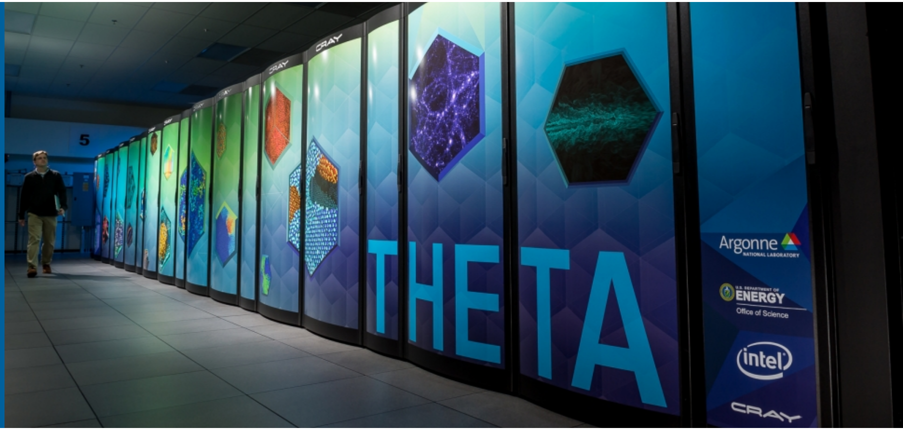
2020 VIRTUAL ALCF COMPUTATIONAL PERFORMANCE WORKSHOP

CRAY PERFORMANCE ANALYSIS TOOLS (CRAYPAT)



JAEHYUK KWACK
Argonne Leadership Computing Facility

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

May 6, 2020

AGENDA

- Overview
- Two modes to use CrayPat
  - "LITE" mode
  - In-depth analysis
- Performance counters
- CrayPat API
- Apprentice2

CRAY PERFORMANCE ANALYSIS TOOLS

- Whole program performance analysis with
  - Novice and advanced user interfaces
  - Support for MPI, SHMEM, OpenMP, UPC, CAF
  - Load imbalance detection
  - HW counter metrics (hit rates, computational intensity, etc.)
  - Observations and inefficiencies
  - Data correlation to user source
- Sampling, tracing with runtime summarization, full trace (timeline) mode available
- Support CCE, Intel and GCC compilers
- Apprentice2 provides visual interface to performance data

TWO MODES OF USE

- CrayPat-lite for novice users, or convenience

- CrayPat for in-depth performance investigation and tuning assistance

- Both offer:
  - Whole program analysis across many nodes
  - Indication of causes of problems
  - Suggestions of modifications for performance improvement

“LITE” MODE

“LITE” MODE

- Load performance tools instrumentation module

```
$ module unload darshan
$ module load perftools-base
$ module load perftools-lite
```

If you use “PrgEnv-intel” module, you will need to load “gcc” module in addition. Your application will use Intel Compilers, but CrayPat still needs some header files from GNU compilers.

- Build program (no modification to makefile)

```
$ make
```

⇨

```
$ a.out (instrumented program)
```

- Run program (no modification to batch script)

```
$ aprun a.out
```

⇨

```
Condensed report to stdout
a.out*.rpt (same as stdout)
a.out*.ap2 files
```

EXAMPLE CRAYPAT-LITE OUTPUT

```
#################################################################
#                                                               #
#            CrayPat-lite Performance Statistics                #
#                                                               #
#################################################################
CrayPat/X:  Version 7.0.4 Revision e00a493  09/12/18 13:16:44
Experiment:                   lite  lite-samples
Number of PEs (MPI ranks):   2,048
Numbers of PEs per Node:        64  PEs on each of  32  Nodes
Numbers of Threads per PE:       1
Number of Cores per Socket:     64
Execution start time:  Tue Apr 30 19:32:14 2019
System name and speed:  nid00340  1.301 GHz (nominal)
Intel Knights Landing CPU  Family:  6  Model: 87  Stepping:  1
DRAM: 192 GiB DDR4-2400 on 1.3 GHz nodes
MCDRAM: 7.2 GHz, 16 GiB available as quad, cache (100% cache)

Avg Process Time:            416.85 secs
High Memory:               76,302.9 MiBytes     37.3 MiBytes per PE
Observed CPU clock boost:     107.7 %
Instr per Cycle:               1.14
Observed CPU cycle rate:       1.38 GHz
I/O Read Rate:             1.996614 MiBytes/sec
I/O Write Rate:            0.512512 MiBytes/sec
```

```
Table 1:  Profile by Function (limited entries shown)
  Samp% |      Samp |    Imb. |  Imb. | Group
        |           |    Samp | Samp% |  Function=[MAX10]
        |           |         |       |   PE=HIDE
 100.0% | 41,447.1 |      -- |    -- | Total
|-----------------------------------------------------------------
|  46.6% | 19,305.8 |      -- |    -- | USER
||----------------------------------------------------------------
||  32.2% | 13,353.9 |   874.1 |  6.1% | genral_
||   6.2% |  2,561.7 |   217.3 |  7.8% | xyzint_
||   3.9% |  1,606.8 |   140.2 |  8.0% | rt123_
||   3.1% |  1,270.5 |   176.5 | 12.2% | build_abket_
||================================================================
|  45.5% | 18,863.6 |      -- |    -- | BLAS
||----------------------------------------------------------------
||  22.7% |  9,425.9 |   679.1 |  6.7% | gotoblas_dgemm_kernel_knl
||  12.8% |  5,294.0 |   428.0 |  7.5% | gotoblas_dgetrf_single_knl
||   3.9% |  1,622.9 |   276.1 | 14.5% | gotoblas_dlaswp_plus_knl
||   1.8% |    765.3 |    88.7 | 10.4% | gotoblas_dgemv_n_knl
||   1.6% |    646.2 |   262.8 | 28.9% | gotoblas_dgemm_itcopy_knl
||================================================================
|   6.3% |  2,627.8 |      -- |    -- | MPI
||----------------------------------------------------------------
||   6.1% |  2,537.6 | 1,619.4 | 39.0% | MPI_ALLREDUCE
||================================================================
|   1.5% |    629.2 |      -- |    -- | ETC
|=================================================================
```

IDENTIFY HIGH TIME CONSUMING AREAS

```
Table 2:  Profile by Group, Function, and Line (limited entries shown)

  Samp% |     Samp |    Imb. |  Imb. | Group
        |          |    Samp | Samp% |  Function=[MAX10]
        |          |         |       |   Source
        |          |         |       |    Line
        |          |         |       |     PE=HIDE

 100.0% | 41,447.1 |      -- |    -- | Total
|----------------------------------------------------------------------
|  46.6% | 19,305.8 |      -- |    -- | USER
||---------------------------------------------------------------------
||  32.2% | 13,353.9 |      -- |    -- | genral_
3|        |          |         |       |  vsvb.f90
||||-------------------------------------------------------------------
4|||   1.6% |    645.6 |    88.4 | 12.0% | line.3729
4|||   1.3% |    550.5 |    90.5 | 14.1% | line.3818
4|||   1.1% |    457.2 |   100.8 | 18.1% | line.3829
4|||   2.2% |    929.3 |   145.7 | 13.6% | line.3840
4|||   1.2% |    498.7 |    79.3 | 13.7% | line.3862
4|||   2.2% |    908.9 |   155.1 | 14.6% | line.3867
```

MPI RANK REORDERING

```
================  Observations and suggestions  ======================

MPI Grid Detection:

    There appears to be point-to-point MPI communication in a 35 X 60
    grid pattern. The 20.3% of the total execution time spent in MPI
    functions might be reduced with a rank order that maximizes
    communication between ranks on the same node. The effect of
    several rank orders is estimated below.

    A file named MPICH_RANK_ORDER.Grid was generated along with this
    report and contains usage instructions and the Custom rank order
    from the following table.


        Rank     On-Node     On-Node   MPICH_RANK_REORDER_METHOD
       Order    Bytes/PE   Bytes/PE%
                            of Total
                            Bytes/PE


      Custom   4.050e+09      34.77%  3
         SMP   2.847e+09      24.45%  1
        Fold   1.025e+08       0.88%  2
  RoundRobin   6.098e+01       0.00%  0
```

- Maximize on-node communications and minimize inter-node communications
- "Observations" in output helps detect point-to-point MPI communication and suggests ways to reorder MPI ranks to reduce inter-node communication
- In addition to other files, a MPICH_RANK_ORDER is produced in the subdirectory
- If CrayPat-lite decides work is well balanced across the nodes, it will not be produced

MEMORY TRAFFICS AND FILE I/O

Table 3: Memory Bandwidth by Numanode (limited entries shown)

Memory Traffic GBytes	DDR Memory Traffic GBytes	MCDRAM Memory Traffic GBytes	Thread Time	Memory Traffic GBytes / Sec	Numanode Node Id=[max3,min3] PE=HIDE
33,445	153.02	33,292	417.182412	80.17	numanode.0
33,306	14.33	33,292	417.140768	79.84	nid.4022
33,292	0.16	33,292	417.120838	79.81	nid.345
33,285	26.95	33,258	417.128666	79.80	nid.346
32,867	0.19	32,867	417.100249	78.80	nid.343
32,811	14.82	32,797	417.133453	78.66	nid.3734

Table 5: File Input Stats by Filename (limited entries shown)

Avg Read Time per Reader Rank	Avg Read MiBytes per Reader Rank	Read Rate MiBytes/sec	Number of Reader Ranks	Avg Reads per Reader Rank	Bytes/ Call	File Name PE=HIDE
0.405698	0.079402	0.195717	1	83,259.0	1.00	stdin
0.000023	0.000023	1.001237	32	3.1	8.00	_Unkno_

Table 6: File Output Stats by Filename (limited entries shown)

Avg Write Time per Writer Rank	Avg Write MiBytes per Writer Rank	Write Rate MiBytes/sec	Number of Writer Ranks	Avg Writes per Writer Rank	Bytes/ Call	File Name PE=HIDE
0.152658	0.064385	0.421762	1	1357.0	49.75	orbitals
0.000218	0.000458	2.095105	1	10.0	48.00	stdout
0.000092	0.000469	5.107664	32	15.4	32.00	_Unkno_

```
Program invocation:  /home/user/test

For a complete report with expanded tables and notes, run:
  pat_report /gpfs/mira-home/user/test+42377-340s

For help identifying callers of particular functions:
  pat_report -O callers+src /gpfs/mira-home/user/test+42377-340s
To see the entire call tree:
  pat_report -O calltree+src /gpfs/mira-home/user/test+42377-340s

For interactive, graphical performance analysis, run:
  app2 /gpfs/mira-home/user/test+42377-340s

================  End of CrayPat-lite output  ========================
```

DATA FROM PAT_REPORT

- Default reports are intended to be useful for most applications

- Don't need to rerun program to get more detailed data

- Different aggregations, or levels of information available
  - Get fine-grained thread-imbalance information for OpenMP program
    - `$ pat_report –s pe=ALL –s th=ALL`

MORE IN-DEPTH ANALYSIS AND BOTTLENECK DETECTION

HOW TO USE CRAYPAT

- Update modules and build your application

```
$ module unload darshan
$ module load perftools-base perftools
$ make
```

- Instrumentation example:

```
$ pat_build my_program
```

- Run program

```
$ aprun my_program+pat
```

- Create report

```
$ pat_report my_program.xf > my_report
```

If you use “PrgEnv-intel” module, you will need to load “gcc” module in addition. Your application will use Intel Compilers, but CrayPat still needs some header files from GNU compilers.

PAT_BUILD

- No special flags required in general (e.g., -g is not required)
- With any optimization flag (e.g., -O0, -O1, -O2, -O3)

- Instrumentation options
  - 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`

SAMPLE VS TRACE

- Sample mode
  - Checks program counter and call stack 100 times per second
  - Minimal effect on execution
- Trace mode
  - Trace code inserted
  - Other information such as MPI message size
  - Cray compiler only – loops and loop lengths
  - Trace of small routines affects runtime
- Trace routines from sample run
  - Two step approach – sample, and then trace

PREDEFINED TRACE WRAPPERS (-g tracegroup)

▪ blas	Basic Linear Algebra subprograms
▪ caf	Co-Array Fortran (Cray CCE compiler only)
▪ hdf5	manages extremely large data collection
▪ heap	dynamic heap
▪ io	includes stdio and sysio group
▪ lapack	Linear Algebra Package
▪ math	ANSI math
▪ mpi	MPI
▪ omp	OpenMP API
▪ pthreads	POSIX threads
▪ shmem	SHMEM
▪ sysio	I/O system calls
▪ system	system calls
▪ upc	Unified Parallel C (Cray CCE compiler only)

For a full list, please see **pat_build(1)** man page

CONTROL DATA COLLECTION W/ RUNTIME OPTIONS

- Runtime controlled through **PAT_RT_XXX** environment variables

- Examples of control
  - Enable full trace
  - Change number of data files created
  - Enable collection of CPU, network or power counter events
  - Enable tracing filters to control trace file size (max threads, max call stack depth, etc.)

- Cray supports raw counters, derived metrics and thresholds for:
  - Processor (core and uncore)
  - Network

PERFORMANCE COUNTERS OVERVIEW

Set PAT_RT_PERFCTR environment variable

- papi counters (see via pat_help or papi_avail on a compute node)
- 132 native counters (see via pat_help or papi_native_avail on a compute node)
- 41 derived counters (see pat_help)
- 6 predefined groups (see pat_help)
  - Groups together counters for experiments
    - 0: Cycles and instructions with LLC misses and references
    - _FIXED: Cycles and instructions always available
    - hbm: L2 cache misses and FE stall cycles
    - mem_bw: memory bandwidth dram and mcdran
    - mem_bw_dram: dram bandwidth near and far
    - mem_bw_mcdram: mcdram bandwidth near and far

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);
```

CRAYPAT API EXAMPLES

A Fortran example

```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
```

A C example

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

```c
// 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
```

APPRENTICE2

CRAY APPRENTICE2

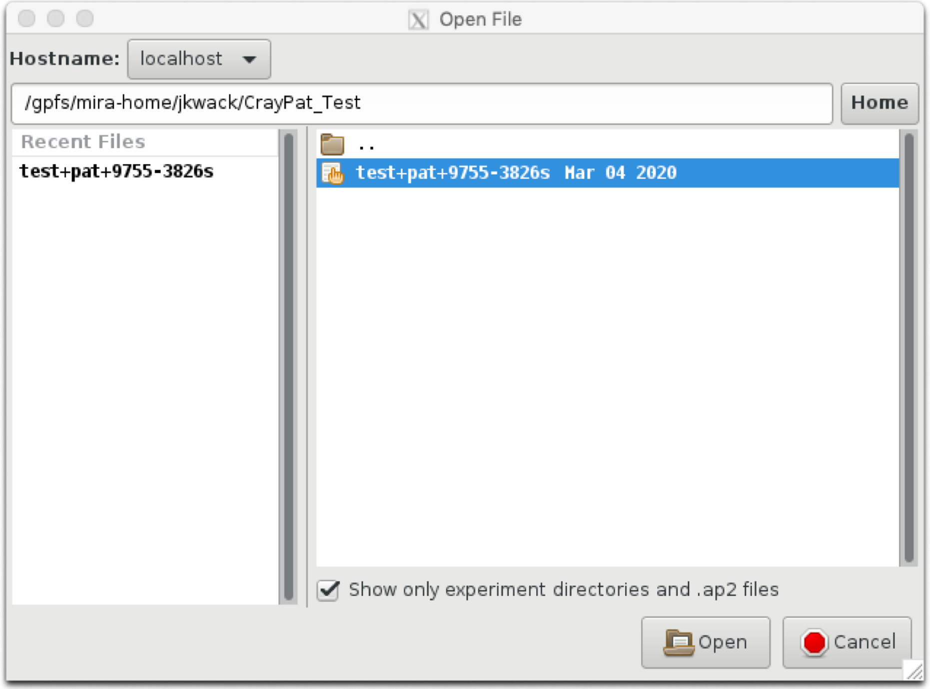
- A GUI tool for visualizing and manipulating the performance analysis data captured during program execution
  - Use pat_report to open the initial .xf data file(s) and generate the .ap2 file(s)
  - Use Cray Apprentice2 to open and explore the .ap2 file(s) in further detail.

- An example on a login node on Theta

```
$ module unload darshan
$ module load perftools-base perftools
$ app2
```

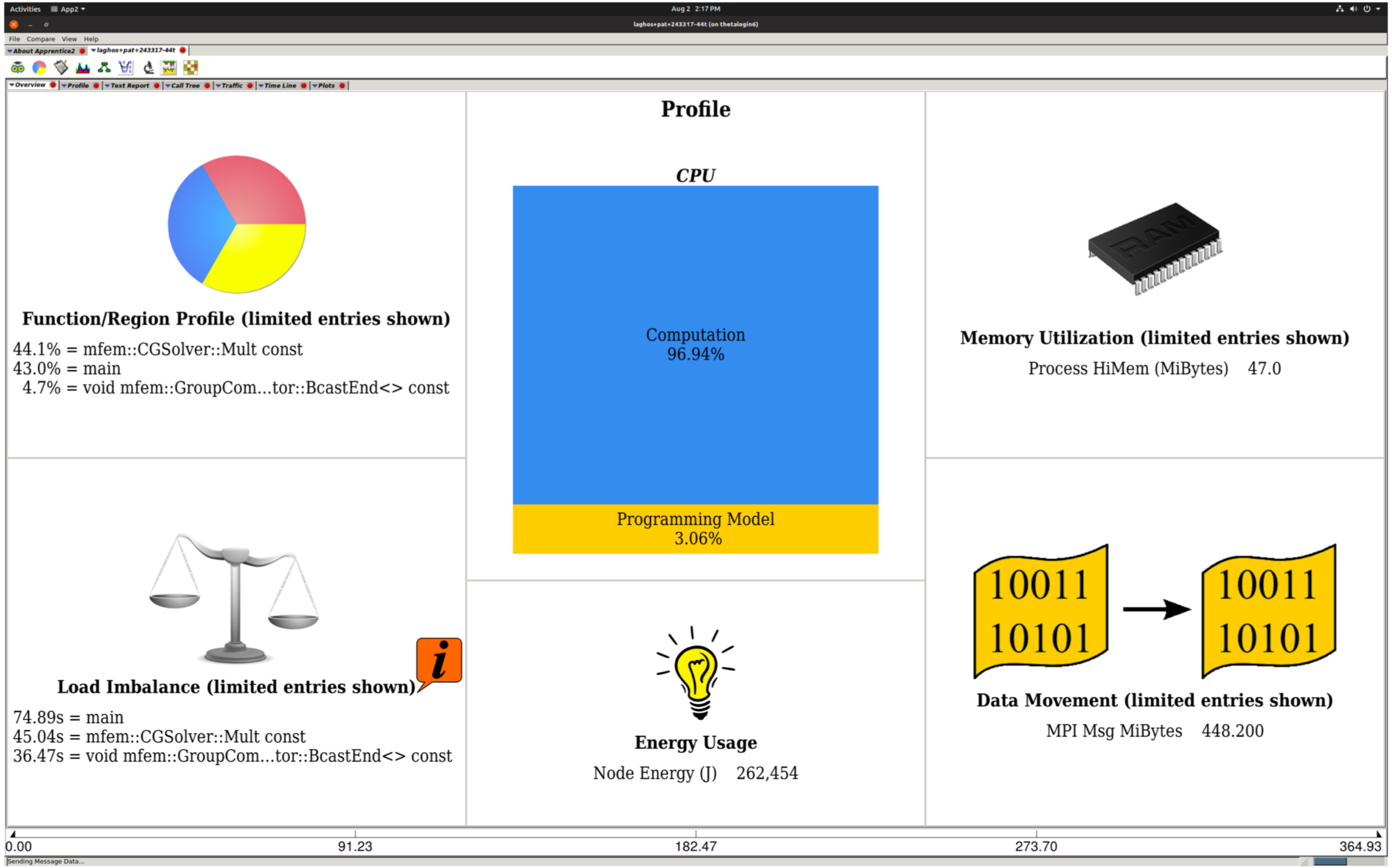
APP2


About Apprentice2
File
Compare
View
Help
CRAY
APPRENTICE2


Open File
Hostname:
localhost
/gpfs/mira-home/jkwack/CrayPat_Test
Home
Recent Files
test+pat+9755-3826s
..
test+pat+9755-3826s Mar 04 2020
Show only experiment directories and .ap2 files
Open
Cancel

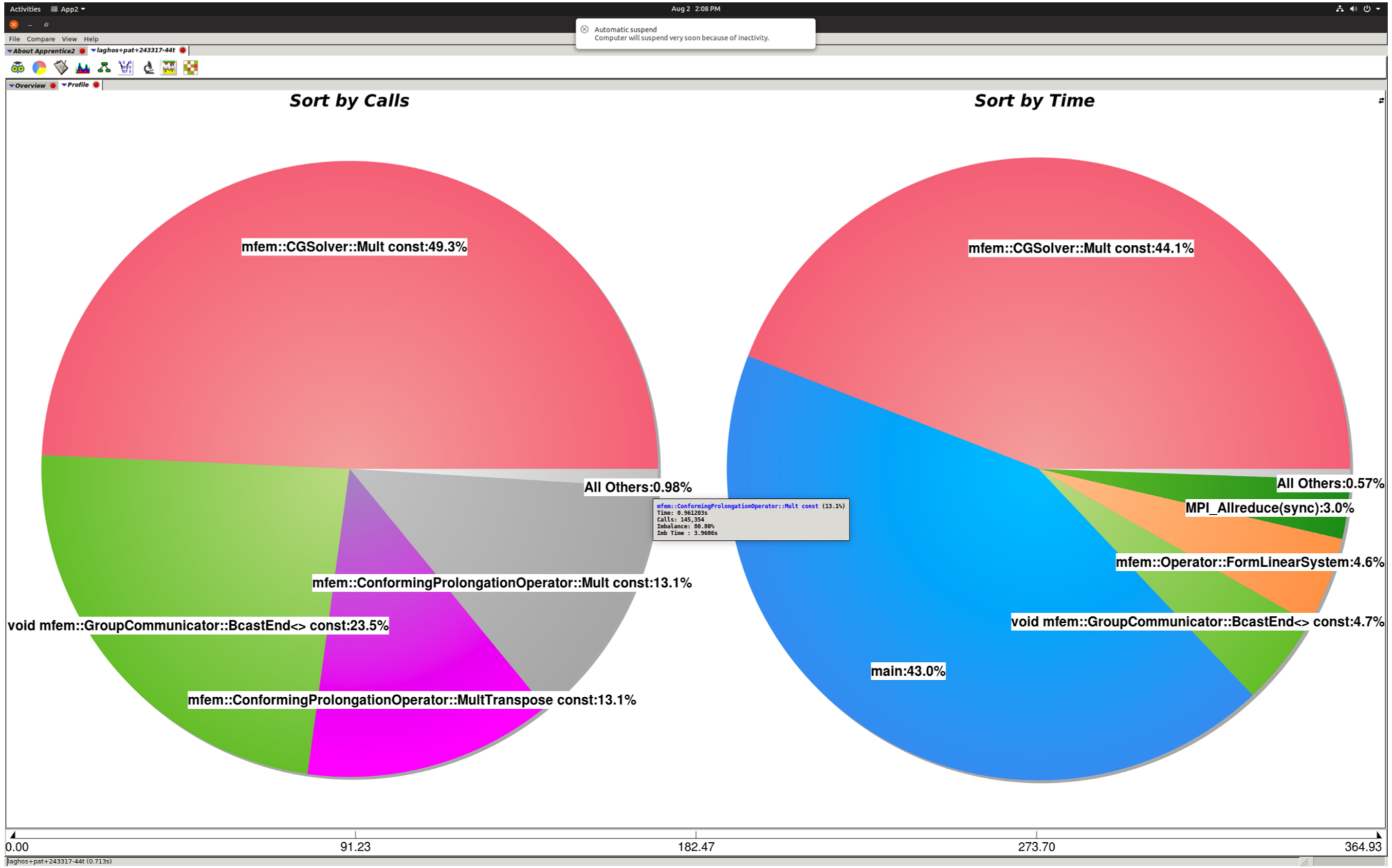
ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)

APP2



ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)

APP2



ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)

APP2

Activities App2 Jul 26 4:27 PM

Comparison (on thetalogin6)

File Compare View Help

Comparison | Comparison

laghos+pat+243317-44t

Profile

Time	Percent	Calls	Imb %	Imb Time	Function
160.542149	44.10	546842	22.00	45.042	mfem::CGSolver::Mult const
156.409358	42.96	1	32.50	74.895	main
17.168106	4.72	260952	68.30	36.474	void mfem::GroupCommunicator::BcastEnd<> const
16.786473	4.61	1984	66.10	32.354	mfem::Operator::FormLinearSystem
11.096076	3.05	303	89.60	9.943	MPI_Allreduce(sync)
0.961203	0.26	145354	80.80	3.961	mfem::ConformingProlongationOperator::Mult const
0.957412	0.26	145368	78.50	3.426	mfem::ConformingProlongationOperator::MultTranspose const
0.022764	0.01	3337	5.20	0.001	hypre_MPI_Iprobe
0.020669	0.01	303	10.00	0.002	MPI_Allreduce
0.018696	0.01	2671	23.40	0.006	hypre_MPI_Test
0.017994	0.00	20	48.70	0.017	mfem::ParMesh::GenerateOffsets const
0.017637	0.00	2	10.60	0.002	mfem::ParGridFunction::ProjectCoefficient
0.009623	0.00	24	73.70	0.027	mfem::GroupTopology::Create
0.009271	0.00	1	99.30	0.009	MPI_Finalize(sync)
0.006537	0.00	647	81.10	0.028	hypre_MPI_Testall
0.005533	0.00	48	71.40	0.004	MPI_Barrier(sync)
0.005155	0.00	1	98.70	0.005	MPI_Init(sync)
0.003859	0.00	992	92.80	0.047	mfem::IterativeSolver::SetPrintLevel
0.003454	0.00	4	90.80	0.003	MPI_Reduce(sync)
0.002844	0.00	1	21.90	0.001	exit
0.002288	0.00	170	52.40	0.002	mfem::ParFiniteElementSpace::GenerateGlobalOffsets const
0.001513	0.00	48	10.70	0.000	MPI_Barrier
0.001357	0.00	8	42.50	0.001	mfem::ParFiniteElementSpace::Build_Dof_TrueDof_Matrix const
0.000884	0.00	88	39.40	0.001	mfem::ParFiniteElementSpace::GetEssentialVDofs const
0.000413	0.00	25	73.20	0.001	hypre_MPI_Isend
0.000347	0.00	21	87.10	0.002	hypre_MPI_Recv
0.000292	0.00	1	66.50	0.001	void mfem::GroupCommunicator::ReduceEnd<> const
0.000201	0.00	22	66.00	0.000	hypre_MPI_Send
0.000193	0.00	27	53.80	0.000	mfem::ParFiniteElementSpace::ConstructTrueDofs
0.000128	0.00	26	56.80	0.000	hypre_MPI_Irecv
0.000108	0.00	4	70.40	0.000	MPI_Reduce
0.000096	0.00	4	96.20	0.002	hypre_MPI_Waitall
0.000064	0.00	35	96.70	0.002	hypre_MPI_Comm_size
0.000057	0.00	30	60.00	0.000	hypre_MPI_Comm_rank
0.000033	0.00	18	60.50	0.000	hypre_MPI_Get_count
0.000021	0.00	4	16.30	0.000	hypre_MPI_Wait
0.000009	0.00	2	22.10	0.000	mfem::ParMesh::ParMesh
0.000006	0.00	1	39.70	0.000	MPI_Finalize
0.000004	0.00	1	35.30	0.000	MPI_Init
0.000004	0.00	1	38.90	0.000	mfem::MPI_Session::GetRankAndSize
0.000003	0.00	1	39.40	0.000	MPI_Comm_size

laghos+pat+169892-40t

Profile

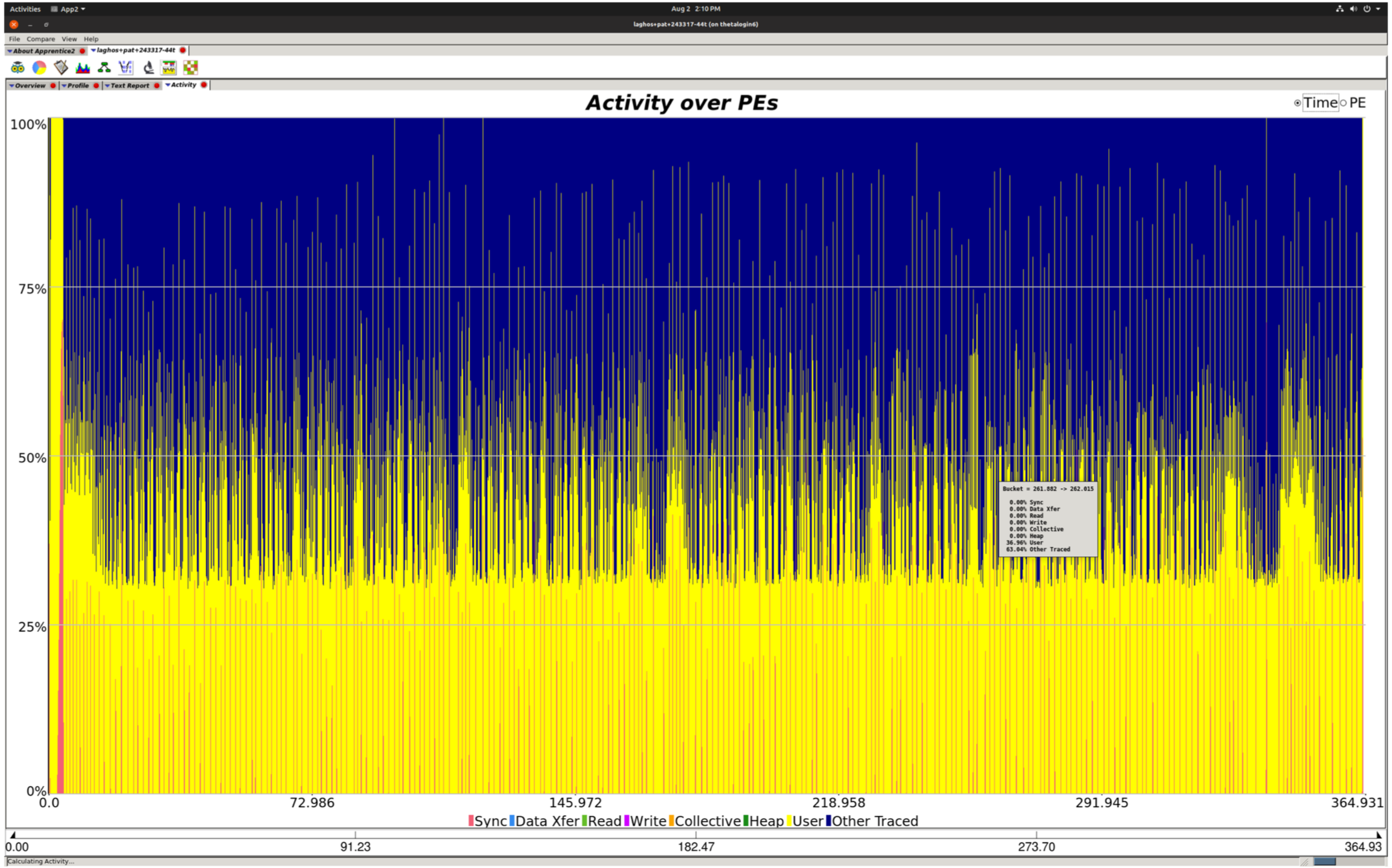
Time	Percent	Calls	Imb %	Imb Time	Function
155.924814	58.63	1	19.60	37.848	main
87.889941	33.05	417395	14.50	14.843	mfem::CGSolver::Mult const
10.540185	3.96	207204	69.60	23.822	void mfem::GroupCommunicator::BcastEnd<> const
9.020672	3.39	303	97.50	8.791	MPI_Allreduce(sync)
0.853729	0.32	102366	79.40	3.238	mfem::ConformingProlongationOperator::MultTranspose const
0.850314	0.32	1984	63.10	1.440	mfem::Operator::FormLinearSystem
0.736359	0.28	102356	76.30	2.326	mfem::ConformingProlongationOperator::Mult const
0.021932	0.01	303	11.50	0.003	MPI_Allreduce
0.019443	0.01	2	1.20	0.000	mfem::ParGridFunction::ProjectCoefficient
0.017552	0.01	20	16.90	0.004	mfem::ParMesh::GenerateOffsets const
0.012727	0.00	1832	7.60	0.001	hypre_MPI_Iprobe
0.008565	0.00	18	65.40	0.016	mfem::GroupTopology::Create
0.008355	0.00	184	27.00	0.003	mfem::ParFiniteElementSpace::GenerateGlobalOffsets const
0.008327	0.00	1165	40.00	0.006	hypre_MPI_Test
0.006127	0.00	653	60.40	0.009	hypre_MPI_Testall
0.005619	0.00	48	70.90	0.004	MPI_Barrier(sync)
0.005164	0.00	1	98.70	0.005	MPI_Init(sync)
0.004275	0.00	992	97.90	0.168	mfem::IterativeSolver::SetPrintLevel
0.003628	0.00	4	94.70	0.003	MPI_Reduce(sync)
0.003239	0.00	1	98.00	0.003	MPI_Finalize(sync)
0.002606	0.00	1	25.10	0.001	exit
0.001778	0.00	48	96.30	0.042	MPI_Barrier
0.001066	0.00	8	36.70	0.001	mfem::ParFiniteElementSpace::Build_Dof_TrueDof_Matrix const
0.000642	0.00	64	32.10	0.000	mfem::ParFiniteElementSpace::GetEssentialVDofs const
0.000469	0.00	19	55.00	0.001	hypre_MPI_Isend
0.000419	0.00	1	68.80	0.001	void mfem::GroupCommunicator::ReduceEnd<> const
0.000247	0.00	15	94.20	0.004	hypre_MPI_Recv
0.000191	0.00	17	69.20	0.000	hypre_MPI_Send
0.000160	0.00	20	40.30	0.000	mfem::ParFiniteElementSpace::ConstructTrueDofs
0.000108	0.00	4	63.60	0.000	MPI_Reduce
0.000102	0.00	21	66.00	0.000	hypre_MPI_Irecv
0.000068	0.00	4	98.50	0.004	hypre_MPI_Waitall
0.000051	0.00	27	97.50	0.002	hypre_MPI_Comm_size
0.000046	0.00	25	43.50	0.000	hypre_MPI_Comm_rank
0.000024	0.00	13	41.00	0.000	hypre_MPI_Get_count
0.000020	0.00	4	16.50	0.000	hypre_MPI_Wait
0.000009	0.00	2	22.00	0.000	mfem::ParMesh::ParMesh
0.000006	0.00	1	35.30	0.000	MPI_Finalize
0.000004	0.00	1	44.70	0.000	MPI_Init
0.000004	0.00	1	44.20	0.000	mfem::MPI_Session::GetRankAndSize
0.000003	0.00	1	42.40	0.000	MPI_Comm_size

Wallclock: 364.9307s / 267.3565s

laghos+pat+169892-40t (0.086s)

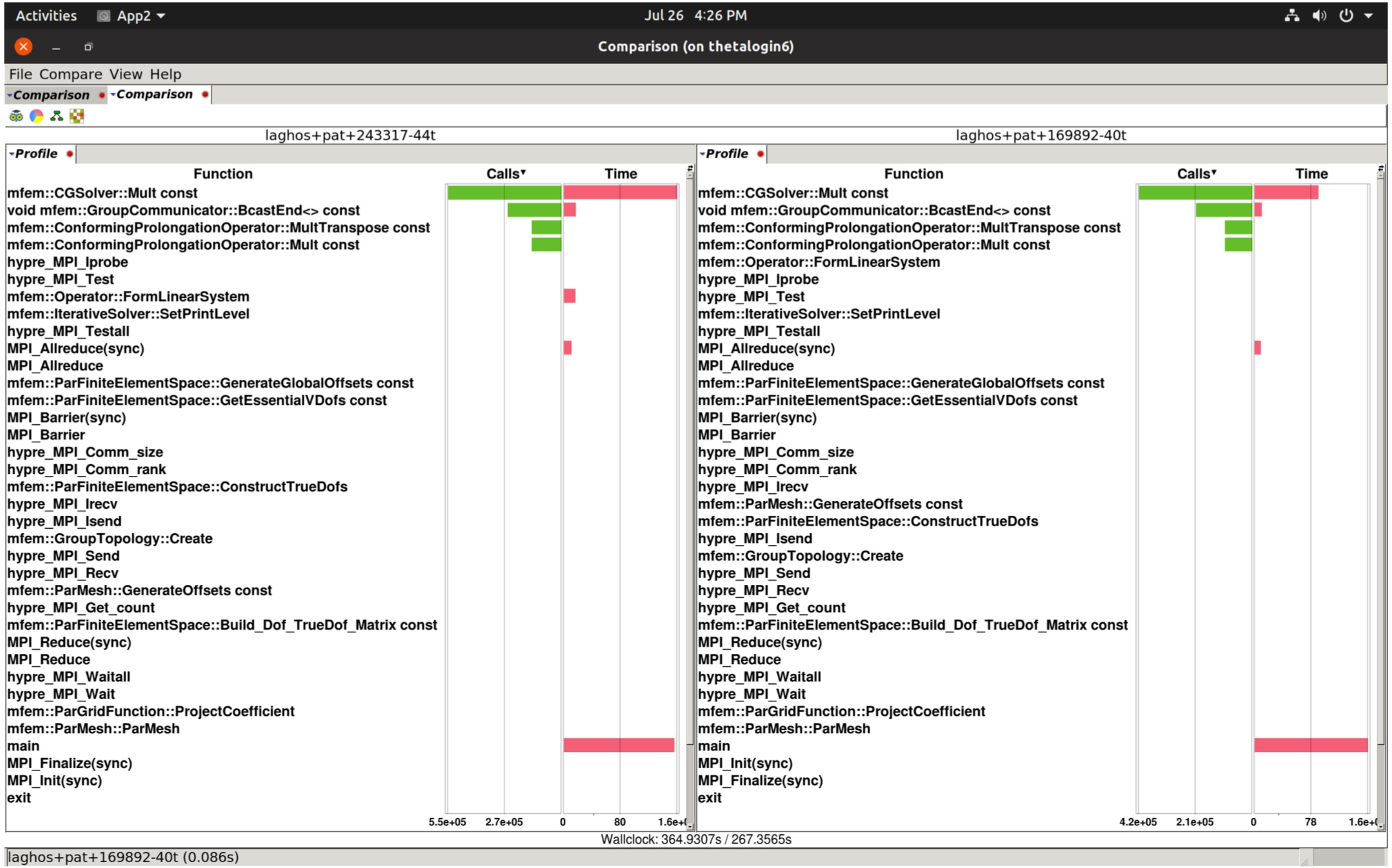
APP2

ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)



ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)

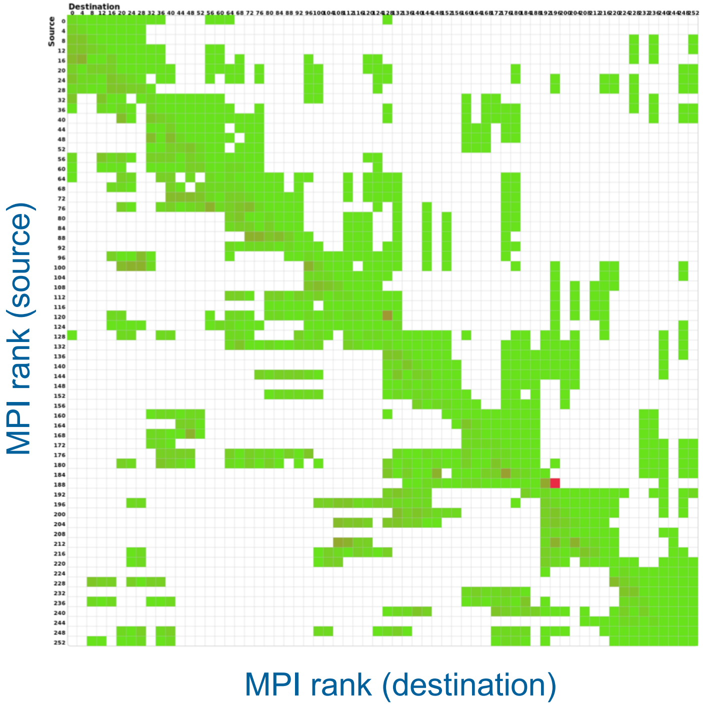
APP2



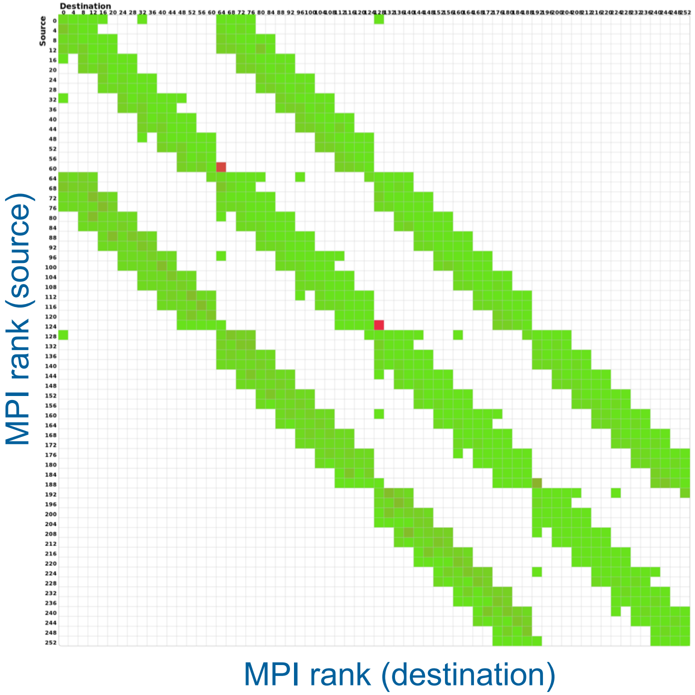
ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)

APP2

The default MPI partitioning



An optimal MPI partitioning



ECP proxy application, Laghos on Theta
By ALCF undergraduate intern, Michael Simon (2019)

APP2

The default MPI partitioning

```
 Samp% |     Samp |    Imb. |  Imb. | Group
       |          |    Samp | Samp% |  Function
       |          |         |       |   PE=HIDE

100.0% | 24,741.0 |      -- |    -- | Total
|-----------------------------------------------------------------------------
|  57.1% | 14,131.4 |      -- |    -- | USER
||----------------------------------------------------------------------------
||  19.4% |  4,794.7 | 2,544.3 | 34.8% | mfem::hydrodynamics::MassPAOperator::Mult
const
||   8.5% |  2,095.3 | 1,085.7 | 34.3% | mfem::Mult
||   6.5% |  1,617.6 |   998.4 | 38.3% | mfem::MultAtB
||   4.2% |  1,039.3 |   581.7 | 36.0% | mfem::DenseMatrix::CalcEigenvalues const
||   3.4% |    835.4 |   420.6 | 33.6% |
mfem::hydrodynamics::FastEvaluator::GetVectorGrad const
||   1.7% |    410.4 |   236.6 | 36.7% | mfem::MultABt
||   1.6% |    400.8 |   188.2 | 32.1% |
mfem::hydrodynamics::ForcePAOperator::MultTransposeHex const
||   1.6% |    391.1 |   196.9 | 33.6% | mfem::DenseMatrix::CalcSingularvalue const
||   1.5% |    374.7 |   192.3 | 34.0% |
mfem::hydrodynamics::LagrangianHydroOperator::UpdateQuadratureData const
||   1.3% |    320.8 |   177.2 | 35.7% |
mfem::hydrodynamics::ForcePAOperator::MultHex const
||============================================================================
|  37.7% |  9,322.7 |      -- |    -- | MPI
||----------------------------------------------------------------------------
||  18.1% |  4,470.3 | 4,647.7 | 51.2% | MPI_Allreduce
||  12.9% |  3,187.4 | 6,818.6 | 68.4% | MPI_Waitall
||   5.9% |  1,450.3 | 3,355.7 | 70.1% | MPI_Waitany
||============================================================================
|   4.5% |  1,117.7 |      -- |    -- | ETC
||----------------------------------------------------------------------------
||   1.6% |    397.8 |   212.2 | 34.9% | __libm_hypot_e7
|=============================================================================
```

An optimal MPI partitioning

```
 Samp% |     Samp |    Imb. |  Imb. | Group
       |          |    Samp | Samp% |  Function
       |          |         |       |   PE=HIDE

100.0% | 20,111.8 |      -- |    -- | Total
|-----------------------------------------------------------------------------
|  69.9% | 14,057.7 |      -- |    -- | USER
||----------------------------------------------------------------------------
||  23.9% |  4,800.4 |   289.6 |  5.7% | mfem::hydrodynamics::MassPAOperator::Mult
const
||  10.4% |  2,096.8 |   671.2 | 24.3% | mfem::Mult
||   8.1% |  1,620.0 |   586.0 | 26.7% | mfem::MultAtB
||   5.2% |  1,038.3 |    89.7 |  8.0% | mfem::DenseMatrix::CalcEigenvalues const
||   4.1% |    830.8 |    92.2 | 10.0% |
mfem::hydrodynamics::FastEvaluator::GetVectorGrad const
||   2.1% |    412.4 |   147.6 | 26.5% | mfem::MultABt
||   2.0% |    405.2 |    29.8 |  6.9% |
mfem::hydrodynamics::ForcePAOperator::MultTransposeHex const
||   1.9% |    390.1 |   114.9 | 22.8% | mfem::DenseMatrix::CalcSingularvalue const
||   1.9% |    378.7 |    58.3 | 13.4% |
mfem::hydrodynamics::LagrangianHydroOperator::UpdateQuadratureData const
||   1.6% |    322.6 |    43.4 | 11.9% |
mfem::hydrodynamics::ForcePAOperator::MultHex const
||   1.1% |    222.9 |    43.1 | 16.3% | mfem::Vector::Norml2 const
||============================================================================
|  23.7% |  4,761.6 |      -- |    -- | MPI
||----------------------------------------------------------------------------
||  17.7% |  3,561.3 | 1,340.7 | 27.5% | MPI_Allreduce
||   4.4% |    880.3 | 1,937.7 | 69.0% | MPI_Waitany
||============================================================================
|   5.6% |  1,122.5 |      -- |    -- | ETC
||----------------------------------------------------------------------------
||   2.0% |    395.1 |    73.9 | 15.8% | __libm_hypot_e7
|=============================================================================
```

SUMMARY

- Two modes to use CrayPat
  - "Lite" mode
  - In-depth analysis
- Performance counters
- CrayPat API
- Apprentice2

- ALCF CrayPat user-guide: https://www.alcf.anl.gov/support-center/theta/craypat
- For more supports, please reach out to JaeHyuk Kwack (jkwack@anl.gov) or ALCF Performance Engineering Group

THANK YOU!