The International Journal of Advanced Manufacturing Technology (2020) 108:3717–3724
https://doi.org/10.1007/s00170-020-05567-5

ORIGINAL ARTICLE

Development of a hybrid DLT cloud architecture for the automated use of finite element simulation as a service for fine blanking

Joachim Stanke[1] · Martin Unterberg[1] · Daniel Trauth[1] · Thomas Bergs[1]

Received: 9 January 2020 / Accepted: 2 June 2020 / Published online: 24 June 2020

Abstract

Networking and digitization in manufacturing enable novel methods of data-driven analysis and optimization of processes through cross-process data availability. The creation of digital twins plays an important role in this. However, not all data relevant for a digital twin can be measured directly in the process. Therefore, methods are needed that enable the modelling of quantities that are difficult or impossible to measure directly in the process, such as the finite element method. In many companies, however, neither the know-how nor the necessary IT infrastructure for finite element simulations is available. External commissioning processes are also not suitable for achieving the goals of higher productivity and agility pursued with the digitization and networking of manufacturing processes. In this contribution, an architecture is presented that enables the fully automated use of finite element simulation as a service. The architecture is developed using the case study of fine blanking. First, the requirements of the architecture to be created are determined. Important characteristics of the architecture should be scalability as well as interfaces and means of payment suitable for machine communication. In addition, ensuring data integrity is an important requirement when creating the digital twin. Based on the identified requirements, an architecture is then presented that meets these requirements by using cloud computing and distributed ledger technologies and interfaces that can directly process measurement signals from the process and communicate with the architecture. Finally, the capability of the architecture is tested, possible applications and limitations are discussed, and future extensions are considered.

Keywords Fine blanking · Finite element analysis · Distributed ledger technology · Simulation as a service

Abbreviations

AAS	As a service
API	Application programming interface
AWS	Amazon Web Services
DLT	Distributed ledger technology
FE	Finite element
FEM	Finite element method
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IT	Information technology
JSON	JavaScript Object Notation
M2M	Machine to machine
MQTT	Message Queuing Telemetry Transport
NIST	National Institute of Standards and Technology
USB	Universal Serial Bus

1 Introduction

Fine blanking is a manufacturing process for the mass production of components from sheet material with particularly high-quality requirements. The main sales market for fine blanked components is the automotive industry. The current change in the automotive industry is pushing the fine blanking industry to access new sales markets [1]. Due to the pressure of competing manufacturing processes in these markets, the fine blanking industry must increase its productivity and agility in order to be more competitive. The demand for shorter development times combined with higher productivity represents a current challenge for the fine blanking industry [2]. The fine blanking process is designed according to the state of the art based on experience [3]. This approach is not suitable for mastering current challenges in fine blanking [2]. The digital transformation currently taking place in manufacturing technology

✉ Joachim Stanke
j.stanke@wzl.rwth-aachen.de

[1] Laboratory for Machine Tools and Production Engineering WZL of RWTH Aachen University, Campus-Boulevard 30, 52074 Aachen, Germany

has the potential to maximize the agility and productivity of fine blanking companies simultaneously for the first time and thus to overcome the current challenges [4]. An important component of digital transformation is the formation of digital twins. Digital twins are data-based models that describe real objects or processes with sufficient precision [5]. With digital twins of real-world objects, the state of the object over the entire life cycle can be tracked. With this knowledge, data-driven modelling of cause-and-effect relationships in fine blanking and thus novel ways of product and process design can be enabled. Additionally, digital twins enable greater transparency in production and individual optimization of products along the entire value chain [6].

However, the digital transformation also poses economic challenges for manufacturing companies and especially for the fine blanking industry. One of the biggest challenges is the lack of the necessary information technology (IT) infrastructure and the required technical personnel [7]. Not all companies have the necessary IT maturity for the digital transformation of manufacturing [8]. Geissbauer et al. carried out a study on the digital transformation of factories. The authors found that 81 % of the companies surveyed stated that finding employees with the skills required for digital transformation is the biggest challenge in human resource management [9]. The digital twin in fine blanking cannot be created from directly measurable quantities alone, since some important physical variables can only be measured with great difficulty or are not non-destructive testable. These variables can be determined using methods such as the finite element method (FEM). According to the state of the art, FEM is primarily used in the development phase to design the process and products. Alternatively, it is possible to use FEM with actual data measured in the process. This way, it is possible to determine quantities for the formation of the digital twin that are difficult to measure in the process. If the FEM is used in this way, it is referred to as a virtual sensor. However, FEM also requires powerful IT hardware and skilled personnel and thus represents an economic challenge.

One way to minimize the barriers to the usability of FEM is to provide it as a service. As a service (AAS) is a model of cloud computing from information technology. According to the National Institute of Standards and Technology (NIST), cloud computing is defined as a computing model "for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [10]. Cloud computing is motivated by the idea that data processing and storage can be carried out more efficiently by computing in large resource pools and storage systems accessible via a network [11]. The properties of cloud computing enable the reduction of infrastructure costs, minimize entry barriers, and offer scalability [12]. This efficiency results from the availability of resources in line with demand, demand-based payment, cost efficiency, and the low labor intensity of the model [11]. If this approach could be transferred to manufacturing and finite element simulation could be implemented as a service using cloud computing, a barrier towards the digital transformation of manufacturing technology and fine blanking technology could be overcome.

Helo et al. used a cloud-based implementation of a virtual supply chain for manufacturing. The authors' work showed that the cloud-based approach can yield transparency of the supply chain across multiple companies, but also that further work has to be done on the topics of data security and that domain-specific approaches are needed because of the complexity of processes and products [13]. Al-Shdifat et al. showed that cloud computing can be used to develop an efficient framework for remote monitoring services of the health status of manufacturing equipment. The authors concluded that more research must be carried out to provide architectures for efficient internet of things (IoT)–enabled monitoring services in industrial environments [14]. Taylor et al. developed a platform for the implementation of simulation software for manufacturing on cloud infrastructures [15]. The work focuses on a method for the deployment of the software itself and not on the integration of the simulation into an architecture usable by machines.

Literature review shows that the adaptation of cloud computing concepts for manufacturing potentially has great benefits. However, for the development of a finite element (FE) simulation AAS architecture which can be used fully automatically by a manufacturing machine, some challenges must be overcome: An architecture is needed which makes the FE simulation usable via a machine to machine (M2M) communication. Additionally, a method is needed that enables automated communication between a fine blanking machine and an FE simulation AAS. The architecture to be created must be able to guarantee data integrity and the chosen approach must be scalable to be economically relevant. It is currently unknown, how such an architecture can be created. The aim of this contribution is therefore the development of an architecture for the FE simulation AAS, which enables the use of the service in the form of M2M communication. The architecture is developed using the case study of fine blanking.

The work is divided into the following parts. Firstly, the requirements for the architecture to be created are defined and the required technologies are compiled. Secondly, the concept and implementation of the architecture are presented. Finally, the results are summarized and discussed.

2 Requirement analysis

The concept of a FE simulation AAS must include a monetary transaction technology to enable payment of the service, because it is supposed that the customer of the service and the provider of the service itself are different economical actors. The monetary transaction method must meet special requirements. Since the service is to be used in the form of M2M communication, the monetary transaction technology must be usable through an application programming interface (API). Furthermore, the transactions must be free of charge and must not cause any significant delay, because the ultimate goal of digitization is to increase agility and productivity. Since manufacturing data is to be considered sensitive, the security of the transaction technology must also be guaranteed. To prevent data alterations, a method to ensure and verify data integrity is also needed. These requirements can be met by using distributed ledger technologies (DLT). The purpose of DLT is the tamper-proof documentation of transactions. This is achieved by the fact that there is no central authority in a DLT that decides on the correctness of a transaction, instead a network of participants keeps track of all transactions. New transactions are only added after a consensus has been reached between the network participants. The best-known variant of DLT is the blockchain technology, like that used in Bitcoin. However, blockchain technology is not suitable for use in manufacturing since transaction costs are high and the processing time for transactions is too long. The blockchain technology does not offer scalability, i.e., the speed of the transactions and the maximum possible number of transactions cannot be increased by additional network participants. However, the concept to be developed should represent a scalable solution for manufacturing technology, so transferability to the wide range of manufacturing processes can be ensured. IOTA is an advance of blockchain technology, which avoids the disadvantages of blockchain [16]. By changing the linear structure of the blockchain to a directed acyclic graph, scalability of the maximum possible transactions per time unit is made possible [17]. Furthermore, transactions with IOTA are free of charge and have a shorter transaction duration. IOTA has been specifically developed for Internet of Things applications and is therefore particularly well suited for the development of networked manufacturing processes [16].

For the service to be used in the form of M2M communication without the involvement of personnel, the complete numerical modelling process must be automated. This can be achieved by implementing an FE model as a programmed script. The finite element analysis software Abaqus allows automated modelling in the form of scripts in the programming language Python and is therefore well suited to enable automated modelling [18].

Additionally, a standardized format is required for the transmission of data between the fine blanking machine and the FE simulation AAS. The JavaScript Object Notation (JSON) is suitable for this, because the format is very flexible and human readable, has a low overhead, and can be interpreted by all common programming languages [19]. These features make it well suited for an Internet of Things and a service-oriented architecture.

3 Architecture concept and implementation

The proposed concept can be divided into the fine blanking machine, the FE simulation server, and the service management. Figure 1 gives an overview of the FE service architecture. The fine blanking process takes place in the fine blanking machine. From the fine blanking machine, the process parameters are transmitted to the service and the service is paid for. The DLT IOTA is used to pay for the service. Service management takes over the administration of the FE simulation orders and transactions. The FE simulations are carried out on an FE simulation server. The results are persisted manipulation-proof with the help of IOTA.

The commissioning of the FE service begins with the production of the fine blanked parts. The fine blanked parts are cut in the fine blanking machine, and the process forces and kinematics of the tool are measured for each manufactured component and transmitted to an edge computer (see Fig. 2).

The measured variables are the blank holder force F_{BH}, the counter punch force F_{CP}, and the punch position (see Fig. 3). The blank holder force is measured using four Kistler type 9021A piezoelectric force sensors installed in the fine blanking tool. A force sensor of type 9031A from Kistler installed in the fine blanking tool is used to measure the counter punch force. The measured signals of the force sensors are forwarded to a charge amplifier of the company Kistler of the type LabAmp 5167A and are converted there into a 0–10 V output. The kinematics of the tool are determined with a laser position sensor from Waycon of type LAS-T5 with a 0–10 V output built into the tool. A National Instruments cDAQ 9178 with two NI 9215 modules is used for the analog-to-digital conversion of the 0–10 V output signals. The cDAQ 9178 is connected to the edge computer via a universal serial bus (USB) interface and is read out via a programmed connector. The connector uses a Node.js library that uses the C API from National Instruments called NI DAQmx. The data is transferred from the connector to a Python script for automated detection of the fine blanking strokes after publish-subscription pattern using message queuing telemetry transport (MQTT). The detection of the fine blanking strokes is carried out based

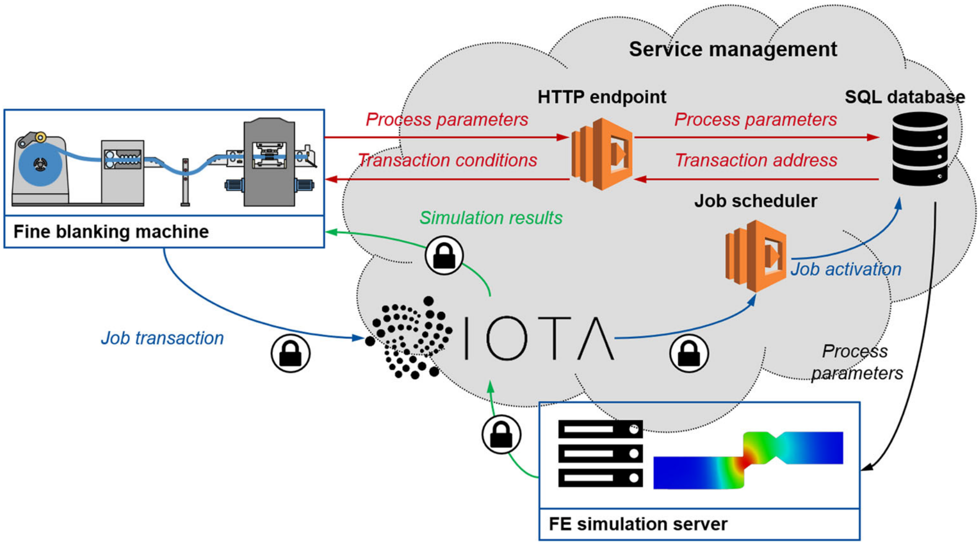

Fig. 1 Architecture concept of the FE simulation AAS

on the position measurement. The data stream is used to check when a certain threshold value has been exceeded and when it has fallen below it again. The data streams recorded in the time span between these thresholds belong to a fine blanking stroke. The size of the threshold values can be derived from known kinematic boundary conditions.

The tool geometry (V-ring height h_{V1}, h_{V2}, V-ring position a_V, die chamfer height h_D), the workpiece thickness s, and the workpiece material are entered once, as metadata via a user interface on the edge computer. If a tool or material change occurs, the changed values are entered again via the user interface. The metadata is aggregated with the measured forces and tool position to form data sets that can be used as boundary conditions for an FE simulation. The data sets are forwarded to the FE simulation AAS via a network interface using JSON format. Additionally, an IOTA address is transmitted. This IOTA address is used by the service for the result and signature transmission after completion of the FE simulation order. Thus, the results can be found and verified afterwards by the fine blanking machine.

The service management is the link between the fine blanking machine and the FE simulation server. A serverless architecture was chosen as the basis for the service management, as this architecture provides good scalability and efficient resource utilization. Amazon Web Services (AWS) was chosen as the platform for hosting the serverless applications. The interface to the FE service is a hypertext transfer protocol (HTTP) endpoint to which the process data of the fine blanking process is transmitted as a JSON file (see Fig 1). The HTTP endpoint was implemented in JavaScript for the Node.js runtime environment as an AWS Lambda function. The process data arriving at the service is stored in a SQL database. The Relational Database Service from AWS was used for this purpose. Additionally, to process the data, a status variable is set to “unpaid” in the SQL database to show that the FE simulation order has been received but is still awaiting payment. An unused IOTA transaction address for the requested FE simulation is then loaded from the SQL database and sent to the fine blanking machine via HTTP together with the amount of IOTA tokens to be transferred. The edge computer of the fine blanking machine receives this IOTA transaction address

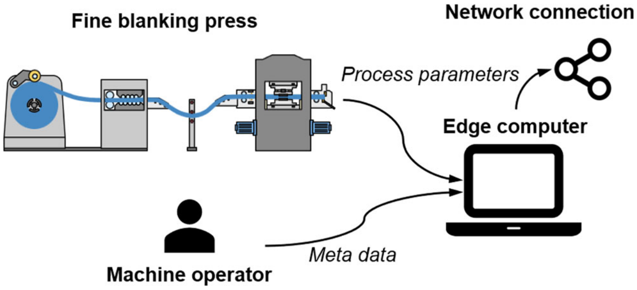

Fig. 2 Fine blanking system with service architecture integration

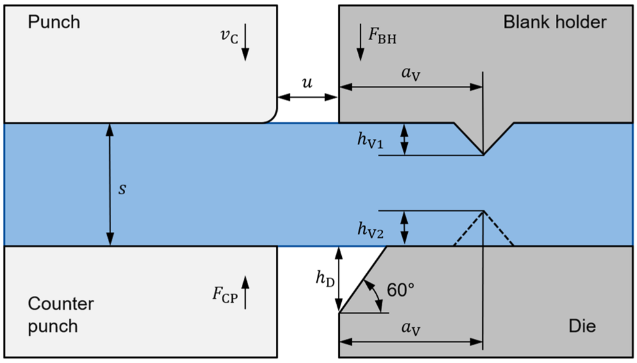

Fig. 3 Parameters of the fine blanking process: v_c: cutting speed; F_{BH}: blankholder force; F_{CP}: counter punch force; s: workpiece thickness; h_{V1}: upper V-ring height; h_{V2}: lower V-ring height; a_V: V-ring position; h_D: die chamfer; u: blanking clearance

and the amount to be transferred. The IOTA transaction is then executed using the IOTA Python API and the transmitted transaction address. The Job Scheduler of the service management checks the IOTA network for new transactions. Like the HTTP endpoint, this Job Scheduler was implemented as an AWS Lambda function. If a transaction to an open FE simulation job is found and this transaction is validated, the status variable of the job in the SQL database is set to "paid."

For the FE simulation server, a validated FE model from previous work was used. The FE simulation model was defined as rotationally symmetric (see Fig 4). The radius of the fine blanked part is variable and the fine blanking process of inner and outer contours can be simulated. For the FE simulation of fine blanking processes with cutting lines with inconstant radii, it is possible to map the process forces from the real geometry to a rotationally symmetrical analogy component. The V-ring force is calculated for this case according to:

$$F_{\mathrm{V\ analog}} = \frac{l_{\mathrm{analog}}}{l_{\mathrm{real}}} \cdot F_{\mathrm{V\ real}} \tag{1}$$

with the V-ring force of the analogy component $F_{\mathrm{V\ analog}}$, the V-ring length of the analogy component l_{analog}, the V-ring length of the real component l_{real}, and the experimentally measured V-ring force $F_{\mathrm{V\ real}}$. This calculation rule is derived from the determination of the V-ring force for fine blanking according to the state of the art. The counter punch force is calculated using the same method:

$$F_{\mathrm{CP\ analog}} = \frac{A_{\mathrm{analog}}}{A_{\mathrm{real}}} \cdot F_{\mathrm{CP\ real}} \tag{2}$$

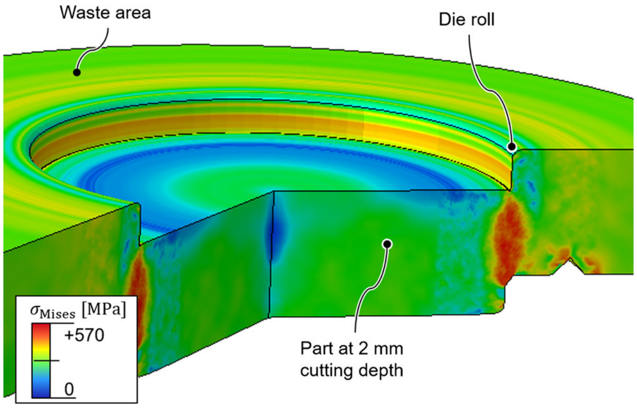

Fig. 4 FE simulation of the fine blanking process. Avg. cutting speed $v_{\mathrm{c\ avg.}}$ = 13.32 mm/s; blankholder force F_{BH} = 230 kN; counter punch force F_{CP}, 115 kN; workpiece thickness s = 6 mm; lower V-ring height h_{V2}, 0.8 mm; V-ring position a_{V}, 3.0 mm; die chamfer h_{D}, 1 mm; blanking clearance u, 0.05 mm; workpiece radius, 11 mm; workpiece material, 16MnCr5 (1.7131, AISI/SAE 5115); tool material, high speed steel

with the counter punch force of the analogy component $F_{\mathrm{CP\ analog}}$, the surface area of the analogy component A_{analog}, the surface area of the real component A_{real}, and the counter punch force measured in the process $F_{\mathrm{CP\ real}}$. For the time integration, an explicit time integration scheme has been used, since the fine blanking process is highly dynamic and non-linear. The tools of the fine blanking process have been modeled as rigid bodies, and the workpiece is modeled using an elastic-plastic material mode. To consider the influence of the heating of the workpiece due to the dissipation of forming energy during the cutting process on the material properties, a temperature-dependent material model was used and the FE simulation is temperature-coupled. An extension to three-dimensional FE simulation is also possible using the developed architecture, but outside of the scope of this work.

For the validation of the FE model 54 simulations were carried out. The cutting speed, the blankholder force, and the counter punch force were varied. The validation parameter is the die roll height, as this is the parameter which is supposed to be calculated by the FE simulation. The results can be found in Fig. 5. A full factorial design of experiments has been used. The different values of the varied parameters can be found in Table 1. The experiments are started with the lowest of the values for all parameters. Then first, the counter punch is increased one step and then another. After this, the blankholder force is increased one step and then the counter punch force is increased one step after another again. Similarly, the cutting force is increased after the blank holder force and the counter punch force are increased to their highest value. The procedure is continued until all possible combinations of the values for the counter punch force, the blankholder force, and the cutting speed have been reached. When looking at the results, a good qualitative agreement of the FE-simulation

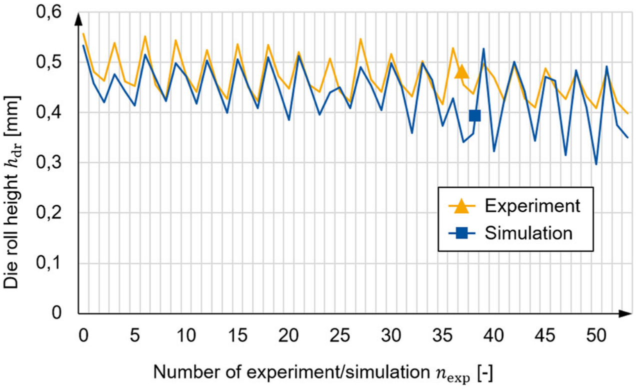

Fig. 5 Validation results for the simulation. Fixed parameters: workpiece thickness s = 6 mm; lower V-ring height h_{V2}, 0.8 mm; V-ring position a_{V}, 3.0 mm; die chamfer h_{D}, 1 mm; blanking clearance u, 0.05 mm; workpiece radius, 11 mm; workpiece material, 16MnCr5 (1.7131, AISI/SAE 5115)

Table 1 Variable parameters of the validation. $v_{\mathrm{c\,avg}}$: Avg. cutting speed; F_{BH}: blankholder force; F_{CP}: counter punch force

$v_{\mathrm{c\,avg}}$ [mm/s]	F_{BH} [kN]	F_{CP} [kN]
13.32	250	100
25.94	350	300
33.11	450	500
39.45		
53.19		
66.45		

results with the experimental results can be observed. The average deviation of the FE simulation results and the experiments is 0.035 mm and the average relative deviation is 7.6 %. Considering the qualitative and quantitative results of the validation experiments, the FE simulation results are in good agreement with the experimental results. For further information on the FE model, refer to [20]

The process parameters received by the FE simulation server are initially available in the SQL database of the service management. A controller script written in PHP runs on the FE simulation server and searches the SQL database for new FE simulation jobs with the status “paid” (see Fig. 6). If a new FE simulation job is found, the Job.php is called with the process parameters. The simulation orders are processed according to the first-in-first-out principle. The FE model is defined using an Abaqus Python script and thus can be generated automatically. The Job.php script replaces the variables in this model script with the transmitted process parameters of the FE simulation order. A command-line interface command is then used by the Job.php script to execute the model script to generate the FE model in Abaqus and create a working directory for the FE simulation. The FE model is then available in the working directory as an input file. The Job.php script then uses the command-line interface of Abaqus to start the FE simulation. After the FE simulation has been started, the status of the FE simulation job in the SQL database is updated to “in progress.” The Job.php script checks the status of the simulation at regular intervals by examining the .log file in the working directory of the FE-Simulation. The .log file is created by Abaqus and updated with status information during a FE simulation. The successful completion of a simulation is indicated in the .log file by the status “Completed.” If this status is found in the .log file, the status of the FE simulation job in the SQL database is also set to Completed. If an error occurs during FE simulation, the status of the FE simulation job is set to “Error.” The Job.php script checks in the SQL database whether FE simulations have been completed. If successful FE simulations are found, another Abaqus Python script is started to perform post processing. This post processing script automatically evaluates the FE simulation and generates results from it. These results include important key numbers of the fine blanking process as well as distributions of physical quantities such as stress, deformation and temperature. The die roll height (see Fig. 4) was chosen as the key number for the fine blanking process because it is a highly relevant quality measurement for fine blanking and important for subsequent processes while it is difficult to be measured directly in the process. The die roll height is a scalar value and stored in the SQL database. The distributions of stress, deformation, and temperature are multidimensional values and therefore stored as image files in an S3 object storage with a reference to the storage location being stored in the SQL database. This allows all results to be retrieved from the SQL database. Additionally, a signature of the results is created using the SHA256 hash function. With the signature and the calculated die roll height, an IOTA transaction with the 0 tokens is created and executed. This way, the data is stored in the IOTA DLT. As the transaction address, the address transmitted by the fine blanking machine during the order creation is used. This means that the results can be retrieved and verified by the customer via this IOTA transaction.

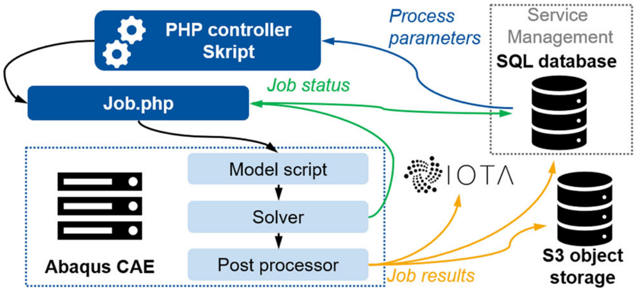

Fig. 6 Design of the FE simulation server

4 Results

To evaluate the created architecture, tests are carried out to determine the time required for the service usage and then strengths, weaknesses, and possible optimizations of the created service are discussed. The process is divided into three parts to test the time required to order the service. These parts are the parameter definition, the HTTP request, and the IOTA transaction. The parameter definition includes the process of generating a new unused IOTA address for the result transfer and creating the JSON object from the IOTA address and the process parameters for the FE simulation job. The part of the HTTP request includes the time from the fine blanking machine’s request to the FE simulation AAS to the service’s response with the transaction conditions. The last part of the IOTA transaction includes the time

Table 2 Benchmark results of the developed architecture

Experiment	1	2	3	4	5	6	7	8	Average
Parameter definition	0.654 s	0.713 s	0.622 s	0.655 s	0.743 s	0.640 s	0.667 s	0.631 s	0.666 s
HTTP request	0.718 s	1.056 s	1.045 s	1.046 s	1.055 s	1.055 s	0.637 s	0.638 s	0.906 s
IOTA transaction	1.828 s	1.944 s	1.876 s	1.916 s	1.823 s	2.087 s	1.728 s	1.763 s	1.871 s
$\sum$	3.200 s	3.713 s	3.543 s	3.617 s	3.621 s	3.782 s	3.029 s	3.032 s	3.442 s

required by the IOTA Python API to send a transaction. The duration of the FE simulation is not considered here as it is independent of the architecture created. EU-Central-1 was chosen as the region in AWS, since it is the closest to the location of the fine blanking machine. The fine blanking machine is located in Aachen. The IOTA node used for connecting to the IOTA network was also hosted by AWS. For the test of the architecture, 8 jobs were submitted to the created architecture and the required duration was measured in each case. The results are shown in Table 2. The last row indicates the average over the 8 experiments. The results do not depend on the exact transmitted parameters, but on the byte size of the transmitted data packet. The data packet size for the experiments performed was 99.3 kB. The size of the data packets is mostly independent of the transmitted parameters. The geometric parameters from the experiments can be found in Fig. 4.

The results show that the parameter definition takes about 0.7 s on average. It should be noted that most of the time needed is used for the generation of the IOTA address. One possibility to significantly reduce the duration at this point is to generate the IOTA addresses in batches and in an eternal method. The HTTP request takes on average approx. 0.9 s. This duration is mostly caused by the time the connection to the AWS server takes to start the lambda function. The actual work performed by the lambda function takes a relatively short time. The execution of the IOTA transaction takes about 1.9 s on average. The duration is completely used by the IOTA Python API. In total, the execution of an order on the Edge Computer of the fine blanking machine takes about 3.5 s and in no case more than 4 s.

5 Discussion and outlook

According to the state of the art in fine blanking, up to 2 strokes per second can be carried out on a modern servo mechanical fine blanking machine. This means that an FE simulation cannot be ordered for every fine blanking stroke with the proposed concept. However, this can be remedied by parallelization. With an average order duration of less than 4 s and a maximum of 2 strokes per second, a parallelization of 8 simultaneously running threads with the current method would be sufficient to send an order for each stroke.

One factor not considered is the duration of the FE simulation. A rotationally symmetrical fine blanking model was used as the FE model in order to require minimum simulation time. Nevertheless, the FE simulation requires approx. 1–3 h. An FE simulation of 100% of all components is therefore only possible with considerable use of resources. However, since the digital twin is used to track products along the entire value chain and life cycle, FEM can still play an important role for the digital twin. Fine blanked components pass through a number of stations until they are installed in a car, for example. There can be several hours or even days between these stations. The data of the digital twin can therefore be used at the following stations, even if a FE simulation takes several hours.

Based on the results of the developed architecture, there are two possibilities for how the FE simulation AAS can be used in production. Firstly, for the generation of the digital twin of a fine blanking component with FE simulation data, only FE simulations have to be submitted if one of the process parameters has changed significantly, as otherwise the result would be the same and therefore results from previous calculations can be reused. A second possibility is to use the data from the FE simulation together with the experimental process data to build faster real-time models using machine learning. The developed architecture can easily be extended with such an additional service by implementing a machine learning server at the place of the FE simulation server and implementing an additional HTTP endpoint that complements the existing API. The potential of this approach will be further explored in future studies.

Acknowledgements The authors would like to thank the German Research Foundation DFG for the kind support within the Cluster of Excellence "Internet of Production" (Project ID: 390621612).

Funding Information Open Access funding provided by Projekt DEAL.

References

1. Zheng Q, Zhuang X, Zhao Z (2019) Production engineering. 13(10), pp 61
2. Bauernhansl T, Hompel M, Vogel-heuser B (2014) *Industrie 4.0 in Produktion Automatisierung und Logistik: Anwendung · Technologien · Migration (Springer Fachmedien Wiesbaden)*
3. Djavanroodi F, Pirgholi A, Derakhshani S (2010) Materials and manufacturing processes. 25, pp 864. https://doi.org/10.1080/10426910903367444
4. Kagermann H, Lukas WD, Wahlster W (2011) VDI nachrichten. 13(1)
5. Kuhn T (2017) Informatik-Spektrum. 40(5), pp 440. https://doi.org/10.1007/s00287-017-1061-2
6. Schuh G, Walendzik P, Luckert M, Birkmeier M, Weber A, Blum M (2016) ZWF Zeitschrift fü,r wirtschaftlichen Fabrikbetrieb. 111(11), pp 745
7. (2015). Was kann industrie 4.0? und können sie das auch? potenziale für die deutsche industrie. Tech. rep., IBM Deutschland GmbH
8. Ghobakhloo M (2018) Manufacturing technology management. 29(6), pp 910
9. Geissbauer R, Schrauf S, Berttram P, Cheraghi F (2017) Digital factories 2020: shaping the future of manufacturing (PricewaterhouseCoopers GmbH Wirtschaftspruefungsgesellschaft)
10. Mell P, Grance T, et al. (2011) The NIST definition of cloud computing (Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology
11. Marinescu DC (2017) Cloud computing: theory and practice (Morgan Kaufmann)
12. Terrazas G, Ferry N, Ratchev S (2019) Computer industry. 109(204). https://doi.org/10.1016/j.compind.2019.03.005
13. Helo P, Shamsuzzoha A, Sandhu M (2016) International Conference on Industrial Engineering and Operations Management, Detroit, USA. pp 150-155
14. Al-Shdifat A, Emmanouilidis C (2018) Procedia manufacturing. 16, pp 31
15. Taylor S, Kiss T, Terstyanszky G, Kacsuk P, Fantini N (2014) Simulation series. 46, pp 89
16. Florea BC (2018) 2018 7th Mediterranean Conference on Embedded Computing (MECO). pp. 1–4. https://doi.org/10.1109/MECO.2018.8406041
17. Popov S (2016) The tangle. Tech rep
18. Gamble K, Pilling M, Wilson A (1995) Composite structures, 32, pp 265. https://doi.org/10.1016/0263-8223(95)00033-X
19. Wehner P, Piberger C, Gohringer D (2014). pp. 1–4. https://doi.org/10.1109/ReCoSoC.2014.6861361
20. Stanke J, Trauth D, Feuerhack A, Klocke F (2017) Journal of Physics Conferrence Series. 896, pp 012096. https://doi.org/10.1088/1742-6596/896/1/012096