July 2012

EFFICIENT COMPUTATIONAL STRATEGY FOR CLOUD COMPUTING ENVIRONMENT

DEEPTI THENG

G.H.R.C.E., Nagpur, deepti.theng@gmail.com

L. G. MALIK

G.H.R.C.E., Nagpur, lgmalik@yahoo.com

S. U. NIMBHORKAR

G.H.R.C.E., Nagpur, nimsonali12@yahoo.com

Follow this and additional works at: <https://www.interscience.in/ijcct>

Recommended Citation

THENG, DEEPTI; MALIK, L. G.; and NIMBHORKAR, S. U. (2012) "EFFICIENT COMPUTATIONAL STRATEGY FOR CLOUD COMPUTING ENVIRONMENT," *International Journal of Computer and Communication Technology*: Vol. 3 : Iss. 3 , Article 13.

DOI: 10.47893/IJCCT.2012.1143

Available at: <https://www.interscience.in/ijcct/vol3/iss3/13>

This Article is brought to you for free and open access by the Interscience Journals at Interscience Research Network. It has been accepted for inclusion in International Journal of Computer and Communication Technology by an authorized editor of Interscience Research Network. For more information, please contact sritampatnaik@gmail.com.

EFFICIENT COMPUTATIONAL STRATEGY FOR CLOUD COMPUTING ENVIRONMENT

DEEPTI THENG , L. G. MALIK & S. U. NIMBHORKAR

G.H.R.C.E., Nagpur

Email: deepti.theng@gmail.com, lgmalik@yahoo.com & nimsonali12@yahoo.com

Abstract – The problem of distributing virtual machine (VM) images to a set of distributed compute nodes in a Cross-Cloud computing environment is considered as the main issue in this paper. Here the term Cross-Cloud Computing refers to the environment with multiple cloud sites. Self Scheduling schemes are mostly used for high performance as they represent adaptive schemes where allocation is done in run time. From review it is observed that Factorizing Self Scheduling (FSS) offers better load balance when execution time changes widely and randomly. Here, Heterogeneous Factorizing Self-Scheduling (HFSS) algorithm is proposed and considered for the implementation of virtual machine scheduling in cross-cloud computing environment. Such an implementation is not yet done with cross-cloud platform which is proposed here. Another feature of the project is that, data sent by user for computation on cloud environment is stored in encrypted form. Data Encryption Standard (DES) is used for Integration of storage encryption & decryption.

Keywords-virtualization, cloud computing, virtual machine (VM) image, cross-cloud computing, scheduling, DES.

I. INTRODUCTION

Cloud computing [1] [2], the current emerging trend in delivering IT services, can address many of the problems. By means of virtualization technologies, Cloud computing offers to end users a variety of services covering the entire computing stack, from the hardware to the application level, by charging them on a pay per use basis. Another important feature, from which scientists can benefit, is the ability to scale up and down the computing infrastructure according to the application requirements and the budget of users. Scientific computing involves the construction of mathematical models and numerical solution techniques to solve scientific, social scientific and engineering problems. These models often require a huge number of computing resources to perform large scale experiments or to cut down the computational complexity into a reasonable time frame. These needs have been initially addressed with dedicated high-performance computing (HPC) infrastructures such as clusters or with a pool of networked machines in the same department, managed by some “CPU cycle scavenger” software such as Condor [3].

Any information is available anywhere in the world at any time. That was not possible few years ago. Cloud Computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about over-provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under-provisioning for one that becomes wildly popular, thus missing potential customers and revenue. [4]

This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT.

Nowadays a lot of enterprises or users need to process a long amount of information data or need to do some complex operations, for example some mathematical models calculations, and they need a high amount of process power to resolve it. Maybe the power of a personal computer can be not enough to finish in a determinate time one task and maybe this task is so punctual that is pointless to acquire the hardware necessary to do it. Also sometimes a user wants to use a program for few times, for example for making some practices in university, and maybe it's pointless to make install this user a program, like Scilab, so for both problems it is convenient to bring the possibility of use some programs without the needing of install in own computers and with the possibility of have more calculation power. So there will be necessary schemes to be able of executing a problem in the cloud of the university with all fasted way as they expect from their desktop systems.

The following problems are considered to be solved:

- The primary goal is efficient VM image distribution of users job
- Achieving a good workload distribution
- Providing security to the users data

The new concept of Cloud Computing offers dynamically scalable resources provisioned as a service over the Internet and therefore promises a lot of economic benefits to be distributed among its adopters. Along with the benefits described in [5], Cloud Computing also raises severe concerns especially regarding the security level provided by such a concept. . Eight encryption methods were compared based on P-value, Rejection rate and finally

based on time consuming for each method in [6]. Each of the algorithms was evaluated on desktop environment and cloud environment by using encryption algorithms; it includes RC4, RC6, AES, DES, 3DES, MARS, Two-Fish, and Blow-Fish. From the results of [6], the evaluation of eight modern encryption techniques shows that RC6, AES, DES and Blowfish results were slightly better than other-encryption methods. But, Blow-Fish and DES is more suitable when a focus is on time of encryption method.

This paper is an extension of [7], in which cloud environment creation and execution of proposed work is given using windows azure cloud platform.

II. PROPOSED SYSTEM APPROACH

Depending upon a analysis of different Virtual Machine (VM) load balancing algorithms from previous chapter a new VM load balancing algorithm has been proposed here and implemented in windows azure Virtual Machine environment of cloud computing and also tested on LAN desktop scenario in order to achieve better response time and cost. In this detailed execution plan is given for a job.

Whenever a request comes from a client it will be first accepted at the appropriate cloud server. At this local cloud site the incoming request will be served by the scheduler to better match the requirement at that cloud server. This is elaborated in the following steps:

- Client sends a request to perform some computation on cloud.
- First it goes to the respective serve requested for computation.
- If the request is given to a cloud server then it is scheduled to the best computational node (actual compute nodes) with the help of proposed algorithm by the scheduler.
- Once the request reaches the desired computing node, then the client-server gets connected and computations are carried out.
- This process goes on continuously for all the incoming requests.

This workflow is followed smoothly with the peer-to-peer computing network. This network provides high scalability, as it is possible to add more peers to the system and can scaled to larger network [8]. Thus it is good for large scale computations.

On one cloud server of student management, the two servers will hold two different databases for students as a storage computation. The incoming request will be partitioned and will be getting computed on the respective servers holding corresponding required database. In this way request will be partitioned and effective load balance at

heterogeneous environment will be achieved. System will be evaluated by such a real life applications at each cloud server.

A. System Architecture

System Architecture for LAN scenario of Cloud Environment is best depicted in figure 1. It is divided into three parts as shown below. In first section all the users are shown, which are accessing computing services from the cloud servers. A cloud is formed consisting of a server and m-different computing nodes which are running at server side.

Fig. 1 LAN scenario of Cloud Environment

A scheduler is executed at server node upon accepting computing request from users. This is the simple scenario detailed execution at each of these is given below.

At Cloud Server:

Fig. 2 Architecture for Server

At first server application should be ready and executed to start the computations. In figure 2, all the processes executed at server side are shown in there execution sequence. There is an Application Manager who will manage execution of all these processes at server. Three main managers which are controlled by

the Application Manager are Communication Manager, GUI Manager and Task Transfer Manager. Communication Manager is used to communicate server with computing nodes and users. It runs two processes i.e. sender and receiver. At receiver it will execute Welcome Protocol, Goodbye Protocol, SystemInfo Protocol, and Task Transfer Protocol. These all are executed through multi-socket programming. Whenever server receives a welcome protocol, it will execute a GUI manager to enter the connected Computing Node or User to their respective list which is displayed by GUI Manager on the Server Main Window. When SystemInfo protocol executed by Computing Nodes, at server side it will receive continuous updated system information of computing nodes. This information is stored first on to a database and then given to GUI manager to display on Server Main Window. Similarly, when user asks for computation Task Transfer Protocol will be executed by the User. Upon receiving it at server, server will activate the Task Transfer Manager. Task Transfer Manager will take the information of CPU utilizations of all connected and running computing nodes from GUI manager. Then list is arranged in an ascending order of CPU utilization, so that all Computing Nodes with minimum CPU utilization are placed first in the list. Depending upon the factorization of the task first few required computing nodes from the list are selected for the computation. The response of task is accepted and sent by the Task Transfer Manager through sender process multi-socket programming. Lastly whenever any computing node or user gets exit, the Goodbye Protocol is invoked. Upon receiving request through the Goodbye protocol from Computing node, the server will disconnect the respective computing node as well as all the entries of this node is get deleted from the database and computing node is removed from the list of current running computing nodes displayed on Server Main Window. When same is received from User, the server will disconnect services provided to the respective user and delete its entry from the Remote User's list displayed on Server Main Window.

At Computing Node:

Fig. 3 Architecture for Computing Node

Figure 3 shows the processes that will be executed at computing nodes of server. There is a client application manager for each of the computing node who will manage all these processes running at respective computing node. These processes include sender application, client RMI and client monitor tray for Computing Node. Client RMI is responsible to handle the services that are provided to the Remote Client i.e. User. The called service among factorial, square root, average, etc. will get computed by the selected computing node on users' data. At sender side various protocols are designed and called according to the computing node interaction. When firstly the computing node is started, it will call the welcome protocol. So the computing node is connected to the server and appears in the list of Remote Computers i.e. the list of computing nodes running at server side. After successful connection, immediately it will run SystemInfo protocol, through this computing node will send its system information to server. There is a Client System Monitor Tray, it will continuously track the changes in system information of computing node, i.e. CPU usage, RAM usage etc. The changed and updated information is continuously forwarded to server through the SystemInfo protocol. Thus server will have a continuous track of current system information of all connected and running computing nodes.

At User System:

Fig. 4 Architecture for User (RClient)

Remote Client is considered as User here. Thus RClient Application Manager is executed at User side which controls all processes running at User. Similar to Computing Node, User will run Communication Manager to communicate with the server. It has two processes running i.e. sender and receiver. Through the sender Welcome Protocol, Task Transfer Protocol and Goodbye Protocols are sending to server. The response of the selected tasks are accepted at receiver and displayed through the GUI Manager. There is a Client RMI invoked as soon as user requested for the computations. All the offered computations and provision for data is shown on Task Placement Frame running at user side.

In LAN scenario of cloud environment, a user is facilitated with Task Placement Frame. This frame

will display all computing services available at cloud server. There is a provision to accept users data on which user wants to perform computing services. Here, now computing services considered are finding factorial, calculating square root and calculating average of list provided by the user.

If user wants to do only one computation, then a first minimally loaded computing node is selected by the server and computations will be performed. When more than one computation included in the users task, then first n computing nodes with minimum CPU utilization among all are selected and these computations are performed one on each. Server collects the result and given to the user who invoked Task Transfer Protocol. Computing nodes are already sorted in ascending order of their CPU utilization and computations are allocated according to the descending complexity of required computation.

B. Heterogeneous Factoring Self Scheduling (HFSS)

In order to achieve the load balance and reduce the execution time of each node in the cloud computing environment, this study proposes modified algorithm called Heterogeneous Factoring Self Scheduling (HFSS). The HFSS scheduling guarantees that suitable service nodes will be assigned to execute the task in the minimum execution time.

As VM workload i.e. users computational task is assigned to a node which is minimally loaded, this section describes the method of selecting minimally loaded node. A computing node is said to be minimally loaded if its CPU utilization is minimum among all available computing nodes.

Algorithm: Following are the algorithm steps to get the minimally loaded computing node.

Input:

- A non empty list of client computer (CCL)
- A non empty list of Rclient computer (RCL)
- Client CPU percentage (CCP)
- Task to be performed(TP)

Output:

Response to the task as per CCP utilization and task complexity.

While(true) (Repeat through Server is running)

If(TP !=null and CCL!=null) then

Find complexity of each task

Create CPU percentage list of computing nodes in ascending order

Create hash-map to Map the time complexity and CPU percentage list to get the desired client name.

End if

If(task is factorial) then

with the help of fact Key get client name from the hash-Map

Transfer the task to the specific hash-map client

End if

If(task is avg) then

with the help of avg Key get client name from the hash-Map

Transfer the task to the specific hash-map client

End if

If(task is sqrt) then

with the help of sqrt Key get client name from the hash-Map

Transfer the task to the specific hash-map client

End if

Combine the result of each operation and transfer response to the specific Rclient

End while

In order to reach the load balance and reduce the execution time of each node in the cloud computing environment, this study proposes. The HFSS scheduling guarantees that suitable service nodes will be assigned to execute the task in the minimum execution time.

The modified and proposed algorithm is as below:

At Scheduler:

- Accept VM request of user at scheduler
- Scheduler will detect corresponding computing nodes in ascending order of their CPU utilization
- Factorize the tasks according to the computations involved as well as complexity of the required computation
- As mentioned in above algorithm steps, scheduler will map minimum loaded computing node with the computational task complexity.
- Based on the mapping result task are transferred to the respective computing nodes.
- Computational result will be accepted from corresponding computing node and provided to the user.

The process continues for the each coming request

At Compute Node:

- Update its system information and send it to the server.

- Compute node assigned for the task will track its execution time upon accepting the request till it gets completed.
- After successful completion of the task execution, computing node will send computation result along with the time required to compute it.
- And it will update its system information, also sends same to the server.

This partitioning provides better load balance as both computing node performance as well as task complexity is matched for better performance. It will boost the performance of cloud server.

Storage Encryption and Decryption

In paper [9], three attacks on online storage services were described. It includes, Identifying Files, Learning the Contents of Files and A Covert Channel. The authors of this paper suggested that, clients of cloud storage services, regardless of the service they use, can stop it by encrypting their data before the storage service's client software operates on their files. DES is selected for implementation by taking into mind the results of previous research work and major design objectives of the DES which best suited for the requirements.

Data Encryption Algorithm: DES uses the two basic techniques of cryptography - confusion and diffusion. At the simplest level, diffusion is achieved through numerous permutations and confusions are achieved through the XOR operation [10].

- Substitution-Permutation:
 - ✓ 64-bits input and output blocks
 - ✓ 56-bit key with additional parity bits
- Information data is cycled 16 times through a set of substitution and permutation transformations: highly non-linear input-output relationship
- Very high throughput rate achievable

The main parts of the algorithm are as follows:

- Fractioning of the text into 64-bit (8 octet) blocks;
- Initial permutation of blocks; Breakdown of the blocks into two parts: left and right, named L and R;
- Permutation and substitution steps repeated 16 times (called rounds);
- Re-joining of the left and right parts then inverse initial permutation.

DES has 16 rounds, meaning the main algorithm is repeated 16 times to produce the cipher text. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a

key using a brute-force attack. So as the number of rounds increases, the security of the algorithm increases exponentially [11].

III. EXPERIMENTAL RESULTS

Various outputs for LAN cloud scenario are shown and explained in this section.

Fig. 5 Server Flash Window

This is just a flash window displayed at server side upon startup. Indicates the start of cloud server, and view is made so.

Fig. 6 Server Main Frame Window

This is a main window displayed at server system. It is divided into three parts as shown above. One is to display list of computing nodes currently available at server side. It is shown in Remote Computers section. Second is Remote User, which displays the list of current users accessing cloud. In third window system information like Computer Name, CPU Model, CPU Cores, Operating System, RAM Size, CPU Usage, and RAM Usage etc. for respective computing nodes is displayed as a table. Out of which CPU usage of all computing node is used to find minimally loaded node. Here it is showing two computing nodes running and a user is running.

Fig. 7 User Main Window

This is the main window displayed at user side. It includes options for Task Placement and Exit under System Tag. Task Placement option is used by user to place a task for computations at cloud server. Various options for task placement and data are given in next window of Task placement. Other is Exit option which is used to disconnect user and cloud server connection. Upon exit, it deletes all information about users send to cloud server and list of current users at server side is updated.

Fig. 8 User Task Placement Window

This is the task placement window provided at user side. As shown above user can perform three different computations at cloud server which includes, calculating factorial, calculating average, calculating SQRT. As an input user has to select respective tasks and provide corresponding data for execution of selected tasks. At output three parameters will be displayed including, result of selected task for selected data. It also displays the client name, this is the name of computing node at server side, to which the task was transferred for execution/to do computations. The time required to execute task at computing node is also displayed in milliseconds.

Fig. 9 Three tasks selected for execution by user

In above figure 9, user has selected three tasks for execution. The status of server when a complete job was received at server is shown in below figure 10, where there are two computing nodes running. According to status first computing is having large CPU utilization than second computing node. Thus tasks were scheduled according to the ascending order of computing node's CPU utilization.

Fig. 10 Server Status

CONCLUSION

From previous study a Heterogeneous Factorizing Self-Scheduling algorithm is proposed and considered for the implementation for virtual machine scheduling in cloud environment. Such an implementation is not yet done which is proposed here. For a better test performance, a LAN scenario of cloud environment is created and HFSS is evaluated on the same. The results of evaluation shows that, the factoring and computing a task with better CPU utilization and complexity consideration boosts the performance of cloud server. Thus, the objective of distributing virtual machine i.e. users task to the physical computing node on cloud computing environment is achieved successfully.

While executing complete job on one computational node requires more time for execution. If the same is performed on this system with HFSS,

less execution time is required with faster response. This will also help to better utilization of cloud resources, solving the problem of underutilization and overutilization of computing resources. In addition to the proposed HFSS, another feature added to this project is storage encryption and decryption. This is just considered as the good feature and implemented using better encryption results from previous research on cloud computing.

Further factoring tasks will be a good research work. In future one can do research to solve following problems

- Solving task dependency and factoring large tasks for further execution on multiple computing nodes.
- Implementation of more types of application programs
- Strong analysis and integration of strong encryption and decryption.

REFERENCES

- [1] Michael Feldman. Sun Boosts HPC On-Demand Offering. <http://www.hpcwire.com/topic/networks/17909554.html>, 2008.
- [2] Apache, "Hadoop", <http://lucene.apache.org/hadoop/>, 2006.
- [3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud computing. Technical Report No. UCB/EECS-2009-28, University of California at Berkeley, USA, Feb. 10, 2009.
- [4] Wen-Chung Shih, Shian-shyong Tseng, "Performance Study of Parallel Programming on Cloud Computing Environment: Using MapReduce", IEEE Computer Society, 2010.
- [5] Freeman TL, Hancock DJ, Bull JM, Ford RW., "Feedback guided scheduling of nested loops", Proceedings of the 5th International Applied Parallel Computing (PARA) Workshop, Bergen, Norway, 2000 (Lecture Notes in Computer Science, vol. 1947). Springer: Berlin, 2001; 149–159.
- [6] "4 Cloud Computing Security Policies You Must Know". CloudComputingSec. 2011, [Online]: http://digg.com/news/technology/4_cloud_computing_security_policies_you_must_know_cldo_computing
- [7] Deepti Theng, K.N.Hande, "VM Management for Cross-Cloud Computing Environment", IEEE International Conference on Communication Systems and Network Technologies (CSNT), Page(s): 731-735, 2012
- [8] Ian T. Foster, C. Kesselman, S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations", In Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing, pages 1–4, London, UK, 2001. Springer-Verlag.
- [9] Danny Harnik, Benny Pinkas, Alexandra Shulman-Peleg, "Side Channels in Cloud Services Deduplication in Cloud Storage", Copublished by the IEEE Computer and Reliability Societies, November/December 2010.
- [10] Coppersmith, D., Thomas J., "The Data Encryption Standard (DES) and its strength against attacks", IBM Journal of Research and Development, Volume: 38, Issue: 3, Page(s): 243-250, 1994.
- [11] Mandal, A.K., Parakash, C., Tiwari, A., "Performance evaluation of cryptographic algorithms: DES and AES", IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), Page(s): 1- 5, 1-2 March 2012