Scaling Towards Exaflops with Heterogeneous Architectures

Summeracademy in Leysin
AG 2 – Green Computing

Arne Hensel

University of Hanover

August 2016

Outline

Most algorithms are like baking recipies, tailored for a single person / processor:

- First, do A,
- Then do B,
- Continue with C,
- And finally complete by doing D.

Produce one cupcake withinn one clockcycle:

Figure 1: Relatively slow cupcake production.

Production of 10^{18} cupcakes

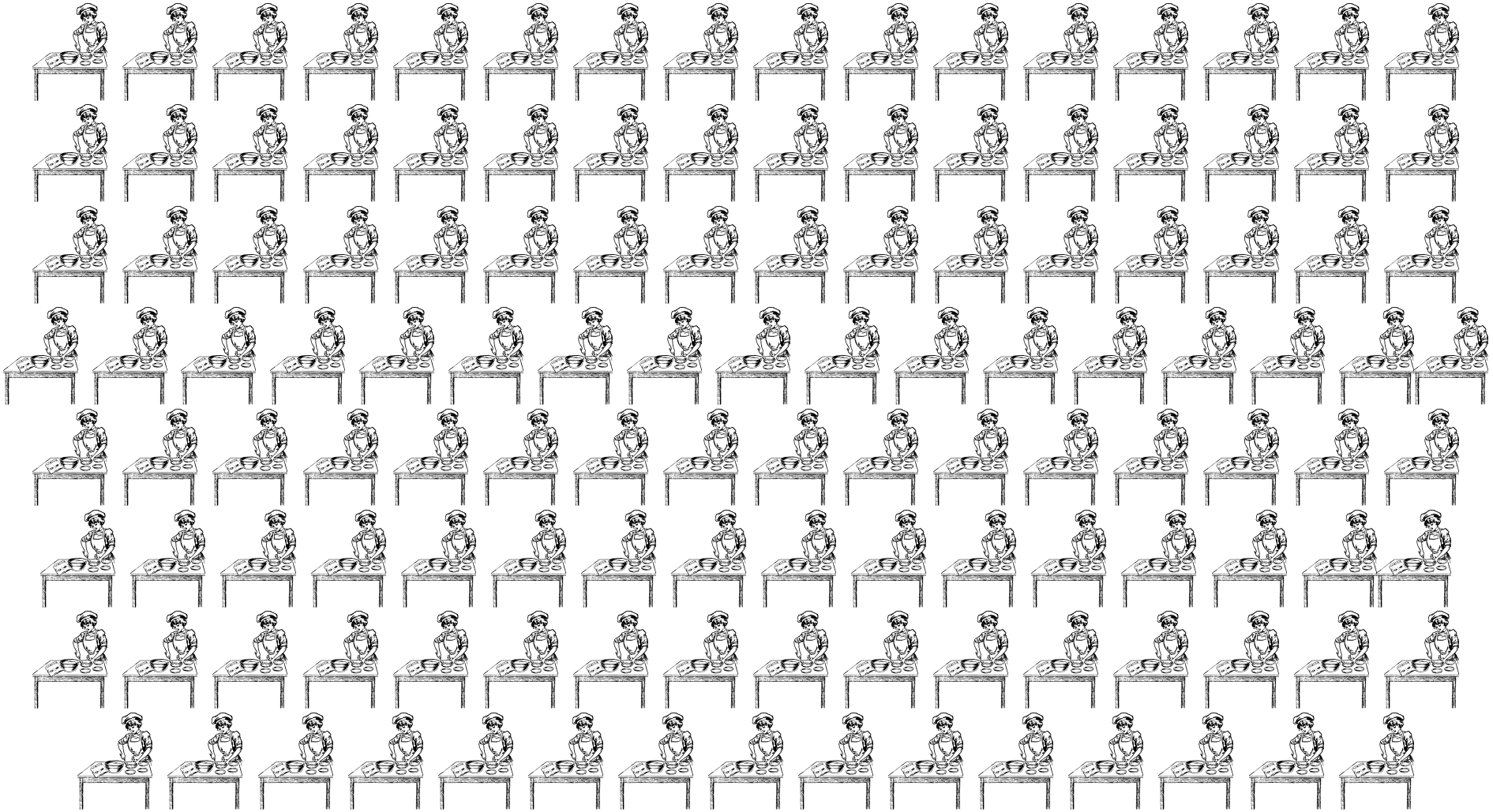


Figure 2: Incredibly fast production of exa-cupcakes

"Homogeneous" [Bla96]

- The hardware of each processor guarantees the same storage representation and the same results for operations on floating point numbers.

"Homogeneous" [Bla96]

- The hardware of each processor guarantees the same storage representation and the same results for operations on floating point numbers.
- If a floating point number is communicated between processors, the communication layer guarantees the exact transmittal of the floating point value.

"Homogeneous" [Bla96]

- The hardware of each processor guarantees the same storage representation and the same results for operations on floating point numbers.
- If a floating point number is communicated between processors, the communication layer guarantees the exact transmittal of the floating point value.
- The software (operating system, compiler, compiler options) on each processor also guarantees the same storage representation and the same results for operations on floating point numbers.

Speeding up with: [VM15]

- Specialized processors
  - One chip - one core

Speeding up with: [VM15]

- Specialized processors
  - One chip - one core
- Multicore-processors
  - One chip - multiple cores

Speeding up with: [VM15]

- Specialized processors
  - One chip - one core
- Multicore-processors
  - One chip - multiple cores
- Multiprocessors
  - Multiple chips - multiple cores

Speeding up with: [VM15]

- Specialized processors
  - One chip - one core
- Multicore-processors
  - One chip - multiple cores
- Multiprocessors
  - Multiple chips - multiple cores
- Cache size
  - 45nm POWER7 with 32MB cache
  - 32nm POWER7+ with 80MB cache
  - 22nm POWER8 with 96MB cache

Speeding up with: [VM15]

- Specialized processors
  - One chip - one core
- Multicore-processors
  - One chip - multiple cores
- Multiprocessors
  - Multiple chips - multiple cores
- Cache size
  - 45nm POWER7 with 32MB cache
  - 32nm POWER7+ with 80MB cache
  - 22nm POWER8 with 96MB cache
- Interconnect bandwidth
  - Replace PCIe with NVLink (up to 12x faster)

Speeding up with: [VM15]

- Specialized processors
  - One chip - one core
- Multicore-processors
  - One chip - multiple cores
- Multiprocessors
  - Multiple chips - multiple cores
- Cache size
  - 45nm POWER7 with 32MB cache
  - 32nm POWER7+ with 80MB cache
  - 22nm POWER8 with 96MB cache
- Interconnect bandwidth
  - Replace PCIe with NVLink (up to 12x faster)

Homogeneous: one type of core

Moore's Law

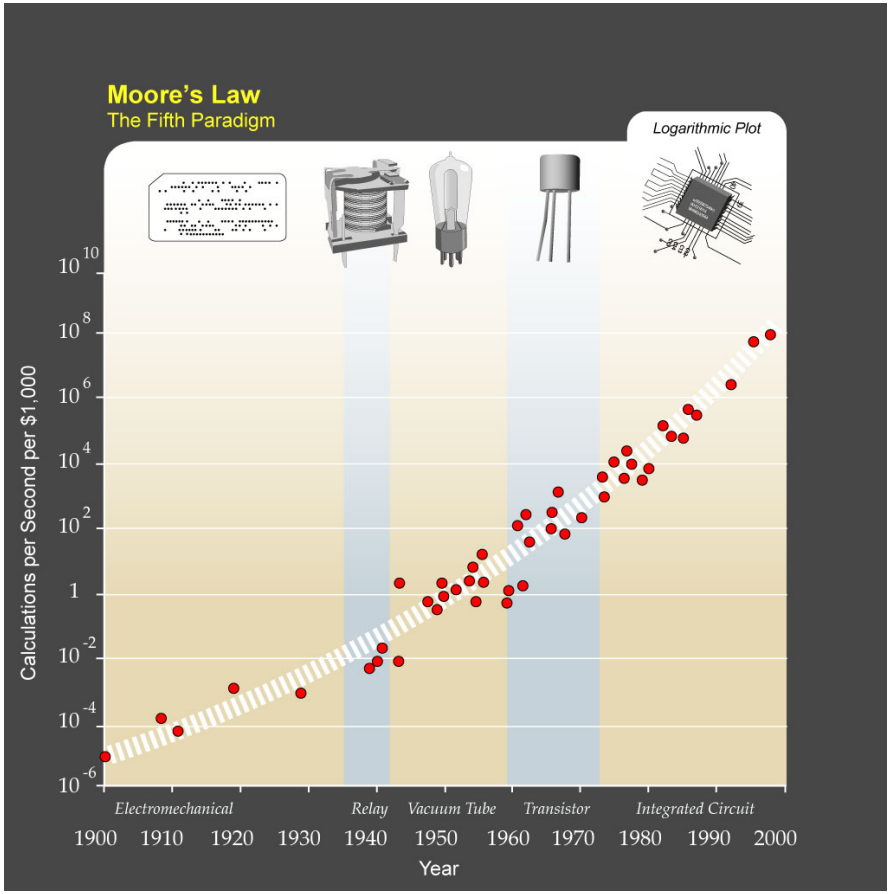


Figure 3: Moore's Law (Image credit: Von Courtesy of Ray Kurzweil and Kurzweil Technologies, Inc. - en:Image:PPTMooresLawai.jpg, CC BY 1.0, https://commons.wikimedia.org/w/index.php?curid=1273707)

Increasing transistor density

$\Leftrightarrow$

Increasing calculation speed

?

Technical issues: miniaturization

- Physical problems with transistor density

Technical issues: miniaturization

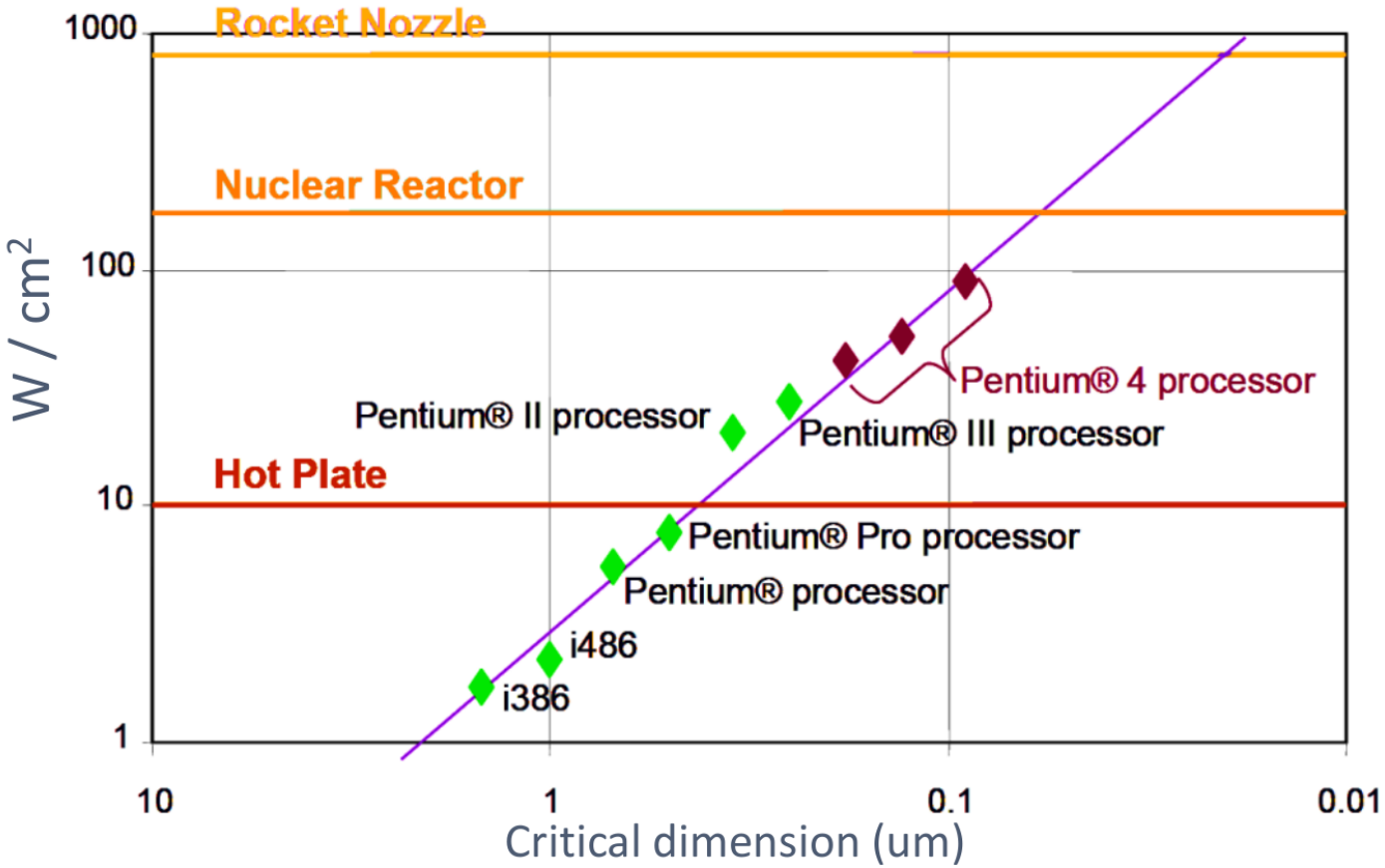


Figure 4: Power Wall (Image credit: G. Taylor, "Energy Efficient Circuit Design and the Future of Power Delivery")

What is about parallelism?

$$P_d \propto f_{clock}^3$$

[1] Brodtkorb et al. State-of-the-art in heterogeneous computing, 2010

What is about parallelism?

$$P_d \propto f_{clock}^3$$

- Single-core: 100 % power, 100 % frequency, 100 % performance

[1] Brodtkorb et al. State-of-the-art in heterogeneous computing, 2010

What is about parallelism?

$$P_d \propto f_{clock}^3$$

- Single-core: 100 % power, 100 % frequency, 100 % performance
- Dual-core: 100 % power, 85 % frequency, 90 % performance each[1]

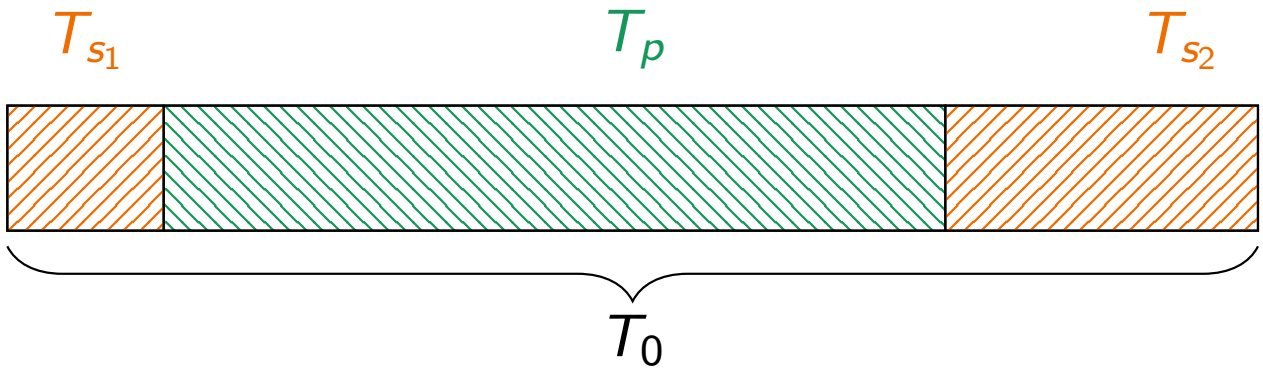
[1] Brodtkorb et al. State-of-the-art in heterogeneous computing, 2010

Software issues: parallelism and Amdahl's Law

- Speed up:

$$S = \frac{T_0}{\frac{T_p}{l} + T_s} = \frac{1}{\frac{T_p}{T_0 \cdot l} + \frac{T_s}{T_0}} = \frac{1}{\frac{\beta}{l} + (1 - \beta)} \leq \frac{T_0}{T_s}$$

- l . . . level of parallelism (available PUs)
- T_0 . . . time for processing
- T_p . . . time for parallel-processing
- T_s . . . time for serial-processing
- β . . . fraction of the computationtime influenced by the parallelism (T_p/T_0)



Software issues: parallelism and Amdahl's Law

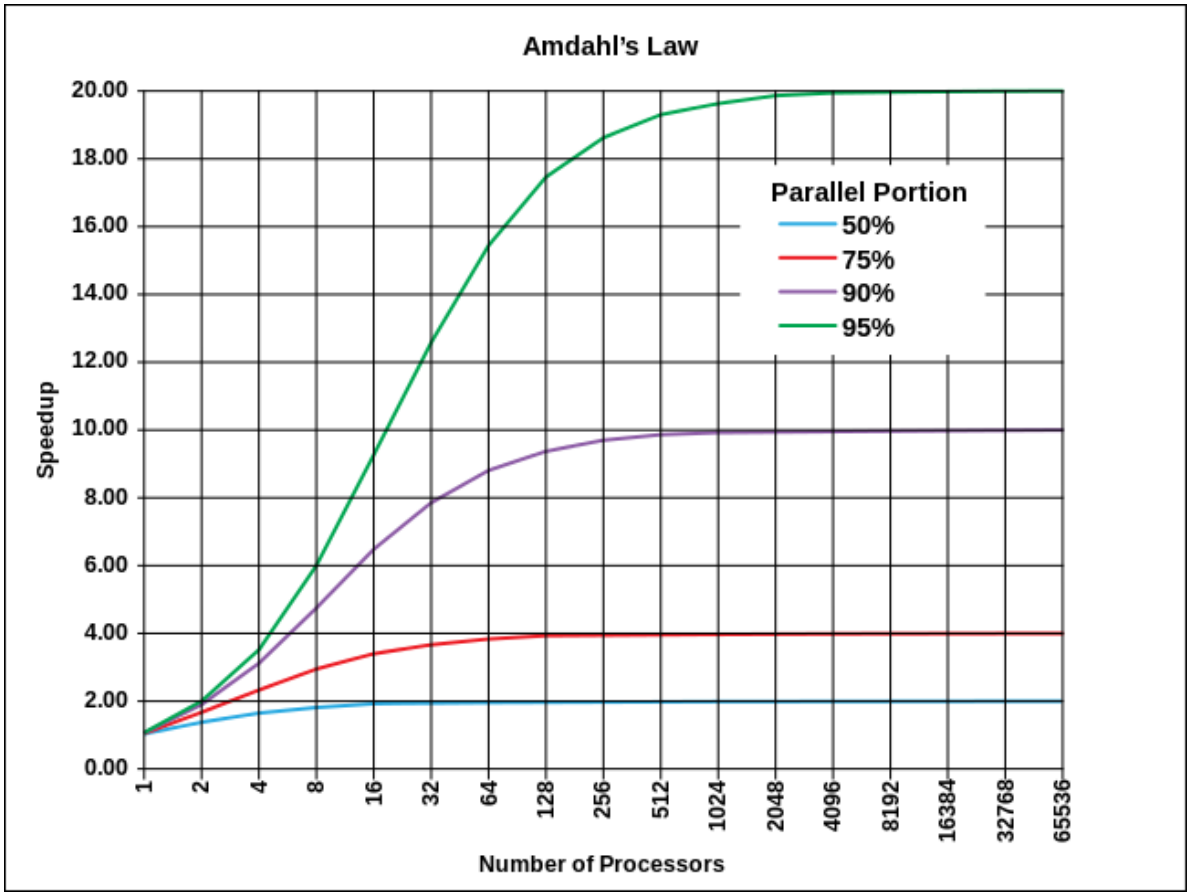


Figure 5: Amdahl's Law (Image credit: By Daniels220 at English Wikipedia - Own work based on: File:AmdahlsLaw.png, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=6678551

Technical issues: miniaturization

- Physical problems with transistor density
- Wire-delays due to more transistors

Technical issues: miniaturization

- Physical problems with transistor density
- Wire-delays due to more transistors
- Von-Neumann-Bottleneck

Technical issues: miniaturization

- Physical problems with transistor density
- Wire-delays due to more transistors
- Von-Neumann-Bottleneck

. . . easy to program, but the homogenenous approach will fail at some point. . .

Challenges with homogeneous exascale-computing

"Also important is the likely degradation of reliability due to the multiplicative factors of MTBF[2] and scale of components." [Zim03]

[2] Mean Time Between Failures

Challenges with homogeneous exascale-computing

"Also important is the likely degradation of reliability due to the multiplicative factors of MTBF[2] and scale of components." [Zim03]

"While reliability is a major concern for Exascale, another key-challenge is to minimize energy consumption, both for economic and environmental reasons." (Estimated threshold $\approx$ 20MW) [HR15]

[2] Mean Time Between Failures

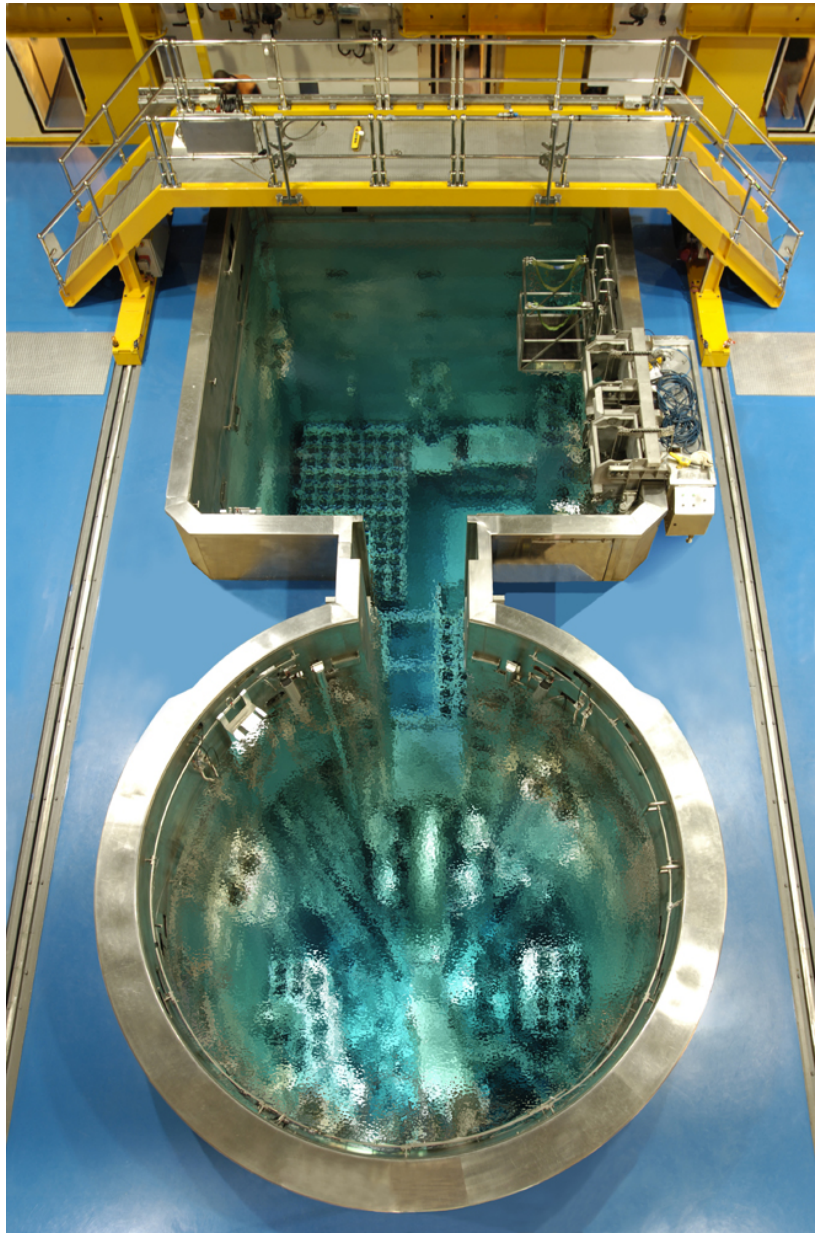


Figure 6: Nuclear power plant for your computer. (Image Credit: Ansto. http://www.ansto.gov.au/)

Your weekness is our strength!

- CPU: latency-critical applications

Your weekness is our strength!

- CPU: latency-critical applications
- GPU: throughput-critical applications

Your weekness is our strength!

- CPU: latency-critical applications
- GPU: throughput-critical applications
- Wastage of energy while allocating tasks
  - Homogeneous: CPU stays idle, GPU works
  - Homogeneous: GPU stays idle, CPU works
  - Heterogeneous: work-division

Your weekness is our strength!

- CPU: latency-critical applications
- GPU: throughput-critical applications
- Wastage of energy while allocating tasks
  - Homogeneous: CPU stays idle, GPU works
  - Homogeneous: GPU stays idle, CPU works
  - Heterogeneous: work-division

    top 15 systems in Green500 are heterogeneous-systems [VM15]

Your weekness is our strength!

- CPU: latency-critical applications
- GPU: throughput-critical applications
- Wastage of energy while allocating tasks
  - Homogeneous: CPU stays idle, GPU works
  - Homogeneous: GPU stays idle, CPU works
  - Heterogeneous: work-division
    top 15 systems in Green500 are heterogeneous-systems [VM15]
- work division at run-time (long lists: GPU; short lists: CPU)

"Heterogeneous"

. . . not homogeneous . . .

"Heterogeneous"

. . . not homogeneous . . .

More specific: hardware designed for a certain task

containing different types of cores

Heterogeneous cupcake production

Figure 7: Heterogeneous architecture of a bakery. (Image Credit: Kathrin Blum)

"Heterogeneous"

- Fused/integrated systems
  - Example: APU[3] with CPU-GPU on one chip

[3] Accelerated Processing Unit

[4] Peripheral Component Interconnect Express

"Heterogeneous"

- Fused/integrated systems
  - Example: APU[3] with CPU-GPU on one chip
- Discrete systems
  - Example: CPU-GPU connection via PCIe[4] bus.

[3] Accelerated Processing Unit

[4] Peripheral Component Interconnect Express

Good mixtures

CPU-GPU

Scalar *and* vector processor for small data sets/certain algorithms *and* large data sets/numerical computation.

Good mixtures

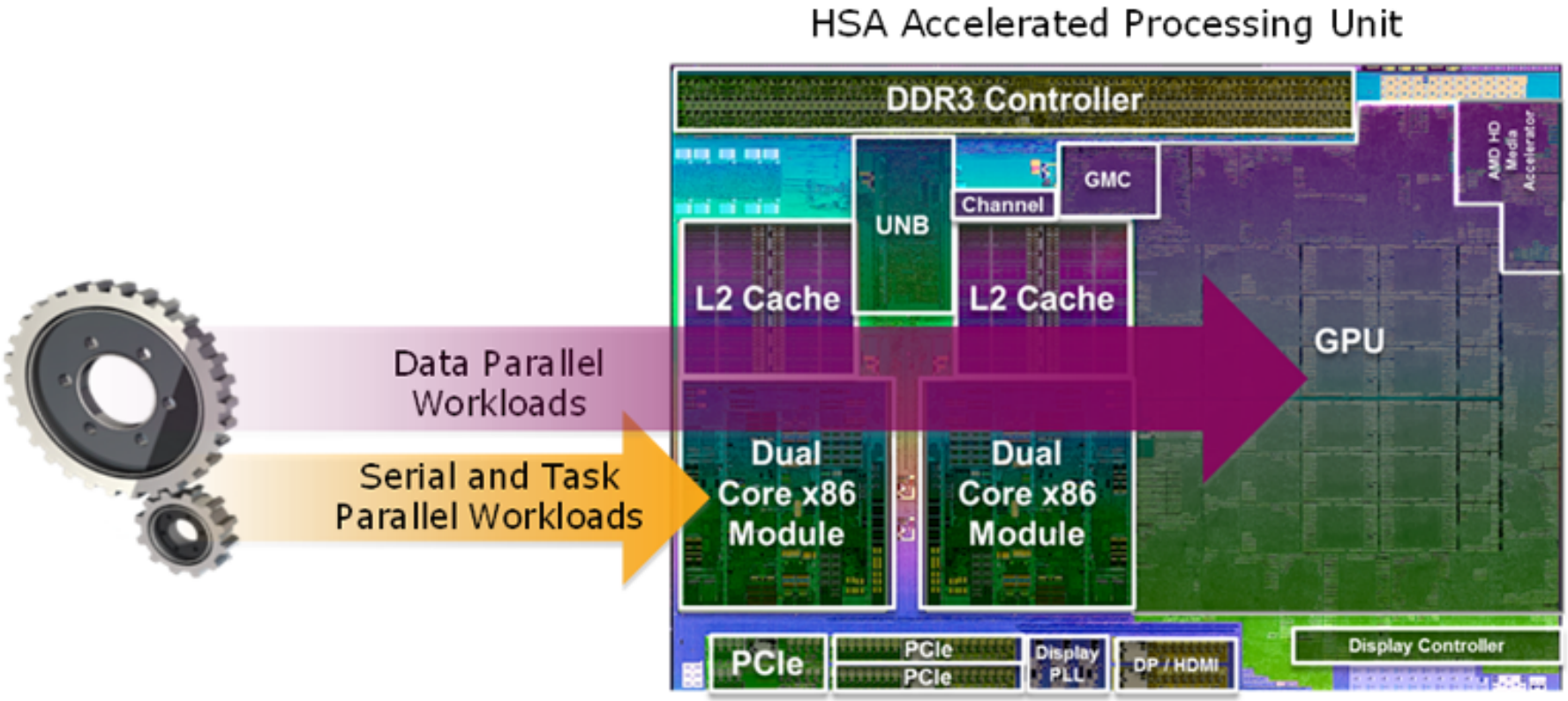


Figure 8: HSA APU. (Image credit: AMD, http://amd- dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/HSAAcceleratedProcessingUnit.png

Good mixtures

The shared memory withinn cell-processor

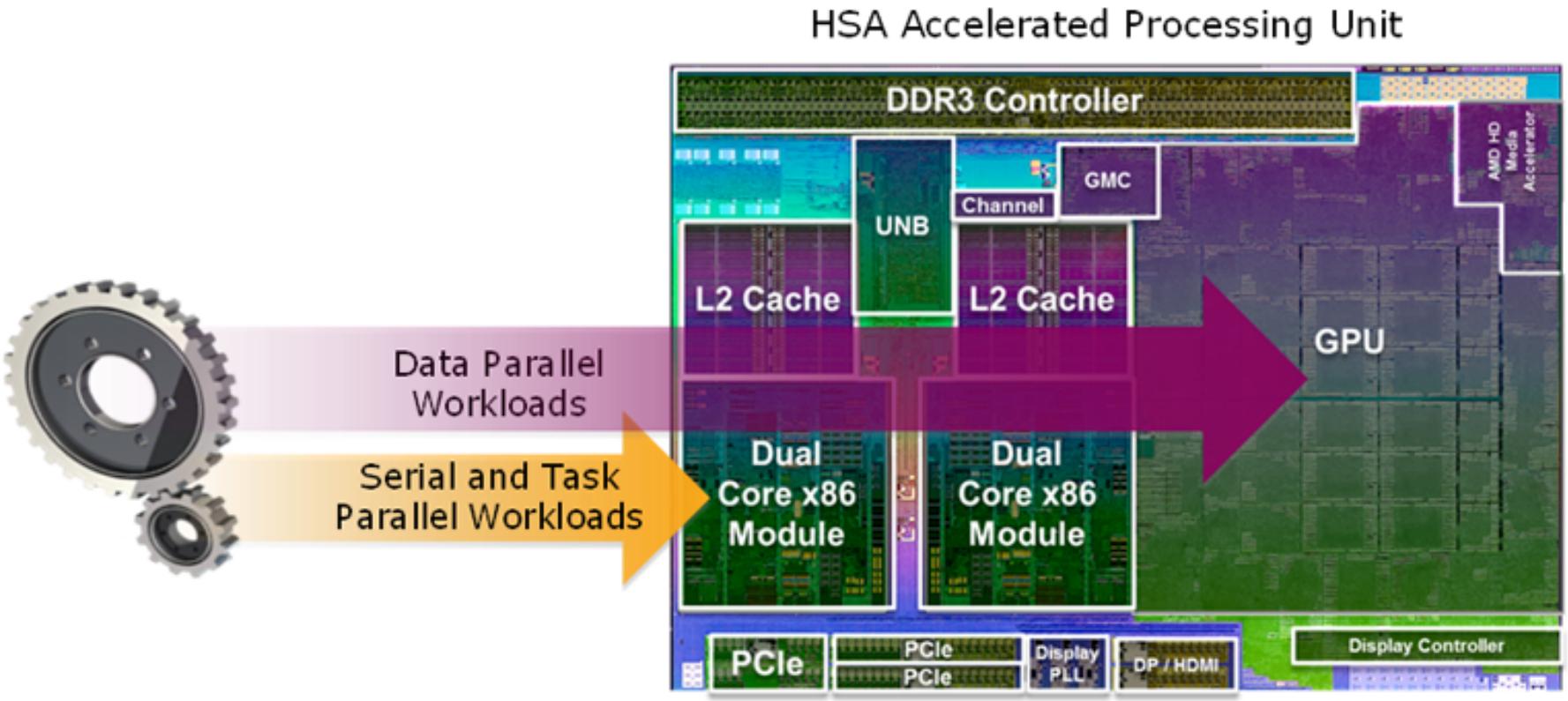


Figure 9: Shared memory. (Image credit:

"Bad" "mixtures"

Intel Larrabee [Rol09]

- GPCPU: graphics processing with CPUs
- scalar and vector unit
- f_{Peak}: 2TFlops

▸ To: Larrabee Revealed

"Bad" "mixtures"

Intel Larrabee [Rol09]

- GPCPU: graphics processing with CPUs
- scalar and vector unit
- f_{Peak}: 2TFlops

▸ To: Larrabee Revealed

Failed!

Issues

- Different architecture of PUs

Issues

- Different architecture of PUs
- Different performance

Issues

- Different architecture of PUs
- Different performance
- Available programming models

Issues

- Different architecture of PUs
- Different performance
- Available programming models
- Microarchitecture level
- ...

ProcessingUnit-specific

1. Discrete or fused

ProcessingUnit-specific

1. Discrete or fused
2. Computation power of the PUs

ProcessingUnit-specific

1. Discrete or fused
2. Computation power of the PUs
3. Memory bandwidth for data transfer between PUs

ProcessingUnit-specific

1. Discrete or fused
2. Computation power of the PUs
3. Memory bandwidth for data transfer between PUs
4. Strength and weekness of every PU (e.g. GPUs reduced performance in double-precision computations)

Objective-specific

- Energy efficient computing ▸ To: Energy Efficiency
- Performance
- Fairness

Application-specific

- Nature of algorithms
- Subdivision and allocation of work
- Dependencies to previous tasks (where was it executed?)

Programming heterogeneous systems

Desired code with . . .

- Low complexity

Programming heterogeneous systems

Desired code with . . .

- Low complexity
- High portability

Programming heterogeneous systems

Desired code with . . .

- Low complexity
- High portability
- Support of debugging and performance tools

Programming heterogeneous systems

Charm++ is based on C++. The unit of concurrency is a **share object** *with member functions and additional* **entry methods**, *which are asynchronously invoked member functions. This allows the programmer to arbitrary create asynchronous tasks (e.g. broadcast, reductions, near neighbour interactions, data parallelism) [KK11]*

Programming heterogeneous systems

Charm++ provides:

- Accelerated entry methods
  - *may or may not* be executed on accellerator

[5] Single Instruction, Multiple Data
[6] Application Programming Interface

Programming heterogeneous systems

Charm++ provides:

- Accelerated entry methods
  - *may or may not* be executed on accellerator
- Accelerated blocks
  - blocks of code at *global scope*. May called by accellerated entry methods and normal entries.

[5] Single Instruction, Multiple Data
[6] Application Programming Interface

Programming heterogeneous systems

Charm++ provides:

- Accelerated entry methods
  - *may or may not* be executed on accellerator
- Accelerated blocks
  - blocks of code at *global scope*. May called by accellerated entry methods and normal entries.
- SIMD[5] instruction abstraction
  - Map operations from API[6] to hardware's SIMD extensions.

[5] Single Instruction, Multiple Data
[6] Application Programming Interface

Programming heterogeneous systems

Key concept:

Hardware dependent code interpretation of "flexible" written code.

Existing heterogeneous systems

- *XD1* by Cray (ranked # 161 of Top500 in November 2005)
- *Titan* by Cray (ranked # 3 of Top500 and # 73 of Green500 in June 2016)

XD1: let the hardware meet your requirements

The XD1 [Sha06] by Cray

Figure 10: The XD1 supercomputer. (Image credit: Cray 2005)

XD1: let the hardware meet your requirements

- Theoretical peak: 3.633 TFlop/s
- Sold for 100.000$
- FPGA[5] coprocessor-model
- Hardware reconfigurable heterogeneous HPC[6]

[5] Field Programmable Gate Array
[6] high- performance-computing

XD1: let the hardware meet your requirements

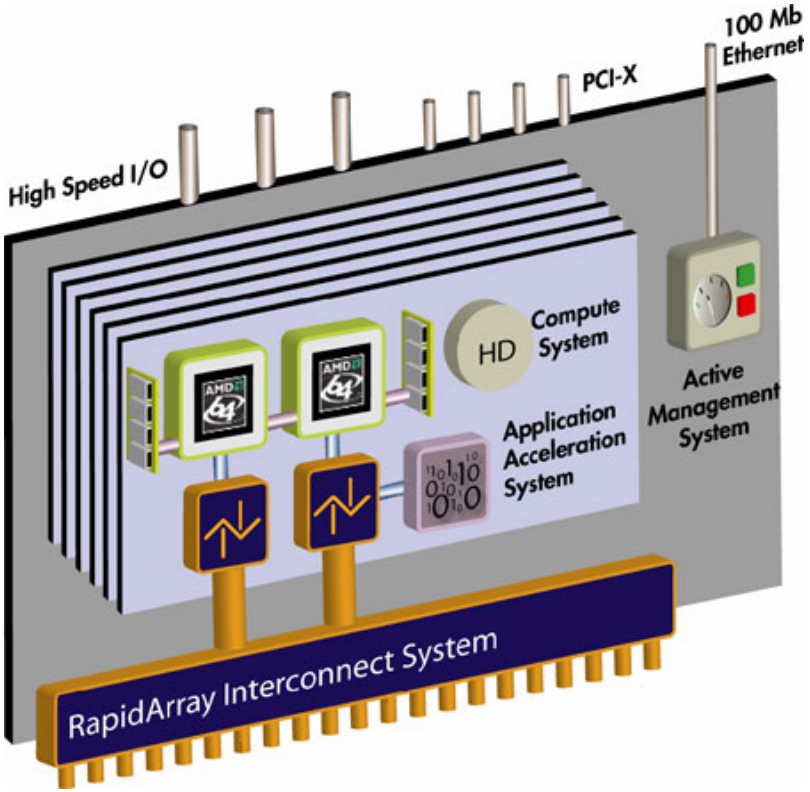


Figure 11: The XD1 supercomputer.
(Image credit: Cray)

- Processors: AMD Opteron 64-bit to deliver 58 GFLOPs per chassis

[a] Application Accelleration System
[b] Field Programmable Gate Arrays

XD1: let the hardware meet your requirements

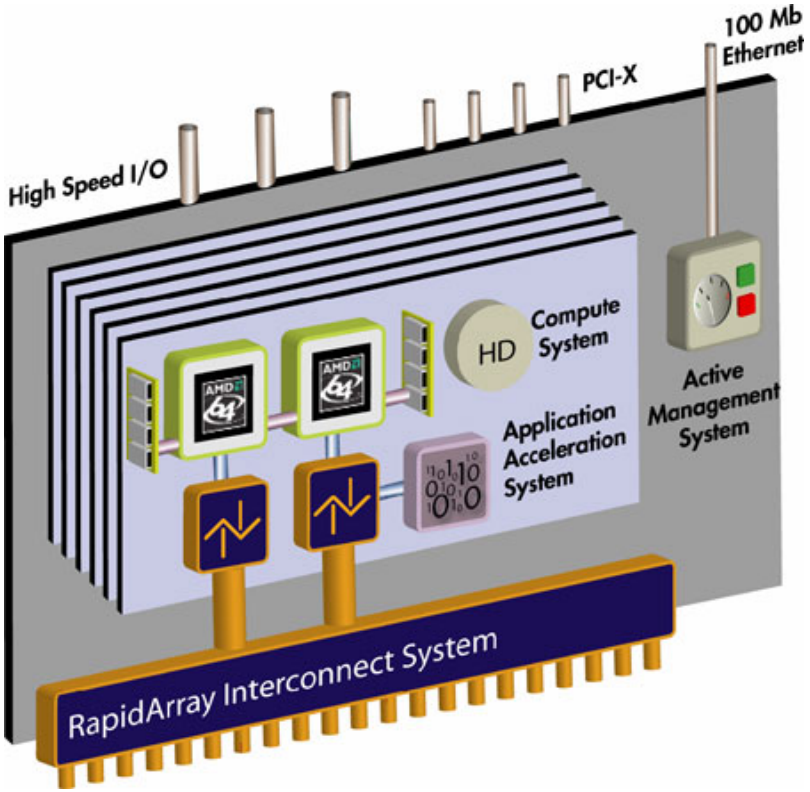


Figure 11: The XD1 supercomputer.
(Image credit: Cray)

- Processors: AMD Opteron 64-bit to deliver 58 GFLOPs per chassis
- System: specialized Linux

[a] Application Accelleration System
[b] Field Programmable Gate Arrays

XD1: let the hardware meet your requirements

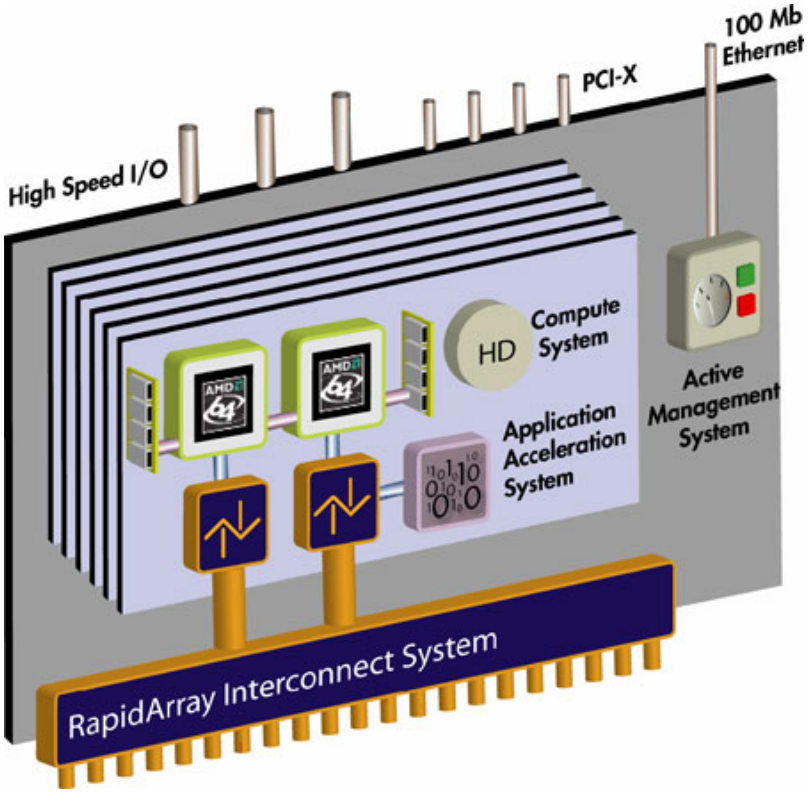


Figure 11: The XD1 supercomputer.
(Image credit: Cray)

- Processors: AMD Opteron 64-bit to deliver 58 GFLOPs per chassis
- System: specialized Linux
- RapidArray^{TM} Interconnect: 96GB/s

[a] Application Accelleration System
[b] Field Programmable Gate Arrays

XD1: let the hardware meet your requirements

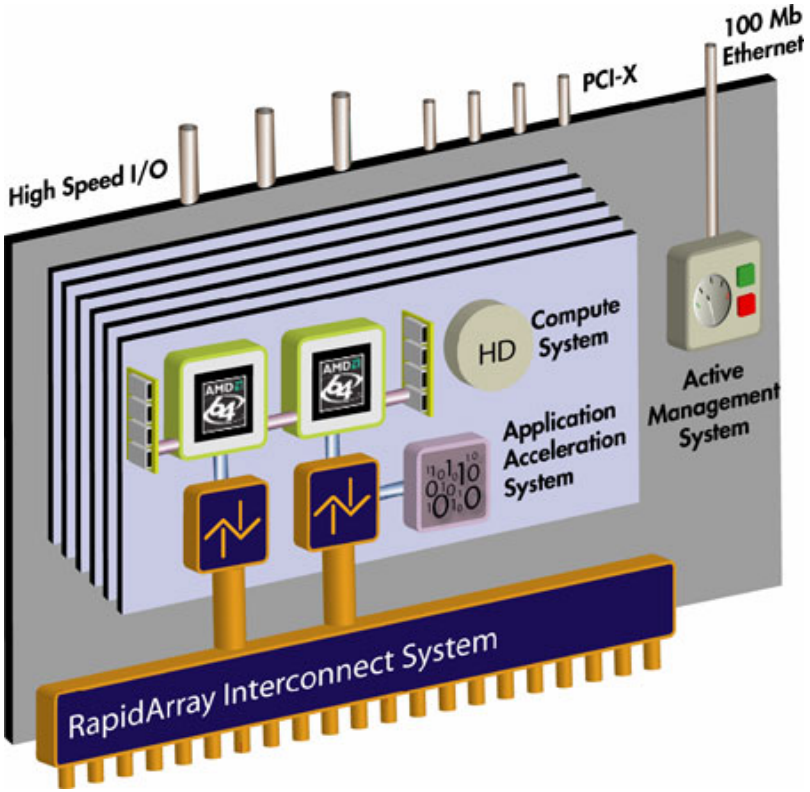


Figure 11: The XD1 supercomputer.
(Image credit: Cray)

- Processors: AMD Opteron 64-bit to deliver 58 GFLOPs per chassis
- System: specialized Linux
- RapidArray^{TM} Interconnect: 96GB/s
- AAS[a]: Six Xilinx Virtex-II Pro^{TM} FPGAs[b]

[a] Application Accelleration System
[b] Field Programmable Gate Arrays

XD1: let the hardware meet your requirements

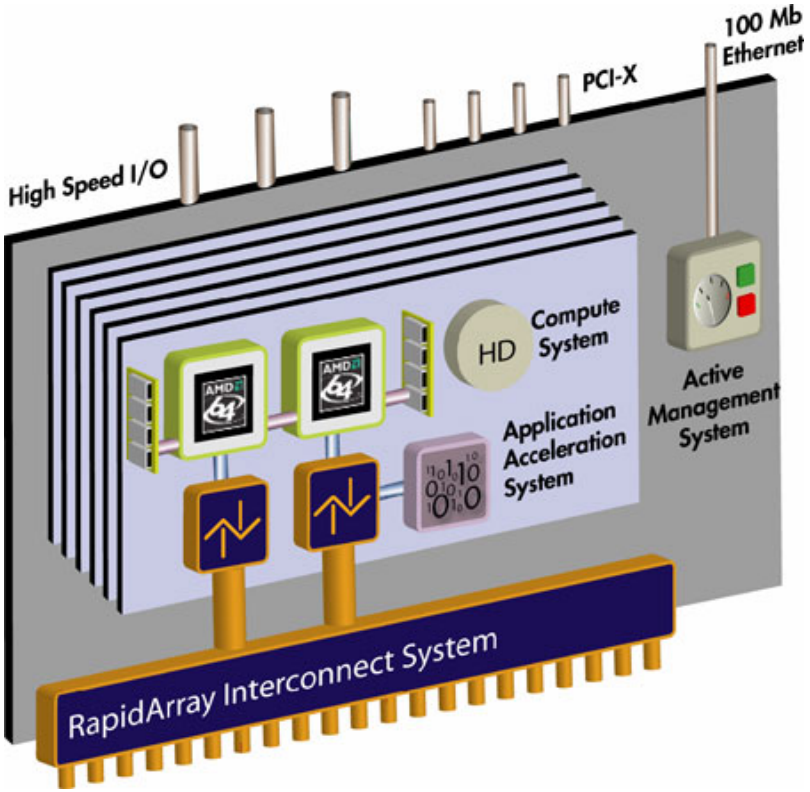


Figure 11: The XD1 supercomputer. (Image credit: Cray)

- Processors: AMD Opteron 64-bit to deliver 58 GFLOPs per chassis
- System: specialized Linux
- RapidArray^{TM} Interconnect: 96GB/s
- AAS[a]: Six Xilinx Virtex-II Pro^{TM} FPGAs[b]
- Active Management: realtime observation of computer's health

[a] Application Accelleration System
[b] Field Programmable Gate Arrays

XD1: let the hardware meet your requirements

The application acceleration subsystem incorporates reconfigurable computing capabilities to deliver superlinear speedup of targeted applications.
[...]
Well suited to functions such as all-reduce operations, searching, sorting and signal processing, the application acceleration subsystem acts as a coprocessor to the AMD Opterons, handling the computationally intensive and highly repetitive algorithms that can be significantly accelerated through parallel execution.

(Credit: Cray XD1 Datasheet)

XD1: let the hardware meet your requirements

The application acceleration subsystem incorporates reconfigurable computing capabilities to deliver superlinear speedup of targeted applications.
[...]
Well suited to functions such as all-reduce operations, searching, sorting and signal processing, the application acceleration subsystem acts as a coprocessor to the AMD Opterons, handling the computationally intensive and highly repetitive algorithms that can be significantly accelerated through parallel execution.

(Credit: Cray XD1 Datasheet)

XD1: let the hardware meet your requirements

The application acceleration subsystem incorporates reconfigurable computing capabilities to deliver superlinear speedup of targeted applications.
[...]
Well suited to functions such as all-reduce operations, searching, sorting and signal processing, the application acceleration subsystem acts as a coprocessor to the AMD Opterons, handling the computationally intensive and highly repetitive algorithms that can be significantly accelerated through parallel execution.

(Credit: Cray XD1 Datasheet)

Titan

The Titan by Cray

Figure 12: The Titan supercomputer. (Image credit: Cray, 2012)

Titan

- First hybrid to perform over 10 petaFLOPS (CPU-GPU)
- Theoretical peak: 27 petaFLOPS
- Power consumption: 8.2 MW
- 2,142.77 MFLOPS/W

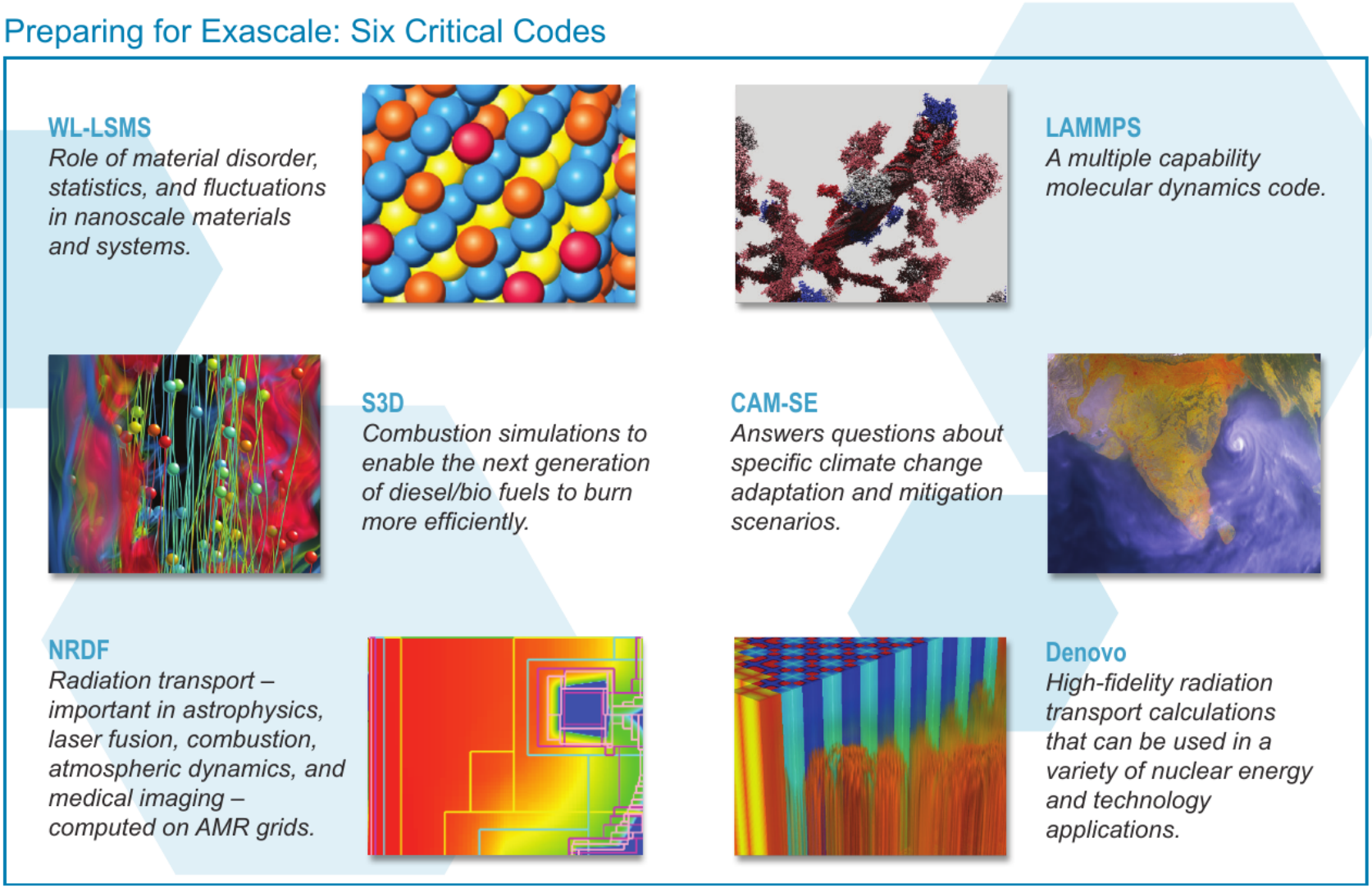


Figure 13: Titan's tasks. (Image credit: Oak Ridge Leadership Computung Facility)

Summary

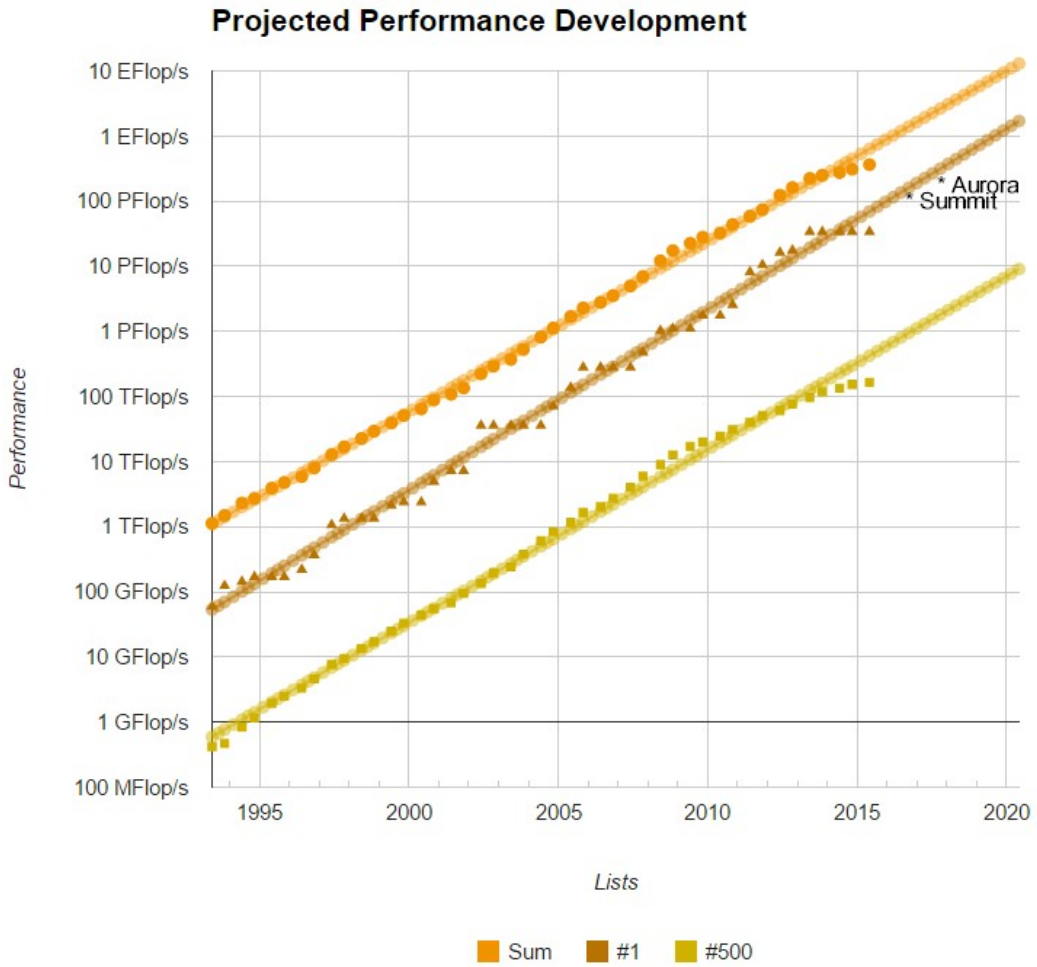
- Homogeneous systems are outreached
- HSA: application specific hardware design
- Heterogeneous systems can be
  - faster
  - more energy efficient
  - hard to program

Summary

- Homogeneous systems are outreached
- HSA: application specific hardware design
- Heterogeneous systems can be
  - faster
  - more energy efficient
  - hard to program

HSA holds the potential to open the era of exascale-supercomputing.

Outlook: still a long way to go...



- China's Tianhe-3 planned for 2020
- Stagnation of Top500 in 2014

Figure 14: The less steep trend of performance since 2014. (Image credit: Nicole Hemsoth)

Thank you!


L. S. Blackford, A. Cleary, J. Demmel, I. Dhillon, J. Dongarragif, S. Hammarlinggif, A. Petitet, H. Ren, K. Stanley, and Whaley R. C. "Practical Experience in the Dangers of Heterogeneous Computing". German. In*LAPACK Working Note 112*, vol. (1996). URL: http://www.netlib.org/utk/papers/practical-hetro/paper.html (cit. on pp. 7–9).


M. Chiesi. "Heterogeneous Multi-core Architectures for High Performance Computing". PhD thesis. University of Bologna, 2014.


J. L. Hennessy and D. A. Patterson. *Computer Architecture - A Quantitative Approach*. Ed. by T. Green. Elsevier Inc., 2012.


T. Herault and Y. Robert. *Fault-Tolerance Techniques for High-Performance Computing*. Ed. by T. Herault and Y. Robert. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-20943-2 (cit. on pp. 28, 29).

D. M. Kunzmann and L. V. Kale. "Programming Heterogeneous Systems". In*IEEE International Parallel and Distributed Processing Symposium*, vol. (2011) (cit. on p. 60).

S. Kaxiras and M. Martonosi. *Computer Architecture techniques for Power-Efficiency*. Ed. by M. D. Hill. Morgan & Claypool, 2008. DOI: 10.2200/S00119ED1V01Y200805CAC004.

D. A. Patterson and J. L. Hennessy. *Computer organization and Design*. Ed. by T. Green. Elsevier Inc., 2014.

Sascha Roloff. *Multicore-Architekturen*. German. University of Erlangen. Sept. 2009. URL: http://www2.in.tum.de/hp/file?fid=311 (cit. on pp. 45, 46, 87, 93).

A. Shan. "Heterogeneous Processing: a Strategy for Augmenting Moore's Law". In*Linux Journal*, vol. (2006) (cit. on p. 66).

J. Vetter and S. Mittal. "A survey of CPU-GPU heterogeneous computing techniques". In*ACM Computing Surveys*, vol. (2015) (cit. on pp. 10–15, 32–36, 94).

H.P. Zima, K. Joe, M. Sato, Y. Seo, and M. Shimasaki. *High Performance Computing: 4th International Symposium, ISHPC 2002, Kansai Science City, Japan, May 15-17, 2002. Proceedings*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003. ISBN: 9783540478478. URL: https://books.google.de/books?id=3QVrCQAAQBAJ (cit. on pp. 28, 29).

The Intel Larrabee

revealed!

Intel Larrabee Project

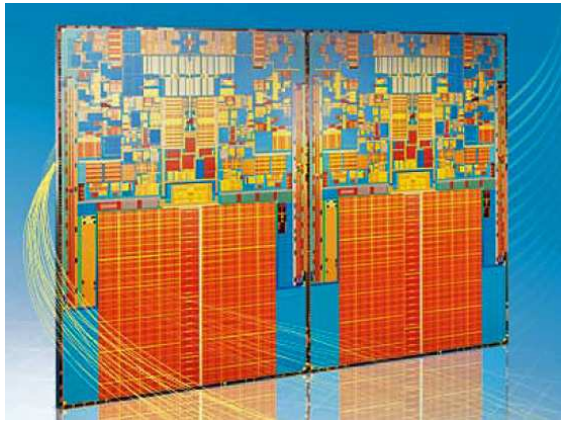
The following extra-slides were extracted from [Rol09].

◂ Back to: "Bad" "Mixtures"

Intel Larrabee

Graphics Processing with CPU

- GPCPU
  - Berechnung von grafikbasierten Anwendungen mit CPUs
- Homogener Multicore-Prozessor
  - 32 Kerne (basieren auf x86 CPUs)
  - flexibel programmierbar
- schnelles Verbindungsnetz

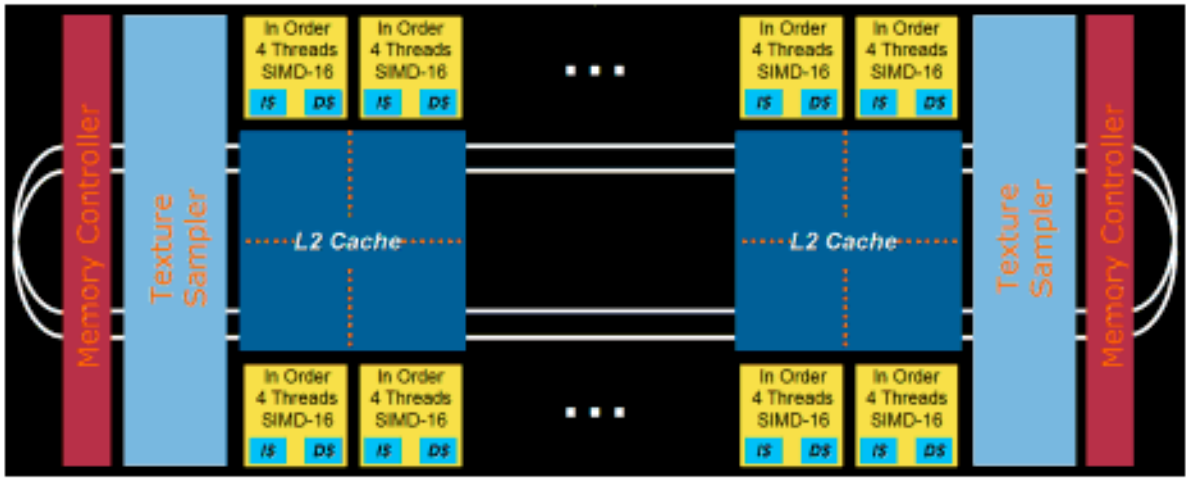


- 2 TFlops Peakperformance

Intel Larrabee

Die Architektur

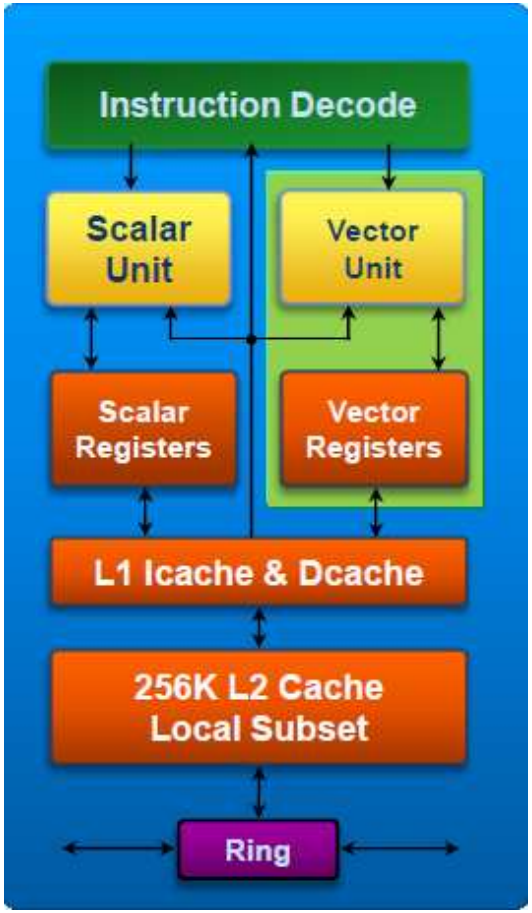
- Gesamtansicht
  - In-Order Kerne mit L1-, L2-Cache (voll kohärent)
  - Verbindungsbus als bi-direktionaler Ring
  - Textureinheiten
  - Speichercontroller



Intel Larrabee

Die Architektur

- Larrabee Core
  - Skalare Einheit (x86 kompatibel)
    - Skalare Register
  - Vektor Einheit (SIMD-Erweiterung)
    - Vektor Register
  - Kommunikation zw. Registern über L1-Cache
  - 256 KB L2-Cache
  - Kommunikation zw. den Cores implizit über L2-Cache Synchronisation!



Intel Larrabee

Die Architektur

- Skalare Einheit
  - zuständig für skalare Operationen
  - 64-bit Unterstützung
  - 4 Hardware-Threads
- Vektor Einheit
  - Berechnung von Ganzzahlen und Fließkommazahlen mit einfacher, doppelter Genauigkeit
  - 512-bit SIMD Befehle
  - 16x 32-bit Operation pro Takt

Intel Larrabee

Parallelisierung

- Larrabee kann Aufgaben einer GPU und einer CPU übernehmen
- Grafik Pipeline beschleunigt Bildverarbeitung
  - Rechenkraft kommt aus den Vektor Einheiten
- Breites Spektrum an parallelem Rechnen
  - General Purpose Programmierung parallel durch viele Cores und viele Threads

The Intel Larrabee

revealed!

Intel Larrabee Project

The following extra-slides were extracted from [Rol09].

◂ Back to: "Bad" "Mixtures"

Green HS techniques

[VM15]

Workload partitioning

At runtime evaluation of workload of CPU, GPU and FPGA, including data transfer time and computation time of all PUs.

DVFS based techniques

Workload division and voltage scaling are modeled for a given performance constraint. (Trade-off model for performance and energy consumption)

Resource scaling techniques

Nodes in large Clusters are shut-down when usually staying idle. Scaling voltage and work-division for "just-in-time" calculations.

◂ Back to: Challenges of HC

The NVLink

by NVIDA

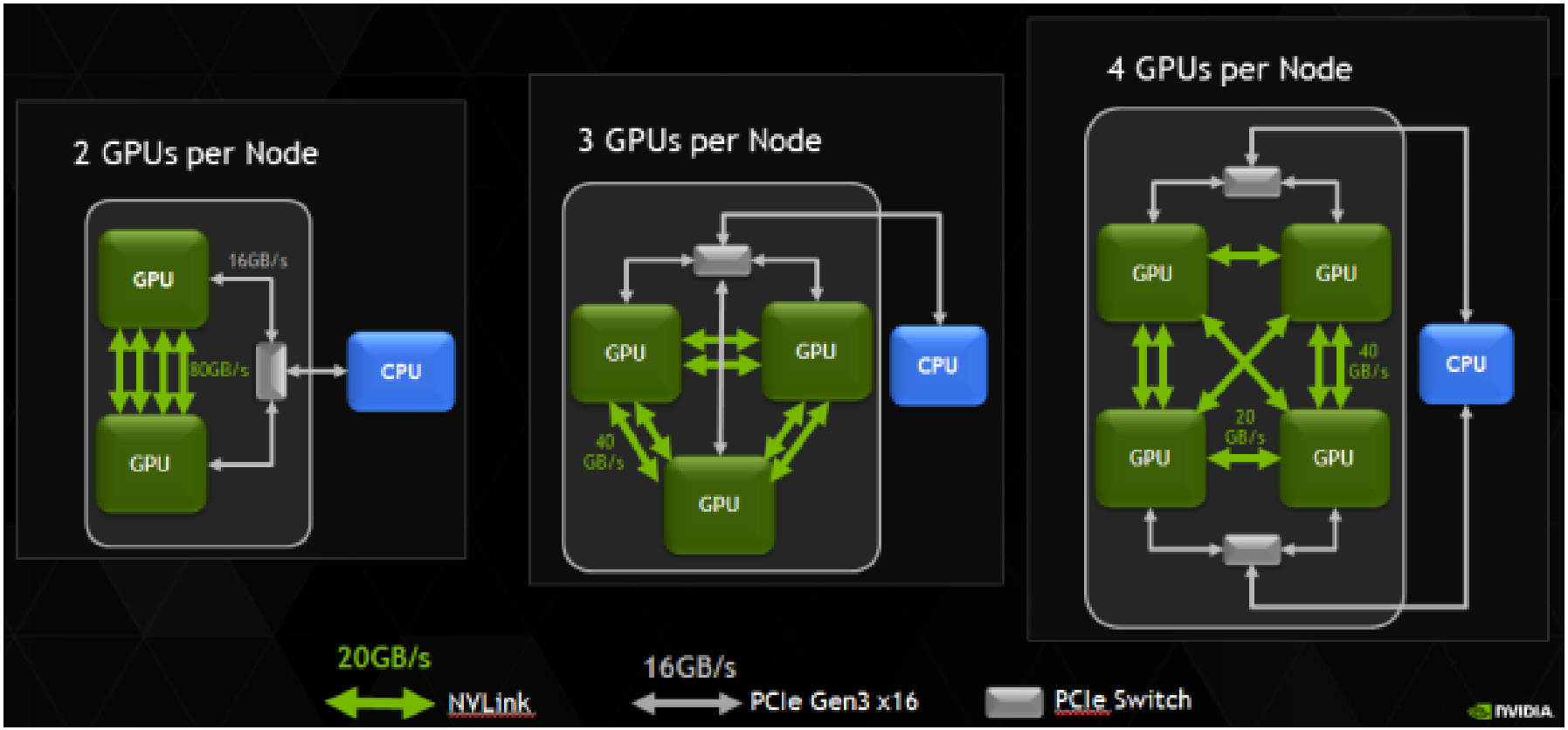


Figure 15: The NVLink to replace the PCIe. (Image credit: NVIDIA)

◂ Back to: Homogeneous Systems