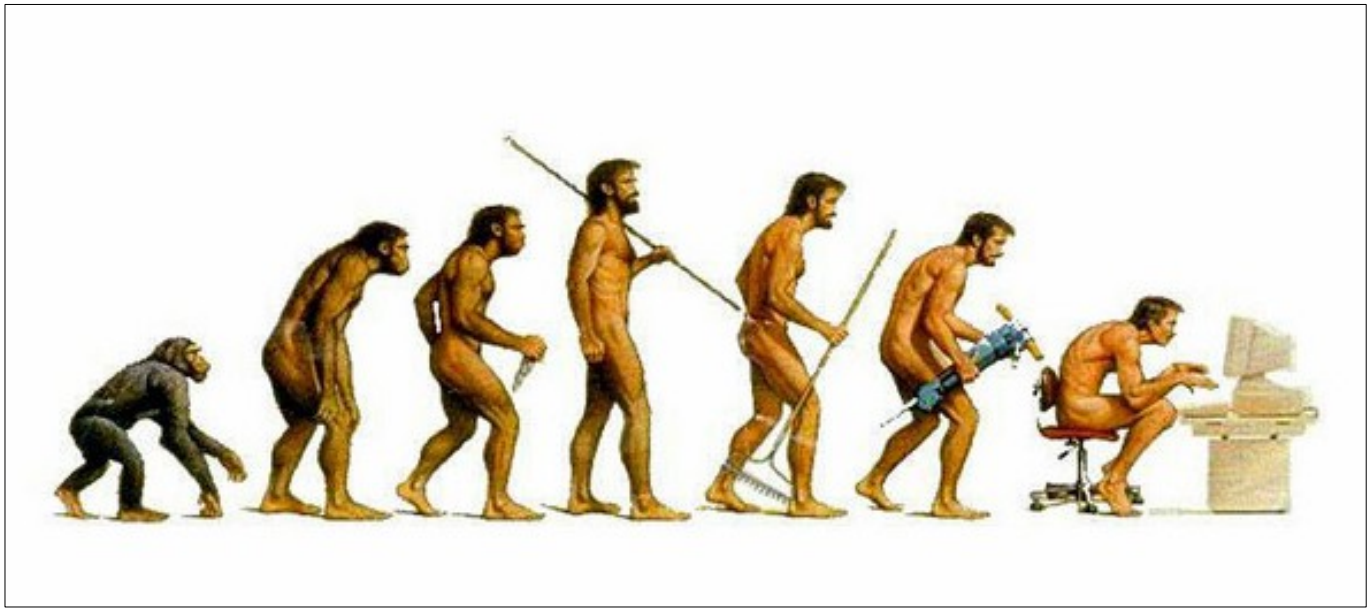
Horizons

University of BRISTOL

Question:



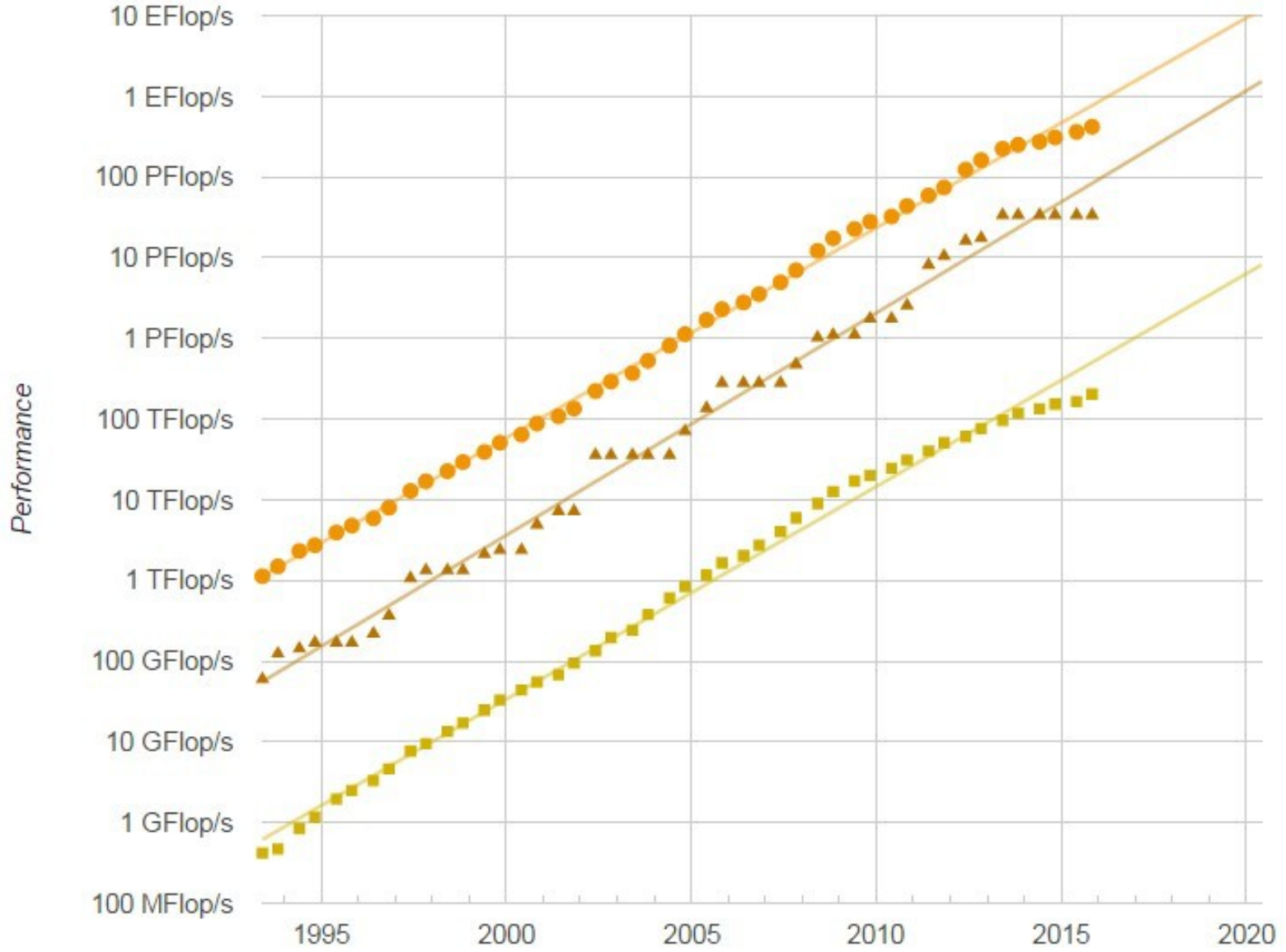
Doing the assignments has been fun … right?:)
But...
What languages and hardware will you be programming in the future?

Never trust...

...someone who claims to know the future!

University of BRISTOL

Top500 trends shifting



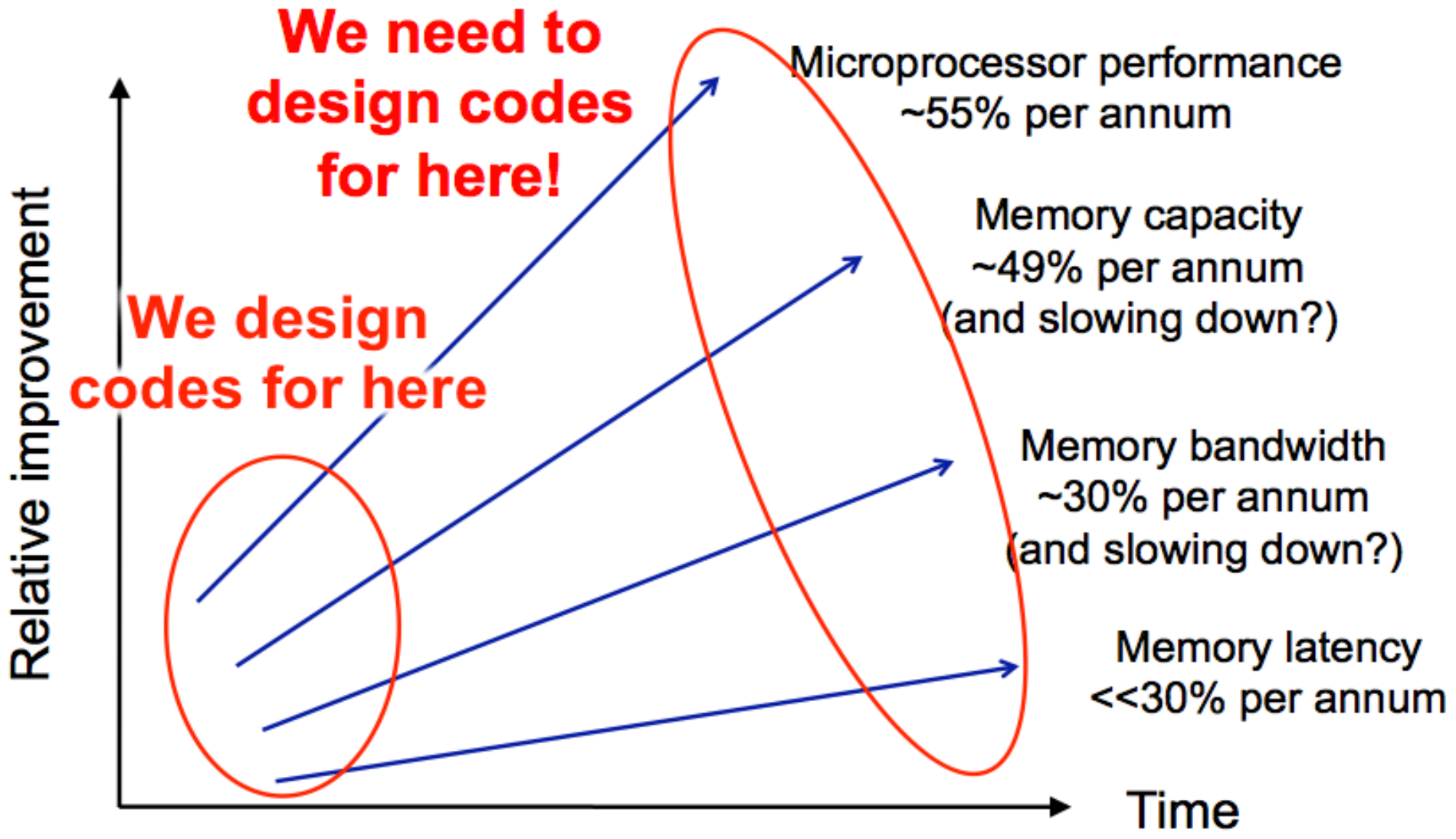
Sum of all machines

#1 machine

#500 machine

University of BRISTOL

The most important HPC trends



University of BRISTOL

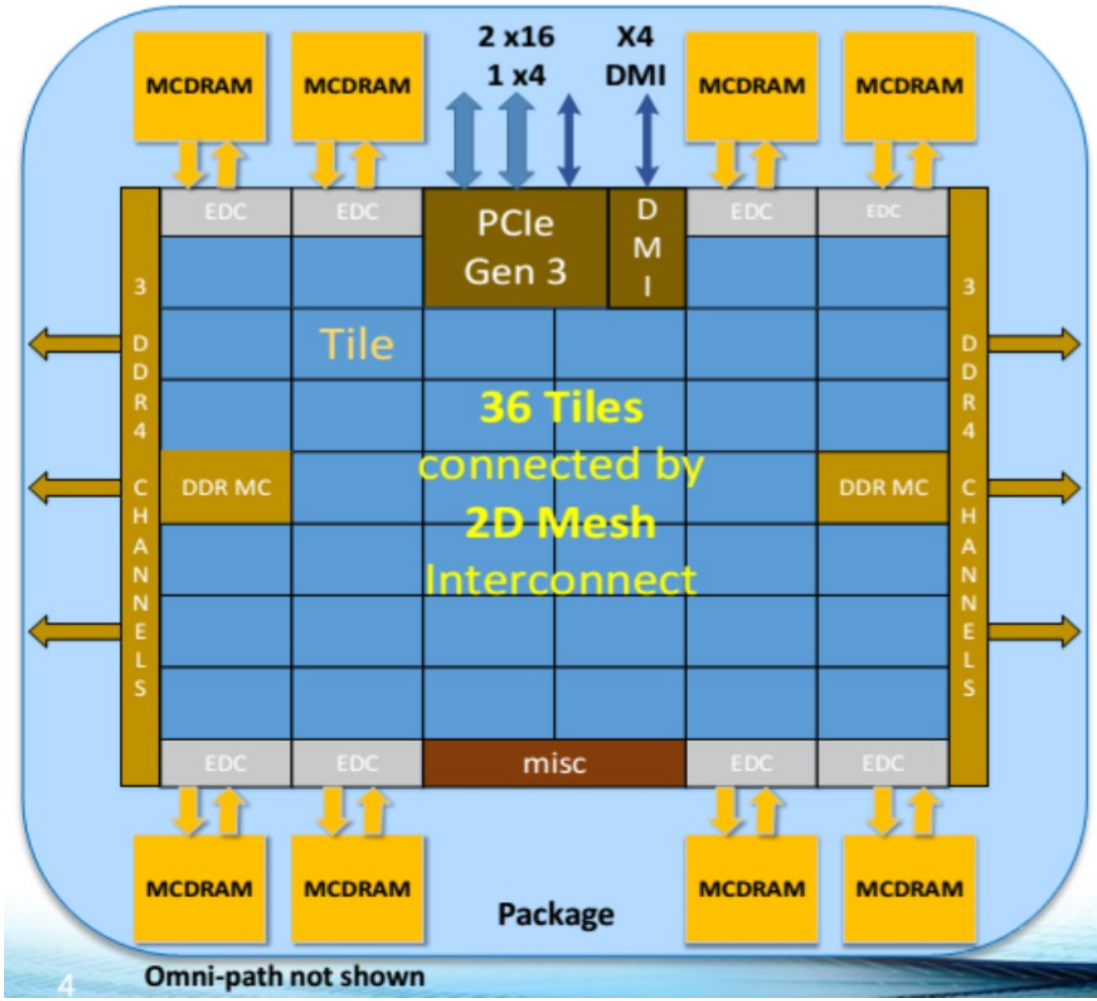
But new architectures are on the way

From 2016:

- Intel Xeon Phi “Knights Landing” (**KNL**)
  - Up to 72 lightweight cores per CPU
  - Two AVX-512 units *per core*
    - 32 FLOPs per cycle per core!
  - Includes 16GB of “stacked” high-bandwidth DRAM with ~500GB/s bandwidth, and also >> 100GB of regular DDR4 DRAM with ~100GB/s bandwidth
  - Self booting – i.e. doesn't have to be a co-processor like KNC, instead KNL can run the OS itself
  - Been deployed in >100 PFLOP machine “Cori” at NERSC

Knights Landing (2016)

Knights Landing Overview



Chip: 36 Tiles interconnected by **2D Mesh**

Tile: 2 Cores + 2 VPU/core + 1 MB L2

Memory: MCDRAM: 16 GB on-package; High BW

DDR4: 6 channels @ 2400 up to 384GB

IO: 36 lanes PCIe Gen3. 4 lanes of DMI for chipset

Node: 1-Socket only

Fabric: Omni-Path on-package (not shown)

Vector Peak Perf: 3+TF DP and 6+TF SP Flops

Scalar Perf: ~3x over Knights Corner

Streams Triad (GB/s): MCDRAM : 400+; DDR: 90+

Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. 1Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). 2Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory. Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

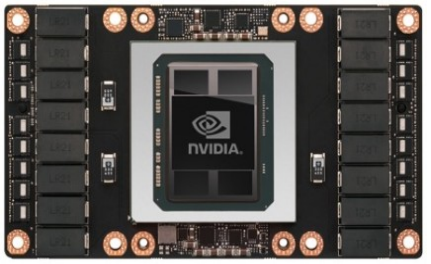
University of BRISTOL

Cori Supercomputer (2016)

- At NERSC, LBNL
- Big Cray, phase 2 based on KNL
- Went live early 2017
- ~9,300 nodes, each a single KNL
- ~28 PFLOP/s double precision (28×10^{15})
- Programmed in OpenMP & MPI
- Future versions will have **>50,000 nodes…**
- **https://www.nersc.gov/users/computational-systems/cori/cori-phase-ii/**

NVIDIA Volta GPU (2017)

- IBM and NVIDIA teaming up so that POWER CPUs can directly connect to GPUs
  - Uses a new interconnect called NVlink
- Very different philosophy from Intel
  - Fewer, much faster (“fatter”) nodes
  - 2 POWER9 CPUs with 2 or 4 Volta GPUs, compared to just 1 Intel KNL per node
- Will be used to build the humungous “Sierra” and “Summit” supercomputers in late 2017
  - Several ***hundred*** PFLOP/s each!!!



University of BRISTOL

NVIDIA Volta (2017)

Aiming for:
- >10 TFLOP/s per GPU
- 500-1000 GB/s per GPU
- Shared virtual memory across CPUs & GPUs
- 80 GB/s between GPUs
  - PCIe has 16GB/s peak!

OAK RIDGE National Laboratory

LAWRENCE LIVERMORE NATIONAL LABORATORY

SUMMIT
150-300 PFLOPS
Peak Performance

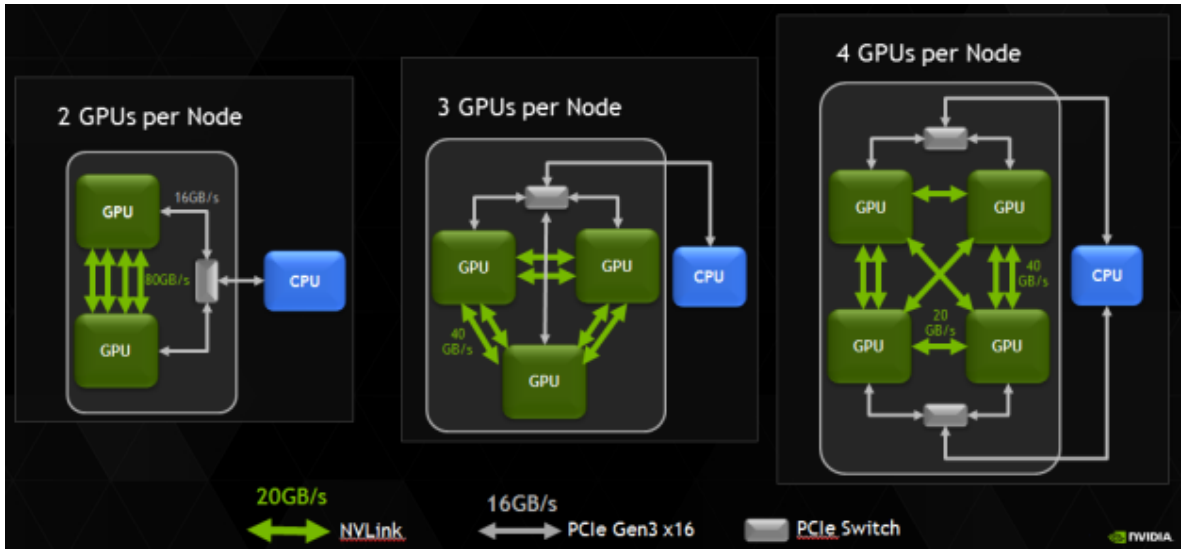
SIERRA
> 100 PFLOPS
Peak Performance

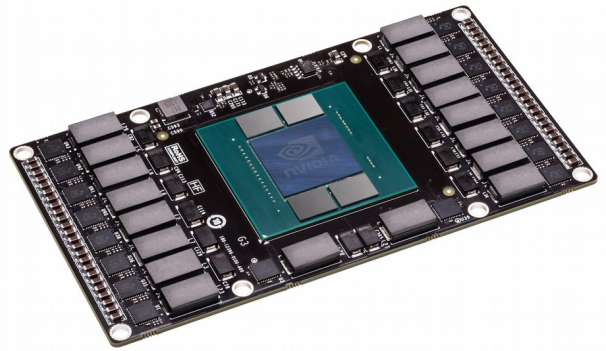
IBM POWER CPU + NVIDIA Volta GPU

NVLink High Speed Interconnect

>40 TFLOPS per Node, >3,400 Nodes

2017





https://hpcuserforum.com/presentations/colorado-sept2015/NVIDIA_Posey2.pdf
http://devblogs.nvidia.com/parallelforall/how-nvlink-will-enable-faster-easier-multi-gpu-computing/

University of BRISTOL

Some Software Trends to Watch..

- Improved shared memory languages, e.g. Kokkos, Raja, Cilk/Cilk++, SYCL, …
  - Can target both CPUs and GPUs
- Compiler directives (OpenMP v4.5, OpenACC)
- Use of threaded libraries, e.g. Threading Building Blocks (TBB)
- Partitioned Global Address Space (PGAS) languages, e.g. Unified Parallel C (UPC)
- Parallel tasking frameworks (HPX, OMP, ...)
- You can try all of these on BCp3.

PGAS (Partitioned Global Address Space): The Big Idea

- Distributed memory programming (e.g. MPI) is difficult and error-prone.
- In contrast, shared memory programming models are much easier to understand.
- Create a single, global address space (shared memory paradigm), but have it partitioned (to exploit locality of reference and hence gain performance).
- Examples include: Chapel, CoArray Fortran, SHMEM, UPC, Titanium, Fortress, x10...

Does PGAS have a future?

- PGAS has been around for ages but has never really taken off
- This is partly because there are several competing standards, and partly because the hardware hasn't supported it efficiently
- Efficient hardware support should radically improve with next generation Cray and Intel interconnects (2018 onwards)
- So we expect PGAS to be much more popular, at least for writing new codes, in the future

How will OpenMP evolve?

```
/* parallel axpy operation in OpenMP 4.5 */
#pragma omp target data map(tofrom y[:n]) map(to x[:n])
{
  #pragma omp target teams distribute
  for (ii=0; ii<n; ii++) {
    y[ii] = a*x[ii] + y[ii];
  }
}
/* only y[] is read back from the device */
```

- Should make GPU programming easier
- Can write portable CPU/GPU code in theory too

Software Trends Summary

- Lots of emerging ways to program shared memory parallel systems
  - Tend to focus on threads
  - Increasingly C++ focused
  - Examples include OpenMP, Cilk, TBB, Kokkos, ...
- Accelerators should become easier to program
  - OpenMP 4.5 supports GPUs, KNL, …
  - Kokkos, Raja, SYCL et al can support CPUs and GPUs
- PGAS
  - Distributed memory parallelism through a shared memory-like parallel programming model.
- Parallel tasking frameworks will become important

Hardware Trends Summary

- We'll soon have hundreds of cores per node
  - Millions of cores per system
- Even embedded processors, such as phones, tablets etc, will have multiple cores, and heterogeneous combinations of CPUs, GPUs and other accelerators
- Wide vector units (512-bit will be common)
- Deep memory hierarchies:
  - L1-L3, HBM, DRAM, NVRAM, SSD, disk, tape, ...

Advanced HPC course intro

- Advanced parallel programming languages:
  - OpenMP 4.x, OpenCL, Kokkos, TBB, Chapel
  - MPI+X
- Advanced parallel programming assignment:
  - Parallelise a lattice Boltzmann 2D structured grid code
  - Using MPI+X
- Material delivered as a series of seminars from experts in the field
  - E.g. have core developers of Chapel and Kokkos coming
- Much more challenging coursework than the Intro unit!
  - 2D halo exchange using MPI
  - More open ended, active exploration required on your part
  - See: https://github.com/UoB-HPC/UoB-HPC-LBM-2016

Intro to HPC course summary

- **All** computer systems are now parallel computers!
- Hardware parallelism will continue to ***increase exponentially***
- You now have a valuable, rare skill
  - **Put it on your CV!**
- Please talk to me about HPC **internships** & **projects**
  - **Including with ACRC next summer!!!**
- With HPC techniques you can literally **change the world** – go and do it!
  - And let me know when you do! ;-)