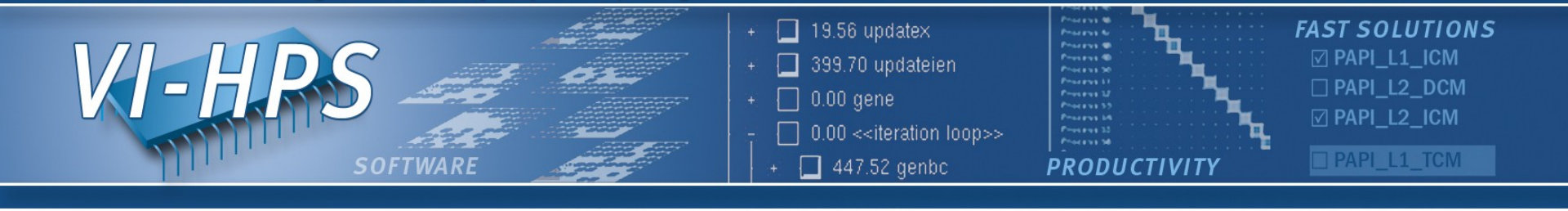


PAPI:
the Performance API

Dan Terpstra, Heike Jagode
terpstra | jagode@eecs.utk.edu
February 18th 2009

THE UNIVERSITY of TENNESSEE

Table of Contents

* With thanks to Paul Drongowski of AMD:
http://developer.amd.com/pages/1212200690.aspx

What's PAPI?

- Middleware that provides a consistent and efficient programming interface for the performance counter hardware found in most major microprocessors.

- Started as a Parallel Tools Consortium project in 1998
  - Goal was to produce a specification for a **portable interface** to the hardware performance counters.

- Countable events are defined in two ways:
  - Platform-neutral **Preset Events** (e.g., PAPI_TOT_INS)
  - Platform-dependent **Native Events** (e.g., L3_CACHE_MISS)

- Preset Events can be derived from multiple Native Events (e.g. PAPI_L1_TCM might be the sum of L1 Data Misses and L1 Instruction Misses on a given platform)

PAPI High Level Calls

1. PAPI_num_counters()
   - get the number of hardware counters available on the system
2. PAPI_flips(**float** **rtime*, **float** **ptime*, **long long** **flpins*, **float** **mflips*)
   - simplified call to get Mflips/s (floating point instruction rate), real and processor time
3. PAPI_flops (**float** **rtime*, **float** **ptime*, **long long** **flpops*, **float** **mflops*)
   - simplified call to get Mflops/s (floating point operation rate), real and processor time
4. PAPI_ipc (**float** **rtime*, **float** **ptime*, **long long** **ins*, **float** **ipc*)
   - gets instructions per cycle, real and processor time
5. PAPI_accum_counters (**long long** **values*, **int** *array_len*)
   - add current counts to array and reset counters
6. PAPI_read_counters (**long long** **values*, **int** *array_len*)
   - copy current counts to array and reset counters
7. PAPI_start_counters (**int** **events*, **int** *array_len*)
   - start counting hardware events
8. PAPI_stop_counters (**long long** **values*, **int** *array_len*)
   - stop counters and return current counts

Example: Low Level API

```
#include "papi.h"
#define NUM_EVENTS 2
int Events[NUM_EVENTS]={PAPI_FP_OPS,PAPI_TOT_CYC},
int EventSet;
long long values[NUM_EVENTS];

/* Initialize the Library */
retval = PAPI_library_init (PAPI_VER_CURRENT);
/* Allocate space for the new eventset and do setup */
retval = PAPI_create_eventset (&EventSet);
/* Add Flops and total cycles to the eventset */
retval = PAPI_add_events (&EventSet,Events,NUM_EVENTS);

/* Start the counters */
retval = PAPI_start (EventSet);

do_work();  /* What we want to monitor*/

/*Stop counters and store results in values */
retval = PAPI_stop (EventSet,values);
```

- Follow along if you like…
  - cluster-sun (Sun Niagara2 cluster ; PAPI 3.6.0)
  - mars (Altix/Itanium; PAPI 3.6.2)
  - jump (IBM POWER6; PAPI 3.6.2)
    - module load papi

PAPI Utilities: *papi_cost*

```
jump01lm: ~>papi_cost -h
This is the PAPI cost program.
It computes min / max / mean / std. deviation for PAPI start/stop
   pairs and for PAPI reads.  Usage:

    cost [options] [parameters]
    cost TESTS_QUIET

Options:

  -b BINS       set the number of bins for the graphical
   distribution of costs. Default: 100
  -d            show a graphical distribution of costs
  -h            print this help message
  -s            show number of iterations above the first 10 std
   deviations
  -t THRESHOLD  set the threshold for the number of iterations.
   Default: 100,000
```

PAPI Utilities: *papi_avail*

```
jump01lm: ~>papi_avail -h
Usage: papi_avail [options]
Options:

General command options:
        -a, --avail   Display only available preset events
        -d, --detail  Display detailed information about all
                                  preset events
        -e EVENTNAME  Display detail information about specified
                             preset or native event
        -h, --help    Print this help message


This program provides information about PAPI preset and native
   events.
```

PAPI Utilities: *papi_avail*

```
jump01lm: ~>papi_avail
Available events and hardware information.
--------------------------------------------------------------------------------
Vendor string and code   : IBM (-1)
Model string and code    : POWER6 (16384)
CPU Revision             : 1081344.000000
CPU Megahertz            : 4704.000000
CPU Clock Megahertz      : 0
CPU's in this Node       : 16
Nodes in this System     : 1
Total CPU's              : 16
Number Hardware Counters : 6
Max Multiplex Counters   : 32
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

    Name        Code    Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes   Yes  Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   No   Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes   No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  Yes   Yes  Level 3 data cache misses
PAPI_L3_ICM  0x80000005  Yes   No   Level 3 instruction cache misses
[...]

-------------------------------------------------------------------------
Of 103 possible events, 34 are available, of which 7 are derived.
avail.c                                  PASSED
```

PAPI Utilities: *papi_avail*

```
jump01lm: ~>papi_avail -a
Available events and hardware information.
--------------------------------------------------------------------------------
Vendor string and code   : IBM (-1)
Model string and code    : POWER6 (16384)
CPU Revision             : 1081344.000000
CPU Megahertz            : 4704.000000
CPU Clock Megahertz      : 0
CPU's in this Node       : 16
Nodes in this System     : 1
Total CPU's              : 16
Number Hardware Counters : 6
Max Multiplex Counters   : 32
--------------------------------------------------------------------------------
The following correspond to fields in the PAPI_event_info_t structure.

    Name        Code    Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes  Level 1 data cache misses
PAPI_L1_ICM  0x80000001  No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  No   Level 2 data cache misses
PAPI_L2_ICM  0x80000003  No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  Yes  Level 3 data cache misses
...
PAPI_FP_OPS  0x80000066  Yes  Floating point operations
-----------------------------------------------------------------
Of 34 available events, 7 are derived.

avail.c                                  PASSED
```

PAPI Utilities: *papi_avail*

```
jump01lm: ~>papi_avail -e PAPI_L1_DCM

Available events and hardware information.
--------------------------------------------------------------------------------
[...]
Event name:                     PAPI_L1_DCM
Event Code:                     0x80000000
Number of Native Events:        2
Short Description:              |L1D cache misses|
Long Description:               |Level 1 data cache misses|
Developer's Notes:              ||
Derived Type:                   |DERIVED_ADD|
Postfix Processing String:      ||
 Native Code[0]: 0x400000c7     |PM_LD_MISS_L1|
 Number of Register Values:     2
 Register[ 0]:    0x00000000    |PowerPC64 event code|
 Register[ 1]:    0xffffb9c4    |PowerPC64 counter_cmd code|
 Native Event Description:      |Load references that miss the Level 1 Data cache. Combined
    unit 0 + 1.|

 Native Code[1]: 0x40000134     |PM_ST_MISS_L1|
 Number of Register Values:     2
 Register[ 0]:    0x00000000    |PowerPC64 event code|
 Register[ 1]:    0xffffb9c4    |PowerPC64 counter_cmd code|
 Native Event Description:      |A store missed the dcache. Combined Unit 0 + 1.|

--------------------------------------------------------------------------------
avail.c                                 PASSED
```

PAPI Utilities: *papi_native_avail*

```
jump01lm: ~>papi_native_avail


Available native events and hardware information.
-------------------------------------------------------------
[...]

Event Code    Symbol   | Long Description |
-------------------------------------------------------------
0x40000000   PM_0INST_FETCH  | No instructions were fetched this cycles (due t |
           | o IFU hold, redirect, or icache miss)                            |
Groups:   55 188
0x40000001   PM_1PLUS_PPC_CMPL  | A group containing at least one PPC instruct |
           | ion completed. For microcoded instructions that span multiple gro |
           | ups, this will only occur once.                                  |
Groups:   54
[...]


0x40000227   PM_RUN_INST_CMPL  | Number of run instructions completed.          |
Groups:   37  46
-------------------------------------------------------------------------------
Total events reported: 552
native_avail.c                          PASSED
```

PAPI Utilities: *papi_event_chooser*

```
jump01lm: ~>papi_event_chooser

Usage: eventChooser NATIVE|PRESET evt1 evt2 ...
```

PAPI Utilities: *papi_event_chooser*

```
jump01lm: ~>papi_event_chooser PRESET PAPI_L1_DCM


Event Chooser: Available events which can be added with given events.
-----------------------------------------------------------------------
    Name         Code    Deriv Description (Note)
PAPI_L1_LDM  0x80000017  No   Level 1 load misses
PAPI_L1_STM  0x80000018  No   Level 1 store misses
PAPI_TOT_INS 0x80000032  No   Instructions completed
PAPI_LD_INS  0x80000035  No   Load instructions
PAPI_SR_INS  0x80000036  No   Store instructions
PAPI_TOT_CYC 0x8000003b  No   Total cycles
PAPI_LST_INS 0x8000003c  Yes  Load/store instructions completed
PAPI_L1_DCA  0x80000040  Yes  Level 1 data cache accesses
PAPI_L1_DCR  0x80000043  No   Level 1 data cache reads
PAPI_L1_DCW  0x80000046  No   Level 1 data cache writes
-----------------------------------------------------------------------
Total events reported: 10
event_chooser.c                                 PASSED
```

PAPI Utilities: *papi_event_chooser*

```
jump01lm: ~>papi_event_chooser PRESET PAPI_L1_DCM PAPI_TOT_INS


Event Chooser: Available events which can be added with given events.
-----------------------------------------------------------------------

    Name        Code    Deriv Description (Note)
PAPI_L1_LDM  0x80000017  No   Level 1 load misses
PAPI_L1_STM  0x80000018  No   Level 1 store misses
PAPI_TOT_CYC 0x8000003b  No   Total cycles
-----------------------------------------------------------------------
Total events reported: 3
event_chooser.c                                 PASSED
```

PAPI Utilities: *papi_command_line*

```
jump01lm: ~>papi_command_line PAPI_FP_OPS
Successfully added: PAPI_FP_OPS

PAPI_FP_OPS :   100000000

--------------------------------
Verification: None.
 This utility lets you add events from the command line interface to see if they work.
command_line.c                          PASSED


jump01lm: ~>papi_command_line PAPI_FP_OPS PAPI_L1_DCA
Successfully added: PAPI_FP_OPS
Successfully added: PAPI_L1_DCA

PAPI_FP_OPS :   100000000
PAPI_L1_DCA :   120034404

--------------------------------
Verification: None.
 This utility lets you add events from the command line interface to see if they work.
command_line.c                          PASSED
```

Performance Measurement Categories

- Efficiency
  - Instructions per cycle (IPC)
  - Memory bandwidth
- Caches
  - Data cache misses and miss ratio
  - Instruction cache misses and miss ratio
- L2 cache misses and miss ratio
- Translation lookaside buffers (TLB)
  - Data TLB misses and miss ratio
  - Instruction TLB misses and miss ratio
- Control transfers
  - Branch mispredictions
  - Near return mispredictions

The Code

```
#define ROWS 1000              // Number of rows in each matrix
#define COLUMNS 1000           // Number of columns in each matrix
```

```
void classic_matmul()
{
   // Multiply the two matrices
   int i, j, k;
   for (i = 0; i < ROWS; i++) {
     for (j = 0; j < COLUMNS; j++) {
        float sum = 0.0;
       for (k = 0; k < COLUMNS; k++) {
          sum +=
             matrix_a[i][k] * matrix_b[k][j];
        }
        matrix_c[i][j] = sum;
      }
   }
}
```

```
void interchanged_matmul()
{
   // Multiply the two matrices
   int i, j, k;
   for (i = 0; i < ROWS; i++) {
     for (k = 0; k < COLUMNS; k++) {
       for (j = 0; j < COLUMNS; j++) {
          matrix_c[i][j] +=
             matrix_a[i][k] * matrix_b[k][j];
       }
     }
   }
}

// Note that the nesting of the innermost loops
// has been changed. The index variables j and k
// change the most frequently and the access
// pattern through the operand matrices is
// sequential using a small stride (one.) This
// change improves access to memory data through
// the data cache. Data translation lookaside
// buffer (DTLB) behavior is also improved.
```

ICL UT

PAPI

IPC – instructions per cycle

- Measure instruction level parallelism
- An indicator of code efficiency

PAPI High Level

```
int events[] = {PAPI_TOT_CYC, PAPI_TOT_INS};

realtime[0] = PAPI_get_real_usec();
retval = PAPI_start_counters(events, 2);
classic_matmul();
retval = PAPI_stop_counters(cvalues, 2);
realtime[1] = PAPI_get_real_usec();
```

PAPI Low Level

```
int events[] = {PAPI_TOT_CYC, PAPI_TOT_INS};

retval = PAPI_library_init (PAPI_VER_CURRENT);
retval = PAPI_create_eventset(&EventSet);
retval = PAPI_add_events(EventSet, events, 2);
realtime[0] = PAPI_get_real_usec();
retval = PAPI_start(EventSet);
classic_matmul();
retval = PAPI_stop(EventSet, cvalues);
realtime[1] = PAPI_get_real_usec();
```

IPC – instructions per cycle


```
Measurement                          Classic mat_mul         Reordered mat_mul
================================================================================


High Level IPC Test (PAPI_{start,stop}_counters)
Real time                             13.6106 sec              2.9762 sec
IPC                                    0.3697                  1.6939
PAPI_TOT_CYC                       24362605525             5318626915
PAPI_TOT_INS                        9007034503             9009011245


Low  Level IPC Test (PAPI low level calls)
Real time                             13.6113 sec              2.9772 sec
IPC                                    0.3697                  1.6933
PAPI_TOT_CYC                       24362750167             5320395138
PAPI_TOT_INS                        9007034381             9009011130
```

- Both PAPI methods are consistent
- Roughly 460% improvement in reordered code

ICL UT

PAPI

Data Cache Access

Cache miss: a failed attempt to read or write a piece of data in the cache

→ Results in main memory access with much longer latency

→ Important to keep data as close as possible to CPU

Data Cache Misses can be considered in 3 categories:

- **Compulsory misses: Occurs on first reference to a data item**
  - **Prefetching can help**
- **Capacity misses: Occurs when the working set exceeds the cache capacity**
  - **Spatial locality: use all the data that is loaded into the cache**
  - **Smaller working set (blocking/tiling algorithms)**
- **Conflict misses: Occurs when a data item is referenced after the cache line containing the item was evicted earlier.**
  - **Temporal locality: reuse a word as long as possible**
  - **Data layout; memory access patterns**

L1 Data Cache Access

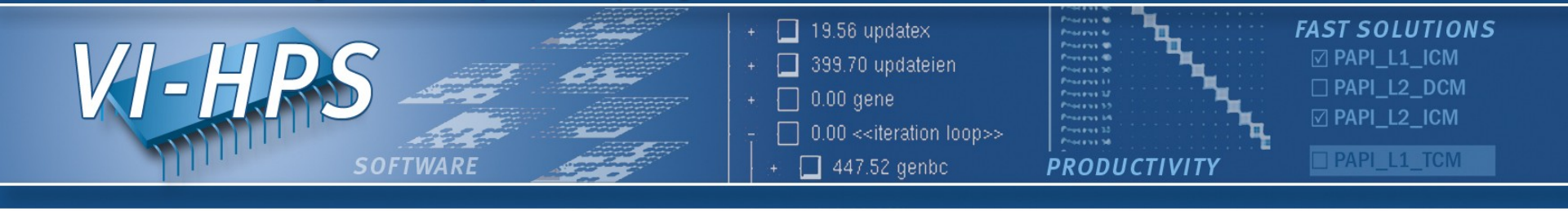
```
Measurement                              Classic mat_mul         Reordered mat_mul
=============================================================================

DATA_CACHE_ACCESSES                      2,002,807,841        3,008,528,961
DATA_CACHE_REFILLS:L2_MODIFIED:L2_OWNED:L2_EXCLUSIVE:L2_SHARED
                                           205,968,263           60,716,301
DATA_CACHE_REFILLS_FROM_SYSTEM:MODIFIED:OWNED:EXCLUSIVE:SHARED
                                            61,970,925            1,950,282
---------------------
PAPI_L1_DCA                              2,002,808,034        3,008,528,895
PAPI_L1_DCM                                268,010,587           62,680,818
---------------------

Data Cache Request Rate                  0.2224 req/inst      0.3339 req/inst
Data Cache Miss Rate                     0.0298 miss/inst     0.0070 miss/inst
Data Cache Miss Ratio                    0.1338 miss/req      0.0208 miss/req
```

- Two techniques
  - First uses native events
  - Second uses PAPI presets only
- ~50% more requests from reordered code
- 1/4 as many misses per instruction
- 1/6 as many misses per request

Reference:

- http://icl.cs.utk.edu/papi/
- http://icl.cs.utk.edu/papi/faq/

Partner Logos:



PAPI:
the Performance API

Dan Terpstra, Heike Jagode
terpstra | jagode@eecs.utk.edu
February 18th 2009

THE UNIVERSITY of TENNESSEE