Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master Thesis

Handling Interferences on a Multi-Operator CEP Node with Load Shedding

Timo Manuel Gutierrez, B.Eng.

Course of Study:	Software Engineering, M.Sc.
Examiner:	Prof. Dr. Kurt Rothermel
Supervisor:	Henriette Röger, M.Sc., Dr. Sukanya Bhowmik
Commenced:	June 8th, 2021
Completed:	December 8th, 2021

Abstract

The complex event processing (CEP) paradigm defines a set of systems that have the goal of finding important patterns in a streams of low-level events. They combine them in order to form information with a higher level of abstraction. These systems are susceptible to fluctuation in the workload of the event streams. Load peaks can lead to an increased latency which is undesirable in real-time scenarios. On limited hardware nodes this effect can be mitigated by using load shedding i.e. dropping a portion of the incoming events. The latency is also influenced by other operators sharing the same physical node. This thesis proposes a workflow which reduces processing time and therefore the latency by using this interference effect. For this purpose a neural network is trained. It is used for constructing a lookup table containing target system states leading to a specific reduction in processing time. These states are then translated into shedding configurations. While the implemented solution has its limitations, the result strongly indicates that the handling of the interference effect can be used to reduce processing times.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

CEP complex event processing. 3

CPU central processing unit. 23

DAG directed acyclic graph. 18

EPL event pattern language. 19

FSM finite state machine. 19

GAN generative adversarial network. 56

IoT Internet of Things. 15

MSE mean squared error. 26

QoR quality of results. 20

RSS residual sum of squares. 27

SGD stochastic gradient descent. 25

SPA stream processing application. 29

TCP transmission control protocol. 20

YAML YAML Ain't Markup Language. 22

1 Introduction

The digitalisation of everyday life leads to an enormous amount of data being generated. Across all different industries, from finance to manufacturing, applications gather data and seek to derive meaningful information from it. The most prominent examples are Internet of Things (IoT) applications. Sensor networks capture snapshots of their environment in the form of timely annotated events. This leads to a stream, composed of messages containing specific data. These notifications are not limited to sensor readings. For example they can also be produced by other software systems or by human input.

Said vast streams of messages created the need for a new way of processing data. Instead of analysing already persisted data, the events are handled as they come in an online manner. A specific class of these systems is represented by CEP applications. CEP is a scalable paradigm for detecting patterns in event streams. This means that those systems are capable of identifying complex situations. An example for such a situation could be a fire detection in an office building. Given data about a certain room, its temperature and the carbon monoxide value the system could warn about the hazardous conditions.

Another aspect of these types of systems is that they require near real-time response to patterns found. The chosen example makes this clear. The systems cope with this requirement by setting a fixed bound on the end-to-end latency of the application. This bound must not be exceeded. Due to a multitude of possible causes the workload of the event stream fluctuates. Unfortunately, peak workloads can lead to the CEP system being overloaded which directly has a negative impact on the latency. This effect can be avoided with a set of measures, for example parallelism. That requires additional resources. When working on machines with limited resources, which is usually the case in edge computing scenarios, the overload can be countered using a technique called load shedding. The idea is to discard a portion of the incoming events in order to mitigate the peak and thereby reducing the latency to meet the bound. When and how many events to discard is subject to the application and the magnitude of the overload.

An CEP application consist of a set of processes which are responsible for searching for matches of the patterns. When two or more reside at the same hardware, their executions influence eachother because they have to share the resources. This interference can have a negative influence on the latency as well, if one process is a bottleneck for latency because the neighbouring process is claiming too much of the computing power. This thesis tries to find a way of reliably countering this effect in order to reduce latency using load shedding.

The thesis is structured as follows:

Chapter 2 - Fundamentals and Background lays the foundation of the thesis and thereby gives an insight in the concepts and terminology needed for following its further course.

Chapter 3 - Related Work and Problem Statement sets the thesis in relation to similar work in order to highlight its boundaries. With a distinct picture of the objectives the chapter is concluded by formulating the problem statement.

Chapter 4 - Approach serves as a means of formalising how the solution of the problem stated is intended to be designed, implemented and evaluated.

Chapter 5 - Model Definition assesses the performance of different variables regarding their suitability for describing the interference effect and documents how the model is composed.

Chapter 6 - Handling Interference with Load Shedding consists of describing the implementation, designing an experiment and presenting the results.

Chapter 7 - Discussion concludes the results of the experiments and discusses perks and limitations of the implemented solution

Chapter 8 - Summary and Outlook summarises the thesis and gives an outlook on possible future work

2 Fundamentals and Background

To better understand the context of this thesis, this chapter introduces concepts, topics and techniques that are important for the further course of this work. It starts by explaining in detail what CEP is and in which areas it is applied (Section 2.1). Furthermore, the different aspects of the concept are introduced in order to get an insight into what is behind the terminology used in this thesis. The chapter then continues by defining a system model for the scenario of a multi-operator CEP node (Section 2.2). This model is used as reference for solving the problem stated by this thesis. The third and last section of this chapter is concerned with an existing solution for predicting processing times in a similar scenario (Section 2.3). It serves as a basis for solving the task at hand and is therefore crucial for understanding the background of this work.

2.1 Complex Event Processing

The idea behind the CEP domain stems from the need for a class of systems that enables the processing of events originating from the external world in a complex manner. This means in particular filtering and combining information from different sources in order to reason about a certain state of the systems periphery [CM12]. An example of that could be a sensor network for a large scale industrial machinery that wants to inform about safety hazards. The sensors continuously send events containing timely annotated properties of their environments like temperatures, positions of moving parts or distances to surrounding objects. A complex situation the system wants to detect might be a person walking into the trajectory of a robotic arm in order to stop the process in time or react by evading. More prominent examples are CEP systems that are used in areas like business process analytics, analysis of stock prices or intrusion detection in networks [CM12; EB09]. They all have in common that there is a need for processing data directly as it comes. There is no point in storing the data, as the application is concerned with the most recent state of the system [CM12]. The focus lies on the combination of various event types in a limited time frame. Furthermore, in contrast to similar paradigms the goal of CEP systems is not to aggregate these different types of data but to detect a pattern associated with a certain meaning [CM15].

In the following, the terminology and composition of such CEP systems is introduced and defined. Understanding the different components and terms is crucial for following the remainder of this thesis. In addition to that, the concept of load shedding is explained in more detail. When a part of a CEP system gets overloaded it may be the case that timely restriction on the processing of the events can no longer be ensured. For this case load shedding serves as a way of countering the negative effect on the latency by dropping events and thereby reducing load. To conclude the section, the CEP framework used in this thesis is described.

2.1.1 Complex Events

An event in CEP can be seen as a notification about a real-world property [CM12]. These specific events that contain such information are called primitive events. They are annotated with a timestamp and a type. The type indicates the content of the event, the so called payload. For example this could be the temperature or the carbon monoxide value for a fire detection scenario. In CEP, the primitive events are then combined in the specification of a pattern. This pattern represents a certain state of the environment the system wants to inform about. Once the pattern is detected within a time interval a new event is generated which represents the occurrence of said state. These events are then called complex events [Hed20]. A complex event is an instance of a pattern match at a specific time. It is an occurrence of the meaning represented by the pattern and can serve as a notification on its own. This leads to an event stream of the complex events, which then can be used for further processing in following patterns on an other abstraction level.

2.1.2 Operator Graph

At the core a CEP system consists of one or more operators. They are linked via input and output events forming a directed acyclic graph (DAG) [SBFR20]. The edges represent communication channels between the operators. While every operator has its own task, they are only logically separated. The whole graph may be distributed over heterogeneous infrastructure where it is also possible that several operators share one node. One operator has either one pattern or a set of patterns assigned to it [SBFR20]. Given the primitive events as an input, it is responsible for matching against the pattern, processing the match and produce an output event. This complex event is then forwarded to all its succeeding neighbours in the graph. Following the events through the DAG this leads to further abstraction with each operator.

2.1.3 Patterns and Matching

Two important parts in the CEP workflow are defining patterns and their matching. There are several aspects to how a complex event can be composed. A critical parameter is the time span in which the pattern match is searched. Based on the chosen window, different sets of events are considered. Another one is the composition of the pattern. It reflects on the different possible combinations of the source events. Furthermore the procedure used for matching is important because it has a direct effect on the performance of the system. In the following these concepts are explained closer.

Windows

Typically the arriving events are stored in a queue. Depending on the chosen window for processing, different partitions of said events are looked at during the matching process. Windows can be based on a time-span, event count or logically derived form the pattern. Also combinations are possible [Hed20; SBR19]. Based on the definition of a partition an event can be part of multiple windows. A window is defined by its length and the shift that is used to proceed to the next window. If the shift is smaller than the size, this means that the two spans are overlapping and therefore events might be in two or even more windows. For each window the events are assessed independently [SBR19]. This means that in each of them the event can contribute to a possible match.

Pattern Operators

For defining patterns, event specification languages are used. Sometimes also called event pattern language (EPL) [Hed20; RW10]. The languages differ at special types of operators but all of them support a set of core operators [Hed20]. Namely they are “sequence”, “conjunction”, “disjunction” and “negation”. Given a stream of events $\mathbb{S} = \{e_1, e_2, e_3, ..., e_n\}$ the event e_n denotes the n-th element in the stream. Furthermore e_1 occurred before e_2, meaning that there is a global order in the system. This can be achieved using the timestamps of the events and a tie breaker [SBR19]. Matching on a sequence then means searching for a set of events with specific types and order. For example the pattern $\{A; B\}$ on window $\mathbb{S}_{seq} = \{b_1, a_1, c_1, b_2\}$ would return $\{a_1, b_2\}$ as a match. A conjugation is a logical AND operator. Regardless of the order the pattern matches only for simultaneous occurrences. The pattern $\{A \wedge B\}$ on window $\mathbb{S}_{and} = \{b_1, a_1, c_1, b_2\}$ would return $\{b_1, a_1\}$. Similarly a disjunction works as an inclusive OR. Here it is sufficient that only one of the types specified is present. Given a window $\mathbb{S}_{or} = \{b_1, b_2, c_1, a_1, a_2, b_3\}$ and the pattern $\{A \vee B\}$ a possible set of matches could be $[\{b_1, a_1\}, \{b_2, a_2\}, \{b_3\}]$. The result depends on how the disjunction is evaluated over the window. In the example it takes the largest possible matches under the condition that every event fitting is included once. Another approach could be to stop the matching on the current window immediately after the first match, i.e. b_1 is found. The negation is always based on an entire window or a sequence. In the scenario of a window that would mean the entire window does not contain an event of the specified event type. When an event type is negated in a sequence like $\{A; \neg B; C\}$ it would be interpreted as an event of type A not followed by a event of type B until the next occurrence of an event with type C. [Hed20]

Matching with Finite State Machines

The task of matching a pattern can be seen as a finite state machine (FSM) [RW10]. Depending on the type of pattern a different FSM is constructed. The states define to which degree the pattern is fulfilled. Transitions for the state machine, denoted as δ, are derived from the relations in the pattern. For each event of the window δ is consulted and if it contains a transition for the event type the FSM moves to the next state. If the event type of the currently processed event is not part of the pattern there are two options. If it is for example a conjunction relation the event can be ignored, i.e.

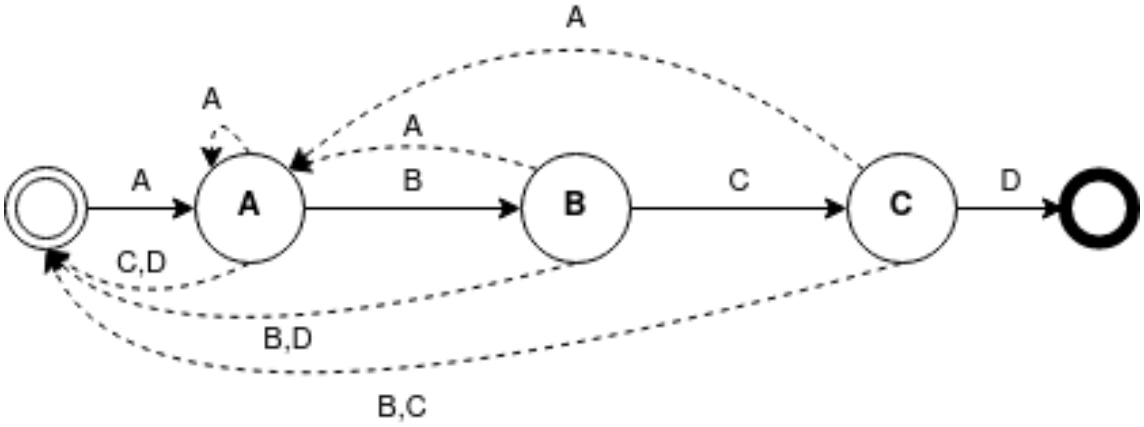

Figure 2.1: Example of a state machine for the sequence pattern A;B;C;D

a transition to the same state. In a sequence on the other hand the state machine would transition back to the initial state. An example of that is shown in Figure 2.1. Once a FSM terminates in the final state a pattern match is detected.

2.1.4 Load Shedding

An operator of a CEP system receives events from multiple sources. These sources emit their events at a certain rate. This rate may vary depending on numerous factors of the system. For example a sensor that detects motion on a factory entrance will have a higher emission rate at the change of shifts than in the time span in between. These peaks are not necessarily time-based or periodical. Another example could be how many machines of a shop floor are currently operated. Depending on the amount the emission rate may rise. This kind of bursty input data is common in CEP systems [HBN13]. Based on the magnitude of the peak, it is possible that the operator will overload, meaning that the hardware resources reach their limits. This results in an increased time needed for processing which leads to a lower throughput and a growing queue. Depending on the infrastructure this effect could be mitigated by replicating the operator on another node. Particularly in a cloud computing scenario this would be a sensible response, since in that case spawning additional resources is easily achieved [SBR19]. In other scenarios that might not be the case. If an operator solely has to rely on the hardware it currently runs on, a different solution has to be found.

Most CEP application have the need for a real-time processing of the data because the reaction to a detected complex event should follow as fast as possible. Since an overloaded operator increases the latency with which these matches are found, the load has to be reduced for ensuring a bound on said latency. This concept is called load shedding. The idea behind it is to drop a certain set of incoming events. In contrast to the congestion control of the transmission control protocol (TCP), there are no transmissions [TÇZ07]. A dropped event will not be recovered. This will ultimately have a negative effect on the quality of results (QoR) [SBFR20; TÇZ07]. Depending on the strategy used for shedding the events, different scenarios could arise. Dropping events from a sequence pattern can lead to important complex events not being detected. When a negation operator is used in the pattern a false positive could be detected [SBFR20].

In order to keep the negative impact of the load shedding on the QoR as small as possible, most shedding techniques do not just randomly drop events. A more sophisticated approach is to rank the incoming events according to a specified measure. It has the goal of assessing how useful the single event is for the operator and then the shedding begins to drop with the least valuable event. This measure is called utility of an event [SBFR20; SBR19; ZVW20]. The implementation of how it is composed may differ among different solutions. One aspect that can be considered in a utility function are domain specific characteristics of an event or a pattern. In a business process analysis it might be the case that a certain process violation requires immediate response due to regulations while another can be seen as less important. Such a scenario could be implemented by assigning weights $\mathbb{W} = \{w_1, w_2, w_3, ..., w_n\}$ to the pattern $\mathbb{Q} = \{q_1, q_2, q_3, ..., q_n\}$ of an operator [SBFR20; ZVW20]. In that case the utility can be assessed by the likelihood of an event ending up completing a pattern with a higher weight. Another approach would be to have a look on the internal state of the system. A means of assessing the utility can also be how likely an event is to complete and close an already opened state machine [SBR19].

The assessment of the utility allows for sorting the incoming events and with that brings an order to which events should be dropped first. How many events have to be shed and when to start with it are different questions. Therefore, the detection and determining of an overload is also part of the load shedding concept. This can be achieved by estimating, on event reception, if the bound on the latency will be exceeded. For the estimation the queuing and processing times of events, as well as the time needed for shedding can be considered. In a similar way one can approximate the amount of events that is needed to be dropped [SBFR20].

2.1.5 PRECEPT II Framework

The CEP framework used in this thesis is the "PRECEPT II Framework". It is part of the research project of the same name, that is conducted by the Institute for Parallel and Distributed Systems at the University of Stuttgart. Its goal is to propose different techniques for load shedding in CEP applications, that ensure a given bound on the latency. At a later stage of the project this is extended to optimise load shedding over the entire operator graph considering downstream effects and spatial differences of operators [Ins21]. In the following the different components of the framework are introduced. Furthermore, some implementation details which are particularly important to this thesis are highlighted.

Producers

Producers are the entities of the framework that create the primitive events. They are annotated with a timestamp, an id, a specific type and contain a payload, representing for example a sensor network which captures properties of its surroundings. For each producer an output rate can be specified which indicates the amount of events that are emitted per second. One producer process typically emits one single type of event. With these parameters the producers can be seen as a means of creating a specific load on the designed CEP application, which enables to create and evaluate different behaviours the system might show.

Operators

The operators are building the core of the framework. They receive the events either directly from a producer in form of a primitive event or from a complex event from another operator. An operator has a specific set of patterns assigned to it. The pattern operators currently implemented in the framework are "sequence" and "conjunction" (here called AND-Operator). Operators implement the matching for patterns with state machines. It keeps them as long as they are not terminated or for a specified time span. The resulting operator graph can be distributed over several nodes. A node is the physical device the operator runs on. This distinction allows for designing heterogeneous topologies with the framework. As a communication channel between the operators, Apache Kafka® is used. Each Operator has its own topic to which it is subscribed to. Sending an event to the operator then means to publish a message to its topic.

Sinks

Sinks form the end of an event stream through the operator graph. In the current implementation, the complex events are processed by printing them to the console. So a sink can be used for showing the output of the operator graph. In a broader sense that could be seen as the component where the reaction to a complex event is implemented.

Metrics Consumer

A special component is the “Metrics Consumer”. It is used for logging meta-data gathered by the other components of the system. This may include properties of the system like the length of the queue but also measured data about events. For example the start and end timestamps of the processing of an event. These values then can be used to calculate the processing time. This leads to an additional task the “Metrics Consumer” handles. The gathered logging data can also be further aggregated or combined in calculations. This component therefore plays a crucial role in the analysis of a systems behaviour. Gathered and further processed data can either be written to files or stored in a database.

Load Shedding

Because of its importance to this thesis it is further highlighted how the load shedding is implemented in the framework. The shedders reside at the operator component. They are configured via a dedicated control topic of the event streaming platform. Currently implemented is a form of probabilistic shedding. The amount of events that should be dropped is defined by a probability for each event type. It is configured which proportion of the incoming events should be processed. For a load reduction of 20% this would translate to the configured value of 0.8 for a certain event type. The 0.8 out of 1.0 indicates that for each event of said type that arrives at the operator the likelihood of being processed is 80%. In the other cases the event is discarded.

Scenarios

Lastly, the scenarios are the way the framework intents to design a system. They serve as a means of defining the topology. A scenario is a YAML Ain’t Markup Language (YAML) based configuration file. There the whole use case can be modeled. Along with a few parameters the configuration begins by listing all operators. They are defined by an unique operator id. Furthermore, a set of pattern is assigned to to the operator. The pattern itself is defined by a specific python implementation of a state machine. In the YAML file of the scenario this file is then referenced and in addition to that a destination for the output event is specified per pattern. The destination can either be another operator or a sink. In the second part of the configuration the nodes are defined. A node has an id as well in order to be able to distinctively run this part of the scenario on a dedicated machine. Each of the operators then can be assigned to one of the nodes.

2.2 System Model of a Multi-Operator Node

With the CEP framework a scenario is modelled which should reflect a real-world use case as closely as possible. Its goal is to build a multi-operator CEP node in order to induce interference between the processes running on it. This interference is based on the assumption that sharing one hardware, the distinct load on each of the operators also influences the performance of the other. Especially the divided utilisation time of the central processing unit (CPU) is expected to be causing interference. Therefore the scenario is modelled with the strict directive that no other process runs on the same node. The resulting system model of the design process is explained in the following.

Like depicted in Figure 2.2 the system is separated in two distinct hardware nodes. The central one, "node 2", serves as the multi-operator CEP node. On it reside the two operators of this CEP scenario. The node is in itself an isolated and limited hardware because any interference the processes might face should not be induced by other components of the system like for example the producers. On "node 1" on the other hand reside all other parts necessary for the execution. Namely, the already mentioned producers for the different event types, Apache Kafka®, the two sinks and the metrics consumer. The sinks are included for the sake of completeness and for indicating that the designed workflow is indeed functioning. The "metrics consumer" will play an important role, because the data gathered and calculated has to be extended for getting a clear insight into the influence of the interference. As communication channels two different technologies are used. Previous work has shown that using the Apache Kafka® for transmitting the primitive events from the producers to the operators results in an error of the specified emission rate [Gla21]. For countering this effect a light-weight messaging library with the name ZeroMQ is used. All other commutation channels are still provided by Apache Kafka®.

The patterns specified for the operator are fairly simple. As Figure 2.2 shows, there are three different event types that are emitted to the operators. The types are 1, 2 and 3. Both operators have exactly one pattern and it is the same for the two of them. It is a simple and evenly distributed conjunction of the three, i.e. $\{1, 2, 3\}$. For a match, one event of each type needs to arrive at operator regardless

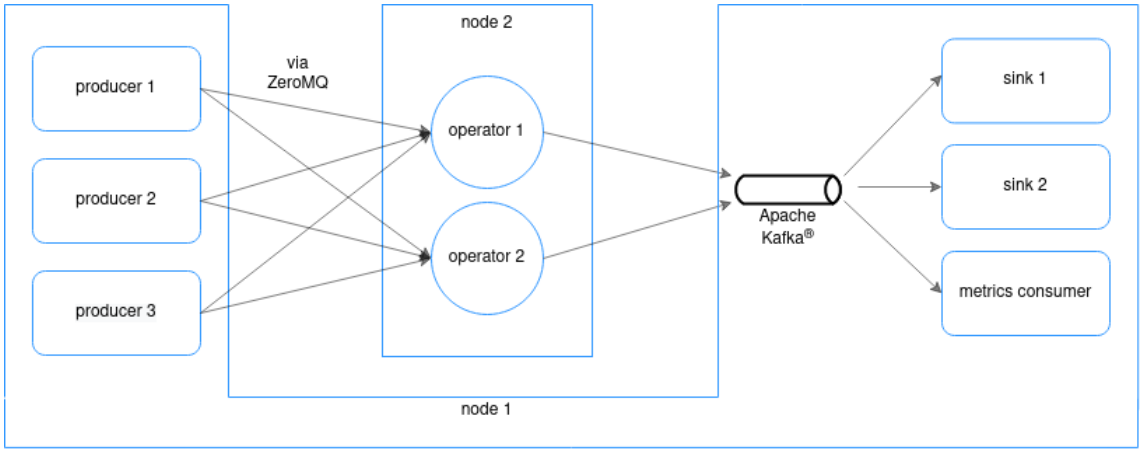

Figure 2.2: System model of a multi-operator CEP node

of the order. Once a match is found, “operator 1” will forward its complex output event to “sink 1”. Respectively, “operator 2” sends to “sink 2” after matching. Both of them gather data about the conditions and properties of the process and log them via the “metrics consumer”.

To conclude the system model, Table 2.1 shows the properties of the hardware used for “node 1”. This is stated in order to give a perspective for the scale of the scenario. Furthermore, it provides details about how limited the given hardware is. For the interference, the most interesting properties will be the number of cores and the clock rate. The hardware properties of “node 2” can be neglected for the problem of this thesis.

Property	Value
Name	Raspberry Pi 3 Model B+
RAM	1GB
Cores	4
Architecture	ARM
Clock rate	1.4 Ghz

Table 2.1: Hardware properties of node 1

2.3 Predicting Processing Times

Another important part of the background of this thesis is the prediction of processing times. In a scenario similar to the one described here, previous work has tried to find a way of predicting the processing time of “Operator 2” by building a regression model from variables of “Operator 1”. This solution has been formulated in the master thesis of Glaub with the title “Modeling Interferences of CEP Operators on Limited Resources” which was conducted with the Institute for Parallel and Distributed Systems at the University of Stuttgart [Gla21]. It can be seen as a precursor to this thesis.

Finding out which variables influence the interference and therefore the processing time of the second operator is also a crucial sub-task of this thesis and therefore the findings will serve as a starting point for the analysis. The work defined two metrics as promising indicators for the influence on the processing time. First, the arrival rate of events and second, a newly defined metric, the balance score. It tries to capture the balance of the arrival rates of the different event types with respect to the defined pattern. The work then proceeds with formulating its task as a regression problem. Based on the chosen metrics three models are fitted. A linear regression model, a polynomial regression model and a neural network. The results were compared using the coefficient of determination, also called “r-squared value” (r^2). They reached up to 0.8433 for the polynomial regression and 0.9622 for the neural network. Since the best r^2 value is 1.0, those are two good results which indicate that the chosen metrics are indeed a feasible way of inferring interference.

The remainder of this section is used to given a brief insight into two aspects of the work by Glaub that are particularly important for the further course of this thesis. Namely, the neural network and the coefficient of determination. This servers as a means of understanding the internal processes of fitting a neural network as part of a regression analysis and for clarifying the terminology.

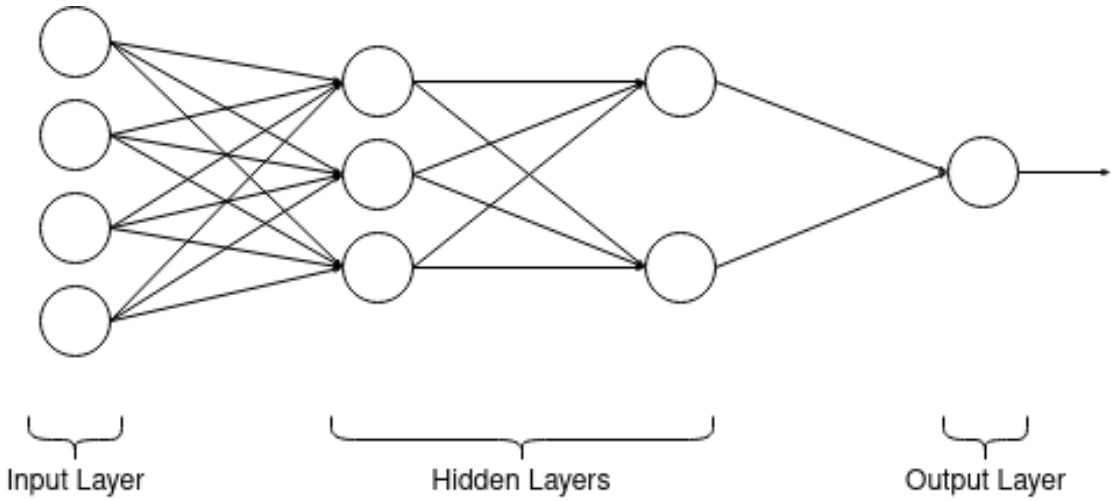

Figure 2.3: Layered architecture of a neural network

2.3.1 Neural Networks

A neural network can be a powerful tool for classification problems but also for regression analysis. It consists of so called neurons. Together, all neurons of a network form a directed graph like shown in Figure 2.3. The start of the graph form a group of neurons called “Input layer”. They each represent a feature of the specified model. This means that given an observation of the model each neuron of the input layer takes on the respective value. These values are forwarded to each neuron of the following layer, i.e. the first “Hidden Layer”. Within the hidden layers, each neuron calculates a weighted sum of its inputs and forwards the resulting value to all successors in the next layer. The neuron stores a specific weight for every input it has. This continues until the graph ends in the “Output Layer”. The last layer then is determining the result with a specified function, depending on the problem stated. After this short introduction of the process behind a neural network, two important calculations are explained. The activation function that each neuron utilises and the stochastic gradient descent (SGD) which is used for updating the weights of the neurons. [Nie15]

Activation Function

For the neurons of each layer an activation function is specified. Its purpose is to define if or to which degree the neuron is activated. For that it normalises the weighted sum of inputs at the operator because otherwise values from negative to positive infinity would theoretically be possible. The degree to which a neuron is activated can be seen as a measure of how much it contributes to the result given a certain input.

$$\sigma = \frac{1}{1 + e^{-x}} \tag{2.1}$$

In Equation (2.1) the sigmoid activation function is depicted. There is a whole set of different activation function but this brief introduction will only focus on sigmoid. It projects x, which is the weighted sum of the inputs, into a value between 0 and 1. The function is steepest in the middle

so that most inputs will be either closer to 0 or closer to 1 on the other end. This way the sigmoid activation function supplies clear results without losing out on smoothness. Figure 2.4 shows an example of a neuron with three input values. Given a set of weights for the neuron the weighted sum will be calculated as $x = \sum_{n=1}^{3} w_i x_i + b$ where b denotes the bias. The bias can be seen as an additional parameter to influence the results of the activation function in a certain direction. To this sum the sigmoid function will be applied. The resulting value is the activation of the neuron and is forwarded to the next layer. [Nie15]

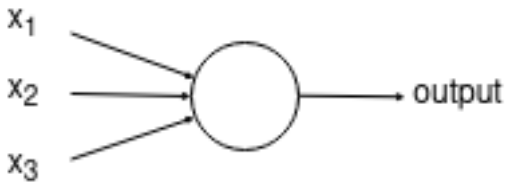

Figure 2.4: Example of a neuron with three incoming inputs

Stochastic Gradient Descent

Until now the process of how the result of a neural network is calculated given an observation was explained. The part missing is how it is trained. For fitting a neural network an annotated training dataset is needed. It consists of a couple of independent variables, also called features and assigned to them the respective dependent variable. The idea then is to run every observation through the neural network and calculate by how much the predicted value deviates from the actual value of the dependent variable in the dataset. This function is called cost-function. A common example for a cost function is the mean squared error (MSE). The calculation of which is shown in Equation (2.2).

$$C(w, b) = \frac{1}{2n} \sum_{x} ||y(x) - a||^2 \tag{2.2}$$

Given the weights denoted by w and biases b of all operators the cost is computed. In Equation (2.2) $y(x)$ is the result of the prediction by the neural network for an input x. Like mentioned in the explanation a then stands for the actual value of the dependent variable from the observation x. The goal while training the network is to iteratively get the cost function as close to zero as possible. This is achieved by updating the weights in each iteration. In Equation (2.3) the update function is displayed. The step size, also called learning rate is denoted by η. It is a parameter that controls the magnitude of the update. Furthermore, ∇C represents the partial derivatives of C given all weights and biases. This is called the gradient vector. Subtracting the gradient vector from the weights and biases moves their values closer to the optimal result, in this case the cost function closer to zero. That is the reason why the learning rate has to be chosen carefully. A value close to 1 might lead to a fluctuating update, since the gradient may have been too steep and surpassed the optimal value. This procedure of optimising parameters is called gradient descent. The stochastic part of it is using a probabilistic estimation of ∇C by only considering a subset. This is done due to high computational cost of calculating ∇C entirely. [Nie15]

$$(w', b') = (w, b) - \eta \nabla C \tag{2.3}$$

2.3.2 Coefficient of Determination (r^2)

The coefficient of determination, also called "r-squared", measures the goodness to fit of a regression [Jac18]. This can be interpreted as how well an independent variable or a set of independent variables explain a dependent variable. It will be used to assess the quality of a regression throughout this thesis. The calculation for it is shown in Equation (2.4).

$$r^2 = 1 - \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{2.4}$$

Starting at the numerator, e_i denotes the residual of the i-th observation which is computed with $e_i = y_i - f_i$, where f_i is the predicted value for the i-th observation while y_i is the actual value of its dependent variable. The entire numerator is therefore the residual sum of squares (RSS), which is a measure for the deviation of the model to the actual data. The denominator on the other hand is the total sum of squares, given by the sum of squared distances of y_i to the mean $\bar{y}$. This means that as a measure the r^2 value sets the residuals considering the prediction in a relation to the general variation in the data. Subtracting the quotient from 1 leads to 1 being the optimal value. On the least optimal side this could theoretically lead to an r^2 value up to negative infinity.

3 Related Work and Problem Statement

After getting an idea of the fundamentals and the overall context of this work through the previous chapter, here the perspective is set to closely related works. The area of CEP and load shedding in particular are subject to various interesting research questions. To get an idea of the periphery of the thesis, this chapter gives a brief insight into some selected works. The goal is to highlight similarities and distinctions which contributes to defining boundaries to the concerned domain. This aims to create a clear picture of the task at hand and will be the basis for subsequently formulating the problem statement.

Finding and applying techniques for handling certain properties of a system is a reoccurring problem. For example optimising latency over heterogeneous infrastructure in stream processing applications (SPAs) [RBR19]. The goal metric when it comes to assessing the timely performance of such a system under load is the entire end-to-end latency of the system. The idea behind it is to decentralise control models for the latency across different parts of the SPA. Instead of one model controlling the entire system, each part can utilise the model most fitting for its given conditions. The end-to-end latency consists of a multitude of different efforts like queuing, processing and the communication over various channels [RBR19]. This leads to a set of mechanics that can be applied in order to control the latency at the specific level. One of them is for example introducing parallelism, meaning to split the input stream of events, handle the split stream by replicated parallel instances of the operator and merging the results afterwards [RBR19]. For solving the overall problem, each part of the SPA gets a latency budget assigned as a part of the global bound on the end-to-end latency. The system then uses the mechanics which work best in the individual case to achieve said budget. In contrast to the system model of this thesis, here the measures taken are aiming at directly influencing the process they are targeting. Nonetheless, the goal of reducing latency and within that processing time respectively is a connecting feature. Therefore handling the interference for the sake of reducing processing time can be seen as another mechanic in the described set. The two problems are related but the focus of the introduced approach is to provide a framework to globally optimise the end-to-end latency while this thesis is concerned with influencing the processing time and therefore the latency on a single multi-operator node.

Another big focus of this work lies on load shedding. There, the trade-off is between gaining performance in form of lower latency and losing performance in terms of the QoR due to the loss of matches. In order to keep the quality as high as possible while still achieving the desired outcome, there are approaches that try to reason about which event to keep and which to drop by assigning them a certain utility measure. One example is to look at the utility of partial matches [SBFR20]. Instead of focusing on incoming events, the idea is to evaluate the partial matches i.e. the open state machines regarding their utility as soon as the system needs to reduce the load. Here, the utility of such a partial match is given by the probability that it is completed, the estimated time it takes until it is realised and the assigned priority of the concerned pattern [SBFR20]. The open state machines with the lowest value of this resulting metrics are dropped first. A different approach is to probabilistically shed incoming events based on their position within a certain processing

window of the operator [SBR19]. It defines a model based on previously detected positions of events that lead to a match, and with that builds a function for predicting the utility. As a third option the two described also can be combined [SBR20]. This results in a system that measures the utility of an incoming event in respect to the partial matches present in the processing window. In this case the dropping of an event means preventing it from being processed by a partial match [SBR20]. The three examples highlight the important considerations that come with using load shedding for ensuring a given bound on the latency. This task is similar to the one of this thesis. The most prominent distinction however, is that the main focus in this work lies on weighing out the performance loss of one operator for the sake of improving the other while the introduced approaches have the dilemma between latency and quality of a single operator.

Another task that has been solved was the prediction of processing times, by modelling interference of CEP operators on limited resources [Gla21]. The solution has been introduced in Section 2.3 on page 24. It leads directly to the formulation of the problem statement of this thesis. Taken the previous work as a baseline, the task at hand is to extend the model in a way that it enables the system to construct feasible shedding configurations, given a limited processing time. The overall goal, as it is with the related work mentioned above, is to contribute to the compliance with a specified bound on the latency. Since the interference between two operators sharing a processing unit only has direct affect on the processing time, it is used as a target metric. The reduction of it then has further implications on the end-to-end latency, like for example also reducing the length of the queue. Furthermore, the QoR also should be considered. This exclusively concerns the QoR of "Operator 1" because shedding is only applied at this entity, which leads to an adverse effect on the quality. Explicitly this means that out of a set of viable shedding configuration for achieving the limit on processing time, the implemented solution should pick the quality-optimal one. In addition to that, the developed workflow for achieving the task has to be integrated in the existing CEP framework.

4 Approach

On the basis of previous work, the goal of this thesis is to find a way of deriving shedding configurations for one operator which positively influence the performance of the other operator. Given a specific bound on the latency the system should be enabled to assess the system state and derive a shedding configuration that leads to the stated requirement in form of a decreased processing time. In the previous sections, the baseline for this work was described and in contrast to related work a problem statement was derived. Going forward, this section formalises the process followed in order to achieve a meaningful result.

The idea of this thesis is to build a model which reflects the interference between two operators sharing one processing unit. It should be capable of deriving a goal system state, given a specified bound on the processing time. Furthermore, this state has to be translated into a shedding configuration that leads to a desired outcome. One part of this endeavour is realised by the work introduced in Section 2.3. For the task of predicting processing times in this scenario, one also needs a representation of the underlying interference. The additional part of the problem concerned here, is to reverse the lookup. For predicting the processing time the input is an approximation of the system state, while the task at hand is to get a system state that leads to a processing time. To achieve the mentioned reverse lookup, this thesis proposes a straightforward approach. The model trained to predict processing times can be queried with a comprehensive set of defined system states with the goal of building a table consisting of the results. This can then be used to find the desired result pair given a specified bound. Proceeding from there, the implementation has to find a way to transform the approximation of the system state into shedding configurations. There it should be taken into consideration how the shedding will influence the QoR. The procedure for constructing the configuration should aim for optimising the quality within the feasible spectrum.

This outlined how the task of handling the interference with load shedding is planned to achieve. In addition to that, this process has to be integrated into the existing framework. The previous work was executed as a static analysis of the gathered data. In contrast to that this approach has the goal to provide a workflow that can be executed in an online manner. Most importantly this means that the different steps like building the model or setting a bound to the processing time can be triggered or executed via control messages. Furthermore, this also establishes the need for a central storage of the gathered data i.e. a database.

Figure 4.1 portrays how this is translated into the existing system model. As a storage for the output data a database is introduced. Namely, an InfluxDB. It is a time-series database that stores timely annotated datapoints. The metrics are calculated and stored by the “Metrics Consumer” on an event-level. For the creation of the model, a higher abstraction of the data is necessary. The chosen type of database enables exactly that. Instead of single entries the implementation can query for time windows and already apply aggregation functions to them. For the implementation itself a new component is added to the system model. The so called “Shedder Controller”. Its tasks are directly derived form the already described workflow. Building the model and once ready, constructing and

applying a shedding configuration for a given limit on processing time. As a communication channel to the operator the Apache Kafka® is used. The topic for that already exists in the framework. The "Shedder Controller" only has to bring the shedding configuration message into the expected format and the shedder will be updated at the operator.

After the workflow is successfully implemented it has to be evaluated. For that purpose the thesis proposes an experiment design. It will be executed under changing conditions and then analysed with a different point of view. Furthermore, the result has to be compared to an execution without the implemented workflow in order to be able to compare the actual impact the solution made.

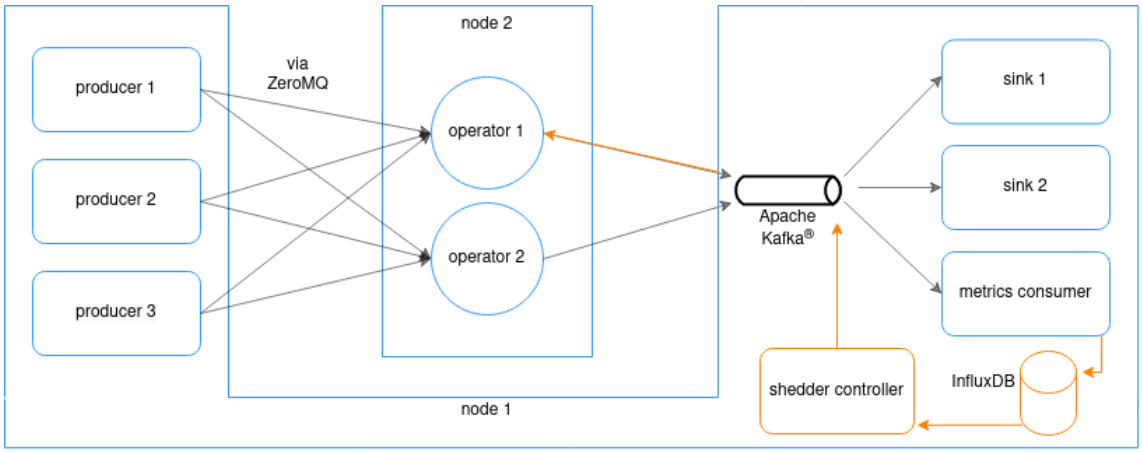

Figure 4.1: Enhanced system model of a multi-operator CEP node

5 Model Definition

For finding a solution to the problem of this thesis, a model has to be defined. It is needed for two tasks in particular. Training the neural network and constructing shedding configurations. For both of them the chosen variables should reflect the state of the system as closely as possible. Regarding the system state it is worth noting that the focus of the model lies on properties which are likely to be indicators for the influence of the interference. This means, while there are many metrics which give insight into a part of the system state, only some are of interest for this use case. For example the time an event spends in the queue of an operator is a property of the system state but it might not directly contribute to explaining the influence of interference of two processes. Therefore, this work focuses on metrics that have proven to be correlated to said phenomena. This is achieved by testing potential metrics on their impact. For each of them a set of producer configurations is created which is explicitly designed to induce a steadily increasing value of said metric at "Operator 1". The ones for "Operator 2" will remain the same. In that way, it can be shown how this property, when changed on one operator, influences the processing time of the other. In addition to reflecting the process of the interference, the model also has to be used for deriving shedding configuration. That means that another requirement for the chosen variables is that they can be controlled by shedding.

This chapter starts with describing the dependent variable, the processing time, in Section 5.1. Since this is the metric, the analysis and also the goal of this thesis is based on, it will be described in more detail. This includes showing how it is calculated and what time spans it includes and which not. The chapter proceeds with the introduction and analysis of three selected metrics. Namely, the arrival rate (Section 5.2), the balance score (Section 5.3) and the matches rate (Section 5.4). All of the three were promising candidates for the model and are therefore analysed regarding their impact on the interference. Based on that, it is argued and decided which of them will be used for the model.

5.1 Processing Time

The processing time is the metric that is used as an indicator for the performance of the system. This thesis focuses mostly on the timely aspect of the performance of a CEP system. While for example the QoR may be measured with a utility metric concerning the matches of an operator, this work concentrates on how fast the events are processed. For that purpose the processing time is chosen. At its core lies, like the name suggests, the processing of a single event. It is the time span the operator spends on said event, within which it has to utilise the shared resources. This is important because the sharing of a processing unit is what induces the interference. Therefore, a measure taken at "Operator 1" that influences the processing time at "Operator 2" can be seen as a way of handling the interference. This assumption is the basis of this work.

$$\text{p-time} = \text{stop\_processing} - \text{start\_processing} \tag{5.1}$$

In Equation (5.1), it can be seen how the processing time is computed. The equation itself seems trivial but the timeline shown in Figure 5.1 explains in more detail of what different parts the metric is composed. As an event arrives at the operator it is queued. If the operator happens to be not overloaded, the event finds the queue empty and is immediately processed. In other cases where a queue is built up it might have to spend some time there before the processing starts. This time is explicitly not part of the metric. At the start of the process, the open state machines are checked for a pattern match with the new event. If there is none the process had to check every open state machine, depending on the pattern it may have opened a new one and terminated the processing afterwards. Therefore, the times of the two following phases are zero and the processing time ends. In case there is a match, the execution at the operator proceeds to build an output event. At the timeline Figure 5.1 this part is labled “processing”. Afterwards the output event is sent to the operators successors in the graph and the measuring of the processing time stops. Those are the possible compositions of the metric. In theory it could vary depending on how much of the process is actually executed. For the scenario at hand this variation can be neglected. Here, the processing time is always measured at “Operator 2”, where the incoming event flow is steady and evenly distributed over the pattern. This means that on average every third event will be a match, ruling out scenarios where the metric is distorted by disproportionately stressing one part of the processing e.g. the matching.

Figure 5.1: Composition of the processing time

5.2 Arrival Rate

The first of the three independent variables discussed here is the arrival rate. It is seen as a viable candidate for influencing the interference. The rationale behind that is straightforward. A higher arrival rate induces a higher load on the operator and therefore the shared processing unit. To reiterate, all the metrics that are discussed in the remainder of this chapter are measured and adjusted at “Operator 1”. They are designed to create an effect on the processing time of “Operator 2”, only by changes to “Operator 1”. The tools at hand in this part of the work are the producers for the different event types. For each of them a specific output rate can be configured which directly translates to the arrival rate at the operator for the respective event type. From this can be derived that the arrival rate can be assessed at the level of each type. For the metric in mind, the perspective is set to be for the entire operator. This means that the arrival rate of an operator can be seen as the sum of arrival rates for the individual event types.

#	Type 1	Type 2	Type 3	Arrival Rate
1	7.5	7.5	7.5	22.5
2	10.5	10.5	10.5	31.5
3	13.5	13.5	13.5	40.5
4	16.5	16.5	16.5	49.5
5	19.5	19.5	19.5	58.5
6	22.5	22.5	22.5	67.5
7	25.5	25.5	25.5	76.5
8	28.5	28.5	28.5	85.5
9	31.5	31.5	31.5	94.5
10	34.5	34.5	34.5	103.5

Table 5.1: Producer configurations for the arrival rate analysis

Equation (5.2) shows how this is computed. The metric is assessed over an interval of events. Said interval is denoted as t or $t+1$ respectively. In the numerator the amount of arrived events is referenced. Therefore, $\text{events}_{t,t+1}$ can be read as the events that arrived between t and $t+1$. The denominator is the time passed between the two assessments in seconds.

$$\text{ar}_{t+1} = \frac{|\ \text{events}_{t,t+1}\ |}{\text{timestamp}_{t+1} - \text{timestamp}_t} \tag{5.2}$$

The arrival rate is therefore the incoming amount of events across all types per second. As described in the introduction to this chapter, the metrics are tested for their influence on the dependent variable. For that purpose the system is set under a specific load of events and the resulting metrics are measured at the two operators. In Table 5.1 the configurations for the three producers are listed together with the resulting overall arrival rate. Each of the configurations is run for 30 seconds and repeated 10 times, so that possible outliers can be detected and the result gets more reliable. It is starting with a fairly low event output rate of 7.5 per type which results in an expected total arrival rate of 22.5 events per second. From there the rates are steadily increased with each configuration. The intention behind that is to observe the change of processing time at “Operator 2” alongside.

The result can be seen in Figure 5.2. In order to depict the impact of the configurations two visualisations are chosen. The left visual is a series of box plots that are drawn for each setting from Table 5.1. Along the y-axis and therefore also within the plots the processing time is depicted. On the left side there is a scatter plot showing the processing time against the arrival rate. Each datapoint represents the average of all executions of one configuration. In addition to that a linear regression line is fitted and displayed together with its 95% confidence interval. The function in the upper left corner represents the regression line. Directly underneath, the r^2 value for the datapoints is shown.

Both of the visualisations show a strong upward trend with rising arrival rate. From the boxplots one can gain a deeper insight into the distribution of the data that is missed when only considering the averages. Here, in most of the configurations the data is comparably close together with configuration 4 being the most prominent exception. Since the values of the processing time are in a small range, a certain spread was to be expected. Nonetheless, the medians also increase steadily. The scatter plot together with the regression line emphasises the correlation as well. Together with an r^2 value

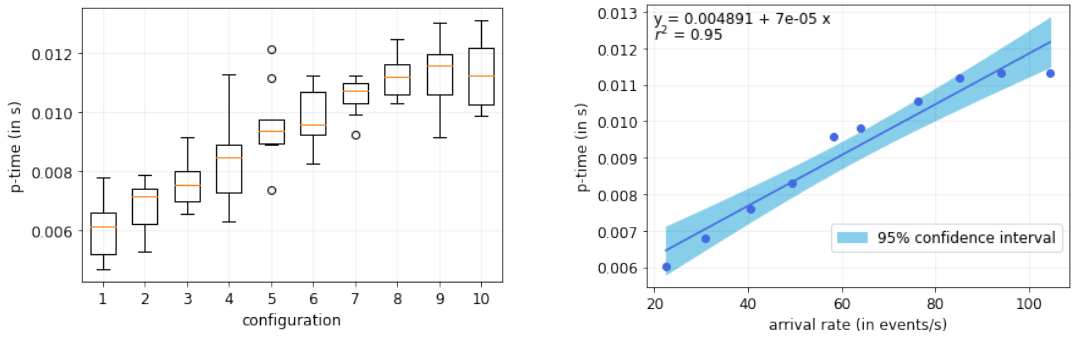

Figure 5.2: Scatter (right) and box (left) plots of the arrival rate under different configurations

of 0.95 this strongly indicates that the arrival rate is a viable candidate for the model. The results underline the intuitive assumption that increasing the load on "Operator 1" through emitting more events leads to an increased processing time on "Operator 2".

5.3 Balance Score

The idea of the balance score stems from the work introduced in Section 2.3. While for the assessment of the arrival rate as a metric in the previous section, the output rates of the three event types of the pattern were evenly distributed, which does not necessarily have to be the case in a real-world scenario. They can be imbalanced so that for every type the events arrive at a different rate. This directly influences the amount of matches that are realised and processed, depending on the pattern. Furthermore, it has an impact on how many open state machines the operator has to keep. The more imbalanced the events are with respect to the pattern, the higher is the amount of open state machines because the existing ones get closed at a lower rate than new ones get opened. From this behaviour the rationale behind using the balance score for controlling the interference is derived. If the arriving events are imbalanced and the number of open state machines increases, the time spent for trying to find a match increases alongside. As mentioned in Section 5.1, for a newly arriving event the operator searches through every existing state machine if there is no match to be found beforehand. The assumption here is that for this reason a higher amount of open state machines increases the load on the shared processing unit.

$$\mathrm{bs}_t = \max[|\mathrm{ar}_{a,t} - \mathrm{ar}_{b,t}| : \forall_{a,b} \in \{1,2,3\}] \tag{5.3}$$

Following that line of thought, the balance score is an approximation of the number of open state machines. Like depicted in Equation (5.3), here the arrival rates per event type are utilised. For the calculation of the balance score all absolute distances from each individual arrival rate to all remaining other is computed. Here, $\mathrm{ar}_{a,t}$ can be read as the arrival rate of event type a in the interval t. The balance score is then the maximum of the computed distances. This calculation is directly tailored for an evenly distributed pattern. In case of such a pattern, the maximal arrival rate is the one that sets the number of state machines per interval. In contrast to that, the minimum arrival rate says how many of them are closed, leading to the conclusion that the distance between the two rates, i.e. the maximum distance, indicates the amount of state machines that remain open in the interval.

#	Type 1	Type 2	Type 3	Arrival Rate	Balance Score
1	34.5	34.5	34.5	103.5	0
2	31.5	31.5	40.5	103.5	9
3	28.5	28.5	46.5	103.5	18
4	25.5	25.5	52.5	103.5	27
5	22.5	22.5	58.5	103.5	36
6	19.5	19.5	64.5	103.5	45
7	16.5	16.5	70.5	103.5	54
8	13.5	13.5	76.5	103.5	63
9	10.5	10.5	82.5	103.5	72
10	7.5	7.5	88.5	103.5	81

Table 5.2: Producer configurations for the balance score analysis

The producer configurations for the assessment of the balance score are shown in Table 5.2. For testing its influence on the interference the arrival rate is set as fixed. Only the proportions between the three event types differ. Starting with the first configuration, with a balance score of 0, the output rates remain evenly distributed. From there on, the rates of Type 1 and Type 2 decrease, while the rate of Type 3 increases by the sum of the reduction from the other two. This arrangement ensures a continuously increasing balance score among the configurations.

Looking at the result from the executions at Figure 5.3, it becomes apparent that the balance score does not perform as assumed. In the scatter plot of the averages over the configurations the datapoints are distributed seemingly at random. This is underlined by the linear regression line and its large 95% confidence interval. The r^2 value of 0.02 suggests that the balance score is not a fitting measurement for explaining the change in processing time on "Operator 2". Furthermore, the overall range of the processing times is significantly smaller than for the arrival rate. At the left side the box plots indicate a similar conclusion. The majority of the quartile boxes are overlapping across the configurations. In addition to that the data is also more spread than for the arrival rate. Even though the balance score was used for predicting processing time by the previous work, it does not seem to fit in this scenario.

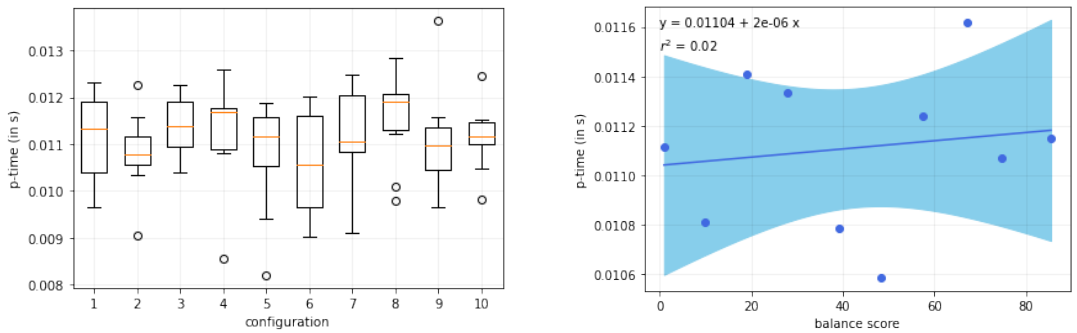

Figure 5.3: Scatter (right) and box (left) plots of the balance score under different configurations

As a result of the dissatisfying performance of the balance score, the cause of it was investigated in greater depth. The previous work, which found the balance score to be a valuable feature for the model, executed the experiments on a hardware with deviating properties of the one used in this case [Gla21]. This key difference suggests that it is the reason for the obviously varying result. The investigation narrowed it down to the time needed for sending an output event having a higher weight in the overall processing of an event than expected in relation to the time needed for matching. Where the idea behind the balance score was that the search through the open state machines creates a workload on the shared processing unit, in reality this effect was countered by the configurations with more matches creating and sending their output events. This lead to an alignment across the configurations. In short, the time needed for sending and the time needed for matching balanced eachother out. All of this indicates that on the hardware this procedure was executed, the sending operation is more expensive than on the other.

5.4 Matches Rate

The insights given by the investigation of the issues with the balance score lead to the idea of another metric. The matches rate. Since the sending operation seemed to be the costly factor, this metric aims to capture how often it has to be executed. This happens only if the currently processed events end up finalising a state machine, i.e. a full match on the pattern is found. Therefore, the amount of matches determines the frequency of which the sending operation is executed.

$$\mathrm{mr}_{t+1} = \frac{|\ \mathrm{matches}_{t,t+1}\ |}{\mathrm{timestamp}_{t+1} - \mathrm{timestamp}_{t}} \tag{5.4}$$

Equation (5.4) shows how the matches rate is computed. Similar to the arrival rate, this metric is the number of pattern matches at the operator per second. The computation is therefore the same as the one in Equation (5.2). Only the numerator is different. Here, $\mathrm{matches}_{t,t+1}$ denotes the pattern matches that occurred between t and $t+1$. As is the arrival rate, the matches rate is also directly linked to the arrival rates of each individual event type.

#	Type 1	Type 2	Type 3	Arrival Rate	Matches Rate
1	48	7.5	48	103.5	7.5
2	46.5	10.5	46.5	103.5	10.5
3	45	13.5	45	103.5	13.5
4	43.5	16.5	43.5	103.5	16.5
5	42	19.5	42	103.5	19.5
6	40.5	22.5	40.5	103.5	22.5
7	39	25.5	39	103.5	25.5
8	37.5	28.5	37.5	103.5	28.5
9	36	31.5	36	103.5	31.5
10	34.5	34.5	34.5	103.5	34.5

Table 5.3: Producer configurations for the matches rate analysis

For an evenly distributed pattern, like in the example of this thesis the matches rate can also be calculated as the minimum arrival rate of the three event types. This can be seen when looking at the producer configurations for this assessment in Table 5.3. As before the arrival rate is set to be the same across all the executions, in order to only measure the effect of the concerned metric. The matches rate starts at 7.5 at configuration one. It continuously increases through the increment of the "Type 2" output rate, while the other two rates are adjusted to meet the summed arrival rate.

The chosen set of configurations led to the result portrayed in Figure 5.4. Looking at the box plots on the left, one can see that the values are more spread than the ones from the arrival rate. Nonetheless, an upward trend is also visible. This is emphasised by the linear regression line on the scatter plot to the right. Even though there are visible outliers in both visualisations the r^2 value ends up at 0.73. That is worse than the one for the arrival rate but still a reasonably good result. It indicates that the matches rate can, to a certain degree, explain the influence of interference on the processing time of "Operator 2". This highlights not only its worth for achieving the handling of interference, but also underlining the assumptions made about the cost of the sending operations. Therefore, the matches rate will also be used in the model.

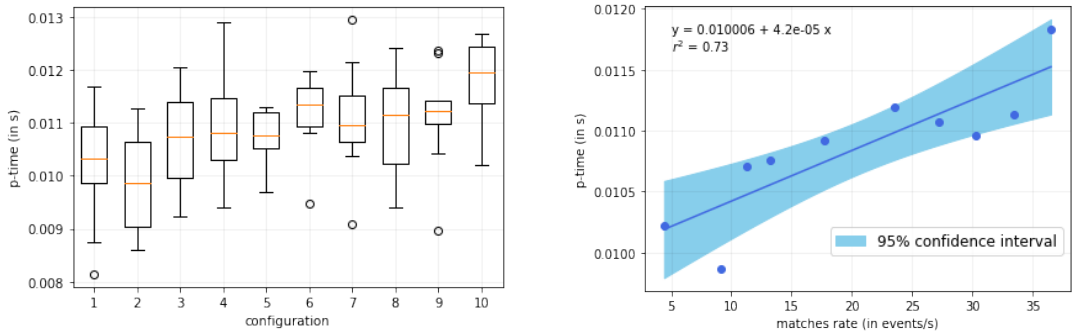

Figure 5.4: Scatter (right) and box (left) plots of the matches rate under different configurations

6 Handling Interference with Load Shedding

At the core of this thesis lies the intention to utilise the interference of two operators sharing a hardware for the sake of reducing processing time. The previous chapter introduced the model that is set to reflect the underlying process. In this chapter said model is applied to an implementation that aims to realise the handling of interference. The first section is concerned with how to derive shedding configurations from the model at hand. Section 2.3 introduced a way of using a neural network for predicting processing times given a set of input variables. The task here is to find a solution that enables the reversed lookup. Given a desired processing time (dependent variable), the implementation needs to be able to translate it into independent variables. In the case of this work those are arrival rate and matches rate. Furthermore, the implemented solution will not be able to simply set the two rates at "Operator 1". It will need to construct shedding configurations because shedding is the lever of the CEP framework in order to influence the operators. This implementation will be explained in Section 6.1. Once the solution is introduced, the next step is to define an experiment. Serving as a means of formalising how the designed process works end to end, Section 6.2 describes in detail the role each component of the system plays for achieving the desired goal. In addition to that, it also clarifies the way this work attempts to carry-out a coherent analysis of the influence the implemented approach has. The execution of the experiment allows for data being gathered and evaluated. This is done in Section 6.3, with the goal of assessing the applied treatments performance. Furthermore, it is described in detail and explained what insights the experiments gave into the process of handling interference through load shedding.

6.1 Generating Shedding Configurations

Like already introduced in Chapter 4 this thesis adds a new component to the system model. The so called "Shedder Controller". Within said component the solution for the problem at hand is implemented. It communicates with the other components of the system using the Apache Kafka® topics. Especially the existing control topic is utilised for sending new shedding configurations to "Operator 1". But before this is possible the controller first has to build a process for how to get from a desired processing time to said configuration. The data it needs for that is stored in time-series database InfluxDB. From the system model one can derive that the "Metrics Consumer" calculates the variables of the model defined in Chapter 5 and stores them in the database. After that they are available to the "Shedder Controller". The idea then is to use the data in order to train a model for predicting processing times in the same manner like described in Section 2.3. This model then can be used to construct a lookup table over the entire range of independent variables stored in the database. In the end this table has a set of arrival rate and matches rate pairs assigned to their respectively predicted processing time, so that this can be used to query for specific target processing times. Once a model is queried, it has to be translated into a shedding configuration. In the following these two tasks, the table construction and the shedding configuration construction, and their implementation are described and explained.

Parameter	Value
Loss Function	MSE
Activation Function	sigmoid
Optimizer	Adam
Hidden Layers	1
Hidden Neurons	64
Epochs	100
Batch Size	25

Table 6.1: Parameters of the trained neural network

6.1.1 Table Construction

The time-series database stores the data on event-level. Each entry is annotated with its respective timestamp, which determines the index in these kinds of databases. A query is therefore always executed over a specific time frame. Within this frame, the result is separated in windows which are aggregated with a function. For the purpose of getting data for training the neural network, the time frame is determined by the execution length of the experiment while the window length is set to 3 seconds. This aggregation window has shown to work best with the gathered data. As an aggregation function, the mean is chosen. The result set is then cleaned of all zero values (beginning of the data gathering) and separated into the different sets needed for training and testing the neural network. In Section 2.3 was described what was found to be a suitable way for predicting processing times. This will be the basis for the implementation of this work. Table 6.1 shows the parameters chosen for training the neural network. They are similar to what was proposed by the previous work but differ at some points. Most notably, the batch size was decreased to 25. This was due to the aggregation of the data prior to the learning. Aggregating the training data through the time-series query leads to a smaller sample size and therefore reducing the batch size balances this out. Setting this parameter is a direct result of choosing an aggregation window and also a size for the overall time frame. The other changes to the parameters were due to performance optimisation. After a neural network is fitted, the training set is used for evaluation. As a metric for how well the model performs, the r^2 value is used. Due to the randomness in the fitting of a neural network, the results may vary form execution to execution. In a scenario that aims for assessing the general quality of a resulting model, one would average the r^2 value over a multitude of iterations. This is not suitable in this case, since the best fitted model is needed for utilising it in the further process. Hence, the fitting is repeated as long as the quality is beneath a given threshold. In this scenario this threshold is set to 0.8. It is chosen because although theoretically the score can hit as high as 1.0 and the previous work reached a value of 0.96, the r^2 value for this particular model fluctuates depending on the gathered data. To ensure that the execution proceeds even with a slightly worse data basis, a r^2 value greater than 0.8 is seen as sufficient.

Once a neural network is fitted and tested to meet the requirements, the construction of the table begins. The idea is to get fine-grained series of processing times which are linked to a specific pair of arrival rate and matches rate that leads to it. In order to achieve this, combinations of the two independent variables are generated and handed to the neural network. Starting point is the

calculation of the minimum and maximum of each variable that was seen in the training set, like shown in Equation (6.1).

$$\begin{aligned} \mathrm{ar}_{max} &= \max[\ \mathrm{ar} \mid \forall \mathrm{ar} \in \mathrm{AR}_{training}] \\ \mathrm{ar}_{min} &= \min[\ \mathrm{ar} \mid \forall \mathrm{ar} \in \mathrm{AR}_{training}] \end{aligned} \tag{6.1}$$

Here, only the calculation steps for the arrival rate are displayed. The ones for the matches rate are conducted in the same manner. The maximum and minimum arrival rate, denoted as ar_{max} and ar_{min}, serve as boundaries for the construction. This stems from the assumption, that the training data is representative of all possible states of the system. Equally distributing the input configurations over this span, should lead to a table where each realistically achievable state is included. The distance of arrival rates between two successive configurations is defined as Δ_{ar}, like depicted in Equation (6.2).

$$\Delta_{ar} = \frac{\mathrm{ar}_{max} - \mathrm{ar}_{min}}{|\mathbf{T}|} \tag{6.2}$$

This measure is defined by the range given by ar_{max} and ar_{min} in relation to the desired size of the table $|\mathbf{T}|$. As Δ_{ar} gets smaller with $|\mathbf{T}|$ getting larger, it can be seen as a means of controlling the granularity of the table. In the case of this work $|\mathbf{T}|$ is set to 10000 as that is approximately the threshold where the granularity is no longer limited by the amount of table entries but the model itself. The entries are tuples that are constructed iteratively. Equation (6.3) shows the calculation of said tuples.

$$\begin{aligned} \mathbf{T}_i &= (\mathrm{ar}_{min} + i\Delta_{ar}\ ,\ \mathrm{mr}_{min} + i\Delta_{mr})\ ,\ \forall i \in \mathbb{N}_0\ ;\ i < |\mathbf{T}| \\ \mathbf{P}_i &= \mathrm{NN}(\mathbf{T}_i)\ ,\ \forall i \in \mathbb{N}_0\ ;\ i < |\mathbf{T}| \end{aligned} \tag{6.3}$$

One entry consists of an arrival rate and a matches rate value, that are calculated according to their respective span in the training data. Each entry of $\mathbf{T}$ is identified by its index i, which will be used to link it to its respective processing time in $\mathbf{P}$. In the calculation of $\mathbf{P}_i$, NN denotes the neural network and therefore $\mathrm{NN}(\mathbf{T}_i)$ represents the processing time predicted by the network given the input variables $\mathbf{T}_i$. As one can see the constructed table is two-parted. First, there is $\mathbf{T}$ which contains the possibly targeted system states. Secondly, $\mathbf{P}$ serves as an index for the querying process. The idea behind that is to utilise $\mathbf{P}$ in order to search the space for a desired processing time. Once found, the index directly links the result to the model in $\mathbf{T}$.

6.1.2 Shedding Configuration Construction

The second task is to use the constructed table for querying the desired system state and translate it into a shedding configuration. In the scenario of this work it will be sent to the shedder of "Operator 1". There, the events will be dropped according to the defined distribution. A shedding configuration in the framework consists of a probability for each event type in the pattern. This probability states which share of the events with the specified type should remain processed. So a shedding configuration of 0.75 for an event type will lead to 75% of load being processed and 25% will be dropped. This set of probabilities is put into a control message, containing one configuration per operator. Since "Operator 2" remains completely untouched in this scenario, it can be ignored.

The process starts with finding a way for getting the best fitting result, given a targeted processing time. This should be achieved by the procedure shown in Equation (6.4).

$$\begin{aligned} \mathbf{D}_i &= \mid \mathbf{P}_i - p' \mid ,\ \forall i \in \mathbb{N}_0 \ ;\ i < |\mathbf{T}| \\ i' &= \arg\min_i [\mathbf{D}_i] \end{aligned} \tag{6.4}$$

Here, the desired processing time specified by the execution is denoted as p'. At first the absolute distances from all index entries to the value of p' are calculated. As measure of how close the record is to the goal state, this is used to decide which index to take. The one with the shortest distance to p' is then declared as i'. This identifier is queried from $\mathbf{T}$. Therefore, $\mathbf{T}_{i'}$ will be the model a shedding configuration is constructed for.

Once the target system state is found the translation can begin. Given that the configuration is a distribution of shedding probabilities based on the current load, the process will need information about the recent state of the system. For that purpose the time-series database is utilised once again. This time the arrival rate will be handled at event type level, because the minimum of the individual rates will set the matches rate. The rest has to be adjusted accordingly. Algorithm 6.1 lays out the implementation in pseudo code. In line three and four the procedure starts by calculating the needed deduction in arrival rate and matches rate. For the first one the sum of the single arrival rates per event type is taken and then subtracted by the arrival rate of $\mathbf{T}_{i'}$. In contrast to that, the reduction in matches rate is derived from the minimum of the three. That is because the current minimal arrival rate will be used to set the targeted matches rate. This is possible because of the

Algorithm 6.1 Translation of the system state into a shedding configuration

1: **procedure** GENERATESHEDDINGCONFIGURATION($\mathbf{T}_{i'}$, ar_1, ar_2, ar_3)
2: result $\leftarrow \{\}$
3: $\Delta_{\mathrm{ar}'} \leftarrow \sum_{j=1}^{j\leq 3} \mathrm{ar}_j - \mathbf{T}_{i'}[0]$
4: $\Delta_{\mathrm{mr}'} \leftarrow \min[\mathrm{ar}_j \mid j \in 1,2,3] - \mathbf{T}_{i'}[1]$
5: **if** $\Delta_{\mathrm{ar}'} < 0$ or $\Delta_{\mathrm{mr}'} < 0$ **then** // not plausible -> not able to generate config
6: **return**
7: **end if**
8: $\mathrm{j}_{\min} \leftarrow \arg\min_j [\mathrm{ar}_j \mid j \in 1,2,3]$
9: result[$\mathrm{j}_{\min}$] $\leftarrow \frac{\mathrm{ar}_{j_{\min}} - \Delta_{\mathrm{mr}'}}{\mathrm{ar}_{j_{\min}}}$
10: $\Delta_{\mathrm{ar}'} \leftarrow \Delta_{\mathrm{ar}'} - \Delta_{\mathrm{mr}'}$
11: **if** $\sum_{j\in\{1,2,3\}\setminus\{j_{min}\}} [\mathrm{ar}_j] - 2(\mathrm{ar}_{j_{\min}} - \Delta_{\mathrm{mr}'}) \geq 0$ **then**
12:
13: **for** $j \in \{1,2,3\}\setminus\{j_{min}\}$ **do**
14: $\Delta \leftarrow \min[\mathrm{ar}_j - \mathrm{ar}_{j_{\min}} - \Delta_{\mathrm{mr}'},\ \Delta_{\mathrm{ar}'}]$
15: result[j] $\leftarrow \frac{\mathrm{ar}_j - \Delta}{\mathrm{ar}_j}$
16: $\Delta_{\mathrm{ar}'} \leftarrow \Delta_{\mathrm{ar}'} - \Delta$
17: **end for**
18: **else**
19: result $\leftarrow$REDUCEALLEQUALLY()
20: **end if**
21: **return** result
22: **end procedure**

evenly distributed pattern. Lines eight and nine show the calculation of said value. The minimal of three rates is reduced by $\Delta_{mr'}$ and the remaining proportion is the first one to be added to the result at its index $j_{\min}$. In line ten the needed reduction in overall arrival rate is subtracted by the already achieved deduction. Afterwards the remaining $\Delta_{ar'}$ has to be reduced from the other two event types. The ideal case is that it is possible to do that without overstepping the bound set by the matches rate. This is verified in line eleven. If that is the case the algorithm continues by reducing each of the two remaining arrival rates by the largest margin possible. Either this means by the distance to the current minimal arrival rate or the entirety of the remaining $\Delta_{ar'}$. In case the check in line eleven fails, the procedure is discontinued and instead all arrival rates are decreased evenly. This will result in a lower matches rate than targeted but the overall arrival rate can be fulfilled. In reality that exceptional behaviour has been rarely the case and is therefore included more for the sake of completeness than an actual impact on the system. Afterwards the result contains proportions that should remain for each event type. It is returned by the algorithm and can be sent to the operator via the control topic.

6.2 Experiment Design

In order to get a meaningful and reproducible result for this work, a process for the experiment has to be formalised. It has the goal to highlight the different steps undertaken to achieve the handling of interferences with load shedding. Furthermore, it becomes apparent how the different components of the system interact with each other during the execution. This is shown in the sequence diagram in Figure 6.1. The experiment is seperated into three phases. Learning-phase, pre-phase and post-phase. Each of the three is executed over a 10 minutes span. The first one is designed to collect the data necessary for the training of the model. After that the experiment enters a time span in which a certain system state is simulated, called the pre-phase. Its purpose is to create a reference for the evaluation of the treatment. In case of this experiment the treatment is the application of the shedding configuration, which happens as soon as the pre-phase terminates. After that, in the post-phase, the data is gathered again. The third part of the experiment leads directly to the evaluation where the results are compared.

As can be seen in Figure 6.1, time is not the only dimension the experiment is subdivided in. It also spans across the components of the system. Each of them has a specific role to play. The component that is labeled 'experiment execution' serves as a starting point. It is a script that orchestrates the execution. There, in the learning-phase, the producers are started. In order to get a diverse dataset for the training, the producers for the three event types change their output rate randomly in 30 second intervals. The rates vary in a range of 15 to 70 events per seconds. After the producers stop, a control event is sent to the 'shedder controller'. This event indicates that the table can be constructed. The successful completion is then indicated by another control event, signalling the 'experiment execution' to proceed to the pre-phase. At reception the producers are started again with a specific output rate, that is oriented at the maximum arrival rates during the learning-phase. Continuing for the specified 10 minutes the 'experiment execution' then decides on targeted reduction of the processing time and sends the inquiry to the 'shedder controller'. The shedding configuration is generated and updated at the shedder of "Operator 1". Immediately after the script sends the control event it restarts the producers with the same configuration like in the pre-phase, so that the shedding

can take effect. At the end the evaluation is performed by calculating metrics showing how the treatment impacted the system state. During the entirety of the three phases “Operator 2” remains untouched. It only receives its standard load of 25 events per second.

The experiment described represents one run through the process. There are two variables that strongly influence the impact the treatment has. Namely, the targeted processing time and the reference load chosen at the beginning of the pre-phase. In order to be able to quantify what influence these parameters have, the experiment is executed with different combinations of them. For the reference load, in the following also called baseline, the starting point is the maximum arrival rate and matches rate seen during the learning-phase. From there it steadily decreases in 5% steps. The targeted processing is handled similarly. It is derived from the maximum processing time and is decreased in 5% steps. To conclude, the resulting configurations for the experiment are the cross product of the two variables. For each baseline and target decrease pair, the experiment is executed once. In order to evaluate the influence of these parameters, the model should stay the same. This means that the learning phase is only executed one time and then skipped for successive executions.

In addition to evaluating the influence of different parameterisations on the implemented approach, the overall performance has to be assessed. This work tries to create a comparable reference by additionally executing the experiment with a naive approach. Instead of creating and using a model to find a targeted state, this implementation only reduces arrival and matches rate by the same percentage as the goal processing time is. This way allows an assessment of how good or bad the constructed model represents the underlying process of the interference.

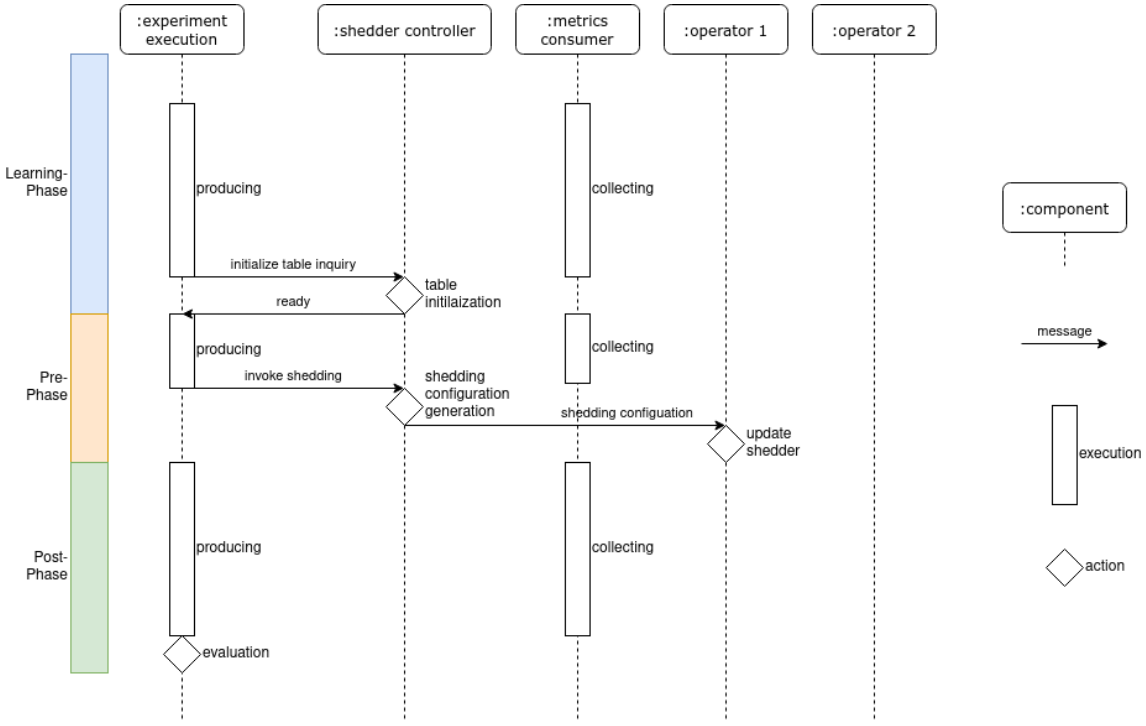

Figure 6.1: Sequence diagram of the experiment execution

Measure	Value
baseline	70%
target decrease	35%

Table 6.2: Configurations of the experiment execution chosen for the example

6.3 Evaluation

There are several different aspects of the experiment that are assessed in the evaluation. The first thing that comes to mind is having a look at the immediate impact that the treatment made on the system regarding the given goals. How did the shedding configuration translate to the different arrival rates on "Operator 1"? Did the shedding actually decrease the processing time of "Operator 2"? This serves as a first indicator on how the impact of handling interference looks like. After that, the assessment focuses on the how well the approach worked. When trying to achieve a specific reduction in processing time it is of utmost interest to know how close the result came to the defined target. Was the treatment able to meet it? Does it overshoot to a lower time or does it fall short with a higher processing time than expected? These results will then be compared to the naive approach in order to tell what difference utilising the table made. The third aspect of the evaluation is the resulted quality loss which will also be compared to the one of the naive approach. When using shedding for the sake of gaining faster processing times, losing out on pattern matches is the resulting trade off. Because of that it is important to assess how the treatment influences the matches rate. This can highlight at which cost the the system can improve processing times through handling interference.

For showing the general influence of the treatment, one example configuration is picked out. The one shown in Table 6.2 turned out to have the best result given the higher target decrease and is therefore chosen for highlighting the course of the experiment. To reiterate, the experiment with this configuration is executed with 70% of the maximum processing time during the learning-phase. The target decrease of 35% is used for querying the look-up table for the model that fits the desired processing time best, which is then translated into a shedding configuration.

Said values are shown in Table 6.3, alongside with their respective outcome in the post-phase. They are listed as averages over the corresponding phase. In addition to that, the table also contains the error from the target. For the two independent variables of the model, arrival rate and matches rate, the error is a measure for how precise the shedding configuration construction works and how well the shedder converts the configuration. With -3.17% for the arrival rate and -8.09% both fall short of the target that was aimed at. Nonetheless, the values get close to the targeted ones with several

Variable	Pre-Phase	Target	Post-Phase	Error
arrival rate*	165.3767	58.6251	60.4809	-3.17%
matches rate*	48.4095	17.9326	19.3835	-8.09%
p-time**	0.01550	0.0100	0.00999	0.78%

* in $events/s$, ** in s

Table 6.3: Results of the example configuration in the post-phase

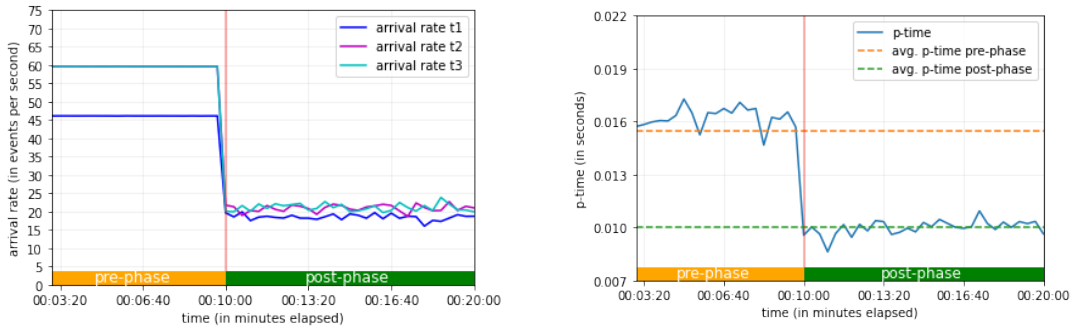

Figure 6.2: Line plots of the arrival rates (left) and processing time (right) during pre- and post-phase

smaller deviations like it can be seen in Figure 6.2. The right graph starts in the pre-phase where the arrival rates of the three event types on "Operator 1" stay continuously the same until the switch of phases.

This switch is depicted by the red vertical line. In the post-phase, when the shedding is invoked, the arrival rates drop drastically. This is a result of the configuration, which was in this particular case 40% for Type 1 events and 35% for both Type 2 and Type 3 events. The percentage within the shedding configuration state which proportion of events to keep. The arrival rates are averaged over 20 second windows, in order to smooth the visualisation. Nonetheless the values fluctuate around their actual target. These deviations are most likely due to the fairly small arrival rates. The shedder decides randomly, with respect to distribution set by the shedding configuration, which event to drop or to keep. Given that there were higher arrival rates, the average after shedding would also catch up with the target. The error of these first two variables has to be looked at isolated, because it describes the performance of the mechanic implemented and does not directly relate to the task of handling interference. The values for the processing time on the other hand do. For this set of configurations the implemented process worked almost perfectly. The achieved post-phase processing time deviates from the targeted one by only 0.78% which is in the area of microseconds. As depicted in the left graph of Figure 6.2 the processing time of "Operator 2" also fell sharply right after the switch of phases. The overall setup for the graph is the same as for the right one. Most notably the processing time is also averaged over 20 second windows. In addition to that, the two horizontal lines (dashed) indicate the overall average processing time in their respective phase. While the processing time also fluctuates, the data shows that the chosen approach is able to derive a model and its resulting shedding configuration, that lead to a specifically targeted value. This emphasises that handling interference can be a valuable contribution to improving performance on a multi-operator node.

That being said, the error under which this approach operates strongly varies across the different configurations of baselines and target decreases. Highlighted in Figure 6.3, there are stark contrasts between the experiments. The left of the two heat maps depicts the errors across all execution realised with the lookup table. On the y-axis are the baselines from which the reduction should be made. They start at 100%, meaning the maximum value during the learning-phase, and go down to 55%. This limit is chosen because, as visible in the heat map, the values start to almost exclusively fall short of the excepted processing time. Furthermore, at this threshold the result becomes increasingly meaningless due to the fact that the targeted time values were not seen during

the learning-phase. The shown corridor is therefore representative of the chosen setup of hardware and use case. On the x-axis one can see the targeted decrease in processing time. Starting from 5% and going up to 45%. The rationale behind this limit is the same as for the baselines. The first thing notable is that up to a decrease of 35% the overwhelming majority of experiments gets close to their target or overshoots. Even though achieving a lower processing time than expected is a set back in terms of precisely reaching a goal condition, it can be seen as better outcome than falling short. A lower result means that, in a real world example, the system would still have reached its goal when it comes to reducing processing time below a specific threshold. In addition to that, the error indicates that it is within the physical constraints of the system to reach the target set. Particularly the span between 15% and 20% target decrease across most different baselines seems to fall under that category. In contrast to that, the values between 40% and 45% target decrease are on the other side of the spectrum. They do not meet the goal and end up with a higher processing time, with it reaching its maximum with the 55% baseline and 45% target decrease at −35%. This is due to the already mentioned physical constraints, which is also highlighted by a triangular tendency towards the left bottom corner. At this area the configurations tend to come closer to or even undercut the minimum processing time seen during the learning-phase. Lastly, the area between 5% and 10% plus the span from 25% to 35% in target decrease came close to meeting their respective goals. There, the chosen example configuration is situated. Most of the errors are in the field of one digit percentages. The pattern seen across the heat map is not entirely consistent. From the insight the experiments gave to this point, one would expect the triangular tendency noticeable in the bottom right corner to go right through the middle of the heat map. But it does not entirely. The negative errors in the bottom right corner make a start in this direction but the development is interrupted in the lower part of the two columns of 15% and 20% reduction. One explanation for this could be that the data generated in the learning-phase through randomly generated arrival rate combinations and the resulting model do not provide accurate representations for these processing time spans. Overall, this result shows that even though the reduction of processing time through handling interference is practically possible with the chosen approach, the error of how precise the outcome is achieved varies.

As a means of controlling the effect of the implementation, the experiments were also conducted with a naive approach. The result of which is displayed in the right heat map of Figure 6.3. With lower reduction the 5% column contains low errors. Here the naive approach performs very accurate. But with increasingly larger target decrease the errors climb rapidly. Furthermore, it only develops in the negative direction, meaning that the processing times achieved are longer than the targeted ones. Therefore the attempt to handle interference with a naive approach consistently falls short for

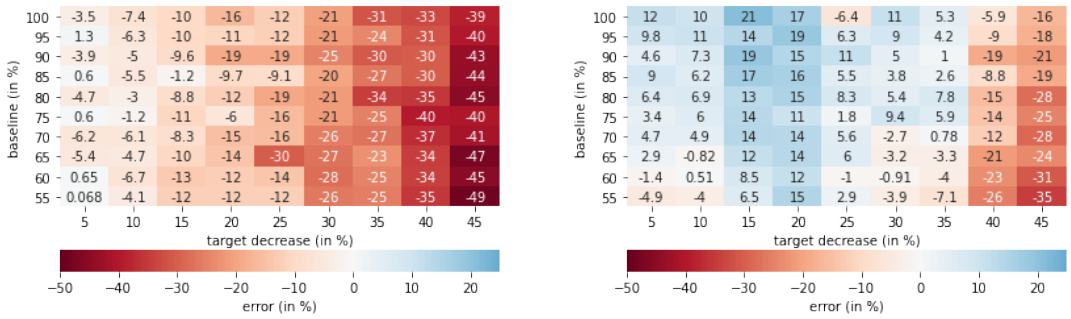

Figure 6.3: Heat maps showing the error from the targeted processing time

reductions higher than 5%. This indicates that the implemented approach of this work has a real impact on how the processing time develops. Comparing the two heat maps side by side, it becomes apparent that the right one generally consists of lower values which do not consistently increase or decrease along one axis. Together with the fact that most of the targeted processing times can be achieved, this suggests that the implemented lookup table is an approximation for the underlying process of the interference.

To conclude the evaluation, the last aspect to look at is the loss of matches induced by the shedding. In real-world applications this is a significant downside and therefore can be seen as the price to pay for exploiting the interference of two operators sharing resources. Figure 6.4 shows an entirely different perspective of the experiments. It contains two scatter plots. One for the results of the naive approach (right) and one for the lookup table approach (left). The y-axis depicts the decrease of the matches rate in percent. This is plotted against the reduction of processing time in percent. For that visualisation the targeted time does not play a role. It is solely concerned with how many matches are lost for a realised reduction of processing time. Looking at the right graph, the most prominent feature is that the data points end after 45% reduction of the matches rate and 25% decrease of processing time. For the y-axis value this is not surprising because by design the naive approach did only reduce by the percentage of the given target. The x-axis values on the other hand had no such limitation. This means that for the naive approach it was simply not possible to reach a higher reduction in processing time, emphasising that the interference follows a more complex process. Furthermore, the data points, while having an upward drift, are agglomerated. For most decreases in processing there are several matches rate reductions that lead to said outcome. A similar agglomeration can be found in the left plot. But there it is only in the upper left corner which is due to the configurations that aimed for a target decrease of 40% and upward. Like shown in Figure 6.3 those experiments fell short of their target and now fall into the area of 25% - 40% processing time reduction in this graph. Besides of these values, the decrease in matches rate needed appears to be strongly correlated. The plot also suggests that due to the matches rate decrease already reaching 80%, there is a hard limit on to which percentage the processing time can be decreased. Intuitively this is obvious because the whole approach only sheds from “Operator 1” while “Operator 2” keeps its load, but this visualisation is able to show it. In case of this approach that limit seems to be around 40%.

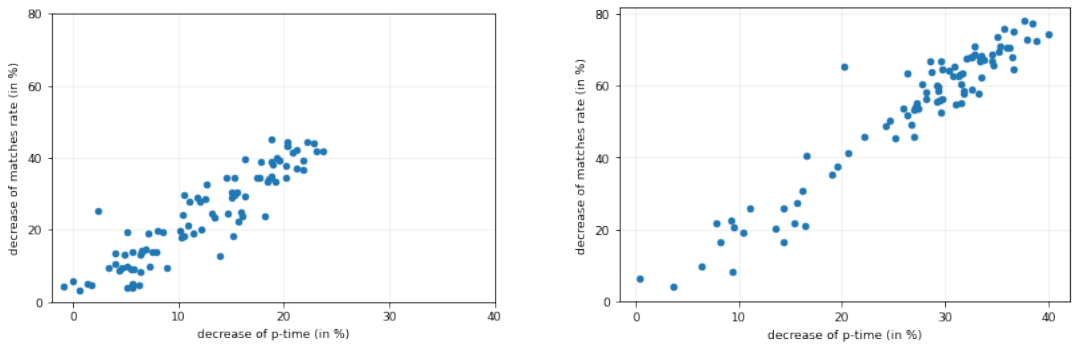

Figure 6.4: Scatter plots showing the reduction in matches rate and processing time

7 Discussion

After defining, implementing and evaluating the solution, the results of the thesis are discussed in this chapter. They are compared with the overall goals of the work. The outcome of the different parts of the implementation is critically questioned and concluded. This serves for highlighting the difficulties of the stated problem of this thesis. Furthermore, it gives insight into the extent to which the found solution is seen to be a valid approach to handling interference with load shedding.

In Chapter 5 the model for the neural network was defined. The chosen approach was to start with the metrics from the model that has proven to work for the prediction of processing times within the same system model. The arrival rate on operator level and the balance score. While the arrival rate showed an strong correlation with the development of the processing time of "Operator 2", the balance score surprisingly did not. Through an in-depth analysis, this behaviour was narrowed down to deviating hardware which lead to a different weighting of matching and sending operations regarding their utilisation of the shared processing unit. As an answer to that phenomena the matches rate was introduced, which made use of the previously discovered impact of the sending operation after matching. While the resulting model of matches rate and arrival rate turned out to also be a feasible feature set, this discovery raises concerns over the application of this exact composition on heterogeneous infrastructure. The result of the model definition of this thesis indicates that for non-comparable hardware used as a multi-operator CEP node, the model would have to be adjusted. More precisely one of the two, balance score or matches rate, would have to be selected, since it is either the sending operation that balances out the matching or said matching procedure weighing out the sending operation. An additional approach could be to discard the two metrics altogether and find a model that is entirely oblivious to changes to the hardware and could therefore handle heterogeneous infrastructure. One of which could consist of only the individual arrival rates per event type, because most of the time there is a way of deriving the metrics from them. This suggests that the information that for example the matches rate conveys is also found in those individual rates. Regardless of the limitations raised here, the defined model proved to be a viable means of approximating the interference between two operators sharing a processing unit.

Chapter 6 started by introducing the approach chosen for enabling the reversed lookup of the neural network. The idea was to simply create a fine-grained table of possible system states, identified by their arrival and matches rate, and assign them the respective processing predicted by the neural network. This way a given bound on the latency can be searched and its target system state retrieved. The following evaluation showed that there are limits to this approach. Recalling the errors to the targeted processing times for reduction higher than 40%, the table came to its lower boundary. In this area the targeted decrease in processing time, in parts, was no longer represented in the table. This was an expected behaviour, since it only consisted of entries that were seen during the learning-phase. If such a scenario was to occur in a real use case the table and the neural network would have to be rebuilt, since this indicates the model no longer represents the conditions under

which the system is running. Nonetheless, despite its simplicity the table approach has shown to work seamlessly for providing the lookup to the model. The overall performance will be discussed at the end of this chapter.

Another crucial part of implementing the workflow was to enable the "Shedder Controller" to transform the targeted arrival and matches rate pairs into applicable shedding configurations. Once a bound and therefore a targeted reduction is set to the processing time, the constructed model is queried and the goal system state is retrieved. For turning this into a shedding configuration proportional to the current system state, the controller first reduced the minimal arrival rate of the three event types to align with the matches rate. Afterwards the remaining two rates are matched to reduce to the point of the remaining reduction needed, set by the arrival rate of the goal system state. The limitations to this approach are quite obvious and were already mentioned in Section 6.1. This procedure only works for evenly distributed patterns. With a different distribution or even another more complex type of pattern this would fail because then it exploits the assumption that the minimal individual arrival equals the matches rate. This is not the case in other scenarios. Furthermore, the described procedure only returns an exact shedding configuration if the current system state allows for both variables, the arrival rate and matches rate, to be met. This is a more general problem because given a targeted system state that consists of multiple variables that are derived from an probabilistic prediction, theoretically it always could be that they contradict each other when confronted with a real system state. In this particular experiment design though, the exceptional cases have shown to occur rarely. In the overwhelming majority of executions the procedure was capable of meeting the target arrival rate and matches rate with an exact shedding configuration. Even though the resulting system state fluctuated during the shedding, the deviation in the on average achieved result was within a reasonable margin.

Looking at the overall result of the executed experiments, there are several points to discuss. The first thing to mention is that for some of the executions the implemented workflow worked almost perfectly. Given the already small ranges of processing times, a deviation of 0.78% in the presented example is a formidable result. Furthermore, in more than half of the experiments the error (absolute) ended up being 10% or lower. While there are also way higher deviations seen in the result this indicates that the model and with it the implemented approach is capable of reaching close to a specified bound on the processing time. Some of the positive errors went as high as 21%. In terms of adhering to the exact processing time specified this is an insufficient result. This limitation is likely due to a misrepresentation in the gathered data. Another observation is that approximately one third of the executions overshot the given target. When it comes to ensuring a bound on the processing time these experiments have fulfilled the requirement. Despite that, overshooting by large margins render some of the shedding at "Operator 1" as useless. It suggests that a lower shedding rate could have lead to the same bound being held, meaning that some of the loss in the QoR could have been avoided. Which leads directly to the next point. The achieved reduction in processing time comes at a cost. Namely the loss of QoR on one of the operators. For the shown example that met its requirement closely, the reduction in matches per second was 62.96%. This was traded for an 35% reduction in processing time at the other operator. The main focus of this thesis regarding performance measures was on the timely aspect of the execution. The QoR was only considered while constructing shedding configurations. This is reflected in the result. While exploiting the interference of the two operators in order to achieve processing times reductions up to 40% makes it appear as a powerful means of additionally influencing the latency of the system, one always has to consider the extent of adverse effect on the QoR on its counterpart.

All in all the result of this thesis is a mixed one. On the one side it showed that the interference has a crucial impact on the processing time, which makes it an important consideration when dealing with overloaded operators. On the other side it became apparent that there are limitations to how precise the implemented approach reaches a specific target. Due to the probabilistic nature of the prediction and possible heterogeneity of the infrastructure the achieved results can vary. Furthermore, the solution proposed here puts a great burden on one operator with respect to its QoR. Therefore, the result of this thesis can be seen as strong indicator for the potential of handling interference on multi-operator CEP nodes while also stressing that the chosen approach is in most parts one-dimensional.

8 Summary and Outlook

The task of a CEP system is to detect complex situations in an event stream. Because most applications require near real-time reaction to these situations, there is a fixed bound on the end-to-end latency. If an operator, which computes the complex patterns, is overloaded due to load peaks in the event stream, the bound on the latency might not be possible to keep. In this case, given that there is no alternative e.g. in a scenario with limited resources, load shedding is used in order to mitigate the impact on the latency. When one or more operators reside at the same hardware node they are subject to interference from each other. The goal of this thesis was to handle this interference in order to reduce the processing times of events and therefore the latency.

The first task for that was to define a model that can reliably represent the impact of the interference from one operator on the other. Previous work indicated that the arrival rate and the balance score, which assesses the distribution of individual arrival rates by event type in relation to their occurrences in the pattern, are two metrics that fulfil this requirement. Testing this assumption lead to a different result in this thesis. While the arrival rate was a good candidate for the model as well, the balance score showed next to no valuable correlation to the processing time of the second operator. This was due to the usage of different hardware for executing the scenario. The hardware that was used here seems to result in costlier sending operations after matching. Based on that finding, the balance score metric was replaced with the matches rate which showed a positive correlation.

For the purpose of integrating the workflow of handling interference into the existing CEP framework, a new component called “Shedder Controller” was introduced. Its tasks are to train a neural network, build a lookup table for target system states and to construct shedding configurations that lead to said system states. The table is designed to contain arrival rate and matches rate pairs alongside a specific processing time predicted by the neural network, so that it can be queried for a configuration that meets the targeted reduction in processing time the closest. The range of target system states in the table is determined by the minimal and maximal values of the metrics seen during the learning-phase. When a reduction is needed the “Shedder Controller” retrieves the configuration, constructs the shedding configuration with a defined procedure and updates the shedder of the operator.

This workflow was then evaluated. The evaluation showed that with this approach it is possible to reach a targeted reduction in processing time with an error down to 0.78%. In contrast to that it also became apparent that in numerous cases the error is a lot higher. Most prominently the results that targeted a reduction close or exceeding the limit of what was seen during the learning-phase fell short of their goal. Furthermore, other target processing times were undercut with errors up to 21%. Theoretically this means that the bound on the processing time still holds but it could be a deal breaker depending on the application. Additionally the workflow was compared to an experiment execution with a naive approach. The intention behind that was to control the outcome of the implemented solution, ensuring that it had a real impact on the results. The effect of the naive approach drew a clear picture. In nearly every execution the error was negative, reaching down to

−49%, meaning that the resulted processing was 49% larger than the one targeted. In comparison the implemented workflow made a significant difference. An additional aspect highlighted in the evaluation was the loss in QoR that was induced by load shedding, making clear that the reduction of the processing time on "Operator 2" was only possible due to a large sacrifice of achieved pattern matches on "Operator 1".

The evaluation was followed by a discussion on the result of the entire thesis. It highlighted the limitations of the approach and further clarified how the results are to be interpreted. The thesis had a stark focus on achieving the reduction in processing time. This result was achieved. When taking a global perspective of the performance of a CEP system this is only one dimension of the overall performance. This lead to certain aspects being neglected. In a holistic view on the system the adverse effect on the QoR of "Operator 1" and the precision on achieving targeted processing times would be of greater importance.

As the results of this thesis strongly indicate that the handling of the interference effect on a multi-operator CEP node is an influential tool for reducing processing times, a variety of new research questions arise. In the following an outlook on the three most promising future works is given.

Like already mentioned, this work lacked a holistic view on the performance of the CEP system. Instead of focusing on only one dimension, the implemented solution could be incorporated into other existing approaches. The handling of interference could unfold its full potential in a scenario where it is used to contribute to meeting a latency bound. There it would be merely an additional means reducing processing time like for example parallelisation or load shedding directly on the concerned node. The combination can distribute the loss in QoR more evenly. Furthermore, it can be part of a global optimisation procedure, assessing the utility of the loss with the help of sophisticated techniques, thus further mitigating the negative effects. There are a few important solutions that would have to be found in order to realise such an approach. One of which would be that the system needs a way of globally balancing the effect of the interference. This thesis tested a scenario in which the interference was measured unidirectional. In order to find an optimal solution a bidirectional assessment may have to be considered.

One reason for the resulting processing times deviating from their target could be that the model and the table miss out on certain characteristics of the underlying process. Therefore it could be an interesting endeavour to experiment with other ways of modelling the inference and generating shedding configurations. The application of the "Shedder Controller" is explicitly designed to enable such an attempt. For that the implementation of the workflow is decoupled from the one for constructing the lookup table including the fitting of the neural network. This way the chosen model is interchangeable. One promising candidate for building such a model is using an approach inspired by generative adversarial networks (GANs). GANs are almost distinctively used for generating images. A prominent example is the generation of human faces resulting in deceptively real images[1] of people that do not exist [KLA+19]. The idea is to use two neural networks that compete against each other. This leads to one of the networks learning a reversed model that has an output in the same form as the input of its counterpart. The approach could form a foundation for designing a model that can generate shedding configurations for specific processing times. Instead of randomly generating output it would have to be able to output a specific target configuration. This is only one

[1] https://thispersondoesnotexist.com/ (last followed on December 07, 2021)

direction, an attempt for a different model could be heading. Others are also imaginable. The only requirement for the introduction of a new model is that it takes variables that can be gathered as an input and allows for querying shedding configurations given a processing time target.

Another viewpoint a future work could take, is to look at a different facet of the interference. The operators of a CEP system form a DAG. In an edge computing scenario this graph can be distributed over several limited nodes. The scenario concerned in this thesis was about two operators sharing a node without any further interaction between them. There could also be a different scenario. If "Operator 2" would follow "Operator 1" in the graph so that at some point the complex output events of "Operator 1" will end up at "Operator 2" the situation gets a new dimension. This scenario creates a downstream effect. The shedding of events at the first operator will have an effect on the load of the second operator. Therefore, the interference is no longer limited to the shared processing unit but also extended to said downstream effect. A possible task could be to incorporate that effect in the existing model.

Bibliography

[CM12] G. Cugola, A. Margara. "Processing Flows of Information: From Data Stream to Complex Event Processing". In: *ACM Comput. Surv.* 44.3 (June 2012). ISSN: 0360-0300. DOI: 10.1145/2187671.2187677. URL: https://doi.org/10.1145/2187671.2187677 (cit. on pp. 17, 18).

[CM15] G. Cugola, A. Margara. "The Complex Event Processing Paradigm". In: Jan. 2015. ISBN: 978-3-319-20061-3. DOI: 10.1007/978-3-319-20062-0_6 (cit. on p. 17).

[EB09] M. Eckert, F. Bry. "Aktuelles Schlagwort: Complex Event Processing (CEP)". In: *Informatik Spektrum* 32.2 (2009), pp. 163–167. URL: http://www.pms.ifi.lmu.de/publikationen/#PMS-FB-2009-5 (cit. on p. 17).

[Gla21] S. Glaub. "Modeling Interferences of CEP Operators on Limited Resources". MA thesis. University of Stuttgart, May 2021 (cit. on pp. 23, 24, 30, 38).

[HBN13] Y. He, S. Barman, J. F. Naughton. *On Load Shedding in Complex Event Processing*. 2013. arXiv: 1312.4283 [cs.DB] (cit. on p. 20).

[Hed20] U. Hedtstück. *Complex Event Processing: Verarbeitung von Ereignismustern in Datenströmen*. Springer Berlin Heidelberg, 2020. ISBN: 9783662615751. URL: https://books.google.de/books?id=A76MzQEACAAJ (cit. on pp. 18, 19).

[Ins21] Institute for Parallel and Distributed Systems, University of Stuttgart. *PRECEPT II*. Dec. 2021. URL: https://www.ipvs.uni-stuttgart.de/departments/vs/research/projects/precept2/ (cit. on p. 21).

[Jac18] P. J. R. E. Jacek Welc. *Applied Regression Analysis for Business*. first. Springer International Publishing, 2018. DOI: https://doi.org/10.1007/978-3-319-71156-0 (cit. on p. 27).

[KLA+19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *CoRR* abs/1912.04958 (2019). arXiv: 1912.04958. URL: http://arxiv.org/abs/1912.04958 (cit. on p. 56).

[Nie15] M. Nielsen. *Neural Networks and Deep Learning*. 2015. URL: http://neuralnetworksanddeeplearning.com/index.html (cit. on pp. 25, 26).

[RBR19] H. Röger, S. Bhowmik, K. Rothermel. "Combining It All: Cost Minimal and Low-Latency Stream Processing across Distributed Heterogeneous Infrastructures". In: *Proceedings of the 20th International Middleware Conference*. Middleware '19. Davis, CA, USA: Association for Computing Machinery, 2019, pp. 255–267. ISBN: 9781450370097. DOI: 10.1145/3361525.3361551. URL: https://doi.org/10.1145/3361525.3361551 (cit. on p. 29).

[RW10] D. B. Robins, R. Wa. "Complex Event Processing". In: *In CSEP 504*. 2010 (cit. on p. 19).

[SBFR20] A. Slo, S. Bhowmik, A. Flaig, K. Rothermel. “pSPICE: Partial Match Shedding for Complex Event Processing”. In: *CoRR* abs/2002.04436 (2020). arXiv: 2002.04436. URL: https://arxiv.org/abs/2002.04436 (cit. on pp. 18, 20, 21, 29).

[SBR19] A. Slo, S. Bhowmik, K. Rothermel. “ESPICE: Probabilistic Load Shedding from Input Event Streams in Complex Event Processing”. In: *Proceedings of the 20th International Middleware Conference*. Middleware ’19. Davis, CA, USA: Association for Computing Machinery, 2019, pp. 215–227. ISBN: 9781450370097. DOI: 10.1145/3361525.3361548. URL: https://doi.org/10.1145/3361525.3361548 (cit. on pp. 18–20, 30).

[SBR20] A. Slo, S. Bhowmik, K. Rothermel. “HSPICE: State-Aware Event Shedding in Complex Event Processing”. In: *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems*. DEBS ’20. Montreal, Quebec, Canada: Association for Computing Machinery, 2020, pp. 109–120. ISBN: 9781450380287. DOI: 10.1145/3401025.3401742. URL: https://doi.org/10.1145/3401025.3401742 (cit. on p. 30).

[TÇZ07] N. Tatbul, U. Çetintemel, S. Zdonik. “Staying FIT: Efficient Load Shedding Techniques for Distributed Stream Processing”. In: *Proceedings of the 33rd International Conference on Very Large Data Bases*. VLDB ’07. Vienna, Austria: VLDB Endowment, 2007, pp. 159–170. ISBN: 9781595936493 (cit. on p. 20).

[ZVW20] B. Zhao, N. Q. Viet Hung, M. Weidlich. “Load Shedding for Complex Event Processing: Input-based and State-based Techniques”. In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 2020, pp. 1093–1104. DOI: 10.1109/ICDE48307.2020.00099 (cit. on p. 20).

All links were last followed on December 07, 2021.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature