Hardware Performance Monitoring: Sun's Perspective

Marty Itzkowitz
Project Lead,
Sun Studio™ Performance Tools
Introduction

- Users of HW Performance Monitoring
- HW Performance Monitoring Tools
  - Sun's Profiling Tools
    - Application and Kernel
  - Sun's Monitoring Tools
    - Application, System, and Kernel
  - HW Counter Libraries and APIs
- HW Counter Requirements
Users and their Objectives

• Users—various types
  ○ End users
  ○ End user developers
  ○ ISV developers
  ○ Compiler writers
  ○ Field engineers and system tuners
  ○ System SW developers
  ○ Chip designers

• Objectives for all:
  ○ What can I change to make things faster?
End User Tools

- Monitoring Tools
  - Both application and system
  - General overview of behavior
  - Curiosity, more than anything else:
    - End Users have few knobs to turn
    - Can change configurations:
      - Memory
      - Processors
      - Disk controllers
      - Domain partitioning
      - Application migration
End User Developer Tools

• Monitoring tools
  ○ Show a quick overview, system performance
  ○ Get data on what to profile

• Profiling Tools
  ○ Memory Performance: Cache, TLB
    ▪ vs. Functions/source-lines/instructions
    ▪ vs. Data types
    ▪ vs. Cachelines, memory buses, controllers, pages, ...
  ○ Floating-point Performance
  ○ Report data against user's programming model

• Library APIs
  ○ To instrument code regions
ISV Developer Tools

• Very much like end user developers
  ◦ But less likely to use library APIs
Compiler Writer Tools

- **Monitoring and profiling**
  - Compiler itself
  - Compiler-generated code
- **Generated instrumentation**
  - Using APIs
- **Feedback-directed optimization**
  - Memory profiling to inform better layout, striding
  - Branch-mispredict profiles
  - Other opportunities?
Field Engineer/System Tuner Tools

• Monitoring and profiling
  ○ Both user applications and system
  ○ Configuration management (like end users)
    ▪ Memory
    ▪ Processors
    ▪ Disk controllers
    ▪ Domain partitioning
System SW Developer Tools

- System-monitoring tools
  - Observability into OS
- Kernel profiling tools
- Runtime monitoring
  - Data to support page placement and migration
  - Understand scheduling
    - Particularly on CMT systems

- Issue: conflict with other uses of HW
  - System-level tools lock out user-level tools
Chip-designer Tools

• Not too much interest in current chips
  ○ Care about long-term development
  ○ Sometimes exploit tools on existing systems to inform next-generation designs
    ▪ Long design cycles make this problematic
      □ Leapfrog?
    ▪ Mostly use simulators, not old-generation chips
    ▪ Do use current HW monitoring to validate traces

• Unanswerable questions =>
  ○ Better HW monitoring in next chip to get answers
    ▪ Or so we hope
Sun's Profiling Tools

• Application profilers
  ○ Sun Studio—collect /Analyzer

• Kernel profilers
  ○ Sun Studio—er_kernel/Analyzer

• No multiplexing of HW counters
  ○ Hard to get valid statistics with multiplexing
  ○ Implies multiple runs
    ▪ If more counters of interest than counter registers
collect/analyzer

- collect -h <ctr>,<interval>, ...
  - As many counters as the HW allows
  - Automagic register assignment

- Memory counters
  - Prefix with + to record actual instruction, VA, PA
    - US-III,IV
      - Counter skid => backtracking to get data, which may fail
        - Newer chips will fix that

- Can record multiple experiments
  - Aggregate them for presentation
  - In lieu of multiplexing
Application Profiling–Analyzer
Memory Performance Profiling
Application Profiling—Analyzer
Disassembly Display

Performance Analyzer [test.1.er]

<table>
<thead>
<tr>
<th>Functions</th>
<th>Callers-Callees</th>
<th>Source</th>
<th>Lines</th>
<th>Disassembly</th>
<th>PCs</th>
<th>Data Layout</th>
<th>Data Objects</th>
<th>MemObj</th>
<th>Timeline</th>
<th>Leak List</th>
<th>Statistics</th>
<th>Experiments</th>
</tr>
</thead>
</table>

User CPU (sec): 0.0
DB and EB Stall Cycles (sec): 0.0
Sync Wait (sec): 0.0
Sync Wait Count: 0

Source File: /attest.c
Object File: /attest.o
Load Object: <attest>

0. 0. 0. 0. [1281] 145e4: ld [v02 + 208], v05
0. 0. 0. 0. [1281] 145e6: ld dd [v03 + 3], r18 (No type information)
0. 0. 0. 0. [1278] 145ec: ncp
0. 0. 0. 0. [1279] 145f0: ncp
0. 0. 0. 0. [1281] 145f4: <branch target>

6. 11.316 0. 0. [1281] 145f4: ld [v00], v10
{structure:workStruct_t -}.{float sum_ctr}
[1281] 145f6: inc $v5
[1281] 145fc: cmp $v5, $v1
[1281] 14600: fstsd $f0, $f4
[1281] 14604: fadd $f4, $f9, $f6
[1281] 14606: fbtos $f6, $v2
[1281] 1460c: st $f2, [v00]
{structure:workStruct_t -}.{float sum_ctr}
[1279] 14616: ncp
[1279] 14614: ncp
[1281] 14618: bl,pt %icc,0x145f4
[1281] 1461c: ncp
[1281] 14620: retl
[1281] 14624: ncp

1282. }
## Application Profiling—Analyzer
### Data Objects Display

### Performance Analyzer [mcf.1.er]

<table>
<thead>
<tr>
<th>Functions</th>
<th>Callers.Callees</th>
<th>Source</th>
<th>Lines</th>
<th>Disassembly</th>
<th>PCs</th>
<th>DataLayout</th>
<th>DataObjects</th>
<th>MemObj</th>
<th>Timeline</th>
</tr>
</thead>
</table>

### Summary

- **Data Object:** `{structure:arc -}

- **Scope:** (Global)

- **Type:** structure:arc

- **Member of:**
  - **Offset:** 54
  - **Size:** 8
  - **Elements:** 8

### Data for Selected Object:

- **Offset Size Name**
  - 0 8 `{structure:arc -}.point
  - 8 8 `{structure:arc -}.point
  - 16 8 `{structure:arc -}.point
  - 24 8 `{structure:arc -}.point

### Process Times (sec) / Counts

- **E$ Stall Cycles:** 169.701 (54.87%)
- **E$ Read Misses:** 952528575 (59.51%)

---

### Table

<table>
<thead>
<tr>
<th>Name</th>
<th>E$ Stall Cycles</th>
<th>E$ Read Misses</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Total&gt;</td>
<td>309.307 (100.00%)</td>
<td>1600644196 (100.00%)</td>
</tr>
<tr>
<td>169.701</td>
<td>54.87%</td>
<td>952528575 (59.51%)</td>
</tr>
<tr>
<td>133.257</td>
<td>43.08%</td>
<td>628018840 (39.24%)</td>
</tr>
<tr>
<td>79.401</td>
<td>25.67%</td>
<td>476014304 (29.79%)</td>
</tr>
<tr>
<td>57.323</td>
<td>21.77%</td>
<td>341310239 (21.32%)</td>
</tr>
<tr>
<td>57.212</td>
<td>21.73%</td>
<td>411612354 (25.73%)</td>
</tr>
<tr>
<td>22.645</td>
<td>9.58%</td>
<td>127203816 (7.95%)</td>
</tr>
<tr>
<td>20.167</td>
<td>8.52%</td>
<td>80302619 (5.45%)</td>
</tr>
<tr>
<td>13.611</td>
<td>4.40%</td>
<td>13700411 (0.66%)</td>
</tr>
<tr>
<td>8.345</td>
<td>2.70%</td>
<td>48601488 (3.10%)</td>
</tr>
<tr>
<td>6.207</td>
<td>2.03%</td>
<td>20006624 (1.50%)</td>
</tr>
<tr>
<td>6.207</td>
<td>2.02%</td>
<td>19096745 (1.19%)</td>
</tr>
<tr>
<td>5.494</td>
<td>1.76%</td>
<td>26007680 (1.62%)</td>
</tr>
<tr>
<td>3.456</td>
<td>1.12%</td>
<td>17400322 (1.09%)</td>
</tr>
<tr>
<td>3.133</td>
<td>1.01%</td>
<td>2110633 (1.32%)</td>
</tr>
<tr>
<td>1.367</td>
<td>0.44%</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>0.944</td>
<td>0.31%</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>0.576</td>
<td>0.19%</td>
<td>600018 (0.04%)</td>
</tr>
<tr>
<td>0.522</td>
<td>0.17%</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>
### Application Profiling-Analyzer

**Data Layout Display**

![Performance Analyzer Interface](image)

**Summary**

**Data Object:** `<Total>`
- **Scope:** Global
- **Type:** Synthetic

**Member of:**
- **Offset:**
- **Size:** 0
- **Elements:**

**Process Times (sec)/Counts**
- **E$ Read Misses:** 144039363 (100.00%)
- **E$ Stall Cycles:** 24.211 (100.00%)

**Data for Selected Object:**

<table>
<thead>
<tr>
<th>Read Misses</th>
<th>Stall Cycles (Sec)</th>
<th>Name + Offset</th>
<th>Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>144039363</td>
<td>24.211</td>
<td><code>* +offset .element</code></td>
<td><code>&lt;Total&gt;</code></td>
</tr>
</tbody>
</table>

**Functions**

- 143800000: `structure:foo` -
  - 300000: 0.042 `/+(pointer+structure:foo_next)`
  - 300000: 0.166 `+(pointer+structure:foo freight)`
  - 0: 0.037 `/+(pointer+structure:foo left)`

- 143200000: 23.674 `/+(int Code)`
  - 0: 0.056 `/+(int fval)`
  - 0: 0.042 `/+(int fleft)`
  - 0: 0.037 `/+(int fnet)`

- 239383: 0.158 `<Scalars>`
  - 0: 0.079 `{int iter}`

- 0.071: Backtracking traversed a branch target
Application Profiling–Analyzer
Memory Objects Display
Application Profiling—Analyzer
Floating-point Performance Display
Application Profiling—Analyzer
Timeline Display
Kernel Profiling–Analyzer

- Similar invocation to `collect`
  - Can use loadable driver (internal tool)
    - For clock- or HWC-profiling
  - Use DTrace (Solaris 10)
    - Clock-profiling now
    - HWC-profiling coming
- Experiment read with Analyzer
  - Same as user profiling
- Can record simultaneous user/kernel
Sun's Monitoring Tools

- Application monitoring tools
  - *cputrack*
  - *ripc2* (built on top of *cputrack*) (internal tool)
  - *bw* (internal tool)

- System monitoring tools
  - *cpustat*
  - *busstat*

- Allow/exploit multiplexing of HW counters
  - Valid statistics much more likely than for profiling
  - Use to see which counters to profile against
Application Monitoring—qutrack

qutrack output — periodic, multiplexed

U.1U3 111U1 1 tick 117422342 16629 # pic0=Cycle_cnt,pic1=DTLB_miss.sys
0.206 111U0 1 tick 59287055 29285156 # pic0=Instr_cnt,pic1=Re_DC_miss.sys
0.204 111U0 1 tick 544973 5320111 # pic0=Dispatch0_IC_miss,pic1=Re_SC_miss.sys
0.503 111U0 1 tick 259749 8 # pic0=Dispatch0_br_target,pic1=Re_DC_miss.sys
0.503 111U0 1 tick 97217 452879 # pic0=Dispatch0_2nd_br,pic1=Dispatch0_mispred.sys
0.603 111U0 1 tick 396249 2962149 # pic0=Dispatch_rs_mispred,pic1=Re_RAW_miss.sys
0.703 111U0 1 tick 79686301 5267 # pic0=Rstall_storeQ,pic1=Rstall_PP_use.sys
0.872 111U0 1 tick 3100459 18 # pic0=Rstall_UU_use,pic1=Re_FPU_bypass.sys
0.932 111U0 1 tick 67808560 176 # pic0=Cycle_cnt,pic1=DTLB_miss.sys
1.052 111U0 1 tick 148401091 2241460 # pic0=Instr_cnt,pic1=Re_DC_miss.sys
1.112 111U0 1 tick 62640 131944 # pic0=Dispatch0_IC_miss,pic1=Re_DC_miss.sys
1.122 111U0 1 tick 359 8 # pic0=Dispatch0_br_target,pic1=Re_DC_miss.sys
1.351 111U0 1 tick 180 2969 # pic0=Dispatch0_2nd_br,pic1=Dispatch0_mispred.sys
1.412 111U0 1 tick 633 89665 # pic0=Dispatch_rs_mispred,pic1=Re_RAW_miss.sys
1.531 111U0 1 tick 297442 35764303 # pic0=Rstall_storeQ,pic1=Rstall_PP_use.sys
1.612 111U0 1 tick 1350398 8 # pic0=Rstall_UU_use,pic1=Re_FPU_bypass.sys
1.712 111U0 1 tick 116030624 20 # pic0=Cycle_cnt,pic1=DTLB_miss.sys
1.832 111U0 1 tick 150250502 7114364 # pic0=Instr_cnt,pic1=Re_DC_miss.sys
1.922 111U0 1 tick 44256 3965119 # pic0=Dispatch0_IC_miss,pic1=Re_SC_miss.sys
2.032 111U0 1 tick 16478 4 # pic0=Dispatch0_br_target,pic1=Re_FPU_bypass.sys
2.132 111U0 1 tick 3102 32512 # pic0=Dispatch0_2nd_br,pic1=Dispatch0_mispred.sys
2.212 111U0 1 tick 1843 3160 # pic0=Dispatch_rs_mispred,pic1=Re_RAW_miss.sys
2.342 111U0 1 tick 15692 1659570 # pic0=Rstall_storeQ,pic1=Rstall_PP_use.sys
2.492 111U0 1 tick 129468 4 # pic0=Rstall_UU_use,pic1=Re_FPU_bypass.sys
2.522 111U0 1 tick 32731512 43 # pic0=Cycle_cnt,pic1=DTLB_miss.sys
2.612 111U0 1 tick 134166672 26865067 # pic0=Instr_cnt,pic1=Re_DC_miss.sys
2.732 111U0 1 tick 75839 4082663 # pic0=Dispatch0_IC_miss,pic1=Re_SC_miss.sys
2.822 111U0 1 tick 14306 6 # pic0=Dispatch0_br_target,pic1=Re_FPU_bypass.sys
2.912 111U0 1 tick 2664 29214 # pic0=Dispatch0_2nd_br,pic1=Dispatch0_mispred.sys
3.062 111U0 1 tick 11692 16148 # pic0=Dispatch_rs_mispred,pic1=Re_RAW_miss.sys
3.122 111U0 1 tick 10685 733062 # pic0=Rstall_storeQ,pic1=Rstall_PP_use.sys
3.212 111U0 1 tick 102127 8 # pic0=Rstall_UU_use,pic1=Re_FPU_bypass.sys
3.342 111U0 1 tick 152910322 66 # pic0=Cycle_cnt,pic1=DTLB_miss.sys
3.452 111U0 1 tick 166866401 31695212 # pic0=Instr_cnt,pic1=Re_DC_miss.sys
3.512 111U0 1 tick 49366 2928255 # pic0=Dispatch0_IC_miss,pic1=Re_SC_miss.sys
3.662 111U0 1 tick 20517 6 # pic0=Dispatch0_br_target,pic1=Re_FPU_bypass.sys
3.722 111U0 1 tick 1930 19260 # pic0=Dispatch0_2nd_br,pic1=Dispatch0_mispred.sys
3.912 111U0 1 tick 17800 21546 # pic0=Dispatch_rs_mispred,pic1=Re_RAW_miss.sys
4.022 111U0 1 tick 122909 0 # pic0=Rstall_UU_use,pic1=Re_FPU_bypass.sys
4.132 111U0 1 tick 128806410 53 # pic0=Cycle_cnt,pic1=DTLB_miss.sys
Application Monitoring—rip2

Aggregated output

Application stall information

<table>
<thead>
<tr>
<th>Processor Event</th>
<th>Ticks</th>
<th>Sec</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG_IC_miss</td>
<td>2933194905</td>
<td>2.782</td>
<td>0.3%</td>
</tr>
<tr>
<td>DG_br_targ_calc</td>
<td>700867035</td>
<td>0.673</td>
<td>0.1%</td>
</tr>
<tr>
<td>DG_snd_br</td>
<td>9375570</td>
<td>0.009</td>
<td>0.0%</td>
</tr>
<tr>
<td>DG_nispred</td>
<td>42219615</td>
<td>0.400</td>
<td>0.0%</td>
</tr>
<tr>
<td>D_rS_nispred</td>
<td>24962760</td>
<td>0.024</td>
<td>0.0%</td>
</tr>
<tr>
<td>RS_storeQ</td>
<td>3329449805</td>
<td>79.015</td>
<td>9.1%</td>
</tr>
<tr>
<td>Re_FP_use</td>
<td>5796651065</td>
<td>54.905</td>
<td>6.3%</td>
</tr>
<tr>
<td>Re_IJ_use</td>
<td>9970661385</td>
<td>9.453</td>
<td>1.1%</td>
</tr>
<tr>
<td>Re_FP_hypero</td>
<td>2220</td>
<td>0.000</td>
<td>0.0%</td>
</tr>
<tr>
<td>Re_MAV_miss</td>
<td>2586694685</td>
<td>2.549</td>
<td>0.3%</td>
</tr>
<tr>
<td>Re_DC_miss</td>
<td>46406630820</td>
<td>440.219</td>
<td>50.6%</td>
</tr>
<tr>
<td>Re_EC_miss</td>
<td>14224687395</td>
<td>135.223</td>
<td>15.5% (in DC miss)</td>
</tr>
<tr>
<td>Re_FRP_miss</td>
<td>4470</td>
<td>0.000</td>
<td>0.0%</td>
</tr>
<tr>
<td>DTLB_miss</td>
<td>74345460</td>
<td>5.995</td>
<td>0.7%</td>
</tr>
<tr>
<td>total</td>
<td>62830031565</td>
<td>596.018</td>
<td>68.5%</td>
</tr>
</tbody>
</table>

Stall times

Floating point traps

Events per instruction

Measurements of which processor events contributed to stall time.
# cpustat  -c pic0=Instr_cnt,pic1=Re_DC_miss,sys -c pic0=Dispatch0_IC_miss,pic1=Re_EC

<table>
<thead>
<tr>
<th>time</th>
<th>cpu</th>
<th>event</th>
<th>pIc0</th>
<th>pIc1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.009</td>
<td>1</td>
<td>tick 2059035807</td>
<td>33974649</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>5.010</td>
<td>0</td>
<td>tick 1551600712</td>
<td>22778438</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>10.009</td>
<td>0</td>
<td>tick 457169390</td>
<td>37188376</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>10.010</td>
<td>1</td>
<td>tick 328284013</td>
<td>41892660</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>15.009</td>
<td>1</td>
<td>tick 1451190423</td>
<td>157834999</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>15.010</td>
<td>0</td>
<td>tick 1462169361</td>
<td>341406198</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>20.009</td>
<td>1</td>
<td>tick 570276705</td>
<td>226545705</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>20.010</td>
<td>0</td>
<td>tick 586545226</td>
<td>126403413</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>25.009</td>
<td>0</td>
<td>tick 1279091800</td>
<td>122489122</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>25.010</td>
<td>1</td>
<td>tick 1850824519</td>
<td>36369893</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>30.009</td>
<td>1</td>
<td>tick 493026173</td>
<td>128362677</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>30.010</td>
<td>0</td>
<td>tick 586294847</td>
<td>81941837</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>35.009</td>
<td>0</td>
<td>tick 1946205173</td>
<td>229090921</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>35.010</td>
<td>1</td>
<td>tick 1351275940</td>
<td>175492857</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>40.009</td>
<td>1</td>
<td>tick 555076554</td>
<td>74295003</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>40.010</td>
<td>0</td>
<td>tick 570543507</td>
<td>101949954</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>45.009</td>
<td>0</td>
<td>tick 1681264848</td>
<td>286191421</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>45.010</td>
<td>1</td>
<td>tick 1318075599</td>
<td>133867092</td>
<td># pic0=Instr_cnt,pic1=Re_DC_miss,sys</td>
</tr>
<tr>
<td>50.009</td>
<td>0</td>
<td>tick 577788523</td>
<td>111924098</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
<tr>
<td>50.010</td>
<td>1</td>
<td>tick 646882743</td>
<td>155645759</td>
<td># pic0=Dispatch0_IC_miss,pic1=Re_EC_miss,sys</td>
</tr>
</tbody>
</table>
# System Monitoring -- cpustat

System-wide version of cputrack

- Memory counters

```bash
# cpustat -c MC_reads_0,MC Writes_0,sys -c MC_reads_1,MC Writes_1,sys -c MC_reads

<table>
<thead>
<tr>
<th>time</th>
<th>cpu</th>
<th>event</th>
<th>pic0</th>
<th>pic1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.010</td>
<td>0</td>
<td>tick</td>
<td>379284</td>
<td>435924</td>
</tr>
<tr>
<td>5.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10.010</td>
<td>0</td>
<td>tick</td>
<td>83775</td>
<td>2800104</td>
</tr>
<tr>
<td>15.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>15.010</td>
<td>0</td>
<td>tick</td>
<td>416247</td>
<td>480723</td>
</tr>
<tr>
<td>20.010</td>
<td>0</td>
<td>tick</td>
<td>179454</td>
<td>2909166</td>
</tr>
<tr>
<td>20.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>25.010</td>
<td>0</td>
<td>tick</td>
<td>2157225</td>
<td>1011898</td>
</tr>
<tr>
<td>25.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>30.010</td>
<td>0</td>
<td>tick</td>
<td>385880</td>
<td>3055634</td>
</tr>
<tr>
<td>30.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>35.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>35.010</td>
<td>0</td>
<td>tick</td>
<td>304115</td>
<td>426714</td>
</tr>
<tr>
<td>40.010</td>
<td>0</td>
<td>tick</td>
<td>656979</td>
<td>3392755</td>
</tr>
<tr>
<td>40.011</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>45.010</td>
<td>0</td>
<td>tick</td>
<td>1668996</td>
<td>746562</td>
</tr>
<tr>
<td>45.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>50.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>50.010</td>
<td>0</td>
<td>tick</td>
<td>738173</td>
<td>3366560</td>
</tr>
<tr>
<td>55.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>55.010</td>
<td>0</td>
<td>tick</td>
<td>1449829</td>
<td>728740</td>
</tr>
<tr>
<td>60.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>60.012</td>
<td>0</td>
<td>tick</td>
<td>891850</td>
<td>3072477</td>
</tr>
<tr>
<td>65.010</td>
<td>1</td>
<td>tick</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>65.010</td>
<td>0</td>
<td>tick</td>
<td>1042867</td>
<td>575276</td>
</tr>
</tbody>
</table>
```

Marty Itzkowitz, Sun Microsystems  
3/1/05 10:15:26 am
Application Monitoring—bw

Show memory bandwidth of application

• Based on cpustat
  ○ Read memory counters, massage output

nubbins% bw <target>

Read memory bandwidth: 1023.58642578125 MB/sec (total bytes = 17172930560)
Write memory bandwidth: 998.464149475098 MB/sec (total bytes = 16751448704)
Total memory bandwidth: 2022.05057525635 MB/sec (total bytes = 33924379264)
Elapsed time : 16 secs
## System Monitoring - `busstat`

### Bus performance

```
# busstat -w pcis0

<table>
<thead>
<tr>
<th>time</th>
<th>dev</th>
<th>event0</th>
<th>pic0</th>
<th>event1</th>
<th>pic1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>200</td>
<td>dvma_stream_rd</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>98</td>
<td>dvma_stream_rd</td>
<td>98</td>
</tr>
<tr>
<td>3</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>197</td>
<td>dvma_stream_rd</td>
<td>197</td>
</tr>
<tr>
<td>4</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>93</td>
<td>dvma_stream_rd</td>
<td>93</td>
</tr>
<tr>
<td>5</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>90</td>
<td>dvma_stream_rd</td>
<td>90</td>
</tr>
<tr>
<td>6</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>89</td>
<td>dvma_stream_rd</td>
<td>89</td>
</tr>
<tr>
<td>7</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>312</td>
<td>dvma_stream_rd</td>
<td>312</td>
</tr>
<tr>
<td>8</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>92</td>
<td>dvma_stream_rd</td>
<td>92</td>
</tr>
<tr>
<td>9</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>100</td>
<td>dvma_stream_rd</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>9204</td>
<td>dvma_stream_rd</td>
<td>9204</td>
</tr>
<tr>
<td>11</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>2688</td>
<td>dvma_stream_rd</td>
<td>2688</td>
</tr>
<tr>
<td>12</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>22237</td>
<td>dvma_stream_rd</td>
<td>22237</td>
</tr>
<tr>
<td>13</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>98</td>
<td>dvma_stream_rd</td>
<td>98</td>
</tr>
<tr>
<td>14</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>111</td>
<td>dvma_stream_rd</td>
<td>111</td>
</tr>
<tr>
<td>15</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>116</td>
<td>dvma_stream_rd</td>
<td>116</td>
</tr>
<tr>
<td>16</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>104</td>
<td>dvma_stream_rd</td>
<td>104</td>
</tr>
<tr>
<td>17</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>515</td>
<td>dvma_stream_rd</td>
<td>515</td>
</tr>
<tr>
<td>18</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>1330</td>
<td>dvma_stream_rd</td>
<td>1330</td>
</tr>
<tr>
<td>19</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>1252</td>
<td>dvma_stream_rd</td>
<td>1252</td>
</tr>
<tr>
<td>20</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>1172</td>
<td>dvma_stream_rd</td>
<td>1172</td>
</tr>
<tr>
<td>21</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>189</td>
<td>dvma_stream_rd</td>
<td>189</td>
</tr>
<tr>
<td>22</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>100</td>
<td>dvma_stream_rd</td>
<td>100</td>
</tr>
<tr>
<td>23</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>90</td>
<td>dvma_stream_rd</td>
<td>90</td>
</tr>
<tr>
<td>24</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>108</td>
<td>dvma_stream_rd</td>
<td>108</td>
</tr>
<tr>
<td>25</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>100</td>
<td>dvma_stream_rd</td>
<td>100</td>
</tr>
<tr>
<td>26</td>
<td>pcis0</td>
<td>dvma_stream_rd</td>
<td>989</td>
<td>dvma_stream_rd</td>
<td>989</td>
</tr>
</tbody>
</table>
```
HW Counter Library APIs

• Typical use by users
  ○ Read before and after block of code
  ○ Compute deltas

• API needs
  ○ Select counters, preset and read values
    ▪ System call access
    ▪ Direct user-mode reading?
      ▫ Should be lower overhead than system call

• Profiling APIs
  ○ Specify counters, intervals for each
  ○ Usually for tool developers, not end users
HW-Counter Requirements, I

• Documented behavior
  ○ Actual behavior, not design intent or schematics
  ○ *Verified behavior!*
    ▪ If they're not verified, they don't work

• Counters for all critical resources
  ○ Interesting events
  ○ Stall-cycle counts, not just event counts
    ▪ Measures the *cost* of events

• Wide counter registers
  ○ Minimize need for OS to “widen” them to 64-bits

• As many registers as possible
HW-Counter Requirements, II

• Counter contexts
  ○ Counters virtualizable to user context
    ▪ *i.e.*, Kernel save and restore on context switch
    ▪ Overflow attributable to user context
  ○ System-wide – for system monitoring

• CMT Issues
  ○ Independent counter sets for each HW thread
  ○ Counters for resource conflict among threads
  ○ Measure number of active threads (binning?)

• Wide-instruction chips
  ○ Counters for how many instructions in each cycle
HW-Counter Requirements, III

- Counter interface – kernel driver
  - Read and write entire width of counter
  - Specify arbitrary interval for profiling
  - Specify arbitrary method for sampling
HW-Counter Requirements, IV

• Profiling-interrupt infrastructure
  ○ Easily attributable to specific counter
  ○ Delivered as precisely as feasible
    ▪ *i.e.*, interrupt is synchronous to the context triggering it
    ▪ If interrupt is deferred, allow SW to wait deterministically
  ○ Delivered with VA/PA (memory-related counters)
    ▪ User needs to know which data objects are getting misses

• Instruction-sampling infrastructure
  ○ *c.f.*, Profile-me
  ○ Detailed information on sampled instruction
HW-Counter Constraints, I

• Everything is a tradeoff ...
  ○ Complexity, alas, is somewhere
    ▪ SW wants it in HW
    ▪ HW wants in in SW

• Area constraints
  ○ Lead to pressure to reduce number of counters
    ▪ But SW wants infinite number (approximately)
  ○ Lead to pressure to reduce width of counters
    ▪ But SW prefers to never deal with rollover
      □ But still need comparator to trigger profile interrupt
  ○ Are especially problematic for CMT
HW-Counter Constraints, II

• Routing constraints
  ○ Lead to delays due to long wires
  ○ Imply difficulty with counter placement
  ○ May make precise, synchronous delivery difficult

• Complex pipelines
  ○ Make precise, synchronous delivery difficult

• Mapping events to registers
  ○ SW wants any event to map to any register
Acknowledgements

(Alphabetic)

Russ Blaine, SW, Solaris
Darryl Gove, SW, CPE
Adam Talcott, HW, SSG
Mario Wolczko, Sun Labs
Marty Itzkowitz
marty.itzkowitz@sun.com