Welcome

Virtual tutorial starts at 15.00 BST

archer

epcc

THE UNIVERSITY OF EDINBURGH

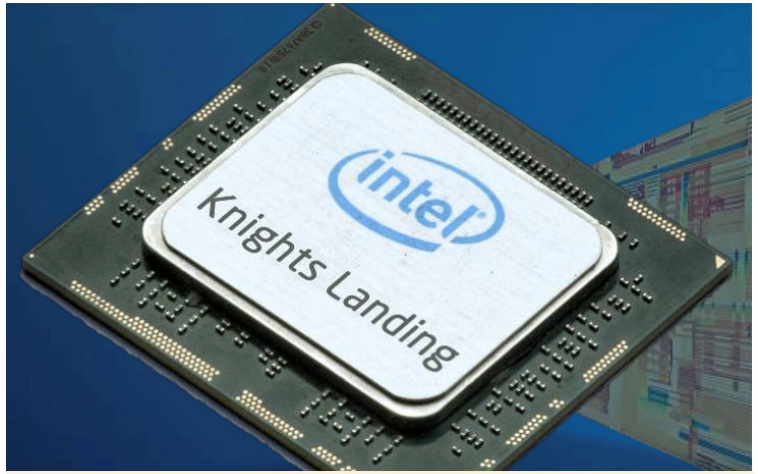
Introduction to KNL

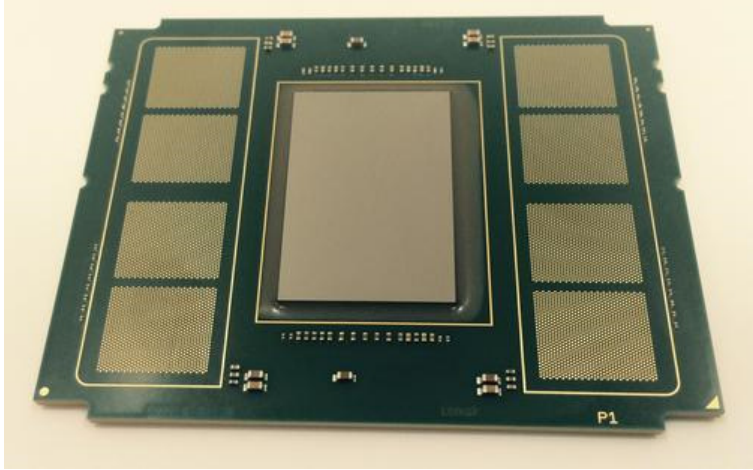
Adrian Jackson
adrianj@epcc.ed.ac.uk
@adrianjhpc

Many slides from Intel presentations

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





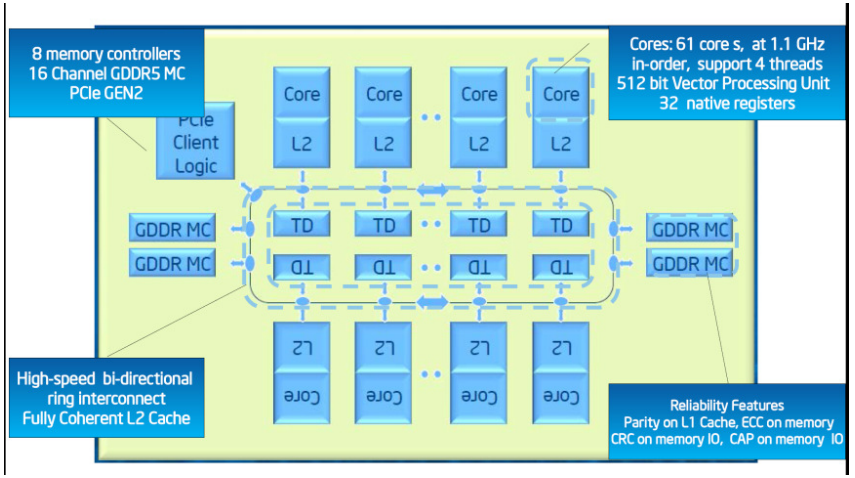
archer

epcc

THE UNIVERSITY OF EDINBURGH

KNC – 1st Gen

- Knights Corner
  - Many Integrated Cores (MIC) architecture
  - HPC/computational simulation primary market
  - Co-processor
    - PCIe Card
  - 60 cores/240 threads/1.054 GHz
  - 8 GB memory/320 GB/s bandwidth
  - 512-bit SIMD instructions
- Hybrid between GPU and many-core CPU



archer

epcc

THE UNIVERSITY OF EDINBURGH

KNC – 1st Gen

	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

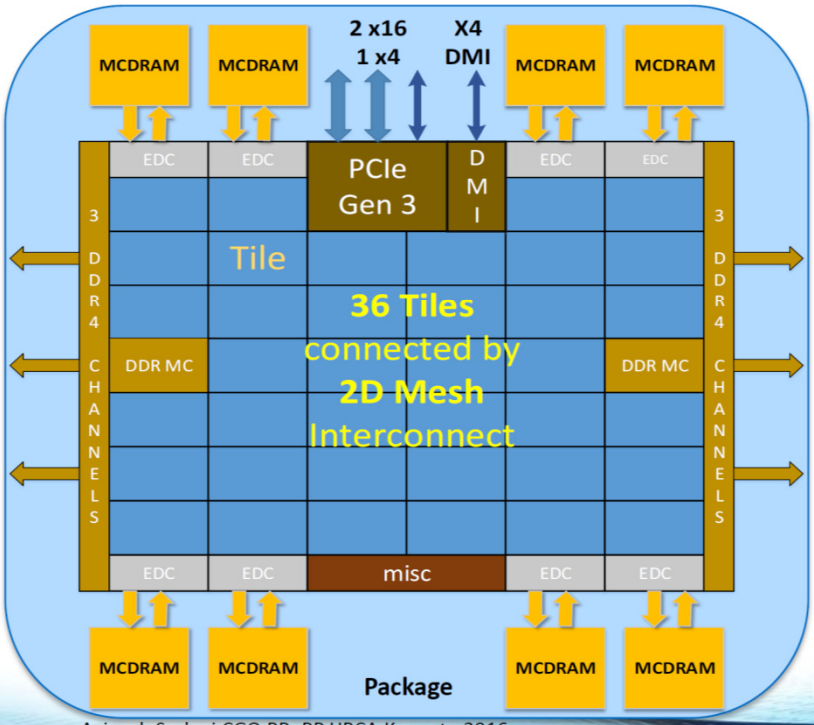
- Usable in different ways
  - Offload kernels
  - “Native” direct run applications

KNC: Achievable Performance

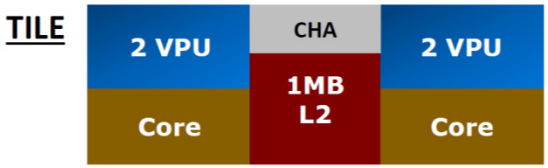
- 1 to 1.2 TFlop/s double precision performance
  - Dependent on using 512-bit vector units
  - And FMA instructions
- 240 to 352 GB/s peak memory bandwidth
- ~60 physical cores
  - Each can run 4 threads
  - Must run at least 2 threads to get full instruction issue rate
  - Don't think of it as 240 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

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

Note: not all specifications shown apply to all Knights Landing SKUs
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 [illegible] or software design or configuration may affect actual performance.

archer

epcc

THE UNIVERSITY OF EDINBURGH

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

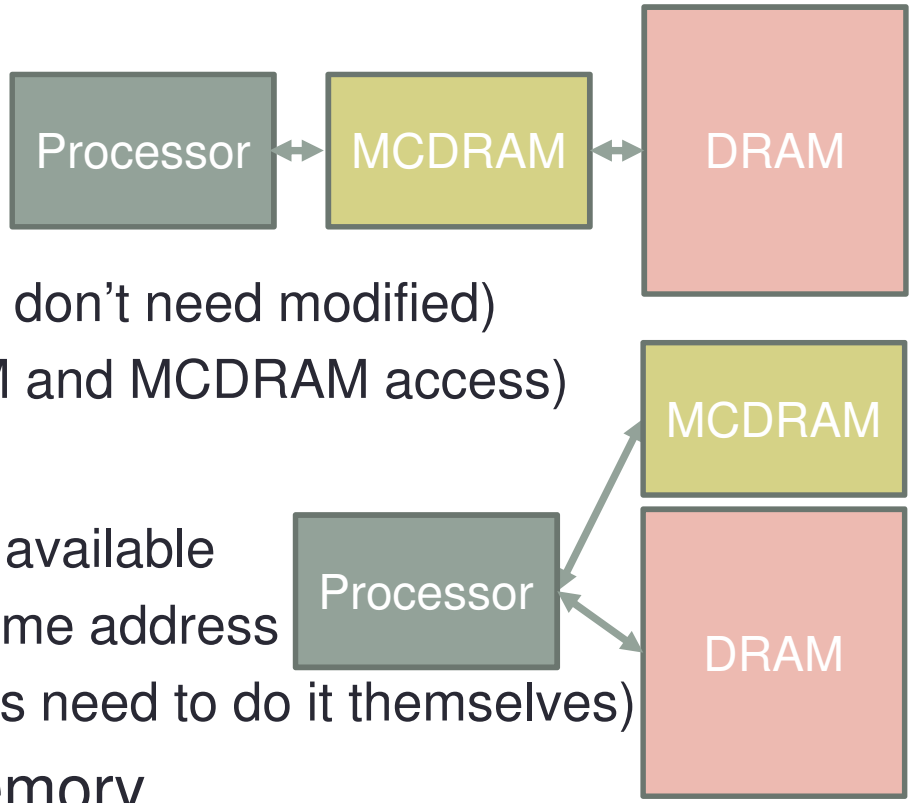
Picture from Avinash Sodani's talk from hot chips 2016

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 channels

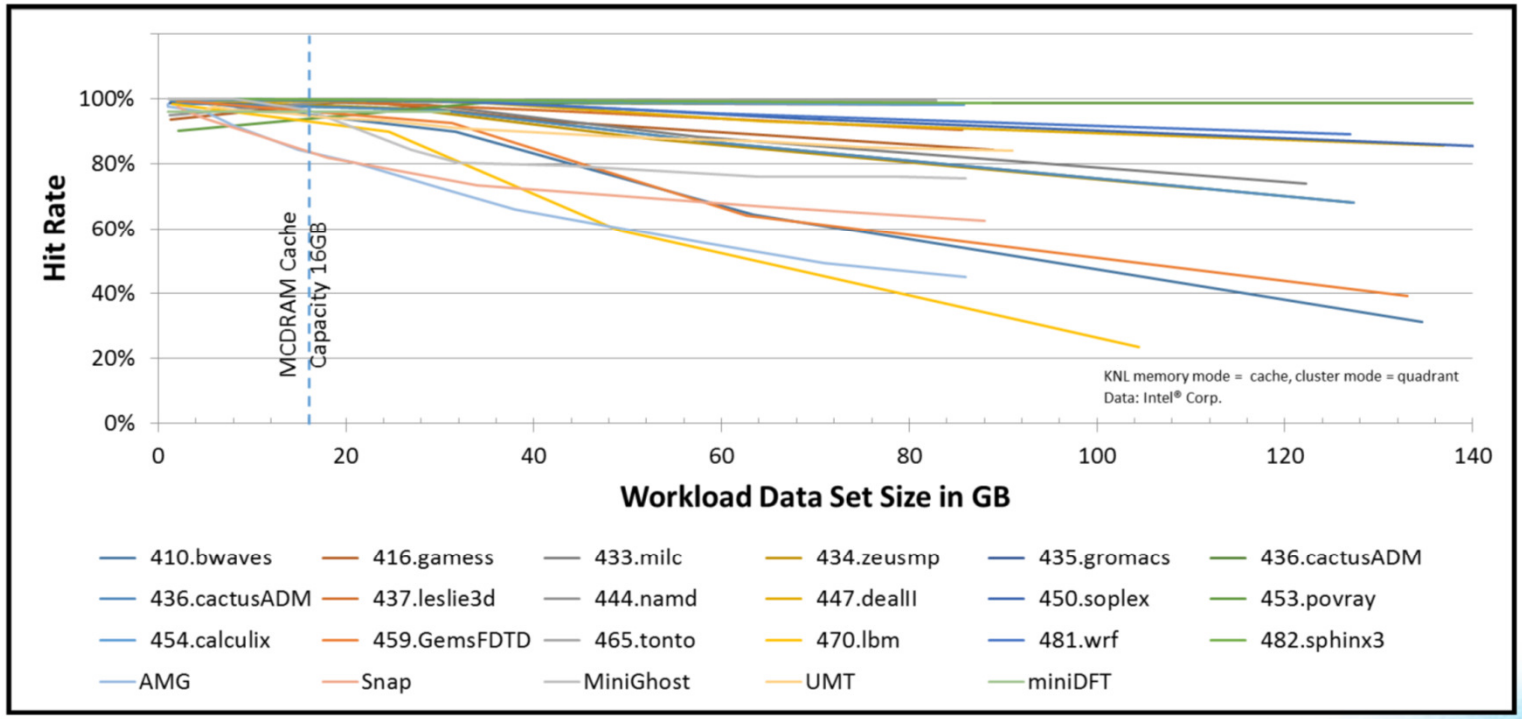
Memory Modes

- 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



Cache mode

MCDRAM Cache Hit Rate



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you purchases, including the performance of that product when combined with other products. KNL results measured on pre-production parts. Any difference in system hardware or software design or configuration may affect actual performance. For more information go to http://www.intel.com/performance *Other names and brands may be claimed as the property of others

MCDRAM performs well as cache for many workloads
Enables good out-of-box performance without memory tuning

Slide from Intel

Using flat mode

- Set bulk memory policy
  - Preferred or enforced memory for application
  - MCDRAM exposed as NUMA node
  - Use `numactl` program
- Example code:
  - Check available memory

```
[Xajacks@eln4 Mg2SiO4-geom]$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22
node 0 size: 49090 MB
node 0 free: 32586 MB
node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23
node 1 size: 49152 MB
node 1 free: 28820 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

  - Fails if exhausts memory

```
mpirun -n 64  numactl -m 1 ./castep.mpi forsterite
```

  - Tries to used preferred memory, falls back if exhausts memory

```
mpirun -n 64  numactl -p 1 ./castep.mpi forsterite
```

Allocating MCDRAM

A Heterogeneous Memory Management Framework

MEMKIND

- Defines a plug-in architecture.
- Each plug-in is called a “kind” of memory.
- Built on top of jemalloc: the FreeBSD OS default heap manager.
- Partition is defined by functions that provide inputs for operating system calls.
- High level memory management functions can be over-ridden as well.
- https://github.com/memkind

HBWMALLOC

- Implements easy model for KNL.
- Implemented using memkind; simplifies plug-in (kind) selection.
- Provides support for 2MB and 1GB pages.
- Select fallback behavior when on package memory does not exist or is exhausted.
- Check for existence of on package memory.

Jeff Hammond
Intel Parallel Computing Lab

Allocating MCDRAM

End Goal Usage: Code Snippets

Heap allocation in C

```
float * fv1 = malloc(sizeof(float) * 1000);
float * fv2 = hbw_malloc(sizeof(float) * 1000);
```

Allocatable arrays in Fortran

```
    REAL, ALLOCATABLE :: A(:), B(:), C(:)
!DIR$ ATTRIBUTES FASTMEM :: A
    NSIZE=1000
!   allocate array 'A' from MCDRAM
    ALLOCATE (A(1:NSIZE))
!   Allocate arrays that will come from DDR
    ALLOCATE  (B(1:NSIZE), C(1:NSIZE))
```

Automatic variables will be allocated in DDR in flat mode.

This means you may need to convert from automatic to heap arrays or use hybrid mode if such data is used in a bandwidth-intensive way.

Standard containers in C++ (not documented upstream yet)

```
std::vector<float, hbwmalloc::hbwmalloc_allocator<float> > vec;
```

Jeff Hammond
Intel Parallel Computing Lab

- Fortran:
  - FASTMEM is Intel directive
- Wrapped hbw_malloc
  - Call malloc directly in Fortran
  - https://github.com/jeffhammond/myhbwmalloc

```
use fortran_hbwmalloc
include 'mpif.h'
integer offset_kind
parameter(offset_kind=MPI_OFFSET_KIND)
integer(kind=offset_kind) ptr
INTEGER(C_SIZE_T) param
type(C_PTR) localptr
      real (kind=8) r8
      pointer (pr8, r8)
if (type.eq.'r8') then
        param = 8*dim
        localptr = hbw_malloc(param)
      else if (type.eq.'i4') then
        param = 4*dim
        localptr = hbw_malloc(param)
      end if
ptr = transfer(localptr,ptr)
    if (type.eq.'r8') then
      call c_f_pointer(localptr, pr8)
      call zeroall(dim,r8)
end if
```

Emulating MCDRAM

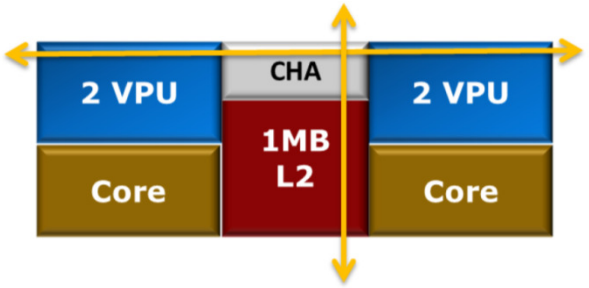
- Using multiprocessor node can emulate MCDRAM affect on application:
  - `export MEMKIND_HBW_NODES=0`
  - `mpirun -n 64  numactl -m 1 -N 0 ./my_application`
- Force all application memory to be allocated on the memory of the other processor
  - HBW memory will be allocated on the local memory
  - This will have lower latency, which isn't accurate
  - But will give an idea of the impact of higher bandwidth

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

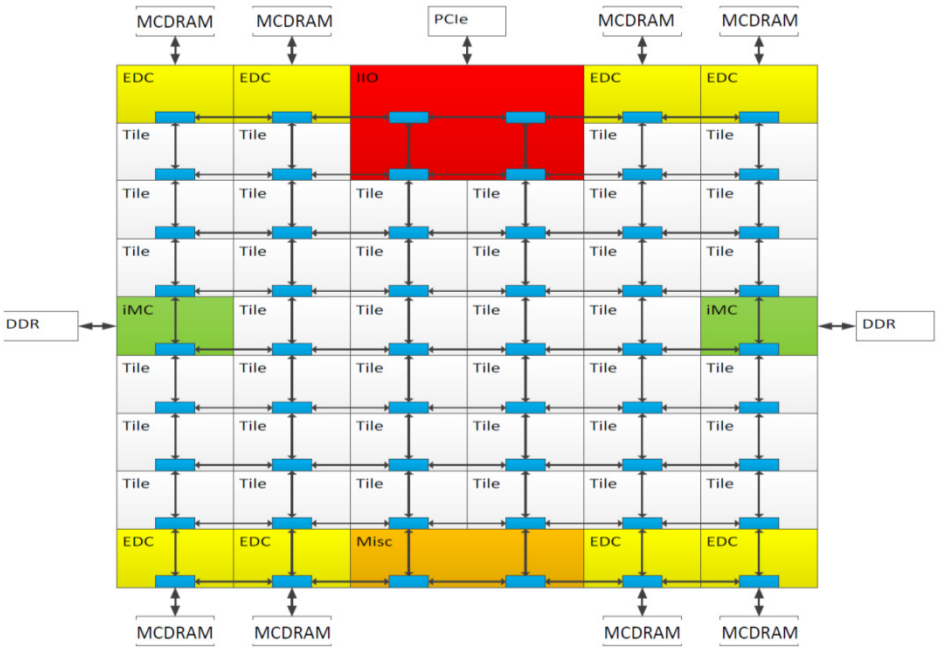
L2: 1MB 16-way. 1 Line Read and ½ Line Write per cycle. Coherent across all Tiles

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

archer
epcc
THE UNIVERSITY OF EDINBURGH

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

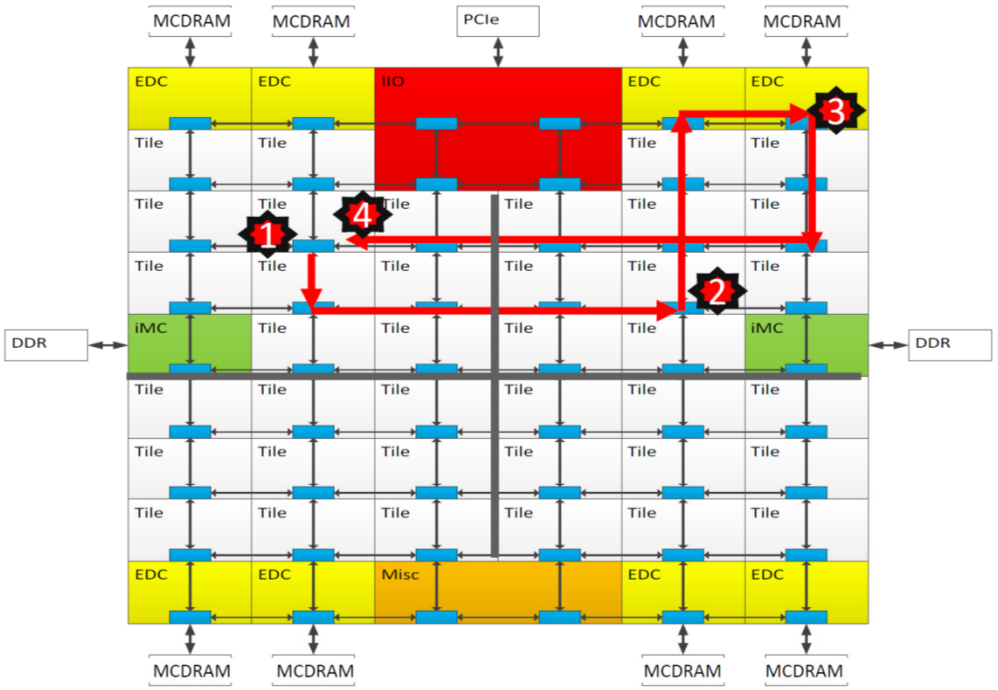
(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering

archer

epcc

THE UNIVERSITY OF EDINBURGH

KNL
Cluster Mode: Quadrant



Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. SW Transparent.

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Avinash Sodani CGO PPoPP HPCA Keynote 2016

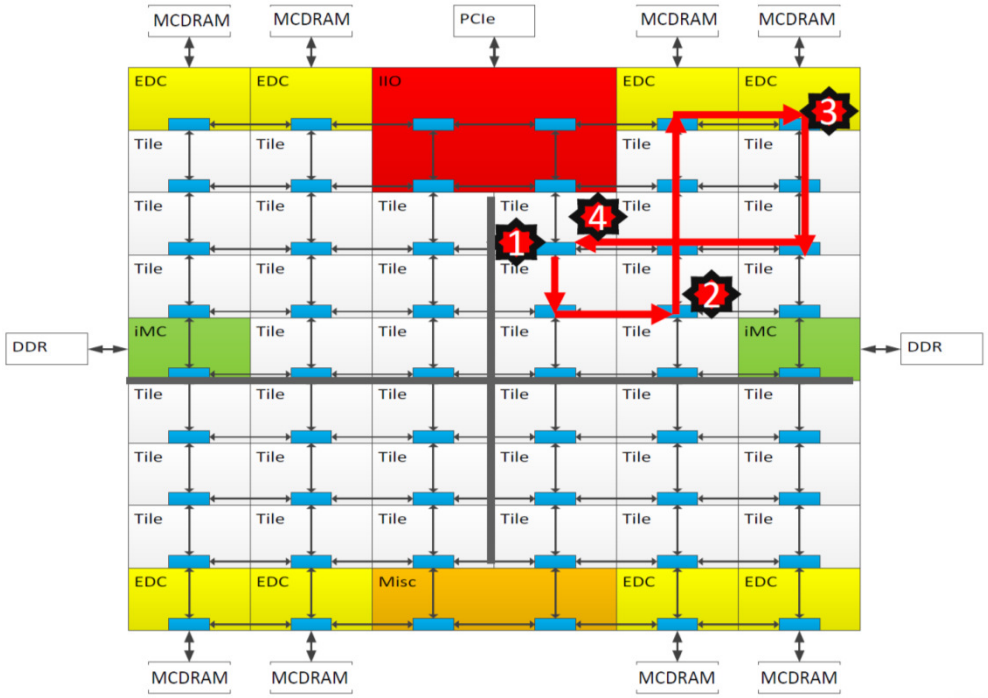
archer

epcc

THE UNIVERSITY OF EDINBURGH

KNL

Cluster Mode: Sub-NUMA Clustering (SNC)



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Avinash Sodani CGO PPoPP HPCA Keynote 2016

Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

Looks analogous to 4-Socket Xeon

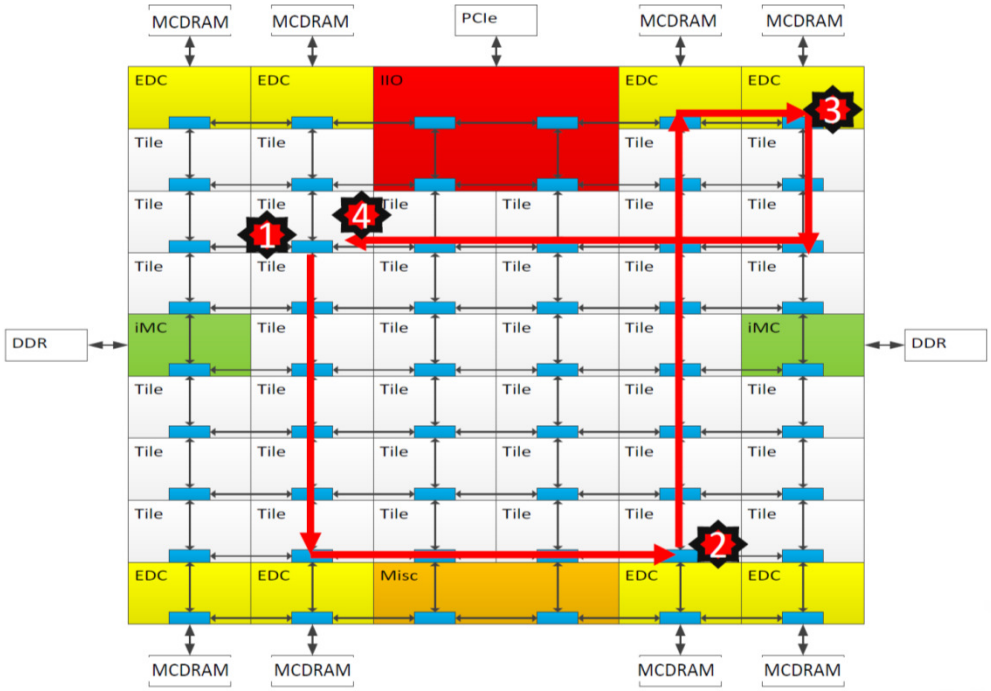
Affinity between Tile, Directory and Memory

Local communication. Lowest latency of all modes.

SW needs to NUMA optimize to get benefit.

KNL

Cluster Mode: All-to-All



Address uniformly hashed across all distributed directories

No affinity between Tile, Directory and Memory

Most general mode. Lower performance than other modes.

Typical Read L2 miss

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

Programming the KNL

- Standard HPC
  - MPI
  - OpenMP
  - mkl
  - Intel compilers
- Add:

```
-xMIC-AVX512
```

- Run with standard MPI job launcher

```
mpirun -n 128 -f $PBS_NODEFILE ./myapp
```

- If using only 1 MPI rank and OpenMP to fill up cores
  - If also using SNC have to enable all memory access

```
numactl -m 4,5,6,7
```

Multi-KNL run

Probably not what will be used on ARCHER Cray KNL

Configuring KNL

- Different memory modes and cluster options
  - Configured at boot time
    - Switching between cache and flat mode
    - Switching cluster modes

- More on ARCHER/Cray specific details in the third virtual seminar in this series
  - Wednesday 12th October 2016 15:00

Test data hardware

- 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

GS2 on KNL

- GS2 ported and run on KNL:
  - Small test cases: sweet spots: 1,2,4,8,16,32,176,352,….
  - ARCHER 2.10 minutes (24 cores) (7% imbalance)
  - KNL without fast mem 3.08 minutes (64 cores) (20% imbalance)
  - KNL with fast mem 1.74 minutes (64 cores)
  - KNL in cache mode 1.76 minutes (64 cores)
  - Broadwell 1.38 minutes (36 cores)

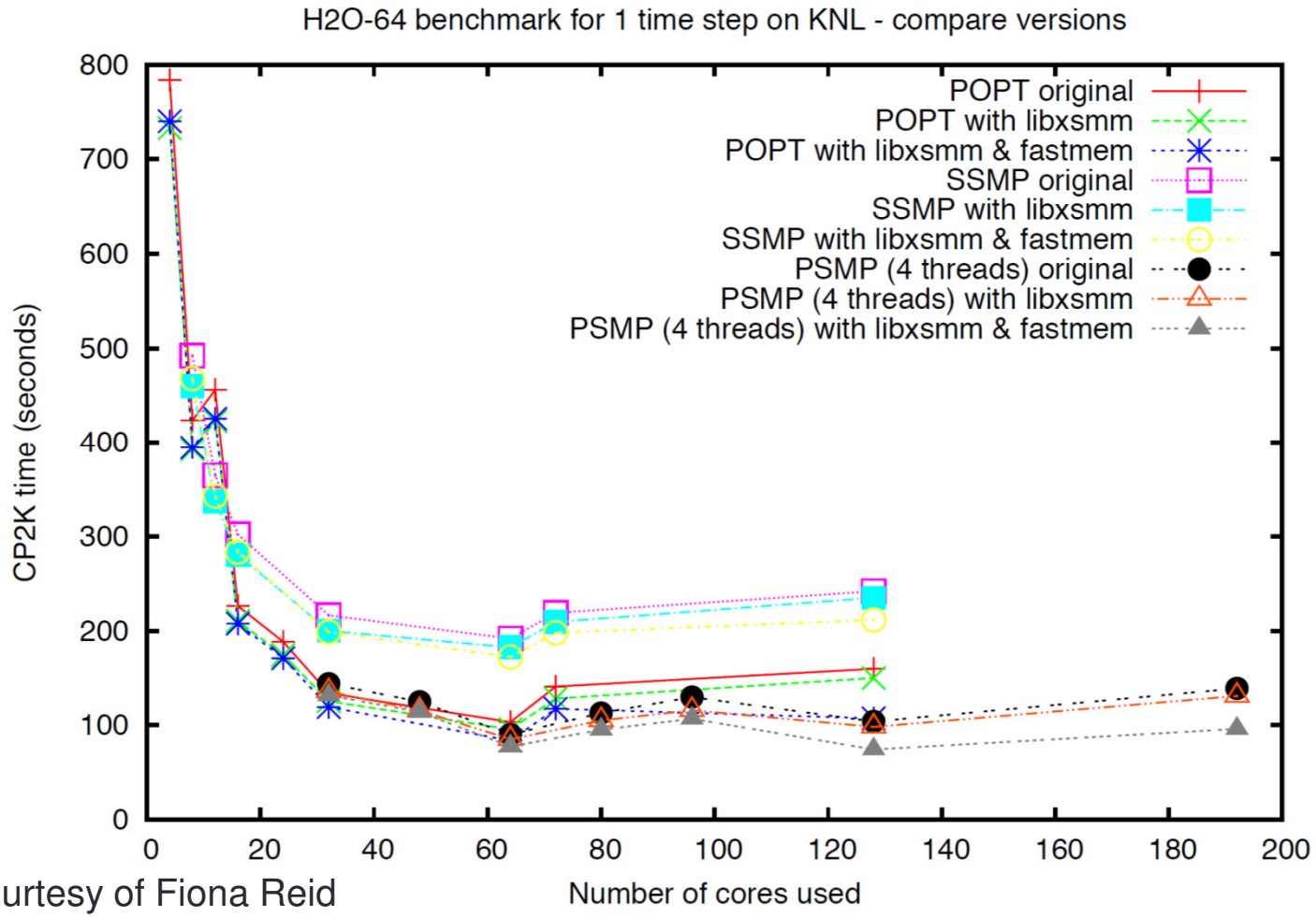
GS2 Port to KNC Xeon Phi

- Profiling of vectorisation of GS2 shows good performance
- Pure MPI code performance
  - ARCHER (2x12 core Xeon E5-2697, 16 MPI processes): 3.08 minutes
  - Host (2x8 core Xeon E5-2650, 16 MPI processes): 4.64 minutes
  - 1 Phi (176 MPI processes): 7.34 minutes
  - 1 Phi (235 MPI processes): 6.77 minutes
  - 2 Phi's (352 MPI processes): 47.71 minutes
- Hybrid code performance
  - 1 Phi (80 MPI processes, 3 threads each): 7.95 minutes
  - 1 Phi (120 MPI processes, 2 threads each): 7.07 minutes

CASTEP

- MgSiO4-Geom benchmark:
  - ARCHER: 24 cores
    - Total time = 102.27 s
  - KNL: 24 cores
    - Total time = 156.63 s
  - KNL: 64 cores
    - Total time = 149.65 s
  - KNL: 64 cores cache mode
    - Total time = 146.88 s
  - Broadwell: 36 cores
    - Total time = 38 s

CP2K



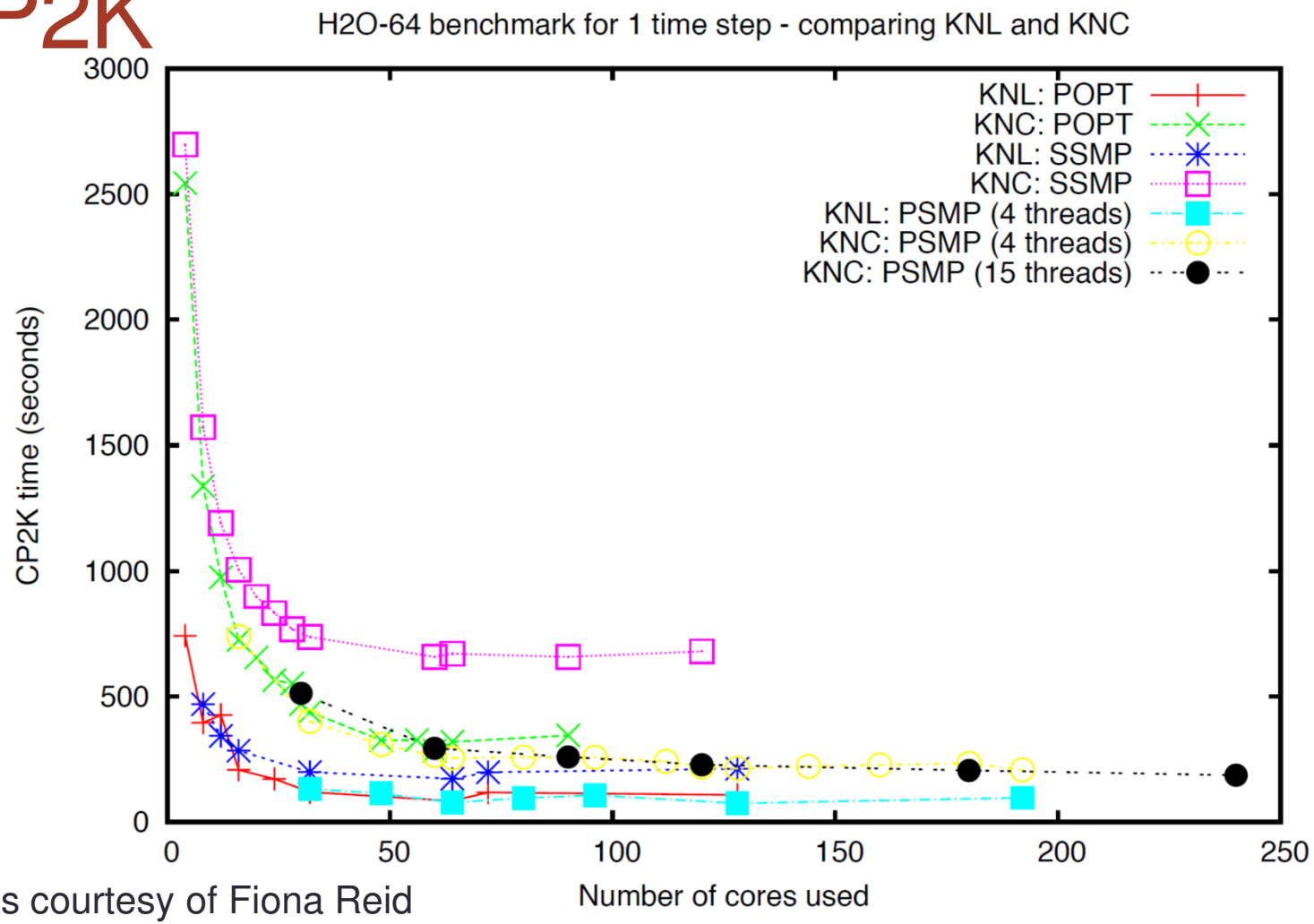
Results courtesy of Fiona Reid

archer

epcc

THE UNIVERSITY OF EDINBURGH

CP2K



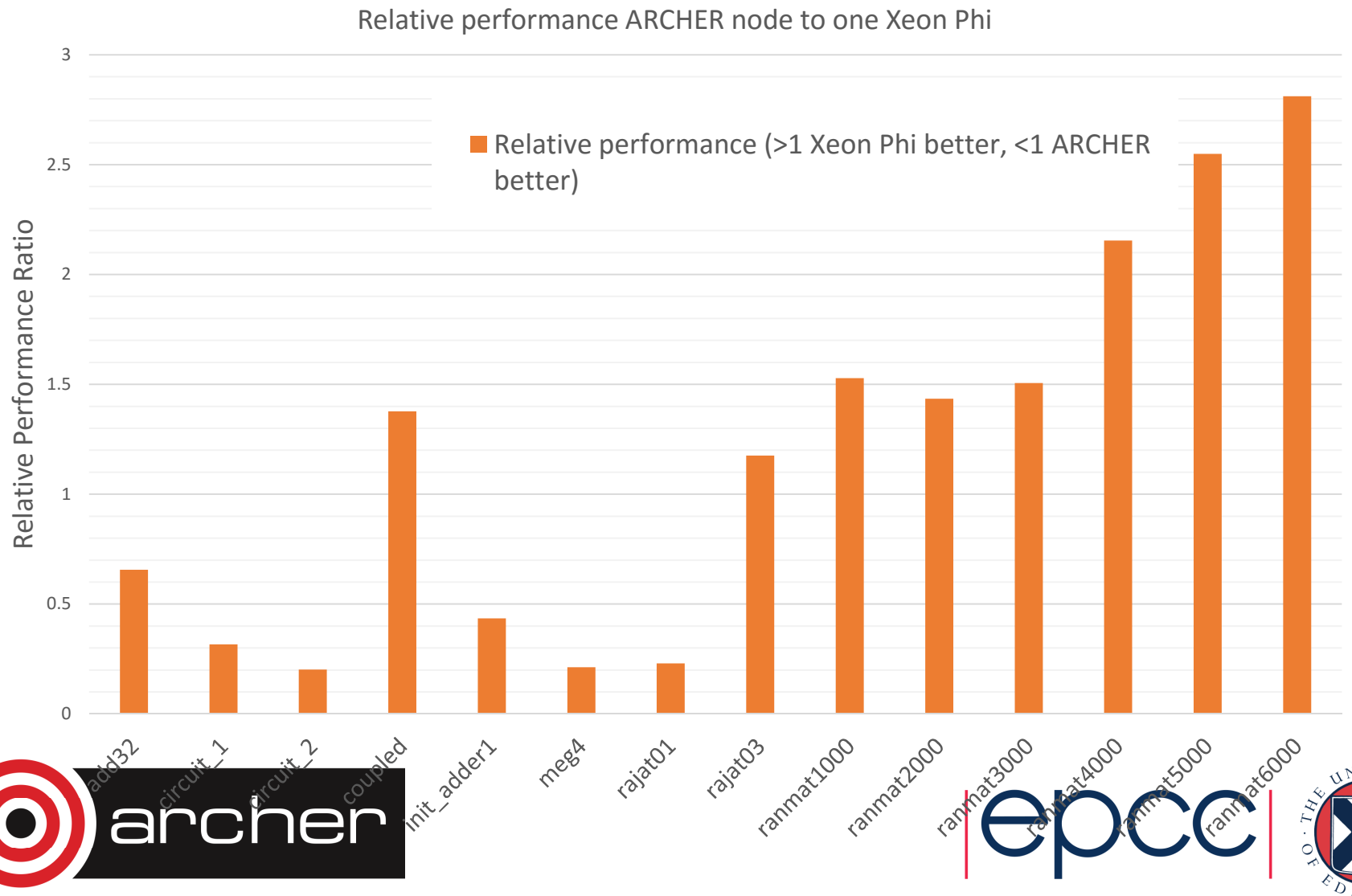
Results courtesy of Fiona Reid

archer

epcc

THE UNIVERSITY OF EDINBURGH

LU factorisation (KNC)

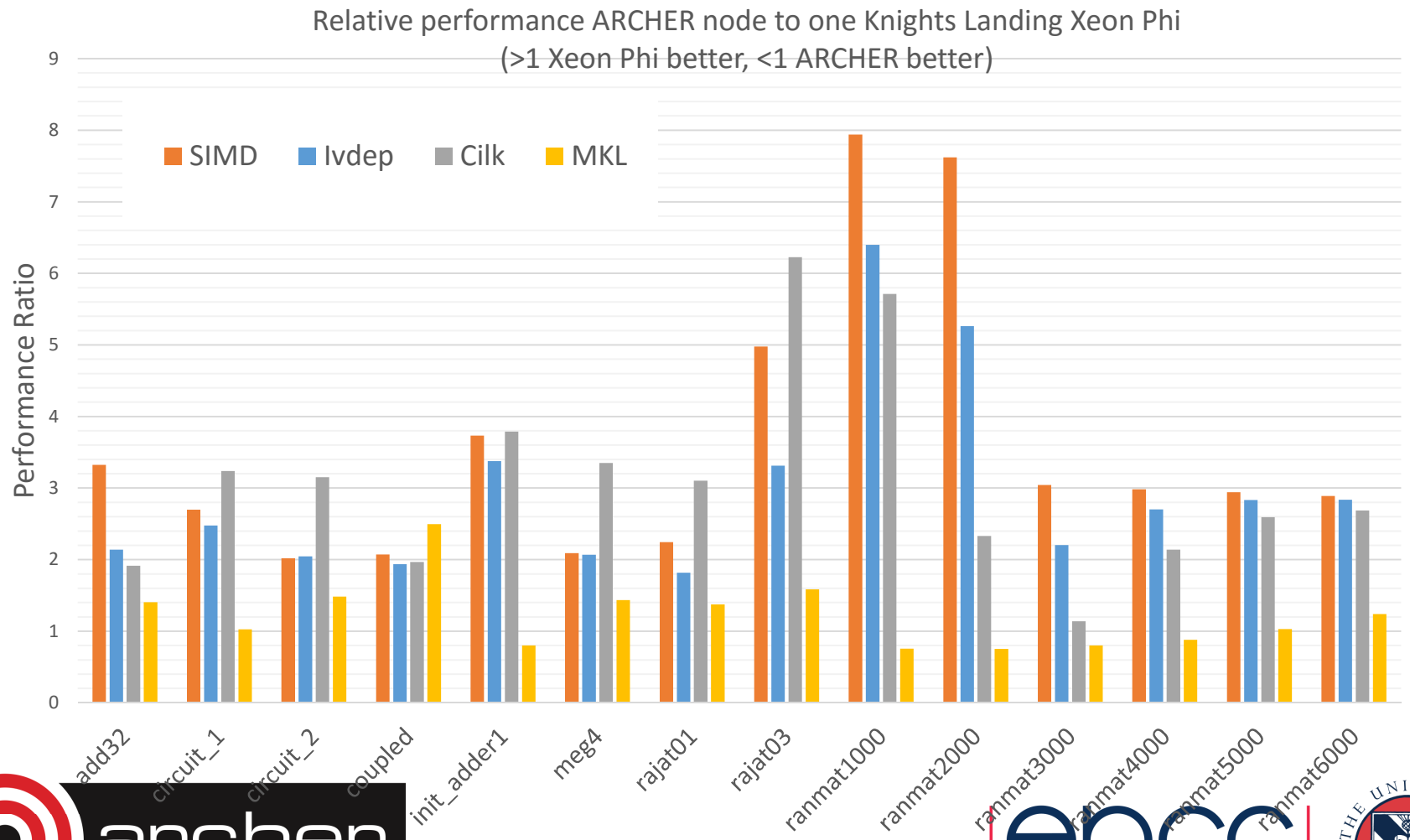


archer

epcc

THE UNIVERSITY OF EDINBURGH

LU Factorisation

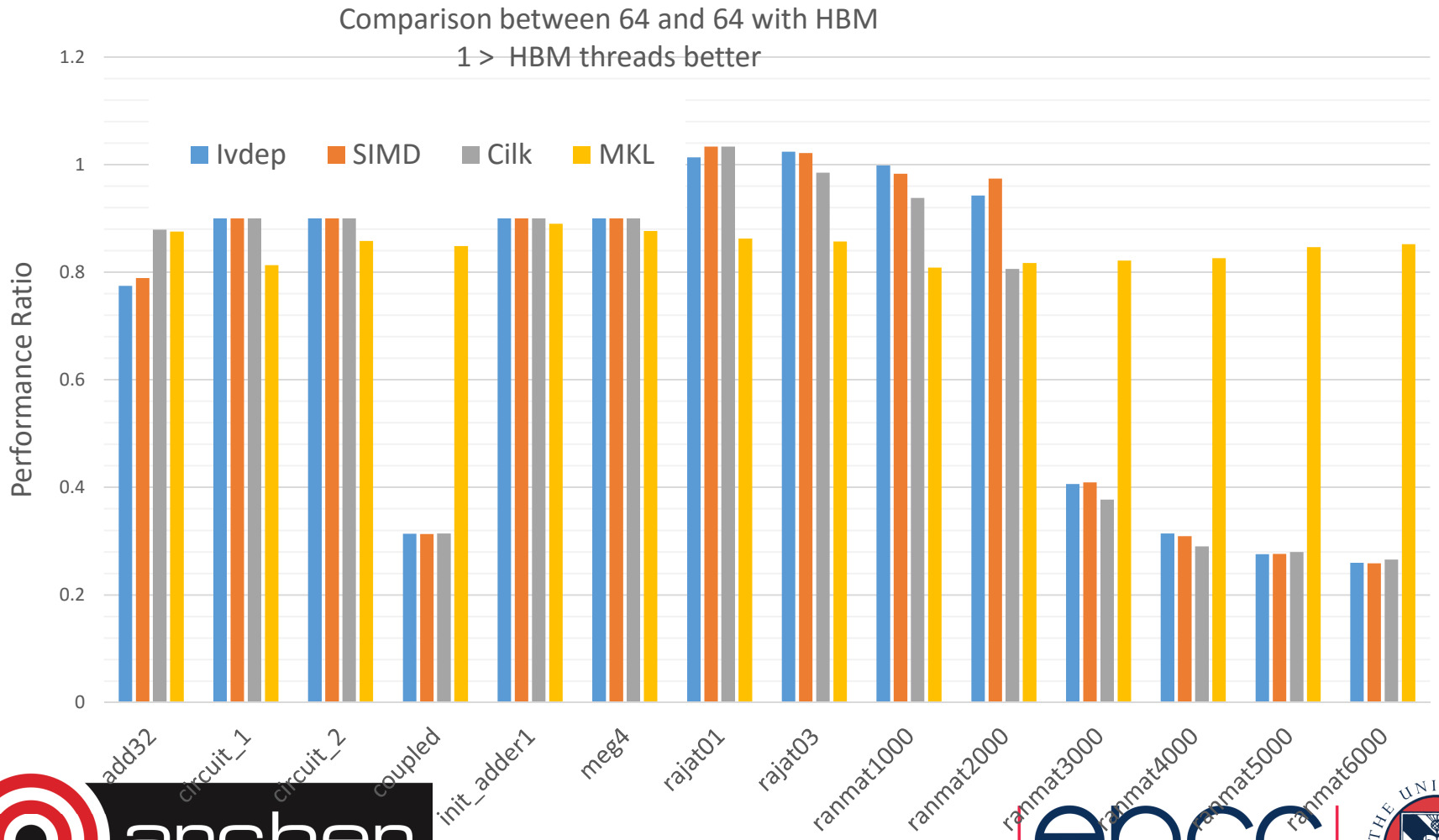


archer

epcc

THE UNIVERSITY OF EDINBURGH

LU factorisation

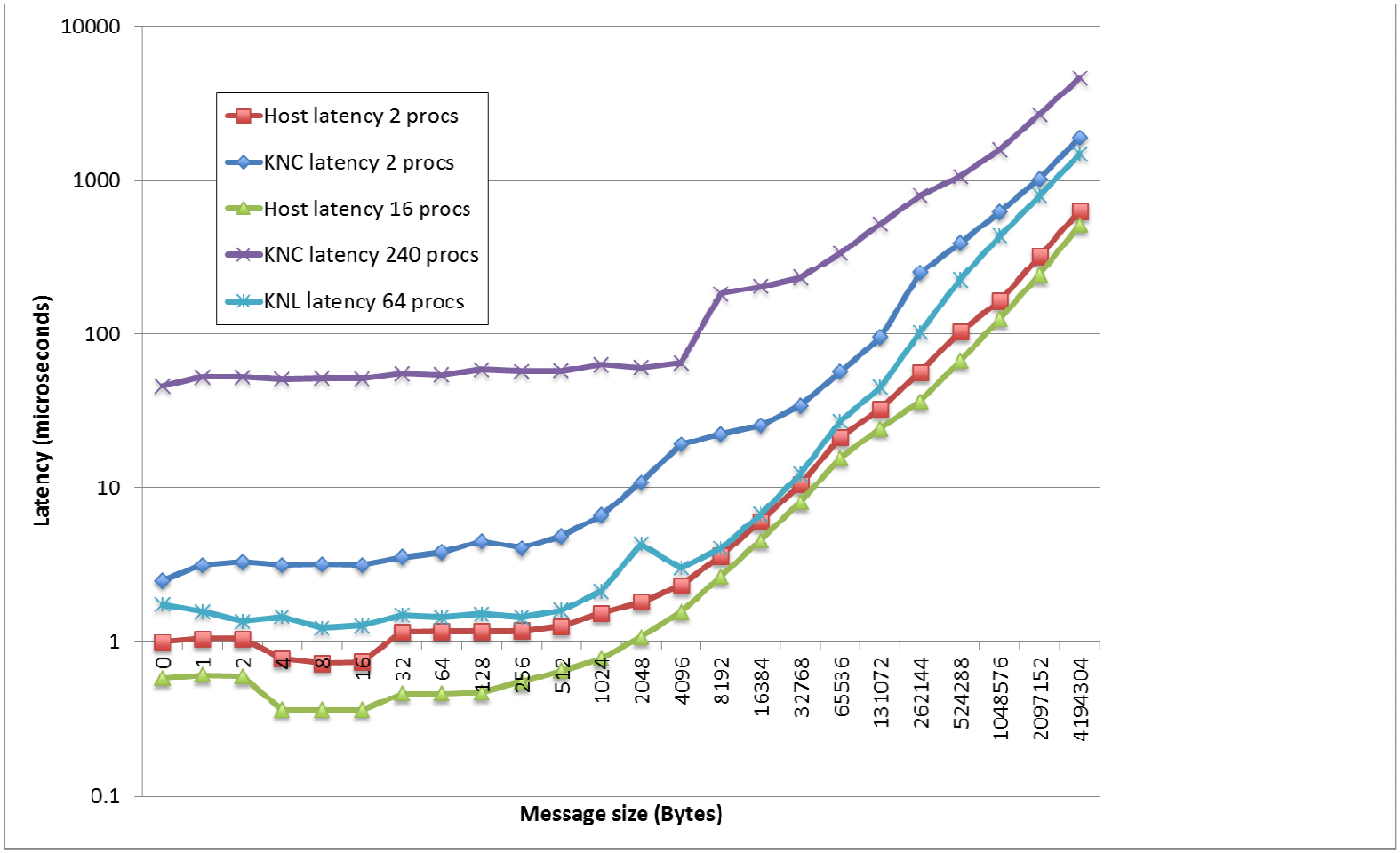


archer

epcc

THE UNIVERSITY OF EDINBURGH

MPI Performance - PingPong

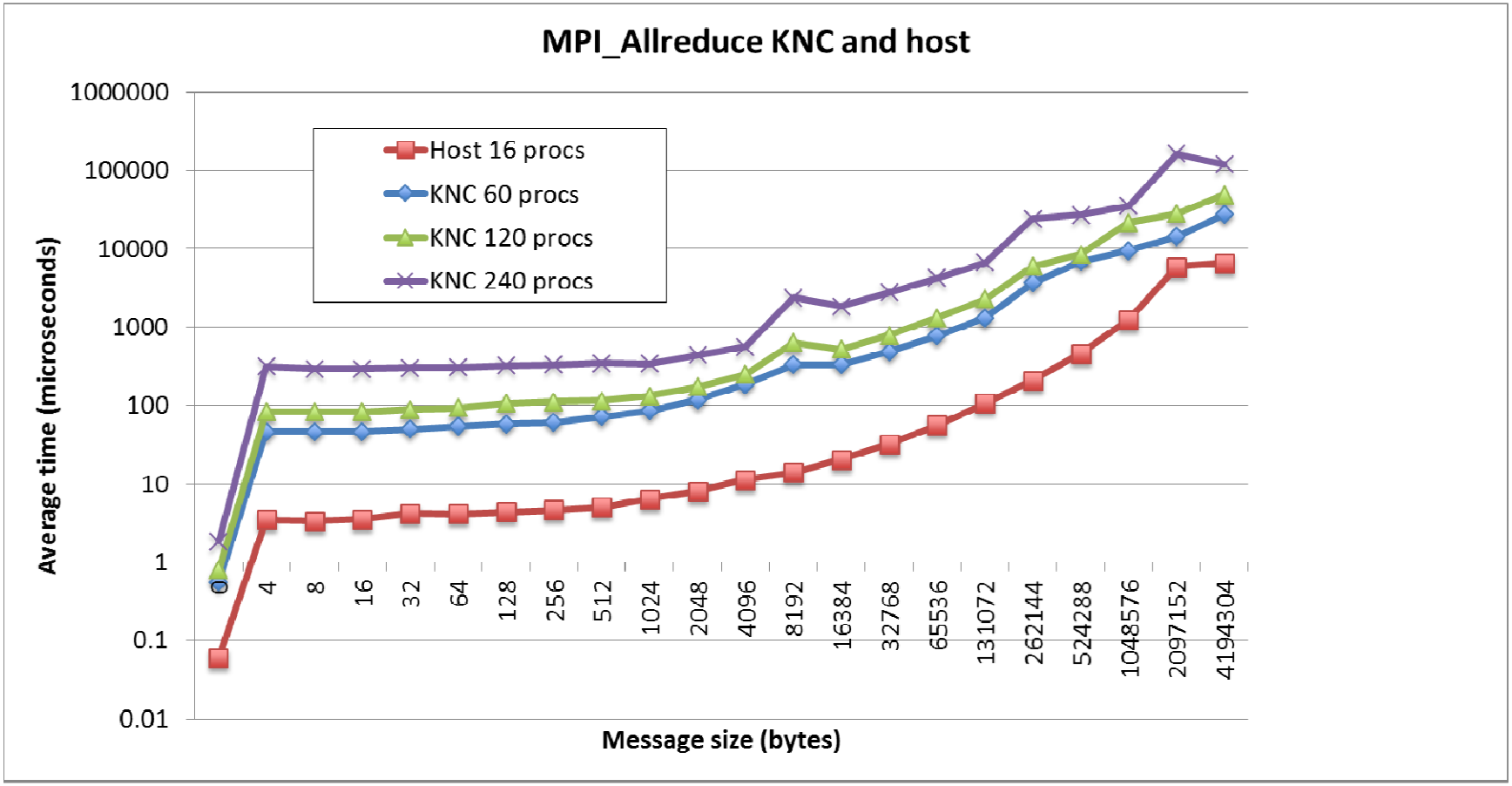


archer

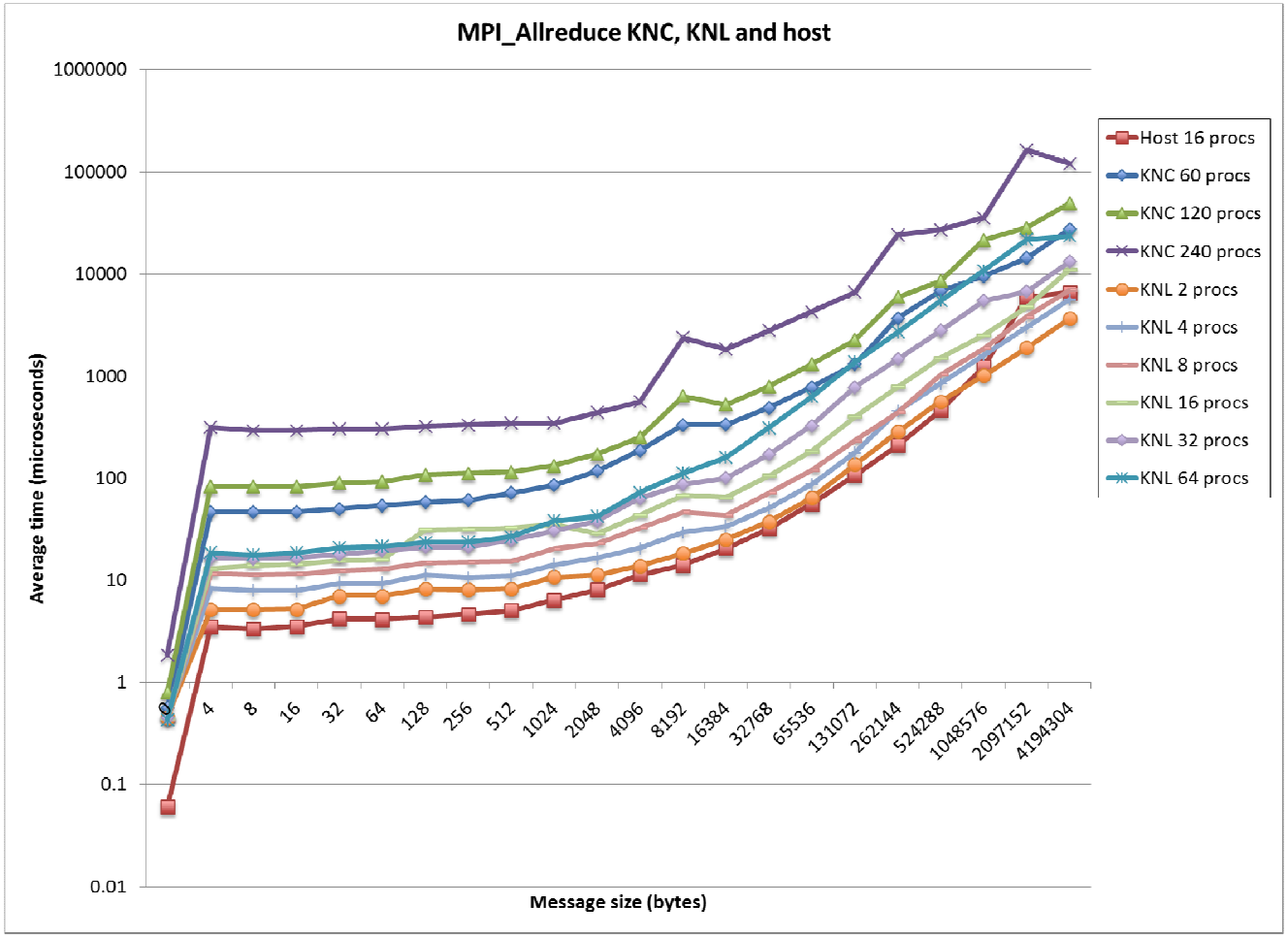
epcc

THE UNIVERSITY OF EDINBURGH

MPI Performance - Allreduce



MPI Performance - Allreduce

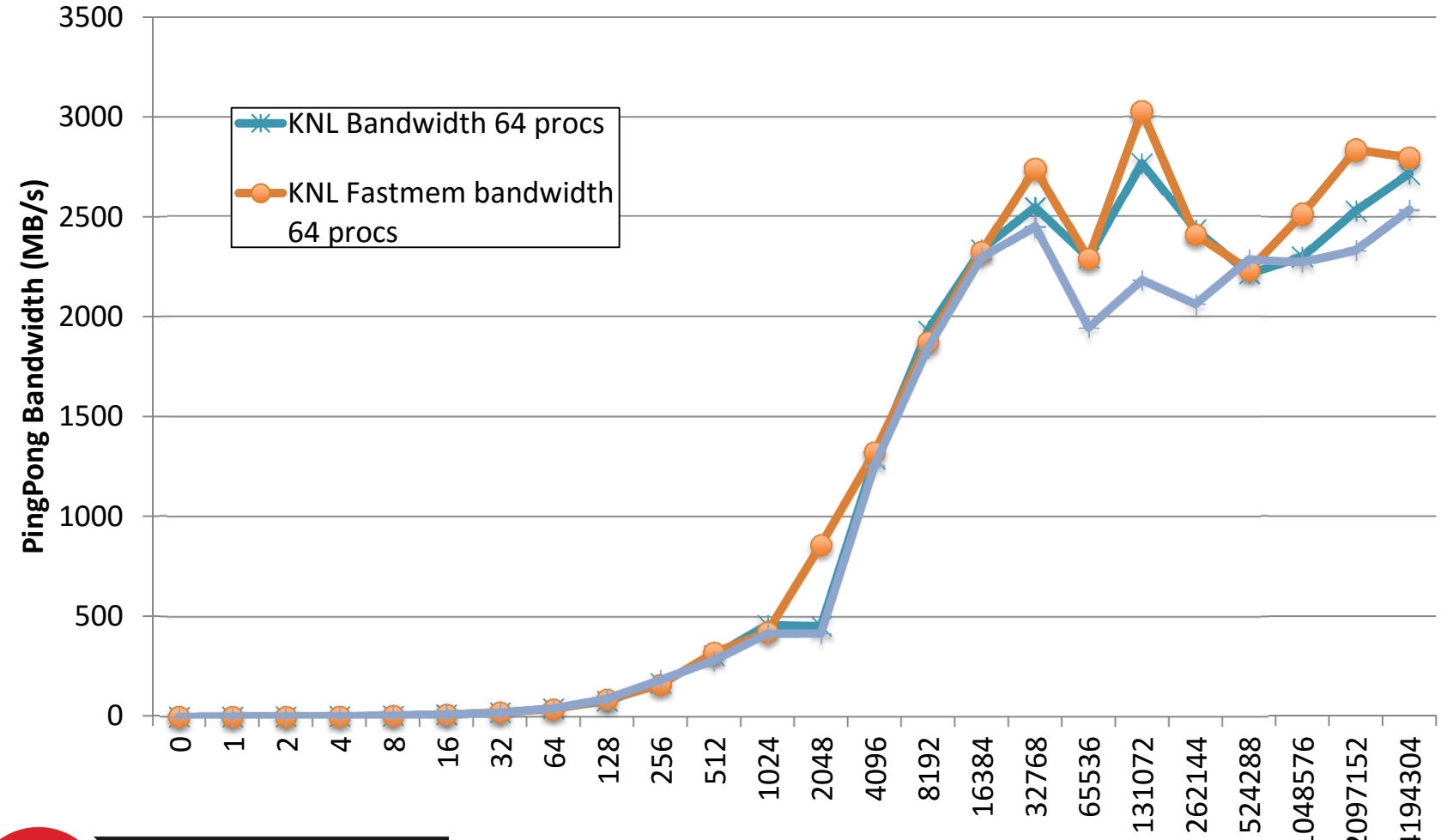


archer

epcc

THE UNIVERSITY OF EDINBURGH

MPI Performance – PingPong – Memory modes

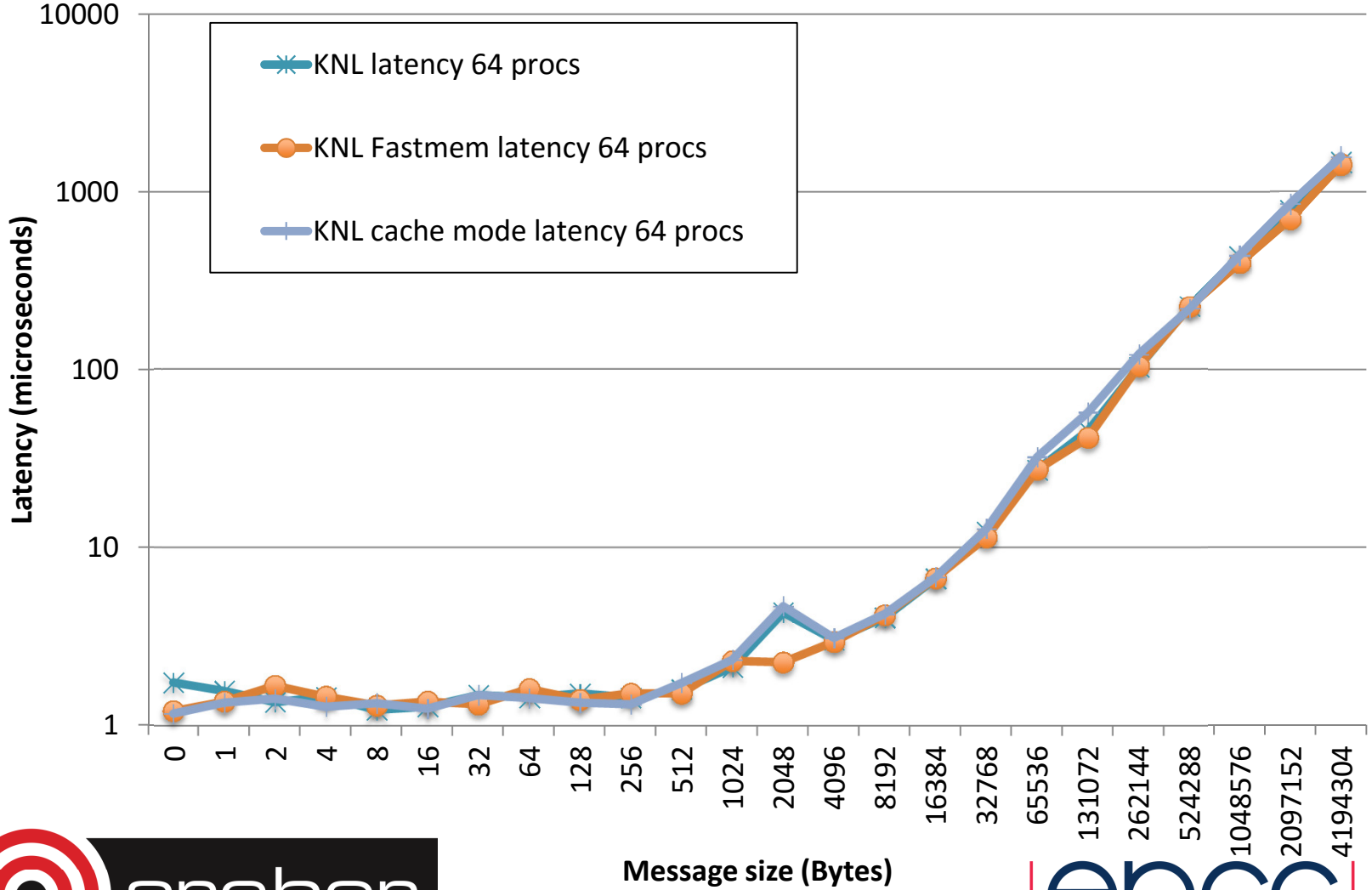


archer

epcc

THE UNIVERSITY OF EDINBURGH

MPI Performance – PingPong – Memory modes

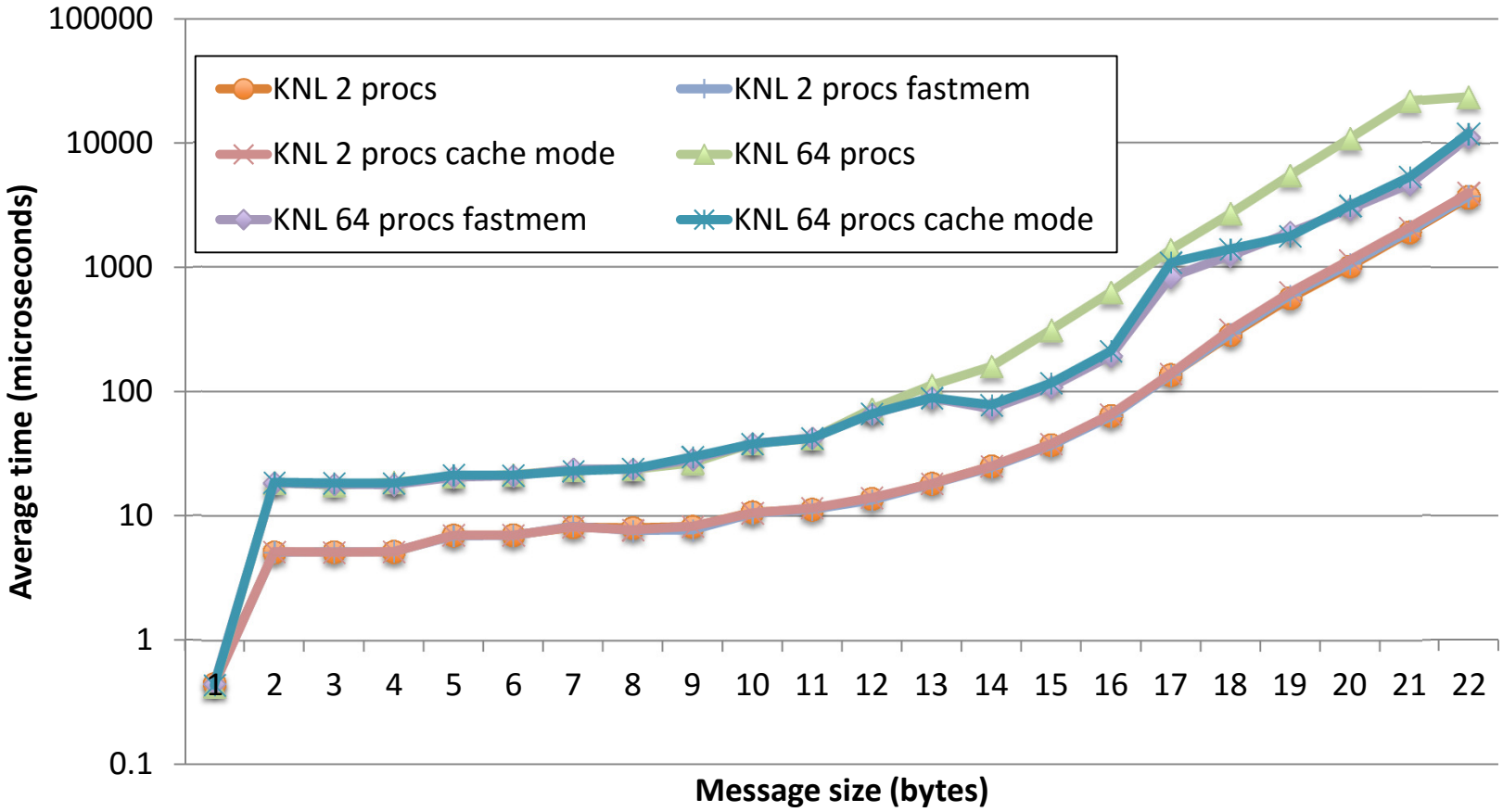


archer

epcc

THE UNIVERSITY OF EDINBURGH

MPI_Allreduce KNL different memory modes for 2 and 64 processor benchmarks



archer

epcc

THE UNIVERSITY OF EDINBURGH

ARCHER KNL

- 12 KNLs in test system
- Should be available mid-October
  - ARCHER users get access
  - Non-ARCHER users can get access through driving test
- Initial access will be unrestricted
  - After first month will be budgeted
- More details in the next virtual seminar(s)
- eCSE funding for KNL porting/optimisation
  - Next virtual tutorial: Wednesday 21 Sept 2016 15:00
  - Using ARCHER KNL: Wednesday 12th October 2016 15:00

EPCC IPCC

- EPCC has IPCC collaboration with Intel
- Working on porting and optimising codes on Xeon Phi
- Training and support of Xeon Phi
- Get in touch if you've got any questions, or something you'd like to collaborate on

Goodbye

Virtual tutorial has finished

Please check here for future tutorials and training

http://www.archer.ac.uk/training

http://www.archer.ac.uk/training/virtual/

archer

epcc

THE UNIVERSITY OF EDINBURGH