Advanced Research Computing

UNIVERSITY OF LEEDS

ARC3 Upgrade

Mark Dixon

m.c.dixon@leeds.ac.uk

ARC3 Upgrade

UNIVERSITY OF LEEDS

- ARC3 entered production 1st March 2017
- Work began on upgrade in April
  - Storage upgraded 1st August
  - Extra standard and high memory nodes added 11th August
  - NVIDIA GPU and Intel “Knights Landing” nodes added 21st September

ARC3 Upgrade

UNIVERSITY OF LEEDS

ARC3 now has the following for general use:

- 2 login nodes (`arc3.leeds.ac.uk`)
- 252 standard compute nodes: **6048** (up from 3960) cores of Intel Xeon “Broadwell” E5-2650v4 (12 cores per CPU) & 32TB RAM
- **836 TB** (up from 390 TB) parallel filesystem (`/nobackup`)
- **4** (up from 2) high-memory nodes, each with 24 cores and 768G RAM
- 2 GPU nodes, each with 2x NVIDIA K80 cards
- **6 GPU nodes, each with 4x NVIDIA P100 12G cards**
- **2 Intel Xeon Phi “Knights Landing” nodes**
- Mellanox FDR InfiniBand interconnect, 56Gbit/s, 2:1 blocking

ARC3 Upgrade

UNIVERSITY OF LEEDS

- Why NVIDIA GPUs?
  - Certain problem types perform well on them
  - Widespread use of NVIDIA GPUs across campus, faculties and problem domains
  - Many popular applications have GPU versions
  - Provide enough GPUs in single node to add capability
  - On-ramp for access to GPUs on regional machines
- Why Intel Xeon Phi's?
  - On-ramp for access to Knights Landing on national/regional machines
  - Representative of Intel architectures in the future (for some value of “future”)

ARC3 Upgrade

UNIVERSITY OF LEEDS

Requesting GPUs and KNLs:

- NVIDIA GPUs
  - 2 GPU nodes, each with 24 cores, 128G, 2x NVIDIA K80 cards:
    `-l coproc_k80=<num>` (`<num>` can be `1` or `2`, job allocated 12 cores and 64G per card requested, each card has 2 GPUs)
  - 6 GPU nodes, each with 24 cores, 256G, 4x NVIDIA P100 12G cards:
    `-l coproc_p100=<num>` (`<num>` can be `1`, `2`, `3` or `4`, job allocated 6 cores and 64G per card requested, each card has 1 GPU)
- Intel Xeon Phi
  - 2 “Knights Landing” nodes, each with 64 cores (plus 4-way hyperthreading), 96G RAM and 16G of high bandwidth memory:
    `-l nodes=1,tpp=64,node_type=256thread-112G,mcdram_mode=<mode>`
    (`<mode>` can be `cache`, `memory` or `half`)

ARC3 Upgrade

UNIVERSITY OF LEEDS

	NVIDIA GPUs	Intel Xeon Phi
Programming	NVIDIA CUDA (C dialect, nvcc compiler).	All existing x86 programming tools work.
	OpenCL (C API, gcc compiler, non-proprietary).	Compile with correct instruction set: `-xMIC-AVX512` (Intel), `-mavx512f -mavx512er -mavx512cd -mavx512pf` (GNU).
	Looking at OpenMP / OpenACC (code annotation of Fortran, C or C++).	
Profiling and debugging	NVIDIA CUDA provides a profiling and optimisation tool.	All existing x86 programming tools work. Use compiler vectorisation reports: -qopt-report -qopt-report-phase=vec (Intel), -O3 -fopt-info-vec (GNU)
	Looking at integrating Allinea ddt and map.	
	Looking at integrating TAU parallel profiler.	

Vectorisation in a nutshell:

- Where the compiler can automatically rewrite something like this:

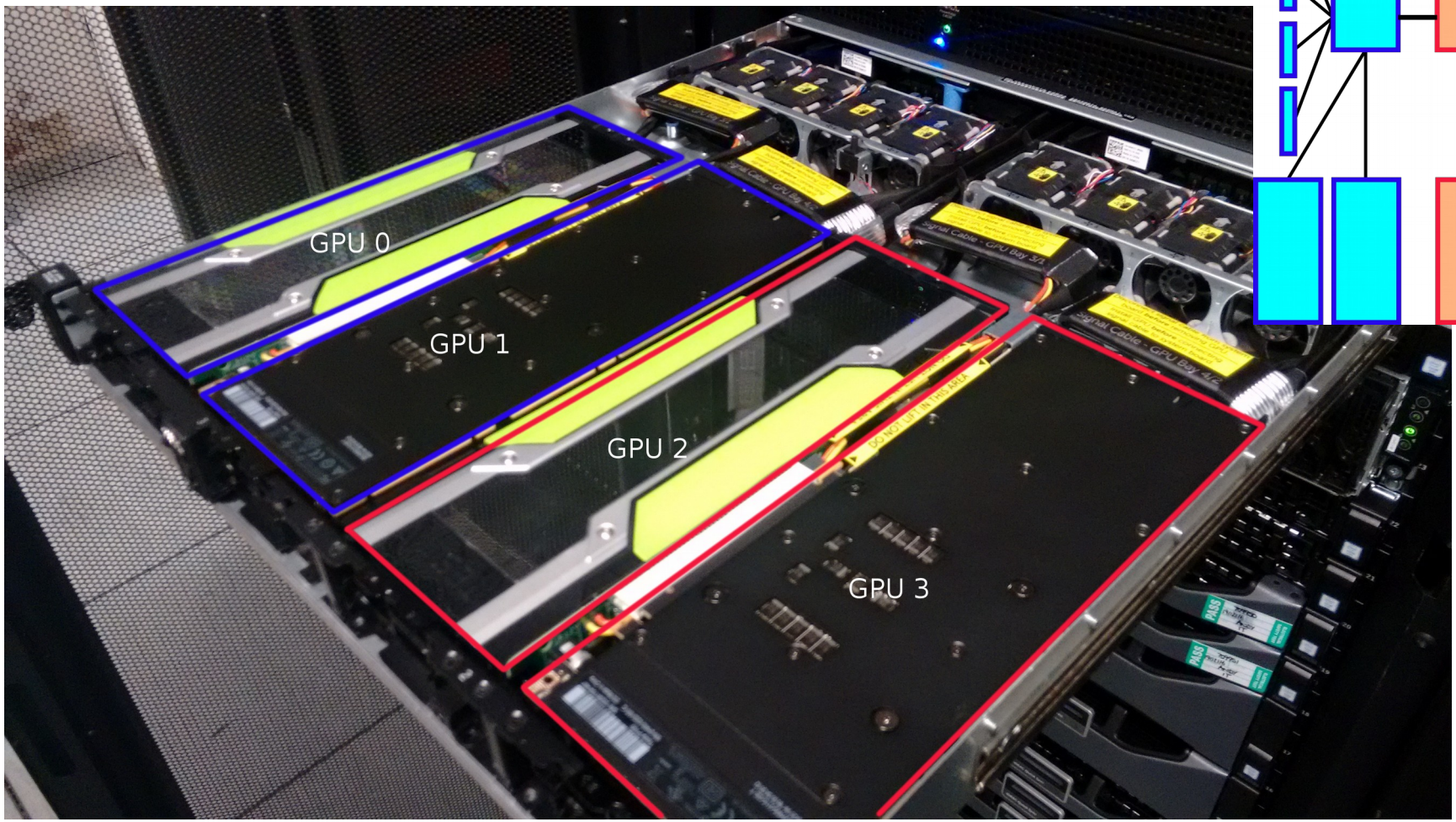
```
for (i = 0, i < SIZE; i++) {
    c[i] = a[i] * b[i];
}
```

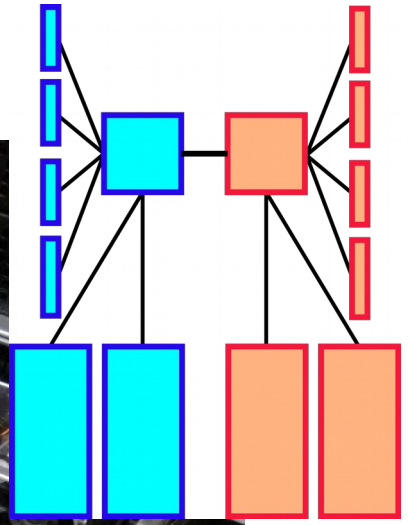
- Into something like this (loop unrolling):

```
for (i = 0, i < SIZE; i += 4) {
    c[i]   = a[i]   * b[i];
    c[i+1] = a[i+1] * b[i+1];
    c[i+2] = a[i+2] * b[i+2];
    c[i+3] = a[i+3] * b[i+3];
}
```

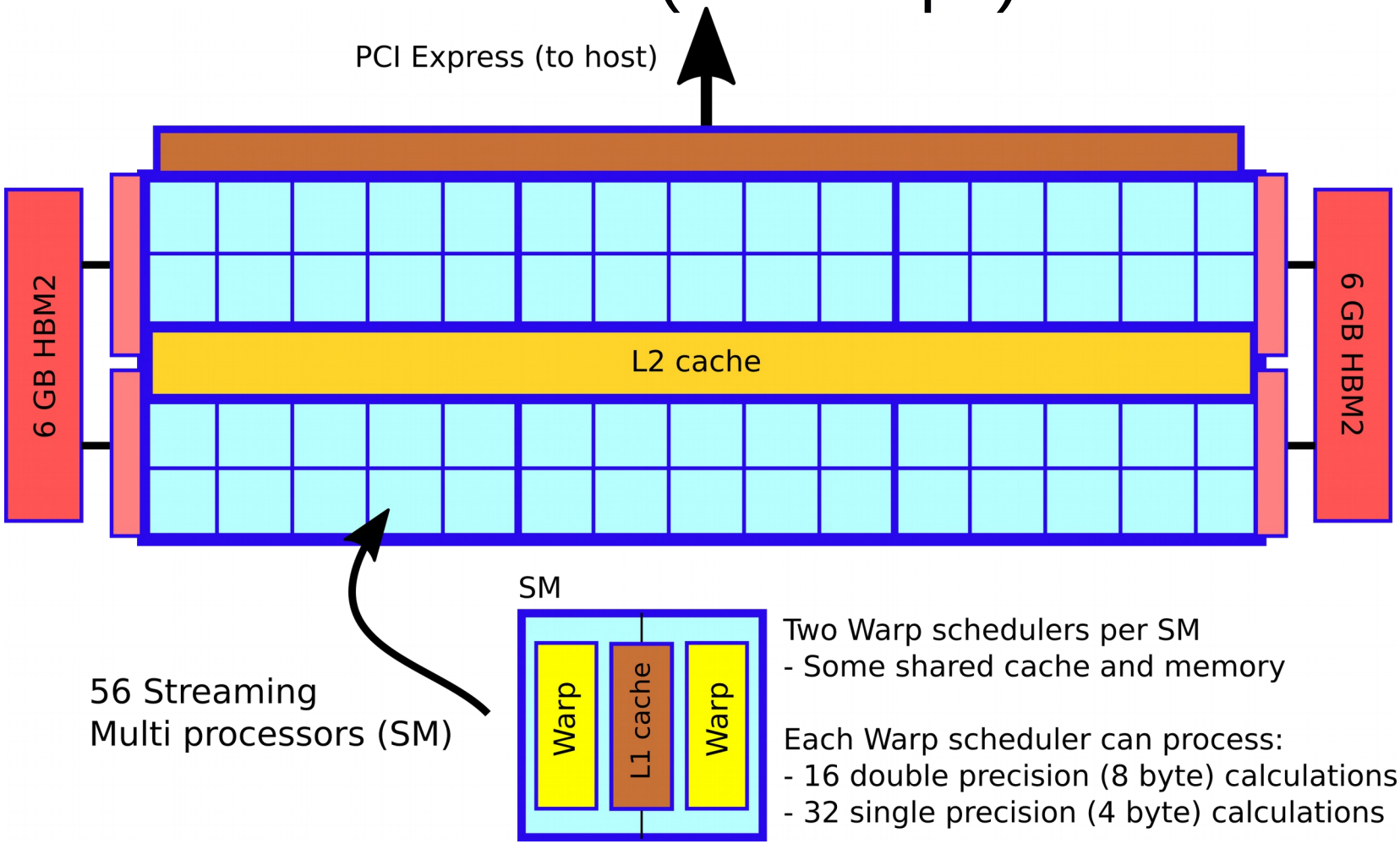
And then the memory layout means it can issue a single instruction to do this.

P100 GPU node

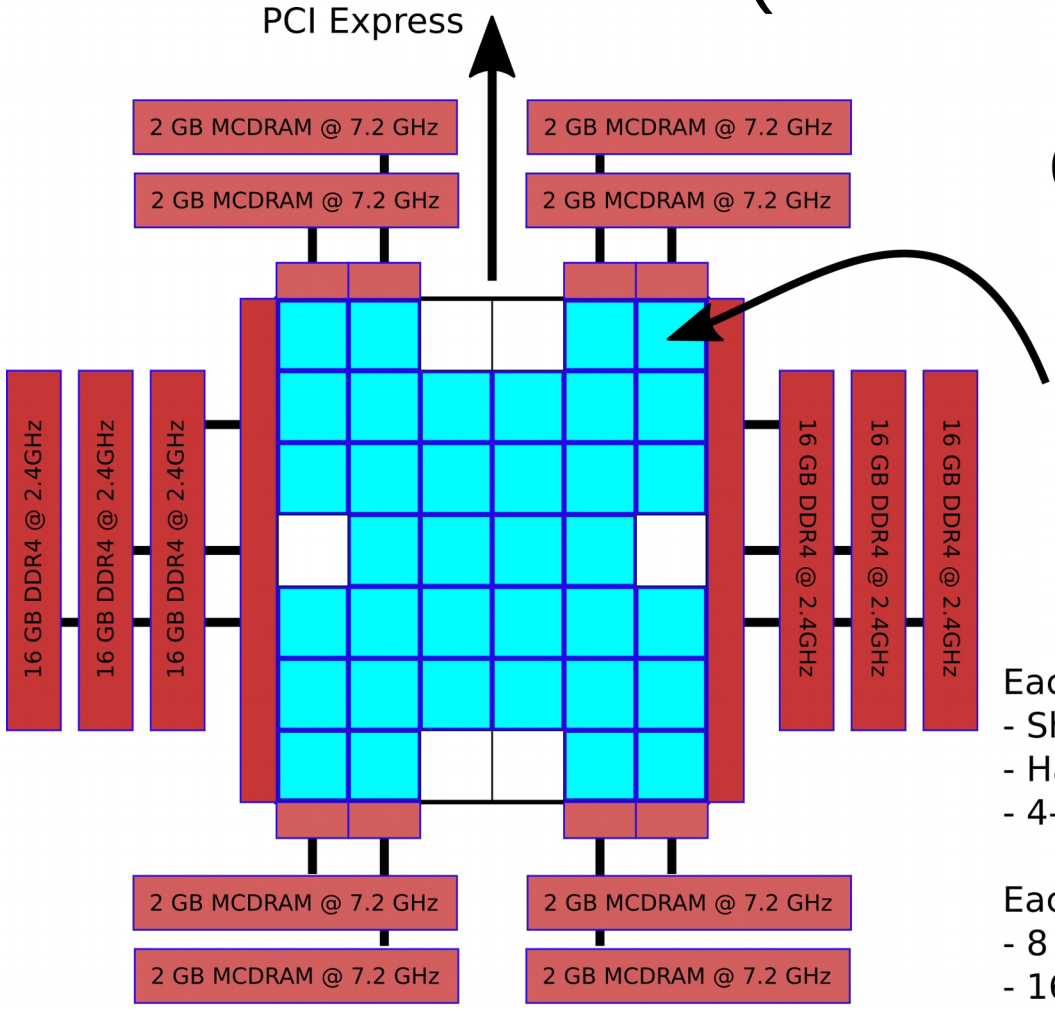




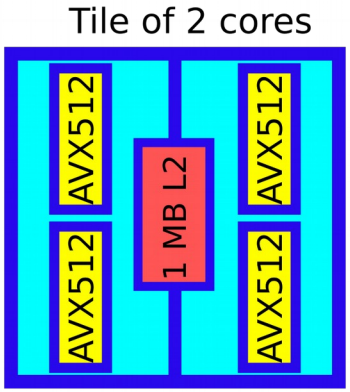
NVIDIA P100 GPU (4.7 Tflops)



Intel Xeon Phi node (2.6 Tflops)



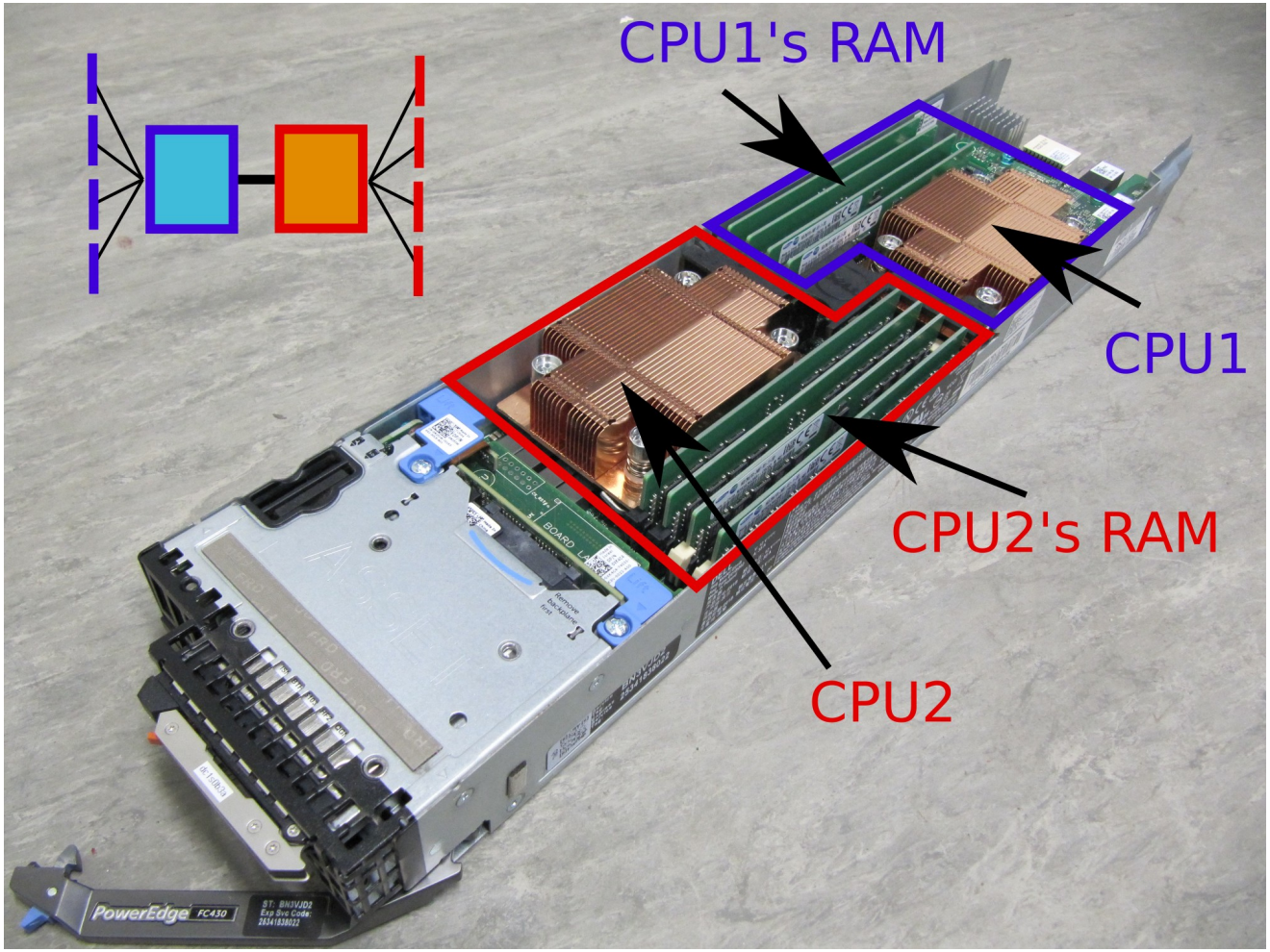
64 cores



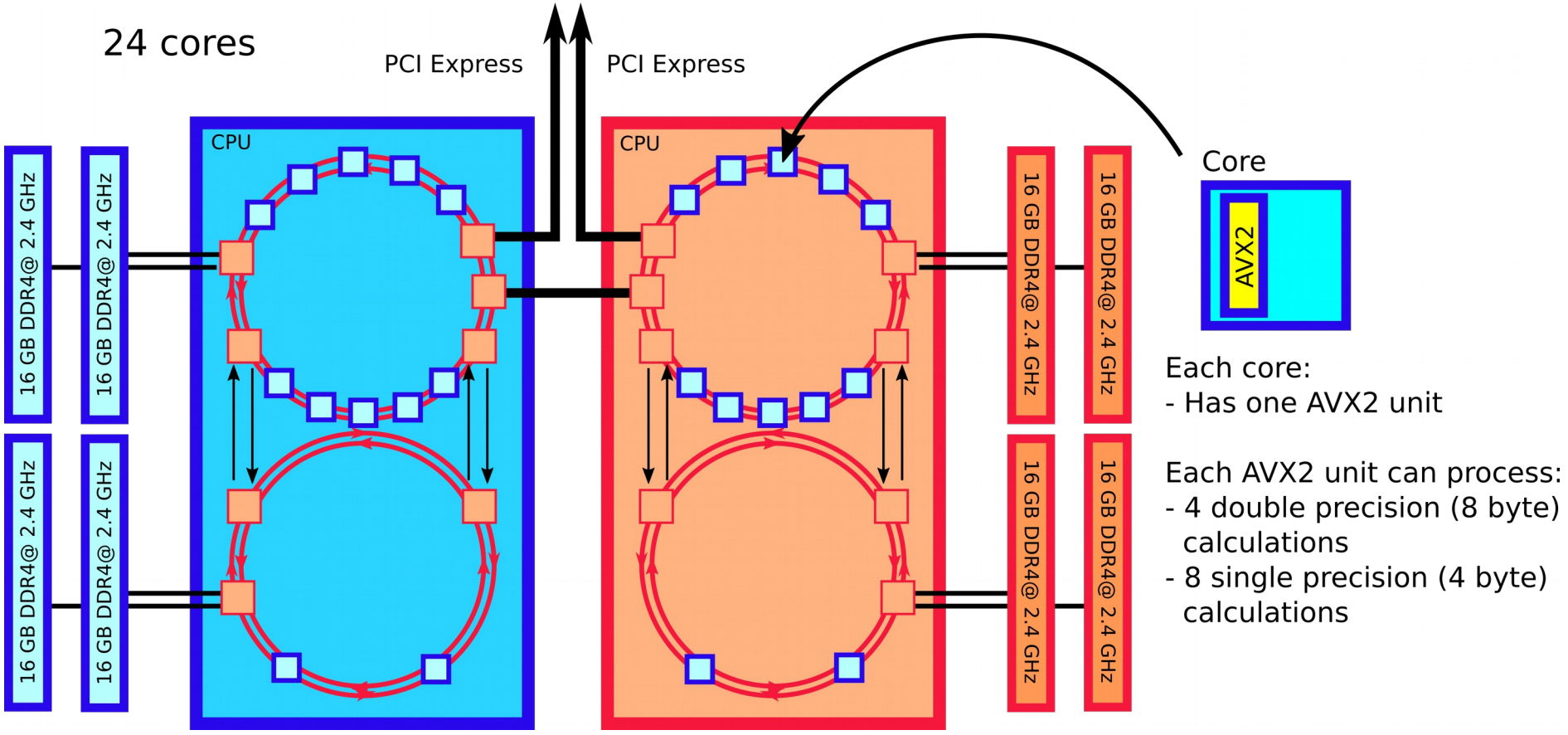
Each core:
- Shares a 1 MB L2 cache
- Has two AVX512 units
- 4-way Hyperthreading

Each AVX512 unit can process:
- 8 double precision (8 byte) calculations
- 16 single precision (4 byte) calculations

Standard compute node (845 Gflops)



Standard compute node (845 Gflops)



Each core:
- Has one AVX2 unit

Each AVX2 unit can process:
- 4 double precision (8 byte) calculations
- 8 single precision (4 byte) calculations

ARC3 Upgrade

UNIVERSITY OF LEEDS

Comparison:

	NVIDIA K80	NVIDIA P100	Intel KNL	2x Intel E5-2650 v4
Calc/s (DP)	2.9 Tflops	4.7 Tflops	2.6 Tflops	845 Gflops
Clock	875 MHz	1.3 GHz	1.3 GHz	2.2 GHz
Vector units (calcs)	104 Warps (48/16 SP/DP)	112 Warps (32/16 SP/DP)	128 AVX512 (16/8 SP/DP)	24 AVX2 (8/4 SP/DP)
Mem size		12 GB	16 GB	128 GB
Mem Bandwidth (per DP flop)		~500 GB/s (~0.11 B/flop)	~500 GB/s (~0.21 B/flop)	~100 GB/s (~0.13 B/flop)

Current focus is on lots of concurrent, more powerful, “vector”, instructions: each instruction operates on multiple numbers at once.

- Those numbers need to be contiguous in memory
- Those numbers need to have a particular byte alignment in memory
- Need big enough arrays to keep vector pipelines full

But on average can only move a fraction of a floating point number from main memory per possible calculation. As ever, cache reuse is incredibly important