Fast reboots using kexec

Fast reboots using kexec

- HP servers take a long time to boot
- If only we could reboot but skip the BIOS
- We can, using kexec

kexec replaces the running Linux kernel by booting directly into a new one

- Kernel support was already enabled
- Userspace tool added to all targets
- Init system updated to support kexec reboots
- p8tools updated to support kexec reboots

Fast reboots using kexec

```python
# Prepare a kexec kernel. On older images, this will simply fail.
extlinux = unit.shell.extlinux
image = extlinux['default']
cmdline = extlinux['append']
cmdline += ' BOOT_IMAGE=' + image
image = '/boot' + image

unit.shell('kexec -l %s --append=%s', image, cmdline)

# Run reboot command. On newer images, this will reboot using
# kexec if the load succeeded.
unit.shell('reboot')
```
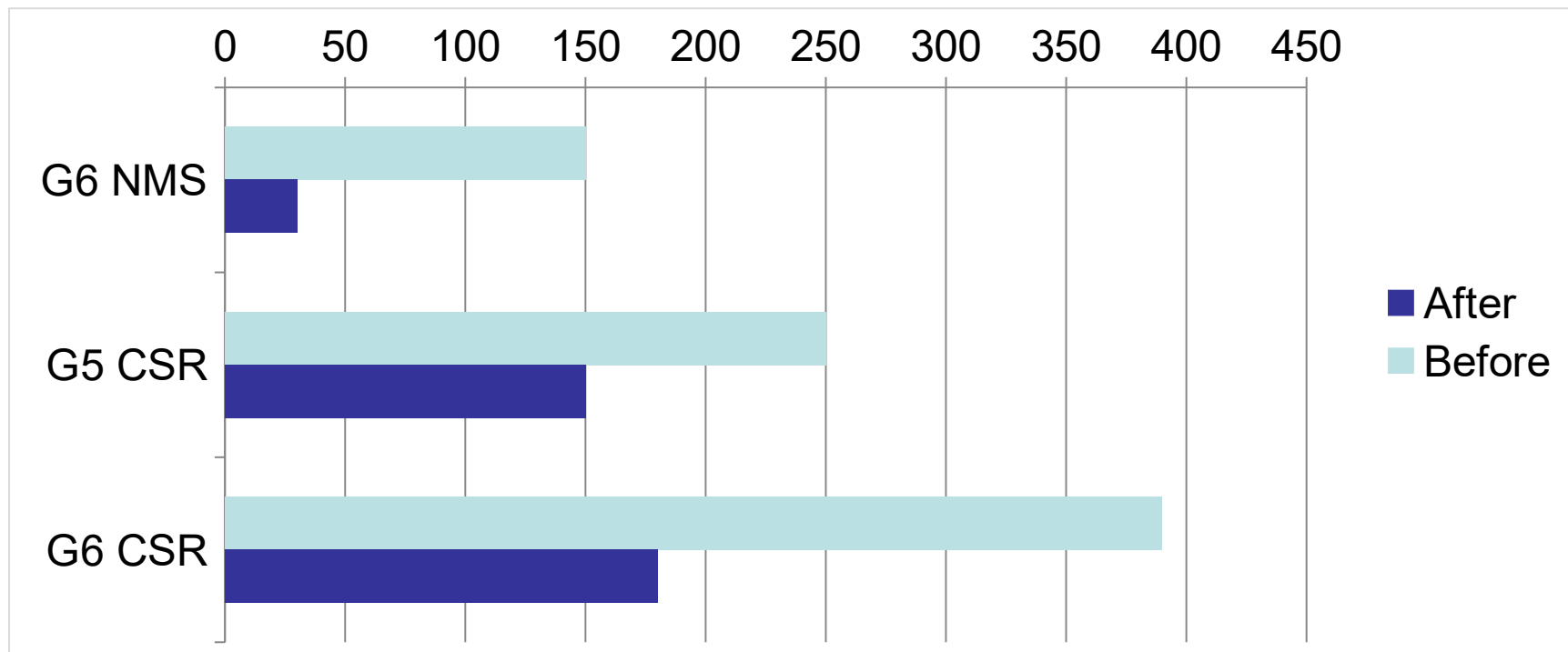
Fast reboots using kexec

trunk/development/src/p8e/p8-init-setup/p8/init.d/shutdown Tabular | Unified

r24872	r25634	
107	107	`sync`
108	108	
	109	`# If a kernel was prepared for kexec loading, execute that.`
	110	`kexec -e`
	111	
	112	`# Otherwise, do ordinary reboot.`
109	113	`echotoall "Doing ordinary reboot now..."`
110	114	`reboot -f`

Fast reboots using kexec

Annotated log messages (a proposal)

Annotated log messages (a proposal)

- We'll distinguish between two types of log messages.

- **Non-annotated log messages**
  - This is currently all of our log messages.
  - In the future, should go only into the debug log.
  - Print to stderr without priority prefix or other metadata.

- **Annotated log messages:**

  ```
  TU-17 obcu-tu-comm[5617]: [W1002] Gateway
  operation def failed; OCN A interface not found
  ```

- Go in the regular syslog file.

- The mandatory **log point code** specifies:
  - Severity (*C*ritical, *E*rror, *W*arning, *N*otice, *I*nfo).
  - Unique serial number for reference.

Annotated log messages (a proposal)

- A given serial number not only identifies the log message, but also the specific point in the code (hence "log point") producing the log message.

- If two pieces of code produce the same log message, it'll still be different numbers.

Annotated log messages (a proposal)

- Code wishing to emit annotated log points must use a common utility function/macro.

- This enables automated extraction of log point metadata from the source code.

Annotated log messages (a proposal)

- ## In C/C++:

```c
#include "airlink/logging.h"

if (alive_telegram_timeout <= alive_telegram_interval) {
    LOGPOINT(W1001, "Alive telegram timeout (%d ms) should"
        " be larger than send interval (%d ms)",
        alive_telegram_timeout, alive_telegram_interval);
    /*
    Consequence: TU will disregard alive telegrams from other TUs, causing
    repeated failover events and impacting redundancy.

    Mitigation: Ensure that timeout is larger than the broadcast interval,
    or use the default values for the vars.
    */
}
```

Annotated log messages (a proposal)

• In Bash (and POSIX sh):

```
# Optional (LOGPOINT is available both as a binary and as a shell function):
. /run/bin/utilities.sh

if [ "$alive_telegram_timeout" -le "$alive_telegram_interval" ]; then
    LOGPOINT W1001 'Alive telegram timeout (%d ms) should be larger than send
interval (%d ms)' \
        "$alive_telegram_timeout" "$alive_telegram_interval"
    # Consequence: TU will disregard alive telegrams from other TUs, causing
    # repeated failover events and impacting redundancy.
    #
    # Mitigation: Ensure that timeout is larger than the broadcast interval,
    # or use the default values for the vars.
fi
```

Annotated log messages (a proposal)

- ## In Python:

```python
from framework.logging import LOGPOINT

if alive_telegram_timeout <= alive_telegram_interval:
    LOGPOINT('W1001', 'Alive telegram timeout (%d ms) should'
        'be larger than send interval (%d ms).',
        alive_telegram_timeout, alive_telegram_interval)
    """
    Consequence: TU will disregard alive telegrams from other TUs, causing
    repeated failover events and impacting redundancy.

    Mitigation: Ensure that timeout is larger than the broadcast interval,
    or use the default values for the vars.
    """
```

- `LOGPOINT` is the magic keyword
- `printf` formatting used everywhere
  - for consistency *and*
  - to enable extraction of the log message
  - Sorry, C++ iostreams
- Long description goes into a comment just below `LOGPOINT`

- Developer tools needed to:
  - generate unused log point serials, ensuring they don't clash with other developers' (uncommitted!) serial numbers
  - extract log point metadata during build for the end-user documentation

Annotated log messages (a proposal)

- End-user tools needed to:
  - view log message documentation on the units

```
W1001(6)                    Airlink User Manual                    W1001(6)


NAME
       W1001 - warning from the obcu-tu-comm module

SYNOPSIS
       [W1001] Gateway operation X failed; OCN X interface not found

DESCRIPTION
       This  indicates  that the TU network interfaces are improperly config-
       ured; as a result, obcu-tu-comm cannot manage the gateway addresses.

   Consequence
       Communication with wayside and gateway failover will be impaired.

   Mitigation
       Ensure that the TU has OCN addresses (10.3.X.X and 10.4.X.X) correctly
       configured on the wired Ethernet interfaces (ethN).

SEE ALSO
       obcu-tu-comm(6)



Airlink 8.1.4                                                      W1001(6)
Manual page W1001(6) line 1/27 (END) (press h for help or q to quit)
```