Multi-core Performance Analysis

HPC Computation

Performance Analysis

- Compiler Feedback
- HWPC Data
- Load Balance

Compiler Feedback

- Before optimizing code, it’s critical to know what the compiler does to your code
  - Loop optimizations
  - Vectorization
  - Prefetching
  - …
- Equally important to what the compiler does is what it doesn’t do, and why
  - Data dependencies
  - Misplaced branches
  - Unknown loop counts
  - …

Enabling Compiler Feedback

- Portland Group
  - `Minfo=all`
  - `Mneginfo`
  - `Minfo=ccff` (Common Compiler Feedback Format)
- Cray
  - `rm` (Fortran)
  - `hlist=m` (C/C++)
- Intel
  - `vec-report1`
- Pathscale
  - `LNO:simd_verbose=ON:vintr_verbose=ON:prefetch_verbose=ON`
- GNU
  - `ftree-vectorizer-verbose=1`

Compiler Feedback Examples: PGI

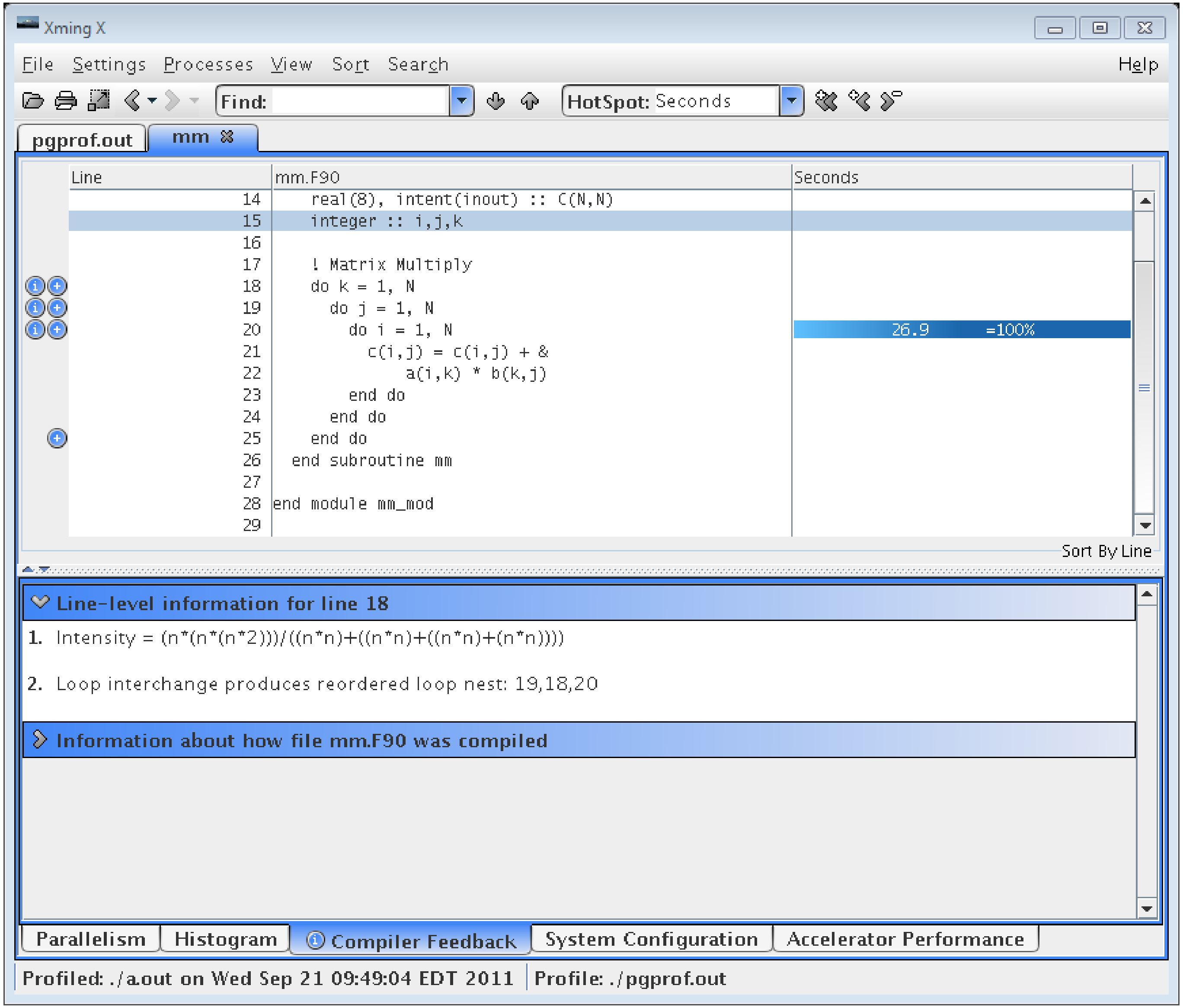
```
! Matrix Multiply
do k = 1, N
  do j = 1, N
    do i = 1, N
      c(i,j) = c(i,j) + &
           a(i,k)*b(k,j)
    end do
  end do
end do
```

```
mm:
   18, Loop interchange
  produces reordered loop
  nest: 19,18,20
   20, Generated 3
  alternate loops for the
  loop
       Generated vector
  sse code for the loop
       Generated 2
  prefetch instructions
  for the loop
```

PGI CCFF Usage

```
ftn -fast -Minfo=all,ccff -Mneginfo -Mprof=ccff
  mm.F90
pgcollect ./a.out
pgprof ./a.out
```

CCFF in PGProf



```
14     real(8), intent(inout) :: C(N,N)
15     integer :: i,j,k
16
17     ! Matrix Multiply
18     do k = 1, N
19       do j = 1, N
20         do i = 1, N                      26.9   =100%
21           c(i,j) = c(i,j) + &
22               a(i,k) * b(k,j)
23         end do
24       end do
25     end do
26   end subroutine mm
27
28 end module mm_mod
29
```

Line-level information for line 18

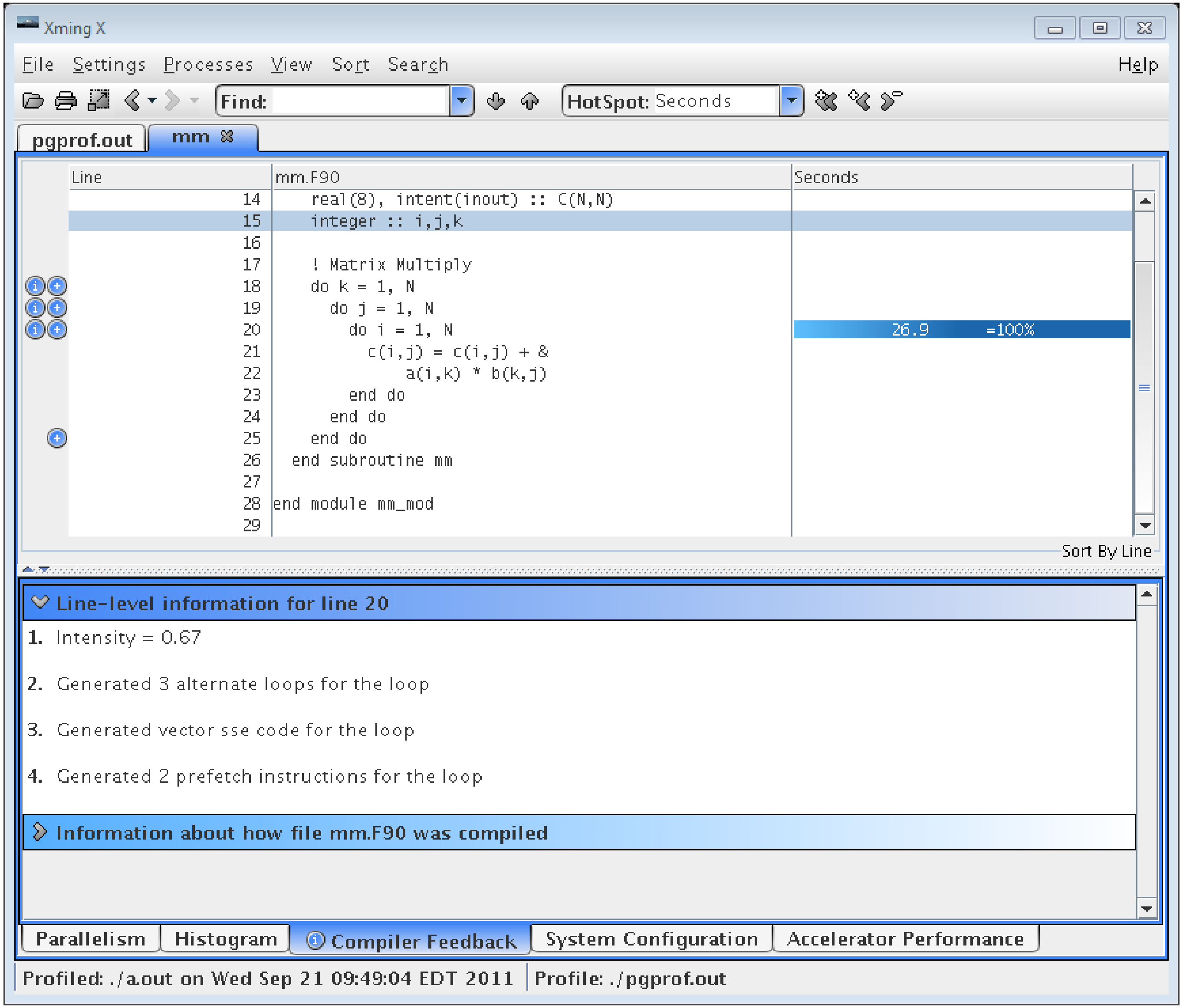
1. Intensity = (n*(n*(n*2)))/((n*n)+((n*n)+((n*n)+(n*n))))

2. Loop interchange produces reordered loop nest: 19,18,20

Information about how file mm.F90 was compiled

Profiled: ./a.out on Wed Sep 21 09:49:04 EDT 2011 | Profile: ./pgprof.out

CCFF in PGProf (cont.)



```
14      real(8), intent(inout) :: C(N,N)
15      integer :: i,j,k
16
17      ! Matrix Multiply
18      do k = 1, N
19        do j = 1, N
20          do i = 1, N                         26.9   =100%
21            c(i,j) = c(i,j) + &
22                a(i,k) * b(k,j)
23          end do
24        end do
25      end do
26    end subroutine mm
27
28  end module mm_mod
29
```

Line-level information for line 20

1. Intensity = 0.67
2. Generated 3 alternate loops for the loop
3. Generated vector sse code for the loop
4. Generated 2 prefetch instructions for the loop

Information about how file mm.F90 was compiled

Profiled: ./a.out on Wed Sep 21 09:49:04 EDT 2011 | Profile: ./pgprof.out

Compiler Feedback Examples: Cray

```
18.  ib------------<     do k = 1, N
19.  ib ibr4-------<       do j = 1, N
20.  ib ibr4 Vbr4--<         do i = 1, N
21.  ib ibr4 Vbr4              c(i,j) = c(i,j) + &
22.  ib ibr4 Vbr4                  a(i,k) * b(k,j)
23.  ib ibr4 Vbr4-->         end do
24.  ib ibr4------->       end do
25.  ib------------>     end do
```

i – interchanged
b – blocked
r – unrolled
V - Vectorized

```
ftn-6007 ftn: SCALAR File = mm.F90, Line = 18
  A loop starting at line 18 was interchanged with the loop starting at line 19.
ftn-6254 ftn: VECTOR File = mm.F90, Line = 18
  A loop starting at line 18 was not vectorized because a recurrence was found on "C" at line
    21.
ftn-6049 ftn: SCALAR File = mm.F90, Line = 18
  A loop starting at line 18 was blocked with block size 32.
ftn-6294 ftn: VECTOR File = mm.F90, Line = 19
  A loop starting at line 19 was not vectorized because a better candidate was found at line
    20.
ftn-6049 ftn: SCALAR File = mm.F90, Line = 19
  A loop starting at line 19 was blocked with block size 8.
ftn-6005 ftn: SCALAR File = mm.F90, Line = 19
  A loop starting at line 19 was unrolled 4 times.
ftn-6049 ftn: SCALAR File = mm.F90, Line = 20
  A loop starting at line 20 was blocked with block size 256.
ftn-6005 ftn: SCALAR File = mm.F90, Line = 20
  A loop starting at line 20 was unrolled 4 times.
ftn-6204 ftn: VECTOR File = mm.F90, Line = 20
  A loop starting at line 20 was vectorized.
```

Compiler Feedback Examples: Pathscale

```
(mm.F90:20) Vectorization is not likely to be beneficial (try -
   LNO:simd=2 to vectorize it). Loop was not vectorized.
(mm.F90:20) Vectorization is not likely to be beneficial (try -
   LNO:simd=2 to vectorize it). Loop was not vectorized.
(mm.F90:20) Vectorization is not likely to be beneficial (try -
   LNO:simd=2 to vectorize it). Loop was not vectorized.
(mm.F90:20) Vectorization is not likely to be beneficial (try -
   LNO:simd=2 to vectorize it). Loop was not vectorized.
(mm.F90:19) Generated 40 prefetch instructions for this loop

=== After adding -LNO:simd=2 ===

(mm.F90:20) Loop has too many loop invariants. Loop was not
   vectorized.
(mm.F90:20) LOOP WAS VECTORIZED.
(mm.F90:20) LOOP WAS VECTORIZED.
(mm.F90:20) LOOP WAS VECTORIZED.
(mm.F90:19) Generated 52 prefetch instructions for this loop
```

Compiler Feedback Examples: Intel

```
mm.F90(20): (col. 9) remark: LOOP WAS VECTORIZED.
mm.F90(20): (col. 9) remark: LOOP WAS VECTORIZED.
mm.F90(20): (col. 9) remark: LOOP WAS VECTORIZED.
```

Compiler Feedback Examples: GNU

```
mm.F90:20: note: LOOP VECTORIZED.
mm.F90:11: note: vectorized 1 loops in function.
```

Gathering Runtime Performance Data

- Performance data can be gathered in numerous ways with a range of detail and intrusiveness
  - Sampling - Snapshot of data collected periodically - very light weight
  - User timers - User inserts timers at logical places - slightly heavier, very intrusive to code
  - Code instrumentation - Tool inserts instrumentation automatically into the code

- Degrees of detail
  - Sampling - high level overview, low details
  - Profiling - summation over time, more detailed
  - Tracing - record of events over time, very detailed and expensive

CrayPAT Automatic Performance Analysis (APA)

- CrayPAT provides a mechanism for guiding user experiments, known as APA
- User first makes lightweight, sample-based run
- Data from initial run is used to suggest appropriate parts of code for gathering more detailed information
  - Attempts to exclude routines that would add overhead and focus on routines that are likely to be important

Important Runtime Data

- Time spent in important routines, libraries, and loop nests
- Hardware Performance Counters (HWPC)
- Load imbalance data
- Communication
  - Time
  - Routines
  - Message sizes
- I/O Data

Sampling Output (Table 1)

```
Notes for table 1:
...
Table 1:  Profile by Function
 Samp % | Samp |  Imb.  |   Imb.  |Group
        |      |  Samp  | Samp %  | Function
        |      |        |         |  PE='HIDE'

 100.0% |  775 |    -- |     -- |Total
|------------------------------------------
|  94.2% |  730 |    -- |     -- |USER
||-----------------------------------------
||  43.4% |  336 |  8.75 |   2.6% |mlwxyz_
||  16.1% |  125 |  6.28 |   4.9% |half_
||   8.0% |   62 |  6.25 |   9.5% |full_
||   6.8% |   53 |  1.88 |   3.5% |artv_
||   4.9% |   38 |  1.34 |   3.6% |bnd_
||   3.6% |   28 |  2.00 |   6.9% |currenf_
||   2.2% |   17 |  1.50 |   8.6% |bndsf_
||   1.7% |   13 |  1.97 |  13.5% |model_
||   1.4% |   11 |  1.53 |  12.2% |cfl_
||   1.3% |   10 |  0.75 |   7.0% |currenh_
||   1.0% |    8 |  5.28 |  41.9% |bndbo_
||   1.0% |    8 |  8.28 |  53.4% |bndto_
||=========================================
|   5.4% |   42 |    -- |     -- |MPI
||-----------------------------------------
||   1.9% |   15 |  4.62 |  23.9% |mpi_sendrecv_
||   1.8% |   14 | 16.53 |  55.0% |mpi_bcast_
||   1.7% |   13 |  5.66 |  30.7% |mpi_barrier_
|==========================================
```

Sampling Output (Table 2)

```
Table 2:  Profile by Group, Function, and Line

 Samp % | Samp |  Imb. |   Imb. |Group
        |      |  Samp | Samp % | Function
        |      |       |        |  Source
        |      |       |        |   Line
        |      |       |        |    PE='HIDE'

 100.0% |  777 |    -- |     -- |Total
|---------------------------------------------
|  94.2% |  732 |    -- |     -- |USER
||--------------------------------------------
||  43.4% |  337 |    -- |     -- |mlwxyz
3|        |      |       |        | ldr/mhd3d/src/mlwxyz.f
||||------------------------------------------
4|||   2.1% |   16 |  1.47 |   8.9% |line.39
4|||   2.8% |   22 |  2.25 |   9.7% |line.78
. . .
4|||   1.3% |   10 |  1.72 |  14.8% |line.604
4|||   2.4% |   19 |  0.72 |   3.7% |line.634
||||==========================================
||||==========================================
||  16.1% |  125 |    -- |     -- |half
3|        |      |       |        | ldr/mhd3d/src/half.f
||||------------------------------------------
4|||   5.4% |   42 |  6.41 |  13.8% |line.28
4|||  10.7% |   83 |  5.91 |   6.9% |line.40
||||==========================================
||   8.0% |   62 |    -- |     -- |full
3|        |      |       |        | ldr/mhd3d/src/full.f
||||------------------------------------------
4|||   8.0% |   62 |  6.31 |   9.6% |line.22
||||==========================================
. . .
||============================================
|   5.4% |   42 |    -- |     -- |MPI
||--------------------------------------------
||   1.9% |   15 |  4.62 |  23.9% |mpi_sendrecv_
||||------------------------------------------
||||==========================================
||   1.8% |   14 | 16.53 |  55.0% |mpi_bcast_
||   1.7% |   13 |  5.66 |  30.7% |mpi_barrier_
|=============================================
```

CrayPAT Tracegroup (subset)

adios	Adaptable I/O System API
armci	Aggregate Remote Memory Copy
blas	Basic Linear Algebra subprograms
caf	Co-Array Fortran (Cray CCE compiler only)
chapel	Chapel language compile and runtime library API
hdf5	manages extremely large and complex data collections
heap	dynamic heap
io	includes stdio and sysio groups
lapack	Linear Algebra Package
math	POSIX.1 math functions
mpi	MPI
omp	OpenMP API and runtime library API (CCE and PGI only)
shmem	SHMEM
upc	Unified Parallel C (Cray CCE compiler only)

For a full list, please see **man pat_build**

pat_report: Flat Profile

```
Table 1:  Profile by Function Group and Function

 Time % |       Time |Imb. Time |   Imb. | Calls |Group
        |            |          | Time % |       | Function
        |            |          |        |       |  PE='HIDE'

 100.0% | 104.593634 |       -- |     -- | 22649 |Total
|-------------------------------------------------------------
|  71.0% |  74.230520 |       -- |     -- | 10473 |MPI
||------------------------------------------------------------
||  69.7% |  72.905208 | 0.508369 |   0.7% |   125 |mpi_allreduce_
||   1.0% |   1.050931 | 0.030042 |   2.8% |    94 |mpi_alltoall_
||============================================================
|  25.3% |  26.514029 |       -- |     -- |    73 |USER
||------------------------------------------------------------
||  16.7% |  17.461110 | 0.329532 |   1.9% |    23 |selfgravity_
||   7.7% |   8.078474 | 0.114913 |   1.4% |    48 |ffte4_
||============================================================
|   2.5% |   2.659429 |       -- |     -- |   435 |MPI_SYNC
||------------------------------------------------------------
||   2.1% |   2.207467 | 0.768347 |  26.2% |   172 |mpi_barrier_(sync)
||============================================================
|   1.1% |   1.188998 |       -- |     -- | 11608 |HEAP
||------------------------------------------------------------
||   1.1% |   1.166707 | 0.142473 |  11.1% |  5235 |free
|=============================================================
```

pat_report: Message Stats by Caller

```
Table 4:  MPI Message Stats by Caller

    MPI Msg |MPI Msg |  MsgSz |   4KB<= |Function
      Bytes |  Count |   <16B |  MsgSz | Caller
            |        |  Count |  <64KB |  PE[mmm]
            |        |        |  Count |

 15138076.0 | 4099.4 |  411.6 | 3687.8 |Total
|-------------------------------------------------
| 15138028.0 | 4093.4 |  405.6 | 3687.8 |MPI_ISEND
||------------------------------------------------
||  8080500.0 | 2062.5 |   93.8 | 1968.8 |calc2_
3|            |        |        |        | MAIN_
||||----------------------------------------------
4|||  8216000.0 | 3000.0 | 1000.0 | 2000.0 |pe.0
4|||  8208000.0 | 2000.0 |     -- | 2000.0 |pe.9
4|||  6160000.0 | 2000.0 |  500.0 | 1500.0 |pe.15
||||==============================================
||  6285250.0 | 1656.2 |  125.0 | 1531.2 |calc1_
3|            |        |        |        | MAIN_
||||----------------------------------------------
4|||  8216000.0 | 3000.0 | 1000.0 | 2000.0 |pe.0
4|||  6156000.0 | 1500.0 |     -- | 1500.0 |pe.3
4|||  6156000.0 | 1500.0 |     -- | 1500.0 |pe.5
||||==============================================
. . .
```

Hardware Performance Counters

- All modern CPUs provide have some number of performance counters used during chip design/testing
- These counters can be read by the kernel and tools such as PAPI, CrayPAT, and others to gather runtime data about an application
- Because the CPUs have a limited number of counters, it’s often necessary to make multiple runs to gather all of the performance data of interest

Types of Data

- Native Events
  - Each processor has a large set of events that can be counted
  - Names vary between architectures, manufacturers, and processor families
- PAPI Counters
  - PAPI has several counters, which map to native events so that common metrics, such as FLOP counts can be measured in a portable way
- Derived Metrics
  - Raw counter data is difficult to interpret directly, derived metrics are rates and ratios that allow easier interpretation of data
  - Example: FLOP Rate, Cache Hit/Miss Ratio, etc.

Gathering HWPC Data

- PAPI
  - A portable API, developed at the University of Tennessee for reading HWPC
  - User must explicitly insert API calls to gather and interpret the data
- Tools
  - Most performance tools are able to gather HWPC data with little to no code modification
  - Generally able to display data in an understandable manner

PAT_RT_HWPC=1 (Summary with TLB)

```
  PAPI_TLB_DM  Data translation lookaside buffer misses
  PAPI_L1_DCA  Level 1 data cache accesses
  PAPI_FP_OPS  Floating point operations
  DC_MISS      Data Cache Miss
  User_Cycles  Virtual Cycles
========================================================================
USER
------------------------------------------------------------------------
  Time%                                            98.3%
  Time                                          4.434402 secs
  Imb.Time                                            -- secs
  Imb.Time%                                           --
  Calls                          0.001M/sec       4500.0 calls
  PAPI_L1_DCM                   14.820M/sec     65712197 misses
  PAPI_TLB_DM                    0.902M/sec      3998928 misses
  PAPI_L1_DCA                  333.331M/sec   1477996162 refs
  PAPI_FP_OPS                  445.571M/sec   1975672594 ops
  User time (approx)             4.434 secs  11971868993 cycles  100.0%Time
  Average Time per Call                         0.000985 sec
  CrayPat Overhead : Time         0.1%
  HW FP Ops / User time        445.571M/sec   1975672594 ops   4.1%peak(DP)
  HW FP Ops / WCT              445.533M/sec
  Computational intensity         0.17 ops/cycle    1.34 ops/ref
  MFLOPS (aggregate)           1782.28M/sec
  TLB utilization               369.60 refs/miss   0.722 avg uses
  D1 cache hit,miss ratios       95.6% hits         4.4% misses
  D1 cache utilization (misses)  22.49 refs/miss   2.811 avg hits
========================================================================
```

PAT_RT_HWPC=1

Flat profile data

Hard counts

Derived metrics

PAT_RT_HWPC=2 (L1 and L2 Metrics)

```
=========================================================================
USER
-------------------------------------------------------------------------
  Time%                                              98.3%
  Time                                            4.436808 secs
  Imb.Time                                              -- secs
  Imb.Time%                                             --
  Calls                          0.001M/sec         4500.0 calls
  DATA_CACHE_REFILLS:
    L2_MODIFIED:L2_OWNED:
    L2_EXCLUSIVE:L2_SHARED       9.821M/sec       43567825 fills
  DATA_CACHE_REFILLS_FROM_SYSTEM:
    ALL                         24.743M/sec      109771658 fills
  PAPI_L1_DCM                   14.824M/sec       65765949 misses
  PAPI_L1_DCA                  332.960M/sec     1477145402 refs
  User time (approx)             4.436 secs    11978286133 cycles  100.0%Time
  Average Time per Call                           0.000986 sec
  CrayPat Overhead : Time         0.1%
  D1 cache hit,miss ratios       95.5% hits          4.5% misses
  D1 cache utilization (misses)  22.46 refs/miss    2.808 avg hits
  D1 cache utilization (refills)  9.63 refs/refill   1.204 avg uses
  D2 cache hit,miss ratio        28.4% hits         71.6% misses
  D1+D2 cache hit,miss ratio     96.8% hits          3.2% misses
  D1+D2 cache utilization        31.38 refs/miss    3.922 avg hits
  System to D1 refill           24.743M/sec      109771658 lines
  System to D1 bandwidth      1510.217MB/sec    7025386144 bytes
  D2 to D1 bandwidth           599.398MB/sec    2788340816 bytes
=========================================================================
```

PAT_RT_HWPC=5 (Floating point mix)

```
========================================================================
USER
------------------------------------------------------------------------
  Time%                                      98.4%
  Time                                    4.426552 secs
  Imb.Time                                      -- secs
  Imb.Time%                                     --
  Calls                    0.001M/sec       4500.0 calls
  RETIRED_MMX_AND_FP_INSTRUCTIONS:
    PACKED_SSE_AND_SSE2  454.860M/sec   2013339518 instr
  PAPI_FML_INS           156.443M/sec    692459506 ops
  PAPI_FAD_INS           289.908M/sec   1283213088 ops
  PAPI_FDV_INS             7.418M/sec     32834786 ops
  User time (approx)       4.426 secs  11950955381 cycles  100.0%Time
  Average Time per Call                   0.000984 sec
  CrayPat Overhead : Time   0.1%
  HW FP Ops / Cycles                          0.17 ops/cycle
  HW FP Ops / User time  446.351M/sec   1975672594 ops   4.1%peak(DP)
  HW FP Ops / WCT        446.323M/sec
  FP Multiply / FP Ops                       35.0%
  FP Add / FP Ops                            65.0%
  MFLOPS (aggregate)     1785.40M/sec
========================================================================
```

PAT_RT_HWPC=12 (QC Vectorization)

```
=======================================================================
USER
-----------------------------------------------------------------------
  Time%                                        98.3%
  Time                                      4.434163 secs
  Imb.Time                                        -- secs
  Imb.Time%                                       --
  Calls                       0.001M/sec      4500.0 calls
  RETIRED_SSE_OPERATIONS:
    SINGLE_ADD_SUB_OPS:
    SINGLE_MUL_OPS                                 0 ops
  RETIRED_SSE_OPERATIONS:
    DOUBLE_ADD_SUB_OPS:
    DOUBLE_MUL_OPS          225.224M/sec   998097162 ops
  RETIRED_SSE_OPERATIONS:
    SINGLE_ADD_SUB_OPS:
    SINGLE_MUL_OPS:OP_TYPE                         0 ops
  RETIRED_SSE_OPERATIONS:
    DOUBLE_ADD_SUB_OPS:
    DOUBLE_MUL_OPS:OP_TYPE  445.818M/sec  1975672594 ops
  User time (approx)          4.432 secs 11965243964 cycles  99.9%Time
  Average Time per Call                     0.000985 sec
  CrayPat Overhead : Time      0.1%
=======================================================================
```

Vectorization Example

```
========================================================================
USER / calc2_
------------------------------------------------------------------------
  Time%                                        28.2%
  Time                                      0.600875 secs
  Imb.Time                                  0.069872 secs
  Imb.Time%                                    11.9%
  Calls                       864.9 /sec       500.0 calls
  RETIRED_SSE_OPERATIONS:
    SINGLE_ADD_SUB_OPS:
    SINGLE_MUL_OPS                                 0 ops
  RETIRED_SSE_OPERATIONS:
    DOUBLE_ADD_SUB_OPS:
    DOUBLE_MUL_OPS          369.139M/sec   213408500 ops
  RETIRED_SSE_OPERATIONS:
    SINGLE_ADD_SUB_OPS:
    SINGLE_MUL_OPS:OP_TYPE                         0 ops
  RETIRED_SSE_OPERATIONS:
    DOUBLE_ADD_SUB_OPS:
    DOUBLE_MUL_OPS:OP_TYPE  369.139M/sec   213408500 ops
  User time (approx)          0.578 secs  1271875000 cycles  96.2%Time
```

When compiled with vectorization:

```
========================================================================
USER / calc2_
------------------------------------------------------------------------
  Time%                                        24.3%
  Time                                      0.485654 secs
  Imb.Time                                  0.146551 secs
  Imb.Time%                                    26.4%
  Calls                       0.001M/sec       500.0 calls
  RETIRED_SSE_OPERATIONS:
    SINGLE_ADD_SUB_OPS:
    SINGLE_MUL_OPS                                 0 ops
  RETIRED_SSE_OPERATIONS:
    DOUBLE_ADD_SUB_OPS:
    DOUBLE_MUL_OPS          208.641M/sec   103016531 ops
  RETIRED_SSE_OPERATIONS:
    SINGLE_ADD_SUB_OPS:
    SINGLE_MUL_OPS:OP_TYPE                         0 ops
  RETIRED_SSE_OPERATIONS:
    DOUBLE_ADD_SUB_OPS:
    DOUBLE_MUL_OPS:OP_TYPE  415.628M/sec   205216531 ops
  User time (approx)          0.494 secs  1135625000 cycles  100.0%Time
```

Derived Metrics: Computational Intensity

- What: Computational intensity is the ratio of arithmetic to memory operations
  - FLOPS/(Loads + Stores)
- Why: Memory transactions are very expensive in comparison to FLOPs, low computational intensity means that the ALUs are waiting for data
- Interpretation: Higher is better
  - Poor: < 0.5 FLOPs/reference
  - So-so: ~1.0 FLOPs/reference
  - Good: > 1.0 FLOPs/reference

Derived Metrics: Cache Hit Ratios

- What: The ratio of hits to misses for a given cache level.
  - Cache Hits/Cache Misses
- Why: Cache accesses are significantly faster than memory accesses, ideally once a cache line is loaded it will be reused.
- Interpretation: Higher is better
  - Poor: < 90%
  - So-so: 90% - 95%
  - Good: >95%
  - Different levels of cache may have slightly different thresholds, but these are rough guidelines.

Derived Metrics: FLOP Rates

- What: Ratio of floating point operations to time.
  - Rate: FLOPs/time
  - Percentage: (FLOPs/time) / (Peak FLOP/s)
  - Caution: Every processor family reports flops differently. Is a 128b, packed multiply 1 FLOP, 2 64-bit FLOPs, or 4 32-bit FLOPs?
- Why: Measures how efficiently the code uses the floating point units
- Interpretation:
  - While there is value in measuring % of peak, many people put too much emphasis in obtaining a very high % of peak.
  - In reality time to solution or a domain-specific rate (ie. Simulated years/day, Particles/second, etc.) is a better metric.
  - If you do measure flop rates, 10-20% is typically quite good.
    - Few codes get very high % of peak
    - Most codes run happily below 10%

Derived Metrics: Vectorization

- What: Ratio of vector/packed floating point operation to scalar/unpacked operations
  - This is can be tricky to measure, due to differences in the way CPUs report FLOPs.
  - Example: (FLOPs when compiled with vectorization) / (FLOPs when compiled without vectorization)
- Why: All mainstream CPUs are moving to longer vectors (SSE -> AVX -> ??)
- Interpretation: Higher is better.

Other Derived Metrics

- Depending on architecture, other metrics that may be of interest
  - Balance of Adds to Multiplies
  - % FMA instructions
  - TLB Utilization