

A Survey of Performance Modeling of Web Servers

Sheila Srinivas

05 May 2004

1 Introduction

The Internet and the world wide web has experienced explosive growth since its inception. The web interface is becoming popular in corporate intranets as data repositories utilize web servers to distribute information. A major component of many data network traffic loads consists of web traffic. Thus analyzing the performance of web servers is a crucial component in analyzing the performance of data networks. Popular web sites receive millions of hits per day, and it is not uncommon for these sites to exhibit extremely high response times. High response times are a source of frustration for users, and with the growing use of web sites this may damage the reputation of the company offering the web site, leading to loss of business. To cope with this, overload control can be used which means that some requests are rejected in order to achieve a reasonable response time for accepted requests. When the server is operating at or near peak capacity, there is a trade-off between average latencies and the percentage of requests rejected, performance is improved by rejecting a higher percentage of requests. For real request distributions it has been found that Web servers should reject enough requests so that the average load on the system is 95% or less of the maximum capacity in order to prevent latencies from becoming too large. As a consequence, it is important to be able to identify bottlenecks, predict future capacity shortcomings, and determine the most adequate or cost effective way to reconfigure such distributed computing environments to overcome performance problems and cope with increasing workload demands.

Web server performance is a critical issue for sites which service a high volume of requests. Performance modeling is an important part of the research area of Web servers. Performance models are used to estimate the performance measures such as response time, throughput, resource utilization and resource queue length, for a given set of parameters. A performance model is said to be valid if the performance metrics calculated by the model, match the measurements of the actual system within a certain acceptable margin of error. A valid model is the basis of Web server capacity planning. This paper is a survey of the research done in the field of performance modeling of Web servers.

2 Background

2.1 Web servers

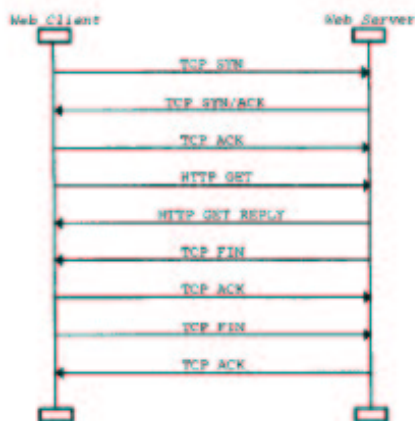


Fig. 1. Client-Server Communication

The fundamental service offered by a web server is access to static or dynamically generated HTML files. Fig. 1 depicts the typical communication between a server and a client

which successfully requests and receives a document from the server. Communication between clients and server is based on HTTP, an application level protocol built on top of TCP/IP. Communication is always in the form of a request-response pair initiated by the client. It involves the opening of a TCP connection with the 3-way handshake. The the client requests for a document by sending a HTTP GET message. Requests arrive at the listener process in the web server and are dispatched to one of a pool of server processes/threads. If the request is for HTML or image, it is retrieved from disk and sent back. If the request is for dynamic content (CGI), the server process creates a child process which runs a CGI script and returns output for the server to send back. The server sends this back to the client in one or more HTTP GET REPLY messages. The TCP connection is closed in both directions using TCP FIN and TCP ACK messages. In HTTP 1.0, if a requested HTML document contains e.g., in-line images then such images are requested separately using the same communication pattern as described above. This means that a TCP connection is opened for each subdocument. A newer version of HTTP, i.e. HTTP 1.1, includes the notion of persistent connections to avoid the overhead of opening separate TCP connections.

2.2 Capacity Planning

Capacity planning [1] is the process of predicting when future load levels will saturate the system and of determining a cost effective way of delaying or overcoming it. Capacity planning of intranets and Web servers can be used to avoid some of the obvious and most common pitfalls, such as Web site congestion and lack of network bandwidth. Before one starts rolling out an intranet or making a Web site available to customers, some key questions should be analyzed. The following are some of these typical planning questions. Is the corporate network able to sustain the intranet traffic? Will Web server performance continue to be acceptable when twice as many people visit the site? Does the company have tools to monitor and evaluate the effectiveness of the web site? Capacity planning methodologies can be used to answer these questions. The main steps of the methodology are: workload characterization, workload model validation and calibrations, performance model development and validation, cost model development and cost/performance analysis. Capacity planning uses these models to predict future shortcomings. In doing so, it is important to consider the evolution of workload due to new and existing applications and the desired service levels. The workload model captures the resource demands and workload intensity characteristics of the load brought to the system by the different types of transactions and requests. The performance model is used to predict response times, utilizations, and throughputs, as a function of the system description and workload parameters. The cost model accounts for software, hardware, telecommunications, and support expenditure.

2.3 Workload Characterization

Workload characterization is the process of precisely describing the system's global workload in terms of its main components. There has been a considerable body of work that characterizes workload on Web servers. In [8] it is shown that Web traffic exhibits a highly bursty characteristic know as self similarity. The prime cause of Web traffic self similarity is the heavy tailed distribution of Web file sizes as a result of which the transmission times are heavy tailed. The silent times due to the influence of user "think time" is also drawn from a heavy tailed distribution [9]. Hence assuming workloads with Poisson Web request arrivals with geometrically distributed file sizes would produce misleading results.

In order to develop a valid workload model we require parameters such as frequency distribution of requests, request interarrival time distribution, file referencing behavior and size of reads and writes which influence the time needed to service a request. Workload can be partitioned into classes depending on the type of request or depending on the requested document size.

3 Performance Modeling

An important aspect of capacity management involves predicting whether a system will deliver performance metrics that meet desired or acceptable service-levels. Performance prediction is the process of estimating performance measures of a computer system for a given set of parameters. Typical performance metrics include average response time, average throughput, resource utilization, average queue length and the blocking probability i.e. the probability that the queue is full and a request is rejected. Performance prediction requires the use of models. Two types of models may be used: simulation models and analytical models. A performance model is said to be valid if the performance metrics calculated by the model match the measurements of the actual system within a certain acceptable margin of error. There are two ways in which systems can be modeled: system-level models and Component-level models.

3.1 System-level models

In system-level performance models, the system being modeled is viewed as a black box or as a resource with a queue. System-level models can be represented by a state transition diagram. The simplest model is M/M/1/K*FCFS with first come first serve as the scheduling discipline. Other variants are M/D/1/K*FCFS, M/G/1/K*FCFS and M/G/1/K*PS (processor sharing). Any model that assumes Poisson arrivals does not model the web traffic accurately. A system level model that models the arrival process as a Markov Modulated Poisson process is the MMPP/G/1/K*PS, which will be investigated in this survey.

3.1.1 MMPP/G/1/K*PS model

The arrival process to the server is modeled as a two-state Markov Modulated Poisson process (MMPP). MMPP's are commonly used to represent bursty arrival traffic to communication systems, such as web servers [3]. The service time distribution is arbitrary. A web server is modeled using an MMPP/G/1/K*PS queue [2] with processor sharing as the queuing discipline as Fig. 2 shows.

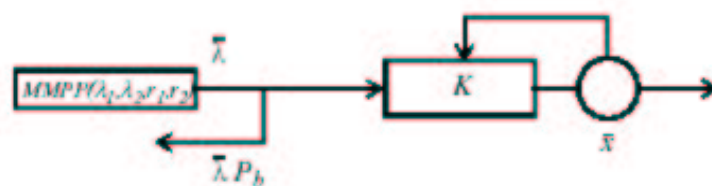


Fig. 2. MMPP/G/1/K*PS Model of a Web Server

A Markov Modulated Poisson process is a doubly stochastic Poisson process whose rate varies according to the state of a continuous time Markov chain. The Poisson process is said to

be modulated by the Markov process. Modulation introduces correlations between successive inter-arrival times. A two state Markov chain is denoted as MMPP-2 or switched Poisson process. Superposition of MMPP's result in another MMPP. Fig. 3 shows a MMPP with two states S1 and S2. The Markov chain changes state from S1 to S2 with intensity r_1 , and transmits back with intensity r_2 . When the MMPP is in state S1, the arrival Poisson process has rate λ_1 , and when the MMPP is in state S2 rate λ_2 is used. Note that if $\lambda_1 = \lambda_2$ then we have the ordinary Poisson process. If $\lambda_2 = 0$, it is called an Interrupted Poisson process. A two state MMPP can be used to model a state where the user is clicking rapidly and a state where the user is clicking moderately or is silent.

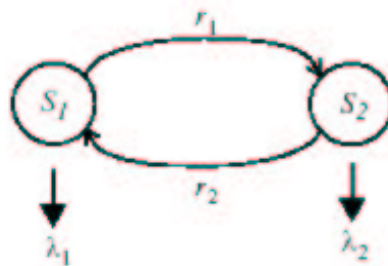


Fig. 3. The MMPP-2 State Machine

The mean λ and the variance v in a two-state MMPP are given as follows.

$$\lambda = \frac{\lambda_1 r_2 + \lambda_2 r_1}{r_2 + r_1} \quad (1)$$

$$v = \frac{r_1 r_2 (\lambda_1 - \lambda_2)^2}{(r_2 + r_1)^2} \quad (2)$$

The service time requirements for jobs in the queue have a general distribution with mean x . A job in the queue receives a small quantum of service and is then suspended until every job has received an identical quantum of service in a round-robin fashion. When a job has received the amount of service required, it leaves the queue. Requests are blocked if the queue has already K requests waiting. The probability of blocking is denoted as P_b . The rate of blocked requests is given by λP_b .

MMPP parameters: $r_1 = 0.05$, $r_2 = 0.95$. The low rate λ_1 , was set to

$$\lambda_1 = 0.75\lambda \quad (3)$$

From 1 we have,

$$\lambda_2 = \frac{(r_1 + r_2) \cdot \lambda - \lambda_1 r_2}{r_1} \quad (4)$$

This means that λ_2 is the high rate and is used 5% of the time according to the setting of r_1 and r_2 . The average response time, throughput and blocking probability are performance metrics that are obtained by simulations for various values of K and service time mean x . These simulations are validated through actual measurements.

3.1.2 Session Based MMPP/G/1 Model [4]

Evaluation of web server performance generally focuses on achievable throughput and latency for request based type of workload as a function of traffic load. A session is a sequence of related requests. Placing an order through a website involves requests relating to selecting a product, providing shipping information, arranging payment agreement and finally receiving a confirmation. So, for a customer trying to place an order, or a retailer trying to make a sale, the real measure of web server performance is its ability to process the entire sequence of requests needed to complete a transaction. When the server is overloaded, a dropped request can occur anywhere in the session. That leads to aborted, incomplete sessions. An overloaded web server can experience severe loss of throughput when measured in completed sessions while still maintaining its throughput measured in requests per second. As an extreme, a web server which seems to be busy satisfying clients requests and working at the edge of its capacity could have wasted its resources on failed sessions and, in fact, not accomplishing any useful work. Statistical analysis of completed sessions reveal that an overloaded web server discriminates against longer sessions. But sessions resulting in sales are generally 2-3 times longer than non-sale sessions. Hence discriminating against long sessions could significantly impact the profitability of commercial web sites. Session based admission control is used to provide a web quality of service guarantees for a server running a session-based workload. Its main goal is to prevent the server from overload. The admission control mechanism will accept a new session only when a server has the capacity to process all future requests related to the session, i.e. the server can guarantee the successful session completion.

Server Model

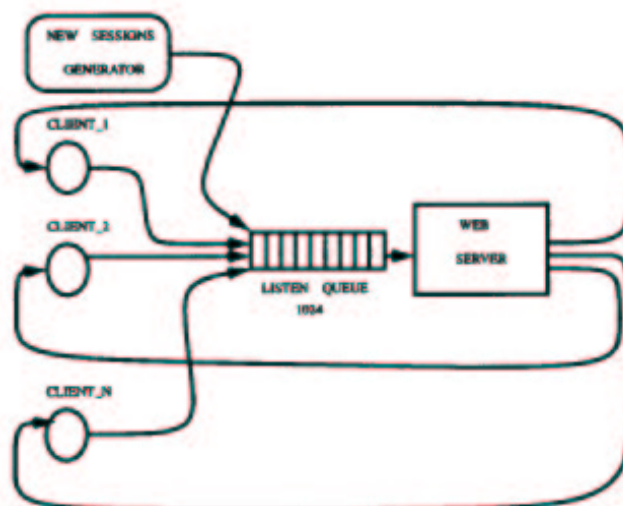


Fig. 4. Basic Simulation Model

Fig. 4 shows the basic structure of the model. It consists of a session workload generator, N clients and a web server. The session workload generator produces a new session request according to specified input model parameters: session load and session length distribution. The Session lengths are assumed to be exponentially distributed. The think time between requests of the same session is also exponentially distributed. The session length distribution and the think time distributions give rise to an arrival process that is similar to a Markov

Modulated Poisson Process. The service time is linearly proportional to the file sizes which are drawn from a general distribution.

A session request (i.e. first request of a session) is sent to a web server and is stored in the server listen queue. The size of the listen queue is limited to 1024 entries. The client issues the next session request only when it receives a reply from the previous request. If the listen queue is full, the connection to the server is refused and both the request and the entire session is aborted.

The basic idea of a session based admission controller is as follows: the server utilization is measured during predefined time intervals (say each second). Using this measured utilization (for the last interval) and some data characterizing server utilization in the recent past, an “observed” utilization is computed. If this observed utilization is above 95%, all new sessions are rejected and only requests from already admitted sessions are serviced. Once the observed utilization drops below 95%, the controller begins to admit and process new sessions. The rejection overhead is shown to be less than 5-10% of the total server work.

3.2 Component-level models

A component-level model takes into account the different resources of the system and the way in which they are used by different requests. A component-level model considers the contention for resources and the queues that arise at each system resource - CPUs, disks, routers and communication lines. Queues also arise for software resources - threads, database locks, and protocol ports. The various queues that represent a distributed system are interconnected, giving rise to a network of queues, called a queuing network. Queuing networks and Layered queuing networks are used to model component-level systems.

3.2.1 Queuing Network Models

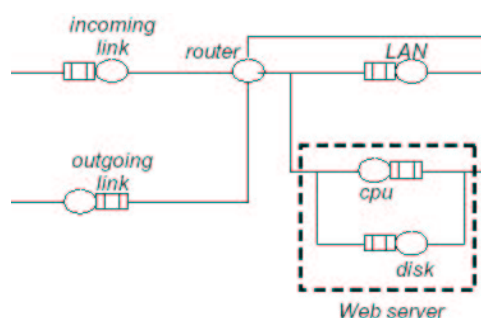


Fig. 5. QN Model of a Web Server

One can think of a computer system as a network of interconnected queues where each queue represents the physical resource (CPUs, disks, network) and the queue of requests waiting to use it. Fig. 5 shows the queuing network (QN) model of a web server connected to a LAN, which is connected to a router that connects the site to the ISP and then to the Internet. There is a large population of unknown size of clients that will access the web server. Thus the web server is modeled as an open multiclass QN model. In an open QN model requests arrive, go through the various resources and leave the system. Different classes in the workload model correspond to HTTP requests of different sizes. This is used to account for the heavy tail in document size distributions. The incoming and outgoing links and the LAN are represented by load-independent queues and the router as a delay queue. The web server is represented

by two load-independent queues: one for the CPU and another for the disk. One could have many more disks and a multiprocessor CPU as well.

Bursts of traffic cause increased congestion on system resources. To account for this increase in congestion, the service demand of the various components is inflated by using a burstiness factor. The time interval of interest τ , during which L requests arrive at the web server, is divided into n equal subintervals of duration τ/n called epochs. The burstiness factor b is defined as the fraction of time during which epoch arrival rate exceeds the average arrival rate.

$$b = \frac{(\text{Number of epochs for which } \lambda_k > \lambda)}{n} \quad (5)$$

λ_k : arrival rate of requests in epoch k

λ : average arrival rate during τ

The input parameters to the model are the service times of the components to process one request of class r and the input arrival rate. Performance metrics like response time, utilization and number of customers in the system is obtained by using an iterative algorithm like Mean Value Analysis.

3.2.2 Layered Queuing Models

Layered Queuing Models (LQM) are used to model distributed application systems where a process can suffer queuing delays both at its node's devices and at its software servers. LQM's are extended Queuing Networks that consider contention for software processes as well as physical resources. If these software delays are ignored, response time and utilization estimates for the system will be incorrect.

The processes may share devices and server processes may also request services from one another. The requests for service amongst processes can be described as a directed graph. A process that requests service from another process is an ancestor of the serving process. Statistically identical processes form a group or class. Groups at level L are permitted to request service only from groups at level $L - 1$. Hardware devices are at the lowest level. Groups that use hardware devices but call on no other software servers are at level 1. It is assumed that there are no cycles in the graph so that the model's requests for service are acyclic. Model parameters like arrival rate, average number of visits to each component and service time of components are estimated from measurements.

The Method of Layers (MOL) [7] and stochastic rendezvous network (SRVN) techniques have been proposed as performance evaluation techniques that estimate the performance behavior of LQMs. Both evaluation techniques are based on approximate MVA. The MOL is an iterative technique that decomposes an LQM into a series of QNMs. Performance estimates for each of the QNMs are found and used as input parameters of the other QNMs. The purpose of MOL is to find a fixed point where the predicted values for mean process response times and utilizations are consistent with respect to all of the submodels. At that point the results of the MVA calculations approximate the performance measures of the system under consideration. Intuitively, this is the point at which predicted process response times and utilizations are balanced so that each process in the model has the same throughput whether it is considered as a customer in a QNM or as a server.

An LQM for a Web Server

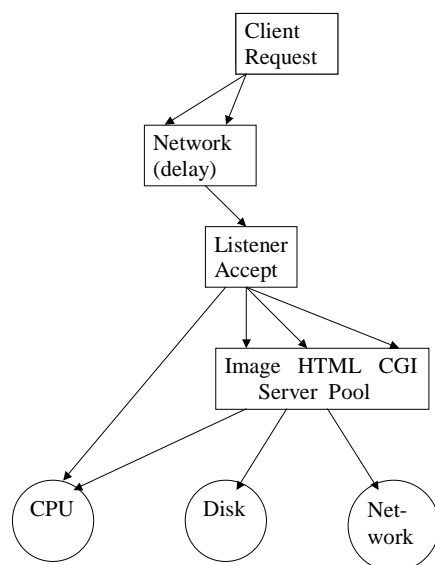


Fig. 6. LQM for Web Server

An LQM for the web server is shown in Fig. 6 [6]. The client class generates the workload for the server, it has a single service with the name *Request*. The Listener process has a single service named *Accept* that accepts client requests and forwards them to the server pool. This is reflected in the LQM as a visit by the client class to the Listener process and then a visit to the server pool. The server pool offers *Image*, *HTML*, and *CGI* services to the client. The *Image* and *HTML* requests use processor and disk resources of the server process. The *CGI* service spawns another process to execute a corresponding CGI program. For simplicity, the spawned processes CPU demand is included in that of the *CGI* service. The network is represented as a delay center. The CPU and disk have processor sharing disciplines.

4 Conclusion

This survey presents many system-level and component-level performance models. Web traffic has been shown to be inherently bursty. Therefore modeling a system assuming Poisson arrivals is clearly inadequate. Some of the system-level performance models, model the arrival process as a Markov Modulated Poisson process to account for the burstiness of arrival traffic. Other component-level performance models take the burstiness of arrival traffic into account by inflating the service demands of the components by a burstiness factor measured from HTTP logs. The Queuing Network Model is inadequate to model distributed software systems, since it considers contention only for the physical devices. Layered Queuing Models overcome the disadvantages of QNM's by considering the delays introduced by the software processes. However LQM's are based on the assumption that request graphs are acyclic and that processes do not request services from other processes at the same level or at a higher level. This clearly is a limitation of LQM's since it does not describe all possible systems.

References

- [1] D. A. Menasce and V. A. F. Almeida, Capacity Planning for Web Performance: metrics, models, and methods, Prentice Hall, 1998.

- [2] Cao, I.; Andersson, M.; Nyberg, C.; Kibl, M., "Performance Modeling of an Apache Web Server with Bursty Arrival Traffic", Telecommunications, 2003, ICT 2003, 10th International Conference on , Volume: 2 , Feb 23 - Mar 1, 2003
- [3] S.L. Scott, P. Smyth, "The Markov Modulated Poisson process and Markov Poisson Cascade with Applications to Web Traffic Modeling", Bayesian Statistics, Oxford University Press, 2003.
- [4] Cherkasova and Phaal, "Session Based Admission Control: a Mechanism for Improving Performance of Commercial Web Sites", IEEE, 1999
- [5] Cao, I.; Andersson, M.; Nyberg, C.; Kibl, M., "Web server performance modeling using an M/G/1/K*PS queue", Telecommunications, 2003, ICT 2003, 10th International Conference on , Volume: 2 , Feb 23 - Mar 1, 2003
- [6] J. Dilley, R. Friedrich, T. Jin, and J. Rolia, "Web server performance measurement and modeling techniques," Performance Evaluation, vol. 33, pp. 5–26, 1998.
- [7] Rolia J.A. and Sevcik, "The Method of Layers", IEEE Trans. on Software Engineering, Vol. 21, No. 8, August, 1995.
- [8] Mark E. Crovella and Azer Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Trans. Networking, Vol. 5, No. 6, Dec 1997
- [9] Martin F. Arlitt, Carey L. Williamson, "Internet Web servers: workload characterization and performance implications", IEEE/ACM Trans. Networking, Vol. 5, No. 5, Oct 1997