Intel® System Studio
Intel® VTune™ Amplifier for Systems
Overview
What you will learn from this slide deck

• Intel® VTune™ Amplifier for Systems technical training for System & Application code running Linux*, Android* & Tizen™

• In-depth explanation of specifics for each development environment mentioned above

• Please see subsequent slide decks for in-depth technical training on other components

• Note: There are 2 other slides decks with in-depth topics for
  • Intel® Energy Profiler
  • Advanced VTune™ Amplifier for Android*
Intel® VTune™ Amplifier 2014 for Systems
Power & Performance profiling for Embedded and Mobile Devices

... Spending Time?

- Focus tuning on functions taking time

<table>
<thead>
<tr>
<th>Function</th>
<th>CPU Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm_2</td>
<td>3.560s</td>
</tr>
<tr>
<td>do_form</td>
<td>3.560s</td>
</tr>
<tr>
<td>algorithm_1</td>
<td>1.412s</td>
</tr>
<tr>
<td>BaseThreadInit</td>
<td>0.000s</td>
</tr>
</tbody>
</table>

... Wasting Time?

- See cache misses on your source

```
<table>
<thead>
<tr>
<th>Line</th>
<th>MEM_LOAD... LLC_MISS</th>
</tr>
</thead>
<tbody>
<tr>
<td>475 float rz, ry, rz =</td>
<td>any</td>
</tr>
<tr>
<td>476 float param1 = (rz &lt; 0)</td>
<td>30.000</td>
</tr>
<tr>
<td>477 float param2 = (rz &lt; 0)</td>
<td></td>
</tr>
<tr>
<td>478 bool neg = (rz &lt; 0)</td>
<td></td>
</tr>
</tbody>
</table>
```

... Waiting Too Long?

- See locks by wait time

```
<table>
<thead>
<tr>
<th>Wait Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
</tr>
<tr>
<td>Poor</td>
</tr>
<tr>
<td>Ok</td>
</tr>
<tr>
<td>Ideal</td>
</tr>
<tr>
<td>Wait Count</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wake-up Object/Call Stack</th>
<th>Total Wake-up Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Timer</td>
<td>1,763</td>
</tr>
<tr>
<td>Kernel Timer</td>
<td>1,353</td>
</tr>
<tr>
<td>IRQ 44 - i315</td>
<td>5,499</td>
</tr>
<tr>
<td>IRQ 12 - i8042</td>
<td>5,489</td>
</tr>
<tr>
<td></td>
<td>472</td>
</tr>
</tbody>
</table>

... Waking-up Too

- See wakeup causes on your source

- Linux & Windows host to Linux targets cross sampling
- Low overhead
- No special recompiles

Advanced profiling for power efficiency and scalable multicore performance
Intel® VTune™ Amplifier for Systems
Cross-platform Power & Performance Analysis

Remote Data Collection

Power and Performance Analysis
- Collects data on target device
- Analyze results on host system

Flexible data collection, configuration, and control

Low overhead sampling
No hardware instrumentation required
View results in source or assembly

Cross platform sampling of processor & SoC-wide events
Intel® VTune™ Amplifier 2014 for Systems
Power & Performance profiling for Embedded and Mobile Devices

Power Profiler
Find issues that affect power and energy consumption

Performance Profiler
Find performance bottlenecks

Java* JIT Profiler
Find performance issues in Java stack

Detect and help fix issues across all layers of the IA platform
Intel® VTune™ Amplifier 2014 for Systems
Analyzes Platform-Wide Power Consumption

- Displays processes for events and causes that wake-up the processor
- Correlates CPU, SoC components, and Linux/Android Wakelocks activities
- Analyzes effects of the interaction of different IP blocks with the SoC
- Comprehensive analysis coverage
  - Sleep State Analysis (C-state, S-State, D-State)
  - Frequency Analysis (P-State)
  - Analysis of User Wakelocks, Kernel Wakelocks, S0ix, D0ix states, and S3 (suspend-to-RAM) tracing
- Powerful filtering

Uniquely identify cause of wake-ups & provide timer call stacks
Analysis of Intel processor blocks that are not in the core
- Memory bandwidth for Intel® Core™ Processor
- Memory bandwidth and QPI bandwidth for Intel® Xeon™ Processor
- Cache Box support for both client and server parts
Supported OSs

Host:
- Red Hat Enterprise* Linux* 5, 6
- Ubuntu* 10.04 LTS, 12.04 LTS, 13.04
- Fedora* 17, 18
- Wind River* Linux* 4, 5
- openSUSE 12.1
- SUSE LINUX Enterprise Server* 11 SP2
- Microsoft* Windows* 7,8

Target:
- Yocto Project* 1.3, 1.4, and newer based environment
- CE Linux* PR32 based environment
- Tizen* IVI 1.0, 2.0
- Wind River* Linux* 4, 5 based environment
Performance profiling:
Intel® VTune™ Amplifier 2014 for Systems

Host

VTune GUI

Vtune result

VTune collector binary runs on target and stores result on target (local storage like card or NFS mounted)

Target device

amplxe-runss.py

control collection

SSH

amplxe-runss

transfer data/modules

SSH

Vtune result

driver

Data is opened in GUI and symbols are resolved using modules stored in result dir
User can specify search dir with separate debug files if needed

CLI interface for remote collection. Transfers data collected remotely back to host automatically together with application modules for symbol resolution

Simple python script (no remote collection in GUI)
Using SSH protocol for data transfers
Flexible collection configuration + control (pause/resume/stop)
Intel® VTune™ Amplifier 2014 for Systems

More Profiling Data
- CPU power and frequency
- Statistical call counts
- Hardware events + stacks
  Lower overhead, Higher resolution
  Finds hot spots in small functions
- Uncore event counting
  More accurate bandwidth analysis
- Ivy Bridge events
- Haswell events
  Updates as new processors ship

Easier To Use
- Source view for inlined code
  (For Intel® and GCC* compilers)
- Remote Collection
- Task annotation API
  Label and visualize tasks.
- User defined metrics
  Create meaningful metrics from events
- More/better advanced profiles
  (e.g., Bandwidth)

Activity in CPU

Easy to use, wealth of data, powerful analysis
Performance Tuning Methodology using VTune™ Amplifier 2014 for Systems

Use a top-down approach: system tuning, then algorithmic/application tuning, then micro-architectural tuning

General process for algorithm and micro-architectural tuning:

• Find hotspots

• Focus on top hotspot
  – Determine efficiency: Use Concurrency Analysis, Stalls/Uop Analysis, or Code examination
  – If inefficient, look for source of in-efficiency using Locks and Waits Analysis, Micro-Architectural metrics, or Code examination (If efficient, go to next hotspot)
  – Optimize if necessary

• Repeat!
Intel® VTune™ Amplifier 2014 for Systems – Hotspot Analysis
By drilling down to the source code level you can see line-by-line and instruction-by-instruction, where your application is spending its time.
In the general exploration viewpoint you can see if your application has exceeded the thresholds for our performance metrics. Metrics that exceed defined thresholds are colored in pink.
You can also see which functions in your program had the most of a particular event. (for example Branch Mispredict)
CPU Power Analysis
Intel® VTune™ Amplifier 2014 for Systems

To decrease CPU power usage minimize wake-ups

- Identify wake-up causes
  - Timers triggered by application
  - Interrupts mapped to HW intr level
  - Show wake-up rate
- Display source code for events that wake-up processor
- Show CPU frequencies by CPU core (CPU frequencies can change by CPU activity level)

Uniquely identifies the cause of wake-ups and give timer call stacks
Overview of power analysis

Idle vs. Active
• Do nothing efficiently
• Hurry up and get idle.
  e.g. Multi-threading (distributing work evenly across cores)

Optimize Sleep Behavior
• Minimize sporadic wakeups.
• Schedule all periodic activities from the app into same wakeup period.
• What is waking h/w from low power states? Why?

Optimize Utilization
• What is active? Why is it active?
• Minimize Polling Loops. Use event driven framework when possible.
• Turn devices off. Open devices can prevent the system from entering power saving state.
CPU C-States / P-States

- **P0** - CPU active at highest frequency (HFM)
- **Pn** - CPU active at lowest frequency (LFM)
- **C0** - CPU active (In any P-state)
- **C1** - Core clock is Off
- **C3/C4** - Reduced Voltage, Partial L2 cache flush
- **C6** - Core Off, L2 cache flush, state saved to SRAM

The deeper the sleep state:
- **more power saving**
- **but longer to wake up**

Power Higher  
Latency Greater
## CPU Sleep States

### Flexible C-States to Select Idle Power Level vs. Responsiveness

<table>
<thead>
<tr>
<th>Active state</th>
<th>C0</th>
<th>C1</th>
<th>C3</th>
<th>C4</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core voltage*</td>
<td><img src="image" alt="Core voltage" /></td>
<td><img src="image" alt="Core voltage" /></td>
<td><img src="image" alt="Core voltage" /></td>
<td><img src="image" alt="Core voltage" /></td>
<td><img src="image" alt="Core voltage" /></td>
</tr>
<tr>
<td>Core clock</td>
<td><img src="image" alt="Core clock" /></td>
<td><img src="image" alt="Core clock" /></td>
<td><img src="image" alt="Core clock" /></td>
<td><img src="image" alt="Core clock" /></td>
<td><img src="image" alt="Core clock" /></td>
</tr>
<tr>
<td>PLL</td>
<td><img src="image" alt="PLL" /></td>
<td><img src="image" alt="PLL" /></td>
<td><img src="image" alt="PLL" /></td>
<td><img src="image" alt="PLL" /></td>
<td><img src="image" alt="PLL" /></td>
</tr>
<tr>
<td>L1 caches</td>
<td><img src="image" alt="L1 caches" /></td>
<td><img src="image" alt="L1 caches" /></td>
<td><img src="image" alt="L1 caches" /></td>
<td><img src="image" alt="L1 caches" /></td>
<td><img src="image" alt="L1 caches" /></td>
</tr>
<tr>
<td>L2 cache</td>
<td><img src="image" alt="L2 cache" /></td>
<td><img src="image" alt="L2 cache" /></td>
<td><img src="image" alt="L2 cache" /></td>
<td><img src="image" alt="L2 cache" /></td>
<td><img src="image" alt="L2 cache" /></td>
</tr>
<tr>
<td>Wakeup time*</td>
<td>active</td>
<td><img src="image" alt="Wakeup time" /></td>
<td>partial flush</td>
<td><img src="image" alt="Wakeup time" /></td>
<td><img src="image" alt="Wakeup time" /></td>
</tr>
<tr>
<td>Idle power*</td>
<td><img src="image" alt="Idle power" /></td>
<td><img src="image" alt="Idle power" /></td>
<td><img src="image" alt="Idle power" /></td>
<td><img src="image" alt="Idle power" /></td>
<td><img src="image" alt="Idle power" /></td>
</tr>
</tbody>
</table>

* Rough approximation

*Other brands and names are the property of their respective owners.
Tracing C-States

VTune power driver does not cause wakeups by using kernel tracepoints to drive the collection of data.

CPU_IDLE tracepoint
Counter Reads (TSC, MPERF, C-State Residency MSRs)

C0
Sample (Cx+C0+Wakeup Cause)

Cx
Wakeup tracepoints
Intel® VTune™ Amplifier 2014 for Systems
Sleep states power analysis view
Intel® VTune™ Amplifier 2014 for Systems
Sleep states power analysis view
Small Increases in Processor Speed Results in Large Increases in Power

Processor Power and Processor Frequency

Power vs. Frequency Curve for Single Architecture
Intel® VTune™ Amplifier 2014 for Systems
Frequency states power analysis view
Intel® VTune™ Amplifier 2014 for Systems
Frequency states power analysis view
Analysis of Intel uncore blocks supported via SEP. Details:
- Memory bandwidth for Intel® Core™ Processor;
- Memory bandwidth, QPI bandwidth for Intel® Xeon™ Processor;
- Cache Box (Cbo) is supported for both client and server parts;
Product Installation of sep

• On target (whether you have built on target or host)
  • Load driver
    – ./insmod-sep3
• Once you have loaded the sep driver you need to source the environment to have access to sep.
  – source $SEP_INSTALL/bin/setup_sep_runtime_env.sh
Collecting performance data on the Yocto Project* using sep

1. Prepare target
   - Command line tool for collecting performance data.
   - Learn the installation requirements and setup device drivers.

2. Pick an event to sample and configure PMU
   - Cache misses, branch mis-predictions, Dependency/pipeline stalls

3. Start SEP sampling routine and application
   - Performance Monitoring Unit (PMU) periodically interrupts the processor
   - Time based sampling
   - Event based sampling
   - Both architectural and non-architectural processor events can be monitored using sampling and counting technologies
SEP command line example

```
sep -start -d 20 -ec "CPU_CLK_UNHALTED.CORE", "INST_RETIRED.ANY", "CPU_CLK_UNHALTED.REF", "DATA_TLB_MISSES.DTLB_MISS", "MEM_LOAD_RETIRED.L2_MISS" -out my_data
```

With this run of sep:

- `-d 20` specified a run of 20 seconds
- `-ec` specifies the events to be collected.
- `-out` specifies the name, note a suffix of `.tb6` will be used.
- For a list of supported events:
  - `sep -el`
Intel® VTune™ Amplifier 2014 for Systems
General exploration analysis types

Through extensive analysis Intel has determined a list of events and metrics that are often useful at providing initial data on applications.

In addition to providing useful metrics, it also provides built-in rules that will notify you when its thresholds have been exceeded.

VTune Amplifier XE has “General Exploration” analysis types built in for many of the “big core” processors. When run on an embedded system this data must be collected using sep. (see the following slide)
Running an Intel® Atom™ processor general exploration via sep

In order for Intel® VTune™ Amplifier 2014 for Systems to report Atom processor based metrics we need to specify a specific sequence of events. This event list is known as the “General Exploration” event list for the Intel® Atom™ processor.

sep -start -em -ec
"BR_INST_RETIRED.MISPRED.PS,BUS_LOCK_CLOCKS.ALL_AGENTS,CPU_CLK_UNHALTED.CORE,CPU_CLK_UNHALTED.REF,CYCLES_DIV_BUSY,DATA_TLB_MISSES.DTLB_MISS,EXT_SNOOP.ALL_AGENTS.HITM,FP_ASSIST.S,ICACHE.MISSES,INST_RETIRED.ANY,ITLB.MISSES,MACHINE_CLEARS.SMC,MEM_LOAD_RETIRED.L2_HIT.PS,MEM_LOAD_RETIRED.L2_MISS.PS,MISALIGN_MEM_REF.LD_SPLIT.AR,MISALIGN_MEM_REF.ST_SPLIT.AR,PAGE_WALKS.CYCLES,REISSUE.OVERLAP_STORE.AR,SIMD_ASSIST,UOPS.MS_CYCLES,UOPS_RETIRED.ANY" -app ./tachyon_find_hotspots
Importing SEP data into the Intel® VTune™ Amplifier 2014 for Systems GUI

1. Create new project
   • File->New->Project

2. Set Search directories for the project
   - Source
   - Symbols
   - Binaries


4. File->Import Result
   • “Import file.tb6” into project
NDA packages of Intel® VTune™ Amplifier 2014 for Systems

Prerequisites:

- VTune already needs to be installed.

Install the NDA add-on package over the installed product.

- On Windows:
  - Run Amplifier_XE_2013-update*_win_nda.msi

- On Linux:
  - uppack vtune_amplifier_xe_2013_update*_nda_tar.gz
  - Run install.sh from top-level folder
Memory Bandwidth Limitations

**Why:** Bandwidth bottlenecks increase the latency at which cache misses are being serviced

**How:** Bandwidth Profile

**What Now:**
- Compute the maximum theoretical bandwidth per socket for your platform in GB/s: \((<\text{MT/s} \times 8 \text{ Bytes/clock} \times <\text{num channels}>)/1000\)
- Run bandwidth analysis on your application. If total bandwidth per socket is > 75% of the maximum theoretical bandwidth, your application may be experiencing loaded (higher) latencies
- If appropriate, make system tuning adjustments (upgrading/balancing DIMMs, disabling HW prefetchers)
- Reduce bandwidth usage if possible: remove ineffective SW prefetches, make algorithmic changes to reduce data storage/sharing, reduce data updates, and balance memory access across sockets.
Intel® VTune™ Amplifier for Systems
Features for Android

Performance and Power Profiler brings 20+ years of technology to Android* devices based on Intel® architecture

Identify C/C++ hotspots on all Intel® architecture devices

Includes Intel® Energy Profiler
Finds Actionable wake-up, sleep state, frequency and temperature data which help find software that is causing unwanted power use.

Identify Hardware Bottlenecks (such as Cache Misses, Branch Mispredictions, etc)

Java Source/Dex or Assembly drill down to Dalvik JITted functions.
Summary

• Comprehensive software development tools solution set for embedded devices and intelligent systems

• Integrates into cross-build environments for Yocto Project*, Wind River* Linux*, and custom Linux*

• Covers all phases of development

• Powerful open source debug enhancements through GDB and SVEN

• Power Analysis, Performance Analysis, Thread Checking & Memory Checking

For more information, to evaluate, or purchase: http://intel.ly/system-studio
Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY
ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS
DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR
IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES
RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY
PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on
Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using
specific computer systems, components, software, operations and functions. Any change to any of
those factors may cause the results to vary. You should consult other information and performance
tests to assist you in fully evaluating your contemplated purchases, including the performance of that
product when combined with other products.

Copyright © , Intel Corporation. All rights reserved. Intel, the Intel logo, Atom, Core, Xeon, Cilk and
VTune are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for
optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and
SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or
effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-
dependent optimizations in this product are intended for use with Intel microprocessors. Certain
optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer
to the applicable product User and Reference Guides for more information regarding the specific
instruction sets covered by this notice.