

ColdFusion 4.5.1 Windows NT Performance Fact Sheet and Sizing Guide



About ColdFusion

ColdFusion is a high-performance, scalable, open platform for delivering Web applications that range from simple database-driven pages to full e-commerce solutions on intranets, extranets and the Internet.

First released in 1995, ColdFusion has become the leading cross-platform Web application server. ColdFusion was the first Web application server on Windows NT, and as a pioneer in the application server market, ColdFusion has established a broad base of support with thousands of customers from small consulting groups to multinational corporations. Used by almost half of the Fortune 500, ColdFusion provides the fastest way to develop and deliver scalable enterprise and e-commerce applications.

Introduction

This document outlines performance and configuration considerations of a ColdFusion Web application. It summarizes testing that Allaire performed in its Enterprise Scalability Lab and details several examples of sizing ColdFusion Web application deployments. It includes the results of performance and scalability tests on the ColdFusion "Tack2++" application, a version of the "Tack2Plus" sample application included with ColdFusion, as well as configuration information and a copy of the test script actions. These tests can serve as a general guide for customers deploying ColdFusion-built applications. Customers can use this Performance Fact Sheet to assist with deployment planning to help ensure that sufficient resources are available for successful deployment.

Performance & Scalability Testing

Web site deployments vary in size and complexity, from several servers to several hundred servers, depending on many factors. ColdFusion deployment scenarios typically involve multiple machine configurations, facilitated by the built-in server-clustering capabilities of ColdFusion Server Enterprise edition. The clustering capabilities of ColdFusion Enterprise provide a high degree of scalability and fault-tolerance with dynamic load balancing, and network and service-level failover.

Testing was performed on 1, 2, 4, and 8 Compaq dual-processor 1850R machines¹ running ColdFusion Server 4.5.1, and 2 quad-processor Compaq 6400R machines in a Compaq DISA configuration running MS SQL Server 7.0 Enterprise Edition as the database server. The 2, 4, and 8 machine cluster configuration was performed to provide examples of:

- 1) ColdFusion's linear scalability
- 2) Sample test data from which customers can extrapolate for use as a general guide when sizing and deploying ColdFusion application deployment configurations

Customers using these results to size their ColdFusion application server environments must test their own applications using an enterprise-level load-testing tool. This step is critical to determine sizing requirements for their specific application. Effective load testing techniques will determine the applications' performance and scalability characteristics. It can also help find and remove application and site performance bottlenecks, and test the application's "wide" (multiple machine) vs. "deep" (multiple processors) scaling characteristics. Widely varying traffic patterns, peak traffic periods, and future growth must also be taken into account to properly size a production server configuration prior to deployment.

¹ Single machine results are provided for completeness. In addition to the Compaq model 1850R, the Compaq model 6400R server also makes an excellent NT ColdFusion server machine and is 4-processor capable.

ColdFusion 4.5.1 Performance Fact Sheet and Sizing Guide

Customers are strongly urged to load test as soon as a working prototype of the application is available and should continue load testing through development on the entire working application before it is placed into production.

For customers wishing assistance with this process, Allaire Consulting offers a 5-day onsite *Performance Analysis & Tuning* consultation, which can help ensure the success of application deployment. Allaire Training also offers a 2-day *Performance Analysis and Tuning* course designed to help customers gain the necessary knowledge to properly load test their applications, identify and fix site bottlenecks and help ensure deployment success.

The key variables measured or controlled in the load testing performed for this Fact Sheet were:

Page Response Time

The time experienced by the Web browser from URL submission until all page elements have been downloaded and page rendering is complete.

Simultaneous Users

Concurrent site users simulated using the Segue SilkPerformer 3.5.1 load-testing tool.

Number of Servers

The number of Spectra server machines that make up a cluster of servers.

Spectra Requests/time – The number of Spectra requests processed per second, minute, hour, and 24-hour period.

Page Views/time

The number of browser pages displayed per second, minute, hour, and 24-hour period.

HTTP Hits/time

The number of browser HTTP hits experienced per second, minute, hour, and 24-hour period.

The ColdFusion Tack2++ Web Application

The Tack2Plus sample application included with ColdFusion is a reasonable, although uncomplicated, ColdFusion e-Commerce Web application for a fictitious company selling surf and sail-related gear on the Internet. It was tested to provide data that customers can use as a starting point for sizing their production server clusters.

The Tack2++ application tested for this Fact Sheet is a new version of the original Tack2Plus sample application included with ColdFusion. It preserves all of the functional characteristics of the original Tack2Plus application, but makes use of ColdFusion's template and query caching features, as would a real-world production ColdFusion Web application.

The Tack2++ application was load tested in order to test capacity in terms of both application throughput and number of supportable simultaneous users. While results will vary with customer applications of varying scope and complexity, customers can expect to realize similar results with similar applications and server configurations.

In our testing, four types of users were simulated during the load testing to model, as closely as possible, the real-world traffic mix that an e-Commerce site is likely to experience. These user types are described below:

User1: Brief, curious visitors who visit the site's main page, enter the site, then leave.

User2: Browsers, who visit, browse and drill into several product categories, then leave.

User3: Shoppers who visit, browse product categories, perform a site search, add a few items to their shopping cart, then leave, abandoning their shopping cart.

User4: Committed shoppers, who visit, browse product categories, perform a site search, add items to their shopping cart, commit to checkout and place their order for purchase before leaving the site.

The tests were performed with the following user load mix, expressed as a percentage of the total load:

User1: 25%

User2: 25%

User3: 25%

User4: 25%

The user load mix used in this round of testing can be considered more demanding on the server than a typical load mix normally experienced on public e-commerce sites. Studies have shown approximately 90% of all traffic consists of browsing activities, and less than 5% of all users actually add items to the shopping cart and checkout. It is important to note that although the test was setup to disperse transactions with each having a 25% chance of occurrence, the randomization feature of the testing tool caused certain transactions to occur more often than others.

Test Randomization Methodology

The transaction mix was weighted such that all transactions had a 25% chance of execution until the simulation period is reached. Additionally, each simulated user was configured to report only their total transaction time and their individual per-page timings to reduce the impact of any overhead that the tool might experience while executing the test. In addition to executing the tests with a random selection of transactions, we used a Random Think Time between each user's visit to the pages. This random think time was implemented using an exponential distribution, seeded with a mean value of 14.0 seconds per page, calculated for each page visit. The fourteen-second timing that we used was based on the average time it took for a selection of our employees to fully read the contents of each page. The exponential distribution of those times represents the odds that they would actually read the entire pages contents or not read them, remember them or inspect them for updates. Using an exponential distribution more than 75% of all instances should fall within less than ½ standard deviation of the mean (in this case seed) and that the outlying instances should be extremely far from the mean, best to simulate a dropped session or interruption in web surfing.

ColdFusion Clustered NT Server Test Results

The ColdFusion Tack2++ application proved linearly scalable through the 8-machine cluster test. That is, page response time grew in direct proportion to the amount of load applied in the tests. The application serviced 5,440 simultaneous clients at a rate of about 66 million HTTP hits, or about 25 million ColdFusion-generated page views per day. At this rate, the average page response time was about 4 seconds, with server requests evenly distributed throughout a 24-hour day. This translates to a rate of roughly 1.9 billion HTTP hits, or 763 million page views per month, with server requests evenly distributed throughout a 30-day month.

The test data also reveals that ColdFusion request throughput remained nearly constant at this rate while maintaining an average page response time of about 1 second for 4,280 simultaneous users, 2 seconds for 4,720 simultaneous users, and 8 seconds for 6,720 simultaneous users.

Figure 1 illustrates the linear scalability of the application in HTTP hits serviced per day for 1, 2, 4, and 8 servers with an average page response time of about 1 second.

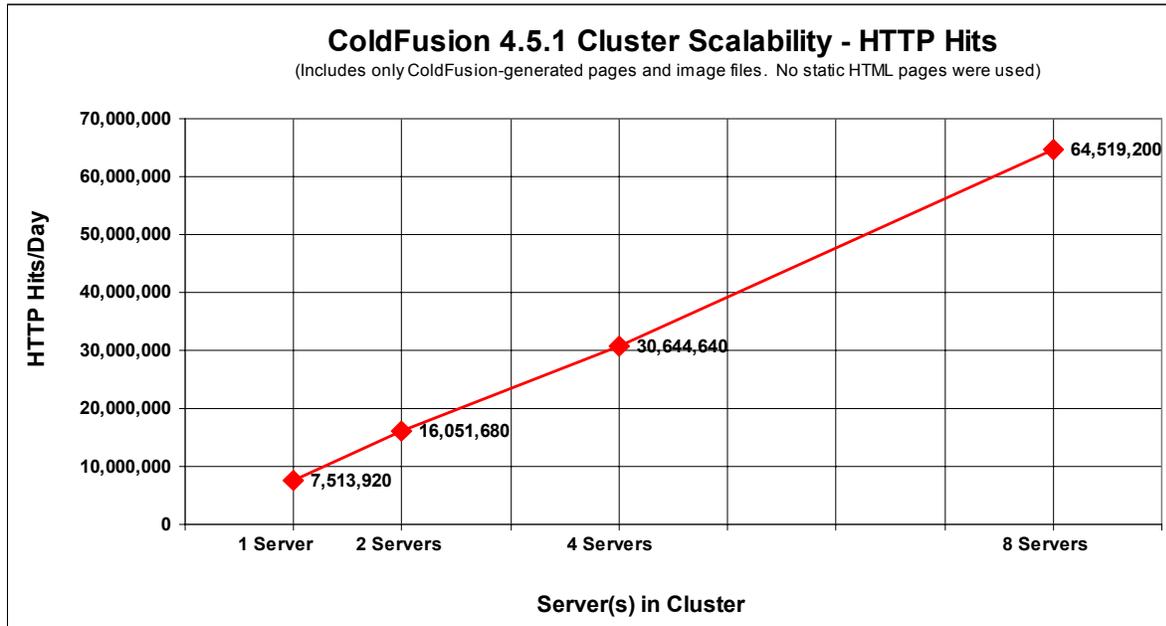


Figure 1 - Tack2++ Cluster HTTP Throughput

Figure 2 illustrates the linear scalability of the application in terms of page views serviced per day for 1, 2, 4, and 8 servers.

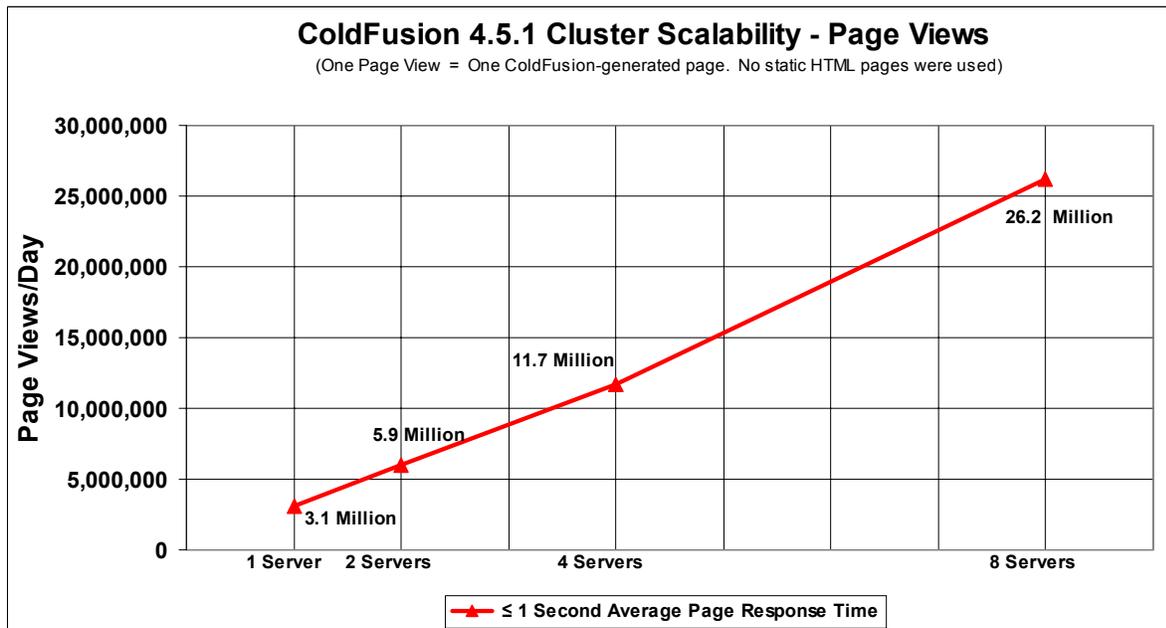


Figure 2 - Tack2++ Cluster Page View Throughput

Figure 3 illustrates the linear scalability of ColdFusion as the number of simultaneous users able to be serviced for each response time target threshold (1, 2, 4, and 8 seconds) for 1, 2, 4, and 8 servers.

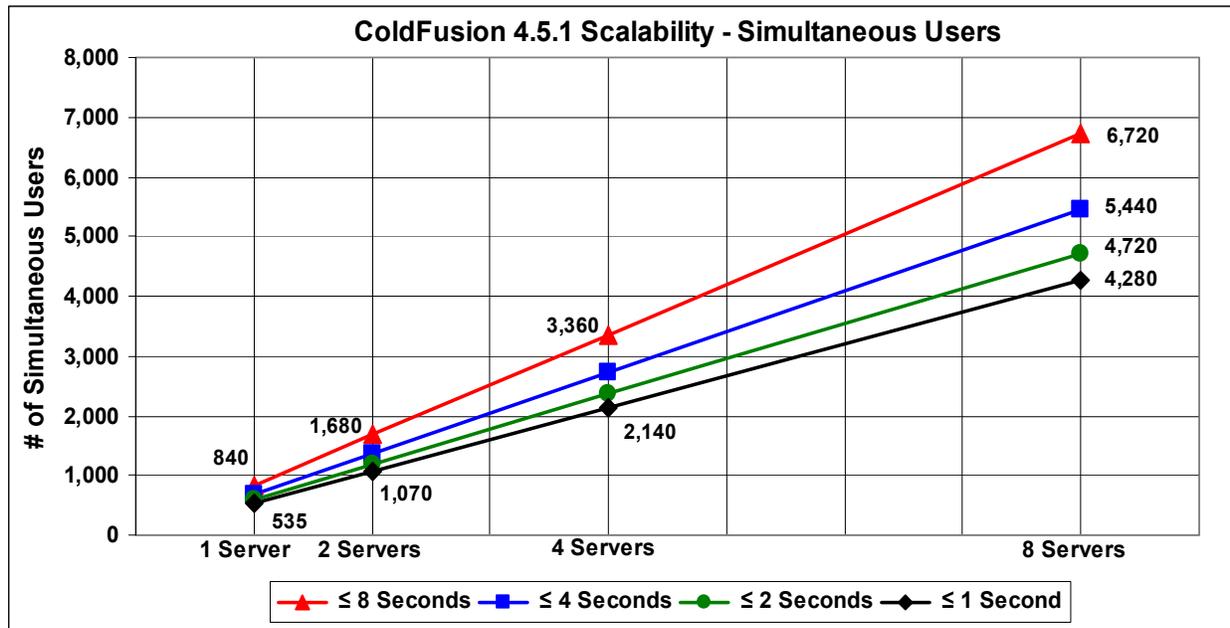


Figure 3 - Tack2++ Cluster Simultaneous Users

Figures 4 and 5 detail the summarized load testing data for the clustered server Tack2++ NT tests. Since the Tack2++ application consists of exactly one ColdFusion template request for each application page view (i.e. no raw HTML pages or HTML framesets are used), the number of ColdFusion request is equal to the number of page views. The data for ColdFusion requests and page views reflect this one-to-one relationship.

Tack2++ Cluster Load Test Data - ColdFusion 4.5.1 NT								
Target Page Resp. Time (s):	1 Server				2 Server Cluster			
	1	2	4	8	1	2	4	8
# Virtual Users:	535	590	680	840	1,070	1,180	1,360	1,680
SECTION 1 Page Response Time (s):	1.08	2.07	3.76	7.93	0.93	1.98	4.07	7.92
HTTP Hits/Sec:	87.0	87.9	89.1	91.8	185.8	189.1	189.6	190.3
HTTP Hits/Min:	5,218	5,271	5,345	5,508	11,147	11,348	11,377	11,415
HTTP Hits/Hour:	313,080	316,260	320,700	330,480	668,820	680,868	682,620	684,900
HTTP Hits/Hour X 24:	7,513,920	7,590,240	7,696,800	7,931,520	16,051,680	16,340,832	16,382,880	16,437,600
Page Views/Sec:	36.2	36.0	50.5	50.7	69.4	72.4	72.6	73.3
Page Views/Min:	2,171	2,163	3,029	3,040	4,165	4,342	4,357	4,396
Page Views/Hour:	130,276	129,769	181,751	182,391	249,920	260,499	261,401	263,733
Page Views/Hour X 24:	3,126,613	3,114,453	4,362,027	4,377,387	5,998,080	6,251,983	6,273,627	6,329,600
CF Requests/Sec:	36.2	36.0	50.5	50.7	69.4	72.4	72.6	73.3
CF Requests/Min:	2,171	2,163	3,029	3,040	4,165	4,342	4,357	4,396
CF Requests/Hour:	130,276	129,769	181,751	182,391	249,920	260,499	261,401	263,733
CF Requests/Hour X 24:	3,126,613	3,114,453	4,362,027	4,377,387	5,998,080	6,251,983	6,273,627	6,329,600

Figure 4 - Tack2++ (1) and (2) Server Cluster Load Test Data

Tack2++ Cluster Load Test Data - ColdFusion 4.5.1 NT								
Target Page Resp. Time (s):	4 Server Cluster				8 Server Cluster			
	1	2	4	8	1	2	4	8
# Virtual Users:	2,140	2,360	2,720	3,360	4,280	4,720	5,440	6,720
SECTION 1 Page Response Time (s):	0.94	2.01	4.00	8.08	1.01	2.09	3.91	8.11
HTTP Hits/Sec:	354.7	359.6	368.1	366.8	746.8	742.3	768.6	778.2
HTTP Hits/Min:	21,281	21,577	22,084	22,009	44,805	44,539	46,117	46,693
HTTP Hits/Hour:	1,276,860	1,294,620	1,325,040	1,320,540	2,688,300	2,672,340	2,767,020	2,801,580
HTTP Hits/Hour X 24:	30,644,640	31,070,880	31,800,960	31,692,960	64,519,200	64,136,160	66,408,480	67,237,920
Page Views/Sec:	135.8	142.0	148.6	150.3	303.6	267.2	277.9	285.4
Page Views/Min:	8,148	8,522	8,915	9,020	18,215	16,034	16,674	17,122
Page Views/Hour:	488,886	511,330	534,870	541,183	1,092,887	962,056	1,000,435	1,027,330
Page Views/Hour X 24:	11,733,257	12,271,923	12,836,887	12,988,385	26,229,289	23,089,338	24,010,448	24,655,911
CF Requests/Sec:	135.8	142.0	148.6	150.3	303.6	267.2	277.9	285.4
CF Requests/Min:	8,148	8,522	8,915	9,020	18,215	16,034	16,674	17,122
CF Requests/Hour:	488,886	511,330	534,870	541,183	1,092,887	962,056	1,000,435	1,027,330
CF Requests/Hour X 24:	11,733,257	12,271,923	12,836,887	12,988,385	26,229,289	23,089,338	24,010,448	24,655,911

Figure 5 - Tack2++ (4) and (8) Server Cluster Load Test Data

ColdFusion Single NT Server Test Results

On a single NT machine, the Tack2++ application proved capable of servicing 840 simultaneous users with an average page response time of 7.93 seconds. The Tack2++ application also maintained an average page response time of less than 1.0 second as load increased to beyond 535 simultaneous users.

Figure 6 below illustrates the response time measured as the number of simultaneous users increased from 1 to 840 for a single Compaq 1850R with 2 x 500Mhz processors and 512MB RAM.

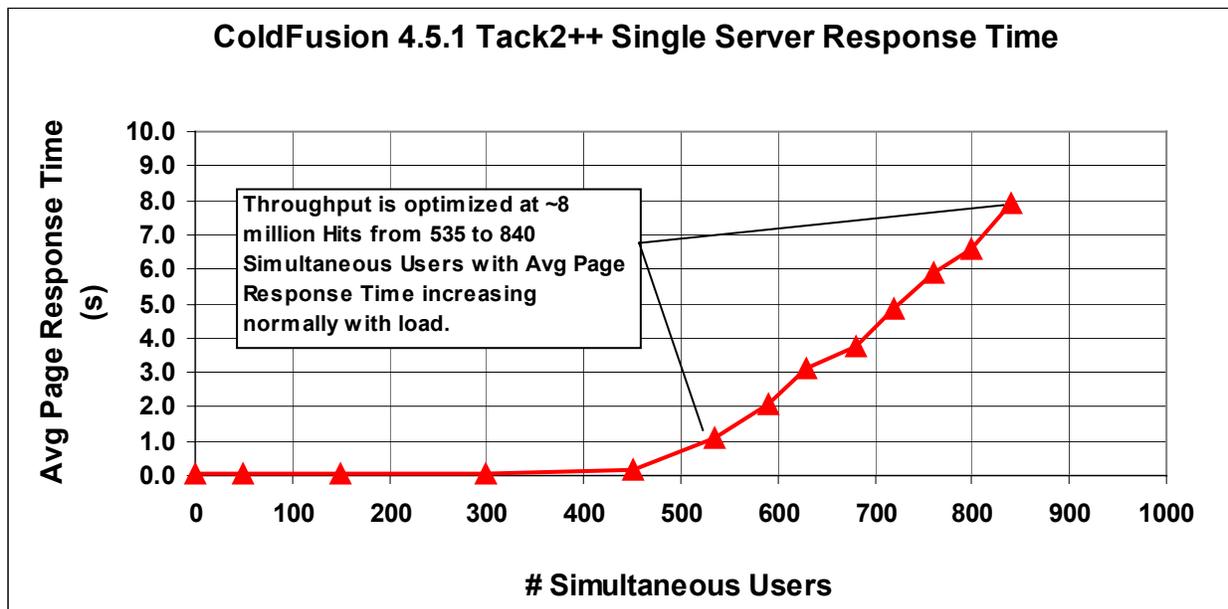


Figure 6 - Tack2++ Single Server Response Time

Figure 7 below illustrates single server HTTP throughput as the number of simultaneous users increased from 1 to 840. Throughput reached the optimal range of about 8 million HTTP hits at 535 simultaneous users and remained relatively constant through the 840 simultaneous user load level.

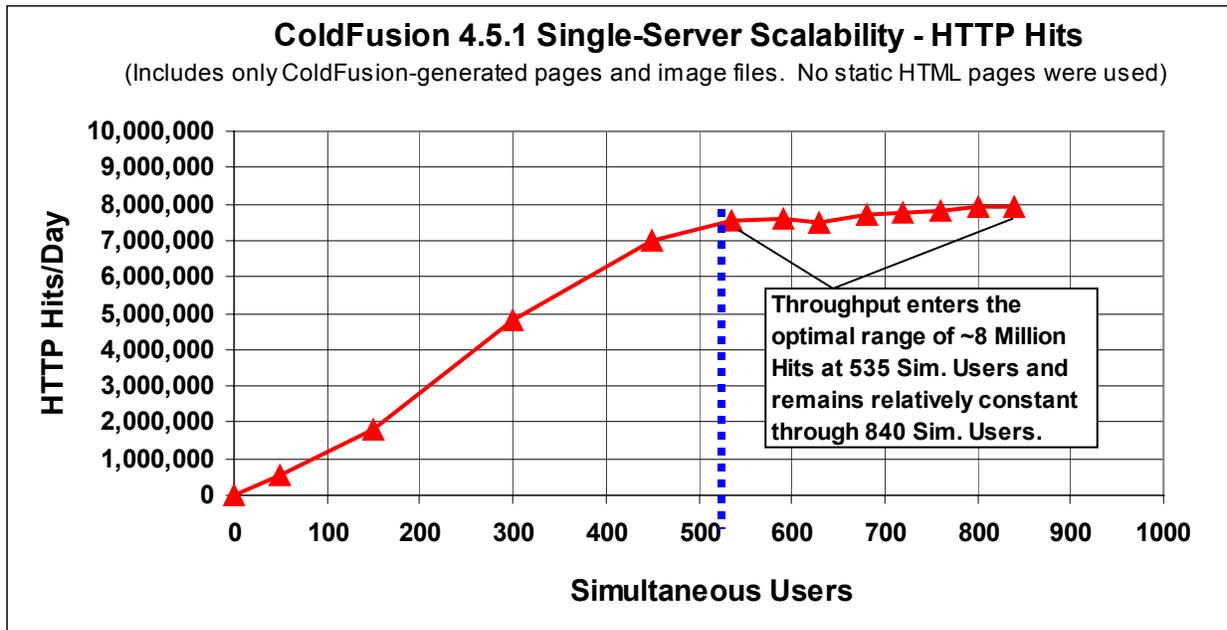


Figure 7 - Tack2++ Single Server HTTP Throughput

Figure 8 illustrates single server page view throughput as the number of simultaneous users increased from 1 to 840. Throughput reached the optimal range of about 3.2 million page views at 535 simultaneous users and remained relatively constant through the 840 simultaneous user load level.

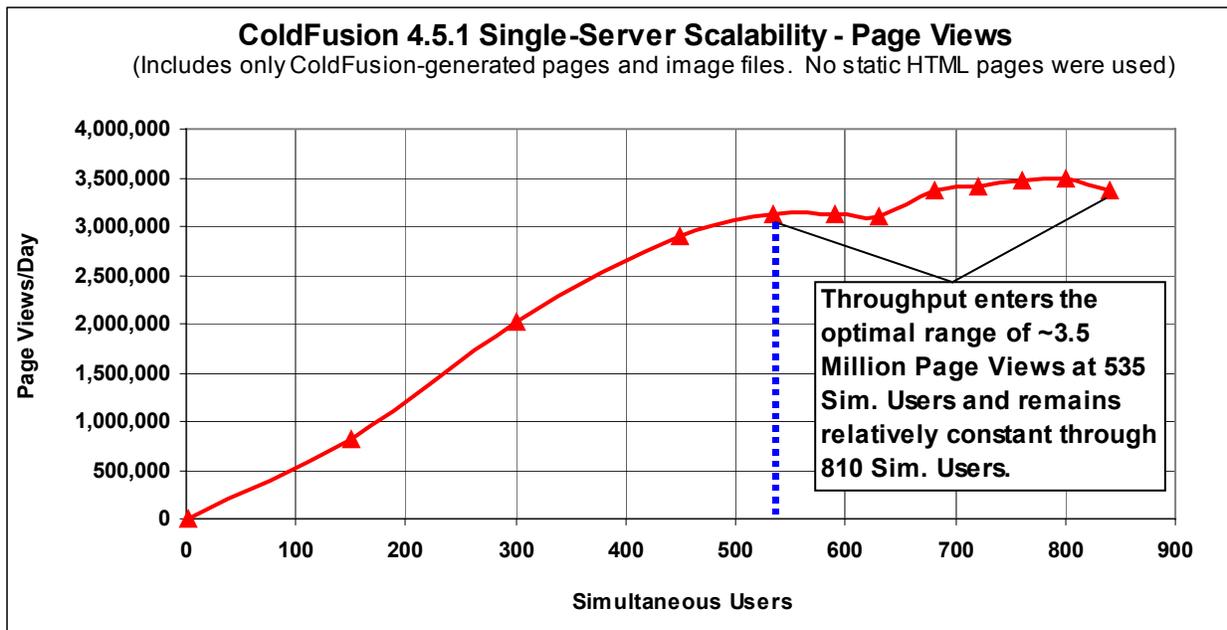


Figure 8 - Tack2++ Single Server Page View Throughput

Figure 9 details the load testing data summary for the single server NT Tack2++ tests. The test data reveals that a single Compaq 1850R dual-processor NT server capable of servicing about 7.8 million HTTP hits, or about 3.2 million page views per day, with server requests evenly distributed throughout the day. This translates to a rate of roughly 234 million HTTP hits, or 98 million page views per month, with server requests evenly distributed throughout the month.

This rate of requests is achieved while maintaining sub-second (≤ 1.0 second average page response time, and servicing 535 simultaneous users. Again, the optimal throughput range of ~3.2 million page views, or ~7.5 million HTTP hits is realized at about 535 simultaneous users.

ColdFusion 4.5.1 - Tack2++ Single-Machine Load Test Data - Windows NT								
					~1 Sec.	~2 Sec.	~4 Sec.	~8 Sec.
Avg Page Response Time (s):	0.04	0.03	0.04	0.04	1.08	2.07	3.76	7.93
# Virtual Users:	1	50	150	300	535	590	680	840
HTTP Hits/Sec:	0.2	6.6	20.8	55.8	87.0	87.9	89.1	91.8
HTTP Hits/Min:	12	393	1,245	3,349	5,218	5,271	5,345	5,508
HTTP Hits/Hour:	720	23,580	74,700	200,940	313,080	316,260	320,700	330,480
HTTP Hits/Hour X 24:	17,280	565,920	1,792,800	4,822,560	7,513,920	7,590,240	7,696,800	7,931,520
Page Views/Sec:	0.1	3.2	9.5	23.4	36.2	36.0	38.9	39.1
Page Views/Min:	6	190	572	1,404	2,171	2,163	2,337	2,345
Page Views/Hour:	348	11,410	34,331	84,213	130,276	129,769	140,208	140,702
Page Views/Hour X 24:	8,344	273,837	823,951	2,021,120	3,126,613	3,114,453	3,364,992	3,376,841
CF Requests/Sec:	0.1	3.2	9.5	23.4	36.2	36.0	38.9	39.1
CF Requests/Min:	6	190	572	1,404	2,171	2,163	2,337	2,345
CF Requests/Hour:	348	11,410	34,331	84,213	130,276	129,769	140,208	140,702
CF Requests/Hour X 24:	8,344	273,837	823,951	2,021,120	3,126,613	3,114,453	3,364,992	3,376,841

Figure 9 - Tack2++ Single Server Test Data

Using the ColdFusion Tack2++ Test Results for Product Server Sizing

Customers whose ColdFusion applications have characteristics similar to those of the Tack2++ application apply these results directly to assist in sizing their production server configurations. However, customers whose ColdFusion applications have characteristics dissimilar to those of the ColdFusion Tack2++ application can also use these test results. This can be accomplished by factoring into account the performance degradation or improvement the application demonstrates relative to the ColdFusion Tack2++ application.

Sizing Methodology

By arriving at one of the key variables measured or controlled in the load testing performed for this Fact Sheet, and knowing the desired average page response time, you can Look up or extrapolate the estimated number of servers likely to be required to service a given ColdFusion application's anticipated load.

The sizing methodology used in the sizing examples can be summarized in five basic steps:

- Step 1: Select a load metric to use for sizing.
- Step 2: Estimate the peak-period load the site is expected to experience.
- Step 3: Adjust the peak-period load estimate.
- Step 4: Identify a peak-period page response-time limit.
- Step 5: Look up the estimated number of servers required from ColdFusion Tack2++ test data.

Below is an overview of the sizing methodology:

Step 1: Select a preferred load metric to use for sizing:

- a) Page views
- b) HTTP hits, or
- c) Simultaneous users

Customers with existing load estimates based on other metrics, such as unique site visitors per month, may need to gather additional data or perform additional calculations to arrive at one of these metrics to use this sizing methodology effectively.

Step 2: Estimate the peak-period load the site is expected to experience:

- a) Page Views
Estimate the average number of page views per hour the Web site will need to support during the hour of peak site load.

For example, 120,000 page views per hour during the hour of peak site load.

- b) HTTP Hits
Estimate the number of HTTP hits per hour the Web site will need to support during the hour of peak site load

For example, 350,000 HTTP hits per hour during the hour of peak site load.

- c) Simultaneous Users
Estimate the number of people (users) who will use the Web site concurrently during the hour of peak site load.

For example, 800 simultaneous users during the hour of peak site load.

Customers may need to convert their existing page view and HTTP hit load estimates from monthly, weekly, or daily to hourly to use this methodology effectively. One possible approach to this conversion is shown below:

Peak load page view conversion:

$$Page\ views\ per\ hour = \left(\frac{est.\ page\ views\ per\ month}{avg\ num\ days\ per\ month} \right) / \# peak\ hours\ per\ day$$

For example:

$$\begin{aligned} Page\ views\ per\ hour &= \left(\frac{18,000,000\ page\ views\ per\ month}{30\ days\ per\ month} \right) / 6\ peak\ hours\ per\ day \\ &= 600,000 / 6 \\ &= 100,000 \end{aligned}$$

Peak load HTTP hit conversion:

$$HTTP \text{ hits per hour} = \left(\frac{\text{est. HTTP hits per month}}{\text{avg num days per month}} \right) / \# \text{ peak hours per day}$$

For example:

$$\begin{aligned} HTTP \text{ hits per hour} &= \left(\frac{63,000,000 \text{ HTTP hits per month}}{30 \text{ days per month}} \right) / 6 \text{ peak hours per day} \\ &= 2,100,000 / 6 \\ &= 350,000 \end{aligned}$$

Step 3: Adjust the peak-period load estimate:

If the ColdFusion application being sized is more complex (more resource-intensive) or less complex (less resource-intensive) than the ColdFusion Tack2++ application, the peak-period load estimate can be multiplied by an “adjustment factor” to help account for its performance characteristics when using the Tack2++ load test data.

For example, the following adjustment factors might be used, depending on site complexity and resource utilization:

Resource Utilization of Application Being Sized	Adjustment Factor	Example Adjusted Peak Load for 100,000 Page Views/Hour	Example Adjusted Peak Load for 350,000 HTTP Hits/Hour	Example Adjusted Peak Load for 800 Simultaneous Users
Extra Light	0.25	25,000	87,500	200
Light	0.5	50,000	175,000	400
Medium (Tack2++)	1.0	100,000	350,000	800
Medium-Heavy	2.0	200,000	700,000	1,600
Heavy	4.0	400,000	1,400,000	3,200

In these examples a peak-period load of 100,000 page views per hour, 350,000 HTTP hits per hour and 800 simultaneous users is multiplied by the adjustment factor to yield the adjusted peak-period load.

This value may more accurately reflect the difference between the ColdFusion Tack2++ application data and the application being sized.

Any number of additional adjustments to the peak load estimate can be made to account for application-specific attributes different from those of the Tack2++ application tested, such as:

- Number of HTTP requests, HTML frames, or ColdFusion requests per page view
- Number of DB queries per page view
- Other resource-intensive or prolonged external protocol calls to external servers, etc.

Step 4: Identify a peak-period page response-time limit:

Determine a limit on how many seconds a site user may have to wait for a page during periods of peak load. For example: 1, 2, 4, or 8 seconds. Generally, the lower the tolerable response-time limit is set, the more servers and resources are required to sustain the page response-time limit.

Step 5: Look up the estimated number of servers required from ColdFusion Tack2++ test data:

Using the adjusted peak-period load estimate and the desired page response-time, look up the number of servers required from the ColdFusion Tack2++ cluster load test data results provided in this Fact Sheet.

Since the ColdFusion Tack2++ application proved scalable across server machines, an alternative sizing technique is to:

- a) Look up the desired page response-time in the ColdFusion Tack2++ single server test results data,
- b) Determine what load level the single server was capable of handling with this response time, and
- c) Calculate the estimated number of servers required by dividing the single server load capacity into the adjusted peak-period load estimate, as:

$$\text{Number of servers} = \frac{\text{Adjusted peak period load per hour}}{\text{Single server load capacity per hour for desired responsetime}}$$

For example, from Figure 9, a single server running ColdFusion Tack2++ proved capable of servicing 134,988 page views per hour with a response time of about 4 seconds.

If the adjusted peak-period load estimate is 550,000 page views per hour, then:

$$\begin{aligned} \text{Number of servers} &= \frac{550,000}{134,988 \text{ (for 4 second page response time)}} \\ &= 4.07 \text{ (or about 5 servers)} \end{aligned}$$

This second approach can be useful if the desired load level and response time is not directly listed in the test data from the ColdFusion Tack2++ cluster tests. This can happen when the number of servers required exceeds the eight machines tested.

The examples that follow should be used only as a general guide for performing production server sizing. Customers must perform similar load testing and sizing exercises using load test data from their specific ColdFusion applications to ensure accurate server sizing. Sizing exercises should be performed on a regular basis once a site is placed into production, and should factor in existing load patterns and growth projections to ensure successful load handling and a satisfied user base.

Customers should always consider peak-period loads in their estimates, allowing for occasional load spikes, which can often occur unexpectedly, but have a dramatic, and often lasting, negative effect on users' perceptions of the site.

EXAMPLE 1: Using Page Views

A customer is deploying a ColdFusion application and expects to receive 70 million page views per month. The customer wants to maintain an average page response time of no more than about 4 seconds during peak-periods. They are expecting to see most of this load distributed somewhat evenly throughout a 6-hour period each day of the month.

The application makes extensive use of logging and database queries. The customer believes the application makes “heavy” use of server resources compared to the ColdFusion Tack2++ application tested in this fact sheet.

To estimate the number of servers required to adequately service this load within the desired response time periods during peak load:

Step 1: Select a load metric to use for sizing:

The customer has estimated load in page views.

Step 2: Estimate the peak-period load the site is expected to experience:

$$\begin{aligned} \text{Page views per hour} &= \left(\frac{70,000,000 \text{ page views per month}}{30 \text{ days per month}} \right) / 6 \text{ peak hours per day} \\ &= 2,333,333 / 6 \\ &= 388,889 \end{aligned}$$

Step 3: Adjust the peak-period load estimate:

Using the example adjustment factors provided earlier:

$$\begin{aligned} \text{Adj. page views per hour} &= \text{est. page views per hour} \times \text{adjustment factor} \\ &= 388,889 \times 4.0 \text{ (adjustment factor for "heavy")} \\ &= 1,555,556 \end{aligned}$$

Step 4: Identify a peak-period page response-time limit:

The customer has identified 4 seconds as the desired peak-period page response-time limit.

Step 5: Look up the estimated number of servers required from ColdFusion Tack2++ test data:

Using the ColdFusion Tack2++ cluster test data in Figure 3 and Figure 4, an attempted Look up of 1,555,556 page views per hour at 4 seconds reveals the 8 server cluster tested serviced 1,079,978 page views at 4 seconds, less than required for this customer's application.

Since the ColdFusion Tack2++ application proved scalable across server machines, the estimated number of servers can be calculated by dividing the single server load capacity for 4 seconds from Figure 9 (134,988) into the adjusted peak-period load estimate:

$$\text{Number of Servers} = \frac{1,555,556}{134,988 \text{ Page Views (for 4 second page response time)}}$$

$$= 11.52 \text{ (or about 12 servers)}$$

EXAMPLE 2: Using HTTP Hits

A customer is deploying a ColdFusion application and their best estimates put the expected load at a rate of 225 million HTTP hits per month. The customer wants to maintain an average server page response time of no more than about 4 seconds during peak-periods. They are expecting to see most of this load distributed somewhat evenly throughout a 6-hour period each day of the month.

The customer believes the application makes “medium-heavy” use of server resources compared to the ColdFusion Tack2++ application tested in this Fact Sheet.

To estimate the number servers required to adequately service this load within the desired response time periods during peak load:

Step 1: Select a load metric to use for sizing:

The customer has estimated load in HTTP hits.

Step 2: Estimate the peak-period load the site is expected to experience:

$$\begin{aligned} \text{HTTP Hits per hour} &= \left(\frac{225,000,000 \text{ HTTP Hits per month}}{30 \text{ days per month}} \right) / 6 \text{ Peak Hours per day} \\ &= 7,500,000 / 6 \\ &= 1,250,000 \end{aligned}$$

Step 3: Adjust the peak-period load estimate:

Using the example adjustment factors provided earlier:

$$\begin{aligned} \text{Adj. HTTP hits per hour} &= \text{est. HTTP hits per hour} \times \text{adjustment factor} \\ &= 1,250,000 \times 2.0 \text{ (adjustment factor for "medium-heavy")} \\ &= 2,500,000 \end{aligned}$$

Step 4: Identify a peak-period page response-time limit:

The customer has identified 4 seconds as the desired peak-period page response-time limit.

Step 5: Look up the estimated number of servers required from ColdFusion Tack2++ test data:

Using the ColdFusion Tack2++ cluster test data in Figure 3 and Figure 4, a Look up of 2,500,000 HTTP hits per hour at 4 seconds reveals the 8 server cluster tested serviced 2,664,432 HTTP hits per hour at 4 seconds, adequate for this customer’s application.

EXAMPLE 3: Using Simultaneous Users

A customer is deploying a ColdFusion application and their best estimates put the expected load at a rate of 6,000 simultaneous site users during periods of peak load. The customer wants to maintain an average page response time of no more than about 2 seconds during peak-periods.

The customer believes the application makes “light” use of server resources compared to the ColdFusion Tack2++ application tested in this Fact Sheet.

To estimate the number servers required to adequately service this load within the desired response time periods during peak load:

Step 1: Select a load metric to use for sizing:

The customer has estimated load in simultaneous users.

Step 2: Estimate the peak-period load the site is expected to experience:

The customer has estimated load at a rate of 6,000 simultaneous site users during periods of peak load.

Step 3: Adjust the peak-period load estimate:

Using the example adjustment factors provided earlier:

$$\begin{aligned} \text{Adj. sim users} &= \text{est. sim users} \times \text{adjustment factor} \\ &= 6,000 \times 0.5 \text{ (adjustment factor for "light")} \\ &= 3,000 \end{aligned}$$

Step 4: Identify a peak-period page response-time limit:

The customer has identified 2 seconds as the desired peak-period page response-time limit.

Step 5: Look up the estimated number of servers required from ColdFusion Tack2++ test data:

Using the ColdFusion Tack2++ cluster test data in Figure 3 and Figure 4, a Look up of 3,000 simultaneous users at 2 seconds reveals that the 4-server cluster serviced 2,360 simultaneous users at 2 seconds, and the 8-server cluster serviced 4,720 simultaneous users at 2 seconds. The 8-server cluster would then be capable of servicing the required 3,000 simultaneous users with an average 2-second page response time.

However, calculating the number of servers required using the single server statistics in Figure 9 would provide a more granular cluster-sizing estimate.

Since the ColdFusion Tack2++ application proved scalable across server machines, the estimated number of servers can be calculated by dividing the single server load capacity for 2 seconds from Figure 9 (575) into the adjusted peak-period load estimate:

$$\text{Number of Servers} = \frac{3,000}{575 \text{ Simul Users (for 2 second page response time)}}$$

= 5.23 (*or about 6 servers*)

Conclusion

This fact sheet has outlined the performance and production sizing configuration considerations of a ColdFusion application. It has provided examples of the linear scalability of an Allaire ColdFusion 4.5.1 application with test data that customers can use as a general guide in the initial stages of deployment and capacity planning.

To ensure a successful deployment, however, it is important that customers test their own applications using an enterprise-level load-testing tool to determine specific sizing requirements for their applications. Customers should perform load testing as soon as a working prototype of the application is available, and load testing should include the entire application before it is placed into production.

Testing Details – Additional Observations

Here are some additional observations made during the testing at the maximum tested load of 6,720 simultaneous users with 8 ColdFusion servers.

ColdFusion / Web server machines:

CPU Utilization:

- ~94-100%

Memory Utilization:

- ~287MB used out of 512MB physical RAM on machine
- ColdFusion Server: ~129MB
- IIS 4.0 Web Server: ~95MB

Physical Disk I/O:

- ~1-2%

Database Server Machine:

CPU Utilization:

- ~59-68%

Memory Utilization:

- ~350MB used out of 1GB physical RAM on machine
- MS SQL Server 7.0: ~310MB

Physical Disk I/O:

- ~50%

Network Utilization:

- ~6-13% utilization of 100MBs, as reported by MS NetMon (from the MS SMS Resource Kit)

Testing Details – Server Hardware/Software:

ColdFusion/Web Servers:

1, 2, 4, and 8 Compaq 1850R, dual-500MHz Intel processors, 512MB machines, running:

- o Microsoft Windows NT 4.0 Server, SP4 and IIS 4.0
- o ColdFusion 4.5.1 Server NT Enterprise
- o Microsoft Data Access Components (MDAC) 2.1 GA

The Tack2++ application resided locally in the Web root of each ColdFusion Server machine.

SQL Server:

1 “active” Compaq 6400R, 4 x 500MHz PIII Xeon Intel processors, 1GB RAM machine configured in a Compaq Distributed Internet Server Array (DISA) Architecture, running:

ColdFusion 4.5.1 Performance Fact Sheet and Sizing Guide

- Microsoft Windows NT 4.0 Server Enterprise Edition
- Microsoft SQL Server 7.0 SP1 Enterprise Edition

1 “passive” Compaq 6400R, 4 x 500MHz PIII Xeon Intel processors, 1GB RAM machine configured in a Compaq Distributed Internet Server Array (DISA) Architecture, running:

- Microsoft Windows NT 4.0 Server Enterprise Edition
- Microsoft SQL Server 7.0 SP1 Enterprise Edition

More information on the DISA Architecture can be found here:

<http://www.compaq.com/solutions/internet/disa-summary-faq.html>

Testing Details – Load Generation

Loads were generated from a selection of the following machines

Compaq 1850R, 2 x 500MHz Intel processors, 512MB machines running

- Microsoft Windows NT 4.0 Server, SP4
- Segue SilkPerformer 3.5.1 Agent on all machines (up to 500 Virtual Users/machine)
- Microsoft Data Access Components (MDAC) 2.1 GA

Compaq Prosignia Server 720, 1 x 500MHz Intel processors, 512MB machines running

- Microsoft Windows 2000 Server
- Segue SilkPerformer 3.5.1 Agent on all machines (up to 500 Virtual Users/machine)
- Segue SilkPerformer 3.5.1 Multi-Machine Controller on one of the machines to control testing
- Microsoft Data Access Components (MDAC) 2.1 GA

* It is important to note that all users were disseminated across SilkPerformer agent machines as equally as possible in order to balance the driven load. Additionally, no virtual users were executed from the Multi-Machine Controller system

Testing Details – Tuning & Settings

OS and Web server settings were configured according to Allaire KB Article #11772: “Platform-Specific Performance Settings”, and Allaire KB Article #11773: “Performance-Related Resources”.

ColdFusion Administrative Settings were set as follows:

Limit simultaneous requests: 6
Suppress White space by default: OFF
Enforce Strict Attribute Validation: ON
Template Cache Size: 40960
Trusted Cache: ON
Default Client Variable Storage: Cookie
Session Variable Default Timeout: 20 min
Application Variable Default Timeout: 1 day
Automatic Shared Variable Checking and Locking: NONE for All
Data Source Maintain Database Connections: ON for All
Enable Performance Monitoring: OFF
All Other Debugging Settings: OFF

ODBC & OLE DB

The Tack2++ SQL Server 7.0 data source was setup in the ColdFusion Administrator as an ODBC data source and no problems or errors were experienced during testing.

In Allaire KB Article #564: "Using Microsoft Access Databases in a Production Environment", ODBC and OLEDB connections are discussed regarding Access, but also potentially apply to MS SQL Server 7.0:

"Most ColdFusion users use ODBC connections to Access, but they have proven to be the least scalable and robust in internal load testing. Apart from different configuration of the data sources in the ColdFusion Administrator, OLE DB and ODBC connections to Access function programmatically identically. The reliability and robustness of OLE DB connections under load proved to be substantially greater than under ODBC. For this reason, it is advisable to use OLE DB connections to Access data sources instead of ODBC connections whenever possible."

Allaire recommends customers test and evaluate the use of OLE DB as a replacement for ODBC connections for MS Access or MS SQL Server 7.0 data sources.

The KB Article describes the procedure for setting up an OLE DB connection, and can be found at:

<http://www.allaire.com/Handlers/index.cfm?ID=1540&Method=Full>

DNS Settings

A round-robin name was set up on the domain DNS server that would return the IP address of each of the cluster members in a round-robin fashion for all cluster testing. As testing progressed from 2 to 4 to 8-server clustering, the additional server name entries were added to the round-robin DNS server using Microsoft DNS Manager.

For the single server machine tests, the direct machine name was used in the load test scripts rather than the round-robin name used for the multi-server cluster testing.

ClusterCATS, Cluster and per-Server Settings

ClusterCATS was installed with ColdFusion 4.5.1 Enterprise Edition by selecting the "Load Balancing" option during installation. For the multi-server machine tests, a logical ClusterCATS cluster was set up using the ClusterCATS explorer to include the appropriate number of machines for the 2, 4 and 8-server machines tests.

Since no per-client session state information for the Tack2++ application is maintained in individual server memory, the cluster option, "Enable Session-Aware Load Balancing" was not enabled.

The ClusterCATS option "passive mode" was also set for each server in the cluster.

Testing Details – Tack2++ Load Test Script Actions

The Tack2++ load test script is comprised of the following actions:

User1:

ColdFusion 4.5.1 Performance Fact Sheet and Sizing Guide

01_U1_Home - In a browser, enter the Tack2++ home page URL <http://machinename/tack2++/index.cfm>
02_U1_Enter - Click the logo to enter the Tack2++ site and view the main product categories.

User2:

03_U2_Home - In a browser, enter the Tack2++ home page URL <http://machinename/tack2++/index.cfm>
04_U2_Enter - Click the logo to enter the Tack2++ site and view the main product categories.
05_U2_BrSails - Click on the Sails Browse button.
06_U2_AeroBolt - Click on the AeroBolt II link.
07_U2_CatList - Click the Return to Category List link.
08_U2_Masts - Click the Masts Browse button.

User3:

09_U3_Home - In a browser, enter the Tack2++ home page URL <http://machinename/tack2++/index.cfm>
10_U3_Enter - Click the logo to enter the Tack2++ site and view the main product categories.
11_U3_Boards - Click on the Boards Browse button.
12_U3_RocketBoard - Click on the RocketBoard link.
13_U3_Return - Click on the Return to Current Category link.
14_U3_Mondo - Click on the MondoSlab link.
15_U3_ReturnCatList - Click on the Return to Category List link.
16_U3_Sails - Click on the Sails Browse button.
17_U3_Bjorn - Click on the Bjorn link.
18_U3_AddToCart - Click on the Add to Cart button.
19_U3_NavBoards - Click on the Boards link from the floating navigation menu.
20_U3_Rocket - Click on the RocketBoard link.
21_U3_AddToCart - Click on the Add to Cart button.
22_U3_Search - Enter ballistics in the floating navigation search box and click the Go button.

User4:

23_U4_Home - In a browser, enter the Tack2++ home page URL <http://machinename/tack2++/index.cfm>
24_U4_Enter - Click the logo to enter the Tack2++ site and view the main product categories.
25_U4_Boards - Click on the Boards Browse button.
26_U4_NavSails - Click on the Sails link from the floating navigation bar.
27_U4_CAAD4 - Click on the CAAD4 link.
28_U4_AddToCart - Click on the Add to Cart button.
29_U4_ReturnCatList - Click on the Return to Category List link.
30_U4_Masts - Click on the Masts Browse button.
31_U4_LightningRod - Click on the LightningRod link.
32_U4_AddToCart - Click on the Add to Cart button.
33_U4_ReturnToLastCat - Click on the Return to Last Category link.
34_U4_RipperStik - Click on the RipperStik link.
35_U4_Search - Enter kaboom in the floating navigation search box and click the Go button.
36_U4_CartDetails - Click on the Details link from the floating navigation menu.
37_U4_Checkout - Click on the Checkout button.
38_U4_NewCustomer - Click on the New Customer button.
39_U4_SubmitInfo - Complete the form with the following information and click the Submit button:

FirstName	"First"
LastName	"Last"
CompanyName	"Company"
Address1	"Address1"
Address2	" "
City	"City"

ColdFusion 4.5.1 Performance Fact Sheet and Sizing Guide

Province	"MA"
PostCode	"02140"
Phone	"617-761-2000"
Fax	" "
Email	"lastname@foo.com"
Username	"lastname"
Password	"foo"
PswdConfirm	"foo"
NewCustomer	"xxxx"

40_U4_ConfirmPayment - Select "Visa" from the "Card Type" drop-down. Select "2002" from the year drop-down. Enter "4111111111111111" in the "Card Number" field and click the "Confirm Payment" button.

Testing Details – Tack2++ Application Test Modifications and Availability

Since the Tack2++ sample application shipped with ColdFusion serves a strictly educational role, and is not representative of the performance characteristics of a deployed ColdFusion application, several modifications were made for testing. The modifications fell into three categories:

1. Page caching was implemented where appropriate
2. Query result-set caching was implemented where appropriate
3. Modifications to use domain-level cookies to support
4. The application.cfm was removed, since no application-level settings, client or session management were necessary

The complete, modified Tack2++ application and MS SQL Server 7.0 DB object creation scripts can be downloaded from:

http://download.allaire.com/Tack2_App.zip

The specific modifications were:

1. In index.cfm, page caching was enabled using CFCACHE. The following line of code was added as the first line of code to this template:

```
<CFCACHE ACTION="CACHE" TIMEOUT="#DateAdd("h", "-24", Now())#">
```

2. In _showcart.cfm, the following line of code was changed to implement query result-set caching:

```
<CFQUERY DATASOURCE="ServerPerform" NAME="GetCategories"  
cachedwithin="#CreateTimeSpan(1,0,0,0)#">
```

3. In killcart.cfm, the following line of code was changed to implement domain-level cookies:

```
<CFCOOKIE NAME="CartID" EXPIRES="NOW" DOMAIN="#mydomainname#">
```

4. In results.cfm, the following line of code was changed to implement query result-set caching:

```
<CFQUERY DATASOURCE=#datasource# NAME="GetItems"  
cachedwithin="#CreateTimeSpan(1,0,0,0)#">
```

5. In showcategories.cfm, the following code was changed to implement page-level caching, query result-set caching, and use domain-level cookies:

```
<!--- clear last category cookie --->
<CFIF IsDefined("Cookie.LastCategory")>
  <cfcookie name="LastCategory" expires="NOW"
  DOMAIN="#mydomainname#">
</cfif>

<CFCACHE ACTION="CACHE" TIMEOUT="#DateAdd("h", "-24", Now() )#">

<!--- This CFQUERY retrieves the categories from the database --->
<cfquery name="GetCategories" datasource="#datasource#" dbtype="ODBC"
  cachedwithin="#CreateTimeSpan(1,0,0,0)#">
  SELECT CategoryID, CategoryName, CategoryDesc FROM StoreCategories
</cfquery>
```

6. In showitems.cfm, the following code was changed to implement page-level caching, query result-set caching, and use domain-level cookies:

```
<!--- Execute if not called by the CFCACHE tag --->
<CFIF NOT IsDefined("URL.CFNoCache")>
  <CFINCLUDE TEMPLATE="_showcart.cfm">
  <!--- Set this for later --->
  <CFCOOKIE NAME="LastCategory" VALUE=#CategoryID#
  DOMAIN="#mydomainname#">

  <!--- If no category is currently selected, go to the category
  selection page --->
  <CFIF NOT IsDefined("CategoryID")>
    <CFLOCATION URL="showcategories.cfm">
  </CFIF>
</CFIF>

<!--- Re-run and fully regenerate this template at least once every 24
hours --->
<CFCACHE ACTION="CACHE" TIMEOUT="#DateAdd("h", "-24", Now() )#">

<!--- This CFQUERY retrieves the name of the current category --->
<CFQUERY DATASOURCE=#datasource# NAME="GetCurrentCategory"
  cachedwithin="#CreateTimeSpan(1,0,0,0)#">
  SELECT CategoryName FROM StoreCategories
  WHERE CategoryID = #Val(CategoryID)#
</CFQUERY>

<!--- This CFQUERY retrieves the items that belong
to the current category --->
<CFQUERY DATASOURCE=#datasource# NAME="GetItems"
  cachedwithin="#CreateTimeSpan(1,0,0,0)#">
```

The following lines were deleted as they were not used:

```
<!--- Set this for later --->
<CFCOOKIE NAME="LastCategory" VALUE=#CategoryID#>

<CFINCLUDE TEMPLATE="_showcart.cfm">
```

7. In showoneitem.cfm, the following code was changed to implement page-level caching, query result-set caching, and use domain-level cookies:

The following code was deleted:

ColdFusion 4.5.1 Performance Fact Sheet and Sizing Guide

```
<!--- This CFQUERY retrieves the item that will be viewed --->
<CFQUERY DATASOURCE=#datasource# NAME="GetItem" MAXROWS="1">
    SELECT StoreItems.*, StoreManufacturers.Name
    FROM StoreItems, StoreManufacturers
    WHERE StoreItems.ManufacturerID =
StoreManufacturers.ManufacturerID
    AND ItemID = #Val(ItemID)#
</CFQUERY>
```

The following code was added/changed:

```
<!--- Execute if not called by the CFCACHE tag --->
<CFIF NOT IsDefined("URL.CFNoCache")>
    <HTML>
    <HEAD>
    <TITLE>Online Store - Browse Items</TITLE>
    </HEAD>

    <BODY BGCOLOR="#FFCC00" LINK="Maroon"
background="images/storebg.gif" leftmargin=5 topmargin=0>

        <TABLE BORDER="0" CELSPACING="0" CELLPADDING="0">
        <TR>
            <TD WIDTH=130 HEIGHT=15 NOWRAP ALIGN="RIGHT">
                <CFINCLUDE TEMPLATE="_showcart.cfm">
            </CFIF>

<!--- Re-run and fully regenerate this template at least once every 24
hours --->
<CFCACHE ACTION="CACHE" TIMEOUT="#DateAdd("h", "-24", Now() )#">

<!--- This CFQUERY retrieves the item that will be viewed --->
<CFQUERY DATASOURCE=#datasource# NAME="GetItem" MAXROWS="1"
cachedwithin="#CreateTimeSpan(1,0,0,0)#">
    SELECT StoreItems.*, StoreManufacturers.Name
    FROM StoreItems, StoreManufacturers
    WHERE StoreItems.ManufacturerID =
StoreManufacturers.ManufacturerID
    AND ItemID = #Val(ItemID)#
</CFQUERY>
```

