A Brief Introduction to HPCToolkit

Keren Zhou
Department of Computer Science
Rice University

http://hpctoolkit.org
Outline

• Overview of Rice’s HPCToolkit
• OpenMP issues
• Using HPCToolkit’s GUIs to analyze program performance
• Other capabilities
Rice University’s HPCToolkit

• Employs binary-level measurement and analysis
  — observe fully optimized, dynamically linked executions
  — support multi-lingual codes with external binary-only libraries

• Uses sampling-based measurement (avoid instrumentation)
  — controllable overhead
  — minimize systematic error and avoid blind spots
  — enable data collection for large-scale parallelism

• Collects and correlates multiple derived performance metrics
  — diagnosis often requires more than one species of metric

• Associates metrics with both static and dynamic context
  — loop nests, procedures, inlined code, calling context

• Supports top-down performance analysis
  — identify costs of interest and drill down to causes
    – up and down call chains
    – over time
HPCToolkit Workflow

- compile & link
- source code
- optimized binary
- profile execution [hpcrun]
- call path profile
- binary analysis [hpcstruct]
- program structure
- interpret profile correlate w/ source [hpcprof/hpcprof-mpi]
- presentation [hpcviewer/hpctraceviewer]
- database
For dynamically-linked executables, e.g., Linux clusters
— compile and link as you usually do: nothing special needed
— For statically-linked executables (Cray default)
— add monitoring by using `hpclink` as prefix to your link line
 Measure execution unobtrusively
- launch optimized application binaries
  - dynamically-linked: launch with \texttt{hpcrun}, arguments control monitoring
  - statically-linked: environment variables control monitoring
- collect statistical call path profiles of events of interest
Call Path Profiling

Measure and attribute costs in context

- Sample timer or hardware counter overflows
- Gather calling context using stack unwinding

Call path sample:
- Return address
- Return address
- Return address
- Instruction pointer

Calling context tree:

Overhead proportional to sampling frequency...
...not call frequency
HPCToolkit Workflow

- Analyze binary with **hpcstruct**: recover program structure
  - analyze machine code, line map, debugging information
  - extract loop nests & identify inlined procedures
  - map transformed loops and procedures to source

**presentation**
[hpctraceviewer]

**interpret profile correlate w/ source**
[hpcprof/hpcprof-mpi]

**databa se**

**compile & link**

**profile execution**
[hpcrun]

**binary analysis**
[hpcstruct]

**call path profile**

**program structure**
HPCToolkit Workflow

- Combine multiple profiles
  — multiple threads; multiple processes; multiple executions
- Correlate metrics to static & dynamic program structure

presentation
[hpcviewer/hpctraceviewer]
**HPCToolkit Workflow**

- **Presentation**
  - **explore performance data from multiple perspectives**
    - rank order by metrics to focus on what’s important
    - compute derived metrics to help gain insight
      e.g. scalability losses, waste, CPI, bandwidth
  - graph thread-level metrics for contexts
  - explore evolution of behavior over time

**Diagram Description**

- **source code** → **optimized binary** → **profile execution** → **call path profile** → **program structure**
- **compile & link**
- **binary analysis**
  - [hpcstruct]
- **interpret profile** → **correlate w/ source**
  - [hpcprof/hpcprof-mpi]
- **databa se**
- **presentation**
  - [hpcviewer/ hpctraceviewer]
Code-centric Analysis with hpcviewer

- function calls in full context
- inlined procedures
- inlined templates
- outlined OpenMP loops
- loops

Source pane

View control

Metric display

Navigation pane

Metric pane
Profiling compresses out the temporal dimension—temporal patterns, e.g. serialization, are invisible in profiles.

What can we do? Trace call path samples—sketch:

- N times per second, take a call path sample of each thread
- Organize the samples for each thread along a time line
- View how the execution evolves left to right
- What do we view?

Assign each procedure a color; view a depth slice of an execution.
Time-centric analysis: load imbalance among threads appears as different lengths of colored bands along the x axis.
OpenMP: A Challenge for Tools

- Large gap between threaded programming models and their implementations

User-level calling context for code in OpenMP parallel regions and tasks executed by worker threads is not readily available

- Runtime support is necessary for tools to bridge the gap
Challenges for OpenMP Node Programs

- Tools provide implementation-level view of OpenMP threads
  - asymmetric threads
    - master thread
    - worker thread
  - run-time frames are interspersed with user code

- Hard to understand causes of idleness
  - long serial sections
  - load imbalance in parallel regions
  - waiting for critical sections or locks
OMPT: An OpenMP Tools API

- **Goal:** a standardized tool interface for OpenMP
  - prerequisite for portable tools
  - missing piece of the OpenMP language standard

- **Design objectives**
  - enable tools to measure and attribute costs to application source and runtime system
    - support low-overhead tools based on asynchronous sampling
    - attribute to user-level calling contexts
    - associate a thread’s activity at any point with a descriptive state
  - minimize overhead if OMPT interface is not in use
    - features that may increase overhead are optional
  - define interface for trace-based performance tools
  - don’t impose an unreasonable development burden
    - runtime implementers
    - tool developers
Integrated View of MPI+OpenMP with OMPT

LLNL’s luleshMPI_OMP (8 MPI x 3 OMP), 30, REALTIME@1000

source view

thread view

metric view
OpenMP Tool API Status

- HPCToolkit supports OpenMP 5.0 OMPT
- OMPT prototype implementations
  - LLVM (emerging: OpenMP 5.0)
    - interoperable with GNU, Intel compilers
  - IBM LOMP (currently targets OpenMP 4.5)
- Ongoing work
  - refining OpenMP 5.0 OMPT support in LLVM OpenMP
  - refining OpenMP 5.0 OMPT support in HPCToolkit
    - asynchronous call stack assembly for lightweight monitoring
HPCToolkit Capabilities for GPU Code

MPI + OpenMP 4.5 or CUDA GPU accelerated applications
Other Capabilities

• **Measure hardware counters using Linux perf_events**
  — available events can be listed with
    - `hpcrun -L`
    - launching a binary created by hpclink with environment setting
      `HPCRUN_EVENT_LIST=LIST`
  — frequency based sampling: 300/s per thread or machine max
    - no need to set periods or frequencies unless you want precise control
  — hardware event multiplexing
    - measure more events than hardware counters

• **Kernel sampling**
  — measure activity in the Linux kernel in addition to your program
    - e.g., allocating and clearing memory pages
  — not available on BG/Q
  — measurement and attribution subject to system permissions
    - detailed attribution not available on NERSC or ANL systems
HPCToolkit at NERSC

• NERSC cori
  — a setup script or a set of module loads
  — source /global/cscratch1/sd/kz21/env-static.sh
  — source /global/cscratch1/sd/kz21/env-shared.sh

• Man pages
  — automatically added to MANPATH by the aforementioned command
HPCToolkit at ORNL

• On Summit
  — module use /gpfs/alpine/csc322/world-shared/modulefiles
  — module load hpctoolkit

• Man pages
  — automatically added to MANPATH by the aforementioned command
GUIs for your Laptop

- Download binary packages for HPCToolkit’s user interfaces on your laptop
  - [http://hpctoolkit.org/download/hpcviewer](http://hpctoolkit.org/download/hpcviewer)
Detailed HPCToolkit Documentation

http://hpctoolkit.org/documentation.html

• Comprehensive user manual:
  
  
  — Quick start guide
  — essential overview that almost fits on one page
  
  — Using HPCToolkit with statically linked programs
  — a guide for using hpctoolkit on BG/Q and Cray platforms
  
  — The hpcviewer and hpctraceviewer user interfaces
  
  — Effective strategies for analyzing program performance with HPCToolkit
  — analyzing scalability, waste, multicore performance ...
  
  — HPCToolkit and MPI
  
  — HPCToolkit Troubleshooting
  — why don’t I have any source code in the viewer?
  — hpcviewer isn’t working well over the network ... what can I do?

• Installation guide
Advice for Using HPCToolkit
Using HPCToolkit

- Add hpctoolkit’s bin directory to your path using softenv
- Adjust your compiler flags (if you want full attribution to src)
  - add -g flag after any optimization flags
- Add hpclink as a prefix to your Makefile’s link line
  - e.g. hpclink CC -o myapp foo.o ... lib.a -lm ...
- See what sampling triggers are available on Cray
  - use hpclink to link your executable
  - launch executable with environment variable
    HPCRUN_EVENT_LIST=LIST
    - you can launch this on 1 core of 1 node
    - no need to provide arguments or input files for your program
      they will be ignored
Monitoring Large Executions

• Collecting performance data on every node is typically not necessary

• Can improve scalability of data collection by recording data for only a fraction of processes
  — set environment variable `HPCRUN_PROCESS_FRACTION`
  — e.g. collect data for 10% of your processes
    – set environment variable `HPCRUN_PROCESS_FRACTION=0.10`
Digesting your Performance Data

• Use hpcstruct to reconstruct program structure
  — e.g. hpcstruct your_app
    – creates your_app.hpcstruct

• Correlate measurements to source code with hpcprof and hpcprof-mpi
  — run hpcprof on the front-end to analyze data from small runs
  — run hpcprof-mpi on the compute nodes to analyze data from lots of nodes/threads in parallel
    – notes
      much faster to do this on an x86_64 vis cluster (cooley) than on BG/Q
      avoid expensive per-thread profiles with --metric-db no

• Digesting performance data in parallel with hpcprof-mpi
  — qsub -A ... -t 20 -n 32 --mode c1 --proccount 32 --cwd `pwd` \
   /projects/Tools/hpctoolkit/pkgs-vesta/hpctoolkit/bin/hpcprof-mpi \
   -S your_app.hpcstruct \
   -l /path/to/your_app/src/+ \
   hpctoolkit-your_app-measurements.jobid
Analysis and Visualization

• Use hpcviewer to open resulting database
  — warning: first time you graph any data, it will pause to combine info from all threads into one file

• Use hpctraceviewer to explore traces
  — warning: first time you open a trace database, the viewer will pause to combine info from all threads into one file

• Try our our user interfaces before collecting your own data
  — example performance data
    http://hpctoolkit.org/examples.html
Installing HPCToolkit GUIs on your Laptop

- See http://hpctoolkit.org/download/hpcviewer
- Download the latest for your laptop (Linux, Mac, Windows)
  - hpctraceviewer
  - hpcviewer

A Note for Mac Users
When installing HPCToolkit GUIs on your Mac laptop, don’t simply download and double click on the zip file and have Finder unpack them. Follow the Terminal-based installation directions on the website to avoid interference by Mac Security.