

Predictive tuning for client/server applications

Introduction

In today's fast-paced IT environment, applications must be built and deployed quickly and efficiently—often with little attention paid to performance and no time for pre-pilot performance testing. Predictive tuning allows developers to assess the effects of server processing, network bandwidth or network latency on applications and presents options for addressing performance problems before deployment and costly infrastructure expenditures.

This document describes how Compuware Application Expert helps developers and performance engineers build applications that meet end-user response time requirements.

Definitions

Performance problem

A performance problem is defined as an end-user business function or transaction, which does not complete within the time frame required by the business. Response time delays can occur at any point within the system—in one of the servers, the client machine or across the network—and can be caused by the element itself or by the behavior of the application in relation to that element. These delays can be associated with one or more application threads.

Application thread

For purposes of distributed application performance analysis, the term “application thread” describes an entire sequence of frames on the network that constitutes a single, complete action by an application task. The thread includes the command and response data. Examples of application threads include:

- SELECT, INSERT, UPDATE and EXECUTE statements in a database environment
- HTTP GET or PUT requests for web client interfaces
- OPEN, READ and WRITE functions in a file server environment.

An integrated system

Performance testing is usually performed with developers focusing on individual coding modules; DBAs focusing on the database, server size and power; and network administrators focusing on the network. To tune an application for its intended deployment environment, the application must be viewed as part of an integrated system, including the clients, server(s) and the network (a typical integrated network is shown in Figure 1). While this can be accomplished by piloting the application in production, other testing methods can deliver insight into performance earlier and provide the opportunity for product performance enhancements. If the application must scale to support hundreds or thousands of clients, for example, load testing provides server capacity planning data.

Although this is useful information for sizing the server, it does not take into account the network where it will be running. Compuware’s predictive tuning methodology includes an analysis of the performance impact the production network will have on the application, and identifies specific functions as tuning opportunities, based on their potential to deliver response time improvements.

Predictive tuning

What is predictive tuning?

Predictive tuning is the process of tuning a system for its intended deployment environment, rather than for its measured environment. Production environment components are created to test for performance constraints in a system. This typically involves deploying the application in a pilot, long after application changes can be implemented easily. Hence, these tests serve to define the system’s limitations rather than permit its expansion. Predictive tuning is based on a lab-friendly testing solution that analyzes integrated system performance, in particular the performance relationships between system architecture, application logic, network characteristics and server processing speed of software.

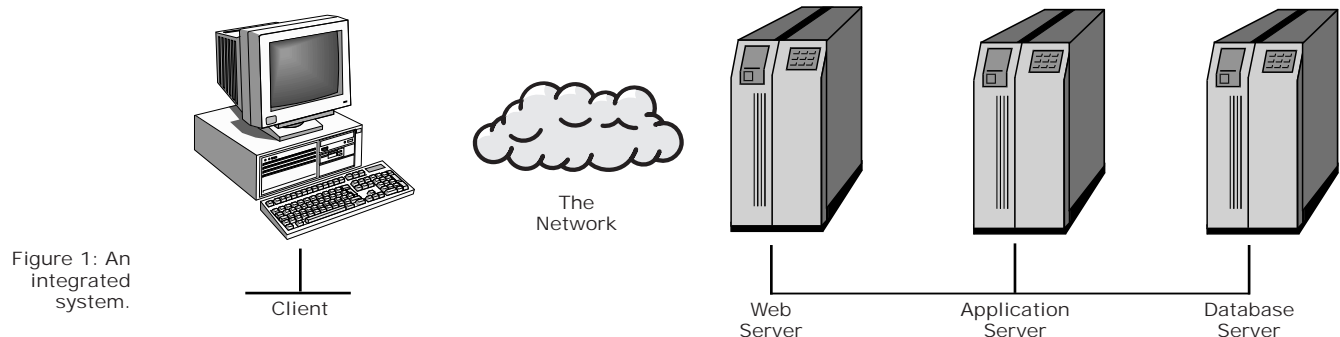


Figure 1: An integrated system.

Predictive tuning classifies the transaction as bound by server (processing), network bandwidth or network latency. It then quantifies the sensitivity of each application thread to the predicted performance constraint. The result is a clear understanding of the performance impact contributed by each component of the integrated system that provides organizations the opportunity to correct problems before deployment, defining actionable and measurable performance improvement solutions.

The elements of predictive tuning

Effective predictive tuning requires some basic information about the application to be analyzed.

The test environment

The test environment needs to include a client machine, the server(s) supporting the application, a local network connecting these together and the distributed application. Developers must be able to execute the transaction across a local network from a client machine. Application Expert captures the transaction's data packets from the network to perform its analysis.

The business performance goal

To determine how to tune the system, it's helpful to know the response time end-users expect. This helps identify what efforts are needed to meet these goals. Alternatively, predictive tuning can evaluate the system's performance efficiency, essentially as a means to validate acceptable production behavior.

The targeted deployment environment

Predictive tuning requires a simple definition of the intended deployment environment, understanding where the server(s) will be located, and basic abstractions of the network paths to connect users to these servers. If the network does not yet exist, or server locations have not yet been defined, predictive tuning can determine appropriate network infrastructure requirements and ideal server locations.

The process of predictive tuning

Application Expert's Capture utility provides a simple means of capturing transaction traces from a test network. A separate capture of each transaction should be reviewed for tuning.

Application Expert uses common values of bandwidth (network speed) and latency (network delay) definitions as inputs to its Response Time Prediction algorithms. The prediction output quantifies each transaction's performance in terms of sensitivity to the core elements of the integrated system—server processing, client processing, network bandwidth and network latency.

Application Expert's Thread Analysis is used to prioritize the application threads most sensitive to this environment for review. Attention is given first to tuning those threads that will consume the greatest amounts of the response time "budget"—the end-user performance goal.

Predictive tuning: An example

The test environment

To illustrate the value of predictive tuning, it's helpful to review a real-world example. In this scenario, Application Expert captures the communication between two Oracle servers in a development test lab. The application updates product information between two databases, and is intended to run between New York and Brussels. A test network is shown in Figure 2.

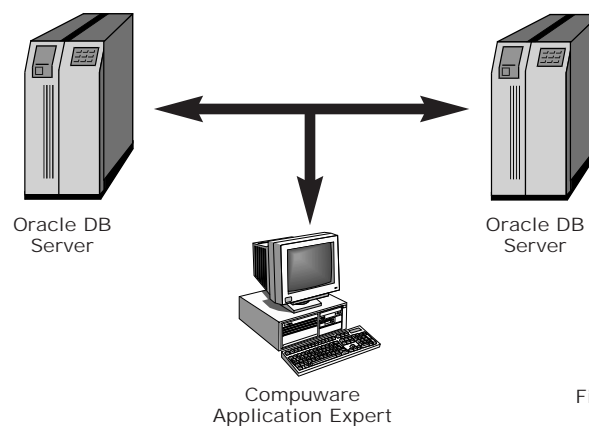


Figure 2: The test environment.

Performance goals

The expected performance goal is to complete the transaction within 10 minutes.

The targeted deployment environment

The transaction will run between two data centers, one in New York and the other in Brussels. The current link between New York and Brussels is a 256 Kbps PVC, operating at about 10 percent utilization. "Ping" measurements determine that the approximate network latency is about 210 milliseconds.

The predictive tuning process

Once the capture is complete, Application Expert profiles the performance of the transaction. This can be viewed from multiple perspectives. The Response Time Analysis view (shown in Figure 3) allocates the overall transaction response time to processing time in each server as well as network transmission time. It should be noted that the task duration is 204 seconds.

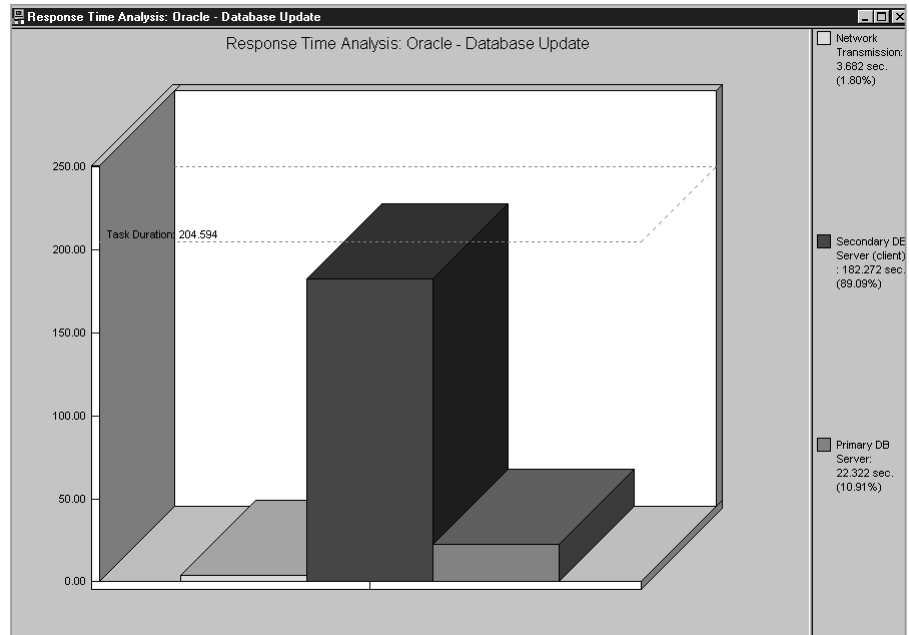


Figure 3: Response Time Analysis.

The Response Time Analysis chart shows how much time is allocated to each component in the transaction. That is, how much time is spent in transmitting the data, and how much time is spent processing by the primary and secondary servers. As illustrated in Figure 3, of the total transaction response time of 204 seconds, 182 seconds have been allocated to processing in the primary server, while 3.7 seconds are spent on the local area network.

The Thread Analysis view, as shown in Figure 4, lists the application threads that cross the network, similar to the Oracle trace utility. The performance of each thread is also detailed, and this information is used later to perform predictive tuning.

	Name	Start Time	Duration	Bytes	App. Turns	Server Time	Client Time	Seconds
4	ORACLE:"SELECT * FROM "CO	0.01793	0.25004	4385	13	0.12333	0.12336	100
5	ORACLE:"SELECT * FROM "CO	0.26868	0.00570	1021	3	0.00391	0.00113	
6	ORACLE:"SELECT * FROM "CO	0.27924	0.00462	663	3	0.00311	0.00103	
7	ORACLE:"SELECT ROWID."CO	0.28522	0.44549	4310	10	0.26223	0.18003	
8	ORACLE:"ALTER SESSION SE	0.73187	0.13732	1906	5	0.09456	0.04135	
9	ORACLE:"SELECT * FROM "CO	0.87626	0.18211	1823	6	0.01492	0.16584	
10	ORACLE:"SELECT * FROM "CO	1.05910	0.00558	1126	3	0.00369	0.00117	
11	ORACLE:"SELECT * FROM "CO	1.06904	0.00455	662	3	0.00309	0.00099	
12	ORACLE:"SELECT ROWID."LI	1.07502	3.85489	77428	116	2.67191	1.12207	
13	ORACLE:"ALTER SESSION SE	4.93098	0.07962	1953	5	0.04640	0.03181	
14	ORACLE:"SELECT * FROM "DI	5.01788	0.30093	1859	6	0.01267	0.28689	
15	ORACLE:"SELECT * FROM "DI	5.31952	0.00585	1163	3	0.00365	0.00145	
16	ORACLE:"SELECT * FROM "DI	5.33002	0.00452	658	3	0.00304	0.00100	
17	ORACLE:"SELECT ROWID."DI	5.33610	119.37592	592065	652	110.11199	8.81900	
18	ORACLE:"ALTER SESSION SE	124.71312	0.10312	1947	5	0.07141	0.03031	
19	ORACLE:"SELECT * FROM "DI	124.82449	0.15691	1985	6	0.01021	0.14518	
20	ORACLE:"SELECT * FROM "DI	124.98210	0.01009	1408	3	0.00387	0.00514	
21	ORACLE:"SELECT * FROM "DI	124.99349	0.00486	656	3	0.00327	0.00112	
22	ORACLE:"SELECT ROWID."DI	124.99996	0.93720	6574	12	0.83667	0.09568	
23	ORACLE:"ALTER SESSION SE	125.93826	0.09829	1882	5	0.06634	0.03055	
24	ORACLE:"SELECT * FROM "DI	126.04349	0.14735	1767	6	0.01474	0.13126	
25	ORACLE:"SELECT * FROM "DI	126.19155	0.00606	1130	3	0.00415	0.00120	
26	ORACLE:"SELECT * FROM "DI	126.20196	0.00555	654	3	0.00409	0.00099	
27	ORACLE:"SELECT ROWID."DI	126.20894	0.31838	17672	45	0.14241	0.16195	
28	ORACLE:"ALTER SESSION SE	126.52854	0.08067	1906	5	0.04863	0.03063	
29	ORACLE:"SELECT * FROM "DI	126.61554	0.13837	1527	6	0.01364	0.12357	
30	ORACLE:"SELECT * FROM "DI	126.75465	0.00524	891	3	0.00355	0.00112	
31	ORACLE:"SELECT * FROM "DI	126.76376	0.00463	662	3	0.00314	0.00101	

Figure 4: Thread Analysis.

The Gantt chart, to the right of the Thread Analysis, depicts the duration of each thread. Note that there is one select statement that takes about two minutes to complete, and the remaining statements execute relatively quickly.

The performance goal (end-user response time requirement) is for this transaction to complete in 10 minutes, and developers can use the Response Time Predictor to determine how this transaction will perform on the production network.

In Figure 5, the top bar in Application Expert's Response Time Predictor represents a measured, or baseline, response time captured in the lab. The second bar predicts the response time if the application is deployed in the existing network; the prediction of 53 minutes does not meet the response time requirement. What if the network bandwidth was increased? The third bar, defined with a bandwidth increase to T1, illustrates the negligible improvement.

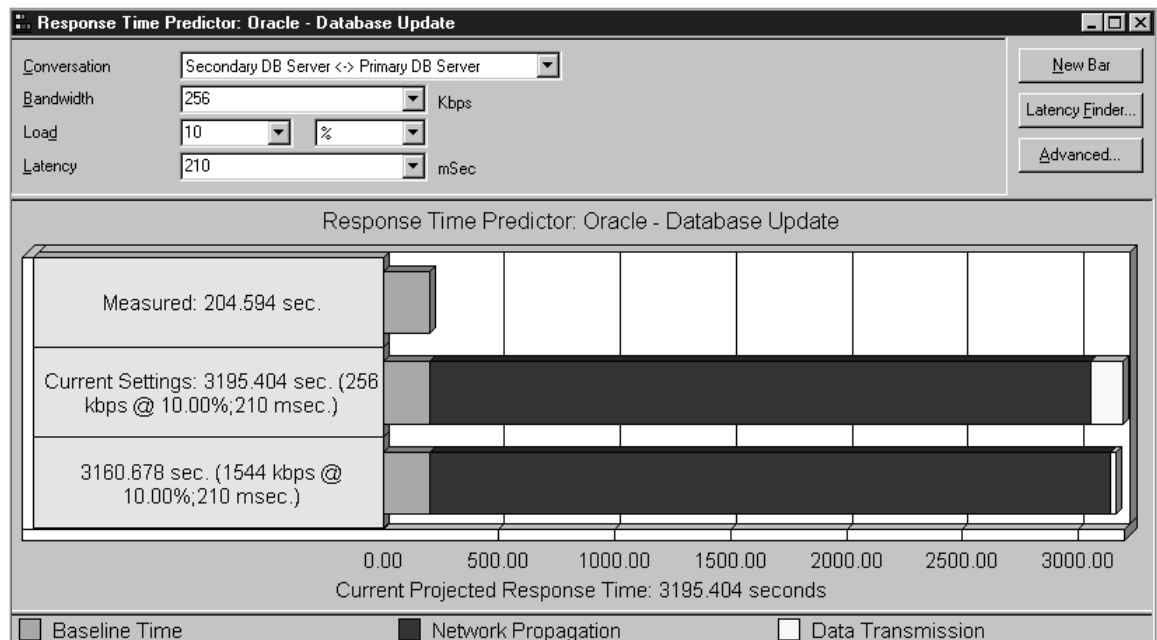


Figure 5: Response Time Predictor.

The Response Time Predictor illustrates clearly the transaction will not meet the end-user business requirement, and adding network bandwidth will not significantly improve performance. The dominant portion of each prediction bar is blue, indicating extreme sensitivity to network latency. The Thread Analysis is now reviewed, with an eye towards those threads that are latency-bound, i.e., are sensitive to the latency in the intended deployment environment. Latency sensitivity is driven by the number of application turns, or the request/response handshaking incurred by the thread. In Figure 6, the Thread Analysis tabular view is sorted in descending order by the number of application turns.

Thread Analysis: Oracle - Database Update											
	Name	Start Time	Duration	Bytes	App. Turns	Bytes/App. Turn	Server Time	Client Time	Server Rate (bps)	Client Avg. Burst (Bytes)	Server Avg. Burst (Bytes)
1	All threads	0.00000	204.59415	4602099	13672	336.6	155.97454	226.38234	135712.2	82.0	253.0
2	ORACLE:"SELECT ROWID,"EN	185.41959	19.17456	1787422	6881	259.8	8.69032	9.05761	529445.7	75.0	184.0
3	ORACLE:"SELECT ROWID,"DQ	177.87128	6.92430	661314	2509	263.6	3.24741	3.14918	545337.7	75.0	188.0
4	ORACLE:"SELECT ROWID,"TI	130.43288	18.47843	851981	1771	481.0	3.56073	14.24051	309158.7	77.0	403.0
5	ORACLE:"SELECT ROWID,"TH	149.71196	3.08186	280337	1020	274.8	1.33803	1.52009	526973.0	75.0	199.0
6	ORACLE:"SELECT ROWID,"DI	5.33568	119.26274	590653	647	912.9	110.06370	8.75522	33932.2	131.0	781.0
7	ORACLE:"SELECT ROWID,"UJ	1.07460	3.85411	76012	111	684.8	2.71414	1.08012	137293.4	88.0	595.0
8	ORACLE:"SELECT ROWID,"ND	153.95430	19.44279	109136	105	1039.4	18.30390	1.05629	39112.5	134.0	905.0
9	ORACLE:"SELECT ROWID,"TH	173.83439	3.07353	44619	64	697.2	2.06325	0.97541	100715.3	92.0	604.0
10	ORACLE:"SELECT ROWID,"DR	126.20852	0.18859	16263	40	406.6	0.06025	0.11543	543064.5	86.0	320.0
11	ORACLE:"SELECT ROWID,"DR	126.76908	0.42642	16159	27	598.5	0.04679	0.36590	257514.4	90.0	508.0
12	ORACLE:"SELECT ROWID,"US	127.64876	0.86505	10847	16	677.9	0.83211	0.02450	79598.0	140.0	537.0
13	ORACLE:"SELECT * FROM "UJ	177.53204	0.09103	2263	8	282.9	0.04991	0.03337	177164.8	173.0	109.0
14	ORACLE:"SELECT * FROM "DR	126.75423	0.57056	2307	8	288.4	0.08417	0.48461	12857.5	173.0	114.0
15	ORACLE:"SELECT * FROM "DI	0.26826	0.46203	2439	8	304.9	0.05879	0.40136	18111.2	174.0	130.0
16	ORACLE:"SELECT * FROM "DI	124.98168	0.95506	2818	8	352.3	0.05115	0.90177	11559.5	179.0	172.0
17	ORACLE:"SELECT * FROM "AE	177.85537	7.03724	2442	8	305.3	0.05539	6.97996	1205.0	172.0	132.0
18	ORACLE:"SELECT * FROM "GR	127.62957	0.92341	3093	8	386.6	0.05761	0.86339	14658.8	175.0	211.0
19	ORACLE:"SELECT * FROM "TH	177.15710	0.11690	2342	8	292.8	0.04932	0.06578	65422.9	173.0	119.0
20	ORACLE:"SELECT * FROM "DR	126.19113	0.33577	2539	8	317.4	0.08630	0.24750	27733.1	171.0	145.0
21	ORACLE:[?] "Logon"	0.01751	0.25004	2941	8	367.6	0.11100	0.13680	33882.3	235.0	132.0
22	ORACLE:"SELECT * FROM "TH	148.69520	3.17819	2631	8	328.9	0.05158	3.12457	3083.5	175.0	153.0
23	ORACLE:"SELECT * FROM "TH	149.15308	0.17292	2286	8	285.8	0.08590	0.08525	41083.8	174.0	111.0
24	ORACLE:"SELECT * FROM "TH	153.06591	0.10687	2265	8	283.1	0.04901	0.05611	65125.9	174.0	108.0
25	ORACLE:"SELECT * FROM "DI	5.31910	119.39251	2575	8	321.9	0.05193	119.33858	79.9	172.0	149.0
26	ORACLE:"SELECT * FROM "SD	130.41615	18.51496	2514	8	314.3	0.06531	18.44770	485.2	173.0	140.0
27	ORACLE:"SELECT * FROM "TH	153.44738	0.18171	2255	8	281.9	0.04075	0.13922	37817.9	174.0	107.0
28	ORACLE:"SELECT * FROM "DI	1.05868	3.87081	2542	8	317.8	0.07641	3.79243	2380.9	173.0	144.0
29	ORACLE:"SELECT * FROM "TH	153.93765	19.58430	2514	8	314.3	0.07411	19.50824	457.1	174.0	139.0
30	ORACLE:"SELECT * FROM "PA	129.13952	0.80399	2423	8	302.9	0.04448	0.75763	10328.5	173.0	129.0

Figure 6: Thread Analysis sorted by application turns.

Note that the thread at the top of the list, the one that will contribute the most to the response time problem, took only 19 seconds to complete in the lab test. Running this through Application Expert's Response Time Predictor will show the performance impact this single thread will have if not tuned. Figure 7 shows this thread in isolation and its predicted response time.

This thread consumes less than 10 percent of the measured (lab) response time, yet will consume almost 50 percent of the response time in the intended deployment environment. If the transaction were bandwidth- or processing-bound, then the Thread Analysis would identify those threads that are bandwidth-sensitive or processing-sensitive.

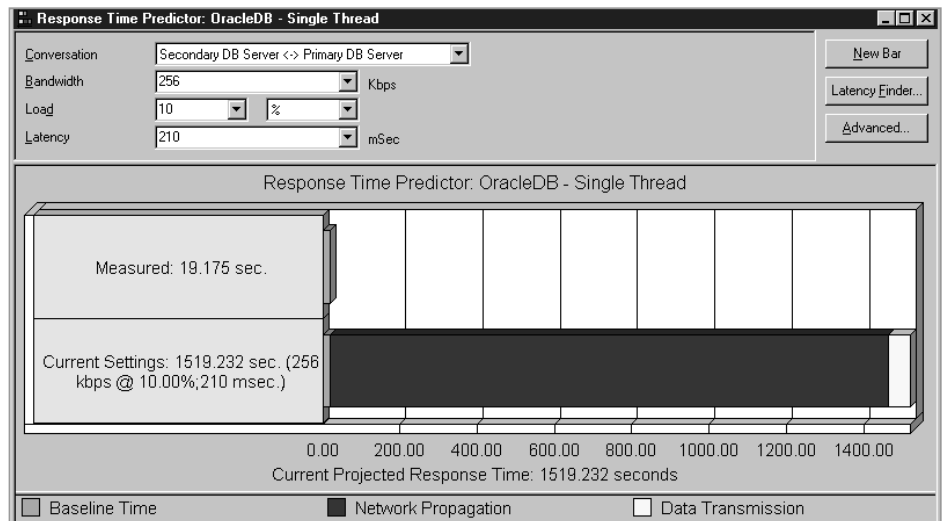


Figure 7: Response Time Predictor, single thread.

Summary

Predictive tuning is a significant component in managing rapid application deployment risk. It provides quantified performance and ROI feedback on architectural decisions, including application logic, network infrastructure options, server placement and data replication. The benefits can be measured in terms of the end-user response times, eliminating time wasted by guessing the reasons for production problems.

Compuware products and professional services— delivering quality applications

Compuware is a leading global provider of software products and professional services which IT organizations use to develop, integrate, test and manage the performance of the applications that drive their businesses. Our software products help optimize every step in the application life cycle—from defining requirements to supporting production service levels—for web, distributed and mainframe platforms. Our services professionals work at customer sites around the world, sharing their real-world perspective and experience to deliver an integrated, reliable solution.

Please contact us to learn more about how our comprehensive products and services can help your organization improve productivity, create higher quality applications and ensure performance in production.

All Compuware products and services listed within are trademarks or registered trademarks of Compuware. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other company or product names are trademarks of their respective owners.
© 2002 Compuware Corporation



www.compuware.com