The Water Fountain vs. the Fire Hose: An Examination and Comparison of Two Large Enterprise Mail Service Migrations

Craig Stacey, Max Trefonides, Tim Kendall, Brian Finley
Argonne National Laboratory

Abstract

Mail administrators will inevitably face a situation where they will need to migrate their users from one server to another, not infrequently migrating to a different service altogether. In 2008, two divisions of Argonne National Laboratory found themselves needing to migrate their users from disparate divisional mail servers to a central, institutional Zimbra Collaboration Server. Each group approached the situation from a different direction, driven by different motivations, timelines, and external forces; each ultimately achieved its goal, one more smoothly than the other. The first migration was driven by a high sense of urgency resulting in a "fire hose" approach, an en masse move followed by a grand switchover; the second migration was a more measured "water fountain" approach, taking in lessons learned during the first migration. Examining the processes, decisions, and tools used in each conversion yields a roadmap of successes and pitfalls that should prove useful to any systems administrators facing a similar task, regardless of the timeline within which they must work.

1. Overview

Argonne National Laboratory is served by a central IT services division, the Computing and Information Systems (CIS) division. As well, many of the programmatic divisions have their own IT staffs of varying sizes. This paper focuses on the work of the IT support groups from two of those divisions, the Mathematics and Computer Science division (MCS) and the Materials Science Division (MSD).

CIS offers services, including e-mail, to any of the divisions at Argonne. Until 2008, this e-mail service was provided solely as Microsoft Exchange. In mid-2008, Argonne began offering a choice between Exchange and Zimbra Collaboration Suite.

Prior to this migration project, both MCS and MSD ran their own e-mail services rather than using the central mail services for varying reasons that will be detailed below. MCS and MSD each maintains its own IT support groups, providing a number of services besides e-mail. Diagrams detailing the flow of mail to these divisions both before and after this migration are included in the appendices.

MCS consists of nearly 200 researchers, programmers, students, and visitors, with another 250 external collaborators. The division is home to several hundred workstations, three large clusters, and other high-performance computing resources. Aside from managing this diverse group of resources, the group also provides standard IT services such as web, mail, data storage, backup, and networking services. Management of these resources and services is handled by a single IT organization, the MCS Systems team, comprising 10 individuals with varying skill sets and specialties, as well as anywhere from 1 to 4 undergraduate students each summer, depending on workload and availability of interesting projects.

MSD is the focal point for research in materials science at Argonne National Laboratory and consists of over 200 researchers, students and staff. The MSD IT Operations group supports this division, providing support for over 200 workstation and several small clusters. MSD IT Operations also provides standard IT services similar to those provided by the MCS Systems team. The IT Operations team comprises 3 full time employees and 2 part time co-op students.

2. Mathematics and Computer Science Division (The Fire Hose)

MCS ran its own mail services, with user mailboxes provided by Cyrus IMAP on an AIX server with 6 TB (available) of fibre channel attached storage, an installation that was set up in 1998. Approximately 500 user mailboxes were active at the time of this migration, with another 200 lying dormant as their owners forwarded their mail elsewhere, totaling approximately 450 GB of mailbox data.

In this section, the process is described from the perspective of the MCS Systems team, with a summary of the CIS perspective at the end.

2.1. MCS Decision Process

The existing mail system was showing its age. In 2006, MCS began the process of evaluating an upgrade path for mail services. Ultimately, we decided to go with the same approach we'd been using for the past 8 years; mailbox services provided by Cyrus, though instead using a Linux server since our AIX expertise was lacking.

While we try to avoid these situations, an extended period of limited funding, staffing changes, and an overcommitted IT staff resulted in systems and services getting replaced only when they broke or failed to meet the existing need. As this server was generally rock solid, it was often overlooked, and its replacement was not considered an urgent matter.

While testing and implementation plans were being put into place, we became aware of a growing desire within our user community for shared calendaring and other collaboration tools. We entered into a joint trial of the Zimbra Collaboration Server (ZCS) with CIS, with MCS's focus being the calendaring component.

Several months later, the mail upgrade project was stalled as a result of other emergencies; however, the Zimbra pilot was going well. Realizing that the ZCS service inherently provides mailbox services, we re-evaluated our mail server upgrade plan.

Our decision to continue to provide our own mail services was driven by a number of factors, but one of the main motivators was that we needed to be in control of the data and service. When something goes wrong, our users expect us to be able to fix it, and fix it quickly. The prospect of outsourcing our mail services and leaving our IT staff unable to directly support it did not appeal to either the IT staff or management. To an outsider, this may seem simply territorial, and there is certainly some truth to be found in that thinking. However, historically, the relationship between the organizations that would become MCS and CIS had its rough patches. MCS management preferred a nimble set of services focused solely on advancing its research, and many saw CIS as slow, bureaucratic, and controlling. Overcoming this prejudice was not an easy task, but this seemed an opportune time to try.

By virtue of the fact that the Zimbra experiment in CIS was in its pilot stage, coupled with the fact that MCS was its largest user base, MCS systems administrators were given administrative access to the service. After confirming that this access would be continued in the production-level service, we decided to make Zimbra the mailbox service for the division.

We note that the scope of this migration was limited specifically to user mailboxes. MCS was not going to cease providing mail services; we still ran Majordomo-based mailing lists (which would be converted to mailman lists later in the year), as well as trouble ticket systems for ourselves and other groups, and virtual domain services. Therefore our solution needed to be able to support our remaining the primary Mail Exchanger for the domains we controlled, sending user-bound mail to the central service. This Zimbra solution fit the bill nicely.

Work on the conversion began in earnest as our mail server was continuing to show its age. For instance, the release of Mac OS 10.5 brought with it a new version of Mail, which many of our users use. This new version handled offline IMAP actions in a slightly different fashion from previous versions, and it was a way that seemed to cause our version of Cyrus IMAP to choke. This resulted in repeated error messages to the users and an ever-growing list of queued actions, as each failure caused a new copy of the offline action to be queued. As one can imagine, this situation got less and less bearable as time went on. Additionally, large mailings could bring the service to a crawl, and we were entering into a time of year when drafts of proposals would regularly be sent to large distribution lists. While it may not be the most efficient way of collaborating on a document, e-mailing Word and PowerPoint documents is certainly the most prevalent method among our users. Most significantly, as errors would occur and failures became increasingly frequent with the advancing age of the server, we felt we were increasingly in danger of losing mail.

2.2. MCS Migration Plan and Implementation

Problems with our existing server notwithstanding, we had what was fundamentally a simple problem – find a way to move messages from one IMAP server to another. We did a fair amount of research to locate the existing tools that could accomplish this move, since something of our own construction would likely be too much effort for what should fundamentally be a solved problem. Based on this web research, consensus in the community seemed to be that using imapsync [Lamiral] would be the most reliable method of accomplishing this migration. Likewise, Zimbra's own recommendations in migrating to a new server recommended this path of action [Zimbra]. In using this tool, however, we had to consider the limitations of our setup:

- The old mail server (cliff) was being pushed to its limits already; therefore our migration could not be too aggressive on the server.
- Because imapsync uses the IMAP protocol, it requires us to know the users' passwords on both systems. While we could set their passwords on the new Zimbra server, since they were not yet using it, we could not know their passwords on the existing IMAP server, as it was NIS-bound and using their regular workstation passwords.

We circumvented the first problem by limiting ourselves to two concurrent syncs – testing indicated this was an acceptable load. The password problem was more complex, but our situation allowed us to employ a creative workaround. Because our mail server was NIS-bound, we attempted to use a local */etc/passwd* entry on the machine, allowing us to login with our system password, and allowing the users to login with their NIS passwords as a fallback. Alas, this did not work on our version of AIX, but it did give us the idea that solved the problem. We could create entries for a "ghost" user on the mail server with the same UID and path as the real user that used our system password. At the same time, our script would modify the flat *mailboxes* file that Cyrus used to map mailboxes.

After testing confirmed our scripts were doing the right things, we ran the migration process day and night over a period of weeks. Various pitfalls were encountered along the way because of a number of situations our testing did not predict, such as disks filling up, networks going down, and previously undetected corruption in mailboxes.

Throughout the entire migration process, this corruption in mailboxes posed a significant challenge, as there was no predictable method to discover the corruption until we tried traversing the mailbox structure and reading individual messages; indeed, the messages appeared to be normal in any index of a mailbox, and the problems appeared only when the messages would be read. The most common corruption seemed to be in the oldest mailboxes; indeed, some employees had mail archives dating back into the late 1990s. While this in and of itself should not have caused a problem, the errors seemed to be caused by the varied (and no doubt interesting) lives these messages led. Some originated on the precursor to the mail system we were replacing (Sun OS 4.1.4's Mail with Sendmail and qpopper). These messages were stored in a monolithic mbox file, POPped off the server by the user into their e-mail client of choice, then later reimported into the Cyrus server via IMAP. Our most plausible theory is that changes in header format and attachment handling is what caused Cyrus to fail on loading these messages, as almost all of the corrupt messages were messages from Exchange users with attachments. These messages would have to be removed from the inbox by hand. Happily, while the IMAP clients seemed to fail on transferring or reading the messages, they were generally capable of deleting the messages; only a small handful of messages required a file-level delete and mailbox reindex.

Ultimately, it became clear that the IMAP-only method of syncing was not going to solve this problem; it was simply too slow. Connections were timing out on large mailboxes, resulting in incomplete data syncs. Also, the problems on cliff were steadily getting worse, and it became clear we needed an aggressive schedule for the migration. We came to the realization that our beloved-yet-overworked "little engine that could" was on the verge of switching from "I think I can" to "I think I'm done." As such, the slow-and-steady approach was beginning to look like it was a bigger risk than charging forward. It was the end of February, and an organization-wide maintenance window was scheduled for the weekend of April 26 and 27. We chose this weekend for the migration as most services would be down already, and users would expect a loss of service over that weekend.

Our second pass at moving the data involved getting the raw mailbox data onto the new server via rsync, using Zimbra's command line tools on the server to import the data into user mailboxes, and then using imapsync to synchronize the flags on the mailboxes with their counterparts on the old server.

To ensure we did not bring the main Zimbra server's network to a crawl during our data sync, we synced to a development server with the intention of mounting the disk on the production server for the import. As is becoming evident to the reader, things rarely worked out the way we planned them, and this was no exception. The initial sync of the data took an excruciatingly long time, though we were hopeful the import into the production server would be a much quicker operation, since the slowness of cliff would be out of the equation. Alas, the nature of the SAN defeated us; it turned out both the rsync *and* the mailbox import were I/O bound, and our development server's SAN was not as robust as the primary storage on the production server.

Our self-imposed April 26 deadline was fast approaching, and our progress was indicating we would probably finish the initial data sync the day before we were to switch over. The IMAP syncs were alphabetical by user name, and this estimate was based on progress throughout the alphabetical list of usernames. Astute readers should be able to predict the next pitfall we hit – our largest mailbox was one of the last ones alphabetically. On average, our user mailboxes were several hundred megabytes, with users rarely crossing into the gigabyte range. This particular user had a mailbox of over 20 GB. We realized we would have to handle this mailbox out of band if we were to meet our deadline, and we started syncing it and other large mailboxes concurrently, outside the automated process.

On the morning of April 26, it was evident that the sync would not be finished. Because the outage window had already been announced, we plowed ahead and attempted to finish the migration, figuring a day for the heavy work and all of Sunday to finish and tie up loose ends.

We turned off all incoming mail and started the final sync of user mailboxes with imapsync, which would capture any messages that arrived since the initial sync, along with setting the message flags for the users. This script ran throughout the day as we set up the new rules on our SMTP relays to direct mail to the correct servers. Because the old mail server would still be processing Majordomo mailing lists, we had to detect whether mail was for a user or mailing list and route it accordingly. By Saturday night, things seemed to be progressing well, and our spot checks on mailboxes looked okay, except message flags did not seem to be getting set correctly.

On Sunday, it became clear what was going wrong; the "ghost" user on the old server had full access to the user's mailboxes but did not share the message flags, and all messages were seen as "new". (We later surmised that even though there was a single copy of each Cyrus indexing file per mailbox, that file was storing a set of flags for each username that accessed the mailbox, as opposed to each UID; thus, all messages were "New" to the ghost user.) Because the mail system was now down, we uncoupled cliff from NIS and used only the local */etc/passwd* file, allowing us to use the user's real mailbox with our system password. This strategy solved the message flag issue, but we became aware of another issue. The script calling imapsync was supposed to be nondestructive; messages deleted on the old server should not be deleted on the new server. Because of a misreading of the configuration, however, this was not the case, and messages *were* being purged from the new mailboxes in some cases. In theory, such purging should not have mattered. However, perceptive readers will remember we flipped the switch on delivery of new messages on Saturday morning. Hence, any new mail delivered to the new mailbox was being deleted as soon as that user's sync was run.

Also, because of mailbox corruption on the old server, a handful of users had their mailboxes emptied on the new server. We needed to reconstruct these mailboxes from backups, or in some "friendly user" cases (i.e., fellow sysadmins), the users restored their own mailboxes from their local backups.

The migration was not yet complete, but users were getting anxious. By 7:00 Sunday evening, users whose accounts we had deemed to be fully migrated were allowed into their new mailboxes. Surprisingly, this did not go poorly. In fact, the feedback we received during this "early access" period helped us in later diagnoses. From what we were hearing, we could determine that in most cases, where the user's mailboxes were small, the migration was a success. However, for users with large or complexly organized mailboxes, it became apparent rather quickly that the migration was not complete. Entire years' worth of mail were missing from the mailboxes of some users who kept large archives.

By Monday morning, it was clear we had much more work to do to finish the migration. We announced to the users that we had reason to suspect the migrated mailboxes were not complete and that we would instead implement an approach whereby new mail continued arriving at the new mail server, and users would migrate their own mail via their mail clients, with help from the IT support staff where required.

This manual user-initiated sync took place over the next two months in a gradual process, with most users being completely migrated by mid-May. In part we were able to accomplish this migration by announcing that the old server would be shut down at the end of May. As someone wise beyond his years once said, "Announce the demise of the old [system] well in advance of really discontinuing it" [Evard94].

In the cases of users with large or deeply nested folder hierarchies, we engaged in a great deal of "handholding" to guide them through the process. Unfortunately, these users tended to be among the less technical savvy in the division, and as such the workload in that handholding was significant. Also, as outlined below, some mailboxes could not be migrated at all without some server-side tweaks.

We emphasize that the MCS users were marvelously patient throughout this process. Indeed, a key in maintaining this level of patience was proper communication. As noted in Tom Limoncelli's AT&T Network migration, a high level of communication and status updates will make the users feel more a part of the process (and less a victim of it) [Limoncelli97].

2.3. MCS Pitfalls and Lessons Learned

With each approach we devised, the plan seemed foolproof on paper, and at each step of the way, something popped up proving us wrong. The list of things that went wrong reads like a proof of "Murphy's law."

The combination of imapsync and our aging mail server were incapable of moving the mailboxes. In fact, IMAP itself had great difficulty in handing some of the user mailboxes. Often, users would archive mail into folders they would never again look at. As these folders grew in the number of messages contained therein, some reached a size that would make it impossible to access them over an IMAP client; as the old mail server struggled to stat the files, the connection would time out. To get around this situation, we would manually break up the mailboxes into smaller folders, reindex the folder, and begin anew.

The rsync of the mailbox data was restarted numerous times because of failing disks, high CPU loads, and network outages. In some cases these syncs had been running uninterrupted for days before crashing. With each restart, we lost precious time as file systems were compared.

The misconfiguration of imapsync in our migration script was a significant pitfall. By using imapsync incorrectly and losing messages, we undid a significant portion of the work that was accomplished. Human error is going to happen in any venture driven by humans and can be easily compounded by late, stress-filled nights that follow long, stress-filled days. In short, a simple typo of a flag was a devastating blow to both our progress and morale. A second set of eyes on these scripts would have gone a long way toward solving this problem.

Numerous restarts in various parts of this project plagued us. In the period between January 3 and April 25, we started from "square one" five times after a previous plan of action proved unworkable. Instead of having 4 months to migrate, we effectively had 2 weeks. This time constraint ramped up our stress levels, knowing that delaying the move could only exacerbate the situation, living in fear of the old mail server falling over.

All of the work we did to move the data from the old server to the new server was ultimately abandoned. This was, perhaps, the hardest blow to our collective psyche. The "brass ring" throughout this process was our knowing we'd done all the heavy lifting for our users, and they'd not have to deal with the migration themselves. Instead, not only did we go through a tremendous effort, but it was for naught.

Because a significant source of angst in this process was the lack of documentation, we continue to ensure we do not run into this in the future. Much of the old system was simply undocumented, existing only in the head of the previous mail administrator – clearly not a sustainable method of operating. We have ramped up our efforts in documenting processes and configurations, and we've ensured that more administrators are involved in the operation and configuration of the servers, avoiding the single-point-of-knowledge problem we typically faced.

The biggest contributing factor to our problems with this migration was related to the age of the hardware, operating system, and software of our production mail server. Combined with poor documentation, this left us with an aging mail system that for years had generally performed well with little intervention, and nothing but fading institutional memory on how to repair or tweak it. And, as is the case with any stable rock in a dynamic ecosystem, it had acquired roots and tendrils embedded in it that we are to this day still trying to disengage.

As noted in Section 2.1, the root cause of the age of this system was its generally working as expected during a period of time where only "squeaky wheels" got the oil. Economizing on hardware by holding off upgrades can often seem prudent, and sometimes unavoidable, but it almost certainly leads to an inflated TCO in the longer run. Tallying the amount of work hours involved in extricating a long used and encrusted system from a reasonably complex environment would be an interesting exercise. Following a long-term plan for regular retirement and refreshing of hardware would have gone a long way toward mitigating much of our problems.

A technical factor in this process was the Cyrus IMAP mailbox database. This monolithic flat text, single-file database used by the version of Cyrus IMAP that we were running proved to both hamper and help our migration. We were hampered because the file was fragile, had a rigid format dependent on tabs, spaces, and sorting (requiring a different sorting than provided by AIX's sort command), and was prone to corrupt the mail stream when things went wrong. It helped because we had an easily scriptable way to insert the systems users in order to be able to get access to the users' mailboxes, by ensuring the "ghost user" was either the first or last alphabetically (i.e. "aaaaaaaa" and "zzzzzzzz").

In Section 2, we mentioned that we had 500 active accounts and an additional 200 that were later determined to be dormant, resulting in our moving 40% more users than we needed to. We gave thought to indentifying the unused mailboxes prior to migration, so as to avoid the work of moving users who no longer existed. A small amount of effort was put to this task, but we soon called it off as we discovered most of these users had very small mailboxes, and weeding them out from the process would be more work than simply migrating everyone wholesale. With a slower approach, it's more likely we would have taken the time to cull these unused accounts prior to a move – it was largely a decision based on the time left and the level of effort available.

We point out that, over time, our account and resource expiration policies have been disabling and deleting these mailboxes, and almost all have been removed with little work on our behalf.

The next time we have to perform a migration of mailboxes, we'll be far more likely to employ the process we ended up using after all other plans failed. We would choose a cutover date when all new mail will be delivered to the new server, and allow users to migrate their own mail with help from IT support before an announced deadline wherein the old mail server would be shut off.

While we certainly engaged in testing, we failed to properly identify the edge cases. In some cases we chose what we expected to be difficult mailboxes on which to audition new migration methods, yet we had a knack for choosing examples that, while certainly large and well aged, were problem free. A better sampling for our testing would have gone a long way to identifying many of the pitfalls in advance of our migration deadline.

When coming up with our migration plan, CIS recommended we employ a more staged rollout. We opted to go "all-in" as we did not feel we had the luxury of the time required to engage in such a migration. Of course, the irony of this situation is the mail server we were convinced was going to fall over at any moment stayed up through the manual migration process. In fact, it was finally retired in August of 2009.

It's also important to consider that our group tries to make things as seamless for our users as possible, and all of our research indicated we *would* be able to accomplish this migration with little to no user impact. Aside from updating their mail client configurations, the only change our users were supposed to notice was a faster and more reliable mail service. We have certainly learned that this was too lofty a goal in the given circumstances.

2.4. CIS Challenges and Participation in the MCS Migration

From the CIS perspective, Zimbra had been very successful as a pilot service, but we had no true experience running Zimbra as a production service, or with any significant data or user load. Going from a dozen gigabytes of mail to trying to appropriately scale the system to instantly take on roughly half a terabyte of mail data and 500 users was a cause of some concern, and a bit of a challenge.

Zimbra allows for separation of disk volumes for performance and cost reasons. CIS provisioned the production system with separate volumes for redo logs, primary mail store, and secondary mail store, among others. Mail flow into the system, including messages added via IMAP, first land in the redo logs, then the primary mail store. A weekly scheduled Zimbra HSM process then migrates old mail from the primary mail store to the larger secondary mail store on lower-performance, less expensive disk.

One unanticipated effect of the "fire hose" approach was the need to closely monitor volume consumption on these separate volumes; in particular the redo log and primary mail store volumes, neither of which was intended to be able to completely contain the amount of data being transferred during the MCS migration.

As the redo log volume filled up, it was necessary to manually invoke an incremental backup using Zimbra's self-backup facility. The Zimbra self-backup facility allows for atomic point in time restores, and does so by replaying appropriate bits from the redo-logs, which it copies to a backup volume during incremental backups. As the primary mail store filled up, it was necessary to preemptively invoke the Zimbra HSM facility. Fortunately, message age persisted in the migrated mail, therefore allowing this process to work.

The HSM process, as the solution to the primary mail store filling up, was fairly easy to identify. It just made sense, we already understood how it worked, and had intended it for this purpose, just not on this schedule. On the other hand, we had no prior experience with the redo logs growing out of hand. Previously, the already scheduled daily incremental backups automatically handled them, so we had no prior need to pay them any notice – it just worked. This is a good example of the challenges of accurately modeling behavior of a system at scale in a small or simulated environment.

CIS wasn't too concerned about high load placed on the Zimbra server during the MCS migration, as they were the first production user base to migrate. In other words, if the migration caused performance issues, they would be affecting only themselves. This was a luxury that future groups making the migration would not be able to have.

3. Materials Science Division (The Water Fountain)

MSD ran an iPlanet mail server on a Sun server with approximately 120 mailboxes including service accounts. A majority of the mailboxes were active at the time of migration, as MSD had been doing some house cleaning to keep adequate free space. At the start of migration there were over 190 GB of mail.

In this section, the migration process is described from the perspective of the MSD IT Operation group.

3.1. MSD Decision Process

The current MSD IT operations staff had inherited an aging Sun e-mail server that was getting more costly to maintain. Maintenance contracts and the cost of adding additional storage were cost prohibitive because of the age of the server. Additionally, as the existing server had been installed and operated by administrators no longer with the division, there was a lack of expertise with this install.

MSD IT Operations was relatively new department to MSD, as IT support had been handled by an Argonne division that had been dissolved. Despite having a new IT staff, the division had inherited an aging IT infrastructure built and maintained by another group. Because of this older infrastructure MSD wanted to explore the possibility of using the CIS e-mail systems, yet we were apprehensive about relinquishing control. The division is accustomed to having its services run by a support group whose only responsibility is their own division. Bearing this in mind, we did give some consideration to bringing a new e-mail server online. But since we had so many other infrastructure problems to deal with, we felt the benefit far outweighed the consequences of migrating e-mail services to CIS. Additionally, using CIS e-mail gave us the advantage of using Argonne's central Active Directory authentication, as MSD users were tired of having several different authentication methods.

Since MSD had a large Mac OS user base, moving to CIS Exchange servers was not our first choice because of the various issues Mac OS users can have with connecting to Exchange. (Historically, the laboratory's Exchange server did not interact well with Entourage. This problem was solved after our migration was finished.) At this time we became aware of the CIS Zimbra pilot project and started a dialog with CIS and MCS regarding migrating to Zimbra. After MSD completed initial testing and conversations with both the Zimbra lead and the MCS lead, MSD joined MCS on the Zimbra pilot test. This was in the early spring of 2008, but unfortunately several other more urgent projects needed attention, delaying the start of planning of the migration until late July 2008. It was during this pilot test that MCS performed its migration. After the process was complete, the MSD administrators met with our MCS colleagues to discuss their process.

Since other commitments by IT staff had delayed work on the migration, we, too, started to feel a sense of urgency. We had two factors influencing our deadline; our maintenance contract on the Sun server was expiring in late 2008, and our SSL certificate would expire shortly after that. MSD did not want to incur the cost of renewing either of them, knowing the service was bound for decommissioning. Also, during a recent divisional review, there were many large e-mail attachments going back and forth among the users, resulting in one weekend where mail delivery came to a near standstill because of lack of storage space. Even after the review, it was a struggle to keep 10 GB free on the mail store.

Because divisional administrative support staff and senior management need to collaborate with others in the laboratory, a decision was made to migrate these users to the central Exchange server. Otherwise, all MSD users were to be migrated to the Zimbra server.

3.2. The Plan and the Pitfalls

Once we decided to use Zimbra as our primary server, new employees received accounts on the Zimbra service. Initially this was limited to postdocs, since Argonne's Zimbra service was still technically in the pilot phase. With the installation of ZCS 5.0, it was officially moved to production status, and we started adding all new employees' mailboxes to the Zimbra server. This relieved some of the storage issues on the current MSD mail server, allowing MSD IT operations to work out the remainder of the migration planning without quite so much urgency.

MSD looked at using imapsync; but after meeting with MCS and discussing the problems they had with it, doing an all-at-once approach was ruled out. Among the several reasons not to use imapsync was the need to know the user's password; MSD would not have access to user's AD account password for the Zimbra e-mail accounts. Furthermore, from a general customer satisfaction perspective, doing one user at a time was far more appealing, as we could start with a few users and test the migration process, hammering out any issues. Other reasons included the experience of some of our IT staff with e-mail migrations from previous positions at other organizations that employed expensive third party tools to perform a behind the scenes migration. Based on this experience, MSD IT knew we would most likely end up touching every workstation anyway.

The process we settled on was a new feature available in Zimbra, the import component of the web interface. We used this tool because it off-loaded the migration from the client to the server. Thus, the migration process did not tie up the user's workstation during the move, which was especially beneficial when dealing with older machines or a large mailbox migration. Since the Zimbra Web Client (ZWC) allowed users to add and check external POP and IMAP accounts, we had the user log into the ZWC and add the user's old MSD account. This approach caused the Zimbra server to import all the user's mail completely as a server-side action, regardless of whether the user is logged in on the ZWC. During the mail import MSD changed the primary e-mail alias to point to the Zimbra server. Once the account had fully loaded in the web interface, we then moved and arranged the folders or contents of folders to the Zimbra account's mailbox tree to mirror the old MSD folder structure. Once completed, we deleted the old account from the ZWC and set up the user's e-mail client to access the new account.

During the migration MSD encountered some users that were off-site a vast majority of the time. To assist these users, MSD wrote up documentation on how to do their own migration. Additionally, some users preferred to do their own migration because it provided an opportunity to cleanup their e-mail.

After MSD started doing several migrations a day, the Zimbra server started to slow tremendously, affecting other division as well. Migrations were halted while the Zimbra team investigated. After finding the root cause was Zimbra's indexing of attachments, we decided to turn off this feature for the time being. With attachment indexing off, migrations were much faster, even with heavy e-mail users (5 GB+ mail boxes), and there was no impact on other users' experience with the system. This issue did not arise during the MCS migration, because no other users were interactively using the service during their migration, so the high machine load was not noticed.

Rather than simply moving alphabetically through the mailboxes, scheduling was done with some consideration to the user's mailbox size: we started with the smaller mailboxes to make sure the process was working. Once the process was established and server concerns were addressed, we based the schedule primarily on the user's convenience. We scheduled it in batches and tried to get as many done in one batch as possible.

With any migration like this, one must address setting user expectations accurately on access to the old data. MSD established a policy that a user's old e-mail account would remain accessible for 7 days after the migration but only through the web interface. After 7 days the password on the mail account was changed; after 30 days the account was deleted from the server. This policy was largely adhered to except in some instances requiring us to set up access to an old MSD mailbox because something was not migrated or we missed changing an e-mail alias.

Another hurdle was some users were having e-mail addressed to the fully qualified divisional e-mail address (user@division.anl.gov) instead of the main Argonne alias (user@anl.gov). In the setup that existed at the time, any mail sent to user@msd.anl.gov would be directed to Argonne's mail gateway, then handed off to our own mail server; and as long as that server was still in the migration process, that setup had to be maintained. Since migrated users simply had their @anl.gov alias directed to their new Zimbra mailbox, they would not experience this problem, but these users who had distributed their internal MSD address needed their old e-mail account kept active longer while they alerted their senders and mailing lists. Other difficulties were the occasionally corrupted e-mail message on the old MSD mail server, as this would stop the Zimbra mail import. Once the corrupted e-mail message was deleted, the mail import would function as expected.

As a side-benefit of this migration, it allowed us to perform some account cleanup. MSD identified users who had retired but were still using their MSD mail account, as well as users who were forwarding their mail to outside services, a discouraged-but-within-policy practice.

We used the mail migration as an opportune time to update many systems to the latest versions of their e-mail client and web browser. For consistency purposes we used the Firefox web browser to perform the migration, but in this process we found some users still were using Firefox 1.0, a long-outdated version.

3.3. MSD Pitfalls and Lessons Leaned

MSD IT, with the insight gained by the MCS migration experience, was able to create a more controlled migration process. Our biggest hurdle was sticking to the plan: specifically, scheduling each user, keeping track of migrations, and following through with all users. Adhering to this last step proved problematic, because, once we had all but a few the users migrated, we let other issues take priority and the last of the migrations took a back seat. Unlike MCS, we thought our e-mail server running with a light load would last awhile. Despite our migration going generally smoother than MCS's, we were not immune to the assumption that would be proven quite demonstrably wrong.

Of this handful of accounts on the old server, most were service accounts, not used by any particular user. However, we did have two user accounts left. One was a former division director who proved difficult to schedule. Since he was moving to Exchange, his migration required more coordination with CIS, as their Exchange administrators would need to assist in the migration. We also had a user we thought had been migrated to another division's e-mail server because he had been transferred to that division, but who turned out to still be using our old server. At the time we were getting ready to start migrating these account, our aging (and now unsupported) Sun server crashed in spring 2009. Since another division was involved, we combined efforts to bring the server back up. But the server had experienced nearly catastrophic failure; the data drives were intact, but we had no access to them without spending considerable time and money.

Fortunately, the former division director had a local cached copy of most of his e-mail, and we were able to use this for the migration. Unfortunately, the other user accidently deleted his locally cached copy, and we were unable to recover all of his older e-mail. We are still exploring our options for recovery, but the server is still offline. We quickly recreated most of the service accounts, but we are still finding some as we continue to review mail logs.

We've learned to follow through on our tasks and see them to completion. Also, we will do a better job confirming that work we think is done *actually is done*. Moreover, documentation can be improved, and properly documenting which service accounts we've created and what they're used for will help us a great deal down the road.

3.4. CIS Challenges and Participation in the MSD Migration

From the CIS perspective, the MSD migration was much more straightforward than the MCS migration. MSD engaged CIS early in their process. Based on experience gained from the MCS migration, and new features available in Zimbra that MCS helped explore and test, CIS was able to work with MSD to create a migration plan that worked well for them and minimized the impact on the Zimbra service and on MSD by spreading the migration out over time.

Both MCS and MSD handled their own migrations, engaging CIS when necessary. After the initial planning phases, the MSD migration was much more hands off for CIS. The one exception was the attachment indexing issue mentioned above.

CIS imposes no limits on mailboxes in our Exchange and Zimbra services and allows individual messages as large as 100 MB. Some of the components of mail systems work well with smallish messages but exhibit strain when processing large messages. At the time of the MSD migration, the attachment indexing process was a multithreaded Java process that had issues handling large attachment sizes. The net result was a dramatic increase in load on the system, both for CPU and disk, resulting in the Zimbra server being so slow it was almost unusable. Upon identifying the offending process, we disabled attachment indexing via a simple check box in the Zimbra admin GUI, and migrations were able to resume. We note, for Zimbra's sake, that there is a new facility that can be selected for attachment indexing that is proving to better handle large attachments, and is resulting in a consistently lower system load.

4. Conclusions

Hindsight is, of course, 20/20, and one can easily look at both migrations and conclude that it's obvious what to do and what to avoid. Of course, every situation is different, and a careful examination of what went wrong and *why* can often lead to insights on how to avoid similar pitfalls when one is pushed down a similar path. In this section, we look at what each of the divisions took away from the process, having seen the results from each other's migration.

4.1. MCS

In many ways, performing an e-mail migration like this is not unlike performing a number of other types of migrations in the IT world, whether it's physically moving a datacenter, or implementing a new network topology, or deploying a new authentication scheme. In other ways, however, they can be vastly different, and it's in recognizing these differences that we can make better choices. Outside influences, customer demands, and occasionally the laws of physics can get in the way of how we expect things to play out.

MCS would obviously opt for a more measured approach in future migrations. The plan employed by MSD holds great appeal; however, two important factors exist. First, this option was not available on the version of Zimbra the lab was running at the time of our migration. Second, testing on our old mail server indicated that this implementation would not have worked for much the same reason imapsync failed; an aging server combined with enormous mailboxes results in timeouts and dropped connections.

Instead, time permitting, a well-documented and user-driven migration would be our likely course of action when undertaking a migration of this size. As in the prior-cited Tenwen paper, we would build the new system separate from the old one, move the users' delivery to the new system, and help them move their old data to it on their own schedule, within the constraints of our ability to maintain and run that old system. After a well-publicized and finite period of time, we would decommission the old system [Evard94].

As a service organization, it is always an admirable goal to inconvenience one's users as little as possible, but there are situations, such as this, where it's simply not attainable. A side benefit of a user-driven migration is an increased likelihood that users will be more selective as to which data must be maintained – our users can be notoriously bad at pruning unneeded data, resulting in just the sort of bloat that led to some of the issues we faced.

However, time is not always flexible, and when faced with an immovable deadline, one sometimes has no alternative but to jump in with both feet and try to solve the problem to the best of one's ability. If one absolutely had to do a migration like this, our implementation plan could have worked with better parameters, though it would by no means be the preferred solution. Certainly, a longer outage window and fewer false starts would have helped, but significant user input would still be required because of the corruption in the data being moved. Aggressive scanning of the mailboxes using IMAP tools could have identified these problems well in advance and allowed us to repair or remove the troublesome data well in advance. Likewise, we could have front-loaded the heavy work by migrating the heaviest users first, rather than the easily scriptable alphabetical method. Indeed, when it became evident that certain users had disproportionately large mailboxes, we hand-started syncs on their mailboxes outside the automated process.

We note that in no way were the pitfalls and encumbrances the fault of the targeted mail server software or the server itself. We believe we would have faced these challenges regardless of the chosen path, largely because of the age of the existing mail server, and its inability to handle the volume of mail we were moving.

4.2. MSD

MSD's biggest issue was with actually completing the project. This left us with several loose ends we needed to deal with in crisis mode when the Sun server crashed, as opposed to a controlled shutdown of the old server.

The server crash notwithstanding, MSD would definitely use the same basic method again if faced with another similar migration, albeit with better follow-through. This user-centric migration allowed a lot of buy-in from the most important IT customer – the end user. It reduced the potential lost productivity of the scientist if a one-shot migration had been done. It was labor intensive for MSD IT Operations, but the benefit of reaching out to the user on an individual basis reduced call volume and follow-up issues. Also, we were able to resolve most issues in a timely manner, instead of trying to deal with several dozen users at once.

4.3. Avoiding Disaster

Many papers have been written describing IT moves, including the already cited [Evard94, Limoncelli97], as well as [Schimmel93, Cha98], dealing with moves and migrations both physical and virtual. Every move is different; each comes with its own pitfalls. Every time a group undertakes a project of such magnitude, there exists the opportunity to achieve both fantastic successes and extraordinary failures. The right steps taken beforehand can tip the scales more in favor of the former. Included in the appendices is the premigration checklist that we can now construct from our experiences, and would have dearly loved to have read prior to beginning the project.

Author Biographies

Craig Stacey is a full time computer geek, part time stand-up comic, aspiring photographer and writer, passionate beer enthusiast, and frequent wearer of pants. He is also the IT manager for the Mathematics and Computer Science Division at Argonne National Laboratory and longs to spend more time doing system administration and less time doing paperwork. His e-mail address is stace@mcs.anl.gov, and he is fond of monkeys and robots.

Adam Max Trefonides has been a UNIX Systems Administrator for many years. Prior to holding his current position as a senior systems administrator in the Mathematics and Computer Science Division at Argonne National Lab he was responsible for the team that, among many other duties, took care of the central e-mail systems at the University of Chicago, (in other words e-mail was his fault). Prior to working for the computers he was a cross-country trucker, carpenter, welder, sculptor and unemployment recipient. He maintains his trucker license for when the Internet fad ends. His e-mail address is maxadam@mcs.anl.gov.

Tim Kendall is a systems administrator and the primary Mac specialist in the Materials Science Division at Argonne National Laboratory. He loves Science Fiction of all types and was a professional photographer for 18 years before switching to IT. He helps run the Two Way Street Coffee House that has been in operation since 1970 presenting live folk music every Friday night. His e-mail address is tkendall@anl.gov.

Brian Elliott Finley is the deputy manager of Unix, storage, and operations for the Computing and Information Systems division at Argonne National Laboratory and is the lead on the Argonne Zimbra project. He holds a number of technical certifications and has created, maintained, or otherwise contributed to several open source software projects, including SystemImager and WiFi Radar. Mr. Finley lives in Naperville, IL, US with his wife, four children, one large dog, and a toad. He can be reached at finley@anl.gov.

Acknowledgments

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

[Cha98] Lloyd Cha et al., “What to Do When the Lease Expires: A Moving Experience,” in Proceedings of the Twelfth Systems Administration Conference (LISA ’98), pp. 168-174, Boston, MA, 1998

[Evard94] Rémy Evard, “Tenwen: The Re-engineering of a Computing Environment,” in 1994 LISA Proceedings, pp. 37-46, San Diego, CA, 1994

[Lamiral] **imapsync**, Gilles Lamiral (developer), http://www.linux-france.org/prj/imapsync/

[Limoncelli97] Tom Limoncelli, “Creating a Network for Lucent Bell Labs Research South,” in 11th Systems Administration Conference (LISA '97) Proceedings, pp. 123-140, San Diego, CA, 1997

[Schimmel93] John Schimmel, “A Case Study on Moves and Mergers”, in Seventh System Administration Conference (LISA ’93), pp. 93-98, Monteray, CA, 1993

[Zimbra] Zimbra Wiki, “Mail Migration instructions,” http://wiki.zimbra.com/index.php?title=Mail_Migration

Appendix: Suggested Premigration checklist

As noted in Section 4.3, this is the checklist MCS should have used, constructed from the experiences gained from not using such a checklist.

Two months prior to migration

1. Inform users of the migration plan. Encourage data clean-up. Make clear and obvious the date the new service will begin.
2. Ensure user mailboxes are free of corruption. Aggressively scan mailboxes for errors using IMAP protocols. Instruct users on methods to test for problem mailboxes, including deleting problem messages.
3. Archive inactive mailboxes, and take them offline.
4. Compare list of active mailboxes with log files to identify users who are not logging in to check mail. Flag potentially inactive accounts, attempt to notify owners.
5. Identify exceptionally large mailboxes and work with owners to identify actual user needs and expectations – perhaps the mail client is configured to never empty the trash, for example.

One month prior to migration

6. Repeat items 1 through 5.
7. Go over potentially inactive account list from step 4, identify those actually inactive (eg, owner unreachable), and archive them.
8. Identify all accounts to be migrated, and create them on new server.
9. Ensure new account creation process is creating mailboxes on existing server and new server.
10. Hold training session with users demonstrating migration procedure.

One week prior to migration

11. Repeat items 1 through 5.
12. Ensure all accounts to be migrated are ready for service.
13. Hold another training session demonstrating migration procedure.
14. Ensure adequate availability for IT staff on migration day and the days that follow.
15. Post mail client configuration instructions so users can be ready for the switch. Adjust centrally managed mail client configurations.

One day prior to migration

16. Reiterate new service date very publicly. Post signs, and website announcements, send e-mails.
17. Ensure configuration instructions for mail clients are trivially available, trivially locatable, and correct.
18. Re-ensure IT staff availability.

Migration day

19. Buy lunch for the IT staff.
20. Implement migration plan.

Appendix: MCS Migration Scripts and Configuration Files

imapsyncbatch.sh - *used to launch imap sync sessions between cliff and Zimbra, this file lived on a third host named "owney" as cliff's SSL implementation was too old to open encrypted IMAP sessions to the Zimbra server. This is the version that contains the errant "-- delete2" that resulted in deletions from the Zimbra folders. stage1.mcs.anl.gov was the temporary hostname for the Zimbra mailboxes during migration.*

```
#!/bin/bash

USER1="zzzzzzzz"
USER2=$1@stage1.mcs.anl.gov
HOST1=cliff.mcs.anl.gov
HOST2=zimbra.anl.gov
DATE=`date "+%Y-%m-%d_%H:%M:%S"`
EXCLUDE="Trash|Viral"
SPLIT1=20
PASS1=/root/migration_scripts/cpass
PASS2=/root/migration_scripts/zpass
logfile=/sandbox/zzzzzzzz/log/$1-imapsync.log
userlog=/sandbox/zzzzzzzz/log/imapsync.log


cd /sandbox/zzzzzzzz/tmp
echo `pwd` >> $logfile
## Begin IMAPSync
echo "" >> $logfile
echo "------------------------------------" >> $logfile
echo "IMAPSync started for $1   $DATE" >> $logfile
echo "" >> $userlog
echo "------------------------------------" >> $userlog
echo "IMAPSync started for $1   $DATE" >> $userlog
echo "Settings: Excluding: $EXCLUDE, $SPLIT1 messages per" >> $logfile
echo "" >> $logfile

                echo "Starting $USER2 at $DATE" >> $logfile
echo "" >> $logfile
                imapsync \
                --nosyncacls --syncinternaldates \
                --nofoldersizes \
                --split1 $SPLIT1 \
                --exclude $EXCLUDE \
                --host1 $HOST1 \
                --user1 $USER1 \
                --passfile1 $PASS1 \
                --port1 993 \
                --host2 $HOST2 \
                --user2 $USER2 \
                --passfile2 $PASS2 \
                --port2 993 \
                --ssl1 \
                --ssl2 \
                --noauthmd5 \
                --delete2 \
                --buffersize 8192000 \
                --regextrans2 's/^Journal$/Journal-old/i' \
                --regextrans2 's/^Briefcase$/Briefcase-old/i' \
                --regextrans2 's/^Calendar$/Calendar-old/i' \
                --regextrans2 's/^Contacts$/Contacts-old/i' \
                --regextrans2 's/^Notes$/Notes-old/i' \
                >> $logfile
                echo "$DATE Finished $USER2" >> $logfile
                echo "" >> $logfile
# need some sanity checks here?

echo "" >> $logfile
echo "IMAPSync Finished for $1  $DATE" >> $logfile
echo "------------------------------------" >> $logfile
echo "" >> $userlog
echo "------------------------------------" >> $userlog
echo "IMAPSync Finished for $1  $DATE" >> $userlog
```

linker-forward.sh - *used to create /var/imap/mailboxes file on cliff with ghost users. This version traverses the alphabet from a to z, linking the user being synced with the ghost user "aaaaaaaa." The script needed to maintain the sorting and whitespaces contained within the existing file. As noted at the bottom, this script directly calls the above "imapsyncbatch.sh" on owney via an SSH session. The end of that SSH session allows this script to increment to the next user. A similar script, linker-reverse.sh, performed a similar job, albeit from z to a, linking the user being synced to the "zzzzzzzz" ghost user.*

```
#!/bin/ksh -x

## /root/migration_scripts/linker-forward.sh
## created by maxadam@mcs.anl.gov 3/2008
## modified by stace@mcs.anl.gov 4/2008
## with input from many quarters
##
## This script prepares cliff for migrating a user to zimbra.
## It is designed to work in tandem with linker-reverse.sh,
## to add parallelprocessing.
## What it does:
## Generates the userlist
## Moves a link to a commented version of /etc/inetd.conf in
## place and refreshes imapd in order to halt any new imap
## connections.
## Cleans the aaaaaaaa user out of the /var/imap/mailboxes file
## and copies the file to a working copy
## Creates the symlink for the aaaaaaaa user that points to the
## mail directory
## Backs up the mailboxes file, appending the current username
## Copies the modified mailboxes file into place
## Re-enables imap
## Runs imapsyncbatch on owney with $user as the single argument
## over ssh

log=/var/log/linker-forward.log
lock=/root/migration_scripts/locked
if [ ! -f $log ]; then
```

```
        touch $log
fi
 for i in `grep user /var/imap/mailboxes | awk '{print $1}' | awk -F . '{print $2}'| sort -u |
egrep -v ^aaaaaaaa | egrep -v ^zzzzzzzz` ; do
  while [ -f $lock ]; do
    sleep 20
  done
  touch $lock
  inetdpid=`ps -ef | grep '[i]netd' | awk '{ print $2 }'`
  echo "`date "+%Y-%h-%d@%H:%M:%S"`     Linking mailboxes for user ${i} to zzzzzzzz" >> $log
  if  [ ! -f "/etc/inetd.conf.off" ] 2>&1 >> $log; then
        echo "/etc/inetd.conf.off does not exist or is not an ordinary file! exiting." >> $log
        exit 1
  elif  [ ! -f "/etc/inetd.conf.on" ] 2>&1 >> $log; then
        echo "/etc/inetd.conf.on does not exist or is not an ordinary file! exiting." >> $log
        exit 1
  elif [ ! -L "/etc/inetd.conf" ] 2>&1 >> $log; then
        echo "/etc/inetd.conf is not a symlink or does not exist! Exiting." >> $log
        exit 2
  else echo "`date "+%Y-%h-%d@%H:%M:%S"`        Halting imapd" >> $log
        rm /etc/inetd.conf
        ln -sf  /etc/inetd.conf.off /etc/inetd.conf
        kill -HUP $inetdpid
        echo "`date "+%Y-%h-%d@%H:%M:%S"`       imapd halted" >> $log
        cp /var/imap/mailboxes /var/imap/mailboxes.backup-forward
  fi
  echo "`date "+%Y-%h-%d@%H:%M:%S"`       Making links for ${i}" >> $log
  egrep -v ^user.zzzzzzzz /var/imap/mailboxes  > /var/imap/mailboxes-f.${i}
  egrep "default        ${i}    " /var/imap/mailboxes | \
       sed s/^user.${i}/user.zzzzzzzz/ | \
       sed s/"default   ${i}    "/"default      zzzzzzzz        "/ >> /var/imap/mailboxes-f.${i}
  if [ ! -s /var/imap/mailboxes-f.${i} ] ; then
        echo "`date "+%Y-%h-%d@%H:%M:%S"`        Abort, empty mailboxes file" >> $log
        rm /etc/inetd.conf
        ln -sf /etc/inetd.conf.on /etc/inetd.conf
        kill -HUP $inetdpid
        exit 3
  fi
  rm -f /var/spool/imap/user/zzzzzzzz
  ln -sf /var/spool/imap/user/${i} \
         /var/spool/imap/user/zzzzzzzz
  if ! /bin/ls -l /var/spool/imap/user/zzzzzzzz | grep ${i} 2>&1 >> $log ; then
       echo "`date "+%Y-%h-%d@%H:%M:%S"`        Abort, link bad" >> $log
        rm /etc/inetd.conf
        ln -sf /etc/inetd.conf.on /etc/inetd.conf
        kill -HUP $inetdpid
       exit 4
  fi
  echo "`date "+%Y-%h-%d@%H:%M:%S"`       Links made" >> $log
  echo "`date "+%Y-%h-%d@%H:%M:%S"`       Copying mailboxes-f.${i} to mailboxes" >> $log
  if [ -s /var/imap/mailboxes-f.${i} ] ; then
         cp /var/imap/mailboxes-f.${i} /var/imap/mailboxes
  else
        echo "`date "+%Y-%h-%d@%H:%M:%S"`        Abort, empty mailboxes file" >> $log
```

```
      rm /etc/inetd.conf
      ln -sf /etc/inetd.conf.on /etc/inetd.conf
      kill -HUP $inetdpid
      exit 5
  fi
  echo "`date "+%Y-%h-%d@%H:%M:%S"`       Attempting to restart imapd" >> $log
  if  [ ! -f "/etc/inetd.conf.off" ] 2>&1 >> $log; then
      echo "/etc/inetd.conf.off does not exist or is not an ordinary file! exiting." >> $log
      exit 1
  elif  [ ! -f "/etc/inetd.conf.on" ] 2>&1 >> $log; then
      echo "/etc/inetd.conf.on does not exist or is not an ordinary file! exiting." >> $log
      exit 1
  elif [ ! -L "/etc/inetd.conf" ] 2>&1 >> $log; then
      echo "/etc/inetd.conf is not a symlink or does not exist! Exiting." >> $log
      exit 2
  else echo "`date "+%Y-%h-%d@%H:%M:%S"`       Restarting imapd" >> $log
  rm /etc/inetd.conf
  ln -sf /etc/inetd.conf.on /etc/inetd.conf
      kill -HUP $inetdpid
  echo "`date "+%Y-%h-%d@%H:%M:%S"`       imapd restarted" >> $log
  fi
  sleep 1
  echo "`date "+%Y-%h-%d@%H:%M:%S"`       Starting imapsyncbatch for ${i} on owney" >> $log
  rm $lock
  ssh -t zzzzzzzz@owney.mcs.anl.gov /root/migration_scripts/imapsyncbatch.sh ${i}
done
```

/var/imap/mailboxes snippet - *head and tail of the /var/imap/mailboxes generated by the scripts above. Recall that, at the filesystem level, the ghost users' spool directories would be symlinks to the actual users' directories.*

```
user.aaaaaaaa   default aaaaaaaa        lrswipcda
user.aaaaaaaa.Quarantine        default aaaaaaaa        lrswipcda
user.aaaaaaaa.SPAM      default aaaaaaaa        lrswipcda
user.aaaaaaaa.Viral     default aaaaaaaa        lrswipcda
user.aaaaaaaa.sent-mail default aaaaaaaa        lrswipcda
user.aammar     default aammar  lrswipcda
user.aammar.Drafts      default aammar  lrswipcda
[…]
user.zzhang     default zzhang  lrswipcda
user.zzhang.Drafts      default zzhang  lrswipcda
user.zzhang.Quarantine  default zzhang  lrswipcda
user.zzhang.SPAM        default zzhang  lrswipcda
user.zzhang.Trash       default zzhang  lrswipcda
user.zzhang.Viral       default zzhang  lrswipcda
user.zzhang.sent-mail   default zzhang  lrswipcda
user.zzzzzzzz   default zzzzzzzz        lrswipcda
user.zzzzzzzz.Quarantine        default zzzzzzzz        lrswipcda
user.zzzzzzzz.SPAM      default zzzzzzzz        lrswipcda
user.zzzzzzzz.Viral     default zzzzzzzz        lrswipcda
user.zzzzzzzz.sent-mail default zzzzzzzz        lrswipcda
```

Appendix: Mail Routing Diagrams

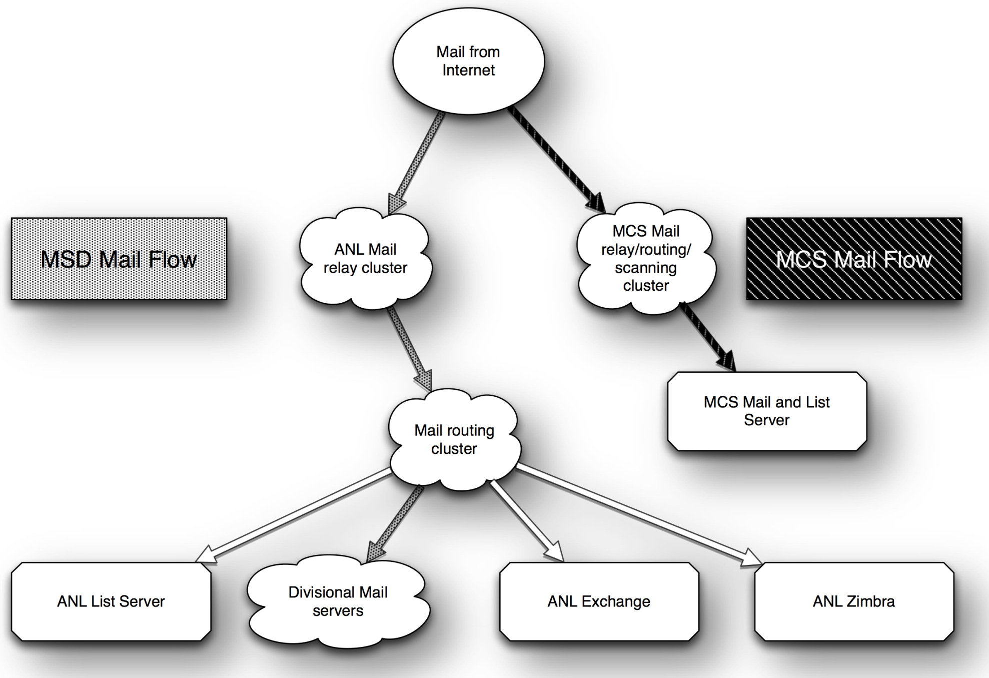

Figure 1 - Mail flow before migration project

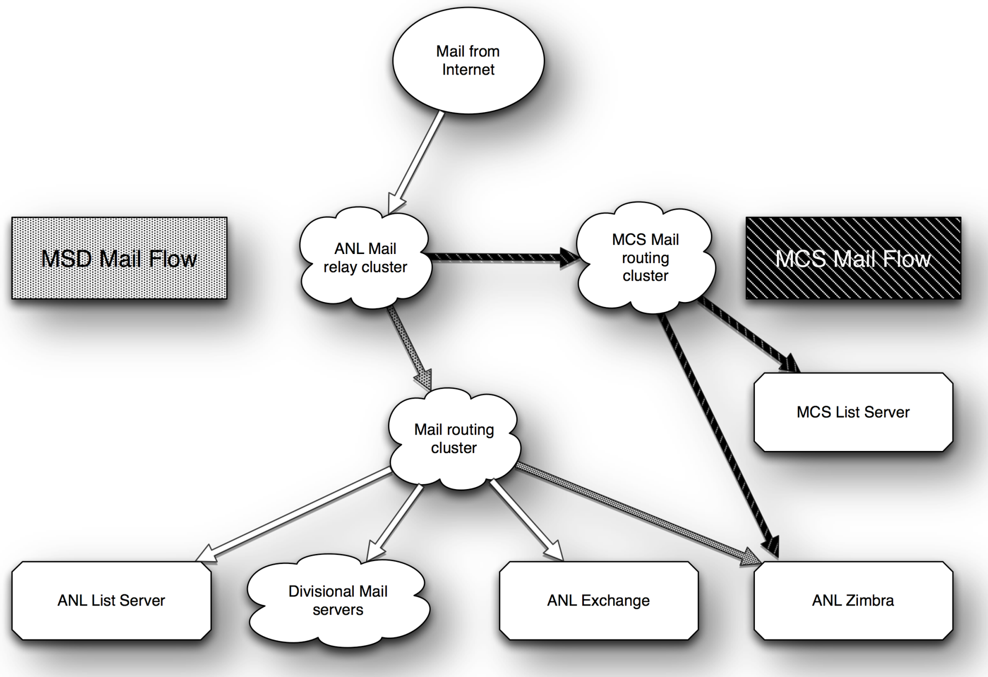

Figure 2 - Mail flow after migration project