Introduction to profiling

Martin Čuma
Center for High Performance Computing
University of Utah
m.cuma@utah.edu
Overview

- Profiling basics
- Simple profiling
- Open source profiling tools
- Intel development tools
  - Advisor XE
  - Inspector XE
  - VTune Amplifier XE
  - Trace Analyzer and Collector
Why to profile

- Evaluate performance
- Find the performance bottlenecks
  - inefficient programming
  - memory I/O bottlenecks
  - parallel scaling
Tools categories

• Hardware counters
  – count events from CPU perspective (# of flops, memory loads, etc)
  – usually need Linux kernel module installed

• Statistical profilers (sampling)
  – interrupt program at given intervals to find what routine/line the program is in

• Event based profilers (tracing)
  – collect information on each function call
Simple profiling

- Time program runtime
  - get an idea on time to run and parallel scaling
- Serial profiling
  - discover inefficient programming
  - computer architecture slowdowns
  - compiler optimizations evaluation
  - gprof
Open source tools

- Vendor based
  - AMD CodeAnalyst
- Community based
  - perf
    - hardware counter collection, part of Linux
  - oprofile
    - profiler
  - drawback – harder to analyze the profiling results
• We have a limited license
• Tools for all stages of development
  – Compilers and libraries
  – Verification tools
  – Profilers
• More info
• Intel Parallel Studio XE 2015 Cluster Edition
  – Compilers (C/C++, Fortran)
  – Math library (MKL)
  – Threading library (TBB)
  – Thread design and prototype (Advisor)
  – Memory and thread debugging (Inspector)
  – Profiler (VTune Amplifier)
  – MPI library (Intel MPI)
  – MPI analyzer and profiler (ITAC)
• Thread checking
  – Data races and deadlocks
• Memory checker
  – Like leaks or corruption
• Standalone or GUI integration
• More info

**Intel Inspector**

- **Source the environment**
  ```
  source /uufs/chpc.utah.edu/sys/pkg/intel/composerxe-2011.1.107/bin/compilervars.csh intel64
  addpath "/uufs/chpc.utah.edu/sys/pkg/intel/inspector_xe_2011/bin64:$PATH"
  ```

- **Compile with** `-tcheck -g`
  ```
  ifort -openmp -tcheck -g trap.f
  ```

- **Run tcheck**
  ```
  inspxe-gui – graphical user interface
  inspxe-cl – command line
  ```

- **Tutorial**
  ```
  ```
• Serial and parallel profiler
  – multicore support for OpenMP and OpenCL on CPUs, GPUs and Xeon Phi
• Quick identification of performance bottlenecks
  – various analyses and points of view in the GUI
• GUI and command line use
• More info
• **Source the environment**
  
  ```bash
  source /uufs/chpc.utah.edu/sys/pkg/intel/ics/vtune_amplifier_xe_2015/amplxe-vars.csh
  ```

• **Run VTune**
  
  ```bash
  amplxe-gui – graphical user interface
  amplxe-cl – command line (best to get from the GUI)
  ```

Can be used also for remote profiling (e.g. on Xeon Phi)

• **Tuning guides for specific architectures**

• Thread design and prototyping
  – Analyze, design, tune and check your threading design before implementation
  – Explore and test threading options without disrupting normal development
  – Predict threading errors & performance scaling on systems with more cores

• More info
• MPI profiler
  – traces MPI code
  – identifies communication inefficiencies
• Collector collects the data and Analyzer visualizes them
• More info
• Source the environment

```
source /uufs/chpc.utah.edu/sys/pkg/intel/ics/itac/std/bin/
  itacvars.csh
source /uufs/chpc.utah.edu/sys/pkg/intel/ics/impi/std/intel64/
  bin/mpivars.csh
```

• Using Intel compilers, can compile with `-trace`

```
mpiifort -openmp -trace trap.f
```

• Run MPI code

```
mpirun -trace -n 4 ./a.out
```

• Run visualizer

```
traceanalyzer a.out.stf &
```

• CHPC site

```
https://wiki.chpc.utah.edu/display/DOCS/Intel+Cluster+Toolkit
```

https://wiki.chpc.utah.edu/display/DOCS/Intel+Cluster+Toolkit