Towards Self Configured Multi-Agent Resource Allocation Framework for Cloud Computing Environments

M.N.Faruk [#1], Dr.D.Sivakumar [*2]
Research Scholar, Bharath University, Selaiyur, Chennai.[#1]
Professor, Dept of IT, Easwari Engineering College, Ramapuram, Chennai. [*2]
justdoit.fff@gmail.com [#1]
dgsivakumar@sify.com [*2]

Abstract - The construction of virtualization and Cloud computing environment to assure numerous features such as improved flexibility, stabilized energy efficiency with minimal operating costs for IT industry. However, highly unpredictable workloads can create demands to promote quality-of-service assurance in the mean while promising competent resource utilization. To evade breach on SLA's (Service-Level Agreements) or may have unproductive resource utilization, In a virtual environment resource allocations must be tailored endlessly during the execution for the dynamic application workloads. In this proposed work, we described a hybrid approach on self-configured resource allocation model in cloud environments based on dynamic workloads application models. We narrated a comprehensive setup of a delegate stimulated enterprise application, the new Virtenterprise_Cloudapp benchmark, deployed on dynamic virtualized cloud platform.

Keywords: Resource Allocation, virtualization, Cloud computing, Agents framework.

I. INTRODUCTION

In Recent development on virtualization and Cloud Computing intend at segregating applications and resources from the primary hardware infrastructures. This delivers rapid flexibility on resources (e.g., CPU, Bandwidth, and Memory) which can be allocated and utilized on demand basis and personalized with dynamic system workloads. Henceforth the Cloud based resource allocations framework would be enlarged on both extremes by elastic mode, through dazzling the load intensity and service anxiety of mounted applications. However, virtualization permits reducing the substantial number of data centers by deploying multiple autonomous virtual machines (VM's) on the equivalent physical hardware. By humanizing energy efficiency, this creates considerable amount of investment for both infrastructure providers and service providers. Moreover, the above mentioned benefits arrived with most expensive bigger system difficulty and vibrant, making it demanding to deliver Quality-of-Service (QoS) assures with highly changeable application workloads. Service providers intended to face the following difficulties during establishments: the number of resources allocated and utilized to a new service mounted in the virtualized cloud infrastructure on-the-fly, to assure mutual agreement between cloud service provider and the end user. The number of resource allocations of mounted applications and dynamic assignment of services depend on their nature and workloads. The numbers of substantial resources are mandatory to prolong at escalating load conditions due to floating workloads. The number of resources can be deployed, without compromising Service Level Agreements, Responding such questions necessitate the capability to determine at run-time performance of deployed applications which may affect if workload is dynamic, through which the capability to forecast the outcome of dynamic resource allocations to acclimatize the system. This paper carried out online application performance prediction framework. The latter it permits to proactively deploy fresh workload conditions with the system evading SLA breach or inefficient resource utilization. Over the years, a numerous performance prediction methods relies on architecture-based performance model have been originated by the performance engineering community [15]. However, these methodologies are embattled for offline usage at system design models and consumption time, and these generally engaged for evaluating substitute system models and for resolving and competence planning before engaging the datacenters into production. The benefit of such models, balanced [10, 11, 13], based on traditional performance models.

They could potentially permit to explicitly confine the performance influences of software architecture, the application custom profiles as well as the completing environment. In the meanwhile the stability of core architecture characteristically should not change throughout the execution, the resource allocations setup at the various levels of the deployment domain may change regularly. In this context, the input sequences conceded by the service should have direct contact on the group of software core components concerned in executing the different natures of service and their inner performance and resource demands. Henceforth, the detailed performance model is confining the performance-relevant features of both the software level architecture and the multi-layered domain. In this paper, we deliver a novel approach to self-configured resource allocation framework in cloud environments. We investigate to make use of these models to treat online performance analysis and to forecast the dynamic variations in cloud user workloads, also to forecast the implication of relevant reconfiguration actions, commencement to evade SLA violations or inefficient resource utilization. We analyzed through the experimented setup with a prototype application model, the new Virtenterprise_Cloudapp benchmark, adopted in a virtualized cloud environment. The setup defined as a proof-of-analysis presenting the stability of using architecture level performance models during run-time. The proposition of this article are: i) a autonomous loop of self-configured framework and a relevant resource allocation algorithmic model for virtualized cloud environments depend on online performance samples, ii) an detailed view of our approach in the prototype on a enterprise application, the new Virtenterprise_Cloudapp scale, of a sensible range and complexity, iii) the investigational assessment of the urbanized framework representing its effectiveness and convenient applicability. The other relevant things of this paper are includes as follows: Section 2 offers some groundwork on performance analysis with models; Section 3 illustrates our self-configured resource allocation framework. In Section 4, we depict the prototype model of the Virtenterprise_Cloudapp benchmark, the ensuing performance analysis, and discussion on results of our approach. Finally, we conclude with integration of our work.

We differentiate between evocative architecture-level performance samples and extrapolative performance samples. The former illustrates performance-relevant features of software platform and execution environments (UML models embedded with performance observations). The latter confine the chronological system nature that used for forecasting the performance in the sense of logical or simulation methods. Over the past decade, a number of architecture-level concert meta-models for explaining performance-related scenarios of software architectures platform and execution domains have been detailed by the performance engineering standards and the most outstanding examples are meta-models which are KLAPER, PCM and CSM and MARTET user profiles and UML SPT [9, 14]. The general target of this created works is to forecast the aspects of system performance by renovating architecture-level performance samples into predictive performance oriented samples in a mechanical or logical manner. They are usually tend to evaluate substitute system designs or for sizing and capacity forecasting before deploying the system into production. We investigate the main use of architecture-based performance samples as a proof for online performance forecasting through system operation [5]. These samples allows for modeling the architectural top layers through their configurations and the nature of included services explicitly. Henceforth, the comparative performance analysis of different execution domains or software heaps can be modeled and captured. Particularly when modeling virtualized cloud platforms, where virtualization models may tend to change during execution, an explicit design of the performance manipulation is valuable. In modern trend, with the rapidly increasing deployment of component based software engineering models and the software performance evaluation standards has keen on embedding and extending predictable performance engineering methods to support component- based machines which empirically used as basement for building featured enterprise level applications. The most prominent component-oriented performance modeling languages has described in the form of parameterization with supporting tool through PCM (Palladio Component Model) [8,9]. In this paper, we define the Palladio Component Model as architecture-based performance model by allowing to explicitly sampling dissimilar usage profiles and improper resource allocations. To obtain predictions from the PCM samples, we depict another framework called Simucom that inherits a queuing-network based simulation [1].

In pursuance to obtain the dynamic behavior and resource utilization of a component, the Palladio Component Model has four factors [1]. Perceptibly, the component's dynamic execution may cause issues with its performance. In addition, the components may fall on other external services whose performance factors need to be analyzed. Moreover, in both way the component is enhanced (the usage profile together with crucial service input parameters, and the execution platform in which the component is dynamically deployed). PCM permits model processing services like, CPUs and HDD. The resource allocation prototype explains the mapping of component occurrences from the system components to resources which are detailed in the resource environment prototype. The basic interface model describes the user activities. It confines the primary services that are referred during run-time (dynamic) and the request mapping frequency through which services are referred and finally the input parameters will be approved by them.

II. SELF- CONFIGURED RESOURCE ALLOCATION FRAMEWORK

In this section we describes our self- configured resource allocation algorithmic model rely on a control loop model.

A. CONTROL LOOP ADAPTATION

The deployment of control loop is detailed in the Fig. 1. The Forecasting Agent, we consider that dynamic changes of the workload are both declared by the end users or by methods similar to workload forecasting [3]. In the utility agent, we use the detailed software performance samples to forecast the effect of tangible changes and to conclude which actions need to take. The Forecasting phase further implements the reconfiguration options taken into consideration. The Modern cloud virtualization and middleware technologies offer several possibilities for feasible dynamic resource allocation with system reconfiguration. The virtualization permits to add/remove virtual server's cores to virtual machines or to modify the hypervisor scheduling constraints, for example, increasing/decreasing the capability (CAP) parameter. Application servers naturally generate application server cluster models and cluster nodes. Henceforth, virtualization permits to tradeoff and switch virtual machines from one physical datacenter to another physical datacenter.

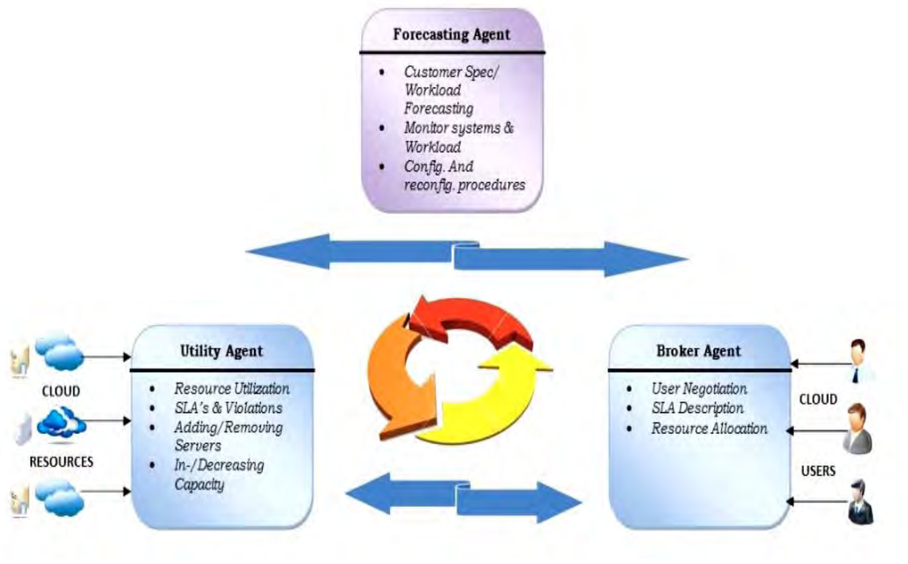

Fig. 1: The Model Configuration process

The described dynamic reconfiguration alternates have their precise advantages and reasonable drawbacks. Most of them used during run-time (on-the-fly) but needs an extraordinary system configuration or promote elevated reconfiguration overhead (Virtual Machines live migration). Hence, for our self- configured resource allocation algorithm, we spotlight on adding/removing virtual datacenters (CPU's) and adding/removing application based servers to an server cluster.

B. RESOURCE ALLOCATION ALGORITHM

In this segment, we deliver a prescribed definition of resource allocation algorithm. This algorithm involves of two major phases: GET phase and SET phase. The GET phase allocates required tangible resources until all client's Service Level Agreement are fulfilled. The SET phase has interim behavior and optimizes the resource effectiveness by de-allocating tangible resources in which the resources are not fully utilized efficiently. We deliver the algorithm general formation, through which it can be used on different varieties of resources models and different platforms.

Formally, the Cloud virtual domain can be symbolized by 3-tuples

$O = (R, S, N)$ where, $R = \{r_1, r_2, ..., r_o\}$is the group of resources models (different types of Virtual Machines (VM's) are embedded in the cloud environment),

$S = \{s_1, s_2, \ldots s_n\}$ is the set of services accessible in the cloud environment,

$N = \{n_1, n_2, ..., c_l\}$ is the set of client workloads and corresponding SLAs. Each $n_q \in N$ is described as triple (s, φ, ρ) where $s \in S$ is the service is being used, φ is represented as the different requests workload intensity (expected arrival rate), and ρ refers the Average Response Time (described in SLA) requested by the client. We also inherit the other following functions:

$E \in [S \rightarrow 2^R]$ type of resource are needed by the service $s \in S$, F ∈ [S × R→ Q ^{s,t}] modulated as resource allocation assignment of each given service s ∈ S for a set of instances Q^{s,t} of resource type r ∈ R (Virtual machines instances). For Each resource type request is considered to be owed a number of equal processing resources (HDDs, CPUs). officially, the resource models instances q ∈ Q^{s,t} is described in the form of triple $(\pi, \mu, \bar{\mu})$, where π defines the reasonable processing rate of its current virtual resources, μ is the no. of processing virtual resources currently allocated to the clients. $\bar{\mu}$ is the utmost no. of currently processing resources can be allocated dynamically (no. of CPUs mounted on each physical machine).

D ∈ [S → T^{+}] specifies the demand factor of the given service s ∈ S in the equivalent variable units as the processing rate(π) of the virtual resource type. We conclude the following performance metrics:

$N_{tot}(n)$ The total number of user requests described in client workload intensity. n ∈ N completion time per unit (request throughput).

$T_{avg}(n)$ Is the given average response time of every service request in the specified client workload n ∈ N and

$R_{util}(r)$ is the avg. utilization of defined virtual resource type r ∈ R over all illustrations of the virtual resource,

$\bar{R}_{Util}(r)$ is the maximum permitted avg. utilization variable for every resource type r ∈ R.

Finally, we classify the subsequent predicates:

P($N_{tot}(n)$) for c ∈ C is simplified as ($N_{tot}(n)$ = c[φ]),

P($T_{avg}(n)$) for c ∈ C is simplified as ($T_{avg}(n) \leq$ c[ρ]),

P($R_{util}(r)$) for r ∈ R is simplified as ($R_{util}(r) \leq \bar{R}_{Util}(r)$).

For a internal configuration detailed by a resource allocation function F that can be satisfied by the following conditions (∀n ∈ N: P ($N_{tot}(n)$) ∧ P ($R_{util}(r)$) ∧ (∀r ∈ R: P ($R_{util}(r)$).This condition is confirmed in terms of our online performance forecast method.

The client workloads changes dynamically every time ~~N~~ $\tilde{N}$ (for example if a new client workload intensity $\tilde{n}$ = (s, φ, ρ) is programmed for dynamic execution or transform in the workload intensity factor φ of a previous workload forecast), we presume that the online performance prediction method to predict the effect of this dynamic behavior in the overall client system workload. If any SLA violation is identified with any client, the GET phase of our given algorithm is mounted in which dynamically allocates additional tangible resources until all client's Service level agreement are satisfied. After the GET phase completed, the SET phase is started to collect logs and optimize the resource efficiency. If there are no SLAs violation happens, the SET phase starts immediately. In the subsequent section we elaborate GET and SET phases in more detail.

C. GET PHASE

The representation of this algorithm described in mathematical approach of pseudo code presents more fundamental heuristic for allocating resources dynamically for different kinds of services through which the client SLAs requirements are satisfied.

Algorithm 1: Get Phase – Resource Allocation Model

```
While $\exists n \in \tilde{N} : \neg P(R_{util}(r))$ do_
    For all $r \in E(n[s]) : \neg P(R_{util}(r))$ do
        While $cap(n,r) \leq \overline{cap}(n,r)$ do
            if $\exists q \in F(n,[s],r) : q(\mu) < q(\bar{\mu})$ then
                $q[\mu] \leftarrow q[\mu] + 1$
            Else
                $F(n,[s],r) \leftarrow F(n,[s],r) \cup \{\hat{q}\}$ End_if
        End_while
    End_for
End_while
```

essentially, while being a client average response time, the SLA is violated, the above mentioned algorithm dramatically increases the quantity reasonably of allocated resources with all virtual resource types incurred by the required service that are currently go beyond their maximum permitted utilization $R_{util}(r)$ This is because of the hypothesis that SLA violations are rooted by minimum of one virtual resource type and the violated SLA becomes serious bottleneck.

By increasing the total number of allocated resources formation as follows, If there are samples of the over utilized virtual resource type r (VM's) which has a few processing resources availability (like Virtual CPUs) that are not promoted yet, the additional resources are need to be allocated, or else, a new type of resource instance $\hat{q}$ is included (new Virtual CPUs is initiated).In our algorithm describes, an elevated resource instance sequentially to the total required capacity. Our algorithm also presumes that there is an countless amount of cloud resources available at various clusters henceforth, this required capacity increase repetitively until the quantity of allocated resources reaches on various clients. The capacity CAP (n, r) defined as

$$\overline{cap}(n,r) = \left[\frac{\sum_{n \in N} n[\varphi].D(n[s])}{\sum_{n \in N} n[\varphi].D(n[s])} \right].cap(n,r) --- (1)$$

The above mentioned equation is an predictable upper bound capacity of all virtual resource type r required to hold client workload n, based on a aspect considered by the specification changes dynamically and the already allotted capacity cap(n, r). This is being considered as the ratio of the recently specified arrival rates from various centers and the actual arrival rates on both integrated with their corresponding defined virtual resource demands. It is projected to diminish the number of perspectives by which online performance forecast to be executed when penetrating for a reasonable configuration.

D. SET PHASE

The SET phase targeted to optimize the resource efficiency by demanding to release virtual resources which are not utilized at the maximum by the respective client workloads.

Algorithm 2: Set Phase – Optimization with Allocated resources

```
For all $n \in N$ do
    while $\exists r \in E(n[s]) : \overline{R}_{Util}(r) - R_{util}(r) \geq \varepsilon$ do
      if $\exists q \in F(n,[s],r) : q(\mu) > 0$ then
         $q[\mu] \leftarrow q[\mu] - 1$
         if $\neg P(R_{util}(r))$ then
         $q[\mu] \leftarrow q[\mu] + 1$
     End_if
   If $q[\mu] = 0$ then
       $F(n,[s],r) \leftarrow F(n,[s],r) \setminus \{q\}$
          End_if
      End_if
   End_while
End_for
```

The resource optimization algorithm is elaborated to overall client workloads n ∈ N. During execution there is a available virtual resource type t is directed to service s of the currently given workload intensity n whose delta tradeoff between the maximum utilization $R_{util}(r)$ and current possible utilization $R_{util}(r)$ is larger than a predefined assumed constant ε, the total amount of resources allocated to this possible current service will be minimized, Hence with for all resource type instance q of r which presently has some allocated resources(e.g. Virtual CPUs) and the total number of allocated resources will be reduced. If the client SLAs violation has to be forecasted after this drastic change, the current change is partially reversed. In case of no instance has outstanding allocated resources, the current instance q can be detached through a group of defined virtual resource type instances (Virtual machines may be unavailable). Notably the group of scheduled virtual resource instances q can also turn out to be empty, in case there is no service left in the queue using the corresponding virtual resource type r.

III. EXPERIMENT CONFIGURATION

In the experimental setup of hardware environment, we mount five blade servers from a cloud cluster virtual environment. Each prescribed server is being prepared with two or three Intel Xeon E5400 4-core CPUs working with 2.66 GHz and 32 GB of internal memory. The defined machines are connected through a 1 GB LAN cable. The above figure simplifies the experiment setup. On each machine has Citrix Xen-Server 5.5 considered as the cloud virtualization layer in all the XenServer's VMs, we mount all the benchmark components like (Supplier emulator, application servers, load balancer, driver agents).

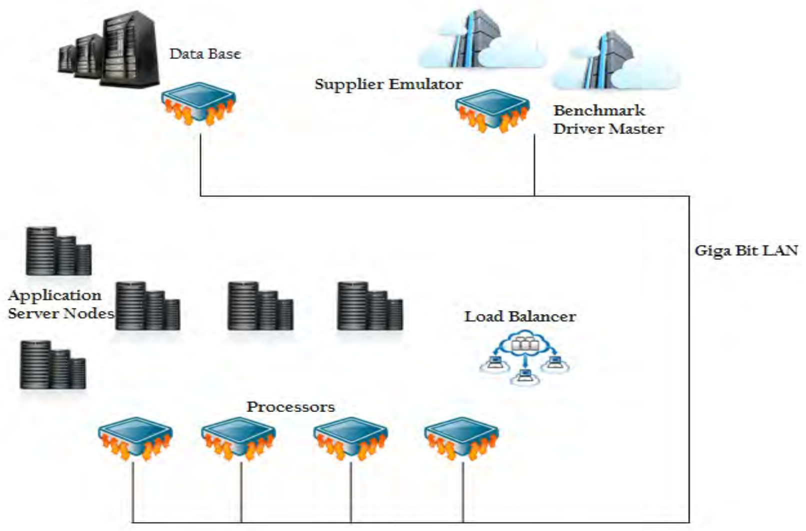

Fig. 2: The Experiment setup

Each component works with its own Virtual Machines, at initial stage configuration this mounted with two virtual CPUs. The operating system view these VMs perform Cent-OS 5.3. As Java Enterprise Edition as application server, we also assume the Oracle Web-logic Server (WLS) version 10.3.3. The load balancer processed with proxy ver.1.4.8 by using round-robin mechanism as a primary load balancing technique. The Benchmark master driver agents are incorporated through Faban-Framework that is migrated with this benchmark. The database is an Oracle ver.11g database server mounted on each Virtual Machines (VM) with eight Virtual CPU (VCPU's) on each individual node has Windows Server 2008 operating system. The Virtenterprise_Cloudapp benchmark application is detailed in a cluster of Web-logic servers (WLS) nodes. For the assessment perspective, we mentioned reconfiguration options relating to the VMs are prepared with Web-Logic Servers cluster nodes and the VCPU's. The WLS nodes may added/removed with WLS cluster nodes also the VCPU's are added/removed with VM. These reconfigurations are applicable at run-time (on-the-fly), Therefore this can be functioned while the Virtenterprise_Cloudapp application is mounted. In our simulations, the VMs map to every virtual resource type instances through reallocation algorithm and their VCPU's are designed to the capacity parameter (CAP).

IV. EVALUATION

The evaluation segment includes various scenarios and approaches presented in the following section.

A. MOUNTING A NEW SERVICE

In the first scenario is proposed to calculate the outcome of allocation while a new service is being deployed in the virtual cloud environment during run time. Imagine that there are four different services proposed in our virtual cloud environment running on single node with mounted two VCPU's by default configuration mentioned as $VServ_0$ .The SLAs of presently running services are narrated as: (Create-Vehicle Web Service, 17, 84ms), (Purchase, 11.5, 90ms), (Storage, 10.5, 90ms), (Browse, 35, 96ms). This specification defines the same data as the most of the common client workload specifications. Fig.3.

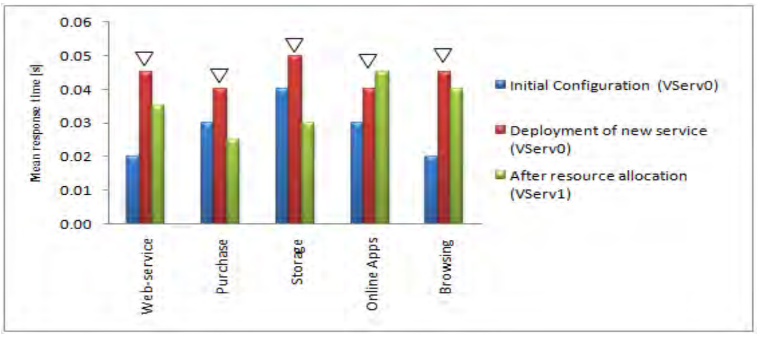

Fig. 3: The current response times each mounted services and their respective SLAs (denoted by ∇) before and after resource reallocation while adding a new service.

The newly added service with the respective SLA (Create-Vehicle Web Service, 17, 84ms) is added. To guarantee that all the defined SLAs are still preserved following deployment of each new service, our self-configured resource allocation technique is initiated. The results are drawn in *Fig. 4(i)*. After mounting the newly deployed services to the model, the mechanism forecast the SLA violations on the Create-Vehicle Web Service and Purchase services. Hence, the GET Phase of the reconfiguration algorithm initiates and refers a load increase by one, henceforth adds an extra VCPU to the existing already mounted node which is mentioned as VServ$_1$. After this variation, the simulation determines the fulfilled SLAs, therefore the algorithm go into the SET-Phase and tends to decrease the overall quantity of already used resources with all workload intensities, but this fails because after the SLA's of Create-Vehicle Web Service and Purchase service are again desecrated. Therefore, the resultant values of our configuration algorithms involves of single node with three VCPU's. The above mentioned behavior was established in given experiments drawn in *Fig.4 (ii)*. The measurement depicts that by default resource allocation with the SLA for Create-Vehicle Web Service and Purchase service cannot be continued. However, subsequent resource allocation sequence proposed by our algorithm, all SLA's are satisfied finally.

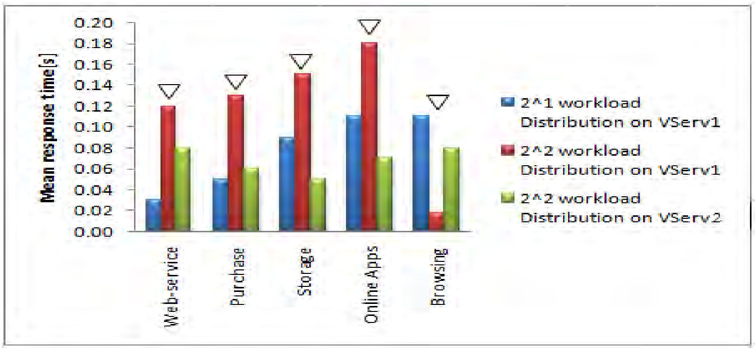

Fig. 4. (i) Escalating workload from 2^1 to 2^2

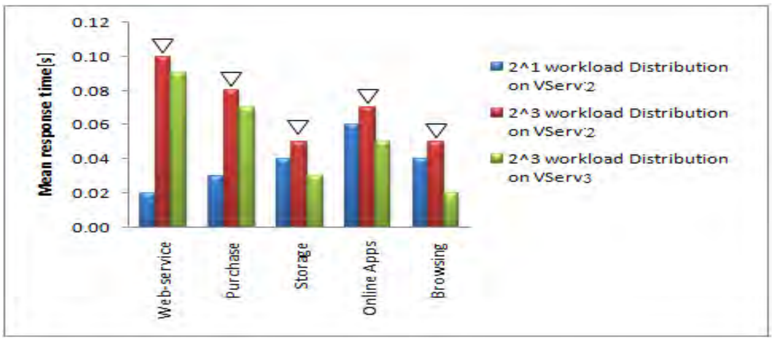

Fig. 4.(ii) escalating workload from 2^2 to 2^3

Fig. 4(i & ii) the response times with different dynamic workload from 2^1 to 2^2 and 2^2 to 2^3, respectively (SLAs denoted by ∇).

Our initial stage the five services are processing on one cluster node with three VCPUs (VServ$_1$) with doubled the average workload and the subsequent SLAs (Create-Vehicle Web service, 50, 94ms), (Purchase, 40, 84ms), (storage, 45, 140ms), (online apps, 35, 110ms), (Browsing, 60, 140ms) which are partially satisfied at initial stage. Now, we boost the workload to 2^2 with the average load. For this newly mounted workload, the reallocation mechanism identifies the basic violation of the SLA's and prefer the reallocating the predefined system resources which is using two node. Applying this configuration to the benchmark application, the SLA's are satisfied formerly. For the dimension results see Fig. 4 (i). In the subsequent step, we escalate the cluster workload intensity to 2^3 with the standard load, without modifying the SLAs. Again, this tend to a breach the SLA's in proposed simulation results. Henceforth, we deeply analyze our algorithm, finding a new appropriate configuration with three standard nodes.

The experiment figures are described Fig. 4 (ii). However, the results show that after evolving reallocation mechanism still the SLA of the Browse service is violated. This is not because of mistakenness of our method, but relatively due to elasticity problems of the database machine which is present, again which is not commanding sufficiently to handle the newly mounted different workloads while accompanying the original SLAs. Hence with, we certain to give more efficient database through which the SLA's should be satisfied.

B. WORKLOAD DECREASE

This feature is to calculate our proposed framework at different scenarios in which the workload is reduced. The goal is to release resources which are not utilized proficiently and therefore escalating the system efficiency. Consider that the situation all services is processed with 2^3 the common workload intensity on three nodes which needs totally 11 VCPU's and all the SLA's are satisfied (VServ$_3$). Now, we reduce the workload into 2^2 the average load.

For this modification, our approach forecast that two standard nodes with a total number of 7 VCPUs (VServ$_4$) are adequate to handle the reduced workload. The dimension results are depicted in Fig. 5, the representation shows that the recommendation is possibly correct. However notably the configuration VServ$_4$ with the average response time increases drastically, but still the SLAs is pleased.

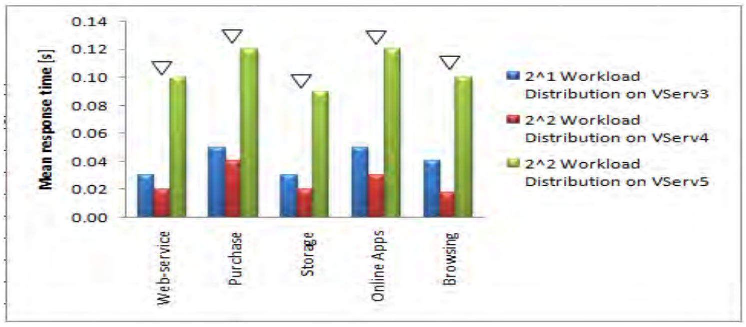

Fig. 5: The response times for processing 2^2 workload intensity before and after reconfiguration and with fewer resources.

The confirmation on resource allocations may not be further condensed while obtaining the SLAs, we investigate further on allocating resources manually to single node with 4 VCPUs (VServ$_5$). The consequences for this configuration displays this would infringe the SLAs, practically we previously found configuration would be valid.

V. RELATED WORK

In most of the related work which is configured resource allocation mechanisms is done on offline mode that can be identified in the areas of capacity identification, planning and resource management. In recent trends the virtualization and Cloud environment also shows other variations on automatic resource configuration and management. In this segment, we describe a detailed notion of the concluding aspects and then deliver some case study in detail. Many of the researches worked with resource allocation issues using various approaches similar to multiple knapsack, bin- packing, issues, etc. in context the dynamic resource allocation was previously research pointed this issues using regular linear optimization methods[11]. Non-linear optimization techniques are purely based on simulated annealing [7] and fuzzy logic [8]. However, the resource allocation method issues in virtualized environment which are more difficult due to the depletion of virtually organized resources. The emergent of cloud computing idea with multiple platforms, there are quite a few methodologies on QoS and resource management methods during runtime [2, 8, 11, 15]. Moreover, these advancements are frequent on a high configured resource management issues and contract with very coarse-grained resource allocation mechanisms issues (e.g. are limited to only mounting and dismounting Virtual Machines [2]) or target on different optimization techniques.

VI. CONCLUSION

In this presented work, we invented a novel approach to self-configured resource allocation framework on-the-fly. We described performance models to forecast the effect of fluctuations in the different cloud service workloads and the respective system reconfiguration actions. By using cloud virtualization methods, we implement these dynamic allocations to the Virtenterprise_Cloudapp benchmark to assess the samples for online performance prediction. The resultant values depicts that our framework can be embedded to dynamic environment to discover efficient resource allocations formations and satisfying specified SLAs. For an example, in the defined method we can save and utilize up to 40 percent of the resources. In the upcoming future, we propose to expand our resource allocation framework with enhanced heuristics for discovering resource allocations. Moreover, we sketch to evaluate our framework with different types of resources. In addition to that, we suggest that the consequence of dynamically shared resources in virtualized cloud environments by enlarging the performance methods with virtualization techniques and variant physical resources network and storage.

REFERENCES

[1] GENI System Overview. http://www.geni.net/.

[2] J. Almeida, V. Almeida, C. Francalanci, D. Ardagna and M. Trubian, "Resource Management in the Autonomic Service-Oriented Architecture," in ICAC '06:IEEE Inter. Conf. on Autonomic Computing, 2006, pp. 84–92.

[3] D. Chess, A. Segal, I. Whalley, and S. White, "Unity: Experiences with a Prototype Autonomic Computing System," in 2004. Proceedings. International Conference on Autonomic Computing, 2004, pp. 140–147.

[4] 'Amazon Elastic Compute Cloud'. http://aws.amazon.com/ec2/.

[5] X. Wang, D. Lan, G. Wang, X. Fang, M. Ye, Y. Chen and Q. Wang,"Appliance based Autonomic Provisioning Framework for Virtualized Outsourcing Data Center," in ICAC '07. Fourth Int. Conference on, 2007, pp. 29–29.

[6] F. Hermenier, X. Lorca, J. M. Menaud, G. Muller, and J. Lawall, "Entropy: A Consolidation Manager for Clusters," in VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2009, pp. 41–50.

[7] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. Intl. Journal of Supercomputer Applications, 11(2):115–128, 1997.

[8] J. Frey, T. Tannenbaum, M. Livny, I. Foster S. Tuecke. Condor- G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing, 5(3):237–246, 2002.

[9] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pages 59–72, New York, NY, USA, 2007. ACM.

[10] C. Zhao, S. Zhang, Q. Liu, "Independent Tasks Scheduling Based on GeneticAlgorithm in Cloud Computing", IEEE Computer society,978-1-4244-3693-4,2009.

[11] H. Zhong, K. Tao and X. Zhang, "An Approach toOptimizedResource Scheduling Algorithm forOpen-source Cloud Systems" ,IEEE Computer Society, volume no.978-0-7695-4106-8, 2010.

[12] B. Sotomayor, K. Keahey, and I. Foster,"Combining batch execution and leasing using virtual machines",ACM 17th International Symposium on High Performance Distributed Computing, pp. 87-96,2008.

[13] M. Mazzucco, D. Dyachuk, and R. Deters, "Maximizing Cloud Providers' Revenues via Energy Aware Allocation Policies," in 2010 IEEE 3rd Int. Conf. on Cloud Computing. IEEE, 2010, pp. 131–138.

[14] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas, "Demystifying energy consumption in grids and clouds," in Green Computing Conference, 2010 International, 2010, pp. 335–342.

[15] B. Rajkumar, B. Anton, and A. Jemal, "Energy efficient management of data center resources for computing: Vision, architectural elements and open challenges," in Int. Conf. on Parallel and Distributed Processing Techniques and Applications, Jul. 2010.