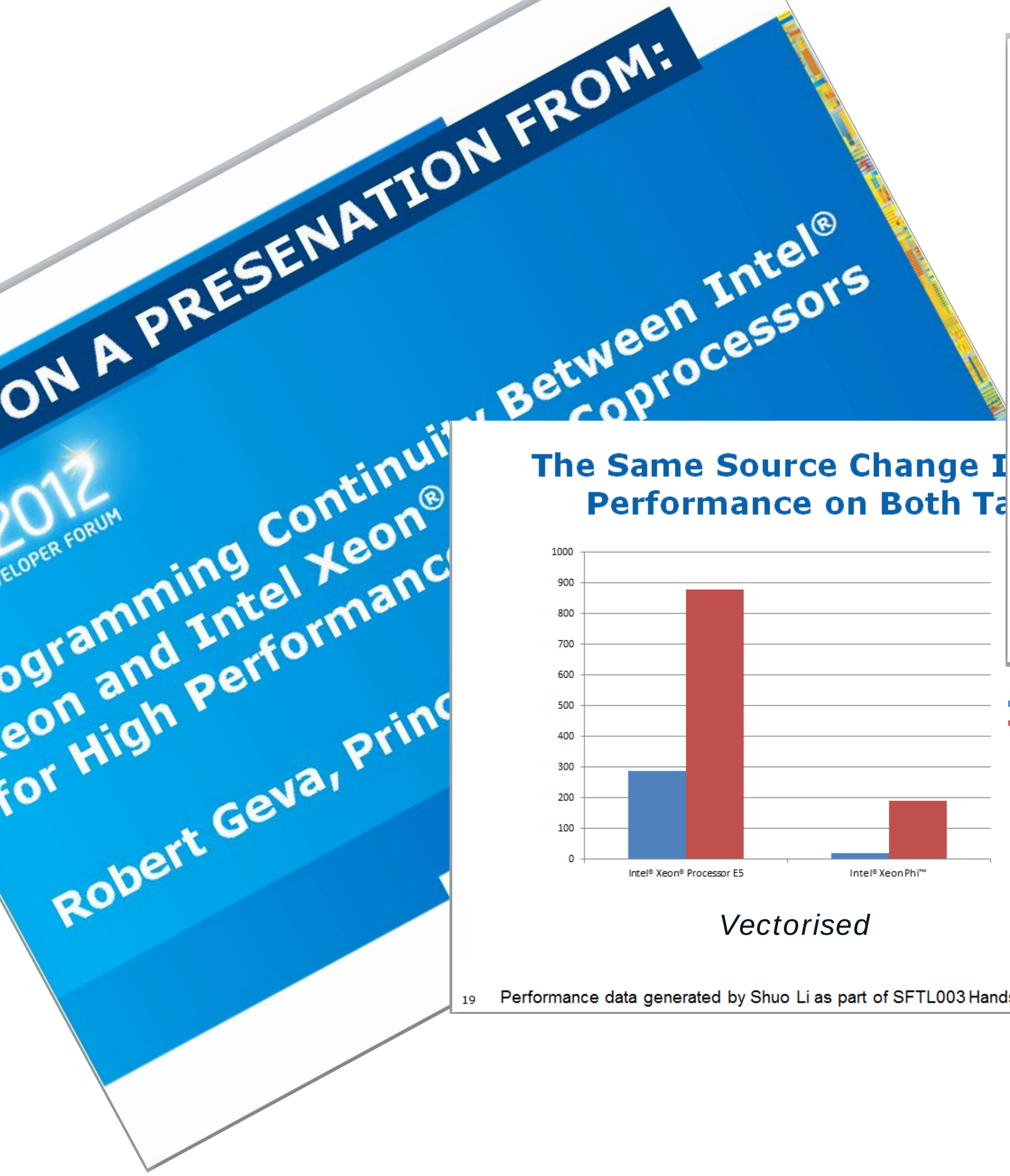
Programming for Intel® Xeon Phi™

Stephen Blair-Chappell

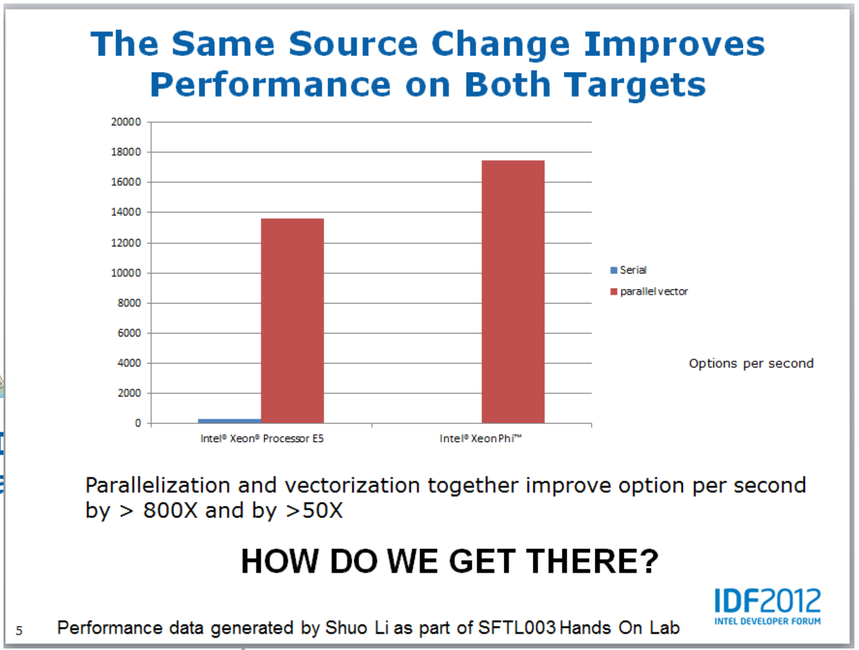


...ON A PRESENATION FROM:

...2012 ...ELOPER FORUM

...ogramming Continuity Between Intel® ...eon and Intel Xeon® ...Coprocessors ...for High Performanc...

Robert Geva, Princ...



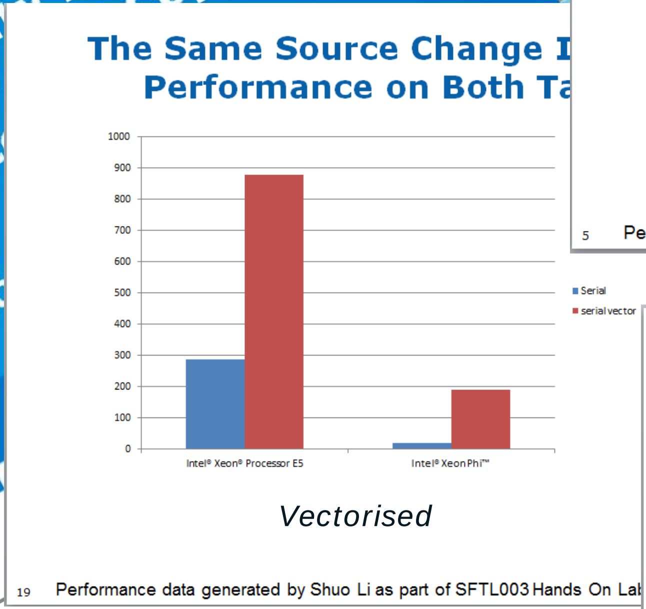
The Same Source Change Improves Performance on Both Targets

Parallelization and vectorization together improve option per second by > 800X and by >50X

HOW DO WE GET THERE?

5 Performance data generated by Shuo Li as part of SFTL003 Hands On Lab

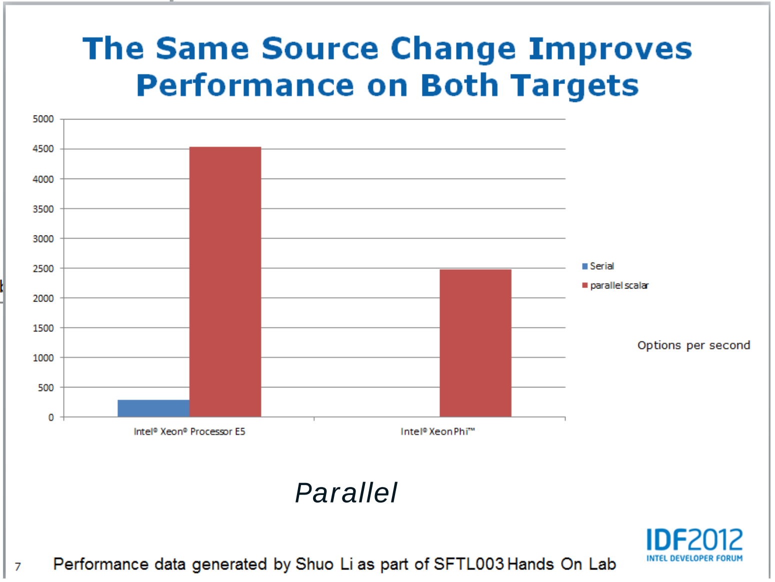
IDF2012 INTEL DEVELOPER FORUM



The Same Source Change I... Performance on Both Ta...

Vectorised

19 Performance data generated by Shuo Li as part of SFTL003 Hands On Lab



The Same Source Change Improves Performance on Both Targets

Parallel

7 Performance data generated by Shuo Li as part of SFTL003 Hands On Lab

IDF2012 INTEL DEVELOPER FORUM

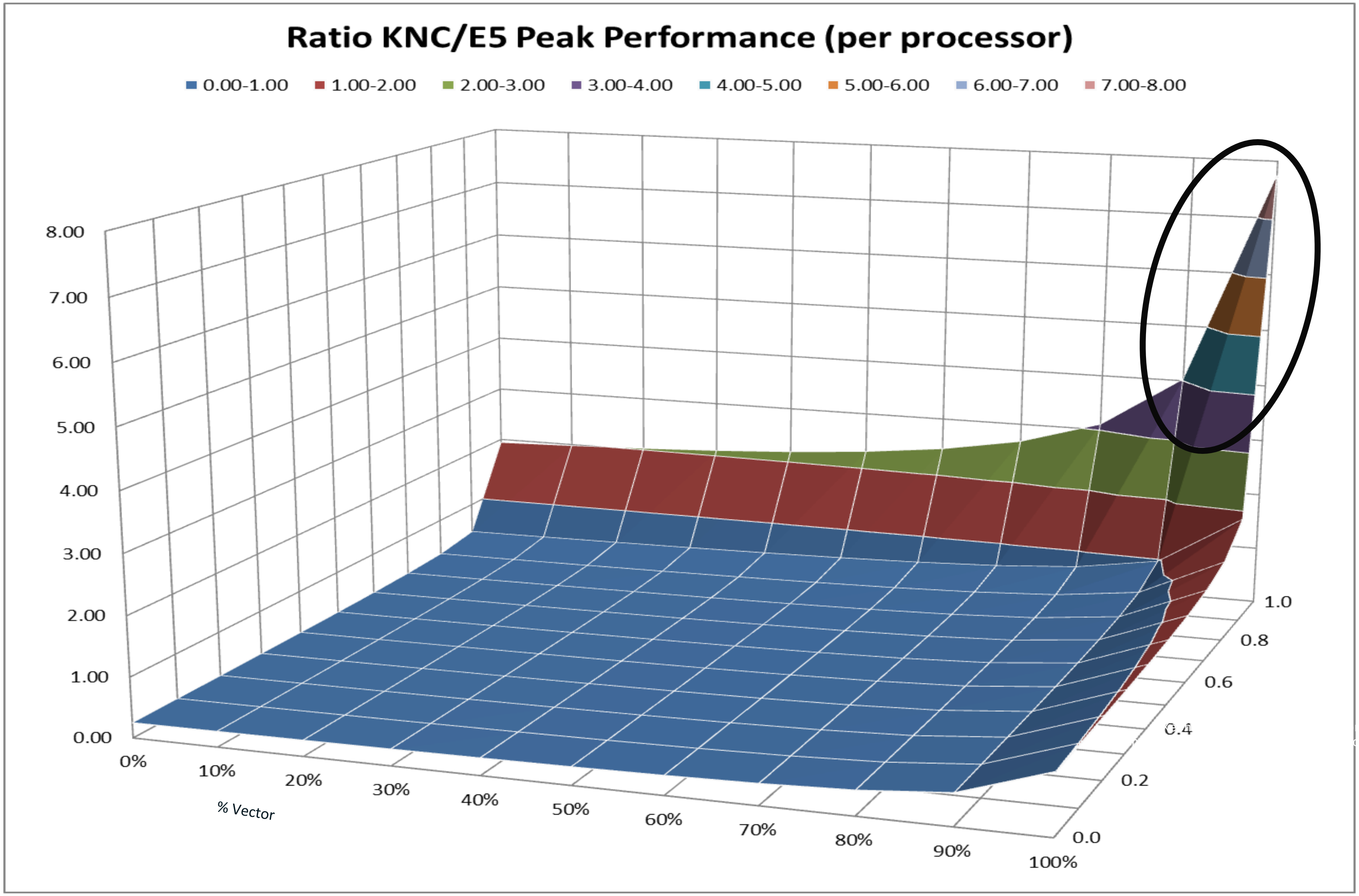
On the graphs, bigger is better

Code must be

highly Parallel

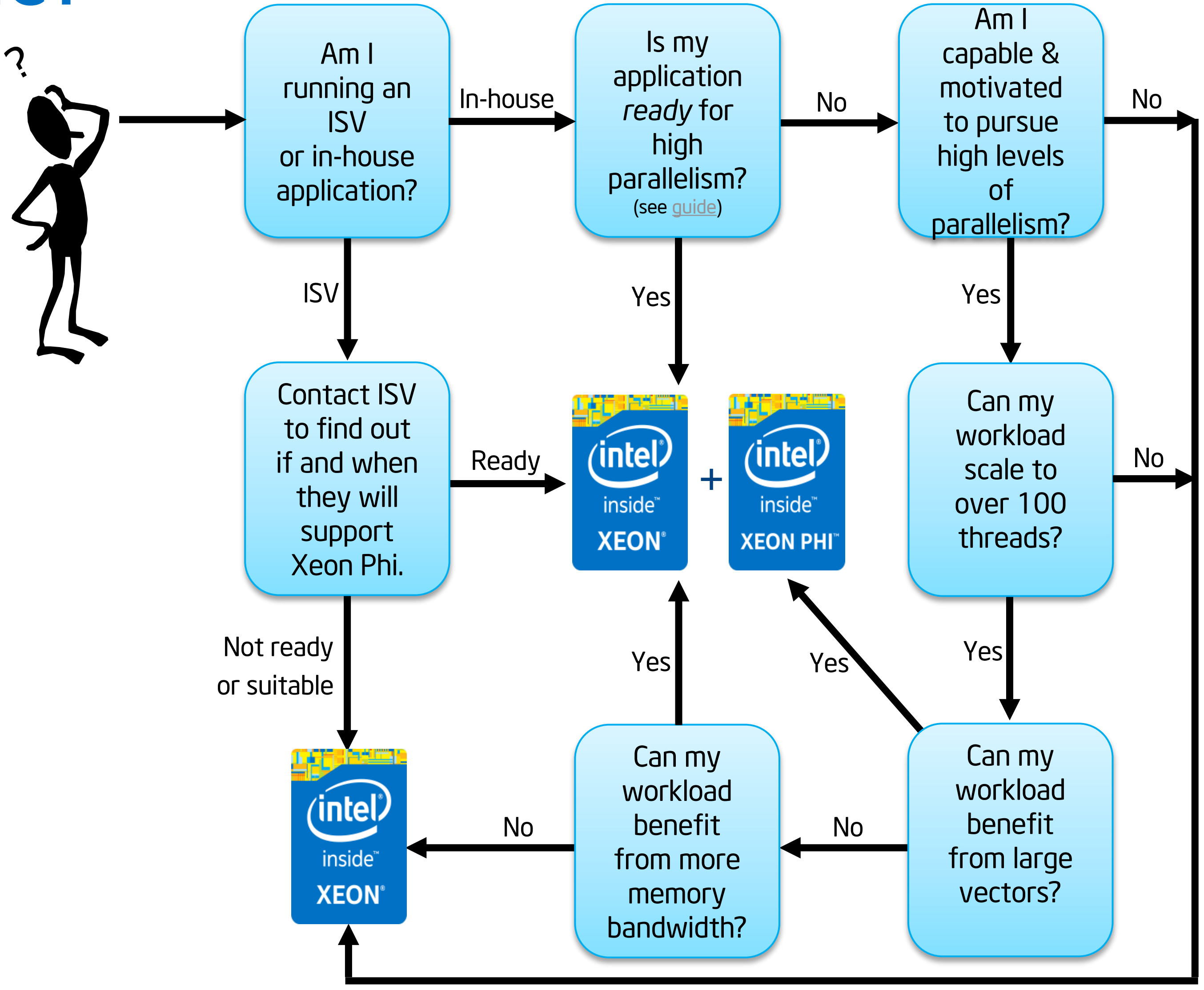
effectively Vectorised

Application Performance: Intel® Xeon Phi™ Coprocessor



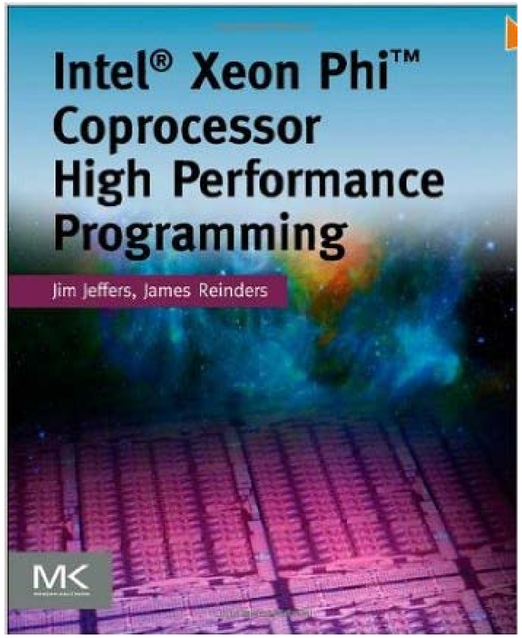
For more information go to http://www.intel.com/performance

Is the Intel® Xeon Phi™ Coprocessor right for me?


Am I running an ISV or in-house application?
In-house
ISV
Is my application *ready* for high parallelism? (see guide)
No
Yes
Am I capable & motivated to pursue high levels of parallelism?
No
Yes
Contact ISV to find out if and when they will support Xeon Phi.
Ready
Not ready or suitable
intel inside XEON
+
intel inside XEON PHI
Can my workload scale to over 100 threads?
No
Yes
Can my workload benefit from large vectors?
Yes
No
Can my workload benefit from more memory bandwidth?
Yes
No
intel inside XEON

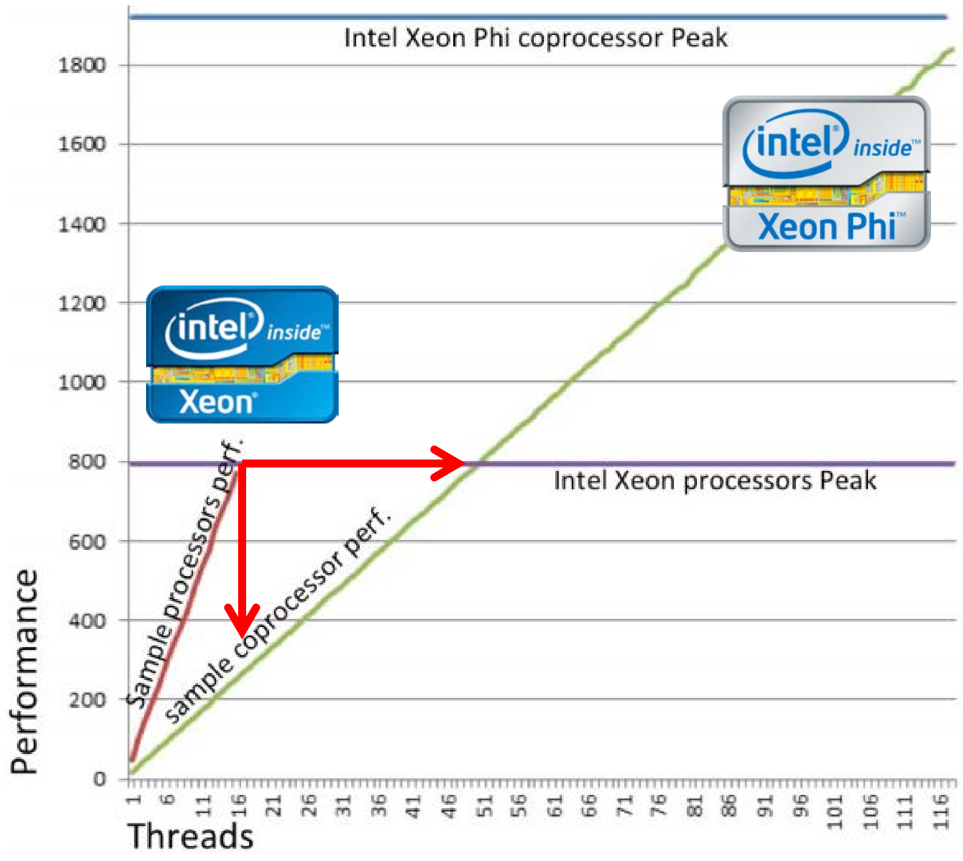
How many threads ?

“An application must scale well past one hundred threads to qualify as highly parallel”


Intel® Xeon Phi™ Coprocessor High Performance Programming
Jim Jeffers, James Reinders

Jim Jeffers
James Reinders.
ISBN: **978-0124104143**

Parallel Performance Potential



If your performance needs are met by a an Intel Xeon® processor, they will be achieved with fewer threads than on a coprocessor

On a coprocessor:

- Need more threads to achieve same performance
- Same thread count can yield less performance

Intel Xeon Phi excels on *highly parallel applications*

Optimization Notice

Different Ways of Inserting Vectorised Code

Use Performance Libraries
(e.g. IPP and MKL)

Compiler: Fully automatic vectorization

Cilk Plus Array Notation

Compiler: Auto vectorization hints
(#pragma ivdep, …)

User Mandated Vectorization
(SIMD Directive)

Manual CPU Dispatch
(__declspec(cpu_dispatch …))

SIMD intrinsic class (F32vec4 add)

Vector intrinsic (mm_add_ps())

Assembler code (addps)

Ease of use

Faster Code

Programmer control

The Lab for this section uses
Automatic Vectorisation

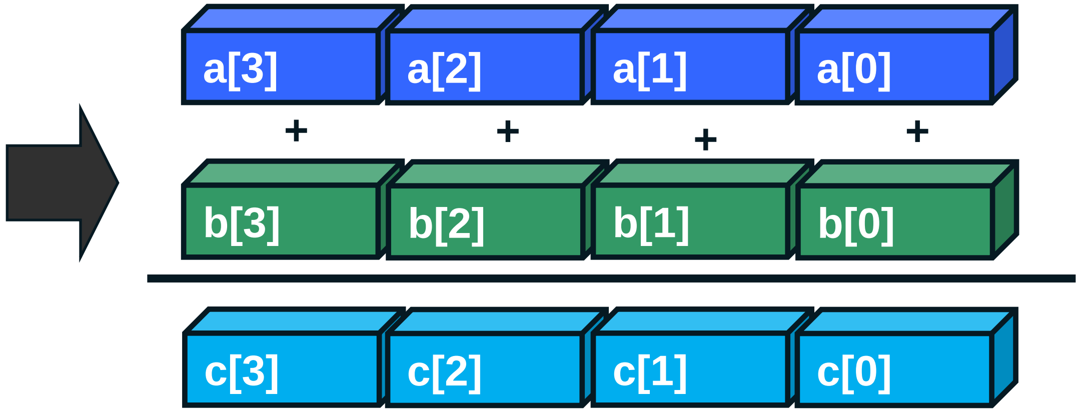
Optimization Notice

Vectorisation is …

Faster Code

1999	2000	2004	2006	2007	2008	2009	2011	2012\2013	2012
SSE	SSE2	SSE3	SSSE3	SSE4.1	SSE4.2	AES-NI	AVX	AVX2	MIC
70 instr Single-Precision Vectors Streaming operations	144 instr Double-precision Vectors 8/16/32 64/128-bit vector integer	13 instr Complex Data	32 instr Decode	47 instr Video Graphics building blocks Advanced vector instr	8 instr String/XML processing POP-Count CRC	7 instr Encryption and Decryption Key Generation	~100 new instr. ~300 legacy sse instr updated 256-bit vector 3 and 4-operand instructions	Int. AVX expands to 256 bit Improved bit manip. fma Vector shifts Gather	512-bit vector

```
for (i=0;i<MAX;i++)
   c[i]=a[i]+b[i];
```



Key Differentiators
Xeon Phi vs Workstation

More Cores

Slower Clock Speed

Wider SIMD registers

Faster Bandwidth

In-order pipeline

Theoretical Peak Flops Performance Example

Frequency * Num Sockets * Num Cores * Vector Width * FP Ops

Two socket Intel® Xeon® E5-2670 Processor

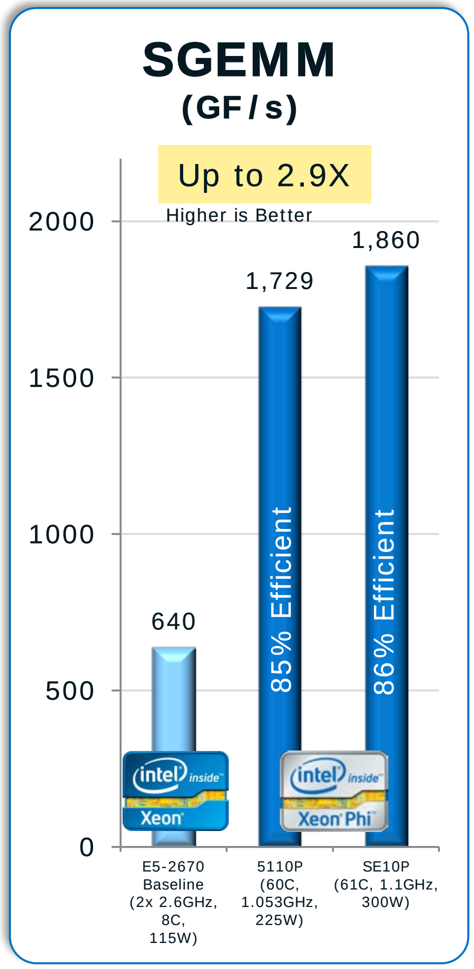
Freq	Sockets	Num Cores	Vector Width	FP Ops	GFlops
2.6	2	8	4	2	666

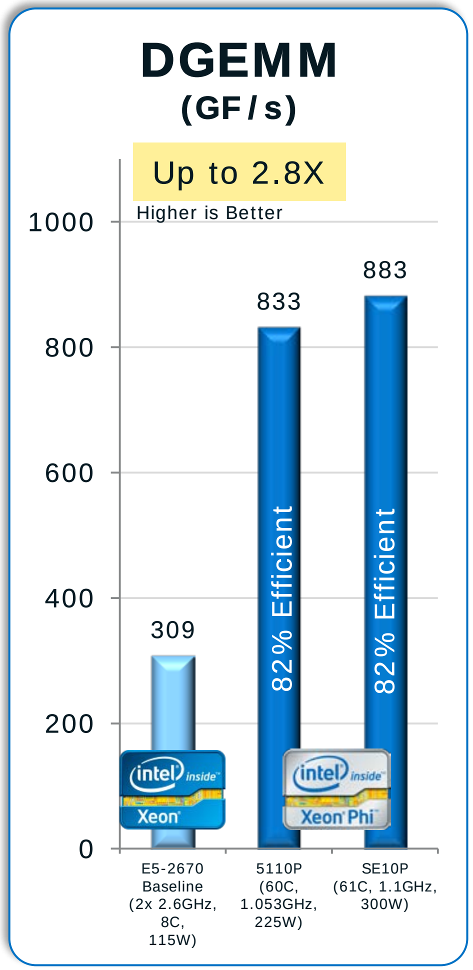
Single card Xeon Phi Coprocessor (B0)

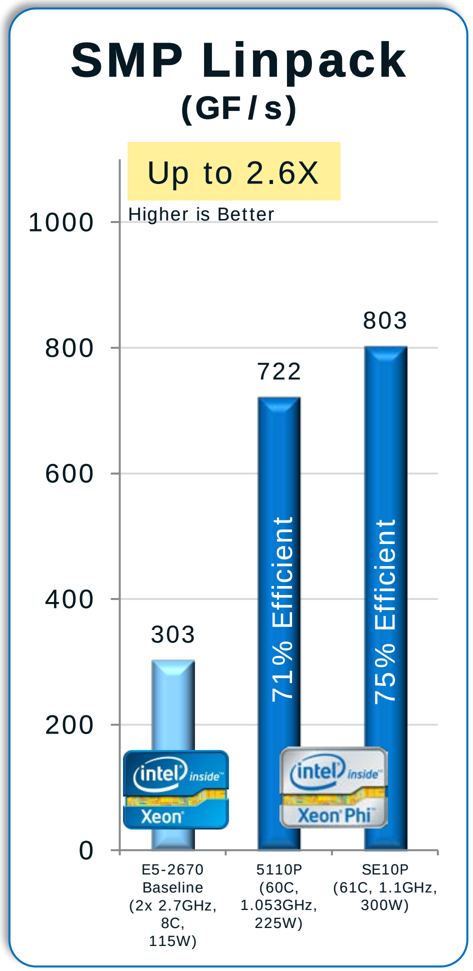
Freq	Sockets	Num Cores	Vector Width	FP Ops	GFlops
1.091	1	61	16	2 (using FMA)	2,128

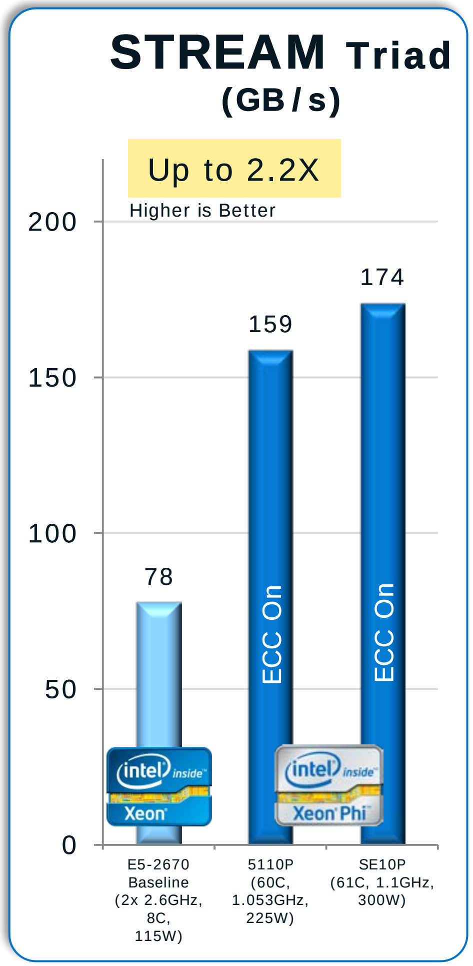
x3.20

Synthetic Benchmark Summary (Intel® MKL)









Coprocessor results: Benchmark run 100% on coprocessor, no help from Intel® Xeon® processor host (aka native)

Source: Intel Measured results as of October 26, 2012 Configuration Details: Please reference slide speaker notes.

For more information go to http://www.intel.com/performance

Optimization Notice

Intel® Xeon Phi™ Coprocessor:

Increases Application Performance up to 10x

Updated

Segment	Customer	Application	Performance Increase[1] vs. 2S Xeon*
Energy	Acceleware	8th order isotropic variable velocity	Up to 2.23x
	Sinopec	Seismic Imaging	Up to 2.53x[2]
	CNPC (China Oil & Gas)	GeoEast Pre-Stack Time Migration (Seismic)	Up to 3.54x[2]
Financial Services	Financial Services	BlackScholes SP Monte Carlo SP	Up to 7.5x Up to 10.75x
Physics	Jefferson Labs	Lattice QCD	Up to 2.79x
Finite Element	Sandia Labs	miniFE (Finite Element Solver)	Up to 2x[3] Up to 1.3x[5]
Solid State Physics	ZIB (Zuse-Institut Berlin)	Ising 3D (Solid State Physics)	Up to 3.46x
Digital Content Creation/Video	Intel Labs	Ray Tracing (incoherent rays)	Up to 1.88x[4]
	NEC	Video Transcoding	Up to 3.0x[2]
Astronomy	CSIRO/ASKAP (Australia Astronomy)	tHogbom Clean (Astronomy image smear removal)	Up to 2.27x
	TUM (Technische Universität München)	SG++ (Astronomy Adaptive Sparse Grids/Data Mining)	Up to 1.7x
Fluid Dynamics	AWE (Atomic Weapons Establishment - UK)	Cloverleaf (2D Structured Hydrodynamics)	1.77x

Notes:

1. 2S Xeon* vs. 1 Xeon Phi* (preproduction HW/SW & Application running 100% on coprocessor unless otherwise noted)
2. 2S Xeon* vs. 2S Xeon* + 2 Xeon Phi* (offload)
3. 8 node cluster, each node with 2S Xeon* (comparison is cluster performance with and without 1 Xeon Phi* per node) (Hetero)
4. Intel Measured Oct. 2012
5. 8 node cluster, each node with 2S Xeon* (comparison is cluster performance with Xeon only vs. Xeon Phi *only (1 Xeon Phi* per node) (Native)

Source: Customer Measured results as of October 22, 2012 Configuration Details: Please reference slide speaker notes.

For more information go to http://www.intel.com/performance

A Tale of Two Architectures

	Intel® Xeon® processor	Intel® Xeon Phi™ Coprocessor
Sockets	2	1
Clock Speed	2.6 GHz	1.1 GHz
Execution Style	Out-of-order	In-order
Cores/socket	8	Up to 61
HW Threads/Core	2	4
Thread switching	HyperThreading	Round Robin
SIMD widths	8SP, 4DP	16SP, 8DP
Peak Gflops	692SP, 346DP	2020SP, 1010DP
Memory Bandwidth	102GB/s	320GB/s
L1 DCache/Core	32kB	32kB
L2 Cache/Core	256kB	512kB
L3 Cache/Socket	30MB	none

Your code will benefit from running on Xeon Phi if …

- It is highly scalable

- Is effectively vectorised

  or bandwidth constrained

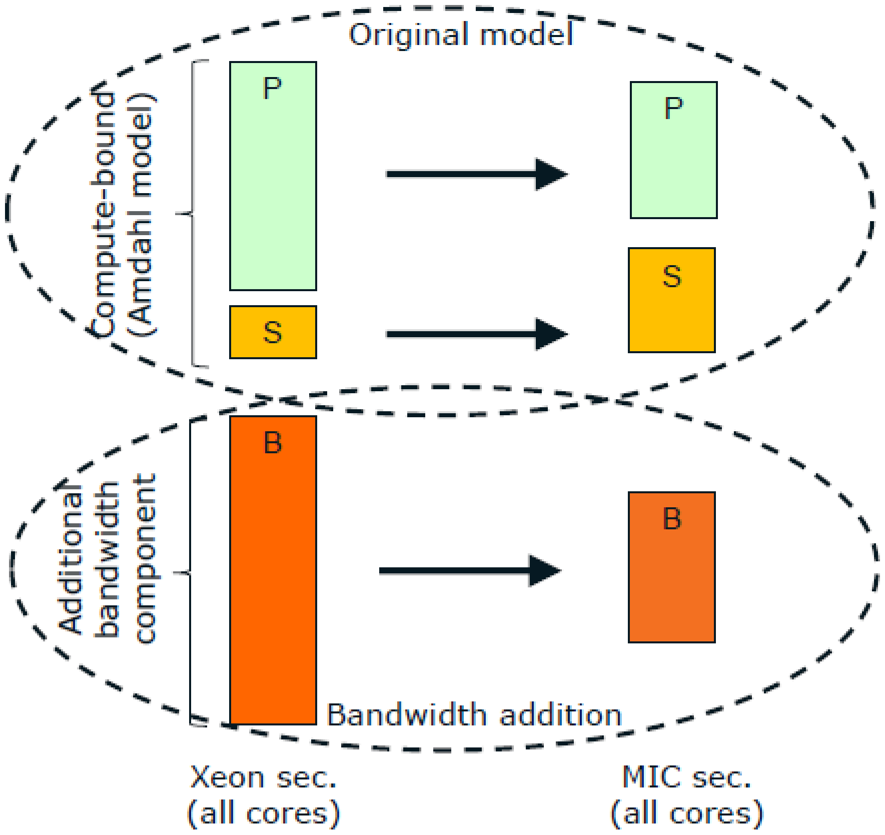
Three things to consider

Three components to consider

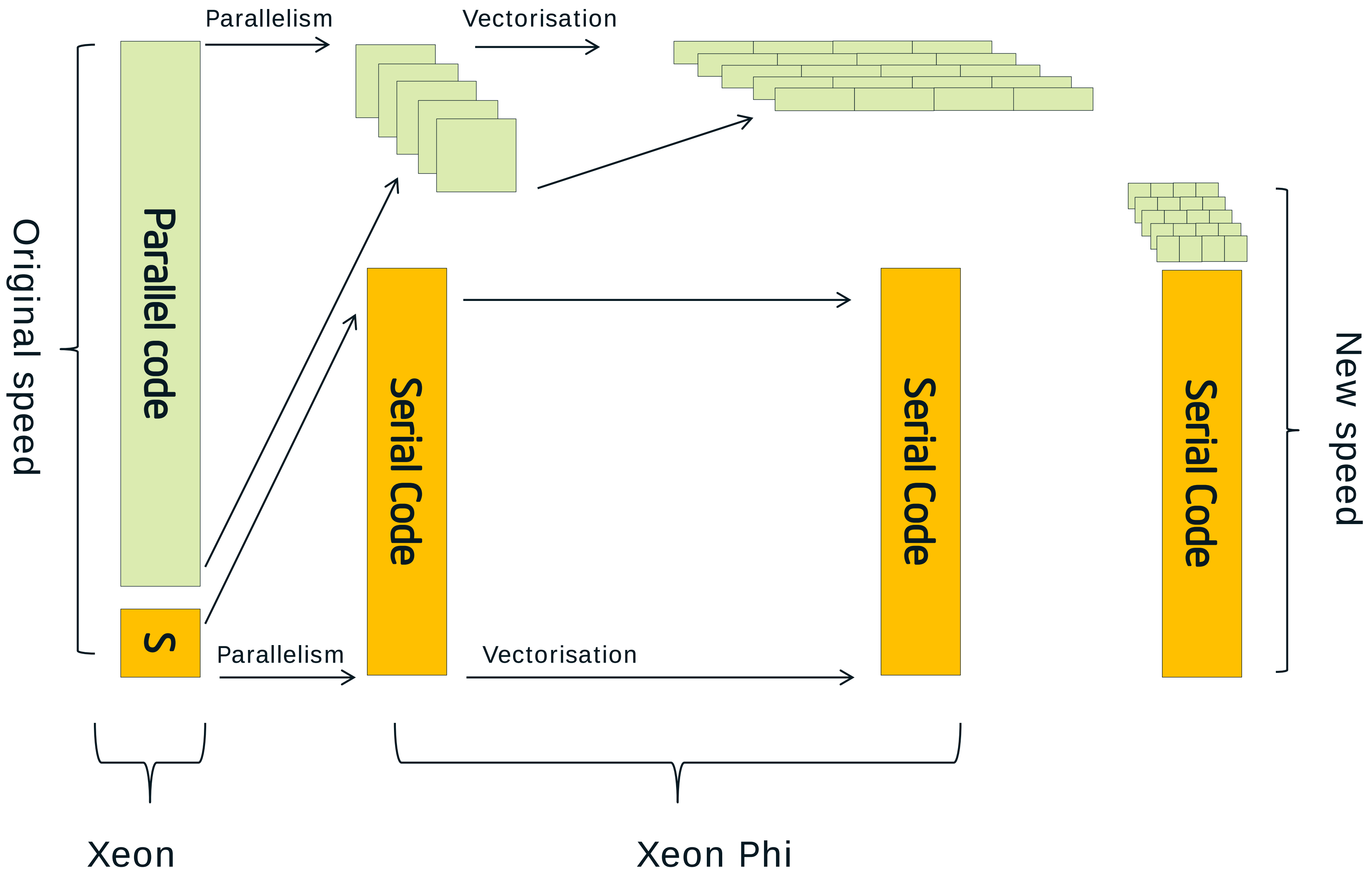
P – the parallel part of the program

S – the serial part of the program

B – the bandwidth constrained part of the program



Compute bound



The Parallel, Vector and Clock Factors

Parallel Factor =
Num Xeon Cores / Num Phi Cores

16 / 61 = 0.26229

Vector Factor =
(Xeon Vector Length * Xeon Instruction Level Parallelism) /
(Phi Vector Length * Phi Instruction Level Parallelism)

AVX-FMA**	4 * 2 / 8 * 2 = .5
AVX-non-FMA	4 * 2 / 8 * 1 = 1
SSE-FMA**	2 * 2 / 8 * 2 = .25
SSE-non-FMA	2 * 2 / 8 * 1 = .5

Clock Factor =
Xeon Frequency / Phi Frequency

3.1/1.09 = 2.844

Combined = Parallel Factor * Vector Factor * Clock factor

AVX-FMA**	0.26229 * .5 * 2.844 = 0.373
AVX-non-FMA	0.26229 * 1 * 2.844 = 0.746
SSE-FMA**	0.26229 * .25 * 2.844 = 0.187
SSE-non-FMA	0.26229 * .5 * 2.844 = 0.373

NB we are comparing 2 socket SNB with single coprocessor (64 bit floating point doubles)

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

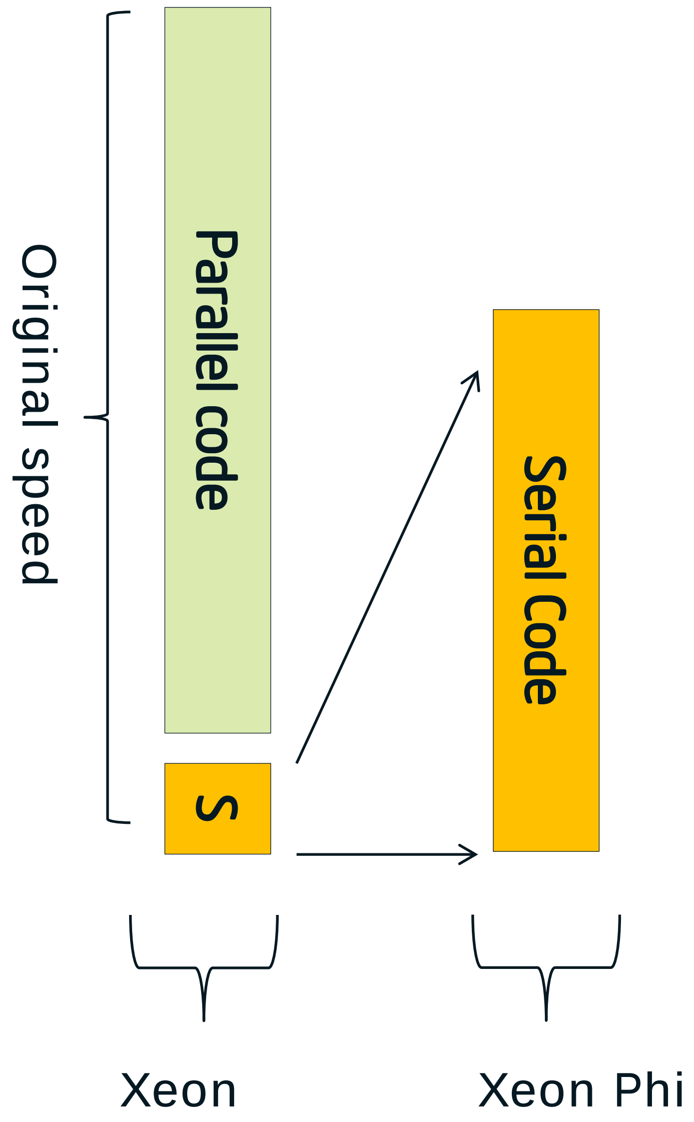
FMA**
x5.38
Faster
(SSE2)

FMA**
x2.68
Faster
(AVX)

Non-FMA
X2.68
Faster
(SSE2)

Non-FMA
x1.34
Faster
(AVX)

The Serial Factor



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

Back of the Envelope Calculation

- Measure percentage your code is vectorized.
- Measure how parallel your code is.
- Detect any Memory intense parts of Code.
- Scale-up (or down!) the values to take into account Xeon Phi

Factors (2.6 GHz Clock)

Host	SIMD	Serial	Vector	Parallel	Clock
Single socket 2.6 GHz. FMA**	AVX	4.772	0.5	0.1333	2.386
	SSE2		0.25		
Single socket 2.6 GHz No FMA	AVX	9.544	1		
	SSE2		0.5		
Twin socket 2.6 GHz FMA**	AVX	4.772	0.5	0.2666	
	SSE2		0.25		
Twin socket 2.6 GHz No FMA	AVX	9.544	1		
	SSE2		0.5		

Xeon: 8 cores per socket Phi: Using 60 of 61 cores

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

NOTE: Serial Factor already includes the Clock factor

Factors (3.1 GHz.)

Host	SIMD	Serial	Vector	Parallel	Clock
Single socket 3.1 GHz. FMA**	AVX	5.69	0.5	0.1333	2.844
	SSE2		0.25		
Single socket 3.1 GHz No FMA	AVX	11.38	1		
	SSE2		0.5		
Twin socket 3.1 GHz FMA**	AVX	5.69	0.5	0.2666	
	SSE2		0.25		
Twin socket 3.1 GHz No FMA	AVX	11.38	1		
	SSE2		0.5		

Xeon: 8 cores per socket Phi: Using 60 of 61 cores

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

NOTE: Serial Factor already includes the Clock factor

‘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**

What you experience in practice may be different from these figures.
These are only ‘back of the envelope’ figures.

LAB 1 – Activity 1
A Quick Smoke Test

LAB 1 – Activity 2
Measuring Vectorisation

LAB 1 – Activity 3
Measuring Concurrency

Legal Disclaimer & Optimization Notice