AMD's Unified CPU & GPU Processor Concept

Advanced Seminar *Computer Engineering*

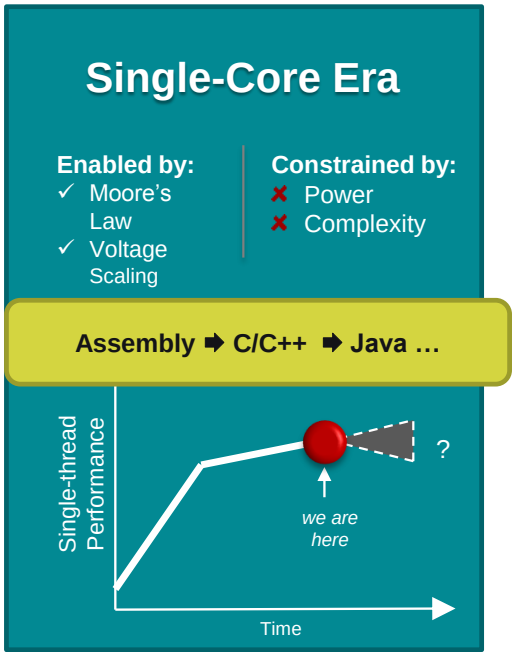
Sven Nobis

Institute of Computer Engineering (ZITI)
University of Heidelberg

February 5, 2014

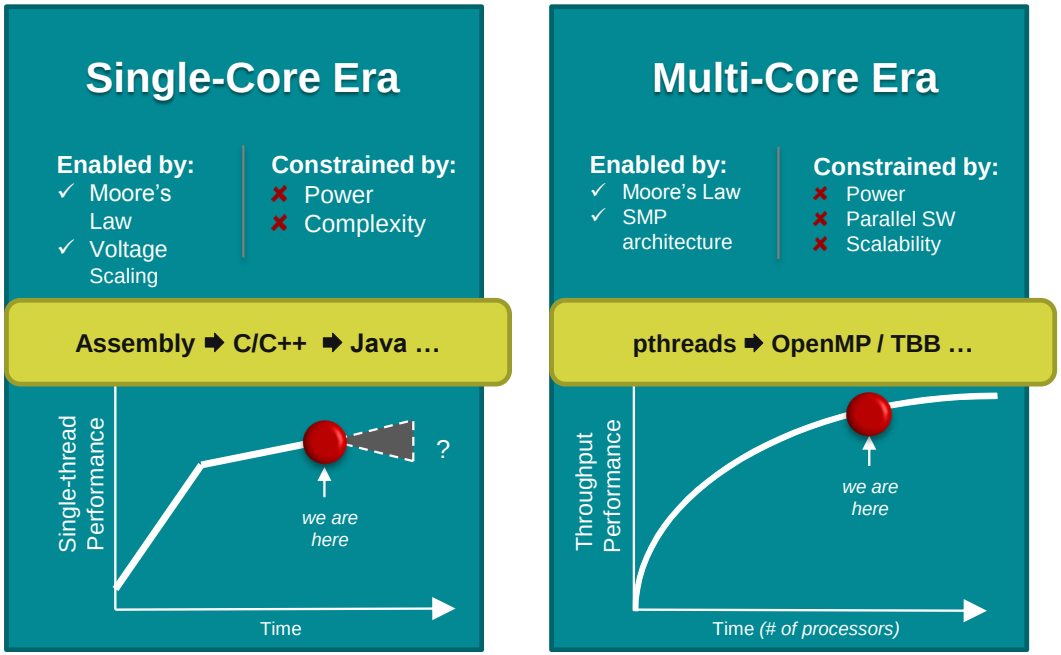
Overview

Previous: Single-Core Era



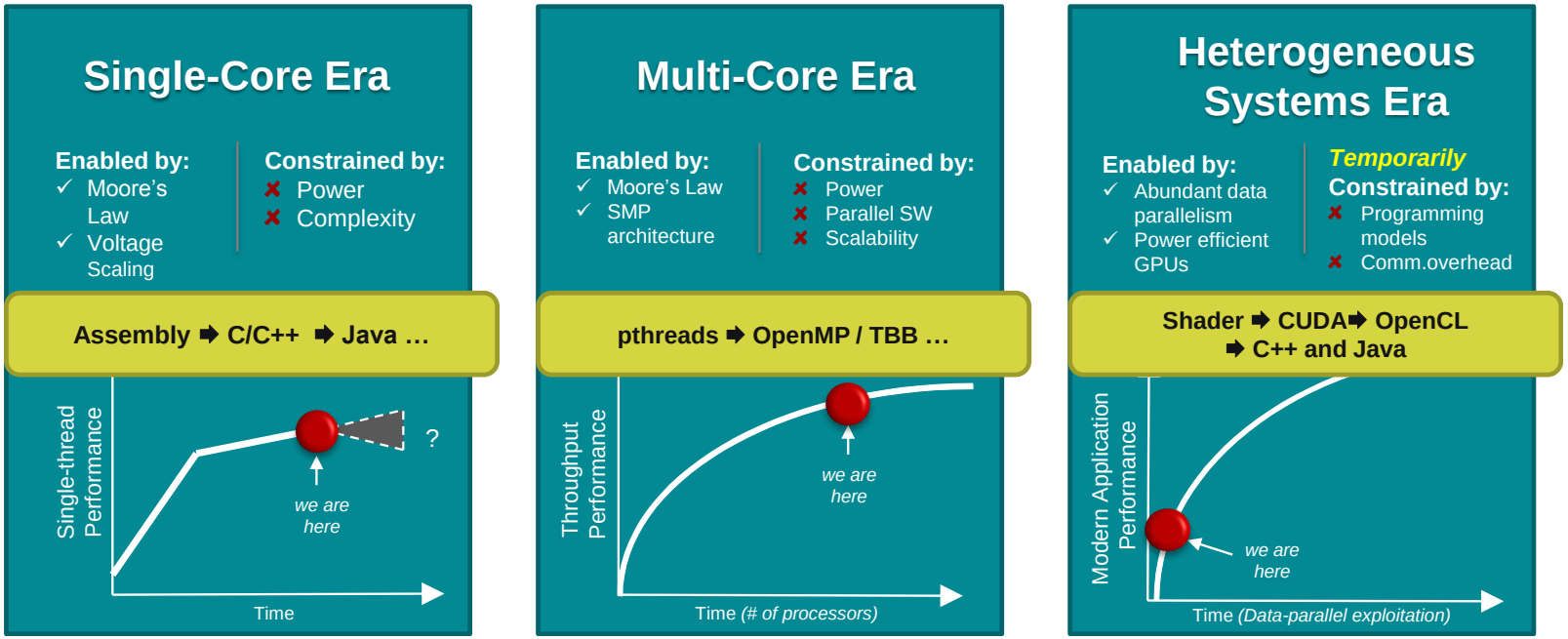
[8, P. 5]

Today: Multi-Core Era



[8, P. 5]

Today till future: Heterogeneous System Era

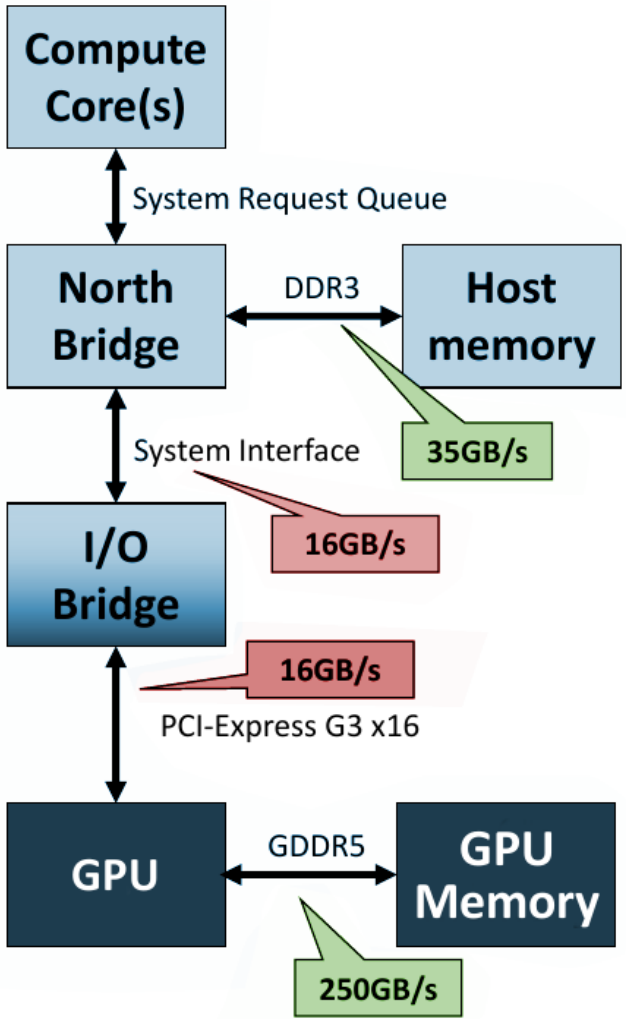


[8, P. 5]

Introduction

- Today's problems on CPU / GPU programming
  - programmability barrier
  - communication costs
- Solution
  - AMD's Unified CPU & GPU Processor Concept?

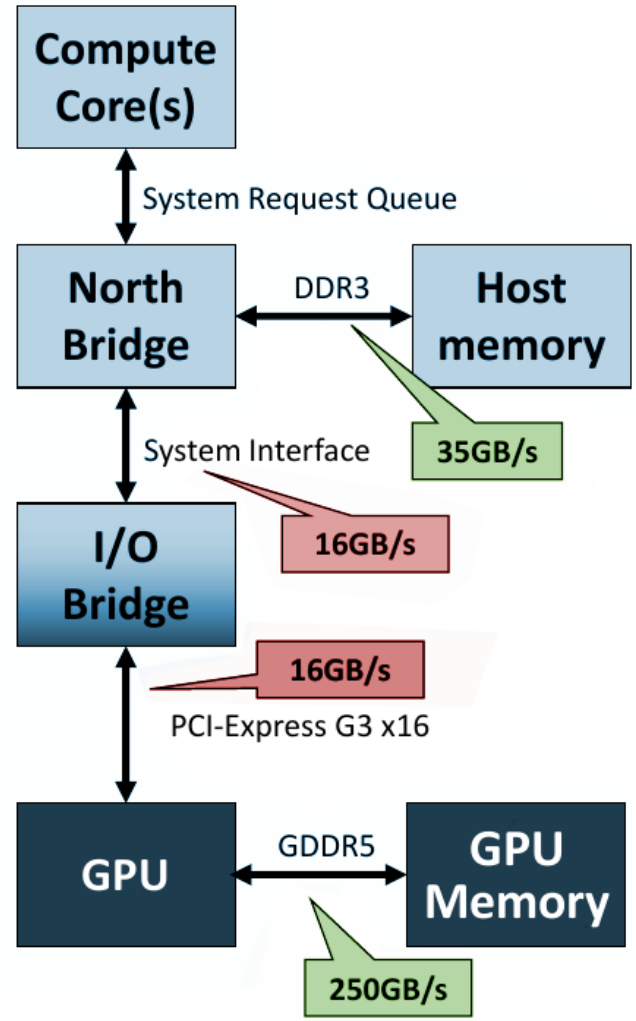
    → **Heterogeneous System Architecture (HSA)**



[3, P. 4]

Introduction

- Today's problems on CPU / GPU programming
  - programmability barrier
  - communication costs
- Solution
  - AMD's Unified CPU & GPU Processor Concept?
    - → **Heterogeneous System Architecture (HSA)**



[3, P. 4]

Overview

CPU vs. GPU

CPU:

LCU

Latency Compute Unit

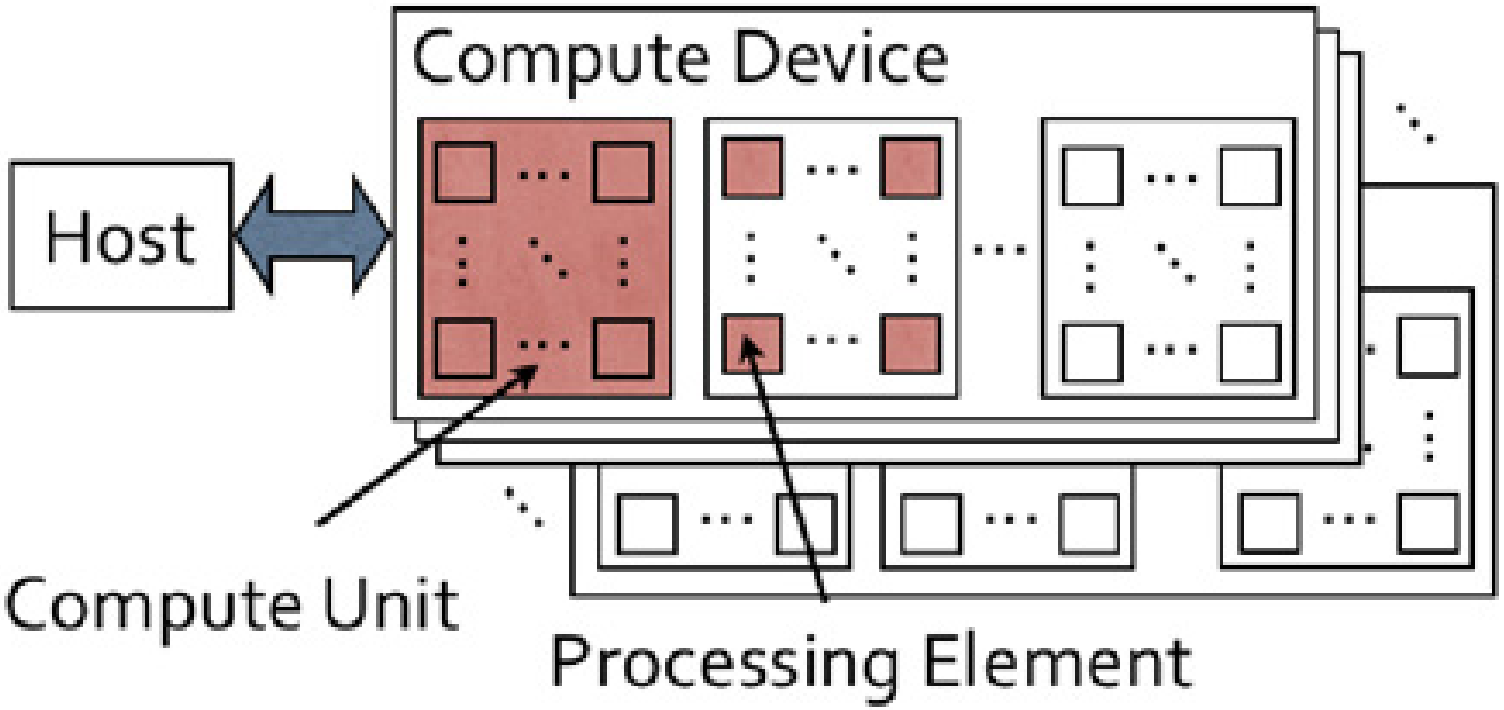
GPU:

TCU

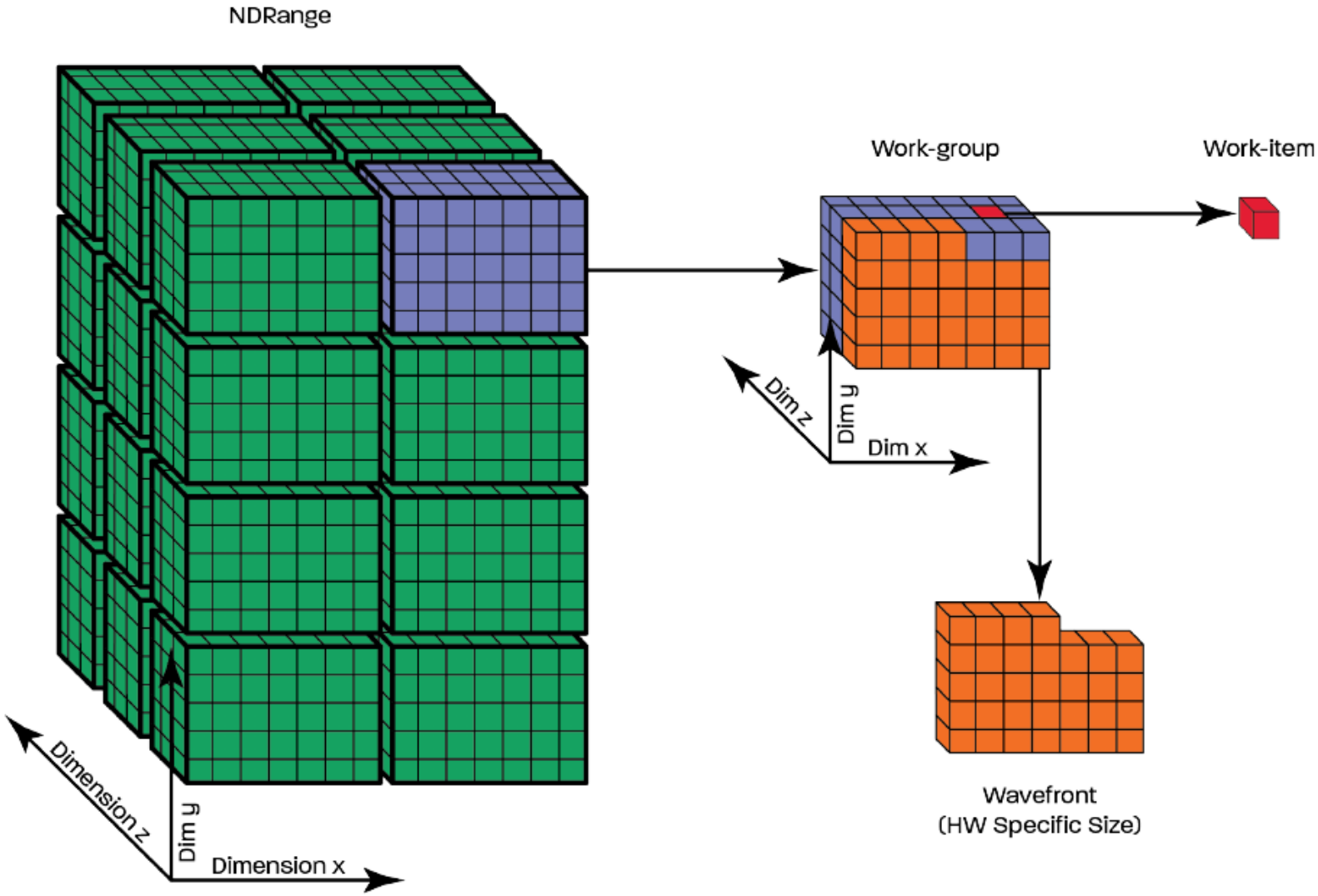
Throughput Compute Unit

OpenCL & CUDA

- Both well-established platforms for GPU programming
- Compute Unified Device Architecture (CUDA)
  - Proprietary
  - Only for NVIDIA GPUs
- Open Computing Language (OpenCL)
  - Open standard
  - ATI, NVIDIA, Intel, ...
  - Not only GPUs



[10]



[5, P. 11]

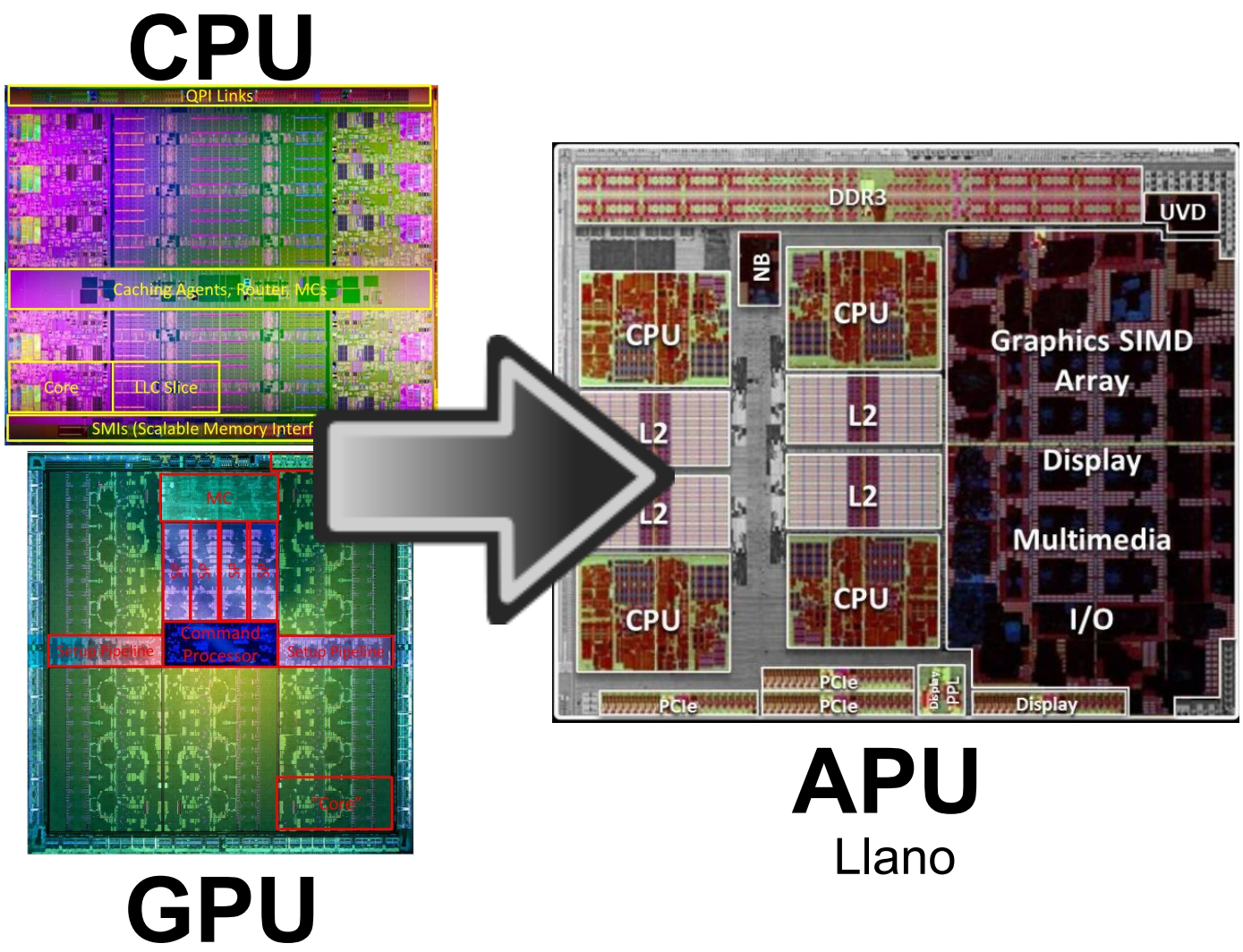
Overview

Related Work

- In CUDA [4]
  - Unified Virtual Addressing (UVA) in CUDA 4
  - Unified Memory in CUDA 6
  - → Developer view to the memory
    - Implicit copy & pinning
- In OpenCL
  - Shared Virtual Memory
- Copy is still necessary (for fast access)

Overview

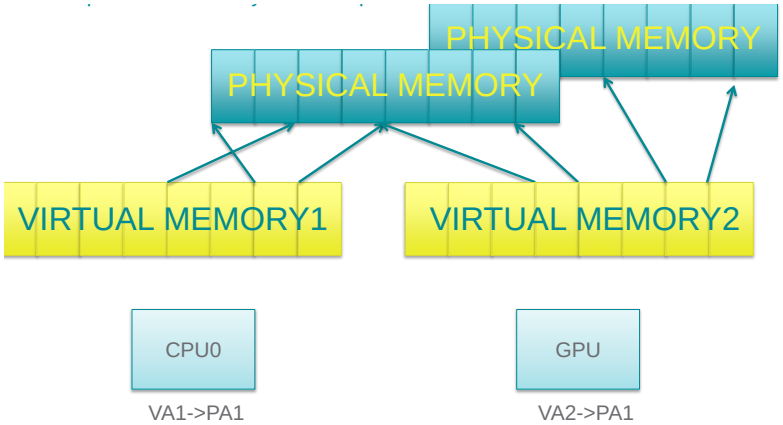
CPU and GPU cores in a single die



[3, P. 2] [7, P. 7]

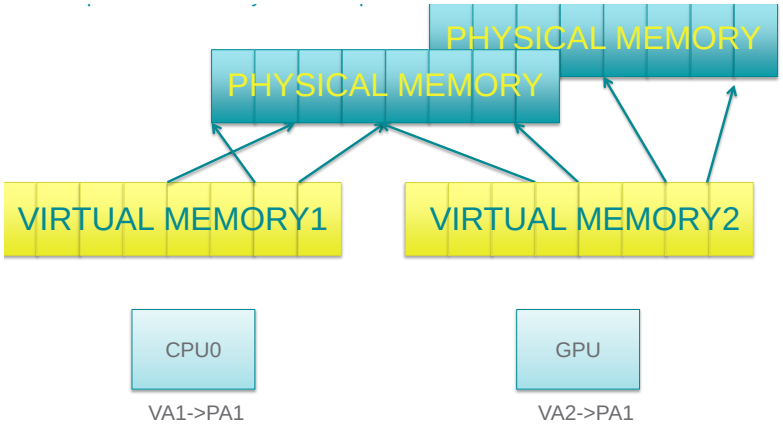
hUMA: Heterogeneous Unified Memory Access

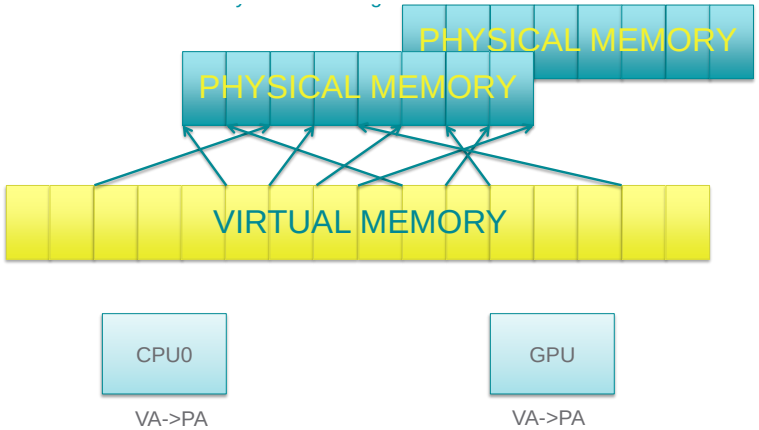
- Today: Non-Uniform Memory Access
  - Different/partitioned physical memory per compute unit
  - Multiple virtual memory address spaces
- hUMA: Heterogeneous Unified Memory Access
  - Same physical memory
  - Same virtual memory for all compute units



hUMA: Heterogeneous Unified Memory Access

- Today: Non-Uniform Memory Access
  - Different/partitioned physical memory per compute unit
  - Multiple virtual memory address spaces
- hUMA: Heterogeneous Unified Memory Access
  - Same physical memory
  - Same virtual memory for all compute units





[2, P. 7], [2, P. 8]

hUMA: Heterogeneous Unified Memory Access (2)

- **Required:** hUMA Memory Controller
- **Features**
  - Shared page table support
    - Same large address space as the CPU
    - Page faulting
  - Coherent memory regions
    - Fully coherent shared memory model
    - Like on today's SMP CPU systems

Overview

Concepts

- Unified Address Space
  - Already mentioned with hUMA
- Unified Programming Model
- Queuing
- HSA Intermediate Language

Concepts
Unified Programming Model

- Current programming models
  - → Treating the GPU as a remote processor
- Extending existing concepts to use HSA
  - Programming languages like C++
  - *Task parallel* and *data parallel* APIs like `C++ AMP`
- Stay in developers environment

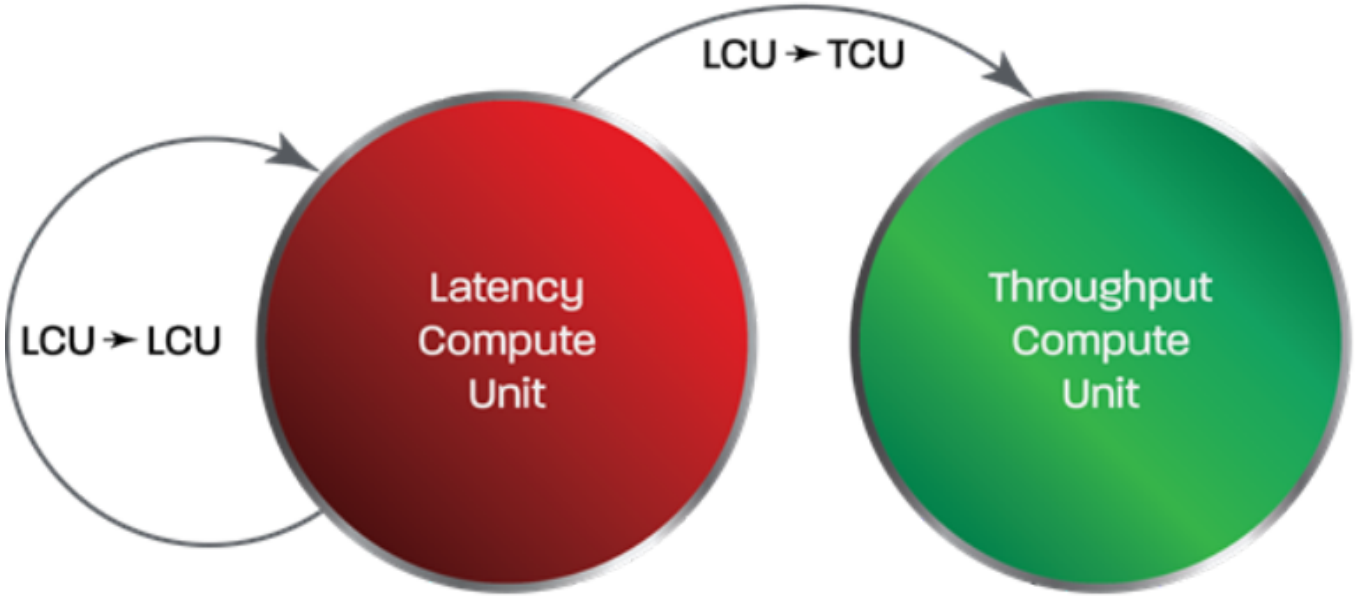
```
#include <iostream>
#include <amp.h>
using namespace concurrency;
int main() // "Hello World" in C++ AMP
{
    int v[11] = {'G', 'd', 'k', 'k', 'n', 31, 'v', 'n', 'q', 'k', 'c'};

    array_view<int> av(11, v);
    parallel_for_each(av.extent, [=](index<1> idx) restrict(amp)
    {
        av[idx] += 1;
    });

    for(unsigned int i = 0; i < av.extent.size(); i++)
        std::cout << static_cast<char>(av(i));
}
```

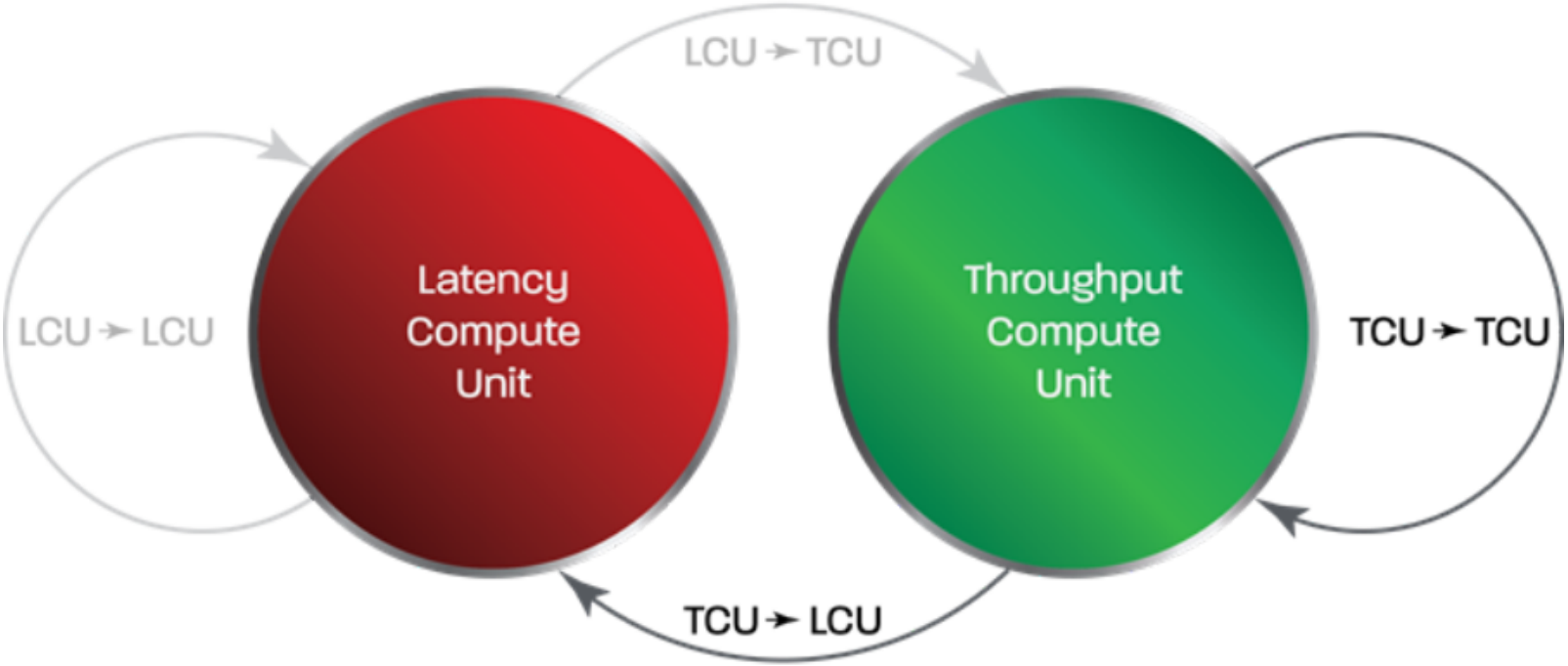
[6]

Concepts
Queuing - Current



[5, P.9]

Concepts
Queuing - New!



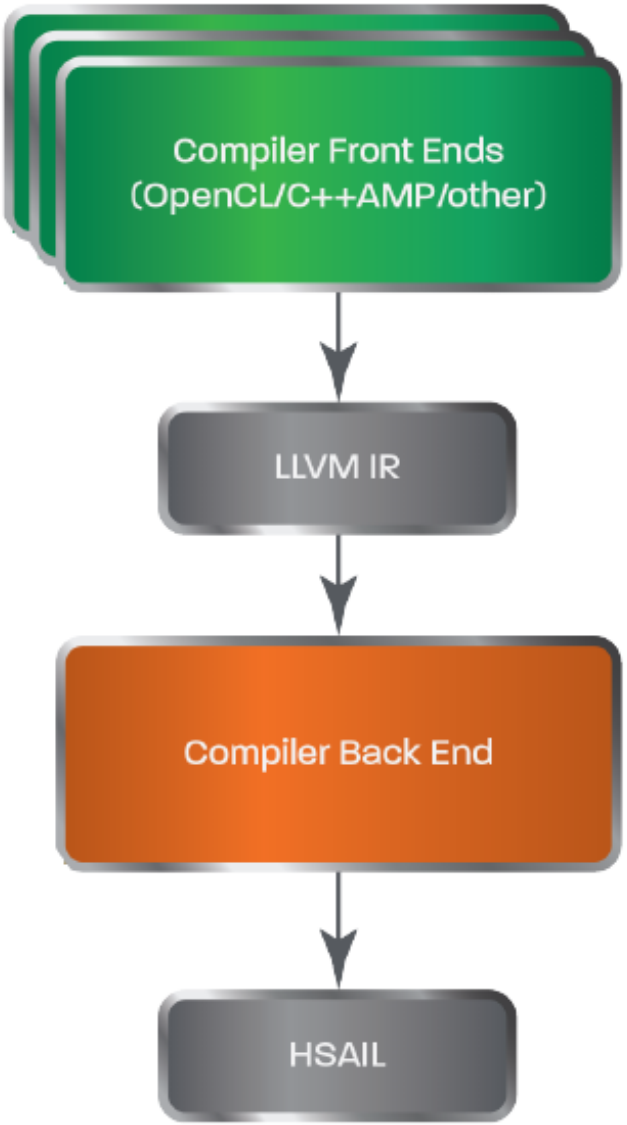
[5, P.9]

- HSAIL: HSA Intermediate Language
  - Bytecode
  - Designed for data parallel programming
  - GPU independent
- Generated by compilation stack *(later)*
- Bytecode is compiled at runtime
  - to the *Hardware Instruction Set* of the current device
- Execution Model is similar to OpenCL

System Components

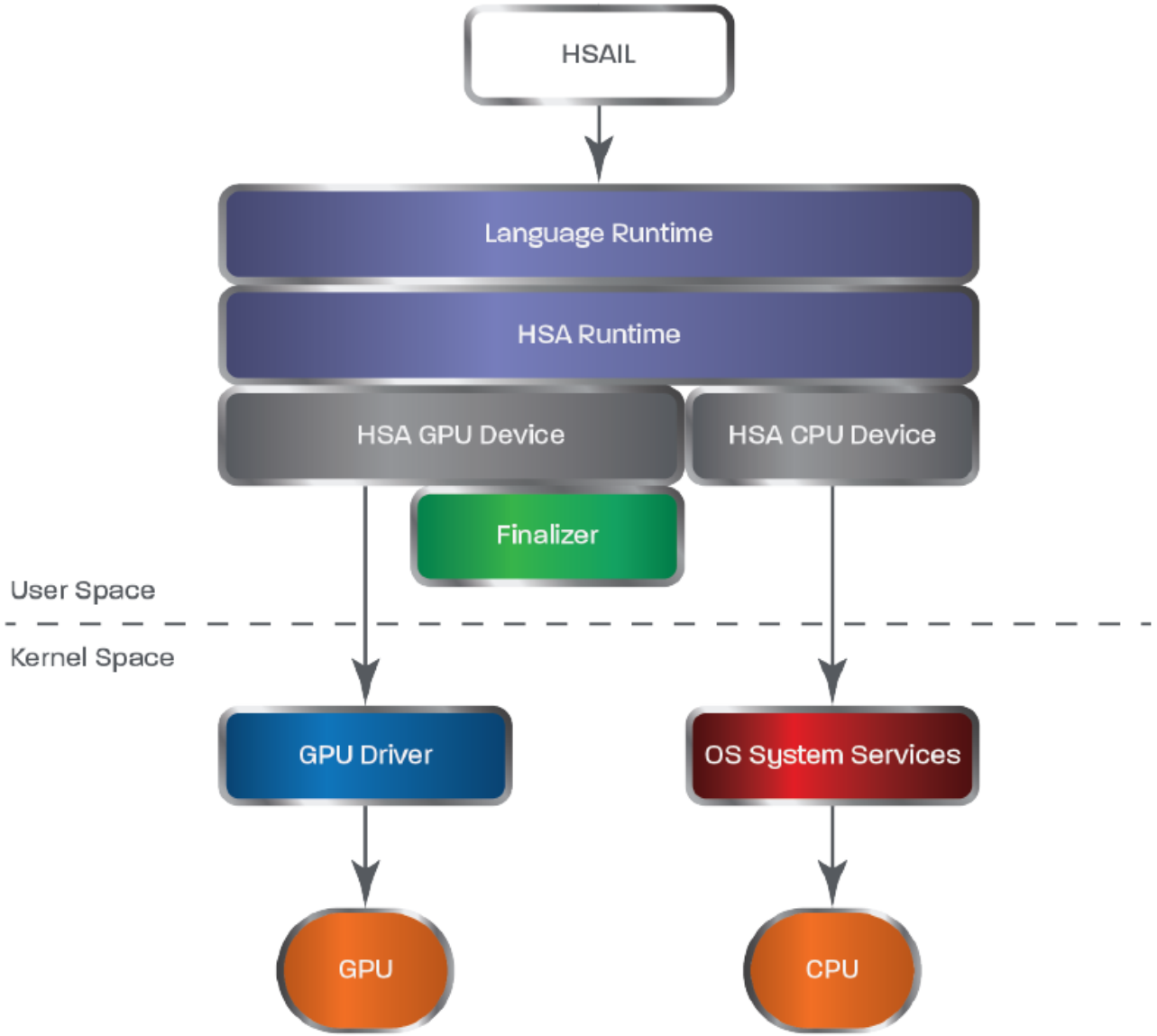
- APU
- Software stack
  - Compilation Stack
  - Runtime Stack
  - System (Kernel) Software

System Components
Compilation Stack



[5, P. 15]

System Components
Runtime-Stack



[5, P. 16]

Development Tools

- OpenCL
- C++ AMP: C++ Accelerated Massive Parallelism
- BOLT Library
- Aparapi

Development Tools
OpenCL

"HSA is an optimized platform architecture for OpenCL
- Not an alternative to OpenCL" [8, P. 13]

- OpenCL on HSA will benefit from its features

Simple Example:

```
#include <bolt/sort.h>
#include <vector>
#include <algorithm>

void main()
{
    // generate random data (on host)
    std::vector<int> a(1000000);
    std::generate(a.begin(), a.end(), rand);

    // sort, run on best device
    bolt::sort(a.begin(), a.end());
}
```

[9, P.5]

Simple Example:

```
#include <bolt/transform.h>
#include <vector>

struct SaxpyFunctor
{
   float _a;
   SaxpyFunctor(float a) : _a(a) {};

   float operator() (const float &xx, const float &yy) restrict(cpu,amp)
   {
         return _a * xx + yy;
   };
};

void main() {
   SaxpyFunctor s(100);
   std::vector<float> x(1000000); // initialization not shown
   std::vector<float> y(1000000); // initialization not shown
   std::vector<float> z(1000000);

   bolt::transform(x.begin(), x.end(), y.begin(), z.begin(), s);
};
```

[9, P.6]

Overview

Conclusion

- Interesting concept
  - Simplifies development
  - Open up new possibilities
- Open platform
- In heavy development
  - Missing hardware with hUMA
    - → Outlook
  - Software components not ready

→ A lot of potential

Outlook

- Middle of January 2014:
  - Kaveri APU is available [1]
  - Desktop APU
  - Support for
    - hUMA
    - Queuing
  - Can connect both DDR3 and GDDR5 [11]
- Server APU follows:
  - Berlin
  - ARM-Based: Seattle

[11]

References I

[1] Benz, Benjamin: *AMD fordert mit Kaveri Intels Core i5 heraus*. Heise Online. http://heise.de/-2085447. Version: Januar 2014

[2] Bratt, Ian: *HSA Queueing*. HOT CHIPS 2013. http://www.slideshare.net/hsafoundation/hsa-queuing-hot-chips-2013. Version: August 2013

[3] Fröning, Holger: *Lecture 02 – CUDA Programming*. Lecture: GPU Computing, 2013

[4] Harris, Mark: *Unified Memory in CUDA 6*. http://devblogs.nvidia.com/parallelforall/unified-memory-in-cuda-6/. Version: November 2013

References II

[5] Kyriazis, George: A Heterogeneous System Architecture: Technical Review / HSA Foundation. AMD, August 2012. – Forschungsbericht. – Rev. 1.0 S.

[6] Moth, Daniel: *"Hello world" in C++ AMP*. http://blogs.msdn.com/b/nativeconcurrency/archive/2012/03/04/quot-hello-world-quot-in-c-amp.aspx. Version: März 2012

[7] Rogers, Phil: *THE PROGRAMMER'S GUIDE TO THE APU GALAXY*. AMD Fusion Developer Summit. http://www.slideshare.net/hsafoundation/afds-keynote-the-programmers-guide-to-the-apu-gal Version: Juni 2011

References III

[8] Rogers, Phil: *Heterogeneous System Architecture Overview*. HOT CHIPS 2013. http://de.slideshare.net/hsafoundation/hsa-intro-hot-chips2013-final. Version: August 2013

[9] Sander, Ben: *BOLT: A C++ Template Library for HSA*. AMD Fusion Developer Summit. http://www.slideshare.net/hsafoundation/bolt-for-hsa-by-ben-sanders. Version: Juni 2012

[10] Staff, AMD: *OpenCL™ and the AMD APP SDK v2.4*. http://developer.amd.com/resources/documentation-articles/articles-whitepapers/opencl-and-the-amd-app-sdk-v2-4/. Version: April 2011

References IV

[11] Windeck, Christof: *AMD Kaveri: Feinheiten aus den Datenblättern*. Heise Online. `http://heise.de/-2088349`. Version: Januar 2014