PDC Summer School 2010
PAPI performance hardware counters HOWTO

August 21, 2010

1 Introduction

PAPI is a library that monitors hardware events when a program runs. This makes it possible to see e.g. exactly how many cache misses that occurred. When optimizing numerical codes this is highly valuable information. *Papiex* is a tool that makes it easy to get access to performance counters using PAPI. Here we describe how to run Papiex on Ferlin.

PAPI and PapiEx were mainly written by Phil Mucci. Please consult the PapiEx homepage[1] for documentation and examples.

2 Modules

The relevant modules are included in the `summer` module. Otherwise, load modules

```
> module add i-compilers monitor papi papiex
```

The Intel compilers are not necessary to run Papiex, but we will use them in an example later.

3 Basic usage

You compile your program as usual, then call

```
> papiex -e <EVENT> ./my_prog
```

where `<EVENT>` is the hardware event you want to profile (see next section). In some cases multiple events can be monitored at the same time.

4 Available events

To see a list of all events that PAPI can profile on the current machine, do

```
> papi_avail
```

Some of the most interesting counters on the Ferlin nodes are

[1]http://icl.cs.utk.edu/ mucci/papiex/

PAPI_TOT_CYC	Total cycles
PAPI_L1_DCM	Level 1 data cache misses
PAPI_L1_DCH	Level 1 data cache hits
PAPI_L1_TCA	Level 1 total cache accesses
PAPI_L2_TCM	Level 2 cache misses
PAPI_L2_TCH	Level 2 total cache hits
PAPI_TLB_DM	Data translation lookaside buffer misses
PAPI_FP_INS	Floating point instructions
PAPI_FP_OPS	Floating point operations
PAPI_VEC_INS	Vector/SIMD instructions
PAPI_BR_MSP	Conditional branch instructions mispredicted
PAPI_BR_PRC	Conditional branch instructions correctly predicted

5 Case study: Scale matrix

Consider the problem of multiplying each element of a matrix with a scalar, as in this C99 code:

```
#include <stdlib.h>

int main(int args, char* argv[])
{
    const int N = atoi(argv[1]);
    double* mat = (double*) malloc(N*N*sizeof(double));

    for(int i=0; i<N*N; i++) /* initialize some values */
        mat[i] = 2;

    for(int i=0; i<N; i++)
        for(int j=0; j<N; j++)
            mat[i*N + j] *= 10.2; /* scale each value*/

    free(mat);
}
```

Build this program with

```
> icc -std=c99 scale_mat.c -o scale_mat
```

and run with

```
> ./scale_mat <matrix size>
```

You may wish to turn of optimizations (use the flag -O0) for some tests. Clearly in this example, a cache miss should occur every 8 loads (since the cache line on Ferlin is 8 doubles). If this wasn't obvious, as in a real code, we could let PapiEx tell us this by doing

```
> papiex -e PAPI_L2_DCM ./scale_mat 5000
PAPI_L2_DCM ..................................      3.13236e+06
```

This agrees with what we expect: $5000^2/8 = 3125000$.