Hardware Performance Monitoring
with PAPI

Dan Terpstra
terpstra@cs.utk.edu

CScADS Autotuning Workshop
July 2007
What’s PAPI?

♦ Middleware that provides a consistent programming interface for the performance counter hardware found in most major micro-processors.

♦ Countable events are defined in two ways:
  - Platform-neutral Preset Events
  - Platform-dependent Native Events

♦ Preset Events can be derived from multiple Native Events

♦ All events are referenced by name and collected into EventSets for sampling

♦ Events can be multiplexed if counters are limited

♦ Statistical sampling is implemented by:
  - Software overflow with timer driven sampling
  - Hardware overflow if supported by the platform
Where’s PAPI

- PAPI runs on most modern processors and Operating Systems of interest to HPC:
  - IBM POWER{3, 4, 5} / AIX
  - POWER{4, 5, 6} / Linux
  - PowerPC{-32, -64, 970} / Linux
  - Blue Gene / L
  - Intel Pentium II, III, 4, M, Core, etc. / Linux
  - Intel Itanium{1, 2, Montecito?}
  - AMD Athlon, Opteron / Linux
  - Cray T3E, X1, XD3, XT{3, 4} Catamount
  - Altix, Sparc, SiCortex...
  - ...and even Windows!
  - ...but not Mac 😞
History of PAPI

♦ http://icl.cs.utk.edu/papi/

♦ Started as a Parallel Tools Consortium project in 1998

♦ Goal:

“Produce a specification for a portable interface to the hardware performance counters available on most modern microprocessors”.

CScADS Autotuning
Figure 2: Timeline of releases for each tool represented in the project. The vertical dashed lines indicate SC conference dates where the tools are regularly demonstrated. TAU’s v1.0 release occurred at SC’97.

SDCI HPC Improvement:
High-Productivity Performance Engineering
(Tools, Methods, Training) for the NSF HPC Applications
Submitted January 22, 2007

Allen D. Malony, Sameer Shende, Shirley Moore, Nicholas Nystrom, Rick Kufrin

CScADS Autotuning
Some Tools that use PAPI

♦ TAU (U Oregon)  http://www.cs.uoregon.edu/research/tau/
♦ HPCToolkit (Rice Univ)  http://hipersoft.cs.rice.edu/hpctoolkit/
♦ KOJAK (UTK, FZ Juelich)  http://icl.cs.utk.edu/kojak/
♦ PerfSuite (NCSA)  http://perfsuite.ncsa.uiuc.edu/
♦ Titanium (UC Berkeley)  http://www.cs.berkeley.edu/Research/Projects/titanium/
♦ SCALEA (Thomas Fahringer, U Innsbruck)  http://www.par.univie.ac.at/project/scalea/
♦ Open\|Speedshop (SGI)  http://oss.sgi.com/projects/openspeedshop/
♦ SvPablo (UNC Renaissance Computing Institute)  http://www.renci.unc.edu/Software/Pablo/pablo.htm

CScADS Autotuning
Two exposed interfaces to the underlying counter hardware:

1. The low level API manages hardware events in user defined groups called *EventSets*, and provides access to advanced features.

2. The high level API provides the ability to start, stop and read the counters for a specified list of events.
PAPI Hardware Events

♦ **Preset Events**
  - Standard set of over 100 events for application performance tuning
  - No standardization of the exact definition
  - Mapped to either single or linear combinations of native events on each platform
  - Use `papi_avail` utility to see what preset events are available on a given platform

♦ **Native Events**
  - Any event countable by the CPU
  - Same interface as for preset events
  - Use `papi_native_avail` utility to see all available native events

♦ **Use `papi_event_chooser` utility to select a compatible set of events**
Data and Instruction Range Qualification

- Generalized PAPI interface for data structure and instruction address range qualification
- Applied to the specific instance of the Itanium2
- Extended an existing PAPI call, `PAPI_set_opt()`, to specify starting and ending addresses of data structures or instructions to be instrumented

```c
option.addr.eventset = EventSet;
option.addr.start = (caddr_t)array;
option.addr.end = (caddr_t)(array + size_array);
retval = PAPI_set_opt(PAPI_DATA_ADDRESS, &option);
```

- An instruction range can be set using `PAPI_INSTR_ADDRESS`
- `papi_native_avail` was modified to list events that support data or instruction address range qualification.
PAPI Preset Events

- Of ~100 events, over half are cache related:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_L1_DCH</td>
<td>Level 1 data cache hits</td>
</tr>
<tr>
<td>PAPI_L1_DCA</td>
<td>Level 1 data cache accesses</td>
</tr>
<tr>
<td>PAPI_L1_DCR</td>
<td>Level 1 data cache reads</td>
</tr>
<tr>
<td>PAPI_L1_DCW</td>
<td>Level 1 data cache writes</td>
</tr>
<tr>
<td>PAPI_L1_DCM</td>
<td>Level 1 data cache misses</td>
</tr>
<tr>
<td>PAPI_L1_ICH</td>
<td>Level 1 instruction cache hits</td>
</tr>
<tr>
<td>PAPI_L1_ICA</td>
<td>Level 1 instruction cache accesses</td>
</tr>
<tr>
<td>PAPI_L1_ICR</td>
<td>Level 1 instruction cache reads</td>
</tr>
<tr>
<td>PAPI_L1_ICW</td>
<td>Level 1 instruction cache writes</td>
</tr>
<tr>
<td>PAPI_L1_ICM</td>
<td>Level 1 instruction cache misses</td>
</tr>
<tr>
<td>PAPI_L1_TCH</td>
<td>Level 1 total cache hits</td>
</tr>
<tr>
<td>PAPI_L1_TCA</td>
<td>Level 1 total cache accesses</td>
</tr>
<tr>
<td>PAPI_L1_TCR</td>
<td>Level 1 total cache reads</td>
</tr>
<tr>
<td>PAPI_L1_TCW</td>
<td>Level 1 total cache writes</td>
</tr>
<tr>
<td>PAPI_L1_TCM</td>
<td>Level 1 cache misses</td>
</tr>
<tr>
<td>PAPI_L1_LDM</td>
<td>Level 1 load misses</td>
</tr>
<tr>
<td>PAPI_L1_STM</td>
<td>Level 1 store misses</td>
</tr>
</tbody>
</table>

- Repeat for Levels 2 and 3...
Other cache and memory events:

**Shared cache**
- **PAPI_CA_SNP**: Requests for a snoop
- **PAPI_CA_SHR**: Requests for exclusive access to shared cache line
- **PAPI_CA_CLN**: Requests for exclusive access to clean cache line
- **PAPI_CA_INV**: Requests for cache line invalidation
- **PAPI_CA_ITV**: Requests for cache line intervention

**TLB**
- **PAPI_TLB_DM**: Data translation lookaside buffer misses
- **PAPI_TLB_IM**: Instruction translation lookaside buffer misses
- **PAPI_TLB_TL**: Total translation lookaside buffer misses
- **PAPI_TLB_SD**: Translation lookaside buffer shootdowns

**Resource Stalls**
- **PAPI_MEM_SCY**: Cycles Stalled Waiting for memory accesses
- **PAPI_MEM_RCY**: Cycles Stalled Waiting for memory Reads
- **PAPI_MEM_WCY**: Cycles Stalled Waiting for memory writes
- **PAPI_RES_STL**: Cycles stalled on any resource
- **PAPI_FP_STAL**: Cycles the FP unit(s) are stalled
### PAPI Preset Events (iii)

#### Program flow:

<table>
<thead>
<tr>
<th>Branches</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_BR_INS</td>
<td>Branch instructions</td>
</tr>
<tr>
<td>PAPI_BR_UCN</td>
<td>Unconditional branch instructions</td>
</tr>
<tr>
<td>PAPI_BR_CN</td>
<td>Conditional branch instructions</td>
</tr>
<tr>
<td>PAPI_BR_TKN</td>
<td>Conditional branch instructions taken</td>
</tr>
<tr>
<td>PAPI_BR_NTK</td>
<td>Conditional branch instructions not taken</td>
</tr>
<tr>
<td>PAPI_BR_MSP</td>
<td>Conditional branch instructions mispredicted</td>
</tr>
<tr>
<td>PAPI_BR_PRC</td>
<td>Conditional branch instructions correctly predicted</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Conditional Stores</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAPI_CSR_FAL</td>
<td>Failed store conditional instructions</td>
</tr>
<tr>
<td>PAPI_CSR_SUC</td>
<td>Successful store conditional instructions</td>
</tr>
<tr>
<td>PAPI_CSR_TOT</td>
<td>Total store conditional instructions</td>
</tr>
</tbody>
</table>
PAPI Preset Events (iv)

- **Timing, efficiency, pipeline:**

  - `PAPI_TOT_CYC`: Total cycles
  - `PAPI_TOT_IIS`: Instructions issued
  - `PAPI_TOT_INS`: Instructions completed
  - `PAPI_INT_INS`: Integer instructions completed
  - `PAPI_LST_INS`: Load/store instructions completed
  - `PAPI_SYC_INS`: Synchronization instructions completed
  - `PAPI_BRU_IDL`: Cycles branch units are idle
  - `PAPI_FXU_IDL`: Cycles integer units are idle
  - `PAPI_FPU_IDL`: Cycles floating point units are idle
  - `PAPI_LSU_IDL`: Cycles load/store units are idle
  - `PAPI_STL_ICY`: Cycles with no instruction issue
  - `PAPI_FUL_ICY`: Cycles with maximum instruction issue
  - `PAPI_STL_CCY`: Cycles with no instructions completed
  - `PAPI_FUL_CCY`: Cycles with maximum instructions completed
  - `PAPI_HW_INT`: Hardware interrupts
PAPI Preset Events (v)

♦ Floating point:

- PAPI_FP_INS: Floating point instructions
- PAPI_FP_OPS: Floating point operations
- PAPI_FML_INS: Floating point multiply instructions
- PAPI_FAD_INS: Floating point add instructions
- PAPI_FDV_INS: Floating point divide instructions
- PAPI_FSQ_INS: Floating point square root instructions
- PAPI_FNV_INS: Floating point inverse instructions
- PAPI_FMA_INS: FMA instructions completed
- PAPI_VEC_INS: Vector/SIMD instructions
What’s a Native Event?

**PMD: AMD Athlon, Opteron**

**PMC: Intel Pentium II, III, M, Core; AMD Athlon, Opteron**

**PMC: Pentium 4**
CScADS Autotuning

Intel Pentium Core: L2_ST

...{ .pme_name = "L2_ST",  .pme_code = 0x2a,  .pme_flags = PFMLIB_CORE_CSPEC,  .pme_desc = "L2 store requests",  .pme_numasks = 7  },
{ .pme_name = "MESI",  .pme_desc = "Any cacheline access",  .pme_umasks = {
  { .pme_uname = "MESI",  .pme_udesc = "Any cacheline access",  .pme_ucode = 0xf
  },
  { .pme_uname = "I_STATE",  .pme_udesc = "Invalid cacheline",  .pme_ucode = 0x1
  },
  { .pme_uname = "S_STATE",  .pme_udesc = "Shared cacheline",  .pme_ucode = 0x2
  },
  { .pme_uname = "E_STATE",  .pme_udesc = "Exclusive cacheline",  .pme_ucode = 0x4
  },
  { .pme_uname = "M_STATE",  .pme_udesc = "Modified cacheline",  .pme_ucode = 0x8
  }
},
{ .pme_name = "SELF",  .pme_desc = "This core",  .pme_ucode = 0x40
},
{ .pme_name = "BOTH_CORES",  .pme_desc = "Both cores",  .pme_ucode = 0xc0
}
}
{ .pme_numasks = 7
},
...

PRESET,
PAPI_L2_DCA,
DERIVED_ADD,
L2_LD:SELF:ANY:MESI,
L2_ST:SELF:MESI
How many counters does it take...
Performance Counters:
- 48 UPC Counters
  - shared by both CPUs
  - External to CPU cores
  - 32 bits :(
- 2 Counters on each FPU
  - 1 counts load/stores
  - 1 counts arithmetic operations
- Accessed via blg_perfctr
- 15 Preset Events
  - 10 PAPI presets
  - 5 Custom BG/L presets
- 328 native events available
Cell Broadband Engine

- Each Cell contains: 1 PPE and 8 SPEs.
  - ...and 1 PMU external to all of these.
  - 8 16-bit counters configurable as 4 32-bit counters.
  - 1024 slot 128-bit trace buffer
  - 400 native events

- Working with IBM engineers on
  - developing perfmon2 libpfm layer for Cell BE
  - Linux Cell BE kernel modifications
  - Porting PAPI-C (LANL grant)
Top500 Operating Systems

CScADS Autotuning
Written by Mikael Petterson
- Labor of love...
- First available: Fall 1999
- First PAPI use: Fall 2000

Supports:
- Intel Pentium II, III, 4, M, Core
- AMD K7 (Athlon), K8 (Opteron)
- IBM PowerPC 970, POWER4, POWER5
Perfctr Features

♦ **Patches the Linux kernel**
  - Saves perf counters on context switch
  - Virtualizes counters to 64-bits
  - Memory-maps counters for fast access
  - Supports counter overflow interrupts where available

♦ **User Space Library**
  - PAPI uses about a dozen calls
Perfctr Timeline

♦ Steady development
  ➢ 1999 – 2004

♦ Concerted effort for kernel inclusion
  ➢ May 2004 – May 2005

♦ Ported to Cray Catamount; Power Linux
  ➢ ~ 2005

♦ Maintenance only
  ➢ 2005 →
Perfmon

- Written by Stephane Eranian @ HP
- Originally Itanium only
  - Built-in to the Linux-ia64 kernel since 2.4.0
- System call interface
- libpfm helper library for bookkeeping
Perfmon2*

- Provides a generic interface to access PMU
  - Not dedicated to one app, avoid fragmentation
- Must be portable across all PMU models:
  - Almost all PMU-specific knowledge in user level libraries
- Supports per-thread monitoring
  - Self-monitoring, unmodified binaries, attach/detach
  - multi-threaded and multi-process workloads
- Supports system-wide monitoring
- Supports counting and sampling
- No modification to applications or system
- Built-in, efficient, robust, secure, simple, documented

* Slide contents courtesy Stephane Eranian
Perfmon2

- Setup done through external support library
- Uses a system call for counting operations
  - More flexibility, ties with ctxsw, exit, fork
  - Kernel compile-time option on Linux
- Perfmon2 context encapsulates all PMU state
  - Each context uniquely identified by file descriptor
- \texttt{int perfmonctl(int fd, int cmd, void *arg, int narg)}

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFM_CREATE_CONTEXT</td>
<td>PFM_READ_PMDS</td>
<td>PFM_START</td>
</tr>
<tr>
<td>PFM_WRITE_PMCS</td>
<td>PFM_LOAD_CONTEXT</td>
<td>PFM_STOP</td>
</tr>
<tr>
<td>PFM_WRITE_PMDS</td>
<td>PFM_UNLOAD_CONTEXT</td>
<td>PFM_RESTART</td>
</tr>
<tr>
<td>PFM_CREATE_EVTSET</td>
<td>PFM_DELETE_EVTSET</td>
<td>PFM_GETINFO_EVTSET</td>
</tr>
<tr>
<td>PFM_GETINFO_PMCS</td>
<td>PFM_GETINFO_PMDS</td>
<td></td>
</tr>
<tr>
<td>PFM_GET_CONFIG</td>
<td>PFM_SET_CONFIG</td>
<td></td>
</tr>
</tbody>
</table>

CScADS Autotuning
Perfmon2 Features

- **Support today for:**
  - Intel Itanium, P6, M, Core, Pentium4, AMD Opteron, IBM Power, MIPS, SiCortex
- **Full native event tables for supported processors**
- **Kernel based Multiplexing**
  - Event set chaining
- **Kernel based Sampling/Overflow**
  - Time or event based
  - Custom sampling buffers
Next Steps

♦ Kernel integration
  ➢ ‘Final’ integration testing underway
  ➢ Possible inclusion in 2.6.22 kernel
♦ Implemented by Cray in CNK, X2
♦ Cell BE
  ➢ Port with IBM engineers is underway
♦ Leverage libpfm for PAPI native events
  ➢ Migration completed for P6, Core, P4, Opteron
♦ PAPI testing on perfmon2 patched kernels
  ➢ Opteron currently being tested
  ➢ Woodcrest/Clovertown testing planned
Component PAPI (PAPI-C)

 Goals:

- Support simultaneous access to on- and off-processor counters
- Isolate hardware dependent code in a separable 'component' module
- Extend platform independent framework code to support multiple simultaneous components
- Add or modify API calls to support access to any of several components
- Modify build environment for easy selection and configuration of multiple available components

Will be released (RSN*) as PAPI 4.0
Current PAPI Design

PAPI Framework Layer

Low Level API

High Level API

PAPI Component Layer

Kernel Patch

Operating System

Perf Counter Hardware

Portable

Platform Dependent

CScADS Autotuning
Component PAPI Design

PAPI Framework Layer

- Low Level API
- Hi Level API
- Devel API

PAPI Component Layer (network)
- Kernel Patch
- Operating System
- Perf Counter Hardware

PAPI Component Layer (CPU)
- Kernel Patch
- Operating System
- Perf Counter Hardware

PAPI Component Layer (thermal)
- Kernel Patch
- Operating System
- Perf Counter Hardware

CScADS Autotuning
PAPI-C Status

♦ PAPI 3.9 pre-release available with documentation
♦ Implemented Myrinet substrate (native counters)
♦ Implemented ACPI temperature sensor substrate
♦ Working on Infiniband and Cray Seastar substrates (access to Seastar counters not available under Catamount but expected under CNL)
♦ Asked by Cray engineers for input on desired metrics for next network switch
♦ Tested on HPC Challenge benchmarks
♦ Tested platforms include Pentium III, Pentium 4, Core2Duo, Itanium (I and II) and AMD Opteron
PAPI-C New Routines

- `PAPI_get_component_info()`
- `PAPI_num_cmp_hwctrs()`
- `PAPI_get_cmp_opt()`
- `PAPI_set_cmp_opt()`
- `PAPI_set_cmp_domain()`
- `PAPI_set_cmp_granularity()`
PAPI-C Building and Linking

- CPU components are automatically detected by *configure* and included in the build
- CPU component assumed to be present and always configured as component 0
- To include additional components, use configure option
  ```
  --with-<cmp> = yes
  ```
- Currently supported components
  - `with-acpi = yes`
  - `with-mx = yes`
  - `with-net = yes`
- The make process compiles and links sources for all requested components into a single library
## Myrinet MX Counters

<table>
<thead>
<tr>
<th>Category</th>
<th>Counters</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>LANAI_UPTIME, COUNTERS_UPTIME, BAD_CRC8, BAD_CRC32, UNSTRIPPED_ROUTE, PKT_DESC_INVALID, RECV_PKT_ERRORS, PKT_MISROUTED, DATA_SRC_UNKNOWN, DATA_BAD_ENDPT, DATA_ENDPT_CLOSED, DATA_BAD_SESSION, PUSH_BAD_WINDOW, PUSH_DUPLICATE, PUSH_OBSOLETE, PUSH_RACE_DRIVER, PUSH_BAD_SEND_HANDLE_MAGIC, PUSH_BAD_SRC_MAGIC, PULL_OBSOLETE, PULL_NOTIFY_OBSOLETE, PULL_RACE_DRIVER, ACK_BAD_TYPE, ACK_BAD_MAGIC, ACK_RESEND_RACE, LATE_ACK</td>
</tr>
<tr>
<td>acks</td>
<td>ACK_NACK_FRAMES_IN_PIPE, ACK_BAD_ENDPT, NACK_ENDPT_CLOSED, NACK_BAD_SESSION, NACK_BAD_RDMAWIN, NACK_EVENTQ_FULL, SEND_BAD_RDMAWIN, CONNECT_TIMEOUT, CONNECT_SRC_UNKNOWN, QUERY_BAD_MAGIC, QUERY_TIMED_OUT, QUERY_SRC_UNKNOWN, RAW SENDS, RAW RECEIVES, RAW_OVERSIZED_PACKETS, RAW_RECV_OVERRUN, RAW DISABLED, CONNECT_SEND, CONNECT_RECV, ACK SEND, ACK_RECV, PUSH_SEND, PUSH_RECV, QUERY SEND, QUERY_RECV</td>
</tr>
<tr>
<td>replies</td>
<td>REPLY_SEND, REPLY_RECV, QUERY_UNKNOWN, DATA_SEND_NULL, DATA_SEND_SMALL, DATA_SEND_MEDIUM, DATA_SEND_RNDV, DATA_SEND_PULL, DATA_RECV_NULL, DATA_RECV_SMALL_INLINE, DATA_RECV_SMALL_COPY, DATA_RECV_MEDIUM, DATA_RECV_RNDV, DATA_RECV_PULL, ETHER_SEND_UNICAST_CNT, ETHER_RECV_BIG_CNT, ETHER_RECV_MULTICAST_CNT, ETHER_RECV_SMALL_INLINE, ETHER_RECV_SMALL_COPY, ETHER_RECV_MEDIUM, ETHER_RECV_RNDV, ETHER_RECV_PULL, ETHER_SEND_MULTICAST_CNT, ETHER_RECV_NO_CREDITS, PACKETS_RESENT, PACKETS_DROPPED, MAPPER_ROUTES_UPDATE</td>
</tr>
<tr>
<td>errors</td>
<td>ROUTE DISPERSION, OUT_OF_SEND_HANDLES, OUT_OF_PULL_HANDLES, OUT_OF_PUSH_HANDLES, MEDIUM_CONT_RACE, CMD_TYPE_UNKNOWN, UREQ_TYPE_UNKNOWN, INTERRUPTS_OVERRUN, WAITING_FOR_INTERRUPT_DMA, WAITING_FOR_INTERRUPT_ACK, WAITING_FOR_INTERRUPT_TIMER, SLABS_RECYCLING, SLABS_PRESSURE, SLABS_STARVATION, OUT_OF_RDMA_HANDLES, EVENTQ_FULL, BUFFER_DROP, MEMORY_DROP, HARDWARE_FLOW_CONTROL, SIMULATED_PACKETS_LOST, LOGGING_FRAMES_DUMPED, WAKE_INTERRUPTS, AVERTED_WAKEUP_race, DMA_METADATA_RACE</td>
</tr>
</tbody>
</table>

CScADS Autotuning
Myrinet MX Counters

| LANAI UPTIME                  | ACK_NACK_FRAMES_INPIPE | ACK_RECV       | ROUTE_DISPERSION          |
| COUNTERS_UPTIME              | NACK_BAD_ENDPT        | ACK_SEND       | OUT_OF_SEND_HANDLE        |
| BAD_CRC8                     | NACK_ENDPT_CLOSED     | ACK_RECV       | OUT_OF_PULL_HANDLES       |
| BAD_CRC32                    | NACK_BAD_SESSION      | ACK_RECV       | OUT_OF_PUSH_HANDLES       |
| UNSTRIPPED_ROUTE             | NACK_RDMACOMIN        | ACK_RECV       | MEDIUM_CONT_RACE          |
| PKT_DESC_INVALID             | NACK_EVENTQ_FULL      | ACK_RECV       | CMD_TYPE_UNKNOWN          |
| RECV_PKT_ERRORS              | SEND_BAD_RDMACOMIN    | ACK_RECV       | UREQ_TYPE_UNKNOWN         |
| PKT_MISROUTED                | CONNECT_TIMEOUT       | ACK_RECV       | INTERRUPTS_OVERRUN        |
| DATA_SRC_UNKNOWN             | CONNECT_SRC_UNKNOWN   | ACK_RECV       | WAITING_FOR_INTERRUPT_DMA|
| DATA_BAD_ENDPT               | QUERY_BAD_MAGIC       | ACK_RECV       | WAITING_FOR_INTERRUPT_ACK|
| DATA_ENDPT_CLOSED            | QUERY_TIMED_OUT       | ACK_RECV       | WAITING_FOR_INTERRUPT_TIM|
| DATA_BAD_SESSION             | QUERY_SRC_UNKNOWN     | ACK_RECV       | ER                      |
| PUSH_BAD_WINDOW              | RAW_SEND              | ACK_RECV       | SLABS_RECYCLING           |
| PUSH_DUPLICATE               | RAW_RECEIVES          | ACK_RECV       | SLABS_PRESSURE            |
| PUSH_OBSOLETE                | RAW_OVERSIZED_PACKETS | ACK_RECV       | SLABS_STARVATION          |
| PUSH_RACE Driver             | RAW_RECV_OVERRUN      | ACK_RECV       | OUT_OF_RDMA_HANDLES       |
| PUSH_BAD_SEND_HANDLE_MAGIC   | RAW_DISABLED          | ACK_RECV       | EVENTQ_FULL               |
| PUSH_BAD_SRC_MAGIC           | CONNECT_SEND          | ACK_RECV       | BUFFER_DROP               |
| PULL_OBSOLETE                | CONNECT_RECV          | ACK_RECV       | MEMORY_DROP               |
| PULL_NOTIFY_OBSOLETE         | ACK_SEND              | ACK_RECV       | HARDWARE_FLOW_CONTROL     |
| PULL_RACE_DRIVER             | ACK_RECV              | ACK_RECV       | SIMULATED_PACKETS_LOST    |
| ACK_BAD_TYPE                 | PUSH_SEND             | ACK_RECV       | LOGGING_FRAMES_DUMPED     |
| ACK_BAD_MAGIC                | PUSH_RECV             | ACK_RECV       | WAKE_INTERRUPTS           |
| ACK_RESEND_RACE              | QUERY_SEND            | ACK_RECV       | AVERTED_WAKEUP_RACE        |
| LATE_ACK                     | QUERY_RECV            | ACK_RECV       | DMA_METADATA_RACE         |
|                             |                       |               |                           |
Multiple Measurements

- The HPCC HPL benchmark with 3 performance metrics:
  - FLOPS; Temperature; Network Sends/Receives
    - Temperature is from an on-chip thermal diode
Multiple Measurements (2)

- The HPCC HPL benchmark with 3 performance metrics:
  - FLOPS; Temperature; Network Sends/Receives
    - Temperature is from an on-chip thermal diode
Eclipse PTP IDE

CScADS Autotuning
Performance Evaluation within Eclipse PTP
TAU and PAPI Plugins for Eclipse PTP

CScADS Autotuning
Potential Autotuning Opportunities

- Provide feedback to compilers or search engines
- Run-time monitoring for dynamic tuning or selection
- Minimally intrusive collection of algorithm/application statistics
- How little data do we need?
  - How can we find the needle in the haystack?
- Other suggestions?
Conclusions

- PAPI has a long track record of successful adoption and use.
- New architectures pose a challenge for off-processor hardware monitoring as well as interpretation of counter values.
- Integration of perfmon2 into the Linux kernel will broaden the base of PAPI users still further.
Hardware Performance Monitoring with PAPI

Dan Terpstra
terpstra@cs.utk.edu

CScADS Autotuning Workshop
July 2007