Divisible Load Theory A New Paradigm For Load Scheduling in Cloud Systems

Karishma Sethi[1], Neelam Joshi[2]
[1]Dept of Computer Science
[2]Assistant Professor, Dept of Computer Science
[1, 2] MPCT College, Gwalior, MP, India

Abstract- *Cloud computing can be defined as a paradigm for computation of a large number of systems connected in a private or public network to provide a dynamically scalable infrastructure application, data and storage. Cloud computing can be viewed as a distributed process which processes data transferred from personal computers and servers to the computers located at different clusters and use the high speed of the network.*

Cloud computing [1]stores the large amount of data from multiple users and distributes the resources in the open environment, as the amount of data stored increases in the open environment,the task of balancing the load arises, thus making load balancing a challenge in cloud computing.

Keywords- Cloud computing, load balancing, Divisible load scheduling .

I. INTRODUCTION

Load balancing is the solution to the problem of overloaded servers in a cloud environment [1]. This balancing is achieved by distributing larger processing load to smaller processing nodes for enhancing the performance of systems in the cloud.

It helps in allocation of computing resources fairly. It also helps in minimizing resource consumption by proper load balancing techniques. It also helps in implementing the failure of overloadednodes in the system. Different load balancing technique helps the system in network by providing maximum throughput with minimum response time.

II. DISTRIBUTED LOAD BALANCING FOR CLOUDS

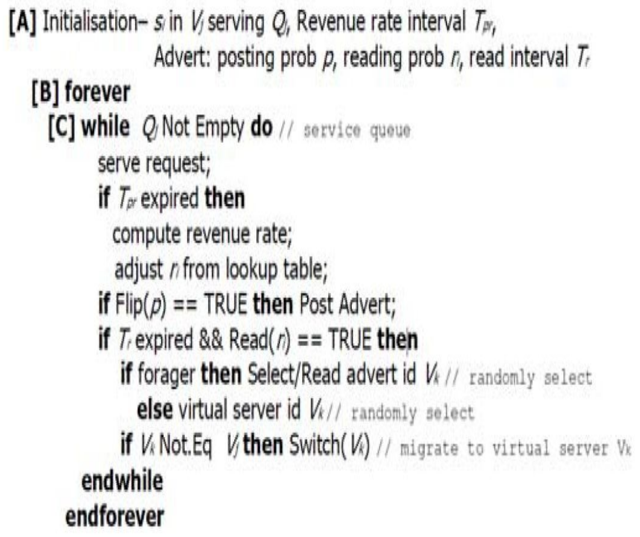
In complex and large systems, the need for load balancing increases tremendously, so as to achieve load balancing in such systems different load balancing techniques should act at the components of the clouds in such a way that the load of the whole cloud is balanced. For this purpose, the three solutions which can be applied to a distributed system[7] are :- Honeybee foraging algorithm, a biased random sampling and active clustering[2].

2.1 Honeybee Foraging Algorithm

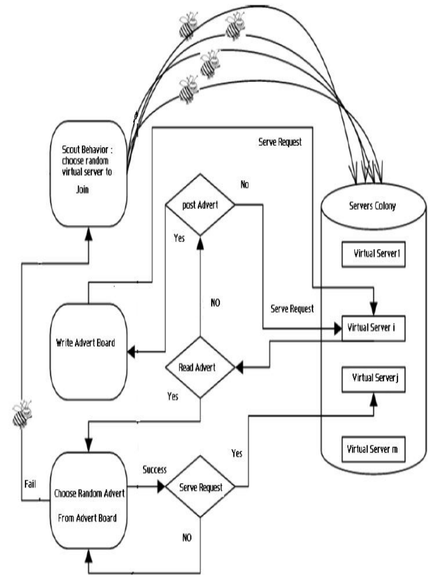
This is a behaviour-basedalgorithm, which is inspired by honeybees and their strategy to find food. Two classes of bees are defined in this algorithm: forager and scout. A forager bee searches for the appropriate source of food , upon finding one they return to their beehive and performs “waggle dance”. The source of food is selected based on the quality, quantity and distance of the source of food from the beehive. Waggle dance gives an idea of how much food is left and hence results in more exploitation of the food source.

In case of cloud computing, a cloud consists of different servers and these virtual servers and processes different request from the users. Each server acts as a forager or scout and processes a request from its queue, after the successful completion of a request each server calculates a profit or reward, which is similar to the quality that the bees show in their waggle dance and places the advert on the board.

Initially each server chooses virtual server randomly to serve a request, as the request is completed, load of current server is calculated and compared with the overall virtual servers load. If the load of a particular virtual server is low then the next request will be assigned to this virtual server but if the load of that virtual sever is high then the scout reads the advert board and will follow the listed virtual servers from the advert board. Generally the load is calculated in terms of CPU utilization.



```
[A] Initialisation– $s_i$ in $V_j$ serving $Q_j$, Revenue rate interval $T_{pr}$,
                    Advert: posting prob $p$, reading prob $r_i$, read interval $T_r$
  [B] forever
    [C] while $Q_j$ Not Empty do // service queue
          serve request;
          if $T_{pr}$ expired then
            compute revenue rate;
            adjust $r_i$ from lookup table;
          if Flip($p$) == TRUE then Post Advert;
          if $T_r$ expired && Read($r_i$) == TRUE then
            if forager then Select/Read advert id $V_k$ // randomly select
              else virtual server id $V_k$ // randomly select
            if $V_k$ Not.Eq $V_j$ then Switch($V_k$) // migrate to virtual server $V_k$
      endwhile
    endforever
```



2.2 Biased Random Sampling

Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each indegree directed to the free resources of the server.

Regarding job execution and completion,wheneveranodedoesorexecutesajob,itdeletesanincomingedge,whichindicates reduction in the availability of free resource. After completion of a job, the node creates an incoming edge, which indicates an increase in the availability of free resource. The addition and deletion of processes is done by the process of random sampling. The walk starts at any one node and at every step a neighbor is chosen randomly. The last node is selected for allocation for load. Alternatively, another method can be used for selection of a node for load allocation, that being selecting a node based on certain criteria like computing efficiency, etc. Yet another method can be selecting that node for load allocation, which is under loaded i.e. having highest in degree. If b is the walk length, then, as b increases, the efficiency of load allocation increases. We define a threshold value of b, which is generally equal to log n experimentally.

A node upon receiving a job, will execute it only if its current walk length is equal to or greater than the threshold value. Else, the walk length of the job under consideration is incremented and another neighbor node is selected randomly. When a node then in the graph executes a job, an incoming edge of that ode is deleted. After completion of the job, an edge is created from the node initiating the load allocation process to the node, which was executing the job.

Finally what we get is a directed graph. The load balancing scheme used here is fully decentralized, thus making it apt for large network systems like that in a cloud.

2.3 Active Clustering

Active Clustering works on the principle of grouping similar nodes together and working on these groups. The process involved is:

A node initiates the process and selects another node called the matchmaker node from its neighbors satisfying the criteria that it should be of a different type than the former one.

The so called matchmaker node then forms a connection between a neighbor of it which is of the same type as the initial node. The matchmaker node then detaches the connection between itself and the initial node. The above set of processes is followed iteratively.

III. PROPOSED WORK

The time required for completing a task with in one process is very high. So the task is divided into number of sub-tasks and each sub-task is given one job. Let the task S is divided into number of sub-tasks S1, S2, S3...Sn. Out of these some are executed sequentially and some are executed parallel. So the total time period for completing the task decreases and hence the performance increases. These sub-tasks can be represented in a graph structure known as state diagram. An example is given below.

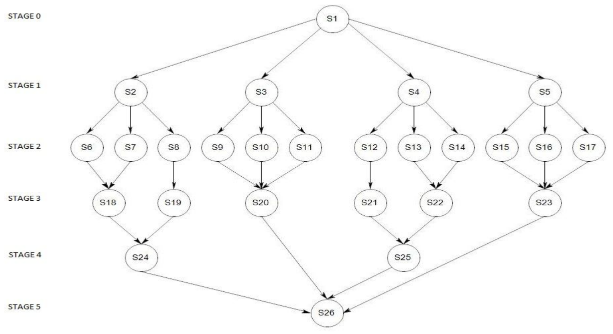


Figure : State Diagram

S1 is executed first. S2,S3,S4 and S5 can be executed in parallel during the same time slice. S18 requires the execution of S6 and S7 both, but S19 requires the execution of S8 and so on for all the sub tasks as shown in the state diagram. Our aim is to execute these tasks in different nodes of a distributed network so that the performance can be enhanced.

IV. DESCRIPTION

The distributed network may follow different topologies. The tasks are distributed over the whole network. One topological network connects with the other through a gate- way. One of the physical topologies forming a cloud is shown in the figure 1.

This distributed network is a cloud, because some of the nodes are Mobile clients, some of them are Thin and some are Thick clients. Some of them are treated as masters and some are treated as slaves. There are one or more datacenters distributed among the various nodes, which keeps track of various computational details. Our aim is to apply the Divisible Load Scheduling Theory(DLT) [5] for the clouds of different sizes and analyze different performance parameters for different algorithms under DLT and compare them.

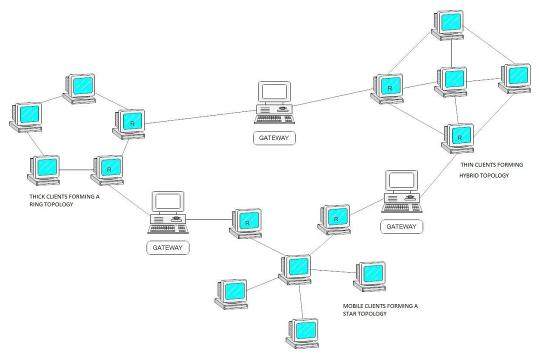

GATEWAY
GATEWAY
GATEWAY
R
THICK CLIENTS FORMING A RING TOPOLOGY
THIN CLIENTS FORMING HYBRID TOPOLOGY
MOBILE CLIENTS FORMING A STAR TOPOLOGY

Figure 1: A cloud showing different topologies

V. DIVISIBLE LOAD SCHEDULING THEORY IN CLOUDS

Introduction

Divisible load scheduling theory (DLT) [4]in case of clouds is an optimal division of loads among a number of master computers, slave computers and their communication links. Our objective is to obtain a minimal partition of the processing load of a cloud connected via different communication links such that the entire load can be distributed and processed in the shortest possible amount of time.

The whole Internet can be viewed as a cloud of many connection-less and connection- oriented services. The concept of load balancing in Wireless sensor networks can also be applied to clouds as WSN is analogous to a cloud having number of master computers (Servers) and number of slave computers(Clients).

The slave computers are assumed to have a certain measurement capacity. We assume that computation will be done by the master computers, once all the measured data is gathered from corresponding slave computers. Only the measurement and communication times of the slave computers are considered and the computation time of the slave computers is neglected. Here we consider both heterogeneous and homogeneous clouds. That is the cloud elements may possess different measurement capacities, and communication link speeds or the same measurement capacities, and communication link speeds. One slave computer may be connected to one or more master computers at a certain instant of time.

In DLT in case of clouds, an arbitrarily divisible load without having any previous relations is divided and first distributed among the various master computers (for simplicity here the load is divided equally between the master computers) and the each master computer distributes the load among the corresponding slave computers so that the entire load can be processed in shortest possible amount of time. An important reason for using DLT is its flexibility, tractability, data parallelism and computational difficulties [8].

System Model

The cloud that we have considered here is a single level tree (star) topology consisting of K number of master computers and each communicating N number of slave computers as shown in Fig 2.

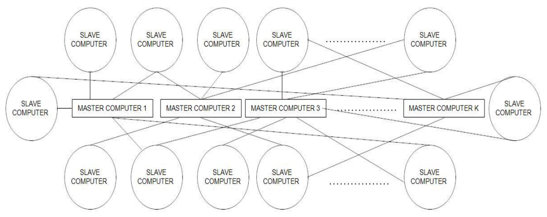


Figure 2 : K number of master computers each joining N number of slave computers in single level Tree network (STAR Topology)

It is assumed that the total load considered here is of the arbitrarily divisible kind that can be partitioned into fractions of loads to be assigned to all the master and slave computers in the cloud. In this case each master computer first assigns a load share to be measured to each of the corresponding *N* slave computers and then receives the measured data from each slave. Each slave then begins to measure its share of the load once the measurement instructions from the respective master have been completely received by each slave. We also assume that computation time is negligible compared with communication and measurement time.

5.1 Parameters, Definitions and Notation[7]

β*ki* The fraction of load that is assigned to a slave *i* by master *k.*

*aki*A constant that is inversely proportional to the measuring speed of slave *i* in the cloud.

b_{ki} A constant that is inversely proportional to the communication speed of link i in the cloud.

Tms Measurement intensity constant. This is the time it takes the ith slave to measure theentireloadwhen*aki* =1.Theentireassigned measurement loadcanbemeasuredonthe ith slave in time *akiTms.*

*Tcm*Communication intensity constant. This is the time it takes to transmit all of the measurement load over a link when *bki* = 1. The entire load can be transmitted over the ith link in time *bkiTcm*.

*Tki*The total time that elapses between the beginning of the scheduling process at t =0 and the time when slaver i completes its reporting to the master k, i =0,1,... ,*N*. This includes, in addition to measurement time, reporting time and idle time.

*Tfk*This is the time when the last slave of the master k finishes reporting (finish time or make-span).

Tfk = *max*(*Tk*1,*Tk*2,*Tk*3,...,*TkN*).

Tf This is the time when the last master receives the measurement from its last slave.

Tf = *max*(*Tf*1,*Tf*2,*Tf*3,...,*TfN*).

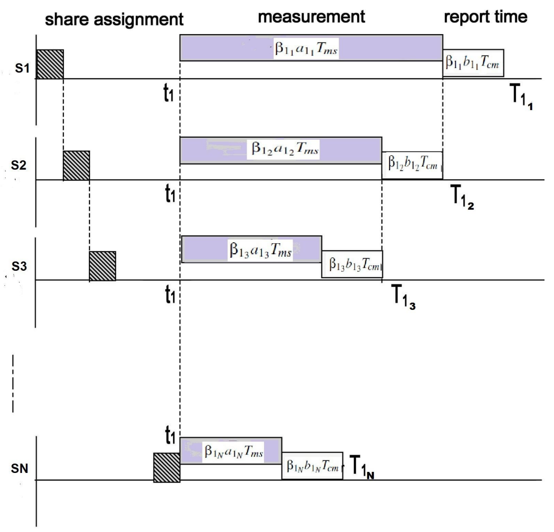
Some of the above used parameters and notations are taken from. These parameters were already used for finding closed form equations for load balancing for Wireless Sensor Networks.

5.2Measurement and Reporting Time

When Measurement starts Simultaneously and Reporting is done sequentially

Initially when time t = 0, all the slaves are idle and the master computers start to communicate with the first slave of the corresponding slaves in the cloud. By time t = t1, each slave will receive its instructions for measurement from the corresponding master as shown in fig 3. It is assumed that after measurements are made, only one slave will report back to the root master at a time (or we can say only a single link exists between them).

The slaves here receive a fraction of load from their corresponding master sequentially and the computation will start after each slave completely receives its load share.


share assignment
measurement
report time
S1
S2
S3
SN
t1
$\beta_{1_1}a_{1_1}T_{ms}$
$\beta_{1_1}b_{1_1}T_{cm}$
T_{1_1}
$\beta_{1_2}a_{1_2}T_{ms}$
$\beta_{1_2}b_{1_2}T_{cm}$
T_{1_2}
$\beta_{1_3}a_{1_3}T_{ms}$
$\beta_{1_3}b_{1_3}T_{cm}$
T_{1_3}
$\beta_{1_N}a_{1_N}T_{ms}$
$\beta_{1_N}b_{1_N}T_{cm}$
T_{1_N}
Figure 3: Timing diagram for single level tree network with a master computer and *N* slaves, which report sequentially.

Let us consider the first master computer and its corresponding group of slaves. From the definition of *Tki* , we can write

$T11 = t1+β11a11Tms+β11b11Tcm$ (1)
$T12 = t1+β12a12Tms+β12b12Tcm$ (2)
.
.
$T1N = t1+β1Na1NTms+β1Nb1NTcm$ (3)

The total measurement load originating at all the master computers is assumed to be normalized to a unit load. Thus each master computer will handle (1/K) load. So

$β11 +β12 +β13 +...+β1N−1 +β1N = 1/K$ (4)

Based on the timing diagram, we can write

$β11a11Tms=β12a12Tms+β12b12Tcm$
(5)

$β12a12Tms =$
$β13a13Tms+β13b13Tcm$ (6)
.
.
$β1N−2\ a1N−2\ Tms = β1N−1\ a1N−1\ Tms + β1N−1\ b1N−1\ Tcm$
(7)
$β1N−1\ a1N−1\ Tms = β1N\ a1N\ Tms + β1N\ b1N\ Tcm$
(8)

A general expression for the above set of equations is

$β1i = \quad s1iβ1i−1$ (9)

where

$s1i = a1i−1Tms/(a1iTms +b1iTcm)$

and i = 2,3,...,*N*.

The above recursive equation for

β1*i* can be rewritten in terms of β11 only as

$iβ1i =∏s1jβ11$ (10)

j=2
Now using the above sets of equations and the normalization equation, one can solve forβ11 as

$Ni\ \ β11 +∑∏s1jβ11 =1/K$ (11)

i=2 *j*=2

So β1*i* can be written as

$β1 = \underline{\qquad 1 \qquad .}$
$K(1+∑Ni=2∏ij=2s1j)$ (12)

Putting in eq-(10),

$β1i = \underline{\ ∏ij=2s1j \qquad .}$
$K1+∑Ni=2∏ij=2s1j$ *(13)*

where i =2,3,4,... ,*N*.

The minimum measuring and reporting time of the network will then be given as

$Tf = t1+\underline{(a11Tms+b11Tcm)}$
$K(1+∑Ni=2∏ij=2s1j)$ (14)

Similarly, we can obtain the generalized equation for master computer *r* as

$Tfr = t1+\underline{(ar1Tms+br1Tcm)}$
$K(1+∑Ni=2∏ij=2srj)$ (15)

In case of homogeneous networks (same measurement capacities and link speeds),
we can write

$s11 = s12 = s13 = ... = s1N−1 = s1$
$a11 = a12 = a13 = ... = a1N = a1$
$b11 = b12 = b13 = ... = b1N = b1$

So, eq-(5) becomes

$β1\ (1+s1 +s21 +...+sN−2 +sN−1) = 1/K$
(16)

Where$s1 = a1Tms/(a1Tms+b1Tcm)$. Simplifying the above equation,

$β1 = 1−s1 /K (1 − s N1)$ (17)

The master computer 1 will use the value of β11 to obtain the amount of data that has to be measured by the rest of the N-1 slaves corresponding to it by using the following equation:

$β1 =β1s1i−1$(18)

where i =2,3,4,... ,N.

The minimum measuring and reporting time of the homogeneous network will then be given as

$$Tf = t1 + \frac{(a1Tms + b1Tcm)(1-s1)1}{K(1-sN1)} \quad (19)$$

This measurement and reporting time of the network approaches $t1 + (b1Tcm)/K$ as N approaches infinity. So the reporting time supresses the measurement time when the no. of slaves to a corresponding master approaches infinity . Similarly we can obtain the above expression for rest of the master computers.

5.3When the Measurement starts Simultaneously and Reporting ends Simultaneously

Here each of N slave computers corresponding to a master computer in the cloud finish reporting at the same time. The cloud will have the same report finishing time for each slave corresponding to a master. That is each slave has a separate channel to its master as shown in the timing diagram of the network.

In this case the slaves receive their share of load from the master concurrently and start computation after completely receiving their share of load. Each slave begins to mea- sure its share of the load at the moment when all finish receiving their measurement instructions from the corresponding master. From the definition of *Tki* , we can write

$$T11 = t1 + \beta 11 a11 Tms + \beta 11 b11 Tcm \ (20)$$
$$T12 = t1 + \beta 12 a12 Tms + \beta 12 b12 Tcm \ (21)$$

.

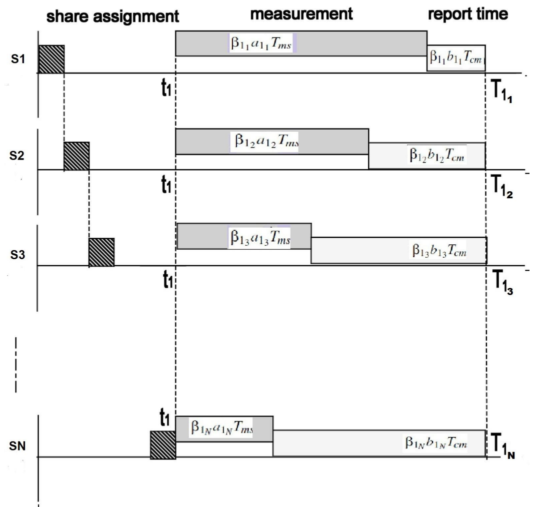

share assignment
measurement
report time
S1
S2
S3
SN
t1
T_{1_1}
T_{1_2}
T_{1_3}
T_{1_N}
$\beta_{1_1}a_{1_1}T_{ms}$
$\beta_{1_1}b_{1_1}T_{cm}$
$\beta_{1_2}a_{1_2}T_{ms}$
$\beta_{1_2}b_{1_2}T_{cm}$
$\beta_{1_3}a_{1_3}T_{ms}$
$\beta_{1_3}b_{1_3}T_{cm}$
$\beta_{1_N}a_{1_N}T_{ms}$
$\beta_{1_N}b_{1_N}T_{cm}$

Figure 4: Timing diagram for a master computer and *N* slaves with simultaneous reporting termination (adopted from [8])

$$T1N = t1 + \beta 1 Na1NTms + \beta 1 Nb1NTcm (22)$$

The total measurement load originating at all the master computers is assumed to be normalized to a unit load. Thus each master computer will handle (1/K) load. So

$$\beta 11 + \beta 12 + \beta 13 + ... + \beta 1N-1 + \beta 1N = 1/K \quad (23)$$

In this case since all processors stop reporting at the same time, we have $T11 = T12 = T13 = ... = T1N$.

Based on the timing diagram, we can write for master computer 1 and its slaves,

$$\beta 11 r11 = \beta 12 r12 \ (24)$$
$$\beta 12 r12 = \beta 13 r13 \ (25)$$

.

.

.

$$\beta 1_{N-2} r1_{N-2} = \beta 1_{N-1} r1_{N-1} \quad (26)$$
$$\beta 1_{N-1} r1_{N-1} = \beta 1_N r1_N \quad (27)$$

where,

$r1_i$ = a1Tms +b1Tcm, i=1,2…..N

Putting the above equations in eq.-(24),

$$\beta 11 = \frac{1}{K(1 + r1 \sum N1_{i=1/2\, r1})} \quad (28)$$

So we can write $\beta 1i$ as

$$\beta_{1_i} = \frac{\frac{1}{r_{1_i}}}{K(\sum_{i=1}^{N} \frac{1}{r_{1_i}})} \quad (29)$$

From the above expression, it can be easily seen that the share of each slave corresponding to its master will entirely depend on the combined speed of the measurement and communication of that slave. The minimum measurement and reporting time of the network will then be given as

$$Tf = T1 = t1 + \frac{(a1\ Tms + b1\ Tcm)1/r1_1}{K(\sum^{N}_{i=1\ /r1i}} \quad (30)$$

Similarly for the master computer p, the generalised equation will be

$$Tfp = T1p = t1 + \frac{ap_1Tms + bp_1Tcm)1_{/r11}}{K(\sum N_{i=1}\ {}^{1/}{}_{r1i})}$$
(31)

For the case of a homogeneous network, each slave corresponding to a master in the network shares the load equally. That is, $\beta 1i = 1/(KN)$, for i =1,2,3,... ,N. So, the minimum measuring and reporting time of the network will be

$$Tf1 = t1 + \frac{a1Tms + b1Tcm}{KN} \quad (32)$$

Similarly we can obtain the above expression for rest of the master computers.

Conclusion

This chapter describes the concept of divisible load scheduling theory and how it can be applied in case of clouds. It also explains the proposed system model, the various notations used and analysis of measurement and reporting time for the two cases that we have considered.

VI. PERFORMANCE EVALUATION

6.1 Introduction

Here we consider the following two cases. In the first case the measurement and reporting time is plotted against the number of slaves corresponding to a master, where the link speed b is varied and measurement speed a is fixed. In the second case, the measurement and reporting time is plotted against the number of slaves corresponding to master, where link speed b is fixed and measurement speed a is varied.

6.2 When Measurement starts Simultaneously and Reporting is done sequentially

In Fig5, the measurement/report time is plotted against the number of homogeneous slaves corresponding to a master when the value of the communication speed b is varied from 0 to 1 at an interval of 0.3 and the value of measurement speed a is fixed to be 1.5. In all cases Tcm =1 and Tms = 1. From the figure we can infer that the faster the communication speed, the smaller the measurement/report time and the measurement/report time levels off after a certain number of slaves for each performance curve. Number of master computers in the cloud doesn't have significant contribution to measurement/report time of a single master.

Fig. 6 shows for the case when the inverse measuring speed a is varied from 1 to 2 at an interval of 0.3 and the inverse link speed b is fixed to be 0.2. The result confirms that the measurement time approaches $b1Tcm$, which in this case is 0.2, as N approaches infinity.

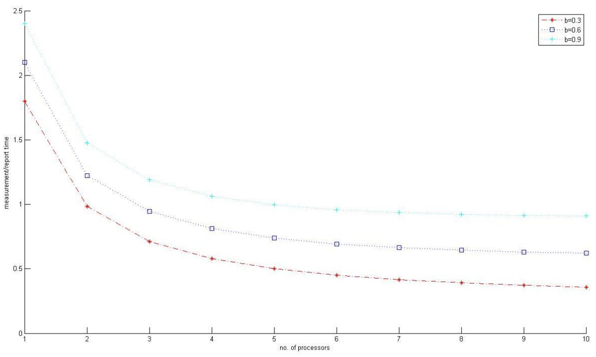


Figure 5: Measurement/report time versus number of slaves corresponding to master and variable inverse link speed b for single level tree network with master and sequential reporting time.

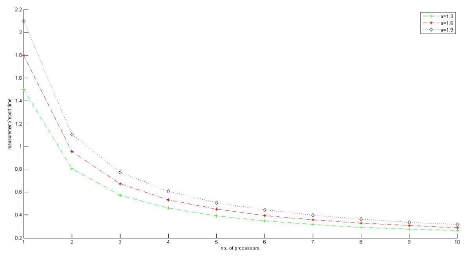


Figure6:Measurement/report timeversus number of slaves corresponding to master and variable inverse measuring speed a for single level tree network with master and sequential reporting time.

6.3 When the Measurement starts Simultaneously and Reporting ends Simultaneously

In Fig. 7, the measurement/report time is plotted against the number of slaves corresponding to a master for the simultaneous measurement start simultaneous reporting termination case. The value the inverse link speed b is varied from 0 to 1 at an interval of 0.3 while the inverse measuring speed a is fixed to be 1.5. In this case the minimum finish time decreases as the number of slaves under a master in the network is increased. This assumes that the communication speed is fast enough to distribute the load to all the slaves under a master.

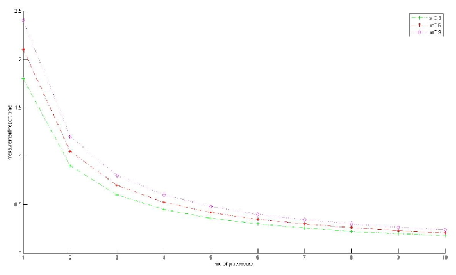


Figure 7: Measurement/report time versus number of slaves under a master and variable inverse link speed b for single level tree network with master

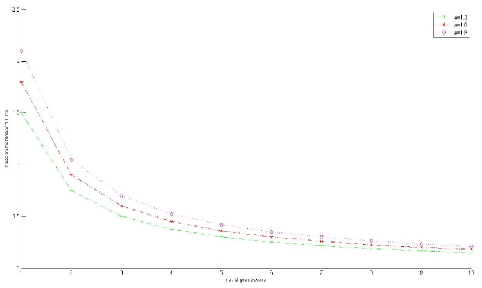


Figure 8: Measurement/report time versus number of slaves under a master and variable inverse measuring speed a for single level tree network with master

Fig. 8 shows for the case when the inverse measuring speed a is varied from 1 to 2 at an interval of 0.3 and the inverse link speed b is fixed to be 0.2.

Conclusion

This chapter evaluates the performance of the two cases that we have considered in this paper. It also shows the simulation results that we have got.

VII. CONCLUSION AND FUTURE WORK

7.1 Conclusion

Fig9 shows the comparison between the measurement/reporting time of both the approaches for the same no. of slave computers corresponding to the same master. Here, for both the cases,1 is taken as the inverse link speed b and 0.5 as the inverse measurement speed a. Number of master computers is taken to be constant equal to 50. The plot shows that the measurement/reporting time is smaller in case of simultaneous reporting as compared to sequential reporting. It is because in case of sequential reporting, some of the slaves receive almost zero load from its master. Number of effective slaves in this case is less as compared to the simultaneous reporting case.Hence with increase in number of slaves with respect to a master, the finishing time remains almost same in case of sequential reporting whereas in case of simultaneous reporting, the finishing time decreases for the increase in number of slaves corresponding to a single master. The graph shows that the finishing time can be improved by increasing the number of slaves under a master computer in a cloud only to some extent before saturation in case of sequential measurement and sequential reporting strategy. But finishing time can be decreased significantly in case of simultaneous measurement start and simultaneous reporting termination by increasing the number of slaves under a single master computer.

As of now basic concepts of Cloud Computing and Load balancing has been discussed and some existing load balancing algorithms have ben studied, those can be applied to clouds. Also, we have studied the closed-form solutions for minimum measurement and reporting time for single level tree networks with different load balancing strategies. The performance of these strategies with respect to the timing and the effect of link and measurement speed was studied. A comparison is also made between different strategies.

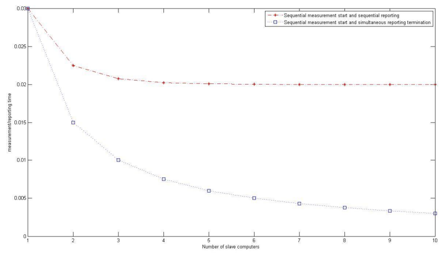


Figure 9: Comparison of Measurement/report time versus number of slaves under a single master under the same conditions of link speed and measurement speed for both cases of reporting

7.2 Future Work

Cloud Computing is a large and vast concept and a very important role is played by load balancing in case of Clouds. There is a wide scope of improvement in the area. Here, only two divisible load-scheduling algorithms that may be applied to clouds, have been discussed, but there are still many approaches that can be applied to balance the load in clouds.Performance of the given algorithms may be increased by varying the parameters.

REFERENCES

[1] Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Ap- proach, TATA McGRAW-HILL Edition 2010.

[2] Martin Randles, David Lamb, A. Taleb - Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Con- ference on Advanced Information Networking and Applications Workshops.

[3] Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceed- ings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, 2008, June 23-26.

[4] Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Sys- tems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[5] http://www-03.ibm.com/press/us/en/pressrelease/ 22613.wss

[6] http://www.amazon.com/gp/browse.html?node=2015900 11

[7] Martin Randles, Enas Odat, David Lamb, Osama Abu-Rahmeh and A. Taleb-Bendiab, ”A Comparative Experiment in Distributed Load Balancing”, 2009 Second Interna- tional Conference on Developments in eSystems Engineering.

[8] Mequanint Moges, Thomas G.Robertazzi, ”Wireless Sensor Networks: Scheduling for Measurement and Data Reporting”, August 31, 2005