Processing Big Data in Motion: Core Components and System Architectures with Applications to the Maritime Domain

Nikos Giatrakos, Antonios Deligiannakis, Konstantina Bereta, Marios Vodas, Dimitris Zissis, Elias Alevizos, Charilaos Akasiadis, and Alexander Artikis

Abstract Rapidly extracting business value out of Big Data that stream in corporate data centres requires continuous analysis of massive, high-speed data while they are still in motion. So challenging a goal entails that analytics should be performed in memory with a single pass over these data. In this chapter, we outline the challenges of Big streaming Data analysis for deriving real-time, online answers to application inquiries. We review approaches, architectures and systems designed to address these challenges and report on our own progress within the scope of the EU H2020 project INFORE. We showcase INFORE into a real-world use case from the maritime domain and further discuss future research and development directions.

Keywords Big Data · Cross-platform optimisation · Data streams · Data synopses · Online machine learning · Complex event forecasting · Maritime situation awareness

N. Giatrakos (✉) · A. Deligiannakis
Athena Research Center, Athens, Greece
e-mail: ngiatrakos@athenarc.gr; adeli@athenarc.gr

K. Bereta · M. Vodas · D. Zissis
MarineTraffic, Athens, Greece
e-mail: konstantina.bereta@marinetraffic.com; marios.vodas@marinetraffic.com; dzissis@marinetraffic.com

E. Alevizos · C. Akasiadis · A. Artikis
NCSR Demokritos, Institute of Informatics and Telecommunications, Athens, Greece
e-mail: alevizos.elias@iit.demokritos.gr; cakasiadis@iit.demokritos.gr; a.artikis@iit.demokritos.gr

E. Curry et al. (eds.), *Technologies and Applications for Big Data Value*,
https://doi.org/10.1007/978-3-030-78307-5_22

1 Challenges of Big Streaming Data

Today, organisations and businesses have the ability to collect, store and analyse as much data as they need, exploiting powerful computing machines in corporate data centres or the cloud. To extract value out of the raw Big Data that are accumulated, application workflows are designed and executed over these infrastructures engaging simpler (such as grouping and aggregations) or more complex (data mining and machine learning) analytics tasks. These tasks may involve data at rest or data in motion.

Data at rest are historic data stored on disks, getting retrieved and loaded for processing by some analytics workflow. Analytics tasks participating in such a workflow perform computations on massive amounts of data, lasting for hours or days. They finally deliver useful outcomes. Using a running example from the maritime domain, historic vessel position data are used to extract Patterns-of-Life (PoL) information. These are essentially collections of geometries representing normal navigational routes of vessels in various sea areas [78], used as the basis for judging anomalies.

Data in motion involve Big streaming Data which are unbounded, high-speed streams of data that need to get continuously analysed in an online, real-time fashion. Storing the data in permanent storage is not an option, since the I/O latency would prevent the real-time delivery of the analytics output. Application workflows get a single look on the streaming data tuples, which are kept in memory for a short period of time and are soon stored or discarded to process newly received data tuples.

At an increasing rate, numerous industrial and scientific institutions face such business requirements for real-time, online analytics so as to derive actionable items and timely support decision-making procedures. For instance, in the maritime domain, to pinpoint potentially illegal activities at sea [54] and allow the authorities to timely act, position streams of thousands of vessels need to be analysed online.

To handle the volume and velocity of Big streaming Data, Big Data platforms such as Apache Flink [2], Spark [5] or toolkits like Akka [1] have been designed to facilitate scaling-out, i.e., parallelising, the computation of streaming analytics tasks horizontally to a number of Virtual Machines (VM) available in corporate computer clusters or the cloud. Thus, multiple VMs simultaneously execute analytics on portions of the streaming data undertaking part of the processing load, and therefore throughput, i.e., number of tuples being processed per time unit, is increased. This aids in transforming raw data in motion to useful results delivered in real time. Big Data platforms also offer APIs with basic stream transformation operators such as filter, join, attribute selection, among others, to program and execute streaming workflows. However useful these facilities may be, they only focus on a narrow part of the challenges that business workflows need to encounter in streaming settings.

First, Big Data platforms currently provide none or suboptimal support for advanced streaming analytics tasks engaging Machine Learning (ML) or Data Mining (DM) operators. The major dedicated ML/DM APIs they provide, such as

MLlib [5] or FlinkML [2], do not focus on parallel implementations of streaming algorithms.

Second, Big Data platforms by design focus only on horizontal scalability as described above, while there are two additional types of scalability that are of essence in streaming settings. Vertical scalability, i.e., scaling the computation with the number of processed streams, is also a necessity. Federated scalability, i.e., scaling the computation one step further out, to settings composed of multiple, potentially geo-dispersed computer clusters, is another type of required scalability. For instance, in maritime applications, vessels transmit their positions to satellite or ground-based receivers. These data can be ingested in proximate data centres and communicated only on demand upon executing global workflows, i.e., involving the entire set of monitored vessels, over the fragmented set of streams.

Third, Big Data technologies are significantly fragmented. Delivering advanced analytics requires optimising the execution of workflows over a variety of Big Data platforms and tools located at a number of potentially geo-dispersed clusters or clouds [30, 34, 36]. In such cases, the challenge is to automate the selection of an optimal setup prescribing (a) which network cluster will execute each analytics operator, (b) which Big Data platform available at this cluster, and (c) how to distribute the computing resources of that cluster to the operators that are assigned to it.

AQ1

Connecting the above challenges to a real-world setting from the maritime domain, on a typical day at MarineTraffic,[1] 100GB vessel position data and approximately 750M messages (volume, velocity—horizontal scalability) are processed online. This data is complemented by other data sources such as satellite image data of tens of TBs [54]. At any given time, MarineTraffic is tracking over 200K vessels in real-time (vertical scalability) over a network of approximately 5K stations (federated scalability). Additionally, the analysis engages a variety of Big Data platforms including Apache Spark, Flink, Akka and Kafka (details in Sect. 3).

Finally, applications often require an additional level of abstraction on the derived analytics results. Consider a vessel that slows down, then makes a U-turn and then starts speeding up. Such a behaviour may occur in case of an imminent piracy event where a vessel attempts to run away from pirates. The application is not interested in knowing the absolute speed, heading or direction information in the raw stream. Instead, it wants to receive continuous reports directly on a series of detected, *simple* events (`slowing down`, `U-turn`, `speeding`) and the higher level, *complex* `piracy` event or to be able to forecast such events [79]. Complex Event Processing (CEP) and Forecasting (CEF) encompass the ability to query for business rules (patterns) that match incoming streams on the basis of their content and some topological ordering on them (CEP) or to forecast the appearance of patterns (CEF) [31, 33, 35].

In this chapter, we discuss core system components required to tackle these challenges and the state of the art in their internal architectures. We further describe

[1] https://www.marinetraffic.com.

how we advance the state of the art within the scope of the EU H2020 project INFORE. Finally, we showcase the INFORE approach into a real-world use case from the maritime domain. We, however, stress that INFORE applies to any application domain, and we refer the interested reader to [34] for more application scenarios.

This chapter relates to the technical priorities (a) Data Management, (b) Data Processing Architectures and (c) Data Analytics of the European Big Data Value Strategic Research & Innovation Agenda [77]. It addresses the horizontal concerns Cloud, HPC and Sensor/Actuator infrastructure of the BDV Technical Reference Model and the vertical concern of Big Data Types and Semantics (Structured data, Time series data, Geospatial data). Moreover, the chapter relates to (a) Knowledge and Learning, (b) Reasoning and Decision Making, (c) Action and Interaction and (d) Systems, Hardware, Methods and Tools, cross-sectorial technology enablers of the AI, Data and Robotics Strategic Research, Innovation and Deployment Agenda [76].

2 Core Components and System Architectures

2.1 The Case for Data Synopses

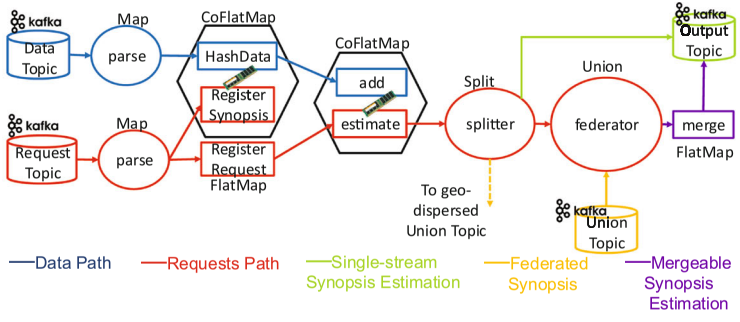
Motivation There is a wide consensus in the stream processing community [25, 26, 32] that approximate but rapid answers to analytics tasks, more often than not, suffice. For instance, detecting a group of approximately 50 highly similar vessel trajectories with sub-second latency is more important than knowing minutes later that the group actually composes 55 such streams with a similarity value accurate to the last decimal. In the latter case, some vessels may have been engaged in a collision. Data synopses techniques such as samples, histograms and sketches constitute a powerful arsenal of data summarisation tools useful across the challenges discussed in the introduction of this chapter. Approximate, with tunable quality guarantees, synopses operators including, but not limited to [25, 26, 32, 46], cardinality (FM Sketches), frequency moment (CountMin, AMS Sketches, Sampling), correlation (Fourier Transforms, Locality Sensitive Hashing [37]), set membership (Bloom Filters) or quantile (GK Quantile) estimation, can replace respective exact operators in application workflows to enable or enhance all three types of required scalability as well as to reduce memory utilisation. More precisely, data summaries leave only a footprint of the stream in memory and they also enhance horizontal scalability since not only is the processing load distributed to a number of available VMs, but also it is shed by letting each VM operate on compact data summaries. Moreover, synopses enable federated scalability since only summaries, instead of the full (set of) streams, can be communicated when needed. Finally, synopses provide vertical scalability by enabling locality-aware hashing [37, 38, 46].

Related Work and State of the Art From a research viewpoint, there is a large number of related works on data synopsis techniques. Such prominent techniques are reviewed in [25, 26, 32] and have been implemented into real-world synopses libraries, such as Yahoo!DataSketch [9], Stream-lib [8], SnappyData [57] and Proteus [7]. Yahoo!DataSketch [9] and Stream-lib [8] are libraries of stochastic streaming algorithms and summarisation techniques, correspondingly, but implementations are detached from parallelisation and distributed execution aspects over streaming Big Data platforms. Apache Spark provides utilities for data synopsis via sampling operators, CountMin sketches and Bloom Filters. Moreover, SnappyData's [57] stream processing is based on Spark and its synopses engine can serve approximate, simple sum, count and average queries. Similarly, Proteus [7] extends Flink with data summarisation utilities. Spark utilities, SnappyData and Proteus combine the potential of data summarisation with horizontal scalability, i.e., parallel processing over Big Data platforms, by providing libraries of parallel versions of data synopsis techniques. However, they neither handle all types of required scalability nor cross Big Data platform execution scenarios.

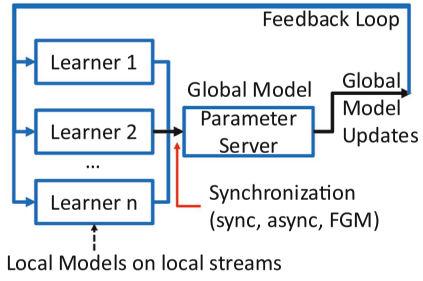
AQ2

INFORE Contribution In the scope of the INFORE project, we have developed a Synopses Data Engine (SDE) [46] that advances the state of the art by tackling all three types of the required scalability and also accounting for sharing synopses common to various running workflows and for cross-platform execution. INFORE SDE goes far beyond the implementation of a library of data summarisation techniques. Instead, it also implements an entire component with its own internal architecture, employing a Synopses-as-a-Service (SDEaaS) paradigm. That is, the SDE is a constantly running service (job) in one or more clusters (federated scalability) that can accept on-the-fly requests for start maintaining, updating and querying a parallel synopsis built on a single high-speed stream (e.g. vessel) of massive data proportions (horizontal scalability) or on a collection of a large number of streams (vertical scalability). The SDEaaS is customisable to specific application needs by allowing dynamic loading of code for new synopses operators at runtime, with zero downtime for the workflows that it serves.

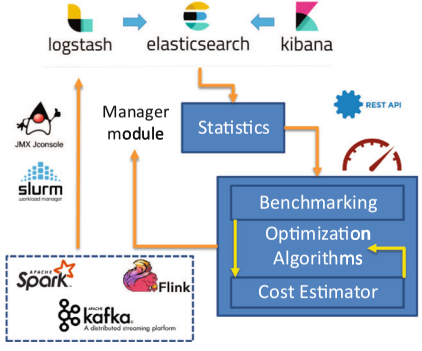
The architecture of INFORE SDEaaS [46] is illustrated in Fig. 1a. INFORE's SDEaaS proof-of-concept implementation is based on Apache Kafka and Flink. Nevertheless, the design is generic enough to remain equally applicable to other Big Data platforms. For instance, an equally plausible alternative would be to implement the whole SDE in Kafka leveraging the Kafka Streams API. Nonetheless, Kafka Streams is simply a client library for developing micro-services, lacking a master node for global cluster management and coordination. Following Fig. 1a, when a request for maintaining a new synopsis is issued, it reaches the `RegisterRequest` and `RegisterSynopsis` `FlatMaps` which produce keys for workers (i.e., VM resources) which will handle this synopsis. Each of this pair of `FlatMaps` uses these keys for a different purpose. `RegisterRequest` uses the keys to direct queries to responsible workers, while `RegisterSynopsis` uses the keys to update the synopses on new data arrivals (blue-coloured path). In particular, when a new streaming data tuple is ingested, the `HashData` `FlatMap` looks up the



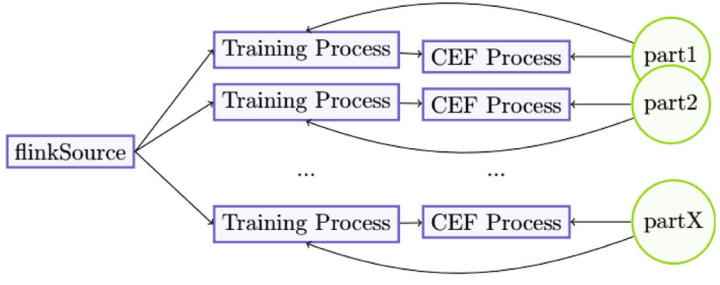
(a) SDEaaS Component (Condensed View).



(b) Parameter Server View.



(c) Optimizer Component.



(d) CEF Component.

Fig. 1 Internal architecture of key INFORE components

keys of `RegisterSynopsis` to see to which workers the tuple should be directed to update the synopsis. This update is performed by the `add FlatMap` in the blue-coloured path. The rest of the operators in Fig. 1a are used for merging partial synopses results [11] maintained across workers or even across geo-distributed computer clusters. Please refer to [46] for further details. In Sect. 3.2.3, we analyse the functionality of a domain-specific synopsis building samples of vessel positions.

2.2 Distributed Online Machine Learning and Data Mining

Motivation As discussed in Sect. 1, ML/DM APIs such as Spark's MLlib [5] or FlinkML [2] are focused on analysing data at rest. Therefore, advanced analytics tasks on data in motion call for filling the gap of a stream processing-oriented ML/DM module. ML and DM algorithms that can meet the challenges discussed in the introduction of this chapter are those that (1) are online, i.e., restricting themselves on a single pass over the data instead of requiring multiple passes, and (2) can run in a distributed fashion, i.e., they are parallelisable and thus the load can be distributed to parallel learners and parallel predictors across a number of VMs so as to provide the primitives for horizontal scalability over Big Data platforms and computer clusters. There exists a variety of algorithms that satisfy these preliminary requirements in diverse ML/DM categories, including [18, 34, 69] classification (such as (Multiclass) Passive Aggressive Classifiers, Online Support Vector Machines, Hoeffding Trees, Random Forests), clustering (BIRCH, Online k-Means, StreamKM++) and regression (Passive Aggressive Regressor, Online Ridge Regression, Polynomial Regression) tasks. These algorithms are designed or can be adapted to get executed in an online, distributed setting. The primary focus, then, is not on the algorithms themselves, but on the architecture an ML/DM module should be built upon, so that various algorithms can be incorporated and also allow for vertical scalability, federated scalability and cross-platform execution with reduced memory utilisation.

Related Work and State of the Art Towards this direction, the two most prominent approaches and modules that exist in the literature are StreamDM [19] and Apache SAMOA [48]. StreamDM is a library of ML/DM algorithms designed to be easily extensible with new algorithms, but dedicated to run on top of the Spark Streaming API [5]. Thus, it does not cover cross-platform execution scenarios, also lacking provisions for vertical and federated scalability. The only framework with a clear commitment to the cross-platform execution goals is Apache SAMOA. SAMOA is portable between Apache Flink, Storm [6] and Samza [4]. When it comes to its model of computation, the architecture of SAMOA follows the Agent-based pattern. In other words, an algorithm is a set of distributed processors that communicate with streams of messages. Little more is provided, which is intentional [48], claiming that a more structured model of computation reduces the applicability of the framework.

The state of the art in distributed ML and DM architectures is the Parameter Server (PS) distributed model [51] as illustrated in Fig. 1b, where a set of distributed learners receive portions of the training streams and extract local models in parallel. The local models are from time to time synchronised to extract a global model at the PS side. The global model is then communicated back to learners via a feedback loop (Fig. 1b). Consider for instance a set of learners each handling a subset of vessel streams within the scope of a vessel type classification task. The learners coordinate with the PS sending their locally trained classification models, while the PS responds back with an up-to-date global model. The PS paradigm enhances horizontal and federated scalability via the option of an asynchronous (besides synchronous) synchronisation policy to reduce the effect of stragglers and bandwidth consumption, respectively. In the synchronous policy, learners are communicating with the PS in predefined rounds/batches, while in the asynchronous case each learner decides individually as to when it should send updates to the PS. Performance-wise, the synchronous policy does not encourage enhanced horizontal scalability because when many learners are used, the total utilisation is usually low, should only few stragglers exist. The asynchronous one is the policy of choice in large-scale ML; the processing speed is much higher when many learners are used and the training is more scalable.

AQ3

The PS paradigm has been criticised for limited training speed due to potential network congestion at the PS side and for severely getting affected by low-speed links between the learners and the PS. Under these claims, a number of decentralised ML/DM architectures have evolved which employ a more peer-to-peer alike structure, where the training rationale is based on gossiping [42, 70]. The drawback of these approaches, though, is that it is unclear how the continuously updated, but decentralised, global model can be directly deployed for real-time inference purposes. This is because knowing the network node holding the updated global model at any given time requires extra communication. Hence, in case we want to train and simultaneously deploy the updated global ML/DM models at runtime, such a decentralised architecture does not seem to mitigate low-speed issues but moves the problem to the prediction, instead of the training, stage.

INFORE Contribution In the scope of the INFORE project, we follow a PS distributed model [51]. As is the case with the SDEaaS described in the previous section, INFORE's ML/DM module includes provisions for cross-platform execution scenarios by receiving input and output streams in JSON formatted Kafka messages. Moreover, the communication between learners and the PS is performed using a lightweight middleware where a generic API for PS and learner (bidirectional) communication is provided. In that, learners can be implemented over any Big Data platform and run in any cluster, while still being able to participate in the common ML/DM task. Besides learners, INFORE's ML/DM module includes a separate pipeline of parallel predictors that can communicate with the PS in order to receive up-to-date global models continuously extracted during the training process and directly deploy them for inference purposes.

INFORE's ML/DM module accounts for vertical and boosts horizontal scalability as well. This is achieved by using INFORE's SDEaaS to partition streams to learners or to allow learners to operate on compact stream summaries, correspondingly. Remarkably, to effectively encounter congestions or low-speed links and also allow to easily and effectively deploy/update the developed models, instead of resorting to decentralised approaches [42, 70], we develop our own synchronisation policy termed FGM [67] (Fig. 1b) that improves the employed PS paradigm. The new synchronisation protocol strengthens horizontal (within a cluster) and federated scalability by bridging the gap between synchronous and asynchronous communication. Instead of having learners communicating in predefined rounds/batches (synchronous) or when each one is updated (asynchronous), FGM requires communication only when a concept drift (i.e., the global model has significantly changed based on some criterion) is likely to have occurred. This is determined based on conditions each learner can individually examine.

2.3 *Distributed and Online CEF*

Motivation Big Data analytics tools mine data views to extract patterns conveying insights into what has happened, and then apply those patterns to make sense of the fresh data that stream in. This only permits to react upon the detection of such patterns, which is often inadequate. In order to allow for proactive decision-making, predictive analytics tools that allow to forecast future events of interest are required. Consider, for instance, the ability to forecast and proactively respond to hazardous events, such as vessel collisions or groundings, in the maritime domain. The ability to forecast, as early as possible, a good approximation to the outcome of a time-consuming and resource-demanding computational task allows to quickly identify possible outcomes and save valuable reaction time, effort and computational resources. Diverse application domains possess different characteristics. For example, monitoring of moving entities has a strong geospatial component, whereas in stock data analysis this component is minimal. Domain-specific solutions (e.g. trajectory prediction for moving objects) cannot thus be universally applied. We need a more general Complex Event Forecasting (CEF) framework.

Related Work and State of the Art Time-series forecasting is an area with some similarities to CEF, with a significant history of contributions [56]. However, it is not possible to directly apply techniques from time-series forecasting to CEF. Time-series forecasting typically focuses on streams of (mostly) real-valued variables and the goal is to forecast relatively simple patterns. On the contrary, in CEF we are also interested in categorical values, related through complex patterns and involving multiple variables. Another related field is that of prediction of discrete sequences over finite alphabets and is closely related to the field of compression, as any compression algorithm can be used for prediction and vice versa [17, 20, 24, 63, 64, 73].

The main problem with these approaches is that they focus exclusively on next symbol prediction, i.e., they try to forecast the next symbol(s) in a stream/string of discrete symbols. This is a serious limitation for CEF. An additional limitation is that they work on single-variable discrete sequences of symbols, whereas CEF systems consume streams of events, i.e., streams of tuples with multiple variables, both numerical and categorical. Forecasting methods have also appeared in the field of temporal pattern mining [22, 50, 71, 75]. A common assumption in these methods is that patterns are usually defined either as association rules [13] or as frequent episodes [53]. From the perspective of CEF, the disadvantage of these methods is that they usually target simple patterns, defined either as strictly sequential or as sets of input events. Moreover, the input stream is composed of symbols from a finite alphabet, as is the case with the compression methods mentioned previously.

INFORE Contribution In a nutshell, the current, state-of-the-art solutions for forecasting, even when they are domain-independent, are not suitable for the kind of challenges that INFORE attempts to address. In INFORE, the streaming input can be constantly matched against a set of event patterns, i.e. arbitrarily complex combinations of time-stamped pieces of information. An event pattern can either be fully matched against the streaming data, in which case events are detected, or partially matched, in which case events are forecast with various degrees of certainty. The latter usually stems from stochastic models of future behaviour, embedded into the event processing loop, which project into the future the sequence of events that resulted to a partial event pattern match, to estimate the likelihood of a full match, i.e. the actual occurrence of a particular complex event.

Given that INFORE's input consists of a multitude of data streams, interesting events may correlate sub-events across a large number of different streams, with different attributes and different time granularities. For instance, in the maritime domain relevant streams may originate from position signals of thousands of vessels which may be fused with satellite image data [54] or even acoustic signals [40]. It is necessary to allow for a highly expressive event pattern specification language, capable of capturing complex relations between events. Moreover, the actual patterns of what constitutes an interesting event are often not known in advance, and even if they are, event patterns need to be frequently updated to cope with the drifting nature of streaming data. Not only do we need an expressive formalism in order to capture complex events in streams of data, but we also need to do so in a distributed and online manner.

Towards this direction, the CEF module of INFORE uses a highly expressive, declarative event pattern specification formalism, which combines logic, probability theory and automata theory. This formalism has a number of key advantages:

- It is capable of expressing arbitrarily complex relations and constraints between events. We are thus not limited to simple sequential patterns applied to streams with only numerical or symbolic values.
- It can be used for event forecasting and offering support for robust temporal reasoning. By converting a pattern into an automaton, we can then use historical

data to construct a probabilistic description of the automaton's behaviour and thus to estimate at any point in time its expected future behaviour.

- It offers direct connections to machine learning techniques for refining event patterns, or learning them from scratch, via tools and methods from the field of grammatical inference. In cases where we only have some historical data and some labels, we must find a way to automatically learn the interesting patterns. This is also the case when there is concept drift in the streaming data and the patterns with which we started may eventually become stale. It is therefore important to be able to infer the patterns in the data in an online manner.

INFORE's CEF module is built on top of Apache Kafka and Flink and has the ability to handle highly complex patterns in an online manner, constantly updating its probabilistic models. Figure 1d shows one possible scheme (pattern-based) for structuring multiple parallel CEF pipelines. As shown in the figure, each such pipeline processes a different CEF query [33, 35]. It is composed of a training process, which estimates the probabilities of a future event to occur, as well as a CEF process that utilises these probabilities to actually forecast complex events. Finally, one implementation detail is that each pipeline also receives a subset of the patterns (part1 to partX in Fig. 1d). The role of these loops is similar to the feedback loop of Fig. 1b. Remarkably, the CEF module can also act as a CEP one since it can not only predict but also detect occurred events of interest [14].

2.4 Geo-distributed Cross-Platform Optimisation

Motivation All the aforementioned advanced stream processing techniques and technologies will only serve their goal if they are properly used. Consider, for instance, that we perfectly tune the execution of a synopsis, ML/DM or CEF operator in a specific cluster, but we assign the execution of the downstream operator of a broader workflow to a distant cluster. The execution speed up achieved for one operator may be diminished by network latency of long network paths. Therefore, developing algorithms for optimising the execution of streaming workflows (a) over a network of many clusters located in various geographic areas, (b) across a number of Big Data platforms available in each cluster and (c) simultaneously elastically devoting VMs and resources (CPU, memory, etc.) is a prerequisite to efficiently deliver in practice real-time analytics. Within a cluster, common optimisation objectives include throughput maximisation, execution latency and memory usage minimisation, while in multi-cluster settings communication cost, bandwidth consumption and network latency are also accounted for. Quality-of-Service (QoS) and computer cluster (CPU, memory, storage) capacity constraints also apply to these objectives.

Related Work and State of the Art There are a number of works that assign the execution of operators targeting at optimising network-related metrics, such as communication cost and network latency, while executing global analytics

workflows across a number of networked machines or computer clusters. The seminal work of SBON [59] seeks to optimise a quantity similar to network usage ($dataRate \times latency$), but with a squared latency, across multi-hop paths followed by communicated data. An important limitation in SBON is that by using such a blended metric, the optimisation process cannot support constrained optimisation per metric (communication cost or latency). Due to that, also other related techniques [49, 59, 62] which employ blended metrics cannot incorporate resource or QoS constraints while determining operators' assignment to clusters. Although some [49, 62] claim to support latency constraints, this comes after having determined where an operator will be executed. Finally, the approach of Geode [72] purely focuses on minimising bandwidth consumption in the presence of regulatory constraints, but it does not account for network latency.

A series of works aim at optimising the execution of analytics operators within a single computer cluster. Such works focus on optimal assignment of operators to VMs such that high performance (mainly, in terms of throughput) and load balancing among VMs is achieved; subject to multiple function, resource and QoS constraints. Related works mainly provide optimisations on load assignment and distribution, load shedding, resource provisioning and scheduling policies inside the cluster. In Medusa [16], Borealis [10], Flux [68] and Nexus [23], the focus is to primarily balance the load, choose appropriate ways to partition data streams across a number of machines and minimise the usage of available resources (CPU cycles, bandwidth, memory, etc.) while maintaining high performance.

Another category of techniques examines the optimisation of network-wide analytics, simultaneously scaling-out the computation of an operator to the VMs of the cluster that undertakes its execution. JetStream [61] trades-off network bandwidth minimisation with timely query answer and correctness, but while exploring the cluster at which an operator will be executed, it restricts itself to the MapReduce rationale (i.e. the operator is executed at the cluster where data rests), nearest site of relevant data presence or a central location. Iridium [60], basically targeting optimisation of analytics over data at rest, assumes control over where relevant data are transferred and moves these data around clusters to optimise query response latency. SQPR [45] and [21] propose more generic frameworks for the constraint-aware optimal execution of global workflows across clusters, and they also optimise resources devoted to each operator execution at each cluster. However, [21, 45] do not account for cross-platform optimisation in the presence of different Big Data technologies.

Systems such as Rheem [12], Ires [27], BigDawg [28] and Musketeer [39] are designed towards cross-platform execution of workflows, but they can only optimise the processing of data at rest,[2] instead of data in motion. Furthermore, only Rheem accounts for network-related optimisation parameters such as communication cost.

[2] BigDawg supports stream processing over S-Store and Rheem supports JavaStreams, but no alternatives are included to allow for optimising across different streaming platforms.

INFORE Contribution The INFORE Optimiser is the first complete solution for streaming operators [30, 34]. INFORE's Optimiser is not simply the only one which can simultaneously instruct the streaming Big Data platform, cluster and computing resources for each analytics operator, but also it does so for a wide variety of diverse operator classes including (1) synopses, (2) ML/DM, (3) CEF and (4) stream transformations. INFORE's Optimiser incorporates the richest set of optimisation criteria related to throughput, network and computational latency, communication cost, memory consumption and accuracy of SDE operators, and it also accounts for constraints per metric, fostering the notion of Pareto optimality [30, 34].

The internals of INFORE Optimiser are illustrated in Fig. 1c. We use a statistics collector to derive performance measurements from each executed workflow. Statistics are collected via JMX or Slurm[3] and are ingested in an ELK stack[4] while monitoring jobs. A Benchmarking submodule automates the acquisition of performance metrics for SDE, OMLDM and CEF/CEP operators run in different Big Data platforms. The Benchmarking submodule utilises statistics and builds performance (cost) models. Cost models are derived via a Bayesian Optimisation approach inspired by CherryPick [15]. The cost models are utilised by the optimisation algorithms [30, 34] to prescribe preferable physical execution plans.

3 Real-Life Application to a Maritime Use Case

3.1 Background on Maritime Situation Awareness (MSA)

According to the US National Concept of Operations for Maritime Domain Awareness,[5] "Global Maritime Intelligence is the product of legacy, as well as changing intelligence capabilities, policies and operational relationships used to *integrate all available data*, information, and intelligence in order to identify, locate, and track potential *maritime threats*. Global MSA results from the *persistent monitoring* of maritime activities in such a way that *trends and anomalies* can be identified".

Maritime reporting systems are distinguished into two broad categories: *cooperative* and *non-cooperative*. An example of a *cooperative* maritime reporting system is the Automatic Identification System (AIS) [43]. All commercial vessels above 300 gross tonnage are obliged to bear AIS transponders. AIS forms the basis of a lot of MSA applications, such as the MarineTraffic vessel tracking platform. Other cooperative, but not public, maritime reporting systems are the Long Range Identification and Tracking system (LRIT) [44], as well as the Vessel Monitoring

[3] https://docs.oracle.com/javase/tutorial/jmx/overview/, https://slurm.schedmd.com/.

[4] https://www.elastic.co/what-is/elk-stack.

[5] https://web.archive.org/web/20111004213300/http://www.gmsa.gov/twiki/bin/view/Main/MDAConOps.

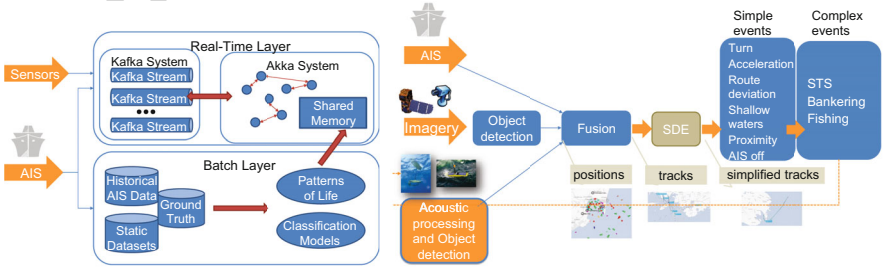
System (VMS) [29] for fishing vessels. Radar on-board or ashore installations can be used as maritime surveillance systems, such as the ones installed by default in a vessel's bridge, as well as in ports. Thermal cameras and satellite imagery can also be used as additional monitoring systems for vessels. Due to the time elapsed between the actual image acquisition from a satellite and its availability on the satellite repository that can be several hours, satellite imagery data do not offer real-time snapshots of the maritime domain but can be used combined with other sources such as AIS to "fill in the gaps" of AIS coverage (e.g., identify the whereabouts of a vessel while its transponder was switched off).

Global and *continuous* monitoring of the maritime domain as well as the identification of trends and anomalies require to address the challenges pointed out throughout this chapter as well as the following generic Big Data challenges described in the scope of the maritime domain:

- *Volume*, the number of available surveillance systems and sensors increases.
- *Velocity*, applications rely on continuous monitoring (e.g., vessel tracking) and need to process high velocity streaming data in real time.
- *Variety*, data from heterogeneous surveillance systems should be combined.
- *Veracity*, most of the maritime data sources are heavily prone to noise requiring data cleaning and analysis tasks to filter out unnecessary or invalid information.
- *Value*, as the availability of more sources of maritime data as well as the advanced Big Data processing, ML and AI technologies that are now available can help to maximise the derived knowledge that can be inferred from maritime data.

3.2 Building Blocks of MSA Workflows in the Big Data Era

Figure 2b shows an example of a generic workflow, implemented in the Maritime Use Case of the INFORE project, for MSA purposes. Different applications may include a subset of operators of Fig. 2b or implement different steps. In the



(a)Anomaly Detection at MarineTraffic. (b)MSA workflow for maritime applications.

Fig. 2 MSA infrastructure and workflow

following, we describe the functionality of the workflow operators of Fig. 2b which serve as the building blocks of modern MSA applications.

3.2.1 Maritime Data Sources

The kinds of data sources that are provided as input in a typical MSA application (Fig. 2b) are the following:

- Vessel positions. Data about vessel positions derive from vessel reporting systems, the most popular of which is AIS. AIS forms the ground of a wide variety of MSA applications. AIS relies on VHF communication: Vessels send AIS messages that contain dynamic information (e.g., information about the current voyage, such as vessel position, speed, heading, etc.) as well as static information (e.g., vessel identifier, dimensions, etc.). For real-time applications, positional data arrive in streaming fashion to the data consumers.
- Data from other sensors. Some applications do not rely only on one source of information. For example, AIS data can be combined with acoustic data, thermal camera data and satellite data. Vessel detection algorithms are applied on this data to extract the positions of vessels. For example, AI techniques are applied on satellite imagery to extract the vessel positions which is important in the cases when a vessel is out of AIS coverage [54].
- Other datasets describing assets and activities in the maritime domain. These are datasets that describe ports, harbours, lighthouses, the boundaries of areas of interest, bathymetry datasets (e.g., for shallow waters estimation), datasets containing vessel schedules, weather data, etc. These datasets are often combined with other data (e.g., vessel positions) in order to enrich the information displayed to the end-users (e.g., the different layers of the MarineTraffic Live Map).

Kafka [3] is used at the data ingestion layer, as a fast, scalable and fault-tolerant messaging system for large data (at rest or in motion) portions.

3.2.2 Maritime Data Fusion

Data from multiple sources besides AIS, such as radars and cameras, are available in real time though in order to be used in MSA modules they must be fused together with AIS and create a unified map. This essentially translates to a need for tracking algorithms that can monitor moving objects globally and in real time using overlapping detections from multiple sensors. The Fusion operator in Fig. 2b is a custom operator with distributed implementations in order to achieve this goal. Trackers are comprised of three main components [65, 66]: (a) a method for the assignment of detections to tracks, (b) the prediction of a target's movement and (c) the architecture of the tracker that coordinates how the detections are processed.

A detection arriving to the tracker can be assigned to a track using three strategies, and each tracker implementation is based on one of them. The first way is to simply choose the track that is closest to the detection, which has the lowest computational complexity but it is not accurate in cases where two objects move very close to each other. The second method focuses on improving the accuracy in cases where a detection is close to multiple tracks by deferring the final assignment until more detections arrive, thus making a more informed decision but at the cost of significantly increasing the complexity and decreasing the responsiveness (i.e., real-time challenge). The third approach stands between the two methods and allows that a detection is assigned to multiple tracks as soon as it arrives, thus increasing the accuracy satisfactorily without increasing complexity.

Each moving object is characterised by certain physical parameters and constraints according to which several kinematic models can predict its movement under different conditions. A simple option is to choose one model, such as constant velocity that assumes the object maintains the last speed, but this affects the accuracy when an object manoeuvres. A better option is to use multiple models, such as constant turn and acceleration, at the same time so that the tracker is able to successfully detect a manoeuvring target.

3.2.3 SDE Operator For Trajectory Simplification

The plethora of incoming data from multiple overlapping sources poses a challenge for data processing workflows. A data synopsis technique with which this challenge can be tackled is trajectory simplification, i.e., reducing the amount of data (positions) so that the computational effort required is reduced as well. The ideal goal is to keep only those positions that are adequate in order to recreate the trajectory with minimal losses in the accuracy of the data processing workflow.

For that, we use INFORE's SDEaaS (Sect. 2.1) which includes an application-specific synopses, namely STSampler. The STSampler scheme resembles the concept of threshold-guided sampling in [58] but executes the sampling process in a more simplistic, yet effective in practice, way. More precisely, the sampling process is executed in a per stream fashion, i.e., for the currently monitored trajectory of each vessel separately. The core concept is that if the velocity and the direction of the movement of the vessel do not change significantly, the corresponding AIS message is not sampled. The last two reported trajectory positions are cached in the `add FlatMap` of Fig. 1a. When an AIS message holding information about the current status of the vessel streams in via `HashData`, the `add FlatMap` computes the change in the velocity between the lastly cached and the new AIS report, i.e., $\Delta_{vel} = |vel(prev) - vel(now)|$, and compares this value to a velocity threshold T_{vel}. Using the previously cached points, the vector describing the lastly reported direction of the vessel $dir(prev)$ is computed, while using the last cached and the newly reported positions we also compute $dir(now)$. Then, we compare $\Delta_{dir} = |dir(prev) - dir(now)|$ against a direction threshold T_{dir}. If at least one of these deltas does not exceed the corresponding threshold, the newly received

AIS message is not included in the sample by the `add FlatMap`. This holds, provided that a couple of additional spatiotemporal constraints are satisfied: (a) the time difference between the newly received AIS message and the last one that was included in the sample does not exceed a given time interval threshold T_{tdiff} and (b) the distance among the most recently sampled and the current position of the vessel does not surpass a distance threshold T_{dist}. SDEaaS is implemented in Flink instead of Kafka, for the reasons explained in Sect. 2.1.

3.2.4 Complex Maritime Event Processing

A very important module of the modern MSA applications is the Maritime Event Detection module. This is essentially a CEP module tailored to the maritime domain. For now, our analysis concentrates on distributed and online CEP, i.e., detecting complex events, while future work will also exploit the potential of CEF (Sect. 2.3). A description of some of the most common vessel events that can occur in the maritime domain is provided below:

- *Turn*: A vessel turns to a different direction.
- *Acceleration*: A vessel accelerates.
- *Route Deviation*: The course of a vessel deviates from "common" routes.
- *Shallow waters*: A vessel navigates in shallow waters.
- *Proximity*: A vessel is in close distance to another vessel.
- *Out of coverage*. A vessel is out of coverage with respect to one or more vessel monitoring systems such as AIS [47].

The events described above are simple events, i.e., they can be computed without depending on other events. Complex events, on the other hand, are events composed from other events. Below we provide examples of complex events:

- *Ship-to-ship*: Transfer of cargo between vessels.
- *Bunkering*: One vessel provides fuel to another vessel.
- *Tugging*: A smaller vessel (a tug) is tugging another vessel.
- *Piloting*: A smaller vessel (pilot vessel) approaches a bigger vessel so that the pilot of the vessel boards the bigger vessel in order to help it navigate into a port where special local conditions apply.
- *Fishing*: A vessel is engaged in fishing activities.

For distributed processing of streaming data in the CEP context, the Akka framework is used [1]. Akka adopts an Actor-based architecture based on message-passing communication, and it is preferred due to the fact that it is more customisable than Spark and Flink. Each Actor, run in parallel instances, is responsible for detecting a simple or complex event as those described above (Fig. 2a and b).

3.2.5 ML-Based Anomaly Detection

The ML algorithms that are relevant to the MSA workflow relate to Deep Neural Network techniques for classifying vessels according to their type (such as cargo, fishing vessel) [54]. Moreover, we are investigating ML-based techniques such as Random Forests for classifying vessel trajectories and recognise simple or complex events in them. This effort is also aided by advanced ML-based operators we have developed to extract the common routes followed by the majority of vessels for every voyage, defined as a pair of origin and destination ports [78]. At the moment, these ML tasks are performed in an offline fashion mostly using Spark's MLlib [5], which we also use to estimate sea-port area regions in [55]. The outcomes of this process performed at the batch layer of Fig. 2a can then be used as added value knowledge to the event detection or the Fusion operator of Fig. 2b. Our ongoing work focuses on incorporating INFORE's module (Sect. 2.2) to materialise ML/DM analytics in an online, real-time fashion, where possible (see restrictions on satellite images in Sect. 3.1).

3.2.6 MSA Workflow Optimisation

Across the workflow of Fig. 2b, the INFORE Optimiser is responsible for prescribing the parallelisation Degree, and the provisioned resources for the maintained trajectory synopses (Sect. 3.2.3) determine the computer cluster and the number of Akka Actors devoted to MSA-related CEP tasks (Sect. 3.2.4). The Optimiser can also do the same for ML-based anomaly detection tasks (Sect. 3.2.5). An initial workflow execution plan can be re-optimised and adjusted at runtime to adapt (e.g., by increasing/decreasing the number of Akka Actors) to changing data stream distributions or to a load of concurrently executed maritime workflows. Moreover, the ongoing integration of the INFORE CEF module will allow the Optimiser to prescribe the most efficient implementation among Akka (Sect. 3.2.4) and Flink (Sect. 2.3) options for event processing tasks.

4 Future Research and Development Directions

Future research and development directions mainly lie in the synergies of ML/DM, Synopses, CEP/CEF and optimisation technologies discussed in this chapter.

Resource-Constrained ML/DM Resource-Constrained ML/DM goes beyond data processing over distributed, but computationally powerful infrastructures such as computer clusters or the cloud. The objective in resource constrained ML/DM is to bridge the gap between the very high computation and communication demands of state-of-the-art ML algorithms, such as Deep Neural Nets and Kernel Support Vector Machines, and the goal of running such algorithms (e.g. various classifiers)

on a large, heavily distributed system of resource-constrained devices. Resource-constrained devices, such as sensors, pose limitations to the power supply, memory, computation and communication capacity. Fast and efficient classifiers requiring reduced power and memory should be developed, along with novel algorithms to train, apply and update the classifiers. Synergies between synopses and distributed, online ML/DM utilities are critical for such tasks.

Optimisation over Internet of Things (IoT) Platforms Optimisation over Internet of Things (IoT) platforms, since existing optimisation frameworks, should be extended to allow for planning the execution of workflows taking into consideration the whole set IoT features including: (a) resource scarcity, (b) hardware heterogeneity, (c) data heterogeneity, (d) dynamic population of devices, (e) mobility of devices, (f) security aspects over massively distributed architectures, and (g) resilience and accuracy of analytics in the presence of device failures.

CEP/CEF-Oriented Synopses CEP/CEF-Oriented Synopses techniques tailored for CEP/CEF are becoming a necessity. The work in [41] was the first to point out that load shedding schemes tailored for CEP are missing and that shedding the load in CEP significantly differentiates itself from doing so in conventional streaming settings. A few more approaches emerged since then [52, 74], but still little attention has been paid on the distributed environments and the mergeability properties of such techniques [11].

Acknowledgments This work has received funding from the EU Horizon 2020 research and innovation program INFORE under grant agreement No. 825070.

References

1. Akka v. 2.5.32. https://akka.io/. Accessed 15 September 2020.
2. Apache Flink v. 1.12. https://flink.apache.org/. Accessed 15 September 2020.
3. Apache Kafka v. 2.3. https://kafka.apache.org/. Accessed 15 September 2020.
4. Apache Samza v. 1.5.1. http://samza.apache.org/. Accessed 15 September 2020.
5. Apache Spark v. 2.4.4. https://spark.apache.org/. Accessed 15 September 2020.
6. Apache Storm v. 2.1. https://storm.apache.org/. Accessed 15 September 2020.
7. Proteus project. https://github.com/proteus-h2020/. Accessed 15 September 2020.
8. Stream-lib. https://github.com/addthis/stream-lib/. Accessed 15 September 2020.
9. Yahoo datasketch. https://datasketches.github.io/ Accessed 15 September 2020.
10. Abadi, D. J., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J. H., Lindner, W., Maskey, A. S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., Zdonik, S. (2005). The design of the borealis stream processing engine. In *CIDR*.
11. Agarwal, P. K., Cormode, G., Huang, Z., Phillips, J. M., Wei, Z., & Yi, K. (2013). Mergeable summaries. *ACM Transactions on Database Systems*, *38*(4), 26:1–26:28.
12. Agrawal, D., Chawla, S., Rojas, B., et al. (2018). RHEEM: enabling cross-platform data processing - may the big data be with you! *Proceedings of the VLDB Endowment*, *11*(11), 1414.
13. Agrawal, R., Imielinski, T., & Swami, A. N. (1993). Mining association rules between sets of items in large databases. In *SIGMOD*.

14. Alevizos, E., Artikis, A., Paliouras, G. (2018). Wayeb: a tool for complex event forecasting. In *LPAR*.
15. Alipourfard, O., Liu, H., Chen, J., Venkataraman, S., Yu, M., & Zhang, M. (2017). Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics. In *NSDI*.
16. Balazinska, M., Balakrishnan, H., Stonebraker, M. (2004). Contract-based load management in federated distributed systems. In *NSDI*.
17. Begleiter, R., El-Yaniv, R., & Yona, G. (2004). On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, *22*, 385–421.
18. Benczúr, A., Kocsis, L., & Pálovics, R. (2018). Online machine learning in big data streams. arXiv: 1802.05872.
19. Bifet, A., Maniu, S., Qian, J., Tian, G., He, C., & Fan, W. (2015). Streamdm: Advanced data mining in spark streaming. In *ICDMW*.
20. Bühlmann, P., Wyner, A. J., et al. (1999). Variable length Markov chains. *The Annals of Statistics*, *27*(2), 480–513.
21. Cardellini, V., Grassi, V., Lo Presti, F., & Nardelli, M. (2016). Optimal operator placement for distributed stream processing applications. In *DEBS*.
22. Cho, C., Wu, Y., Yen, S., Zheng, Y., & Chen, A. L. P. (2011). On-line rule matching for event prediction. *VLDB Journal*, *20*(3), 303–334.
23. Cipriani, N., Eissele, M., Brodt, A., Grossmann, M., & Mitschang, B. (2009). Nexusds: a flexible and extensible middleware for distributed stream processing. In *IDEAS*.
24. Cleary, J. G., & Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, *32*(4), 396–402.
25. Cormode, G., Garofalakis, M. N., Haas, P. J., & Jermaine, C. (2012). Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends Databases*, *4*(1–3), 1–294.
26. Cormode, G., & Yi, K. (2020). *Small summaries for big data*. Cambridge University Press. https://doi.org/10.1017/9781108769938
27. Doka, K., Papailiou, N., Tsoumakos, D., Mantas, C., & Koziris, N. (2015). Ires: Intelligent, multi-engine resource scheduler for big data analytics workflows. In *SIGMOD*.
28. Elmore, A. J., Duggan, J., Stonebraker, M., Balazinska, M., Çetintemel, U., Gadepally, V., Heer, J., Howe, B., Kepner, J., Kraska, T., Madden, S., Maier, D., Mattson, T. G., Papadopoulos, S., Parkhurst, J., Tatbul, N., Vartak, M., Zdonik, S. (2015). A demonstration of the bigdawg polystore system. *Proceedings of the VLDB Endowment*, *8*(12), 1908.
29. FAO: VMS for fishery vessels. http://www.fao.org/fishery/topic/18103/en. Accessed 15 May 2019.
30. Flouris, I., Giatrakos, N., Deligiannakis, A., & Garofalakis, M. N. (2020). Network-wide complex event processing over geographically distributed data sources. *Information Systems*, *88*, 101442.
31. Flouris, I., Giatrakos, N., Garofalakis, M. N., & Deligiannakis, A. (2015). Issues in complex event processing systems. In *IEEE TrustCom/BigDataSE/ISPA* (Vol. 2)
32. Garofalakis, M. N., Gehrke, J., & Rastogi, R. (Eds.). (2016). *Data stream management - processing high-speed data streams*. Data-centric systems and applications. Springer. https://doi.org/10.1007/978-3-540-28608-0
33. Giatrakos, N., Alevizos, E., Artikis, A., Deligiannakis, A., & Garofalakis, M. N. (2020). Complex event recognition in the big data era: a survey. *VLDB Journal*, *29*(1), 313–352.
34. Giatrakos, N., Arnu, D., Bitsakis, T., Deligiannakis, A., Garofalakis, M. N., Klinkenberg, R., Konidaris, A., Kontaxakis, A., Kotidis, Y., Samoladas, V., Simitsis, A., Stamatakis, G., Temme, F., Torok, M., Yaqub, E., Montagud, A., Ponce, M., Arndt, H., Burkard, S. (2020). Infore: Interactive cross-platform analytics for everyone. In *CIKM*.
35. Giatrakos, N., Artikis, A., Deligiannakis, A., & Garofalakis, M. N. (2017). Complex event recognition in the big data era. *Proceedings of the VLDB Endowment*, *10*(12), 1996.
36. Giatrakos, N., Artikis, A., Deligiannakis, A., & Garofalakis, M. N. (2019). Uncertainty-aware event analytics over distributed settings. In *DEBS*.

37. Giatrakos, N., Deligiannakis, A., Garofalakis, M. N., & Kotidis, Y. (2020). Omnibus outlier detection in sensor networks using windowed locality sensitive hashing. *Future Generation Computer Systems*, *110*, 587–609.
38. Giatrakos, N., Kotidis, Y., & Deligiannakis, A. (2010). PAO: power-efficient attribution of outliers in wireless sensor networks. In *DMSN*. https://doi.org/10.1145/1858158.1858168
39. Gog, I., Schwarzkopf, M., Crooks, N., Grosvenor, M. P., Clement, A., & Hand, S. (2015). Musketeer: all for one, one for all in data processing systems. In *EuroSys*.
40. Goldhahn, R., Braca, P., Ferri, G., Munafo, A., & Lepage, K. (2014). Adaptive bayesian behaviors for AUV surveillance networks. In *UAC*.
41. He, Y., Barman, S., & Naughton, J. F. (2014). On load shedding in complex event processing. In *ICDT*.
42. Hegedüs, I., Danner, G., & Jelasity, M. (2019). Gossip learning as a decentralized alternative to federated learning. In *DAIS*.
43. IMO. (2017). Technical characteristics for an automatic identification system using time division multiple access in the VHF maritime mobile frequency band. Technical report, ITU. https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1371-5-201402-I!!PDF-E.pdf
44. IMO. (2018). Long-range identification and tracking system. Technical report, IMO. http://www.imo.org/en/OurWork/Safety/Navigation/Documents/LRIT/1259-Rev-7.pdf
45. Kalyvianaki, E., Wiesemann, W., Vu, Q. H., Kuhn, D., & Pietzuch, P. (2011). Sqpr: Stream query planning with reuse. In *ICDE*.
46. Kontaxakis, A., Giatrakos, N., & Deligiannakis, A. (2020). A synopses data engine for interactive extreme-scale analytics. In *CIKM*.
47. Kontopoulos, I., Chatzikokolakis, K., Zissis, D., Tserpes, K., & Spiliopoulos, G. (2020). Real-time maritime anomaly detection: detecting intentional AIS switch-off. *International Journal of Big Data Intelligence*, *7*(2), 85–96.
48. Kourtellis, N., Morales, G. D. F., & Bifet, A. (2018). Large-scale learning from data streams with apache SAMOA. CoRR abs/1805.11477. http://arxiv.org/abs/1805.11477
49. Kumar, V., Cooper, B. F., & Schwan, K. (2005). Distributed stream management using utility-driven self-adaptive middleware. In *ICAC*.
50. Laxman, S., Tankasali, V., & White, R. W. (2008). Stream prediction using a generative model based on frequent episodes in event sequences. In *KDD*.
51. Li, M., Andersen, D., Park, J. W., et al. (2014). Scaling distributed machine learning with the parameter server. In *OSDI*.
52. Li, Z., & Ge, T. (2016). History is a mirror to the future: Best-effort approximate complex event matching with insufficient resources. *Proceedings of the VLDB Endowment*, *10*(4), 85–96.
53. Mannila, H., Toivonen, H., & Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, *1*(3), 259–289.
54. Milios, A., Bereta, K., Chatzikokolakis, K., Zissis, D., & Matwin, S. (2019). Automatic fusion of satellite imagery and AIS data for vessel detection. In *FUSION*.
55. Millefiori, L. M., Zissis, D., Cazzanti, L., & Arcieri, G. (2016). A distributed approach to estimating sea port operational regions from lots of AIS data. In *IEEE BigData*.
56. Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons.
57. Mozafari, B. (2019). Snappydata. In *Encyclopedia of Big Data Technologies*. Springer.
58. Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., & Theodoridis, Y. (2017). Online event recognition from moving vessel trajectories. *GeoInformatica*, *21*(2), 389–427.
59. Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., & Seltzer, M. (2006). Network-aware operator placement for stream-processing systems. In *ICDE*.
60. Pu, Q., Ananthanarayanan, G., Bodik, P., Kandula, S., Akella, A., Bahl, P., & Stoica, I. (2015). Low latency geo-distributed data analytics. In *SIGCOMM*.
61. Rabkin, A., Arye, M., Sen, S., Pai, V. S., & Freedman, M. J. (2014). Aggregation and degradation in jetstream: Streaming analytics in the wide area. In *NSDI*.
62. Rizou, S. (2013). *Concepts and algorithms for efficient distributed processing of data streams*. University of Stuttgart. https://doi.org/10.18419/opus-3209

AQ4

63. Ron, D., Singer, Y., & Tishby, N. (1993). The power of amnesia. In *NIPS*.
64. Ron, D., Singer, Y., & Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, *25*(2–3), 117–149.
65. Rong Li, X., & Jilkov, V. P. (2003). Survey of maneuvering target tracking. part i. dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, *39*(4), 1333–1364.
66. Rong Li, X., & Jilkov, V. P. (2005). Survey of maneuvering target tracking. part v. multiple-model methods. *IEEE Transactions on Aerospace and Electronic Systems*, 41(4), 1255–1321.
67. Samoladas, V., & Garofalakis, M. N. (2019). Functional geometric monitoring for distributed streams. In *EDBT*.
68. Shah, M. A., Hellerstein, J. M., Chandrasekaran, S., & Franklin, M. J. (2003). Flux: An adaptive partitioning operator for continuous query systems. In *ICDE*.
69. Silva, J., Faria, E., Barros, R., Hruschka, E., Carvalho, A., Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys*, *46*(1), 1–31.
70. Tang, H., Lian, X., Yan, M., Zhang, C., Liu, J. (2018). D^2: Decentralized training over decentralized data. In *ICML*.
71. Vilalta, R., & Ma, S. (2002). Predicting rare events in temporal domains. In *ICDM*.
72. Vulimiri, A., Curino, C., Godfrey, P. B., Jungblut, T., Padhye, J., Varghese, G. (2015). Global analytics in the face of bandwidth and regulatory constraints. In *NSDI*.
73. Willems, F. M. J., Shtarkov, Y. M., & Tjalkens, T. J. (1995). The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, *41*(3), 653–664.
74. Zhao, B., Hung, N. Q. V., & Weidlich, M. (2020). Load shedding for complex event processing: Input-based and state-based techniques. In: *ICDE*.
75. Zhou, C., Cule, B., & Goethals, B. (2015). A pattern based predictor for event streams. *Expert Systems with Applications*, *42*(23), 9294–9306.
76. Zillner, S., Bisset, D., Milano, M., Curry, E., Robles, A.G., Hahn, T., Irgens, M., Lafrenz, R., Liepert, B., O'Sullivan, B., & Smeulders, A. (Eds.). (2020). Strategic research, innovation and deployment agenda - AI, data and robotics partnership. Third Release. Brussels. BDVA, euRobotics, ELLIS, EurAI and CLAIRE (September 2020).
77. Zillner, S., Curry, E., Metzger, A., Auer, S., & Seidl, R. (Eds.). (2017). *European big data value strategic research & innovation agenda*. Big Data Value Association.
78. Zissis, D., Chatzikokolakis, K., Spiliopoulos, G., & Vodas, M. (2020). A distributed spatial method for modeling maritime routes. *IEEE Access*, 8, 47556–47568.
79. Zissis, D., Chatzikokolakis, K., Vodas, M., Spiliopoulos, G., & Bereta, K.: A data driven approach to maritime anomaly detection. In *MSAW* (2019).

AUTHOR QUERIES

AQ1. Please check edit to the sentence “In such cases, the challenge is to...” is fine.
AQ2. Please check the edit to the sentence “However, they neither handle...” is fine.
AQ3. Please check the sentence “Performance-wise, the synchronous policy...” for clarity.
AQ4. Please note that the references in the list have been arranged alphabetically and corresponding citations have been changed accordingly. Please check.