

Supporting Capacity Planning for DB2 UDB*

Hamzeh Zawawy^{1,2}, Patrick Martin¹ and Hossam Hassanein¹

¹School of Computing
Queen's University
Kingston, Ontario
Canada K7L 3N6

²IBM® Toronto Laboratory
Toronto, Ontario
Canada M3C 1H7

Abstract

Capacity planning is the process of determining the most cost-effective computing environment that meets the current and future demands of a computing system. Capacity planning is important for large database management systems (DBMSs) whose performance can be greatly affected by the amount of available resources and whose workloads can change significantly over time. In this paper, we examine support for capacity planning for DBMSs. We describe an analytical model that is used to estimate performance under different scenarios and discuss how to obtain the input parameters to the model for an OLTP workload running on IBM® DB2® Universal Database™. We then give an example of how the model can be used to solve a typical capacity planning problem.

1 Introduction

Database management systems (DBMSs) such as DB2 Universal Database¹ are key

components of most large organizations' information technology infrastructure and often require significant amounts of both computer and human resources. It is therefore reasonable for an organization to try and provide the most cost-effective configuration for its DBMS. The problem of determining this configuration is made more difficult by the fact that a DBMS's workload, and hence its demand for resources, may change significantly over time.

Computer capacity planning is the process of monitoring and projecting computer workload and specifying the most cost-effective computing environment that meets the current and future demands for resources. Capacity planning estimates the amount of hardware resources, such as CPU, memory, and disks, needed to satisfy a set of performance requirements under a changing workload for a certain application. It allows predicting if and when system saturation will occur [1,2]. Capacity planning can be performed when a system is first deployed in order to determine an appropriate hardware configuration. It can also be performed while a system is operational in order to predict the impact of changes to the workload, the hardware configuration, or both.

A capacity planning study typically consists of the following steps [3]:

* This research is supported by IBM Canada Ltd., the National Science and Engineering Research Council of Canada (NSERC) and Communication and Information Technology Ontario (CITO).

¹ IBM, DB2, and DB2 Universal Database are trademarks or registered trademarks of

International Business Machines Corporation in the United States, other countries, or both.

1. **Understand the current environment, if it exists.**
2. **Build a model of the system.** In our case, we choose to use analytical techniques, specifically queuing network models, to develop the models.
3. **Characterize the system's workload.** We partition the workload into classes based on resource usage. We identify the main parameters of the workload, such as number of clients or transaction rate, and quantify the demands each class places on the system resources.
4. **Validate the model.**
5. **Use the model to predict the impact of new scenarios.**

The goal of our research is to develop a tool to support capacity planning for DB2 Universal Database. We envision a tool that can be used by DBAs to help configure new systems and reconfigure existing ones. In this paper, we focus on the development of the analytical model that will form the basis of the tool. We then study the performance of DB2 to find relationships between performance values and the parameters. We use mean value analysis (MVA) [1] to solve the model and obtain performance indices for various configurations.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 describes a simple queuing network model to represent DB2 and outlines preliminary experiments that we performed to validate this model and to characterize the demand of each type of workload transaction from resources. Section 4 gives an example using our capacity planning approach. Section 5 draws some conclusions and suggests some future work.

2 Related Work

Many references on capacity planning discuss issues such as performance evaluation, workload characterization, and the financial aspects of capacity planning [1, 2, 3, 4, 9, 11]. Books, such as Lazowska [1] and Ferrari [11],

emphasize the techniques and tools developed in performance evaluation. These tools include hardware and software monitors, simulation, analytical modeling, and benchmarking. Lam [3] examines the theory and practices of computer capacity planning in a more rigorous manner than previous studies. Menasce [2] shows how mathematical modeling can give clear representation of the real world. Menasce [9] presents concepts and formulas for effectively planning Web-based systems. Through the use of many real-world examples, Menasce shows how to use quantitative methods to analyze Web-based applications.

3 Model Development

The main contribution of the paper is the development of a queuing network model of DB2 that can be used to carry out capacity planning studies. We perform the first four steps in a capacity planning study, which were described above, to develop the model.

3.1 Understanding the System

We chose to run experiments with two different computer systems to ensure the independence of our model from any hardware configuration. Since some of the designed experiments need to be run over systems with multiple CPUs, multiple disks, or multiple disk controllers, or a combination of these, we have chosen computer systems with these features.

The first system used in the experiments is an IBM Netfinity® 5000 with dual 400 MHz Pentium® II processors with 512 KB cache, 1 GB of RAM, and 4 hard disk drives. The second system is an IBM POWERserver® 704 with a single 200 MHz Pentium processor, 1 GB of RAM, 4 disk controllers, and 16 hard disk drives. Both systems run with the Microsoft® Windows NT® operating system Version 4.0 and DB2 Universal Database Version 7.0.

We model a transaction in terms of its use of three main resources, namely, CPU, main memory, and disks. We only consider the buffer area to be the main memory resource in this study. We characterize a transaction as a series of logical

page accesses. A logical page access corresponds to a request for a new page from a database object, that is, a user-defined table or index. Each logical page access receives an amount of service from the CPU, from main memory and, if the page requested is not in the buffer area, from the disks. The probability that a logical page access involves a disk access is given by the miss rate of the buffer area.

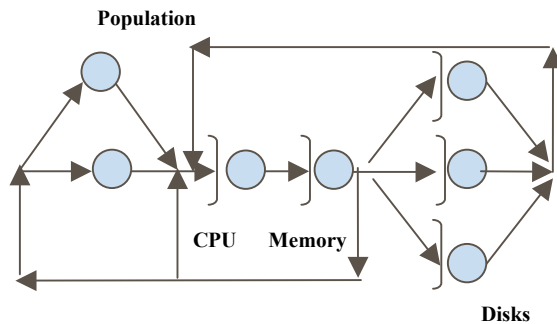


Figure 1: DB2 Queuing Network Model

3.2 Building the Model

A queuing network model represents the behavior of transactions in the system. Figure 1 shows a queuing network model that represents a DB2 system on a computer with a single CPU, 3 disk drives, and a number of users concurrently using the system. A queue is associated with each resource. A client generates a transaction, which is composed of a set of logical page accesses. Each page access first receives its service from the CPU and then from main memory. If the access is a miss in the buffer area, then the transaction receives service from a disk and then loops back for the next logical page access. If the access is a hit on the buffer area, then the transaction loops back for the next logical page access or leaves the system. We assume the DBMS is run with its maximum number of clients and so use a closed model.

3.3 Workload Characterization

We use the TPC-C online transaction processing (OLTP) benchmark [7] to generate the

workload for this study. We assign each type of transaction in the benchmark to be a workload class in the study since each transaction type has a different behavior and different service demands on the various resources.

In a capacity planning study, the level of the basic component of the workload applied to the queuing network model must first be defined. For a DBMS, the level of the basic component of the workload could be high-level batch jobs, which are sets of business transactions, individual business transactions, individual SQL commands or even CPU instructions. Since users typically set the target performance goals at the business transaction level, we choose business transactions to be our basic workload component. This choice makes the translation of user performance requirements and goals into our model a straightforward one-to-one mapping.

We used the Windows NT performance monitor to collect data about the system. The parameters that are of interest to us are as follows:

- **% Processor Time:** The percentage of non-idle processor time spent servicing DB2.
- **% Disk Time:** The percentage of elapsed time that the selected disk drive is busy servicing read or write requests.
- **Commit Statements Attempted:** The total number of SQL COMMIT statements that were attempted.
- **The total run time:** The time during which the driver was run against DB2.

Before it can complete, each transaction needs a certain amount of service from each resource (CPU, memory, disk). The service time needed from each resource, called the service demand, varies depending on the class of the transaction. To compute the service demand for a certain class of transactions from a resource, we use the following utilization law:

$$Demand = \frac{Utilization * Time}{Completions} \quad (1)$$

where *Demand* is the service time a class of transactions requires from a resource, *Utilization* is the percentage of time this resource is busy serving this class, *Time* is the total time the experiment is run, and *Completions* is the number of completed transactions of this class..

We ran preliminary experiments to characterize the resources and to study the behavior of the service demand of resources when the buffer and the population sizes vary. The population size refers to the number of users concurrently using the system or the number of transactions that exist in the system at a certain time. A 90% confidence level with $\pm 10\%$ confidence intervals was compiled for each data point.

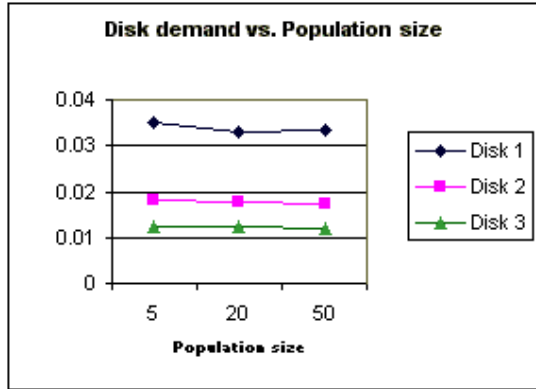


Figure 2: Disk demand vs. Population size

3.3.1 Disk Demand vs. Population Size

An increase in the population size leads to longer queue lengths at all resources. By definition, the service demand does not include the queueing time; therefore, it is not affected by the variation in the population. Figure 2 shows the independence of the disk demand from the population size. The minor discrepancies observed in the graph are within the confidence intervals and are considered statistically insignificant.

3.3.2 Disk Demand vs. Buffer Size

The disk demand depends on the number of disk accesses, which in turn depends on the

buffer size. We can therefore deduce that the disk demand is highly dependent on the buffer size. In order to find a simple mathematical relationship between the two parameters, we use a well-known equation from Belady's virtual memory study [6] that relates the hit rate (H) to the buffer size (M):

$$H = 1 - a * M^b \quad (2)$$

where a and b are constants.
By definition,

$$H = 1 - \frac{P}{L} \quad (3)$$

where P is the number of disk accesses and L is the number of logical page accesses. We can then say:

$$\frac{P}{L} = a * M^b \quad (4)$$

Rewriting equation (4), we get:

$$P = a * L * M^b \quad (5)$$

Dividing by the total number of commits C , then:

$$P_i = a * L_i * M^b \quad (6)$$

where P_i is the number of disk accesses per transaction and L_i is the number of logical accesses per transaction.

Multiplying by S , which is the average time that one block needs to be transferred from the disk, we get the disk demand D :

$$D = S * a * L_i * M^b \quad (7)$$

S , a , and L_i are all constants, so if we let $a'' = S * a * L_i$, then this follows:

$$D = a'' * M^b \quad (8)$$

Every transaction class, and mix of transaction classes, has its own set of values for the parameters a'' and b . We ran two experiments in order to calculate a'' and b for every class. In

one experiment, we had a relatively small buffer size ($M1$); in the other experiment, we had a large buffer size ($M2$). If we let $D1$ and $D2$ be the demand values at $M1$ and $M2$, respectively, then a'' and b can be calculated as follows:

For a'' :

$$a'' = \frac{D_1}{M_1^b} \quad (9)$$

For b :

$$b = \frac{\ln\left(\frac{D1}{D2}\right)}{\ln\left(\frac{M1}{M2}\right)} \quad (10)$$

3.3.3 Disk Demand within a Mix of Classes

It is a challenging task to estimate the percentage of disk utilization associated with each class when the different types of transactions are run together. We use an approach based on the utilization law to estimate the proportion of disk utilization of each class of transactions in a mix.

Proposition: The disk demand D_i of a class i is directly proportional to the product of the total utilization of the disk U_{Disk} and the response time of this class R_i , and inversely proportional to the total population size N .

$$D_i = U_{Disk} * \left(\frac{R_i}{N}\right) \quad (11)$$

Justification: For a general type of workload consisting of n different types or classes of transactions, each class contributes a percentage of the utilization of the disk. Let U_{Disk} be the utilization of the disk when the mix of classes is run, and let U_i be the utilization contribution of a class i .

$$U_{Disk} = \sum_{i=1}^n U_i \quad (12)$$

Let N be the total population, and let N_i be the population size of class i .

$$N = \sum_{i=1}^n N_i$$

We see that:

$$U_{Disk} = U_{Disk} * \left(\frac{N}{N}\right) = U_{Disk} * \sum_{i=1}^n N_i / N \quad (13)$$

$$U_{Disk} = \sum_{i=1}^n U_{Disk} * \left(\frac{N_i}{N}\right)$$

Using equations (12) and (13), we obtain:

$$\sum_{i=1}^n U_i = \sum_{i=1}^n U_{Disk} * \left(\frac{N_i}{N}\right) \quad (14)$$

One solution for equation (14) is the case when the disk utilization of each class in the mix of classes is proportional to the number of transactions of that class in the mix². That is:

$$U_i = U_{Disk} * \left(\frac{N_i}{N}\right) \quad 1 \leq i \leq n \quad (15)$$

Let X_i and R_i be the throughput and the response time of class i . Let D_i be the demand of class i from the disk. Using the utilization law, we get

$$U_i = D_i * X_i = D_i * \left(\frac{N_i}{R_i}\right) \quad (16)$$

Equating equations (15) and (16), we get:

$$D_i * \left(\frac{N_i}{R_i}\right) = U_{Disk} * \left(\frac{N_i}{N}\right)$$

Removing N_i from both sides of the equation, we obtain the disk demand for a class i in a mix of classes:

$$D_i = U_{Disk} * \left(\frac{R_i}{N}\right) \quad (17)$$

In the case of multiple disks, and assuming that the database tables are evenly

² While this may not always be the case, we later show empirically the validity of this claim

distributed over the disks, the demand of each class from each disk will vary depending on the table distribution. In order to estimate the demand in this case, we apply the same formula above to each disk. The demand of a class i in mix of classes from disk r , $D_{i,r}$, is

$$D_{i,r} = U_r * \left(\frac{R_i}{N} \right) \quad (18)$$

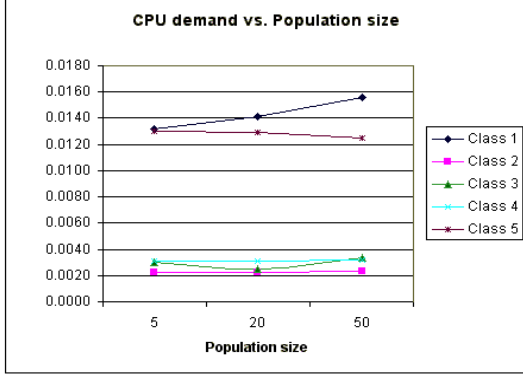


Figure 3: CPU demand vs. Population size

3.3.4 CPU Demand vs. Population Size

The CPU demand is calculated in a similar fashion to the disk demand. It does not include the queuing time for the CPU, and is independent of the number of transactions concurrently in the system. The results of experiments to verify the independence of the CPU demand from the population size are shown in Figure 3. The minor discrepancies observed in the graph are within the confidence intervals and are considered statistically insignificant.

3.3.5 CPU Demand vs. Buffer Size

As described earlier in Section 3.2, data is read from the disk into the buffer, where it is made available to the transaction running on the CPU. In general, the smaller the amount of data a transaction needs, the smaller the amount of buffer space that is needed to store this data. For each class of transaction, there exists a specific buffer threshold beyond which all the data required by the transaction fits into memory. At the threshold, or above it, the number of disk

accesses is minimized so the class is said to be CPU-intensive, and the CPU demand is independent of the buffer size. Below the threshold size, the class is disk-intensive and the CPU is underutilized. The relatively slow disk therefore binds performance, and the CPU activity is mostly supporting disk activity, which is a function of the buffer size. The CPU demand in this lower range of buffer sizes behaves in a similar fashion to the disk demand, as shown in Figure 4.

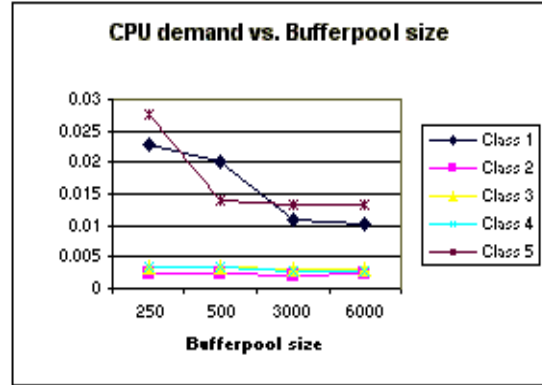


Figure 4: CPU demand vs. Buffer size

3.3.6 CPU Demand for Multiple CPUs

In the case of multiple CPUs, where the processing load is shared among the different CPUs, transaction processing is equally divided among the CPUs. We can model this fact by multiplying the service rate of the CPU resource by the number of CPUs. This has the effect of decreasing the overall CPU demand in the model by the same factor [12].

3.3.7 Memory Service Demand

The time to transfer data into and out of memory is negligible so the service demand of the memory is not significant. We include memory as a resource in our model as a way of capturing the impact of the buffer. We assume there is sufficient memory to hold the OS processes, DB2 processes, and the buffer.

3.4 Model Validation

We validate our queuing network model against the performance of DB2 on two different computer systems. System measurements were made and then used to calculate our queuing network model parameters. Using the model parameters, we estimated the performance indices. We validated the model by first comparing its performance indices with the performance indices measured directly from the system, and then statistically by applying the t-test statistical fitness test. The results of the t-test for this experiment show that the differences are insignificant with a 90% confidence level. Results of the experiment are shown in Figure 5.

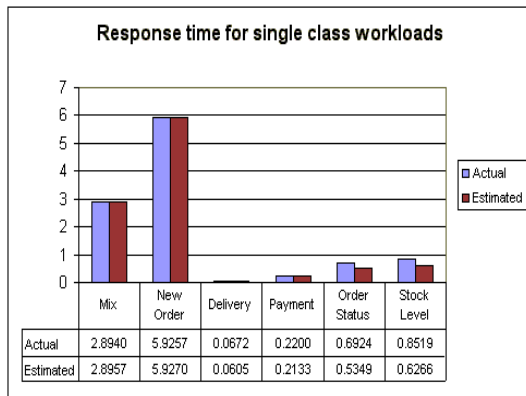


Figure 5: Real and Estimated Response Times

3.5 Summary

We can follow the procedure used in this section to produce an analytical model for any DB2 Universal Database system running an OLTP workload. The model can then be used to perform capacity planning for that system. In general, our approach works as follows:

- Using the formulas and techniques outlined above, we produce graphs that represent the behavior of the service demand of all the transaction classes on the CPU, memory, and disks involved.
- The service demands and population sizes of each class are fed into the MVA tool, which

returns estimates of the performance indices under the specified conditions.

4 Example Capacity Planning Study

As an example, we consider a typical capacity planning situation in which different system configuration alternatives must be evaluated. We assume that a database administrator is expecting an increase of 250% in the average number of users, which will likely result in a degradation in the quality of service. Without loss of generality, we assume a database system similar to that in the previous section so that we can reuse the service demands calculated above.

The current system has 20 concurrent users and the size of the buffer used in the database is 250 pages. The average response time of original system under the standard TPC-C workload is 3.3 seconds. The expected increase in the size of the population is 150%, which means a total of 50 users. Average response time under the new conditions must not exceed 3.5 seconds.

The first step is to find out if the new response time will exceed the required maximum of 3.5 seconds. Based on our results in section 3, we know that the resource demands are independent of the population size. Using the disk demand values from the previous experiment, we can estimate that the response time will jump to 8.3 seconds when the population goes up to 50 users, and the other settings remain unchanged. This is above the maximum acceptable response time (3.5 seconds). We will therefore have to take corrective action to prevent the system performance from degrading.

We consider the following three options to correct the situation:

- Increase the buffer size;
- Add two disks and redistribute the tables randomly over the three disks;
- Add an extra CPU, so the load can be divided between the two CPUs.

We can use our model to estimate the impact of each of the options on the average response time.

Option 1: Increase the buffer

Figure 6 shows the response time of the new system versus buffer size. We see that the minimum buffer size needed to achieve a response time below 3.5 seconds is 1950 pages.

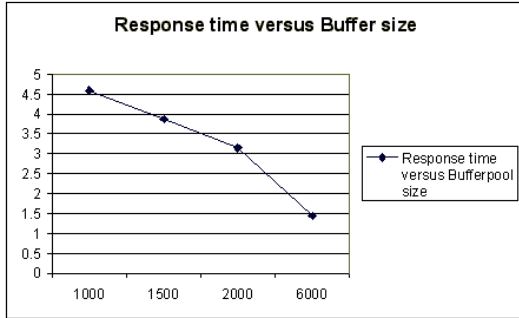


Figure 6: Response time vs. Buffer size

Option 2: Add two disks and redistribute the tables randomly over the three disks

Using CPU and disk service demand curves, we can estimate the response time of the system that contains three disks under a population size of 50 users and a buffer size of 250. By inputting the service demand values into the MVA tool, we obtain an estimate of the response time of 1.96 seconds, which is below the critical response time.

Option 3: Add an extra CPU

If we compare the service demand of the CPU and the disk, we can clearly notice that the CPU is not the bottleneck resource, so any improvement in the CPU's performance will not lead to improving the overall system's performance. As a result, we can deduce that options 1 or 2 will produce the desired results, but option 3 will not.

In general, the results from the model varied by about 10% from actual system performance. Conducting the example study emphasized the fact that the largest impact on the response time of the system comes from the bottleneck resource. The impact of the other resources on the system's response time is near to

null. This is because the transaction waits for the bottleneck resource service completion before departing from the system, even if the other resources have completed serving the transaction. Thus improving the performance of resources other than the bottleneck resource will not have a significant impact on a system's response time.

5 Conclusions

In this paper, we examined how to provide support for capacity planning studies of DBMSs. The work discussed here is the first step in the eventual development of a capacity planning tool for DB2 Universal Database. Although we specifically use DB2 and the Windows NT Version 4.0 performance monitor in our study, the techniques are not specific to these platforms and can be applied to any other database management system. In addition, the general ideas introduced in section 3, such as formulating the resource demands, and the scheme used in section 4 can be used as guidelines for capacity planning studies for other types of systems, such as distributed systems and Web-based systems, which have transaction-based workloads.

The main contribution of this paper is the design and validation of a queuing network model to capture the main features of the DB2 system's behavior. The paper also establishes relationships between the workload and the model's parameters. A relationship between disk service demand and the buffer size of a DBMS was set up. In addition, estimating the demand of an individual class from a resource, when other classes are running at the same time, has traditionally proven to be an awkward task. We managed to establish such a relationship by taking an assumption and then verifying it both mathematically and experimentally.

We are considering several extensions to the work discussed here. First, we plan to conduct similar studies for other benchmark workloads such as TPC-W [7] (Web workload) and TPC-H [7] (online analytical processing workload). We will then have a set of models that can be used to predict performance of different system configurations under different types of workloads.

Second, we plan to extend the model to account for multiple buffer pools. Third, we would like to make it possible for users of our models to specify specific workloads as input if their workloads are not sufficiently similar to any of the benchmark workloads.

References

- [1] Ed Lazowska, John Zahorjan, Scott Graham, and Ken Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*, Prentice Hall, Englewood Cliffs, N.J., 1984.
- [2] Daniel Menasce, Virgilio A.F. Almeida, Larry Dowdy, *Capacity Planning and Performance Modeling*, Prentice Hall, 1994.
- [3] Shui Lam and K. Hung Chan, *Computer Capacity Planning: Theory and Practice*, Academic Press, 1987.
- [4] Raj Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons Inc. 1991.
- [5] Martin Kienzle and Ken Sevcik, Survey of Analytic Queueing Network Models of Computer Systems, *1979 Conference on Simulation, Measurement and Modeling of Computer Systems*.
- [6] L. Belady, A Study of Replacement Algorithms for a Virtual-Storage Computer, *IBM Systems Journal* 5 (2), July 1966.
- [7] *TPC Benchmarks*, Transaction Processing Performance Council, <http://www.tpc.org/information/benchmarks.asp>, February 2002.
- [8] M.E. Senko, V.Y. Lum, and P.J. Owens, A File Organization evaluation model,

Proceedings IFIP Conference 1968, pp. 514-19.

- [9] Daniel Menasce, *Capacity Planning for Web Performance: Metrics, Methods & Benchmarks*, Prentice Hall 1998.
- [10] Ken Sevcik, Data Base System Performance Prediction Using An Analytical Model, *IEEE Computer*, February 1981.
- [11] D. Ferrari, *Computer Systems Performance Evaluation*, Prentice Hall, 1978.
- [12] H. Zawawy, *Capacity Planning for Database Management Systems using Analytical Modeling*, M.Sc. thesis, Dept of Computing and Information Science, Queen's University, 2002.

Trademarks

DB2, DB2 Universal Database, IBM, Netfinity, and POWERserver are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Microsoft and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Pentium is a trademark of Intel Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.
