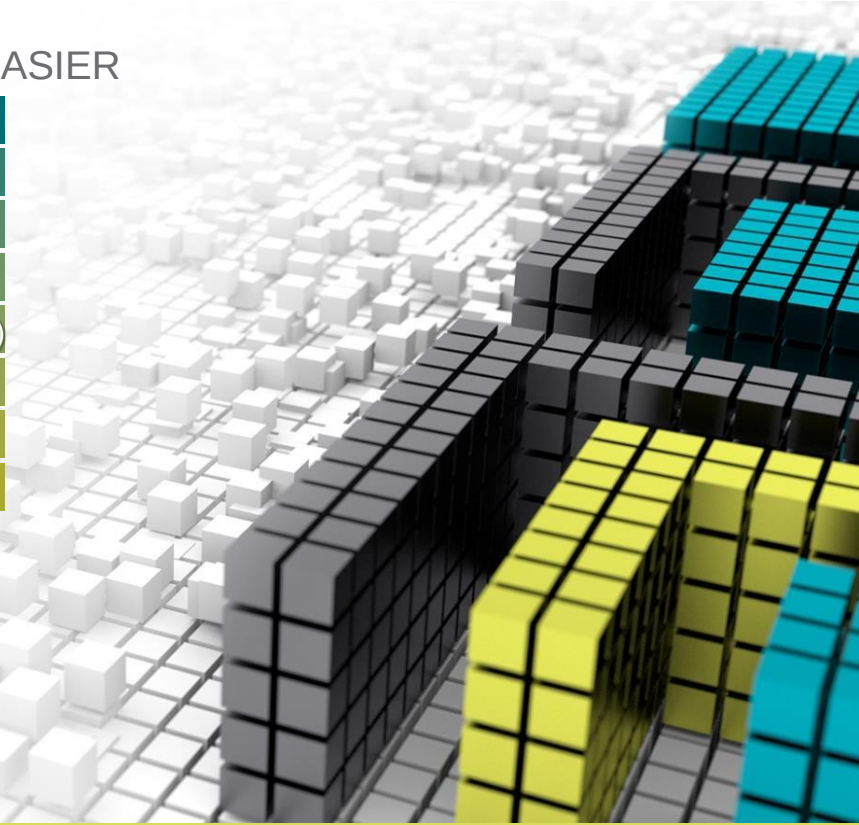
THE HETEROGENEOUS SYSTEM ARCHITECTURE
IT'S BEYOND THE GPU

PAUL BLINZER
AMD INC, FELLOW, SYSTEM SOFTWARE
SYSTEM ARCHITECTURE WORKGROUP CHAIR
HSA FOUNDATION

THE HSA VISION

MAKE HETEROGENEOUS PROGRAMMING MUCH EASIER

1	Single source programming in common high-level languages
2	Enable the programming language of the developer
3	Eliminate data copies
4	Common address space
5	Standardized command submission to the processor (GPU, …)
6	Eliminate software layers between application and hardware
7	ISA agnostic for CPU, GPU and other accelerators
8	Open source software stack



High performance

Low power

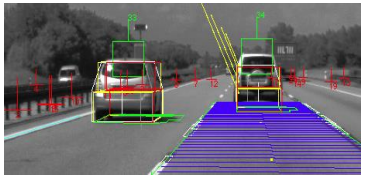
Extensible to other accelerators on the SoC

END USERS BENEFIT FROM HSA WITH 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 analytics, DNN 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 PILLARS OF HSA

- To bring accelerators forward as a first class processor within the system
  - **Unified process address space across all processors (Shared Virtual Memory)**
  - **Processors operating with the application's pageable system memory**
  - Memory coherency between CPU and HSA components simplifies "data collaboration"
  - Well-defined relaxed consistency memory model suited for many high level languages
  - Platform atomics
  - Architected "memory-based" signals and event mechanisms between processers
  - User mode dispatch/scheduling via AQL (eliminates "drivers" from the dispatch path)
  - QoS through pre-emption and context switching*
- Some non-HSA platforms support a few of these platform features
  - In combination these features greatly simplify programmability

HSA – AN OPEN PLATFORM

- Open Architecture, membership open to all
- Delivered via royalty free standards
  - Royalty Free IP, Specifications and APIs
- ISA agnostic for both CPU and GPU
  - Vendors for x86, MIPS, ARM and many GPU architectures
- Membership from all areas of computing
  - Hardware companies
  - Operating Systems
  - Tools and Middleware
  - Applications
  - Universities

MEMBERS DRIVING HSA

HSA FOUNDATION

Founders

AMD ARM Imagination MEDIATEK QUALCOMM SAMSUNG

Promoters

LG Electronics GPT HX

Supporters

Arteris FABRICENGINE codeplay allinea Lawrence Livermore National Laboratory MULTICORE WARE Stream Computing Performance Engineers

Sandia National Laboratories OAK RIDGE National Laboratory Linaro Argonne NATIONAL LABORATORY KISHONTI SUSE

Contributors

ITRI Industrial Technology Research Institute ANALOG DEVICES apical mobica CANONICAL CEVA ETRI 한국전자통신연구원 HUAWEI MARVELL

OMP SYNOPSYS SONICS SONY make.believe Texas Instruments VIVANTE Swarm64

BROADCOM ubuntu tensilica VIK S3 GRAPHICS TOSHIBA Leading Innovation

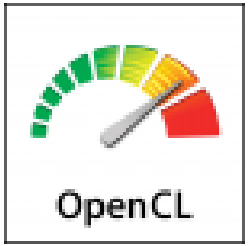
Academic

NTHU Programming Language Lab NTHU System Software Lab University of BRISTOL THE UNIVERSITY of EDINBURGH informatics ILLINOIS COMPUTER SCIENCE RICE Unconventional Wisdom 清華大學 Tsinghua University

T.E.I. of Crete THE UNIVERSITY of MISSISSIPPI

OPENCL™ WITH HSA
NOT OPENCL™ VS HSA!

- HSA is an optimized platform architecture, which runs OpenCL™ very well
  - It is a complementary standard, not a competitor to OpenCL™
  - It is focused on the hardware and system platform runtime definition more than an API itself
  - It supports many more languages than C/C++, including managed code languages
- OpenCL™ on HSA benefits from a rich and consistent platform infrastructure
  - Pointers shared between CPU and GPU (Shared Virtual Memory), Avoidance of wasteful copies
  - Low latency dispatch
  - Improved and consistent memory model
  - Virtual function calls
  - Flexible control flow
  - Exception generation and handling
  - Device and platform atomics

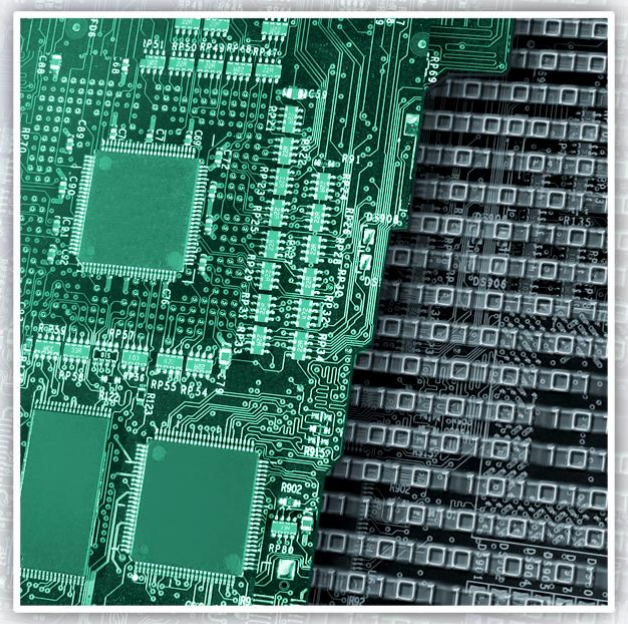


TERMS

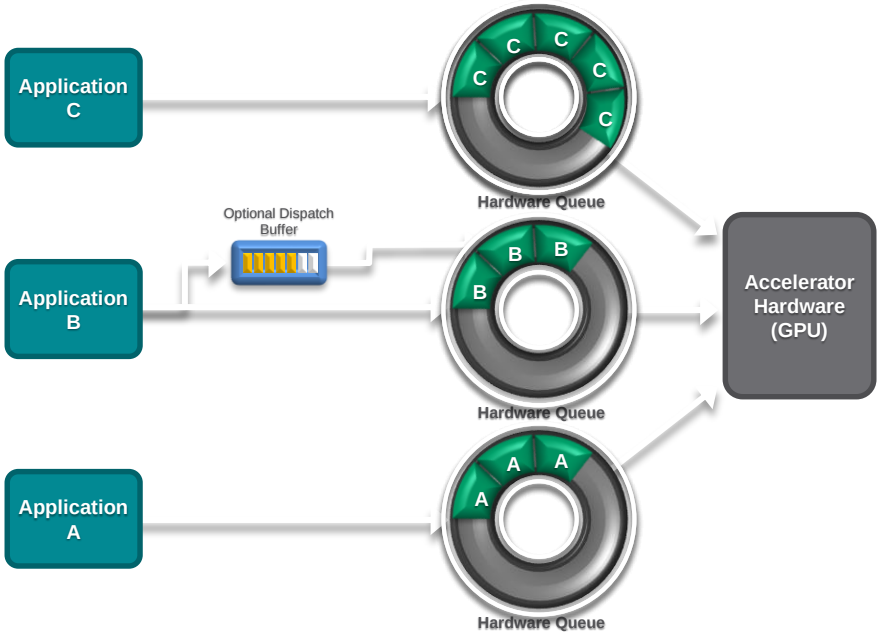
- Host (CPU)
  - An agent that supports a native CPU instruction set
  - Can dispatch commands to kernel agents
  - Can construct Architected Query Language (AQL) packets
  - Can also act as a kernel agent
- Kernel Agent (GPU, DSP, ISP, etc.)
  - An agent that supports HSAIL
  - Has an AQL packet processor
  - Can dispatch commands to any kernel agent
    - Including itself
- Other Agent
  - An agent that participates in the HSA memory model

HSA MEMORY MODEL

- Defines data visibility, ordering between all threads in the HSA System
- Designed to be compatible with C++11, Java, OpenCL and .NET Memory Models
- Relaxed consistency memory model for parallel compute performance
  - HRF based definition, scopes, relaxed atomics
  - Plan: formal definition, automated verification
- Visibility controlled by:
  - Load.Acquire
  - Store.Release
  - Fences



HSA COMMAND AND DISPATCH FLOW



SW view:

- User-mode dispatches to HW
- No Kernel Driver overhead
- Low dispatch times
- CPU & GPU dispatch APIs

HW view:

- HW / microcode controlled
- HW scheduling
- Architected Queuing Language (AQL)
- HW-managed protection

HSA QUEUING MODEL

User mode queuing

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

Architected Queuing Layer (AQL)

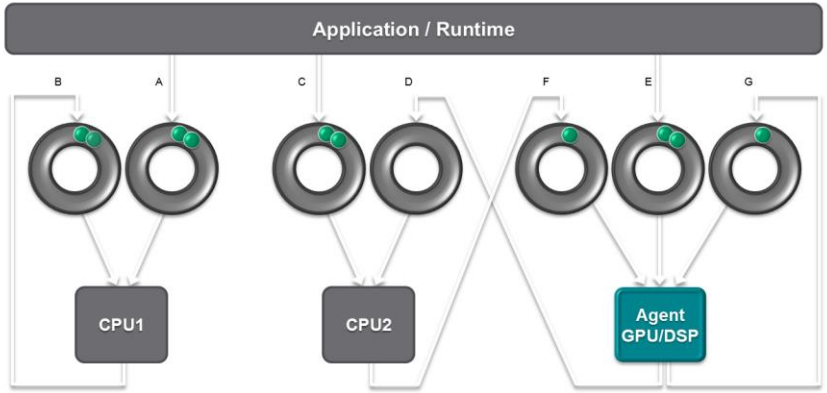
- Single compute dispatch path for all hardware
- No driver translation, direct to hardware
- Standard across vendors!
- Guaranteed backward compatibility

Allows for dispatch to queue from any agent

- CPU or GPU or DSP or FPGA, etc.

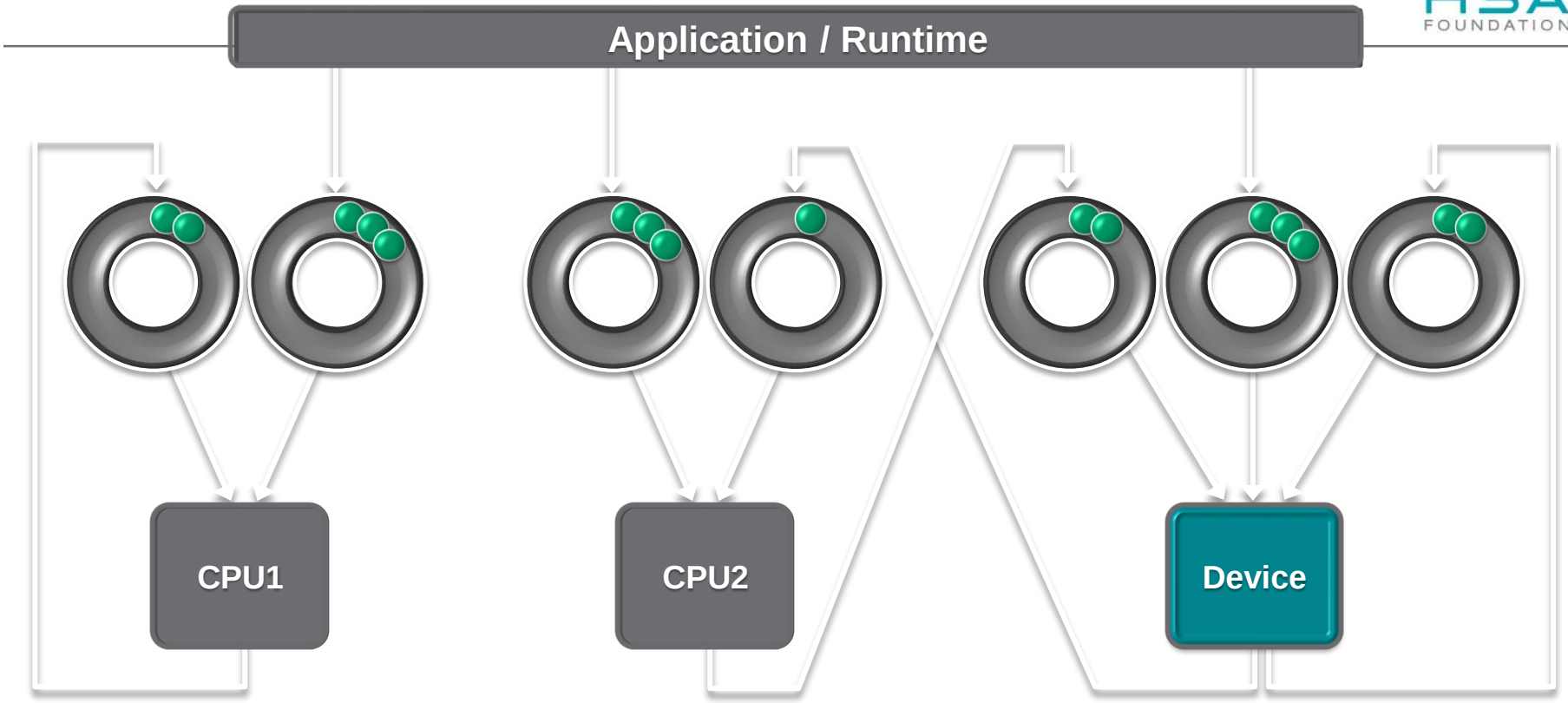
Agent self enqueue enables

- Recursion, Tree traversal, Wavefront reforming



Requires coherency and shared virtual memory

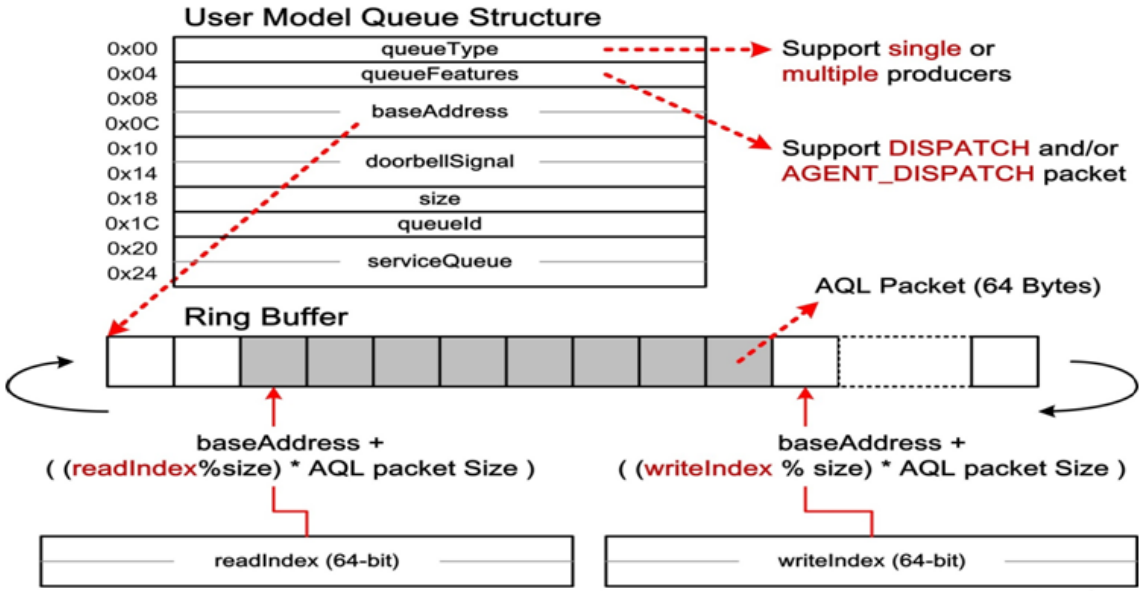
COMMAND AND DISPATCH CPU <-> DEVICE



THE AQL QUEUE DEFINITION

- AQL queue structure

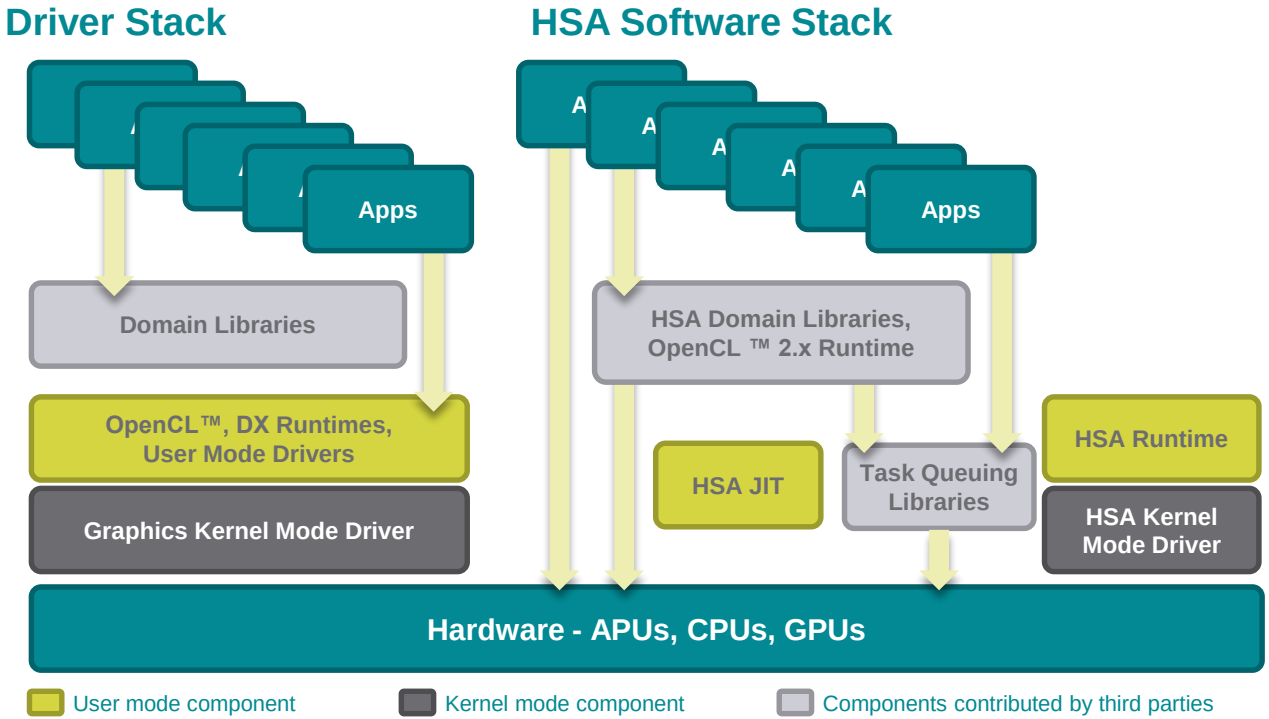
```
typedef struct hsa_queue_s {
    hsa_queue_type_t type;
    uint32_t features;
    uint64_t base_address;
    hsa_signal_handle_t doorbell_signal;
    uint32_t size;
    uint32_t id;
    uint64_t service_queue;
} hsa_queue_t
```



HSA SECURITY AND EXECUTION MODEL

- HSA components operate in the same security infrastructure as the host CPU
  - User and privileged memory distinction
  - Hardware enforced process space isolation
  - Page attributes (Read, write, execute) protections enforced by HW, apply as defined by system
  - Internally, the platform partitions functionality by privilege level
  - User mode queues can only run AQL packets within the defined process context
- HSA defines Quality of Service requirements
  - Requires support for mechanisms to schedule both HSA and non-HSA workloads for devices that support both task types with appropriate priority, latency, throughput and scheduling constraints.
  - Context Switch
  - Preempt
  - Terminate and Context Reset

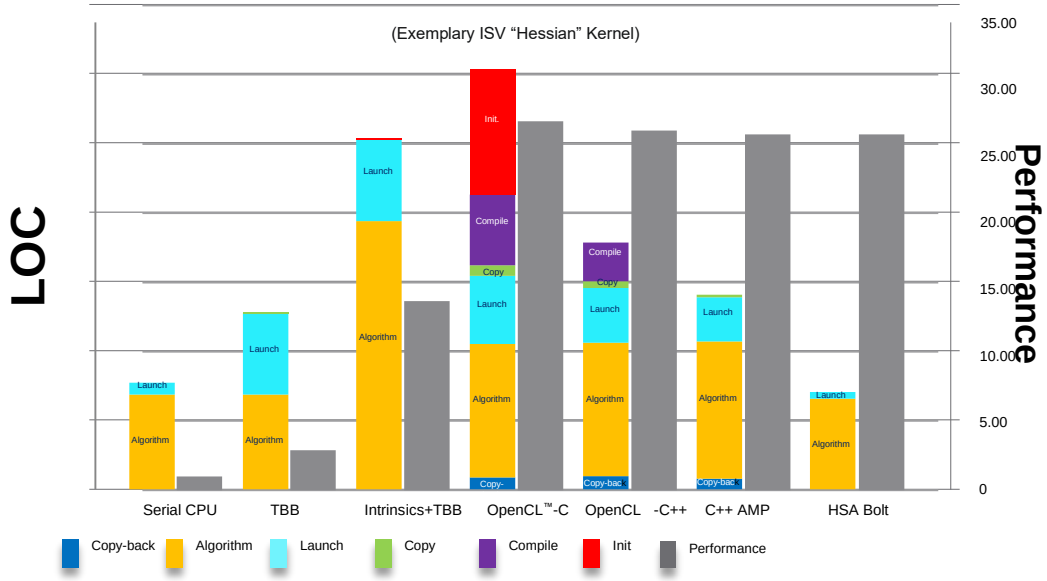
HSA - EVOLUTION OF THE SOFTWARE STACK



HSA INTERMEDIATE LAYER - HSAIL

- HSAIL is a virtual ISA for parallel programs
  - Finalized to ISA by a JIT compiler or “Finalizer”
  - ISA independent by design for CPU & GPU
- Explicitly parallel
  - Designed for data parallel programming
- Support for exceptions, virtual functions, and other high level language features
- Agent Dispatch to call OS and system runtime
  - GPU/accelerator code can call directly to OS and other system runtime services, I/O, printf, etc.!
- Debugging, Profiling support requirements

LINES-OF-CODE AND PERFORMANCE COMPARISONS



HSA SPECIFICATIONS

- HSA System Architecture Specification
  - Version 1.0 defines discovery, memory model, queue management, atomics, etc
- HSA Programmers Reference Specification
  - Version 1.0 defines the HSAIL language and object format
- HSA Runtime Software Specification
  - Version 1.0 defines the APIs through which an HSA application uses the platform
- All released specifications can be found at the HSA Foundation web site:
  - www.hsafoundation.com/standards

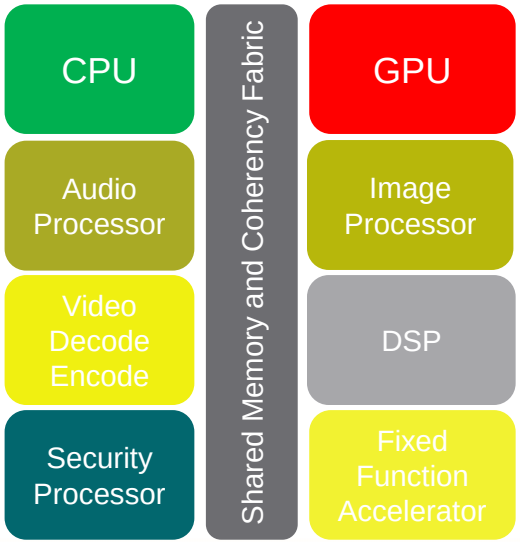
HSA OPEN SOURCE SOFTWARE

- HSA features a full open source Linux execution and compilation stack
  - Allows a single shared implementation for many components
  - Enables university research and collaboration in all areas
  - Because it's the right thing to do

- Many open source applications & frameworks, ported to HSA more in the works
  - Native Languages: Kalmar C++17, HCC, LLVM, GCC, CLOC/SNACK, Python, Java, …
  - API's, Frameworks: POCL, Docker, OpenMP, OKRA, HIP, …
  - Research: Multi2sim, HSAEmu, gem5, ViennaCL, …
  - And many applications using OCL 2.0 or HSA stack
  - Github & Bitbucket repositories have much, much more…

LOOKING BEYOND THE DATA PARALLEL COMPUTE APPLICATION

- The initial 1.0 release of the HSA specifications focuses on data parallel compute language and apps
  - Focus is on integrating GPUs into the general high-level language software infrastructure
  - But the next generations of the specifications will apply to other domains
  - With their domain-specific HW processor language focus
  - Updates to 1.1 specification are very close to release
- By design the HSA infrastructure is quite easy to extend
  - Initial focus is on data parallel compute tasks
  - But other areas of Domain Specific Processors are under consideration
  - Architected Topology infrastructure allows to reliably identify and address domain specific accelerator capabilities
- By design the HSA infrastructure is easy to virtualize
  - Programming model does leverage few, simple hardware & platform paradigms (queues, signals, memory) for its operation
  - Future spec work may put additional requirements to cover such environments



WHAT DO I NEED TO START PLAYING?

- **A10-8800 (Carrizo)** system
  - Carrizo system recommended, many different vendors (HP, Dell, Toshiba, Lenovo, …) available
  - No discrete GPU in the system (for now, using default pre-built image)
  - Other HSA members will release HSA compliant hardware within the year
- Ubuntu 14.04 64bit LTS or Fedora 21
  - OpenSuSE and Redhat work too, but not officially supported (require rebuild of image)
- https://github.com/HSAFoundation/HSA-Drivers-Linux-AMD
  - https://github.com/HSAFoundation/ has a collection of compilers, tools, debuggers, …
- AMD's "Boltzmann" initiative will support HSA subset for HPC on discrete GPU
  - Simplifies porting from proprietary APIs via HIP, HSA compatible programming model (ROCR)
  - Go to http://gpuopen.com for more

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

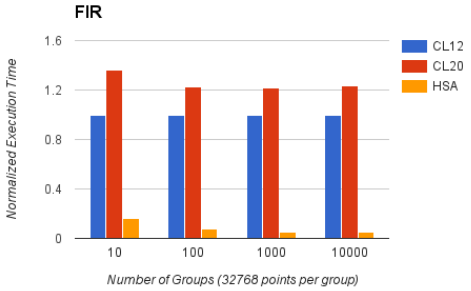
- Copy to/from SVM
- Data copy cannot be avoided, since the space for SVM is limited

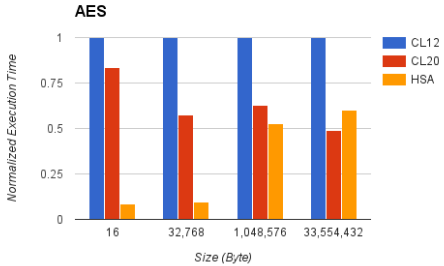
HSA – Unified Memory Space – Fine Grained Sync

- Regular pointer
- No explicit copy

Results

- HSA compute abstraction
- NO performance penalty
- Measured on Kaveri (A pre-HSA 1.0 device)
- AMD Carrizo (HSA 1.0 compliant) improves performance for coherent transactions





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

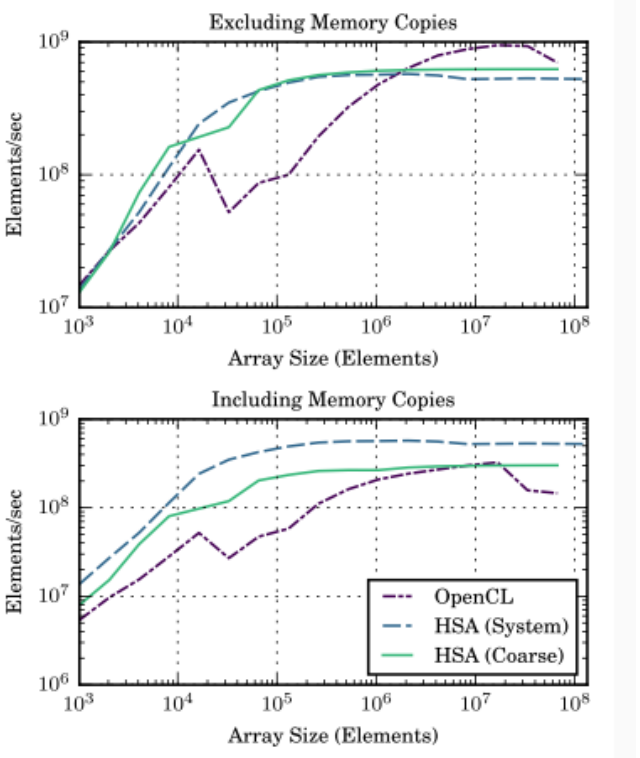
- Matches or outperforms OpenCL

Course Grained SVM

- Matches OpenCL buffers for bandwidth
- More predictable performance

Fine Grained SVM

- Faster kernel dispatch
- Larger allocations
- Shared data structure



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

IN SUMMARY…

- HSA is not about a specific API, feature or runtime
  - It is about a paradigm to efficiently access the various heterogeneous components in a system by software
  - It allows application programmers to use the languages of their choice to efficiently implement their code
- HSA is not about a specific hardware or vendor or Operating System
  - It defines a few fundamental requirements and concepts as building blocks software at all levels can depend on
  - HW vendors can efficiently expose their compute acceleration features to software in an architected way
  - OS, runtimes and application frameworks can build efficient data and task parallel runtimes leveraging these
  - Application software can more easily use the right tool for the job through high level language support
- HSA is an open and flexible concept
  - Collaborative participation through the HSA Foundation is encouraged for companies and academia
  - The first set of standards by the HSA Foundation is released, first products are available and a number of language and application frameworks are available
  - This is a good time to engage, lots of research opportunities
- HSA Foundation sponsors research on heterogeneous platform technologies

ACKNOWLEDGEMENTS

- With thanks to Dr. John Glossner, Ben Sander, Greg Stoner and others in the HSA Foundation for some materials and feedback

ANY QUESTIONS?

- Of course there are, so go ahead ☺

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

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