Towards data-driven additive manufacturing processes

Citation for the original published paper (version of record):
Gulisano, V., Papatriantafilou, M., Chen, Z. et al (2022). Towards data-driven additive manufacturing processes. Middleware 2022 Industrial Track - Proceedings of the 23rd International Middleware Conference Industrial Track, Part of Middleware 2022: 43-49.
http://dx.doi.org/10.1145/3564695.3564778

Towards Data-Driven Additive Manufacturing Processes

Vincenzo Gulisano, Marina Papatriantafilou
Chalmers University of Technology
{vincenzo.gulisano,ptrianta}@chalmers.se

Zhuoer Chen, Eduard Hryha, Lars Nyborg
Centre for Additive Manufacture-Metal (CAM2)
Chalmers University of Technology
{zhuoer.chen,hryha,lars.nyborg}@chalmers.se

ABSTRACT

Additive Manufacturing (AM), or 3D printing, is a potential game-changer in medical and aerospatial sectors, among others. AM enables rapid prototyping (allowing development/manufacturing of advanced components in a matter of days), weight reduction, mass customization, and on-demand manufacturing to reduce inventory costs. At present, though, AM has been showcased in many pilot studies but has not reached broad industrial application. Online monitoring and data-driven decision-making are needed to go beyond existing offline and manual approaches.

We aim at advancing the state-of-the-art by introducing the STRATA framework. While providing APIs tailored to AM printing processes, STRATA leverages common processing paradigms such as stream processing and key-value stores, enabling both scalable analysis and portability. As we show with a real-world use case, STRATA can support online analysis with sub-second latency for custom data pipelines monitoring several processes in parallel.

CCS CONCEPTS

• **Applied computing** → *Computer-aided manufacturing*; • **Computing methodologies** → Machine learning approaches; • **Information systems** → **Online analytical processing engines**.

KEYWORDS

Additive Manufacturing, Powder Bed Fusion - Laser Beam, Stream Processing, Big Data

ACM Reference Format:
Vincenzo Gulisano, Marina Papatriantafilou and Zhuoer Chen, Eduard Hryha, Lars Nyborg. 2022. Towards Data-Driven Additive Manufacturing Processes. In *23rd International Middleware Conference (Middleware '22 Industrial Track), November 7–11, 2022, Quebec, QC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3564695.3564778

1 INTRODUCTION

Additive Manufacturing (AM), or 3D printing, refers to a large family of technologies that can revolutionize industrial processes like the biomedical and aerospace ones. Among other benefits, AM offers rapid prototyping (to study/create new objects in a few days), weight reduction, mass-customization of part geometry, and on-demand manufacturing to reduce inventory cost [5, 26].

Middleware 2022, November 7 – 11, 2022, Québec City, Québec, Canada

ACM ISBN 978-1-4503-9917-3/22/11.
https://doi.org/10.1145/3564695.3564778

A particular type of AM technology, focus of this work, is Powder Bed Fusion - Laser Beam (PBF-LB). PBF-LB creates complex-shaped parts from metal powders (e.g., titanium, nickel, and aluminum alloys) with mechanical properties that are comparable to wrought counterparts. A PBF-LB machine fuses consecutive thin layers of powder particles spread on a flat build platform. Each layer is melted by the laser beam according to the 2D slices of a 3D object. The typical thickness of a powder layer is 20-100 μm, and the laser beam is typically 100 μm in diameter, allowing to manufacture highly intricate features not achievable through e.g., casting.

As PBF-LB matures into industrialization, there are increasing demands on the quality assurance of PBF-LB-produced parts. Quality assurance challenges stem from the various sources of variations that can cause quality issues [4]. The state-of-art quality control of PBF-LB products relies on historical data (e.g., porosity levels, mechanical properties, fatigue strength) collected through experiments for a certain combination of process parameters, material, powder batch, and machine. Such practices are expensive and often not sufficient for ensuring quality, because of process/product-specific factors such as the state of the feedstock powder, the maintenance of the machine and the part's location in the build chamber [4, 10].

Online sensing devices that measure the process signatures during the build with high spatial and temporal resolutions have been developed over the years to provide traceability of individual parts [7]. The large volume of fine-grain data (up to hundreds of GBs per build) is at disposal of AM system experts (or experts, in short) for the identification of quality issues and further optimization of the process. However, to make proper use of such sensing devices on an industrial scale, a data-driven approach is needed to automate the analysis and store relevant data for future references of the products, eventually enabling feedback loop control. Most of the analysis is still manual, batch-based, and run upon completion of a printing process, with experts deciding if a printed piece is to be kept and how to adjust future printing processes based on their expertise and the newly acquired data. Ideally, truly data-driven quality assurance should allow to:

(1) Make timely decisions so that a printing process showing signs of defects is re-configured or terminated as soon as possible, saving energy, material, time, and thus being more sustainable.
(2) Monitor a process by integrating/fusing information from the current process as well as past ones.
(3) Offer experts with complex analysis semantics that scale.

We show that data pipelines that rely on the stream processing paradigm, combined with a key-value store, can address these challenges. Towards the definition of a general framework, we introduce the STRATA prototype and, with a real-world use case, show it can support online analysis with sub-second latency for custom data pipelines monitoring several processes in parallel.

Organization: § 2 introduces preliminaries about the stream processing paradigm, § 3 covers our problem formulation, § 4 overviews STRATA and its API, § 5 presents our use-case and subsequent evaluation (intended both to exemplify how STRATA can support the applications, as well as to provide a first assessment of its performance), § 6 discusses related work, and § 7 concludes the paper.

2 PRELIMINARIES (STREAM PROCESSING)

Stream processing *continuous queries* (or simply queries) are Directed Acyclic Graphs of *operators* that transform unbounded *streams* of *tuples* (see § 5 for a sample query). Tuples have two *attributes*: the metadata carries the timestamp τ and other *sub-attributes*, while the payload carries key-value sub-attributes. The timestamp $t.\tau$ is the *event time* set by the source/operator upon creation of tuple t. A tuple's combined notation is $\langle \tau, \ldots, [k_1{:}v_1, k_2{:}v_2, \ldots] \rangle$.

Queries are run by Stream Processing Engines (SPEs). When a query is deployed, *Sources* forward data to it while *Sinks* deliver its results to users. Acknowledging that SPEs let users define custom operators, they usually provide a common set of *native* operators, including both *stateless* operators – which process tuples one by one – and *stateful* operators – whose outputs depend on multiple input tuples. Common stateless native operators [1, 3, 12] are:

- **Map**: which produces an arbitrary number of output tuples for each input tuple by selecting one or more of the input tuples' sub-attributes, optionally applying functions to them.
- **Filter**: which is used to decide whether a certain tuple should be forwarded or discarded based on a condition.

Common stateful native operators [1, 3, 12] are:

- **Aggregate**: which maintains a *window* of size WS and advance WA of the most recent input tuples and aggregates them (e.g., with functions such as max, min, or sum) possibly defining a set of group-by (GB) sub-attributes to aggregate together only tuples sharing the same value for such sub-attributes. For each GB value, windows cover periods $[\ell WA, \ell WA + WS)$, with $\ell \in \mathbb{N}$.
- **Join**: which defines a left (L) and a right (R) input stream, and produces a tuple combining the sub-attributes of tuples $t_L \in L$ and $t_R \in R$ for each pair $\langle t_L, t_R \rangle$ satisfying a predicate P and so that $|t_L.\tau - t_R.\tau| \leq$ WS. If an optional set of GB sub-attributes is defined, P is checked only for pairs sharing the same GB values.

Note that two key advantages of composing a query by relying on native operators are (1) that such query can seamlessly leverage the APIs SPEs make available to boost performance through parallel, distributed, and elastic execution, and (2) portability, since most SPEs offer similar stateless/stateful operators [14, 15].

3 PROBLEM DEFINITION

PBF-LB printing processes usually involve two parties, the PBF-LB machine and the expert (Figure 1A). The latter submits a printing job, collects the printed parts as well as any available data, and, finally, analyzes the piece and/or data to decide whether to keep or not the parts, also using the newly acquired data/knowledge for future printing jobs. Such a setup has two main downsides. First, the expert becomes aware of defects or other reasons to discard a piece only upon completion of a printing process. Hence, even if signs of such a defect could have been sensed earlier, this inevitably

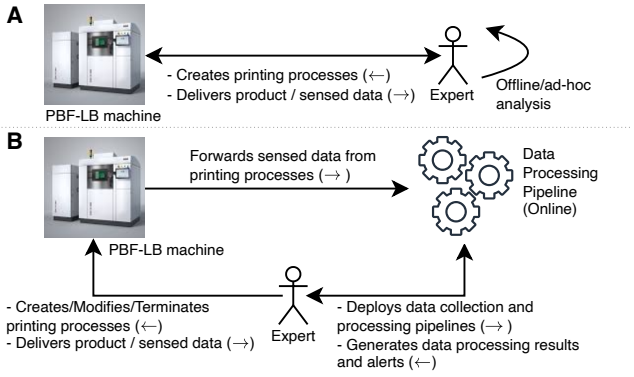

Figure 1: Existing Quality Assurance (A) and Quality Assurance envisioned by data-driven AM manufacturing (B).

results in a waste of expensive resources. Second, this setup leads to experts resorting to manual and non-automated procedures for data analysis and knowledge extraction, with poor knowledge sharing across printing jobs and experts.

Our goal is to promote the shift shown in Figure 1B. In this case, the expert is still responsible for submitting each printing job, but can also submit custom data pipelines. The latter retrieve live data from the PBF-LB machine, analyze it on the fly, and report live information to the user that, manually or relying on dedicated tools/scripts, can thus decide whether to continue, re-adjust, or terminate an ongoing process. For simplicity, from now on we use the term expert to refer both to the user as well as the scripts/tools (s)he uses to coordinate the PBF-LB machine and the data processing layer. The challenges/requirements to realize such a shift include:

(1) **Rich semantics and intermixing of at-rest and streaming data** Experts usually rely on a wide range of analysis steps, from simple filtering to approximation and ML tasks. For some of the complex analysis tasks, information from previous jobs needs to be matched with the one from the running ones (and similarly, live information needs to be maintained and later shared with other jobs). Note that parts of a given data pipeline can be shared by different experts and/or across jobs.
(2) **Low latency analysis** Latency usually expresses the time interval between the output of a result and the time when all the data that led to such a result were made available to the data handling system [8, 24, 28]. The need for low-latency analysis stems from the need to reduce the material, time, and energy wasted from the moment all the data indicating a defect are made available to the data pipeline. Note that in certain cases, as also shown in § 5, there might be strict QoS deadlines indicating the maximum latency tolerated in producing a certain result.
(3) **High throughput analysis** A manufacturing facility can count on many PBF-LB machines, each sensing data at a different time granularity and producing varying data volumes. The proposed architecture should thus support high-throughput analysis in terms of tuples it can ingest per time unit [8, 24, 28].

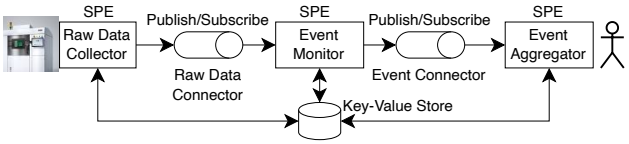

Figure 2: Overview of STRATA's architecture

4 THE STRATA FRAMEWORK

Figure 2 shows STRATA's architecture. In the following, we describe each module, its API, and the implementation details.

Key-Value Store: As discussed in § 3, in-situ monitoring for data-driven PBF-LB processes builds both on live data as well as data-at-rest. To accommodate such a need, this module offers a key/value store that is accessible by all the other modules.

Raw Data Collector: To account for the heterogeneous sensing units of PBF-LB machines, this module defines data-specific collectors (e.g., for Optical Tomography images, see § 5) to gather information/printing parameters of jobs submitted to a machine.

Raw Data Connector: a publish/subscribe module to share raw data from the PBF-LB machine with the next processing module.

Event Monitor: is the module for inspecting each layer individually and detecting relevant events.

Event Connector: a publish/subscribe module that exposes individual events to the following module.

Event Aggregator: this module allows to aggregate relevant events both within and across layers. Across layers, events are automatically grouped by STRATA based on the specimen they refer to. Events produced by this layer are delivered to the expert.

Note that a common practice in PBF-LB processes is to print several specimens within a single job to maximize the usage of the available printing volume within each process. In such a case, layer portions that refer to different specimens could be analyzed in a pipelined/parallel fashion. Such a disjoint analysis could also make sense for a single specimen if some parts of it are more sensitive to defects than others. To account for this possibility, and to support low-latency/high-throughput analysis through parallelism, STRATA's API offers methods to express which parts of a layer/specimen could be processed independently.

The Raw Data Collector and the Event Monitor modules have been designed as separate modules so that multiple event detection methods can be continuously deployed, run (potentially in parallel), and decommissioned. A similar motivation holds for separating the Event Monitor and Event Aggregator modules. Because of these choices, distinct pipelines from one or more users can overlap.

Table 1 presents STRATA's API. Parameters in squared brackets are optional. We refer the readers to § 5 for a complete example.

Implementation. STRATA relies on an SPE for data analysis and a pub/sub infrastructure for data exchange (Figure 2). The chosen SPE is Liebre [18], a lightweight SPE for scale-up servers [14], commonly used for AM in-situ monitoring [7]. Pub/sub connectors run in Apache Kafka [2] while the key-value store runs in RocksDB [27].

When it comes to the APIs implementation, each data collector instantiated via `addSource` runs as a Source within the underlying SPE. All other analysis methods rely on the composition of native operators to aggregate the data and forward it to other analysis tasks or the expert. The choice of relying on compositions of native operators brings two benefits. As mentioned in § 2 and also discussed in [25], it implies that the API methods can be executed in a distributed, parallel, elastic fashion by the underlying SPEs. Also, since all major SPEs offer similar native operators, it means STRATA can be also seamlessly ported to other SPEs.

Module	API Method / Schema of the output stream
Key-Value Store	`store(`k`,`v`)`/`get(`k`,`v`)`
	Used to persist/retrieve data at-rest. Can be invoked by all other API methods.
Raw Data Collector	`addSource(`src, s_{out}`)` $\langle \tau, job, layer, [k_1{:}v_1, k_2{:}v_2, \ldots] \rangle$
	Adds a Source whose resulting stream s_{out} carries tuples with timestamp (τ) and unique identifiers for the current printing job (job) and the layer to which the data refers to ($layer$) in their metadata sub-attributes, and an arbitrary set of sensor-specific key value pairs $k_i{:}v_i$ in their payload.
Event Monitor	`fuse(`$s_{in1}, s_{in2}, s_{out}$`,`$[WS, WA]$`,`$[GB]$`)` $\langle \tau, job, layer, [k_1{:}v_1, k_2{:}v_2, \ldots] \rangle$
	The method assumes s_{in1} and s_{in2} are streams generated by a Source or resulting from an invocation of the `fuse` *method. It then fuses tuples from such streams that have the same job and $layer$ sub-attributes. If parameters Window Size (WS) and Window Advance (WA) are not defined, the method will only fuse tuples that also share the same τ. Otherwise, it will fuse tuples falling in the same windows. The optional parameter GB can specify further sub-attributes used to group tuples into windows. Each output tuple concatenates all the key-value pairs found in the input tuples being fused. The method assumes that, for each set of fused tuples, each key is unique.*
Event Monitor	`partition(`$s_{in}, s_{out}, \mathcal{F}$`)` $\langle \tau, job, layer, specimen, portion, [k_1{:}v_1, k_2{:}v_2, \ldots] \rangle$
	The method assumes s_{in} is generated by a Source or the `fuse` *method. Based on the user-defined function $\mathcal{F}$, each input tuple in s_{in} is transformed in an arbitrary number of output tuples in s_{out}. For each such tuple, the input tuple metadata is copied, and enriched by the $specimen$ and $portion$ sub-attributes, which $\mathcal{F}$ is expected to return. If no* `partition` *function is defined, STRATA assumes each tuple produced by a Source or method* `fuse` *is to be processed as a whole, and sets default values for the $specimen$ and $portion$ sub-attributes.*
Event Monitor	`detectEvent(`$s_{in}, s_{out}, \mathcal{F}$`)` $\langle \tau, job, layer, specimen, portion, [k_1{:}v_1, k_2{:}v_2, \ldots] \rangle$
	The method assumes s_{in} is generated by a Source, method `fuse`*, or method* `partition`*. The user-defined function $\mathcal{F}$ transforms each input tuple in s_{in} in an arbitrary number of output tuples in s_{out} with the given schema.*
Event Aggregator	`correlateEvents(`$s_{in}, s_{out}, L, \mathcal{F}$`)` $\langle \tau, job, layer, specimen, [k_1{:}v_1, k_2{:}v_2, \ldots] \rangle$
	The method assumes s_{in} is generated by method `detectEvent`*. Based on the user-defined function $\mathcal{F}$, the method aggregates all the output tuples generated by method* `detectEvent` *for a certain layer and specimen (i.e., for all the portions of such a specimen) as well as the events of the previous L layers, hence supporting both intra- and inter-layer analysis. The key-value pairs in each output tuple's payload are defined by $\mathcal{F}$.*

Table 1: STRETCH's API

5 USE-CASE AND EVALUATION

Use-case description

In this section, we present a real-world use-case implemented on top of the STRATA framework. Such a use-case is intended both to exemplify the data analysis applications STRATA can support as well as to provide a first assessment of its data analysis performance.

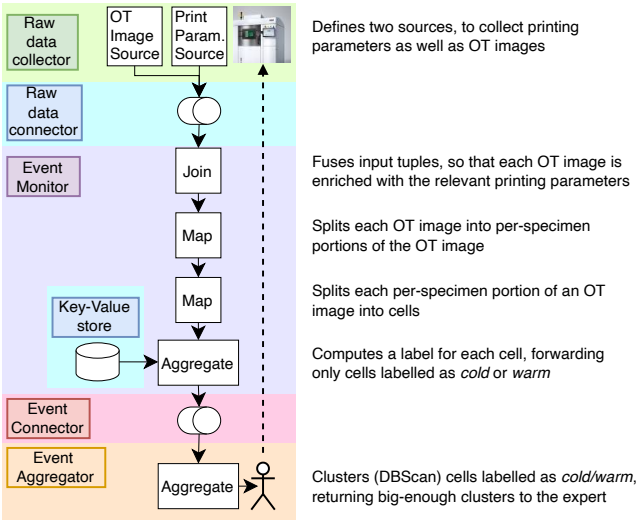

Figure 3: Data pipeline detecting portions of the specimen(s) being printed that have been melted with too-low or too-high thermal energy, known to result in poor material structure.

In this use-case, the expert wants to know if portions of the specimen(s) being printed are melted with too-low or too-high thermal energy since such portions can result in poor material structure. Optical Tomography (OT) is used to produce, for each layer, a long-exposure image that captures the light emissions from the melt pool and solidified material. In the resulting OT images, too-low and too-high thermal energy values are identified based on whether the reported light emanation value is below or above a threshold value, the latter computed based on historical information from previous jobs. Subsequently, the corresponding specimen portions affected by too-low/too-high thermal energy are clustered, within and across layers, and reported when bigger than a certain volume.

Figure 3 provides a simplified view of the different parts of the pipeline associated with this use-case while Algorithm 1 shows a simplified version of the APIs invocations from the use-case. Figure 3 shows the operators deployed in the different modules of STRATA. In the following, we cover the operators for the Raw Data Collector, Event Monitor, and Event Aggregator modules.

Raw Data Collector. The pipeline defines two operators: for retrieving information about the printing jobs submitted at the PBF-LB machine (Alg. 1 L1), and for collecting the OT images generated during each process (Alg. 1 L2). The latter source generates tuples that, besides the timestamp τ and the *job* and *layer* sub-attributes, carry an OT image as a 2D numerical array in which each value indicates the light emanation intensity of a pixel in the image.

Algorithm 1: Simplified listing of the query from Figure 3.

```
1 addSource(new PrintingParameterCollector(),pp)
2 addSource(new OTImageCollector(),OT)
3 fuse(OT,pp,OT&pp)
4 partition(OT&pp,spec,isolateSpecimen())
5 partition(spec,cell,isolateCell())
6 detectEvent(cell,cellLabel,labelCell())
7 correlateEvents(cellLabel,out,L,DBSCAN())
```

Event Monitor. In this layer, the sources' data is fused enriching each OT image with the respective printing parameters (Alg. 1 L3). Since WS, WA, and GB parameters are not specified when invoking `fuse`, the query relies on a Join matching tuples from the 2 streams using sub-attributes τ, *job*, and *layer* for the GB parameter.

The resulting tuples are fed to a Map instantiated by the `partition` method (Alg. 1 L4). For each OT image, a series of smaller images each containing the pixels of one of the specimens being printed is produced by the `isolateSpecimen()` method. The information about which pixels refer to each specimen is found in the key-value sub-attributes generated by the Printing Parameters source.

A second invocation of the `partition` method (Alg. 1 L5) is used to further partition the pixels of each specimen into cells through the `isolateCell()` function. The latter are then fed to the `detectEvent` method (Alg. 1 L6), which classifies each cell as *very cold*, *cold*, *regular*, *warm*, or *very warm* using the `labelCell()` function and forwards a tuple only for cells classified as very cold or very warm. The Aggregate operator instantiated by the `detectEvent` method gets the relevant thresholds from the key-value store.

Event Aggregator. The final portion of the pipeline clusters together the individual events referring to specimen portions that are affected by too-low or too-high thermal energy. Density-based clustering (in particular the DBSCAN, Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise) is used to cluster close-by portions and instantiated by invoking method `correlateEvents` (Alg. 1 L7). The choice of the method is motivated through earlier work in defect detection, where k-means clustering was adopted [29]; DBSCAN is a preferred alternative, both due to the fact that the number of sought clusters is not known in advance, as well as due to its additional capabilities to detect clusters of arbitrary shapes and sizes [6] and be both more accurate and efficient [16, 22, 23, 30]. Parameter L refers to the number of previous layers through which each cluster can expand. In this case, the function passed to method `correlateEvents` also returns an image of the resulting clusters that is later inspected by the expert.

Evaluation setup

We use a higher-end server, mounting an Intel Xeon E5-2637 v4 @ 3.50GHz (4 cores, 8 threads) and 64 GB RAM. The server runs Ubuntu 18.04 and OpenJDK 1.8.0. In this work, we examine the data acquired in a PBF-LB printing process conducted on an EOS M290 machine (an industrial PBF-LB machine) equipped with a state-of-the-art online OT sensor. The OT system takes a long exposure image of the process area with 250×250 mm size for each processed layer using an sCMOS camera. The monitored data are in the form of gray-scale images of 2000×2000 pixels, each of 8Mb in size, with the gray value of each pixel representing the intensity of the melt pool during the PBF-LB process. Each OT image is forwarded by the PBF-LB machine at the completion of the corresponding layer.

Data. The printing process used to evaluate the presented pipeline builds 12 specimens, each of 25 (width) $\times$ 50 (length) $\times$ 23 (height) mm in dimensions. Within each block, three small cylinders are defined to later measure the three-dimensional distribution of process

defects with X-ray Computed Tomography. Along the build height direction, each block is broken down into 23 stacks of 1 mm height. Within each stack, the laser is set to scan at a certain orientation angle to the gas flow, which flows from the back to the front of the machine to remove process by-products such as smoke and spatter particles [17]. The different scanning orientations incur different interactions between the generated spatter and the local gas flow, creating potential sites for defects to appear. Figure 4 shows the OT image of a specimen together with its resulting clustering based on the observed thermal energy levels.

Evaluation metrics

We evaluate STRATA's performance for this use-case in terms of latency and throughput (see § 3). Shown results are based on 5 repetitions of each experiment. Based on the PBF-LB machine being used, there is a gap of approximately 3 seconds between the completion of a layer and the beginning of the next one during which the machine removes the leftover powder and recoats the printing surface for the following one. A *QoS threshold* of 3 seconds is thus assumed for the use-case pipeline outcomes' latency to allow for an online decision about whether to continue, update, or terminate the process before the next layer starts being printed.

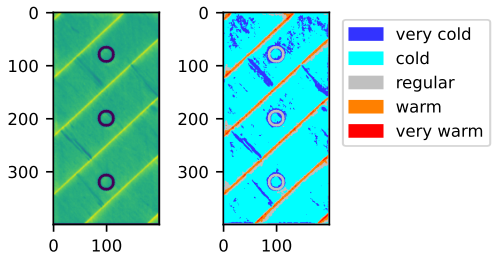

Figure 4: OT image of a specimen together with its resulting clustering based on the observed thermal energy levels.

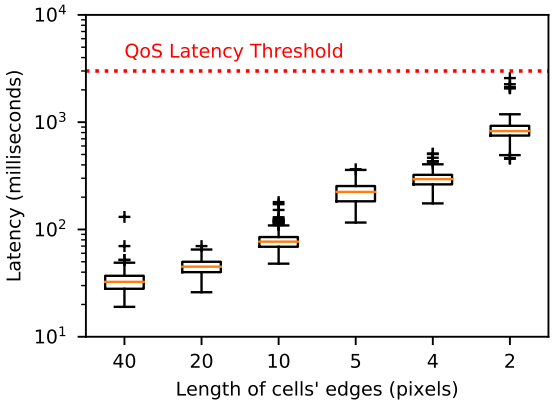

Figure 5: Boxplots of the latency values observed for cells of various sizes monitored and clustered by the query in Alg. 1.

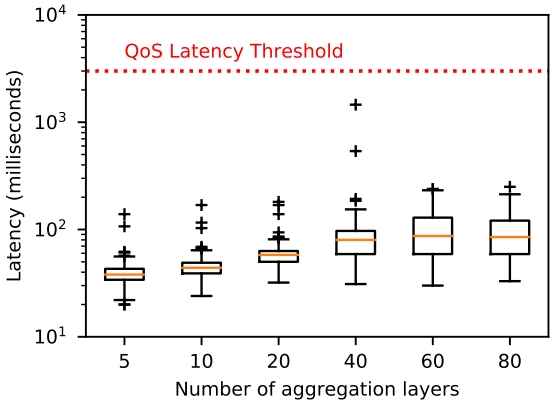

Figure 6: Boxplots of the latency values observed when clustering a different number of layers using the query in Alg. 1.

Evaluation results

In our first experiment, we measure the latency with which up-to-date results are delivered upon the reception of one OT image. To account for different accuracy levels, we variate the length of the cell edge so that method `isolateCell()` (Alg. 1 L5) separates cells with sizes varying from 40x40 to 2x2 pixels (5 to $0.25mm^2$). The results are shown in Figure 5. For each cell size, the observed latency values are shown using a boxplot. As expected, the smaller the area of a cell, the higher the number of cells to be analyzed within and across layers, and the higher the processing latency. In our prototype implementation, STRATA is always able to meet the QoS threshold for all cell sizes, up to the limit case of $2X2$ pixels cells (smaller cells would imply per-pixel analysis).

To further study how parameters settings impact the performance metrics being evaluated, in our second experiment we investigate the effect of changing the number of previous layers clustered together in method `correlateEvents` (parameter L, Alg. 7 L7). As shown in Figure 6, we variate L from 5 layers ($0.2mm$) to 80 layers ($3.2mm$). Also in this case, despite the expected growth trend, all reported latency values are lower than the QoS threshold.

Since live OT images come within a period of minutes (in general, the actual time depends on the shape and number of specimens being printed), we complement our experiments with a third one in which input data is replayed as fast as possible while checking whether the aforementioned latency threshold is respected. Such an experiment can provide an estimate of how fast OT images from historic data can be reprocessed or of the number of jobs that could be processed in parallel (e.g., from various PBF-LB machines).

In this case, we focus on two cell sizes, namely 20x20 and 10x10. The results are shown in Figure 7. The upper part of the figure shows the resulting processing throughput (in terms of thousands of cells processed per second) for an increasing number of OT images fed to the query per second. The bottom part shows the resulting average processing latency. As expected, the throughput initially grows linearly with the number of OT image/s fed to the query while the latency remains low until the query processing capacity is exceeded, the throughput flattens and the latency grows

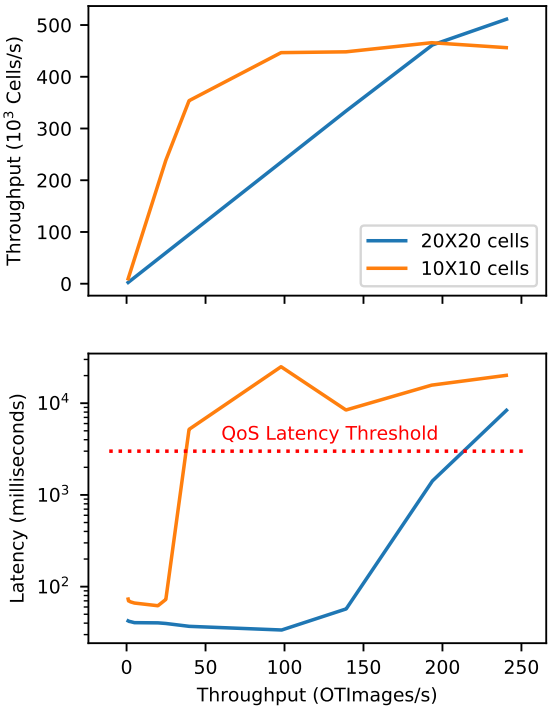

Figure 7: Throughput/latency for various cell sizes and increasing number of OT Images/s fed to the query from Alg. 1.

with a steeper curve. This common pattern [13] is observed for both cell sizes. Since, as aforementioned, smaller cells result in a higher number of cells to analyze, the throughput curve for the 10x10 cells reaches the max value and flattens before that of the 20x20 cells (at approximately one-fourth of the latter, since as expected each 20x20 cell corresponds to 4 10x10 cells). These results indicate our prototype implementation can sustain processing rates of 10s to 100s of OT images/s, thus reprocessing past printing jobs in seconds or processing data from many PBF-LB machines in parallel.

6 RELATED WORK

Quality assurance based on monitoring/event detection in AM depends on aspects like the intended part characteristics, the material properties, the process parameters, and the hardware/software control of the process, among others [19]. Also, as AM processes should be monitored from the scale of particles (~$100\mu m$) to that of entire parts (dm), it is necessary to define the relevant analysis to be conducted effectively and efficiently on monitoring data. Consequently, advanced process monitoring combined with the appropriate implementation of data analytics for quality assurance is a must. Important concerns are the reusability of powder, alloy design for AM, and taking advantage of PBF-LB being an inherently rapid cooling technology, while considering the possible metallurgical constraints and powder-process gas interactions during printing.

Regarding data analysis in AM, [11] comprehensively categorizes monitoring problems and, as also elaborated by [20, 21], lists ML approaches as candidates, acknowledging the need for process awareness of geometries, materials, and flaw types with continuous rather than batch-based data pipelines, emphasizing the need for low-latency/high-throughput in-process monitoring. To the best of our knowledge, STRATA is the first general-purpose framework that, rather than discussing a specific monitoring/quality assurance process, defines a general-purpose and scalable API for intra- and inter-layer data analysis for PBF-LB printing processes.

Regarding ML for streaming data, [9] comprehensively summarizes the state of the art in ACM's core forum on data mining, data science, and analytics. Highlighted open problems are about evolving graph data, from images, text, and other non-structured data sources, including pattern mining in those and associated data structures. As we show, STRATA's API offers general aggregation methods to equip ML operators, as is the case for the DBSCAN clustering method part of our real-world use-case.

7 CONCLUSIONS

We presented STRATA, a framework for scalable, low-latency, and high-throughput data-driven AM with a special focus on PBF-LB processes. To the best of our knowledge, ours is the first contribution that, rather than an ad-hoc solution to analyze a specific feature of an PBF-LB printing process (e.g., a type of defect), proposes a general API for a broad set of intra- and inter-layer data analysis pipelines. With a prototype built on top of state-of-the-art data handling components, STRATA offers an API that internally leverages common APIs of Big Data frameworks (e.g., stateless/stateful stream processing operators) to support efficiency and portability.

In the future, we plan to extend the portfolio of use-cases and related performance studies accounting for common features of PBF-LB processes, such as e.g., the material used as powder, the shape of the object being printed, or the type of monitored defect.

ACKNOWLEDGMENTS

This work has been conducted in the framework of the Centre for Additive Manufacture–Metal (CAM2) supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA). Funding from the VR grant "EPITOME" (2021-05424); Chalmers AoA frameworks Energy and Production, WPs INDEED, and "Scalability, Big Data and AI", resp., and the European Union's Horizon 2020 research and Innovation Programme Additive Manufacturing using Metal Pilot Line (MANUELA) under grant agreement n. 820774 and Demonstration of Infrastructure for Digitalization enabling industrialization of Additive Manufacturing (DiDAM) research project (VINNOVA Project ID 2019-05591) are gratefully acknowledged.

REFERENCES

[1] Apache Flink. 2022. https://flink.apache.org. Accessed:2022-6-27.
[2] Apache Kafka. 2022. https://kafka.apache.org/. Accessed:2022-6-27.
[3] Apache Storm. 2022. http://storm.apache.org. Accessed:2022-6-27.
[4] Zhuoer Chen, Xinhua Wu, and Chris HJ Davies. 2021. Process variation in Laser Powder Bed Fusion of Ti-6Al-4V. *Additive Manufacturing* 41 (2021), 101987.
[5] Brett P Conner, Guha P Manogharan, Ashley N Martof, Lauren M Rodomsky, Caitlyn M Rodomsky, Dakesha C Jordan, and James W Limperos. 2014. Making sense of 3-D printing: Creating a map of additive manufacturing products and services. *Additive Manufacturing* 1 (2014), 64–76.
[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *2nd Conf. on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press, 226–231.

[7] Sarah K Everton, Matthias Hirsch, Petros Stravroulakis, Richard K Leach, and Adam T Clare. 2016. Review of in-situ process monitoring and in-situ metrology for metal additive manufacturing. *Materials & Design* 95 (2016), 431–445.
[8] Xinwei Fu, Talha Ghaffar, James C Davis, and Dongyoon Lee. 2019. Edgewise: A Better Stream Processing Engine for the Edge. In *USENIX Annual Technical Conference (ATC) 19*. USENIX, WA, USA, 929–946.
[9] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. 2019. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter* 21, 2 (2019), 6–22.
[10] Paul R Gradl, Darren C Tinker, John Ivester, Shawn W Skinner, Thomas Teasley, and John L Bili. 2021. Geometric feature reproducibility for laser powder bed fusion (L-PBF) additive manufacturing with Inconel 718. *Additive Manufacturing* 47 (2021), 102305.
[11] Marco Grasso and Bianca Maria Colosimo. 2017. Process defects and in situ monitoring methods in metal powder bed fusion: a review. *Measurement Science and Technology* 28, 4 (2017), 044005.
[12] Vincenzo Gulisano. 2012. *StreamCloud: An Elastic Parallel-Distributed Stream Processing Engine.* Ph. D. Dissertation. Universidad Politécnica de Madrid.
[13] Vincenzo Gulisano, Ricardo Jimenez-Peris, Marta Patino-Martinez, Claudio Soriente, and Patrick Valduriez. 2012. Streamcloud: An elastic and scalable data streaming system. *IEEE Transactions on Parallel and Distributed Systems* 23, 12 (2012), 2351–2365.
[14] Vincenzo Gulisano, Hannaneh Najdataei, Yiannis Nikolakopoulos, Alessandro V. Papadopoulos, Marina Papatriantafilou, and Philippas Tsigas. 2022. STRETCH: Virtual Shared-Nothing Parallelism for Scalable and Elastic Stream Processing. *IEEE Transactions on Parallel and Distributed Systems* (2022), 1–18. https://doi.org/10.1109/TPDS.2022.3181979
[15] Martin Hirzel, Robert Soulé, Scott Schneider, Buğra Gedik, and Robert Grimm. 2014. A catalog of stream processing optimizations. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 1–34.
[16] Amir Keramatian, Vincenzo Gulisano, Marina Papatriantafilou, and Philippas Tsigas. 2022. IP.LSH.DBSCAN: Integrated Parallel Density-Based Clustering through Locality-Sensitive Hashing. In *European Conference on Parallel Processing*. Springer.
[17] Alexander Ladewig, Georg Schlick, Maximilian Fisser, Volker Schulze, and Uwe Glatzel. 2016. Influence of the shielding gas flow on the removal of process by-products in the selective laser melting process. *Additive Manufacturing* 10 (2016), 1–9.
[18] Liebre SPE. 2022. https://github.com/vincenzo-gulisano/Liebre. Accessed:2022-6-27.
[19] Mahesh Mani, Shaw Feng, Brandon Lane, Alkan Donmez, Shawn Moylan, and Ronnie Fesperman. 2015. Measurement science needs for real-time control of additive manufacturing powder bed fusion processes. (2015).
[20] Mohammad Montazeri, Abdalla R Nassar, Alexander J Dunbar, and Prahalada Rao. 2020. In-process monitoring of porosity in additive manufacturing using optical emission spectroscopy. *IISE Transactions* 52, 5 (2020), 500–515.
[21] William Mycroft, Mordechai Katzman, Samuel Tammas-Williams, Everth Hernandez-Nava, George Panoutsos, Iain Todd, and Visakan Kadirkamanathan. 2020. A data-driven approach for predicting printability in metal additive manufacturing processes. *Journal of Intelligent Manufacturing* 31, 7 (2020), 1769–1781.
[22] Hannaneh Najdataei, Vincenzo Gulisano, Philippas Tsigas, and Marina Papatriantafilou. 2022. pi-Lisco: parallel and incremental stream-based point-cloud clustering. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 460–469.
[23] Hannaneh Najdataei, Yiannis Nikolakopoulos, Vincenzo Gulisano, and Marina Papatriantafilou. 2018. Continuous and parallel lidar point-cloud clustering. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 671–684.
[24] Dimitris Palyvos-Giannas, Vincenzo Gulisano, and Marina Papatriantafilou. 2019. Haren: A Framework for Ad-Hoc Thread Scheduling Policies for Data Streaming Applications. In *Proceedings of the 13th ACM International Conference on Distributed and Event-Based Systems (DEBS '19)*. ACM, Darmstadt, Germany, 19–30. https://doi.org/10.1145/3328905.3329505
[25] Dimitris Palyvos-Giannas, Bastian Havers, Marina Papatriantafilou, and Vincenzo Gulisano. 2020. Ananke: a streaming framework for live forward provenance. *Proceedings of the VLDB Endowment* 14, 3 (Nov. 2020), 391–403. https://doi.org/10.14778/3430915.3430928
[26] C. Pauzon, B. Hoppe, T. Pichler, S. Dubiez-Le Goff, P. Forêt, T. Nguyen, and E. Hryha. 2021. Reduction of incandescent spatter with helium addition to the process gas during laser powder bed fusion of Ti-6Al-4V. *CIRP Journal of Manufacturing Science and Technology* 35 (2021), 371–378. https://doi.org/10.1016/j.cirpj.2021.07.004
[27] RocksDB. 2022. https://http://rocksdb.org/. Accessed:2022-6-27.
[28] Anshu Shukla, Shilpa Chaturvedi, and Yogesh Simmhan. 2017. RIoTBench: An IoT Benchmark for Distributed Stream Processing Systems. *Concurrency and Computation: Practice and Experience* 29, 21 (2017), e4257. https://doi.org/10.1002/cpe.4257 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4257
[29] Robert Snell, Sam Tammas-Williams, Lova Chechik, Alistair Lyle, Everth Hernández-Nava, Charlotte Boig, George Panoutsos, and Iain Todd. 2020. Methods for rapid pore classification in metal additive manufacturing. *Jom* 72, 1 (2020), 101–109.
[30] Yiqiu Wang, Yan Gu, and Julian Shun. 2020. Theoretically-Efficient and Practical Parallel DBSCAN. In *2020 SIGMOD Int. Conf. on Management of Data*. ACM, 2555–2571.