

# COMP3320/COMP6464: Hardware Performance Counters

Alistair Rendell

See various articles linked to lecture note web site

## 7.1 Why

- We've seen that modern machines are complex
  - pipelining
  - superscalar
  - load/store architectures
  - memory hierarchy
- Understanding observed performance is not easy
- Performance counters enable us to count critical events and provide a very accurate means of assessing how well we are using the computer system

*Can you suggest some quantities that we might want to count?*

## 7.2 Hardware Performance Counters

- All (nearly?) modern microprocessors have them
- Typically a group of registers that keep track of programmable events
- Provide high resolution data on many performance related variables, e.g.
  - cycles
  - instruction count
  - floating point operations
  - cache references
  - TLB misses
  - data/instruction stalls
- Enable vendors to better understand the performance of existing code on their hardware
- Enable users to build better (faster) software

## 7.3 Accessing Hardware Counters

- Cray YMP provided HPM that gave info on vector lengths and flops
  - enabled users to quote macho flops!
- Used to build metrics
- Used in system wide tools (Solaris provides: cputrack, cpustat, har)
- Accessed via libraries
  - vendor specific: (libcpc, libpctx)
  - portable (PCL, PAPI)
  - futher GUI often provided to provide higher level analysis
- There are no standard counters
  - different vendors have different counters
  - different generations of the same processor may have different counters (eg UII and UIII)

## 7.4 Simple Metrics

$$\text{MFLOPS} = \frac{\text{Flp\_Instr\_Exec}}{\text{Cycles}} \times \text{clock}(MHz)$$

$$\text{MIPS} = \frac{\text{Int\_Instr\_Exec}}{\text{Cycles}} \times \text{clock}(MHz)$$

$$\text{IPC} = \frac{\text{All\_Instr\_Exec}}{\text{Cycles}}$$

$$\text{L1Hits} = 1 - \frac{\text{L1\_Misses}}{\text{Loads} + \text{Stores}}$$

$$\text{L2Hits} = 1 - \frac{\text{L2\_Misses}}{\text{L1\_Misses}}$$

$$\text{Branch\_rate} = \frac{\text{Decoded\_Branches}}{\text{Total\_Instruction\_Exec.}}$$

## 7.5 More Complex Metrics

$$\text{L1 - L2\_bandwidth} = \frac{L1\_Misses \times L1\_Line\_Size}{Cycles} \times \text{clock}(MHz)$$

$$\text{L2 - RAM\_bandwidth} = \frac{L2\_Misses \times L2\_Line\_Size}{Cycles} \times \text{clock}(MHz)$$

$$\text{Data\_Stall} = \frac{Load\_Use + Load\_Use\_Raw + Store\_Buf\_Full}{Cycles}$$

## 7.6 Sun Performance Counters

- Two 32 bit counters (PIC0 and PIC1)
  - max 4 billion events - so will overflow
  - programmed to record different events (ultrall)

Code	Register	Meaning
Cycle_cnt	0+1	Accumulated cycles
Instr_cnt	0+1	Completed instructions (not annulled ones)
IC_ref	0	I-cache references
IC_hits	1	I-cache hits
DC_rd	0	D-cache read reference
DC_rd_hit	1	D-cache read hit
DC_wr	0	D-cache write reference
DC_wr_hit	1	D-cache write hit
EC_ref	0	Total e-cache references
EC_hit	1	Total e-cache hits
EC_wb	1	e-cache misses that do writebacks

## 7.7 UltraSPARC III Counters

event0: Cycle\_cnt Instr\_cnt Dispatch0\_IC\_miss IC\_ref DC\_rd DC\_wr  
EC\_ref EC\_snoop\_inv Dispatch0\_br\_target Dispatch0\_2nd\_br  
Rstall\_storeQ Rstall\_IU\_use EC\_write\_hit\_RTO EC\_rd\_miss  
PC\_port0\_rd SI\_snoop SI\_ciq\_flow SI\_owned SW\_count\_0  
IU\_Stat\_Br\_miss\_taken IU\_Stat\_Br\_count\_taken  
Dispatch\_rs\_mispred FA\_pipe\_completion MC\_reads\_0  
MC\_reads\_1 MC\_reads\_2 MC\_reads\_3 MC\_stalls\_0 MC\_stalls\_2

event1: Cycle\_cnt Instr\_cnt Dispatch0\_mispred EC\_wb EC\_snoop\_cb  
IC\_miss\_cancelled Re\_FPU\_bypass Re\_DC\_miss Re\_EC\_miss  
IC\_miss DC\_rd\_miss DC\_wr\_miss Rstall\_FP\_use EC\_misses  
EC\_ic\_miss Re\_PC\_miss ITLB\_miss DTLB\_miss WC\_miss  
WC\_snoop\_cb WC\_scrubbed WC\_wb\_wo\_read PC\_soft\_hit  
PC\_snoop\_inv PC\_hard\_hit PC\_port1\_rd SW\_count\_1  
IU\_Stat\_Br\_miss\_untaken IU\_Stat\_Br\_count\_untaken  
PC\_MS\_misses MC\_writes\_0 MC\_writes\_1 MC\_writes\_2  
MC\_writes\_3 MC\_stalls\_1 MC\_stalls\_3 Re\_RAW\_miss  
FM\_pipe\_completion Re\_endian\_miss

## 7.8 Metrics for Ultra II and Ultra III

Metric	Ultra II	Ultra III
mips	$\text{instr\_cnt}/\text{tick\_cnt}*\text{clock}$	$\text{instr\_cnt}/\text{tick\_cnt}*\text{clock}$
cpi	$\text{cycle\_cnt}/\text{instr\_cnt}$	$\text{cycle\_cnt}/\text{instr\_cnt}$
flops		$(\text{fa\_pipe\_completion} + \text{fm\_pipe\_completion}) / \text{tick\_cnt}*\text{clock}$
address bus utilization	$2 * (\text{ec\_ref} - \text{ec\_hit} + \text{ec\_wb}) / (\text{tick\_cnt} * \text{bus\_clock}/\text{cpu\_clock})$	$(\text{ec\_misses} + \text{ec\_wb}) / (\text{tick\_cnt} + \text{bus\_clock}/\text{cpu\_clock})$
D-cache miss rate	$1 - (\text{dc\_rd\_hit} + \text{dc\_wr\_hit})/(\text{dc\_rd} + \text{dc\_wr})$	$(\text{dc\_rd\_miss} + \text{dc\_wr\_miss})/(\text{dc\_rd} + \text{dc\_wr})$
I-cache miss rate	$1 - \text{ic\_hit}/\text{ic\_ref}$	$(\text{ic\_miss} - \text{ic\_miss\_canceled}) / (\text{ic\_ref} + \text{ic\_miss} - \text{ic\_miss\_canceled})$
L2 miss rate	$1 - \text{ec\_hit}/\text{ec\_ref}$	$\text{ec\_misses}/\text{ec\_ref}$
I-TLB miss rate		$\text{itlb\_miss}/\text{ic\_ref}$
D-tlb miss rate		$\text{dtlb\_miss}/(\text{dc\_rd} + \text{dc\_wr})$
branch rate		$(\text{i\_u\_stat\_br\_count\_taken} + \text{i\_u\_stat\_br\_count\_untaken}) / \text{instr\_cnt}$
Data stall rate	$(\text{load\_use} + \text{load\_use\_raw} + \text{dispatch0\_storebuf})/\text{cycle\_cnt}$	$(\text{re\_dc\_miss} + \text{rstall\_storeq} + \text{rstall\_iu\_use} + \text{re\_raw\_miss} + 12 * (\text{re\_pc\_miss} + \text{re\_fpu\_bypass} + \text{re\_endian\_miss})) / \text{cycle\_cnt}$

## 7.9 Sun Tools: CPUSTAT

- Provides a system wide report on CPU counters
  - need to be root or have suid set to root

```
cpustat -c eventspec [-c eventspec ] ... [-ntD][interval[count]]
```

```
cpustat -c Cycle_cnt,Instr_cnt -c EC_ref,EC_misses 1 5
```

time	cpu	event	pic0	pic1	
1.007	1	tick	48959	4640	# pic0=Cycle_cnt,pic1=Instr_cnt
1.007	0	tick	271729	39874	# pic0=Cycle_cnt,pic1=Instr_cnt
2.007	0	tick	8768	390	# pic0=EC_ref,pic1=EC_misses
2.007	1	tick	11071	1410	# pic0=EC_ref,pic1=EC_misses
3.007	0	tick	118110	47512	# pic0=Cycle_cnt,pic1=Instr_cnt
3.007	1	tick	238131	57790	# pic0=Cycle_cnt,pic1=Instr_cnt
4.007	0	tick	6215	116	# pic0=EC_ref,pic1=EC_misses
4.007	1	tick	2214	197	# pic0=EC_ref,pic1=EC_misses
5.007	0	tick	139530	76730	# pic0=Cycle_cnt,pic1=Instr_cnt
5.007	1	tick	607836	228673	# pic0=Cycle_cnt,pic1=Instr_cnt
5.007	2	total	1424295	455219	# pic0=Cycle_cnt,pic1=Instr_cnt
4.007	2	total	28268	2113	# pic0=EC_ref,pic1=EC_misses

## 7.10 Sun Tools: CPUTRACK

- Provides counter information on a specific process
  - does not require root access

```
cpustrack -c eventspec [-c eventspec ] ... [-efntvD] [-N count] [-o  
pathname] [-T interval] command [args]
```

```
# cputrack -T 0 -fve -c Cycle_cnt,Instr_cnt sh -c date
```

```
time    pid lwp    event    pic0    pic1
0.008   7193  1    init_lwp    0        0
0.024   7193  1      fork                # 7194
0.028   7194  1    init_lwp    0        0
0.031   7194  1    fini_lwp    85848    32003
0.031   7194  1      exec    85848    32003
0.000   7194  1      exec                # 'date'
0.043   7194  1    init_lwp    0        0
Tue Apr 1 23:23:10 EST 2003
0.053   7194  1    fini_lwp    824313   366668
0.053   7194  1      exit    824313   366668
0.058   7193  1    fini_lwp    1058364  411754
0.058   7193  1      exit    1058364  411754
```

## 7.11 Hard Coded Instrumentation: libcpc

```
/* example_libcpc.c */
/* cc -fast -fsimple=2 -xchip=ultra3 -xarch=v8plusa \
   example_libcpc.c -lcpc -o example_libcpc */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <inttypes.h>
#include <libcpc.h>
#include <errno.h>

#define N (500019+1)
#define TIMES 100

int main(int argc, char **argv) {

    int i,j,k,nl,nll;
    double sum = 0.0, b[N], a[N];
    double t1=0.0,t2=0.0;
    int cpuver, iter;
    char *setting = NULL;
    cpc_event_t event;
    long long fltadd=0, fltmult=0, fltaddtheo=0, fltmulttheo=0;
```

```

if ((cpuver = cpc_getcpuver()) == -1) {
    printf("no performance counter hardware \n");
} else {
    printf("hardware identifier %d\n",cpuver);
}

if ((setting = getenv("PERFEVENTS")) == NULL)
    setting = "pic0=FA_pipe_completion,pic1=FM_pipe_completion";

if (cpc_strtoevent(cpuver,setting,&event) != 0)
    printf("Cannot measure %s on this processor\n",setting);
setting = cpc_eventtostr(&event);

if (cpc_bind_event(&event, 0) == -1)
    printf("cannot bind lwp %d %s\n",_lwp_self(),
           strerror(errno));

nl = N-1;
fltaddtheo = ((long long)TIMES)*(N-1);
fltmulttheo = (((long long)TIMES)*(N-1));

{cpc_event_t before, after;
sum = 0.0;
for (j = 0; j < TIMES; j++) {
    for (i = 0; i < N; i++)

```

```

        b[i] = 0.3 + 0.001*j + 0.00001*i;
for (i = 0; i < N; i++)
    a[i] = 0.1 + 0.003*(j-1) + 0.0004*i;
if (cpc_take_sample(&before) == -1) exit(-1);
    for (k = 0; k < N-1; k++)
        sum = sum + a[k]*b[k];
if (cpc_take_sample(&after) == -1) exit(-1);
fltadd += (after.ce_pic[0] - before.ce_pic[0]);
fltmult += (after.ce_pic[1] - before.ce_pic[1]);
}
}

printf("Measured Flt Adds %lld\n",fltadd);
printf("Measured Flt Mults %lld\n",fltmult);
printf("Theoretical Flt Adds %lld\n",fltaddtheo);
printf("Theoretical Flt Mults %lld\n",fltmulttheo);
printf("sum = %14.7e\n", sum);
}
apr900@alto example_libcpc
hardware identifier 1002
Measured Flt Adds 50002100
Measured Flt Mults 50001900
Theoretical Flt Adds 50001900
Theoretical Flt Mults 50001900
sum = 1.8451251e+10

```

## 7.12 Problems and Alternatives?

- Counters provide lots of nice information, but care required
  - use of correct counter
  - counter working correctly (lmbench)
  - hard coded correctly
  - special consideration missed (madd)
  - scripts correct and working
  - counter overflow
- What are the alternatives?