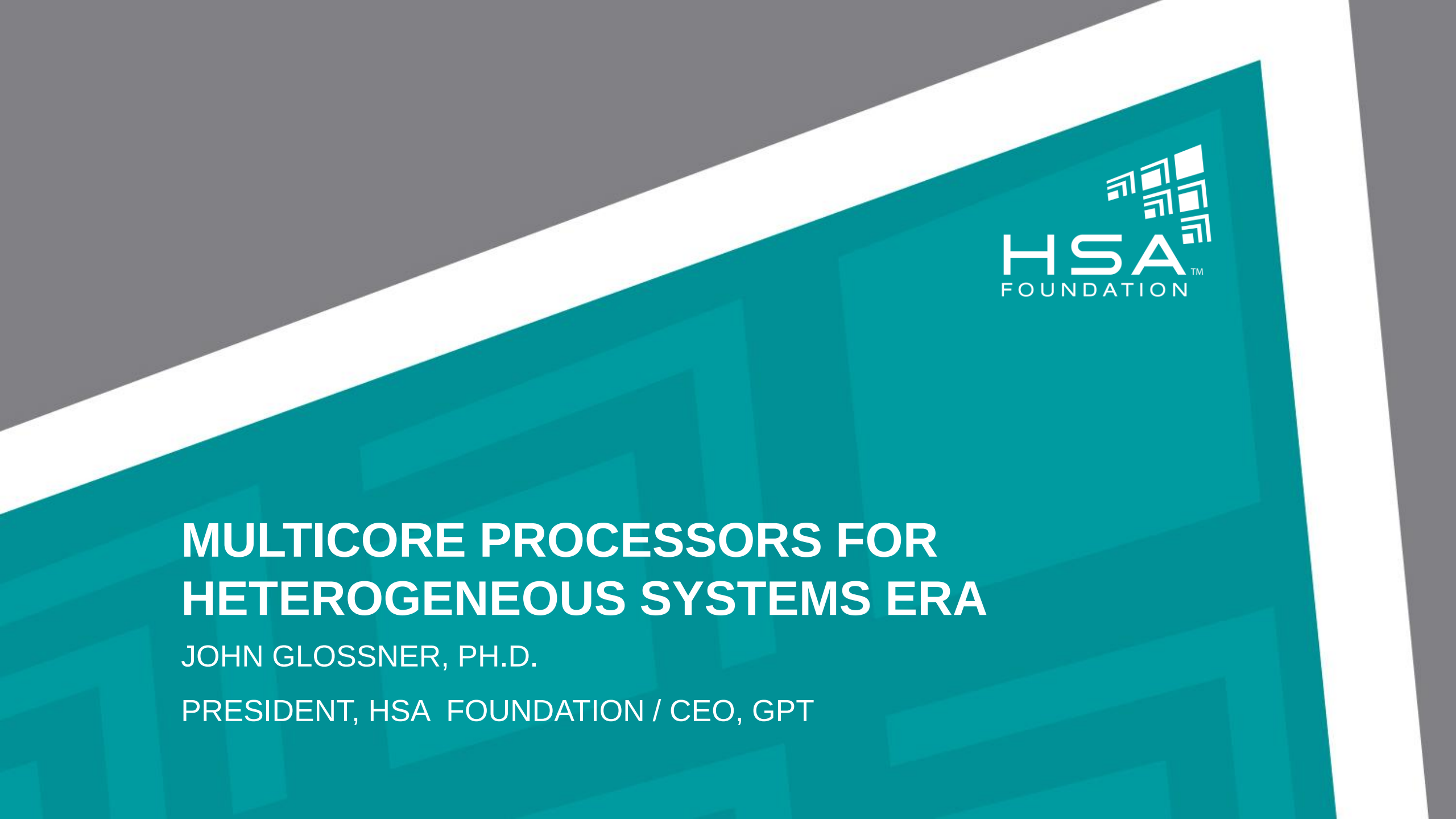


MULTICORE PROCESSORS FOR HETEROGENEOUS SYSTEMS ERA

JOHN GLOSSNER, PH.D.

PRESIDENT, HSA FOUNDATION / CEO, GPT

AGENDA

About HSA
- Founding
- Member Companies

Heterogeneous Programming Problem

HSA Solution
- Hardware
- Software
  - Infrastructure
  - HSAIL

Portable Applications Programming
- OpenCL
- C++17

Preliminary Results
- Pre- 1.0 Hardware

Products and Announcements

Conclusions

ABOUT HSA

HETEROGENEOUS SYSTEM ARCHITECTURE FOUNDATION

HSA FOUNDATION

Founded in June 2012

Developing a new platform for heterogeneous systems

Working groups established to define the platform

- V1.0 Specifications March 2015

First compatible hardware

- Carrizo - October 2015

www.hsafoundation.com

MEMBERS DRIVING HSA

HSA FOUNDATION

Founders

AMD, ARM, Imagination, MediaTek, Qualcomm, Samsung

Promoters

LG Electronics, GPT HX

Supporters

allinea, Argonne National Laboratory, UChicago Argonne LLC, Arteris, codeplay, FABRIC ENGINE, Kishonti, Lawrence Livermore National Laboratory, Linaro, MULTICORE WARE, Oak Ridge National Laboratory, Sandia National Laboratories, Stream Computing Performance Engineers, SUSE

Contributors

ANALOG DEVICES, CANONICAL, CEVA, DMP, ETRI 한국전자통신연구원 Electronics and Telecommunications Research Institute, HUAWEI, ITRI Industrial Technology Research Institute, LUXOFT, MARVELL, mobica, SONICS, SONY make.believe, Swarm64, SYNOPSYS, tensilica, Texas Instruments, TOSHIBA Leading Innovation, VIA

Academic

icsa, NTHU Programming Language Lab, NTHU System Software Lab, Northeastern, RICE Unconventional Wisdom, Tsinghua University, Tampere University of Technology, T.E.I. of Crete, The University of Mississippi, UNT University of North Texas, University of Bristol, The University of Edinburgh informatics, Illinois Computer Science, UWE Bristol

HSA – AN OPEN PLATFORM

Open Architecture, membership open to all

- HSA Programmers Reference Manual
- HSA System Architecture
- HSA Runtime

Delivered via royalty free standards

- Royalty Free IP, Specifications and APIs

ISA agnostic

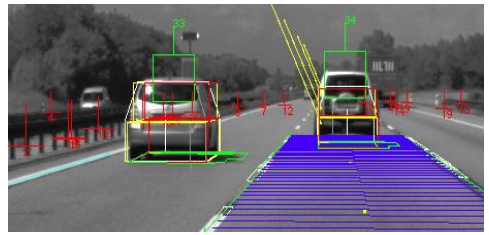
- CPU, GPU, DSP, FPGA, etc.

APPLICATIONS THAT RUN FASTER AND AT LOWER POWER

Always on, visually aware devices will offer greater capability in a lower power budget, scaling with every advance in app processing

Mobile and tablet devices will use the CPU, GPU and DSP working seamlessly together for content creation, gaming and more

Intelligent cloud video analytics will be more efficient, and make best use of every server upgrade



Sophisticated ADAS real-time analytics will be easier to develop, adapt to any platform, and be more robust

HSA architecturally integrates the accelerators in today's complex SoCs to be easily and efficiently utilized by application developers

THE PROBLEM

HETEROGENEOUS APPLICATION DEVELOPMENT

WHAT'S THE PROBLEM?

Heterogeneous processors are widely available

Huge compute capability

- Acceleration Units (GPU, DSP, FPGA)
- CPU Cluster-based computer

Coherency

- Established in high-end
- Migrating to mainstream mobile and consumer

BUT…

Heterogeneous programming models not standardized

Multi-core/device applications difficult to optimize or scale

Non-portable application developer ecosystems

HSAF brings compute app abstraction to heterogeneous platforms

HSAF HARDWARE CONTRIBUTIONS

JIM MCGREGOR, TIRIAS RESEARCH

…HSAF has had a profound impact on hardware architectures

… even Intel‘s

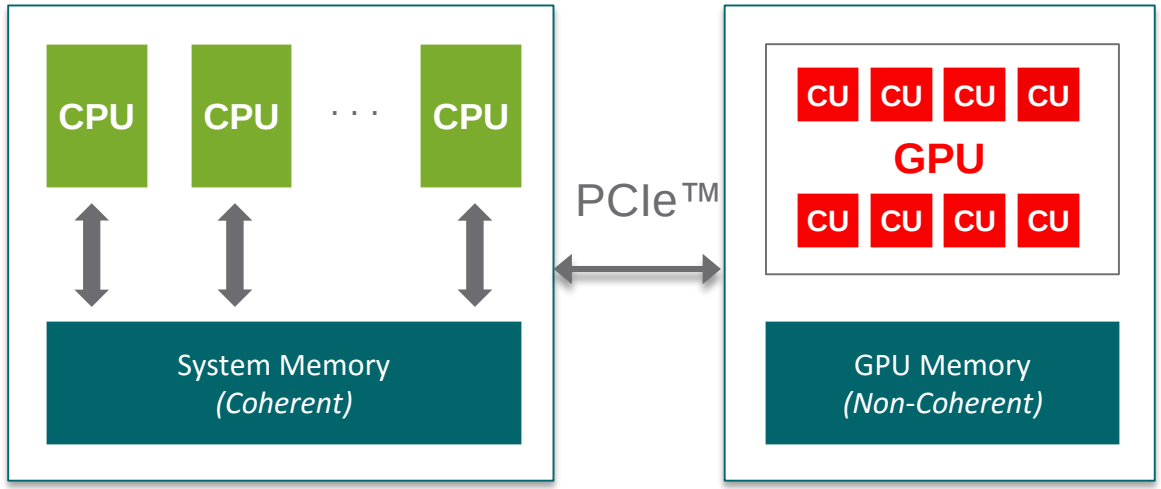
GEN0: LEGACY COMPUTE

Memory challenges

- Multiple memory pools
- Multiple address spaces
- High overhead OS dispatch
- Data copies across PCIe

New programming languages

- Dual source development
- Proprietary environments
- Expert programmers only



DSPs/FPGAs have the same issue

GEN1: ON CHIP INTEGRATION

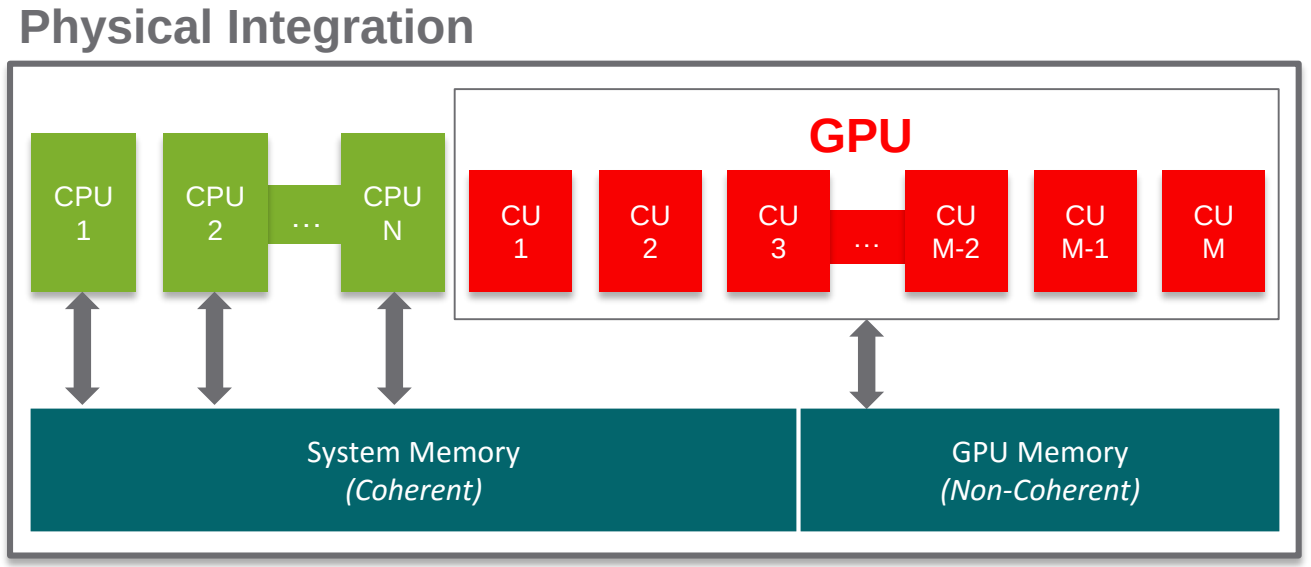
Physical Integration

- Good first step

Some copies gone

- Two memory pools remain
- Still queue through the OS

Still requires expert programmers



DSPs/FPGAs have the same issue

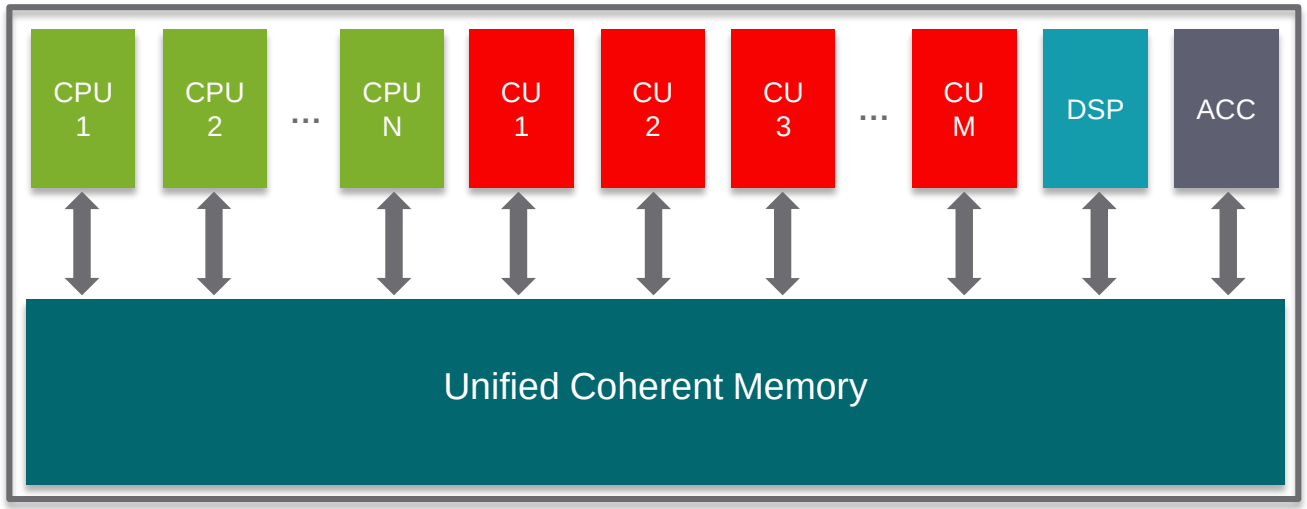
GEN2: AN HSA ENABLED SOC

Unified Coherent Memory

- data sharing across all processors

Processors architected to operate cooperatively

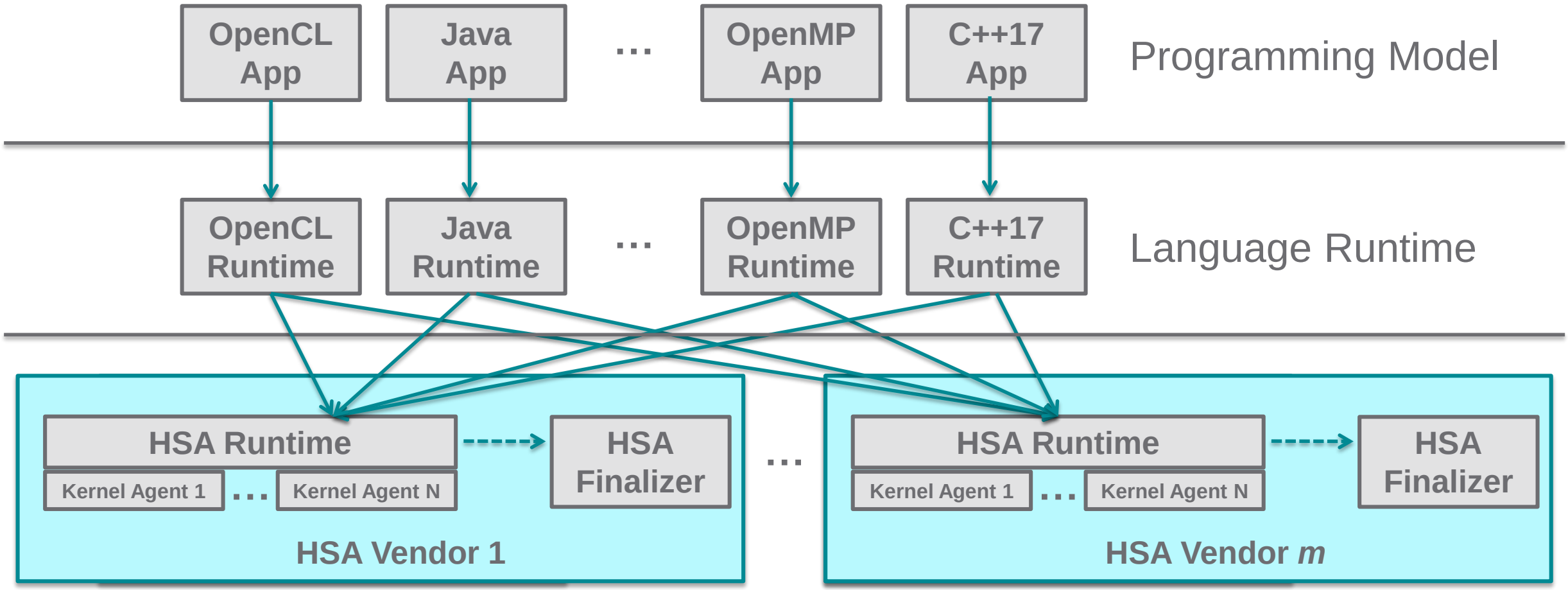
Applications enabled to run on different processors at different times



HSAF SOFTWARE

INFRASTRUCTURE

S/W ARCHITECTURE WITH HSA RUNTIME



HSA PROGRAMMING MODEL

Single source

- Host and agent code side-by-side in same source file
- Written in same programming language

Single unified coherent address space

- Freely share pointers between host and agent
- Similar memory model as multi-core CPU

Parallel regions identified with existing language syntax

- Typically same syntax used for multi-core CPU

HSAIL is the compiler IR that supports these programming models

Supported Languages

- OpenCL 2+, OpenMP, C++AMP, Java
- Future: C++17

HSA Core Runtime API

- Thin user-mode API
- High performance dispatch
  - Across multiple vendors

Functions

- Initialization and Shut Down
- Notifications (Synchronous/Asynchronous)
- Agent Information
- Signals and Synchronization (Memory-Based)
- Queues and Architected Dispatch
- Memory Management

HSA MEMORY MODEL

Compatible with C++, Java, OpenCL and .NET Memory Models

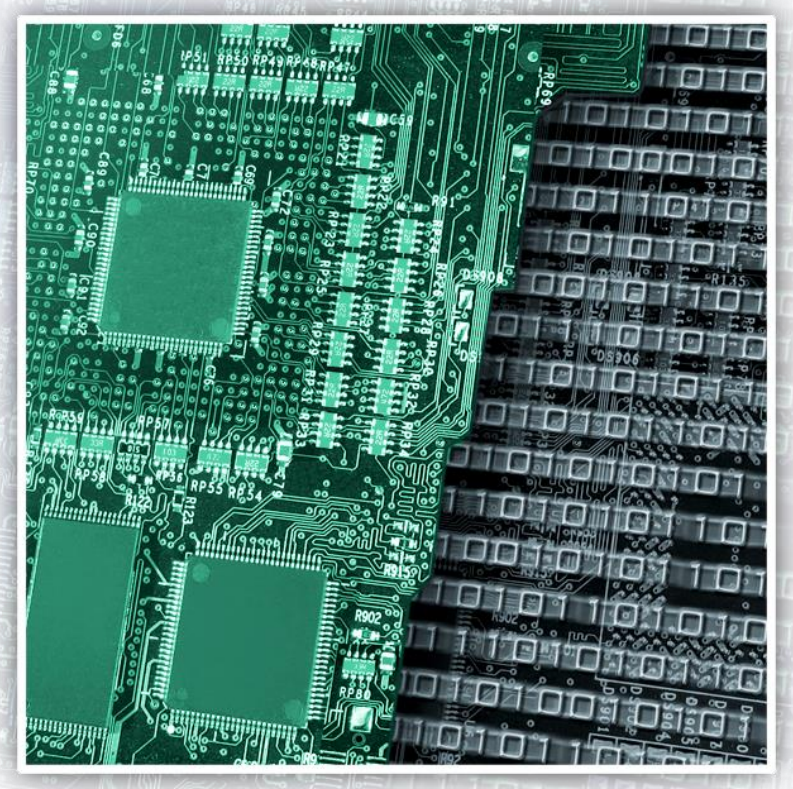
Defines visibility ordering between all threads

Relaxed consistency memory model

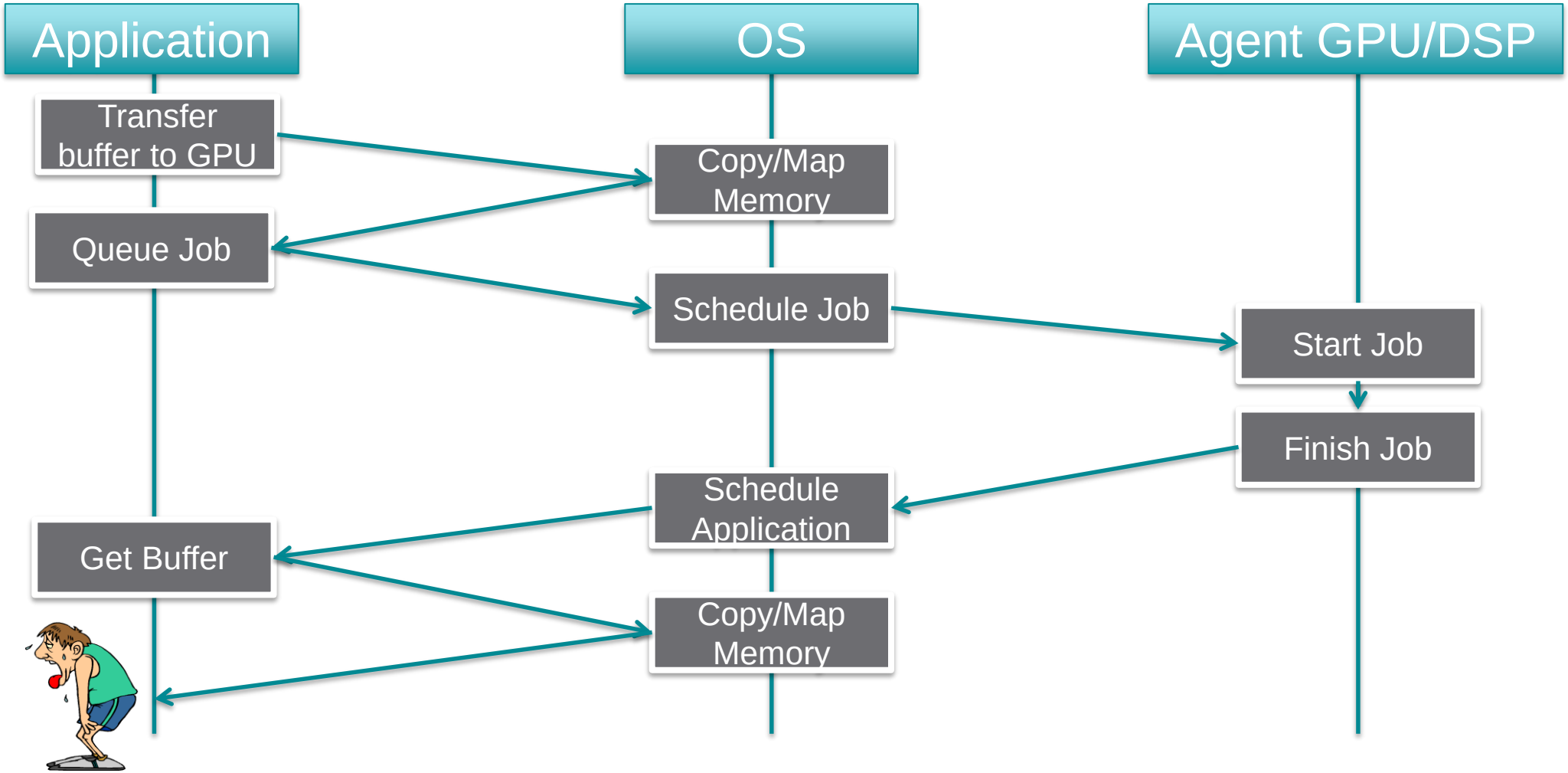
- Parallel compute performance

Visibility controlled by:

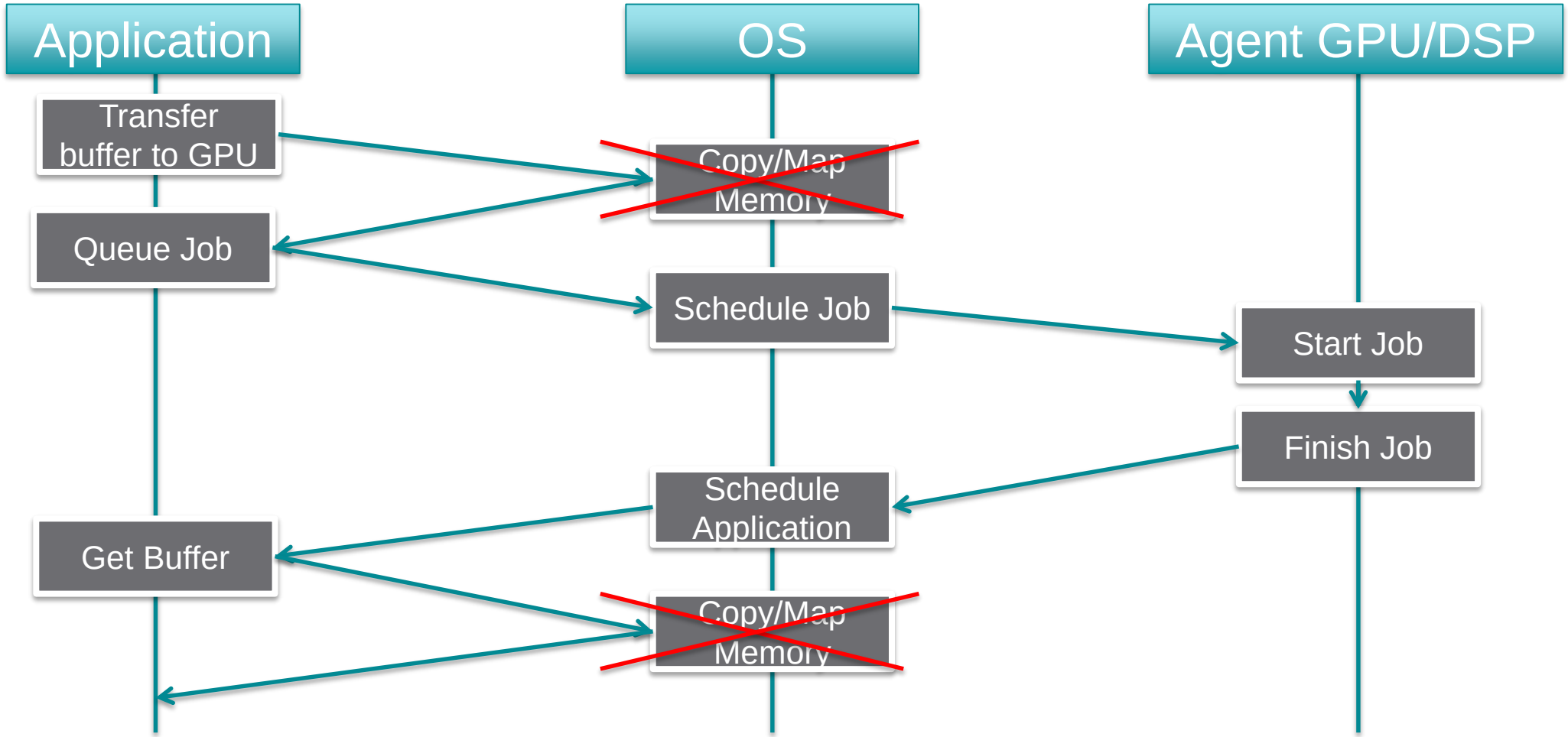
- Load.Acquire
- Store.Release
- Fences



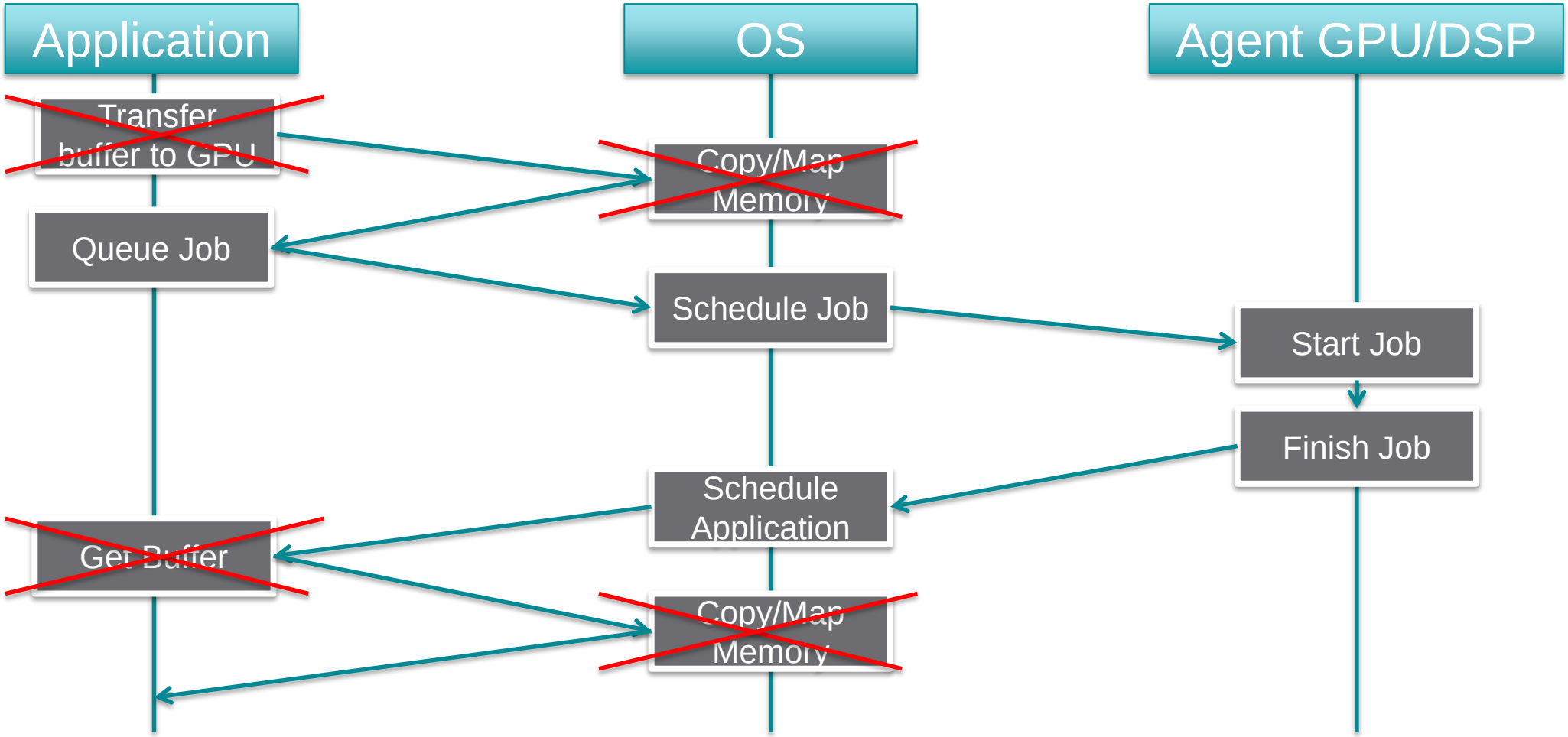
MOTIVATION (TODAY'S PICTURE)



WITH SHARED VIRTUAL MEMORY



WITH COHERENT CACHE MEMORY



SIGNALS

HSA agents support signaling

- creation/destruction using runtime APIs

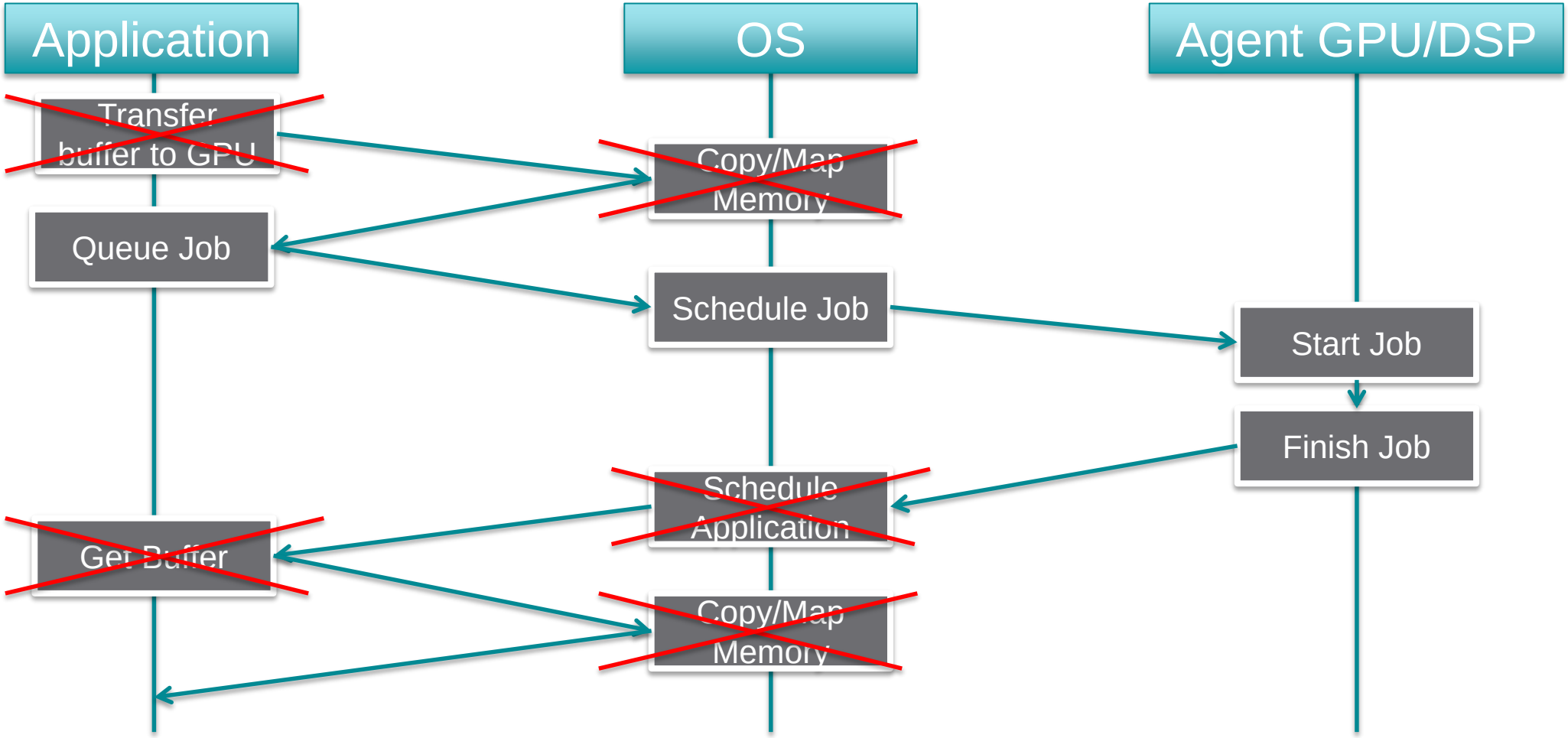
Any Agent can directly access signals

- wake up HSA agents waiting upon the object
- Query current object
- Wait on the current object
  - Using conditions

Advantages

- Asynchronous events between agents
  - Doesn't require CPU
- Common idiom for work offload
- Low power waiting

WITH SIGNALING



HSA QUEUING MODEL

User mode queuing

- Low latency dispatch
- Application dispatches directly
- No OS or driver required

Architected Queuing Layer (AQL)

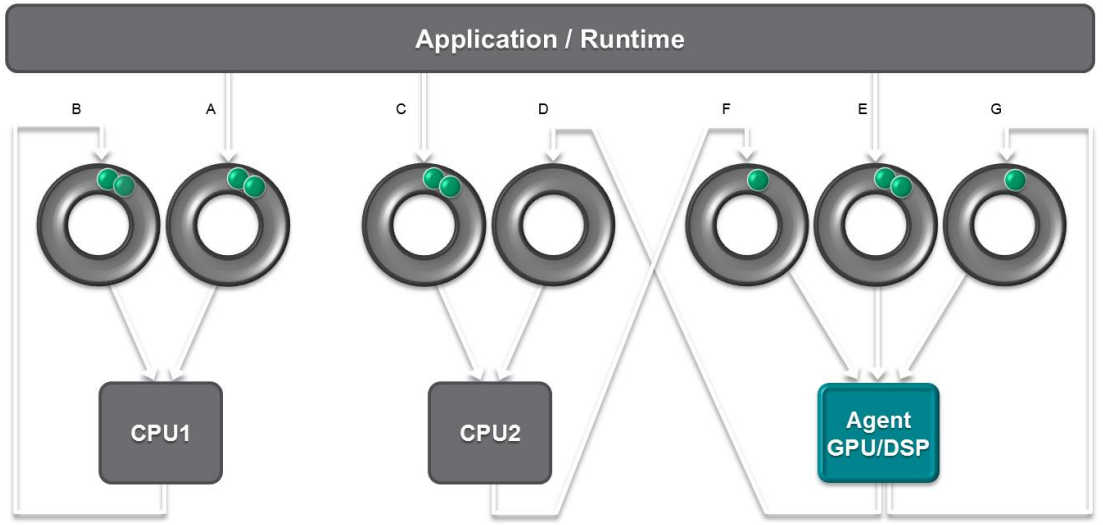
- Single compute dispatch path for all hardware
- No driver translation, direct to hardware
- Standard across vendors!
- Guaranteed backward compatibility

Allows for dispatch to queue from any agent

- CPU or GPU or DSP or FPGA, etc.

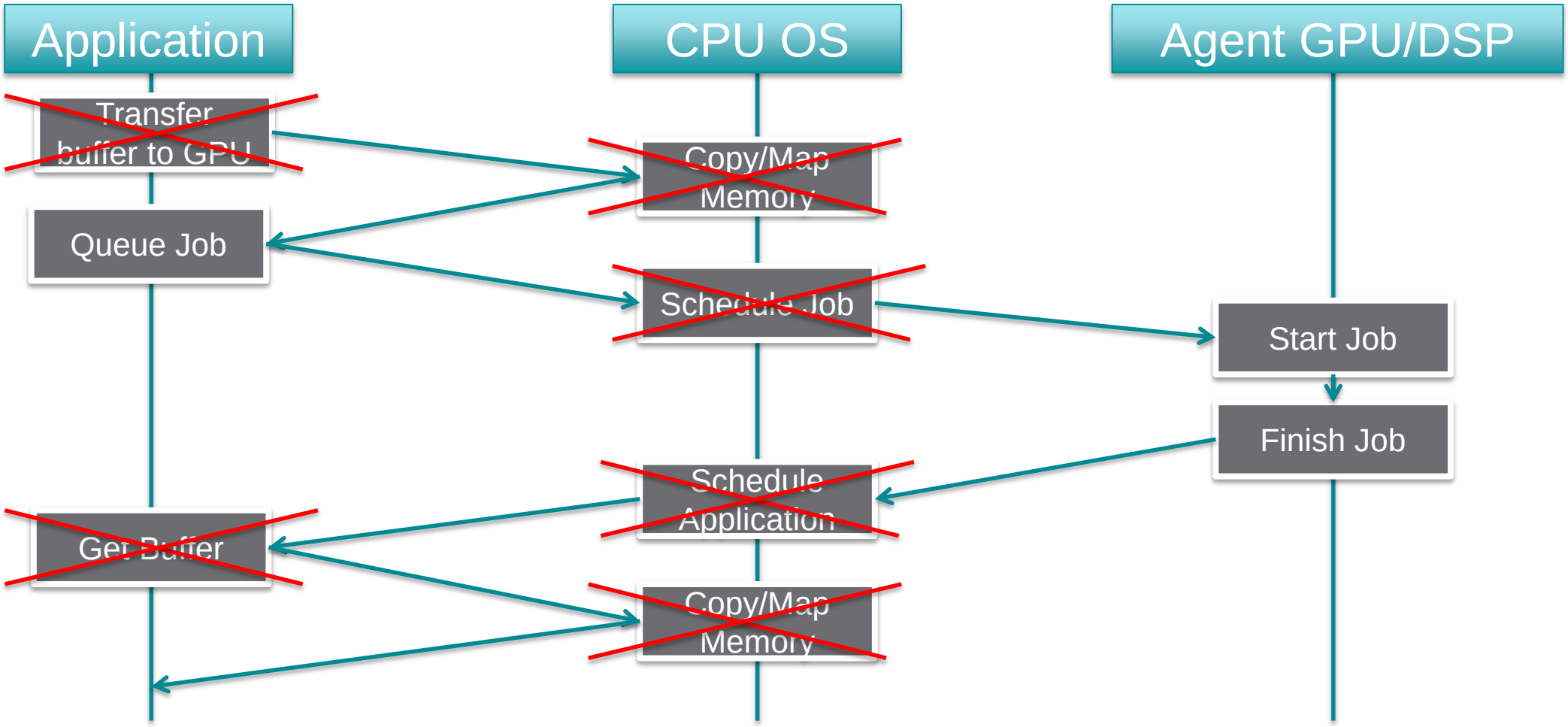
Agent self enqueue enables

- Recursion, Tree traversal, Wavefront reforming

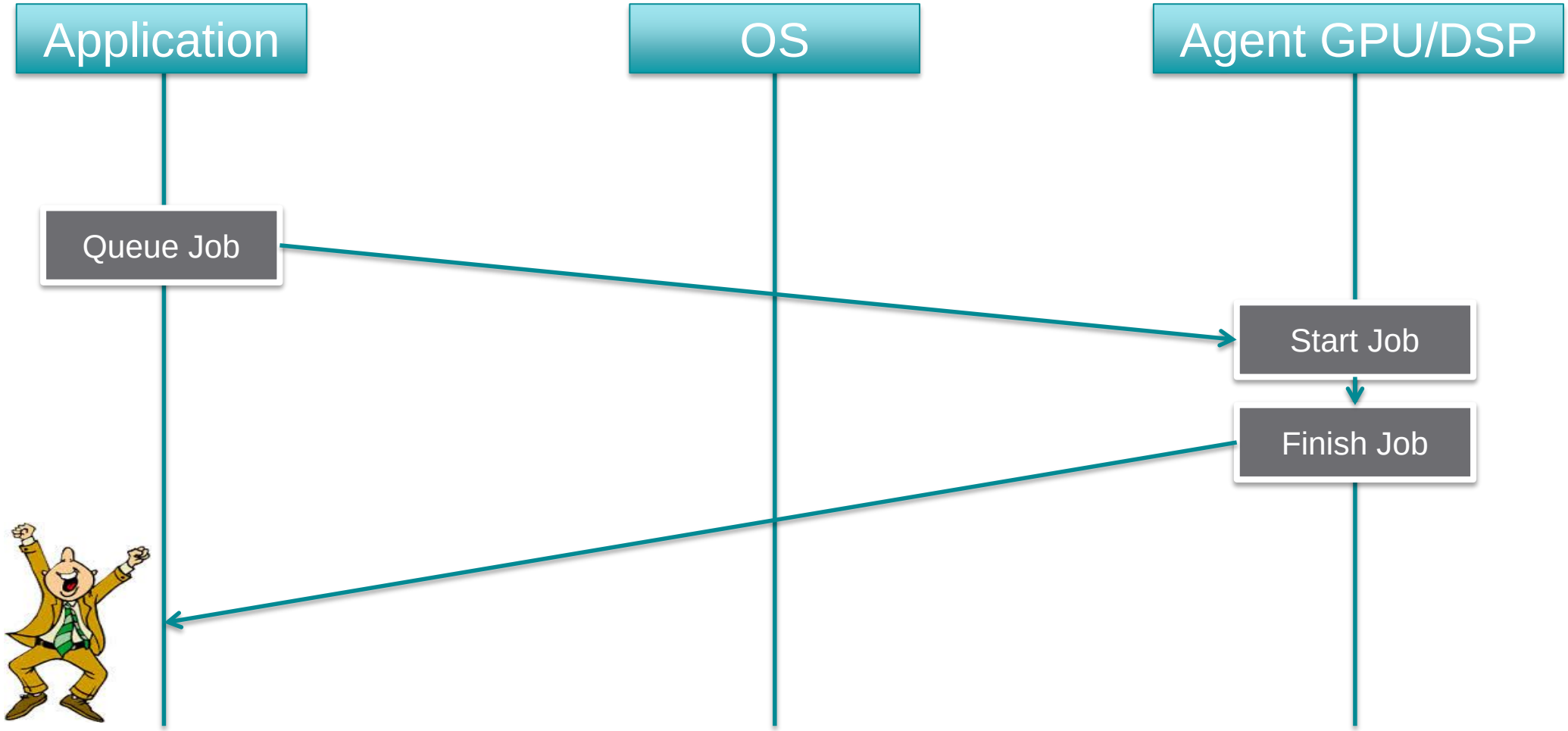


Requires coherency and shared virtual memory

WITH USER MODE QUEUING



FINAL PICTURE: SVM + CACHE COHERENCY + SIGNALS + USER MODE QUEUES



HSA INTERMEDIATE LANGUAGE (HSAIL)

BYTECODE FOR HETEROGENEOUS SYSTEMS

THE PORTABILITY CHALLENGE

CPU ISAs – Backwards Compatible

- ISA innovations added incrementally (ie NEON, AVX, etc)
  - ISA retains backwards-compatibility with previous generation
- HSA instruction-set architectures: ARM, MIPS, and x86

Kernel Agent ISAs – No Backwards Compatibility

- GPU, DSP, DNN, Image Signal Processor, Custom Accelerators, etc.
- Massive diversity of architectures in the market
  - Each vendor has own ISA - and often several in market at same time
- Compatibility via APIs (OpenGL, DirectX, OpenCV)

HSA INTERMEDIATE LAYER — HSAIL

Virtual ISA for parallel programs

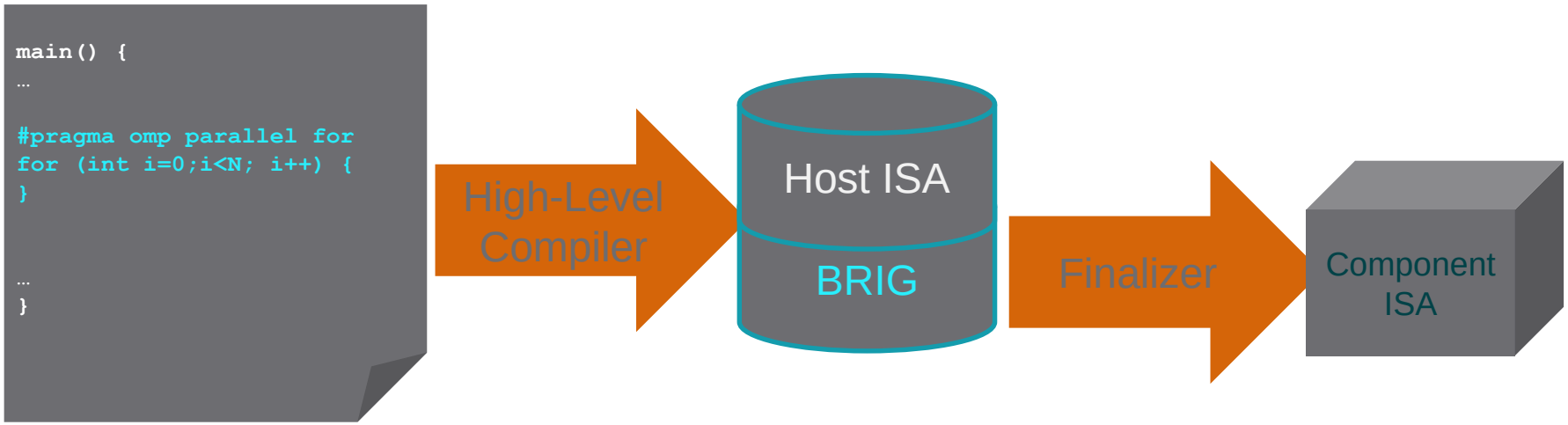
- Finalized to native ISA by a compiler
  - Dynamic or Offline
- ISA independent by design

Multiple HLL Support

- Exceptions, virtual functions, etc.
- Java, C++, OpenMP, C++, Python, etc

Explicitly parallel

- Designed for data parallel programming



PORTABLE APPLICATIONS PROGRAMMING

FROM OPENCL TO C++17

OPENCL – ALL OPEN SOURCE TOOLS

AMD CLOC or TuT POCL

- Generates HSAIL from OpenCL

GPT/Parmance gccBrig

- A BRIG language front-end to GCC
  - BRIG: Binary Representation of HSAIL
  - Translated to GCC's tree intermediate representation
- Optimization by GCC
  - Including vectorization/SIMD optimizations

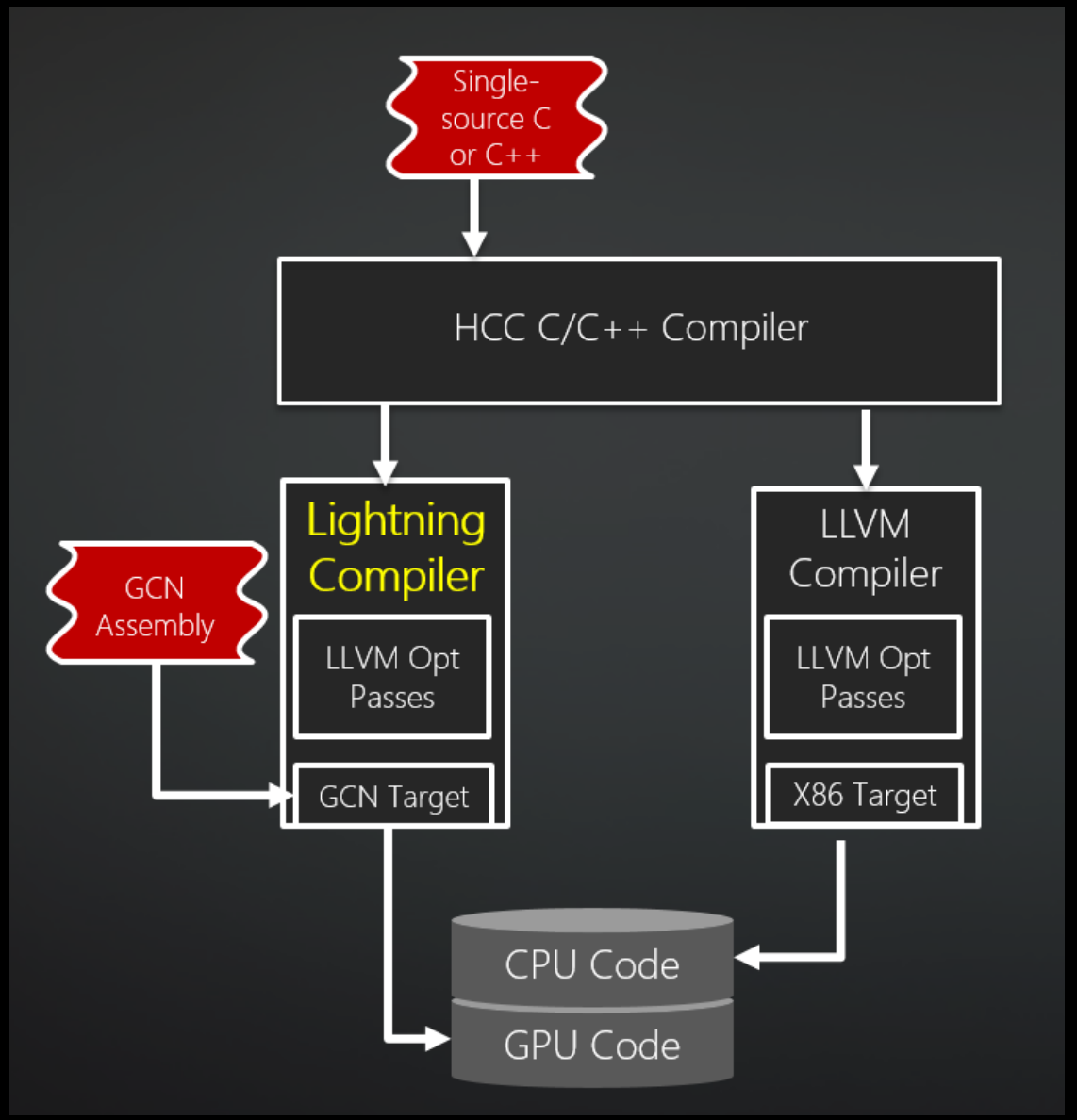
Benefits

- Allows use of GCC for finalization
- Vendor independent
  - No need to know proprietary Instruction Set Architecture
- CPU/VLIW/MIMD HSA kernel agent support

HCC - HETEROGENEOUS COMPUTE COMPILER

C++ & C COMPILER

- Compiler Architecture
  - Single-source : Host and device code in the same source file
  - Fully Open Sourced Compiler using CLANG/LLVM
- Expressing Parallelism with HCC
  - C++ parallel_for_each + lambda
  - Parallel STL
- HCC generates both CPU and GPU code
  - Traditional CPU programs can be compiled with HCC
  - Heterogeneous programs
    - Compiles the host code for CPU
    - Compiles the kernel/parallel region for the GPU
- Programmer Optimizable
  - Control, pre-fetch, discard data movement
  - Run asynchronous compute kernels
  - Access GPU scratchpad memories



CUDA PORTING WITH HCC C++

AMD

- Challenge:
  - Many existing GPU-accelerated apps use proprietary CUDA language and infrastructure
- Strategy:
  - Make it easy to port from CUDA to a *common* C or C++ programming model
  - Common = resulting code runs through either CUDA NVCC or HCC C++ compiler
  - Can use best development tools on either Nvidia or AMD platform
  - *Provide customers with choice in hardware and development tools*
- Implementation
  - HIP = "**H**eterogeneous-compute **I**nterface for **P**ortability"
  - Header maps hip* calls to CUDA RT or HSA RT
  - Strong subset of CUDA RT functionality, focus on most commonly used functions
  - Some HCC support to make porting easier
  - Can use #ifdef for tricky cases and performance tuning

PARALLELISM IN C++

AMD

- C++11 introduced threads, memory model, async, thread-local-storage
  - Before this, C++ basically a sequential single-stream language
- Next steps
  - Stated Goal : by 2020, express all forms of intra-node parallelism directly in C++
  - SIMD, multi-core CPU, GPU (not much FPGA)
- Highlights:
  - Parallel STL – extend template Algorithms (sort, scan, reduce) so they can be run in parallel
  - Executors – control where and how a function executes.
    - Default policies for sequential, parallel, vector, thread-pool
    - Can be customized and extended.
  - SIMD Parallelism Types and Operations
  - Concurrency TS – specify dependence between commands using continuations [then()] and join points [when()]
  - “for_loop” – OpenMP-like syntax for C++, allows loop to see position
- http://open-std.org/JTC1/SC22/WG21/docs/papers/2016/#mailing2016-02

ACCELERATED PYTHON

AMD

Numba speed up your applications with high performance functions written directly in Python

- Can compile Python code to run on GPU
  - Built-in functions for group, grid index, barriers provided in Python
  - Well-defined subset of Python is supported
  - GPU version will use open-source LLVM compiler provided by AMD
- Rich set of options to optimize for APU or discrete GPU
  - Async execution, specify group size, use shared memory
  - Data transfer can be performed implicitly based on kernel arguments

ANACONDA on Boltzmann a New Class of Performance to Python

PERFORMANCE RESULTS

GEN1: FIR & AES

FIR is a memory-intensive streaming workload

AES is a compute-intensive streaming workload

CL12 – cl_mem buffer

- Copy to/from the device

CL20 – SVM buffer – Coarse Grain Sync

- Copy to/from SVM
- Data copy cannot be avoided, since the space for SVM is limited

HSA – Unified Memory Space – Fine Grained Sync

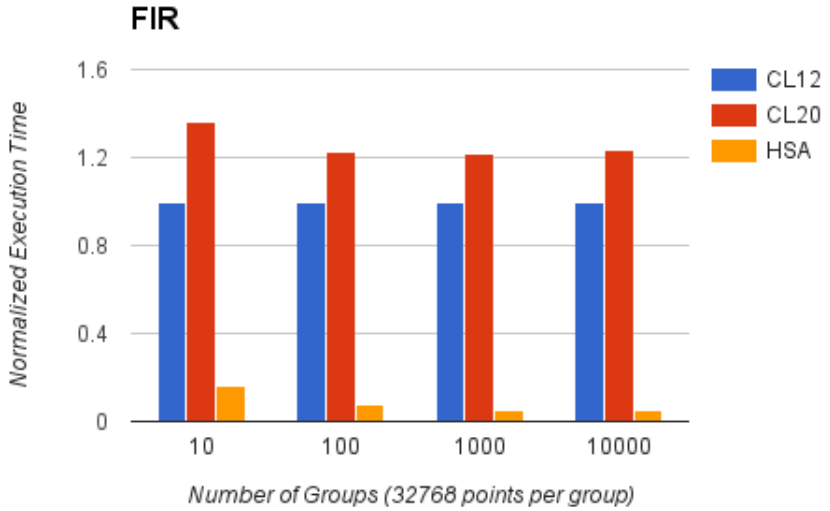
- Regular pointer
- No explicit copy

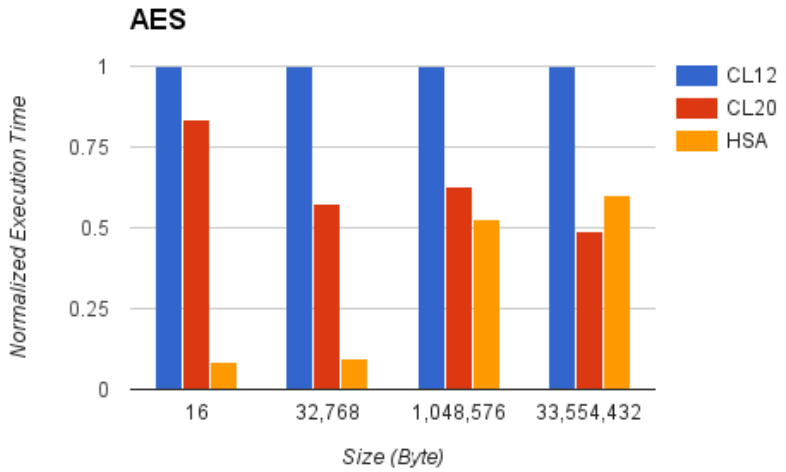
Results

- HSA compute abstraction
- NO performance penalty

Not all algorithms run faster

- Measured on Kaveri (A pre-HSA 1.0 device)
- Limited Coherent throughput





Northeastern

Saoni Mukherjee, Yifan Sun, Paul Blinzer, Amir Kavyan Ziabari, David Kaeli,*A Comprehensive Performance Analysis of HSA and OpenCL 2.0,* **Proceedings of the 2016 International Symposium on Program Analysis and System Software,** April 2016, to appear.

BLACK-SCHOLES

C++ on HSA

- Matches or outperforms OpenCL

Course Grained SVM

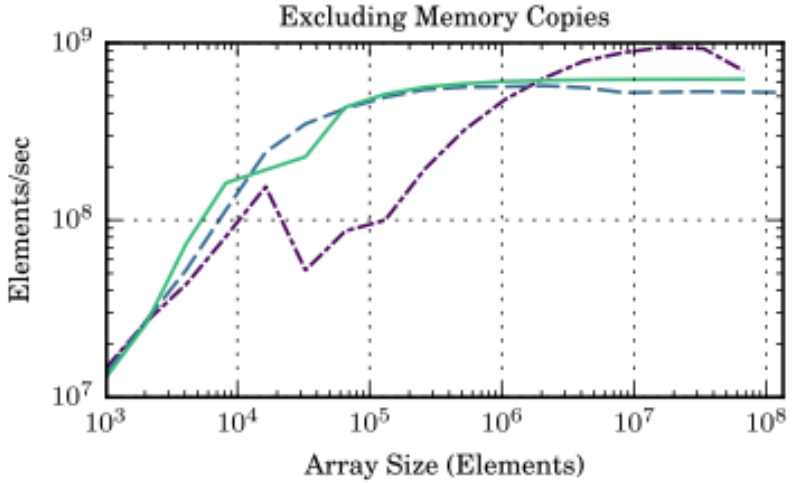
- Matches OpenCL buffers for bandwidth
- More predictable performance

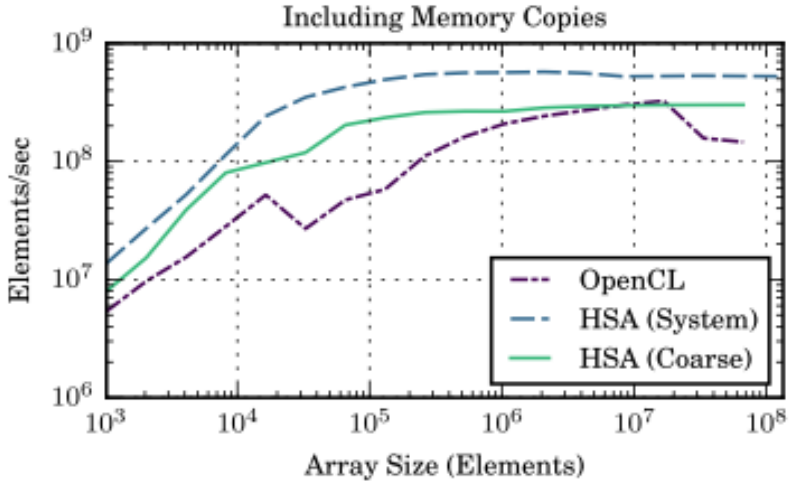
Fine Grained SVM

- Faster kernel dispatch
- Larger allocations
- Shared data structure

Results

- HSA compute abstraction
- NO performance penalty

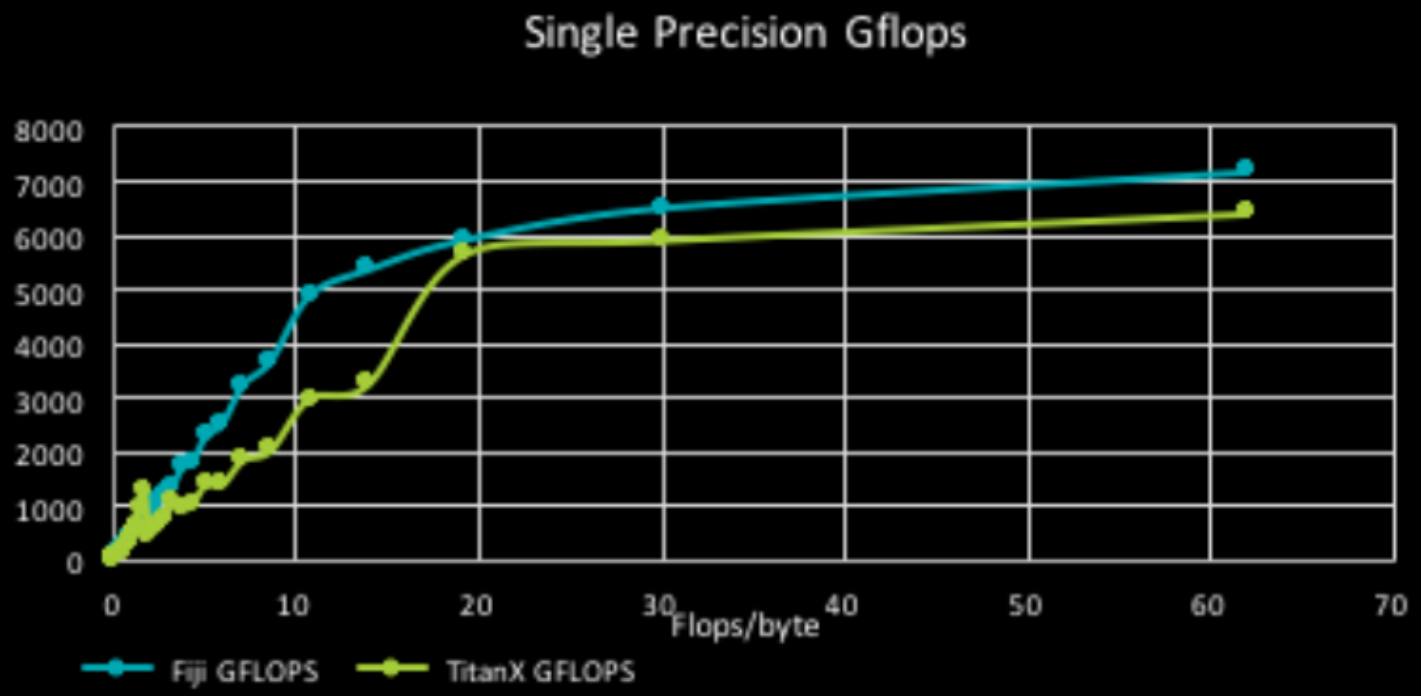




Source: Ralph Potter – Codeplay. Presentation made to SG14 C++ Workgroup

CUDA RESULTS USING HIP

- No manual intervention
- Direct Conversion from CUDA
- Runs on any HSA system



HSA PRODUCT UPDATES

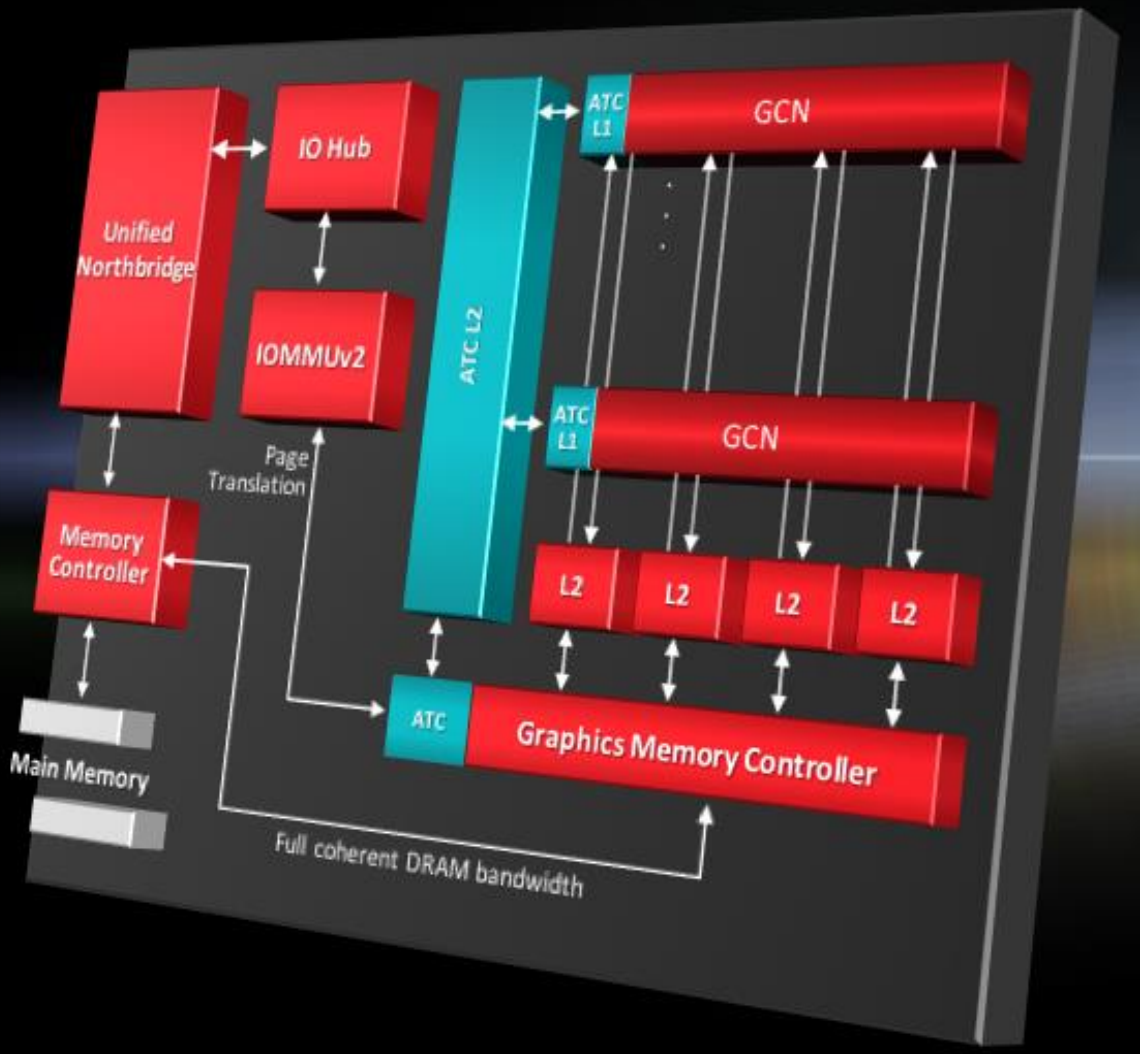
FROM HSA FOUNDATION MEMBER COMPANIES

“CARRIZO” IS AMD’S SECOND APU PRODUCT WITH HSA FEATURES

FULLY HSA ENABLED

HSA ACCELERATION FEATURES

- Address Translation Cache (ATC) hierarchy
  - Improves virtual memory address translation throughput for “pointer is a pointer” data sharing between CPU and GPU
- Full hardware cache coherence at maximum DRAM bandwidth between GPU and CPU caches
- Support for Wavefront and Compute task pre-emption and context switching
  - Improves work scheduling efficiency
- HSA QoS scheduling support



ARM AND HSA

ARM, as a founder member, has been committed to the HSAF since launch

- Actively contributes to the HSA specifications and working groups
- Is committed to the continued development of this important standard

ARM customer base is showing increasing interest in HSA features for their next generation SoCs

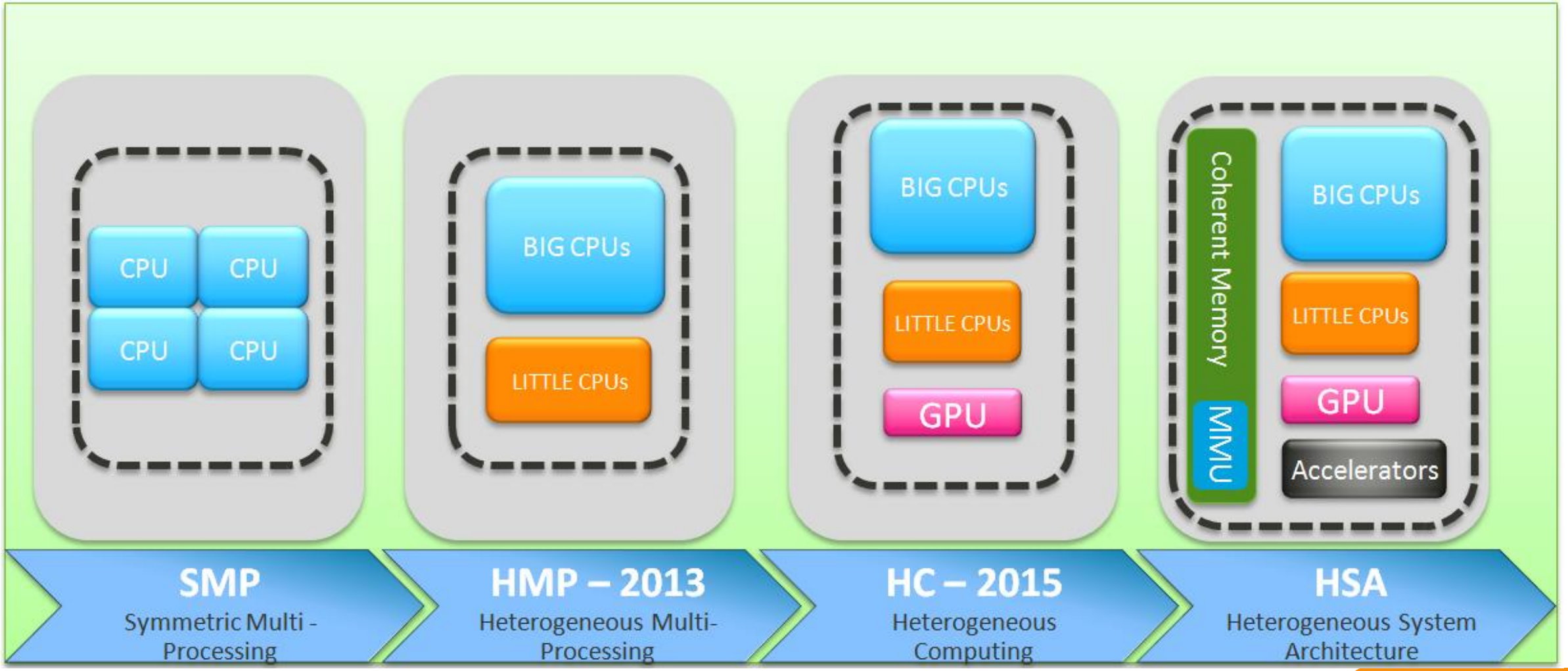
ARM customers can already build real heterogeneous systems based on, for example:

- ARM Cortex-A72 high performance application processor
- ARM Mali-T880 compute enabled GPU
- ARM CoreLink CCI-500 cache coherent interconnect
- ARM CoreLink CCN cache coherent network family

ARM is actively developing next generation processor and interconnect IP to extend the system capabilities aligned with HSA standards including:

- Full memory coherency
- Shared Virtual Memory

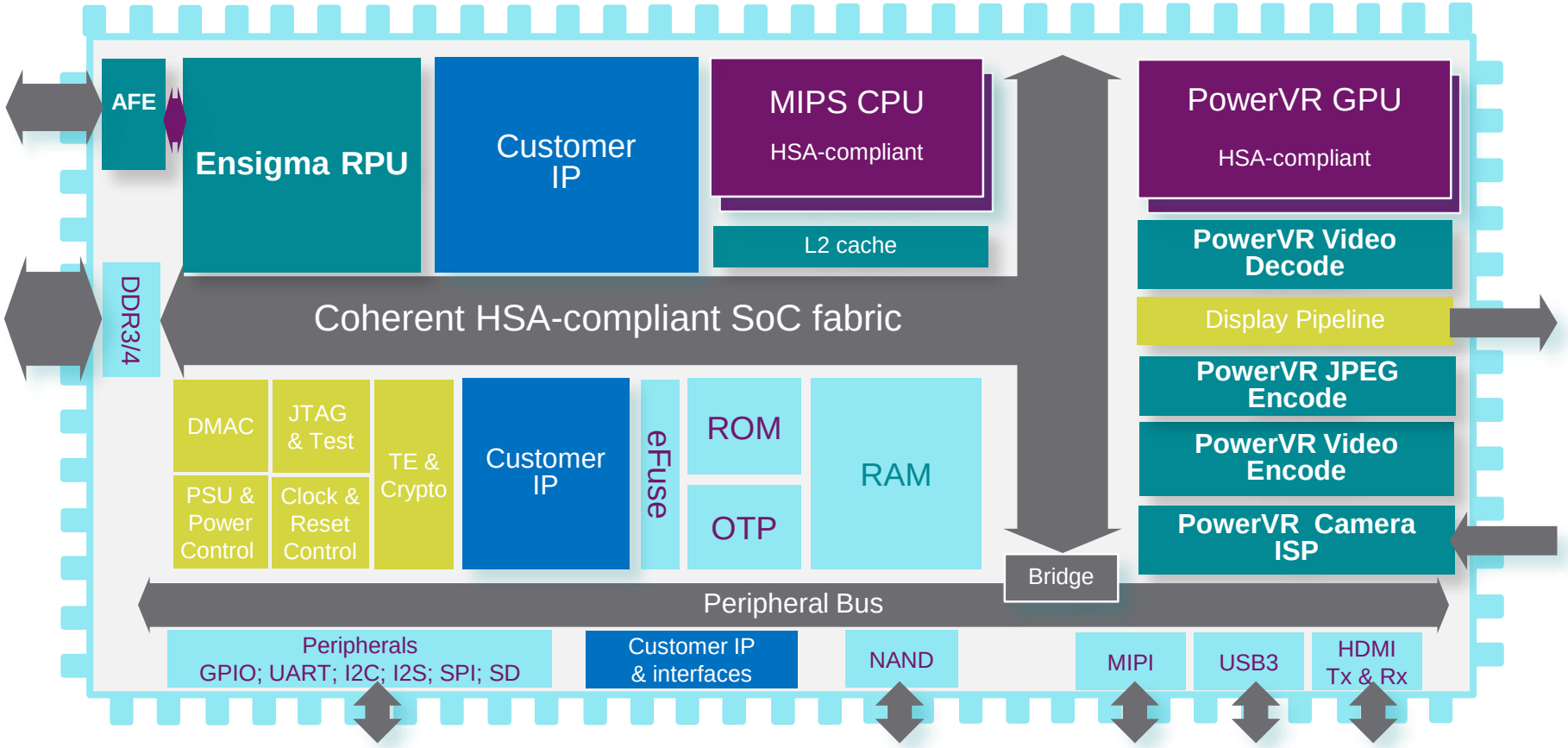
HSA ROADMAP AT MEDIATEK



10/5/2015

IMAGINATION HSA COMPLIANT IP COMING SOON

Imagination Smart Vision IP Platform



We will be rolling out:

- HSA across all MIPS I-class and P-class CPUs
- HSA across all PowerVR GPUs
- HSA compliant fabric solutions

CONCLUSIONS

THE HSA TECHNICAL SUMMARY

MAKE HETEROGENEOUS PROGRAMMING MUCH EASIER

1	Single source programming	Single tool chain
2	Any programming language	C++, Python, JavaScript, …
3	Eliminate data copies	Performance!
4	Common address space	A pointer is a pointer
5	Standardized command submission to Agents (GPU / DSP)	A common dispatch language
6	Eliminate software layers between application and hardware	Efficient
7	ISA agnostic for CPU, GPU, DSP, and more	x86, ARM, MIPS, PowerVR, Mali, Adreno, GPT, …
8	Open source software stack	Open Access!

High performance

Low power

Extensible to other accelerators on the SoC

SUMMARY

2012 goal of changing chip H/W architecture achieved

- Cache Coherent shared virtual memory

2014-2015 S/W architecture to support H/W

- March 2015 V1.0 specs
- Programmed in any language (C++, Python, OpenCL)

2016 H/W platforms arriving

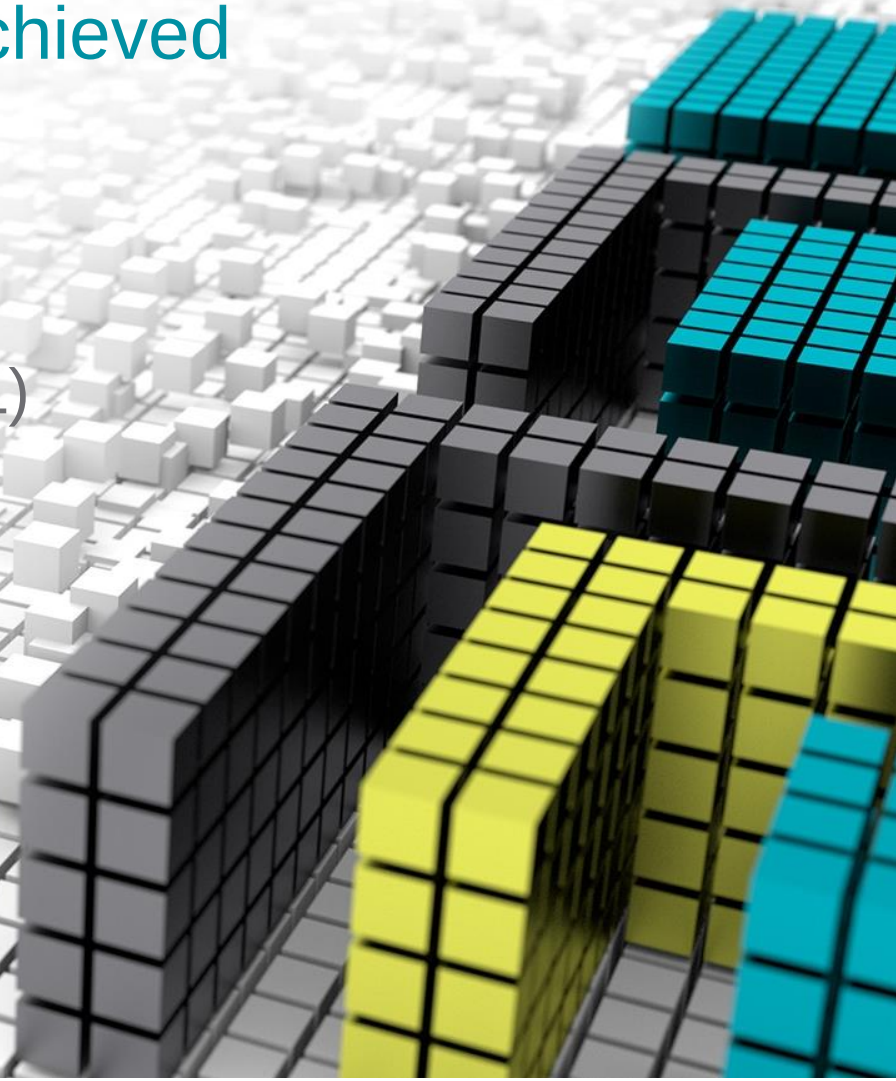
- AMD's Carrizo (Dell, Asus, Lenovo)
- Others to follow

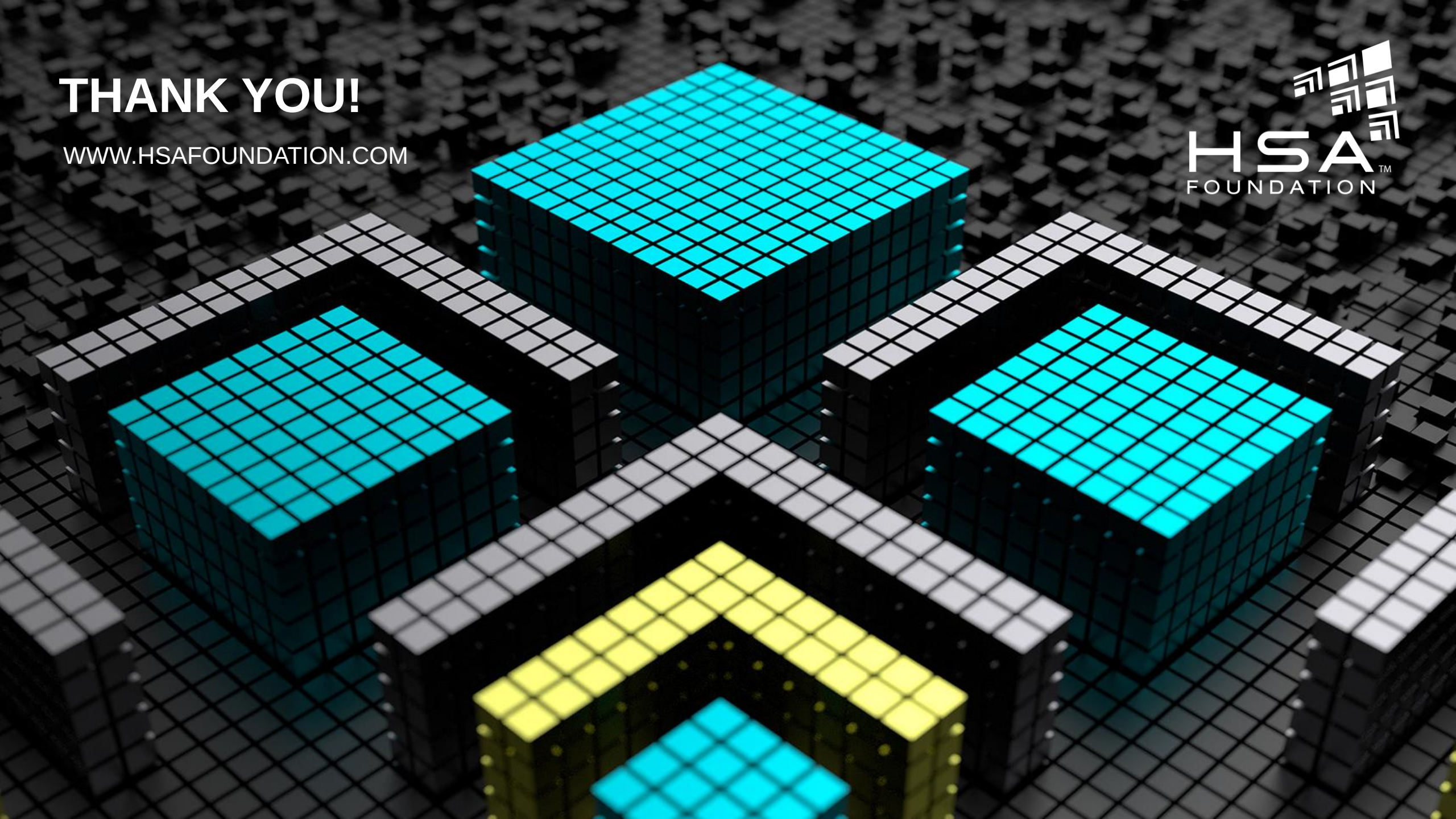
Excellent performance results

- No performance penalty for HSA abstraction

Multivendor support

- Coming soon




THANK YOU!
WWW.HSAFOUNDATION.COM
HSA™
FOUNDATION