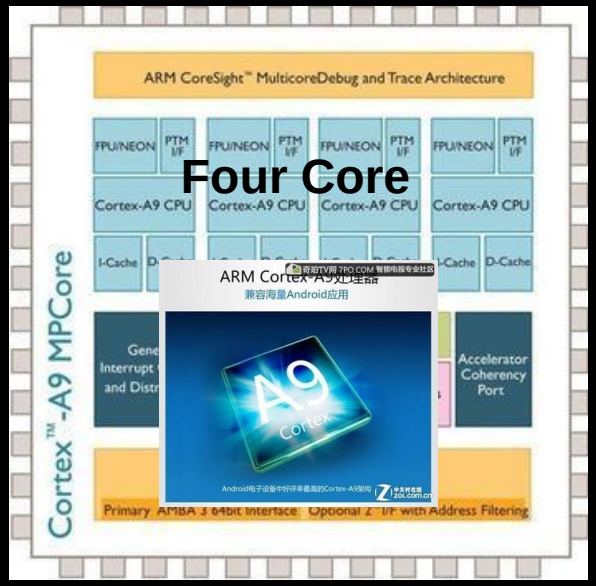
面向未来异构处理器体系和编程模型

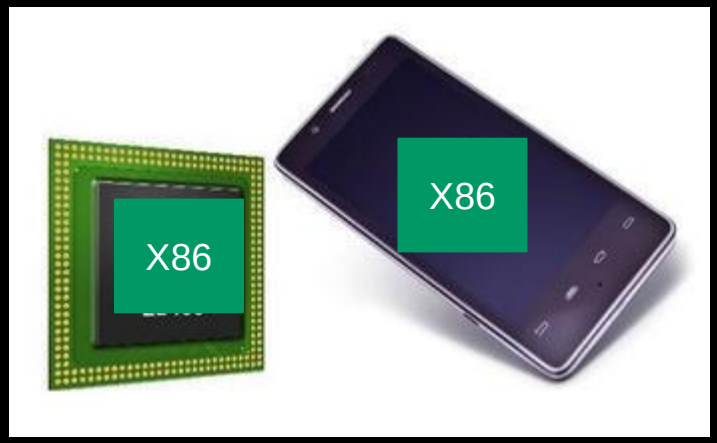
Heterogeneous System Architecture(HSA)

楚含进, Mar 2013

当低功耗和移动成为考核标准

核数增加能够解决移动和低功耗与性能矛与盾吗？





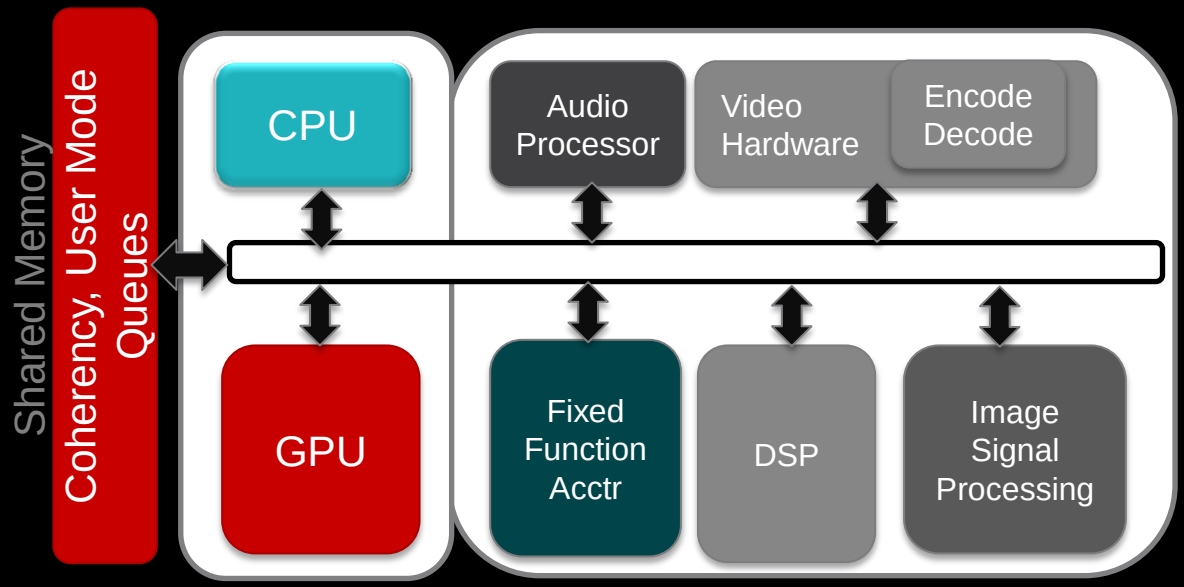
多核？

多核在低功耗和移动的需求下能否抑制走下去？

HSA ARCHITECTURE – 异构的本质：通用处理器和专属处理器的博弈与融合

GPU compute C++ support
User Mode Scheduling
Fully coherent memory between CPU & GPU
GPU uses pageable system memory via CPU pointers
GPU graphics pre-emption
GPU compute context switch

当CPU不再成为设计门槛

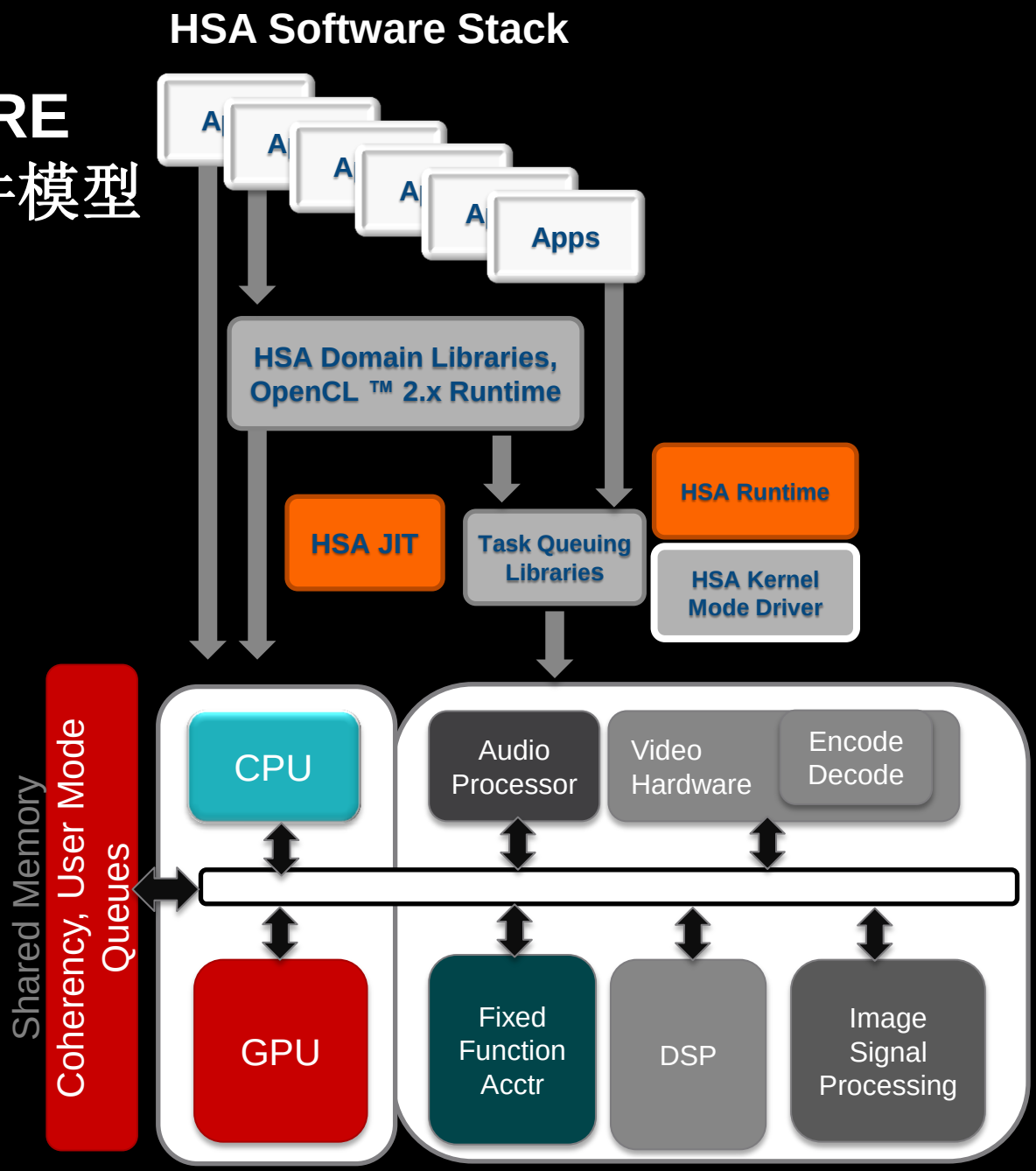


GOALS FOR THE HETEROGENEOUS SYSTEM ARCHITECTURE – 异构系统不应增加编程的困难

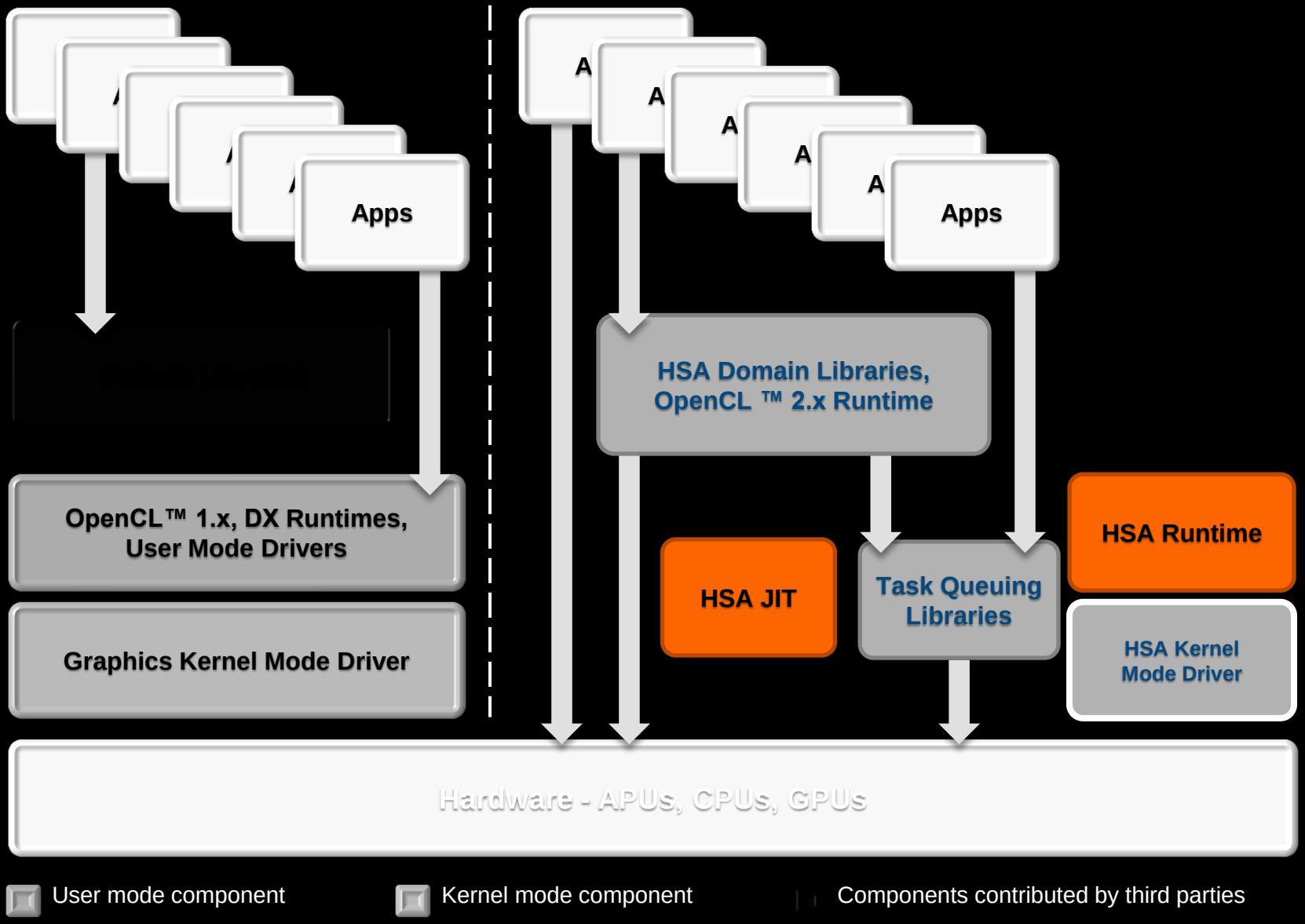
- Advanced Natural User Interfaces & Presence Capabilities
- Rich Cloud Computing User Experiences
- Perceptual Computing Experiences
- Bring Hollywood Class Realism to Real-time Entertainment

HSA ARCHITECTURE
新架构，新软件模型

GPU compute C++ support
User Mode Scheduling
Fully coherent memory between CPU & GPU
GPU uses pageable system memory via CPU pointers
GPU graphics pre-emption
GPU compute context switch



Driver Stack



HSA INTERMEDIATE LANGUAGE - HSAIL

Designed for C99, C++ 2011, Java, Renderscript, OpenCL, C++ AMP

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

Syscall methods

- GPU code can call directly to system services, IO, printf, etc

OPENCL™ AND HSA

HSA is an optimized platform architecture for OpenCL™

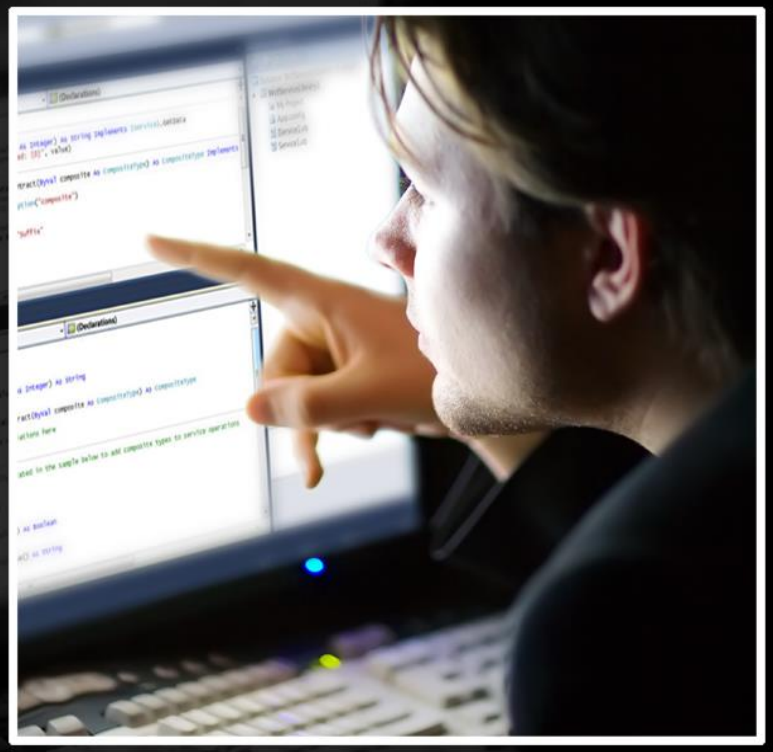
- Not an alternative to OpenCL™

OpenCL™ on HSA will benefit from

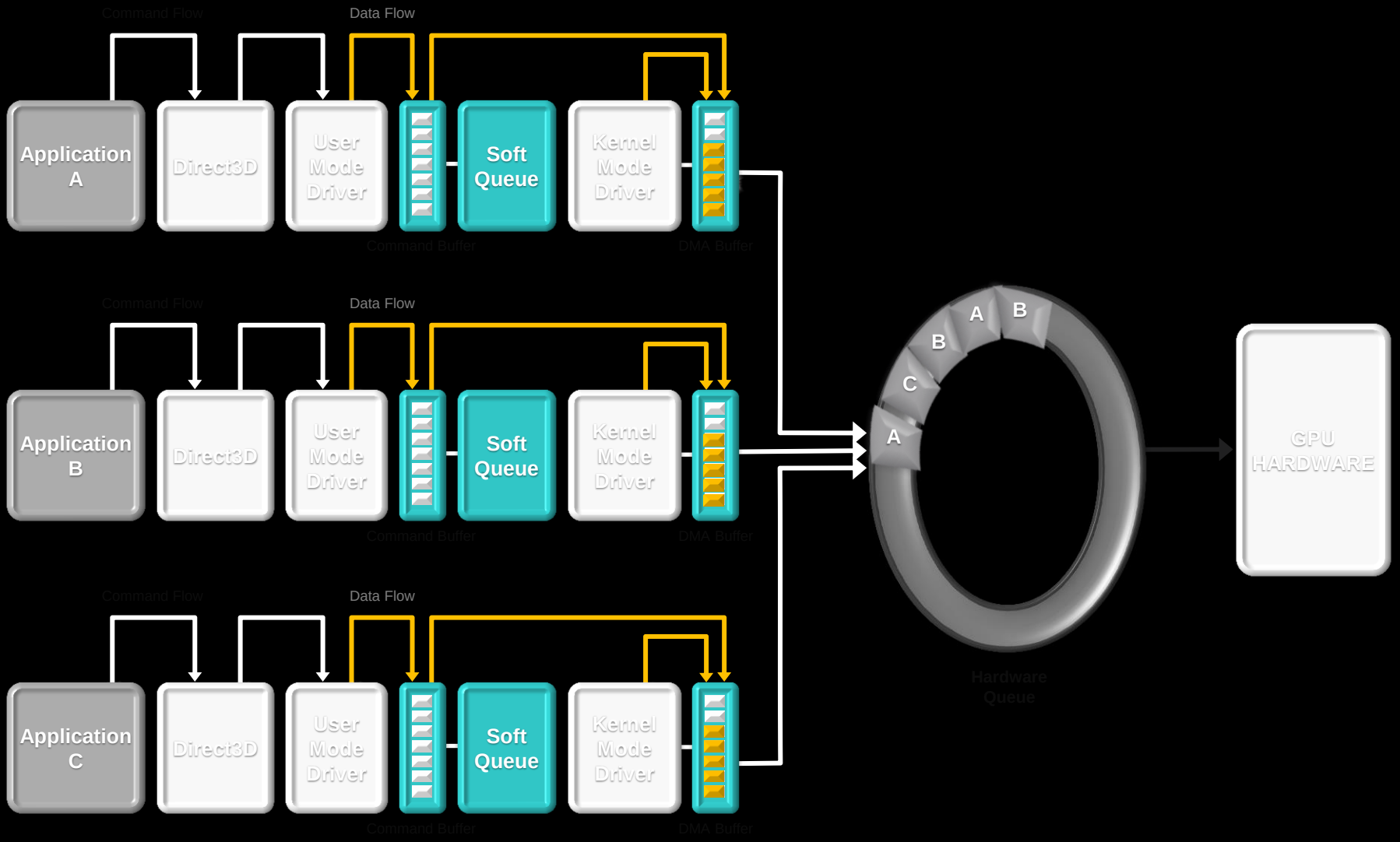
- Avoidance of wasteful copies
- Low latency dispatch
- Improved memory model
- Pointers shared between CPU and GPU

HSA also exposes a lower level programming interface, for those that want the ultimate in control and performance

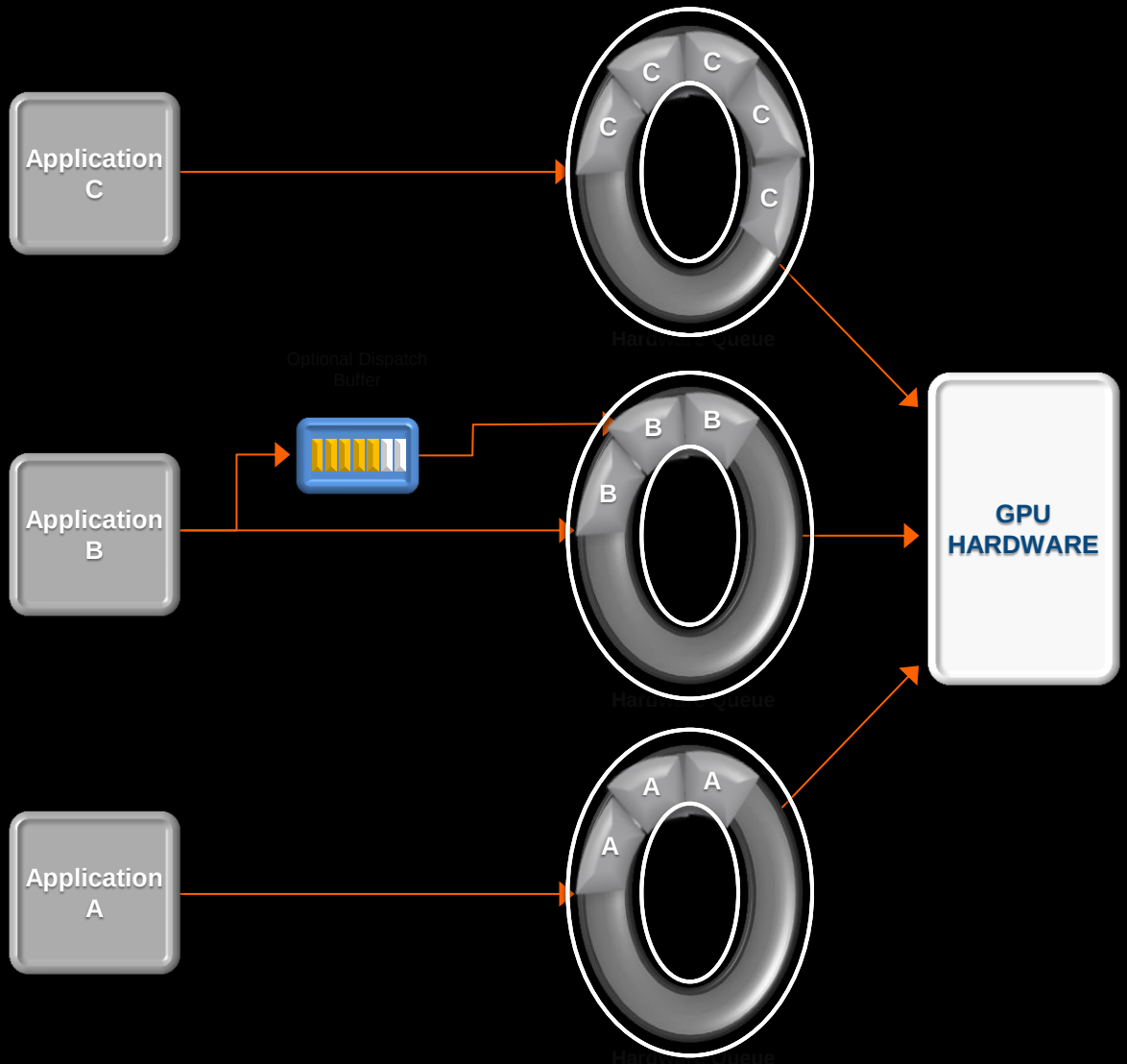
- Optimized libraries may choose the lower level interface



TODAY'S COMMAND AND DISPATCH FLOW



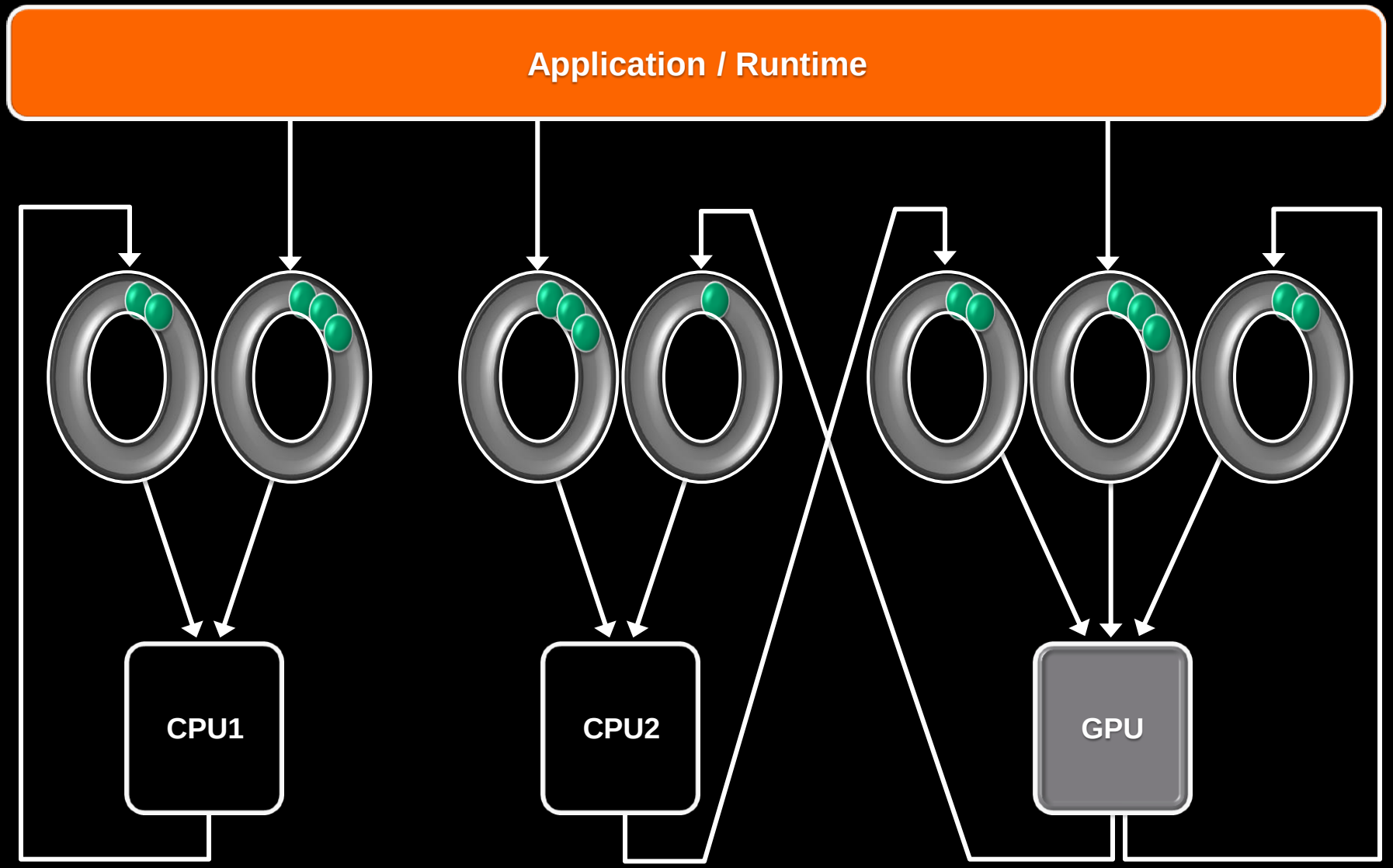
HSA COMMAND AND DISPATCH FLOW



- Application codes to the hardware
- User mode queuing
- Hardware scheduling
- Low dispatch times

- No APIs
- No Soft Queues
- No User Mode Drivers
- No Kernel Mode Transitions
- No Overhead!

COMMAND AND DISPATCH CPU <-> GPU



HSA AND MOBILE: 异构与智能移动平台

Unifies the platform architecture for multiple hardware vendors

- Avoid the need for unique ports for each vendor

High-performance and very power efficient architecture creates a rich foundation for compute intensive application

- Brings greater security to platform via privileged memory support on the GPU and other co-processors and the ability to preempt or kill process on GPU.
- GPU and Co-Processor now supported in a Unified Coherent Memory with a consistent memory model.
- Zero data copy to device combined with very low latency kernel dispatch.
- Support for safe process control of the GPU.
- Support for user mode queues that more closely maps to android runtime

Improved debugging and performance analysis of co-processors.

HOW: 一个业界统一的开放标准

Founders

Promoters

Supporters

Contributors

Academic

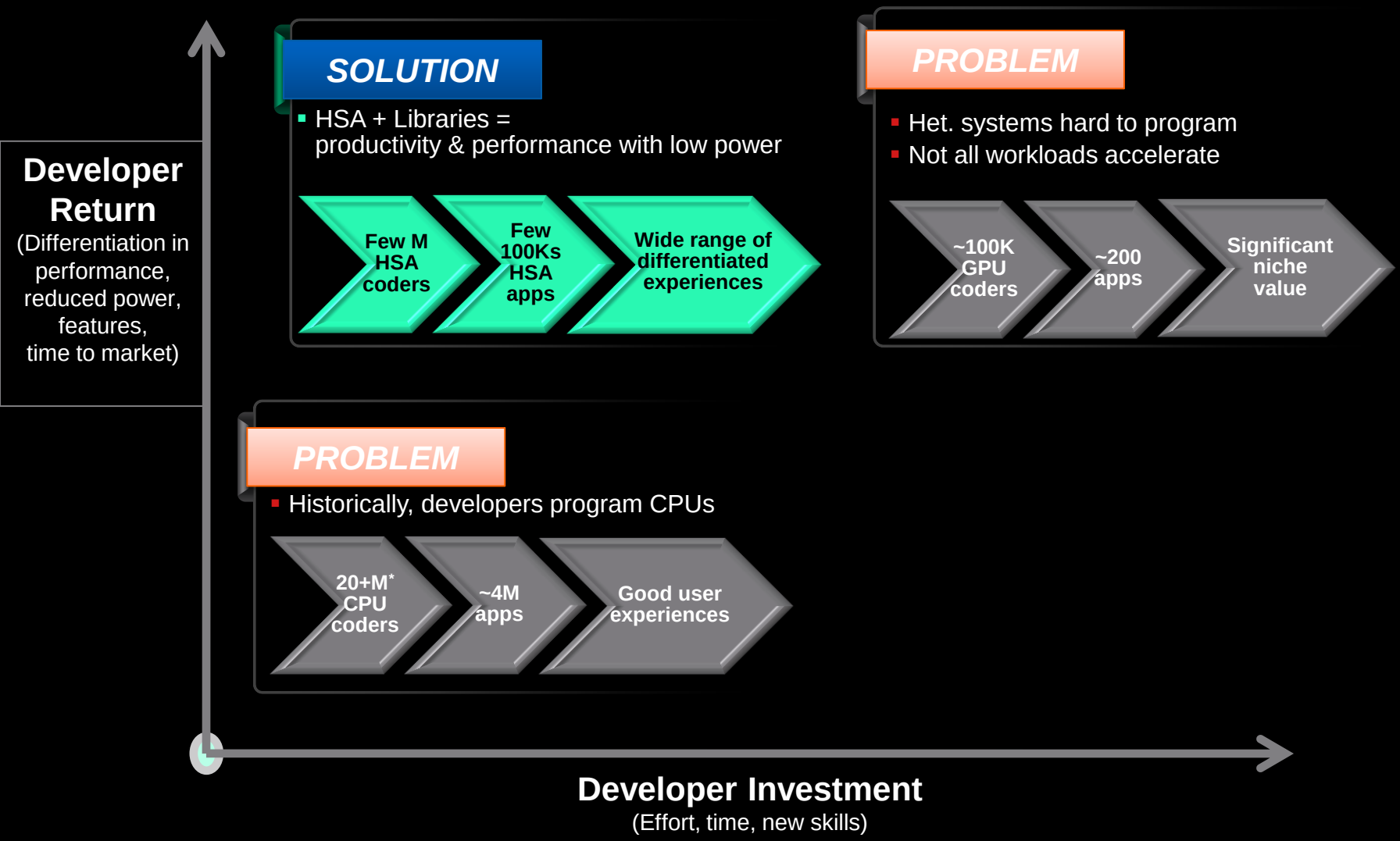
University of Illinois
Computer Science

Associates

RESEARCH TOPICS IN HSA

Category	Description	Comments
Languages/Compilers	Higher-level languages. GPU languages are primitive today. OpenCL is a good expert tool. Look into domain specific languages (graphics, math). Ex: HSA could have a database accelerator component	
	Split compilation model – high level compliers & low level compilers and how to make them work well together	
	How to run best on a device with multi ISA's	
Software Run-Time	Classic load balancing. Look for new ways to partition algorithms automatically in the runtime. Simultaneous running of multiple kernels or multiple applications. Quality of service & virtualization. Scheduling for complex status graphs and scheduling dynamic parallelism	
System Architecture	• Bandwidth/memory arch (balancing BW with compute) • Load balancing • Memory configurations: Stack memory devices will eventually appear and systems will change around idea of bandwidth. Shared memory stacks – what are the implications? • TCU/LCU ratios	
Hardware	Logical split between split function hardware. • Applying HSA to non-GPU devices (DSPs, FPGAs, etc.) • Heterogeneous conformance optimization - how to run a program that runs well on all different HSA platforms and hardware	
	Memory system design: low cost support for coherency and would give programmers a way to optimize their use of coherence	
	Security: looking into securing systems	
	Efficient synchronization primitives	
	3D graphics pipes – integration with HSA	

THE HSA OPPORTUNITY



*IDC

一个异构计算爱好者的家园

技术是交流和进步的一种方式