center for excellence in parallel programming

A tale of two era of supercomputing
Now and Tomorrow

Romain Dolbeau

Bull atos technologies

with apologies to Charles Dickens

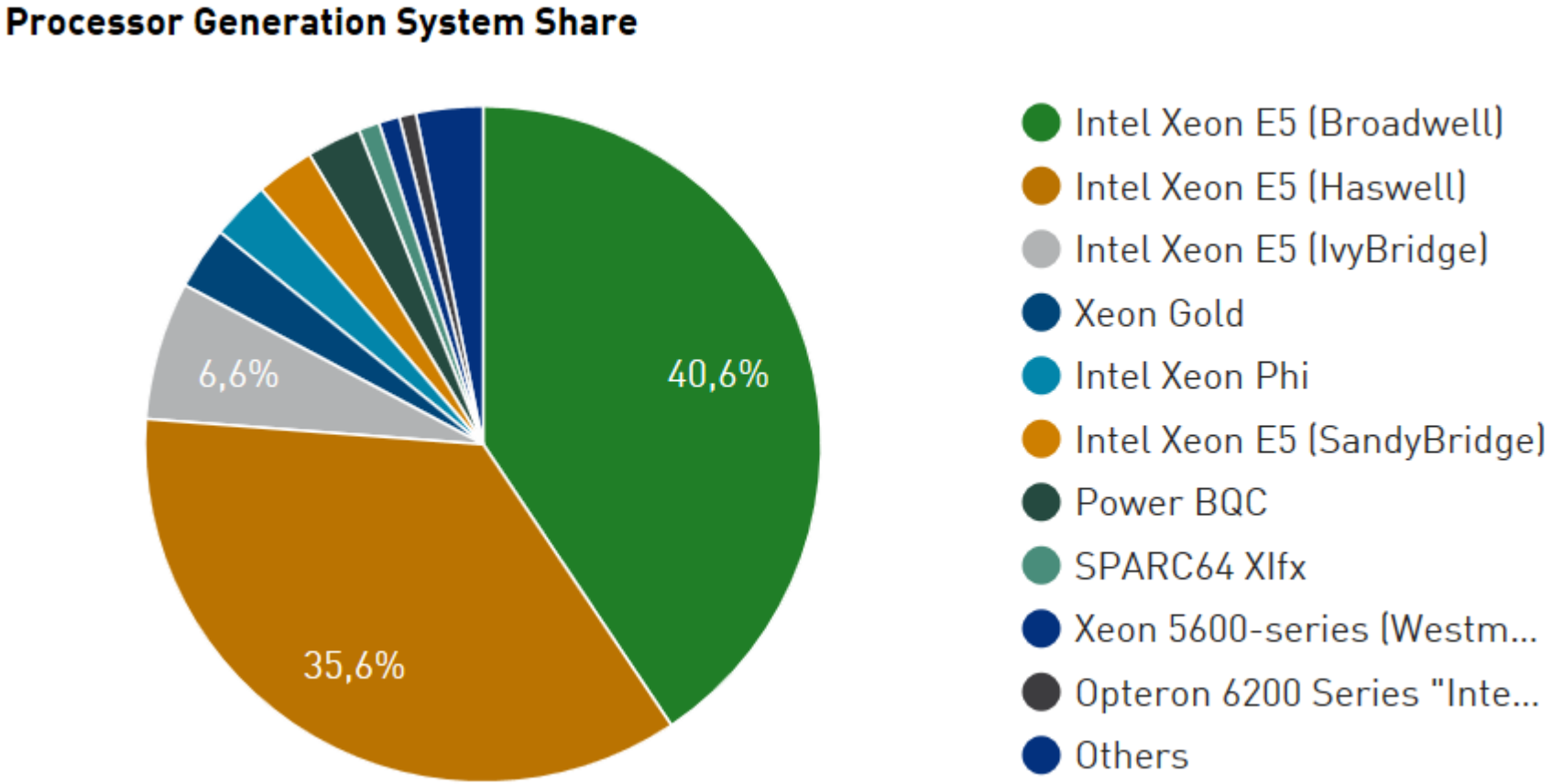
Agenda

1. Now
   - Where we are
2. The recent past
   - How did we get there
3. The end of Moore's Law
   - Can we just go on as before
4. Alternatives
   - Where else could we be
5. The Future
   - Where might we go

Now

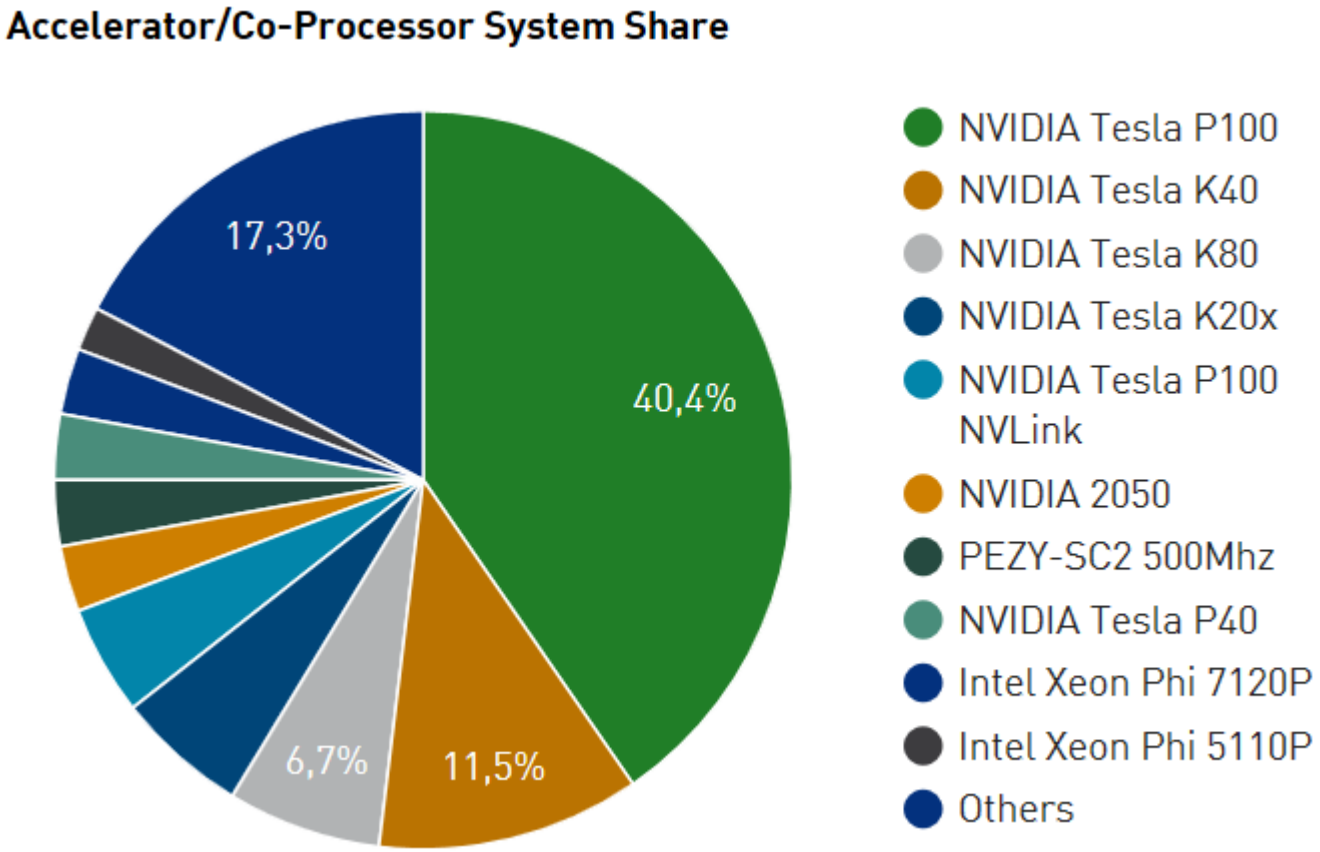
The best of times or the worst of times ?

Top500, November 2017
Processors



Bull
atos technologies

Top500, November 2017
Accelerators (~20% of systems)



Top500, November 2017
Observations

- The top500 seems a very boring list…
- Intel Xeon dominate the CPU landscape
- Nvidia GPU dominate the ~20% of accelerated systems

- It seems the current era of HPC is the triumph of conformity !

- But what if we look at the details ?

- But first a one-question quiz for the audience:
  - **When was the last time the all-powerful Intel got #1 at the top500 with a pure CPU system ?**
    - That is Intel CPUs, no accelerators (not even Phi)
  - *ASCI Red in from 1997/11 to 2000/06 …*

Is the era of the general-purpose CPU over?

- For a couple of decades, general-purpose CPU have ruled the landscape of HPC *in volume*
  - Since the move from vector machines & dedicated massively parallel systems
    - Cray vector systems, NEC SX, …
    - Thinking Machines Connection Machine, …
- The advent of GPGPU computing during the past decade was the first breach of the GPCPU monopoly *in volume*
- The Top500 is usually led by highly specific systems:
  - 2008: Roadrunner #1: GPCPU + Accelerator IBM PowerXCell
  - 2009: Jaguar #1: GPCPU – AMD Opteron
  - 2010: Tianhe-1A #1: GPCPU + GPU – Intel + Nvidia
  - 2011: K computer #1: GPCPU – Fujitsu SPARC
  - 2012/06: Sequoia #1: GPCPU – IBM BlueGene/Q
  - 2012/11: Titan #1: GPCPU + GPU – Intel + Nvidia
  - 2013: Tianhe-2: CPU + Accelerator – Intel + Intel Phi
  - since 2016: TaihuLight : Specific CPU - Sunway

Is the era of the general-purpose CPU over?

- As of November 2017, about 1-in-5 systems of the Top500 are accelerated
  - NVidia Tesla of various generations
  - Intel Xeon Phi
  - PEZY-SCnp
  - AMD FirePro of various generations
- GPCPU still endures in the volume business by their comparative ease-of-use
  - Programmability is key
- But even major GPCPU vendors are moving toward more complex programming using large vectors
  - Intel AVX-512
  - ARM Scalable Vector Extensions
- Small cores or big cores ?
- Specific CPU and accelerators are leaders in Performance/Watts
  - Deep Learning is pushing toward highly specific silicon

The recent past
we had everything before us, we had nothing before us

Intel x86-64: From "NetBurst" to "Skylake", timeline (1: before Nehalem)

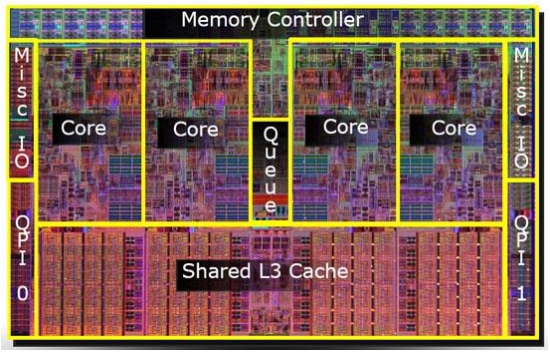
- The Intel "Nocona" Xeon was introduced in June 2004, and was the first Intel server CPU featuring EM64T, a.k.a. AMD64, a.k.a. x86-64
  - Based on the "NetBurst" microarchitecture introduced with the Pentium 4
  - 64 bits registers, 64 bits pointer, and vector extension up to SSE3
  - 90 nm, single core, from 2.8 GHz to 3.6 GHz
  - Replaced in May 2006 by the dual-core, 65 nm "Dempsey" (Xeon 50xx), which introduced VT-x (virtualization support)

- The Intel "Woodcrest" Xeon (51xx) was introduced in June 2006
  - Based on the "Core" microarchitecture
  - Added SSSE3
  - 65 nm, dual core, from 1.6 GHz to 3 GHz (and quad-core "Clovertown", 53xx)

- The Intel "Wolfdale-DP" Xeon (52xx) was introduced in November 2007
  - 45 nm shrink of "Core" (a.k.a. Penryn)
  - Added SSE4.1
  - Dual-core, 1.86 to 3.4 GHz (and quad-core "Harpertown", 54xx)

Intel x86-64: From "NetBurst" to "Skylake", timeline (2: before Haswell)

- The Intel "Gainestown" Xeon (55xx) was introduced in January 2009
  - Based on the "Nehalem" microarchitecture
  - Memory controller integrated in the CPU instead of the chipset
    - No more shared "Front Side Bus"
    - "NUMA" design
    - Pioneered by AMD with the Opteron family, like AMD64
  - QPI for inter-socket communications
  - Three-level cache hierarchy with shared LLC (a.k.a. shared L3)
  - Added SSE4.2, I/O virtualization (VT-d, VT-c), …
  - 45 nm, quad-core, up to 3,6 GHz
  - 32 nm shrink as "Westmere" (56xx), adding AES, with up to 6 cores



- The Intel "Sandy Bridge-EP" (E5-26xx) was introduced in March 2012
  - Based one the "Sandy Bridge" microarchitecture
  - Added AVX (doubling the vector register size)
  - 32 nm, up to 8 cores, up to 3 GHz + Turbo
  - 22 nm shrink as "Ivy Bridge" (E5-26xx v2) with up to 12 cores

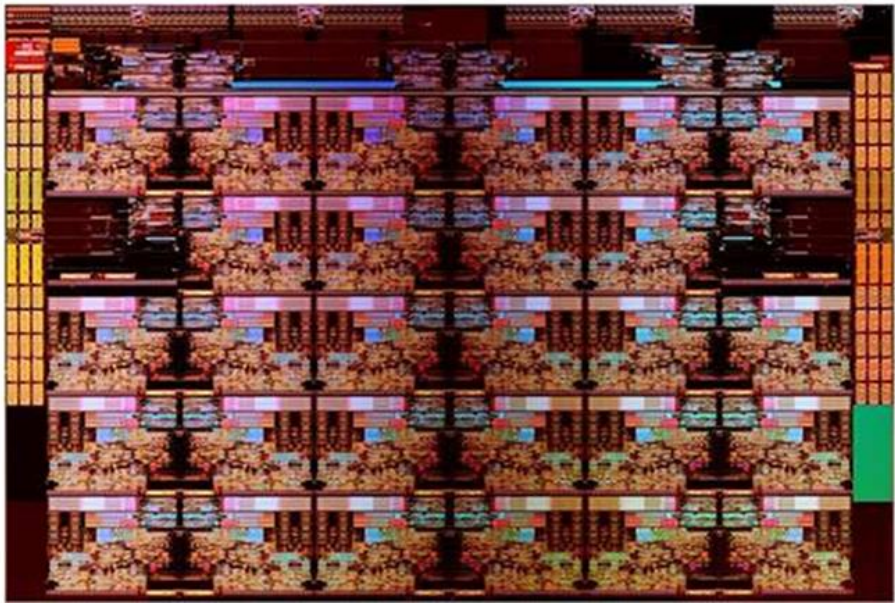
Intel x86-64: From "NetBurst" to "Skylake", timeline (3: before Skylake)

- The Intel "Haswell-EP" Xeon (E5-26xx v3) was introduced in September 2014
  - Based on the "Haswell" microarchitecture
  - Added AVX2 and FMA (Fused Multiply-Add) and BMI
  - Introduced the "AVX clock rate" (lower than nominal)
  - 22 nm, up to 18 cores, up to 3.7 GHz
  - 14 nm shrink as "Broadwell" (E5-26xx v4), with up to 22 cores and faster FMA, ADX
- "Broadwell" is the current "normal" Xeon range
  - Until "Skylake" is officially introduced

- The Intel "Knights Landing" Xeon (Phi 72xx) was introduced in June 2016
  - Based on the "many core" (MIC) "Knights Landing" microarchitecture
  - Derived from the "Atom" range of cores, very different ("small") cores
  - Supports all instruction sets from "Broadwell" except transactional memory
  - Added AVX512F, AVX512CD, AVX512ER, AVX512PF (double vector register width, again)
  - 14 nm, up to 72 cores, up to 1.4 GHz

Intel x86-64: From "NetBurst" to "Skylake", timeline (4: current)

- The Intel "Skylake" Xeon (Xeon Scalable) was introduced in July 2017
  - Quite different from the already available "consumer" "Skylake"
  - Server "Skylake" …
    - Supports AVX512F, AVX512CD
      - But not AVX512ER, AVX512PF
    - Introduces AVX512BW, AVX512DQ, AVX512VL
    - Has a different cache hierarchy
      - Non-inclusive, smaller L3
      - Larger L2
      - First major change since "Nehalem"
    - Uses UPI for inter-socket communications
      - Replace QPI
      - First major change since "Nehalem"



Intel x86-64: From "NetBurst" to "Skylake"

- A long lineage of "big core"
  - Single-thread performance is excellent
- Ever-growing number of cores
  - From 4 in Nehalem to up to 28 in Skylake
- Ever-growing memory bandwidth
  - From 32 GB/s per socket in Nehalem to 128 GB/s in Skylake
    - Twice the channels at twice the frequency
- Ever growing performance !
- So… what's the worry ?

The End of Moore's Law

Spring of hope or winter of despair ?

La fin de la Loi de Moore et les applications HPC

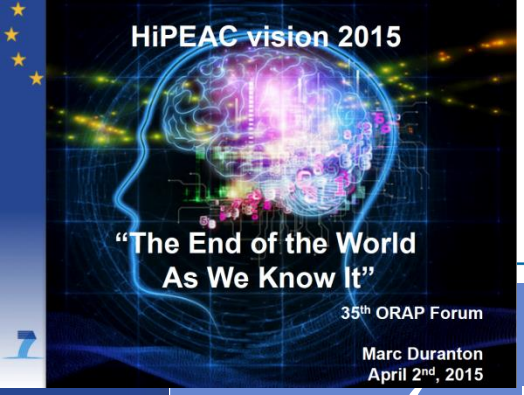
François Bodin,
Université de Rennes 1 / Irisa

Forum Orap, 2 avril 2015

The end of the Moore's law and the HPC applications

The « laws »

- The laws that struggle
  - The Moore's law: transistors density
  - The Dennard's law (1974): constant energy density
  - The Kryder's law: observes that storage density of magnetic disks increased faster than chip density
- The laws that carry on
  - The Rock's law: the price of a semiconductor chip plant doubles every four years
  - The Amdahl's law about speedup
  - The Gustafson's law about « weak scaling »



The end of Dennard Scaling

Parameter (scale factor = a)	Classic Scaling
Dimensions	1/a
Voltage	1/a
Current	1/a
Capacitance	1/a
Power/Circuit	$1/a^2$
Power Density	I
Delay/Circuit	1/a

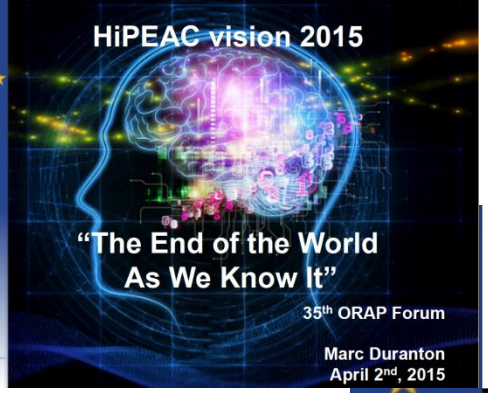
Everything was easy:

- Wait for the next technology node
- Increase frequency
- Decrease Vdd

⇒ Similar increase of sequential performance

⇒ No need to recompile (except if architectural improvements)

Source: Krisztián Flautner "From niche to mainstream: can critical systems make the transition?"

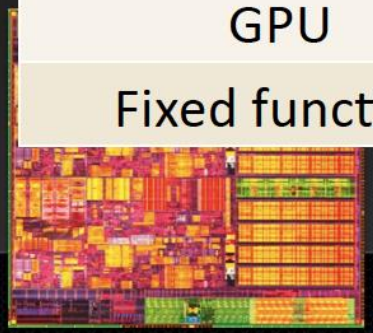


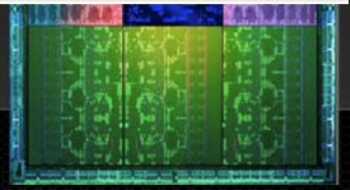
Specialization leads to more efficiency

CPU
1690 pJ/flop

GPU
140 pJ/flop

Type of device	Energy / Operation
CPU	1690 pJ
GPU	140 pJ
Fixed function	10 pJ


Westmere
32 nm


Kepler
28 nm

Source from Bill Dally (nVidia) « Challenges for Future Computing Systems »
HiPEAC conference 2015

What is peak performance ?

- Basic equation

$$\frac{flop/s}{node} = \frac{flop}{second} \times \frac{cores}{node}$$

- In details

Fused Multiply-Add = 2

Frequency ; 1-5 GHz

Vector ; CPU = 2-8, GPU >= 512

Multi/Many Cores ; few - dozens

$$\frac{flop/s}{node} = \left\{ \begin{array}{l} \frac{flop}{operation} \\ \times \frac{operations}{instruction} \\ \times \frac{instructions}{cycle} \end{array} \right\} \left(\begin{array}{l} micro \\ architecture \end{array} \right) \left\{ \begin{array}{l} \times \frac{cycles}{second} \\ \times \frac{cores}{socket} \\ \times \frac{sockets}{node} \end{array} \right\} \left(\begin{array}{l} machine \\ architecture \end{array} \right)$$

Superscalar Pipelines ; 1-2

Clustering ; several thousands

The end of Moore's Law (1)

- It's getting harder and more expensive to cram billions of transistors in a chip
- Moving data around is a bigger problem than ever
  - There's more of it, and we compute much faster
- Every transistor spent on making the CPU faster at single-thread is not available for computation
  - Out-of-order execution
  - Branch prediction
  - Wide-issue
  - …

The end of Moore's Law (2)

- If the HPC community wants more FLOPs …

1. New and future hardware will be harder to leverage
   - Less transistors available to help the programmer
   - More transistors for pure compute

2. Need to tune more and more the low-level parts to the underlying hardware
   - Otherwise, little gain from newer hardware

- True for GPCPU, GPGPU, Accelerators, dedicated hardware, …

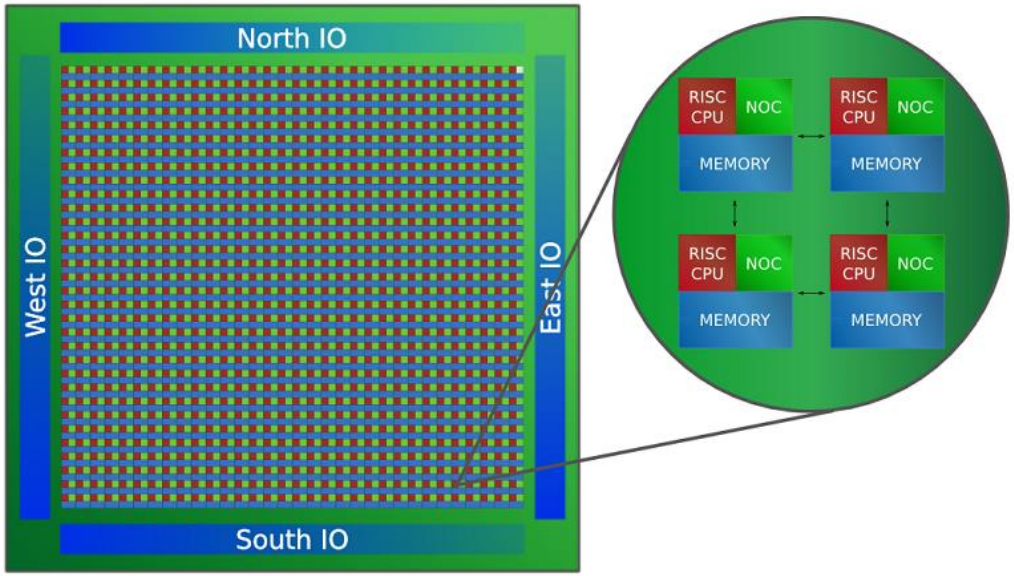
The alternatives
epoch of belief or epoch of incredulity ?

Many players in the market

- From CPU to various accelerators, many players are trying to redefine the HPC landscape
- On the conventional CPU side of things
  - AMD makes an Epyc comeback
    - (too easy, I know)
  - ARM is pushing hard in the HPC market
  - ARM licensees are starting to take position
    - Well – Cavium is…
  - The Europeans might finally show up at the party
    - "European Processor Initiative", EPI, still in infancy
- On the more "exotic" side of things
  - Big Chinese player like Sunway
  - Many smaller player with high-efficiency dedicated hardware

Epiphany-V: 75 DP Gflops/watt !

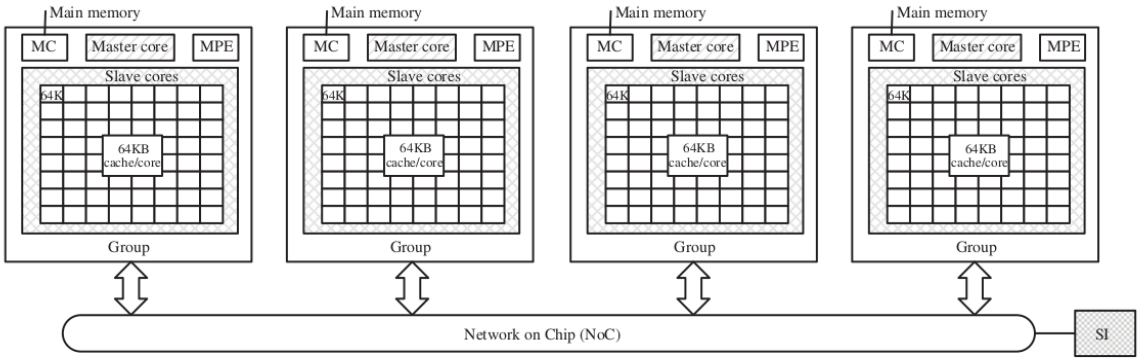
- Taped-out (09/16) Adapteva Epiphany V announce 75 DP Gflops/watt ☺
  - Actual silicon still 4-6 months later ?
  - DARPA research project …
- 1024 cores ! ☺
- Can be connected as a chip-level grid to get millions of cores…
- But programming model identical to the previous Epiphany III
  - No MMU or any advanced feature – no OS, accelerator only
  - Local memory is 64 KiB per core
    - 32 KiB for EpiIII, shared betv
    - But very fast – size and perfo
  - Access to main memory of questionable performance in the reference board
- Exascale efficiency…
- Also Exascale complexity?



At-scale example: Sunway TaihuLight

- The Sunway TaihuLight is leader of the Top500 since June 2016
- Many cores
  - 260 cores per node – 4 times 64+1
  - Main core of each group of 65 is a "real" core, with OS support & caches
  - Other 64 are "compute" core, with 64 KiB of scratchpad memory
    - No caches
    - Similar to Epiphany, or the per-multiprocessor "shared" memory in GPU
  - Only 1.45 GHz, only 8 flop/cycle
- Not a lot of memory, only 32 GiB / node

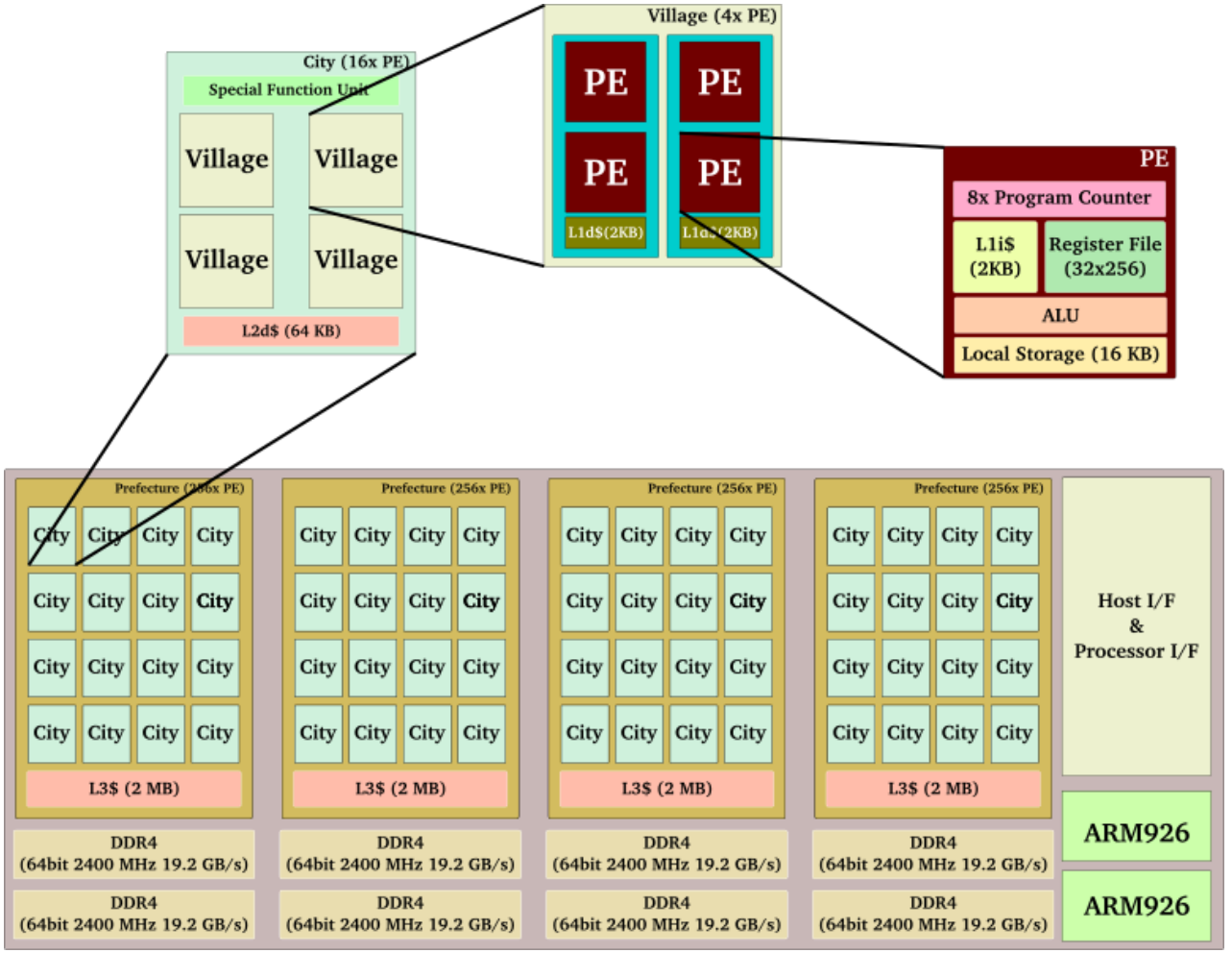
- Programming models include OpenCL & OpenACC
  - The 64 cores are really accelerators, not GPCPU



- As of 2018, a desktop version is announced in China

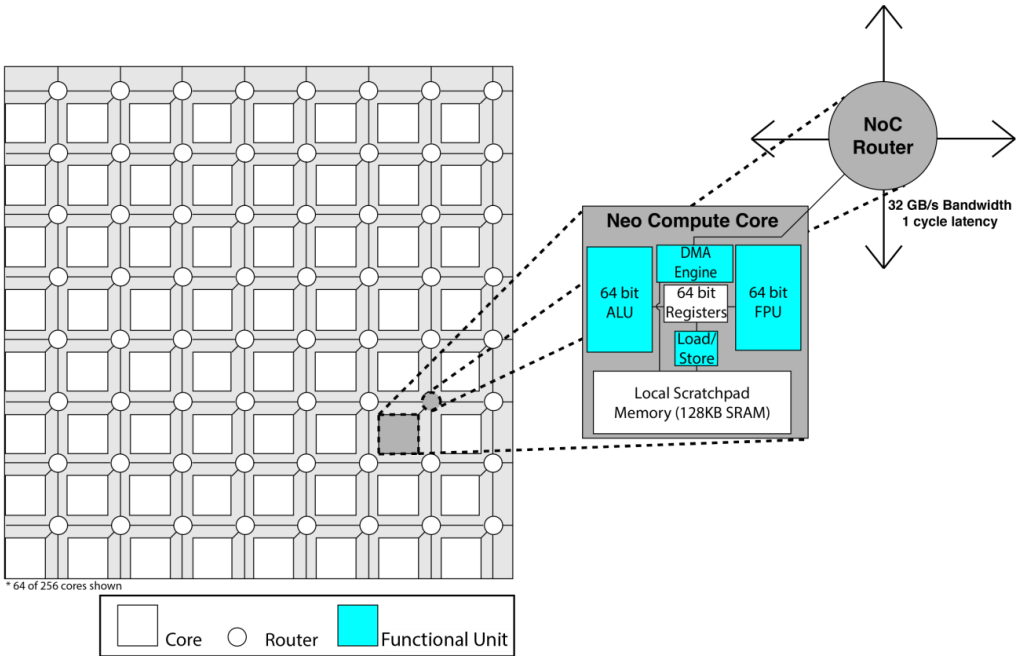
Pure HPC acclerator: PEZY-SC

- 2 ARM926 controller cores
- 1024 compute RISC cores
- PCIe Gen2 (SC) or Gen3 (SCnp)



REX Neo: another future player

- "256 GFLOPs (Double Precision) or 512 GFLOPs (Single Precision) at 64 to 128 GFLOPs/watt" (from rexcomputing.com)
- Pretty much the same global schematic as the Epiphany…
  - Or previously the Tilera chips
  - Also similar to the KalRay design
  - Or some aspect of Skylake
  - …
- Also a DARPA project …



- … posit in the second version?
  http://web.stanford.edu/class/ee380/Abstracts/170201-slides.pdf

GPCPU, accelerators or specific CPU ?

- To reach Exascale, Flop/Watt will be key
- GPCPU is easy to use, but grossly inefficient in terms of power usage
- HPCCPU such as Intel Knights Landing are more power-efficient
  - Less than even more specific hardware
  - But easier to use than more specific hardware
- GPGPU is even more power-efficient
  - But still less than really dedicated hardware
  - And still easier than really dedicated hardware…

Credit: Andreas Olofsson, "Epiphany-V: A 1024 processor 64-bit RISC System-On-Chip"

- TaihuLight is #3 at the Green500, about 10% less efficient than #1
- #1 & #2 in Green500 are using PEZY-SCnp accelerators

Chip	GFLOPS/mm^2	GFLOPS/W	W/mm^2
P100	7.7	18.8	0.40
KNL	5.27	14.69	0.35
Broadwell	2.85	9.08	0.31
Epiphany-V	8.55	TBD	TBD

optimistic

Table 12: Normalized Double Precision Floating Point Peak Performance Numbers. An arbitrary 500MHz operating frequency is used for Epiphany-V.

Chips & technology

Credit: Andreas Olofsson, "Epiphany-V: A 1024 processor 64-bit RISC System-On-Chip"

Chip	Company	Nodes	FLOPS	Area	Transistors	Power	Process	Ref
P100	Nvidia	56	4.7T	610	15.3B	250W	16FF+	[23]
KNL	Intel	72	3.6T	683	7.1B	245W	14nm	[24]
Broadwell	Intel	24	1.3T	456	7.2B	145W	14nm	[25]
Kilocore	UC-Davis	1000	N/A	64	0.6B	39W	32nm	[26]
Epiphany-V	Adapteva	1024	2048 * F	117	4.5B	TBD	16FF+	

optimistic

Table 11: Processor Comparisons. Nodes are programmable elements that can execute independent programs, FLOPS are 64-bit floating point operations, Area is expressed in mm^2. Epiphany performance is expressed in terms of Frequency ("F").

Knights Landing / Xeon Phi 72xx, a HPCCPU (1)

+ Wider vector with AVX-512
  - Up to 8 DP or 16 SP elements in a vector
+ More cores
  - Up to 72 cores per socket (64 or 68 initially)
+ MCDRAM
  - Up to 16 GB of high-bandwidth memory on-socket
- Lower frequency compared to Haswell / Broadwell
- Only one socket per system
- Simpler/smaller core: narrower issue, less out-of-order capabilities, …

▶ Broadwell node
  - Dual E5 2680v4 nom. freq. : 2 * 4 * 2 * 2.4 * 14 * 2 = 1075,2 DP GFlop/s
  - Dual E5 2680v4 AVX freq. : 2 * 4 * 2 * 2.1 * 14 * 2 = 940.8 DP GFlop/s
▶ Knights Landing node
  - 7250 nom. freq. : 2 * 8 * 2 * 1.4 * 68 * 1 = 3046.4 GFlop/s
  - 7250 AVX freq. : 2 * 8 * 2 * 1.2 * 68 * 1 = 2611.2 GFlop/s

Knights Landing / Xeon Phi 72xx, a HPCCPU (2)

- – Is more dependent on vectorization than Haswell/Broadwell
- \+ The penalties of not vectorizing are lower than for GPU

- – Single-thread performance is lower than Haswell/Broadwell
- \+ Single-thread performance is higher than SW26010 (TaihuLight) let alone Epiphany 5

- \+ Tools are shared with previous Intel CPUs

- \+ Previous libraries can be directly re-used, even if not yet optimized
  - Exotic hardware needs all required libraries ported before becoming usable

- \+ Many optimizations for Knights Landing are also useful on regular Xeons

- Most of the analysis, and some of the porting effort, of codes will be re-usable in future architecture

Volta V100, GPGPU for HPC

- The Volta-based Tesla V100 is very HPC-oriented
- V100 is a massively parallel, cache-coherent, GPU-like HPC device
  - No longer simply leveraging consumer offering
  - The previous Pascal P100 was already a mostly HPC design, different from P40 and the GeForce devices
- V100 is even more versatile than the P100 was
  - Less tightly coupled CUDA threads
  - Better caching
  - Overall, gives better performance than P100 with less effort
    - Easier to get good performance from pure OpenACC code, for instance

- NVlink interconnect allows for much better cooperation between V100 devices
  - But requires the new SXM2 form factor and specially designed host systems
  - PCI express card are lower clocked and can't efficiently use Nvlink
  - NVlink also allows for a shared memory space between CPU and GPU with IBM POWER9

ARM, a choice for “small cores” ? (1)

- Going back to the “peak” formula, the idea of “small cores” is simply to
  1. Use simpler/smaller cores even if they have lower peak
  2. Use more of them to create the peak performance

- ARM is a runaway success in the mobile world, powering most of phones (dumb, smart and feature alike) and tablets and even “phablets”
  - And set-top boxes and other no-so-mobile devices
  - And very small cores for IoT
  - Also in the small computer market with Android-based laptops
- But NEON is very limited in scope (only 128 bits registers, no easy masking, similar to SSE in many respect) and implementations (A57 can only do 4 DP Flops/cycle)
- Is Scalable Vector Extension the answer?
  - Imagine a not-so-big core with 512 bits, maskable vector, and use plenty of them…
  - ... does that remind you of anything?

ARM, a choice for “small cores” ? (2)

- Knights Landing use such an approach
  - Smaller cores based on the Atom range
  - Retain very wide vectors to keep up a high theoretical peak per core
- Theoretical peak is *very* high
  - 2.5-3 Tflops / socket

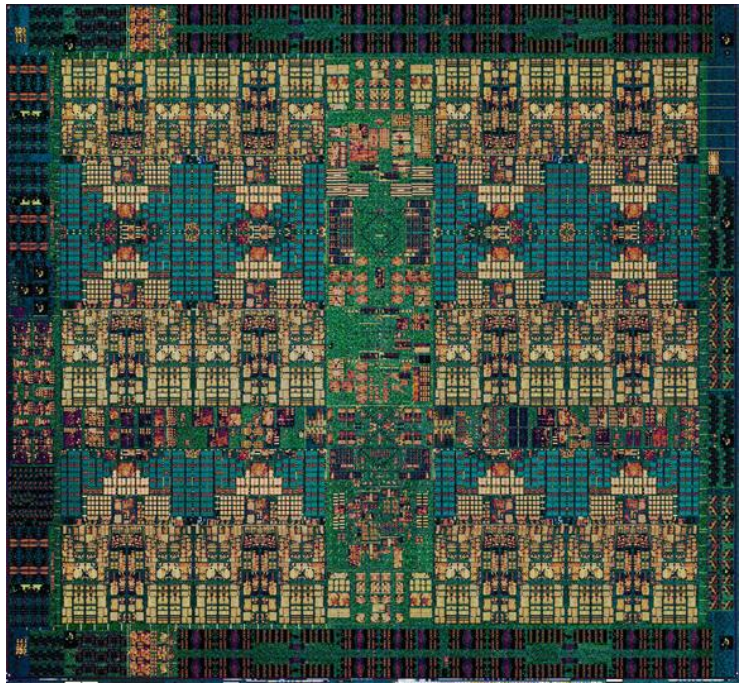
- SVE-based ARM systems likely to need similar effort to use efficiently
  1. Proper exploitation of vector capability
  2. Minimizing sequential parts that are even costlier on smaller cores

- KNL frequency, like Haswell and Broadwell, gets lower when intensely using the FMA capabilities…
  - So does Skylake…
  - Power usage + thermal dissipation are limiting factor

- Will ARM retain its power advantage after adding power-hungry operators?

POWER8 & 9: big cores !

- POWER8 is up to 12 cores at up to 5 GHz
- Wide issues, up to 8 instructions per cycle
  - And up to 8 threads per core to help saturate the pipelines
- "Only" 8 Flops per cycle
  - 4 FMA per cycle
  - Support SIMD but isn't entirely reliant on it for performance
  - Very high per-thread performance
- Quite power-hungry: the 10 cores, 2.92 GHz nominal (3.5 GHz Turbo) POWER8 has a TDP of 190W
  - … for 233.6 GFlops nominal
- POWER9 pushes to 12 SMT8 cores or 24 SMT4 cores at 4 GHz
  - So both 96 execution threads



AMD “Naples” / Epyc: semi-big ? (1)

- The server processor from AMD codename “Naples”, sold as “Epyc”
- Based on the “Zen” microarchitecture
  - Released March 2017 with the “Ryzen”-branded consumer CPU
- Each Epyc is build from 4 dies (chips) in a multi-chip module
  - Believed to be more cost-effective than a large monolithic die by AMD
- Only has half the Flops/cycle of “Haswell”/”Broadwell”
  - Only 128 bits-wide pipeline,
    4 of them (2 ADD, 2 MUL)
  - So ¼ of KNL or SKX…
- Clearly not “big” from a FLOPs point-of-view !
- But …

AMD “Naples” / Epyc: semi-big ? (2)

- It has more cores per socket than all the conventional Xeon
  - Up to 32 cores per socket, at 2.2 GHz nominal for the Epyc 7601
- More bandwidth per socket than HSW/BDW/SKX/KNL
  - 8 channels per socket, vs. 4/4/6/6
- More private L2 cache per core than HSW/BDW
  - 512 KiB vs. 256 KiB
  - Less than SKX which has 1 MiB/core
- Slightly less shared L3 cache per core than HSW/BDW
  - 2 MiB vs. 2.5 MiB
  - More than SKC with has 1.375 MiB/core
- Excellent single-thread performance for non-AVX workload

FPGA: algorithms in hardware (1)

- "Field-Programmable Gate Array"
- A bunch of silicon gates that can be reconfigured in the field
  - As an array of basic "logic block", typically with LUT (look-up table), adder, flip-flop, … and some embedded memory cells
- Allows almost any algorithms/functions to be implemented "in silico"
- Can offer very low-latency and very high performance for specialized use
- Widely used for hardware simulation, for signal processing, for ultra low latency applications (High Frequency Trading, …), …
- Commonly independent chips, now also available with "hard" core
  - i.e. Xilinx Zynq 7xxx (Cortex A9 + FPGA) or Zynq UltraScale (A53 + FPGA)
  - Intel has bought FPGA manufacturer Altera, and has announced Skylake + FPGA

FPGA: algorithms in hardware (2)

- But some drawbacks…
  - Originally required programming in Hardware Description Language (HDL) such as VHDL or Verilog, very different from C let alone Fortran or C++
  - Suppliers now offers higher level tools (for C, OpenCL, …) but at a potential performance cost
    - Overhead from the abstraction layers
  - Reconfiguring the device takes time, so to implement more than one algorithm…
    - Static partitioning: algorithms are all in the silicon at the same time, but each has only a fraction of the resources available
    - Dynamic reconfiguration: each algorithm can use the full device, but there is a time penalty switching from one algorithm to another
  - FP operators can be very expensive if they are not "hardwired" (in "hard blocks") in the device
  - Likely need interconnection with a host CPU (PCIexpress, or via some form of memory sharing)
  - Expensive hardware

The Future

we had everything before us,
we had nothing before us

What's next

- We already have an established ecosystem around mostly Intel and NVidia
  - Part 1 & 2 of this talk
- We know we can't just rely on "more of the same" to get us to Exascale and beyond, with Moore's law in jeopardy
  - Part 3
- Many players are positioning themselves to offer an alternative to the current status quo
  - Part 4

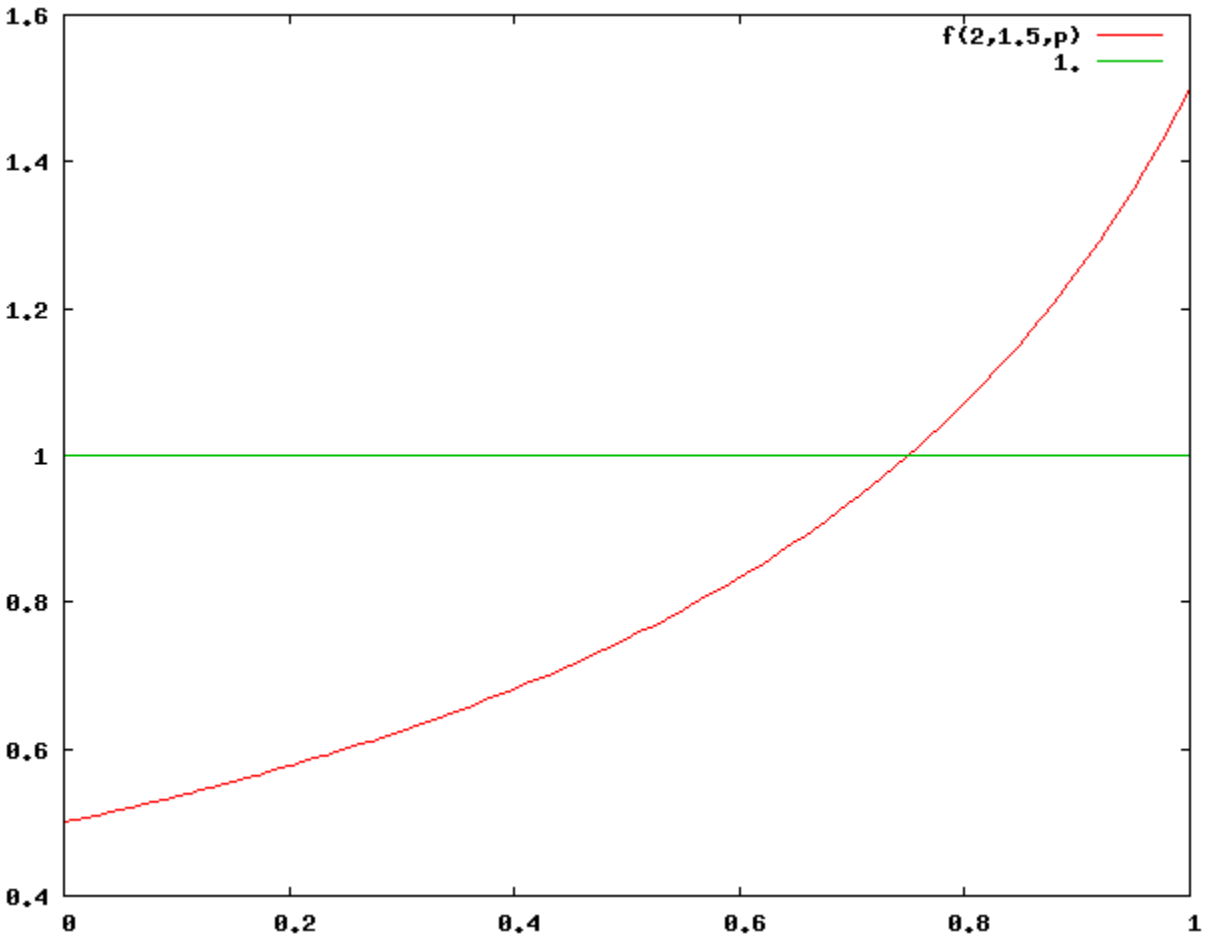
- Big cores? Small Cores? Accelerators? Or something… revolutionary?

Big Cores vs. Small Cores. vs. Amdahl's Law (1)

- Let's revisit Amdahl' Law
  - The original define the speed-up as $\frac{1}{(1-p)+\left(\frac{p}{n}\right)}$
  - Where p is the fraction of the time in parallel mode and n the gain from the parallel mode (e.g., the number of cores)
- Assume a baseline system with a reference performance of 1 for all p
- Intuitively, if we use slower cores but more of them…
  - We will slow down both parts $1-p$ and p/n as the cores are slower
  - But speed-up the parallel part since n is higher – so we should be slower for p close to 0, and faster for p close to 1
- Let's plot this for $p \in [0;1]$ and see what kind of parallelism we need to be faster with smaller cores…
  - Completely abstract curve – we just slow down the sequential part and speed-up the parallel part and see what happens

Big Cores vs. Small Cores. vs. Amdahl's Law (2)

- Here we have 3 times as many cores, but twice as slow
  - Needs 75% of perfectly parallel code just to match the reference system
  - Maximum speed-up is of course $3/2 = 1.5$
- Many current "small cores" systems have a "big core" for the sequential part : typically the host processor for the accelerated system (GPU, etc.)



Dedicated hardware

- Dedicated hardware has been common for a long time
  - Host Channel Adapter, Host Bus Adapter: I/O, networking, etc., …
  - Graphical Processing Units
- But usually dedicated to "general purpose"
  - Every laptop/desktop/workstation needs a graphic display (nowadays)
  - Every server needs I/O and networking
- Re-purposed hardware for alternative use
  - GPGPU usage for computing
- But true, single-task dedicated hardware is coming for highly specialized tasks
  - Google Tensor Processing Unit (TPU), Nvidia Tensor Cores in V100
  - FPGA embedded in CPU

Non-volatile DIMMs (1)

- *Breaking news – Intel Optane in DIMM form factors have been officially announced (May 31st) for GA in 2019*

- Many technologies have been used to implement "Non-volatile RAM"
  - The most obvious are all kind of battery-protected DRAM over the decades
  - Usually as I/O accelerators, sometimes as main memory (NVDIMM-N,F,P,…)

- The issues are price and size of the NVRAM
  - Expensive and not bigger than regular DRAM

- The new promises of some NVDIMMs such as Intel Optane are
  - Performance similar to DRAM, so usable as main memory like DRAM
  - Non-volatile, so usable as permanent storage like Flash
  - Size larger than DRAM
  - Price per GB between Flash and DRAM

Non-volatile DIMMs (2)

- If promises are kept, then the entire I/O stack will need a complete overall
- Access to permanent storage via Load/Store instructions, like memory, instead of I/O-dedicated API and going through the kernel, device drivers, dedicated controller, …
- A single DIMM is about 20 GB/s of raw performance …
  - About 16-18 GB/s of STREAM performance for DRAM
- A single node memory bandwidth is comparable to most common building block for large-scale parallel filesystem
  - DDN ES14KX is 50 GB/s per base enclosure, 500 GB/s per rack
- Even seen as regular I/O devices, NVDIMMs change the trade-offs between I/O access and memory usage
- But they may completely change the landscape of HPC

Non-volatile DIMMs (3)

- "checkpointing" can become extremely easy
  - Any code that doesn't change data "in-place" but use double-buffering becomes "trivial" to checkpoint
    - Just need to know what was the current step was when it was stopped
- Most large-scale HPC programs are of the "load data, do lots of computations of many kinds, write back results" types
  - Because the load data/write back results setps from/to permanent storage are very, very expensive
- If data can be retained in memory easily, then big codes could be broken down into much smaller pieces
  - Faster, easier to write
  - More maintainable/optimizable
  - Communicates data via pointer to permanent memory
  - Dataflow becomes obvious for tuning/reorganization/checkpointing

Non-volatile DIMMs (4)

- Ultimately, move from a compute-centric view to a data-centric view

- Compute-centric
  - The data has to be moved from where it resides (permanent storage) to where it will be worked with (volatile memory + compute devices such as CPU)
  - Multiple compute devices (nodes, GPUs, …) means data moves multiple times

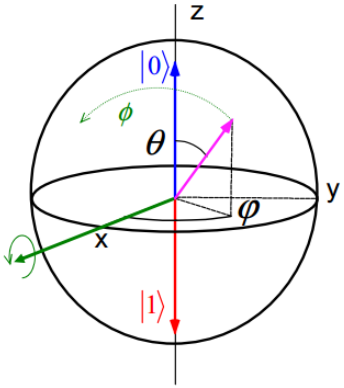
- Data-centric
  - The data doesn't need to move from permanent storage
  - Compute devices works directly on the permanent storage
  - Ultimately, compute devices can be optimized for a given type of computations
    - Having multiple devices hooked to the same memory but with only a subset in use at any given time
    - Routable memory with silicon photonics has been suggested by HP(E) for a while now

Quantum Computing

- Not quite there yet, but lots of work being done
- Completely different paradigm and algorithms
- See the recent ORAP Forum for some excellent introduction to Quantum Computing, Quantum Algorithms, …
  - 41st Forum: Quantum Computing
- Atos at the forefront with the QLM quantum emulator
  - ☺

$$|qb\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$\cos\frac{\theta}{2}e^{-\frac{i\varphi}{2}}|0\rangle + \sin\frac{\theta}{2}e^{+\frac{i\varphi}{2}}|1\rangle$$



Thanks

For more information please contact:

M+ 33 6 40 19 64 42
romain.dolbeau@atos.net

Bull
atos technologies

Extra Slides

dd-mm-yyyy

Bull
atos technologies

Non-x86 vectorization

The ARM Scalable Vector Extension (ARM @ HotChips 2016) (1)

Expanding ARMv8 vector processing

- ARMv7 Advanced SIMD (*aka* ARM NEON instructions) now 12 years old
  - Integer, fixed-point and non-IEEE single-precision float, on *well-conditioned* data
  - 16×128-bit vector registers
- AArch64 Advanced SIMD was an evolution
  - Gained full IEEE double-precision float and 64-bit integer vector ops
  - Vector register file grew from 16×128b to 32×128b
- New markets for ARMv8-A are demanding more radical changes
  - ✓ Gather load & Scatter store
  - ✓ Per-lane predication
  - ✓ Longer vectors
  - … but what is the preferred vector length?

ARM

Bull atos technologies

The ARM Scalable Vector Extension (ARM @ HotChips 2016) (2)

Introducing the Scalable Vector Extension (SVE)

- There is no preferred vector length
  - Vector Length (VL) is hardware choice, from 128 to 2048 bits, in increments of 128
  - *Vector Length Agnostic* (VLA) programming adjusts dynamically to the available VL
  - No need to recompile, or to rewrite hand-coded SVE assembler or C intrinsics
- SVE is not an extension of Advanced SIMD
  - A separate architectural extension with a new set of A64 instruction encodings
  - Focus is HPC scientific workloads, not media/image processing
- Amdahl says you need high vector utilisation to achieve significant speedups
  - Compilers often unable to vectorize due to intra-vector data & control dependencies
  - SVE also begins to address some of the traditional barriers to auto-vectorization

3 ARM

Bull atos technologies

ARM SVE, by comparison

- 32 vector registers, same as AVX512
- Variable vector register length (128-2048 bits)
  - But Fujitsu announces 512 bits for their first implementation, same as AVX512
- 8 (directly usable) + 8 (scratch) "predicate" registers, similar to AVX512 which has 8 directly usable "masking" registers
- Scatter/Gather, same as AVX512
  - But with some "speculative" support for uncounted loops
- …

- So, the main differences are going to be
  1. How much computation hardware is implemented in practice
     - NEON was 128 bits like SSE, but many implementations only use a single 64 bits pipeline (X-Gene, ThunderX) or a single 128 bits pipeline (A57)
     - Also see AMD vs. Intel, etc.
  2. The "Vector Length Agnostic" approach

- Does VLA matter for HPC ?

ARM SVE:VLA, "Vector Length Agnostic" & HPC

- The size of registers is not hardwired in the instruction set
  - Loops work on an implementation-dependent number of elements
- Instruction set offers a lot of support
  - Address computation in abstract vector width
  - etc.
- Bonanza for the embedded world: even when there is many implementations with many different vector width, one binary can be near-optimal everywhere !

- But in the HPC world, we often know what hardware we have: a single implementation, a single vector width, a single optimal instruction set
  - i.e. AVX2 for Haswell, SSE4.2 for Nehalem or AVX512* for KNL
- If a loop is known to have a certain number of elements at compile time, it can be fully unrolled, loop control is removed, and merged with the enclosing loops
  - Can't do that with variable-width registers…

- VLA could be nice for ISV, but is probably not really useful for in-house code

RISC-V Vector Extension (proposal, 11/2016)

https://riscv.org/wp-content/uploads/2016/12/Wed0930-RISC-V-Vectors-Asanovic-UC-Berkeley-SiFive.pdf

- "RISC-V (pronounced "risk-five") is a new instruction set architecture (ISA) that was originally designed to support computer architecture research and education and is now set to become a standard open architecture for industry implementations under the governance of the RISC-V Foundation. The RISC-V ISA was originally developed in the Computer Science Division of the EECS Department at the University of California, Berkeley." (from risc-v.org)

- RISC-V is designed to be very extensible: 32, 64 or 128 bits architecture, with many optional extensions (atomic, floating-point, etc.)
- One extension currently at the "proposal" stage is a vector extension

- Each available vector register (up to 32) is configured with a width and type
  - Easier to mix data types
- Up to 8 mask vectors

- Nice concept, but still in its infancy