OPERATING SYSTEM UPGRADES AT RHIC*

S.Binello†, J.Laster, R.Katz, J.Piacentino, A.Fernando
Collider-Accelerator Department, BNL, Upton, NY, USA

Abstract

Upgrading hundreds of machines to the next major release of an Operating system (OS), while keeping the accelerator complex running, presents a considerable challenge. Even before addressing the challenges that an upgrade represents, there are critical questions that must be answered. Why should an upgrade be considered? (An upgrade is labor intensive and includes potential risks due to defective software.) When is it appropriate to make incremental upgrades to the OS? (Incremental upgrades can also be labor intensive and include similar risks.) When is the best time to perform an upgrade? (An upgrade can be disruptive.) Should all machines be upgraded to the same version at the same time? (At times this may not be possible, and there may not be a need to upgrade certain machines.) Should the compiler be upgraded at the same time? (A compiler upgrade can also introduce risks at the software application level.) This paper examines our answers to these questions, describes how upgrades to the Red Hat Linux OS are implemented by the Controls group at RHIC, and describes our experiences.

INTRODUCTION

Operating system upgrades are labor intensive, disruptive and almost always come with incompatibilities and software bugs. Deciding when to perform OS upgrades at RHIC has been an evolving process. On the one hand, there is the desire to maintain a stable functioning controls system. On the other hand, there is the desire for new features and capabilities that come with new versions of an OS.

Operating system upgrades come in two flavors. The first is a major OS upgrade, where significant new features and revisions are released. At RHIC, where we use the Linux OS distributed by Red Hat, major releases are reflected by version number changes, such as Red Hat Enterprise Linux(RHEL)5 to RHEL6. The second type consists of updates to the current OS version that mainly consist of bug fixes, security fixes, and minor enhancements. For Red Hat, this is reflected by minor version number changes such as RHEL6.1 to RHEL6.2.

OS UPGRADES AND UPDATES AT RHIC

The Controls group is responsible for upgrading several hundred Linux machines at RHIC. Over time, we have mostly reached a consensus that major upgrades to the OS will be performed based on Red Hat's schedule for life cycle support. In the past, this has been a rather challenging schedule, in that its basic support phase (called Production 1) was limited to four years. Lately, that schedule has been relaxed a little, as Red Hat has extended its basic support phase to five and one-half years [1]. The Production 1 phase differs from subsequent phases in that it includes software enhancements and support for new hardware.

Mindful of the inherent risks involved in adopting a new major OS release, we have taken the position that we will not upgrade to a new version within the first year of its introduction.

Aside from major OS upgrades, Red Hat also makes minor releases, as well as continuous asynchronous security and bug fixes, called erratas. Red Hat provides software that has the capability to install these erratas automatically when they become available. However, to maintain strict control over our update schedule, we have decided not to use this feature. Instead, we update machines annually to coincide with the RHIC shut-down. As we intend to run the whole year with a particular OS update, and unlike major OS upgrades, our policy has been to install the most recent minor release, and to incorporate the erratas at that time.

Our goal is to perform no updates or upgrades to our systems while the accelerators are operational. However, we make an exception for bug fixes that correct problems that we actually encounter, and lately we have also decided to include security erratas deemed critical by ITD. Our intention is to limit disruptions, and reduce the possibility of introducing problematic software while the accelerators are running. Whether it's a major OS upgrade or simply an update, it is a rare occurrence where it has not also introduced bugs that have caused a wide range of disruptions, running the gamut from kernel crashes, to NFS issues, to desktop difficulties. Just recently a relatively innocuous application like the Nautilus file browser was found to be causing major performance problems with our NAS file server.

Upgrade Schedule

In an effort to extend the amount of test time, all upgrades and updates are scheduled as early as possible during the summer months when the accelerators are not running. To ensure that we always have a copy of the software packages that are installed on our systems, the packages are retrieved from Red Hat and stored in a local repository. This repository is frozen once testing is completed and upgrades begin.

As mentioned earlier, exceptions are made in the event that we experience problems with a particular software

*Work performed under Contract Number DE-AC02-98CH10886 with the auspices of the US Department of Energy.
† sev@bnl.gov

ISBN 978-3-95450-139-7

package. This has occurred several times, typically right after an upgrade. Sometimes these upgrades can be limited to a subset of machines that are specifically affected by the problem at hand.

How Encompassing Should The Upgrade Be?

In general, we attempt to upgrade most of our machines to the same OS level. However, there are certain machines that we do not necessarily always upgrade. We have several classes of Linux machines that serve specific functions:

- Developer Consoles - used by individuals to develop software
- Process Servers - used to run controls server software such as loggers and managers.
- Main Control Room (MCR) Consoles – multi-headed displays used by operators in MCR.
- System Servers - used to run system services such as NIS, NTP, DHCP, FTP.
- Archive Servers - NFS servers that store historical data.
- Field Consoles - used throughout the complex to run Controls applications
- Assorted servers with dedicated functions - such as NX, compute, compile, version control (ClearCase).

Note, our main file servers are Network Attached Storage (NAS) devices that are not included in the Red Hat OS Upgrade. However, they too are usually upgraded by the vendor during the same time period.

In general, we upgrade all machines except for system and archive servers. System servers perform limited, but highly critical functions. Archive servers are NFS file servers that contain accelerator data logged over the years.

The consensus has been that these types of servers would not benefit much by an OS upgrade, and may actually suffer from the introduction of a bug that could have a widespread negative impact. However, it does mean that we should monitor the release of software specific to the functionality of these machines, and do a cost analysis to determine whether an upgrade would be beneficial. For example, our archive servers may benefit from the availability of NFS 4 with the release of RHEL6.

Though we prefer to upgrade the remainder of our machines to the same OS level, this has not always been possible. Running more than one major version of an OS creates some challenges in that applications built on the newer version do not always run on the older one, due to the unavailability of newer versions of libraries such as glibc and other standard C and C++ libraries.

This last year we operated in just such a heterogeneous environment, where some machines ran under RHEL5 while others ran under RHEL6. Instead of creating two versions of an application, one for each version of the OS, we created an environment where applications were strictly run from only RHEL6 machines, where both old and new applications were able to run. In addition, as we still supported the building of executables on RHEL5 machines, we also needed to maintain separate development environments.

Though we ran successfully with this configuration for almost a year, it did create additional complexities and some confusion. So, in general we would try to avoid this situation, but it is an option.

Compiler Upgrades

Included with OS upgrades and updates are equivalent upgrades and updates to the associated compiler. A compiler upgrade can also introduce risks at the software application level. Though the Controls group has been transitioning software development to JAVA, the bulk of our software is still in C++.

Compiler upgrades and related libraries, mirror OS upgrades in that major version changes (ex: gcc3 to gcc 4) are associated with major Red Hat OS releases (ex: RHEL5 to RHEL6), while minor updates are released along with OS updates.

In the past, Red Hat has facilitated the transition to a new OS by providing a compatibility compiler package that allowed the migration to the new OS without having to migrate to the new compiler as well. Though this still involved some tailoring of our environment to use these packages, it proved to be an extremely convenient way for us to tackle two interdependent, time-consuming, and complex upgrades. As of RHEL6, Red Hat no longer supports this migration path, instead it now backports the new compiler so that migration and testing needs to be started on the older OS.

During our last upgrade to RHEL6 from RHEL5, both compiler and OS were upgraded at the same time. This was not an ideal situation, but was necessitated by Red Hat's new policy. Our approach, in this case, was to upgrade machines that were not used for software development in parallel to testing and upgrading the compiler.

During the transition from one major compiler release to the next, we temporarily create two environments, so as to allow development under either compiler. This means separate build commands, shared object libraries and executables, with the logic to automatically find and select the appropriate ones.

A Typical Upgrade Path

As indicated earlier, not all our upgrades necessarily follow the same path. But, based on current Red Hat policy we expect that going forward the first task will be to upgrade and test the compiler and then upgrade the OS. This is actually an order we prefer, and have followed several times already.

ISBN 978-3-95450-139-7

An OS upgrade to our controls systems normally begins with an upgrade to just one machine. From this machine, an environment is set up to allow compiles while also coexisting with the previous version of the OS. Once this environment is established, all libraries and applications are recompiled to address any incompatibilities with the new compiler and libraries. A small but critical subset of the applications is released to a specific location where they can be accessed by any machine that is also upgraded to the new OS.

Once a test machine has been successfully converted, additional developer consoles from the Controls group are upgraded. In this manner, the set of applications built and exercised under the new OS slowly grows naturally as software developers build and release applications. No attempt is made to build and release all applications at once under the new OS. Instead, there is a natural migration to the new executables as developers work on them. In the end, the set of Controls executables will be a mix of applications built under older versions of the OS, and those recently built under the new version. With this approach it is quite possible to have a set of executables that have been built over a wide range of versions of the operating system.

After a period of testing within the Controls group, the upgrade is expanded to include desktops used by physicists and consoles used in the MCR. In this way we try to constrain and resolve upgrade issues within the Controls group, before they are encountered by the wider user community. The one time that we did not follow this approach we found that many of our users struggled with various desktop issues.

During this transition period there is an ongoing parallel effort to upgrade a few sample machines from each of the various categories described above, especially MCR consoles and process servers. Albeit, as we are in shut-down mode these machines are not as heavily used as they would be normally, but the basic functionality can still be tested.

After some period in which the Controls group has actively worked in the new desktop environment, built, released and executed applications under the new OS, and tested sample machines, a full-fledged migration to the new OS is initiated.

Testing

For the most part, our approach to testing entails upgrading and testing sample machines and actually employing them for the purpose for which they are intended. We then gradually expand the number machines that are upgraded from each class.

At BNL there are several smaller accelerators within the RHIC complex whose start-up is scheduled to occur much earlier than RHIC. This provides for a natural, phased-in, approach to stress test the upgraded systems.

About a month before RHIC becomes operational, a dry-run is performed where applications are tested, and implicitly this also serves to test the underlying system upgrades. Tests also exist to ensure that consoles in MCR are configured and operating correctly. Specific attention is paid to the desktop environment.

SUMMARY

Operating system upgrades are complex, time consuming, and are labor intensive. More often than not they introduce defective or incompatible software. Our approach to OS upgrades and updates has been evolving over the years. We continue to refine our approach to balance the need for newer software with the need to maintain a stable, functioning controls system. Our upgrade schedule is mostly influenced by Red Hat's life cycle support schedule. To limit risks we delay adoption of any new major release for at least a year, and perform updates annually. To limit disruptions we perform OS upgrades or updates only during the summer, when the accelerators are not operational. Looking forward, we expect to make use of virtual machines to facilitate the upgrade process.

REFERENCES

[1] Red Hat web site, https://access.redhat.com/support/policy/updates/errata

ISBN 978-3-95450-139-7