A CBR MAC/FAC Approach for Cloud Management

Eric Kübler, Miriam Herold and Mirjam Minor
Goethe University, Frankfurt am Main 60325, Germany

Keywords: BPMN, CBR, Cloud Management.

Abstract: In modern working world, efficiency is important. This includes the execution of workflows. Cloud computing offers is costumers, pay as you go pricing models which can be used to improve the cost efficiency of workflows execution. To benefit form this pricing model, a smart cloud management concept to assign tasks to cloud resources is required. Simple solutions struggle with over-, and under-provisioning problems or lack the needed flexibility. In this paper we present our CBR approach with MAC/FAC similarity function and experiments for task placements in cloud computing. Therefore we implemented a prototype and compared the results with the results of human experts.

1 INTRODUCTION

In todays working world, efficiency is a key element for success. Cloud computing offers nearly infinite resources on-demand on a pay as you go pricing model (Mell and Grance, 2011). Cloud computing can help to improve the cost efficiency of a company by reducing their applications' demand for buying and hosting hardware. However, a smart approach for cloud management is necessary to handle the cloud resource effective. Further, the working processes can be optimized to save resources. A way to achieve this goal is the use of workflows that organize the processes. A workflow is defined by the Workflow Management Coalition (Workflow Management Coalition, 1999) as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules". A task, also called activity is defined as follows: "A process [...] consists of one or more activities, each comprising a logical, self-contained unit of work within the process. An activity represents work, which will be performed by a combination of resource (specified by participant assignment) and/or computer applications (specified by application assignment)" (Workflow Management Coalition, 1999, p. 14). The combination of cost efficiency and process efficiency is a logical step that has led to a new business model. This model is known under different names: Workflow as a Service (WFaaS) (Cushing et al., 2012), Business Process as a Service (BPaaS) (Accorsi, 2011) or Business Integration as a Service (BIaS) (Chang et al., 2012). The basic idea that is common for the three models is to execute workflows in a cloud. The execution of a workflow in a cloud means that the cloud provides the workflow with resources and software that are required to complete the tasks of the workflow. While BIaS targets on integrating existing cloud services, WFaaS and BPaaS take a workflow as a starting point. BPaaS puts a slightly stronger focus on integrating human work than WFaaS. Gartner define BPaaS as "the delivery of business process outsourcing (BPO) services that are sourced from the cloud and constructed for multitenancy. Services are often automated, and where human process actors are required, there is no overtly dedicated labor pool per client. The pricing models are consumption-based or subscription-based commercial terms. As a cloud service, the BPaaS model is accessed via Internet-based technologies." (Gartner, 2019). An example for BPaaS is the provision of employee surveys including virtual machines with an installed opinion polling and analytics software with human actors who design a survey. An example for WFaaS is the deployment of a web server with a certain web service that renders images for a scenario such as automatical summarisation of video surveillance data. Each task has specific requirements that need to be fulfilled. For instance, a task that renders images has a high demand for CPU power, but less for disk storage and requires a web services that renders images. In this work, a cloud node denotes a set of (virtual) hardware and software. The assignment of a task to a

Kübler, E., Herold, M. and Minor, M.
A CBR MAC/FAC Approach for Cloud Management.
DOI: 10.5220/0008165001200129
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 120-129
ISBN: 978-989-758-382-7

fitting cloud node is called a task placement. It is not efficient, to start a new cloud node for each new task. On the other hand, the optimization of a task placement can be reduced to the bin packing problem if the tasks should be assigned to a finite set of cloud nodes. Consequently, a smart approach is required to decide whether and when a new node is required or whether a task can be assigned to an existing node. It is challenging to find a feasible solution that avoids over- and under-provisioning and matches all requirements. In addition, time restrictions might occur for tasks that have a time-out timer or where humans are involved who become frustrated if they have to wait for too long. It is therefore essential that the approach is fast, which disqualifies long running optimization algorithms. In this paper, we investigate whether Case-based Reasoning (CBR) provides a feasible solution for the task placement problem. The idea of CBR is that similar problems have similar solutions (Aamodt and Plaza, 1994). The approach of using CBR for workflow execution in a cloud is not new. One of the earliest works in this field is from Maurer et al. (Maurer et al., 2013). They apply CBR to implement automatic cloud management. Aamodt and Plaza describe a case as follows: "In CBR terminology, a case usually denotes a problem situation. A previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems" (Aamodt and Plaza, 1994). A problem situation in our approach is a task placement with over- or under-provisioning problems, missing requirements, or unused resources. A case comprises a task placement with problems, such as violated Service Level Agreements (SLAs), missing web services or unused resources. A case also comprise a solution that contains another task placement that solves the problems from the previous task placement. A sample solution is a newly started VM with the missing web service, the shutdown of the unused resources or the increasing of the resources to avoid future SLA violations. We introduce a MAC/FAC approach for case retrieval. MAC/FAC is a synonym for "many are called, few are chosen". The basic idea of MAC/FAC (Forbus et al., 1995) is to collect in a first (fast) step promising candidates and determine their exact similarity values in a second step. The advantage of this approach is that the number of cases for which the similarity function has to be calculated can be reduced. In a workshop paper (Kübler and Minor, 2019), we discussed which problems can occur and propose a MAC/FAC similarity function. In this work we empirically investigate the feasibility the similarity function for task placements. Now we have implemented the MAC/FAC function, different workflows from the music mastering domain, executed some instances of the workflows and added (based on the real measured values) some artificially created cases to the case base. We report on experiments with two human experts who ranked the usefulness of the solutions. We compare the results with our MAC/FAC similarity approach and discuss the benefits of the MAC and the FAC component. The CBR approach will be embedded into our Workflow Cloud Framework (WFCF) which was recently implemented and tested with an rule based approach (Kübler and Minor, 2018). WFCF is a connector based integration framework to integration workflow management tools with cloud computing.

2 RELATED WORK

A wide-spread approach to compute the similarity between two graphs is to determine the edit-distance (Cushing et al., 2012; Minor et al., 2007). As shown in section 3 our case is an attributed forest (or could be a tree, if we add an underlying hardware-root-node) with labeled nodes, and therefore the edit-distance could be a possible solution. However, as shown in (Zeng et al., 2009), the determination of an edit-distance for graphs is NP-complete. Another way is to determine the similarity of complex cases as described in (Lopez De Mantaras et al., 2005). The problem of structural similarity is the high computational effort. Instead of indexing vocabularies as suggested by the authors, we use the MAC/FAC approach (Forbus et al., 1995). This should be a more flexible approach, where the main factor for similarity are the executed tasks and the problems in the case. An MAC/FAC approach for workflows was introduced in (Bergmann and Stromer, 2013). Further, as stated in (Armbrust et al., 2010), a good balance between over- and under-provisioning of resources is a challenging issue that is an important aspect for cloud computing in general (Baun et al., 2011). The simplest methods provide resources in a static way. Such systems do not adjust themselves to a new situation causing under- or over-provisioning (Shoaib and Das, 2014). There are several other works in the literature that addresses the problem of resource provisioning in the cloud with different approaches (Shoaib and Das, 2014; Pousty and Miller, 2014; Quiroz et al., 2009; Bala and Chana, 2011; Rodriguez and Buyya, 2017). However, these approaches have the problem that the provisioning is either very static, does not make use of the capabilities of a cloud, or that the approaches are not implemented yet and therefore rather theoretically. Other approaches aim at a deeper inte-

gration of clouds and workflows (Wang et al., 2014; Korambath et al., 2014). They deeply integrate workflow and cloud technology, reducing the occurrence of over-provisioning and under-provisioning. However, they strongly depend on the used cloud and workflow management tools. Therefore, they are very limited in their options to exchange either the used cloud or workflow management tool or both. This leads tho a high risk of vendor lock-in, that should be avoided.

3 SIMILARITY OF TASK PLACEMENTS

In this section we describe the structure of our cases, the MAC/FAC method and the similarity function for our cases.

3.1 Case Representation

As mentioned before, a task placement is the assignment of the currently active tasks to cloud resources. Fig. 1 gives an example. In this example, the tasks *Task*2 and *Task*3 are active, where *Task*1 is already done. Task3 is assigned to a container named *CON*2 where *Task*2 is assigned to a VM named *VM*2 and *Task*4 is assigend to *CON*3. Assigned means that *CON*2, *CON*3 and *VM*2 host the software that the tasks needs to execute, for example a web service or an Office Suite. The task calls the web service, or the user uses a remote desktop connection to work with the Office Suite.

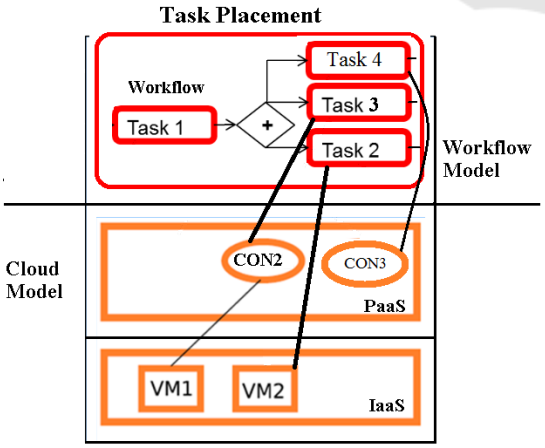

Figure 1: A simple illustration of a task placement with a task assigned to a VM and another to a VM.

3.2 Relevance of Case Parts for Retrieval

Next, we discuss what should be important for the similarity of two task placements. A task placement has plenty of parameters that could be considered for the similarity. In this work, we consider a cloud node either as a virtual machine (VM), or a container. Both have a set of resources, this can include, but is not limited to, CPU, GPU, memory, disc storage, network capacity and different kinds of installed software. For comparison, even two cloud nodes with the same set of resources (same CPU, installed software etc.) could considered as different, if the resource utilization is different. For example, a cloud node with 4GB of memory and a utilization of 100% of the memory, should be more similar to a node with 4GB memory and 90% utilization than a node with 4GB memory and a utilization of 10%. Though, a VM and a container share some attributes, it makes a difference if a node is a VM or a container. For example, a container can be migrated relatively easy from one VM to another, even if the VMs are hosted on different cloud providers. This is not so easy and sometimes even impossible for an entire VM. So the solution for a VM can not be always the migration to an other host, this is possible for a container. In this case, it is necessary to propagate the new URL or IP address to the workflow. Therefore, to compare two cloud nodes, it is important to distinguish whether it is a VM or container, to know the set of resources and the utilization of the hardware resources. The driving force of the task placements are the tasks. Without any task, there is generally no need for any cloud node. To determine the cloud resources required for a task, we introduced in one of our previous works the concept of task characteristics in cloud computing (Kübler and Minor, 2016). In short, the idea is to label tasks with its needs and give a hint of the foreseeable resource usage. In our current work, we extend the idea of characteristics so that for example the characteristic ”compute intensive” now has a value of 0 to 4 to indicate how intensive the task uses the CPU and not just a binary value of 0 or 1 to determine whether the task is compute intensive or not. Other characteristics that determine if a task is long or short running, were replaced by a values that contain the minimal, maximal and average execution time. Another important aspect should be the problems that a placement has. As mentioned before, this could be for example violations of Service Level Agreements (SLA) or the violation of internal constraints for example that there should be no cloud node active that is not in use.

3.3 MAC/FAC Approach

It is clear that this can lead to many parameters to compare for similarity. And at this point we even haven't discussed the similarity of sets of tasks and sets of cloud nodes. For performance reason, we use the MAC/FAC Principle as introduced in (Forbus et al., 1995), to distribute the effort. The idea is to collect fast a set of few promising case candidates (MAC step) and investigate the similarity of the candidates in more detail in a second step (FAC step). As mentioned before, the most important aspect for similarity are the tasks. Therefor in the MAC step we compare the currently active tasks of the problem case with the currently active tasks in cases stored in the case base. The currently active tasks is represented by a set of vectors. Since tasks are Vectors (which include the values for the task characteristics, the name and so on), the hamming distance or the euclidean distance can be used to determine similarity. But in this case, we have to consider the similarity between two sets of vectors. There are some metrics for defining the similarity between two sets, for example the Hausdorff metric (Huttenlocher et al., 1993) or the sum of minimum distances (Eiter and Mannila, 1997). The problem with these heuristics is, that it is very easy to create a case where the similarity is very high but the sets are by intuition very dissimilar. For example in one set are only equal tasks. This could be for example a set with only a single type of image rendering tasks and the other set contains different tasks but one rendering task and a mapping did map all rendering task of the first tasks to the single rendering tasks of the second set, then the similarity would be 1 for these two sets. of course, this is not desired. Another option is not to use a metric for building a single mapping, but to build all possible mappings for the vectors and chose the mapping with the minimum weight like the Kuhn-Munkers algorithm (Kuhn, 1955; Munkres, 1957). Though, this method is very compute intensive as mentioned in (Agrawal et al., 1993).

Our solution for a fast approach that is not that much vulnerable for special cases, uses an intersection of the task sets to determine the similarity. Let T_1 and T_2 be a Set of Tasks, then is the function $sim_T(T_1,T_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$. The benefit of this function is, that it covers the difference between the size of the sets, as well as the actually equal tasks. This will help in the FAC step to find more relevant task placements in a reasonable amount of time. Beside the tasks, it is important to consider the problems that a task placement has. As mentioned before, this could be SLA or constraint violations. The SLAs and constraints are stored in a a vector, are ordered by name and contains the number of violations for each SLA or constraint. We call this an SLA vector or Problem Vector. To compare two SLA vectors, we first make sure that both vectors have the same parameters, which means the same SLAs and constraints. Let $slav_1$ an SLA vector with the sla $slav_1 = SLA1_1, SlA2_1$ and $slav_2 = SlA2_2, SLA3_2$ with the values $SLA1_1 = 1, SLA2_1 = 3, SLA2_2 = 2, SLA3_2 = 1$. In a first step we add in both SLA vectors the missing parameters but set their value to 0. The new vectors are $slav'_1 = SLA1, SlA2, SLA3'$ and $slav'_2 = SLA1', SlA2, SLA3$ with $SLA1_1 = 1, SLA1'_2 = 0, SLA2_1 = 3, SLA1_2 = 2, SLA3_2 = 1, SLA3'_1 = 0$. Now we can compute the euclidean distance between two SLA vectors. Let $q_i \in SLA1$ and $p_i \in SLA2$, than is the euclidean distance for the SLA vectors $d_{sla}(p,q) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(q_i - p_i)^2}$. The similarity function is now $sim_{sla}(slav_1, slav_2) = \frac{1}{1+d_{sla}(slav_1, slav_2)}$. The MAC step is a combination of the similarity of the current active tasks, and the problems, therefore we chose an aggregated similarity function $sim_{mac} = \frac{sim(TP_1,TP_2)_T=(T_1,T_2)+sim_{sla}(slav_1,slav_2)}{2}$, where TP_1 and TP_2 are task placements with $T_1, slav_1 \in TP_1$ and $T_2, slav_2 \in TP_2$. Depending on the test results, we may add some weights to the components of the similarity function, but for now we consider the tasks and the problems as equally important.

After a fast determination of promising candidates, the FAC step compares the candidates with the current problem situation in more detail. Here is the placement of the tasks, the resources of the cloud nodes and their utilization important, as well as the question, which tasks are to be executed next.

3.4 Similarity of Tasks in Placements

As shown before in Fig. 1, a task placement can be seen as a tree, if all VMs and containers that are not related to a VM, are assigned to an abstract "hardware" node, as shown in Fig. 2. This tree is unordered and labeled, where the labels are the resources that a cloud node contains. Graph isomorphism is NP complete as well known, but to compute the edit distance between two unordered labeled trees is also NP complete, as shown in (Zhang, 1996).

The problem is not easier, if we alter the tree and add new nodes with null values for the labels, to get a more generalized structure as shown in 3.

Instead we compare the vector of tasks and their related cloud nodes. We call this a task_cloud_vector. Such a vector contains a task, a container and a VM. As mentioned earlier, it is important not to compare

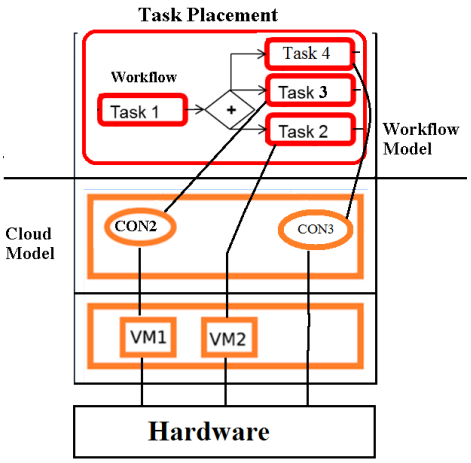

Figure 2: Task placement as tree with hardware as the root and the tasks as leafs.

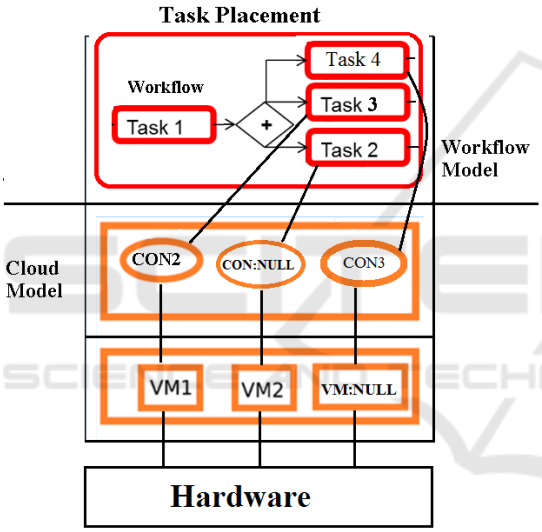

Figure 3: Task placement as tree with the hardware as root and abstract nodes for a more generalized model.

containers with VMs.

A task within the task_cloud_vector is represented by a sub-vector of parameters with their values, the characteristics and a set of tasks that could be executed next. The idea of this set is to consider the next tasks for a foresight of the workload that coming next. Therefore we use once again the euclidean distance to determine the similarity between the two characteristic vectors and the similarity for intersection for the next tasks. Let $TASK1, TASK2$ are set of tasks and $t1, t2$ are vectors of characteristics with $t1 \in TASK1, t2 \in TASK2$. The euclidean distance is then defined as $d_{tasks}(t1,t2) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t2_i - t1_i)^2}$. Let $tn1 \in TASK1, tn2 \in TASK1$ the set with the next task. The similarity function is now $sim_{ntask}(T_1,T_2) = \frac{1}{1+d_{task}(t1,t2)} + \frac{|tn1 \cap tn2|}{|tn1 \cup tn2|}$. As shown below, we need a distance function for a special algorithm. Therefore we build the distance function with $d_{ntask}(TASK1, TASK2) = |TASK1 \cup TASK2| - |TASK1 \cap TASK2|$.

A cloud node contains its hardware resources, the utilization of the hardware resources and additional software or information, which are stored as a set of tags. Let $cn = (r,u,tag)$ describe a cloud node, r is a vector of hardware resources (for example 4 cpu cores, 16GB Memory ect), u is a vector of resource utilization in percentage and tag is a set of tags (for example tag = windows8, tomcat7, jre7). For the distance between the resources, we use once again the euclidean distance $d_r(r1,r2)$, as well as for the utilization $d_u(u1,u2)$. The similarity functions is then again $sim_r(r1,r2) = \frac{1}{1+d_r(r1,r2)}$ and $sim_u(u1,u2) = \frac{1}{1+d_u(u1,u2)}$. To determine the similarity of the set of tags we build the intersection and compare it with the merged sets: $sim_{tags}(tag1,tag2) = \frac{|tag1 \cap tag2|}{|tag1 \cup tag2|}$. Similar to the next task we need a distance function for the tags: $d_{tags}(tag1,tag2) = |tag1 \cup tag2| - |tag1 \cap tag2|$.

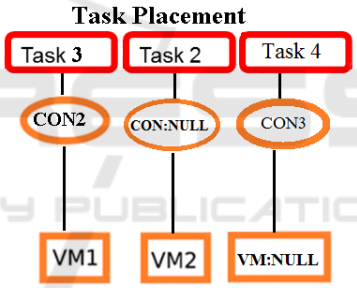

Figure 4: Task placement as paths.

The overall distance function for two task_cloud_vectors is $d_{tcv}(tcv1,tcv2) = d(tags1,tags2) + d_u(u1,u2) + d_r(r1,r2) + d_{ntask}(t1,t2)$. Similar to the MAC step we might add weights in the future.

3.5 Similarity of Entire Task Placements

After defining the similarity for two task_cloud_vectors, the next step is do define the similarity between two sets of vectors. For this we have chosen the Kuhn-Munkers algorithm (also called Hungarian algorithm) as described in (Kuhn, 1955; Munkres, 1957). This algorithm builds a minimum weight mapping for bipartite graphs, our in this case between two sets of vectors, where the edge weight is the distance between two task_cloud_vectors. In a first step a distance matrix must be built, that

contains the distance from each task_cloud_vector in the first set to each task_cloud_vector in the second set. The Kuhn-Munkers algorithm requires a square matrix. If the two sets have a different number of vectors, we add to the smaller set dummy vectors and set their edge weight to infinite. Because of the strict selection in the MAC step, based on the intersection of the tasks, there should be not many dummy vectors in our matrix. After building the matrix, the Kuhn-Munkers algorithm successively improves the mapping between both sets. To determine the similarity between two sets of vectors, after Kuhn-Munkers has finished, we build the sum of the edges between the sets is build. Since this is a distance function, the similarity function for Kuhn-Munkers is $sim_{km}(tcv1,tcv2) = 1 - \frac{1}{1+d_km(tcv1,tcv2)}$ In (Agrawal et al., 1993) is mentioned, that the run time of Kuhn-Munkers is $O(n^4)$ where n is the number of task_cloud_vectors, but that the run time can be improved to $O(n^3)$. This is very compute intensive, in particular this has to be computed for each candidate, selected during the MAC step, therefor the selection of the MAC step should be very strict.

4 EVALUATION

In this section, we discuss the evaluation of our similarity function. The experimental setup is as follows: A case base with real, measured cases is extended by artificial cases. Two human experts are employed in a leave-one-out study to create a golden standard of retrieval results. An ablation study compares the results of both, the FAC step only and the full MAC/FAC retrieval, with the case ranking from the golden standard. To achieve the experimental data, we have implemented seven workflows from the music mastering domain in jBPM citejBPM. Figure 5 depicts one of the implemented workflows. The first three tasks 'init parameters', 'generate random set', and 'choose file' are automated tasks, which don't need web services for their functionality, followed by a sequence of automated tasks that are executed by web services, and a final task that writes the resulting music data automatically to a file.

The further six workflows have at least one task that requires a web service. The *Channels_Only* workflow, for example, uses the channels web service, the *Limiter_Only* workflow uses the limiter web service. To create a set of cases for our case base we executed each workflow with real music data in a real cloud environment and recorded the workflow at an arbitrary state of execution where no SLA violation was observed. These snapshots serve as the solution

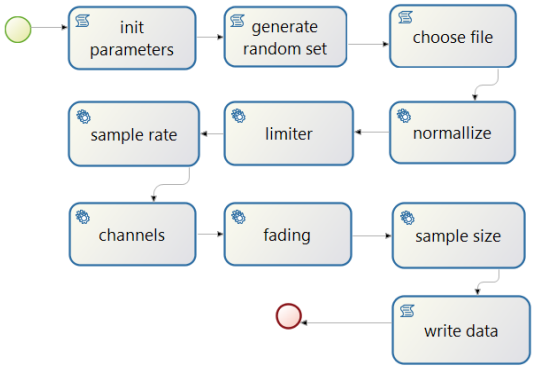

Figure 5: Workflow from the music mastering domain.

part of a case each. We extended our case base with artificial cases. Some of them are just variations of the measured cases with different resource utilization values or slight changes of the execution environment. Further cases have been created manually, to include artificial cases that do not have any similar solutions. Each case includes one to three active tasks that require a web service. In the artificial cases, every task was randomly assigned to either a container or a VM. For the real measured cases, we used the cloud platform OpenShift (OpenShift, 2019) as an execution environment. In those cases, the tasks were assigned to containers only. The overall case base comprises of 30 cases. The problem part of the cases was artificially produced from the solution part with some rules. In a first step we infused some problems to the cases, such as CPU / Memory over-provisioning / under-provisioning, Storage under-provisioning or a Missing web service. After the problem parts were configured, the solution placement was copied and manually adapted to create the problem part. If the problem was a missing web service, the cloud node for the according task was deleted. If the problem was a provisioning problem, the node was either altered to a different cloud node (container or vm) with a different set of hardware resources (either higher or lower as before depending whether the problem was an over- or under-provisioning problem), or the hardware set was just altered for the cloud node. For our experiments, two human experts selected randomly 14 of the 30 cases as queries (the problem case), and ranked the other 29 cases (solution cases) into three categories for each query.The categories were:

Very Helpful. The solution of the solution case can be applied to the problem case without any changes.

Partial Helpful. The solution of the solution case can be applied to the problem case with some adjustments (for example: starting a different web

service, or increasing / decreasing resources with a different amount).

Not Helpful. The solution of the solution case can not be applied to the problem case at all (because of different problems).

We investigated how our MAC/FAC approach performed in comparison to the human experts. In addition, we investigated the results for only applying the MAC step, or the FAC step alternatively. The aim was to get insights whether the MAC or the FAC step can run on its own to achieve similar results or whether the combination of MAC/FAC is a more feasible solution. For this experiment, we consider all cases with a similarity value of 90% or higher as very useful. For partial helpful cases we have chosen the two highest similarity values (rounded) lower than 90%. This can include several cases if some cases have achieved the same similarity values(after rounding). All other cases are considered as not helpful. Table 1 shows the results for very helpful cases.

Table 1: Results for very helpful cases for the MAC/FAC approach in comparison to the human experts.

Case Number	Human	MAC/FAC	MAC only	FAC only
case1	3	3	3	3
case2	none	none	none	none
case4	26	26	19, 24, 26	26
case5	10, 16	10, 16	10, 16	10, 16
case6	15	15	15, 17, 22, 25	15
case10	5, 16	5, 16	5, 16, 23	5, 16
case16	5, 10	5, 10	5, 10, 23	5, 10
case18	none	none	none	none
case20	none	none	none	none
case22	none	none	none	none
case25	none	none	none	none
case27	none	none	none	none
case28	none	none	none	none
case30	none	none	none	none

As one can see, if there was a perfectly fitting solution for the human experts, the algorithm always suggests the same cases. However, the MAC step only approach found some false positive very helpful cases. The MAC step only considers the tasks and the problems of the case. However, for case 4, 6, 10 and 16, the assigned nodes were different, i.e. the solutions from the solution cases 4, 6, 10 and 16 are not applicable without any changes and therefore not very helpful, only partial helpful. For the partial helpful cases the picture is quite different, as depicted in Table 2.

Table 2: Results for partial helpful cases for the MAC/FAC approach in comparison to the human experts.

Case Number	Human	MAC/FAC	MAC only	FAC only
case1	4 5 9 10 11 16 17 18 21 26	8 12	8 12	4 5 6 9 10 11 14 15 16 17 18 19 21 22 23 25 26 28 29
case2	11	5 8 9 10 11 12 16 21 23 29	2 13	5 9 10 16 21 29
case4	1 3 5 6 9 10 11 14 15 16 17 18 21	11 19	7 11 27 30	1 3 5 6 9 10 15 16 18 19 21
case5	1 3 4 6 9 11 14 15 17 18 21 23 26	12 23	2 10 12 13 20	1 3 4 6 9 15 18 21 23 26
case6	1 2 3 4 5 9 10 11 14 16 17 18 21 25 26 27 30	8 14	8 14 27 30	1 3 4 5 9 10 14 16 17 18 21 22 26
case10	1 3 4 9 11 14 15 17 18 20 21 26	2 12 13 20 23	2 12 13 20	1 3 4 6 9 15 18 21 23 26
case16	1 3 4 6 9 11 14 15 171821 23 26	12 23	2 12 13 20	1 3 4 6 9 15 18 21 23 26
case18	1 3 4 5 6 9 10 11 14 15 16 17 21 28 29	11 12 29	7 11 12 14 20 29	14 29
case20	7 14	5 7 10 11 13 14 16 18 23 28 29	2 5 10 12 16 18 23 28	18 28
case22	6 8 12 19 23 28 30	817	8 14 27 30	6 15 17
case25	27	6 8 14 15 17 22	8 14 27 30	1 3 4 5 6 9 10 14 15 16 17 18 19 21 22 26 28
case27	25	4 6 7 8 11 14 15 19 22 24 25 26	4 6 15 17 19 22 24 25 26 30	6 8 13 15 25 30
case28	12 14 18 19 23 30	11 12 14 18 20 29	7 11 12 14 20 29	1 3 4 5 6 9 10 14 15 16 18 19 20 21 25 26 29
case30	19 23 28	78 11 14 19	4 6 7 14 15 17 19 22 24 25 26	1 2 3 4 5 6 7 9 10 14 15 16 17 18 19 20 21 22 24 25 26 28 29

If the case does not provide a perfect solution, the results of the human experts and the algorithms are different. Table 3 shows the cases, that have the human experts and the algorithms in common. In all cases, the human experts have a different set of partial helpful cases than the algorithms. The MAC/FAC algorithm has in many cases fewer suggestions for partial helpful cases then the other algorithms. The most suggestions has the FAC step. As Table 4 shows, the ratio of correct suggestions to the overall number of suggestions is for the FAC only step the best. The worst is achieved by the MAC only step. The MAC/FAC approach performs slightly better than MAC only. It seems that the MAC step is the weak spot in this similarity function. The goal of the MAC step is to collect a set of promising candidates for the

FAC step, but in its current form it filters too many feasible cases.

Table 3: The partial helpful cases of the algorithms in accordance with the human experts.

Case Number	MAC/FAC	MAC only	FAC only
case1	none	none	4, 5, 9, 10, 11, 16, 17, 18, 21, 26
case2	11	none	none
case4	11	11	1, 3, 5, 6, 9, 10, 15, 16, 18, 21
case5	23	none	1, 3, 4, 6, 9, 15, 18, 21, 23, 26
case6	14	14	1, 3, 4, 5, 9, 10, 14, 16, 17, 18, 21, 26
case10	20	20	1, 3, 4, 6, 9, 15, 18, 21, 23
case16	23	none	1, 3, 4, 6, 9, 15, 18, 21, 26
case18	11, 29	11, 14, 29	14, 29
case20	7	none	none
case22	8	8	6
case25	14	14	14
case27	25	25	25
case28	12, 14, 18	12, 14	14, 18, 19
case30	19	19	19, 28

Table 4: Percentages of correct suggestions of cases for the algorithm.

	MAC/FAC	MAC only	FAC only
Cases in common	16	12	60
False positive suggestions	70	73	149
Percentage of correct suggestions	22,9%	16,4%	40,3%

To get an impression of the execution time of the particular algorithms, we created in addition to our initial case base (with 1-3 instances per case) some new case bases with 30 cases each. Each case consists of 10, 50 or 100 active tasks. Table 5 depicts the average execution time (in ms) that was taken to compute the similarity for two cases.

Table 5: Average execution time (in ms) to compute the similarity for two cases.

Number of tasks in case	MAC/FAC	MAC only	FAC only
1-3 tasks	0,04	less then 0.01	0,04
10 tasks	0,14	0,04	0,3
50 tasks	4,2	2,2	7,78
100 tasks	22,8	0,46	44,96

The results show clearly that the MAC step is very fast, even for a larger set of tasks. The FAC step seems with only 45ms for 100 tasks also relatively fast. But this is only the time to compare two cases. If the casebase contains 1000 cases, and for each case the FAC similarity has to be computed sequentially, the entire computation time sums up to around 45sec. Obviously, it depends on the number of instances per case and the number of cases in the case base. Splitting the similarity determination seems a good idea. However, to do so the MAC step should deliver a more promising set of candidates than it currently does.

5 CONCLUSION

In this paper we presented the evaluation of our MAC/FAC approach for the task placement problem in cloud computing. The results of the experiments show that the approach performs well if there is a very similar case to be retrieved. However, if there does not exist an obvious candidate in the case base the results differ from the suggestions of the human experts[1]. The MAC step in particular needs some adjustments to perform better. It seems that the focus on tasks and problems is not sufficient to find feasible candidates, the structure of complex cases is more important than expected. The FAC step on the other hand performs much better, but the execution time is as high as expected. Since the similarity for each case is independent from the other cases, this problem can be parallelised very well to improve the execution time. Still, the performance of the FAC step should be improved too. In future, we will adjust the weights of the similarity function. In particular, the part of the resource utilization should be valued less in comparison to the other parts. The next step will be to investigate if we can improve the results by changing the structure of the cases in such a way that not the entire placement but only the Task_Cloud_Vector with the problem is included. This would reduce the complexity and therefore improve the execution time and probably the result. On the other hand, if a case only contains a part of the placement, migration actions would be much more complicated. One idea to solve this problem is a dynamic case structure that depends on the degree of complexity of the problem description. Short representations of the Task_Cloud_Vector model only would be sufficient for simple problem cases while a long representation covering the entire task placement is required for rather complex cases. The results of this work highlight the feasibility of CBR for cloud management and contributes a novel use case scenario for MAC/FAC approaches.

[1]Special thanks to our human experts Bahram Salimi and Jan Höcher.

REFERENCES

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59.

Accorsi, R. (2011). Business process as a service: Chances for remote auditing. In *2011 IEEE 35th Annual Computer Software and Applications Conference Workshops*, pages 398–403. IEEE.

Agrawal, R., Faloutsos, C., and Swami, A. (1993). Efficient similarity search in sequence databases. In *International conference on foundations of data organization and algorithms*, pages 69–84. Springer.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53(4):50–58.

Bala, A. and Chana, I. (2011). A survey of various workflow scheduling algorithms in cloud environment. In *2nd National Conference on Information and Communication Technology (NCICT)*, pages 26–30. sn.

Baun, C., Kunze, M., Nimis, J., and Tai, S. (2011). *Cloud Computing - Web-Based Dynamic IT Services*. Springer.

Bergmann, R. and Stromer, A. (2013). MAC/FAC retrieval of semantic workflows. In *Proceedings of the Twenty-Sixth International International Florida Artificial Intelligence Research Society Conference*, page 6.

Chang, V., Walters, R. J., and Wills, G. (2012). Business integration as a service. *International Journal of Cloud Applications and Computing (IJCAC*, 2(1):16–40.

Cushing, R., Belloum, A. S., Korkhov, V., Vasyunin, D., Bubak, M., and Leguy, C. (2012). Workflow as a service: An approach to workflow farming. In *Proceedings of the 3rd International Workshop on Emerging Computational Methods for the Life Sciences*, ECMLS '12, pages 23–31. ACM. event-place: Delft, The Netherlands.

Eiter, T. and Mannila, H. (1997). Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133.

Forbus, K. D., Gentner, D., and Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141–205.

Gartner (2019). Gartner definition for business process as a service. https://www.gartner.com/it-glossary/business-process-as-a-service-bpaas/, 2019-04-20.

Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863.

Kübler, E. and Minor, M. (2016). Towards a case-based reasoning approach for cloud provisioning. In *CLOSER 2016 - Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy 23-25 April, 2016*, volume 2, pages 290–295. SciTePress.

Kübler, E. and Minor, M. (2018). An intelligent cloud management approach for the workflow-cloud framework WFCF. In *Proceddings of 13th International Conference of Software Technologies ICSOFT 2018*, pages 841–847.

Kübler, E. and Minor, M. (2019). Experience management for task placements in a cloud. *Accepted for publication*.

Korambath, P., Wang, J., Kumar, A., Hochstein, L., Schott, B., Graybill, R., Baldea, M., and Davis, J. (2014). Deploying kepler workflows as services on a cloud infrastructure for smart manufacturing. *Procedia Computer Science*, 29:2254–2259.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Lopez De Mantaras, R., Mcsherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M. L., Cox, M. T., Forbus, K., Keane, M., Aamodt, A., and Watson, I. (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(3):215.

Maurer, M., Brandic, I., and Sakellariou, R. (2013). Adaptive resource configuration for cloud infrastructure management. *Future Generation Computer Systems*, 29(2):472–487.

Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg*, page 7.

Minor, M., Tartakovski, A., and Bergmann, R. (2007). Representation and structure-based similarity assessment for agile workflows. In Weber, R. O. and Richter, M. M., editors, *Case-Based Reasoning Research and Development*, volume 4626, pages 224–238. Springer Berlin Heidelberg.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

OpenShift (2019). Openshift. https://www.openshift.com, 2019-04-20.

Pousty, S. and Miller, K. (2014). *Getting Started with OpenShift*. "O'Reilly Media, Inc.".

Quiroz, A., Kim, H., Parashar, M., Gnanasambandam, N., and Sharma, N. (2009). Towards autonomic workload provisioning for enterprise grids and clouds. In *Grid Computing, 2009 10th IEEE/ACM International Conference on*, pages 50–57. IEEE.

Rodriguez, M. A. and Buyya, R. (2017). A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments: Workflow scheduling algorithms for clouds. *Concurrency and Computation: Practice and Experience*, 29(8):e4041.

Shoaib, Y. and Das, O. (2014). Performance-oriented cloud provisioning: Taxonomy and survey. *CoRR*, abs/1411.5077.

Wang, J., Korambath, P., Altintas, I., Davis, J., and Crawl, D. (2014). Workflow as a service in the cloud: Architecture and scheduling algorithms. *Procedia Computer Science*, 29:546–556.

Workflow Management Coalition (1999). Workflow management coalition glossary & terminology. last access 011-23-2018.

Zeng, Z., Tung, A. K. H., Wang, J., Feng, J., and Zhou, L. (2009). Comparing stars: on approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36.

Zhang, K. (1996). A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222.