Modernising firmware testing on POWER with Python and pre-release everything

Stewart Smith, OPAL Architect @ IBM
stewart@linux.vnet.ibm.com - @stewartsmith
What is POWER?
What is POWER9?
[29474169641,5] CENTAUR: FSI host: 0x0 cMFSIO port 7
[29474624679,5] CENTAUR: Found centaur for chip 0x0 channel 1
[29475048275,5] CENTAUR: FSI host: 0x0 cMFSIO port 2
[29475479987,5] CENTAUR: Found centaur for chip 0x0 channel 0
[29475891837,5] CENTAUR: FSI host: 0x0 cMFSIO port 3
[29476415012,5] PSI[0x000]: Found PSI bridge [working=1, active=0]
[890889699,5] BT: Interface initialized, IO 0x000e4
[1249687767,4] SLW: HB-provided idle states property found
[1249839980,5] NVRAM: Size is 576 KB
[1448125204,3] NVRAM: Layout appears sane
[1448251009,5] CAPI: Preloading ucode 200ea
[1448347995,5] FLASH: Queueing preload of 2/200ea
[1448797386,5] FLASH: Queueing preload of 0/0
[1448830140,7] FFS: Partition map size: 0x1000
[1449486797,5] FLASH: Queueing preload of 1/0
[1450255119,5] Chip 0 Found PBCQ2 at /xscom@3fc0000000000/pbcq@2012800
[1450545250,5] Chip 0 Found PBCQ1 at /xscom@3fc0000000000/pbcq@2012400
[1451090556,5] Chip 0 Found PBCQ0 at /xscom@3fc0000000000/pbcq@2012000
[1451738447,3] PHB0: Base location code not found!
[1514377310,3] PHB1: Base location code not found!
[1574807759,3] PHB2: Base location code not found!
[1635151368,5] PCI: Resetting PHBs...
[2249691669,5] PCI: Probing slots...
platform: add zaius
~1.1 Million LOC POWER firmware specific

  plus buildroot, linux, busybox, petitboot.....
It, it!
It re延续， it!
Oh, look, this was actually caught by CI but the CI scripts are so broken they don’t class not booting as a failure:

```
spawn ipmitool -H 9.3.29.140 -I lanplus -U ADMIN -P admin sol activate
[SOL Session operational. Use ~? for help]
3.15788|Ignoring boot flags, incorrect version 0x0
3.17619|ECC error in PNOR flash in section offset 0x01CB1000

3.17623|System shutting down with error status 0x60F
3.17628|System shutting down with error status 0x90000002

sol_logger.exp: EOF detected
```
Testing for POWER8...
Dev -> Test -> Ship
Firmware, is big.
Firmware Development Process (Simplified)
Firmware testing is a large integration testing effort
Existing tests?
open-power / skiboot

Build #1253

master  PM: Fixed generation of MTSPR instruction in STOP API.
STOP API generates SPR restore instruction for a given SPR.
Commit fixes the generation of mtsp instruction by

- Commit 0618926
- Compare 43eb8bb..0618926
- Branch master

Prem Shanker Jha authored  Stewart Smith committed

Build Jobs

- #1253.1  RUN_CONTAINER=ubuntu-12.04  8 min 45 sec
- #1253.2  RUN_CONTAINER=ubuntu-16.04  6 min 46 sec
- #1253.3  RUN_CONTAINER=ubuntu-latest  6 min 24 sec
- #1253.4  RUN_CONTAINER=centos6  10 min 53 sec
- #1253.5  RUN_CONTAINER=centos7  6 min 27 sec
- #1253.6  RUN_CONTAINER=fedora24  4 min 14 sec
- #1253.7  RUN_CONTAINER=fedora25  4 min 14 sec
- #1253.8  RUN_CONTAINER=fedora26  4 min 40 sec
- #1253.9  RUN_CONTAINER=fedora27  4 min 14 sec
- #1253.10  RUN_CONTAINER=debian-stretch  6 min 40 sec

Allowed Failures

- #1253.11  RUN_CONTAINER=fedora-rawhide  4 min 9 sec
- #1253.12  RUN_CONTAINER=debian-unstable  5 min 9 sec
# LCOV - code coverage report

**Current view:** top level  
**Test:** skiboot.info  
**Date:** 2018-01-26 11:12:20

<table>
<thead>
<tr>
<th>Directory</th>
<th>Line Coverage</th>
<th>Functions</th>
<th>Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccan/endian</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>ccan/list</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>ccan/stdc</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>core</td>
<td>77.3%</td>
<td>85.0%</td>
<td>-</td>
</tr>
<tr>
<td>external/trace</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>hdata</td>
<td>43.1%</td>
<td>62.7%</td>
<td>-</td>
</tr>
<tr>
<td>hw</td>
<td>17.6%</td>
<td>33.3%</td>
<td>-</td>
</tr>
<tr>
<td>hw/iptable</td>
<td>63.9%</td>
<td>57.1%</td>
<td>-</td>
</tr>
<tr>
<td>include</td>
<td>95.0%</td>
<td>95.7%</td>
<td>-</td>
</tr>
<tr>
<td>libc</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>libc/sttype</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>libc/stdio</td>
<td>97.5%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>libc/stdlib</td>
<td>94.8%</td>
<td>80.0%</td>
<td>-</td>
</tr>
<tr>
<td>libc/string</td>
<td>67.5%</td>
<td>70.6%</td>
<td>-</td>
</tr>
<tr>
<td>libfdt</td>
<td>35.0%</td>
<td>50.0%</td>
<td>-</td>
</tr>
<tr>
<td>libfstb</td>
<td>14.9%</td>
<td>16.7%</td>
<td>-</td>
</tr>
<tr>
<td>libcrypt/test</td>
<td>100.0%</td>
<td>100.0%</td>
<td>-</td>
</tr>
<tr>
<td>test</td>
<td>0.0%</td>
<td>0.0%</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Lines:** 3815/6409 (59.5%)  
- **Functions:** 399/537 (74.3%)  
- **Coverage:** -

Generated by: LCOV version 1.12
## LCOV - code coverage report

**Current view:** top level  
**Test:** skiboot-boot.info  
**Date:** 2015-05-28  

<table>
<thead>
<tr>
<th>Directory</th>
<th>Line Coverage</th>
<th>Functions</th>
<th>Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccac/emdian</td>
<td>25.0%</td>
<td>20.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>ccac/list</td>
<td>54.5%</td>
<td>54.5%</td>
<td>54.5%</td>
</tr>
<tr>
<td>ccac/stc</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>core</td>
<td>56.6%</td>
<td>65.3%</td>
<td>65.3%</td>
</tr>
<tr>
<td>hdata</td>
<td>65.5%</td>
<td>87.5%</td>
<td>87.5%</td>
</tr>
<tr>
<td>hw</td>
<td>44.8%</td>
<td>46.9%</td>
<td>46.9%</td>
</tr>
<tr>
<td>hw/ast-hmc</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>hw/ec</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>hw/fsp</td>
<td>44.0%</td>
<td>57.9%</td>
<td>57.9%</td>
</tr>
<tr>
<td>hw/ipmi</td>
<td>0.7%</td>
<td>2.6%</td>
<td>2.6%</td>
</tr>
<tr>
<td>include</td>
<td>77.7%</td>
<td>73.5%</td>
<td>73.5%</td>
</tr>
<tr>
<td>libc</td>
<td>82.8%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>libc/ctype</td>
<td>17.6%</td>
<td>16.7%</td>
<td>16.7%</td>
</tr>
<tr>
<td>libc/stdio</td>
<td>34.0%</td>
<td>28.6%</td>
<td>28.6%</td>
</tr>
<tr>
<td>libc/stdlib</td>
<td>27.5%</td>
<td>20.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>libc/string</td>
<td>72.6%</td>
<td>82.4%</td>
<td>82.4%</td>
</tr>
<tr>
<td>libfdt</td>
<td>26.3%</td>
<td>35.2%</td>
<td>35.2%</td>
</tr>
<tr>
<td>libflash</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>libinput</td>
<td>14.5%</td>
<td>11.5%</td>
<td>11.5%</td>
</tr>
<tr>
<td>platforms/astbmc</td>
<td>8.3%</td>
<td>20.0%</td>
<td>20.0%</td>
</tr>
<tr>
<td>platforms/ibm-fsp</td>
<td>87.9%</td>
<td>88.0%</td>
<td>88.0%</td>
</tr>
<tr>
<td>platforms/maribo</td>
<td>96.7%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
<tr>
<td>platforms/rheus</td>
<td>4.7%</td>
<td>9.1%</td>
<td>9.1%</td>
</tr>
</tbody>
</table>

**Lines:** 10908  
**Total:** 24690  
**Coverage:** 44.2%  
**Functions:** 981  
**Total:** 1930  
**Coverage:** 50.8%  
**Branches:** 3899  
**Total:** 12472  
**Coverage:** 31.3%  

Generated by: [LCOV version 1.9](https://lcamtuf.coredump.cx/lcov/)
"Will I regret merging this patch?"
Catch bugs early and often
Internal, (partly) automated testing
Test Frameworks

(yes, plural)
**How Standards Proliferate:**
(See: A/C chargers, character encodings, instant messaging, etc)

**Situation:**
There are 14 competing standards.

**14?! Ridiculous!**
We need to develop one universal standard that covers everyone's use cases.

**Yeah!**

**Soon:**

**Situation:**
There are 15 competing standards.
"We should have an Open Source OpenPOWER test suite"
Initial commit of op-test-framework
Goal: Testing for an open source project, as an open source project
A Perl Script.
A Perl Script that calls `system()` a lot.
A Perl Script that calls system() a lot, parsing XML with system("grep -v ..|grep -v...").
A Perl Script that calls system() a lot, parsing XML with system("grep -v ..|grep -v..."), which calls another perl script (via system())
A Perl Script that calls `system()` a lot, parsing XML with `system("grep -v .. | grep -v...")`, which calls another perl script (via `system()`), passing data to it via environment variables.
A Perl Script that calls `system()` a lot, parsing XML with `system("grep -v ..|grep -v...")`, which calls another perl script (via `system()`), passing data to it via environment variables, which calls another perl script (to parse XML),
A Perl Script that calls `system()` a lot, parsing XML with `system("grep -v ..|grep -v...")`, which calls another perl script (via `system()`), passing data to it via environment variables, which calls another perl script (to parse XML), to work out what to pass to a shell script.
A Perl Script that calls system() a lot, parsing XML with system("grep -v ..|grep -v..."), which calls another perl script (via system()), passing data to it via environment variables, which calls another perl script (to parse XML), to work out what to pass to a shell script, which then constructs a python script.
A Perl Script that calls `system()` a lot, parsing XML with `system("grep -v ..|grep -v...")`, which calls another perl script (via `system()`), passing data to it via environment variables, which calls another perl script (to parse XML), to work out what to pass to a shell script, which then constructs a python script, which then runs a test.
A Perl Script that calls `system()` a lot, parsing XML with `system("grep -v ..|grep -v...")`, which calls another Perl script (via `system()`), passing data to it via environment variables, which calls another Perl script (to parse XML), to work out what to pass to a shell script, which then constructs a Python script, which then runs a test, for example "ipmitool power on"
python -m unittest test_module.TestClass.test_method
make check
Not just Organisational Change
Innovation tokens and Spoon Theory
Failure is not an option
(Not a Python Programmer)
Why not rewrite?
Vision: Tests as a deliverable
Strategy: slowly boil the frog.
Short term wins: add tests, run them
What do we test?
How do we report test results?
One major refactor later...
(Ab)using Python UnitTest
12hr -> 20mins test runs
Requirements (test runner):
• Python 2.7
• ipmitool

Requirements (system being tested):
• fwts, linux-tools-common, lm-sensors, ipmitool, i2c-tools
• pciutils, opal-prd, opal-utils
Desired workflow:

$ git clone https://github.com/open-power/op-test-framework
$ cd op-test-framework
$ ./op-test <host and bmc login info> --host-pnor firmare-to-test.pnor
Machines:
• Tuleta (2 socket POWER8, IBM FSP)
• Habanero (1 socket POWER8, AMI BMC)
• Firestone (2 socket POWER8, AMI BMC)
• Garrison (2 socket POWER8 with GPUs, AMI BMC)
• p8dsu (2 socket POWER8, SuperMicro BMC)
• ZZ (2 socket POWER9, IBM FSP)
• p9dsu (2 socket POWER9, SuperMicro BMC)
• Zaius (2 socket POWER9, OpenBMC)
• Witherspoon (2 socket POWER9 with GPUs, OpenBMC)

Simulators:
• Mambo (functional simulator)
• Qemu
What do tests look like?
class BootToPetitbootShell(BasicIPL):
    def runTest(self):
        self.system.goto_state(OpSystemState.OFF)
        self.system.goto_state(OpSystemState.PETITBOOT_SHELL)
class BootTorture(unittest.TestCase, TestPCI):
    BOOT_ITERATIONS = 1024

    def setUp(self):
        conf = OpTestConfiguration.conf
        self.system = conf.system()
        self.pci_good_data_file = conf.lspci_file()

    def runTest(self):
        self.c = self.system.sys_get_ipmi_console()
        for i in range(1,self.BOOSTER_ITERATIONS):
            print "Boot iteration %d..." % i
            self.system.goto_state(OpSystemState.PETITBOOT_SHELL)
            self.system.host_console_unique_prompt()
            self.c.run_command("head /sys/firmware/opal/msglog")
            self.c.run_command("tail /sys/firmware/opal/msglog")
            if self.pci_good_data_file:
                self.check_pci_devices()
            self.c.run_command("dmesg -r|grep '^[4321]'")
            self.c.run_command("grep ',[0-4]" /sys/firmware/opal/msglog")
            self.system.goto_state(OpSystemState.OFF)
class PetitbootDropbearServer(unittest.TestCase):
    def setUp(self):
        conf = OpTestConfiguration.conf
        self.cv_SYSTEM = conf.system()

    def runTest(self):
        self.cv_SYSTEM.goto_state(OpSystemState.PETITBOOT_SHELL)
        print "Test Dropbear server not running in Petitboot"

        c = self.cv_SYSTEM.sys_get_ipmi_console()
        self.cv_SYSTEM.host_console_unique_prompt()
        c.run_command("uname -a")
        # we don't grep for 'dropbear' so that our naive line.count
        # below doesn't hit a false positive.
        res = c.run_command("ps|grep drop")
        print res
        for line in res:
            if line.count('dropbear'):
                self.fail("Dropbear is running in the skiroot")
        pass
```python
def verify_cpu_freq(self, i_freq, and_measure=True):
    l_cmd = "cat /sys/devices/system/cpu/cpu0/cpufreq/cpuinfo_cur_freq"
    cur_freq = self.c.run_command(l_cmd)

    if not cur_freq[0] == i_freq:
        # (According to Vaidy) it may take milliseconds to have the
        # request for a frequency change to come into effect.
        # So, if we happen to be *really* quick checking the result,
        # we may have checked before it has taken effect. So, we
        # sleep for a (short) amount of time and retry.
        time.sleep(0.2)
        cur_freq = self.c.run_command(l_cmd)

    self.assertEqual(cur_freq[0], i_freq,
                     "CPU frequency not changed to %s" % i_freq)

    if not and_measure:
        return

    frequency_output = self.c.run_command("ppc64_cpu --frequency")
    freq = {}
    for f in frequency_output:
        m = re.match(self.ppc64cpu_freq_re, f)
        if m:
            freq[m.group(1)] = int(decimal.Decimal(m.group(2)) * 1000000)

    # Frequencies are in KHz
    print repr(freq)
    self.assertAlmostEqual(freq["min"], freq["max"], delta=(freq["avg"] / 100),
                           msg="ppc64_cpu measured CPU Frequency differs between min/max when frequency set explicitly")

    self.assertAlmostEqual(freq["avg"], freq["max"], delta=(freq["avg"] / 100),
                           msg="ppc64_cpu measured CPU Frequency differs between avg/max when frequency set explicitly")

    delta = int(i_freq) / (100)
    print "Set %d, Measured %d, Allowed Delta %d" % (int(i_freq), freq["avg"], delta)

    self.assertAlmostEqual(freq["avg"], int(i_freq), delta=delta,
                           msg="Set and measured CPU frequency differ too greatly")
```

Workarounds for Python
# pxssh has a nice 'echo=False' mode, but only
# on more recent distros, so we can't use it :(  
p = pxssh.pxssh()
# Work-around for old pxssh not having options= parameter
p.SSH_OPTS = p.SSH_OPTS + " -o 'StrictHostKeyChecking=no'"
p.SSH_OPTS = p.SSH_OPTS + " -o 'UserKnownHostsFile /dev/null' "
p.force_password = True
p.logfile = self.logfile
self.pxssh = p
return p
# Ubuntu likes to be "helpful" and alias grep to include color, which isn't helpful at all. So let's go back to absolutely no messing around with the shell if self.username != "root":
    p.sendline('sudo -s')
    p.expect("password for")
    p.sendline(self.password)
    p.set_unique_prompt()

p.sendline('exec bash --norc --noprofile')
p.set_unique_prompt()
Wrapping pexpect
Torture Tests
class ConsoleIpmiTorture(unittest.TestCase):
    def setUp(self):
        conf = OpTestConfiguration.conf
        self.cv_HOST = conf.host()
        self.cv_IPMI = conf.ipmi()
        self.cv_SYSTEM = conf.system()
        self.torture_time = 2400

    def ipmi_interface_torture(self):
        # OOB IPMI Torture
        torture_time = self.torture_time
        self.thread_list = []
        cmd_list = ["sdr list", "fru print", "sel list", "sensor list","power status"]
        for j in range(1,3):
            for idx, cmd in enumerate(cmd_list):
                num = j*(idx + 1)
                thread = OobIpmiThread(num, "Thread-%s" % num, cmd, torture_time)
                thread.start()
                self.thread_list.append(thread)

    def console_torture(self):
        thread = SolConsoleThread(1, "SOL-Thread", self.test, self.torture_time)
        thread.start()
        self.thread_list.append(thread)

    def runTest(self):
        self.setup_test()
        self.ipmi_interface_torture()
        self.console_torture()

    def tearDown(self):
        # Wait for all the thread to finish
        for thread in self.thread_list:
            thread.join()
find /
SOL session closed by BMC
<insert missing characters here>
BMCs are terrible
except BMCDisconnected as e:
    print "# %s %s" % str(e)
    print "# We can possibly continue..."
    print "# Failing current command and attempting to continue"
    self.terminate()
    self.connect()
    console = self.get_console()
    print "# On reconnect, attempt to cancel last command (ctrl-c)"
    # Note: this is a terrible idea. If BMC vendors created reliable
    # SoL implementations this kind of crap wouldn't be needed.
    # This is incorrect on so many levels it's not funny. For a start,
    # we really don't want to send random control characters during
    # boot!
    console.sendcontrol('c')
    try:
        rc = console.expect([BMC_DISCONNECT,
                            "\[console-pexpect\]#\$"], 10)
    
        if rc == 0:
            self.terminate()
            raise BMCDisconnected(BMC_DISCONNECT)
    except pexpect.TIMEOUT:
        print "# No response from BMC... trying 'mc reset cold'"
        self.mc_reset()
        self.terminate()
        self.connect()
        console = self.get_console()
        console.sendcontrol('c')
        raise CommandFailed("ipmitool", BMC_DISCONNECT, -1)
20Mbit serial port vs TTY Layer
/op-test --run testcases.Console.Console32k
Current State: Test Coverage
Current State: Adoption
./op-test --config-file wio82.conf
   --flash-skiboot ~/skiboot/skiboot.lid.xz.stb
   --host-pnor ~/op-build/output/images/witherespoon.pnor.squashfs.tar
Thank you.