Performance Testing for GDB

Yao Qi
yao@codesourcery.com

CodeSourcery/MentorGraphics

2014-09-13

Introduction

- We need something to test and measure GDB perf first,
- No standard mechanism to show the performance,

- Proposed the perf testing framework in Aug and Sep 2013,
- Write some perf test cases and patches for perf improvements from Sep to Nov.

source:accurate-measurement.jpg

"if you cannot measure it, you cannot improve it."

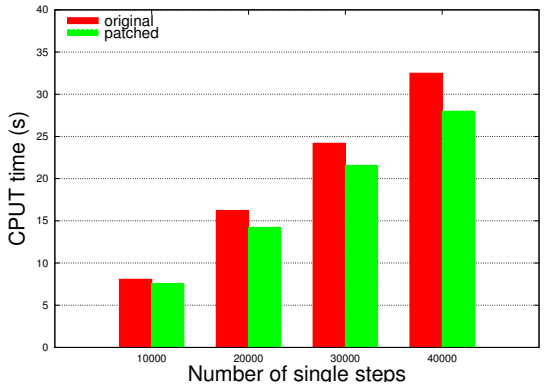
Lord Kelvin

Thank Doug and Sanimir for the comments and patient review.

Goals

- Know the new perf testing framework in GDB and how to write perf test cases on top of it,

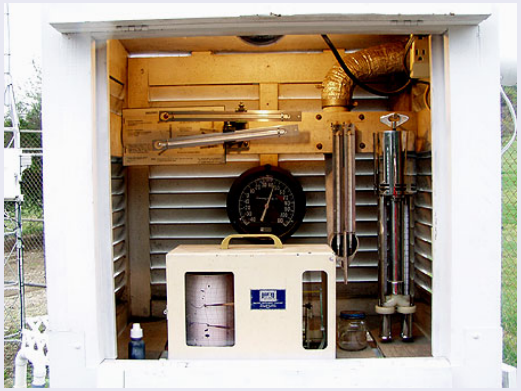
- Demonstrate how do they show the perf improvements,



- Show how do we use it to track GDB performance

Requirements

- easy to write test case for a certain aspect, i.e. symbol lookup or single step
- basic measurements are provided (such as cpu time and memory usage), but test specific measurement can be added too.

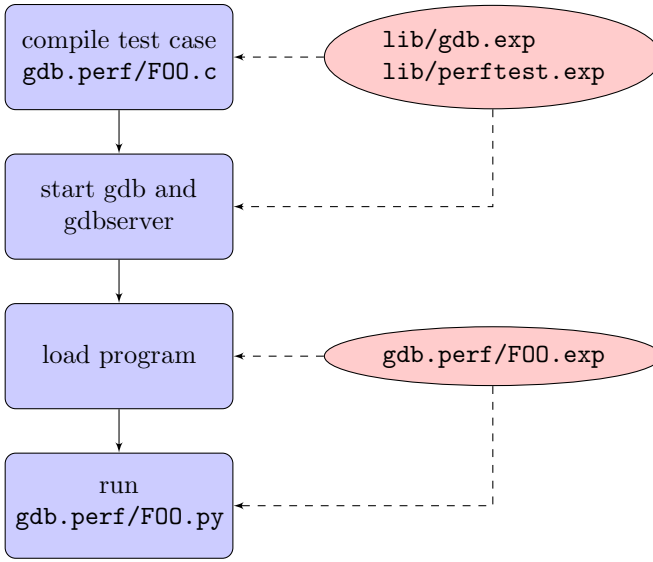


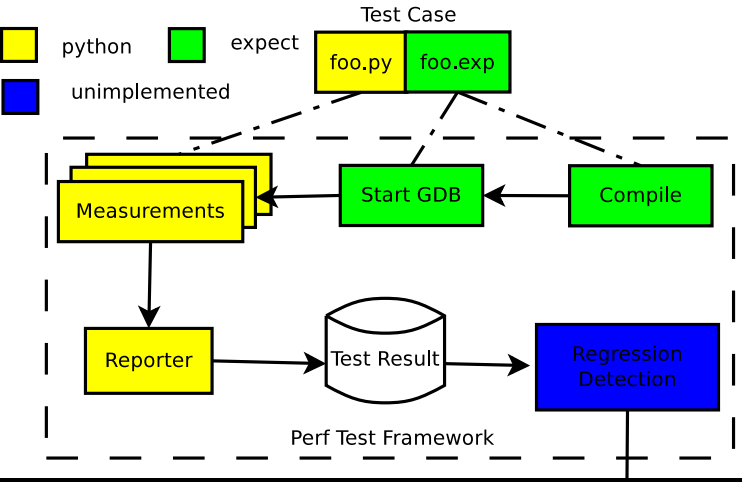
source:shelter1.jpg

- both native debugging and remote debugging are supported,
- test case has warm up optionally,
- save results in different formats,
- the executable in test case can be pre-compiled,

Design and implementation

- Use `DejaGNU` to invoke compiler and start GDB (and/or GDBserver),
- GDB loads a python script in which some operations are performed,
- The framework collects the performance data and save them in the desired format,
- Detect performance regressions,





Test case example: skip-prologue.exp

```
1  PerfTest::assemble {
2      if { [gdb_compile "$srcdir/$subdir/$srcfile" ${binfile} executable {debug}] != "" } {
3          return -1
4      }
5
6      return 0
7  } {
8      clean_restart $binfile
9
10     if ![runto_main] {
11         fail "Can't run to main"
12         return -1
13     }
14 } {
15     global SKIP_PROLOGUE_COUNT
16
17     set test "run"
18     gdb_test_multiple "python SkipPrologue\($SKIP_PROLOGUE_COUNT\).run()" $test {
19         -re "Breakpoint $decimal at \[^\n\]*\n" {
20             # GDB prints some messages on breakpoint creation.
21             # Consume the output to avoid internal buffer full.
22             exp_continue
23         }
24         -re ".*$gdb_prompt $" {
25             pass $test
26         }
27     }
28 }
```

COMPILE

START UP

RUN

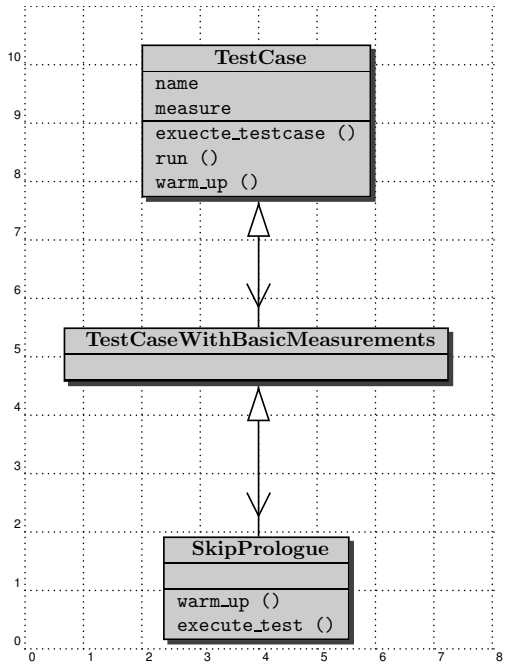
Test case example: skip-prologue.py

```
1  from perftest import perftest
2
3  class SkipPrologue(perftest.TestCaseWithBasicMeasurements):
4      def __init__(self, count):
5          super(SkipPrologue, self).__init__("skip-prologue")
6          self.count = count
7
8      def _test(self):
9          for _ in range(1, self.count):
10             # Insert breakpoints on function f1 and f2.
11             bp1 = gdb.Breakpoint("f1")
12             bp2 = gdb.Breakpoint("f2")
13             # Remove them.
14             bp1.delete()
15             bp2.delete()
16
17     def warm_up(self):
18         self._test()
19
20     def execute_test(self):
21         for i in range(1, 4):
22             gdb.execute("set code-cache off")
23             gdb.execute("set code-cache on")
24             self.measure.measure(self._test, i)
```

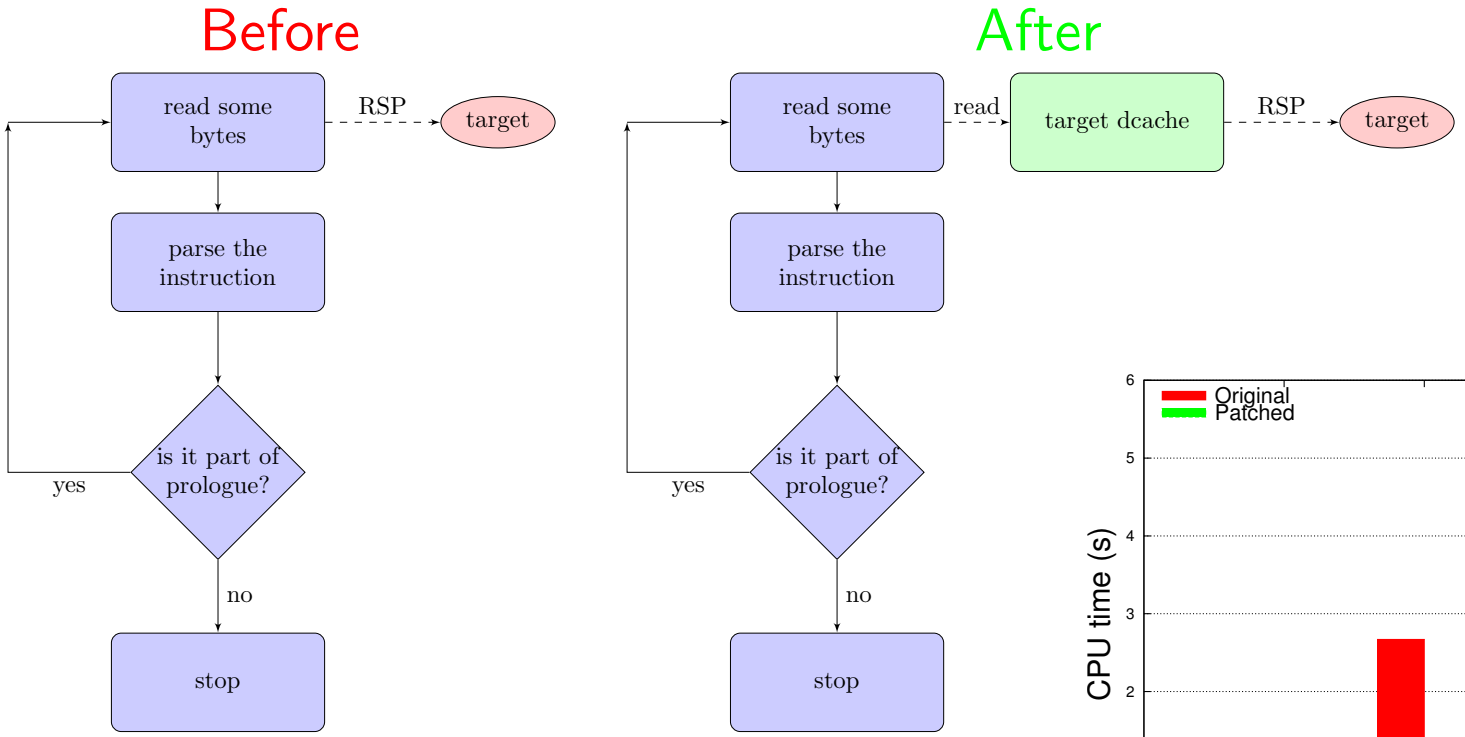
Extend from TestCaseWithBasicMeasurements

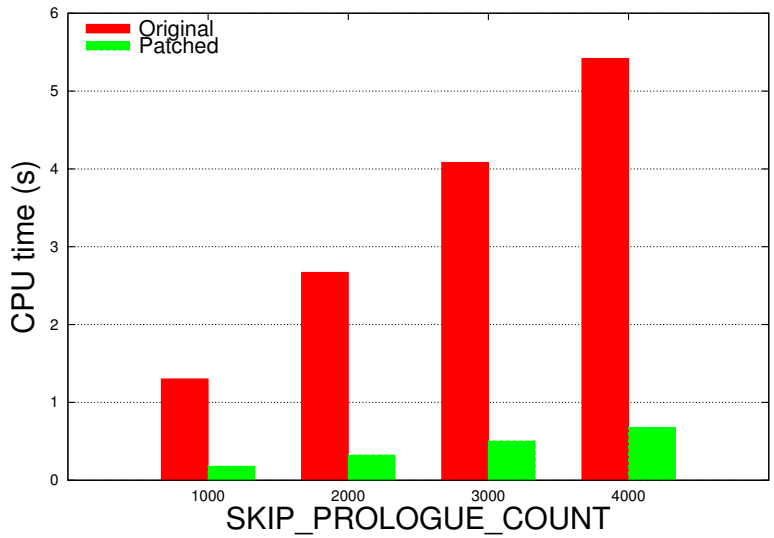
Test

Measure the test



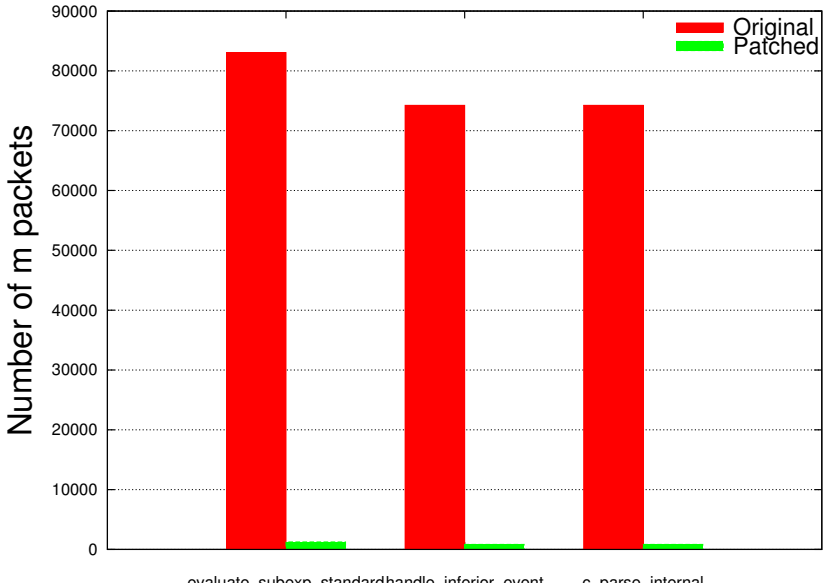
Cache code access for skip-prologue

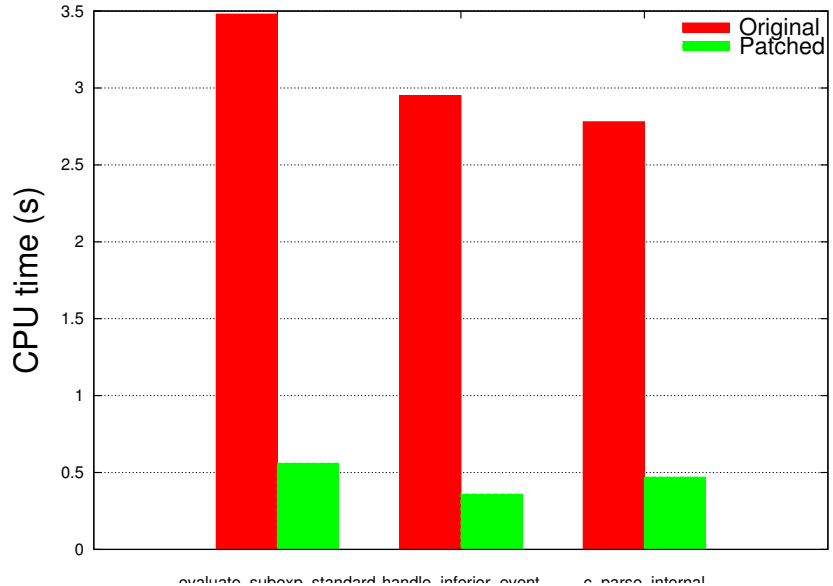




note that only x86 and x86_64 make use that.

Cache code access for disassemble



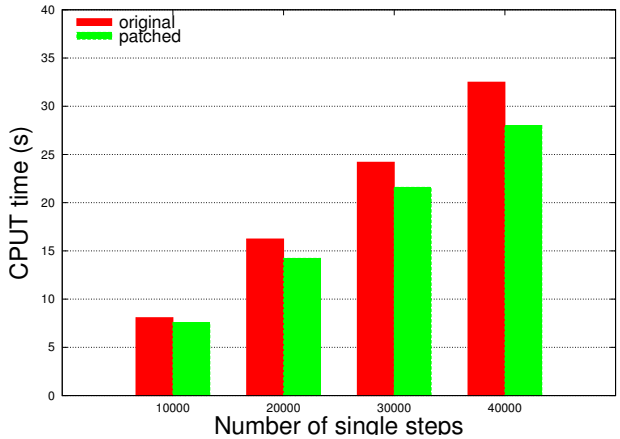


My workflow of improving perf

- Examine the RSP log and locate some unnecessary packets

```
(gdb) si
Sending packet: $qTStatus#49...
Sending packet: $Z0,80483c7,1#b4...Packet received: OK
Sending packet: $Z0,4ce5b6b0,1#6e...Packet received: OK
Sending packet: $QPassSignals:...
Sending packet: $vCont;s:p1b15.1b15;c#20...
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $mbfffef40,40#c0...
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $z0,80483c7,1#d4...Packet received: OK
Sending packet: $z0,4ce5b6b0,1#8e...Packet received: OK
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
Sending packet: $qXfer:traceframe-info:read::0,fff#0b...Packet received: E01
```

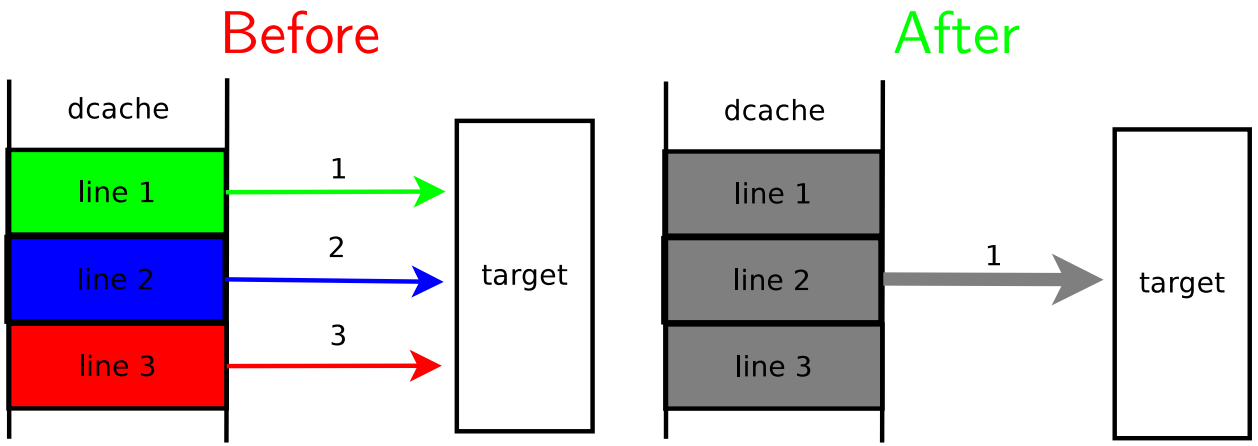
- Investigate and hack the code,
- Show the perf improved by the patch,

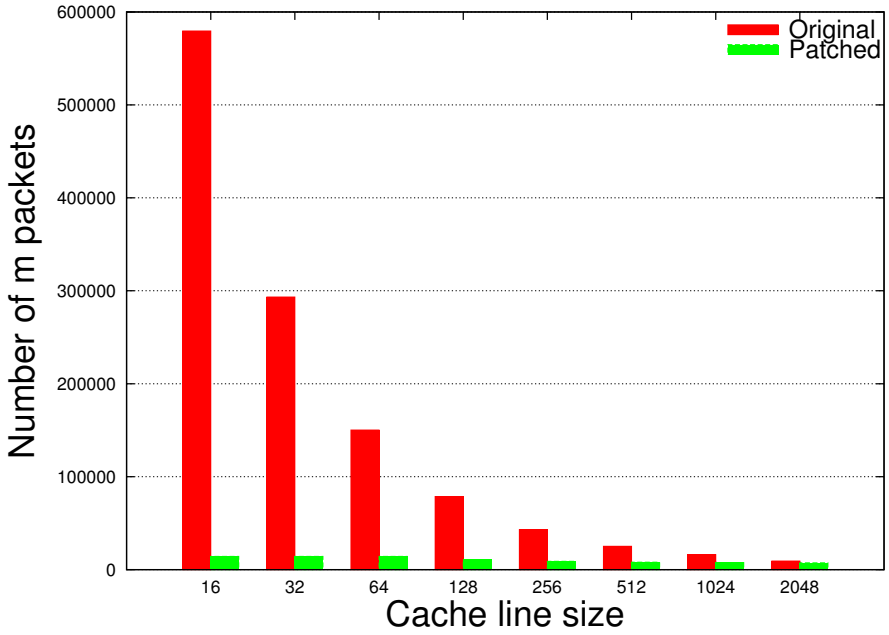


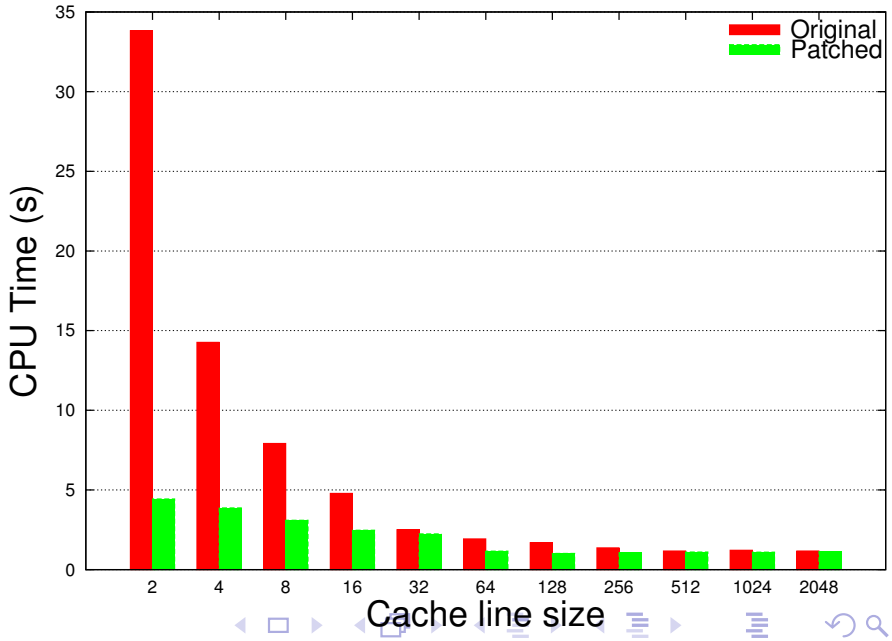
Typical profling doesn't help

```
-   2.91%  gdbserver  libc-2.14.90.so      [.] __strcmp_sse4_2
     __strcmp_sse4_2
-   1.92%  gdbserver  libc-2.14.90.so      [.] vfprintf
   - vfprintf
      - 74.54% _IO_vsprintf
           52.60% handle_tracepoint_query
        + 47.40% 0x1
```

Read multiple cache lines in dcache_xfer_memory

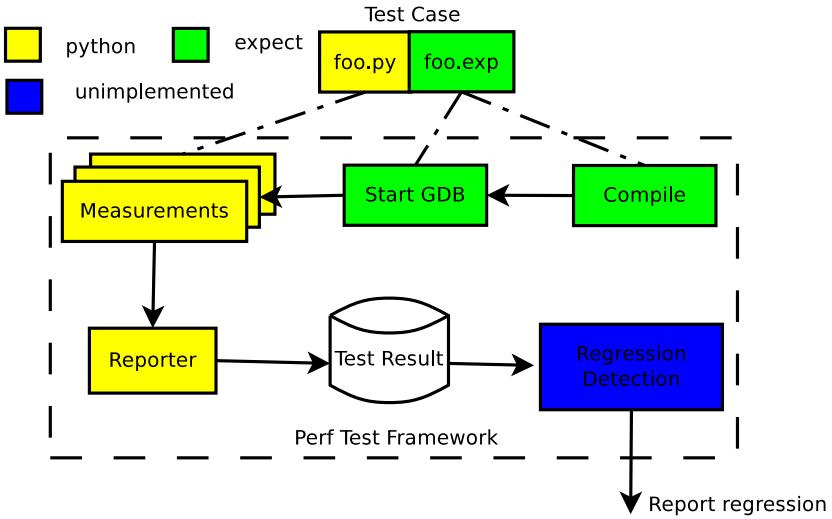


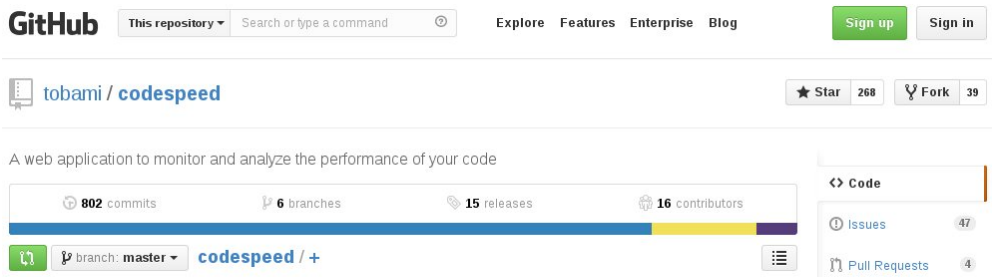




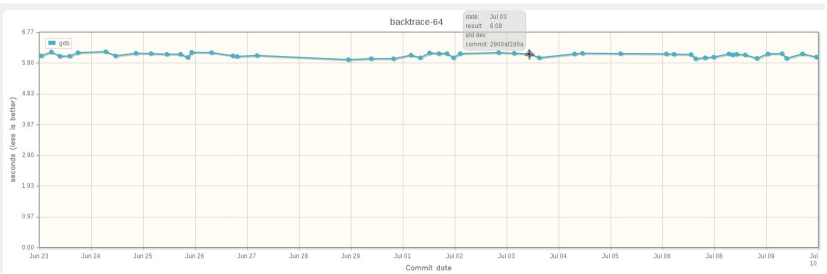
Track and compare GDB performance

Still have no way to track and compare perf data





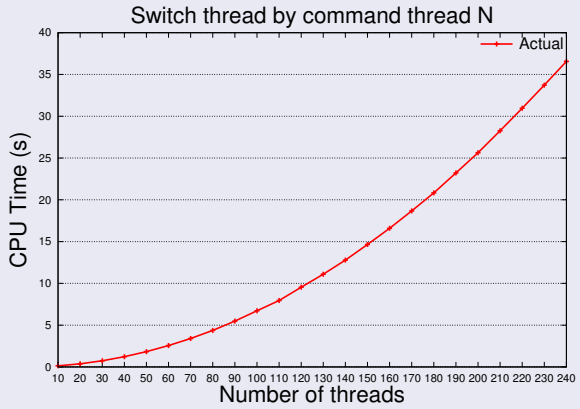
- A web application to monitor and analyze the performance,
- Written by python and easy to set up,
- It doesn't meet our needs perfectly, and it is not actively maintained,



Observations and future work

observations

- Command `thread` doesn't scale,

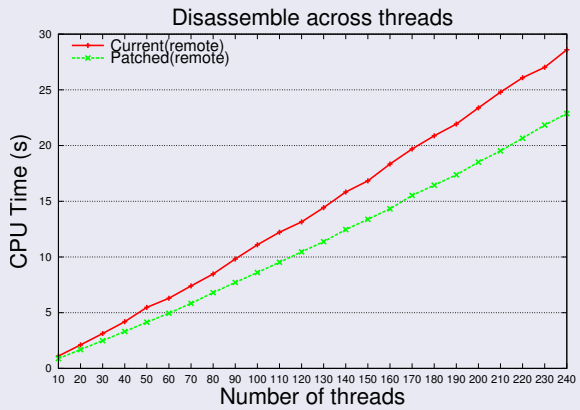



```
1  static int
2  do_captured_thread_select (struct ui_out *uiout, void *tidstr)
3  {
4    ...
5    /* Since the current thread may have changed, see if there is any
6       exited thread we can now delete.  */
7    prune_threads ();
8
9    return GDB_RC_OK;
10 }
```

- dcache is conservative for multi-threaded code,

```
1  enum target_xfer_status
2  dcache_read_memory_partial (struct target_ops *ops, DCACHE *dcache,
3                              CORE_ADDR memaddr, gdb_byte *myaddr,
4                              ULONGEST len, ULONGEST *xfered_len)
5  {
6    /* If this is a different inferior from what we've recorded,
7       flush the cache.  */
8
9    if (! ptid_equal (inferior_ptid, dcache->ptid))
10     {
11       dcache_invalidate (dcache);
12       dcache->ptid = inferior_ptid;
13     }
```

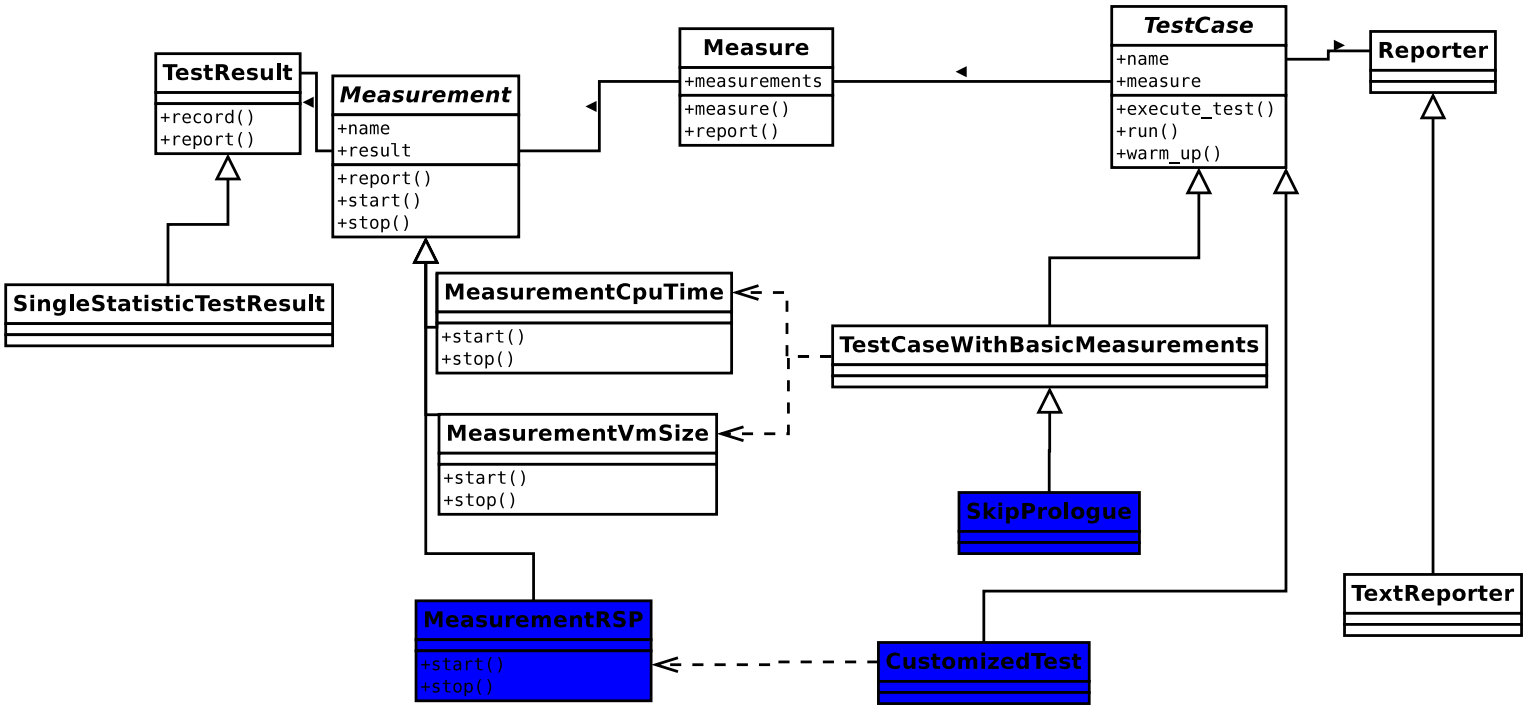
Remove?



future work

Compare perf data and alarm if regression is found. More perf test cases

Framework


TestResult
+record()
+report()
Measurement
+name
+result
+report()
+start()
+stop()
Measure
+measurements
+measure()
+report()
TestCase
+name
+measure
+execute_test()
+run()
+warm_up()
Reporter
SingleStatisticTestResult
MeasurementCpuTime
+start()
+stop()
TestCaseWithBasicMeasurements
MeasurementVmSize
+start()
+stop()
SkipPrologue
TextReporter
MeasurementRSP
+start()
+stop()
CustomizedTest