ONEAPI DEVSUMMIT AT SC21

Argonne NATIONAL LABORATORY

GETTING READY TO AURORA EXA-SCALE SUPERCOMPUTER USING INTEL ADVISOR ROOFLINE ON INTEL CPUS AND GPUS

JAEHYUK KWACK
ALCF Perf. Engr. Group

ZAKHAR MATVEEV
Intel

U.S. DEPARTMENT OF ENERGY

11/14/2021

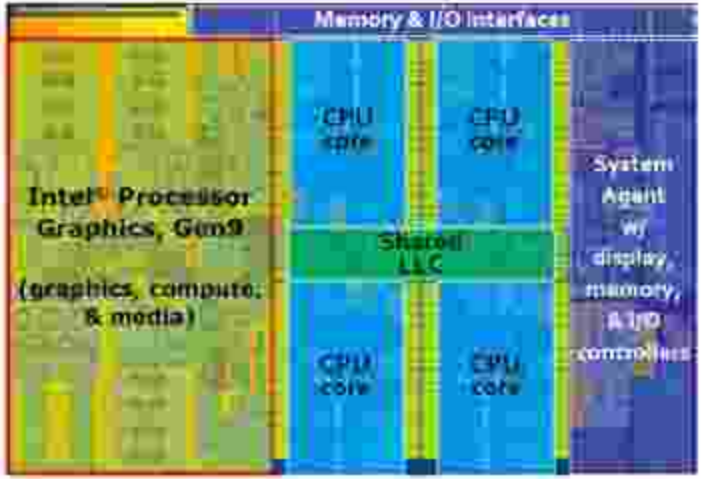
OVERVIEW OF AURORA TESTBED SYSTEMS

AURORA: A HIGH-LEVEL VIEW

- One of coming exa-scale systems supported by US DOE
  - 21 science/engineering project teams are actively working for exa-scale systems via ECP (Exascale Computing Project)
- Intel-HPE machine arriving at Argonne
  - Sustained Performance **> 1 ExaFlops**
- Intel Xeon processor and Intel Xᵉ GPUs
  - 2 Xeons (Sapphire Rapids)
  - 6 GPUs (Ponte Vecchio [PVC])
- Greater than 10 PB of total memory
- HPE/Cray XE platform and HPE Slingshot network
- Filesystem
  - Distributed Asynchronous Object Store (DAOS)
    - ≥ 230 PB of storage capacity
    - Bandwidth of > 25 TB/s
  - Lustre
    - 150 PB of storage capacity
    - Bandwidth of ~ 1TB/s

INTEL GEN9 GPU ON AURORA TESTBED

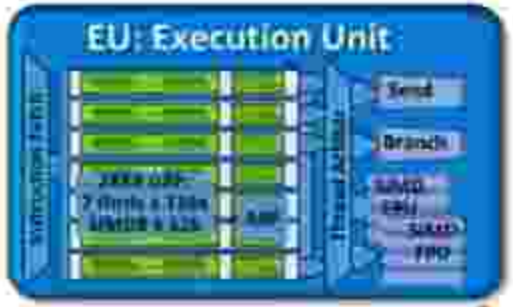
- Gen9: Double precision peak performance: 100-300 GF
  - Low by design due to power and space limits
  - Integrated GPU
- Hardware hierarchies
  - A GPU tile has multiple slices
  - A slice has multiple Sub-Slices
  - A sub-slice has multiple EUs



Slice

Sub-Slice

Execution Unit (EU)



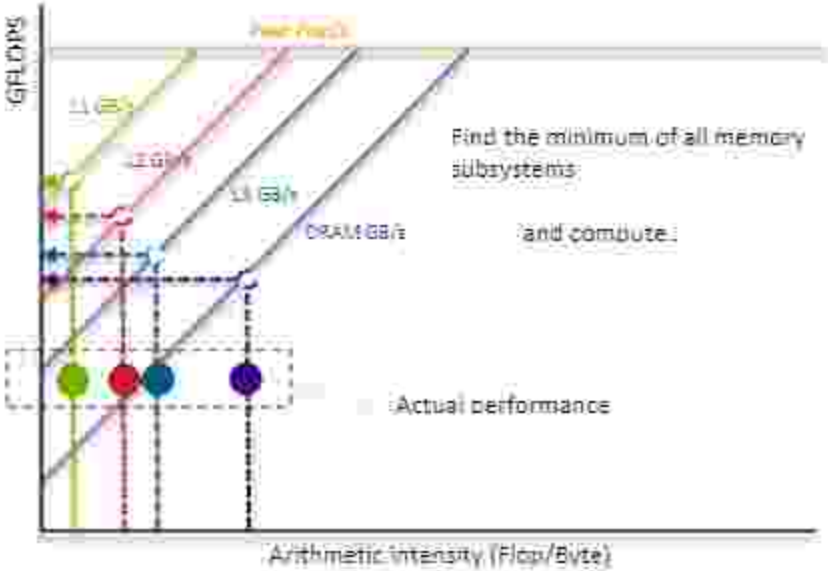
ADVISOR ON INTEL CPUS AND GPUS

AUTOMATED ROOFLINE ANALYSIS

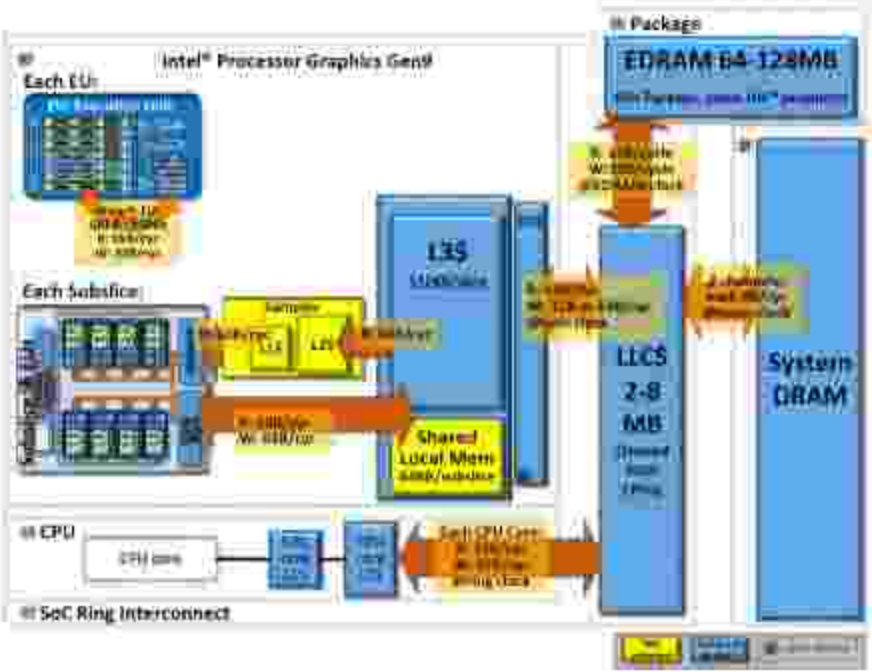
Via Intel Advisor

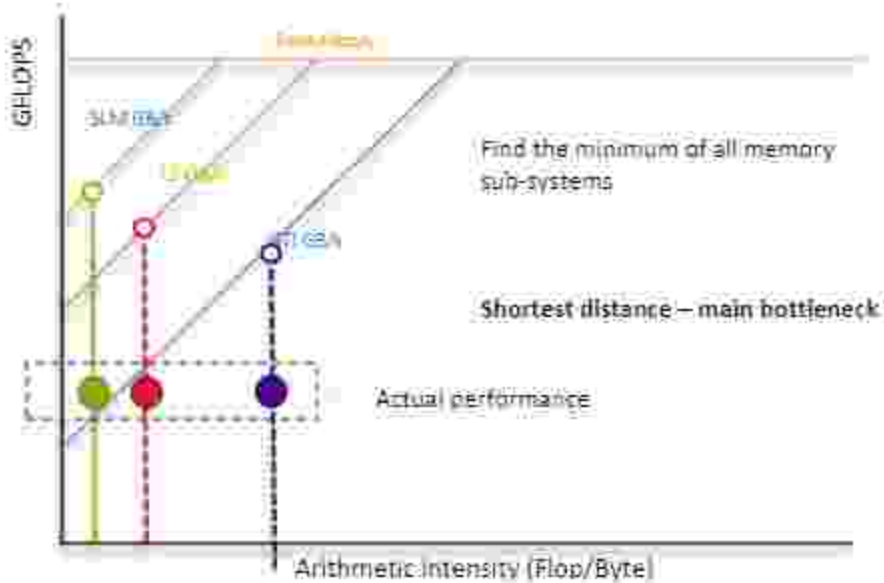
- Intel Advisor is a part of the Intel oneAPI Base Toolkit
- Performance headroom against hardware limitations
- Identify which optimizations will payoff the most
- Detect and prioritize bottlenecks
- Visualize optimization progress
- GPU roofline analysis
  - GPU recommendations
  - See the CPU and GPU performance side-by-side
  - GPU application performance characterization

- Memory Level Roofline (MLR) on CPU



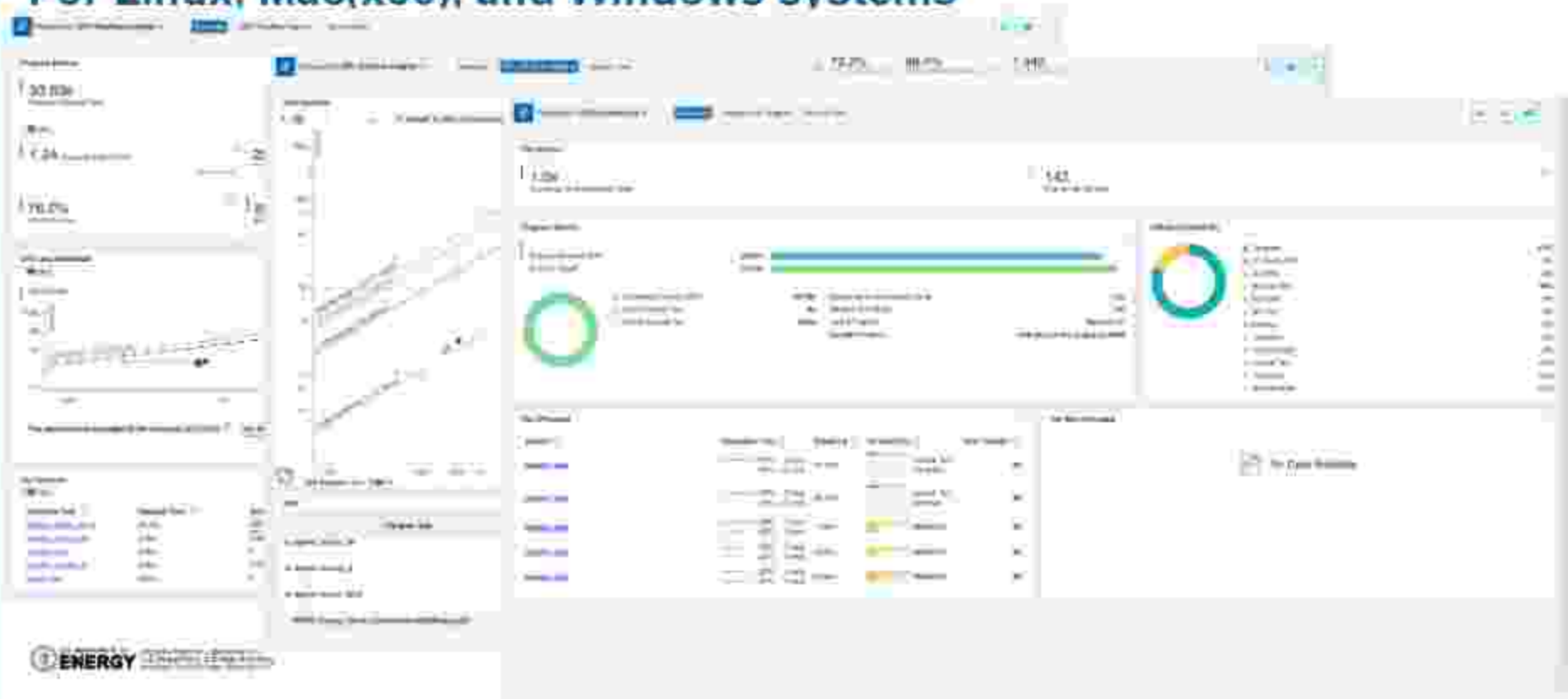
MEMORY LEVEL ROOFLINES (MLR) ON GPU





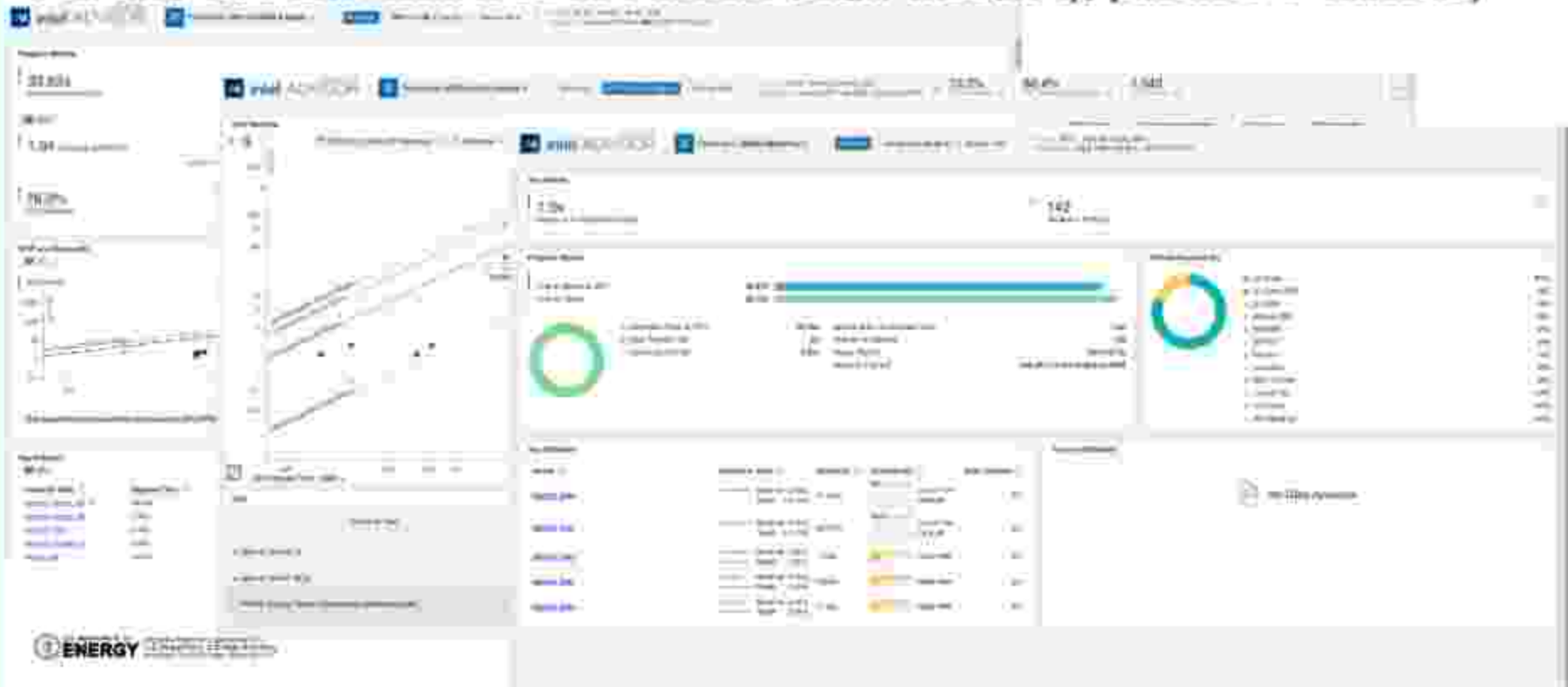
ADVISOR GUI INTERFACE

For Linux, Mac(x86), and Windows systems



“FAT” STAND-ALONE HTML REPORT

For any system with web browsers (e.g., Mac(M1), phones & tablets)



ARGONNE ADVISOR ROOFLINE USE-CASES

ARGONNE ADVISOR ROOFLINE USE-CASES

Presented at roofline performance analysis tutorials (@SC, ECPAM)

Application	ANL PoC	Programming models	Field of Science
AMR-Wind	JaeHyuk Kwack	SYCL/DPC++	Wind farm simulation (CFD/FSI)
GAMESS RI-MP2	Colleen Bertoni	OpenMP Target	Electronic structure method
NekBench	Kris Rowe	OCCA with OpenCL backend	Computational Fluid Dynamics
TestSNAP	Yasaman Ghadar	OpenMP Target	Material Science / Chemistry
LAMMPS kernel	Chris Knight	OpenMP Target, OpenCL, SYCL/DPC++	Material Science / Chemistry
XSBench/RSBench	John Tramm	OpenMP Target	Monte Carlo neutron transport
SW4Lite	Brian Homerding	RAJA with SYCL/DPC++ backend	Seismic modeling

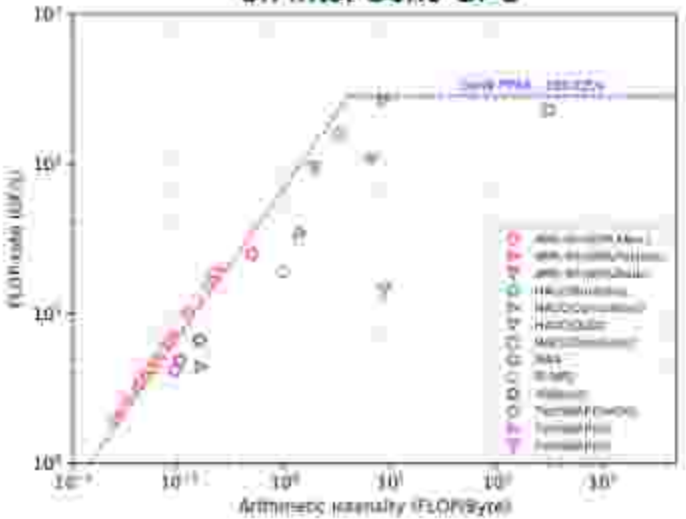
SC21 P3HPC WORKSHOP (2021)

Performance, Portability & Productivity in HPC @ SC21 on November 14th, 2021

- Title: Evaluation of Performance Portable Programming Models and Framework across AMD, Intel and NVIDIA GPUs using Applications and Mini-Apps
- Authors: J. Kwack, J. Tramm, C. Bertoni, Y. Ghadar, B. Homerding, E. Rangel, C. Knight, and S. Parker
- Employed ECP applications and mini-apps

Application	Kernel	Application	Kernel
AMR-Wind	MLABec	SW4	curvilinear4sg
	MLPoisson	GAMESS RI-MP2	RIMP2@omp
	MLNode	XSBench	XSBench$omp
HACC	BarExtras	TestSNAP	FusedDeiDrj
	Corrections		Uj
	DuDt		Yi
	Geometry		

EDRAM-based Advisor roofline results on Intel Gen9 GPU



PORTING
GAMESS RI-MP2 MINI-APP
FROM INTEL CPU TO INTEL GPU

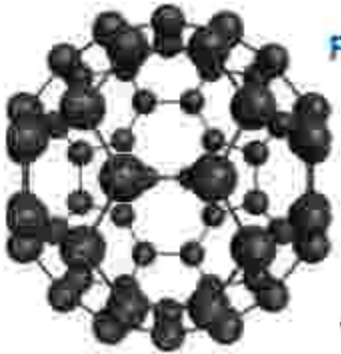
GAMESS AND RI-MP2 MINI-APP

- GAMESS is the General Atomic and Molecular Electronic Structure System
  - General-purpose electronic structure code (many methods and capabilities)
  - ~1 million lines of Fortran
  - Optional C/C++ GPU-accelerated libraries/applications in GAMESS
- Scientific problem of interest
  - FMO/RI-MP2 calculations towards accurate simulations of catalysis reactions inside a mesopourous silica nanoparticle
- RI-MP2 mini-app
  - Computes RI-MP2 perturbative correction to the Hartree-Fock energy
  - RI-MP2 mini-app examples

Coronene

H_2O clusters

Fullerene



INTRO OF RI-MP2 MINI-APP

An ECP Proxy Application

- Computes RI-MP2 perturbative correction to the Hartree-Fock energy

$$E^{(2)} = \sum_{i \le j}^{occ} (2 - \delta_{ij}) \sum_{ab}^{vir} \frac{(ia|jb)[2(ja|jb) - (ib|ja)]}{\epsilon_i + \epsilon_j - (\epsilon_a + \epsilon_b)}$$

- Simplifies expression with the RI approximation letting it be written in terms of matrix multiplication

$$(ia|jb) = \sum_{n}^{naux} B^i_{an} B^j_{bn}$$

- Combine i,j, loops over occ orbitals, and then reduce over a,b, virtual orbitals to compute the final correlation energy

```
for(int iACT=0;iACT<nACT;iACT++){
  for(int iACT=0; iACT<=iACT;iACT++){

    // A kernel to compute QVV via Matrix Multiplication
    QVV[0:NVIR][0:NVIR] = B32[iACT][0:NVIR][0:NAUXBASD] *
                          B32[jACT][0:MAXXBASD][0:NVIR];

    // Accumulating E2 using QVV[:][:], ei[i][i], and eab[:][:]
    for(int iB=0; iB<NVIR; iB++) {
      for(int iA=0; iA<NVIR; iA++) {
        Tijab = QVV[iB*NVIR][iA] /
                ( ei[iACT][iACT] + eab[iB][iA]);
        Qt = QVV[iB][iA] + QVV[iB][iA];
        E2_t += Tijab * (Qt - QVV[iA][iB]);
      }
    }
    fac = (iACT==jACT) ? (1.0E0) : (2.0E0);
    E2 -= E2_t *fac;
  }}
```

RI-MP2 CODE VERSIONS

- We constructed four RI-MP2 variants to explore with w25.rand.
- These variants add progressive/incremental levels of optimization as follow:
  - **V0-CPU**: an initial version,
  - **V1-CPU**: QVV computations use MKL DGEMM,
  - **V3-GPU**: offloaded V1-CPU to GPU using OpenMP Target offloading model with optimal values for num_teams, and threads_limit
  - **V5-GPU:** QVV computation uses less MKL DGEMM calls by restructuring an outer loop; For E2, IACT loop is distributed subslices w/ "target teams distribute", and then IB loop uses EUs w/ "parallel for" at each subslice, with optimal values for num_teams, and threads_limit

V0-CPU

- QVV: Computing QVV
- E2:E2 accumulation

```
double *QVV;
double E2_local=0.0E0;
QVV = new double[NVIR*NVIR];

for(int IBeta=0;IBeta<NBeta;IBeta++)
for(int IAlfa=0;IAlfa<=IBeta;IAlfa++){
```

QVV

```
  // Compute QVV
  #pragma omp parallel for collapse(2)
  for (int a = 0; a < NVIR; ++a) {
  for (int b = 0; b < NVIR; ++b) {
  for (int i = 0; i < NOCCBASE; ++i) {
    QVV[a,b] += Bbi[IAlfa,a,i]*Bbi[IBeta,b,i];
  }}}
```

E2

```
  // Accumulate E2
  double E2_tmp=0.0;
  for(int IRow=0; IRow<NVIR; IRow++)
  for(int ICol=0; ICol<NVIR; ICol++)
    double Eiajb = QVV[IRow,ICol] / ( eAi[IAlfa,IRow] - eAi[IBeta,ICol] );
    double Q_t = 2*QVV[IRow,ICol];
    E2_t += Eiajb * ( Q_t - QVV[ICol,IRow] );
  }   // Loop for IRow and ICol
  double FAC = (IAlfa==IBeta) ? 1.0E0 : 2.0E0;
  E2_local += FAC*E2_t;
}   // Loop for IBeta and IAlfa

*E2 = *E2 + E2_local;
delete[] QVV;
```

- QVV L2 cache bound
  - 31.33 GF/s
  - L2 AI= 0.5

Data traffic (Bytes) ratios:
L2 L3 DRAM = 4.2 2.2 1

- E2 L2 cache bound
  - 5.13 GF/s
  - L2 AI= 0.091

Data traffic (Bytes) ratios:
L2 L3 DRAM = 4.3 1.4 1

V1-CPU

- QVV: Computing QVV using MKL DGEMM
- E2:E2 accumulation

```
double *QVV;
double E2_local=0.0E0;
QVV = new double[NVIR*NVIR];

for(int iSCT=0;iSCT<nSCT;iSCT++){
for(int iACT=0;iACT<nACT;iACT++){

    // Compute QVV
int m=NVIR;
int n=NVIR;
double one = 1.0;
double zero = 0.0;
dgemm('T','N',&m,&n,&k,&one,XCCS[iSCT][iACT].O,&k,&XCCS[iSCT][iACT].O,&k,&zero
,QVV,&m);

    // Accumulate E2
    double E2_sc=0.0;
    for(int ISC=0; ISC<NVIR; ISC++){
    for(int IAC=0; IAC<NVIR; IAC++){
       double tijab = QVV[ISC,IAC] / ( eij[iACT][iACT] - eab[ISC,IAC] );
       double O_sc = 2*QVV[ISC,IAC];
       E2_sc += tijab *  O_sc - QVV[IAC,ISC] ;
    }   // Loop for IAC and ISC
    double E2C = (iACT==iACT ? 0.5E0 :  1.0E0);
    E2_local += E2C*E2_sc;
}   // Loop for iACT and iACT

*E2 = *E2 + E2_local;
delete[] QVV;
```

QVV

E2

- QVV DP FMA bound
  - 61.96 GF/s
  - L2 AI= 1.34

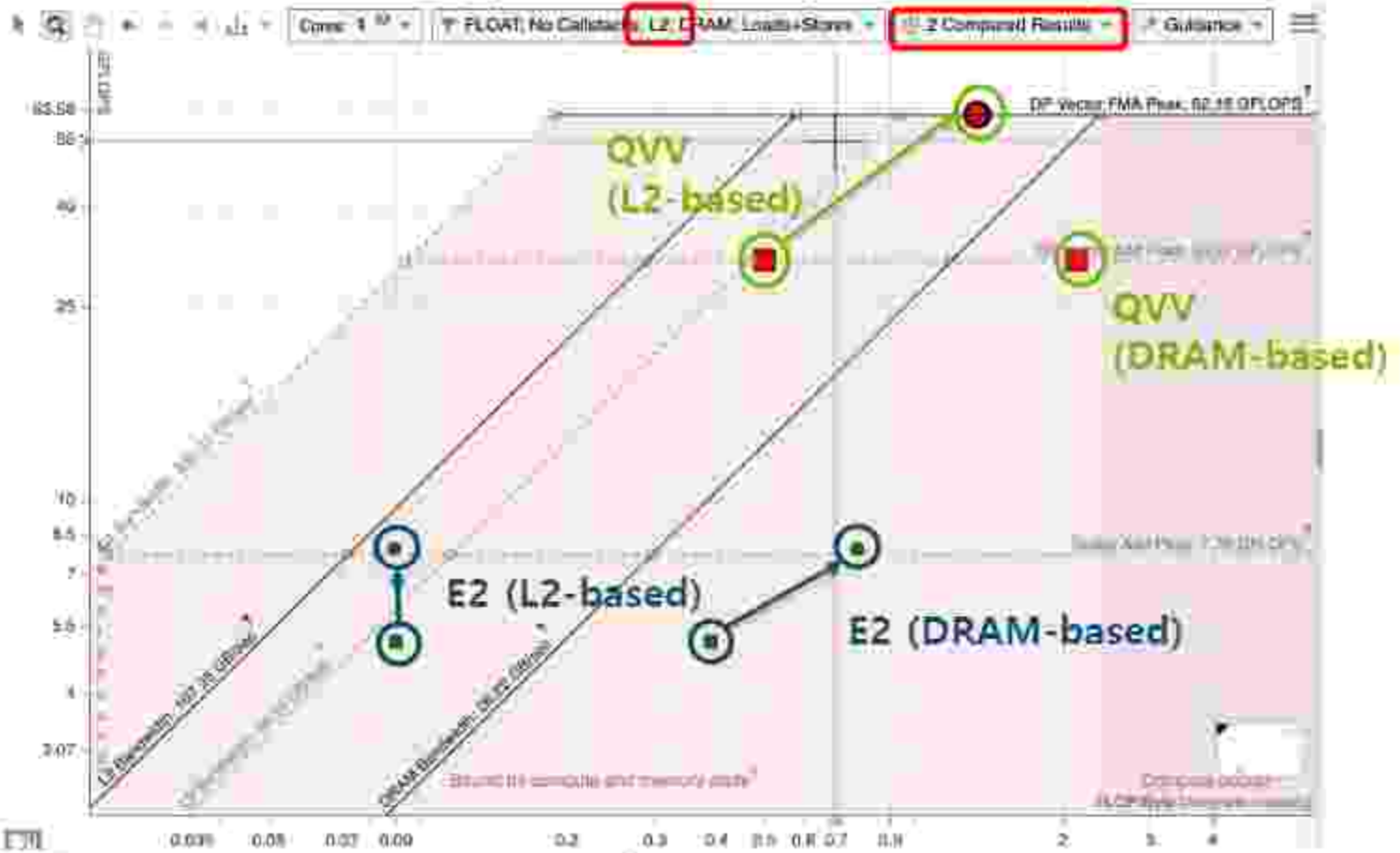
Data traffic (Bytes) ratios:
L1 L2 L3 = 4.7 2.0 1

- E2 L2 cache bound
  - 7.95 GF/s
  - L2 AI= 0.090

Data traffic (Bytes) ratios:
L2 L3 DRAM = 8.6 2.9 1

V0-CPU to V1-CPU

- QVV computation is compute-bound (by MKL) w/ much lower memory pressure; it consequently increases performance of E2.



V3-GPU: initial CPU-like version

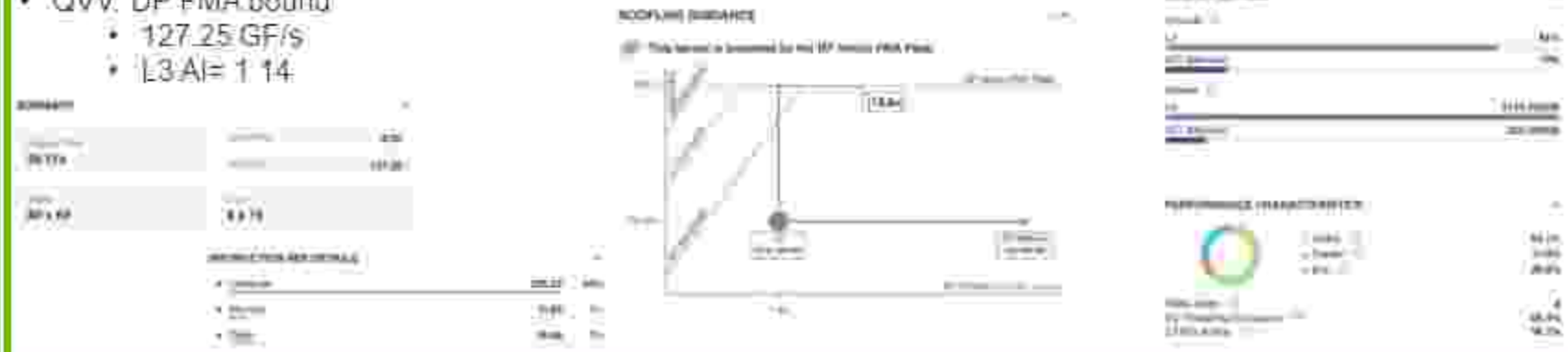
```
double *QVV;
double E2_local=0.0E0;
int dummy=0,
QVV = new double[NVIR*NVIR];
double *B3I2, *B3I2;

#pragma omp target enter data map(alloc:QVV[0:NVIR*NVIR])
device(dnum)
#pragma omp target enter data
map(to:e1[0:NACT*NACT],e2[0:NVIR*NVIR],B3I2[0:B3Isize])
device(dnum)
for(int JACT=0;JACT<NACT;JACT++)
for(int IACT=0;IACT<=IACT;IACT++){

    // Compute QVV
    int nvNVIR;
    int kvNVIRxNVIR;
    double one = 1.0;
    double zero = 0.0;
    B3I1 = &B3I1[IACT,0,0];
    B3I1 = &B3I1[JACT,0,0];
    #pragma omp target variant dispatch
use_device_ptr(B3I1, B3I2,B3I2,QVV) device(dnum)
    dgemm('T','N', &n,&m, &k,&one,B3I1,&k,B3I2,&k,&zero,QVV,&n);

    // Accumulate E2
    double E2_tmp=0.0;
    #pragma omp target teams distribute parallel for
reduction(+:E2_tmp) map(tofrom:E2) shared_list[72] device(dnum)
    for
```

```
        for(int IA=0; IA<NVIR; IA++)
        for(int IB=0; IB<NVIR; IB++)
            double Tijab = QVV[IB,IA] / (e1[IACT,IACT]+e2[IE,IA]);
            double Q_ij = 2*QVV[IB,IA];
            E2_t += Tijab * (Q_ij - QVV[IA,IB]);
        }; // loop for IA and IB
    } // end target teams distribute parallel for
    double fac = (IACT==JACT) ? 1.0E0 : 2.0E0;
    E2_local += fac*E2_t;
}  // loop for IACT and JACT

*E2 = *E2 + E2_local;
delete[] QVV;

#pragma omp target exit data map(release:QVV[0:NVIR*NVIR])
device(dnum)
#pragma omp target exit data
map(release:e1[0:NACT*NACT],e2[0:NVIR*NVIR],B3I2[0:B3Isize])
device(dnum)
```

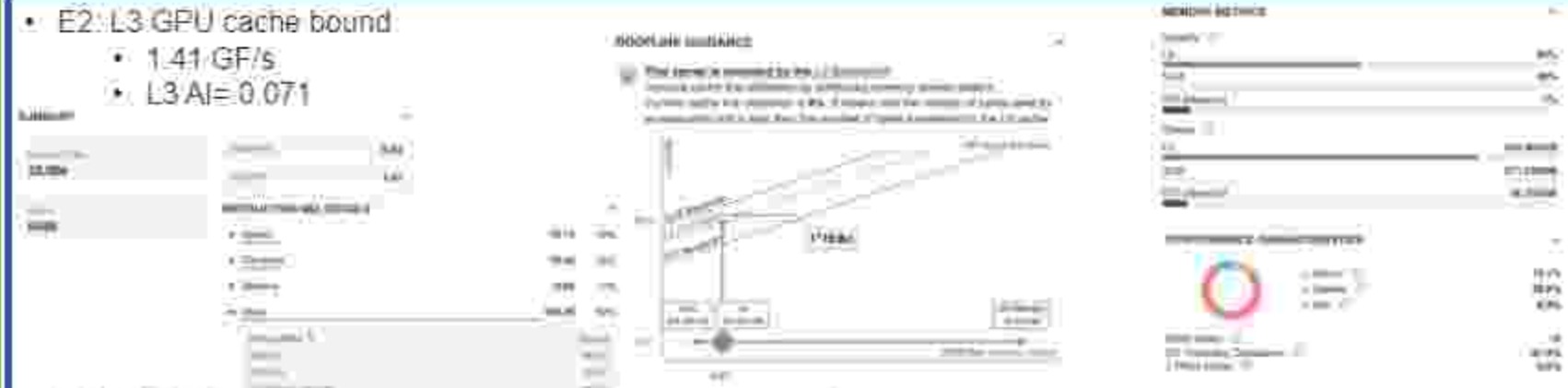
- Computing QVV using MKL DGEMM w/ GPU offloading
- E2 accumulation w/ GPU offloading

V3-GPU: initial CPU-like version

- QVV: DP FMA bound
  - 127.25 GF/s
  - L3 AI= 1.14



- E2: L3 GPU cache bound
  - 1.41 GF/s
  - L3 AI= 0.071



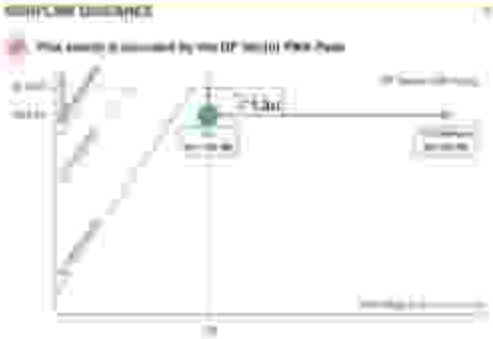
V5-GPU: restructured loops for better GPU performance

```
double *QVV;
double E2_local=0.0E0;
int dnum=0;
QVV = new double[NVIR*NACT*NVIR];
double *B32;

#pragma omp target enter data map(alloc:QVV[0:NVIR*NACT*NVIR])
device(dnum)
#pragma omp target enter data
map(to:eij[0:NACT*NACT],eab[0:NVIR*NVIR],B32[0:B32alse])
device(dnum)
for(int IACT=0;IACT<NACT;IACT++){

    // Compute QVV
    int sz=NVIR*(IACT+1);
    int m=NVIR;
    int k=NACT*NACT;
    double one = 1.0;
    double zero = 0.0;
    B32 = &B32[IACT,0,0];
    #pragma omp target variant dispatch use_device_ptr(B32,B32,QVV)
device(dnum)
    dgemm('T','N',sz,m,k,one,B32,k,B32,k,zero,QVV,m);

    // Accumulate E2
    #pragma omp target teams distribute reduction(+ E2_local)
num_teams(80) thread_limit(1) device(dnum)
    for(int IACT=0 ; IACT<=IACT; IACT++){
       double E2_c=0.0;
       #pragma omp parallel for reduction(+:E2_c)
       for(int IBA=0; IBA<NVIR; IBA++)
       for(int IBB=0; IBB<NVIR; IBB++)
          double Tijab = QVV[IB,IACT,IA] / (eij[IACT,IACT] +
eab[IB,IA] );
          double Q_c = 2*QVV[IB,IACT,IA];
          E2_c += Tijab * (Q_c - QVV[IA,IACT,IB]);
       }  // loop for IA and IB
       double FAC = (IACT==IACT) ? 1.0E0 : 2.0E0;
       E2_local += FAC*E2_c;
    }  // loop for IACT
}  // loop for JACT

*E2 = *E2 + E2_local;
delete[] QVV;

#pragma omp target exit data map(release:QVV[0:NVIR*NACT*NVIR])
device(dnum)
#pragma omp target exit data
map(release:eij[0:NACT*NACT],eab[0:NVIR*NVIR],B32[0:B32alse])
device(dnum)
```

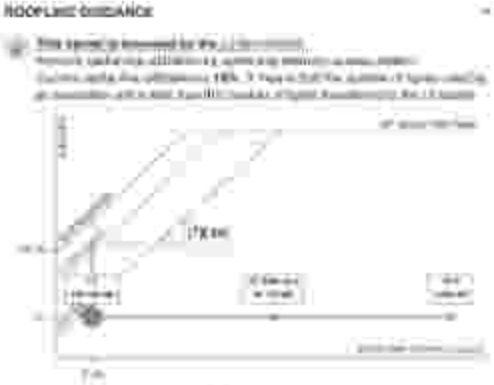
- Computing QVV with less MKL DGEMM offloading calls on GPU by restructuring an outer loop
- E2 accumulation w/ hierarchial GPU offloading (IACT-loop to Subslices (SS), and IB-loop to Execution Units (EUs)

V5-GPU: restructured loops for better GPU performance

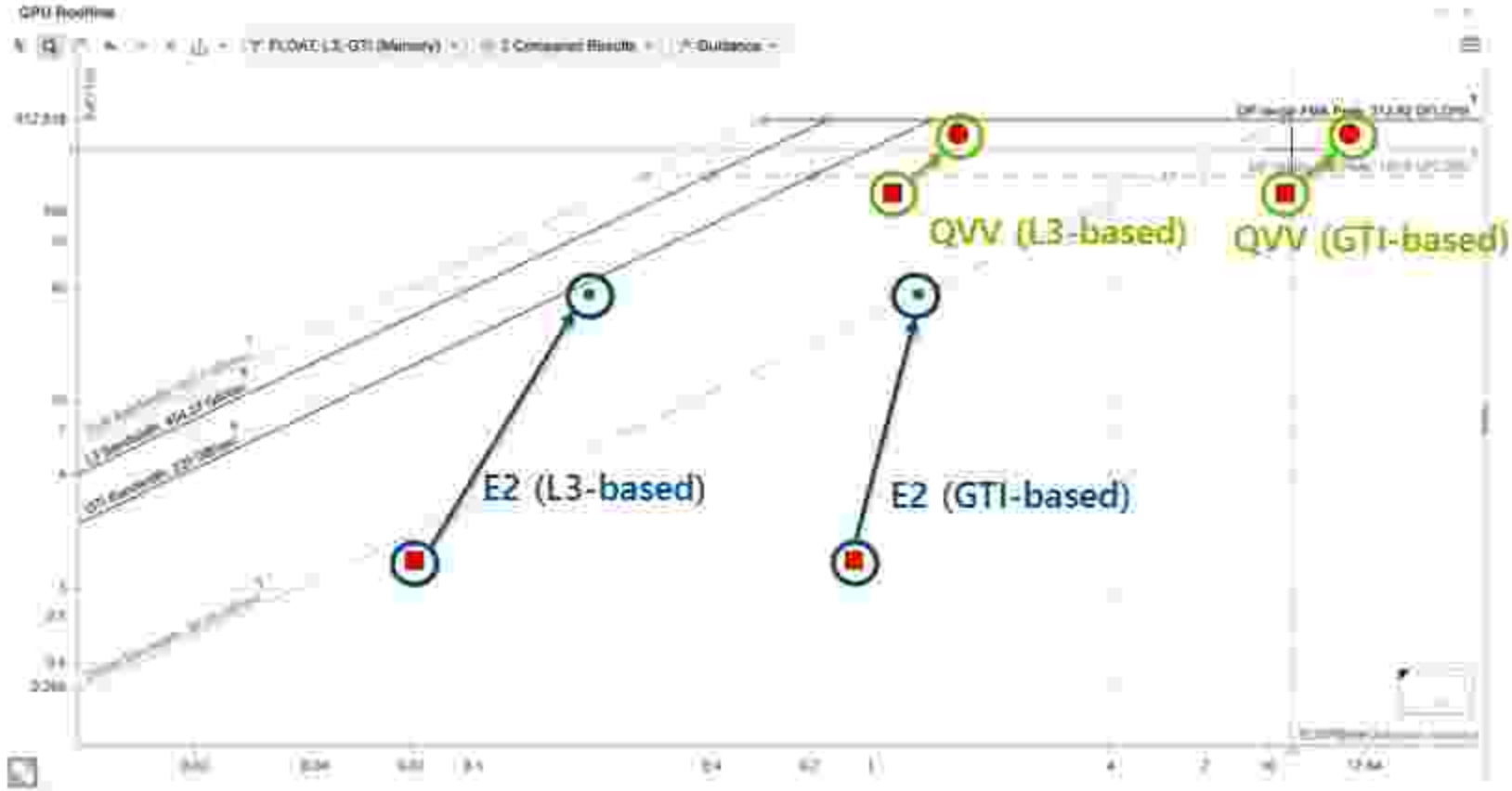
- QVV: DP FMA bound
  - 262.84 GF/s
  - L3 AI= 1.66



- E2: L3 GPU cache bound
  - 37.0 GF/s
  - L3 AI= .019



Comparison from V3-GPU to V5-GPU



SUMMARY & CONCLUDING REMARKS

PERFORMANCE RESULTS

H_2O clusters

Tested Input: w25.rand (random data with structure of 25 H_2O clusters)
Employed Compute Node: Intel(R) Xeon(R) CPU E3-1585 v5

- CPU: Intel Xeon Skylake @ 3.5GHz (4 cores)
- GPU: Intel Gen9 GT4e @ 1.15GHz (72 EUs)

Selected versions	QVV		E2		Overall	
	Target	Time(s)	Target	Time(s)	Time(s)	Speedup over V00
V0-CPU	CPU	156.68	CPU	1.34	158.02	1.0x
V1-CPU	CPU	86.41	CPU	0.92	87.33	1.8x
V3-GPU	GPU	32.38	GPU	23.06	55.44	2.9x
V5-GPU	GPU	21.49	GPU	0.86	22.35	7.1x

CONCLUDING REMARKS

- Argonne application developers and other US DOE collaborators have been actively porting their applications to Aurora testbed systems with Intel oneAPI toolkits for the coming Aurora Exa-scale supercomputer at Argonne National Laboratory.
- As an example, we present four versions of GAMESS RI-MP2 mini-app on Intel CPU and GPUs.
- Intel Advisor successfully provides the detailed performance data via roofline analysis features on both Intel CPUs and GPUs.
- Call-to-actions
  - If you are working on ECP projects and interested in Aurora testbeds with Intel oneAPI toolkit, you can get the access via https://www.jlse.anl.gov/accessing-jlse-resources/.
  - If you are interested in Intel GPUs with Intel oneAPI toolkits for your own applications, you may try the Intel DevCloud system via the following link
    - https://www.intel.com/content/www/us/en/developer/tools/devcloud/overview.html
  - Enjoy!

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 **Colleen Bertoni** at Argonne National Laboratory for discussing GAMESS RI-MP2 miniapp, **Kirill Rogozhin, Yulia Efimenko,** and **Ekaterina Guseva** for **Advisor** and **Louise Huot** for **MKL** for providing technical supports for issues, reviewing data and providing feedback for this study.

THANKS!