Indian Journal of Science and Technology, Vol 9(47), DOI: 10.17485/ijst/2016/v9i47/106800, December 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Dynamic Resource Prediction and Allocation in Clouds using Pattern Matching

Harshpreet Singh[1*] and Rajneesh Randhawa[2]

[1]School of Computer Science and Engineering, Lovely Professional University, Phagwara - 144411, Punjab, India; harshpreet.17478@lpu.co.in

[2]Department of Computer Science, Punjabi University, Patiala - 147002, Punjab, India; drrajneeshrandhawa@gmail.com

Abstract

Objective: Cloud computing provides computational power as utility which can be acquired on demand. Cloud providers should have the ability to predict the future resource usage of an application. The prediction of resources aims to accommodate the dynamic resource requests generated by the applications and providing the resources when required. As the user's demand changes over time, it becomes hard to maintain uniformity in delivering the service. **Method**: This paper proposes a dynamic technique that predicts resource usage in advance using simple patter matching technique. The prediction is based on the initial usage of the resources and executing tasks. The resources are offered as packages where each package has a bundle of processing capability. Initially the algorithm monitors the current usage and starts predicting and allocating resources after the usage pattern is available. **Findings:** The experiment displays initial failure and gradually increases in hit as prediction is more accurate with more available patterns. This way, there source usage can be predicted and necessary resources can be made ready in advance so that the user can get necessary services on demand. The prediction accuracy of the algorithm is 97.08%. The delay in initializing the resources is minimized by 25.381%. **Application:** The resource usage can be predicted and necessary resources can be made ready in advance so that the user can get necessary services on demand. Resource prediction can improve the quality of service parameter and cloud providers can be aware of the changing demands of the applications.

Keywords: Cloud Computing, Patter Matching, Resource Allocation, Resource Prediction, Resource Monitoring

1. Introduction

A cloud is a connected network of servers and computational resources that is used to process and store user data over the internet. This set of resources can be visualized as a single network that is tightly coupled together and can stand alone and operate on its own. Cloud also reflects as utility computing as it provides various computer components (Processor, GPU Card, Motherboard, PSU etc.) assembled together as a machine and offered when required. Cloud components like servers, data storage units, processing machines, power units, which all are connected together in a single network offers huge computational power to process information[1].

The concept of cloud was introduced in similar to the legacy computer systems like distributed and grid computing. Due to the availability of cloud over the internet and ease of acquiring the power of resources without having they physically present over the user space offered great advantages. Cloud bundle can be acquired and accessed using a thin client and a complete computer system processing the jobs like a typical computer system is available. Cloud services are generally offered as pay-per-use services but could differ according to the vendor (monthly, yearly etc.)[2]. The users are able to choose the services according to the processing ability of the system or computational requirement of the application. Acquiring of the resources by user possess an overhead

*Author for correspondence

for the service provider to make the computing nodes available for the user if the demand increases[3].

The paper aims at reviewing and analyzing the problem of resource prediction for the executing applications. The hosted applications can demand for any number of resources with scaling processing demands from the cloud provider under a service level agreement known widely as Request-on-demand. Initially it takes some time to fulfill the demand in certain cloud models like the models based on Virtual Machines (VM). A VM takes minutes to start and work properly. The waiting time to initiate and get a fully working VM hinders the response time and throughput of the executing applications. There needs to be some technique where the resources can be assigned instantly without increasing the waiting time.

The main objective of the paper is to propose an algorithm which efficiently and dynamically predicts the usage of the application and make the cloud resources available to the user in advance. The technique aims to increase the availability of computational resources and response time of applications before the user application fails the execution.

To predict the resource usage and allocate the resources to the user with great efficiency.

- Make the technique of allocation dynamic so that the resources can be allocated or deal located during the runtime.
- Increase the transparency between the user and the cloud.

2. Literature Review

This section provides a brief review of work done in resource prediction and allocation in cloud computing.

In[4] authors form a cost-analysis matrix and a resource requirement matrix for each client that is connected to the cloud. By using the employed techniques, they tend to make accurate predictions about the future requirements of a client by considering past matrix data. The matrices are fed into an ANN to calculate future resource requirements. The main objective of this technique is to predict accurate future resource requirements based on past profiled data. This technique has used simpler AI techniques which would fail to predict in complex environments and applications which would have high resource requirements.

In[5] authors introduced new techniques of migrating entire code methods over to the cloud servers by taking several parameters in consideration. Instead of always connecting multiple computers to virtualize the environment, Think Air uses VM's running on single or multiple computers. There are other techniques too which migrates the code on to the cloud but Think Air proves to be ahead of them by using advanced profiling techniques and its own API which makes it really easy to handle offloading. It also exploits various parallelization techniques to execute applications in parts to speed up the entire run time of application. Think Air use VMs and each VM running on a computer can be in pause, running or stop state. Each VM takes ~300 ms to go from pause to start and ~32 secs to go from stop to start state. Stop to start time is totally reasonable for large applications but ~300 ms for smaller applications is not since an application can be easily assigned 6-7 VM's (parallel execution) which elevates the waiting to ~2 secs.

Authors in[6] focused on forming a totally heterogeneous cloud of mobile platforms. This cloud is totally dynamic in nature. The nodes (mobiles) are determined on fly as the server runs and a new node come in and exeunt the cloud. The main objective of this technique is to form a dynamic cloud of resources rather than a static cloud. Even though they have implemented their own algorithm, the technique still suffers from inaccurate calculations of mobile nodes and large down-times.

Authors in[7] introduces a resource allocation technique purely based on Java. It only aims the applications which has huge work head time and takes up so much energy and resources that it's not possible for them to run on a mobile device. So, this technique divides the application into multiple parts and offloads the heavier parts on to the server and leaves the thin parts (GUI) on client machine. The time each client has to wait before it starts the work is ~150 secs unless the target VM is already in operation (being shared). Hungarian method and entropy formula are used in[8] to make hybrid algorithm that efficiently lets a cloudlet to serve a client at a time rather than letting all connected clients to use it and overload it. In Hungarian method, it tries to choose what client to serve based on the requirements. In entropy formula, it tries to choose what client to serve based on amount of information it requires to exchange. Combining these two, the result is an efficient algorithm. It lets a cloudlet server one client

at a time. Decrease the probability of getting overloaded. Only serve the applications which are used frequently. It won't let multiple clients to share the cloudlet parallel.

Higher levels of resource allocation management are discussed in[9]. The concerned objective is how to maximize the revenue that the seller (company that provides services) earns. Resources are generally sold in bundles. The resources that can grouped together homogeneously are called Substitutable resources (multiple hard disks belong to same group) and resources that are grouped together to form a bundle are complimentary resources (hard disk space, bandwidth, CPU can be grouped in one bundle). These bundles aren't just given to the needy users. They are first auctioned. Each user has a user type to them that expresses the priority of the user. Higher is the type, more will be the chances of winning of that user. So, two factors are considered while choosing the winner, the price that the user offers and its type. Authors in[10] solely concentrates on energy preservation in mobile cloud computing by taking account of various techniques available. It does this by efficiently operating the available hardware since the hardware is what consumes the energy. It minimizes hardware use by making strict decisions.

New methods to cut down the energy needed to run the cloud servers and VM servers were proposed in[11] . They have developed two algorithms for this. In the first algorithm, they try to distribute the VMs in a manner to decrease power consumption. Traditionally, the load of VMs is distributed evenly to all the servers. So, all the servers are set to ON state no matter how much data they have to execute. This wastes a lot of power since all the servers and up and running. In this article, they have developed an algorithm which makes sure that the servers which are already running are given more loads to fully utilize them such that there is no need to run extra servers. A technique that can auto scale the resource allocation by predicting the future resource usage is proposed in[12]. The prediction is done by feeding the past usage data to the Bayesian network. The Bayesian network is first fed the training data to give it the idea of how the future patterns would be when it would be first put to use. After the network is trained, the actual usage data is fed into the network and future usage is predicted. Authors in[13] compared different prediction models considering the past data. The models discussed are the supervised learning models, meaning that they first need to be trained using training data so that the error in predicting can be minimized in the future. After the models has been trained using the training data, the model can be put to real use. The accuracy of each model depends on how the model was trained.

A technique is proposed in[14] that feeds on the past resource usage data to predict the future data. This is done by using Fuzzy Neutral Network (FNN). An FNN is an extension of Artificial Neural Network that can accurately predict future data. Fuzzy c-means taken together with the subtractive clustering algorithm is adopted to optimize the convergence features and learning speed. Authors in[15] used dynamic programming to make resource predictions for Hybrid IaaS Clouds using Workload-Tailored Elastic Compute Units. Elastic Compute Units (ECU) is a part of Amazon's web service EC2. This article modifies the implementation of ECU to make WECU which has 24% higher accuracy in predicting future resource usage. Taking the Cloud provider in consideration, the revenue generated by the provider must be maximized since this is one of the objectives of a Cloud provider. If the revenue/profit is good, the provider would be willing to give out even better services and support.

Techniques to maximize the revenue generated in different situations for demand of resources is considered and reviewed in[16]. The conditions can be when the resource demand is known in advance, when the resource demand is not known in advance and when the resource demand is in range. For the first case, a linear programming model which gives out maximized revenue scenario. By implementing the linear model, the Cloud provider can know in advance the resources it would need to put up for the client so that the revenue could be maximized and resource wastage could be minimized. For the second case where resource demand isn't known in advance, a 2 stage Stochastic programming model is used. In the first stage, a rough estimate of user's demand is calculated based on the partial information given by the user. In the second stage, accurate calculations are made to find out the demand. In the third case, the provider considers the worst case, that is, the demand is as uncertain as it could be or in a range. For this, the Robust Optimization Model is used to estimate the usage in advance so that the revenue can be maximized.

A technique based on Stochastic Integer Programming to solve the problem of Cloud provider revenue maximization problem under uncertainty[17]. It considers that the user's resource demand is uncertain initially. It assumes both on demand model and reservation model. In on-Demand model, the user can demand more or less

resources any time during the subscription. It follows the Pay-Per-Use model. On the other hand, reservation model allows the user to give all the usage details in advance with no change in the usage. So, the provider can plan the resource provision ahead of time. In addition to Stochastic Integer Programming, Hungarian method, Deterministic Equivalent Formulation (DEF), Benders Decomposition and Sample Average Approximation are also used. Hungarian method is used to optimally assign VM to physical machines. This is needed to reduce the resource consumption and increase the revenue generated by the Cloud provider. Then, the two staged stochastic programming is used to solve the complexity of optimization problems when the user's demand is uncertain. DEF is applied to distribute probabilities in different scenarios to reduce the demand cost. The Benders Decomposition algorithm divides the integer problem into simpler and smaller problems. This way, the complex and complicating variables and broken down into simpler variables. The SAA is also used to reduce the reservation cost and increase the revenue.

An online temporal data mining system called ASAP (A Self-Adaptive Prediction System for Instant Cloud Resource Demand Provisioning) is developed in[18]. This system self-adapts to varying user resource demands and trains itself to predict resource consumption for the future. Getting a VM ready may take tens of minute. This delay is not acceptable at all for sensitive applications. The way around this is to predict the usage in advance so that the VM can be made ready ahead of time. ASAP proposes a cost measurement called cloud prediction cost which guides the system to predict better models. If the resources are over provisioned, the cloud provider would face wastage of resources and hence, less revenue. If the resources are under provisioned, the user would have to face problems. So, an optimal measurement is needed which is calculated by this model. ASAP assumes that there are different types of VM that can be provisioned. All the different types have different amounts of resources such as internet Bandwidth, Hard Disk Space, etc. ASAP have 3 main modules, Raw Data Filtering and Aggregating Module, Model Generation Module and Demand Prediction Module. First off, a stream of information is recorded for all users. This stream contains information like VM Type, Request Start Time/End Time, Customer ID, etc. Only the required data is extracted by the Raw Data Filtering and Aggregating Module and a time series is generated. This time series data is fed to the Model Generation Module which further builds models of different predictors. A Predictor stores data feed in a model file for different VM types. There are N numbers of predictors, where N is the total number of VM types. Demand Prediction Module then reconstructs the predictors. It then utilizes a two-stage ensemble mechanism to predict future demand.

PDRS (Prediction based Dynamic Resource Scheduling) technique to auto scale the resources for cloud systems based on Virtualization is proposed in[19]. This system first uses a VM resource demand predictor based on ARIMA (Autoregressive Integrated Moving Average) to make resource allocation adaptive on each VM. Then prediction based dynamic resource scheduling algorithms are proposed to reduce the number of Physical machines used. This is achieved by optimally allocating the VMs to PMs such that resources are not wasted for each PM and they are fully utilized. This way the SLA (Service level agreement) is not breached. The main goal of this article is to reduce the number of Physical machines used and maximize the VMs on each PM. There are two major components, the Control pane and cluster. A cluster is a server which has several PMs available. There's one control pane for each cluster. Control pane handles all the tasks related to provision and prediction. Each cluster has a component called Node Agent (NA). NA collects all the resource usage statistics and executes all the commands received from the Control pane. Control pane has three components called Elastic Scheduler, Resource Demand Predictor and system monitor. System monitor in the central module that monitors resource usage defines the measurement period and collects the resource usage statistics from all the NA's. Resource Demand Predictor is an ARIMA based prediction model. It's based on time series data. It receives the data from system monitor and predicts the usage patterns for future. After the predictions have been done, Elastic Scheduler comes in. It takes the prediction data from RDA and determines the placement of VMs on PMs. It does this so that the PM is fully utilized with minimum resource wastage.

A technique called MAIU that offloads code to cloud to conserve battery and resources on mobiles is proposed in[20]. This is needed because the resource needy applications can drain the battery pretty fast. In the era of Smart phones, mobile phone companies are already having a hard time developing techniques to conserve battery. As in[5], the code is offloaded to the cloud which is then executed there and results are sent back to the mobile

application. This cuts down the processing needed to execute the code locally. Unlike other techniques, MAUI considers both client side and programmer side equally. In[5] a lot developer effort is needed in marking the code which can be offloaded. But MAUI makes it easy for the developer as it automatically identifies the heavier code modules from the application and offloads them when there's scarcity of resources. A technique proposed in[21] speeds up the application execution by assigning maximum number of VMs to one application while ensuring that QoS don't go down and the consumer stays satisfied. When a user request is received by the Cloud provider, the provider can either accept or reject the request. This decision is made by analyzing if there are enough VMs available. If we over allocate the VMs to applications, there won't be enough VMs left to allocate to other applications. So, a balance is needed which is achieved by[21]. It uses Semi-Markov Decision Process (SMDP) to achieve the objective. It maximizes the revenue generated, QoS and user satisfaction.

A methodology named PRESS (PRedictive Elastic reSource Scaling) is developed in[22] that extracts fine-grained patterns from resource demand data and adjust the future allocations automatically. PRESS is able to provision resources with less than 5% over estimation error and almost 0 under estimation error thus avoiding SLA breaches. For the workloads that has repeating patterns, PRESS makes a signature using signal processing techniques for the historic resource usage and uses it to predict future usage. This is done by using a Fast Fourier Transform (FFT) to calculate the frequently occurring patterns.

3. Proposed Methodology

The proposed technique allocates resources to the user, monitors the usage during regular intervals and adjusts the allocated resources in a manner such that they aren't either over provisioned or under provisioned. Cloud Provider (CP) is able to take the initial requirement of the user and initialize the resources. Requirements generated by the user are mapped with the number of VM's required which are then made available and are assigned on demand. A VM is an emulated machine which behaves just like the physical machine and is transparent to the user. One physical computer machine can create as many VMs as the hardware available. For instance, if there is a machine with 16 GB RAM, 3.4 GHz octa-core processor, 4 GB VRAM, a RAID setup of 4 1-TB HDDs, it can easily handle 5low end VMs.

A single user can be assigned multiple VMs based on the user's requirement. The problem arises when a new VM needs to be allocated instantly. If the user needs a new VM and requests one, the VM can take anything from one to four minutes to start and become fully operational. To overcome the delay in providing the resource, a technique based on patters as used in[23] is simplified and proposed. It starts when the user registers at the CP. The user provides the initial requirement and a max requirement of the application. Based on these parameters, the initial resources are assigned and the usage is monitored by the CP. After certain time unit, the monitored usage is kept as the Initial Resource Provision Array (RPA). This serves as a temporary parameter to predict the upcoming usage of user.

The user is offered resources as packages. A package has a one letter name to identify it. There are different packages that a user can request. New packages can be made depending on the demand of the users. A package has various computer components. For instance,

- A means a resource up to 256 MB RAM, up to 1 GB HDD space.
- B means a resource with capability between 256-512 MB RAM, 1 GB-2 GB HDD space.
- C means a resource with capability between 512 MB-1024 MB RAM, 2-3 GB HDD space.
- D means a resource with capability between 1024 MB-1536 MB RAM, 3-4 GB HDD space.

Four packages are defined above from which a user can demand any of the packages. Likewise, tens of different combinations can be formed. These elements are called RPEs (Resource Provision Elements). Each index of the RP array defines the timeline. For instance, each index refers to 30-minute interval. After the user has given an estimate of resources it needs and maximum resources it can reach, the initial filling of the RPA starts. There's a pre-determined parameter called "Initial RPA Size". The RPA is first filled till Initial RPA Size. During this initial filling, no prediction takes place. After the initial RPA has been filled, the prediction process starts.

As an example let us consider the following RP array sample demonstrated in Figure 1,

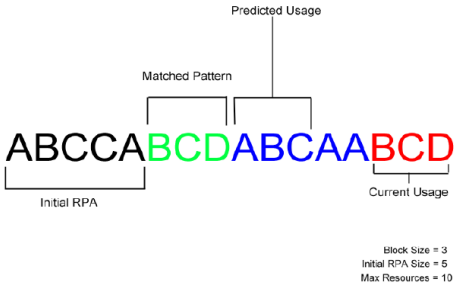

Figure 1. Prediction and monitoring.

ABCCABCDABCAABCD It means that the user used "A" amount of resources in first 15 minutes, "B" amount during the second interval, "C" amount in the third interval and so on. This way we keep a track of the user's usage. The first 5 characters tell us the Initial RPA. This means that user requested ABCCA during the Initial RPA time. During the monitoring of Initial RPA, the system doesn't predict anything. It just assigns whatever resources the user is requesting as the system is unable to predict the usage. After finding the Initial RPA, the system starts predicting the resources. For instance, in figure, the current usage is BCD. Depending on this usage, the system goes back and tries to find a matching pattern. It finds "BCD" pattern at index 6. Depending on the block size, the system picks up the next 3 usage patterns "ABC". Out of "ABC", C is the biggest resource package. So, the system adds "C" resources to the resource pool so that the user can be assigned resources.

Depending on the total resources in the resource pool, there can be 2 scenarios as given in Figure 2

- Resource Pool has enough resources to assign incoming user request.
- Resource Pool doesn't have enough resources to assign to user request.

In scenario 1, the parameter "Hit" gets incremented. This means that the system was successful in fulfilling the user current request depending on the past predictions. On the other hand, if it was scenario 2, parameter "Miss" would have been incremented instead. This means that the system failed in correctly predicting the resources in the past. The Ratio of Hit and Miss tells the performance of the system.

The following algorithm defines all the brief steps as explained in the above example.

(Algorithm1)

Input: Resources

Output: Predicted Resource List

Allocate the resources and initialize the RP array.

Fill the RPA till the Initial RPA Size.

Repeat the steps a, b, c and d until the bill cycle is complete.

Monitor the usage.

Predict the next usage (Algorithm 2).

Keep the resources ready to be used by user.

Assign the resources.

End

In the above algorithm, before the RP array can be used to predict usage, it needs to be filled with enough data that can be referred for prediction. For instance, first 10 RPEs are filled as the user is using the resources. After these 10 RPEs has been defined and filled, the system can go ahead and predict the usage.

The prediction is done using pattern matching algorithm. Since our RP array has been transformed into a basic array of strings and matching algorithms can be applied to search for patterns.

(Algorithm 2)

Input: T is the text array,

Output: P is the pattern to be searched

n ← length[T]

m ← length[P]

For s = 0 to n-m,

do

If P[1 → m] = T[s+1 → s+m]

Print "Found" and exit

End

The user resource usage can be searched and predicted using this technique. The size of pattern is fixed in the beginning as 5 so that continuous usage of 5 fixed intervals is being searched. When an exact pattern is found in the RP array, the next usage of 2 or 3 hours is picked up and resources are put to work accordingly.

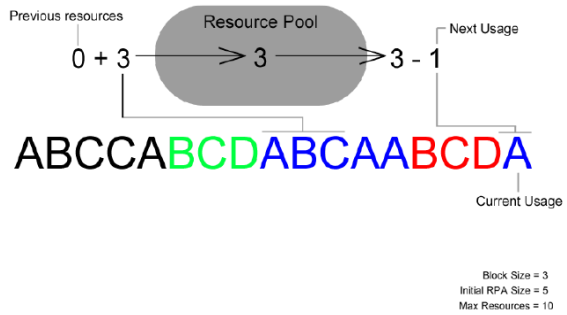

Figure 2. Current usage and predicting resource usage.

4. Results

This section of the paper presents the implementation details of the proposed load predication algorithm. For evaluating the proposed approach, a cloud environment is considered which executes compute intensive and data intensive tasks. The user submits different workflows consisting of dependent jobs with varied resource requirements. The workflows considered consist of 100 tasks to 400 tasks, where tasks varied from 10^4 to 10^6 million instructions with data movement from 50 MB to 1000 MB. Each job is allocated individual resources and is not executed twice on different resources. A total of 10 RPE's are considered for executing jobs at parallel coming from different workflows.

The user submits the request for the minimum resource package and maximum resource package which can be acquired by the workflows. After getting the resource requests, the workflows are submitted for execution and initial filling of the RPA starts. In the beginning the number of misses' increase as there is no prediction at this time interval. After passing the initial RPA the number of hits gradually increases and the number of miss decrease. The ratio of hit over miss is computed and if the ratio is less than three the performance is considered as bad and if ratio is greater than or equal to three is considered as good.

Figure 3 shows the resource utilization of the workflows submitted for execution. The figure presents the comparison of task demands for resources without using the prediction algorithm and after using the prediction algorithm. The prediction of resources using the pattern based prediction is 97.08% more accurate. Figure 4 provides the comparison of execution time of tasks before and after prediction. The execution time of tasks is computed by adding the initialization time required for starting the resource. Since the proposed algorithm predicts the usage and initialize the resources the before the request is generated and the overall execution time is decreased. The delay of starting the VM and providing it to the user application is reduced at an average of 25.381%.

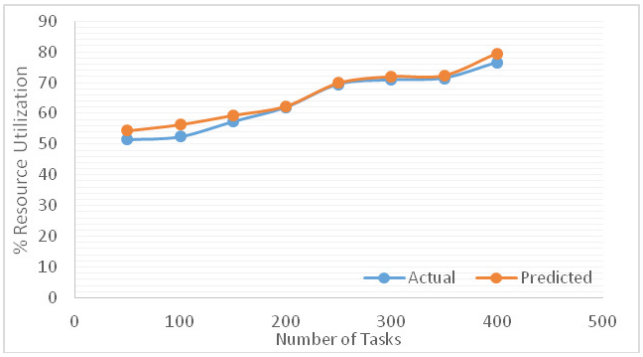

Figure 3. Actual and predicted resource utilization.

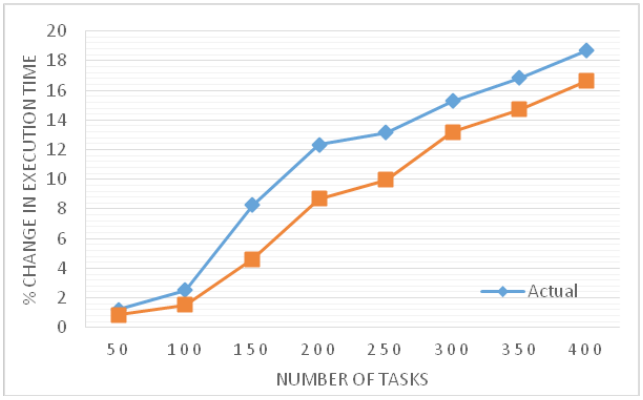

Figure 4. Percentage change in execution time.

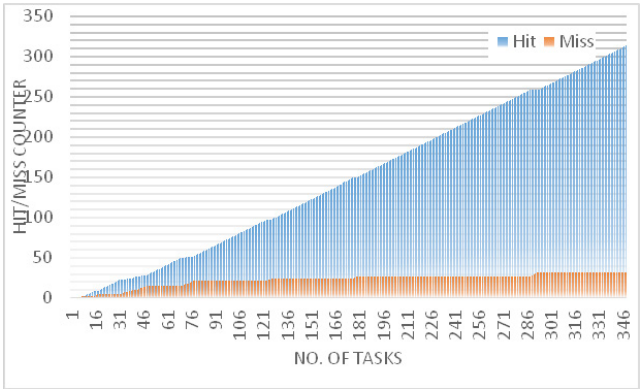

Figure 5. Hits and Miss.

shows comparison between hit and miss for different tasks. This shows an abstraction of overall performance of the system. compares Resource Demand with resource pool. The depiction of this graph is to show how the

resource pool has enough resources to fulfill the demand of user. Most of the times, the resources in resource pool are more than the demand of the user. The resources are at abundance as the demand of user is unpredictable and high. This means that the user will mostly be assigned more resources instantly without having to face any delay.

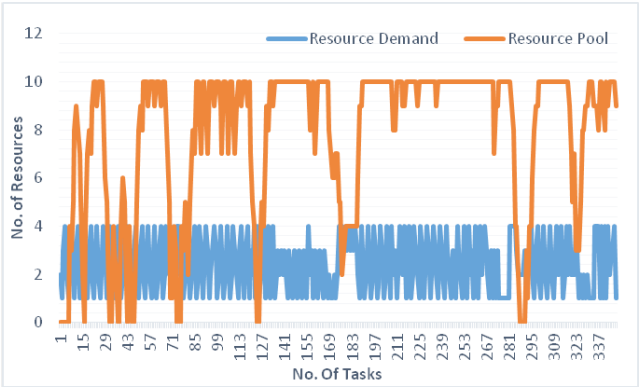

Figure 6. Resource demand and resource pool for tasks.

5. Conclusion and Future Work

Amid the changing user and application resource requirement, the resource usage prediction helps in designing the system which can manage workload and plan the capacity of the provider at a given time interval. The proposed prediction algorithm aims in proving the resources as bundle and predicts the bundle to be used in the near future. Such predictions help the provider to get the resource initiated before the request is generated. The predictions are not accurate enough to predict the exact bundle required, for which the algorithm helps to initiate more than one bundle which could be requested by the user. The Cloud provider is able to reduce the expenditure and increase the transparency by predicting the usage of the resources. The delays that the user had to face earlier during resource allocation are reduced to a bare minimum and the Quality of Service parameters are increased. The experimental results show the increased utilization of the resources. Two main parameters, Hit and Miss shows the initial running of the proposed algorithm. The objective is to keep Hit as high as possible and Miss as low as possible. The patter matching can be further enhanced in order to predict the usage in advance. Energy consumption can also be reduced by further checking both the static and dynamic energy consumption while executing the workflows.

6. References

1. Singh H, Randhawa R. CPSEL: Cloud provider selection framework for ranking and selection of cloud provider. International Journal of Applied Enggineering and Research. 2015; 10:18787–810.
2. Singh H, Randhawa R. Evaluation framework for selection and ranking of cloud providers. International Journal of Advancement Computing and Technology. 2015; 7:31–7.
3. Vaithiyanathan V, Lavanya M. Load prediction algorithm for dynamic resource allocation. Indian J Sci Technol. 2015; 8:6–11.
4. Sood SK, Sandhu R. Matrix based proactive resource provisioning in mobile cloud environment. Simul Model Pract Theory. 2015; 50:83–95.
5. Kosta S, Aucinas A, Hui P, et al. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. IEEE Proceedings of INFOCOM; 2012. p. 945–53.
6. Khalifa A, Eltoweissy M. Collaborative autonomic resource management system for mobile cloud computing.
7. Ferber M, Rauber T, Torres MHC, et al. Resource allocation for cloud-assisted mobile applications. IEEE 5th International Conference on Cloud Computing (CLOUD); 2012. p. 400–7.
8. Praveena A, Poonguzhali E. Resource allocation and storage using hungarian method in mobile cloud computing. Int J Adv Res. 2013; 3:218–22.
9. Zhang Y, Niyato D, Wang P. An auction mechanism for resource allocation in mobile cloud computing systems. International Conference on Wireless Algorithms, Systems and Applications. 2013. p. 76–87.
10. Miettinen AP, Nurminen JK. Energy efficiency of mobile clients in cloud computing. Hot Cloud. 2010; 10:4.
11. Younge AJ, Von Laszewski G, Wang L, et al. Efficient resource management for cloud computing environments. International proceedings on Green Computing Conference; 2010. p. 357–64.
12. Bashar A. Autonomic scaling of cloud computing resources using BN-based prediction models. IEEE 2nd International Conference on Cloud Networking (CloudNet); 2013. p. 200–4.
13. Amooee G, Minaei-Bidgoli B, Bagheri-Dehnavi M. A comparison between data mining prediction algorithms for fault detection (Case study: Ahanpishegan co.). International Journal of Computer Science Issues. 2011; 8(6):425-31.
14. Chen Z, Zhu Y, Di Y, et al. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. Computer Intell Neuroscience. 2015; 17.

15. Imai S, Chestna T, Varela CA. Accurate resource prediction for hybrid IaaS clouds using workload-tailored elastic compute units. Proceedings of IEEE/ACM 6th International Conference on Utility and Cloud Computing; 2013. p. 171–8.
16. Prabhu L, Vasudeva Rao J. Resource allocation in mobile cloud computing using optimization techniques. International Journal Wireless Communication Network Technology. 2015; 4:30–6.
17. Babu MC, Umamageswari S. OCRP in dynamic resource allocation using virtual machines for cloud computing environment. International Journal of Computer Science and Information Technology. 2014; 5:1002–7.
18. Jiang Y, Perng C, Li T, et al. Asap: A self-adaptive prediction system for instant cloud resource demand provisioning. IEEE Proceedings of 11th International Conference on Data Mining; 2011. p. 1104–9.
19. Huang Q, Shuang K, Xu P, et al. Prediction-based dynamic resource scheduling for virtualized cloud systems. Journal of Networks. 2014; 9:375–83.
20. Cuervo E, Balasubramanian A, Cho D, et al. MAUI: Making smartphones last longer with code offload. Proceedings of the 8th International conference on Mobile systems, applications and services; 2010. p. 49–62.
21. Liang H, Xing T, Cai LX, et al. Adaptive computing resource allocation for mobile cloud computing. Int J Distrib Sens Networks. 2013.
22. Gong Z, Gu X, Wilkes J. Press: Predictive elastic resource scaling for cloud systems. Proceedings of International Conference on Network and Service Management; 2010. p. 9–16.
23. Zhang L, Zhang Y, Jamshidi P, et al. Service workload patterns for Qos-driven cloud resource management. Journal of Cloud Computing. 2015; 4:1–21.