Intel GFX CI and IGT

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

Martin Peres & Arek Hiler

Feb 3rd 2018

Agenda

- Introduction: Linux and its need for CI
- IGT GPU Tools - our testsuite
- State of Intel GFX CI, and future plans
- Lessons learnt
- Dealing with Linux in products

Linux and its unique development model

- The Linux kernel is massive:
  - 63 to 70 days between releases
  - 14k commits per release
  - 9 commits per hour in average in the main tree
  - ~1500 developers, 10+% of hobbyists and 250 companies (Intel #1)
  - ~25M lines of code
  - 100s of integration trees and 6 stable trees

Linux and its unique development model

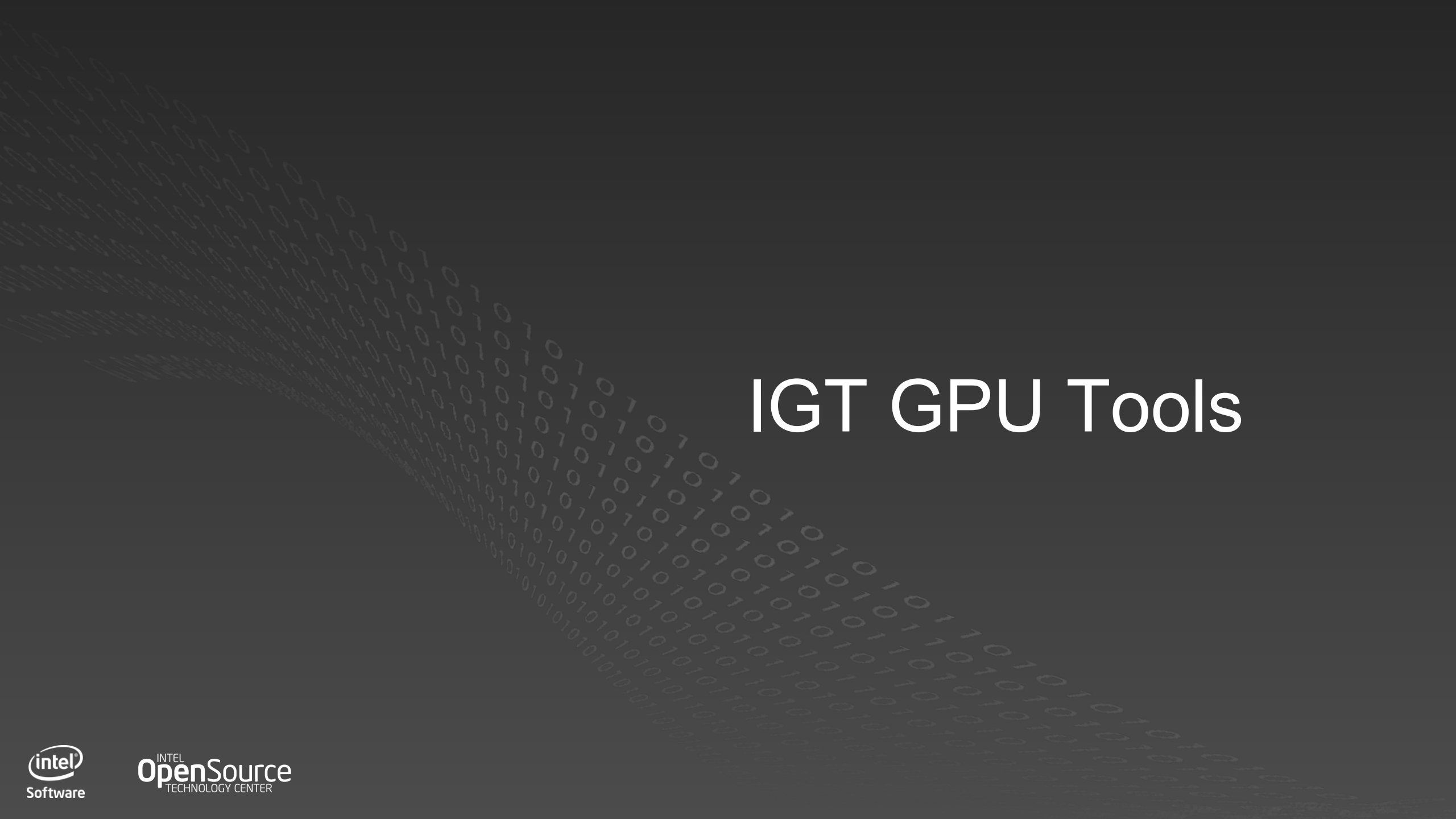
- The Linux kernel has no architects, but it has rules:
  - No user-visible regression: if updating breaks a program, the change is reverted.
  - Kernel changes need to be open source.
  - No new kernel feature without an open source userspace (especially true for DRM).

Why do we need Continuous Integration (CI)?

- Pre-merge testing allows putting the cost of integration on the person making changes:
  - less time spent on bug fixing in post merge (where reverts are hard to get accepted);
  - provides better global understanding to developers;
  - keeps the integration tree in working condition at all time;
  - it scales better with the number of developers!

- Challenges:
  - Keeping the integration tree working is difficult:
    - back merges from Linux bring thousands of line of code without integration testing.
  - Flowing fixes to stable branches may also break them:
    - requires testing the integration of patches for stable trees too.

IGT GPU Tools



intel Software

INTEL OpenSource TECHNOLOGY CENTER

IGT GPU Tools

What is it?

- a collection of tools for development and testing of the DRM drivers
- (actually mostly tests)

What has changed?

- the name (previously Intel GPU Tools)
- mailing list (intel-gfx@fdo -> igt-dev@fdo)
- autotools -> meson

IGT Tests

```
% ./run-tests.sh -l | wc -l
61572(-ish)

% ./run-tests.sh -l | grep amd | wc -l
18

% ./run-tests.sh -l | grep vc4 | wc -l
27

% ./run-tests.sh -l | grep kms | wc -l
1546

% ./run-tests.sh -l | grep gem | wc -l
2379 (59499 with gem_concurrent)
```

IGT: More Than Intel

Why other drivers?

- because they are DRM too
- because KMS is not driver specific
- because APIs have to be consistent across vendors
- because why duplicate effort?

What has to be done?

- better separation of Intel code
- handling multiple GPUs per host

Running With Non-Intel Drivers

Nouveau		**VC4**		**NVIDIA**	
pass:	125	pass:	118	pass:	20
fail:	77	fail:	102	fail:	510
skip:	4179	skip:	4184	skip:	3887
warn:	36	warn:	2	warn:	2
		timeout:	4		
		dmesg-warn:	2		
		dmesg-fail:	5		
total:	4417	total:	4417	total:	4417

A lot of unnecessary kms skips/fails because of Intel-isms = a lot of low hanging fruits.

Intel GFX CI

intel Software

INTEL OpenSource TECHNOLOGY CENTER

Objectives of Intel-GFX-CI

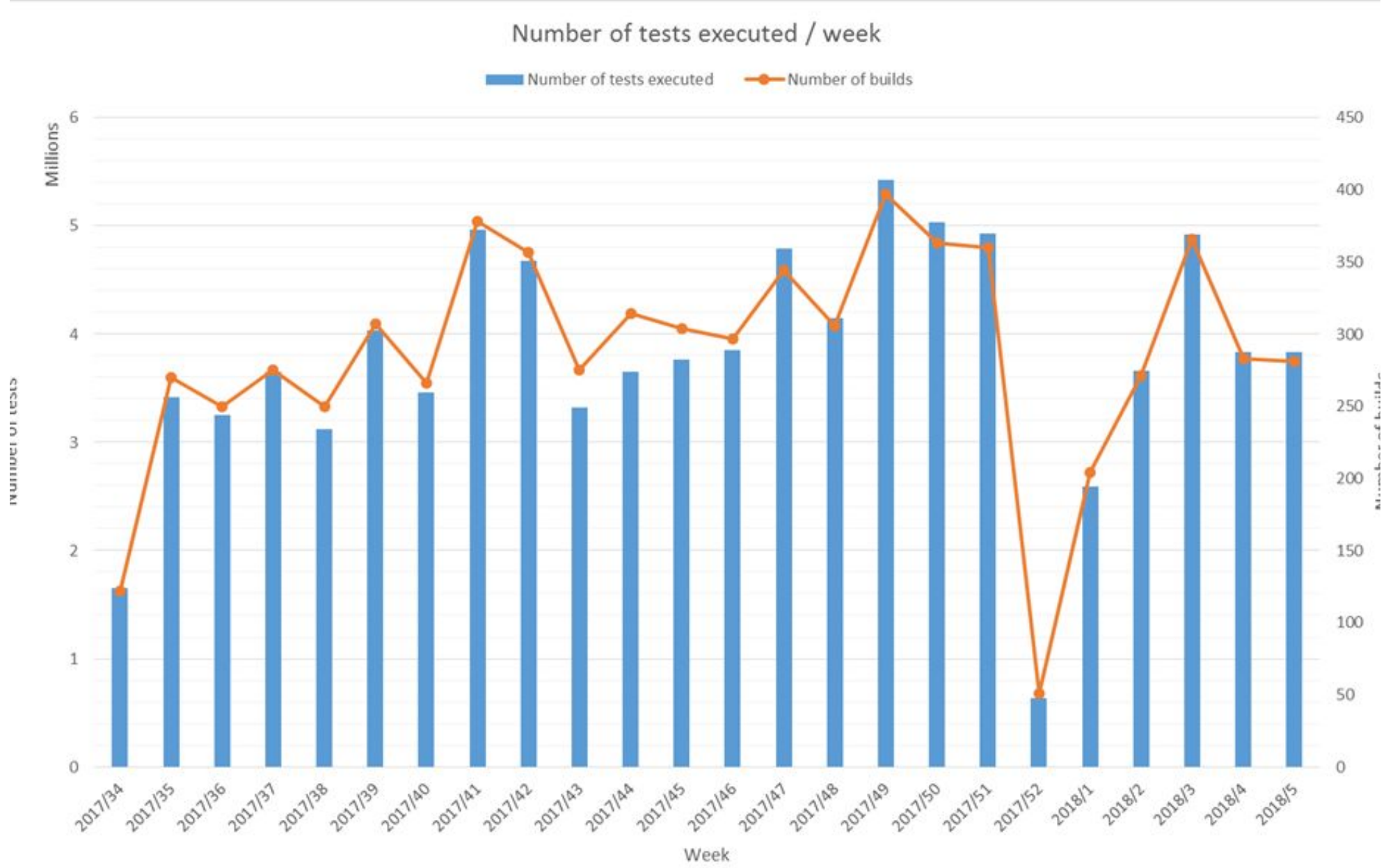
- Provide an accurate view of the state of the HW/SW (all supported combinations).

- 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);

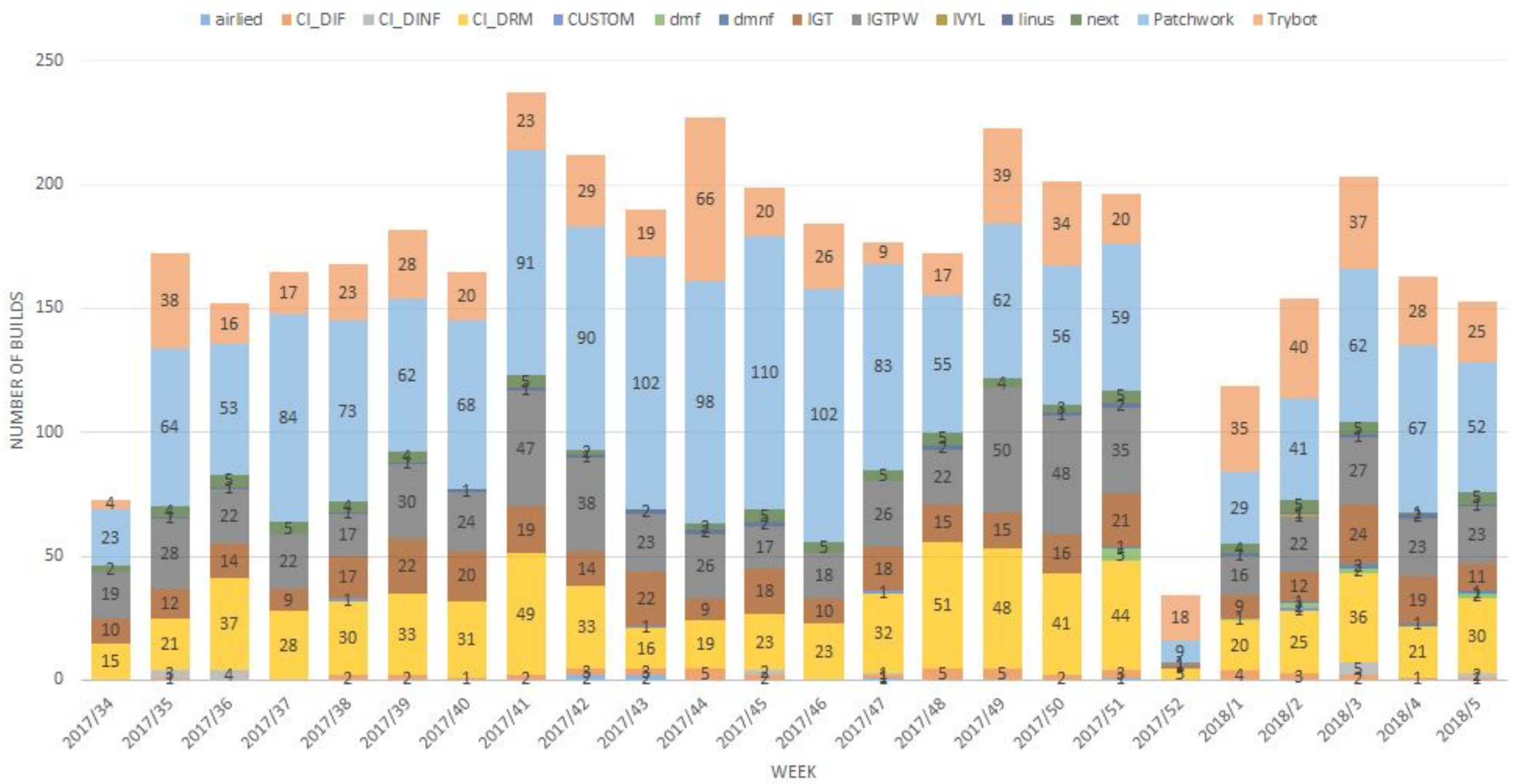
Intel GFX CI - https://intel-gfx-ci.01.org

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, Dave Airlie's branch
- Machines (total of 74 systems / 21 different platforms (Gen 3 to upcoming Gens)):
  - GDG (Gen3, 2004) -> CNL (not released yet)
  - sharded machines: 7 KBL, 8 HSW, 7 SNB, 8 APL, 6 GLK
  - SKL Xeon
  - GVT-d BDW and SKL (Virtualization)
- Displays interfaces: HDMI, DVI, DP, eDP, DP-MST, DSI, TB, LVDS
- Test suites - IGT:
  - fast-feedback: 288 tests, ran on all machines
  - full KMS + some GEM tests: ~2700 tests, ran on sharded machines
- Throughput
  - from 22k tests/day (Aug 2016) to +850k tests/day (now)
  - bug filing: usually under half a day during working hours


Number of tests executed / week
Number of tests executed
Number of builds
Millions
Number of tests
Number of builds
Week
0
1
2
3
4
5
6
0
50
100
150
200
250
300
350
400
450
2017/34
2017/35
2017/36
2017/37
2017/38
2017/39
2017/40
2017/41
2017/42
2017/43
2017/44
2017/45
2017/46
2017/47
2017/48
2017/49
2017/50
2017/51
2017/52
2018/1
2018/2
2018/3
2018/4
2018/5

NUMBER OF BUILDS RUN / WEEK



Intel-GFX CI: Roadmap

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

Intel-GFX CI: New tools

New tools about to be deployed:

- **CI Bug Log NG:** a missing link between bug tracking and execution results
  - matches failures to known issues, reducing noise in pre-merge
  - helps with bug filing and tracking
  - is a reimplementation of the original CI Bug Log

- **EzBench:** auto-bisection of changes in performance, rendering, and unit tests
  - takes care of the variance in results
  - needs more work to get multi-component deployment and bisection

Intel-GFX CI: Let’s collaborate!

- **Self Tests:** If you have Linux self tests that are somewhat related to graphics, network, sound, or suspend, we can run some of those tests in our farm!

- **IGT:** Please contribute new tests for KMS and/or your driver!

- **Infrastructure**: We are looking into Open Sourcing our CI tools!

Contacts

Tomi Sarvela

- Infrastructure and most of the automation software

Arkadiusz Hiler

- IGT and FDO's Patchwork maintainer, back up for Tomi

Martin Peres

- Ezbench and CI bug log maintainer, Bug filing (secondary)

Marta Löfstedt

- Main bug filer, IGT/i915 developer

Petri Latvala

- IGT maintainer, Ezbench

Questions / discussion

IGT - The Low Hanging Fruits

- kms_busy, kms_color, kms_draw_crc, kms_frontbuffer_tracking and perf_pmu do useless modeset just to skip

Lessons learnt

Key findings to replicate our system

- What is not tested continuously is broken.
- Bug trackers are 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, ...);
  - components' compilation automation;
  - automatic deployment and reboot;
  - external watchdog.
- Humans:
  - good lab engineer to maintain the infrastructure;
  - qualified engineers to file 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

Linux in products

Using Linux in products

- Most products using Linux have outdated kernel
  - your phone is likely using Linux 3.10 (June 2013);
  - Linux 3.10.108 is the latest released (November 2017);
  - Linux 4.14 is the latest major version (24 major versions after 3.10).

- Upstream integration reduces your product's TTM and increase security:
  - see https://wtarreau.blogspot.com/2017/11/look-back-to-end-of-life-lts-kernel-310.html
  - see https://phd.mupuf.org/files/xdc2017_upstream_dev.pdf

Conclusion

CI makes upstream development easier!