Accelerators in FLAMEs

Johannes M Dieterich
Strategic Leadership Optimization Team (HPC)
Machine Learning Software Engineering
Radeon Technologies Group
Advanced Micro Devices, Inc.

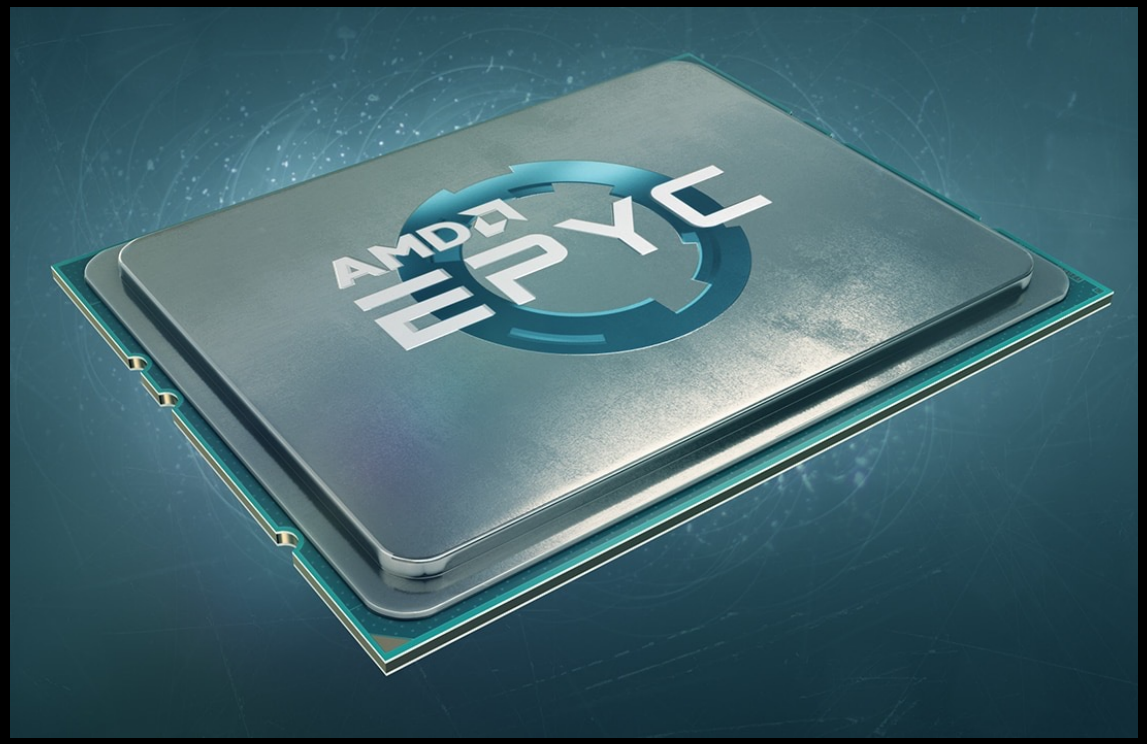
OLCF-5: Frontier

- World's first exascale computer June 2022 TOP500 with 1.102 Exaflop/s
- Top of June 2022 GREEN500 at 62.68 Gigaflops/Watt
- 9,472 AMD EPYC™ 64 core 2 GHz CPUs
- 37,888 AMD Instinct™ MI250X accelerators
- 1 EPYC : 4 MI250X
- Places 1-4 on June 2022 GREEN500 are AMD EPYC™ with AMD™ MI250X systems.

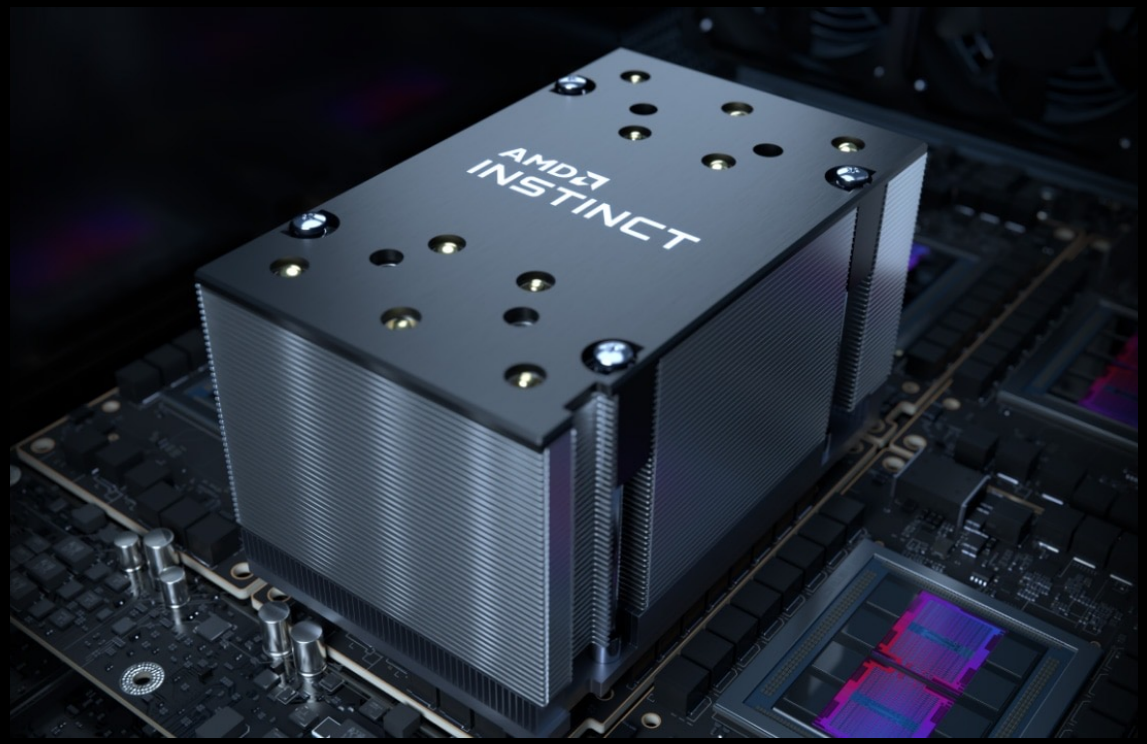
Photo under CC BY 2.0 from https://www.flickr.com/photos/olcf/52117623843/

TOP500: https://www.top500.org/lists/top500/2022/06/ GREEN500: https://www.top500.org/lists/green500/2022/06/

Peak performance differences



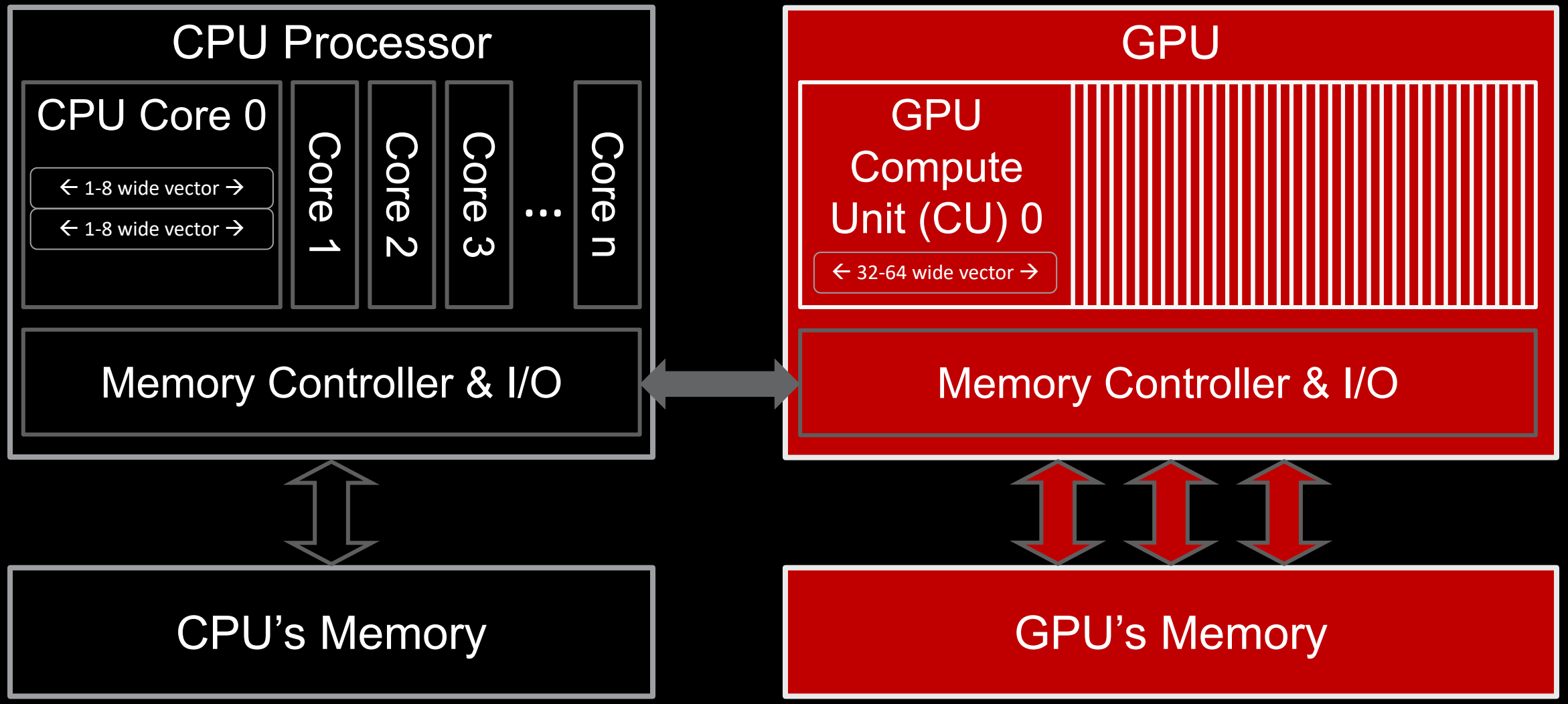
2.0 GHz 64 core AVX256 FMAs
Double precision peak: approx. 1 TFLOPs



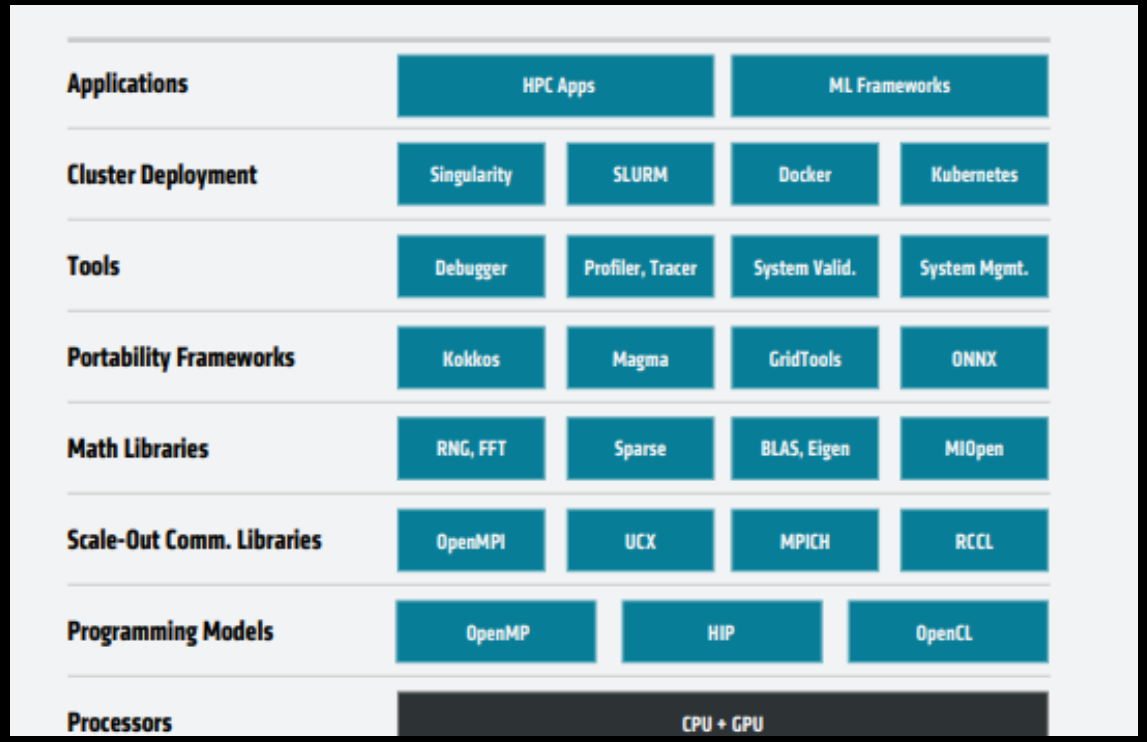
Single AMD Instinct™ MI250X
Double precision peak: 47.9 TFLOPs
Double precision matrix peak: 95.7 TFLOPs

https://www.amd.com/en/processors/epyc-server-cpu-family
https://www.amd.com/en/products/server-accelerators/instinct-mi250x

Illustration of a CPU+GPU Heterogeneous System



AMD ROCm™ platform – open-source acceleration



Full OSS kernel to application stack

```
__global__ void saxpy (unsigned N, float
a, float *x, float *y) {
        unsigned i = blockIdx.x
                * blockDim.x + threadIdx.x;
        if (i < N) y[i] = a*x[i] + y[i];
}


hipLaunchKernel(saxpy,
                dim3(N/blockSize),
                dim3(blockSize),
                0, 0, N, a, x, y); }
```

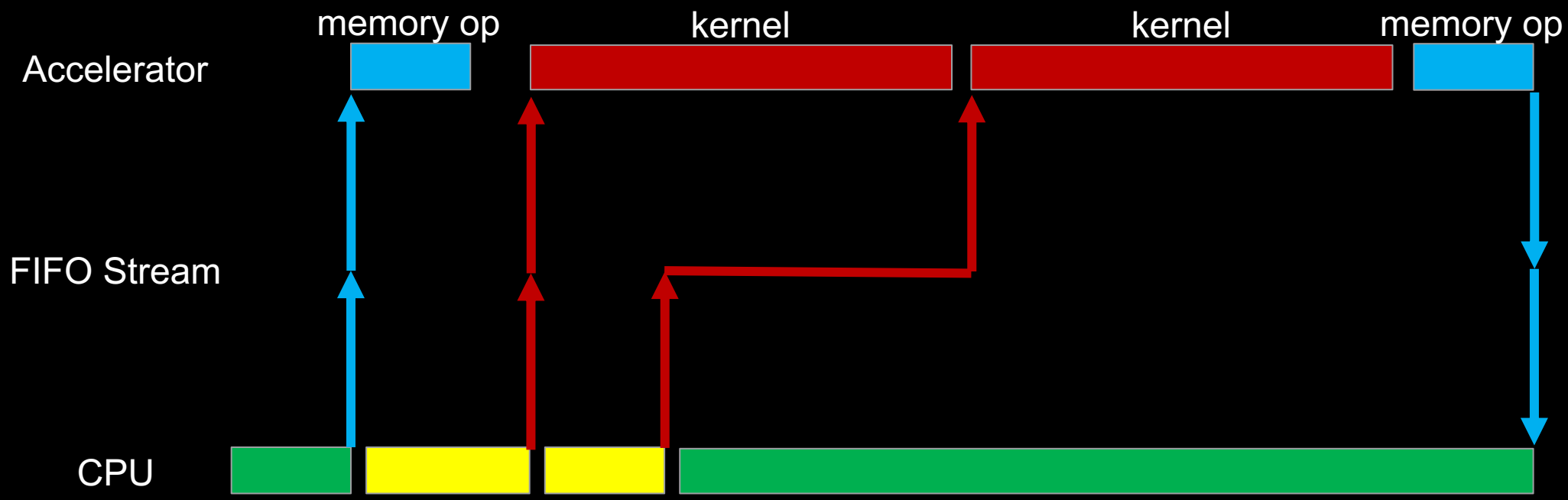
HIP – SIMT programming model

Acceleration (in a very small nutshell)

CPU

- Ensure high arithmetic intensity of kernel and ideal hardware use (e.g., occupancy, matrix extensions, …)

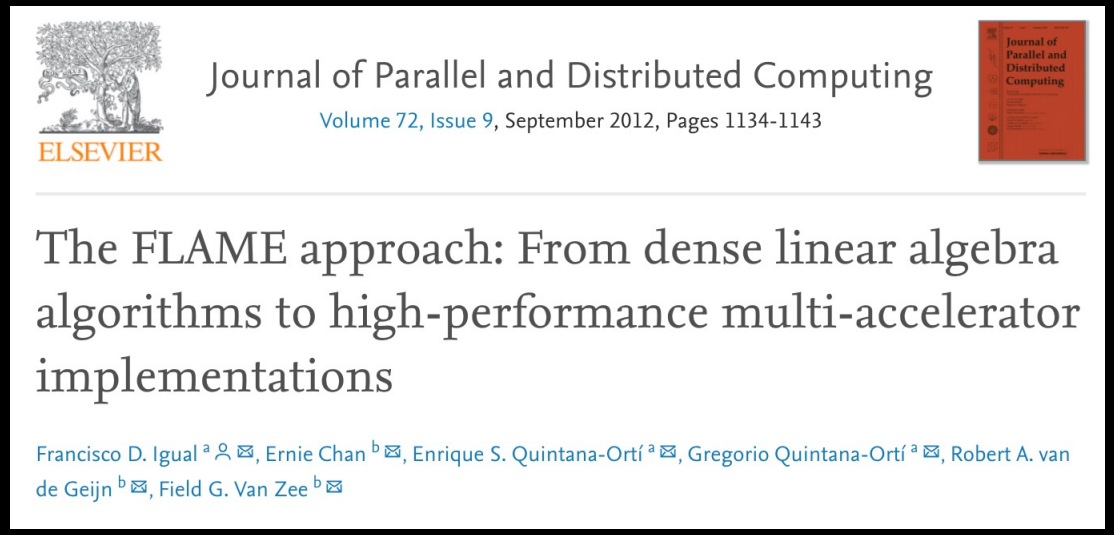
Acceleration (in a very small nutshell)



- Optimize data transfers / memory operations
- Ensure high arithmetic intensity of kernel and ideal hardware use (e.g., occupancy, matrix extensions, …)
- Execute kernels back-to-back

- Minimize host overhead
- Maximize asynchronous chain length

libFLAME/FLASH – how accelerator programming fits into classic dense LA



ELSEVIER

Journal of Parallel and Distributed Computing

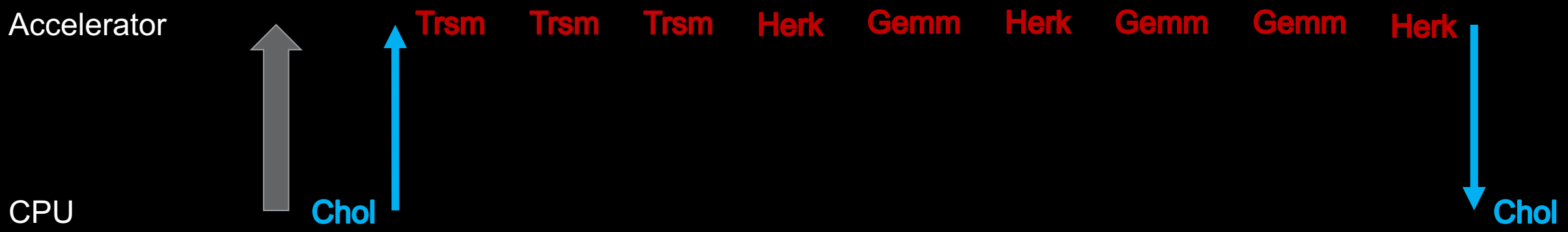
Volume 72, Issue 9, September 2012, Pages 1134-1143

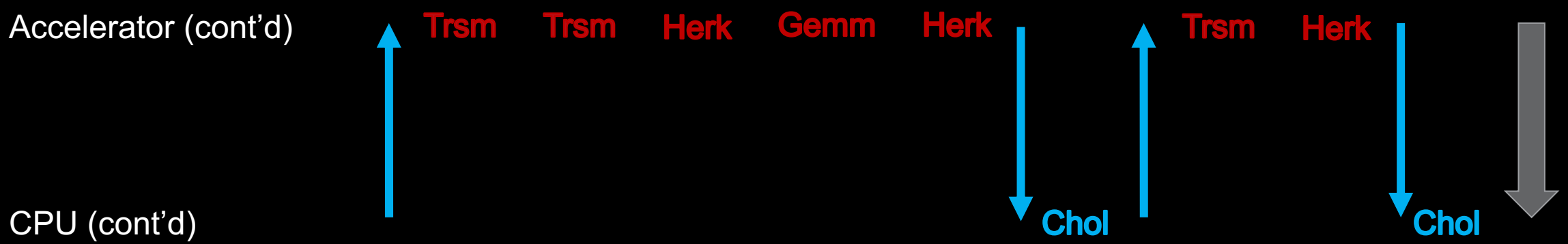
The FLAME approach: From dense linear algebra algorithms to high-performance multi-accelerator implementations

Francisco D. Igual [a], Ernie Chan [b], Enrique S. Quintana-Ortí [a], Gregorio Quintana-Ortí [a], Robert A. van de Geijn [b], Field G. Van Zee [b]

- LAPACK-style algorithms by tiles as a DAG of dense matrix tasks
- Acceleration of BLAS-level tasks on (multiple) CUDA accelerators
- Discrete memory model – tracking of ”dirty” tiles and minimal traffic to/from discrete tile pool
- Synchronous memory operations

4x4 tile Cholesky decomposition on single accelerator (non-HIP)





HIP backend features

At compile time:

- `--enable-supermatrix --enable-hip --enable-multithreading=pthread`

At runtime:

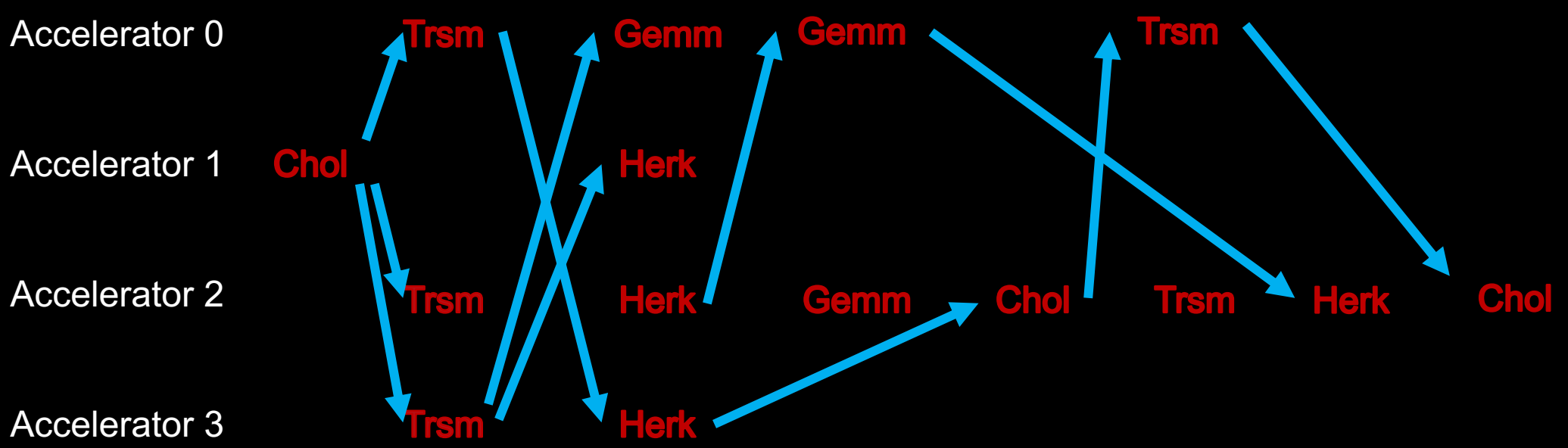
- Choice of discrete and managed memory HIP backend
- Match #SuperMatrix threads to #accelerators
- Select tile size for SuperMatrix
- Select #tiles in device cache (discrete backend)
- Reads are synchronous
- FLA_Obj host buffers are allocated pinned (discrete) or managed memory

Implemented FLAME tasks for all [s,d,c,z]:

- Through rocSOLVER:
  - Chol, Trinv, Eig_gest, LU_piv, LU_piv_copy
- Through rocBLAS:
  - Gemm, Hemm, Herk, Her2k, Symm, Syrk, Syr2k, Trmm, Trsm, Gemv, Trsv, Axpy, Scal, Scalr
- Through HIP APIs:
  - Copy, Copyr

4x4 tile Cholesky decomposition on single accelerator – discrete HIP backend

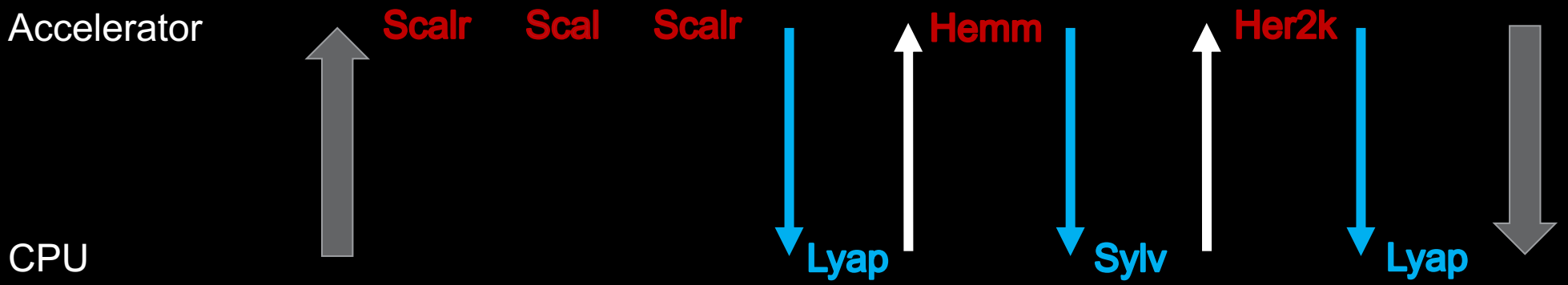
4x4 tile Cholesky decomposition on multiple accelerators – discrete HIP backend



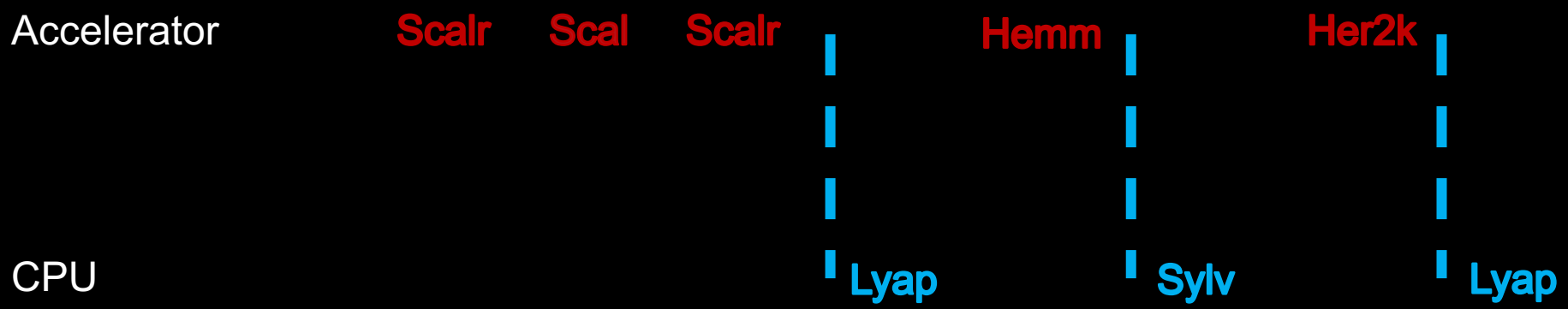
A cross-device memory operation (synchronous, via host)

2x2 tile Lyapunov equation solver using a single accelerator

Discrete memory HIP backend



Managed memory HIP backend



Work in progress

General Improvements

- Implement more FLAME operations for HIP
- `lapack2flash` wrappers to access HIP acceleration automatically

Multi-Accelerator

- Cross-device communication should not be synchronous via the host
- Use device peer2peer communication on AMD Infinity Fabric™ and dirtying of respective host blocks

Copyright and disclaimer


AMD