RESEARCH ARTICLE OPEN ACCESS

A Reliable System for Scaling Application over Clouds

Sreepathi .B [1], Sheela B.P [2], Saisunethra K [3], Spandana [4]
SnehaM.S [5], Rashmi.W [6]

Head of the Department [1], Assistant Professor [2]
Department of Information Science &Engineering
V.T.U/ R.Y.M.E.C, Bellary
Karnataka - India

ABSTRACT

In this paper we introduce a reliable system for scaling application over clouds used for load balancing and application scaling on a cloud. The basic philosophy of our approach is defining a reliable and optimal operation regime and attempting to maximize the number of servers operating in this regime. Idle and lightly-loaded servers are switched to one of the sleep states to save energy. Load balancing of the entire cloud system can be handled dynamically by using virtualization technology where it becomes possible to remap virtual machines (VMs) and physical resources according to the change in load. In order to achieve the best performance, the virtual machines have to fully utilize its services and resources by adapting to the cloud computing environment dynamically. The load balancing and proper allocation of resources must be guaranteed in order to improve resource utility.

Keywords:- Reliable Scaling System, Application Scaling, Idle Servers, Server Application Manager Samk ,Virtual Machine Monitor (VMM) Server Consolidation

I. INTRODUCTION

On a cloud computing platform, dynamic resources can be effectively managed using virtualization technology. Virtualization technologies enable application computation and data to be hosted inside virtual containers (e.g. virtual disks) which are decoupled from the underlying physical resources. These virtualization- based clouds provide a way to build a large computing infrastructure by assessing remote computational, storage and network resources. Since a cloud typically comprises a large amount of virtual and physical servers, to efficiently managing this virtual infrastructure has attracted considerable interest in recent years. Thus, the important objectives of this paper are to determine How to achieve a reliable system for scaling application over clouds and effective load balance in a cloud computing environment.

II. MOTIVATION AND RELATED WORK

Cloud Computing concepts date back to the 1950s when large-scale mainframes were made available to schools and corporations. The mainframe's colossal hardware infrastructure was installed in what could literally be called a ―server room‖ (since the room would generally only be able to hold a single mainframe), and multiple users were able to access the mainframe via ―dumb terminals‖ – stations whose sole function was to facilitate access to the mainframes. Due to the cost of buying and maintaining mainframes, an organization wouldn't be able to afford a mainframe for each user, so it became practice to allow multiple users to shareaccess to the same data storage layer and CPU power from any station. By enabling shared mainframe access, an organization would get a better return on its investment in this sophisticated piece of technology.

In the last few years packaging computing cycles and storage and offering them as a metered service became a reality. Large farms of computing and storage platforms have been assembled and a fair number of Cloud Service Providers (CSPs) offer computing services based on three cloud delivery models SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service).Warehouse-scale computers (WSCs) are the building blocks of a cloud infrastructure. A hierarchy of networks connect,50, 000 to 100, 000 servers in a WSC. The servers are housed in racks; typically, the 48 servers in a rack are connected by a 48-port Gigabit Ethernet switch. The switch has two to eight up-links which go to the higher level switches in the network hierarchy.

The motivation of the survey of existing load balancing techniques in cloud computing is to encourage the amateur researcher to contribute in developing more efficient load balancing algorithms. This will benefit interested researchers to carry out further work in this research area.

This paper is organized as follows:
Section III presents the system model,
Section IV shows the study and analysis of the existing load balancing techniques in cloud computing,
Section V identifies the metrics considered in the existing load balancing techniques and carries out the comparison between them based on those identified metrics and
Section VI concludes the paper. To the best of our knowledge, none of the techniques has focused on energy consumption and carbon emission factors that is a dire need of cloud computing.

III. A RELIABLE SCALING SYSTEM MODEL

There are three primary contributions : a new model of cloud servers that is based on different operating regimes with various degrees of \energy efficiency" (processing power versus energy consumption); a novel algorithm that performs load balancing and application scaling to maximize the number of servers operating in the energy-optimal regime; and analysis and comparison of techniques for load balancing and application scaling using three differently-sized clusters and two different average load profiles. The objective of the algorithms is to ensure that the largest possible number of active servers operate within the boundaries of their respective optimal operating regime. The actions implementing this policy are: (a) migrate VMs from a server operating in the undesirable-low regime and then switch the server to a sleep state; (b) switch an idle server to a sleep state and reactivate servers in a sleep state when the cluster load increases; (c) migrate the VMs from an overloaded server, a server operating in the undesirable-high regime with applications predicted to increase their demands for computing in the next reallocation cycles.

SYSTEM ARCHITECTUR

MODULES

- *System Model*
- *Server*
- *Creating Load*
- *A reliable system model*

The fig:3.1 shows the architecture of the reliable scaling system with client servers and reliable scaling system

MODULES DESCRIPTION:

System Model:
In this module, we design the system, such that client makes request to server. Usually, a it is designed with adequate resources in order to satisfy the traffic volume generated by end-users. In general, a wise provisioning of resources can ensure that the input rate is always lower than the service rate. In such a case, the system will be capable to efficiently serve all users' requests. Applications for one instance family have similar profiles, e.g., are CPU-, memory-, or I/O-intensive and run on clusters optimized for that profile; thus, the application interference with one another is minimized.

A reliable system model

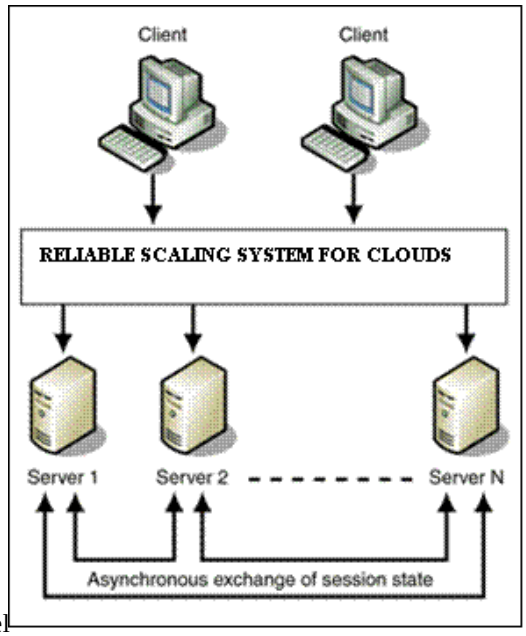

FIG 3.1:RELIABLE SCALING SYSTEM ARCHITECTURE

The normalized system performance and the normalized power consumption are different from server to server; yet, warehouse scale computers supporting an instance family use the same processor or family of processors and this reduces the effort to determine the parameters required by our model. In our model the migration decisions are based solely on the vCPU units demanded by an application and the available capacity of the host and of the other servers in the cluster. The model could be extended to take into account not only the processing power, but also the dominant resource for a particular instance family, e.g., memory for R3, storage for I2, GPU for G2 when deciding to migrate a VM. This extension

would complicate the model and add additional overhead for monitoring the application behavior.

Server

The term server consolidation is used to describe: switching idle and lightly loaded systems to a sleep state; (2) workload migration to prevent overloading of systems; or (3) any optimization of cloud performance and energy efficiency by redistributing the workload. In this module we design the Server System, where the server processes the client request. Cloud is a large distributed system of servers deployed in multiple data centers across the Internet. The goal of a cloud is to serve content to end-users with high availability and high performance. Cloud serves a large fraction of the Internet content today, including web objects (text, graphics and scripts), downloadable objects (media files, software, documents), applications (e-commerce, portals), live streaming media, on-demand streaming media, and social networks. Besides better performance and availability, cloud also offload the traffic served directly from the content provider's origin infrastructure, resulting in cost savings for the content provider.

Creating Load

In this module, we create the load to the server. Though, in this paper we focus exclusively on critical conditions where the global resources of the network are close to saturation. This is a realistic assumption since an unusual traffic condition characterized by a high volume of requests, i.e., a flash crowd, can always overfill the available system capacity. In such a situation, the servers are not all overloaded. Rather, we typically have local instability conditions where the input rate is greater than the service rate. In this case, the balancing algorithm helps prevent a local instability condition by redistributing the excess load to less loaded servers.

A reliable system model

The objective of the algorithms is to ensure that the largest possible number of active servers operate within the boundaries of their respective optimal operating regime. The actions implementing this policy are: (a) migrate VMs from a server operating in the undesirable-low regime and then switch the server to a sleep state; (b) switch an idle server to a sleep state and reactivate servers in a sleep state when the cluster load increases; (c) migrate the VMs from an overloaded server, a server operating in the undesirable-high regime with applications predicted to increase their demands for computing in the next reallocation cycles. We present a new mechanism for redirecting incoming client requests to the most appropriate server, thus balancing the overall system requests load. Our mechanism leverages local balancing in order to achieve global balancing. This is carried out through a periodic interaction among the system nodes. Depending on the network layers and mechanisms involved in the process, generally request routing techniques can be classified in cloud request routing, transport-layer request routing, and application-layer request routing.fig:3.2 shows the dataflow diagram of a reliable scaling system.fig 3.3 shows the UML diagram of reliable scaling system.

.

VI. RELIABLE SCALING ALGORITHMS

The objective of the algorithms is to ensure that the largest possible number of active servers operate within the boundariesof their respective optimal operating regime. The actions implementing this policy are: (a) migrate VMs from a server operating in the undesirable-low regime and then switch the server to a sleep state; (b) switch an idle server to a sleep state and reactivate servers in a sleep state when the cluster load increases; (c) migrate the VMs from an overloaded server, a server operating in the undesirable-high regime with applications predicted to increase their demands for computing in the next reallocation cycles. The clustered organization allows us to accommodate some of the desirable features of the strategies for server consolidation. For example, when deciding to migrate some of the VMs running on a server or to switch a server to a sleep state, we can adopt a conservative policy similar to the one advocated by auto scaling to save energy. Predictive policies, such as the ones discusse timal regime when historical data regarding its workload indicates that it is likely to return to the optimal regime in the near future.

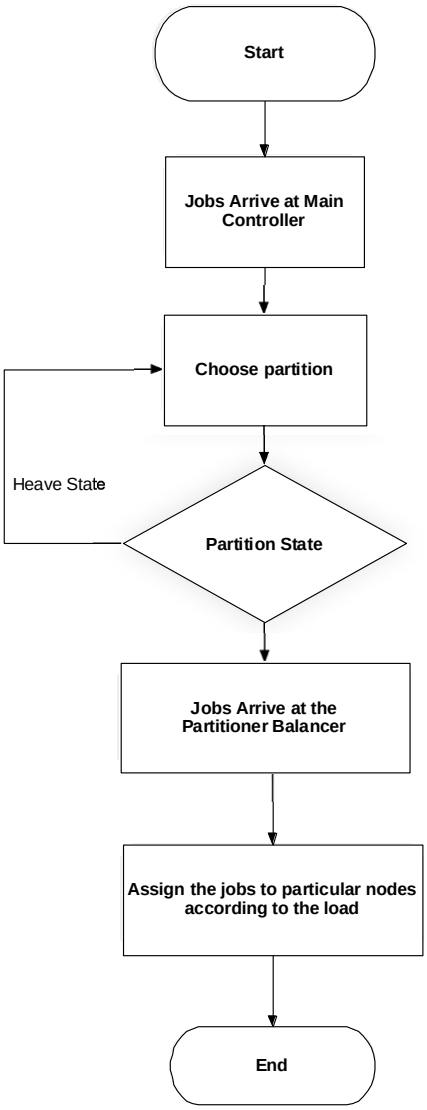

Fig: 3.2 Data Flow diagram of a Reliable scaling system

Scaling decisions. The Server Application Manager SAM_k is a component of the Virtual Machine Monitor (VMM) of a server S_k. One of its functions is to classify the applications based on their processing power needs over a window of w reallocation intervals in several categories: rapidly increasing resource demands (RI), moderately increasing (MI),stationary (S), moderately decreasing (MD), and rapidly decreasing (RD). This information is passed to the cloud leader whenever there is the need to migrate the VM running the application SAM_k interacts with the cluster leader and with the application managers of servers accepting the migration of an application currently running on server S_k. A report sent to the cluster leader includes the list of applications currently running on S_k, their additional demands of over the last reallocation cycle and over a window of w reallocation intervals, and their classification as RI/MI/S/MD/RD over the same window. The scaling decisions are listed in the order of their energy consumption, overhead, and complexity:

1) Local decision - whenever possible, carry out vertical application scaling using local resources only.
(2) In-cluster, horizontal or vertical scaling - migrate some of the VMs to the other servers identified by the leader; wake-up some of the servers in a sleep state or switch them to one of the sleep states depending on the cluster workload.
(3) Inter-cluster scaling - when the leader determines that the cluster operates at 80% of its capacity with all servers running, the admission control mechanism stops accepting new applications. When the existing applications scale up above 90% of the capacity with all servers running then the cluster leader interacts with the leaders of other clusters to satisfy the requests. This case is not addressed in this paper. All decisions take into account the current demand for processor capacity, as well as the anticipated load.

1. Local, vertical scaling. The anticipated load at the end of the current and the following scheduling cycle allow the server to continue operating in the optimal regime.

$$g_k(t+\tau_k) \leq \gamma_k a_k^{opt,h} \;\&\; \gamma_k a_k^{opt,l} \leq g_k(t+2\tau_k) \leq \gamma_k a_k^{opt,h}. \quad (1)$$

2. In-cluster scaling. The anticipated load could force the server to transition to the suboptimal-low, suboptimal-high,or undesirable-high regimes, respectively:

$$\begin{gathered} g_k(t+\tau_k) \leq \gamma_k \alpha_k^{opt,h} \;\&\; \gamma_k \alpha_k^{sopt,l} \leq g_k(t+2\tau_k) \leq \gamma_k \alpha_k^{opt,l} \\ g_k(t+\tau_k) \leq \gamma_k \alpha_k^{opt,h} \;\&\; \gamma_k \alpha_k^{opt,h} \leq g_k(t+2\tau_k) \leq \gamma_k \alpha_k^{sopt,h} \\ g_k(t+\tau_k) \leq \gamma_k \alpha_k^{opt,h} \;\&\; g_k(t+2\tau_k) > \gamma_k \alpha_k^{sopt,h}. \end{gathered} \quad (2)$$

The leader includes the server in the WatchList when Equations 2 are satisfied.

3. Inter-cluster scaling. The anticipated load could force the server to transition to the undesirable-low regime.

$$g_k(t+\tau_k) \leq \gamma_k \alpha_k^{opt,h} \;\&\; g_k(t+2\tau_k) \leq \gamma_k \alpha_k^{sopt,l}. \quad (3)$$

The leader includes the server in the MigrationList in case of Equation 3 In addition to the lazy approach .when a server

operates within the boundaries of the sub optimal high regime until its capacity is exceeded, one could use an anticipatory strategy. To prevent potential SLA violations in this case we force the migration from the suboptimal-high region as soon as feasible.

Synchronous operation. The algorithm executed by SAM-k every T_k units of time is:

1. Order applications based on the demand. Compute the actual rate of increase or decrease in demand over a window of w reallocation cycles
2. Compute servers available capacity.. If Equations 1 or 2 are satisfied send an imperative request for application migration.
3. If first or third equations in 2 are satisfied send a warning including application history.
4. Else, reallocate CPU cycles to allow each application its largest rate of increase .If time elapsed from the last status report is larger than in-cluster reporting period send an update to the leader

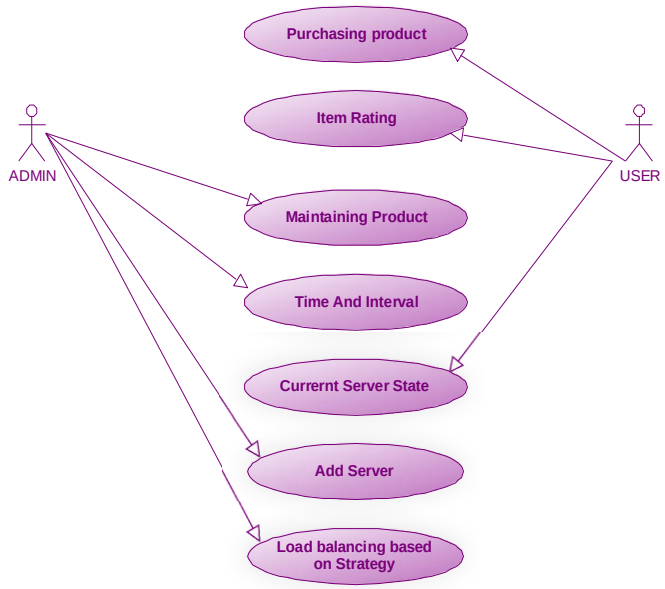

Fig:3.4 UML Diagram of Reliable scaling system

Advantages Of Proposed System:

- After load balancing, the number of servers in the optimal regime increases from 0 to about 60% and a fair number of servers are switched to the sleep state.
- There is a balance between computational efficiency and SLA violations; the algorithm can be tuned to maximize computational efficiency or to minimize SLA violations according to the type of workload and the system management policies

IV. CONCLUSION

In the project, we implemented an optimized reliable scaling system which combines multi-strategy mechanism with the prediction mechanism. We adopt different control strategy for different domains which are divided according to the host resource utilization. We design and implement optimized control strategy for load balancing based on live disruptive migration of virtual Machine

The realization that power consumption of cloud computing centers is significant and is expected to increase substantially in the future motivates the interest of the research community in energy-aware resource management and application placement policies and the mechanisms to enforce these policies. Low average server utilization and its impact on the environment make it imperative to devise new energy-aware policies which identify optimal regimes for the cloud servers and, at the same time, prevent SLA violations A quantitative evaluation of an optimization algorithm or an architectural enhancement is a rather intricate and time consuming process; several benchmarks and system configurations are used to gather the data necessary to guide future developments. For example, to evaluate the effects of architectural enhancements supporting Instruction-level or Data-level

Parallelism on the processor performance and their power consumption several benchmarks are used. The results show different numerical outcomes for the individual applications in each benchmark. Similarly, the effects of an energy-aware algorithm depend on the system configuration and on the application and cannot be expressed by a single numerical value.

Research on reliable scaling system and resource management in large scale systems often use simulation for a quasi-quantitative and, more often, a qualitative evaluation of optimization algorithms or procedures. As stated in \First, they (WSCs) are a new class of large-scale machines driven by a new and rapidly evolving set of workloads. Their size alone makes them difficult to experiment with, or to simulate efficiently." It is rather difficult to experiment with the systems discussed in this paper and this is precisely the reason why we choose simulation. The results of the measurements reported in the literature are difficult to relate to one another. For example, the wakeup time of servers in the sleep state and the number of servers in the sleep state are reported for the Auto Scale system; yet these configures would be different for another processor, system configuration, and application. We choose computational efficiency, the ratio of the amount of normalized performance to normalized power consumption, as the performance measure of our algorithms. The amount of

useful work in a transition processing benchmark can be measured by the number of transactions, but it is more difficult to assess for other types of applications. SLA violations in a transaction processing benchmark occur only when the workload exceeds the capacity of all servers used by the application, rather than the capacity of individual servers. Thus, in our experiment there is no SLA violation because there are servers operating in low-load regimes. We need to balance computational efficiency and SLA violations; from The lazy approach would eliminate this effect. Even the definition of an ideal case when a clairvoyant resource manager makes optimal decisions based not only on the past history, but also on the knowledge of the future can be controversial. For example, we choose as the ideal case the one when all servers operate at the upper boundary of the optimal regime; other choices for the ideal case and for the bounds of the have regimes could be considered in case of fast varying, or unpredictable workloads. The have-regime model introduced in this paper reflects the need for a balanced strategy allowing a server to operate in an optimal or near-optimal regime for the longest period of time feasible. A server operating in the optimal regime is unlikely to request a VM migration in the immediate future and to cause an SLA violation, one in a sub-optimal regime is more likely to request a VM migration, while one in the undesirable high regime is very likely to require VM migration. Servers in the undesirable-low regime should be switched to a sleep state as soon as feasible. The model is designed for clusters built with the same type of processors and similar configurations; the few parameters of the model are then the same for all the servers in the cluster. The clustered organization allows an effective management of servers in the sleep state as they should be switched proactively to a running state to avoid SLA violations. It also supports effective admission control, capacity allocation, and load balancing mechanisms as the cluster leader has relatively accurate information about the available capacity of individual servers in the cluster. Typically, we see a transient period when most scaling decisions require VM migration, but in a steady-state, local decisions become dominant.

REFERENCES

[1] D. Kusic, J. O. Kephart, N. Kandasamy, and G. Jiang/Power and performance management of virtualized computing environments via lookahead control." Proc 5th Int. Conf. Autonomic Comp. (ICAC2008), pp. 3{12,2008.

[2] C. Tung, M. Steinder, M.Spreitzer, and G. Paci_ci. \A scalable application placement controller for enterprise data centers." Proc. 16th Int. Conf. World Wide Web (WWW2007), 2007

[3] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. "Energy-aware autonomic resource allocation in multi- tier virtualized environments." IEEE Trans. on Services Computing, 5(1):2–19, 2012.

[4] [2] J. Baliga, R.W.A. Ayre, K. Hinton, and R.S. Tucker. "Green cloud computing: balancing energy in process- ing, storage, and transport." Proc. IEEE, 99(1):149-167,2011.

[5] [3] L. A. Barroso and U. Hozle. "The case for energy- proportional computing." IEEE Computer, 40(12):33–37, 2007.

[6] [4] L. A. Barossso, J. Clidaras, and U.Hözle. The Data- center as a Computer; an Introduction to the Design of Warehouse-Scale Machines. (Second Edition). Morgan & Claypool, 2013.

[7] [5] A. Beloglazov, R. Buyya "Energy efficient resource man- agement in virtualized cloud data centers." Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Comp., 2010

[8] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon,S. Cholia, J. Shalf, H. Wasserman, N. J. Wright. \Performance analysis of high performance computing applications on the Amazon Web services cloud." Proc. IEEESecond Int. Conf. on Cloud Computing Technology and Science, pp. 159{168, 2010.

[9] D. Gmach, J. Rolia, L. Cherkasova, G. Belrose, T. Tucricchi, and A. Kemper. \An integrated approach to resource pool management: policies, e_ciency, and quality metrics." Proc. Int. Conf. on Dependable Systems and Networks, pp. 326{335, 2008.

[10] Google. \Google's green computing: e_ciency at scale."http://static.googleusercontent.com/external content/untrusted dlcp/www.google.com/en/us/green/pdfs/google-green-computing.pdf (Accessed on August 29, 2013).