Yet Another Overview on Performance Measurement and Analysis Tools *for Intel x86-64 Architecture & Linux OS*

Andrea Valassi on behalf of the UP team (CERN IT-DI-LCG)
Slides prepared by Servesh Muralidharan and David Smith

WLCG-HSF Workshop, Napoli, 28th March 2018

Introduction

Open Source Tools

- Performance Analysis
  - Linux perf events
  - gperftools
  - GNU gprof
- Memory or I/O Analysis
  - valgrind
  - iostat
  - vmstat
- Miscellaneous
  - Linux /proc filesystem

Commercial Tools

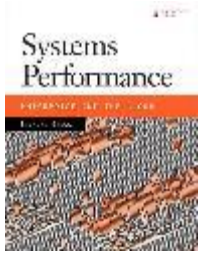
- Intel Compiler Optimization Report
- Intel VTune
- Intel Advisor

Let us know what else
the HEP community uses and
we will add them to our list!!!

We are compiling a list with guidelines

Community Tools

- HEP Experiments
  - prmon
  - igprof
- Others
  - MALT, numaprof
- WLCG-UP
  - FOM, Trident



Focusing on CPU, I/O, memory – not including network (netstat, nw2), debugging (gdb, Coverity)...

Which tools are most used in the experiments?

- ATLAS
  - valgrind, gperftools, /proc (via Athena perfmon instrumentation) – during development
  - prmon – for production jobs
- CMS
  - valgrind, igprof, vtune – during development
  - /proc – for production jobs (data stored on ElasticSearch)
- LHCb
  - valgrind, VTune (both with Gaudi integration, e.g. start/stop)
  - (much less: perf, gperftools)

See also Ben Couturier's talk in the software development session

Thanks to Johannes Elmsheuser, David Lange, Sebastien Ponce for the feedback!

Linux perf profiling

- Source: https://perf.wiki.kernel.org/index.php/Main_Page
- Linux command included in tools/perf
- Instrument processor pmu counters, tracepoints, probes
- Hardware counter profiling such as:
  - instructions executed, cache-misses, branch mis-prediction, memory transfer rate, etc.,
- Event sampling and source code event annotation

gperftools

- Source: https://github.com/gperftools/gperftools
- A collection of thread safe performance tuned malloc library with a few lightweight performance analysis tools
- TCMALLOC - thread safe performance tuned malloc library
- HEAP PROFILER – analyze heap memory usage (pprof)
- HEAP CHECKER – catch heap memory leaks
- CPU PROFILER – call graph analysis (pprof)

GNU gprof

- Source: https://sourceware.org/binutils/docs/gprof/
- Execution time for different parts of the program
- Analysis through flat profile, call graph and annotated source for execution time

valgrind

- Source: http://valgrind.org/
- Dynamic analysis and instrumentation
- Ability to instrument the operations of a program
- The instrumentation *framework* supports extensible tools
- Memcheck, Cachegrind, Callgrind, Helgrind, DRD, Massif, etc…

iostat

- Source: https://linux.die.net/man/1/iostat
- Statistics related to Input and Output operations on block devices or partitions, and CPU usage according to the scheduler
- Snapshot tool, i.e. collects data system wide for a given resource
- Does not provide instrumentation or source code annotation

vmstat

- Source: https://linux.die.net/man/8/vmstat
- Collects virtual memory usage, summary of block device I/O, CPU usage
- Collects data system wide

Linux /proc filesystem

- Source: https://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html
- Virtual filesystem supported in Linux OS
- Variety of real time system information
- Supports altering kernel behavior at runtime

Intel Compiler Optimization Report

- Source: https://software.intel.com/en-us/articles/getting-the-most-out-of-your-intel-compiler-with-the-new-optimization-reports
- Optimization report that provides insight into compiler transformations
  - –SIMD instructions, automatic parallelization, loop transformations, etc.,

Intel VTune Amplifier

- Source: https://software.intel.com/en-us/intel-vtune-amplifier-xe
- Advanced analysis through a single GUI interface
- Supports CPU, GPU, memory and threading performance analysis
- Variety of derived metrics and proprietary analysis techniques
- Source code annotation with different analysis views for e.g. Top Down

Intel Advisor

- Source: https://software.intel.com/en-us/advisor
- Focuses on vectorization and cache efficiency
- In-depth memory dependency analysis for loops
- Roofline analysis to highlight candidates for optimization

HEP Experiment's Application Tools

- PRMON (PRocess MONitor)
  - –Source: https://github.com/HSF/prmon
  - –Program to monitor the resource consumption of a process and its children
- IgProf (the Ignominious Profiler)
  - –Source: https://igprof.org
  - –CPU profiler (sampling), memory profiler, or instrumentation mode

Others

- MALT
  - –Source: https://github.com/memtt/malt
  - –Tool to find where memory is allocated
- Numaprof
  - –Source: https://github.com/memtt/numaprof
  - –NUMAPROF is a NUMA memory profiler based on Intel Pintools

WLCG-UP Tools

- FOM Tools
  - Source: https://github.com/FOM-Tools/FOM-Tools
  - Studies memory allocation patterns of an application
  - Identifies obsolete memory
- Trident (under development)
  - A three pronged approach to analyzing node utilization
  - Combines CPU, Memory and IO node monitoring under a single tool

Tool	CPU	Memory	IO	System	App	Overhead	Annotate	Need recompile
Linux perf	✓	✓	✓	✓	✓		✓	✗
gperftools	✓	✓	✗	✗	✓		✓	✗
gprof	✓	✗	✗	✗	✓		✓	✓
valgrind	✓	✓	✗	✗	✓		✓	✗
iostat	✓	✗	✓	✓	✗		✗	✗
vmstat	✓	✓	✓	✓	✗		✗	✗
/proc	✓	✓	✓	✓	✓		✗	✗
ICC Report	✓	✓	✗	✗	✓		✓	✓
VTune	✓	✓	✗	✗	✓		✓	✗
Advisor	✓	✓	✗	✗	✓		✓	✗
prmon	✓	✓	✗	✗	✓		✗	✗
IgProf	✓	✓	✗	✗	✓		✓	✗
MALT	✗	✓	✗	✗	✓		✓	✗
NUMAPROF	✗	✓	✗	✗	✓		✓	✗
FOM	✗	✓	✗	✗	✓		✓	✗
Trident	✓	✓	✓	✓	✗		✗	✗

Conclusions and outlook

- Large and powerful arsenal of tools
  - –Mastering even only one of them takes time – training is very important
  - –Build wrappers around some tools to lower the threshold for less experienced users?
  - –Community tools better than experiment-specific ones, if generic tools not enough
- Important to identify the most important tools and build expertise for them
  - –Let us know if other tools that you heavily rely upon is missing
- Important to correlate tools and application-level information
  - –This is especially important to build a cost model for the infrastructure
  - –Throughput and resource needs for different workflows and application phases