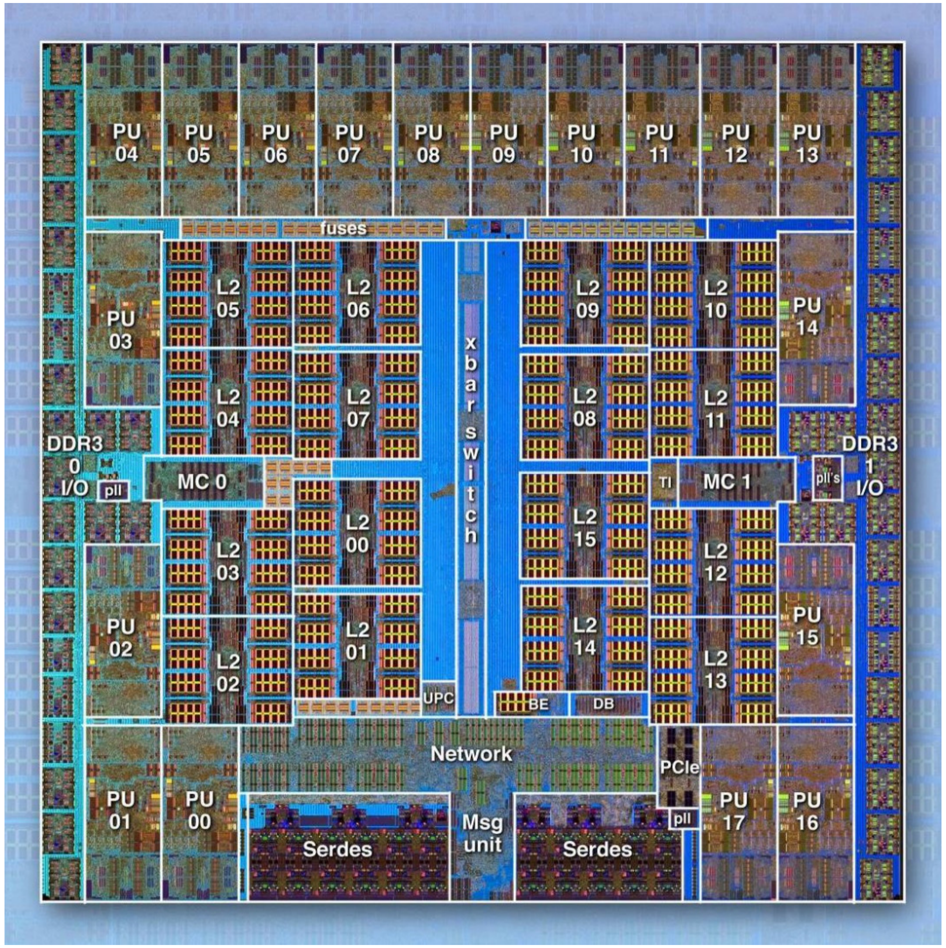
INTRODUCTION TO THE ARCHER KNIGHTS LANDING CLUSTER

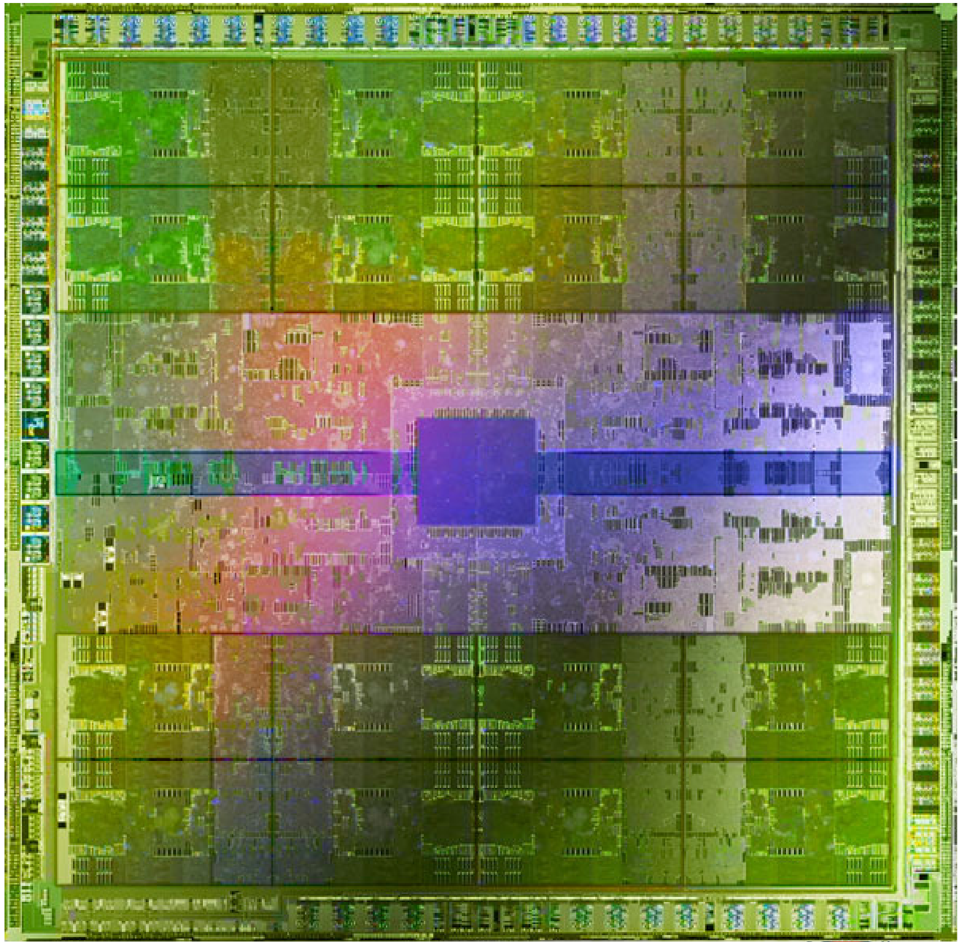
Adrian Jackson
adrianj@epcc.ed.ac.uk
@adrianjhpc

Processors

- The power used by a CPU core is proportional to Clock Frequency x Voltage^2
- In the past, computers got faster by increasing the frequency
  - Voltage was decreased to keep power reasonable.
- Now, voltage cannot be decreased any further
  - 1s and 0s in a system are represented by different voltages
  - Reducing overall voltage further would reduce this difference to a point where 0s and 1s cannot be properly distinguished
- Other performance issues too…
  - Capacitance increases with complexity
  - Speed of light, size of atoms, dissipation of heat
- And practical issues
  - Developing new chips is incredibly expensive
- Must make maximum use of existing technology
- Now parallelism explicit in chip design
  - Beyond implicit parallelism of pipelines, multi-issue and vector units

Multicore processors





Accelerators

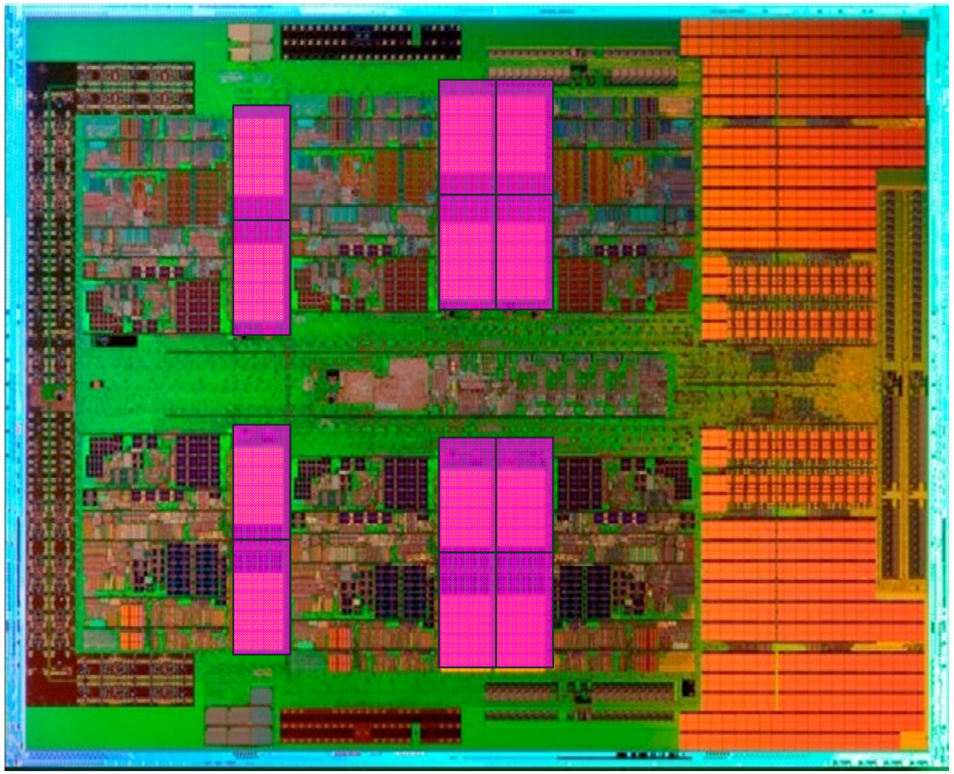
- Need a chip which can perform many parallel operations every clock cycle
  - Many cores and/or many operations per core
  - Floating Point operations (FLOPS) what is generally crucial for computational simulation
- Want to keep power/core as low as possible
- Much of the power expended by CPU cores is on functionality not generally that useful for HPC
  - Branch prediction, out-of-order execution etc

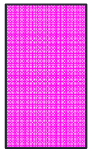
Accelerators

- So, for HPC, we want chips with simple, low power, number-crunching cores
- But we need our machine to do other things as well as the number crunching
  - Run an operating system, perform I/O, set up calculation etc
- Solution: “Hybrid” system containing both CPU and “accelerator” chips

AMD 12-core CPU

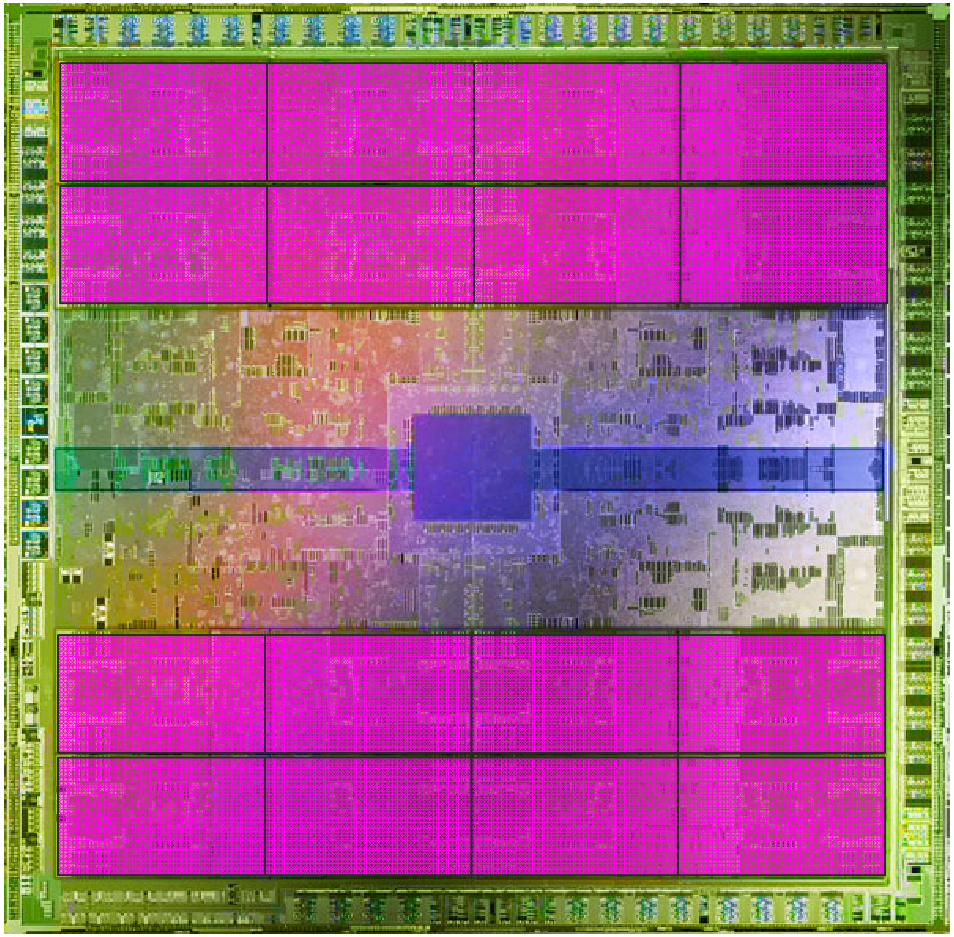
- Not much space on CPU is dedicated to compute





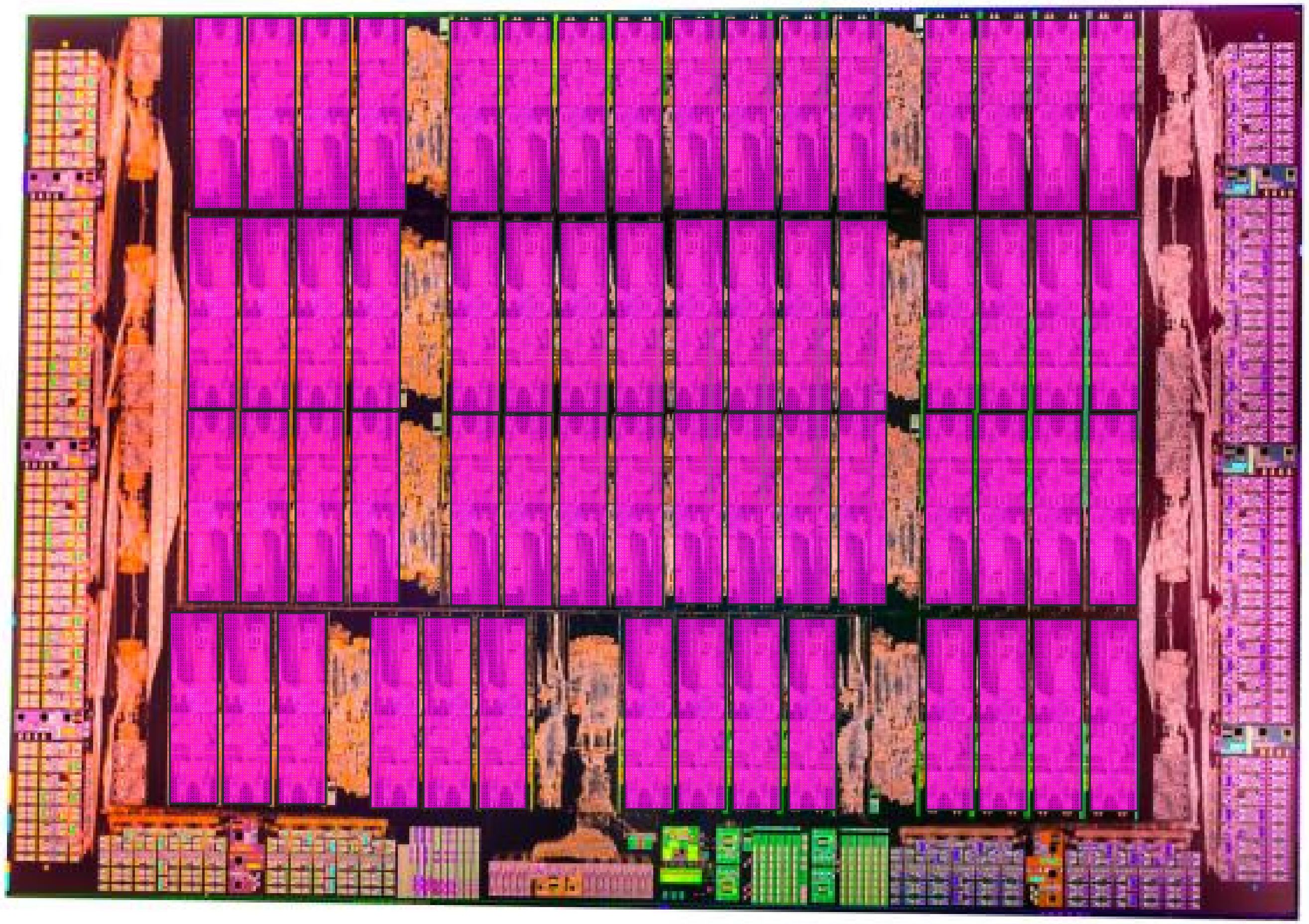
= compute unit
(= core)

NVIDIA Fermi GPU



= compute unit
(= SM
= 32 CUDA cores)

Intel Xeon Phi (KNC)



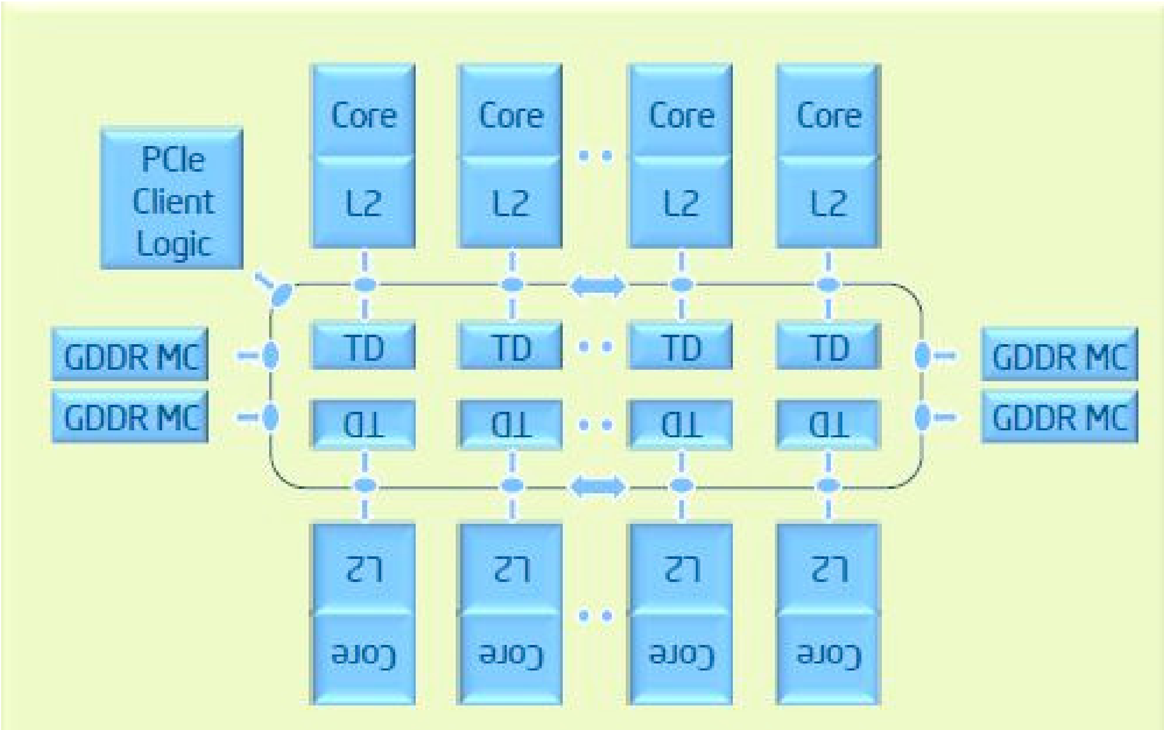
= compute unit (= core)

Intel Xeon Phi - KNC

- Intel Larrabee: “A Many-Core x86 Architecture for Visual Computing”
  - Release delayed such that the chip missed competitive window of opportunity.
  - Larrabee was not released as a competitive product, but instead a platform for research and development (Knight’s Ferry).
- 1st Gen Xeon Phi Knights Corner derivative chip
  - Intel Xeon Phi – co-processor
  - Many Integrated Cores (MIC) architecture. No longer aimed at graphics market
    - Instead “Accelerating Science and Discovery”
  - PCIe Card
  - 60 cores/240 threads/1.054 GHz
  - 8 GB/320 GB/s
  - 512-bit SIMD instructions
- Hybrid between GPU and many-core CPU

KNC

- Each core has a private L2 cache
- “ring” interconnect connects components together
- cache coherent



KNC

- Intel Pentium P54C cores were originally used in CPUs in 1993
  - Simplistic and low-power compared to today's high-end CPUs

- Philosophy behind Phi is to dedicate large fraction of silicone to many of these cores

- And, similar to GPUs, Phi uses Graphics GDDR Memory
  - Higher memory bandwidth that standard DDR memory used by CPUs

KNC

- Each core has been augmented with a wide 512-bit vector unit
- For each clock cycle, each core can operate vectors of size 8 (in double precision)
  - Twice the width of 256-bit “AVX” instructions supported by current CPUs
- Multiple cores, each performing multiple operations per cycle

KNC

	3100 series	5100 series	7100 series
cores	57	60	61
Clock frequency	1.100 GHz	1.053 GHz	1.238 GHz
DP Performance	1 Tflops	1.01 TFlops	1.2 TFlops
Memory Bandwidth	240 GB/s	320 GB/s	352 GB/s
Memory	6 GB	8 GB	16 GB

KNC Systems

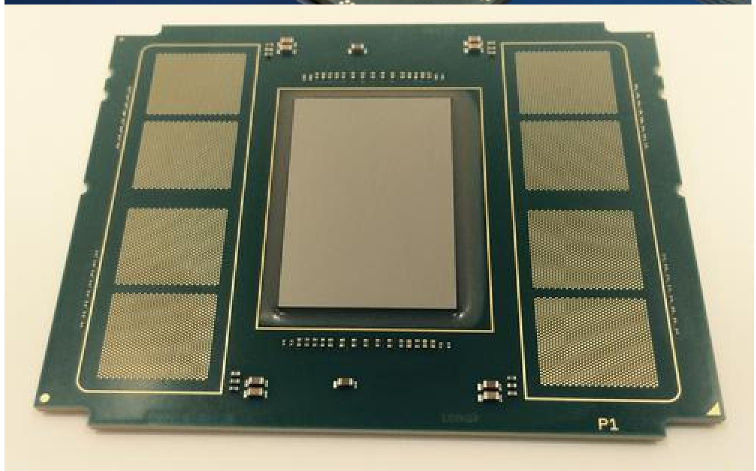
- Unlike GPUs, Each KNC runs an operating system
- User can log directly into KNC and run code
  - “native mode”
  - But any serial parts of the application will be very slow relative to running on modern CPU
- Typically, each node in a system will contain at least one regular CPU in addition to one (or more) KNC
- KNC acts as an “accelerator”, in exactly the same way as already described for GPU systems.
- “Offload mode”: run most source code on main CPU, and offload computationally intensive parts to KNC

KNC: Achievable Performance

- 1 to 1.2 TFlop/s double precision performance
  - Dependent on using 512-bit vector units
  - And FMA instructions
- 240 to 352 GB/s peak memory bandwidth
- ~60 physical cores
  - Each can run 4 threads
  - Must run at least 2 threads to get full instruction issue rate
  - Don't think of it as 240 threads, think of it as 120 plus more if beneficial
- 2.5x speedup over host is good performance
  - Highly vectorised code, no communications costs
- MPI performance
  - Can be significantly slower than host

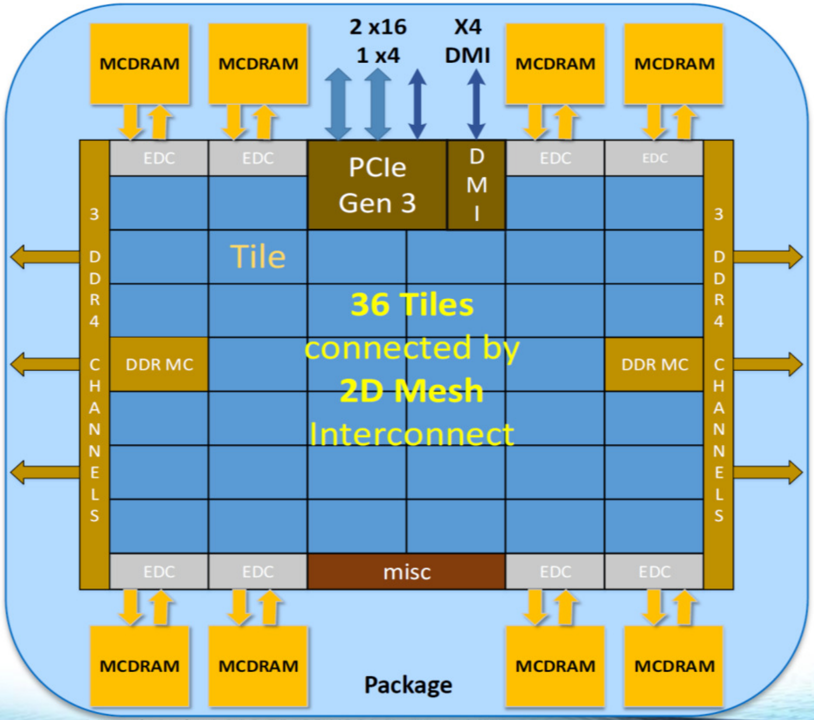
Xeon Phi – Knights Landing (KNL)

- Intel's latest many-core processor
  - Knights Landing
  - 2nd generation Xeon Phi
- Successor to the Knights Corner
  - 1st generation Xeon Phi
- New operation modes
- New processor architecture
- New memory systems
- New cores

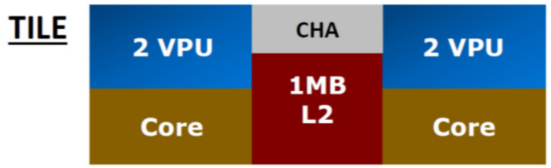


KNL

Knights Landing Overview



Avinash Sodani CGO PPoPP HPCA Keynote 2016



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

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

Memory: MCDRAM: up to 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

Fabric: Intel® Omni-Path Fabric on-package (not illustrated)

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

Scalar Perf: ~3x over Knights Corner

Streams Triad (GB/s): MCDRAM : 450+; DDR: ~90

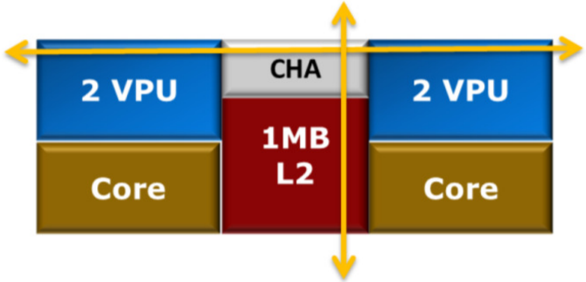
Improvements	What/Why
Self Boot Processor	No PCIe bottleneck
Binary Compatibility with Xeon	Runs all legacy software. No recompilation.
New Core: Atom™ based	~3x higher ST performance over KNC
Improved Vector density	3+ TFLOPS (DP) peak per chip
New AVX 512 ISA	New 512-bit Vector ISA with Masks
Scatter/Gather Engine	Hardware support for gather and scatter
New memory technology: MCDRAM + DDR	Large High Bandwidth Memory → MCDRAM Huge bulk memory → DDR
New on-die interconnect: Mesh	High BW connection between cores and memory
Integrated Fabric: Omni-Path	Better scalability to large systems. Lower Cost

KNL

KNL Tile:

2 Cores, each with 2 VPU

1M L2 shared between two Cores



Core: New OoO Core. Balances power efficiency, parallel and single thread performance.

2 VPU: 2x AVX512 units. 32SP/16DP per unit. X87, SSE, AVX, AVX2 and EMU

L2: 1MB 16-way. 1 Line Read and ½ Line Write per cycle. Coherent across all Tiles

CHA: **C**aching/**H**ome **A**gent. Distributed Tag Directory to keep L2s coherent. MESIF protocol. 2D-Mesh connections for Tile

Avinash Sodani CGO PPoPP HPCA Keynote 2016

KNL vs KNC

KNIGHTS LANDING VS. KNIGHTS CORNER FEATURE COMPARISON

FEATURE	INTEL® XEON PHI™ COPROCESSOR 7120P	KNIGHTS LANDING PRODUCT FAMILY
Processor Cores	Up to 61 enhanced P54C Cores	Up to 72 enhanced Silvermont cores
Key Core Features	In order 4 threads / core (back-to-back scheduling restriction) 2 wide	Out of order 4 threads / core 2 wide
Peak FLOPS[1]	SP: 2.416 TFLOPs • DP: 1.208 TFLOPs	Up to 3x higher
Scalar Performance[1]	1X	Up to 3x higher
Vector ISA	x87, (no Intel® SSE or MMX™), Intel IMIC	x87, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, Intel® AVX, AVX2, AVX-512 (no Intel® TSX)
Interprocessor Bus	Bidirectional Ring Interconnect	Mesh of Rings Interconnect

1- Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes.
Any differences in your system hardware, software or configuration may affect your actual performance.

L2 cache sharing

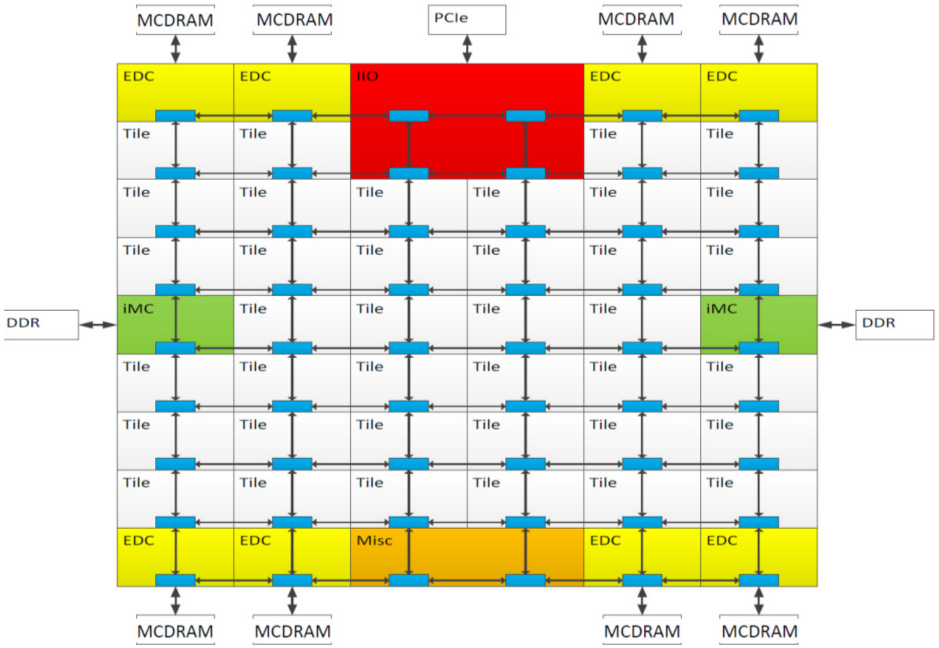
- L2 cache is shared between cores on a tile
- Capacity depends on data locality
  - No sharing of data between core: 512kb per core
  - Sharing data: 1MB for 2 cores
- Gives fast communication mechanism for processes/threads on same tile
  - May lend itself to blocking or nested parallelism

Hyperthreading

- KNC required at least 2 threads per core for sensible compute performance
  - Back to back instruction issues were not possible
- KNL does not
  - Can run up to 4 threads per core efficiently
  - **Running 3 threads per core is not sensible**
    - Resource partitioning reduces available resources for all threads
  - A lot of applications don't need any hyperthreads
    - Much more like ARCHER Ivybridge hyperthreading now

KNL

KNL Mesh Interconnect



Mesh of Rings

- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

Cache Coherent Interconnect

- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

Three Cluster Modes

(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering hemisphere

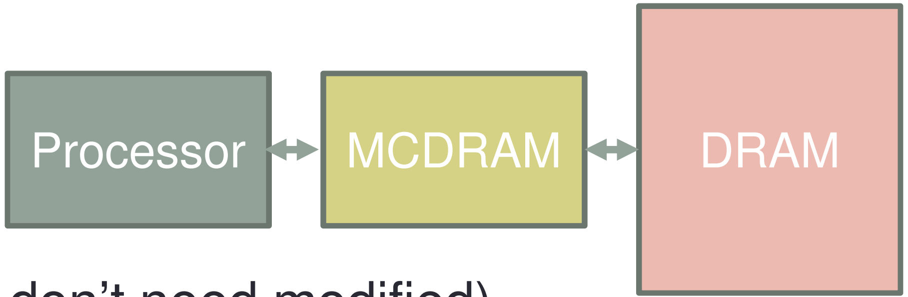
Avinash Sodani CGO PPoPP HPCA Keynote 2016

Memory

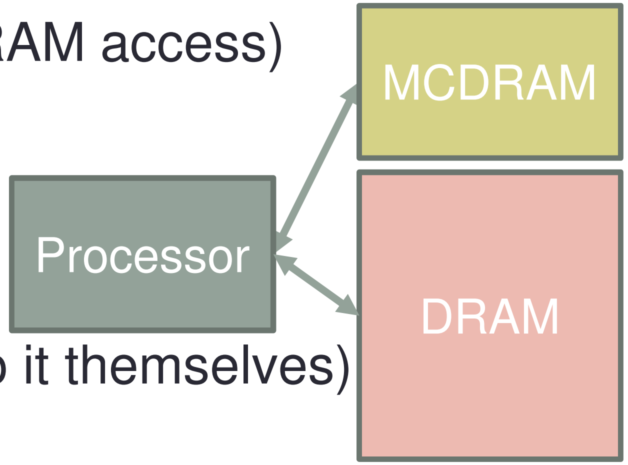
- Two levels of memory for KNL
  - Main memory
    - KNL has direct access to all of main memory
    - Similar latency/bandwidth as you'd see from a standard processors
    - 6 DDR channels
  - MCDRAM
    - High bandwidth memory on chip: 16 GB
    - Slightly higher latency than main memory (~10% slower)
    - 8 MCDRAM controllers/16 channels

Memory Modes

- Cache mode
  - MCDRAM cache for DRAM
  - Only DRAM address space
  - Done in hardware (applications don't need modified)
  - Misses more expensive (DRAM and MCDRAM access)



- Flat mode
  - MCDRAM and DRAM are both available
  - MCDRAM is just memory, in same address
  - Software managed (applications need to do it themselves)



- Hybrid – Part cache/part memory
  - 25% or 50% cache

Compiling for the KNL

- Standard KNL compilation targets the KNL vector instruction set
  - This **won't** run on standard processor
  - Binaries that run on standard processors **will** run on the KNL
- If your build process executes programs this may be an issue
  - Can build a fat binary using Intel compilers

  `-ax MIC-AVX-512,AVX`

  - For other compilers can do initial compile with KNL instruction set
    - Then re-compile specific executables with KNL instruction set
    - i.e. `-aAVX` for Intel, `-hcpu=…` for Cray, `-march=…` for GNU

ARCHER KNL

- 12 nodes in test system
- ARCHER users get access
  - Non-ARCHER users can get access through driving test
- Initial access will be unrestricted
  - Charging will come in soon (near end of November)
  - Charging will be same as ARCHER (i.e. 1 node hour = 0.36 kAUs)
- Each node has
  - 1 x Intel(R) Xeon Phi(TM) CPU 7210 @ 1.30GHz
    - 64 core/4 hyperthreads
    - 16GB MCDRAM
  - 96GB DDR4@2133 MT/s

System setup

- XC40 system integrated with ARCHER
  - Shares /home file system
- KNL system has it's own login nodes: `knl-login`
  - Not accessible from the outside world
  - Have to login in to the ARCHER login nodes first
  - ssh to login.archer.ac.uk then ssh to knl-login
  - Username is same as ARCHER account username
- Compile jobs there
  - Different versions of the system software from the standard ARCHER nodes
- Submit jobs using PBS from those nodes
- Has it's own /work filesystem (scratch space)

  `/work/knl-users/$user`

Programming the KNL

- Standard HPC - parallelism
  - MPI
  - OpenMP
    - Default `OMP_NUM_THREADS` may be 256
  - mkl
- Standard HPC – compilers
  - module craype-mic-knl (loaded by default on knl-login nodes)
  - Intel compilers

    `-xMIC-AVX512` (without the module)
  - Cray compilers

    `-hcpu=mic-knl` (without the module)
  - GNU compilers

    `-march=knl` or `-mavx512f -mavx512cd -mavx512er -mavx512pf` (without the module)

Running applications on the XC40

- You will have a separate budget on the KNL system
  - Name is: `k01-$USER` i.e. `k01-adrianj`
- Use `PBS` and `aprun` as in ARCHER
  - Standard PBS script, with one extra for selecting memory/communication setup (more later)
  - Standard `aprun`, run 64 MPI processes on the 64 KNL cores:

```
aprun -n 64 ./my_app
```

- 256 threads per KNL processor
  - Numbering wraps, i.e. 0-63 the hardware cores, 64-127 wraps onto the cores again, etc…
  - Meaning core 0 has threads 0,64,128,192, core 1 has threads 1,65,129,193, etc…

Running applications on the XC40

- For hyperthreading (using more than 64 cores):

```
OMP_NUM_THREADS=4
aprun -n 256 -j 4 ./my_app
```

or

```
aprun -n 128 -j 2 ./my_other_app
```

- Should also be possible to control thread placement with `OMP_PROC_BIND`:

```
OMP_PROC_BIND=true
OMP_NUM_THREADS=4
aprun -n 64 -cc none -j 4 ./my_app
```

Configuring KNL

- Different memory modes and cluster options
  - Configured at boot time
    - Switching between cache and flat mode
    - Switching cluster modes
- For ARCHER XC40 Cluster configuration is done through batch system (PBS)
- Modes can be requested as a resource:

```
#PBS -l select=4:aoe=quad_100
#PBS -l select=4:aoe=snc2_50
```

  - This is in the form :`aoe=numa_cfg_hbm_cache_pct`
- Available modes are:
  - For the NUMA configuration (`numa_cfg`): a2a, snc2, snc4, hemi, quad
  - For the MCDRAM cache configuration (`hbm_cache_pct`): 0, 25, 50, 100
- So for quadrant mode and flat memory (MCDRAM and DRAM separate) this would be:

```
#PBS -l select=4:aoe=quad_0
```

- Currently not enabling changing KNL setup

Current configuration

```
adrianj@login:~> apstat -M
NID Memory(MB) HBM(MB) Cache(MB) NumaCfg
 44      98304   16384     16384    quad
 45      98304   16384     16384    quad
 46      98304   16384     16384    quad
 47      98304   16384     16384    quad
 48      98304   16384     16384    quad
 49      98304   16384     16384    quad
 50      98304   16384     16384    quad
 51      98304   16384     16384    quad
 52      98304   16384     16384    quad
 53      98304   16384     16384    quad
 54     114688   16384         0    quad
 55     114688   16384         0    quad
adrianj@login:~>
```

Test data hardware

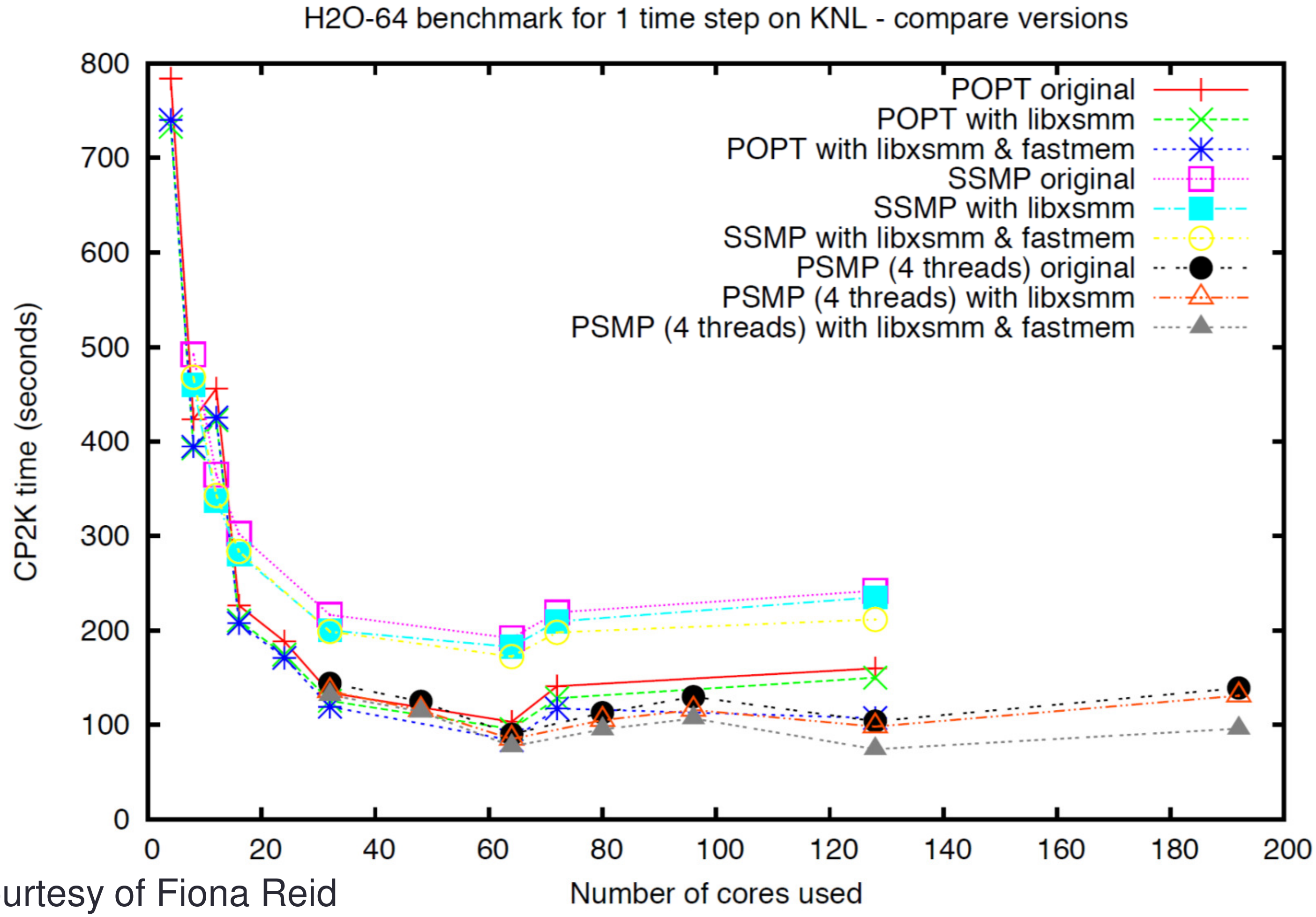
- Same KNL as the ARCHER ones
- Intel(R) Xeon Phi(TM) CPU 7210 @ 1.30GHz
  - 64 core
  - 16GB MCDRAM
  - 215W TDP
  - 1.3Ghz TDP, 1.1Ghz AVX
  - 1.6Ghz Mesh
  - 6.4GT/s OPIO
  - 96GB DDR4@2133 MT/s

Performance

- Initial performance experiences with a single KNL

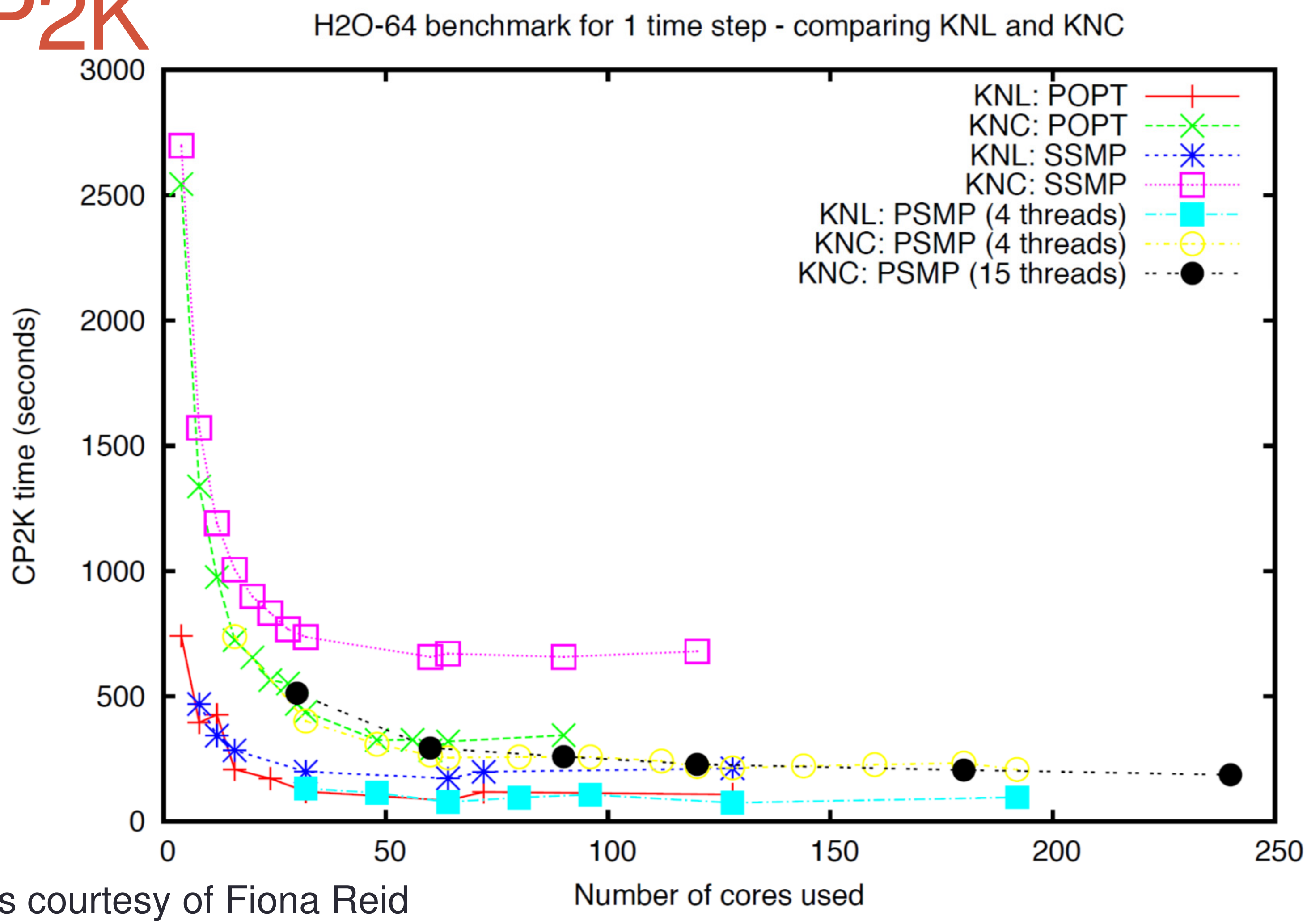
Application	KNC	KNL	KNL HB	IvyBridge	Broadwell
COSA		561	450	497	349
GS2	400	184.2	103.8	126.6	83.4
CASTEP		149	146	102	38

CP2K



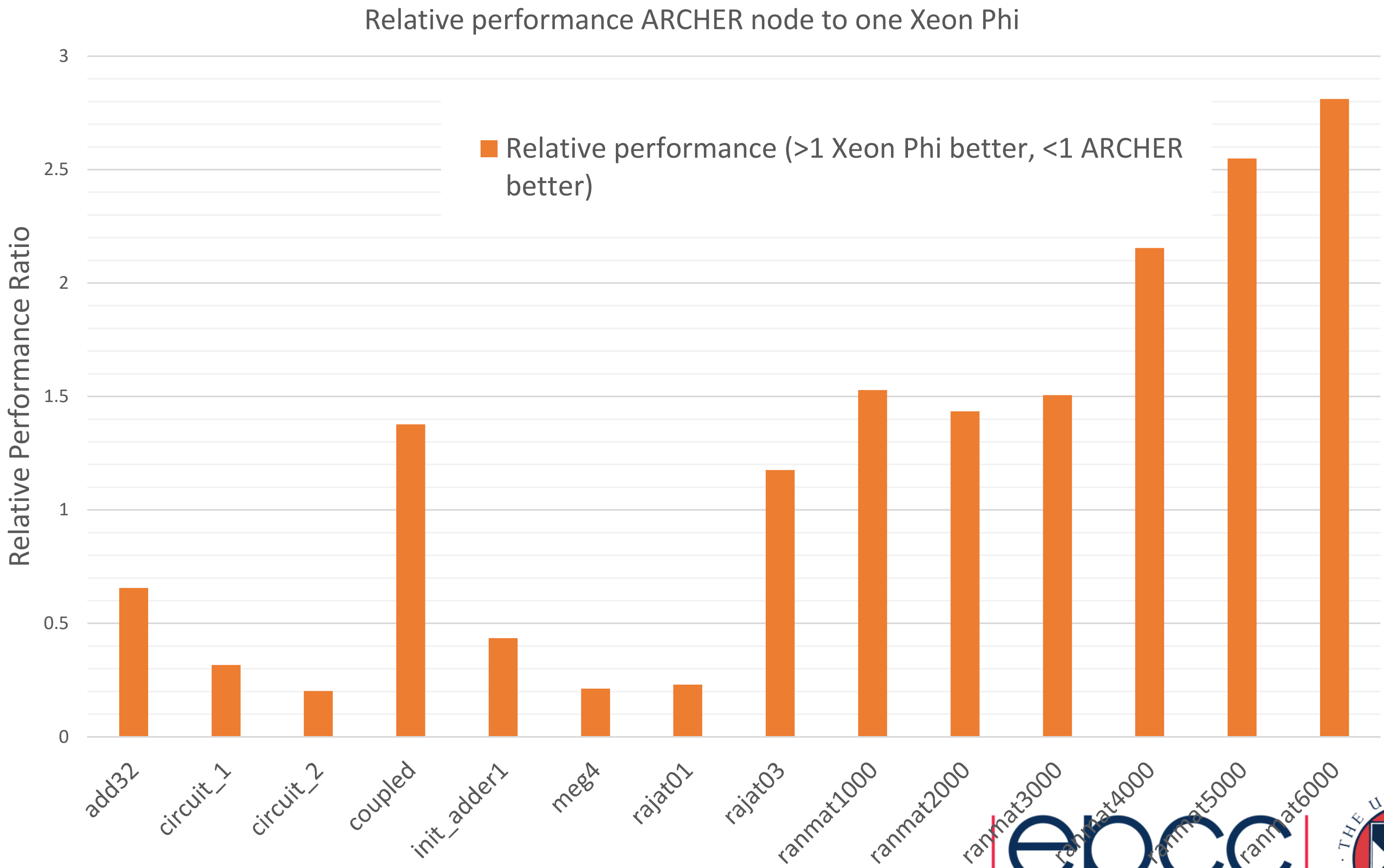
Results courtesy of Fiona Reid

CP2K

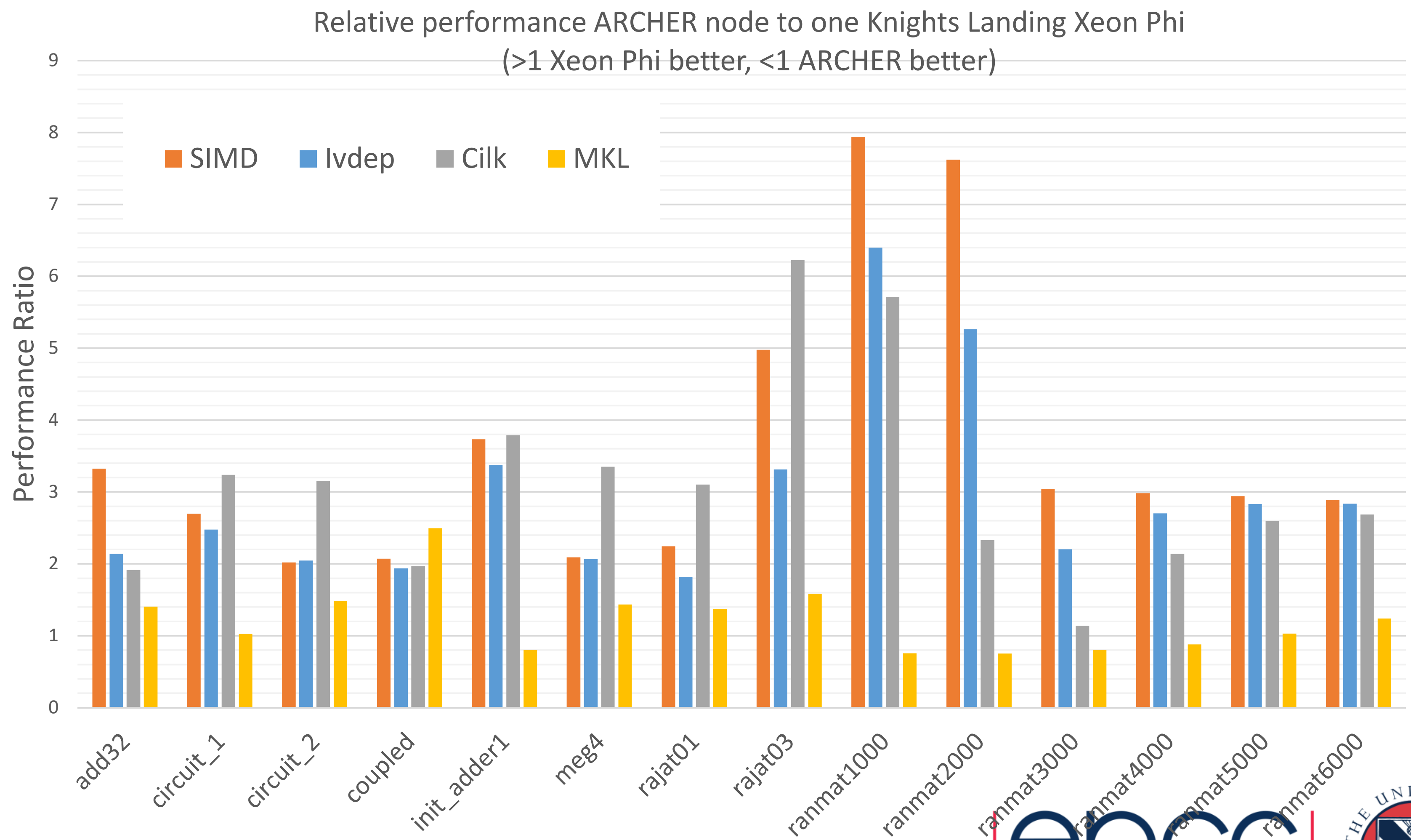


Results courtesy of Fiona Reid

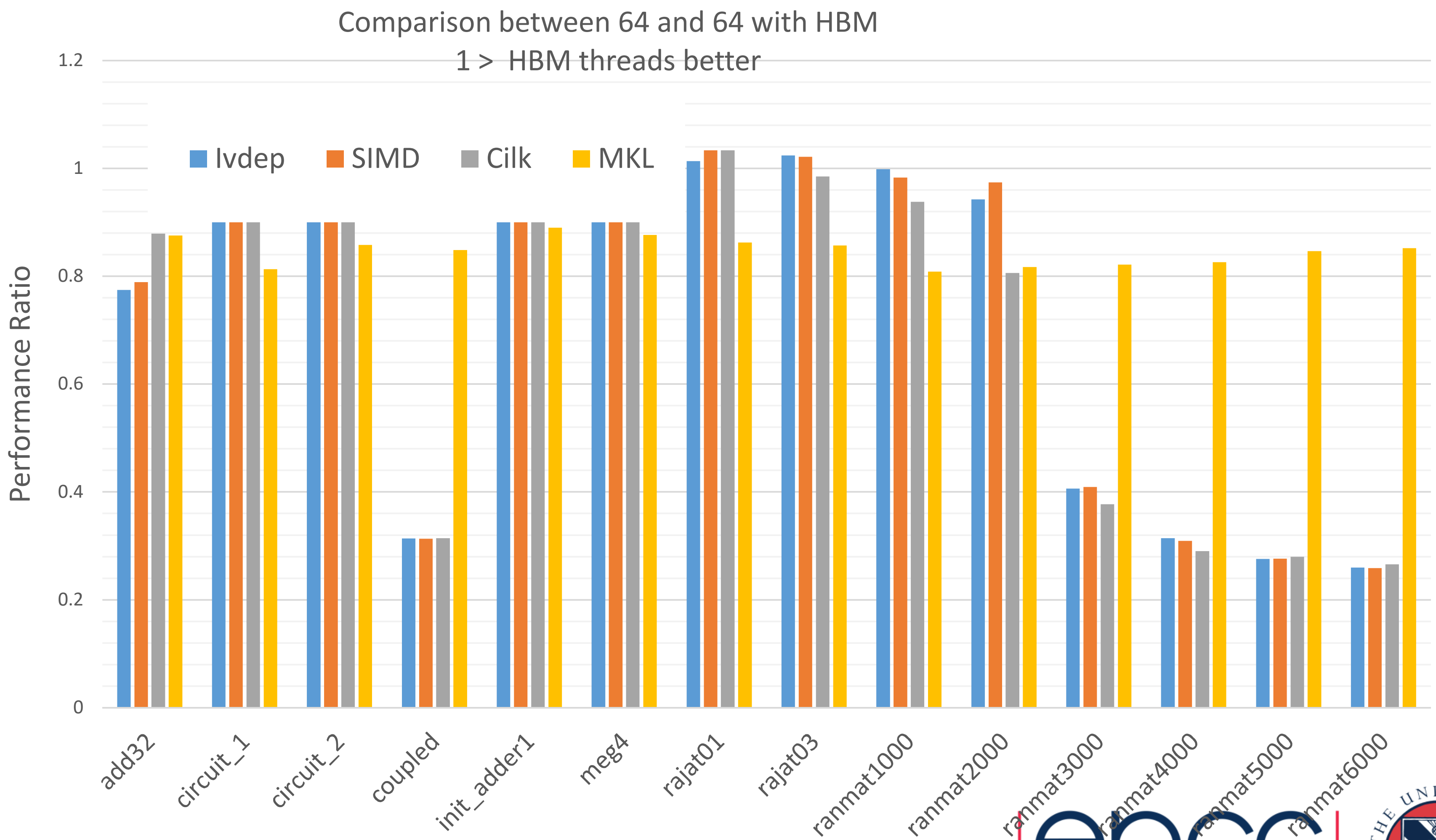
LU factorisation (KNC)



LU Factorisation



LU factorisation



Performance multi-node

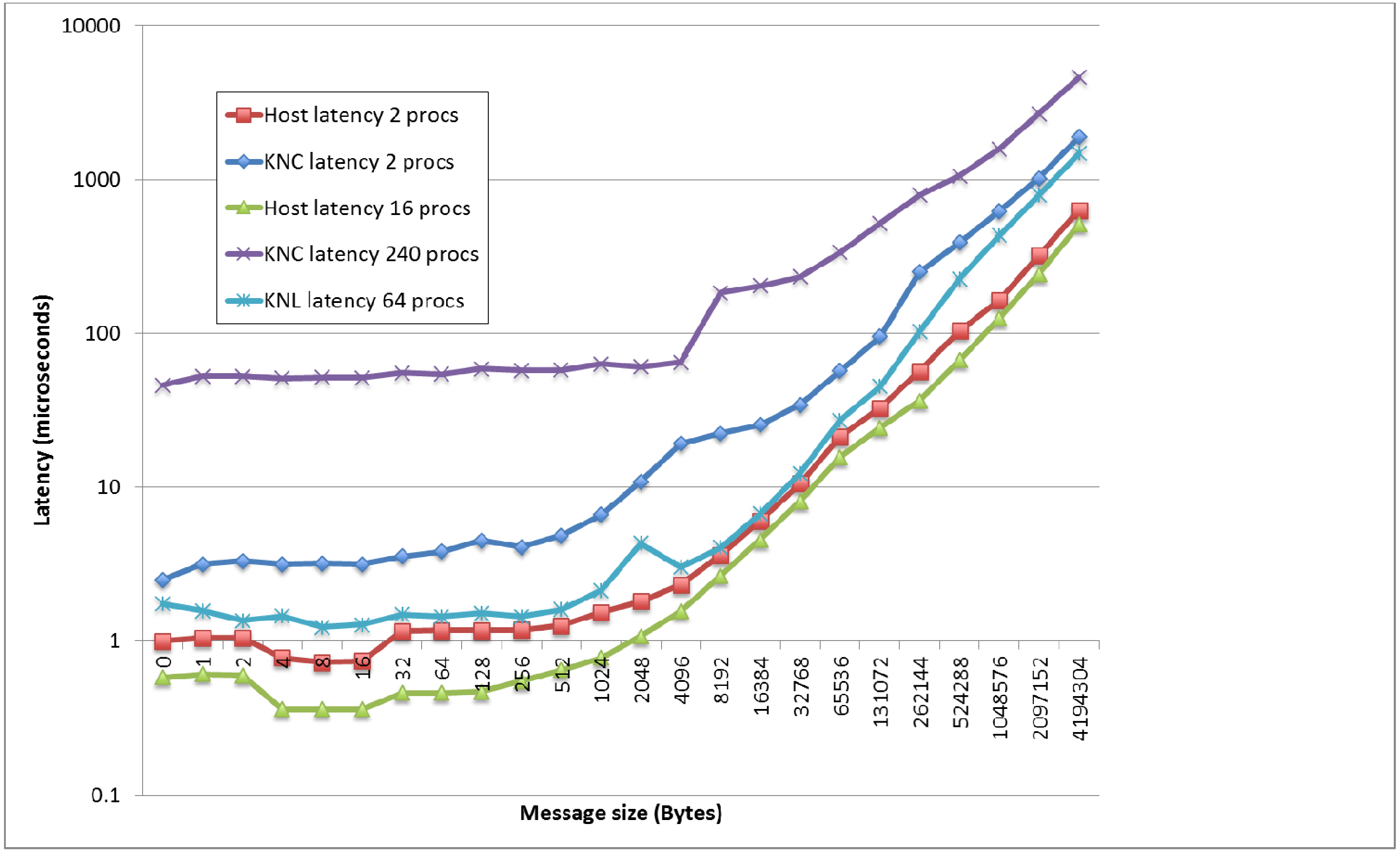
- COSA - Fluid dynamics code
  - Harmonic balance (frequency domain approach)
  - Unsteady navier-stokes solver
  - Optimise performance of turbo-machinery like problems
  - Multi-grid, multi-level, multi-block code

ARCHER	Broadwell	KNL	KNL HB	KNL 2 Node HB
497	349	561	450	197

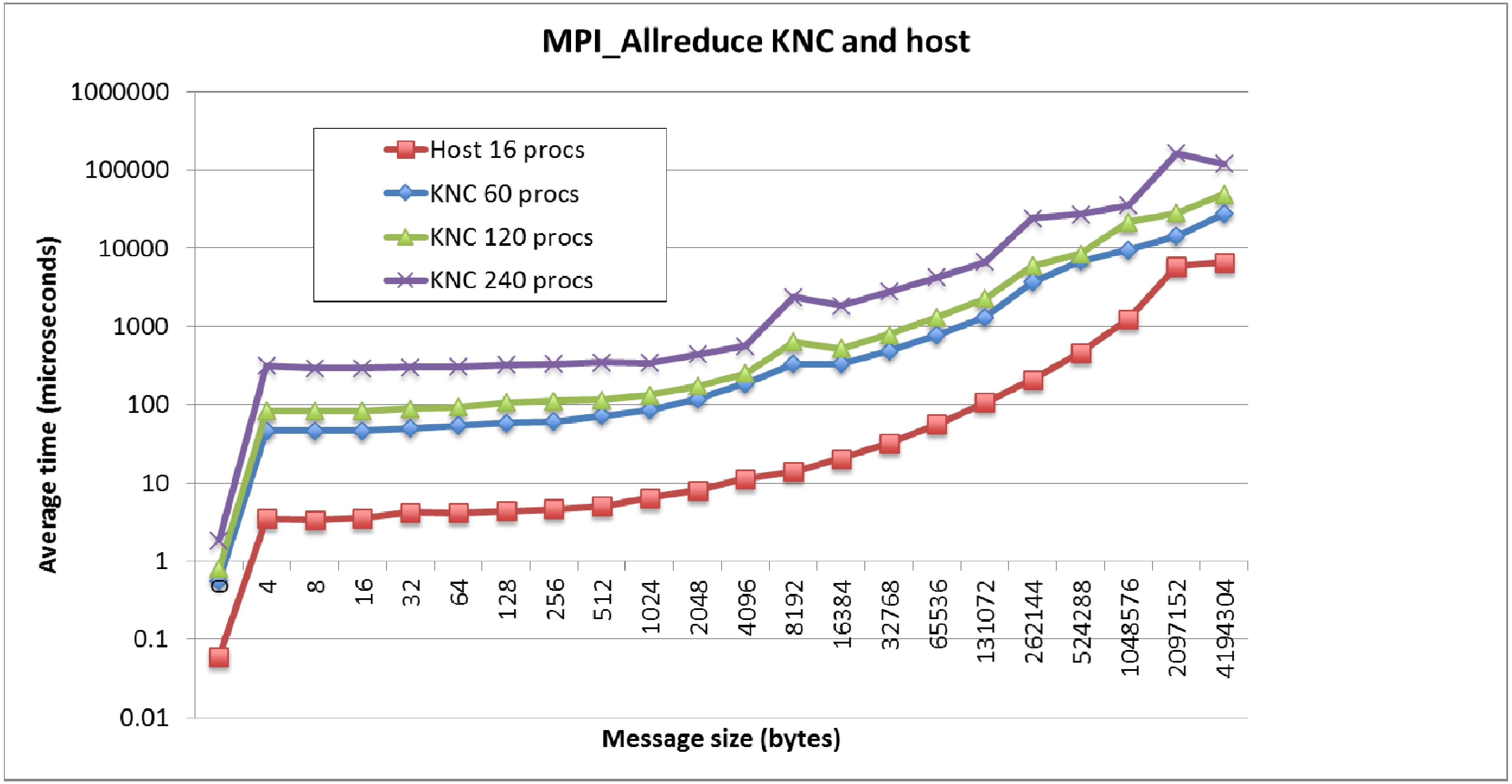
- GS2

ARCHER	Broadwell	KNL	KNL HB	KNL 2 Node	KNL 2 Node HB
126	84	185	103	103	70

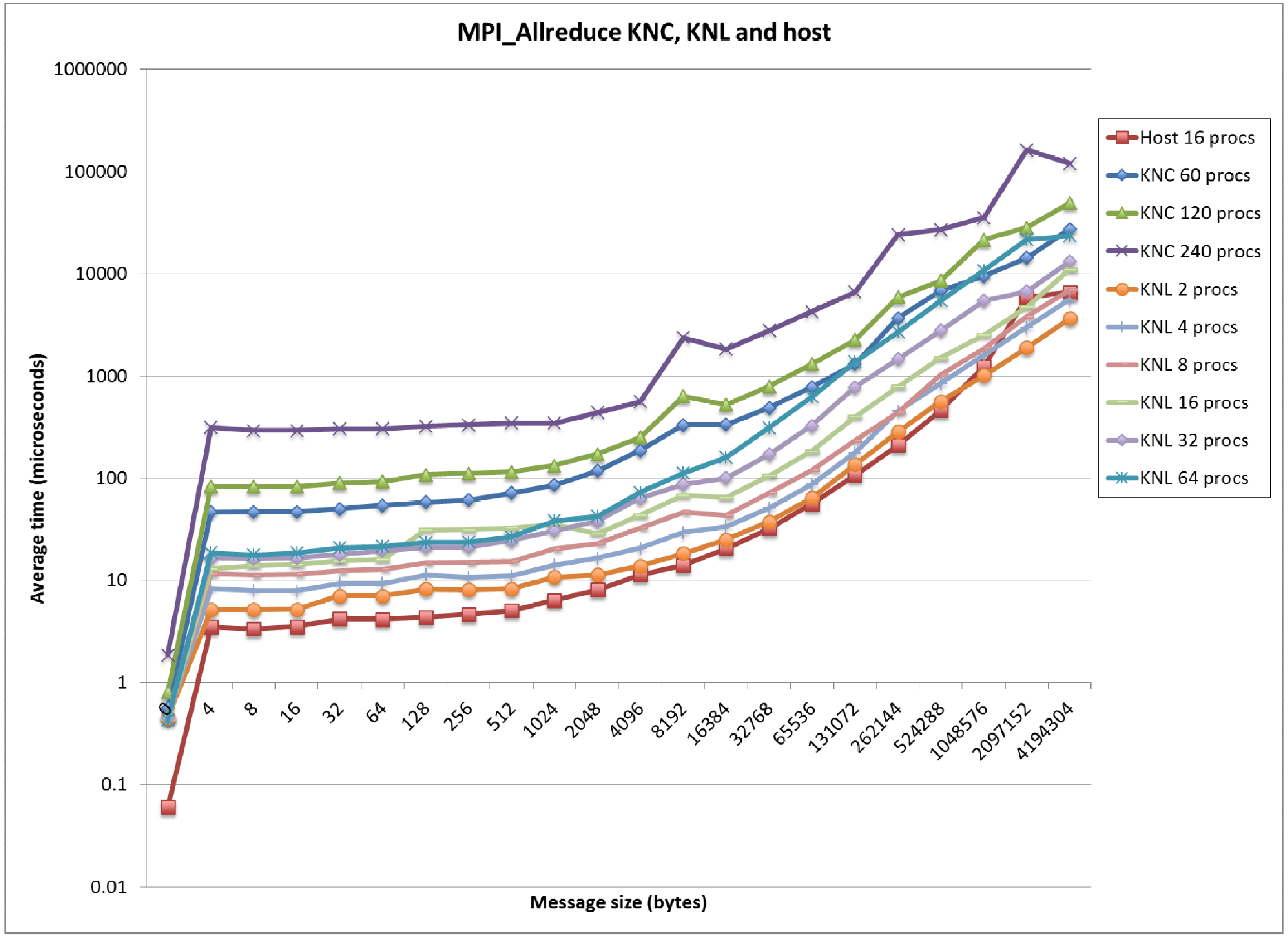
MPI Performance - PingPong



MPI Performance - Allreduce



MPI Performance - Allreduce



Questions?

mwuJ1RxYW8T8