RESEARCH ARTICLE OPEN ACCESS

Multi Agent System in Distributed Agent Network

K.Ravikumar[1], A. Surendar[2]
([1]Asst.professor, Dept.of.Computer science, Tamil University (Established by the Govt.of.Tamilnadu), Thanjavur.)
([2]Research Scholar, Dept.of.Computer Science, Tamil University, Thanjavur.)

----------**********----------

Abstract:

Cloud computing is a working; process 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 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. Resources allocation policies are computed by protocols. Our contribution includes outlining distributed middleware architecture and presenting one of its key elements: a gossip protocol that (1) ensures fair resource allocation among sites, (2) dynamically adapts the allocation to load changes and (3) scales both in the number of physical machines and applications. The protocol continuously executes on dynamic, local input and does not require global synchronization, as other proposed gossip protocols.

Keywords — **Agent, User Interface.**

----------**********----------

1. INTRODUCTION:

Cloud Computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. This technology allows for much more efficient computing by centralizing data storage, processing and bandwidth. The cloud computing model is comprised of a front end and a back end. These two elements are connected through a network, in most cases the Internet. The front end is the vehicle by which the user interacts with the system; the back end is the cloud itself. The front end is composed of a client computer, or the computer network of an enterprise, and the applications used to access the cloud. The back end provides the applications, computers, servers, and data storage that creates the cloud of services.

We propose a gossip protocol that ensures fair resource allocation among applications, dynamically adapts the allocation to load changes and scales both in the number of physical machines and sites.

2. 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.

2.1.EXISTING SYSTEM

Sybil detection mechanisms rely on the assumption that the benign region is fast mixing. we

recast the problem of finding Sybil users as a semi-supervised learning problem, Sybil detection methods decrease dramatically when the benign region consists of more and more communities they cannot tolerate noise in their prior knowledge about known benign or Sybil nodes and they are not scalable.

Dynamic resource management for a large-scale cloud environment is problematic one. We propose a gossip protocol that ensures fair resource allocation among sites/applications, dynamically adapts the allocation to load changes and scales both in the number of physical machines and sites/applications. We present a protocol that computes an optimal solution without considering memory constraints and prove correctness and convergence properties. Next, we extend that protocol to provide an efficient heuristic solution for the complete problem, which includes minimizing the cost for adapting an allocation. The protocol continuously executes on dynamic, local input and does not require global synchronization, as other proposed gossip protocols do

3. PROBLEM DEFINITION

Our contribution includes outlining distributed middleware architecture and presenting one of its key elements: a gossip protocol that ensures fair resource allocation among sites/applications, dynamically adapts the allocation to load changes and scales both in the number of physical machines and sites/applications. The protocol continuously executes on dynamic, local input and does not require global synchronization, as other proposed gossip protocols

3.1. METHODOLOGY:

3.1.1.Gossip Protocol perform:

Gossip-based protocols have recently gained notable popularity. Apart from traditional applications for database replication gossiping algorithms have been applied to solve numerous other practical problems including failure detection , resource monitoring and data aggregation.

3.1.2.Advantage

- 1.Continuous execution is possible and process
- 2.Suitable for heavy execution processes.
- 3.Resource allocation handling is possible as this work.
- 4.No need for the global synchronization because the protocol gossip already capability for load balancing

4. USER INTERFACE DESIGN:

In this module we design the windows for the project. These windows are used to send a message from one peer to another. We use the Swing package available in Java to design the User Interface. Swing is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs. In this module mainly we are focusing the login design page with the Partial knowledge information. Application Users need to view the application they need to login through the User Interface GUI is the media to connect User and Media Database and login screen where user can input his/her user name, password and password will check in database, if that will be a valid username and password then he/she can access the database.

4.1.CLIENT APPLICATION COMMUNICATION -PROTOCOL

Distributed middleware architecture. Distributed protocols are gossip protocols, specifically. We presented architecture and a generic gossip protocol for application placement in a cloud environment. The protocol can be instantiated for specific management objectives. It computes a distributed heuristic solution to an optimization problem.

1.A generic protocol for application placement;
2.Instantiations for different management objectives.
3.The protocol can be run in a synchronous or asynchronous mode.
Three abstract methods:
1. InitInstance() is the initialization method for the specific gossip protocol.
2. ChoosePeer() is the method for selecting a peer for gossip interaction.

updatePlacement() is the method for recomputing the local state during a gossip interaction.

4.2.SERVICE PROVIDER RESOURCE DEPLOYEMENT

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.

4.3. SERVICE OVERLAY MANAGEMENT

The overlay manager implements VIRTUAL machines in the cloud and provides each APPLICATION 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.

5. 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 resources.

6. CONCLUSION

We are implement a gossip protocol that computes in distributed and continuous fashion, a heuristic solution to resource allocation problem for dynamically changes the resource demand. We evaluate the protocol. we make a significant contribution towards engineering a resource management middleware for cloud environments. We identify a key 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. We presented a gossip protocol P* that computes, in a distributed and continuous fashion, a heuristic solution to the resource allocation problem for a dynamically changing resource demand.

7. REFERENCES

[1] M. Jelasity, A. Montresor, and O. Babaoglu, “Gossip-based aggregationin 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.