Boot time testing with ftrace

Laura Nao, Collabora Ltd

LPC 2024

Motivation

- Automatic detection of boot slowdowns
  - No kselftest available upstream
  - Utils available in tools/ and scripts/ for manual inspection
  - Challenging to run in CI/non-interactive environments

Available Tools Upstream

- scripts/bootgraph.pl
  - Converts dmesg output into an SVG showing function timing
  - Requires `CONFIG_PRINTK_TIME=y` and `initcall_debug` option
- tools/power/pm-graph/bootgraph.py
  - Generates an HTML kernel boot timeline up to init
  - Reads dmesg output, requires `initcall_debug` option
  - Supports ftrace w/ function_graph tracer and pre-defined configuration

Boot Events

- Tracking time for key boot events/functions
  - Kernel log
  - ftrace
- Identify critical events/functions to trace
  - All initcalls
  - Subset of specific (critical) events
    - See upcoming presentation on boot phases:

      Initiatives in boot time reduction - boot time markers, boot phases and profile-guided optimizations @ LPC2024

Automated Boot Time Test

- Goal
  - Detect regressions in the boot time
- Requirements
  - Upstream test
    - Limited to the pre-init phase
  - Minimal dependencies and maintenance
  - Generic and CI-compatible

Automated Boot Time Test

- Test elements
  - Way to trace key boot events
    - Kernel log, ftrace
  - Parser to extract the timestamps
  - Slowdown detection mechanism
    - Reference values from previous known good boot
    - Variance allowed in start/end time or duration (universal or per-event)

Automated Boot Time Test

- Proposed Kselftest
  - [RFC PATCH 0/1] Add kselftest to detect boot event slowdowns
- Approach
  - Configure ftrace to track timings for specific boot events
  - Compare their timestamps against reference values provided in YAML format
  - Flag any deviation beyond a specified delta

Automated Boot Time Test

- **tools/testing/selftests/boot-time/bootconfig**: configures ftrace and lists key boot events
  - Highly flexible, avoids editing the cmdline manually for the test
  - See also: https://www.kernel.org/doc/html/latest/trace/boottime-trace.html

```
ftrace {
    event.kprobes {
        populate_rootfs_begin.probes = "populate_rootfs"
        unpack_to_rootfs_begin.probes = "unpack_to_rootfs"
        run_init_process_begin.probes = "run_init_process"
        run_init_process_end.probes = "run_init_process%return"
    }
}
```

Automated Boot Time Test

- **tools/testing/selftests/boot-time/config**: enables boot time tracing and attaches the bootconfig file to the kernel
  - See also: https://www.kernel.org/doc/html/latest/trace/boottime-trace.html

```
CONFIG_TRACING=y
CONFIG_BOOTTIME_TRACING=y
CONFIG_BOOT_CONFIG_EMBED=y
CONFIG_BOOT_CONFIG_EMBED_FILE="tools/testing/selftests/boot-time/bootconfig"
```

Automated Boot Time Test

- **tools/testing/selftests/boot-time/kprobe_timestamps_to_yaml.py**: extracts event names and timestamps from the trace and writes them to a YAML file (run once on a known good kernel)

```
$ ./kprobe_timestamps_to_yaml.py kprobe-timestamps.yaml
debugfs is already mounted at /sys/kernel/debug
Generated kprobe-timestamps.yaml
```

```
$ cat kprobe-timestamps.yaml
populate_rootfs_begin: 0.438616
run_init_process_begin: 7.549203
run_init_process_end: 7.553013
unpack_to_rootfs_begin: 0.438799
```

Automated Boot Time Test

- **tools/testing/selftests/boot-time/test_boot_time.py**: compares current trace timestamps against YAML reference, reports any deviation beyond a specified delta

```
$ ./tools/testing/selftests/boot-time/kprobe_timestamps_to_yaml.py kprobe-timestamps.yaml 1
debugfs is already mounted at /sys/kernel/debug
TAP version 13
1..4
ok 1 populate_rootfs_begin
ok 2 run_init_process_begin
ok 3 run_init_process_end
ok 4 unpack_to_rootfs_begin
# Totals: pass:4 fail:0 xfail:0 xpass:0 skip:0 error:0
```

Automated Boot Time Test

- **tools/testing/selftests/boot-time/test_boot_time.py**: compares current trace timestamps against YAML reference, reports any deviation beyond a specified delta

```
$ ./tools/testing/selftests/boot-time/kprobe_timestamps_to_yaml.py kprobe-timestamps.yaml 1
debugfs is already mounted at /sys/kernel/debug
TAP version 13
1..4
ok 1 populate_rootfs_begin
# 'run_init_process_begin' differs by 2.705185 seconds.
not ok 2 run_init_process_begin
# 'run_init_process_end' differs by 2.705190 seconds.
not ok 3 run_init_process_end
ok 4 unpack_to_rootfs_begin
# Totals: pass:2 fail:2 xfail:0 xpass:0 skip:0 error:0
```

Feedback Received

- Reuse available tools, e.g. tools/power/pm-graph/bootgraph.py
  - Need to add machine-readable output (e.g. JSON, YAML)
  - Timestamp parser might need some adjustments, depending on the events being traced (e.g. kprobes)

Help Needed

- **Identify key boot events/functions to trace**
  - How would you define “boot” based on your use cases? Should we track subset of critical events or all initcalls?
    - Determines ftrace config and amount of test output
- **Results format**
  - Tracking initcalls => great amount of PASS/FAIL/SKIP statuses. Show failures only?
- **Thoughts on reusing bootgraph.py**
  - Need to change output format and adjust parser
- **Determine location/format for reference values file (out-of-tree)**
  - Other tests need reference values (e.g. devices/probe, devices/exist)
    - Towards common mainline device testing @ LPC2024
    - Adding benchmark results support to KTAP/kselftest @ LPC2024
- **Variance between subsequent runs**
  - Per event or universal?
- **Focus on duration or start/end times?**
  - Using start/end times might cause cascade of errors
  - Using duration => more tracepoints

Thank you!

We are hiring

col.la/careers