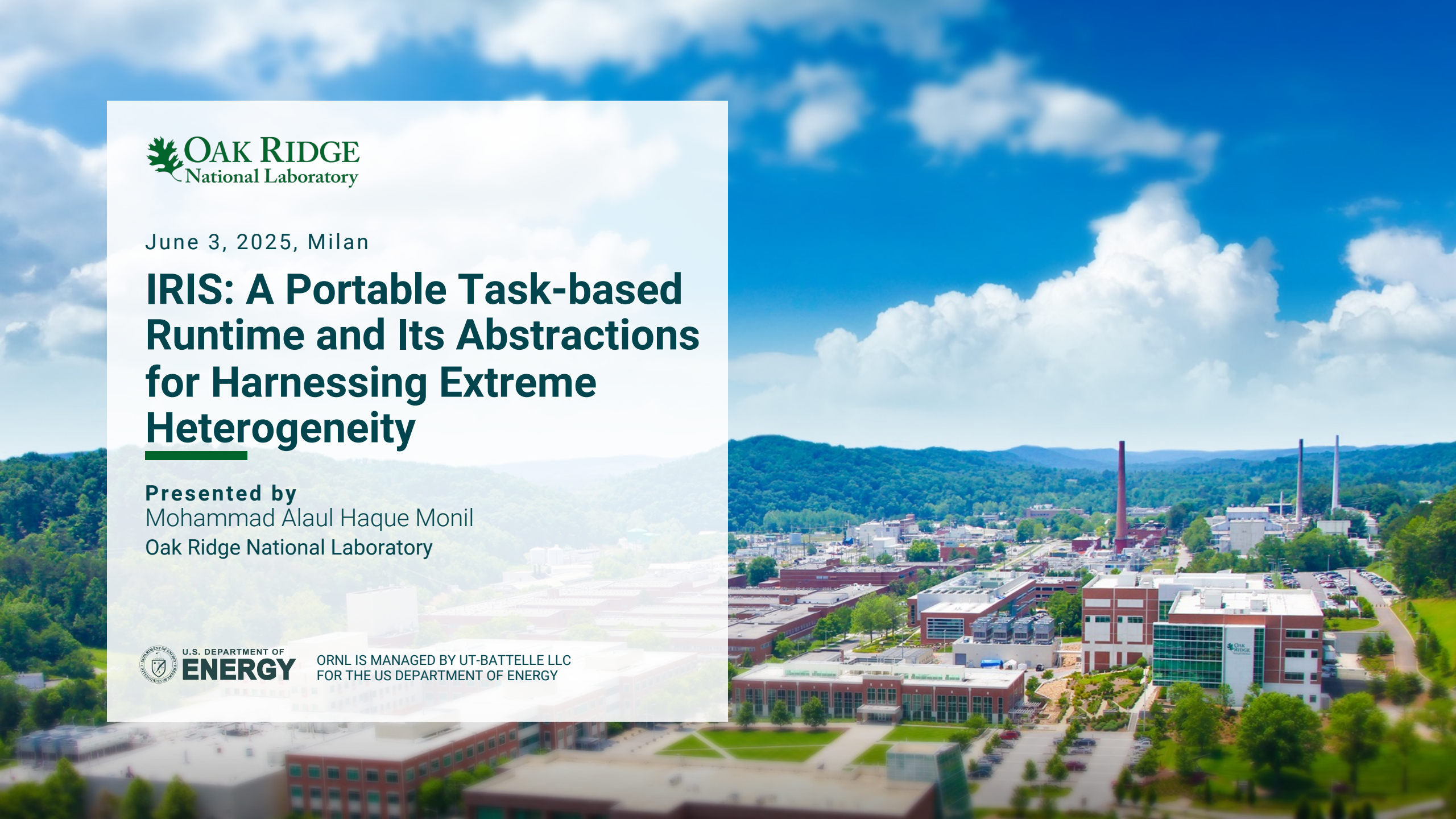


OAK RIDGE National Laboratory

June 3, 2025, Milan

IRIS: A Portable Task-based Runtime and Its Abstractions for Harnessing Extreme Heterogeneity

Presented by
Mohammad Alaul Haque Monil
Oak Ridge National Laboratory

U.S. DEPARTMENT OF ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

Outline

- IRIS Runtime Overview
  - Motivation
  - Core capabilities
  - Short overview of high level abstractions built on IRIS Runtime
- MatRIS: A portable linear algebra library for multi-device heterogeneity
- IRISX: Exposing auto tuning opportunity multi-device heterogeneity for scientific computing
- ChatIRIS: A generative AI based approach for task-based programming

IRIS Runtime: Motivation

- HPC platforms, embedded systems and mobile devices are heterogeneous
- For energy efficiency, experts predict more heterogeneity in HPC domain (Reference: keynote of DOE Energy Efficiency Workshop 2025)
- Current portable abstractions provide partial solution to the problem of heterogeneous orchestrations
- For the case of extreme heterogeneity, an ideal portable solution should provide
  - Architecture agnostic programming model regardless of underlying architecture
  - Automatic orchestration and data movement in multi-device heterogenous system
- IRIS Runtime aims to provide the ideal solution through its abstractions, programming model and runtime features

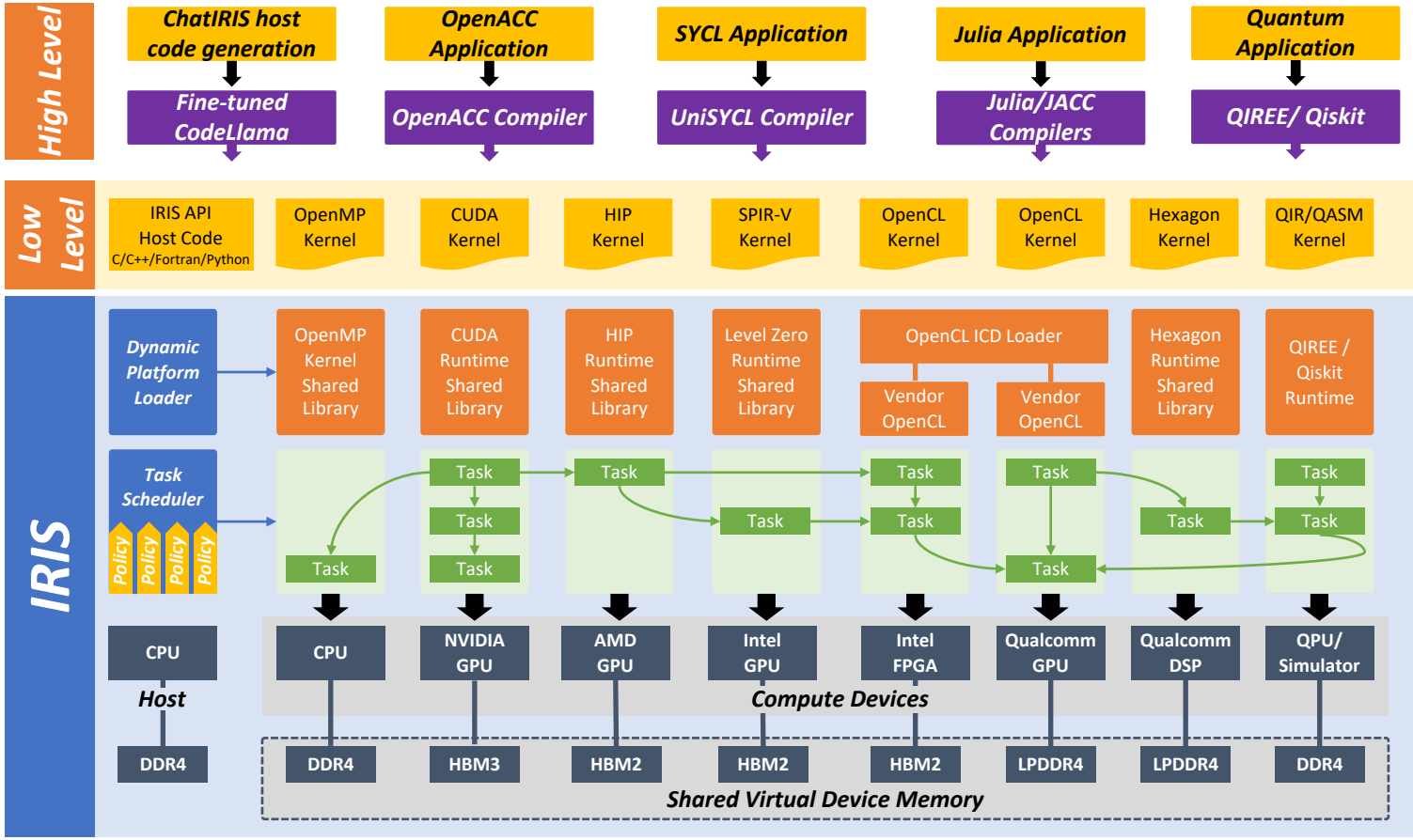
IRIS Runtime : A Runtime for Extreme Heterogeneity

IRIS Programming Model

- Architecture agnostic Task and Memory objects
- Task points to kernels
- Dependencies
- Ability to create task graphs
- Supports C, C++, Python, Fortran

Supported Hardware

- Target: HPC, Cloud, and embedded systems
- Supports CPUs, GPUs, FPGAs and DSPs



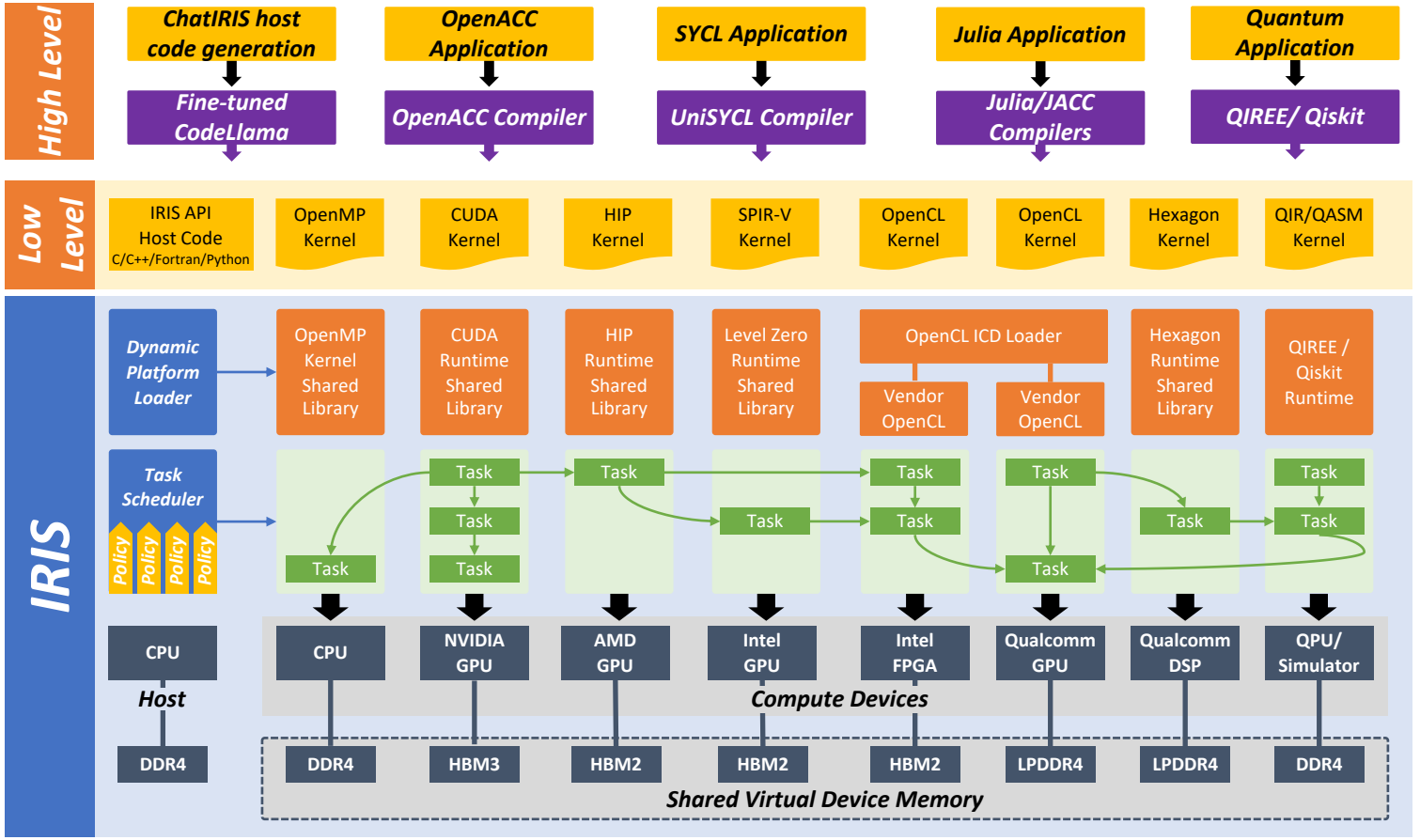
- IRIS Website: https://iris-programming.github.io/
- IRIS Public: https://github.com/ORNL/iris
- IRIS Private (most updated): https://code.ornl.gov/brisbane/iris

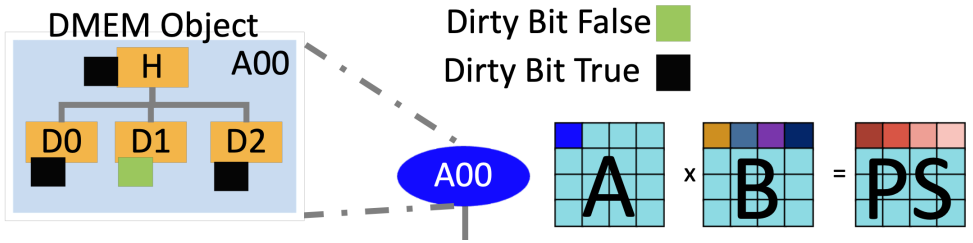
2024 R&D 100 WINNER

IRIS Runtime : A Runtime for Extreme Heterogeneity

Memory Model

- Handles heterogenous memory address spaces for application data objects
- Automatic data movement (DMEM)
  - To and from Device
  - Device to device
  - Automatic Flush/write of data objects to host
  - Proactive and reactive communication
  - Asynchronous communication

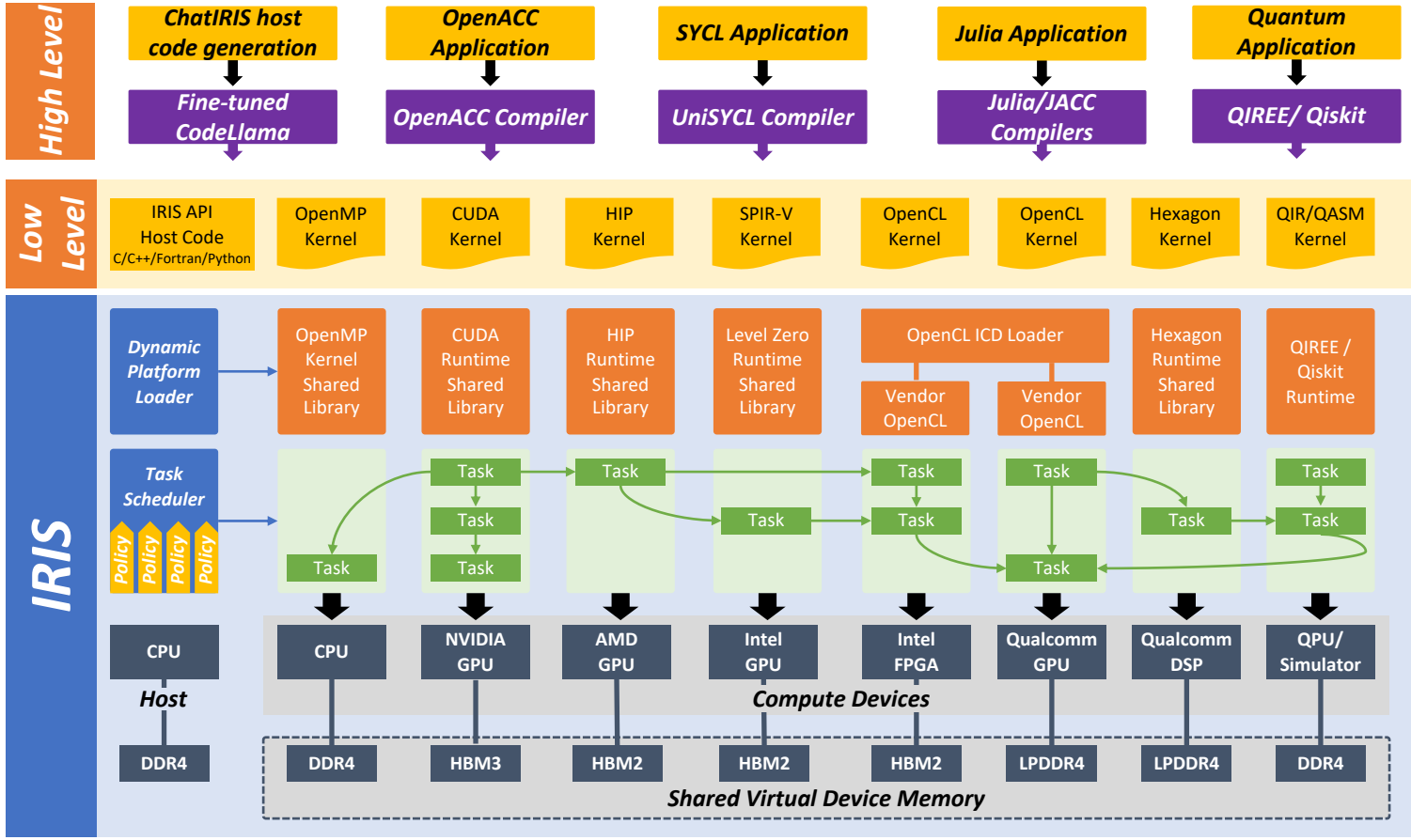


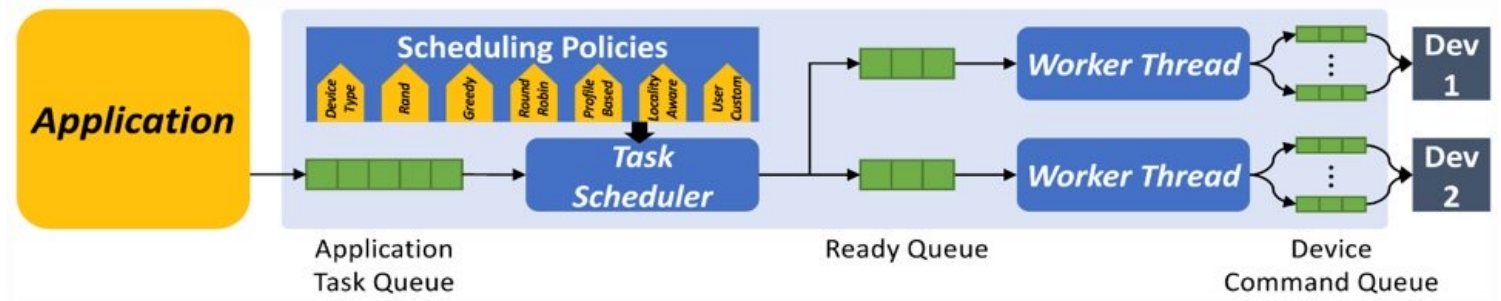


IRIS Runtime : A Runtime for Extreme Heterogeneity

Execution Model

- Dynamic platform loader
- Data flow enabled DAG creation
- Asynchronous execution with computation and communication overlap using streams
  - Dedicated streams for computation and communication
- DAG and Task fusion
- A rich set of schedulers
  - Work stealing
  - Locality
  - Graph Neural Network based
  - Custom schedulers





IRIS Host and Kernels for SAXPY

Host Code:

C++ | C | Python | Fortran

```cpp
int main(int argc, char** argv) {
  size_t SIZE;
  float *X, *Y, *Z;
  float A = 10;
  int ERROR = 0;
  iris::Platform platform;
  platform.init(&argc, &argv, 1);

  SIZE = argc > 1 ? atol(argv[1]) : 8;

  X = (float*) malloc(SIZE * sizeof(float));
  Y = (float*) malloc(SIZE * sizeof(float));
  Z = (float*) malloc(SIZE * sizeof(float));

  for (int i = 0; i < SIZE; i++) {
    X[i] = i; Y[i] = i;
  }

  iris::DMem mem_X(X, SIZE * sizeof(float));
  iris::DMem mem_Y(Y, SIZE * sizeof(float));
  iris::DMem mem_Z(Z, SIZE * sizeof(float));

  iris::Task task;
  void* params0[4] = { &mem_Z, &A, &mem_X, &mem_Y };
  int pinfo0[4] = { iris_w, sizeof(A), iris_r, iris_r };
  task.kernel("saxpy", 1, NULL, &SIZE, NULL, 4, params0, pinfo0);
  task.flush_out(mem_Z);
  task.submit(1, NULL, 1);

  for (int i = 0; i < SIZE; i++) {
    if (Z[i] != A * X[i] + Y[i]) ERROR++;
  }
  free(X);  free(Y); free(Z);

  platform.finalize();
  return 0;
}
```

Kernels

CUDA | HIP | OpenCL | OpenMP | Hexagon | Xilinx C++

```cpp
static void saxpy(float* Z, float A, float* X, float* Y) {
  size_t i;
#pragma omp parallel for shared(Z, A, X, Y) private(i)
  IRIS_OPENMP_KERNEL_BEGIN(i)
  Z[i] = A * X[i] + Y[i];
  IRIS_OPENMP_KERNEL_END
}
```

CUDA | HIP | OpenCL | OpenMP | Hexagon | Xilinx C++

```cpp
extern "C" __global__ void saxpy(float* Z, float A, float* X, float* Y) {
  size_t id = blockIdx.x * blockDim.x + threadIdx.x;
  Z[id] = A * X[id] + Y[id];
}
```

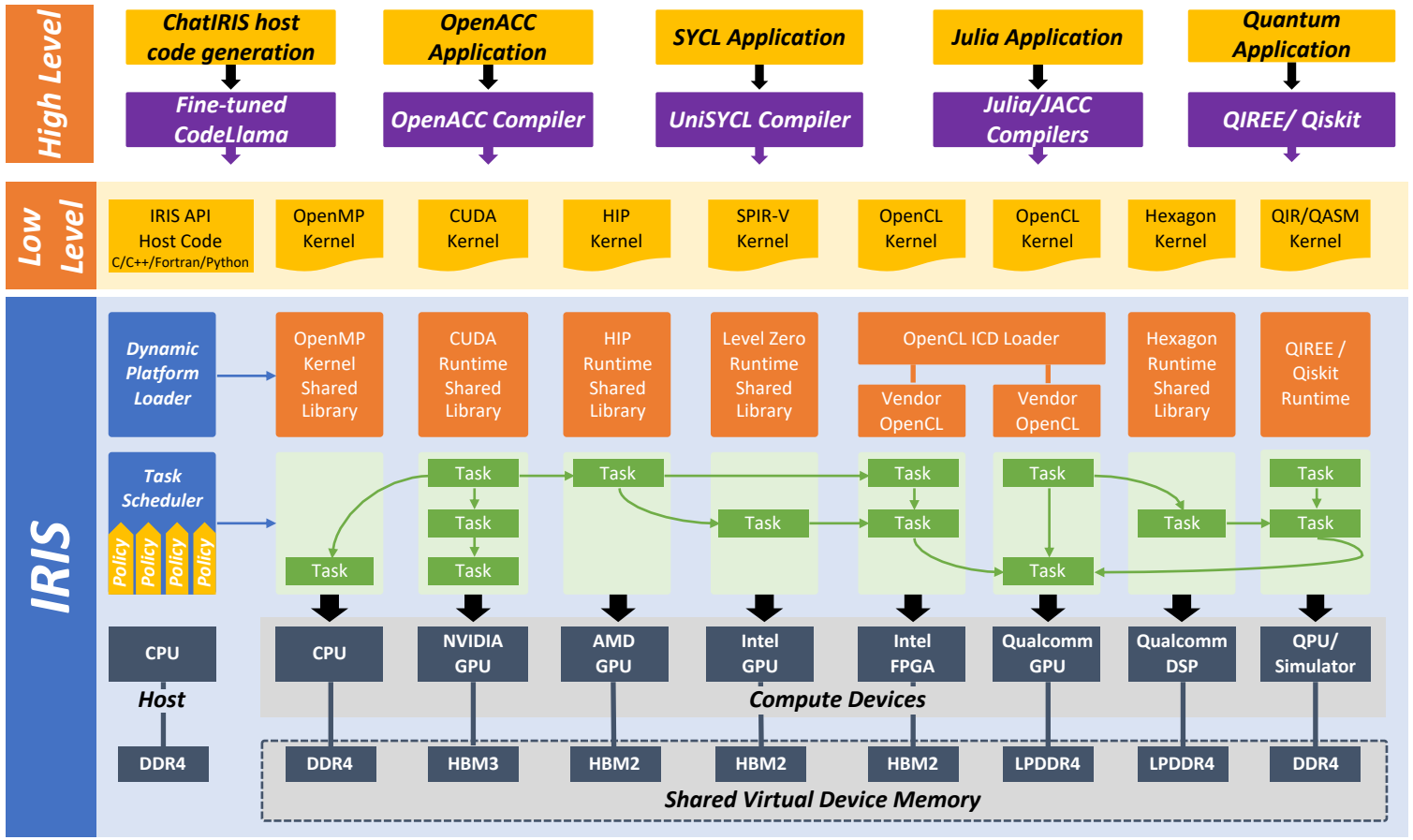
IRIS Runtime : High Level Abstractions

Why?

- Programming productivity

High Level Programming models and abstractions

- Automatic generation of task code
- Extracting the kernels to be invoked from tasks
- IRIS as a back end for orchestration
- Ensures portability



IRIS Runtime : High Level Abstractions

High Level Abstractions for Programming models

- SYCL
- OpenACC
- Julia

Quantum Computing in IRIS

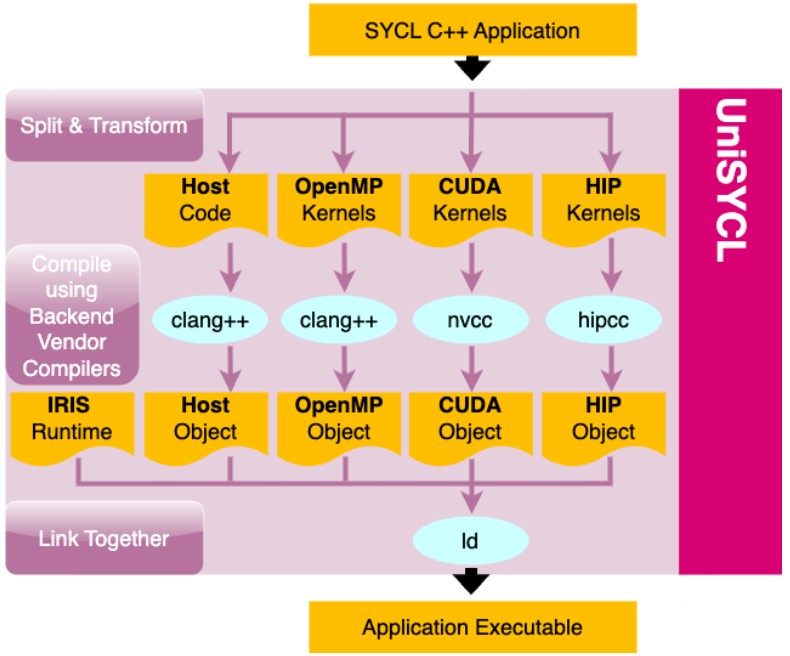
- Q-IRIS

Domain Specific Abstraction

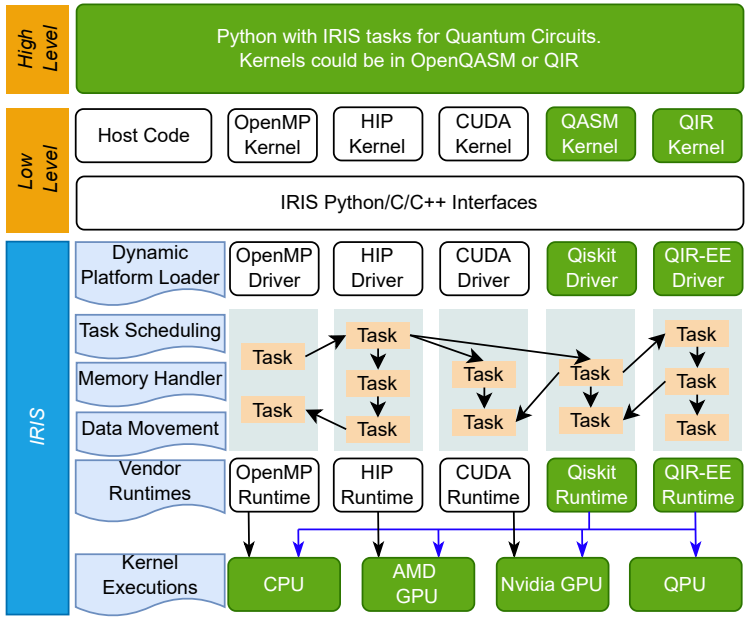
- **MatRIS: Math Library**
- **IRISX: Scientific computing**

Gen AI for Generating Task Code

- **ChatIRIS**



Uni-SYCL: An example of IRIS enabling high level programming models



Q-IRIS: An example of Forward looking heterogeneous architectures

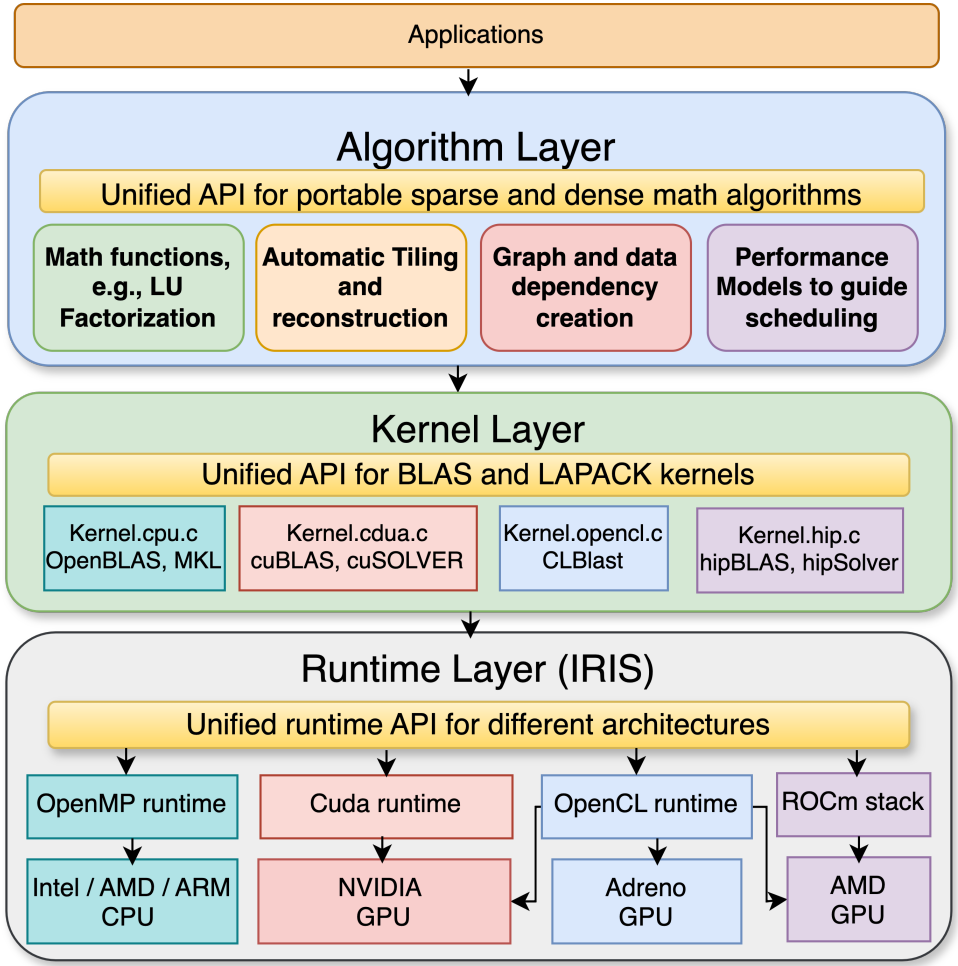
MatRIS: Performance Portable Math Library of IRIS Runtime for Multi-device Heterogeneity

A performance portable abstraction for math library

- Philosophy: Implement once, deploy anywhere
- Focus: in-node extreme heterogeneity

Provides architecture-agnostic and serial frontend

- Frontend: same as state-of-the-art math libraries
- Underneath: portability and heterogeneity by IRIS



MatRIS: Layered Abstractions

Runtime Layer (IRIS Runtime)

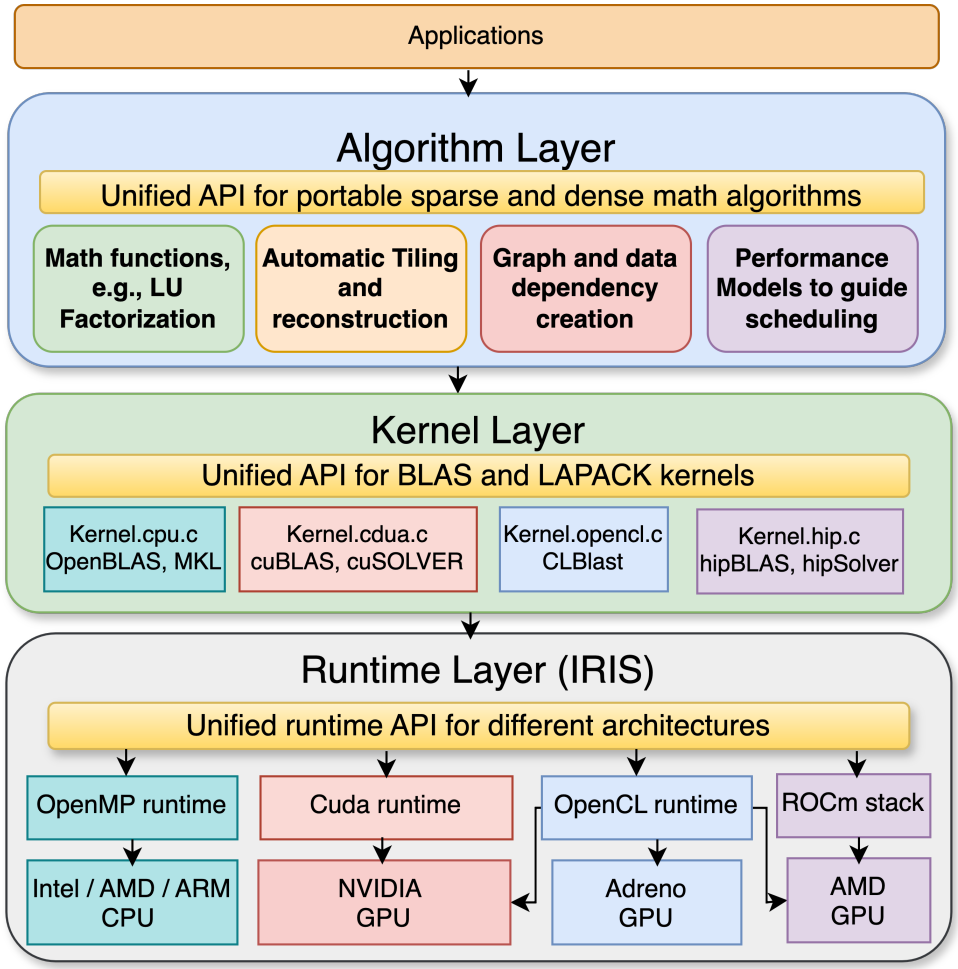
- Orchestrates vendor and open-source runtimes

Kernel Layer

- Invokes vendors and open-source math library kernels
- Unified APIs for BLAS and LAPACK kernels

Algorithm Layer

- Defines architecture-agnostic algorithm
- Portable and heterogeneous implementation of GETRF, POTRF, TRSM, GEMM, POSV, and GESV
- Supports portable multi-GPU batched operations



MatRIS: Serial Code to Heterogeneity

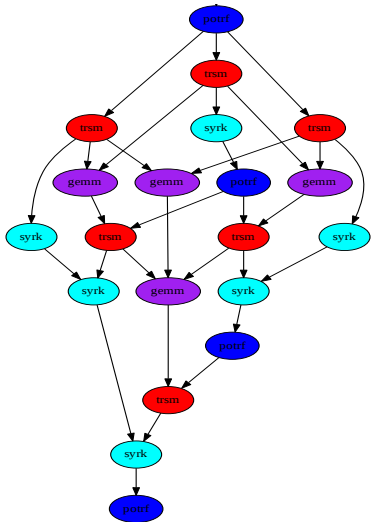
```
Tiling2D<DTYPE> A_tile(A, M, N, row, col,..);
if (uplo == MATRIS_BLAS_LOWER) {
    for (int k = 0; k < A_tild.row_tiles_count(); k++) {
        IRIS_FUNC(graph_, POTRF_TYPE)
            ( graph... A_tild.GetAt(k,k).IRISMem()...);
        for (int m = k+1; m < A_tild.row_tiles_count(); m++) {
        IRIS_FUNC(graph_, TRSM_TYPE)
            ( graph...A_tild.GetAt(k,m).IRISMem()...);
        }
        for (int m = k+1; m < A_tild.row_tiles_count(); m++) {
        IRIS_FUNC(graph_, SYRK_TYPE)
            ( graph...A_tild.GetAt(m,m).IRISMem());
        for (int n = k+1; n < m; n++) {
            IRIS_FUNC(graph_, GEMM_TYPE)
                ( graph...A_tild.GetAt(n,m).IRISMem());
        }
        }
    }
}
```

Serial Code

MatRIS: Serial Code to Heterogeneity

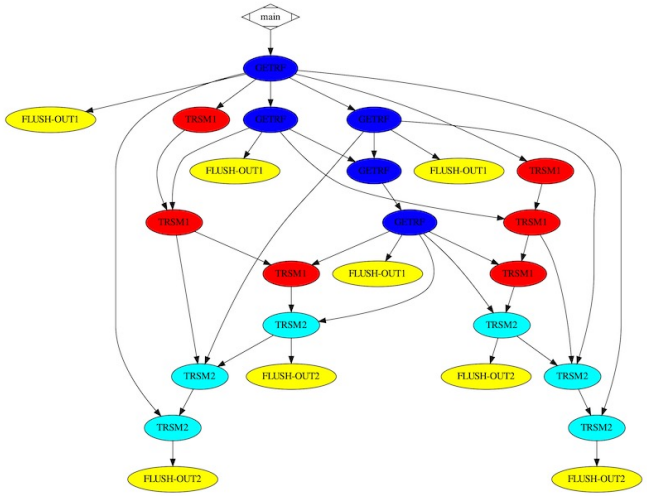
```
Tiling2D<DTYPE> A_tile(A, M, N, row, col,..);
if (uplo == MATRIS_BLAS_LOWER) {
    for (int k = 0; k < A_tild.row_tiles_count(); k++) {
        IRIS_FUNC(graph_, POTRF_TYPE)
            ( graph... A_tild.GetAt(k,k).IRISMem()...);
        for (int m = k+1; m < A_tild.row_tiles_count(); m++) {
            IRIS_FUNC(graph_, TRSM_TYPE)
                ( graph...A_tild.GetAt(k,m).IRISMem()...);
        }
        for (int m = k+1; m < A_tild.row_tiles_count(); m++) {
            IRIS_FUNC(graph_, SYRK_TYPE)
                ( graph...A_tild.GetAt(m,m).IRISMem());
            for (int n = k+1; n < m; n++) {
                IRIS_FUNC(graph_, GEMM_TYPE)
                    ( graph...A_tild.GetAt(n,m).IRISMem());
            }
        }
    }
}
```

Serial Code

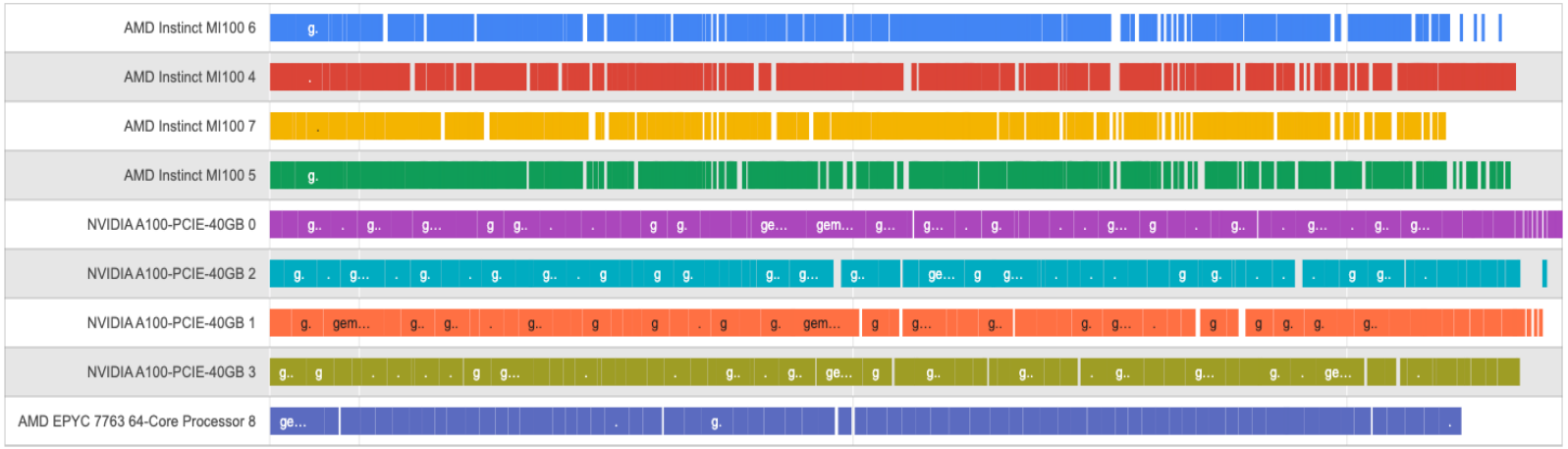


DAG Generation

OR



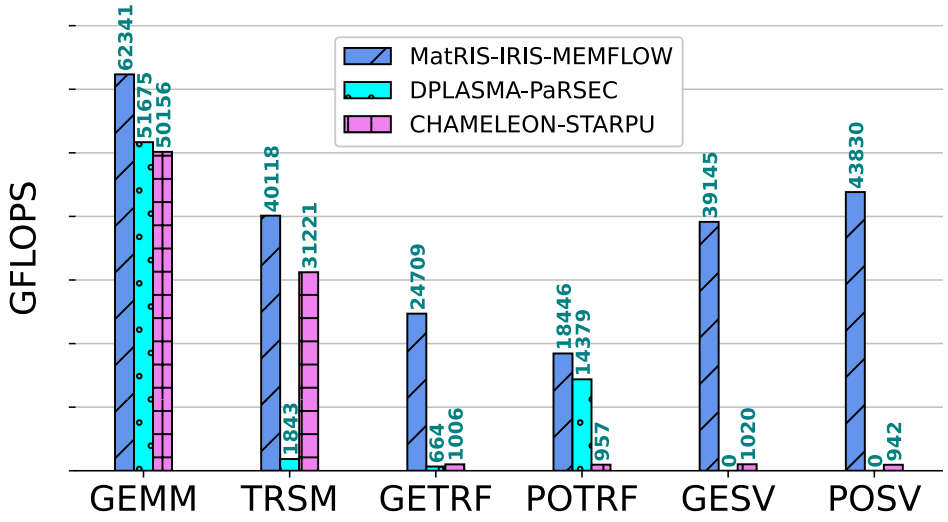
Fused GESV = GETRF + TRSM + TRSM



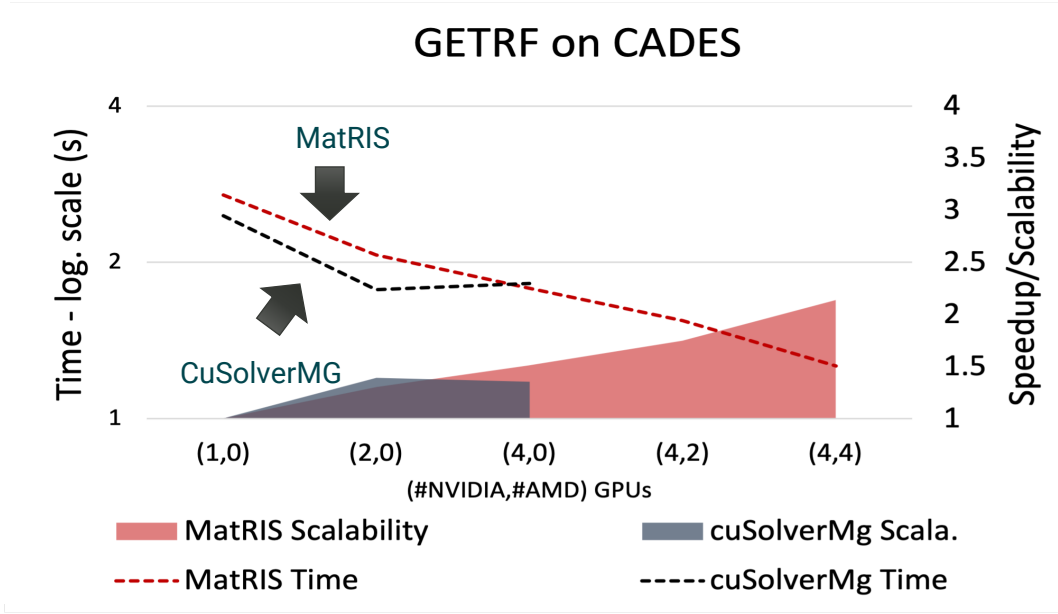
Heterogeneous Execution

MatRIS: Results and Autotuning Opportunities

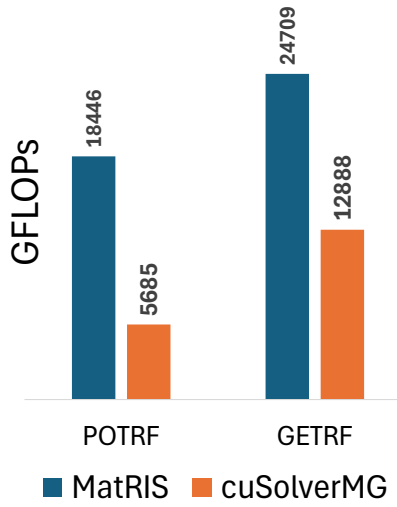
- Scales beyond vendor boundary
  - Advantage of heterogeneous orchestration
- Not only portability
  - Comparison with open source and vendor libraries
- Exposes auto-tuning opportunities



Matrix size; 32,768x32,768, Tile Size: 4,096x4,096,
Platform: 4 NVIDIA A100 GPUs
(No data shown for GESV and POSV in DPLASMA)

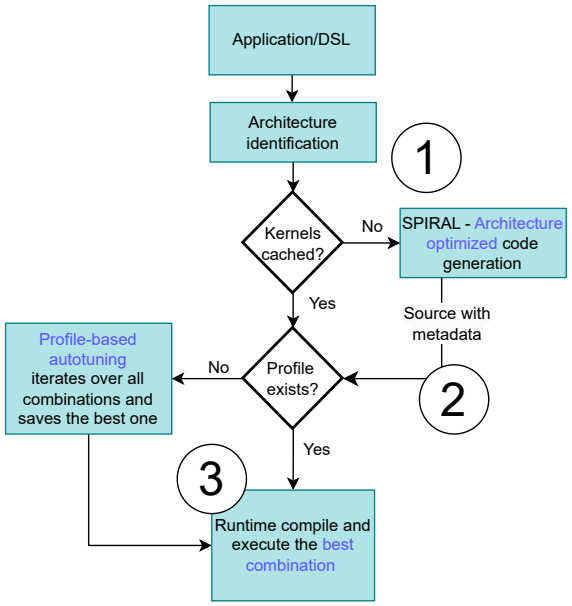


Matrix size; 32768x32768,
Tile Size: 4096x4096,
Platform: 4 NVIDIA A100 GPUs

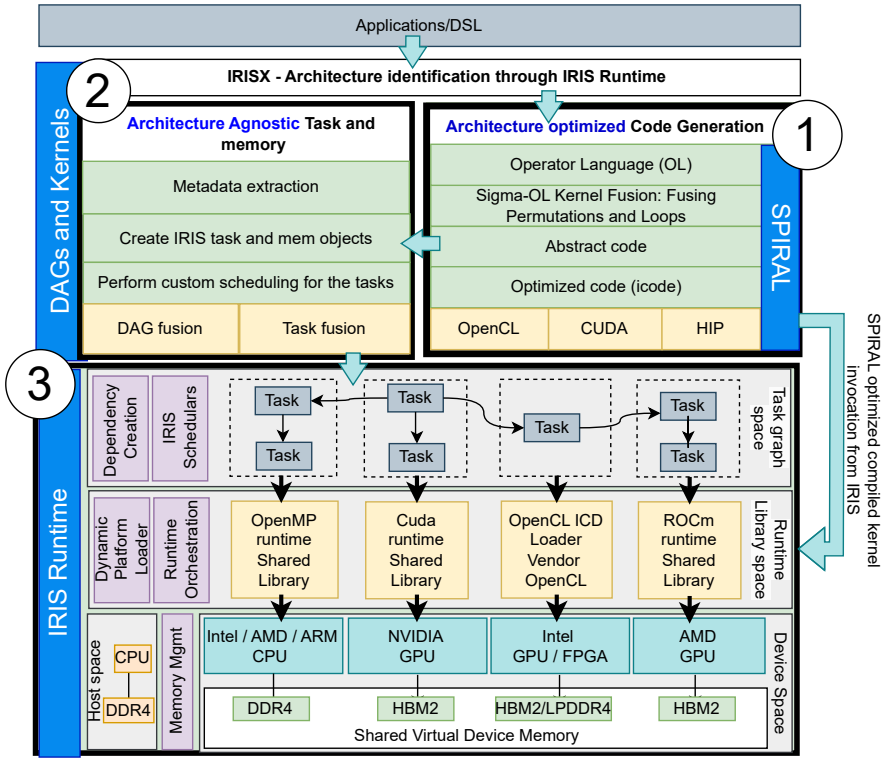


IRISX: A performance portable solution for scientific computing

- Combines code generation and runtime orchestration
  - IRIS: Architecture agnostic frontend
  - SPIRAL: Architecture optimized Kernels
- Capable of adjusting concurrency based on hardware
- Kernel and graph-level transformation using SPIRAL code generation engine and IRIS Runtime
- Finds the best configuration suited for a platform



Block diagram of IRISX

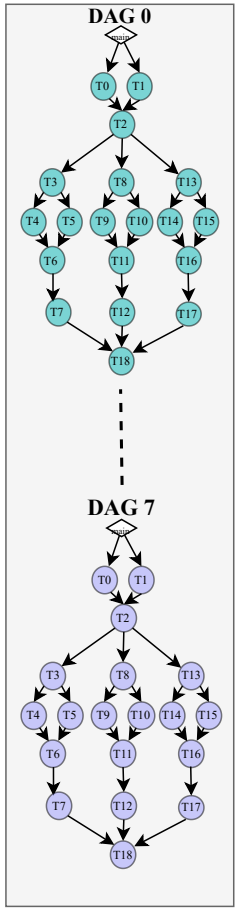


Detail software stack of IRISX

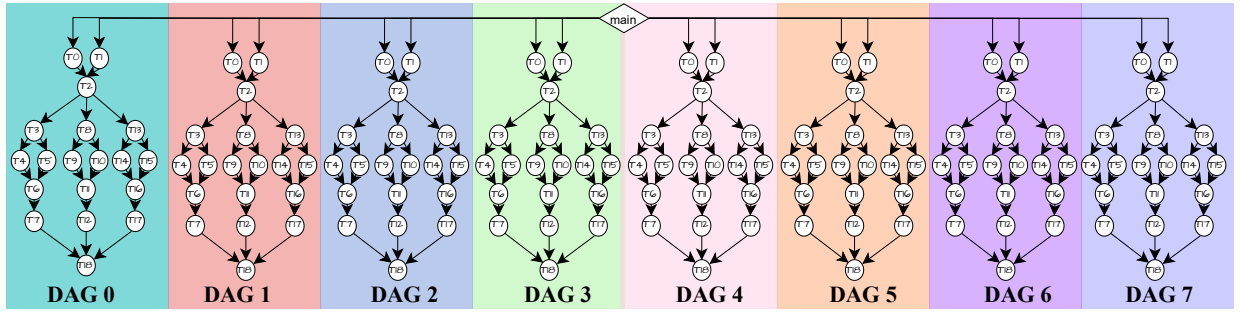
IRISX: DAG Transformation

DAG transformation of 3D Euler equation for controlling concurrency

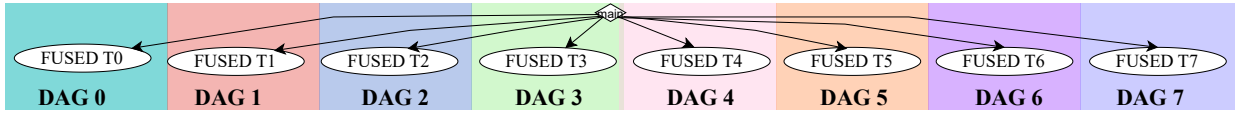
- **Kernel Fusion:** Fuses multiple kernels using SPIRAL
- **Task Fusion:** Fuses multiple Tasks using IRIS Runtime
- **DAG Fusion:** Fuses multiple DAGs using data flow analysis in IRIS Runtime



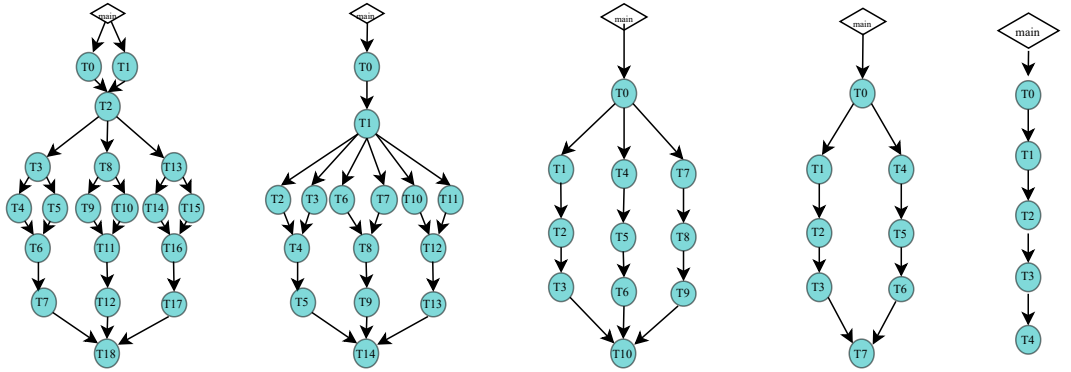
(a) DAG Serial: Basic Task Graph from Proto in IRIS



(b) DAG fusion using automatic data flow in IRIS. A different representation of (a).



(c) DAG with task fusion in IRIS to reduce the number of tasks. A different representation of (a) and (b)

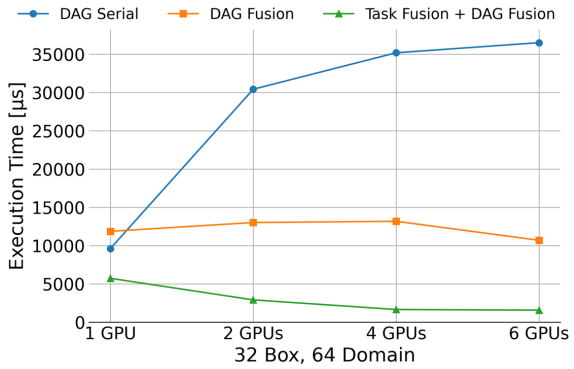


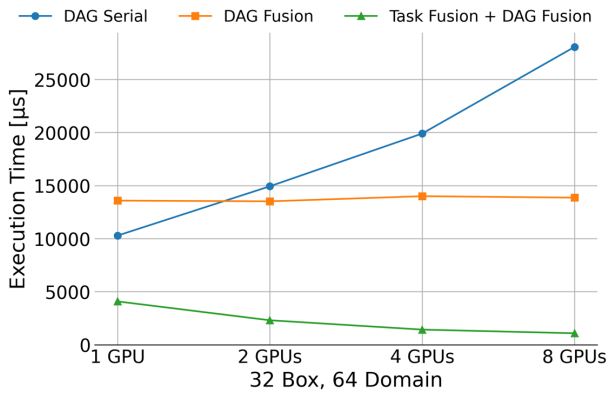
(d) IRIS task representation of the same computation using various kernel fusion options from SPIRAL.

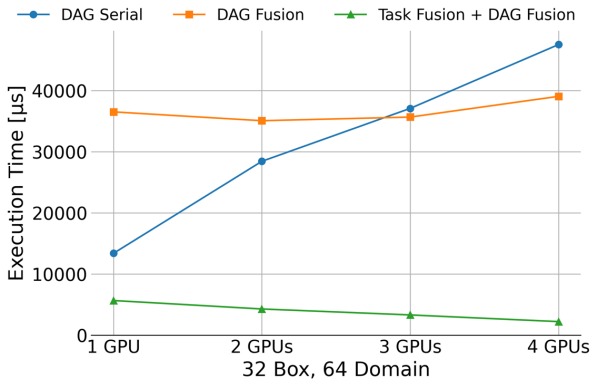
IRISX: Trade-off between Overhead and Concurrency

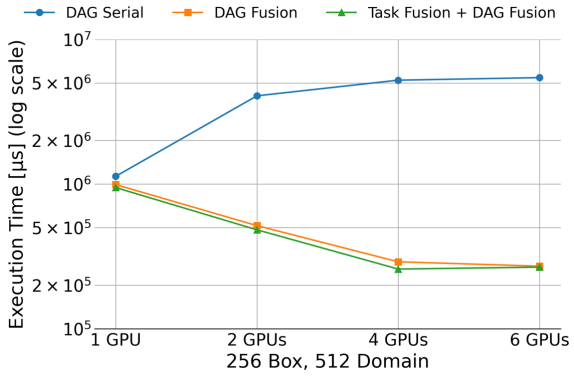
Dominated by task overhead

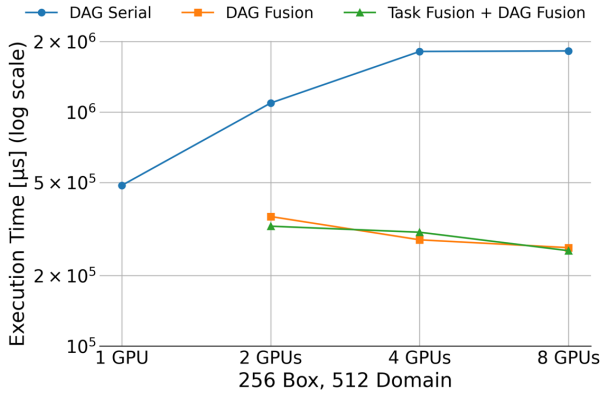
Concurrency provides benefit

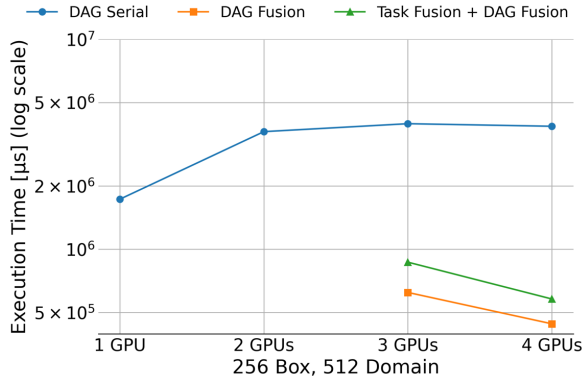












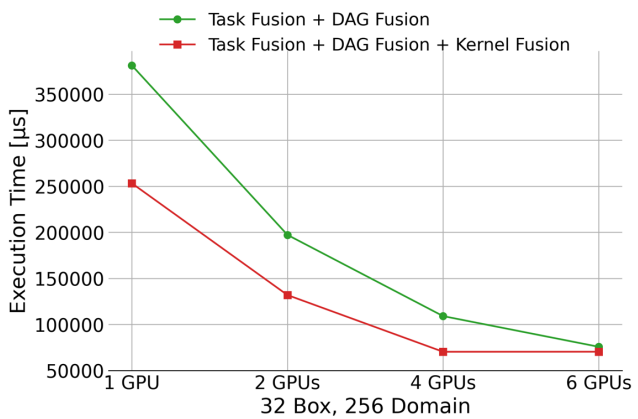
Aurora with 6 Intel Max 1550 GPUs

Frontier with 8 AMD MI250X GCDs

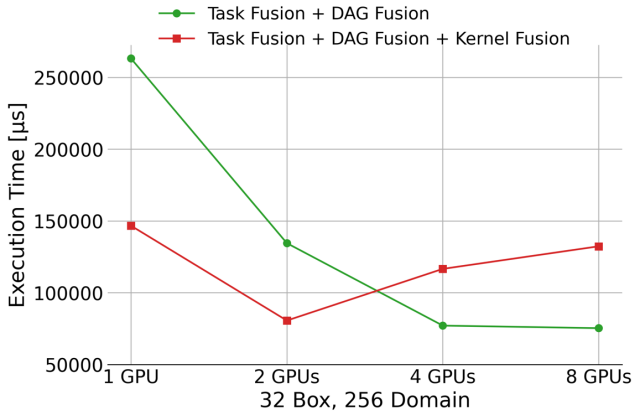
Equinox with 4 NVIDIA V100 GPUs

IRISX: Kernel Fusion and Comparison

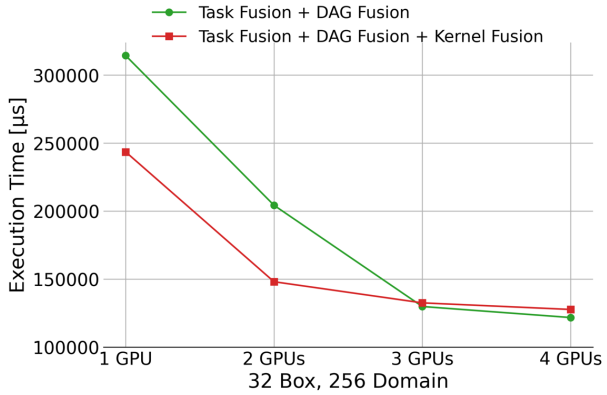
Kernel fusion exposes trade-offs



Aurora with 6 Intel Max 1550 GPUs

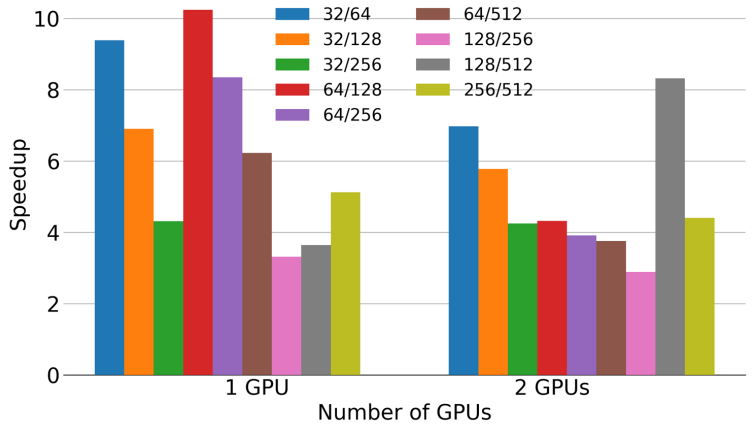


Frontier with 8 AMD MI250X GCDs



Equinox with 4 NVIDIA V100 GPUs

- Auto-tuning problem
- Tuned IRISX provides up to 10X performance improvement when compared to Proto Library



ChatIRIS on ChatHPC

- ChatHPC:
  - Building the foundations for an AI Assisted and Productive HPC ecosystem
- A simple 3-step iterative process:
  - (1) Fine-tuning -> AI assistant (optimizers)
    - Training Data
  - (2) Testing -> Learning gaps
    - Testing Data != Training Data
    - Testing Data may contain question about capabilities non-fined-tuned
  - (3) Refinement -> More capable AI assistants
    - Expert-Supervised Refinement Data
- ChatHPC Python Library
  - Code Llama (7B), LoRA, and PyTorch

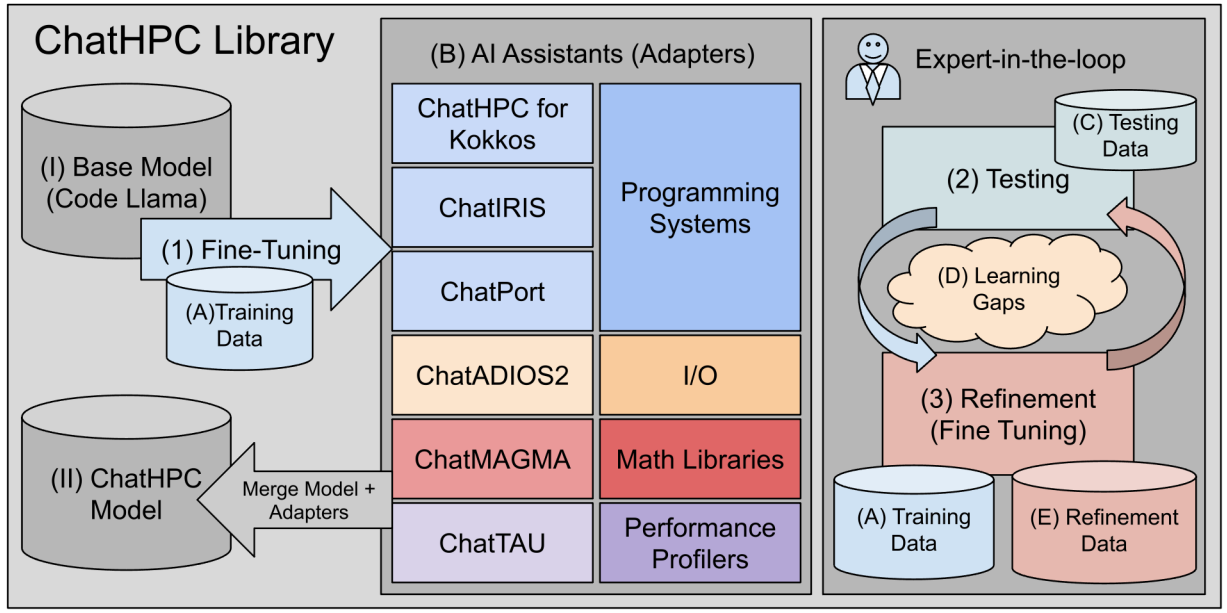
ChatHPC CLI:

```
$ chathpc train   # Finetune the assistent.
$ chathpc verify  # Verify the assistent on training set.
$ chathpc test    # Test the asssistent on unseen data.
$ chathpc run     # Interactivly run the assistent.
```

Interactive run session:

```
$ chathpc ()> /context
Context: Introduction to Kokkos
$ chathpc (Introduction to Kokkos)> What is LayoutLeft?
LayoutLeft refers column-major layout where consecutive entries in
      the same column of a 2-D array are contiguous in memory.
```

Figure 3: Example of the CLI for the ChatHPC Python library.



ChatIRIS: X to IRIS

Finetuning CodeLlama for IRIS task code

- Providing question, context and Answer

Translate the next CUDA code to IRIS:

```
alpha = 2.0;
X = (float *) malloc(SIZE * sizeof(float));
Y = (float *) malloc(SIZE * sizeof(float));
Z = (float *) malloc(SIZE * sizeof(float));
for (int i = 0; i < SIZE; i++) { X[i] = i; Y[i] = i;}
cudaMalloc((void**)&d_x, SIZE * sizeof(float));
cudaMalloc((void**)&d_y, SIZE * sizeof(float));
cudaMalloc((void**)&d_z, SIZE * sizeof(float));
// Initialization of arrays
cudaMemcpy(d_x, x, SIZE* sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_y, y, SIZE* sizeof(float), cudaMemcpyHostToDevice);
int threads = 512;
int blocks = floor(SIZE/threads) + 1;
saxpy<<<blocks, threads>>>(alpha, d_x, d_y, SIZE);
cudaDeviceSynchronize()
sdot<<<blocks, threads>>>(d_z, d_x, SIZE);
cudaDeviceSynchronize()
cudaMemcpy(z, d_z, SIZE * sizeof(float), cudaMemcpyDeviceToHost);
cudaFree(d_x);
cudaFree(d_y);
cudaFree(d_z);
free(X);
free(Y);
free(Z);
```

ChatIRIS output:

```
// Architecture-agnostic IRIS task code
alpha = 2.0;
iris_init(&argc, &argv, 1);
iris_mem iris_x;
x = (float *) malloc(SIZE * sizeof(float));
iris_data_mem_create(&iris_x, x, SIZE * sizeof(float));
iris_mem iris_y;
y = (float *) malloc(SIZE * sizeof(float));
iris_data_mem_create(&mem_Y, Y, SIZE * sizeof(float));
iris_mem iris_z;
z = (float *) malloc(SIZE * sizeof(float));
iris_data_mem_create(&mem_z, z, SIZE * sizeof(float));
// Initialization of arrays
iris_graph graph;
iris_graph_create(&graph);
iris_task saxpy_task;
iris_task sdot_task;
iris_task_create(&saxpy_task);
iris_task_create(&sdot_task);
void* saxpy_params[3]={&alpha, &iris_x, &iris_y};
int saxpy_params_info[3]={sizeof(A), iris_w, iris_r};
iris_task_kernel(task0, "saxpy", 1, NULL, &SIZE, NULL, 3,
    saxpy_params, saxpy_params_info);
void* sdot_params[2]={&iris_z, &iris_x};
int sdot_params_info[2]={iris_w, iris_r};
iris_task_kernel(task1, "sdot", 1, NULL, &SIZE, NULL, 2,
    sdot_params, sdot_params_info);
iris_graph_task(graph, saxpy_task, iris_default, NULL);
iris_graph_task(graph, sdot_task, iris_default, NULL);
iris_graph_submit(graph,iris_any,1);
free(X);
free(Y);
free(Z);
iris_finalize();
```

ChatIRIS: Tuning Result

Capabilities

- Portability
  - OpenMP, CUDA, HIP to IRIS task code
- HIP was not used for fine-tuning
- Provided good accuracy with a small set of Data

Model/Context	CodeLlama	ChatIRIS Initial	ChatGPT
Documentation	0.0%	**95.0%**	70.0%
OpenMP -> IRIS	0.0%	**82.5%**	10.0%
CUDA -> IRIS	0.0%	**90.0%**	50.0%
HIP -> IRIS	0.0%	**81.6%**	10.0%

Collaboration and Funding

IRIS is supported by Various DOE and DOD funding

- DOE ASCR Project Durban for AI For Science
- DOE ASCR Project MAGNET for competitive portfolio Research (Fairbanks)
- DOE ASCR Project Bluestone
- DARPA Project Cosmic
- DOD Project Brisbane

Conclusion and Potential Collaboration

- IRIS: Intelligent runtime with task-based programming model and runtime orchestration for extreme heterogeneity
- High level programming models uses IRIS as a back end which enable them for heterogenous orchestration
- IRIS opens the door for new heterogeneous architectures by providing a readily available software stack
- IRIS exposes new opportunities for auto-tuning taking heterogeneity into consideration

IRIS Runtime Documentation

Contact: monilm@ornl.gov

Thanks