Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

# UNIX To Linux Migration :
*Tips & Tricks*

**Seung-Do Yang, RHCA**
syang@redhat.com
*Sales Engineer*
**Red Hat Korea**

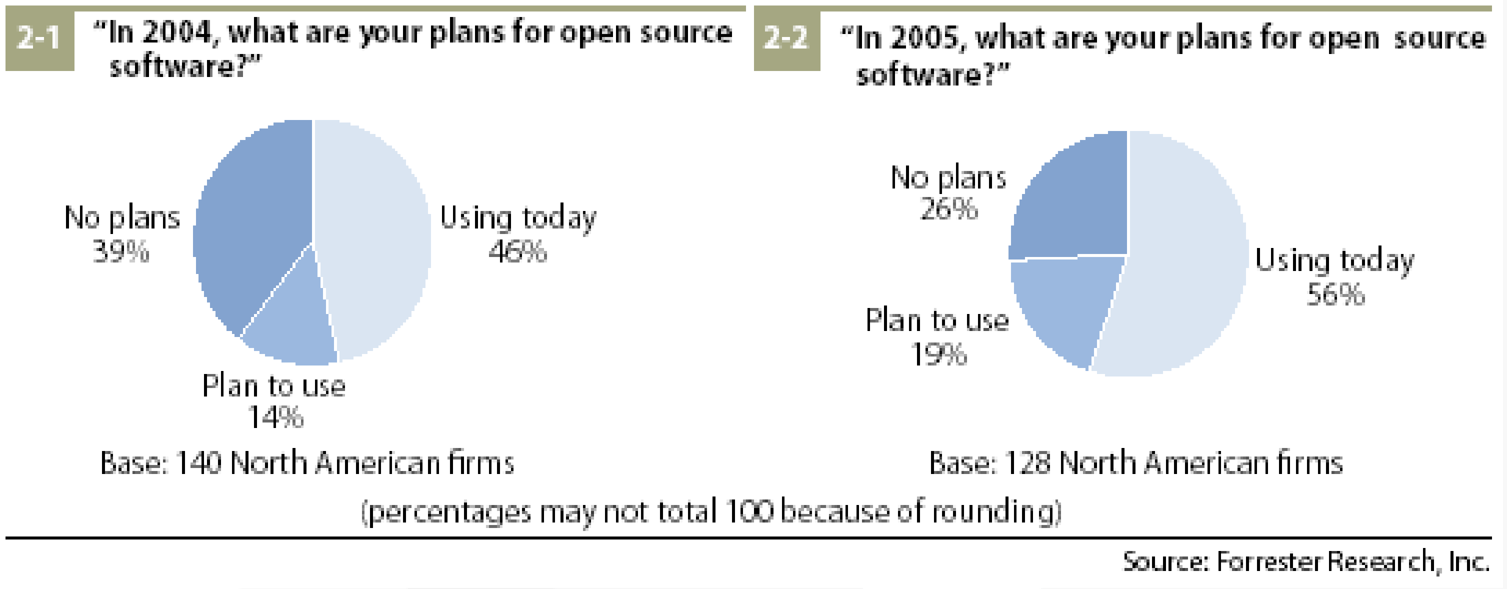
## Agenda

- Introduction
- Why migrate to RHEL 4?
- RHEL 4 migration
  - Migration targets
  - Best practices around core builds
  - Application porting
- Development Issues
- Next steps

OPEN SOURCE SYMPOSIUM 2006

# Survey: no longer why, rather when?

- “Recommendation: Dont hesitate to bring in Linux or open source software if it meets your needs”
- Survey results comparing 2005 to 2004 show:
  - 10% more customers using open source software
  - 13% fewer customer with “no plans” to use open source software



Source: ‘O pen Source Usage Up, But Concerns Linger,’ Forrester Research, Inc, June 2005

# "Proprietary Unix capabilities at commodity prices"

- **Business case**
  - Price / Performance
  - Choice & flexibility
  - Security
  - Rate of innovation
- **Technical case**
  - Ease of migration
  - Unix skills transfer
  - Ease of sys. management
  - Breadth of ecosystem

## Why Migrate to Linux?

Linux has many advantages, neither Unix nor Windows has them all:

- Wide ranging support for commodity hardware
- Same OS supported from PDAs to super computers
- Support for seven architectures in RHEL:
  - Different architectures provide
    - the same API
    - Optimizations for different tasks
  - No hardware vendor lock-in
- Completely open APIs with no disadvantages for any ISV

Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

## Best practices methodology for creating, deploying & managing a core RHEL build

- Clearly identify the goal: a one-size fits all foundation or a very-highly tuned, application-specific configuration
- Have a detailed understanding of the environment and framework into which the systems using the build must be incorporated. This includes:
  - Authentication/security configuration
  - Network configuration
  - Existing monitoring and management solutions
  - Storage and backup tools and processes

## Core build best practices... continued

- Gather the software requirements for this environment. This includes required RPMs, 3rd party dependencies, and in-house developed software
- Package in RPM Package Manager (RPM) format as many applications and utilities as possible
- Gather operating system and application tuning and optimization parameters

## Typical migration 'targets'

- Third party applications
- Infrastructure, Messaging
- 3 tiers – Web, Application & Database
- Custom development e.g. C, C++, COBOL, FORTRAN

# Targets: Infrastructure, Messaging, Web, Java, Database

- File Server
- NFS, Samba, LDAP, eDirectory, Veritas
- Print Server
- Samba, lpd
- DNS Server
- Bind
- Iptables, Kerberos, ssl, vpn
- Build Server
- Gcc, make, CVS
- Custom Utility Server

- Custom Messaging Systems
- Tibco RV
- Financial/Market Data Feeds
- Reuters RMDS
- Mail Routing
- Lotus, Sendmail, Ximian, Binari
- Instant Messaging
- Jabber

- HTTP Servers
- Apache, Iplanet, Zeus
- Web Caching
- Squid
- Content Engine
- Server-side Applications
- CGI, PHP, Perl, Shells

- RDBMS
- Oracle, DB2, Sybase, Informix, PostgreSQL, MySQL
- DB Application Servers
- Oracle

- JSP Servlet Engines
- Tomcat
- J2EE Application Servers
- WebSphere, WebLogic, OAS, JBoss
- JDKs/SDKs
- IBM, Sun, BEA, Blackdown

## Targets: C/C++ Apps

- C Development Environment
- Gcc, gdb, make, gprof, CVS, Rational
- C Runtime Environment
- glibc
- 3rd Party APIs and Libraries
- Analysis
- Core/Kernel Dump facilities
- High Availability
- Red Hat Cluster Manager

## Questions to ask

- What application packages do you currently run on Solaris that will need to run on Linux?
- Do you use middleware? If so, what is the satisfaction rate of the product on Linux?
- Are there any specific development tools that will need to run on Linux? Are they available?
- What hardware dependencies will need to be supported with Linux? For example, is there tight integration in your code with Sun's SPARC processor?
- Are there any hooks to the Solaris kernel in your code?
- Are there issues with the network that will need to be addressed?
- What will you do about your direct access storage devices (DASD)?
- With Solaris you can assign processes to a particular processor. Is this being done on your current box?

## Keys to Portability

- Standards, standards and more standards
  - If you start from a source code base that uses proprietary standards, fix that first!
- What standards are important?
  - ANSI (C & C++), POSIX/The Open Group & Free Standards Group (LSB now includes The Open Group Single UNIX specifications)
- Other considerations
  - Languages designed for portability.
    - • Java, Perl, C#, etc.
- Middleware that helps with portability

## Possible Issues

- Shell script behavior
  - use perl
- Init scripts/chkconfig
  - #comment: chkconfig: 2345 55 10
- Possible API incompatibility
  - Red Hat has backported NPTL support for RHEL3 (which is based on the Linux 2.4 kernel). RHEL 3 supports both NPTL and Linux threads.

## Possible Issues(cont.)

- Endianness (big-endian & little-endian)
  - Linux: /usr/include/endian.h
  - The 80-90% case is going to be networking code. Use messaging middleware. Use a text based approach
- 32-bit versus 64-bit platforms.
  - Not make assumptions about the size of data types
- Availability of Commercial-off-the-Shelf Applications and Middleware

## Methods Of Porting

- **Method 1**: Switch to GNU compilers on Solaris first
- **Method 2**: copying the code to Linux and compiling there

- **Method 1** recommended. First compiling the code on Solaris and your SPARC server will make the port that much easier. If the application can be recompiled and regenerated on Solaris using the GNU tools, it can later be moved over to Linux. Only API issues can impede progress

## RHEL 4 Tools (GNU vs Solaris)

- GNU CC (GCC)
- Tiny Cobol
- GNU Cobol2C
- GNU Fortran
- J2EE

- Test with GCC on Solaris
- Test with Solaris make
- Once-debugged, move to RHEL
- Few flag differences, nothing major

## Problems when Migrating

A developer faces a number of problems:

- Different tools (compiler, linker)
  - Different options
  - Differences in the accepted language
- Differences in the programming environment
  - Difference standards (or lack thereof)
  - Noncompliance to standards
  - Closed-source libraries not available
- Different runtime characteristics
  - Same code might run faster or slower
  - If code runs slower, code needs rewrite

# Differing Options

## Compiler Differences

| GCC | Sparc cc | Description |
|---|---|---|
| -static | -Bstatic | Statically link the application |
| -O | -fast | maximize speed of compiled app |
| -shared | -G | The linker creates a shared object with this flag |
| -save-temps | -keeptmp | Compiler not to remove temp files |
| -M | -H | Prints path name of files being compiled |
| ...more | | |

## Linker Differences

| GNU | Solaris | Description |
|---|---|---|
| -shared | -G | Generates a shared object |
| -static | -a | mode and prevents linking with shared libraries |
| -rpath path | -R path | Specifies search direction to run-time linker |
| ...more | | |

## Preparation of Migration

A number of preparation steps can ease porting:

- The GNU tools (compiler, linker, etc) are available on other OSes as well
  - Compile project on the other OS with the tools to be used on RHEL
    - Eliminate dependencies on language extensions of old compiler
    - Use correct options for tools
  - Enable all warnings and eliminate them
    - Add -Wall -Wextra to compiler command line
- Identify platform-specific interfaces used
  - Solaris threads vs POSIX threads
  - Rewrite code without these interfaces

# Compile on Linux

After the preparation getting compilation started is easy

Possible remaining problems:

- Remaining platform-specific code (e.g., #ifdef __solaris__)
- Remaining architecture-specific code
  - Assembler code for different architecture (SPARC or PA RISC vs x86-64)
  - Different assembler syntax (Intel vs AT&T on x86)
  - Endianess problems (big vs little)
  - 32-bit vs 64-bit issues
- Closed-source libraries not available on Linux
  - Persuade 3rd party ISV to port libraries
  - Find replacement among plethora of libraries available on Linux

OPEN SOURCE SYMPOSIUM 2006

## Aside from Compilation

The compilation process is not the only step which needs adjustment:

- Debugging: gdb is available on other platforms as well
  - Limited GUI capabilities outside Eclipse
- Memory handling debugging:
  - Purify available for Linux
  - Non-proprietary solutions:
    - Purify-like: valgrind
    - Special compile mode: mudflap
- Profiling:
  - gprof: old-Unix style, coarse granularity, exact call tree
  - Oprofile: system-wide profiling; kernel, applications, or DSOs
  - SystemTap: detailed kernel performance analysis
  - Frysk and Dogtail

## Standard Compliance

Goal of standard compliance is easier migration:

- Linux complies to POSIX wherever possible
  - Minor differences exist
  - No formal POSIX testing (nobody volunteered recently to pay $$$)
- Linux supports more POSIX options than any Unix OS
  - http://people.redhat.com/drepper/posix-option-groups.html
- A program using POSIX interfaces correctly should need almost no porting
  - API specified in standard (names of headers, data types interfaces)
  - Semantic specified to a great extend
    - Programs must not use unspecified behavior
    - Difficult to test this does not happen

## Standard Compliance(cont.)

- gcc with glibc, libstdc++, and libgcj implement
  - Almost all of ISO C99 (only some minor features missing)
  - Most of ISO C++
    - Accepted language very close to ISO C++ (unlike other compilers)
    - Main missing feature: export keyword
    - C++ library fully supported and highly optimized
  - Fortran90 support
    - Not complete, but usable
  - Java support
    - gcj supports compilation to native code: higher speed
    - gij provides interpreter
    - libgcj mostly complete library support (as of Java 2)

## Java

Certified Java environments available:

- Sun JVM
  - Available for x86, x86-64, and IA-64
- IBM JVM
  - Available on all seven architectures
- BEA JVM
  - Available for x86, x86-64, and IA-64
- Soon: Apache Harmony
- J2EE stacks
  - From the JVM providers
  - Jonas
  - JBoss

## Migrating from Windows

It is a completely different story:

- The Windows API has nothing in common with the POSIX API
  - No 1-to-1 mapping
  - Some Windows APIs cannot be implemented in terms of POSIX interfaces
- Use of Windows (wine) emulation libraries not real migrating
  - Incomplete, always will be since MSFT is adding to it
  - Incredibly inefficient
  - GUI incompatible with native interface
- Possible to use POSIX interfaces on Windows
  - Unix Services for Windows
  - Cygwin

## Adapting to Linux Environment

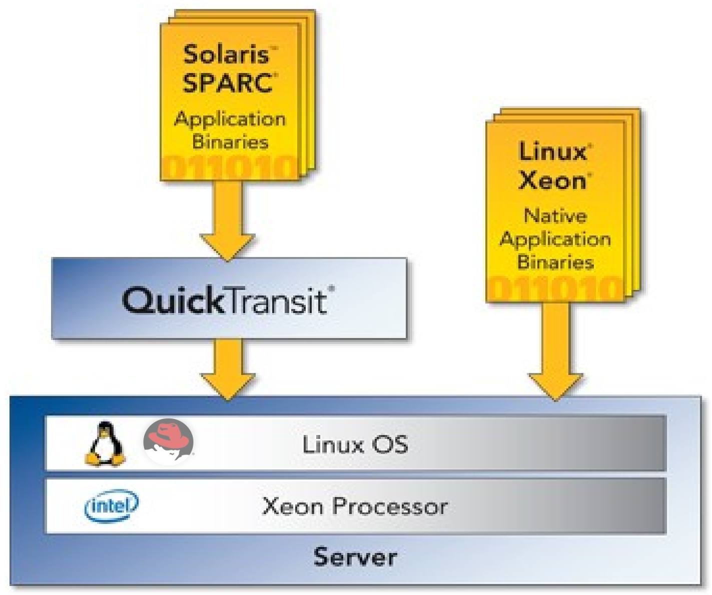
Last step: make the migrated application fit in

- If MOTIF widget set is used, convert to use gtk+
  - Native look & feel
  - Interaction with Linux applications through bonobo
  - Better resource usage
- Use Linux-specific interfaces
  - For performance
  - To reduce risk in programming
- Add support for advanced security
  - Extension to SELinux policy
  - Adjust build process to take advantage of  ExecShield and related security features

## Red Hat Ready Partners' Solutions

Your Solaris applications (binaries) will:

- Immediately install and run (as is) on Linux/x86 machine
- Run (through QuickTransit)
- Have the same (full) functionality as on subject platform


Solaris SPARC Application Binaries
011010
Linux Xeon Native Application Binaries
011010
QuickTransit
Linux OS
Xeon Processor
intel
Server

## SUMMARY

- Planning and Assessment
  - Goals (lower cost, better performance, reliability, etc.)
  - Portability of in-house code
  - Availability of 3rd party applications and middleware
  - In-house and 3rd party resources
  - Tier-by-Tier vs. App-by-App
  - Key Milestones (external requirements)
  - Lifecycle plan
  - Standards – “What few things must be the same so that everything else can be different?”
- Implementation
- Deployment
- Management

Power of Open Source Architecture
OPEN SOURCE SYMPOSIUM 2006

## Unix to Linux Migration (U2L)

- Planning and Assessment
- Implementation
  - Training
  - Change platform first or last?
  - C, C++, Java, etc.
  - IA-32, IA-64, AMD64, IA-32E
  - Migrating code
  - Migrating data
  - Publication of standards
  - Performance and Functionality testing
  - Evaluation of Open Source Architecture
- Deployment
- Management

## Unix to Linux Migration (U2L)

- Planning and Assessment
- Implementation
- Deployment
  - Training
  - RPM
  - Production testing
  - Certification(s)
  - High availability
  - Virtualization
  - Red Hat Network
- Management

## Unix to Linux Migration (U2L)

- Planning and Assessment
- Implementation
- Deployment
- Management
  - Training
  - Security errata
  - Updates
  - Upgrades

## Unix to Linux Migration (U2L)

- Planning and Assessment (4-8 weeks)
- Implementation (1-4 weeks)
- Deployment (1-12 weeks)
- Management (2-4 weeks)

- Custom kernel code, device drivers, etc., will vary tremendously
- Of course, production databases need 3-6 months of burn-in time (no different from upgrading from release N to release N+1 of UNIX)
- Of course, major re-engineering takes longer than straightforward porting
- But...many migration projects were finished before they were officially started! When in doubt, **give it a try**c— hances are you won't find any *new* problems

# Ensuring Success? Red Hat Migration Assessment Service

## What is it?

- Prepares successful migration
- Reviews technical infrastructure, applications & systems management
  - Provides specific analysis of software, performance & savings
  - Delivers migration plan

## Why is the assessment so critical?

- Ensures successful migration
- Speeds up implementation
- Decreases risk –right first tim e

## More information

- Red Hat Migration Center
  - www.redhat.com/rhel/migrate/
- How to migrate from Red Hat Linux to Red Hat Enterprise Linux: A Technical Paper
  - www.redhat.com/whitepapers/rhel/RHL_to_RHEL_Overview.pdf
- Migrating to Red Hat Enterprise Linux from Red Hat Linux - Benefits and Guidelines
  - http://www.redhat.com/whitepapers/rhel/Migrate_RHEL.pdf

Questions? Seung-Do Yang, syang@redhat.com