Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp200-205)

Performance Evaluations on Caching Systems using Closed Queuing Network Modeling

K. Y. Wong
Computer Studies Program
Macao Polytechnic Institute
Av. de Luis Gonzaga Gomes, Macao, China

K. H. Yeung
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Avenue, Kowloon, Hong Kong, China
eeayeung@cityu.edu.hk

Abstract: - This paper provides an analytically study on the conventional Internet caching system and the caching system using a load reduction technique. The performance metrics are request response time, disk utilization, and system throughput gain. The two systems are modeled as closed queuing networks, and the technique used to solve the networks is Mean Value Analysis (MVA).

Key-Words: Web proxy, Web caching, Closed Network, Performance Evaluations.

1 Introduction

The use of proxy servers (aka Web cache servers) provides many benefits: reducing user-perceived retrieval latencies, reducing the number of requests reaching remote servers, and reducing overall network activities. However, if a proxy is overloaded, the network performance will be degraded and the average request response time will be substantially increased. Since a proxy server is typically located at the boundary of a network and intercepts all outgoing Web requests, it is usually the performance bottleneck of entire system.

In one of our previous study [1], we proposed a solution called Site-based dispatching technique to solve the overloading condition in Web cache systems. In this paper, we analytically study the conventional Internet caching system and the caching system using SBD technique. The objectives are to study how a overloaded proxy affect the overall request response time, and how the simple SBD technique could solve the problem. The two systems are modeled as closed networks, and the technique used here to solve the network is called Mean Value Analysis (MVA). Section 1.1 reviews the SBD technique whereas section 1.2 reviews MVA.

1.1 Review of SBD

The site-based Dispatching (SBD) technique is to reduce the proxy's workload. The operations can be seen in Fig. 1. Each browser in a LAN keeps a hot-site list. The list contains the name of popularity Web sites (refer to [1] for the maintenance of the list). If a user issues a request that targets one of the sites on the list, the browser will forward the request to the proxy; if otherwise, to the remote Web site directly. As can be seen, some requests have bypassed the proxy, the proxy load can be reduced. As will be proved later, this SBD technique can shorten the response time by 43% without affecting the overall hit performance.

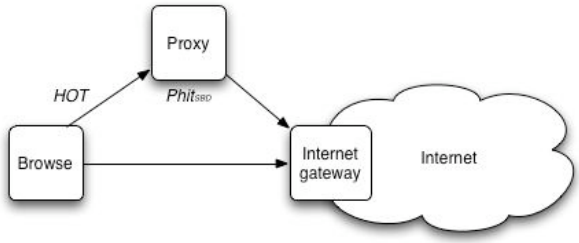


Fig. 1. The SBD system.

1.2 Review of MVA

Considering a closed queuing network with K queues, we define:

- V_i: average number of visits to queue i by a customer
- S_i: average service time of a customer at queue i per visit to the queue
- D_i: service demand at queue i, defined as $V_i \times S_i$.
- $R_i(n)$: average residence time at queue i when there are n customers in the system. This is the total waiting time plus the total service time, over all visits to queue i.
- $n_i(n)$: average number of requests at queue i when there are n customers in the system.

$X(n)$: the system throughput when there are n customers in the system.

$T(n)$: average response time when there are n customers in the system. This is equal to the sum of residence times over all queues.

The idea of MVA is to start with the known quantities $R_i(0)=n_i(0)=0$, $i=1,\ldots,K$ (corresponding to an empty system) and then calculate $R_i(1)$ and $n_i(1)$ (corresponding to one customer in the system), then calculate $R_i(2)$ and $n_i(2)$, and so on until the desired quantities $R_i(n)$ and $n_i(n)$ are obtained. This calculation is based on recursively using the following equations:

$$R_i(n) = \begin{cases} D_i & \text{delay resource} \\ D_i\left[1 + n_i\left(n_i - 1\right)\right] & \text{queuing resource} \end{cases} \quad (1)$$

$$X(n) = \frac{n}{\sum_{i=1}^{K} R_i(n)} \quad (2)$$

$$n_i(n) = X(n) \times R_i(n) \quad (3)$$

$$T(n) = \sum_{i=1}^{K} R_i(n) \quad (4)$$

Readers interested in the detailed derivations of the above equations should refer to [2].

2 Modeling of Conventional Caching Systems

The performance of conventional Web caching systems has been well studied by Menasce and Almedia [3]. In the following, we will first review the works by Menasce and Almedia, which will help our later discussions on the performance analysis for SBD system.

2.1 Queueing Network Model

As presented in [3], the queueing network model of a conventional proxy system can be modeled as a closed network with eight queues as shown in Fig. 2. The eight queues are:

clients: a delay queue that represents the set of clients. The time spent at this queue is the client think time which represents the time spent by a client (between when he starts to receive a object until he requests a new object).

LAN: a load-independent queue that represents the LAN.

router: a delay queue that represents the router with small latency. The router is named as Internet gateway in Fig. 1.

outLink: a load-independent queue that represents the transmitting link to the ISP.

Internet: a load-independent queue that represents the delays at the ISP, at the ISP's link to the Internet, at the Internet itself, and at the remote servers.

inLink: a load-independent queue that represents the receiving link to the ISP.

CPU: a load-independent queue that represents the CPU in the proxy.

disk: a load-independent queue that represents the disk in the proxy.

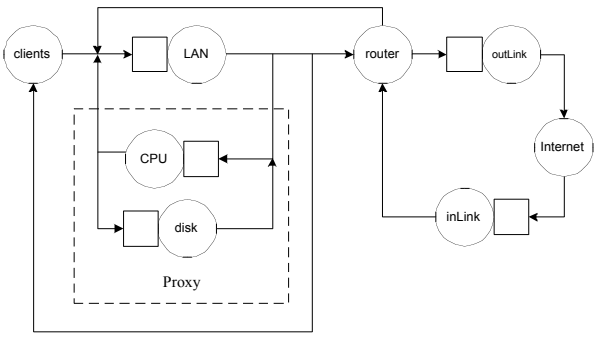


Fig. 2. The queueing network model of a conventional caching system.

2.2 Parameters

The following parameters have to be considered in this model:

EffectiveClients:	effective number of clients which is the number of client workstations actively using the Web
BrowserRate (requests/sec):	Rate in HTTP requests/sec at which a browser requests a new object
LANBandwidth (Mbps):	LAN bandwidth
HTTPRequestSize	average size of the HTTP

(Bytes):	requests generated by the browser
DocumentSize (kBytes):	average size of all requested objects
Phit	fraction of requests that can be served from the proxy server's cache
HitCPUTime (sec):	CPU time needed to process the request at the proxy and return the object to the client
MissCPUTime (sec):	CPU time needed to process the request at the cache proxy server, request the object from the originating server, atom it into the cache, perform replacement policy if needed, and send the object to the client
DiskTime (msec/kBytes):	disk time at the proxy
MaxPDU (Bytes):	maximum PDU size for the LAN's network layer protocol
RouterLatency (usec/datagram):	router latency per passing datagram
LinkBandwidth (kbps):	bandwidth of the connection to the that ISP
InternetRTDelay (msec):	Internet round trip delay between the ISP and the remote originating server
InternetDataRate (kBps):	Internet data transfer rate,

2.3 Service Demands

Finding the service demands of network queues (i.e., sum of the service times at a queue over all visits to that queue) is critical when using MVA. Having obtained them, the response time can be easily obtain using the MVA equations (i.e., eqns. (1)-(4)). In [3], Menasce and Almedia first consider the basic browser-server system in which a proxy has not been deployed. Based on the parameters given above, they derive the service demands D_i for queue i in the model of the basic browser-server system. Then, by using D_i, they obtain the expressions of the service demands for the conventional caching system model. The expressions are indicated by the superscript *proxy* and are repeated in Table 1.

Queue	Service demand
LAN	$D_{LAN}^{proxy} = Phit \times D_{LAN} + (1 - Phit) \times 2 \times D_{LAN} = (2 - Phit) \times D_{LAN}$
router	$D_{router}^{proxy} = (1 - Phit) \times D_{router}$
outLink	$D_{outLink}^{proxy} = (1 - Phit) \times D_{outLink}$
Internet	$D_{Internet}^{proxy} = (1 - Phit) \times D_{Internet}$
inLink	$D_{inLink}^{proxy} = (1 - Phit) \times D_{inLink}$
CPU	$D_{CPU}^{proxy} = Phit \times HitCPUTime + (1 - Phit) \times MissCPUTime$
Disk	$D_{disk}^{proxy} = DiskTime \times DocumentSize$

Table 1. The service demands for the queues in the model of conventional caching system. In this table, D_{LAN}, D_{router}, $D_{outLink}$, $D_{Internet}$ and D_{inLink} are the service demands for the queues in the model of a basic browser-server system with no proxy cache. The derivation of these service demands can be found in [3].

3 Modeling of SBD Caching Systems

This section is to model the system using Site-based Dispatching (SBD) technique, which is used to reduce the proxy's workload. The queueing network model, considered parameters, and expressions of service demands for a SBD caching system can be obtained by extending the works discussed above. These are detailed as follow.

3.1 Queueing Network Model

Fig. 3 shows a queueing network model for a SBD caching system. Its difference when compared with the previous model is that, a decision notation is added between the proxy and the router. It is because when using SBD, browsers will make a decision to forward a request whether to the proxy, or to the remote servers directly (via the router) bypassing the proxy.

3.2 Parameters

To calculate the service demand in this network model, we need to add the following two parameters besides the ones discussed previously:

HOT: fraction of requests whose site names can be found in the Hot-Site list and the requests are sent to the proxy

$Phit_{SBD}$: fraction of requests that can be served from the proxy's cache by using the SBD technique

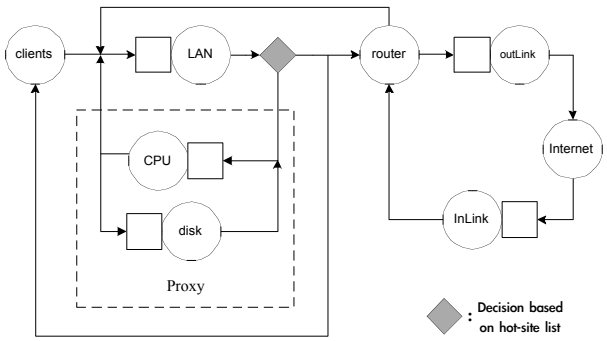


Fig. 3. The queueing network model of SBD caching systems.

3.3 Service Demands

Since the SBD technique will cause requests to traverse the network differently from that of a conventional caching system, the expressions of service demands shown in Table 1 have to be revised for the SBD system. We use the superscript *SBD* to indicate the service demands in this case. It is obvious that D_{client}^{SBD} equals D_{client}. On the other hand, since only requests with site name appearing in the hot-site list will be sent to the proxy, the service demands at the CPU and the disk of the proxy can be reduced by the addition of the fraction HOT. That is,

$$D_{CPU}^{SBD} = HOT \times [Phit_{SBD} \times HitCpuTime + (1 - Phit_{SBD}) \times MissCpuTime] \quad (5)$$

$$D_{disk}^{SBD} = HOT \times [D_{disk}] \quad (6)$$

When the requested object is hot, the service demand of LAN is similar to the one obtained in the case of conventional caching system. However, when the requested object is not hot, the service demand is equal to the value computed in the case of basic browser-server system, D_{LAN}, as the proxy is bypassed. Therefore,

$$D_{LAN}^{SBD} = HOT \times D_{LAN}^{proxy} + (1 - HOT) \times D_{LAN} \quad (7)$$

The remote Web server will be contacted either when the requested object is not hot (with probability *HOT*), or when it is hot but miss in the proxy (with probability $HOT \times (1\text{-}Phit_{SBD})$. Therefore, the service demands of the router, outgoing link, Internet, and incoming link can be obtained by,

$$D_{router}^{SBD} = [HOT \times (1 - Phit_{SBD}) + (1 - HOT)] \times D_{router} \quad (8)$$

$$D_{outLink}^{SBD} = [HOT \times (1 - Phit_{SBD}) + (1 - HOT)] \times D_{outLink} \quad (9)$$

$$D_{Internet}^{SBD} = [HOT \times (1 - Phit_{SBD}) + (1 - HOT)] \times D_{Internet} \quad (10)$$

$$D_{inLink}^{SBD} = [HOT \times (1 - Phit_{SBD}) + (1 - HOT)] \times D_{inLink} \quad (11)$$

4 Performance Evaluation using MVA

4.1 System Parameters

To compare the response time of a conventional caching system to that of a system using SBD technique, the values of the parameters defined in section IIIC have to be first set. To set the parameters, we have run computer simulations to simulate the operation of SBD systems. We used two real access traces obtained from the two proxies both at the National Lab of Applied Network Research [4]. One of the proxies is located at the NCSA at Urbana-Champaign, Illinois (UC) while another is located at the MAE West Exchange Point in San Jose, California (SJ).

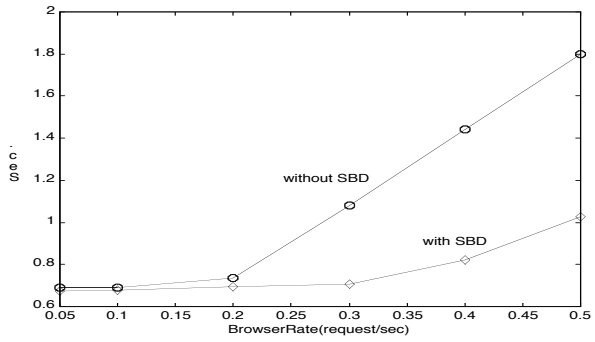
In order to examine the proxy performance improvement, we need to consider the environment that the proxy is the performance bottleneck, and therefore assume the parameter setting shown in Table 2.

4.2 Results

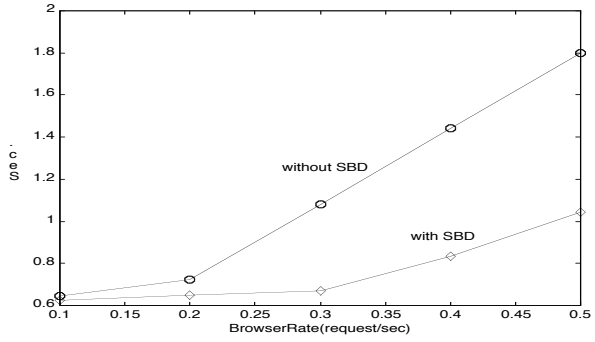
After setting the system parameters, the service demands for the two systems can be obtained by substituting the parameter values to the service demand equations (use the equations listed in Table 1 for the conventional system; use equations (5)-(11) for the SBD system). Finally, the response time for each system can be found by using MVA.

The following typical values are used:		
HTTPRequestSize	300 bytes	
HitCPUTime	0.25 msec	
MissCPUTime	0.50 msec	
DiskTime	0.3 msec per kB read	
MaxPDU	65,535 bytes	
RouterLatency	50usec / datagram	
LANBandwidth	100 Mbps	
InternetRTDelay	100 msec	
InternetDataRate	20 KB/sec	
LinkBandwidth	1544 kbps	
EffectiveClients	500	
The following values are obtained from the simulation results:		
	SJ	UC
DocumentSize	17 KB	22 KB
Phit	0.32	0.26
HOT	0.58	0.57
$Phit_{SBD}$	0.5	0.42
The following value is varied:		
BrowserRate	0.1-0.5 requests/sec	

Table 2. The used parameter setting.

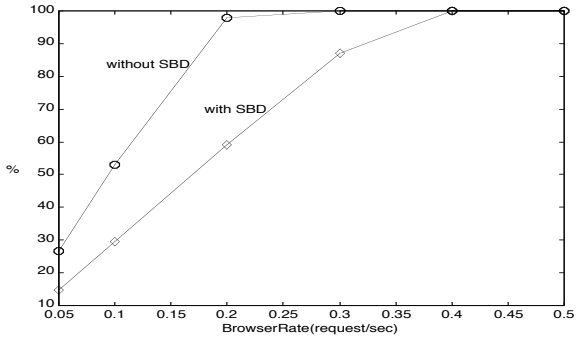


(a) UC

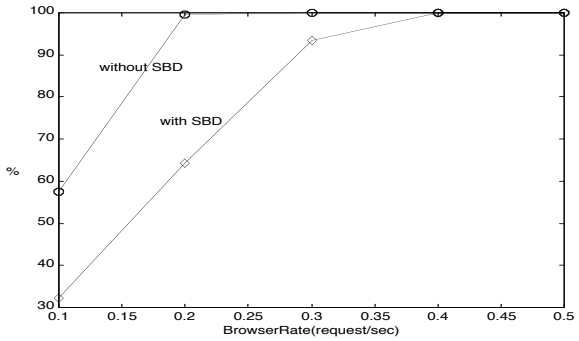


(b) SJ

Fig. 4 Response time.



(a) UC



(b) SJ

Fig. 5 Disk utilization.

As shown in Fig. 4, the SBD technique effectively reduces the response time for all traces and all BrowserRate. When BrowserRate reaches 0.5 request/sec, the reductions in response time are about 43% (by (1.8-1.03)/1.8) and 42% for UC and SJ traces respectively. It is expected as the proxy load has been substantially reduced by SBD.

MVA also gives a network queue's utilization and the system throughput. Fig. 5 shows the disk utilizations difference between the two systems. Considering the conventional system, it can be seen that as BrowserRate increases, the disk utilization increases as well. And when BrowserRate reaches 0.2 request/sec, the disk tends to be 100% utilized for both traces. As the BrowserRate continues to increase, the proxy (or the disk) becomes the performance bottleneck in the system, resulting in high response time and limiting the system throughput. To tackle this problem, the SBD technique can be used. As shown in Fig. 5, the use of the technique can effectively reduce the proxy load. It is because some requests are forwarded to remote servers directly without passing through the proxy.

On the other hand, by using the MVA equations, the system throughputs of the conventional and SBD system for UC trace are 487 requests/sec and 278 requests/sec, respectively. And the that for SJ trace are 479 requests/sec and 277 requests/sec, respectively. Therefore, the system throughput gain is 487/278 = 1.75 and 479/277 = 1.73 for UC and SJ traces respectively.

5. Conclusion

We have analytically studied the conventional Web caching system and the caching system using SBD technique. Based on the performance metrics of request response time, disk utilization, and system throughput gain, it is shown that the simple load reduction technique can improve the overall performance substantially.

Unlike the solution of adding more proxies which aims at increasing the total capacity of the caching system, the SBD technique aims at solving the fundamental problem by bypassing the unnecessary requests reaching the caching system, resulting in lesser load to the proxy, higher throughput gain and faster response time. As shown in section 4, the SBD technique provides 48% response time reduction but also 1.75 system throughput gain.

Another advantage of the technique is that it can be easily implemented with the use of a proxy auto-configuration (PAC) script [5]. PAC script was first introduced by Netscape Navigator to enable dynamic proxy selection, and has since been supported by almost all today's browsers and adopted by many Internet systems due to its simplicity and effectiveness. The script can be stored in any host in the network (the host storing the script is actually the hot-site manager mentioned in this study). Every time the browsers start, they fetch the PAC script from the host and store it locally. Before each URL retrieval, they first execute the script to determine which proxy to be used. As can be seen, the use of PAC script matches very well to the idea of hot-site list. In our implementation, we use the PAC script to determine whether the client forwards a request to the proxy or the remote server. In other words, the SBD technique can be achieved by expressing the hot-site list in the form of a PAC script. Fig. 6 shows the sample PAC script used in a SBD system, which forwards requests to the internal proxy only if they belong to one of the six hot sites.

References

[1] K. Y. Wong and K. H. Yeung, "Site Based Dispatching (SBD) Technique for Proxy Load Reduction", Proceedings for the International Symposium in Consumer Electronics, pp.54-61, Hong Kong, Dec. 2000.

[2] M.Reiser and S. Lavenberg, "Mean-value analysis of closed multi-chain queuing networks," J.ACM, vol. 27, no. 2, 1980.

[3] D. A. Menasce, L. W. Dowdy, V. Almeida, "Performance by Design : Computer Capacity Planning By Example," Prentice Hall, 2005.

[4] National Lab of Applied Network Research. Sanitized access log. Available at ftp://ircache.nlanr.net/Traces/.

[5] Navigator Proxy Auto-Config File Format. Available at http://home.netscape.com/eng/mozilla/2.0/relnotes/demo/proxy-live.html.

```
FindProxyForURL (url, host, method)
{
    if  ( dnsDomainIs(host, "hotsite1.com")||
          dnsDomainIs(host, "hotsite 2.com")||
          dnsDomainIs(host, "hotsite 3.com")||
          dnsDomainIs(host, "hotsite 4.com")||
          dnsDomainIs(host, "hotsite 5.com")||
          dnsDomainIs(host, "hotsite 6.com"))

        return "PROXY proxy1:8080";

  else
       return "DIRECT";
}
```

Fig. 6. A sample PAC for SBD systems.