Laboratorul 06 - Analiza Performantei Programelor

The purpose of this lab is the familiarization with the field of application profiling & debugging, through the means of dedicated tools for spotting performance bottlenecks.

We will focus on several open source and commercial tools, as follows:

- valgrind / kcachegrind (on own systems & hp-sl.q cluster queue)
- Solaris Studio (on hp-sl.q cluster queue)
- perf (on your own systems)
- **You will record all your findings in a single text document and upload it on the Moodle.**

1. Valgrind / KCachegrind

Valgrind is a tool used for memory debugging, memory leak detection and profiling. It is also a generic framework for creating dynamic analysis tools, such as memory checkers [1].

Valgrind is in essence a virtual machine using just-in-time compilation techniques, including dynamic recompilation. It is important to keep in mind that nothing from the original program ever gets run directly on the host processor. Instead, it will translate the input program into a simpler form called Intermediate Representation (IR), which is processor neutral. After this transformation, a tool [2] is called to do whatever transformation of the IR it needs and the resulting IR is then translated back into machine code and ran on the host processor.

The tools available in Valgrind are:

- **memcheck**. Used to detect memory-management problems and it is aimed at C and C++ programs. All memory reads and writes are checked, and all calls to malloc/new/free/delete are intercepted. Therefore it can detect memory leaks, access to invalid memory, weird initialization values, overflows, etc. Memcheck runs programs about 10-30x slower than normal;
- **kcachegrind**. Used to profile CPU cache. It performs detailed simulation of the I1, D1 and L2 caches in order to pinpoint the sources of cache misses. It identifies the number of cache misses, memory references and instructions executed for each line of source code. Cache grind runs programs about 20-100x slower than normal;
- **callgrind**. It is an extension to cachegrind and provides all the information that the latter offers, plus extra information regarding call graphs. In order to view the results, a visualization tool called KCachegrind [3] can be used;
- **massif**. It is a heap profiler and it performs detailed heap profiling by taking regular snapshots of a program's heap and produces a graph showing heap usage over time, including information about which parts of the program are responsible for the most memory allocations. Massif runs programs about 20x slower than normal;
- **hellgrind** and drd. These tools are thread debuggers which find data races in multithreaded programs. They look for memory locations which are accessed by more than one (POSIX) pthread, but for which no consistently used (pthread_mutex_) lock can be found;
- other 3rd party tools can be found here [4].

TASK 1: Install valgrind on your computers (or run vallgrind/kcachegrind on the hp-sl.q queue) and run the memcheck tool, for the primes.c app with tests specified in the skeleton. Record your comments / timings and observations in the text document of lab 6.

Intrati pe frontend-ul `fep.grid.pub.ro` folosind contul de pe cs.curs.pub.ro, utilizand comanda `ssh -X username@fep.grid.pub.ro`. Executati comanda `qlogin -q hp-sl.q` pentru a accesa una din serverele de pe coada hp-sl.q.

Pentru a rula si observa functionalitatile valgrind/kcachegrind urmariti urmatoarea secventa si urmati apoi indicatiile din codul primes.c.

```
[@fep7-1 ]$ qlogin -q hp-sl.q
[@hpsl-wn01 ~]$ wget -O primes.c.zip http://ocw.cs.pub.ro/courses/_media/asc/lab6/primes.c.zip
[@hpsl-wn01 ~]$ module load compilers/gnu-6.2.0
[@hpsl-wn01 ~]$ gcc -o prime-ex primes.c
[@hpsl-wn01 ~]$ valgrind --tool=callgrind -v --dump-every-bb=10000000 ./prime-ex
[@hpsl-wn01 ~]$ kcachegrind
```

Recomandăm sa va delogati mereu de pe serverele din cluster dupa terminarea sesiunii, utilizand comanda `logout`

Alternativ, daca ati uitat sesiuni deschise, puteti verifica acest lucru de pe fep.grid.pub.ro, utilizand comanda `qstat`. In cazul in care identificati astfel de sesiuni "agatate", le puteti sterge (si va rugam sa faceti asta), utilizand comanda `qdel -f ID` unde ID-ul il identificati din comanda anterioara `qstat`.

Daca nu veti face aceasta delogare, veti putea ajunge in situatia in care sa nu va mai puteti loga pe nodurile din cluster.

2. Solaris Studio

Pentru a utiliza utilitarul pentru profiling Oracle Solaris Studio, trebuie sa efectuati urmatorii pasi:

```
[@fep7-1 ]$ qlogin -q hp-sl.q
[@hpsl-wn01 ~]$ module avail

------------------------------------------------------------ /usr/share/Modules/modulefiles -------------------------------------------------------------
dot         module-git  module-info modules     null        use.own

------------------------------------------------------------------- /etc/modulefiles --------------------------------------------------------------------
compilers/gnu-4.9.4                     libraries/cuda-10.2                     libraries/opencv-3.1.0-gcc-4.9.4
compilers/gnu-5.4.0                     libraries/cuda-7.5                      libraries/openmpi-2.0.1-gcc-4.9.4
compilers/gnu-6.2.0                     libraries/cuda-8.0                      libraries/openmpi-2.0.1-gcc-5.4.0
compilers/solarisstudio-12.5            libraries/cuda-9.0                      utilities/intel_parallel_studio_xe_2016
libraries/cuda                          libraries/cuda-9.1                      utilities/opencl
[@hpsl-wn01 ~]$ module load compilers/solarisstudio-12.5 compilers/gnu-6.2.0
[@hpsl-wn01 ~]$ module list
Currently Loaded Modulefiles:
  1) compilers/gnu-6.2.0            2) compilers/solarisstudio-12.5
[@hpsl-wn01 ~]$ gcc -g -o prime-ex primes.c
[@hpsl-wn01 ~]$ collect ./prime-ex
[@hpsl-wn01 ~]$ analyzer test.1.er
```

Atentie, in rulari succesive, va trebui sa incarcati arhive cu numere mai mari X "test.X.er".

Un scurt tutorial este disponibil aici (aveti grija sa incarcati in prealabil modulul Solaris Studio):
http://cluster.grid.pub.ro/index.php/cluster-howto/30-profiling/76-profiling-with-sun-studio-analyzer
[http://cluster.grid.pub.ro/index.php/cluster-howto/30-profiling/76-profiling-with-sun-studio-analyzer]

O descriere si resurse suplimentare puteti obtine de asemenea pe site-ul oficial:
https://docs.oracle.com/cd/E37069_01/html/E37073/gkodh.html [https://docs.oracle.com/cd/E37069_01/html/E37073/gkodh.html]

Daca doriti sa va instalati Oracle Solaris/Developer Studio, o puteti face de aici:
https://www.oracle.com/tools/developerstudio/downloads/developer-studio-jsp.html
[https://www.oracle.com/tools/developerstudio/downloads/developer-studio-jsp.html]

Pentru rularea paralela trebuie urmarite urmatoarele comenzi:

```
[@hpsl-wn01 ~]$ gcc -g -fopenmp -o prime-omp primes.c
[@hpsl-wn01 ~]$ collect ./prime-omp
[@hpsl-wn01 ~]$ analyzer test.1.er
```

Aceeasi observatie ca mai sus, probabil ca datele experimentale sunt intr-o alta arhiva cu numar > 1, in formatul test.X.er, cu X > 1.

TASK 2: Run Solaris Studio Profiler on the hp-sl.q queue servers from our cluster by following the steps above on the primes.c app with tests specified in the skeleton. Explore the various screens and reports of the tool, with the "Functions, Timeline, Call Tree, Source, Callers-Callees" tabs. Record your comments and observations in the text document of lab 6.

Recomandăm sa va delogati mereu de pe serverele din cluster dupa terminarea sesiunii, utilizand comanda `logout`

Alternativ, daca ati uitat sesiuni deschise, puteti verifica acest lucru de pe fep.grid.pub.ro, utilizand comanda `qstat`. In cazul in care identificati astfel de sesiuni "agatate", le puteti sterge (si va rugam sa faceti asta), utilizand comanda `qdel -f ID` unde ID-ul il identificati din comanda anterioara `qstat`.

Daca nu veti face aceasta delogare, veti putea ajunge in situatia in care sa nu va mai puteti loga pe nodurile din cluster.

3. Analysis the Tachyon raytracing engine

In this section, we will focus on analyzing a software application. We will analyze both a serial and a parallel implementation. The application is called "tachyon" and you can find the source code attached to this lab.

On your own system, before compilation, you must install the X11 dev tools and create a set of symlinks. For Ubuntu 64 bit, we must do the following:

- install dependencies

```
sudo apt-get install libx11-dev
```

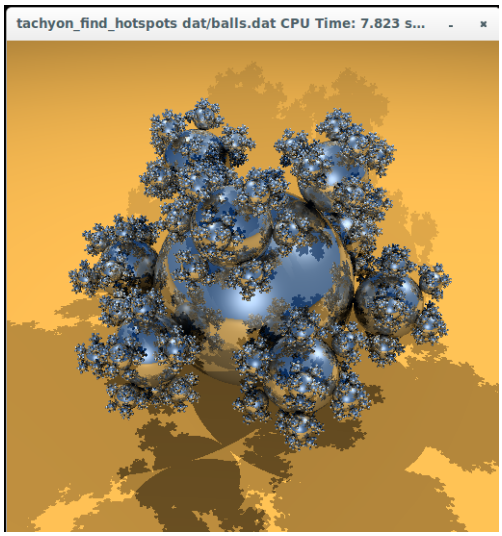
- create the symlinks:
    -
```
sudo mkdir /usr/lib64
```

- `sudo ln -s /usr/lib/x86_64-linux-gnu/libX11.so /usr/lib64/libX11.so`
- `sudo ln -s /usr/lib/x86_64-linux-gnu/libXext.so /usr/lib64/libXext.so`

To compile it, you must extract the archive to local disk and run make. You can test the compilation by running in the same directory:

```
./tachyon_find_hotspots dat/balls.dat
```

You should see a window like the one below:



Pentru a rula si observa functionalitatile acestei unelte urmariti urmatoarea secventa de pasi si indicatiile din codul primes.c.

```
[@fep7-1 ]$ qlogin -q hp-sl.q
[@hpsl-wn01 ~]$ wget -O tachyon_vtune_amp_xe.tgz http://ocw.cs.pub.ro/courses/_media/asc/lab6/tachyon_vtune_amp_xe.tgz
[@hpsl-wn01 ~]$ module load compilers/solarisstudio-12.5 compilers/gnu-6.2.0
[@hpsl-wn01 ~]$ gunzip tachyon_vtune_amp_xe.tgz
[@hpsl-wn01 ~]$ tar -xvf tachyon_vtune_amp_xe.tar
[@hpsl-wn01 ~]$ cd tachyon
[@hpsl-wn01 ~]$ make
```

TASK 3: Download the archive (on your system or the cluster) from the wiki (asc:lab6:tachyon_vtune_amp_xe.tgz), compile, run and test the application. Report the runtimes in the text document of lab 6.

Pentru rularea seriala si paralela + analiza cu Solaris Studio a codului de raytracing Tachyon trebuie urmarite urmatoarele comenzi:

```
[@hpsl-wn01 ~]$ collect ./tachyon_find_hotspots
[@hpsl-wn01 ~]$ analyzer test.X.er
[@hpsl-wn01 ~]$ collect ./tachyon_analyze_locks
[@hpsl-wn01 ~]$ analyzer test.Y.er
```

unde X si Y sunt ID-urile experimentelor pentru versiunea seriala, respectiv paralele.

TASK 4: Analyze using Solaris Studio the serial and parallel versions of the Tachyon app. Report the findings in the text document of lab 6.

Using Valgrind on the Tachyon Code

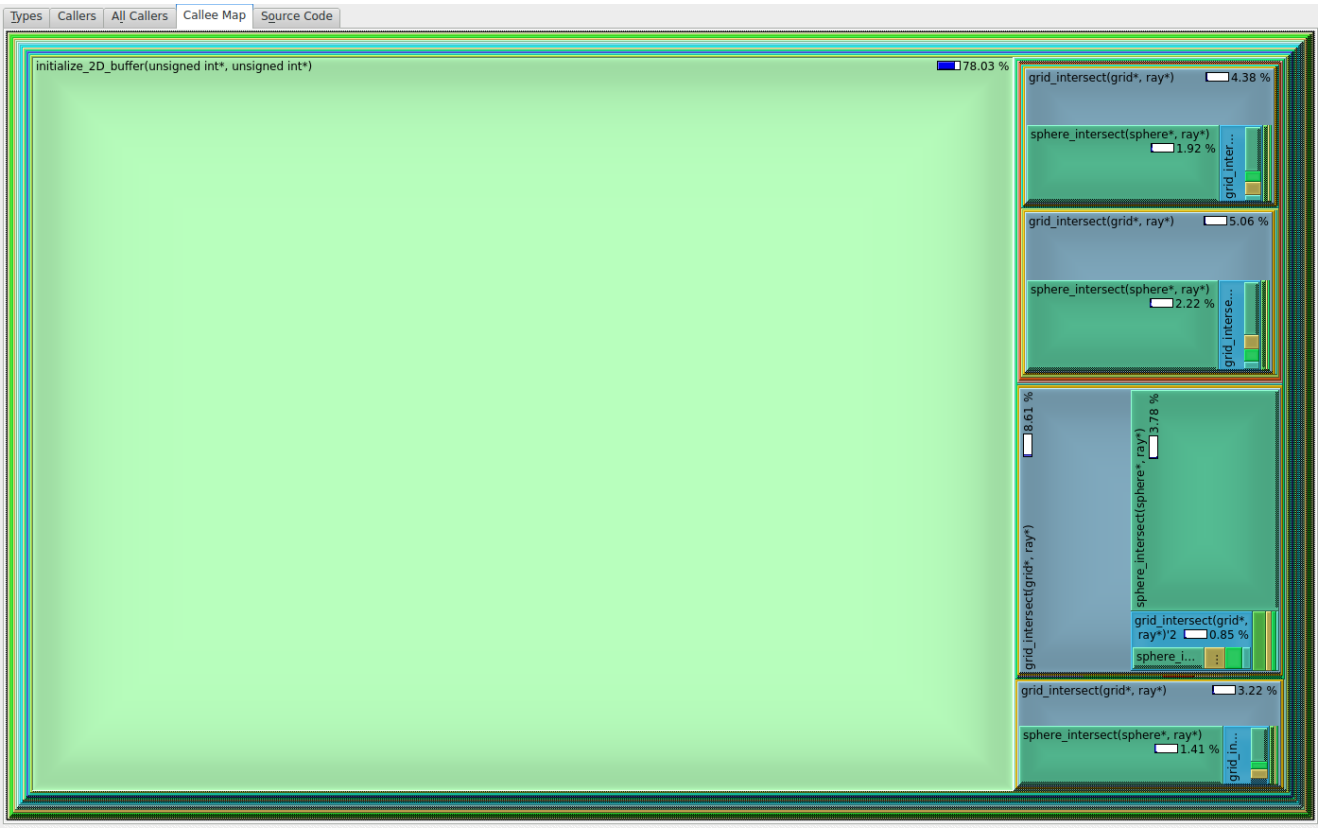
1. Make sure you have Valgrind and KCachegrind installed on the system (or login on the hp-sl.q queue) and the application in the initial state, without any modifications on your system

```
sudo apt-get update
sudo apt-get install valgrind kcachegrind
```

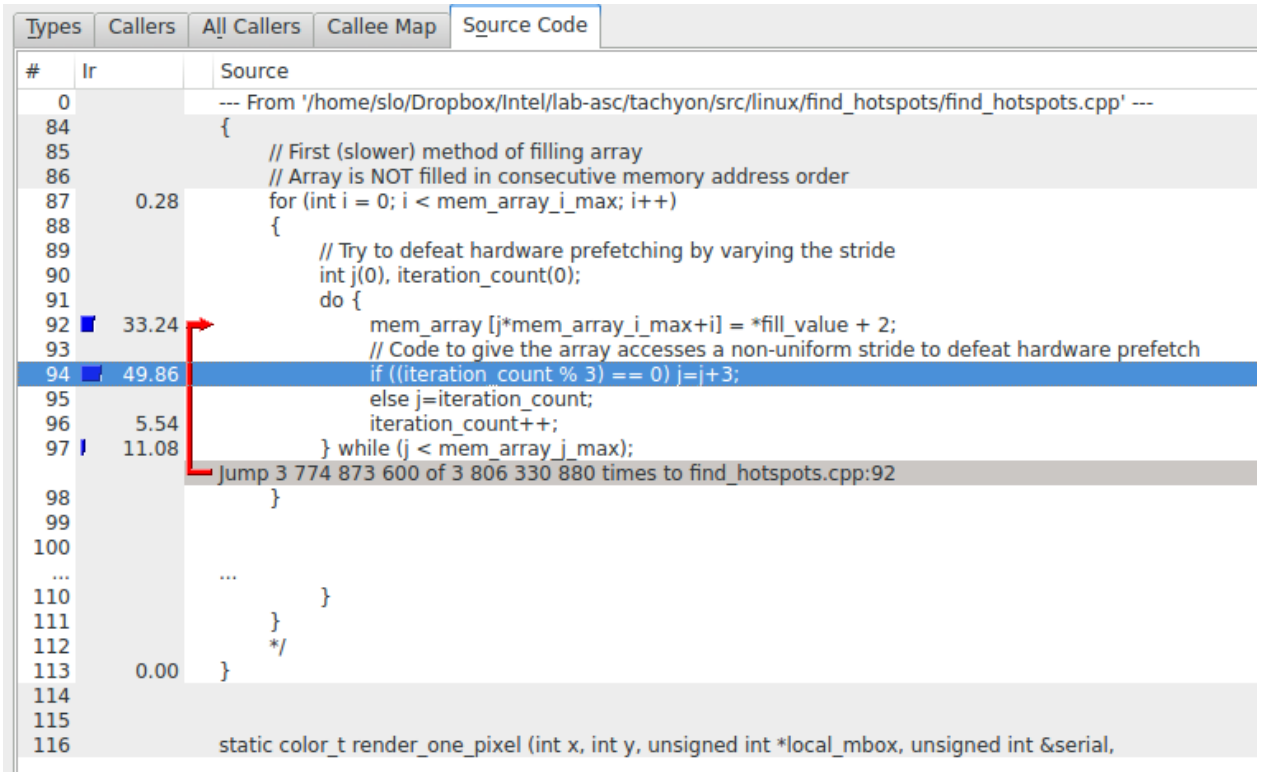
2. We will use the tool *callgrind* to get information from the running application. Run the following command line:

```
valgrind --tool=callgrind --collect-jumps=yes --dump-instr=yes --collect-systime=yes -- ./tachyon_find_hotspots dat/balls.dat
```

3. Open the profile in KCachegrind and click on the Calee Map tab. Also, make sure that the buttons *% Relative*, *Cycle detection* and *Relative to parent* are selected. You should see something like this:



From this image, we can see that valgrind measured that about 78% of the total time was spent in the initialize_2D_buffer function. Double click the square containing the function name, then select the "Source code" tab and you will see the problematic code.



TASK 5: Analyze using vallgrind/kcachegrind the serial and parallel versions of the Tachyon app. Report the findings in the text document of lab 6.

Recomandăm sa va delogati mereu de pe serverele din cluster dupa terminarea sesiunii, utilizand comanda `logout`

Alternativ, daca ati uitat sesiuni deschise, puteti verifica acest lucru de pe fep.grid.pub.ro, utilizand comanda `qstat`. In cazul in care identificati astfel de sesiuni "agatate", le puteti sterge (si va rugam sa faceti asta), utilizand comanda `qdel -f ID` unde ID-ul il identificati din comanda anterioara `qstat`.

Daca nu veti face aceasta delogare, veti putea ajunge in situatia in care sa nu va mai puteti loga pe nodurile din cluster.

4. Perf

Perf is a performance analysis tool, available in the Linux kernel since version 2.6.31 [5]. The userspace control application is accessed from the command line and provides a number of subcommands. Unlike Valgrind, perf is capable of statistical profiling of both the entire system (kernel and userspace) and per process PID basis. It supports hardware performance counters, tracepoints, software performance counters (e.g. hrtimer), and dynamic probes (for example, kprobes or uprobes).

Perf is used with several subcommands:

- **stat**: measure total event count for a single program or for the whole system for a specified time period;
- **top**: top-like dynamic view of hottest functions;
- **record**: measure and save sampling data for single program;
- **report**: analyze file generated by perf record;
- **annotate**: annotate sources or assembly;
- **sched**: tracing/measuring of scheduler actions and latencies;
- **list**: list available events.

1. Make sure you have perf installed on the system and the application in the initial state, without any modifications. **You can only run perf as root. You can only do this on your system.** 2. Run the following command line:

```
perf record -a -g -- ./tachyon_find_hotspots
```

For other perf parameters, you can read this link [http://www.brendangregg.com/perf.html] 3. Run the following command line to view the collected results:

```
perf report
```

You should see a screen like the following:

```
Samples: 223K of event 'cycles', Event count (approx.): 79199296496
  Children      Self  Command          Shared Object                          Symbol
+   28.64%     0.00%  tachyon_find_ho  [unknown]                              [.] 0000000000000000
+   28.05%     0.00%  tachyon_find_ho  [unknown]                              [.] 0x0000000100000005
+   27.64%    27.57%  tachyon_find_ho  tachyon_find_hotspots                  [.] _Z20initialize_2D_bufferPjS_
+   15.58%     1.04%  dropbox          dropbox                                [.] PyEval_EvalFrameEx
+   15.26%     0.00%  dropbox          [unknown]                              [.] 0x0000000000ac9dc0
+   14.52%     0.00%  dropbox          librsyncffi.compiled._librsyncffi.so   [.] 0xffff8048acaa238b
+   14.52%     0.00%  dropbox          librsync.so.1                          [.] rs_sig_file
+   14.52%     0.00%  dropbox          librsync.so.1                          [.] rs_whole_run
+   14.52%     0.00%  dropbox          librsync.so.1                          [.] rs_job_drive
+   14.38%     0.00%  dropbox          librsync.so.1                          [.] rs_job_iter
+   14.36%     0.00%  dropbox          librsync.so.1                          [.] 0xffff8048accb4dc3
+   14.27%     0.00%  dropbox          librsync.so.1                          [.] 0xffff8048accb5edd
+   13.70%    13.67%  tachyon_find_ho  tachyon_find_hotspots                  [.] _ZL14grid_intersectP4gridP3ray
+    8.80%     0.00%  dropbox          librsync.so.1                          [.] 0xffff8048accb5d40
```

From this image you can see that perf will display the symbol for the function that takes the most amount of CPU time in red. In our case it's the *_Z20initialize_2D_bufferPjS_*, which translates in the C source code into the same function as with VTune and Valgrind.

Hint: To find out the demangled name, use the c++filt command:

```
c++filt _Z20initialize_2D_bufferPjS_
```

TASK 6: (Bonus 2p) You can only run perf as root. You can only do this on your system. Make sure that *perf* is installed on your computers. Use it to test / check the tachyon code (serial and parallel) with the same parameters as before. Record the information in the text document for lab 6.

Recomandăm sa va delogati mereu de pe serverele din cluster dupa terminarea sesiunii, utilizand comanda `logout`

Alternativ, daca ati uitat sesiuni deschise, puteti verifica acest lucru de pe fep.grid.pub.ro, utilizand comanda `qstat`. In cazul in care identificati astfel de sesiuni "agatate", le puteti sterge (si va rugam sa faceti asta), utilizand comanda `qdel -f ID` unde ID-ul il identificati din comanda anterioara `qstat`.

Daca nu veti face aceasta delogare, veti putea ajunge in situatia in care sa nu va mai puteti loga pe nodurile din cluster.

Reference

- http://valgrind.org/ [http://valgrind.org/]

- http://valgrind.org/info/tools.html [http://valgrind.org/info/tools.html]
- http://kcachegrind.sourceforge.net/html/Usage.html [http://kcachegrind.sourceforge.net/html/Usage.html]
- http://valgrind.org/downloads/variants.html [http://valgrind.org/downloads/variants.html]
- https://perf.wiki.kernel.org/index.php/Main_Page [https://perf.wiki.kernel.org/index.php/Main_Page]
- https://software.intel.com/en-us/intel-parallel-studio-xe [https://software.intel.com/en-us/intel-parallel-studio-xe]
- http://www.brendangregg.com/perf.html [http://www.brendangregg.com/perf.html]
- https://www.oracle.com/tools/developerstudio/downloads/developer-studio-jsp.html [https://www.oracle.com/tools/developerstudio/downloads/developer-studio-jsp.html]

Resources

- Responsabilii acestui laborator: Emil Slusanschi [mailto:emil.slusanschi@cs.pub.ro], Alexandru Patrascu si Octavian Moraru.
- PDF laborator
- Aplicatie laborator (tachyon)
- Aplicatie de test laborator

asc/laboratoare/06.txt · Last modified: 2021/02/16 20:42 (external edit)