Reducio: Data Aggregation and Stability Detection for Industrial Processes Using In-Network Computing

Liam Tirpitz
Data Stream Management & Analysis
RWTH Aachen University
Aachen, Germany
tirpitz@cs.rwth-aachen.de

Ike Kunze
Communication & Distributed Systems
RWTH Aachen University
Aachen, Germany
kunze@comsys.rwth-aachen.de

Philipp Niemietz
Manufacturing Technology Institute
RWTH Aachen University
Aachen, Germany
p.niemietz@mti.rwth-aachen.de

Anna Kathrin Gerhardus
Industrial Engineering and Ergonomics
RWTH Aachen University
Aachen, Germany
k.gerhardus@iaw.rwth-aachen.de

Thomas Bergs
Manufacturing Technology Institute
RWTH Aachen University
Aachen, Germany
t.bergs@mti.rwth-aachen.de

Klaus Wehrle
Communication & Distributed Systems
RWTH Aachen University
Aachen, Germany
wehrle@comsys.rwth-aachen.de

Sandra Geisler
Data Stream Management & Analysis
RWTH Aachen University
Aachen, Germany
geisler@cs.rwth-aachen.de

Abstract

Modern manufacturing environments handle increasing numbers of raw data streams that carry large volumes of data. Searching this raw data for anomalous events, such as faults, failures, or degrading product quality, across multiple data sources can help engineers optimize the underlying manufacturing processes. Yet, facilitating corresponding analyses on the massive data streams is challenging: it requires powerful processing platforms, but strict latency requirements or limited bandwidths often make sending the full raw data to suitable, centralized locations (in the cloud or locally) impossible. As a middle ground, the processing logic for finding relevant events can also be placed on the edge, close to the data source, but this still requires high-speed compute capabilities.

In this paper, we show that in-network computing provides an effective solution to this dilemma. In particular, we design *Reducio*, which detects relevant events directly on the data path and dynamically adapts where and in which resolution data is forwarded for further analysis. Underneath, *Reducio* leverages the process semantics of clocked manufacturing processes to first aggregate raw data streams across multiple sensors and independent machines on a shop floor. It then uses the aggregates to detect anomalous events and assess the stability of the underlying processes to switch between data resolutions and identify machine or sensor malfunctions. We demonstrate the practicality of *Reducio* by applying a Tofino prototype to the clocked process of fineblanking in several experiments, which reveal that *Reducio* can detect instabilities in a timely manner while reducing the data volumes by up to 90 % without losing important process information.

DEBS '25, Gothenburg, Sweden
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1332-3/25/06
https://doi.org/10.1145/3701717.3730547

CCS Concepts

• **Networks** → **In-network processing**; **Programmable networks**; • **Information systems** → **Data streaming**.

Keywords

In-Network Computing, Edge Processing, Data Stream Processing, Data Reduction, Anomaly Detection

ACM Reference Format:
Liam Tirpitz, Ike Kunze, Philipp Niemietz, Anna Kathrin Gerhardus, Thomas Bergs, Klaus Wehrle, and Sandra Geisler. 2025. Reducio: Data Aggregation and Stability Detection for Industrial Processes Using In-Network Computing. In *The 19th ACM International Conference on Distributed and Event-based Systems (DEBS '25), June 10–13, 2025, Gothenburg, Sweden.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3701717.3730547

1 Introduction

In modern manufacturing environments, machines and their connected sensors generate vast amounts of raw data at increasing velocities [42]. Collecting important data across all relevant streams of a single machine or a connected shop floor can be used to optimize maintenance intervals [47], predict the quality of manufactured parts [30], or detect anomalous events [7]. The latter is of particular interest as anomalies are often linked to a decreased manufacturing quality [13] and can also give early indications for problems with process resources, such as impending machine failures. For example, clocked and discrete (mass production) processes are typically set up once, after which their operations are executed for thousands of repetitions (cycles) with minimal fluctuations. Phases with larger deviations, thus, represent anomalies and can provide crucial information on the physical state of the process.

To run retrospective analysis, data can be stored in data lakes [12], enabling a holistic view of past manufacturing steps. However, the large volume of raw data, paired with its high velocity, puts a significant burden on the communication and storage infrastructure [9].

In particular, manufacturing companies need to either invest in expensive on-premise compute resources or rely on cloud solutions and require networks capable of handling the raw data volumes. Consequently, naïve concepts scale poorly to modern manufacturing environments with many machines and sensors, so companies often avoid deploying automated solutions. Hence, there is a strong need to sensibly reduce the load on the infrastructure and the network edge, without losing relevant process information.

Focusing on the detection and analysis of anomalous events mentioned above, most data, by definition, does not exhibit anomalies and is, therefore, of limited interest. Hence, resource-constrained processes can be optimized by first detecting anomalies and then only persisting data related to these occurrences in full for closer inspection later [1]. During stable phases without anomalies, coarsely tracking the process via sampled data can suffice [20]. The key challenge lies in the timely assessment of the process state and detection of anomalous events, which requires analyzing multiple massive data streams in near real-time. Suitable approaches are needed that facilitate aggregation, processing, analysis, and decision-making on heterogeneous hardware across multiple connected data streams in the cloud-edge continuum [15, 44, 45]. Possible target platforms range from powerful server hardware to low-power edge devices.

In-network computing (INC) opens up a new solution space by utilizing the high processing rates of programmable networking devices (PNDs) and their privileged position on the data path. Indeed, the broader related work has already studied executing general functions, such as windowing [4], filtering, mapping, and joining [10] on PNDs. Previous research also worked on accelerating queries in established frameworks such as Apache Spark using functions in the network [36] and supporting complex event processing with PNDs [17]. In contrast, most works studying INC in industrial settings only perform simple per-packet processing and usually ignore the larger semantic context [5, 11, 18, 31, 32]. However, these semantics are important for handling complex events in industrial data streams, which requires analyzing the raw sensor data for base events in the first place. While recent works have started exploring this new dimension [22, 40], INC approaches fully leveraging these semantics for data reduction are still missing.

In this paper, we present *Reducio*, an adaptive analytic system targeting clocked and discrete industrial processes. In short, *Reducio* can (i) summarize entire process cycles in a handful of characteristic values, (ii) combine these values across multiple sensors and multiple machines, and (iii) use this information for assessing the stability of the underlying physical processes and detecting anomalous events in the process. Based on this assessment, *Reducio* adaptively forwards data in different resolutions, e.g., using full resolution for anomalous cycles and only forwarding the characteristic values otherwise. For this paper, we prototype *Reducio* on the Intel Tofino [37], demonstrating its capabilities with fineblanking, an example of a clocked and discrete process. Our evaluation on large data series from a research fineblanking line shows that by aggregating event data and detecting anomalous events, we can significantly reduce the transmitted data volumes while still providing the required information at most times. Additionally, *Reducio* scales to medium-sized shop floors with tens to hundreds of independent processes as long as they use sensors that can be captured with the same metrics. With *Reducio*, we provide a building block toward network-accelerated data stream processing on the industrial edge. Overall, the contributions of this paper are as follows:

- *Reducio*, our modular, event-based INC system for data aggregation on the edge, leverages the semantics and cyclic behavior of clocked processes to substantially reduce data rates while preserving key information.
- Our Tofino prototype robustly detects individual process cycles, aggregates their raw data in a few metrics, determines process stability and adjusts the forwarded data resolution.
- We demonstrate *Reducio*'s feasibility by applying it to the fineblanking process and show that we can reduce data volumes by up to 90 % while still being able to detect harmful events, such as equipment failures and process instabilities.

Structure. In Sec. 2, we introduce clocked industrial processes and outline how their repetitive nature can be leveraged for intelligent data reduction. We also discuss related work on industrial data processing and motivate the potential for INC. Based on these considerations, we design *Reducio* in Sec. 3, present our P4-based prototype targeting clocked processes in Sec. 4, and show how we can apply it to fineblanking in Sec. 5. We extensively evaluate *Reducio* with data from a research fineblanking line in Sec. 6 and broadly discuss the applicability of *Reducio* beyond our example use case in Sec. 7. We conclude this paper in Sec. 8.

2 Industrial Data Processing

The increasing interconnection and monitoring of industrial processes, e.g., encouraged by Industry 4.0 [23], International Data Spaces [28], or the Internet of Production [29], lead to high data volumes with a high velocity of up to several Gbps, significantly challenging today's communication and processing infrastructures [9]. In the following, we illustrate the requirements of industrial settings with the example of *clocked and discrete processes* before discussing related work in the edge-cloud continuum.

2.1 Monitoring Clocked Manufacturing

Clocked and discrete processes are often used in mass production and consist of a fixed set of manufacturing operations that are repeatedly executed for a large number of cycles. For example, stamping processes are executed with 15 to 1400 strokes per minute depending on the concrete process setup and the used machinery, resulting in consistent process signals during execution (cf. Fig. 1.1-Fig. 1.2). As the processes usually run without changing parameters, differences between cycles can hint at changes in the physical process state (Fig. 1.3-Fig. 1.4). Thus, monitoring process signals, both on short time scales and over longer periods, can help to detect anomalous events and changes in process behavior.

How to monitor clocked processes? During manufacturing operations, directly measuring actual physical conditions, such as stresses and tensions, is often infeasible as the tools are typically inaccessible. Similarly, mathematical or simulation models cannot describe the conditions with sufficient accuracy, as they cannot account for the stochastic nature of, e.g., material or machine components. Instead, sensor systems are deployed to record auxiliary signals. Depending on the process, various signals are relevant, either independently or in combination, such as forces (Fig. 1.1-Fig. 1.3) or acoustic emissions (Fig. 1.4) in sheet metal forming or

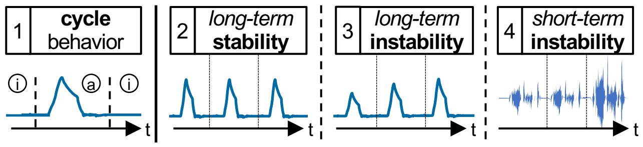

Figure 1: Clocked process signals feature low (inactive ⓘ) and high (active ⓐ) deflection areas (1). Consistent cycle-to-cycle manifestations characterize a stable state (2). Long- (3) or short-term (4) fluctuations can indicate problems.

pressure in die casting. In practice, signals are often acquired at high frequencies (e.g., up to 1 MHz in sheet metal forming [38]) across multiple sensors, resulting in vast quantities of data that need to be transmitted, analyzed, and stored for each cycle.

How to interpret the signals? Once a mass production process is set up and optimized, engineers aim to maintain consistent process conditions such that signal variations can indicate undesirable deviations in the physical process state. In a stable state, minimal signal deviations between cycles are considered normal and are often caused by machine dynamics (Fig. 1.2). Larger instabilities, e.g., over longer periods (Fig. 1.3) or on short time scales (Fig. 1.4), can indicate machine damage or a decrease in workpiece quality. Hence, monitoring the signals can help, e.g., change the tool early enough to reduce waste and avoid longer downtime.

Challenges for process monitoring. The collective sensor volumes are often too large for direct processing as, e.g., shop floors of smaller to medium-sized companies are already strained when operating multiple acoustic emission sensors as their infrastructure is not designed for such tasks. Reducing the data as much as possible is a straightforward solution to allow for scalability, but it comes with an inherent trade-off: data should be reduced only to the extent that it can still satisfy the requirements of associated use cases, e.g., the detection of anomalous events, even when this requires combining information from multiple independent sensor streams. Furthermore, sensors may fail themselves, e.g., due to the harsh manufacturing environments, which in turn causes problems for data-driven approaches relying on them. To better illustrate real use case needs, we next present two example scenarios.

2.1.1 Scenario 1: Predicting Workpiece Quality. Workpieces are typically inspected manually in regular intervals, e.g., every 500 produced workpieces, to verify that they meet defined quality requirements. However, such inspections do not allow assessing the quality of *all* workpieces, which is commonly thought to be infeasible. Addressing this gap, previous studies have proposed predicting workpiece quality based on process signals, postulating that fluctuations exceeding a specific threshold may signify quality degradation. Havinga et al. [13], e.g., trace quality deviations back to problems with machine components by closely analyzing captured signals for which they require high-resolution data. This approach can be realized by analyzing all cycles with full resolution, which, however, causes high data volumes and strains the infrastructure.

Solution angle and requirements. As workpiece quality is only expected to be affected in unstable phases, an alternative is to only cover unstable cycles with high-resolution data while data for the remaining cycles can be subsampled for scalability. This variant requires an *accurate* online assessment of process stability, e.g., based on long-term signal fluctuations (Fig. 1.2 vs. Fig. 1.3), to ensure high coverage, i.e., capture all cycles with fluctuations. If feasible, this option has immense potential for data reduction and may significantly relieve the infrastructure while still ensuring that the quality of workpieces likely to exhibit degradation can be assessed. Under the reasonable assumption that only a few machines experience anomalies at a given time, we can scale this approach to many machines and larger shop floors while limiting the requirements on the network infrastructure.

2.1.2 Scenario 2: Detecting Machine Failures. Tools for clocked processes are typically used reliably for thousands of operations, but at some point, their quality starts to degrade quickly, and there may be additional failures in the machine setup or the auxiliary machinery. Promptly detecting such failures is crucial to prevent subsequent damage to other machine components. Monitoring process signals can enable the detection of equipment malfunction sufficiently in advance of a critical failure [27] as, e.g., sudden drastic changes in acoustic emissions can be an early indicator (cf. Fig. 1.4). Yet, this concept requires analyzing each process cycle, again causing high data volumes and latencies if done remotely.

Solution angle and requirements. Executing such analyses close to the process can reduce data volumes but requires a *fast* online assessment of the process stability across multiple data streams such that the latency of detecting potentially dangerous deviations in the sensor signals is low enough for timely reactions.

In summary, closely monitoring industrial processes can help strengthen automated decision-making in production while the process monitoring itself can be optimized using process knowledge. Next, we discuss the most relevant related work and why it does not provide appropriate solutions for the described scenarios.

2.2 Related Work on Industrial Data Processing

State-of-the-art data stream processing systems, such as Apache Flink [2] or Spark Streaming [43], can flexibly analyze large data volumes in near real-time. However, they assume homogeneous and powerful hardware as it exists in cloud environments [45] while industrial settings are heterogeneous and often resource-constrained [34] in terms of compute and network bandwidth toward centralized (cloud) environments. Recent stream processing systems consider such edge settings [25, 35, 45] and collaborative cloud-edge environments [15, 41]. However, the entailed edge components still require infrastructure and compute resources capable of handling massive data streams. Addressing these shortcomings and aiming for cost-effective solutions, work on INC deploys important data stream and event processing operations to networking hardware [4, 17] which provides the needed compute capabilities, including high processing rates and fast reaction times, as well as the required networking capabilities at the same time.

In-network computing. INC typically uses PNDs that have a high-speed data plane and a slower control plane: the former processes packets at very high rates with a restricted set of operations, while the latter can perform arbitrary computations at much slower speeds. Through the use of specialized hardware, the programmable data plane combines the advantages of software-defined behavior with the processing speeds of traditional networking devices, enabling innovative use cases. The strengths of the fast data plane and the privileged position of PNDs — on-path and close to the data source — are well-suited for industrial data stream processing.

Györgyi et al. [11], e.g., reduce industrial traffic on restricted networks by filtering out unnecessary messages based on protocol information, Kunze et al. [18] and Sankaran et al. [31, 32] perform computations on the payload, and Cesen et al. [5] use PNDs for simple decision-making, i.e., triggering emergency stop signals for robots. Overall, these approaches solve their tasks by looking at *individual* packets of a stream which aligns with the main strengths of PNDs but neglects dependencies spanning across longer data series often found in industrial systems (cf. Sec. 2.1).

Recently, some research has taken steps toward leveraging this streaming nature of data on PNDs, which is significantly more challenging. Laki et al. [22], e.g., consider a robot arm control scenario in which a control plane program first computes movement trajectories and passes waypoints along the planned paths to the data plane of a PND and Kunze et al. [21] monitor change rates in sensor data of continuous processes. These are then compared to position data from the arms, and real-time-critical control signals for the next movements are calculated upon approaching the waypoints. Similarly, Wang et al. [40] control a production line from a PND. As illustrated by these approaches, considering long-term dependencies inherent to the streaming nature of industrial sensor data opens further possibilities for monitoring and controlling machinery.

Research gap. The discussed approaches either perform pure per-packet processing or target low-latency feedback for industrial control applications. However, no solution focuses on enabling the analysis of industrial sensor data, which, in turn, enables advanced applications, such as process anomaly detection [7]. In fact, many of today's manufacturing shop floors lack automated solutions, which means that the mentioned benefits are seldom realized in practice. What is needed are solutions that can dynamically detect and react to changes in the process state, e.g., adaptively reducing the often large volumes of data with regard to the required information.

Contribution. In this paper, we propose *Reducio*, which comprehensively implements and evaluates the processing of industrial data streams on PNDs. With regard to observing long event series on network hardware, we build on early ideas by Kunze et al. [20], who sketched ideas for leveraging longer-term dependencies of industrial sensor data using INC pipelines. In particular, we extend these considerations by (i) supporting multiple sensors and machines in generic clocked processes with diverse sensor types, (ii) introducing additional process state assessment methodologies, and (iii) considering complex sensor failures. Additionally, we provide a broad experimental evaluation based on the scenarios introduced in Sec. 2.1 using datasets collected at a real fineblanking line (cf. Sec. 6). In the following, we first give a general design overview of *Reducio* in Sec. 3 before we present how our concepts map to a prototypical implementation for the Intel Tofino in Sec. 4.

3 *Reducio* – Design Overview

Reducio leverages the cyclic behavior of clocked and discrete manufacturing processes to reduce the data volumes of industrial sensor streams based on the process context and the unique characteristics of each cycle (cf. Sec. 2.1). While low data resolutions are usually sufficient, full-resolution data must be available for close inspection if anomalous events are detected. *Reducio* maps these considerations to two main components, illustrated in Fig. 2: (1) we identify process cycles (events) using the characteristic sensor profiles and summarize them with a few meaningful sensor- and process-dependent key metrics. (2) We use these summaries to assess the current process state before choosing a suitable data resolution. In the following, we motivate and discuss these building blocks.

3.1 Process Cycle Detection and Aggregation

Reducio relies on the correct separation of subsequent process cycles. To separate cycles independent of the used control system, we exploit discrete process characteristics, which typically have an *active* and *inactive* period, indicated by low and high deflections, as also illustrated in Fig. 1.

Cycle detection. *Reducio* detects individual cycles by tracking at least one sensor signal and identifying the start or end of the active period. To support multi-sensor setups and ensure an unambiguous behavior, we always define a *primary* sensor that triggers the detection. *Reducio* can further correlate the results of all available sensors to identify if the *primary* sensor is broken or provides inaccurate readings, such that we can switch the detection to another source.

Data aggregation. When the *primary* sensor detects a cycle, *Reducio* generates a summary consisting of key metrics, such as maximum, minimum, or mean values, gathered from all available sensors. The metrics depend on the monitored signals and the process. We can then forward the resulting low-volume stream of summary packets to subsequent processing components, or we can use it internally for further analyses. In this paper, we first use the summary packets to assess the state of the process before deciding on the required data resolution for forwarding.

3.2 Stability Assessment and Data Resolution

Reducio uses the cycle summaries to assess process stability. While the exact assessment inherently depends on the characteristics of each process, stability can generally be determined by tracking fluctuations in a series of subsequent cycles. Specifically, a stable process is characterized by little short- and long-term fluctuations (cf. Fig. 1.2), while unstable processes exhibit fluctuations. In stable phases, process signals only show expected deviations in a given window of tolerance, which allows us to safely minimize data resolution to key metrics. During unstable phases, *Reducio* forwards the raw sensor data to allow for detailed monitoring of anomalies.

Long-term instability. Instability in the form of small cycle-to-cycle fluctuations amounting to larger differences over longer durations (cf. Fig. 1.3) is relevant for assessing workpiece quality.

Short-term instability. Other instabilities manifest on shorter time scales (cf. Fig. 1.4). They might require immediate reactions and can often be detected via cycle-to-cycle thresholds.

3.3 Additional Considerations

Our design allows observing process signals across multiple sources, for which *Reducio* needs to be configured to the concrete setup.

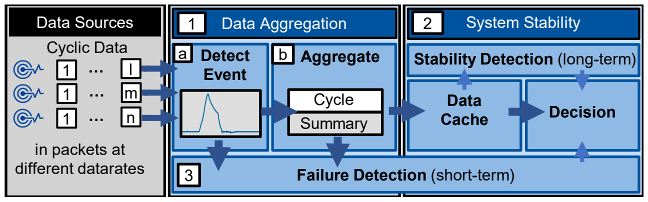

Figure 2: Our system consists of three main components. The *Data Aggregation Unit* detects and aggregates individual process cycles while the *System Stability Detection Unit* uses the aggregated information to assess the process stability. The *Failure Detection Unit* reacts to immediate deviations.

Configuring *Reducio* for a specific process. When applying *Reducio* to a new process, it needs to be adapted to the process characteristics and the deployed sensor profiles. In particular, we need to (i) parameterize the cycle detection to correctly distinguish active and inactive periods, (ii) choose key metrics, and (iii) configure the stability assessment for the specific properties of the process.
Subsequent processing steps. *Reducio* provides a basic building block for event detection and identification for a whole class of industrial manufacturing processes. Depending on the process, different subsequent processing steps could, e.g., directly trigger behavior on the shop floor or automatically tune process parameters. Furthermore, *Reducio* could integrate with data-stream- and complex-event-processing frameworks, providing intelligent data reduction capabilities beyond single machines across connected shop floors. However, in this paper, we focus on *Reducio*'s core abilities and how we can realize them in high-performance networking hardware, specifically the Intel Tofino platform [14].

4 *Reducio* – Tofino-based Prototype

We prototype *Reducio* for Tofino, a PND that enables processing rates of up to 100 Gbps per port, and implement our concept with three main components as shown in Fig. 2: $\boxed{1}$ a data aggregation unit that detects process cycles and summarizes the related portion of the sensor data stream using key metrics, $\boxed{2}$ a system stability detection unit that uses the summaries to assess process stability, and $\boxed{3}$ a failure detection unit that monitors the sensor signals (and their relations) for deviations to enable fast reactions to sensor or machine failures. Next, we briefly summarize relevant conceptual specifics of Tofino before presenting our prototype.
Platform specifics. The Tofino uses a pipeline-based, per-packet processing model to achieve high processing rates on programmable ASICs [14, 19]. Operations and memory are statically assigned to a specific pipeline stage and can be used once per pipeline pass. Each packet passes each stage once. We can perform a fixed number of consecutive operations, which are limited to those that can be executed fast and with predictable latency: multiplications are possible if one factor is statically defined and access to stateful memory (i.e., registers) has to be atomic, limiting the program to simple read-update-write operations. These pipelines can be defined using the P4 programming language. While registers can be grouped in arrays, we can only access one entry for each packet. Each packet can either be (a) forwarded to an external target, (b) recirculated, i.e., reinserted at the front, or (c) sent to the control plane. From the general-purpose control plane, we can read and write the current state from and to the registers. With these specifics in mind, we next present the components of our prototype, focusing on relevant details and challenges when realizing the concepts from Sec. 3.

4.1 Data Aggregation Unit

The *data aggregation* unit detects process cycles and aggregates sensor data streams as conceptualized in Sec. 3.1. To enable aggregation over massive data streams and the detection of events with low latency, its fundamental components are implemented entirely in the Tofino data plane with additional failover functionality provided by the control plane as described in the following.

4.1.1 Event Detection. We detect process cycles (events) in sensor data streams via the pronounced active and inactive periods of sensor profiles of clocked processes (cf. Sec. 3.1).
Sensor profile tracking. Events are usually characterized by high deflections of the observed sensor signal, which settle after the event is completed. Our event detection, thus, monitors the sensor data streams to identify events based on two parameters: (i) a *noise threshold* γ, and (ii) a *phase change delay* n_d. In short, sensor readings below γ are generally too low to be part of the actual event and most likely correspond to the inactive phase characterized by noise areas before and after the event. We further require n_d consecutive readings below the noise threshold γ to consider an event completed to filter outliers and avoid false detections.
Parameterization. γ and n_d are process-specific parameters that need tuning when the process changes. Currently, we manually analyze data samples of the process and determine γ and n_d based on domain expertise. In the future, the parameters could also be learned automatically, e.g., by initially forwarding the raw data to the control plane for a few strokes, where we could then detect the process frequency [39] and tune the parameters accordingly. Since the data plane configuration can be updated at runtime, parameters can also be adjusted for changes in process behavior.
Multiple sensors and sensor failover. *Reducio* can simultaneously monitor multiple sensors, each sensor using an individual parameterization of γ and n_d. For an unambiguous cycle detection, we always designate a single *primary* sensor per machine and collect event data across all sensors associated with the machine simultaneously when the primary sensor detects a cycle. However, individual sensors might fail due to the harsh manufacturing environments. Hence, to mitigate failures of the primary sensor, our prototype supports sensor failover. For this, the control plane constantly compares the event detection rates across all available sensors. If a majority of sensors detect events at a different rate than the primary sensor, we switch to another sensor. In contrast to approaches that specifically require redundant sensor signals [24], our solution can consider different sensor types and frequencies. Additionally, it is sensitive to a larger range of failure modes beyond complete sensor failures as it explicitly leverages the semantics of process cycles instead of plainly comparing raw data streams.

4.1.2 Aggregation. We summarize the data between the start and end of a cycle using maximum, minimum, and mean values. We track each metric for each event in a dedicated P4 register. For

minimum and maximum values, we constantly compare the current value with the current maximum and minimum. However, the specifics of Tofino do not allow directly aggregating the mean. Instead, we capture the sum of all sensor values and the number of values for each event, which can be processed to the mean value during any subsequent processing on general-purpose hardware.
Scope. In general, our system can calculate characteristics for multiple streams at the same time, either for each stream independently or by considering multiple streams for the same metric. For example, we could determine the maximum value measured across multiple sensors or the maximum for each sensor individually. Furthermore, a single sensor stream can be part of multiple aggregations, and multiple machines can be observed individually.
Collection. Once we detect the end of an event based on the primary sensor, we collect the aggregated values from the registers belonging to related sensors, i.e., all sensors belonging to the same machine, in a single *summary packet* and reset the register values for the next event. To generate and emit the summary packet immediately, we recirculate a single packet per event as we need to read state from multiple stages. We then use the aggregated information to assess the stability of the underlying manufacturing process, which we describe in more detail in the following.

4.2 System Stability Detection Unit

The *system stability detection* unit monitors event data over a series of events to assess the *long-term* stability of the process, realizing the concepts presented in Sec. 3.2. For this, we need to define a window of considered sensor readings and a way to analyze the gradient and standard deviation. Due to the complexity of the involved compute operations, we split the assessment logic to the joint capabilities of the data and control plane: a ring-based cache runs on the data plane and holds the n latest values, while we analyze the gradient and standard deviation on the control plane to assess the stability.
Aggregate cache. The first part of the stability detection is a ring-based cache that buffers a specific metric for the latest n events. We implement this cache in the data plane using one register array and one register to account for Tofino's memory access patterns: the register array stores the actual values, while the single register stores a pointer to the array position holding the latest value.
Gradient and standard deviation analyzer. The actual stability detection uses the data from the aggregate cache and is implemented in the control plane. In regular, configurable intervals, the control plane retrieves the key values of the most recent m events from the data plane cache (where $m < n$) and then assesses the process stability. The standard deviation is calculated of the last $k < n$ values, where k is configurable and determines the reactiveness toward phase changes: smaller values result in faster phase changes but may also cause more incorrect change detections. The analyzer retrieves only m new values that are not yet cached in the control plane and no more than the k values needed for the currently relevant stability metrics. Finally, the analyzer yields a classification of the process stability. The system could be configured to detect multiple different classes. In the simplest case, we use a binary classification, differentiating between stable and unstable.
Decision logic. The data plane labels each summary with the current classification. The control plane reconfigures the forwarding

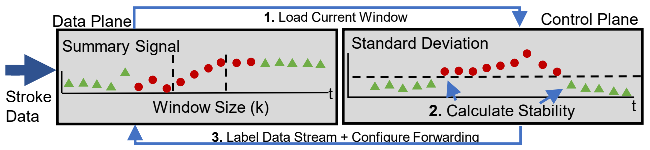

Figure 3: The *System Stability Detection Unit* uses the aggregated cycle data to assess the process stability, e.g., based on the standard deviation and reacts by labeling packets and configuring forwarding accordingly.

behavior of the data plane in case the stability assessment changes, forwarding the aggregated cycle summaries in stable phases and the raw stream if instability is detected.

4.3 Failure Detection

In addition to assessing the long-term stability, *Reducio* can also detect short-term machine malfunctions and cope with sensor failures. As described in Sec. 3.2, we check for system failures by defining thresholds for (i) absolute differences in a signal between cycles or between multiple signals in the same cycle, or (ii) absolute values of individual metrics. This enables us to detect anomalies based on rapid changes in individual signals. Further, we achieve robustness through observing and combining data across sources and detecting deviations between them. For example, multiple sensors of the same type can be observed and deviations between them may indicate an anomalous state of the machine or sensor. Through additional sensors, those states can be differentiated. As part of our failure detection unit, we analyze the stability on multiple sensor signals independently and determine the overall stability as a majority vote across all sources (cf. Sec. 4.1.1). Note, that this semantic approach also works across different signal types and sensor profiles, as long as all combined sources exhibit process stability characteristics.

5 Applying *Reducio* to Fineblanking

Reducio is applicable to many industrial processes but needs to be configured for the unique sensor profiles and dynamics of each process to maximize its effectiveness. Hence, to show *Reducio*'s practical feasibility and demonstrate how it can support specific requirements, we apply our prototype to one example process with different sensor types — fineblanking.

5.1 Intricacies of Fineblanking

Fineblanking is a sheet metal forming process designed for high production quality, e.g., in the automotive and aerospace industries [16]. It uses a triple-acting press where three characteristic forces (punch, V-ring, and counterpunch force) interact to form a complex load collective and perform the actual blanking of the material for workloads of several thousand strokes at speeds of up to 140 strokes per minute. We can monitor each fundamental force signal with one or multiple sensors to detect asymmetries in the load collective. Additionally, acoustic emission (AE) sensors provide auxiliary information on the machine health and can detect early stages of wear and friction [27]. During fineblanking, errors in the lubrication regime or damage to active tool elements can influence the force signal profiles [3]. Likewise, analyzing the profiles can reveal increasing tool wear [26] or deviations in the workpiece quality [13]. Hence, fineblanking is a representative example of clocked and discrete manufacturing processes, as the considerations of Sec. 2.1 apply. Next, we present how we configure *Reducio* to monitor a real fineblanking process.

5.2 Configuring *Reducio* for Fineblanking

Reducio can be configured for the class of clocked and discrete processes, of which fineblanking is one representative.

Stroke detection. Our prototype detects individual strokes by observing steep increases and decreases in various force signals (punch and counterpunch force) as well as an AE signal. Due to its cyclic profile and low noise characteristics, we choose a single punch force sensor as the primary sensor to detect strokes, and we fall back to another source if a failure is detected.

Data aggregation. For each stroke, our data aggregation unit collects the most insightful process-specific key metrics: the maximum positive force applied to the workpiece and the sum of acoustic emissions during the stroke. The maximum force can be used to detect fluctuations between two strokes and to determine the long-term process stability. High acoustic emissions can indicate immediate problems, such as a jammed tool, or help predict future issues, such as tool breakages.

Stability detection. In general, we use the standard deviation of the captured maximum positive force over a window of strokes to detect instability, where a standard deviation over a certain threshold indicates unstable behavior. However, in addition to stable and unstable phases, the fineblanking process also entails an additional *ramp-up* phase. In this state, the machine needs some time to warm up, and the measured forces constantly increase before stabilizing. In this paper, we focus primarily on the default phases of *Reducio*: *stable* and *unstable*. Hence, we detect and filter the initial ramp-up phase by monitoring the gradient over a window of strokes and consider the ramp-up to be completed once the gradient levels out. However, by utilizing the modularity of our approach, we could also configure *Reducio* to explicitly consider the ramp-up behavior as a dedicated process phase and classify packets accordingly.

Failure detection. We configure an additional failure detection mechanism that incorporates the AE sensor as it is well-suited for early detection of critical process failures [27]. In particular, when observing a sudden drastic increase in the AE signal or an out-of-bounds force signal, we detect a critical process failure. Using domain knowledge, we define thresholds for the maximum expected force during a stroke, as well as the expected AE signal energy over the duration of each stroke. In our prototype, we use the sum aggregation function to measure the overall energy by summing up all absolute AE readings for each stroke. After detailing how *Reducio* can be configured for a clocked process, we next assess its performance with real-world data recorded on a fineblanking line.

6 Evaluation

For this evaluation, we focus on *Reducio*'s different components based on our Tofino prototype that we configured for fineblanking. We start by studying how different sensor types and parameterizations influence *Reducio*'s fundamental ability to detect individual strokes before we evaluate *Reducio*'s theoretical aggregation capabilities. We then assess *Reducio* in two larger application scenarios: First, inspired by the goal of predicting workpiece quality based on sensor data (cf. Sec. 2.1.1), we analyze how well *Reducio* can provide critical information on process instabilities while still reducing the data rate. Second, we evaluate *Reducio*'s ability to detect events that require immediate attention, specifically process and sensor failures. Based on our findings, we then discuss the implications for the envisioned deployment scenario of multi-machine shop floors in the subsequent Sec. 7. In the following, we start by presenting our general evaluation setting and methodology.

		Strokes			Unstable Phases		
		Stable				length	
ID	Steel	Yes	No	Hz	#	μ	σ
E1	42CrMo4	15995	4524	50	715	6.3	7.8
E2	C60	23143	5948	50	778	7.6	9.2
E3	16MnCr5	9594	4535	120	221	20.5	92.9
E4	58CrV4	18788	11162	50	851	13.1	40.8
E5	42CrMo4	830	99	50	10	9.8	6.4

Table 1: The experimental datasets E1-E5 differ in the processed steel, the number of stable/unstable strokes, the executed strokes per minute, as well as the number of unstable phases and their length distribution.

6.1 Methodology

We evaluate *Reducio* using sensor data recorded from a real fineblanking line in a research environment during five experiments. The process forces were measured by three piezoelectric sensors placed at different positions of the punch, acquiring data with a frequency of 10 kHz. Additionally, an AE sensor measured mechanical vibrations with a frequency of 1 MHz close to the punch. Table 1 shows the main experiment parameters and statistically characterizes the portion of observed instabilities during each process.

Datasets. The main distinguishing factor of the experiments is the processed steel, whose characteristics change the process forces. For example, the alloyed QT-steel 58CrV4 is significantly harder than the alloyed case hardening steel 16MnCr5, such that process forces are generally higher. As a result, we observed different numbers of unstable phases of varying lengths, which we determined following Niemietz et al. [26]. As *Reducio* fundamentally leverages process stability, these different stability characteristics allow us to assess *Reducio*'s performance from multiple perspectives. Note that *E5* was aborted as the punch broke, resulting in an interesting dataset for the analysis of short-term machine failures.

Evaluation. To enable reproducibility, we replay the four recorded sensor signals from two machines, *M1* and *M2*, through separate network interfaces connected to one Tofino-based switch running *Reducio*. The resulting (potentially reduced) data stream is sent to machine *M2*, where it is captured for analysis. Additionally, we log the state of our control plane, including information about decisions made by our prototype. We use this information to study the performance of our *online* approach, e.g., comparing the detection capabilities of the *Stability Analyzer* with an *offline* algorithm. While the offline algorithm uses the same methodology as described in Sec. 4.2, it determines unstable phases on the full set of captured summary packets instead of making near real-time decisions. Since

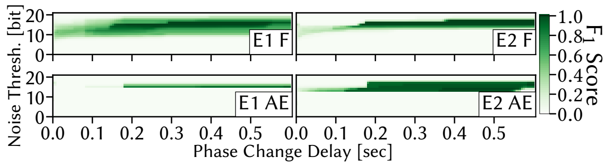

Figure 4: F_1 score (colored) of the stroke detection for different values of *noise threshold* γ and *phase change delay* n_d for the E1 and E2 datasets, and force (F) and acoustic emission (AE) sensors. High scores indicate correctly identified cycles.

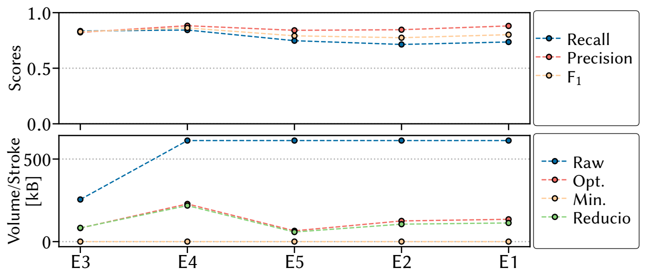

Figure 5: *Reducio*'s practical stability assessment (top) and data reduction (bottom) performance for all datasets, ordered by decreasing mean length of their unstable phases.

the offline approach is not limited by access intervals, it provides an optimal baseline for the timely detection of process instability.

6.2 Data Aggregation Unit

Reducio's first component, the *data aggregation unit*, initially detects individual process cycles and then summarizes each cycle in a single summary packet using key metrics across multiple data streams. In the following, we first study the detection of cyclic events before we turn our focus to the *aggregation* capabilities.

6.2.1 Cycle Detection. The correct detection of individual cycles from the available signals is essential to ensure that exactly one summary packet is generated for each cycle. Different processes, source materials, and sensor types cause signals with different profiles, amplitudes, sampling rates, and noise levels, for which we can configure our detection using the *noise threshold* γ and *phase change delay* n_d. For fineblanking, the *force* signals provide a visually distinguishable profile for each stroke, while *AE* signals are significantly noisier and might make reliable detection more challenging. Hence, we evaluate the impact of the profile and the parameterization on the correctness of the event detection.

Approach. We simulate the *stroke detection* over a range of parameters for the E1 and E2 datasets as they represent materials with sufficiently different processing characteristics and include recorded force and AE data. For the subsequent stability detection, missed strokes increase delay and reduce accuracy. Wrongly detected strokes based on noise could heavily impact the stability metrics and cause the system to falsely claim process instability. Therefore, the cycle detection parameters must be optimized such that the number of detected real strokes is maximized (true positive), the number of missed strokes is minimized (false negative), and the number of detections without a corresponding real stroke is also minimized (false positive). We evaluate the accuracy and robustness of our cycle detection by checking if *Reducio* detected each stroke at the correct point in time compared to the process knowledge contained in the original datasets.

Results. Fig. 4 shows the stroke detection rates from our simulation depending on different parameter settings for γ and n_d, where a darker shade of green indicates a higher F_1 score and, therefore, a good detection of cycles (many true positives, few false positives). The range of available parameter choices varies across the sensor types and the processed material. Regarding the used sensor types, we can, e.g., observe that the force sensor in experiment E1 has a very large range of *good* parameters but cannot be configured *perfectly* with a maximum F_1 score of 0.996. In contrast, the drastically smaller parameter range of the E1 AE sensor allows for perfect stroke detection. Regarding the used material, the AE sensor in E2 has a broader range of parameters compared to its E1 counterpart and still allows for perfect detection, while the force sensor in E2 has a smaller range but yields a slightly higher F_1 score of 0.999.

Takeaway. *We conclude that* Reducio *can reliably detect cycles from different sensor profiles, if parameterized correctly, but sensors used for the cycle detection need to be selected carefully. Ideally, the profiles of such sensors reflect the cyclic behavior of the underlying process and feature a measurable deflection during the active phases. However, the detection parameterization still needs to be adjusted when changing process parameters, e.g., when a different material is processed. Yet, even if sensor profiles are unsuitable or no suitable parameters exist, they can still be part of the subsequent aggregation triggered by another sensor with reliable event detection.*

6.2.2 Data Aggregation. *Reducio*'s data aggregation summarizes the raw data of entire cycles into a single summary packet, significantly reducing the data volume without losing the most important data points that are relevant during regular operation. The efficiency of the data reduction partly depends on the process frequency and the sampling rates of the observed sensors and generally decreases with increasing process speeds and lower sampling rates. We can mathematically determine the *minimum* data volume for a single cycle for varying frequencies based on the packet sizes, i.e., in our case, 51 B packets for raw sensor signals and 97 B for summary packets. For realistic speeds of fineblanking, i.e., up to 140 strokes per minute, and sensor sampling rates of 10 kHz (force) or higher (e.g., AE), *Reducio* could theoretically reduce the data volume in our configuration to less than 0.05 % of the raw data stream, showing the immense potential of such an aggregation technique. In practice, however, *Reducio* only applies the data reduction if it classifies the process as stable. Consequently, the *actual* data volume also depends on the stability properties of the process (cf. Table 1) and the *stability analyzer* component. For that reason, we now explicitly focus on *Reducio*'s overall performance and investigate how well it addresses the requirements outlined in Sec. 2.1.

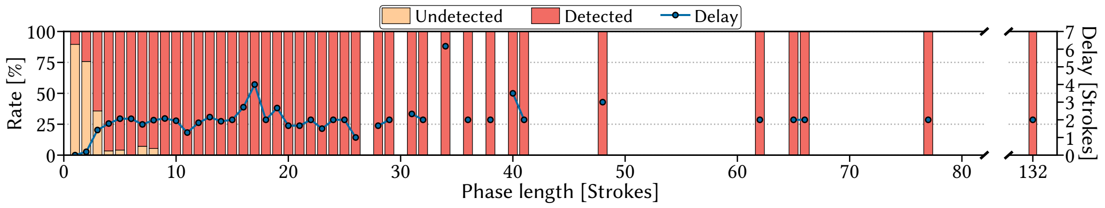

Figure 6: Detection rate and delay for unstable strokes across different phase lengths for the E2 experiment.

6.3 System Stability and Failure Detection

We evaluate *Reducio* based on our general considerations for clocked and discrete processes in Sec. 2.1 with two scenarios of practical relevance. First, we analyze if *Reducio* provides high coverage of unstable strokes with high-resolution data while minimizing the overall data rate. Second, we assess if *Reducio* is able to promptly detect tool and sensor defects at low latencies.

6.3.1 Scenario 1 - Workpiece Quality. In manufacturing, workpieces produced during stable process phases are often considered to be of good quality, while larger-than-normal fluctuations can be indicators of decreased quality. However, fluctuations do not necessarily require the immediate shutdown of the machine. Instead, it is more important to record as many *anomalous* strokes as possible with the full sensor resolution while keeping the data rate as low as possible when the process is operating as expected. Hence, we study *Reducio*'s coverage of anomalous strokes in relation to its potential to save bandwidth using all datasets and configuring it to use a force sensor for the stroke detection.

Coverage. Fig. 5 (top) shows precision, recall, and F_1 scores for the detection of anomalous strokes for all datasets, which we sort based on the mean length of unstable phases from largest (left, E3) to smallest (right, E1). Based on our system design, we expect *Reducio* to perform worse for shorter phases of instability. The recall indicates how well unstable strokes were detected and decreases if unstable strokes are missed. In our use case, this results in the loss of relevant information we intend to collect during anomalies. The precision considers how many of the detected strokes were unstable and decreases if more stable strokes are incorrectly labeled as unstable. In our use case, this results in unnecessarily detailed data transmissions and, therefore, an increased utilization of network bandwidth. Since we want to optimize both metrics, we are interested in the F_1 score as the harmonic mean between recall and precision. For all considered datasets, *Reducio* achieves F_1 scores of over 0.75. As expected, for datasets with longer mean unstable phases (left), *Reducio* performs better.

Volume reduction. Fig. 5 (bottom) shows *Reducio*'s achieved reduced data rate in comparison with (i) the raw data rate, i.e., the baseline case of sending *full resolution* data, (ii) the minimum data rate, i.e., only sending summary packets, and (iii) the theoretical optimum of always forwarding data in the desired level of detail, i.e., sending the raw data for anomalies and the minimum for stable strokes. As can be seen, *Reducio* always performs close to the optimum, only slightly undershooting the theoretical optimum and achieving a reduction to less than 20 % of the original data rate for most experiments. The worse performance for E3 and E4 of around 35 % can be explained by a lower sensor reading rate per stroke due to higher punch frequencies at constant sampling rates (E3) and high rates of unstable phases (both). Additionally, we notice a small systemic delay of *Reducio*, which impacts both the stroke coverage and the volume reduction. Hence, we next study this aspect in more detail using the example of E2.

Detection delay. Fig. 6 illustrates the mean detection delay in strokes (dots) together with the percentage of detected unstable strokes (bars) for unstable phases of different lengths contained in the E2 dataset. We consider an unstable phase to be detected if *Reducio* captures data at full resolution for at least one stroke of the unstable phase. Based on this definition, inspecting Fig. 6 reveals that our approach cannot reliably detect very short phases of unstable strokes. This is reasoned by two aspects: (1) the *Stability Detection* in the control plane reacts to data only *after* a stroke (delay of at least one stroke due to the previous aggregation), and (2) we always compute our stability metrics over windows of multiple strokes. However, as soon as detection becomes possible, we react quickly with delays of only up to three strokes in most cases, with only very few outliers mainly explained by complete sensor outages. This delay also explains why the coverage achieved by *Reducio* depends on the mean length of unstable phases. Overall, our results are in line with our goal of detecting longer unstable phases.

Takeaway. *Our results indicate that our stability detection is generally capable of achieving a good coverage of unstable strokes while drastically reducing the overall data rate for realistic datasets. While* Reducio*, conceptually, cannot detect very short unstable phases, we achieve high coverage of the more important longer unstable phases, which can be detected with reasonable latencies.*

Given *Reducio*'s inability to identify short unstable phases, we next focus on the second scenario and consider machine defects and sensor failures that require short-term reactions.

6.3.2 Scenario 2 - Detecting Failures. *Reducio*'s extensive monitoring capabilities allow for various failure detection mechanisms, which we demonstrate in two examples.

Tool defects. Tool defects and malfunctioning periphery equipment are typical root causes of lasting machine damage, and it is important to detect them quickly. By focusing on the E5 dataset, which contains a critical process failure, we can evaluate *Reducio* with regard to such issues. Specifically, we aim to detect the issue as early as possible, ideally before the actual failure occurs. For this, we set up *Reducio* with three 10 kHz force sensor data streams used for the stroke detection and an additional 1 MHz AE sensor. We then configure the *failure detection* to raise alarms for out-of-bounds strokes on the AE signal. Analyzing *Reducio*'s behavior in this setting, we find that *Reducio* can predict the tool breakage 61

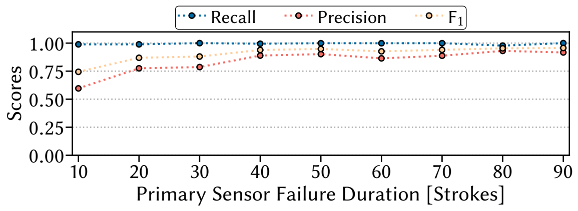

Figure 7: *Reducio*'s use of the primary sensor which periodically fails and works for different durations.

strokes earlier compared to only observing individual force signals, which illustrates how *Reducio*'s capability of tracking multiple heterogeneous sources and considering multi-sensor events can be sensibly used for improving its detection performance.

Sensor defects. In addition to the defects of machine parts, the rough environment of manufacturing machines also increases the probability of sensor failures, which also need to be detected and dealt with as quickly as possible. To limit the impact on production, our goal is to combine results from multiple sensors and switch to reliable sources automatically, aiming to use the designated primary sensor whenever possible. To evaluate this scenario, we load *Reducio* with four copies of a single force sensor but let the primary sensor fail completely in periodic intervals of different durations, i.e., it is active for n strokes, then inactive for n strokes, and so forth. Fig. 7 shows recall, precision, and F_1 scores of our detection mechanism, where true positives are cases where the primary sensor is inactive and we have correctly switched to a secondary sensor, while true negatives mean that we use the primary sensor and it is active. As can be seen, we almost never use the primary sensor if it is offline, while we increasingly use secondary sensors correctly for increasing offline times of the primary sensor. This is because we compare the detection of all tracked sensors and change to a stable sensor as soon as the current sensor is classified as an outlier. Before we switch back to the primary sensor, we require it to be stable for a number of subsequent cycles since we prioritize correct detections over the use of the primary sensor, assuming that the fallback is sufficient in the meantime. Overall, these results show that *Reducio* quickly switches back to the primary sensor as soon as it recovers.

Takeaway. *The two examples show that* Reducio *can effectively identify machine faults and cope with sensor failures by integrating data from multiple sources. While failure handling depends on the process,* Reducio*'s diverse monitoring capabilities enable a large range of possible mechanisms addressing these needs.*

In conclusion, our evaluation shows that *Reducio* is capable of detecting process cycles across heterogeneous sensor sources in a failure-tolerant fashion and uses the aggregated cycle information to detect failures and long-term process instabilities for multiple sensors on a single machine. With this approach, we can reduce the overall data volume by up to 90 % without losing important process information. In the following, we discuss the implications of our findings for the wider envisioned deployment scenario in Sec. 7.

7 Discussion

Manufacturing companies are set to benefit from collecting and analyzing the vast amounts of industrial data streams generated by machines and sensors in modern manufacturing environments [42]. However, many manufacturing companies are still unable to reap any benefits as their shop floors are not fit for handling these data volumes in near real-time, so automated data collection and analysis solutions are largely missing in today's shop floors.

Reducio aims to support near real-time analysis and data collection for clocked and discrete processes, through preprocessing and analyzing data streams on network devices on the edge. For this, we leverage process semantics to significantly reduce data volumes when processes are in a stable state while we still provide high-resolution data for anomalous phases to enable a detailed analysis of, e.g., the workpiece quality when it is needed most. In this paper, we have shown that *Reducio* seamlessly maps to PNDs and that we can effectively track the fineblanking process.

In this section, we discuss how *Reducio* can be adapted for other manufacturing processes with different data processing requirements and how it could be integrated into comprehensive data stream processing platforms for edge environments, delineating that *Reducio* is not limited to the presented fineblanking process.

7.1 Scalability

Reducio relies on anomalies only occurring infrequently. Hence, for larger deployments targeting many machines on the same shop floor, we can provision resource capacities based on the expected data reduction rate (cf. Sec. 6.2) and the number of processes expected to simultaneously be in an anomalous state as *Reducio* drastically reduces the data volumes for all other machines. While this approach cannot handle the worst-case scenario where *all* processes have anomalies at the same time, we argue that this risk is far outweighed by the gain of enabling companies to deploy data-driven analyses in the first place by reducing requirements.

Currently, *Reducio* is deployed on a single, independent PND that is positioned on the data path close to the data source, which enables high communication efficiency and supports multiple connected and independent data streams across different machines on a shop floor. In the future, we aim to broaden our scope and look at diverse, distributed edge systems consisting of multiple PNDs interacting with each other and integrate *Reducio* with general-purpose data stream processing systems for cloud-edge environments, such as NebulaStream [45]. In this setting, we envision data aggregation primitives and event definitions to be deployed on PNDs from such systems, integrated into dynamic, cross-platform processing pipelines. This way, the resulting streams and detected events can be forwarded and processed on general-purpose hardware, enabling, e.g., complex event processing, including event data from heterogeneous sources. By defining abstract events and processing targets, workloads could be deployed on any target in the cloud-edge continuum based on optimization goals and available resources. Due to their privileged position and high processing rates, PNDs could play a critical role in such heterogeneous, multi-platform pipelines.

In addition to these general considerations on the target process, *Reducio*'s scalability also depends on the selected hardware platform. For this paper, we have implemented *Reducio* on a Tofino PND as we target larger deployments with multiple machines.

***Reducio* on Tofino.** The prototype used throughout this paper uses resources on two separate hardware pipelines for data aggregation and stability assessment, running on a single PND. Each pipeline represents an independent hardware block of the Tofino platform. Both pipelines occupy up to 83.3 % of the available stages, although they are mostly only lightly loaded. In particular, our prototype uses less than 10 % of the available TCAM and match-action cross bars while needing up to 20.1 % of the SRAM and 15.6 % of the ALUs of the stability assessment pipeline. Overall, our prototype leaves plenty of room for other functionality such that companies investing in a PND can also co-locate other relevant functionality or scale to larger deployments. If we utilize the full available resources on both pipelines, our Tofino 1 prototype can track up to 10 metrics for a single machine. With a conservative cache size of 500 cycles (i.e., incorporating three minutes in our fineblanking example), *Reducio* can scale up to about 100 independent machines, as long as all machines can be captured with the same set of metrics. This is a limitation of our implementation, which cannot combine multiple metrics in a single physical register. As discussed in Sec. 5.1, industrial companies often operate multiple identical or similar machines in parallel. Therefore, this requirement is fulfilled in those cases. With the more recent Tofino 2 hardware, we can increase the scalability to 18 metrics per machine for up to 100 machines, which shows that the technological advances in PND hardware increasingly alleviate resource bottlenecks. Overall, deploying *Reducio* on Tofino is particularly attractive for large, specialized companies that are interested in tracking a limited number of key metrics across a larger number of similar machines on a single shop floor.

7.2 Applicability

Reducio's reduction concepts are applicable to a wide range of industrial processes, which we demonstrated for fineblanking in this paper. In the following, we discuss *Reducio*'s applicability to other processes.

Related processes. While each clocked process has a unique operating principle, they all have similar characteristics, challenges, and difficulties and produce periodic, repeating signals from which deviations can indicate process instability. Production rates in other stamping processes can exceed 2000 strokes per minute, and typical manufacturing facilities employ approximately 10 to 80 machines in parallel production lines. Hence, these settings are ideal for benefiting from *Reducio*'s data reduction capabilities on the shop floor.

Metrics. For our prototype, we have provided a limited set of core metrics. While sufficient for realizing the analysis for fineblanking and illustrating *Reducio*'s fundamental concepts, tracking other processes might require alternative metrics. For example, related work has shown that tracking statistical measures, such as median, variance, and standard deviation [8], is possible using P4 in general, and reformulations and approximations of mathematical problems [18, 33] can be implemented as metrics on Tofino.

Stability detection. Our stability detection is a two-component mechanism making binary decisions on the process stability from the control plane. However, we can easily adapt this binary notion to further classes by using additional metrics in the control plane and defining new processing and forwarding rules. We could also realize this component entirely in the data plane, e.g., based on related work [6, 46], if even faster responses are required, but the feasibility depends on the specific stability metrics for a given process. *Reducio*, therefore, allows individual analysis and classification modules to be deployed either in the control plane or the data plane. To parameterize these classes, we currently rely on explicitly given domain knowledge. In the future, this process might be automated by observing a longer series of raw or aggregated cycles in advance and learning the expected behavior under normal conditions.

Other applications. Beyond analyzing anomalous phases, *Reducio* can also support other tasks. For example, *Reducio* could automatically trigger workpiece inspections or automatic rejections based on the detected signal deviations. Embedding *Reducio* in the overall process control is another straightforward extension, e.g., to signal to the control system that an action is required to automatically correct the detected deviations. However, possible control responses are highly process- and control-specific such that general solutions are hard to design. Further, through integrating *Reducio* into distributed cloud-edge infrastructures and providing extensible building blocks to define events and processing primitives, generic, event-driven data streams could be observed, analyzed, aggregated, and combined with external information.

8 Conclusion

The growing sensor data streams in modern industrial environments are one important source for expanding domain knowledge and helping to improve manufacturing processes. However, they also pose a challenge for industrial companies whose forwarding, processing, and storing infrastructures are typically not built for such immense volumes. What is needed are solutions that enable these improvements while simultaneously reducing the data rates and the infrastructural load, ideally only keeping those data points that hold important information.

Addressing this need, *Reducio* targets clocked processes, which are characterized by repeatedly executed process cycles with very characteristic sensor profiles. We leverage this knowledge to detect individual process cycles and summarize their information in a handful of key metrics. The metrics allow us to assess the process stability, detect anomalous events, and, ultimately, decide the resulting data resolution to reduce the transmitted data volumes while still providing the required level of information in critical situations. Our evaluation on a real fineblanking process proves that *Reducio* can satisfy the requirements for two distinct application scenarios, and we further demonstrate how different parameterizations allow for performance finetuning of our event and stability detection. With appropriate parameters, *Reducio* accurately summarizes individual events from raw sensor data and detects longer unstable phases reliably, while only very short phases may be missed.

Overall, our work contributes to a vision of hybrid solutions making full use of the edge-cloud continuum as our approach allows, e.g., the summary packets to be sent to a central cloud while analyzing anomalous traffic in more detail on the edge. More practically applicable solutions in this vein will be necessary to enable sustainable and efficient industrial environments of the future.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

References

[1] R. Aaij, S. Akar, and J. Albrecht et al. 2019. Design and Performance of the LHCb Trigger and Full Real-Time Reconstruction in Run 2 of the LHC. *Journal of Instrumentation* 14, 04 (2019), P04013–P04013.

[2] Alexander Alexandrov, Rico Bergmann, and Stephan Ewen et al. 2014. The Stratosphere Platform for Big Data Analytics. *The VLDB Journal* 23, 6 (2014), 939–964.

[3] A. M. Bassiuny, Xiaoli Li, and R. Du. 2007. Fault Diagnosis of Stamping Process Based on Empirical Mode Decomposition and Learning Vector Quantization. *International Journal of Machine Tools and Manufacture* 47, 15 (2007), 2298–2306.

[4] Bochra Boughzala, Christoph Gärtner, and Boris Koldehofe. 2022. Window-based parallel operator execution with in-network computing. In *Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems.*

[5] Fabricio E Rodriguez Cesen, Levente Csikor, Carlos Recalde, Christian Esteve Rothenberg, and Gergely Pongrácz. 2020. Towards Low Latency Industrial Robot Control in Programmable Data Planes. In *Proceedings of the 2020 IEEE Conference on Network Softwarization (NetSoft).*

[6] Roy Friedman, Or Goaz, and Ori Rottenstreich. 2021. Clustreams: Data Plane Clustering. In *Proceedings of the 2021 ACM SIGCOMM Symposium on SDN Research (SOSR).*

[7] Chuanchao Gao, Heejong Park, and Arvind Easwaran. 2021. An Anomaly Detection Framework for Digital Twin Driven Cyber-Physical Systems. In *Proceedings of the 2021 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS).*

[8] Sam Gao, Mark Handley, and Stefano Vissicchio. 2021. Stats 101 in P4: Towards In-Switch Anomaly Detection. In *Proceedings of the 2021 ACM Workshop on Hot Topics in Networks (HotNets).*

[9] René Glebke, Martin Henze, Klaus Wehrle, Philipp Niemietz, Daniel Trauth, Patrick Mattfeld, and Thomas Bergs. 2019. A Case for Integrated Data Processing in Large-Scale Cyber-Physical Systems. In *Proceedings of the 2019 Hawaii International Conference on System Sciences (HICSS).*

[10] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication.*

[11] Csaba Györgyi, Károly Kecskeméti, Péter Vörös, Géza Szabó, and Sándor Laki. 2021. In-Network Solution for Network Traffic Reduction in Industrial Data Communication. In *Proceedings of the 2021 IEEE International Conference on Network Softwarization (NetSoft).*

[12] Rihan Hai, Sandra Geisler, and Christoph Quix. 2016. Constance: An Intelligent Data Lake System. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data (SIGMOD).*

[13] Jos Havinga and Ton van den Boogaard. 2017. Estimating Product-to-Product Variations in Metal Forming Using Force Measurements. *AIP Conference Proceedings* 1896, 1 (2017), 070002.

[14] Intel. 2023. Intel® Tofino™. https://www.intel.com/content/www/us/en/products/details/network-io/intelligent-fabric-processors/tofino.html

[15] Tao Jing, Xitian Tian, Hao Hu, and Liping Ma. 2022. Deep Learning-Based Cloud–Edge Collaboration Framework for Remaining Useful Life Prediction of Machinery. *IEEE Transactions on Industrial Informatics* 18, 10 (2022), 7208–7218.

[16] Fritz Klocke. 2013. *Manufacturing Processes 4.* Springer Berlin Heidelberg.

[17] Thomas Kohler, Ruben Mayer, Frank Dürr, Marius Maaß, Sukanya Bhowmik, and Kurt Rothermel. 2018. P4CEP: Towards In-Network Complex Event Processing. In *Proceedings of the 2018 Morning Workshop on In-Network Computing (NetCompute).*

[18] Ike Kunze, René Glebke, Jan Scheiper, Matthias Bodenbenner, Robert H. Schmitt, and Klaus Wehrle. 2021. Investigating the Applicability of In-Network Computing to Industrial Scenarios. In *Proceedings of the 2021 IEEE International Conference on Industrial Cyber-Physical Systems (ICPS).*

[19] Ike Kunze, Moritz Gunz, David Saam, Klaus Wehrle, and Jan Rüth. 2021. Tofino + P4: A Strong Compound for AQM on High-Speed Networks?. In *Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management.*

[20] Ike Kunze, Philipp Niemietz, Liam Tirpitz, René Glebke, Daniel Trauth, Thomas Bergs, and Klaus Wehrle. 2021. Detecting Out-Of-Control Sensor Signals in Sheet Metal Forming Using In-Network Computing. In *Proceedings of the 2021 IEEE International Symposium on Industrial Electronics (ISIE).*

[21] Ike Kunze, Dominik Scheurenberg, Liam Tirpitz, Sandra Geisler, and Klaus Wehrle. 2024. In-Situ Model Validation for Continuous Processes Using In-Network Computing. In *Proceedings of the 2024 IEEE International Conference on Industrial Cyber-Physical Systems.*

[22] Sándor Laki, Csaba Györgyi, József Pető, Péter Vörös, and Géza Szabó. 2022. In-Network Velocity Control of Industrial Robot Arms. In *Proceedings of the 2022 USENIX Symposium on Networked Systems Design and Implementation (NSDI).*

[23] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. 2014. Industry 4.0. *Business & Information Systems Engineering* 6, 4 (2014), 239–242.

[24] Steffen Lindner, Marco Häberle, Florian Heimgaertner, Naresh Nayak, Sebastian Schildt, Dennis Grewe, Hans Loehr, and Michael Menth. 2020. P4 In-Network Source Protection for Sensor Failover. In *Proceedings of the 2020 IFIP Networking Conference.*

[25] Pinchao Liu, Dilma Da Silva, and Liting Hu. 2021. DART: A Scalable and Adaptive Edge Stream Processing Engine. In *Proceedings of the 2021 USENIX Annual Technical Conference (ATC).*

[26] Philipp Niemietz, Mia J. K. Kornely, Daniel Trauth, and Thomas Bergs. 2022. Relating Wear Stages in Sheet Metal Forming Based on Short- and Long-Term Force Signal Variations. *Journal of Intelligent Manufacturing* 33, 7 (2022), 2143–2155.

[27] Philipp Niemietz, Martin Unterberg, Daniel Trauth, and Thomas Bergs. 2021. Autoencoder Based Wear Assessment in Sheet Metal Forming. *IOP Conference Series: Materials Science and Engineering* 1157, 1 (2021), 012082.

[28] Boris Otto and Matthias Jarke. 2019. Designing a Multi-Sided Data Platform: Findings from the IDS Case. *Electronic Markets* 29, 4 (2019), 561–580.

[29] Jan Pennekamp, René Glebke, and Martin Henze et al. 2019. Towards an Infrastructure Enabling the Internet of Production. In *Proceedings of the 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS).*

[30] Prahalad K. Rao, Jia (Peter) Liu, David Roberson, Zhenyu (James) Kong, and Christopher Williams. 2015. Online Real-Time Quality Monitoring in Additive Manufacturing Processes Using Heterogeneous Sensors. *Journal of Manufacturing Science and Engineering* 137, 6 (2015).

[31] Ganesh C. Sankaran, Joaquin Chung, and Raj Kettimuthu. 2021. Leveraging In-Network Computing and Programmable Switches for Streaming Analysis of Scientific Data. In *Proceedings of the 2021 IEEE International Conference on Network Softwarization (NetSoft).*

[32] Ganesh C. Sankaran, Krishna M. Sivalingam, and Harsh Gondaliya. 2023. P4 and NetFPGA-Based Secure In-Network Computing Architecture for AI-Enabled Industrial Internet of Things. *IEEE Internet of Things Journal* 10, 4 (2023), 2979–2994.

[33] Naveen Kr Sharma, Antoine Kaufmann, Thomas Anderson, Arvind Krishnamurthy, Jacob Nelson, and Simon Peter. 2017. Evaluating the Power of Flexible Packet Processing for Network Resource Allocation. In *Proceedings of the 2017 USENIX Symposium on Networked Systems Design and Implementation (NSDI).*

[34] Emiliano Sisinni, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. 2018. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics* 14, 11 (2018), 4724–4734.

[35] Tarek Stolz, István Koren, Liam Tirpitz, and Sandra Geisler. 2023. GALOIS: A Hybrid and Platform-Agnostic Stream Processing Architecture. In *Proceedings of the International Workshop on Big Data in Emergent Distributed Environments (BiDEDE).*

[36] Muhammad Tirmazi, Ran Ben Basat, Jiaqi Gao, and Minlan Yu. 2020. Cheetah: Accelerating Database Queries with Switch Pruning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data.*

[37] Liam Tirpitz and Ike Kunze. 2025. Zenodo, Sourcecode - Reducio:v1.0. https://doi.org/10.5281/zenodo.15320003

[38] Martin Unterberg, Marco Becker, Philipp Niemietz, and Thomas Bergs. 2023. Data-Driven Indirect Punch Wear Monitoring in Sheet-Metal Stamping Processes. *Journal of Intelligent Manufacturing* (2023).

[39] Michail Vlachos, Philip Yu, and Vittorio Castelli. 2005. On Periodicity Detection and Structural Periodic Similarity. In *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM).*

[40] Zeyu Wang, Jingao Xu, Xu Wang, Xiangwen Zhuge, Xiaowu He, and Zheng Yang. 2023. Industrial Knee-jerk: In-Network Simultaneous Planning and Control on a TSN Switch. In *Proceedings of the 2023 ACM International Conference on Mobile Systems, Applications, and Services (MobiSys).*

[41] Wenjing Xiao, Yiming Miao, Giancarlo Fortino, Di Wu, Min Chen, and Kai Hwang. 2022. Collaborative Cloud-Edge Service Cognition Framework for DNN Configuration Toward Smart IIoT. *IEEE Transactions on Industrial Informatics* 18, 10 (2022), 7038–7047.

[42] Shen Yin and Okyay Kaynak. 2015. Big Data for Modern Industry: Challenges and Trends. *Proc. IEEE* 103, 2 (2015), 143–146.

[43] Matei Zaharia, Reynold S. Xin, and Patrick Wendell et al. 2016. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* 59, 11 (2016), 56–65.

[44] Xianzhi Zeng and Shuhao Zhang. 2023. A Hardware-Conscious Stateful Stream Compression Framework for IoT Applications (Vision). In *Proceedings of the 17th ACM International Conference on Distributed and Event-Based Systems.*

[45] Steffen Zeuch, Ankit Chaudhary, Bonaventura Del Monte, Haralampos Gavriilidis, Dimitrios Giouroukis, Philipp M. Grulich, Sebastian Breß, Jonas Traub, and Volker Markl. 2020. The NebulaStream Platform: Data and Application Management for the Internet of Things. In *Proceedings of the 2020 Conference on Innovative Data Systems Research (CIDR).*

[46] Changgang Zheng, Zhaoqi Xiong, Thanh T. Bui, Siim Kaupmees, Riyad Bensoussane, Antoine Bernabeu, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2022. IIsy: Practical In-Network Classification. *arXiv:2205.08243* (2022).

[47] Tiago Zonta, Cristiano André da Costa, Rodrigo da Rosa Righi, Miromar José de Lima, Eduardo Silveira da Trindade, and Guann Pyng Li. 2020. Predictive Maintenance in the Industry 4.0: A Systematic Literature Review. *Computers & Industrial Engineering* 150 (2020), 106889.