International Journal of Computational Intelligence Systems, Vol. 10 (2017) 363–374

Speculation based Decision Support System for Efficient Resource Provisioning in Cloud Data Center

R.Leena Sri[1], Dr.N.Balaji[2]

[1]Information Technology,Thiagarajar College of Engineering
Madurai,Tamilnadu,India
rlsit@tce.edu

[2]Head ,Information Technology,K.L.N.College of Engineering
Madurai,Tamilnadu,India
balaji_jet@yahoo.com

Received 22 September 2015

Accepted 3 November 2016

Abstract

Cloud Computing is a utility model that offers everything as a service and supports dynamical resource provisioning and auto-scaling in datacenter. Cloud datacenter must envision resource allocation and reallocation to meet out the unpredictable user demand that touted gains in the model. This impact a need of adaptive and automated provisioning of resources aligned with clients SLA amidst the time variant environment in cloud. The robustness of dynamic resource provisioning is based on quick multiplexing virtual resources into physical servers. In this paper we put forward a prediction based automated resource allocation model induced by speculation mechanism. Our model guarantees dodging over/under utilization of resources and minimizes the cost economically without compromising Quality of Service. We regain the speculation concept that uses a past resource access pattern to predict future possible resource access. We introduce the confidence estimation factor to address the historical variability of the current pattern to improve the prediction accuracy. Experimental results show that our proposed model offer more adaptive resource provisioning as compared to heuristic and other machine learning algorithms.

Keywords: Decision support system, Dynamic Resource Provision, Framework Design and Analysis, speculation.

1. Introduction

Cloud computing made the computing services as a utility model. According to Right Scale 2015 State of the cloud report, 93% of users are using cloud services out of 88% using public and 63% using private cloud and this percentage augments year by year. Cloud attracts many users for its dynamic resource provisioning and auto scaling behavior. Cloud users have no knowledge or control over how their demands get processed in cloud infrastructure. The Resource Allocation System (RAS) mechanism in the cloud infrastructure exposes the finite resources in the data center as unlimited resources for both the cloud consumer and developer.[15] The virtual machines (instances) are multiplexed with physical servers based on user demands. Early studies show that the time taken for instantiating VM's into the particular host is tentatively 5-15 minutes termed as instance setup time.[29] The instance set up time must be optimized and speed-up to improve the resource provisioning strategies. Many prediction and optimization strategies are used in this case such as simple Heuristics with thresholds, Neural Networks, Linear regression, Artificial Intelligence algorithms like ant colony, game theory. The total phases involved in automatic resource provisioning are resource modeling, offering, monitoring, and selection.[14]

Today most cloud datacenters faces the challenges of huge power consumption and carbon emission during resource management. Initiating efficient admission control, capacity allocation, load balancing, energy optimization and Quality of Service (QoS) guarantees policies would probably reduce over-provisioning or under-provisioning of resources to the server and optimizes energy efficiency. See Ref. 6, 8, 10, 17 for more about resource management policies. Hence there is a need of framework that has the mechanism to implement the above said resource management policies to reduce over/under provisioning of resources irrespective of their need as per Ref. 4, 7, 9, 19. With this as an objective our Speculative Resource Provisioning (SRP) framework mechanizes the resource provisioning by predicting effective resources for the client/developer based on its past utilization. We use speculation induced model named Speculative Resource Provisioning (SRP) for adaptive resource provisioning and to improve the instance setup time which probably attract large cloud users. Speculation is used to predict the exact resources for an application and whose accuracy provides solutions for under or over utilization of the resource. In particular, we use patterns from past resource accesses for a service to speculate the tentative resources needed to execute the application: the prior knowledge of the characteristics of the host, from past resource provision, prevents over- or under-utilization. Thus, the speculation ensures Quality of Service (QoS), provider reliability, consumers Service Level Agreement (SLA) and enhances user trust, to make them feel they pay only for what they use.

In our framework we use dataset generated from Web Services and an Application Service Benchmark tool TPC-App[27] for testing and training our speculative resource provisioning model. The Accuracy of the prediction model is validated by confidence estimation methodology which has proven success in Branch Prediction in Multi scalar architecture.

The rest of this paper is organized as follows: Section 2 explains Literature Review, Section 3 describes our algorithm Speculation Resource Provisioning and Section 4 presents implementation, Experimental Setup, and Algorithm efficiency. Section 5 shows performance results, and Section 6 concludes the paper.

2. Literature Review

In this section the challenges of existing cloud resource provisioning and need for framework to address the challenges are given in brief.

2.1. Challenges in Cloud Resource Provisioning

There are many algorithms supported in different resource allocation mechanisms[26] like the Datacenter control algorithm (DCA), energy management structural framework, RC2 algorithm etc., Datacenter control algorithm (DCA), proposes VM queuing to the same servers through online control decision, based on application workloads. In DCA algorithm there are three stages Admission control, routing and resource allocation. The user requests are validated through Admission control and routed to the server which has the minimum queue of VM. Hence time taken for allocating resources to the VM is much reduced. The algorithm is used to implement energy efficient resource allocation in virtualized datacenter.[31]

The structural framework for energy management virtual datacenter made out of dispatcher, Local and Global Manager.[19] In this framework developer's request VMs and such requests are handled by Dispatcher. The dispatcher distributes the requests among the VM. The VM usage, server states are managed by Local Manager and monitored information is sent to the Global Manager. The Global manager uses ranking algorithm to prioritize the VM which is allocated to the server. The framework focuses energy management as well as set of the idle server is switched off to optimize the energy consumption.

In Ref.2. Resource allocation algorithm is proposed with the motivation of resource overload avoidance and green computing. The authors designed automated resource allocation with load prediction algorithm. Skewness concept was introduced to measure the server unevenness for predicting the server load. In Ref.3, the authors used heuristic based gossip protocol P* to address dynamic resource management in the large cloud datacenter. The protocol ensures the fair resource allocation, adaptive towards load and resource scalability.

All such algorithms have the criteria to ensure QoS, availability and responsiveness for the client's SLA which was further addressed in Ref.15, 16, 26. Cloud service providers permit their clients to indicate the need for resources like CPU speed, memory size, network bandwidth, and storage capacity etc. for capacity planning in the cloud. The service requirement of the user may vary over time; hence the monitoring of the current service request and change of future requests must be supported. Therefore, Heuristic and Artificial Intelligence based resource selection and optimization techniques are induced into a current Resource Allocation System (RAS). At times, clients reserve more resources than needed which results in

surplus resource left unused. Cloud service provider offers these resources using different pricing models such as Pay-per-use model, Spot instance model, reserved instance pricing model as discussed in Ref.7, 9. Pay-per-use model is used under unpredictable workload characteristics. Spot instance model is used for flexible workload completion scenario. Reserved instance pricing model is used for long haul duty.

In practice clients may not predict and reserve random amount of resources for their applications which may results in resource migration as discussed in Ref. 1, 2, 11. The issues in resource migration are significant downtime and service disruptions, which lead to degradation in cloud performance.[24] At the point when the number of live migration increases, the bandwidth consumption will increase and raise issues of violating the predefined SLA such as response time, throughput and latency.[25] The pay-as-you-go model and dynamic resource provisioning features of cloud reduces the static provisioning overhead of over provisional or under provisional of infrastructural resources. There emerges an exploration difficulty to devise an intelligent and adaptive Resource Allocation System (RAS) which focus on lower expense rate and higher execution slots. The obligation of the RAS is to intercede between the resource provider and client and to withstand the proactive scaling in the cloud as discussed in Ref. 3, 5, 6, 12, 14.

2.2. Proposed Framework

Many high performance systems for example multiprocessors, file systems etc. undergo speculation approach for quick response. The speculation is the process that executes the operation before it is considered as vital or not. The speculation mechanism of program execution happens in two ways: 1. Predicting the subordinate processes, when required, during critical path processing and verifying the predicted subordinate process in parallel, 2. Predicting which subordinate processes should proceed to reach an outcome. If it turns out the subordinate process is not needed and any changes made by it gets reverted and the results are ignored. The objective is to provide more concurrency if more esources are available. The usage of speculation approach is based as illustrated in Fig.1. In our Speculative Resource Provisioning System (SRP), the speculation module is executed alongside the resource identification module.

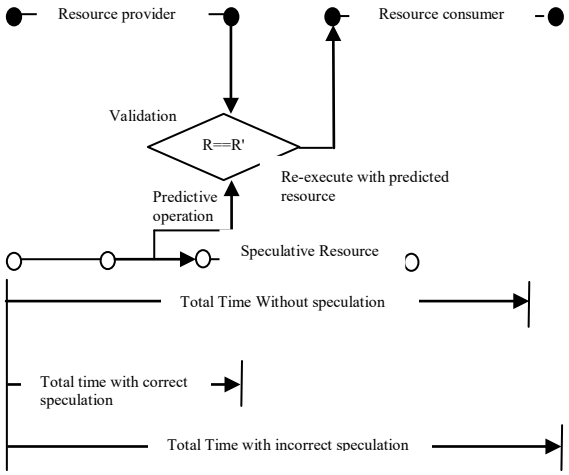

Fig.1. Resource Speculation

Our system attempts to maximize the application performance with an optimal cost for the service provider. We compared our result with other machine learning prediction based algorithm such as Neural Network (NN) and Linear Regression (LR) proposed for adaptive resource provisioning.[29]

Our model is implemented in a hierarchical architectural based cloud framework with the components of centralized cloud controller and instance controller. The resource submission and resource selection supporting modules are initiated by Cloud controller, and resource control and monitoring modules are incorporated in the instance controller. The instance controller intermittently screens the resources, for example, Resource density, Load factor and the current quantitative units are provided to the cloud controller when the new service request arrives. The Resource Density (RD) parameter represents the intensity of computing power per communication bandwidth. The lower the value indicates the more bandwidth resultant of high communication.

$$RD_i = \frac{\sum_{j=1}^{n} Processor\ Speed_j}{\sum_{j=1}^{n} Communication\ Bandwidth_j} \quad (1)$$

RD_i denotes the Resource Density of the Host. It is the aggregation of set of virtual resources $j=(1...n)$, the other parameter Load Factor (LF) provides the overall load at some instant, this is essential to take care the computation component. The resources that have a high computation compared to communication would prefer LF unit than RD

$$LF = \sum_{i=1}^{n} \%\, Utilization\ of\ Processor_i \quad (2)$$

ET denotes the execution time to run 'N' jobs on p_n processors. $t_1(N)$ is the sequential time, $t_{comm}(N)$ is the communication time. The communication overhead may depend on factors such as network latencies, message delays and communication volumes.

$$ET = \frac{t_1(N)}{p_n} + t_{comm}(N) \quad (3)$$

EC_i is the Execution cost of the application runs in the host i, ϕ is the unit of cost determined by resource provider. The cost may comprise many resource components like CPU utilization rate, memory usage, network bandwidth consumption and disk accesses.

$$EC_i = ET_i * \phi \quad (4)$$

The client sends its demand to the cloud controller alongside the SLA. In view of client SLA, the cloud controller alleviates computational risk management and guarantees that the Client's SLA never violates the cloud infrastructure QoS; on the off chance that it is succeeding, then the request is further processed by submitting it to the instance controller. When a request is submitted to the instance controller, the Speculative Resource Provisioning (SRP) module is executed in parallel to know the prior resource requirement for the request.

3. Speculation Resource Provisioning (SRP) Mechanism

Branch prediction in computer architecture spotlights on enhancing the execution of pipelined microprocessors by precisely foreseeing early regardless of whether an adjustment in control flow will happen. Our prediction mechanism is logically inspired with hardware supported Two-level Adaptive Training Branch Prediction that supports high performance processors by minimizing the penalty associated with mispredicted branches on the premise of data gathered at run-time as indicated in Fig 2.

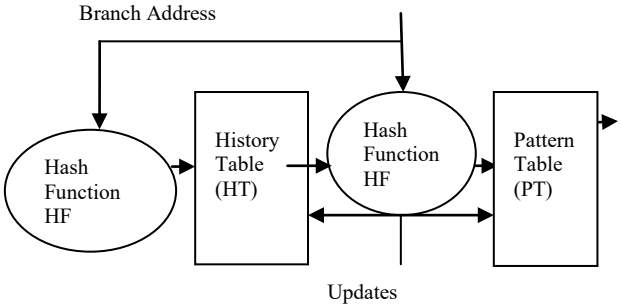

Fig.2. Two level adaptive prediction mechanism

Predictions are generated by pattern-based predictors which use patterns of past resource provisioning to figure future resource allocation. The predictor helps dynamic resource provisioning by in prior instantiating demanded virtual machines, dynamic resource provisioning, workload administration, framework estimation, scope, organization and element guideline era in the cloud which is discussed in Ref. 8, 16, 17, 18, 23. The client who seeks service relies on cloud brokering services to lease resources for their application. There are prediction based resource measurement and provisioning strategies available which uses many machine learning algorithms such as Neural Network (NN) and Linear Regression (LR). The machine learning algorithms are used for designing adaptive resource management to satisfy the resource demands.[29] Our work differs in case of predicting the performance of the resource in prior even before execution by considering its past performance. This procedure is roused by the genuineness of appointing right work to the right individual for showing signs of improvement in result. The cloud controller gets the user's jobs along with the SLA. The SLA determines the qualities of the application, the minimum and maximum requirements of the computational resources and the due date. Our predictor in view on Fig 3 keeps up a history table and pattern table for each and every service requisition along SLAs. The history table is a sequence of 'n' history entries that records the most recent service request along with SLA. The pattern table is a record of all the observed sequences of requests alongside the predicted patterns as given in Fig.3

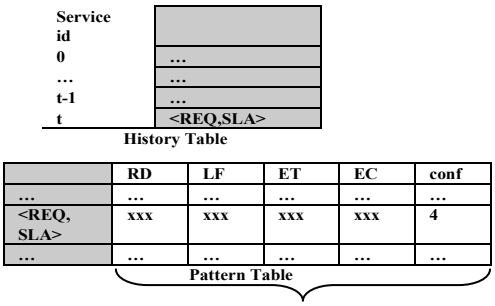

Fig.3. Two level predictor

When the cloud controller receives the service request (S_{req}) it validates the request whether it violates the service provider QOS and then it stores in the history table before searching the resource availability. At whatever point entries in the history table (HT) are made, it is then searched in the pattern table (PT) whose first entries matches the current contents of the history table using the KMP algorithm discussed later. The pattern table consists of application characteristics along with specifications like Resource Density (RD), Load Factor (LF), Execution Time (ET) and Execution Cost (EC). The pattern values are the exact resource requirement for the user's job specification based on past computation. The dynamic resource scaling that provides inappropriate patterns at the situation of Slashdot effect or sudden traffic surge must be seriously considered during pattern table entries. Meantime multi tenancy nature of the cloud uses a statistical standard deviation approach for normalized patterns, especially for performance parameter such as resource density and load factor.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \qquad (5)$$

N is the total number of patterns, μ is the mean value of the patterns, x_i is the set of patterns.

Assumptions are made as clients operating within the same application domain and also the historical knowledge of the pattern matching are taken for predicting resource usage. Our pattern matching algorithmic rule uses the fundamental of Knuth-Morris-Pratt (KMP) pattern matching algorithm. The KMP matching algorithm rule uses degenerating property (pattern having same sub-patterns superficial over once among the pattern) of the pattern and enhances the worst case complexity to O (n). The fundamental plan behind KMP's algorithmic rule is that it effortlessly predicts the sub patterns shown within the string even if crisscross happens amid string comparison. We tend to take this information to withdraw from coordinating the characters that we all know can in any case match. The KMP algorithmic rule ought to be somewhat adjusted for approximate matching. The two types of guess matching errors are introduced: instant error and cumulative error for guaranteeing scale independence amid pattern comparison.[28]

In order to discover the distinction between the current and the pattern table entry it would be standardized to a floating point interval [0 to 1] rather the huge whole numbers. The distance is calculated between the patterns by subtracting the values and higher weight is assigned for the smaller distinction pattern. If there is no match between the service request pattern and patterns within the pattern table, the service pattern is recorded as a new entry within the pattern table. The cloud controller undergoes a bin-packing optimization scheduling algorithm during the resource discovery and therefore the result of the resource demand from the speculation module is taken for consideration.

3.1. How Speculation works?

Our speculative based Resource Allocation System uses two operational messages named SPEC (speculation) and REQ (resource request). Whenever a cloud controller receives the resource requisition it also receives the resource list with current resource parameters like RD, LF from instance controller. In meantime it performs traditional Resource identification using REQ operation to spot the available resources. Meanwhile, once a pattern of resource provisioning for the service request is acquired from the pattern table as discussed in section 3, the controller can use this information to predict future provision for the similar request and speculatively issue SPEC operation of the potential resource. The user request would be handled by the SPEC initiated resource. The execution time and the cost will also be judged in advance via pattern table entries. If it gets slightly deviated the best values will be recorded along with Execution Time (ET) and Execution Cost (EC) in the pattern table. Conceptually, the controller should also attempt to issue SPEC whenever a resource scaled up or down.

When a server receives a SPEC request, it first checks the timestamp to ensure the instantaneous virtual resources. If this check fails, it generates a response message that sets the confidence level for the pattern that triggered the SPEC request to the minimum value to inhibit future SPEC requests until the resource provision pattern is re-established. If the resource receives the REQ message followed by SPEC message, then the confidence level (conf) gets incremented ensuring high confidence in envisioning future prediction. However, our algorithm still avoids some portion of the resource allocation latency while issuing server state updates based on SPEC request and allowing the controller to resume processing. Meanwhile, when the response to the resource request arrives at the remote node, it is discarded.

When the resource is scale up or down, the servers that are encountering load variations issue SPEC while waiting for the remaining similar servers to adopt the load variation. This methodology lessens the overhead of speculative processing by overlapping it with the latency of the newly configured cloud resources. In any case, once all servers configured, any remaining speculative processing servers are permitted to depart from the adjustment of load variation and resource density. Speculative Processing ensures non-expansion of idleness because of workload varieties. Controller executes a speculative action, updating the load variation in the predicted server using a SPEC request message. Notwithstanding the resource provisioning parameters, the SPEC request contains the current vector timestamp. The controller records speculative activities in the history table as though they have triggered by an actual resource request by it. Neglecting to do so could lead our predictor to observe false patterns. For example, if a SPEC request not recorded in the history table, but rather succeeded in staying away from resource provision latency, the predictor would record this as a pattern. At this stage, when the node just had detached during scale down won't consider for resource prediction in future iterations. In the event of immense load and resource density deviation after scale up or down, the server is idle and is willing to designate resource to the recently arrived request. This recently changed state of the server is upgraded in the controller part. Finally, at the time the SPEC request arrives, the accepting server may have officially issued its redesigned state to the controller in lieu of giving resource for the SPEC request. In this situation, the predictor has effectively predicted the server; however it is not early enough to avoid a load and resource conflict.

3.2. Confidence Estimation

Confidence estimation is a process for assessing the performance of the prediction. The confidence level (labeled conf in fig. 3) is to distinguish the patterns having a low probability of predicting future accesses. The confidence level is maintained within a small range, and increments and decrements beyond this range are simply reset as high or low confidence value respectively. For every prediction, confidence estimator assigns a counter value to track "high confidence C_{HC}" or "low confidence C_{LC}" to the predictor. The high confidence parameter gets incremented when our speculation works prior enough to predict the exact resources for the request i.e., SPEC message followed by REQ message. The lower confidence estimates the delay in predicting the resource i.e., REQ message followed by SPEC message. The prediction hit (pred_hit) and prediction miss (pred_miss) are calculated for evaluating accuracy during prediction.

$$pred_{hit} = \frac{C_{HC}}{(C_{HC} + C_{LC})} \quad (6)$$

$$pred_{miss} = 1 - pred_{hit} \quad (7)$$

The percentage of the prediction hit ratio will increase the QOS such as responsiveness and throughput.

3. Implementation

We have conducted our experiment of speculative resource provisioning in the cloud by implementing all the methodologies as mentioned in section 3. Our experiment is arranged in two stages. Stage I - constructing a pattern table, and Stage II - contrasting our outcome and other machine learning algorithm, such as Neural Network and Linear Regression. First, we have generated the historical data by running a standard Web service and an Application service benchmark, such as TPC-App.[27]

Stage I: Training and Pattern table construction

We have chosen TPC-App benchmark tool which simulates the activities of B2B transaction application server 24 x 7 environments for low level characterization of resource usage. The environment is chosen to stress application server and observe the service requisition rate and resource utilization in a real-time scenario. The TPC-App generated transaction, interactive command, processes and threads, HTTP requests and the service usage are the major component of our workload model. The intensity of workload is based on random arrival rate, number of clients and number of processes or threads in execution. Those are the describing factors of resource consumption during the workload such as processor time, memory usage, I/O time, disk operation etc.CPU time, I/O time are the pair which characterize the execution of request at the server and the sample resource usage patterns information's are shown in Table 1. The components are summarized based on the parameter as suggested in the Table.1 and average parameter values of all components are shown. There are two performance metrics used in TPC-App benchmarking, Web Service Interaction per Second (SIPS) and Total SIPS. We pick the resource utilization of an average day from TPC-App session undergoing the business transaction classified in 8 sessions (small-medium (L), medium-large (M), large (H), large-medium (M), medium (L), medium-small (T), small (T)) based on performance metrics in Fig.4. The benchmark is initiated every 180 minutes and 200 samples are collected in an interval for number of months. We use random sampling with an interval of 1 minute and linear combination of the data sets to generate the needed workload.

Table 1: Average Resource Utilization from the workload model

Transaction Classes	Freq.	Bandwidth (kbps)		Processor Capacity (MIPS)		RAM (MB)		Deadline (ms)
		Min	Max	Min	Max	Min	Max	
Trivial (T)	10%	10	100	270	500	100	500	300
Light (L)	20%	55	500	500	1500	750	1500	150
Medium (M)	30%	40	700	330	594	367	1334	367
Heavy (H)	40%	225	875	418	753	400	1375	150

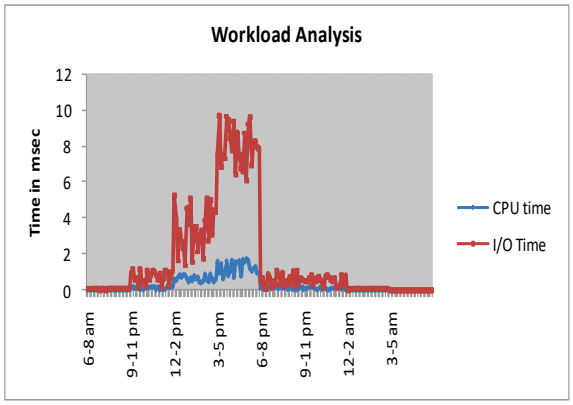

Fig.4. TPC-App Workload Analysis for synthetic workload

For an approximation of resource (CPU, memory, network) usage, related to the number of client session we dissect the workload of the service for 6 months and the findings are depicted in Fig.4. The service workload is under 30% between 12 am to 5am, which is the proper time to aggregate the resource usage patterns such as Resource density, Load Factor for the quantity of service sessions. The workload identification is critical while predicting the accurate resource to forecast the resource prerequisite right then and there. The sampled data set is then used to automate the resource provisioning by multiplexing the appropriate number of Virtual Machines (VM) to the host/Physical Machines (PM) based on workload. The Speculative Resource Provision (SRP) framework deployed in Eucalyptus preprocesses the resultant traces and constructs the pattern table entries to automate resource provisioning speculatively.

Stage II: Machine Learning Algorithms and Speculation

The machine learning algorithms for dynamic resource provisioning such as Neural Network, Linear Regression are used to improve the responsiveness of the cloud provider system, since the time usually taken for resource provisioning in scalable environment is hardly 10-15 minutes. We have chosen, the machine learning algorithms like Neural Network and Linear Regression to evaluate our framework prediction strategy. These algorithms have shown significant impact in the areas like character recognition, image compression, stock market prediction and health care applications.[30].

Neural Networks (NN)

To develop common neural network architectures, we use *Neuroph a lightweight Java neural network framework* comprises of open source Java library which depicts basic NN concepts. The parameters of our experiment are learning rate 0.6, no. of hidden layers =1, number of hidden neurons=6. The learning rate should not be too slow or too fast. We have chosen single hidden layer since most of the neural network based forecasting models achieved its effectiveness in single hidden layer.[30] The neural network consist of set of three layers namely input layer $(x_1 \ldots x_n)$, hidden layer $(h_1 \ldots h_m)$ and output layer $(o_1 \ldots . o_p)$. Hidden layer lies in between input and output layer. Neurons in each layer are interconnected with neurons in other layer which is termed as synapses. The weights are assigned using weight vector w_p to the synapses based on difference between expected and actual output. The updating weight vector is called as learning rate (ρ) which ranges [0, 1].The weight vectors are updated based on Eq.8

$$w_p(t+1) = w_p(t) + \rho \sum_{i=1}^{n} w_{pi} x_i(i) \qquad (8)$$

Linear Regression (LR)

We have implemented multiple linear regression prediction model importing java package *java.math3.stat. regression.*;* . This model provides the relation between independent one or more input variable $(x_1 \ldots x_m)$ with dependent output variable (y) using linear equation. In real time there might be number of input variable and hence it is termed as Multiple Linear Regression. The unknown parameter α_i is used for determining the best fitted model.

$$y = \alpha_0 + \sum_{i=1}^{m} \alpha_i \, x_i \qquad (9)$$

Speculative Approach (spec)

Eucalyptus is a well known private-IaaS platform. In particular, Eucalyptus gives the Amazon Web Services abstractions and services in private infrastructure. The Eucalyptus cloud setup includes web service components such as Cloud controller (CLC), Walrus, Cluster controller (CC), Node Controller (NC) and Storage Controller (SC). The two hypervisors supported in Eucalyptus are KVM and Xen. CLC acts as a user and administrative interface for cloud management and performs high-level resource scheduling and system accounting. Along with CLC functionalities we have added two new interfaces, one for initiating speculative operation (spec_Int.java) discussed above in section 3 and the other for periodic resource monitoring module (rsrcMon_Int.java) in a scheduled time interval. The CC manages instance (i.e., Virtual machines) execution and ensuring user Service Level Agreements (SLAs) per cluster. The Storage Controller (SC) is proportional to AWS Elastic Block Store (EBS). The Node Controller (NC) hosts the virtual machine instances and deals the virtual system endpoints. Walrus offers constant stockpiling to the greater part of the virtual machines in the Eucalyptus cloud and can be utilized as a basic HTTP put/get storage as a service solution. We have implemented our algorithm and evaluated the outcome with/without initiating speculation algorithm. A random (RAN, RA) paradigm is chosen for Cluster and Node controller. There will be changes in the behavior each time when the algorithms run multiple times. The random criterion precisely mimics the real scenario of cloud service providers in a dynamic scaled environment.

5. Results and Discussion

Response time is the significant factor that has many impacts on cloud computing performance. We measured the response time of the service using Neural Network, Linear Regression and Speculative resource provisioning for the dataset generated by TPC-App benchmarking tools to mimic the real time scenario as mentioned in the section 3 and 4.

I. Comparison of Machine learning Algorithm with Speculation

The response time of the neural network algorithm with varying workload in a day is given in Fig.5.The time spent for forecasting the resources are inherently related to system response time. The average response time is termed as 10.98 minutes. During training phase neural network is fed with input vectors and the weight vectors. The response time fluctuates during peak workloads due to extensive disparities in training rates (learning times).

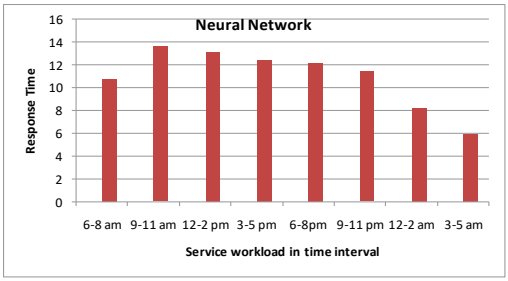

Fig.5. Response time vs workload time interval in Neural Network approach

Linear regression model relates one or more multi valued input variable and an output variable using a linear equation in observed data. The response time measured using linear regression for the varying workload is shown in Fig.6.

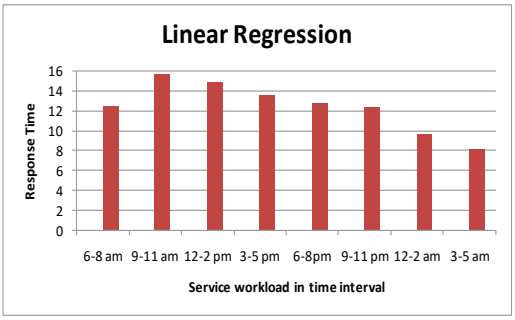

Fig.6. Response time vs workload time interval in Linear Regression approach

Recall the goal of the speculation algorithm in multi-tenant cloud services regarding the optimization of the resource utilization. We have inferred difference in the response time using with/without speculation as in Fig.7.Thus, all of these statistical metric comparisons go in favor of our Speculative Resource Provisioning methodology.

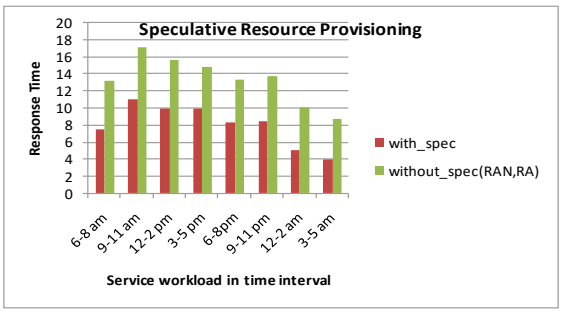

Fig.7. Response time vs workload time interval in Speculative Resource Provisioning approach

We have estimated the response time prior to the workload intervals in two ways. The first way without implementing our speculation algorithm and using eucalyptus cluster and node controller identifier Random and RA scheduled algorithm and tend to know the response time of resource allocation during synthetic workload variation. A comparison of different approaches for predicting/forecasting the resource based on the response time is shown in Table 2.

Table 2: Comparison of response time among machine learning and Speculation

Approach	Response Time (min)
Neural Network	10.98
Linear Regression	12.39
Heuristic (RAN, RA)	13.34
Speculative	6.75

The service provider expects their resources utilized to the fullest. The unevenness in the resource utilization will influence degradation in resource performance. According to the outcome inferred in Fig.8 the speculation algorithm provides efficient resource utilization even during peak workload. The expected resource utilization (service) time incorporates the time spent both waiting for and using the resource. The waiting time for the resource allocation will take meager time when contrasted with other machine learning algorithms Neural Network (NN) and Linear Regression (LR). The decision time spent for resource allocation in speculation is overtaken by rapid service time.

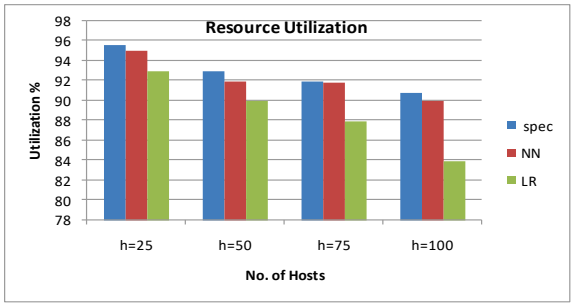

Fig.8. Resource Utilization vs. number of hosts

The resource utilization component of the system using machine learning algorithms, Neural Network (NN) and Linear Regression (LR) is less when compared with our speculation algorithm. This component gets diminished when there is increase in number of hosts as given in Fig.8. The reason behind this is the resources are difficult to pre-judge during peak of workload. The host

level load balancer is underpinning the resources to the dynamic servers according to the criterion applied to Instance controller for the nonspeculation scenario for example NN and LR. In Fig. 9, the effective utilization factor of the resources is demonstrated by varying the client sessions through TPC-App client emulator from 25 to 100. The resource utilization is effectively measured by concurrent usage and its availability. The idleness of the resources during high demand will not consider as a profit for the resource providers. The resource utilization factor provides how effectively the resources are utilized during extreme load. For this test case, we varied the number of users and infer the prominent improvement in speculation i.e. Spec algorithm when compared with a machine learning algorithm.

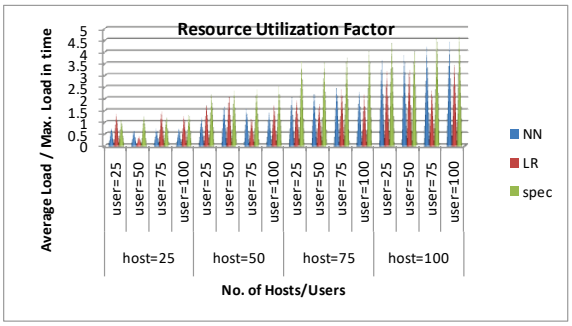

Fig.9. Resource Utilization factor vs. number of client session

The cloud providers favor the end user by offering the best QoS for their service with the lowest price. Customers are accounted based on resource utilization and consumption of service such in case of Amazon EC2. The service consumer prefers economic resource usage. Our algorithm extends the functionality of AWS; As shown in Fig.10.Our approach attracts pay-as-you-go model users by providing effective resource usage and rapid response time.

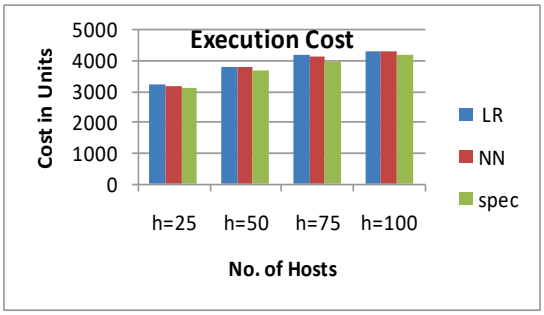

Fig.10. Execution Cost vs. number of hosts

The Energy efficiency is related to effective resource utilization. The moderate resource utilization and lack of idleness upfront the energy consumption. The resources that are not active periodically monitored through command /proc/meminfo and are kept in standby mode until reserved for future allocation.

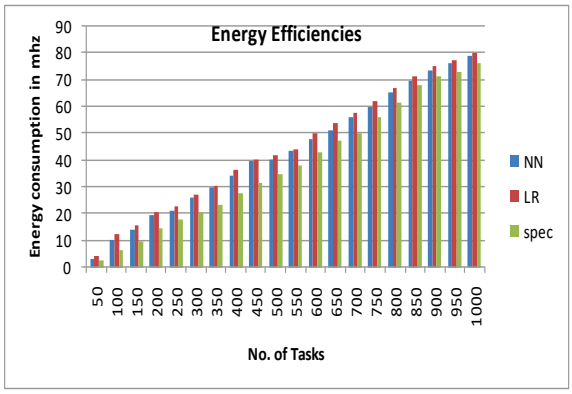

Fig.11.Energy Efficiency based on different task

The dynamic reallocation of the tasks to the virtual machine and its processing time measured for energy efficiency. Fig.11 shows higher energy savings in our approach rather than other machine learning algorithm.

II. Prediction Accuracy of Speculative Approach

The prediction analysis of speculative approach is undergone by confidence estimation as discussed above in section 3.2. Fig.12 presents the accuracy achieved through the Speculative Resource Provisioning system. The predictor achieves >90 percent accuracy during less workload period rather above 80 percent accuracy during peak workload because of invariable load experienced by the resource. But this 80% doesn't affect the performance of the cloud as it would consider as normal processing service with incorrect speculation. The prediction accuracy is measured not only based on frequency of misprediction rather deferred prediction rate is also considered.

We have chosen random jobs (J_1 to J_8) which run in different transaction classes (T, L, M, H) for finding out predicted arrival and service rate.

Table 3:Job Characteristics

Transaction Class	Jobs
Trivial	J_1,J_5
Small	J_3,J_9,J_{10}
Medium	J_4,J_6,J_7
Heavy	J_2,J_8

History Table Depth (HD)

We need to consider the significant of the history depth (HD) parameter during speculative operation. The

history depth determines the number of prior resource accesses in predicting appropriate configured VM multiplexed to PM. HD is related with prediction and in general when history depth increases the prediction accuracy may also improve. The results of prediction accuracy for the history depth 2 (HD_2) and 3 (HD_3) are given in Fig.12.

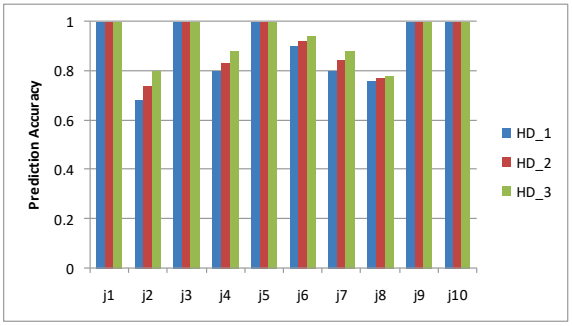

Fig.12: Speculative Prediction accuracy

For Job J2, J4 abrupt rise in accuracy between HD_1 and HD_3, this shows that several frequently accessing patterns share a common sequence of three accesses. As from the above fig it is clearly depicted the prediction accuracy increases while increasing the depths of the history table but it turns down for time constraint application.

6. Conclusion

The aim of our paper is to provide adaptive and intelligent resource allocation framework. In our paper we have focused on design, implementation and evaluation of the framework which ensures automated resource provisioning. The performance of the framework is validated through how the resource allocation mechanism will treat the response time of the task under various workloads. Our approaches shows approximately 25% improvement over other specified algorithm in terms of response time, resource utilization and energy efficiency. Response time inherits the decision taken during VM to PM mapping. Our framework robustly multiplexes virtual to physical resources based on speculative mechanism.

We employ pattern based prediction for determining the exact resource requirement for an application execution which eventually prevent over/under allocation of a resource. The maximum resource utilization and minimum execution time are the touting factors for the cloud users as it inclined on resource performance and cost. Our outcomes suggested a far better approach to work out the performance in virtualized environments by observing service response time, execution time, and resource utilization. We strive to focus on increasing prediction accuracy in future as dynamic resource provisioning is an NP-complete problem. Reliability is strongly emphasized in our system as it offers consumers trustworthiness i.e., they know that they are paying only for what they use. Throughout the study, we have evaluated our speculative approach with several common machine learning algorithms, with a view to provide accurate forecasting ahead of time. We exemplified our proposed prediction techniques in the context of the dataset obtained using TPC-App, a benchmark which is well-established for web service applications. All the resource provisioning stages are facilitated by appropriate resources in our Framework.

References

1. Pooyan Jamshidi, Aakash Ahmad, Claus Pahl, *Cloud Migration Research: A Systematic Review*, (IEEE Transactions on Cloud Computing, , no. 1,2013).
2. Zhen Xiao, Weijia Song, Qi Chen, *Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment*, (IEEE Transactions On Parallel and Distributed Systems, VOL. 24, NO. 6, June 2013).
3. Fetahi Wuhib, Rolf Stadler, and Mike Spreitzer, *A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments*, (IEEE Transactions On Network And Service Management, VOL. 9, NO. 2, June 2012).
4. Hamid Mohammadi Fard, Radu Prodan, and Thomas Fahringer, *A Truthful Dynamic Workflow Scheduling Mechanism for Commercial Multicloud Environments*, (IEEE Transactions On Parallel And Distributed Systems, VOL. 24, NO. 6, June 2013).
5. Kyle Chard, Kris Bubendorfer, *High Performance Resource Allocation Strategies for Computational Economies*, (IEEE Transactions On Parallel And Distributed Systems, VOL. 24, NO. 1, January 2013).
6. Mohamed Abu Sharkh, Manar Jammal, Abdallah Shami, Abdelkader Ouda, *Resource Allocation in a Network-Based Cloud Computing Environment: Design Challenges, Cloud Networking And Communications*, (IEEE Communications Magazine,November 2013).
7. Imad M. Abbadi and Anbang Ruan, *Towards Trustworthy Resource Scheduling in Clouds*, (IEEE Transactions On Information Forensics And Security, VOL. 8, NO. 6, June 2013).
8. Chrysa Papagianni, et.al.,, *On the Optimal Allocation of Virtual Resources in Cloud Computing Networks*, (IEEE Transactions On Computers, VOL. 62, NO. 6, June 2013).

9. Parnia Samimi, Youness Teimouri , Muriati Mukhtar, *A combinatorial double auction resource allocation model in cloud computing*, (Elsevier, Information Sciences, February 2014).
10. Sadeka Islam, Jacky Keung, Kevin Lee, Anna Liu, *Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud*, (Elsevier, Future Generation Computer Systems 28,155-162, May 2011).
11. Harold C. Lim, Shivnath Babu , Jeffrey S. Chase, Sujay S. Parekh, *Automated Control in Cloud Computing: Challenges and Opportunities*, (Workshop on Automated Control for Datacenters and Cloud ACDC'09, June 2009).
12. Hameed Hussain et.al., *A survey on resource allocation in high performance distributed computing systems*, (Elsevier, Parallel Computing vol.39, 709–736, November 2013).
13. Chunlin Li · La Yuan Li, *Optimal resource provisioning for cloud computing Environment*, (Springer, Journal of Supercomputing vol.62:989–1022, November 2012).
14. Dong Huang, Bingsheng He and Chunyan Miao, *A Survey of Resource Management in Multi-Tier Web Applications*, (IEEE Survey Communications and Tutorials vol.16,August 2014).
15. Hao Zhuang, Xin Liu, Zhonghong Ou, Karl Aberer, *Impact of Instance Seeking Strategies on Resource Allocation in Cloud Data Centers*, (IEEE Sixth International Conference on Cloud Computing, pp. 27-34, July 2013).
16. Junwei Cao,Keqin Li, et al, *Optimal Power Allocation and Load Distribution for Multiple Heterogeneous Multicore Server Processors across Clouds and Data Centers*, (IEEE Transactions On Computers, vol. 63, January 2014).
17. Gong Chen, Wenbo He et.al, *Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services*, (5th USENIX Symposium on Networked Systems Design and Implementation,2008).
18. Tuah.N.J, M. Kumar, S. Venkatesh, *Resource-aware Speculative Prefetching in Wireless Networks*, (Springer, Wireless Networks vol.9(1), 61–72, January 2003).
19. Beloglazov.A, R. Buyya, Y.C. Lee, A. Zomaya, *A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems*, (Advances in Computers, Elsevier vol.82, 2011).
20. R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*, (Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, June 21 - 24, 2009).
21. S.K. Garg, R. Buyya, *Network CloudSim: modelling parallel applications in cloud simulations*, (Fourth IEEE International Conference on Utility and Cloud Computing (UCC), pp. 105–113, December 2011).
22. R. Buyya, R. Ranjan, R.N. Calheiros, *Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: challenges and opportunities*, (International Conference on High Performance Computing & Simulation, HPCS'09, IEEE , pp. 1–11, 2009).
23. Chingchit, S., Kumar, M., Bhuyan, L.N., *A Flexible Clustering and Scheduling Scheme for Efficient Parallel Computation*, (13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing, April 1999).
24. Wenjin Hu, Andrew Hicks, Long Zhang,*A Quantitative Study of Virtual Machine Live Migration*, (ACM Cloud and Autonomic Computing Conference (*CAC 2013*) August, 2013).
25. William Voorsluys et.al., *Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation*, (CloudCom '09 Proceedings of the 1st International Conference on Cloud Computing,2009) Pages 254 – 265.
26. Glauco Estácio Gonçalves, et.al., *Resource Allocation in Clouds: Concepts, Tools and Research Challenges* (XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2011).
27. Daniel F. García, Javier García, Manuel García, Ivan Peteira, *Benchmarking Of Web Services Platforms :An Evaluation with the TPC-App benchmark*, (IEEE Computer, vol.36,2003).
28. Caron, Eddy, Frédéric Desprez, and Adrian Muresan. *Forecasting for Cloud computing on-demand resources based on pattern matching*. (2010): 26.
29. Li, X. Yang, S. Kandula, M. Zhang, *Cloudcmp: comparing public cloud providers*, (IMC '10 Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 1–14, 2010).
30. Mohammad Hindi Bataineh, *Artificial Neural network for studying human performance*, (thesis, University of IOWA, 2012).
31. Urgaonkar, R., Kozat, U., Igarashi, K, and Neely, M., *Dynamic Resource Allocation and Power Management in Virtualized Data Centers*, (IEEE Network Operations and Management Symposium (NOMS),2010).