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
- Interpreted languages profiling
- https://www.surveymonkey.com/r/7PFVFCY
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
    - Trick how to get gprof to work in parallel:
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
HPC OS tools

• HPC Toolkit
  – A few years old, did not find it as straightforward to use

• TAU
  – Lots of features, which makes the learning curve slow

• Scalasca
  – Developed by European consortium, did not try yet
• We have a 2 concurrent users license
• Tools for all stages of development
  – Compilers and libraries
  – Verification tools
  – Profilers
• More info

https://www.chpc.utah.edu/documentation/software/intel-parallelXE.php
Intel tools

- Intel Parallel Studio XE 2018 Cluster Edition
  - Compilers (C/C++, Fortran)
  - Distribution for Python
  - Math library (MKL)
  - Data Analytics Acceleration Library (DAAL)
  - Threading library (TBB)
  - Vectorization or thread design and prototype (Advisor)
  - Memory and thread debugging (Inspector)
  - Profiler (VTune Amplifier)
  - MPI library (Intel MPI)
  - MPI analyzer and profiler (ITAC)
Intel VTune Amplifier

• 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
  module load vtune

• Run VTune
  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
Intel Advisor

• Vectorization advisor
  – Identify loops that benefit from vectorization, what is blocking efficient vectorization and explore benefit of data reorganization

• Thread design and prototyping
  – Analyze, design, tune and check threading design without disrupting normal development

• More info
Intel Advisor

- Source the environment module load advisorxe
- Run Advisor
  - advixe-gui – graphical user interface
  - advixe-cl – command line (best to get from the GUI)
- Create project and choose appropriate modeling
- Getting started guide
Intel Trace Analyzer and Collector

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

• Using Intel compilers, can compile with \texttt{-trace}
  mpiifort \texttt{-openmp -trace trap.f}

• Run MPI code
  mpirun \texttt{-trace -n 4 ./a.out}

• Run visualizer
  traceanalyzer a.out.stf &

• CHPC site
Interpreted languages profiling

- With increased use of interpreted languages, their performance is becoming important
- Matlab
  - Profiling ecosystem in the IDE
- Python
  - Python modules or IDEs
- R
  - Profiling libraries or RStudio
Matlab

- **profile** command turns on/off profiling
- Profile is then displayed in the IDE
- Click on each function to show line-by-line profile

- Performance improvement strategies
• profile and cProfile modules
  – Text based output, optional format with pstats, analysis with Stats

• Plethora of other tools
  – E.g. line profiling with line_profiler

• Some IDEs display profiles
  – Spyder
- **Rprof function** to profile
- **summaryRprof** to display
- RStudio has a profile interface called profviz

- Performance improvement strategies
  http://adv-r.had.co.nz/Profiling.html
Summary

- Serial profilers
  - gprof, perf
- Intel tools
  - VTune, AdvisorXE, ITAC
- Interpreted languages profiling
  - Matlab profile
  - Python profile, Cprofile
  - R Rprof, profviz
- https://www.surveymonkey.com/r/7PFVFCY
• https://www.surveymonkey.com/r/7PFVFCY