



AI-assisted Performance Optimization for OpenMP GPU Programming

**Vivek Kale, Principal Member of Technical Staff
Sandia National Laboratories**



SAND2025-7827



Motivating Example Code: MPI-only

Needs: 1. Scientific Discovery 2. Engineering Innovation 3. service for Industry, AI/ML

stencil.cpp

```
#include <mpi.h>
int main(int argc, char* argv[])
{
    MPI_Init(&argc, &argv); // assume a weak scaling where each process gets n+2 elements
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    double* u = malloc(sizeof(double)*n+2);
    double* unew = malloc(sizeof(double)*n+2);
    int tstep = 0;
    while(tstep < 1000){
        if(myrank != 0) { MPI_Irecv(leftBoundary, ...); MPI_Isend(leftBoundary, ...);}
        if (myrank != (numprocs - 1)){ MPI_Irecv(rightBoundary); MPI_Isend(rightBoundary);}
        MPI_Waitall(...); u[0] = leftBoundary; u[n+1] = rightBoundary;

        for (i = 0; i < n+2; i++)
            unew[i] = (u[i-1] + u[i] + u[i+1])/3.0;

        MPI_Allreduce(err, MAX);
        tstep++;
    }

    MPI_Finalize();
}
```

Computational
operation

Local operation

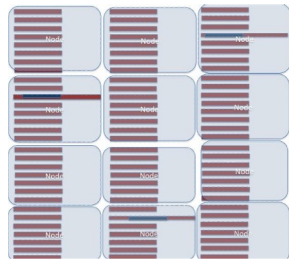
MPI communication operation

```
CC stencil.cpp -lmpich; mpirun -n 4 ./a.out 100 1000;
```

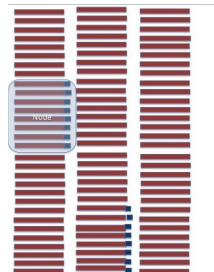
Platform: supercomputer, cloud

Slow OpenMP Prohibits MPI Scalability!

Time



Noise delays every timestep
on some node^{1,2}



Performance improves if we can
perfectly redistribute work within
each node³

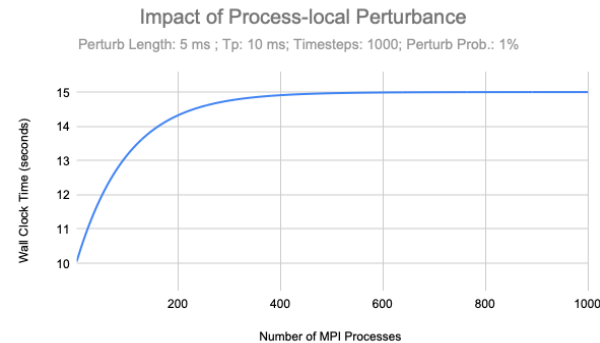
Example: 10 ms timestep, 10 of 1000 timesteps delayed by 5 ms

1 node

- Noise delay: 50 ms
- Time with noise: 10.05 seconds → 0.5% slower.

1000 nodes

- Noise delay: $1000 \times (1 - (1 - 0.01^{1000})) \times 5 \text{ ms} = 4449 \text{ ms}$
- Time with noise: 14.449 seconds → 50% slower!

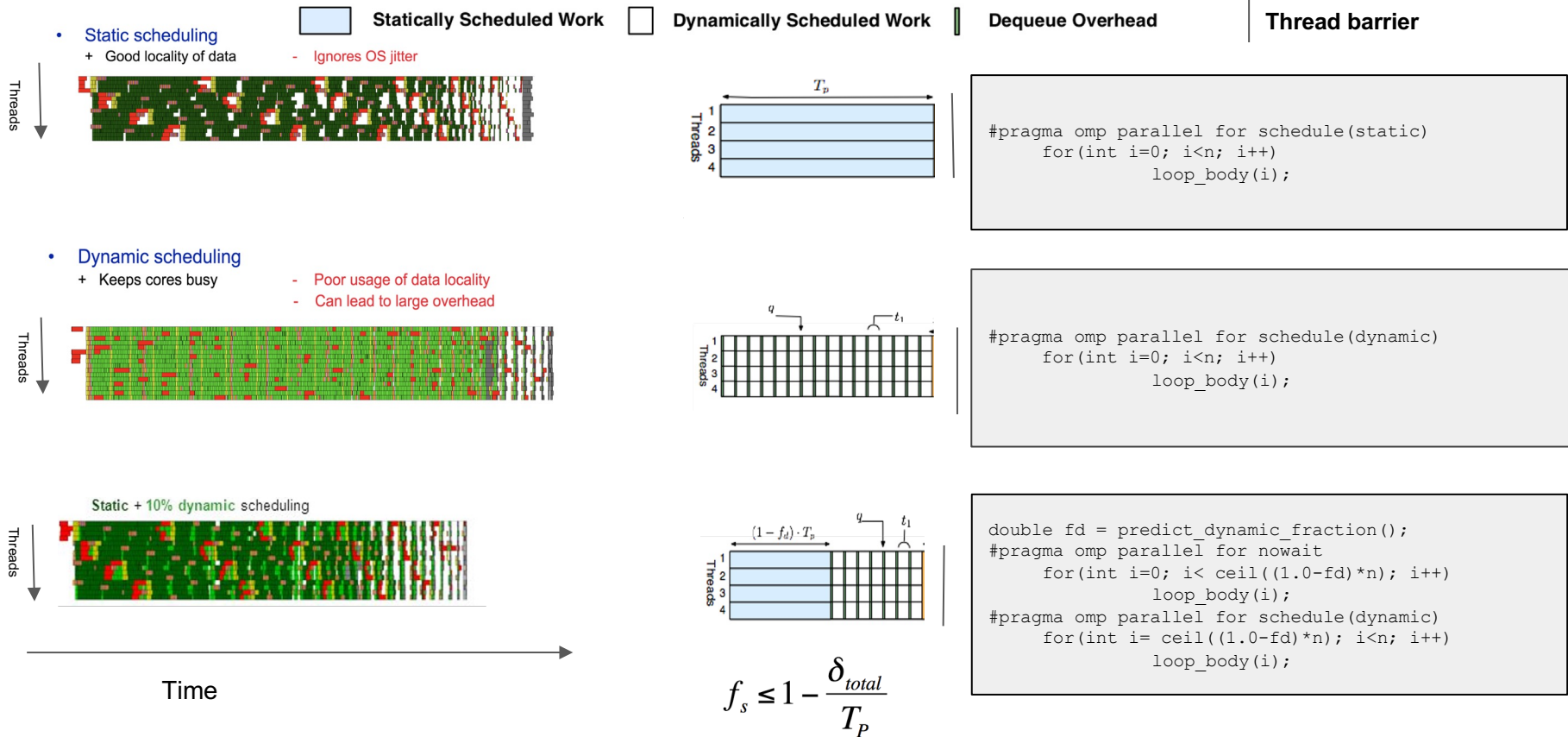


- Noise Amplification problem → Open CS problem: Wait-free consensus

[1] Torsten Hoefler and Timo Schneider and Andrew Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10), November 2010.



[2] Vivek Kale. [Low-overhead Scheduling to Improve Performance of Scientific Applications](#). Dissertation, May 2015. University of Illinois at Urbana-Champaign.

AI/ML to Find Right OpenMP Strategy



- More nodes means higher average delta
- More nodes means best-performing f_s lower.

AI/ML guided OpenMP Performance Tuning

1. **Generate:** Identify new OpenMP algorithmic strategy's via defining control points 
2. **Tune:** Tune each new strategy's control points 
3. **Search:** find best-performing algorithmic strategy with best-performing parameters

For the above to work, need:

1: Prototyping and experimentation

2: OpenMP features

3: Software tools

[4], Daejin Jo and Taehwan Kwon and Eun-Sol Kim. **Token Manipulation Generative Adversarial Network for Text Generation**. <https://arxiv.org/abs/2005.02794>

[5] X. Wu, M. Kruse, P. Balaprakash, H. Finkel, P. Hovland, V. Taylor, and M. Hall, "Autotuning PolyBench benchmarks with LLVM Clang/Polly loop optimization pragmas using Bayesian optimization (extended version)," *Concurrency and Computation. Practice and Experience*, Volume 34, Issue 20, 2022. ISSN 1532-0626 DOI: [10.1002/cpe.6683](https://doi.org/10.1002/cpe.6683)

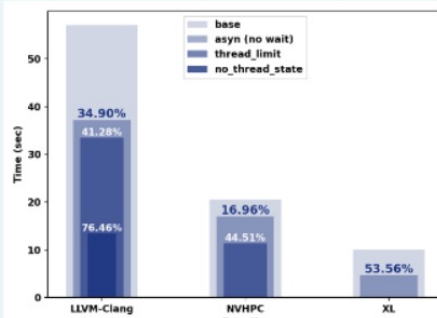


1: Control Points for OpenMP Tuning



CPU-GPU Multi-parameter Tuning

- Different Programming Models
- Tune: {host-device coordination}



Stepwise tuning of asynchronous version of OpenMP offload program

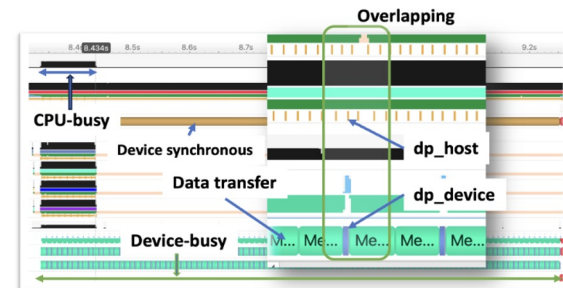


(A) sync. clang/LLVM .

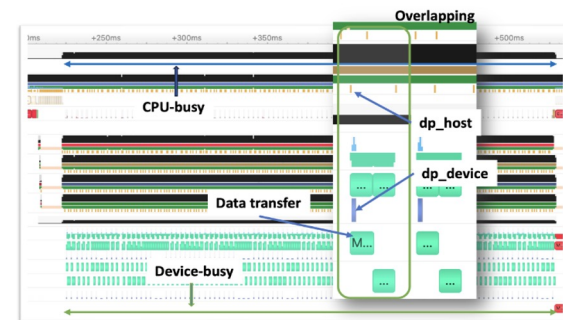


(B) Asynchronous clang/LLVM.

- OpenMP+CUDA
- Tune: {intra-CPU and Intra-GPU} x {host-device coordination} x {host-device load balancing}



(a) Original with 1 CUDA Stream.



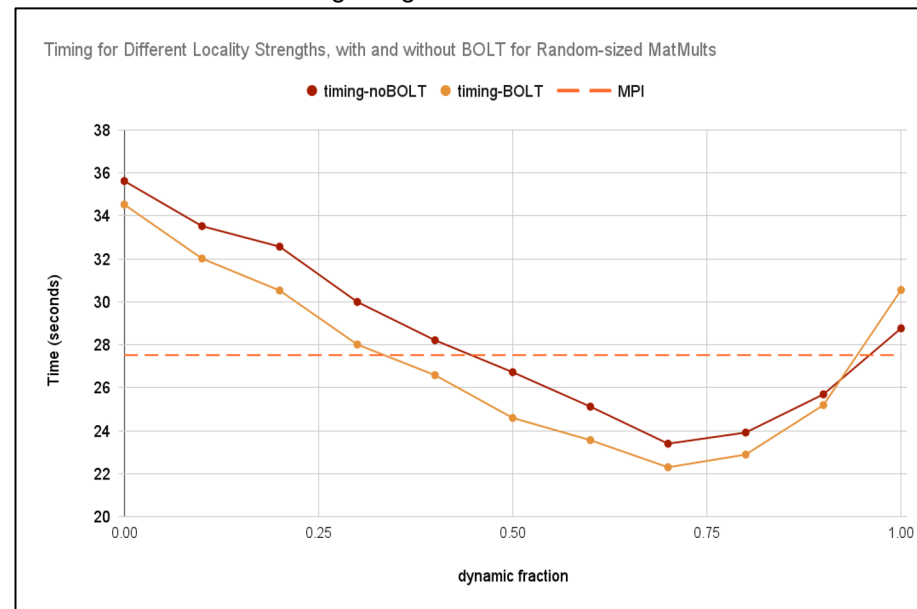
(b) Using 3 CUDA Streams.

[6] Mathialakan Thavappiragasam and Vivek Kale. **OpenMP's Asynchronous Offloading for All-pairs Shortest Path Graph Algorithms on GPUs**. IEEE/ACM HiPar 22 at SC '22. November 17, 2022. Dallas, Texas, USA.

[7] Mathialakan Thavappiragasam and Vivek Kale. **CPU-GPU Performance Tuning for Improving Performance of Modern Scientific Applications on Exascale Supercomputers**. IEEE's International Conference on High-Performance Computing (HiPC) 2023. Goa, India. December 18-21, 2023.

Full Node Multi-Objective Multi-Parameter Tuning

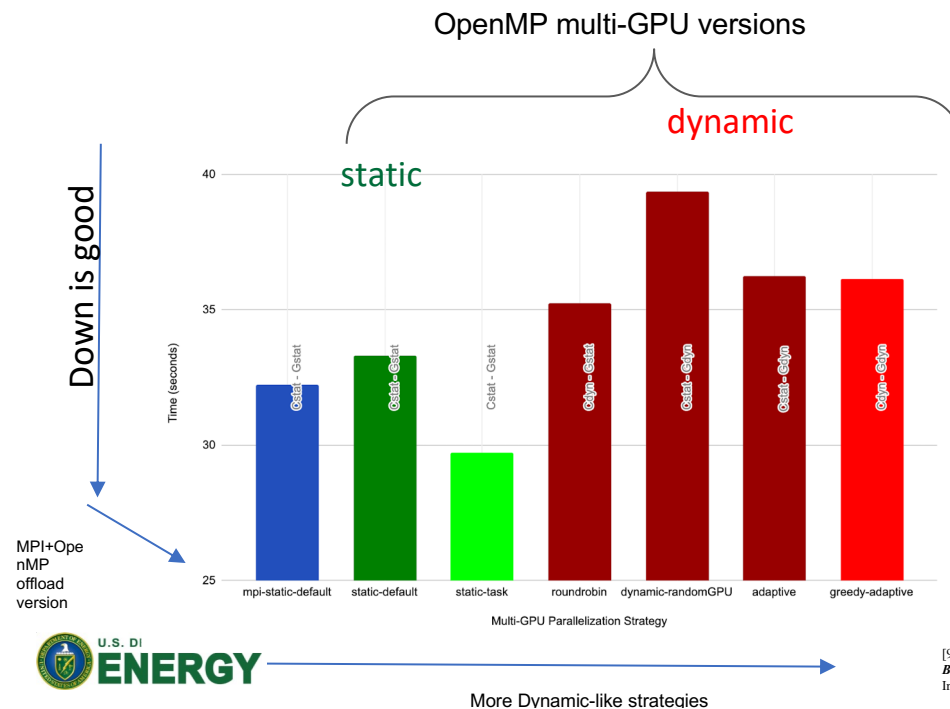
- Randomized Mat Mul Summit
- GitHub for task-to-GPU scheduling prototype: github.com/SOLLVE/openmp-rts
- Experiments are done on node of Summit using clang15



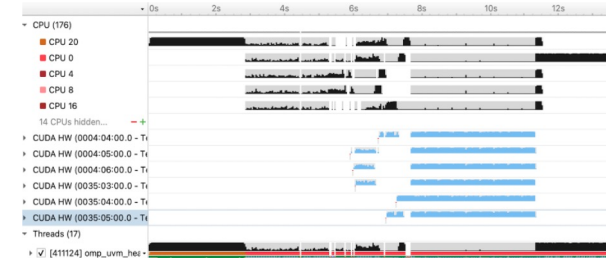
→ Prototype library for the LLVM OpenMP runtime that supports OpenMP task-to-multiGPU scheduling improves performance over OpenMP static by 43.6% and MPI version by 16.8% when affinity scheduling through low-overhead static/dynamic scheduling.

Multi-objective Multi-parameter Tuning: Heat2D

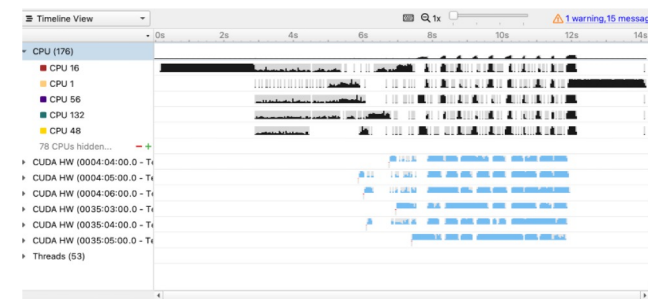
- **Application:** Stencil heat2D with problem size 32768 x 32768, 100 timesteps, Grain size 4
- **Platform:** Spectrum MPI, LLVM OpenMP, use '#pragma omp requires unified_shared_memory', Perlmutter
- **Runtime params:** 1 OpenMP thread per core. For OpenMP multi-GPU versions, used 2 MPI processes per node. cpu to gpu binding.



Static-task



RandomGPU



[9] Mathialakan Thavappiragasam, Vivek Kale, Oscar Hernandez and Ada Sedova. *Addressing Load Imbalance in Bioinformatics and Biomedical Applications: Efficient Scheduling across Multiple GPUs*. In Proceedings of 12th International Workshop on High Performance Bioinformatics and Biomedicine. December 9th, 2021. Houston, Texas



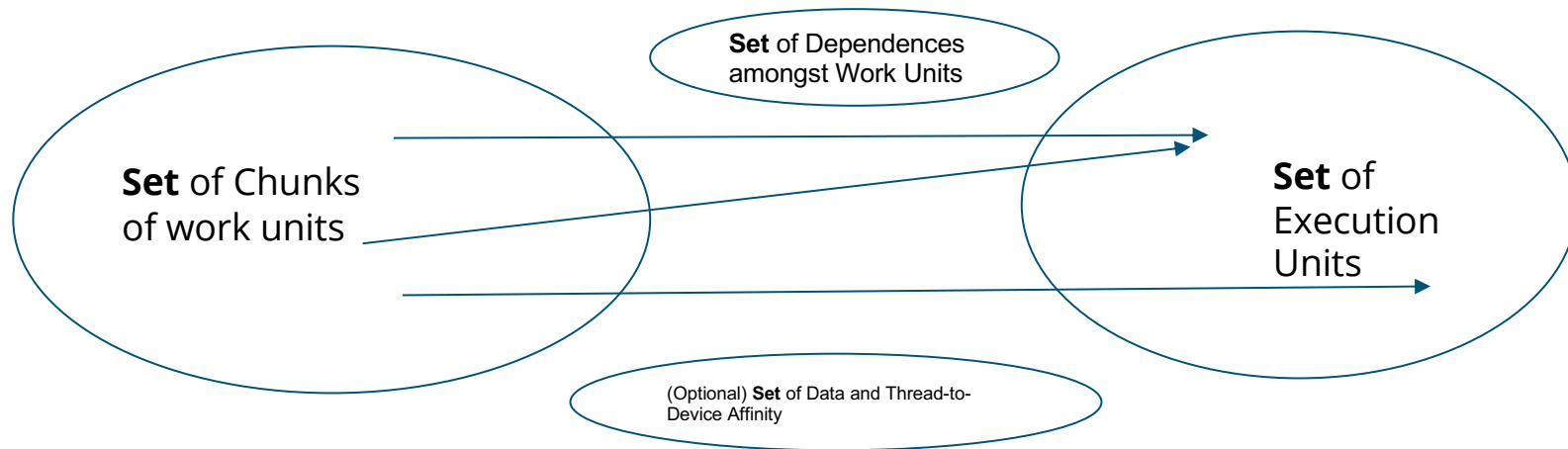
2: OpenMP Features for Enabling AI/ML Guided Tuning



OpenMP Set Object: A Way to More Easily Use AI/ML for OpenMP

- Augment OpenMP to be higher-level and consider node-level heterogeneity
- Can consider literature on parallel algorithm design¹⁰
- Define sets of entities to describe parallelization of computation in OpenMP:
 1. A **set** of (OpenMP) work units is distributed across
 2. a **set** of (OpenMP) devices/threads in way that is
 - constrained by a **set** of dependencies amongst work units and
 - aims to satisfy a **set** of OpenMP associations between work units and/or devices/threads.¹¹

```
#pragma omp set(setName) [omp_list_t_kind: {listelem_id_1, listelem_id_2, listelem_id_n}]
```



```
#pragma omp set(myChunkSet) [chunk: {{0, 3}, {4, 7}, {8, 11}, {12, 15}}]
```

```
#pragma omp set(myThreadSet) [thread: {1, 5, 2, 3}]
```

[10] https://relate.cs.illinois.edu/course/cs554-f23/f/slides/slides_02.pdf

[11] Vivek Kale and Thomas R. W. Scogland. [OpenMP Sets for Sustainable HPC Software Technology](#). Arxiv. Technical Report. September 8, 2023.



Putting it Together: Using Sets in OpenMP for Heterogeneous Programs

```
#pragma omp parallel num_threads()
#pragma omp single
{
    #pragma omp target spread(levelNum) teams distribute parallel for \
    set(devices:<[device_list]>,<levelNum>) set(chunks:<chunk_list>) spread_schedule(<levelNum>,<strategy>)
    map(close: to: A[omp_spread_start:omp_spread_size] ) \
    map(from: B[omp_spread_start:omp_spread_size])
        depend(in:A[omp_spread_start:omp_spread_size]) nowait
        for (i = 0; i < n; i++)
            doWork(i);
}
```

- Parallelism
- Work Sharing
- Loop Scheduling / Tasking
- Data dependences
- Communication
- Affinity

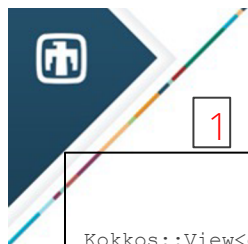
- **spread_schedule**: sets distribution **strategy** and **chunk size**; has all schedules from OpenMP in **schedule** clause of **parallel for** and can also take in set of chunk-to-device assignments, which is a user-defined schedule.

[12] Vivek Kale, Christian Iwainsky, Michael Klemm, Jonas H. Muller Kondorfer and Florina Ciorba. *Toward a Standard Interface for User-defined Scheduling in OpenMP*. Fifteenth International Workshop on OpenMP. September 2019. Auckland, New Zealand.

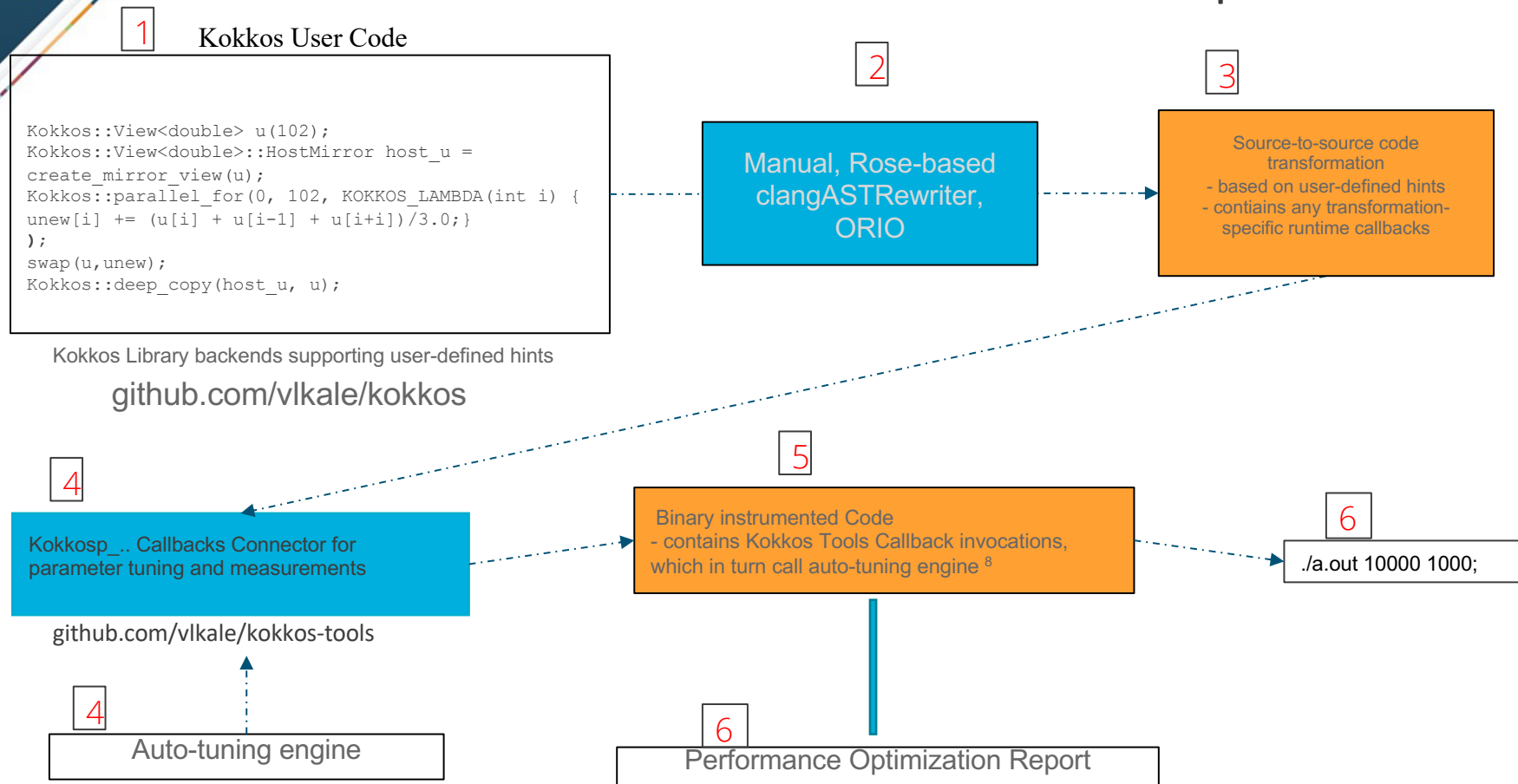
[13] Raul Torres, Vivek Kale, Abid Malik, Tom Scogland, Roger Ferrer and Barbara M. Chapman. *Support in OpenMP for Multi-GPU Parallelism*. The International Conference for High Performance Computing Networking, Storage, and Analysis. Extended Abstract and Poster. November 19, 2021. St. Louis, Missouri, USA.



3: Automation Tools for AI/ML Guided OpenMP Performance Tuning



From Manual to Automated Performance Improvement



[14] David Boehme, Kevin Huck, Shravan Kale, Vanessa Surjadidjaja, Vivek Kale. Sophisticated Tool Support to Aid Performance Portable Programming. SC 23. Extended Abstract and Poster. November 14, 2023. Denver, Colorado, USA.



LLM-based GPU Source Code Translation with CodeRosetta

coderosetta.com

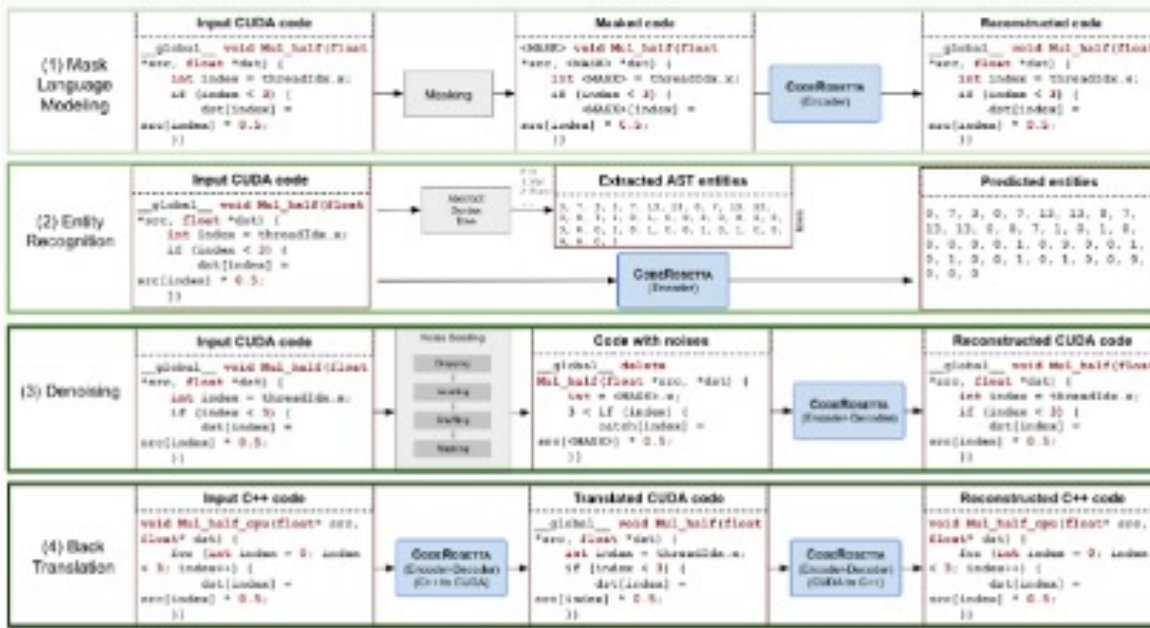


Figure 1: Pretraining Objectives
C++ to CUDA

Model	BLEU	CodeBLEU	Compilation Accuracy (%)
GPT4	46.98	64.45	96.1
Gemini-Ultra	57.06	61.18	80
Gemini-Pro	54.82	64.20	75.5
TransCoder	72.21	71.03	83.8
BabelTower	74.00	77.12	92.8
CODEROSETTA	76.90	78.84	98.85

- Encoder-decoder, 0.8B Parameter Model
- Recent work on OpenMP → helps C++ to Kokkos

→ Small lightweight model effective for auto-parallelization

Bhattacharjee, Huck, Jannessari, Kale. *AI-assisted Tool Support for C++ HPC Applications on NVIDIA GPUs*. GTC25. March 2025.



Kokkos Runtime Auto-Tuning with APEX

- ▶ New Internal tunable parameters (4.5)
 - ▶ RangePolicy - Occupancy (with code modification) for CUDA execution space
 - ▶ MDRangePolicy - Occupancy (with code modification) for CUDA execution space
 - ▶ TeamPolicy - Occupancy (with code modification) for CUDA execution space
- ▶ Occupancy value tuned between [5:100], step size of 5
- ▶ Natural extension of PR #3379: "Experimental feature: control cuda occupancy"

"...passing `prefer(policy, DesiredOccupancy(33))` to a `parallel_for()`, `parallel_reduce()`, or `parallel_scan` will bypass the block size deduction that tries to maximize the occupancy and adjust the launch parameters (by fixing the block size and requesting shared memory) to achieve the specified occupancy. The desired occupancy is in percent."



NEW: ML-guided GPU bottleneck diagnosis³

Use Nsight metrics to identify: (1) thread divergence
(2) bank conflicts (3) uncoalesced memory accesses

→ Can reduce tuning parameter search via more profiling data, obtained through ML



1: github.com/kokkos/kokkos-tools

2: github.com/UO-OACISS/apex/

3: Under Review



Conclusion

- **Challenge:** large search space for OpenMP optimization for MPI+OpenMP
- **Opportunity:** AI/ML can prune large search space significantly
- **Approach:**
 - Training via experimentation of OpenMP strategies, identifying control points
 - Easier machine generation and tuning via OpenMP Set interface.
 - Search for best-performing OpenMP strategies via Kokkos Tools auto-tuning
- **Outcomes:**
 - 8.3x speedup via software tools appropriate control point transformations and tuning on single GPU; 1.7x speedup on multi-GPU experimentation
 - New OpenMP features and software tools for use.



Ongoing and Future Work

- More investigation of Kokkos Tools auto-tuning with OpenMP+CUDA
 - See new MDRange Tuning feature in APEX TPL at github.com/kokkos/kokkos-tools/ added July 2024
- Experiment with HPX + Kokkos and/or Charm++ + Kokkos → in progress with Kokkos-comm subproject of Kokkos
- Consider full-fledged applications for protein folding and weather simulation.
 - Early work in August 2024 on Kokkos-E3SM Scream to tune OpenMP team size parameter at runtime

Thanks!



OpenMP[®]

SC25 Booth Talk Series

openmp.org

OpenMP API specs, forum,
reference guides, and more

link.openmp.org/sc25talks OpenMP SC25 booth talk
videos and presentations