PoHM: A Novel Approach for Managing QoS in Live Transcoding using Cloud based Heterogeneous Map Reduce Model

Preetha Evangeline D, Farzana Banu K, Dr. Anandhakumar Palanisamy, Cephas Paul Edward

Department of Computer Technology, Anna University, Chennai

***Abstract*— Performing compute intensive task with minimal completion time along with predefined Quality of Service is one of the challenging issues in Real time multimedia processing. This can be achieved by cloud Mapreduce computational operations which perform parallel operations in Virtual machines. The problems addressed in this paper are the procurement problem which states the number of heterogeneous VMs to be procured to run Mapreduce program to complete the execution of jobs with minimal completion time and cost. The next problem is the placement problem where the laborers (Mapper/Reducer) have to be assigned to the available virtual machines in such a way they utilize even the residual resources available to meet the QoS. The paper proposes a Novel (PoHM) Procurement of Heterogeneous Machine algorithm which procures heterogeneous VMs to run Mapreduce based upon the requirements of the task needed to complete the execution. The highlight of the proposed work is, the residual resources of the already procures VMs are utilized before renting new VMs on demand which reduces the overall computational cost. Our proposed PoHM algorithm was enhanced with Best-Fit strategy (BFS) which proved around 39% better result when compared with Homogeneous algorithm, 17% better than RFS and 14% better than FFS algorithms.**

***Index Terms*— Cloud based MapReduce; Live Transcoding; parallel processing; Heterogeneous VMs; Quality of Service**

INTRODUCTION

Hadoop Mapreduce is a model which performs parallel operations for processing the data independent of the specific applications. As Multimedia data are usually heterogeneous and of large size, it has to be efficiently processed and especially for live multimedia streaming data has to process with high speed on the fly to guarantee QoS. Mapreduce operations are classified into two phase which is the Mapper phase and the reducer phase. Mapreduce was traditionally introduced for performing parallel processing in cluster which consist a group of systems connected together. The main objective of parallel processing is to minimize the computation time and to utilize the available resources efficiently [15] [16] [17] [18] [19] [20] [1]. Zhao et al performed data mining using k-means clustering. Mapreduce was enhanced with k-means clustering to perform parallel operations which proved better results [11]. Matsunga et al proposed Cloudblast where VMs are used to perform Mapreduce operations for bioinformatics application [12]. He et al in his work based on regression and machine learning used Mapreduce for parallel learning [13]. Husain et al used Mapreduce for parallel query processing of large RDF graph [14]. The most important challenge associated with Mapreduce parallel computation is to guarantee Quality of Service with cost effective procurement of VMs. More the resources rented/leased from the cloud, more the cost will be.

In order to satisfy the Quality of Service of a particular application, the set of tasks associated with it has to be placed in the VMs and the computational resources required for the execution must be provided with minimal cost. This is the procurement problem for which the solution is proposed in this paper. Procurement of VMs is of two types, (i) Homogeneous procurement and (ii) Heterogeneous procurement. In homogeneous procurement, the VMs that are rented from the cloud are of the same type, same specification and equal set of Mappers/Reducers are placed in VMs. This type of procurement is easy to implement since the size of the task and its resource requirements are not considered [2] [3] [4] [5] [6] [7] [8] [9]. During recent days, heterogeneous procurement of VMs has come into practice [10]. The proposed methodology uses heterogeneous procurement where the number of tasks assigned to a VM differs and according to the requirement of resource to complete the execution, tasks are being mapped to the VMs of varying types. Intelligent decision has to be taken in order to place suitable number of laborers (Mappers/Reducers) to VMs with varying resources (CPU or cores and Memory capacity). It is proved that the procurement of Heterogeneous VMs is cost effective and guarantees QoS when compared with procurement of homogeneous machines.

The paper proposes a Novel procurement of Heterogeneous machine (PHM) algorithm which runs Mapreduce program. This algorithm limits the procurement of new VMs from the cloud as well as utilizes residual resources of the procured VMs efficiently. To guarantee QoS with minimal completion time, Best Fit Strategy (BFS) is included with the proposed algorithm. The capacity of the VMs are estimated based on the CPU (or cores) and the memory which is responsible for computational speed. The virtual machines that are rented from the cloud are classified as smaller and larger VMs and based upon the requirements, these VMs are rented. Say for instance a small VM consists of three CPU and its price is $3/hr and for a large VM which consists of six CPU and its price is $6/hr. An

execution of job requires four tasks to be completed with two CPU's for each task, hence when we go in for homogeneous VMs, the requirement is either four small VMs or two larger VMs and the computational cost comes upto 12-16&. But when heterogeneous VMs are considered we would need one large VM and one small VM to complete the task and the cost comes around 10$.

In our proposed work the residual resources (CPU's & memory) are reutilized when new job has to be executed and only if there is a demand new VMs are adopted from the cloud. The proposed PHM algorithm is compared with homogeneous approach, first fit strategy and random strategy which shows the proposed algorithm outperforms than other approaches.

The rest of this paper is organized as follows: Section 2 describes the detailed Literature survey of the problems handled by MapReduce both is cloud based and non cloud based environment, section 3 briefs the problem formulation and the constraints represented by mathematical equations, section 4 explains the Proposed PoHM Algorithm and its procedure, section 5 gives the Experimental set up, section 6 discusses about the results obtained and graphs were plotted comparing the proposed algorithm with other baseline algorithms and section 7 concludes the summary of the paper with performance improvement.

RELATED WORKS

This section explains a detailed literature survey on the issues in non cloud based and cloud based Mapreduce job execution.

A. Map Reduce without cloud

Zaharia et al proposed LATE a scheduler for Hadoop Mapreduce in order to minimize the computation time. The traditional Hadoop configuration was carried out where the worker nodes are placed on the computing nodes [16].

Kc and Anyanwu in their work proposed a scheduler for hadoop based on the constraint like deadline that is specified by the user. Two Mapper/ Reducer were placed on every node [41].

Wolf et al proposed a Novel optimizing algorithm to minimize the number of slots assigned for Mapreduce workloads. This optimizing algorithm concentrates on optimizing the execution time while managing minimum number of slots [15].

Verma et al proposed an automatic resource inference and allocation engine designed for Mapreduce programming model to guarantee execution of jobs before its deadline. Here the framework consists of fixed number of workers allocated to every node to perform the operation which imposes overhead and causes underutilization of resources in few nodes [40].

Lin et al in their work gives detailed review on various data scheduling algorithms as well as adaptive tasks. The problem identified in this paper is assignment of slots issue where the slot has to be assigned to the slave nodes which were not mentioned clearly. Traditional hadoop implementation was set up to run Mapreduce in volunteering computing environment [18].

Wang et al proposed a framework for automatic control approach to dynamically assign the slots for tasks on each computing node. Utilization of resources showed improvement in each cluster [31].

Polo et al in his work used the above mentioned dynamic assignment of task slots but the objective here is to meet the deadline of jobs using Mapreduce model [32].

Herodotou et al proposed Starfish which is a s self tuning system to set parameters in a Mapreduce model for optimizing the job execution time. The number of task slots to be assigned in the nodes was decided automatically by the proposed system [19].

Literature survey done so far speaks about the issues such as improving the utilization of resources in a cluster based computing, meet the deadline of the jobs, how to assign slots for the task in a node and how to minimize the execution time. The problem of adopting VMs to guarantee Quality of Service was not discussed and cost efficient computing was not concentrated.

B. Map Reduce with cloud

The reason behind the adoption of cloud technology with Mapreduce is because the number of virtual machines that could be rented or leased on demand from the cloud through infrastructure as a service.

Tian et al in their work explained about provisioning resources for optimizing the cost by deploying a single Mapreduce job in the public cloud [2].

Abdelbaky et al proposed Object driven Scheduler to minimize the nukber of virtual machines rented from the cloud. Although the VMs obtained were heterogeneous systems, one VM was allowed to run only one job which lead to high computational time and increased the number of VMs that has to be rented [4].

Hwang and Kim studied cost effective provisioning of VMs from the cloud. They adopted a simple method of assigning equal number of Mapper/Reducer in every VM. It was cost effective but imposed overhead [5].

Lama et al proposed an automated system for provisioning of jobs in Hadoop Mapreduce model. The system concentrated on procuring VMs to achieve specified QoS with minimal cost. Optimizing the number of workers on a node was not specified in this paper. General rule was followed limited to one Mapper and Reducer each for a smaller VM and two Mapper and Reducer were allocate each for larger VM [6].

Chen et al analyzed a model for cost functioning and derived a relationship among execution time, job size and the available cloud resources to meet the deadline with less cost. Homogeneous number of Mappers and Reducers were placed on the VMs instead studies were made on optimizing the number of VMs to be procured [8].

Herodotou et al proposed Elastisizer along with Starfish to solve the problem of cluster sizing. The system could predict the exact type of VM that can be rented from the cloud to best suit the Mapreduce task [3].

Palanisamy et al proposed resource provisioning model called Cura which estimates the right VM type to run Mapreduce jobs. In order to analyze the job profiles, starfish model was leveraged [9]. Cardosa et al studied the arrangement of VMs on to the physical machine suitable to run the Mapreduce jobs with energy aware constraints. The physical machines were limited and they were compared to bins. The problem solved here was to fill the bins (physical machines) with VMs so as to minimize the energy consumption [7]. And in our paper the problem is much similar but the number of physical machines is not limited and we have considered heterogeneous VMs that could be procured from the cloud.

Xu and Tang et al in their work considered the placement problem of Mapper and Reducer to the virtual machines but the issue which was not addressed in this paper was the spare resources available in the VMs which were not considered for utilization, instead new VMs were rented again from the cloud which was not cost effective [10].

In our proposed work, utilization of Residual resources in the existing VMs was considered for reutilization which reduced the number of VMs to be procured for new set of jobs.

Similar problems based on Mapreduce placement were discussed in [33] [34] [35]. Mostly these problems were compared to bin packing problem but the type of bins were limited and many algorithms were proposed to solve these issues [7] [38] [39].

PROBLEM FORMULATION

In this paper, Real time private cloud was set from where heterogeneous VMs were adopted for running Mapreduce program for the completion of tasks. Since multimedia data was considered to test the computation time, transcoding tasks were performed using Mapreduce. Initially set of workloads were assigned to the available VMs based on the resource requirement and while processing, new set of workloads were queued up for processing. Few VMs inspite of processing the tasks still had free resources (CPU and Memory) where further jobs can be placed and executed. In order to minimize the cost, tasks that were capable of with the available residual resources were allocated on the existing VMs and the tasks waiting in the queue were places on the new set of VMs rented from the cloud.

Let us consider there are 'n1' new workloads waiting to be processed and 'n' be the existing workloads that are being processed on the existing VMs. To guarantee minimum computation time of i^{th} new task ($1 \leq i \leq n$) a minimum of f_i^M Mapper and f_i^R Reducer has to be allocated for execution of i^{th} task with QoS. The VMs running the Mapreduce program are represented with two resources (i) M_i^{CPU} and (ii) R_i^{Mem}. The Mapping phase requires M_i^{CPU} and M_i^{Mem} to perform the operation and the Reducer phase requires R_i^{CPU} and R_i^{Mem} to perform reduce operation of the i^{th} task in a VM.

Let 'A' denote the VMs that were procured earlier and 'Â' be the VMs newly rented from the cloud. VMs with equal capacity (in terms of same CPU and Memory) are grouped together as 'm' groups where $\hat{A} = \bigcup_{j=1}^{m} \hat{A}j$ and $A = \bigcup_{j=1}^{m} A\,j$ which denotes Â and A belong to the same group 'j' with similar capacity and price.

Further A_c is an instance of the VMs to be used where $A_C \in A \bigcup \hat{A}$ and $1 \leq c \leq |A| + |\hat{A}|$. For the instance A, the CPU capacity A_c^{CPU}, Memory A_c^{Mem} and price be A_c^P, then the assignment procedure will be $A_c^a = \{Z_{C1}^M, Z_{C2}^M \ldots\ldots Z_{Cn}^M, Z_{C1}^R, Z_{C2}^R, \ldots\ldots Z_{Cn}^R,\}$. Here the Mappers and Reducers are assigned for the set of tasks to be executed. For eg Z_{C1}^M *and* Z_{C1}^R are the number of Mapper and Reducer that are assigned to the i^{th} task of A_c. We have a set of existing VMs 'A' and a set of tasks 'n' that has been assigned with enough number of Mappers and Reducers. Now our objective is to schedule new tasks 'n1' that is remaining after reassigning few newer tasks to the already existing VMs 'A'. The minimum cost for the procurement of VMs from cloud is given by

$$min \sum_{j=1}^{m} A_j^P . (|A_j| + |\hat{A}_j|) \quad (1)$$

Number of Mappers required for the total tasks placed on the VMs is given by

$$\sum_{c=1}^{|A_j|+|\hat{A}_j|} Z_{Ci}^M = f_i^M, 1 \leq i \leq n \quad (2)$$

Number of Reducers required for the total number of tasks placed on the VMs is given by the equation

$$\sum_{c=1}^{|A_j|+|\hat{A}_j|} Z_{Ci}^R = f_i^R, 1 \leq i \leq n \quad (3)$$

CPU requirements needed by a Mapper and a Reducer in a VM is given by

$$\sum_{i=1}^{n} \left(Z_{ci}^M, M_i^{CPU} + Z_{ci}^R, R_i^{CPU}\right) \leq A_j^{CPU}, \forall_{A_j} \in A_j \quad (4)$$

Similarly Memory requirements is given by

$$\sum_{i=1}^{n} \left(Z_{ci}^M, M_i^{Mem} + Z_{ci}^R, R_i^{Mem}\right) \leq A_j^{Mem}, \forall_{A_j} \in A_j \quad (5)$$

Task has to be scheduled accordingly where the CPU and Memory required by the Mapreduce program for the new set of tasks plus the CPU and Memory used by the previous set of tasks shouldn't exceed the maximum available CPU and

memory of a VM. This constraint is mathematically given as, for CPU

$$\sum_{i=1}^{n}\left(Z_{ci}^{M}.M_i^{CPU}+Z_{ci}^{R}.R_i^{CPU}\right)+\sum_{i=1}^{n'}\left(Y_{ci}^{M}.M_i^{CPU}+Y_{ci}^{R}.R_i^{CPU}\right)\leq A_c^{CPU},\forall_{A_j}\in A_j \qquad (6)$$

For Memory,

$$\sum_{i=1}^{n}\left(Z_{ci}^{M}.M_i^{Mem}+Z_{ci}^{R}.R_i^{Mem}\right)+\sum_{i=1}^{n'}\left(Y_{ci}^{M}.M_i^{Mem}+Y_{ci}^{R}.R_i^{Mem}\right)\leq A_c^{Mem},\forall_{A_j}\in A_j \qquad (7)$$

PROPOSED WORK

The proposed (PHM) Procurement of Heterogeneous Machines algorithm is an optimization algorithm which consists of two procedures, (i) Canon Generation and (ii) Combinatorial Canons. The canon generation procedure generates small set of patterns which serves as the input for the second procedure. The generated patterns are the possible no of allocations of Mapper/Reducer in varying types of VMs. This can be expressed as an N-dimensional array such that

$$V_j^k=\left\{W_{jk}^1,W_{jk}^2\ldots.W_{jk}^i,\ldots,W_{jk}^N\right\}$$

Where V_j^k is the Virtual Machine that is used to load the Mapper/Reducer in the form of the generated patterns of the k[th] VM which is of j[th] type and 'N' is the total number of Mapper/Reducer used to process n' (new tasks) that arrives. W_{jk}^i is the number of laborers of the i[th] type and the laborer may be a Mapper or a Reducer. The pattern that is generated is feasible if it satisfies the equations (4) and (5) respectively.

The laborers (Mapper/Reducer) fall under the same category if their requirements (number of CPU's and Memory) are the same. Hence the dimensional problem can be formulated as, Number of dimensions 'N' (Types) is however less than '2n' (2 x number of newly arrived tasks). In order to simplify our problem, we recognize the Residual resources of the existing VMs and consider them as a new set of VMs since the residual resources are going to be reutilized for the new set of tasks n'. Consider C' being the new VM type and let N_j be the existing number of VMs same as the newly procured VM type, such that $C < j \leq C + C'$ where C is the already existing VM of the same type as that of the newly arrived VM.

C. Canon Generation

The feasible pattern generated for every type of VMs represents the arrangement pattern of the laborers (Mapper/Reducer). Initially the pattern set is generated for a particular type of VM and later patterns are generated for all types available. Iterations are repeated until all the laborers are placed on VMs. There are two algorithms involved in this procedure where the input to be given is the laborers L^j (total number of Mappers and reducers) and their resource requirements shouldn't exceed the maximum available resource of the j[th] type. And let l_i be one of the laborer (either a mapper or a reducer) in L^j. Next the output of the algorithm is the feasible set of canons generated for the j[th] type of VM and these are stored in the set TS_j. Best-fit strategy is implemented in the algorithm where the laborers L^j are allocated to the j[th] type VM. In Best-fit strategy the resource requirements of the laborers are mapped to the suitable type of VM and then allocated inorder to reduce the cost of procurement of new VMs. Another advantage of Best-fit strategy is additional time for placing the remaining workers is omitted since we plan and place the best workers in the beginning of the allocation process. Inorder to select the best laborer that exactly fits we assign a proxy weighs to the type of resources and the selection of laborers is given by $S_i = P^{CPU}.S_i^{CPU} + P^{Mem}.S_i^{Mem}$ where P^{CPU} and P^{Mem} are the proxy weights assigned to the resources and S_i^{CPU} and S_i^{Mem} are the requirements of the laborer l_i. Regarding the selection of proxy weights we adopt $P^{CPU}=\sum_{i=1}^{|L^j|}S_i^{CPU}$ and $P^{Mem}=\sum_{i=1}^{|L^j|}S_i^{Mem}$. After obtaining the set of patterns for the j[th] type, the procedure is repeated for all types of VMs accompanied by Best-fit strategy (BFS).

The second part of the procedure consists of algorithm which finds the patterns for all the type of VMs and iterations takes place. The inputs for algorithm 2 are the laborers L involved in the existing tasks execution process and the output will be the generated patterns for all types of VMs present. Initially algorithm 2 performs $C + C'$ loops, selecting the laborers L^j from L $(=\bigcup_{j=1}^{C}L^j)$ and then runs algorithm 1 to place the generated patterns for the j[th] type in the set TS_j. In case if few laborers are not assigned on the VM A_j, then algorithm 2 will try assigning them to other type of VMs and generates the new set of patterns. The iteration continues until all $C + C'$ loops are iterated and the laborers in L are assigned to VMs of all types $TS=\bigcup_{i=1}^{C}TS_j$.

Algorithm1: Canon Generation for the j[th] type of VM

Input: L^j;

Output: TS_j

Step 1: initialize the Canon generation for the j[th] type TS_j;

Step 2: implement BFS and add the patterns generated to TS_j;

Step 3: for k=1 to N

Step 4: end for

Step 5: return TS_j;

Algorithm 2: Canon generation for 'N' Types of VMs available

Input: L;

Output: TS_N;

Step 1: initialize the canon generation for every type of VM TS_j (1≤ j≤ c);

Step 2: for j=1 to $C + C'$ do

Step 3: select laborers L^j from L

Step 4: run algorithm 1 (L^j) and add the generated patterns to TS_j

Step 5: $L' \leftarrow L - L^j; j' = j;$

Step 6: while $L' \neq \emptyset$ do

Step 7: if $j' < c; j' = j' + 1;$ otherwise $j' = 1;$

Step 8: run algorithm 1 (L^j) and add the patterns generated into TS_j;

Step 9: end while

Step 10: end for

Step 11: set $TS_N = \bigcup_{i=1}^{c} TS_j;$

Step 12: remove the duplicated canon from TS_N

Step 13: return TS_N

D. Combinatorial Canons

Algorithm1 and Algorithm2 generates all the feasible canon for all types of VMs where $TS_N = \bigcup_{i=1}^{c} TS_j$. In our next procedure of combinatorial canon where most suitable combination of patterns are picked which has high impact in minimizing the total cost of VMs. The optimal combinations are formed so that the number of VMs rented from the cloud is minimal leading to cost effective Qos guaranteed solution.

EXPERIMENTAL SETUP

The real time experimental set up was created using Ubuntu Cloud Infrastructure with MAAS and Juju. Ubuntu 14.04 was used which consist of packages for building a private cloud server. MaaS makes it easier to prepare the hardware infrastructure for the deployment of dynamically scalable services and this feature is vital for the cloud as the need for on-demand resources may keep fluctuating. As per the need of the application new nodes can be added/ removed and re-deployed between the services. When the server is in the declared state it has to be brought into the MaaS for the commissioning process to begin and get ready for deployment. From users point of view sufficient numbers of nodes are available in the MaaS (Metal as a Service) and juju-jitsu the helper application and its deploy-to option allow to co-locate the services. Minimum of 10 machines are required for the deployment including the Bootstrap node and a MAAS server.

The table below shows the type of machines that were added as nodes and served as VMs to run the Mapreduce application. In our experimental set up, we have used 12 Acer Veriton X4630G personal computer each with the configuration of 64 bit Windows 7 operating system architecture, i5 processor, 4GB RAM and 500 GB of hard drive capacity, one Dell precision 690 workstation of specification Quad-core Intel Xeon 5300 series processor with upto 1333 MHz front side bus and 2x 4GB shared and two laptops one with 4GB RAM and another with 2GB RAM 64 bit Windows 7 Operating system with i3 processor. Total of 14 machines were added and workstation which is the bootstrap server which maintains the Openstack dashboard for managing the resources.

TABLE 1. System Configurations

Name	Processor	No of machines	Installed Memory (RAM)	System Type	Operating System
Type 1	Quad-core Intel Xeon 5300 series processor	1	2x 4GB	Workstation	Ubuntu 14.04
Type 2	Intel® Core™ i5 CPU 650 @ 3.20 GHz	12	4.00 GB (1.87 GB usable)	Desktop PCs	64 bit Windows 7 Professional
Type 3	Intel® Core™ i3 CPU M330 @ 2.13 GHz	1	2.00 GB (1.86 GB usable)	Laptop	64 bit Windows 7 Ultimate
Type 4	Intel® Core™ i3-4005U CPU @ 1.70 GHz	1	4.00 GB	Laptop	64 bit Windows 7 Ultimate

MapReduce 2.0 (MRv2) was used for performing parallel operations on available VMs deployed on Maas. For our transcoding process using Mapreduce We set the streaming rate to be 500kbps using http as the streaming protocol. Helix server was used to stream videos and the video chunks were divided into 20Kb were each chunk corresponds to 0.03 seconds of streaming rate. The media files were transcoded into MPEG4 file format which is suitable for all smart devices.

RESULTS AND DISCUSSION

The performance of our proposed work was tested for transcoding of live streams using Mapreduce operations. In our experiment 14 VMs were adopted from our Openstack private cloud set up. And these machines were classified under four Types based on their CPU and Memory. Figure 1 shows the result of our experiment based on the time taken for computation. The chart clearly proves that our proposed PHM performs better which shows the computation time 9 secs when 12 heterogeneous VMs were considered to perform the Mapreduce operations. The proposed algorithm was compared with baseline algorithms such as First-fit strategy, Random-Fit strategy and Homogeneous strategy. Our proposed PHM algorithm was enhanced with Best-Fit strategy (BFS) which proved around 39% better result when compared with

Homogeneous algorithm, 17% better than RFS and 14% better than FFS algorithms.

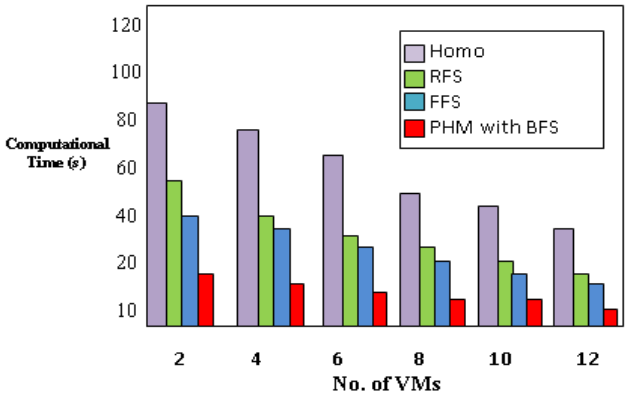

Fig. 1. Transcoding time (secs)

Figure 2 explains, when number of heterogeneous VMs are increases more type of laborers (Mappers/Reducers) can be assigned to them which reduce the computational cost as well increase the resource utilization of VMs. Whereas in the case of homogeneous machines, even if the number of machines increase, the type of laborers remains constant which means equal number of Mappers and Reducers are assigned to every VM which badly impacts on the cost of procurement and computational time. Figure 3 compares the computational times needed for live streaming and video on-demand.

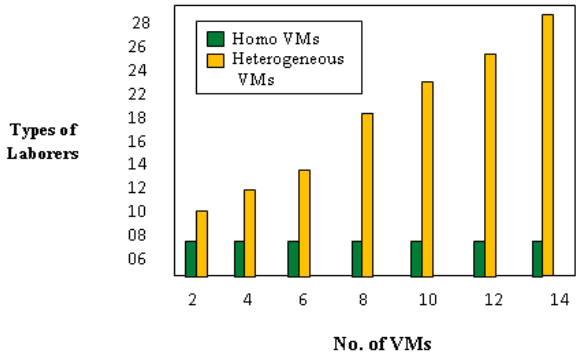

Fig. 2. Assignment of Mappers/Reducers

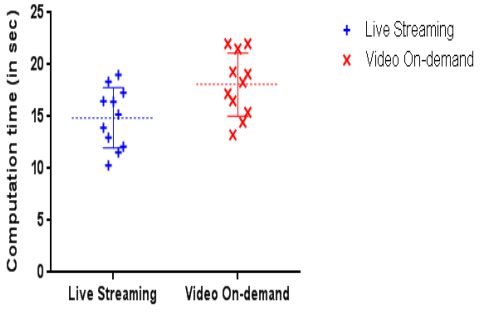

Fig. 3. Computational times for Live streaming and Video on-demand

Figure 4 compares the computational time of both live streaming and video on-demand as we vary the number of worker types correspondingly. We see a reduction in the computational times as the number of worker types increases. Figure 5 compares the computational time for the same live streaming and video on-demand but varying the number of worker nodes within a fixed number of worker types. It also has similar results as the previous case.

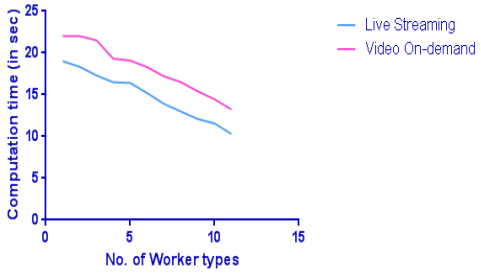

Fig. 4. Computational time as no. of worker types varies

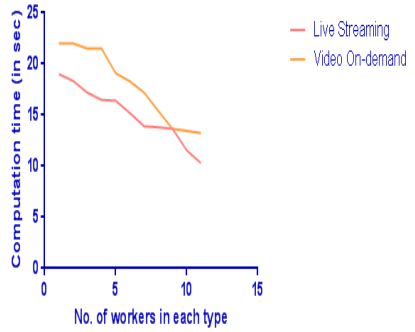

Fig. 5. Computational time as no. of workers in each type varies

Figure 6 compares the various approaches - RFS, FFS and the proposed approach.

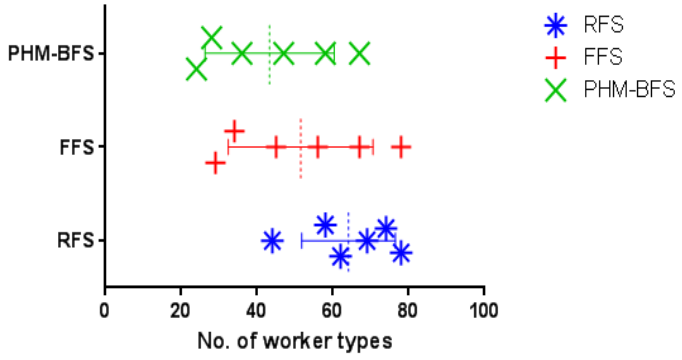

Fig. 6. Comparison of approaches with variation in no. of worker types

VII. CONCLUSION

The paper has proposed a Novel algorithm (PoHM) Procurement of Heterogeneous Machines to guarantee QoS which is minimizing the computational time of the given job.

Cloud based MapReduce model is used to transcode live streams in parallel and results were obtained. The problems handled in this paper is when and how many heterogeneous VMs to be procured from cloud on demand and another problem addressed is the placement problem where the laborers (Mapper/Reducer) have to be assigned to the available virtual machines in such a way they utilize even the residual resources available to meet the QoS cost effectively. Literature survey was carried out on various problems handled using Mapreduce model in both cloud and non cloud environments. Real time Cloud setup was built using Openstack from where VMs were procured to run the Mapreduce program. Proposed work was compared with three baseline algorithm which proves our proposed PoHM algorithm was enhanced with Best-Fit strategy (BFS) which proved around 39% better result when compared with Homogeneous algorithm, 17% better than RFS and 14% better than FFS algorithms.

REFERENCES

[1]. J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguad, *"Resource-aware adaptive scheduling for MapReduce clusters,"* in Middleware 2011, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 7049, pp. 187–207.

[2]. F. Tian and K. Chen, *"Towards optimal resource provisioning for running MapReduce programs in public clouds,"* In Proc. IEEE 4th Int. Conf. Cloud Computing, 2011, pp. 155–162.

[3]. H. Herodotou, F. Dong, and S. Babu, *"No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics,"* In Proc. ACM 2nd Symposium on Cloud Computing, 2011, p. 18.

[4]. M. AbdelBaky, H. Kim, I. Rodero, and M. Parashar, *"Accelerating MapReduce analytics using CometCloud,"* in Proc. IEEE 5th Int. Conf. Cloud Computing (CLOUD), 2012, pp. 447–454.

[5]. E. Hwang and K. H. Kim, *"Minimizing cost of virtual machines for deadline-constrained MapReduce applications in the cloud,"* in Proc. ACM/IEEE 13th Int'l Conf. Grid Computing (GRID), 2012, pp. 130–138.

[6]. P. Lama and X. Zhou, *"Aroma: Automated resource allocation and configuration of MapReduce environment in the cloud,"* in Proc. ACM 9th Int. Conf. Autonomic computing, 2012, pp. 63–72.

[7]. M. Cardosa, A. Singh, H. Pucha, and A. Chandra, *"Exploiting spatio-temporal tradeoffs for energy-aware mapreduce in the cloud,"* IEEE Transactions on Computers, vol. 61, no. 12, pp. 1737– 1751, Dec 2012.

[8]. K. Chen, J. Powers, S. Guo, and F. Tian, *"Cresp: Towards optimal resource provisioning for mapreduce computing in public clouds,"*, IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 6, pp. 1403–1412, June 2014.

[9]. B. Palanisamy, A. Singh, and L. Liu, *"Cost-effective resource provisioning for mapreduce in a cloud,"*, IEEE Transactions on Parallel and Distributed Systems, vol. PP, no. 99, pp. 1–1, 2014.

[10]. X. Xu and M. Tang, *"A more efficient and effective heuristic algorithm for the MapReduce placement problem in cloud computing,"*, In Proc. IEEE 7th Int. Conf. Cloud Computing, 2014, to be published.

[11]. W. Zhao, H. Ma, and Q. He, *"Parallel k-means clustering based on mapreduce,"* in Cloud Computing, ser. Lecture Notes in Computer Science, M. Jaatun, G. Zhao, and C. Rong, Eds. Springer Berlin Heidelberg, 2009, vol. 5931, pp. 674–679.

[12]. A. Matsunaga, M. Tsugawa, and J. Fortes, *"Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications,"* in Proc. IEEE 4th Int. Conf. eScience, 2008, pp. 222–229.

[13]. Q. He, T. Shang, F. Zhuang, and Z. Shi, *"Parallel extreme learning machine for regression based on mapreduce,"* Neurocomputing, vol. 102, pp. 52 – 58, 2013.

[14]. M. Husain, L. Khan, M. Kantarcioglu, and B. Thuraisingham, *"Data intensive query processing for large rdf graphs using cloud computing tools"*, In Proc. IEEE 3rd Int. Conf. Cloud Computing, July 2010, pp. 1–10.

[15]. J. Wolf, D. Rajan, K. Hildrum, R. Khandekar, V. Kumar, S. Parekh, K.-L. Wu, and A. Balmin, *"FLEX: A slot allocation scheduling optimizer for MapReduce workloads,"* in Middleware 2010, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg,2010, vol. 6452, pp. 1–20.

[16]. M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, *"Improving mapreduce performance in heterogeneous environments."* in *OSDI*, vol. 8, no. 4, 2008, p. 7.

[17]. S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi, *"Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud,"* in Proc. IEEE 2nd Int. Conf Cloud Computing Technology and Science (CloudCom), Nov 2010, pp. 17–24.

[18]. H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, *"Moon: Mapreduce on opportunistic environments,"* In Proc. ACM 19th Int. Symposium on High Performance Distributed Computing, 2010, pp. 95–106.

[19]. H. Herodotou and S. Babu, *"Profiling, what-if analysis, and costbased optimization of mapreduce programs,"* In Proc. Int. Conf. VLDB Endowment, vol. 4, no. 11, pp. 1111–1122, 2011.

[20]. J. Polo, D. Carrera, Y. Becerra, and M. Steinder, *"Performancedriven task co-scheduling for MapReduce environments,"* In Proc. IEEE Symp. Network Operations and Management Symposium (NOMS), 2010, pp. 373–380.

[21]. C. Bellettini, M. Camilli, L. Capra, and M. Monga, "*Distributed ctl model checking in the cloud*," arXiv preprint arXiv:1310.6670, 2013.

[22]. S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, *"The hibench benchmark suite: Characterization of the MapReduce-based data analysis,"* in Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on. IEEE, 2010, pp. 41–51.

[23]. CPLEX, "IBM CPLEX Optimizer." [Online]. Available: http://ibm.com/software/commerce/optimization/cplexoptimizer/

[24]. Ganglia, "Ganglia monitoring system." [Online]. Available: http://ganglia.sourceforge.net/

[25]. Hadoop, "Hadoop releases." [Online]. Available: http://hadoop.apache.org/releases.html

[26]. VMware, "VMware homepage." [Online]. Available: http://www.vmware.com

[27]. J. Kang and S. Park, *"Algorithms for the variable sized bin packing problem,"* European Journal of Operational Research, vol. 147, no. 2, pp. 365–372, 2003.

[28]. *M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms,"* Journal of Parallel and Distributed Computing, vol. 70, no. 9, pp. 962–974, 2010.

[29]. B. S. Baker, *"A new proof for the first-fit decreasing bin-packing algorithm,"* Journal of Algorithms, vol. 6, no. 1, pp. 49–70, 1985.

[30]. H. Dyckhoff, *"A typology of cutting and packing problems,"* European Journal of Operational Research, vol. 44, no. 2, pp. 145 –159, 1990.

[31]. K. Wang, B. Tan, J. Shi, and B. Yang, "*Automatic task slots assignment in hadoop MapReduce,"* In Proc. 1st Workshop Architectures and Systems for Big Data, ser. ASBD '11, New York, NY, USA, 2011, pp. 24–29.

[32]. J. Polo, Y. Becerra, D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade, *"Deadline-based MapReduce workload management,"* IEEE Trans. Network and Service Management, vol. 10, no. 2, pp. 231–244, June 2013.

[33]. A. Verma, P. Ahuja, and A. Neogi, *"pmapper: power and migration cost aware application placement in virtualized systems,"* in Middleware 2008. Springer, 2008, pp. 243–264.

[34]. [34] S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubramanian, K. Talwar, L. Uyeda, and U. Wieder, "*Validating heuristics for*

virtual machines consolidation," Microsoft Research, MSR-TR-2011-9, 2011.

[35]. M. Monaci and P. Toth, *"A set-covering-based heuristic approach for bin-packing problems,"* INFORMS Journal on Computing, vol. 18, no. 1, pp. 71–85, 2006.

[36]. M. Haouari and M. Serairi, *"Heuristics for the variable sized bin-packing problem,"* Computers and Operations Research, vol. 36, no. 10, pp. 2877–2884, 2009.

[37]. S. Srikantaiah, A. Kansal, and F. Zhao, *"Energy aware consolidation for cloud computing,"* In Proc. 2008 Conf. Power aware computing and systems, vol. 10. San Diego, California, 2008.

[38]. B. Palanisamy, A. Singh, L. Liu, and B. Jain, *"Purlieus: Locality-aware resource allocation for mapreduce in a cloud,"* In Proc. 2011 *Int. Conf. High Performance Computing, Networking*, Storage and Analysis, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 58:1–58:11. [Online]. Available: http://doi.acm.org/10.1145/2063384.2063462.

[39]. A. Verma, L. Cherkasova, and R. H. Campbell, *"ARIA: Automatic resource inference and allocation for MapReduce environments".*

[40]. K. Kc and K. Anyanwu, *"Scheduling hadoop jobs to meet deadlines,"* In Proc. IEEE *2nd Int. Conf. Cloud Computing* Technology and Science (CloudCom), Nov 2010, pp. 388–392.

[41]. T. White, Hadoop: The Definitive Guide. *O'Reilly Media*, Inc., 2009.