BLUE WATERS

SUSTAINED PETASCALE COMPUTING

VISIONARY COMPUTING

CUG 2018 STOCKHOLM

# Roofline Analysis with CrayPat and Roofline-based Performance Projections for a Future Architecture

JaeHyuk Kwack, Galen Arnold, Celso Mendes and Gregory H Bauer
National Center for Supercomputing Applications

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

*This paper has been accepted and will be published as an article in a special issue of Concurrency and Computation Practice and Experience on the Cray User Group 2018.*

# The Blue Waters system[1,2]

## As of May 15, 2018,

24 IN THE PAST HOURS

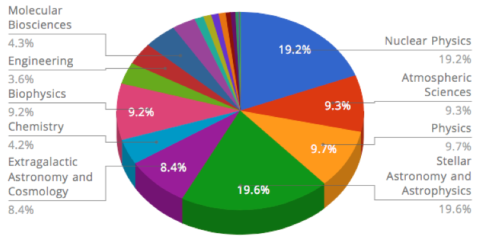
| JOBS STARTED | JOBS QUEUED | JOBS COMPLETED |
|---|---|---|
| 2899 | 3117 | 2628 |

TOTAL COMPUTING POWER DELIVERED

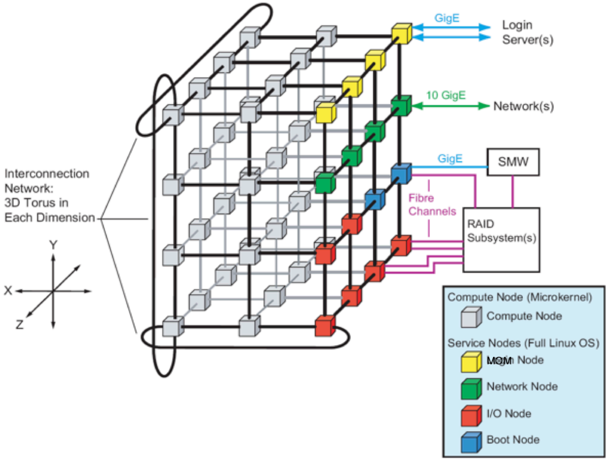
Since Blue Waters went into production on March 28, 2013, it has provided 22.5 billion core - hours to scientists and engineers across the country .

22,582,875,998

CURRENT RUNNING JOBS BY SCIENCE AREA



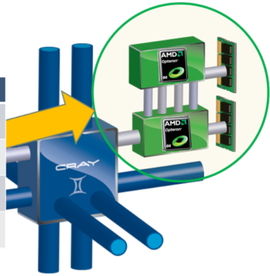

## 24x24x24 3D Torus network




## 22,640 XE6 nodes

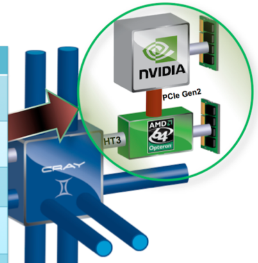
**XE6 Compute Node**
- Dual-socket AMD-Opteron
- 4x channel 1600 DDR3 memory
- High speed HT3 network link
- Upgradeable
- Blend with XK6 GPU systems

| Node Characteristics | |
|---|---|
| Number of Cores | 16 |
| Peak Performance | 313.6 Gflops/sec |
| Memory Sizes Available | 64 GB per node |
| Memory Bandwidth (Peak) | 102.4 GB/sec |



## 4,228 XK7 nodes

| XK7 Compute Node Characteristics | |
|---|---|
| Host Processor | AMD Series 6200 (Interlagos) |
| Host Processor Performance | 156.8 Gflops |
| Kepler Peak (DP floating point) | 1.32 Tflops |
| Host Memory | 32GB<br>51 GB/sec |
| Kepler Memory | 6GB GDDR5 capacity<br>> 180 GB/sec |

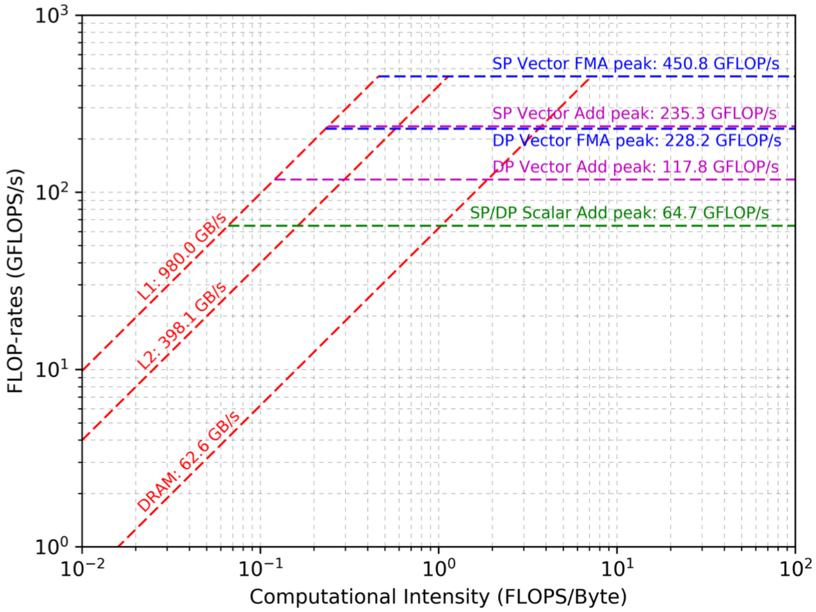



# Why roofline performance analysis model matters to us?

- We want to measure "real application performance" reflecting the NSF workload
- SPP benchmark at CUG-2012 [3]
  - NAMD: molecular dynamics
  - MILC,CHROMA,VPIC: particle physics
  - SPECFEM3D: geophysics
  - WRF: weather forecast
  - PPM: astrophysics
  - NWCHEM, GAMESS: quantum chemistry
  - QMCPACK: material science
- Updated SPP benchmark at CUG-2017 [4]
  - AWP-ODC: geophysics
  - CACTUS: astrophysics
  - MILC: quantum chromodynamics
  - NAMD: molecular dynamics
  - NWCHEM: computational chemistry
  - PPM: astrophysics
  - PSDNS: turbulence
  - QMCPACK: material science
  - RMG: electronic structure of materials
  - VPIC: movement of charged particles
  - WRF: weather forecast
- We wanted to get a visually intuitive performance analysis plot for our SPP applications
  - to characterize their performance bottlenecks
  - and get an idea about their performance on our next generation machine
- Roofline performance analysis model was attractive, but we could not find a reliable tool for that.

# ERT measurements of an XE node

- Empirical Roofline Tool (ERT) version 1.1.0 [5]
- Employed compilers
  - GNU: gcc/4.9.3 and gcc/6.3.0
  - Cray: cce/8.5.8
  - PGI: pgi/17.5.0
- Considered Peak Flop-rates
  - Vector FMA peak: w/SIMD and FMA4
  - Vector Add peak: w/ SIMD and w/o FMA4
  - Scalar Add peak: w/o SIMD and FMA4
- Cores and cache hierarchy of an XE node
  - 32 integer core, 16 FPUs, 4 NUMA domains
  - L1 data cache: 16 KB/integer core
  - L2 data cache: 2048 KB/FPU
  - L3 data cache: 8192 KB/ NUMA domain



# COMPUTATIONAL INTENSITIES BASED ON HARDWARE PERFORMANCE COUNTERS

# Validation via kernels for computing geometric series

- The $n^{th}$ order of geometric series

  $OM(n) = 1+M+M^2+M^3+ \dots +M^n$
  Data movement = 2 variables
  Minimum FLOPs = 2n – 1
  CI = (2n-1) / (2 variables)

- 4176 test cases
  - The order of the series: 1 to 29 (i.e., 29 cases)
  - The size of array per MPI rank: $64^2$, $256^2$, $1024^2$, and $4096^2$ (i.e., 4 cases)
  - Variable type: 4-byte SP and 8-byte DP (i.e., 2 cases)
  - Compiler type: gnu, cray, and pgi (i.e., 3 cases)
  - Optimization level: O0 and O3 (i.e., 2 cases)
  - Implementation type: inline, recursive loop, and flat loop (i.e., 3 cases)

- Three types of implementation

***Inline-implementation:***
Loops for i and j from 1 to m:
  OM(i,j) = 1.0 + M(i,j) + M(i,j)^2 + M(i,j)^3 + ... + M(i,j)^n

***Recursive loops:***
Loops for i and j from 1 to m:
  OM(i,j) = 1.0
  A loop for k from 1 to n:
    OM(i,j) = OM(i,j)*M(i,j) + 1.0

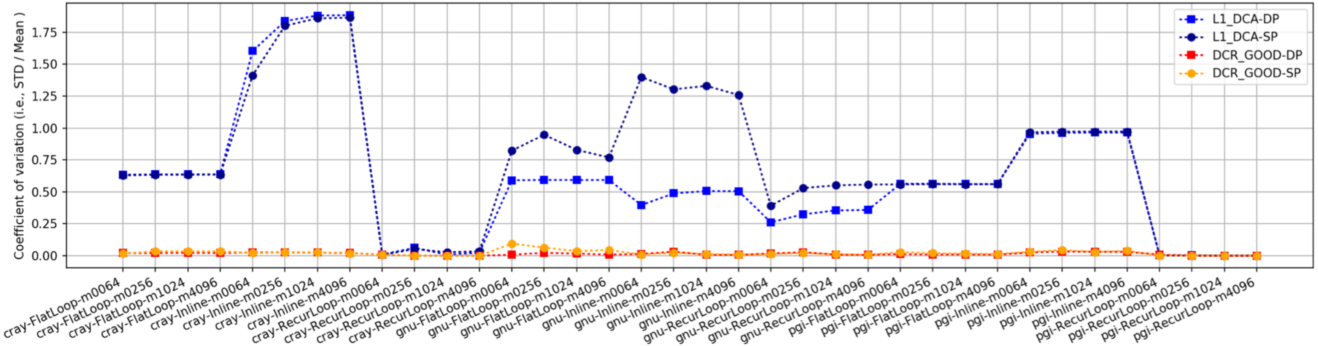
***Flat loops:***
Loops for i and j from 1 to m:
  OM(i,j) = 1.0
  A loop for k from 1 to n:
    OM(i,j) = OM(i,j) + M(i,j)^k

# Employed hardware counters to measure data transfers

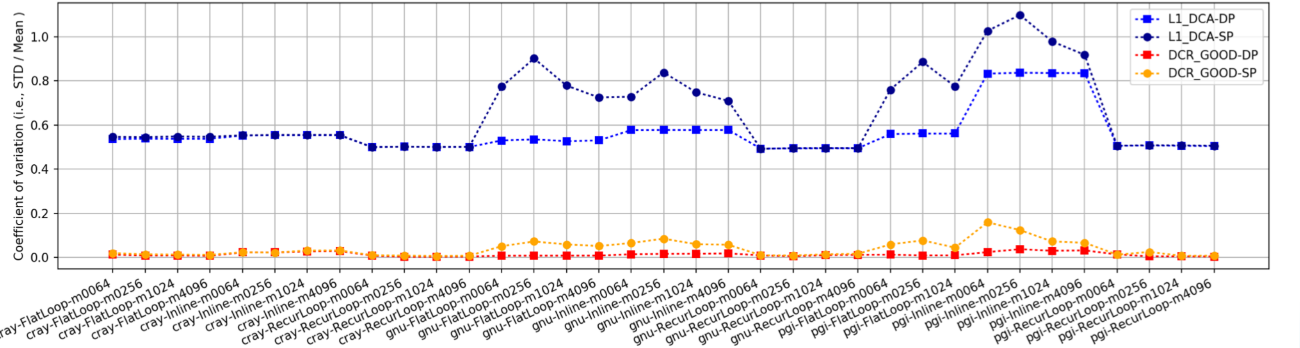
- L1_DCA
  - PAPI_L1_DCA, or DATA_CACHE_ACCESSES
  - Level 1 data cache accesses
  - CrayPat default counter for computing computational intensity
  - A single reference = a single vector length (i.e., 16 bytes = 128 bits)
- DCR_GOOD
  - DATA_CACHE_REFILLES_FROM_L2_OR_NORTHBRIDGE:GOOD
  - The number of data cache refills satisfied from the L2 cache and/or the system with valid final status
  - A single reference = a 64-byte transfer
- Other counters have been tested in the same way
- Instrumenting the code with CrayPat
  - Focusing on kernels of interest using CrayPat API (PAT_region_begin/end and PAT_record)
  - Enabling the CrayPat API via “pat_build -w *myprogram*”
  - Setting the runtime variable for hardware performance counters (i.e., PAT_RT_PERFCTR)

# Coefficients of variation (i.e., STD/mean) of hardware counter data

- **The O3 optimization**

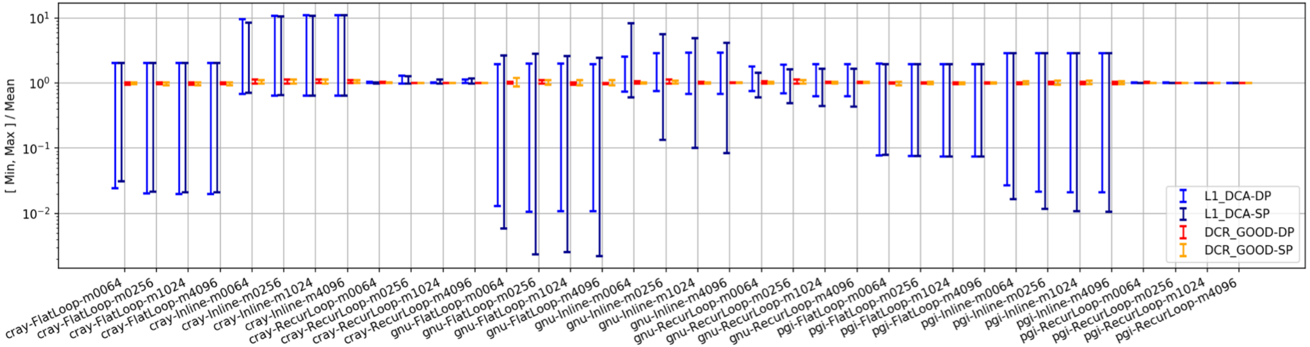



- **The O0 optimization**

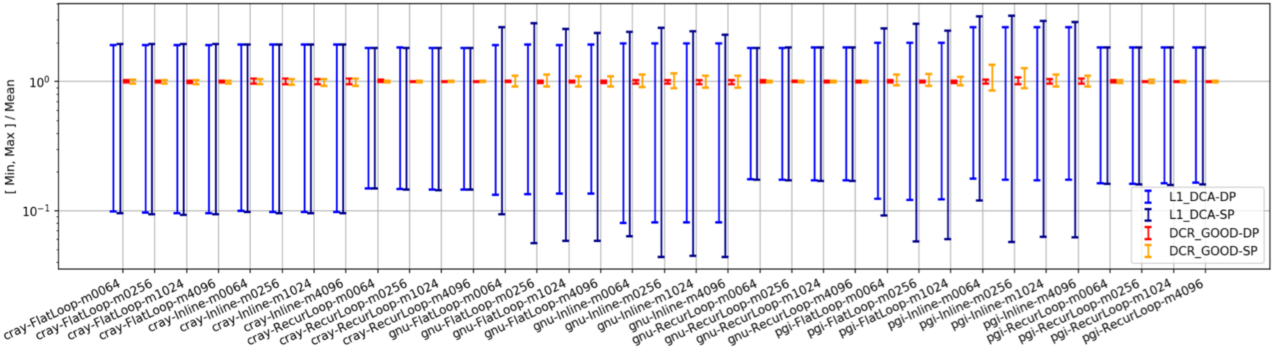


# Range of MIN and MAX of byte measurements

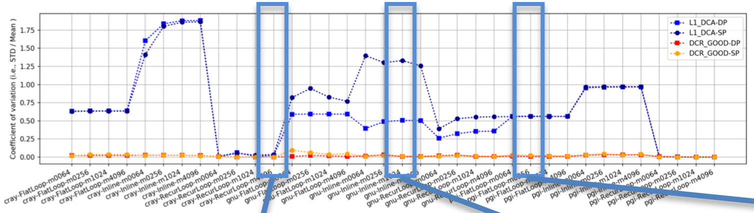
- **The O3 optimization**

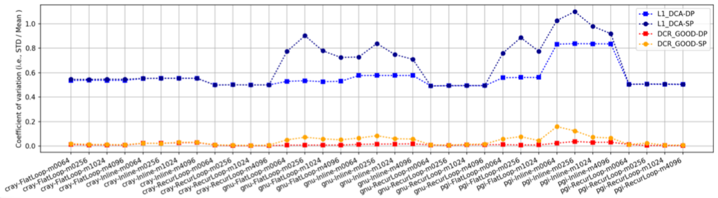



- **The O0 optimization**

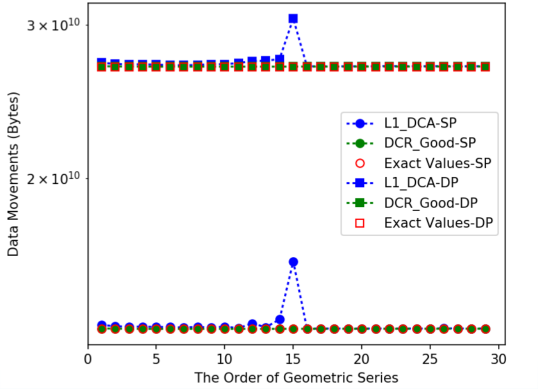


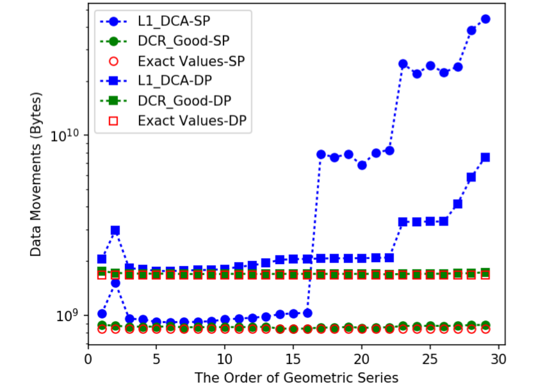
# Raw data of byte measurement for selected cases

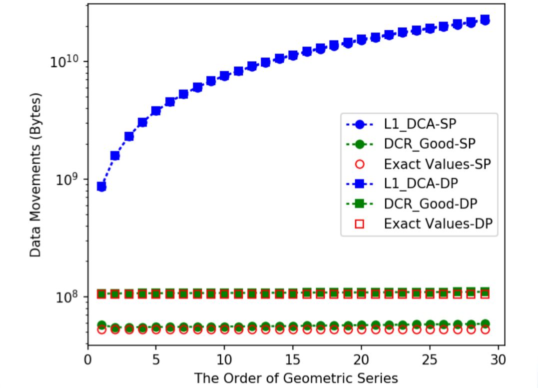


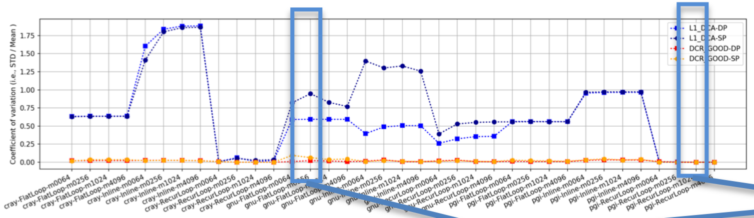

- Cray-RecurLoop-m4096-O3
- Gnu-Inline-m1024-O3
- Pgi-FlatLoop-m0256-O3

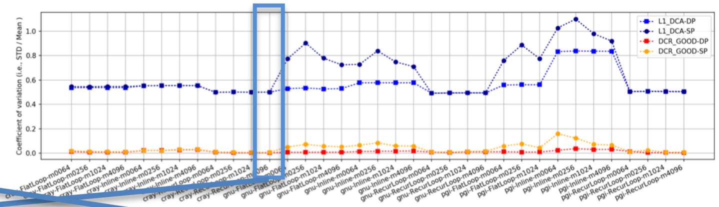


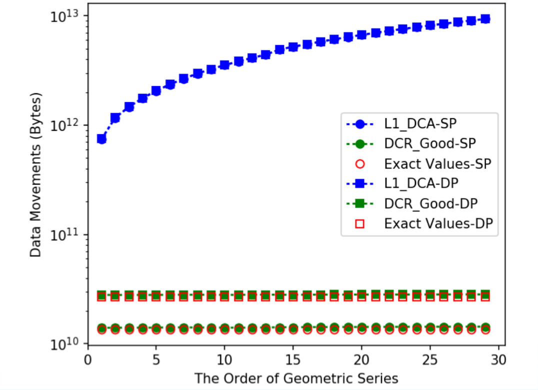



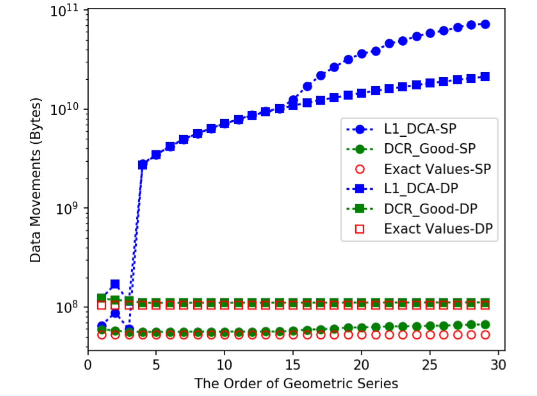
# Raw data of byte measurement for selected cases

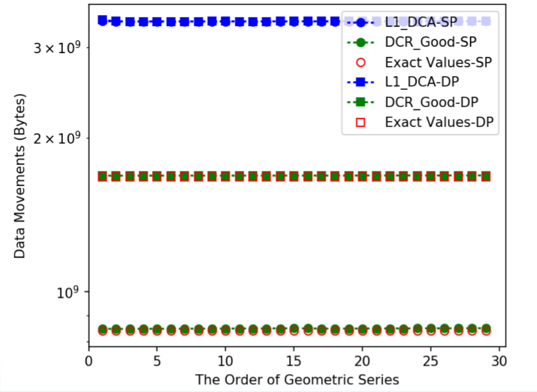



- Cray-RecurLoop-m4096-O0
- Gnu-FlatLoop-m0256-O3
- Pgi-RecurLoop-m1024-O3

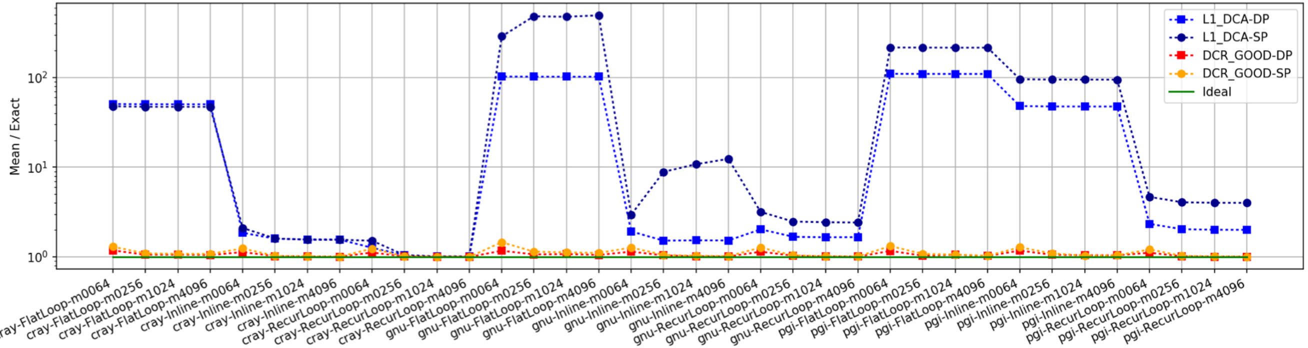


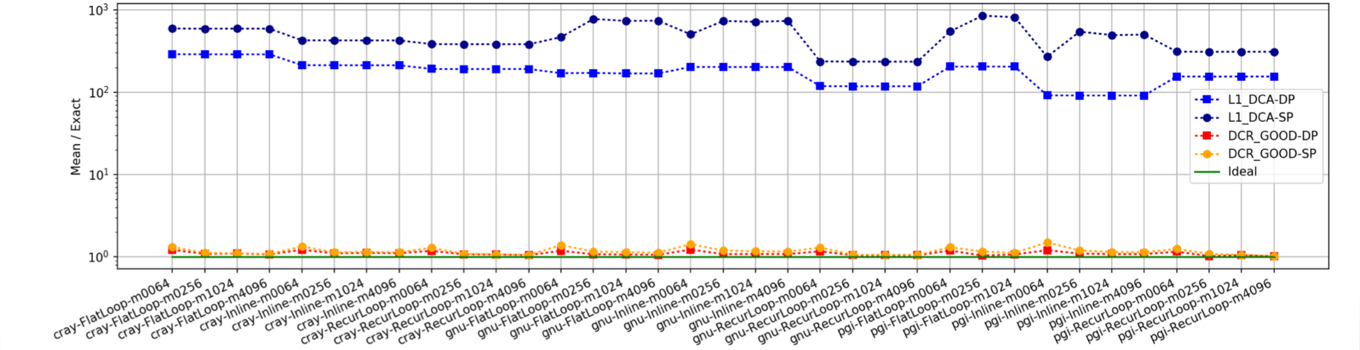



# Ratio of the byte measurements over the exact transfers

- The O3 optimization




- The O0 optimization



# Validation via 2D/3D stencil codes

- The main kernels of 2D/3D stencil codes

```
5-point 2D stencil code:
Loops for iter from 1 to niter:
      PAT_region_begin()
      Loops for i and j from 2 to n+1:
            OM(i,j) = 0.25* ( M(i+1,j) + M(i-1,j) + M(i,j+1) + M(i,j-1) )
      PAT_region_end()
      Loops for i and j from 2 to n+1:
            M(i,j) = OM(i,j)
```

```
7-point 3D stencil code:
Loops for iter from 1 to niter:
      PAT_region_begin()
      Loops for i, j and k from 2 to n+1:
            OM(i,j,k) = 0.1666666666666666667 * ( M(i+1,j,k) + M(i-1,j,k)
                        + M(i,j+1,k) + M(i,j-1,k) + M(i,j,k+1) + M(i,j,k-1) )
      PAT_region_end()
      Loops for i and j from 2 to n+1:
            M(i,j,k) = OM(i,j,k)
```

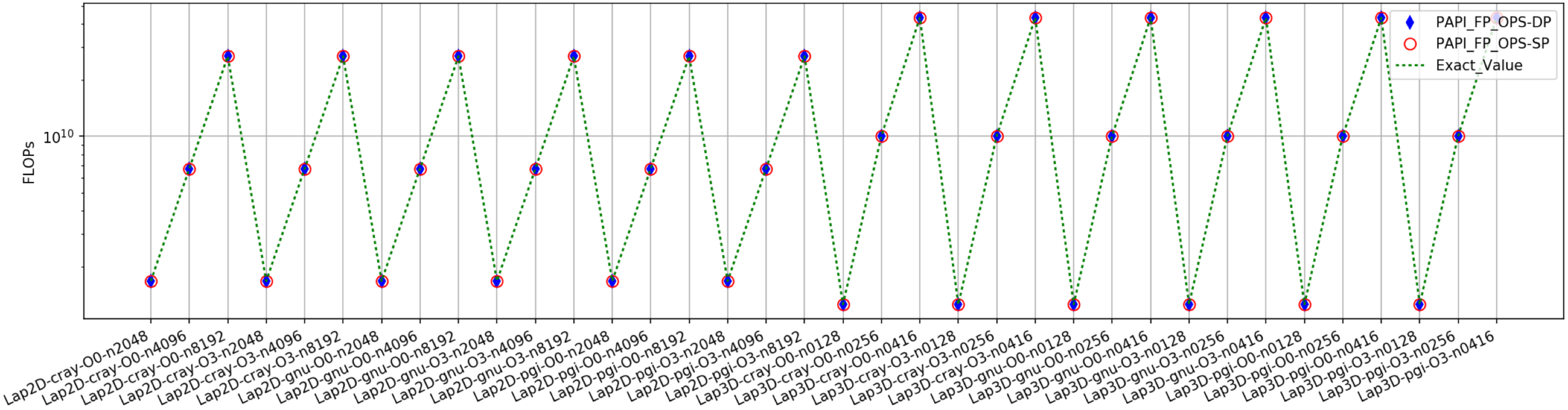
- FLOPs, data transfers, and Comp. Intensity (CI)

***5-point 2D stencil code:***
FLOPs = 4*n*n*niter
Data Movements = 5*n*n*niter
CI for DP = 4./5/8 = 0.1 FLOPS/byte
CI for SP = 4./5/4 = 0.2 FLOPS/byte

***7-point 3D stencil code:***
FLOPs = 6*n*n*n*niter
Data Movements = 7*n*n*n*niter
CI for DP = 6./7/8 = 0.1071 FLOPS/byte
CI for SP = 6./7/4 = 0.2143 FLOPS/byte

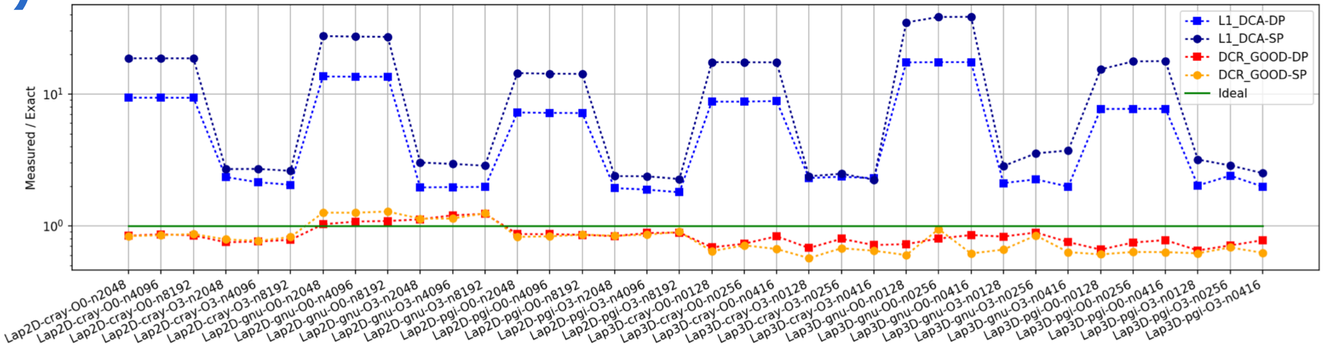
- 72 test cases
  - Stencil size per MPI rank (i.e., $(n+2)^2$ for 2D and $(n+2)^3$ for 3D): $(2048+2)^2$, $(4096+2)^2$, $(8192+2)^2$ for 2D, and $(128+2)^2$, $(256+2)^2$, $(416+2)^2$ for 3D (i.e., 6 cases in total)
  - Variable type: 4-byte SP and 8-byte DP (i.e., 2 cases)
  - Compiler type: gnu 4.9.3, cray 8.5.8, and pgi 17.5.0 (i.e., 3 cases)
  - Optimization level: O0 and O3 (i.e., 2 cases)

# FLOP measurements

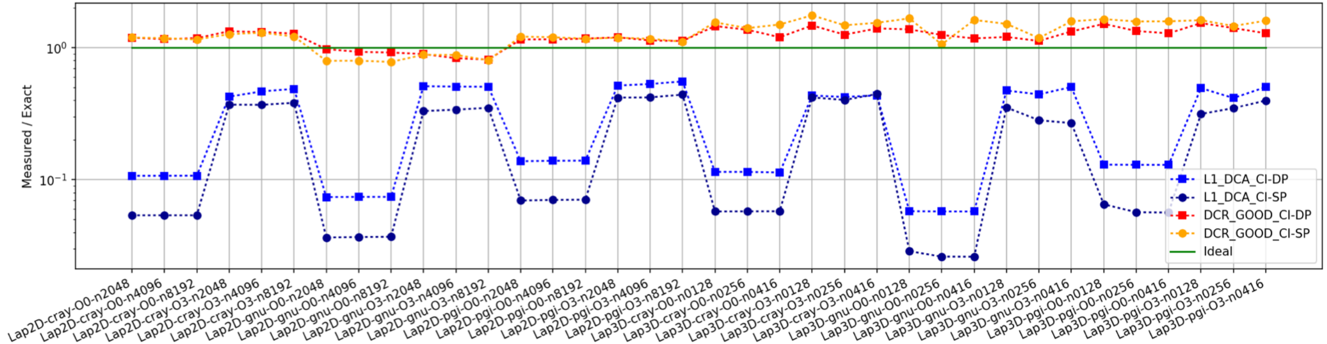


# Ratio of the byte measurements over the exact transfers and Comp. Intensity (CI)

- Byte measurements

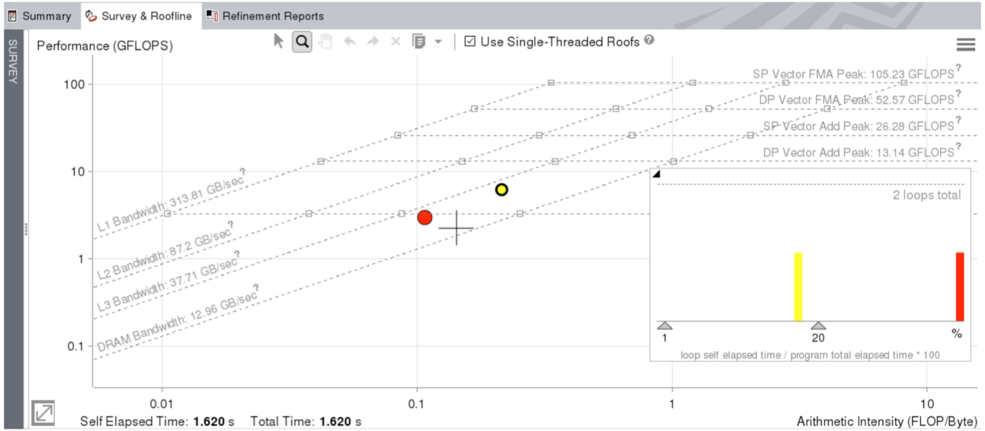



- Computational Intensity



# Roofline analysis via Intel Advisor – 2D/3D stencil kernels

- Test platform
  - A dual Intel E5-2680V4 (Broadwell) processor node of UIUC campus cluster
  - Intel/18.0 with -O3 –xHOST
- Limitations of Intel Advisor
  - Not compatible with other vendors' processors (e.g., AMD, ARM, and NVIDIA)
  - Not compatible with other compilers (e.g., CRAY, PGI, GNU, ARM)
  - Requiring at least two executions (i.e., Survey analysis, and Trip Counts Analysis with FLOP), so not suitable for large-scale simulations




A screen shot of Intel Advisor Roofline features; 3D kernel with $(256+2)^3$ stencil

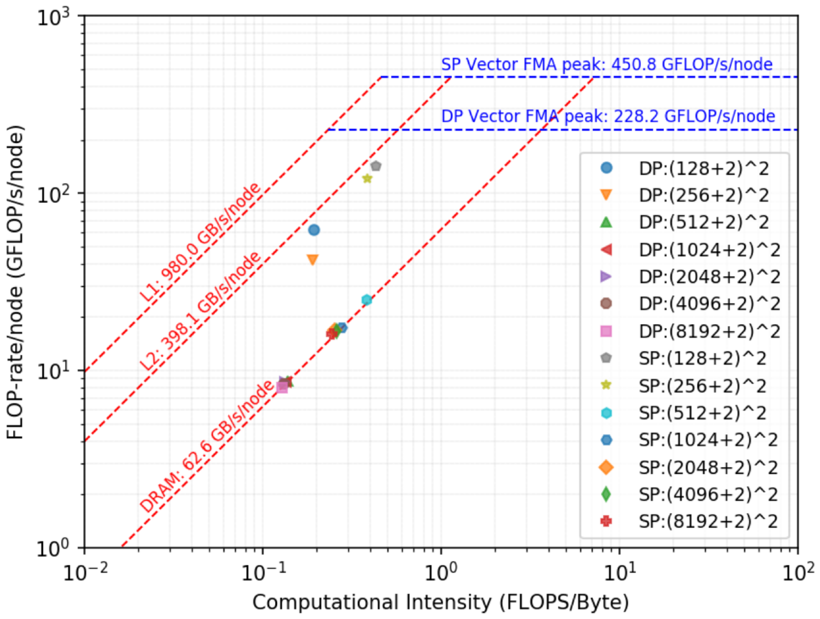
# CRAYPAT-BASED ROOFLINE ANALYSIS FOR HPC BENCHMARKS AND POPULAR KERNELS

# 5-point 2D stencil kernel

- Cray cce/8.5.8 with the O3 optimization
- Access to two arrays (i.e., M and OM)
- Required memory size in total

L2 data cache (1024 KB/core)

| Stencil | SP | DP |
|---|---|---|
| $(128+2)^2$ | 135 KB | 270 KB |
| $(256+2)^2$ | 532 KB | 1.065 MB |
| $(512+2)^2$ | 2.1 MB | 4.2 MB |
| $(1024+2)^2$ | 8.4 MB | 16.8 MB |
| $(2048+2)^2$ | 33.6 MB | 67.2 MB |
| $(4906+2)^2$ | 134 MB | 269 MB |
| $(8192+2)^2$ | 537 MB | 1.074 GB |

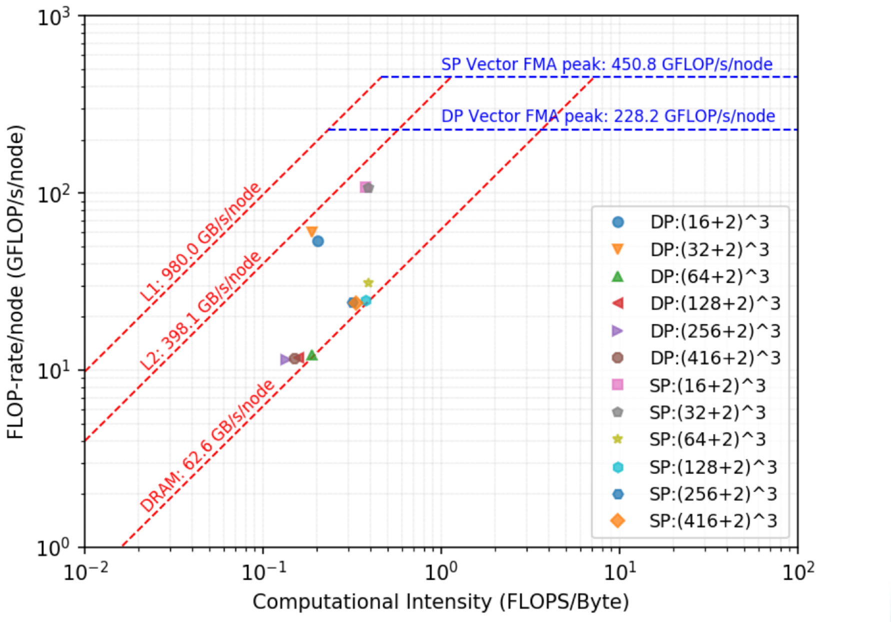


# 7-point 3D stencil kernel

- Cray cce/8.5.8 with the O3 optimization
- Access to two arrays (i.e., M and OM)
- Required memory size in total

| Stencil | SP | DP |
|---|---|---|
| $(16+2)^3$ | 47 KB | 93 KB |
| $(32+2)^3$ | 314 KB | 629 KB |
| $(64+2)^3$ | 2.3 MB | 4.6 MB |
| $(128+2)^3$ | 17.6 MB | 35.2 MB |
| $(256+2)^3$ | 127 MB | 275 MB |
| $(416+2)^3$ | 584 MB | 1.169 GB |

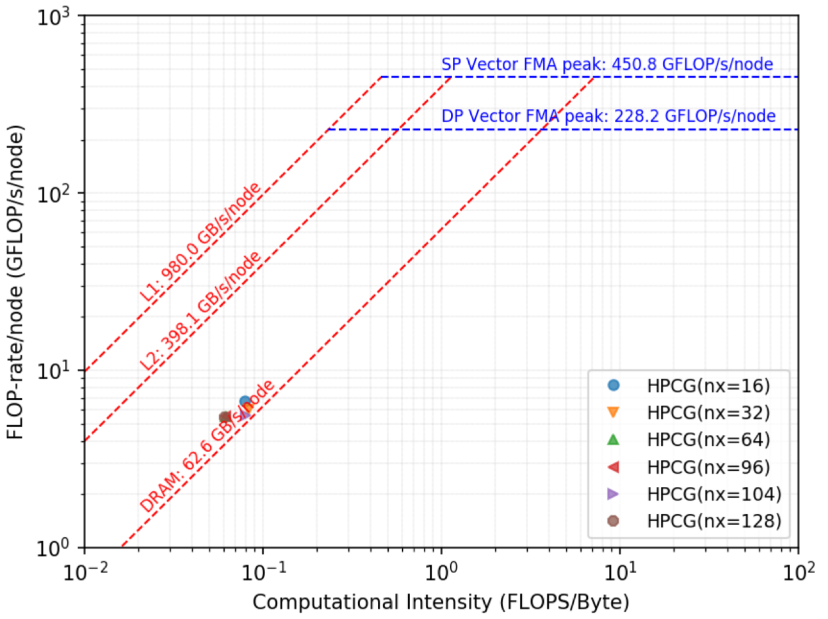
L2 data cache (1024 KB/core)



# HPCG[6]

- GNU gcc/4.9.3 with the O3 optimization
- A 27-point stencil operator with
  - dense and sparse computations,
  - truncated multi-grid V cycles,
  - unstructured sparse triangular solver
- 128 MPI ranks on 4 XE nodes
- $16^3$ to $128^3$ per MPI rank

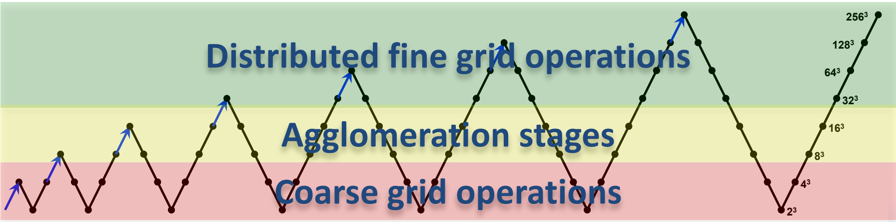
- Mostly DRAM bandwidth bounded performance



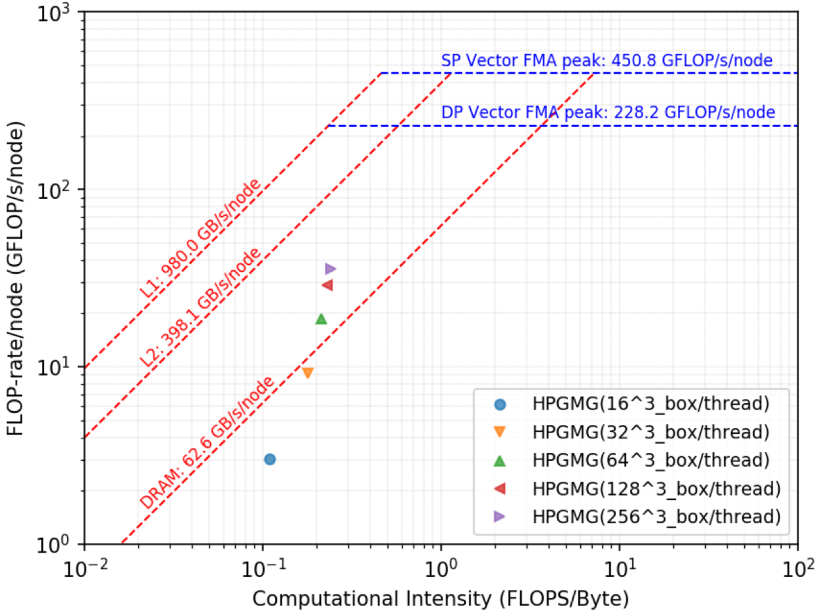
# HPGMG-FV[7]

- GNU gcc/4.9.3 with the O3 optimization
- The $4^{th}$-order accuracy finite volume
- The full multi-grid (FMG) F-cycle
- 512 MPI ranks on 16 XE nodes
- $16^3$ to $256^3$ cubes per MPI rank

- Most critical performance bottleneck is load imbalance via FMG F-cycle



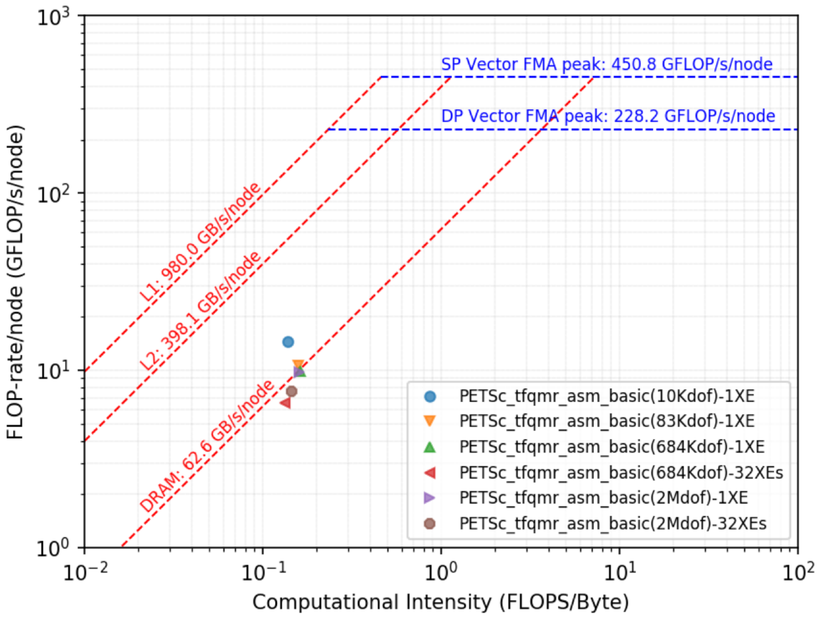

Source: Williams (hpgmg.org), HPGMG BoF, SC-16, 2016



# Sparse Linear Solver

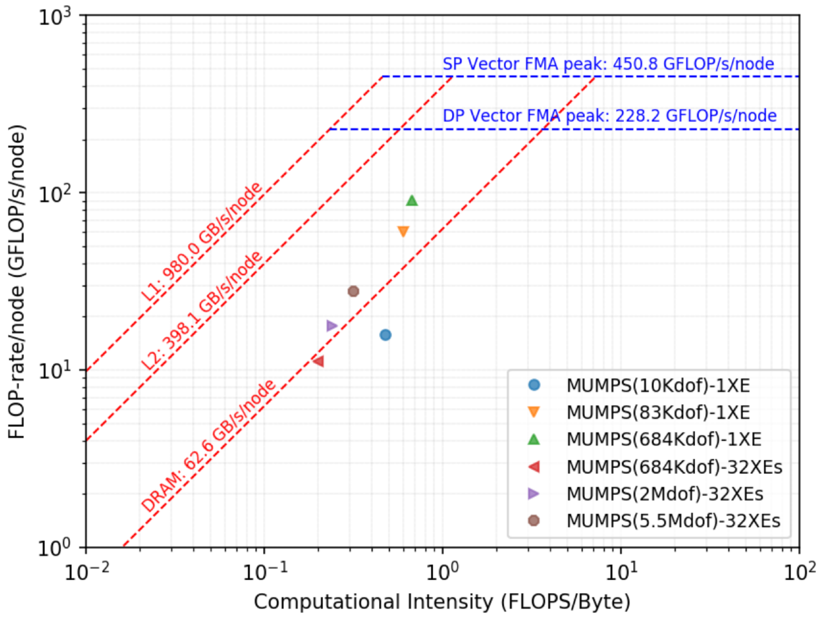
- Sparse iterative solver
  - PETSc[8]: cray-petsc/3.6.3.0
  - The transpose free QMR (i.e., tfqmr) solver
  - The classical additive Schwarz (i.e. asm_basic) preconditioner
- Sparse direct solver
  - MUMPS[9] and SuperLU[10]
  - cray-tpsl/16.03.1
- CSR matrices from a CFD code[11]
  - 10K, 83K, 684K and 2M unknowns
- Employed XE nodes
  - 16 MPI ranks on 1 XE node
  - 512 MPI ranks on 32 XE nodes

**PETSc: tfqmr solver with asm_basic preconditioner from PETSc**

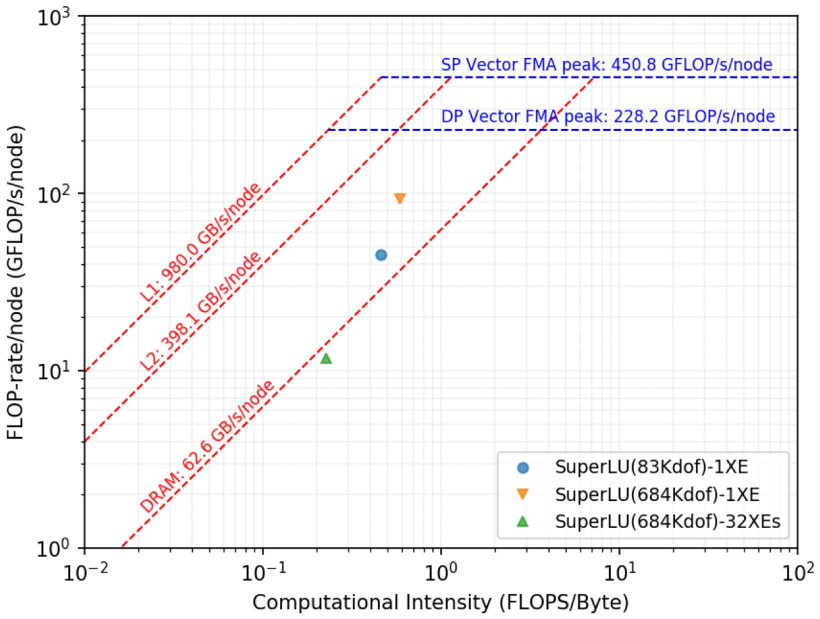


# Sparse Linear Solver

## MUMPS



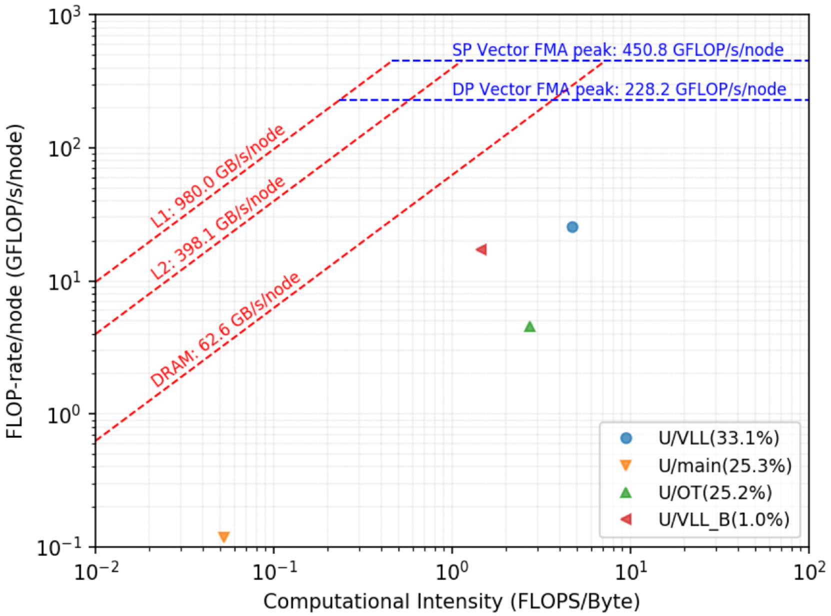

## SuperLU



# ROOFLINE-BASED PERFORMANCE PROJECTION FOR A NEW PROCESSOR

# Surface Extraction with TIN-based Search-space Minimization code[12]

- Four critical groups of SETSM code based on a CrayPat report on an XE node
  - U/VLL: 33.1% (997.6s), multi-threaded
  - U/main: 25.3% (671.3s), single-threaded
  - U/OT: 25.2% (757.8s), multi-threaded
  - U/VLL_B: 1.0% (31.1s), multi-threaded
  - Others: 15.4% (464.1s)
- Hypothesis for the performance projection
  - Performance of multi-threaded groups is bounded by DRAM bandwidth of the NODE
  - Performance of the single-threaded group is bounded by DRAM bandwidth of the NUMA domain



# Surface Extraction with TIN-based Search-space Minimization code[12]

- Wall time projections based on our hypothesis

$$WT_{target}^{threaded} = WT_{XE} * (BW^{NODE})_{XE} / (BW^{NODE})_{target}$$
$$WT_{target}^{serial} = WT_{XE} * (BW^{NUMA})_{XE} / (BW^{NUMA})_{target}$$
$$WT_{target}^{overall} = (\text{sum of } WT_{target}) / 0.846$$

- Target nodes at UIUC campus cluster

| Processor type | DRAM bandwidth per node (GB/s) (1) | DRAM bandwidth per NUMA (GB/s) |
|---|---|---|
| Dual Interlagos | 59.6 | 14.9(b) |
| Dual Ivy Bridge EP (E5-2670V2) | 93.5 | 46.7(c) |
| Dual Haswell (E5-2680V3) | 112.3 | 56.2(c) |
| Dual Broadwell (E5-2680V4) | 125.1 | 62.5(c) |

(a) Measured via the stream benchmark, (b) Four NUMA domains per node, (c) Two NUMA domains per node

- Roofline-based performance projection

| Processor type | Projected wall time (s) | Speed up over an XE |
|---|---|---|
| Dual Ivy Bridge EP (E5-2670V2) | 1715.3 | 1.76 |
| Dual Haswell (E5-2680V3) | 1427.7 | 2.11 |
| Dual Broadwell (E5-2680V4) | 1282.2 | 2.35 |

- Comparison with measured performance

| Processor type | Measured wall time (s) | Error of projection (%) |
|---|---|---|
| Dual Ivy Bridge EP (E5-2670V2) | 1603.0 | 7.00 |
| Dual Haswell (E5-2680V3) | 1293.0 | 10.42 |
| Dual Broadwell (E5-2680V4) | 1168.0 | 9.78 |

- What to do for more accurate projections
    - Characterizing more groups of the code

# Concluding remarks & future work

- A reliable and practical method for the roofline analysis model with CrayPat is proposed
  - It is compatible with various compilers (e.g., Cray, GNU and PGI compilers)
  - It can be extended to other processors (e.g., AMD, ARM, GPU and Intel)
  - It is scalable as much as CrayPat is.
- Selected hardware performance counters for the roofline analysis have been validated thoroughly via manually derived reference data
- CrayPat-based roofline analysis has been performed for HPCG, HPGMG, PETSc, MUMPS and SuperLU.
- An example of the roofline-based performance projections for other processors is presented.
- We plan to perform similar analyses with GPU nodes and other processors (e.g., ARM, and Intel).
- We will share our developed python scripts (General Roofline Evaluation Gadget – GREG) via NCSA public GitHub repository (https://github.com/ncsa/GREG)

# 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.

# References

1. B. Bode, M. Butler, T. Dunning, W. Gropp, T. Hoefler, W. Hwu, and W. Kramer (alphabetical), “The Blue Waters Super-System for Super-Science,” Contemporary HPC Architectures, Jeffery Vetter editor. Sitka Publications, November 2012.Edited by Jeffrey S . Vetter, Chapman and Hall/CRC 2013, Print ISBN: 978-1-4665-6834-1, eBook ISBN: 978-1-4665-6835-8.
2. W. Kramer, M. Butler, G. Bauer, K. Chadalavada and C. Mendes, “Blue Waters Parallel I/O Storage Sub-system,” High Performance Parallel I/O, Prabhat and Quincey Koziol editors, CRC Publications, Taylor and Francis Group, Boca Raton FL, 2015, Hardback Print ISBN 13:978-1-4665-8234-7.
3. G. Bauer, T. Hoefler, W. Kramer and R. Fiedler, "Analyses and Modeling of Applications Used to Demonstrate Sustained Petascale Performance on Blue Waters," Proceeding of the Cray Users Group meeting (CUG2012), Stuttgart, 2012.
4. G. Bauer, V. Anisimov, G. Arnold, B. Bode, R. Brunner, T. Cortese, R. Haas, A. Kot, W. Kramer, J. Kwack, J. Li, C. Mendes, R. Mokos, C. Steffen, "Updating the SPP Benchmark Suite for Extreme-Scale Systems," Proceedings of the Cray Users Group Meeting (CUG2017), Redmond, WA, May 2017.
5. CS Roofline Toolkit Bitbucket site, https://bitbucket.org/berkeleylab/cs-roofline-toolkit.git
6. J. Dongarra, M. Heroux and P. Luszczek “HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems,” Technical Report, Electrical Engineering and Computer Sciente Department, Knoxville, Tennessee, UT-EECS-15-736, November, 2015.
7. S. Williams, “4th Order HPGMG-FV Implementation,” HPGMG BoF, Supercomputing, November 2015.
8. S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, “PETSc Web page,” https://www.mcs.anl.gov/petsc, 2018.
9. P. R. Amestoy, A. Guermouche, J.-Y. L’Excellent and S. Pralet, “Hybrid scheduling for the parallel solution of linear systems,” Parallel Computing, 32(2),136-156, 2006.
10. X. S. Li and J. W. Demmel, “A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems,” ACM Trans. Mathematical Software, 29(2),110-140, 2003.
11. J. Kwack, G. Bauer and S. Koric, “Performance Test of Parallel Linear Equation Solvers on Blue Waters – Cray XE6/XK7 system,” Proceedings of the Cray Users Group Meeting (CUG2016), London, England, May 2016.
12. M.J.Noh,I.M.Howat,C.C.Porter,M.J.Willis,P.J.Morin,“Arctic Digital Elevation Models (DEMs) generated by Surface Extraction from TIN-Based Searchspace Minimization (SETSM) algorithm from RPCs-based Imagery,” American Geophysical Union, Fall General Assembly 2016.

# QUESTIONS ?