

ACHIEVABLE PERFORMANCE

Adrian Jackson
adrianj@epcc.ed.ac.uk



Peak 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
- MPI performance
 - Can be significantly slower than host



Achievable hardware performance

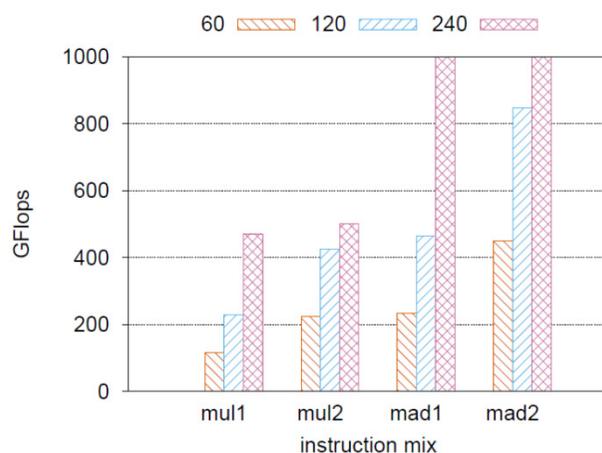


Fig. 2. Arithmetic throughput using different numbers of threads (60, 120, 240), different instruction mixes (mul1 versus mad), and issue widths (using one and two independent instruction streams).

From “An Empirical Study of Intel Xeon Phi”

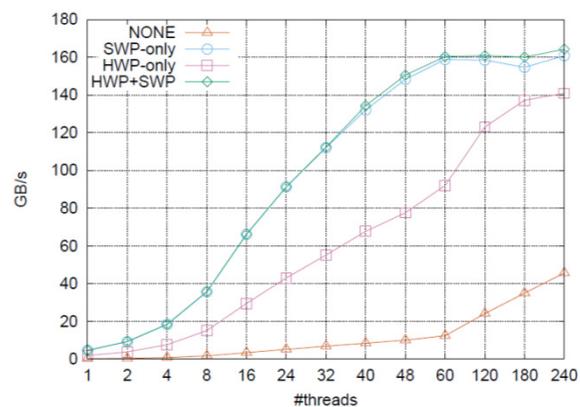
TABLE I
THE LATENCY OF VECTOR INSTRUCTIONS ON XEON PHI (IN CYCLES)

Instruction	Category	Latency
kand, kor, knot, kxor	mask instructions	2
vaddpd, vfmadd213pd, vmulpd, vsubpd	arithmetic instructions	4
vcvtdq2pd, vcvtxpntdq2ps, vcvtxpntps2dq, vcvtps2pd	convert instructions	5
vpermd, vpermf32x4	permutation instructions	6
vexp223ps, vlog2ps, vrcp23ps, vrsqrt23ps	extended mathematical instructions	6

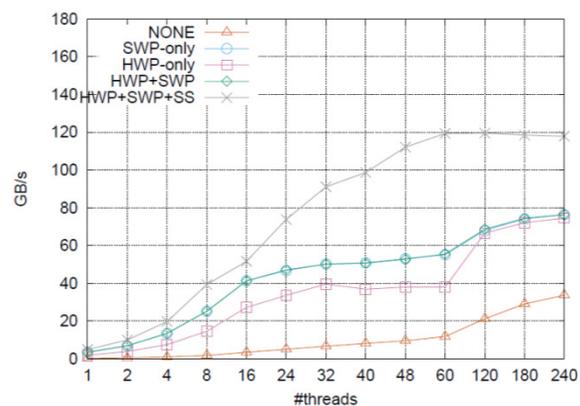
From “An Empirical Study of Intel Xeon Phi”

Achievable hardware performance

- $\sim 1\text{ns} \Rightarrow 1 \text{ cycle}$



(a) read



(b) write

Fig. 6. Read and write memory bandwidth. From "An Empirical Study of Intel Xeon Phi"

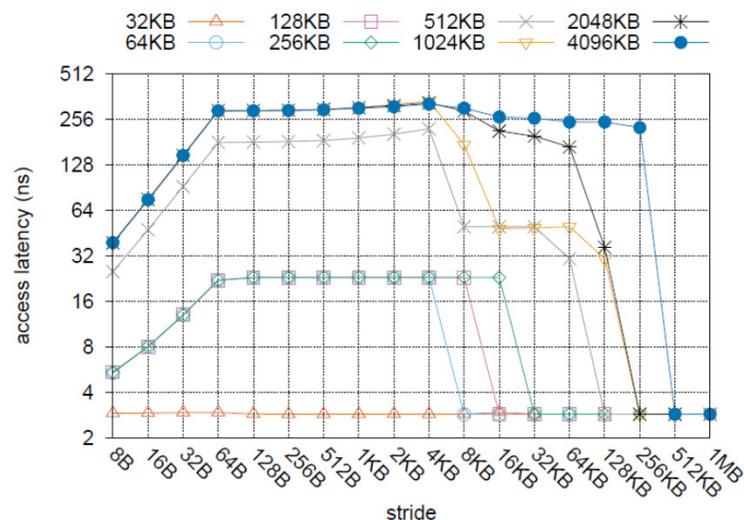


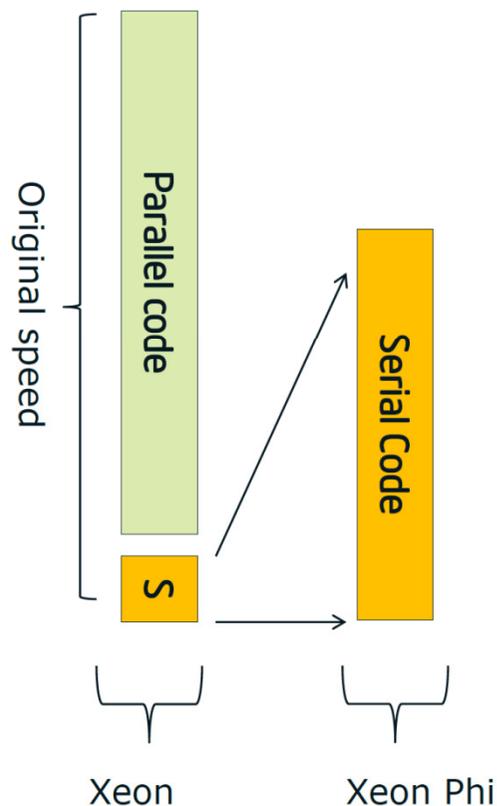
Fig. 4. Average memory latency when changing strides and datasets. The x-axis is logarithmic and it represents the pointer chasing stride.

From "An Empirical Study of Intel Xeon Phi"



Serial code

The Serial Factor



Slide from Intel

Serial Factor =
Clock Factor * ILP Factor * Issue Factor

Where

Clock Factor = $2.6 / 1.09$

For FMA type calculations
ILP Factor*** = $2/2 = 1$

For non-FMA type calculations
ILP Factor = $2/1$

Issue factor =

Num cycles to issue instruction on Phi /
Num cycles to issue instruction on Xeon
= $2/1$

Note: in single threaded code Xeon Phi uses
two cycles to issue an instruction
(in threaded mode it takes just one cycle)

** FMA: source code is capable of using Fused Multiple Add
when built for Xeon Phi

FMA
x4.77
slower**

**Non-FMA
x9.54
slower**

Speedups

'Finger in the air' speedups (from 2 socket 2.6Ghz SSE2)

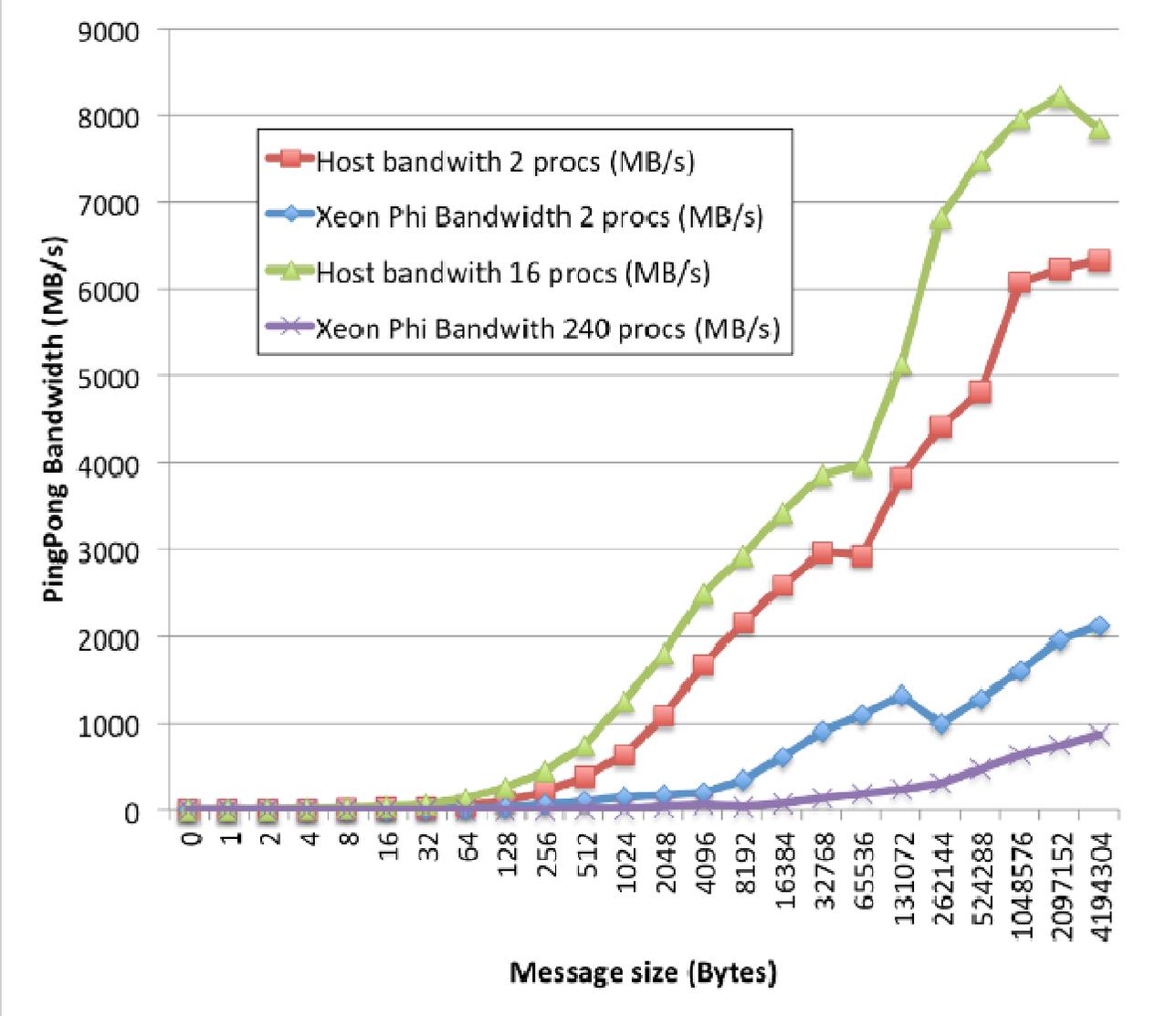
- An application that is **highly parallel** and effectively **vectorised** will speed up by **x2.5**
- An application that is **highly parallel** but **not vectorised** will speed up by **x1.3**
- An application that is **not parallel** but **is vectorised** will **slow down** by **x1.5**
- A **Serial** application will **slow down** by **x12.0**
- A **Bandwidth** constrained application will speed up by **x2.4**

Slide from Intel

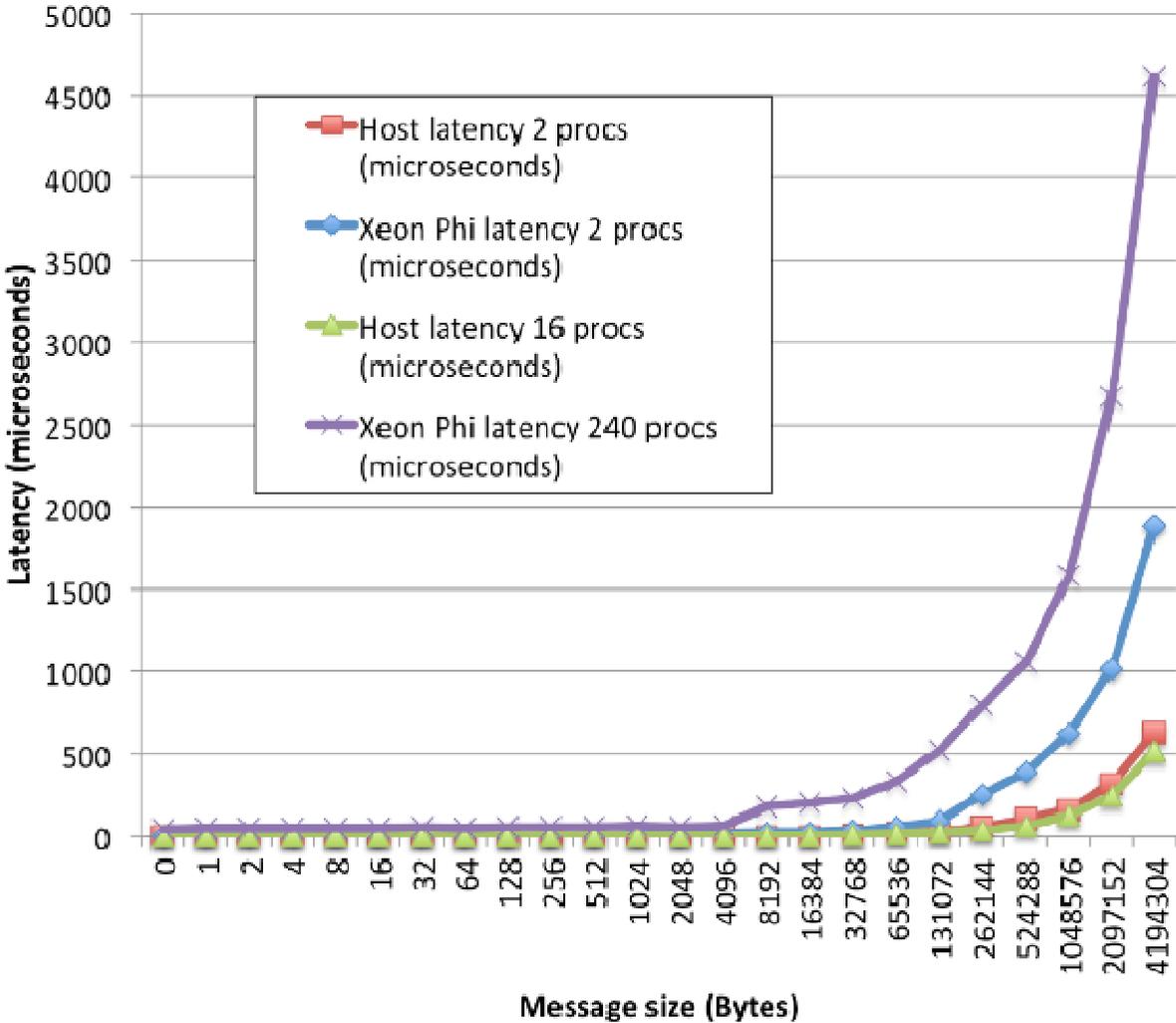
| epcc |



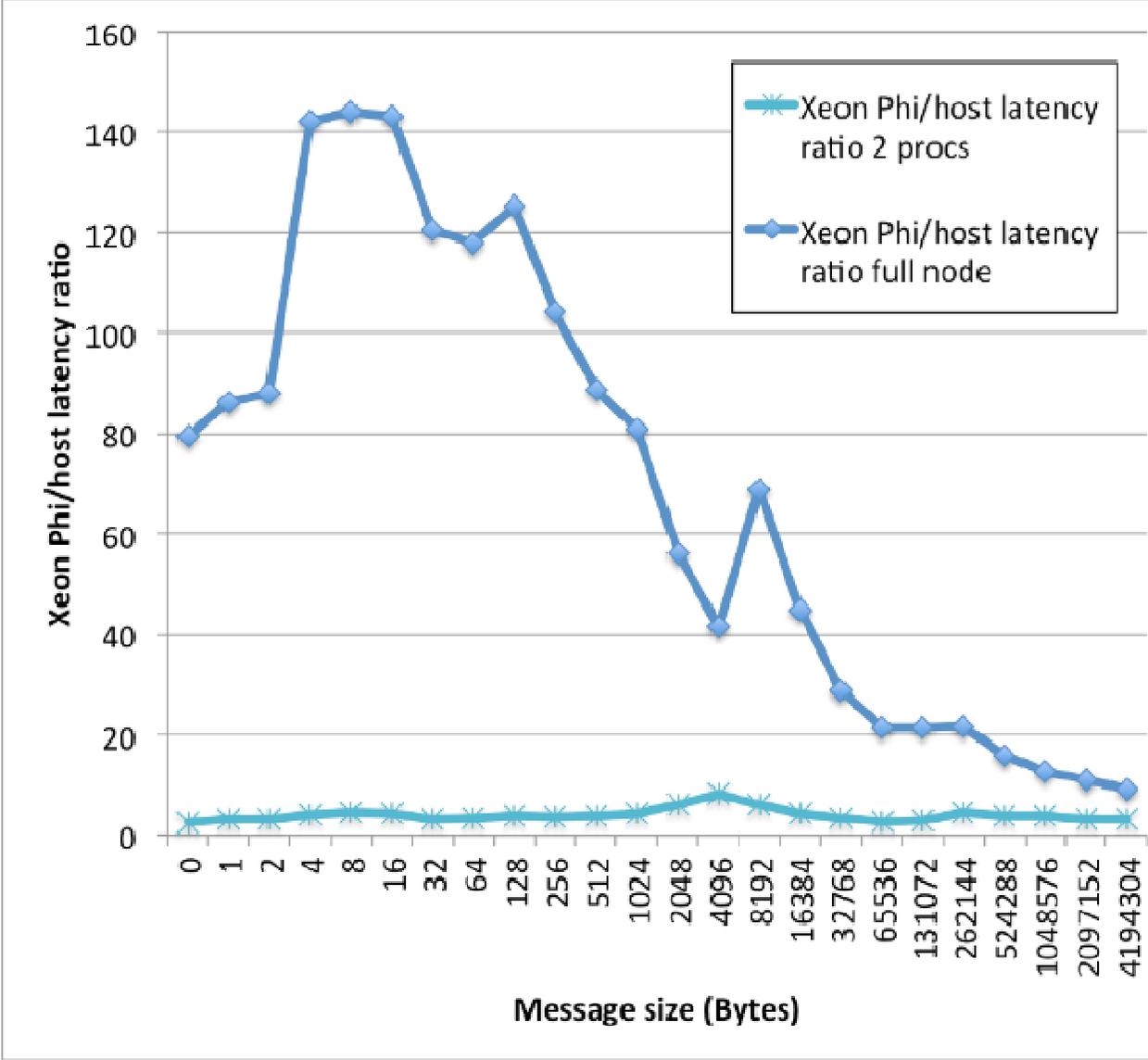
PingPong Bandwidth



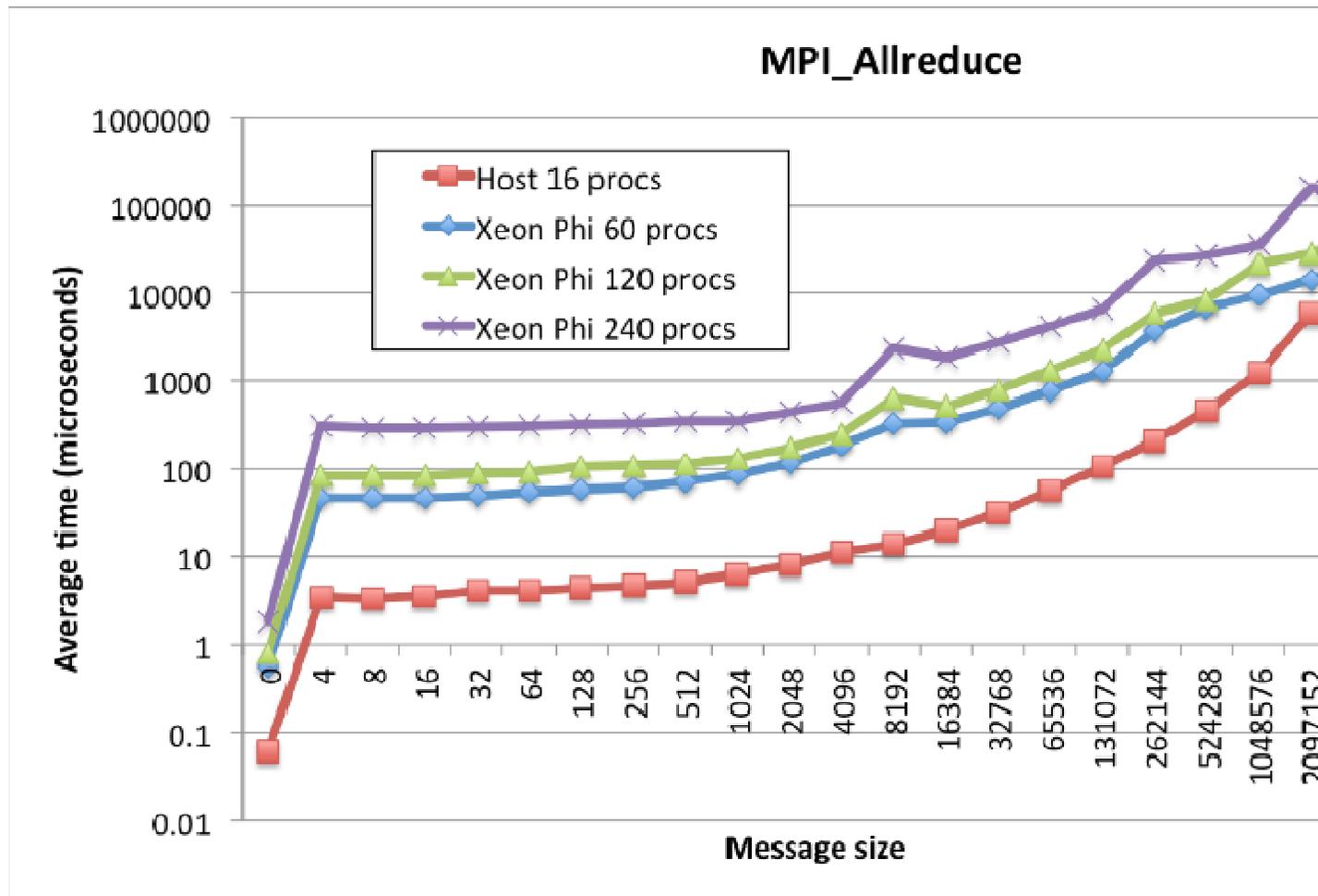
PingPong Latency



PingPong Latency



MPI_Allreduce



Summary

- Xeon Phi hardware has potential for high peak performance
 - Particularly at single precision
- Achievable performance generally lower
 - Especially if memory or communication bound
- Full vectorisation and FMA essential for highest performance
 - Can still get good performance without
- Good memory re-use essential for highest performance
 - Same as for normal CPU code