IDEAS PRODUCTIVITY
BEST PRACTICES FOR HPC SOFTWARE DEVELOPERS

EVALUATING PERFORMANCE PORTABILITY OF HPC APPLICATIONS AND BENCHMARKS ACROSS DIVERSE HPC ARCHITECTURES

JAEHYUK KWACK, COLLEEN BERTONI, YASAMAN GHADAR, THOMAS APPLENCOURT, HUIHUO ZHENG, JOHN TRAMM, BRIAN HOMERDING, ESTEBAN RANGEL, CHRISTOPHER KNIGHT, SCOTT PARKER

ALCF/CPS, Argonne National Laboratory

April 13th, 2022

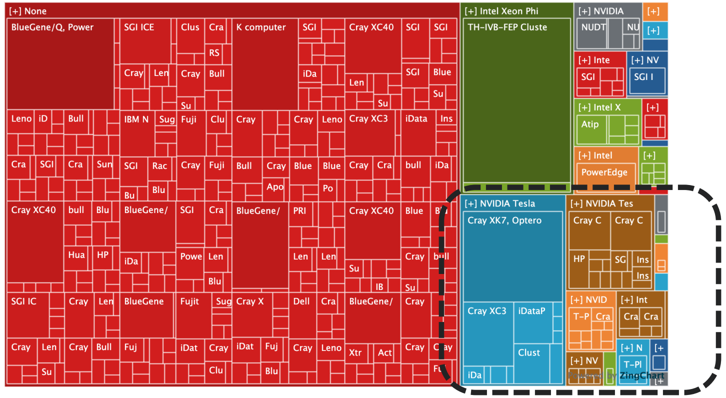
OVERVIEW

- HPC Architecture over time
- Performance study: Intel/ARM CPUs and NVIDIA GPU in 2018
- Performance Portability study: AMD, Intel and NVIDIA GPUs in 2021

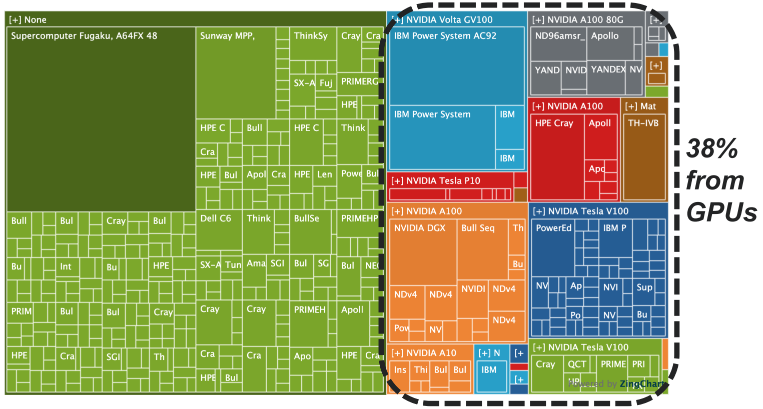
Reference: https://www.top500.org

HPC ARCHITECTURE OVER TIME

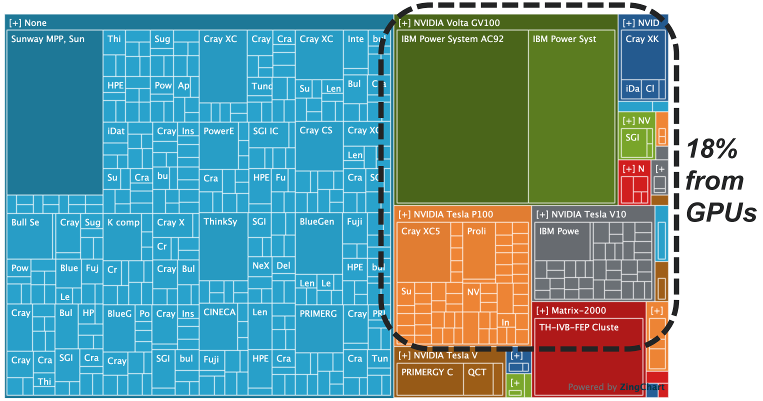
Peta-Scale era (2015)



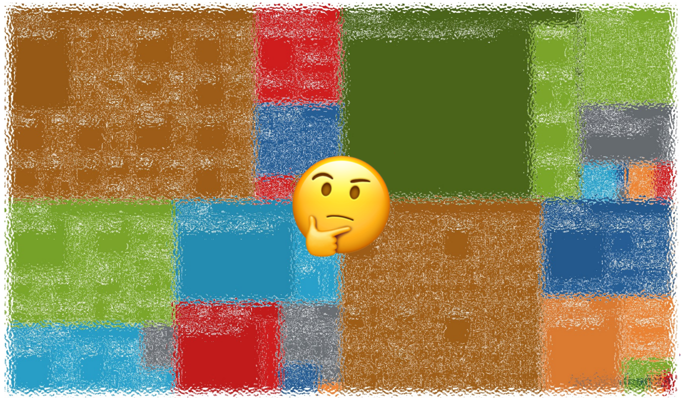
Almost ExaScale era (2021)



Pre-ExaScale era (2018)



ExaScale era (soon)



Department of Energy (DOE) Roadmap to Exascale Systems

An impressive, productive lineup of *accelerated node* systems supporting DOE's mission

Pre-Exascale Systems

First U.S. Exascale Systems*

2012 **2016** **2018** **2020** **2021-2023**

ORNL
Cray/AMD/NVIDIA

ORNL
IBM/NVIDIA

FRONTIER

ORNL*
HPE/AMD

ANL
IBM BG/Q

ANL
Cray/Intel KNL

ANL*
Intel/HPE

LBNL
Cray/Intel Xeon/KNL

LBNL
HPE/AMD/NVIDIA

More GPU-accelerated systems

LLNL
IBM BG/Q

LANL/SNL
Cray/Intel Xeon/KNL

LLNL
IBM/NVIDIA

CROSSROADS

LANL/SNL
HPE/Intel

LLNL*
HPE/AMD

ECP EXASCALE COMPUTING PROJECT

Reference: https://science.osti.gov/-/media/bes/besac/pdf/201907/1330_Diachin_ECP_Overview_BESAC_201907.pdf

WHAT WE ARE LOOKING FOR…

- Most application developers would like to have one programming model
  - that would run everywhere (i.e., portability)
  - and give decent performance (i.e., performance portability)
  - ,so they can produce more scientific/engineering outputs. (i.e., productivity)

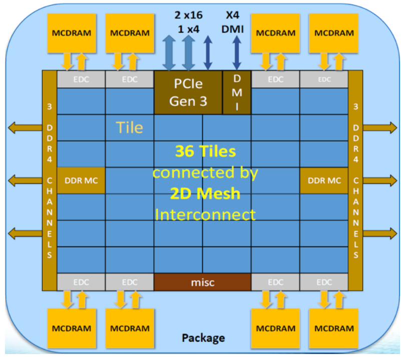
- Unfortunately, I don’t think we have that.

- This talk is about our effort to explore the past and current state of things in terms of
  - basic portability for programming models,
  - how to evaluate performance,
  - and how to approach understanding performance portability

PERFORMANCE STUDY IN 2018 (INTEL/ARM CPUS AND NVIDIA GPUS)

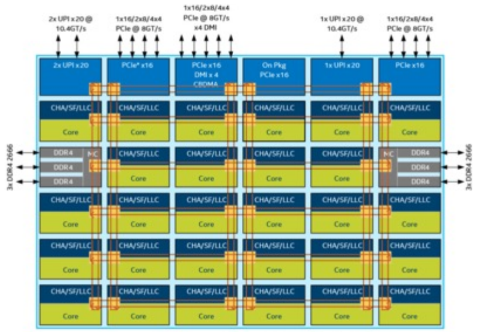
JaeHyuk Kwack, Thomas Applencourt, Colleen Bertoni, Yasaman Ghadar, Huihuo Zheng, Christopher Knight, and Scott Parker

EMPLOYED ARCHITECTURES IN 2018



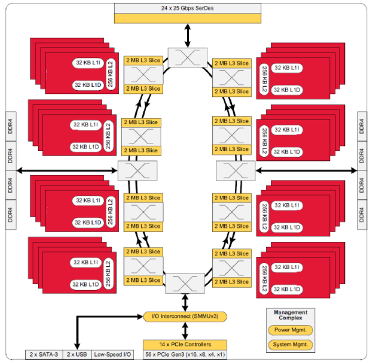
Intel Xeon Phi KNL processor

- Intel KNL 7320 processor
- 16 GB MCDRAM memory w/ 192 GB DDR memory
- 32 tiles w/ 2 cores/tile
- AVX-512 instructions
- 1.3 GHz reference frequency



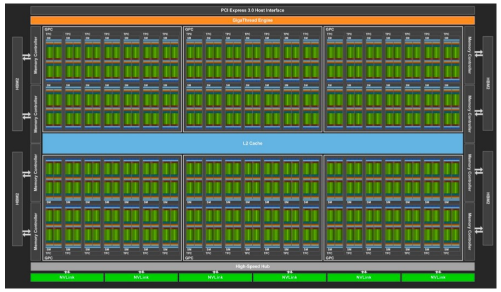
Intel Xeon Skylake processor

- Intel Xeon 8180M processor
- 395 GB DDR memory/node
- 28 core x86 processor (14 nm+)
- 2 AVX-512 FMA units/core
- 2.5 GHz reference frequency



ARM Thunder X2 processor

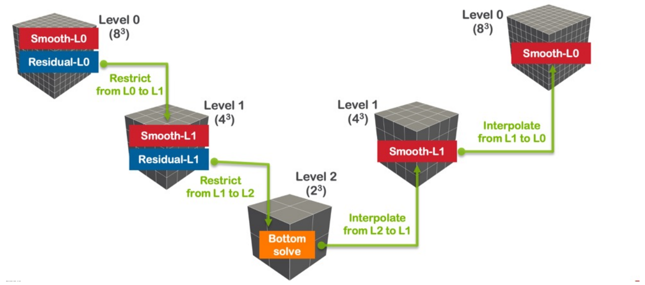
- Arm Marvell ThunderX2 CN9975 processor
- 217 GB DDR memory/node
- 28 core Arm v8.1 processor (16 nm+)
- 2 NEON 128-vectors engines/core
- 2.2 GHz reference frequency (2.5 GHz on Turbo mode)

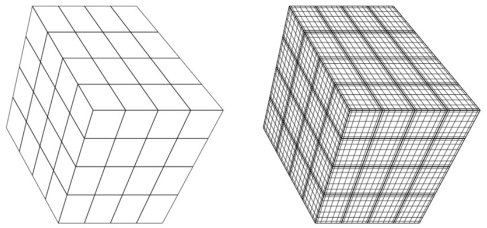


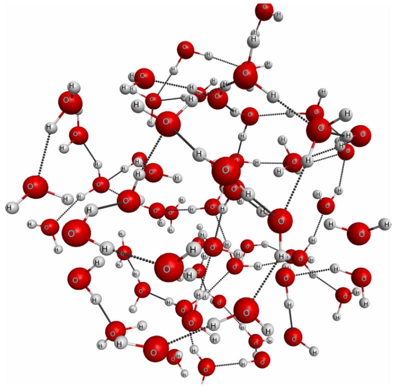
NVIDIA V100 GPU

- NVIDIA V100 SXM2 GPU
- 32 GB HBM memory
- 80 SMs with 32 FP64 CUDA cores/SM and 8 Tensor cores/SM
- 1.53 GHz maximum frequency

ECP BENCHMARKS AND APPLICATIONS





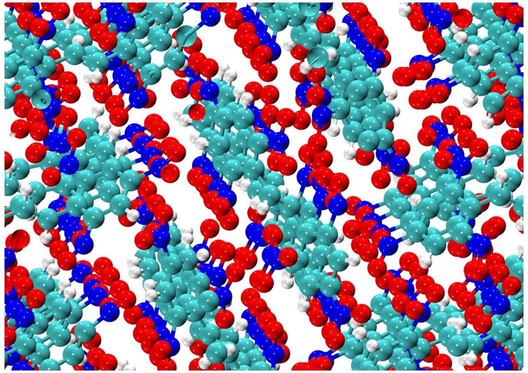


- HPGMG-FV
  - Solving an elliptic problem on isotropic Cartesian grids with 4th order accuracy
  - Employing the Full Multi-grid (FMG) F-cycle
  - A series of progressively deeper geometric multi-grid V-cycles
  - MPI+OpenMP for CPU
  - MPI+CUDA for GPU

- Nekbone
  - A mini-app derived from the Nek5000 CFD code (a high order, incompressible Navier-Stokes CFD solver based on the spectral element method)
  - Standard Poisson equation in a 3D box domain with a block spatial domain decomposition among MPI ranks.
  - MPI+OpenMP for CPU

- GAMESS
  - A general quantum chemistry and ab initio electronic structure code (e.g., ab initio SCF energies, force fields, perturbative corrections to Hartree-Fock, near-linear scaling fragmentation methods, and so on)
  - Mainly written in Fortran
    - A MPI parallelization library (DDI library) written in C
    - MPI + OpenMP for CPU
    - MPI + CUDA for GPU

ECP BENCHMARKS AND APPLICATIONS



- LAMMPS
  - A molecular simulation code commonly used for modeling various states of matter (liquids, surfaces, solids, biopolymers) and supports multiple physical models, particle types, and sampling methods
  - MPI + OpenMP for CPU
  - MPI + Kokkos for GPU

QMCPACK

- QMCPACK
  - An ECP application for ab-initio electronic structure calculations
  - Each OpenMP thread executes an independent Markov chains or walkers. After each walker has completed a number steps, the simulation is completed.
  - Version: QMCPACK v3.7.0 with SoA (i.e., Structure-of-Array)
  - Input (a.k.a. S32)
    - 32 repeats of a NiO primitive cell leading to 128 atoms and 1536 electrons

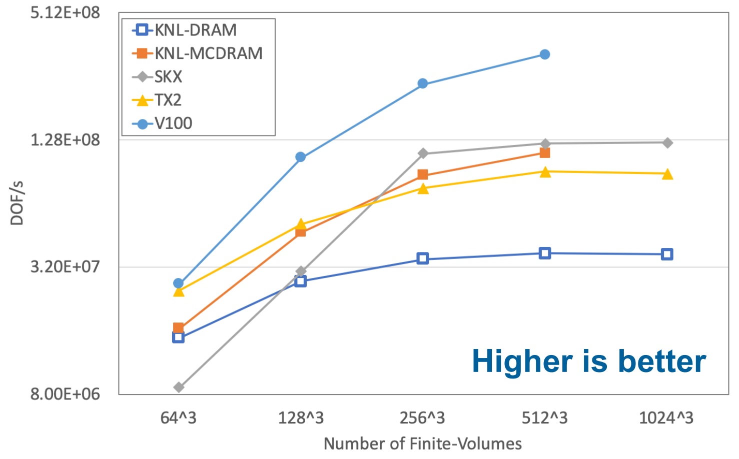
Qbox
First-Principles Molecular Dynamics

- QBOX
  - A C++ MPI/OpenMP scalable parallel implementation of first-principles molecular dynamics based on the plane-wave, pseudopotential density functional theory formalism
  - Using FFTW for 3D Fast Fourier Transformation and ScaLAPACK for parallel dense linear algebra.
  - Linking against the vendor provided libraries for FFT and ScaLAPACK
  - MKL on SKX and KNL
  - ArmPL on TX2

HPGMG-FV

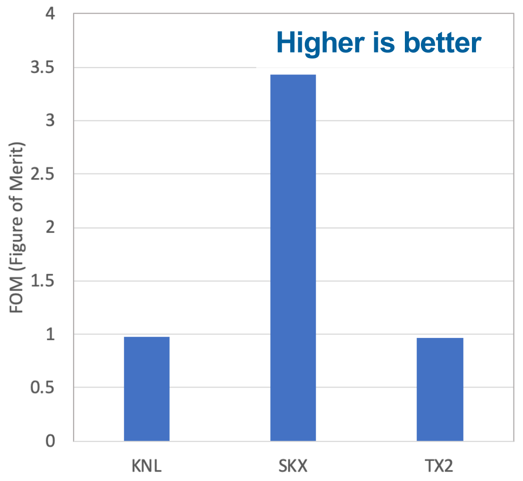
- Source
  - MPI+OpenMP version (commit: a0a5510) MPI+CUDA version (commit: 5ad473d)
- Runtime configurations

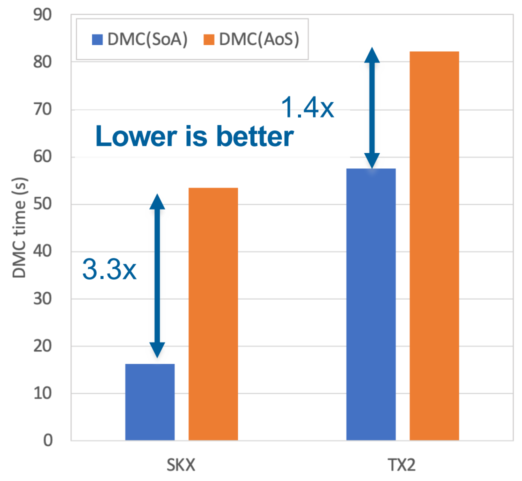
Processor	Number of MPI ranks	Number of Threads per MPI rank	Total Threads
KNL	64	1	64
SKX	16	7	112
TX2	16	7	112
V100	1	7	all GPU cores



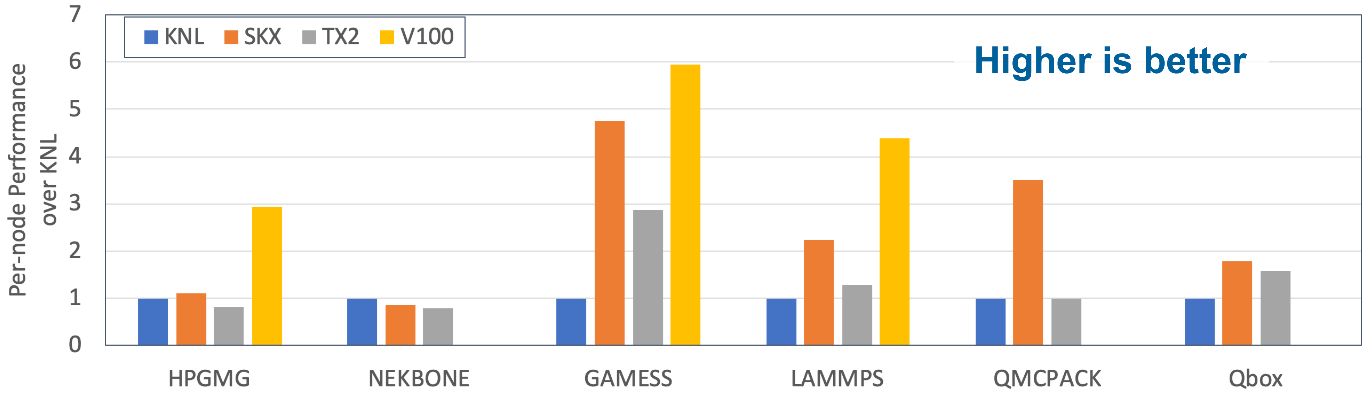
QMCPACK

- Figure of merit (FOM): how many walkers have been moved in one second.
- Different impacts of SoA optimization on SKX and TX2
- SoA (Structure-of-Array) vs. AoS (Array-of-Structure)
  - The performance gain by SoA depends on the data cache performance.
  - The speedup by SoA is much higher on SKX than on TX2, because the data cache performance of SKX is much better than the cache performance of TX2.



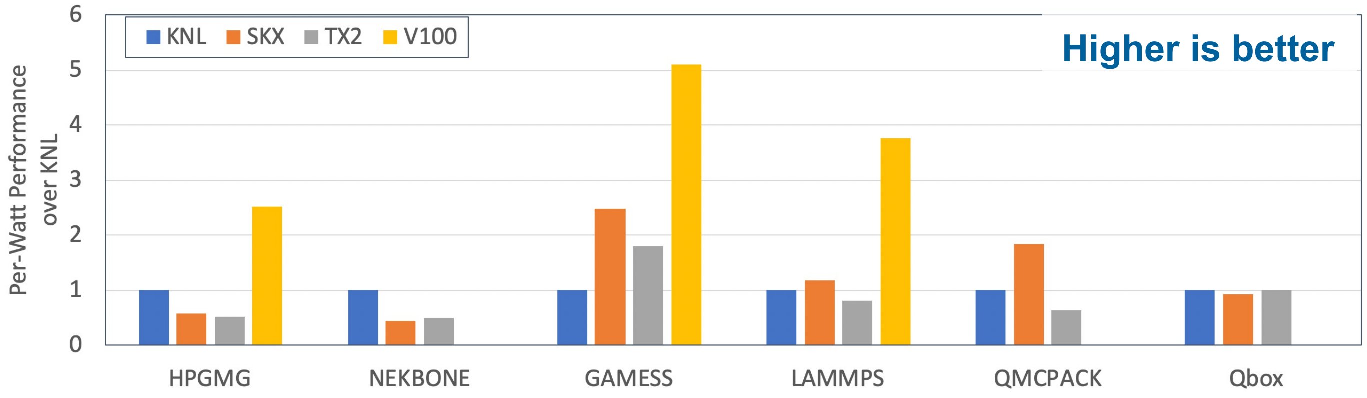


Per-node performance



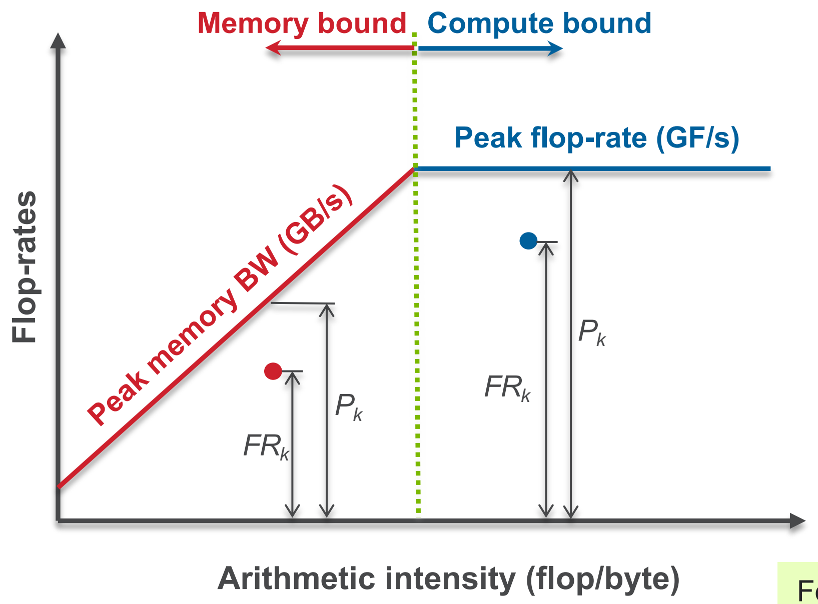
Per-watt performance

- TDP (Thermal Design Power)
  - KNL: 215W/socket, 215W/node
  - SKX: 205W/socket, 410W/node
  - TX2: 170W/socket, 340W/node
  - V100: 250W/socket



ROOFLINE EFFICIENCY

Evaluating roofline efficiency using profiling tools



- FR_k : measured flop-rates of a kernel
- P_k (the maximum attainable performance for the kernel)

$$= \min\begin{cases}\text{peak memory BW} * \text{arithmetic intensity} \\ \text{peak flop−rate}\end{cases}$$

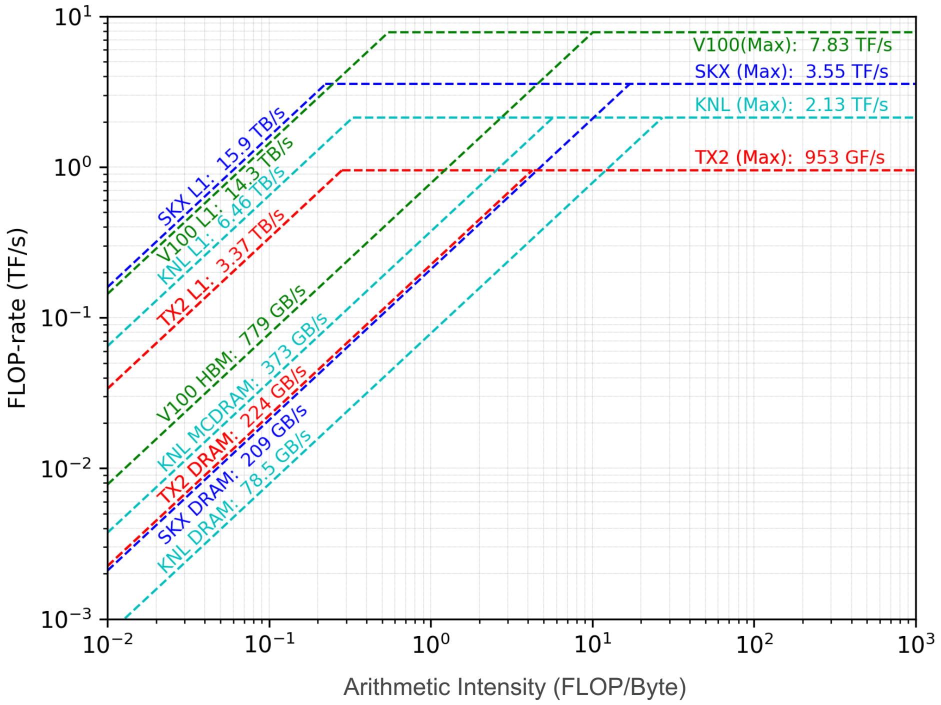
- Roofline efficiency $E_k = FR_k \,/\, P_k$
  - equivalent to bandwidth efficiency for memory bound kernel
  - equivalent to flop-rate efficiency for compute bound kernel

For more details, please check the following IDEAS HPC-BP webinars
- Using the Roofline Model and Intel Advisor (8/16/2017)
- Quantitatively Assessing Performance Portability with Roofline (1/23/2019)

MEASURED PEAK PERFORMANCE

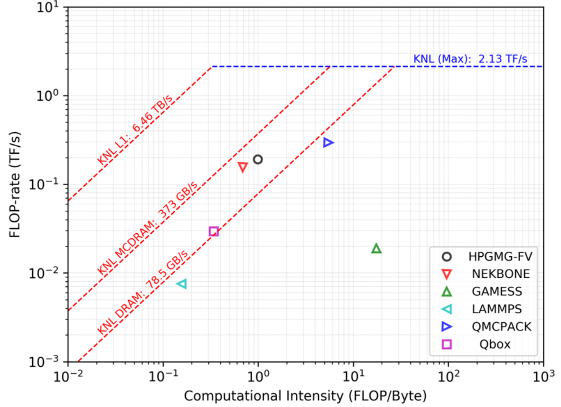
- Measured via Empirical Roofline Tool
  - TX2 peak flop-rate from DGEMM
  - V100 L1 is the theoretical peak.

	Flop-rate (TF/s)	L1 (TB/s)	L2 (TB/s)	LLC (GB/s)	DRAM (GB/s)
KNL	2.13	6.46	1.911	373	78.5
Dual SKX	3.55	15.91	4.55		209
Dual TX2	0.953	3.37	2.63	1091	224
V100	7.83	14.336	3.35		779

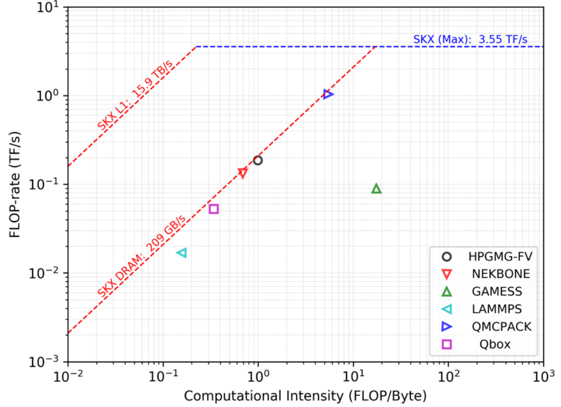


ROOFLINE-BASED PERFORMANCE EFFICIENCY

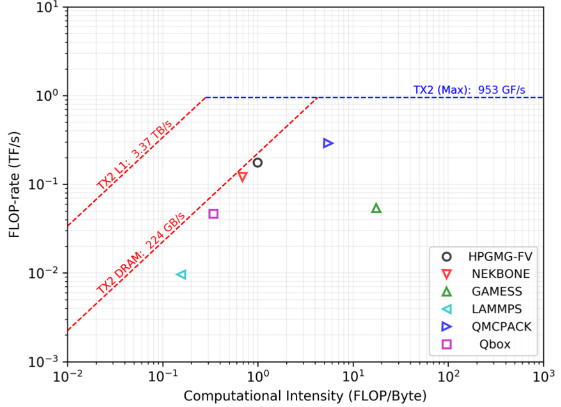
Intel Xeon Phi KNL processor



Intel Xeon Skylake processor

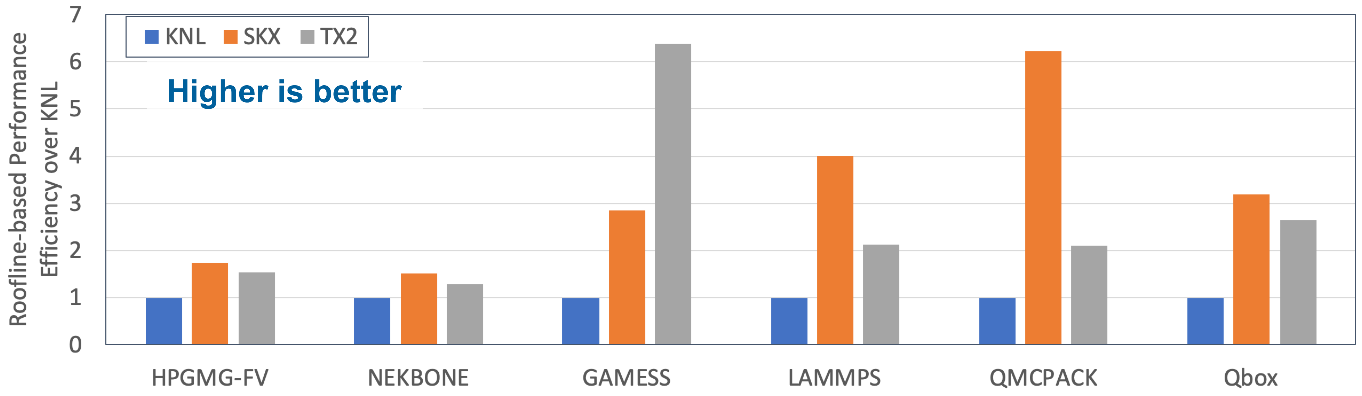


ARM Thunder X2 processor



Relative Roofline-based Performance Efficiency

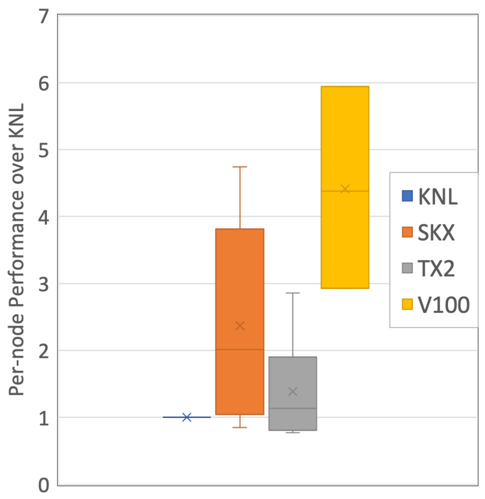
	KNL	SKX	TX2
HPGMG-FV	1.00	1.73	1.54
NEKBONE	1.00	1.52	1.29
GAMESS	1.00	2.85	6.39
LAMMPS	1.00	4.00	2.13
QMCPACK	1.00	6.21	2.10
Qbox	1.00	3.18	2.64



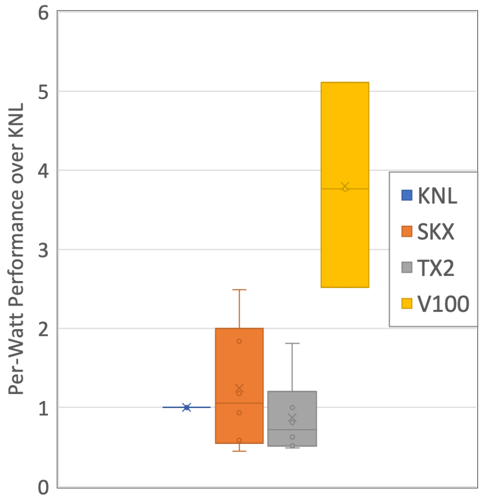
SUMMARY

- Executed performance tests
  - for 2 HPC benchmarks (i.e., HPGMG-FV, and NEKBONE) and 4 HPC applications (i.e., GAMESS, LAMMPS, QMCPACK, and Qbox)
  - on four types of processor architectures (i.e., KNL, SKX, TX2 and V100)

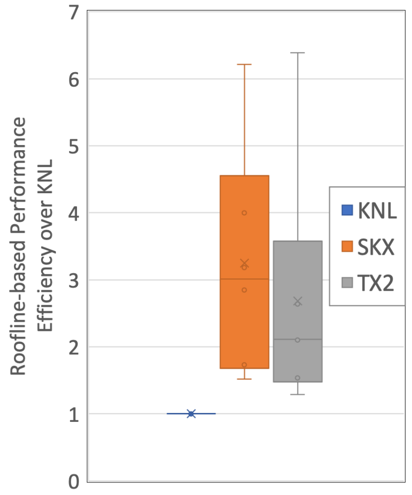
Per-node performance



Per-watt performance



Roofline-based Efficiency



CHALLENGES WITH PORTABILITY IN 2018

Limited demand on portability

- Multiple CPU vendors (e.g., AMD, ARM, IBM, Intel, and so on)
  - C, C++, and Fortran with OpenMP were well supported by most of vendors including LLVM, and GNU
  - Mostly portable across CPUs
- Single GPGPU vendor (i.e., NVIDIA)
  - CUDA for the best performance from NVIDIA without portability
  - OpenACC for portability between CPUs and GPUs from a limited number of vendors (e.g., PGI, Cray, GNU)
- Portability layers from HPC community
  - Kokkos and RAJA with OpenMP and CUDA backends
- Many application developers considered CPUs as their primary architecture, while several developers managed additional branch for GPGPU.

CHANGES AND IMPROVEMENTS IN 2021

More demand on portability

- Multiple CPU vendors (e.g., AMD, ARM, IBM, Intel, and so on)
  - C, C++, and Fortran with OpenMP have been well supported by most of vendors including LLVM, and GNU
  - Mostly portable across CPUs
- ~~Single~~ Multiple GPGPU vendors (e.g., NVIDIA, AMD, and Intel)
  - CUDA for the best performance from NVIDIA without portability
  - OpenACC for portability between CPUs and GPUs from a limited number of vendors (e.g., NVIDIA, HPE, GNU)
  - OpenMP Target Offloading support for GPUs by multiple vendors (e.g., AMD, GNU, HPE, IBM, Intel, LLVM, NVIDIA)
  - SYCL and HIP for AMD/Intel/NVIDIA GPUs
- Increased use of portability layers from HPC community
  - Kokkos and RAJA with OpenMP, CUDA, HIP and SYCL backends for CPUs and AMD/Intel/NVIDIA GPUs
- More application developers consider GPGPUs as their primary architecture for the best performance.
- New challenge is to make their applications performance portable across multiple GPGPU architectures.

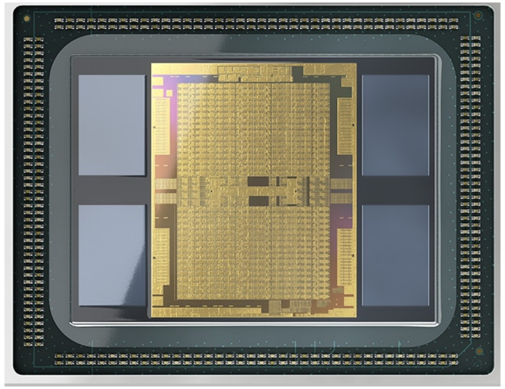
PERFORMANCE PORTABILITY STUDY IN 2021 (AMD, INTEL AND NVIDIA GPUS)

JaeHyuk Kwack, John Tramm, Colleen Bertoni, Yasaman Ghadar, Brian Homerding, Esteban Rangel, Christopher Knight, Scott Parker

WHY PERFORMANCE PORTABILITY ON GPUS?

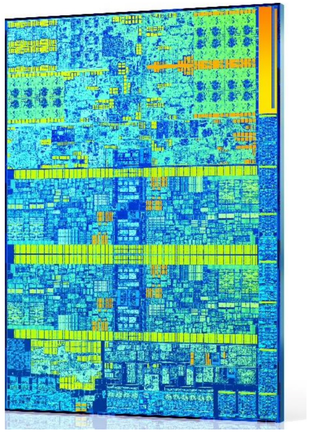
- Accelerator-based systems are one of the dominant designs in the exascale era
  - New NVIDIA GPU systems (NERSC/Perlmutter, CINECA/Leonardo, Argonne/Polaris)
  - New Intel GPU systems (Argonne/Aurora, LRZ/SuperMUC-NG phase II)
  - New AMD GPU systems (Oak Ridge/Frontier, Lawrence-Livermore/El Capitan, CSC-IT/LUMI)
- It is a great challenge for developers attempting to make their applications portable across those HPC platforms
- US DOE has supported 21 projects with more than three dozen applications for coming exascale systems via Exascale Computing Project (ECP).
  - What is the best way to assess the application performance across the systems?
- In this study, we investigate performance portability of a subset of ECP applications and related mini-apps across AMD, Intel and NVIDIA GPUs.

EMPLOYED GPU SYSTEMS



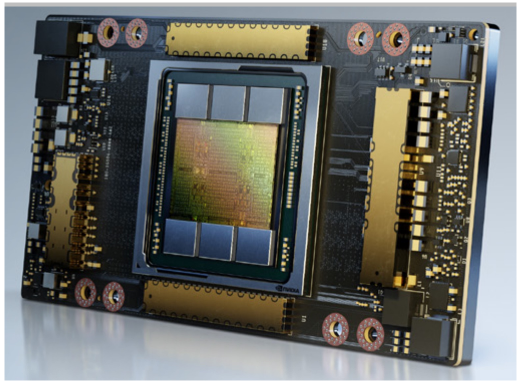
AMD MI100 GPU (credit: AMD)

- 32 GB HBM2 memory
- 120 compute units with 7,680 stream processors
- Up to 11.5 TF/s with FP64



Intel Gen9 GPU (credit: Intel)

- 64 GB DDR4 memory with 128 MB eDRAM memory
- 9 subslices with 72 execution units (EUs)
- Up to 331 GF/s with FP64

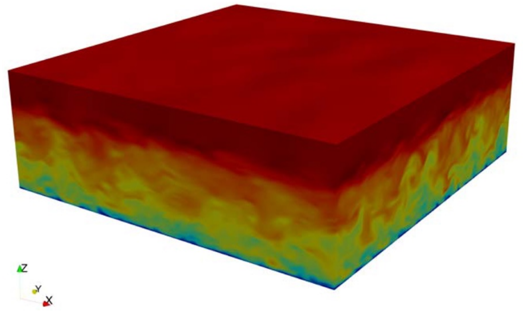


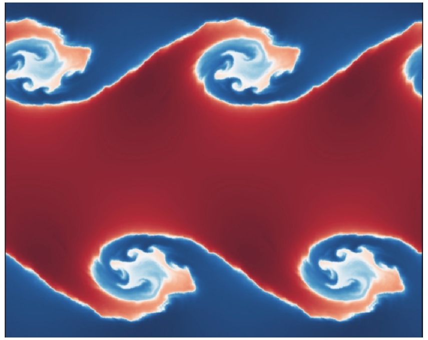
NVIDIA A100 GPU (credit: NVIDIA)

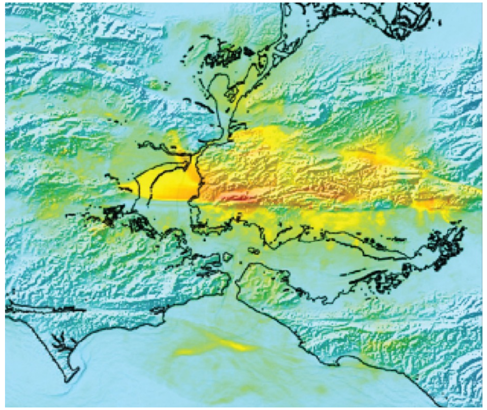
- 40 GB HBM2 memory
- 108 SMs with 6912 CUDA cores and 432 Tensor cores
- Up to 9.7 TF/s with FP64

Remark: *Intel X^e brand high performance discrete GPUs are not publicly available at the time of this study. The integrated Gen9 GPU is therefore the most suitable Intel GPU for evaluation of HPC applications currently available.*

ECP APPLICATIONS







- AMR-Wind
  - ECP ExaWind project for wind farm simulations
  - A structured-grid CFD background solver
  - ***AMReX* framework** serves as a portability layer
  - Tested atmospheric boundary layer (ABL) flows in a cubic box

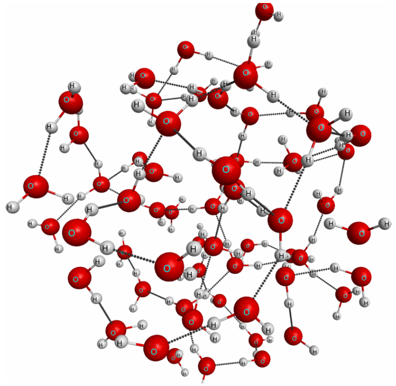
- HACC CRK-SPH
  - ECP ExaSky project for cosmological simulations
  - CRK-SPH to resolve gas dynamics
  - CUDA codes are migrated to ***DPC++/SYCL*** programming model by Intel DPCT
  - Tested 8 rank N-body simulation

- SW4
  - ECP EQSim project for regional-scale ground motion simulations
  - SW4 is for seismic wave propagation
  - ***RAJA* portability layer** is used with CUDA, HIP and SYCL execution policies
  - Tested a topology near Berkeley, CA

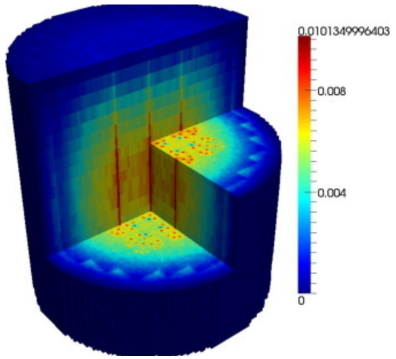
Image sources

AMR-Wind:	https://www.nrel.gov/wind/assets/pdfs/future-of-hpc-webinar-2020-07-30.pdf
HACC:	https://www.sciencedirect.com/science/article/pii/S0021999116306453
SW4:	https://geodynamics.org/cig/software/sw4/

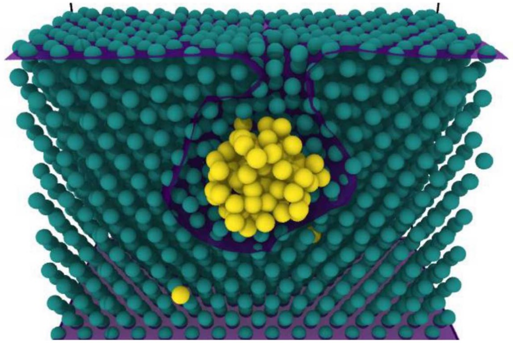
ECP MINI-APPS



- RI-MP2 (GAMESS)
  - ECP GAMESS project for quantum chemistry methods
  - RI-MP2 is a perturbative correction to HF
  - ***OpenMP target offloading*** is used on GPUs
  - Tested the energy reduction kernel in this study



- XSBench (OpenMC)
  - ECP ExaSMR project for modular nuclear reactor simulations
  - Represents the MC transport method
  - Ported to multiple prog. models
  - Tested a code with ***OpenMP target offloading*** in this study



- TestSNAP (LAMMPS)
  - ECP EXAALT project for molecular dynamics simulations
  - A mini-app for the SNAP potential from LAMMPS
  - Ported to multiple prog. models
  - Tested ***Kokkos*** implementation in this study

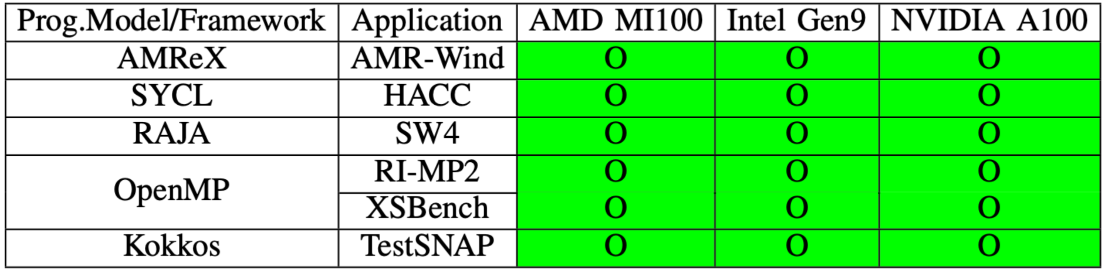
Image sources

RI-MP2(GAMESS):	https://github.com/jkwack/GAMESS_RI-MP2_MiniApp
XSBench(OpenMC):	https://www.sciencedirect.com/science/article/pii/S030645491400379X
TestSNAP(LAMMPS):	https://www.osti.gov/servlets/purl/1645897

PORTABILITY

Green lights for portability across AMD, Intel and NVIDIA GPUs

- All of the applications, mini-apps, and their associated kernels have been demonstrated to run across NVIDIA, AMD, and Intel GPUs.
- All of the portability approaches employed (SYCL, OpenMP, Kokkos, RAJA, and AMReX) have therefore been successful in enabling portability.



Prog.Model/Framework	Application	AMD MI100	Intel Gen9	NVIDIA A100
AMReX	AMR-Wind	O	O	O
SYCL	HACC	O	O	O
RAJA	SW4	O	O	O
OpenMP	RI-MP2	O	O	O
	XSBench	O	O	O
Kokkos	TestSNAP	O	O	O

PERFORMANCE PORTABILITY

Yes, portable. Performance portable?

- How to assess the performance portability of the applications
- Pennycook's performance portability metric (PPM)

$$
\text{¶P}_k(H) = \begin{cases} \dfrac{|H|}{\sum_{i \in H} \dfrac{1}{E_k(i)}}, & \text{if } i \text{ is supported } \forall i \in H; \\ 0, & \text{otherwise.} \end{cases}
$$

  - PPM is a harmonic mean of efficiency ($E_k(i)$).
    - PPM is a good metric to represent overall efficiency across a set of platforms (H).
  - Two types of efficiencies recommended
    - Architectural efficiency: the achieved performance as a fraction of peak hardware performance
    - Application efficiency: the achieved performance as a fraction of best observed performance based on the most optimized implementation

EVALUATION OF PERFORMANCE PORTABILITY

Performance portability metrics w/ roofline efficiency

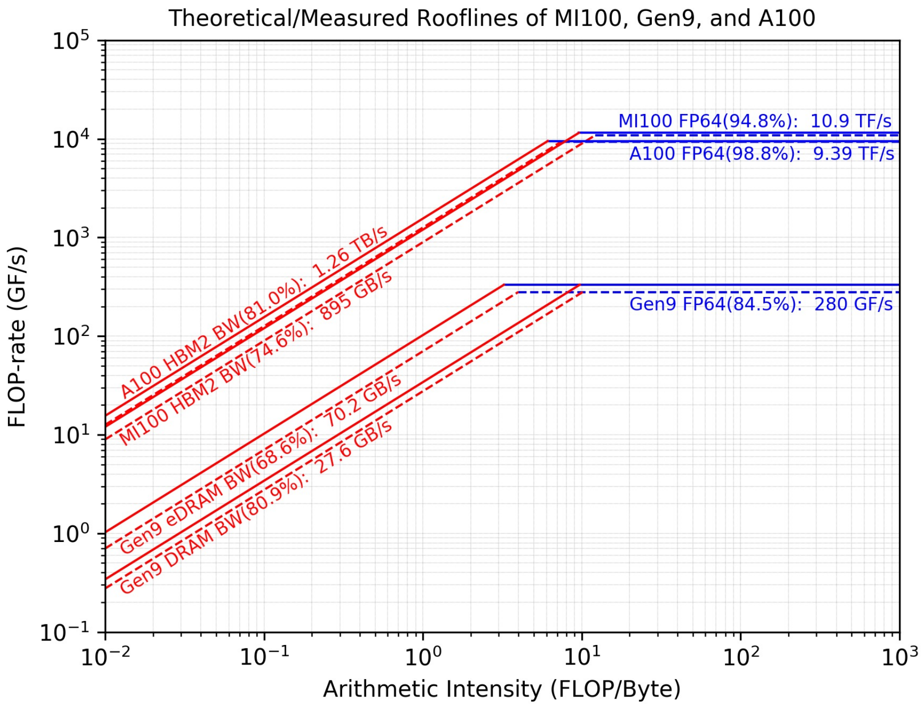
- Challenges in using architectural or application efficiency for PPM
  - Peak flop-rates may be too restrictive to represent the peak hardware performance for architectural efficiency
  - Requiring a determination of the relevant bottleneck on each hardware platforms for actual architectural efficiency
  - Or, requiring development of a fully optimized kernel implementation for each hardware platform for application efficiency

- Roofline efficiency can be used as an approximation for architectural efficiency
  - Considering kernels are highly performance portable if they fully utilize peak memory bandwidths or peak flop-rates on a set of platforms of interest.
  - Better than using peak flop-rates for architectural efficiency
  - For kernels with low roofline efficiencies, further investigation for performance portability may be performed in addition.
  - Doesn't require development of a fully optimized kernel implementation for each hardware platform

MEASURED PEAKS FOR ROOFLINE ANLAYSIS

Measured peaks and rooflines of the GPUs

GPU	FP64 (TF/s)	BW (TB/s)	Balance (F/B)	FP64 effi. (%)	BW effi. (%)
AMD MI100	10.9	0.895 (HBM2)	12.2	94.8	74.6
Intel Gen9	0.280	0.0702 (eDRAM)	4.0	84.6	68.8
		0.0276 (DRAM)	10.1		80.8
NVIDIA A100	9.39	1.26 (HBM2)	7.5	98.8	81.0



Measured via Empirical Roofline Toolkit (ERT)
Compiler version:

- hipcc version 4.3.21300-5bbc51d8 for AMD
- dpcpp vesion 2021.4.0 for Intel
- nvcc version 11.3.109 for NVIDIA

PROFILING TOOLS FOR ROOFLINE ANALYSIS

Intel Advisor, NVIDIA Nsight, ROCm Profiler

Intel Advisor

Intel Advisor has supported GPU roofline analysis features; it uses a binary instrumentation tool, GT-Pin for FLOP counts, and its overhead is relatively higher than using hardware performance counters.

NVIDIA Nsight

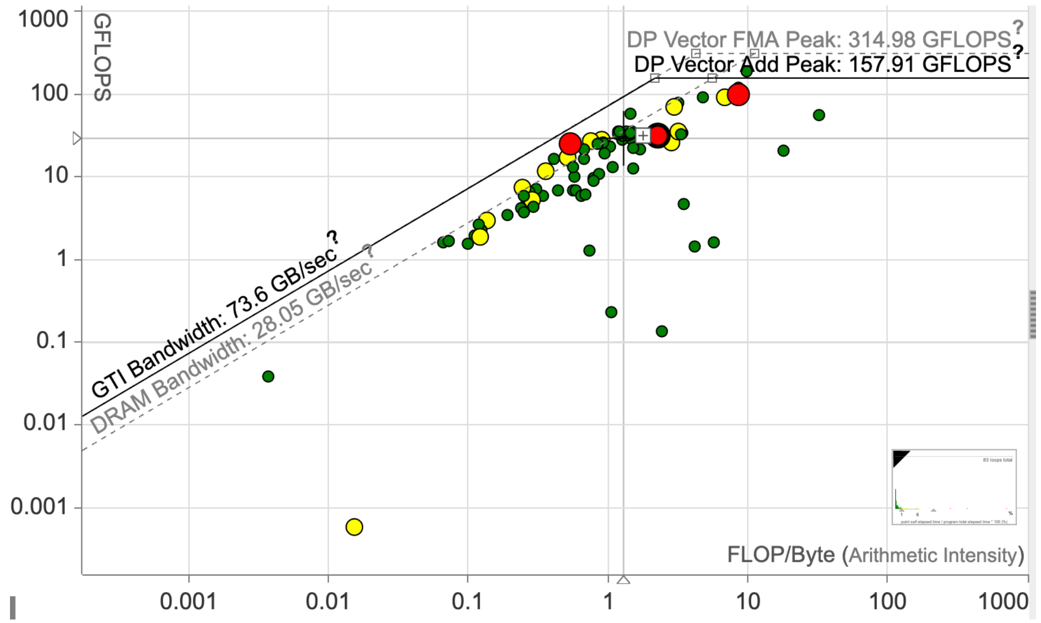
NVIDIA Nsight Compute provides roofline analysis features; it supports CUDA and OpenMP target offloading models; however, OpenCL is not supported by NVIDIA tools, while OpenCL applications are portable on NVIDIA GPUs. We hope NVIDIA tools will support OpenCL programming model soon.

AMD ROCm profiler

AMD ROCm profiler is used to collect performance data from hardware counters and derived metrics. Since MI100 has no dedicated FLOP counters, we assumed that FLOP counts on MI100 are similar to FLOP counts on A100. We hope the next generation of AMD GPUs and SDK will provide a reliable method for FLOP measurements.

AMR-WIND

An example of processing roofline performance analysis data

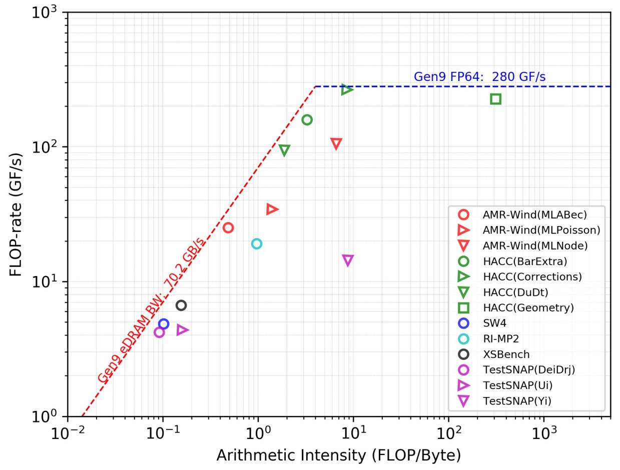


- AMR-Wind roofline-based performance data

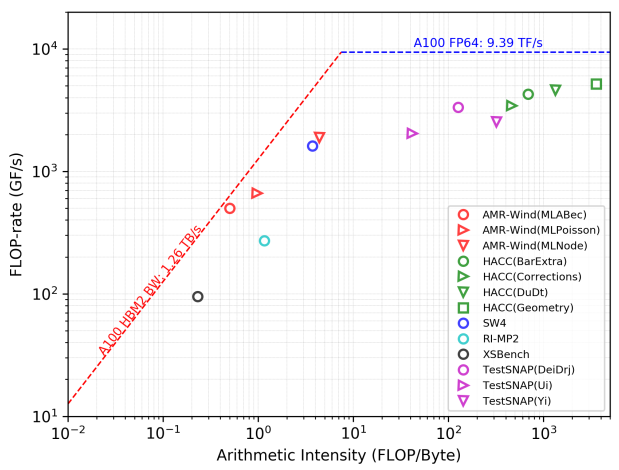
GPU	Kernel	AI_k	FR_k (GF/s)	Bound	P_k (GF/s)
Intel Gen9	MLABecLaplacian::Fsmooth[a]	0.483	25.2	Memory	33.9
	MLPoisson::Fsmooth[a]	1.40	34.4	Memory	98.5
	MLNodeLaplacian::Fsmooth[b]	6.61	106.	Compute	280.
NVIDIA A100	MLABecLaplacian::Fsmooth[a]	0.5	498.	Memory	630.
	MLPoisson::Fsmooth[a]	0.97	661.	Memory	1220
	MLNodeLaplacian::Fsmooth[b]	4.4	1880	Memory	5530
AMD MI100 (estimated)	MLABecLaplacian::Fsmooth[a]	0.523	200.	Memory	467.
	MLPoisson::Fsmooth[a]	1.65	125.	Memory	1470
	MLNodeLaplacian::Fsmooth[b]	3.63	1120	Memory	3250

ROOFLINE PLOTS

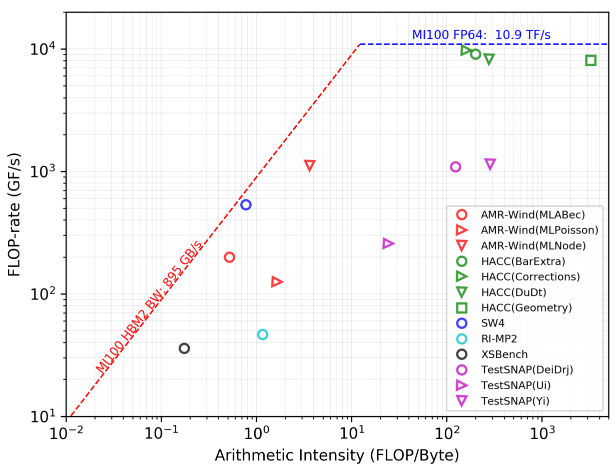
Measured on Intel, NVIDIA GPUs, and Estimated on AMD GPU



Intel Gen9 GPU



NVIDIA A100 GPU

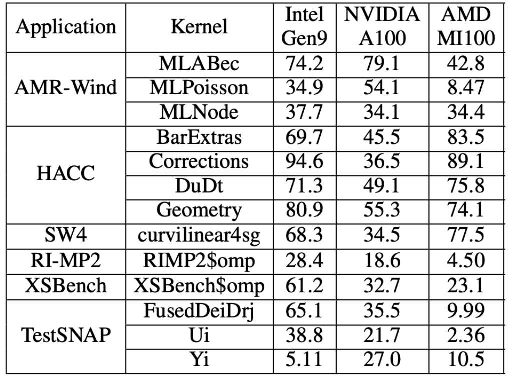


AMD MI100 GPU (estimated)

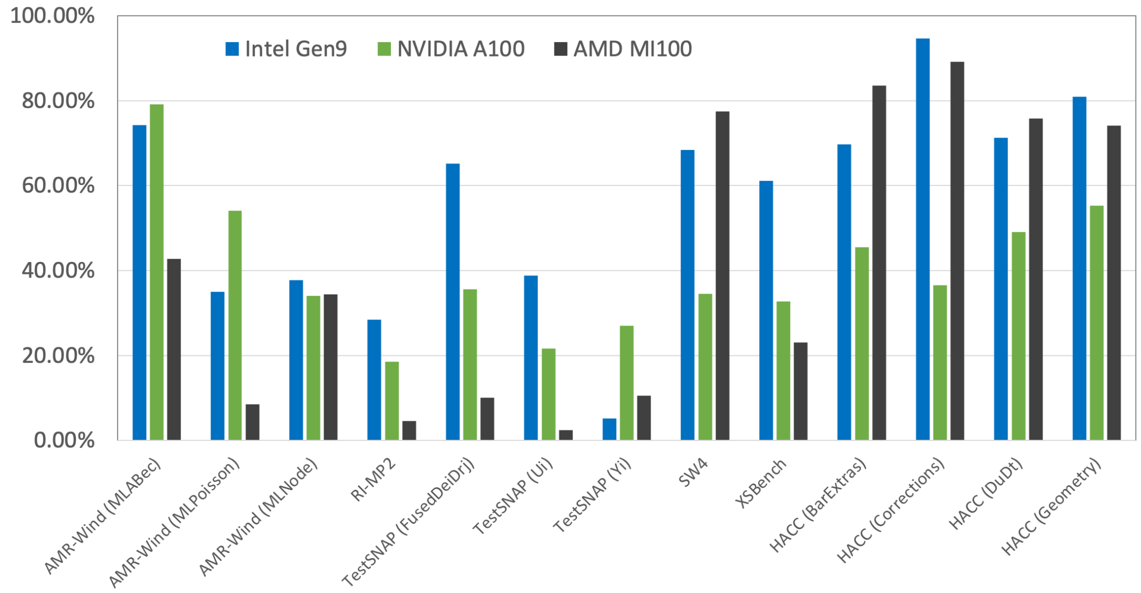
ROOFLINE EFFICIENCY

- Roofline efficiencies on AMD, Intel and NVIDIA GPUs are computed for kernels of interest

- Average efficiency on GPUs
  - Intel Gen9: 56%
  - NVIDIA A100: 40%
  - AMD MI100: 41%
  - We think it is due to the smaller size of Intel GPU used in this study



Application	Kernel	Intel Gen9	NVIDIA A100	AMD MI100
AMR-Wind	MLABec	74.2	79.1	42.8
	MLPoisson	34.9	54.1	8.47
	MLNode	37.7	34.1	34.4
HACC	BarExtras	69.7	45.5	83.5
	Corrections	94.6	36.5	89.1
	DuDt	71.3	49.1	75.8
	Geometry	80.9	55.3	74.1
SW4	curvilinear4sg	68.3	34.5	77.5
RI-MP2	RIMP2$omp	28.4	18.6	4.50
XSBench	XSBench$omp	61.2	32.7	23.1
TestSNAP	FusedDeiDrj	65.1	35.5	9.99
	Ui	38.8	21.7	2.36
	Yi	5.11	27.0	10.5

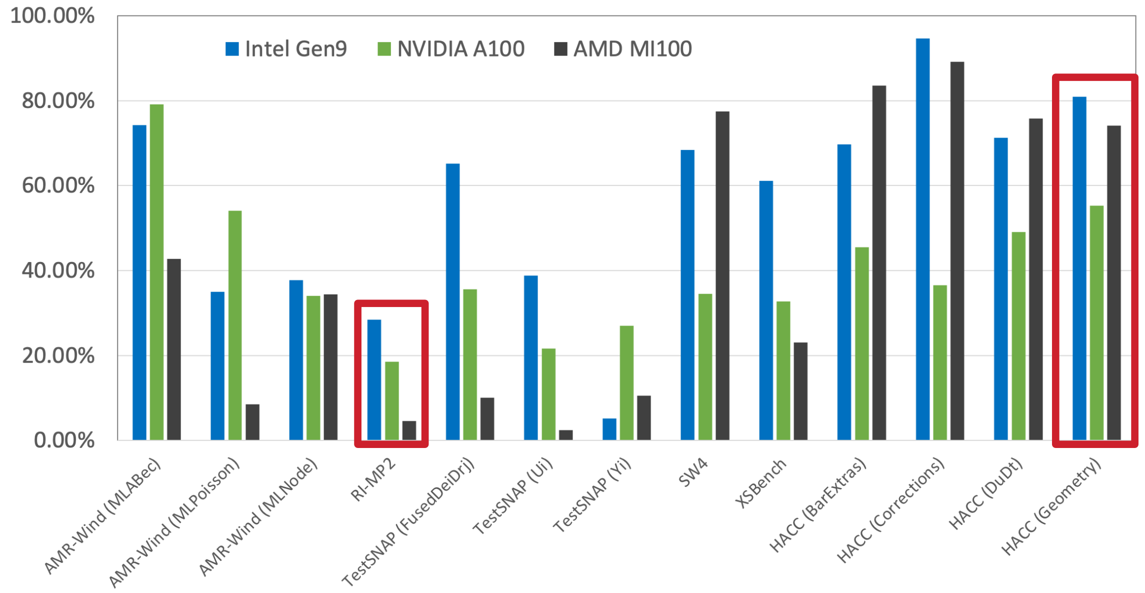


PERFORMANCE PORTABILITY

- Performance Portability Metric (PPM)
  - A harmonic mean of roofline efficiencies across AMD, Intel, and NVIDIA GPUs
  - A good metric to represent overall efficiency across the GPUs

- Observation
  - PPM helps us understand performance portability
    - HACC (Geometry)
      - PPM = 68.3% (higher is better)
    - RI-MP2
      - PPM = 9.64%

Application	Kernel	Intel Gen9	NVIDIA A100	AMD MI100	Perf.Port. Metric(%)
AMR-Wind	MLABec	74.2	79.1	42.8	60.6
	MLPoisson	34.9	54.1	8.47	18.2
	MLNode	37.7	34.1	34.4	35.3
HACC	BarExtras	69.7	45.5	83.5	62.1
	Corrections	94.6	36.5	89.1	60.9
	DuDt	71.3	49.1	75.8	63.0
	Geometry	80.9	55.3	74.1	68.3
SW4	curvilinear4sg	68.3	34.5	77.5	53.0
RI-MP2	RIMP2$omp	28.4	18.6	4.50	9.64
XSBench	XSBench$omp	61.2	32.7	23.1	33.2
TestSNAP	FusedDeiDrj	65.1	35.5	9.99	20.9
	Ui	38.8	21.7	2.36	6.06
	Yi	5.11	27.0	10.5	9.15



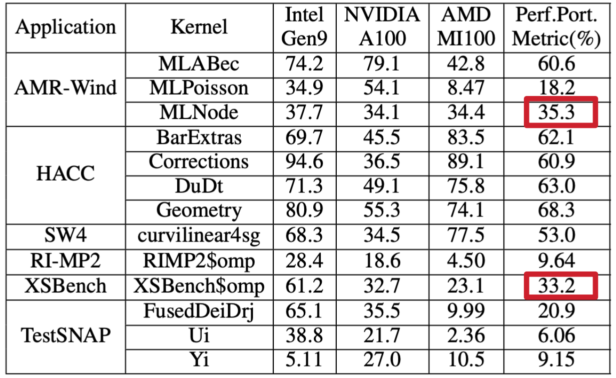
Argonne National Laboratory is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

CLUSTERS OF KERNELS BASED ON PPM

- A cluster with high PPM scores
  - Mostly performance portable kernels
- Middle PPM cluster
  - Somewhat performance portable kernels, and possibly benefit from further investigation
- A cluster with low PPM scores
  - Less performance portable based on roofline performance analysis
  - For this group, more investigation needed to identify critical bottlenecks of kernels
  - Need to consider some other factors not captured by roofline analysis
    - Memory latency, cache performance, atomic operation performance, instruction throughput, pipeline designs of processing units, NUMA effect, and so on

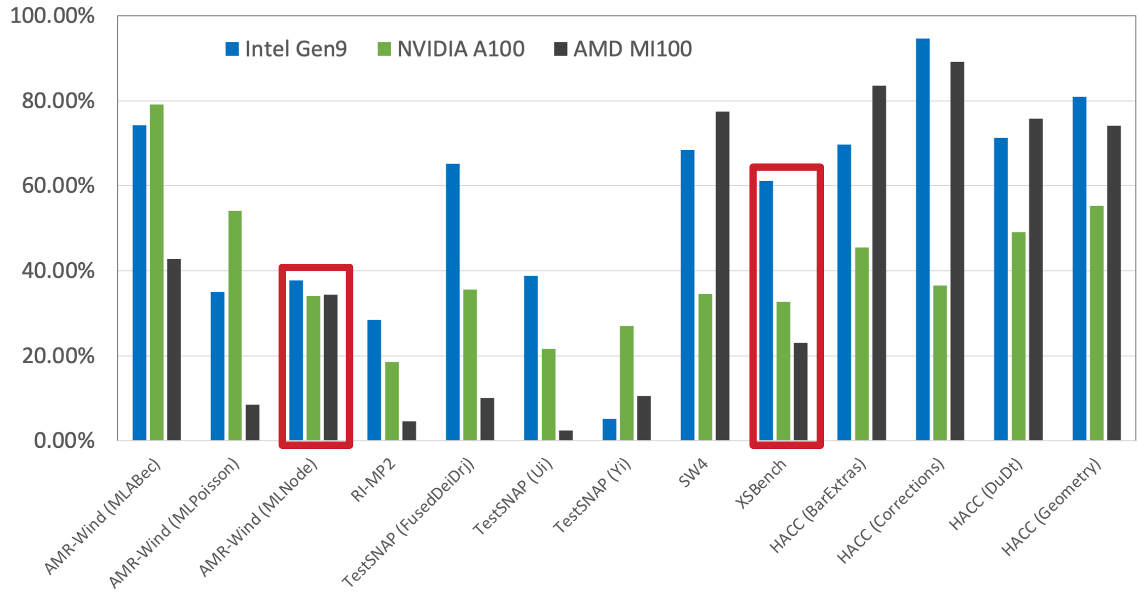
Cluster	Kernel	PPM(%)
High PPM	HACC Geometry	68.3
	HACC DuDt	63.0
	HACC BarExtras	62.1
	HACC Corrections	60.9
	AMR-Wind MLABec	60.6
Middle PPM	SW4 curvilinear4sg	53.0
	AMR-Wind MLNode	35.3
	XSBench	33.2
	TestSNAP FusedDeiDrj	20.9
	AMR-Wind MLPoisson	18.2
Low PPM	RI-MP2	9.64
	TestSNAP(Yi)	9.15
	TestSNAP(Ui)	6.06

PERFORMANCE VARIATION



Application	Kernel	Intel Gen9	NVIDIA A100	AMD MI100	Perf.Port. Metric(%)
AMR-Wind	MLABec	74.2	79.1	42.8	60.6
	MLPoisson	34.9	54.1	8.47	18.2
	MLNode	37.7	34.1	34.4	35.3
HACC	BarExtras	69.7	45.5	83.5	62.1
	Corrections	94.6	36.5	89.1	60.9
	DuDt	71.3	49.1	75.8	63.0
	Geometry	80.9	55.3	74.1	68.3
SW4	curvilinear4sg	68.3	34.5	77.5	53.0
RI-MP2	RIMP2$omp	28.4	18.6	4.50	9.64
XSBench	XSBench$omp	61.2	32.7	23.1	33.2
TestSNAP	FusedDeiDrj	65.1	35.5	9.99	20.9
	Ui	38.8	21.7	2.36	6.06
	Yi	5.11	27.0	10.5	9.15

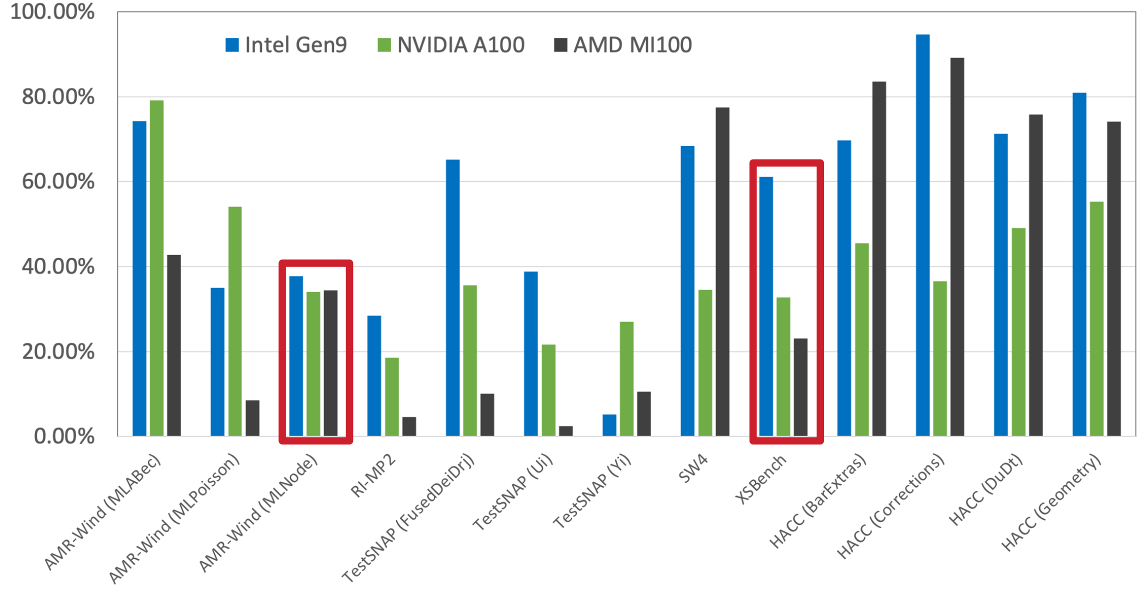
- Observation about variation
  - PPM vs. Consistency
    - AMR-Wind (MLNode)
      - PPM = 35.3%
    - XSBench
      - PPM = 37.7%
    - Can we say both kernels are similarly performance portable?
    - Additional metrics are helpful to understand consistency in addition to the performance portability metric



PERFORMANCE CONSISTENCY METRIC

- Additional metrics for performance consistency in this study
  - Std.Dev/Avg
  - Min/Max
- Observation about variation
  - PPM vs. Consistency
    - AMR-Wind (MLNode)
      - PPM = 35.3%
      - Std.Dev/Avg = 5.7% (lower is better)
      - Min/Max = 90.4% (higher is better)
    - XSBench
      - PPM = 37.7%
      - Std.Dev/Avg = 50.8%
      - Min/Max = 37.7%

Application	Kernel	Intel Gen9	NVIDIA A100	AMD MI100	Perf.Port. Metric(%)	Std.Dev. /Avg	Min /Max
AMR-Wind	MLABec	74.2	79.1	42.8	60.6	0.302	0.541
	MLPoisson	34.9	54.1	8.47	18.2	0.705	0.157
	MLNode	37.7	34.1	34.4	35.3	0.057	0.904
HACC	BarExtras	69.7	45.5	83.5	62.1	0.291	0.544
	Corrections	94.6	36.5	89.1	60.9	0.437	0.385
	DuDt	71.3	49.1	75.8	63.0	0.218	0.648
	Geometry	80.9	55.3	74.1	68.3	0.189	0.683
SW4	curvilinear4sg	68.3	34.5	77.5	53.0	0.377	0.445
RI-MP2	RIMP2$omp	28.4	18.6	4.50	9.64	0.700	0.158
XSBench	XSBench$omp	61.2	32.7	23.1	33.2	0.508	0.377
TestSNAP	FusedDeiDrj	65.1	35.5	9.99	20.9	0.748	0.153
	Ui	38.8	21.7	2.36	6.06	0.871	0.061
	Yi	5.11	27.0	10.5	9.15	0.804	0.189



PRODUCTIVITY

Portability layers increase productivity with some limitations

- The portability layers work as an aid to the productivity of the application developers since they reduce or eliminate the need for multiple code branches for different platforms.
- Several challenges
  - Architectural differences may results in multiple branches of codes for performance
    - Kokkos is portable across CPU and GPU, but TestSNAP has independent code branches for CPU and GPU for performance
    - The GPU branch increases the number of FLOPs with avoiding global memory read-writes ultimately ended up being a net benefit on GPU platforms.
  - Partial implementation of programming model specifications across different platforms
    - OpenMP target offloading, SYCL or other open standard specifications have partial implementations per platform at the moment.
    - Developers need to use common subset for their target platforms, till full specification are fully supported across platforms
  - Further performance optimization
    - It will be challenging to improve the performance on a specific platform w/o making an additional code branch
    - Need patience to use different performance tools interface

LESSON LEARNED

- Thanks to well developed portability layers (i.e., SYCL, OpenMP, RAJA, Kokkos, AMReX), all of the applications and mini-apps evaluated in this study were able to **portably run across AMD, Intel, and NVIDIA GPU platforms** with minimal to no changes in their code base.
- Getting performance data across multiple platforms is difficult, so this is a challenge to performing **performance portability analysis**.
- Estimating performance efficiencies is challenging, but using **roofline efficiency** can be a good approximation; however, kernels with low roofline efficiencies need further performance investigation.
- Additional metrics for **performance consistency** can be beneficial to understand performance variability across platforms.
- Observations on **productivity pitfalls** such as:
  - the need for code branching for CPU and GPU,
  - partially implemented specifications on some platforms.

ACKNOWLEDGEMENT

- This work was supported by
  - the **Argonne Leadership Computing Facility**, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357,
  - and by the **Exascale Computing Project** (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration).
- We also gratefully acknowledge the computing resources provided and operated by the **Joint Laboratory for System Evaluation** (JLSE) at Argonne National Laboratory.
- We would like to thank our collaborators:
  - **Programming Models and Architectures (PMA)** working group at Argonne National Laboratory: Abhishek Bagusetty, Kevin Harms, Ye Luo, Vitali Morozov, Kris Rowe, and Brice Videau
  - Sam Williams (LBNL), Zakhar Matveev(Intel) and their teams for roofline model discussion
  - Jon Rood (NREL), Paul Mullowney (NREL), and Weiqun Zhang (LBNL) for technical supports for AMR-Wind and AMReX
  - Rahul Gayatri (LBNL) and Daniel Arndt (ORNL) for discussing about TestSNAP
  - Christian Trott (SNL) and David Poliakoff (SNL) for discussing other performance bottlenecks on GPUs

THANKS!