A Capacity Planning Methodology for Distributed E-Commerce Applications

I. Introduction

Most of today's e-commerce environments are based on distributed, multi-tiered, component-based architectures. The inherent complexity of such architectures makes it extremely difficult for system deployers to estimate and plan the size and capacity needed to provide the desired service levels. This document presents a methodology for carrying out capacity planning studies of large distributed e-commerce applications. The methodology walks the capacity planner in a step-by-step fashion, through the process of analyzing the performance of e-commerce systems and determining the right sizing and capacity of the deployment environments.

The main steps of the methodology are:

1. **Characterizing the Business Case**
2. **Functional Analysis**
3. **Characterizing the User Behavior**
4. **Characterizing the IT Infrastructure**
5. **Cost Modeling**
6. **Characterizing the Workload**
7. **Performance Model Development**
8. **Performance Prediction**

The following models are used:

1. **Business Model**
2. **Functional Model**
3. **User Behavior Model**
4. **Resource Model, consisting of:**
   - **IT Infrastructure Model**
   - **Cost Model**
   - **Workload Model**
   - **Performance Model**

The methodology is depicted in Figure 1.

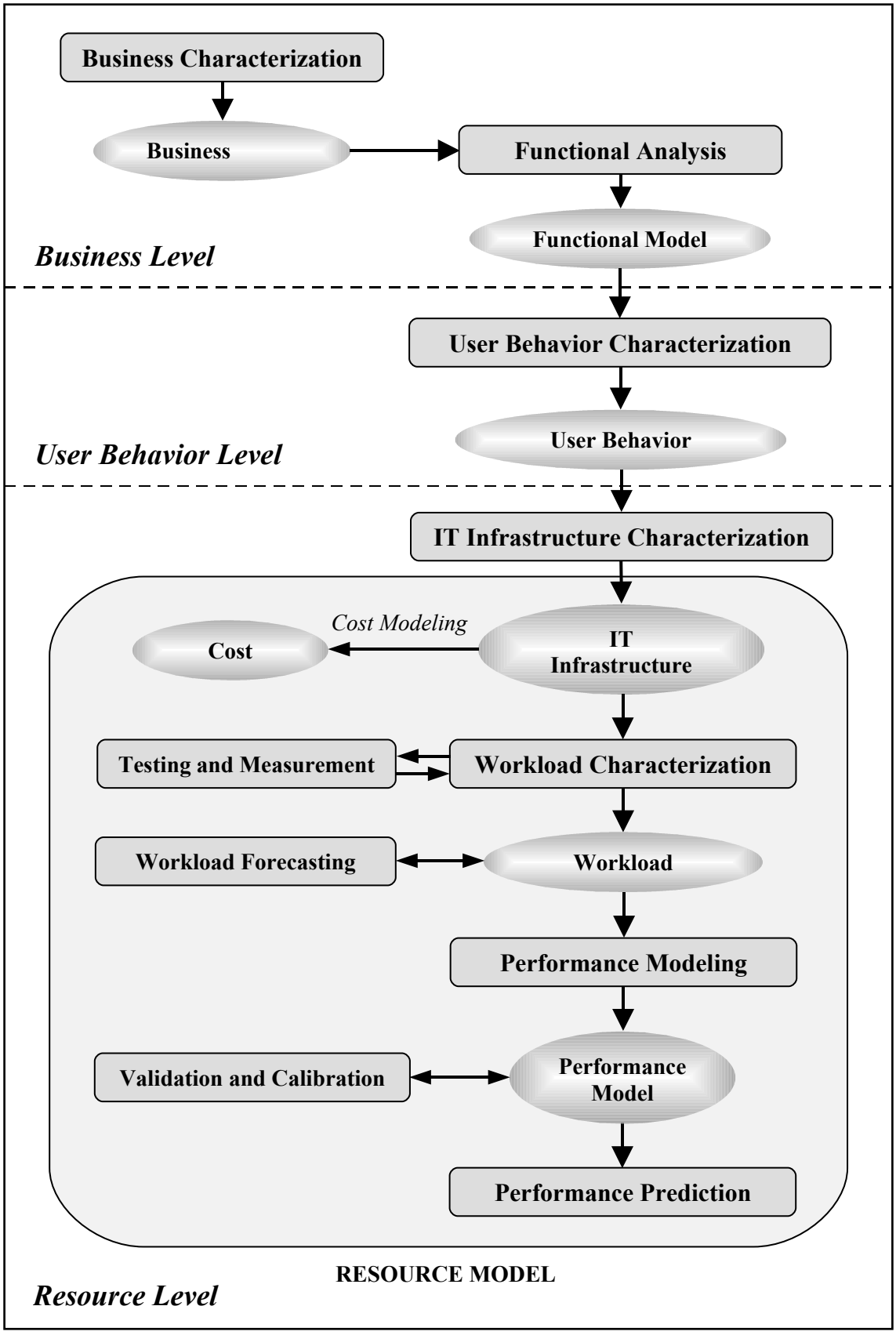

Figure 1: A Capacity Planning Methodology for Distributed E-Commerce Applications

II. The Methodology

1. Characterizing the Business Case

The Capacity Planning Methodology starts with the **Business Characterization** step. This step aims to provide a general description of the business case, focusing on the main services provided and the business processes and activities needed to support these services. The result of the business characterization is the **Business Model.**

2. Functional Analysis

The next step is to perform a **Functional Analysis** of the business and build a **Functional Model** of the system. The functional model describes in detail all functions that are provided by the e-commerce system. It is important to note that this analysis covers not only functions seen by customers, but also any internal functions used for management and administration purposes. During this phase one should also identify the different types of users that will use the functions provided by the system. Here again not only external customers should be taken into account, but any users (e.g. management staff) that use directly or indirectly the functions of the system.

In essence, the business and functional models yield a *use case view of the system*. The use case view captures the *actors* (the entities which interact with the system) and the *use cases* (the units of functionality) of the system.

Here are some examples of functions that might be provided by an e-commerce system:

- Browse
- Login
- Logoff
- Customer Registration
- Customer Profile Modification
- Customer Search
- Search Connections
- Offer Creation
- Travel Reservation
- Payment (method selection)
- User Authentication
- Ticketing

3. Characterizing the User Behavior

The next step is the **User Behavior Characterization**. The purpose of this step is to build a **User Behavior Model**, which describes the user behavior when using the system services. More specifically, the user behavior model captures the e-commerce functions used during a session, the frequency of access to the various functions, the navigational patterns and the times between access to the various services offered by the system. The user behavior model can be useful for the workload characterization phase later in the capacity planning process.

The User Behavior Model typically consists of a number of so-called **User Behavior Diagrams**, which show the behavior of different users when using the system services. Figure 2 below shows an example of a UBD. As we can see the diagram depicts the different states in which the user can be found during a *session* (a sequence of consecutive requests issued by the user) and the possible transitions between states. Depending on the availability of data the UBD should also include the transition probabilities between the states. This would make it possible to compute the average number of times the user sends a request for a particular e-commerce function during a session. This information could be especially useful later in the workload characterization step.

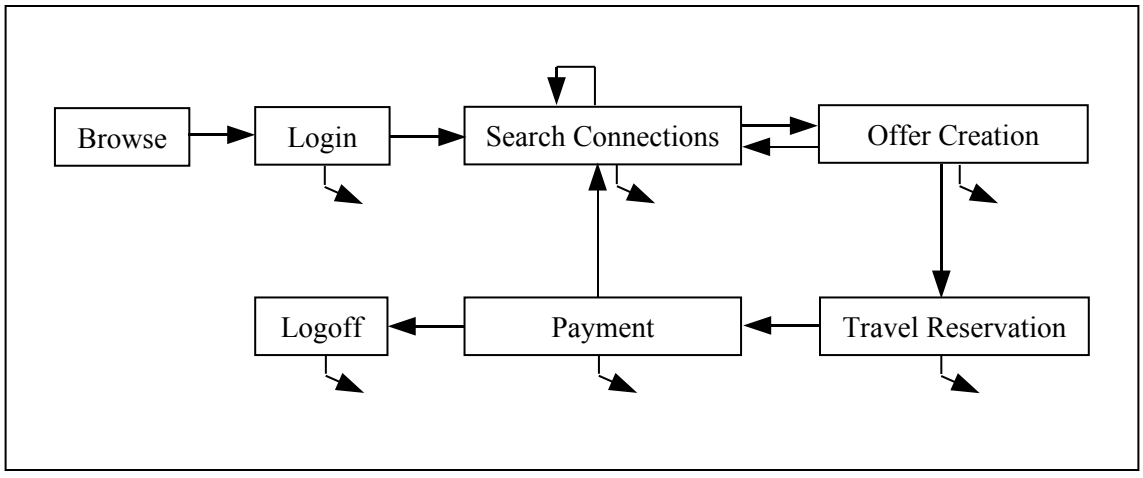

Figure 2: An example User Behavior Diagram (UBD)

4. Characterizing the IT Infrastructure

Once the user behavior is analyzed and a user behavior model is built, the next step is to characterize the IT infrastructure. This step generates an IT infrastructure model which describes in detail the hardware and software resources used to implement the system. More specifically, the infrastructure model should specify things such as:

- The technological architecture of the e-commerce system – tiers, communication links, etc.
- Hardware platforms used - server machines, disk subsystems, load balancers and other hardware resources, etc.
- Software platforms used (e.g. operating systems, web servers, application servers, database servers, DNS servers, LDAP servers, TPMs etc.)
- Communication equipment used – networks, routers, firewalls, switches, etc.
- Communication protocols used

In essence, the purpose of the IT infrastructure model is to break down the system architecture into **components** (e.g. web server, application server, database server) and then describe in detail each of those components in terms of the hardware and software platforms used. Also the communication resources (networks, routers, etc.) used for inter-component communication must be specified. Figure 3 depicts a possible IT Infrastructure of an E-Commerce System.

5. Cost Modeling

The IT infrastructure model provides the basis for building a **cost model** of the system. The cost model is used in order to estimate the costs that would be incurred in setting up the production environment, deploying the e-commerce system and running it on a day-to-day basis. Note that in the first iterations of the capacity planning process, information about the sizing of the production environment is usually not available. Therefore one should at this point concentrate on the costs of the different components (servers, networks, etc.), which make up the IT infrastructure, without taking into account the sizing (e.g. number of web servers). Later when the system is sized appropriately one should come back to the cost model and make some more concrete cost estimations. Both startup and operating costs should be taken into consideration.

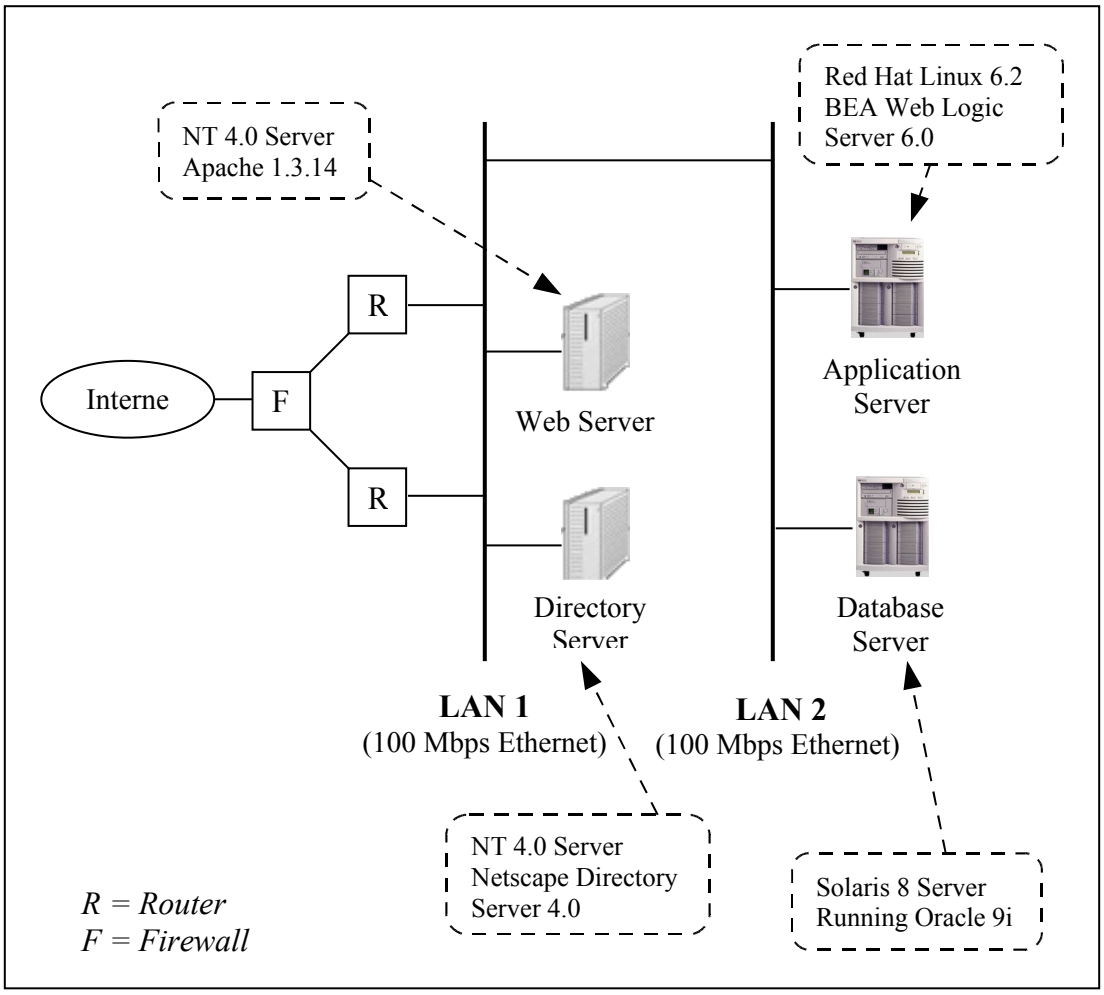

Figure 3: A typical IT Infrastructure of an E-Commerce System

6. Characterizing the Workload

One of the most important steps in the capacity planning process is the **Workload Characterization** step. The purpose of this step is to precisely describe in a qualitative and quantitative manner, the global workload of the e-commerce system.

Following is a list of the main steps in the workload characterization process:

- *Determine which Components are used by each E-Commerce Function*
- *Identify the Transactions at the Component-Level*

- *Describe the Inter-Component Interactions for each E-commerce Function*
- *Identify the Physical Resources used at the Component-Level*
- *Measure Workload Intensity and Service Demands for each Transaction type*
- *Measure Service Demands at the Network Resources*
- *Summarize measurements and partition the Workload*
- *Characterize the Workload at the Global Level*
- *Forecast workload evolution*

We will now try to describe in detail each of these steps.

6.1 Determine which Components are used by each E-Commerce Function

The characterization process starts by taking each function from the functional model of the system and determining the IT infrastructure components (described in the IT infrastructure model) involved in its execution. The relationship between e-commerce functions and components used is documented in a matrix as illustrated in Table 1 below.

E-Commerce Function	Web Server	Directory Server	Application Server	Database Server
1. Browse	√			
2. Login	√	√		
3. Logoff	√			
4. Customer Registration	√	√		√
5. Customer Profile Modification	√	√		√
6. Customer Search	√	√		√
7. Search Connections	√		√	√
8. Offer Creation	√		√	√
9. Travel Reservation	√		√	√
10. Payment (method selection)	√		√	√

11. User Authentication		√		

Table 1: E-Commerce Functions vs. Components used

6.2 Identify the Transactions at the Component-Level

The next step is to identify the transactions, which are involved in the execution of each e-commerce function. Note that we use the term **transaction** to refer to any *generic unit of work* executed at a particular system component. These transactions are often called *basic components* of the workload. Following are a few examples:

- HTTP request served by a web server
- LDAP request served by a directory server
- database transaction executed at a database server
- remote method invocation (or in general a remote procedure call) executed at an application server

For each e-commerce function one needs to specify its transactions and the components where they are executed. Table 2 below illustrates this.

E-Commerce Function	Transaction	Component
Customer Registration	AddCustomer	Web Server
	RegisterCustomer	Directory Server
	StoreCustomerProfile	Database Server
	SendConfirmation	Web Server

Table 2: Transactions Associated with the Customer Registration Function

6.3 Describe the Inter-Component Interactions for each E-commerce Function

Once transactions are identified and documented, one should proceed to describe in detail the internal interactions between the system components involved in the execution of each e-commerce function. Different notations can be used to document these interactions. In the example shown in Figure 4 we use the so-called **Client-Server Interaction Diagram (CSID)**.

A separate CSID should be created for each e-commerce function. The diagram depicts the flow of control during the execution of the e-commerce function and the order in which transactions are executed. The diagram also shows which communication networks are used and the average sizes (depending on whether such

information is available) of the messages exchanged between the components. The latter are shown in the square brackets over the arrows.

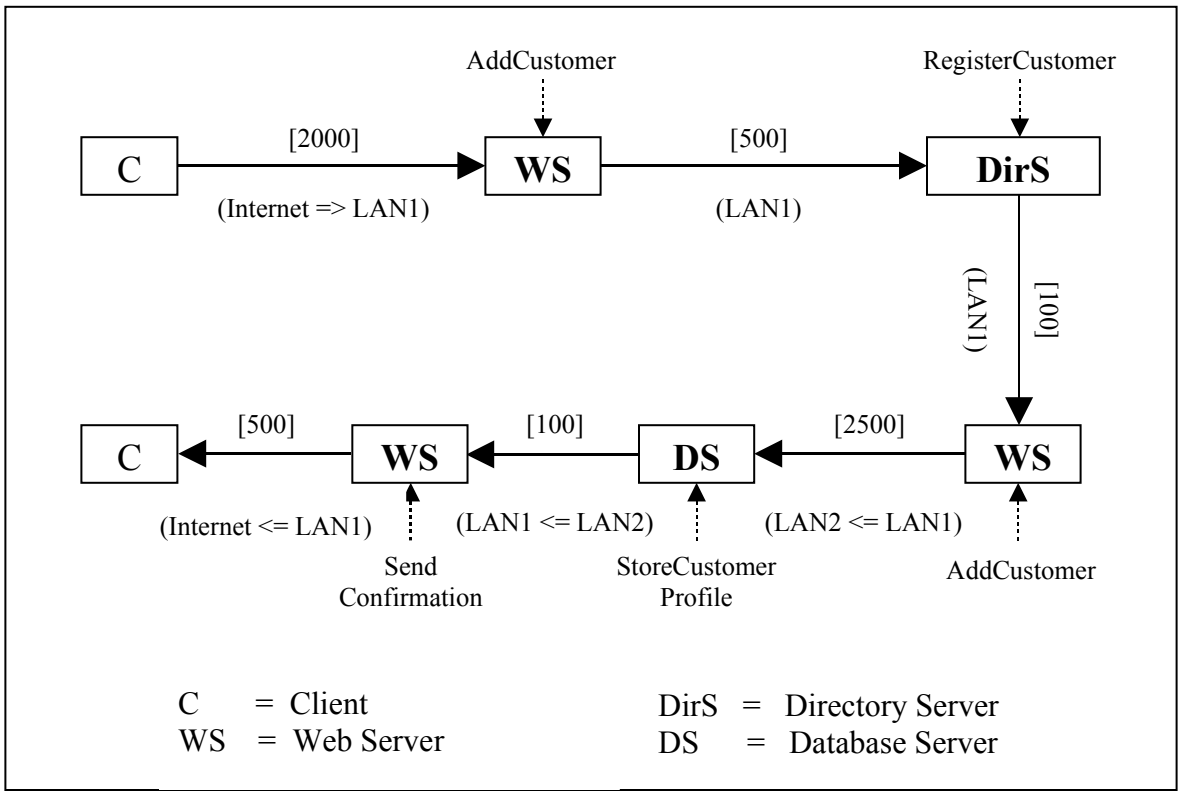

Figure 4: Example of a CSID for the Customer Registration Function

6.4 Identify the Physical Resources used at the Component-Level

The next step is to identify the physical resources, which are used at the component-level in order to execute a particular transaction. For example, each database transaction uses the CPU, the disk subsystem and some memory of the machine on which the database server is running.

6.5 Measure Workload Intensity and Service Demands for each Transaction type

Once resources are identified one needs to quantify each transaction type in terms of its usage of physical resources at the component level. More specifically, transactions need to be characterized by *workload intensity* and *service demand* parameters.

Workload intensity parameters give an indication of the number of transactions that contend for use of the physical resources. Two common ways to specify the workload intensity of a transaction are the following:

- specify the average transaction arrival rates (the number of arriving transactions per unit time)
- specify the average number of transactions in execution simultaneously

The **service demand** of a transaction at a resource is the total amount of time spent by the transaction receiving service at the resource. Note that a particular resource can be visited multiple times during the execution of a transaction. The service demand is the total service time over all visits to the resource during the execution of the transaction.

In order to obtain the workload intensity and service demand parameters one needs to run some tests on the system and collect performance measurements. Typically *load-testing tools* are used to generate load on the system and measure its performance. In addition to the monitoring facilities provided by load-testing tools one can also use operating system utilities, accounting systems and program analyzers to get some additional resource-usage and performance data. Workload intensity and service demand parameters are obtained either directly from the measurements or derived from other parameters that are measured directly.

6.6 Measure Service Demands at the Network Resources

In addition to estimating service demands at the resources within the IT components one should also estimate the service demands at the various network resources (e.g. LANs, routers, switches, firewalls, etc.) used to exchange data between components. The service demand at a network resource is the total amount of time spent transferring data through that resource during the execution of a request.

6.7 Summarize measurements and partition the Workload

As already noted, the workload intensity and service demand parameters can be obtained by running tests and collecting performance measurements. When the workload intensity is high, large amounts of measurement data can be collected. Working with such data is not practical especially if the workload characterization results are to be used as input data to performance-predictive analytical models. Instead, what is usually done is that data is analyzed and substituted with a more compact summarized representation. Let's consider the following example:

A database server was monitored during a peak period of 1 hour. During this period the CPU time and the I/O time for each of the 5000 transactions executed were measured. Some transactions used very little CPU and I/O, while other more complex transactions used substantially more CPU and I/O. After analyzing the measurements

it was found that 70% of the transactions required on average 3.8 msec of CPU time and 4.4 msec of I/O time. 20% of the transactions required 20 msec of CPU time and 15 msec of I/O time. The last 10% of the transactions required 56 msec of CPU time and 30 msec of I/O time. We can use this summary data as a substitute for the detailed data of the 5000 transactions.

We will use the term **workload class** to refer to a group of transactions (or requests) which are similar in terms of their resource usage. In the above example we partitioned the workload into 3 classes. If the motivation for partitioning the workload was only the compactness of the representation one could equally well use a single class, which represents the average resource consumption over all transactions. The problem is that as we saw in the example real workloads are usually composed of *heterogeneous components*. Therefore one should be careful not to lose the characteristics of the workload and end up with a non-representative model. The ultimate goal of the workload partitioning is to obtain a compact representation, which captures the main characteristics of the workload and breaks it down into a series of homogeneous classes. The resulting representation constitutes the so-called **Workload Model**, which is the end product of the workload characterization step.

6.8 Characterize the Workload at the Global Level

Once the workload is characterized at the component level, depending on the availability of measurement data, one should try to conduct a more general *global workload characterization*. That is, instead of working with parameters (workload intensity and service demands) on a per transaction basis, one should try to obtain some parameter data on a per request basis. Note that by **request** we mean a request for execution of an e-commerce function, which itself might entail the execution of a number of transactions at the various components. The level of detail and accuracy, to which this can be taken, depends highly on the concrete system under study and particularly on the sources of performance data available. The challenge is how to correlate independent measurements taken at the various components of the architecture (e.g. web server, application server, database server) and associate them with the execution of the same e-commerce function. The ultimate goal here is to estimate the average service demands that requests for execution of the e-commerce function place on the physical resources at the different components.

In addition to measuring the workload intensity and service demand parameters, it would be especially beneficial, if one can also measure the total time spent at each component during the serving of a request. This would enable us to break down the system response time in terms of components (web server, database server, etc.) and functions (login, browse, search, travel reservation). Such information can be used to

pinpoint the system components where requests spend most time and isolate potential system bottlenecks.

6.9 Forecasting workload evolution

After characterizing the current workloads on the system, one should also try to predict how workloads are likely to evolve in the future. This is called **workload forecasting** and it is an important part of the capacity planning process. There are two main approaches to workload forecasting - *quantitative* and *qualitative*. The quantitative relies on the existence of historical data, which is analyzed in order to estimate the future values of workload parameters. The qualitative approach is a subjective method, based on intuition, previous experience, expert opinions and other relevant information. Again the level to which workload forecasting can be realized depends on the availability of historical data.

7. Performance Model Development

The next step of the methodology is to build a performance model, which captures the performance and scalability characteristics of the system under study. Models represent the effect of the workload on the system resources and more particularly the delays caused by the contention for hardware resources resulting from increasing load levels. There are different types of performance models which all have their strong and weak sides. The methodology presented here, uses the so-called **analytical models**.

Analytical models are based on a number of formulas and computational algorithms, which are used to model the times that requests spent at the different system resources either receiving service or waiting (queueing) for being served. Such models are usually based on the *theory of queueing networks* and their main use is to predict the performance of a system as a function of the system's description and workload parameters. Basically, analytic performance models require as input the data included in the IT infrastructure and workload models. By applying some mathematical formulas and laws one can estimate performance measures of the system such as average response time, transaction throughput and resource utilization. Performance models can be used to analyze the overall system performance, as well as the performance of individual components.

Once performance models are built, they need to be *validated* in order to make sure the performance metrics calculated through their use match the measurements of the actual system within an acceptable margin of error. This is usually done by comparing the performance measurements taken during the workload characterization phase with the measurements computed by means of the models. If the difference between values is outside the acceptable levels of error, the performance models need to be *calibrated*.

The IT infrastructure model, the cost model, the workload model and the performance model make up the **Resource Model** of the system.

8. Performance Prediction

Once performance models are validated they can be used to predict the performance of the system under different scenarios. For instance, one could be interested to know what would be the performance of the system if the number of web servers were doubled. By modifying the validated model to reflect this change in the IT infrastructure one could predict performance metrics under the new circumstances. Similarly, one could vary the workload intensity parameters and predict system performance under higher workload levels (e.g. what would be the maximum throughput of the web server if the number of "travel reservation" requests increases by 60%).

The results could be used to plan the sizing and capacity of the production environment needed to guarantee that applications would be able to meet future workload levels while keeping performance at an acceptable level.

The quality and the amount of data provided by performance models are highly dependent on the level of detail and the accuracy of the input data provided. Of particular importance is the data included in the workload model, namely the workload intensity and service demand parameters obtained through empirical testing. Making precise measurements is a great challenge and should be taken really seriously, because it is a prerequisite for building models which are representative.

In any case one should keep in mind that any results obtained through the use of models would always be approximate and there is never an absolute guarantee that systems will behave exactly as indicated by the preliminary analysis. Models described in this document help us to estimate the amount of hardware needed to provide enough processing resources. Often though, there are other factors (e.g. contention for software resources, etc.) which might have significant impact on performance and prevent the system from behaving as indicated by the performance models. The best approach to isolate such factors is to load test the system and analyze how performance measurements change as load levels increase.

III. Conclusion

This document presented a methodology for carrying out performance and scalability analysis for the purpose of capacity planning of distributed e-commerce

applications. As already mentioned a number of times, the level of detail and the extent to which this methodology can be realized is extremely dependent on the concrete case scenario. Some of the activities presented here might not be feasible in particular situations, especially when the system is in the early stages of development. It is important to remember that the capacity planning process is an iterative process that continues after system deployment. During each iteration models are refined and updated to reflect the current state of the system.