ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Load Balancing Models and Algorithms for the Cloud Based on Cloud Partitioning

K. Mahesh
M.Tech Student,
Department of CSE
GITAM University
Hyderabad.

B. Rajendra Prasad Babu
Assistant Professor,
Department of CSE
GITAM University
Hyderabad.

Abstract:

Cloud computing is efficient and scalable but maintaining the stability of processing so many jobs in the cloud computing environment is a very complex problem with load balancing receiving much attention for researchers. Load balancing in the cloud computing surroundings has an imperative impact on the performance. Excellent load balancing makes cloud computing more efficient and improves user satisfaction. In this paper we are presenting various load balancing techniques for cloud partitioning.

Keywords:

Load Balancing, Cloud Partitioning, Load Balancing Models, Public Cloud.

I.INTRODUCTION:

In cloud computing, one of the core design principles is dynamic scalability, which guarantees cloud storage service to handle growing amounts of application data in a flexible manner or to be readily enlarged. By integrating multiple private and public cloud services, hybrid clouds can effectively provide dynamic scalability of service and data migration. For example, a client might integrate the data from multiple private or public providers into a backup or archive file (fig. 1), or a service might capture the data from other services from private clouds, but the midway data and results are stored in hybrid clouds. A load balancing is a method of dividing computing loads among numerous hardware. Due to unpredictable job arrival pattern and the capacities of node in cloud differ for load balancing problem. In this load control is crucial to improve system performance and maintenance [1].

NIST gave a definition of cloud computing as a model for permitting convenient, ubiquitous, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [2]. Load balancing schemes depending on whether the system dynamics are important can be either static and dynamic [3]. Static schemes do not use the system information and are less difficult while dynamic schemes will bring additional costs for the system but can change as the system status changes. A dynamic method is used for its flexibility. This representation has a main controller and balancers to assemble and analyze the information. Therefore the dynamic control has little persuade on the other working nodes. Then the system status provides a basis for choosing the right load balancing strategy.

Although Provable Data Possession (PDP) schemes evolved around public clouds offer a publicly accessible remote interface to check and manage the tremendous quantity of data, the preponderance of presented PDP schemes is incapable of satisfying such an inherent requirement of hybrid clouds in terms of bandwidth and time. Although envisioned as a promising service platform for the Internet, this new data storage paradigm in —Cloud brings about many challenging design issues which have profound influence on the security and performance of the overall organization. One of the largest apprehensions with cloud data storage is that of data integrity verification at un-trusted servers. Inside the cloud, the clients themselves are unreliable or cannot afford the overhead of performing frequent truthfulness verifies. Therefore, for realistic use, it seems more balanced to equip the verification protocol with public verifiability that is predictable to play a more significant role in achieving economies of scale for Cloud Computing [4].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

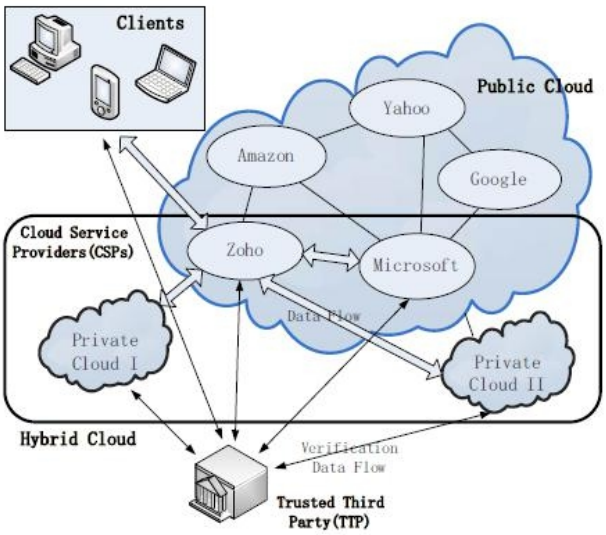

Figure 1: Data storage architecture for hybrid clouds [4].

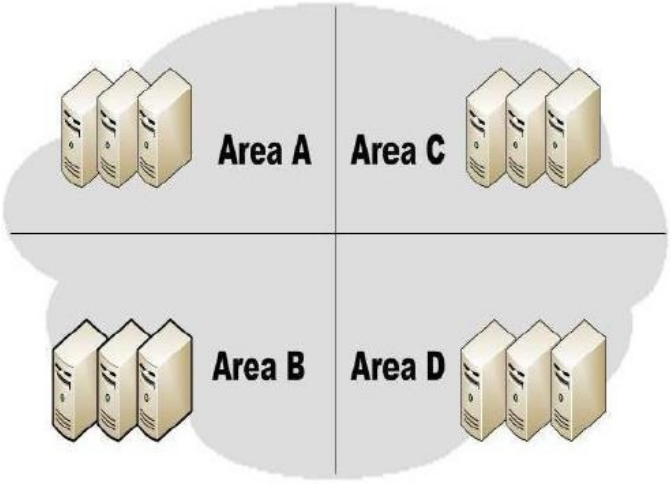

Figure 2: Typical Cloud Partitioning.

There have been many studies of load balancing for the cloud environment. However, load balancing in the cloud is still a new problem that needs new architectures to adapt to many changes.

There are many load balancing algorithms, for example Equally Spread Current Execution Algorithm, Round Robin, and Ant Colony algorithm.A public cloud is based on the standard cloud computing model along with service provided by a service provider [5]. A large public cloud will include many nodes and the nodes in different geographical locations.

A cloud partition is a subarea of the public cloud with divisions based on the geographic locations. The architecture is shown in Fig.2. The load balancing strategy is based on the cloud partitioning concept. After the cloud partitions, the load balancing then starts: when a job arrives at the system through the main controller deciding which cloud partition should receive the job.

Then the partition load balancer decides how to assign the jobs to the nodes. When the load condition of a cloud partition is standard, this partitioning can be proficient locally. If the cloud partition load status is not standard, this job should be transmitted to another partition [1].

In its most basic form, cloud balancing provides an organization with the ability to distribute application requests across any number of application deployments located in data centers and through cloud-computing providers. Cloud balancing takes a broader view of application delivery and applies specified thresholds and service level agreements (SLAs) to every request. The use of cloud balancing can result in the majority of users being served by application deployments in the cloud providers' environments, all though the local application exploitation or internal, private cloud might have more than enough capacity to serve that user [6]. Cloud balancing uses a global application delivery solution to determine, on a per user or customer basis, the most excellent location from which to deliver an application. The decision-making process should include traditional global server load balancing (GSLB) parameters such as:

- Application response time.
- User location.
- Availability of the application at a given implementation location.
- Time of day.
- Current and total capacity of the data center or cloud computing environment in which the application is deployed.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Cloud balancing also presents automation that not only frees up human resources in IT but reduces errors by eliminating the manual performance of repetitive tasks. Applications can be deployed to the cloud with pre-configured templates for security, resources required, and monitoring. Routing decisions must be made in an automated fashion, excluding current cloud balancing solutions permits automated consideration of many more criteria [6].

When the cloud partition is idle, many computing resources are available and relatively few jobs are arriving. In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be used. There are many simple load balance algorithm methods such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin [7]. The Round Robin algorithm is used here for its cleanness. The Round Robin algorithm is one of the simplest load balancing algorithms that passes each latest request to the next server in the queue. The algorithm does not trace the status of each connection so it has no status information. In the regular Round Robin algorithm, each node has a similar opportunity to be selected. Nevertheless, in a public cloud, the configuration and the performance of each node will be not the same; as a result, this scheme may overload some nodes. Consequently an enhanced Round Robin algorithm is used, which called —Round Robin based on the load degree evaluation [1].

Load balancing is a relatively new technique that facilitates networks and resources by providing a maximum throughput with minimum response time. Separating the traffic among various servers, data can be sent and received without lots of delay. Various types of algorithms are available that helps traffic loaded between available servers. Without load balancing, users could feel timeouts, delays, and possible long system responses. Load balancing solutions usually apply redundant servers which help a better distribution of the communication traffic so that the website availability is conclusively settled [8].As far as load balancing algorithm is concerned; Weighted Active Monitoring Load Algorithm, static load balancing algorithm, dynamic load balancing algorithm are commonly used. The Weighted Active Monitoring Load Algorithm is implemented, modifying the Active Monitoring Load Balancer by assigning a weight to each VM for achieving better response time and processing time.

In dynamic load balancing algorithm are three types Simulated Annealing (SA), Orthogonal Recursive Bisection (ORB), and Eigenvector Recursive Bisection (ERB). SA directly minimizes the cost function by a process analogous to slow physical cooling. ORB method cuts the graph into two by a vertical cut, and then cuts each half into two by a horizontal cut, and then each quarter is cut vertically, and so on. ERB also doing similar to ORB but the cutting is done using an eigenvector of a matrix with the same sparsity structure as the adjacency matrix of the graph. Static load balancing algorithm equally divide load among available server. By this approach the traffic on the servers will be disdained easily and consequently it will make the situation more imperfectly [8].

Static algorithm divides the traffic equally, is announced as round robin algorithm. However, there were lots of problems appeared in this algorithm. Therefore, weighted round robin was defined to improve the critical challenges associated with round robin. In this algorithm each servers have been assigned a weight and according to the highest weight they received more connections. Dynamic algorithms designated proper weights on servers and by searching in whole network a lightest server preferred to balance the traffic.

However, selecting an appropriate server needed real time communication with the networks, which will lead to extra traffic added on system. Distributed system load balancing is still an active area of research in which load balancer attempts to improve the performance of a distributed system by using the processing power of the entire system to smooth out periods of high congestion at individual nodes, this is done by transferring some of the workload of heavily loaded nodes to other nodes for processing [8].

Major problem with the current load balancing algorithm is they does not consider the current utilization of VM resources. These algorithms divide the upcoming request equally without considering the available memory and storage space and current CPU utilization of the VM resource. These applications are dependent on other applications. These applications are executed either in parallel or sequentially. Cloud users try to access the multiple instances of different applications during a short time period. This will cause a significant arrival peak [9].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

As cloud computing is a new area for research and development, developing a dynamic load balancing algorithm is a major challenge for cloud service provider. This algorithm will ensure the optimum utilization of cloud resources. The rest of paper is organized as follows. In Section II describes about background information of cloud computing. Section III describes related work in fields of authentication systems followed by a conclusion in Section IV.

II. BACKGROUND:

Cloud computing is Internet based computing, whereby shared resources, software and information are provided to computers and other devices on-demand, like a public utility. Infrastructure as a Service is a single tenant cloud layer where the Cloud computing vendor‘s dedicated resources are only shared with contracted clients at a pay-per-use fee. This greatly minimizes the need for huge initial investment in computing hardware such as servers, networking devices and processing power. Software as a Service also operates on the virtualized and pay-per-use costing model whereby software applications are leased out to contracted organizations by specialized SaaS vendors. This is traditionally accessed remotely using a web browser via the Internet. Platform as a service cloud layer works like IaaS but it provides an additional level of —rented functionality. Clients using PaaS services transfer even more costs from capital investment to operational expenses but must acknowledge the additional constraints and possibly some degree of lock-in posed by the additional functionality layers.

Cloud computing is the problem of load balancing. Further, while balancing the load, certain types of information such as the number of jobs waiting in queue, job arrival rate, CPU processing rate, and so forth at each processor, as well as at neighboring processors, may be exchanged among the processors for improving the overall performance. Good load balance will improve the performance of the entire cloud. However, there is no common method that can adapt to all possible different situations. Various methods have been developed in improving existing solutions to resolve new problems. Each particular method has advantage in a particular area but not in all situations. Therefore, the current model integrates several methods and switches between the load balance method based on the system status.

III. RELATED WORK:

In 2013, Xu, Gaochao et al [1] presented A load balancing model based on cloud partitioning for the public cloud. The load balancing model is aimed at the public cloud which has numerous nodes with distributed computing resources in many different geographic locations. Thus, this model divides the public cloud into several cloud partitions. When the environment is very large and complex, these divisions simplify the load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs while the balancer for each cloud partition chooses the best load balancing strategy. The load balance solution is done by the main controller and the balancers. The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller data sets will lead to the higher processing rates. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs. The relationship between the balancers and the main controller is shown in Fig.3.

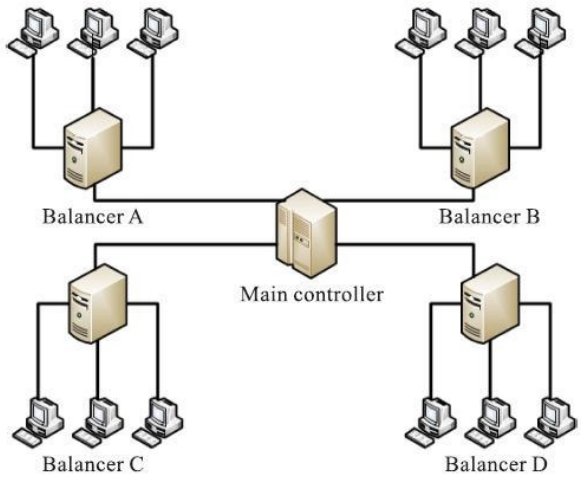

Figure 3: Relationships between the main controllers, the balancers, and the nodes.

When a job arrives at the public cloud, the first step is to choose the right partition. The cloud partition status can be divided into three types: (1) Idle: When the percentage of idle nodes exceeds , change to idle status. (2) Normal: When the percentage of the normal nodes exceeds , change to normal load status.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

(3) Overload: When the percentage of the overloaded nodes exceeds , change to overloaded status. The parameters , and are set by the cloud partition balancers. The main controller has to communicate with the balancers frequently to refresh the status information. The cloud partition balancer gathers load information from every node to evaluate the cloud partition status. This evaluation of each node‘s load status is very important. The first task is to define the load degree of each nodes. The node load degree is related to various static parameters and dynamic parameters. The static parameters include the number of CPU‘s, the CPU processing speeds, the memory size, etc. Dynamic parameters are the memory utilization ratio, the CPU utilization ratio, the network bandwidth, etc [1].

When the cloud partition is inactive, many computing resources are accessible and relatively few jobs are incoming. In this phase, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be employed. There are many straightforward load balance algorithm methods such as the Weight Round Robin, the Random algorithm, and the Dynamic Round Robin [7]. When the cloud partition is normal, jobs are arriving much faster than in the idle state and the situation is far more complex, so a different strategy is used for the load balancing. Each user wants his jobs completed in the shortest time, so the public cloud needs a method that can complete the jobs of all users with reasonable response time [1].

Shivaratri et al [3] focuses on the problem of judiciously and transparently redistributing the load of the system among its nodes so that overall performance is maximized. They also discussed several key issues in load distributing for general-purpose systems, including the motivations and design trade-offs for load-distributing algorithms. They also presented load-distributing policies used in existing systems and draw conclusions about which algorithm might help in realizing the most benefits of load distributing. They compare various load distribution algorithms with their benefits and losses. The ability of load distributing to improve performance is intuitively obvious when work arrives at some nodes at a greater rate than at others, or when some nodes have faster processors than others. Performance advantages are not so obvious when all nodes are equally powerful and have equal workloads over the long term [3].

Zhu, Yan et al [4] suggested “Efficient provable data possession for hybrid clouds. They focused on the construction of PDP scheme for hybrid clouds, supporting privacy protection and dynamic scalability. They first provide an effective construction of Cooperative Provable Data Possession (CPDP) using Homomorphic Verifiable Responses (HVR) and Hash Index Hierarchy (HIH). This construction uses homomorphic property, such that the responses of the client‘s challenge computed from multiple CSPs can be combined into a single response as the final result of hybrid clouds. By using this mechanism, the clients can be convinced of data possession without knowing what machines or in which geographical locations their files reside. More importantly, a new hash index hierarchy is proposed for the clients to seamlessly store and manage the resources in hybrid clouds. Their experimental results also validate the effectiveness of our construction [6]. Lori MacVitte presented a Cloud Balancing: The Evolution of Global Server Load Balancing. Cloud balancing is still new, but the technology to add value is available today.

The ability to distribute connections across the globe based upon an array of inputs such as geographic location, device type, the state of servers in one location or another, and balanced loads is real. There will no doubt be more advances in the future as cloud balancing becomes more mainstream. A solution that is poised to take on new standards and enables use of existing standards, such as IPv6 and DNSSEC, should be the first stop for IT in the quest for agile data centers. Cloud computing has introduced a cost-effective alternative to building out secondary or even tertiary data centers as a means to improve application performance, assure application availability, and implement a strategic disaster-recovery plan. When they can leverage cloud application deployments in addition to local application deployments, organizations gain a unique opportunity to optimize application delivery from technical and business standpoints [6].Doddini Probhuling L. presented Load Balancing Algorithms In Cloud Computing. They discussed with the cloud computing requirements for access control, migration, security, data availability, trust issues and sensitive information. The algorithms used in the cloud computing for load balancing, this information might be useful in the research associated with cloud computing.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

The genetic programming paradigm permits the evolution of computer programs which can perform alternative computations conditioned on the outcome of intermediate calculations, which can perform computations on variables of many different types, which can perform iterations and recursions to achieve the desired result, which can define and subsequently use computed values and subprograms, and whose size, shape, and complexity is not specified in advance [8].

Naimesh D. Naik and Ashilkumar R. Patel proposed Load Balancing Under Bursty Environment for Cloud Computing. Current algorithms do not consider the bursty workloads and hence it will decrease the system performance. Their proposed algorithm considers the current VM resource utilization and bursty workloads for distributing the load to each VM instances. They expect that using the proposed algorithm cloud service provider can meet the service level agreements (SLA) without purchasing additional resources. This algorithm also ensures that none of VM resources is over utilized when another one is underutilized. This will increase the system performance and provide faster response time. This will also increase the economic profit of an organization as all the resources are better utilized so there is no need for extra resources for handling the request [9].Experimental setup of the Load balancing in the cloud computing requires the allocation of the requests that has been made by the requester to the resource.

Load Balancer balances the mechanism of allocating the proper resource to the proper request to maintain the balance. There are variety of Algorithms for Load Balancing for Cloud Computing. Different Algorithm Uses different strategy to balance the load by allocating the request to resource which is free at that time of period. A monitoring agent would be continuously monitoring the CPU usage, memory and storage space usage and expected load and current load data for each virtual instances. All the data are transferred to the load balancer by monitoring agent. Basedon the data of each virtual instances the request is transfer the appropriate node controller server where virtual machines are running and from where different instances are provided to different users [9].As cloud computing is a new area for research and development, developing a dynamic load balancing algorithm is a major challenge for cloud service provider. This algorithm will ensure the optimum utilization of cloud resources.

This algorithm will provide faster response time and it will improve the system performance in the case of changing user demands. This will help the cloud service provider to meet the service level agreements. This algorithm will cut the economic cost for an organization because less resources will be required than static algorithms to handle the user requests [9]. In year 2012, Kaviani, Nima et al [10] proposed MANTICORE: A framework for partitioning software services for hybrid cloud. They discussed MANTICORE, a framework that allows software developers to analyze a monolithic (i.e single host) software service to make it suitable for hybrid cloud. In order to verify the accuracy of the cost models, they compare the execution and data transfer measurements of models generated by MANTICORE to those of real deployments for DayTrader. Since our profiling data was collected on the premise machine, the models and the real deployment for the machine on premise are identical. In a hybrid or a full cloud deployment, the models are generated by applying linear fitting techniques. They compared the generated cost models with the real deployment of the DayTrader application for settings where the deployment is fully in the cloud or is MHD [10].

They [10] proposed an extension to existing application partitioning techniques to provide for hybrid deployment of software services. The evaluation on DayTrader showed that the new approach can more effectively contribute towards an optimized hybrid cloud deployment.In particular, it showed that: the costs of a hybrid deployment extrapolated from monitoring a single-host test version of the service were at least 81.3% accurate; the context-sensitive modeling of service behavior provided a better representation to optimize placement of software function execution; this formulation of the objective function for optimization allows developers to tune the tradeoff between end-to-end round-trip time and deployment costs; and that a hybrid deployment using MANTICORE, while providing similar scalability as a full cloud deployment, offers better round-trip latency [10].

In recently year 2013, Patel, Parveen et al [11] suggested Ananta: cloud scale load balancing. In the proposed Ananta (meaning infinite in Sanskrit) resulted from examining the basic requirements, and taking an altogether different approach. Ananta is a scalable software load balancer and NAT that is optimized for multitenant clouds.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

It achieves scale, reliability and any service anywhere via a novel division of the data plane functionality into three separate tiers. At the second tier, a scalable set of dedicated servers for load balancing, called multiplexers (Mux), maintain connection flow state in memory and do layer-4 load distribution to application servers. A third tier present in the virtual switch on every server provides stateful NAT functionality. Using this design, no outbound traffic has to pass through the Mux, thereby significantly reducing packet processing requirement. Another key element of this design is the ability to offload multiplexer functionality down to the host. As discussed in §2, this design enables greater than 80% of the load balanced traffic to bypass the load balancer and go direct, thereby eliminating throughput bottleneck and reducing latency. This division of data plane scales naturally with the size of the network and introduces minimal bottlenecks along the path [11].

Ananta has been implemented as a service in the Windows Azure cloud platform. We considered implementing Ananta functionality in hardware. However, with this initial Ananta version in software, they were able to rapidly explore various options in production and determine what functions should be built into hardware, e.g., they realized that keeping per-connection state is necessary to maintain application uptime due to the dynamic nature of the cloud. Similarly, weighted random load balancing policy, which reduces the need for per-flow state synchronization among load balancer instances, is sufficient for typical cloud workloads. They also consider the evaluation of these mechanisms, regardless of how they are implemented, to be a key contribution of this work. More than 100 instances of Ananta have been deployed in Windows Azure since September 2011 with a combined capacity of 1Tbps. It has been serving 100,000 VIPs with varying workloads.

It has proven very effective against DoS attacks and minimized disruption due to abusive tenants. Compared to the previous solution, Ananta costs one order of magnitude less; and provides a more scalable, flexible, reliable and secure solution overall [11]. Scale-out In-network Processing: Figure 4 illustrates the main components of a traditional load balancer. For each new flow, the load balancer selects a destination address (or source for SNAT) depending on the currently active flows and remembers that decision in a flow table.

Subsequent packets for that flow use the state created by the first packet. Traditional NAT and load balancing algorithms (e.g., round-robin) require knowledge of all active flows; hence all traffic for a VIP must pass through the same load balancer. This forces the load balancer into a scale-up model. A scale-up or vertical scaling model is one where handling more bandwidth for a VIP requires a higher capacity box.

Network routers, on the other hand, follow a scale-out model. A scale-out or horizontal scaling model is one where more bandwidth can be handled by simply adding more devices of similar capacity. Routers scale out because they do not maintain any per-flow state that needs synchronization across routers and therefore one can add or remove additional routers easily. Ananta design reduces the in-network functionality needed for load balancing to be such that multiple network elements can simultaneously process packets for the same VIP without requiring per-flow state synchronization [11].

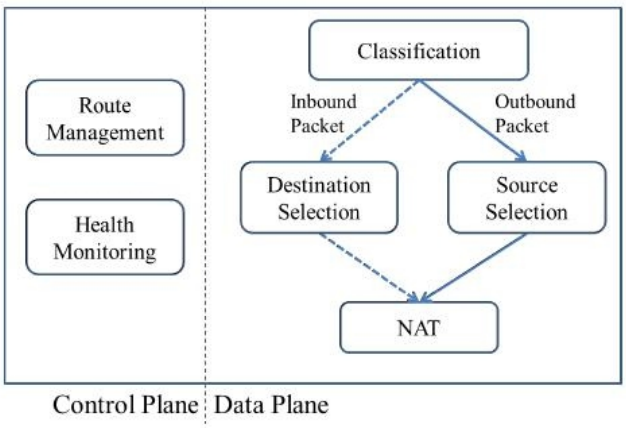

Figure 4: Components of a traditional load balancer.

This design choice is enabled because they can make certain assumptions about our environment. One of the key assumptions is that load balancing policies that require global knowledge, e.g., weighted round robin (WRR), are not required for layer-4 load balancing. Instead, randomly distributing connections across servers based on their weights is a reasonable substitute for WRR. In fact, weighted random is the only load balancing policy used by our load balancer in production. The weights are derived based on the size of the VM or other capacity metrics [11].

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

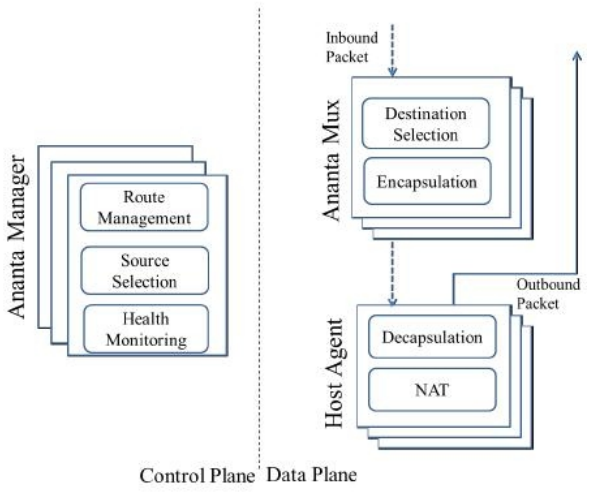

Figure 5: The Ananta Architecture.

Ananta is a loosely coupled distributed system comprising three main components (Figure 5)—Ananta Manager (AM), Multiplexer (Mux) and Host Agent (HA). To better understand the details of these components, let first discuss the load balancer configuration and the overall packet flow. All packet flows are described using TCP connections but the same logic is applied for UDP and other protocols using the notion of pseudo connections. Ananta consists of three components — Ananta Manager, Ananta Mux and Host Agent. Each component is independently scalable. Manager coordinates state across Agents and Muxes. Mux is responsible for packet forwarding for inbound packets. Agent implements NAT, which allows all outbound traffic to bypass Mux. Agents are co-located with destination servers [11].

The load balancer receives a VIP Configuration for every VIP that it is doing load balancing and NAT for. An Endpoint refers to a specific transport protocol and port on the VIP that is load balanced to a set of DIPs.Packets destined to an Endpoint are NAT'ed to the DIP address and port. SNAT specifies a list of IP addresses for which outbound connections need to be Source NAT'ed with the VIP and an ephemeral port. A unique feature of Ananta is a distributed NAT for outbound connections. Even for outbound connections that need source NAT (SNAT), Ananta ensures that outgoing packets do not need to go through Mux. In order to scale to the 100s of terabit bandwidth requirement of intra-DC traffic, Ananta offloads most of the intra-DC traffic to end systems. This is done by a technique we call Fastpath.

The key idea is that the load balancer makes its decision about which DIP a new connection should go to when the first packet of that connection arrives. Once this decision is made for a connection it does not change. Therefore, this information can be sent to the HAs on the source and destination machines so that they can communicate directly. This results in the packets being delivered directly to the DIP, bypassing Mux in both directions, thereby enabling communication at full capacity supported by the underlying network. This change is transparent to both the source and destination VMs [11].

The Ananta Manager (AM) implements the control plane of Ananta. It exposes an API to configure VIPs for load balancing and SNAT. Based on the VIP Configuration, it configures the Host Agents and Mux Pools and monitors for any changes in DIP health. Ananta Manager is also responsible for keeping track of health of Muxes and Hosts and taking appropriate actions. The AM uses Paxos to elect a primary, which is responsible for performing all configuration and state management tasks. Ananta uses BGP to achieve scale out among multiple active instances of Mux. A traditional approach to scaling out middle box functionality is via DNS. Each instance of the middle box device, e.g., load balancer, is assigned a public IP address.

The authoritative DNS server is then used to distribute load among IP addresses of the instances using an algorithm like weighted round robin. When an instance goes down the DNS server stops giving out its IP address. This approach has several limitations. First, it is harder to get good distribution of load because it is hard to predict how much load can be generated via a single DNS resolution request. For example, load from large clients such as a mega proxy is always sent to a single server. Second, it takes longer to take unhealthy middle box nodes out of rotation due to DNS caching – many local DNS resolvers and clients violate DNS TTLs. And third, it cannot be used for scale out of stateful middle boxes, such as a NAT [11].In same year, P. Jamuna and R.Anand Kumar proposed Optimized Cloud Partitioning Technique to Simplify Load Balancing. This model divides the cloud environment into several partitions by making use of cloud clustering technique, helps the providers to simplify the process of load balancing. Thus this proposed technique achieves higher performance and stability in cloud.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Arriving patterns of jobs are unpredictable, thus allocating and processing of many jobs over cloud environment among various node is a complex problem. And the capacity of each node also differs from each other. Hence load should be balanced among multiple nodes in order to improve the stability and system performance.The public cloud environment which has enormous nodes over large and different geographical location. Our proposed model divides the cloud environment into several partitions. This partitioning technique helps the providers to simplify the process of load balancing [12].

The huge cloud environment consist of numerous node and it is partitioned into an n clusters based on our Cloud clustering technique. Our proposed model consists of main controller which controls all load balancer in each cloud cluster. The main controller maintains all the details include index table, its current status information of all load balancer in each cluster. The index table consists of both static parameters (number of CPU, Processing speed, memory size etc.) and dynamic parameters (network bandwidth, CPU and memory utilization ratio).

The cloud environment includes numerous node and the nodes are different geographical location. Portioning of large cloud into cluster into cluster helps to manage its performance effectively. The algorithm used in this is able to automatically supervise the load balancing work through load balancer assigned to each cluster. Thus CPU and Memory can be utilized properly. Thus our proposed technique achieves higher performance, stability, optimal resource utilization, minimize response time and application down time over cloud environment [12].

Application partitioning has been explored extensively for client-server architectures in a variety of network settings: i) in a wide-area setting [13], ii) within a LAN [14], or iii) across a wireless network [15]. These client-server systems are different from our setting since we are in essence working towards partitioning between backend servers, not server and client. At a low-level this difference manifests itself as different optimization goals for partitioning. In client-server partitioning the goal is to partition software by pushing code tightly coupled to front-end interaction towards the client and pushing the code that works on shared persistent data to the server.

Partitioning for hybrid clouds differs in that extra constraints on geographical placement of code, types of instances leased from the cloud, and their associated charges in the public cloud must be taken into consideration. Within the context of cloud, Volley [16] increases performance and reduces data center traffic by data relocation, yet it does not deal with code partitioning. Efforts like Conductor [17] and HybrEx [18] suggest hybrid deployments for cost optimizations or security considerations in the cloud, but their main focus in on Map/Reduce type of applications. Other approaches such as CloudCmp [19], CiteSeerx [20], and the work by Truong & Dustdar [21] look into cost savings of software service deployment to the cloud by analyzing resource consumption; yet their cost-considerations do not drive deployment decisions using partitioning techniques.

CloneCloud [22] optimizes for high performance and minimum resource usage for applications in mobile and embedded devices by offloading the execution to the cloud; but optimizing towards a cheaper deployment is not the focus for CloneCloud. Cloudward Bound [23] and COPE [24] optimize for cost of deployment in a hybrid setting, however there are several differences between these approaches and MANTICORE.

For these approaches, partitioning and relocation of components happens at the level of application servers (or VMs) not the finer level of code entities (i.e. functions) as is the case with MANTICORE. Furthermore, none of these approaches supports context sensitivity. Finally, while COPE does not account for latency, Cloudward Bound enforces the accepted latency by defining an upper limit constraint, whereas MANTICORE allows for software developers to decide about their preferred cost-to-latency ratios.

IV. CONCLUSION:

It is important to evaluate solutions for cloud balancing implementations with an eye toward support for the needs of an actual IT department. The global and local application delivery solution chosen to drive a cloud balancing implementation should be extensible, automated, and flexible, and the vendors involved need to look favorably upon standards. There are challenges associated with the implementation of such a strategy, some of which might take years to address.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

But the core capabilities of global and local application delivery solutions today make it possible to build a strong, flexible foundation that will enable organizations to meet current technical and business goals and to extend that foundation to include a more comprehensive cloud balancing strategy in the future.

Acknowledgment:

I would like to thank B. Rajendra Prasad Babu Assistant Professor, for accepting me to work under his valuable guidance. He closely supervised the work over the past few months and advised many innovative ideas, helpful suggestions, valuable advices and support.

REFERENCES:

[1]Xu, Gaochao, Junjie Pang, and Xiaodong Fu. "A load balancing model based on cloud partitioning for the public cloud." IEEE Tsinghua Science and Technology, Vol. 18, no. 1, pp. 34-39, 2013.

[2]P. Mell and T. Grance, —The NIST definition of cloud Computing, online available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2012.

[3]N. G. Shivaratri, P. Krueger, and M. Singhal, —Load distributing for locally distributed systems, Journal Computer, vol. 25, no. 12, pp. 33-44, Dec. 1992.

[4]Zhu, Yan, Huaixi Wang, Zexing Hu, Gail-Joon Ahn, Hongxin Hu, and Stephen S. Yau. "Efficient provable data possession for hybrid clouds." In Proceedings of the 17th ACM conference on Computer and communications security, pp. 756-758. ACM, 2010.

[5]A. Rouse, —Public cloud, available at: http://searchcloudcomputing.techtarget.com/definition/public-cloud.

[6]Lori MacVittie, —Cloud Balancing: The Evolution of Global Server Load Balancing, F5 white paper, online available at: http://www.f5.com/pdf/white-papers/cloud-balancing-white-paper.pdf, 2013.

[7]D. MacVittie, "Intro to load balancing for developers — The algorithms, https://devcentral.f5.com/blogs/us/introto-load-balancing-for-developers-ndash-the-algorithms, 2012.

[8]Doddini Probhuling L. —Load Balancing Algorithms In Cloud Computing, International Journal of Advanced Computer and Mathematical Sciences, ISSN 2230-9624. Vol. 4, Issue 3, pp. 229-233, 2013.

[9]Naimesh D. Naik and Ashilkumar R. Patel —Load Balancing Under Bursty Environment for Cloud Computing, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181, Vol. 2, Issue 6, pp. 17 – 26, June – 2013.

[10]Kaviani, Nima, Eric Wohlstadter, and Rodger Lea. "MANTICORE: A framework for partitioning software services for hybrid cloud." In Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, pp. 333-340, 2012.

[11]Patel, Parveen, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A. Maltz, Randy Kern et al. "Ananta: cloud scale load balancing." In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 207-218. ACM, 2013.

[12]P. Jamuna and R.Anand Kumar — Optimized Cloud Partitioning Technique to Simplify Load Balancing, International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, Volume 3, Issue 11, pp. 820 – 822, November 2013.

[13]S. Chong, J. Liu, A. Myers, X. Qi, K. Vikram, L. Zheng, and X. Zheng, —Building secure web applications with automatic partitioning, in Proceedings of the Symposium on Operating Systems Principles (SOSP), 2009.

[14]G. Hunt and M. Scott, —The Coign automatic distributed partitioning system, in Proc. of the Symposium on Operating Systems Design and Implementation (OSDI), 1999.

[15]R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, —Wishbone: Profile-based Partitioning for Sensornet Applications, in Proc. of the NSDI, 2009.

[16]S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, and A. Wolman, —Volley: Automated data placement for geo-distributed cloud services. in NSDI, 2010.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[17]A. Wieder, P. Bhatotia, A. Post, and R. Rodrigues, —Orchestrating the deployment of computations in the cloud conductor, in NSDI, 2012.

[18]S. Y. Ko, K. Jeon, and R. Morales, —The HybrEx model for confidentiality and privacy in cloud computing, in Proc. of HotCloud, 2011.

[19]A. Li, X. Yang, S. Kandula, and M. Zhang, —CloudCmp: Shopping for a Cloud Made Easy, HotCloud, 2010.

[20]P. Teregowda, B. Urgaonkar, and C. Giles, —CiteSeerx: a Cloud Perspective, in Proc. of the HotCloud Workshop, 2010.

[21]H. L. Truong and S. Dustdar, —Composable cost estimation and monitoring for computational applications in cloud computing environments, Procedia CS, vol. 1, no. 1, pp. 2175–2184, 2010.

[22]B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, —Clonecloud: elastic execution between mobile device and cloud. in Proc. of the European Symposium on Operating Sytems (EuroSys), 2011.

[23]M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, —Cloudward bound: planning for beneficial migration of enterprise applications to the cloud, in SIGCOMM, 2010.

[24]C. Liu, B. T. Loo, and Y. Mao, —Declarative automated cloud resource orchestration, in Proc. of the Symposium on Cloud
Computing, 2011.