Nathan A Hanna. A Case Study in System Migration to Linux in a Development and Production Environment. A Master's Paper for the M.S. in I.S. degree. April, 2004. 35 pages. Advisor: Scott Adams

This case study describes the process the planning, the implementation, the testing, the migration, and the evaluation that took place in the migration of the main departmental server for the School of Information and Library Science at the University of North Carolina at Chapel Hill. This system is responsible for departmental mail and web services, as well as functioning as a developmental and research system for the faculty, staff, and students within the school's programs.

The paper discusses the process of migration, from evaluating the new architecture to the final testing and modification following the implementation of the new system. The migration was from Sun Solaris on Sparc hardware to Linux running on Intel. The migration migrated 86 Gb worth of data and 800 users with at most 25 minutes of downtime for the critical applications.

Headings:

Systems migration – Management – Case studies

Linux

Computer Systems - Management

A CASE STUDY IN SYSTEM MIGRATION TO LINUX IN A DEVELOPMENT AND PRODUCTION ENVIRONMENT

by
Nathan A Hanna

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2004

Approved by

_____

Scott Adams

1 Introduction

There comes a time with every computing system that upgrading becomes necessary. When completing these upgrades there are many things that must be done in order for the new system to successfully replace the preexisting system. This case study is based on the planning, the implementation, and the evaluation of the migration of a small, departmental server. This migration was within a mixed computing environment at the University of North Carolina at Chapel Hill within the School of Information and Library Science. The main usage of the system being migrated was as a web server and email server for the department. The system acts as both the production machine for serving departmental web services as well as a development machine for students and faculty making use of websites for instruction and research.

The original system, which was being replaced in this migration, was brought online July, 1998. This migration occurred over a week long period where the base operating system was installed, the user data was migrated, and finally the applications which the system was intended to serve were brought online. The migration in 1998 caused problems because of the length of time that was needed to get the new system online and fully functional. This is why one of the major goals of the present migration was to keep the downtime and perceivable problems for the users of the system to a minimum. There has also been a great increase in the computing needs and demands of the department; the new system would need to allow for this growth and future growth within the programs in the department.

Prior to the migration beginning, there was research done in order to determine the current best practices for UNIX system migrations. One of the major problems with this kind of research is that many of the groups doing leading work with migration to new systems do so as a service, and therefore it appears that they do not want to give much detail into the process. One of these providers is HP, who provides a general summary about options for replacing Solaris systems with Linux systems. The summary provides reasons for migration, analysis of general considerations for migration, and finally a sales pitch on why their services will benefit migration (HP Development Company, 2003).

The migration was done with the Director of Information Technology at the School of Information and Library Science. The system migration broke down into five phases. These phases took place over a three month period. The first phase was the planning phase where the goals of the migration and the process to be used were evaluated. The second phase was the implementation of the initial system setup. The third phase is the testing and configuration of the new system. The fourth phase is the migration from the original system to the new system. The fifth, and final phase, is the evaluation and modification of the system setup.

2 The Planning Phase

The planning phase occupied a large majority of the system migration and upgrade. The first part of planning was the evaluation of what was currently in place. The second component was the evaluation of the options for the new system. The final element was planning the configuration and setup for the new system.

2.1 Evaluation of present system

The original system was run on a Sun Enterprise 450. The operating system was Solaris 7. The disks in the system were configured using a software RAID known as DiskSuite. When the system was originally purchased there was not an option for a hardware RAID. For this reason there were many bottlenecks in throughput with the system being based on this software RAID. With this software RAID there was the demand on the system processors and resources for every disk write and read. A major benefit of the system upgrade was purchasing a system with hardware RAID, which would increase system performance significantly.

The hardware RAID provides specialized processors for dealing with and interacting disks in the RAID array. This provides the benefit of removing these demands from the main system resources as well as allowing more robust and dependable subsystems to support the RAID. The disks currently available were also far superior to those available when the original system was configured. These new disks provided high disk speed that allow larger data throughput while also providing faster seek and access speeds. A combination of replacing the software with specialized hardware as well as the improved base hardware would greatly improve the operation of the system.

An important issue was to determine the current usage of the system being replaced. There were the common issues, such as a running web server and an email server, but there where other services on the system on which people were dependant although the system was not initially designed for this. These services were the ones that needed to be determined and explored prior to migration. Contributing to the issue was that the system was both a production system for many of these services while also a development and research system for the faculty, staff, and students of the department. It

was important to evaluate these other uses. Many of these included issues such as the system being used to run custom research applications that were dependant on other applications. These included Wolfram *Mathematica* as well as being used as a system to connect to the *MySQL* database server which runs on another system in the network.

2.2 Evaluation of options

The next portion of the migration planning was to evaluate the options for the new system which would replace the original system. The two main options that were available were to keep the system running Solaris on Sun hardware or to migrate to a Linux system running on Intel based hardware. Based on hardware costs the decision was made to select the Intel option. This left the option of which Linux distribution to use for the system.

There were many different distributions from which to choose. At ibiblio.org over 50 different distributions are offered. The selection of the distribution was one of the most substantial issues concerning the new system. When evaluating the migration it was discussed that there would need to be the support and updates to the software so that the system would remain secure and updated for the lifetime of the machine. There was also the need to have a system that would be supported for a long enough period of time to protect from having to migrate due the end of life on the operating system support.

A key issue raised with the discussion of Linux, and open source software in general, is whether it is ready for production environments since there is not one company or resource to turn to when there are problems with the software. There is also the risk that if an open source project decides to stop updating its project software there is

the chance that a production, mission-critical application would be left unsupported, and vulnerable to security vulnerabilities.

On a production system there is a need to have a method for managing and monitoring the patch level of the various applications and system utilities installed on that system. As with any system, application patches are released daily. The vulnerabilities in any one of these packages can impact additional applications because they are interdependent. For example, if a patch is needed for the *OpenSSL* package it often becomes necessary to also reload the *SSH*, SSL web server, and *SSL* based *imap* and *pop*3 subsystems. Without an organized patch management system a systems administrator could spend all of their time monitoring and patching, which would prevent them from being able to complete the other tasks required of them.

In the department where the migration was taking place there was a need for a easy to manage and accurate patch management system. Redhat® Enterprise Linux provided this sort of patch management and reporting. The *up2date* utility is a utility provided by Redhat when a subscription to the software is purchased. The Redhat network provides notification when patches to the system are available. The *up2date* utility allows the administrator to selectively upgrade the patches as needed. The updates are made through the RPM Package Management utility known as *RPM*.

RPM is both the individual package as well as the system used for managing the status of the packages. It allows for the checking of the cross requirements before installing packages. This ensures that one application is not installed when there are conflicts with associated applications and/or libraries. For many of the applications and utilities on a system, these packages, with their default configurations need no changes

made to work with the system. Some of the *RPM* packages configuration files need to be slightly modified to customize the system to the optimal compatible configuration.

2.3 Evaluation of migration strategies

One of the last things that must be completed in the migration of a system is to migrate the data for the users and the system to the newer system. In some environments the data is stored separate from the systems that make use of it, but in this environment the data used by the system is stored locally on that system. There were a total of 86 gigabytes of data that were going to need to be moved from the original system to the new system. The goal of the migration was to find a way to migrate the data while maintaining the permissions and properties of the data. When evaluating the environment it was determined that the data would best be migrated over the network. Each system would be on a 100 Mbit network connection. The main files that needed to be moved were the personal folders of the users on the system, the documents in the /htdocs folder, which is the root for the web server, and finally there was the need to migrate the mail inbox folders for the system which were located in /var. .

The options available were to migrate the data using *scp* (secure copy protocol) streams to migrate the data, to ftp the data from one machine to the other, or finally the options of *rsyncing* the data from one folder to another. One of the major problems was that the *scp* protocol has a high overhead because of the inefficiencies in the protocol, which are related to the processing needed for securing the data connection and securing the data during transfer. *FTP* eliminates many of the problems such as the overhead from securing the connection, but there are problems with permissions on the migration of the data. The third option, which was the one that was finally selected, was the use of *rsync*.

Rsync is a tool that allows the synchronization of files on different systems. There is also an archive option, which maintains all time/date, permission, and file information for each file and folder that is migrated. The connection allows use of *rsh* and then makes use of its own transfer protocol to migrate the data between the systems as needed. *Rsync* was chosen based on the speed, the option to open multiple migration channels, and the ability to easily reorganize the data structures on the new machine while migrating. Tests were done with each of the methods for data migration, and *rsync* was by far the most user-friendly, fastest, and provided many options that were ideal for a data migration in this context.

2.4 Evaluation of opportunities presented by migration

The migration of the entire system presented the staff with a great opportunity to make additional changes to the organization of the system. One of the reasons for the upgrade and migration of the system was a problem with disk space on the original system. Since the original system was brought online there has been a steady increase in the number of users, as well as in the volume of data that individual users were storing on the system.

2.4.1 Data Organization

In order to complete the migration all of the data would need to be migrated. This presented an opportunity to evaluate the organization of the data on the system. With the majority of the information on the system being located in the user folders and the inability to add any additional disk space within the machine presented possible problems for the future. The potential of running out of disk space is always a threat and it seemed

very likely at the current rate of data growth that additional disk space might be needed prior to other hardware replacement.

On the original system the entire user file space was organized on a common home directory located on the large data partition (/export). At the time of migration it was decided to move to a more robust organization scheme. It was decided to organize the user space within the root home folder in a way that the if additional disk space were needed it would be possible to migrate portions of the data to a new disk structure but not have to hinder access to portions that did not need to be moved. It was decided that the easiest and most user friendly organizational method was to add a folder for the first letter of a username. Each user folder would then be stored in the relevant folder. In order to keep the users with an environment similar to the previous environment symbolic links were added so that /export/home/*/foo would appear to be the location of their files, where /export/home/foo was the previous location of their files. If at a later time additional disk mounts were added the data of particular user folder could be relocated to the new disk space and then the symbolic link would simply need to be changed to handle that move.

2.4.2 User Access

One of the largest concerns with the migration revolved around the ability to migrate user authentication method. Although changing the type of authentication was not intended to be changed, there was the fear that the newer version of the hash for password verification would be incompatible. With the largest functional password on the original system being eight characters there was the fear that it would be incompatible with the new system, which has the ability to store and verify much longer passwords. In

order to test this it would be necessary to make use of the same hash from the original *shadow* file in the new *shadow* file. The hash in the *shadow* file was functional on the new system, which was one of the major issues to allow the efficient migration of the system to occur with fewer problems for the users.

Another major issue that was dealt with in the migration was considering what types of access would be available. The original system provided access for login via *SSH*. *Telnet* had previously been disabled. File management was supported via *FTP* as well as via the *SFTP* and *SCP*. With the increasing need for security on systems the decision was made that access would be limited to secure methods whenever possible. This meant that *FTP* would need to be removed as an access option. This was done in an attempt to prevent passwords from being sent over unencrypted channels to the system. In evaluating the options to connect it was decided that the availability of *SFTP* and *SCP* clients was enough to warrant the change. Allowing secure access to the web server also meant there was a need to allow *SSL* based access via https. This would allow forms and information to be passed securely between client and server.

The other main method of user access is through email. The system supported both *pop3* and *imap* connections for sending and reading email. It was decided that secure *imap* and secure *pop3* would be provided on the new system following the migration. There was still the problem that not all email clients that are used to check mail on the system are able to operate with the secure versions of the protocols. For these reasons it was decided that the less secure versions of the applications would still need to be available. As a secure substitute there was also a web based email client included that

would allow for mail to be accessed via https. Evaluation of their availability will be continued at a later date as more email clients begin integrating secure mail protocols.

With each of the possible changes made to the system there was always a discussion of the acceptable level of security versus the acceptable level of usability. With this being a learning system, where students are allowed explore and try things, it was often necessary to error on the side of accessibility over security. Needing additional access for learning would not normally be an issue in a production system.

3 Implementation

The implementation phase was where the decisions made in planning were implemented into the live system setup. This included the installation of the operating system, the installation and configuration of critical applications, the installation of additional packages that are needed to make the system appear as much like the original system as possible, and to make improvements in the setup wherever possible.

3.1 Operating System Installation

The installation of Redhat Enterprise Linux has become very user friendly. It was decided that Redhat Enterprise Linux ES 3.0 would be installed on this system. The installation was done using the Redhat provided installer. When planning the system it was noted that this system was both a production server for departmental resources as well as a development and research machine for faculty and students. This affected the installation because it meant that although a limited installation would be desired for the production aspects of the machine there would need to be additional applications, libraries, and languages installed for the development and research occurring on the machine. Normally, it would be preferred that a system have only the packages installed

that were needed for the operation of that machine. When unnecessary packages are available on the system it both clutters the layout of the machine, but more critically these applications can still present vulnerabilities. Since they are not necessary on the system it is also possible that they are forgotten and therefore patching can be overlooked.

The installation of the system was done using another hostname and IP address. This was done so that the new system could be tested and configured while the original system could remain online and operational. Having the original system remain online enabled the new system to be setup and tested before the migration of the user data and the services.

3.2 Operating System customization

After the initial installation and configuration of the system to operate in the environment it was necessary to plan and execute the migration of operating system configuration. These are items that are included in any Linux or Unix system. The format of these configuration files are often the same irrelevant of particular operating systems. Most of the files that needed to be migrated were used to manage users' access to the system and the general management of the system into the specific networked computing environment.

The first files that needed to be migrated were the *shadow* file, the *passwd* file, and the *group* file. These store information that is critical for authenticating and verifying user access. Additionally, the *hosts* file also needed to be customized with the specific settings to resolve systems on the network. Although network based DNS infrastructure will resolve other systems, there are some aliases that are stored in this system that are not resolved by DNS. Also, maintaining a separate *hosts* file allows for

internal stability during campus network outages. If the network DNS becomes unavailable, the locally stored *hosts* file will allow the department server to connect to machines within the *hosts* file.

The *aliases* file also needed to be migrated to the new system. These would be needed to keep the mail system operating in the same way that it operates on the original system. There was also the need to verify that all the normal system operations were migrated. This included verifying and migrating the *cronjobs* on the system. These serve the purpose of performing scheduled system tasks without the interaction of users or administrators. With this system it was found that some of the *cronjobs* that were still set to run were not needed anymore. Some *cronjobs* were for services that were either being removed from the system, or were replaced with new packages. The new system also has new facilities to help with scheduling these tasks. There are *cron.daily, cron.monthly, cron.hourly*, and *cron.weekly* facilities included. These allow for scripts to be placed in the respective folders. The scripts in the folders are run by the system, and these folders simplify the management of the *cronjobs* since many *cronjobs* being run would work into one of these time frames.

3.3 Critical Application Configuration

Following installation and configuration of the operating system the next step was to install the critical applications for the operation of the system. For most of the applications it was possible to make use of the Redhat packages for the installs along with the default configurations of those applications. For the more critical applications there was a need for the customization of the install. These applications included the mail agent and the web server, *Sendmail* and *Apache*, respectively. These applications

are some of the most complex applications because of their high usage, need for high availability, and vulnerability to security threats. Since these applications are exposed to outside users it is very important they be properly configured for a specific machine.

One of the greatest benefits of upgrading a system as opposed to starting a new system was that there were configuration files from which to start. When migrating the system the versions were very similar on these critical applications. Web and mail service applications had been kept up to date as improvements and updates were made available. The configuration files for each of these applications were matched as closely as possible to ensure the same functionality. One major difference was the need for many changes in the *Sendmail* configuration file. *Sendmail* had many improvements made in security because of problems with mail server exploitation. With these improvements it was necessary to create a new configuration file that would implement these improvements. An application that is important to users is the inclusion of the *vacation* application, which works with *Sendmail* to respond to emails when the user is not available to respond. *Vacation* is an example of an application that is not necessary for the functionality of the system, but is seen as very important by some users.

The last remaining services that needed to be installed and configured were the *imap* and *pop3* services. One of the benefits was to install the secure counterparts of each of these applications. This was done in an effort to continue the trend of having secure methods of connecting with the server in the hopes of protecting passwords. The packages chosen were the Washington University instances of these services. The configuration of each of these was simply to build the packages, which were included in

the *pine* source package. Each of these needed to be configured to startup and allow connections for users. These were set to start in the *xinetd* on the system.

3.4 Evaluate replacement technologies

Many technologies are used on systems that are only needed by the administrators in order to make daily operation more manageable. One of the most burdensome issues for an administrator can be dealing with logs on a Unix or Linux system. Logs can become a problem because they very often can grow rapidly and occupy disk space on the system. One way of limiting this problem is to remove old logs from the system on a regular basis. Removing logs can also cause problems because an administrator might need the logs to evaluate system usage as well as evaluate problems on the system. Scripting the rotation of logs can be easy, but setting up these scripts and changing multiple scripts if there are changes in logging requirements can become a great annoyance to administrators. The original system log rotation was handled using a manually scripted method. *Logrotate* is a utility that ties together many of these log rotation scripts as well as providing an easier method for changing logging procedures in one location.

Logwatch is another tool that was easily installed on the system to deal with log management. *Logwatch* is a customizable tool which will automatically parse logs to the requirements put into the configuration file. *Logwatch* helps by saving the time needed to look through log files for common problems. *Logwatch* emails the results of parsing the logs so that the system administrator can evaluate the logs in a concise and clear format.

The web server logs provide information that is useful for both the administrators and the users of a system. On this system, logs are very important because the web server is probably the most important services provided as many organizations, users, and administrators of the system serve web pages. Web usage logs store who is visiting a web server, what they are visiting, how the visitor got to the site, and many other useful details.

On the original system this was done using A*nalog*, which parsed the log files into a more useful format. *Analog* extracted general server usage information after parsing the web server log files. The new utility that performed this function would be the *Webalizer* log analysis tool.

Webalizer provides great detail from the files, but it also is more efficient at parsing the files. *Webalizer* has the benefit of providing parsing files on a daily level, and storing those results into a history file that can provide up to date monthly statistics as well as the daily statistics. This cache means that each part of the file only needs to be read once and a running total can be kept of the elements being parsed. *Webalizer* is also very customizable through a simple-to-use configuration file. One sacrifice was detailed daily statistics. Although there are daily averages the *Webalizer* tool does not provide the detailed daily results that were provided with the *Analog* web utility.

Yet another issue to be dealt with was the ability to search the web pages located on the system. On the original system this search function was provided by a locally stored indexing utility, *Htdig*. *Htdig* would crawl through and index the webpage that were stored on the web server so that it would be available for easy searching. After analyzing the time and resources needed to regularly index these files it was decided to

eliminate this utility. Instead it was decided to provide a search interface via Google®. Google is probably the best known search engine available at this time. By limiting its returns to those of this particular web server it provides very good search results. Since Google already provides a very powerful search function making it unnecessary to occupy system resources by creating its own, local index.

Another feature on the new system is the use of the *locate* utility for file location. On the previous system in order to locate a "missing" file one would use the *find* utility. The problem with *find* is that if a system-wide search was needed the find function would have to look through all folders recursively, which was very time, disk, and processor intensive. The *locate* utility takes a daily scan of the system and indexes the files and folders on the system. This scan can be taken by root at any time if an updated index is needed, but normally can be run when there are lower system resource demands. This index allows very fast, accurate searching of the entire system architecture.

3.5 Build missing applications

Although the Redhat Enterprise Linux distribution came with hundreds of applications and utilities there are many that were included on the previous system that were not included on this particular distribution. It was necessary to acquire these packages and configure them on the system. Although there were many other packages on the system that provide the same functionality as these packages there was the need to keep the previous applications on the system. One of the major reasons was to allow users to continue using their prior configuration files and to allow them to have access to the prior functionality.

Two of these packages are provided in the same source package. These are the *pico* application and the *pine* application. *Pine* is an email management client. *Pine* is a very powerful and user friendly email client given that it does not include a GUI. *Pine* has built in filtering, folder management, mail management, and composition functionality. The configurations are stored locally in the individual user's home folders. In order to allow users to continue using these configuration files it was very important to provide them with the same application. *Pico* is a file editing package that is built along with *pine*. *Pico* provides the same composition interface as is used in the *Pine* package. It is much friendlier than *vim* or *emacs* and performs simple file editing very well.

Another application needed by the users is the *lynx* package. *Lynx* is a text based web viewing package. It allows for viewing web sites in a text environment, which is how users connect to the system. The *links* package is included in the distribution and provides similar functionality. However, in order to provide the users with the familiar application it was important to install *lynx* on the system.

4 Testing

The largest benefit of setting the system up before migration of services and users in this implementation was the ability to test, tweak, and reconfigure the system before bringing it online for users. With the new system there was the problem that many of the applications used were available for the new operating system, but the versions or configurations were different for many of them. Although there was the base configuration, the new versions often meant that new features and old features needed to be configured to provide the same functionality as well as improving the setup using the new features.

4.1 Testing applications

Testing the applications and services on the system was a process that involved trying to interact with the system in the same fashion the original system would interact. This involved testing the features of the web server. One problem was that it was difficult to fully test the pages because the server had to be configured slightly different. For example, the virtual servers could not be tested because it was not possible to get the server to respond to the virtual domains without changing the pointers in DNS, which would have affected the ability to make use of the pre-existing active system.

It was also difficult to fully test the configuration of the *Sendmail* daemon because it would not be able to receive emails to the domain as opposed to the specific system address. The system was tested for functionality using the available server address, but the final test of response to the email domain would not be able to be fully completed until the system was fully functional.

The system was tested to determine if the *imap* and *pop3* servers were correctly configured. This involved connecting to the system using the specific machine address of the new system. The response to the client was easily verified and testing the reading and management of email was verified using this configuration.

4.2 Limited usage to test functionality

User interaction was tested on the system. Although the general subsystems were tested and functioning via the external connections there was also the need to test the functionality of the applications that were used by the users when they connected locally via a remote shell session. This would first require testing the secure shell sessions. Users would need to be created and their environments tested. The two system administrators, the applications developer, and a research faculty member, all advanced

users of the system, were all asked to log into the system and try to use the system as compared to the original system.

One of the major differences between the original system and the new systems was the preferred shell. On the new system the preferred shell would be the *bash* shell. The shell that most of the users were using on the original system was *tcsh*. Although there are many similarities between these shells that benefits of the bash shell made it worthwhile to attempt to migrate the settings of the environments. With the setup of the shell environments it was found that many of the older configurations, including the path setting and the environment variables were no longer needed, or were going to have to be modified for the new system. Some of these variable changes were based on the change between configuration setting of the new systems and some were because the newer versions of applications would either no longer need the variables or the variables value itself would need to be changed for the new version.

The scripts that were used in managing the system had to be modified. The normal user creation script needed to be modified because of the new home folder organization. A custom script was created to make use of the benefits of the system's *adduser* script and yet be able to handle the hierarchical folder structure (Appendix A).

5 Migration

After the configuration and analysis was complete one of the final steps was to migrate the data from the original system to the new system. Following the planning of the migration, the users needed to be notified about how the migration would impact their use of the system. It is very important for the administrators to protect the integrity of the data during the migration as well to keep the impact of the migration as minimal as

possible for the critical system services. The detailed planning of the migration helped to control the impact of the migration.

5.1 Data migration planning

After testing and configuration were completed, it was time to start planning and initiating the final migration of the system. First, sample migration steps were tested. With the selection of the *rsync* migration it was necessary to determine approximately how long the migration of the data would take. The user files and the root web files were approximately 140 Gb. Based on the many partial data migration tests it was determined that the migration should take about eight hours to move almost 140 Gb of files.

When looking at the data on the system it was found that there were many projects and users that were no longer actively using the space that they were occupying. There were many accounts that had over 8 Gb of data. Many of these accounts were projects that were no longer active on the system. The users who were responsible for these accounts were contacted and informed of the need to reduce the size of these accounts or the administrators would have to reduce the size for them. There were many users who had accounts on the system that were no longer accessing the system. These accounts were archived into compressed files and stored on alternate systems in case the user needed to gain access to these files even though they have been inactive for an extended period of time. Once archived to remote locations, the accounts were purged from the system. There was also an attempt to locate users who had not logged into the system for over a year and/or had not checked their email in over a year. This was the threshold for determining inactive accounts. After purging out of date or unnecessary data from the system the volume of data that would need to be migrated was reduced to

approximately 80 Gb. The size of the mail data files on the system was approximately 6 Gb. This meant that 86 Gb of data needed to be migrated to the new system. This brought the estimated migration time to approximately 5 hours.

The migration testing occurred by migrating subsets of data to the new system and comparing the times needed for each migration. These test migrations ranged from a smaller migration of 3 to 5 Gb up to a test migration of 14 Gb. The test migrations were done at different times of day in order to account for system usage, but they were done during lower system usage times in order to reduce the impact on normal operation.

5.2 User notification

Once an estimate of the migration time was determined, it was necessary to notify the users of the system downtimes and outline expectations for service availability during the migration. The methods of notification were to post a notice on the message of the day, which is loaded every time a user connects to the system via *ssh* to get a remote shell on the system. Emails were sent to mailing lists that would notify the majority of the active users on the systems; these lists included current students, faculty, and staff within the organization. The notices included general information about system changes that would affect services for users. They were notified of the changes in data organization and changes to the methods of remote file updates. Included in the notice was a warning that the system would be unavailable during different phases of the migration.

5.3 Data migration control

In order to limit the changes that would be made to data as it was being migrated there was the need to limit user access to the system during the actual time of migration. This would mean that once migration began users would not be able to modify the files

that were being moved. If the users were able to modify the files there were the chances that some of those changes would be missed in the migration. The time selected for the migration was a holiday so that the number of users actively attempting to use the system would be reduced. This was done in an effort to reduce the impact of the migration on the system users. Users were advised that although the migration should take about 6 hours there was the possibility of unforeseen problems that would limit the ability to complete the migration in the expected timeframe. For this reason the users were notified that access could be limited for up to two days. Given the holiday season it was determined that this would be acceptable downtime for user login in the worst case scenario.

The migration process included first limiting logins by modifying the *passwd* file. The *passwd* file was modified to give users an invalid shell. It was decided that remote access would need to remain open so that the administrators could more easily monitor the migration of the data on the system. All the users except the system service accounts, the root administrator account, and the system administrators accounts were disabled from getting a local shell. The *imap* and *pop3* services on the machine were also disabled so that users would be unable to modify their mail files. Although email would still be able to be received into these mail files it was desired that the users not be able to modify and manage their mail files.

5.4 Protection of critical services during migration

Although logins would be closed for the users there were other services which would not allow for such a long-term downtime. There services were the web server and the mail services provided by the system. Although users would not be able to check

their email, in order to avoid changes, it was desired to keep downtimes of these services at a minimum. The reason for this is that the people affected by these services having extended downtimes could not easily be notified prior to the migration. Since the web server is the public face of the department on the Internet it was determined that as limited a downtime as possible was desired. There was also the problem that if the email server is down for an extended period of time there can be the loss of emails as the remote systems are unable to deliver the messages. The longest time that was acceptable for mail service downtime was determined to be 4 hours. This timeframe was chosen as the worst case downtime because 4 hours is the normal time that a mail server will attempt to deliver a message before notifying the sender that the message was undeliverable.

5.5 Data Migration Strategy

5.5.1 Data Needing Migration

A series of scripts were used to migrate the user data (Appendix B). These scripts were designed to allow for multiple channels of data migration between the systems. There were 5 different *rsync* data channels that were users to migrate the data. These channels were created by determining the volume of data which would be in each second level folder under the new data organization schema. This was done so that the five streams would end at approximately the same time. With the volume of data to migrate it was estimated that it would take slightly over 4 hours to migrate user folders.

Once the user data was migrated the web server file migration started. This was approximately 10 Gb worth of data. This migration was set to begin following the

completion of the user data migration. Based on test migration it was estimated that this would take approximately forty-five minutes.

The final pieces of data that needed to be migrated were the user mail files. These accounted for approximately 6 Gb worth of data. This migration would take an estimated half an hour to complete the migration of the mail data.

5.5.2 Data Migration Steps

In order to protect the data and complete the migration there were a series of steps limiting modification while still allowing most services to operate normally. First the *passwd* file was replaced with a modified file. The *passwd* file had the shell for users set to an invalid shell so logins were disabled. It was important to move the *passwd* file over the current file. If the *passwd* file is moved and then an administrator attempts to put the modified *passwd* file in its place the move will fail because a *passwd* file is not available to verify permissions for the move to occur.

Once the modified *passwd* file is in place the user data migration was started. Following the completion of the user data migration the web document migration was started. Upon the completion of this data, mail folder access was halted so that they could be migrated. The *imap*, *pop3*, and *smtp* services on the machine were brought down. The migration of the mail files began immediately following the disabling of the mail services. This was the portion of the migration that has a very limited possible downtime.

While this migration of data was occurring the *Apache* web server and *Sendmail* configuration were switched to the have what would be the new system name on the new system. For *Apache* this involved changing the IP to the IP of the original system as well

as changing the *hostname* of the server to the original systems *hostname*. In order for *Sendmail* to respond to the correct hostname the *sendmail.cf* file was changed to listen to the original hostname.

Upon the completion of the mail data migration the remaining changes were to disconnect the original system from the network and change the networking configuration for the new system. The *hostname* and the IP address of the original system were put onto the new system. Finally, the system was rebooted in order to test that the configuration was set correctly in the startup scripts so that the system changes would be set when system reboots would be needed.

6 Evaluation and Follow-Up

6.1 Migration Evaluation

The migration was completed as planned. The process of migrating the operating system went very smoothly, with few problems. The problems that did arise were very easily resolved with research of the documentation for the respective application. The planning of the migration was the most time demanding, along with evaluating and implementing the changes to be made following the migration. With many of the applications being on the original system, as well as the new system the most complex problems involved differences which were intentional.

The data migration was faster than it had been expected. This was most likely because of the resources gained by the system being inactive for some user services as well as the lower utilization of services due to the migration being completed on a holiday. The final migration time was the user data being migrated in two hours and fifty minutes to migrate approximately 70 Gb of data. During the move the data was

simultaneously reorganized into the new structure without incident. One issue in particular was difficult to evaluate. The migration to a new file system type showed a difference in the volume than appeared on the original system. The web documents and the mail files were also migrated without incident. They were migrated in well under their planned migration time as well.

The overall downtime of services was approximately twenty five minutes for email services, including *smtp*, *imap*, and *pop3*. This was well under the goal of keeping the service migration under four hours. The downtime for the web server was approximately ten minutes. The downtime for remote logins to the system was approximately five hours. All of these downtimes were well under the possible estimated downtimes for the migration.

6.2 Change Evaluation

The major changes to the system were the data organization, the elimination of *FTP*, and the replacement of some of the system utilities and applications. Some of these had a larger impact on the users and some of these had a larger impact on the administrators.

There were some problems that came about from the reorganization of the data. Although many of the application were able to resolve the symbolic links that would keep the data organization identical to the original system organization, there were some things that could not resolve the system change. The main situation that presented the problem was some of the users who had paths stored in scripts and access files. These included, most notably, the *.htaccess* files that users had protecting folders and/or files that were served through the web server. Within the *.htaccess* files, there were file paths now

incorrectly mapped for the folders that were being protected. It was necessary to notify users who had access restriction files within their folders. The first portion of the file paths needed to be changed in order to include the folder with the first letter of the username for each folder. Once the additional folder was entered in the path the problems were resolved. Despite these minor problems the use of the symbolic links for organization was successful and should allow for the future expansion of disk space without incident.

The other change that had the most impact on the users was the elimination of *FTP*. The most critical piece of information in dealing with this was to notify users of the alternatives for uploading files and file modifications. Although there was an attempt to notify users of this impending change before it happened there were many users who were not prepared for the change. Once the users were pointed in the direction of a secure *FTP* client there were not problems with accessing the system. Because the secure *FTP* operates in a nearly identical fashion there was not any new documentation needed on how to deal with the system.

The utilities that were changed, *Logrotate*, *Webalizer* utility, and *Logwatch,* were some of the most substantial improvements for the management and monitoring of the system. The *Webalizer* tool has provided more detailed and more efficient web server log analysis. The *Logrotate* utility has provided a flawless log management system. With the more frequent rotations the disk space occupied by logs has been reduced. The ability to manage logs has also become easier with the common, simplified facility to manage their archiving and rotation. *Logwatch* has provided simplified reports about *Sendmail* issues, access attempts for all of the services that allow outside connections,

uses of administrative system utilities, and disk usage information. These concise reports are detailed enough that they give enough information to monitor usage and system status without making use of any other system utilities.

6.3 Overlooked Applications

Despite the best efforts in the system analysis and migration planning there were a few items that were overlooked in preparing the new system for usage. Many of these dealt with the web server, because it was one of the services that was used in such a variety of ways and by so many individuals. Other problems revolved around the fact that the system was used for so many different reasons. Some of the more demanding users made use of the system for virtually all of their computing needs, and some of these needs were missed due to a small group of users.

The largest group of issues dealt with the *Apache* web server configuration. One of the largest problems was that the default character set of the server was different that it had previously been. The problem was noticed because web pages that were written using Microsoft Office packages were not loading correctly. Once the character set was changed to ISO-8859-1 the problem was resolved. It was also discovered that some *cgi* scripts were not migrated in the move. The reason for this was traced to being located on the partition with the web server configuration files. Although the root document folder was changed, the location of the default *cgi-bin* location was not moved. These scripts had to be recovered from the original system. One of the scripts had to be recompiled from updated code in order to function on the different platform.

Also related to *cgi* scripts was a problem with user *cgi-bin* folders. This problem was discovered by users having problems getting *cgi* scripts to load correctly. This was

due to a change in the web server configuration for security reasons. With *cgi* scripts being a serious risk, due to possible poorly written scripts or users granting permissions incorrectly for *cgi* scripts, open access was eliminated. The workaround was to alias and enable scripting for users on a case by case basis of needing it activated.

An additional problem was that there was not a text based news reader on the new system. This was overlooked because it was not seen as being used on the old system, and therefore did not get compiled on the new system until the problem was reported. Furthermore, there was a problem with the java development package on the system. Although there was a java package installed and was tested on the new system it was not a software development package. A java development package needed to be installed in order to enable java software to be developed and compiled on the system.

References

Hewlett-Packard Development Company (2003). *HP Optimizes Solaris-to-Linux Migration*. Retrieved September 15, 2003, from http://www.hp.com/go/linux

Bibliographic References

HTTP Server Project Documentation (2003). Retrieved October 6, 2003, from http://httpd.apache.org

Barrett, Bradford. *The Webalizer Home Page*. Retrieved October 20, 2003 from http://www.webalizer.com.

Bower, Kirk (2003). *Logwatch Documentation*. Retrieved October 20, 203 from http://www2.logwatch.org:81/tabs/docs/ .

Ht://Dig Group (2003). Retrieved November 3, 2003 from http://www.htdig.org/ .

IMAP Documentation (2003). Retrieved November 24, 2003 from http://www.washington.edu/imap/

Pine Documentation (2003). Retrieved November 24, 2003 from http://www.washington.edu/pine/

RedHat Licensing and Documentation (2003). Retrieved December 8, 2003, from http://www.redhat.com

RSYNC Documentation (2003). Retrieved September 30, 2003, from http://rsync.samba.org

RPM User Documentation (2003). Retrieved September 22, 2003, from http://www.rpm.org

Sendmail Documentation (2003). Retrieved November 17, 2003 from http://www.sendmail.org

Appendix A

Custom adduser script for adding users in the new disk scheme.

```
#adduser.pl
#!/usr/bin/perl -w
BEGIN: while (1) {
        &ADDUSER();
        print "\nTo exit type ^c otherwise enter new data.\n";
        }
        # Get the name and pid number from the command line.
sub ADDUSER {
print "Enter the first and last name of the newuser.\n";
chomp ($name = <STDIN>);
print "Enter the userid. \n";
chomp ($userid = <STDIN>);
        # create the userids
$var =substr($userid,0,1);
$userid =~ tr/A-Z/a-z/;
system("useradd -m -c \"$name\" -d /export/home/$var/$userid -g users $userid");
        # Begin the loop again.  Get the name and pid number again from
        # the command line.
system("passwd --stdin $userid");
}
```

Appendix B

Sample of migration scripts. There were four scripts which migrated the data similar to this first script.

```
#migrate_home_1.sh
echo "a" >> /fs1/migrationresults
date >> /fs1/migrationresults
rsync -al rsynctest@ruby.ils.unc.edu::home/a* /export/home/a/
echo "a" >> /fs1/migrationresults
date >> /fs1/migrationresults
echo "b" >> /fs1/migrationresults
date >> /fs1/migrationresults
rsync -al rsynctest@ruby.ils.unc.edu::home/b* /export/home/b/
echo "b" >> /fs1/migrationresults
date >> /fs1/migrationresults
echo "c" >> /fs1/migrationresults
date >> /fs1/migrationresults
rsync -al rsynctest@ruby.ils.unc.edu::home/c* /export/home/c/
echo "c" >> /fs1/migrationresults
date >> /fs1/migrationresults
echo "d" >> /fs1/migrationresults
date >> /fs1/migrationresults
rsync -al rsynctest@ruby.ils.unc.edu::home/d* /export/home/d/
echo "d" >> /fs1/migrationresults
date >> /fs1/migrationresults
echo "1" >> /fs1/migrationresults
date >> /fs1/migrationresults
rsync -al rsynctest@ruby.ils.unc.edu::home/1* /export/home/1/
echo "1" >> /fs1/migrationresults
date >> /fs1/migrationresults
```