Performance Tuning I

by Theresa Ray of Tensor Information Systems, Inc.

Sponsored by Apple Computer, Inc.
Apple Developer Connection

Performance Tuning I

by Theresa Ray of Tensor Information Systems, Inc.

Performance tuning your WebObjects application is a critical step on the road to successful deployment of a web site. By properly coding, testing and administrating your app, your web site can be a great success! This document is part one of a two part series.

The importance of performance tuning your application

We've all been irritated by web sites with agonizingly slow response times, or which contain unnecessarily large images that take an eternity to download. You do NOT want your web site to be difficult or frustrating for its users. Proper performance tuning of your WebObjects application can ensure that your users will have a responsive and enjoyable visit to your site

Thorough optimization of a WebObjects application requires many steps. Most apps don't need to be tuned in every possible area. Occasionally, a very large-scale application will require complete top-to-bottom inspection and optimization. This document, and is successor (Performance Tuning II) strive to identify most of the key areas in which a WebObjects application COULD be tuned. You will have to decide which areas of optimization will be most meaningful for your deployment, and implement those areas. Optimization of a web site is never destructive, only costs a moderate amount of development time, and results in a solid, scalable system.

This document, Performance Tuning I, focuses on proper development techniques which result in a well-tuned application. The next document in this series, Performance Tuning II, focuses on additional testing and deployment techniques that achieve further optimization.

Optimizing your WebObjects code

There are many areas in the development of your WebObjects code where choosing the proper configuration can result in an optimally performing web site. By being aware of all the options available and their effect on your application, you can strike a balance between the requirements of your application and optimal performance of your web site.

Session management

Session management (or state storage) is one area where the developer can strike an appropriate balance between the application's requirements and optimal performance of the application. A good tuning technique is to store the minimum amount of information required by your application in the session. In an application that will have a large number of simultaneous active sessions, restricting the information stored in the session can make a large difference in the amount of memory used by your application, and thereby increase its performance.

Proper choice of state information storage can also have a significant effect on an application. By default, WebObjects stores its session information in a memory-resident object in the application. This restricts load balancing options (more on load balancing in Performance Tuning II, and in the online documentation under the title *Serving WebObjects*), since it requires a user of your application to return to the same instance of the WebObjects application as he or she navigates through your site. For example, if you have ten instances of your

WebObjects application running, and Mary connects with instance number four on her initial visit to the site, she must return to instance number four as she navigates throughout the site. She cannot move to another instance of the application until she starts a new session.

In contrast, choosing to store the session information on the file system, in the database, in hidden fields within the HTML or in a cookie, allows the user to connect to any instance of the WebObjects application as he or she navigates through your site. This provides for a truly load balanced system. Of course, several of these options have their own drawbacks. Storing session information in hidden fields in HTML is not optimal if you have a moderate or large amount of session information, since that information is transferred from client to server with each request. Cookie storage of session information is not a good choice if you wish to allow users who disable cookies to use your site. Storage of session information in a database may not be an optimal solution if your application already does a large amount of database storage and retrieval. And if your file system is on the verge of being overloaded and overly accessed, storage of state on the file system may not provide a tuned solution either. You have to analyze your application's architecture and your deployment configuration and requirements to determine which mechanism will result in the best possible performance for your application.

Remembering to set a timeout for your session is a critical part of optimizing your application. With no session timeout set, every user session that is begun is maintained until the application is terminated. This will cause your application to grow unabated, and result in unacceptable performance. Setting the session timeout by sending the message `this.setTimeOut(3600)` in the session's init method results in the session being deallocated (along with all of the memory it required) after 10 minutes (the argument for this method is in seconds). You need to choose the appropriate session time-out for your site.

Development vs. Deployment Code Techniques

Code written in WebScript is so convenient during development of your application! After making changes to your logic, there is no need to recompile. All you are required to do is reload the page, and voila! Your new code changes have been implemented and are reflected immediately! Well, what's optimal for development is rarely optimal for deployment. WebScript is interpreted real-time by WebObjects when creating a response or processing a request. As with any interpreted language, scripts run significantly slower than languages that are compiled, such as Java and Objective-C. Therefore, for optimum performance of your web site, you should convert all components which are not extremely minimal in nature from WebScript (or Perl, if you choose to use the third-party WOPerl tool) into Java or Objective-C before deployment.

Additionally, WebObjects by default does not cache its component definitions. Component definitions consist of the .html and .wod files which identify how to render a page within your web site. Caching of these components is off by default so that during development and testing, you are not required to restart the application in order to see changes that have been made to these files. For most applications, enabling caching of component definitions before deployment is a highly recommended procedure. To enable component caching in your application, you can invoke the `this.setCachingEnabled(true)` message in the application's init method, or by specifing the -c argument on the command line when launching your WebObjects application.

There are a few unusual applications which have very generic components and choose from a set of .html and .wod files in real-time by overriding the templateWithName method. Enabling component caching for these types of applications will result in unpredictable behavior, as it disables the mechanism implemented in the templateWithName: method.

During development of your applicaton, you likely used many logWithFormat: statements, or had tracing options enabled, in order to gain visibility into your application's behavior without having to invoke the debugger

continually. You will want to remove as many unnecessary log statements as you can and turn off all tracing options before deployment, so that your application is not wasting time or file system resources by logging information that is no longer pertinent. Some logging, of course, is prudent. Most deployment applications have implemented an exception handler to prevent users of the site from seeing the lengthy WebObjects debugging error message when an exception is thrown. Logging the reason for the exception and meaningful variable values in your exception handler will provide insight as to why the users are experiencing difficulty, and will ease troubleshooting of your production application.

Memory Leaks

If your application is written entirely in Java or WebScript, this section does not pertain to you, since WebObjects manages memory allocation and deallocation for these components. If, however, you have developed any of your application in Objective-C, you will want to thoroughly analyze your memory allocation and deallocation, as you are responsible for releasing any memory that you retained. Remember that alloc, retain and copy are all messages that increment the retain count for an object, and which require a release or autorelease message to be sent in order to decrement the retain count and allow the object to be freed.

Forgetting to release the memory associated with an object is termed a "memory leak", and can cause your application's memory requirements to grow over time (how fast depends on the size of the leak), and will result in poor performance. WebObjects provides a tool for NT called ObjectAlloc (found in the Developer/Applications area after installation), which allows you to observe object allocation/deallocation activity in an OpenStep application (including a WebObjects application) in real time.

Unfortunately, there are no tools currently available for UNIX platforms. However, by monitoring the amount of memory used by your application in conjunction with the number of active sessions, you can get a good indication whether or not you have a memory leak, and how bad that leak is. Finding it requires you to either use ObjectAlloc on NT, or manually review your code.

Optimizing HTML

Optimization of HTML is another way to improve the performance of your application. Following good HTML programming practices, such as minimizing the number and size of images within a page, provides a good start on tuning your application. There are, however, some additional areas of analysis necessary for WebObjects programs. Any reusable components that exist in your application must also be considered when determining the size of the page you are returning to your users. You may have minimized all the images on your parent components, but neglected to optimize your navigation bar images, for example.

When using WORepetitions in your page, you should take a moment to analyze the information contained within the repetition. Is the WORepetition creating rows of a table for the results of a fetch from the database? If so, you should implement a fetch size limit, or show the results in batches (20 per page, for example), so that you don't end up trying to return 1000 rows in a table that must be rendered by the user's browser.

In general, non-optimal HTML contained within a WORepetition can also provide some unwieldy results. For those of you who do not use WebObjects Builder to create your HTML, some web design packages produce notoriously redundant HTML. For instance, instead of setting up the font for the document as a whole, some apps will stick font tags in with every cell for a table. When rows of a table are contained within a WORepetition, the result can be an unnecessarily large HTML file. Even if you use WebObjects Builder to create HTML, it is never a bad idea to take a quick look at the HTML being generated to see if it can be optimized.

Optimizing Database Access

This is a topic worthy of an entire survival guide, and a class on database performance tuning. There are so many details associated with tuning the database access of your application, that they cannot be covered here. However, this guide will address the major topics involved in optimization of database access. For more information, consult additional documentation available for your database, and the *Enterprise Objects Framework Developer's Guide*.

Your application will typically require information common for all users, such as a list of states for a name/address form. You may want to prefetch the tables that are used by the application purely for look-up requirements. This will increase the memory required by your application somewhat, but look-up tables are usually fairly small in size, and the time saved by not having to request the information from the database for every user session can make it a worthwhile investment. Each user session would refer to the application for the common information.

Prefetching can also be useful when a session initializes. There may be information which is not generic enough to prefetch in the application, but which you can identify at the beginning of a session. If it is not a huge amount of information, you may want to consider gathering this information upon the initial request of the user (for instance, information about a previous visit to the site). The response time for the first page of the site will be somewhat slower because of the additional database access, but subsequent requests by that user will be processed faster. Again, it depends on the requirements of your application.

Optimization of your data model can result in the most significant response time improvement for your application. Particularly for large, complex data models, denormalization can provide very optimal performance. In a normalized data model, all the information is cleanly segmented into relevant tables, using relationships (foreign keys) to join tables containing related information. However, if the information you need is spread across several different tables in the database, the application must perform a fetch for each table in the database from which it needs information.

For example, suppose that you have a data model that has a Customer table, a NameAddress table, a Household table, and a UserProfile table. Unfortunately, some frequently accessed page of your application requires information from all of these tables to display the appropriate interface for the user. The application will have to fetch from each of these tables to gather the information for the display.

In a denormalized data model, all of the information needed to display a single page would optimally be located in a single table in the database. However, this makes the table large, complex, and confusing. Of course, finding the appropriate balance between normalized and denornalized is what you want to do. If you find that you are routinely accessing tables for only one or two attributes, you may want to consider replicating or moving those attributes into another more frequently accessed table.

Ensuring that your database has indexes created for attributes that you commonly use in qualifiers for your fetches is also critical to the performance of your application. In a Customer table, which may contain hundreds of thousands of records, missing one key index on an attribute can result in a query which takes ten minutes or more to execute. Indexes take up space in the database, so you do not want to blindly index every attribute either. You need to analyze the qualifiers that your application routinely uses, and have your DBA ensure that those queries are optimally tuned.

If you find yourself with queries that take a long time to execute, even after optimizing your data model, you may want to investigate using additional EOF options, such as supplying fetch hints in your SQL, or canceling a fetch after a certain number of rows is returned.

Conclusion

Correctly designing and implementing your WebObjects application results in a solid foundation for a responsive web site. Use of the techniques described in this guide will assist you in ensuring that your application is tuned and streamlined. In the next guide, Performance Tuning II, techniques for testing the success of your application design and configuring deployment options will be discussed.

Phone: 817-335-7770
E-mail: theresa@tensor.com
URL: http://www.tensor.com

Resources…

http://gemma.apple.com/techinfo/techdocs/enterprise/WebObjects
WebObjects Developer's Guide
Enterprise Objects Framework Developer's Guide
http://www.omnigroup.com/MailArchive/WebObjects
http://www.omnigroup.com/MailArchive/eof
http://www2.stepwise.com/cgi-bin/WebObjects/Stepwise/Sites
ftp://dev.apple.com/devworld/Interactive_Media_Resources
http://www.apple.com/developer
http://developer.apple.com/media
http://til.info.apple.com/

About the Author…

Theresa Ray is a Senior Software Consultant for Tensor Information Systems in Fort Worth, TX (*http://www.tensor.com)*. She has programmed in OPENSTEP (both WebObjects and AppKit interfaces) on projects for a wide variety of clients including the U.S. Navy, the United States Postal Service, America Online, and Lockheed-Martin. Her experience spans all versions of WebObjects from 1.0 to 4.0, EOF 1.1 to 3.0, NEXTSTEP 3.1 to OPENSTEP 4.2, Rhapsody for Power Macintosh, and yellow-box for NT. In addition, she is an Apple-certified instructor for WebObjects courses.

Tensor Information Systems is an Apple partner providing systems integration and enterprise solutions to its customers. Tensor's employees are experienced in all Apple technologies including OPENSTEP, NEXTSTEP, Rhapsody, EOF and WebObjects. Tensor also provides Apple-certified training in WebObjects, Oracle consulting and training, as well as systems integration consulting on HP-UX. And Oracle.

You may reach Theresa by e-mail: *theresa@tensor.com* or by phone at (817) 335-7770.

Interactive Media Resources Include:

- **Interactive Media Guidebooks**
- **Market Research Reports**
- **Survival Guides—Technical "How To" Guides**
- **Comarketing Opportunities**
- **Special Discounts**

Apple Developer Connection

As Apple technologies such as QuickTime, ColorSync, and AppleScript continue to expand Macintosh as the tool of choice for content creators and interactive media authors, the Apple Developer Connection continues its commitment to provide creative professionals with the latest technical and marketing information and tools.

Interactive Media Resources

Whether looking for technical guides from industry experts or for market and industry research reports to help make critical business decisions, you'll find them on the Interactive Media Resources page.

Apple Developer Connection Programs and Products

ADC programs and products offer easy access to technical and business resources for anyone interested in developing for Apple platforms worldwide. Apple offers three levels of program participation serving developer needs .

Membership Programs

Online Program—Developers gain access to the latest technical documentation for Apple technologies as well as business resources and information through the Apple Developer Connection web site.

Select Program—Offers developers the convenience of technical and business information and resources on monthly CDs, provides access to prerelease software, and bundles two technical support incidents.

Premier Program—Meets the needs of developers who desire the most complete suite of products and services from Apple, including eight technical support incidents and discounts on Apple hardware.

Standalone Products

Apple offers many standalone products that allow developers to choose their own level of support from Apple or enhance their Select or Premier Program membership. Choose from the following products and begin enjoying the benefits today.

Developer Connection Mailing—Subscribe to the Apple Developer Connection Mailing for the latest in development tools, system software, and more.

Technical Support—Purchase technical support and work directly with Apple's Worldwide Developer Technical Support engineers.

Apple Developer Connection News—Stay connected to Apple and developer-specific news by subscribing to our free weekly e-mail newsletter, *Apple Developer Connection News*. Each newsletter contains up-to-date information on topics such as Mac OS, Interactive Media, Hardware, Apple News and Comarketing Opportunities.

Macintosh Products Guide

The most complete guide for Macintosh products! Be sure to list your hardware and software products in our *free* online database!

http://developer.apple.com/media

The ultimate source for creative professionals.