COMPLEMENTARY ORIENTED ALLOCATION ALGORITHM FOR CLOUD COMPUTING

PIOTR ORZECHOWSKI

Academic Computer Centre in Gdansk (CI TASK)
Gdansk University of Technology
Narutowicza 11/12, 80-233 Gdansk, Poland

(received: 18 August 2017; revised: 18 September 2017;
accepted: 26 September 2017; published online: 10 October 2017)

Abstract: Nowadays cloud computing is one of the most popular processing models. More and more different kinds of workloads have been migrated to clouds. This trend obliges the community to design algorithms which could optimize the usage of cloud resources and be more efficient and effective. The paper proposes a new model of workload allocation which bases on the complementarity relation and analyzes it. An example of a case of use is shown and an increase in the workload execution is presented.

Keywords: cloud computing, complementary workloads, workloads allocation, IaaS cloud, resource provisioning

DOI: https://doi.org/10.17466/tq2017/21.4/u

1. Introduction

Cloud computing has become the most popular model to deploy IT solutions of any kind. Different models such as: IaaS (Infrastructure as a Service), PaaS (Platform as a Service) or SaaS (Service as a Service), available in different cloud environments, allow users to deploy their solutions in a simple way and reduce the costs of maintaining the infrastructure [1]. Users can reserve resources at any moment and they imagine that resources are unlimited. On the other hand providers have to take care of the infrastructure and its utilization. There are many methods of optimizing a virtual machine, workloads and allocation of applications which are more or less appropriate to the specific environment characteristics depending on the resource management technique [2, 3]. Different techniques could focus on various objectives such as: energy efficiency, SLA-aware, load balancing or network load minimization. Moreover, resource management could be investigated on different levels such as: application (software) level, virtualization (platform)

level and deployment (infrastructure) level. Each resource allocation method could be assigned to one layer only, but it has always impact on many layers and multiple objectives.

In this paper different methods of workload allocation are considered and a new approach using the idea of complementarity is proposed. In Section 2 cloud management strategies are shown and a new strategy is proposed. In Section 3 the idea of complementary workloads is explained and its application to the management strategy is discussed. In Section 4 an allocation algorithm is presented and an example experiment is described. Finally the pros and cons of these solutions are discussed.

2. Management approaches to workload allocation

Cloud workload allocation is a very important problem which has a direct impact on the efficiency and power usage in data centers. There are many scheduling mechanisms with different priorities and objectives to achieve [3]. A different optimization function could be found in the literature depending on the objectives which have to be achieved. In [3] the authors propose a taxonomy to organize the resource management algorithms. The paper presents a survey through different techniques and shows examples of multi-criteria optimized solutions.

A similar classification scheme of resource scheduling methods is proposed in [4]. The authors focus on different issues and approaches of resource scheduling and main performance metrics. The paper specifies six classification types focusing on the following evaluation parameters used in the studies: cost aware, energy aware, efficiency aware, load balancing aware, QoS aware and utilization aware.

The range and variety of scheduling algorithms in cloud computing is broad. Researchers use more and more complex techniques to optimize the efficiency of algorithms. For example, evolutionary computation algorithms have received increasing attention in recent years [3]. A good example could be a whole group of algorithms which use genetic algorithms [5, 6], ant colony algorithms [7, 8] and particle swarm optimization [9, 10].

There are many other examples of resource scheduling algorithms in the literature. All of them could be applied in different environments to optimize specific objectives depending on the provider requirements. Most of them are complicated algorithms and setting up could be a challenge. In the real cloud platform world some algorithms of resource allocation algorithms are provided out of the box. There are some interfaces which allow providers to extend their functionality in a simple manner. An example *IaaS* platform could be OpenStack [11] (the most popular free platform for cloud computing). OpenStack (specifically the *nova scheduler* component provides mechanism named filters). This solution selects an appropriate node for the workload by executing a chain of filters which check some resources such as: CPU, RAM, Disk, etc. Such approach, provided in OpenStack, is quite simple but has a wide range of possible

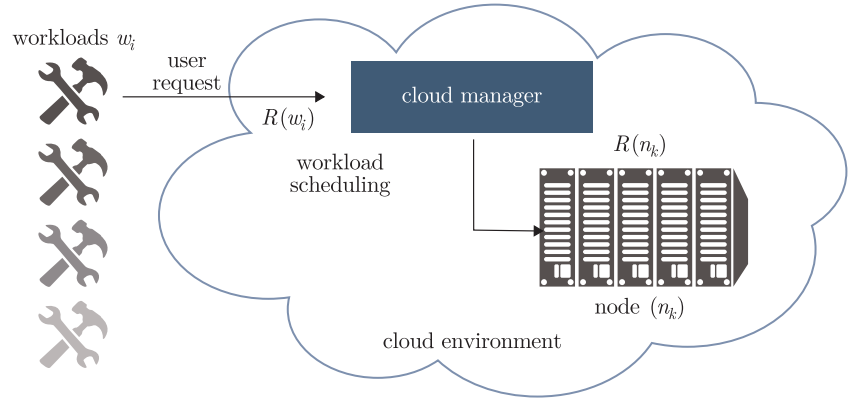

Figure 1. General cloud workload allocation model

extensions and this is what providers are looking for. Algorithms which are efficient, accomplish their objectives and are easy to deploy.

In general algorithms provided in open sourced and free cloud platforms mostly optimize the number of used servers or balance the load between servers. A general algorithm template of the allocation process is similar (see Figure 1). Clients request a cloud manager to run their workloads (w_i). The cloud manager uses defined rules in scheduling the algorithm and based on workload requirements decides on which node (n_i) the requested workload is spawned. The resources required by the workload w_i are defined as $R(w_i)$ and all resources available on node n_i as $R(n_i)$.

Two variables could be defined for further analysis. The first is named Resources Used and shows how many resources are reserved on node n_k at the moment t: $RU(n_k,t)$ (see Equation (1)). The second is named Resources Free and shows how many resources are available for new reservations on node n_k at the moment t: $RF(n_k,t)$ (see Equation (2)).

$$RU(n_k,t) = \sum_{i \in W(n_k,t)} R(w_i) \tag{1}$$

$$RF(n_k,t) = R(n_k) \setminus RU(n_k,t) \tag{2}$$

Two example models of workload scheduling could be distinguished. The optimization functions could be: (1) minimize active nodes where workloads are scheduled, (2) balance the load between nodes. The first objective is to pack workloads in the smallest set on nodes. Then all the unused nodes could be powered off and theoretically the energy consumption will be the smallest (energy efficiency factors should be considered). The second function assumes that the load should be balanced between all nodes so that algorithm schedule workloads should have a similar number of resources reserved on each node. Using such algorithms in a cloud could take effect in small workload efficiency [12]. This effect is caused by some conflicts on resources required by workloads which are executed on the same node but they are not handled by the allocation algorithm.

3. Complementarity oriented approach

The considered model of a cloud node is presented in Figure 2. Many resources could be distinguished. The usage of each resource could be measured on the host (cloud node). The problem is to measure some of them on a virtual machine (guest node), where the workload is executed. The most trivial usages to measure are virtual cpus, memory and storage size, network bandwidth. However, there are some resources which could not be measured per workload. Representative examples could be the CPU cache, memory bandwidth, network card buffer usage. This immeasurable host node resources could be significant for the workload execution, but in most cases they are not considered in the context of workload scheduling. Most of the allocation algorithms consider the basic node resources only.

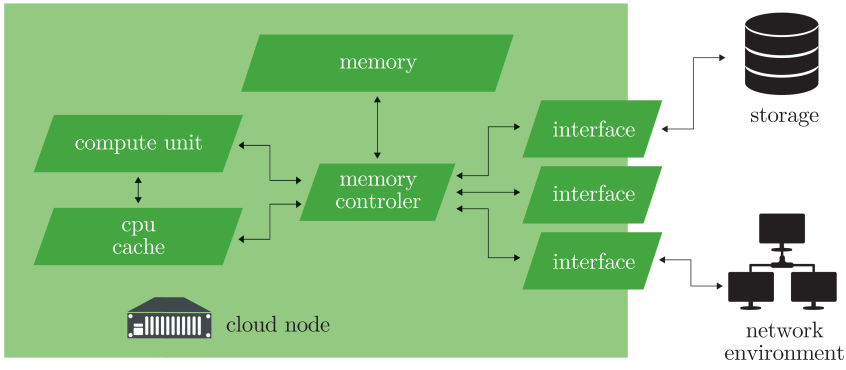

Figure 2. Hardware node model

The following problems could be distinguished based on the mentioned literature and the proposed model:

1. The workload resource requirements ($R(w_i)$) hardly depend on a concrete node architecture (n_k) and should be defined for each (w_i, n_k) pair.
2. $R(w_i, n_k) \neq R(w_i, n_l)$ assuming that nodes k and l are heterogeneous and $R(w_i, n_k)$ denotes the resources used by workload w_i on node n_k.
3. Resources used by workloads w_i and w_j working on same node n_k cannot be represented as an arithmetic sum of the resource requirements:

$$R(w_i + w_j, n_k) \neq R(w_i, n_k) + R(w_j, n_k) \tag{3}$$

Assuming that not every significant resource could be defined and/or measured, then some other method should be used to describe if the workload is running properly. A possible solution to achieve it is to define some quality parameters for workloads. Each workload can have its own set of such parameters (named $Q(w_i)$). Depending on the type of workloads different quality parameters could be considered. Some example parameters could be:

- workload execution time (for some batch workloads);
- mean response time;
- the number of requests per second with no error response.

Let us define some quality parameter for workload w_i named q_l and a period of workload execution $\Delta t = t_2 - t_1$. The workload could be scheduled to an empty node or a node where some other workload executes.

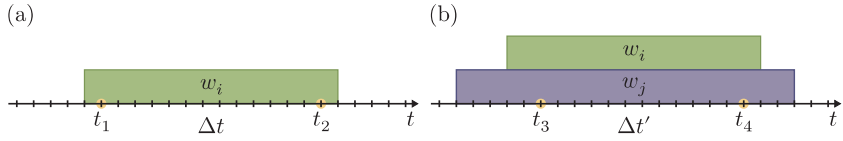

Figure 3. Workload allocation during experiments on node n_k: (a) workload w_i scheduled on node, (b) workload w_i running on node with one another workload w_j

The first example has been presented in Figure 3 (a). Workload w_i is the only workload running on node n_k (Equation (4)). During Δt the measured value of parameter $q_l(w_i)$ is equal to v_1 (Equation (5)).

$$\forall t \in \Delta t: RU(n_k, t) = R(w_i) \tag{4}$$

$$q_l(w_i) = v_1 \tag{5}$$

The second example assumes that there is another workload running on a node when workload w_i has been scheduled (Equation (6)). Then, in the same duration Δt quality parameter q_l has the value v_2 (7). The situation is shown in Figure 3 (b).

$$\forall t \in \Delta t: RU(n_k, t) = R(w_i) \cup R(w_j) \tag{6}$$

$$q_l(w_i | w_j, n_k) = v_2 \tag{7}$$

Having two experimental values v_1 and v_2 let us define the influence of workload w_j on the measured workload w_i. Such impact will be defined as complementarity factor α (see Equation (8)).

$$\alpha_{i,j} = \frac{v_2}{v_1} = \frac{q(w_i | w_j, n_k)}{q(w_i, n_k)} \tag{8}$$

The complementarity factor defines the workload complementarity definition: workloads w_i and w_j are complementary on node n_k only when there exists some quality parameter for workload w_i the value of which does not change when the workload is executed with or without workload w_j (see Equation (9)).

$$w_i \underset{n_k}{\sim} w_j \Leftrightarrow \exists_{q_l(w_i|w_j,n_k) \in Q(w_i)} : \alpha_{i,j} \approx 1 \tag{9}$$

Based on the results of experiments and computed α factors for different workloads in a set of workloads W a special matrix (named **complementarity matrix**) could be defined (see Equation (10)).

$$\forall w_i, w_j \in W: \tilde{A}_{i,j}(n_k) = \alpha_{i,j} \tag{10}$$

4. Allocation algorithm for cloud

Many of schedulers in cloud platforms allow users to extend their behavior by implementing new layers named filters. The proposed approach with workload complementarity could be applied to such a filter and be used. An algorithm shown below (see Figure 4) is an example of such filter which can be used to select the best node for a workload where interference with other running workloads will be the smallest.

```
1  begin
2      N ← set of nodes
3      w_i ← workload to schedule
4      R(w_i) ← resources required to allocate for workload w_i
5      n_s ← node selected to schedule
6      λ_s ← selected node complementarity factor
7      now ← current moment t
8
9      n_s = null
10     foreach n in N:
11         if RF(n,now) >= R(w_i):
12             λ_n = 0
13             if W(n) is empty:
14                 n_s = n
15             else:
16                 foreach w in W(n):
17                     λ_n += Ã[w_i,w]
18                 end
19                 if λ_n < λ_s:
20                     λ_s = λ_n
21                     n_s = n
22                 end
23             end
24         end
25     end
26     return n_s
27 end
```

Figure 4. Scheduling filter with complementarity matrix usage

A theoretical example of the complementarity algorithm effectiveness is presented below. The experiment has the following assumptions. 8 workloads have to be scheduled. Only one resource is considered: CPU cores. The workload requires 1 to 4 cores each. The testbed environment consists of 3 nodes (10 cores each). Each workload has the same quality parameter set $Q(w_i) = q_l(w_i)$. To simplify further considerations, let us define $q_l(w_i)$ as the time required to handle 1000 external requests from 5 users. An additional assumption is that any workload could not have a better quality parameter $q_l(w_i)$ when some other workload works on the same node. This implies that the complementary matrix $\tilde{A}$ could have only the parameters $\alpha_{i,j} \geq 1.0$. The complementarity matrix elements in the example

were randomly selected from the range between 1 and 2. Matrix $\tilde{A}$ is a symmetric matrix.

$$\tilde{A} = \begin{bmatrix} 1.943 & 1.451 & 1.845 & 1.792 & 1.713 & 1.913 & 1.659 & 1.216 \\ 1.451 & 1.374 & 1.459 & 1.422 & 1.491 & 1.433 & 1.768 & 1.410 \\ 1.845 & 1.459 & 1.995 & 1.144 & 1.476 & 1.839 & 1.745 & 1.637 \\ 1.792 & 1.422 & 1.144 & 1.400 & 1.648 & 1.310 & 1.759 & 1.497 \\ 1.713 & 1.491 & 1.476 & 1.648 & 1.173 & 1.571 & 1.100 & 1.697 \\ 1.913 & 1.433 & 1.839 & 1.310 & 1.571 & 1.572 & 1.140 & 1.450 \\ 1.659 & 1.768 & 1.745 & 1.759 & 1.100 & 1.140 & 1.845 & 1.715 \\ 1.216 & 1.410 & 1.637 & 1.497 & 1.697 & 1.450 & 1.715 & 1.881 \end{bmatrix}$$

The results of simulations are presented in Figure 5. The plot on the left side of the figure presents the scheduling results with the balanced load algorithm usage. The plot on the right side shows the results of scheduling with the complementarity algorithm usage.

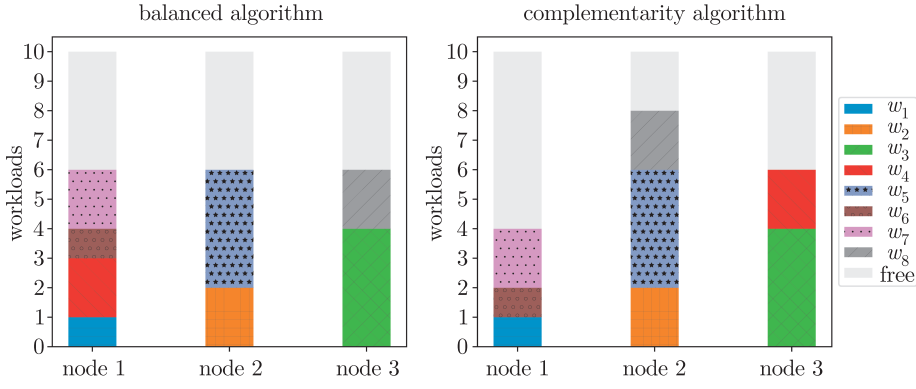

Figure 5. Scheduling results for two algorithms with different objectives. First plot with balanced load. Right plot with workloads complementarity matrix

Table 1 presents sums of complementarity factors for each workload counted after scheduling was finished. For each workload this factor was counted according to the equation: $\lambda(w_i) = \sum_{j \in W(n_k) \setminus w_i} \alpha_{i,j}$.

The assumed quality parameter was the time needed to handle 1000 requests from 5 users. In the presented simulation the time needed to handle requests by all workloads is 3.175 times longer with a balanced algorithm than it could be, if each workload worked on a separate node. This time is only 2.6125 times longer for the complementarity algorithm. The proposed approach lets the workload execution efficiency increase by about 20%.

5. Final remarks

Utilization of a complementary approach tends to a reduction of the impact between the workload executed on the same node. An example appliance could be OpenStack filter implementation which uses a complementary matrix to select the node for the workload.

Table 1. $\lambda(w_i)$ factors counted after balanced and complementarity scheduling was done

workload	balanced $\lambda(w_i)$	complementarity $\lambda(w_i)$
w1	5.36	3.57
w2	1.49	2.9
w3	1.64	1.14
w4	4.86	1.14
w5	1.49	3.19
w6	4.36	3.05
w7	4.56	2.8
w8	1.64	3.11
average($\lambda(w_i)$)	3.175	2.6125

The presented approach requires future works focused on experiments with workloads of different types running in a cloud. A complementary matrix should be determined in advance what requires additional experiments for workload execution.

Depending on access to the hardware characteristics, especially more specific and detailed parameters, could allow the proposed approach to be improved. An example of such parameter could be the size of a memory bandwidth used by a specific workload.

The entire solution could be optimized with the usage of workload classes to minimize the matrix size and simplified experiments required to prepare such matrixes for different type of nodes. Such workload classification has been proposed in [13] and carried out by the Gdansk University of Technology in cooperation with Intel Technology Poland as a part of the research project entitled the Recommendation Component for Intelligent Computing Clouds.

References

[1] Zhang Q, Cheng L and Boutaba R 2010 *Journal of Internet Services and Applications* **1** (1) 7 doi: 10.1007/s13174-010-0007-6

[2] Mustafa S, Nazir B, Hayat A, Khan A u R and Madani S A 2015 *Computers and Electrical Engineering* **47** 186

[3] Zhan Z-H, Liu X-F, Gong Y-J, Zhang J, Chung H S-H and Li Y 2015 *ACM Comput. Surv.*, ACM, **47** (4) 63:1 doi: 10.1145/2788397

[4] Madni S H H, Latiff M S A, Coulibaly Y and Abdulhamid S M 2016 *Journal of Network and Computer Applications* **68** (Supplement C) 173 doi: 10.1016/j.jnca.2016.04.016

[5] Zhao Ch, Zhang S, Liu Q, Xie J and Hu J 2009 *Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing*, *Procs WiCOM'09*, IEEE Press 5548

[6] Verma A and Kumar P 2012 *International Journal of Advanced Research in Computer Science and Software Engineering* **2** 111

[7] Chimakurthi L and Kumar S D M 2011 *CoRR* [Online] available at: `http://arxiv.org/abs/1102.2608` [Accessed: 7-Jun-2017]

[8] Gao Y, Guan H, Qi Z, Hou Y and Liu L 2013 *J. Comput. Syst. Sci.*, Academic Press, Inc., **79** (8) 1230 doi: 10.1016/j.jcss.2013.02.004

[9] Chen W N and Zhang J 2012 *A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints*, *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* 773 doi: 10.1109/ICSMC.2012.6377821

[10] Li H-H, Chen Z-G, Zhan Z-H, Du K-J and Zhang J 2015 *Renumber Coevolutionary Multiswarm Particle Swarm Optimization for Multi-objective Workflow Scheduling on Cloud Computing Environment*, *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ACM 1419 doi: 10.1145/2739482.2764632

[11] OpenStack 2017 [Online] available at: `https://www.openstack.org/` [Accessed: 7-Nov -2017]

[12] Krawczyk H, Proficz J and Daca B 2013 *Prediction of Processor Utilization for Real-Time Multimedia Stream Processing Tasks*, *Proceedings of Distributed Computing and Internet Technology* Hota Ch, Srimani P K (ed.), Springer 278 doi: 10.1007/978-3-642-36071-8_22

[13] Orzechowski P, Proficz J, Krawczyk H and Szymanski J 2016 *Categorization of Cloud Workload Types with Clustering*, *Proceedings of the International Conference on Signal, Networks, Computing, and Systems*, Springer 303