IXPUG 2025, TACC

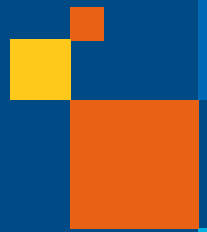
OpenMP in oneAPI:
Empowering Scientific Computing from Laptop to Aurora Exascale Supercomputer

April/15/2025

Jeongnim Kim, DCAI, Intel Corp.

Patrick Steinbrecher and Xinmin Tian, DCAI, Intel Corp.

Ye Luo, Argonne Leadership Computing Facility, Argonne National Laboratory



intel®

DSE

Legal Notices and Disclaimers

Performance varies by use, configuration and other factors. Learn more on the Performance Index site.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

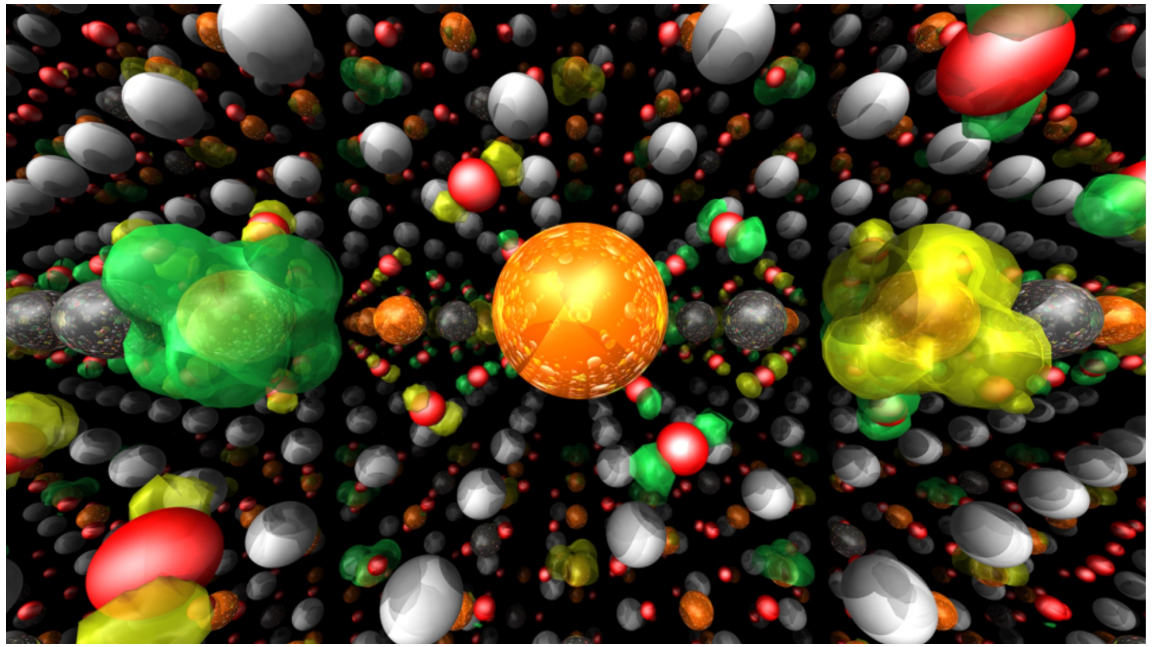
Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Agenda

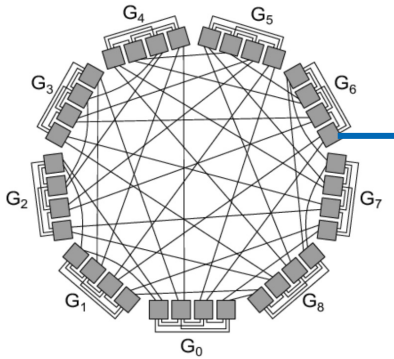
- OpenMP on hierarchical heterogeneous platforms
- OpenMP highlights
  - Versions 5.x and 6
  - Supports in oneAPI
- Enhanced OpenMP composability in oneAPI
- OpenMP applications
- Closing



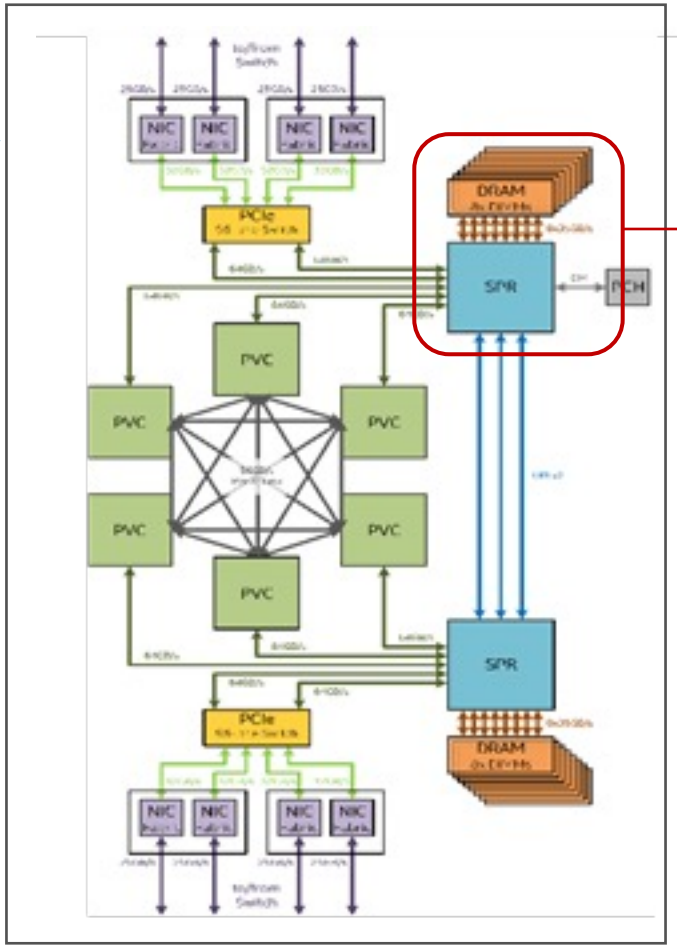
Researchers are using the QMCPACK code to accurately predict complex materials properties, such as the electron spin density of cobalt-doped silver nickel oxide delafossite depicted in this image. Image by ALCF Visualization and Data Analytics Team. https://www.alcf.anl.gov/news/argonne-s-aurora-supercomputer-set-supercharge-materials-discovery

Making Most of Hierarchical Heterogenous Platforms

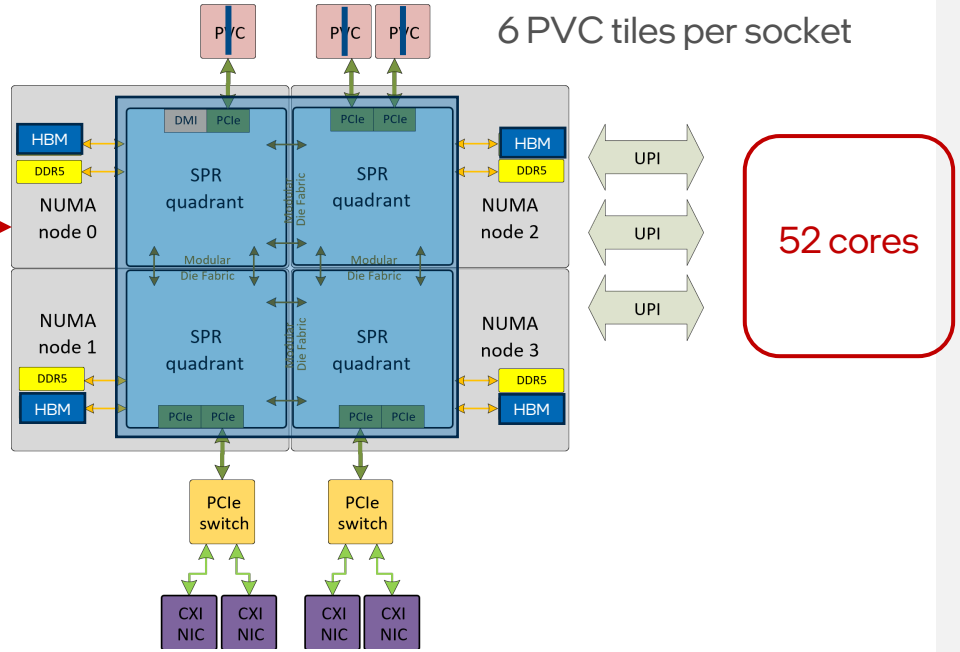
Dragonfly network*



Aurora Node



SPR Socket



Aurora Exascale Supercomputer
https://www.alcf.anl.gov/aurora

Can my problem fit to a "fast" memory domain?

- Socket (512 GB) + 6PVC tiles (384 GB)
- 8 cores (85 GB) + PVC tile (64 GB)
- 1 core (10 GB) + 1/8 PVC tile (8 GB)

OpenMP for *shared-memory* parallel programming

* Fig 2(a), DOI 210.1109/CCGRID.2017.93

OpenMP, multi-platform shared-memory programming

https://www.openmp.org

Uses a *portable*, *scalable* model that gives programmers a simple and flexible interface for developing *parallel applications* for platforms ranging from the standard *desktop* computer to the *supercomputer*.

https://en.wikipedia.org/wiki/OpenMP

Key features include

- Directives-Based Parallelism
- Shared Memory Model
- Task and Loop Parallelism
- Device Offloading
- Portability

```
void axpy(const float *a, const float *b, float *c, int n)
{
#pragma omp target teams distribute parallel for \
        map(to:a[0:n],b[0:n]) map(from:c[0:n])
  for (int id = 0; id < n; id++)
  {
    c[id] = a[id] + b[id];
  }
}
```

```
float a[64]={1}, b[64]={2}, c[64];
axpy(a,b,c,64);
```

caller

OpenMP Specifications Highlights

Version 5, a Major Leap Forward

- Memory management enhancements
  - *declare mapper* for custom data mapping
  - *allocate* to control memory allocations
- Tasking improvements
  - *taskloop* simplify parallel loop execution
  - *depend* clause enhancement
- Device offloading
  - Improved *target* for heterogenous computing
  - *require* to specify device capabilities
- Loop transformations
  - *loop* for flexible loop parallelization
  - *order(concurrent)* for safe execution
- Error handling
- Tools and debugging

Version 6.0 for Easier Programming

- Enhanced device features
  - *workdistribute* directive divides the execution of array notation
  - improved support for unified memory models and asynchronous data transfers.
  - target enter/exit data for finer control over data movement.
- Improved loop transformations
  - Use of loop fusion, reversal and interchange.
- Greater user control of storage resources and memory spaces
- Support for parallelization of the latest C, C++ & Fortran language standards
- Interoperability with accelerator programming models, e.g. SYCL/DPC++

What's new in 2024 Intel® Compiler Releases?

18 out of 65 OpenMP 6.0 new features are supported in Intel compilers released in 2024 for enhancing Intel® Xeon™ CPUs and Xe GPUs support.

OpenMP 6.0 feature support, for example

- *Loop interchange* support, *OMP_PLACES* extensions, new runtime APIs, *interop clause* for dispatch construct, *device_type*(host | nohost | any) clause for *target* construct, new environment variable such as *omp_thread_limit* , *append_args* takes *prefer_type, groupprivate, declare target local,* … etc.

Support C++23, Fortran 2008, Fortran 2018, Fortran 2023 with OpenMP Offloading

OpenMP CPU performance tuning for AI models (llama, resnet50, Bert, GPT, mobilenet, etc.) on Xeon-based AI PCs

New support for in-order asynchronous offloading

Improved Unified Shared Memory (USM) Support

Enhanced OpenMP and SYCL/DPC++ Composability

Optimization report improvements and thread/memory/address sanitizer support.

Performance Optimizations (auto-vectorization for GPUs, do-concurrent auto-offloading, NG-range for SIMT and explicit SIMD execution, loop optimizations, … etc.)

Enhanced OpenMP Composability in oneAPI

Enhanced OpenMP Composability in oneAPI

- MPI+OpenMP threading/task
  - SMP parallel programming, optimized for a node with multiple numa domains

⊗OpenMP offload to Intel GPUs

⊗MKL & performance libraries

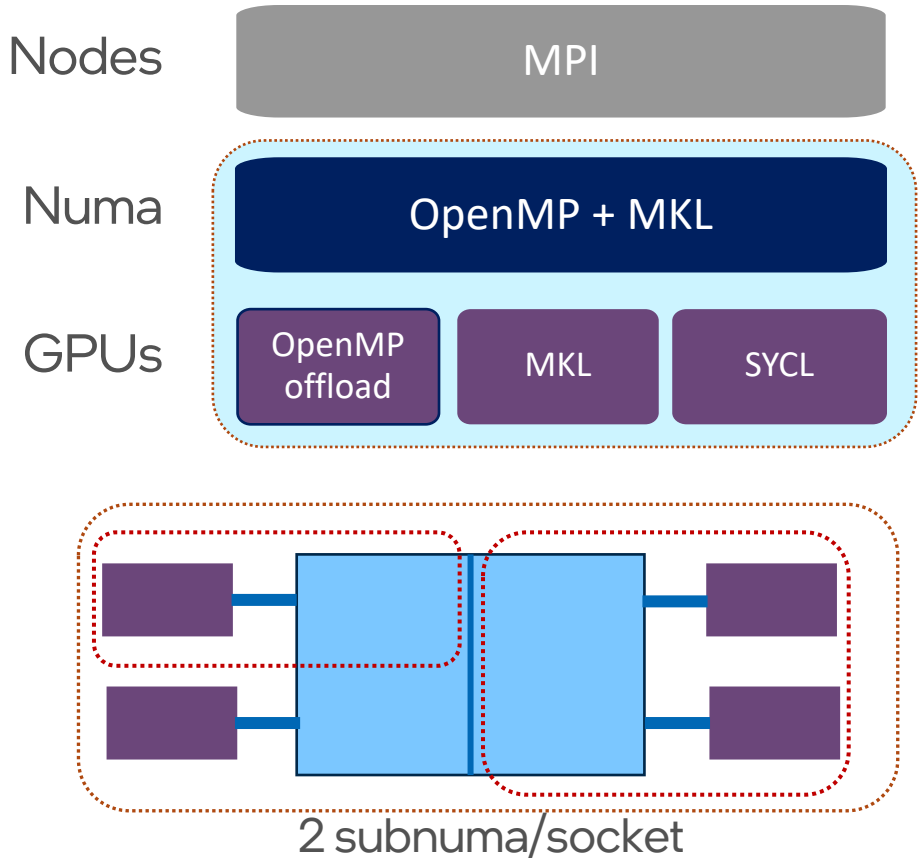
⊗SYCL user kernels

⊗oneCCL, oneSHMEM

⊗python

- Frameworks

Expanded Distributed/Shared Parallel Programming Model



Enhanced OpenMP Composability in oneAPI

- MPI+OpenMP threading/task
  - SMP parallel programming, optimized for a node with multiple numa domains

⊗OpenMP offload to Intel GPUs

⊗MKL, performance libraries

⊗SYCL user kernels

⊗oneCCL, oneSHMEM

⊗python

- Frameworks

```
! Calling dgemm on the CPU
call dgemm(transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c_ref, ldc)

! Calling dgemm on the GPU
!$omp target data map(a,b,c)
!$omp dispatch
call dgemm(transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c, ldc)
!$omp end target data
```

```
a = (double *)omp_target_alloc_shared(sizea * sizeof(double), dnum);
b = (double *)omp_target_alloc_shared(sizeb * sizeof(double), dnum);
c = (double *)omp_target_alloc_shared(sizec * sizeof(double), dnum);

// run gemm on host, use standard oneMKL interface
dgemm("N", "N", &m, &n, &k, &alpha, a, &lda, b, &ldb, &beta, c_ref, &ldc);

#pragma omp dispatch device(dnum)
dgemm("N", "N", &m, &n, &k, &alpha, a, &lda, b, &ldb, &beta, c, &ldc);
```

Enhanced OpenMP Composability in oneAPI

- MPI+OpenMP threading/task
  - SMP parallel programming, optimized for a node with multiple numa domains

⊗OpenMP offload to Intel GPUs

⊗MKL, performance libraries

⊗SYCL user kernels

⊗oneCCL, oneSHMEM

⊗python

- Frameworks

Initialize SYCL queue by OpenMP interop

```
int dnum = omp_get_default_device();

omp_interop_t interop;
#pragma omp interop device(dnum) init(prefer_type("sycl"),targetsync: interop)

int result;
sycl::queue *Q = static_cast<sycl::queue *>(
      omp_get_interop_ptr(interop, omp_ipr_targetsync, &result));
```

Mix OpenMP, SYCL and MKL

```
T *A_device = sycl::malloc_device<T>(m*k, *Q);
T *B_device = sycl::malloc_device<T>(k*n, *Q);
T *C_device = sycl::malloc_device<T>(m*n, *Q);

auto e1 = Q->copy(B_host, B_device, k * n);
auto e2 = Q->parallel_for( sycl::range<1>(n*m),
    [=](sycl::id<1> i) { C_device[i] = T{}; });

#pragma omp target teams distribute parallel for collapse(2)
for(int i = 0; i < n; ++i)
  for(int j = 0; j < m; ++j)
    A_device[i * m + j] = compute(i, j);

oneapi::mkl::blas::column_major::gemm(*Q,                    // MKL SYCL API
    transA, transB, m, n, k, alpha, A_device, ldA,
    B_device, ldB, beta, C_device, ldC, {e1,e2}).wait();
```

OpenMP Interop: Foreign Runtime Environment Values

https://www.openmp.org/wp-content/uploads/OpenMP-API-Additional-Definitions-2-1.pdf

foreign-runtime-ids		datatypes			
id	name	targetsync	device_context	device	platform
1	cuda	cudaStream_t	N/A	int	N/A
2	cuda_driver	CUstream	CUcontext	CUdevice	N/A
3	opencl	cl_queue	cl_context	cl_device	cl_platform
4	sycl	sycl::queue	sycl::context	sycl::device	sycl::platform
5	hip	hipStream_t	hipCtx_t	hip_device_t	N/A
6	level_zero	ze_command_queue_handle_t	ze_context_handle_t	ze_device_handle_t	ze_driver_handle_t
7	hsa	hsa_queue_t*	N/A	hsa_agent_t	N/A

➡ Supported in oneAPI

hsa: Heterogenous System Architecture Foundation

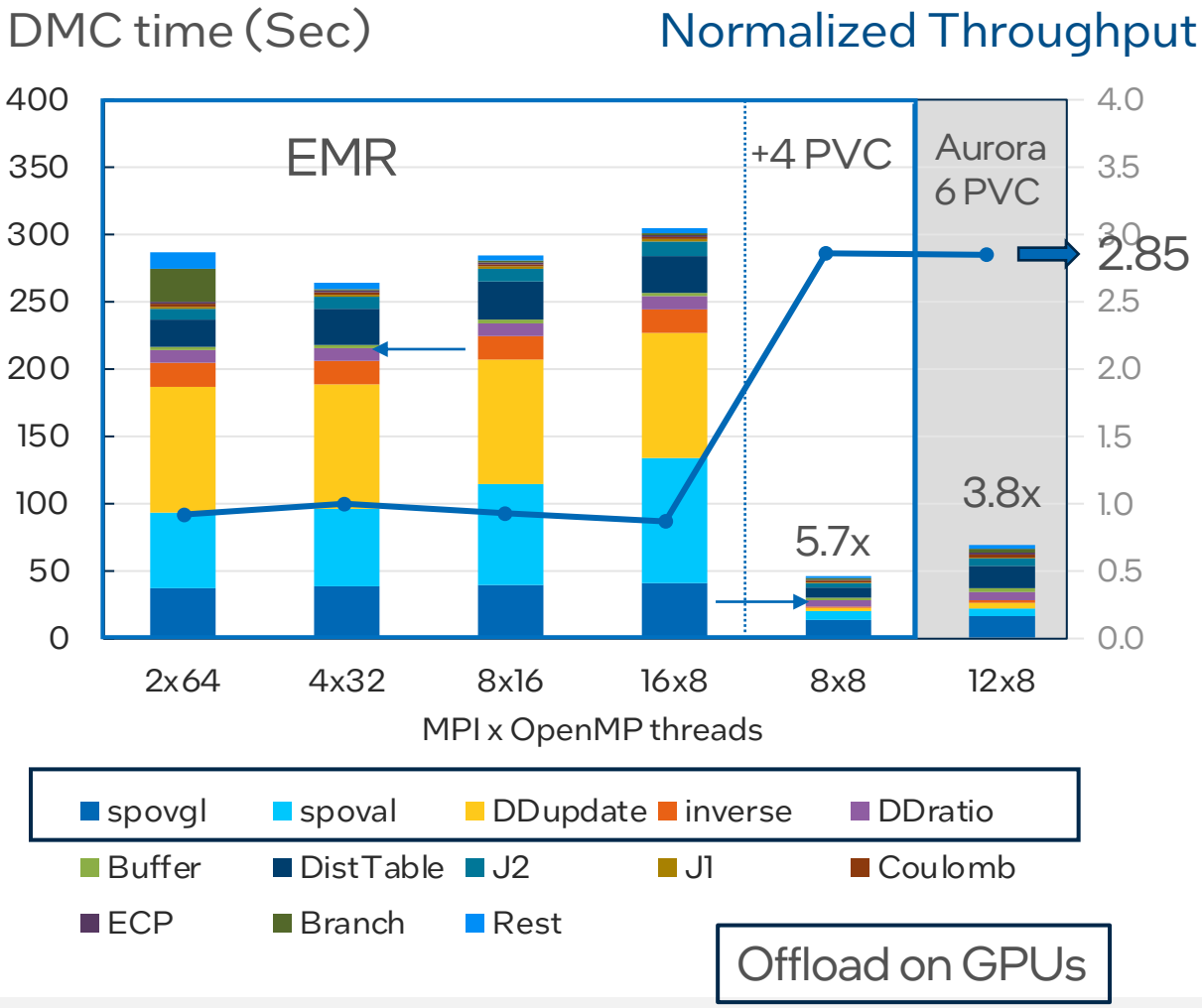
QMCPACK with OpenMP Interop

http://www.qmcpack.org

- Use MPI, OpenMP, OpenMP/SYCL offload and MKL
- FOM, MC Samples/sec (throughput)
- Optimal MPI rank and OpenMP threads depend on HW and problems - memory footprint, BW utilizations, FLOPs
- NiO -a512 (6144 els) on EMR + 4PVC
  - 4x32 for CPU-only: memory (DDR/L3) optimization
  - 8x8 for with 8 PVC tiles
  - 12x 8 on Aurora node with 12 tiles

EMR + 4 PVC node
- INTEL(R) XEON(R) PLATINUM 8592+
  - 128 Cores, 4 NUMA nodes, 2 SOCKETS
- 4 Data Center GPU Max 1550 (code name PVC)
- oneAPI 2025.1

FOM : Figure of Merit of QMC calculations
Throughputs normalized by 4x32 (1 rank per NUMA)



OpenMP Interoperability for AI/HPC Ecosystem

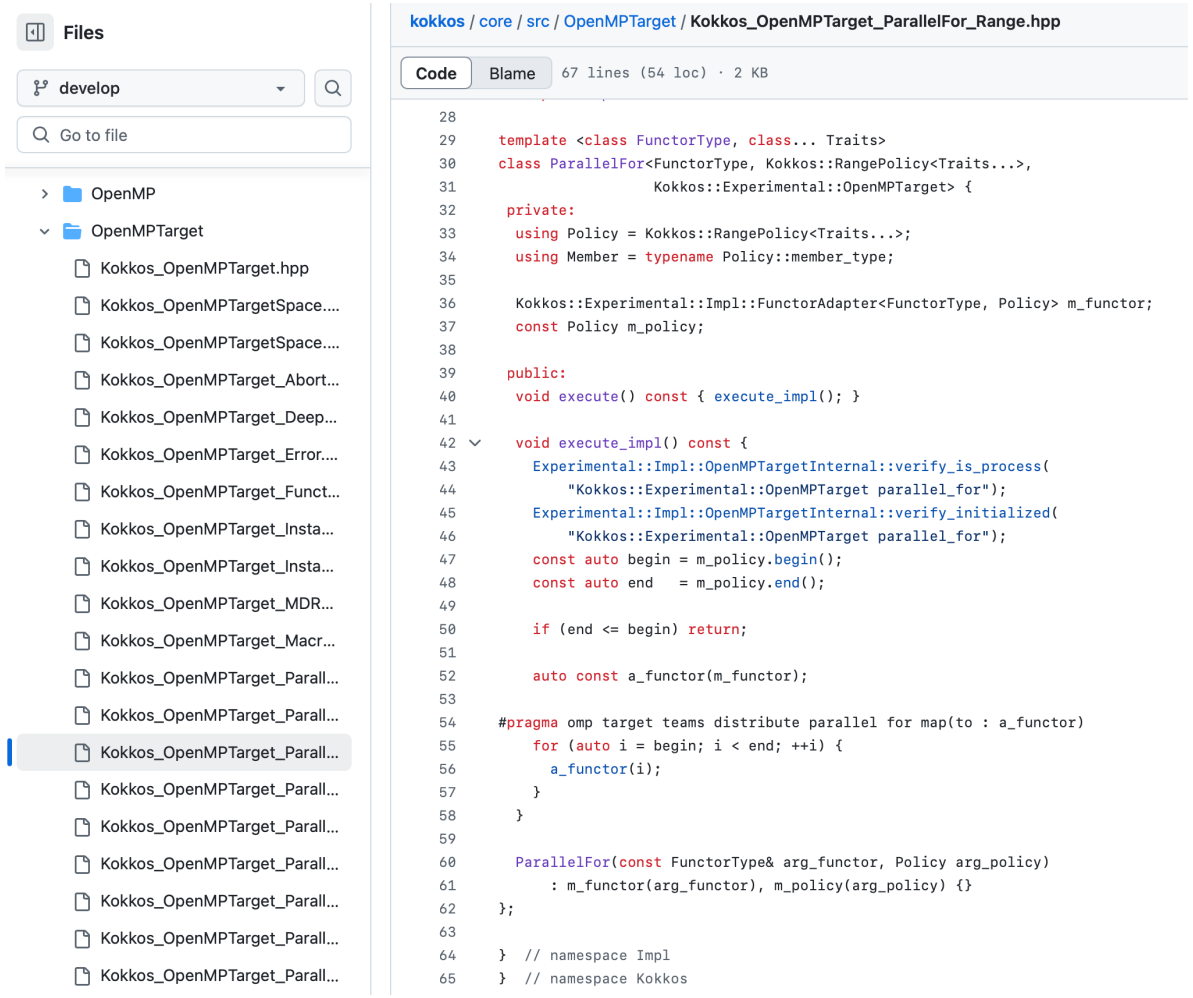
- Core to all OpenMP/MKL applications: QMCPACK, GAMESS, VASP, BerkeleyGW, ThunderSVM, …
- Facilitate use of SYCL libraries with minimum changes
- Integrated in OpenMP backend of frameworks: Kokkos-OMP-OMPT
- Interfacing with Python: numpy, dpctl/dpnp
- Opens up AI/HPC applications: tight integration of pytorch/JAX and user OpenMP codes, tinker-HP
- Any programming model with C/C++ binding

Require performance tuning like any scientific HPComputing application!

OpenMP Applications

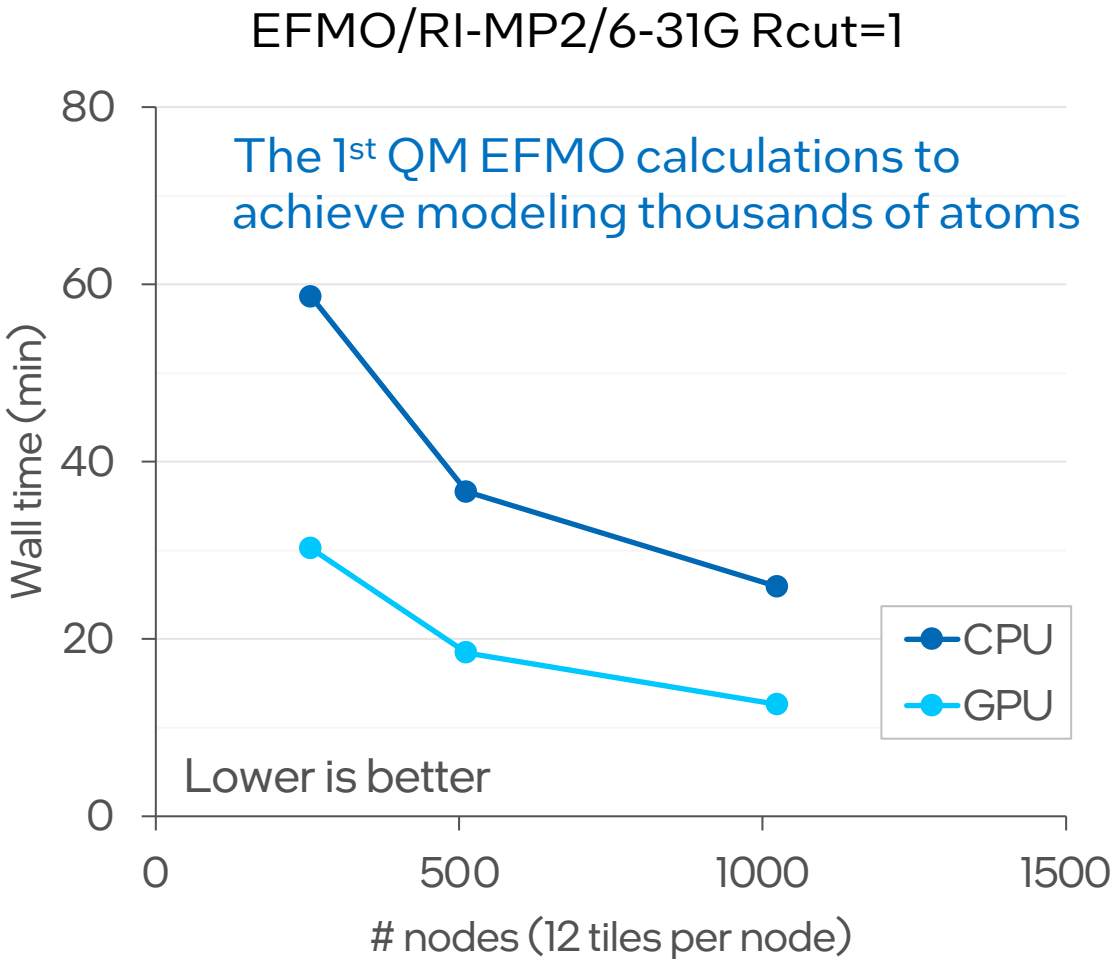
Enabled by oneAPI

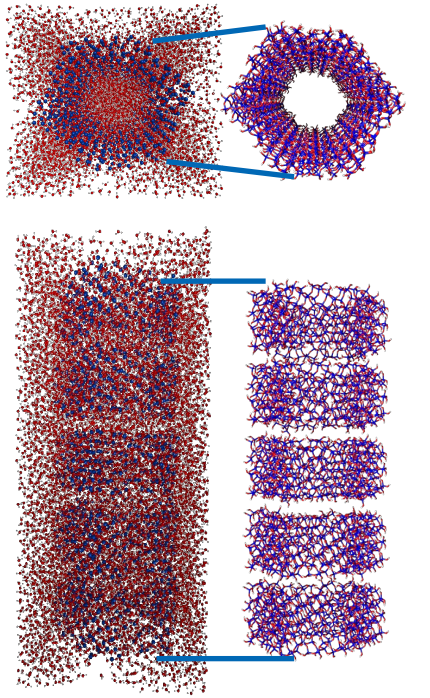
Kokkos Backend with OpenMP Target



- Kokkos has GPU support through OpenMP. See:
  - https://github.com/kokkos/kokkos/blob/develop/core/src/OpenMPTarget
- See implementation of simple Kokkos::parallel_for on the left
- More advanced use cases of OpenMP in Kokkos are:
  - Hierarchical parallelism
  - User-defined reductions

GAMESS Effective Fragment Molecular Orbital method on PVC

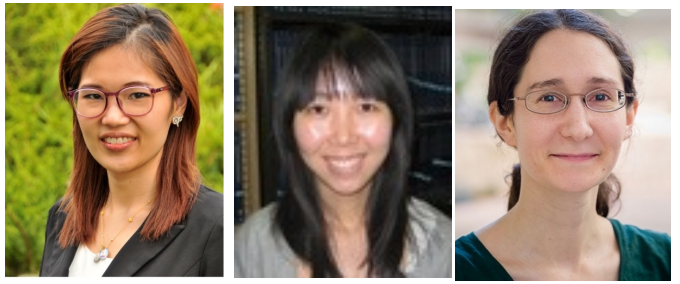




Hydrated MSNs
Total: 68490 atoms

Key to achievement

New efficient MAKEFP
New GPU codes
- RHF, CPHF, & TDHF
- Correlated methods
- RI-MP2
- RI-CC



T. Sattasathuchana & P. Xu, Univ. of Hawaii - Manoa, and C. Bertoni, ANL

Calculations on US-DOE Exascale Supercomputers Auroa (ALCF) and Frontier (OLCF)

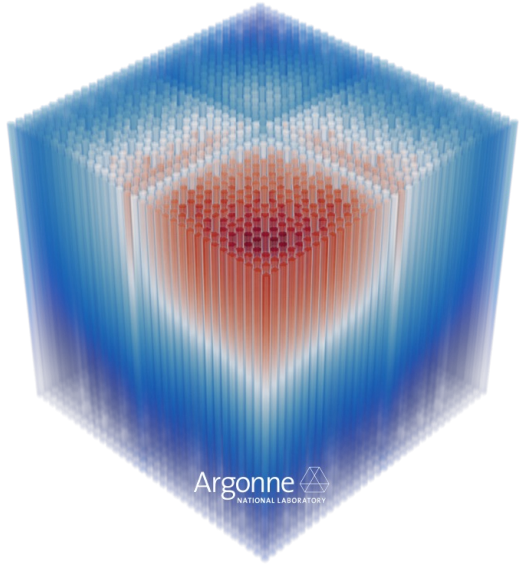
https://www.msg.chem.iastate.edu/gamess/
https://sites.google.com/view/tosa-qc-research-group

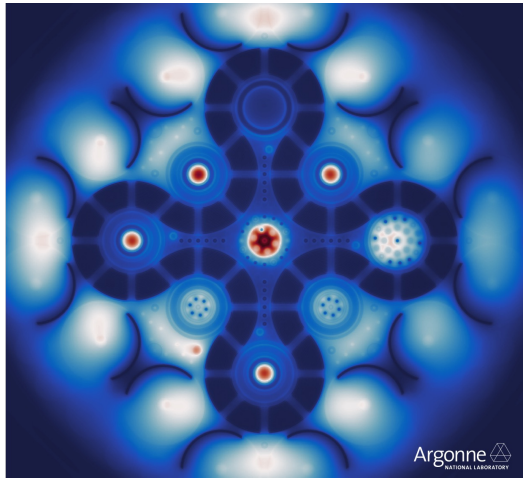
OpenMC Performance on PVC

Monte Carlo Particle Transport Application

John Tramm, ANL, ISC 2023

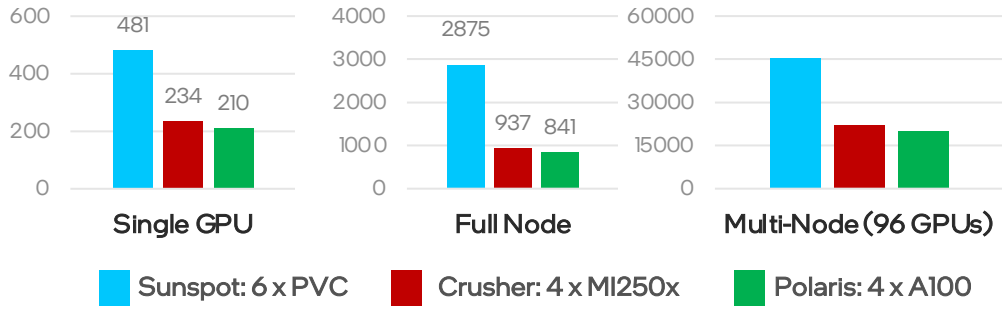
ECP project name: ExaSMR
Project PI: Steve Hamilton
https://docs.openmc.org

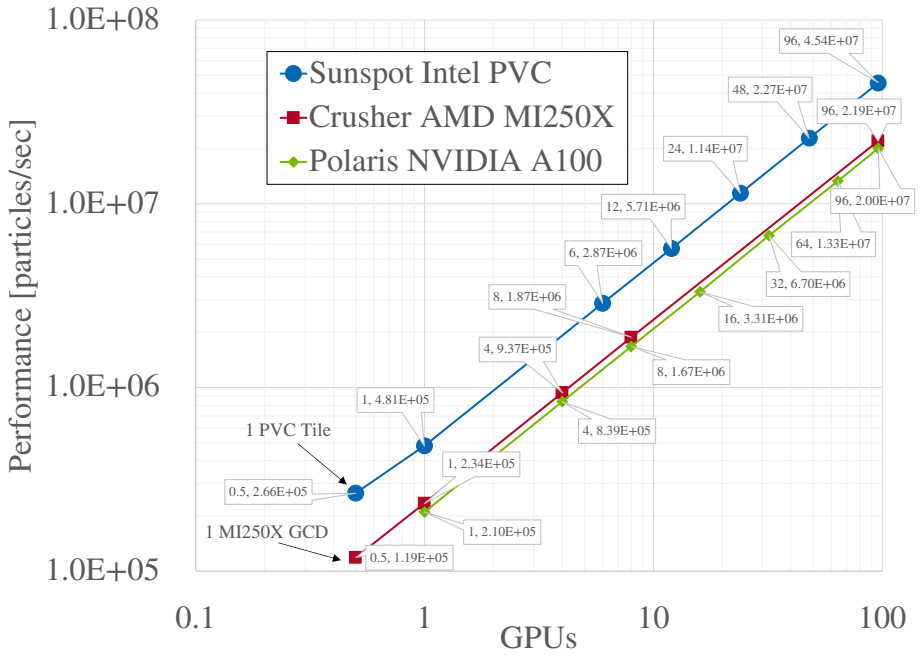




Application Summary: OpenMC is a Monte Carlo particle transport application that has recently been ported to the OpenMP target offloading programming model for use on GPU-based systems. The Monte Carlo method employed by OpenMC is considered the "gold standard" for high-fidelity simulation while also having the advantage of being a general-purpose method able to simulate nearly any geometry or material without the need for domain-specific assumptions. However, despite the extreme advantages in ease of use and accuracy, Monte Carlo methods like those in OpenMC often suffer from a very high computational cost. The extreme performance gains OpenMC has achieved on GPUs, as compared to traditional CPU architectures, is finally bringing within reach a much larger class of problems that historically were deemed too expensive to simulate using Monte Carlo methods. The leap in performance that GPUs are now offering carries with it the potential to disrupt a number of engineering technology stacks that have traditionally been dominated by non-general deterministic methods. For instance, faster MC applications may greatly expand the design space and simplify the regulation process for new nuclear reactor designs -- potentially improving the economics of nuclear energy and therefore helping to solve the world's climate crisis.

Performance [kparticles/sec] (higher is better)

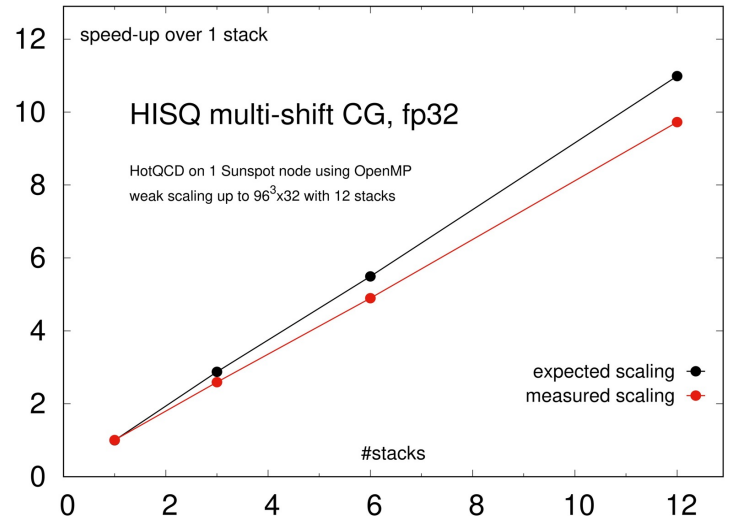




Lattice QCD on PVC with OpenMP Target

QUDA:

HotQCD:

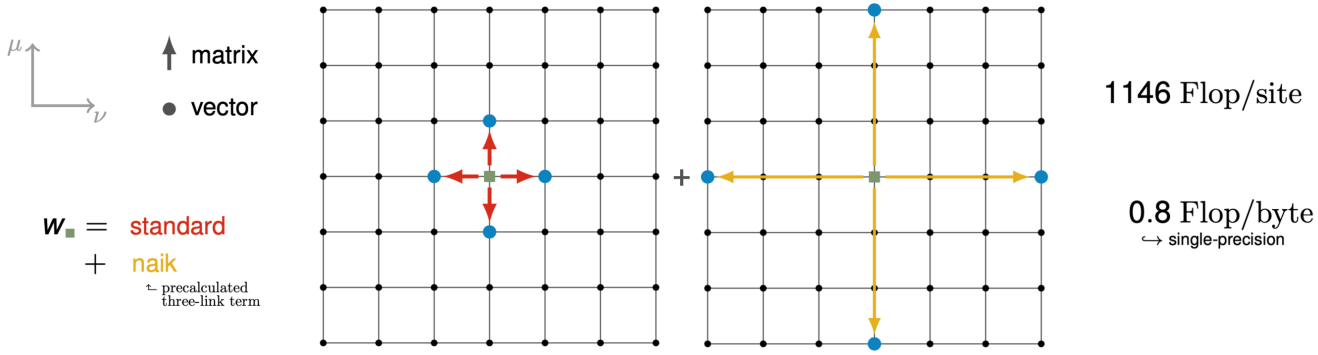


$$w_n = \sum_{\mu=0}^{4} \left[\left(U_{n,\mu} v_{n+\mu} - U^{\dagger}_{n-\mu,\mu} v_{n-\mu} \right) + \left(N_{n,\mu} v_{n+3\mu} - N^{\dagger}_{n-3\mu,\mu} v_{n-3\mu} \right) \right]$$

complex 3-dim vector

complex 3×3 matrix

$U(3)$ matrix
↪ reconstruct from 14 floats



1146 Flop/site

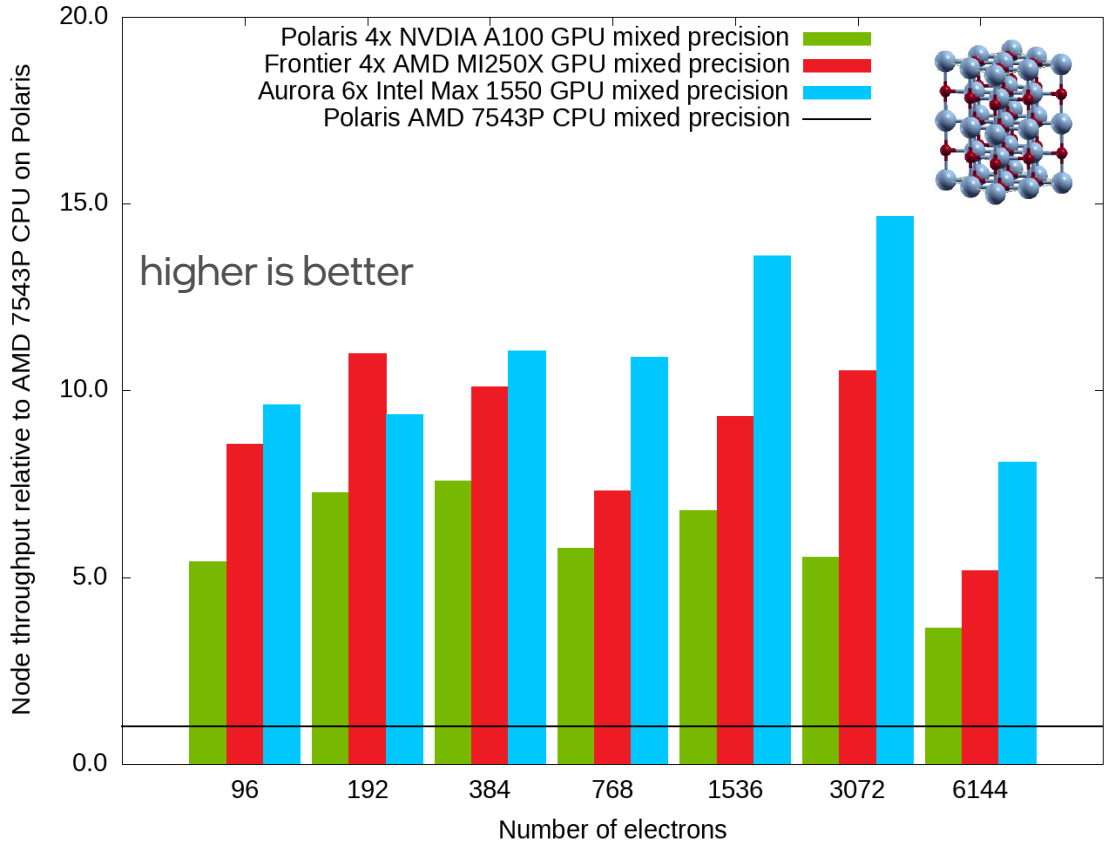
0.8 Flop/byte
↪ single-precision

https://www.usqcd.org

OpenMP: Portability, Performance and Productivity Tool for Scientific High-Performance Computing

- Supported by multiple compilers on diverse CPUs and GPUs
- Portable codes with specializations as needed
- Interoperable with performance libraries and device programming models, SYCL/CUDA/HIP …
- Enhanced control of allocations, vectorizations, code transformations, executions for performance tuning
- Incremental development, always science-ready
- OpenMP 6 for Easier Programming: *Develop Parallel Programs Easily and More Control to Developers*

QMCPACK NiO ECP Benchmark Node Throughput



Ye Luo (ALCF), March 2025. http://www.qmcpack.org

Resources

- https://www.openmp.org
- https://software.intel.com/oneAPI
- oneAPI GPU optimization guide
- Training and events at Argonne Leadership Computing Facility
- Contact jeongnim.kim_at_intel.com


intel