An on Demand Virtual CPU Arhitecture based on Cloud Infrastructure

Erhan Gokcay
Software Engineering Department, Atilim University, Incek, Ankara, Turkey

Keywords: Cloud Framework, Parallel Computation, CPU on Demand, Cloud CPU.

Abstract: Cloud technology provides different computational models like, including but not limited to, infrastructure, platform and software as a service. The motivation of a cloud system is based on sharing resources in an optimal and cost effective way by creating virtualized resources that can be distributed easily but the distribution is not necessarily parallel. Another disadvantage is that small computational units like smart devices and less powerful computers, are excluded from resource sharing. Also different systems may have interoperability problems, since the operating system and CPU design differs from each other. In this paper, an on demand dynamically created computational architecture, inspired from the CPU design and called Cloud CPU, is described that can use any type of resource including all smart devices. The computational and data transfer requirements from each unit are minimized. Because of this, the service can be created on demand, each time with a different functionality. The distribution of the calculation over not-so-fast internet connections is compensated by a massively parallel operation. The minimized computational requirements will also reduce the interoperability problems and it will increase fault tolerance because of increased number of units in the system.

1 INTRODUCTION

Advances in cloud systems are increasing rapidly as users are discovering cost and performance benefits of cloud systems. Some features can be listed as efficient resource sharing, security, flexibility and on-demand service. Available hardware and software resources can be shared among different users and/or systems with a higher granularity than before. This is important because most of the time the computing resources of a system is never utilized fully in standalone mode. There are different deployment models (Vouk, 2008) (Zhang, 2010) (Dillon, 2010) (Sonisky, 2011) like private, public, community and hybrid models. Almost anything is provided as a service.

The term cloud has different definitions. We could say that clouds are a large pool of virtual and easy to reach resources (hardware and/or software). These resources can be dynamically adjusted and assigned depending on the demand. Cloud computing is based on several old concepts like Service-oriented architecture (SOA), distributed and grid computing (utility computing) (Foster, 2008) and virtualization described in (Youseff, 2008) (Vaquero, 2009) and (Vouk, 2008). Security is always an issue in computing systems. With a distributed approach protection, security and privacy issues become more important and this issue is analyzed in (Hashizume, 2013) (Liu, 2011) and (Basu, 2012).

There is a great deal of work to create a standard for the services provided such as Distributed Management Task Force (DMTF) which is an interoperable cloud infrastructure management standard focuses on interoperability (Buyya, 2009). Storage Networking Industry Association (SNIA) SNIA standards are used to manage the data, storage, information and also address the issues such as interoperability, usability, and complexity (Popovic, 2010) (snia.org). OGF standards Open Virtual Machine Format (OVF) is a platform independent format which provides the features like efficiency, flexibility, security and mobility of virtual machines (snia.org) in order to achieve interoperability.

One of the studies in this area is done in (Botta, 2016) where the focus is on the integration of Cloud and IoT. Although the integration is discussed in

Gokcay, E.
An on Demand Virtual CPU Arhitecture based on Cloud Infrastructure.
DOI: 10.5220/0006239803510357
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 323-329
ISBN: 978-989-758-243-1

detail, still IoT devices are passive elements, not contributing to the computation power of Cloud.

The computational units are getting decentralized but the limit of this process needs to be answered. The growing IoT concept needs to be merged to Cloud systems and the interoperability problems need to be solved as well.

The research question is that how small the computational units can get in a Cloud environment, how these small units can be configured, how IoT devices can contribute to the Cloud computation and how smaller devices will help to the interoperability problem.

Chapter 2 describes the basic challenges of cloud systems and the motivation of the paper. Chapter 3 describes the new service architecture called Cloud CPU. In Chapter 4, the basic building blocks of Cloud-CPU architecture is explained. Chapter 5 describes the execution flow of Cloud-CPU operations. Finally, Chapter 6 summarizes the advantages of the Cloud-CPU system.

2 CLOUD CHALLENGES

The cloud computing faces many challenges that are related to the data interoperability and portability, governance and management, metering and monitoring, security which are addressed by MOSAIC (Opensource API and Platform for Multiple Clouds) (Petcu, 2013). There are interoperability problems between cloud systems and services because of different services depend on different operating systems and CPU types. Each vendor is providing a different service with a different set of tools. Most cloud computing systems in operation today are proprietary, rely upon infrastructure that is invisible to the research community, or are not explicitly designed to be instrumented and modified by systems researchers (Nurmi, 2009).

Although there are many open-source cloud systems for researchers as the development of cloud computing, they still are using a different Application Programming Interface (API) from each other. For IaaS, there are some popular open-source cloud systems, such as Eucalyptus (Nurmi, 2009), Open Nebula (opennebula.org), Nimbus (nimbusproject.org), etc.

A different weakness of cloud computing is the exclusion of not-so-powerful units from the system as a source of the services provided. Smart devices and computational units with less resources are basically consumers in a cloud system as described in (Khan, 2004). Those devices, although very high in terms of connected units, are using cloud systems but not providing any resource back.

3 CONFIGURABLE ON DEMAND SERVICE

In order to remove the dependency problem to the service provider, a new framework is needed in such a way that the consumer or user can configure the services needed, including functionality and computing power, from the cloud system. Instead of providing a high level service and computation, cloud systems will provide low level services or computations without any final computational goal and the user will combine these services to configure its own customized and dedicated service.

The design is inspired from the design of a CPU and therefore similar terminology will be used in the paper. The framework will be called Cloud CPU with references to CPU building blocks, although the name similarity does not mean a one-to-one duplication of CPU architecture and certainly there is no intention to replace a CPU with the service. The Cloud CPU represents the specific service that the user is trying to configure. CPU instructions represent minimized services or computations received from cloud units. Registers represent basic storage units needed in the service.

3.1 Reduced Cloud Service Model

The design of a CPU had faced a reduction in terms of complexity of the instruction set. The early designs include complex instructions (CISC), whereas later designs include very simple instructions (RISC) to execute a program with higher efficiency. The building blocks of RISC architecture can be designed with less errors.

The Cloud CPU framework is using a similar approach where simple services are used instead of complicated computations and therefore the implementation is called a Reduced Cloud Service Model (RCSM). The required computation is build on top of very basic cloud services which are very easy to implement by any unit attached to the cloud.

Because of basic service requirement from each node, any smart device can contribute to the Cloud CPU. A simple device can provide a simple arithmetic operation whereas a complicated device or unit can provide a more sophisticated calculation.

3.2 Cloud CPU Execution Flow

The Cloud CPU execution flow is given in Figure 1. The final collection of nodes and the sub-services provided as a whole creates the Cloud CPU for this particular application or service and it can be saved for future use. For another computation or service requirement, the same procedure will be repeated to create another Cloud CPU (CCPU).

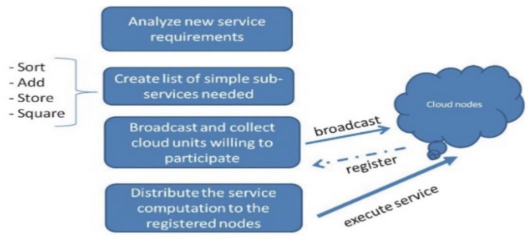
Analyze new service requirements
- Sort
- Add
- Store
- Square
Create list of simple sub-services needed
Broadcast and collect cloud units willing to participate
Distribute the service computation to the registered nodes
Cloud nodes
broadcast
register
execute service

Figure 1: Execution Flow of Cloud CPU.

4 CLOUD CPU ARCHITECTURE

The configurable service is called Cloud CPU and once it is created, it can be saved for future use. The instruction set represents the minimized collection of sub-services. Program counter represents a service that controls the flow of the execution. Register sets represent storage for the service. Process represents all the information that represents the created service or Cloud CPU. The Compiler here represents the process to decide the services needed and created. The configurable service will be explained as a virtual CPU creation and execution. Helper services needed in the flow can also be saved in the cloud which decentralize the operation. The CCPU architecture is given in Figure 2.

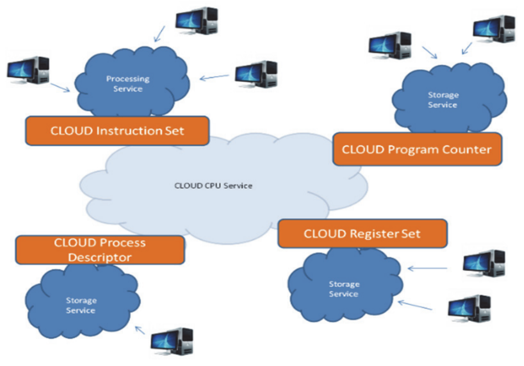

Figure 2: Cloud CPU basic elements.

A CCPU, as its names implies, needs instructions to execute. The instruction set of a CCPU is also provided as a service where the set will be called Cloud Instruction Set (CIS). Since a huge amount of nodes are connected to a cloud system, the number of CCPU services that can be provided by the system is limited only by the capacity of the cloud system. Each Cloud CPU can use different instruction sets, registers and architecture.

For a given CCPU created in the cloud, nodes or resources connected to the cloud will register themselves to provide the required CCPU services. The node assignments may change with time due to failures or high load constraints where multiple resource assignments for each service will provide fault tolerance and protect the CCPU.

Since there are multiple CCPU's implemented, each node in the cloud may registers itself to more than one service, fully or partially. For example one node may have resources to execute two instructions from one CCPU and four instructions from another CCPU. Another node may register itself to store ten registers from one CCPU and four registers from another CCPU.

4.1 Cloud CPU ID (CCPUID)

The virtual CCPU created should have an id, so that it can be identified and referred later. During the creation of CCPU, several nodes in the cloud should register themselves to keep CCPU IDs and to keep track of all requests asking for a CCPU for execution. The instruction and register set of the CCPU, the set of nodes that registered themselves to implement Cloud Instruction Set (CIS) and Cloud Register Set (CRS) are also part of the information attached to CCPU ID.

4.2 Cloud Instruction Set (CIS)

The instruction set provided by the cloud is called Cloud-Instruction Set (CIS). Each node in the system registers itself to execute some or all of the CIS's of a CCPU. Since multiple nodes can provide this service, there is a great deal of fault tolerance and parallel execution in the system. Each node can serve to multiple CCPU's with a subset of CIS depending its capacity and load. Since the instruction set is user defined and not forced by the hardware anymore, complicated high level instructions can also be implemented as part of the CCPU like sorting or like any other high level data manipulation.

4.3 Cloud Register Set (CRS)

Each instruction may need temporary storage (i.e. registers). The register service provided by the cloud is called Cloud Register Set (CRS). Since the cloud provides a data storage service already to its clients, CRS can use the underlying data storage services of the cloud system. CRS will be associated differently to each CCPU. High level register types can be created like queue types or any other high level storage types, since the register design is not limited by the hardware.

4.4 Cloud CPU Compiler (CCC)

The compiler will have two basic functionalities in the system. The first one is to compile the given program (decide sub-services needed to finish the required computation) using the specified CCPU as the target. The other functionality, different from a regular compiler, is to decide the instruction set required to create a CCPU for the given program.

4.5 Cloud Program Counter (CPC)

Each computation or process needs a program counter, so that the system can calculate the next instruction (sub-service) to execute. Using the distributed data storage service of the cloud, the Cloud Program Counter (CPC) is implemented easily. During the creation of the CCPU, nodes will register themselves to store and execute the CPC where the CPC created is associated with the process through a Cloud Process Descriptor (CPD). Since the CPC information is shared among the registered nodes for this service, it will provide the required fault tolerance. If one node fails, the other registered node will continue the execution.

4.6 Cloud Process Descriptor (CPD)

There is a need to identify each process that executes in the cloud. Also CPC should keep track of the process in the cloud. It can locate the process using CPD. This information again is saved in the cloud by nodes who registered themselves to provide the needed service.

5 CLOUD CPU OPERATIONS

There are several operations that may take place in the system. The user will decide to the desired service (represented by compiling a program) using a specific CCPU as the target using the compiler CCC and submit the program to the cloud as a Cloud Process (CPR). In the cloud, a CPC will be formed to execute the CPR. The operations required in the system are explained below.

5.1 Cloud Program Compilation

The program compilation is very similar to a regular compilation. Each compiler converts a programming language to the assembler program using the hardware platform that the compiler is running on. If you are running the compiler on an I7 Intel processor, the compiler will use the instruction set of I7 CPU in the conversion. The difference in Cloud Process Compilation (CCC) is that the CPU hardware is not fixed anymore and the target CCPU can be changed for each compilation and most importantly it can be created from scratch. There are regular cross-compilers for different targets but still those target CPU's are fixed and you need to move the compiled program to the corresponding platform to run. On the other hand several CCPU types can coexist and run in this structure. The flow diagram is given in Figure 3.

The decision for the target CPU can be handled in different ways. The first decision is whether a new CCPU should be created for compilation or an existing one should be used. The user may also specify the target CCPU. Depending on the level of the similarity, either the compiler will use an existing CCPU type and compiles the program for it, or it creates a new CCPU and the program is compiled for the new CCPU type created. If the decision is left to the compiler, the decision should be based on the programming language to be compiled, cloud capacity, complexity of the new CCPU needed and the time for the compilation for an existing CCPU. The decision can change depending on a system and it is not hard-wired.

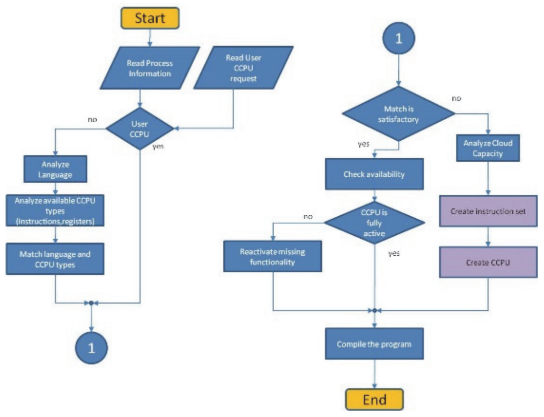

Figure 3: Cloud Program Compilation.

5.2 Cloud Instruction Set Creation

Determining the instruction list and storage types of the CCPU is a difficult process. There are almost infinitely many combinations, since the virtual hardware is flexible and it can behave as you wish, the instructions are not limited by the hardware anymore. In addition to the number of combinations, there are other factors as the interconnection speed, storage capacity and processing speed of nodes. An exhaustive list cannot be created here as it will be out of scope of this paper.

The creation of the CCPU process can be initiated by the compiler as well as by the user. A basic guideline can be given as follows and shown in Figure 4.

- The programming language is analyzed.
- A decision will be made if the language should be implemented as it is or it should be converted to a different language. There are several examples to this type of decision. We can have a microcontroller that can execute BASIC language directly. The second choice is to convert (compile) the BASIC language to a more basic RISC type language with few basic machine instructions.
- The cloud will be analyzed in terms of capacity, speed and availability.
- It is possible to have predefined templates for a language carefully designed by a user.
- The compiler will check these templates if there is a close match to the current language. It can decide to use one of the templates or create a new one.
- If a new template is needed by the compiler, the scope will be limited to the instructions in the current program. A new implementation may not cover all possible constructs of the language, since the compiler scope is limited by the current program. On the other hand, a user created template will cover the whole language elements.
- If the decision is to create a new instruction set, in that case the compiler will create a list of needed instructions using a Cloud Instruction Description (CID). Type can be class, function, statement, arithmetic or other constructs necessary. The description should list the required functionality as an operation or as a pseudo algorithm. Complexity can be given or can be calculated from the algorithm. Required storage types should also be listed. This information is needed so that each node in the cloud can respond to the request.

5.3 Cloud CPU Creation

Once the CCPU type is determined, the node performs two different actions depending on the state of the CCPU explained below and given in Figure 5.

- Use an existing CCPU

The initiating node will check if all the nodes registered to execute this CCPU are still active or not. If not, new nodes are invited to join to execute the missing instructions and/or architecture.

- Create a new CCPU

The instruction list created by the compiler for a new CPU is sent as a broadcast to the cloud to invite interesting nodes to implement the required instructions. Systems with desired properties to implement the required functionality will respond and register themselves to execute a specific function. For example a system with high speed storage may respond to create the registers and/or queues, and another system with high speed processing units may respond to implement a complicated instruction in the CCPU

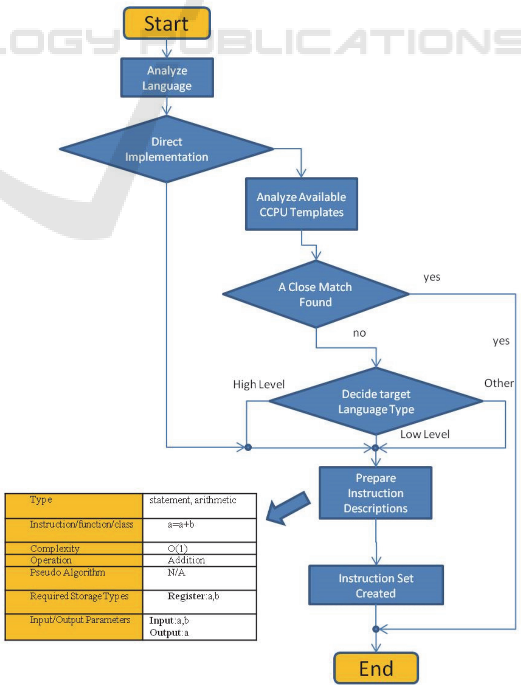

Figure 4: Cloud Instruction Set Creation.

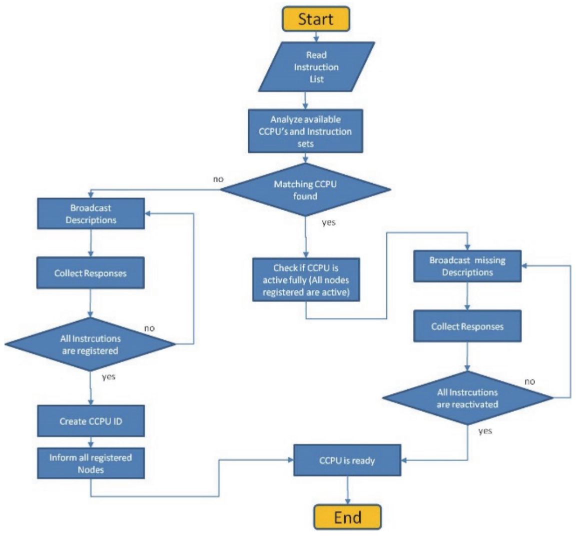
Start
Read Instruction List
Analyze available CCPU's and Instruction sets
Matching CCPU found
no
yes
Broadcast Descriptions
Collect Responses
All Instrcutions are registered
no
yes
Create CCPU ID
Inform all registered Nodes
Check if CCPU is active fully (All nodes registered are active)
Broadcast missing Descriptions
Collect Responses
All Instrcutions are reactivated
no
yes
CCPU is ready
End

Figure 5: Cloud CPU Creation.

6 CONCLUSIONS

In this paper, a new idea to design and implement a configurable service (virtual CPU) using the cloud architecture is introduced where the units of a CPU core represent nodes or services provided by a cloud system and the data transfer between them is replaced by a data transfer inside the cloud system. The CPU functionality becomes a service provided by the cloud system.

By pushing the CPU units to a cloud system, a giant virtual CPU is created where each unit of the CPU can have multiple implementations, support, and parallel execution and fault tolerance. The basic element of the new cloud service is the CPU. This new service will be called as a virtual CPU or more accurately as a Cloud CPU (CCPU).

The granularity of the service provided by each node is minimized which means that the cloud provides only the basic functional blocks and instructions of a CPU. This minimization has several advantages and the motivation can be listed as follows.

- **Fault tolerance**: The first advantage is fault tolerance where a node failure only will affect a single instruction or register, not the whole computation (which would have been provided by that node in a normal cloud system). Since the provided functionality is minimal, failure of a CCPU service or node can be replaced easily by another node.
- **Security**: Each node, executing only a few instructions, does not see the whole program or computation. The result of a single computation or instruction on a node does not mean anything at all by itself. This feature protects the process from attackers on top of the normal security measures.
- **Scalability**: The functionality provided by each node to a specific CCPU is minimal; therefore any required expansion and/or reduction can be done very easily and quickly. It won't be that easy to add a large server with a computational service to the system.
- **Computational Requirements**: Since only small and minimal functions are implemented in the cloud, nodes with minimal computational power can join to the system. This is getting more important as very small units are getting connected to the internet (internet of things).
- **Interoperability**: To provide a service by a node in a cloud system, the service should be written by a programming language supported by the operating system of that node. Different types of nodes should agree on the data format and/or calculation to distribute the computation to several nodes. In the proposed system, the node needs to provide a minimal computation where the format is decided during setup for that CCPU. Therefore even though each node may have a different OS, providing the functionality would be very easy.
- **Optimization of resources**: Adding or removing servers on demand creates large fluctuations in a cloud system where the resources may be over determined or under determined respectively. Since each resource comes with a cost attached to it, a better optimized resource structure will result in a better optimized budget.
- **Heterogeneous CPU implementations**: Since each node contributes to CCPU using minimal functionality, many virtual CPU's can be generated at the same time depending on the capacity of the cloud system registered for the service. Each CPU can have a completely different set of instructions and/or registers and other units different from each other. The cloud can create a Pentium CPU and a SPARC CPU at the same time. Instead of using a design created before, each time a new CPU (on-demand CPU) can be created in the system as well. The new design can be reused later or deleted.
- **Hardware independent virtual CPU**: Although each regular CPU is different in hardware, a high level language can hide this complexity

and difference from the user by converting the high level language into a specific CPU assembly language. The hardware differences are hidden from the programmer. Using a similar concept, a virtual CPU can be implemented by using any combination of simple or complicated instructions and data storage units (registers, queues, sorted lists) and the underlying hardware is hidden from the user. A SPARC system can provide an "ADD" instruction for the virtual CPU where a PENTIUM CPU can provide a "SORT" instruction. The virtual CCPU is hiding the hardware details like how an instruction is implemented and it is focusing on what is implemented.

- **Process and/or programming language specific virtual CPU**: The virtual CPU can be created on demand or permanently in the system. Depending on the process, either a specialized virtual CPU is created and the process is executed on it, or a general virtual CPU is created and/or reused to execute the process. With this implementation it is possible to create a C-specific virtual CPU.
- **Parallel Operation**: The Cloud CPU creates a massively parallel system where normal systems are limited in terms of number of cores and/or CPU's.

REFERENCES

Youseff, L., Butrico, M. and Da Silva, D., 2008. Toward a unified ontology of cloud computing, *Grid Computing Environments Workshop (GCE '08).*

Vaquero, L., Rodero-Merino, L., Caceres J., and Lindner, M., 2009. A break in the clouds: towards a cloud definition. *SIGCOMM Computer Communication Review*, Volume 39, Number 1, pp. 50-55, January.

Vouk, M., Cloud computing- issues, research and implementations, 2008. *Journal of Computing and Information Technology (CIT)* Volume 16, Number4, pp. 235–246.

Sonisky, B., 2011. Chapter 1, *Cloud computing bible*,Wiley publishing inc.

Zhang, Q., Cheng, L., and Boutaba,R.,2010. Cloud computing : state of the art and research challenges. *Journal of Internet Services and Applications*, vol. 1, pp. 7–18.

Dillon, T., Wu, C., and Chang, E., 2010. Cloud Computing: Issues and Challenges, *in 24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 27-33.

Hashizume, L., G Rosado, D., Fernndez-Medina, E., B Fernandez, E., 2013. An analysis of security issues for cloud computing, *Journal of Internet Services and Applications*, Page no 1-13.

Foster, I., Zhao, Y., Raicu, I., and Lu, S., 2008. Cloud Computing and Grid Computing 360-degree compared, *in Grid Computing Environments Workshop*, pp. 1–10.

Liu, Y., Ma, Y., Zhang, H., Li, D., Chen, G., 2011. A Method for Trust Management in Cloud Computing:Data Coloring by Cloud Watermarking, *International Journal of Automation and Computing*, DOI: 10.1007/s11633-011-0583-3 ,Page no 280-285

Basu, A., Vaidya, J., Kikuchi, H., Dimitrakos, T., and Nair, S., 2012. Privacy preserving collaborative filtering for SaaS enabling PaaS clouds, *Journal of Cloud Computing: Advances, Systems and Applications*, Page no 1-14.

Buyya, R. Shin Yeo, C., Venugopal, S., Broberg, J., Brandic, I., 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, *Future Generation Computer Systems,* 25 ,Page no 599 616.

Popovic, K., Hocenski, Z., 2010. Cloud Computing Security Issues and Challenges, *MIPRO* ,May 24-28,Opatija, Croatia, Page no 344-349.

snia.org, *Cloud Data Management Interface* (CDMI), Available at: http://www.snia.org/techactivities/standards/currstandards/cdmi, accessed on 9.8.

Petcu, D. Di Martino, B., Venticinque, S., Rak, M., Mhr, T., Esnal Lopez, G., Brito, F., Cossu, R., Stopar, M., 2013. Experiences in building a mOSAIC of clouds , *Journal of Cloud Computing: Advances, Systems and Applications*, Page no 1-22.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, S. So-man, G., Youseff, L., and Zagorodnov, D., 2009. The Eucalyptus open-source cloud-computing system, *IEEE International Symposium on Cluster Computing and the Grid* (CCGrid '09).

opennebula.org, *Open Nebular*, Available at: http://www.opennebula.org.

nimbusproject.org, *Nimbus*, http://nimbusproject.org.

Khan, A., Othman, M., Madani, A., and Khan, A., 2014. A Survey of Mobile Cloud Computing Application Models, *in IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 393-413, First Quarter.

Botta, A., Donato ,W., Persico V., Pescapé , A., 2016, Integration of Cloud computing and Internet of Things: A survey, *Future Generation Computer Systems*, Volume 56, , Pages 684-700, March