6-26-2018

Modeling and Simulation of Multi-tier Enterprise IT System

Abhinay Puvvala
Tata Consultancy Services, p.abhinay@gmail.com

Veerendra Kumar Rai
Tata Consultancy Services, veerendrak.rai@tcs.com

Modeling and Simulation of Multi-tier Enterprise IT System

Completed Research Paper

Abhinay Puvvala
Tata Research Development and Design Centre, Tata Consultancy Services, 54 B, Hadapsar Industrial Estate, Pune-411013, India
p.abhinay@gmail.com

Veerendra K Rai
Tata Research Development and Design Centre, Tata Consultancy Services, 54 B, Hadapsar Industrial Estate, Pune-411013, India
veerendrak.rai@tcs.com

Abstract

This paper discusses modelling and simulation of multi-tier enterprise IT system. The layers in multi-tier architecture consist of web layer, application layer and database layer. Entities in the multi-tier system have been abstracted out into 3 categories- consumer, resource and router. Existing modelling and simulation frameworks for multi-tier systems focus on power management or performance of load balancing algorithms. Our framework enables seamless modelling, simulation, and experimentation of a wide range of what-if scenarios in multi-tier systems while encapsulating all the variations that arise due to configuration, composition, design and deployment. As an illustration, we discuss and simulate prediction of bottleneck scenario with results.

Keywords: Enterprise IT, Multi-Tier System, Simulation framework, What-if scenarios

Introduction

Propelled by the prevalence of internet based applications, the multi-tier architecture has gradually become the industry standard for developing any scalable enterprise application. The logical and often, physical, separation of presentation, processing and data management functions in an application is the defining characteristic of a multi-tier system. The flexibility and modularity such a setup offers is a major appeal to system administrators. Although referred to as multi-tier or N tier architecture, the number of tiers is often limited to three, as each additional layer leads to additional complexity without adding benefits (Schuldt, 2009).

A typical multi-tier system has the presentation layer at the top. The presentation layer is what user sees and interacts with. The function of this interface is two-fold - translate users' actions and pass them on to the logic layer and transform the results back to a format that user can comprehend. Following the presentation layer is the logic layer, where all the logical decisions, evaluations, calculations are made. This layer is also responsible for all the data movement across tiers. Further down is the data management layer, where all the database servers reside. The interaction with the database is done using standard languages such as SQL queries using database specific protocol over TCP/IP. Over the last decade, multi-tier architecture has been used for the design of applications like ecommerce websites, web hosting, social networking sites and content delivery platforms. Although all these applications have the same underlying architecture, they vary in terms of composition, configuration and other deployment requirements. Evaluating the performance to determine the resources needed to deliver a pre-specified Quality of Service for different application environments under varying load and system

capacity is an extremely challenging task. Also, each application tier has its own set of performance characteristics and modelling them in unison is not straightforward. Further, the concurrency limits and caching at each tier would complicate the task in hand (Urgaonkar et al. 2005).

One way to conduct such analyses is to build test beds that closely resemble the actual environments. However, testbeds are very expensive and more importantly are restricted by their ability to replicate varied environments. While mathematical representations are not cost prohibitive, the inadequate level of endogeneity associated with them limits the range of what-if analyses that can be performed. Also due to this, mathematical representations fail to capture the interplay between various critical variables that are beyond model boundaries leading to imprecise evaluations.

An alternative is to build a simulation environment where various hypotheses can be tested prior to provisioning capacity. In the case of IT systems, where access to the infrastructure incurs huge costs, simulation-based approaches offer significant benefits to evaluate various what-if scenarios at minimal costs. In addition, providers can use these simulation environments to assess various resource leasing strategies by varying load and corresponding pricing strategies.

In this paper, we propose a framework that enables seamless modelling, simulation, and experimentation of multi-tier systems while encapsulating all the variations that arise due to configuration, composition, design and deployment. The framework bases itself on the agent based modelling paradigm. The framework, in addition to providing support for modelling and simulation, can evaluate various scheduling and allocation policies of atomic components in the system such as load balancers, hypervisors etc. Further, the iterative nature of agents in the framework enables the user to expand or contract layers within each tier in line with the level of granularity needed for analysis.

Related Work

Simulation frameworks

The literature is replete with simulation frameworks for IT infrastructure systems such as Grid platforms, Cloud platforms etc. Grid systems are mostly used to deliver high performance computational services for data intensive scientific applications. SimGrid (Casanova, 2001) is a generic framework for simulation of large scale distributed systems such as Grids, HPC, P2P and Cloud systems. It can be used to evaluate heuristics, prototype applications or even assess legacy MPI applications by simulations. GangSim (Dumitrescu and Foster, 2005) is also a simulator for Grid systems with a focus on job scheduling under complex workload and system characteristics. Alternatively, GridSim (Buyya and Murshed, 2002) is an event-driven simulation toolkit for heterogeneous Grid resources. It supports modelling of grid entities, users, machines, and network, including network traffic. While these simulators focus primarily on Grid systems with distributed environment, there are Cloud specific simulators in the literature too.

CloudSim (Calheiros et al., 2011), a simulation framework for Cloud computing, supports modelling and simulation of large scale cloud computing infrastructure which includes individual components such as data centers, virtual machines and computing nodes. CloudSim can also evaluate various policies used in day to day handling of Cloud computing infrastructure. The simulator allows the flexibility to switch between space and time shared processor core allocation to virtualized services. Further, there are variants of CloudSim such as NetworkCloudSim (Garg and Buyya, 2011) and CloudAnalyst (Wickremasinghe et al., 2010) that focus on certain aspects of simulations in Cloud environments.

There are other simulators like iCanCloud (Castane et al. 2012), GreenCloud (Kliazovich et al., 2012) that provide platforms for modelling and simulation of Cloud environments. GreenCloud (Kliazovich et al., 2012) is a simulation environment for energy-aware cloud computing data centres. Along with the workload distribution, the simulator is designed to capture details of the energy consumed by data center components (servers, switches, and links) as well as packet-level communication patterns in realistic setups. iCanCloud (Castane et al. 2012) specializes in handling large scale environments, with customizable hypervisors for integrating various cloud brokering policies.

Multi-tier systems

Researchers have modelled multi-tier Systems in different ways. Urgaonkar et al. (2005) have represented multi-tier systems as a series/network of queues to study the behaviour under scenarios ranging from capacity provisioning, application configuration, bottleneck identification and request policing. Application of queueing models to represent multi-tier systems has been popular across the research community. Slothouber (1996) has proposed a queuing model of a Web server serving static content. The model employs a network of four queues— two for the Web server and the other two to represent the Internet communication network. Similarly, Chandra et al. (2003) have represented a single resource of a webserver with queueing networks. The parameterized model was then used to determine resource allocation needed to meet response time targets. Menasce (2003) has used both Markov chain model and queuing network model to represent multi-tier systems. Stewart et al. (2005) modelled each tier as an autonomous component to assess the overall performance.

Some studies such as (Abdelzaher et al., 2002; Villela et al., 2007; Cecchet et al., 2002; Chen et al., 2006) have modelled each tier in a multi-tier system in isolation, thereby limiting the impact of concurrency on system level performance. While some studies (Lieu et al., 2009; Hsu and Feng, 2005; Rusu et al., 2006; Elnozahy et al. 2003) have modelled multi-tier systems with a view to study their energy consumption and test various power management techniques. MDCSim (Lim et al., 2009) is a multi-tier system simulation platform focuses on performance and power issues at varying workloads. DRepSim (Tang et al. 2005) is another simulator developed for studying the performances of the dynamic replication algorithms. The impact of replication algorithms on average response time has been studied in this work.

The Framework

The multi-tier framework consists of three principal entities- consumers, resources and routers. Entity view of multi-tier framework can be abstracted in terms of these entities. Consumers are entities who arrive with the purpose of being served by the 'resources'. Batch jobs, logical reads and user calls are some of the instances of consumers. Resources are entities which serve the existing consumers. Database, applications and webservers are some of the instances of resources. Routers implement the policy by which resources are allocated to consumers or adding more capacity to resources.

Figure 1 shows an entity relationship model comprising of consumer, resources and routers as entities and their attributes.

Consider a typical multi-tier application consisting of 3 tiers – Web server, App server and Database server. Each of these layers comprise of multiple servers (Figure 2(a)). While servers on Web and App layers are replicas and exist to primarily share load on that layer, the servers on Database layer are not replicas. The data stored on D1 and D2 is different. As often seen in hardware systems, there is a hierarchy of virtualization beneath each of these servers. As shown in the Figure 2- (b), servers on all the layers are hosted on 3 virtual machines – VM1,VM2 and VM3. The servers D1, W1, A1 are hosted on VM1. VM1 runs on Windows environment and is hosted on S1 along with VM2 which also is on Windows environment. Application server A2 and web server W2 are hosted on VM2. The servers D2 and W3 are hosted on VM3. VM3 runs on Linux environment (Table 1). Further another layer of virtualization connects these virtual machines to physical servers is shown in Figure 2- (c). While VM1 and VM2 are hosted on S1, VM3 is hosted on S2.

The workload on the system can be represented in two ways - session based or request based. Whereas session based representation is more aggregated and dependent on the chosen time epoch, request based representation is more granular. In here, we take the more granular approach i.e. request based representation of workload. Further, requests are of multiple types differing due to their intended purpose. From a modelling standpoint, the difference between requests reflects on the path traversed by them within the system. A simple ping request may not go beyond web layer, while a search request on an ecommerce site may go all the way till the database layer. Each request will have different resource requirements at each layer. In some cases, requests spawn into multiple child requests that would later retreat back to the parent request after achieving their intended purpose.

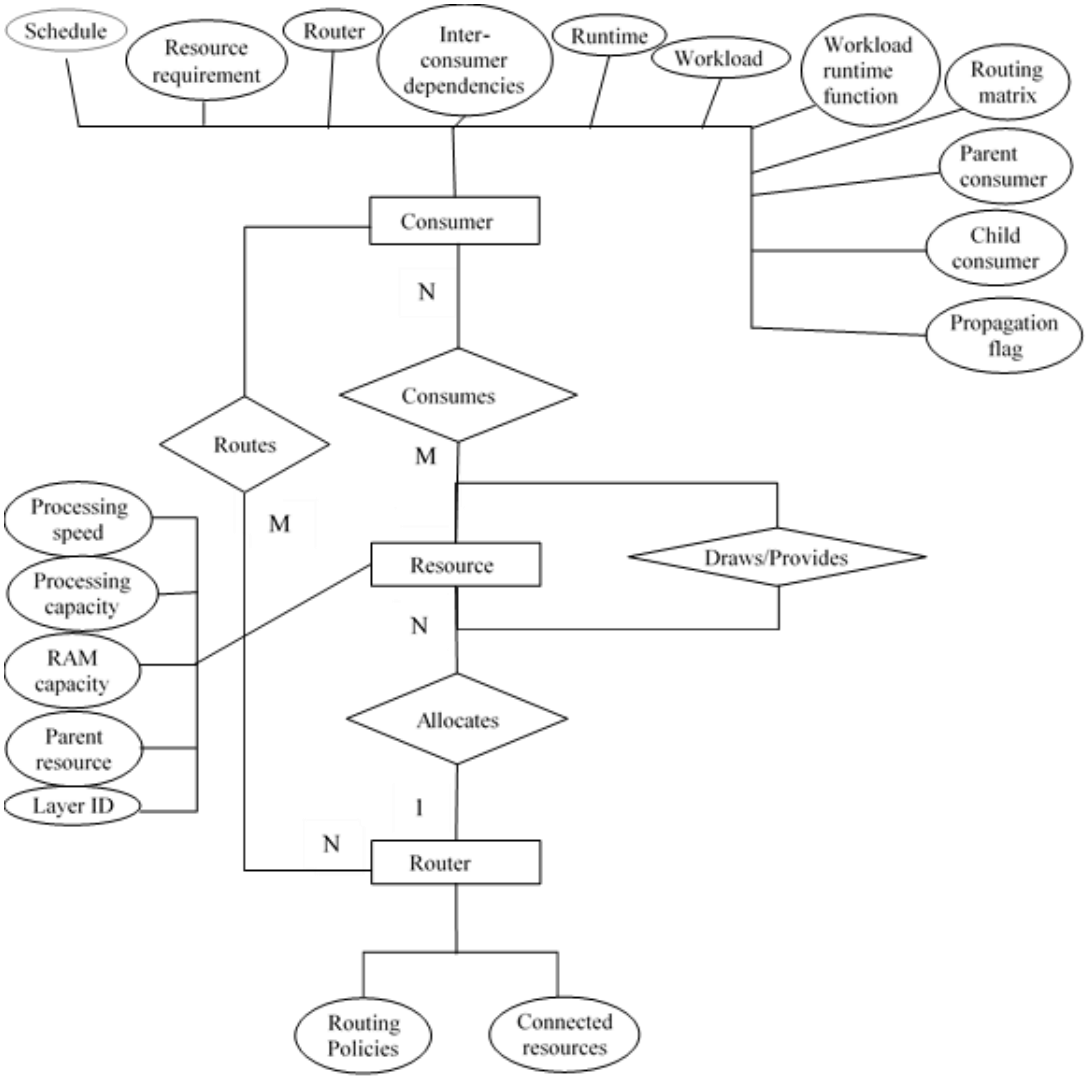


Figure 1. Entity Relationship Model

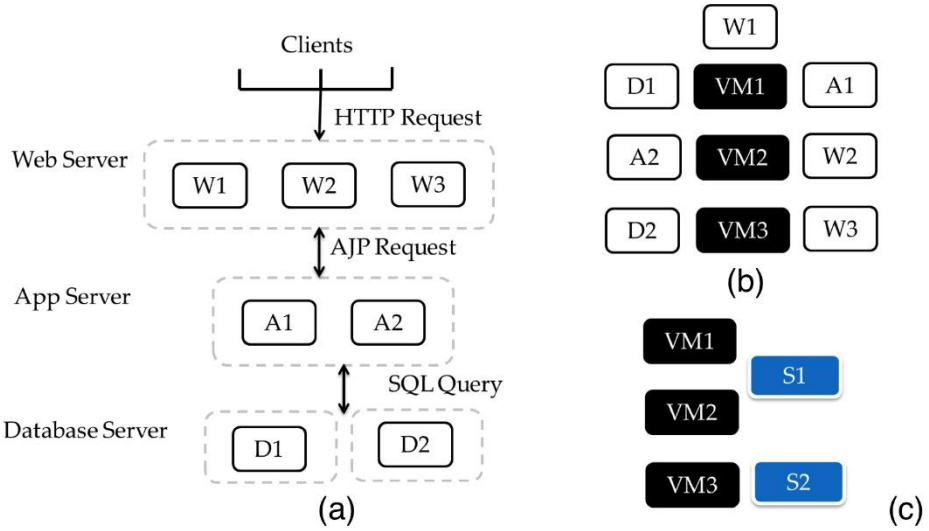
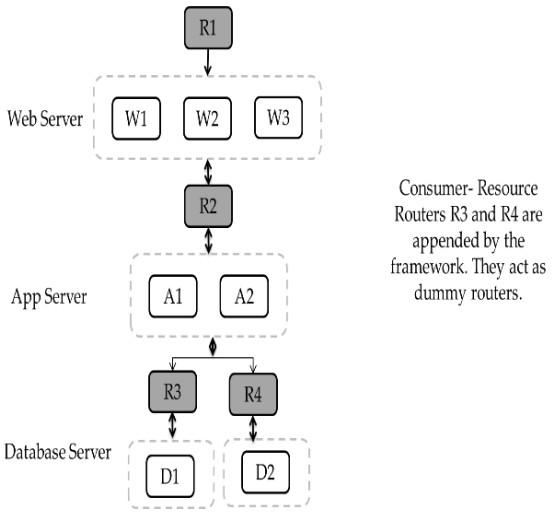


Figure 2. Multi-tier system setup

Table 1: The Servers' setup

Server(s)	Description
W1,W2,W3	Apache, Windows (W1&W2) and Linux (W3)
A1, A2	Tomcat, set up on Windows
D1, D2	Database Servers, setup on Linux D1 and D2 are not replica servers
VM1, VM2, VM3	Virtual machines, VM1 and VM2 are on Windows, VM3 is on Linux
S1 and S2	Physical servers

This actual system is realigned as per the proposed framework (Figure 2). As seen in figure 3(a), Load balancers on Web and App layers are represented by Routers R1 and R2. Both these routers are of Consumer-Resource kind. The consumers, requests in this system, are dynamically routed to one of the servers in the layer based on an active policy. Due to the presence of replica servers on Web and App layers, hence load balancers are required for routing requests amongst servers. However, the database layer has D1 and D2 which are not replicas and are there for different purposes. Despite this, the framework supplements a dummy router in front of both the servers. The framework assumes that every server is succeeded by a router. In this case, R3 and R4 are the dummy routers that route every consumer directly to the resources beneath them.



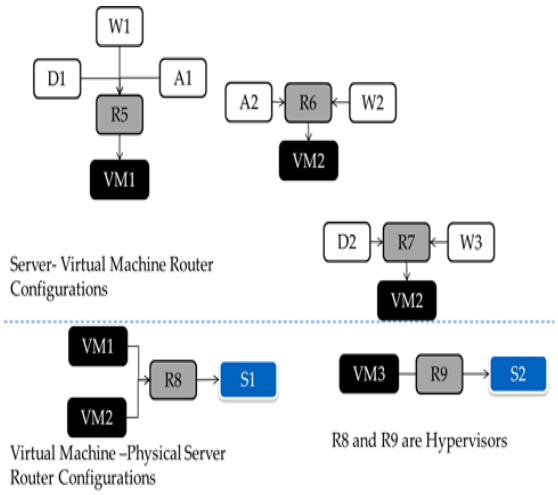


Figure 3a: System re-aligned as per framework **Figure 3b: Server level realigned system**

Also present in the system are Resource-Resource routers which dynamically provision capacity to the servers above them by drawing capacity from the servers beneath them. Figure 3(b) shows the hierarchy of servers and the routers in between for the multi-tier system in discussion. R5 draws capacity from the Windows based virtual machine VM1 to provision for W1, D1 and A1. Similarly, R6 and R7 connect A2, W2 with VM2 and D2 and W3 with VM3 respectively. At the next level of hierarchy, VM1, VM2 are hosted on S1 through the hypervisor R8 and VM3 is hosted on S2 through the hypervisor R9.

Another key aspect of modelling multi-tier systems is replicating the path of consumers/requests (R_i). We do so by augmenting our queuing network with a subsystem modelling that replicates the request path by a matrix with inter layer movement probabilities. Each layer is equipped with an infinite queue (T_i) which means that bottlenecks are not propagated backwards across layers. At each layer, a request can continue its forward propagation with a probability p_i or return to the previous tier with a probability

1-p_i. Once a request starts treading backwards, it would continue backward propagation till it is out of the system as given in figure 4.

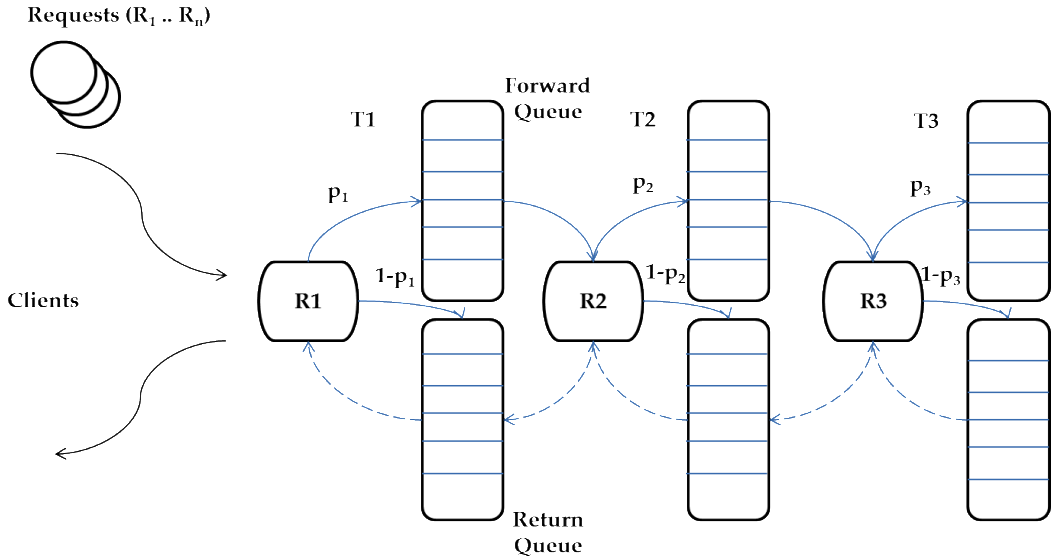


Figure 4: Consumer Routing Probabilities across Layers

Consumer Evolution

To illustrate let's see how a consumer C1 (request) traverses through the simulator. As seen in step 1 of figure 5, C1 enters the system by pinging router R1, which is a load balancer for the web layer. R1, based on its routing policy, allots C1 to one of W1, W2 and W3. Further, based on routing probabilities, C1's next stop is determined. It could either be propagating forward to the App layer or bounce back to the client post processing on one of the servers in web layer. At web layer, C1 keeps hold of a server chunk for a given amount of time based on its workload requirements. In the next stage, based on whether it is an asynchronous or synchronous request, C1 keeps a part of server's capacity or just moves on to the next server. In the case where C1 holds a chunk of server, a child request is generated (C11). C11 at App layer follows a similar process like C1 at the Web layer. However, at the database layer, where the request may potentially be in need of data from both D1 and D2 servers, C11 spawns into C111 and C112, routed to D1 and D2 servers respectively. From a simulation point of view, aggregate variables such as routing probabilities, distributions that reflect the load on database servers dictate the movement and generation of consumers. The backward propagation is fairly straightforward. Child consumers roll back to their parents and the process iterates till one consumer (request) remains. The consumer is then routed out of the system. As discussed previously, the server capacity is also dynamically determined based on the capacity of various routers between each server hierarchy.

Application Use Case

A key challenge of Enterprise IT administration is predicting potential bottleneck scenarios (Urgaonkar et al., 2005). A basic definition of bottleneck is, in a pipe like setup, a bottleneck is said to exist when the outflow is lesser than the inflow. The task of predicting bottlenecks is further complicated in systems with multi-tier architecture due to characteristics such as concurrency across layers, lack of homogeneity (processing times, capacity etc.) across tiers and the presence of a number of bottleneck prone zones. In addition, the complex physical and virtual separation of resources makes it harder to identify the source of bottleneck even if a bottleneck situation is identified.

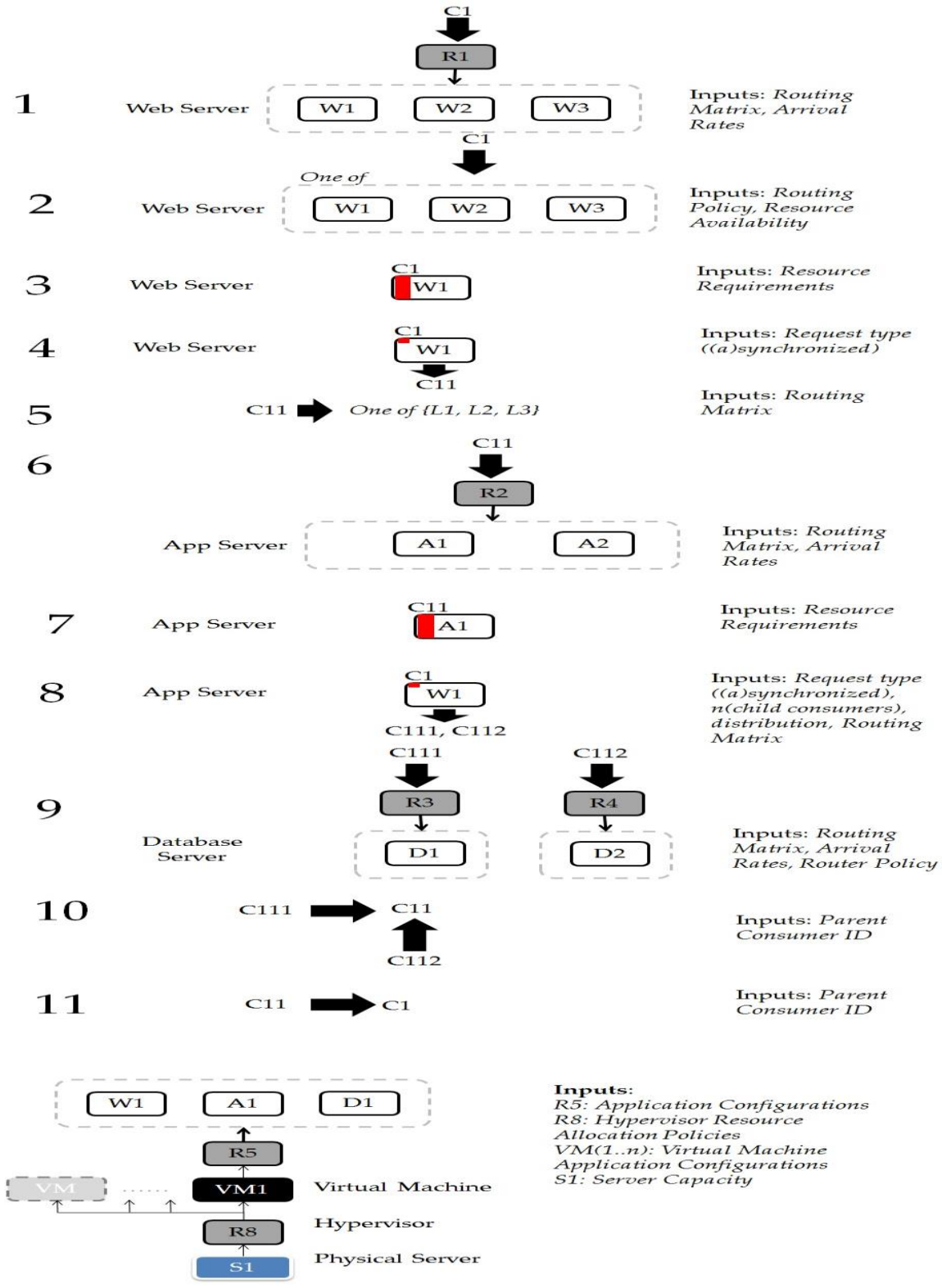


Figure 5: Consumer evolution

The complexity in bottleneck prediction can be divided into two sub problems – predicting the workload at which bottleneck occurs and locating the bottleneck. Typically, testbeds are subjected to varying workloads to identify breaking points in the system. As suggested earlier, building and maintaining a

testbed is an expensive proposition. The simulation framework proposed in this paper can provide a cost effective alternative to such testbeds.

Once the actual system is modelled using the framework, the first step is to stage the simulation model with varying workloads. During this process, we collect the data of different metrics. Barring the basic set of metrics such as processing times, queue lengths etc., the selection of metrics is drawn from the set of levers available to the system administrator. The workload variation allows the user to derive correlation between changes in workload and system performance which is also constrained by the SLAs imposed on the system. While this process of staging workloads is done on testbeds or actual systems as well, they take a different route hereafter to minimize the cost that comes from redoing the process. They instead train machine learning classifiers on the accumulated metric data and save them into a model. This model is then used for predicting bottlenecks. In our approach, we pick a metric that reflects the system performance and compare the same metric at a previous workload in the series of varying workloads. For example, we identify a potential bottleneck when the slope of the sojourn time of the request at a workload becomes k times greater than the slope of the expected sojourn time at the previous workload. By drawing correlations with metrics segregated layer wise, bottleneck can also be located in the process.

A key aspect of simulations is tuning the model to replicate an actual system. Broadly, there are 3 criteria to validate the behaviors of simulations - (1) temporal precedence of the cause, (2) covariation between the presumed cause and effect, and (3) the need to rule out alternative explanations (Cook and Campbell, 1979). Calibration is a very elaborate process that ensures that model formally positing the causal link between structure, captured in terms of equations and parameters, and behaviour, the simulated output generated by the interaction of the equations and initial conditions (Oliva, 2003). To do so, it is essential to capture system specifications at multiple levels. This is a challenge in itself. Although hardware (resource) specifications are relatively easy to source, data on consumers' resource usage at each layer is difficult to gather. System administrators use IT infrastructure monitoring tools such as Nagios to observe servers, applications, switches, routers and other components of Enterprise IT systems. However, configuration and maintenance of such tools is an expensive proposition. For illustrating the framework and the Use Case in this section, we have subjected a multi-tier data centre with varying workloads. Requests are categorized into three kinds – high, low, medium, based on their resource requirements.

The system is subjected to varying workloads as shown in figure 6a. The queue lengths at each layer are shown in figures 6 - b, c, d, e and f. While figures 6 - b,c,d show the requests propagating forward, 6-e and f are for requests waiting for resources on their way out. The queue length indicates the number of requests waiting for cores at any given time epoch. Predictably, the queue lengths at all layers are showing trends of exponential growth as the system is subjected to increasing workload. The processing capacity and resource requirements at each layer determine the length of each queue.

From these patterns, using the approach described above to determine the occurrence of bottleneck. The forward queue at layer 2 first experiences a bottleneck situation. The saw toothed pattern seen in the layers 2 and 3 at later stages shows the evolving nature of bottlenecks. While the first occurrence as per the SLA definitions is in layer 2, bottleneck reverse propagates to layer 1 there by regulating the request inflow of layers further down. The pattern is a composite of a ramp function and a saw tooth function. The saw toothed nature of request queue at layer 3 is more pronounced whereas the ramp nature is more visible at layer 2. The oscillatory and evolving nature of bottlenecks is the reason behind this behavior.

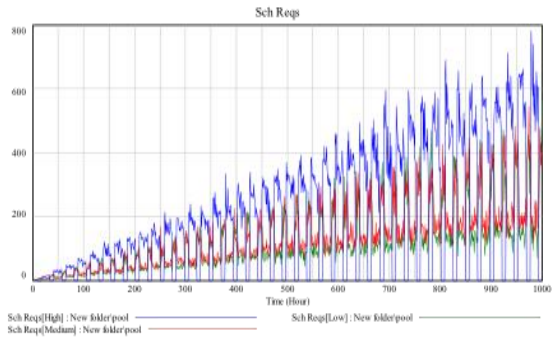


Figure 6a: System workload

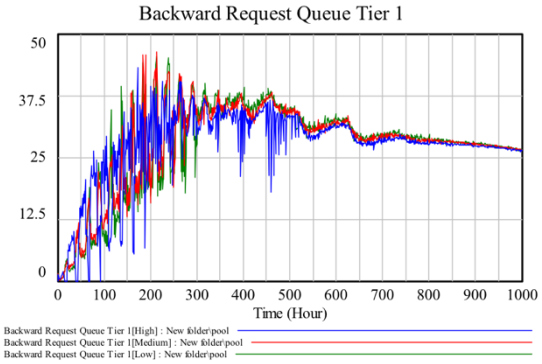


Figure 6b: Request queue at layer 1

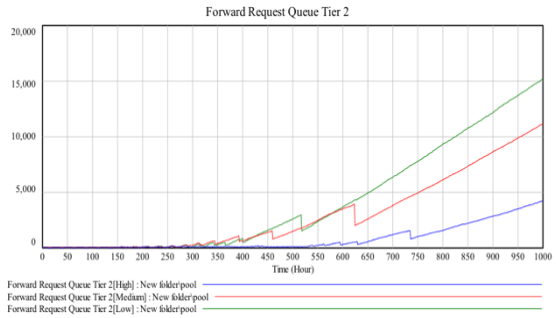


Figure 6c: Request queue at layer 2

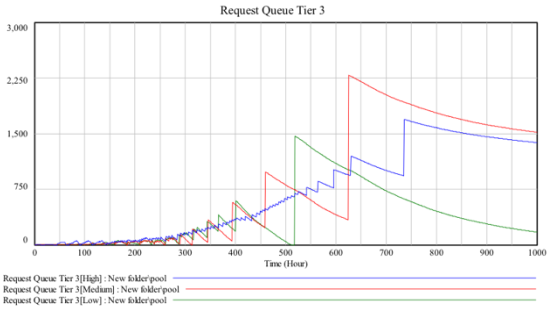


Figure 6d: Request queue at layer 3

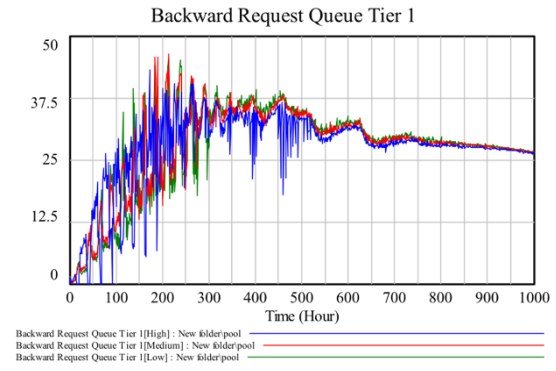


Figure 6e: Request queue at layer 1

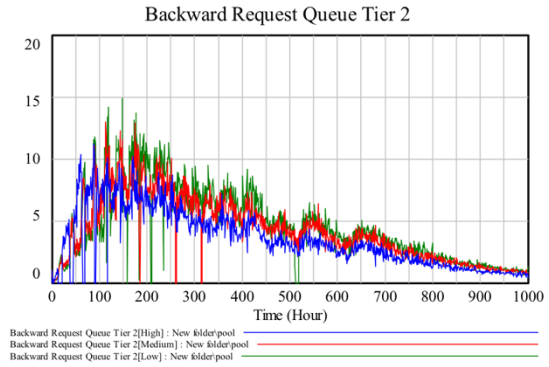


Figure 6f: Request queue at layer 2

Conclusion

In this paper we discussed a framework for modelling and simulation of multi-tier systems. We showed the entity view of multi-tier system consisting of three basic categories- consumers, resources and routers. We established a multi-tier system set up consisting of web layer, application layer, database layer, servers and virtual machines. We also discussed consumer routing probabilities across layers and sequence-based and request based representation of workload on the system. Consumer evolution wherein we trace the traversal of a request across layer was depicted and elucidated in detail.

The primary purpose of the modelling approach taken in this paper was to create a test bed for what-if analysis. A use case of bottleneck prediction was demonstrated using our modelling framework. Other What-if scenarios pertaining to capacity provisioning, application configuration and request policing

etc. can also be modelled using this framework. Besides, this framework lets users conduct analysis at different levels of granularity, thereby enabling operability under conditions with limited data. In addition, the bottom up nature of the framework means various policies (scheduling, allocation etc.) of each atomic component in a multi-tier system, such as load balancers and hypervisors, can be evaluated.

References

Abdelzaher, T. F., Shin, K. G., & Bhatti, N. 2002. "Performance guarantees for web server end-systems: A control-theoretical approach. IEEE transactions on parallel and distributed systems", 13(1), 80-96.

Buyya, R., & Murshed, M. 2002. "Gridsim: A toolkit for the modelling and simulation of distributed resource management and scheduling for grid computing. Concurrency and computation: practice and experience", 14(13-15), 1175-1220.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. 2011. "CloudSim: a toolkit for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and experience", 41(1), 23-50.

Casanova, H. 2001. "Simgrid: A toolkit for the simulation of application scheduling", In Cluster computing and the grid. Proceedings. first ieee/acm international symposium on(pp. 430-437). IEEE.

Castane, G. G., Nunez, A., & Carretero, J. 2012. "iCanCloud: A brief architecture overview", In Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on (pp. 853-854). IEEE.

Cecchet, E., Marguerite, J., & Zwaenepoel, W. 2002. "Performance and scalability of EJB applications", In ACM Sigplan Notices (Vol. 37, No. 11, pp. 246-261). ACM.

Chen, J., Soundararajan, G., & Amza, C. 2006. "Autonomic provisioning of backend databases in dynamic content web servers", In Autonomic Computing, 2006. ICAC'06. IEEE International Conference on (pp. 231-242). IEEE.

Cook, T. D., & Campbell, D. T. 1979. "*Quasi-experimentation: Design and analysis for field settings",* Rand McNally

Dumitrescu, C. L., & Foster, I. 2005. "GangSim: a simulator for grid scheduling studies", In Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on (Vol. 2, pp. 1151-1158). IEEE.

Elnozahy, M., Kistler, M., & Rajamony, R. 2003. "Energy conservation policies for web servers", In Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems-Volume 4 (pp. 8-8). USENIX Association.

Garg, S. K., & Buyya, R. 2011. Networkcloudsim: "Modelling parallel applications in cloud simulations", In Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on (pp. 105-113). IEEE.

Hsu, C. H., & Feng, W. C. 2005. "A power-aware run-time system for high-performance computing", In Proceedings of the 2005 ACM/IEEE conference on Supercomputing (p. 1). IEEE Computer Society.

Kliazovich, D., Bouvry, P., & Khan, S. U. 2012. "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers", The Journal of Supercomputing, 62(3), 1263-1283.

Lim, S. H., Sharma, B., Nam, G., Kim, E. K., & Das, C. R. 2009. "MDCSim: A multi-tier data center simulation, platform", In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on (pp. 1-9). IEEE.

Liu, J., Poff, D., & Abali, B. 2009. "Evaluating high performance communication: a power perspective", In Proceedings of the 23rd international conference on Supercomputing (pp. 326-337). ACM.

Oliva, R. 2003. "Model calibration as a testing strategy for system dynamics models", *European Journal of Operational Research*, *151*(3), 552-568.

Rusu, C., Ferreira, A., Scordino, C., & Watson, A. 2006. "Energy-efficient real-time heterogeneous server clusters", In Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE(pp. 418-428). IEEE.

Schuldt, H. 2009. "Multi-tier architecture", In Encyclopedia of database systems (pp. 1862-1865). Springer US.

Stewart, C., & Shen, K. 2005. "Performance modelling and system management for multi-component online services", In Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation-Volume 2 (pp. 71-84). USENIX Association.

Tang, M., Lee, B. S., Yeo, C. K., & Tang, X. 2005. "Dynamic replication algorithms for the multi-tier data grid", Future Generation Computer Systems, 21(5), 775-790.

Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., & Tantawi, A. 2005. "An analytical model for multi-tier internet services and its applications", In ACM SIGMETRICS Performance Evaluation Review (Vol. 33, No. 1, pp. 291-302). ACM.

Villela, D., Pradhan, P., & Rubenstein, D. 2007. "Provisioning servers in the application tier for e-commerce systems", ACM Transactions on Internet Technology (TOIT), 7(1), 7.

Wickremasinghe, B., Calheiros, R. N., & Buyya, R. 2010. Cloudanalyst: "A cloudsim-based visual modeller for analysing cloud computing environments and applications", In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on (pp. 446-452). IEEE.