Intel GFX CI

What services do we provide, our roadmaps, and lessons learnt!

Martin Peres

Sept 20th 2017

Agenda

- Introduction: Why CI, and objectives
- State of Intel GFX CI, and future plans
- Lessons learnt

Why do we need Continuous Integration (CI)?

- CI allows putting the cost of integration on the person making changes:
  - It scales better with the number of developers!
  - Less time spent on bug fixing in post merge
  - Provides better global understanding to developers
- CI keeps the integration tree in working condition at all time

Objectives of CI

- Provides an accurate view of the state of the HW/SW
- Results should be:
  - Transparent: Should contain the full HW and SW configuration
  - Fast: Basic results in under 30 minutes, complete ones in half a day
  - Visible: make the results public and hard to miss (reply in ML)
  - Stable: noise level should be zero (be aggressive at blacklisting unstable tests)

Intel GFX CI

intel Software

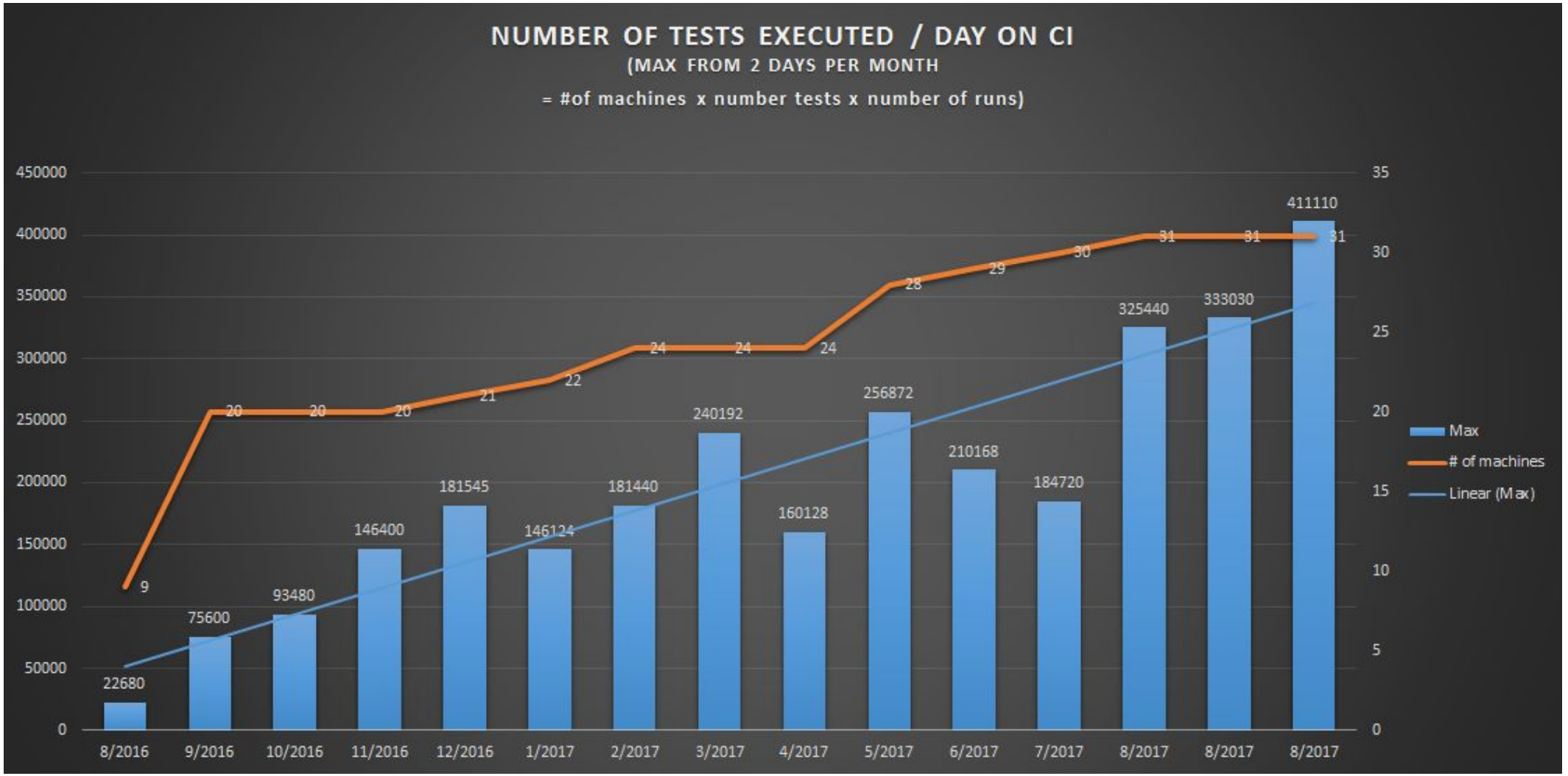
INTEL OpenSource TECHNOLOGY CENTER

Intel GFX CI - https://intel-gfx-ci.01.org

Current state

- Provide timely, public, stable and transparent results for:
  - Trees:
    - Pre-merge: DRM-tip, IGT
    - Post-merge: DRM-tip, Linus' tree, Linux-next, *-fixes, drm-internal
  - Machines (total of 40 systems / 19 different platforms (Gen 3 to current))
    - GDG (Gen3, 2004) -> GLK (not released yet)
    - Sharded machines: 6 KBL, 6 HSW, 6 SNB, 8 APL
    - SKL Xeon
    - GVT-d BDW and SKL (Virtualization)
  - Test suites:
    - IGT:
      - Fast-feedback: 279 tests, ran on all machines
      - Full KMS + some GEM tests: ~2500 tests, ran on sharded machines
  - Throughput
    - From 22k tests/day (Aug 2016) to +400k tests/day (Aug 2017) (see next slide)
    - Bug filing: usually under 1h during working hours

CI throughput per day (from 08/2016 until today)



Intel-GFX CI: Roadmap

Plans

- Provide timely, visible, stable and transparent results for:
- Machines:
  - Keep adding new platforms / hardware configurations
  - More display types (including chamelium)
- Test suites:
  - Full IGT on all machines. Requires:
    - Developers to improve IGT to run in < 6 hours (kms, gem, prime)
    - Squashing all patch series in one tree
    - Auto-bisect issues to the offending patch series
  - Performance and rendering. Requires:
    - EzBench support
    - Better prioritization of tasks for machine time

Contacts

Tomi Sarvela

- Infrastructure and most of the automation software

Martin Peres

- Ezbench, CI bug log, bug filing

Arkadiusz Hiler

- IGT maintainer, back up for Tomi, Pre-silicon CI

Petri Latvala

- IGT maintainer, Ezbench support

Lessons learnt

Key findings to replicate our system

- What is not tested continuously is broken
- Bugzilla is not a good tool to track test failures
- Noise is the enemy #1:
  - Treat every failure as a bug
  - Run tests in a loop
  - Collect failure statistics and history!
- Make sure developers own the CI system
  - The CI team works for developers
  - Developers suggest improvements to the systems and improve test suites
- Have automated metrics for everything!
- Took us a year to get the basic IGT testing stable on 2004+ hardware

What is needed for HW CI

- Requirements for making a useful CI system:
  - Infrastructure:
    - Physical space
    - Enough power and cooling
    - Power cutters for all machines
    - Reliable network (the simpler the better)
  - Hardware:
    - Machines with different configurations (chipsets, RAM, connectors, screens)
    - Ways to resume the machine (RTC wake, …)
  - Software:
    - Scheduling jobs (Jenkins, ...)
    - Graphics stack compilation automation
    - Automatic deployment and reboot
    - External watchdog
  - Humans:
    - Qualified engineers to make bugs
    - Developers to act quickly on bug reports

Challenges of doing kernel CI

- Booting garbage kernels:
  - Boot, network, and/or filesystem broken
- Getting traces out, especially during suspend/resume:
  - Kernel parameters: use “nmi_watchdog=panic,auto panic=1 softdog.soft_panic=1”
  - Use pstore for EFI-capable HW, serial consoles for others
- Dealing with memory corruptions:
  - Will trash your partitions
  - Need automated script to re-deploy machines

CI Bootstrapping

- Step 0: Gather hardware, and test suites
- Step 1: Run the test suites automatically on this hardware
- Step 2: Report failures to a tool that will check if the failure is known
- Step 3: File bugs about unknown failures
- Step 4: When no new failure happen for some time, add to pre-merge
- Step 5: Goto step 0

Conclusion

CI is good!

Join us, and let's collaborate!

Questions / discussion