Rebootless Linux Kernel Patching with Ksplice Uptrack at BNL

RHIC relativistic heavy ion collider — RHIC & ATLAS Computing Facility at Brookhaven National Laboratory — ATLAS EXPERIMENT

Christopher Hollowell <hollowec@bnl.gov>
James Pryor <pryor@bnl.gov>
Jason Smith <smithj4@bnl.gov>

Introduction

Ksplice/Oracle Uptrack is a software tool and update subscription service which allows system administrators to apply security and bug fix patches to the Linux kernel running on servers/workstations without rebooting them. The software is actively being used at the RHIC/ATLAS Computing Facility (RACF) at Brookhaven National Laboratory (BNL), on roughly 2,000 hosts running Scientific Linux and Red Hat Enterprise Linux.

Uptrack At RACF

Primarily being utilized on our processor farm.

Also installed on critical hosts where reboots are highly disruptive.

Rebooting our processor farm systems requires advance scheduling.

- -Most of our users do not checkpoint the progress of their batch or interactive jobs.
- -Many jobs executed on our farm are long running: often 24 hours or more.
- -A large percentage of the nodes have dual roles as dCache, XROOTD, or HDFS data servers.

Scheduling reboots, and implementing them in a “rolling” manner (subsets of the farm at a time) can take days or weeks.

- -Our facility supports many experiments: requires interaction and compromise with several different computing coordinators.
- -Takes a considerable amount of administrative time to implement and execute.
- -Systems are exposed to critical kernel vulnerabilities until they are rebooted.

Experience With Uptrack

We've had a positive experience with the software since we began using it in April 2011.

- -Updates generally become available on or close to the day they are released by the Scientific Linux (SL) developers.
  - -Occasionally Uptrack updates are available even before they are released for SL.
  - -Uptrack has minimized the time we are exposed to critical kernel vulnerabilities, and eliminated user impact.
- -Uptrack updates have caused no instabilities/crashes on our hosts.
- -Ksplice/Oracle has been very flexible with us.
  - -They modified their software such that it would function with a locally caching site proxy. All our systems obtain their updates through this host.
  - -While only security updates are automatically provided through the Uptrack subscription service, a bug fix update was provided when we requested it:
    - -RHEL Bugzilla #596548: dentry_stat->nr_unused corruption

Applying Updates

Very straightforward.

- -Run “*uptrack-upgrade -y*” on the hosts to be updated.
- -Status monitoring via a supplied web interface.
  - -”*uptrack-show*” displays local updates.

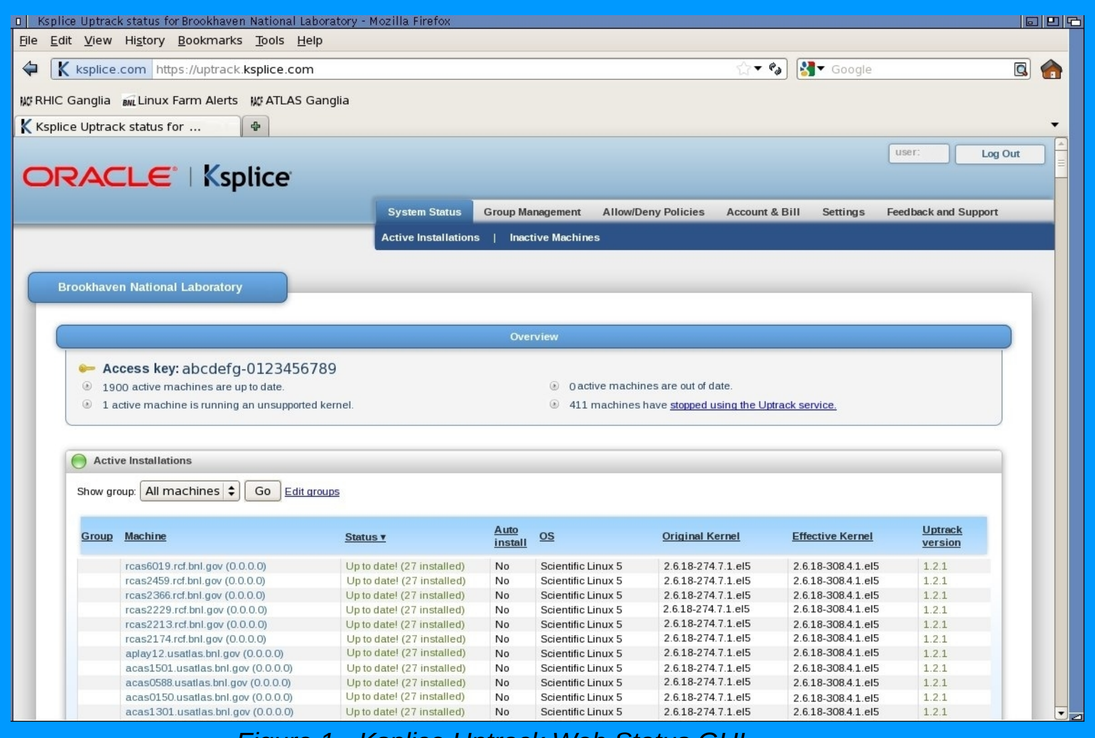

Figure 1 - Ksplice Uptrack Web Status GUI

Oracle Acquisition

Oracle acquired Ksplice on July 21, 2011.

Continued support for existing customers' Red Hat Enterprise and Scientific Linux installations.

- -No impact on our operation so far.
- -New customers may only obtain licenses for Uptrack use on Oracle Enterprise Linux.
  - -If enough people in the HEP/NP community request RHEL/SL Uptrack licenses, perhaps we can change Oracle's position?

Still possible to download the “raw” Ksplice utility

- -Software released under the GNU GPL.
- -Allows one to create and insert their own rebootless kernel patches.
- -Assuming there are no changes affecting the layout or semantics of data structures, source patches from RHEL can be used without modification.
  - -According to Ksplice's engineers, this is usually the case for security updates.
- -http://oss.oracle.com/ksplice/software

How Ksplice Works

Compiles the kernel from the original source with and without the specified patch.

- -Object files are compared for differences.
- -”*-ffunction-sections*” passed to gcc to simplify object code analysis: each function is assigned its own ELF section.

```
% cat /home/build/dmesg_open.patch
--- linux/fs/open.c.orig      2012-04-27 17:02:06.000000000 -0400
+++ linux/fs/open.c   2012-04-27 17:01:51.000000000 -0400
@@ -1180,6 +1180,8 @@
 {
        long ret;

+       printk("Open file[%s,%u]: %s\n", current_thread_info()->task->comm, current_thread_info()->pid, filename);
+
        if (force_o_largefile())
                flags |= O_LARGEFILE;

% cd rpmbuild/SPECS

% rpmbuild -bp kernel.spec
...
% cd ../BUILD/kernel-2.6.18
% ksplice-create --id=dmesgopen --patch=/home/build/dmesg_open.patch linux-2.6.18.x86_64
Starting kernel builds (this process might take a long time)...
rm ksplice-revert-stamp
  CHK     include/linux/version.h
  CHK     include/linux/utsrelease.h
make: Entering directory `/home/build/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64'
...
  LD [M]  /tmp/ksplice-tmp-Dsj57E/kmodsrc/ksplice-dmesgopen_vmlinux-old.ko
make: Leaving directory `/home/build/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64'
make: Entering directory `/home/build/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64'
  INSTALL /tmp/ksplice-tmp-Dsj57E/kmodsrc/ksplice-dmesgopen_vmlinux-new.ko
  INSTALL /tmp/ksplice-tmp-Dsj57E/kmodsrc/ksplice-dmesgopen_vmlinux-old.ko
make: Leaving directory `/home/build/rpmbuild/BUILD/kernel-2.6.18/linux-2.6.18.x86_64'
Ksplice update tarball written to ksplice-dmesgopen.tar.gz

% su -

# ksplice-apply ksplice-dmesgopen.tar.gz
Done!

# cat /etc/redhat-release
Scientific Linux SL release 5.3 (Boron)

# dmesg
...
Open file[sendmail,4983]: /proc/loadavg
Open file[gmond,4955]: /proc/stat
Open file[gmond,4955]: /proc/loadavg
Open file[atd,5051]: .
Open file[sendmail,4983]: /proc/loadavg
Open file[irqbalance,4511]: /proc/stat
Open file[irqbalance,4511]: /proc/interrupts
Open file[irqbalance,4511]: /proc/net/dev
Open file[irqbalance,4511]: /proc/irq/114/smp_affinity
Open file[irqbalance,4511]: /proc/irq/154/smp_affinity
Open file[irqbalance,4511]: /proc/irq/106/smp_affinity
Open file[clear,22517]: /etc/ld.so.cache
Open file[clear,22517]: /usr/lib64/libncursesw.so.5
Open file[clear,22517]: /lib64/libdl.so.2
Open file[clear,22517]: /lib64/libc.so.6
Open file[clear,22517]: /usr/share/terminfo/x/xterm
Open file[sendmail,4983]: /proc/loadavg
Open file[cat,22518]: /etc/ld.so.cache
Open file[cat,22518]: /lib64/libc.so.6
Open file[cat,22518]: /usr/lib/locale/locale-archive
Open file[cat,22518]: /etc/redhat-release
Open file[dmesg,22519]: /etc/ld.so.cache
Open file[dmesg,22519]: /lib64/libc.so.6
Open file[dmesg,22519]: /usr/lib/locale/locale-archive


# ksplice-undo dmesgopen

# dmesg
...
Open file[lsmod,22521]: /etc/ld.so.cache
Open file[lsmod,22521]: /lib64/libc.so.6
Open file[lsmod,22521]: /proc/modules
Open file[ksplice-undo,22520]: /sys/module/ksplice_dmesgopen/ksplice/debug
Open file[ksplice-undo,22520]: .
Open file[ksplice-undo,22520]: .
Open file[ksplice-undo,22520]: /sys/module/ksplice_dmesgopen/ksplice/stage
Open file[ksplice-undo,22520]: /sys/module/ksplice_dmesgopen/ksplice/stage
 <6>ksplice: Update dmesgopen reversed successfully
```

Figure 2 – Using the “raw” Ksplice utility to create and install a custom rebootless patch to sys_open() on a machine at BNL. When loaded, the change causes the kernel to log calling process name, pid, and filename each time the open() system call is executed.

Kernel modules are inserted which contain the modified functions, and code which replaces the first instruction of each original function in memory with a “jmp” instruction to the patched version.

Before the running kernel is modified, it is quiesced by calling “*stop_machine()*”. Checks are also performed to verify that the stack traces of all kernel threads do not contain references to replaced functions.