HETEROGENEOUS SYSTEM ARCHITECTURE

Phil Rogers,
AMD Corporate Fellow
HSA Foundation President
GOALS

Make the unprecedented processing capability of the APU as accessible to programmers as the CPU is today

Dramatically expand the APU software ecosystem in client and server

Enable immersive applications whether hosted locally or in the cloud
APU: ACCELERATED PROCESSING UNIT

- The APU has arrived and it is a great advance over previous platforms
- Combines scalar processing on CPU with parallel processing on the GPU and high bandwidth access to memory
- How do we make it even better going forward?
  - Easier to program
  - Easier to optimize
  - Easier to load balance
  - Higher performance
  - Lower power
A NEW ERA OF PROCESSOR PERFORMANCE

Single-Core Era

Enabled by:
✓ Moore’s Law
✓ Voltage Scaling

Constrained by:
× Power
× Complexity

Assembly ➔ C/C++ ➔ Java

Multi-Core Era

Enabled by:
✓ Moore’s Law
✓ SMP architecture

Constrained by:
× Power
× Parallel SW
× Scalability

pthreads ➔ OpenMP / TBB …

Heterogeneous Systems Era

Enabled by:
✓ Abundant data parallelism
✓ Power efficient GPUs

Temporarily Constrained by:
× Programming models
× Comm. overhead

Shader ➔ CUDA ➔ OpenCL ➔ C++ and Java

Assembly ➔ C/C++ ➔ Java

pthreads ➔ OpenMP / TBB …

Shader ➔ CUDA ➔ OpenCL ➔ C++ and Java

© Copyright 2012 HSA Foundation. All Rights Reserved.
EVOLUTION OF HETEROGENEOUS COMPUTING

Proprietary Drivers Era
- Graphics & Proprietary Driver-based APIs
  - “Adventurous” programmers
  - Exploit early programmable “shader cores” in the GPU
  - Make your program look like “graphics” to the GPU
  - CUDA™, Brook+, etc

Standards Drivers Era
- OpenCL™, DirectCompute Driver-based APIs
  - Expert programmers
  - C and C++ subsets
  - Compute centric APIs, data types
  - Multiple address spaces with explicit data movement
  - Specialized work queue based structures
  - Kernel mode dispatch

Architected Era
- Heterogeneous System Architecture
- GPU Peer Processor
  - Mainstream programmers
  - Full C++
  - GPU as a co-processor
  - Unified coherent address space
  - Task parallel runtimes
  - Nested Data Parallel programs
  - User mode dispatch
  - Pre-emption and context switching
HSA FEATURE ROADMAP

**Physical Integration**
- Integrate CPU & GPU in silicon
- Unified Memory Controller
- Common Manufacturing Technology

**Optimized Platforms**
- GPU Compute C++ support
- User mode scheduling
- Bi-Directional Power Mgmt between CPU and GPU

**Architectural Integration**
- Unified Address Space for CPU and GPU
- GPU uses pageable system memory via CPU pointers
- Fully coherent memory between CPU & GPU

**System Integration**
- GPU compute context switch
- GPU graphics pre-emption
- Quality of Service
COMMITTED TO OPEN STANDARDS

- HSA Foundation believes in open and de-facto standards
  - Compete on the best implementation
- Open standards are the basis for large ecosystems
- Open standards always win over time
  - SW developers want their applications to run on multiple platforms from multiple hardware vendors
HETEROGENEOUS SYSTEM ARCHITECTURE – AN OPEN PLATFORM

- Open Architecture, published specifications
  - HSAIL virtual ISA
  - HSA memory model
  - HSA system architecture
- ISA agnostic for both CPU and GPU
- HSA Foundation formed in June 2012
- Inviting partners to join us, in all areas
  - Hardware companies
  - Operating Systems
  - Tools and Middleware
  - Applications

© Copyright 2012 HSA Foundation. All Rights Reserved.
HSAIL 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
- Syscall methods
  - GPU code can call directly to system services, IO, printf, etc
- Debugging support
HSA MEMORY MODEL

- Designed to be compatible with C++11, Java and .NET Memory Models
- Relaxed consistency memory model for parallel compute performance
- Loads and stores can be re-ordered by the finalizer
- Visibility controlled by:
  - Load.Acquire
  - Store.Release
  - Barriers
HSA SOFTWARE STACKS
APPLICATIONS, SYSTEM SOFTWARE AND PROGRAMMING MODELS
APPLICATION AREAS WITH ABUNDANT PARALLEL WORKLOADS

**Biometric Recognition**
- Secure, fast, accurate: face, voice, fingerprints

**Augmented Reality**
- Superimpose graphics, audio, and other digital information as a virtual overlay

**Content Everywhere**
- Content from any source to any display seamlessly

**Beyond HD Experiences**
- Streaming media, new codecs, 3D, transcode, audio

**Natural UI & Gestures**
- Touch, gesture, and voice

**AV Content Management**
- Searching, indexing and tagging of video & audio, multimedia data mining
INTRODUCING HSA BOLT – PARALLEL PRIMITIVES LIBRARY FOR HSA

- Easily leverage the inherent power efficiency of GPU computing
  - Common routines such as scan, sort, reduce, transform
  - More advanced routines like heterogeneous pipelines
  - Bolt library works with OpenCL or C++ AMP

- Enjoy the unique advantages of the HSA platform
  - Move the computation not the data

- Finally a single source code base for the CPU and GPU!
  - Developers can focus on core algorithms

- Bolt Preview available in AMD APP SDK 2.8 (2H-2012)
HSA SOLUTION STACK

Application SW

HSA Software Drivers

Differentiated HW

Application

Domain Specific Libs (Bolt, OpenCV™, … many others)

OpenCL™ Runtime

DirectX Runtime

Other Runtime

Legacy Drivers

HSA Runtime

HSAIL

HSA Finalizer

Knl Driver

GPU ISA

GPU(s)

CPU(s)

Other Accelerators

GPU ISA

HSA SOLUTION STACK

Application

Domain Specific Libs (Bolt, OpenCV™, … many others)

OpenCL™ Runtime

DirectX Runtime

Other Runtime

Legacy Drivers

HSA Runtime

HSAIL

HSA Finalizer

Knl Driver

GPU ISA

GPU(s)

CPU(s)

Other Accelerators

HSA SOLUTION STACK

Application

Domain Specific Libs (Bolt, OpenCV™, … many others)

OpenCL™ Runtime

DirectX Runtime

Other Runtime

Legacy Drivers

HSA Runtime

HSAIL

HSA Finalizer

Knl Driver

GPU ISA

GPU(s)

CPU(s)

Other Accelerators

HSA SOLUTION STACK

Application

Domain Specific Libs (Bolt, OpenCV™, … many others)

OpenCL™ Runtime

DirectX Runtime

Other Runtime

Legacy Drivers

HSA Runtime

HSAIL

HSA Finalizer

Knl Driver

GPU ISA

GPU(s)

CPU(s)

Other Accelerators
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
MAKE GPUs EASIER TO PROGRAM: PRIMARY PROGRAMMING MODELS

- Microsoft C++AMP
  - Address huge population of developers
  - Integrated in Visual Studio and Windows 8 Metro
  - Microsoft Community Promise License for open platform use

- Java acceleration
  - Aparapi on OpenCL today
  - Project Sumatra to add HSA support in an OpenJDK for Java 8
  - Driving to have Sumatra absorbed into Java 9
AMDM’S OPEN SOURCE COMMITMENT TO HSA

- We will open source our linux execution and compilation stack
  - Jump start the ecosystem
  - Allow a single shared implementation where appropriate
  - Enable university research in all areas

<table>
<thead>
<tr>
<th>Component Name</th>
<th>AMD Specific</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSA Bolt Library</td>
<td>No</td>
<td>Enable understanding and debug</td>
</tr>
<tr>
<td>HSAIL Code Generator</td>
<td>No</td>
<td>Enable research</td>
</tr>
<tr>
<td>LLVM Contributions</td>
<td>No</td>
<td>Industry and academic collaboration</td>
</tr>
<tr>
<td>HSA Assembler</td>
<td>No</td>
<td>Enable understanding and debug</td>
</tr>
<tr>
<td>HSA Runtime</td>
<td>No</td>
<td>Standardize on a single runtime</td>
</tr>
<tr>
<td>HSA Finalizer</td>
<td>Yes</td>
<td>Enable research and debug</td>
</tr>
<tr>
<td>HSA Kernel Driver</td>
<td>Yes</td>
<td>For inclusion in linux distros</td>
</tr>
</tbody>
</table>
ACCELERATED WORKLOADS

CLIENT AND SERVER EXAMPLES
HAAR Face Detection

CORNERSTONE TECHNOLOGY
FOR COMPUTERVISION
LOOKING FOR FACES IN ALL THE RIGHT PLACES

Quick HD Calculations
Search square = 21 x 21
Pixels = 1920 x 1080 = 2,073,600
Search squares = 1900 x 1060 = ~2 Million
LOOKING FOR DIFFERENT SIZE FACES – BY SCALING THE VIDEO FRAME

More HD Calculations
70% scaling in H and V
Total Pixels = 4.07 Million
Search squares = 3.8 Million
Final HD Calculations
Search squares = 3.8 million
Average features per square = 124
Calculations per feature = 100
Calculations per frame = 47 GCalcs

Calculation Rate
30 frames/sec = 1.4TCalcs/second
60 frames/sec = 2.8TCalcs/second

…and this only gets front-facing faces
“Trinity” A10-4600M (6CU@497Mhz, 4 cores@2700Mhz)

AMD A10 4600M APU with Radeon™ HD Graphics; CPU: 4 cores @ 2.3 MHz (turbo 3.2 GHz); GPU: AMD Radeon HD 7660G, 6 compute units, 685MHz; 4GB RAM; Windows 7 (64-bit); OpenCL™ 1.1 (873.1)
PERFORMANCE CPU-VS-GPU

“Trinity” A10-4600M (6CU@497Mhz, 4 cores@2700Mhz)

AMD A10 4600M APU with Radeon™ HD Graphics; CPU: 4 cores @ 2.3 MHz (turbo 3.2 GHz); GPU: AMD Radeon HD 7660G, 6 compute units, 685MHz; 4GB RAM; Windows 7 (64-bit); OpenCL™ 1.1 (873.1)
By seamlessly sharing data between CPU and GPU, HSA allows the right processor to handle its appropriate workload.

**Increased Performance**

**Decreased Energy Per Frame**
ACCELERATING JAVA
GOING BEYOND NATIVE LANGUAGES
JAVA ENABLEMENT BY APARAPI

Aparapi = Runtime capable of converting Java™ bytecode to OpenCL™

Developer creates Java™ source

Source compiled to class files (bytecode) using standard compiler

For execution on any OpenCL™ 1.1+ capable device

OR execute via a thread pool if OpenCL™ is not available
JAVA AND APARAPI HSA ENABLEMENT ROADMAP

Application
APARAPI
JVM

OpenCL™
CPU ISA
GPU ISA

HSA Finalizer
CPU ISA
GPU ISA

HSA GPU

HSA CPU

CPU

GPU

HSAIL

HSA Runtime
LLVM Optimizer

IR

HSA Finalizer
CPU ISA
GPU ISA

HSA CPU

HSA GPU

CPU

GPU

HSAIL

HSA Finalizer
CPU ISA
GPU ISA

HSA CPU

HSA GPU

CPU

GPU

© Copyright 2012 HSA Foundation. All Rights Reserved.
ACCELERATING MEMCACHED

CLOUD SERVER WORKLOAD
MEMCACHED

- A Distributed Memory Object Caching System Used in Cloud Servers
- Generally used for short-term storage and caching, handling requests that would otherwise require database or file system accesses
- Used by Facebook, YouTube, Twitter, Wikipedia, Flickr, and others
- Effectively a large distributed hash table
  - Responds to store and get requests received over the network
  - Conceptually:
    - store(key, object)
    - object = get(key)
OFFLOADING MEMCACHED KEY LOOKUP TO THE GPU

Key Look Up Performance

<table>
<thead>
<tr>
<th></th>
<th>Data Transfer</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multithreaded CPU</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Radeon HD 5870</td>
<td>0.5</td>
<td>3.5</td>
</tr>
<tr>
<td>“Trinity” A10-5800K</td>
<td>2.5</td>
<td>2</td>
</tr>
<tr>
<td>Zacate E-350</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Execution Breakdown

<table>
<thead>
<tr>
<th></th>
<th>100%</th>
<th>80%</th>
<th>60%</th>
<th>40%</th>
<th>20%</th>
<th>0%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multithreaded CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Radeon HD 5870</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Trinity” A10-5800K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zacate E-350</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6189209

© Copyright 2012 HSA Foundation. All Rights Reserved.
ACCELERATING B+TREE SEARCHES
CLOUD SERVER WORKLOAD
B+TREE SEARCHES

- B+Trees are a fundamental data structure
  - Used to reduce memory & disk access to locate a key
  - Can support index- and range-based queries
  - Can be updated efficiently

- B+Trees are used by enterprise DB applications
  - SQL: SQLite, MySQL, Oracle, and others
  - No-SQL: Apache CouchDB, Tokyo Cabinet, and others
    - Audio search, video copy detection

A simple B+Tree linking the keys 1-7. The linked list (red) allows rapid in-order traversal.
By efficiently sharing data between CPU and GPU, HSA increases performance versus Multi Threaded CPU, even for tree structures that reside in host memory.

With HSA, DB can be larger than GPU memory, and can be shared.

- HSA lets us move compute to data
  - Parallel search can move to GPU
  - Sequential updates can remain on CPU

<table>
<thead>
<tr>
<th>Platform</th>
<th>Size &lt; 1.5 GB</th>
<th>Size 1.5-2.7 GB</th>
<th>Size &gt; 2.7 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>dGPU (memory size = 3GB)</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>HSA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>


AMD A10 4600M APU with Radeon™ HD Graphics; CPU: 4 cores @ 2.3 MHz (turbo 3.2 GHz); GPU: AMD Radeon HD 7660G, 6 compute units, 685MHz; 4GB RAM
ACCELERATING SUFFIX ARRAY CONSTRUCTION

CLOUD SERVER WORKLOAD
Suffix Arrays are a fundamental data structure
- Designed for efficient searching of a large text
  - Quickly locate every occurrence of a substring $S$ in a text $T$

Suffix Arrays are used to accelerate in-memory cloud workloads
- Full text index search
- Lossless data compression
- Bio-informatics
ACCELERATED SUFFIX ARRAY CONSTRUCTION ON HSA

By efficiently sharing data between CPU and GPU, HSA lets us move compute to data without penalty of intermediate copies.

By offloading data parallel computations to GPU, HSA increases performance and reduces energy for Suffix Array Construction versus Single Threaded CPU.

Skew Algorithm for Compute SA


AMD A10 4600M APU with Radeon™ HD Graphics; CPU: 4 cores @ 2.3 MHz (turbo 3.2 GHz); GPU: AMD Radeon HD 7660G, 6 compute units, 685MHz; 4GB RAM

© Copyright 2012 HSA Foundation. All Rights Reserved.
VASCULAR IMAGE ENHANCEMENT
EMBEDDED MEDICAL WORKLOAD
VASCULATURE IMAGE ENHANCEMENT

- Automatic image enhancement over volumetric CT/MRI data
- Used to diagnose vascular pathologies
HSA increases performance and reduces energy for Medical Image Analysis by offloading data parallel compute to GPU.
EASE OF PROGRAMMING

CODE COMPLEXITY VS. PERFORMANCE
LINES-OF-CODE AND PERFORMANCE FOR DIFFERENT PROGRAMMING MODELS

(Exemplary ISV “Hessian” Kernel)

Serial CPU  TBB  Intrinsics+TBB  OpenCL™-C  OpenCL™-C++  C++ AMP  HSA Bolt

Copy-back  Algorithm  Launch  Algorithm  Algorithm  Algorithm  Algorithm  Algorithm

Algorithm  Algorithm  Algorithm  Compile  Compile  Compile  Compile  Compile

Launch  Algorithm  Algorithm  Copy  Copy  Copy  Copy  Copy

Performance

AMD A10-5800K APU with Radeon™ HD Graphics – CPU: 4 cores, 3.80GHz (4.20GHz Turbo); GPU: AMD Radeon HD 7660D, 6 compute units, 800MHz; 4GB RAM.

Software – Windows 7 Professional SP1 (64-bit OS); AMD OpenCL™ 1.2 AMD APP (937.2); Microsoft Visual Studio 11 Beta.

© Copyright 2012 HSA Foundation. All Rights Reserved.
THE HSA FUTURE

PARALLEL PROCESSING FOR ALL OF US!

Highly productive programmers
+ Scalable performance
+ Power efficiency

= AMAZING USER EXPERIENCES
THE HSA FOUNDATION MEMBERSHIP

Founders

Promoters

LG Electronics

Supporters

Contributors

Academic

Associates