Advanced High Performance Computing

COMS30006

Simon McIntosh-Smith simonm@cs.bris.ac.uk

NASA hurricane simulation

University of BRISTOL

Course Goals

To build on the foundation of the Intro to HPC course, and expose you to Advanced HPC technologies, including:

- Architectures :
  - Many-core and GPU
  - Next-gen interconnects
  - New memory hierarchies
- Parallel programming
  - Many-core systems
  - Heterogeneous architectures
  - Techniques for fault tolerance
- Trends

OpenMP

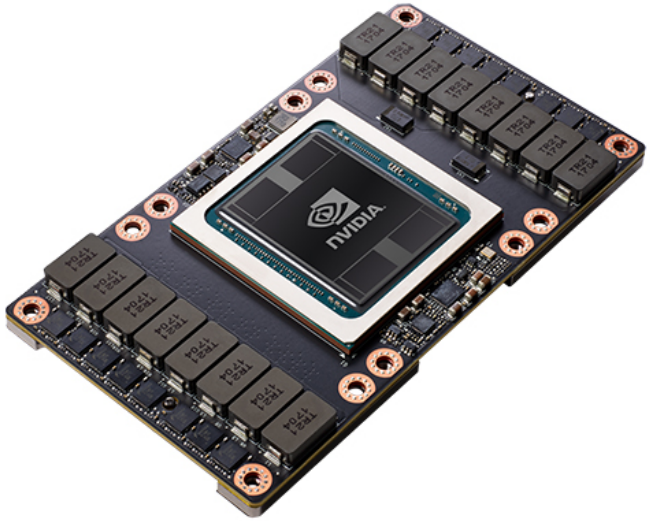
OpenCL

MPI

A key theme throughout will be ***massive parallelism***

Assumptions and Prerequisites

- You've done (and enjoyed) Intro to HPC
- You're already decent programmers
- You're familiar with the C language
- You wanted more time in the Intro course to explore interesting technologies and ideas
- You're a self starter
- You're up for a challenge!



Course Outline

- 10 credit unit → **100 hours in TB2**
  - ~10 hours per week, or 2 hours per weekday
- Split into ~**12 hours for lectures, ~88 hours for labs, coursework** and **personal study**
  - Note: we'll front-load the lectures
- Assessment is **100% coursework** based

Lab Classes

- These are a **core** part of the course
- A chance to discuss ideas or get help with bugs
- MVB 2.11 PC lab, Thursdays 14:00-16:00
- Experienced lab assistants, all using advanced HPC techniques and technologies in their PhDs
  - James Price, Tom Deakin, Matt Martineau, George Pawelczak, Patrick Atkinson & Andrei Poenaru






Course Timetable

- **Two assessments, both using BlueCrystal *phase 4*:**
  1. **Porting a lattice Boltzmann (LBM) code to flat MPI**
  2. **An MPI+X optimised version of LBM running on all the CPUs and GPUs on 4 nodes at the same time**
- ***You need to register for an account on BlueCrystal phase 4 by the end this Tuesday to be ready for the lab this Thursday***
- Not everything we teach you will be assessed
- But **all of it** will be useful and important in your future (that's why we're covering it!)

Course Timetable

- **Week 13-17** 5 weeks Flat MPI Formative
- **Week 19-24** 6 weeks MPI+X 100%

- Deadlines all **5pm on the Friday**
  - Deadlines at end of weeks 17 and 24

- **PLAN** your time to avoid deadline clashes with other units!!!
- Don't leave things to the last minute…

Resources

- **Blue Crystal phase 4 (~£3m):**
  - ~15,000 cores, >800 TFLOP/s
  - 525 nodes, dual 2.4 GHz Intel 14-core processors (Intel Xeon E5-2680 v4 CPUs)
  - 128 GiB memory/node
  - Infiniband high-speed network
  - IBM's General Parallel File System (GPFS)
  - Petascale storage system (~1000TB)
- Accelerator technologies:
  - 64x NVIDIA P100 Tesla GPUs

See: https://www.acrc.bris.ac.uk/acrc/phase4.htm

What Advanced HPC issues will we cover?

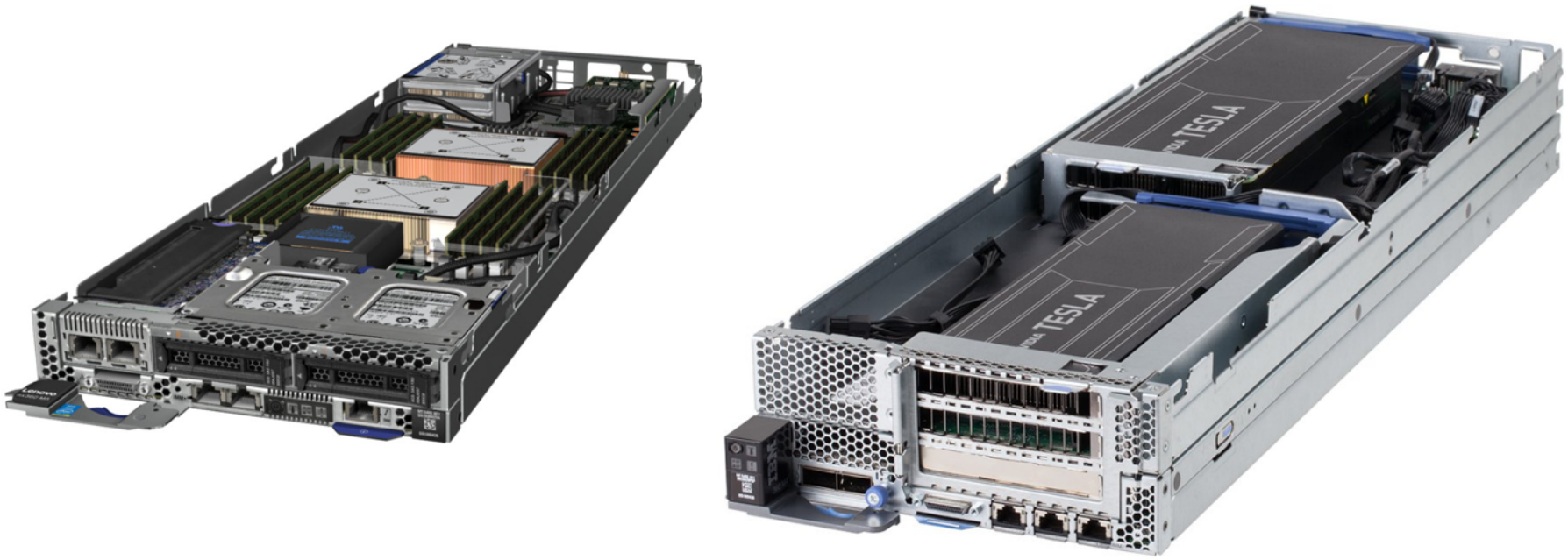
Programming many-core processors

Sunway TaihuLight at NSC in China (fastest in the world)

- 93.01 PetaFLOPS (93.01×10^{15}), 40,960 CPUs
- 260 cores per CPU

Top500: http://top500.org

A node of phase 4



A 2xCPU blade (left) and 4xGPU blade (right)

See: https://lenovopress.com/tips1195-nextscale-nx360-m5-e5-2600-v3

A chassis of phase 4

Lenovo NeXtScale nx360 M5 chassis:

- 6U, up to 12 servers in each (42U per rack)

See: https://lenovopress.com/tips1195-nextscale-nx360-m5-e5-2600-v3

Heterogeneous systems in the Top500

- Tokyo Tech's TSUBAME was first in 2006
  - Started with ClearSpeed, now using GPUs
- Now dozens of GPU-based systems in existence, more on their way
  - Piz Daint, #3 in world, fastest outside China, 19.6 PetaFLOP/s using ~5,000 NVIDIA P100 GPUs

“Fat Nodes”

- Some of the fastest machines currently being built will have dual CPUs and 4-6 GPUs / node
  - Summit: 2xPOWER9 CPUs + 6xV100 GPUs / node
  - 4,600 nodes → 100-200 PFLOP/s at 15 MW

- https://www.olcf.ornl.gov/summit/
- https://www.nextplatform.com/2017/10/05/clever-machinations-livermores-sierra-supercomputer/

Compute Node

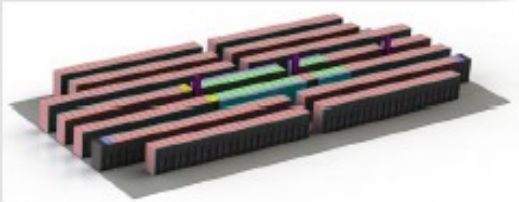
2 IBM POWER9 CPUs
4 NVIDIA Volta GPUs
NVMe-compatible PCIe 1.6 TB SSD
256 GiB DDR4
16 GiB Globally addressable HBM2 associated with each GPU
Coherent Shared Memory

Compute Rack

Standard 19”
Warm water cooling

Compute System

4320 nodes
1.29 PB Memory
240 Compute Racks
125 PFLOPS
~12 MW



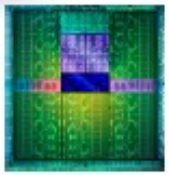
Components

IBM POWER9

- Gen2 NVLink

NVIDIA Volta

- 7 TFlop/s
- HBM2
- Gen2 NVLink



Mellanox Interconnect

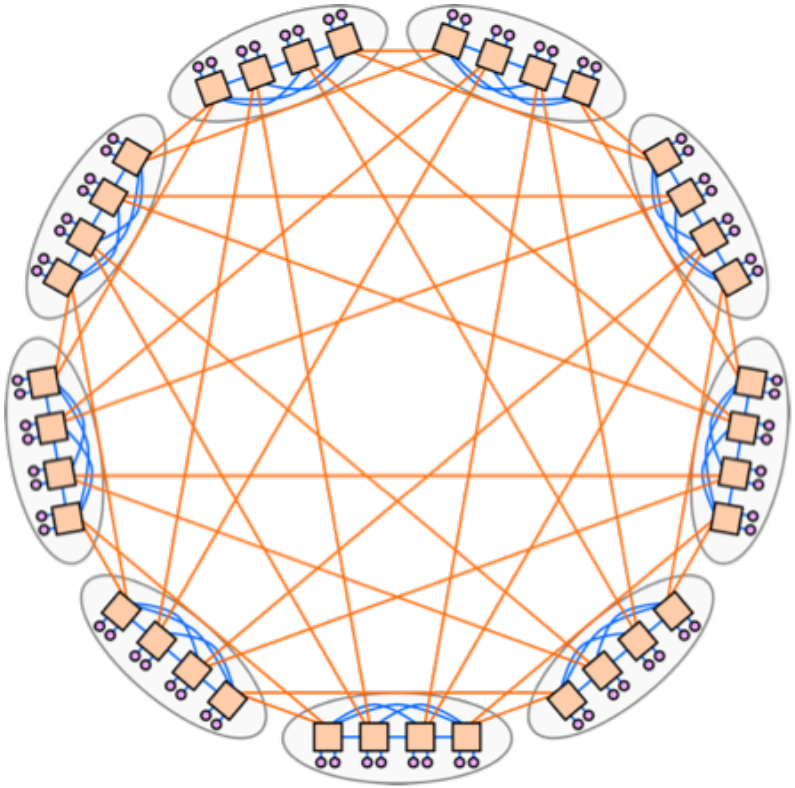
Single Plane EDR InfiniBand
2 to 1 Tapered Fat Tree

GPFS File System

154 PB usable storage
1.54 TB/s R/W bandwidth

Interconnects

- Today's fastest interconnects:
  - Typically fat-tree topologies (**folded-Klos**):
    - Dragonfly etc.
  - InfiniBand or similar
  - Little opportunity left to improve latency
  - Lots more bandwidth to come, but is that useful?
- We were expecting interconnects to become more integrated with the CPUs about now, but this hasn't happened.
  - Why?
- Instead, PCI Express Gen 4 is gaining in popularity
  - 2X PCIe gen 3's 16 GB/s each way

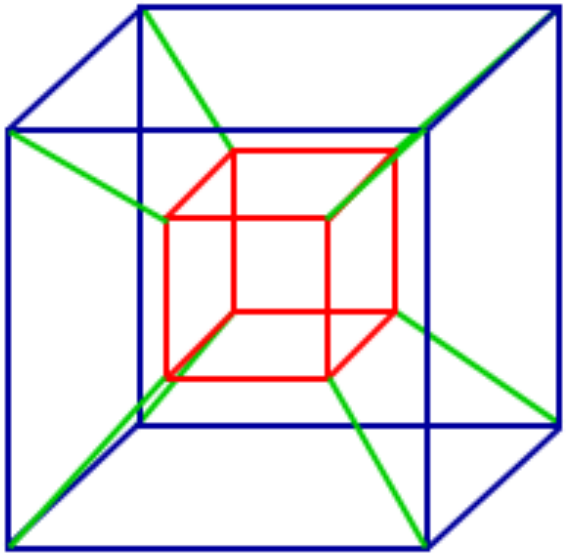


Dragonfly network example:

- 9 groups
- 8 nodes per group

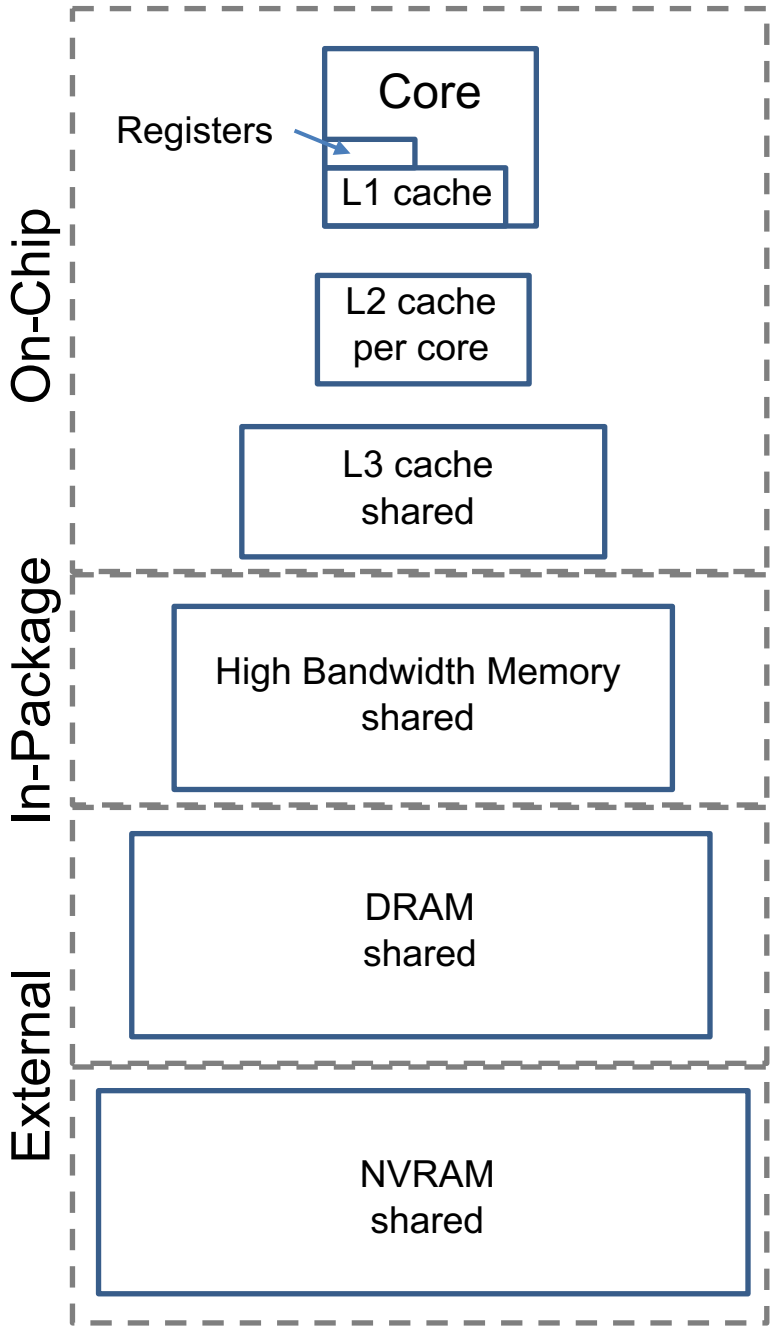
Hypercube network example:

- $N = 2^m$
- 4D (N=16, m=4)
- 16 nodes, each with 4 links



Memory hierarchies

- Blue Crystal phase 3 & 4:
  - CPUs with private L1, L2, shared L3, DRAM
- Summit & Sierra:
  - CPUs (L1-L3, DDR)
  - GPUs (L1, L2, HBM)
  - All memory visible to all processors via a cache-coherent virtual memory system – 512GB per node
- Adding in further levels with NVRAM
  - Adds Terabytes of non-volatile fast storage (like slow DRAM) per node – e.g. ~1TB per node in Summit
  - Radically alters the way we think of storage – could be fully distributed and embedded within the compute with vast aggregate bandwidth, compared to traditional “off the side” attached storage systems



Level 1 (L1) cache (per core):

- Tens of KBytes (KiB) in size
- 1-4 cycles to access

L2 cache (per core):

- Hundreds of KiB in size
- 8-12 cycles to access

L3 cache (shared):

- Tens of MBytes (MiB) in size
- 20-30 cycles to access

High bandwidth memory (HBM, shared):

- Tens of GBytes (GiB) in size
- 200-400 cycles to access

Main memory (DRAM, shared):

- Hundreds of GBytes (GiB) in size
- 200-400 cycles to access

Non Volative memory (NVRAM, shared):

- TBytes (TiB) in size, ~4X slower?

Advanced parallel programming

- Asynchronous many-core programming:
  - Threading Building Blocks (TBB)
- Heterogeneous programming for combined CPU-GPU systems:
  - OpenCL for close to the metal C/C++ code
  - OpenMP 4.5 for C, C++ and Fortran
  - Kokkos for C++ lambda-style parallelism
- MPI+X for large-scale distributed systems
  - X = … OpenMP | OpenCL | Kokkos | MPI | …

Advanced parallel programming

- Are there more ‘productive’ approaches?
- PGAS (Partitioned Global Address Space)
  - Designed for distributed memory systems
  - Present a global (shared) address space
  - Make the notion of locality explicit
  - Goal is to combine the performance of explicit distributed memory programming techniques (such as MPI) with the productivity and ease of use of shared memory approaches (such as OMP)
- We’ll look at Chapel as an example of PGAS

https://chapel-lang.org

So what?

- These advances in computer architectures, memory hierarchies, and interconnects, ***fundamentally change*** how we think about algorithms
  - Need orders of magnitude more parallelism
  - Need to exploit multiple architectures with very different characteristics at the same time
  - Data flow, re-use etc., all change radically
  - Performance portability becomes even more necessary and an even bigger challenge

- Most scientific software developers are completely unprepared for this
- **You** can change that

Summary

- **High Performance Computing** is enabling fundamental breakthroughs in science, engineering and medicine that will increasingly affect every person on the planet
- HPC is becoming increasingly challenging:
  - Orders of magnitude **more parallelism required**
  - **Heterogeneous architectures**
  - Rapidly **deepening memory hierarchies**

Further reading

- Find out about the Blue Crystal phase 4 supercomputer:
  - https://www.acrc.bris.ac.uk/acrc/phase4.htm

- Start understanding about lattice Boltzmann methods:
  - https://en.wikipedia.org/wiki/Lattice_Boltzmann_methods