

A Policy-Based Application Service Management in Mobile Cloud Broker

Woojoong Kim^(✉) and Chan-Hyun Youn

Department of Electrical Engineering, KAIST, Daejeon, Korea
{w.j.kim, chyoun}@kaist.ac.kr

Abstract. In this paper, to deploy scientific application service among advanced computing applications to the mobile cloud environment, we integrate mobile cloud system with policy-based resource management providing SLA adaptive resource management called as mobile cloud broker. However, we do not use the conventional policy-based resource management because of some problems which cannot guarantee the performance of cloud resource required by user. To resolve this problem, we propose the policy based resource management for the mobile cloud system providing scientific application service to provide the cost efficient SLA adaptive resource management to guarantee SLA required by cloud service user while minimizing cost. We describe the function and architecture of mobile cloud broker and the proposed policy-based resource management scheme in the mobile cloud broker. In addition, we show that the proposed policy-based resource management guarantee the QoS of scientific application and reduces the cost compared to the conventional cloud broker system through the evaluation.

Keywords: Mobile cloud computing · Mobile cloud broker · Policy-based resource management

1 Introduction

As the demand for high computing-intensive mobile application have been increased, there have been the attempts to apply cloud computing service to mobile environment for various mobile services [1]. Through mobile cloud computing, it is expected that the limited resources of mobile device can be overcome and mobile device accommodates the high performance computing application. To realize mobile cloud computing, a main way is to build the mobile cloud system to provide the specific mobile services to mobile users. There are many previous works in this way providing the various services such as mobile business, mobile commerce, mobile learning and mobile healthcare [4–7]. In the case of advanced computing application service (e.g. scientific application service focused on this paper) which is computing-intensive and requires many computing and storage resource [2], it is important to guarantee service level agreement (SLA) such as deadline, budget required by user [3]. However the conventional mobile cloud systems have difficulty to guarantee the SLA required by user because they do not have some functions to realize SLA adaptive resource management which enables dynamic virtual machine (VM) scaling to control QoS

based on SLA for high performance computing. Y. Jinhui et al. [8] proposed the mobile-cloud framework to constitute and execute scientific workflow on bioinformatics using mobile device. This framework supports the collaboration of researchers on an experiment of genome bioinformatics in mobile device. However, only the basic functions such as service repository, service composition, service execution and the user interface are considered to execute scientific workflow on mobile device. In addition, the SLA required by user is not considered and the resource scheduling and provisioning on each sub-task within workflow also is not considered for executing scientific workflow. To resolve this problem of the conventional mobile cloud system, we apply the policy-based resource management. There is the previous work on policy-based resource management proposed by Ren et al. [9]. However it is difficult for Ren's work to guarantee the performance of cloud resource required by user. Even in the same VM specification (the number of CPU cores, memory size and storage size), there are different performance between cloud resources provided by CSPs because of the heterogeneity [10] of cloud resource. However, Ren's work considers the VMs which have the same VM specification as the same performance so, cannot provide the optimal resource on performance and cost. In this paper, to resolve this problem of Ren's work, we propose the policy based resource management scheme for the mobile cloud system providing scientific application service to provide the cost efficient SLA adaptive resource management to guarantee SLA required by cloud service user while minimizing cost.

2 Mobile Cloud Broker for Scientific Application

To provide a brokering service for scientific application in mobile cloud, we integrate the mobile cloud system with the proposed policy-based resource management which provides cost effective and SLA adaptive resource management. This system is called as mobile cloud broker in this paper. The mobile broker system for scientific application in mobile cloud is described with its functionality in Fig. 1. There is connector which is the mobile application installed in mobile device to access to the mobile cloud broker. Connector is mobile application and provides easy interface for user to sign up and login. After login, connector accesses user's own virtual device provided by mobile cloud broker and show remote control screen of the virtual device through VNC client. In the mobile cloud broker system, the virtual device service which provides the augmented resource virtualized to mobile environment (called as Virtual Device in this paper) with various contents providing some services (e.g. scientific application service, business application service) is provided. Mobile cloud broker admits user to virtual device service and provides the connection between mobile device and virtual device. Mobile cloud broker basically store and maintain the executable file and meta-data of content for the specific service through virtual device. Especially for typical scientific application services represented by workflow, mobile cloud broker provides workflow designer through virtual device to receive the requirement (i.e. SLA) from user and submit it to the policy-based resource management function (Figs. 2 and 3).

The proposed policy-based resource management is the key function which makes it possible for the mobile cloud system to provide scientific application while satisfying

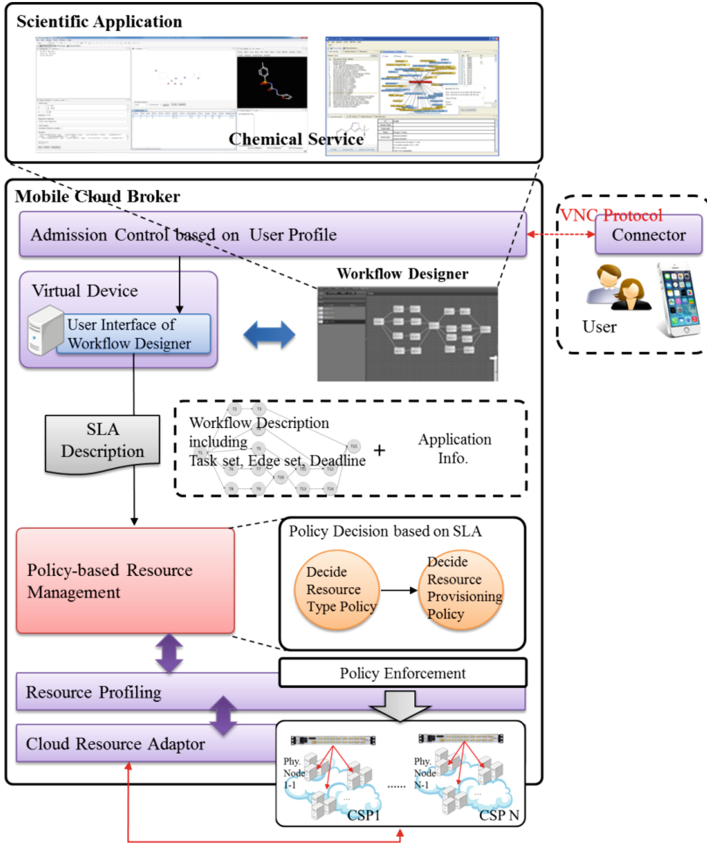


Fig. 1. The model of the integrated broker for scientific application in mobile cloud

user's SLA by providing cost effective SLA adaptive resource management. Policy means the strategy to satisfy SLA required by user for providing service. Policy-based resource management provides the convenient and transparent resource management interface for system to be adaptive about user SLA [16, 17]. In our case, we define two phases of policy – resource type policy, resource provisioning policy. Resource type policy is the strategy to decide the resource type (i.e. VM flavor type) for each sub-task within requested workflow based on given SLA (e.g. deadline, budget). Resource provisioning policy is the strategy to decide the certain physical node of the certain cloud among multi-cloud for creating VM based on given SLA (e.g. computing-intensive application, network-intensive application). The proposed policy-based resource management in the mobile cloud broker provides several policies for each phase and the adaptiveness on various user's SLA. The detail of these policies will be explained in Sect. 3. Scientific application services considered in this paper are represented as Directed Acyclic Graph (DAG) as [3] and are submitted by user with SLA description to the policy-based resource management function through workflow designer of virtual device in the mobile cloud broker. The SLA description which user

should specify in the workflow designer is composed of user id uid , application information A and workflow description G . Application information A represents the characteristics of an application such as computing-intensive and network-intensive. To represent the workflow G of DAG, n_i denotes the i th node of workflow G (node is identical to sub-task) in our notation. The connection or dependency from node i to j in the workflow is represented as edge $e_{i,j}$ and the set of edge is represented as E . In addition, the deadline D , i.e. the execution time constraints required by user for requested workflow, is also included in the workflow. The workflow description is represented as $\{N = \{n_1, n_2, \dots, n_k\}, E, D\}$.

Firstly, the workflow in the submitted SLA description is parsed and divided into sub-tasks. Then, the two phases of policy are decided based on the submitted SLA. Secondly, the flavor type of VM is allocated for each sub-task based on the decided resource type policy and the scheduled workflow description g including task-VM flavor type mapping information is returned. Finally, each sub-task within workflow is executed based on the scheduled workflow description by provisioning VM with the decided resource provisioning policy. This provisioning process is support by resource profiling function that predicts and evaluates the computing performance of each physical node (i.e. Server Rack) provided by cloud service providers in multi-cloud environment and maintains the network performance between VMs provisioned by each user. After completing the workflow, the result of requested service is returned through the virtual device.

3 Policy-Based Resource Management

As mentioned in Sect. 2, the proposed policy-based resource management provides two phases of policy for cost effective SLA adaptive resource management.

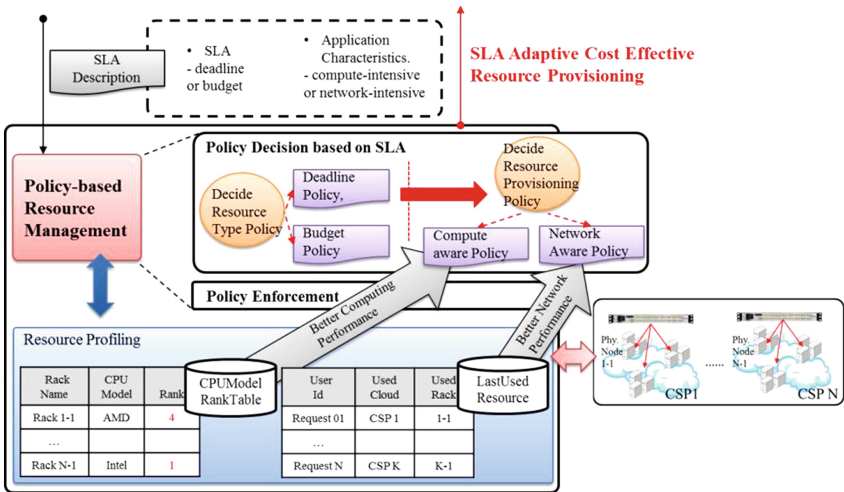


Fig. 2. The procedure of policy-based resource management in C-ARCF

When a request is submitted, the policy-based resource management makes a decision on two phases of policy sequentially for that request. In resource type policy, there are two policies – deadline policy, budget policy according to given SLA from the request. If user requests a service with deadline in SLA, the policy-based resource management decides the deadline policy to satisfy the requested deadline for scheduling workflow while minimizing the cost for leasing VM. If user requests a service with budget in SLA, the budget policy is decided to satisfy the requested budget for scheduling workflow while minimizing the total execution time. These policies use heuristic based traditional workflow scheduling with VM Packing [13] for deadline policy and Loss/Gain [15] for budget policy.

In resource provisioning policy, there are also two policies – compute-aware and network-aware resource provisioning policy according to given SLA. If the application information in submitted SLA has compute-intensive, the compute-aware resource provisioning policy is decided to provide the cloud resource having better computing capability in fixed cost, considering the heterogeneity [10] of cloud resource. To achieve this object, we use CPU model rank table based on benchmark program provided from [14]. The available racks in multi-cloud environment are sorted based on rank in CPU model rank table and the high-ranked rack is provisioned with high priority. Algorithm 1 shows the compute-aware resource provisioning policy.

Algorithm 1. Compute-aware resource provisioning policy

Input : uid^k, f (uid^k : request id, f : flavor type)

Output : created VM

```

01: if  $P^k$  is compute-aware provisioning policy
02:   sort availableRacks using CPU Model Rank Table based on benchmark program
03:   for  $r^i \in \text{availableRacks}$ 
04:     if  $r^i$  is available for flavor type  $f$ 
05:       createdVM = createNewVM( $r^i, f$ )
06:     return createdVM
07:   end if
08: end for
09: end if

```

If the application information in submitted SLA has network-intensive, network-aware resource provisioning is decided to provide the cloud resource having better network performance on the past created VM for each user. For the application which need lots of data transmission between tasks, this policy can guarantee the data transmission time between VMs. To achieve this object, last used resource table is maintained for each user. When user request first, in order words, there is no available last used resource information, load-balancing based provisioning is done so VM is provisioned to the rack having maximum remaining resource capacity. After creating the VM, the cloud and rack of the created VM is recorded to last used resource table. When user requests later and there is available last used resource information, the VM in the same rack with the last used resource is provisioned if possible. If there is not available resource capacity in last used resource for requested flavor type, new rack as close as possible to the rack of last used resource is decided. After finding the new rack and creating VM, the new rack is updated to last used resource table for the corresponding user. After policy decision process, workflow scheduling and executing

process with resource provisioning are continued based on two decided policies and the result of the request is returned to user. Algorithm 2 show the network-aware resource provisioning policy.

Algorithm 2. Network-aware resource provisioning policy

Input : uid^k, f (uid^k : request id, f : flavor type)

Output : created VM

```

01: if LastUsedResource of  $uid^k$  is available
02:   cloud  $c^k$ , rack  $r^k \leftarrow$  get LastUsedResource of  $uid^k$ 
03:   if rack  $r^k$  in cloud  $c^k$  is available for flavor type  $f$ 
04:     createdVM = createNewVM( $r^k, f$ )
05:     return createdVM
06:   else
07:     sort availableRacks in closest order from  $r^k$ 
08:     for  $r^i \in$  availableRacks
09:       if  $r^i$  is available for flavor type  $f$ 
10:         createdVM = createNewVM( $r^i, f$ )
11:         record  $\{uid^k, c^k, r^k\}$  into LastUsedResource
12:         return createdVM
13:       end if
14:     end for
15:     return null
16:   end if
17: else
18:   maxCapacityRack  $\leftarrow$  max( remainCapacity( $r^j$ ) ) ... (  $r^j \in$  rackSet of  $c^i$ ,  $c^i \in$  cloudSet )
19:   if maxCapacityRack is available for flavor type  $f$ 
20:     createdVM = createNewVM(maxCapacityRack,  $f$ )
21:     record  $\{uid^k, c^k, r^k\}$  into LastUsedResource
22:     return createdVM
23:   else
24:     return null
25:   end if
26: end if

```

In conclusion, C-ARCF can guarantee the several user's SLA adaptively for scientific application in the policy-based resource with two phases of policy management.

4 Experiments

We evaluate the proposed policy-based resource management in this paper compared to the policy-based resource management of Ren's work. Especially, we evaluate the performance of the proposed policy, i.e. compute-aware resource provisioning policies (Algorithm 1), of the resource provisioning policy phase in this paper except the network-aware resource provisioning policy and the policies in the resource type policy phase which uses traditional workflow scheduling. For the experiment of compute-aware resource provisioning policy, we use QSAR service of MapChem service [9] as compute-intensive application which is an integrated chemical application for collaborative pharmaceutical research and the available input data types are sdf30, sdf100, sdf200(sdf100 means the input data which includes a hundred of chemical compounds

info expressed by structure data format [12]). We use openstack cloud environment and the available resource policies in openstack cloud environment are small type(1 CPU, 2 GB MEM, 10 GB Disk), medium type(2 CPU, 4 GB MEM, 10 GB Disk) and large type(4 CPU, 8 GB MEM, 10 GB Disk). The SLA of QSAR service is defined as deadline and it is set in random on each request. We build the cloud testbed using 6 nodes with the deployment of openstack cloud environment [11]. Node 1, 5 is nova controller and node 2, 3, 4, 6 are nova compute nodes. Nova controller manages the operation between the nova compute nodes such as create or delete instance or snapshot and received the result from the nova compute nodes. Each node has the hardware specification described in Table 1.

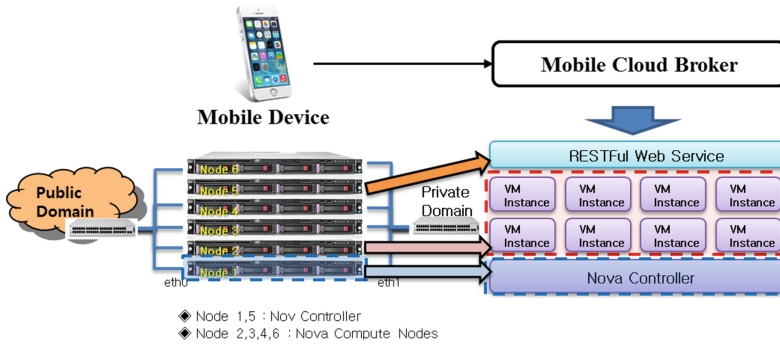


Fig. 3. The experiment environment of integrated broker with openstack

Table 1. The specification of computing node in openstack environment

	Node1	Node2	Node3	Node4	Node5	Node6
Function	Nova Controller Node	Nova Compute Node	Nova Compute Node	Nova Compute Node	Nova Controller Node	Nova Compute Node
Specification	H/W: Intel, Xeon E5620 2.4G, Core 16, MEM 16G, HDD 1T OS: Ubuntu 12.04				H/W: Intel, Xeon W3520 2.67G, Core 8, MEM 16G OS: Ubuntu 12.04	
IP address	Eth0 (143.248. 152.64)	Eth0 (143.248. 152.61)	Eth0 (143.248. 152.62)	Eth0 (143.248. 152.63)	Eth0 (143.248. 152.16)	Eth0 (143.248. 152.17)

In this experiment environment, to evaluate the adaptiveness on SLA required by user, we measure the SLA violation when the requests are occurred in the fixed interval over a period of time first. Second to evaluate the ability which guarantees SLA required by user while minimizing cost, we measure the total cost when the requests are occurred in the fixed interval over a period of time. We repeat the experiment with different request interval over a period of time. In addition, we refer the metrics which Ren et al. use for evaluation so, we use *relative cost* [9] which does not have monetary

meaning in reality but, the cost has theoretical meaning for comparison between algorithms or models. A lower relative cost means the lower monetary cost in reality. Therefore, we can save the monetary cost in reality by reducing the relative cost.

- **The result of policy-based resource management.**

We make the request in the different interval time (4 s, 3.6 s, 3.2 s, 2.8 s, 2.4 s, 2 s, 1.6 s, 1.2 s, 0.8 s) within 3 min. The input data type of the request is randomly chosen from available input data such as sdf30, sdf100, sdf200. QSAR service of MapChem service is used and the deadline is chosen randomly as mentioned above. Since QSAR service of MapChem service is compute-intensive application, compute-aware resource provisioning policy (Algorithm 1) is decided in this experiment.

Table 2. Cost performance comparison of proposed system and conventional system

Request Interval Time(sec)	The Total Cost of Integrated broker(RC)	The Total Cost of MapChem Broker(RC)
4	2107242	9239967
3.6	2355759	7844055
3.2	2812881	7227801
2.8	3120249	5403198
2.4	3850263	11913414
2	4251492	8037645
1.6	5781423	12119163
1.2	7465920	17783175
0.8	10515201	20728950

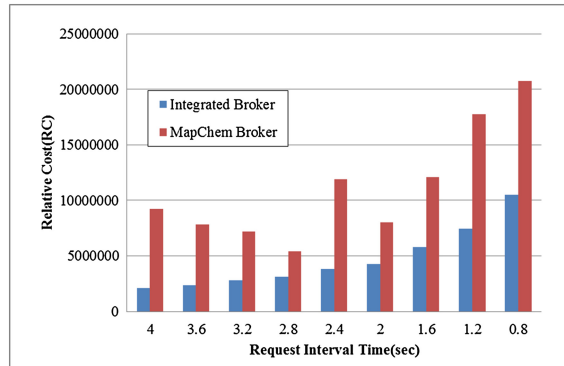
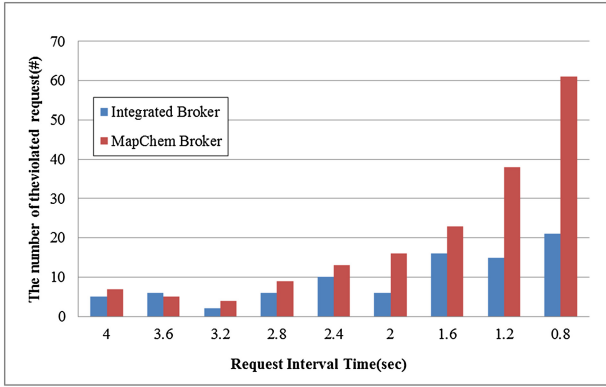


Fig. 4. Cost performance comparison of proposed system and conventional system defined in Table 2

In Fig. 4, we see that the total cost of Ren's work is higher than that of the proposed policy-based resource management over all interval times of the request. Ren's work prepares the extra VM in advance with auto-scaling to reduce the delay from VM initiation time. According to the proposed auto-scaling in Ren's work, at least one empty

Table 3. SLA violation comparison of proposed system and conventional system

Request Interval Time(sec)	Total SLA Violation of Integrated broker(#)	Total SLA Violation of MapChem Broker(#)
4	5	7
3.6	6	5
3.2	2	4
2.8	6	9
2.4	10	13
2	6	16
1.6	16	23
1.2	15	38
0.8	21	61

**Fig. 5.** SLA violation performance comparison of proposed system and conventional system defined in Table 3

VM is always kept. This mechanism makes the waste of resource and occurs the high cost. On the other hand, the proposed policy-based resource management do not makes the extra resource because we judge that the VM initiation time is not critical compared to the execution time of a request. Since the proposed policy-based resource management launches VM only when a request is submitted and release VM immediately after the requests in VM are finished, the cost of the proposed policy-based resource management is lower than that of Ren's work. In addition, the proposed policy-based resource management can allocate the proper amount of cloud resource for the request to guarantee the SLA while minimizing the cost with predicted performance. However, Ren's work cannot allocate the proper amount of cloud resource based on the SLA of the request because Ren's work allocated VM statically based on the size of the input data type. For example, sdf30 is only mapped to small type and sdf100 is only mapped to medium type and sdf200 is only mapped to large type.

For measuring SLA violation, we also makes the request in the different interval time (4 s, 3.6 s, 3.2 s, 2.8 s, 2.4 s, 2 s, 1.6 s, 1.2 s, 0.8 s) within 3 min with the same

environment as above experiment. The metric of SLA violation is defined as the number of request which violate the deadline required by user.

Figure 5 shows that Ren's work has the higher SLA violation over the entire request interval time compared to the proposed policy-based resource management. In addition, as the interval time of request decrease, the gap of SLA violation between Ren's work and the proposed policy-based resource management is increased. Therefore, in the high rate of request, the proposed policy-based resource management will show better performance on SLA violation. In the same reasons on the result of cost experiment, Ren's work cannot allocate the proper amount of cloud resource based on the SLA of the request because Ren's work allocated VM statically based on the size of the input data type and does not consider the SLA required by user. However, the proposed policy-based resource management can achieve cost effective SLA adaptive resource management which guarantee the SLA while minimizing the cost.

5 Conclusion

In this paper, we integrate the mobile cloud system with the policy-based resource management for scientific application to provide the scientific application service such as chemical and bio applications with satisfying the SLA required by user in mobile cloud environment. However for this, we do not use the conventional policy-based resource management because of some problems mentioned in Sect. 1. Therefore, to resolve the problem, we propose the policy based resource management for the mobile cloud system providing scientific application service to provide the cost efficient SLA adaptive resource management to guarantee SLA required by cloud service user while minimizing cost. Finally, we evaluate that the proposed policy-based resource management guarantee the SLA required by user and reduces the cost compared to the conventional policy-based resource management. Therefore, we prove that the proposed policy-based resource management can achieve cost effective SLA adaptive resource management which guarantee the SLA while minimizing the cost for providing scientific application service in mobile cloud broker.

Acknowledgments. This research was supported by Next-Generation Information Computing Development Program through the NRF funded by the Ministry of Education, Science and Technology (2010-0020732) and the MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2014.

References

1. Srirama, S.N., Paniagua, C., Flores, H.: CroudSTag: social group formation with facial recognition and mobile cloud services. *Procedia Comput. Sci.* **5**, 633–640 (2011)
2. Ostermann, S., Iosup, A., Yigitbasi, N.M., Prodan, R., Fahringer, T., Epema, D.: An early performance analysis of cloud computing services for scientific computing. Technical report PDS-2008-006, TU Delft, 3 December 2008. <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>

3. Yu, J., Buyya, R.: Scheduling Scientific Workflow Applications with Deadline and Budget Constraints Using Genetic Algorithms *Scientific Programming*, pp. 217–230. IOS Press, Amsterdam (2006)
4. Huang, D., Zhang, X., Kang, M., Luo, J.: Mobicloud: a secure mobile cloud framework for pervasive mobile computing and communication. In: *Proceedings of 5th IEEE International Symposium on Service-Oriented System Engineering* (2010)
5. Yang, X., Pan, T., Shen, J.: On 3G mobile e-commerce platform based on cloud computing, pp. 198–201, August 2010
6. Gao, H., Zhai, Y.: System design of cloud computing based on mobile learning. In: *Proceedings of the 3rd International Symposium on Knowledge Acquisition and Modeling (KAM)*, pp. 293–242, November 2010
7. Doukas, C., Pliakas, T., Maglogiannis, I.: Mobile healthcare information management unitizing cloud computing and android OS. In: *Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (EMBC)*, pp. 1037–1040, October 2010
8. Yao, J., et al.: Facilitating bioinformatic research with mobile cloud. In: *The Second International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2011* (2011)
9. Ren, Y.: A cloud collaboration system with active application control scheme and its experimental performance analysis, Master thesis, Korea Advanced Institute of Science and Technology (2012)
10. Farley, B., et al.: More for your money: exploiting performance heterogeneity in public clouds. In: *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM (2012)
11. Openstack foundation (2012). <http://www.openstack.org/>
12. Dalby, A., Nourse, J.G., Hounshell, W.D., Gushurst, A.K.I., Grier, D.L., Leland, B.A., Laufer, J.: Description of several chemical structure file formats used by computer programs developed at molecular design limited. *J. Chem. Inf. Model.* **32**(3), 244 (1992)
13. Kang, D.-K., et al.: Cost adaptive workflow scheduling in cloud computing. In: *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication*. ACM (2014)
14. CPU model rank table. <http://www.cpubenchmark.net/>
15. Sakellariou, R., Zhao, H.: Scheduling workflows with budget constraints. In: Gorlatch, S., Danelutto, M. (eds.) *Integrated Research in GRID Computing. CoreGRID Series*, pp. 189–202. Springer, New York (2007)
16. Changkun, W.: Policy-based network management. In: *Proceedings of IEEE WCC 2000-ICCT* (2000). inf.ufrgs.br
17. Simplifying network administration using policy-based management - DC Verma, IBMTJWR Center, Y Heights - Network, IEEE (2002). ieeexplore.ieee.org

Cloud Computing

5th International Conference, CloudComp 2014, Guilin,
China, October 19-21, 2014, Revised Selected Papers

Leung, V.C.; Lai, R.X.; Chen, M.; Wan, J. (Eds.)

2015, IX, 245 p. 94 illus., Softcover

ISBN: 978-3-319-16049-8