Design and Evaluation of a Scalable Hierarchical Application Component Placement Algorithm for Cloud Resource Allocation

Maryam Barshan, Hendrik Moens and Filip De Turck
Ghent University - iMinds, Department of Information Technology
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
Email:maryam.barshan@intec.ugent.be

***Abstract*—In the context of cloud systems, mapping application components to a set of physical servers and assigning resources to those components is challenging. For large-scale clouds, traditional resource allocation systems, which rely on a centralized management paradigm, become ineffective and inefficient. Therefore, there is an essential need of providing new management solutions that scale well with the size of large cloud systems. In this paper a distributed and hierarchical component placement algorithm is presented, evaluated and compared to a centralized algorithm. Each application is represented as a collection of interacting services, and multiple service types with differing placement characteristics are considered. Our evaluations show that the proposed algorithm is at least 84.65 times faster and offers better scalability compared with a central approach, while the percentage of servers used and fully placed applications remains close to that of the centralized algorithm.**

***Index Terms*—cloud management, application placement, hierarchical systems.**

I. Introduction

Nowadays many companies use of cloud technologies to reduce costs, increase flexibility and to respond faster to customer needs. Although the benefits of cloud systems are considerable, numerous challenges remain, among them, effective supervision of resource usage, scalability and in particular resource allocation. The resource allocation or the application placement refers to the act of deciding where on the clusters of servers, the applications are placed [1]. Managing placement of a large number of applications on a large number of cloud servers leads to unsustainable administrative costs, requiring automated approaches for such a management task.

In cloud management literature, application placement deployments often rely on traditional systems in which the management of application placement is typically centralized. With the expected hundreds of thousands of servers and significant customer demands in next generation of cloud systems, centralized approaches won't be able to scale well. As a solution, scalability can be achieved by dividing a cloud into different administrative domains, with independent managers for each domain and efficient interaction among these managers.

In this paper, we address the problem of application placement for large-scale cloud environments with the following design goals: scalability and performance. The proposed hierarchical algorithm executes faster than a centralized approach and each management cluster maintains a partial view of the network. Consequently the resource allocation process is scalable both in the number of cloud servers (up to 512000 servers) and the number of applications placed onto the cloud servers. Furthermore, as a part of our presented approach a local application placement policy is followed for each administrative domain. This partial solution tries to minimize the number of servers used which is mostly effective to reduce the power consumption of large-scale datacenters [2]. Additionally, as the requirements of application components differ, two general component types have been defined: database and computational components. Moreover, as a constraint each server is just allowed to place one kind of application component, but when there is enough capacity, components can be placed on the same server. To make it clear, Figure 1 shows an illustrative example of application component placement into a small cluster of cloud servers.

In the context of modern cloud platforms, the application placement process consists of placing the application components to a set of VMs (Virtual Machines) and then deploying to the physical infrastructure [3], [4]. In this paper we assume that the components are already encapsulated in VMs or similar containers, so the applications consist of multiple VMs.

The rest of the paper is organized as follows. In Section II, we discuss related work. Section III describes the architecture of the proposed method. In Section IV the algorithms are discussed in detail. Then in Section V we evaluate the proposed algorithms. Finally in Section VI, we conclude the paper.

II. Related Work

As shown in [5] the problem of application placement is NP-hard, making it desirable to find near-optimal solutions. Recently many approaches to the application placement have been proposed that each focus on different aspects of the problem [6]. While many approaches such as [3], [7], [8] and [9] rely on the central approaches, [10] offers a distributed resource management architecture. This distributed approach does not take communication between application components into account however. [11] focuses on resource allocation in IaaS clouds, maximizing resource utilization and request acceptance rate. Another work [12] clarifies the definition of distributed cloud and the challenges of resource allocation

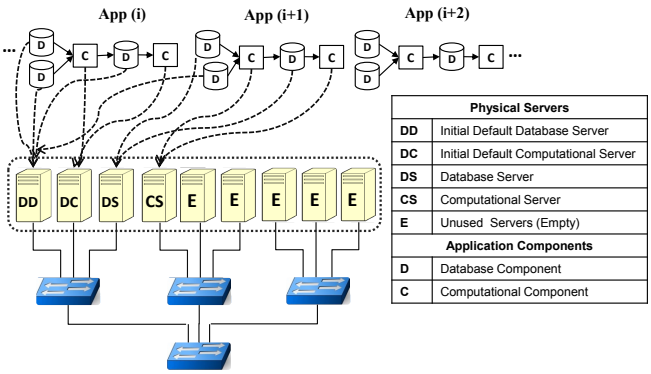


Fig. 1: The process of application component placement.

on distributed clouds. Based on the definition of "the best-fit placement" in [5], our solution follows the same rule which is finding a feasible server whose residual capacity is minimal. Nevertheless, this work differs from our approach as it is centralized and not network-aware.

In our previous work [13] we presented an optimal ILP-based solution with the main focus of cost-effectiveness along with a central heuristic algorithm. This algorithm is however centralized, making it only useful for smaller datacenters. In this paper a decentralized algorithm is designed and evaluated which can be applied in large-scale cloud environments. Our other previous work [14] focuses on component-based application as well as network-awareness. The main focus of the paper was on placement of multi-tenant component-based applications where VMs are shared between the applications and no placement constraints between different application component types were considered. In this paper, our approach works differently as it offers a VM-level placement, the application component is characterized as being either a database or a computational one. Furthermore, multiple dedicated types of servers are defined here, one type for hosting computational components and another type for database components.

Generally, what distinguishes our method from other approaches are: 1) our solution is decentralized and scalable; 2) application modeling is component-based with two different types, and interaction between those components affects the placement process; 3) SLA agreements and the properties of underlying network are respected; 4) servers are not allowed to host different component types and multiple same-type components can be placed onto a single node; and 5) our method minimizes the number of servers used.

III. Description of Models

The model consists of three parts:

The model of the physical cloud system: The physical cloud system is a graph consists of homogeneous server and links.Each server can be either a database or computational server and no backup is assumed.

The model of the management plane: The management plane relies on multi-layered hierarchical architecture in which three types of manager are defined: LLM (Low Level Manager), MLM (Mid Level Manager) and RLM (Root Level Manager). The LLMs are located in the lowest level of management hierarchy, the RLM in the top level and MLMs in middle ones. The number of management levels (|ML|) and the number of supported servers (|SS|) for each LLM are taken as inputs and the branch factor of each tier (μ) is calculated.In addition, the number of supported servers and the number of levels determine the number of LLMs. By calculating the level branch factor the number of MLMs can be achieved as follows.

$$\mu = \sqrt[|ML|-1]{|LLM|} \tag{1}$$

$$|LLM| = \lceil |S|/|SS| \rceil \tag{2}$$

$$|MLM| = \sum_{level=1}^{|ML|-2} \mu^{((|ML|-1)-level)} \tag{3}$$

The model of the applications: The architecture of the applications is service oriented and the topology of the services is a graph. The components are the nodes and connections between these components form the directed links of the application graph. Each component has a specific amount of CPU, memory and storage demands. Also, the application link bandwidth and delay demands can be considered as the SLA.

IV. Algorithms

In this section two centralized and hierarchical approaches are discussed which we refer to as the Centralized Cloud Mapping Algorithm (CCMA) and the Hierarchical Cloud Mapping Algorithm (HCMA) respectively.

A. Centralized Cloud Mapping Algorithm (CCMA)

This algorithm first arbitrarily chooses two nearby nodes as the default database and the default computational servers. For each application the algorithm goes through all the components and tries to allocate resources to each component. In order to have minimal bandwidth overhead, the algorithm uses the Dijkstra shortest path algorithm [15] for mapping the application links. However, there are two situations in which the application component cannot be placed, either because of physical node or link limitations. Node limitation occurs when there is not enough residual CPU, memory or storage capacity in one of default servers. In the latter case again there are two situations.First the application components cannot be connected because there is no link to connect application components located on different servers. Second, no possibility because bandwidth or delay demands cannot be responded.

No matter which situation is causing unsuccessful component placements, the Next Server Selection (NSS) process should be followed to choose another default server. In the NSS process, a Breadth First Search (BFS) algorithm [15] is run with the current default server as the start vertex. However, when link limitation occurs, first another server is chosen temporarily and then the algorithm checks the path availability and SLA fulfillment, and sets it as a default server provided that choosing this server satisfies both conditions. Otherwise,

the placement is not successful. In this case the algorithm must backtrack and start again with new default servers and continue until the NSS process is unable to find a new server. This algorithm differs from our previously designed algorithm in [13] because in this paper the next server is found by calling the BFS algorithm, whereas in the previous work the process searches among the neighbours and in the case of no unvisited neighbour another arbitrary server was chosen, an approach which is less efficient for large-scale networks.

B. Hierarchical Cloud Mapping Algorithm (HCMA)

The GCMA is run on every manager and the CCMA is only run on LLMs. Based on this algorithm, all the placement requests can be sent to the current *active LLM*. The current active LLM is determined arbitrary when the algorithm starts. Each manager has two states: "full" and "not full". A manager is "full" when all its managed servers get fully occupied. The active LLM will be replaced when its state changes to "full". The next active LLM is chosen by the parent of the current active LLM. In order to interact between different managers within the management plane, another algorithm is designed which is referred to as the GCMA (Global Cloud Mapping Algorithm). The GCMA runs on every manager. For each newly arriving application, the HCMA invokes the GCMA with the current active LLM and a "new request" message. In the GCMA three types of messages are defined: "new request", "from the parent" and "full". For more clarification Figure 2 is depicted. Then, the GCMA sends the application request to the current active LLM by calling the CCMA, the centralized algorithm. In hierarchical approach the CCMA is run on every LLM. If this administrative domain was not able to place the entire application components, the status of the current active LLM changes to "full". Afterward, this LLM calls the GCMA with a "full" message to its parent.

Global Cloud Mapping Algorithm (GCMA): The GCMA is a hierarchical algorithm (Algorithm 1). Based on this algorithm when a request arrives, three cases can be distinguished:

1) A request is received by the highest level manager (RLM): The request will be forwarded to the next unvisited domain with a "from the parent" message. If all domains are full the cloud system is not able to place this application.

2) A request is received by the mid-level manager (MLM): The request will be forwarded by applying the same policy to one dedicated lower-level manager with a "from the parent" message until the target LLM is reached. If all domains are full, the status of this manager turns to "full" and the request with a "full" message will be forwarded to the parent.

3) A request is received by the lowest level manager (LLM): In this level all the request messages would be either "new request" or "from the parent". No matter who is the sender, the manager runs the CCMA algorithm. In a saturation case , the LLM has to send the newly arriving requests to its parent and introduce itself as a "full" manager.

```
input: application a, manager m, requestSender r, message
       s
Impossibility_a ← false;
Currentstate ← Save the current system state;
while (Mapped_a = false & Impossibility_a = false) do
    Set current system state to CurrentState;
    if (m_type = LLM & s=("new request" OR "from the
    parent")) then
        if (one of the default servers=null) then
            full_m ← true;
            GCMA(a, parent_m, m, "full");
        else
            CCMA(m,a);
        end
    end
    if (m_type ≠ LLM & s = ("full" OR "from the parent" ))
    then
        for (ch ∈ children_m) do
            if (full_ch =false & ch≠r) then
                GCMA( a, ch, m, "from the parent");
                return;
            end
        end
        full_m ← true;
        if (m_type = MLM) then
            GCMA (a,parent_m, m, "full") ;
        else
            Impossibility_a ←true;
        end
    end
    if (Impossibility_a =true) then
        Set current system state to CurrentState;
    end
end
```

Algorithm 1: The Global Application Mapping Algorithm (GCMA), which is executed on every manager in the hierarchical approach.

V. Evaluation Details

The physical cloud system is tree-based nd amulti-tier, similar to the current datacenter topologies [4], [6]. To design the physical infrastructure, the number of server nodes (|S|), the number of switch ports (|P|) and the number of levels (|L|) are taken as inputs. This physical cloud environment is a complete *N-ary* tree in which the *N* is the calculated branch factor (β). The branch factor of each tier and the number of intermediate switches (|IS|) are calculated as the follows.

$$\beta = \sqrt[|L|-1]{|S|} \tag{4}$$

$$|IS| = \sum_{level=1}^{|L|-1} \beta^{((|L|-1)-level)} \tag{5}$$

Physical Servers	
DD	Default Database Server
DC	Default Computational Server
F	Full Server
E	Unused Server (Empty)
Application Components	
D	Database Component
C	Computational Component

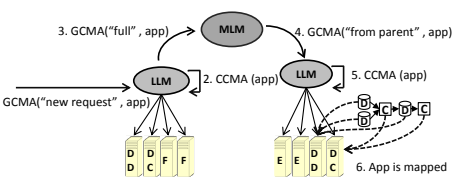


Fig. 2: Different messages for interacting between the managers in GCMA (GCMA: Global Cloud Mapping Algorithm, CCMA: Centralized Cloud Mapping Algorithm).

In our evaluation two types of applications are implemented as shown in Table III. Type 1 refers to the 5-component applications and Type 2 refers to 20-component applications.

This section is divided into three parts. First an evaluation of the CCMA is provided by comparing to an optimal solution based on the ILP-based model we presented in [13]. Then, we evaluate the HCMA by comparing its performance with the CCMA. Finally, we will end the section with a large-scale evaluation of the HCMA.

A. Comparing the CCMA to the ILP-based algorithm

1) Evaluation Set up: The optimal model and the CCMA are evaluated with a configuration of 6 servers arranged in a star topology. The network specification can be observed in Table I. Also, Type 1 applications are used, the number of which varies from 1 up to 9. The experiments are iterated 20 times and the average percentage of used servers are presented.

2) Evaluation Results: In Figure 3 the CCMA is compared to the optimal approach and the percentage of servers used to place all of the applications are depicted. As can be seen in this figure, when it comes to the physical resources usage, the CCMA provides a near-optimal solution compared to the ILP-based algorithm. In 66% of cases, the results of centralized method are equal to the optimal algorithm and in the worst case, with 9 applications, the CCMA uses 8.3 percent more servers. This analysis shows the performance of the CCMA is close to that of the optimal algorithm.

B. Comparing hierarchical algorithm to the centralized approach

We study three case studies. In the first case, 5-component applications are placed on cloud system with 1000 servers and the second case considers a larger scenario with 4096 servers and 20-component applications. In the experiments, we measure the percentage of servers used, the percentage of mapped applications and the execution times per application. Afterward, the impact of different physical infrastructures on the average number of fully mapped applications and the execution time for 1000 up to 4096 servers are analyzed.

1) Evaluation Set up: For the evaluation, the configuration of physical infrastructure is considered to be a 4-tier hierarchical tree topology. For the first scenario, the physical cloud system consists of 1000 servers (4096 for case study 2) in the lowest tier. The number of ports in each intermediate device is 10 (16) which results in 1+10+100 (1+16+256) switches in first three tiers. Consequently the number of physical nodes is 1111 (4369) in the entire cloud system. The specifications of physical cloud resources are shown in Table I.

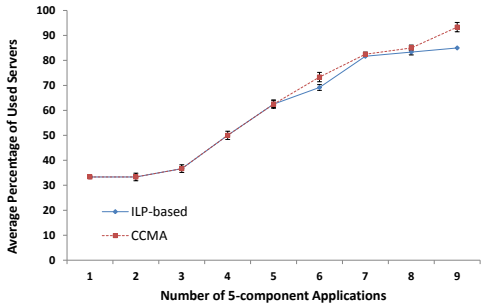


Fig. 3: Comparing the CCMA to the ILP-base algorithm (20 iterations).

TABLE I: The Physical Infrastructure Specifications.

Physical Infrastructure Specifications				
Case study	**|S|**	**|IS|**	**|L|**	**|SP|**
1	1000	111	4	10
2	4096	273	4	16
Physical Server Specifications			**Physical Link Specifications**	
CPU	**Storage**	**Memory**	**bandwidth**	**Delay**
$3GHZ$	$200GB$	$16GB$	$400Mbps$	$3ms$

TABLE II: Management plane infrastructure.

Case study	type	|ML|	|SS|	|LLM|	|MLM|	|RLM|	μ
1	CCMA	1	1000	1	0	0	-
	HCMA	3	10	100	10	1	10
	HCMA	3	40	25	5	1	5
	HCMA	2	100	10	0	1	10
2	CCMA	1	4096	1	0	0	-
	HCMA	3	16	256	10	1	16
	HCMA	3	64	64	5	1	8
	HCMA	2	256	16	0	1	16

Apart from the central management system, three different hierarchical management planes are generated and Listed in Table II. The applications are of Type 1 (Type 2), the number of which varies from 100 up to 1500 (400 up to 4000). The component demands are provided in Table III.

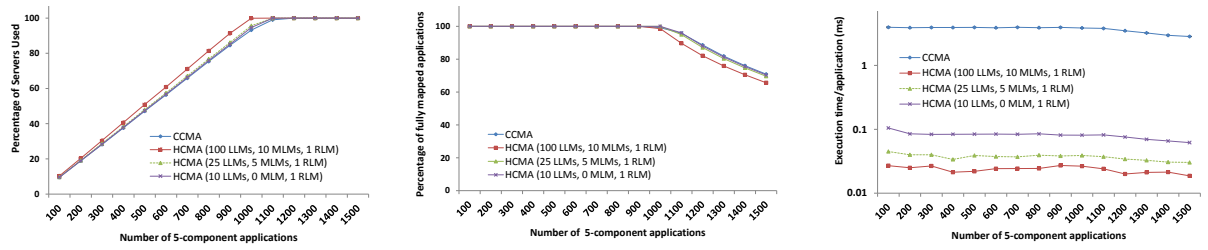
2) Evaluation Results: Figure 4a shows the percentage of used servers for different management planes. As it can be observed in Figure 4a, the number of used servers grows linearly with the number of applications until all the resources are completely occupied. Among all, the CCMA uses the fewest and the HCMA with higher numbers of LLMs uses the most servers. In the worst case the hierarchical scenario with 100 LLMs uses 6.7% more servers. Moreover, the average standard errors is 0.025% for the CCMA and 0.031% on average for the HCMAs.

In Figure 4b, the percentages of placed applications are depicted. As it can be seen the CCMA results in the best performance and the HCMA with 100 LLMs is the worst one. Additionally, application placement failures are expected due to fix number of servers and resource saturation after 1000

TABLE III: Application specifications.

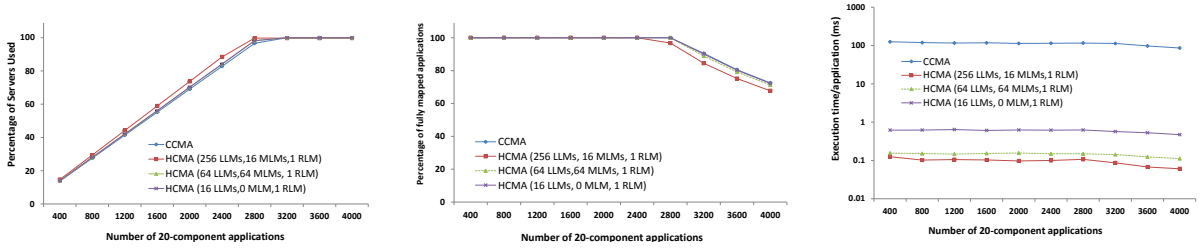
Type	# component	# link	# database	# computational
1	5	4	3	2
2	20	19	14	6

Type	Component demands(Random)			Link demands(Random)	
	CPU	**Storage**	**Memory**	**Delay**	**BW**
1	(1-1000)MHZ	(1-20000)MB	(1-2000)MB	1s	(1-50)Mbps
2	(1-200)MHZ	(1-10000)MB	(1-300)MB	200ms	(1-20)Mbps



(a) The Percentage of Servers Used. (b) The Percentage of placed applications. (c) The execution time per application.

Fig. 4: Case study 1 with 1000 servers and 20 iterations.



(a) The Percentage of Servers Used. (b) The Percentage of placed applications. (c) The execution time per application.

Fig. 5: Case study 2 with 4096 servers and 20 iterations.

applications. Nonetheless, in both figures even in the worst case, the result is within 8% of the best result.

The execution time of hierarchical approaches is promising. As it can be clearly seen in Figure 4c the time in which an application is placed in the CCMA is much higher than the hierarchical approaches, especially in the hierarchical scenario with more LLMs.

Then the result of second case study is presented. As can be observed in Figure 5a, the percentage of used servers increases by adding more applications when the servers are fully occupied. Afterwards, the percentage of mapped applications declines as the newly arriving applications are immediately rejected due to saturated resources. Although, the CCMA shows better performance, the hierarchical management planes use at most 5.6% more resources. Figure 5b compares the percentage of mapped applications in hierarchical approaches to the centralized solution. As it can be seen in the worst case the result of the hierarchical management planes is within 7% of the best result. Also, Figure 5c indicates that the execution time of the CCMA is high compared to the hierarchical scenarios.

We also evaluate the execution time and the number of fully mapped applications in different physical cloud systems with different number of servers and different number of switch ports. The applications are of Type 1 based on Table III. The assumption of physical infrastructures and management planes are provided in Table IV and Table V respectively.

In Figure 6a the number of fully mapped applications is depicted, as the β and consequently the number of servers increases, the number of mapped applications grows. On average the CCMA is able to map 6.2% more applications. However, as can be seen in Figure 6b the execution time of the CCMA dramatically grows when the number of servers increases which makes the centralized algorithms inefficient in large scale cloud systems.

C. Large scale scenarios

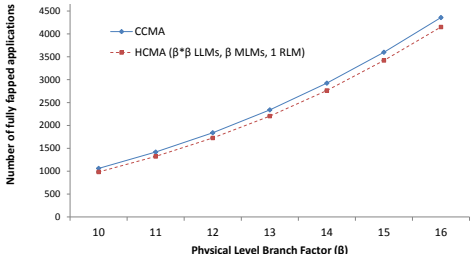
In this phase we focus on the scalability of the presented algorithms and we extend the scale of the experiments up to 512000 servers and more than 540000 5-component applications. In these experiments the number of fully mapped applications and the execution time per application is evaluated. The results are the average value of 10 experiments.

1) Evaluation Set up: The experiments are conducted for different number of servers from 1000 up to 512000. The assumptions of the applications, of the physical networks and of the management planes are provided in Table III, Table IV and Table V respectively.

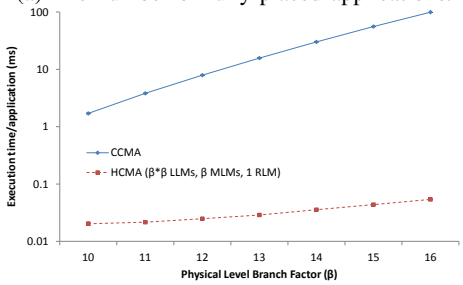
2) Evaluation Results: The number of fully mapped applications for two different hierarchical management plane architectures are compared and shown in Table VI. The numbers of successfully mapped applications are close, but the management plane with more number of supported servers in each administrative domain allocate on average 3.4% more applications. Nonetheless, while the execution time of this approach grows exponentially, the HCMA with more LLMs shows better performance, as it can be clearly seen in Table VI.

TABLE IV: The number of physical devices.

β	**10**	**11**	**12**	**13**	**14**	**15**	**16**
|S|	1000	1331	1728	2197	2744	3375	4096
|IS|	101	122	145	170	197	226	257
β	**20**	**30**	**40**	**50**	**60**	**70**	**80**
|S|	$8K$	$27K$	$64K$	$125k$	$216K$	$343K$	$512K$
|IS|	401	901	1601	2501	3601	4901	6401



(a) The number of fully placed applications.



(b) The execution time per application.

Fig. 6: Case study 3 (20 iterations).

As a result, for experiments larger than 125000 servers, only results of the second hierarchical architecture are provided.

VI. Conclusion

In this paper we presented a method to map the application components on cloud environments. To have a scalable solution, a hierarchical algorithm is designed and evaluated in order to be deployed in large scale cloud management systems. This algorithm takes the characteristic of underlying network into account and works with component-level applications. In our implementation we make a distinction between different components of applications. Also, each physical resource is just allowed to host the same-type components. In our approach during the placement, the relations between the application components have also been considered. The experimental results showed that in large scale clouds our design works efficiently compared to a centralized management system. However, the percentage of nodes used and the percentage of mapped applications remain close to that of the centralized algorithm, in the worst case 6.7% and 8% respectively.

TABLE V: The management plane parameters based on different β values.

Evaluation part	Type	|ML|	|SS|	|LLM|	|MLM|	|RLM|
B	CCMA	1	β^3	1	0	0
	HCMA	3	β	β^2	β	1
C	HCMA1	3	β^2	β	0	1
	HCMA2	3	β	β^2	β	1

TABLE VI: Large scale evaluation results.

Number of fully mapped applications							
β	**20**	**30**	**40**	**50**	**60**	**70**	**80**
HCMA1	$8.5k$	$28.7k$	$68k$	$133k$			
HCMA2	$8.2k$	$28k$	$67k$	$131k$	$227k$	$361k$	$540k$
Execution time per application.							
HCMA1	1.150	4.8	14.6	35.2			
HCMA2	0.037	0.06	0.08	0.09	0.13	0.14	0.17

Acknowledgment

This research is carried out using the Stevin Supercomputer Infrastructure at Ghent University, funded by Ghent University, the Hercules and Flemish Government, department EWI. The work is also partly supported by the iMinds DMS2 project and the FP7 NoE FLAMINGO project.

References

[1] Y. Li, F.-H. Chen, X. Sun, M.-H. Zhou, W.-P. Jiao, D.-G. Cao, and H. Mei, "Self-adaptive resource management for large-scale shared clusters," *Journal of Computer Science and Technology*, vol. 25, no. 5, pp. 945–957, 2010.

[2] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the 2009 conference on USENIX Annual technical conference*, pp. 28–28, USENIX Association, 2009.

[3] R. P. Esteves, L. Z. Granville, H. Bannazadeh, and R. Boutabai, "Paradigm-based adaptive provisioning in virtualized data centers," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 169–176, IEEE, 2013.

[4] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, 2013.

[5] B. Urgaonkar, A. L. Rosenberg, and P. Shenoy, "Application placement on a cluster of servers," *International Journal of Foundations of Computer Science*, vol. 18, no. 05, pp. 1023–1041, 2007.

[6] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014.

[7] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proceedings of the 16th international conference on World Wide Web*, pp. 331–340, ACM, 2007.

[8] T. Kimbrel, M. Steinder, M. Sviridenko, and A. Tantawi, "Dynamic application placement under service and memory constraints," in *Experimental and Efficient Algorithms*, pp. 391–402, Springer, 2005.

[9] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé, "Utility-based placement of dynamic web applications with fairness goals," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pp. 9–16, IEEE, 2008.

[10] F. Wuhib, R. Stadler, and M. Spreitzer, "Gossip-based resource management for cloud environments (long version)," *KTH Royal Institute of Technology, Tech. Rep*, 2010.

[11] A. Nathani, S. Chaudhary, and G. Somani, "Policy based resource allocation in iaas cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 94–103, 2012.

[12] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander, and J.-E. Mangs, "Resource allocation for distributed cloud: concepts and research challenges," *Network, IEEE*, vol. 25, no. 4, pp. 42–46, 2011.

[13] M. Barshan, H. Moens, S. Latré, and F. De Turck, "Algorithms for efficient data management of component-based applications in cloud environments," *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2014)*, 2014.

[14] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck, "Hierarchical network-aware placement of service oriented applications in clouds," *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2014)*, 2014.

[15] T. Cormen, *Introduction to Algorithms*. MIT Press, 2009.