Profiling and Performance Tuning:

Profiling with Allinea MAP

Ramses van Zon
SciNet HPC Consortium
University of Toronto

April 24, 2013

What is MAP?

- Parallel (MPI) performance analyser
- Graphical user interface
- Similar job startup interface and scalability as Allinea DDT
- Easy to use, low overhead

MAP Features

- A sampling profiler with adaptive sampling rates to keep the data volumes collected under control.

  Samples are aggregated at all levels to preserve key features of a run without drowning in data.

- A folding code and stack viewer allows you to zoom into time spent on individual lines and draw back to see the big picture across nests of routines.

- MAP measures memory usage, floating-point calculations and MPI usage.

- Both interactive and batch modes for gathering profile data.

Preparing your executable

- Compile as usual, but with -g at compile and link time to get symbol information.

```
mpicc -g -O2 hello.c -o hello
mpicxx -g -O2 hello.cpp -o hello
mpif90 -g -O2 hello.f90 -o hello
```

- (A bit more involved for statically linked apps)

Profiling the executable: 1. Interactive

- Request a interactive job

```
qsub -I -X -l nodes=1:ppn=8,walltime=1:00:00 -q debug
```

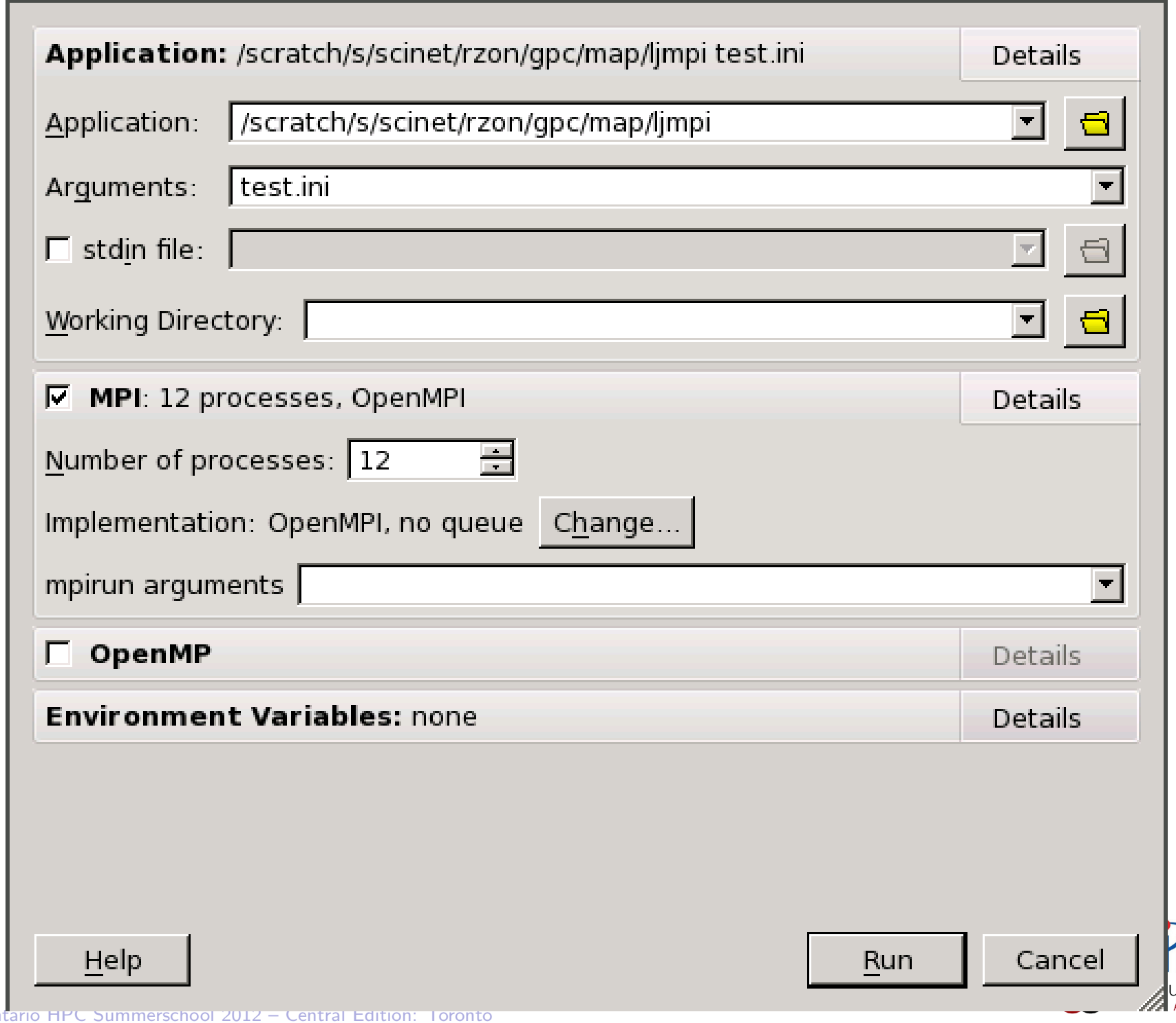
- Load compiler, mpi, and `ddt` module (latter contains map):

```
module load ddt
```

- Run:

```
map APP ARGS
```

- Select number of MPI processes and other options.


Application: /scratch/s/scinet/rzon/gpc/map/ljmpi test.ini
Details
Application:
/scratch/s/scinet/rzon/gpc/map/ljmpi
Arguments:
test.ini
stdin file:
Working Directory:
MPI: 12 processes, OpenMPI
Details
Number of processes:
12
Implementation: OpenMPI, no queue
Change...
mpirun arguments
OpenMP
Details
Environment Variables: none
Details
Help
Run
Cancel

Profiling the executable: 2. Non-interactive

- Write a normal jobscript.
- Replace your `mpirun` command, e.g.

```
mpirun -np P APP ARGS < INPUT
```

with

```
module load ddt
map -profile -n P APP ARGS -stdin INPUT
```

- Submit as usual:

```
qsub JOBSCRIPT
```

- and wait.

What is happening while MAP runs?

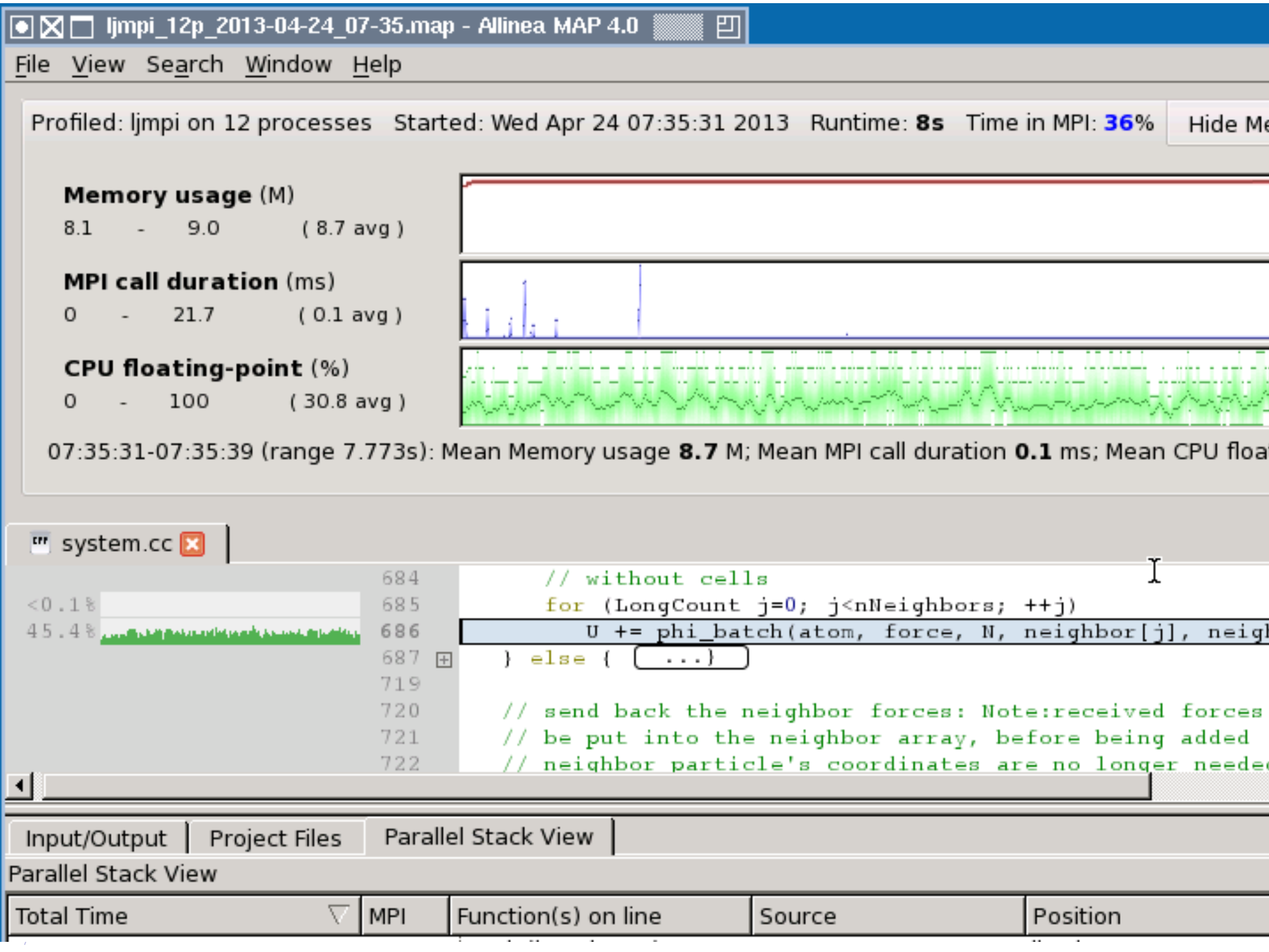
- Program runs inside MAP which collects statistics on your program through wrapper MPI calls.
- This is done by sampling.
- Longer runs will have a slower sampling rate to reduce the profiling data.
- Note that input cannot be given by piping into map. There is a stdin file instead.
- Likewise, output is redirected to an input/output box.

Exploring the results in MAP

- If interactively gathered, map will show the results after the run has finished.
- Otherwise, a MAPFILE with extension `.map` will have been created, which you can feed to map:

```
map MAPFILE
```

- and then you get...


ljmpi_12p_2013-04-24_07-35.map - Allinea MAP 4.0
File View Search Window Help
Profiled: ljmpi on 12 processes Started: Wed Apr 24 07:35:31 2013 Runtime: 8s Time in MPI: 36%
Hide Me
Memory usage (M)
8.1 - 9.0 (8.7 avg)
MPI call duration (ms)
0 - 21.7 (0.1 avg)
CPU floating-point (%)
0 - 100 (30.8 avg)
07:35:31-07:35:39 (range 7.773s): Mean Memory usage 8.7 M; Mean MPI call duration 0.1 ms; Mean CPU floa
system.cc
<0.1%
45.4%
684 // without cells
685 for (LongCount j=0; j<nNeighbors; ++j)
686 U += phi_batch(atom, force, N, neighbor[j], neigh
687 } else { ...}
719
720 // send back the neighbor forces: Note:received forces
721 // be put into the neighbor array, before being added
722 // neighbor particle's coordinates are no longer neede
Input/Output
Project Files
Parallel Stack View
Parallel Stack View
Total Time
MPI
Function(s) on line
Source
Position

Exploring the results in MAP

What can you do now?

- Select the CPU view to see the percentage of vectorized SIMD instructions used in each part of the code
- See how the amount of time spent in memory operations varies over time and processes.
- Zoom in to any part of the timeline, isolate a single iteration and explore its behaviour in detail
- Shows aggregated data and distributions rather than lists of processes and threads

GUI elements of MAP

- Program output
- Source code view
- Parallel stack view
- Project files view
- Metrics view

Available metrics in MAP

- Memory usage
- MPI call duration
- MPI bytes send/received
- MPI point-to-point/collective operations
- CPU floating-point operations
- CPU floating-point vector operations
- CPU integer operations
- CPU integer vector operations
- CPU memory access
- CPU branch

Examples

Extra: X forwarding

- Graphical applications on GPC require an *X-Windows* server.
- X-Windows is the standard graphics environment for unix and linux, and is supported on Mac OS X too.
  For Microsoft Windows, several free X servers exist as well, notably as part of Cygwin and MobaXterm.
- If you have an X server running on your own local machine, you can forward that service while you ssh into scinet:

```
$ ssh  -X USER@login.scinet.utoronto.ca
$ ssh  -X gpc02
$ qsub -X -I -q debug ...
```

- Doing so, your graphical applications, like map, are displayed on your local machine, and listen to you keyboard and mouse.
- This works well if your network connection is good.

Extra: VNC

- If your network connection is not so good (e.g. home internet), *VNC* can be faster.
- This involves:
  1. X-windows, windows manager, and VNC server on GPC
  2. An *ssh tunnel* from your local machine to that node
  3. A VNC client on your local machine.

Extra: VNC

1. On the GPC side, things areautomated using the vnc module:

```
ssh USER@login.scinet.utoronto.ca
ssh gpc04
module load vnc
vnc start
```

   First time, this asks for a password for VNC sessions.
   Note down the PORT number that `vnc start` returns.

2. Start a tunnel from you local machine

```
ssh -L5902:gpc04:PORT USER@login.scinet.utoronto.ca
```

3. With a vnc client, connect to `localhost:2`, e.g.

```
xvncviewer -encodings 'copyrect hextile' localhost:2
```

   In the X-windows environment, left-click to get menu.

More info

- User manual on SciNet:
  /scinet/gpc/tools/ddt/4.0/doc/userguide.pdf
- On VNC:
  http://wiki.scinethpc.ca/wiki/images/3/36/Ttvnc.pdf