EzBench, a tool to help you benchmark and bisect the Graphics Stack's performance

Martin Peres

Intel Open Source Technology Center Finland

September 23, 2016

Summary

Introduction

Current situation

- Complex games/benchmarks are available and used on Linux;
- Drivers are getting more complex as performance improves;
- Users now rely on Open Source drivers for games/apps...

Introduction

Current situation

- Complex games/benchmarks are available and used on Linux;
- Drivers are getting more complex as performance improves;
- Users now rely on Open Source drivers for games/apps...

Risks when merging new code

- Break previous functionalities / rendering;
- Break the performance of a game inadvertly;
- Improve the performance of one app but slow down others.

Introduction

Current situation

- Complex games/benchmarks are available and used on Linux;
- Drivers are getting more complex as performance improves;
- Users now rely on Open Source drivers for games/apps...

Risks when merging new code

- Break previous functionalities / rendering;
- Break the performance of a game inadvertly;
- Improve the performance of one app but slow down others.

$\Rightarrow$ Need to test and benchmark all the platforms and games of interest.

Summary

CI: Objectives and chalenges

Objectives

- Catch changes in unit tests, rendering, performance or power;
- Pin-point the change, to help bug-reporting and fixing;
- Guarantee reproducibility of the results;
- Warn the relevant developers of changes.

CI: Objectives and chalenges

Objectives

- Catch changes in unit tests, rendering, performance or power;
- Pin-point the change, to help bug-reporting and fixing;
- Guarantee reproducibility of the results;
- Warn the relevant developers of changes.

Challenges

- Unit tests, performance, metrics or rendering can be unstable;
- Multiple components interacting with each-other;
- Avoid false positives and false negatives;
- Impossible to test every commit.

CI: Current tools

Current solutions

- Unit testing: Piglit, dEQP, gl-CTS, vk-CTS, more...;
- Performance: Phoronix Test Suite, Sixonix;
- Rendering: Phoronix Test Suite, Anholt's trace-db;
- Job scheduling: Phoronix Test Suite, Jenkins, ...

CI: Current tools

Current solutions

- Unit testing: Piglit, dEQP, gl-CTS, vk-CTS, more...;
- Performance: Phoronix Test Suite, Sixonix;
- Rendering: Phoronix Test Suite, Anholt's trace-db;
- Job scheduling: Phoronix Test Suite, Jenkins, ...

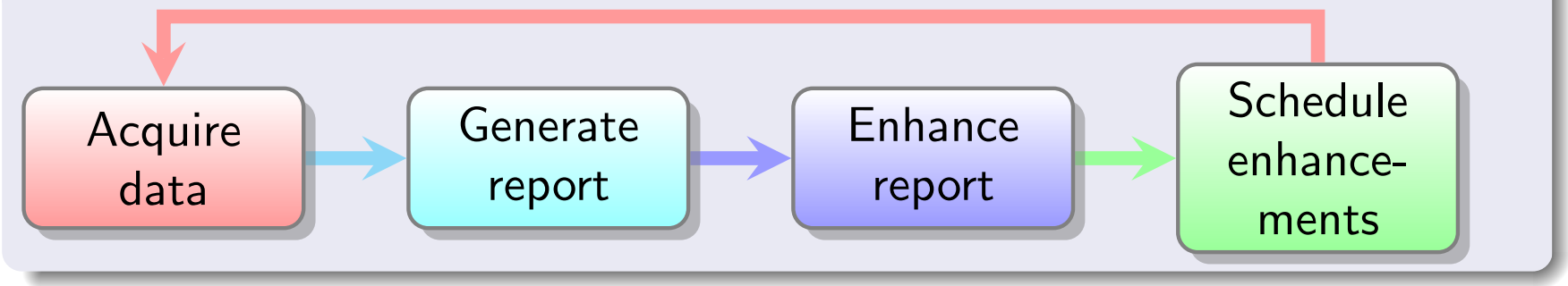
Issue: Great for reporting, not for bisecting

- No feedback loop to address variance issues;
- Environment may have changed;
- Unit tests may flip/flop;
- Rendering may be unstable (yes, it does happen);
- Solution: external runner for them to take care of this!

Summary

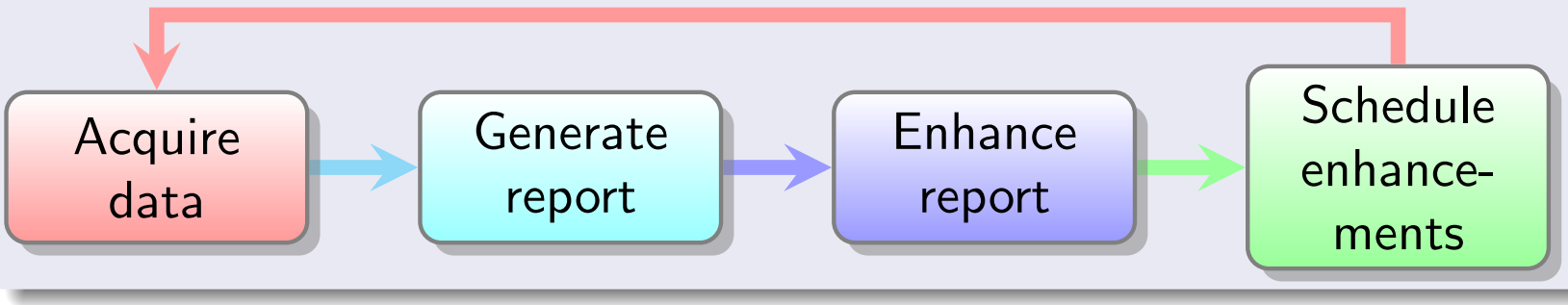
EzBench: General architecture

General flow graph



EzBench: General architecture

General flow graph



Blocks

- Acquire data: Compile/deploy, run tests and collect data/env;
- Generate report: Read from the disk, create a python IR;
- Enhance report: Analyse the data, find changes, report events;
- Schedule enhancements: Request more data (bisect!).

EzBench: Code and license

MIT-licensed code

Available at https://cgit.freedesktop.org/ezbench/

EzBench: Code and license

MIT-licensed code

Available at https://cgit.freedesktop.org/ezbench/

Blocks

- runner: bash-based, handles:
  - compilation and deployment of the component;
  - setting up the environment (X, compositor);
  - running the test.

EzBench: Code and license

MIT-licensed code

Available at https://cgit.freedesktop.org/ezbench/

Blocks

- runner: bash-based, handles:
  - compilation and deployment of the component;
  - setting up the environment (X, compositor);
  - running the test.
- env-dump.so: LD_PRELOADed C library:
  - dump the environments and loaded libs;
  - hook interesting calls (GLX, EGL, GL, X);
  - dump metrics (RAPL, GPU temperature and power usage).

EzBench: Code and license

MIT-licensed code

Available at https://cgit.freedesktop.org/ezbench/

Blocks

- runner: bash-based, handles:
  - compilation and deployment of the component;
  - setting up the environment (X, compositor);
  - running the test.
- env-dump.so: LD_PRELOADed C library:
  - dump the environments and loaded libs;
  - hook interesting calls (GLX, EGL, GL, X);
  - dump metrics (RAPL, GPU temperature and power usage).
- Report generation, enhancing and scheduling: python daemon;
- Reporting: python script generating an HTML file.

EzBench: Features

Features

- Supports:
  - Unit tests: Piglit, dEQP, IGT (WIP);
  - Benchmarks: GPUTest, Unigine, GFX Bench (corporate), ...;
  - Rendering: Apitrace.
- Acquires environment information, for catching changes;
- Analyses variance on data and reproduces changes;
- Auto-bisecting on data, metrics are WIP.

EzBench: Features

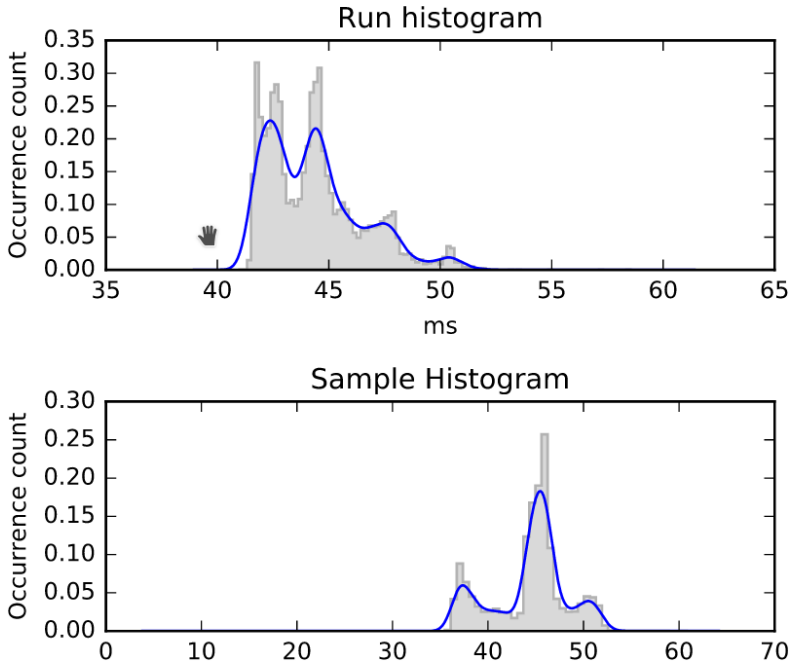
Features

- Supports:
  - Unit tests: Piglit, dEQP, IGT (WIP);
  - Benchmarks: GPUTest, Unigine, GFX Bench (corporate), ...;
  - Rendering: Apitrace.
- Acquires environment information, for catching changes;
- Analyses variance on data and reproduces changes;
- Auto-bisecting on data, metrics are WIP.

Profiles

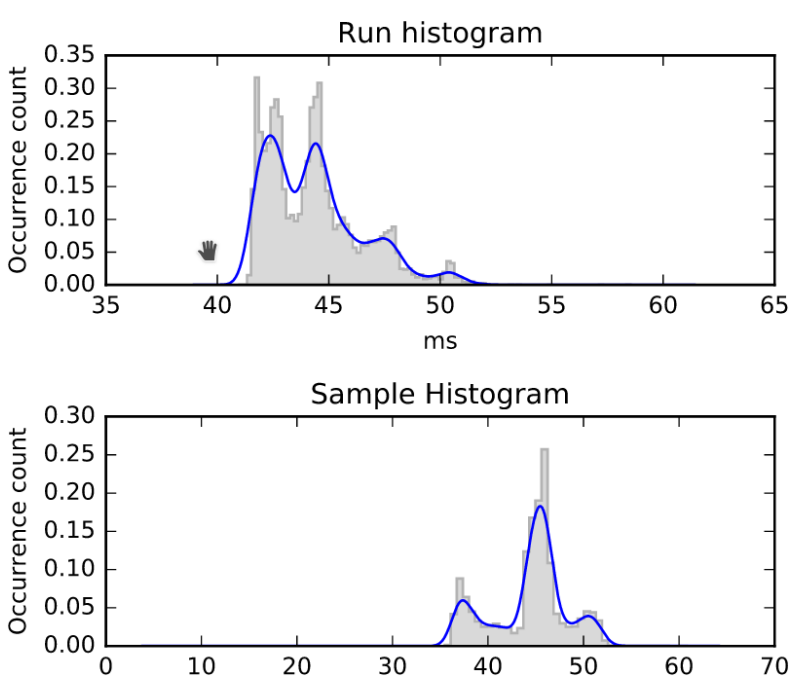
- Mesa: No limitations;
- xf86-video-intel: No limitations;
- Linux: may require an external watchdog.

Examples of variance

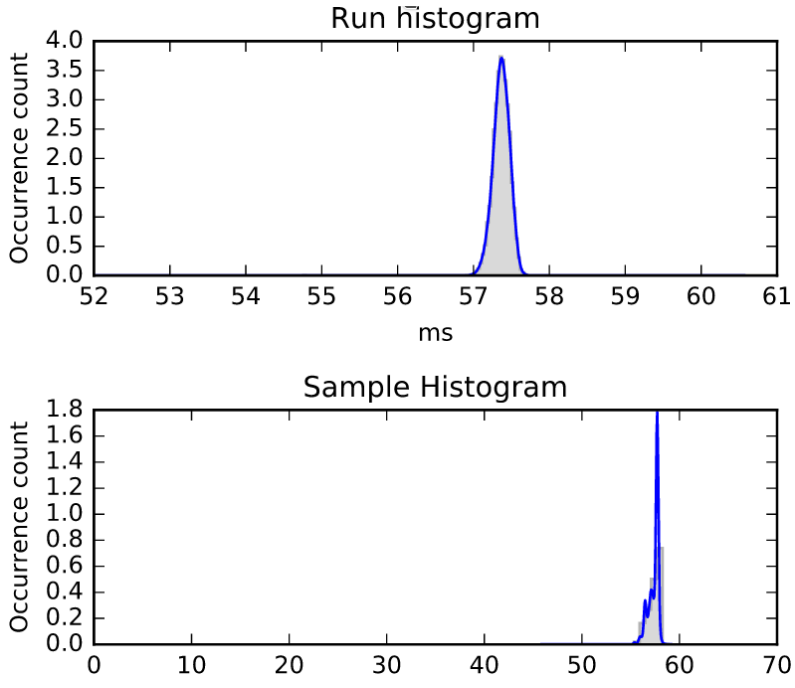


(a) Bad FPS distribution

Examples of variance



(a) Bad FPS distribution



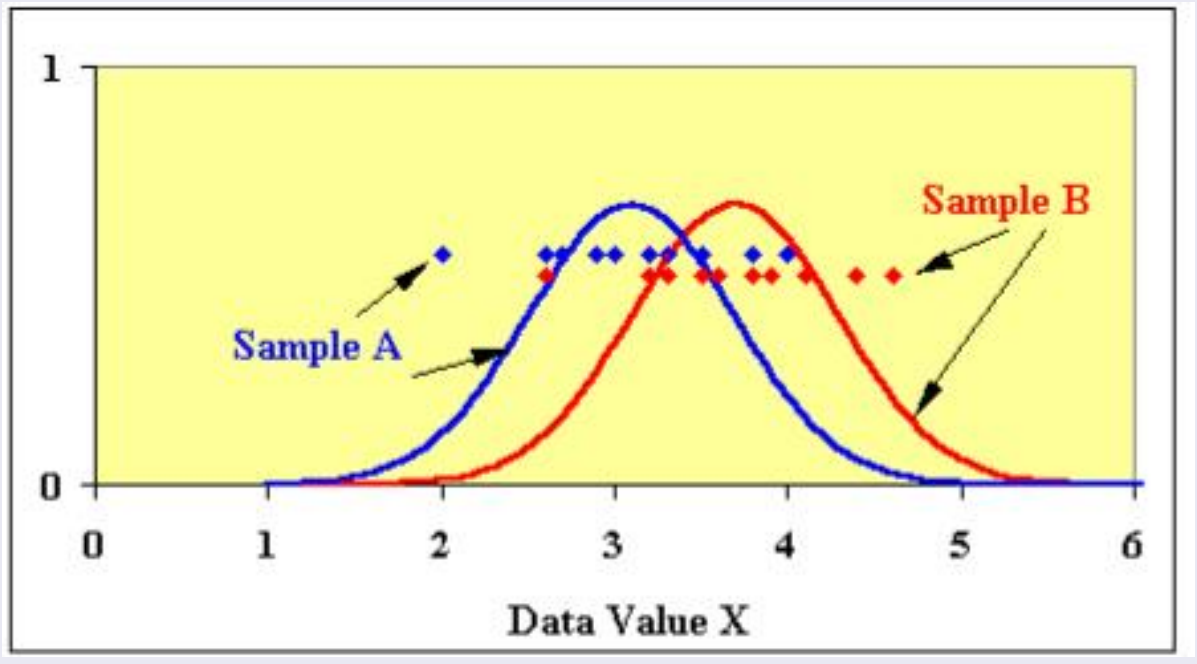
(b) Good FPS distribution

Figure: Examples of variance

EzBench: Handling variance

Student-T test

Check if two data sets belong to the same normal distribution.



Source: http://serc.carleton.edu/introgeo/teachingwdata/Ttest.html

EzBench: Image comparaison

Overview

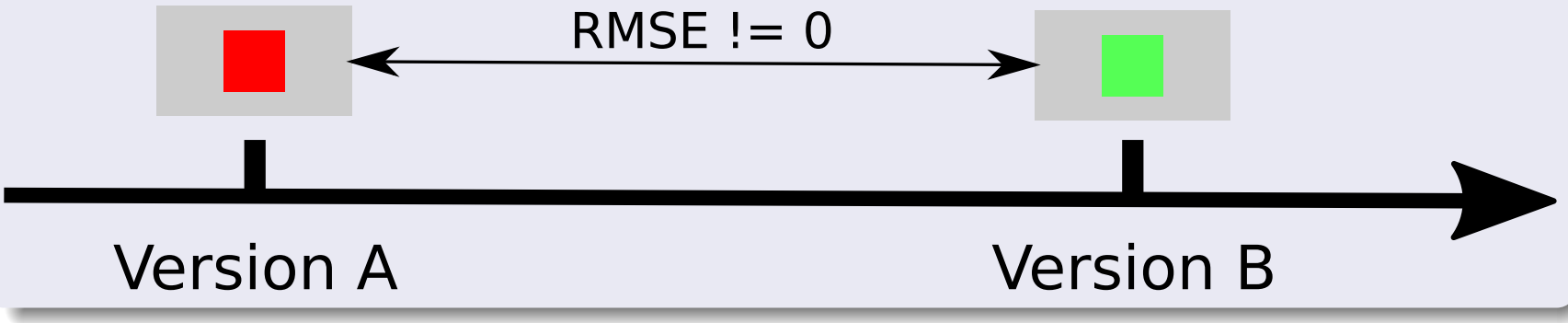
- Contributed by Pekka Jylhä-Ollila (Intel);
- Comparaison done using RMSE and requires 3 steps.

EzBench: Image comparaison

Overview

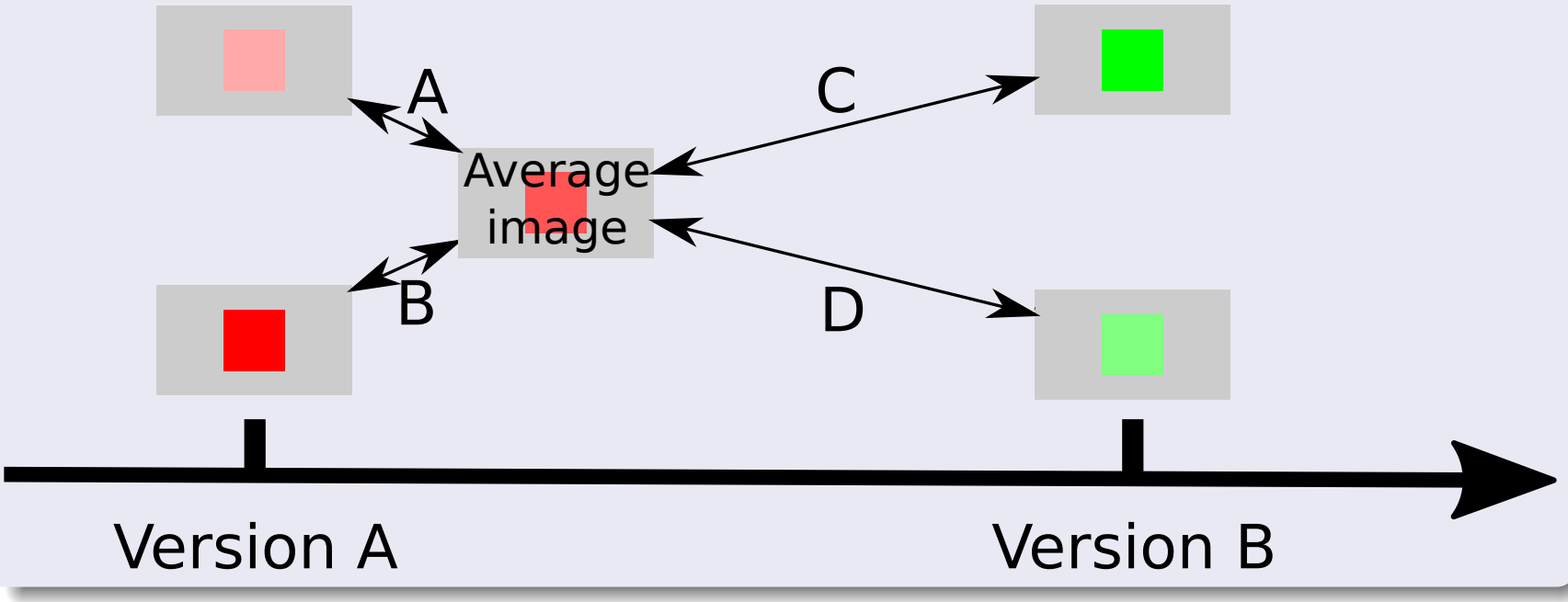
- Contributed by Pekka Jylhä-Ollila (Intel);
- Comparaison done using RMSE and requires 3 steps.

Step 1: Comparing the output of 2 versions



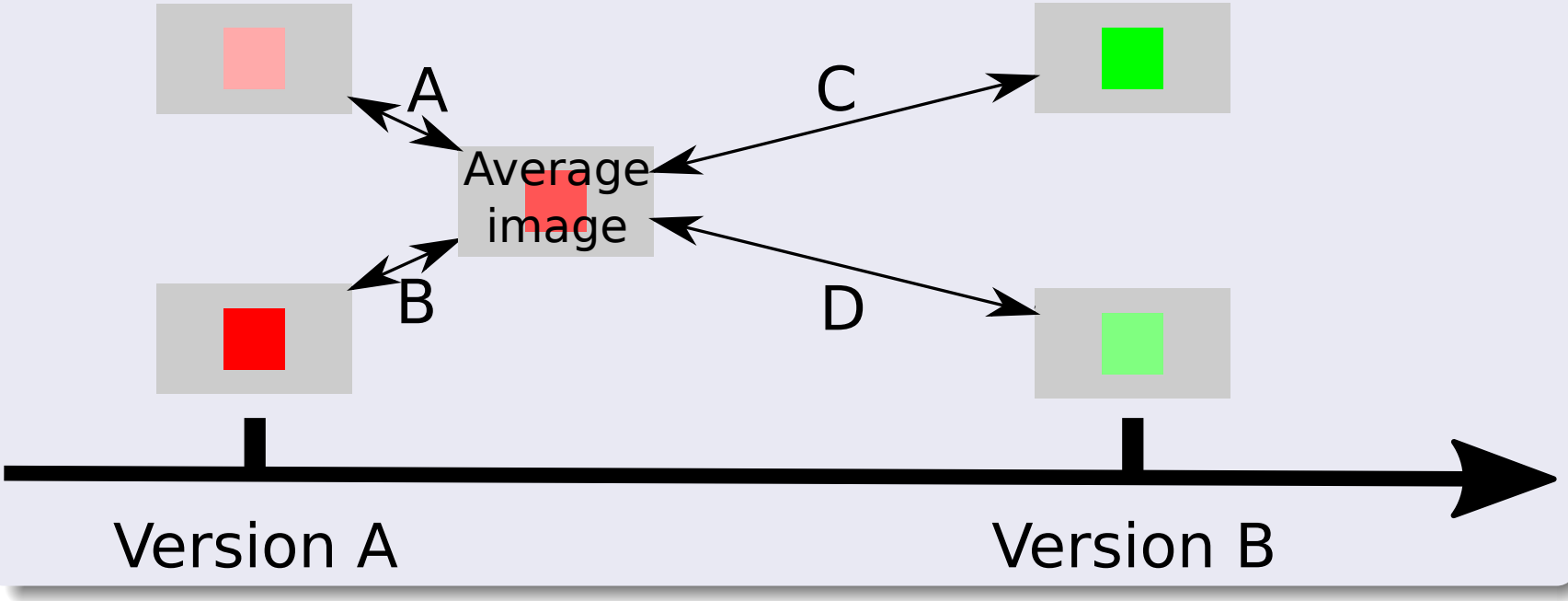
EzBench: Image comparaison

Step 2: Acquire more data and generate averages



EzBench: Image comparaison

Step 2: Acquire more data and generate averages



Step 3: Use the student-t test on the RMSEs

EzBench: Demo time

Demo 1: running loads with the simple runner

- Listing tests;
- Running gtkperf in different environments;
- Showing the generated report;
- Start compiling a new version of mesa.

EzBench: Demo time

Demo 1: running loads with the simple runner

- Listing tests;
- Running gtkperf in different environments;
- Showing the generated report;
- Start compiling a new version of mesa.

Demo 2: Actual reports

- Auto-bisected rendering change (5k commits, 7 months);
- Running gtkperf in different environments;
- Showing the generated report;
- Start compiling a new version of mesa.

EzBench: Needed features for CI

Randomized testing

- Not all tests can be run every day;
- Tests should be added randomly (as time permits);

EzBench: Needed features for CI

Randomized testing

- Not all tests can be run every day;
- Tests should be added randomly (as time permits);

Support changing multiple components at the same time

- EzBench needs to find the component that made the change;
- It thus needs to group data per environment;
- It needs to merge data from similar environments;
- It needs to be able to re-deploy environments;
- It needs to be able to recompile important components.

Summary

Conclusion

Ezbench's Goals

- Automatically annotate a git tree with:
  - Unit test results;
  - Power and performance results;
  - Rendering changes.
- Require as little human intervention as possible;
- Provide reproducible results (environment).

Conclusion

Ezbench's Goals

- Automatically annotate a git tree with:
  - Unit test results;
  - Power and performance results;
  - Rendering changes.
- Require as little human intervention as possible;
- Provide reproducible results (environment).

EzBench tries to take care of the pitfalls of benchmarking

- Environment dumping and diffing;
- Reproduces results and tries to handle variance;
- Is reactive to changes, and self-improving;
- Handles most of the testing automatically.

Questions?

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;
- Bisects performance changes automatically;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;
- Bisects performance changes automatically;
- Provides python bindings to acquire data and parse reports;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;
- Bisects performance changes automatically;
- Provides python bindings to acquire data and parse reports;
- Be crash-resistant by storing the expected goal and comparing it to the current state;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;
- Bisects performance changes automatically;
- Provides python bindings to acquire data and parse reports;
- Be crash-resistant by storing the expected goal and comparing it to the current state;
- Collect the environment information and diff it;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;
- Bisects performance changes automatically;
- Provides python bindings to acquire data and parse reports;
- Be crash-resistant by storing the expected goal and comparing it to the current state;
- Collect the environment information and diff it;
- Detect the variance and peformance changes;

EzBench - Features

Current features

- Modular architecture (profiles, tests and user hooks);
- Automates the acquisition of benchmark data;
- Knows how long it is going to take;
- Generates a report that is usable by developers;
- Bisects performance changes automatically;
- Provides python bindings to acquire data and parse reports;
- Be crash-resistant by storing the expected goal and comparing it to the current state;
- Collect the environment information and diff it;
- Detect the variance and peformance changes;
- Automatically schedule more work to improve the report.

EzBench - Features

TODO

- Watchdog support;

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);
- Add support for PTS as a backend;

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);
- Add support for PTS as a backend;
- Better integrate the build process;

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);
- Add support for PTS as a backend;
- Better integrate the build process;
- React to HW events such as throttling;

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);
- Add support for PTS as a backend;
- Better integrate the build process;
- React to HW events such as throttling;
- Reset the environment to a previous state;

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);
- Add support for PTS as a backend;
- Better integrate the build process;
- React to HW events such as throttling;
- Reset the environment to a previous state;
- Integrate with patchwork to test patch series;

EzBench - Features

TODO

- Watchdog support;
- Handle kernel boot failures (need the watchdog);
- Add support for PTS as a backend;
- Better integrate the build process;
- React to HW events such as throttling;
- Reset the environment to a previous state;
- Integrate with patchwork to test patch series;
- Support sending emails to the authors of changes.