PDC Summer School 2010
Brief notes on serial performance measurement

August 21, 2010

For high performance scientific computing software serial (single node) performance is vitally important. This begs the questions "is my code efficient?" To answer that question performance measurement tools are needed. These, often referred to as profilers, fall into three categories:

Sampling Sampling profilers are based on interrupting the execution of a program and collecting statistics. This will, for instance, reveal which functions/subroutines the program spent the most time in. Hence, a sampling profiler is typically the choice for investigating where it is worth optimizing a program. Two examples (on Linux):

- **Intel VTune**. Commercial and expensive (free non-commercial download for Linux). Fancy GUI, loads of very advanced features.
- **gprof** (the GNU profiler): Command-line interface, part of standard build environment, free. First (re)compile your code with the compiler flags `-pg -g` added. Run the code as usual (this produces a file called `gmon.out` in the working directory). Then, invoke `gprof` with `gprof ./my_prog`. To get line-by-line output, ie. on which lines in the code most times was spent, do

```
> gprof --flat-profile -l ./my_prog
```

Hardware counters Profilers based on hardware counters profile low-level hardware events. Registers used by the hardware manufacturers to debug and evaluate their designs are left and can be used for detailed profiling.

The tool for this on Linux is called PAPI and was developed by Phil J. Mucci. It is free, but quite hard to install (it involves the Linux kernel and several dependencies).

Hardware counters give detailed information not available with sampling profilers. This includes the number of cache misses, branches mispredicted, floating-point operations retired, total number of load instructions etc. With this information (and some work) you can answer two basic questions: “is the program efficient?” and “why is it not efficient?”. Or rather, if the programmer knows enough about the hardware, PAPI will help him answer the second question.

See separate handout for the invocation of PAPI (`papiex`).

Emulators The last category of profilers attempt to give detailed information via hardware emulation instead of hardware counters. This has the benefit of being much simpler to install, but will only cover some aspects of the CPU. They typically focus on the memory hierarchy, i.e. cache statistics (which is often the most interesting metric to programmers). Note, however, that the results are only as accurate as the emulator. If the emulator has incorrect or incomplete parameters for the present architecture it will give a warning (and those warnings are important). Two notable tools are:

- **Accumem** Commercial, GUI, user-friendly. Developed by Prof. Erik Hagersten.
- **Valgrind** Every C-programmer should (!) know and use Valgrind to check for memory leaks and indexing errors. It has several other capabilities, including a cache profiler called `cachegrind`. Invoke with

  ```
  > valgrind --tool=cachegrind ./my_prog
  ```