BLUE WATERS

SUSTAINED PETASCALE COMPUTING

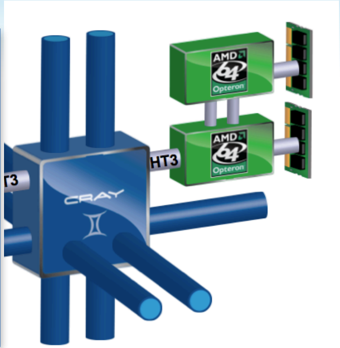
Code profiling with respect to memory using CrayPat

JaeHyuk Kwack & Galen Arnold
SEAS group (help+bw@ncsa.illinois.edu)

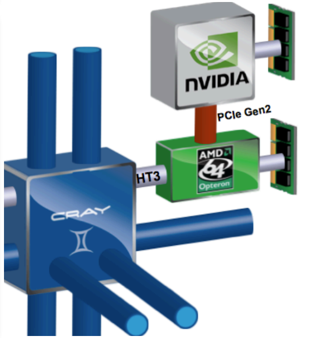
GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

The Blue Waters system

22,640 XE6 compute nodes	
Number of Core Modules	32
Peak Performance	313 Gflops/sec
Memory Size	64 GB per node
Memory Bandwidth (Peak)	102 GB/sec
Interconnect Injection Bandwidth (Peak)	9.6 GB/sec per direction



4,228 XK7 compute nodes with NVIDIA Kepler (GK110) GPUs	
Host Processor	AMD Series 6200 (Interlagos)
Host Processor Performance	156.8 Gflops
Kepler Peak (DP floating point)	1.32 Tflops
Host Memory	32GB, 51 GB/sec
Kepler Memory	6GB GDDR5 capacity, > 180 GB/sec



Introduction of CrayPat

- Performance measurement and analysis
  - Automatic Profiling Analysis
  - Load Imbalance
  - HW counter derived metrics
  - Predefined trace groups provide performance statistics for libraries called by program (blas, lapack, scalapack, petsc, fftw, cuda, hdf5, netcdf, etc.)
  - Support MPI, SHMEM, OpenMP, UPC, CAF, OpenACC
  - Access to network counters
  - Minimal program perturbation
- Limitations
  - Instrumenting only executable binaries
    - Profiling a code using python wrapper cannot be instrumented: Cray is working on it
  - Tracing with many MPI processes yields huge data, while sampling with many MPI ranks is fine

Introduction of CrayPat

- Two modes of use
  - CrayPat-lite for novice users, or convenience
    - % module unload darshan
    - % module load perftools-base perftools-lite
  - CrayPat for in-depth performance investigation and tuning assistance
    - % module unload darshan
    - % module load perftools-base perftools

```
jkwack2@h2ologin2:~> module avail perftools

------------------------------------------------ /opt/cray/modulefiles ------------------------------------------------
perftools/6.2.1            perftools/6.4.3(default)      perftools-base/6.4.6       perftools-lite/6.3.2
perftools/6.2.2            perftools/6.4.6               perftools-lite/6.2.1       perftools-lite/6.4.3(default)
perftools/6.2.3            perftools-base/6.3.0          perftools-lite/6.2.2       perftools-lite/6.4.6
perftools/6.3.0            perftools-base/6.3.1          perftools-lite/6.2.3
perftools/6.3.1            perftools-base/6.3.2          perftools-lite/6.3.0
perftools/6.3.2            perftools-base/6.4.3(default) perftools-lite/6.3.1
jkwack2@h2ologin2:~> _
```

- Compatible programming environments
  - GNU, Intel, PGI and Cray compilers for most of functions
  - Only for Cray compiler: reveal, loop work estimates (with "-h profile_generate"), and so on

How to use CrayPat

- Procedures for instrumentation
  - Building program after updating modules for CrayPat
    - No special flags required in general (e.g., -g is not required)
    - With any optimization flag (e.g., -O0, -O1, -O2, -O3)
  - Instrumenting the original program
    - For the default Automatic Profiling Analysis, % pat_build *my_program*
    - For predefined trace groups, % pat_build -g *tracegroup my_program*
    - For enabling tracing and the CrayPat API, % pat_build -w *my_program*
    - For instrumenting a single function, % pat_build -T *tracefunc my_program*
    - For instrumenting a list of functions, % pat_build -t *tracefile my_program*
    - This produces the instrumented executable *my_program+pat*

How to use CrayPat

- Running the instrumented executable
  - Running it after updating modules for CrayPat
    - % aprun *my_program+pat*
  - This produces a data file *my_program+pat*+PID+node[*s*|*t*].xf
    - *s* for sampling | *t* for tracing
    - For many MPI ranks, the folder *my_program+pat*+PID+node[*s*|*t*] is produced
  - CrayPat Run Time Environment
    - "export PAT_RT_SUMMARY=0" to disable run time summarization before be saved
    - Use "PAT_RT_PERFCTR" for monitoring performance counters (will discuss it later)
- Processing raw performance data and creating report
  - % pat_report *my_program+pat*+PID+node[*s*|*t*].xf
  - This generates an .ap2 file
    - A self-contained archive that can be reopened later using the pat_report command
    - The exported-data file format used by Cray Apprentice2

CrayPat API

- Focusing on a certain region within the code, either to reduce sampling overhead, reduce data file size, or because only a particular region is of interest
- Inserting calls into the program source
- Turning data capture on and off at key points during program execution

- Header files
  - pat_api.h for C
  - pat_apif.h or pat_apif77.h for Fortran

- Compiler macro, CRAY_PAT from the perftools-base module

```
#if defined (CRAY_PAT)
        <CrayPat API calls>
#endif
```

CrayPat API

- API calls in C syntax
  - PAT_record(int *state*)
    - Setting the recording *state* to PAT_STATE_ON or PAT_STATE_OFF
  - PAT_region_begin(int *id*, const char **label*)
  - PAT_region_end(int *id*)
    - Defines the boundaries of a region
    - Regions must be either separate or nested

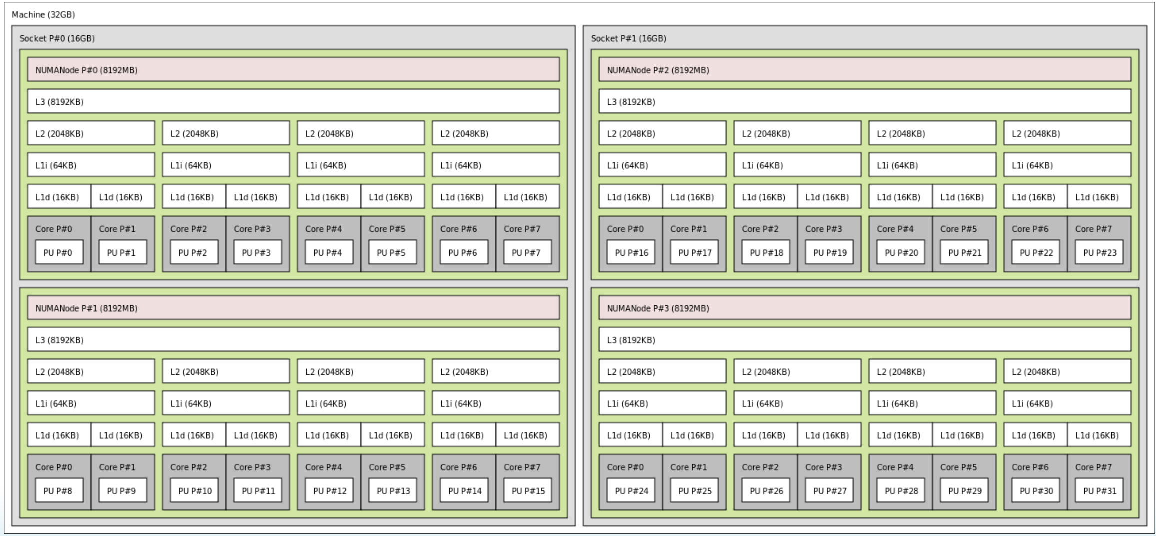
[an example]

```
PAT_record(PAT_STATE_ON);
PAT_region_begin(1, “task_region-1”);
        <tasks;>
PAT_region_end(1);
PAT_region_begin(2, “task_region-2”);
        <tasks;>
PAT_region_end(2);
PAT_record(PAT_STATE_OFF);
```

Profiling w.r.t. memory

- L1 cache
  - 16 KB for data per integer core
  - Latency 3-4 clocks
- L2 cache
  - 2 MB per two integer cores
  - Latency 21 clocks
- L3 cache
  - 16 MB per socket
  - Latency 87 clocks
- DDR memory
  - 64 GB for general XE nodes
  - 128 GB for himem XE nodes
  - 32 GB for general XK nodes
  - 64 GB for himem XK nodes

Memory hierarchy of Bulldozer processors with 32 GB DDR memory



PAT_RT_PERFCTR

- Specifying the performance counters to be monitored during the execution of a program
- More details about hardware counters for AMD Interlargos
  - Four 48-bit performance counters in AMD Interlargos
  - % pat_help counters amd_fam15h

```
jkwack2@h2ologin3:~> pat_help counters amd_fam15h
  Additional topics that may follow "counters amd_fam15h":

    deriv
    groups
    native
    papi

pat_help counters amd_fam15h (.=quit ,=back ^=up /=top ~=search)
=>
```

- PAPI performance counters (i.e.,papi)
  - PAPI_L1_DCA: Level 1 data cache accesses.
  - PAPI_L1_DCM: Level 1 data cache misses.
  - PAPI_L1_DCH: Level 1 data cache hits. (derived)
  - PAPI_FP_OPS: Floating point operations.
  - PAPI_DP_OPS: Floating point operations; optimized to count scaled double precision vector operations.

PAT_RT_PERFCTR

- AMD native performance counters (i.e., native)
  - cray_nb:::L3_CACHE_MISSES
    Number of L3 cache misses for accesses from each core.
    ANY_CORE : Measure on any core
    CORE_0 : Measure on Core1 CORE_1 : Measure on Core1 CORE_2 : Measure on Core2
    CORE_3 : Measure on Core3 CORE_4 : Measure on Core4 CORE_5 : Measure on Core5
    CORE_6 : Measure on Core6 CORE_7 : Measure on Core7 PREFETCH : Count prefetches honly
    READ_BLOCK_ANY : Count any read request READ_BLOCK_MODIFY : Read Block Modify
    READ_BLOCK_EXCLUSIVE : Read Block Exclusive (Data cache read)
    READ_BLOCK_SHARED : Read Block Shared (Instruction cache read)

- Derived metrics (i.e., deriv)
  - D1_D2_cache_hit_ratio: D1+D2 cache hit ratio,
    Computed as min (1, (PAPI_L1_DCA - D1_D2_miss) / PAPI_L1_DCA)
  - Computational_intensity: FP Ops / L1_DCA.
    Computed as fl_pe_sum / PAPI_L1_DCA_pe_sum

PAT_RT_PERFCTR

- Predefined counter groups (i.e., groups)
  - 0: Summary with instructions metrics
    PAPI_TOT_INS, PAPI_FP_OPS, PAPI_L1_DCA, PAPI_L1_DCM
  - 2: L1 and L2 Metrics
    PAPI_L1_DCA, PAPI_L1_DCM, DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:ALL, DATA_CACHE_REFILLS_FROM_NORTHBRIDGE
  - 3: Bandwidth information
    DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:ALL, DATA_CACHE_REFILLS_FROM_NORTHBRIDGE,, OCTWORD_WRITE_TRANSFERS:ALL
  - 23: FP, D1, D2, and TLB
    PAPI_FP_OPS, PAPI_L1_DCA, PAPI_L1_DCM, PAPI_TLB_DM, DATA_CACHE_REFILLS_FROM_NORTHBRIDGE , DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:ALL

Example 1 (profiling cache utilization with CrayPat API)

- Description
  - Compute square of 2D array at each MPI rank:
    OA(i,j) = A(i,j)*A(i,j) with 0 < i,j <= 1024
- Implementation of a CrayPat region in Fortran
  - Adding header
  - Adding a CrayPat region
- Instrumenting the executable
  - % pat_build -w squ_arrays_MPI+O3
- Running the instrumented executable

```
jkwack2@nid00014:~/SEAS/Profiling_Toy_Codes> export PAT_RT_PERFCTR=2
jkwack2@nid00014:~/SEAS/Profiling_Toy_Codes> aprun -n 32 ./squ_arrays_MPI_O3+pat
CrayPat/X:  Version 6.4.6 Revision 7d0d87c  02/20/17 09:52:37
          Number of MPI process              =    32
          Number of rows/columns of the matrix =   1024
Experiment data file written:
/mnt/abc/u/staff/jkwack2/SEAS/Profiling_Toy_Codes/squ_arrays_MPI_O3+pat+9944-32t.xf
Application 410615 resources: utime ~13s, stime ~7s, Rss ~41496, inblocks ~19734, outblocks ~2949
jkwack2@nid00014:~/SEAS/Profiling_Toy_Codes> pat_report squ_arrays_MPI_O3+pat+9944-32t.xf
```

```fortran
program main
use mpi
implicit none
#ifdef CRAYPAT
   include "pat_apif.h"
#endif
```

```fortran
! Turning on Pat_record
#ifdef CRAYPAT
   call PAT_record(PAT_STATE_ON,ierr)
#endif

! Computing square(A)
#ifdef CRAYPAT
   call PAT_region_begin(1,'A(i,j)^2',ierr)
#endif
do i=1,n
   do j=1,n
      OA(i,j) = A(i,j)*A(i,j)
   enddo
enddo
#ifdef CRAYPAT
   call PAT_region_end(1,ierr)
#endif

! Turning off PAT_record
#ifdef CRAYPAT
   call PAT_record(PAT_STATE_OFF,ierr)
#endif
```

Example 1

- CrayPat report

```
==============================================================================
 USER / #1.A(i,j)^2
------------------------------------------------------------------------------
 Time%                                                    67.6%
 Time                                                  0.128852 secs
 Imb. Time                                             0.019737 secs
 Imb. Time%                                               13.7%
 Calls                                7.761 /sec            1.0 calls
 PAPI_L1_DCM                        51.235M/sec       6,601,695 misses
 PAPI_L1_DCA                        21.412M/sec       2,758,946 refs
 DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:
   ALL                              61.468M/sec       7,920,209 fills
 DATA_CACHE_REFILLS_FROM_NORTHBRIDGE 16.263M/sec      2,095,494 fills
 D1 cache hit,miss ratios              0.0% hits         100.0% misses
 D1 cache hit,refill ratio             0.0% hits         100.0% refills
 D1 cache utilization (misses)         0.42 refs/miss      0.05 avg hits
 D1 cache utilization (refills)        0.35 refs/refill    0.04 avg uses
 D2 cache hit,miss ratio              73.5% hits          26.5% misses
 D1+D2 cache hit,miss ratio           36.7% hits          63.3% misses
 D1+D2 cache utilization               1.58 refs/miss      0.20 avg hits
 System to D1 refill                16.263M/sec       2,095,494 lines
 System to D1 bandwidth           992.604MiB/sec    134,111,604 bytes
 D2 to D1 bandwidth             2,759.079MiB/sec    372,781,742 bytes
 Average Time per Call                                 0.128852 secs
 CrayPat Overhead : Time               0.0%
==============================================================================
```

Example 1

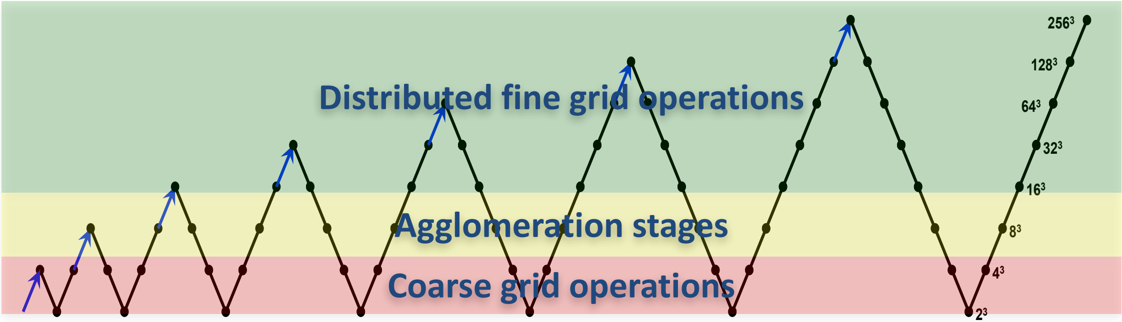
- Updating the loop for stride-1 reference pattern

```
! Turning on Pat_record
#ifdef CRAYPAT
   call PAT_record(PAT_STATE_ON,ierr)
#endif

! Computing square(A) updated
#ifdef CRAYPAT
   call PAT_region_begin(11,'A(i,j)^2 updated ',ierr)
#endif
do j=1,n
   do i=1,n
      OA(i,j) = A(i,j)*A(i,j)
   enddo
enddo
#ifdef CRAYPAT
   call PAT_region_end(11,ierr)
#endif

! Turning off PAT_record
#ifdef CRAYPAT
   call PAT_record(PAT_STATE_OFF,ierr)
#endif
```

```
==============================================================================
 USER / #11.A(i,j)^2 updated
------------------------------------------------------------------------------
 Time%                                                    2.7%
 Time                                                0.005086 secs
 Imb. Time                                           0.002045 secs
 Imb. Time%                                              29.6%
 Calls                               196.616 /sec          1.0 calls
 PAPI_L1_DCM                          29.951M/sec      152,335 misses
 PAPI_L1_DCA                         261.062M/sec    1,327,775 refs
 DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:
   ALL                               105.171M/sec      534,906 fills
 DATA_CACHE_REFILLS_FROM_NORTHBRIDGE  47.189M/sec      240,007 fills
 D1 cache hit,miss ratios              88.5% hits        11.5% misses
 D1 cache hit,refill ratio             59.7% hits        40.3% refills
 D1 cache utilization (misses)          8.72 refs/miss    1.09 avg hits
 D1 cache utilization (refills)         2.48 refs/refill  0.31 avg uses
 D2 cache hit,miss ratio               55.1% hits        44.9% misses
 D1+D2 cache hit,miss ratio            94.9% hits         5.1% misses
 D1+D2 cache utilization               19.43 refs/miss    2.43 avg hits
 System to D1 refill                  47.189M/sec      240,007 lines
 System to D1 bandwidth            2,880.204MiB/sec 15,360,472 bytes
 D2 to D1 bandwidth                3,538.928MiB/sec 18,873,530 bytes
 Average Time per Call                               0.005086 secs
 CrayPat Overhead : Time                0.1%
==============================================================================
```

Example 2 (measuring flop-rates with CrayPat API)

- High Performance Geometric Multi-Grid benchmark
  - HPC performance benchmarking based on full multi-grid(FMG) F-cycle
  - Reporting number of equations solvable per second for three resolutions (i.e., 1h, 2h, and 4h)
- Objective of profiling
  - Measuring flop-rates for three resolutions



Source: Williams (hpgmg.org), HPGMG BoF, SC-16, 2016

Example 2 (measuring flop-rates with CrayPat API)

- Implementation of CrayPat regions in C
  - Add header

```
#ifdef CRAYPAT
#include "pat_api.h"
#endif
```

  - Add CrayPat regions

```
// Adding CrayPat by JaeHyuk Kwack
#ifdef CRAYPAT
PAT_record(PAT_STATE_ON);
#endif
#define DYNAMIC_RANGE 3
double AverageSolveTime[DYNAMIC_RANGE];
for(l=0;l<DYNAMIC_RANGE;l++){
  // if(problem size too small)break;
  #ifdef CRAYPAT
  if(l==0) PAT_region_begin(1,"hpgmg_bench_1h");
  if(l==1) PAT_region_begin(2,"hpgmg_bench_2h");
  if(l==2) PAT_region_begin(3,"hpgmg_bench_4h");
  #endif
  if(l>0)restriction(MG_h.levels[l],VECTOR_F,MG_h.levels[l-1],VECTOR_F,RESTRICT_CELL);
  bench_hpgmg(&MG_h,l,a,b,rtol);
  #ifdef CRAYPAT
  if(l==0) PAT_region_end(1);
  if(l==1) PAT_region_end(2);
  if(l==2) PAT_region_end(3);
  #endif
  AverageSolveTime[l] = (double)MG_h.timers.MGSolve / (double)MG_h.MGSolves_performed;
  if(my_rank==0){fprintf(stdout,"\n\n===== Timing Breakdown ==========================
  MGPrintTiming(&MG_h,l);
}
// Adding CrayPat by JaeHyuk Kwack
#ifdef CRAYPAT
PAT_record(PAT_STATE_OFF);
#endif
```

Example 2 (measuring flop-rates with CrayPat API)

- Instrumenting the executable and running it
  - % pat_build -w hpgmg
  - % export PAT_RT_PERFCTR=23
  - % aprun –n 1024 hpgmg+pat 8 1

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

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

 100.0% | 667.109019 |        -- |    -- |   6.0 | Total
|-------------------------------------------------------------------
|  99.1% | 660.781551 |        -- |    -- |   4.0 | USER
||------------------------------------------------------------------
||  54.8% | 365.506644 |  0.000491 |  0.0% |   1.0 | #1.hpgmg_bench_1h
||  22.4% | 149.585381 | 58.450097 | 28.1% |   1.0 | main
||  12.3% |  82.151491 |  1.380443 |  1.7% |   1.0 | #2.hpgmg_bench_2h
||   9.5% |  63.538036 |  0.022500 |  0.0% |   1.0 | #3.hpgmg_bench_4h
|===================================================================
```

```
Table 4:  Memory High Water Mark by Numa Node

  Process |    HiMem |    HiMem |    HiMem |    HiMem | Numanode
    HiMem |     Numa |     Numa |     Numa |     Numa |  PE=HIDE
 (MBytes) |   Node 0 |   Node 1 |   Node 2 |   Node 3 |
          | (MBytes) | (MBytes) | (MBytes) | (MBytes) |

    452.7 |    120.7 |    111.6 |    110.6 |    109.9 | Total
|-------------------------------------------------------------------
|     115.1 |    108.6 |      2.7 |      2.1 |      1.6 | numanode.0
|     112.8 |      4.4 |    104.5 |      2.2 |      1.7 | numanode.1
|     112.5 |      3.9 |      2.4 |    103.9 |      2.3 | numanode.2
|     112.4 |      3.7 |      2.0 |      2.4 |    104.3 | numanode.3
|===================================================================
```

Example 2

- Region for 1h

```
==========================================================================
 USER / #1.hpgmg_bench_1h
--------------------------------------------------------------------------
 Time%                                                     54.8%
 Time                                                 365.506644 secs
 Imb. Time                                              0.000491 secs
 Imb. Time%                                                 0.0%
 Calls                                  0.003 /sec           1.0 calls
 PAPI_L1_DCM                          57.832M/sec  21,138,096,091 misses
 PAPI_TLB_DM                           0.440M/sec     160,846,785 misses
 PAPI_L1_DCA                       1,267.602M/sec 463,316,889,033 refs
 PAPI_FP_OPS                       1,117.479M/sec 408,445,909,119 ops
 DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:
   ALL                                86.593M/sec  31,650,208,551 fills
 DATA_CACHE_REFILLS_FROM_NORTHBRIDGE   6.377M/sec   2,330,850,470 fills
 HW FP Ops / User time             1,117.479M/sec 408,445,909,119 ops
 Computational intensity                    ops/cycle       0.88 ops/ref
 MFLOPS (aggregate)           1,144,298.24M/sec
 TLB utilization                     2,880.49 refs/miss     5.63 avg uses
 D1 cache hit,miss ratios               95.4% hits          4.6% misses
 D1 cache hit,refill ratio              93.2% hits          6.8% refills
 D1 cache utilization (misses)         21.92 refs/miss      2.74 avg hits
 D1 cache utilization (refills)        14.64 refs/refill    1.83 avg uses
 D2 cache hit,miss ratio                92.6% hits          7.4% misses
 D1+D2 cache hit,miss ratio             99.7% hits          0.3% misses
 D1+D2 cache utilization              297.63 refs/miss     37.20 avg hits
 System to D1 refill                   6.377M/sec   2,330,850,470 lines
 System to D1 bandwidth              389.224MiB/sec 149,174,430,097 bytes
 D2 to D1 bandwidth                4,895.976MiB/sec 1,876,438,917,136 bytes
 Average Time per Call                                365.506644 secs
 CrayPat Overhead : Time                0.0%
==========================================================================
```

Example 2

- Region for 2h

```
===============================================================================
 USER / #2.hpgmg_bench_2h
-------------------------------------------------------------------------------
 Time%                                                    12.3%
 Time                                                 82.151491 secs
 Imb. Time                                             1.380443 secs
 Imb. Time%                                                1.7%
 Calls                                   0.012 /sec         1.0 calls
 PAPI_L1_DCM                           57.165M/sec  4,696,163,776 misses
 PAPI_TLB_DM                            0.413M/sec     33,919,317 misses
 PAPI_L1_DCA                        1,271.030M/sec 104,416,984,071 refs
 PAPI_FP_OPS                        1,057.664M/sec  86,888,685,162 ops
 DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:
   ALL                                 83.716M/sec  6,877,430,703 fills
 DATA_CACHE_REFILLS_FROM_NORTHBRIDGE    6.378M/sec    523,930,239 fills
 HW FP Ops / User time              1,057.664M/sec  86,888,685,162 ops
 Computational intensity                    ops/cycle      0.83 ops/ref
 MFLOPS (aggregate)             1,083,048.07M/sec
 TLB utilization                      3,078.39 refs/miss   6.01 avg uses
 D1 cache hit,miss ratios                 95.5% hits       4.5% misses
 D1 cache hit,refill ratio                93.4% hits       6.6% refills
 D1 cache utilization (misses)            22.23 refs/miss  2.78 avg hits
 D1 cache utilization (refills)           15.18 refs/refill 1.90 avg uses
 D2 cache hit,miss ratio                  92.4% hits       7.6% misses
 D1+D2 cache hit,miss ratio               99.7% hits       0.3% misses
 D1+D2 cache utilization                 291.86 refs/miss  36.48 avg hits
 System to D1 refill                     6.378M/sec    523,930,239 lines
 System to D1 bandwidth                389.258MiB/sec 33,531,535,265 bytes
 D2 to D1 bandwidth                  4,720.388MiB/sec 406,624,029,735 bytes
 Average Time per Call                                 82.151491 secs
 CrayPat Overhead : Time                   0.0%
===============================================================================
```

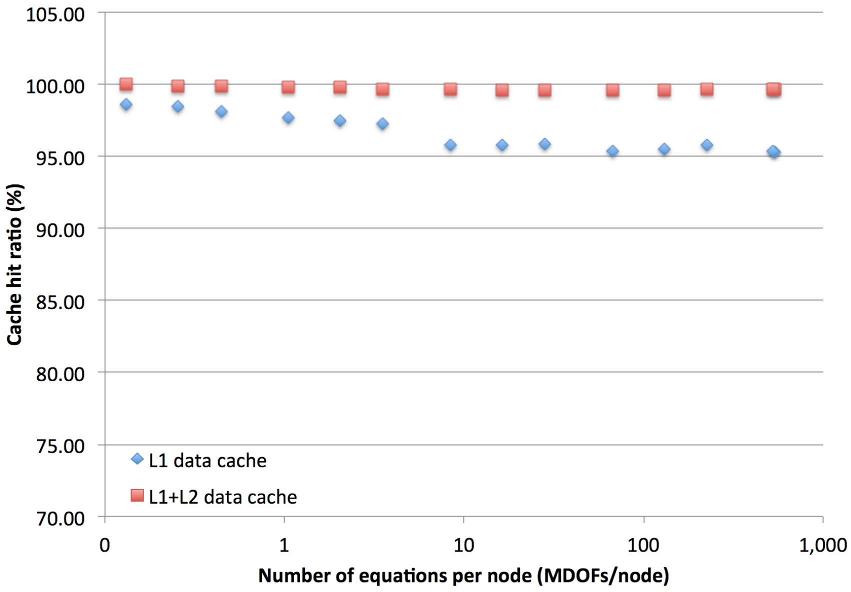
Example 2

- Region for 4h

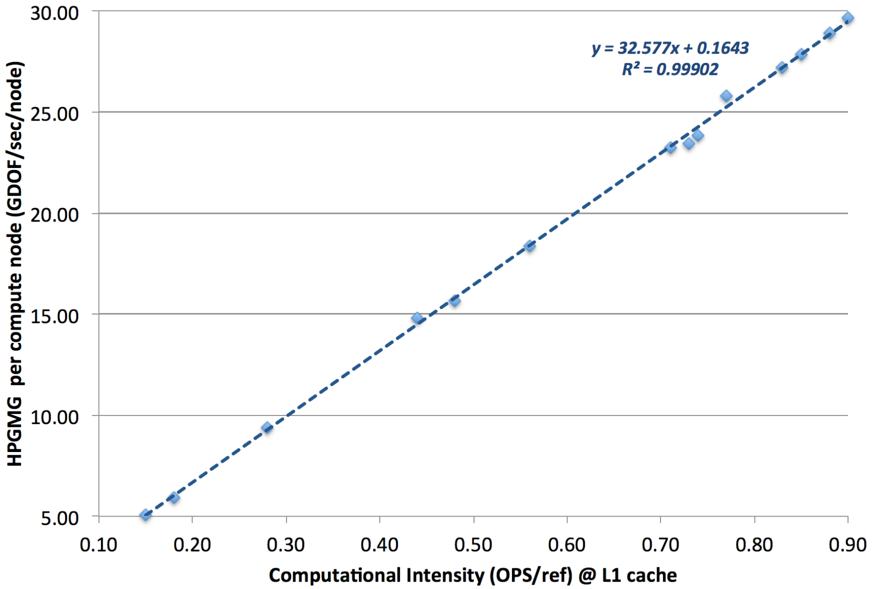
```
==============================================================================
 USER / #3.hpgmg_bench_4h
------------------------------------------------------------------------------
 Time%                                                          9.5%
 Time                                                      63.538036 secs
 Imb. Time                                                  0.022500 secs
 Imb. Time%                                                     0.0%
 Calls                                  0.016 /sec               1.0 calls
 PAPI_L1_DCM                          50.250M/sec      3,192,808,207 misses
 PAPI_TLB_DM                           0.319M/sec         20,268,724 misses
 PAPI_L1_DCA                       1,247.096M/sec     79,238,041,625 refs
 PAPI_FP_OPS                         866.400M/sec     55,049,349,244 ops
 DATA_CACHE_REFILLS_FROM_L2_OR_NORTHBRIDGE:
   ALL                                69.995M/sec      4,447,328,484 fills
 DATA_CACHE_REFILLS_FROM_NORTHBRIDGE   5.680M/sec        360,901,226 fills
 HW FP Ops / User time               866.400M/sec     55,049,349,244 ops
 Computational intensity                    ops/cycle           0.69 ops/ref
 MFLOPS (aggregate)              887,193.52M/sec
 TLB utilization                     3,909.37 refs/miss         7.64 avg uses
 D1 cache hit,miss ratios               96.0% hits              4.0% misses
 D1 cache hit,refill ratio              94.4% hits              5.6% refills
 D1 cache utilization (misses)          24.82 refs/miss         3.10 avg hits
 D1 cache utilization (refills)         17.82 refs/refill       2.23 avg uses
 D2 cache hit,miss ratio                91.9% hits              8.1% misses
 D1+D2 cache hit,miss ratio             99.7% hits              0.3% misses
 D1+D2 cache utilization              305.82 refs/miss         38.23 avg hits
 System to D1 refill                   5.680M/sec        360,901,226 lines
 System to D1 bandwidth              346.685MiB/sec   23,097,678,453 bytes
 D2 to D1 bandwidth                3,925.455MiB/sec  261,531,344,492 bytes
 Average Time per Call                                     63.538036 secs
 CrayPat Overhead : Time                 0.0%
==============================================================================
```

Example 2, after profiling several cases

DOFS/node vs. Cache hit ratio



Computational intensity @ L1 cache vs. HPGMG



Summary

- Procedure for CrayPat instrumentation
  - pat_build
  - pat_report
- CrayPat API
  - Headers
  - pat_regions
- Run-time environments (PAT_RT_PERFCTR)
  - papi
  - native
  - derive
  - groups

QUESTIONS ?

Acknowledgment

This study is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.