National Science Foundation's Industry/University Cooperative Research (I/UCRC) Program

V1-26: Performance, Power, and Precision, of Heterogeneous Systems

Mission-Critical Computing

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

SHREC Annual Workshop (SAW25-26)

University of Pittsburgh

BYU BRIGHAM YOUNG UNIVERSITY

VIRGINIA TECH

UF UNIVERSITY of FLORIDA

January 13-14, 2026

Faculty: Wu Feng and Krish Sundararajah

Students and Post-Students:

Nabayan Chaudhury, Atharva Gondhalekar, Poorna Gunathilaka, Pratheek Prakash Shetty, Ritvik Prabhu, Eric Rippey, Paul Sathre

Number of requested memberships ≥ 6

V1

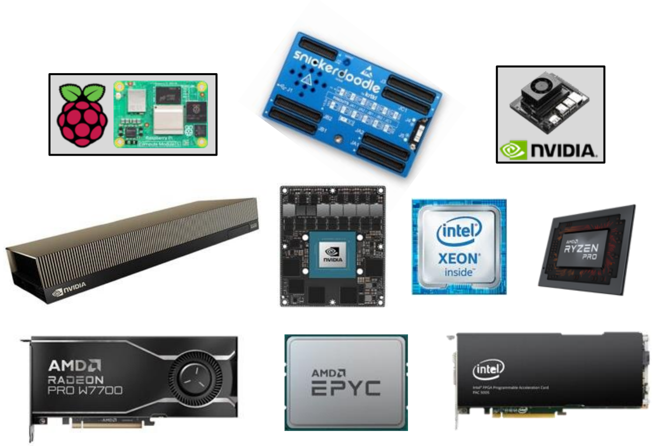
Goal: Productivity → Performance, Power, Precision

- Tuning & optimization of performance, power, and precision (manual → automatic) for productivity in ***heterogeneous*** computing systems: CPU + {CPU, GPU, TPU, … }
  - Akin to DARPA HPCS program for homogeneous systems (e.g., Chapel, Fortress, X10) but for heterogeneous systems (e.g., Chapel, oneAPI → SYCL, OpenSHMEM)
  - Preferred Vehicle: Modern, Open Standard Languages & Runtimes → write once, run anywhere
  - Metrics of Evaluation: Performance, Power / Energy Efficiency, and Precision (e.g., int vs. SP vs. DP)

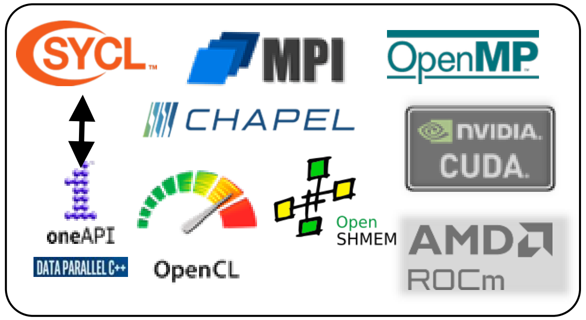
DARPA DEFENSE ADVANCED RESEARCH PROJECTS AGENCY ABOUT US / OUR RESEARCH / NE

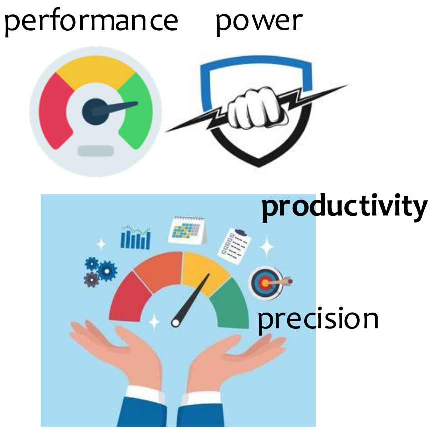
› Defense Advanced Research Projects Agency › High-Productivity Computing

High-Productivity Computing



Programming Ecosystems



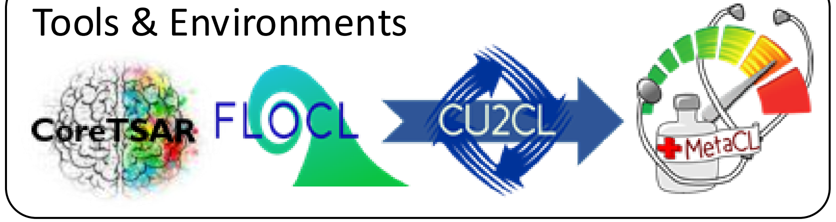


V1

Motivation & Background

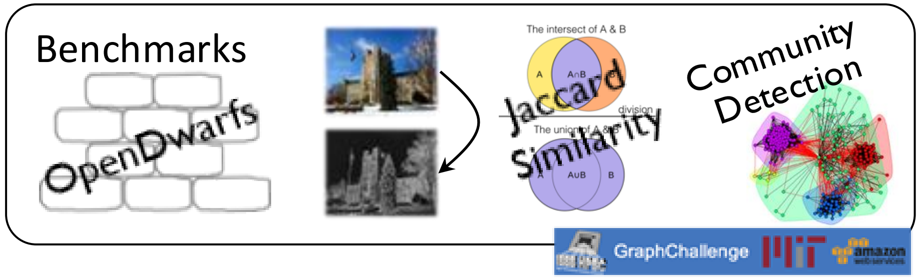
Past CHREC and SHREC memberships on heterogeneous systems

Tools & Environments

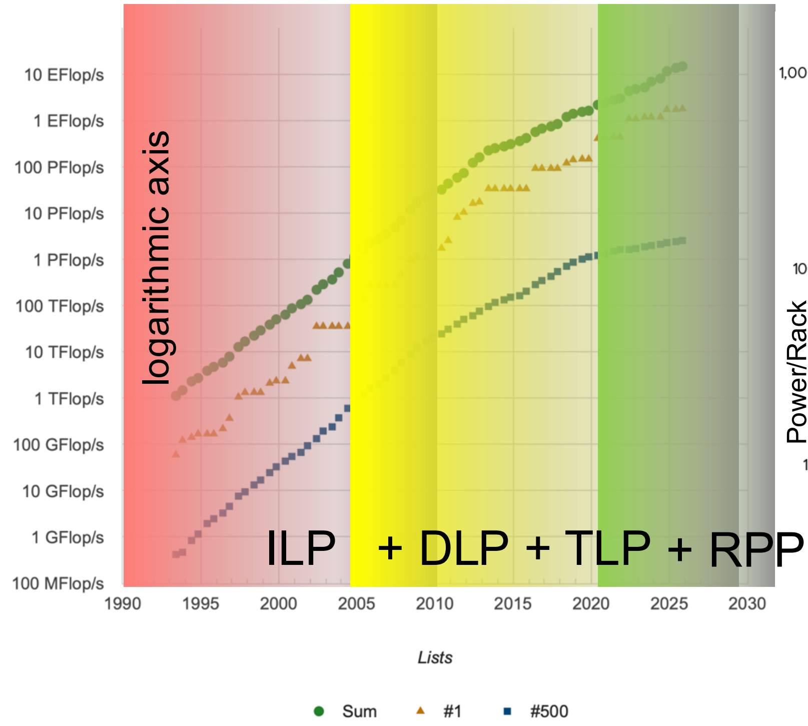


Programming Ecosystems

Benchmarks



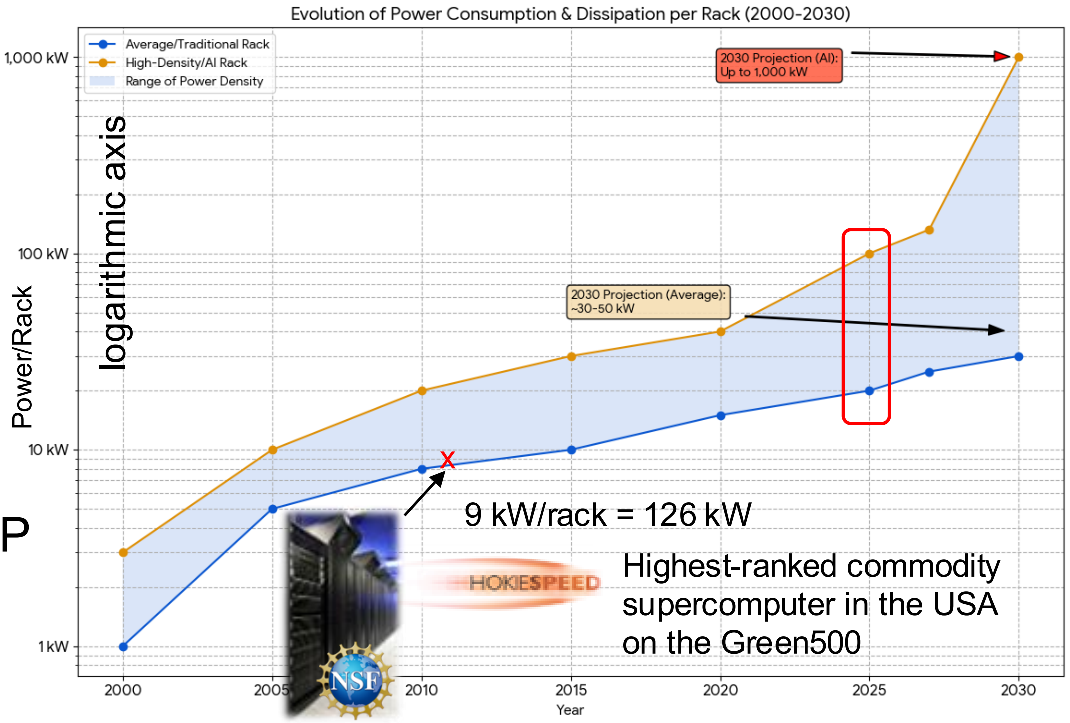
Performance (still matters but …)



logarithmic axis

ILP + DLP + TLP + RPP

Power (becoming an issue …)

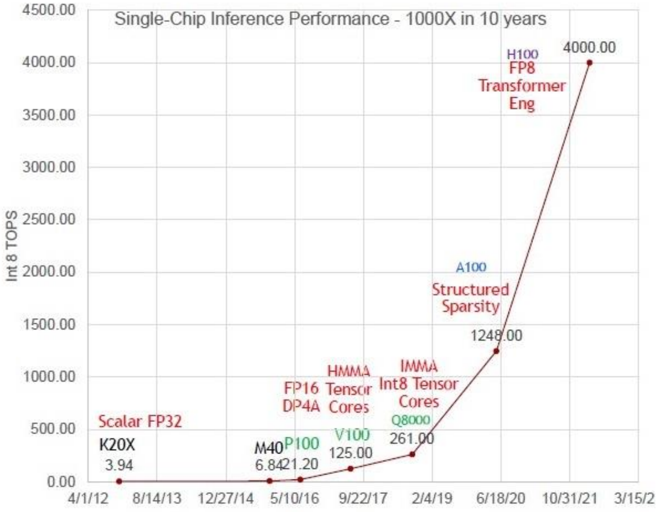


logarithmic axis

9 kW/rack = 126 kW

Highest-ranked commodity supercomputer in the USA on the Green500

Precision (can address both)



RPP: reduced-precision parallelism

logarithmic axis

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

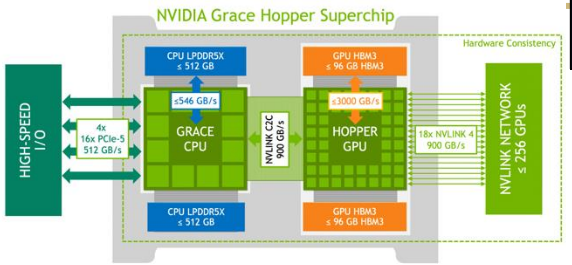
ILP: instruction-level parallelism
DLP: data-level parallelism
TLP: thread-level parallelism

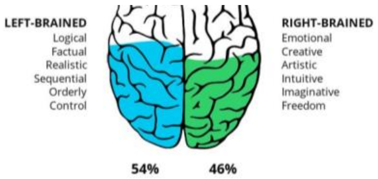
University of Pittsburgh
BYU BRIGHAM YOUNG UNIVERSITY
VIRGINIA TECH
UF UNIVERSITY of FLORIDA

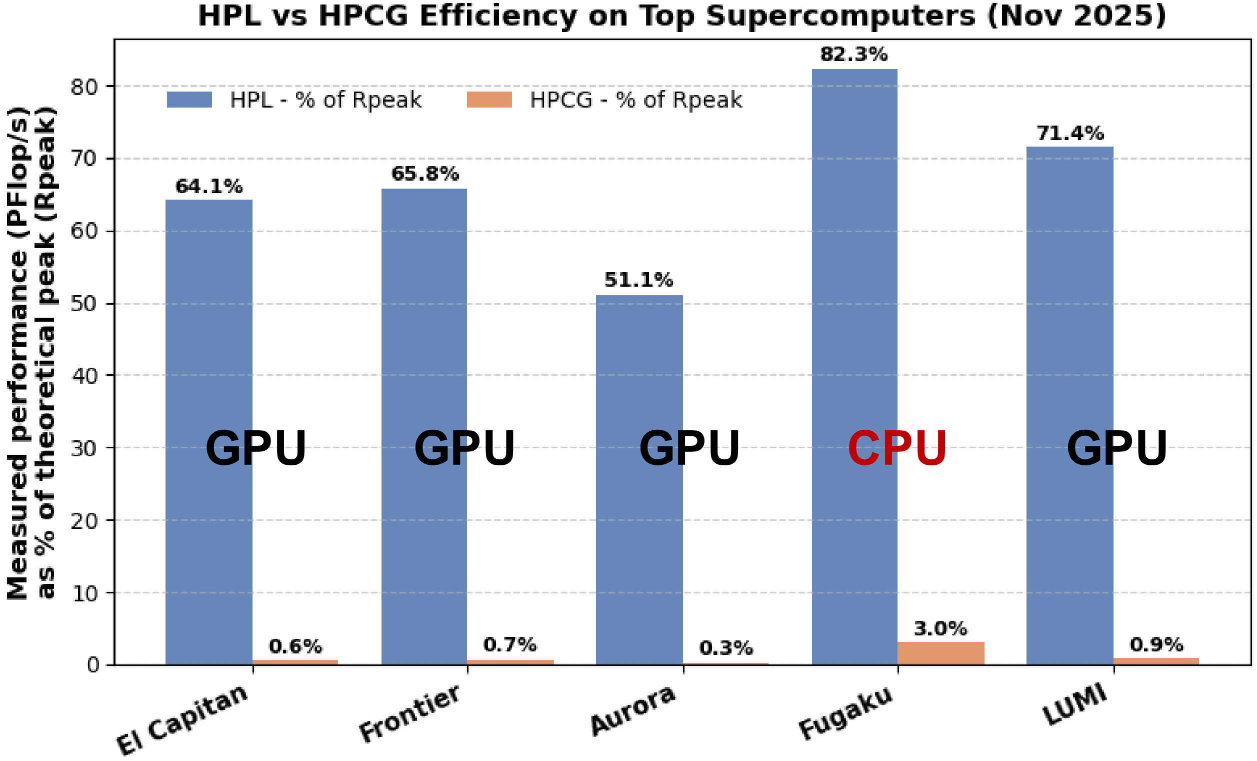
V1

Background: Performance & Power

- **Ranking of fastest supercomputers** (Nov. 2025) based on
  - High-performance LINPACK (HPL) → **REGULAR** workloads
  - High-performance conjugate gradient (HPCG) → **IRREGULAR** workloads
  - GPU: 60%-85% of peak for **regular** workloads; 0%-5% for **irregular** workloads







Supercomputer	TOP500 Rank	HPCG Rank	Accelerator	Peak Performance R_{peak}(PFlop/s)	Power (MW)
El Capitan (LLNL, USA)	1	1	AMD Instinct MI300A GPUs	2821.1	29.7
Frontier (ORNL, USA)	2	3	AMD Instinct MI250X GPUs	2055.0	24.6
Aurora (ANL, USA)	3	4	Intel Data Center Max GPUs	1980.0	38.7
Fugaku (RIKEN, Japan)	7	2	Fujitsu A64FX CPUs (no GPUs)	537.2	29.9
LUMI (CSC, Finland)	9	5	AMD Instinct MI250X GPUs	531.5	7.1

- **Optimizations for performance and power**
  - ✓ Mixed-precision, tuning voltage/frequency
- **Simultaneous co-scheduling for heterogeneity**
  - ✓ CPU + GPU co-scheduling, as appropriate

Approach

V1

- Tune and optimize the ***performance*** of a heterogeneous system
- Characterize the ***power*** and ***energy*** of a heterogeneous system
  - GPU device power via vendor tools (e.g., nvidia-smi)
  - Total system power via power meters and software tools (e.g., RAPL)
- Characterize the ***performance-vs-power*** tradeoff
  - Performance per watt or energy-delay product
  - Power vs. runtime → energy
- Evolve the diversity of ***app benchmarks*** to evaluate the above
  - Regular vs irregular. Double vs. single precision. CPU- vs memory-intensive
- ***Identify the best platform(s)*** and associated ecosystem(s) for performance, power, and/or precision (across many apps)
- ***Enable further performance, power, and precision-aware research**: automated co-scheduling at run-/compile-time, performance vs. power vs. precision tradeoff*

Open Source

OpenCL MPI OpenMP SYCL CHAPEL AMD ROCm oneAPI DATA PARALLEL C++

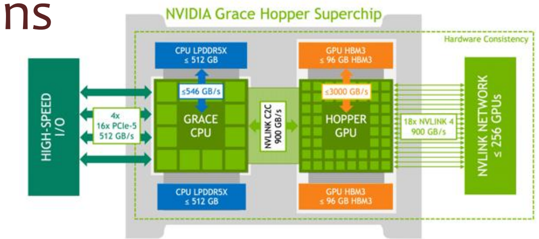
Closed Source

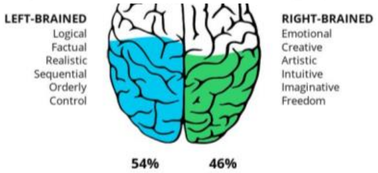
NVIDIA CUDA

Proposed Tasks for V1-26

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

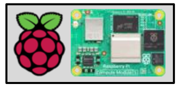
- Task 1: Performance, Power/Energy, & Precision for ***Parallel*** Hetero Computing (2+5)
  - Task 1a: Energy-Efficient/Energy-Dominant Computing for Irregular Applications
  - Task 1b: @Runtime: Simultaneous Co-scheduling on Heterogeneous Devices
  - Task 1c: @Compiler: Simultaneous Co-scheduling on Heterogeneous Devices
  - Task 1d: Portable Runtimes for Heterogeneous Task Graphs
  - Task 1e: Concurrent Data Structures for the GPU

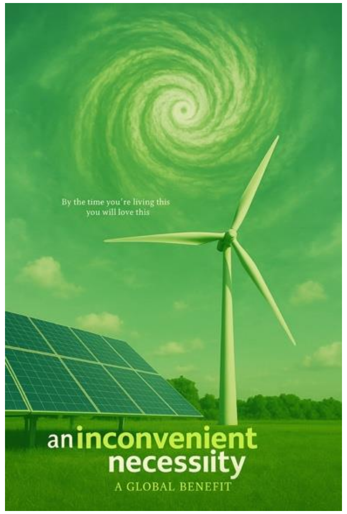




- Task 2: High-Performance ***Distributed*** Computing with GPUs (2+2)
  - Heterogeneous PGAS vs MPI+X for Large-Scale Compute

- Task 3: Performance & Power/Energy for ***Edge*** Computing (1+2)
  - Task 3a: Analysis of Portable Kernel Pipelines for Edge Devices
  - Task 3b: Modeling of Power/Energy Draw via Generative AI



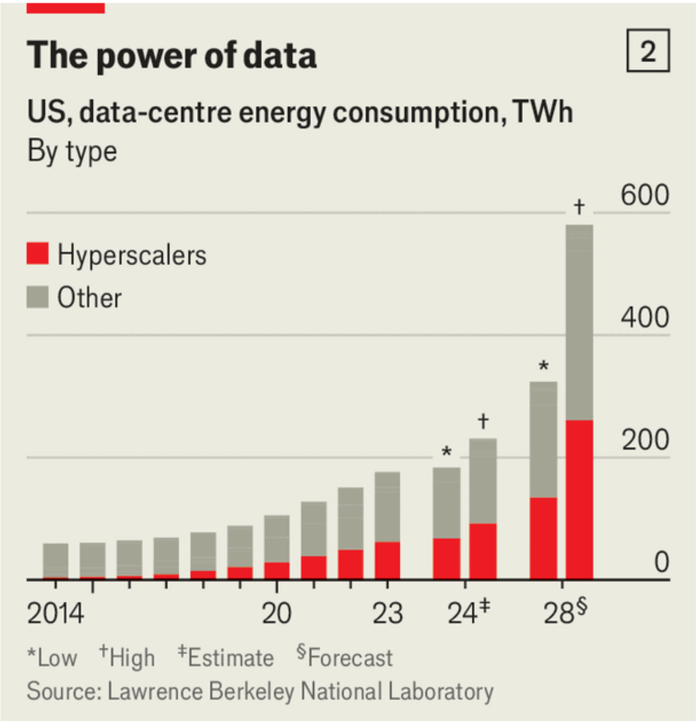


Courtesy: ChatGPT

Task 1a: Energy-Efficient/Energy-Dominant Computing

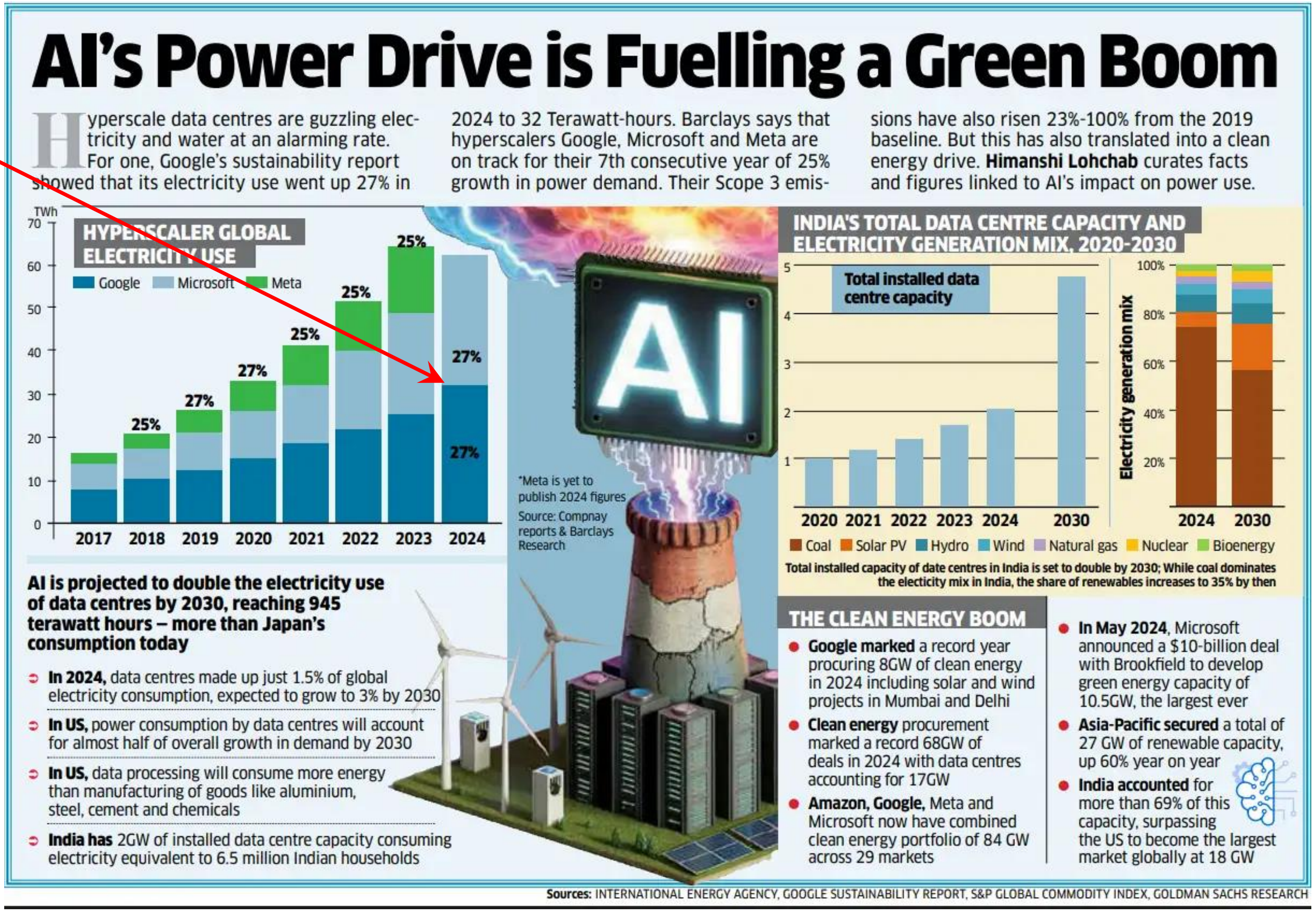
Motivation

- Power & energy are now ***first-order constraints***
  - Hyperscale data center guzzles 20 MW – 50 MW on avg. (with energy consumption ~ 32 TWh)
  - Modern supercomputer uses 10 MW – 40 MW



*Low †High ‡Estimate §Forecast

Source: Lawrence Berkeley National Laboratory



AI's Power Drive is Fuelling a Green Boom

Hyperscale data centres are guzzling electricity and water at an alarming rate. For one, Google's sustainability report showed that its electricity use went up 27% in 2024 to 32 Terawatt-hours. Barclays says that hyperscalers Google, Microsoft and Meta are on track for their 7th consecutive year of 25% growth in power demand. Their Scope 3 emissions have also risen 23%-100% from the 2019 baseline. But this has also translated into a clean energy drive. **Himanshi Lohchab** curates facts and figures linked to AI's impact on power use.

*Meta is yet to publish 2024 figures

Source: Compnay reports & Barclays Research

AI is projected to double the electricity use of data centres by 2030, reaching 945 terawatt hours – more than Japan's consumption today

- **In 2024,** data centres made up just 1.5% of global electricity consumption, expected to grow to 3% by 2030
- **In US,** power consumption by data centres will account for almost half of overall growth in demand by 2030
- **In US,** data processing will consume more energy than manufacturing of goods like aluminium, steel, cement and chemicals
- **India has** 2GW of installed data centre capacity consuming electricity equivalent to 6.5 million Indian households

INDIA'S TOTAL DATA CENTRE CAPACITY AND ELECTRICITY GENERATION MIX, 2020-2030
Total installed data centre capacity
5
4
3
2
1
2020
2021
2022
2023
2024
2030
Electricity generation mix
100%
80%
60%
40%
20%
2024
2030
Coal
Solar PV
Hydro
Wind
Natural gas
Nuclear
Bioenergy

Total installed capacity of date centres in India is set to double by 2030; While coal dominates the electicity mix in India, the share of renewables increases to 35% by then

THE CLEAN ENERGY BOOM

- **Google marked** a record year procuring 8GW of clean energy in 2024 including solar and wind projects in Mumbai and Delhi
- **Clean energy** procurement marked a record 68GW of deals in 2024 with data centres accounting for 17GW
- **Amazon, Google,** Meta and Microsoft now have combined clean energy portfolio of 84 GW across 29 markets
- **In May 2024**, Microsoft announced a $10-billion deal with Brookfield to develop green energy capacity of 10.5GW, the largest ever
- **Asia-Pacific secured** a total of 27 GW of renewable capacity, up 60% year on year
- **India accounted** for more than 69% of this capacity, surpassing the US to become the largest market globally at 18 GW

Sources: INTERNATIONAL ENERGY AGENCY, GOOGLE SUSTAINABILITY REPORT, S&P GLOBAL COMMODITY INDEX, GOLDMAN SACHS RESEARCH

V1

Task 1a: Energy-Efficient/Energy-Dominant Computing

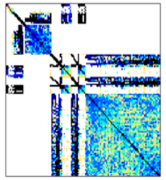
Motivation

- Power & energy are now ***first-order constraints***
  - Hyperscale data center guzzles 20 MW – 50 MW on avg. (with energy consumption ~ 32 TWh)
  - Modern supercomputer uses ~ 10 MW – 40 MW →
- **Power- and energy-aware approaches needed** to align application behaviour with system-level power budgets

Supercomputer	TOP500 Rank	HPCG Rank	Accelerator	Peak Performance R_{peak}(PFlop/s)	Power (MW)
El Capitan (LLNL, USA)	1	1	AMD Instinct MI300A GPUs	2821.1	29.7
Frontier (ORNL, USA)	2	3	AMD Instinct MI250X GPUs	2055.0	24.6
Aurora (ANL, USA)	3	4	Intel Data Center Max GPUs	1980.0	38.7
Fugaku (RIKEN, Japan)	7	2	Fujitsu A64FX CPUs (no GPUs)	537.2	29.9
LUMI (CSC, Finland)	9	5	AMD Instinct MI250X GPUs	531.5	7.1

Approach: Target Apps, Platforms, and Optimizations

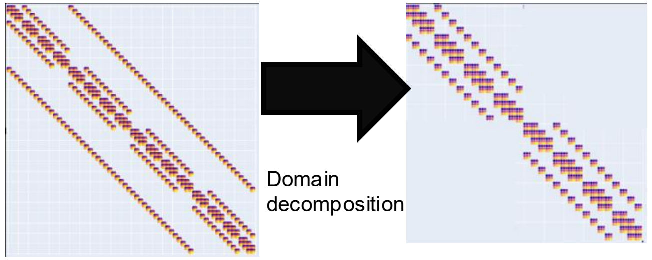
- Apps: Jaccard similarity (JS), conjugate gradient (CG), triangle counting (TC), **[your workload here]** *(see appendix)*
- Platforms: CPU/GPU/APU from AMD, Intel, or NVIDIA
- App-level optimizations
  - (1) mixed- or reduced-precision computing and (2) fine-grained domain decomposition



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

Milestones

1. Application suite for tuning & optimization (i.e., power and energy efficiency)
2. Software-based scripting framework for power and energy measurement
3. Profiling database of power and energy data via hardware & software meters
   - Power → vendor tools (e.g., nvidia-smi, rocm-smi)
   - Energy → integration of power over runtime



Validation of software meters w.r.t. hardware meters?

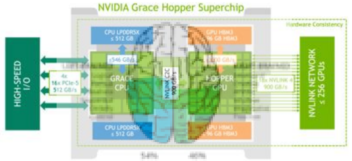
Tasks: Baseline & Optional
(1 + 0)

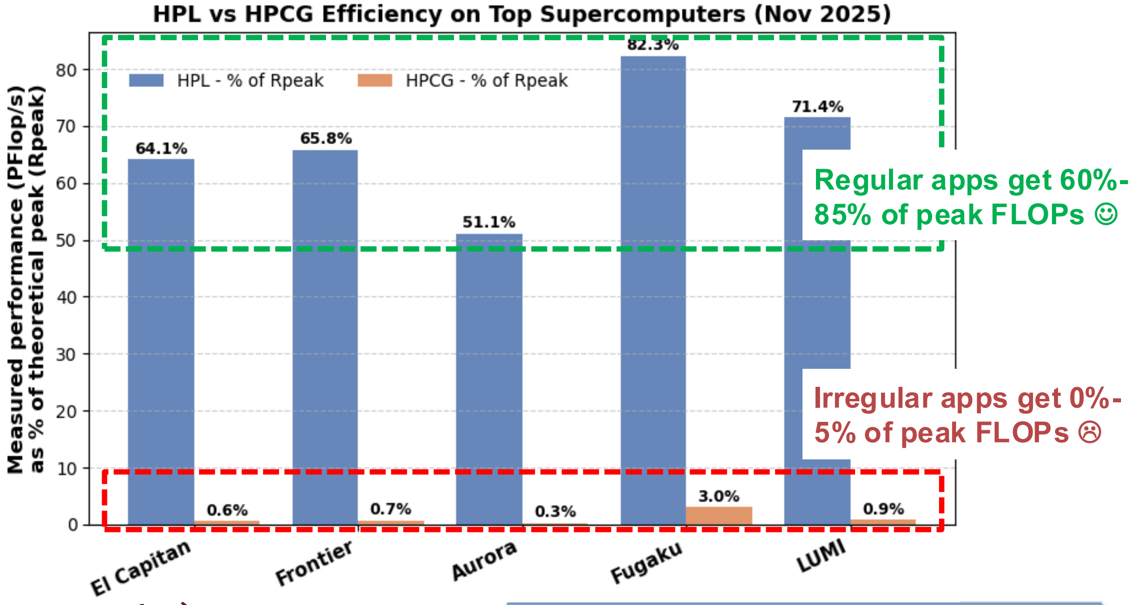
V1

Task 1b: @Runtime Simultaneous Co-scheduling for Heterogeneity

Motivation

- **Supercomputing nodes lean towards GPUs** (3:1 GPU-to-CPU) → presumption: more GPUs will benefit every application
  - Alas, ***NOT*** the case for **irregular applications** → non-coalesced memory accesses, branch divergence, heavy data movement
- Physiologically, we use two brains simultaneously – left and right
  What about "in silico"?
  CPU ***and*** GPU simultaneously?




HPL vs HPCG Efficiency on Top Supercomputers (Nov 2025)

Regular apps get 60%-85% of peak FLOPs ☺

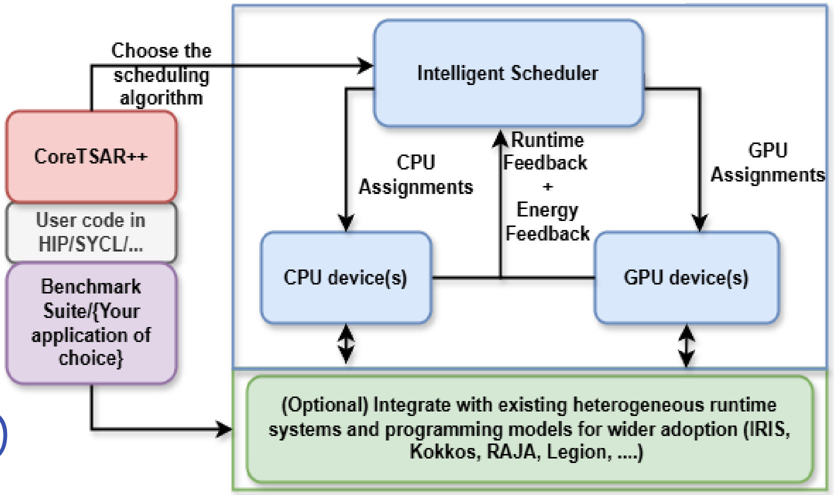
Irregular apps get 0%-5% of peak FLOPs ☹

Approach

- Efficacy of (manual) simultaneous co-scheduling of apps
  - ✓ Broader evaluation across a larger collection of benchmarks (see appendix)
- Build an automated scheduler that fits an accelerator programming model (e.g., OpenMP or [backend of interest] on any xPU) w/ minimal hassle
- Enable automated scheduling for runtime performance & energy cost

Milestones

0. Manual "oracle" co-scheduling of resources
1. CoreTSAR++ automated scheduler with accelerator programming model (e.g., OpenMP on *any* CPU+GPU; optionally, AMD HIP, oneAPI/SYCL Intel)
2. Power measurement methodology (prelude to energy-efficient scheduling)
3. EnergyTSAR++ → New scheduler to minimize runtime energy consumption


Choose the scheduling algorithm

CoreTSAR++

User code in HIP/SYCL/...

Benchmark Suite/{Your application of choice}

Intelligent Scheduler

CPU Assignments

Runtime Feedback + Energy Feedback

GPU Assignments

CPU device(s)

GPU device(s)

(Optional) Integrate with existing heterogeneous runtime systems and programming models for wider adoption (IRIS, Kokkos, RAJA, Legion, ...)

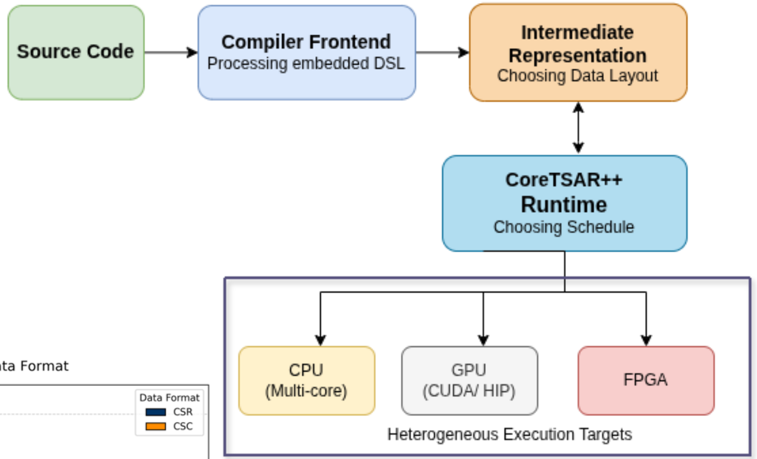
Tasks: Baseline & Optional
(1 + 1)

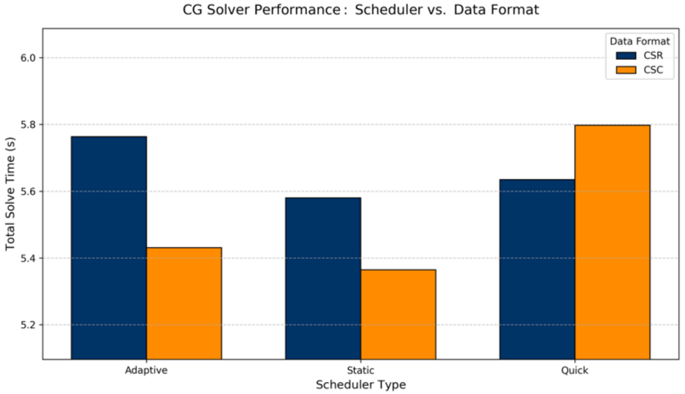
V1

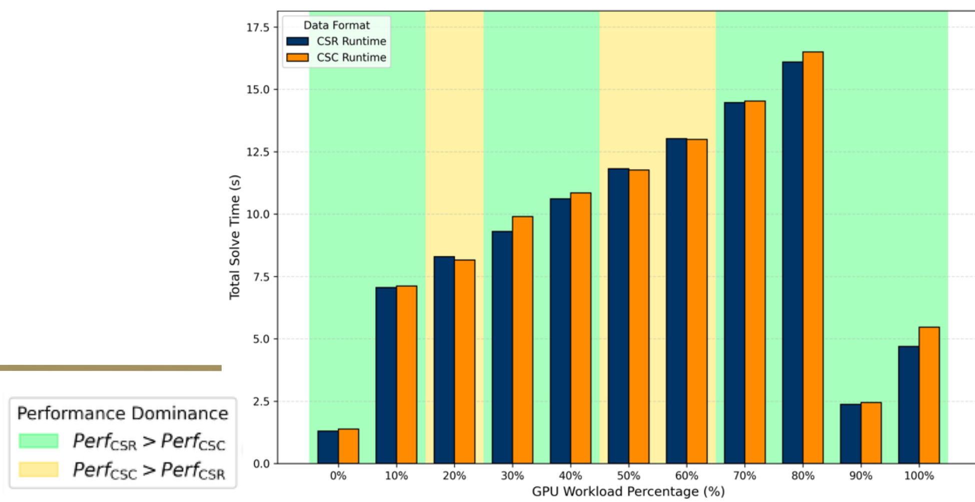
Task 1c: @Compile Simultaneous Co-scheduling for Heterogeneity

Data-Layout Optimizations for Irregular Apps

- Motivation
  - Data layout often influences simultaneous co-scheduling of an app
    - Changing data layout → a schedule favoring different execution targets
  - Challenge: Realizing different schedules by changing data layout
- Approach
  - Template-based embedded DSL (in C++) to abstract data layout from description of computation
  - Compiler plugin to manipulate data layout of program (e.g., MLIR dialect) and generate code accordingly
  - Compiler support to guide co-scheduling of CoreTSAR++ runtime
- Milestones
  1. Bidirectional compiler support for CoreTSAR++ runtime to provide data layout-aware scheduling for hetero execution targets.
  2. Implementation & evaluation of irregular applications from HeCBench in this compilation pipeline.







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

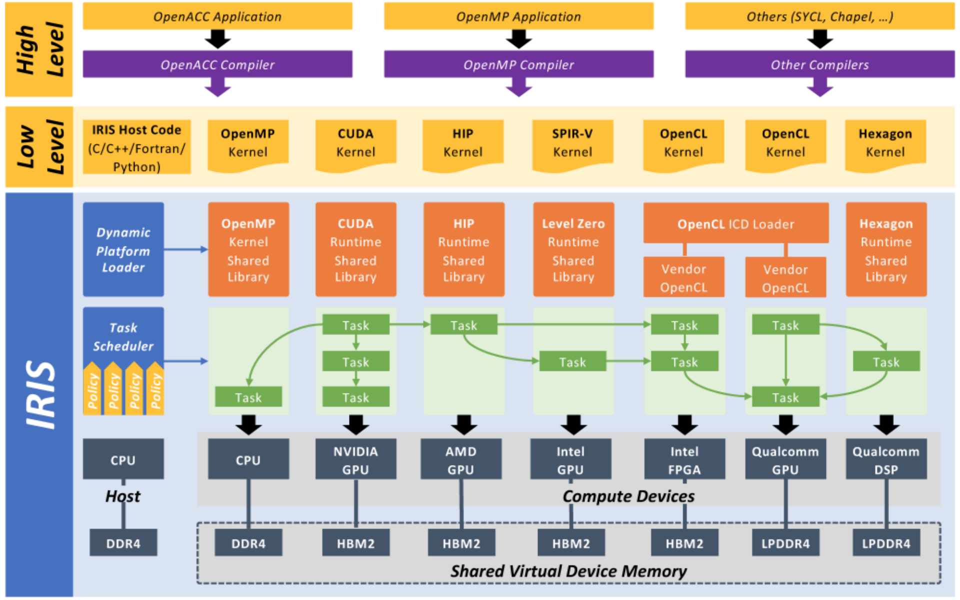
Tasks: Baseline & Optional
(0 + 1)

V1

Task 1d: Portable Runtimes for Heterogeneous Task Graphs

Motivation

- Modern HPC requires device- and system-aware mapping of kernels, communication, and I/O to hardware
- Hardware *migration* (translate, remap, retune) is a significant *cost* [time, human, $$] which slows mission progress
- Portable languages help reduces translation component
- Remapping and re-tuning for new hardware still takes effort!
  - *Intelligent heterogeneous tasking systems* can help!
  - Given a portable representation, model and predict tradeoffs in mapping kernels to different hardware in the system



Approach

- Implement SHREC-related applications using either in OpenMP / OpenARC, or emerging UniSYCL compiler
- Leverage and evaluate the *IRIS portable heterogeneous tasking system*'s ability to achieve high performance

Milestones

1. Identify and migrate/implement a SHREC workload in the IRIS runtime, analyze perf./prod. (Π) (0.5)
2. Evaluate perf./prod. (Π) on traditional heterogeneous HPC (CPU+GPU, ***homogeneous across*** nodes) (1)
3. Evaluate perf./prod. (Π) on *multiply*-heterogenous HPC (CPU+***X***, ***where X differs between nodes***) (1)
4. Evaluate perf./prod. (Π) w/ edge+centralized hybrid workloads w/ heterogeneous platforms (2)
   (i.e. data collection/reduction at the low-power edge, tightly coupled to high-power centralized analysis)

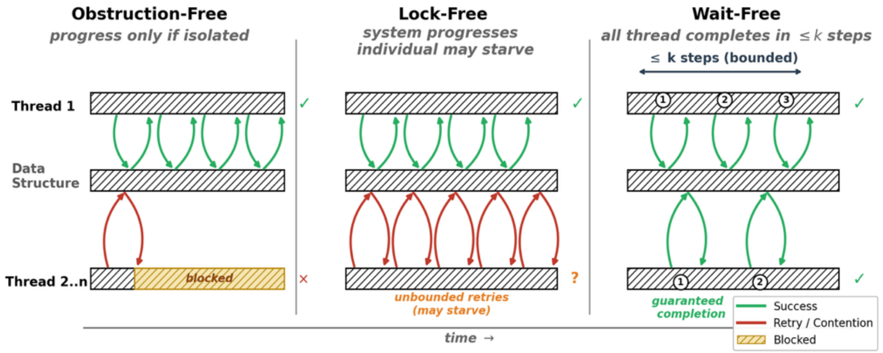
Tasks: Baseline & Optional
(0 + 2)

V1

Task 1e: Concurrent Data Structures for the GPU

Motivation

- CPU concurrent data structures? MATURE (e.g., see Michael & Scott)
- GPU concurrent data structures? NASCENT to NON-EXISTENT
- GPU concurrency bottlenecks NOT addressed by current abstractions



Approach

Analysis of Concurrent Queues for CPU

- Evaluate CPU queue designs

  … with variations in progress guarantees (lock-free / wait-free)

Michael & Scott Queue | Yang & Mellor-Crummey Queue

→

Synthesis of Concurrent Queues for GPU

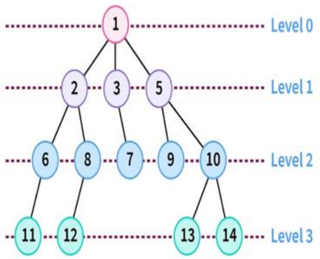
- Refactor for GPU / CPU+GPU architecture
  - ✓ Discrete: AMD MI200, NVIDIA A100/H200
  - ✓ Fused: AMD MI300A, NVIDIA GH200

Refactored M&S Queue | Refactored Y&M Queue

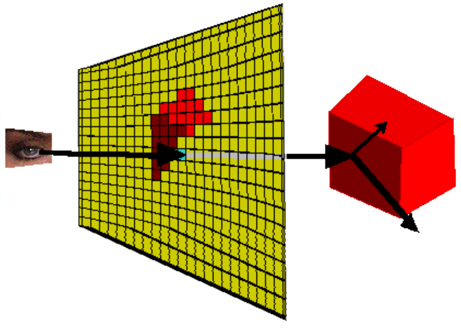
SIMT + atomics

Milestones

1. Adaptation of CPU concurrent queues for GPU
   - Wait-Free-Queue (WFQ), Fetch-and-add based Queue (F&AQ)
2. Synthesis of GPU concurrent queues, e.g., bounded memory, cache-aware
3. Evaluation via microbenchmarks (BFS) and application (ray tracing)



Breadth-First Search



Ray-Tracing

Tasks: Baseline & Optional
(0 + 1)

V1

Task 2: Heterogeneous PGAS vs MPI+X for Large-Scale Compute

Motivation

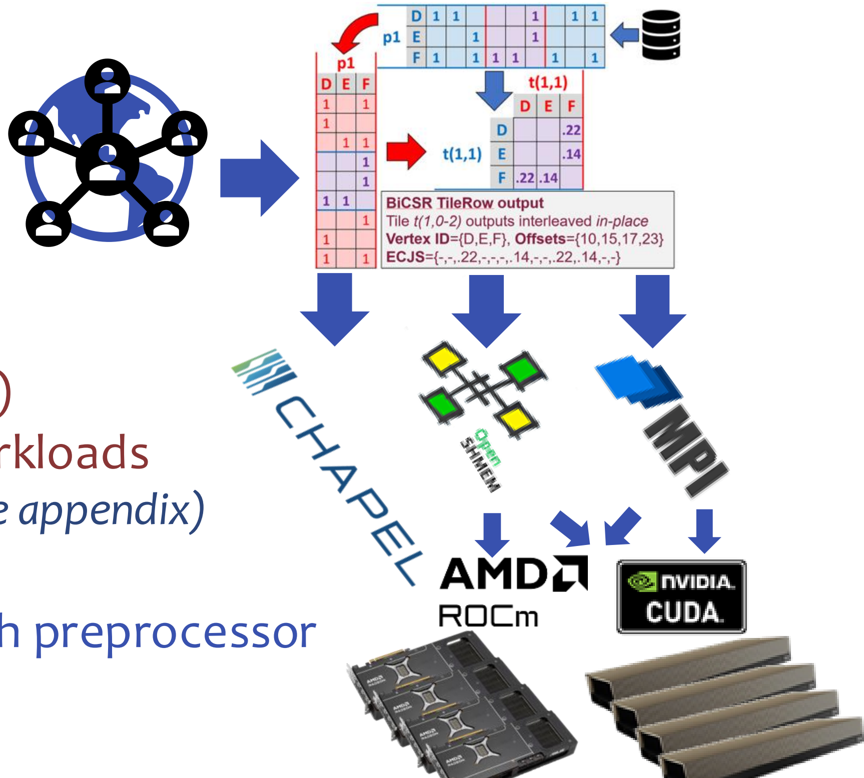
- Proliferation of programming models for *distributed GPUs*
  - MPI+CUDA/HIP, Chapel (PGAS), SHMEM (Open-, NV-/ROCm-)
- Scale-out evaluation needs new inputs too large for 1 GPU (> 100GB)
  - Small (<~1B-edge) graphs starve the GPUs of work when partitioned

Approach

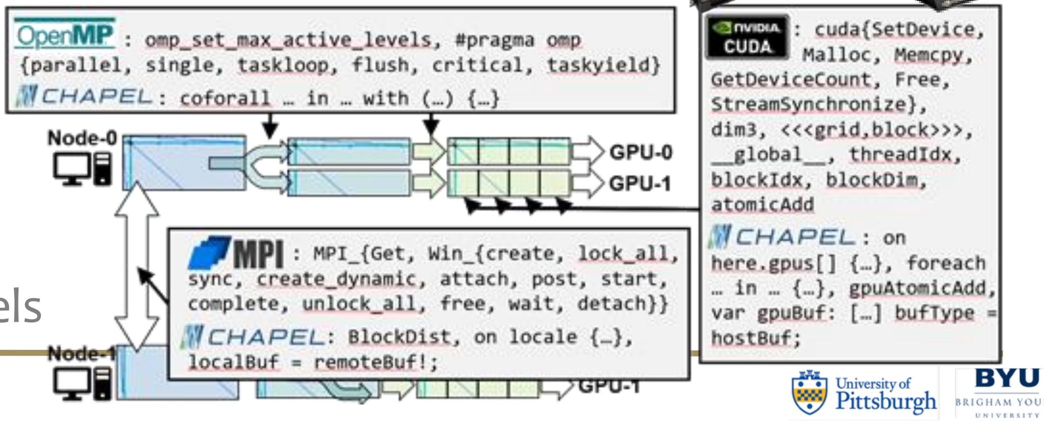
- Implement new preprocessor for *web-scale graphs* (~10B-100B edges)
- Refine and rigorously compare & contrast distributed GPU graph workloads
  - Jaccard similarity (MPI+X, Chapel, OpenSHMEM), … [**your app here**] *(see appendix)*

Milestones

1. LWA (web-scale) → *bidirectional* compressed sparse row (CSR) graph preprocessor
2. Comparative language analysis
   - Intra- and inter-node performance, productivity, power
   - HIP for AMD GPU (MPI+X and OpenSHMEM)
   - CPU- vs. GPU-driven communication models
3. Integration of intra-node hybrid JS approaches
   - Coarse/fine, CPU/GPU, 2d kernels, co-scheduled
4. Additional graph workloads: triangle count, k-truss, etc.
5. Explore reduced-width quantization of intersection kernels



BiCSR TileRow output
Tile *t(1,0-2)* outputs interleaved *in-place*
Vertex ID={D,E,F}, **Offsets**={10,15,17,23}
ECJS={-,-,.22,-,-,-,.14,-,-,.22,.14,-,-}



OpenMP:
```
omp_set_max_active_levels, #pragma omp
{parallel, single, taskloop, flush, critical, taskyield}
```
CHAPEL:
```
coforall … in … with (…) {…}
```
NVIDIA CUDA:
```
cuda{SetDevice,
  Malloc, Memcpy,
GetDeviceCount, Free,
StreamSynchronize},
dim3, <<<grid,block>>>,
__global__, threadIdx,
blockIdx, blockDim,
atomicAdd
```
CHAPEL:
```
on
here.gpus[] {…}, foreach
… in … {…}, gpuAtomicAdd,
var gpuBuf: […] bufType =
hostBuf;
```
MPI:
```
MPI_{Get, Win_{create, lock_all,
sync, create_dynamic, attach, post, start,
complete, unlock_all, free, wait, detach}}
```
CHAPEL:
```
BlockDist, on locale {…},
localBuf = remoteBuf!;
```

Tasks: Baseline & Optional
(**2** + 2)

University of Pittsburgh
BYU BRIGHAM YOUNG UNIVERSITY
VIRGINIA TECH
UF UNIVERSITY of FLORIDA

Task 2: Heterogeneous PGAS vs MPI+X vs SHMEM

- Capturing high-level language tradeoffs for parallel & distributed computing (work in progress)

Feature	Chapel	SHMEM		MPI+X
		Open-	NV-	
GPU-driven communication	NO	NO	YES	Depends on implementation
Consistent GPU/node API	YES	NO	Depends	NO
Vendor-neutral API	YES	YES	NO	Depends on kernel language
Independently-sized per-node GETable allocations	YES	NO	NO	Dynamic-only
Exposed thread blocking	NO	YES	?	YES
Node-specific partial data	YES	If same size	If same size	YES

"Can I transfer GPU pointers without manually staging on the CPU?"

"Do I copy between nodes and between the CPU and GPU with the same API?"

"Can my code run on another vendor's hardware without rewrites?"

"Can I allocate different sizes of communicable data on different ranks?"

"Can I force a user-facing thread to spin in the foreground?"

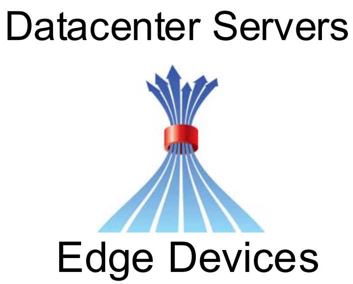
"Can I allocate *non-overlapping* subsets of data across ranks, and still communicate them?"

V1

Task 3a: Analysis of Portable Kernel Pipelines for Edge Devices

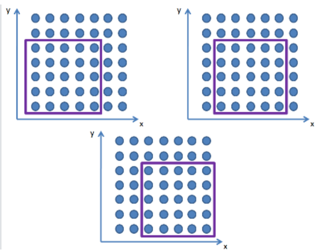
Motivation

- Proliferation of edge devices creates a data bottleneck with centralized processing
- Moving pre-filtering and other compute to the edge reduces aggregate bandwidth and storage
  - Issue: Wide range of edge devices that require ***different*** programming approaches
  - Challenge: Is there a robust path from portable HPC languages to low-power edge devices?



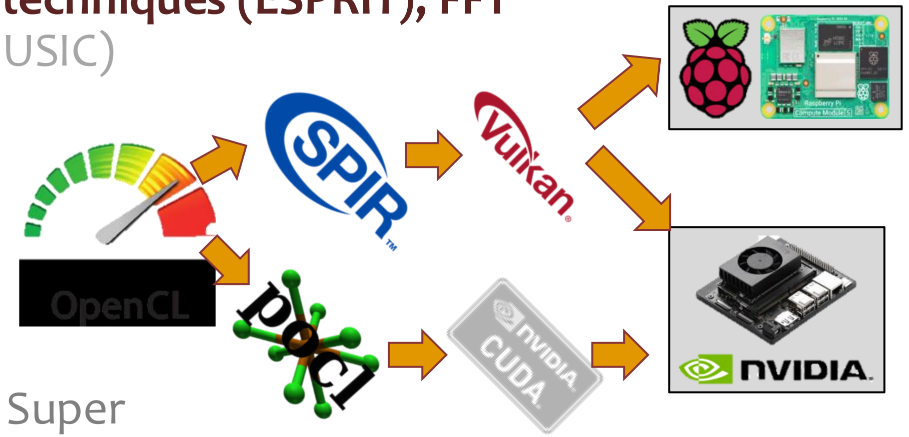
Approach

- Leverage portable, open HPC standards, and open-source toolchains to compute on edge GPU(s)
  - Examples: OpenCL, SYCL, SPIR, Vulkan
- Metrics: performance, power, productivity, and performance/power
- Platforms: Raspberry Pi Compute Module (CM) 5 and Nvidia Jetson Orin Nano Super
- Workloads: **Estimation of signal parameters via rotational invariant techniques (ESPRIT), FFT convolution,** [**your workload here**], multiple signal classification (MUSIC)



Milestones

1. Raspberry Pi CM5 via OpenCL C → SPIR-V → Vulkan (0.5)
   - FFT convolution, ESPRIT→ Perf./Power (Π) analysis
2. Nvidia Jetson Orin Nano via OpenCL C→ SPIR-V→ Vulkan (0.5)
3. MUSIC algorithm on Raspberry Pi CM5 and Nvidia Jetson Orin Nano Super



Tasks: Baseline & Optional
(1 + 1)

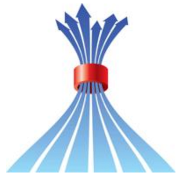
V1

Task 3b: Modeling of Power/Energy Draw via Generative AI

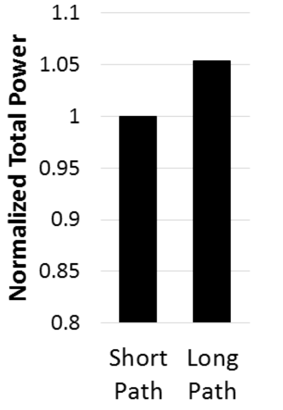
Motivation

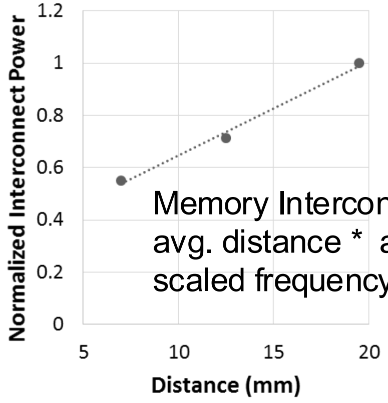
- Environment: Tactical computing on the edge
  (i.e., move from datacenter to the edge)
- Challenge: Edge devices energy-constrained
  - How to deploy generative AI models that are constrained by the "AI Memory Wall," where energy consumption is dominated by data movement
- State of the Art
  - Reliance on *oversimplified* linear models that do NOT capture complex, non-linear dynamics of today's GPU

Datacenter Servers



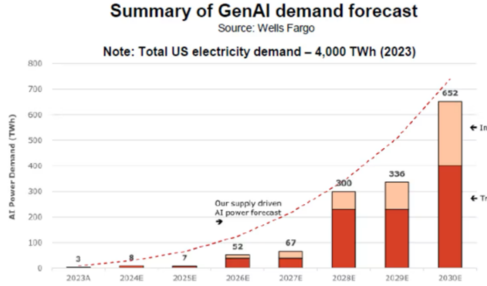
Edge Devices





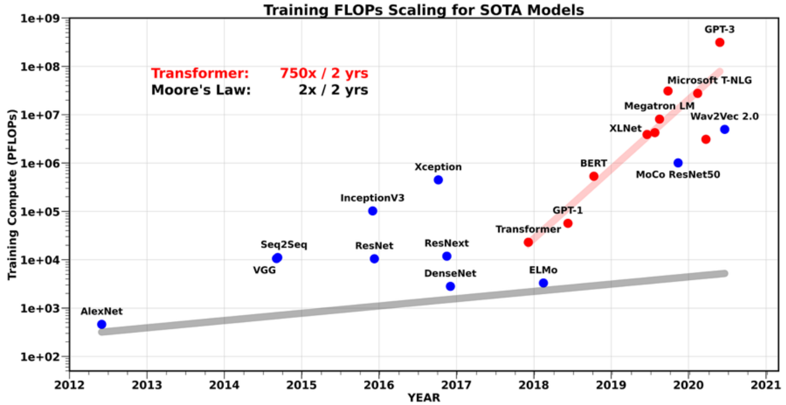
Memory Interconnect Power = Energy/bit/mm * avg. distance * avg. bits/sec * scaled voltage^2 * scaled frequency * avg. toggle rate

Source: Adhinarayanan, Paul, Greathouse, Huang, Pattnaik, Feng. *IISWC*, 2016. **Best Paper Award**


Summary of GenAI demand forecast

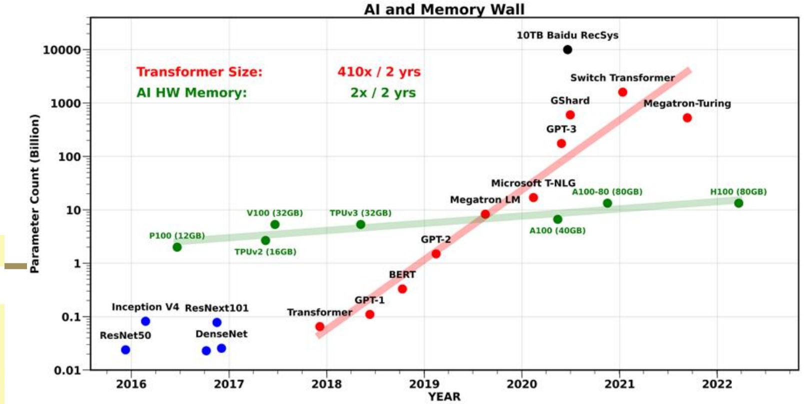
Source: Wells Fargo

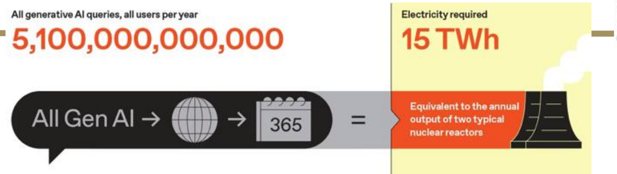
Note: Total US electricity demand – 4,000 TWh (2023)


Training FLOPs Scaling for SOTA Models

Approach

- Memory-centric energy-modeling framework that integrates actual measurements and intelligent optimization to enable accurate prediction and systematic minimization of energy use in transformer-based models
  - Memory access can be ***100-200 times more energy-intensive*** than computation
  - Re-orientation of the energy optimization problem around data movement, creating new pathways to deploy powerful foundational models on edge & tactical hardware


AI and Memory Wall


All generative AI queries, all users per year

5,100,000,000,000

Electricity required

15 TWh

Equivalent to the annual output of two typical nuclear reactors

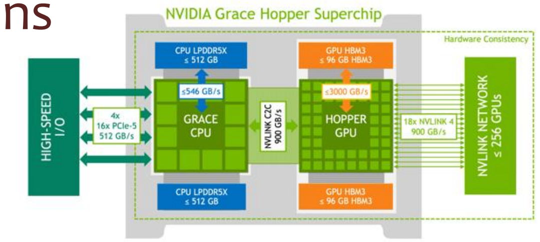
Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

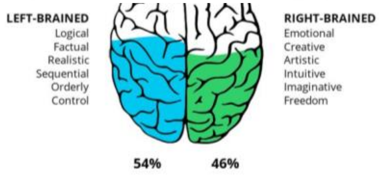
Tasks: Baseline & Optional
(0 + 1)

Proposed Tasks for V1-26

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

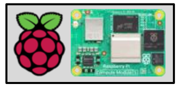
- Task 1: Performance, Power/Energy, & Precision for ***Parallel*** Hetero Computing (2+5)
  - Task 1a: Energy-Efficient/Energy-Dominant Computing for Irregular Applications
  - Task 1b: @Runtime: Simultaneous Co-scheduling on Heterogeneous Devices
  - Task 1c: @Compiler: Simultaneous Co-scheduling on Heterogeneous Devices
  - Task 1d: Portable Runtimes for Heterogeneous Task Graphs
  - Task 1e: Concurrent Data Structures for the GPU

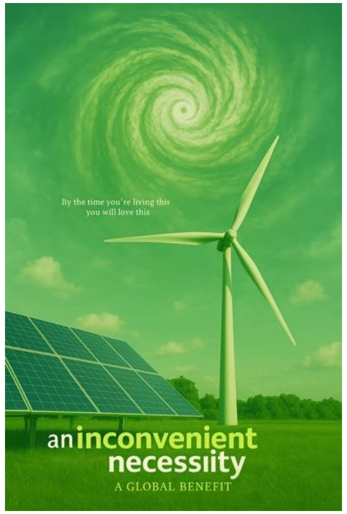




- Task 2: High-Performance ***Distributed*** Computing with GPUs (2+2)
  - Heterogeneous PGAS vs MPI+X for Large-Scale Compute

- Task 3: Performance & Power/Energy for ***Edge*** Computing (1+2)
  - Task 3a: Analysis of Portable Kernel Pipelines for Edge Devices
  - Task 3b: Modeling of Power/Energy Draw via Generative AI





Courtesy: ChatGPT

Appendix

V1

Background & Motivation

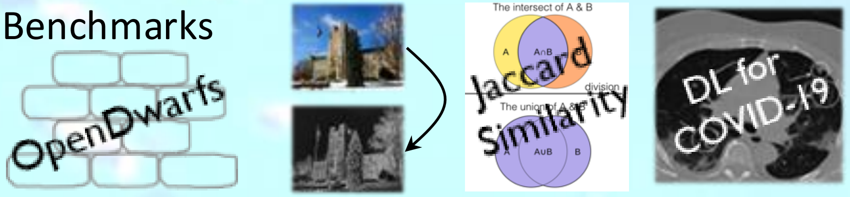
- Extend our R&D to create and analyze an ecosystem of *tools, environments, and benchmarks for heterogeneous computing*

Past CHREC- and SHREC-funded R&D

Tools & Environments

CoreTSAR FLOCL CU2CL MetaCL

Benchmarks



Devices

Programming Ecosystems

OpenMP CHAPEL OpenACC Directives for Accelerators oneAPI
OpenCL MPI SYCL NVIDIA CUDA AMD ROCm Open SHMEM

- **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?
  - Ensure the system's power and precision/accuracy constraints are met?

} Application and platform -dependent

V1

Motivation

How BIG tech plans to feed AI’s voracious appetite for power

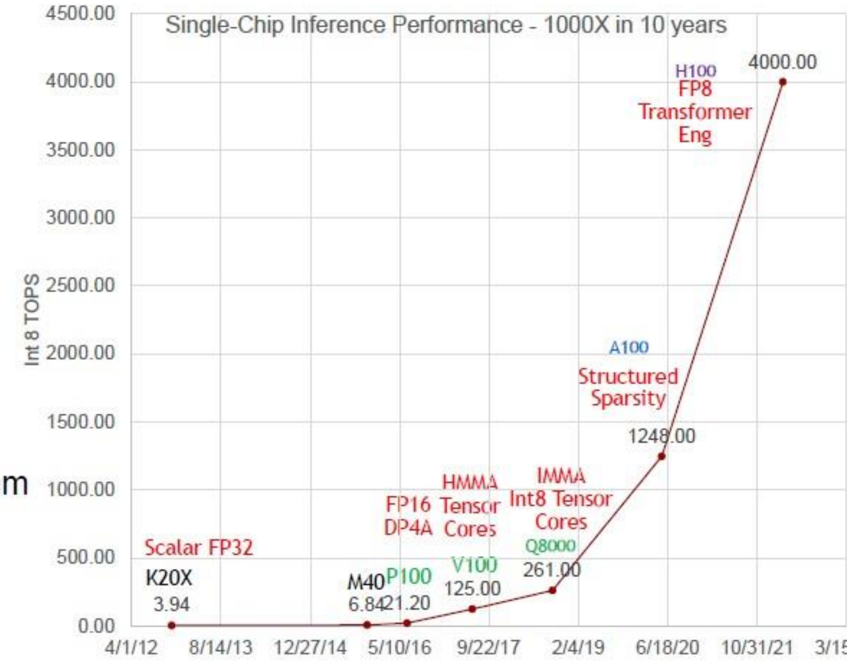
As data centers get more energy-hungry, the hyperscalers get more creative

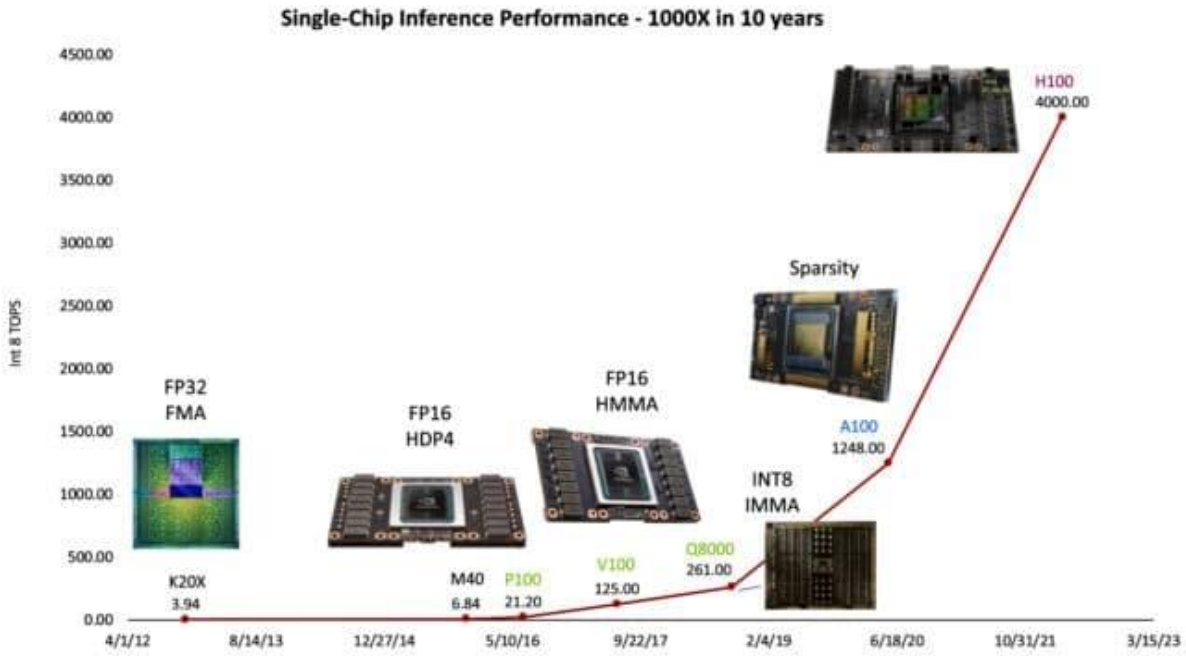
Gains from

Number representation
FP32, FP16, Int8
(TF32, BF16)

Complex instructions
DP4, HMMA, IMMA

Process
28nm, 16nm, 7nm, 5nm



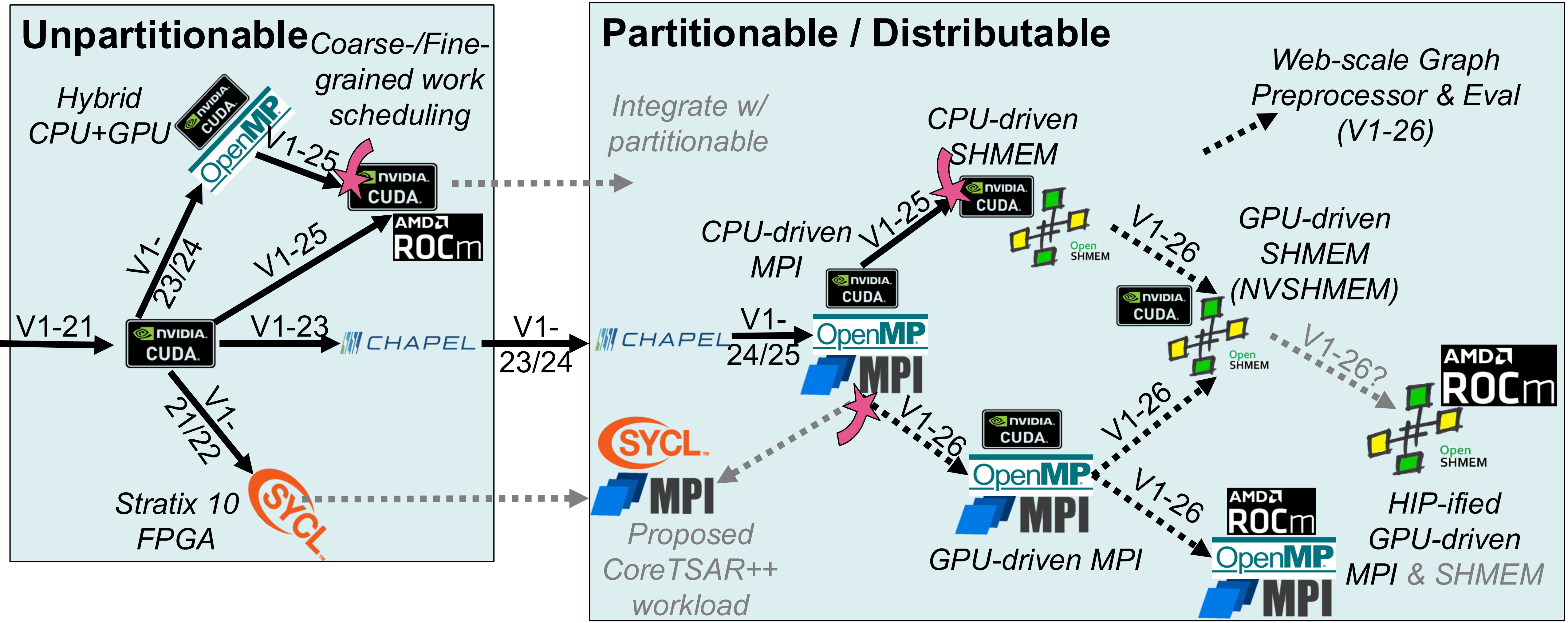


Source: Nvidia Blog

Tasks 1a & 2: Context for Jaccard Similarity

V1

You are here



Unpartitionable

Hybrid CPU+GPU

Coarse-/Fine-grained work scheduling

Stratix 10 FPGA

Partitionable / Distributable

Integrate w/ partitionable

CPU-driven MPI

CPU-driven SHMEM

GPU-driven SHMEM (NVSHMEM)

Web-scale Graph Preprocessor & Eval (V1-26)

Proposed CoreTSAR++ workload

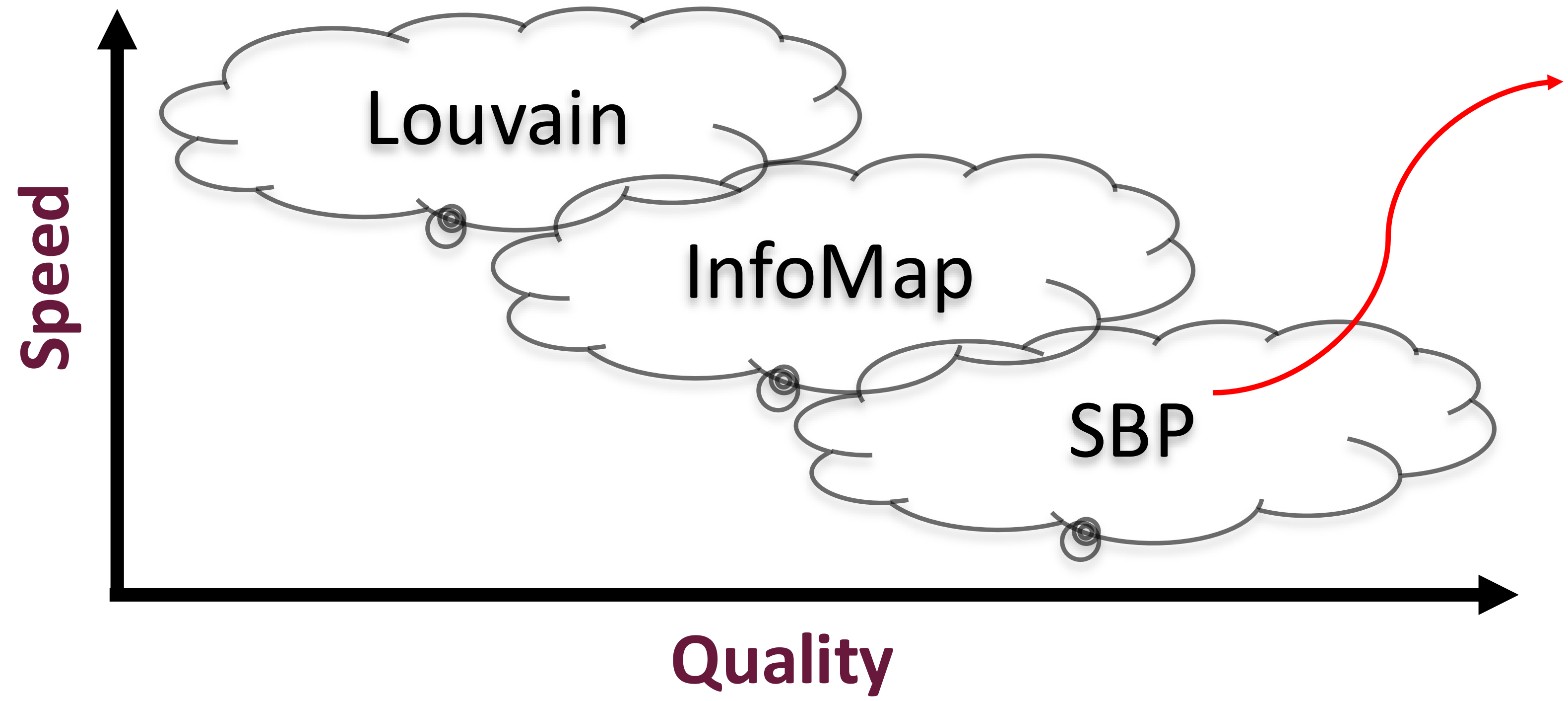
GPU-driven MPI

HIP-ified GPU-driven MPI & SHMEM

Tasks 1a, 1b, 1c, 1d, 2: Prospective Irregular Workload: SBP

Optimal graph clustering is NP-hard → GraphChallenge *

Enable <u>fast and accurate</u> graph clustering in large graphs by accelerating <u>SBP</u>



Wanye, Gleyzer, Kao, and Feng.
MIT Graph Challenge Champion, 2023

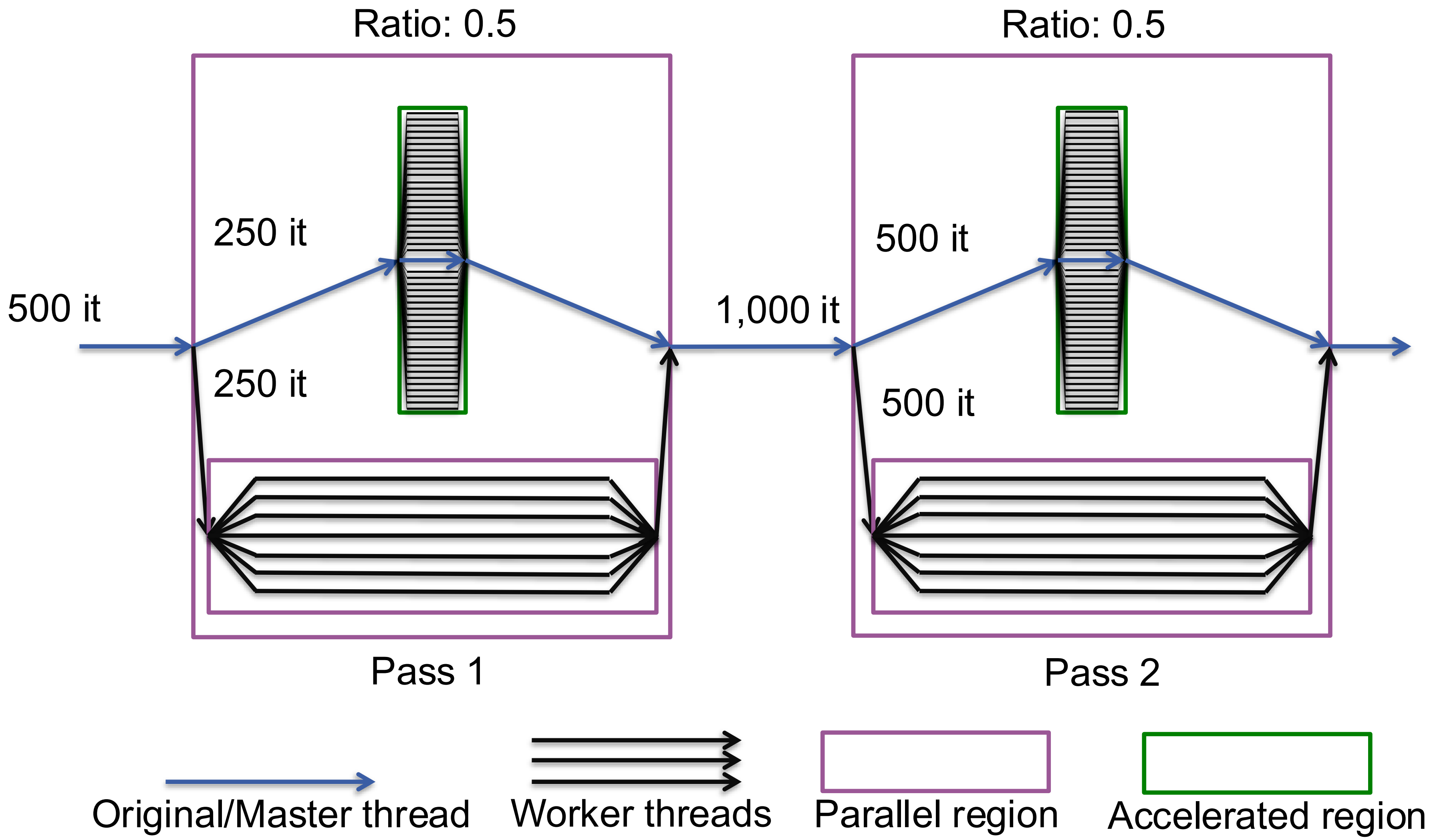
Port SBP to GPU

Apply to applications like network intrusion detection

[*] E. Kao, V. Gadepally, M. Hurley, M. Jones, J. Kepner, S. Mohindra, P. Monticciolo, A. Reuther, S. Samsi, W. Wong, D. Staheli, S. Smith, “Streaming Graph Challenge: Stochastic Block Partition,” *Proc. IEEE HPEC*, 2017

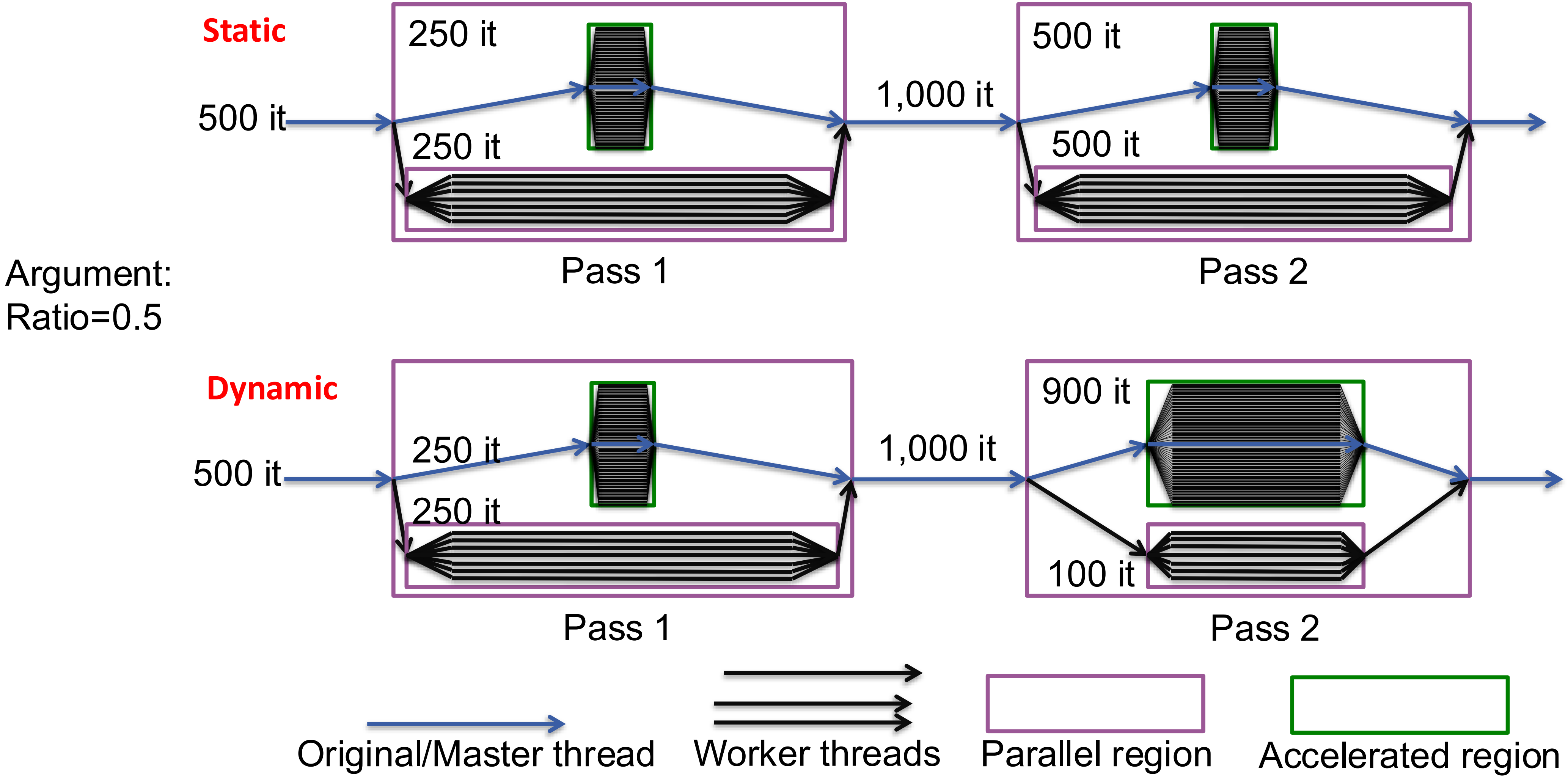
Task 1b: CoreTSAR: Core Task-Size Adapting Runtime

Static Scheduler



Task 1b: CoreTSAR: Core Task-Size Adapting Runtime

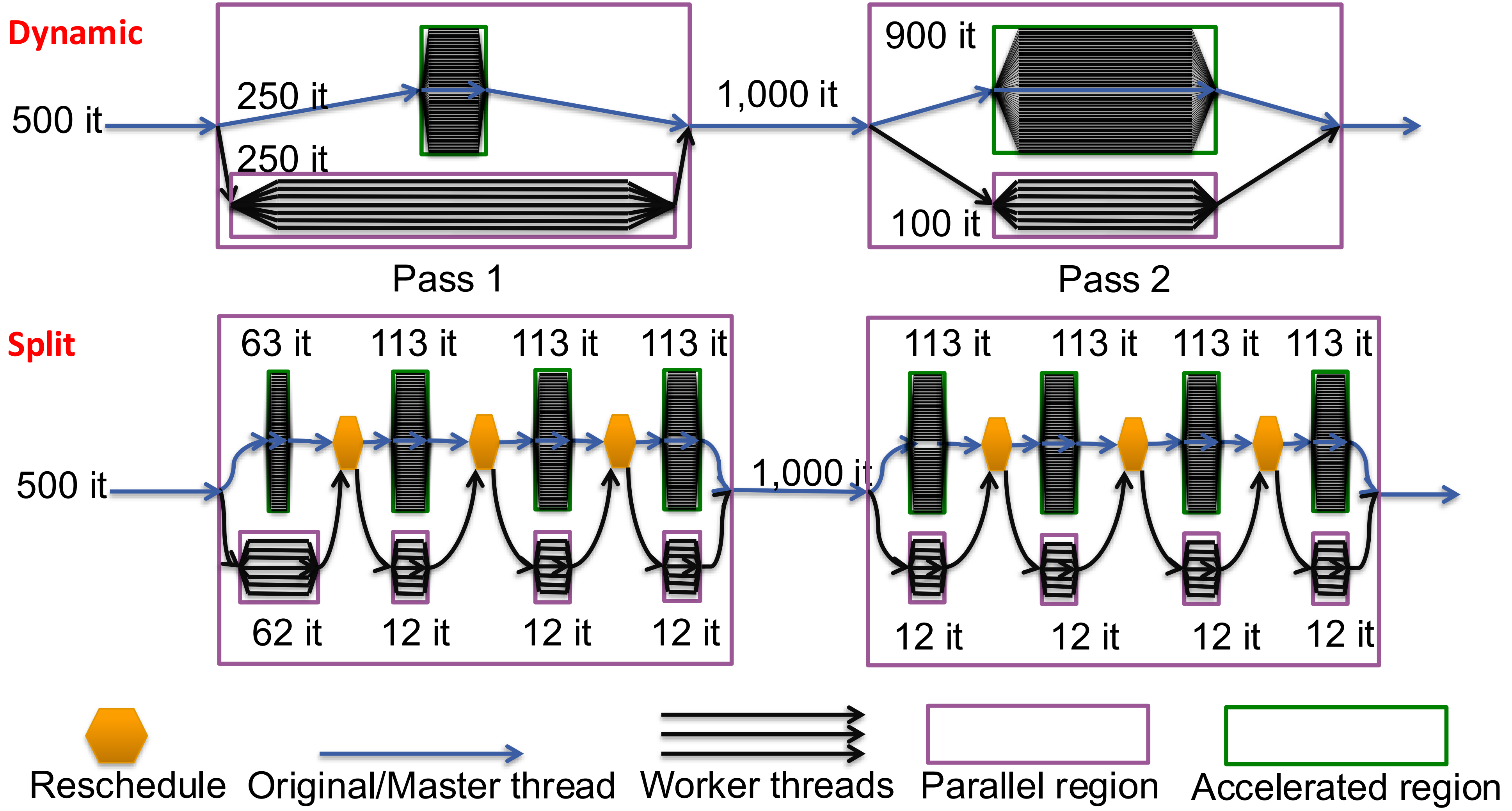
Static vs. Dynamic Scheduler



Argument:
Ratio=0.5

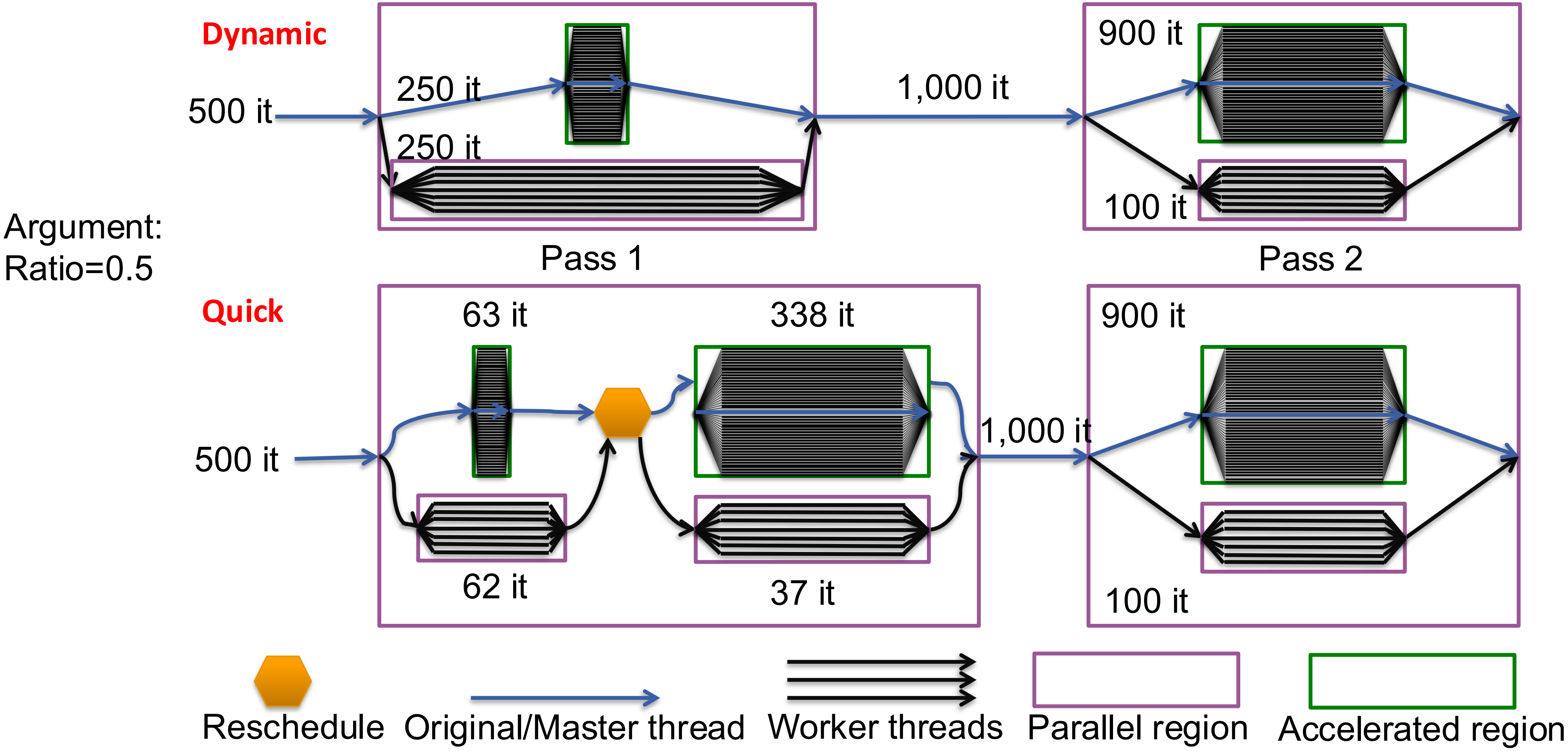
Task 1b: CoreTSAR: Core Task-Size Adapting Runtime

Dynamic vs. Split Scheduler



Task 1b: CoreTSAR: Core Task-Size Adapting Runtime

Split vs. Quick Scheduler



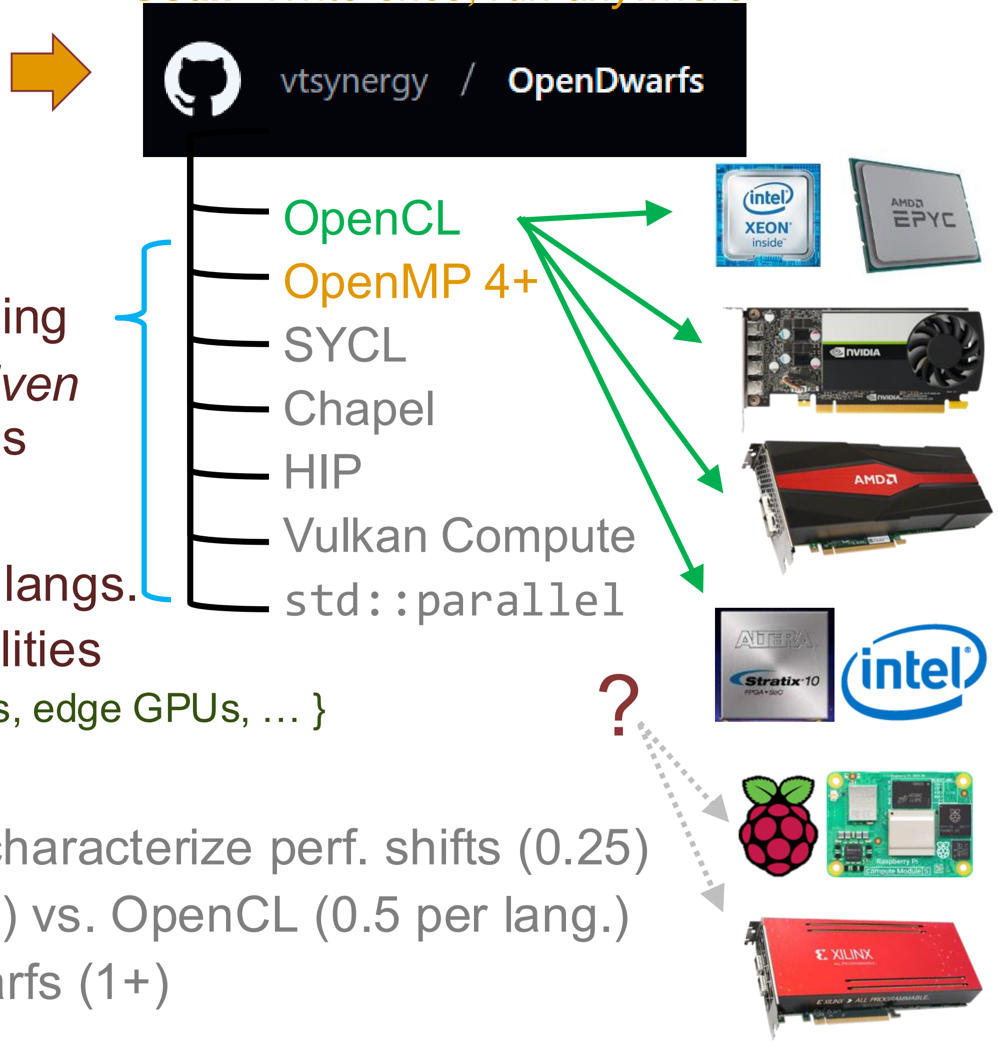
V1

“Task 2b”: Modernization of OpenDwarfs

- Motivation:
  - OpenDwarfs was a CHREC project to show how to map 13 *parallel computational idioms* to *GPUs* via OpenCL
    - Part learning tool, part benchmark suite
    - Eventually extended to Intel/Altera FPGAs
  - Now *many more paths* to portable, heterogeneous computing
  - To bridge *programming gap* between high-level, *library-driven* heterogeneity, need examples of how to write *novel* kernels
- Approach:
  - Showcase idiomatic parallel codes using modern portable langs.
  - Modernize for new classes of devices, and compute modalities
    - {unified memory, PGAS, tensor cores, HBM, hybrid co-scheduling, DSPs, edge GPUs, … }
- Milestones

1. Update existing OpenCL Dwarfs for modern devices → characterize perf. shifts (0.25)
2. Implement Dwarfs in new lang(s)., analyze perf./prod. (Π) vs. OpenCL (0.5 per lang.)
3. Design *partitionable/distributable* variants of existing dwarfs (1+)

Goal: *“Write once, run anywhere”*



Tasks: Optional (1-2 memberships)