Task Placement in a Cloud with Case-based Reasoning

Eric Schulte-Zurhausen and Mirjam Minor
Institute of Informatik, Goethe University, Robert-Mayer-Str.10, Frankfurt am Main, Germany

Keywords: Workflow, Cloud Management, Task Placement.

Abstract: Moving workflow management to the cloud raises novel, exciting opportunities for rapid scalability of workflow execution. Instead of running a fixed number of workflow engines on an invariant cluster of physical machines, both physical and virtual resources can be scaled rapidly. Furthermore, the actual state of the resources gained from cloud monitoring tools can be used to schedule workload, migrate workload or conduct split and join operations for workload at run time. However, having so many options for distributing workload forms a computationally complex configuration problem which we call the *task placement problem*.
In this paper, we present a case-based framework addressing the task placement problem by interleaving workflow management and cloud management. In addition to traditional workflow and cloud management operations it provides a set of task internal operations for workload distribution.

1 INTRODUCTION

Rapid scalability is one of the most important characteristics of cloud computing according to the NIST definition (US Department of Commerce, 2011). In classical workflow management, scalability of workflow execution is achieved by traditional load balancing. A load balancer distributes the workload across multiple workflow engines in order to maintain an acceptable performance level of the workflow management system (WFMS) (Jin et al., 2001). Moving workflow management to the cloud raises novel, exciting opportunities for rapid scalability of workflow execution. Instead of running a fixed number of workflow engines on an invariant cluster of physical machines, both physical and virtual resources can be scaled rapidly. Furthermore, the actual state of the resources gained from cloud monitoring tools can be used to schedule workload, migrate workload or conduct split and join operations for workload at run time. However, having so many options for distributing workload forms a computationally complex configuration problem which we call the *task placement problem*. The Workflow Management Coalition ({Workflow Management Coalition}, 1999) defines a workflow as followed: *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: *A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources(s) to support process execution*.
In this work we use the term task configuration. A task configuration is a set of tasks with their relationships and also a set of parameter. These parameters are not needed for the tasks themselves but for the handling of the tasks through the system.
A cloud configuration is in our work a set of PMs with their resources and a set of VMs assigned to the PMs and sharing some or all of the resources owned by the PM.
We define a task placement as the assignment of automated tasks to virtual machines (VM) and furthermore the assignment of VMs to physical machines (PM). So a task placement is a mapping of a task configuration to a cloud configuration. Our task placement problem includes two kinds of problems: the job placement problem as in (Sharma et al., 2011) and a VM placement problem as in (Jiang et al., 2012). The job placement problem is the problem to assign jobs to VMs and the VM placement problem is the problem to assign VMs to PMs.
In this paper we present a case-based framework addressing the task placement problem by interleaving workflow management and cloud management. In

Schulte-Zurhausen E. and Minor M..
Task Placement in a Cloud with Case-based Reasoning.
DOI: 10.5220/0004944203230328
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 323-328
ISBN: 978-989-758-019-2

addition to traditional workflow and cloud management operations it provides a set of task internal operations for workload distribution. The notion of a case is used to represent a task placement problem. A stack of placement operators allows to organize the case representation in layers. Instead of solving the task placement in a computationally complex optimization problem, case-based reasoning (Aamodt and Plaza, 1994) is used to retrieve similar task placements from the past that can be reused in the current situation. In Section 2, we discuss related work from workflow management and cloud management. Section 3 resumes and extends a layered VM placement model from Maurer et al. (Maurer et al., 2013) that we use for a task placement model in Section 4. Section 5 describes a case-based recommender system that suggests a set of promising placement operators based on the experience stored in the case base. Section 6 addresses a planned evaluation. Finally, we discuss future work and draw some conclusions in Section 7.

2 RELATED WORK

Our definition of a task placement is different form the definition of a task placement used in Sharma et al. (Sharma et al., 2011). In their work the tasks are jobs placed on VMs, but the jobs are independent from each other. Workflow tasks are ordered and their execution order is strict. In their work they describe a task placement constraint. Not all tasks or jobs can be placed on all available VMs due to kernel, operating system or other restrictions. The task placement constraint restricts the task placement.

Tang et al. describe in their work (Tang et al., 2007) the problem to allocate applications with dynamically changing demands to VMs and how to decide how many VMs must be started. They called this problem application placement and their problem is similar to Sharma et al. (Sharma et al., 2011).

The problem to assign workflow tasks to VMs is not new. Wu et al. describe in their work (Wu et al., 2013) their approach to solve the task placement problem for scientific workflow with meta-heuristics. They called this Task-to-VM assignment and implement SwinDeW-C, a cloud workflow system. In their work (Maurer et al., 2013) Maurer et al. present an approach to manage a cloud configuration on different layers. They divide the resource allocation into three parts and called them escalation levels.

The first level is for individual applications. The possible actions are:

- Increase / Decrease incoming bandwidth share by x%
- Increase / Decrease outgoing bandwidth share by x%
- Increase / Decrease memory by x%
- Increase / Decrease CPU share by x%
- Add / Remove allocated storage by x%
- Outsource (move application) to other cloud
- Insource (accept application) from other cloud
- Migrate application to different VM

The second level is for VMs. The possible actions are:

- Increase / Decrease incoming bandwidth share by x%.
- Increase / Decrease outgoing bandwidth share by x%.
- Increase / Decrease memory by x%.
- Add / Remove allocated storage by x%.
- Increase / Decrease CPU share by x%.
- Outsource (move VM) to other cloud.
- Insource (accept VM) from other cloud.
- Migrate VM to different PM.

And the third level is for physical machines:

- Add x computing nodes.
- Remove x computing nodes.

Maurer et al. use the escalation levels to guarantee the scalability and efficient resource utilization of their system in order to fulfill the Service Level Agreements. To manage the resources they used two different types of knowledge management - a rule-based and a case-based reasoning approach.

3 MANIPULATION OPERATORS

In our approach we use a system based on the escalation levels introduced by Maurer et al. (Maurer et al., 2013). We extend this approach by so-called manipulation operators (MO) that manipulate the task and cloud configuration that will be introduced in Section 4. Furthermore, we add the workflow level on top of Maurer's escalation levels namely the application, VM and PM levels. The resulting hierarchy of levels is as follows:

- 1: Workflow level
- 2: Application level
- 3: Virtual machine level
- 4: Physical machine level

We denote the three lower levels as the resource level. The resource level includes the application, virtual and physical machine level. The workflow level supports operations to manipulate a workflow so that a better utilization of the resources can be achieved. The following workflow operations also have a hierarchical order:

- 1: Task split / join
- 2: Data split / join
- 3: Migrate task to different VM

The task split operation divides a task into a bunch of parallel tasks with different parameters. For example: I = $\{image_1,..., image_n\}$ is a set of images. Two different, mutually independent types of render algorithms $r_1(I), r_2(I)$ are given. Task $t_1(r_1(I), r_2(I))$ should perform both algorithms and task t_2 evaluates the result. A task split means that we remove t_1 and replace it with two tasks t_{11} and t_{12}, each of which performs only one render algorithm $t_{11}(r_1(I)), t_{12}(r_2(I))$. Because of the split it is now possible to run t_{11} and t_{12} on different VMs and a duration speed up is possible. The task join operation is the inverse action. With this operator the task t_{11} and t_{12} would be joined to t_1. A join can be useful to reduce the number of running VMs or to reduce the number of tasks that run on a single VM.

The data split operation replicates a task and divides the data that is to be processed. Similar to the previous example, I = $\{image_1,...$, $image_n\}$ is a list of images and task $r_1(I)$ is an algorithm to render images. The task t_1 is the task to render all images $t_1(r_1(image_1,...,image_n))$. The data split operation replicates t_1 into a bunch of parallel tasks where each task processes a subset $I\prime \subseteq I$, for example $t_{11}(r_1(images_1,...images_i))$, $t_{12}(r_1(images_{i+1},...,images_n))$. The data join operation is the inverse action similar to the task join operation. It combines t_{11} and t_{12} to t_1.

The migrate operation migrates a task between VMs. We include this operation despite of the fact that the migration of a task can be considered an application migration. The border between the application level and the workflow level is more distinct due to this decision.

4 PLACEMENT MODEL

We model the task placement problem as a mapping problem between a task configuration and a cloud configuration. A task configuration is a tuple $TC = \{T,P,F\}$. T is a set of tasks $\{t_1,t_2,...,t_n\}$. P is a set of tuples $\{p_1,p_2,...,p_n\}$ whose elements p_i describe the parameters for task t_i. The parameters for each task can be the expected execution time on a default VM for each MBit of data volume, the expected upcoming data volume, the system prerequisites, whether a task split is possible or not and if a data split is possible or not. F is a partial order $\prec$ on T specifying a precedence relation on T. F is induced by the order of tasks given in the workflow. In case that t_1 and t_2 are ordered in a sequence within the workflow, t_2 can only be started when t_1 has finished execution and, thus, (t_1,t_2) is taking part in F.

The cloud configuration CC is a tuple {PM, VM, CCP}. PM is a set of physical machines. Each physical machine is represented as a tuple *(*{incoming bandwidth, outgoing bandwidth, CPU, memory, storage, costs per time unit}). This is the representation of the hardware, offered by the physical machine. The costs per time unit is an abstract value which represents the costs, particularly energy, that occur when the machine is running. VM is, similar to PM, a set of virtual machines. Each virtual machine is represented as a tuple of properties ({incoming bandwidth, outgoing bandwidth, CPU, memory, storage, speed up}) similar to a physical machine except speed up. The speed up represents a factor that influenced the expected execution time of a task. The expected execution time is determined by a default VM configuration as follows. Let $vm_1, vm_2 \in VM$ and vm_1 is the default VM so the speed up sp_1 of vm_1 is 1. Let the quantity of the available resources for vm_2 be higher than the quantity of the available resources for vm_1. So the speed up of sp_2 is $\leq sp_1$. We consider the run time r_i of a task t_i roughly as the product of the speed up and the expected execution time, so that the run time is $r_i * sp_2 \leq r_i * sp_1$.

The cloud configuration placement CCP is a set of tuples of assignments from VMs to PMs. Each VM $vm_i \in VM$ must be assigned to exactly one $pm_j \in PM$, but a PM can contain more than one VM, for example $\{(vm_1,pm_1),(vm_2,pm_1)\}$.

Now we consider the task placement as a tuple {TC, CC, TP}. TC is the task configuration and CC is the cloud configuration. TP is a set of tuples which represent the assignment of the tasks to a VM and their specified order. TP = $\{tp_1,...,tp_i\}$ with $tp_i = \{vm_i, vmt_i, vmtf_i\}$, $vm_i \in VM, vmt \subseteq T$ denotes the set of tasks that is actually running on vm_i, and $vmtf \subseteq T \times T$ specifies the order constraints on tasks.

5 CASE-BASED TASK PLACEMENT

In case-based reasoning, a problem is solved by finding a similar case stored in the case base and reusing it in the new problem situation (Aamodt and Plaza, 1994). A case is defined as an old problem including a solution for this distinct problem. In our approach a case is a set of two task placements tp_t and tp_{t+1}. The problem task placement tp_t is the placement at time t and the solution task placement tp_{t+1} is a reconfigured version of tp_t at point $t+1$. In addition to tp_{t+1}, the solution includes a list of manipulation operators that is required to transform tp_t into tp_{t+1}. To reuse such a case means to replay the manipulation operators that have been used to transform tp_t into tp_{t+1} for the current situation, that is the current task placement. The similarity of two task placements can be determined by measuring the number of manipulation operations that is required to transform the one into the other. In addition, the expected costs, the expected performance and the expected number of SLA violations can be compared.

Aamodt and Plaza described the case-based reasoning (CBR) process as a cycle with four steps as shown in Figure 1. The four steps are:

- *Retrieve* the most similar case or cases
- *Reuse* the information and knowledge in that case to solve the problem
- *Revise* the proposed solution
- *Retain* the parts of this experience likely to be useful for future problem solving

In the retrieve phase, one or more cases with the highest similarity to the current problem are chosen. The similarity of two task placements depend mainly on the cloud configuration, the SLA requirements and the task configuration. In a first step we assume that the SLA requirements are constant. We further assume that the task configuration in a case base can always lead back to the same workflow template through the workflow manipulation operators introduced in Section 3. But the approach is not limited to that. In (Minor et al., 2007) it is shown how to define the similarity between two workflows. The idea is to represent a workflow as a graph and determine the similarity between two graphs.

With the assumption of a constant set of SLA and the workflow template we can define a neighborhood between task placements. The neighborhood between two task placements is defined by the manipulation operations. Let $TP_1 = \{TC_1, CC_1, TP_1\}$ and $TP_2 = \{TC_1, CC_1, TP_2\}$ be two different task placements with the same task and cloud configuration but

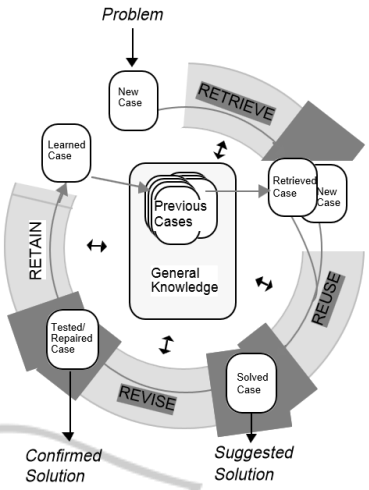

Figure 1: CBR cycle introduced by Aamodt and Plaza.

different task placements and let $\mathrm{Dis}(TP_1, TP_2)$ be a function which determines the distance between two task placements. Be $TP_1 = \{(vm_1, (t_1)), vm_2, (t_2, t_3))\}$ and $TP_2 = \{(vm_1, (t_1, t_2)), vm_2, (t_3))\}$. The distance between them is the number of manipulation operations that is required to convert TP_1 into TP_2. The difference in this example is that the task t_2 in TP_1 is running on VM vm_1 and the task t_2 in TP_2 is running on vm_2. With the operation "'*Migrate application to different VM*"' and the assumption that a task can be considered as an application, $\mathrm{Dis}(TP_1, TP_2) = 1$.

After the retrieve phase follows the reuse phase. In this phase the task placement will be modified by the manipulation operators listed in the solution part of the case. In the revise phase the new case will be evaluated and in the retain phase the new case will be stored in the case base.

6 EVALUATION OF THE CONCEPT

To evaluate the case-based approach we are planning to measure three values: the performance of the workflow execution p, the overall costs oc and the robustness ro for an experimental set of workflows. At this point, the set T includes all tasks of a workflow. That means that one task placement covers the whole workflow. But we consider that when the workflow is divided into parts of currently active tasks and builds a task placement for each of this part, the overall costs will be reduced because the needed infrastructure in terms of VMs and PMs will be very close to the actually required resources. So when the workflow is divided into several task placements we call each place-

ment a Task Placement Segment (TPS). However, the problem with the TPSs is that when there are many TPSs and they are paired very differently, it will take more time to reconfigurate the cloud and thus the performance is reduced.

The robustness in our case is an indicator for measuring the changes within a sequence of TPSs. If the difference between subsequent pairs of TPSs is high, the robustness will be low. To measure the deviation of a TPS from its predecessor, we determine the number n of needed manipulation operations to transform one cloud configuration into another, resulting in $q = \frac{1}{n+1} * 100$. The robustness value qo of the entire sequence aggregates the particular deviation values, for instance by a weighted sum with a logarithmic factor. The performance of the workflow execution depends on the expected execution time r_i of the tasks, the speed up sp_j of the VMs, the reconfiguration time wft_k of the workflow level operators and the reconfiguration time cct_k of the resource level operators of the current TPS tp_k. Each TPS includes one task placement tp_k. The performance is described as: $p = \sum_{k=1}^{n} max_{i \in tp_k}(sp_i * \sum_{j \in vmt_i} r_j) + wft_k + cct_k$. The value of the expected execution time is not just the sum of all expected executions within a TPC, because VMs execute tasks parallel so we search for the maximum makespan of the VMs which is $max_{i \in tp_k}(sp_i * \sum_{j \in vmt_i} r_j)$. In addition, the speed up factor might consider the type of the task. Some tasks could be more memory or network intensive than others which leads to a different speed up for every task depending on the task himself. As a solution, the speed up could be determined for a set of reference task. However, this would require additional effort in classifying workflow tasks.

The overall costs are the sum of the costs of the PMs cpm and the costs of the VMs cvm: $oc = cpm + cvm$. $cvm = \sum_{k=1}^{n} \sum_{i \in tp_k} c_i * \sum_{j \in vmt_i} r_j$. $\sum_{k=1}^{n}$ is the sum over all TPS, $\sum_{i \in tp_k}$ is the sum over all VMs and $c_i * \sum_{j \in vmt_i} r_j$ is the sum over all products of the expected execution r_j and the costs c_i of VM i. The costs for the PMs are similar, in short the costs of the PM pmc multiplied with the highest run time of the VM assigned to the PM. So the formula is: $cpm = \sum_{k=1}^{n} (max_{i \in tp_k}(\sum_{j \in vmt_i} r_j) * pmc)$. We are planning to measure these values to assess the target configuration that is suggested by the retrieval results.

7 SUMMARY AND FUTURE WORK

The task placement problem for workflows in a cloud environment requires an intelligent solution due to its complexity. We think the problem is more difficult than the VM placement problem because it increases the VM placement problem by an additional layer.

In this paper, we present our approach that uses case-based reasoning to find good task placements without a recalculation of the configuration on every single step of the workflow. We belief reasoning techniques are feasible and useful for task placement. Case-based reasoning is a valid method to develop a solution.

The work is still in an early phase of development. It provides a representation and a case-based solution for the task placement problem. In a next step, we will finish the implementation of the prototype and conduct an experimental evaluation. Furthermore, we will deploy a more dynamic approach to determine the changes in speed up of a virtual machine when the manipulation operators change the resources of the VM. Another issue of our future work is to determine the granularity in which the task placement segment should be chosen. To achieve a solution for this, we will implement a configurable prototype in order to conduct further experiments with different setups. The set of values to be measured might be adjusted after first experimental results have been achieved.

We expect the following benefits of the approach. The deep integration of workflow management and cloud management creates novel business opportunities for cloud providers in the area of cloud-based workflow services. Furthermore, the preference on cases with a robust configuration will hopefully reduce the re-configuration costs. Additionally, the number of SLA violations is considered by the approach and, thus, will most probably be reduced. We expect a significantly better performance of the workflow execution service in comparison to simply migrating a traditional workflow management system into a cloud infrastructure as a whole. The novel manipulation operators at workflow level facilitate both scalability at the task and at the data level.

REFERENCES

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

Jiang, J. W., Lan, T., Ha, S., Chen, M., and Chiang, M. (2012). Joint VM placement and routing for data center traffic engineering. In *INFOCOM, 2012 Proceedings IEEE*, page 28762880.

Jin, L.-j., Casati, F., Sayal, M., and Shan, M.-C. (2001). Load balancing in distributed workflow management system. In *Proceedings of the 2001 ACM symposium on Applied computing*, page 522530.

Maurer, M., Brandic, I., and Sakellariou, R. (2013). Adaptive resource configuration for cloud infrastructure

management. *Future Generation Computer Systems*, 29(2):472–487.

Minor, M., Schmalen, D., Koldehoff, A., and Bergmann, R. (2007). Structural adaptation of workflows supported by a suspension mechanism and by case-based reasoning. In Reddy, S. M., editor, *Proceedings of the 16th IEEE Internazional Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'07), June 18 - 20, 2007, Paris, France*, pages 370–375. IEEE Computer Society, Los Alamitos, California. Best Paper.

Sharma, B., Chudnovsky, V., Hellerstein, J. L., Rifaat, R., and Das, C. R. (2011). Modeling and synthesizing task placement constraints in google compute clusters. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, page 3:13:14, New York, NY, USA. ACM.

Tang, C., Steinder, M., Spreitzer, M., and Pacifici, G. (2007). A scalable application placement controller for enterprise data centers. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 331340, New York, NY, USA. ACM.

US Department of Commerce, N. (2011). Final version of NIST cloud computing definition published. Final Version of NIST Cloud Computing Definition Published.

{Workflow Management Coalition} (1999). Workflow management coalition glossary & terminology. last access 05-23-2007.

Wu, Z., Liu, X., Ni, Z., Yuan, D., and Yang, Y. (2013). A market-oriented hierarchical scheduling strategy in cloud workflow systems. *The Journal of Supercomputing*, 63(1):256–293.