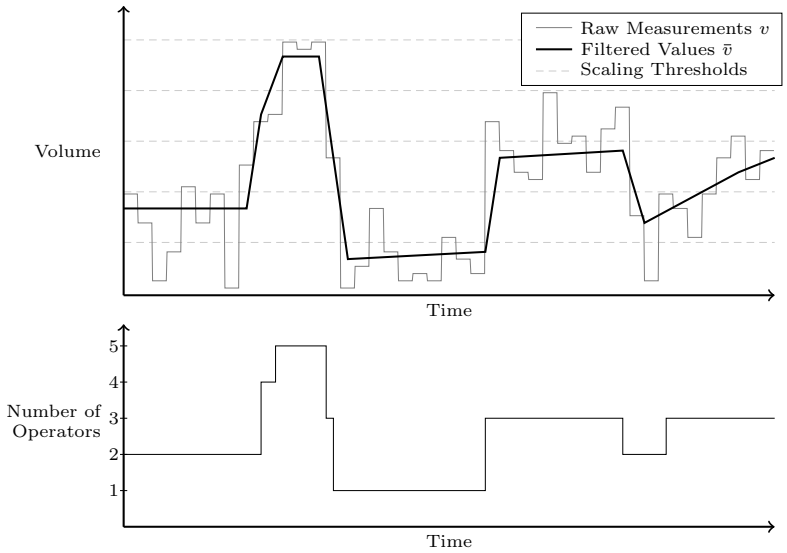

Fig. 3. The same scenario, with additional filtering of volume measurements. Instead of 23 scaling operations, the system only had to perform 7.

4 Implementation

We have implemented the approach described in Section 3 in different ways. The most important distinction between these approaches is the type of filter that is being used to reduce the noise in the signal, and to smoothen the time series of observed data volume used for scaling, i.e., the concrete function used for f. Stemming from the field of signal processing, a common approach of separating noise from signal is employing a low-pass filter [5]. We seek to improve the performance regarding detection of edges and separability in the Fourier domain [17] by proposing two non-linear filters: TVD [21] and EKF [15].

4.1 Linear Smoothing

Amongst the most basic methods in signal processing is linear smoothing (LS). Its essence is the smoothing of a noisy signal by setting each time series element to the arithmetic mean of its neighbors. In scenarios where live data is processed, only the past neighbors can be used, i.e., the window is set to *end* at the current element. Therefore, in its general variant, for a time series $v_0, v_1, \ldots, v_n$, and a given window width w, each filtered element $\bar{v}_x$ is set to the following:

$$f_V(t) = \bar{v}_t = \frac{1}{w} \sum_{x=t-w}^{t} v_x \tag{5}$$

Alternative versions include weights for more recent elements or exponential smoothing. However, since all of those methods essentially build a mean over a

window of past elements, we implemented LS as a baseline reference. A major flaw of all LS algorithms is the fact that they do not detect edges well. In the context of elasticity of stream processing, this means that changes in the volume are not detected immediately, and thus scaling operations are delayed by design.

4.2 Total Variation Denoising

A more advanced approach to smoothing is the approach originally proposed in [21], commonly called TVD [22], or ROF, after the authors' names [20]. The basic notion is that the total variation of a signal is to be minimized. Intuitively, TVD aims to remove the variation induced by noise while keeping the denoised signal as close to the original signal as possible, with respect to the least squares distance function. TVD is insensitive to the frequency ranges of noise and signal, making it more suitable to detecting sudden changes in near-real time, compared to linear methods like low-pass and high-pass filters or Fourier transforms.

Similarly to LS, TVD has one hyperparameter. In the case of TVD, this hyperparameter α determines the degree of smoothing. $\alpha = 0$ indicates no smoothing at all, i.e., the output of TVD is equal to its input, while $\alpha \to \infty$ means that more smoothing is performed, and this smoothing converges towards a steady state which is the denoised signal [21].

In its essence, the underlying TVD minimization problem proposed in the original work [21] is based on the assumption that the functional

$$v(x, y) = \bar{v}(x, y) + n(x, y) \tag{6}$$

expresses the raw signal v as a function of the actual (smooth) signal $\bar{v}$, and n, the additive noise[1]. Following this, the minimization problem is stated as a problem of minimizing the variation (i.e., the integral of changes in gradients):

$$\text{minimize} \int_{\Omega} (v_{xx} + v_{yy})^2 \tag{7}$$

where Ω is the variable domain, v_{xx} denotes the second derivative of v with respect to x. Two additional constraints provided in [22], binding the mean and variance of the raw and the reproduced signal to each other, are not shown here.

In our application of TVD, we have no multivariate functions, i.e., our v_0, v and n only depend on one (discrete) variable, which is the time t. Thus, we do not need to apply partial derivations. Since we record discrete, digital measurements, our definition of variation is also discretized and reduced, as shown in (8).

$$\text{minimize} \sum_{x=1}^{n} |v_x - v_{x-1}| \tag{8}$$

[1] Note that in the original work [21], the raw measured signal was named u_0, and the filtered signal was named u. We have adapted the names to v and $\bar{v}$, respectively, to maintain consistency within our work.

We have used this minimization problem, together with the original constraints, and applied the majorization-minimization algorithm described in [22], which majorizes the total variation minimization problem by its quadratic function, a methodology described in [7].

4.3 Extended Kalman Filter

The EKF is a nonlinear generalization of the Kalman filter [15]. Kalman type filters work by defining state transition and state observation models, and taking into account the noise and its (co-)variance. Again, since we do not have a multivariate function, we only have one variable, which simplifies the computation.

The EKF is based on the notion that there is a transition model F and an observation model G:

$$\frac{dx}{dt} = F(t)x + G(t)c(t) \tag{9}$$

$$z(t) = H(t)x(t) + n(t) \tag{10}$$

where $F(t)$ denotes the state transition, $G(t)$ is the control (input) transition, $c(t)$ is the control function, i.e., the input applied to a system, and x is the state. $H(t)$ is the observation model, i.e., the measurement transformation, $n(t)$ is the additive noise added to the signal, and z is the observed state[2].

In our application, we have simplified the model in that we do not apply any input to the system, but only observe it. Thus, the entire term $G(t)c(t)$ can be eliminated. As state x in the EKF notation, we have used the current volume (v in our notation), as well as the derivative (i.e., change in time, v') of the current volume. Therefore, in our application of EKF, $x = \left[\begin{smallmatrix} v \\ v' \end{smallmatrix}\right]$.

The term $z(t)$ from the EKF notation corresponds to the resulting, filtered volume measurement $\bar{v}$ in our notation. We have used this model in order to apply an unknown input to the estimation. In our case, the unknown input is the actual reason for the volume change, which is a factor we are not able to (generally) include in our model. We therefore allow the change in volume v' to be estimated by the EKF filter using only measurable data [8]. As a state transition, we use a matrix applying v' to v, i.e., we assume that without further input, the volume change will be constantly applied to the volume. The source of the change itself is, in this model, part of the noise, i.e., $n(t)$.

5 Evaluation

In order to evaluate our approach, we simulated a stream processing system using the three presented filters (LS, TVD, EKF) with varying input data volumes, and measured the resulting performance.

[2] Again, in the original approach [15], the control function is denoted as $u(t)$, and the noise is denoted as $v(t)$. We have changed the names to $c(t)$ and $n(t)$, respectively, in order to avoid overloading and maintain consistency.

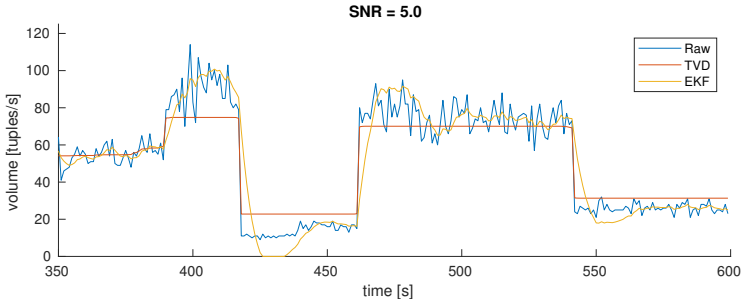

Fig. 4. Excerpt from a simulation with raw volume and filtered values.

To find commonly used and realistic values, we have surveyed literature, and decided to use values from [25]. Following this, we used volumes in the range of 200 to 500 tuples per second, and assumed a scaling threshold was 50 tuples per second. We introduced noise with varying signal-to-noise ratios (SNR). Since the *Rose criterion* states that an SNR of 5 is necessary to discern signal from noise with 100% confidence [24], we used various SNR values near that value for our evaluation (0.5, 0.8, 1.0, 5.0 and 10.0). All experiments were executed for a duration of 1,500 seconds. Volume measurements and filter applications were performed every second. An example is shown in Figure 4, where a few characteristics are visible. Most prominently, the piecewise constant nature of TVD can be seen. TVD also visibly misinterprets the mean of certain segments, since TVD depends on the entire history of the data, not only the values of the range shown, and those values influence its operation. For EKF, a certain momentum is visible, with which it reacts to changes in value.

As metrics, we have used the filtered values for scaling decisions, as described in Sections 3 and 4, and recorded (i) the number of scale-up and scale-down operations, denoted as s^+ and s^-, respectively, and (ii) the amount of time (in seconds) the system spent either over-provisioned, or under-provisioned, denoted as p^+ and p^-, respectively. The resulting metrics from the simulations are shown in Tables 1 and 2 (for SNR $= 0.5$ and 1.0), as well as Tables 3 and 4 (for SNR $= 5.0$ and 10.0). Note that in this work, we did not consider a cost model, but rather recorded the number of scaling operations performed throughout the simulation. For the work at hand, we consider each scaling activity as equally expensive, nevertheless we aim to refine the cost model in our future work.

The primary goal of reducing the frequency of scaling operations (s^+ and s^-) has been reached by both TVD and EKF, in high-noise environments even by over 93% (TVD) and 44% (EKF). However, the results clearly show that regarding scaling performance alone (p^+ and p^-), LS still outperforms EKF and TVD. This was expected, as LS has the tendency to scale without restriction (heavily impacting s^+ and s^-). Nevertheless, we argue that the advantages of reducing scaling frequency outweigh this drawback. For instance, in the case of SNR $= 1.0$, using TVD, a reduction of s^+ and s^- by around 90% causes an increase of p^+ and p^- of only around 8%, i.e., the positive impact in s^+ and s^- is still one order of magnitude higher than the negative impact in p^+ and p^-.

Filter	s^+	s^-	p^+	p^-
LS (Baseline)	208	209	**262**	**332**
TVD	**13**	**13**	312	335
	−195	−196	+50	+3
EKF	115	114	373	335
	−93	−95	+111	+3

Table 1. Results for SNR = 0.5. Best result per metric printed in **bold**.

Filter	s^+	s^-	p^+	p^-
LS (Baseline)	130	130	186	**220**
TVD	**8**	**7**	**154**	224
	−122	−123	+32	+4
EKF	79	78	319	224
	−51	−52	+133	+4

Table 2. Results for SNR = 1.0. Best result per metric printed in **bold**.

Filter	s^+	s^-	p^+	p^-
LS (Baseline)	33	32	**54**	**72**
TVD	**7**	**6**	115	86
	−26	−26	+61	+14
EKF	26	25	110	86
	−7	−7	+56	+14

Table 3. Results for SNR = 5.0. Best result per metric printed in **bold**.

Filter	s^+	s^-	p^+	p^-
LS (Baseline)	27	26	**44**	**64**
TVD	**7**	**6**	113	73
	−20	−20	+69	+9
EKF	23	22	100	73
	−4	−4	+56	+9

Table 4. Results for SNR = 10.0. Best result per metric printed in **bold**.

Considering the difference in performance between TVD and EKF, it becomes clear that TVD is a promising approach in high-noise situations, especially if SNR < 1.0. However, with increasing SNR, EKF starts to outperform TVD, especially in p^+ and p^-. We can observe this for SNR = 10.0. From a purely numeric point of view, this means that EKF is the most promising approach in low-noise situations. For s^+ and s^-, however, EKF, does not reach the performance of TVD, even in low-noise (high SNR) situations. However, looking in detail at the excerpt shown in Figure 4, we also argue that the performance of EKF can be further fine-tuned if the dynamics of the system, expressed in the matrices of EKF, are studied better.

6 Conclusion and Future Work

In this work, we have presented a novel approach of scaling in stream processing systems. Contrary to current state of the art, which uses simple linear filtering to process the volume of incoming data and applies this to make scaling decisions, our approach exploits advanced non-linear filtering methodologies from the field of signal processing to pre-process these volume measurements. This reduces the amount of scaling operations by 15% for low-noise scenarios, and over 94% for high-noise scenarios, while maintaining a comparable provisioning performance.

The two filters presented in detail, TVD and EKF, have been used to show the feasibility of this approach. We therefore propose further research in this area. For EKF, we argue that deeper understanding of the dynamics of volume changes in stream processing would allow for modeling of increasingly precise transformation matrices, further increasing its performance. Therefore, we plan to invest more research into different variations of the EKF parameters, possibly adding QoS metrics from the system itself as inputs for EKF. Furthermore, our next focus is to investigate in detail the computational complexity of our approach, and to use a cost model, similar to existing literature [19]. Finally, we want to evaluate the approach in more detail, and use a real-world data set for the simulation.

Acknowledgements This work is partially supported by the Commission of the European Union within the CREMA H2020-RIA project (Grant agreement no. 637066) and by TU Wien research funds.

References

[1] Daniel J Abadi et al. "The Design of the Borealis Stream Processing Engine." In: *CIDR*. Vol. 5. 2005. 2005, pp. 277–289.

[2] Anton Beloglazov and Rajkumar Buyya. "Energy efficient allocation of virtual machines in cloud data centers". In: *International Conference on Cluster, Cloud and Grid Computing*. IEEE. 2010, pp. 577–578.

[3] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo Calheiros. "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services". In: *Algorithms and architectures for parallel processing* (2010), pp. 13–31.

[4] Raul Castro Fernandez et al. "Integrating scale out and fault tolerance in stream processing using operator state management". In: *International Conference on Management of Data*. ACM. 2013, pp. 725–736.

[5] Antonio Corradi, Mario Fanelli, and Luca Foschini. "VM consolidation: A real case based on OpenStack Cloud". In: *Future Generation Computer Systems* 32 (2014), pp. 118–127.

[6] Schahram Dustdar et al. "Principles of elastic processes". In: *Internet Computing* 15.5 (2011), pp. 66–71.

[7] Mario A.T. Figueiredo et al. "On total variation denoising: A new majorization-minimization algorithm and an experimental comparison-with wavalet denoising". In: *International Conference on Image Processing*. IEEE. 2006, pp. 2633–2636.

[8] Esmaeil Ghahremani and Innocent Kamwa. "Dynamic state estimation in power system by applying the extended Kalman filter with unknown inputs to phasor measurements". In: *Transactions on Power Systems* 26.4 (2011), pp. 2556–2566.

[9] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. "Press: Predictive elastic resource scaling for cloud systems". In: *International Conference on Network and Service Management (CNSM)*. IEEE. 2010, pp. 9–16.

[10] Vincenzo Gulisano et al. "Streamcloud: An elastic and scalable data streaming system". In: *Transactions on Parallel and Distributed Systems* 23.12 (2012), pp. 2351–2365.

[11] Thomas Heinze et al. "Online Parameter Optimization for Elastic Data Stream Processing". In: *Symposium on Cloud Computing*. New York, NY, USA: ACM, 2015, pp. 276–287.

[12] Christoph Hochreiner et al. "Elastic Stream Processing for Distributed Environments". In: *Internet Computing* 19.6 (2015), pp. 54–59.

[13] Christoph Hochreiner et al. "Elastic Stream Processing for the Internet of Things". In: *International Conference on Cloud Computing (CLOUD)*. 2016, pp. 100–107.

[14] Sadeka Islam et al. “How a consumer can measure elasticity for cloud platforms”. In: *3rd International Conference on Performance Engineering*. ACM/SPEC. 2012, pp. 85–96.

[15] Rudolph E Kalman and Richard S Bucy. “New results in linear filtering and prediction theory”. In: *Journal of basic engineering* 83.3 (1961), pp. 95–108.

[16] Arijit Khan et al. “Workload characterization and prediction in the cloud: A multiple time series approach”. In: *Network Operations and Management Symposium (NOMS)*. IEEE. 2012, pp. 1287–1294.

[17] Max A Little and Nick S Jones. “Sparse Bayesian step-filtering for high-throughput analysis of molecular machine dynamics”. In: *International Conference on Acoustics Speech and Signal Processing (ICASSP)*. IEEE. 2010, pp. 4162–4165.

[18] Ming Mao and Marty Humphrey. “Auto-scaling to minimize cost and meet application deadlines in cloud workflows”. In: *International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2011, pp. 1–12.

[19] Gabriele Mencagli, Marco Vanneschi, and Emanuele Vespa. “A cooperative predictive control approach to improve the reconfiguration stability of adaptive distributed parallel applications”. In: *Transactions on Autonomous and Adaptive Systems (TAAS)* 9.1 (2014), p. 2.

[20] Charles A Micchelli, Lixin Shen, and Yuesheng Xu. “Proximity algorithms for image models: denoising”. In: *Inverse Problems* 27.4 (2011), pp. 45009–45038.

[21] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: Nonlinear Phenomena* 60.1-4 (1992), pp. 259–268.

[22] Ivan Selesnick. “Total variation denoising (an MM algorithm)”. In: *NYU Polytechnic School of Engineering Lecture Notes* (2012).

[23] Zhiming Shen et al. “Cloudscale: elastic resource scaling for multi-tenant cloud systems”. In: *2nd Symposium on Cloud Computing*. ACM. 2011, pp. 5–18.

[24] PJ Thomas and PA Midgley. “Image-spectroscopy–I. The advantages of increased spectral information for compositional EFTEM analysis”. In: *Ultramicroscopy* 88.3 (2001), pp. 179–186.

[25] Jielong Xu et al. “T-storm: traffic-aware online scheduling in storm”. In: *34th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2014, pp. 535–544.

[26] Qi Zhang, Lu Cheng, and Raouf Boutaba. “Cloud computing: state-of-the-art and research challenges”. In: *Journal of internet services and applications* 1.1 (2010), pp. 7–18.