



Trade-offs in Designing Web Clusters

Daniel A. Menascé • George Mason University • menasce@cs.gmu.edu

High-volume Web sites often use clusters of servers to support their architectures. A load balancer in front of such clusters directs requests to the various servers in a way that equalizes, as much as possible, the load placed on each.

There are two basic approaches to scaling Web clusters: adding more servers of the same type (scaling out, or horizontally) or upgrading the capacity of the servers in the cluster (scaling up, or vertically).¹ Although more detailed and complex models would be required to obtain more accurate results about such systems' behavior, simple queuing theory provides a reasonable abstraction level to shed some insight on which scaling approach to employ in various scenarios.

Typical questions about Web cluster design include

- Whether to use a large number of low-capacity inexpensive servers or a small number of high-capacity costly ones to provide a given performance level;
- How many servers of a given type are required to provide a certain performance level at a given cost; and
- How many servers are needed to build a Web site with a given reliability.

Using queuing theory, I will examine the average response time, capacity, cost, and reliability trade-offs involved in designing Web server clusters.

Basic Web Cluster Architecture

Figure 1a shows a cluster A with n identical Web servers and a load-balancer that equally distributes the total traffic of λ requests per second among all servers. The n servers in the cluster each

have X requests per second of processing capacity. The cluster's total capacity is therefore nX requests/sec. Figure 1b shows a cluster B with m ($m < n$) identical Web servers, each with a capacity of kX requests/sec ($k > 1$). The cluster's total capacity is mkX requests/sec. Thus, cluster B has fewer servers, but each of them has greater processing capacity than the servers in cluster A.

We can compare design considerations related to clusters A and B under four circumstances:

- *Equal average response time.* The number of servers required in cluster B to obtain the same average response time as in cluster A is a function of several factors including the cluster's request-arrival rate.
- *Equal cluster capacity.* The capacity nX of cluster A is equal to the capacity mkX of cluster B. Thus, $m = n/k$.
- *Equal cluster cost.* The total cost of the n servers of cluster A is the same as the total cost of the m servers of cluster B. Thus, $n C(X) = m C(kX)$, where $C(x)$ is the cost of a server with capacity x , which implies that $m = nC(X) / C(kX)$.
- *Equal cluster reliability.* Reliability is the probability that a system or component operates properly and continuously during a given time period. Let r_A be the reliability of each server in cluster A and r_B be the reliability of each server in cluster B.

In other work,² my colleague and I found that the reliability of clusters A and B, R_A and R_B , can be computed as

$$R_A = 1 - (1 - r_A)^n \quad (1)$$

and

$$R_B = 1 - (1 - r_B)^m. \quad (2)$$

Equating Equations 1 and 2 and applying logarithms yields $m = \lceil n \log(1 - r_A) / \log(1 - r_B) \rceil$.

Next I describe a performance model, which I use to analyze cluster performance in each of these four cases.

A Simple Performance Model

I use a well-known queuing theory result to compute the average response time of a request at a Web cluster. Assuming that requests arrive at the cluster from a Poisson process, that a request's processing time at a server has a general distribution, and that a perfect load-balancer equally distributes the load among all servers in the cluster, we can use the M/G/1 queue result³ (that is, a queue with Poisson arrivals, arbitrarily distributed service times, and a single server) to compute the average response time T for a Web request as

$$T = S + \frac{U \times S(1 + C^2)}{2(1 - U)}. \quad (3)$$

In this equation, S is the request's average service time, C is the coefficient of variation of the service time (the ratio between the service time's standard deviation and the average service time), and U is the server utilization computed as $\lambda_w S$, where λ_w is the average arrival rate of requests to a Web server.

For cluster A, $S = 1/X$ and $\lambda_w = \lambda/n$; for cluster B, $S = 1/(kX)$ and $\lambda_w = \lambda/m$. Note that because the utilization U must be less than one to make the result hold in Equation 3, I can write

$$U_A = \frac{\lambda}{n} \times \frac{1}{X} < 1 \Rightarrow \lambda < nX \quad (4)$$

and

$$U_B = \frac{\lambda}{m} \times \frac{1}{kX} < 1 \Rightarrow \lambda < mkX. \quad (5)$$

Equation 4 says that cluster A supports a maximum theoretical arrival rate of nX requests/sec, and, according to Equation 5, the maximum theoretical arrival rate cluster B supports is mkX requests/sec.

Assuming the random variable representing service time in each server of cluster B is equal to the random variable for service time in cluster

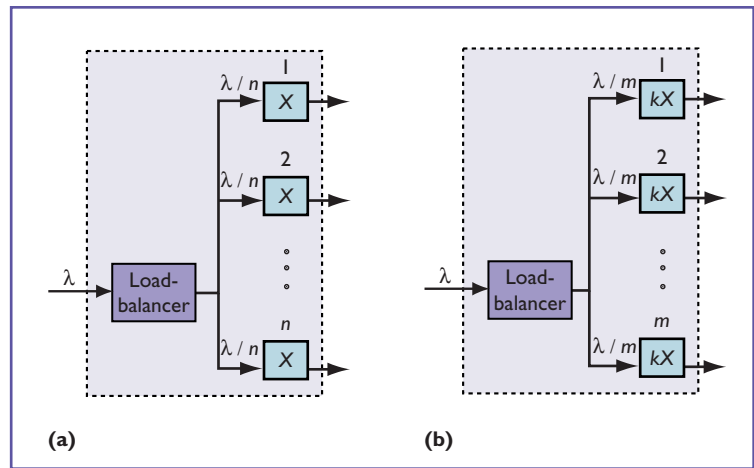


Figure 1. Web cluster architecture designs. (a) Cluster A includes n servers, each with X requests/sec of capacity. (b) Cluster B has m servers, each with kX requests/sec of capacity. Cluster B has fewer servers, but each has greater processing capacity than the servers in cluster A.

A divided by k , the coefficient of variation in the service time is the same for both clusters. I denote this coefficient of variation as C in the following equations. In all numerical examples, I use a high coefficient of variation ($C = 5$) to indicate the high variability in the sizes of files retrieved from Web sites.

Plugging the proper values of S and U into Equation 3 yields the average response time at clusters A and B:

$$T_A = \frac{1}{X} + \frac{\frac{\lambda}{n} \times \left(\frac{1}{X}\right)^2 (1 + C^2)}{2 \left(1 - \frac{\lambda}{n} \times \frac{1}{X}\right)} \quad (6)$$

$$T_B = \frac{1}{kX} + \frac{\frac{\lambda}{m} \times \left(\frac{1}{kX}\right)^2 (1 + C^2)}{2 \left(1 - \frac{\lambda}{m} \times \frac{1}{kX}\right)}. \quad (7)$$

Little's law⁴ lets me compute the average number of requests processed as $\bar{N}_A = \lambda \times T_A$ for cluster A and $\bar{N}_B = \lambda \times T_B$ for cluster B.

The results of these equations provide a good basis for exploring some design trade-offs for Web clusters.

Equal Average Response Time Case

To determine the value of m that makes $T_A = T_B$ for the same value of λ , I equate Equations 6 and 7 and, after some algebraic manipulation obtain:

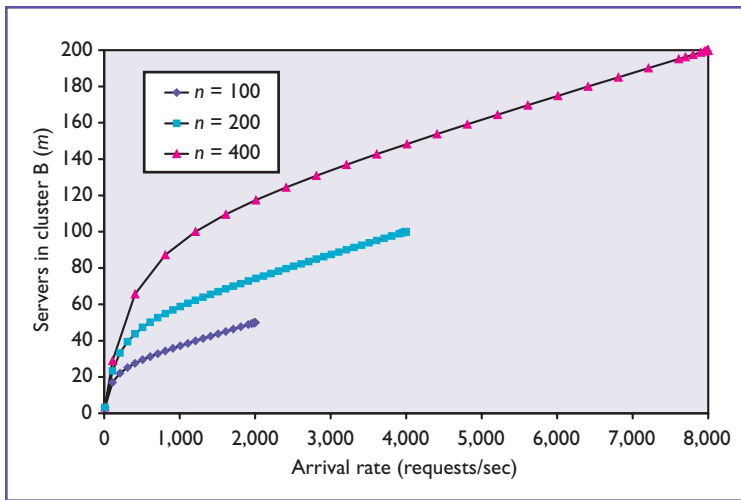


Figure 2. Servers in cluster B for the equal response time case. The number of servers in cluster B varies as a function of the total arrival rate to the cluster. The graph shows the cluster size for three values of the number n of servers in cluster A and for $k = 2$.

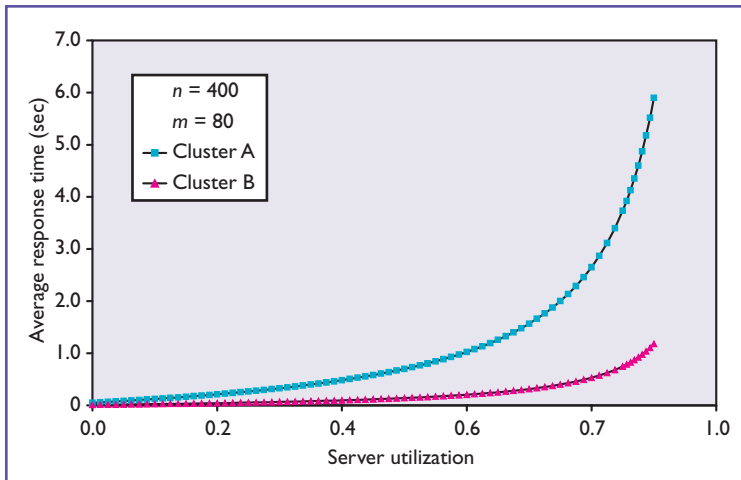


Figure 3. Average response time as a function of server utilization in each cluster for the equal capacity case. Cluster B has 80 servers. Because $k = 5$, cluster B's average response time is always five times smaller than cluster A's. Server utilization is the same for both clusters.

$$m = \frac{1}{kX} \left[\lambda + \frac{1}{\frac{2(k-1)}{\lambda \times (1+C^2)} + \frac{k}{nX - \lambda}} \right]. \quad (8)$$

Note that due to the utilization constraints of equations 4 and 5, $nX > \lambda$ and $m > \lambda / kX$. Thus, equation 8 indicates that the number of servers in cluster B required to obtain the same average response time as cluster A depends on the

- total traffic λ

- average server capacity X of servers in cluster A
- number n of servers in cluster A
- multiplication factor k of the processing capacity of servers in cluster B, and
- coefficient of variation C of the service time.

Equation 8 also indicates that as λ increases and approaches nX requests/sec (its maximum possible value), m tends to n/k . At this point, both clusters have the same total processing capacity.

For all numerical results discussed hereafter, I use the following parameters unless noted otherwise: $n = 400$, $k = 5$, $C = 5$, and $X = 20$ requests/sec. Figure 2 depicts the value of m as a function of λ for $k = 2$ and for three values of n (100, 200, and 400). The three curves in the figure each end when the arrival rate achieves its maximum possible value according to Equations 4 and 5. When cluster A has 400 servers and the cluster receives 4,810 requests/sec, cluster B needs 160 servers to obtain the same average response time of 1.03 seconds. At this point, each server in cluster A is at 60 percent utilization, and the cluster B servers are at 76 percent.

Equation 8 also tells us that for a sufficiently large arrival rate value, m increases linearly with λ at the rate of $(k - 1)/(k^2X)$. As Figure 2 shows, this rate of increase does not depend on n .

Equal Capacity Case

When the two clusters have the same total processing capacity, we can use $m = n/k$ as the value of m in Equation 7. After some algebraic manipulation, we get $T_B = T_A / k$ and $\bar{N}_B = \bar{N}_A / k$, which means that for any value of the arrival rate λ , cluster B's average response time is k times less than cluster A's. On average, however, B can handle k times fewer transactions than cluster A. Equations 4 and 5 show that server utilization is equal for the two clusters in this case.

Figure 3 shows the variation in average response times as a function of the utilization of each server in either cluster, given equal cluster capacity. The figure shows that the average response time of cluster B is always five times ($k = 5$) smaller than that of cluster A.

Equal Cost Case

Because cost is always a key consideration, it is important to compare the clusters' performance when they have the same cost – when $m = n C(X) / C(kX)$. If the server's cost is proportional to its capacity, it is easy to see that m has the same value as in the equal capacity case. By exploring two

types of cost functions – square root and quadratic – we can discuss *sublinear* and *superlinear* cost functions. A sublinear cost function indicates that there is some economy of scale as the server capacity increases (that is, the cost per unit of capacity decreases as the capacity increases). In the super-linear case, the server cost per unit of capacity increases with the capacity.

If $C(x) = \alpha\sqrt{x}$ ($\alpha > 0$), then $m = n/\sqrt{k}$. Using this value of m in Equation 7 yields cluster B's average response time. It is easy to see from Equations 4 and 5 that a server's utilization in cluster A is \sqrt{k} times the utilization of a server in cluster B.

Figure 4 illustrates that the average response time at cluster A grows much faster than at cluster B. When the cluster receives 6,800 requests/sec, for example, a server in cluster A has a utilization of 85 percent versus 38 percent for a server in cluster B. The 3.73-second average response time at cluster A is almost 42 times larger than at cluster B.

If we consider a quadratic cost function of the form $C(x) = \alpha x^2$ ($\alpha > 0$), then $m = n/k^2$. Equations 4 and 5 tell us that the utilization of servers in cluster B is k times the utilization of servers in cluster A.

Figure 5 depicts the average response time for clusters A and B for the quadratic cost function case versus the server utilization in cluster B. For low traffic-intensity values, cluster B has a slightly better performance than cluster A because there is very little queuing at cluster B, and each of its servers is five times faster than cluster A's. As traffic intensity increases, however, queuing at cluster B causes the average response time to increase very quickly. Thus, at high traffic, cluster B is better than A for a square-root cost function, but the reverse is true for a quadratic cost function.

Equal Reliability Case

When reliability is equal at both clusters, $m = \lceil n \log(1-r_A) / \log(1-r_B) \rceil$. Comparing the average response time when each server in cluster A has a reliability of 0.9 and each in cluster B has a reliability of 0.999 suggests that cluster B would need 133 servers to achieve the same reliability as cluster A.

Figure 6 (next page) shows the average response time versus the utilization of servers in cluster A. Cluster B has a significantly lower response time than cluster A. When the server utilization in cluster A is 80 percent, for example, the server utilization in cluster B is 48.1 percent; at 2.65 seconds, the average response time at cluster A is 20 times higher.

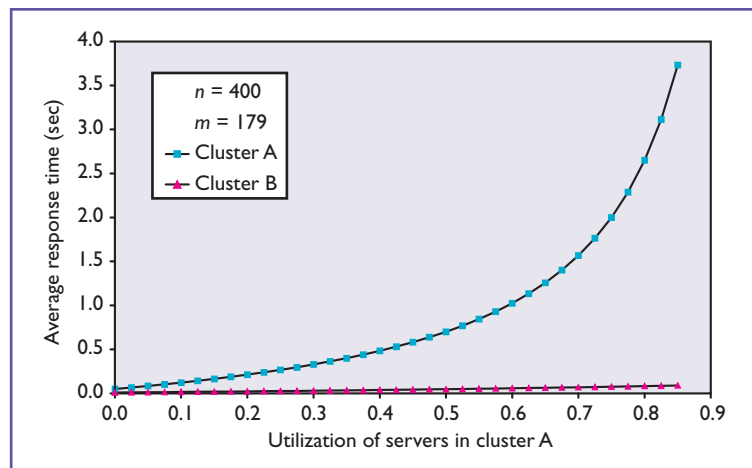


Figure 4. Average response time for clusters as a function of the utilization of servers in cluster A. Both clusters have the same total cost, and the cost of each server grows with the square root of its capacity. Cluster B has 179 servers.

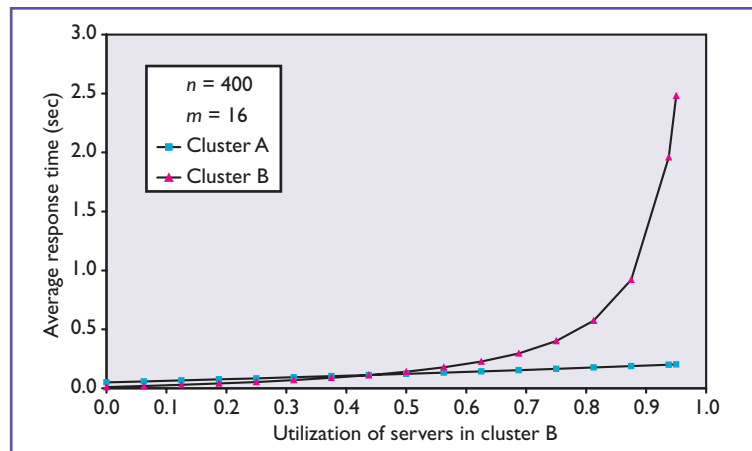


Figure 5. Average response for clusters as a function of server utilization in cluster B. Both clusters have the same total cost, and the cost of each server grows with the square of its capacity. Cluster B performs slightly better with low traffic, but cluster A is considerably faster as traffic intensity increases.

Comparing Design Criteria

Table 1 shows the average response time for cluster B, the number of servers in the cluster, and the utilization of a server in that cluster for the cases I've discussed ($\lambda = 4,800$ requests/sec). The comparison assumes 60 percent utilization for servers in cluster A and an average response time of 1.025 sec at that cluster.

The table indicates that m varies significantly depending on the design criteria for cluster B, as do the response time and utilization at the cluster. For the equal cost case for a quadratic cost function, the utilization at cluster B servers exceeds 100 percent, and the average response time goes to infinity.

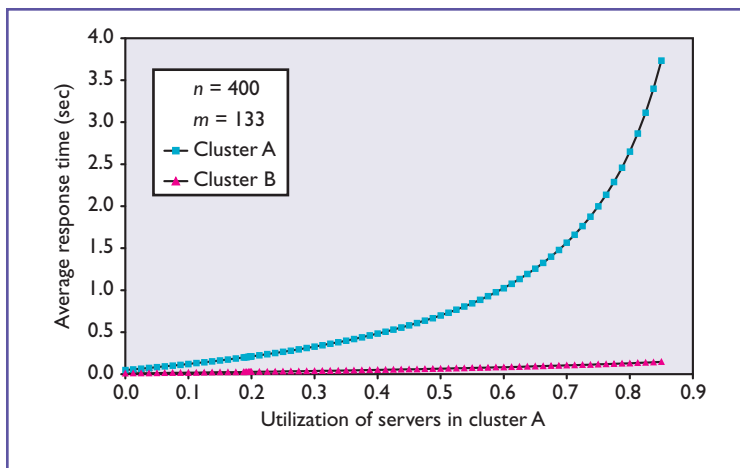


Figure 6. Average response for clusters as a function of server utilization in cluster A. Both clusters have the same reliability. Cluster A has 400 servers and cluster B has 133.

Table 1. Required cluster B size (m) for $l = 4,800$ requests/sec for various cases.

T_B (sec)	m	U_B (%)	Case
1.025	54	89	Equal response time
0.205	80	60	Equal total capacity
0.058	179	27	Equal cost: $C(x) = \sqrt{x}$
∞	16	> 100	Equal cost: $C(x) = x^2$
0.083	133	36	Equal cluster reliability

Conclusion

High-volume Web sites use clusters of Web servers as a way to spread the processing load among various machines, provide increased reliability, reduce response time, and allow for incremental growth in capacity. Some sites have thousands of very inexpensive servers while others use a much smaller number of expensive, high-capacity machines. What approach is the best? The answer depends on the application's response-time and reliability requirements and on the cost constraints. The approach I described in this article provides a simple, but sound, formulation for examining the tradeoffs in this large design space.

The actual performance of Web clusters also depends on issues not addressed here, such as the details of the load-balancing policy^{5,6} used by the load balancer; for simplicity, I have assumed perfect load balancing.

Many complex Web sites, especially e-commerce sites, have a multitiered architecture composed of a load balancer (sometimes more than one), Web

servers, applications servers, and database servers.¹ The servers in each tier are usually arranged in clusters, and caching at all levels affects performance. Here, I have considered a single tier of servers, but the observations can be adapted, under certain conditions, to clusters at any tier.

Finally, many workload characterization studies have shown that the service time distributions at Web servers are heavy-tailed (a Pareto distribution, for example).^{7,8} In simple terms, this means a non-negligible probability that large service time values will occur. Dealing with heavy-tailed distributions is very difficult in analytic models and simulation studies, especially when their variance and mean are infinite. The simple closed-form equations I used here allowed me to explore various design considerations at a high level of abstraction, using arbitrary service time distributions as long as their mean and variance are finite. □

References

1. D.A. Menascé and V.A.F. Almeida, *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning*, Prentice Hall, Upper Saddle River, N.J., 2000.
2. D.A. Menascé and V.A.F. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*, Prentice Hall, Upper Saddle River, N.J., 2002.
3. L. Kleinrock, *Queueing Systems: Volume I: Theory*, John Wiley & Sons, New York, 1975.
4. J.C. Little, "A Proof of the Queuing Formula $L = \lambda W$," *Operations Res.*, vol. 9, no. 3, 1961, pp. 383-387.
5. M. Andreolini, M. Colajanni, and R. Morselli, "Performance Study of Dispatching Algorithms in Multi-tier Web Architectures," *ACM Sigmetrics Performance Evaluation Rev.*, vol. 30, no. 2, Sep. 2002.
6. V. Cardellini, M. Colajanni, and P.S. Yu, "Dynamic Load Balancing on Web Server Systems," *IEEE Internet Computing*, May/June 1999, pp. 28-39.
7. M. Arlitt and C. Williamson, "Web Server Workload Characterization: The Search for Invariants," *Proc. 1996 ACM Sigmetrics Conf. Measurement & Modeling of Computer Systems*, ACM Press, New York, 1996, pp. 126-137.
8. M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *Proc. 1996 ACM Sigmetrics Conf. Measurement & Modeling of Computer Systems*, ACM Press, New York, 1996, pp. 160-169.

Daniel Menascé is a professor of computer science, codirector of the E-Center for E-Business, and director of the master's in e-commerce program at George Mason University. He received a PhD in computer science from UCLA and has published five books, including *Capacity Planning for Web Services: Metrics, Models, and Methods* (Prentice Hall, 2002) and *Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning* (Prentice Hall, 2000). He is a fellow of the ACM and a recipient of the A.A. Michelson Award from the Computer Measurement Group.