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

Research on Heuristic Based Load Balancing Algorithms in Cloud Computing

Mohammed Vaziuddin
M.Tech Student,
Dept of IT,
Sreenidhi Institute of Science and Technology.

K.Sreenu
Assistant Professor,
Dept of IT,
Sreenidhi Institute of Science and Technology.

Abstract:
Since the recommendation of the idea of cloud computing in the year 2006, cloud computing has drawn heaps of consideration from both industry and scholarly territory. A few innovations, for example, virtualization shaped the premise of cloud computing, while some different advances goes about as framework change procedures in distributed computing. Among the utilized advancements, Load adjusting is fundamental and critical in enhancing framework execution and keeping up clients' understanding. In this paper, we concentrate on load adjusting calculations ordinarily utilized as a part of distributed computing. Through our examination we proposed our enhanced online load balancing algorithm. We utilize a few exploratory results to demonstrate its energy and effectiveness. We utilize CloudSim as a test system to check our thoughts.

Keywords:
Cloud computing, Cloud Sim, Load balancing algorithm, Heuristic.

1 Introduction:

Cloud computing is developed as another plan of action in IT Industry. Huge Internet organizations, for example, google, amazon sold its excess figuring and capacity at an aggressive cost to those in need. Cloud computing worldview is well known among both administration suppliers and administration clients for its qualities, for example, minimal effort, high versatility, on request provisioning, and programmed administration. For those administration suppliers, cloud computing is another benefit point by making utilization of hard ware assets possessed.

For those clients, by leasing virtualized assets gave by administration suppliers, a lot of examination concerning IT infra-structures could be saved. Cloud computing sellers utilize virtualization and multi-specialist innovations to enhance framework's use, alongside the well known pay per utilize estimating model to guarantee a long haul advantage, and the administration suppliers may briefly utilize other merchants' distributed computing administrations offered to manage the condition where there's a sudden development in asset necessity. Contrasts from customary supercomputer focuses, server farms work in distributed computing worldview for the most part depend on modest x86 servers facilitating at least one machine.

Another enormous contrast between supercomputer focuses and distributed computing server farms lies in the sort of assignments to be prepared: conventional super-PC focuses for the most part do some logical process, in the view where assets required by submitted undertakings could be known before errands are taken care of, while in distributed computing landscapes server farms handle different sorts of undertakings and the flexible normal for cloud computing confirms that both of errands' kind and size couldn't be known some time recently. Cloud computing is develop in a time where organize association gets to be helpful to get to. Accordingly, distributed computing depends on web to offer administration. Sorted by administration sort, a large portion of the administrations could be grouped into IaaS, PaaS or SaaS.

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

Not at all like conventional web based applications, through web, could clients get virtualized capacity, system, OS and various programming. Load balancing is critical in web based applications and in lattice figuring situation. We utilize Load balancing procedures to adjust the load among servers backend, where the heap is happened by frameworks' clients' prerequisites. That is, in online applications, fundamentally the point of the heap adjust is to appropriate web demands among applications servers or information servers, while in framework based supercomputer situation, we utilize stack adjusting procedures to make works submitted by end clients complete as snappy as could be expected under the circumstances. Load balancing is imperative for distributed computing worldview, as well. In any case, contrasted with matrix registering and the conventional system stack adjusting, there exist different requirements making Load balancing in cloud computing a test work. In Load balancing, what we really do is to discover a component by which undertakings are appropriated to process hub uniformly; keeping away from both problem area and hungry focuses exists. In this way it is a complex to make stacks as adjust as we anticipate. In this paper, we make some examination on Load balancing issue in a heterogeneous cloud computing environment. We concentrate on Load balancing's absolute entirety: its calculation. The accompanying parts are sorted out in a specific order: In "Related works" we give a few materials having done in load adjusting range. In "Proposed strategy" we give our enhanced calculation. In "Test Results" we utilize tests done in CloudSim to demonstrate our contemplations. At last we give the determination and observe into what's to come. Research on Heuristic Based Load Balancing Algorithms

2 Related Works:

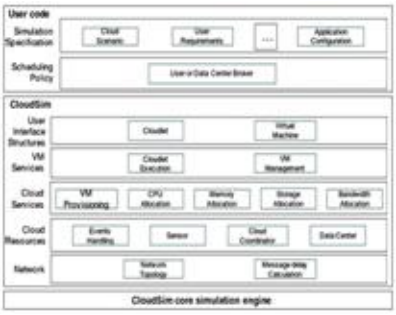
Distinctive needs and loads making diverse Load balancing technique works. Base on various types of load, different Load balancing model and techniques are created.

Azodolmolky S, Wieder P and Yahyapour R presented organize issues exists in IaaS and cloud league in [1]. They call attention to that the issue we confronted at this point in cloud computing system zone is the manner by which to build virtual systems. They professional represented a determination of SDN based applications. By uncouple control streams and information streams in SDN, advanced calculations and components could be created to address the issue of substantial transmission capacity, flexible gadget course of action, and live movement of VMs etc. In [2], Yeo S and Lee H S made scientific models about heterogeneous cloud computing environment's vitality utilization and reaction time. Yeo S and Lee H S dissected electronic vitality expended and execution qualities of cloud, finding that to accomplish better execution in heterogeneous based cloud frameworks, the most exceedingly terrible hub's reaction time ought to be close to 3 times of the brute ones. In [3], Doyle J, Shorten R and O'Mahony D dynamic system and server parts in cloud into a diagram.

They utilize carbon discharge, electronic utilization, together with administration achieve time as variables to settle on Load balancing choices. Doyle J, Shorten R and O'Mahony D utilize a straight weighted total approach on the three elements to acquire the edges' weight esteem in the chart got and they utilize Voronio realistic segment to figure out where to send the solicitations. In [4], Gulisano V, Jimenez-Peris R, Patino-Martinez M, et al. In 2012 proposed a flexible, extendable, register motor "StreamCloud" to manage stream information. Going for giving least assets to fulfill the given requests, they utilize a limit technique to offer assets while making load the adjust. In [5], Zhang Y and Zhou Y proposed another processing worldview, called straightforward registering. By making process "straightforward" to those end clients, work load is appropriated among backend servers and clients' terminals. In [6], Xu G, Pang J and Fu X proposed another heap adjusting model. By segment and switch components composed, they utilize a "brute replay" assessment standard to approach nash

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

equilibria. The player enlisted in the amusement is undertakings and server nodes. Load balancing could be performed in different levels, in a concentrated or decentralized frame. In [7], a definite depiction for Load balancing techniques in assignment level is given. In [8], a novel load adjusting system in VM level is given. By utilizing strategies proposed as a part of [8], a superior movement overhead at runtime is accomplished. In [9], Load balancing in cloud computing is done in system level. Load balancing could be static or element. As depicted in [7], static Load balancing techniques pick up data required before errands are executed, appropriate assignments in an altered way. In this way static Load balancing techniques couldn't accomplish a decent result. Dynamic load adjusting techniques screen framework's state and appropriate framework's heaps as per it. By doing this, dynamic load adjusting techniques could accomplish preferred execution over static ones. In any case, there's more overhead added



3 Proposed Method:

As shown in part 2, load balancing cloud be achieved in various level. But generally speaking, it could be categorized into 2 abstract levels: Job level and task level. Job is consisted of a sequence of tasks. In another word, job is a finite graph where tasks are the vertices and their relationships among are edges. In this paper we focus task level load balancing. Before we raise the method we based and the improvement we done, we Will give our problem definition first:

1. Task number: n, Task Vector: V_t = ð T_1, T_2, . . . T_nÞ^T , here T_i is a single task with resources needed
2. Server number: m, Server Vector: V_s = ð S_1, S_2, . . . S_mÞ, here we have Research on Heuristic Based Load Balancing Algorithms …

$S_i = \alpha$*MIPS + β*BW + γ*MEMORY where $\alpha + \beta + \gamma = 1$, MIPS is short for million instruction per second, BW is short for bandwidth

3.Vectors implying the status of each sever fL_i = ðLT_{1i}, LT_{2i}, . . . LT_{ni}Þ^T j0 < i ≤ mg where L_i is the load status and LT_{ji} ∈ f0, 1g, 0 < j ≤ n, implies whether T_j is assigned to S_i

4.Vectors implying the status of each task

$$fJ_i = \delta\, LT_{i1}, LT_{i2}, \ldots LT_{im}Þj0 < i \le n, \textstyle\sum_x^m{}_{=1} LT_{ix} = 1g$$

Assignment metric A = f L_1, L_2 . . . L_ng = f J_1, J_2, . . .

5. J_ng^T

6. = Value metric: $Value \begin{pmatrix} 0^a11 & \ldots & ^a1m\ 1 \\ \vdots & \ddots & \vdots \\ @^an1 & \cdots & ^anm\ A \end{pmatrix} .*A$ where a_{ij} $= T_i S_i = Time_i$ / $assig\ _n{}^+$

$Time_i$ _ $proc^+ Time_i queue$, here "." is the process of multiplying two metrics

element by element, at the same position.

7. Our object is to find an A to minimal make span time. Actually we make tradeoffs among Timei assign, Timei proc and Timei queue over all tasks. The difficulty lies in that if we do this in a greedy manner, we could not get a social optimal solution in the long run.

It is clearly that small assignment time may leads to big queue time and process time. In turn, small process time may obtained by a long queuing time and assignment time. Also, a small queue time cannot guarantee an ideal process time and assignment time. Basically, it is NP-hard to get a best server to send load. Thus a suboptimal method is acceptable in most cases. M. Mitzenmacher proposed a randomized load balancing method in [12]. He established a natural supermarket model, shown as Fig. 2,

where:

1. Customers arrive as a Poisson stream of rate λn, $\lambda < 1$.
2. The number of servers is n

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

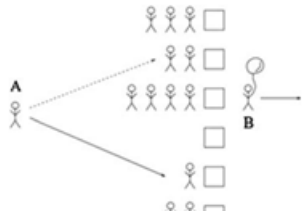


Fig. 2 Supermarket model

3.Each consumer chooses some constant d servers independently and uniformly from the n servers.
4.Customer waits for service at the one with fewest customers.
5.Service time is exponentially distributed with mean 1 Customers are serviced in a FIFO model
6.He proved that when d = 2, there's an exponential improvement in the expected time a customer spends in system over d = 1. And when d = 3, there's a constant factor better than d = 2.

When d = 1, this method is the same with the random method. Random method is susceptible because it always could not work well in small scale and in traditional scenarios the number of backend server points is not large. Thus we modify the step in 4 to this: customer waits in the queue where its estimated time spend in total is minimal among the queues chosen. Here we use the following formula (1) to calculate the estimated execution time Task Tk may spend on processing node Si:

$$m \le i \le k$$

E Timeik Þ = ∑ aij ð 1 Þ ð i = 1

Here a_{mj} is the first task's required time in node s_j
According to the mathematical description, it is clear that this formula is used as heuristic information to make decision. Load balancing could work in centralized or decentralized manner. On one hand, in centralized load balancing, it is easy to achieve the goals we set. But centralized load balancing has some shortcuts such as that it may become a bottleneck of the whole system. Although in a sense we can avoid this problem by adding more load balancing decision nodes, it could not deal with the situation well that the number of backend processing node is big. On the other hand, in decentralized load balancing, it is robust in handling with single point failure while elastic with system scale. But distributed load balancing is a little complex compared to centralized load balancing and the performance may not as good as centralized ones. The algorithm we based and improved could be used in both centralized and decentralized environment for the reason that it relies on little system information and little calculation process. Load balancing algorithms or mechanisms could be either static or dynamic, where static methods get the Information before execution and dynamic ones got information while executing. It is obvious that both the proposed method and improved one works in a dynamic manner.

4 Experiments and the Results:

FFD Algorithm:

One of the most natural heuristics for one dimensional bin packing is a greedy algorithm in which items are sorted by size in decreasing order and then items placed sequentially in the first bin that has sufficient capacity. This algorithm is often referred to as First Fit Decreasing (FFD). FFD is guaranteed to find an allocation with at most 11 9 OP T + 1 bins in the one dimensional case (c.f. [24]) and is known to be effective in practice. There isn't a unique obvious way of generalizing FFD for the multi-dimensional case. One has to decide how to assign a weight to a d-dimensional vector. In this work we've tested two natural options:

$$w(I) = \pi_{i \le d} I_i \quad \text{(FFDProd)} \ldots\ldots (1)$$

$$w(I) = \text{€}_{i \le d} a_i I_i \quad \text{(FFDSum)} \ldots\ldots (2)$$

Where the vector $a = a_1, \ldots a_d$ is a scaling vector of our choosing.

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

The vector a has two functions. First, it is needed to scale and normalize demand across dimensions (which not needed when taking the product of the demands). Secondly, it allows us to weight the demands according their importance, or their likelihood of actually being a bottleneck for placement. We test several ways of doing that.

Bad instances for FFD:

It is easy to see that any greedy algorithm has an approximation ratio of at most 2d. Since this is a weak guarantee it is important to identify inputs for which FFD performs particularly poor. In this section we identify such a family and argue that it is natural enough to motivate us to look for other algorithms. Consider the two-dimensional instance where half the items are of size (1 3 , 1 6), and the other half of size (1 6 , 1 3). In this case, the optimal solution puts four items per bin, two of each kind, while any FFD variant would place three items per bin. This example can be easily generalized to give a worse class of instances.

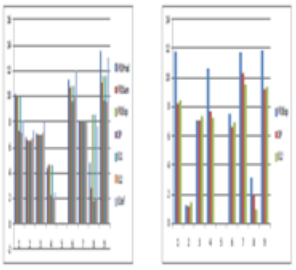


Fig. 1 Results when VM servers are descending while tasks are all the same

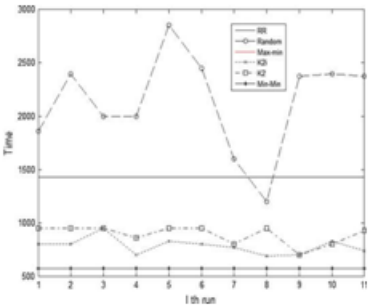


Fig. 2 Results when tasks and VM servers are same separately

Bad Example for our new heuristics We remark that the example of Theorem 1 can be modi_ed to create an instance where the new proposed heuristics are o_ by the same factor. Indeed suppose that the items of type Ti were to beeach split uniformly into (dk)i items of equal size, so that there are n(dk)I d items of size 1 (dk)i times the size of the type Ti items in Theorem 1. This instance forces our algorithms to make the same choices as FFD and results in the same approximation ratio. Note however that these new instances are signi_cantly less robust to perturbations. In the second experiment, we relax the condition where S_i, $i \leq m$ is in a descending manner while keeping tasks all the same. Better than that of the round robin, while the Random has little chance to be better than the Round Robin method. What's more, we can clearly get that K2i is more close to the best line we got in this figure.

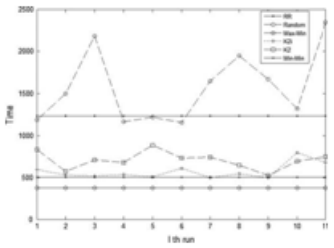


Fig. 3 Results when both VM servers and tasks are not in a sequential manner

In the third experiment, we make S_i, $i \leq m$ not in a sequential manner. In another word, we use the same configurations as in the second experiment. Following

5 Conclusions:

Load balancing procedures are fundamental for each disseminated framework, to a great degree in cloud computing situation. They expect to make Load balancing is to enhancing framework execution while ensure framework clients execution, stack adjusting could be performed in different levels by countless. In this paper, we concentrated on the calculation utilized as a part of Load balancing. In view of the hypothetically demonstrated the force of-two-decisions strategy, we proposed our change to make it more appropriate in distributed computing conditions.

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

Trials are done utilizing CloudSim. CloudSim is a controllable, rehashed capable test system for us to test and check our musings. Cloud computing is another created circulated and organizes based registering worldview. There's a great deal more work about load adjusting in distributed computing should be done later on. Future works may incorporate errands displaying, SLA ensured stack adjusting, Load balancing in server farm level, live VM migration consolidated Load balancing, etc.

References:

1. Azodolmolky S, Wieder P, Yahyapour R (2013) Cloud computing networking: challenges and opportunities for innovations. IEEE Commun Mag 51(7):59–63

2. Yeo S, Lee HHS (2011) Using mathematical modeling in provisioning a heterogeneous cloud computing environment. Computer 44(8):55–62

3. Doyle J, Shorten R, O'Mahony D (2013) Stratus: load balancing the cloud for carbon emissions control. IEEE Trans Cloud Comput 1(1):116–128

4. Gulisano V, Jimenez-Peris R, Patino-Martinez M et al (2012) Streamcloud: An elastic and scalable data streaming system. IEEE Trans Parallel Distrib Syst 23(12):2351–2365

5. Zhang Y, Zhou Y (2013) Transparent computing: spatio-temporal extension on von neumann architecture for cloud services. Tsinghua Sci Technol 18(1):10–21

6. Xu G, Pang J, Fu X (2013) A load balancing model based on cloud partitioning for the public cloud. Tsinghua Sci Technol 18(1):34–39

7. Mathew T, Sekaran KC, Jose J (2014) Study and analysis of various task scheduling algorithms in the cloud computing environment. In: International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 658–664

8. Internet technology and secured transactions (ICITST). IEEE, pp 178–184

9. Wang SC, Yan KQ, Liao WP et al (2010) Towards a load balancing in a three-level cloud computing network. In: 3rd IEEE international conference on computer science and information technology (ICCSIT), vol 1. IEEE, pp 108–113

10. Goyal A (2014) A study of load balancing in cloud computing using soft computing techniques. Int J Comput Appl 92(9):33–39

11. Calheiros RN, Ranjan R, Beloglazov A et al (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Experience 41(1):23–50.

12. Mitzenmacher M (2001) The power of two choices in randomized load balancing. IEEE Trans Parallel Distrib Syst 12(10):1094–1104.