



V1-25: High-Productivity Computing in Heterogeneous Systems



Mission-Critical Computing

NSF CENTER FOR SPACE, HIGH-PERFORMANCE,
AND RESILIENT COMPUTING (SHREC)

SHREC Annual Workshop (SAW24-25)



January 14-15, 2025

Faculty: Wu Feng

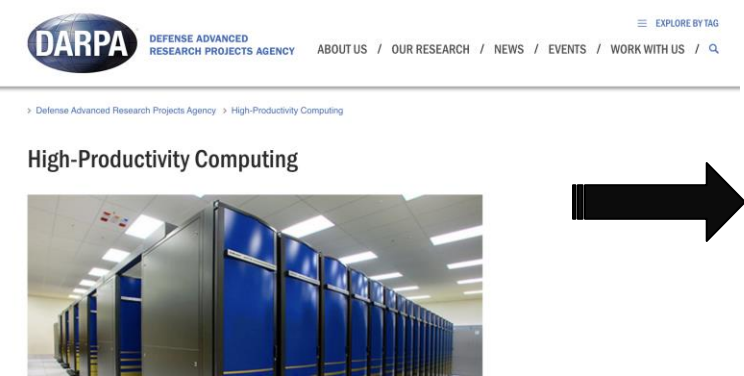
Students:

Nabayan Chaudhury, Atharva Gondhalekar,
Ritvik Prabhu, Eric Rippey, Paul Sathre, Frank Wanye

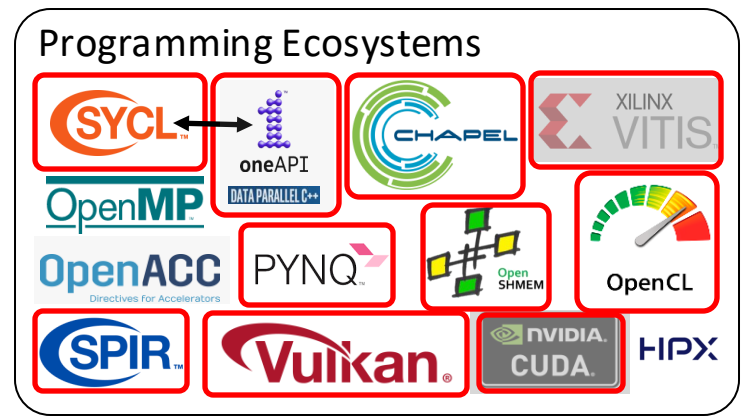
Number of requested memberships ≥ 7

Goal

- Enable **high-productivity computing** in **heterogeneous** computing systems: CPU + {CPU, GPU, FPGA, TPU, ...}
- Similar to DARPA High-Productivity Computing Systems program for homogeneous systems (e.g., Chapel, Fortress, X10) but for heterogeneous systems (e.g., Chapel, SYCL, oneAPI, OpenCL)
- Preferred Vehicle: Modern, Open Standard Languages & Runtime Systems
- Case Studies: Applications and Benchmarks, e.g., Berkeley Dwarfs → OpenDwarfs (@ VT)

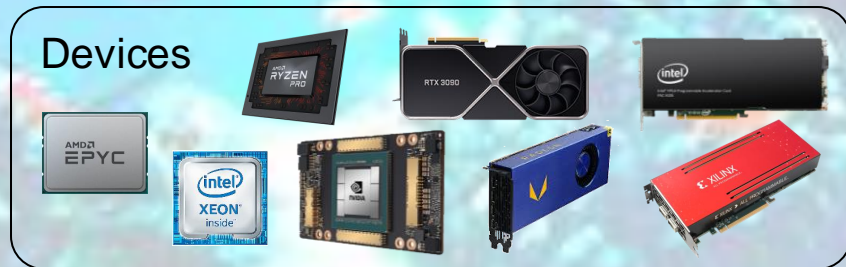
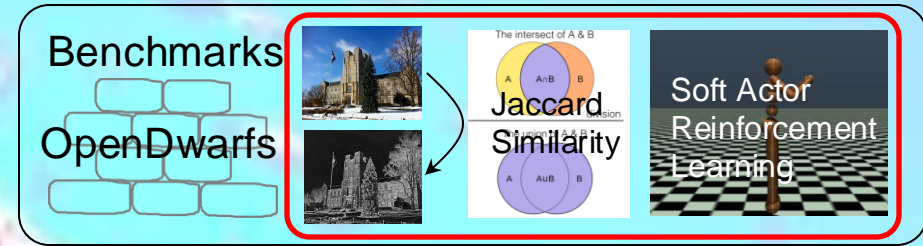


Write ONCE, run ANYWHERE!



Background & Motivation

- Extend our R&D to create and analyze an ecosystem of *high-productivity tools, environments, and benchmarks for heterogeneous computing*



- Challenges: How to *productively* ...**
 - Program an application so it runs on many platforms?
 - Evaluate a processor architecture & compare it to others?
 - Develop back-end optimizations & know that they will work well?
- } Application-dependent

Background: Performance & Productivity (V1-23)

- Sobel Filter on Intel Arria 10, AMD Alveo U250, and NVIDIA RTX 3090



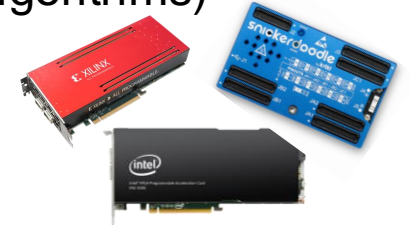
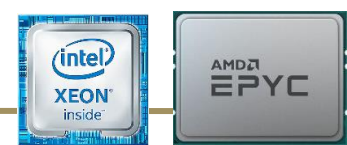
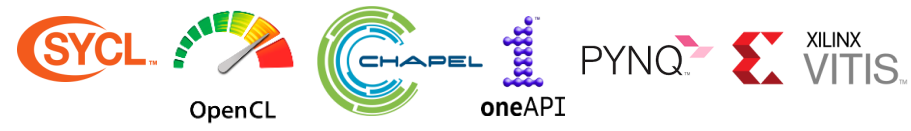
Sobel Filter on 3840 × 2160 Image	Device								
	Intel Arria 10 (FPGA)			AMD Alveo U250 (FPGA)			NVIDIA RTX 3090 (GPU)		
Language	fps	SLOC	Dev Time (hrs)	fps	SLOC	Dev Time (hrs)	fps	SLOC	Dev Time (hrs)
Verilog	132.6	429	305	Not implemented in Verilog			Not functional on GPU		
OpenCL	85.6	270	50	6.6*	275	-	141.4	254	-
oneAPI → SYCL	21.4	139	20	No support for oneAPI			133.1	135	-

fps: frames per second

OpenCL & SYCL: Write once, run anywhere

* Evaluated the **same** OpenCL kernel written for Arria 10 on U250 without any vendor-specific optimizations

- Rigorous **Performance & Productivity** Evaluation of **Representative Apps** (FFT, Jaccard similarity, biconjugate gradient stabilized method – BiCGSTAB, and graph algorithms) in **Different Languages** on **Different Devices** (CPUs, GPUs, and FPGAs)



Approach

- Realize a diverse set of **application benchmarks**
 - Regular vs irregular. Floating point vs integer. CPU- vs memory-intensive.
- Characterize the **productivity** of a heterogeneous system
 - Kernel development time (KDT) \rightarrow wall clock time
 - Source Lines Of Code (SLOC), compressed code size (CCS), and Code Convergence (CC)
- Characterize the **performance-vs-productivity** tradeoff
 - Performance Portability (Φ)
 - Performance-Productivity Product (Π)
- **Identify the best platform** and associated ecosystem for productivity, performance, or both (across many apps)
- **Enable further high-productivity research: automated co-scheduling at runtime, performance vs. precision tradeoff**

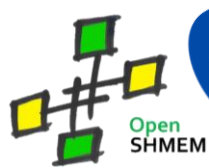


Open Source

Closed Source



Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE,
AND RESILIENT COMPUTING (SHREC)



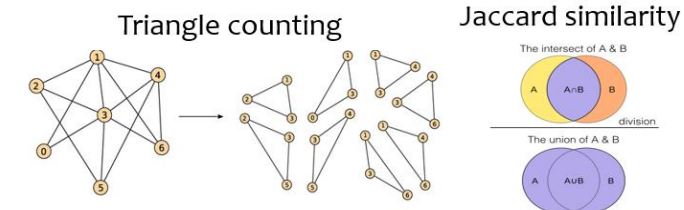
Proposed Tasks for V1-25

Memberships:
(Mandatory+Optional), e.g., (2+1)

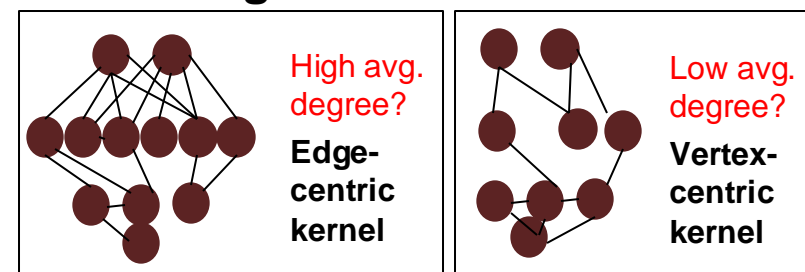
- Task 1: High-Productivity Computing on GPUs: Irregular Apps (3+4)
 - Task 1a: **Graph Algorithms**: Jaccard Similarity, Triangle Counting, ...
 - Task 1b: **Iterative Solvers** for Sparse Systems on GPUs
 - Task 1c: **Portable Kernel Pipelines** for GPU-based Edge Devices
- Task 2: High-Productivity Computing on FPGAs (1+1)
 - Deep Learning on Versal ACAP Devices (Regular & Irregular)
- Task 3: High-Productivity *Heterogeneous* Computing: CPU+GPU+FPGA (3+11)
 - Task 3a: Simultaneous Co-scheduling of Heterogeneous Devices: CPU+GPU+FPGA
 - Task 3b: Heterogeneous PGAS vs MPI+X for Large-Scale Computing
 - Task 3c: Portable Runtimes for Heterogeneous Task Graphs
 - Task 3d: Modernization of OpenDwarfs

Task 1a: Graph Algorithms: Jaccard Similarity, Δ Counting

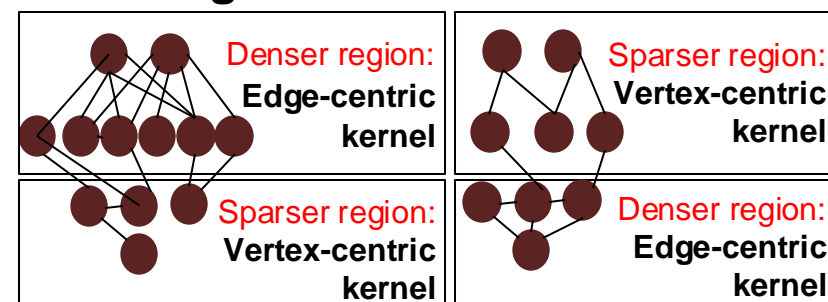
- Motivation: *Graph workloads hard to optimize on GPU*
 - Workload imbalance and irregular memory access patterns
 - Input graph-dependent and GPU architecture-dependent
- Approach
 - Target workloads: Jaccard similarity and triangle counting
 - GPU architecture-specific optimizations
 - Pattern-matching framework that predicts best set of optimizations (based on graph characterization)
- Milestones
 - Multi-dimensional exploration of performance optimization
 - ▶ Memory subsystems of GPU; parallel algorithm type (edge vs vertex-centric); and thread-launch configuration, both static and dynamic
 - Intelligent GPU kernel selection for a given graph via classification (i.e., coarse-grained) and/or regression tree (i.e., fine-grained)
 - Intelligent device (and associated ecosystem) selection for productivity, performance, or both



Coarse-grained kernel selection



Fine-grained kernel selection



Denser region
→ Compute on GPUs



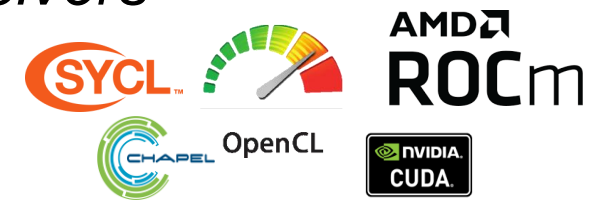
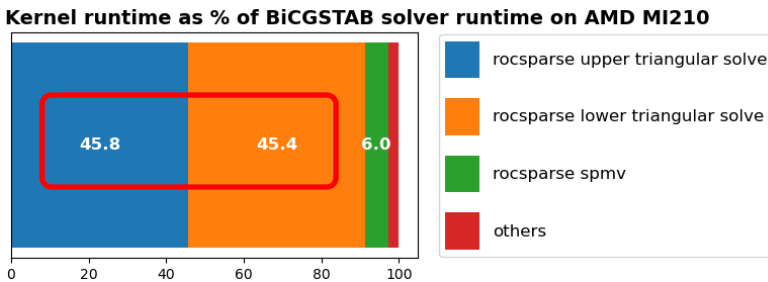
Sparser region
→ Compute on CPUs/FPGAs



Task 1b: Accelerating Iterative Solvers for Sparse Systems

- Motivation: Δ solving, a bottleneck in preconditioned iterative solvers

91% of iterative solver's runtime spent on Δ solves when running preconditioned BiCGSTAB



Ref. implementation: <https://github.com/OPM/opm-simulators/tree/master/opm/simulators/linalg/gpubridge/rocm>

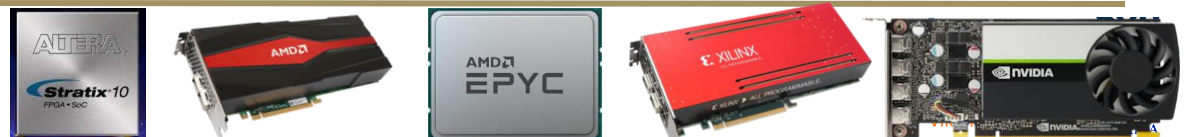
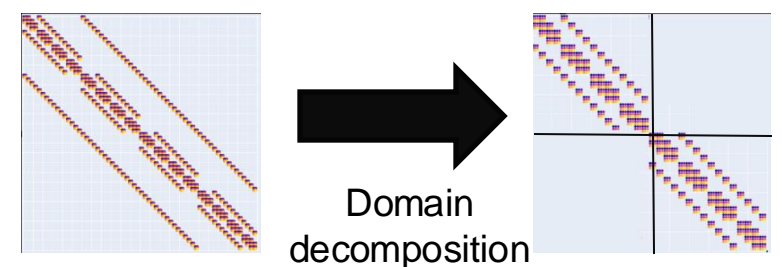
- Approach

- Domain decomposition
 - Partition matrix into subdomains & drop connections between subdomains
- Apply triangular solves *in parallel* to subdomains

$$\frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 1 & -2 & 1 \\ 0 & \dots & \dots & \dots & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-3} \\ u_{N-2} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-3} \\ f_{N-2} \end{pmatrix}$$

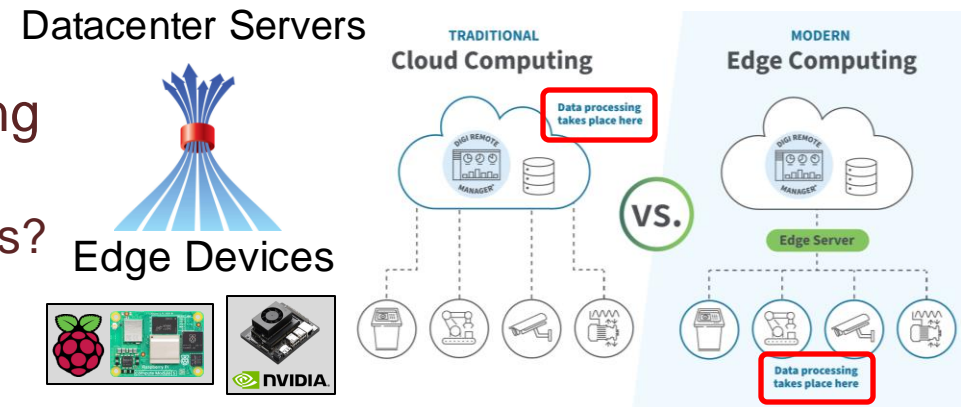
- Milestones

- Implement optimizations for triangular solves for matrices with multiple independent subdomains
- Evaluate impact of domain decomposition on performance and iteration count of the solver
- Characterize performance-vs-productivity tradeoff
 - Performance Portability (Φ)
 - Performance-Productivity Product (Π)





Task 1c: Portable Kernel Pipelines for GPU Edge Devices

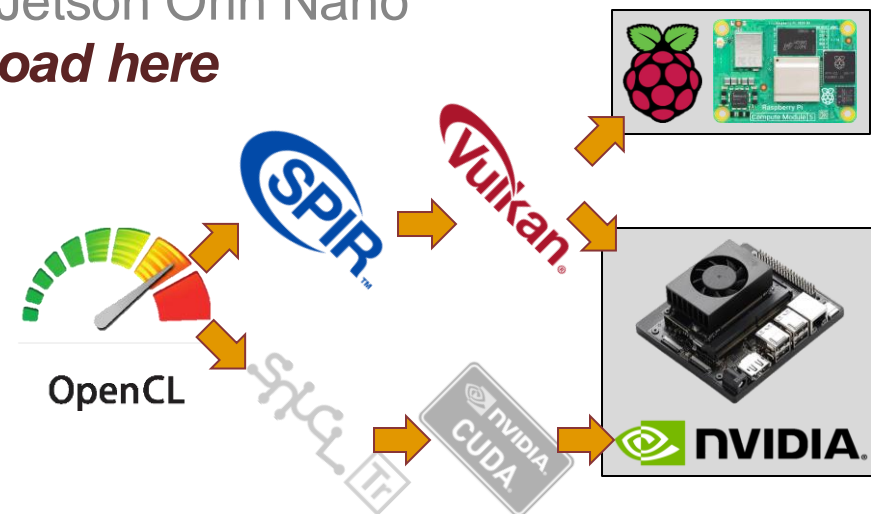
- Motivation: Edge Computing with GPUs
 - Bottleneck: Data xfer from edge to datacenter for processing
 - Productivity: Edge devices w/ varying programming APIs
 - ▶ Route from portable HPC languages to low-power edge devices?



- Approach
 - Move data processing **from datacenter to the edge**
 - Leverage portable & open-source HPC standards and toolchains to compute on edge GPU(s)
 - Evaluate productivity, performance, perf./prod. (II), power, and perf./power
 - Platforms: Raspberry Pi Compute Module (CM) 5 and Nvidia Jetson Orin Nano
 - Workloads: Mixed-radix FFT, FFT convolution, or **your workload here**

Milestones

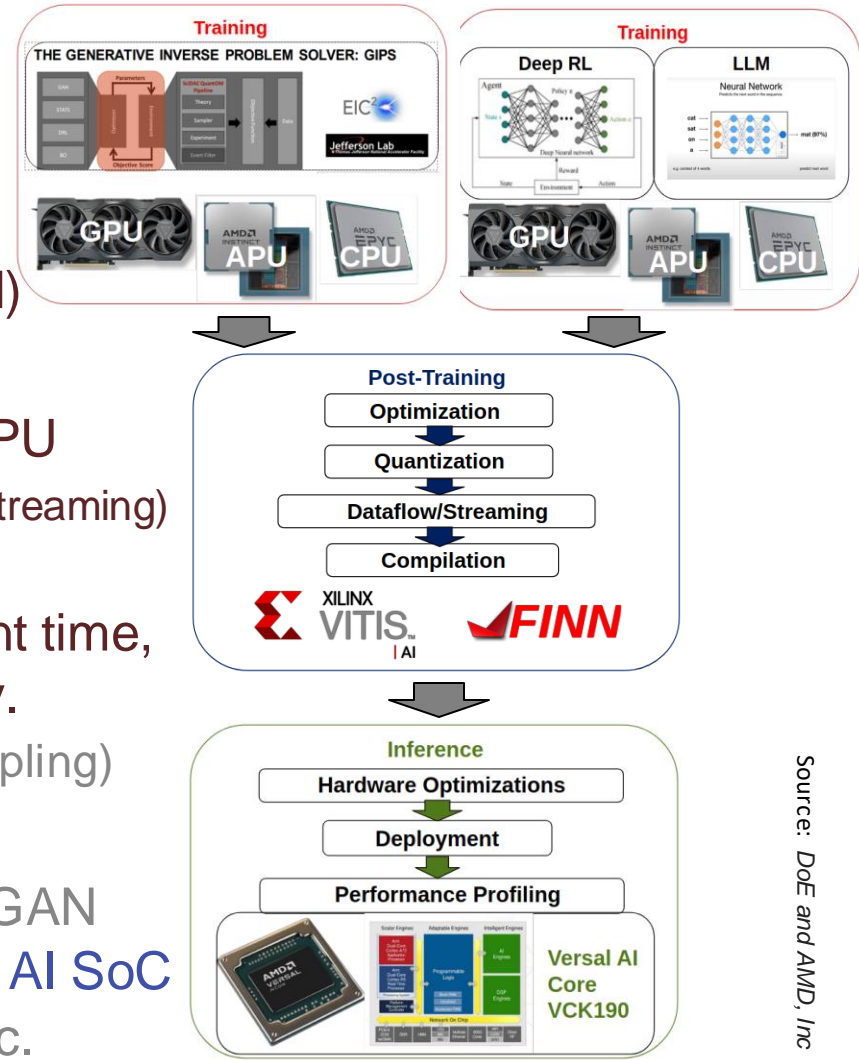
1. Raspberry Pi CM5 via OpenCL C → SPIR-V → Vulkan
2. Nvidia Jetson Orin Nano via OpenCL C → SPIR-V → Vulkan
3. SYCL C++ on either device (via Syllan or similar) 
4. Nvidia Jetson Orin Nano via OpenCL C → SnuCL-Tr → CUDA 



Task 2: High-Productivity Computing on FPGAs

Deep Learning on Versal ACAP Devices

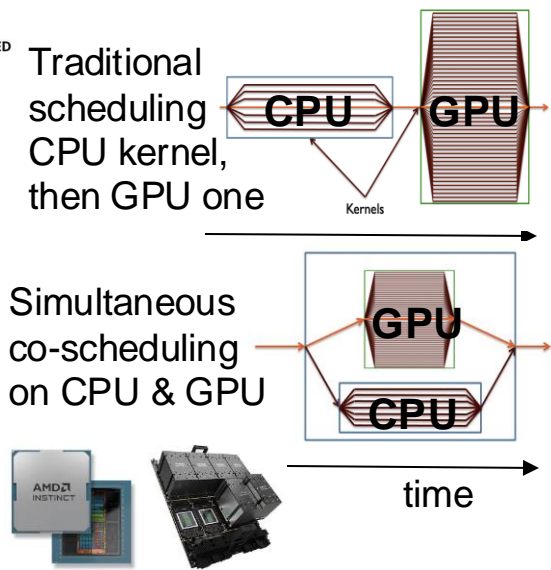
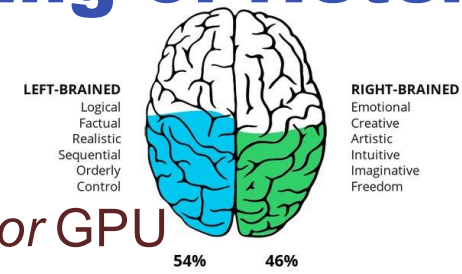
- Motivation: HPC for Deep Learning (DL)
 - Need fast training & inference, e.g., large language model (LLM), deep reinforcement learning (DRL), generative adversarial net (GAN)
- Approach
 - Training: LLMs & DRL (optionally, GANs) on GPU or fused CPU+GPU
 - Post-Training: VITIS AI & FINN for network optimization (quantization/streaming)
 - Inference: GPUs → Versal AI SoCs
 - Metrics: Performance/productivity profiling, i.e., kernel development time, execution time, source lines of code (SLOC), and inference latency.
 - Workloads: Regular (synthetic generation) & irregular (stochastic sampling)
- Milestones
 1. Training & post-training: DRL network (SAC, DDPG), LLM (Llama), GAN
 2. Inference: Vivado+VITIS hardware optimization on GPU & Versal AI SoC
 3. Performance/productivity profiling: KDT, execution time, SLOC, etc.



Source: DoE and AMD, Inc

Task 3a: Simultaneous Co-scheduling of Heterogeneity (CPU+GPU+FPGA) – Details in Appendix

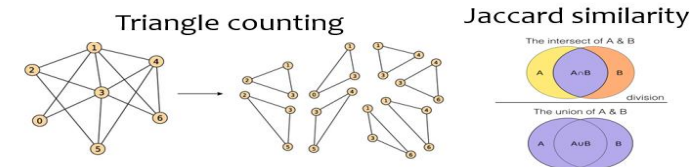
- Motivation: < 10% use of peak compute capability
 - Today: Inefficient use of silicon computing, i.e., *either CPU or GPU*
 - ▶ Physiologically, left & right brain used *simultaneously*
 - ▶ Silicon-wise, should use CPU & GPU brain *simultaneously* (even FPGA)
 - Past Work? Our *CoreTSAR: Core Task-Size Adapting Runtime*, which co-schedules regular apps on CPU+GPU *simultaneously* via Accel. OpenMP



- Approach
 - CoreTSAR++: Generalize co-scheduling for *multi-heterogeneity (CPU+GPU+FPGA)* and across regular & irregular workloads via **SYCL**.
 - Metrics: productivity, performance, perf./prod. (Π) vs. single device



- Milestones: CoreTSAR++ Exploration **SYCL**
 1. Identify & implement appropriate irregular apps to co-schedule
 2. Manually implement & evaluate co-scheduled irregular apps
 3. Automate co-scheduling on heterogeneous system (CPU+GPU+ ...)
 4. Investigate simultaneous co-scheduling using **CHAPEL** via partitioning & multi-device cobegin {...}



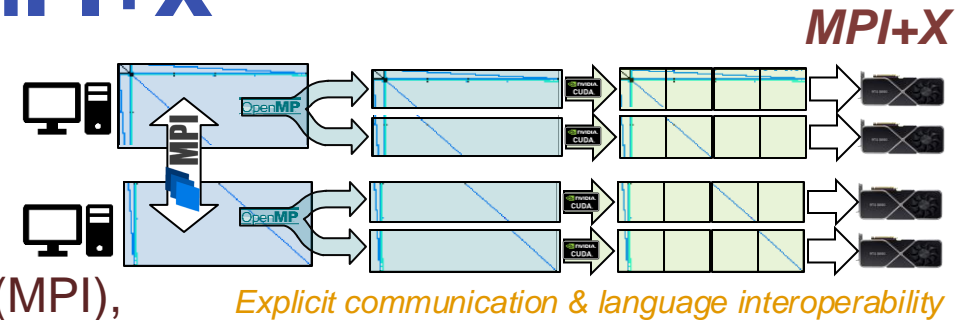
Sparse linear solver \rightarrow Biconjugate gradient stabilized

$$\frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 1 & -2 & 1 \\ 0 & \dots & \dots & \dots & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-3} \\ u_{N-2} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-3} \\ f_{N-2} \end{pmatrix}$$

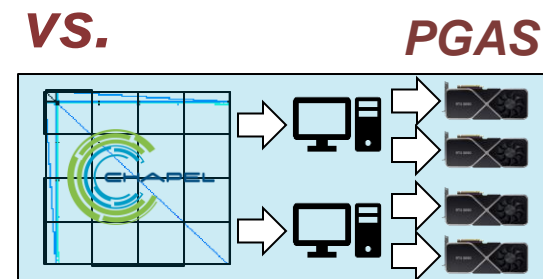
Task 3b: Heterogeneous PGAS vs MPI+X

Motivation


- MPI+X dominates *distributed heterogeneous* computing, where $X \in \{CUDA, HIP, SYCL, OpenCL, OpenMP, \dots\}$
 - Issue: Domain scientists must know low-level communication (MPI), low-level device programming (X), and interoperability between the two!



- Alternative? Partitioned Global Address Spaces (PGAS): CPU → GPU
 - Node/device address spaces are *logically joined* and *implicitly communicate*



Approach

- Transform CPU-based PGAS to heterogeneous PGAS, i.e., 
- Analyze MPI+X vs PGAS on distributed multi-GPUs w/ real-world data
 - Metrics: productivity (SLOC, CCS) vs. performance (runtime) vs. perf./prod. (Π)



- Port Chapel app(s) to other PGAS → one of {OpenSHMEM, HPX, ...}
- Identify limitations to existing heterogeneous PGAS approaches
- Propose and develop workarounds at scale, i.e., hybrid PGAS+X

Milestones

- Chapel vs MPI+CUDA+OpenMP via partitioned Jaccard similarity
- Partitioned Jaccard similarity in OpenSHMEM or HPX
- Interoperating accelerator-aware comms: {NVSHMEM, GPUDirect, etc.}
- Δ solvers / other linear algebra in any PGAS model vs MPI+X
- Your workload** here in **PGAS-of-choice** vs MPI+X

Task 3c: Portable Runtimes for Heterogeneous Task Graphs

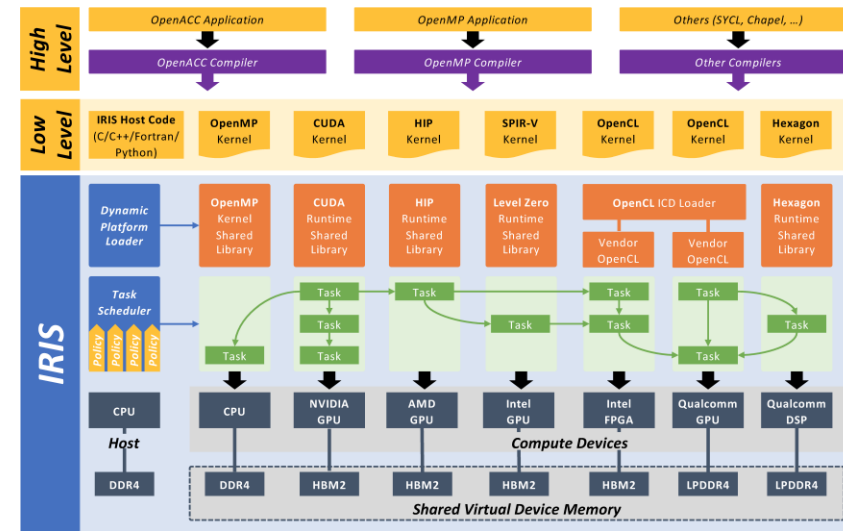
Motivation

- Performance: HPC needs device- & system-aware mapping of kernels, communication, and I/O to hardware
- Infrastructure: Hardware *migration (translate, remap, retune)* is a significant *cost*, which slows mission progress

Approach



- Portability: Leverage portable languages to lower **translation** cost
- Performance: **Remap** and **retune** for new hardware. Alas, \$\$\$.
- Solution: Intelligent heterogeneous tasking system
- Given a portable representation, model & predict tradeoffs in mapping kernels to different hardware in the system
- Case Study
 - Implement SHREC-related app(s) in OpenMP, OpenARC, or emerging UniSYCL compiler
 - Leverage and evaluate the *IRIS portable heterogeneous tasking system's* ability to achieve high performance



Milestones

- Migrate a SHREC workload to IRIS runtime & analyze perf./prod. (II) (0.5)
- Evaluate perf./prod. (II) on typical hetero HPC (CPU+GPU, *homogeneous across nodes*) (1)
- Evaluate perf./prod. (II) on *multi-hetero* HPC (CPU+X, *where X differs between nodes*) (1)
- Evaluate perf./prod. (II) with edge+centralized hybrid workloads w/ hetero platforms (1.5)

Task 3d: Modernization of OpenDwarfs

Goal: "Write once, run anywhere"

- Motivation: *Write Once, Run Anywhere* Benchmark Suite

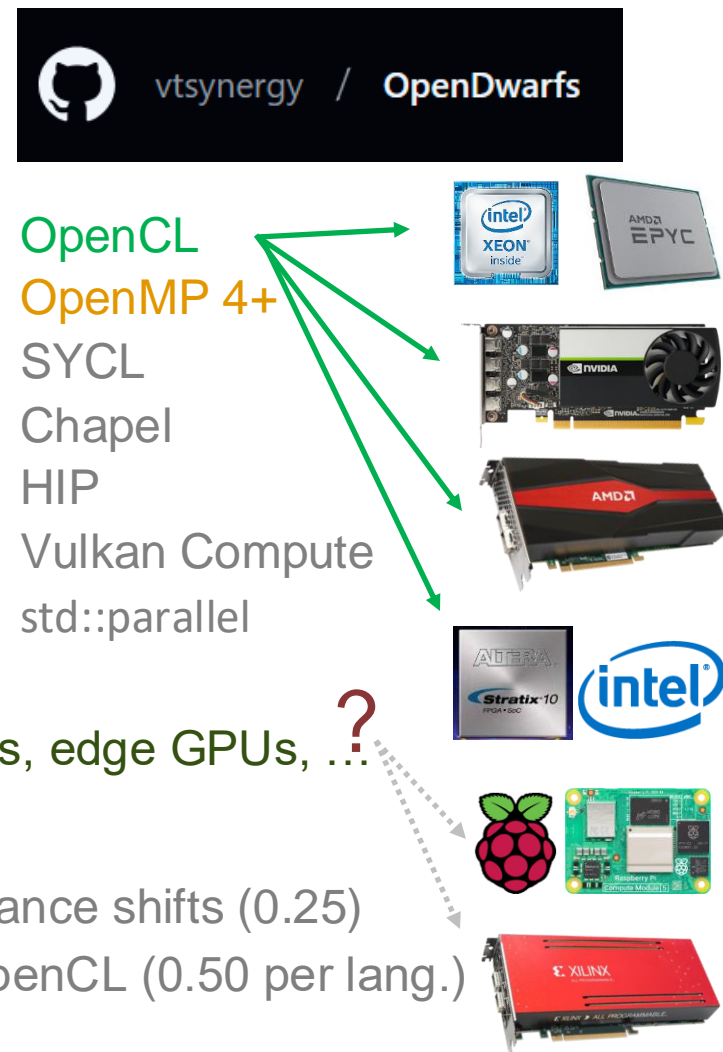
- OpenDwarfs: NSF CHREC project to create a portable suite of 13 *parallel computational idioms* (Berkeley Dwarfs) via OpenCL
 - Portable to CPUs, GPUs, APUs, and eventually Intel/Altera FPGAs
 - Now *many more paths* to portable, heterogeneous computing
- To bridge *programming gap* between portable languages and high-level library-driven heterogeneity, need examples of how to write *novel* kernels

- Approach

- Showcase idiomatic parallel codes using modern portable languages
- Modernize for new classes of devices and compute modalities
 - Unified memory, PGAS, tensor cores, HBM, hybrid co-scheduling, DSPs, edge GPUs, ...

- Milestones

- Update OpenCL OpenDwarfs for modern devices → characterize performance shifts (0.25)
- Implement OpenDwarfs in new languages & analyze perf./prod. (Π) vs. OpenCL (0.50 per lang.)
- Design *partitionable/distributable* variants of existing dwarfs (1+)



Milestones, Deliverables, and Budget

Major Milestones (Tasks: T1-T3)

- T1: High-Productivity Computing *on GPUs: Irregular Apps*
- T2: High-Productivity Computing *on FPGAs*
- T3: High-Productivity Heterogeneous Computing: *CPU+GPU+FPGA*



Deliverables

- Software prototypes and artifacts (typically delivered via github)
- Mid-year and end-of-year reports at SHREC workshops. Optionally, more frequently.
- 2-3 publications at top-tier conference venues or journals



Recommended Budget

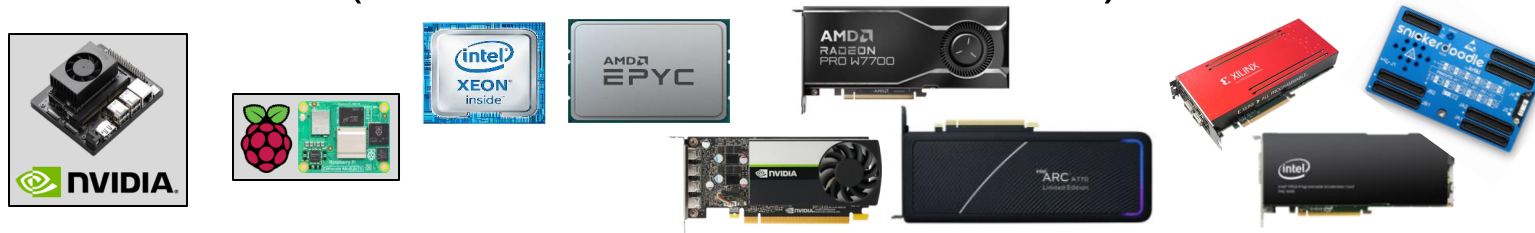
- Minimum: 7 memberships (350 votes)
- Maximum: 23 memberships (1150 votes)



Conclusion

- Enable **high-productivity computing** in **heterogeneous** computing systems: CPU + {CPU, GPU, FPGA, TPU, ...} via open standards: OpenCL, SYCL, Chapel, and emerging programming models
- Evaluate **performance & productivity of representative apps** (OpenDwarfs, FFT, Jaccard similarity, biconjugate gradient stabilized method – BiCGSTAB, and graph algorithms) **on different devices** (CPUs, GPUs, and FPGAs)

Member Benefits



- Direct influence over processors & frameworks studied and apps & datasets used
- Direct benefit from new methods, tools, datasets, codes, models, and insights created as well as new metrics of evaluation
- Direct insights from R&D and analysis