Copenhagen, 26 May 2003

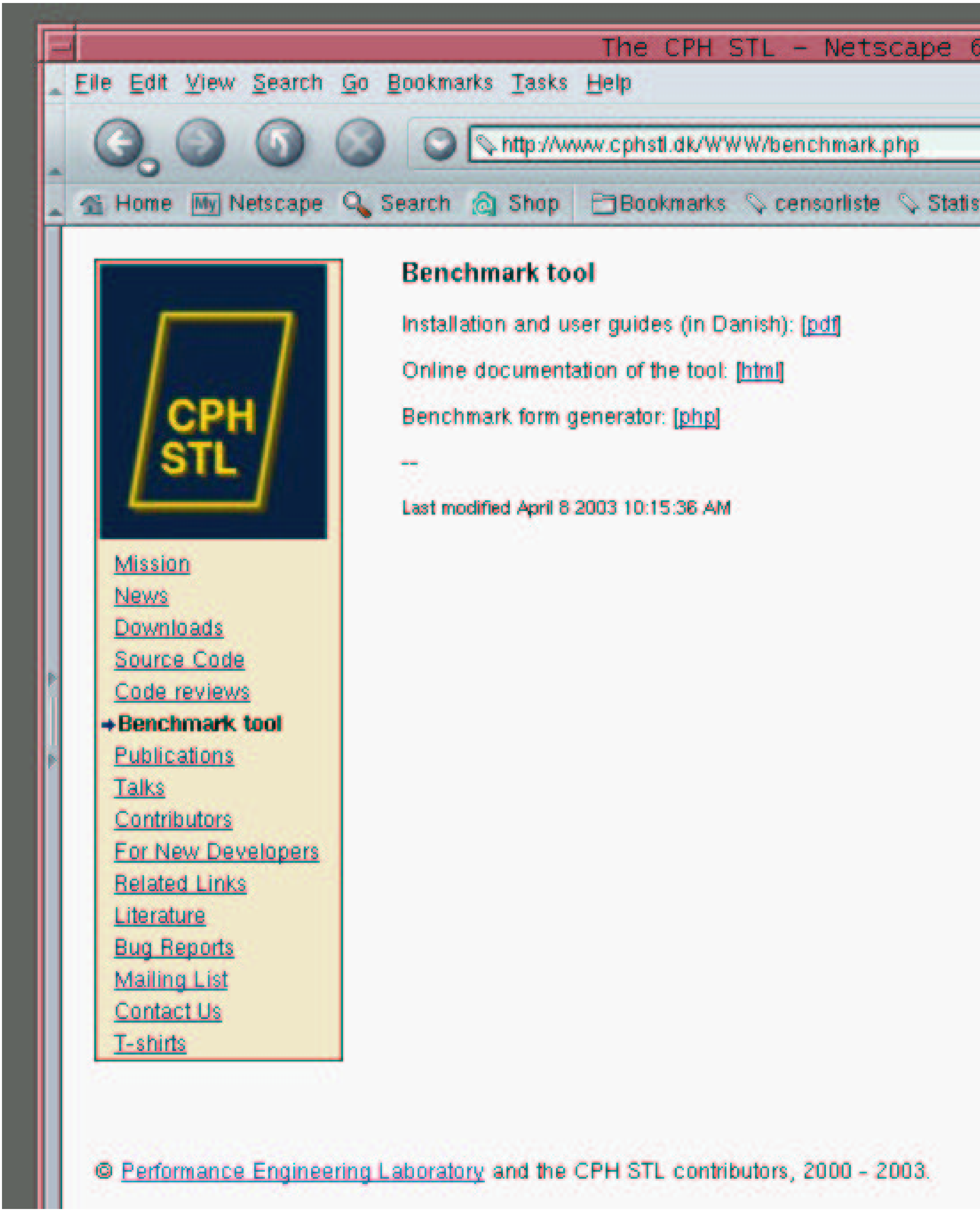
Title:

The first four months of our benchmark tool

Speaker:

Jyrki Katajainen

Datalogisk Institut
Københavns Universitet



Benchmark tool

Installation and user guides (in Danish): [pdf]

Online documentation of the tool: [html]

Benchmark form generator: [php]

--

Last modified April 8 2003 10:15:36 AM

© Performance Engineering Laboratory and the CPH STL contributors, 2000 - 2003.

Development history

1. I presented the initial idea at the first STL workshop in October 2000, see `http://www.diku.dk/undervisning/2000e/e00.505/#workshop`

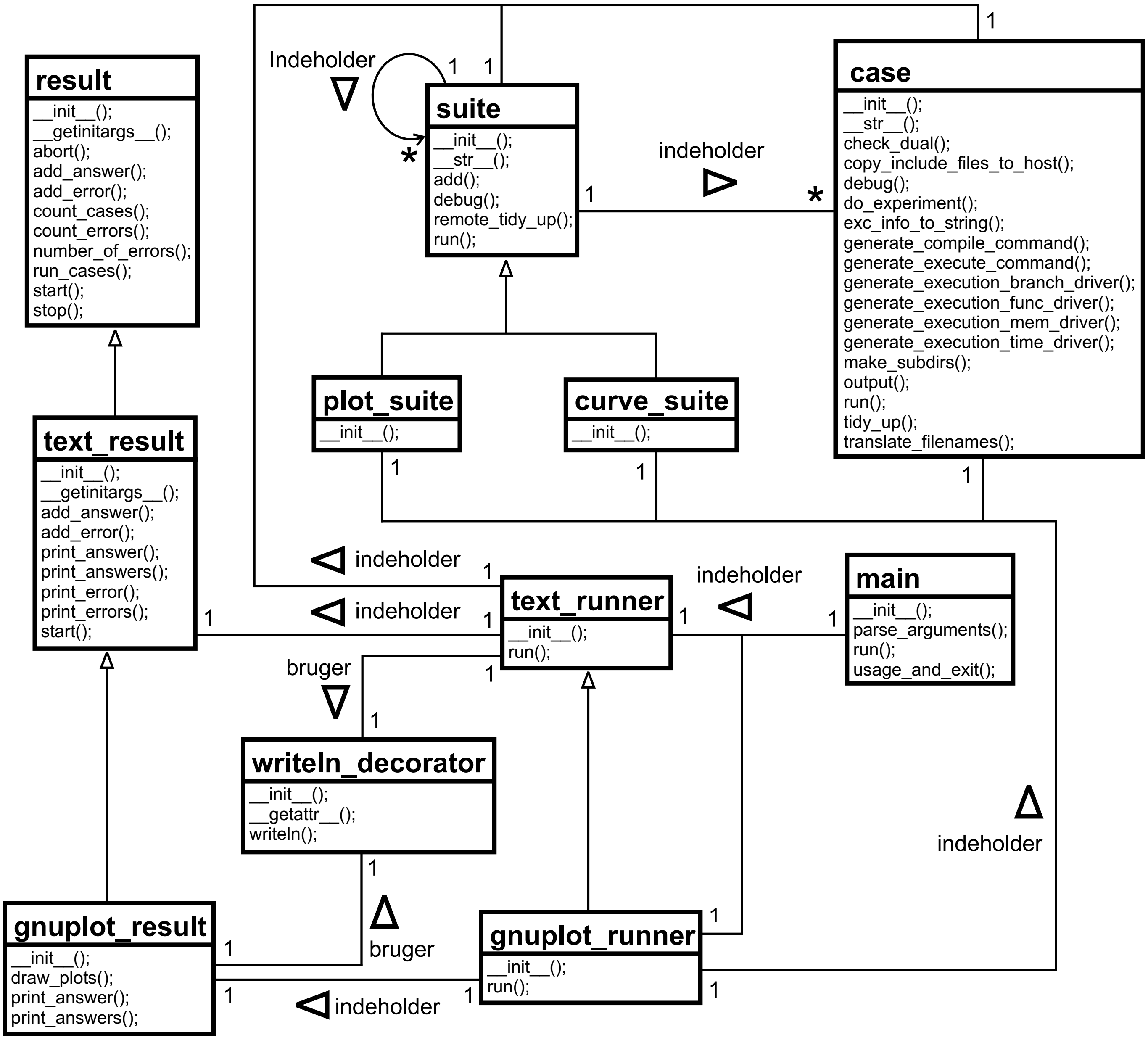
2. I implemented the tool in Python in May 2002, November 2002, December 2002, and January 2003

3. I released revision 1.7 in the Internet on 28 January 2003

4. Christian Søttrup and Jakob Pedersen added the support for benchmarking on a remote computer, web documentation, web form, and new driver generators: `generate_function_call_count_driver`, `generate_branch_misprediction_ratio_driver`, `generate_cache_miss_driver`, and `generate_page_fault_driver`

5. Frederik Rønn provided a new driver generator: `generate_wall_clock_time_driver`

Structure of the tool



result
- __init__();
- __getinitargs__();
- abort();
- add_answer();
- add_error();
- count_cases();
- count_errors();
- number_of_errors();
- run_cases();
- start();
- stop();

suite
- __init__();
- __str__();
- add();
- debug();
- remote_tidy_up();
- run();

case
- __init__();
- __str__();
- check_dual();
- copy_include_files_to_host();
- debug();
- do_experiment();
- exc_info_to_string();
- generate_compile_command();
- generate_execute_command();
- generate_execution_branch_driver();
- generate_execution_func_driver();
- generate_execution_mem_driver();
- generate_execution_time_driver();
- make_subdirs();
- output();
- run();
- tidy_up();
- translate_filenames();

plot_suite
- __init__();

curve_suite
- __init__();

text_result
- __init__();
- __getinitargs__();
- add_answer();
- add_error();
- print_answer();
- print_answers();
- print_error();
- print_errors();
- start();

text_runner
- __init__();
- run();

main
- __init__();
- parse_arguments();
- run();
- usage_and_exit();

writeln_decorator
- __init__();
- __getattr__();
- writeln();

gnuplot_result
- __init__();
- draw_plots();
- print_answer();
- print_answers();

gnuplot_runner
- __init__();
- run();

Benchmarking an empty loop

Function spin(n):

```
void spin(int n) {
  int i = n;
  while (--i >= 0);
}
```

Computer:

Intel Pentium 4 workstation with 1.5 GHz i686 processor (1st-level cache: 8 KB, 8-way associative 2nd-level cache: 256 KB, internal memory: 256 MB)

Operating system:

Red Hat Linux 7.1

Compiler:

g++ C++ compiler 3.0.4 (with option -O3)

C++ experiment class

```
/*
Benchmark for spinning

Author: Jyrki Katajainen
Email: jyrki@diku.dk
$Revision: 1.2 $
$Date: 2005/02/15 12:59:02 $

*/

class spin {
public:
  spin(int n): n(n) {
  }
  void primal() {
    int i = n;
    while (--i >= 0);
  }
private:
  int n;
};
```

Python form

```
"""
Benchmarking the execution time of an empty loop

shell> echo $PYTHONPATH
/home/disk04/jyrki/CPHSTL/Tool/Benchmark/
"""

__author__ = "Jyrki Katajainen"
__email__ = "jyrki@diku.dk"
__version__ = "$Revision: 1.2 $"[11:-2]
__date__ = "$Date: 2005/02/15 12:59:02 $"[7:-2]

import benchmark
import os

class spin_case(benchmark.case):
  def __init__(self, n):
    benchmark.case.__init__(self)
    self.n = n
    self.computer = 'pc-014.diku.dk'
    self.compiler = '/scratch/g++3.0.4/bin/g++'
    self.compiler_options.extend(['-O3'])
    self.include_files.extend(['spin_benchmark.c++'])
    self.dual_exists = 0
    self.constructor_call = 'spin(' + str(n) + ')'
    self.time_unit = 'ns'
    self.driver_file = self.generate_cpu_time_driver()

  def output(self):
    return (self.n, float(self.driver_output) / self.n)

if __name__ == '__main__':
  benchmark.main(\
    task = spin_case(1000000),\
    runner = benchmark.text_runner\
  )
```

Dual exists

To measure the execution time of a function, the experiment has to be repeated several times. Therefore, the function should not have any side-effects.

The experiments are performed in pairs: primal and dual. The running time reported is that required by the primal minus that required by the dual.

Problem: Does a C++ class have a member function dual?

Since C++ does not support reflection, the solution was complicated. To help the tool, the boolean attribute `self.dual_exists` may be given. If this is not given, the tool tries to figure out the answer itself — but it can fail.

Results can vary a lot

```
pc-014> python single_spin.py
.
======================================================
(1000000, 0.99476600000000004)
------------------------------------------------------
Ran 1 benchmark case in 3.656 s
OK
pc-014> python single_spin.py
.
======================================================
(1000000, 2.6301199999999998)
------------------------------------------------------
Ran 1 benchmark case in 4.486 s
OK
```

Sometimes no results at all

```
pc-014> python single_spin.py
E
======================================================
ERROR: __main__.spin_case
Less than 90% of the outcomes differ at most 20% from
the median; try again
------------------------------------------------------
ERROR: __main__.spin_case
Traceback (most recent call last):
  File "benchmark.py", line 170, in run
    answer = self.output()
  File "single_spin.py", line 29, in output
    return (self.n, float(self.driver_output) / self.n)
ValueError: empty string for float()
------------------------------------------------------

======================================================
------------------------------------------------------
Ran 0 benchmark cases in 88.642 s
Errors: 2
```

Solution

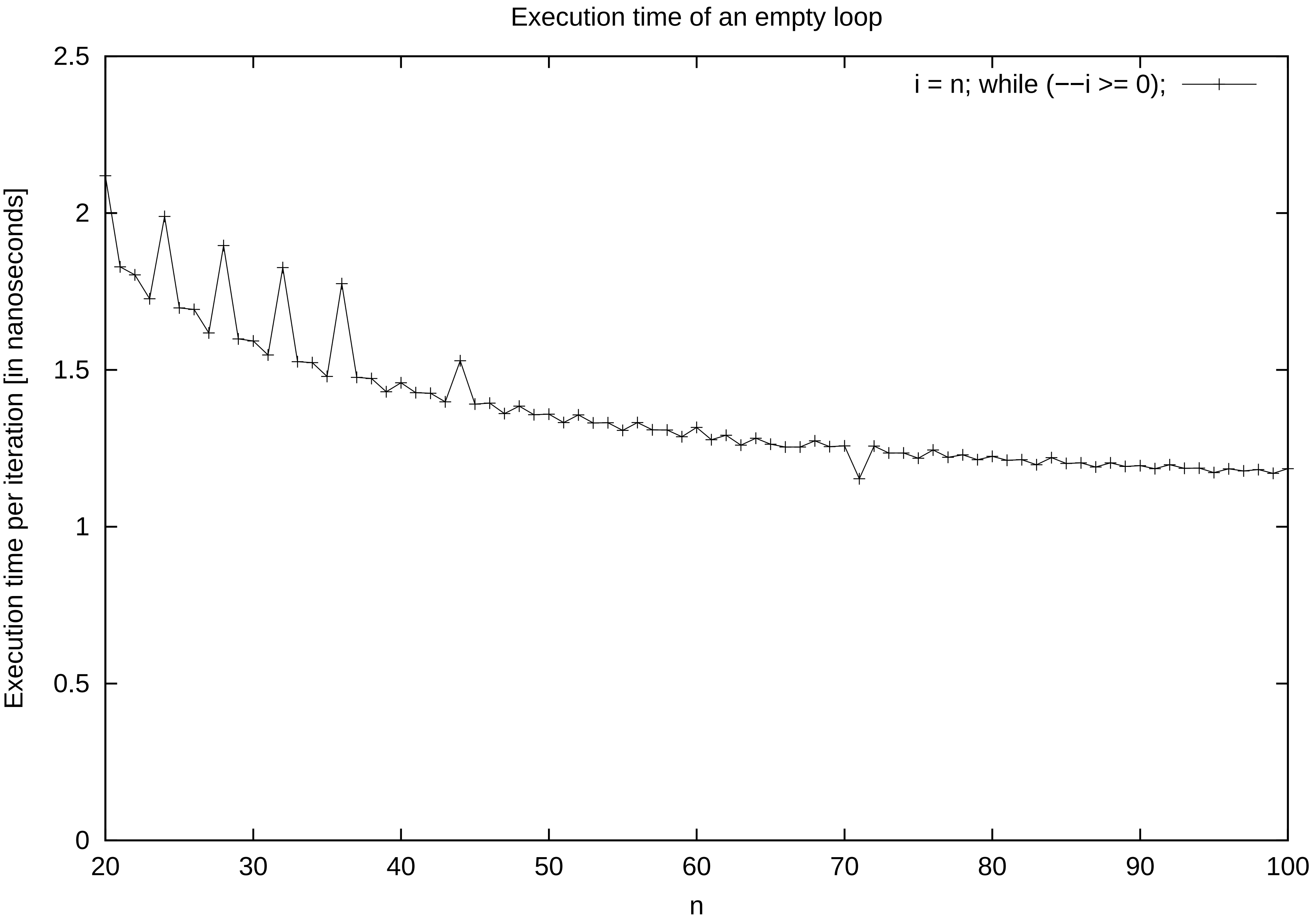
Kill unnecessary processes: `emacs`, `netscape`, etc.

An extract from another form

```
class spin_curve(benchmark.curve_suite):
  def __init__(self, low, high):
    benchmark.curve_suite.__init__(self)
    self.title = ’i = n; while (--i >= 0);’
    for k in range(low, high + 1):
      self.add(spin_case(k))

class spin_plot(benchmark.plot_suite):
  def __init__(self, low, high):
    benchmark.plot_suite.__init__(self)
    self.title = ’Execution time of an empty loop’
    self.xlabel = ’n’
    self.ylabel = \
      ’Execution time per iteration [in nanoseconds]’
    self.add(spin_curve(low, high))
    self.gnuplot_commands += """
set title ’%(title)s’
set xlabel ’%(xlabel)s’
set ylabel ’%(ylabel)s’
""" % self.__dict__

if __name__ == ’__main__’:
  benchmark.main(\
    task = spin_plot(20, 100),\
    runner = benchmark.gnuplot_runner\
  )
```


Execution time of an empty loop
i = n; while (−−i >= 0);
Execution time per iteration [in nanoseconds]
2.5
2
1.5
1
0.5
0
20
30
40
50
60
70
80
90
100
n

Sometimes it takes a long time

Problem: What should happen when the user presses control-C?

The hope is that the results generated so far are reported, but this goal turned out to be difficult to achieve. Sometimes one cannot stop an experiment at all.

Solution

Keep two windows open, and perform `make clean` in the other window. This will remove the programs needed by the benchmark tool and the experiment will stop.

Sometimes there are lot of errors

These are often C++ compiler errors. At least we have had big difficulties with the g++ compiler versions 2.96, 3.0.4, and 3.2.2. In spite of the ISO standard, the C++ compilers are still changing.

“There is something fundamentally wrong . . . ”

But often the underlying error is trivial: Some include file may be missing, sometimes there is a typo in the file name, etc.

Portability problems

```
shell> echo $PYTHONPATH
/home/disk04/jyrki/CPHSTL/Tool/Benchmark/

shell> cat ~/.tcshrc
...
if ( $?PYTHONPATH ) then
  setenv PYTHONPATH ~/CPHSTL/Tool/Benchmark/:$PYTHONPATH
else
  setenv PYTHONPATH ~/CPHSTL/Tool/Benchmark/
endif
...
alias python python2.2
...
```

More testing required

If I overload the method `tidy_up()`

```
 def tidy_up(self):
   "Used for debugging to see the files generated"
   pass
```

for a benchmark case, I get the error

```
ERROR: __main__.heapsort_case
Unable to perform remote tidy up
```

This seems curious since I run the benchmark locally on my own computer.

```
E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.
E.E.E.E.E.E.E.E.E.E.E.E.E.E.E.............................
======================================================
Wrote a gnuplot script in file:
/net/verdande/home/disk04/jyrki/CPHSTL/Report/
Multiway-heaps/Program/Benchmark/plot.68599188887.gp
Wrote a plot in file:
/net/verdande/home/disk04/jyrki/CPHSTL/Report/
Multiway-heaps/Program/Benchmark/plot.68599188887.ps
-------------------------------------------------------

=======================================================
ERROR: __main__.heapsort_case
Unable to perform remote tidy up
------------------------------------------------------------
...
------------------------------------------------------------
Ran 70 benchmark cases in 1234.077 s
Errors: 42
```