Practical Performance

Aravind Sukumaran Rajam

Objective:

Introduction to performance monitoring

Is measuring time enough?

- Only serves an indicator of how fast the code is
- Lacks insight into bottleneck resource

Tools

- Intel vtune
- gprof
- HPC toolkit
- TAU
- gperftools (Google Performance Tools)
- Perf
- papi

perf basics

- perf list // show supported events
- perf stat // collect event counts
- perf record // record events
- perf report // view recorded events

perf list

- perf list

```
 1   branch-instructions OR branches                    [Hardware event]
 2   branch-misses                                      [Hardware event]
 3   bus-cycles                                         [Hardware event]
 4   cache-misses                                       [Hardware event]
 5   cache-references                                   [Hardware event]
 6   cpu-cycles OR cycles                               [Hardware event]
 7   instructions                                       [Hardware event]
 8   ref-cycles                                         [Hardware event]
 9   page-faults OR faults                              [Software event]
10   task-clock                                         [Software event]
11   L1-dcache-load-misses                              [Hardware cache event]
12   L1-dcache-loads                                    [Hardware cache event]
13   L1-dcache-stores                                   [Hardware cache event]
14   L1-icache-load-misses                              [Hardware cache event]
15   LLC-load-misses                                    [Hardware cache event]
16   LLC-loads                                          [Hardware cache event]
17   LLC-store-misses                                   [Hardware cache event]
18   LLC-stores                                         [Hardware cache event]
19   branch-load-misses                                 [Hardware cache event]
20   branch-loads                                       [Hardware cache event]
21   rNNN                                               [Raw hardware event descriptor]
22   .
23   .
24   .
```

last level cache

raw hw counter

perf events

Event type	Description	Example
Hardware event	Measurements obtained using PMU (Low overhead, Availability depends on processor)	instructions
Hardware cache event		L1-dcache-loads
Software event	Events measured by the kernel	page-faults
Tracepoint event	Markers within the kernel source	kmem:kmalloc

Perf

- perf stat ls

```
 1  Performance counter stats for 'ls tmp':
 2
 3           0.341636      task-clock (msec)         #    0.700 CPUs utilized
 4                  0      context-switches          #    0.000 K/sec
 5                  0      cpu-migrations            #    0.000 K/sec
 6                 87      page-faults               #    0.255 M/sec
 7          1,360,809      cycles                    #    3.983 GHz
 8    <not supported>      stalled-cycles-frontend
 9    <not supported>      stalled-cycles-backend
10          1,148,932      instructions              #    0.84  insns per cycle
11            231,008      branches                  #  676.182 M/sec
12             10,294      branch-misses             #    4.46% of all branches
13
14        0.000488092 seconds time elapsed
15
```

Perf example

```
1  constexpr int M = 16384;
2  constexpr int N = 1024;
3  typedef float MAT_T[M][N];
4
5  void kernel(float* in_A, float* in_B)
6  {
7      CAST_TO_MAT(A, in_A, MAT_T);
8      CAST_TO_MAT(B, in_B, MAT_T);
9
10     for (int i = 0; i < M; ++i)
11     {
12         for (int j = 0; j < N; ++j)
13         {
14             A[i][j] = B[i][j] - 1.2;
15         }
16     }
17 }
```

typecasting to 2d matrix
//A[M][N]
//B[M][N]

Perf

Compile: g++ -std=c++11 perf_eg.cpp -O3 -o perf_eg.out

: perf stat -e branches,branch-misses,instructions,page-faults ./perf_eg.out

5,977,367	branches
42,435	branch-misses
68,630,736	instructions
1,436	page-faults

perf raw

- perf list only shows common events
- bugs in mapping of perf events to pmu events
- some event names are confusing (eg. instructions)
  - issued instructions or retired instructions

Solution

use perf raw events

perf raw

use rNNN

Example: perf stat -e instructions,rc0 ...

```
517,004,606      instructions
517,004,606      rc0
```

How to find the counter id?

perf raw

Intel manual

https://software.intel.com/sites/default/files/managed/39/c5/325462-sdm-vol-1-2abcd-3abcd.pdf

Event Num.	Event Mask Name	Umask Value	Description
3CH	UnHalted Core Cycles	00H	Counts core clock cycles whenever the logical processor is in C0 state (not halted). The frequency of this event varies with state transitions in the core.
3CH	UnHalted Reference Cycles[1]	01H	Counts at a fixed frequency whenever the logical processor is in C0 state (not halted).
C0H	Instructions Retired	00H	Counts when the last uop of an instruction retires.
2EH	LLC Reference	4FH	Accesses to the LLC, in which the data is present (hit) or not present (miss).
2EH	LLC Misses	41H	Accesses to the LLC in which the data is not present (miss).
C4H	Branch Instruction Retired	00H	Counts when the last uop of a branch instruction retires.
C5H	Branch Misses Retired	00H	Counts when the last uop of a branch instruction retires which corrected misprediction of the branch prediction hardware at execution time.

Other useful flags

perf stat -r5 ls

- runs perf 5 times are report the average

perf stat -x ls

- print in csv format

eg:

perf stat -x -r5 ls

```
1.837468,r5,r5task-clock,r51837468,r5100.00
4,r5,r5context-switches,r51837468,r5100.00
0,r5,r5cpu-migrations,r51837468,r5100.00
307,r5,r5page-faults,r51837468,r5100.00
2183506,r5,r5cycles,r51855185,r5100.00
<not supported>,r5,r5stalled-cycles-frontend,r50,r5100.00
<not supported>,r5,r5stalled-cycles-backend,r50,r5100.00
1707354,r5,r5instructions,r51855185,r5100.00
340550,r5,r5branches,r51855185,r5100.00
13836,r5,r5branch-misses,r51855185,r5100.00
```

PAPI

- PAPI
  - Performance Application Programming Interface
- A portable API to access hardware performance counters
- Allows to measure counters for a given code section
- Compatible with HPCtoolkit (HPCview) and TAU
- Well documented
- C and Fortan interface

PAPI

Events

- Native Events
  - Set of events countable by the CPU
  - CPU specific bit pattern is used to indicate an event
  - papi_native_avail utility lists all available present events
- Present Events
  - A common set of predefined events
  - Symbolic names are used
  - Derived/Mapped from one or more native events
  - papi_avail utility lists all available present events

papi_avail

```
Available PAPI preset and user defined events plus hardware information.
--------------------------------------------------------------------------------
PAPI Version             : 5.5.1.0
Vendor string and code   : GenuineIntel (1)
Model string and code    : Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz (94)
CPU Revision             : 3.000000
CPUID Info               : Family: 6  Model: 94  Stepping: 3
CPU Max Megahertz        : 4200
CPU Min Megahertz        : 800
Hdw Threads per core     : 2
Cores per Socket         : 4
Sockets                  : 1
NUMA Nodes               : 1
CPUs per Node            : 8
Total CPUs               : 8
Running in a VM          : no
Number Hardware Counters : 11
Max Multiplex Counters   : 384
--------------------------------------------------------------------------------
```

Limited number of GP counters

Multiplexing/Time sharing

papi_avail

```
================================================================================
  PAPI Preset Events
================================================================================
    Name        Code    Avail Deriv Description (Note)
PAPI_L1_DCM  0x80000000  Yes   No   Level 1 data cache misses
PAPI_L1_ICM  0x80000001  Yes   No   Level 1 instruction cache misses
PAPI_L2_DCM  0x80000002  Yes   Yes  Level 2 data cache misses
PAPI_L2_ICM  0x80000003  Yes   No   Level 2 instruction cache misses
PAPI_L3_DCM  0x80000004  No    No   Level 3 data cache misses
PAPI_L3_ICM  0x80000005  No    No   Level 3 instruction cache misses
PAPI_L1_TCM  0x80000006  Yes   Yes  Level 1 cache misses
PAPI_L2_TCM  0x80000007  Yes   No   Level 2 cache misses
PAPI_L3_TCM  0x80000008  Yes   No   Level 3 cache misses
PAPI_CA_SNP  0x80000009  Yes   No   Requests for a snoop
PAPI_CA_SHR  0x8000000a  Yes   No   Requests for exclusive access to shared cache line
PAPI_CA_CLN  0x8000000b  Yes   No   Requests for exclusive access to clean cache line
PAPI_CA_INV  0x8000000c  No    No   Requests for cache line invalidation
PAPI_CA_ITV  0x8000000d  Yes   No   Requests for cache line intervention
PAPI_L3_LDM  0x8000000e  Yes   No   Level 3 load misses
PAPI_L3_STM  0x8000000f  No    No   Level 3 store misses
PAPI_BRU_IDL 0x80000010  No    No   Cycles branch units are idle
PAPI_FXU_IDL 0x80000011  No    No   Cycles integer units are idle
PAPI_FPU_IDL 0x80000012  No    No   Cycles floating point units are idle
PAPI_LSU_IDL 0x80000013  No    No   Cycles load/store units are idle
PAPI_TLB_DM  0x80000014  Yes   Yes  Data translation lookaside buffer misses
PAPI_TLB_IM  0x80000015  Yes   No   Instruction translation lookaside buffer misses
PAPI_TLB_TL  0x80000016  No    No   Total translation lookaside buffer misses
PAPI_L1_LDM  0x80000017  Yes   No   Level 1 load misses
PAPI_L1_STM  0x80000018  Yes   No   Level 1 store misses
PAPI_L2_LDM  0x80000019  Yes   No   Level 2 load misses
PAPI_L2_STM  0x8000001a  Yes   No   Level 2 store misses
PAPI_BTAC_M  0x8000001b  No    No   Branch target address cache misses
PAPI_PRF_DM  0x8000001c  Yes   No   Data prefetch cache misses
PAPI_L3_DCH  0x8000001d  No    No   Level 3 data cache hits
```

papi_avail -e PAPI_VEC_SP -d

```
--------------------------------------------------------------------------------

Event name:                     PAPI_VEC_SP
Event Code:                     0x80000069
Number of Native Events:        4
Short Description:             |SP Vector/SIMD instr|
Long Description:              |Single precision vector/SIMD instructions|
Developer's Notes:             ||
Derived Type:                  |DERIVED_POSTFIX|
Postfix Processing String:     |N0|N1|N2|N3|+|+|+||
 Native Code[0]: 0x40000026 |FP_ARITH:SCALAR_SINGLE|
 Number of Register Values: 0
 Native Event Description: |Floating-point, masks:Number of scalar single precision
     floating-point arithmetic instructions (multiply by 1 to get flops)|

 Native Code[1]: 0x40000027 |FP_ARITH:128B_PACKED_SINGLE|
 Number of Register Values: 0
 Native Event Description: |Floating-point, masks:Number of scalar 128-bit packed
     single precision floating-point arithmetic instructions
     (multiply by 4 to get flops)|

 Native Code[2]: 0x40000028 |FP_ARITH:256B_PACKED_SINGLE|
 Number of Register Values: 0
 Native Event Description: |Floating-point, masks:Number of scalar 256-bit packed
     single precision floating-point arithmetic instructions
     (multiply by 8 to get flops)|

 Native Code[3]: 0x40000029 |FP_ARITH:512B_PACKED_SINGLE|
 Number of Register Values: 0
 Native Event Description: |Floating-point, masks:Number of scalar 512-bit packed
     single precision floating-point arithmetic instructions
     (multiply by 16 to get flops)|

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

PAPI

High level API

- A simple interface to start, read and stop counters for a specified list of events
- Single threaded (Not thread Safe)
- Only PAPI present events are supported
- Coarse grained measurement
- Easier and requires less setup
- Approximately 8 functions are provided

PAPI: High Level API

Function	Description
PAPI_num_counters()	#hardware counters available
PAPI_flips()	Mflips/s, real and processor time
PAPI_flops()	Mflops/s, real and processor time
PAPI_ipc()	instructions per cycle, real and processor time
PAPI_start_counters()	start counting hardware events
PAPI_read_counters()	copy current counts to array and reset counters
PAPI_accum_counters()	add current counts to array and reset counters
PAPI_stop_counters()	stop counters and return current counts

What is the difference between real and processor/cpu time?

PAPI High level

```
1  #include <papi.h>
2  #include <iostream>
3
4  #define PAPI_ERROR_CHECK(X) if((X)<=PAPI_OK) std::cerr<<"Error \n";
5
6  int main(int argc, char *argv[])
7  {
8      int num_hwcntrs = 0;
9
10     // Initialize the PAPI library and get the number of counters
11     PAPI_ERROR_CHECK(num_hwcntrs = PAPI_num_counters());
12
13     std::cout << "Num counters:  " << num_hwcntrs << "\n";
14     return 0;
15 }
```

$: g++ num_counters.cpp -lpapi -o check
$: ./check
11 //output

PAPI High level

```
1  #define PAPI_ERROR_CHECK(X) if((X)!=PAPI_OK) std::cerr<<"Error \n";
2
3  constexpr int M = 5000;
4  constexpr int N = 10000;
5  typedef float MAT_T[M][N];
6
7  void kernel(float* __restrict__ in_A,
8              float* __restrict__ in_B)
9  {
10     int events[] = {PAPI_TOT_CYC, PAPI_L1_DCM};
11     long_long values[2];
12
13     CAST_TO_MAT(A, in_A, MAT_T);
14     CAST_TO_MAT(B, in_B, MAT_T);
15
16     //Start the counters
17     PAPI_ERROR_CHECK(PAPI_start_counters(events, 2));
18
19     for (int j = 1; j < N - 1; ++j)
20     {
21         for (int i = 1; i < M - 1; ++i)
22         {
23             A[i][j] = B[i + 1][j - 1] * 2;
24         }
25     }
26
27     //Stop counter (Will also read the counters)
28     PAPI_ERROR_CHECK(PAPI_stop_counters(values, 2));
29
30     std::cout << "Cycles " << values[0] << " "
31               << "L1 DCM " << values[1] << "\n";
32 }
```

$: g++ -O3 -std=c++11 papi_high_level.cpp -lpapi -o high_lev

$: ./high_lev

Cycles 745033424 L1 DCM 99887201 //output

PAPI: High Level API

Function	Description
PAPI_num_counters()	#hardware counters available
PAPI_flips()	Mflips/s, real and processor time
PAPI_flops()	Mflops/s, real and processor time
PAPI_ipc()	instructions per cycle, real and processor time
PAPI_start_counters()	start counting hardware events
PAPI_read_counters()	copy current counts to array and reset counters
PAPI_accum_counters()	add current counts to array and reset counters
PAPI_stop_counters()	stop counters and return current counts

var += read(hw_counter); reset(hw_counter);

var = read(hw_counter); reset(hw_counter);

PAPI

Low level API

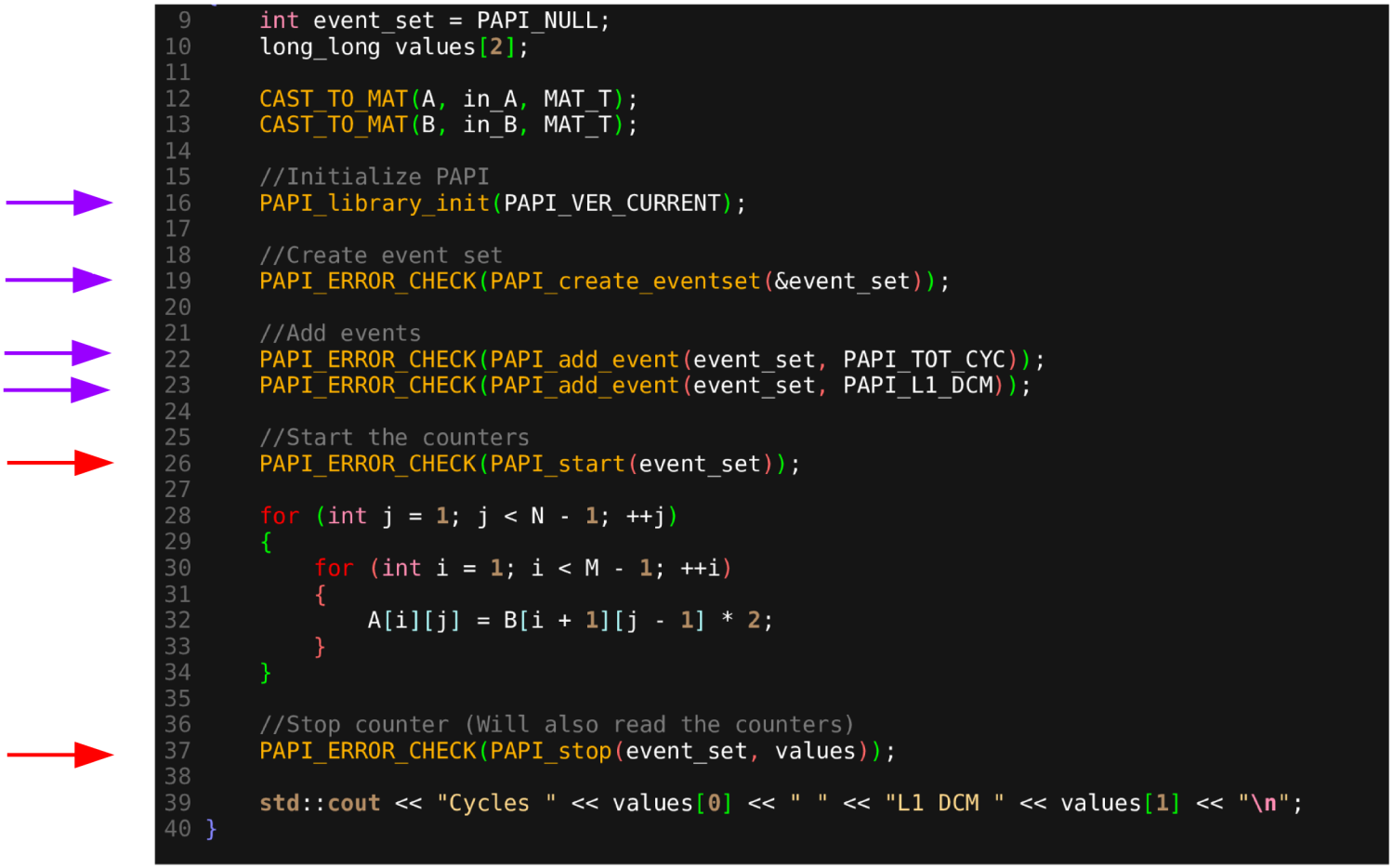
- Manages hardware events in user-defined groups called Event Sets
- Thread safe
- Both native and present events are supported
- Fine grained measurement
- Increased efficiency
- Overflow handling
- Some features are not available on all platforms
- Approximately 50 functions are provided

PAPI: Low Level API

Function	Description
PAPI_library_init()	Initialize library
PAPI_create_eventset()	Create event set
PAPI_add_event()	Add an event to the event set
PAPI_start()	start counting hardware events
PAPI_read()	copy current counts to array
PAPI_accum()	add current counts to array
PAPI_stop()	stop counters and return current counts
PAPI_reset()	reset the counters

Does NOT reset counters

PAPI Low level



```
 9     int event_set = PAPI_NULL;
10     long_long values[2];
11
12     CAST_TO_MAT(A, in_A, MAT_T);
13     CAST_TO_MAT(B, in_B, MAT_T);
14
15     //Initialize PAPI
16     PAPI_library_init(PAPI_VER_CURRENT);
17
18     //Create event set
19     PAPI_ERROR_CHECK(PAPI_create_eventset(&event_set));
20
21     //Add events
22     PAPI_ERROR_CHECK(PAPI_add_event(event_set, PAPI_TOT_CYC));
23     PAPI_ERROR_CHECK(PAPI_add_event(event_set, PAPI_L1_DCM));
24
25     //Start the counters
26     PAPI_ERROR_CHECK(PAPI_start(event_set));
27
28     for (int j = 1; j < N - 1; ++j)
29     {
30         for (int i = 1; i < M - 1; ++i)
31         {
32             A[i][j] = B[i + 1][j - 1] * 2;
33         }
34     }
35
36     //Stop counter (Will also read the counters)
37     PAPI_ERROR_CHECK(PAPI_stop(event_set, values));
38
39     std::cout << "Cycles " << values[0] << " " << "L1 DCM " << values[1] << "\n";
40 }
```

$: g++ -O3 -std=c++11 papi_low_level.cpp -lpapi -o low_lev
$: ./low_lev
Cycles 744870519 L1 DCM 99887136//output

PAPI selected events

Event	Description
PAPI_L1_DCM	Level 1 data cache misses
PAPI_L1_LDM	Level 1 load misses
PAPI_L1_STM	Level 1 store misses
PAPI_TOT_INS	Instructions completed
PAPI_BR_INS	Branch instructions
PAPI_BR_MSP	Conditional branch instructions mis-predicted

Loop interchange

```
#define PAPI_ERROR_CHECK(X) \
        if((X)!=PAPI_OK) std::cerr<<"Error \n";
void kernel(float* in_A, float* in_B)
{
    CAST_TO_MAT(A, in_A, MAT_T);
    CAST_TO_MAT(B, in_B, MAT_T);

    for (int j = 0; j < 10000; ++j)
    {
        for (int i = 0; i < 5000; ++i)
        {
            A[i][j] = B[i][j] * 2;
        }
    }
}
```

```
#define PAPI_ERROR_CHECK(X) \
        if((X)!=PAPI_OK) std::cerr<<"Error \n";
void kernel(float* in_A, float* in_B)
{
    CAST_TO_MAT(A, in_A, MAT_T);
    CAST_TO_MAT(B, in_B, MAT_T);

    for (int i = 0; i < 5000; ++i)
    {
        for (int j = 0; j < 10000; ++j)
        {
            A[i][j] = B[i][j] * 2;
        }
    }
}
```

Compile: g++ -std=c++11 interchange.cpp -O3 -o inter.out -lpapi

**Perf : ** ./inter.out

L1 DCM : 99930900
L1 LDM : 49934356
L1 STM : 50027583

L1 DCM : 6168386
L1 LDM : 3254320
L1 STM : 3121634

Loop interchange

```
1  void kernel(double* in_A, double* in_B)
2  {
3      CAST_TO_MAT(A, in_A, MAT_T);
4      CAST_TO_MAT(B, in_B, MAT_T);
5
6      for (int j = 0; j < N; ++j)
7      {
8          for (int i = 0; i < M; ++i)
9          {
10             A[i][j] = B[i][j] * 2;
11         }
12     }
13 }
```

```
1  void kernel(double* in_A, double* in_B)
2  {
3      CAST_TO_MAT(A, in_A, MAT_T);
4      CAST_TO_MAT(B, in_B, MAT_T);
5
6      for (int i = 0; i < M; ++i)
7      {
8          for (int j = 0; j < N; ++j)
9          {
10             A[i][j] = B[i][j] * 2;
11         }
12     }
13 }
```

Compile: g++ -std=c++11 interchange.cpp -O3 -o inter.out -lpapi

L1 DCM : **99930900**

L1 DCM : **6168386**

Compile: g++ -std=c++11 interchange.cpp -O3 -floop-interchange -o inter.out -papi

L1 DCM : **6168641**

L1 DCM : **6168425**

Loop interchange

```
1  #define PAPI_ERROR_CHECK(X) \
2          if((X)!=PAPI_OK) std::cerr<<"Error \n";
3  void kernel(float* in_A, float* in_B)
4  {
5      CAST_TO_MAT(A, in_A, MAT_T);
6      CAST_TO_MAT(B, in_B, MAT_T);
7
8      for (int j = 1; j < 10000 - 1; ++j)
9      {
10         for (int i = 1; i < 5000 - 1; ++i)
11         {
12             A[i][j] = B[i + 1][j - 1] * 2;
13         }
14     }
15 }
```

Possible dependence

Is interchange valid?

Compile: g++ -std=c++11 interchange.cpp -O3 -floop-interchange -o inter.out -lpapi

L1 DCM : **99903741**

Loop interchange

```
1  #define PAPI_ERROR_CHECK(X) \
2          if((X)!=PAPI_OK) std::cerr<<"Error \n";
3  void kernel(float* in_A, float* in_B)
4  {
5      CAST_TO_MAT(A, in_A, MAT_T);
6      CAST_TO_MAT(B, in_B, MAT_T);
7
8      for (int j = 1; j < 10000 - 1; ++j)
9      {
10         for (int i = 1; i < 5000 - 1; ++i)
11         {
12             A[i][j] = B[i + 1][j - 1] * 2;
13         }
14     }
15 }
```

Note: the __restrict__ qualifier

```
1  #define PAPI_ERROR_CHECK(X) \
2          if((X)!=PAPI_OK) std::cerr<<"Error \n";
3  void kernel(float* __restrict__ in_A,
4              float* __restrict__ in_B)
5  {
6      CAST_TO_MAT(A, in_A, MAT_T);
7      CAST_TO_MAT(B, in_B, MAT_T);
8
9      for (int j = 1; j < 10000 - 1; ++j)
10     {
11         for (int i = 1; i < 5000 - 1; ++i)
12         {
13             A[i][j] = B[i + 1][j - 1] * 2;
14         }
15     }
16 }
```

Compile: g++ -std=c++11 interchange.cpp -O3 -floop-interchange -o inter.out -lpapi

L1 DCM : **99903741**

L1 DCM : **6195738**

Loop fusion

```
#define PAPI_ERROR_CHECK(X) \
        if((X)!=PAPI_OK) std::cerr<<"Error \n";
void kernel(float* in_A, float* in_B, float *in_C)
{
    CAST_TO_MAT(A, in_A, MAT_T);
    CAST_TO_MAT(B, in_B, MAT_T);
    CAST_TO_MAT(C, in_C, MAT_T);

    for (int i = 1; i < 5000 - 1; ++i)
    {
        for (int j = 1; j < 10000 - 1; ++j)
        {
            A[i][j] = B[i + 1][j - 1] * 2;
        }
    }

    for (int i = 1; i < 5000 - 1; ++i)
    {
        for (int j = 1; j < 10000 - 1; ++j)
        {
            C[i][j] = A[i][j - 1] * 2 + B[i + 1][j] * 4;
        }
    }
}
```

```
#define PAPI_ERROR_CHECK(X) \
        if((X)!=PAPI_OK) std::cerr<<"Error \n";

void kernel(float* in_A, float* in_B, float *in_C)
{
    CAST_TO_MAT(A, in_A, MAT_T);
    CAST_TO_MAT(B, in_B, MAT_T);
    CAST_TO_MAT(C, in_C, MAT_T);

    for (int i = 1; i < 5000 - 1; ++i)
    {
        for (int j = 1; j < 10000 - 1; ++j)
        {
            A[i][j] = B[i + 1][j - 1] * 2;
            C[i][j] = A[i][j - 1] * 2 + B[i + 1][j] * 4;
        }
    }
}
```

Compile: g++ -std=c++11 fusion.cpp -O3 -o fusion.out -lpapi

L1 DCM : **15510621**

L1 DCM : **9287599**

Any other benefit?

Loop fusion

```
#define PAPI_ERROR_CHECK(X) \
        if((X)!=PAPI_OK) std::cerr<<"Error \n";
void kernel(float* in_A, float* in_B, float *in_C)
{
    CAST_TO_MAT(A, in_A, MAT_T);
    CAST_TO_MAT(B, in_B, MAT_T);
    CAST_TO_MAT(C, in_C, MAT_T);

    for (int i = 1; i < 5000 - 1; ++i)
    {
        for (int j = 1; j < 10000 - 1; ++j)
        {
            A[i][j] = B[i + 1][j - 1] * 2;
        }
    }

    for (int i = 1; i < 5000 - 1; ++i)
    {
        for (int j = 1; j < 10000 - 1; ++j)
        {
            C[i][j] = A[i][j - 1] * 2 + B[i + 1][j] * 4;
        }
    }
}
```

```
#define PAPI_ERROR_CHECK(X) \
        if((X)!=PAPI_OK) std::cerr<<"Error \n";

void kernel(float* in_A, float* in_B, float *in_C)
{
    CAST_TO_MAT(A, in_A, MAT_T);
    CAST_TO_MAT(B, in_B, MAT_T);
    CAST_TO_MAT(C, in_C, MAT_T);

    for (int i = 1; i < 5000 - 1; ++i)
    {
        for (int j = 1; j < 10000 - 1; ++j)
        {
            A[i][j] = B[i + 1][j - 1] * 2;
            C[i][j] = A[i][j - 1] * 2 + B[i + 1][j] * 4;
        }
    }
}
```

Compile: g++ -std=c++11 fusion.cpp -O3 -o fusion.out -lpapi

L1 DCM : **15510621**
TOT INS : **213103593**
BR INS : **25062200**

L1 DCM : **9287599**
TOT INS : **163043666**
BR INS : **12532258**

Cache line size

Objective: Determine cache line size by experimentation

Available: load/store count

- Design a program to find the cache line size by experimentation

Cache line size

```
1 constexpr int M = std::pow(10, 8);
2
3 void kernel(float* A, float *B, float *C)
4 {
5     for (int i = 0; i < M; ++i)
6     {
7         A[i] = B[i] + C[i] * 2.3;
8     }
9 }
```

L2 STM : **6405219**

//sizeof(float) = 4

#writes = 10^8
#bytes written = #writes * sizeof(float)
= 10^8 * 4
line size = #bytes written / #store misses
= (10^8 * 4) / **6405219**
= 62.45
~ 64 bytes

Examining cache details

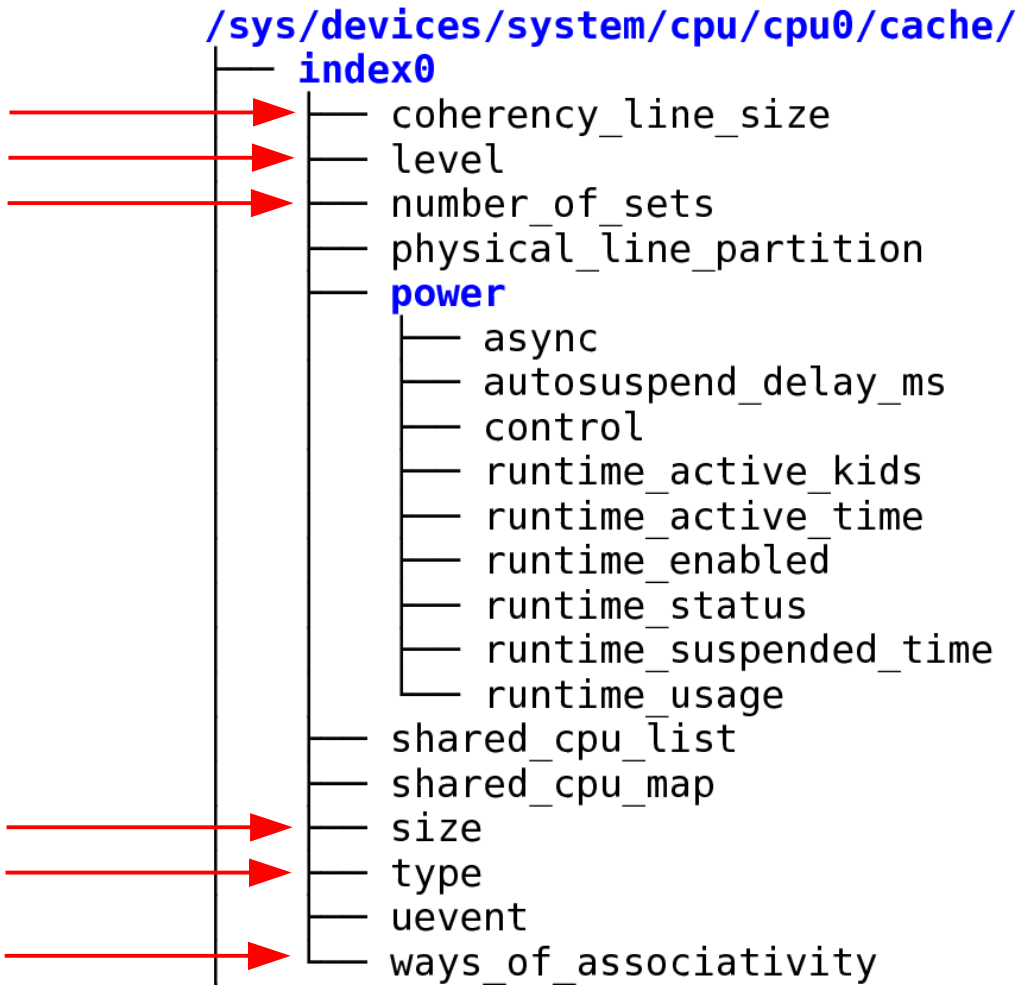
The number of caches, type (instruction/data), associativity etc. can checked by reading the files in

/sys/devices/system/cpu/cpu*/cache/index*/*

eg:

```
cat /sys/devices/system/cpu/cpu0/cache/index3/coherency_line_size
64
```

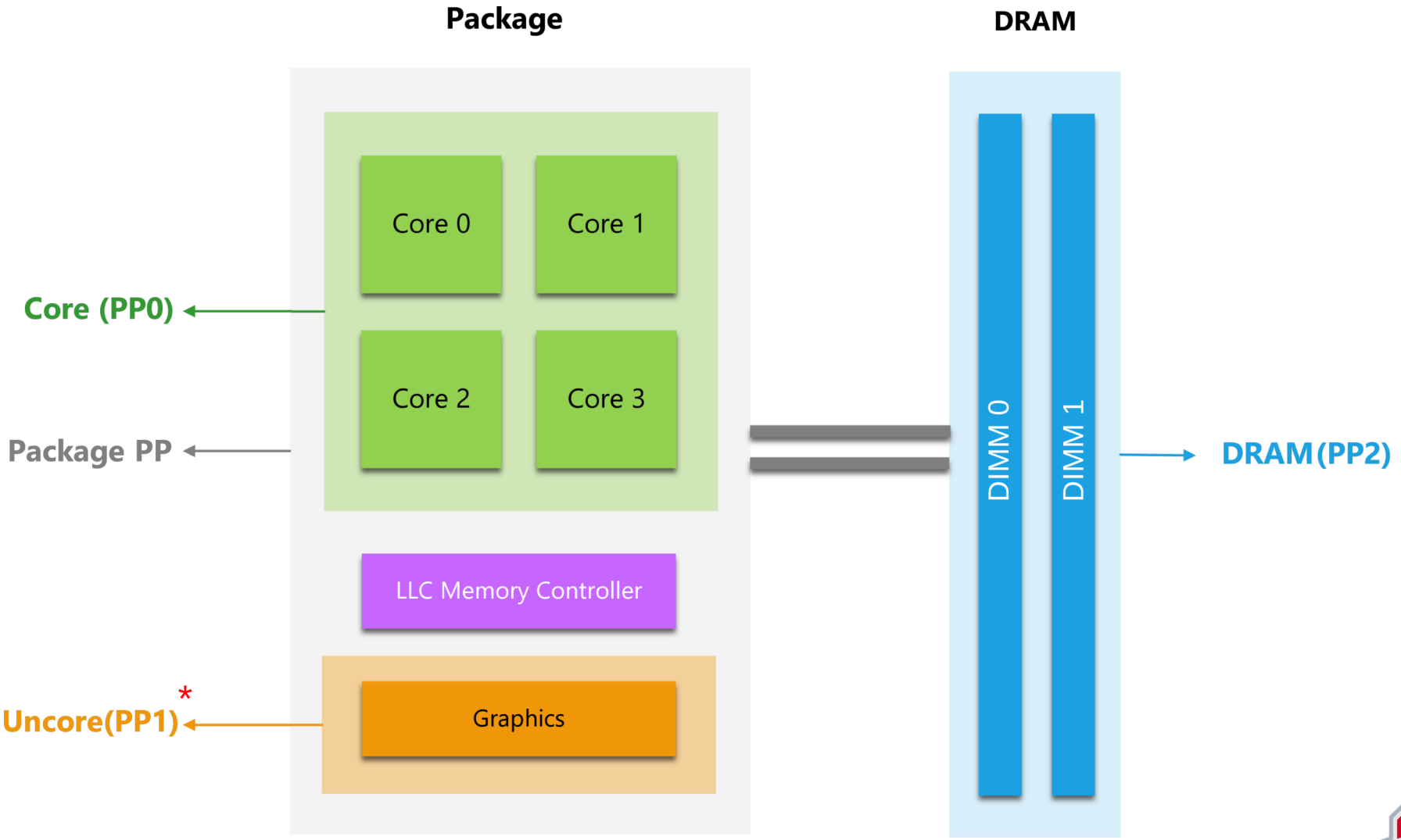
Examining cache details



Power Measurement

- RAPL
  - Running Average Power Limit
  - Perf can read RAPL counters
  - Also accessible by sysfs
- Related terms
  - TDP - Thermal Design Power
  - PP – Power Plane
  - package
  - core (PP0)
  - uncore (PP1)
  - dram (PP2)

Power Measurement



Examining RAPL sysfs

The details of RAPL of sysfs can be found by examining

/sys/class/powercap/intel-rapl/intel-rapl:*/intel-rapl:*:*

Power Measurement

```
/sys/class/powercap/intel-rapl/
├── enabled
├── intel-rapl:0
│   ├── constraint_0_max_power_uw
│   ├── constraint_0_name
│   ├── constraint_0_power_limit_uw
│   ├── constraint_0_time_window_us
│   ├── constraint_1_max_power_uw
│   ├── constraint_1_name
│   ├── constraint_1_power_limit_uw
│   ├── constraint_1_time_window_us
│   ├── device -> ../../intel-rapl
│   ├── enabled
│   ├── energy_uj
│   ├── intel-rapl:0:0
│   │   ├── constraint_0_max_power_uw
│   │   ├── constraint_0_name
│   │   ├── constraint_0_power_limit_uw
│   │   ├── constraint_0_time_window_us
│   │   ├── device -> ../../intel-rapl:0
│   │   ├── enabled
│   │   ├── energy_uj
│   │   ├── max_energy_range_uj
│   │   ├── name
```

package id

eg: Pakage Power Plane`

package id

eg: 2183987317 30

PP id

.
.
.

RAPL

```
1  #define PATH "/sys/class/powercap/intel-rapl/intel-rapl:0/intel-rapl:0:0"
2  int main(int argc, char *argv[])
3  {
4      std::ifstream rapl_sysfs;
5      unsigned long long int before, after;
6
7      /// Read current energy value before running code
8      rapl_sysfs.open(PATH"/energy_uj");
9      rapl_sysfs >> before;
10     rapl_sysfs.close();
11
12     /// Run the code
13     sleep(1);
14
15     /// Read current energy value after running code
16     rapl_sysfs.open(PATH"/energy_uj");
17     rapl_sysfs >> after;
18     rapl_sysfs.close();
19
20     std::cout << "Energy consumed core = "
21               << (after - before) / std::pow(10, 6) << "J\n";
22
23     return 0;
24 }
```