Towards a Cloud-Based Analytics Framework for Assembly Systems

German Terrazas, Lavindra de Silva, and Svetan Ratchev

Faculty of Engineering, University of Nottingham, Nottingham, UK
german.terrazas@nottingham.ac.uk

Abstract. Advanced digitalization together with the rise of cloud technologies is a key enabler for a fundamental paradigm shift known as Industry 4.0 which proposes the integration of the new generation of ICT solutions for the monitoring, adaptation, simulation and optimization of factories. With the democratization of sensors, assembly systems can now be sensorized and the data generated by these devices can be exploited, for instance, to monitor their utilization, operations and maintenance. However, analyzing the vast amount of generated data is resource demanding both in terms of computing power and network bandwidth, especially when dealing with real-time changes to product, process and resource domains. This paper presents a novel cloud-based analytics framework for the management and analysis of assembly systems. It brings together standard open source technologies and the exploitation of cloud computing which as a whole can be adapted to and deployed on different cloud providers, thereby reducing infrastructure costs, minimizing deployment difficulty and providing on-demand access to virtually infinite computing power, storage and network resources.

Keywords: Precision assembly systems, Cloud computing, Data analytics.

1 Introduction

The integration of the new generation of ICT solutions for the monitoring, adaptation, simulation and optimization of factories have enabled cyber-physical production systems (CPPS) to break from the traditional assembly systems automation pyramid comprising the *field level*, *control level*, *process control level*, *plant management level* and *enterprise resource planning level*. This traditional structure now comprises just the control and field levels (i.e., PLCs are kept close to technical processes to deliver the highest performance), with a complete decentralization of the higher levels. With this new organization, some of the features fully automated CPPS are expected to have are autonomy, self-organization, remote diagnosis, predictability, interoperability, real-time control, global tracking and tracing, and efficiency to name a few [1]. This has not affected the way production takes place in an assembly system, which is based on a series of tasks systematically developed and executed throughout the so-called *product domain*, *process domain* and *resource domain*. Although current sophisticated approaches can reflect product changes in the assembly sequence and, in turn, push this

assembling change through the realization process [2], little has been said on how automated assembly systems could learn from the increasing amount of manufacturing data to predict, adapt and control changes at any domain level in real time and efficiently, especially when analyzing the vast amount of generated data is resource demanding both in terms of computing power and network bandwidth. While there is related work in computer-aided process planning, most of them generate outdated and unfeasible process plans due to the lack of integration with the shop floor to capture resource availability or other dynamic changes [3, 4]. Other approaches propose bespoke cloud-based solutions in terms of resource monitoring and optimized adaptive process planning but operate across the process domain and resource domain only and fail to address big data scalability issues [5, 6]. In what follows: Section 2 presents an implemented cloud-based data analytics framework built in terms of open source technologies for the collection and management of shop floor data; Section 3 presents two automated assembly test beds; and Section 4 discusses how they could be integrated within a complete CPPS, and the challenges and opportunities in real-time assembly monitoring at all levels.

2 A Data Analytics Framework for the Cloud

The architecture for our cloud-based data analytics framework can be conceptually seen as a set of layers that, as a whole, depict different types of domain entities arranged across different levels of abstraction together with their relationships and associated behaviors (see Fig. 1). A layer is a logical division that groups software components by functionality without taking into account their physical location. We define four different levels of abstraction comprising the *Knowledge* layer, the *Analytics* layer, the *Application* layer and the *Presentation* layer. These layers can be seen as sequentially arranged, one upon another, where components in a given layer can only communicate with those in the layer above or the layer below. The fundamental concept behind this idea [7] is the isolation of layers, i.e., software components within a layer are independent from those located in other layers and have no knowledge of their internal structure. While the most challenging part of designing an architecture of this kind is to define the layers and their functionalities, using a layered scheme brings domain independence, loose-coupling and reusability benefits. Additionally, the logical separation offered by the four levels of abstraction contributes to the flexibility and scalability of the proposed framework.

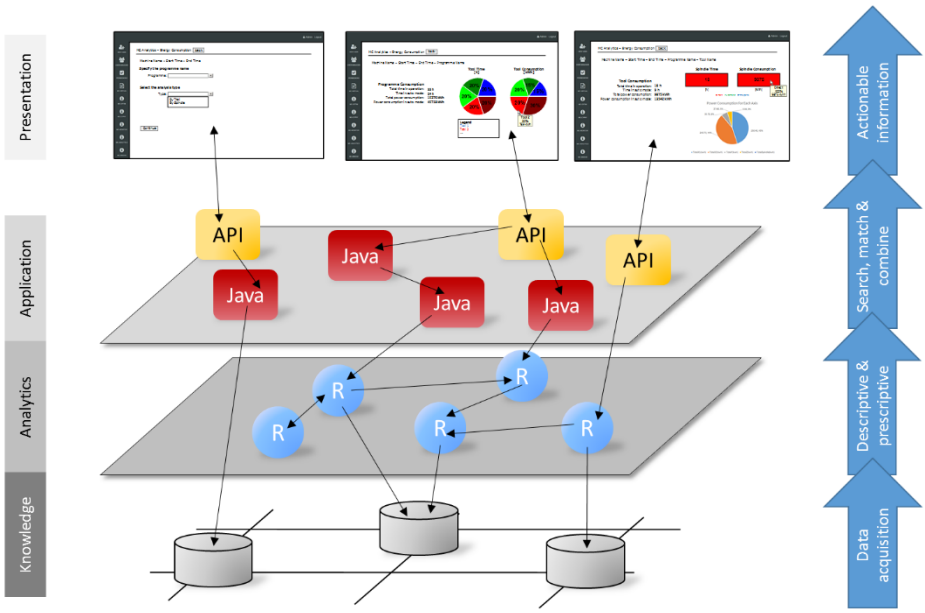

Fig. 1. The conceptual architecture of our analytics framework comprising domain entities across different levels of abstraction. The *Knowledge* layer captures relevant data sources of manufacturing data. This is accessed by descriptive and prescriptive business logic which is searched, matched, combined, configured and effectively provided to consumers for decision making.

The Knowledge layer. This comprises the storage and retrieval of persistent assembly data which forms the basis for data analytics. Technically speaking, this layer comprises document databases, which are one of the main categories of NoSQL databases. Document databases store all the information relating to a given object in a single instance in the database, which makes a document database attractive for cloud-based applications [8, 9], where speed of deployment is important. While implementations differ, this technology assumes that documents encapsulate data in some standard format, such as Java-Script Object Notation (JSON). Document databases allow different types of document structures in a single store and allow the fields within them to be optional, hence providing flexibility when organizing and storing the application data used in the analytics framework as well as a reduction in the storage space and associated costs. In addition, the simplicity of JSON makes it easy to transpose object structures from almost any programming language, making it possible to flexibly manage data both online and offline, similarly to the platform described in [10].

The Analytics layer. This layer provides a dedicated environment where business logic related to data analysis and data mining takes place. The main goal is to deploy software components that are suitable for remote invocation, and in such a way that they encapsulate layer-specific logic, by controlling transactions and coordinating responses in the implementation of the layer's operations. In order to implement analytic components,

we have chosen the R programming language, and Rserve [11] as the technology to support a neat linkage with the *Application* layer, so that the latter may use the services offered by the former. The R technology is chosen because of its well-known support for complex descriptive analytics, through the provision of mechanisms for multicore task distribution, readily usable tests, and many libraries, including specific technologies such as Spark[1], TensorFlow [12], and ProActive [13]. R has also become an important development tool for numerical analysis and machine learning.

The Application layer. This layer provides a dedicated environment implementing logic related to service orchestration. The software components residing in this layer implement the *Command* pattern [14] which will allow the *Presentation* layer, and in fact any other client, to make requests to unspecified business components. Thus, the key participants in this pattern include an abstract *Command class* which declares an interface for executing behavior and *Command subclasses* each of which uses a well-defined REST API[2] for external communication, and implements the specific behavior needed for invoking the actual request on an *Online Service* at the *Analytics* layer. We have chosen Java in order to implement these components, and Jersey as the technology to support a neat link with the *Presentation* layer.

The Presentation layer. This layer displays actionable information to end users. It comprises components needed to display visual content as well as to capture and manage external interactions. Data exchange with the *Application* layer occurs through its well-defined REST API using the JSON data format. Depending on specific use cases associated to data visualization, components in this layer will need to implement suitable methods for processing and presenting results to end users.

3 Deployment on the Cloud

Elastic computing is considered one of the central elements of the cloud paradigm. The term elastic has its origins in physics, where the elasticity of a material is the ability to return to its original state after deformation. In cloud computing, elastic computing is the ability to adapt to workload changes by scaling up and scaling down computing resources automatically, in such a way that at a given point in time the available computing resources match current demand [15, 16, 17]. Thus, elastic computing yields significant cost savings compared to the traditional cloud infrastructure since organizations simply rent computing power on demand as the workload changes. The Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing power in the Amazon Web Services (AWS) cloud platform. One of the advantages to using the former is the elimination of the need to invest in hardware and systems administration up front, allowing applications to be developed, tested and deployed faster. Amazon EC2 can be used to launch as many or as few virtual servers as needed, configure security and net-

[1] https://spark.apache.org/docs/latest/sparkr.html
[2] https://restfulapi.net/

working, and manage storage. In particular, AWS enables users to create tailored *Amazon Machine Images* (AMIs) to quickly and easily start instances customized with everything that is needed to run applications. An *instance* is associated to a *type of instance* that, essentially, defines the hardware configuration of the host computer. In this way, Amazon EC2 provides each instance with a consistent and predictable amount of CPU and memory capacity. Thus, in order to deploy the analytics framework in the cloud, a specific AMI equipped with Apache Tomcat[3], R components and Rserve has been created as depicted in Fig. 2.

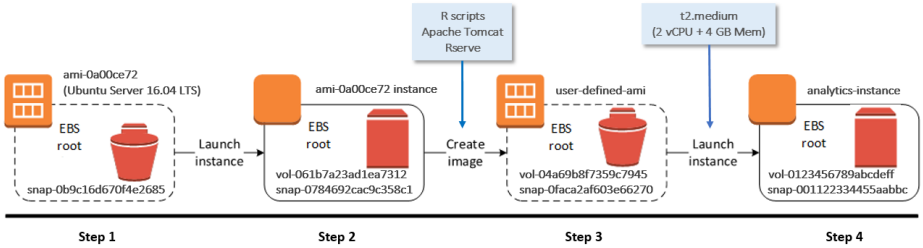

Fig. 2. The instantiation of an AMI which, after being equipped with Apache Tomcat, R components and the Rserve library (steps 1 and 2), becomes a user defined image launched and embodied in a t2.medium instance type (steps 3 and 4).

4 Deployment on Assembly Lines

We envision the instantiation of our analytics framework on at least two assembly lines located at the University of Nottingham. The first of these is the Precision Assembly Demonstrator (PAD) [18], which assembles detent hinges for the interiors of certain vehicle models. The PAD comprises six modular stations: two KUKA six-axis robotic arms with corresponding workspaces, a shared tool changing rack, a testing station, and a shuttle transport system, which links the stations to each other and a loading/unloading station (see Fig. 3). The parts to be assembled into a product are mounted on a pallet and placed on the conveyer, which moves the pallet to each of the stations. When the pallet reaches a robotic arm, it removes the individual parts and performs various assembly operations on them before returning the resulting partial-product back to the pallet. Data that could be collected from these stations includes the specification (e.g. as a sequence of parameterized assembly operations) of the product being assembled, the current layout of the assembly line, the specific operations performed (e.g. picking and gripping), the tools used to perform them (e.g. a gripper and suction tool), the parts on which the operations were performed (e.g. a spring and a ball-bearing), and the parameters used (e.g. pressure and depth of cut). The testing station can additionally store 2D images of the product being assembled, which are taken to check whether the product is being correctly assembled. This station can also store the force applied when

[3] http://tomcat.apache.org

testing hinges, and which ones pass/fail the tests. All data collected can be associated with specific hinges, as they have serial numbers.

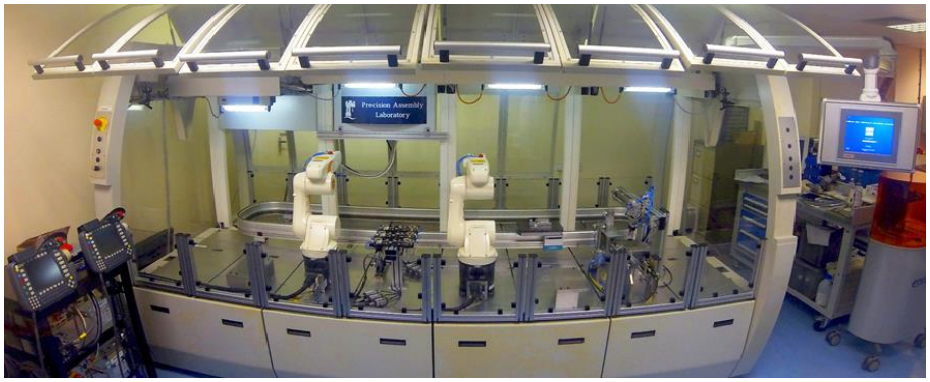

Fig. 3. The PAD assembly system [18] comprising two KUKA six-axis robotic arms with corresponding workspaces, a shared tool changing rack, a testing station, and a shuttle transport system, which links the stations to each other and a loading/unloading station.

The second assembly line is the SMC Pneumatics HAS-200 platform, which has been used to mimic the production of customized pharmaceuticals [19]. The SMC has 8 modular stations connected in a ring-shaped topology by default, which is manually adapted as necessary (see Fig. 4, bottom left). Stations operate as follows. The first loads empty containers ('pills') onto the conveyor belt as customized requests for pills are received by the assembly line. The next three stations are blue, red and yellow particulate ('ingredient') dispensers, respectively, which pick a container, add the required quantity of particulate by weight, and return the container to the belt. The next two stations are testing stations, which measure the absolute quantity of particulate in the container. These stations connect to one that puts a lid on a container, prints a custom label, and sticks it on the lid. Finally, a palletization station removes completed containers for packaging. Data that could be collected from these stations include the layout and operations as before, the weight of each type of particulate added to each container, and the total volume of particulate in each container. All data can be uniquely associated with containers as each of them has a barcode.

5 Real-Time Monitoring of Assembly Lines

Both the PAD and the SMC assembly lines are used in an environment where their layouts or *topologies*, products, and processes can 'evolve'[19]. This can be reflected by the variety, velocity and volume of collected data which could potentially vary over the course of assembly. For instance, if a new 'pill' is requested that uses a previously unused 'ingredient'; a station module is removed for maintenance, resulting in container-weight data being no longer collected; a new process is added that takes thermal images during a force test; a process is upgraded to now take 3D (instead of 2D) images;

or the assembly line's throughput is ramped up. We recognize that these scenarios may represent a real challenge, and envision the big data characterization and subsequent capture, collection, processing, organization and storage for both online and offline data management in the *Knowledge* layer being addressed by the cloud-based platform schematically presented in Fig. 4.

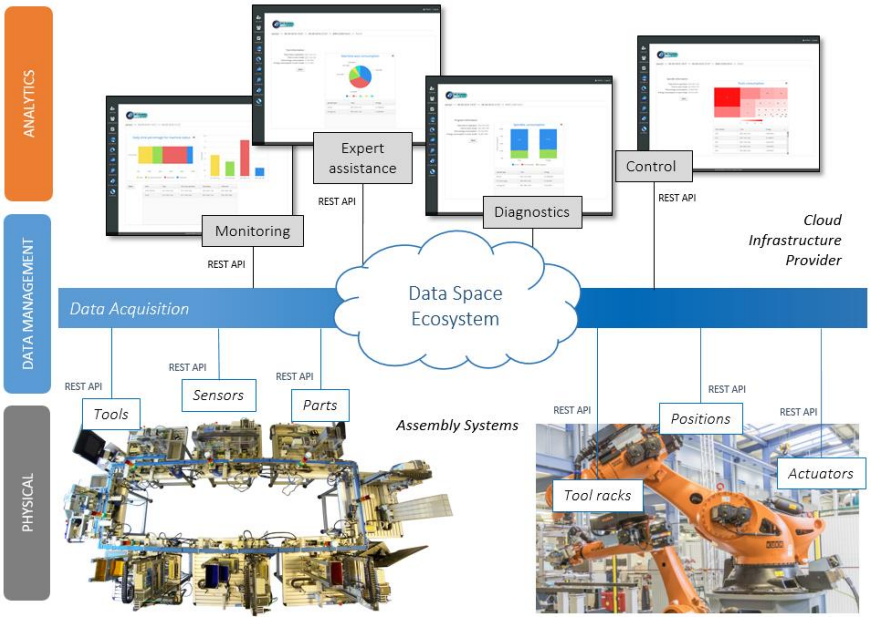

Fig. 4. Assembly lines connected to a data space ecosystem that comprises the data management and analytics framework. Shop floor generated data is collected, processed, stored and made available for monitoring, diagnosis and other types of analytics.

At the resource level, one interesting question to explore would be how transformations in the SMC's topology relate to factors such as the month of the year, the kind of product being assembled, and the availability of assembly stations. For instance, we could verify whether, during peak season, the stations connected in a 'Y' shaped topology tend to be more effective than when they are connected in a ring-shaped topology (which may cause longer queues of containers on the single conveyor belt). At the product level, we could analyze which raw material vendors (if any) tend to be associated with a significantly higher percentage of force-test failures on the PAD, which might be suggestive of lower quality raw materials. At the process level, we could enhance product quality with offline image processing of the stored hinge images in order to recognize and classify hinge defects such as surface cracks, dents and spots. The end user could be presented with analytical insights as depicted in Fig. 4, regarding whether and how such defects correlate to variations in assembly processes and parameters, e.g. in gripper-tool models or the pressure applied during assembly operations.

6 Conclusions

In this work, we have described an implemented architecture for a cloud-based analytics framework comprising different levels of abstraction and based on open source technologies. We have also introduced two different assembly lines, i.e., their layouts, products, and processes, each of which is able to evolve. In this context, we have shed light on challenges and opportunities associated with the product, process, and resource domains, particularly when analyzing large amounts of data is resource demanding in terms of storage and processing power. The authors would like to thank the support of the *EPSRC Cloud Manufacturing – Towards Resilient and Scalable High Value Manufacturing* project under grant agreement EP/K014161/1.

References

1. Monostori, L.: Cyber-physical Production Systems: Roots, Expectations and R&D Challenges. In Procedia CIRP, 17: 9–13 (2014).
2. Ahmad, M., Ahmad, B., Harrison, R., Alkan, B., Vera, D., Meredith, J., Bindel, A.: A Framework for Automatically Realizing Assembly Sequence Changes in a Virtual Manufacturing Environment. In Procedia CIRP, 50: 129–134 (2016).
3. Mourtzis, D., Doukas, M., Vlachou, A., Xanthopoulos, N.: Machine Availability Monitoring for Adaptive Holistic Scheduling: A Conceptual Framework for Mass Customization. In Procedia CIRP, 25: 406-413 (2014).
4. Tapoglou, N., Mehnen, J., Vlachou, K., Doukas, M., Milas, N., Mourtzis, D.: Cloud-Based Platform for Optimal Machining Parameter Selection Based on Function Blocks and Real-Time Monitoring. Journal of Manufacturing Science and Engineering, 137(4): 040909-040909-11 (2015).
5. Zhong, R., Xu, C., Chen, C., Huang, G.: Big Data Analytics for Physical Internet-based intelligent manufacturing shop floors. International Journal of Production Research, 55(9):2610–2621 (2017).
6. Mourtzis, D., Vlachou E., Xanthopoulos, N., Givehchi, M., Wang, L.: Cloud-based adaptive process planning considering availability and capabilities of machine tools. Journal of Manufacturing Systems, 39: 1–8 (2016).
7. Richards, M.: Software Architecture Patterns. O'Reilly Media, Inc. (2015).
8. Hashem, I., Yaqoob, I., Anuar, N., Mokhtar, S., Gani, A. and Ullah Khan, S.: The rise of "big data" on cloud computing: Review and open research issues. Information Systems, 47, 98–115 (2017).
9. Pokorny, J.: NoSQL databases: a step to database scalability in web environment. Journal of Web Information Systems 9:1, 69–82 (2017).
10. Ferry, N., Terrazas, G., Kalweit, P., Solberg, A., Ratchev, S., Weinelt, D.: Towards a Big Data Platform for Managing Machine Generated Data in the Cloud. IEEE Industrial Informatics, pp. 263–270 (2017).
11. Urbanek, S.: Rserve – A Fast Way to Provide R Functionality to Applications. In: Hornik, K, Leisch F., Zeileis A. (eds.). The 3rd International Workshop on Distributed Statistical Computing (2003).
12. TensorFlow: A System for Large-Scale Machine Learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, pp. 265–283, USENIX Association (2016).

13. ProActive Homepage, https://doc.activeeon.com/latest/admin/ProActiveAdminGuide.html, last accessed 2017/10/01.
14. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995).
15. Herbst, N.R., Kounev, S., Reussner, R.: Elasticity in Cloud Computing: What It Is, and What It Is Not. In: Proceedings of the 10th International Conference on Autonomic Computing, pp. 23–27. USENIX (2013).
16. Galante, G., de Bona, L.C.E.: A Survey on Cloud Computing Elasticity. In Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, pp. 263–270. IEEE Computer Society (2012).
17. Coutinho, E. F.; de Carvalho Sousa, F.R., Rego, P.A.L., Gomes, D.G., de Souza, J.N: Elasticity in cloud computing: a survey. Annals of Telecommunications, 70(7–8): 289–309 (2015).
18. Antzoulatos, N., Castro, E., de Silva, L., Rocha, A. D., Ratchev, S., Barata, J.: A multi-agent framework for capability-based reconfiguration of industrial assembly systems. International Journal of Production Research, 55(10): 2950–2960 (2017).
19. Chaplin, J. C., Bakker, O. J., de Silva, L., Sanderson, D., Kelly, E., Logan, B., Ratchev S. M.: Evolvable assembly systems: a distributed architecture for intelligent manufacturing. IFAC-PapersOnLine, 48(3): 2065–2070 (2015).