Profiling in Xen

J. Renato Santos
G. (John) Janakiraman
Yoshio Turner
Aravind Menon

HP Labs

Xen Summit         September 7-8, 2006
Outline

- OProfile introduction
- XenOprofile overview
- Current status
- Further work
- Tutorial
- Profile code execution (application & kernel)

- For selected hardware events, such as:
  a) **unhalted clock cycles**: (most common use)
     - % of time spent on each user/kernel function
  b) **instructions**:
     - % of instructions executed by each user/kernel function
  c) **l2 cache misses**:
     - % of L2 misses in each user/kernel function

- **Statistical profiling**: continually sample currently executing code at every N hardware events. Large set of samples approximates the real hardware event distribution.
CPU: P4 / Xeon, speed 2794.74 MHz (estimated)
Counted GLOBAL_POWER_EVENTS events (time during which processor is not stopped) with a unit mask of 0x01 (mandatory) count 1000000

<table>
<thead>
<tr>
<th>samples</th>
<th>%</th>
<th>app name</th>
<th>symbol name</th>
</tr>
</thead>
<tbody>
<tr>
<td>9652</td>
<td>60.7694</td>
<td>libc-2.3.6.so</td>
<td>fputs_unlocked</td>
</tr>
<tr>
<td>3368</td>
<td>21.2051</td>
<td>libc-2.3.6.so</td>
<td>_IO_file_xsputn@@GLIBC_2.1</td>
</tr>
<tr>
<td>1829</td>
<td>11.5155</td>
<td>yes</td>
<td>(no symbols)</td>
</tr>
<tr>
<td>448</td>
<td>2.8206</td>
<td>libc-2.3.6.so</td>
<td>__i686.get_pc_thunk.bx</td>
</tr>
<tr>
<td>85</td>
<td>0.5352</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>get_offset_pmtmr</td>
</tr>
<tr>
<td>51</td>
<td>0.3211</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>delay_pmtmr</td>
</tr>
<tr>
<td>37</td>
<td>0.2330</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>mark_offset_pmtmr</td>
</tr>
<tr>
<td>31</td>
<td>0.1952</td>
<td>bash</td>
<td>(no symbols)</td>
</tr>
<tr>
<td>28</td>
<td>0.1763</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>timer_interrupt</td>
</tr>
<tr>
<td>25</td>
<td>0.1574</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>page_fault</td>
</tr>
<tr>
<td>23</td>
<td>0.1448</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>default_idle</td>
</tr>
<tr>
<td>21</td>
<td>0.1322</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>e1000_update_stats</td>
</tr>
<tr>
<td>15</td>
<td>0.0944</td>
<td>libc-2.3.6.so</td>
<td>__gconv_transform_utf8_internal</td>
</tr>
<tr>
<td>12</td>
<td>0.0756</td>
<td>libc-2.3.6.so</td>
<td>mbtowc</td>
</tr>
<tr>
<td>11</td>
<td>0.0693</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>ide_inb</td>
</tr>
<tr>
<td>11</td>
<td>0.0693</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>sysenter_past_esp</td>
</tr>
<tr>
<td>9</td>
<td>0.0567</td>
<td>libcrypto.so.0.9.7f</td>
<td>(no symbols)</td>
</tr>
<tr>
<td>6</td>
<td>0.0378</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>do_wp_page</td>
</tr>
<tr>
<td>5</td>
<td>0.0315</td>
<td>libc-2.3.6.so</td>
<td>__int_malloc</td>
</tr>
<tr>
<td>5</td>
<td>0.0315</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>apic_timer_interrupt</td>
</tr>
<tr>
<td>5</td>
<td>0.0315</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>irq_entries_start</td>
</tr>
<tr>
<td>4</td>
<td>0.0252</td>
<td>libc-2.3.6.so</td>
<td>__IO_default_xsputn</td>
</tr>
<tr>
<td>4</td>
<td>0.0252</td>
<td>libc-2.3.6.so</td>
<td>__IO_file_overflow@@GLIBC_2.1</td>
</tr>
<tr>
<td>4</td>
<td>0.0252</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>__copy_to_user_ll</td>
</tr>
<tr>
<td>4</td>
<td>0.0252</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>__d_lookup</td>
</tr>
<tr>
<td>4</td>
<td>0.0252</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>i8042_interrupt</td>
</tr>
<tr>
<td>4</td>
<td>0.0252</td>
<td>vmlinux-syms-2.6.16.13</td>
<td>vfs_write</td>
</tr>
<tr>
<td>3</td>
<td>0.0189</td>
<td>libc-2.3.6.so</td>
<td>__dl_mcount_wrapper_check</td>
</tr>
<tr>
<td>3</td>
<td>0.0189</td>
<td>libc-2.3.6.so</td>
<td>malloc</td>
</tr>
</tbody>
</table>
XenOprofile – System Profiling in Xen

• Extensions to Xen and OProfile for enabling profiling in Xen
• System wide profile:
  – Full profile across multiple domains
  – Covering code in user processes, kernel & Xen (including interrupts, etc.).
• Domain roles in a profiling session
  – Initiator (dom0): Coordinates the session
  – Active domain:
    • Run an instance of Oprofile
    • Process and store its own OProfile samples
  – Passive domain:
    • No OProfile instance running on domain
    • Its samples are processed by initiator (dom0)
Active Domain Profiling

• Complete detailed profile for each guest (xen, kernel & user)
• Requires domain coordination.
  – OProfile commands in dom0 and active domain must follow a given sequence
• Each domain generate its own profiling report.
Passive Domain Profiling

- Does not require OProfile in guest
  - Useful if using other OS’es, e.g. Windows
- Function level profiling only for Xen & kernel (Linux)
  - No detailed profile (functions) for user processes and kernel modules
- Easier to use (No domain coordination)
- Single aggregate OProfile report
Current Status

• Integrated in current unstable tree (first stable release: Xen 3.0.3)
• Current supported architectures:
  – X86 and X86-64 (both Intel and AMD cpu models)
• HVM guest support
  – Final stages of tests for AMD SVM (thanks to help from Tom Woller and Ray Bryant from AMD).
  – Need small fix on user level tools to deal with address overlap in Xen and kernel
• Passive domain support added (thanks to Xiaowei Yang, Intel)
• OProfile integration
  – First version accepted & merged into OProfile CVS tree
  – New version (for passive domain) needs some clean up before submission
  – Date for next OProfile release (including Xen support) is not determined yet
  – Xen patch for OProfile 0.9.1 available in http://xenoprof.sourceforge.net
Further Work (short term)

• Fix known limitations/bugs
  – Cannot profile domain in active mode after profiling it in passive mode and vice-versa.
  – Passive buffer flushed only on dom0 sample (possible overflow/unlikely)
  – Export statistics to user level (buffer overflow, total samples, etc.)
  – Identify source of anonymous samples for user level applications

• OProfile modifications cleanup and merge
  – Current representation of passive domain in sample files not ideal.
    • needs to create multiple symbolic links representing Xen/kernel samples
    • Better approach would use OProfile “–separate” feature.

• Support for other architectures
  – IA64, PPC. (simple for someone familiar with architecture - port from linux)

• Support for Intel VT
  – Need validation of fixes for SVM on VT.

• Support for call-trace profile
  – Work in progress by Intel.
Further Work (long term)

- Performance counter virtualization
  - Enable guests to run any tool that access performance counters (including OProfile) in guest context
  - Performance counter read/write is expensive (~1000 cycles)
    - Need some form of lazy save/restore on context switch

- Performance counter access from privileged domain
  - System wide instrumentation for other tools in addition to OProfile

- Physical CPU awareness in OProfile
  - OProfile can provide individual profiles for each CPU (VCPU in Xen)
  - Capability to separate profiles based on physical CPU may also be useful
  - Combining physical and virtual CPU profile could give insights into effects of Xen CPU scheduling.