ATPESC

(Argonne Training Program on Extreme-Scale Computing)

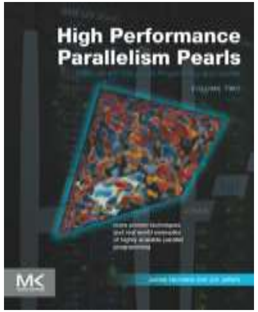
Computer Architecture and Structured Parallel Programming

James Reinders, Intel

August 3, 2015, Pheasant Run, St Charles, IL

09:30– 10:15

I have been fortunate, and I like to share. ☺

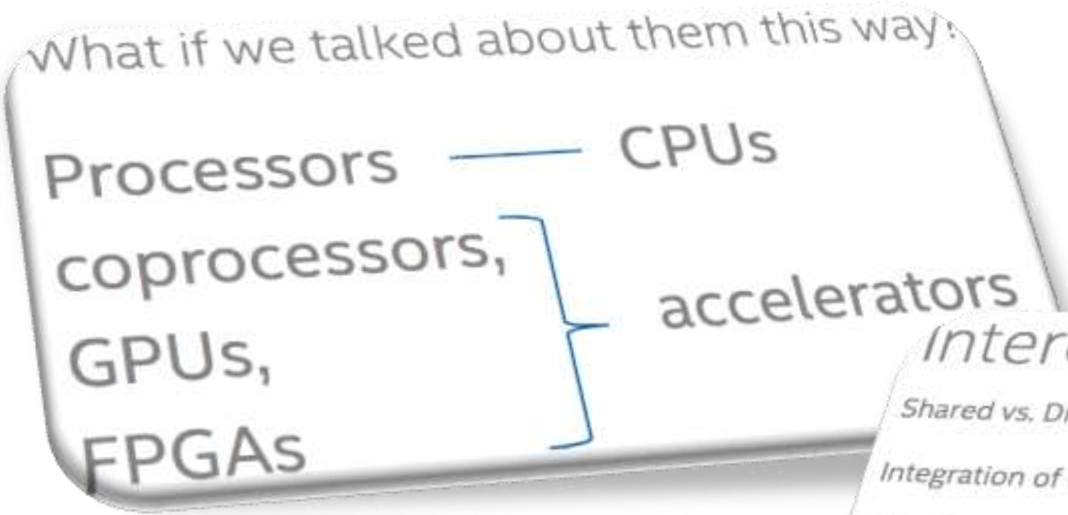


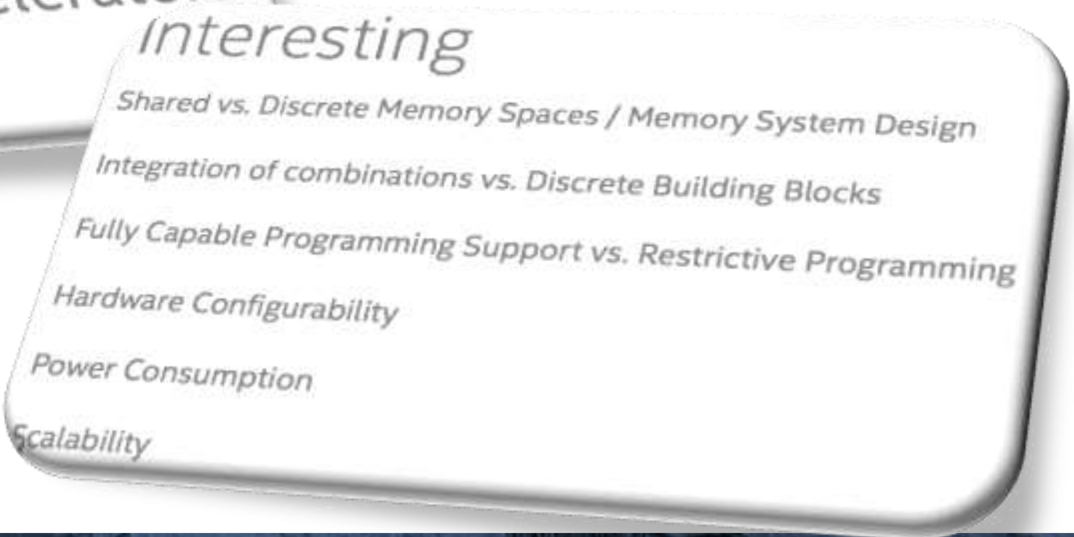
I saw example after example get performance and performance portability with "*just* parallel programming"

I summarize as "inspired by 61 cores"

Computer Architecture is FUN AGAIN

we need to make sure software is not collateral damage.





Performance and Performance Portability should be a requirement.

Processors,

coprocessors,

GPUs &

FPGAs,

What if we talked about them this way?

Processors — CPUs

coprocessors,
GPUs &
FPGAs

accelerators

Interesting

Shared vs. Discrete Memory Spaces / Memory System Design

Integration of combinations vs. Discrete Building Blocks

Fully Capable Programming Support vs. Restrictive Programming

Hardware Configurability

Power Consumption

Scalability

See the Forest

See the Forest

A cliché about someone missing the "big picture" because they focus too much on details:

They "cannot see the forest for the trees."

See the Forest

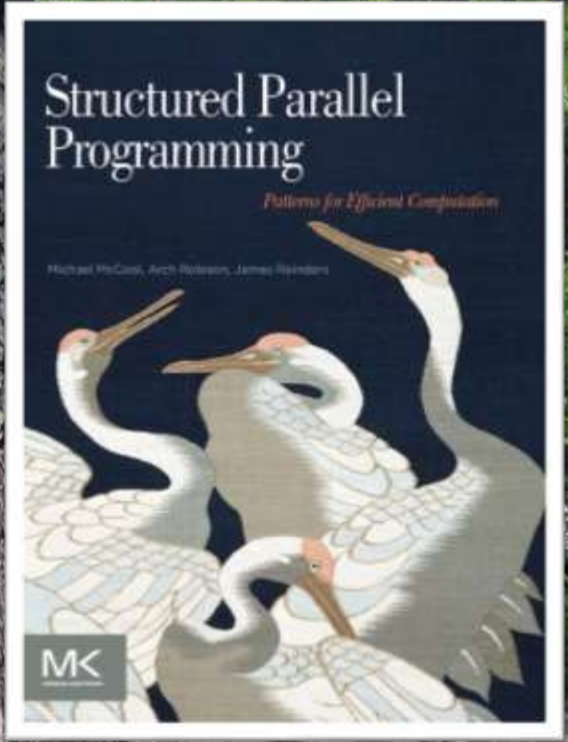
See the Forest

I ♥ architecture.

but...

See the Forest

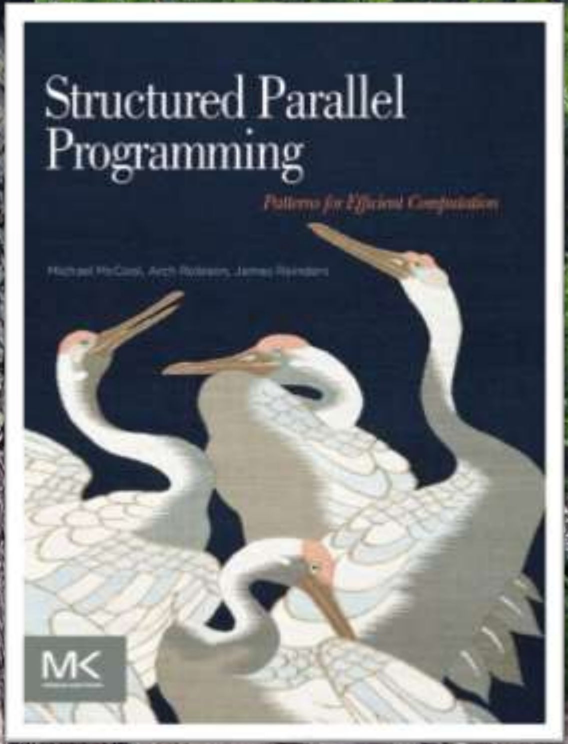
Can you teach parallel programming without first teaching computer architecture?



See the Forest

Can you teach parallel programming without first teaching computer architecture?

(Or without just teaching a single API?)



See the Forest

TREES
Cores
HW threads
Vectors
Offload
Heterogeneous
Cloud
Caches
NUMA

See the Forest

TREES	FOREST
Cores	Parallelism, Locality
HW threads	Parallelism, Locality
Vectors	Parallelism, Locality
Offload	Parallelism, Locality
Heterogeneous	Parallelism, Locality
Cloud	Parallelism, Locality
Caches	Parallelism, Locality
NUMA	Parallelism, Locality

Teach the Forest

Teach the Forest

Increase exposing parallelism.

Increase locality of reference.

Why? Because it's programming that addresses the universal needs of computers today and in the future future.

Teach the Forest

Increase exposing parallelism.

Increase locality of reference.

THIS

IS

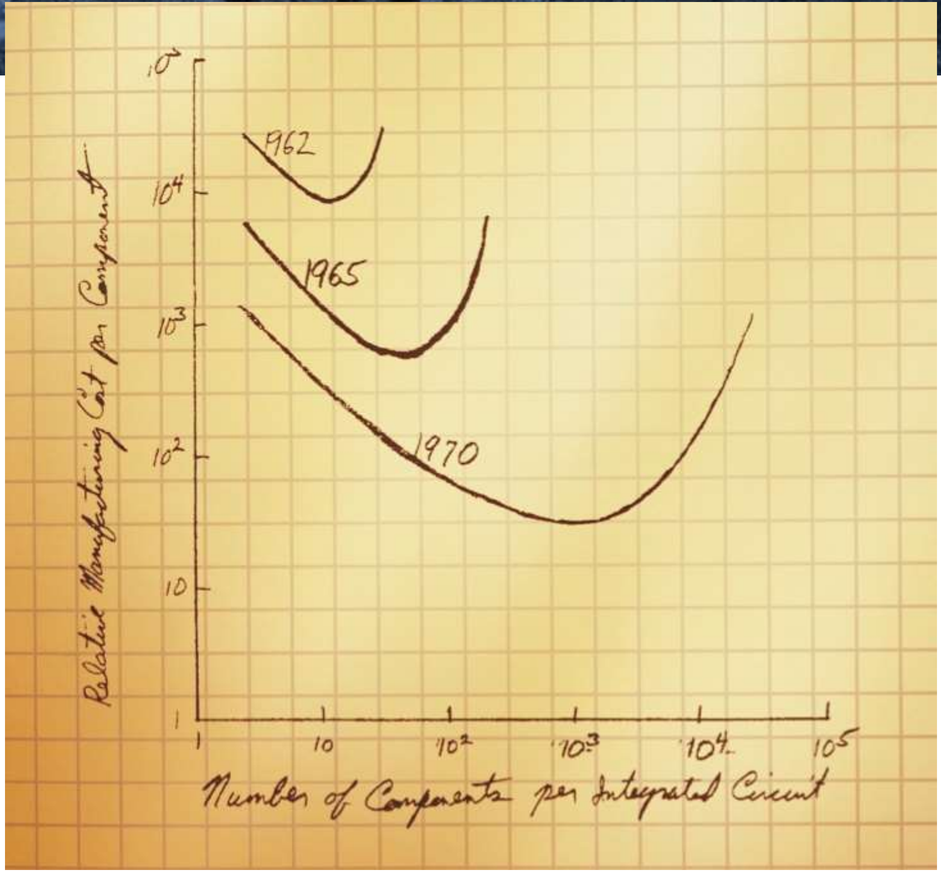
YOUR MISSION

Why so many cores?

Why Multicore?

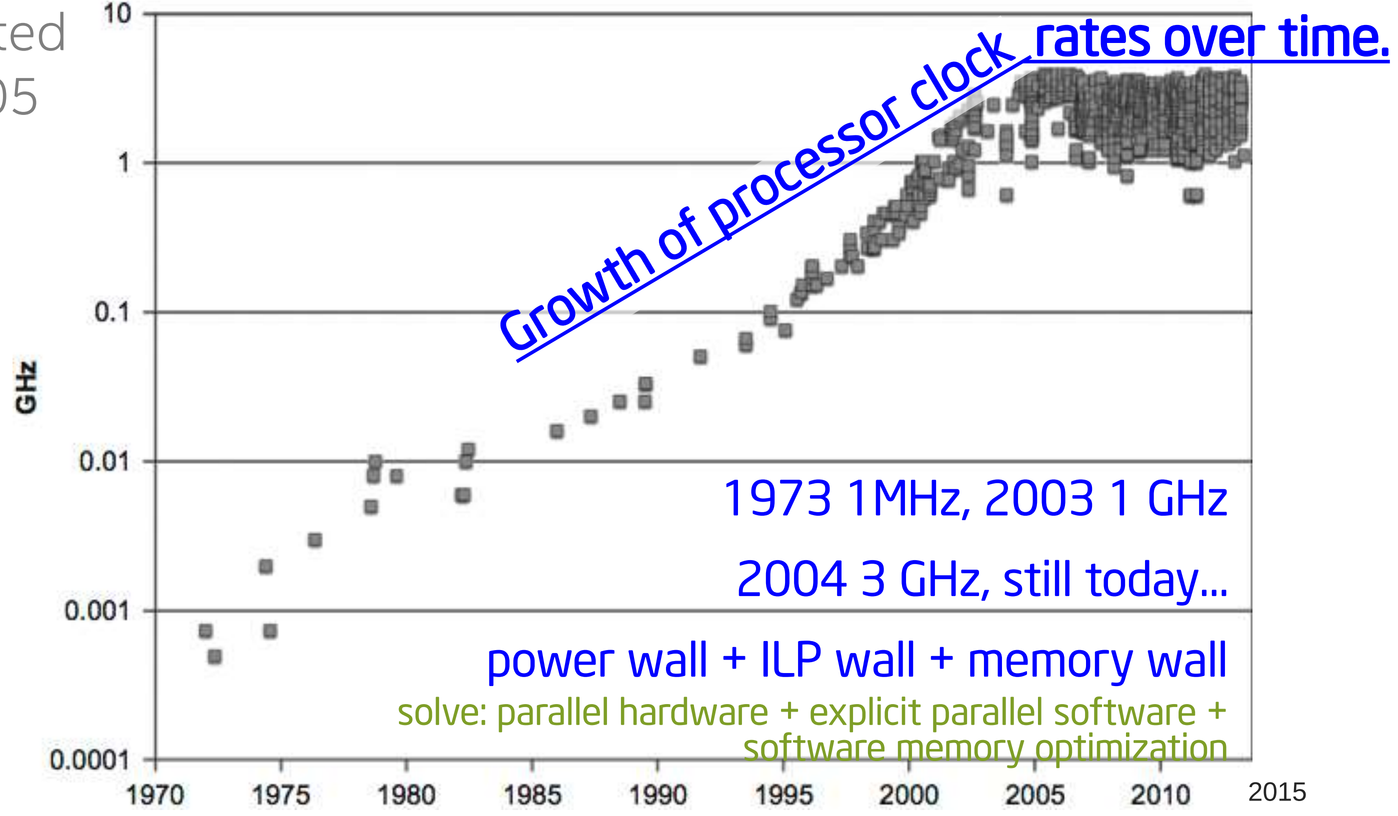
The “Free Lunch” is over, really.

But Moore’s Law continues!



Processor Clock Rate over Time

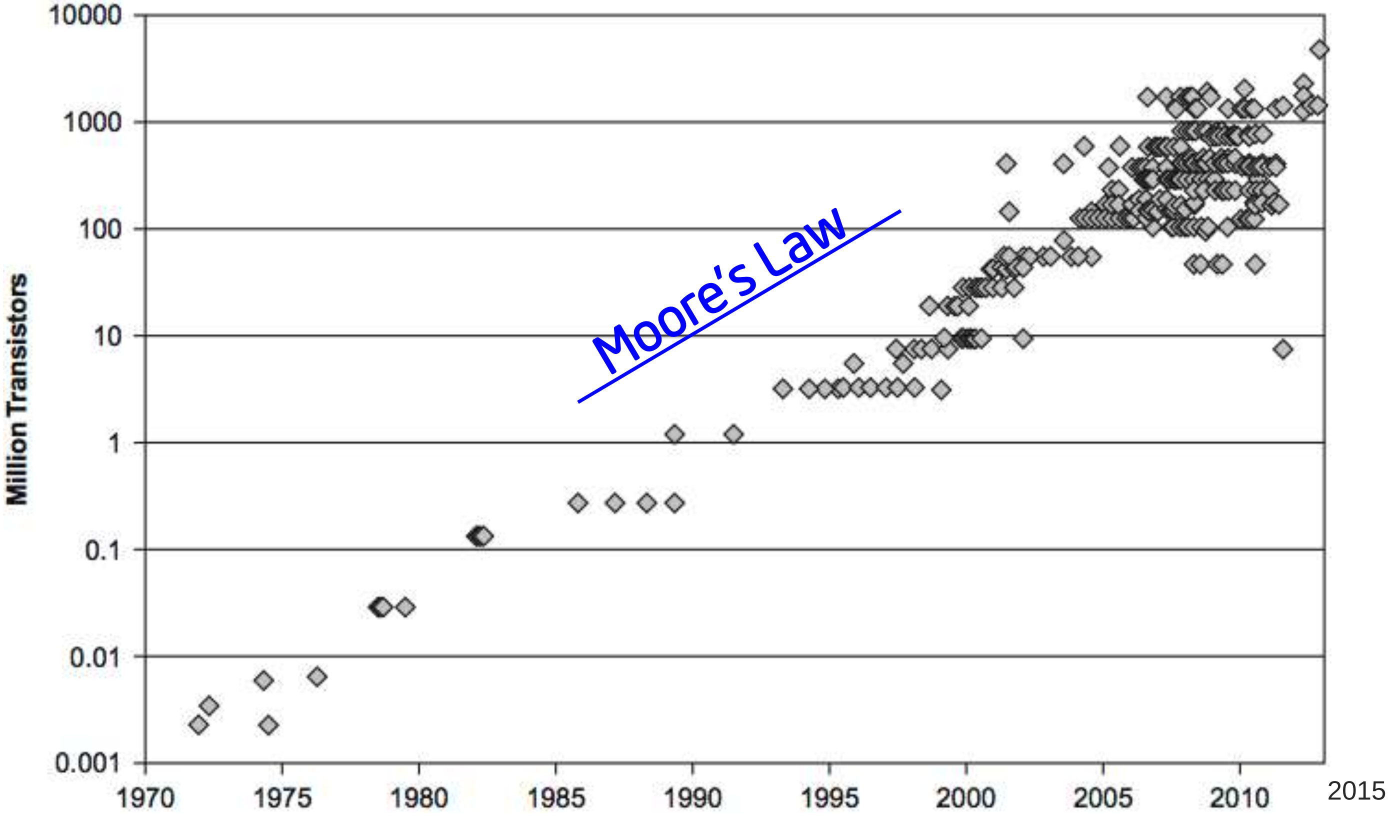
Growth halted around 2005



© 2015, James Reinders, used with permission. http://lotsofcores.com

Transistors per Processor over Time

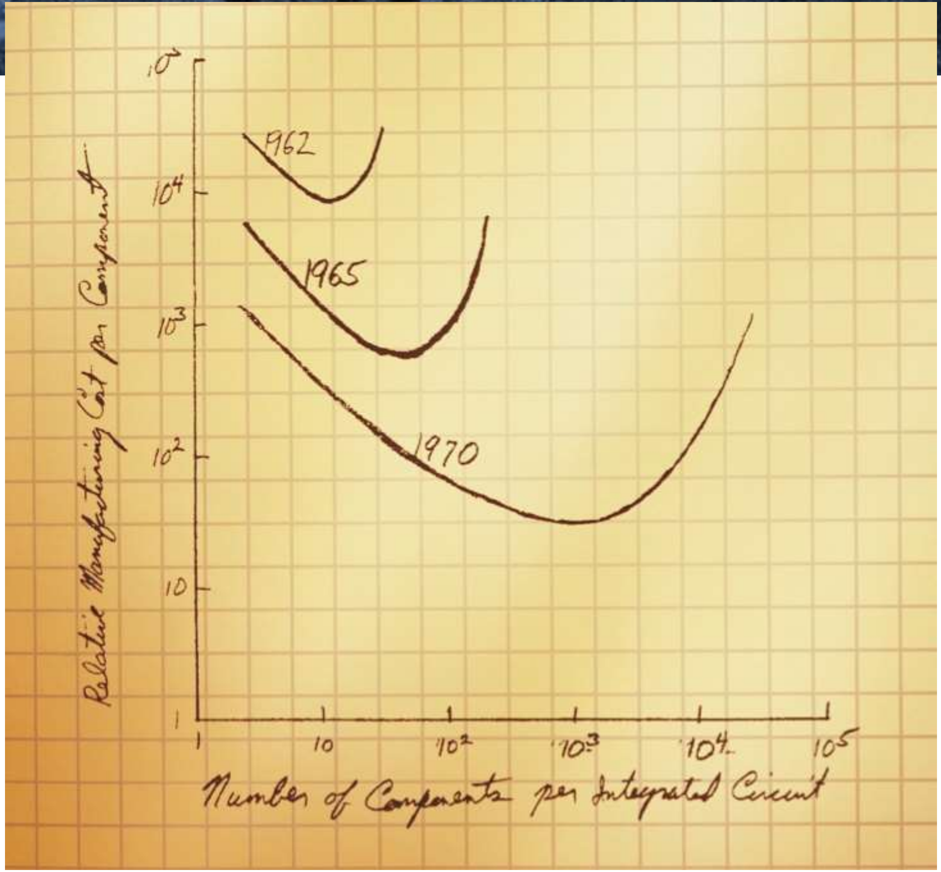
Continues to grow exponentially (Moore's Law)



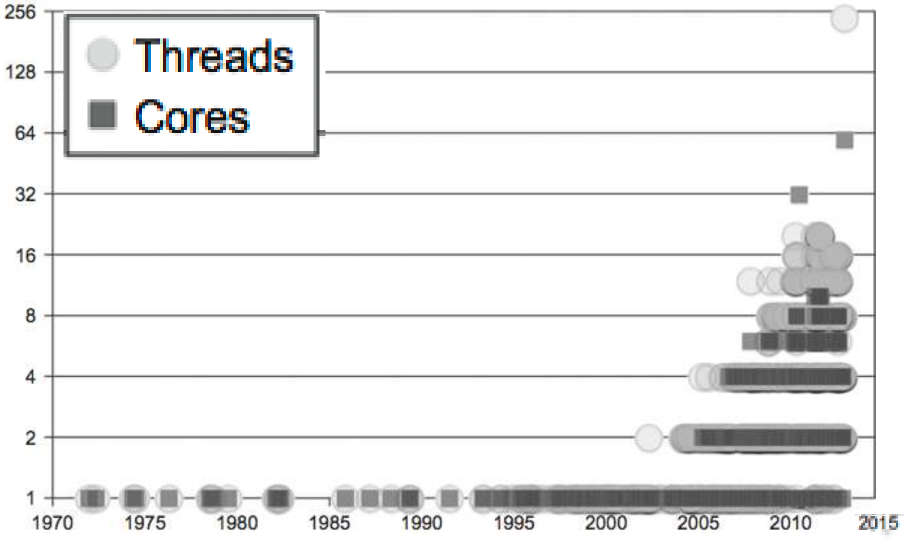
© 2015, James Reinders, used with permission. http://lotsofcores.com

Moore's Law

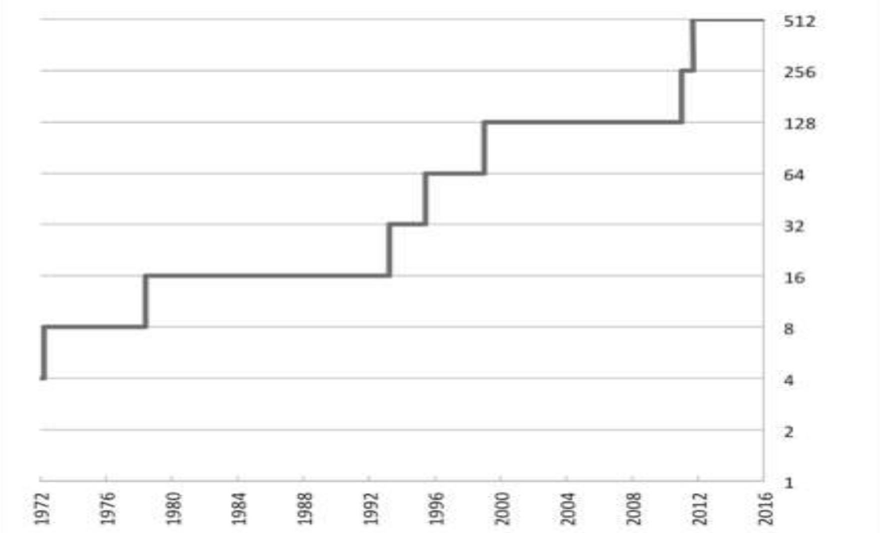
Number of components (transistors) doubles about every 18-24 months.



Core and Thread Counts



Width



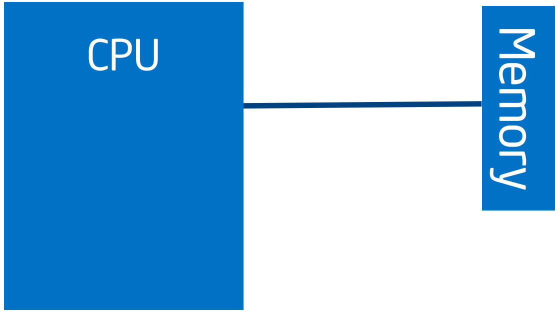
Single core, single thread, ruled for decades.
Multithread: grow die area small % for addition hardware thread(s) sharing resources.
Multicore/Many Core: 100% die area for additional hardware thread without sharing,

Data parallelism: handling more data at once, multibyte, multiword, many words.

© 2015, James Reinders, used with permission. http://lotsofcores.com

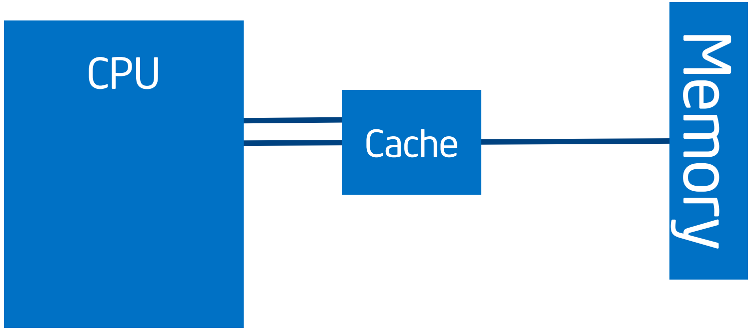
Is this the Architecture Track?

CPU



These were simpler times.

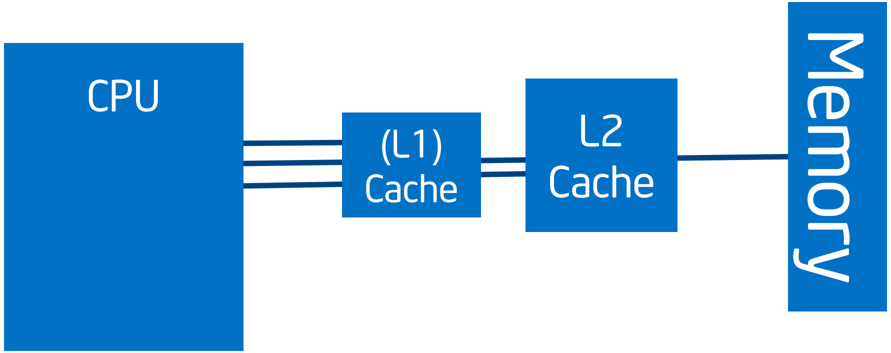
CPU + cache



Memories got "further away" (meaning: CPU speed increased faster than memory speeds)

A closer "cache" for frequently used data helps performance when memory is no longer a single clock cycle away.

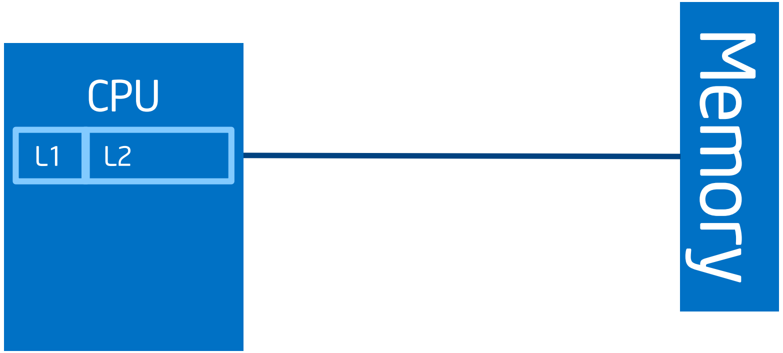
CPU + caches



Memories keep getting “further away” (this trend continues today).

More “caches” help even more (with temporal reuse of data).

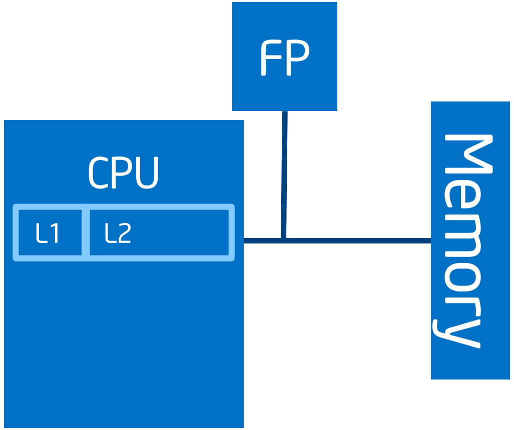
CPU with caches



As transistor density increased (Moore's Law), cache capabilities were integrated onto CPUs.

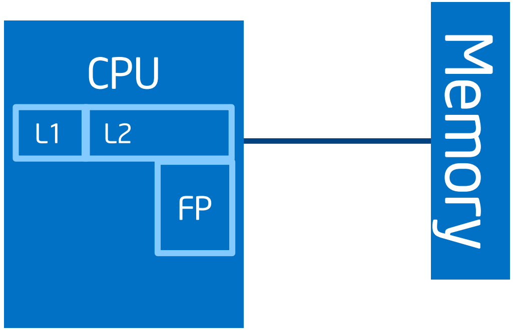
Higher performance external (discrete) caches persisted for some time while integrated cache capabilities increase.

CPU / Coprocessors



Coprocessors appearing first in 1970s were FP accelerators for CPUs without FP capabilities.

CPU / Coprocessors



As transistor density increased (Moore's Law), FP capabilities were integrated onto CPUs.

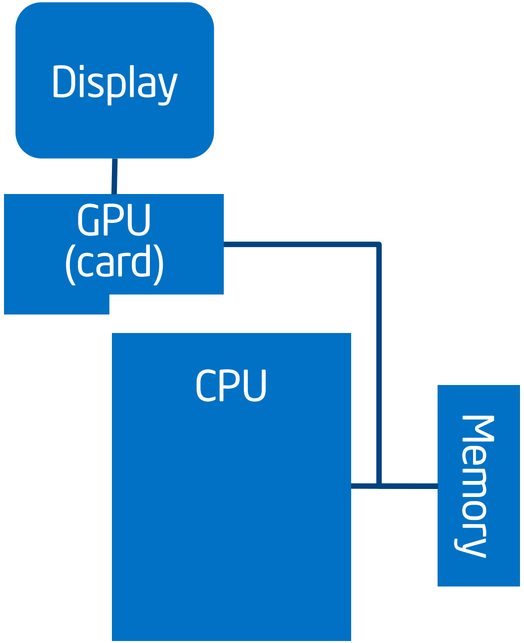
Higher performance discrete FP "accelerators" persisted a little bit while integrated FP capabilities increase.

CPU / Coprocessors

Interest to provide hardware support for displays increased as use of graphics grew (games being a key driver).

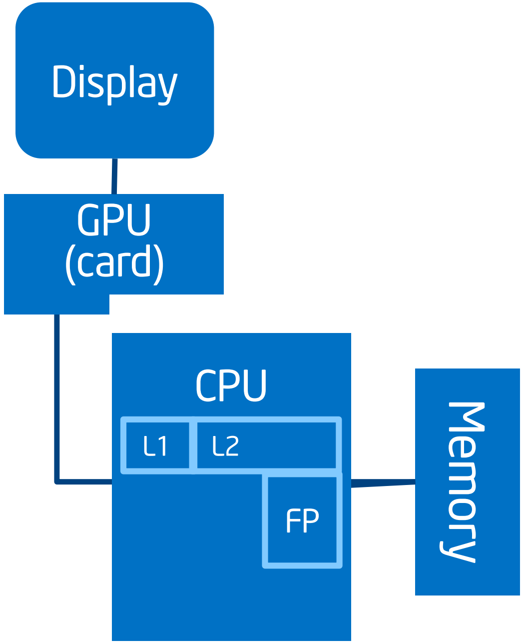
This led to graphics processing units (GPUs) attached to CPUs to create video displays.

Early Design



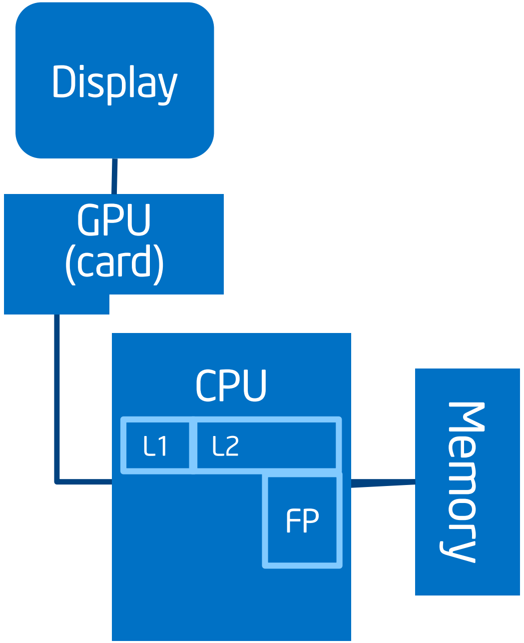
CPU / Coprocessors

GPU speeds and CPU speeds increase faster than memory speeds. Direct connection to memory best done via caches (on the CPU).



CPU / Coprocessors

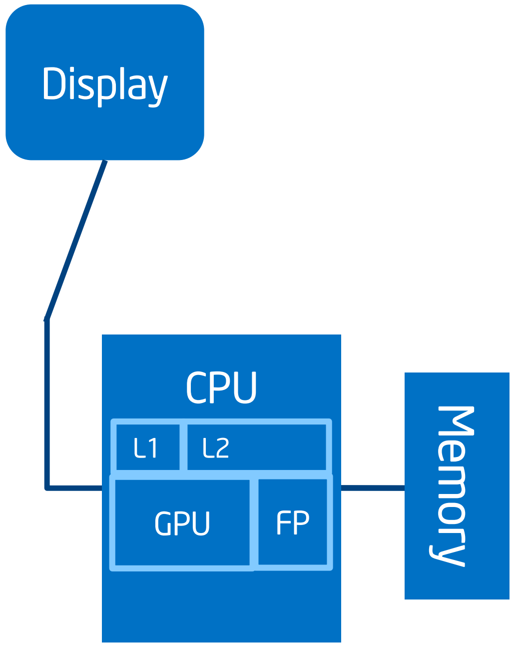
GPU speeds and CPU speeds increase faster than memory speeds. Direct connection to memory best done via caches (on the CPU).



CPU / Coprocessors

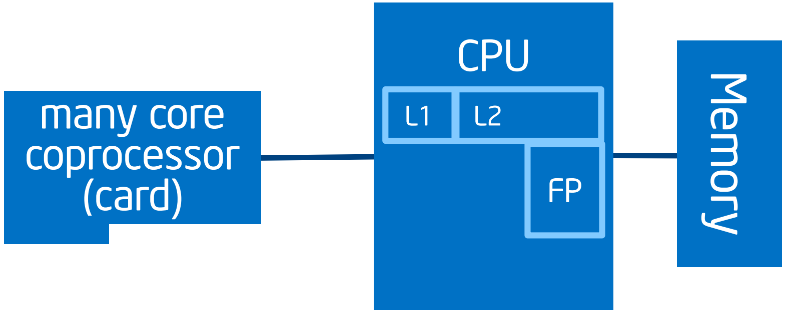
As transistor density increased (Moore's Law), GPU capabilities were integrated onto CPUs.

Higher performance external (discrete) GPUs persist while integrated GPU capabilities increase.



CPU / Coprocessors

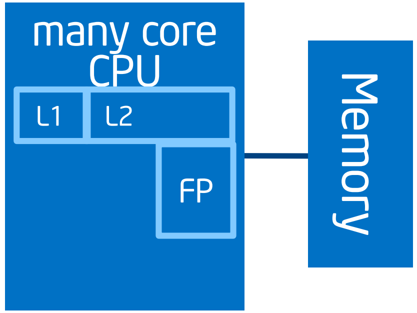
A *many core* coprocessor (Intel® Xeon Phi™) appears, purpose built for accelerating technical computing.



CPU / Coprocessors

As transistor density increased (Moore's Law), many core capabilities will be integrated to create a many core CPU.

("Knights Landing")

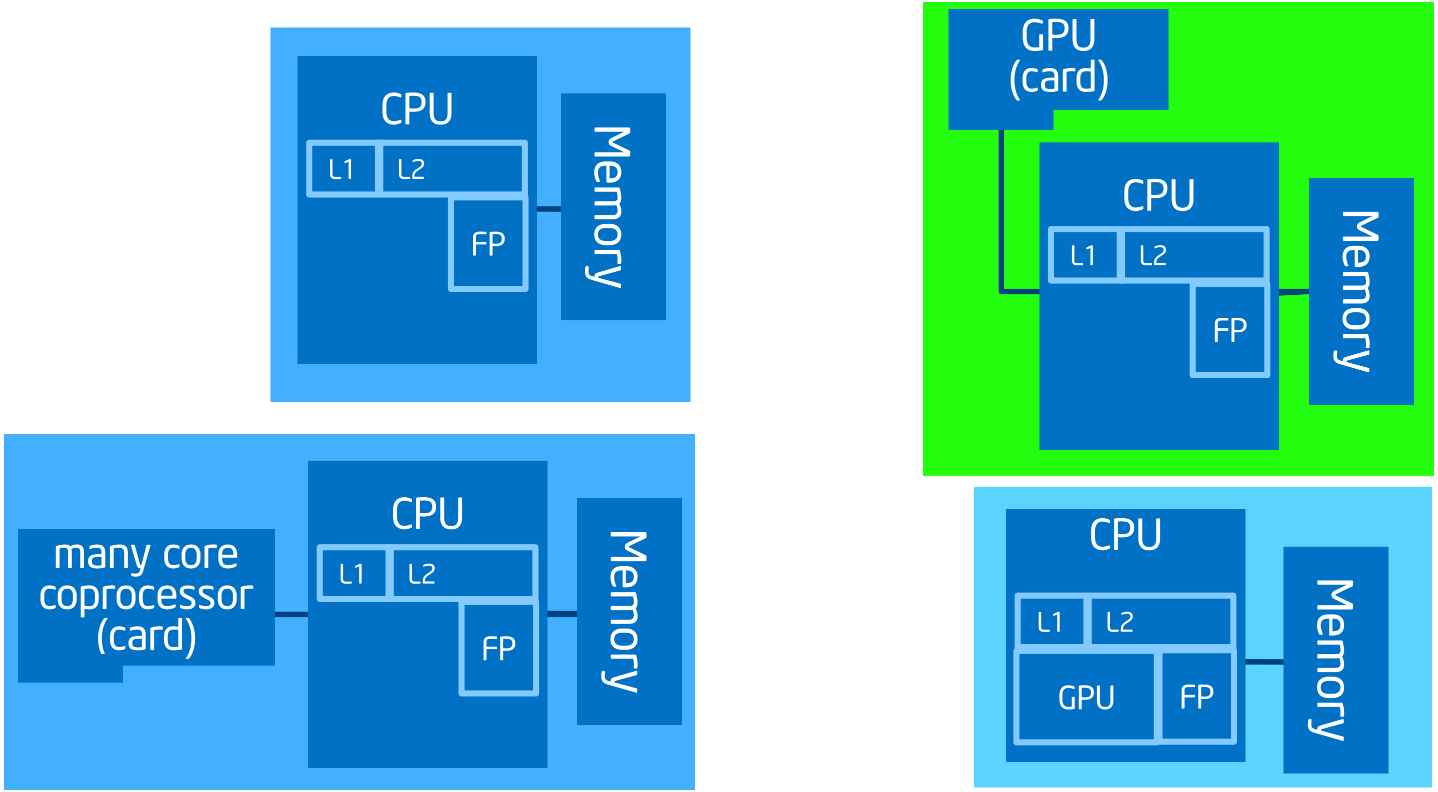


Nodes

"Nodes" are building blocks for clusters.

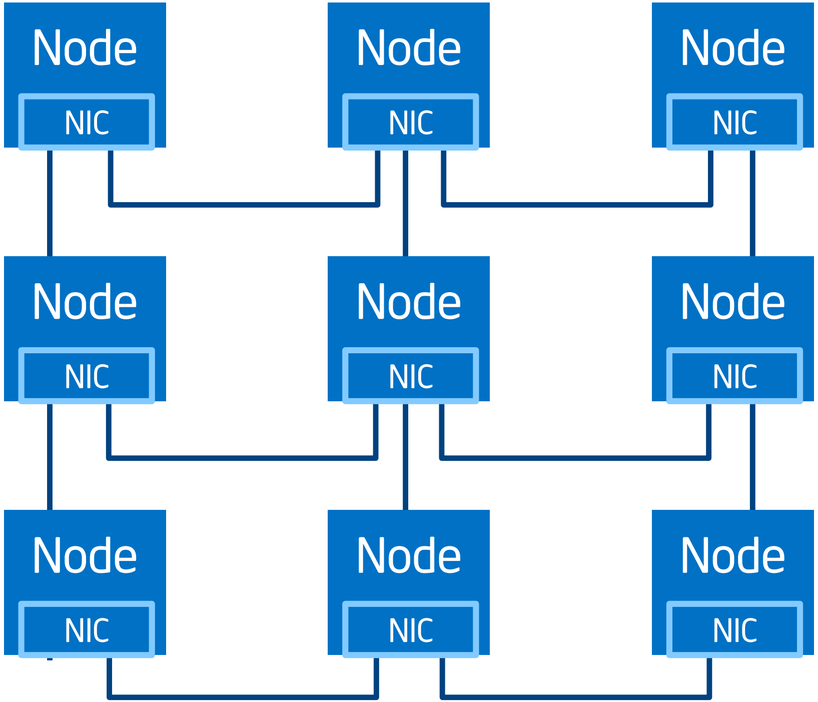
With or without GPUs.

Displays not needed.



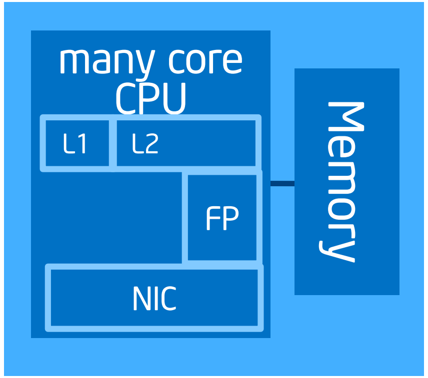
Clusters

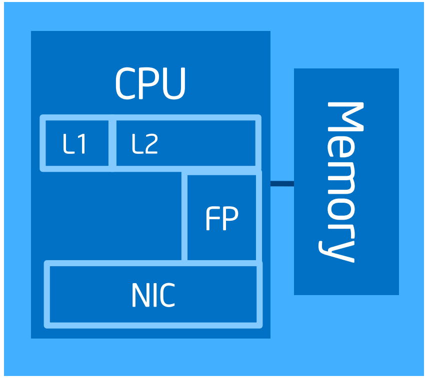
Clusters are made by connecting nodes - regardless of "Nodes" type.

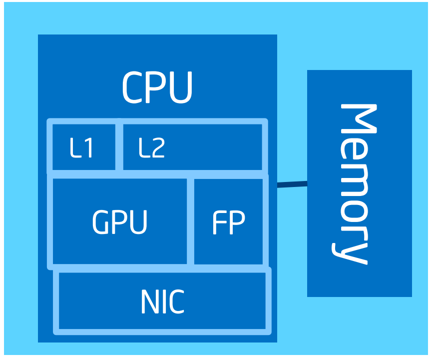


NIC (Network Interface Controller) integration

As transistor density increased (Moore's Law), NIC capabilities will be integrated onto CPUs.

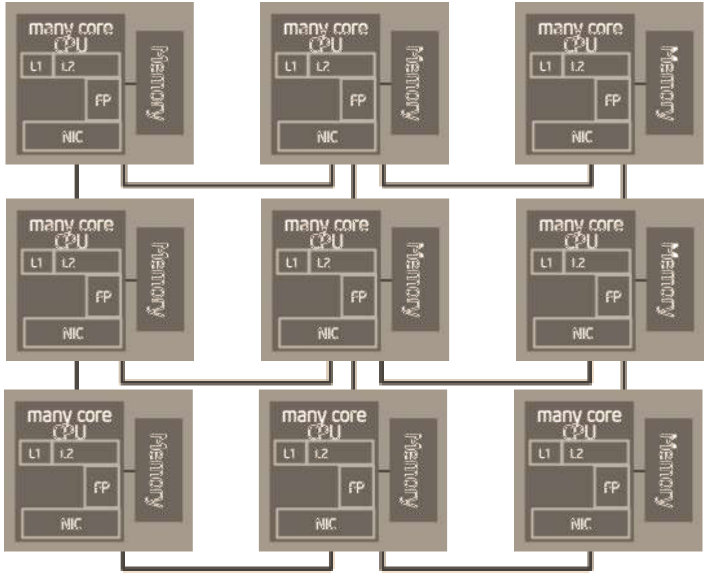






What matters when programming?

- Parallelism
- Locality



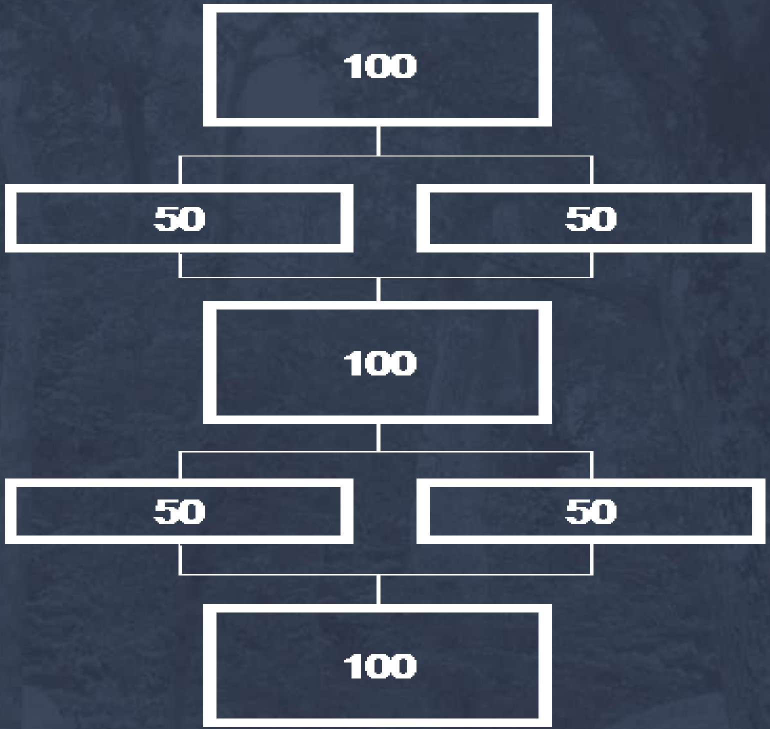
Amdahl who?

How much parallelism is there?

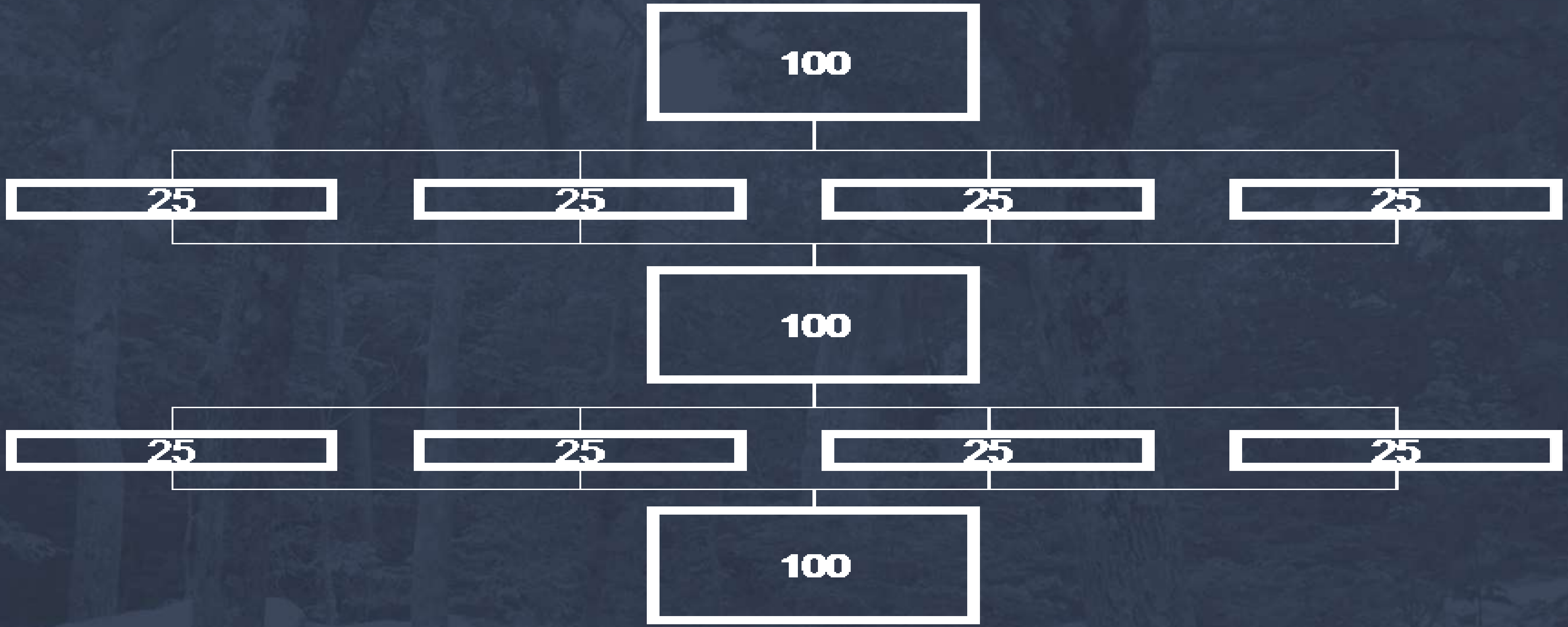
Amdahl's Law

Gustafson's observations on Amdahl's Law

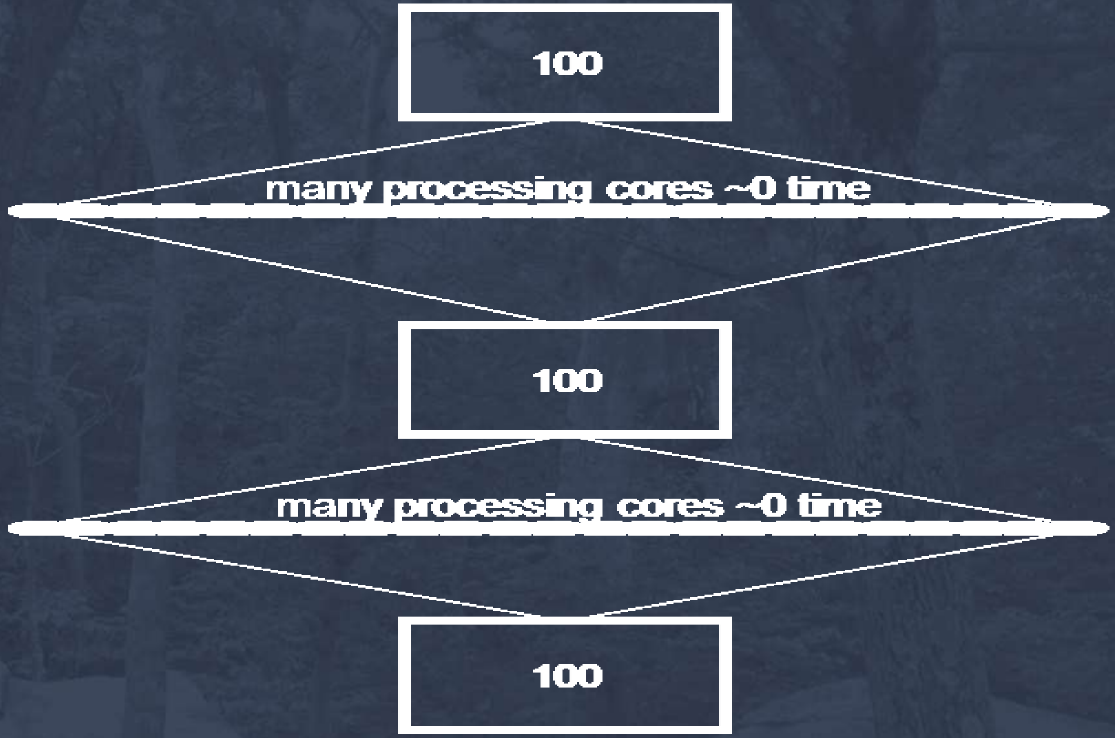
Work 500 Time 500
Speedup 1X



Work 500 Time 400
Speedup 1.25X



Work 500 Time 350
Speedup 1.4X



Work 500 Time 300
Speedup 1.7X

Amdahl's law

"…the effort expended on achieving high parallel processing rates is wasted unless it is accompanied by achievements in sequential processing rates of very nearly the same magnitude."

– Amdahl, 1967

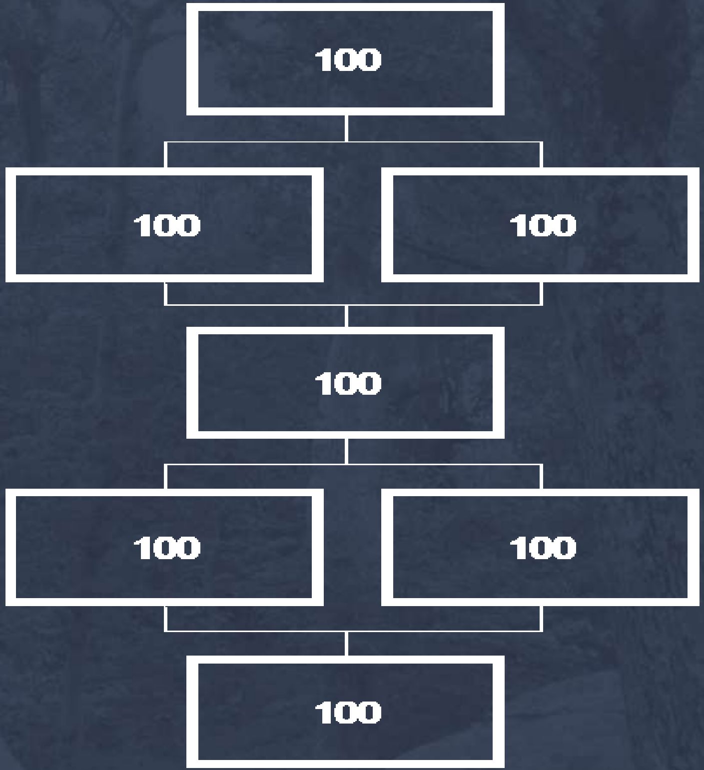
Amdahl's law – an observation

"…speedup should be measured by scaling the problem to the number of processors, not by fixing the problem size."

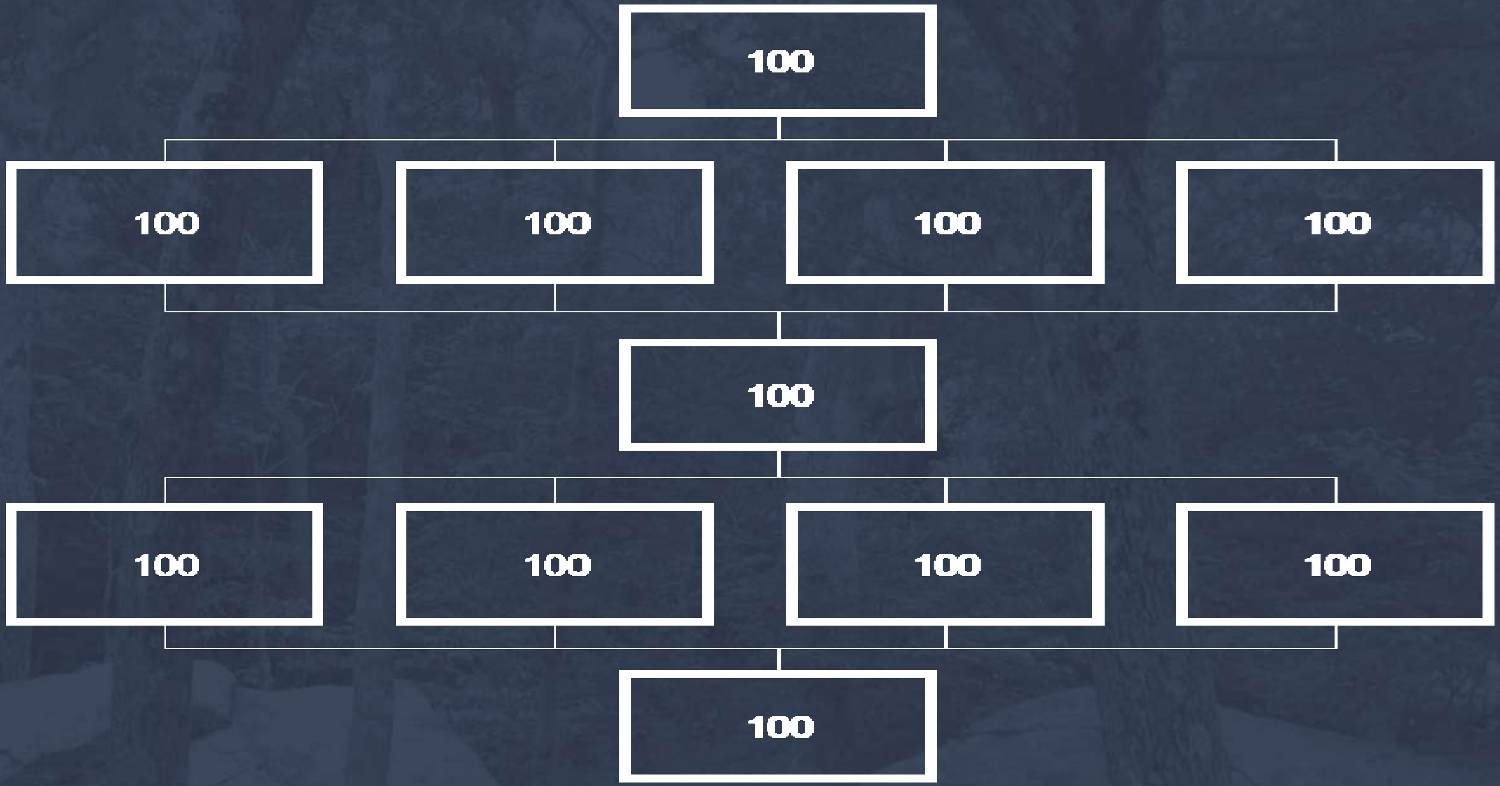
– Gustafson, 1988

Work 500 Time 500

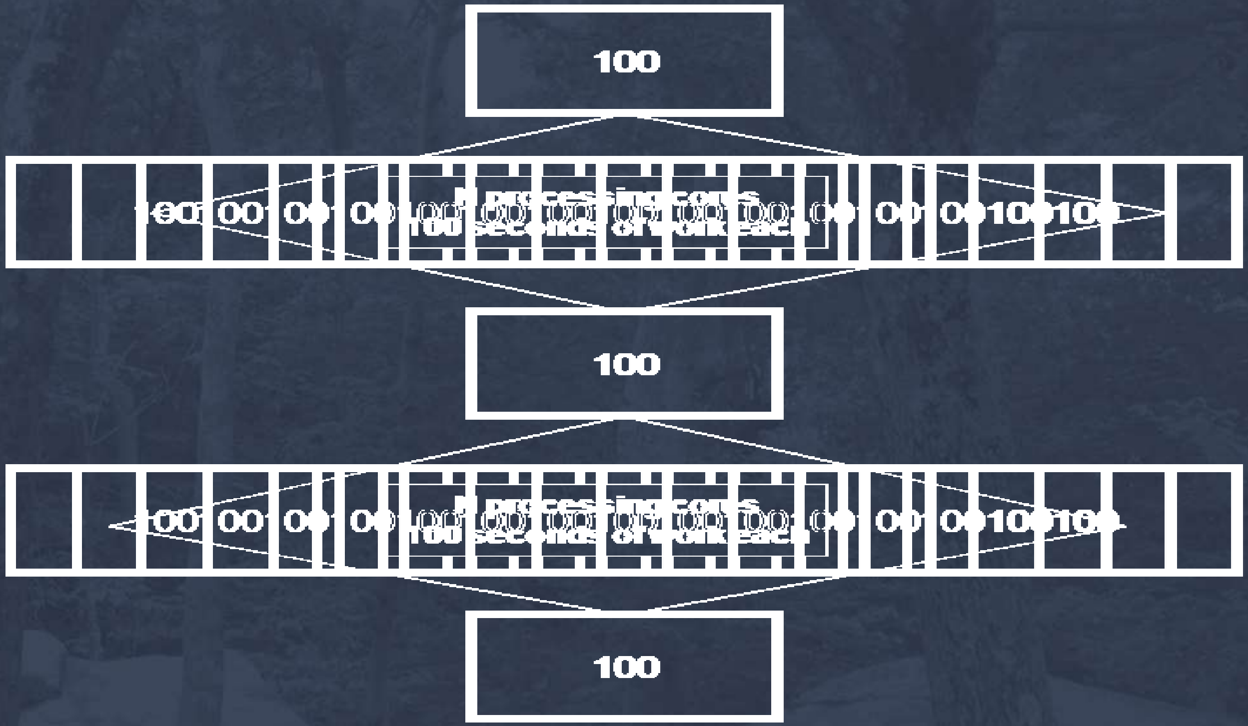
Speedup 1X



Work 700 Time 500
Speedup 1.4X



Work 1100 Time 500
Speedup 2.2X



Work 2*N*100+300 Time 500

Speedup O(N)

How much parallelism is there?

Amdahl's Law

Gustafson's observations on Amdahl's Law

Plenty –

but the workloads need to continue to grow !

Intel® Xeon Phi™ coprocessor

Twice: More than one sustained TeraFlop/sec

ASCI Red: 1 TeraFlop/sec
December 1996

1 TF/s
7264 Intel® Pentium Pro processors

72 Cabinets

Knights Corner
November 2011

1 TF/s
one chip

22nm

One PCI express slot

Source and Photo: http://en.wikipedia.org/wiki/ASCI_Red

Twice: More than one sustained TeraFlop/sec

More than *three* sustained TeraFlop/sec

ASCI Red: 1 TeraFlop/sec
December 1996

1 TF/s
7264 Intel® Pentium Pro processors

1999: upgraded to **3.1 TF/s** with
9298 Intel® Pentium II Xeon processors

72 Cabinets

Knights Corner
November 2011

1 TF/s
one chip

22nm

One PCI express slot

Source and Photo: http://en.wikipedia.org/wiki/ASCI_Red

Twice: More than one sustained TeraFlop/sec
Twice: More than *three* sustained TeraFlop/sec

ASCI Red: 1 TeraFlop/sec
December 1996

1 TF/s
7264 Intel® Pentium Pro processors

1999: upgraded to **3.1 TF/s** with
9298 Intel® Pentium II Xeon processors

72 Cabinets

Knights Corner
November 2011

1 TF/s
one chip

22nm

One PCI express slot

Knights Landing
2015

3 TF/s

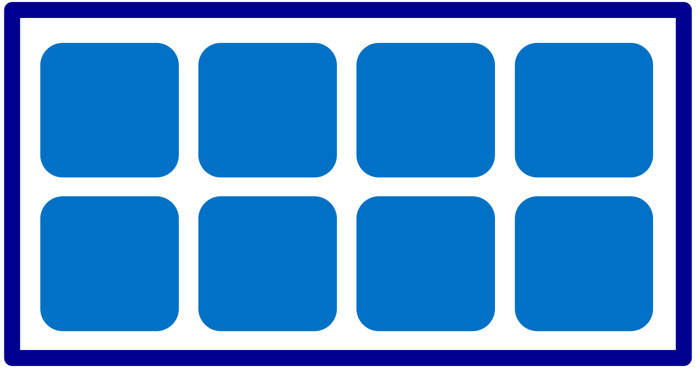
14nm

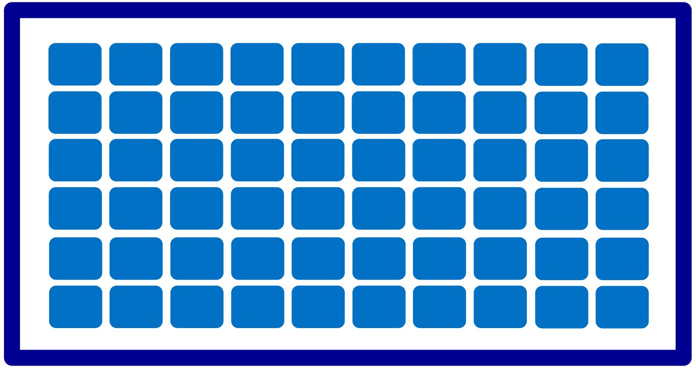
One Processor

Source and Photo: http://en.wikipedia.org/wiki/ASCI_Red

I want you to understand *why*

Design Question: Computation?

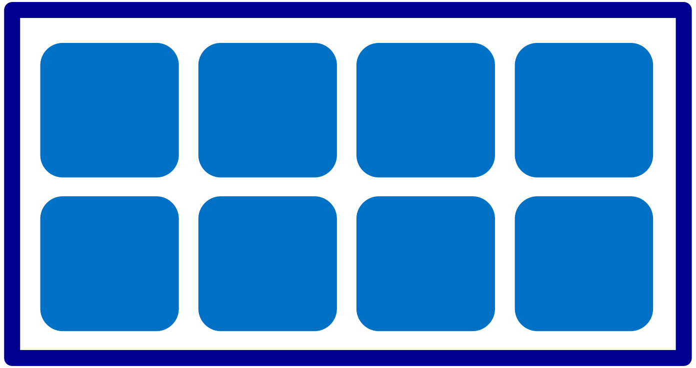


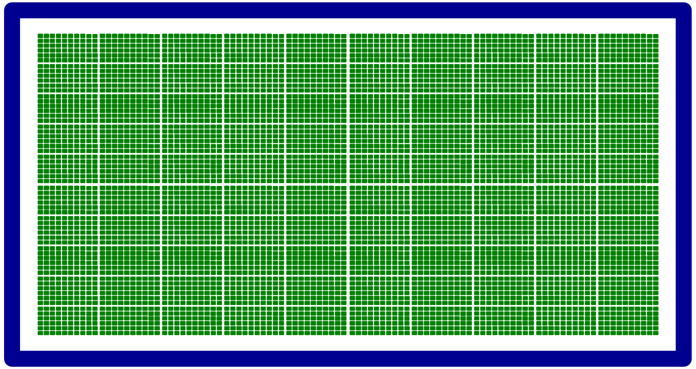


A few powerful vs. Many less powerful.

Diagrams for discussion purposes only, not a precise representation of any product of any company.

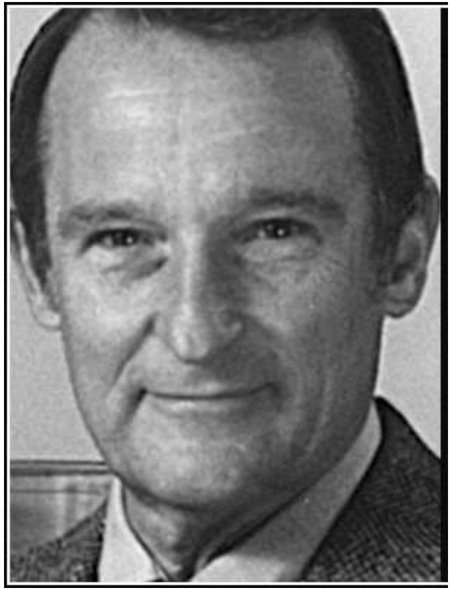
Design Question



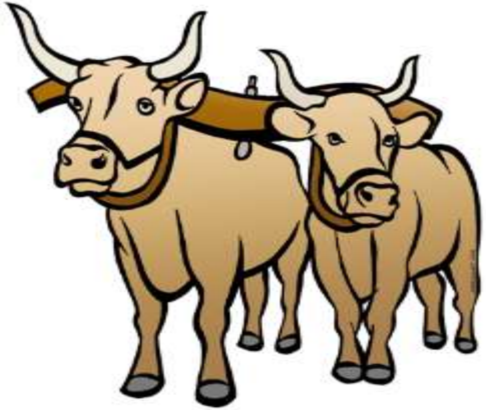


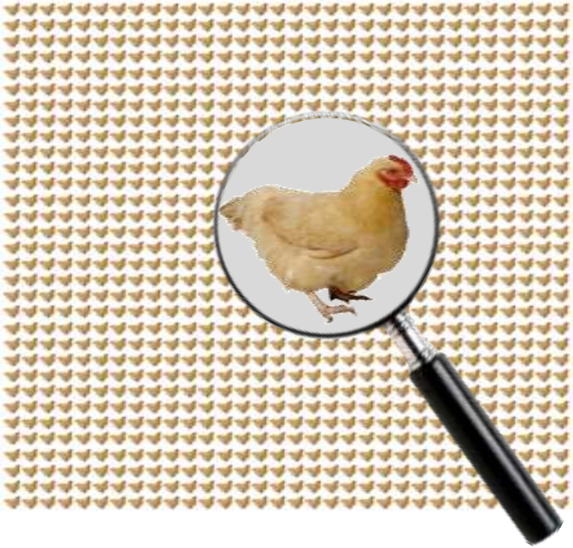
A few powerful vs. Many *much* less powerful *and very restrictive*.

Diagrams for discussion purposes only, not a precise representation of any product of any company.

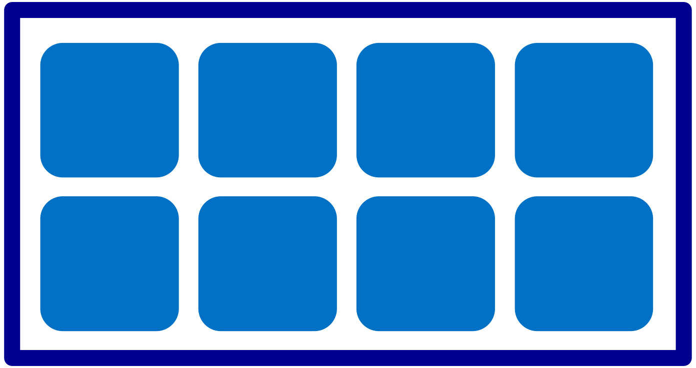


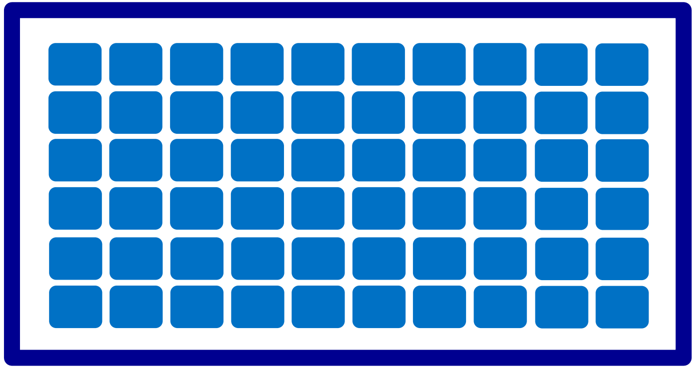
If you were plowing a field, which would you rather use… two strong oxen, or 1024 chickens?





Design Question



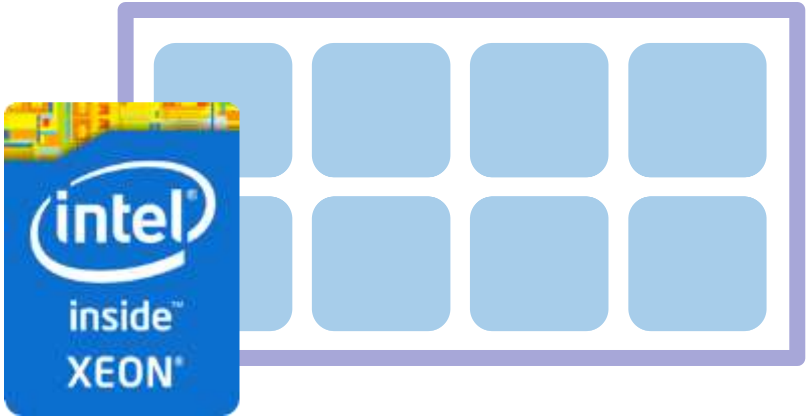


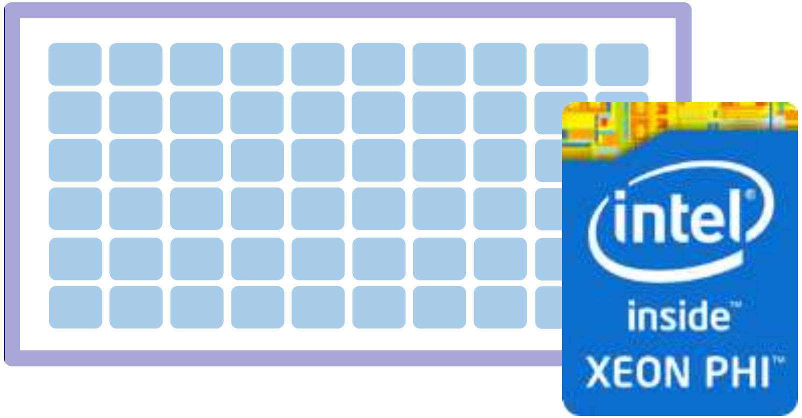
A few powerful vs. Many less powerful.

Same programming models, languages, optimizations and tools.

Diagrams for discussion purposes only, not a precise representation of any product of any company.

Design Question





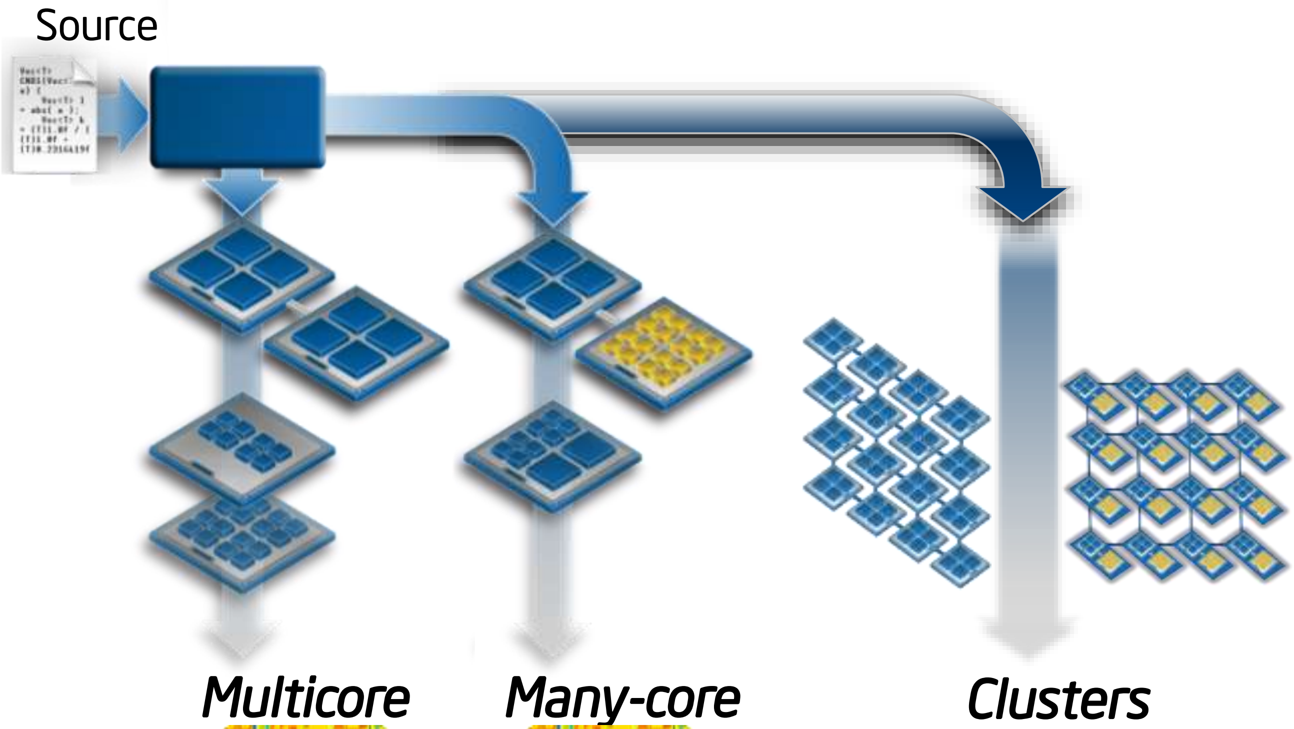
A few powerful vs. Many less powerful.

Same programming models, languages, optimizations and tools.

Diagrams for discussion purposes only, not a precise representation of any product of any company.

vision

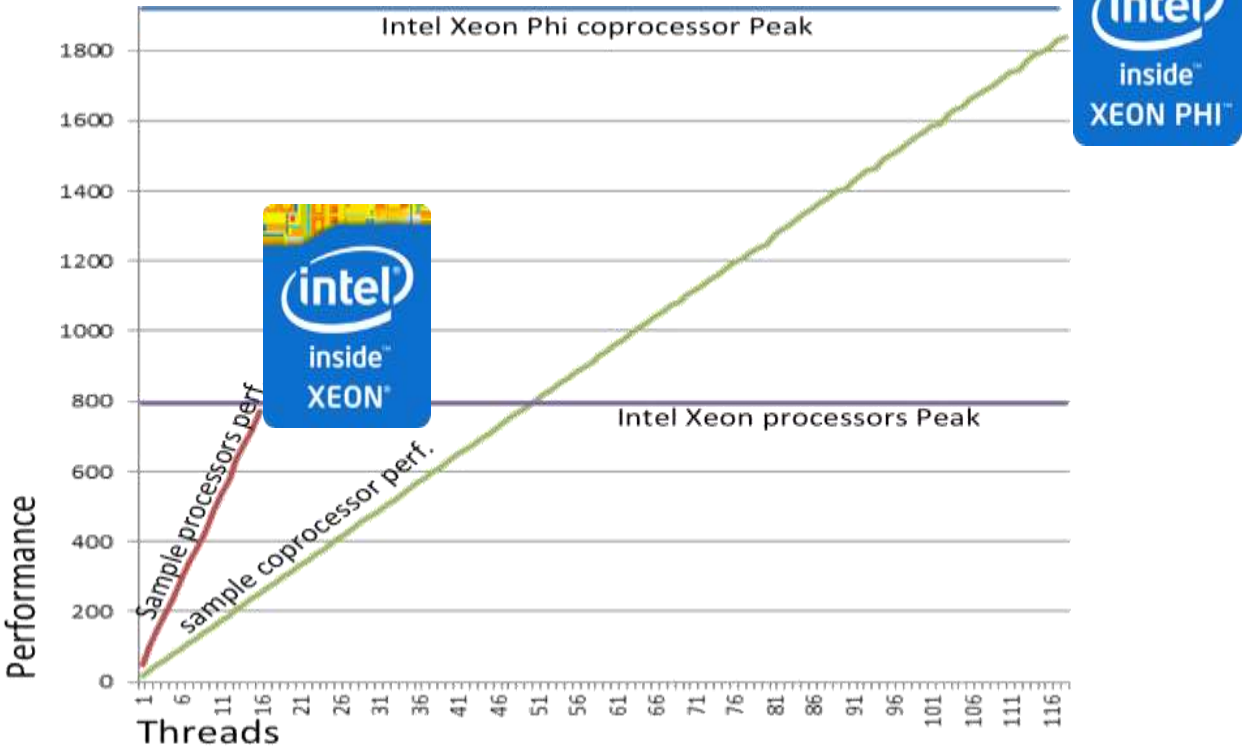
span from *few cores* to *many cores* with consistent models, languages, tools, and techniques



Optimizations for Intel® Xeon® and Intel® Xeon Phi™ products share the same:

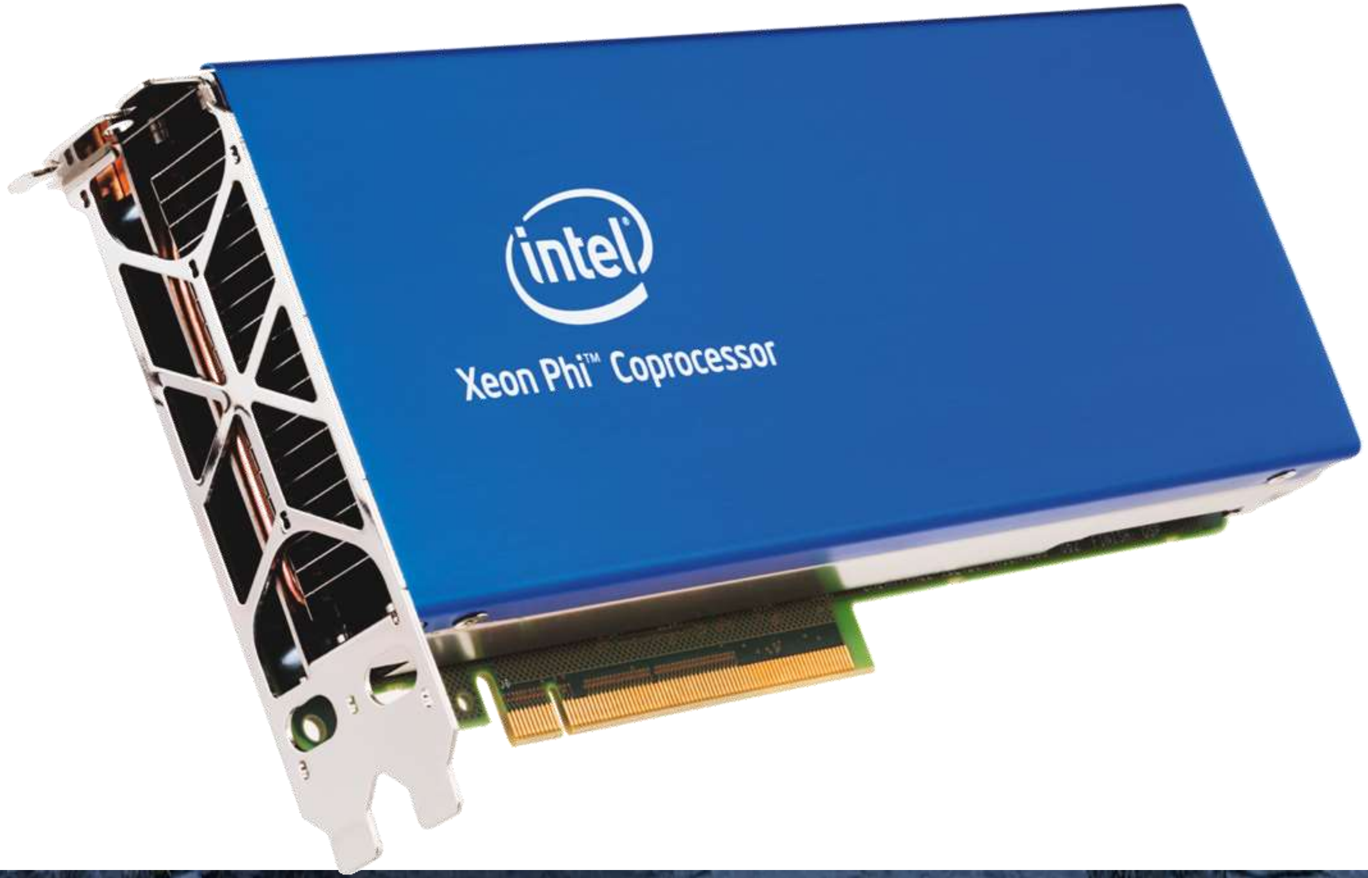
- ✓ Languages
- ✓ Directives
- ✓ Libraries
- ✓ Tools

Picture worth many words



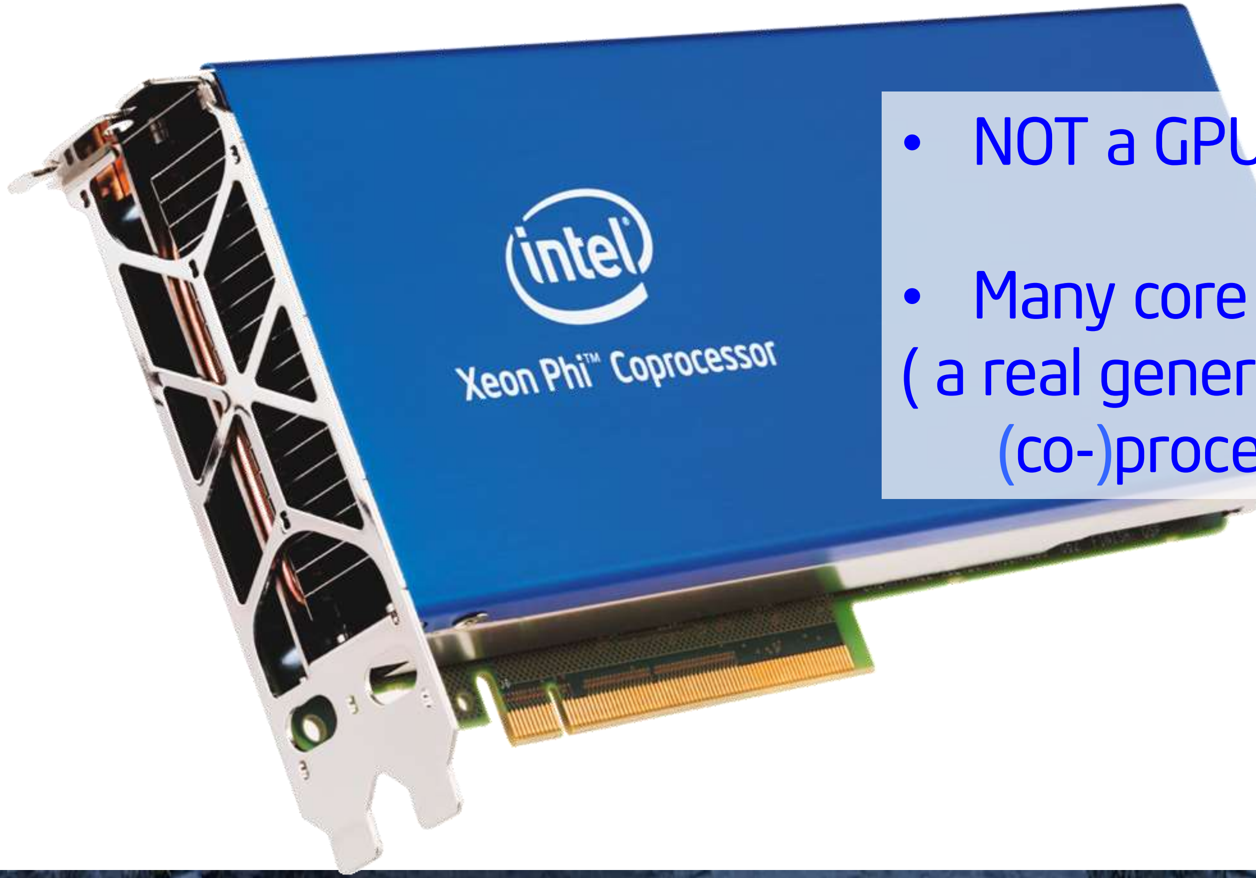
© 2013, James Reinders & Jim Jeffers, diagram used with permission

Intel® Xeon Phi™ Coprocessor



intel
inside™
XEON PHI™

Intel® Xeon Phi™ Coprocessor



- NOT a GPU
- Many core x86

(a real general purpose (co-)processor!)

Intel® Xeon Phi™ Coprocessors

Up to 61 cores, 1.1 GHz, 244 threads.

Up to 16GB memory.

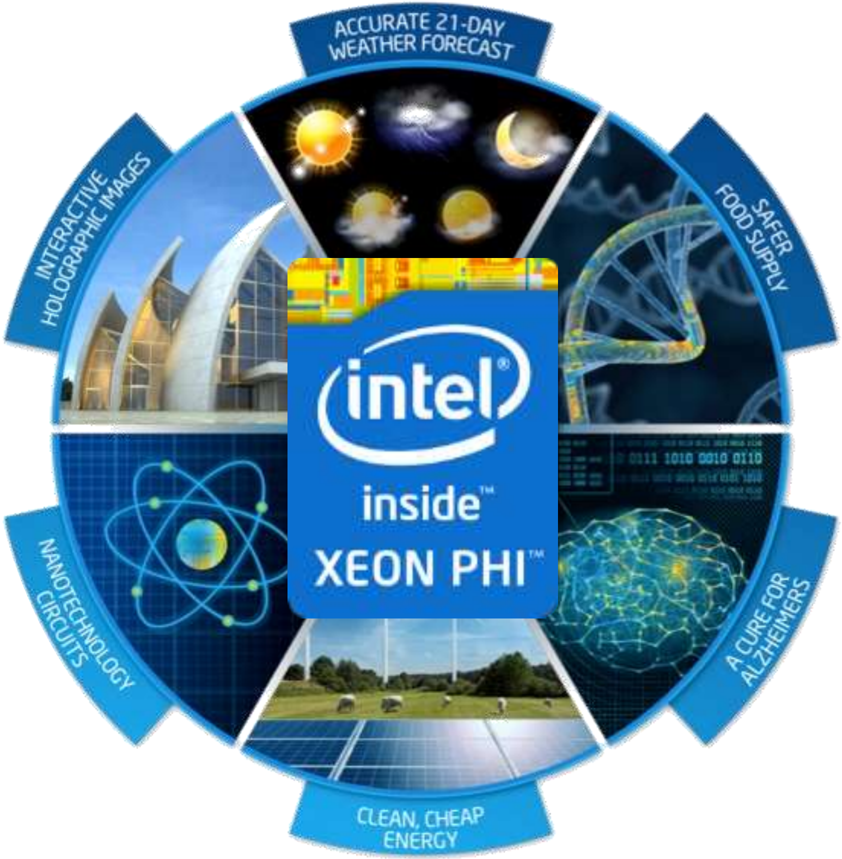
Up to 352 GB/s bandwidth.

Runs Linux OS.

Standard tools, models, languages.

1 TFLOP/s DP FP peak.

Better for parallelism than processor...
Up to 2.2X performance
Up to 4X more power efficient



it is an SMP-on-a-chip running Linux

```
% cat /proc/cpuinfo | head -5
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 11
model           : 1
model name      : 0b/01
%
% cat /proc/cpuinfo | tail -26

processor       : 243
vendor_id       : GenuineIntel
cpu family      : 11
model           : 1
model name      : 0b/01
stepping        : 1
cpu MHz         : 1090.908
cache size      : 512 KB
physical id     : 0
siblings        : 244
core id         : 60
cpu cores       : 61
apicid          : 243
initial apicid  : 243
fpu             : yes
fpu_exception   : yes
cpuid level     : 4
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic mtrr mca pat fxsr ht syscall lm lahf_lm
bogomips        : 2192.10
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

%
```

Next Intel® Xeon Phi™ Processor:
Knights Landing

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

14nm process

standalone standalone CPU or PCIe coprocessor

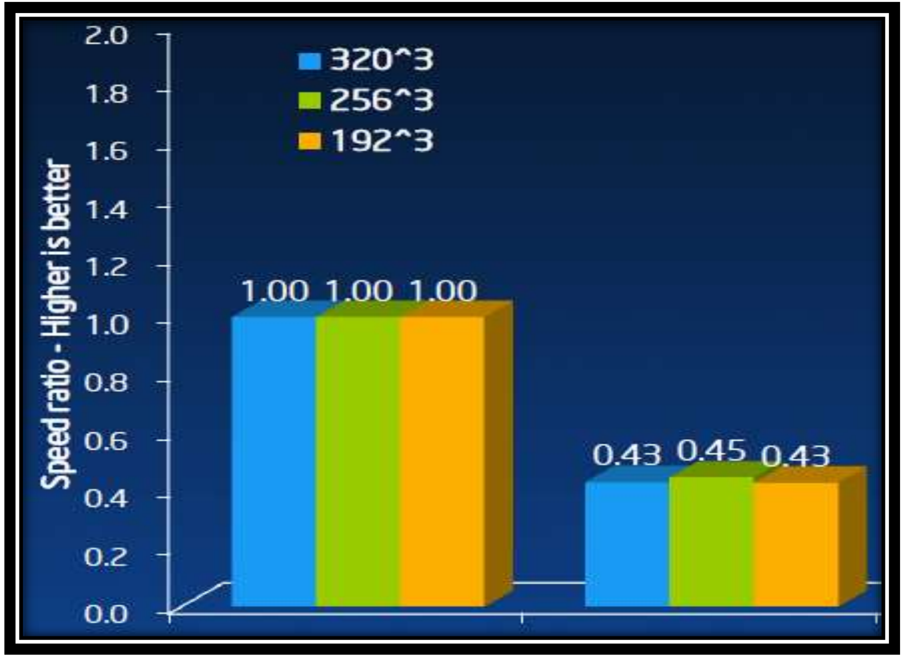
up to 72 cores (“Silvermont” based)

AVX-512 SIMD

on-package high-bandwidth memory

on-package NIC (optional)

Illustrative example



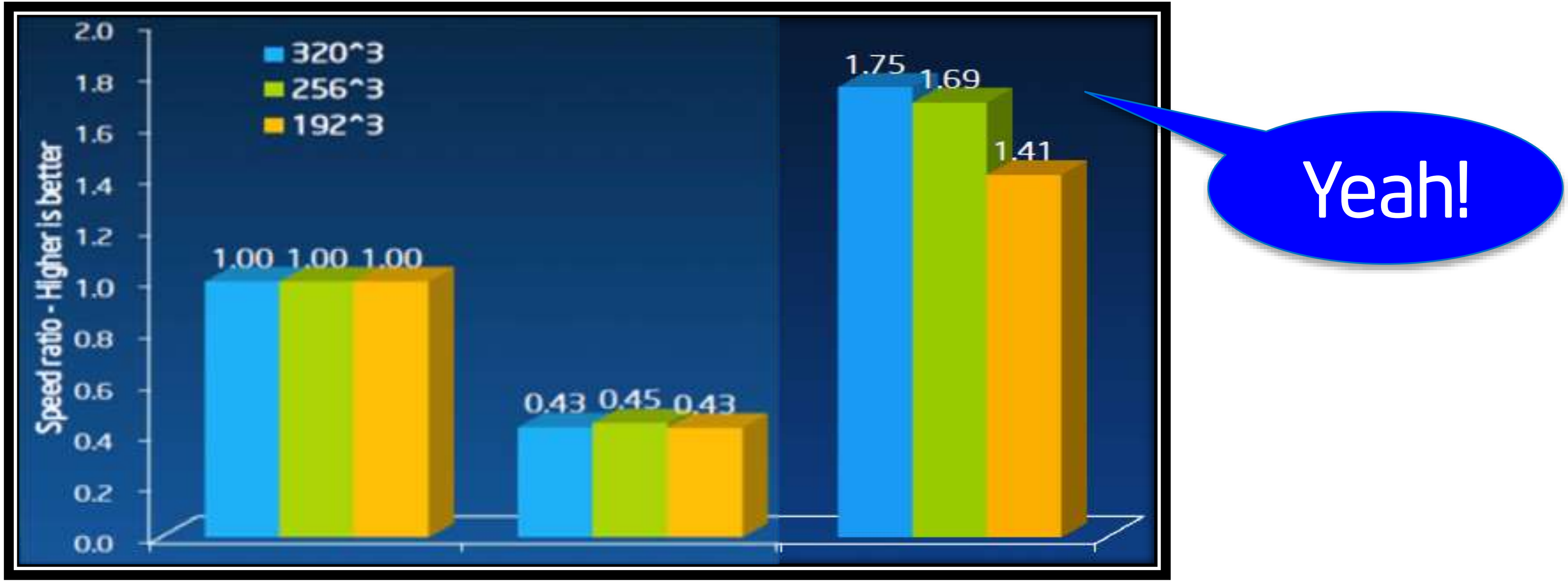
Untuned Performance on Intel® Xeon® processor

Untuned Performance on Intel® Xeon Phi™ coprocessor

Based on an actual customer example.
Shown to illustrate a point about common techniques.
Your results may vary!

Fortran code using MPI, single threaded originally.
Run on Intel® Xeon Phi™ coprocessor natively (no offload).

Illustrative example

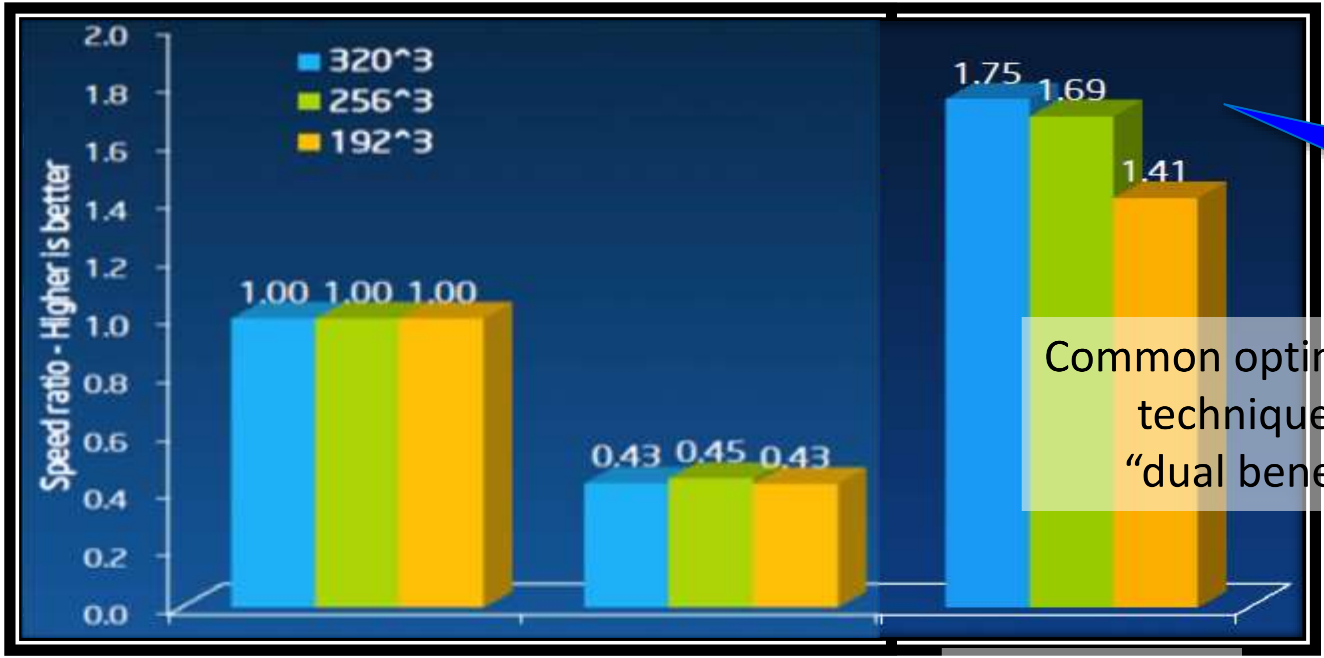


Untuned Performance on Intel® Xeon® processor

Untuned Performance on Intel® Xeon Phi™ coprocessor

TUNED Performance on Intel® Xeon Phi™ coprocessor

Illustrative example



Yeah!

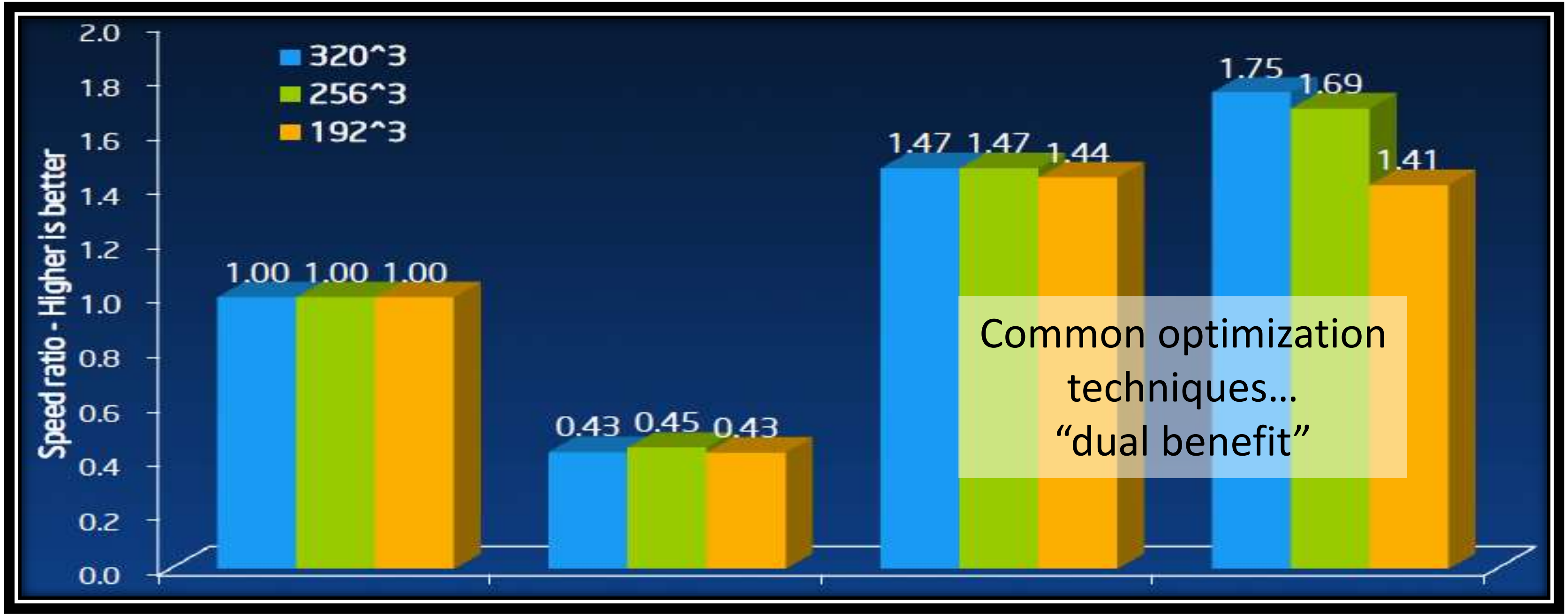
Common optimization techniques… "dual benefit"

Untuned Performance on Intel® Xeon® processor

Untuned Performance on Intel® Xeon Phi™ coprocessor

TUNED Performance on Intel® Xeon Phi™ coprocessor

Illustrative example

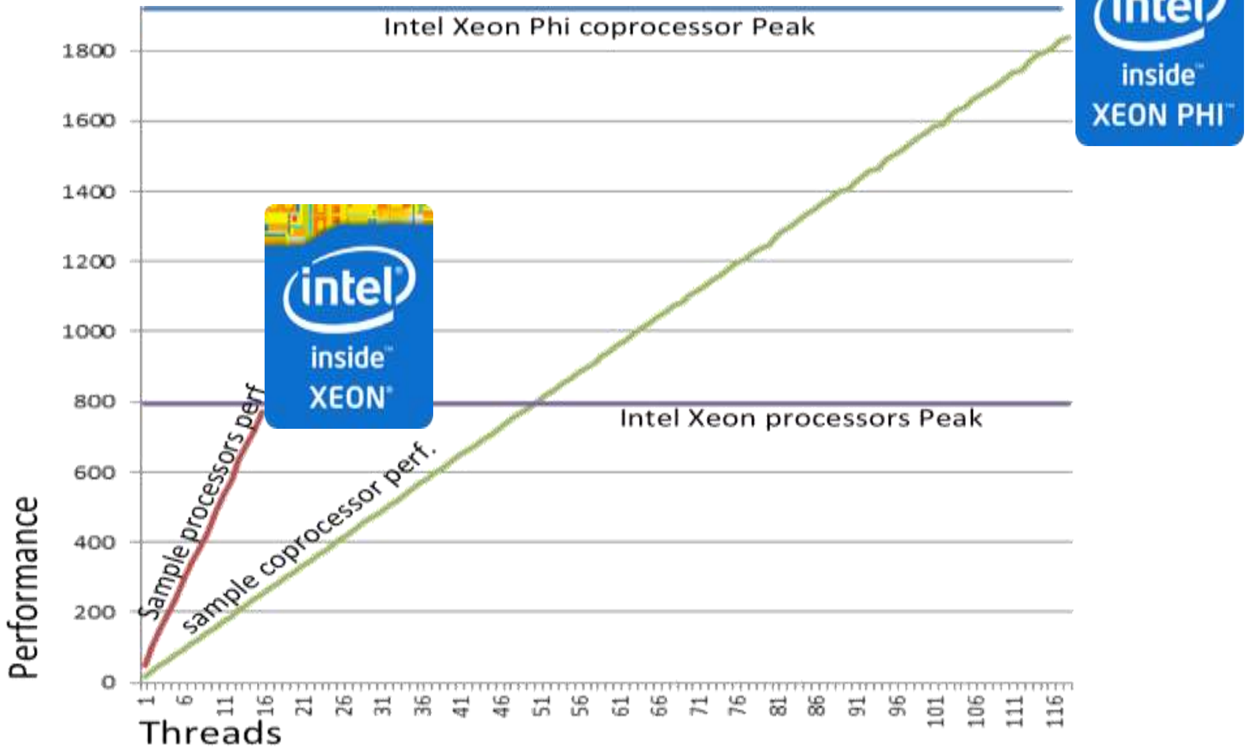


Untuned Performance on Intel® Xeon® processor	Untuned Performance on Intel® Xeon Phi™ coprocessor	TUNED Performance on Intel® Xeon® processor	TUNED Performance on Intel® Xeon Phi™ coprocessor

Processors and Intel® Xeon Phi™ products

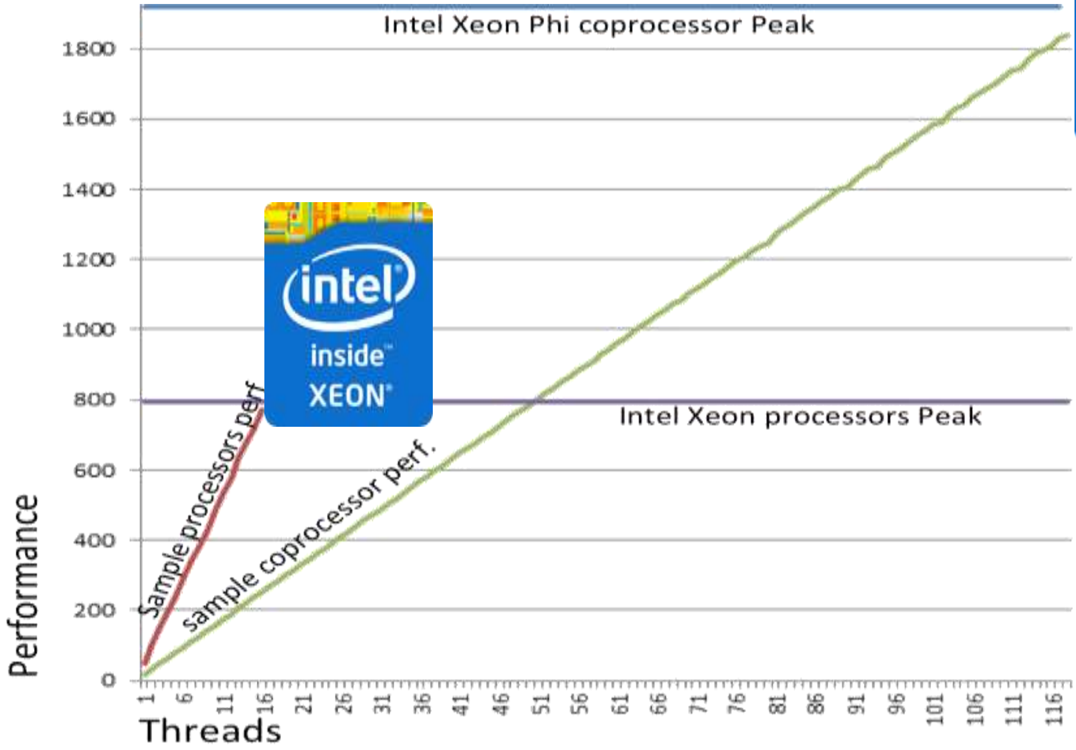
span from few cores to many cores with consistent models, languages, tools, and techniques

Picture worth many words



© 2013, James Reinders & Jim Jeffers, diagram used with permission

Picture worth many words



© 2013, James Reinders & Jim Jeffers, diagram used with permission

GPUs,
FPGAs,
and probably other
"accelerators"
will still offer "alternatives."

They all start by trading off flexibility (generality).

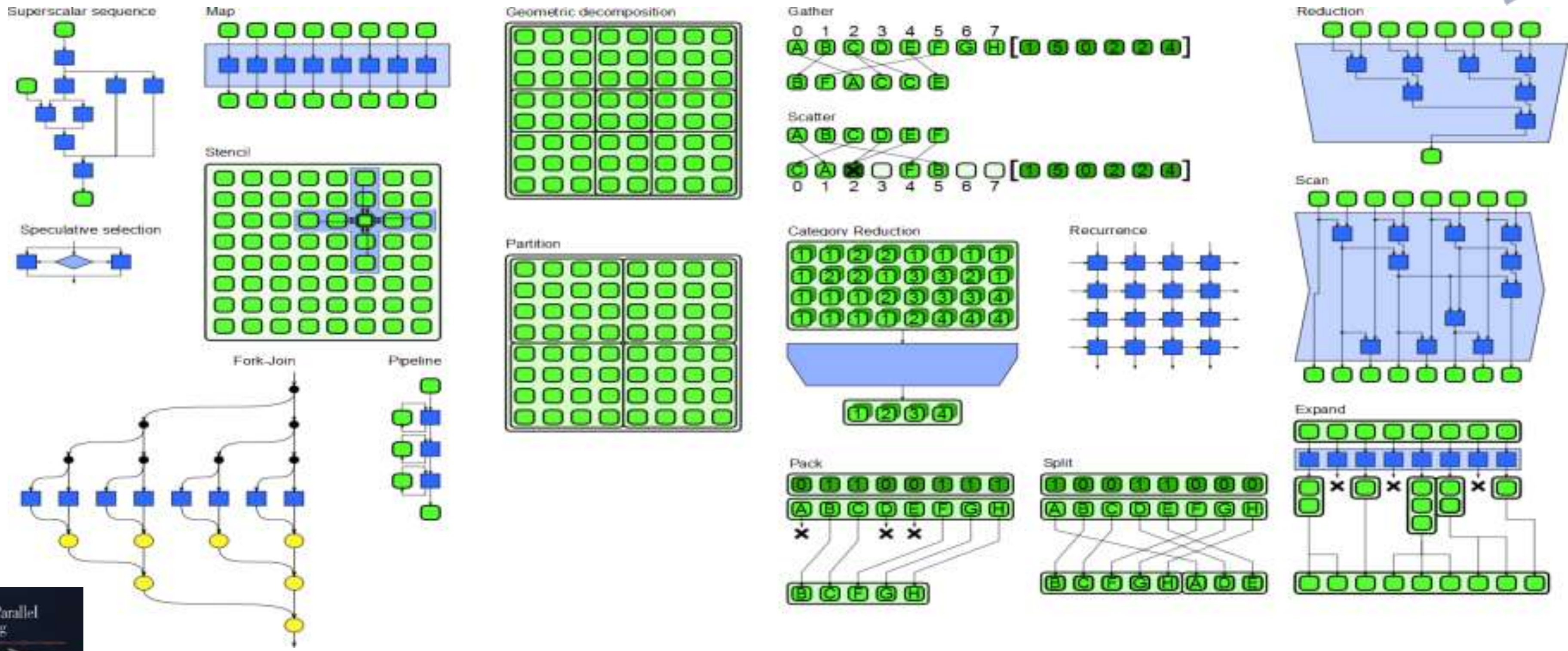
The question to ask is
"is it worth it?"

The answer is "sometimes."

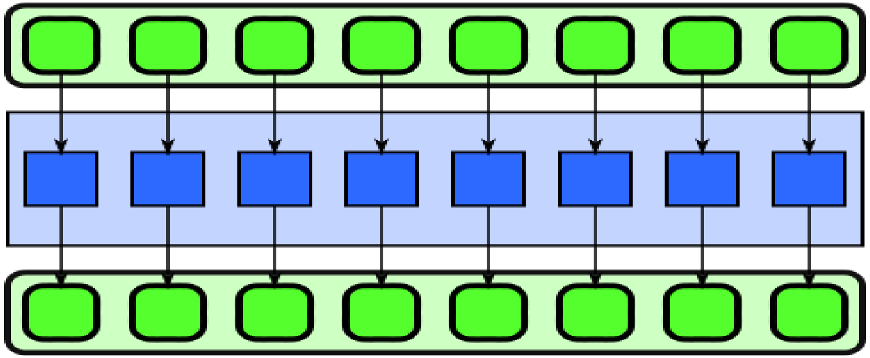
James' assertion:
Non-general solutions will give way to general solutions when we have them.
(if ?)

How do I "think parallel" ?

Parallel Patterns: Overview



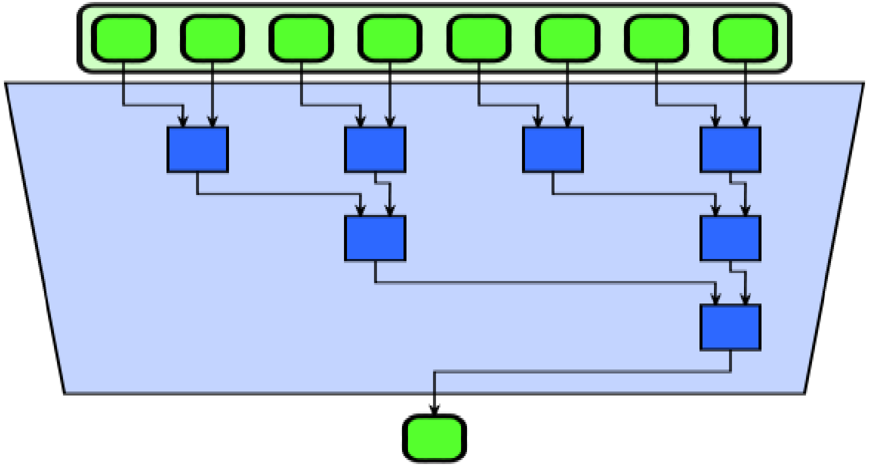
Map



Examples: gamma correction and thresholding in images; color space conversions; Monte Carlo sampling; ray tracing.

- *Map* invokes a function on every element of an index set.
- The index set may be abstract or associated with the elements of an array.
- Corresponds to "parallel loop" where iterations are independent.

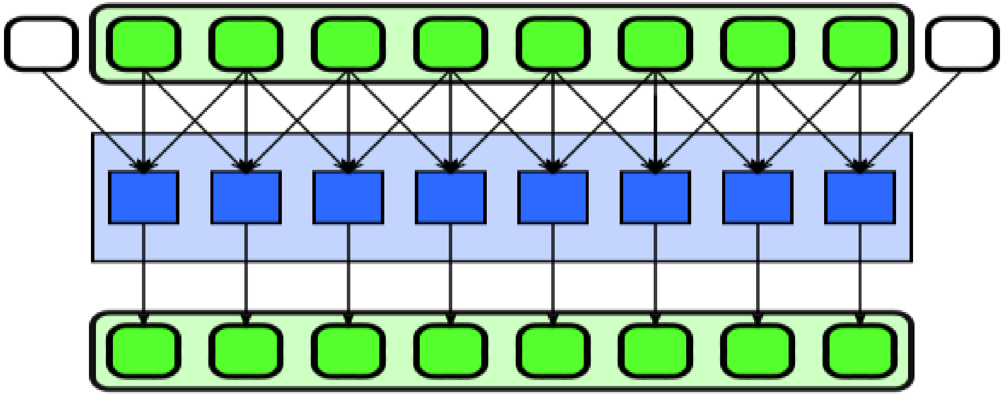
Reduce



Examples: averaging of Monte Carlo samples; convergence testing; image comparison metrics; matrix operations.

- *Reduce* combines every element in a collection into one using an *associative* operator:
  $x+(y+z) = (x+y)+z$
- For example: *reduce* can be used to find the sum or maximum of an array.
- Vectorization may require that the operator *also* be *commutative*:
  $x+y = y+x$

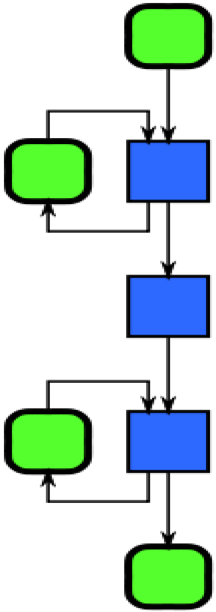
Stencil



Examples: image filtering including convolution, median, anisotropic diffusion

- *Stencil* applies a function to neighbourhoods of an array.
- Neighbourhoods are given by set of relative offsets.
- Boundary conditions need to be considered.

Pipeline



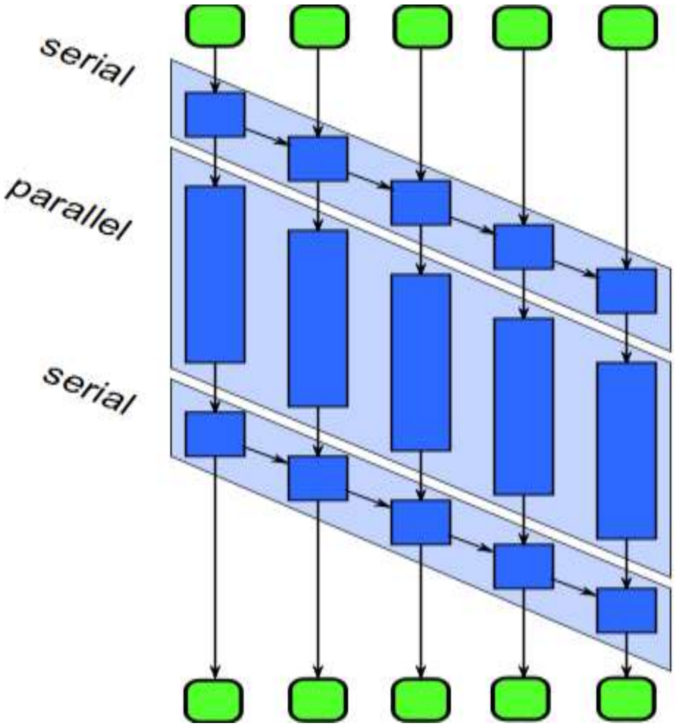
- *Pipeline* uses a sequence of stages that transform a flow of data

- Some stages may retain state

- Data can be consumed and produced incrementally: "online"

Examples: image filtering, data compression and decompression, signal processing

Pipeline



- Parallelize pipeline by
  - Running different stages in parallel
  - Running *multiple copies* of stateless stages in parallel

- Running multiple copies of stateless stages in parallel requires reordering of outputs

- Need to manage buffering between stages

For More Information

Structured Parallel Programming

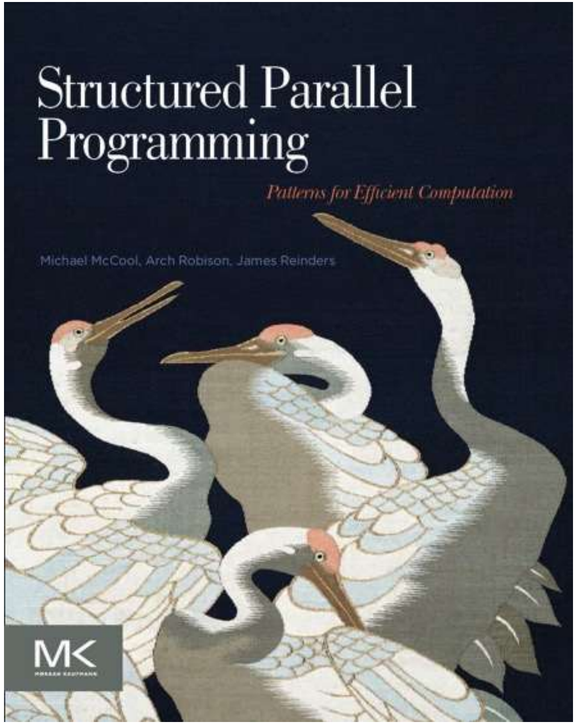
- Michael McCool
- Arch Robison
- James Reinders

Uses Cilk Plus and TBB as primary frameworks for examples.

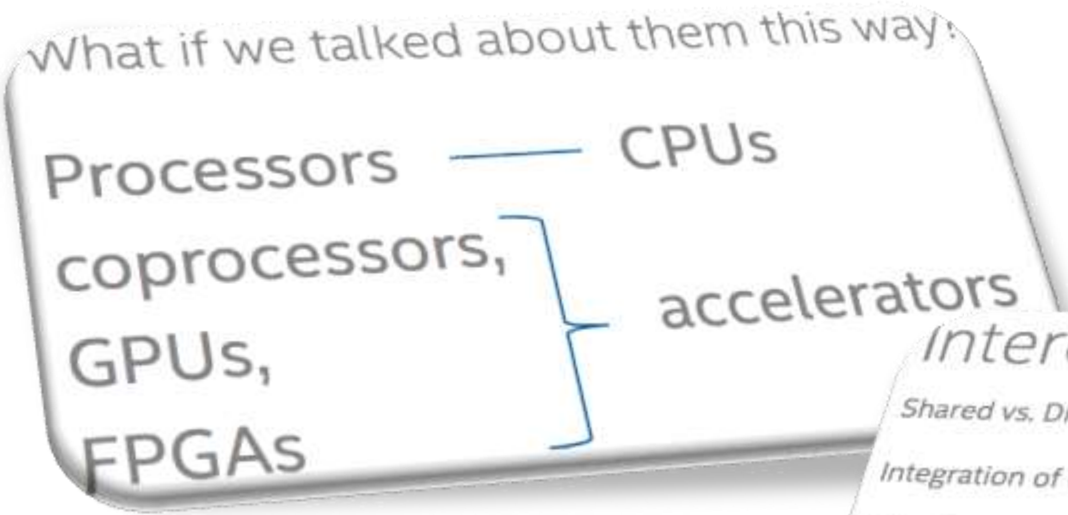
Appendices concisely summarize Cilk Plus and TBB.

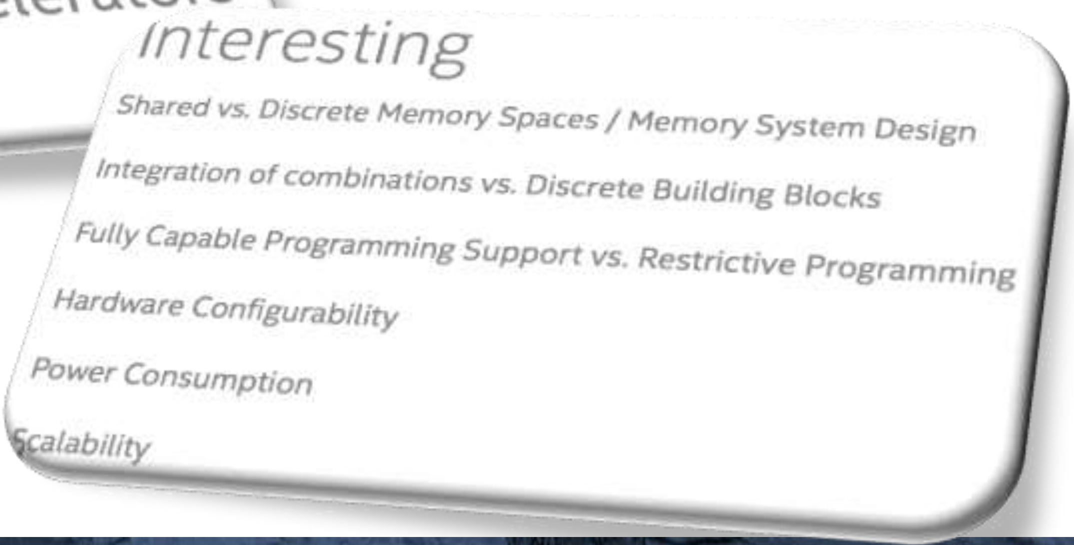
www.parallelbook.com

(pointers to teaching materials, ours and others!)



Computer Architecture is FUN AGAIN





we need to make sure software is not collateral damage.

Performance and Performance Portability should be a requirement.

It’s your Forest

Increase exposing parallelism.

Increase locality of reference.

YOUR MISSION

Questions?

james.r.reinders@intel.com

Break Now

We resume @ 10:45am

(to talk about TBB, OpenMP, SIMD/vectors)

james.r.reinders@intel.com

James Reinders. Parallel Programming Evangelist. Intel.

James is involved in multiple engineering, research and educational efforts to increase use of parallel programming throughout the industry. He joined Intel Corporation in 1989, and has contributed to numerous projects including the world's first TeraFLOP/s supercomputer (ASCI Red) and the world's first TeraFLOP/s microprocessor (Intel® Xeon Phi™ coprocessor). James been an author on numerous technical books, including VTune™ Performance Analyzer Essentials (Intel Press, 2005), Intel® Threading Building Blocks (O'Reilly Media, 2007), Structured Parallel Programming (Morgan Kaufmann, 2012), Intel® Xeon Phi™ Coprocessor High Performance Programming (Morgan Kaufmann, 2013), Multithreading for Visual Effects (A K Peters/CRC Press, 2014), High Performance Parallelism Pearls Volume 1 (Morgan Kaufmann, Nov. 2014), and High Performance Parallelism Pearls Volume 2 (Morgan Kaufmann, Aug. 2015). James is working on a refresh of both the Xeon Phi™ book (original Feb. 2013, revised with KNL information by mid-2016) and a refresh of the TBB book (original June 2007, revised by 2017).

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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 specific computer 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 in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804