A Novel Secured Data Transmission Model with Load Balancing for Cloud Computing

N.Madhavi Latha, Y.Pavan Narasimha Rao
M.tech Scholar, Assistant professor
RamaChandra college of engineering, India

Abstract
Cloud computing is a distribute network for storing and sharing information over internet with scalability. Information security is one the major issue faced by cloud users and service providers. Data retrieval and security are major issues which block clients to adopt clud computing. Also, cloud servers enables the clients to store information on remote servers thereby preventing third party attacks over this data. The main objective of our research work is to store,transmit and extract data in a secure channel.In this proposed model,secured data transmission model using data integrity approach was implemented. Experimental results proved that the proposed integrity model has high accuracy compared to traditional data transmission models.

Index Terms— *cloud server, data integrity, secure transmission.*

I. INTRODUCTION

Cloud computing is utilized for the administrations and applications which keep running on an appropriated system utilizing virtualized assets and got to by basic Web conventions and systems administration guidelines.

Amazon Web Services: One of the best cloud-based organizations is Amazon Web Administrations, which is an Infrastructure as a Service offering that gives you a chance to lease virtual PCs all alone foundation. These new capacities empower applications to be composed and conveyed with insignificant cost and to be quickly scaled and made accessible worldwide as business conditions grant. This is genuinely a progressive change in the way venture processing is made and sent [2]. The most recent decades have fortified the thought that data preparing should be possible all the more effectively midway, on extensive homesteads of processing and stockpiling frameworks open through the Internet. Progressions in systems administration and different territories are in charge of the acknowledgement of the two new processing models and prompted the framework registering development in the mid 1990s and, since 2005, to utility figuring and distributed computing.
LiteGreen, a method to accumulate desktop power by virtualizing the users desktop computing background being a practical machine (virtual machine) then migrate it connecting the users objective desktop machine as well as a Virtual machine server, depending upon if the desktop computing environment continues to be actively used or is inactive. Thus, the users desktop back ground is always on, maintaining its net existencel completely no matter if the users physical desktop machine is shifted and thereby cutback power.
Figure 1 shows an illustration of the front entry design for connection concentrated Internet applications. Users, from side to side dedicated customers, applets, or browser plugins, subject login requests towards the repair cloud. These login needs primary arrive at send off server, which picks appreciable link server and give back its Internet protocol address onto the customer. The customer after that immediately connect with the connection server. The connection server ensures the buyer so if succeeded, a exist TCP link is maintained involving the customer and of course the connection server till the customer logs off.

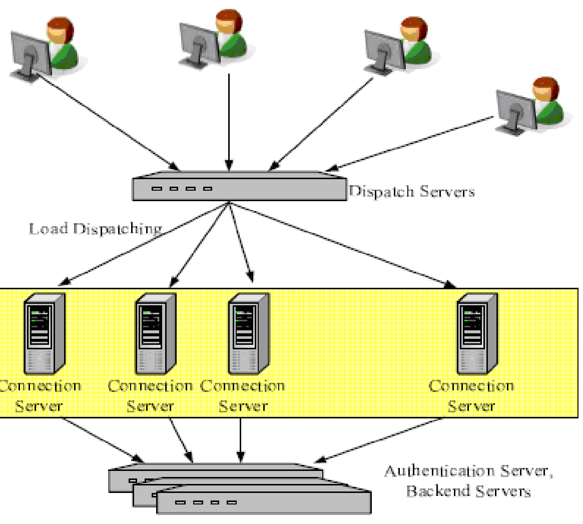


Fig 2.Connection Server Architecture

The TCP connection is normally designed to renew user status (e.g. on-line, busy, off-line, etc.) in order to redistribute additional actions such as conversation and multimedia conferencing to other backend servers. In favor of the application level, each connection server is dependant upon two main constraints: the utmost login rate plus the highest range of sockets it may host. A latest user login rate L is defined as the quantity of latest connection requirements than a connection server processes inside a second. A restriction on login rate Lmax is about to guard CS as well as additional backend services.

II. RELATED WORK

The Dynamic two-phase commit protocol is a variation of Tree 2PC with no foreordained organizer. Understanding messages are sent by every endless supply of the exchange (getting to be prepared). The organizer is resolved progressively by a hustling assertion of messages at the hub where the messages impact. Messages may impact either at an exchange tree hub, or at an edge. In the recent case one of the two edge's hubs is chosen as a facilitator. D2PC is time ideal (among all the examples of a particular exchange tree, and any particular Tree 2PC protocol execution; all occasions have the same tree; every occasion has an alternate hub as organizer): it submits the facilitator and every associate in least conceivable time, permitting a brief arrival of bolted assets.

D2PC is an adjustment of T2PC where the CC is progressively dictated by dashing READY (YES vote) messages, on an each exchange premise, as opposed to being settled, foreordained. For any given exchange D2PC emulates some example of T2PC. We later see that this occasion is ideal (in the arrangement of all cases for a same exchange) in the accompanying sense: It executes the confer choice (i.e., finishes stage 1 of 2PC) in least time. It confers every member in least time.

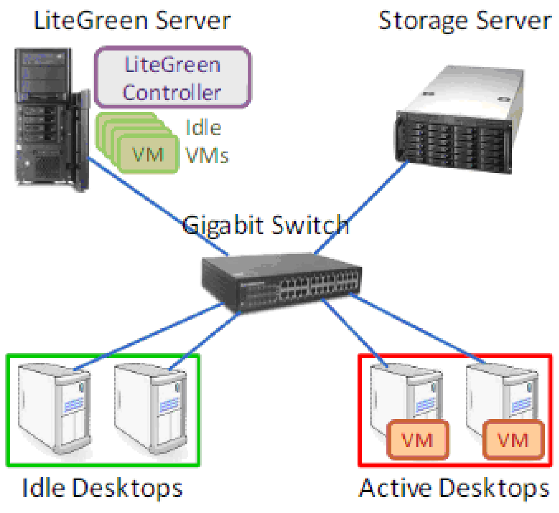


Fig 1. High level liteGreen Architecture

Fig 1 shows the high-level architecture of LiteGreen. The desktop computing communication is enlarged by using a Virtual machine server as well as a common storage node[8]. Generally, present strength be numerous Virtual machine server and/or common storage node. These all essentials are attached by a high speed local area network which can include Gigabit Ethernet. Each desktop machine in addition to server manage a hypervisor. The hypervisor toward the desktop machine hosts a virtual machine a situation where the client OS runs. This virtual machine is migrated towards the server as soon as the user is not just active as well as having the desktop is place to slumber. The users desktop the common storage node, that's also public with the server. The hypervisor according to the server invites the guest virtual machines that might be migrated with it from (idle) desktop machines. The server also contains a controller, which is the simple mind of LiteGreen. The manager receives constant updates from stubs toward the desktop hypervisors toward the grade of user and computing action according to the desktops. The manager also tracks reserve convention according to the server. By this every part of inorder, the regulator orchestrate the movement of virtual machines towards the server and to the desktop machines, and manages the allocation of resources according to the server [1-4]. These data should be secured and information should not be loss in one or the other way. In this project the main work is to protect the data of users. The data which are transacted from the cloud storage area should also be done in more safely manner without giving harm to anyone. Transactional Security should be maintained for cloud transaction in order to protect the transactions. have given the authorization based secure data transactions in cloud computing which identify multiple consistencies issue that can emerge during cloud host transaction process by using week consistency model. He used the algorithm for maintaining the security in different transactions. They identify multiple consistencies issue that can emerge during cloud host transaction process by using week consistency model especially that policy based authority systems are use to apply access control and elaborated a change of lightweight proof enforcement and consistency model defer, punctual, incremental and continue proofs with the view are global consistency then can apply increase the strong protection with the minimum run time overheads. The proof and status it's corrected and collects over the expended a point of time duration below the threat of process an authority policy of the client confident actuality not available circumstances.

Proposed Solution:

Step 1: Create instance job name.

Step 2; Create Job size.

Required powerconsumption=(max power of the processor * required workload)/ 100;

If(job_powerconsumption<threshold)

Monitor job status or add to queue.

Else

Go to Step 3.

Step 3: Job security requirements.

User enters Key pair details of the job.

If(user_key_id==null or user_key_value==null)

Stop creating job.

Else

Store job security details in cloud.

Step 4: Execute Load balancing Algorihtm.

Step 4: Execute VIRTUAL MACHINE control method.

Load balancing Algorithm:

Application usage data: AUD

Application request Service :ARS

Available Resource: AR

```
1. Input: AUD(AR,ARS)
2. availableVIRTUAL MACHINEList //list of available VIRTUAL MACHINEs form each cloud
3. usedVIRTUAL MACHINEList //list of VIRTUAL MACHINEs,currently provision to certain job
4. deployableVirtual machine=null
5. if size(usedVIRTUAL MACHINEList)=size(availbleVIRTUAL MACHINEList) then
6. clear usedVIRTUAL MACHINEList
7. End if
8. for virtual machine in (AUD,AR,ARS) do
9. if virtual machine not in usedVIRTUAL MACHINEList then
10. Add VIRTUAL MACHINE to usedVIRTUAL MACHINEList
11. deployableVirtual machine= virtual machine
12. Break
13. End if
14. End for
15. Save deployableVirtual machine
16. Calculate load balance probable control limit value as:
    • Upper control limit(UCL)=AR+ $(\mathrm{corr}_X + 3 * \sigma_X) * \mathrm{ARS}$
      $(\mu_X + 3 * \sigma_X) * \mathrm{ARS}$
    • Control Limit(CL)=AR+ $\mathrm{Corr} * \mathrm{ARS}$
    • Lower Control Limit(LCL)= AR+ $(\mathrm{Corr}_X - 3 * \sigma_X) * \mathrm{ARS}$
17 Return Bounded values and deployableVirtual machine.
```

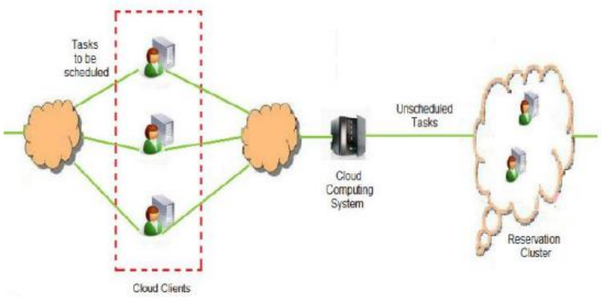
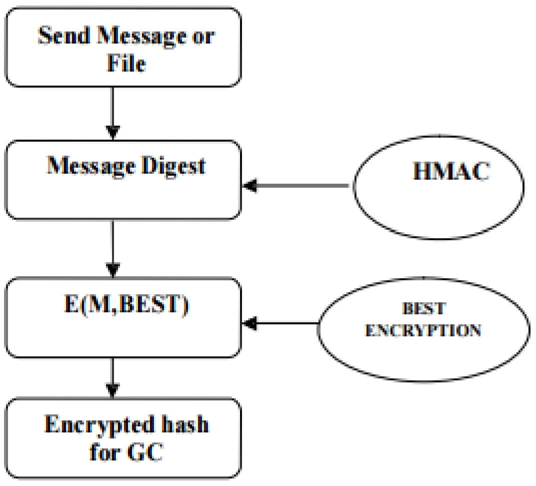


Fig 3: Proposed work flow

In its operations, in order to identify how to stable the load among the virtual machines it first finds out the number of available running virtual machines in the data centre (line 2). In the next step, it gets a record of virtual machines which are previously allocated to job i.e. list of used virtual machines. (line 3). It clears the list if this list is equal to the number of running virtual machines, because that means all the virtual machines are at present allocated to some applications (lines 4-7). Therefore, the first virtual machine from the appropriate and available virtual machine list can be selected for the deployment of the new job request. Lastly, the choosen virtual machine will be additional to the record of old virtual machines so that the load-balancer will not opt for it in the next iteration (lines 8-15).



Cloud server Control Mechanism

```
admissionControl() {
    1.  If (there is any initiated virtual machine)
    2.  For each virtual machinei  in each resource
        provider rp j
    3.  {
    4.  If (! canWait(unew, virtual machinei )) {
    5.  continue;
    6.  }
    7.  Else If (! canInitiateNew(unew, rp j ))
    8.  Return  reject
    9.  If (PotentialScheduleList is  empty)
    10. Return  reject
    11. Else {
    12. Get the list of Virtual machine requests
    13. Monitor the usage details
    14. Get the Load balancing details
    15. List Available virtual machine in virtual
        machine set .
    16. If(LCL<=virtual machine.getUsageData and
        virtual machine.NumberofResources
        <=UCL)
    17. Return accept
    18. Else
    19. Return reject
    20. }
    21. }
```

III. EXPERIMENTAL RESULTS

Experiments are performed by means of the configurations Intel(R) Core(TM)2 CPU 2.13GHz, 2 GB RAM, and the OS platform is Microsoft Windows XP Professional (SP2).

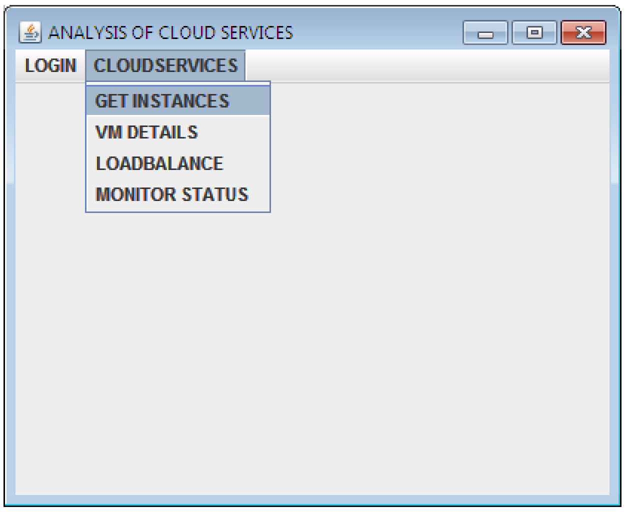


Fig 4: Get instances in the cloud

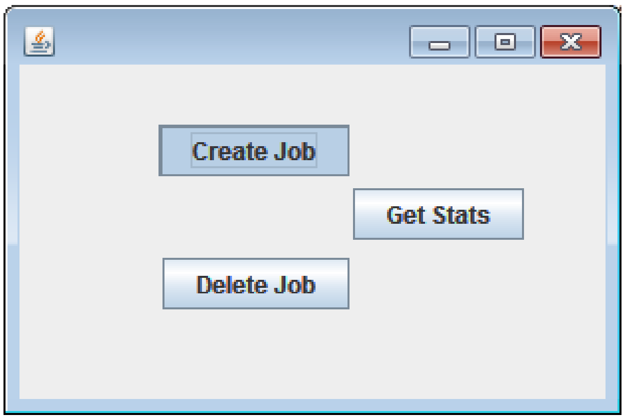
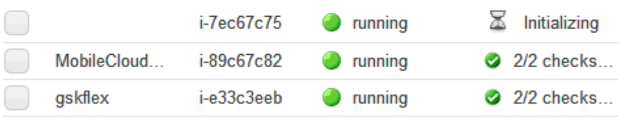


Fig 5: Create an instance in the cloud



		i-7ec67c75	● running	⌛ Initializing
	MobileCloud...	i-89c67c82	● running	✓ 2/2 checks...
	gskflex	i-e33c3eeb	● running	✓ 2/2 checks...

Fig 6: Initializing the instances

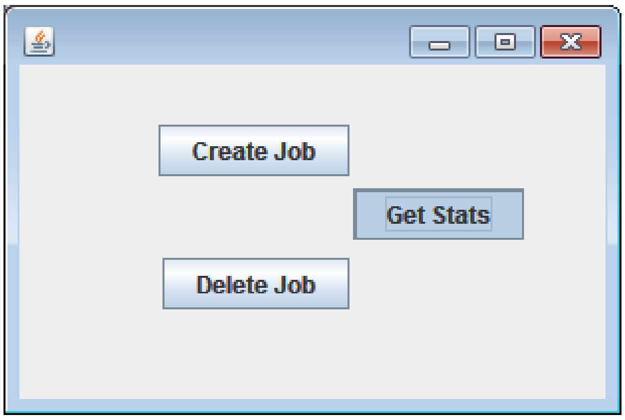


Fig 7: Get the cloud usage statistics

Instance results :[{ReservationId: r-a6e28351,OwnerId: 751633946423,Groups: [],GroupNames: [],Instances: [{InstanceId: i-d1282a14,ImageId: ami-5189a661,State: {Code:

80,Name: stopped},PrivateDnsName: ip-172-31-46-33.us-west-2.compute.internal,PublicDnsName: ,StateTransitionReason: User initiated (2015-10-07 02:04:17 GMT),KeyName: HadoopClusterKey,AmiLaunchIndex: 1,ProductCodes: [],InstanceType: m4.large,LaunchTime: Tue Oct 06 23:52:46 IST 2015,Placement: {AvailabilityZone: us-west-2b,GroupName: ,Tenancy: default},Monitoring: {State: disabled},SubnetId: subnet-babcaed8,VpcId: vpc-805d4fe2,PrivateIpAddress: 172.31.46.33,StateReason: {Code: Client.UserInitiatedShutdown,Message: Client.UserInitiatedShutdown: User initiated shutdown},Architecture: x86_64,RootDeviceType: ebs,RootDeviceName: /dev/sda1,BlockDeviceMappings: [],VirtualizationType: hvm,ClientToken: ParsH1443151317446,Tags: [{Key: SecurityAlg,Value: ParallelCG}, {Key: Name,Value: ParallelMasterNode}],SecurityGroups: [{GroupName: SecurityData,GroupId: sg-899e09ed}],SourceDestCheck: true,Hypervisor: xen,NetworkInterfaces: [{NetworkInterfaceId: eni-fb19629d,SubnetId: subnet-babcaed8,VpcId: vpc-805d4fe2,Description: ,OwnerId: 751633946423,Status: in-use,PrivateIpAddress: 172.31.46.33,PrivateDnsName: ip-172-31-46-33.us-west-2.compute.internal,SourceDestCheck: true,Groups: [{GroupName: SecurityData,GroupId: sg-899e09ed}],Attachment: {AttachmentId: eni-attach-739c2278,DeviceIndex: 0,Status: attached,AttachTime: Fri Sep 25 08:51:57 IST 2015,DeleteOnTermination: true},PrivateIpAddresses: [{PrivateIpAddress: 172.31.46.33,PrivateDnsName: ip-172-31-46-33.us-west-2.compute.internal,Primary: true,}]}],EbsOptimized: true}, {InstanceId: i-d0282a15,ImageId: ami-5189a661,State: {Code: 80,Name: stopped},PrivateDnsName: ip-172-31-46-32.us-west-2.compute.internal,PublicDnsName: ,StateTransitionReason: User initiated (2015-09-25 13:22:13 GMT),KeyName: HadoopClusterKey,AmiLaunchIndex: 0,ProductCodes: [],InstanceType: m4.large,LaunchTime: Fri Sep 25 08:51:57 IST 2015,Placement: {AvailabilityZone: us-west-2b,GroupName: ,Tenancy: default},Monitoring: {State: disabled},SubnetId: subnet-babcaed8,VpcId: vpc-805d4fe2,PrivateIpAddress: 172.31.46.32,StateReason: {Code: Client.UserInitiatedShutdown,Message: Client.UserInitiatedShutdown: User initiated shutdown},Architecture: x86_64,RootDeviceType: ebs,RootDeviceName: /dev/sda1,BlockDeviceMappings: [],VirtualizationType: hvm,ClientToken: ParsH1443151317446,Tags: [{Key: SecurityAlg,Value: ParallelCG}, {Key: Name,Value: SlaveNode}],SecurityGroups: [{GroupName: SecurityData,GroupId: sg-899e09ed}],SourceDestCheck: true,Hypervisor: xen,NetworkInterfaces: [{NetworkInterfaceId: eni-fa19629c,SubnetId: subnet-babcaed8,VpcId: vpc-805d4fe2,Description: ,OwnerId: 751633946423,Status: in-use,PrivateIpAddress: 172.31.46.32,PrivateDnsName: ip-172-31-46-32.us-west-2.compute.internal,SourceDestCheck: true,Groups: [{GroupName: SecurityData,GroupId: sg-899e09ed}],Attachment: {AttachmentId: eni-attach-719c227a,DeviceIndex: 0,Status: attached,AttachTime: Fri Sep 25 08:51:57 IST 2015,DeleteOnTermination: true},PrivateIpAddresses: [{PrivateIpAddress: 172.31.46.32,PrivateDnsName: ip-172-31-46-32.us-west-2.compute.internal,Primary: true,}]}],EbsOptimized: true}]}]You have 2 Amazon EC2 instance(s) running.

Datasize	TransferTIme Existing(s)	TransferTime Proposed(s)
1M	55	37
2M	124	87
3M	198	167
4M	278	243
5M	326	298

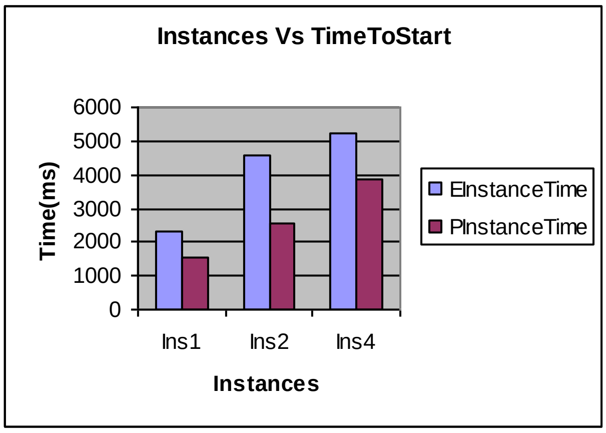


Chart 1: Graph displays the comparison between the number of instances Vs instance initialization time.

IV. CONCLUSION AND FUTURE SCOPE

In cloud environment instances, application deployment, volume creation and their security groups are created, for each instance creation, load balancing are verified in cloud environment. Virtual machine creation and application deployment had been done by using Amazon Web Services. In this research work, we have successfully store,transmit and extract data in a secured channel.In this proposed model,secured data transmission model using data integrity approach was implemented. Experimental results proved that the proposed integrity model has high accuracy compared to traditional data transmission models. In future we may provide security to those properties by allowing access to authenticated users. In future this work can be optimized by using different operating systems.

V. REFERENCES

[1] G. Czajkowski and L. Dayn´es. Multitasking without compromise: a virtual machine evolution. ACM SIGPLAN Notices, 36(11):125.138, Nov. 2001. Proceedings of the 2001 ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA 2001).

[2] S. Devine, E. Bugnion, and M. Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture. US Patent, 6397242, Oct. 1998.

[3] CHEN, Y., DAS, A., QIN, W., SIVASUBRAMANIAM, A., WANG, Q., AND GAUTAM, N. Managing server energy and operational costs in hosting centers. In In Proceedings of the International Conference onMeasurement and Modeling of Computer Systems (2005).

[4] DOYLE, R., CHASE, J., ASAD, O., JIN, W., AND VAHDAT, A. Model- Based Resource Provisioning in aWeb Service Utility. In In Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (2003).

[5] ELNOZAHY, M., KISTLER, M., AND RAJAMONY, R. Energy conservation policies for web servers. In USITS (2003).

[6] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: accounted execution of untrusted code. In Proceedings of the 7th Workshop on Hot Topics in Operating Systems, 1999.

[7] J. S. Robin and C. E. Irvine. Analysis of the Intel Pentium's ability to support a secure virtual machine monitor. In Proceedings of the 9th USENIX Security Symposium, Denver, CO, USA, pages 129.144, Aug. 2000.

[8] LiteGreen: Saving Energy in Networked Desktops Using Virtualization ,Tathagata Das.