PerfSuite
An Accessible, Open Source Performance Analysis Environment for Linux

Rick Kufrin
University of Illinois / NCSA
rkufrin@ncsa.uiuc.edu

Linux Clusters: The HPC Revolution 2005
Chapel Hill, North Carolina
April 27, 2005

Topics

• Motivation and background
• Linux and OSS community developments
• Design and implementation
• Status, examples, use
• Futures (near term and beyond)
NCSA Major System History

• 1985 - 1994: Cray X-MP / Y-MP / 2 (CTSS / UNICOS)
• 1998: IA-32 “SuperCluster” (WinNT)
• 2000 - 2004: PIII cluster (Linux)
• 2002 - 2004: Itanium cluster (Linux)
• 2003 – present: IBM p690 (AIX), Xeon / Itanium 2 clusters (Linux)
• 2005 – present: SGI Altix (Linux)

Migration to Linux and Performance Analysis

• Apart from gprof and time, not a great deal available, especially cross-platform
  – …and on ia64, gprof/profil was broken
• In-progress work for performance data visualization dependent on output of IRIX perfex tool
User perspective “desirables”

• Straightforward, easy-to-use
• Little or no source code modification
• Steer clear of “hacker” / “bleeding edge” development style as much as possible
• Data accessible to the consumer (the user!)

“Enabling” OSS Efforts

• Perfctr (x86, x86-64), Perfmon (ia64) hardware performance counter drivers / libraries
  – UT-K’s PAPI development and user community
• High-performance, stable scripting languages
• Maturation of XML and related technologies
PerfSuite Architecture

• Simplified diagram of key components for profiling & hardware counting
• Shaded components optional

libperfsuite

libpshwpc

libpshwpc

PAPI

PSTcl

glibc

psrun

pprocess

psprocess

psrun

psrun

Items in orange are those used by psrun, items in italics new in 2005.
PerfSuite Tools

- Four performance counter-related utilities:
  - psconfig - configure / select performance events
  - psinv - query events and machine information
  - psrun - generate raw counter data from an unmodified binary
  - psprocess - post-process data

psrun

- Hardware performance counting and profiling with unmodified executables
- Available for x86, x86-64, ia-64
- POSIX threads support
- Automatic multiplexing
- Can be used with MPI
- Optionally collects resource usage system information (e.g., load averages)
psprocess (text output)

PerfSuite Hardware Performance Summary Report
Version : 1.0
Created : Mon Dec 30 11:31:53 AM
Generator : psprocess 0.5
XML Source : psrun-ia64.xml

Execution Information
===========================
Date : Sun Dec 15 21:01:20 2002
Host : user01

Processor and System Information
===========================
Node CPUs : 2
Vendor : Intel
Family : IPF
Model : Itanium
CPU Revision : 6
Clock (MHz) : 800.136
Memory (MB) : 2007.16
Pagesize (KB) : 16

Cache Information
==========================
Cache levels : 3
Level 1
Type : data
Size (KB) : 16
Linesize (B) : 32
Assoc : 4
Level 2
Type : instruction
Size (KB) : 16
Linesize (B) : 32
Assoc : 4

Index Description                                   Counter Value
=================================================================
1 Conditional branch instructions mispredicted...    4831072449
4 Floating point instructions......................   86124489172
5 Total cycles.....................................  594547754568
6 Instructions completed........................... 1049339828741

Statistics
=================================================================
Graduated instructions per cycle.....................         1.765
Graduated floating point instructions per cycle...         0.345
Level 3 cache miss ratio (data).......................         0.557
Bandwidth used to level 3 cache (MB/s).................       385.087
% cycles with no instruction issue....................        10.410
% cycles stalled on memory access....................        43.139
MFLOPS (cycles)..................................       215.905
MFLOPS (wallclock)................................  114.441

psprocess (cont’d)

Index Description                                   Counter Value
=================================================================
1 Conditional branch instructions mispredicted...    4831072449
4 Floating point instructions......................   86124489172
5 Total cycles.....................................  594547754568
6 Instructions completed........................... 1049339828741

Report creation details
Run details
Machine information
Raw counter listings
Counter explanations and index
Derived metrics
Run annotation defined by user

http://perfsuite.sourceforge.net 11

http://perfsuite.sourceforge.net 12
Customizable Derived Metrics

- The metrics derived from the counter data are acquired “on-the-fly”, from an XML description. The user can replace/change.

```
<metric namespace="PAPI" type="ratio">
  <name>PS_RATIO_GINS_CYC</name>
  <description lang="en_US">Graduated instructions per cycle</description>
  <description lang="es">Graduados instrucciones por ciclo</description>
  <definition>
    <apply>
      <divide>
        <ci>PAPI_TOT_INS</ci>
        <ci>PAPI_TOT_CYC</ci>
      </divide>
    </apply>
  </definition>
</metric>
```

Automatic Performance Data Collection Project

- 10% of peak or greater: 12% on Pentium III, 7% on Itanium (from ‘03 pilot project)
- System software conflicts suspended project in ’04, now in process of resumption
- Current implementation relies on direct access/use of Perfctr and Perfmon
- Shared-memory (Altix) apps now included in auto-collection; batch system causes all processes to be measured
Data Reuse By External Tools

Attribute Explorer (Imperial College / IBM) and the Visual Profiler (Sandia Laboratories)

http://perfsuite.sourceforge.net

“Copper” (NSF ST-HEC)

Collaborators:
B. Chapman (Houston)
R. Kufrin (Illinois)
D. Tafti (Virginia Tech)
F. Wolf (Tennessee)
Current Status

• Stable and deployed on production systems @ NCSA since late 2002

• Initial public release: Dec. 2003
  – Available from SourceForge and NCSA
  – Downloaded to approximately 1000 sites from 12/03 - present

• Increase in deployment and training externally (= acceptance)

Near-Term Futures

• Current (“0.6.2 alpha”) version focusing on robustness throughout ’05
  – Expanded API
  – Support for native performance libraries (e.g., Perfmon)

• Java support will be phased in
  – Design and initial implementation completed
  – JVMTI and JNI interfaces provide true “object” feel vs. library wrapper approach