Intel Solutions for High Performance Scientific Applications

Legal Disclaimer

Intel may make changes to specifications and product descriptions at any time, without notice.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations

Intel does not control or audit the design or implementation of third party benchmarks or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmarks are reported and confirm whether the referenced benchmarks are accurate and reflect performance of systems available for purchase.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See www.intel.com/products/processor_number for details.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

Lead-free: 45nm product is manufactured on a lead-free process. Lead is below 1000 PPM per EU RoHS directive (2002/95/EC, Annex A). Some EU RoHS exemptions for lead may apply to other components used in the product package.

Intel, Intel Xeon, Intel Core microarchitecture, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others

HPC Requirements

- You demand more performance:
  - Raw Performance
  - More Performance/$
  - More Performance/Meter^2
  - More Performance/ Watt
- You also desire:
  - Programmability
  - Portability/Flexibility
- Beyond the processor, you demand:
  - Fast I/O, Storage
  - Fast Interconnect (InfiniBand* Technology, proprietary), Switch costs

Intel Platform Architecture

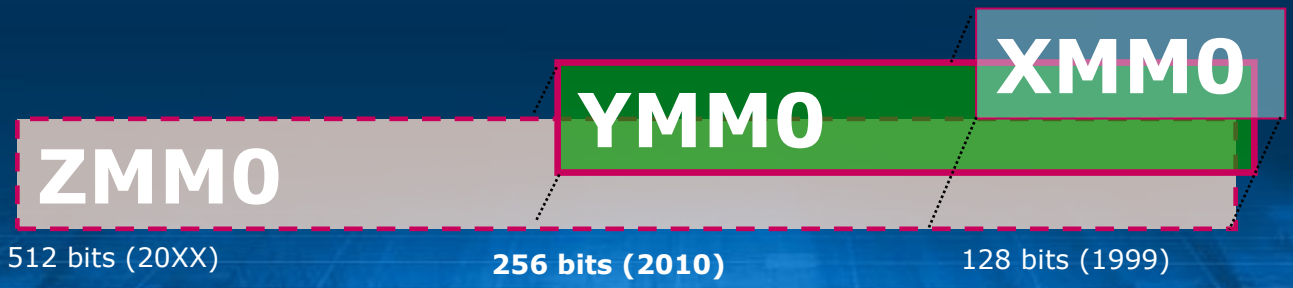
Mainstream PC and Server

Tick	Tock	Tick	Tock	Tick
45nm		32nm		22nm
Penryn	Nehalem	Westmere	Sandybridge	

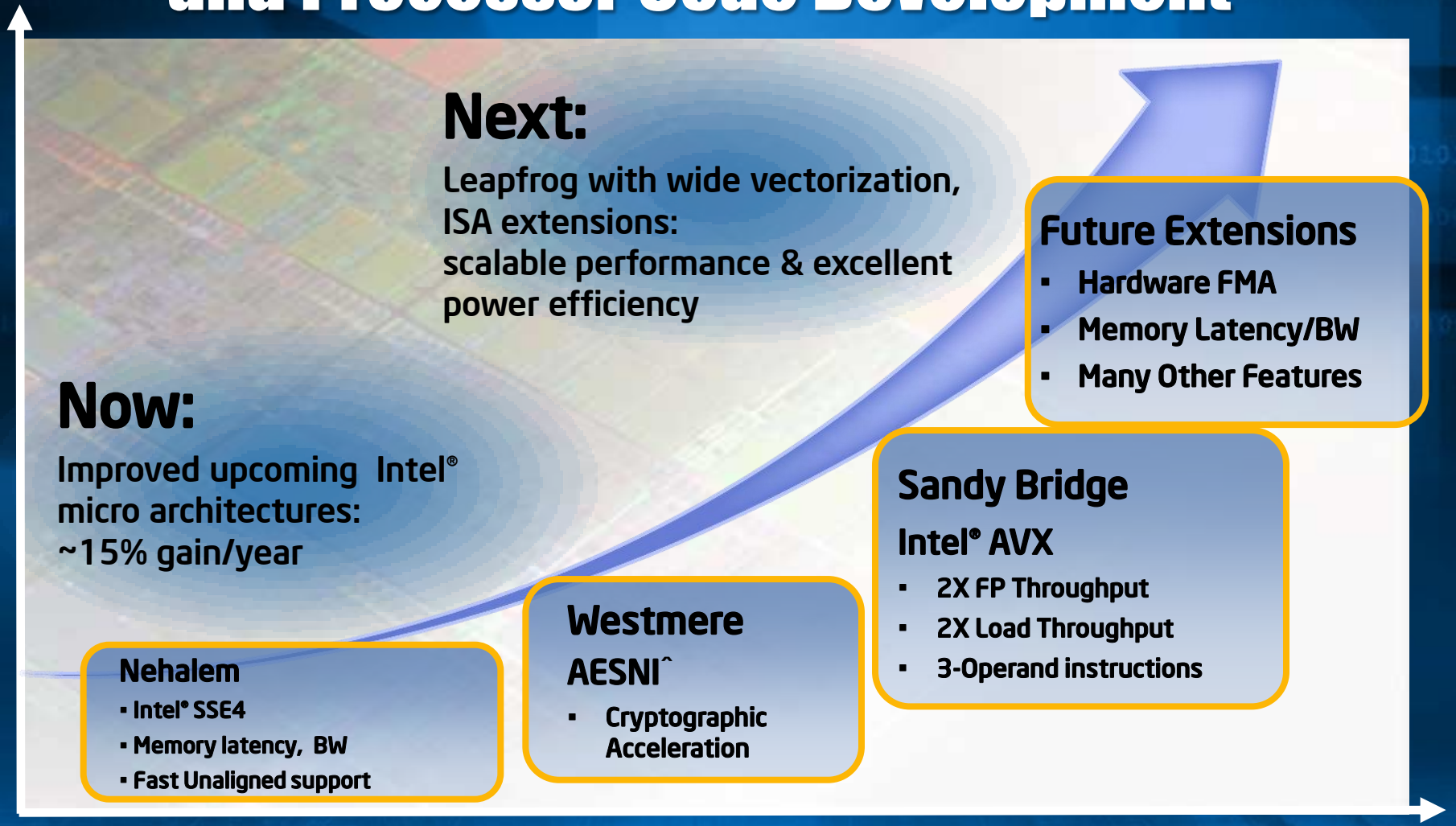
Sandy Bridge: Intel® AVX!

A 256-bit vector extension to SSE

- Intel® AVX extends all 16 XMM registers to 256bits
- Intel® AVX works on either
  - The whole 256-bits
  - The lower 128-bits(like existing SSE instructions)
    - A drop-in replacement for all existing scalar/128-bit SSE instructions
- The new state extends/overlays SSE
- The lower part (bits 0-127) of the YMM registers is mapped onto XMM registers



Setting the Pace for Intel Instruction Set and Processor Code Development

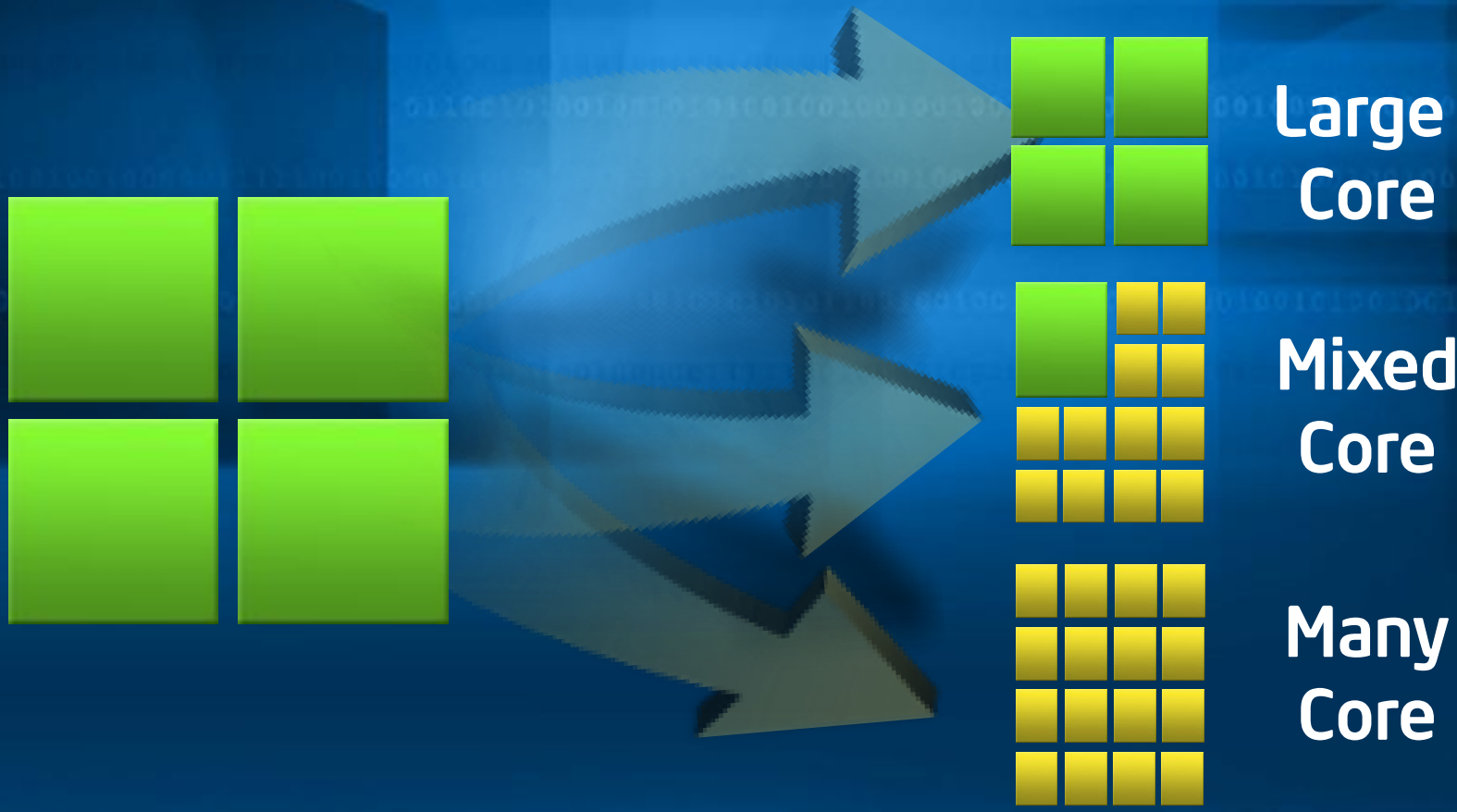


^AESNI - Advanced Encryption Standard New Instruction
AVX – Advanced Vector Extensions

Intel HPC Architectures

Meeting Your Challenges

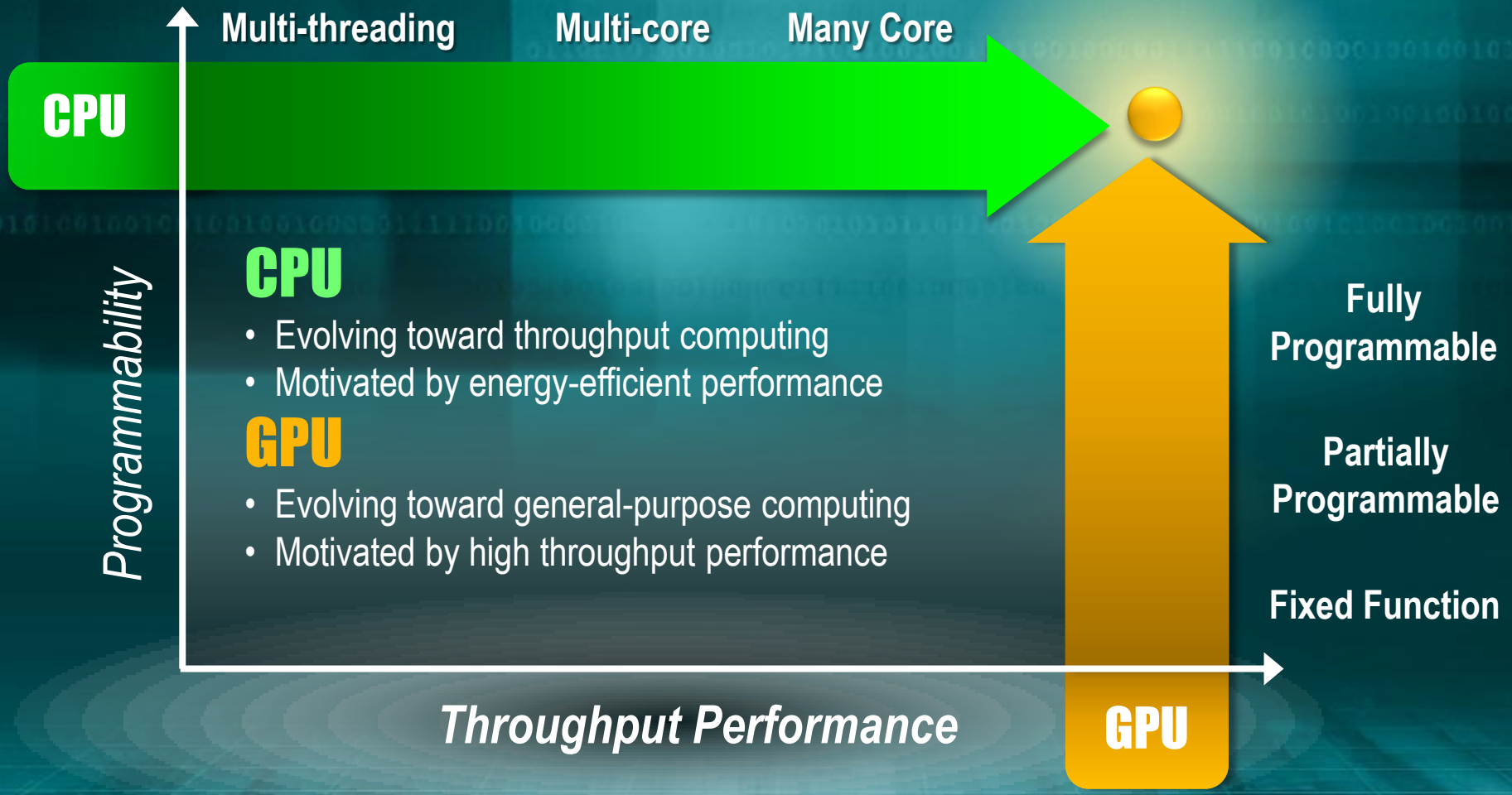
Delivering Performance Across A Spectrum Of Implementations



Performance That Scales Forward

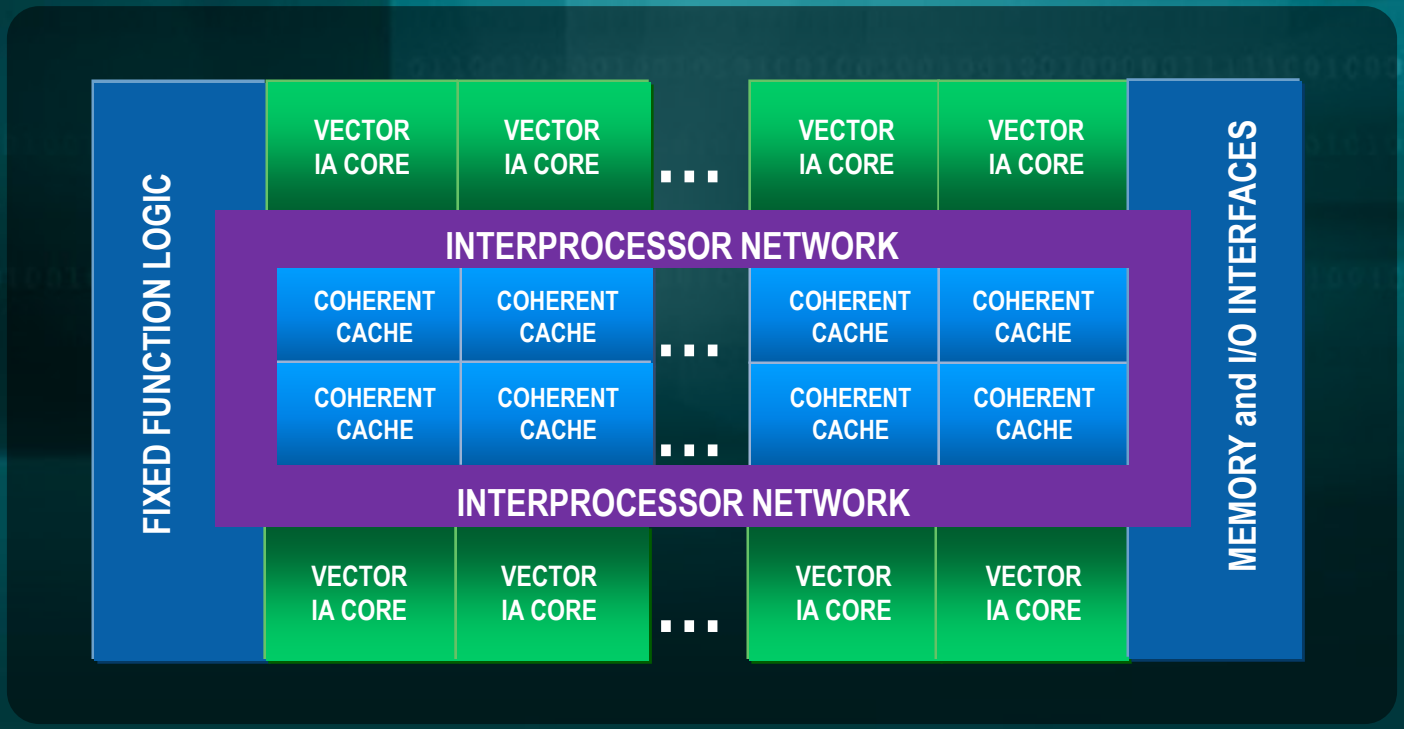
Architecture Evolution:

A Collision Course



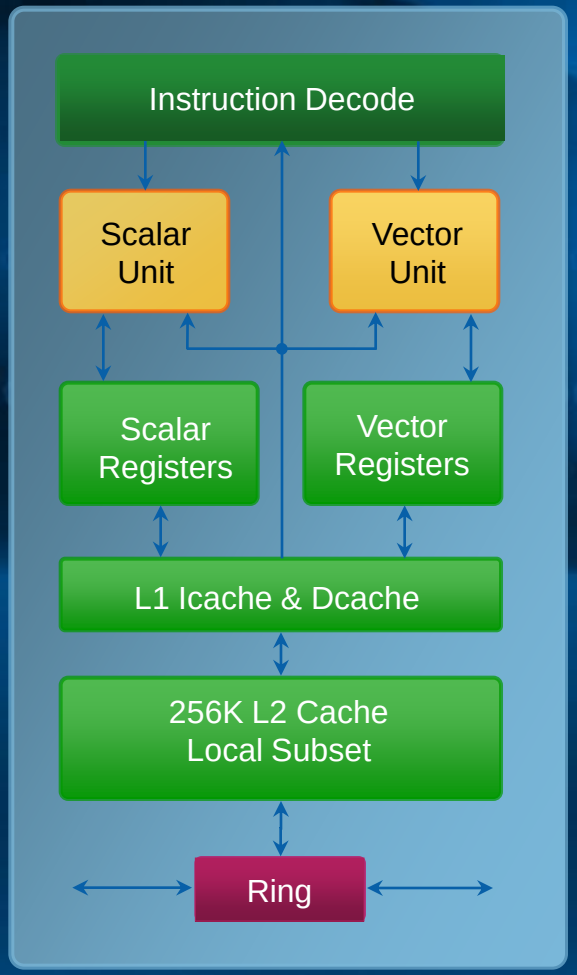
Larrabee Processor

A Computational Co-Processor for the Intel® Xeon® and Core® Families



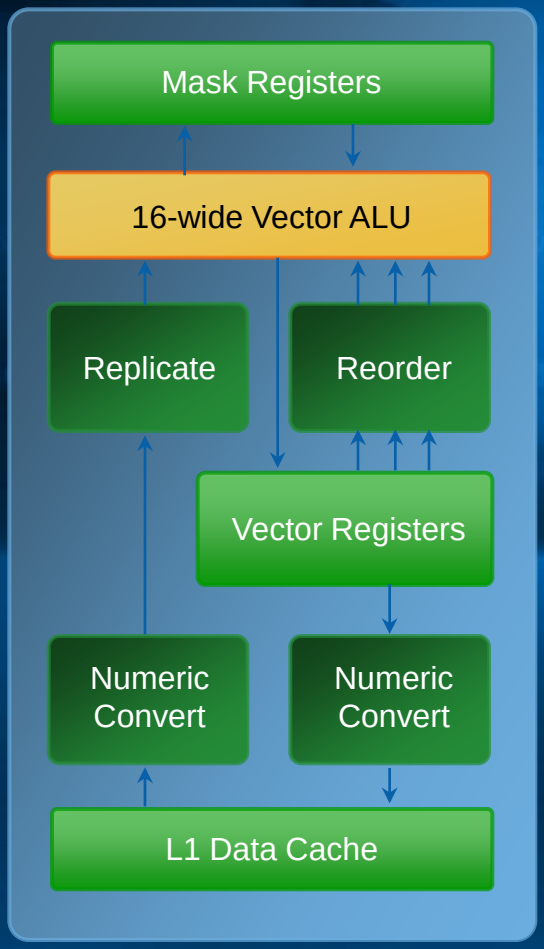
- Many Cores and Many More Threads
- Automatic Memory Management
- Standard IA Programming and Memory Model

Processor Core Block Diagram



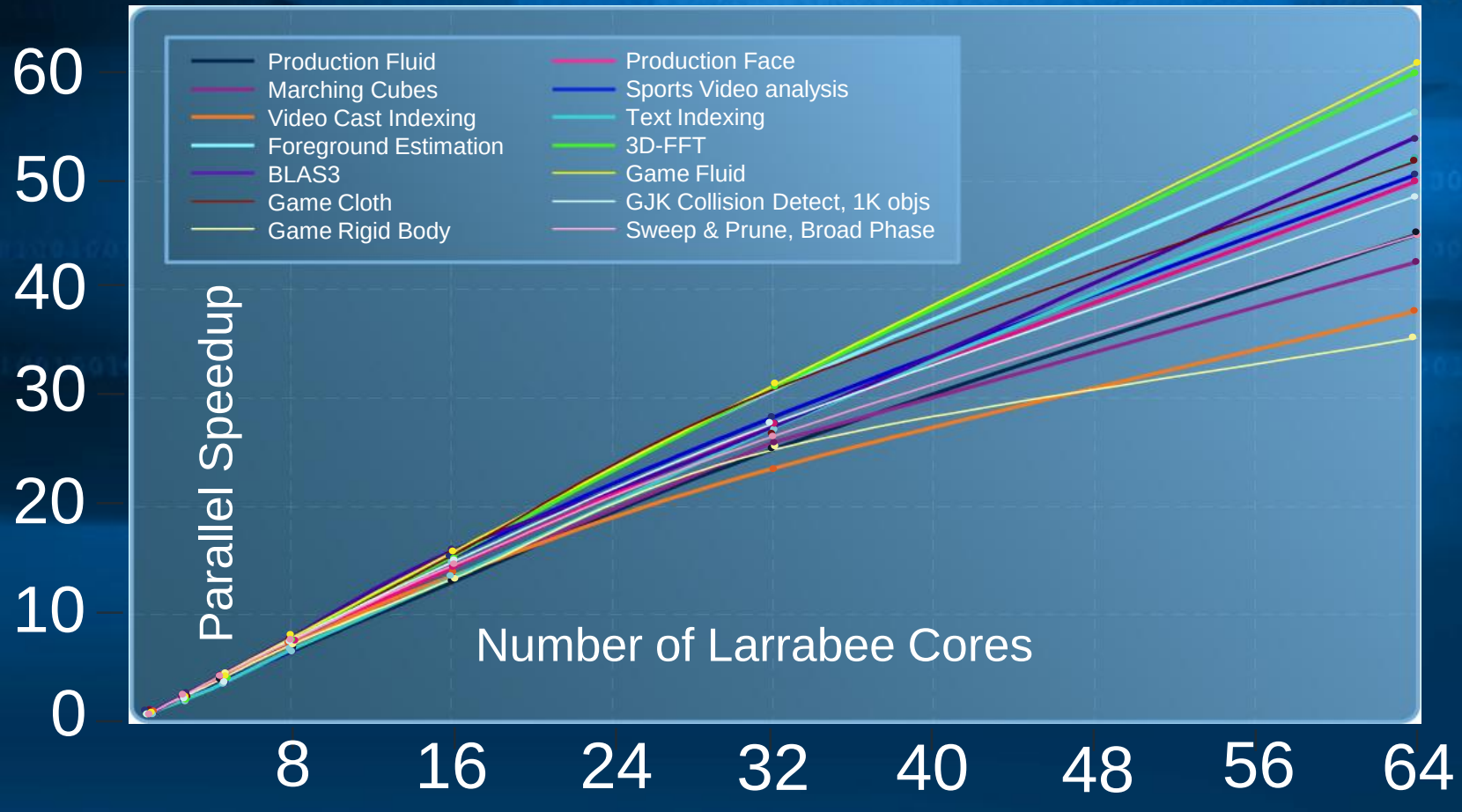
- Separate scalar and vector units with separate registers
- Vector unit: 16 32-bit ops/clock
- In-order instruction execution
- Short execution pipelines
- Fast access from L1 cache
- Direct connection to each core's subset of the L2 cache
- Prefetch instructions load L1 and L2 caches

Vector Unit Block Diagram



- Vector complete instruction set
  - Scatter/gather for vector load/store
  - Mask registers select lanes to write, which allows data-parallel flow control
  - This enables mapping a separate execution kernel to each VPU lane
- Vector instructions support
  - Fast read from L1 cache
  - Numeric type conversion and data replication while reading from memory
  - Rearrange the lanes on register read
  - Fused multiply add (three arguments)
  - Int32, Float32 and Float64 data

Non-graphics Application Scaling

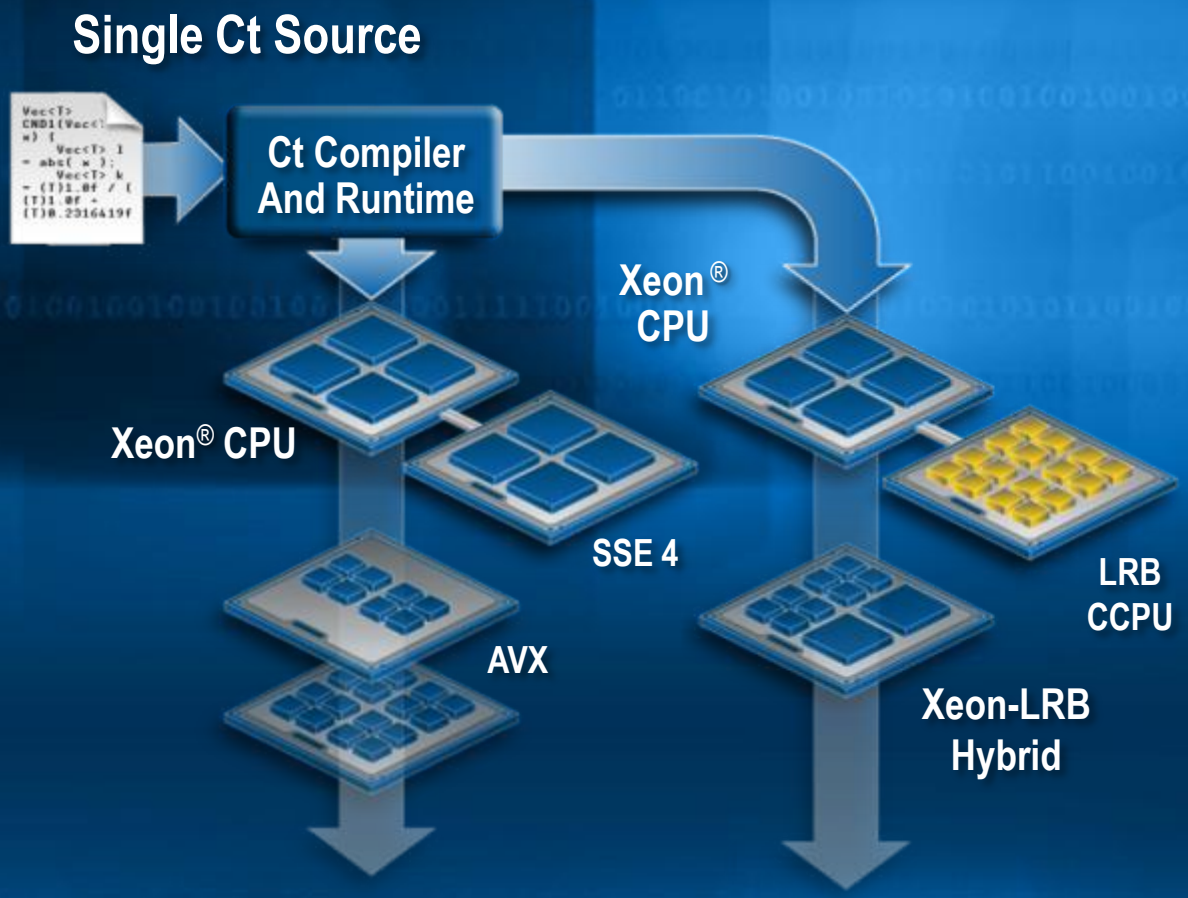


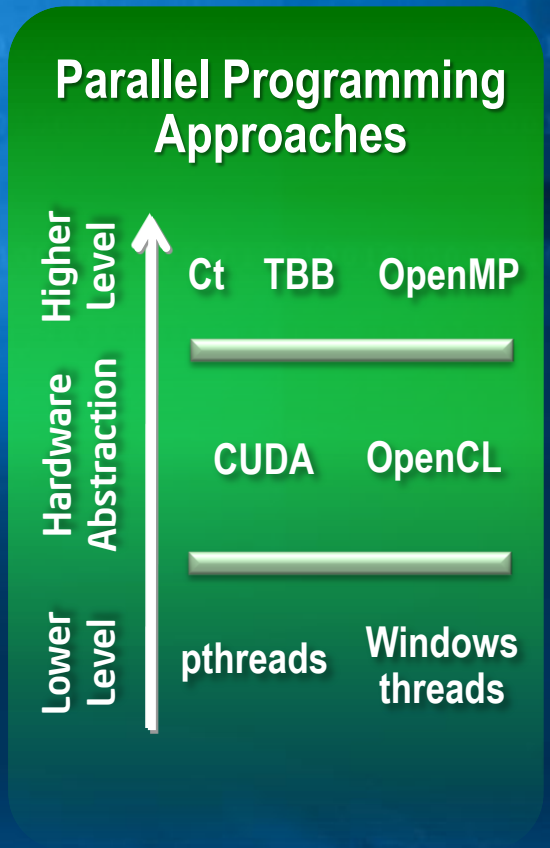
Flexible & programmable for many applications

Data in graph from Seiler, L., Carmean, D., et al. 2008. *Larrabee: A many-core x86 architecture for visual computing*. SIGGRAPH '08: ACM SIGGRAPH 2008 Papers, ACM Press, New York, NY

Making Parallel Computing Pervasive

High-Level Data Parallel Programming With Ct Technology





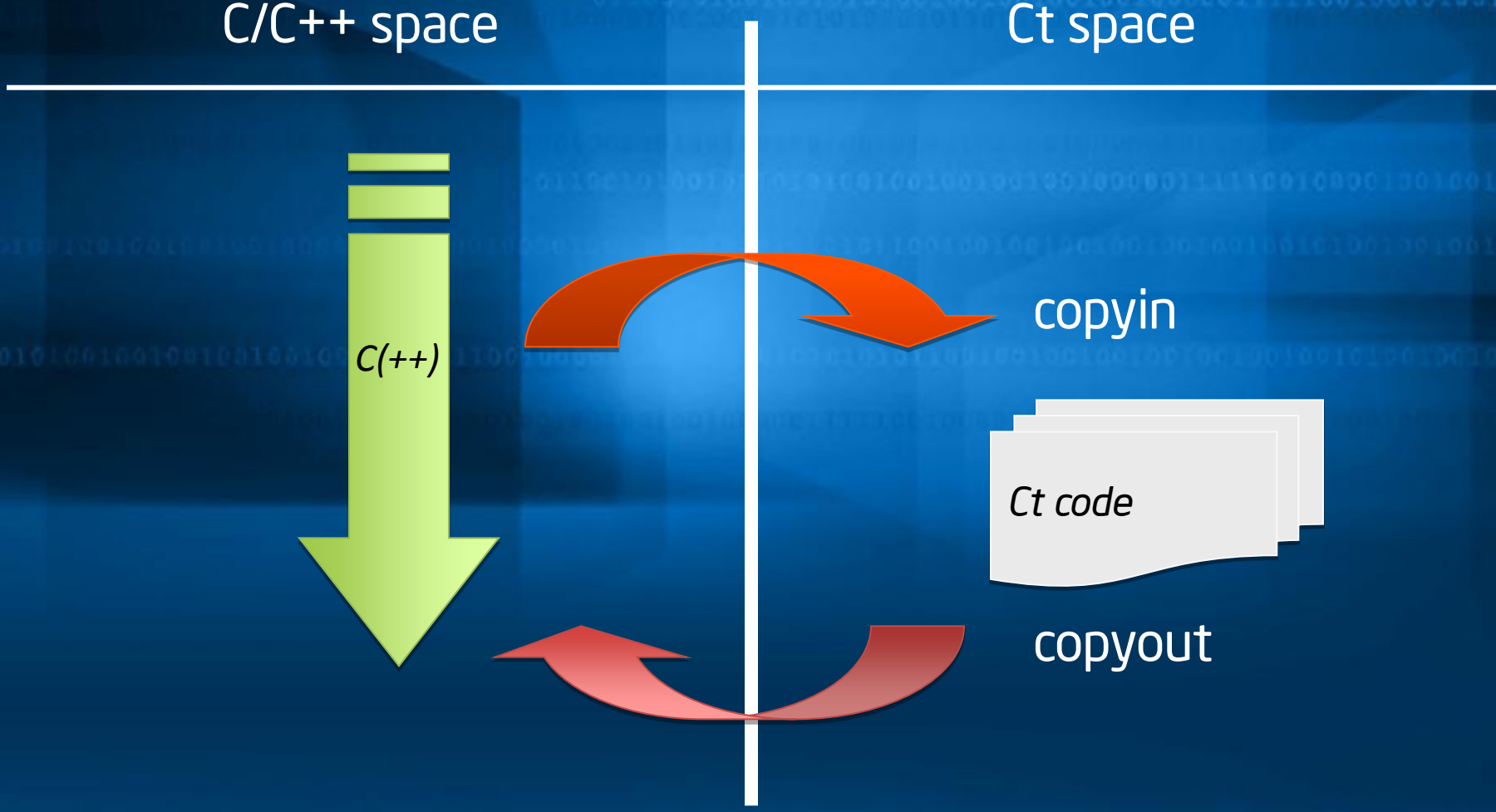
Programmer Writes Serial-Like Code, Ct Automatically Exploits Parallelism

Design Constraints

- Target language: C++
- C++ will continue to be the dominant languages for high performance for the next 5+ years

- …and we *mean* **standard** C++!
- Custom syntactic extensions face huge barriers to adoption
- It is possible to design a desirable semantics through an API-like interface with some Macro magic

Segregated C/C++ and Ct Spaces



Ct is garbage collected; *practically, a small number of Vecs are uncollected by compiler (ref counting)*

More complex: Porting Black-Scholes

Black-Scholes Using C Loops

```
float s[N], x[N], r[N], v[N], t[N];
float result[N];
for(int i = 0; i < N; i++) {

  float d1 = s[i] / ln(x[i]);
  d1 += (r[i] + v[i] * v[i] * 0.5f) * t[i];
  d1 /= sqrt(t[i]);
  float d2 = d1 - sqrt(t[i]);

  result[i] = x[i] * exp(r[i] * t[i]) *
    ( 1.0f - CND(d2)) + (-s[i]) * (1.0f - CND(d1));
}
```

Black-Scholes Using Ct

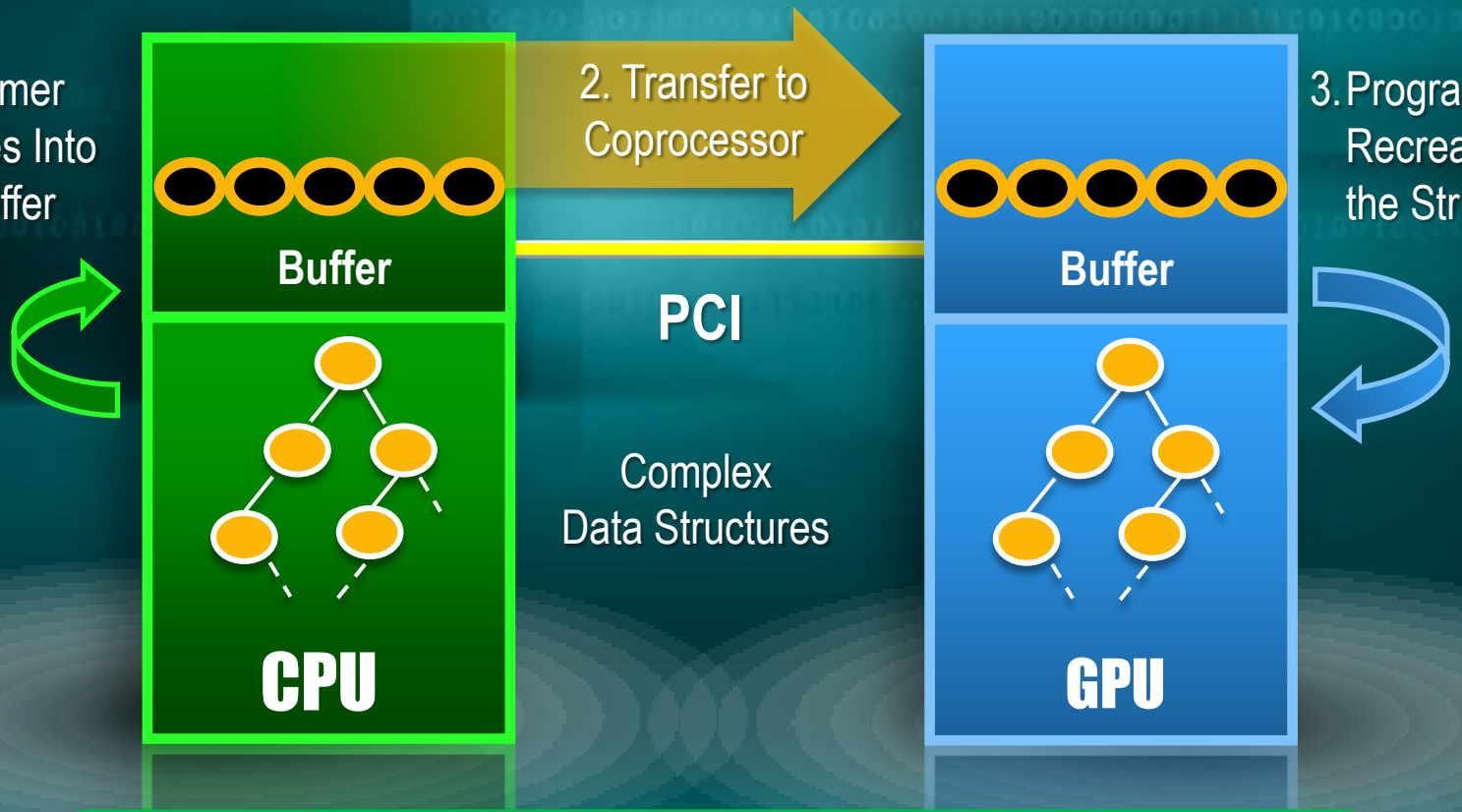
```
#include <ct.h>
using namespace OpenCt;

float s[N], x[N], r[N], v[N], t[N];
float result[N];
Vec<F32> S(s, N), X(x, N), R(r, N), V(v, N), T(t, N);

Vec<F32> d1 = S / ln(X);
d1 += (R + V * V * 0.5f) * T;
d1 /= sqrt(T);
Vec<F32> d2 = d1 - sqrt(T);

Vec<F32> tmp = X * exp(R * T) *
   ( 1.0f - CND(d2)) + (-S) * (1.0f - CND(d1));

tmp.copyOut(result, sizeof(result));
```

1. #include <ct.h> and using namespace
2. Vector operations subsumes loop
3. The Ct code is almost the same as the original loop body
4. copyOut at the end (if you're done!)

RapidMind products & technology

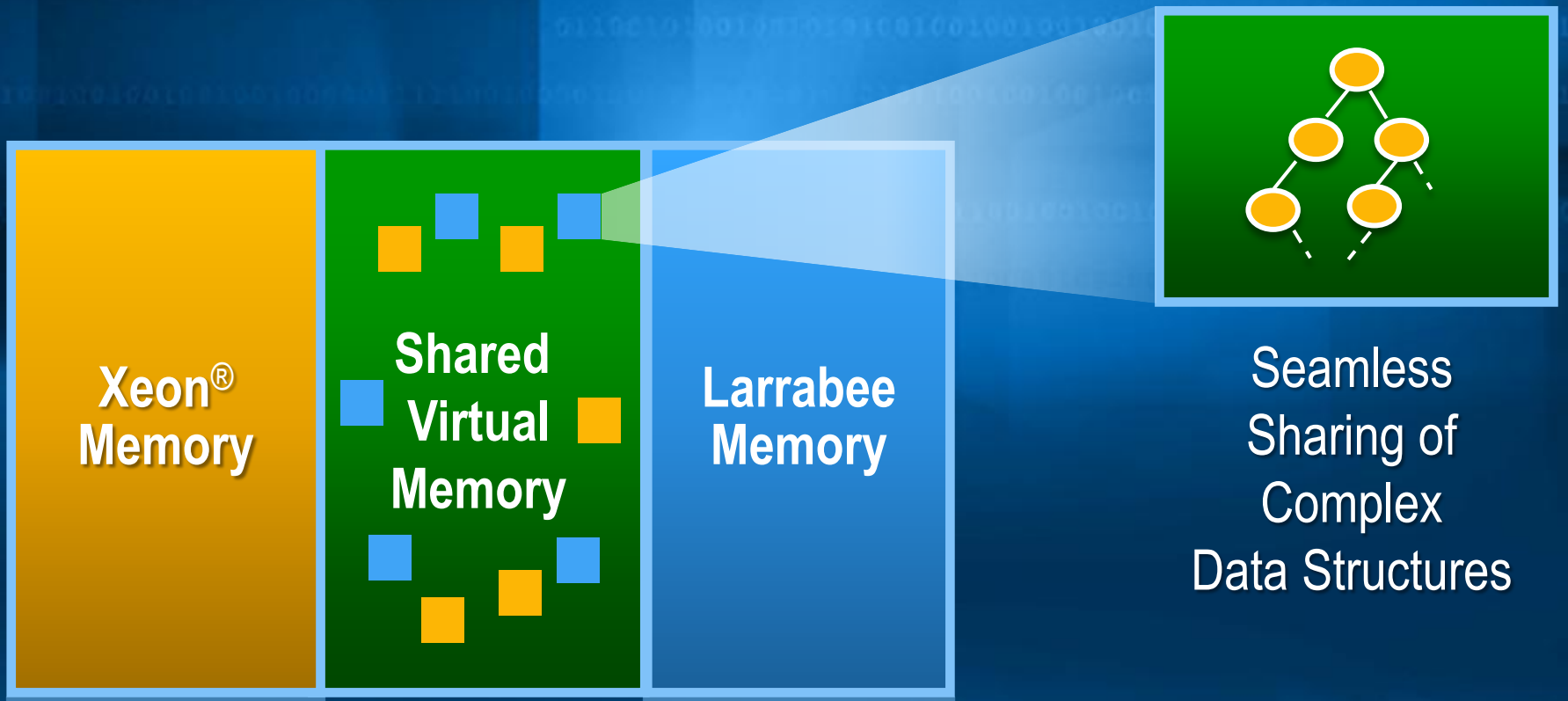
- RapidMind technology is very similar to Ct
  - Provided Types: Value<N,T>, Array<D,V>
  - Program (Computation Container executed in parallel)
  - Dynamic Code Generation
- Additional strengths:
  - Cross platform support (CPU, Cell/B.E., GPU)
  - Additional API features
- Integrating the best of Ct technology with the best of RapidMind technology into a merged product.
- Preserving the investment of our customers in existing products and ensure a smooth transition

Inefficiency of Traditional CPU-GPU Interaction



CPU-GPU with Separate Memory Spaces

M-Y-O: Shared Virtual Memory for IA-Compatible Co-Processors



Larrabee and Many Core for High Performance and Visual Compute

	Multi-Core CPU	GPU	Many-Core Larrabee
High Compute Density			
General-purpose ISA			
High Memory B/W			
Shared Coherent Cache			
Texture Filtering			
Efficient SIMD Scatter-Gather			

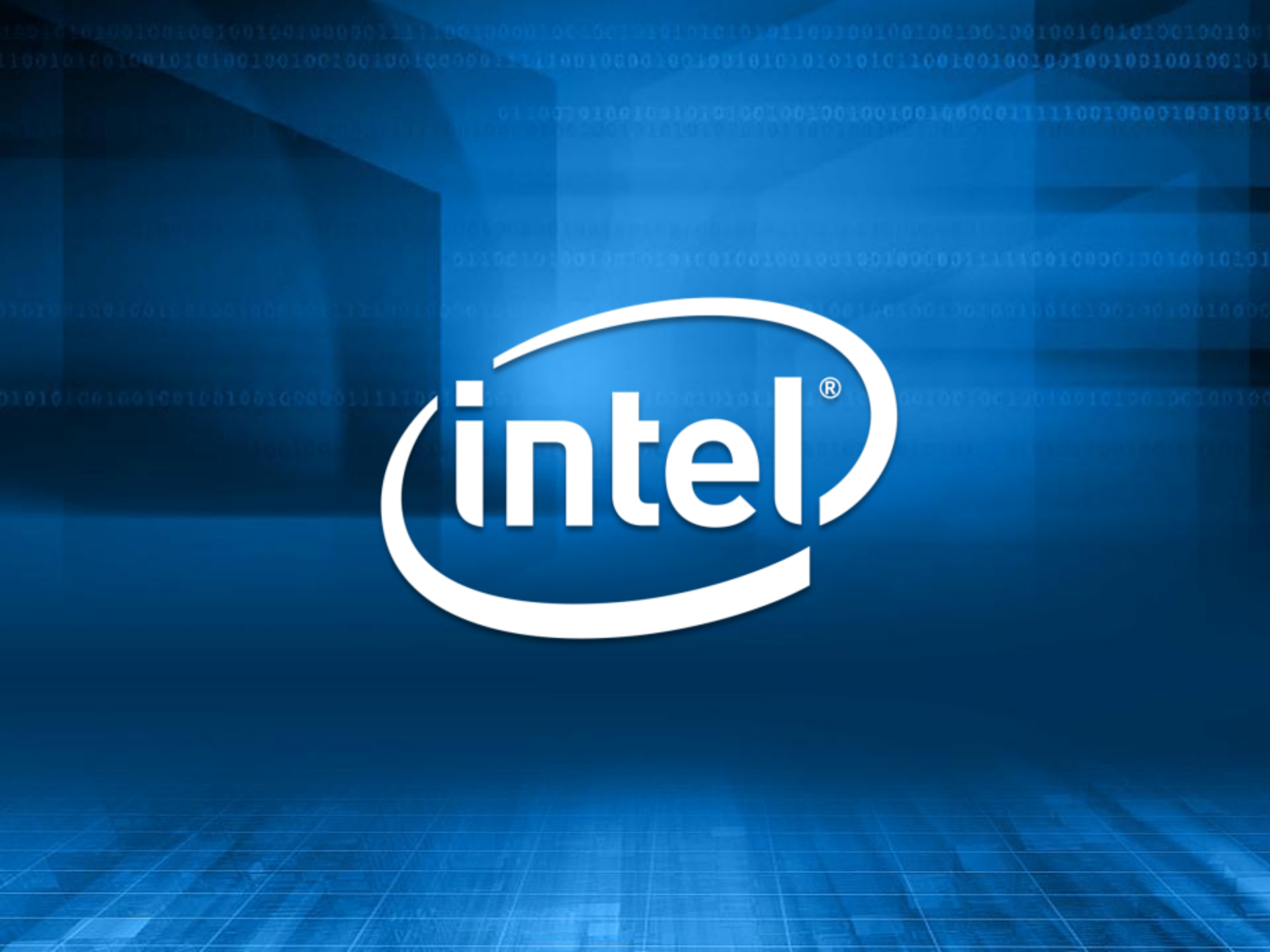
Larrabee: CPU programmability, Tools, Process, Thread, and Data Parallelism

Conclusion

- Future is many-core
- Challenges:
  - Peak performance ≠ to achieved one
  - Parallel programming is hard
- Need to ease the transition with
  - familiar ISA, programming model, tools and libraries
- **Intel current and upcoming architectures offer full parallel platform!**

Thank you!

Questions? / Comments!


intel®