# Chapter 43

# An Analysis of the Web-Based Client-Server Computing Models

Wanlei Zhou, School of Computing/Maths, Deakin University, Geelong, VIC 3217, Australia. wanlei@deakin.edu.au
Andrzej Goscinski, School of Computing/Maths, Deakin University, Geelong, VIC 3217, Australia. ang@deakin.edu.au

## Abstract

*This paper discusses models of Web-based client-server computing systems. We classify Web-based client-server computing systems into four groups and describe them using analytic models. We carry out a performance study of each model based on the changes of various parameters related to applications and components of the Web-based computing systems in order to determine basic properties of these four systems.*

**Keywords:** Web-based client-server computing, performance evaluation, queuing theory.

## 1 Introduction

The Internet and WWW have influenced distributed computing by the global coverage of the network, Web servers distribution and availability, and architecture of executing programs. The question is what model should be used to develop application and system software of Web-based distributed computing systems. We claim here that a natural model of Web-based distributed is the client-server model [Goscinski and Zhou]. Following this, the current image of Web-based distributed computing can be called the Web-based client-server computing.

However, it is not good enough to use the simple client-server model to describe various components and their activities of a Web-based client-server computing system. The Internet, and in particular Web browsers and further developments in Java programming, have expanded the client-server computing and systems. This is manifested by different forms of co-operation between remote computers. The issue is which form is best. For this purpose, this paper tries to classify various Web-based client-server computing systems into four types, build appropriate models for each type, and carry out performance analysis of these models under various conditions related to applications and systems.

## 2 Web-based client-server computing models

We have catergorised the Web-based client-server computing systems into four types: the *proxy computing* model, the *code shipping* model, the *remote computing* model and the *agent-based computing* model.

The proxy computing (PC) model is typically used in Web-based scientific computing. According to this model the client sends data and program to the server over the Web and requests the server to perform the computing. The server receives the request, performs the computing using the program and data supplied by the client and returns the result back to the client. Typically, the server is a powerful high-performance computer or it has some special system programs (such as special mathematical and engineering libraries) that are necessary for the computing. The client is mainly used for interfacing with the user. Figure 1(a) depicts this model.

The code shipping (CS) model is a popular Web-based client-server computing model. A typical example is the downloading and then execution of Java applets on Web browsers, such as Netscape Communicator and Internet Explorer. According to this model, the client makes a request to the server, the server then ships the program (e.g., the Java applets) over the Web to the client and the client executes the program (possibly) using some local data. The server acts as the repository of programs and clients perform the computation and interface with the user. Figure 1(b) illustrates this model.

The remote computing (RC) model is typically used in Web-based scientific computing and database applications [Sandewall]. According to this model, the client sends data over the Web to the server and the server performs the computing using programs residing in the server. After the completion of the computation, the server sends the result back to the client. Typically the server is a high-performance computing server equipped with the necessary computing programs and/or databases. The client is responsible for interfacing with the user. The NetSolve system [Casanova and Dongarra] uses this model. Figure 1(c) depicts this model.

The agent-based computing (AC) model is a three-tier model. According to this model, the client sends either data or data and programs over the Web to the agent. The agent then processes the data using its own programs or using the received programs. After the completion of the processing, the agent will either send the result back to the client if the result is complete, or send the data/program/midium result to the server for further processing. In the latter case, the server will perform the job and return the result back to the client directly (or via the agent). Many commercial systems are based on this model [Chang and Scott, Ciancarini *et al*]. Figure 1(d) shows this model.

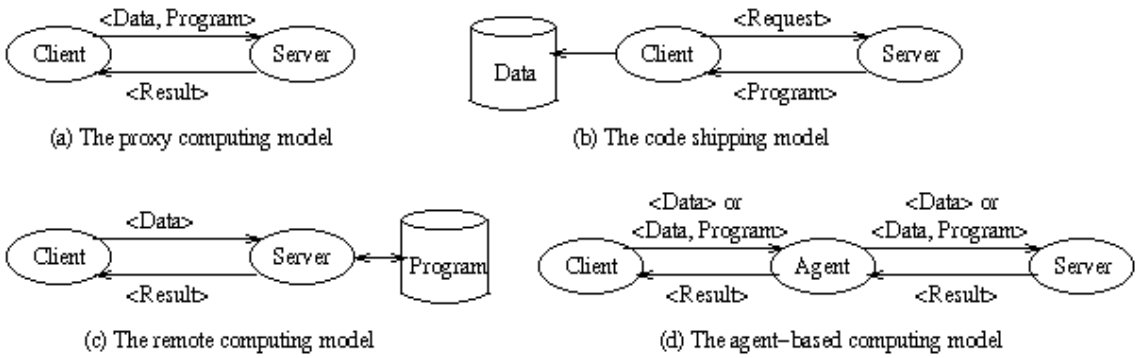



*Figure 1. Models for Web-based client-server computing*

The question arised from the above classification is which Web-based client-server computing model is best for an *application*? i.e., the application that the user is to deploy using the Web. The answer to this question actually depends on many parameters. We are interested in two types of parameters. (1) Parameters related to applications: Data size $D$; Program size $P_s$; Result size $R$; Expected execution time of program $P_t$; Request size $R_q$; Expected execution time of request $R_t$. (2) Parameters related to various components of the Web-based computing systems: Mean time service rate of the server $\mu_S$; Mean time service rate of the communication system $\mu_{CS}$; Mean time service rate of the agent $\mu_A$; Mean time service rate of the client $\mu_C$.

For the PC model, we are interested in $D$, $P_s$, $R$, $P_t$, $\mu_S$, and $\mu_{CS}$. $D$, $P_s$, and $R$ represent the load to the communication system. $P_t$ represents the load to the server, and $\mu_S$ and $\mu_{CS}$ represent the service capabilities of the server and the communication system, respectively. For the CS model, the parameters are $R_q$, $R_t$, $P_s$, $P_t$, $\mu_S$, $\mu_{CS}$, and $\mu_C$. $R_q$ and $P_s$ represent the load to the communication system. $R_t$ represents the load to the server. $P_t$ represents the load to the client, and $\mu_S$, $\mu_{CS}$ and $\mu_C$ represent the service capabilities of the server, the communication system and the client, respectively. For the RC model, the parameters are $D$, $R$, $P_t$, $\mu_S$, and $\mu_{CS}$. $D$ and $R$ represent the load to the communication system. $P_t$ represents the load to the server, and $\mu_S$ and $\mu_{CS}$ represent the service capabilities of the server and the communication system, respectively. For the AC model, the parameters are $D$, $P_s$, $R$, $P_t$, $\mu_S$, $\mu_A$, $\mu_{CS}$, and $p$. $D$, $P_s$, and $R$ represent the load to the communication system. $P_t$ represents the load to the agent and the server,. $\mu_S$, $\mu_A$ and $\mu_{CS}$ represent the service capabilities of the server, the agent and the communication system, respectively. $p$ is the probability that a <Data> or a <Data, Program> message can be serviced solely by the agent. We are also interested in the changes of the response time of each model when the parameters of applications and systems change.

## 3 The analytic models

By analysing Figure 1(a) and Figure 1(c), we can see that the PC and RC models have many common features. Firstly, both of them require message (<Data, Program> or <Data>) transmission from the client to the server and the message (<Result>) transmission from the server to the client. The two message transmissions must be handled by the communication system. Secondly, both of them also require the server to carry out computing using the supplied data. The only difference is that the PC model requires the server to use the program supplied by the client while the RC model uses the program stored in the server. Therefore, we can combine the two models into one analytic model, depicted in Figure 2(a).

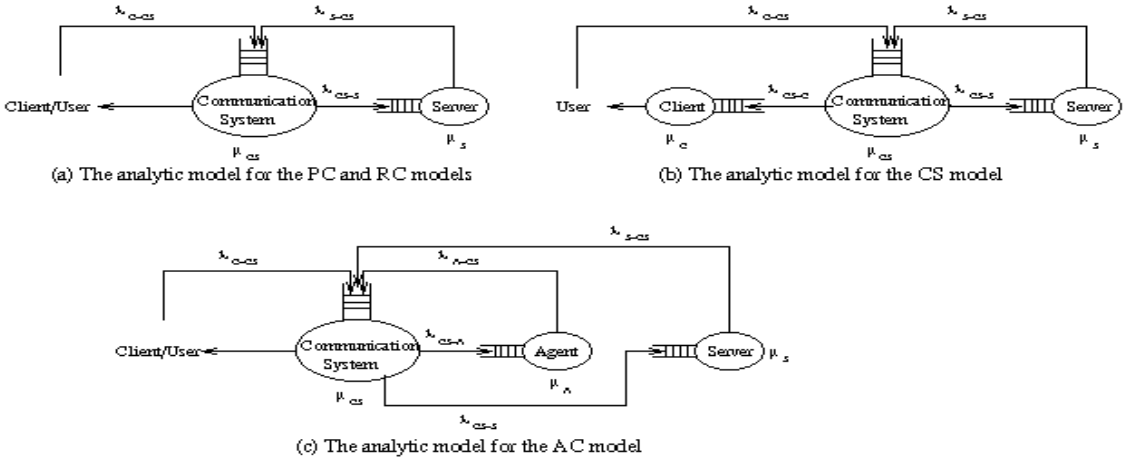



*Figure 2: The analytic model for the PC and RC models.*

In this model, the $\lambda_{C\text{-}CS}$ and $\lambda_{S\text{-}CS}$ represent the load of the communication system. In the case of the PC model, the $\lambda_{C\text{-}CS}$ represents the sizes of the data and the programs ($D$ and $P_s$). For the RC model, the $\lambda_{C\text{-}CS}$ is the size of the data ($D$). For both models, the $\lambda_{S\text{-}CS}$ represents the size of the result ($R$). The $\lambda_{CS\text{-}S}$ represents the load to the server ($P_t$).

Similarly, an analysis of Figure 1(b) can lead us to the analytic model of the CS model as shown in Figure 2(b). In this model, the $\lambda_{C\text{-}CS}$ and $\lambda_{S\text{-}CS}$ represent the load of the communication system. The $\lambda_{C\text{-}CS}$ represents the sizes of the requests ($R_q$) and the $\lambda_{S\text{-}CS}$ represents the sizes of the programs shipped to the client ($P_s$). The $\lambda_{CS\text{-}S}$ represents the the load to the server ($R_t$). The $\lambda_{CS\text{-}C}$ represents load to the client ($P_t$).

An analysis of Figure 1(d) results in the analytic model of the AC model, shown in Figure 2(c). In this model, the $\lambda_{C\text{-}CS}$, $\lambda_{A\text{-}CS}$ and $\lambda_{S\text{-}CS}$ represent the load to the communication system. The $\lambda_{C\text{-}CS}$ represents the sizes of the data ($D$) or the data and the programs ($D$ and $P_s$). The $\lambda_{A\text{-}CS}$ and $\lambda_{S\text{-}CS}$ represent the sizes of the results ($R$). The $\lambda_{CS\text{-}A}$ and $\lambda_{CS\text{-}S}$ represent the load to the agent and the server ($P_t$).

Solving the above analytic models is a very difficult task since the basic assumption, the *flow balance assumption* [MacDougall] for solving queuing networks is not true in these models. Thus, a further simplification is needed to solve these analytic models. We assume that each queue in the analytic models is a M/M/1 queue and all the queues are independent of each other (that is, the flow imbalance feature does not affect our analysis). With this assumption, the average response time of each model can be calculated using the Jackson Theorem [Kant]. We omit the detailed analysis here.

## 4 Result analysis

Figure 3(a) shows the change of the average response time for the PC and RC models when the load ($\lambda_{C\text{-}CS}$ or

$\lambda_{CS\text{-}S}$) changes, for the service rates $\mu_{CS}$=20 and $\mu_S$=25. It can be seen that when the load to the communication system increases (the curve marked by $L_{cs\text{-}s}$), the response time increases slowly until $\lambda_{C\text{-}CS}$ approaches the limit of the communication system ($\lambda_{C\text{-}CS}$=20), in that case the response time increases dramatically. We omit the case for $\lambda_{S\text{-}CS}$ since the calculation result is similar to the case of $\lambda_{C\text{-}CS}$. When the load to the server increases (the curve marked by $L_{s\text{-}cs}$), the response time increases slowly until $\lambda_{CS\text{-}S}$ approaches the limit of the server ($\lambda_{CS\text{-}S}$=25), in that case the response time increases sharply. Thus we can conclude that the PC and RC models show a reasonably good performance if we have light load to the communication system (i.e., the sizes of the <Data, Program>, <Data>, and <Result> are small) and to the server (i.e., the program's expected execution time is short).

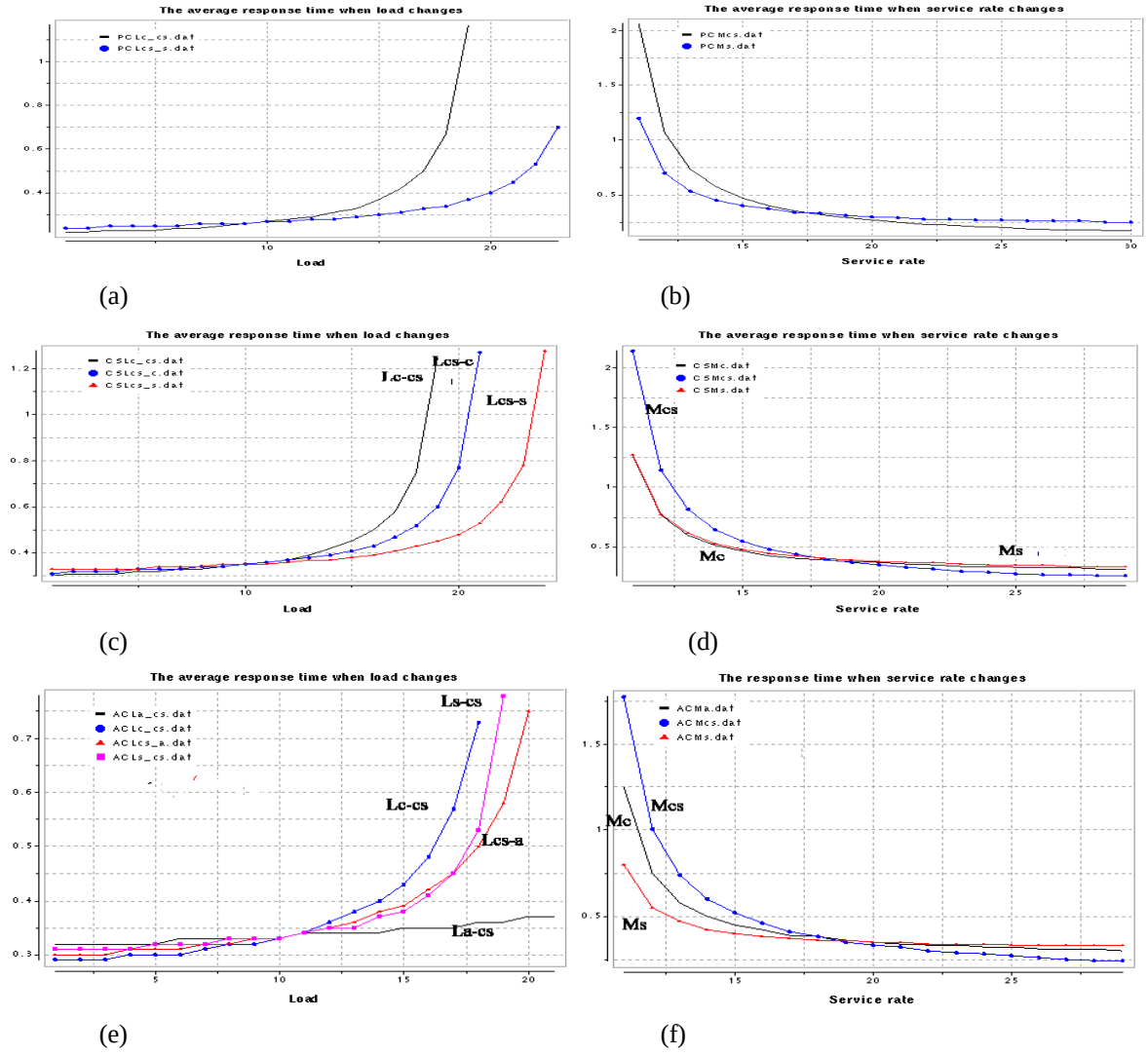



*Figure 3. The average response time when load or service rate changes.*

Figure 3(b) shows the changes of the average response time for the PC and RC models when the service rate changes, for $\lambda_{C\text{-}CS}$=10, $\lambda_{CS\text{-}S}$=10 and $\lambda_{S\text{-}CS}$=10. It can be seen that when the communication system's service rate is low (the curve marked by $M_{cs}$), the response time is very long. However, the response time decreases dramatically and then stablises when $\mu_{CS}$ increases. Similarly, when the service rate of the server is low (the curve marked by Ms), the response time is very long. As $\mu_S$ increases the response time decreases sharply

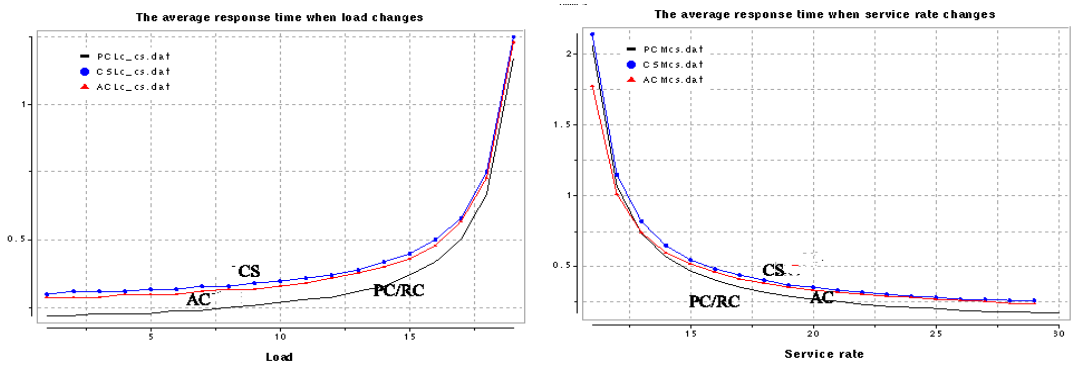
and then becomes stablised. We can then conclude that the PC and RC models have reasonably good performance if the communication system and the server are fast.

Figure 3(c) shows the calculation results for the CS model, for $\mu_{CS}$=20, $\mu_S$=25 and $\mu_C$=22 when the load ($\lambda_{C\text{-}CS}$, $\lambda_{CS\text{-}S}$, and $\lambda_{CS\text{-}C}$) changes. We omit the case for $\lambda_{S\text{-}CS}$ since it is the same as the case of $\lambda_{C\text{-}CS}$. Figure 3(d) shows the changes of the response time when the service rate ($\mu_{CS}$, $\mu_S$ or $\mu_C$) changes, for $\lambda_{C\text{-}CS}$=10, $\lambda_{CS\text{-}S}$=10, $\lambda_{S\text{-}CS}$=10 and $\lambda_{CS\text{-}C}$=10.

The calculation results for the AC model, for $\mu_{CS}$=20, $\mu_S$=25 and $\mu_A$=22 during the calculation of the average response time when the load ($\lambda_{C\text{-}CS}$, $\lambda_{CS\text{-}A}$, $\lambda_{A\text{-}CS}$, or $\lambda_{CS\text{-}S}$) changes, are shown in Figure 3(e). We also omitted the case for $\lambda_{S\text{-}CS}$ as before. Figure 3(f) shows the changes of the response time when the service rate ($\mu_{CS}$, $\mu_S$ or $\mu_A$) changes, for $\lambda_{C\text{-}CS}$=10, $\lambda_{CS\text{-}A}$=10, $\lambda_{A\text{-}CS}$=10, $\lambda_{CS\text{-}S}$=10, and $\lambda_{S\text{-}CS}$=10. In both cases, $p$=0.5.

From Figure 3(c) and Figure 3(e) we can see that for the CS and AC models the response time increases as the load increases. The increment of the response time becomes dramatic when the individual load approaches its corresponding service rate limit. From Figure 3(d) and Figure 3(f) we can see that the response time decreases dramatically and then is stablised as the service rates increase.

We can conclude from Figures 3(a)-(f) that the all the models have good performance if the loads are far lower than their corresponding service rates.




(a) (b)

*Figure 4. Comparison of models.*

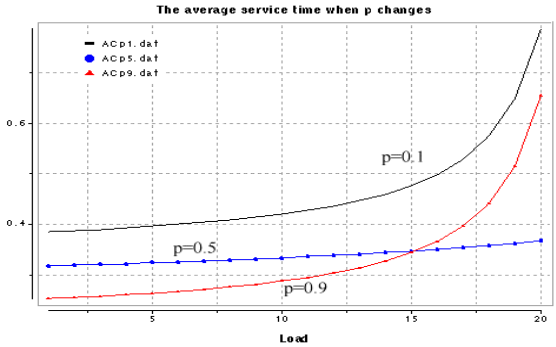



*Figure 5. The average service time when p changes.*

Figure 4 compares the performance of our models. It can be seen that the four models do not have significant difference in performance, although the PC and RC models show slightly better performance since they do not require many stages of computation and communication. It is interesting that the CS and AC models have very similar performance though the AC model is considerably more complex than the CS model. The reason

here is that we have assumed $p$=0.5, i.e., 50% of the requests from clients will be served by the agent only. This can off-set the delay incurred by the other 50% of the requests served by both the agent and the server.

It is then interesting to see the influence of $p$ on the AC model. Figure 5 shows this effect. It can be seen that when $p$ is large, the response time is short if the load is not very heavy (e.g. $p$=0.9). The response time increases as $p$ decreases. This is because that when $p$ is large, most of the request will be treated by the agent only. As $p$ increases, the number of requests processed by both agent and server increases, therefore the response time increases. It is also interesting to see that as the load ($\lambda_{A\text{-}CS}$) increases, the curve marked by $p$=0.9 overtakes the curve marked by $p$=0.5. The reason is that when too many requests are passing through the communication system (i.e., $\lambda_{A\text{-}CS}$ is large and $p$ is large) at the same time, it becomes a bottleneck. The case of $p$=0.5 evenly distributes the traffic between the communication systems and agents/servers.

## 5 Conclusions

In this paper we have classified Web-based client-server computing systems into four computing models, i.e., the proxy computing model, the code shipping model, the remote computing model, and the agent-based computing model. We also have developed analytic models for these Web-based client-server computing models and have evaluated their performance. From the analysis of the initial results, we can reach the following conclusions: (1) All the models can offer good performance when the loads to various service centres (i.e., the communication system, the server, the agent, and the client) are low and these service centres have high service rates. (2) Since information has to flow through various stages of the models, each stage has a potential to become a bottleneck of the whole system. The more stages a model has, the higher the bottleneck potential it has. Therefore, the PC and RC models have the least probability to form a bottleneck, the CS model has a moderate chance to get a bottleneck, whereas the AC model has the highest chance to form a bottleneck. (3) The AC model has a reasonably good performance compared to other simpler models. Nowadays, more and more Web-based applications have shifted to the AC model [Chang and Scott, Duan, Ciancarini *et al*]. According to our analysis, it is possible to achieve good performance using this three-tier (or multi-tier) model as long as the developer is aware of the following design issues: (a) to keep lower traffic loads, (b) to achieve higher service rates, (c) to develop powerful agents that can meet at least half of the client request, and (d) to avoid bottlenecks.

## References


[Casanova and Dongarra] H. Casanova and J. Dongarra, "Network Enabled Solvers for Scientific Computing Using the NetSolve System", *Proceedings of the 3rd IEEE International Conference on Algorithms and Architectures for Parallel Processing*, pp.17-33. Eds. A. Goscinski, M. Hobbs, and W. Zhou, World Scientific, 1997.

[Chang and Scott] J. W. Chang and C. T. Scott, "Agent-based Workflow: TRP Support Environment (TSE)", *The Fifth International World Wide Web Conference*, May 6-10, 1996, Paris, France, http://www5conf.inria.fr/fich_html/papers/P53/Overview.html.

[Ciancarini *et al*] P. Ciancarini, A. Knoche, R. Tolksdorf, and F. Vitali, "PageSpace: An Architecture to Co-ordinate Distributed Applications on the Web", *The Fifth International World Wide Web Conference*, May 6-10, 1996, Paris, France, http://www5conf.inria.fr/fich_html/papers/P5/Overview.html.

[Duan] N. N. Duan, "Distributed Database Access in a Corporate Environment Using Java", *The Fifth International World Wide Web Conference*, May 6-10, 1996, Paris, France, http://www5conf.inria.fr/fich_html/papers/P23/Overview.html.

[Goscinski and Zhou] A. Goscinski and W. Zhou, *Client-Server Model*, The Encyclopedia of Electrical and Electronics Engineering, J. Webster (Ed.), John Wiley & Sons, in print, 1998.

[Kant] K. Kant, *Introduction to Computer System Performance Evaluation*, McGraw-Hill, Inc., 1992.

[MacDougall] M. H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, The MIT Press, Cambridge, MA, USA, 1987.

[Sandewall] E. Sandewall, "Towards a World-Wide Data Base", *The Fifth International World Wide Web Conference*, May 6-10, 1996, Paris, France, http://www5conf.inria.fr/fich_html/papers/P54/Overview.html.