6

A Framework for Flexible and Programmable Data Analytics in Industrial Environments

Nikos Kefalakis[1], Aikaterini Roukounaki[1], John Soldatos[1] and Mauro Isaja[2]

[1]Kifisias 44 Ave., Marousi, GR15125, Greece
[2]Engineering Ingegneria Informatica SpA, Italy
E-mail: jsol@ait.gr; arou@ait.gr; nkef@ait.gr; mauro.isaja@eng.it

This chapter presents a dynamic and programmable distributed data analytics solution for industrial environments. The solution includes an edge analytics engine for analytics close to the field and in line with the edge computing paradigm. Each edge analytics engine instance is flexible and dynamically configurable based on an Analytics Manifest (AM). It is also based on distributed ledger technologies for configuring analytics tasks that span multiple edge nodes and instances of the edge analytics engine. In particular, it leverages ledger services for synchronizing and combining various AMs in factory wide analytics tasks. Based on these mechanisms, the presented distributed data analytics infrastructure is therefore flexible, configurable, dynamic and resilient. Moreover, it is open source and provides Open APIs (Application Programming Interfaces) that enable access to its functionalities. These features make it unique and valuable for vendors and integrators of industrial automation solutions.

6.1 Introduction

A large number of digital automation applications in modern shopfloors collect and process large amounts of digital data as a means of identifying the status of machines and devices (e.g., a machine's condition or failure mode)

or the context of industrial processes (e.g., possible defects in an entire production process), including relevant events [1]. This context is accordingly used to support decision making, including decisions that drive automation and control operations on the shopfloor [2] such as the configuration of a production line or the operational mode of a machine. Therefore, data analytics operations are an integral element of most digital automation platforms [3], which is usually integrated within automation and simulation functionalities.

In this context, the automation platform that has been developed in the scope of the FAR-EDGE project includes also distributed data analytics functionalities. In particular, the FAR-EDGE platform offers functionalities in three distinct, yet complementary domains, namely Automation, Analytics and Simulation [4]. The Analytics domain provides the means for collecting, filtering and processing large volumes of data from the manufacturing shopfloor towards calculating indicators associated with manufacturing performance and automation. Analytics functions are offered by a Distributed Data Analytics (DDA) infrastructure, which enables the definition, configuration and execution of analytics functions at two different levels, namely:

- Local Level Analytics, i.e. at the edge of a FAR-EDGE deployment. These comprise typically analytics functions that are executed close to the field and have local/edge scope, e.g. they collect and process data streams from a part of a factory such as data streams associated with a station within the factory. Local Level Analytics in FAR-EDGE are configured and executed by means of an Edge Analytics Engine (EAE), which runs within an Edge Gateway (EG) and is a core part of the DDA.
- Global Level Analytics, i.e. concerning the factory as a whole and spanning instances of local level analytics. In FAR-EDGE, global level analytics combine information from multiple Edge Gateways (EGs) and instances of the Edge Analytics Engine. They can be configured and executed through an Open API. Global Level analytics are supported by the ledge and the cloud infrastructures of the FAR-EDGE platform.

The distinction between edge/local and global/cloud analytics is very common in the case of Big Data analytics systems (e.g. [5–7]). Moreover, there are different frameworks that can handle streaming analytics at the edge of the network, which is a foundation for edge analytics. The FAR-EDGE DDA infrastructure goes beyond the state of the art of these Big Data systems through employing novel techniques for the flexible configuration of edge analytics and the synchronization of multiple edge analytics deployments. In particular, the FAR-EDGE DDA includes an infrastructure for registering data sources from the plantfloor, as well as for dynamically discovering them.

Moreover, it includes a modular framework for the deployment of analytics functionalities based on a set of (reusable) processing libraries. The latter can be classified in three main types of data processing functions, which enable the pre-processing of data streams (i.e. pre-processing functions), their data analysis (i.e. analytics functions) and ultimately the storage of the analytics results (i.e. storage functions). In FAR-EDGE, edge analytics tasks are described as combinations of various instances of these three processing functions in various configurations, which are specified as part of relevant analytics workflows.

In this context, different edge analytics tasks can be described using well-defined configuration files (i.e. Analytics Manifests (AMs)), which reflect analytics workflows and are amenable by visual tools. This facilitates the specification and configuration of analytics tasks as part of the DDA. In particular, solution integrators and manufacturers can flexibly configure their analytics operations through defining proper AMs. Based on the use of proper visual tools, such definitions can be performed with almost zero programming, which is an obvious advantage of the FAR-EDGE DDA over conventional edge analytics frameworks. Furthermore, the DDA leverages several distributed ledger services for storing and configuring AMs across different edge nodes, which provides a novel, secure and resilient way for specifying and executing global analytics tasks.

This chapter is devoted to the presentation of the DDA infrastructure of the FAR-EDGE project, which has been briefly introduced in [4]. This chapter extends the work in [4] through providing more details on the design and implementation details of the DDA platform. Special emphasis is put in describing and highlighting the unique value propositions of the FAR-EDGE DDA in terms of configurability, programmability and resilience. The description includes dedicated parts for the Edge Analytics Engine (EAE) that enable edge scoped analytics and for the Ledger Services for data analytics configuration and synchronization that enable configurable global analytics. Note also that the DDA infrastructure complies with the overall FAR-EDGE reference architecture, which has been introduced in an earlier chapter, while leveraging digital models that are presented in a subsequent chapter. Hence, the present chapter does not detail the overall architecture of the FAR-EDGE platform and the digital models that are used as part of it, since they are both described in other parts of the book.

The structure of this chapter is as follows:

- Section 6.2 following the chapter's introduction presents the main drivers behind the development of a framework for DDA in industrial

environments, through enhancing conventional and popular frameworks for Big Data analytics and streaming analytics.
- Section 6.3 presents the overall architecture of the DDA, including its main modules.
- Section 6.4 illustrates the edge analytics engine of the DDA, including the anatomy of the analytics workflows.
- Section 6.5 presents the ledger services that enable the synchronization of different manifests across edge nodes.
- Section 6.6 presents information about the open source implementation of the DDA, including information about the underlying technologies that have been (re)used.
- Section 6.7 is the final and concluding section of the chapter.

6.2 Requirements for Industrial-scale Data Analytics

As already outlined, most digital automation platforms need to process large volumes of data (including streaming data) as part of wider simulation, decision making and control tasks. Instead of implementing a data analytics function for every new use case, digital automation platforms can offer entire middleware frameworks that facilitate the distributed data analytics tasks (e.g., [8–10]). These frameworks offer facilities for dynamically discovering data sources and executing data processing algorithms over them. In principle, they are Big Data frameworks that should be able to handle large data volumes that features the 4Vs (volume, variety, velocity and veracity) of Big Data. Beyond these general and high-level requirements, the FAR-EDGE DDA infrastructure has been driven by the following principles:

- **High-Performance and Low-Latency:** The FAR-EDGE DDA enables the execution of data analytics logic with high performance, i.e. in a way that ensures low-overhead and low-latency processing of data streams. This is especially important towards handling high-velocity data streams i.e. data with very high ingestion rates such as data streams stemming from sensors attached to a machine.
- **Configurable:** The DDA is configurable in order to be flexibly adaptable to different business and factory automation requirements, such as the calculation of various KPIs (Key Performance Indicators) for production processes. Configurability should be reflected in the ability to dynamically select the data sources that should be used as part of a data analytics task.

- **Extensible:** The DDA provides extensibility in terms of the supported processing functions, i.e. to provide the ability to implement additional data processing schemes based on fair programming effort. In the case of FAR-EDGE, extensibility concerns the implementation of advanced processing capabilities in terms of pre-processing, analyzing and storing data streams.
- **Dynamic:** The DDA is able to dynamically update the results of the analytics functions, upon changes in its configuration. This is essential towards having a versatile analytics engine that can flexibly adapt to changing business requirements and production contexts in volatile industrial environments where data sources join or leave dynamically.
- **Ledger Integration:** One of the innovative characteristics of the DDA lies in the use of a distributed ledger infrastructure (i.e. blockchain-based services) [11] towards enabling analytics across multiple EGs, as well as towards facilitating the dynamic configuration of the data analytics rules that comprise these analytics tasks.
- **Stream Handling Capabilities:** The DDA can handle streaming data in addition to transactional static or semi-static data. This requirement has been considered in the design and the prototype implementation of the DDA infrastructure, which is based on middleware for handling data streams.

Table 6.1 associates these design principles with some concrete implementation examples and use cases.

Table 6.1 Requirements and design principles for the FAR-EDGE DDA

Design Principles and Goals	Examples and use Cases	DDA Implementation Guidelines
High performance and Lowlatency	Complex data analyses over real-time streams should be performed within timescales of a few seconds. As an example, consider the provision of quality control feedback about an automation process in a station, based on the processing of data from the station. The DDA support the collection and analysis of data streams within a few seconds.	Leverage high-performance data streaming technology as background for the EAE implementation (e.g. ECI's streaming technology)

(*Continued*)

Table 6.1 (Continued)

Design Principles and Goals	Examples and use Cases	DDA Implementation Guidelines
Configurable	A manufacturer needs to calculate multiple Key Performance Indicators (KPIs) such as indicators relating to quality control and performance of the automation processes. The DDA should flexibly support the on-line calculation of the different KPIs within the same instance of the EAE. To this end, the EAE should be easily configurable to support the calculation of all desired KPIs, ideally with minimal or even zero programming. Configurability can be gauged based on the time needed to set up and deploy a data analytics workflow comprising several processing functions. The use of EAE is destined to reduce this time, when compared to cases where data analytics are programmed from scratch (i.e. without support from the EAE middleware).	• Specify and implement DDA as a programmable & configurable engine, which executes analytics configurations specified in appropriate files ("manifests"). • Parse and execute the analytics rules of the configuration files, without a need for explicitly programming these rules
Extensible	The EAE should be extensible in terms of data processing, data mining and machine learning techniques. For example, in cases where deep learning needs to be employed (e.g., estimation of a failure mode in predictive maintenance), the EAE must support the execution of machine learning functions, including AI-based algorithms such as deep neural network. The latter can, for example, support the detection of complex patterns such as production quality degradation patterns.	• Provide a library of analytics functions/capabilities and integrate it within a directory. • Provide the means for discovering and using analytics functions from the library analytics configurations.
Dynamic	The EAE should be able to deploy on the fly (i.e. hot deploy) different data analysis instances. For example, when new KPIs should be calculated, calculation shall be done of the fly, without affecting the rest of deployed KPIs.	Leverage multi-threading and hot deployment capabilities of the selected implementation technologies.

Table 6.1 (Continued)

Design Principles and Goals	Examples and use Cases	DDA Implementation Guidelines
Ledger integration	The EAE must integrate functions from the Ledger Services in order to: (i) access configurations of analytics tasks through ledger smart contracts, such as a large scale distributed analytics tasks; (ii) collecting and analyzing data from multiple edge nodes/gateway through access to the publishing services. This can be, for example, the case there data analytics for calculating a product schedule must be computed, as this is likely to span multiple EGs.	• Represent analytics configurations as smart contracts. • Implement publishing services driven by the smart contracts and leveraging information from multiple edge nodes.
Stream handling capabilities	The EAE must be able to handle data-intensive data streams such as sensor data for predictive maintenance and data from other field devices for quality control in automation.	Leveraging streaming handling and management middleware of the ECI.

6.3 Distributed Data Analytics Architecture

A high-level overview of the DDA Infrastructure is provided in Figure 6.1. The DDA consists of wide range of components, which are described in the following subsections.

6.3.1 Data Routing and Preprocessing

The Data Routing and Pre-processing (DR&P) component is in charge of routing data from the data sources (i.e. notably industrial devices) to the Edge Analytics Engine (EA-Engine). The component includes a **Device Registry**, where the various device and data sources announce (i.e. "register") themselves, as well as the means to access their data (i.e. based on connectivity details such as protocol, IP address and port). The registry makes the system dynamic, as it ensures handling of all data sources that register with it. Moreover, the component provides pre-processing capabilities, which allow for transformations to data streams prior to their delivery to the EA-Engine. Note that the DR&P component is edge-scoped i.e. it is deployed at an Edge Gateway (EG). Likewise, the data sources that are registered and managed in the registry concern the devices that are attached to the specific edge gateway as well.

Along with the Device Registry, the DR&P provides a Data Bus, which is used to route streams from the various devices to appropriate consumers, i.e. processors of the EA-Engine. Moreover, the Data Bus is not restricted to routing data streams stemming directly from the industrial devices and other shopfloor data sources. Rather it can also support the routing of additional data streams and events that are produced by the EA-Engine.

6.3.2 Edge Analytics Engine

The EA-Engine is a runtime environment hosted in an EG, i.e. at the edge of an industrial FAR-EDGE deployment. It is the programmable and configurable environment that executes data analytics logic locally to meet stringent performance requirements, mainly in terms of latency. The EA-Engine is also configurable and comprises multiple analytics instances that correspond to multiple edge scoped analytics workflows.

As shown in Figure 6.1, the EA-Engine comprises several processors, which implement processing functions over the data streams of the Data Bus. As illustrated in a following paragraph, these processors are of three main

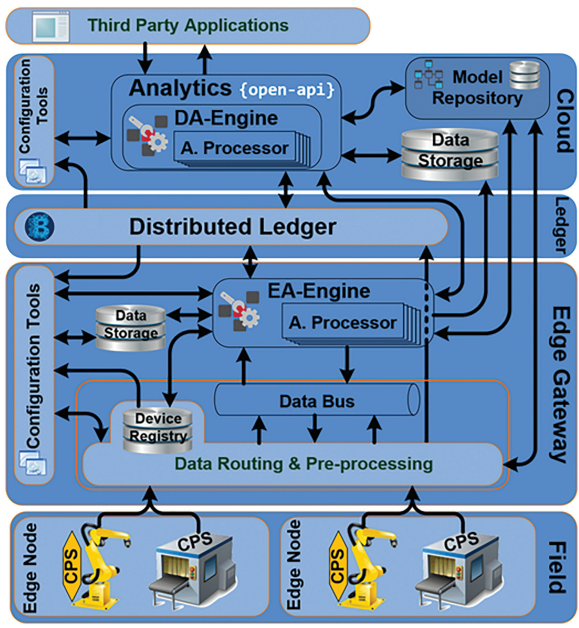

Figure 6.1 DDA Architecture and main components.

types, including processors that store/persist data streams, processors devoted to pre-processing functions, as well as processors in charge of data analytics. Furthermore, the outcomes of the EA-Engine can be written to the Data Bus in order to be consumed by other components and processing functions or even written at local/edge data storage.

6.3.3 Distributed Ledger

The Distributed Ledger is used to orchestrate analytics functionalities across multiple Edge Gateways. It is in charge of maintaining the configuration of different analytics tasks across multiple EGs, which at the same time keep track of their composition in factory-wide analytics tasks. Moreover, the distributed ledger is used to compute the outcomes of factory-wide analytics. Overall, the distributed ledger offers two kinds of services to the DDA, namely Data Publishing Services that synchronize the analytics computations and Configuration Services that synchronize the configuration of the analytics services.

6.3.4 Distributed Analytics Engine (DA-Engine)

While the EA-Engine is in charge of data analytics at edge scope, the DA-Engine is in charge of executing global analytics functions based on the analytics configurations that reside in the distributed ledger. The DA-Engine is configurable thanks to its interfacing with a set of data models that describe the configuration of the DDA infrastructure in terms of edge nodes, edge gateways, data sources and the processing functions that are applied over them as part of the DA-Engine. To this end, the DA-Engine interfaces to a models' repository, which comprises the digital representation of the devices, data sources and edge gateways that are part of the DDA. The Digital Models are kept up to date and synchronized with the status of the DDA's elements. As such, they are accessible from the DR&P and EA-Engine components, which make changes in the physical and logical configuration of the analytics tasks. Note also that the DA-Engine stores data within a cloud-based data storage repository, which is destined to persist and comprise the results of global analytics tasks.

6.3.5 Open API for Analytics

The Open API for Analytics enables external systems to take advantage of the DDA infrastructure functionalities, including both the configuration and

execution of factory-wide analytics tasks, which span multiple edge gateways and take advantage of the relevant EA-Engine instances. Using the Open API any integrator of industrial solutions can specify and execute data processing functions over data streams stemming from the full range of devices that are registered in the device registries of the DR&P components of the DDA infrastructure. As illustrated in the figure, this gives rise to the use of the DDA infrastructure by third-party applications.

The following sections provide insights into the operation and novel features of the EA-Engine and the Distributed Ledger, which endows the DDA with modularity, extensibility and configurability.

6.4 Edge Analytics Engine

6.4.1 EA-Engine Processors and Programmability

One of the unique value propositions of the EA-Engine is that it is configurable and programmable. These properties stem from the fact that it is designed to handle analytics tasks that are expressed based on the combination of three types of processing functions, which are conveniently called "processors". The three types of processors are as follows:

- **Pre-processors**, which perform pre-processing (e.g. filtering) over data streams. In principle, pre-processors prepare data streams for analysis. A pre-processor interacts with a Data Bus in order to acquire streaming data from the field through the DR&P component. At the same time, it also produces and registers new streams in the same Data Bus, notably streams containing the results of the pre-processing.
- **Storage processors**, which store streams to some repository such as a data bus, a data store or a database.
- **Analytics processors**, which execute analytics processing functions over data streams ranging from simple statistical computations (e.g., calculation of an average or a standard deviation) to more complex machine learning tasks (e.g., execution of a classification function). Similar to pre-processors, analytics processors consume and produce data through interaction with the Data Bus.

Given these three types of "processors", analytics tasks are represented and described as combinations of multiple instances of such processing functions in the form of workflow or a pipeline. Such workflows are described through an Analytics Manifest (AM), which specifies a combination of the above processors. Hence, an AM follows a well-defined schema (as shown

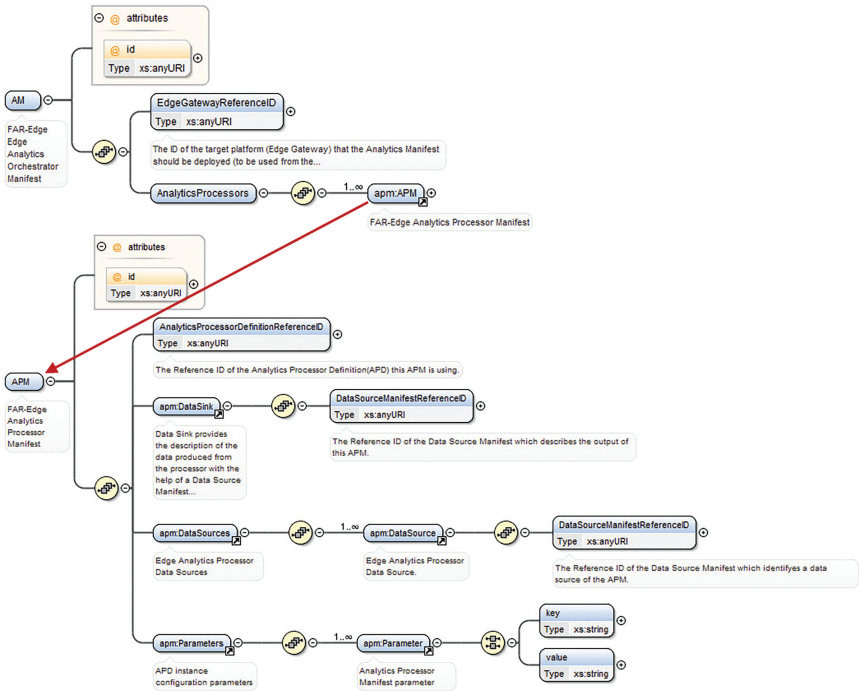

Figure 6.2 Representation of an Analytics Manifest in XML format (XML Schema).

in Figure 6.2), which specifies the processors that comprise the AM. In particular, an AM defines a set of analytics functionalities as a graph of processing functions that comprises the above three types of processors and which can be executed by the EA-Engine.

Note also that an AM instance is built based on the available devices, data sources, edge gateways and analytics processors, which are part of the data models of the DDA. The latter reflect the status of the factory in terms of available data sources and processing functions, which can be used to specify more sophisticated analytics workflows.

6.4.2 EA-Engine Operation

The EA-Engine provides the run-time environment that controls and executes edge analytics instances, which are specified in AMs. In particular, the EA-Engine is able to parse and execute analytics functions specified in an AM, based on the following processes:

- **Parsing:** The EA-Engine parses AMs and identifies the analytics pipeline that has to be executed.
- **Execution:** The EA-Engine executes (applied) the analytic functions that are identified following the parsing. Note that the EA-Engine is multi-threaded and enables the concurrent (parallel) execution of multiple analytics pipelines, which can correspond to different AMs.

Figure 6.3 illustrates an example topology and runtime operations for EA-Engine. In this example, two streams (CPS1 and CPS2) are pre-processed from Analytics Processor 1 (i.e. Pre-Processor) and Analytics Processor 2 (i.e. Pre-Processor) equivalently in order to enable the execution of an analytics algorithm that is in Analytics Processor 3, which is an Analytics Processor. Finally, the pipelines ends-up storing the result to a Data Store based on Analytics Processor 4, which is a Storage Processor. In this example, the EA-Engine is set up and runs based on the following steps:

- **Step 1 (Set-up):** Based on the description of the topology and required processors in the AM, the engine instantiates and configures the required Analytics Processors. Note that the AM is built based on real information about the factory, which is reflected in the digital models of the DDA infrastructure.
- **Step 2 (Runtime):** Analytics Processor 1 consumes and pre-processes streams coming from CPS1. Likewise, Analytics Processor 2 consumes

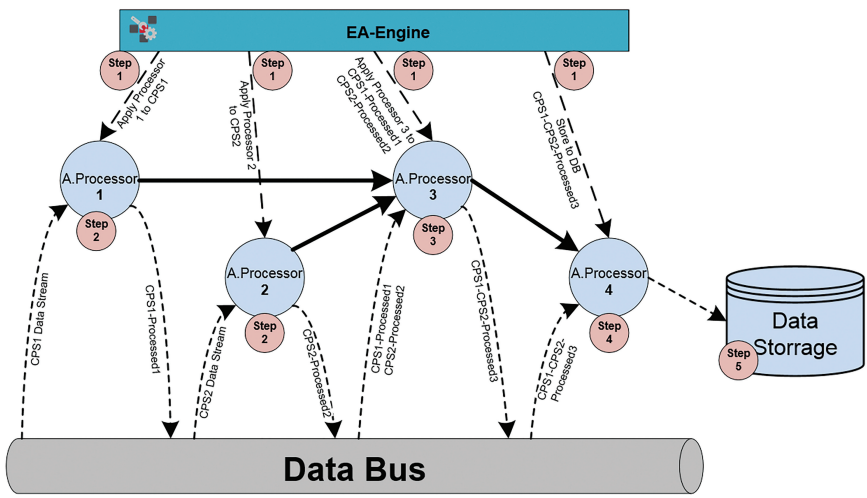

Figure 6.3 EA-Engine operation example.

and pre-processes streams coming from CPS2. In both cases, the streams are accessed through the Data Bus.

- **Step 3 (Runtime):** Analytics Processor 3 consumes the produced streams from Analytics Processor 1 and 2 towards applying the analytics algorithm. In this case, the analytics processor cannot execute without input for the earlier Analytics Processors.
- **Step 4 (Runtime):** Store Analytics Processor 4 consumes the data stream produced from Analytics Processor 3 and forwards it to the Data Store, which persists and data coming from Analytics Processor 4.

This is a simple example of the EA-Engine operation, which illustrates the use of all three types of processors in a single pipeline. However, much more complex analytics workflows and pipelines can be implemented based on the combination of the three different types of processors. The only limiting factor is the expressiveness of the AM, which requires that instances of the three processors are organized in a graph fashion, with one or more processors providing input to others.

Vendors and integrators of industrial automation solutions can take advantage of the versatility of the EA-Engine in two ways:

- First, they can leverage existing processors of the EA-Engine towards configuring and formulating analytics workflows in line with the needs of their application or solution.
- Second, they can extend the EA-Engine with additional processing capabilities, in the form of new reusable processors.

In practice, industrial automation solution integrators will use the EA-Engine in both the above ways, which are illustrated in the following paragraphs.

6.4.3 Configuring Analytics Workflows

Integrators can configure and execute edge-scoped analytics pipelines. The configuration of a new pipeline involves the following steps:

- **Discovery of Devices** and other data sources registered in the device registry. Analytics workflows can only take advantage of devices and data sources that are registered with the DR&P component.
- **Discovery of available processors**, a list of which is maintained in the EA-Engine. The rationale behind this discovery is to reuse existing processors instead of programming new ones. Nevertheless, in cases where the analytics workflow involves a processor that is not yet available,

this processor should be implemented from scratch. However, every new processor will become available for reuse in future analytics workflows.
- **Definition and creation of the Analytics Manifest**, based on the available (i.e. discovered) devices, data sources and processors. As already outlined, an AM comprises a graph of processors of the three specified types, defines the analytics results to be produced and specified where they are to be stored. The specification of the AM can take place based on the use of the Open API of the DDA. However, as part of our DDA development roadmap, we will also provide a visual tool for defining AMs, which facilitate zero-programming specification of the edge analytics tasks.
- **Runtime execution of the AM**, based on the invocation of appropriate functions of the EA-Engine's runtime. This step can be implemented based on the Open API of the DDA, yet it is also possible to execute it through a visual tool.

6.4.4 Extending the Processing Capabilities of the EA-Engine

Integrators can specify additional processing functions and make them available for use as part of the EA-Engine. The extension process involves the following steps:

- **Implementation of a Processor Interface:** In order to extend the EA-Engine with a new processor, an integrator has to provide an implementation of a specific interface i.e. the interface of the processor. In practice, each of the three processor types comes with its own interface.
- **Registration of the Processor to a Registry:** Once a new processor is implemented, it has to become registered to a registry. This will make it discoverable by solution developers and manufacturers that develop AMs for their needs, based on available devices and processors.
- **Using the processor**: Once a processor becomes available, it can be used for constructing AMs and executing analytics tasks that make use of the new processor.

6.4.5 EA-Engine Configuration and Runtime Example

In this section, we use the topology illustrated in Figure 6.3 above in order to provide a more detailed insight into the steps needed to configure the EA-Engine, but also in order to illustrate the interactions between the various components both at configuration time and at run time. As already outlined,

the example involves two devices (CPS1, CPS2), which generate two data streams under a topic each one named after their ID. We therefore need to:

- Apply some pre-processing to each one of the two streams (by Processor 1 and Processor 2).
- Apply an Analytics algorithm (Processor 3) to the pre-processed streams.
- Persist the result to a Data storage (i.e. the Data Storage).

Figure 6.4 illustrates the steps required to register a new processor, build the Edge Analytics configuration (AM), register it to the EA-Engine and instantiate the appropriate Analytics Processors. In particular:

- The user of the EA-Engine (e.g. a solution integrator) registers new Processors required to the Model Repository. To this end, it can use an API or a visual tool.
- In order to set up an AM, all the available processors are discovered from the Model Repository and all the available Data Sources (DSMs) are discovered from the Distributed Ledger.
- The user has all the required information and with the help of the Configuration Dashboards can now set up a valid AM flow for the four Analytic Processors.
- The AM is set up based on a proper combination of devices data streams and processors. In this example, the AM includes the required configurations for Processor 1 (APM1), Processor 2 (APM2), Processor 3 (APM3) and Processor 4 (APM4).

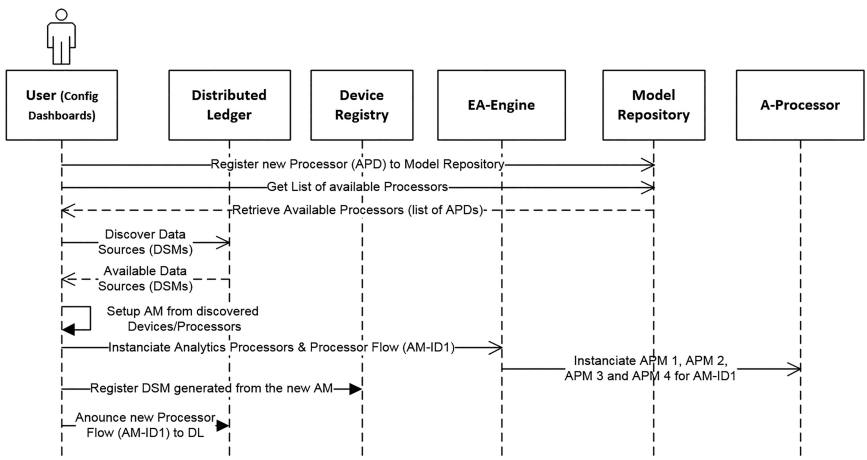

Figure 6.4 EA-Engine configuration example (Sequence Diagram).

- The AM is accordingly sent to the EA-Engine, which instantiates the four Analytic Processors.
- The output of the AM is automatically described in a new DSM, which is registered to the Device Registry as a new Data Source and synchronized with the Distributed Ledger through the Device Registry mechanisms.
- The capabilities of the new processor are also registered to the Distributed Ledger to enable the discoverability of the new processor for future use.

Figure 6.5 illustrates the interactions between the EA-Engine components, when the execution of the AM starts. These include:

- Instructing the EA-Engine to start the execution of the analytics task, as specified in the analytics manifest (AM1). To this end, the EA-Engine retrieves AM1 from the Distributed Ledger in order to instantiate the processors that AM1 comprises.
- The EA-Engine instantiates each of the four EA-Processors described in the AM1. Specifically:
    - As part of the instantiation of Processor 1 (pre-processor), its specification (APM1) contains the configurations of Processor 1,

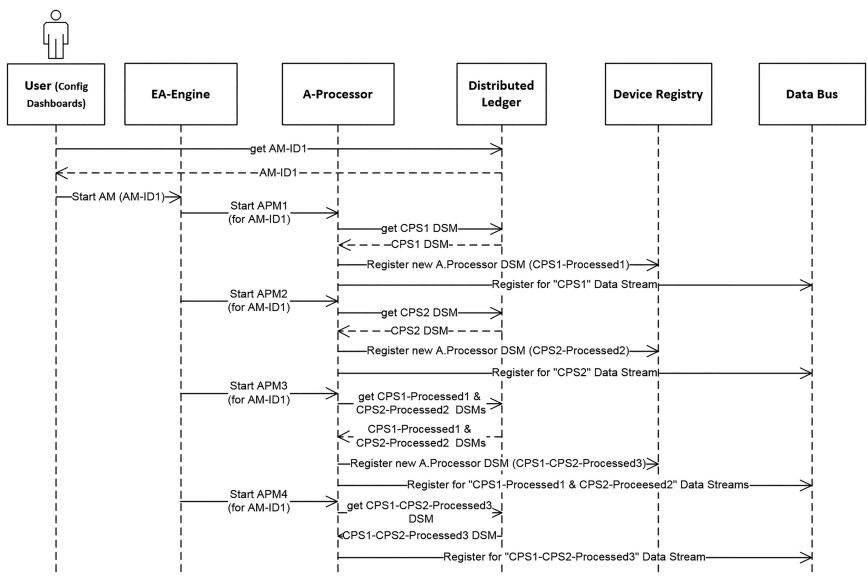

Figure 6.5 EA-Engine initialization example (Sequence Diagram).

which includes data inputs, data outputs and processor attributes required for the instantiation. The data type and data model of CPS1 are retrieved from the Ledger Service in order to apply the pre-processing properly. The processor data output description is provided within a new DSM that is registered to the Device Registry. Then, the EA-Processor (Processor 1) subscribes for the "CPS1" data stream of the Data Bus to apply the required pre-processing.

- As part of the instantiation of Processor 2 (pre-processor), its specification (i.e. APM2) contains the configurations of Processor 2, which includes data inputs, data outputs and processor attributes required for the instantiation. The data type and data model of CPS2 are retrieved from the Ledger Service. Also, the EA-Processor (Processor 2) subscribes for the "CPS2" data stream of the Data Bus in order to apply the required pre-processing.
- As part of the instantiation of Processor 3 (analytics processor), its specification (APM3) contains the configurations of Processor 3. Processor 3 subscribes to the topics named after the IDs of Processor 1 and Processor 2 ("CPS1-Processed 1" and "CPS2-Proceesed 2", respectively) in order to apply the required analytics.
- Finally, as part of the instantiation of Processor 4 (store processor), its specification (APM4) is retrieved from the EA-Storage. Processor 4 subscribes to the topics named after the ID of Processor 3 ("CPS1-CPS2-Processed 3") in order to store it to the data storage.

The runtime operation of the EA-Engine is further presented in Figure 6.6, which illustrates the sequence of runtime interactions of the components of the engine, following the conclusion of the above-listed configurations. At runtime, all the different processors run continuously in parallel until they are stopped from the end-user through a proper API command or based on the use of the visual tool. In particular:

- **Processor 1** gets notified every time new CPS1 data is published and collects it. It applies the required pre-processing and pushes the pre-processed data stream back to the data bus under the topic named after its own ID ("CPS1-Processed 1").
- **Processor 2** gets notified every time new CPS2 data is published and collects it. It applies the required pre-processing and pushes the pre-processed data stream back to the data bus under the topic named after its own ID ("CPS2-Processed 2").

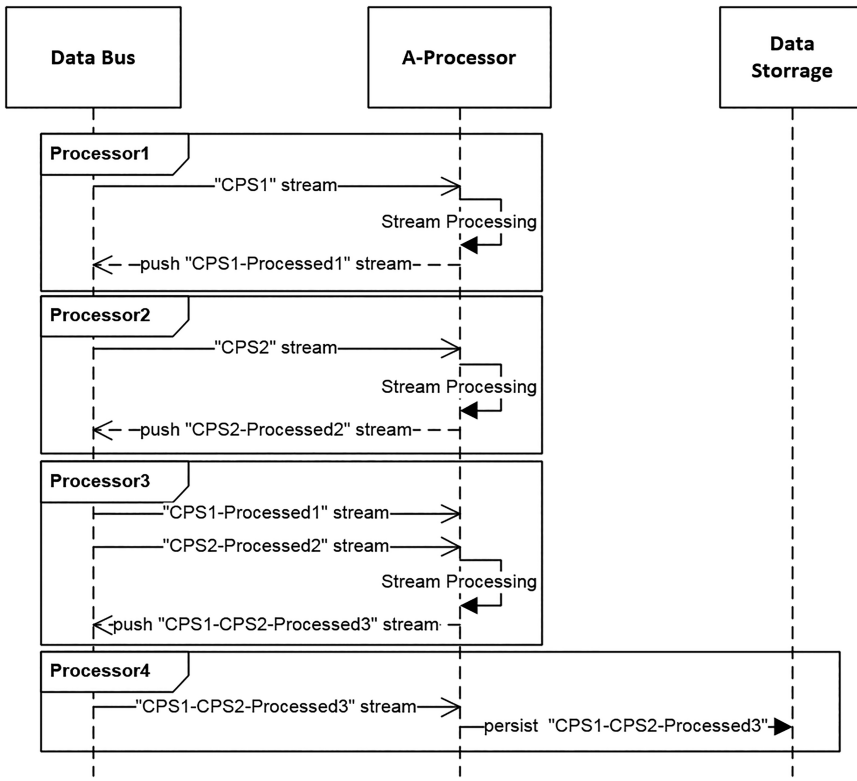

Figure 6.6 EA-Engine runtime operation example (Sequence Diagram).

- **Processor 3** gets notified every time new Processor 1 and Processor 2 data is published and collects it. It applies the required analytic and pushes the processed data stream back to the data bus under the topic named after its own ID ("CPS1-CPS2-Processed 3").
- **Processor 4** gets notified every time new Processor 3 data is published and collects it. It pushes the collected data to the EA-Storage to be persisted.

6.5 Distributed Ledger and Data Analytics Engine

6.5.1 Global Factory-wide Analytics and the DA-Engine

Given the presented functionalities of the EA-Engine, the DA-Engine enables the combination and synchronization of data from multiple edge analytics

pipelines towards implementing factory-wide analytics. At a high level, the concept of global analytics workflows is similar to the one of edge analytics ones. In particular, an Analytics Manifest (AM) is used to express an analytics workflow based on the combination of analytics tasks that are configured and executed at edge gateways based on properly configured instances of the EA-Engine. To this end, a mechanism for constructing AMs that comprise global analytics tasks is provided through the Open API of the DDA. In particular, the Open API provides the means for creating, updating, deleting, managing and configuring global analytics tasks based on the combination and orchestration of edge analytics workflows.

At a lower level, the implementation of the AM configuration and execution mechanism is offered in two flavours:

- **A conventional edge computing implementation**, which is subject to conventional central control. It involves an analytics engine that combines edge analytics workflows to global ones for a central orchestration point. That is in line with the classical edge/cloud computing paradigm.
- **A novel distributed ledger implementation**, which is based on a disruptive cooperative approach without central control. This cooperative approach is based on the deployment and use of ledger services in each one of the edge nodes that participate in the DDA infrastructure. In particular, ledger services are deployed in each of the edge gateways in order to enable a consensus-based approach regarding the configuration of the global analytics task, as well as its execution based on publishing and combination of data from the edge gateways. Such a collaborative approach is fully decentralized and hence does not provide a single point of failure. Moreover, it can be generalized beyond edge gateways in order to enable data analytics workflows that comprise data from field objects (i.e. smart objects) and cloud nodes as well.

The next sub-section illustrates the scope and operation of these ledger services, which enable a novel and more interesting approach to supporting the functionalities of the DA-Engine.

6.5.2 Distributed Ledger Services in the FAR-EDGE Platform

For the implementation of the DA-Engine, we leverage the services of a permissioned blockchain, rather than of one of the popular public blockchains such as Bitcoin and Ethereum. The rationale behind this decision is that permissioned blockchains provide the means for controlling participation and

authenticating participants to the blockchain network, while offering superior performance over public blockchains [12]. The latter performance is largely due to the fact that peer nodes (i.e. participants) in these blockchains need not employ complex Proof-of-Work (PoW) mechanisms. For these reasons, a permissioned blockchain is more appropriate for coordinating and synchronizing distributed processes in an industrial context.

In this context, a Ledger Service is a Chaincode program for IBM's Hyperledger Fabric, which uses some of the utility services that are provided by the FAR-EDGE platform. Chaincode is always designed to support a well-defined, application-specific process. Hence, the DDA implementation is not based on a generic Ledger Service implementation, but rather on application-specific Ledger Service. Nevertheless, four categories of abstract services are defined as part of the Ledger Tier of the FAR-EDGE Architecture, namely Orchestration, Configuration, Data Publishing and Synchronization. These categories are used to classify the application-specific implementations of Ledger Services rather than to denote some general-purpose framework services. In particular:

- **Orchestration Services** are related to edge automation workflows, aiming at synchronizing distributed edge automation tasks in factory-wide automation workflows.
- **Data Publishing Services** support edge analytics algorithms, through the combination of multiple edge analytics pipelines in factory-wide workflows.
- **Synchronization Services** enable the reconciliation of several independent views of the same dataset across the factory.
- **Configuration Services** support the decentralized system administration.

Overall, these four categories of Ledger Services cover all the mandatory platform-level functionality that is required for Edge Computing to deliver its promises in a manufacturing context. The Distributed Ledger of the FAR-EDGE platform can then be used to deploy any kind of custom Ledger Service that meets the secure state sharing and/or decentralized coordination requirements of user applications.

Any concrete Ledger Service implementation is responsible for three things:

- **Defining and managing a data model**. While the global state of the Ledger Service is automatically maintained in the background by the DL-Engine – which logs every state change in the Ledger that is

replicated across all the peer nodes of the system – the data model of such state is shaped in code by the Ledger Service implementation itself. Practically speaking, the data store of a Ledger Service is initialized according to a specific data model by a special code section when the instance is first deployed. Once initialized, no structural changes in the data model occur.

- **Defining and executing business logic**. Application logic is coded in software and exposed on the network as a number of application-specific service endpoints, which can be called by clients. These service calls represent the API of the Ledger Service. Through them, callers can query and change the global state of the Ledger Service. The API can be invoked by any authorized client on the network following some well-documented calling conventions of the DL-Engine. Moreover, we have implemented an additional layer of software in order to simplify the development of client applications: each Ledger Service implemented in the project comes with its own client software library – called Ledger Client – which an application can embed and use as a local proxy of the actual Ledge Service API. The Ledger Client provides an in-process API, which has simple call semantics.
- **Enforcing (and possibly also defining) fine-grained access and/or usage policies**. This is optional one, as a basic level of access control is already provided by the DL-Engine, which requires all clients to have a strong digital identity and be approved by a central authority. When a more fine-grained control is required – e.g. an Access Control List (ACL) applied to individual service endpoints – the Ledger Service implementation is required to manage it as part of its code.

In the specific context of the FAR-EDGE Platform, peer nodes are usually – but not mandatorily – installed on Edge Gateway servers, together with Edge Tier components. This setup allows for DL clients that run on Edge Gateways, like the EA-Engine, to refer to a localhost address by default when resolving Ledger Service endpoints. However, this is not the only possible way to deploy the Ledger Tier in FAR-EDGE-enabled system: peer nodes can easily be deployed on the Cloud Tier to make them addressable from anywhere or even embedded in Smart Objects on the Field Tier to make them fully autonomous systems. In complex scenarios, peer nodes can actually be spread across all the three physical layers of the FAR-EDGE architecture (Field, Edge and Cloud), exploiting the flexibility of the DL enabler to its full extent.

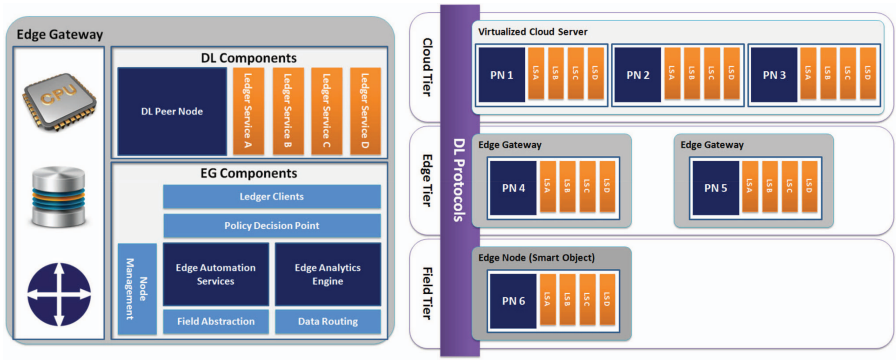

Figure 6.7 DL deployment choices (right) and EG deployment detail (left).

6.5.3 Distributed Ledger Services and DA-Engine

The DA-Engine takes advance of two of the above-listed types of Ledger Services, namely the Data Publishing and Configuration services. In particular, the DDA infrastructure implements Data Publishing and Configuration services at the Ledger Tier, in order to configure factory-wide AMs and to implement the respective analytics. In particular:

- **Configuration Services:** DDA configurations (i.e. AMs) are represented as smart contracts. Each smart contract is executed by the peers (notably edge gateway) that participate in the configuration and execution of the factory-wide AM. A set of Configuration services (Ledger Services) are used to ensure the configuration of the global analytics manifest based on consensus across the participating nodes. In this case, the distributed ledger is used as a distributed database that holds all the analytics configurations (in terms of manifests and their component). This allows the resilient configuration of global analytics without a need for centralized coordination and control from a single point of (potential) failure.
- **Publishing Services:** Publishing Services are implemented in order to compute factory-wide analytics tasks, based on data streams and analytics (i.e. processors) available across multiple instances of the EA-Engine, which are deployed in different Edge Gateways (EGs). The EGs act as peers in this case.

6.6 Practical Validation and Implementation

6.6.1 Open-source Implementation

The DA-Engine is implemented as open-source software/middleware, which is available at the FAR-EDGE github: https://github.com/far-edge/distributed-data-analytics. In the absence of general-purpose Ledger Services, the implementation includes the middleware for edge analytics framework of Section 6.3, as well as an Open API for creating Analytics Manifests for global, factory-wide analytics. Hence, a subset of the DDA architecture has been actually implemented, which is shown in Figure 6.8. As evident from the figure, the open-source implementation includes the EA-Engine and the DA-Engine, without however general-purpose ledger services, which is the reason why the Distributed Ledger database is not depicted in the figure. In a nutshell, the implementation includes and integrates the DR&P, the Data Bus, the Device Registry, the Data Storage (including both cloud and local data storage) and the Model Repository components.

The structure of the open-source codebase is as follows:

- **edge-analytics-engine**, which contains the source code of the EA-Engine component.

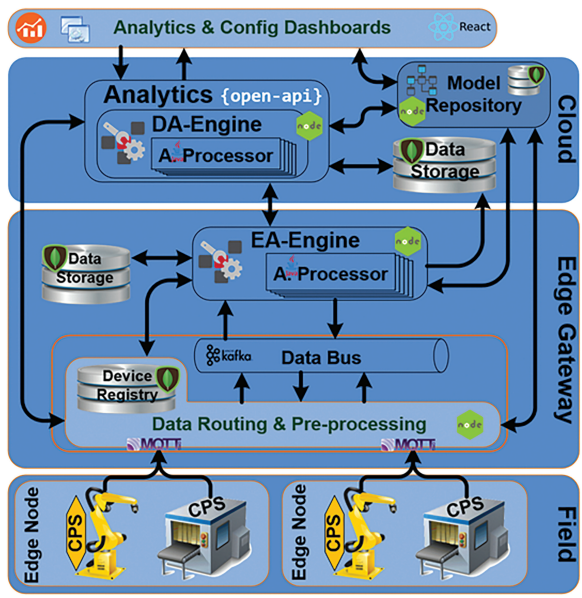

Figure 6.8 Elements of the open-source implementation of the DDA.

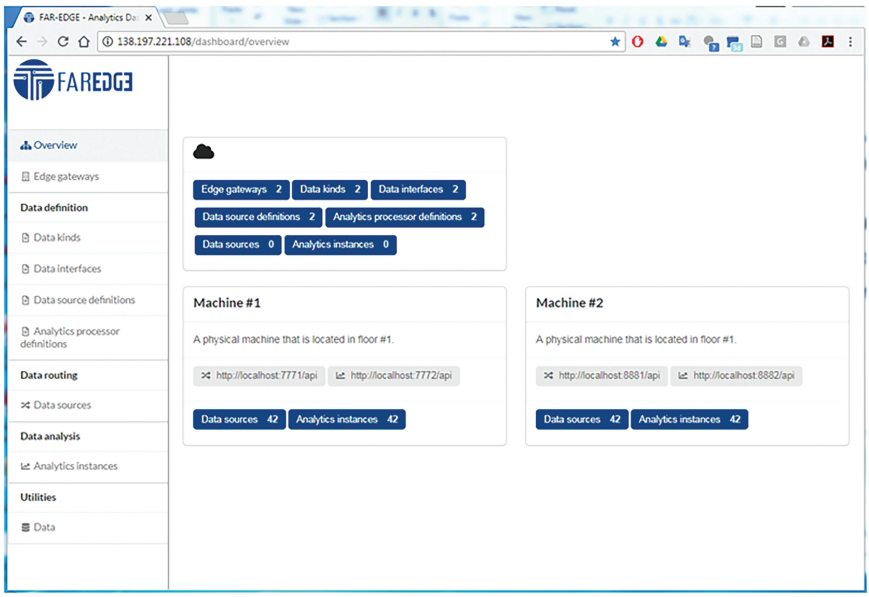

Figure 6.9 DDA Visualization and administration dashboard.

- **open-api-for-analytics**, which contains the component that implements and supports the Open API for Analytics.
- **mqtt-random-data-publisher**, which contains an application that simulates the functionality of DR&P component in order to facilitate the easier setup of simple demonstrators.

Furthermore, a set of administration dashboards that visualize the main entities of the DDA have been implemented. It allows the monitoring and the configuration of main entities like processors, data sources, devices and manifests (see Figure 6.9).

6.6.2 Practical Validation

6.6.2.1 Validation environment

The DDA Infrastructure has been also validated in a pilot plant and specifically in the pilot plant of SmartFactoryKL, which is a network with more than 45 industrial and research organizations that support and use an Industrie 4.0 testbed in Kaiserslautern, Germany. In particular, we set up a relatively simple analytics scenario over three Infrastructure Boxes (IB) of the pilot plant. Each Infrastructure Box (IB) provides energy sensors information

through an MQTT interface (Broker), where Data are provided every 60 seconds. The available energy information provided includes data about the TotalRealPower, the TotalReactivePower, the TotalApparentPower, the TotalRealEnergy, the TotalReactiveEnergy and the TotalApparentEnergy that are consumed and used by the machine. The business rationale behind analyzing this data is to help the plant operator in finding anomalies during production. Indeed, with the power and energy values, it is possible to understand the machine behaviour as well as the "response time" of each business process. Moreover, the use of streaming processing and high-performance analytics enables the identification and understanding of abnormalities almost in real time.

The following components were deployed and used in the pilot plant:

- **The Data Routing and Pre-processing (DR&PP) Component** (including device registry service), which forwards data generated by Field sources.
- **The Edge Tier Data Storage**, which stores data stemming from the EA-Engine and provides a result storage repository.
- **The Model Repository**, which supports the sharing of common digital models, which are used from the various analytics components.
- **The EA-Engine**, which is the programmable and configurable environment that executes data analytics logic locally.
- **The Analytics Processor**, which implements the data processing functionalities for an edge analytics task.

The components are deployed in a Virtual Machine (VM) provided within the Smart Factory premises, which had access to data from the IB based on the MQTT protocol. The DDA has been tested and validated in two different scenarios, involving edge analytics and (global) distributed analytics. Various test cases have successfully run and analytics results have been correctly computed. The following subsections illustrate the setup of the EA-Engine and the DA-Engine in the scope of the two scenarios.

6.6.2.2 Edge analytics validation scenarios

For the Edge Analytics, we provide the hourly daily consumption from each Infrastructure Box for two parameters, namely TotalRealPower and TotalRealEnergy. The following steps have been followed for setting up and modelling the Edge Analytics scenario:

- **IB Modelling:** One Edge Gateway is built with each IB. The latter is modelled in line with the FAR-EDGE digital models for data analytics.

The respective data model is stored at the Data Model repository in the cloud.

- **IB Instantiation & Registration:** The specified Data models are used to generate the Data Source Manifest (DSM) and register it to each Edge Gateway.
- **Edge Analytics Modelling:**The required processor is modelled with the help of an Analytics Processor Definition (APD). In particular, the following processors are defined: (i) A processor for hourly average calculation from a single data stream and (ii) Processor for persisting results in a MongoDB. The above information is also stored at the Data Model repository in the cloud.
- **Edge Analytics Installation & Registration**: The specified Data models are used to generate the Analytics Processor Manifest (APM) for each required Processor, which is registered to the Edge Gateway. The following processors are set up: (i) A Processor for hourly average calculation from the TotalRealPower data stream; (ii) A Processor for hourly average calculation from the TotalRealEnergy data stream; (iii) A Processor for persisting results in the MongoDB of an EG in order to support edge analytics calculations; and (iv) A Processor for persisting results in a global (cloud) MongoDB in order to support (global) distributed analytics. Moreover, an AM is also created in order to combined values and data from the instantiated processors. The AM is registered and started through the API of the EG.

Following the setup and configuration of the system, runtime operations are supported, including the following information flows:

- IBs pushes the data to MQTT broker.
- The DR&P retrieves raw/text data from MQTT broker and pushes them to an Apache Kafka Data Bus.
- The data are retrieved and processed from the Analytics Engine.
- The data are finally stored to the local Data Storage repository.

6.6.2.3 (Global) distributed analytics validation scenarios

For the Distributed Analytics validation, we provide the hourly daily consumption from all IBs for the TotalRealPower and the TotalRealEnergy parameters. The following steps are also needed in addition to setting up the EA-Engine:

- **Distributed Analytics Modelling:** The required processors will be modelled with the help of an Analytics Processor Definition (APD)

construct of the FAR-EDGE data models. The processors that are set up include: (i) A Processor for hourly average calculation for values from a MongoDB and (ii) A Processor for persisting results in a MongoDB. The above information is stored at the Data Model repository, which resides on the cloud.

- **Distributed Analytics Installation & Registration:** The specified data models are used to generate the Analytics Processor Manifest (APM) for each required Processor and are registered to the Cloud. The following processors are registered: (i) A Processor for hourly average calculation from the TotalRealPower parameters for all IBs based on information residing in the (global) MongoDB in the cloud; (ii) A Processor for hourly average calculation from TotalRealEnergy for all IBs based on information residing in the (global) MongoDB in the cloud; and (iii) A Processor for persisting results in the (global) MongoDB in the cloud. An Analytics Manifest (AM) will be generated for combining data from the instantiated Processors. The AM will be registered and started through the Open API of the DA-Engine.

6.7 Conclusions

Distributed data analytics is a key functionality for digital automation in industrial plants, given that several automation and simulation functions rely on the collection and analysis of large volumes of data (including streaming data) from the shopfloor. In this chapter, we have presented a framework for programmable, configurable, flexible and resilient distributed analytics. The framework takes advantage of state-of-the-art data streaming frameworks (such as Apache Kafka) in order to provide high-performance analytics. At the same time however, it augments these frameworks with the ability to dynamically register data sources in repository and accordingly to use registered data sources in order to compute analytics workflows. The latter are also configurable and composed of three types of data processing functions, including pre-processing, storage and analytics functions. The whole process is reflected and configured based on digital models that reflect the status of the factory in terms of data sources, devices, edge gateways and the analytics workflows that they instantiate and support.

The analytics framework operates at two levels: (i) An edge analytics level, where analytics close to the field are defined and performance and (ii) A global factory-wide level, where data from multiple edge analytics deployments can be combined in arbitrary workflows. We have also presented

two approaches for configuring and executing global level analytics: One following the conventional edge/cloud computing paradigm and another that support decentralized analytics configurations and computations based on the use of distributed ledger technologies. The latter approach holds the promise to increase the resilience of analytics deployments, while eliminated single point of failure and is therefore one of our research directions.

One of the merits of our framework is that it is implemented as open-source software/middleware. Following its more extensive validation and the improvement of its robustness, this framework could be adopted by the Industry 4.0 community. It could be really useful for researchers and academics who experiment with distributed analytics and edge computing, as well as for solution providers who are seeking to extend open-source libraries as part of the development of their own solutions.

Acknowlegdements

This work was carried out in the scope of the FAR-EDGE project (H2020-703094). The authors acknowledge help and contributions from all partners of the project.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, M. Hoffmann, 'Industry 4.0', Business & Information Systems Engineering, vol. 6, no. 4, pp. 239, 2014.

[2] J. Soldatos (editor) 'Building Blocks for IoT Analytics', River Publishers Series in Signal, Image and Speech Processing, November 2016, ISBN: 9788793519039, doi: 10.13052/rp-9788793519046.

[3] J. Soldatos, S. Gusmeroli, P. Malo, G. Di Orio 'Internet of Things Applications in Future Manufacturing', In: Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds, Editors: Dr. Ovidiu Vermesan, Dr. Peter Friess. 2016. ISBN: 978-87-93379-81-7.

[4] M. Isaja, J. Soldatos, N. Kefalakis, V. Gezer 'Edge Computing and Blockchains for Flexible and Programmable Analytics in Industrial Automation', International Journal on Advances in Systems and Measurements, vol. 11 no. 3 and 4, December 2018 (to appear).

[5] T. Yu, X. Wang, A. Shami 'A Novel Fog Computing Enabled Temporal Data Reduction Scheme in IoT Systems', GLOBECOM 2017 - 2017 IEEE Global Communications Conference, pp. 1–5, 2017.

[6] S. Mahadev et al. 'Edge analytics in the internet of things', IEEE Pervasive Computing, vol. 14, no. 2, pp. 24–31, 2015.
[7] M. Yuan, K. Deng, J. Zeng, Y. Li, B. Ni, X. He, F. Wang, W. Dai, Q. Yang, "OceanST: A distributed analytic system for large-scale spatiotemporal mobile broadband data", PVLDB, vol. 7, no. 13, pp. 1561–1564, 2014.
[8] A. Jayaram 'An IIoT quality global enterprise inventory management model for automation and demand forecasting based on cloud', Computing Communication and Automation (ICCCA) 2017 International Conference on, pp. 1258–1263, 2017.
[9] J. Soldatos, N. Kefalakis, M. Serrano, M. Hauswirth, A. Zaslavsky, P. Jayaraman, and P. Dimitropoulos 'Practical IoT deployment on Smart Manufacturing and Smart Agriculture based on an Open Source Platform', in Internet of Things Success Stories, 2014.
[10] John Soldatos, Nikos Kefalakis et. al. 'OpenIoT: Open Source Internet-of-Things in the Cloud', OpenIoT@SoftCOM, 2014: 13–25, 2014.
[11] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wan. 'An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends', Proceedings of IEEE 6th International Congress on Big Data, 2017.
[12] Elli Androulaki et al. "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains", Proceedings of the Thirteenth EuroSys Conference (EuroSys '18), Article No. 30, Porto, Portugal, April 23–26, 2018.