NetworkAppliance®
The evolution of storage.™

Determining and Optimizing Web Application Cacheability with NetCache®

Aaron Kapacinskas, Mike Brown, and Chris Stewart Network Appliance, Inc. |
February 2003 | TR-3237

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Table of Contents

1. Introduction

NetCache appliances offer the ability to control the cacheability of Web-based applications in order to improve the response time and decrease bandwidth usage and server load. Network Appliance has documented the benefits of such cacheability controls for commercial applications such as Oracle11i™, Siebel, and SAP. One example can be seen in the report titled "The Impact of NetCache on Siebel 7." For this report, Accenture was engaged by Network Appliance to research the potential effect of the Network Appliance™ NetCache product on the average business process time for a Siebel 7 Web Client in a distributed user environment. NetCache addresses two specific issues within this user environment—latency for the user and overall network traffic. This study focused on the latency issue and found that in single-user tests a NetCache appliance configured for optimum cacheability reduces the average business process time by up to 45% from the baseline (no NetCache). With a NetCache appliance not configured for optimum cacheability, the average business process time was reduced only 23% from the baseline.

This document gives an overview of how to determine an application's inherent cacheability without any NetCache tuning as well as what cacheability rules should be implemented on NetCache to achieve maximum performance. Methods for measuring the performance gain in terms of bandwidth savings and/or latency reduction that is achieved via cacheability tuning are also discussed. The information presented in this document assumes a level of caching knowledge commensurate with the NetCache Deployment Guide.

2. Equipment Requirements

The following lab environment and equipment should be acquired and configured in order to begin the evaluation. The lab environment should reflect a regional office operating as a remote, satellite location to corporate headquarters. One or more client machines are required, along with a NetCache appliance, two switches (one Layer 4 if transparent redirection is required), a WAN simulator, and the application server. It is easiest to do this outside of a production environment, where there is complete control over the application server and all parameters can be reset between tests.

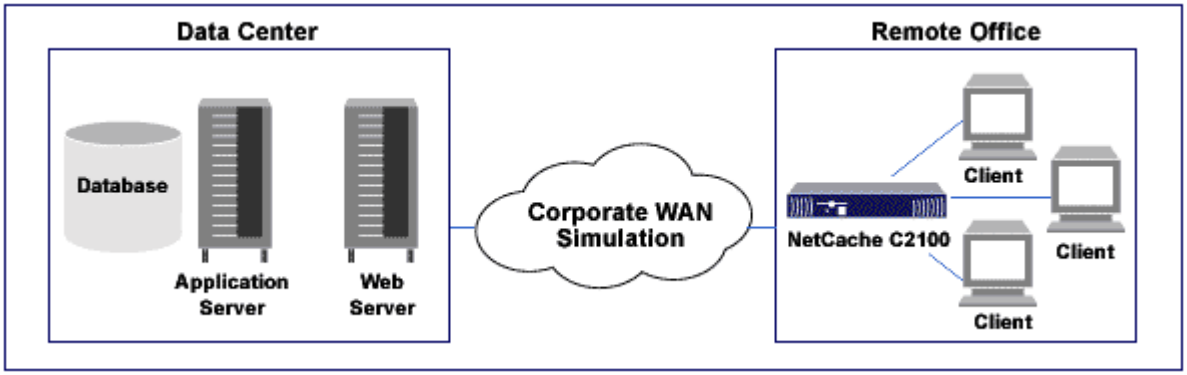

Figure 1) Sample Lab Environment

- User simulation—PCs are used to run the client side of the application through the NetCache appliance and also straight to the application server without NetCache
- WAN simulation—A WAN delay simulator is used to add round-trip delays between the application server, clients, and NetCache; a WAN simulator can be built on a Linux® platform using NistNet, or a Packeteer can be purchased and configured
- Application—A limited-capacity application server environment can be constructed or the existing production application server can be used

3. Testing Overview

Applications can be either "Webified" or "Web-enabled." A Webified application is entirely browser based on the client side, with all objects transported via HTTP and rendered via HTML/XML. These applications have many cacheable components.

A Web-enabled application is often based on Java™. The client in this case is usually several large .jar files that need to be downloaded at run time. The cache is an excellent way to keep those large .jar files close to the end user, dramatically decreasing application start times and saving bandwidth.

In a Web-enabled deployment, consideration needs to be given to any other ports that may be required by the application for communication (for example ports 8002 and 9002 for Oracle11*i*). These ports are application dependent and must be added to the NetCache tunnel list so that communication is uninterrupted.

Proper name resolution is assumed in this document. If the application breaks while going through the cache, host name resolution is usually the cause. This issue should be checked and eliminated before testing begins.

Test scenarios should be run with WAN simulator delays of 0, 50, 100, 200, 300, and 400 milliseconds between the application server and the Web client for the baseline no-cache tests and between the application server and NetCache for the cacheability tests. These delays are representative of round-trip times seen over the range of communication from LAN to transoceanic WAN. There should be no delay between the clients and NetCache, as they are deployed on the same logical and physical network segment.

There are three testing goals:

1) Measure inherent cacheability of the application, without any tuning
2) Determine optimum NetCache cacheability rules to implement
3) Measure performance improvement with cacheability rules implemented

3.1. Measuring Inherent Application Cacheability

The first goal requires running a typical application transaction and does not require any macroing or scripting of the application. The purpose of the test is to run application traffic through the cache in order to get an object profile. Cache hits and misses are examined to see where a custom set of cacheability rules can be implemented that will maximize the amount of traffic that the cache can serve to the client without breaking the application. The transaction needs to be run a minimum of two times with the user redirected through the cache either transparently (via L4 switch or WCCP router) or explicitly (via proxy PAC or browser proxy setting). The first run of the application transaction through the cache is to populate the content. The second run is to evaluate the response based on the now-cached application content. How the cache is handling the objects can be examined in two ways. One is via the detailed NetCache Web logs, and the other is via a packet capture.

The inherent cacheability of an application or Web site is the ratio of cache hits to the total content, i.e., cacheability = hits/total. This can be done for both the object count and the byte count. Dividing byte hits by the total content transferred gives the bandwidth savings. Object hit rates tell the percentage of an application that is cacheable. Object and byte hit-rate percentages can be found in the NetCache GUI in the Data-Web Statistics section.

3.2. Determining Cacheability Rules to Implement

Look first at the NetCache Web access log file to find information needed to determine the types of cacheability rules to implement. The following is an example of typical messages in a NetCache transaction log:

- TCP_HIT/200—This object was in cache and served by NetCache
- TCP_MISS/200—This was the first fetch of a cacheable object
- TCP_MISS_PRIVATE_CCTRL/200—This object is not cacheable: set private by Web server
- TCP_HIT_IMS_NOTMOD/304—This object was served by NetCache: validated with Web server
- TCP_MISS_PRIVATE_COOKIE/200—Not cacheable due to cookie present
- TCP_MISS_PRIVATE_ON_STOP_LIST/200—Not allowed to be cached by NetCache; this goes away during the cache tuning tests

Some examples of rules that could be enabled after evaluating the logs and traces would be:

- Unnecessary IMS requests on infrequently changing images
- Unfounded pragma-no cache directives
- Unnecessary noncached items due to query strings
- Cookie cheats

For example, after noticing that a TCP_MISS/200 was received, a look at the trace for the transaction might indicate that the object has a pragma-no cache header. It may have no adverse effect on the application to create a rule that ignores the no-cache directive for this object. Examination of the rule's impact, by running the application with the rule enabled, will clearly show whether the rule has caused any undesirable effects in the application.

Similarly, IMS requests are often used on objects in applications that are rarely, if ever, updated. A set of .gif files that are buttons on a Webified application screen does not need to be revalidated very often. If the cache is forced to make a check on them for every instance they are used by the application and the WAN delay is noticeable, the impact on application response time can be dramatic. In cases like this, a rule would be created to ignore IMS checks on that certain type of file (and could even get as granular as to ignore the check from certain domains or for certain times, etc.).

Examples and details of other cacheability rules can be found in the NetCache Guide to Caching Protocols and Services (NOW™ Web site login required).

3.3. Measuring Performance Improvement with Cacheability Rules

Once cacheability rules based on log and/or trace examination have been implemented, this test is performed to measure the benefit of the NetCache appliance. Demonstrating the gain can be done in two ways:

1) Monitor switch bandwidth difference

2) Examine an exactly reproducible transaction with and without the cache

Monitoring server-to-cache traffic and cache-to-client traffic on a switch is straightforward. Doing this for the cached and uncached transaction will clearly show the bandwidth savings. This is done by observing the server traffic relative to the total application outgoing traffic from the cache. Comparing the total traffic throughput will demonstrate the byte saving and hit rate of the caching independent of the cache UI statistics.

Exactly reproducing a transaction requires the use of a macroing or scripting tool to automate an exact set of steps and time intervals in a typical transaction. There are many such tools available. For example, Mercury LoadRunner is capable of taking scripted actions, delays, mouseclicks, etc. to exactly duplicate an application process over and over. The benefit of this approach is that transaction runs with and without caching can be objectively compared to demonstrate gains. This approach allows the comparison of "apples to apples."

The scripted procedure also offers a good chance to get some application base-line performance without caching involved.

4. Testing Scope

Testing should be restricted to a set of user interactions/business processes that are representative of an average application Web client. Examples of typical business processes are:

1) Logon, view new additions, create new entry, logout

2) Logon, search for content, create request, create activity, logout

3) Logon, search for multiple accounts, view all open content, view all open requests, logout

5. Test Procedure

The test procedure detailed in Appendix A should be repeated as desired to gather data for averaging. The procedure ensures that the database sizes and application configurations will be exactly matched for each stage of NetCache configuration.

6. Conclusion

NetCache cacheability tuning can often result in a significant improvement in application response time and a decrease in bandwidth usage and application server load. NetApp has documented the benefits of such tuning for commercial applications such as Oracle11*i*, Siebel, and SAP. By following the procedures in this document, users can tune NetCache for optimum delivery of any Web-enabled or Webified application, as well as measure the performance benefit from doing so.

7. Appendix A—Cacheability Testing Procedure

Step Number	NetCache Configuration	Action	Measurements	Purpose
1	N/A	Establish laboratory environment	None	Initial setup
2	Disabled	Execute application transactions	– Bandwidth usage – Response time	Gathers baseline performance data, with no caching
3	No change	Reset application	None	Returns application to initial state

Step Number	NetCache Configuration	Action	Measurements	Purpose
4	Enabled, but no cacheability rules	Execute application transactions	None	– Generates logs to analyze, for determining cacheability rules – Populates NetCache with application content for step 6
5	No change	Reset application	None	Returns application to initial state
6	No change	Execute application transactions	– Bandwidth usage – Response time	Gathers performance data with a populated NetCache appliance, without cacheability rules implemented
7	No change	Reset application	None	Returns application to initial state
8	No change	Flush cache	None	Deletes application content from cache
9	Enabled, with cacheability rules implemented	Determine and implement cacheability rules from logs generated in step 4	None	Preparation for step 10
10	Enabled, with cacheability rules implemented	Execute application transactions	None	– Generates logs to analyze, for determining whether cacheability rules should be updated – Populates NetCache with application content for step 11
11	Enabled, with cacheability rules implemented	Execute application transactions	– Bandwidth usage – Response time	Gathers performance data with a populated NetCache appliance, with cacheability rules implemented
12	No change	Reset application	None	Returns application to initial state

Step Number	NetCache Configuration	Action	Measurements	Purpose
13	No change	Flush cache	None	Deletes application content from cache
14	Enabled, with cacheability rules implemented	Update NetCache with new cacheability rules based on logs in steps 10 and 11	None	Iterative process to optimize cacheability tuning
15	Enabled	Repeat steps 10–14	Various	Repeat until cacheability rules are optimal
16	Various	Repeat steps 2–15	Various	Repeat procedure as desired to gather data for averaging
17	N/A	Document results, compare measurements in steps 2, 6, and 11	N/A	Comparison shows the benefit of cacheability rules

The measurements in steps 2, 6, and 11 allow comparison between the native application performance (without caching), the performance and inherent cacheability of the application without NetCache tuning, and the performance improvement achievable from cacheability tuning.

NetworkAppliance®
The evolution of storage.™

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com