Support Strategies for E-Commerce Sites

Presentation # 303

Written by Don Manzano

E-Services Support Center
Hewlett-Packard

8000 Foothills Boulevard, M\S 5713 Building R21
Roseville, Ca.
95747-5713
Phone: (916) 785-0163
Fax: (916) 748-1920
Email: don_manzano@hp.com

Introduction

"Revolution is a science few are competent to practice" -- Robert A. Heinlein

Revolution barely captures the current wave of Internet progression. The phrase "dot com" has advanced from the DNS computer-geek slang into the consumer driven global dialect. Web-presence has surpassed the computer "brochure" of a company to pursue personalized web experiences to individual consumers. Sites powered by multimillion-dollar enterprise designs offer outstanding free services. For the simple price of eye bothering advertisement, a consumer reaps excellent free benefits. For example, free ISP connection www.bluelight.com and free long distance phone calls, www.dialpad.com.

Seamless to consumers, and hopefully the press, enterprise B2C and B2B sites fight the persistent downtime threat. Scrambling not to make the headlines or news reports, support and development staff tire themselves supporting complex enterprise environments.

The hope that a user friendly, consistently available website will gain e-consumer loyalty is the goal. A site that is crippled or down completely during peak hours can turn e-consumers away, hesitant to return. The obvious goal is for zero downtime. Achieving this goal is less elusive with accurate and precise support strategies. Correct, well thought out support strategies, will guide customer satisfaction and consistent site revenue.

The scope of this presentation will exclude flashy marketing terms or high-level profit projections of B2C and B2B sites over the next five years. For forecasts on B2B and B2C futures turn to "The Gartner Group" www.gartner.com.

Support is not a future forecasted item. If accurate support models were forecasted, the accurate forecast would be "sooner or later, it's going to break". Precise support strategies will minimize downtime and maximize profitability. Site aesthetics are eye catching, but the lack of uptime shames the hardware, design, and management. The shame filters into the press and into the minds of shareholders and pre-IPO venture capitalists.

Internet consumerism is changing. Think when your first online shopping experience was. Maybe it was as early as a day ago, or maybe you have shopped online for two to three years. The online shopper three years ago was more of the technical type. Today, the online

B2C sites are describing consumers as your "everyday shoppers". Online ease, competitive prices, and wide selection attract novice online users to shop, shop, shop.

Regardless of the TV Internet reports and newspaper tech columns forecasting billion dollar sales over the next year or two, your site is what counts! Even a small insurance site, that is nothing more that an online brochure is a 24x7 operation.

Support of commerce sites is beyond a one system or environment setting. Enterprise level applications and third party channel partners are all in the mix together. Three and four-tier architecture link into other multi-level operations to complete one web transaction. For example, buying any product where a credit card number is given. The site that the product was purchased has a support agreement and a dedicated leaseline to a banking institution. The credit card request is sent and the bank, in it's various high-speed decision algorithms, sends back approval. Then, if it is a good site, an order confirmation is given back to the online consumer. There are issues in every step of this process. Solely, these processes have proved stable. It is when placed together, all must work as a new web transaction unit.

The Project

"Imagination is more important than knowledge" – Albert Einstein

If management, marketing, and graphics designers had to think of all the bits and bytes that go into producing a site, websites might look pretty boring. As support engineers and topology designers, we need far out impossible ideas from our companies creativity department. A simple project brought to the design table is not worth doing.

"0 downtime" and "Let's try to achieve 5-9's on this project" are valid support considerations. Acknowledging these two goals will keep the company out of the headlines and into the financial news. Early considerations for support will help ensure the uptime goals for the customer base is accomplished. Companies with seasonal clientele, www.hersheys.com and www.sees.com, generate large revenue during Halloween and Christmas. In this case the site will probably need to be designed to handle a heavy consecutive volume; maybe 15,000 consecutive users.

To get the site to the needed consecutive user base, support plans have to be in place. Recovery is vital if the site panics during peak operation.

Snap Shot

"Anything I've ever done that ultimately was worthwhile... initially scared me to death." -- Betty Bender

Gaining a clear understanding of the supported web transaction for the site is paramount. If the working and moving parts are aloof, then fault isolation latency increases. Even a small B2C site is complex. Here is a snap shot of a small B2C example.

A large site may have a direct connection from one ISP to your site. For example, if your company is a key word on AOL, you may opt to pay the price and have a direct connection to your site for any AOL user. This means that the AOL user, will more than likely get a faster connection, and critical cached application and pages may be displayed faster. For the average company, a connection to a large ISP, such as UUNET, for the backbone is sufficient. When the user types in www.quickthought.net, the request is answered as a DNS request by the ISP. Then the correct IP address is given to the client who then opens a TCP connection with the IP address given.

Quickthought.net will be the fictitious company that will demonstrate a simple understanding of a personalized web page transaction. This example assumes that the user interface with www.quickthought.net had no problems and produces at least index.html to the end user.

There are 3 major system tiers.

1. Webserver
2. Application server
3. Database server

Between each of the above pieces there is a LAN running at 400mbs using Auto Port Aggregation. The firewalls and redirectors are the same, Cisco Secure PIX firewall.

The user enters www.quickthought.net into the browser. The request is answered and the webserver displays a special page called index.html. This edition of index.html is extraordinary because it is just the entrance into the site.

Example:

```
<html><head>
    <title>Quickthought.net discount Coffee</title>
    <meta http-equiv="refresh" content="0;URL= "/coffee/home.jsp”>
</head></html>
```

The contents of index.html actually take you to www.quickthought.net/coffee/home.jsp. This is completely seamless to the user. This connection brings up the default webpage for www.quickthought.net. This method is used for user ease. The marketing staff would strongly object to advertise the URL as www.quickthought.net/coffee/home.jsp on the freeway billboards. Companies that use complicated pieces of middleware for their B2C and B2B sites, use a similar methodology.[1]

This URL is special because it activates the application, which is named coffee, and coffee is serving home.jsp.

The webserver in this example is doing a small job. It is serving index.html that redirects to an application. The application then sends the compiled web page to the webserver who then serves to the client. The webserver is doing no directory look-ups or maxing-out CPU.

Coffee is the name of the middleware application. Applications, such as BroadVision, allow you to name the application that will be communicating with the middleware backend and database so the URL may stay within acceptable marketing terms. For example www.walmart.com sends the user to www.walmart.com/estore/pages/pg_g1.jsp. The application in this example is estore.

Home.jsp has embedded html code that the application adds the dynamic content and then delivers the html code to the webserver. The compiled html document, compiled by the application named coffee and has an area with a “Register as a new user”. The user clicks registers by answering the profiling questions and submits. The onSubmit function sends the profile information to the database. The next time the user logs in, the application coffee will pull the user information from the database and place the information in the valid fields of a template and serve the page. The served page will display in the banner “Hello $USER, welcome back. It’s been $DELTATIME since you’ve been here”. The proceeding values came from the database and were entered on the page. The coffee application used the proprietary API’s to perform the function and, HEY, personalization!

[1] For example: www.walmart.com, www.quicken.com, www.gartner.com, www.theluggagecenter.com

Pre Race Instructions

Supportability discovered

Successful support begins with the basic network and system design. As in benchmarking network designs for NFS, web traffic comes in waves and the system has to be able to handle the strain. The waves of traffic now have to be considered not over 3-4 network topologies, but through the ISP-to the backbone-to the site route-to the webserver-to the application server-to the database for product look up-to inventory management.

Conceptualizing the specific environment is where the support needs to begin. Confusion can be limited if the end to end transaction is supported modularly. Dissecting the modules reduces resolution latency. If there are end user complaints, it is probable that the ISP or backbone could be the problem. There is little a support center can do if the problem is on the user side of the connection. There are many investigative tasks for a site support engineer that is alerted to a site-crippling symptom within the environment. Understanding each separate piece then the integration can optimize resolution of the issues.

The application server or servers is the where the performance problems and site delivery issues will commonly take place. This tier is where the templates are stored for dynamic content placement and where the application executable runs. In the above example this was named coffee.

The application is a sum of all the .jsp files (can also be in perl, java, etc.), the API to the backend database, and the executable of the middleware product. The web content for the application is depended on the push of dynamic content and developer defined templates.

Sites that push advertisements or content built by the application executable before handed to the webserver increases the compilation time. The push of content is the key selling point for many middleware products. The overhead generated by the push of content is negligible to the dynamic generation benefit. The push of content and code changes helps to eliminate development time and developer error.

Support concerns and strategies:

1) Update pages and content on a different network. Separate the web content delivery LAN with the update and maintenance LAN.
2) Establish set maintenance modes and judgement of the content.
3) Establish clear version control of the dynamic content.
4) Monitor the amount of dynamic content built by the application. The more content that is not hard coded, the slower the page will perform. If the page has to server multiple pages in one, for example a nested frame or table, the delivery time will be measured in units of second not milliseconds.
5) Make sure that the redirect, either by index.html or local redirectors that the redirection pages has minimal dynamic content. The redirect may already have an attached latency, try not to increase it with a slow front page. For example, the weight of www.quickthought.net/coffee/home.jsp should be an order of magnitude lighter than the custom profiled pages. When analyzed, judge the site with tools like SiteSeer and WebStone. The weight of the page needs to be extremely small to optimize performance.

Nothing like custom made equipment

Personalization revisited

Dynamic information includes the personalization or profile matching pushed advertisement. Personalized profiling is displayed in MyYahoo! Dynamic content is pulled from the database and pushed to the application to be placed in templates that are severed to the webserver and then to the end user.

For this example, a login function will be activated to pull the user information from the database. The stored user profile information is evaluated against a rule set in the database to push dynamic content. Example, one profile may be a 20-year-old college student that makes under 20,000 a year. Quickthought.net may push an ad for student loans and a deal on golf lessons. The other profile may be an over 45-year-old computer executive with a salary greater than 100,000 a year. Quickthought.net may push an ad for a membership to an executive tennis club and the new BMW sport utility vehicle.

If the login is advantageous to the user, such as Amazon.com's "one Click", the user database could grow to hundreds of thousands of entries. Imagine the user load for MyYahoo!. Consider that there will be duplicate users. Think of how many times you have forgotten your username and login, so you created a new user to use the web site.

Support concerns and strategies:

1) If the user database will be heavily used, make it easy in your site to regain the username and password. This will reduce the amount of duplicate entries.
2) Reduce the size of the customer profile database. Concentrate on maximizing given information in the customer profile instead of adding to it.
3) The application server in this example is at tier 2 and the database is at tier 3. Make sure that the network route to the database is not 2-5 network hops way or across a WAN.

Combination of a large user profile and the product database will require a mass storage device and a fast database system. The middleware platform that is used will have a database schema to user profile data. The requirements for a specific site will vary. For example, if quickthought.net is going to concentrate on large user profiles and large amounts of dynamic content, then the database will be the profile columns and the dynamic information plus evaluation rules to server the dynamic content. Or, quickthought.net could specialize in selling large varieties of coffee with little user profiling. In this case the product database for quickthought.net would need to be quite large. Or, quickthought.net could be adventurous and incorporate large dynamic user profiles with a huge product database. The database would have to be tuned to hold and serve the large amounts of information in a timely manner.

Where the rubber meets the road

Without the web servers, nothing would work

The webserver choice will depend on middleware compatibility. The set of middleware API's will suggest the most compatible webserver. For example, the middleware product BroadVision recommends the Netscape server suite. BroadVision has a set of API's and custom NSAPI module to optimize performance.

The gateway to the middleware application resides in the webserver configuration. In the case of Netscape, the "obj.conf" file has an entry for the BroadVision NSAPI module and defines the gateway name. For www.quickthought.net/coffee/main.jsp , the gateway name is coffee.

The network route to the gateway name application, coffee, is through a firewall leading into tier 2.

Support concerns and strategies:

1) Is the route setup through the firewall to the application server correct?
2) Is the webserver setup correctly and is the correct NSAPI module installed once configured?
3) Is their redundancy in the web servers?
4) Are the web servers able to load balance application requests to tier 2 servers?

The route to the application server will be yet through another firewall with a set of ports defined. Correct port allocation and integration into the application will optimize performance.

The Center of it all

Here is where it happens

The definition of the application is all of the dynamic content templates plus the executable process that is generating the pages and passing the compiled html pages to the webserver. A clear definition of what the project defines as the application has to be established. Large B2B and B2C sites, whether the executable is brokering a price auction or checking the database for the current product order, use an application for the transaction. Home Depot, American Airlines and Wal-Mart are just a few of many that use middleware products like BroadVision to generate dynamic content and facilitate revenue sales.

The term application can have multiple meanings. Break the application into two supportable parts. The first is the integration of the product with HP-UX. The second is the templates and proprietary development platform provided by the middleware product.

Installation of the base enterprise middleware product, commonly called the enterprise sever[2], with the operating system must be confirmed before development or support of the site can begin.

[2] For example: BroadVision One to One Enterprise Server, BEA Enterprise Server

Support concerns and strategies:

1) Middleware products, such as BroadVision and BEA, come with sample applications that act as proof of concept. Make sure that these sample products are installed, and the concept is proven before moving forward.
2) Once the site is up and running with the enterprise server and the application, use proof of concept applications to aid in fault isolation. If the base sample application works on the system, then the issue is with the application and templates. CPU utilization is the lead to the many issues.
3) Send a kill –6 to each of the middleware processes in development to examine the core. This will aid in understanding application stack traces. The stack trace can help find errors in developed templates or custom components relating to the middleware product.

Understand how to use and gather information of the application server. Tools like top, vmstat and glance will display the CPU utilization of the middleware processes. There is a process with BroadVision called the Interaction Manager. In the process table it is displayed as "bvsmgr". Bvsmgr can be blamed or exonerated once a site becomes symptomatic by using top. Top will display if the bvsmgr process is "hogging" CPU. If it is not, then the problem may lie in another bottleneck.

Support concerns and strategies:

1) Use top, vmstat and glance.
2) Understand the process that is processing the web-transactions. For this example, bvsmgr is a key piece in the BroadVision mechanism.
3) Confirm that the middleware is correctly integrated into the HP-UX operating system. As always, confirm patch levels and follow the correct kernel guidelines.

Performance degradation analysis should start with the application server. As mentioned before, TOP, vmstat and glance. If the middleware processes seem in order, troubleshoot the templates that have been developed for the site. For example, if there has been a new user login page installed in the JavaScript template repository and the site is now slow on user look up and login, this is where to start. One long function that lengthens user lookup to the database may have more than an additive effect on performance. If the user login page is slow, other user requests will be queuing up behind it, causing a bottleneck. The bottleneck is not with the user

profile lookup in the database but the execution of the lookup due to the length of the JavaScript function.

Template bottlenecks can be discouraging. Added functionality that did not make the timeline limits is the suspect. New functionality may break the logic of the application and cause slow performance across the whole application. Investing and maintaining an accurate staging area will help eliminate "prime time" errors. Staging is discussed later.

A helpful reminder is always nice

A short review

Start with supporting the idea of the transaction. Remember that the webserver gets the request, and through the defined Meta tag calls a URL that activates the middleware application. www.quickthought.net/coffee/home.jsp.

The following is another short example of profiling and dynamic content using www.quickthought.net.

Starting with the customer's first visit. The user clicks on the register new icon and a profile questionnaire appears. This is going to ask many questions that will be placed in the user profile database. These questions will be evaluated against a set of rules to see which demographic target quickthought.net would like to advertise to. The questions for this example questionnaire are; Name, average cups of coffee per day, caffeinated or decaffeinated. The user enters in the requested information and hits the submit key. The onSubmit action updates the database with the information and assigns a user ID to the profile.

The user leaves the site and comes back an hour later. There is an icon that says "Registered Users Login!" The user keys in the user name and password and hits the "Login!" button. The onLogin action requests the user information from the database and is displayed back to the user a web site dynamically generated for this user. For example, the web site will display "Welcome Back! "$USER" we have a special on super caffeinated coffee for you to brew with our special caffeinated water, $10.99". This deal seams advantages to the user so he hits "buy now". This onAction of "buy now" contacts the payment server so quickthought.net can collect the revenue for the transaction.

The simple transaction in the example can and will be very difficult to support. Here is a checklist.

1) Understand the transaction. Understand what exact host(s) the transactions are going to.
2) Have a default commerce site ready to put up. A scaled down back up site or a complete site mirrored to a different location will ease the solution expectations.
3) Don't be taken by surprise with user traffic. Understand the busy times. An excellent tool for simple monitoring of user traffic is MRTG. MRTG graphs the user traffic to the site with the correlating time.
4) Have an Intranet admin site. This site should be querying each part of the environment. To monitor the performance. A small admin site to query the daemon process should be placed for site monitoring. For example, ping times or performance tools built into the middleware packages. Or, there are advanced monitoring with HP OpenView or Tivoli.
5) Have contingency and routing plans. For example if a network route goes down, have a way to activate another one. It can be as simple as route add delete 1.1.1.1, route add 1.1.2.1.

Check the units of measurement

"Accuracy and precision is not the same, be precise!"—Dr. Natalie Mayer

To start benchmarking the site, start with the application webpages that will be pushed to the webserver. For example, the home.jsp may have little dynamic content, but the underlying entry into the portal application may be large. The second is to use netperf, ttcp, or timeit to measure the network latency between system. The third is to use applications like Homesite by Allaire to weigh the connection time of the developed pages, and finally use Purify to clean any C++ components or adapters.

Simple, simple, simple logic in supporting the application may seem too obvious, but missed.

In benchmarking, CPU usage is a good place to start. Check the systems out by running a series of top or glance to check CPU usage. If the application is using too much CPU at 5,000 concurrent users, the site will have major problems. If the site is at 10 percent CPU with 20,000 users, you may be able to add that functionality that was taken out due to application size.

Some large sites have used customized scripts to simulate web site navigation. Remembering that your web site is one large transaction that goes from system to system is hard to benchmark. There are problems such as memory leaks, icmp redirects in the routing table, misconfigured gateways, long SQL queries, and others that will not manifest itself until the site is loaded with users.

Network bottleneck, misconfiguration, and tuning may not manifest itself until there is heavy traffic load. For analysis and design information please refer to Chapter 11 Managing NFS/NIS by Hal Stern. Follow the methodology for benchmarking NFS traffic. The findings will be extremely relevant.

Practice makes perfect

"Practice has given me confidence in my own ability"— Tiger Woods

The concept of staging the site is paramount. A staged site will allow the developer to replicate, patch, develop, change the site, and test performance without porting it to production. Staging environment also allows for the rapid application development team to incorporate failover mechanisms to fail the site to the staging area if the main site goes down. This can be hard and complicated to do. If the site consists of 25 HP N-class servers in a three-four-tier environment, the staging area could be three systems. It is common in the staging environment for the webserver and the application server to run on the same system, the database on the other and the last acting as the customer code build system. If hardware allows, take a small percentage of the systems and create a fail over site. This will ensure that if you are doing maintenance at 5am EST an insomniac plagued customer can buy coffee from quickthought.net at 2am PST.

The build system may not always be the same exact platform as the production. For example the HP-UX N-class servers can be running the application, but the C++ components will be compiled on a 32bit HP-UX K-class server.

Components written as adapters into the middleware platforms must be run through Purify. Code purity will ensure the reductions of memory leaks. The staging environment is a good last place or the code to run in a purified state then stresses with load simulation. The

simulation will test the integration of the component the middleware, and the operating system. An example of an adapter is a shipping handler. A brick and mortar company that established a web presence may create an adapter or plugin into a legacy payment handler to keep the finance revenue stream the same.

Performance tuning is constant and never fully completed. The site will change with new code and functionality. The site is never a completed product, thus making support a fun.

The End

"This is the end of the world as we know it and I feel fine" – Musical Group "R.E.M."

The key place for web support does not always have to be involved in the project planning phases to be successful. The E-Services Support Center here at Hewlett-Packard rarely receives inside information to the design or is allowed to voice an opinion. B2C and B2B support was defined by a coworker as "Who cares what happened, just fix it; whatever it takes". There are so many moving parts and integration nuances to be perfect support engineers. With the support concerns and strategies proved above, maybe sites would run just a little bit smoother than before.