A Multi-Level Adaptable Components Framework for Unified Networked Environment

P. Sivaraj

Abstract— For service providing systems, incoming service request rate at a server is a crucial parameter. Given that Internet traffic is bursty in nature and it is not usually well shaped by the time it arrives at a server, it is possible to encounter extremely heavy traffic load at a server particularly at a peak time. Coping with such traffic loads may imply dropping request packets at the server buffer outright, undoubtedly an egregious solution for both the service provider and its clientele, or alternatively, a "reduced" quality service may be proffered if service requesters are willing to accept such a compromise. Here, the system expected/target behavior is assumed to be predicated by a high level policy, and in order to serve that specific policy well, the system is expected to adapt its behavior to the monitored change in local network traffic intensity at the server site. If reduced quality of service is available to be dispensed to client set without necessarily dropping them out altogether, system functionality may be maintained, if not improved.

Index Terms— Adaptive systems, self-management, autonomic computing, Unified network, Adaptable components, Network, goal policies.

1 Introduction

The available complex software systems can be customized and configured. These options can be used to adapt the behavior of the software components even at run time in response to the change in execution environment. Proposed system (explain!) is to design a website with multiple combinations of model and view components (explain!). This system will have design of the webpage with database representation. The variations in network traffic at the server cite should trigger website automatically to replace its components with other components [1].

Development of this project need clear idea about particular problems, which indeed helps to develop the project with high performance, and resolve some of the challenges in project. One of the main challenges is to determine the impact of adapting a component in system behavior. The other challenge is to combine different components which are developed by different developers of a team and to adapt the same to achieve a specific change in the behavior of the system [2] [3].

The various approaches taken for self-management of such adaptable component-based system are as follows. Component specifications are used to describe various components used in the system. An Adaptation specification is used to describe the available components for adaptation. A Goal policy is used to describe the desired system behavior inputs and produces a set of Adaptation Rules which are scripted in the system that allows it to adapt a specific component to maintain its desired behavior during the runtime [3].

The various approaches taken for self-management of such adaptable component-based system are as follows.

Product image specifications are used to describe the different components used in the system.

An Adaptation specification is used to describe the available components for adaptations.

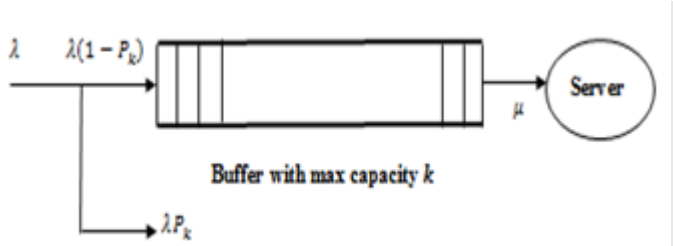
A Goal policy is used to describe the desired system behavior inputs and produces a set of Adaptation Rules which are scripted in the system that allows the system to adapt a specific component to maintain its desired behavior during the runtime.

2 Related Works

Suppose a server of our interest is represented at the network layer level as a system with its Markovian input queue (all requests coming to server are independent, and identically distributed with no remembrance of the quality of the past services they received as individual requests from the server – that is they have lost all the memories of their past service quality) to receive service requests. We assume the mean arrival rate of service request packet to be λ per sec, and its Markovian property does lead to arrival process to be Poisson distributed (inter arrival time to be exponential). Furthermore, we assume that the requests are served at an average rate of μ requests per sec, and the service process is again assumed to be Poisson distributed (in reality, any distribution will be tolerated here with some minor changes in performance outcome) [4] [5].

Furthermore, we assume that the requests are going to be served in FCFS mode. Basically, every operational queue is a finite sized buffer or of finite capacity capable of allowing at any time at most, say, k request packets in it waiting to be served [6]. Therefore, our operational queuing model is an M/M/1/k queue (See Kleinrock, Queuing System) which is diagrammatically shown as follows:

- *P.Sivaraj is currently pursuing masters' degree program in Computer Science Engineering in Sasurie academy of engineering, Coimbatore. E-mail: sivarajmp@gmail.com*



Where the blocking probability is $\lambda P_k = \lambda \rho^k \frac{(1-\rho)}{(1-\rho^{k+1})}$ with traffic intensity $\rho = \frac{\lambda}{\mu}$. The probability that the system is busy (system utilization factor)

$$U = 1 - P_0 = 1 - \frac{1-\rho}{1-\rho^{k+1}} = \rho \frac{(1-\rho^k)}{(1-\rho^{k+1})}$$

In the event of very high traffic flow, we have ρ→1, and using L´ Hospital rule, we conclude that

$$\lim_{\rho \to 1} U = 1 - \frac{1}{k+1} = \frac{k}{k+1}$$

This is extremely high for a large buffer size k.

The average number of customer seeking and getting service in the system, N, i.e. the system load comes out to be

$$N = \frac{\rho}{1-\rho} - \frac{(k+1)\rho^{k+1}}{(1-\rho^{k+1})}$$

And the very first term in it explodes as ρ→1. The same thing is seen happening in the system response time equation

$$R = \frac{N}{\lambda(1 - \frac{(1-\rho)\rho^k}{1-\rho^{k+1}})}$$

As the traffic intensity ρ climbs up to 1.

Obviously, the only solution(s) to this problem is to scale down the service rate μ at the server, and/or cut down services which are time consuming. For example, the loading time of the following photo in full RGB spectra would be considerably larger

As, it requires handling a total pixel map of volume 768×1024×3 of 2359296 bytes. One can cut down the image loading time by requiring only a reduced dimension of, say, 480×640×3 pixel volume, shrinkage of roughly 0.39. Or, if one prefers, one could upload, only the grayscale image instead as shown below,

Required, uploading about 93734 bytes in this case. Here, the shrinkage is about 96%, a considerable advantage to a server, if it is required to pack on its requested page a lot of such RGB pictures [7].

Also, often enough servers are required to provide central tendency of some stored data to requesting users. The data may be too large to download, but the server could provide its distribution characteristics in a 'frugal' mode if response time of a server is a way too high.

For example, let us suppose a given dataset 'Cement' is given, wherein the amount of heat evolved (in Cal) is given by a dependent variable y with 4 independent variables x_1,x_2,x_3 and x_4 values assume all arranged in a table format. If the dataset is large, it may not be feasible to download the data from the server directly; instead, it might be a better strategy to ask for its summarization [8].

One way to summarize it would be to provide some statistics showing data behavior as shown below.

```
> summary(cement)
       x1              x2              x3              x4              y
 Min.   : 1.000   Min.   :26.00   Min.   : 4.00   Min.   : 6   Min.   : 72.50
 1st Qu.: 2.000   1st Qu.:31.00   1st Qu.: 8.00   1st Qu.:20   1st Qu.: 83.80
 Median : 7.000   Median :52.00   Median : 9.00   Median :26   Median : 95.90
 Mean   : 7.462   Mean   :48.15   Mean   :11.77   Mean   :30   Mean   : 95.42
 3rd Qu.:11.000   3rd Qu.:56.00   3rd Qu.:17.00   3rd Qu.:44   3rd Qu.:109.20
 Max.   :21.000   Max.   :71.00   Max.   :23.00   Max.   :60   Max.   :115.90
>
> cor(cement)
           x1         x2         x3         x4          y
x1  1.0000000  0.2285795 -0.8241338 -0.2454451  0.7307175
x2  0.2285795  1.0000000 -0.1392424 -0.9729550  0.8162526
x3 -0.8241338 -0.1392424  1.0000000  0.0295370 -0.5346707
x4 -0.2454451 -0.9729550  0.0295370  1.0000000 -0.8213050
y   0.7307175  0.8162526 -0.5346707 -0.8213050  1.0000000
>
```

For each column, we may get data characteristics defined over 6 attributes (min, max, median, mean, first quantile and the third quantiles). Would that be good enough? At times, when the response time of the server is reasonably low, perhaps one might consider this as a standard data analysis chunk. The correlation matrix as shown above shows pairwise correlation between the variables. For instance, in this case the variables x_2 and x_4 are highly correlated, and their effects on the dependent variable y (with correlations 0.8162 and -0.8213) does suggest a linear model like y=a_0+a_2 x_2-a_4 x_4. It would be real nice to get that model while one is with it in its first attempt, but it might prove to be too costly [8] [9]. If the server is really busy, perhaps, just the summary statistics would be good enough; one could get the rest incrementally.

The issue is: how to reduce the service time of a server that affects its response time without scaring its users away. It is a difficult problem that has to be tackled. If nothing is done to the response time of a popular server, it would lose its appeal. If, on the other hand, its array of service is constantly offered at a lower rate, again it might lose its clients. If a client continuously receives poor quality service from a server, perhaps the server has to be redesigned.

3 Brief Analysis of Adaptation System

This framework allows the system to identify the behavior and performance of the application in the internet, if any network traffic or problems in loading components may then it automatically identify the application performance and replace the corresponding system components in it to provide same effectiveness of the application to users.

3.1 Adaptation Policy

Designing and creating adaptation policies for components in system is one of the critical tasks in the autonomic system it is difficult to define a policies which can configure automatically based on the any given situation. To avoid these situation we can use control theory and rule based expert systems it provides possible solutions to our techniques, however both of them has some disadvantages. We can use hybrid approach which provides modeling and optimization offline to generate suitable configuration, at the same time it helps to encode policies in our application which are used by the application at runtime. This technique and approach is examined in the given issue of dynamic management in virtualized consolidated server environments that host multiple multi-tier applications [9].

Here we can use novel hybrid approach which helps us to improve autonomic behavior of the system in the both server and client side. To predict system behavior and to generate optimal system configuration automatically, we can use this queuing theoretical models in this system. However, instead of manufacturing these configurations on demand, they're made offline to feed a decision-tree learner that produces a compact rule set or adaptation policy that may be employed in rule engines directly, audited, combined with alternative human-produced rules, or just accustomed aid domain specialists in writing and maintaining management policies. This approach of manufacturing entire call rule sets offline has another profit in addition, the modeling answer and improvement is entirely far from the vital path of the system throughout runtime. Therefore, it's attainable to model and optimize ever larger and a lot of advanced systems.

Although the approach is general, we have a tendency to focus here on the matter of expeditiously allocating resources in consolidated server environments. Server consolidation through virtualization technologies is more and more seen as some way to cheaply meet the large demands of area, hardware, and energy that square measure generated by fashionable multi-tier enterprise systems.

By hosting applications on virtual machines, resources are often shared between applications at an awfully fine grain (e.g., CPU cycles). This raises vital challenges, together with handling the immensely completely different responsiveness and performance needs of the multiple applications and managing dynamic modifications in resources demands because the application workloads change. Basically, the management question is a way to provision all the applications to maximize the utility provided whereas taking into consideration service level agreements (SLAs), resource availableness, and workloads.

Our overall approach is made public within the below Figure. The rule set generation method is driven by the rule set creator. It generates a collection of candidate works and invokes the configuration optimizer to see the most effective configuration Copt for every candidate workload W. It then passes these workloads and associated configurations through the decision-tree learner from the wood hen toolkit to get the rule sets [10].

For every work W passed thereto, the optimizer searches through the complete configuration area for the utility increasing configuration. To cypher utility, it invokes the model thinker that for every candidate configuration c and work W, provides AN estimate of the system interval and therefore the C.P.U. utilization of every system part. The C.P.U. utilization data helps the optimizer verify if the configuration is viable on the on the market resources or not. The model thinker uses bedded queuing models to predict mean response times and utilization. The response Times Square measure accustomed cypher overall utility. The queuing models parameters square measure computed victimization an automatic method in AN offline coaching part. This section describes every part well, beginning with the queuing models.

3.2 Application Modeling

Regular queuing networks are used in the multi-tier web application; here we choose layered queuing models as the basis of our work in this system. The reason is that in consolidated server environments with fine-grained electronic equipment management and multiple applications, models got to be correct over a large vary of workloads, high utilizations, and even in configurations that may be terribly unbalanced in terms of resource allocation amongst tiers. Thus, block development that aren't vital in well-provisioned environments, e.g., a bottleneck attributable to the block of front-end computer code threads by an extremely overladen back-end server, should be expressly sculptured. In contrast to customary queuing models, superimposed queuing networks alter such modeling by permitting multiple resources to be consumed by asking at a similar time [9].

Net: Represents the latency introduced by the network. Since we assume that the network uses a pure delay server and not a bottleneck (i.e., no resource sharing). The service time is measured using ICMP ping measurements in the native environment of the system.

Disk: Represents the delay due to disk I/O. To measure the service time transparently, we wrapped each component with an interception library using the LD_PRELOAD environment variable. The library intercepted each read and write call made by the application to compute the mean number of I/O calls a disk and their service time.

Component: (Apache, Tomcat, and MySQL). Represents the processing performed by the software component. The task is modeled using an M/M/n queue, where n is set to the number of maximum software threads each component is configured for (or1in the case of MySQL, which creates threads on demand). The threads execute on a CPU queue with the processor sharing discipline (to approximate time slice OS scheduling).

Using the Servlet.jar we can measure the service time and number of calls for these servers transparently, we instrumented the Servlet.jar file that is used by every application based on Java servlets using binary rewriting programs in the system. Using the instrumentation timestamps we can easily identify the each incoming request from clients and response to the web server, at the same time each request to and response from the database server. Additionally we can measures end to end response time for the entire request and response from the client and server. We can perform this experiment with only one user at a time, before that we need ensures that no queuing delay is present in the current system, and the measurements at each server can be correlated with others. As same as their values in the native environment along with the disk I/O service times are sufficient to compute the service times for each component [10].

VMM: By using VMM in Xen environment we can find the interaction delay induced by it. Due to it dependent on Xen and no other application, the service time for the task have to be assumed equal across all machines. By computing the difference between the service time of each component with the VMM task and without the VMM task we can estimate the time. VMM service time is computed by using knowledge of the measurement points and how many times the VMM was included in each measurement.

Client: By using client the workload for the queuing model is generated. During run time the instantaneous rates of individual transaction can be measured, the set of |Ti| independent open Poisson processes is the workload for each application [ai], one for each transaction type. This allows the model to any mix of transaction types in the workload.

3.3 Rule-Set Construction

In our approach the highest-level component is the rule set constructor. A set of candidate workload WS is randomly generated by this component by using each application which has the highest allowed request rate. For each workload W 2 WS, it will find the best configuration copy (W) by invoking the optimizer. Each host contains a list of nodes where each configuration is encoded as a linear list of physical host. The node that listed inside the host contains the name of the application followed by the name of the name which it belongs and the CPU capacity allocated to it.

The points in the rule-set construction will form a partial "raw rule set" for each of the candidate workload. Workloads which are not evaluated as a part of candidate set can be applied in a rule which is also included inside the complete rule set, hence some form of interpolation is needed. By using the J48 decision tree learner we can generate a final rule set with help of the Weka machine learning toolkit. The form w to a threshold or w to a threshold at each of its branches is the condition under the generated decision tree, where "w to a" is the request rate for transaction t of application a. The path that leads from the root of the tree to that leaf is the conditions that will be satisfied when the lead of the tree encodes the configuration [11].

Using decision tree we can construct multiple functions in the system which helps us to make the decision easily in adaptable components. The rule set is needed for the interpolation so decision tree will provide this first with any workload in the range allowed by the SLA where it is not applicable to the points evaluated by the optimizer. Then for Second, a nested "if-then-else" rule set is obtained from linearizing the decision tree due to it require less expertise to understand than the models that generated in it. Third, all the configurations that included in the systems are known before development due to there is finite number of leaves in the decision tree. For business-critical systems we gathered the knowledge of degree of predictability and verifiability from this. Finally, by learning algorithms aggregate portions which share similar configurations and prune outliers we get the raw rule set table which provided by the tree.

Due to the fact that during run-time and consequence of compaction, the larger sets of points in the systems are used to generate the tree, some loss of accuracy is expected. The severe of loss is evaluated in the next session with a modest number of training points and accurate rule sets are constructed.

3.4 Automating process

Self-configuration: The system will automatically set its configuration according to the needs specified by the user to achieve the high-level goals.

Self-optimization: The system will optimize itself when it meet high level requirement where it simple be seen as dynamic self-configuration.

Self-healing: If any error occurs the autonomic system detects those errors and repairs by itself. This make the system to tolerate any kind of errors occurs in it.

Self-protection: The systems have to be well protected against malicious attacks and from the errors make by the users, without making any change inside the configuration.

3.5 Dynamic Resource Allocation

In Dynamic resource allocation will triggered in workload management for DBMS to do workload reprioritization. Immediate resource reallocation to the workload will be get by dynamically adjusting the workload according to the priority.

In DBMS performance management the main key factors such as buffer pool memory pages and CPU shares are the two shared system resource which considered in this study. Multiple workloads can be run concurrently by the DBMS, where the workloads are classified in different business importance classes with unique performance objectives. According to the business importance level in the workload it will get a certain amount of the shared resources. More resources are assigned to the workloads which have high importance and low resources are assigned to the workloads which has low importance while low importance [11].

Based on an economic model the resources are allocated. To sell the shared system resource the DBMS conducts "auctions", and the resources are bought by the workloads submit "bids" and biding based trade mechanism. Business importance level in the workload is reflected by the Virtual "wealth" is assigned to all the workloads. More wealth is assigned to the workload which have high importance than low importance ones.

Based on the three components of resource model, resource allocation method and performance model, the dynamic resource allocation approach defined in the system. The first

component resource model is helps to allocate the resources and helps to identify the available total amount of the resource to allocate in the system. We using separate buffer pool to assign the every single competing workload at the same time buffer pool memory pages are directly allocated to the workload [9].

Resources that are used in the CPU cannot directly assigned to a workload because resources are used bulk in the system so we need to partition CPU resources by monitoring the number of database agents that are available to serve request from the database server. In this project, we going to use a DB2 DBMS which help us to handle one client request from the database and configuring one database agent to maintain all connection in the server. By examining, the relationship between the databases that are available in the internet and database agents along with CPU utilization of a workload. We can easily identify the more database agents available to define the serve request for a particular workload and the CPU resource workload bandwidth we can easily measure the system performance. The available total amounts of resources are parameters in there source allocation approach, so it can adapt to different system configurations.

There are many numbers of methods and techniques are available for resource allocation which helps us to determine how to identify an optimal resource pair in a buffer pool memory pages and CPU shares for a workload in the system to increase the benefits of the workload performance. However workload needs to collect resources in appropriate amount which cannot define the bottleneck resource of the system. This technique provides greed algorithm which helps to identify resource preferences of a workload in the resource allocation process of the particular system [10].

This resource allocation process is continuously repeated in the system until we get appropriate results. First we need to know about the iteration of the algorithm, we can use its virtual wealth, a workload bids for a unit of the resource (either buffer pool memory or CPU) that it predicts will yield the greatest benefit to its performance. The starting node, n1,1, represents the minimum resource allocation to a workload, namely one unit of buffer pool memory and one unit of CPU, at the beginning of a resource allocation process. The workload then traverses the directed weighted graph to search for the optimal <cpu, mem> pair in order to achieve its performance objective.

3.6 Adaptation selection example

- Response time: low, medium, or high
- Quality: graphical or textual
- Budget: under or over

Note that these three stakeholder objectives suggest three corresponding attributes that are important to select an adaptation.

- Response time: 1 if low, 0.5 if medium, 0 if high
- Quality: 1 if graphical, 0.5 if unchanged, 0 if textual
- Budget: 1 if under or unchanged, 0 if over
- Disruption: 1 if 1, 0.75 if 2, 0.5 if 3, 0.25 if 4, 0 if 5

3.7 Modeling adaptation logic

In this technique, the behavior of the adaptable components of a system results from the dynamic reconfiguration of the service composition associated to the channels used by the core layer of the system. We need to design and develop a rules for the high level adaptation policies to support in the runtime adaptation of the application with the help of these service compositions at a high level of abstraction. These policies helps to the system to locate when and how the service composition is integrated with each channels that are reconfigured in term of logical view of channels, and services, and service compositions. The choice of those policies was driven by the variation necessities of previous systems designed victimization the protocol composition framework.

The specification of associate degree adaptation policy uses components represented in different models:
The service model, that gives a logical description of the services that offered and might be utilized in service compositions.

The channel model, that gives a logical description of the channels whose service compositions are often adapted.
The context model, that describes the context data needed to outline the things during which adaptation is required.

To boot, modeling the difference logic of associate degree application conjointly involves the definition of associate degree application model and a detector model, as explained within the next sections.

4 ALGORITHMS

Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlap the user's purchased and rated items. The algorithm aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user.
Two popular versions of these algorithms are

- Collaborative filtering
- Cluster models.

Other algorithms — including search-based methods and our own item-to-item collaborative filtering — focus on finding similar items, not similar customers. For each of the user's purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them.

4.1 Traditional Collaborative Filtering

A customer is represented as an N-dimensional vector of items by using traditional collaborative filtering algorithm, where N is the number of distinct catalog items. Vectors have the components which are positive for positively rated items and negative for negatively rated items.

By using the inverse frequency the algorithm typically multiplies the vector component to compensate for best-selling items, it is more relevant when making less well-known items. For almost all customers, this vector is extremely sparse.

4.2 Cluster Models

By dividing the cluster model in to many segments we can find the customers who are similar to the user and treat the

task as a classification problem. Assigning the user to the segment containing the most similar customers is the main goal of this algorithm. Recommendations are generated with the help of using the purchases and ratings of the customers in the segment.

By using a clustering or other unsupervised learning algorithm the segments are created and manually determined segments are used by some applications. Clusters or segments are form by grouping the similar customers together with the help of the clustering algorithm groups. Optimal clustering is impractical over large data sets so various forms of greedy cluster generation are used by most applications. Initial set of segments in this algorithms contain one randomly selected customer each. By the existing segments they match the customers to itby using some provision for creating new or merging existing segments.

Sampling or dimensionality reduction is necessary for very large data sets which has high dimensionality. The algorithm generates the segments; the segment with the strongest similarity is choosing to computes the user's similarity to vectors that classifies the user accordingly to it. Some algorithms classify users into multiple segments and describe the strength of each relationship. Rather than comparing the entire customer base with the user to a controlled number of segments the clusters models will get better online scalability and performance than collaborative filtering. Computation of the complex and expensive clustering is run offline even when recommendation quality is low. Numerous customers are group together into a segment by Cluster models, it will match a user to a segment, and for making recommendations it will consider all similar customers in the segment. Because the similar customers found by the cluster are not the most similar customers, the recommendations they produce are less relevant. By using numerous fines grained segments it is possible to improve quality, but finding similar customers using collaborative filtering is expensive as then online user–segment classification.

4.3 Search-Based Methods

Search-based methods are used to search for related items that have the recommendations problem. The algorithm helps to find the items in the given user's purchased and rated items by constructs a search query by the same author, artist, or director, or with similar keywords or subjects.

Search based recommendation algorithms perform and scale well if the user has few purchases or ratings. It is impossible to frame query if a user purchased thousands of items. Subsets of data and reducing quality have to be used in the algorithm. Recommendation quality is relatively poor in all cases. New, relevant, and interesting items are discovered and find by a customer with help of recommendations. There is a chance of fail to achieve the goal by the items if it has the same author or in the same subject category.

5 Implementation

Spring MVC framework is used to design the system and it helps to secure and watch the flow of the system clearly. At the same time the functionalities of the system is developed in the java programming language. Java is high level programming language developed by sun microsystem. It is an object oriented system provides more user friendly function.

In mvc-dispatcher-servlet.xml all the initialization of spring is done in the system to create an application. Need to create corresponding tables for the application. If the table creation is done then we need to establish connection between MySQL to application, for calculating the number view pages and the rules for displaying the components has been scripted in the mvc-dispatcher-servlet.xml. Securing the user personal information and unwanted situation in the application user login is create it provide more security for application and user, this page is configured using the security-config.xml. In web.xml all the servlet details and their locations are defined. Java Server Pages are used for designing the web pages because it provides more functionalities and attractive design to the application. Using the mvc-dispacher-servlet.xml to view object and to implement the Model View Controller framework in spring concept.

Jetty server is used for initializing the project, at the same time connection between the system and the MySQL is also configured. To run my application first we need to start the jetty server, once the jetty server is started then the welcome page will be moved to a local host and the same will be loaded in the web browser. Controller is used to check all pages by providing the corresponding functionalities and execute the model, all the web pages are in the view part of MVC, it will automatically return to the view and print the relevant output whether the application process is complete successfully or fail upon the reason of something.

jMeter is the application developed by the apache foundations. jMeter is used for the purpose to check the web page modifications if the number of users accessing the website increase. jMeter will get the input of web page addresses and its input details and it will run and will display the graph result and the table list for our confirmation

Suppose the accessing of website crosses the CPU memory of above thirty percentages the resolution of the image will automatically change to the lower size image. After running the jMeter, if we refresh the web page, we can view the changes in the adaptable component which will be displayed in the webpage.

6 Simulation and Other Results

6.1 System Design

This section give brief instruction about my application design, here I choose a better technique that help to my application select adaptable product image while the changes may occur on the system in web page retrieval time. My application make changes in the web page based on the number of requests (i.e. execution state) and a high level policy will be drawn which decides the behavior of the system. KPI parameter (Key Performance Indicator) and also based on the information provided for each component are used to create impact on the KPI's and if there is any limitation or specific requirements [1] [3] are needed in the system then these things will create automatically.

Consider an example, many users are requesting to view a

particular product image then there may high traffic occur, so the image will be not visible to some of the requested users based on their network plan and coverage of the network. By replacing the high quality product image to a low quality image may yield significant memory, processor, while if it has less number of traffic then the same adaptations may have negligible impact on the system.

6.2 System Overview

Every time in my system automatically check the CPU usage whenever user request to visit the particular product image if the CPU usage is high then the system automatically redirect to which triggers an event and the adaption rule will be selected and applied .Once system CPU usage is higher than normal it is overloaded then the best suitable component will be selected and will presented to the user. If it is in normal state there will not be any change in the behavior of the system [4] , [5].

7 Conclusion

The proposed methodology, which is used to manage all the behaviors; which are to be adapted in a complex software system. As specified earlier, this technique highly relies on the data provided by the developers who have created the components through the Key Performance Indicator values. The rules are then created which is used to evaluate the adaptive behavior of the system in offline and online phases. The results which are examined during the implementation suggest that the approach evaluates how far the current state is deviated to an optimal state and also how large the impact is on each and every component load which is used to estimate an adaption impact.

In future we can create an application that support for two different videos about the product quality and with the audio support. This functionality provides more advantages to my system. Additionally we can provide user interface to change in the normal and high traffic, using more attractive user interface takes more time to load in the server in the high traffic. So we can create two different user interfaces to adopt based on the CPU usage and the situation.

ACKNOWLEDGMENT

I would like to extend my sincere thanks to all people mentioned below who have helped me in carrying out this research. I record my deep sense of indebtedness and whole hearted gratitude to head of the department, Mrs. N. Poongothai M.E., Assistant Professor and Head, Department of Computer Science and Engineering, Sasurie Academy of Engineering, Coimbatore, for her active involvement, encouragement, caretaking and valuable suggestions in all the way and also shaping the research work and this report. I am highly indebted to Ms. R. Meenatchi M.Tech, Assistant Professor, Department of Computer Science and Engineering for her support, guidance and constant supervision during this work.

I thank all my classmates for their kind cooperation and encouragement.

References

[1] G. Jung, K. Joshi, M. Hiltunen, R. Schlichting, and C. Pu, "Generating Adaptation Policies for Multi-Tier Applications in Consolidated Server Environments," Proc. Int'l Conf. Autonomic Computing, pp. 23-32, 2008.

[2] S.-W. Cheng, D. Garlan, and B. Schmerl, "Architecture-Based Self-Adaptation in the Presence of Multiple Objectives," Proc. Int'l Workshop Self-Adaptation and Self-Managing Systems, pp. 2-8, 2006.

[3] L. Rosa, A. Lopes, and L. Rodrigues, "Modeling Adaptive Services for Distributed Systems," Proc. 23rd ACM Symp. Applied Computing, pp. 2174-2180, 2008.

[4] M. Couceiro, P. Romano, and L. Rodrigues, "A Machine Learning Approach to Performance Prediction of Total Order Broadcast Protocols," Proc. Fourth IEEE Int'l Conf. Self-Adaptive and Self- Organizing Systems, pp. 184-193, 2010.

[5] M. Couceiro, P. Romano, and L. Rodrigues, "Polycert: Polymorphic Self-Optimizing Replication for In-Memory Transactional Grids," Proc. 12th ACM/IFIP/USENIX Middleware Conf., vol. 7049, pp. 309-328, 2011.

[6] R. Grieco, D. Malandrino, F. Mazzoni, and D. Riboni, "Context-aware provision of advanced internet services," in PerCom Workshops, 2006, pp. 600–603.

[7] T. Abdelzaher and N. Bhatti, "Web content adaptation to improve server overload behavior," in WWW8 / Computer Networks, 1999, pp. 1563–1577.

[8] F. Mazzoni, "Efficient provisioning and adaptation of webbased services," PhD Thesis, Universita di Modena e Reggio Emilia, 2006.

[9] G. Iaccarino, D. Malandrino, and V. Scarano, "Personalizable edge services for web accessibility," in Proceedings of the 2006 International Cross-disciplinary Workshop on Web Accessibility. ACM, 2006, pp. 23–32.

[10] S. Souders, "High-performance web sites," Commun. ACM, vol. 51, no. 12, pp. 36–41, 2008.

[11] Bandara, E. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," IEEE International Workshop on Policies for Distributed Systems and Networks, vol. 0, p. 229, 2004.