Profiling MPI codes with Allinea's MAP

Sergey Mashchenko
McMaster University, SHARCNET
Overview

- Introduction
- Using MAP
- Demonstration
Introduction

- Profiling is an important part of code development
  - Almost as important as debugging; can be considered as “code performance debugging”
  - If writing a new code from scratch, profiling of new code blocks should be done continuously, alongside debugging: “performance bugs” made at the early code development stages will be hard or impossible to fix when the code is finished.
  - If converting a serial code to a parallel one (threads, MPI, CUDA, ...), profiling the serial code can be crucial in guiding the parallelization efforts.
Example: serial code profiling

- try_corner_move: 94.23% (0.00%)
- optimization_function: 94.56% (3.49%) 125722×
- lens_design: 5.13% (0.01%) 6344646×
- smart_surface_generation: 3.19% (0.01%) 6344646×
- ray_bender: 87.14% (8.19%) 579529120×
- zsurf_range: 0.61% (0.61%) 10252206×
- surface_ray_intersection: 29.59% (29.59%) 3940700854×
- refraction: 49.81% (37.78%) 3909705069×
- surface_derivative: 12.04% (12.04%) 3890848354×

- stop_absorption: 1.25% (1.23%) 555113482×
- chi_function: 3.58% (3.56%) 1168104×
- flip_rays: 2.43% (2.43%) 502842×
- 0.96% 6358145×
- 0.26% 4285775×
- 2.72% 18079048×
- 0.36% 5966431×
- 0.21% 27849817×
- 0.28% 21954955×
- 29.38% 3912851037×
- 49.53% 3887750114×
- 12.04% 3890848354×
• The serial code was profiled before its conversion to CUDA

• Profiling identified the ~85% of the code readily available for parallelization, and the next 10-14% which could be converted with more efforts

• The plot was generated using these commands:
  – gcc -pg ... -o code
  – ./code
  – gprof ./code | gprof2dot.py | dot -Tpng -o output.png
• SHARCNET web portal lists three officially supported MPI profilers:
  – **OPT**: the old product from Allinea, installed only on requin, not usable for realistically large MPI jobs (say, >8 cores for >30 minutes).
  – **IPM**: open source profiler
  – **MAP**: new profiler from Allinea, installed on orca; integrated with their parallel debugger DDT
MAP overview

- Integrated with DDT debugger – makes it easier to go back and forth between profiling and debugging
- Uses statistical sampling (~1000 samples per rank by default) to dramatically accelerate the profiling process
- As a result, the profiler's overhead is <5%
- Polished and convenient to use GUI
- No need to recompile the code (needs to be compiled with “-g” - same as for debugging)
- Installed on orca, license for 512 MPI ranks
```c
for (j = 1; j <= npoints; j++)
{
    /* global endpoints */
    if ((first + j - 1 == 1) || (first + j - 1 == tpoints))
        newval[j] = 0.0;
    else
        do_math(j);
    }
```

```c
do_math(j)
{
    oldval[] = values[];
    values[] = newval[];
    }
```

```c
allt = (end.tv_sec - start.tv_sec) * 1000000 + (end.tv_nsec - start.tv_nsec) / 1000;
double calculation_rate = ((double)tpoints / (double)allt) * iterations; /* in million points per second */
if (rank == 0) print("points / second: %1.1f (%1.1f per process)\n", calculation_rate, calculation_rate / ntask);
double efficiency = (double)(allt - communication_usec) / (double)allt;
reduce_print("compute / communicate efficiency: %2.1f %2.1f %2.2%\n", (int)(100 * efficiency + 0.5));
```
Using MAP

- Interactive use instructions (works for up to 24 cores):
  - `ssh orca`
  - `ssh orc-dev1` (or dev2, dev3, dev4)
  - `top` (check if the node is busy; no point profiling your code on a busy node)
  - `module load ddt`
  - compile your code with “-O2 -g” switches (or -O3)
  - `map ./code [arguments]`

- Interactive analysis, plus *.map is written which can be analyzed later.
Non-interactive use instructions (for up to 512 cores):
- ssh orca
- compile your code with "-O2 -g" switches (or -O3)
- module load ddt
- sqsub -q mpi -o out -r 1h --nompirun -n 2 map -profile -n 2 ./code [args]

The *.map file produced during both interactive and non-interactive runs can be later analyzed via
- map code.map
MAP requires an X window client

- The GUI part of MAP requires an X window client on your computer
  - Already present under Linux and Mac
  - Under Windows, a third party software is required
  - Mobaxterm is a good (and free) solution for Windows, as it combines three applications in one (ssh, sftp, and X window clients):

  http://mobaxterm.mobatek.net
Demo