International Journal of Advanced Research in

Computer Science Engineering and Information Technology

Volume: 2 Issue: 1 08-Feb-2014,ISSN_NO: 2321-3337

DYNAMIC AND EFFICIENT RESOURCE ALLOCATION ON CLOUD ENVIRONMENT

Mr.j.PraveenChander(Assistant Professor)

M.NAVEENA, S.MADHUBALA, B.ARUNA

Department of computer science, Velammal Institute Of Technology

naveenarukkumani@gmail.com, yesmadhubala@gmail.com, arunajahnavi@gmail.com

ABSTRACT− *In cloud environment dynamic resource allocation is a problematic situation .To resolve this problem, we propose a gossip protocol that ensures fair resource allocation on our sites and applications, dynamically scales and adapts both number of physical machines and allocation to the load changes. We present a protocol that computes an optimal solution without considering memory constraints and prove correctness and convergence properties. This gossip protocol provides an heuristic solution for minimizing the cost of allocation. This work will ensure continuous execution and don't require global synchronization. In proposed system, we make a significant contribution towards engineering a resource management middleware for cloud environments.*

1, INTRODUCTION

Cloud computing" is a term, which involves virtualization, distributed computing, Networking, software and web services. A cloud consists of several elements such as clients ,datacenter and distributed servers. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc. Cloud computing deliver the computing as a services whereby share resources, software, information via Internet which are accessed by the browser. The business software and data are stored in server at Remote Location (CLOUD), Cloud computing provides the kinds of services that are Infrastructure, Software, platform as a services. Gossip Protocol is effective protocol for the dynamic load balance in the distributed system and continuously execute process input & output process. The main principle behind this model is offering computing, storage, and software "as a service. In addition to raw computing and storage, cloud computing providers usually offer a broad range of software services .It will provide services in different ways such as public cloud, private cloud, hybrid cloud. The key feature offered by cloud is users can residue data from anywhere which can be remotely operated in unknown machine by the user. Here we consider the problem of resource allocation and load balancing issues on cloud computing by using gossip protocol in the middleware architecture with the gossiping algorithm in our system, we can replace the drawbacks in our existing system. The main design goals is to satisfy the performance objective, adaptability and scalability.

Keywords- IaaS , Distributed management, Adaptability, Scalability, Resource management, Application placement.

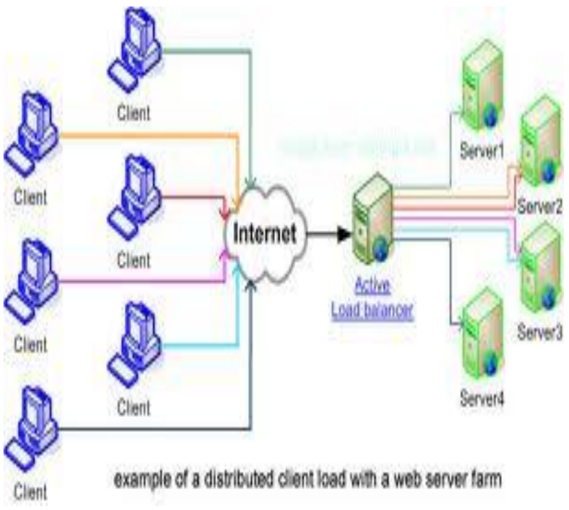
2, SYSTEM ANALYSIS

Existing system:
Application placement in datacenters is often modeled through mapping a set of applications onto a set of machines such that some utility function is maximized under resource constraints. This approach has been taken, and solutions from these works have been incorporated in middleware products. The problem of resource management is application placement and load balancing in processor networks.

Example:
The Service provides some resources to access for the customer, that the resources are access often by the client. The concurrent accessing the capability of bandwidth is to sequencely reduces. Often accesses the resources mean the client need to deal with the Service provider for meet the Service level Objective.

3, EXISTING SYSTEM ALGORITHM

Distributed load balancing:
Definition:
Distributed dynamic load balancing can introduce immense stress on a system in which each node needs to interchange status information with every other node in the system. It is more advantageous when most of the nodes act individually with very few interactions with others.Load balancing in cloud computing systems is really a challenge now. Always a distributed solution is required. Because it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfill the required demands. Jobs can’t be assigned to appropriate servers and clients individually for efficient load balancing as cloud is a very complex structure and components are present throughout a wide spread area. Here some uncertainty is attached while jobs are assigned.



Algorithm Protocol P’computes an optimal solution to

OP(1) and returns configuration matrixA..

```
initialization
1: read ωn,rown(A),Ωn;
2: start the passive and active threads
active thread
3: whiletrue do
4: choosen
```

```
uniformly at random fromN;
5: sendrown(A),Ωnto n;
6: receive rown(A),Ωnfrom n
7: equalize(n
,rown(A),Ωn);
8:writerown(A);
passive thread
1: whiletrue do
2: receive rown(A),Ωnfrom n
3: send row n(A),Ω n to n
4: equalize(n ,row n(A),Ω n);
5: write row n(A);
Proc equalize(j, row j(A),Ω j)
6: l =argmax {vn,vj};
 l=argmin{vn,vj};
7: computeΔωsuch that 1Ω1
(mωm,l−Δω)=1Ωl
(mωm,l+Δω)
8: choose set of modulessand modulem/∈srunning on l
such that
i∈sωi,l <Δωand
i∈sωi, l+ωm,l≥Δω
9: for i∈s do
10: Δω−=ωi,l;αi,l+=αi,l; αi,l =0;
11: αm,l+=αm,lΔω+ωm,l;
; αm,l−=αm,lΔω Ωm ,l
```

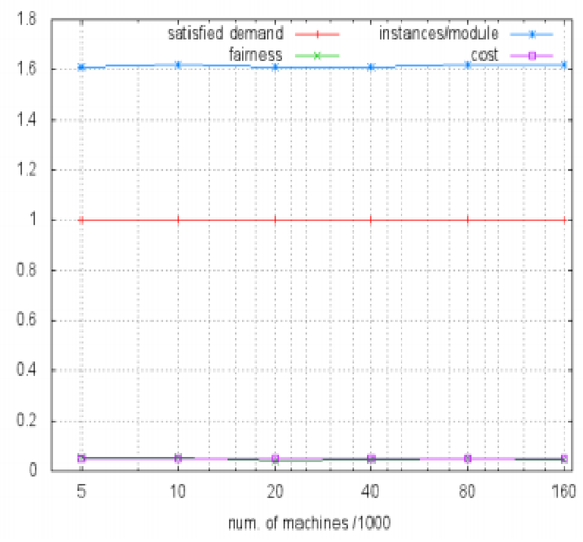
4, Proposed system

This paper introduces a resource allocation protocol that dynamically places site modules (or virtual machines, respectively) on servers within the cloud, Each machine runs a machine manager component that computes the resource allocation policy, which includes deciding the module instances to run. The resource allocation policy is computed by a protocol that runs in the resource manager component. This component takes as input the estimated demand for each module that the machine runs. The computed allocation policy is sent to the module scheduler for implementation/execution, as well as the site managers for making decisions on request forwarding. The overlay manager implements a distributed algorithm that maintains an overlay graph of the machines in the cloud and provides each resource manager with a list of machines to interact. The core contribution of the paper is a gossip protocol, which executes in a middleware platform and meets the design goals. The protocol has two innovative characteristics. First, while gossip protocols for load balancing in distributed systems have been studied before, (to our knowledge) no results are available for cases that consider memory constraints and the cost of reconfiguration, which makes the resource allocation problem hard to solve.

5, Proposed System Algorithm Explanation

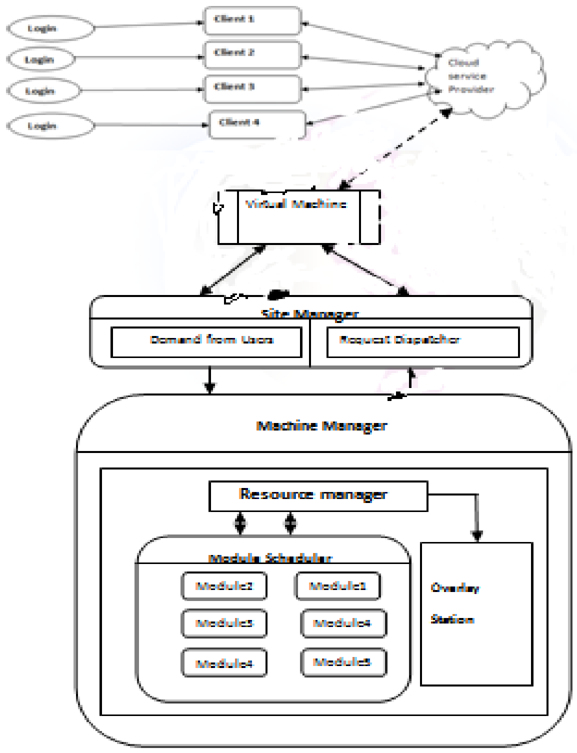
Heuristic algorithm: (implementation)

Branch-and-bound technique and dynamic programming are quite effective but their time complexity often is too high and unacceptable for NP-complete tasks. it always finds the nearest local optimal of low quality. The goal of modern heuristics is to overcome this disadvantage.
Hill-climbing algorithm is effective.
Step 0: Add domain-specific information to select the best path along which to continue searching
Step 1:. Define a heuristic function, h(n), that estimates the “goodness” of a node n. Specifically, h(n) = estimated cost (or distance) of minimal cost path from n to a goal state.
Step 2: The heuristic function is an estimate, based on domain-specific information that is computable from the current state description, of how close we are to a goal
Step 3 : Heuristic h(N)
Step 4: Evaluation function:
$f(N) = g(N) + h(N)$; where:
$g(N)$ is the cost of the best path found so far to N
$h(N)$ is an admissible heuristic
Step 5: Then, best-first search with this evaluation function is called A* search.



SYSTEM ARCHITECTURE:

Datacenter running a cloud environment often contains a large number of machines that are connected by a high-speed network. User accesses the sited hosted by the cloud environment through the public internet. A site is typically accessed through URL that is translated to a network address through a global Directory services such as domain name system. The request of site is routed through the internet to a machine inside the data centre. The resources of cloud are primarily consumed by a module instance where by functionality of sites is made up of one or more modules. Module is service logic of a site. Machine manager component that computes the resource allocation policy, which included deciding the module instance to run. The resource allocation policy is computed by a protocol. This component takes the user input as a estimated demand for each module that machine runs. The allocation policy is sent the module scheduler for execution. Each Site have the site manager. A site manager handles user requests to a particular site. It has two component Demand profiler and request forwarder. The demand profiler estimates the resource demand of each module of the site based on the request.



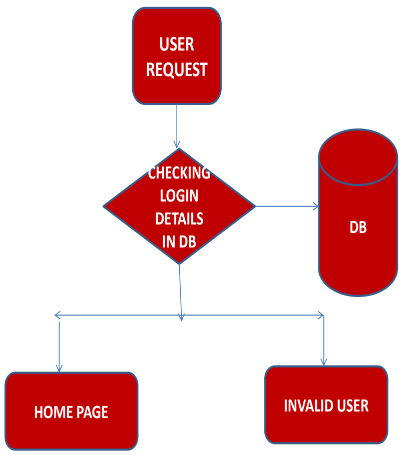
SOFTWARE REQUIREMENTS:

- Ms sql
- Netbeans
- Jdk 1.6

MODULE DESCRIPTION:

User Interface Design:

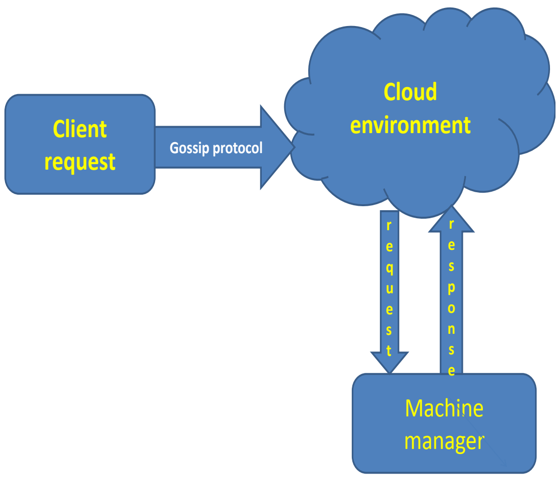
In this module mainly we are focusing the login design page with the Partial knowledge information.we design the windows for the project. These windows are used to send a message from one peer to another.GUI directs user to login screen where user can input his/her user name and password



Implementation of protocol:

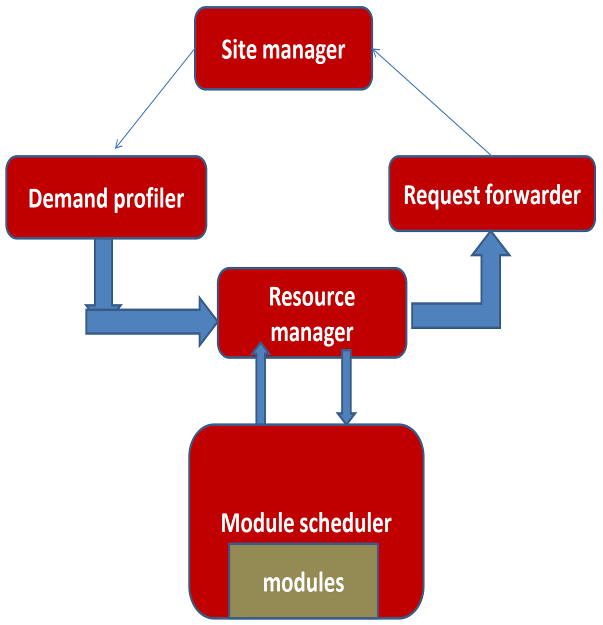
We presented architecture and a generic gossip protocol for application placement in a cloud environment .The protocol can be run in a synchronous or asynchronous mode

Three abstract methods: initInstance () is the initialization method for the specific gossip protocol. choosePeer () is the method for selecting a peer for gossip interaction. updatePlacement() is the method for recomputing the local state during a gossip interaction**.**

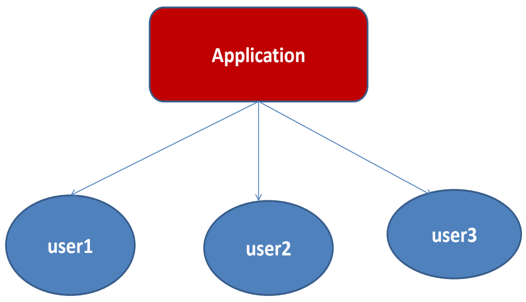


Provider resource management:

Users access sites hosted by the cloud environment through the public Internet. A site is typically accessed through a URL that is translated to a network address through a global directory service, such as DNS. The components of the middleware layer run on all machines. The resources of the cloud are primarily consumed by module instances whereby the functionality of a site is made up of one or more modules. In the middleware, a module either contains part of the service logic of a site each machine runs a machine manager component that computes the resource allocation policy, which includes deciding the module instances to run. The resource allocation policy is computed by a protocol (later in the paper called P*) that runs in the resource manager component. This component takes as input the estimated demand for each module that the machine runs. The computed allocation policy is sent to the Module scheduler for implementation/execution, as well as the site managers for making decisions on request forwarding.

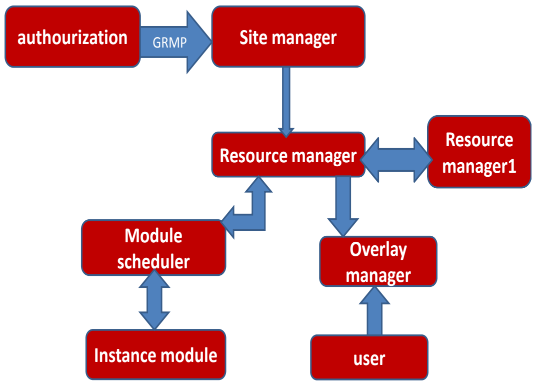


Management Overlay Station:
The overlay manager implements a distributed algorithm that maintains an overlay graph of the machines in the cloud and provides each resource manager with a list of machines to interact with. The overlay station approximate responsibility means in the large cloud environment n number of application and sites are running, each and every sites, application are running by virtual machine to maintained by the Large Cloud Environment, here what we do means to make the user graph of the specific application and the sites.



Adapt Service management:

The authorization hosted their sites/application in the Large Cloud Environment. The Service Level Agreement (SLA) and fine grained from the Cloud service provider and the Authorization. The Service level objectives from the authorization and the site user are also fine grained. In future the authorization needs the enhance such requirement to the own sites and the application. They Designed such features to add their hosted. We address that the placing modules identically instance of modules on machine allocated in cloud resource.



IMPLEMENTATION ON CLOUD:

Step 1: Gossip protocol has the functionality to scale the all machine and fair `
resource allocation and dynamically adapt the load changes.
Step 2: Gossip protocol which executes in a middleware platform
Step 3: Each machine runs a machine manager component that computes the resource allocation policy, which includes deciding the module instances to run
Step 4: Each site has one site manager. A site manager handles user requests to a particular site.

Step 5: The demand profiler estimates the resource demand of each module of the site based on request statistics, QoS targets. This demand estimate is forwarded to all machine managers that run instances of modules belonging to this site
Step 6: Request forwarder sends user requests for processing to instances of modules belonging to this site. Request forwarding decisions take into account the resource allocation policy and constraints such as session affinity.
Step 7: single site manager cannot handle the incoming request stream for a site. However, a scheme for a site manager to scale can be envisioned. For instance, a layer 4/7 switch could be introduced that splits the load among several instances of site managers, whereby each such instance would function like a site manager associated with a single site.

CONCLUSION AND FUTURE ENHANCEMENT

Gossip protocol that computes, in a distributed and continuous fashion, a heuristic solution to the resource allocation problem for a dynamically changing resource demand. The component of such a middleware and present a protocol that can be used to meet our design goals for resource management: fairness of resource allocation with respect to sites, efficient adaptation to load changes and scalability of the middleware layer in terms of both the number of machines in the cloud as well as the number of hosted sites/applications.The results in this paper as building blocks for engineering a resource management solution for large-scale clouds.
Pursuing this goal, we plan to address the following issues in future work: (1) Develop a distributed mechanism that efficiently places new sites. (A mechanism for removing sites is straightforward, since P* will reallocate the freed-up resource.) (2) Extend the middleware design to become robust to various types of failures. (3) Extend the middleware design to span several clusters and several datacenters.

REFERENCES

[1] M. Jelasity, A. Montresor, and O. Babaoglu, “Gossip-based aggregation in large dynamic networks,” ACM Trans. Computer Syst., vol. 23, no. 3, pp. 219–252, 2005.
[2] ——, “T-Man: gossip-based fast overlay topology construction,” Computer Networks, vol. 53, no. 13, pp. 2321–2339, 2009.
[3] F. Wuhib, R. Stadler, and M. Spreitzer, “Gossip-based resource management for cloud environments,” in 2010 International Conference on Network and Service Management.
[4] F. Wuhib, M. Dam, R. Stadler, and A. Clem, “Robust monitoring of network-wide aggregates through gossiping,” IEEE Trans. Network and Service Management, vol. 6, no. 2, pp. 95–109, June 2009.
R. L. Graham, “Bounds on multiprocessing timing anomalies,” SIAM J.
Applied Mathematics, vol. 17, no. 2, pp. pp. 416–429, 1969.
[5] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, “A scalable
application placement controller for enterprise data centers,” in 2007
International Conference on World Wide Web.
[6] H. Shachnai and T. Tamir, “On two class-constrained versions of the

multiple knapsack problem,"Algorithmica, vol. 29, no. 3, pp. 442–467, Dec. 2001.
[7] G. B. Dantzig, "Discrete-variable extremum problems," Operations Research, vol. 5, no. 2, pp. 266–288, 1957.
[8] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," inProc. 1999 Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 1, pp. 126–134.
[9] C. Adam and R. Stadler, "Service middleware for self-managing largescale systems,"IEEE Trans. Network and Service Management,vol.4, no. 3, pp. 50–64, Apr. 2008.
[10] J. Famaey, W. De Cock, T. Wauters, F. De Turck, B. Dhoedt, and P. Demeester, "A latency-aware algorithm for dynamic service placement in large-scale overlays," in 2009 International Conference on Integrated Network Management.
[11] C. Low, "Decentralised application placement,"Future Generation Computer Systems, vol. 21, no. 2, pp. 281–290, 2005.