This is a repository copy of *Scheduling of Smart Factories using Edge Computing and Clouds*.

White Rose Research Online URL for this paper: http://eprints.whiterose.ac.uk/138145/

Version: Accepted Version

**Proceedings Paper:**

**Reuse**
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

**Takedown**
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.
Scheduling of Smart Factories using Edge Computing and Clouds

Piotr Dziurzanski, Jerry Swan and Leandro Soares Indrusiak

Abstract—Reconfiguration-as-a-Service is an emerging trend for dynamic smart factories. This approach exploits cloud-based services to continuously optimise the performance of manufacturing systems. The edge computing paradigm, on the contrary, aims at performing the whole computation at the edge of the network, close to the data sources. In this paper, a trade-off between these two possibilities is analysed. A value-based criterion is proposed for executing optimisation engine either in a cloud or at the edge. Experimental results determine the ranges for both the cloud computation cost and the edge computer’s speed in which manufacturing scheduling leads to higher profits.

I. INTRODUCTION

One of the emerging trends related to smart factories is to migrate some computational tasks (e.g. scheduling of manufacturing processes) from remote clouds in order to be closer to devices that are the source and/or target of such computing, i.e. to the edge between the IoT’s things and the network [1]. One of the key predictions discussed in the IDC report [2] was that in the near future almost half of IoT-created data will be stored, processed, analysed and acted upon at or close to the network edge. This migration is expected to be beneficial in terms of response time, reliability, security and cost effectiveness [3]. It may however be argued that whether a certain computation is to be performed in a cloud or at the network edge should be a dynamic decision. That is, it should be based on the predicted gains and costs of both situational alternatives, rather than decided statically without any situational awareness. In Ismail et al [4], Docker containers were proposed to be executed at the edge. The same containers can be executed in a cloud as well (e.g. by using IBM Cloud Functions) even in the case of different os/architecture combinations (thanks to the experimental Docker feature named Docker manifest). Therefore, the decision on where to execute a certain container can be made dynamically considering the current edge node utilisation or network bandwidth, and also taking into account the urgency of the computation. In most cases, for efficiency and security reasons it may be beneficial to start computation at the edge, since it decreases network traffic and avoids public/shared networks and servers. However, if the computation performed at the edge progresses too slowly, it can be migrated to a cloud. Such approach requires a method to compare the predicted execution time in both edge and cloud. In this paper we follow a method that predicts the benefits of further manufacturing optimisation proposed in ref. [5]. According to that method, each manufacturing order, which requests the production of a particular commodity, is equipped with a value curve, that models the value, expressed in the monetary units, yielded by the manufacturing order over time. This value curve can influence the optimisation process as follows: since further search-based schedule optimisation is occupied with the cost of the cloud nodes performing the computation, it has been shown in that paper that it may be beneficial (grounded in terms of overall monetary cost) to prematurely stop the optimisation and apply the best results found so far. In this paper, we propose to extend that model with the possibility of performing the optimisation at the network edge. Using the proposed technique, on obtaining a manufacturing order, an agent decides not only when to finish the optimisation process, but also whether the computation should be performed at the edge or in a cloud, comparing the predicted monetary gains for all these options.

II. PLATFORM AND APPLICATION MODEL

The class of scheduling problems analysed in this paper concerns manufacturing in which the value gained by an end-user depends on both optimisation solution quality and the time taken by the optimisation process itself. The optimisation process is performed either at the network edge or in a cloud. Suitable application and platform models are proposed below.

A. Platform model

At the network edge, there is a set of \( k \) computing nodes \( N = \{N_1, \ldots, N_k\} \) capable of executing one or more containers (i.e. each node runs a typical Docker container engine). The nodes are heterogeneous and their response time difference is expressed with so-called calibration coefficient \( \zeta_x, x \in \{1, \ldots, k\} \). \( \zeta_x \) denotes the ratio between empirically measured response time of a set of container benchmarks on node \( N_x \) and the averaged response time of the same set of container benchmarks executed on a reference unit in
a public cloud serving as an alternative execution platform, as shown in Fig. 1 (the details of this figure are explained later in this paper). The benchmarks’ response times on a reference unit include the communication cost and the container initialisation time.

In ref. [5], it was assumed that the schedule optimisation engine was containerised and executed in a public cloud using a function as a service facility, which significantly reduced the initialisation time and monetary cost, in comparison with the more prevalent Containers as a Service paradigm. However, such containers can also be executed at the edge of the network, potentially decreasing the optimisation cost. Only when it is predicted that further local optimisation at the edge is likely to be less beneficial than remote execution, are the containers migrated and executed in the cloud.

B. Application model

The application considered in this paper is related to manufacturing scheduling optimisation in a smart factory. At time instants not known a priori, manufacturing orders are submitted. Each of these orders usually concerns the production of several items of a certain product. The role of optimisation is to allocate the manufacturing processes (such as mixing powders, cutting parts etc.) to different machines, select the most appropriate machine modes (e.g. thereby trading production time against energy efficiency) and schedule these processes in time, following the dependency relation between these processes. As discussed for example in ref. [5], such optimisation problems are NP-hard and thus various search-based heuristics are usually applied to find an approximate solution.

Each optimisation process is performed dynamically and concurrently to the manufacturing of the previous orders. Consequently, optimisation results must typically be provided within a limited time span. As long as the factory is busy with the previous orders, the optimisation time does not matter. However, in the case of an idle factory, the time spent on optimisation incurs ongoing factory maintenance costs due to idleness. This phenomenon is well illustrated with the value curve presented in Fig. 2. At time instant \( AT \) a manufacturing order is submitted. The maximal possible profit from this order is equal to \( V_{\text{max}} \), defined as the excess of revenue over cost and denoted in monetary units. As the factory is busy up to time instant \( D \), processing orders submitted and scheduled earlier, the profit value does not change in interval \([AT, D)\). However, after \( D \), the profit value decreases up to a certain point \( Z \), where it reaches 0. If the optimisation process ends at time \( ET \), the maximal potential profit cannot exceed the value of the curve at \( ET \), namely \( VC(ET) \).

III. OPTIMISATION TRADE-OFFS

The scheduling optimisation is performed in a master-slave fashion as illustrated in Fig. 3, in sequential stages indexed with \( i = 1, 2, \ldots \). At each \( i \)-th stage, a set of \( p_i \) containers is executed in parallel by slave nodes. The global master coordinates the execution of containers submitted by the users. The master is responsible for serving the incoming requests and allocates the containers to nodes, for example using the algorithms proposed in ref. [6].

All containers \( S_{i,y} \), \( y \in \{1, \ldots, p_i\} \), are executed either in a cloud or at the network edge. Each container gets the encoded manufacturing order together with the best solutions found so far as its input and after time \( t_{i,y} \) returns the minimal value found by the optimisation for the manufacturing cost of that order, \( f_{i,y} \), together with the corresponding solution.

If \( S_{i,y} \) is executed on edge node \( N_x \), its CPU time slot is proportional to the so-called CPU shares \( \xi_{i,y} \in \{1, \ldots, 1024\} \) (the value of the maximum share is taken directly from the Docker’s –cpu-shares flag). Assuming that the sum of all the CPU shares of containers executed on node \( N_x \) equals \( \Xi_x \), container \( S_{i,y} \) gets \( \theta_{i,y,x} = \xi_{i,y}/\Xi_x \) of the CPU time of node \( N_x \).

Initially, the execution time of the containers \( S_{i,y} \) is difficult to be predict accurately. However, as all these containers are constructed from the same container image and perform optimisation of the same problem size, the workload inside these containers is similar. Thus the response time \( t_{i,y} \) of each container \( S_{i,y} \) can be measured and used by the master node to predict the future response times at the following stages, as described subsequently.

Due to the change of a potential maximal profit from a certain manufacturing order over time as described by a value curve, a clear trade-off between the optimisation time and the optimisation quality can be identified. As a search-based heuristic keeps the best result found so far and continuously explores the search space up to the fulfilment of a certain stopping criterion, it can provide a sub-optimal result at any time. For example, ref. [5] proposed that for the master-slave architecture introduced earlier (Fig. 3), after the \( i \)-th stage the master node gathered the optimisation results \( f_{i,y} \) from containers \( S_{i,y} \) and decided if the continuation of the optimisation process, i.e. triggering the next optimisation stage, was likely to be beneficial considering the given value curve. A similar approach is applied in this paper.

![Fig. 2. An example value curve of manufacturing order O](image)

![Fig. 3. Stages of the optimisation process](image)
Performing the optimisation at the edge is assumed to cost nothing in terms of money as the edge devices are owned by the smart factory and their idle time can be viewed as wasted. This is in contrast to the optimisation cost in a cloud, which for any \( i \)-th stage is nonzero and upperbounded with \( \beta \cdot t_i \cdot p_i \), where \( \beta \) is the cost of a single container execution per one time unit\(^2\), \( p_i \) is the number of slaves executed at the \( i \)-th stage and \( t_i = \max_{y \in \{1, \ldots, p_i\}}(t_{i,y}) \) (see Fig. 4). The execution cost in both these locations can be described with equation

\[
c_i = \Delta_i \cdot (\beta \cdot t_i \cdot p_i),
\]

where \( \Delta_i \) equals 1 if the \( i \)-th stage is executed in a cloud or 0 otherwise. Using these notations, the manufacturing profit yielded by the best solution found in the \( i \)-the stage is described with equation

\[
P_i = VC \left( \sum_{j=1}^{i} t_j \right) - \sum_{j=1}^{i} c_j - f_i,
\]

where \( f_i = \min_{y \in \{1, \ldots, p_i\}}(f_{i,y}) \).

After finishing the optimisation process at stage \( i \), the values of \( t_{i+1} \) and \( f_{i+1} \) can be predicted via extrapolation, for example using the Bluirsch and Stoer algorithm \cite{7}. For history lengths of 3 or less, such extrapolation is either undefined or else the result was empirically determined to be inaccurate: the predicted value of \( f_i \) is then given by the best fitness found so far and that of \( t_i \) by the last (actual) processing time.

If the following, \((i+1)\)-th stage is processed at the edge, value \( \hat{t}_{e_{i+1}} \) predicts its execution time and \( \hat{f}_{i+1} \) predicts the lowest value returned by the slaves. Both these values can be used to predict the profit generated at the edge after the subsequent stage as follows

\[
\hat{P}_{e_{i+1}} = VC \left( \sum_{j=1}^{i} t_j + \hat{t}_{e_{i+1}} \right) - \hat{f}_{i+1} - \sum_{j=1}^{i} c_j.
\]

Let us assume that at the \( i \)-th stage, executed at the edge, the longest computation (lasting \( t_i \)) has been performed by the \( x \)-th node with calibration coefficient \( \zeta_x \) and whose fraction of CPU time for the related container equals \( \vartheta_{i,y,x} \). This container is predicted to be executed for \( \hat{t}_{c_{i+1}} \) in the following stage if executed at the edge. Then the execution time in a cloud of the same container can be assessed with formula

\[
\hat{t}_{c_{i+1}} = \frac{\hat{t}_{e_{i+1}}}{\zeta_x \cdot \vartheta_{i,y,x}}
\]

Then the profit generated after the subsequent stage executed in a cloud can be estimated with equation

\[
\hat{P}_{c_{i+1}} = VC \left( \sum_{j=1}^{i} t_j + \hat{t}_{c_{i+1}} \right) - \hat{f}_{i+1} - \sum_{j=1}^{i} c_j - \hat{c}_{i+1}.
\]

If the current, \( i \)-th stage is executed in a cloud, the execution time of the following stage at the edge can be assessed with equation

\[
\hat{t}_{c_{i+1}} = \hat{t}_{e_{i+1}} \cdot \zeta_x \cdot \vartheta_{i,y,x},
\]

and substituted to equation (3) to estimate the corresponding profit. Value \( \hat{t}_{c_{i+1}} \) is also used to estimate \( \hat{c}_{i+1} \) using equation (1).

The stopping criteria are evaluated by the master node after each stage \( i \). The predicted profit criterion checks the prediction if the execution of the subsequent stage is likely to increase the profit generated by the optimised process or not, regardless it is executed in a cloud or at the edge

\[
P_i > \max(\hat{P}_{e_{i+1}}, \hat{P}_{c_{i+1}}).
\]

Moreover, if \( \hat{P}_{e_{i+1}} \geq \hat{P}_{c_{i+1}} \), the following stage should be executed at the edge. Otherwise, the containers shall be executed in a cloud.

The benefits of similar stopping criteria in a cloud environment has previously been evaluated \cite{5}. In the following section, we apply this approach to a platform consisting of both edge machines and a cloud.

IV. EXPERIMENTAL RESULTS

The edge execution platform described above has been implemented and used together with the original Docker engine in form of two software modules, namely DockerManager and DockerWorker. The former one corresponds to the master node and is run on a machine where Docker may or may not be installed, whereas the latter, executing the slave nodes, requires the presence of the Docker daemon. These modules are depicted in Fig. 1.

In order to evaluate the technique described in this paper, 30 manufacturing orders considered in ref. \cite{5} have been selected to be scheduled in a certain factory. In that factory, there are 8 machine types and each machine can operate in different operating modes, influencing both the processing time and the consumed energy, which in turn influences the manufacturing cost. The number of manufacturing process steps in these orders ranged from 18 to 59. Each of these steps needs to be allocated to a machine operating in a certain mode. The parameters for the associated value curve are \( AT = 0, D = 250 \text{ s}, Z = 500 \text{ s} \) and \( V_{\text{max}} = 5000 \text{ GBP} \), which means that such amount of money would be gained by a plant if both the production and the scheduling cost nothing.

In the first experiment, the migration between cloud and edge computation has been disabled and both these environments have been used for the first stage. The computation

\footnote{For example \( \beta = 0.000017 \text{ USD per second of execution per GB of memory allocated using IBM Functions in August 2018.} \)}
The migration has been observed. For all the considered cases, the time on average. For faster edge (ζ > 1.0), the migration from the edge to cloud is occupied with a minimal overhead. According to the experimental results, optimisation at the edge leads to a slightly better overall profit, and in case of slow or busy edge computers, the possibility of container migration to cloud decreases the computation speed gap between a cloud and edge. Since using the proposed approach leads to comparable if not better profits than optimisation solely in a cloud, considering the additional benefits form edge execution, such as reduction of outbound/inbound network traffic, increased reliability and security, edge platform can be viewed as a promising alternative to cloud computing even for computationally costly tasks.

ACKNOWLEDGEMENT

The authors acknowledge the support of the EU H2020 SAFIRE project (Ref. 723634).

REFERENCES


