Diagnose and Cure Application Performance and Availability Problems

Authored by:

Bill Hayduk
Director of Professional Services
RTTS

It is important for businesses to seriously consider taking proactive steps to manage application performance before issues grow into serious business problems"
- Newport Group

"…Software bugs cost the U.S. economy an estimated $59.5 billion per year…. An estimated $22.2 billion could be eliminated by improved testing that enables earlier and more effective identification and removal of defects."
- U.S. Department of Commerce National Institute of Standards and Technology (NIST)

"…89% of the 800 IT managers contacted say they've experienced software-quality problems within the past year that resulted in higher costs, lost revenue, or both."
- InformationWeek

Through 2005, more than 80 percent of application performance and availability failures will be blamed on network problems, but the network will represent less than 20 percent of the root cause (0.7 probability)."
- the Gartner Group

"Software quality has always been a problem. Companies have been willing to live with that previously. Now they can't live with it because they don't have a business. Apps and software used to support business; now it is the business. That's a major difference."
- IDC

"*The heavily promoted New York fashion show featuring Victoria's Secret's spring lineup took place … online and attracted 1.5 million visitors when they expected a few hundred thousand.*
People attempted to log on to the show, swamping the site, slowing response time, and leaving many frustrated visitors viewing only error messages."
-CNN

In 1999, Victoria's Secret held an on-line show and many customers could either not get in or experienced tremendously slow response time. What would customers have done if this happened to Amazon.com? How many potential customers would immediately take their business over to barnesandnoble.com? It was a good thing for Victoria's Secret that the majority of the viewers were men that would come back again, regardless of the site's performance issues. But very few firms have such a dedicated (and captive) audience.

With the current economic climate of recession and the strong competition for dollars, it is absolutely vital that companies assure the user experience is a good and productive one, or be prepared for customers to take their business elsewhere.

The "Good Old Days"

Back in the "good old days" of the 1990's, the issue of assuring that production applications scaled, provided a good user experience and allowed users to process transactions quickly was typically not crucial, due to the fact that most clients were internal customers (the exception being business-critical applications, such as brokerage firms' trading applications). There was an understanding that software applications were "buggy". But one could accurately estimate the size of the expected user population and it was fairly easy to estimate the maximum concurrency of users. Therefore, scalability/performance tests could easily assist with tuning the application and architecture in a test environment during pre-deployment. Besides, with most architectures being 2-tiered *(figure 1)*, it was relatively easy to pinpoint the offending area: either the client, the network, or the database. Monitoring of the architecture typically consisted of the database administrators (DBAs) periodically watching native monitors of the databases (i.e. Sybase, Oracle, etc.) and the network administrators looking at archaic sniffer captures to determine network saturation.

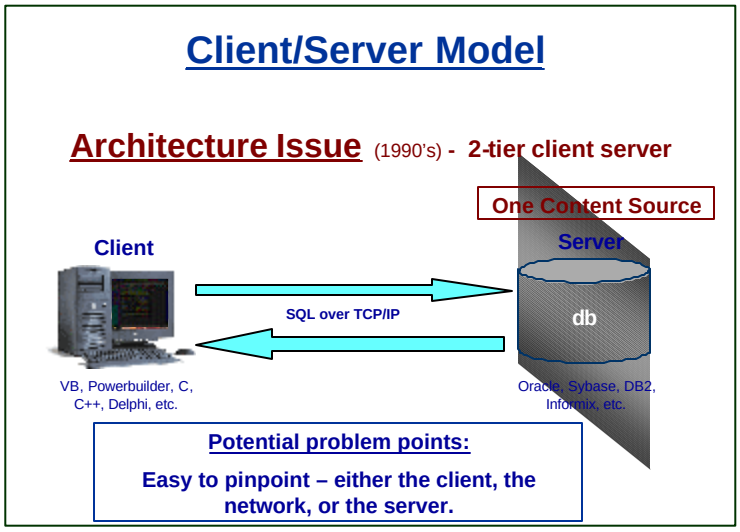


Figure 1

The testing strategy applied was one of "black-box" testing. Black box testing is testing from the interface perspective, where everything that goes on beyond the interface is thought of as a black box, and is not open to the tester. This strategy works well with 2-tiered systems, because of the ease of pinpointing the problematic area.

Companies who were diligent about software testing and test automation implemented a process with testing stages very similar to that in *figure 2*. Performance was hit and miss. If it performed poorly or could not scale well, it was tweaked in production or retested in the test region. Pre-deployment consisted of the unit, integration, system and acceptance phases of testing while post-deployment consisted of troubleshooting the client side for installation errors, the server side for CPU & memory utilization and disk I/O.

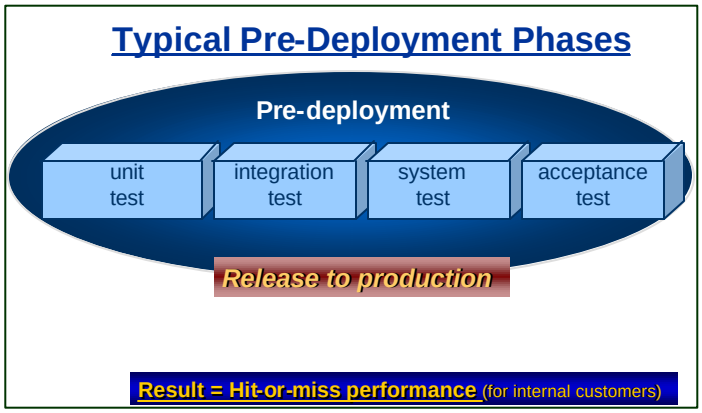


Figure 2

Today and the explosion of e-commerce

Today many things have changed. The most important are:

- **Clients are often external customers, with the ability to leave if service is unsatisfactory.** This is the biggest change – the e-commerce business-to-business (b-to-b) and business-to-consumer (b-to-c) explosion over the Internet has drastically altered the consumer landscape. The good news is that you can now attract clients from vast geographic regions. The bad news is – so can your competitors. The Internet puts the power in the hands of the consumer, providing a very easy path to alternative choices if your offering is unsatisfactory.

- **The user population and usage pattern may be entirely unpredictable.** It may now be impossible to predict the user population of your application at any given time (as with the Victoria's Secret incident).

- **The architecture is heterogeneous and multi-tiered.** The architecture is now a mix of different applications running on different hardware, software and operating system platforms, sometimes combining many different architectures into one heterogeneous system.

- **Any part of the transaction outside your firewall is subject to a wide range of variables.** Because of the complexity and ubiquitous nature of the web, clients could be using a whole host of browsers, different versions of each browser and different settings, different operating systems and versions, and different connection speeds with a different number of hops to the application, all of this beyond your control.

Figure 3 shows a heterogeneous n-tiered web architecture that is probably much simpler than what most firms have deployed.

Obviously building a much more complex model than the 2-tier client/server days, companies now deploy architectures that have the following characteristics:

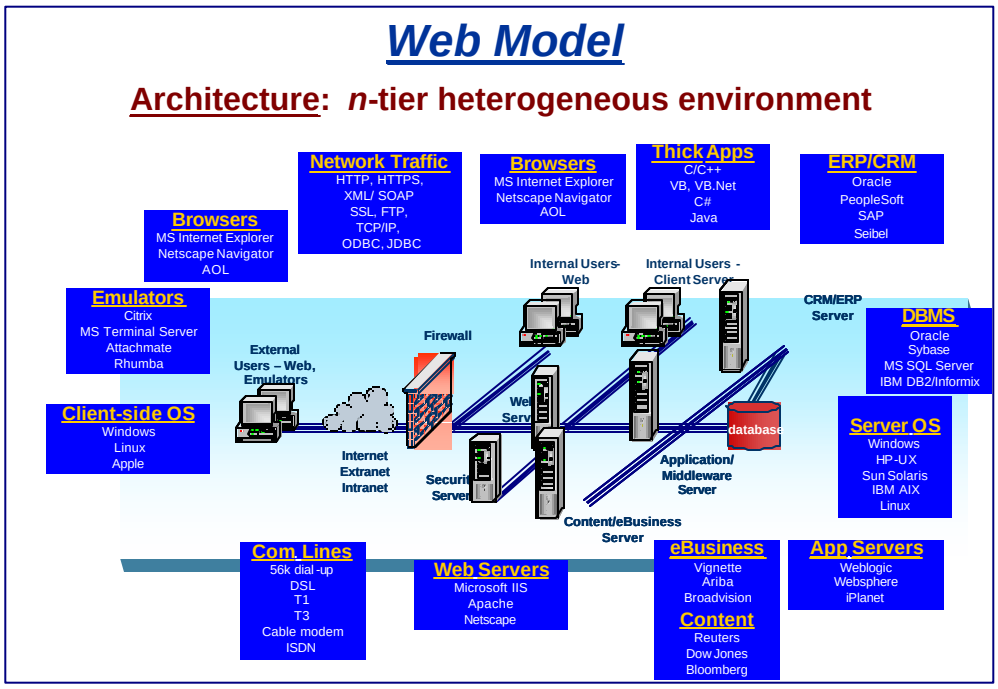


Figure 3

- External clients utilizing different browsers and versions, or thin clients (i.e. Citrix) running on various operating systems.
- Internal clients running thick client applications (C++, VB, etc.).
- These external and internal clients connecting over different communication speeds using many different protocols.
- Interacting with web, application, e-business, content, ERP, CRM and database servers, all sitting on different operating systems.

Now when firms use the old model of black box testing for functionality, performance and scalability, they may or may not find any issues in their unit, integration, system and acceptance phases, but when they roll out the application, it suffers in production. Clients complain and senior management wants an answer.

Some of the reasons the application may not scale in production after they scaled during testing are:

- The hardware in the test environment is not identical to production.
- Routers, switches, etc. are not being utilized during testing.
- If a web architecture is involved, testing does not take into account the internet's characteristics (multiple hops, connection speeds, etc.).
- Security, encryption and compression may not have been implemented during testing.
- Software and/or hardware changes or upgrades may have occurred in production.

So while the application has been tuned in the test environment, it has not been optimized for production.

And then there are the specific business issues that will arise and are critical that they be resolved.

These business issues are:

- End user response time is too slow.
- End user performance exceeds SLA agreements.
- The application failed on the user's workstation.
- The end user cannot process the transaction.

These result in a loss of customer confidence and loss of business due to poor end user experience.

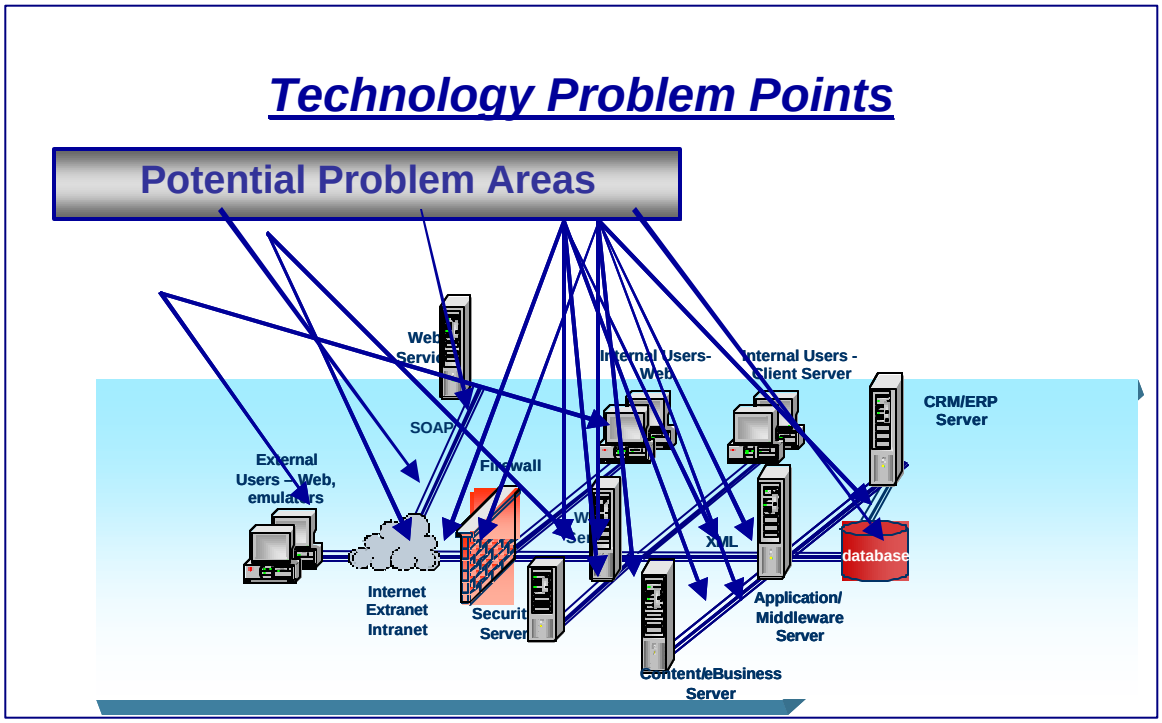


Figure 4

Those technology issues that arise and are difficult to diagnose due to the complexity of the current architecture are:

- Excessive memory or CPU usage on the client.
- Problematic SQL calls or HTTP statements.
- Network latency, chattiness or network error issues.
- Abnormal or unauthorized applications running on the network.
- Unexpectedly large file transfers or data dumps.
- SQL statement errors.
- Services/processes that are not available.
- The inability of technology group and/or operations to view and monitor all components from a single console.

When performing this testing through the black box approach during pre-deployment, we may be able to identify that there are problems with functionality and performance (the answer to the question "*What is the problem?*"), but we will not answer the rest of the questions, specifically:

- *Where is the problem located?*

- *Why is the problem causing poor performance?*

- *How do we fix the problem?*

Customers already know there is a problem (i.e. poor performance). They expect us to assist with answering the "where", "why" and "how" questions. These questions cannot be answered with a black box approach and testing cannot be performed in post-deployment with conventional functional and performance testing tools.

The Solution

The solution to finding and fixing problems in post-deployment comes down to three action activities: monitor, diagnose, and tune *(figure 5).*

Monitor – Periodic checking of client systems, networks, servers and databases.
Diagnose – Troubleshooting the precise nature and cause of performance bottlenecks.
Tune – Fine-tuning the architecture to provide optimal performance.

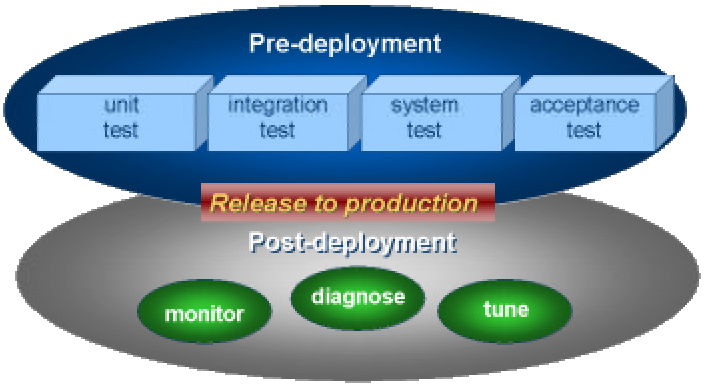


Figure 5

Both monitoring and diagnosing are performed utilizing test tools that allow for drill down into the pertinent information. Tuning is performed manually by adjusting settings on the applicable client and server software, operating systems and hardware for optimum scalability and performance.

Client-Side Monitoring

Monitoring is performed from the client-side and from the server side. Client-side monitoring uses agents to run sample transactions to emulate the user experience. Typically this software can monitor client-side resources and application faults and can do this for web, client/server, 3270 emulators and Citrix applications. Agents reside on user workstation and use a scripting tool. The best of these can initiate network traces of transactions, reporting when threshold is exceeded. Monitoring agents measure the end user application availability and response times – by transaction, location and time period. The agents also provide automatic detailed diagnostics and drill down into transactions that fail to meet service level thresholds. This is a good way to catch the intermittent performance issues. For web and database applications, they display critical timings down to the application thread level *(figure 6).* Also, client-side monitoring tools can send emails, phone messages, pager messages and other alerts when thresholds have been exceeded.

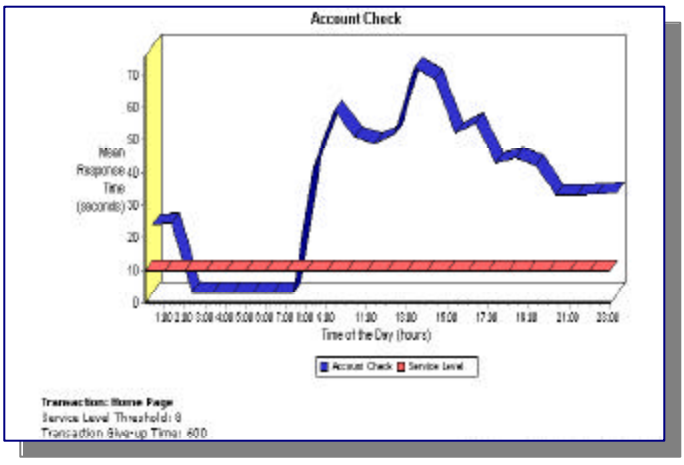


Figure 6

Focus
- o End-user experience
- o Client-side resource monitoring
- o Application fault monitoring
- o Typically monitors web, Windows, 3270 and Citrix transactions

Technology
- o Agent resides on user workstation
- o Uses scripting tool for active or passive modes
- o Initiates trace of transaction, reporting when threshold is exceeded.

Benefits
- o Validation of end user performance
- o Notification of declining performance
- o Alerts execute if business transactions exceed thresholds
- o Diagnoses application faults on end-user PC
- o Answers whether client, network, server or application are the issue

Server-Side Monitoring

Server-side agents can monitor the operating systems, databases, and applications running on Windows, UNIX, and Novell servers. Many tools monitor hundreds of metrics out of the box. They can also correlate the data, so you can quickly understand how database issues are impacting application performance. Many can initiate a corrective action, such as automatically restarting a critical service or process that has failed, which reduces user downtime. Server-side monitors usually have the ability to send notification via emails or paging.

These tools allow you to pinpoint server-side spikes and provide drill-down capabilities to determine the cause of the spike.

The main benefits to using server-side monitoring are: better server management, reduced cost of server expansion, notification and correction of critical server or database issues before applications fail and performance trending for pro-active planning.

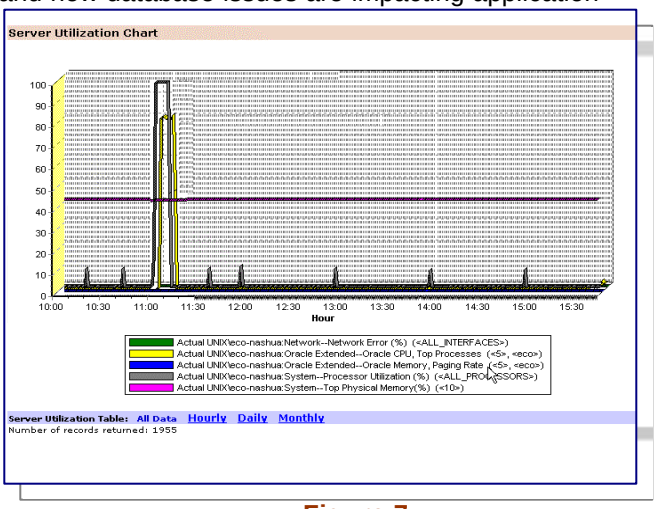


Figure 7

Focus
- o DB servers, application servers, web servers, mail servers
- o Application, web, and database server resource usage, monitoring

Technology
- o Agent resides on server
- o Operating system monitors (i.e. UNIX, Windows)
- o Application monitors (i.e. SAP, PeopleSoft, Oracle apps, MS Exchange, Lotus Notes).
- o Application Server monitors (i.e. WebLogic, Websphere, iPlanet, etc.)
- o Database monitors (i.e Oracle, Sybase, DB2, etc.)

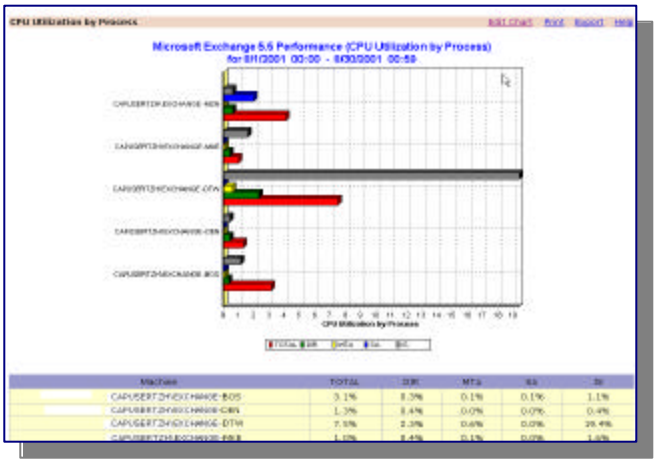


Figure 8

Benefits

- Identify CPU, memory, db/sql problems, service availability
- Troubleshoot with detailed diagnostic reports
- Can re-establish network connections, restart apps.
- Provides notification via paging, email, etc.

Network Monitoring

Network monitoring tools collect application performance metrics (traffic volume, response times) for applications and integrate data from both a LAN and WAN perspective. Network monitors collect statistics from all application conversations on the wire and organizes the data for performance evaluation. These tools help identify the network load and pinpoint unusual activity and can also help validate security policies, analyze security breaches and provide a historical audit trail of who was doing what on your network.

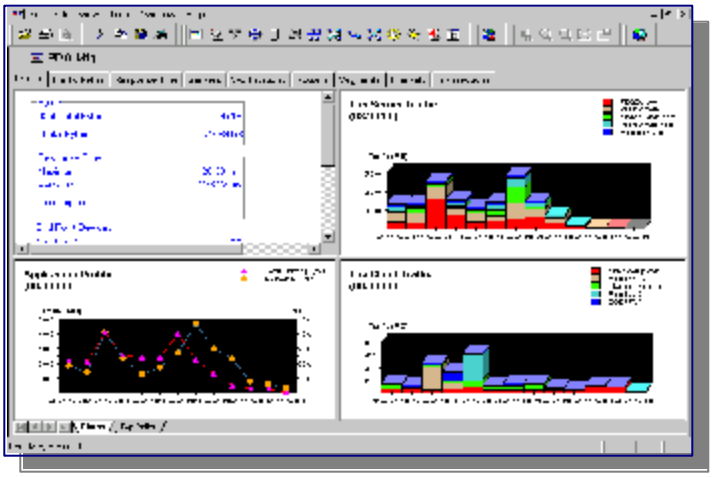


Figure 9

Some tools can view WAN utilization in both directions of a particular WAN link and some can show when traffic bursts occur. One tool can automatically identify all applications and who is using them and when the activity is happening. Network monitoring answers the questions of *what* (application), *who* (workstation), *when* (time of day), *where* (topology), *how much* (traffic), and *how long* (response times). The data generated from the tool provides a convenient way to track network use and abuse. The main benefits of network monitoring tools are: reduced problem analysis and resolution time, better controls on network usage resulting in cost containment of bandwidth upgrades, audit trail and visibility.

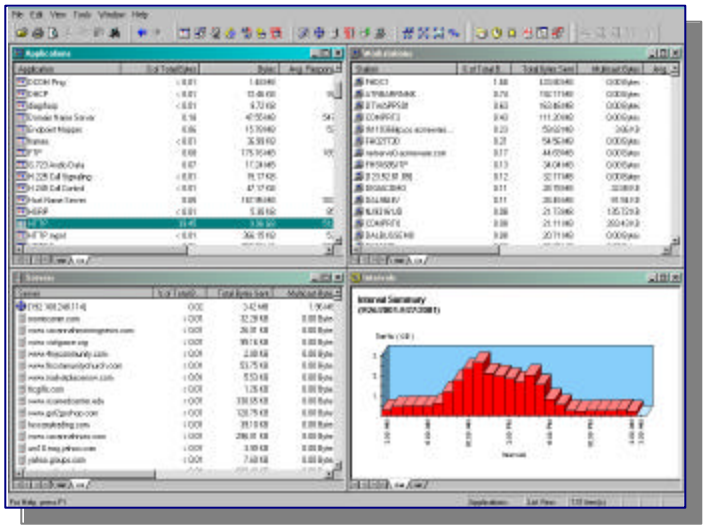


Figure 10

Focus

- Application usage across WANs, LANs
- Determines which applications are being run where

Technology

- Uses hardware probes throughout the network segments
- Can typically identify most commercial applications out of the box

Benefits

- Provides Identification of who, what, when, where, how much and how long
- Tracks network use and abuse
- Drill down to understand poor response times
- Validate security and provide audit trail

Application, Network and Server Diagnostics

Diagnostic tools are utilized for finding and "diagnosing" issues in post-deployment. Tools that separate the application from the network from the servers are excellent at diagnosis. These tools utilize multiple packet capture trace files that are merged to depict a true end-to-end transaction.

These tools help to quickly isolate a poorly performing application and break it down into application threads (HTTP or SQL calls) so that developers and DBAs can understand exactly where the problem is located. They will quickly reveal whether applications that can be improved with increased bandwidth or whether the are performing poorly, due to design flaws. They can determine whether the issue lies in the application, the network or the server (*Figures 11 and 12*).

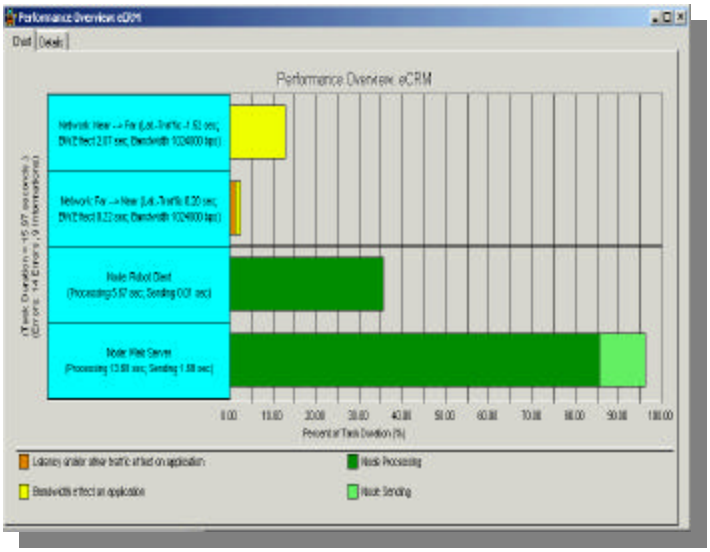


Figure 11

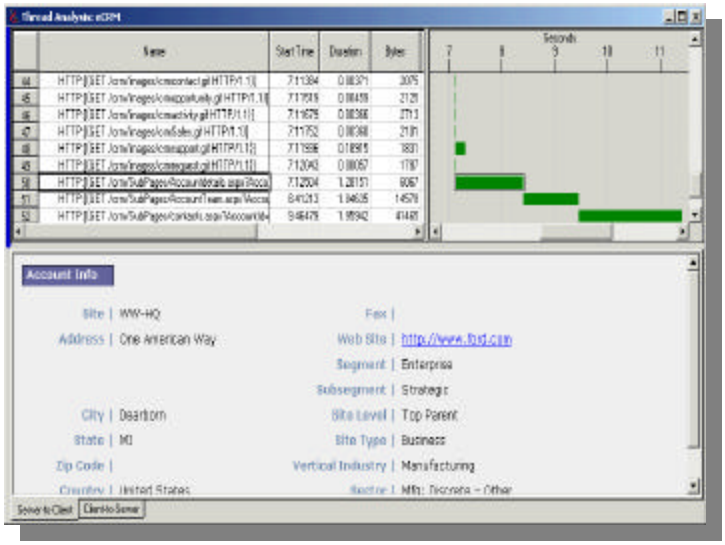


Figure 12

Focus

- Monitor conversation between application and server
- Problematic SQL calls or HTTP statements.
- Diagnose network latency, chattiness or network error issues.

Technology

- Uses packet capture traces to analyze threads
- Can perform direct network capture or import standard sniffer file types

Benefits

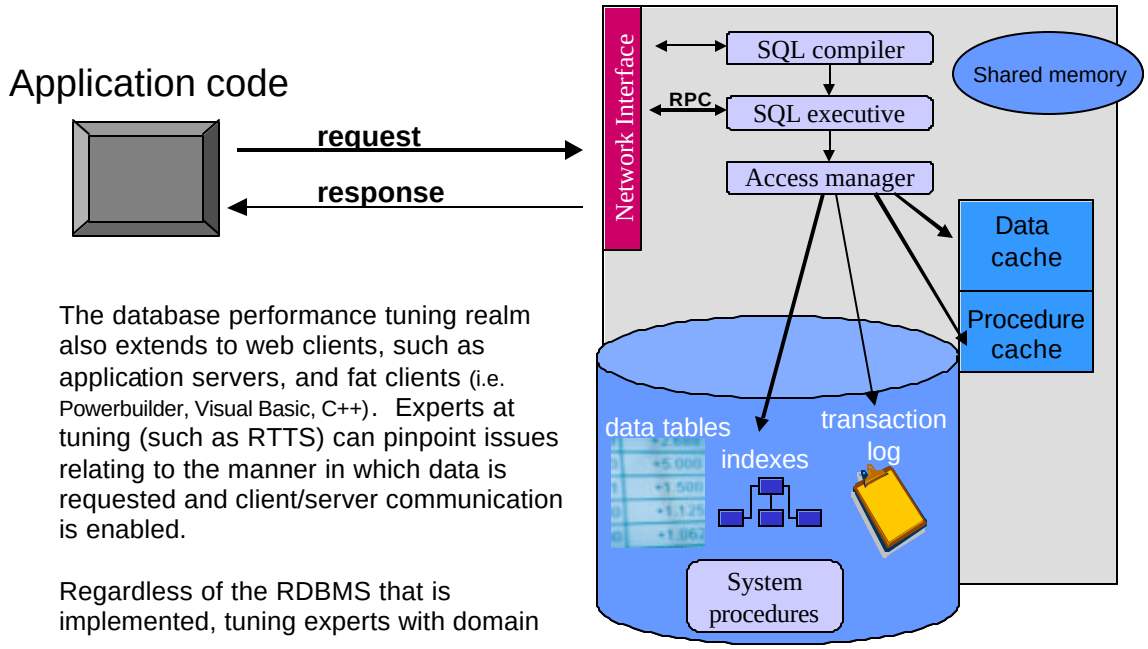
- Pinpoint the source of poor performance.
- Drill down to application threads.
- Visualize performance issues.
- End finger pointing between network, db and app teams.

Tuning

Tuning of database and application servers requires expertise and experience garnered from working on specific IT architecture.

Database Tuning

Database performance tuning is the iterative process of analyzing the ramifications of hardware and/or software configuration changes with the intent of increasing application performance while minimizing costs.



*courtesy of **Sybase, Inc.***

The database performance tuning realm also extends to web clients, such as application servers, and fat clients (i.e. Powerbuilder, Visual Basic, C++). Experts at tuning (such as RTTS) can pinpoint issues relating to the manner in which data is requested and client/server communication is enabled.

Regardless of the RDBMS that is implemented, tuning experts with domain expertise can provide solutions for tuning database servers and their clients. Although the configurable parameter terminology differs by platform, the same performance tuning concepts apply to all database server vendors.

- *Determine the level of tuning* – Component-level tuning or system-level tuning? Do you want to tune the database server as an isolated component or as part of a larger application?
- *Understand the end-user community* – Gather metrics regarding the manner in which the database will be accessed. What SQL queries will be executed? What business transactions will be executed? How often are transactions executed?
- *Gather Performance Requirements* – Determining the exit criteria for tuning needs to be established in order to know when sufficient testing has occurred.
- *Implement diagnostic tools* – Implement tools that issue the necessary SQL queries, updates and deletes that emulate the business scenarios.
- *Application Profilers* – Implement tools to profile transaction characteristics. Determine the network characteristics of a transaction, such as bandwidth utilization and conversational chattiness. Ascertain the CPU utilization on the database server and client, memory utilization, query compilation and execution times.
- *Analyze Results* – Execute the transactions and collect metrics, such as response times, transaction volumes, operating system statistics, database server statistics.

Armed with a proven testing methodology and best practices, experts can provide an integral solution for resolving many issues associated with the relational database management system (RDBMS) and operating system kernel parameters. These may include:

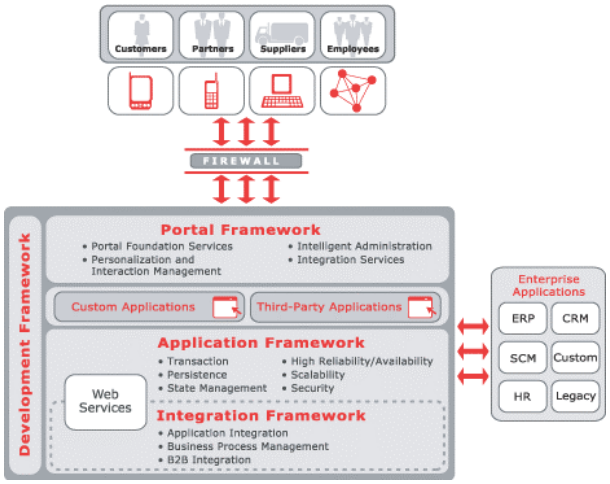
- Providing an inventory of slow or inefficient database queries
- Determining the proper size of connection pools to support the arrival rate of SQL requests
- Discovering the inability of a RDBMS to scale on a multiprocessor database server (RS/6000 SMP)
- Establishing the best configuration sizes for data and procedure caches
- Ascertaining the best hardware platform to implement
- Discerning the most efficient auditing scheme that would prevent deadlocks, while maintaining history of the business processes
- Validating the correct indexes to employ, such as clustered indexes versus non-clustered indexes

As a result, database server capacity and scalability are increased by addressing:

- the use of a small packet size between the client and the server
- chatty conversation over high latency network links
- large amounts of unused data returned to the client
- redundant database queries
- additional tuning methods

Application Server Tuning

Conceptually the role of the application server is rather simple. Application servers comprise the core business processes and logic within a distributed N-tier application and act as the "glue" between client requests for application services and back-end data sources. However, there are many configurable parameters that affect the efficiency of this process, such as available memory, the number of available threads, cache sizes and connection pools, as well as the workload of the end-user community. Requests for application data need to be executed simultaneously and resolved in a timely fashion with finite hardware and software resources.



courtesy of ***BEA Systems***

Since the implementation and objectives of the application server are unique to each individual application, there is quite a bit to gain by tuning the application server based upon its specific business requirements and available resources. For instance, the following symptoms all have been experienced within previous client applications and were solved through incremental configuration changes within a Java application server.

- Fluctuating or periodic increases in end-user response times
- Increasing response times over time
- Under utilization of one or more servers within the application
- Connections refused by the application server
- Memory leaks
- Limited throughput
- Java Exceptions
- HTTP Internal Server Errors

Typical solutions involve tuning the Java Virtual Machine (JVM) heap size and garbage collection, configuring the number of application server threads that handle end-user requests, upgrading to a newer JVM or configuring the connection pool size to a back-end data source.

Regardless of the specific application server vendor, tuning experts (such as RTTS) have standard solutions for tuning applications running within any application server platform, such as WebLogic, WebSphere, IIS/ASP, ColdFusion, JRun, Microsoft Transaction Server (MTS), CGI/FCGI, etc. The same performance tuning concepts apply to all application server vendors, although the terminology differs.

- *Determine the level of tuning* – Component-level tuning, such as Java Servlets, EJB's, COM/DCOM components, or system-level tuning?
- *Understand the end-user community* – Gather metrics regarding the manner in which the application will be used. What components will have a higher degree of exposure and pose the greatest risk? What business transactions will be executed?
- *Gather performance requirements* – Determining the exit criteria for tuning needs to be established in order to know when sufficient testing has occurred.
- *Automate test scripts* – Create automated test scripts that can invoke the necessary component services or drive the business scenario.
- *Application profilers* – Implement ancillary tools to profile transaction characteristics. Determine the network characteristics, CPU utilization, SQL calls that a particular transaction or component exhibits.
- *Analyze results* – Run the planned tests and collect metrics, such as response times, transaction volumes, operating system statistics, application server statistics.

Making the ROI Case

The bottom line is that monitoring, diagnosing and tuning applications, either separately or collectively, will cost more money up front to implement. What is the return that can be expected from this investment? What are the benefits? Let's take a look at these on a high level.

1) You will save money on LAN/WAN upgrades by determining whether performance is actually a bandwidth problem (it usually is not).
2) You will save time on resources, pinpointing the problematic area (database, network, application server or client).
3) You will save expensive downtime by proactively monitoring and diagnosing problems before they happen.
4) You will save clients by monitoring and making sure that any degradation in the user experience is dealt with swiftly.

Conclusion

Although rigorous scalability/performance testing in pre-deployment adds tremendous value with regard to the application's ability to scale in a test environment, it does not assure a quality experience for the user and availability of the application 24x7 in post-deployment. To proactively assure that your application is available, functional, reliable and scalable, the solution for post-deployment is straightforward:

- monitor your application,
- diagnose the problem and
- tune the environment.

These action items are essential for firms concerned with providing a great user experience and with retaining their customers.

About the author

Bill Hayduk, founder/president of RTTS and director of its professional services group, has an excellent reputation in the technology field and is particularly noted for his test methodology and test automation expertise. Over the last 20 years, Bill has successfully implemented large-scale projects at many Fortune 500 firms. He has consulted in various sectors including global banks, brokerage firms, multimedia conglomerates, pharmaceutical, insurance and software companies.

Bill holds a Master of Science (MS) degree in computer information systems from the Zicklin School of Business (Baruch College) and a Bachelor of Arts in Economics from Villanova University. He has been a selected speaker at industry-specific trade conferences, as well as a source of information for corporations and has been referenced in many industry trade publications.

About RTTS

RTTS is a professional services organization that specializes in the testing, monitoring, diagnosing and tuning of IT applications and architecture. Serving Fortune 500 and mid-sized companies nationwide, RTTS has offices in New York, Boston, Phoenix and Orlando. RTTS draws on its expertise utilizing best-of breed products, expert test engineers and proven methodology to provide the foremost end-to-end solution that ensures application functionality, reliability, scalability and network performance.

From strategically planning your entire testing approach to tactically implementing the effort in pre-deployment and then through monitoring transactions and diagnosing bottlenecks and other post-deployment problems, RTTS provides a full-cycle iterative solution for your software quality needs.

RTTS offers full outsourcing of your entire testing needs or can provide you with individual test tool product expertise. We offer expert mentoring and education services, along with a proven game plan for providing knowledge and skills transfer.

To learn more about RTTS, visit www.rttsweb.com.