techfocus

Unix-to-Linux Migration

Unix-to-Linux Migration

A step-by-step approach for data center managers that covers everything from making the business case to getting the best training.

BY KEN MILBERG

INSIDE

Making the Business Case for Migrating from Unix to Linux

WHEN MIGRATING FROM Unix to Linux, the most important case you will need to make is not a technical case but a business case. It's all about the bottom line. How will the business benefit by moving over? What is the total cost of ownership and return on investment for the migration?

The tricky part is choosing the methods you use to extrapolate this information and build your case. A Unix-to-Linux migration may seem like a no-brainer to data center managers, but the people you need to convince don't work in the data center. So you need to succeed in justifying your case.

It's a given in today's economic climate: You simply will not be able to embark on a major migration project without that business case. As part of building your case, you'll need to discuss the limitations of Unix and the features of Linux. You'll also need to elaborate on the justifications for migration and the actual cost of the project. Again, it should all relate to the bottom line.

UNIX CHALLENGES AND LIMITATIONS

Unquestionably, Unix is a mature OS, much more so than Linux. The top Unix versions today are AIX, Solaris and HP-UX. So why would you even want to consider moving to Linux? There are lots of reasons. Here are a few:

- **Vendor lock-in:** You've been running Solaris for 10 years now and are happy with the OS but not with Sun Microsystems' hardware and would prefer to run it on IBM Power Systems. Forget it. If you're running Solaris, you have to stay on Sun hardware unless you want to run Solaris on x86 machines. Similar stories if you prefer AIX or HP-UX.

- **Cost:** Linux is not really free, but there is no doubt you will pay more to a Unix

vendor. Don't look at the license costs alone. Look at the whole package—the cost of the OS maintenance, hardware and ancillary software licenses. You will find everything costs just a little bit more for Unix than Linux.

■ **Fixes, patches and more:** With Unix, you are generally held hostage by your vendor's timetable for releasing that patch or fix. Do you have a crackerjack engineering department that would like to do some work on the kernel? If you're running Unix, forget it.

MAKING THE BUSINESS CASE FOR MIGRATING FROM UNIX TO LINUX

PORTING APPLICATIONS OVER TO LINUX

OVERCOMING COMMON UNIX-TO-LINUX MIGRATION PITFALLS

TRAINING STAFF TO MANAGE LINUX ENVIRONMENTS

■ **Human capital:** Linux administrators earn substantially less than their Unix counterparts. As a systems person I may not be happy about this, but that is the reality. Smart business analysts understand this and factor the cost into their equation.

And have you checked the latest Gartner Inc. reports? Unix continues to decrease in overall market share, while Linux is constantly gaining market share. Don't minimize the importance of that. Everyone watches those numbers, from ISVs to hardware manufacturers to end users. Why do you think hardware vendors like IBM support Linux on their platforms? Is it because they love the competition? No. It's because even though they may love their versions of Unix, they recognize that the future is with Linux.

LINUX ATTRIBUTES THAT BENEFIT BUSINESSES

So what is it about Linux that makes it so desirable? Take a look:

■ **Platform flexibility:** Linux can run on anything, from commodity-based x86 servers to Unix vendors' hardware, such as IBM Power Systems, HP Itanium, x64 AMD Opteron-based systems and Sun Ultra Sparc and even IBM mainframes. No other OS can match the flexibility of platforms that Linux offers, which means that your business has the flexibility to change platforms as it requires.

■ **Deployment of patches:** Traditional development cycles can be endless with commercial vendors. Coming from the open source world, Linux is built to be modular. The software is accessible so you can fix it yourself, or community developers can come up with a patch quickly. With Unix, you're usually locked into your vendor's hardware and that leaves you with fewer options to save money. No other OS can match the flexibility that Linux offers. And flexibility can translate into cost savings.

■ **Compatibility and cost:** Ever try to price out an application on Linux and then the same application on Unix? Invariably the Linux license comes out cheaper. And although it's possible to run open source applications on Unix, open source and Linux go hand in hand like peanut butter and jelly. If availability is important, you cannot go wrong with Linux.

You only need to look at HP's decision several years ago to stop support for its Tru64 version of Unix. People are still angry about that today because Tru64 support for new technologies is no longer available. With Linux, you don't have to worry about being caught up in a vendor's decision or restructuring. Even if one of your Linux vendors goes south, the kernel itself is maintained by Linux.

EVALUATING THE ROI OF MIGRATION

Migrating from any system to any other system will cost money, no question about it. What makes a migration project worth performing is its return on investment (ROI). The ROI is calculated with this metric:

ROI = Cost savings * 100 / investment (or TCO).

With Linux, make sure you measure the cost of the acquiring the hardware as well as the costs of software, migration, training, maintenance and day-to-day operation. Also remember that certain costs are one-time only. Staying with your

current systems may include upgrades to hardware and software. So when considering a migration, compare it not only to what your costs are today but also to what it will take to get up to speed for a modern Linux deployment.

To calculate the ROI, determine the initial cost of the project, which should include the cost of development, hardware, migration, maintenance and support. This includes human capital. At the end of the day, you'll try to project the annual savings that the company will have when you are ready to go into production. Usually you'll want to do this over a three-year period.

You can choose from several methods to calculate long-term cost. The one I like to use is Net Present Value (NPV), which is the present value minus the initial cash outlay. Most finance professionals use this method. There are a lot of numbers to crunch in the NPV method, but you must do this to show your business that it really makes financial sense to migrate from Unix to Linux.

At the end of the day, you'll try to project the annual savings that the company will have when you are ready to go into production.

When quantifying the TCO, break it down into one-time costs and ongoing costs. One-time costs include software licenses, hardware, consulting costs and training. Ongoing costs include hardware maintenance, software maintenance, license maintenance fees, software upgrades and ongoing training.

One of the most compelling reasons to migrate is the annual savings on database licenses alone. Usually when you migrate, you also consolidate. That enables you to run with fewer CPU cores, which can dramatically increase cost savings. ■

Porting Applications over to Linux

AFTER MAKING THE business case for Linux and getting approval, you are ready for the actual porting process. Like any migration, there will likely be pain points that go with the process. Proper planning will go a long way to ensure success and ease those pain points.

The most important aspect of migration is the initial assessment and discovery. Everything from hardware to software to operating system versions to patch levels to application versions must be carefully researched and documented. When porting your applications to Linux, you will need all this information for developing your project plan.

If an application you are looking to migrate to Linux is commercially available, check to see if the vendor supports Linux. Do the same with your database. All popular databases today run on Linux, but make sure you are not running some obscure system whose vendor is no longer in business.

It's also possible that you're running an old version of a modern database like Sybase, which may not be supported. In that case, you can still move to Linux, but your migration will be more difficult. Better to find out the bad news now rather than three months into your project.

You'll also need to decide which distribution you will use. With Linux, there is a range of choices. Generally speaking, in an enterprise environment you can't go wrong with either Red Hat or SUSE using the RHEL and SLES enterprise versions. If your application is homegrown, work with your in-house development team to ensure that they will be able to migrate to Linux without doing an entire rewrite of the code.

Furthermore, you'll need to determine your hardware platform. Will you be running this environment on a clustered group of x86 computers or an HP Itanium? Work with your architectural team to determine the right platform for your code. It's likely that it will be similar to the Unix system you had been using. For example, if your version of Unix was Solaris running on an x86 machine, you

won't be moving it to an IBM mainframe. On the other hand, if you're looking to do a major server consolidation around IBM's System z mainframe architecture, you could be moving away from all your clustered x86 boxes and midrange Unix servers to that one platform. Linux is all about choices, and they are choices that you have with no other OS.

Sometimes the decision is more complicated. If you're a mixed shop and have a fair amount of AIX—which runs on IBM Power Systems—and you want to port a specific application to Linux, it might make the most sense to run Linux on those IBM Power Systems. In this case, you're keeping the same hardware platform, and you might not even have to purchase additional hardware because you can just create another logical partition on that same physical server.

Understand all of the components of each application, and try to break down the components into smaller modular components.

What else needs to be done? You have to identify the project team—developers, architects, coders and administrators. After you receive funding and the project has been formally approved, you'll officially start with a kick-off meeting.

The application assessment is really important, and it is a little different than the infrastructure/server assessment. Find out everything you can about each application—dependencies, hard-coded IP addresses and service-level agreements, among other things.

Find out whether each application functions on product-specific frameworks, such as IBM WebSphere. Understand all of the components of each application, and try to break down the components into smaller modular components.

If possible, find out how many lines of code are in the software and if it uses Unix pipes, message queues, shared memory signals or semaphores. Although these can be ported to Linux, make sure the Linux environment replicates the existing environment as much as possible.

Is any application multithreaded? Depending on your source platform, the com-

plexity of keeping it multithreaded may be high. Make the application review an important part of your process. The more information you find out early on, the easier things will go for you down the road.

IMPLEMENTING THE MIGRATION

The next step in migration is getting a sandbox environment in which to play. Your entire team may not have experience with Linux, and having an environment in which to learn—without fear of breaking anything—is invaluable.

Let's talk code and compilers. Are you running Java or C? Are there any third-party tools that will need to come over? Can they be migrated to Linux?

Let's assume you're using C. Let's also assume you will need to compile some code to move over to Linux. Use the GNU (gcc) compiler because this is the industry standard, and it's the native compiler for Linux. Applications compiled on other platforms will need to be recompiled.

You can choose between two methods of going about your compiled business. One method is recompiling your data on your existing environment, in which case you'll need to make sure you have all the required tools on the box, which includes the source code and makefiles. If you're considering this method, do this only in your test environment and never in your production environment.

The other choice is moving all your data and code to the new box and testing it all from that prototype environment. Think about the hardware platform as well. If you plan on moving the hardware platform, there may be some hardware-specific codes that will certainly trip you up and, in a worst-case scenario, force you to rewrite all your code.

Make sure your developers are part of the process, and don't assume anything. Considerations include runtime APIs, system calls, streams and library support.

Be sure you fully understand the scope of what you're looking to port. This is where your assessment comes in and where you've identified everything about the application and its libraries and dependencies. This way you can quickly ascertain whether the products are available on Linux and where to find them.

There is no question that applications written in Java tend to port more quickly than applications in C. In addition to your applications, you'll need to identify the test environment, user interface requirements, platform-dependent constraints, middleware products and internal skill levels in porting. Each of these areas has some risk.

UPDATING APPLICATIONS

The application piece is crucial in the migration process. In some cases, your applications may already be ported, and there is little to do there. In other cases, you will have to recompile them completely on your new platform. Porting the software may be as easy as doing the recompile and then running validation tests to confirm everything is OK.

The porting process for your apps should include development and testing. When you're migrating your box, make sure you've decided on a method for migrating your database over. Applications that require kernel extensions and device drivers are not easy candidates to support, partly because most kernel APIs do not follow any stringent standards.

Does the application use third-party software components, such as database tools, application servers or other middleware? These will add to the complexity of the migration. Is the application 32-bit or 64-bit? If you're moving from 32-bit to 64-bit, you will need additional time to port. How does your application communicate with the database? Does it use database interfaces such as ODBC or programming languages like C++? These are all aspects you will need to consider. From a staffing standpoint, try to bring in personnel who have experience with these kinds of migration projects.

GAUGING STABILITY AND PERFORMANCE

Application issues are usually discovered within the first several weeks. That's when engineers get their first inkling about what they are up against. That's also

when you may want to revisit your project plan to adjust any delivery dates.

Testing is critical for stability, functionality and performance. Don't spend $2 million on developing new systems and only $2,000 on testing. The sequence of testing usually works this way:

- Porting engineers do unit testing for applications they are porting.
- Application engineers perform functional testing.
- User acceptance testing, or UAT, takes place. This is where the actual business people do the testing.
- Performance engineers conduct performance testing.

During testing, it's essential that the ported application is running against stress tests to ensure that the system can handle the load. You should have already done baseline testing, which takes a snapshot of performance on the source environment—your current production environment.

Your goal is to run similar tests from a stability and performance standpoint. Try to break your systems. Use tools such as Hewlett-Packard's LoadRunner to simulate five times the amount of normal activity.

Testing is critical for stability, functionality and performance. Don't spend $2 million on developing new systems and only $2,000 on testing.

Now is the time to uncover bugs—not when you are in production. It's your opportunity to debug problems before they get into production. Take advantage of testing, and don't let your systems people be the only ones to verify that everything is working. The business side of the house must not only be involved, but it should also write the UAT plans. If your system had high availability on the old environment, make certain that is also tested. ■

Overcoming Common Unix-to-Linux Migration Pitfalls

AFTER SOME INITIAL testing, what are some of the major gotchas you should be looking at now? What kinds of errors or problems commonly arise during Unix-to-Linux migrations? Where should you turn for support? You can sidestep some pitfalls with a little vigilance and some good planning.

IDENTIFY POTENTIAL HARDWARE CHALLENGES

The biggest gotchas when considering Unix-to-Linux migrations are platform-dependent issues, especially when you're moving from RISC to x86 platforms. This is where the concept of *endianess* comes in.

Endianess refers to the ordering of memory used for data representation. It defines how a data element and its bytes are stored in memory. The problem you may encounter is that x86 computers use little endian, while RISC systems use a lot of endian. If you're moving from RISC to x86 or the reverse, porting the code requires that the code be modified. These issues must be uncovered during the assessment stage when you look for platform-dependent constraints.

Some vendors have developed innovative solutions to get around this very issue. For example, IBM developed PowerVM LX86, which is part of its midrange virtualization engine. It uses special software to automatically translate the instruction set to Power instructions so that they do not have to be compiled natively.

Although Linux could run on an IBM Power platform before, it had to be run natively and recompiled for the platform. This is no longer a problem. The translator, which is a part of PowerVM LX86, transforms the x86 Linux calls to Power Linux calls through a three-step process of decoding, optimization and generation of code. This lends itself well to Web applications that repeat similar instructions because frequently used code is cached in memory and does not need to be translated.

Another area to consider is applications that require kernel extensions and device drivers. These are not easy candidates to support, partly because most kernel APIs do not follow any stringent standards. API calls, the number of arguments and the process of loading them into kernel extensions will all function differently on the new platform.

GAUGE APPLICATION SUITABILITY AND AVAILABILITY FOR LINUX

Most commercial and Web applications are suitable to run on Linux. The availability piece is another story.

Although nearly all vendors today have moved their Unix applications to Linux, make sure that your off-the-shelf application has this support before considering a migration. If it doesn't, you do not want to be in the position of having to port that yourself. For applications developed in-house, you'll need a strong development team to help you migrate applications.

Ask colleagues who have done this before how well your applications are likely to move to Linux. Find out how they are running now. And don't be afraid to go straight to your vendor for help. Both Red Hat and SUSE offer programs to help with migration efforts.

DEPLOYMENT ERRORS OR PROBLEMS

What type of errors or problems might you see when doing ports? This is where proper testing is key. Anything can go wrong during porting, so establish a test environment and a lab that tries to break your systems before they are deployed into production.

Two years ago, after my group performed a major migration that appeared to go well, we started getting phones calls regarding the ability of the payroll systems to process checks. This was alarming because we had gone through extensive unit testing as well as user testing and UAT. As it turned out, the problem was not because of anything we did on the migration side. It was because of incompatibility

issues with some PC-based clients that were using an old version of an Oracle client.

This was an important part of our "lessons learned" document. In future migrations, I always made it a point to check the kind of clients that were accessing the server to ensure that this would not happen again.

MAKING THE BUSINESS CASE FOR MIGRATING FROM UNIX TO LINUX

Another problem is with Unix shell scripts. One would think that a Unix shell script that was written in Unix should work the same on Linux. This is not a correct assumption.

PORTING APPLICATIONS OVER TO LINUX

Any script in Unix that needs to interface with your application must be tested carefully. The standard shell for Linux is the bash shell, which is based on the original Unix Bourne shell. In our case, the Unix shell was the Korn shell. There were certain functions that did not come across correctly because of this miss. Always assume that your shell scripts will not work. Test each one that you have.

OVERCOMING COMMON UNIX-TO-LINUX MIGRATION PITFALLS

TRAINING STAFF TO MANAGE LINUX ENVIRONMENTS

GETTING SUPPORT FOR PROBLEMS

Support is dependent on the size of your IT department and the experience of your staff. Do you have several experienced Linux administrators who can pretty much do anything? Or are they mostly Unix administrators who have been trying to pick up Linux?

The level of support required correlates to the type of environment you have. Are your systems running CRM or payroll systems that can cause the company tens of thousands of dollars for every minute of downtime?

It's key to have vendor-specific support. Both Red Hat and Novell offer 24/7 support programs for their distributions. Some Unix hardware vendors, IBM for example, also offer their own support for Linux.

Get support from your hardware vendor, if possible. Moreover, it doesn't hurt to have OS support from both the vendor that supports your Linux distribution, such as Novell for SUSE, and your hardware vendor. Again, you need to take into account the financial impact of downtime on your organization as well as the experience of your staff. ■

Training Staff to Manage Linux Environments

SO YOU'VE JUMPED over some of the hurdles during the process. What about training? What about backup and high availability? How do you monitor performance and tune your systems? How do you ensure adequate skill sets and certifications for your staff?

SYSTEMS MANAGEMENT ACROSS UNIX AND LINUX

Don't let anyone convince you that Unix and Linux are mostly the same. There are many differences—some subtle and some not so subtle.

The three popular Unix flavors that dominate the market—AIX, HP-UX and Solaris—vary in so many ways. These differences include the kernel, performance monitoring and tuning commands, networking, configuration, virtualization and other processes.

With AIX, you have a dominant GUI, or SMIT, short for the system management interface tool. The other flavors of Unix have similar front ends as do the Linux distributions.

Although you can use the command line for most AIX tasks, it's actually recommended that some tasks be performed by SMIT, or else they won't save on a reboot. Networking commands such as **ifconfig** are a good example of this. On Linux, you can use **ifconfig**, and it will save on a reboot.

With AIX, you use certain commands to change kernel processes such as **vmo**, **ioo** and **schedo**. With Solaris, you use **/etc/system**. With Linux, you use **sysctl**.

If you want to view your run-level, with AIX and Solaris you type in **who –r**. With Linux, you use the **runlevel** command. When you want to check your swap space, with AIX you use the **lsps** command, and with Linux you use **swapon – s** or **free**.

When looking for filesystem information, with Solaris you use **/etc/vfstab**. With Linux, you use fstab, and with AIX it's **/etc/filesystems**.

These are just the basic commands. When you're working with logical volumes and filesystems, things can get that much more complex. To be effective, you need training.

Many vendors today offer custom classes specifically for Linux administrators who have come from Unix environments. Look into taking these types of classes from your hardware vendor.

There are also several advantages of staying on the same hardware platform when considering a Unix-to-Linux migration. Let's look at IBM's Power Systems, which can run either AIX or Linux distributions from Red Hat or Novell SUSE. Because the virtualization engine—PowerVM—on IBM Power Systems is the same regardless of whether you run AIX or Linux partitions on your server, managing virtualization and logical partitioning is the same across operating systems. Having to understand only one type of virtualization technology across several OS platforms is a huge plus.

Another plus to staying on one hardware platform is that you don't have to worry about endianess—a concept that can cause big problems when moving to different platforms. And you don't have to learn another architecture or retrain your hardware staff, assuming you have a large enough environment to have a staff.

BACKUP AND HIGH-AVAILABILITY TACTICS

How do you back up your systems now? In most large enterprise IT shops, you would probably use a third-party product like NetBackup or TSM. This kind of software offers both Unix and Linux clients.

If you're using a generic Unix tool to back up the systems portion of your data, you will need to look at equivalent tools for your Linux distribution because your Unix-specific tools will not work. I've even seen problems using generic Unix/Linux tools such as tar and cpio between the different platforms, so be careful.

As far as high availability is concerned, you may be in a position to use similar tools. For example, if you're using Veritas for High Availability on Solaris, Veritas

also comes with a Linux client. Same thing with IBM's PowerHA, formerly referred to as HACMP, which also has a Linux version in addition to the AIX version. The best bet is to use the tools that you have the most experience with, if at all possible.

MONITORING AND PERFORMANCE MEASUREMENT TACTICS

Performance monitoring is also different with Unix and Linux. The following table shows some of the differences among the performance tools for Unix and Linux as well as their purposes. Again, training is key.

Unix/Linux Performance Tools Matrix

OS	COMPREHENSIVE	RAM	CPU	I/O	NETWORK
AIX	nmon, topas, lparmon	vmstat, ssvmon	vmstat, sar	iostat, filemon	netstat, nfsstat
HP-UX	GlancePlus, MeasureWare/ Perfview, Caliper	vmstat	top	iostat	netstat
Tru-64	Collect, sys_check, HP insight manager	vmstat	top	iostat	netstat
RHEL	sysstat, systemTap, oprofile.	vmstat	top, mpstat	iostat	netstat, iperf
SLES	sysstat, SystemTap	vmstat	top, mpstat	iostat	netstat, iperf
Solaris	SE Toolkit, sysperfstat	vmstat	top	iostat	netstat

ENSURING ADEQUATE SKILL SETS OR CERTIFICATIONS

Linux certifications may have come a long way in recent years, but Red Hat still sets the bar with its Red Hat Certified Engineer, or RHCE, available since 1999. With the RHCE certification, you need to pass a full-day hands-on lab consisting of a written test, server install and network lab. Red Hat now has the Red Hat Certified Architect (RHCA) as well as the Red Hat Certified System Administrator (RHCSA), created to more closely align with the Linux administrator job role that is common in IT organizations.

Linux certifications may have come a long way in recent years, but Red Hat still sets the bar with its Red Hat Certified Engineer, or RHCE, available since 1999.

The Linux Professionals Institute (LPI) has the LPI certification, which is designed to be distribution-neutral following the Linux standard base and other related conventions. It is three-tiered: Level 1 is for junior administrators, Level 2 is for intermediates, and Level 3 is for more advanced engineers and administrators.

As a result of a new partnership between LPI and Novell, those Linux professionals who have earned their LPIC-1 status will also have satisfied the requirements for the Novell Certified Linux Administrator, or CLA, certification without any additional cost or exams. Another change on the Novell front is Novell's Certified Linux Desktop Administrator, or CLDA. Those who are new to Linux should start with this entry-level Linux desktop certification. It measures administration skills such as installing, configuring and managing Linux desktops.

LPI now has a partnership with CompTIA, which allows its exam record to be forwarded to LPI. Certification in CompTIA Linux+ Powered by LPI enables candidates to become certified in LPIC-1 as well, enabling further participation in the LPI program if the candidate chooses.

Getting the correct training and certifications for both you and your staff can go a long way in ensuring a successful Unix-to-Linux migration. It's better to think ahead and sign up now than to wish you did after the fact. ■

ABOUT THE AUTHOR:

Ken Milberg is a systems consultant with two decades of experience working with Unix and Linux systems. He is a SearchEnterpriseLinux.com Ask the Experts adviser and columnist as well as the founder and group leader of the NY Metro POWER-AIX/Linux Users Group. Through the years, Milberg has worked for both large and small organizations and has held diverse positions from CIO to Senior AIX Engineer.

SearchEnterpriseLinux.com

Cathy Gagne
Editorial Director
cgagne@techtarget.com

Matt Stansberry
Editorial Director
mstansberry@techtarget.com

Leah J. Rosin
Site Editor
lrosin@techtarget.com

Christine Casatelli
Editor
ccasatelli@techtarget.com

Linda Koury
Director of Online Design
lkoury@techtarget.com

Marc Laplante
Publisher
mlaplante@techtarget.com

Peter Larkin
Associate Publisher
plarkin@techtarget.com

TechTarget
275 Grove Street, Newton, MA 02466
www.techtarget.com

- **Red Hat Migration Center**
- **What's New? Red Hat Enterprise Linux 6!**
- **Why Migrate to Red Hat and IBM**

About Red Hat and IBM:
Red Hat is the most trusted open source provider and the most recognized Linux brand in the world. And IBM is the most widely recognized hardware provider globally. Together we give customers the confidence and ability to deploy a resilient infrastructure throughout the enterprise by providing the advantages of Linux, comprehensive services, solid platforms, and technology leadership. The entire IBM Systems product line is enabled for Red Hat Enterprise Linux, making it easy for your business to take advantage of the power of open standards. Red Hat offers enterprise customers security, stability, performance and overall a highly cost effective platform to migrate Unix based software onto IBM System x platforms. And don't just take our word for it, check out what our customers are saying: http://customers.redhat.com/category/partner/ibm/