AMREF: An Adaptive MapReduce Framework for Real Time Applications*

Fan Zhang[1], Junwei Cao[2,3,*], Xiaolong Song[1], Hong Cai[4] and Cheng Wu[1,3]
[1]*National CIMS Engineering and Research Center, Department of Automation*
Tsinghua University, Beijing 100084, P. R. China
[2]*Research Institute of Information Technology, Tsinghua University, Beijing 100084, P. R. China*
[3]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, P. R. China*
[4]*IBM China Development Laboratory*
[*]*Corresponding email: jcao@tsinghua.edu.cn*

Abstract

This paper presents AMREF, an Adaptive MapREduce Framework designed for an effective use of computational resources in data center networks to deal with real time data intensive applications. AMREF entails its adaptivity from adaptive splitter, adaptive mappers and adaptive reducers in a stochastic control manner. We use three methods, feedback control, stochastic learning with smooth filter and kalman filter to implement the framwork. Comparison among the three methods suggests they can be effectively and efficiently used to reduce the makspan in three different real-world workload scenarios.

Key words: *Adaptive mapreduce; Feedback control; Stochastic learning control; Parallel Processing*

1. Introduction

1.1 Research Background

Cloud computing, from the inception of its concept, has wined much attention in industrial [3] and academic institutions [13]. Up to millions of interconnected servers, or called data center network [8], should be fully leveraged in order to provide on line applications, collaborative gaming, key word searching and commercial transactions.

Infrastructure services, e.g., Mapreduce, Google File System [6], and BigTable [1] are recently proposed and proven useful in large scale parallel applications and can be scaled to thousands of hosts for consistent and fault tolerant services [7]. As the increase of interconnected servers, how to organize a proper organization for parallel applications, say, the architectural structure of mapreduce applications, is very important. A good organization of the splitter nodes (splitters), map nodes (mappers) and reducer nodes (reducers), can not only shorten the execution time of data intensive applications, but also reduce running cost, say, the power consumption of the data center network.

AMREF, an Adaptive MapReduce Framework, is proposed in this paper to show our adaptive strategy that is used to effectively organize the splitter nodes (splitters), map nodes (mappers), reduce nodes(reducers).

*Manuscript submitted to IEEE GCC 2010, on June 30, 2010. All rights Reserved. Part of this work is performed when Fan Zhang is an intern student is IBM CDL sponsored by IBM Ph.d. Fellowship 2010-1011.

1.2 Motivation

Though mapreduce is widely used and implemented in many open source applications, the number of workers that implement the map and reduce functionalities are statically written in configuration files. This is not so flexible since different applications, or even different stages of one application, require different number of works to perform as different roles. Resource over/under provisioning is a serious problem nowadays in large scale internet applications. A proper organization of splitters, mappers and reducers are fundamentally important in this application.

We discuss three scenarios later on to show our solutions to adaptively utilize all these resources. Further, we bring in three typical control methods, feedback control, stochastic learning with smooth filter and kalman filter to implement our framework.

2. Related Work

MapReduce is a simple programming model for developing distributed data intensive application in cloud computing. Since it was proposed by Google for cluster of commodity machines, there have been many following projects. For instance, Hadoop [15] is a Mapreduce framework developed by Apache, and Phonix is another implementation designed for shared memory architecture by Stanford University.

Many researchers have focused on the MapReduce framework and the application of it . For instance, Genetic Algorithms (GAs) naturally fit into an iterative style. Thus, parallelizing genetic algorithms have received many attentions[12].

To implement PGAs, many models have been proposed like MRPGA[9]. It is an extension to the MapReduce model featuring a hierarchical reduction phase. And it is designed on a .NET-based enterprise Grid system using the mapreduce framework as the name shows. Another implement using Hadoop, Virtual Workspaces to perform the Bioinformatic Applications via WAN is named CloudBLAST[10]. These attempts in Bioinfomatic extend the parallel computing into the mapreduce framework to get a better scalability and efficiency.

Semantic inferencing and querying across large-scale RDF triple stores is notoriously slow and a MapReduce-based RDF molecular has been developed by several scientist in The University of Queensland[11]. And in their research, the evaluate the benefits of MapReduce framework in the application that requires integration and querying across large-scale protein-protein interaction database.

The growth of data in Geographical Information Systems (GIS), has far outpaced the growth of the power of a single processor. To deal with this condition, a high performance workflow system MRGIS[2] is proposed to execute GIS applications efficiently.

Another research in High Energy Physics data analyses and Kmesns clustering, with a CGL-MapReduce Model[5] to perform it even more efficently in Hadoop environment.

3. Adaptive mapreduce framework

We first introduce the principles of carrying out mapreduce applications, then we elaborate our application scenario which needs adaptive resource provisioning.

3.1 Preliminaries of mapreduce

In a mapreduce framework, user firstly specify a map function, which is used to process a (key, value) pair to generate a set of intermediate (key, value) pairs. After that, A typical mapreduce framework is shown in figure 1.

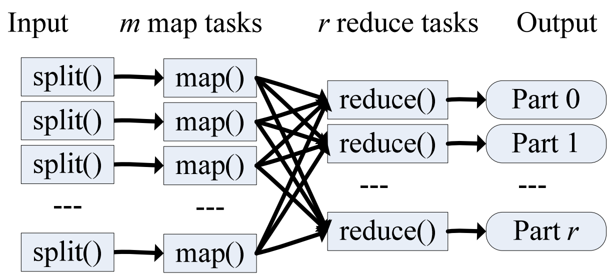


Figure 1. A mapreduce framework which splits the input file into *m* segments, and each segment corresponds to one map function. There are *r* reduce functions to generate *r* separate outputs.

While the open-source project called Hadoop developed by Apache [14] using the same architecture but implemented in Java, which is used in our research. Programs written in this functional style are automatically parallelized and executed on a cluster of computers. In the MapReduce jobs in Hadoop, the masternode which is used to split a job is called jobtracker, while the data nodes to execute the job is called tasktrackers. A jobtracker spilts the data into pieces for tasktrackers to map, and the tasktracker stores the intermediate results of map functions in local disks. When the job is done, a tasktracker will continue to map another data section, until the final results are combined after all the map and reduce processes. To support the data storing, Hadoop has a distributed file system called Hadoop Distributed File System (HDFS). Like the Google File System(GFS), HDFS also replicates the data on datanodes so that the system has a good fault-tolerance in storing and computing. Hadoop schedules the MapReduce computation jobs depending on the data locality and hence it improves the overall I/O throughput. This setup is well suited for an environment where Hadoop is installed in a large cluster of commodity machines.

3.2 Problem formulation

Generally, it is user's duty to specify the number of splitters, mappers and reducers for data intensive applications. It is normally very difficult to optimally predefine the number in order to maximize the operation performance, e.g. makespan and cost, to run a program. On one hand, we should make full use of the nodes; on the other hand, we should balance the load to minimize the meaningless nodes' waiting for an incoming event. We use a real time data intensive application in physics to show how our problem is formulated.

Gravitational Waves (GW) are produced by the movement of energy in mass of dense material which fluctuate space-time structure. LIGO [4] (Laser Interferometer Gravitational wave Observatory) embodies three most sensitive GW detectors in the world which are L1, H1, H2 (two in Hanford and one in Louisiana) jointly built by Caltech, MIT, etc. to detect GW. The detection is very useful for us to explore the mystery of space. Triggers are produced by standard event trigger generators (e.g. Q-pipeline or Omega-pipeline which is used to search for significant transient events and write each event's info as a record (triggers) into plain text file (trigger files)), which are used as an important evidence to show when and where can GW possibly exist. The trigger files currently have the following five column definitions:
(1) central time [GPS Time]
(2) central frequency [Hz]
(3) duration [s]
(4) bandwidth [Hz]
(5) normalized energy [zero dimension]

There are several physical properties that we need to clarify. Suppose T is a given timespan,
(1)The number of triggers are generated during T(TNT);
(2)The second which has the maximum number of triggers during T(MaNT);
(3) The second which has the minimal number of triggers during T(MiNT);
(4) The second which has the maximum normalized energy during T(MaNE);
(5) The second which has the minimal normalized energy during T(MiNE);

The triggers are generated in real time, which causes a big problem that how to properly allocate resources (number of splitters, mappers and reducers) dynamically and automatically in mapreduce programming. This is our major motivations in using adaptive framework to carry out this work.

4. Adaptive mapreduce framework

4.1 Adaptive splitters

Splitter is used to split files awaiting for process into smaller blocks and serve each block to its corresponding mapper node. In the Hadoop application each splitter divides file evenly, which lacks of a proper scheduling mechanism to serve different amount of data based on the different processing capacity of mapper functions in each node. In cloud environment, mappers with variant processing capacity are in data centers. Also, there are many other reasons for this, such as network latency, availability, etc., which causes an unbalance workload in mappers. Taking the above details into account is the major concern of adaptive splitters.

In other words, splitter should have a global view to see which mapper is faster, thus the files for that mapper is relatively more than other mappers. This is the basic idea of load balancing in order to reduce the makespan of our whole application.

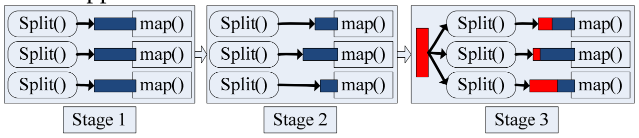


Figure 2. Adaptive splitter. In stage 1, splitter distribute the input file evenly to the three mappers. In stage 2, different mapper with different processing capacity have different length of input files. In stage 3, a new input file is distributed to the three mappers according to their processing capacity.

Splitter adaptively partitions the input file into several segments of different length and serve them to the corresponding mapper. Based on the run time application, how to do this segmentation, especially how to balance the workload into different mappers will be introduced followed in adaptive mappers.

4.2 Adaptive mappers

Mappers are used to convert each (key, value) pairs into intermediate (key, value) pair. The number of mappers that are in a specific Hadoop application scenario is predefined by setting the function setNumMapTasks(int) before the application is started. This number is closely influenced by the total size of the inputs, namely the total number of blocks of the input files after the split step.

In real time scenario, the number of mappers initially set may be not/over enough for the dynamism of the application. Adaptive mapper lies in the automatically increase or decrease the mappers based on the run time application.

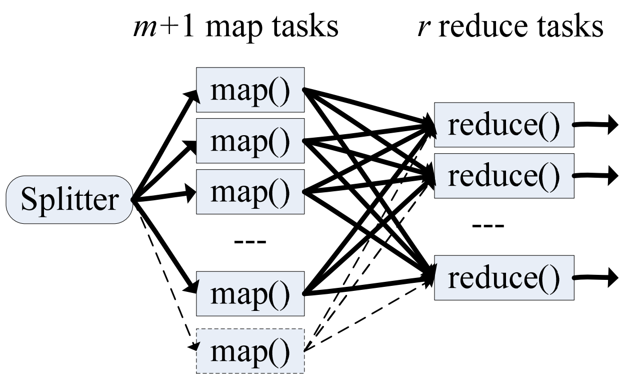


Figure 3. An adaptive map task is adaptively added

Figure 3 demonstrates that an adaptive mapper is increased because of either overburden of the other mappers, or an unbalanced workload between mappers and reducers. The newly created mapper should have a one-to-all connection with the reducers to perform consistent functionality. Similar rule can be applied to adaptively decrease a mapper when the system is idle for a relatively long time. How to define the "relatively long time" will be shown in our followed sections.

4.3 Adaptive reducers

Reducers are used to reduce a set of intermediate values which share a key to a smaller set of values. For runtime multiple attribute applications, each reducer outputs its results related to one attribute. For example, in our LIGO application, each reducer outputs one of the five basic properties in real time. However, there are sometimes when the output of mappers of a specific attribute (one property result) are too fast that the number of reducers are not enough. Fig. 4. demonstrates an adaptive reducer which is parallelly and adaptively added and performs its r+1 output. If it outputs the same attribute with one or many reducers, a further *merge* stage is needed to combine the outputs together.

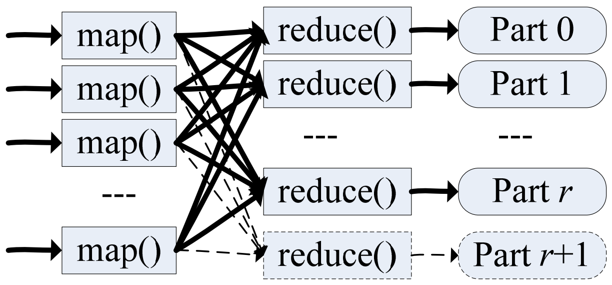


Figure 4. A parallel reduce task is adaptively added

There is another situation that a new reducer should be adaptively and sequentially added whose input is from the output of some or all the existing reducers. This happens when a combined result is needed. For example, if we want to output the statistical seconds between the time span that has the maximum number of triggers and minimum, we should combine the output of the property (1) and (2) and calculate the time span. In this way, a sequential reducer is adaptively added whose input is the output of the first two reducers. Figure 5 demonstrate this scenario with details.

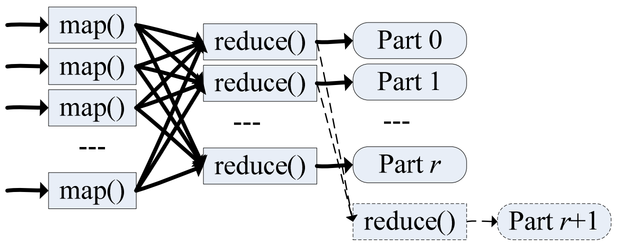


Figure 5. A sequential reduce task is adaptively added. The inputs of this reduce task are generated from the first and second reduce task

Until now, we have introduced the adaptive splitters, mappers and reducers individually in different scenarios. A common problem left open is the way we quantify the adaptivity. We move on in our controlling principles in 4.4 and 4.5 to answer the question.

4.4 Feedback based control

Feedback control is a basic method in control theory. It uses a monitor to get the current status of the system, and usually carry a negative signal to the input according to the variation of the monitor. This is called negative feedback, which gives the system stability and randomness.

But in practice of our adaptive mapreduce, we use a positive feedback control which is a heuristic scheduling mechanism to balance the workload of all the servers. In this positive feedback loop, we measured the utilization of each queue in all the three process: splitting, mapping and reducing. If the utilization of the 95% servers or above in splitting stage surpass 90%, we'll add one server more to release the workload in this stage. Similarly, If the utilization of the 95% servers or above in splitting stage lower than 20%, we'll reduce one server. All these rules are applied to map and reduce stage.

In theory, the feedback control will be a perfect method when the scheduling is sensitive enough and all the cost in increasing or decreasing the mapreducers is not considered. But in simulation and future practice, we will certainly set the stability margin in a proper value to get a excellent result within our view.

4.5 Stochastic learning based control

Stochastic control is a new dynamic control method, it's based on the former behavior of the status in system. In our adaptive model, it's a learning mode for balancing the workload, and the performance is always impressive when handling with stochastic arrivals and disturbance.

Stochastic control rely on the statistical data. With the data we have collected, the learning mode gives prediction of the incoming data, including the incoming time, the amounts, and traffic spikes, then the network will adjust itself to moderate the mutation of incoming data.

For example, in a real-time adaptive mapreduce application, the mutation of incoming data is smooth in most time of a day with only some random noise and disturbances, but in a certain time, just like the rush hour, the data will flood in the system with a tremendous speed.

But a system with stochastic control will perform perfectly with precise prediction collected. It gets a balance of cost and capability in the smooth period, and add mapreducer in advance to moderate the flood in rush hour, in this way, stochastic control will perform well in a real-time data coming system with regularity.

5. Experimental Studies

We first discuss our settings in doing the simulation, then we compare three methods, feedback control, stochastic learning with smooth filter and stochastic learning with kalman filter to justify our work.

5.1 Experimental Settings

We conduct our simulation using SimEvents, which is a software toolkits in Matlab. All the splitters, mappers and reducers are simulated as a queuing network with each node as a queue. We assume that each queue is running under different service capacity, thus the queuing length is varying during the whole process, which entails our needs for adaptive splitters, mappers and reducers.

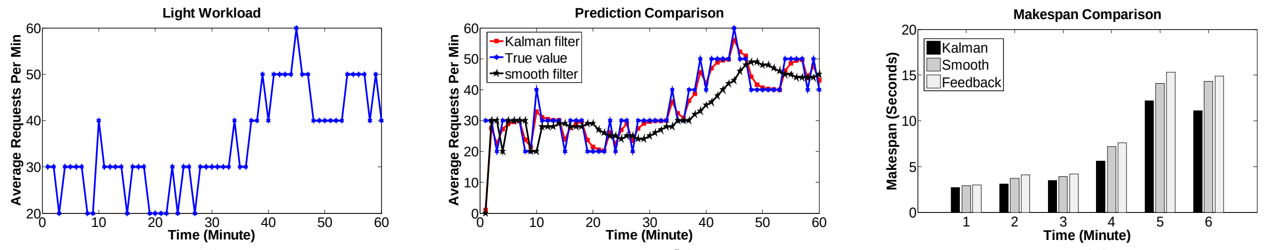
We compare feedback based control with stochastic learning based method. In stochastic learning based control method, we use kalman filter and smooth filter to predict the workload based on the previous stages. We compare the above three methods in our experiment to derive their comparative advantages and disadvantages.

Our workloads are generated based on a Web trace from the 1998 Soccer World Cup site [26]. This trace is an average arrival during each minute over sixty-minute duration as shown in Fig. 6(a), Fig. 7(a) and Fig. 8(a). We select three typical workloads: small, moderate and heavy, to do the simulation. They are different from each other in that the average arrival rate per minute.

We analyze the makespan, which is defined as the execution time of a batch of jobs. This value should be minimized in order to improve the throughput.

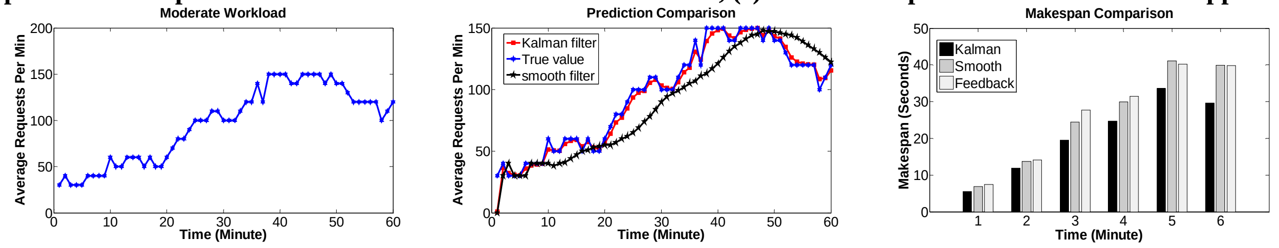
The experimental results are shown from Fig. 6 to Fig. 8.

5.2 Experimental Results



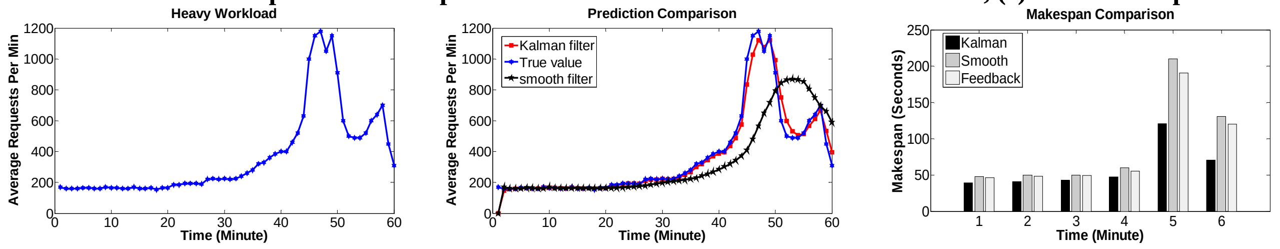
(a) (b) (c)

Figure 6. Comparison under light workload. (a) is the characteristics of the light workload, (b) is the workload and prediction comparison between smooth filter and kalman filter, (c) is the makespan to execute the above application



(a) (b) (c)

Figure 7. Comparison under moderate workload. (a) is the characteristics of the moderate workload, (b) is the workload and prediction comparison between smooth filter and kalman filter, (c) is the makespan.



(a) (b) (c)

Figure 8. Comparison under heavy workload. (a) is the characteristics of the heavy workload, (b) is the workload and prediction comparison between smooth filter and kalman filter, (c) is the makespan .

Based on the above experimental results, several conclusions could be reached.

(1)Smooth filter prediction is not a good enough choice compared with kalman filter prediction. Though there are lots of prediction algorithm choices, we use kalman as our representative example.

(2)Prediction accuracy is closely related with the makespan of a batch of jobs. A good prediction of workload is useful in arranging proper resources, which is much better than static resource provisioning.

(3)Feedback control based method is not efficient at the times the workload is relatively low compared with smooth filter. That disadvantage could be changed into advantage when the workload is relatively high. That's because the prediction error of the smooth is too much, which affects the makespan of the work.

(4)With a high prediction accuracy, the stochastic learning based method is advantageous than other methods whenever the workload is small or large.

(5)Adaptive mapreduce framework, or AMREF, shows that it is widely applicable in different workload scenarios, which justify our motivation to carry out this work.

6. Conclusions and Future Work

We firstly conclude our major contributions in this work and then suggest two possible directions to extend this work.

6.1 Research conclusions

In this paper, we have extended the mapreduce algorithm from a static scheme to a dynamic one using a series of adaptivities, adaptive splitter, adaptive mapper, and adaptive reducers. Our original technical contributions are summarized below :

(1)**Proposing the adaptive scheme**. This adaptivity lie in the three key stages of mapreduce algorithm, which not only can be used to improve the utilization of the servers, but also reduce the makespans in our work.

(2)**Comparing the workload prediction method and their influence on makespan**. We compare the three methods in view of the adaptivity to compare the relative advantages, which is useful in practice to decide which method to adopt in specific scenario.

6.2 Our Future Work

For further research, we suggest to extend the work in the following two directions:

(1) **Implement the AMREF algorithm in real mapreduce applications**. We have simulated and proved the advantage of using AMREF in our work, which is perfectly matched and properly used in large scale cloud applications. We'll carry on this work by implementing real adaptive scenarios for our algorithm in Hadoop and Hive applications.

(2) **Building useful tools to serve for larger virtualized cloud platform**. The matlab simulation to analyze optimal number of virtual resources in our experiment should be packaged into software toolkits in order to make it available for larger virtualized cloud platforms. Our experimental software can be tailed and prototyped toward this end.

Acknowledgement

This work is supported by National Science Foundation of China (grant No. 60803017) and Ministry of Science and Technology of China under National 973 Basic Research Program (grants No. 2011CB302505 and No. 2011CB302805) and National 863 High-tech R&D Program (grants No. 2008AA01Z118 and No. 2008BAH32B03).

References

[1] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes and R. E. Gruber, Bigtable: A Distributed Storage System for Structured Data, OSDI 2006: Seattle, WA, USA.

[2] Q. Chen, L. Wang, and Z. Shang, MRGIS: A MapReduce-Enabled High PerformanceWorkflow System for GIS, 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES). Indianapolis, USA. IEEE Press, 2008.

[3] J. Dean and S. Ghemawat, Mapreduce: Simplified data processing on large clusters, ACM Commun., vol. 51, Jan. 2008, pp. 107-113.

[4] E. Deelman, C. Kesselman, et al, "GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists", *Proc. 11th IEEE Int. Symp. on High Performance Distributed Computing*, pp. 225-234, 2002.

[5] J. Ekanayake, S. Pallickara and G. Fox, MapReduce for Data Intensive Scientific Analyses, DOI 10.1109/eScience.2008.5

[6] S. Ghemawat, H. Gobioff, and S. Leung, The Google File System, SOSP'03, October 19–22, 2003, Bolton Landing, New York, USA.

[7] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang and S. Lu, BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers, SIGCOMM'09, August 17–21, 2009, Barcelona, Spain.

[8] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhangand S. Lu, DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers, SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

[9] C. Jin, C. Vecchiola and R. Buyya, MRPGA: An Extension of MapReduce for Parallelizing Genetic Algorithms, Fourth IEEE International Conference on eScience, 2008, Indiana University, USA.

[10] A. Matsunaga, M. Tsugawa and J. Fortes, CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications, Fourth IEEE International Conference on eScience, 2008, Indiana University, USA.

[11] A. Newman, Y. Li and J. Hunter, Scalable Semantics – the Silver Lining of Cloud Computing, eScience, 2008. eScience '08. IEEE Fourth International Conference on (06 January 2009), pp. 111-118.

[12] M. Nowostawski and R. Poli, Parallel Genetic Algorithm Taxonomy, KES'99, MAY 13, 1999.

[13] L. Youseff, M. Butrico and D. D. Silva, Toward a Unified Ontology of Cloud Computing, Grid Computing Environments Workshop, GCE08, held in conjunction with SC08 , November, 2008.

[14] http://www.apache.org/

[15] http://hadoop.apache.org/