PTU – Intel® Performance Tuning Utility
Intel® PTU Sample View
Basic block grouping and execution flow analysis

[Image of software interface showing a detailed view of computer code and execution flow]
Core™ Architecture PMU

The PMU of Intel® Core™ architecture offers some 700 events (including sub-events) compared to
  – Netburst™ Architecture: Some 140

Event counters (5 in total) are not identical
  – Three frequently used ("basic") events INST_RETIRED.ANY, CPU_CLK_UNHALTED.CORE and CPU_CLK_UNHALTED.REF have dedicated counter registers (1:1 mapping)
  – Some events are only counted by PMC0 (precise and at-retirement events not belonging to the three events above)
  – All other events are counted either by PMC0 or PMC1

Counter structure allows collection of up to 5 events in one run - however, depending on the event selection, typically 2-3 only
  – This is important to remember when running VTune/PTU Sampling
PTU – Background and Motivation

A replacement for Intel® VTune™ Performance Analyzer?
- No! - but an alternative
- A lot of components are shared (e.g. Linux* Sampling drivers)
- A free tool but requires Intel® VTune™ Performance Analyzer license

A tool for experienced developers – Intel internally and externally
- Less focus on user guidance and help functionality as VTune
- Focus is on getting results fast; pragmatic view
- No registration/RPM/registry – simply unpack and start!
- Same look and feel / same GUI - on all platforms (Eclipse-based)
- Has been Intel-internal tool only but now made available to everybody!

A test environment for sophisticated new features/prototypes available in VTune (potentially) only in future releases
- Includes pre-launch platform support

Make simple things easy, make extraordinary possible
Intel® PTU Analysis Methods at a Glance

**Event Based Sampling (PMU hardware Events)**
- Optional event Multiplexing – more events collected in one run
- Includes sophisticated Data Access Profiling
- Includes features to compare multiple runs

**Statistical Call Graph Sampling (with loop detection)**
- No instrumentation / very low overhead
- Statistical (pragmatical) approach – thus not 100% precise

**Instrumentation-based Call Graph/Count Analysis**
- Instrumentation similar to VTune Callgraph but using PIN technology
- Intrusive (factor 2 or more) but 100% precise

**Heap / Memory Access Profiling**
Structural Components

Data collectors – command utilities
  - Driver based and non-driver based collectors
  - Collection controllers

Data Viewers – command utilities
  - Data converters
  - Symbol resolver
  - Filtering, Comparing and Displaying

Eclipsed-based GUI ("a plug-in for Eclipse")
  - Windows* and Linux* - single look and feel
  - Pre-built Eclipse package part of distribution

* Other names and brands may be claimed as the property of others.
PTU Architecture – High Level Overview

Eclipse* GUI

Configuration
Collection
Command Line
Queries
Data

Command Line Collectors
vtsarun, vtssrun

Command Line Viewers
vtsaview, vtsadiff, vtssview

Experiment Directory

SEP

Kernel Driver
PMU
Hardware

Data conversion
(symbols, tb5 -> database)

* Other names and brands may be claimed as the property of others.
# Command Line utilities

<table>
<thead>
<tr>
<th>Collectors</th>
<th>Viewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>vtsarun – Sampling collector</td>
<td>vtsaview – Sampling viewer</td>
</tr>
<tr>
<td></td>
<td>vttdpview – Data profiler viewer</td>
</tr>
<tr>
<td>vtssrun – Statistical Call graph collector</td>
<td>vtsadiff – Hotspot difference viewer</td>
</tr>
<tr>
<td>vtcgrun – Exact Call graph collector</td>
<td>vtssrun – Statistical Call graph viewer</td>
</tr>
<tr>
<td>vthprun – Heap profiler collector</td>
<td>vtcgview – Exact Call graph viewer</td>
</tr>
<tr>
<td></td>
<td>vthpview – Heap profiler viewer</td>
</tr>
</tbody>
</table>

Command line utilities can be used independent of graphical interface. GUI functionality is completely based on these “batch applications”.
Some Interesting Usability Features

Basic blocks as underlying data representation
• Aggregation, manipulation per basic block

Drill-down to Source, Disassembly and Control Flow graph view for a procedure
• All three (can be) displayed simultaneously

Results difference (text mode, csv export, GUI mode)
• GUI for different binaries, and for identical sources

CPU Frequency Changing
• Ability to modify CPU frequency via GUI

Configuration per PMU, export/import

Sampling Hotspot View expansion to show events per CPU

Event over IP chart

Assembler display choice ( Intel versus AT&T )
Main concepts

**Eclipse* Workspace**
- Project
  - Experiment
    - Modules (for experiment sharing)
    - Sources (for experiment sharing)

- Project defines the application and its workload (parameters)
- Simply zipping up an experiment creates a self contained test case

* Other names and brands may be claimed as the property of others.
PTU Profile Configurations

• Project is definition of the workload
• Profile Configuration is definition of collection method
  – You apply a Profile Configuration to a Project to collect Experiment

– GUI generates a command line for collector and invokes it

• Workspace, Project, Profile Configuration are in GUI world only
  – Command line tools know only about Experiments

• 5 predefined configurations come in PTU distribution
  – Basic Statistical Callgraph, Basic Sampling and others
  – You can create your own configurations as well

Project + Profile Configuration produces the Experiment
- Profiling Configuration is orthogonal to the Project concept
- Profiling Configuration created by user can be applied to any Project
Event based Sampling

Use Basic Sampling predefined configuration or create your own with events you need in the experiment.

Granularity in base display is function, module, basic block or instruction (RVA).

Standard drill down to source/asm by row in spreadsheet.
Add the Control Flow Graph with the Tool bar Button
All Three Views at once
Events over IP histogram

- In Hotspot view right click on row of interest, appropriate item invokes Events over IP histogram for selected module
- Drill-down to highlighted region of histogram to source and/or disassembly view
Data Access Profiling

Intel® Pentium® processors, Intel® Core™2 processor family and Itanium® Processor family have Precise Event Based Sampling (PEBS)

Precise events have hardware to capture the Instruction Pointer of the instruction that caused the event

• Ex: On Intel® Core™2 Processor family, an L2 load miss that retrieves a cacheline can be identified with MEM_LOAD RETIRED.L2_LINE_MISS

Further the values of all registers are also captured in a buffer

• At completion of the instruction on Intel® Core™2 Processor Family

• Prior to execution of the instruction on Pentium® 4 Processors

Coupled with the disassembly, the addresses of load and store instructions can be reconstructed
Basic Data Access Profiling – Easy to Use
Pre-Defined Event Set
Data Access Result: Here Heavy Contention on marked Line
Multiple Threads Accessing Different Offsets Indicate False Sharing
From one Hotspot View to two of them...

This was the model that we used so far...

Aggregate over time

But the reality is like this...

Aggregate over time + over one more dimension
Easy Navigation between both “Dimensions”

Filtering - projecting slices onto another dimension

Sorting – repositioning segments of the axes

Applying granularity – changing scale of the axis

Intel® PTU Data Access profiling operates with two hotspots views: Code and Data hotspots
Differences of EBS Measurements

Intel® PTU 3.x supports an analysis of differences of experiments

This requires

• Event names must be the same
• Load Modules have the same names
  – They can be the same, with data taken on different machines
  – They can be different but built from the same source
    • Allowing differences to be analyzed down to source view
  – They can be completely different (sources and binaries)
    • PTU will compare functions with the same names for modules with the same names

Single click on first experiment

Ctrl single click on second experiment

Right click and select compare
Data blocked 2X2 unrolled Matrix Multiply compiled at -O2
Data blocked 2X2 unrolled Matrix Multiply compiled at -O3 -QxT
Only Significant Difference is Cycle Count
Create Difference Display
Control click to select 2 experiments
Right click to select “Compare Experiments”
Differences of Samples
Differences in Cycles Shown in msec to Correct for Comparison of Machines at Different Frequencies
Same Source can Display Difference per Source Line
Statistical Call Graph

Along with standard features supports multithreaded apps and 32 bit binaries on 64 bit OS’s.

Easy to identify path to the hotspot
Exact Call Graph

• Reconstructs the application’s Call graph by instrumenting function entry and exit points using the PIN dynamic instrumentation framework

• Records a caller-callee pair together with timing and call count information

• **Text viewer** provides the following information:
  – List of functions with their parents, children, and timing information
  – Flat data: list of functions with timing and call information with no relationships
  – Per-thread view for multithreaded applications

• Collection overhead for computationally-intensive console applications 1.8-2x
Call Count GUI View

- Displays a limited set of exact call graph data: a function name, module name, and call count, which is a number of times the function was called.
Summary

• Intel® PTU - Easy-to-use performance debugging tool

• Focus on getting results quickly

• The tool is easy to use but the performance methodology supported perfectly by this tool requires some experience – it is not a tool for the beginner!!

• Using it and providing feedback /submitting bugs you help make it better for you and contribute to future Performance Suite
  – User Forum at whatif.intel.com

Call to Action:

• Download, install and use Intel® PTU!