



***Report to***

**Department of Education**

**FAFSA Web Performance Report**

July 7, 2000

**Submitted by**

Beacon Technologies, Inc.  
420-B Gallimore Dairy Road  
Greensboro, NC 27409  
Phone: (336) 931-1295  
Fax: (336) 931-1296  
Internet: [www.beacontec.com](http://www.beacontec.com)

## *Executive Summary*

The focus of Phase 1 of the Mad Dog project was to evaluate the performance of the FOTW application and determine the factors that caused the performance issues during the week of February 27<sup>th</sup>, 2000. The analysis of the network definitely identified a network bottleneck on outbound traffic destined for the end users. The current infrastructure also represented that there was a high CPU utilization level on the Cisco 3600 router that is currently servicing the 4 T-1 lines. The bottleneck has been caused by the extremely high utilization (35 million hits during the peak week) coupled with a large file size (up to 118k) being transmitted back to the users' browsers. The majority of the data being passed to the end users is [JavaScript](#), which is used to perform necessary edit checks on the required fields of the form. NCS stated to us that they perform these checks on the client side to reduce the load on the server per one of the requirements from the Department of Education.

The reason why the students have had so many issues around losing data is that they are not allowed to save this data to the CPS database until the very last step of the process. No data can be [stored](#) on the server until the student reads and fully understands the implications of submitting the application. [NCS](#) stated to us that they could not save the data to their servers primarily because of the US Privacy Act. They stated that this act prevented the use of certain technologies such as session variables and cookies because the data needed to be protected on the client side as well.

The team strongly recommends implementing the major infrastructure enhancements at the VDC. The network capacity should be enhanced to support the application in its current state. Additional rental servers should be added to handle the increasing demand. Also, performing upgrades to existing machines is required to ensure that the application performance is acceptable. The application should be reengineered to take advantage of newer technologies if the privacy hurdles can be overcome.

The links to the specific areas covered in detail follow:

- [Table of overall findings](#)
- [Network Findings](#)
- [UNIX and Web Server Findings](#)
- [Database Findings](#)
- [Application Findings](#)
- [Additional Considerations](#)

## Summary Table

<u>Area of Concern</u>	<u>Findings</u>	<u>Recommendations</u>
Network	<ul style="list-style-type: none"> <li>• An outbound bottleneck on all the circuits.</li> <li>• High CPU utilization on the 3600 Cisco router servicing those circuits.</li> <li>• Token Ring network is older and slower.</li> <li>• Ethernet is only 10mbs and is running at half duplex.</li> <li>• Firewall configuration is using the “N+1” methodology to facilitate DMZ zones.</li> <li>• Round Robin load balancing technique.</li> </ul>	<p><b><u>Immediate</u></b></p> <ul style="list-style-type: none"> <li>• Increase network capacity</li> </ul> <p><b><u>Future</u></b></p> <ul style="list-style-type: none"> <li>• Remove all token-ring segments</li> <li>• Utilize 100mbs Fast Ethernet connections for interconnectivity between the DMZ and the internal segments</li> <li>• Reengineer firewall configuration using current day standards to implement DMZ zones</li> <li>• Switch to fractional T-3 circuits and Cisco 7000 series routers</li> <li>• Implement a more robust load balancing solution that uses algorithms to distribute the load</li> </ul>
Unix and Web Server	<ul style="list-style-type: none"> <li>• High utilization during peak week only</li> <li>• Some servers were using slower processors</li> <li>• Restore/Save function uses only one machine</li> <li>• Default Unix tunable parameters are in use</li> <li>• Default Web tunable parameters are in use</li> </ul>	<ul style="list-style-type: none"> <li>• Increase network capacity to reduce bottleneck</li> <li>• Correct hardware deficiencies on some of the servers to increase performance</li> <li>• Modify the restore/save function to use more servers to balance the load</li> <li>• Add more rental servers to handle load prior to the VDC move</li> <li>• Modify Unix and Web tunable parameters to handle increased load from network capacity increase</li> </ul> <p>Note: Important performance data on Unix servers was not available during the peak on the FAFSA Web Servers</p>
Database	<ul style="list-style-type: none"> <li>• The PIN system runs on a slow, 1 processor machine</li> <li>• Database designs are simple and suit the application</li> <li>• Data servers run a large amount of batch processing</li> <li>• Network bottleneck shielded the databases from performance problems</li> </ul>	<ul style="list-style-type: none"> <li>• The PIN system needs a faster box with more processors</li> <li>• The CPS database may need to be redesigned if FOTW is changed to allow more data saving opportunities</li> <li>• More credence needs to be given to the performance of the mainframe</li> <li>• An environment to perform load testing is needed</li> </ul>
Applications	<ul style="list-style-type: none"> <li>• Several disadvantages to using CGI</li> <li>• Hidden fields are being used to store date, which consequently increases file</li> </ul>	<ul style="list-style-type: none"> <li>• Setup should be modified to utilize CGI threading</li> <li>• Use of hidden fields should be eliminated and replaced with the use of the database, cookies, or session</li> </ul>

---

	<p>size</p> <ul style="list-style-type: none"><li>• JavaScript is being heavily used throughout the site</li><li>• No load testing was done prior to production</li><li>• No database connection pooling was used</li><li>• System is set up to reference a gotofile executable</li></ul>	<p>variables.</p> <ul style="list-style-type: none"><li>• Use of JavaScript needs to be minimized in order to reduce amount of data sent out on problem line</li><li>• Provide a test environment similar to production, a load testing tool, and a load testing team</li><li>• FOTW should be modified to use pooling</li><li>• Data and memory should be shared to improve performance</li></ul>
--	---	--

## *Network Findings*

One of the requirements of the Maddog initiative was to verify and document the network infrastructure. Since this area is an obvious cause for concern, the team had several meetings and tours to ensure that the team from Beacon Technologies, Inc. understood the overall network infrastructure.

The NCS data center ISP is Cable and Wireless, which was formed from an MCI spin off. They provide 4 dedicated T-1's to support the FOTW application. These circuits are banded together to form a 6mb circuit. The NCS router/server/firewall network configuration is supported out of a single dedicated Xylan Omni Switch switching platform. The switch supports both token-ring and Ethernet attached devices together in two VLAN based network segments that are interconnected via RS/6000 hosted Portus firewalls. The switch provides Mac layer translation between the token-ring and the Ethernet devices. The data flows to/from the internal FAFSA-OTW segment through a 16mb token-ring connection in the NSC 1000 router. The NSC 1000 routes data bound to/from the VDC out through a 16mb token-ring connection to a VDC supported 3Com token-ring hub and to the VDC's Cisco 3600 series router. There are 8 T-1's in place between the facilities to support all of the traffic from Iowa City to the VDC.

The graphs of data taken from the week in question indicated the following:

- An outbound bottleneck on all the circuits
- High CPU utilization on the 3600 Cisco router servicing those circuits

The following findings are areas of concern:

- Token Ring network is older and slower
- Ethernet is only 10mbs and is running at half duplex
- Firewall configuration is using the "N+1" methodology to facilitate DMZ zones
- Round Robin load balancing technique

The speed of the current infrastructure needs to be upgraded to improve data transmission between the servers. Also, because of the current firewall configuration, someone using the pin server (i.e. executing a renewal online) has to go through three firewall connections before they can start the process. This provides unnecessary overhead since Firewalls protect data but add latency to investigate the packets. Also, when the FOTW application is located in the VDC, it would then only need to go through one firewall to get to the CPS database. The round robin server provides load sharing, but not true load balancing, since it does not check the current state of the machine that it is passing a request to. The other issue that is of concern is that the end users can bookmark a URL on a particular server. This enables users to go around the round robin server to a server that may be already overburdened. Also, the current save feature only

goes back to one web server, causing additional load on that server, and that needs to be corrected.

## ***Recommendations***

Since the entire application is going to be relocated to the VDC in Meridan Connecticut in the July time frame, our recommendations are divided into the following categories:

### ***Immediate***

The network capacity needs to be immediately increased to alleviate the outbound bottleneck that has been identified. This process has been initiated by NCS already and the plan is to have another 4 T-1's banded together into a separate Cisco 3600 router, thus doubling the current bandwidth to 12mb. The network team plans on balancing the outbound traffic on these routers by altering the default gateways on all servers, accordingly balancing the number of servers per router. They will proactively monitor performance and route the traffic appropriately.

This should reduce the load on the current router and address the current problem with the network load. We agree with this approach and this move will certainly help the current situation. This would be the best short-term fix with the impending migration to the VDC.

### ***Future***

The network infrastructure at the VDC should include the following enhancements:

- Remove all token-ring segments
- Utilize 100mbs Fast Ethernet connections for interconnectivity between the DMZ and the internal segments
- Reengineer firewall configuration using current day standards to implement DMZ zones
- Switch to fractional T-3 circuits and Cisco 7000 series routers
- Implement a more robust load balancing solution that uses algorithms to distribute the load

These recommendations should be strongly considered and evaluated before the application is moved to the VDC. Consolidating the network to the fast Ethernet standard would enhance the performance by removing the Mac layer translations between the token-ring segments and the Ethernet segments. The speed would be approximately 10x faster than the current infrastructure. The proper alignment of servers into "service/dmz" zones would reduce any additional latency issues. The firewalls should be configured for redundancy in the event of a failure.

The T-3 circuits will provide greater bandwidth as well as provide the ability to increase/decrease bandwidth as needed. The pipe could be increased during peak times of the year and reduced during others to help handle performance and control costs. The installation of a load balancing solution that will distribute load based on performance criteria and shield the web servers from direct user access is necessary. This would help improve performance and allow for maintenance to be performed on the servers simply by temporarily removing them from the group.

## ***UNIX and Web Server Findings***

It was difficult to determine from the log files what occurred during the time period including February 27, 2000 and March 2, 2000 because the system log files are stored in a 14 day rolling window and the files containing the raw data had already been purged. Except for the PIN web sever (*eacws1*), only summary performance data was available for the week of February 27.

Each FOTW web server has four virtual web servers running on it: two for the 1999/2000 school year and two for the 2000/2001 school year. One of each set is a non-secure site listening on port 80 and the other is the secure site listening on port 443.

All web servers were configured with default AIX and Netscape tunable parameters.

Default Netscape web server tunable parameters are used and appear to be adequate to support the current system load. During the time period under investigation, the RqThrottle parameter was altered from 512 to 128 sessions on *fafsaws3* and *fafsaws4* to lighten the load on those servers. Those two systems were delivered to NCS by the vendor with slower than ordered CPUs. The slower CPUs lowered the expected work capacity. Lowering RqThrottle causes Netscape Server to refuse to service more than 128 sessions. Lowering the parameter had negligible impact on overall performance.

*Eacws1* is a four CPU server supporting the PIN authentication application. RqThrottle was lowered from 512 to 128 sessions for the same reason as above. Lowering its parameter also had negligible impact on overall performance.

The performance counters available for *eacws1* seem to indicate that during the period of highest stress, the system became CPU bound.

While the system was under high stress, the number of runnable processes increased from between zero and one to 18 to 19 per second. This indicates a CPU bound system. Also, during this time, the system call activity increased from a normal several thousand up to almost 19,000 per second. Context switches increase from low hundreds to low tens of thousands. High context switches and high run queue values would be consistent with a CPU bound system. The summary performance charts indicate that when the PIN web server became CPU bound, the PIN database server CPU time dropped dramatically. It seemed that the web server stopped asking the database server for information.

Unfortunately, without process accounting information, it is impossible to tell what processes were running during the critical time. Process accounting and other non-critical processes are turned off to minimize negative performance impacts.

Of the five web servers in service during the week of February 27 through March 4, three peaked at 100 percent CPU utilization and two peaked at between 68 percent and 70 percent.

Of the three that peaked at 100 percent, one, *fafsaws1*, is the system used to restore a previously saved FAFSA edit session. Since it is the sole system used to restore the data, higher system use would be expected. Even though it peaked at 100 percent, the average utilization on the system during that week was 57 percent.

Both of the other two systems which peaked at 100 percent, *fafsaws3* and *fafsaws4*, were the previously mentioned systems received with slower than ordered processors. Since the load balancing algorithm was round-robin rather than based on system availability and performance, they were overloaded compared to the other systems in the group. Even so, their average system utilization was not over 68 percent.

The two servers that seemed the least stressed were *fafsaws2* and *fafsaws5*. The peak utilization was 70 percent and the average utilization was less than 38 percent.

Even though *fafsaws3* and *fafsaws4* will be upgraded to the correct specifications, and additional servers will be added to the server pool, it will be critical to monitor performance during the next peak period. In addition, it is critical that load testing be done so total site capacity can be determined before the next high demand period and corrective action taken during testing to improve performance.

With increased capacity in the network, it is likely that more people than before will be able access the site. Removing the network bottleneck, upgrading the systems and adding more servers may completely fix the problem, or it may only move the bottleneck someplace else.

### ***Suggestions for improvements in the future***

Given that the outbound network congestion problem will be solved, either by increasing network capacity or decreasing outbound data size, or both, additional tuning is likely to be needed on the UNIX servers to allow them to keep up with the increased traffic.

#### **Items to investigate include:**

- Netscape MaxProcs parameter—increasing this to the number of CPUs should spread the web server workload among the CPUs. While adjusting the parameter, monitor the process run queue to ensure that the number of runnable processes stays within acceptable levels.
- Netscape RqThrottle parameter—this should be adjusted to limit simultaneous connections to the web server. A decision needs to be made: is it better to have a long wait and possible browser timeout or an immediate "site is busy"

notification?

- Netscape MinCGIStubs, MaxCGIStubs and CGIStubIdleTimeout parameters—these should be set to pre-spawn a minimum number of each CGI program and limit the maximum allowed. These parameters work together to control how busy the servers will get running the applications offered by the web server.
- Operating System tunable parameters may have to be altered to support the higher load that might be imposed by the additional server daemons and CGI programs.
- Consider using shared memory to hold static information that is read frequently. This type of data could include the school code lookup table and the gotofile data. Holding this information in shared memory would eliminate the overhead of a file open/file read/file close or a database connect/read/disconnect for school codes for an application renewal or application correction. Instead of the overhead of file or database access, a CGI program would attach to the shared memory segment, read the data, disconnect from the shared memory, and continue. Refreshing the data can be handled by a daemon that updates the data on an appropriate schedule or on demand. Careful synchronization between the daemon and the CGI programs needs to be maintained to prevent reading data during its refresh. A semaphore system would provide the most reliable way to provide a locking mechanism.

***Note:***

All the above suggestions require careful design and implementation planning and compromises in performance and resources like CPU cycles and memory. An environment where stress testing can be performed is highly recommended.

## *Database Findings*

The “FAFSA on the Web” (FOTW) application has two separate areas where its database processing is performed. The first, the PIN authentication system, uses an Oracle relational database management system (RDBMS) running on an IBM RS/6000 with one processor (112 MHz). The other, the Central Processing System (CPS), makes use of a DB2 RDBMS that is housed on an IBM mainframe running OS/390 (V1.3.0). The logical and physical database designs of these two areas were considered during the evaluation.

### **The PIN Database**

The underlying database design of the PIN authentication system is very simple. It has one large, twenty-partition table that has approximately 15 million rows in it. There is one other smaller table consisting of about 50,000 rows. All queries that use these tables were provided and analyzed. The online accesses and updates of these two tables are done without joins and always use the primary keys. The query plans and I/O statistics of the queries suggest that these primary key accesses are very quick and efficient. Any locks held on these tables by the update queries will be short lived.

An evaluation of the Oracle tunable parameters and reports indicate that the Oracle instance is operating normally. These reports do not show any performance problems that are the result of I/O latency. Also, the PIN application has exclusive use of the data server where it resides.

There are 3 batch jobs that run daily on the PIN database. These jobs run in the off hours for a total of about 6 hours per day and are very CPU intensive. They pose a potential problem if the database server is ever stressed. Also, one of the jobs, a bulk import, will hold temporary locks on the tables as it inserts data. These inserts are done one at a time, however, which will give the online application access to the tables when necessary.

The CPU statistics from the week in question (Feb. 28 – Mar. 3) indicate that the database server was not stressed. Also, the NCS Oracle database administrator (DBA) stated that the Oracle instance was readily available during this time. The data gathered implies that the “45 minute wait just to enter a PIN to get started” mentioned in the NYU email was not database related.

### **PIN Database Recommendations**

The apparent network bottleneck in FOTW probably shielded the PIN database infrastructure from any performance problems. However, when the bottleneck is resolved, this data server will be put to the test. The thousands of simultaneous user sessions and database queries, along with the required batch processing, will definitely be

too much for this one processor (112 MHz) machine to handle efficiently. This machine should be upgraded to a two or four processor box. Faster processors should also be considered. The random access to the two tables implies that this system will always have to perform a fairly high amount of physical disk I/O. It will be necessary to monitor the I/O speed closely as the use of this database increases.

While there are certainly other possible table designs to try, this design is so straightforward that it is doubtful that any table changes would yield significant performance improvements.

### **The CPS Database**

All FOTW database processing, other than PIN authentication, occurs on the CPS. This includes “Renewals on the Web” (ROTW), duplicate SAR requests, etc. While the CPS database design is more complicated than that of the PIN database, it is still a very simple design that contains no complex relationships between the tables. The main tables are made up of about 4 million rows each with 200+ columns. Based on the queries that were analyzed, all accesses and updates of these tables via the Web application seem to be done using the primary keys. The largest joins seen were 2-table, identifying relationship equi-joins. This means that the joins were on the primary key and that the key was always provided. These queries are coded to be quick and efficient. Any table locking caused by the update queries should be minimal.

There are many batch jobs, such as the “COMPUTE” process, that run on the CPS mainframe throughout each day. The total duration of these jobs averages between 10 and 12 hours daily. From strictly a logical database perspective, these batch jobs and their underlying tables are designed and coded to be mostly non-intrusive. For the most part, they will hold no locks on tables that are accessed by FOTW. The locks that are held are short lived, so they will allow FOTW equal opportunity to read the data.

There are not many statistics for the mainframe available from the peak week. However, the DB2 DBA stated that the error log on the mainframe did not contain any alerts about CPU or I/O utilization that were cause for concern.

### **CPS Database Recommendations**

The sheer number of columns in the main CPS tables seem excessive from a database design point of view. Extreme row sizes can sometimes cause more physical I/O than there would be in more denormalized designs. However, discussions with the key CPS personnel did indicate that other potential table structures were considered. Also, the tendency of the application to collect all relevant data before saving to the database lends credibility to the current design. If the application is changed to save to the database more often, it may be advantageous to revisit the table structures.

There needs to be more attention given to the mainframe piece of the FOTW application. The general feeling from NCS and the Department of Education is that the

mainframe can handle any load that is put on it. The DB2 DBA stated that he was not asked to look at the mainframe while the performance problems were occurring. While the mainframe was probably not the bottleneck during the peak week, its CPU statistics, I/O statistics, and ability to grant a logon request should not be ignored during future performance considerations.

### **Non-Platform Specific Database Recommendations**

Databases with row counts of this magnitude need to have a stress test done before they are released into production. To perform a reliable test, the IS staff needs to have a test area that is at least similar in size and distribution of data to the production area. Currently, such a test area does not exist for either the PIN system or the CPS.

## *Application Findings*

FAFSA on the Web (FOTW) was written using C++ Common Gateway Interface (CGI), JavaScript, and HTML. CGI handles server-side processing such as database access and complex edit checking while HTML and JavaScript are used on the client to display the application form and handle minor edit checking. The FAFSA process was designed to store user data in HTML hidden fields. As a result, no newly entered data is actually stored in the database until a completed form is submitted. The data is stored in a DB2 database, which is part of the Central Processing System (CPS). An Oracle PIN database is used for user authentication, which is required for FAFSA Renewals, Corrections, and Electronic Signatures.

The C++ CGI programs used throughout the site generate the HTML being sent to the user and process HTML forms. The FOTW web servers on which the CGIs reside are not running server extensions or any other software package that converts the CGI processes to threads. In addition, no settings on the server have been customized to limit the number of CGI processes running at one time.

Hidden fields and JavaScript make up a majority of the HTML being sent to the FOTW client. JavaScript typically makes up 2/3 of the HTML. The HTML is being sent on the outbound T-1 that was flooded during the week of February 27<sup>th</sup>. Hidden variables are being used to store data due to restrictions placed on development from the US Privacy Act. Large amounts of JavaScript were used to handle edit checking as a result of a restriction defined by the Department of Education to limit server processing. The maximum HTML file being sent to the user is 118kb and the average file size is 90kb.

No load testing was performed on the FOTW web site. A limited test environment is currently available. No load testing tool has been selected by the test team.

Database connection pooling is not being used on FOTW.

One of FAFSA's heaviest hit programs is reading a file each time that it is referenced. The program reads data from a servers.txt file, which contains data that does not change frequently.

## *Recommendations*

As mentioned above, the existing FAFSA application is written using C++ Common Gateway Interface (CGI) programs to handle server side processing. Although CGI is used on most of today's web servers, there are many disadvantages to using it. A main disadvantage is the fact that it creates a process on the web server for every CGI request. As a result, the server must create and initialize a new address space for every

process, which typically leads to server performance issues. In addition, most web servers are limited to running a set number of CGI processes at one time, which means that it is possible for the server to run out of processes. Many servers offer plug-in server extensions that can be used with CGI programs to make them more efficient. The plug-ins extend web server functionality and assist with the performance because they create a thread for each program call instead of a process. Unfortunately, the web server extensions are typically proprietary and linked to a single platform. FOTW uses CGI without server extensions, which could be causing performance problems.

Unfortunately, due to the limited web server statistics available for the week of February 27<sup>th</sup>, the impact of CGI on FOTW is unknown. Due to existing performance problems and to avoid potential bottlenecks in the future, the FAFSA application should be written in a language that uses threading or, if possible, the AIX server should run some type of plug-in that forces the use of threading.

The existing FAFSA application uses HTML hidden fields to pass user information throughout the site. The hidden fields and their associated values are part of the HTML, which means that the hidden data is sent to the client along the outbound T-1. The outbound line proved to be saturated during FOTW's busy week; therefore, FAFSA should probably be modified to limit the amount of data being passed along the line. In addition, the more data that has to be passed to the client, the more it slows down the load speed of each page, especially for users with slower connection speeds. Databases, cookies, or session variables are typically used to store data entered by a user. Any of these technologies would decrease the amount of data being passed along the problem line, which would improve performance. NCS explained that they were forced to use hidden variables instead of databases, cookies, or session variables. They stated that cookies were not an option due to restrictions placed upon users at the facilities where the application was getting accessed, such as labs and libraries. Session variables and database tables were not options due to restrictions placed by the US Privacy Act. Hidden variables were the method of choice. In addition to assisting with performance along the outbound line, the use of another technology would offer alternative methods for save/restore rather than the user manually saving and restoring to his or her PC. Currently, if a user loses a connection unexpectedly then he/she loses all data entered if the data had not been saved to the PC prior to the loss of connection.

A large amount of edit checking on the FAFSA site is done using client side JavaScript. Unfortunately, client side JavaScript, as well as hidden variables described above, is always sent to the client as part of an HTML file. The average FAFSA file that is being sent to the user averages close to 90 kb. The maximum file size is 118 kb. The outbound T-1's that control the data being sent to the end users became flooded during the week of February 27<sup>th</sup>, which suggests that the amount of data being sent through the outbound T-1's needs to be minimized. Decreasing the size of JavaScript and HTML code would clearly improve performance. This can be accomplished by moving JavaScript edit checking to a CGI that resides on the server. Since each of the FAFSA HTML files currently being sent to the end user is typically made up of 2/3 JavaScript and 1/3 HTML, the removal of JavaScript would make a significant difference in the amount of data being sent through the outbound T-1's. NCS stated that one of the

reasons that JavaScript is used for edit checking is to avoid an increase in server utilization; therefore, it is important to note that a decrease in the use of JavaScript would require more processing on the server. Since utilization would increase, the threading mentioned above almost becomes a necessity to ensure reasonable performance. If a minimal amount of JavaScript is required on the site, then using single character vars/functions and removing comments could minimize JavaScript code. Setting page size limits, minimizing the size and number of images used, removing unnecessary tags, and removing extra spaces/tabs/quotes can decrease HTML file size. Following the development techniques mentioned above would clearly reduce the volume of data being passed along the outbound line that would ultimately improve FOTW performance. In addition, the amount of data being sent along the outbound line would decrease, therefore allowing the data to be passed faster to students using low end machines. Removing the use of JavaScript would also eliminate the restriction for users to have their browser JavaScript option enabled before running the site.

Optimal performance is critical for highly used web sites such as FAFSA. Load testing is a function that can help to ensure customers of optimal performance and scalability. Load testing provides a method for simulating a large user load, which offers extra opportunity for performance tuning before a site is moved to production. Unfortunately, there was no load testing done on the FAFSA site; therefore, limited opportunity was available for uncovering load problems before the site was moved to production. One of the reasons that the project plan did not include load testing was because of the limited test environment currently available. To assist with performance tuning and to avoid performance problems such as the ones that arose during February/March, the project needs a test environment similar to production, a load testing tool, and a load testing team.

Database Connection Pooling is necessary for highly used web sites that connect to a database server. Pooling is typically used to increase a site's performance. Connecting to a database from a web server can be time-consuming, depending on the speed of the network and the location of the database server. Without connection pooling, a process must connect and disconnect from the database for every database call. With pooling, a pool of connections is built as a web application opens connections to a database. As the application submits connection requests, if one is available in the pool then an existing connection is given to the application, therefore eliminating the need to create a connection for every request. FAFSA does not currently use database connection pooling. Since pooling is typically used on sites to improve performance, FOTW should probably be modified to use pooling, especially since the database server resides in a different location than the web server. This may not be an option to NCS due to the limitations of CGI; therefore, a re-write may be necessary to take advantage of this technique.

Currently the system is set up to reference a gotofile executable, which is used to read a list of servers and redirect the user to a secure or un-secure site. Gotofile is one of the heaviest hit programs on the FAFSA site. The program reads a file that contains a list of servers, then redirects the user based on parameters passed to the program. Since

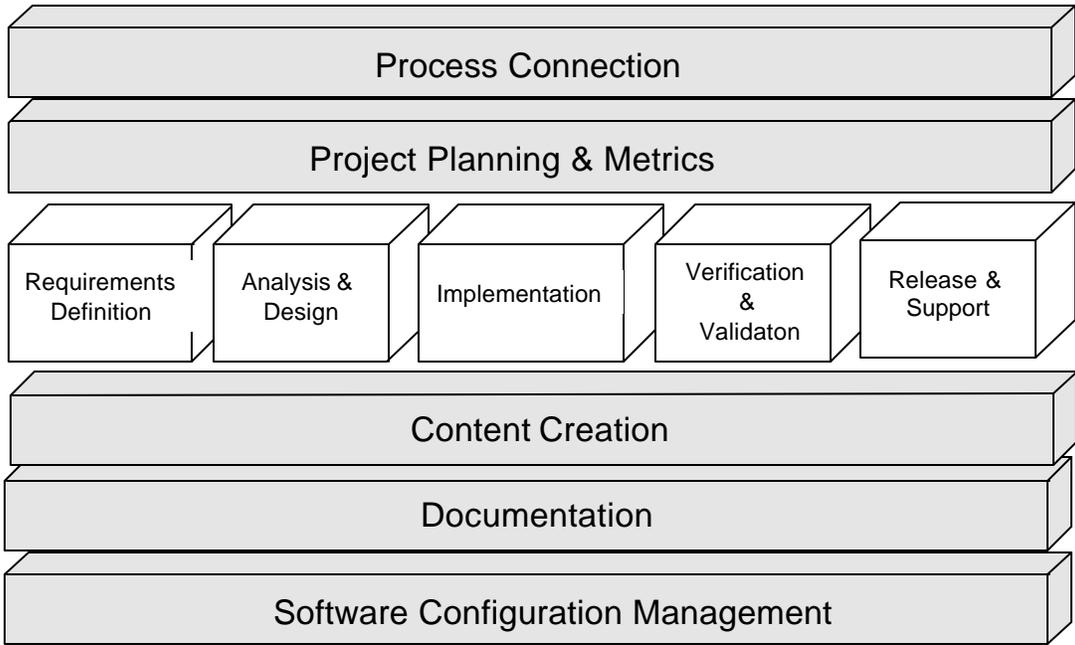
gotofile is one of the heaviest hit processes on the site, the number of file reads is extremely high. The application should be modified to read the server input file one time, at the start of an application, then store server values in shared memory, making them accessible to all processes while minimizing the number of file reads. This change to shared memory would eliminate a significant number of file reads occurring on the server, therefore improving system performance.

# *Additional Considerations*

## **Development Cycle Impact**

While the web is an excellent environment for rapid prototyping, it is not a silver bullet. Business applications moved into or developed for the web space still require old-fashioned planning, testing and tuning.

As the following graphic illustrates, the types of development tools required for web applications are the same as for traditional environments.



Note:

NCS has a development process that they follow; however, until recently they did not have a test environment setup. Also, performance testing was not performed on FOTW before going live into production. Performance testing needs to be incorporated into this process.