Client-Server 연구논문

Modelling the Performance of a Client/Server Database System

이희석
KAIST
University of Nebraska at Omaha

MODELLING THE PERFORMANCE

OF A CLIENT/SERVER DATABASE SYSTEM

Abstract

A client/server system has become the computing architecture for the business organization which seeks competitive edges. Technically, a client/server system places application processing close to the user and thus increases performance. This paper's two primary goals are (i) to present a performance model for client/server database systems and (ii) to demonstrate analytically the effectiveness of client/server computing in comparison with other computing architectures via an illustrative example. The model is most likely to be used as a practical performance guide for client/server computing.

Keywords: Client/Server Computing, Performance Model, Response Time, Distributed Database Systems

I Introduction

A client/server system is one of the most efficient ways of computing in this time of rapidly changing technologies and new software. A client/server computing architecture maximizes the contribution of information-processing technology assets to the company's competitive advantages (Boar 1993). The client/server architecture involves multiple CPUs, typically connected in a LAN (Local Area Network). Two common LAN architectures are the Ethernet, and token ring networks. Some of the CPUs process applications and are designated as *clients*. Another CPU is designated as the *server*, and performs the task of DBMS (Database Management System) processing.

The client/server computing approach increases organizational flexibility, and allows companies to be more responsive to changes. It thus can serve as an effective basis for changes in the business process, organizational structures, management systems, or job designs. It allows the organization to separate user applications from the internal workings of the system; i.e., the client/server computing facilitates reengineering (Hammer 1990, Needelman 1993).

Furthermore, the client/server model has several performance advantages over host-centered computing. Several CPUs are processing applications in parallel. In addition, client applications communicate with a remote database server, using a query language such as SQL. Only requests for DBMS processing and the data that satisfy those requests need to be sent over the communication network. This approach is much more effective than a file server architecture, in which the complete file is sent for client applications. Database server technology brings the capacity of the mainframe DBMS down to the LAN

(Mcgoveran and White 1990).

Because of its popularity, the performance evaluation of client/server computing before its installation is of particular interest to systems designers. Performance management is one of the most important issues in database systems in the LAN. In this paper, we incorporate several key variables that determine the performance of client/server system. The emphasis in on the system response time, the average time a transaction spends in the system. On the basis of these variables, we provide its evaluation expression with the framework of queueing system. This is the first kind of study to provide an analytical performance model for client/server computing. It also highlights the performance advantages of moving form host-centered computing to the client/server model.

II Client/Server Architecture

Like any IS (information system) technology, client/server is an evolving concept. Giving its accurate definition can be difficult because client/server uses some confusing terms, such as peer-to-peer processing, database server system, cooperative processing, and distributed processing.

In client/server environments, a server is defined as a processor that provides services to requesting processors, called clients. Client/server technology expands on the file server architecture in that the server of client/server system is capable of DBMS functions; i.e., it can be called database server. A system more complicated than a client/server system is likely to be a distributed database system that operates multiple database servers. This distinction among file server system, client/server system and distributed database systems

has been discussed in a number of sources (e.g., Kroenke 1992). For the completeness of this paper, we highlight the some of the key features of three systems: the file server system, the client/server system, and the distributed database system.

The file server system distributes not only the application programs but also the DBMS to the client processors; the DBMS is on the local processors. Accordingly, the server in this setting is a file server, and not a database server; the file server is not capable of processing any query languages like SQL, and much more data needs to be sent over the LAN. Because of this limitation, the file server technology is seldom used for distributed transaction processing.

The client/server architecture was developed to ameliorate the performance degradation of the file server architecture. This entropy usually happens when many users start sharing large databases and applications; high performance is necessary, because in today's competitive business environment no one wants to wait a minute to look up product information, or wait an hour to run a report (Roti 1993). The essence of the client/server architecture is to allow client applications to access data managed by the database server (e.g., Sybase or the Microsoft SQL Server); the server is capable of DBMS functions, such as data manipulation, database integrity, concurrency control, recovery, and query optimization. Only requests for DBMS processing and results for this processing need to be transmitted across the LAN. This characteristic of the client/server system can be depicted as shown in Figure 1. It is highly probable that a well designed cooperative client/server model will reduce software maintenance cost, increase software portability, reduce the development cycle, and boost the performance of the network (Berson 1992).

If the system consists of multiple database servers, it is called a distributed database system (Ozsu and Valdureiz 1991). The database itself must be distributed. Distribution of database requires two stages: (i) fragmentation of the entire logical database and (ii) allocation of resulting physical fragments. Such database distributions are subject to different requirements and considerations of computer networks (Sheng and Lee 1992). Because of the coordination of control among several database servers, this system is more complex to develop and operate. Because of this complexity of distributed database systems, the client/server model is becoming the dominant force that will influence computing in 1990s. Many host-computing oriented companies start adopting client/server architecture as their mode of computing.

III Response Time Model

In this section, a model for the performance of client/server computing is presented. It is doubtful that one analytical model will hold in every real world setting. It is well known that a comprehensive performance analysis cannot be studied by analytical or simulation models alone. In this study, we confine ourselves to finding a reasonably satisfying measure to provide a quick reference for system designers. An analytical model will fit this purpose well, because of its capability of parametric studies and comparisons with other distributed computing alternatives. Our main goal is to present an analytical performance measure that incorporates major system parameters, and thus can serve as a stepping stone for further accurate studies, if needed.

Transaction requests in any distributed system may be categorized into DBMS requests

or Non-DBMS requests. DBMS requests must be transferred to the database server, while Non-DBMS requests are processed locally. For the sake of clear presentation, uniform load balancing is assumed. This uniformity has proved to be the optimality condition for load balancing (for a proof, see the Appendix); i.e., requests are assigned to the servers on a uniform basis.

A common performance measure for computer system is the response time, an average time a transaction request spends in the system; i.e., the elapsed time between the end of transmission of the message and the beginning of the receipt of a response message. The following notation is used to evaluate steady-state response time.

1. Transaction traffic:

n^d	number of DBMS request types (i.e., queries)
α	total DBMS request rate to a client processor
α_i	request rate of query type i to a client processor
β	total Non-DBMS request rate to a client processor
λ^{cq}	total queue access rate in client
λ_{sq}	total queue access rate in server

2. Processor measures:

n^c	number of client processors in the system
n^q	number of queues in a processor
n_i^α	number of accesses to queue by query type i
n^β	number of accesses to queue by Non-DBMS request
n^{cq}	average number of accesses to the queue of client by request

n^{sq} average number of accesses to the queue of server by request

ϕ^s service time for single I/O of queue type k of server

ϕ_k^c service time for single I/O of queue type k of client

3. Communication network parameters:

θ^c LAN communication bandwidth

δ^c total data volume transmitted per unit time

μ frame size

τ_i data file size extracted by query type i

n^p average number of packets per DBMS request

4. Performance parameters:

Φ_k^{cp} average processing time in client queue k

Φ^{cp} average processing time in client

Φ_k^{sp} average processing time in server queue k

Φ^{sp} average processing time in server

Φ^p average processing time

Φ^{pc} average communication time per packet

Φ^c average communication time

Φ^{DBMS} average response time for DBMS request

$\Phi^{Non-DBMS}$ average response time for Non-DBMS request

Φ system response time

The transaction requests are served by computing resources such as system disk, CPUs,

or communication line. Those resources generate queues for requests to be served. For the sake of simple analysis, the M/M/1 queuing model (e.g., Molloy 1989) is employed to analyze the client/server model. This kind of model has been found to be valid for the analysis of real-life computing systems with multiple servers in a LAN (Berman and Nigam 1992). Packet transmission is typically used for LAN communication. The system response time consists of two portions, local processing time and communication time. Local processing time represents the delay a request experiences within local processors such as clients and a server. Communication time represents the delay a request experiences in the LAN.

Local processing time per any queue access at any client processor can be measured as follows. Total number of accesses in client queue is represented by

$$\lambda^{cq} = \sum_{i=1}^{n^d} n_i^{\alpha} \alpha_i + n^{\beta} \beta.$$

On the basis of M/M/1 queue principle, the client processing time at any queue is represented as

$$\Phi_k^{cp} = [\phi_k^{c\,-1} - \lambda^{cq}]^{-1}.$$

In consequence, the local processing time in client processor is

$$\Phi^{cp} = \sum_{k=1}^{n^q} \Phi_k^{cp}.$$

For the application programs to be located only in client processors, where all DBMS requests in the system are processed by one database server, the total number of accesses in server queue is

$$\lambda^{sq} = n^c \sum_{i=1}^{n^d} n_i^{\alpha} \alpha_i.$$

The server processing time at any queue is then:

$$\Phi_k^{sp} = [\phi_k^{s-1} - \lambda^{sq}]^{-1}.$$

In consequence, the local processing time in server processor is

$$\Phi^{sp} = \sum_{k=1}^{n^q} \Phi_k^{sp}.$$

Because only DBMS request needs to be processed in database server, the local processing time per request is then ($\alpha = \sum_{i=1}^{n^d} \alpha_i$):

$$\Phi^p = n^{cq}\Phi^{cq} + \frac{\alpha}{\alpha+\beta} n^{sq}\Phi^{sq}.$$

Here,

$$n^{cq} = \frac{\sum_{i=1}^{n^d} n_i^{\alpha}\alpha_i + n^{\beta}\beta}{\alpha+\beta};$$

and

$$n^{sq} = \frac{\sum_{i=1}^{n^d} n_i^{\alpha}\alpha_i}{\alpha}.$$

In order to evaluate data communication time in the LAN, we first measure the data volume to be transmitted by using the equation $\delta^c = \sum_{i=1}^{n^d} \tau_i\alpha_i$. Hence, the average communication time per packet is

$$\Phi^{pc} = [\frac{\theta^c}{\mu} - \frac{\delta^c}{\mu}]^{-1} = \frac{\mu}{\theta^c - \delta^c}.$$

It is assumed that the communications for queries themselves are dominated by those for data requested by queries, and thus ignored. Average communication time per a DBMS request is then

$$\Phi^c = n^p\Phi^{pc},$$

where the average number of packets per DBMS request is calculated by

$$n^p = \sum_{i=1}^{n^d} \frac{\frac{\tau_i}{\mu}\alpha_i}{\alpha} = \frac{\delta^c}{\mu\alpha}.$$

Note that only DBMS requests require LAN communications. In sum, the system response time is then:

$$\Phi = \Phi^p + \frac{\alpha}{\alpha+\beta}\Phi^c.$$

The response times for DBMS and Non-DBMS requests, respectively, are different and computed as:

$$\Phi^{Non-DBMS} = n^{cq}\Phi^{cp};$$

and

$$\Phi^{DBMS} = n^{cq}\Phi^{cp} + n^{sq}\Phi^{sp} + \Phi^c.$$

IV Example

To demonstrate the applicability of the performance model, an example is illustrated. Dae-Won Company is a manufacturing firm in Seoul, South Korea. The president recently decided to attain a competitive edge by converting the current manual supplier management system into a client/server system. The proposed client/server system consists of 40 client processors ($n^c = 40$) that are connected to the database server in a LAN. Three tables are designed and stored in the database system. This database is an edited version of an example by Rothnie and Goodman (1977). The following are the tables.

Table Name	Fields	Table Size (tuples)
S	(S#, City)	10,000
P	(P#, Color)	1,000,000
SP	(S#, P#)	10,000,000

The table *S* consists of two fields, a supplier number and the city where the supplier is located. The table *P* consists of two fields, a part number and the color of the part. The third table *SP* has two fields, a supplier number and the part number of the part produced by the supplier.

Three DBMS requests (queries) are expressed in SQL. The first kind of DBMS request (a list of red parts) is represented as

```
SELECT    *
FROM      P
WHERE     Color    = "Red"
```

SQL command for the second DBMS request (a list of suppliers located in Seoul) is:

```
SELECT    *
FROM      S
WHERE     City     = "Seoul"
```

SQL command for the third kind of request (a list of Seoul suppliers for blue parts) is then:

```
SELECT    S.S#
FROM      S,SP,P
WHERE     S.S#    = SP.S#
AND       SP.P#   = P.P#
AND       S.City  = "Seoul"
AND       P.Color = "Blue"
```

Estimates for hourly request rates are denoted as $\alpha_1 = 10, \alpha_2 = 20, \alpha_3 = 10$, and $\beta = 20$. The processor is assumed to consist of two queues, CPU and disk. Execution of a single CPU access-disk access pair constitutes a processing time. The number of times a request needs to access this queue on the average is estimated:

$$n_1^{\alpha} = 50; n_2^{\alpha} = 80; n_3^{\alpha} = 90; n^{\beta} = 100.$$

Then total number of accesses to queues in client and server (per hour), respectively, is calculated:

$$\lambda^{cq} = 50 \times 10 + 80 \times 20 + 90 \times 10 + 20 \times 100 = 5000;$$

and

$$\lambda^{sq} = 40 \times 3000 = 120000.$$

Computer vendors supply the data on system parameters for CPU and DISK of client and server (the unit is millisecond (ms)); i.e., $\phi_{CPU}^{c} = 5, \phi_{DISK}^{c} = 10, \phi_{CPU}^{s} = 1$, and $\phi_{DISK}^{s} = 4$. Processing time components (ms) in each queue is calculated.

$$\Phi_{CPU}^{cp} = \left[200 - \frac{5000}{3600}\right]^{-1} = 5.03;$$

$$\Phi^{cp}_{DISK} = \left[100 - \frac{5000}{3600}\right]^{-1} = 10.14;$$

$$\Phi^{cp}_{CPU} = \left[1000 - \frac{120000}{3600}\right]^{-1} = 1.03;$$

$$\Phi^{cp}_{DISK} = \left[200 - \frac{120000}{3600}\right]^{-1} = 4.62.$$

The above computation results in

$$\Phi^{cp} = 5.03 + 10.14 = 15.17;$$

and

$$\Phi^{sp} = 1.03 + 4.62 = 5.65.$$

For

$$n^{cq} = \frac{5000}{60} = 83.33;$$

and

$$n^{sq} = \frac{3000}{40} = 75.00,$$

the local processing time per request is calculated:

$$\Phi^{p} = 83.33 \times 15.17 + \frac{40}{60} \times 75.00 \times 5.65(ms) = 1.55(sec).$$

Estimates for DBMS operations in server are:

Number of red parts	= 10 tuples
Number of Omaha suppliers	= 100,000 tuples
Number of Omaha suppliers for blue parts	= 100 tuples

Every tuple is 100 bits long. Hence, the total amount of data to be transmitted is calculated (bits per hour):

$$\delta^{c} = 100 \times (10 \times 10 + 10,000 \times 20 + 100 \times 10) = 2.0011 \times 10^{8}.$$

The data rate of the LAN is 10 Mbps ($\theta^c = 10^7$). The packet size is 10000 bits ($\mu = 10000$). Hence,

$$\Phi^{pc} = \frac{10^4}{10^7 - \frac{2.0011 \times 10^8}{3600}}(sec) = 1.0056(ms).$$

For

$$n^p = \frac{2.0011 \times 10^8}{10^4 \times 40} = 500.275,$$

we have

$$\Phi^c = 500.275 \times 1.0056 = 503(ms).$$

Finally, the system response time is calculated:

$$\Phi = 1.55 + \frac{40}{60} \times 0.503 = 1.89(sec).$$

The client/sever system takes, on the average, approximately 1.89 seconds to process a transaction request. Response time for different types of transactions is of particular interest for database system professionals. It is seen that

$$\Phi^{Non-DBMS} = 83.33 \times 15.17(ms) = 1.26(sec);$$

and

$$\Phi^{DBMS} = 83.33 \times 15.17 + 75 \times 5.65 + 503(ms) = 2.19(sec).$$

DBMS requests require approximately twice the time to process that Non-DBMS requests require, because of the communications with the database server.

The performance superiority of using data servers in client/server system, instead of file servers, is well known. There are fewer communication overheads [0]. To reinforce this principle using the above example, consider a file server system where DBMSs are

distributed over CPUs. After receiving DBMS requests from each CPU, the file server must transmit the entire table across the LAN, i.e., the three DBMS requests need tables (P, S, and all three tables, respectively) across the LAN. Accordingly, 10,000 tuples, 1,000,000 tuples, and 11,010,000 tuples are transmitted, i.e., the total number of bits transmitted per hour is

$$\delta^c = 100 \times (10,000 \times 10 + 1,000,000 \times 20 + 11,010,000 \times 10) = 1.302 \times 10^{10}.$$

Consequently, we have

$$\Phi^{pc} = \frac{10^4}{10^7 - \frac{1.302 \times 10^{10}}{3600}}(sec) = 1.567(ms).$$

For

$$n^p = \frac{1.302 \times 10^{10}}{10^4 \times 40} = 3.255 \times 10^4,$$

the data communication time over the LAN is

$$\Phi^c = 3.255 \times 10^4 \times 1.567 = 51.0(sec).$$

It is of particular interest to note that the average communication time required by the file server system is dramatically larger, approximately hundred times, than the time required by the client/server system (51 sec versus 503 ms). The file server system may deteriorate further because the file server may need to transmit such overhead structures as indices, views, etc.

V Concluding Remarks

This paper introduces a simple guideline for estimating the performance of client/server database systems. The model proposed is simple enough to use in practice, and complex

enough to produce a reasonable estimation of the performance. However, the applicability of the model, and perhaps other limitations, may be present because of its underlying assumptions. Although the model includes the major system parameters for performance management, further refinements can facilitate the particular situations that client/server systems face. For instance, the client servers may be heterogeneous, i.e., have different capacities. Processors may consists of several queues, such as system disk and data disk. In case of the Ethernet LAN, the collisions may degrade the communication performance.

Yet the model is of both practical and conceptual importance. From a practical perspective, the model can fit in any spreadsheet for a quick decision support. It can provide a theoretical foundation and serve as a lead-in to the study of other complex client/server configuration options. The designers may change the values of parameters so as to have different system performance; the guideline by the model can serve as a check list for the performance of client/server database systems.

Appendix

Optimality of Uniform Load Balancing:

The client/server system under consideration consists of client processors that have the same processing capacities. The requests should be served among these processors. The reasoning is based on the performance characteristics of queuing networks. The delay experienced in queues for computer systems increases exponentially, depending upon the traffic intensity of requests (Kleinrock 1976). Therefore, the performance resulting from reducing the request rate at one processor and then increasing the request rate at the

other processor, is always worse than that in the case of equal traffic intensity among two processors. Hence, the uniform traffic intensity among client processors provides optimal performance.

References

[1] L. Berman and R. Nigam, "Optimal partitioning of data bases across multiple servers in a LAN," *Interface,* vol. 22, no. 2, 1992, pp. 18-27.

[2] A. Berson, *Client/Server Architecture,* McGraw-Hill, 1992.

[3] B. H. Boar, *Implementing Client/Server Computing,* McGraw-Hill, 1993.

[4] D. M. Kroenke, *Database Processing: Fundamentals, Design, and Implementation,* Macmillan, 1992.

[5] M. Hammer, "Reengineering work: don't automate, obliterate," *Harvard Business Review,* July-August, 1990, pp. 104-112.

[6] L. Kleinrock, *Queueing Systems, Volume II: Computer Applications,* John Wiley & Sons, 1976.

[7] O. R. Liu Sheng and H. Lee, "Data allocation design in computer networks: LAN versus MAN versus WAN," *Annals of Operations Research,* vol. 36, 1992, pp. 125-150.

[8] D. Mcgoveran and C. J. White, "Clarifying client-server," *DBMS,* vol. 3, no, 12, 1990, pp. 78-90.

[9] M. K. Molloy, *Fundamentals of Performance Modelling,* Macmillan, 1989.

[10] R. Needelman, "Client-server power," *Corporate Computing,* vol. 2., no. 3, 1993, p. 11.

[11] M. T. Ozsu and P. Valdureiz, "Distributed database systems: where are we now?" *Computer,* vol. 24, no. 8, 1991, pp. 68-78.

[12] C. Rothnie and N. Goodman, "A survey of research and development in distributed database management," *Proc. 3rd International Conference on Very Large Data Bases,* October, 1977.

[13] S. Roti, "Client/Server: getting the big picture," *Computerland Magazine,* January/February, 1993, pp. 22-25.

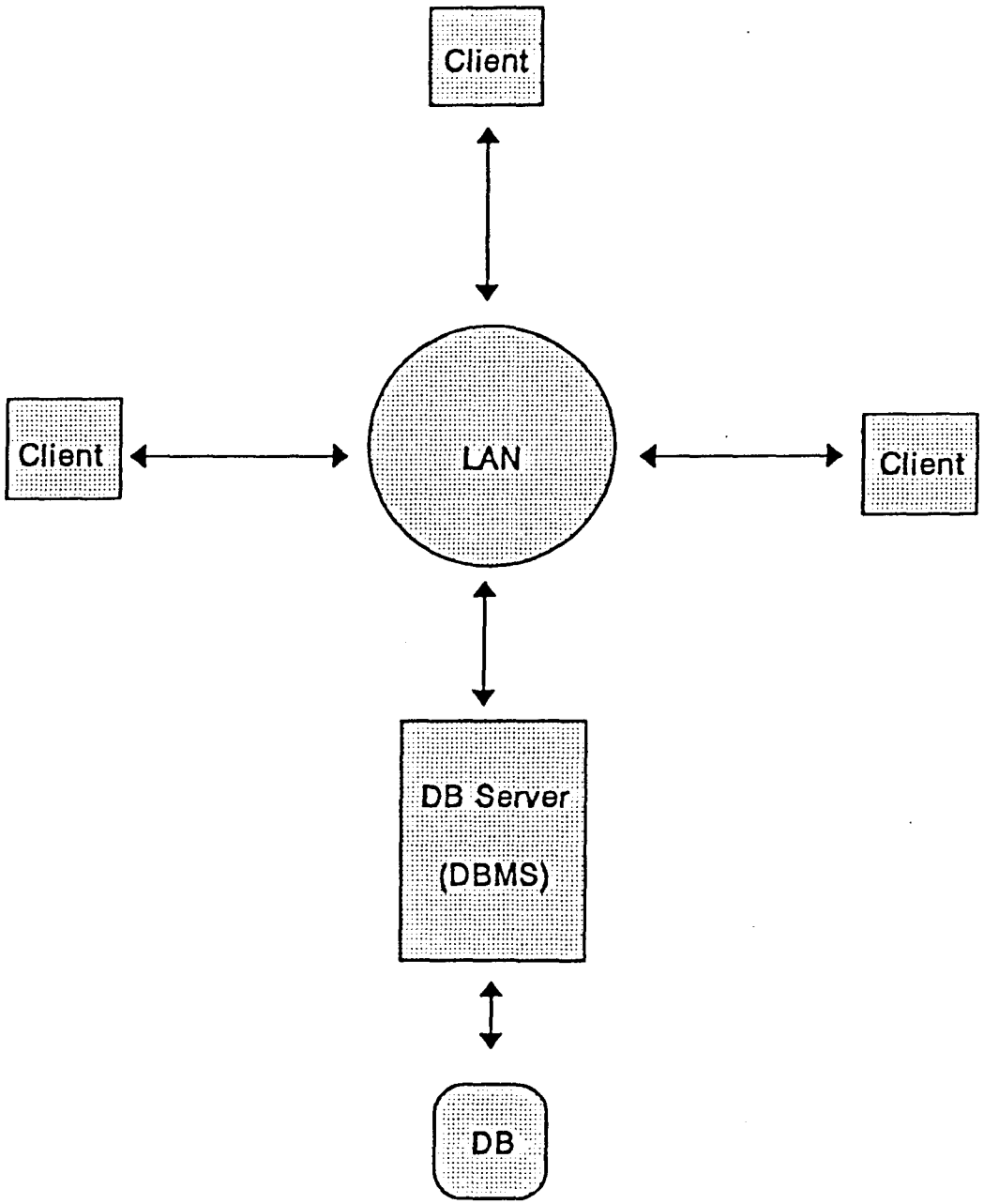

Figure 1. Client/Server Architecture