PARALLELISM, PERFORMANCE & OPTIMIZATION ON INTEL ARCHITECTURE

Stephen Blair-Chappell
Bayncore

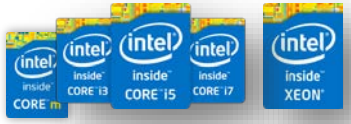
OPENING STATEMENT

"**Parallelism** == **Performance**"

(leads to)

Optimization – *making sure the above statement is true!*

Desktop, Mobile & Server

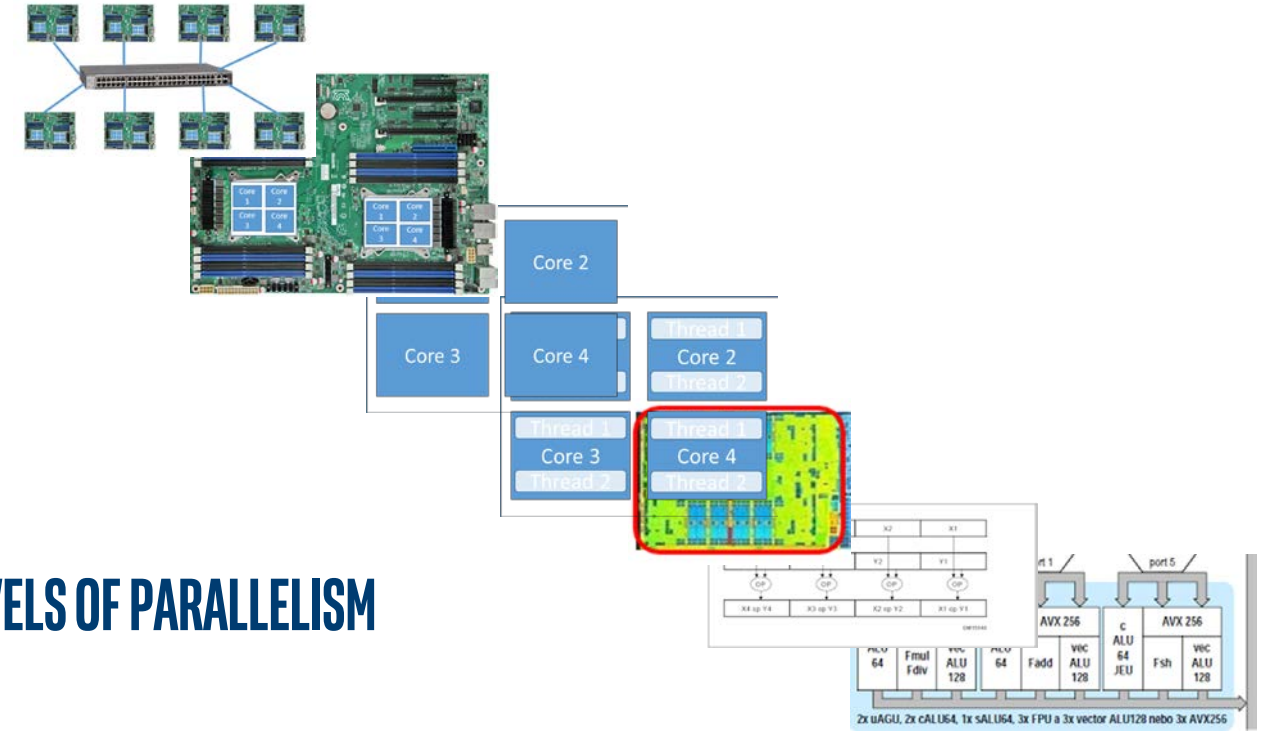


Intel® Core™		2nd Generation Intel® Core™	3rd Generation Intel® Core™	4th Generation Intel® Core™
Nehalem (2008) New Microarchitecture 45nm	Westmere (2010) New Process Technology 32nm	Sandy Bridge (2011) New Microarchitecture 32nm	Ivy Bridge (2012) New Process Technology 22nm	Haswell (2013) New Microarchitecture 22nm

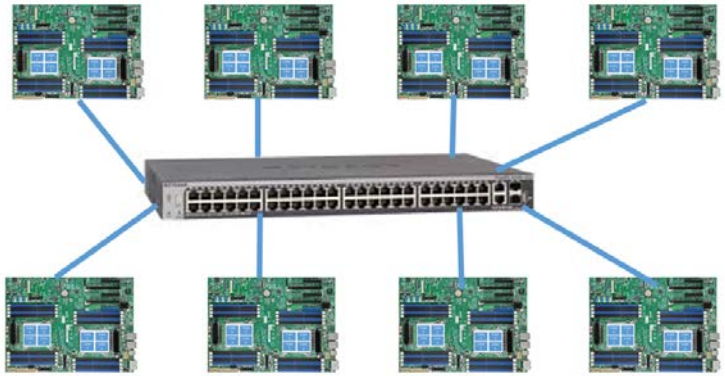
5th Generation Intel® Core™	6th Generation Intel® Core™	7th Generation Intel® Core™	8th Generation Intel® Core™			Future
Broadwell (2014) New Process Technology 14nm	Skylake (2016) New Microarchitecture 14nm	Kaby Lake (2016) enhanced 14nm	Kaby Lake (2017) enhanced 14nm	Skylake X (2017) enhanced 14nm	Coffee Lake (2017) enhanced 14nm	10nm

PARALLELISM ON INTEL ARCHITECTURE

SEVEN LEVELS OF PARALLELISM



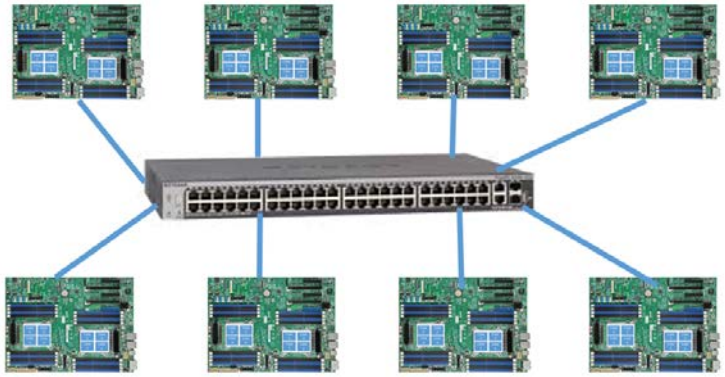
#1 – NODE-LEVEL PARALLELISM



Levels of Parallelism

Node

#1 – NODE-LEVEL PARALLELISM

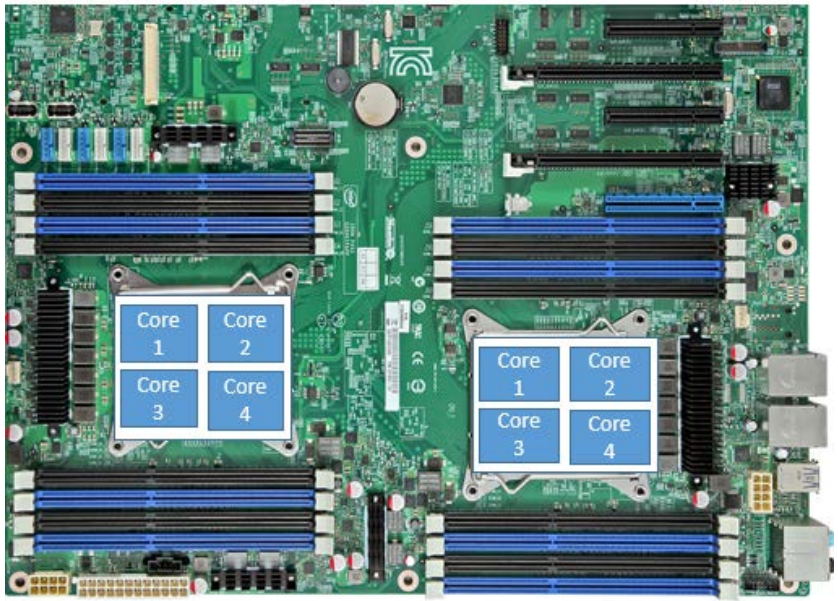


Levels of Parallelism

Node

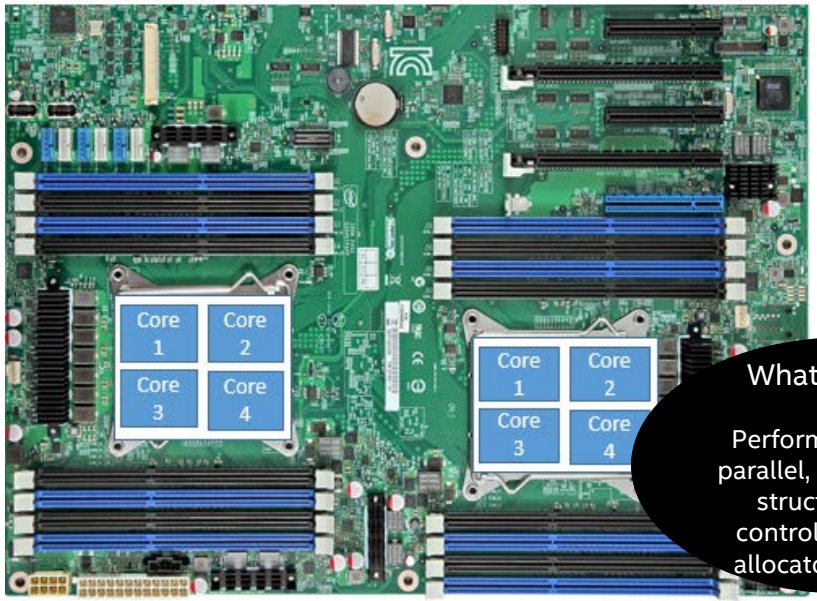
What can I do?
Increase per-node perf., identify scalability issues (Intel Trace Analyzer & Collector), employ comm-avoiding algorithms, etc. (more nodes ;-)

#2 SOCKET-LEVEL PARALLELISM



Levels of Parallelism
Node
Socket

#2 SOCKET-LEVEL PARALLELISM



What can I do?

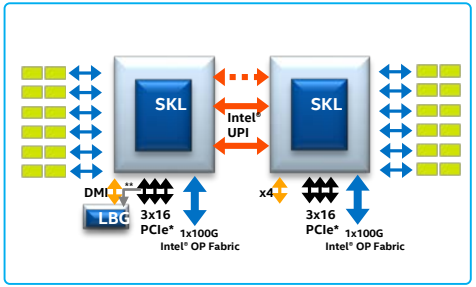
Perform data init. in parallel, separate data structures, take control with NUMA allocator (libnuma).

Do I have NUMA-issues?

Check with 'numactl -i all', or Intel Vtune.

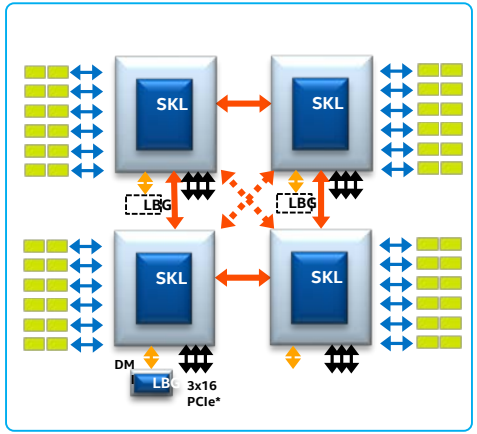
Levels of Parallelism
Node
Socket

2S Configurations



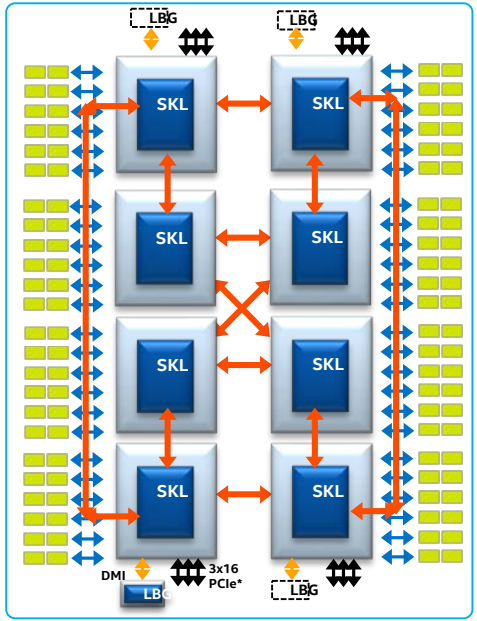
(2S-2UPI & 2S-3UPI shown)

4S Configurations



(4S-2UPI & 4S-3UPI shown)

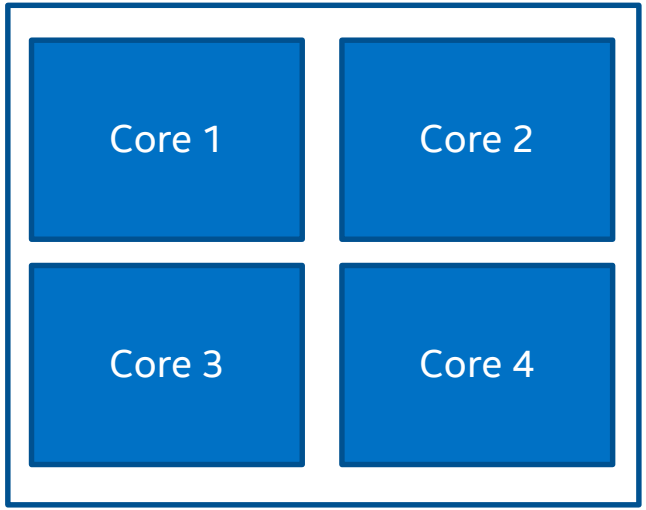
8S Configuration



INTEL® XEON® SCALABLE PROCESSOR SUPPORTS UP TO 8 SOCKETS

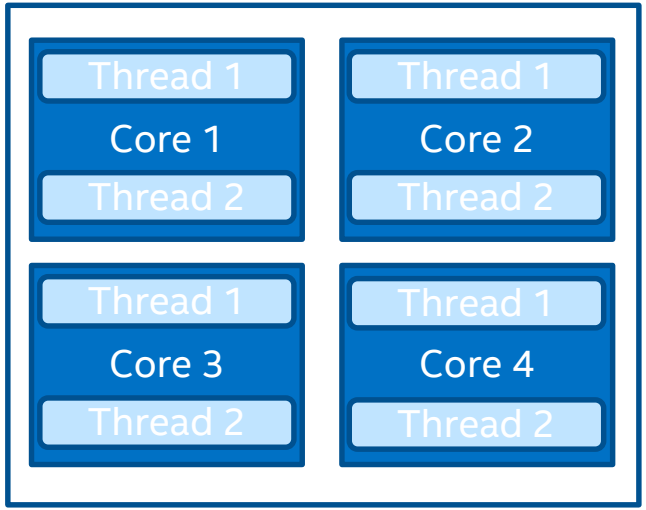
WITHOUT THE NEED FOR AN ADDITIONAL NODE CONTROLLER

#3 CORE / THREAD LEVEL PARALLELISM



Levels of Parallelism
Node
Socket
Core / Thread-Level

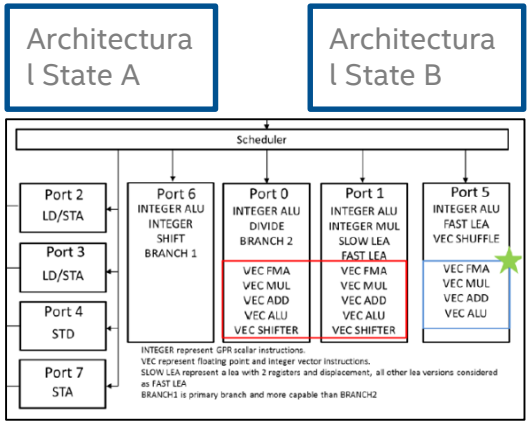
#4 THREAD-LEVEL PARALLELISM (WITH HYPERTHREADING AKA SMT)



Levels of Parallelism
Node
Socket
Core / Thread-Level
(Hyperthreading)

HYPERTHREADING

- Multiple buffers keep arch. state
- Shared execution blocks
- Enabled by BIOS settings

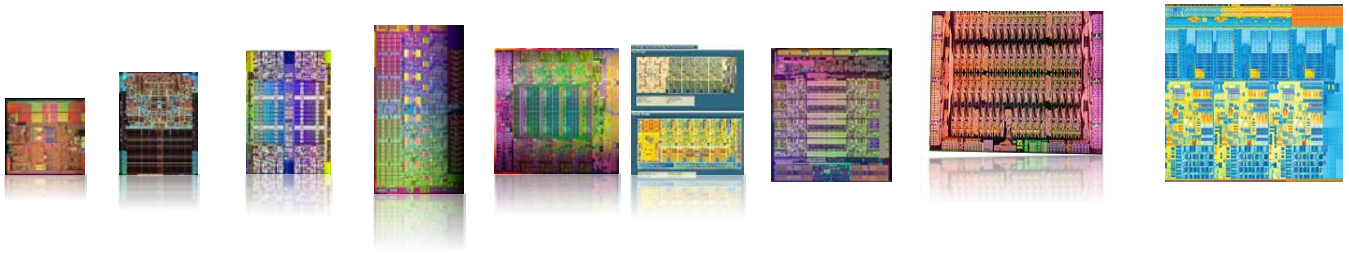


Single Execution Block

Background

- Extracts Instruction Level Parallelism (ILP)
- Complements out-of-order execution
- Intel Core: intra-core slowdowns due to HT are eliminated, BUT slowdown may happen due missing thread-affinization or because of synchronization (locks).

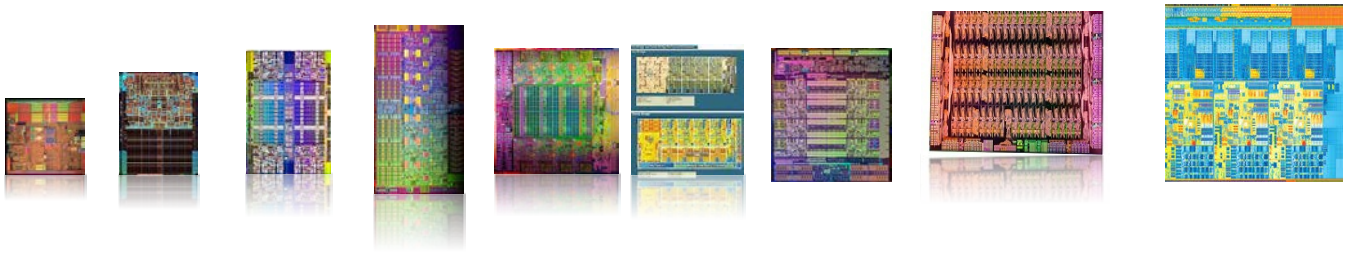
PARALLELISM ON INTEL® ARCHITECTURE (XEON SERVER CORES)



	Intel® Xeon 64-bit	5100	5500	5600	Intel® Xeon E5 SNB	IVB	HSW	BDW	Intel® Xeon scalable processors SKX	Intel® Xeon Phi Coprocessor KNC	Intel® Xeon Phi 2nd gen. KNL
Cores	1	2	4	6	8	10	18	22	28	61	72
Threads	2	2	8	12	16	20	36	44	56	244	288

IVB: Ivy Bridge BDW: Broadwell KNC: Knights Corner SKX: Skylake Server core refines
HSW: Haswell (SKL: Skylake) KNL: Knights Landing SKL client core significanly

PARALLELISM ON INTEL® ARCHITECTURE (XEON SERVER CORES)

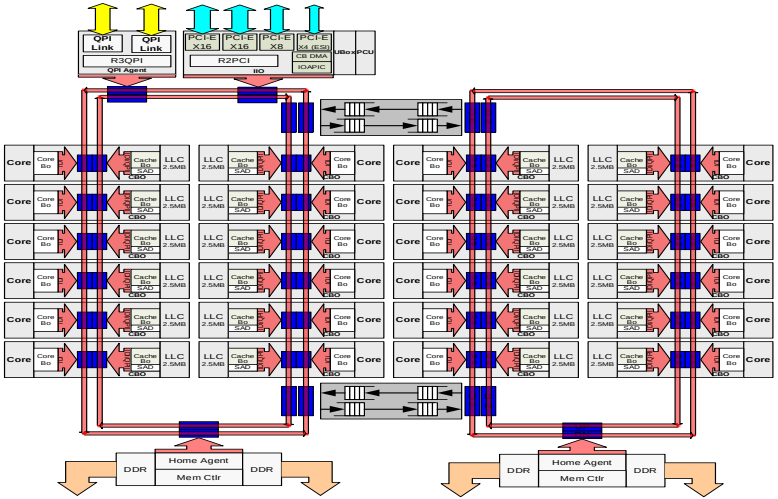


	Intel® Xeon 64-bit	5100	5500	5600	Intel® Xeon E5 SNB	IVB	HSW	BDW	Intel® Xeon scalable processors SKX	Intel® Xeon Phi Coprocessor KNC	Intel® Xeon Phi 2nd gen. KNL
Cores	1	2	4	6	8	10	18	22	28	61	72
Threads	2	2	8	12	16	20	36	44	56	244	288

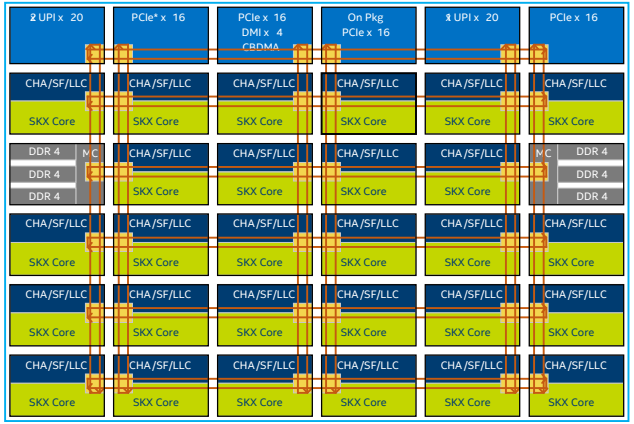
IVB: Ivy Bridge BDW: Broadwell KNC: Knights Corner SKX: Skylake Server core refines
HSW: Haswell (SKL: Skylake) KNL: Knights Landing SKL client core significantly

NEW MESH INTERCONNECT ARCHITECTURE

Intel® Xeon® Processor E7 family (24-core die)



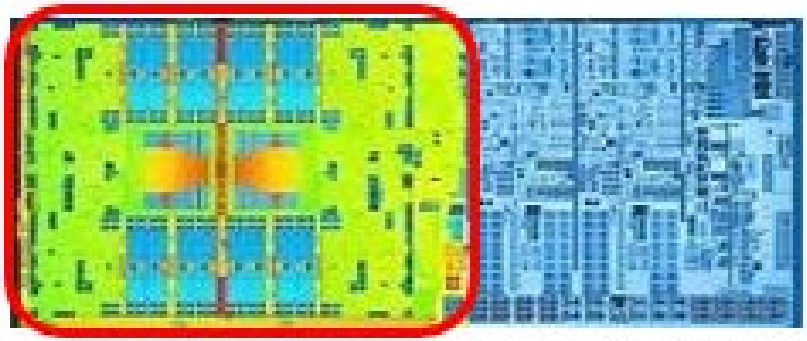
Intel® Xeon® Scalable Processor (28-core die)



CHA Caching and Home Agent ; SF – Snoop Filter ; LLC – Last Level Cache ;
SKX Core Skylake Server Core ; UPI – Intel® UltraPath Interconnect

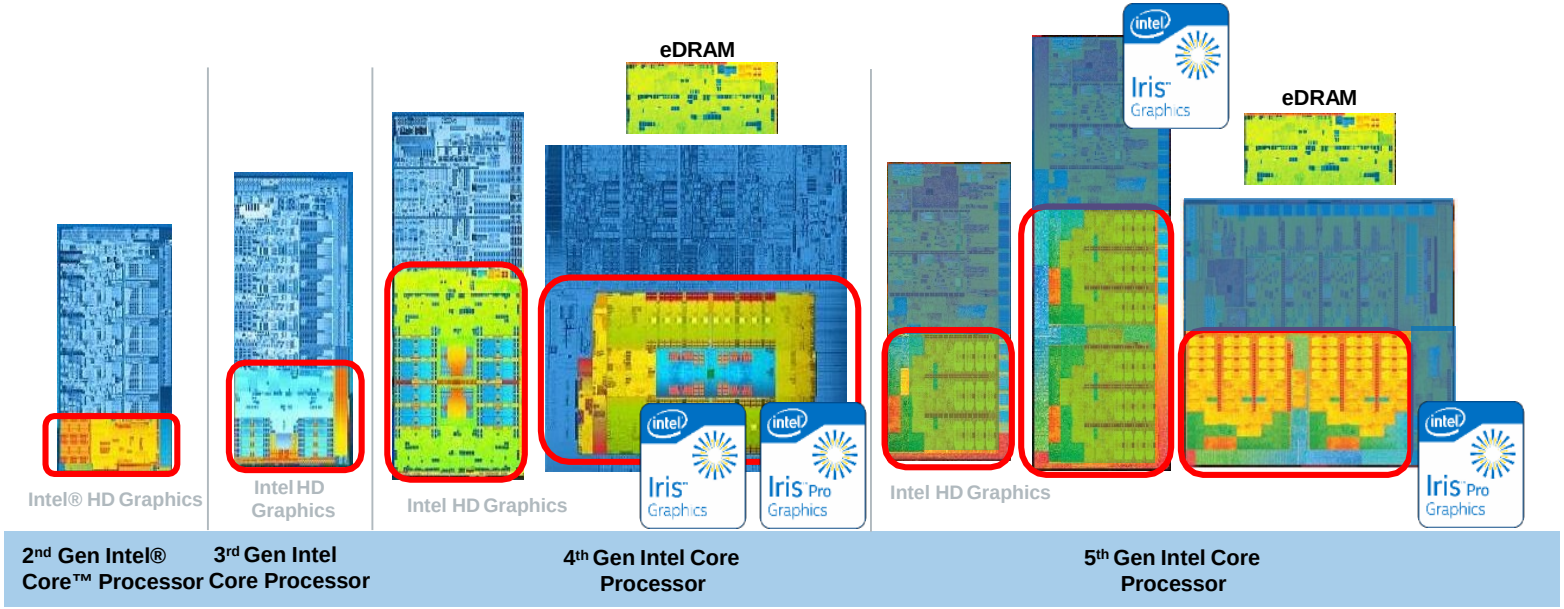
MESH IMPROVES SCALABILITY WITH HIGHER BANDWIDTH AND REDUCED LATENCIES

#5 GPU-CPU PARALLELISM



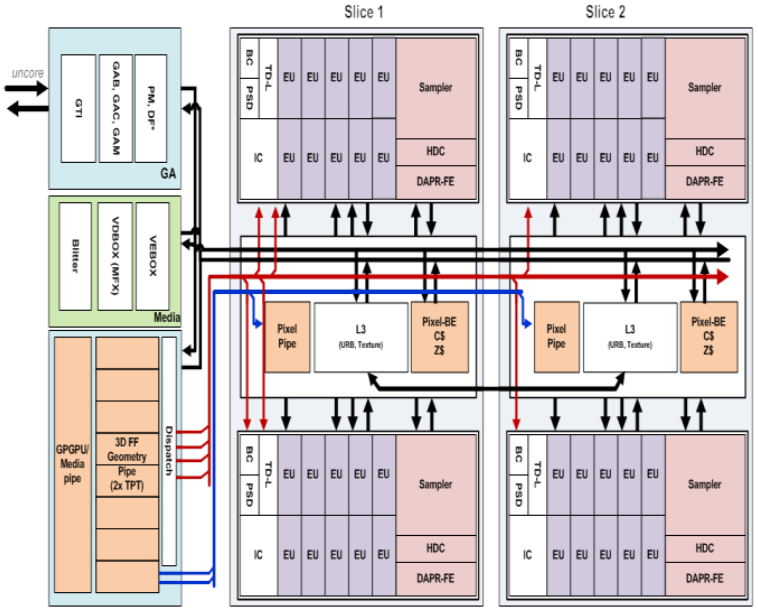
Levels of Parallelism
Node
Socket
Core / Thread-Level
(Hyperthreading)
GPU-CPU

INTEGRATED PROCESSOR GRAPHICS



Lots of compute power for data-parallel (client-)applications

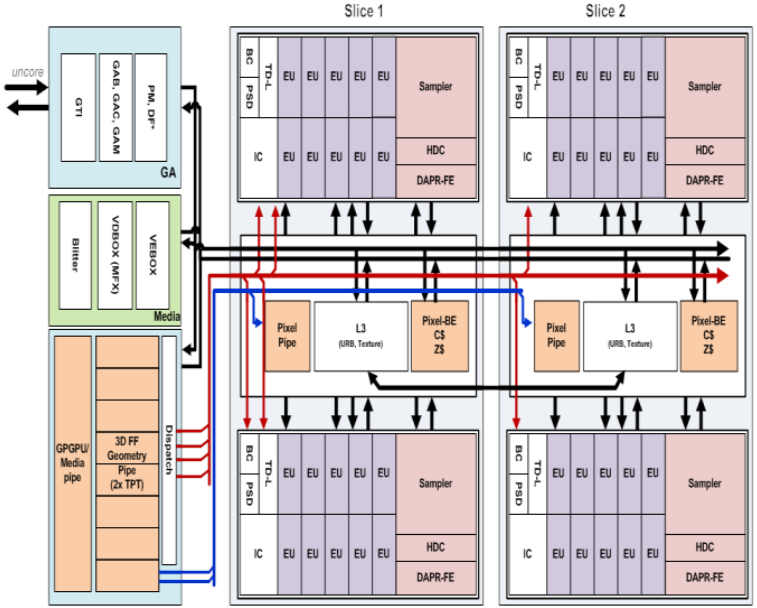
4TH GENERATION INTEL® CORE™ PROCESSORS – HD GRAPHICS

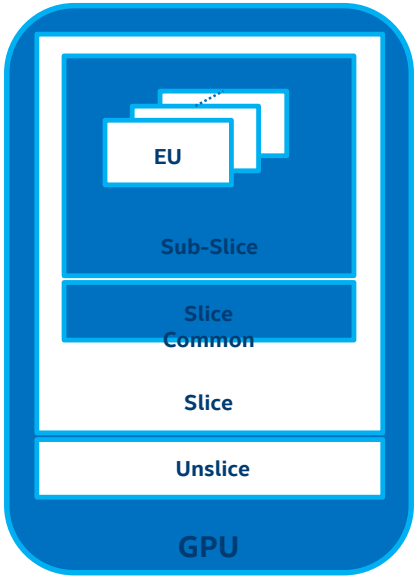


5 Execution Units Per row
2 rows per subslice
2 subslices per slice
2 slices (40 EUs total) in GT3
7 Threads Per EU
    280 threads in GT3
128 Registers per thread
    4KB per thread!
    1120KB in regfile in GT3
256KB data cache per slice (L3 only)

Since SKL, capabilities for GPU-CPU exchange increased (with OpenCL 2.0 memory coherency can be coarse and fine-grained)

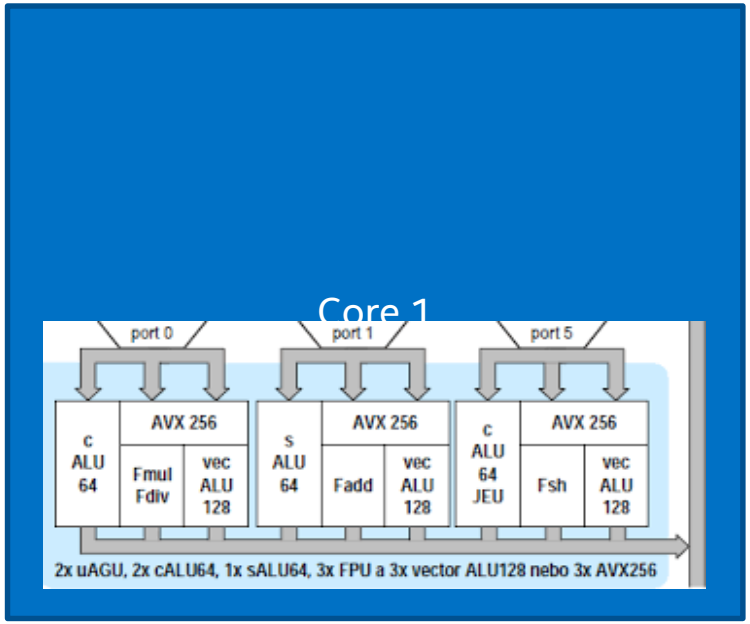
4TH GENERATION INTEL® CORE™ PROCESSORS – HD GRAPHICS





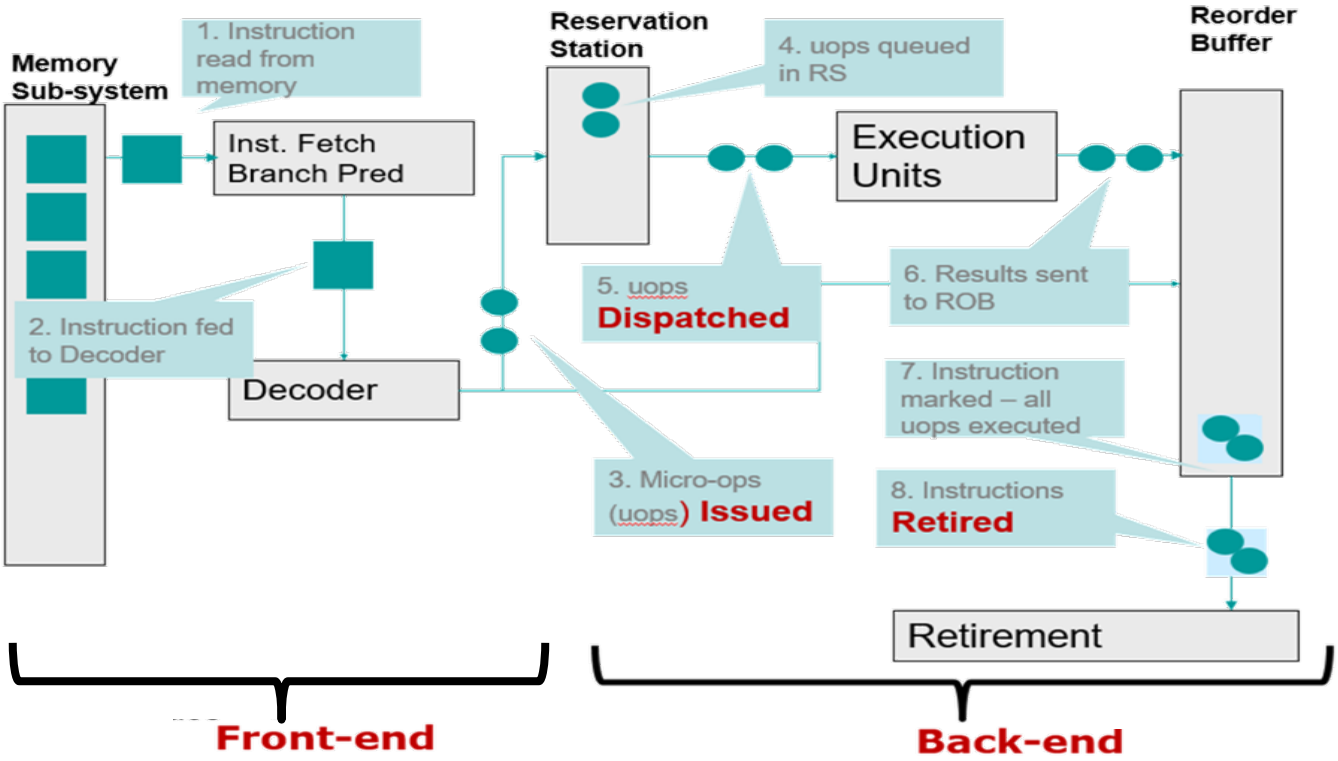
Conceptual structure of the Intel GPU

#6 INSTRUCTION-LEVEL PARALLELISM

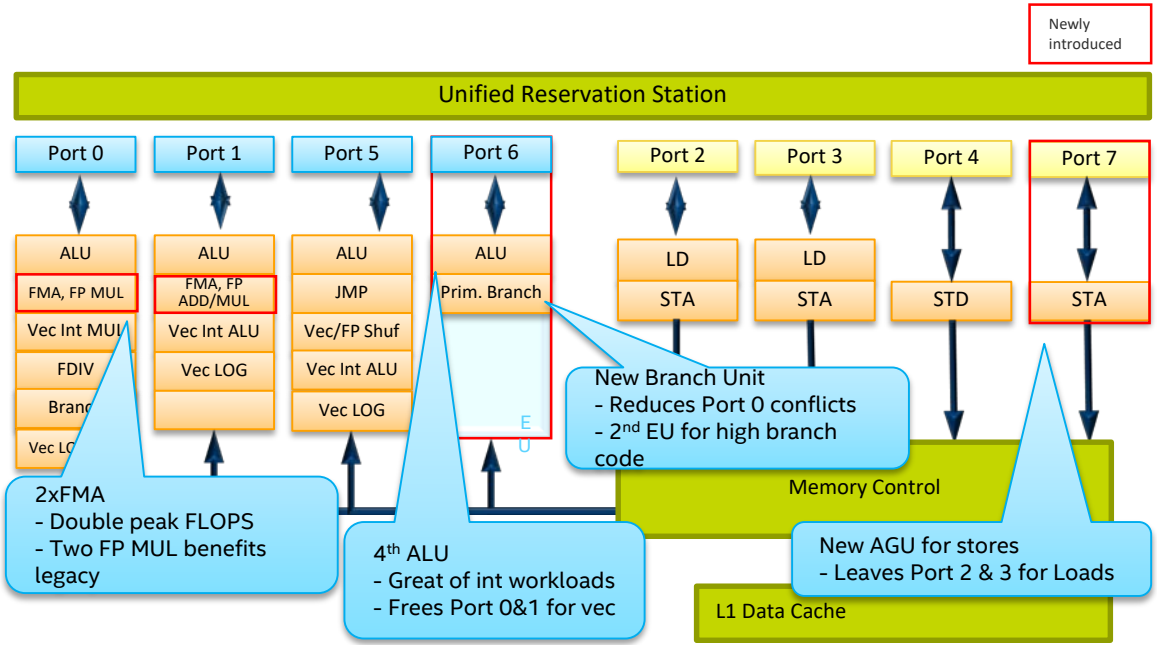


Levels of Parallelism
Node
Socket
Core / Thread-Level
(Hyperthreading)
GPU-CPU
Instruction

The life of a program instruction

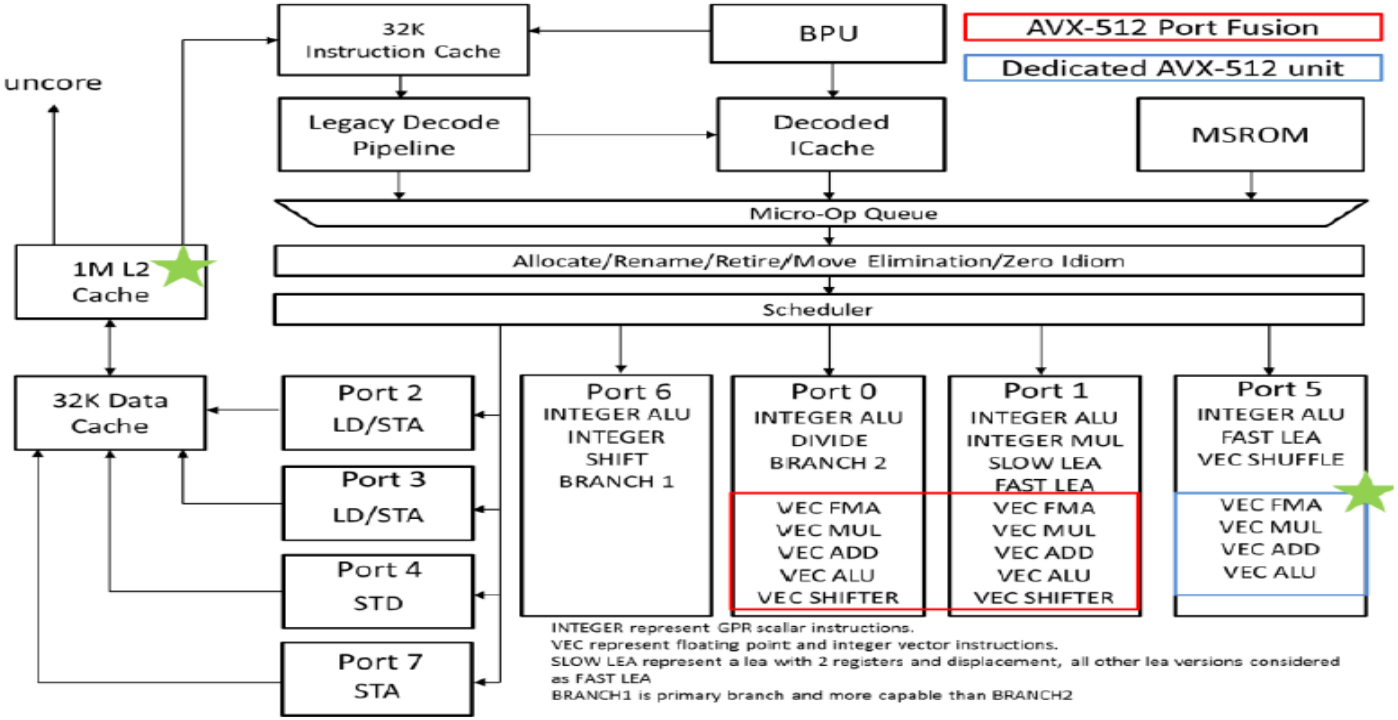


EXECUTION UNITS ON HASWELL/BROADWELL

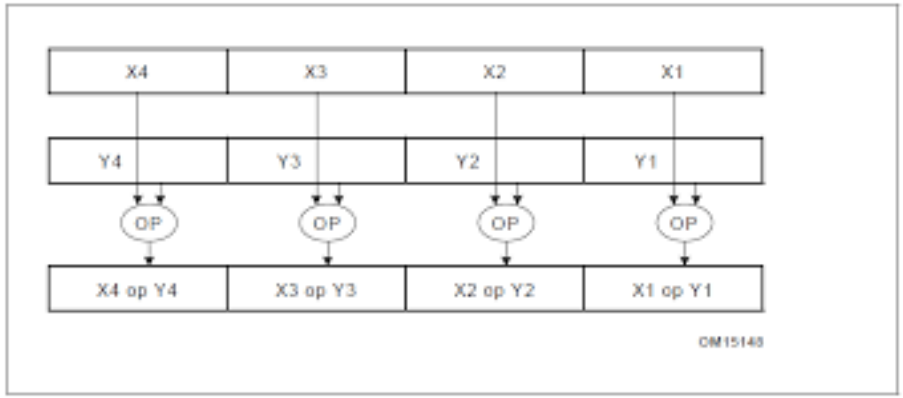


- Race to higher frequencies slowed down
- New logic introduced with new generations, leading to higher complexity
- Fused Multiply Add for performance
- Separate Address Generation Unit for memory address calculations

EXECUTION UNITS ON SKYLAKE SERVER

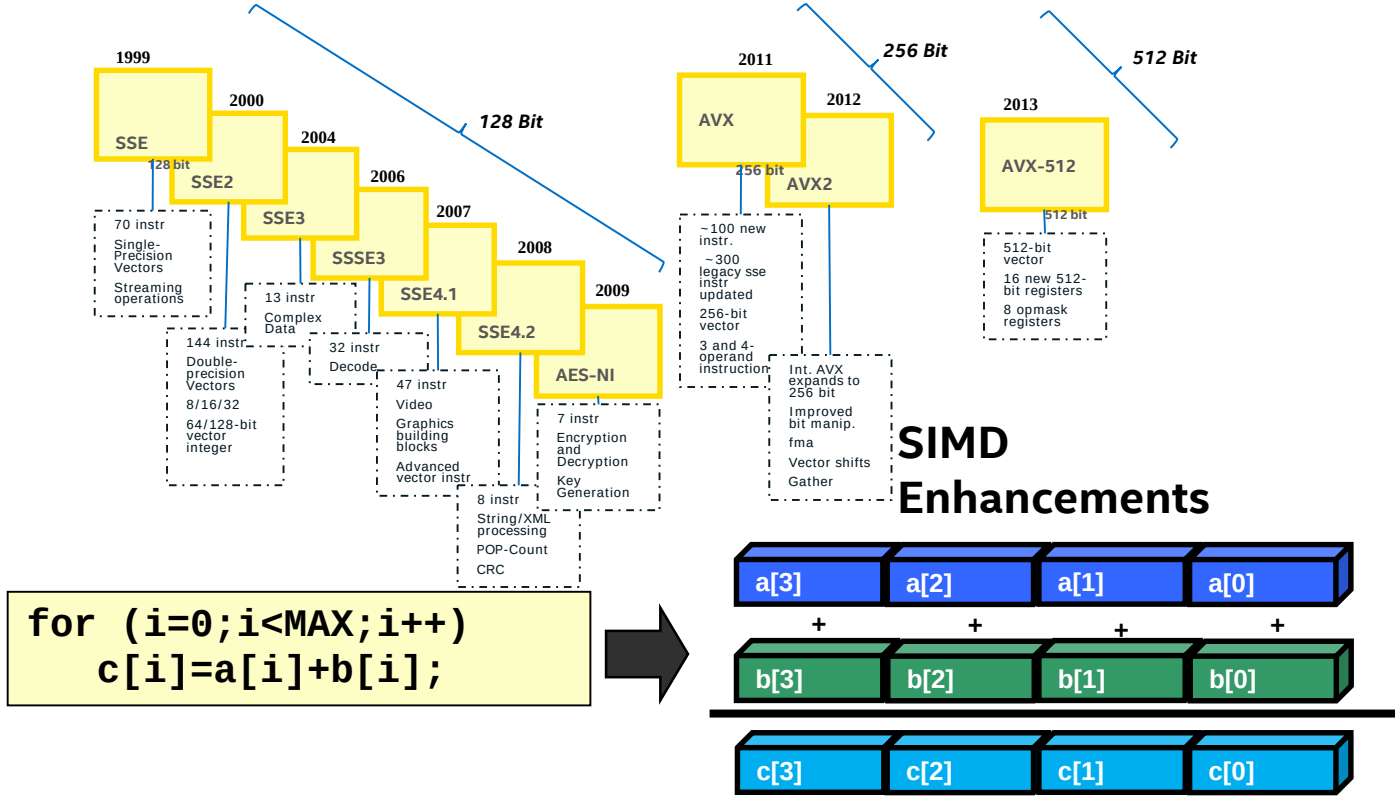


#7 DATA LEVEL PARALLELISM



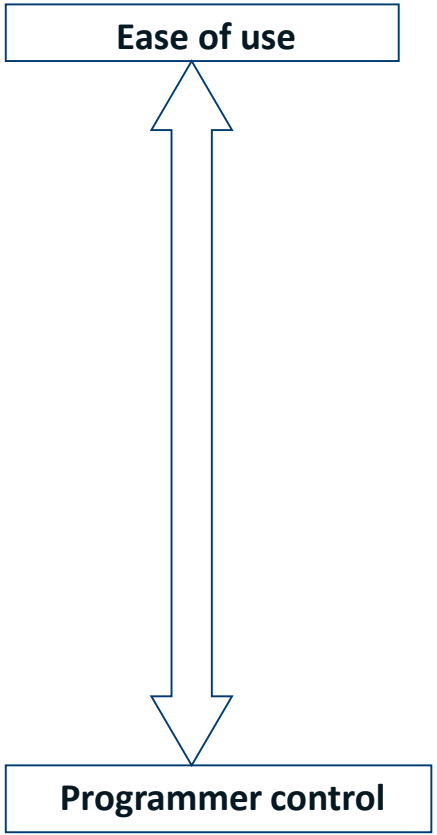
Levels of Parallelism
Node
Socket
Core / Thread-Level
(Hyperthreading)
GPU-CPU
Instruction
Data (Vectorisation)


intel
i386
A80386DX-33 IV
SX366
L1333356
INTEL©'85
intel
i387™DX
ADS
HLDA
RDY
M/IO


1999
SSE
128 bit
70 instr
Single-Precision Vectors
Streaming operations
2000
SSE2
144 instr
Double-precision Vectors
8/16/32
64/128-bit vector integer
2004
SSE3
13 instr
Complex Data
2006
SSSE3
32 instr
Decode
2007
SSE4.1
47 instr
Video
Graphics building blocks
Advanced vector instr
2008
SSE4.2
8 instr
String/XML processing
POP-Count
CRC
2009
AES-NI
7 instr
Encryption and Decryption
Key Generation
128 Bit
2011
AVX
256 bit
~100 new instr.
~300 legacy sse instr updated
256-bit vector
3 and 4-operand instruction
2012
AVX2
Int. AVX expands to 256 bit
Improved bit manip.
fma
Vector shifts
Gather
256 Bit
2013
AVX-512
512 bit
512-bit vector
16 new 512-bit registers
8 opmask registers
512 Bit
SIMD Enhancements
for (i=0;i<MAX;i++)
c[i]=a[i]+b[i];
a[3]
a[2]
a[1]
a[0]
+
b[3]
b[2]
b[1]
b[0]
c[3]
c[2]
c[1]
c[0]

DIFFERENT WAYS OF INSERTING VECTORISED CODE

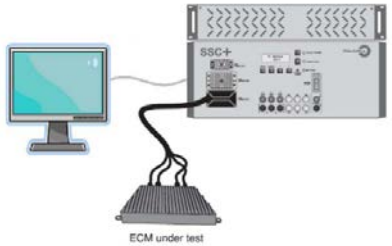
- Performance Libraries (e.g. IPP and MKL)
- Compiler: Fully automatic vectorization
- Cilk Plus Array Notation
- Compiler: Auto vectorization hints (#pragma ivdep, …)
- User Mandated Vectorization (SIMD Directive)
- Manual CPU Dispatch (__declspec(cpu_dispatch …))
- SIMD intrinsic class (F32vec4 add)
- Vector intrinsic (mm_add_ps())
- Assembler code (addps)



AN EXAMPLE

#1 Speedup by upgrading silicon

CPU	No Auto-Vectorisation	With Auto-Vectorisation	Speedup
P4	39.344	21.9	**1.80**
Core 2	5.546	0.515	**10.77**
Speedup	**7.09**	**45.52**	**76**



#2 Speedup by swapping compiler and enabling vectorisation

#3 Verified using VTune

CPU EVENT	Without Vect	With Vect
CPU_CLK_UNHALTED.CORE	16,641,000,448	1,548,000,000
INST_RETIRED.ANY	3,308,999,936	1,395,000,064
X87_OPS_RETIRED.ANY	250,000,000	0
SIMD_INST_RETIRED	0	763,000,000

SEVEN LEVELS OF HARDWARE-SUPPORTED PARALLELISM

Levels of Parallelism
Node
Socket
Core / Thread-Level
(Hyperthreading)
GPU-CPU
Instruction (by CPU internals)
Data (Vectorisation)

PERFORMANCE
IT'S ALL ABOUT MEMORY!

Another Viewpoint

WHICH LAPTOP SHOULD I BUY FOR BEST PERFORMANCE?

"Spend most of you money on extra memory"

MEMORY HIERARCHY

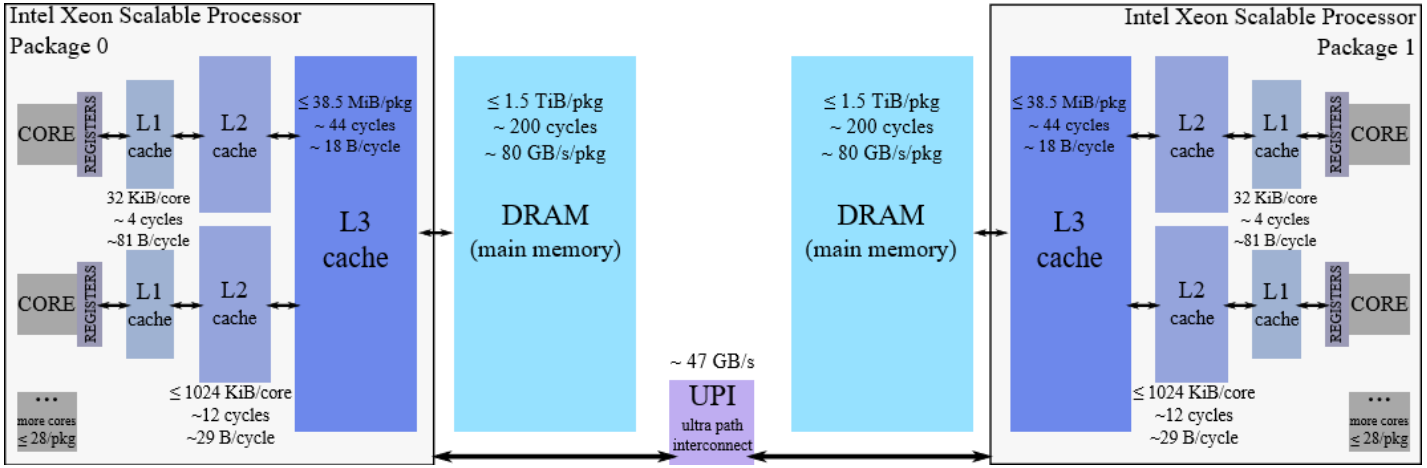
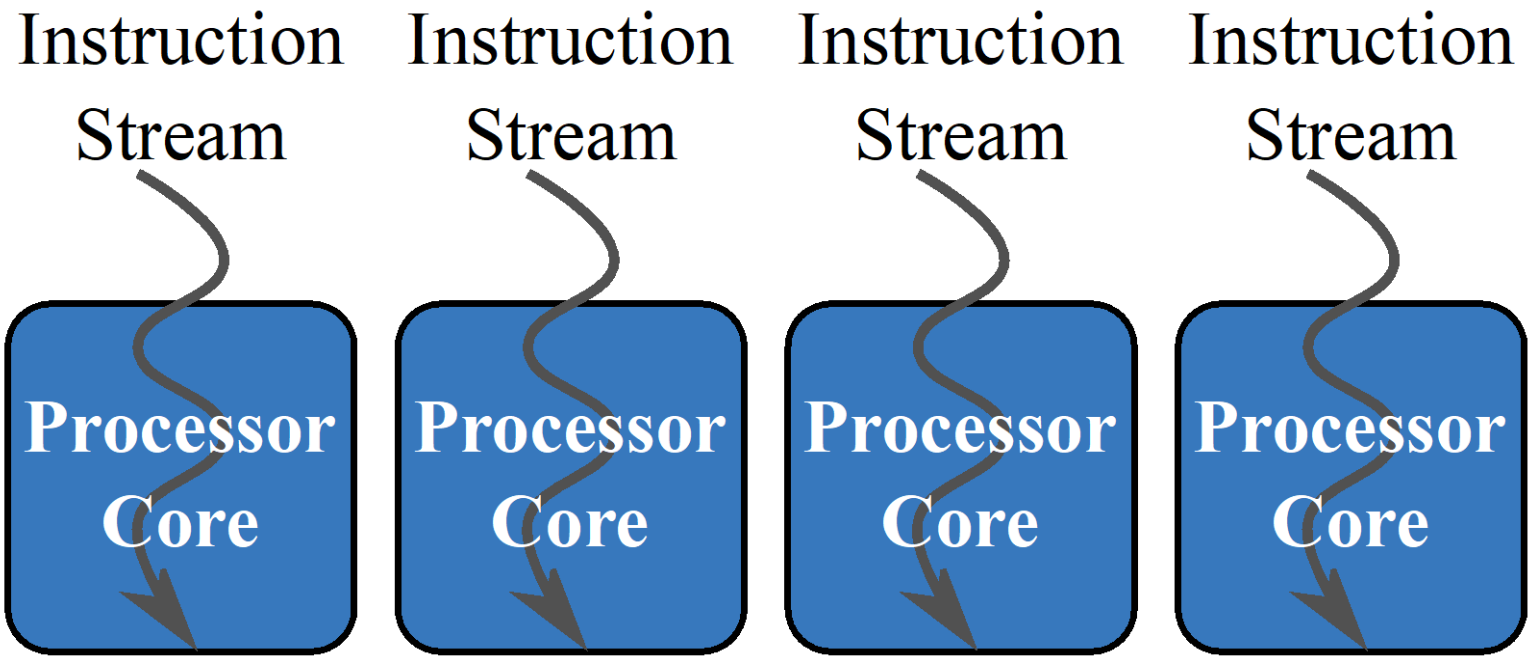


Diagram Source: colfax.com

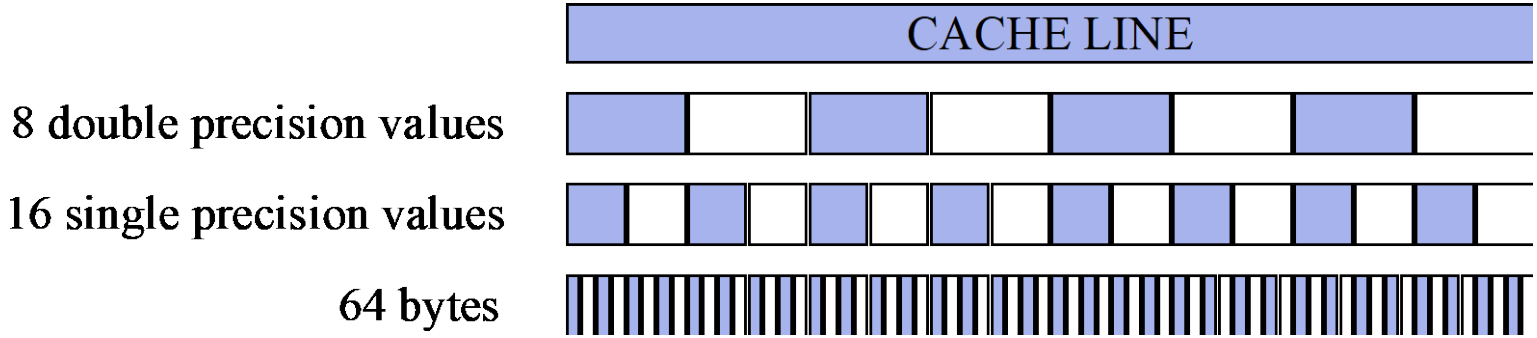
SEVEN LEVELS OF MEMORY HIERARCHY

Levels of Hierarchy	Size	Latency	Bandwidth
Registers	2048 bytes		
L1 Cache	32 KB	4 cycles	81 bytes/cycle
L2 Cache	256 – 1024 KB	14 cycles	29 bytes/cycle
L3 Cache	8 – 38MB	60 cycles	18 bytes/cycle
(On-chip Memory)	Maybe in future versions ?		
Local DRAM	1.5 TB	200 cycles	80 GB/s
Remote DRAM			47 GB/s

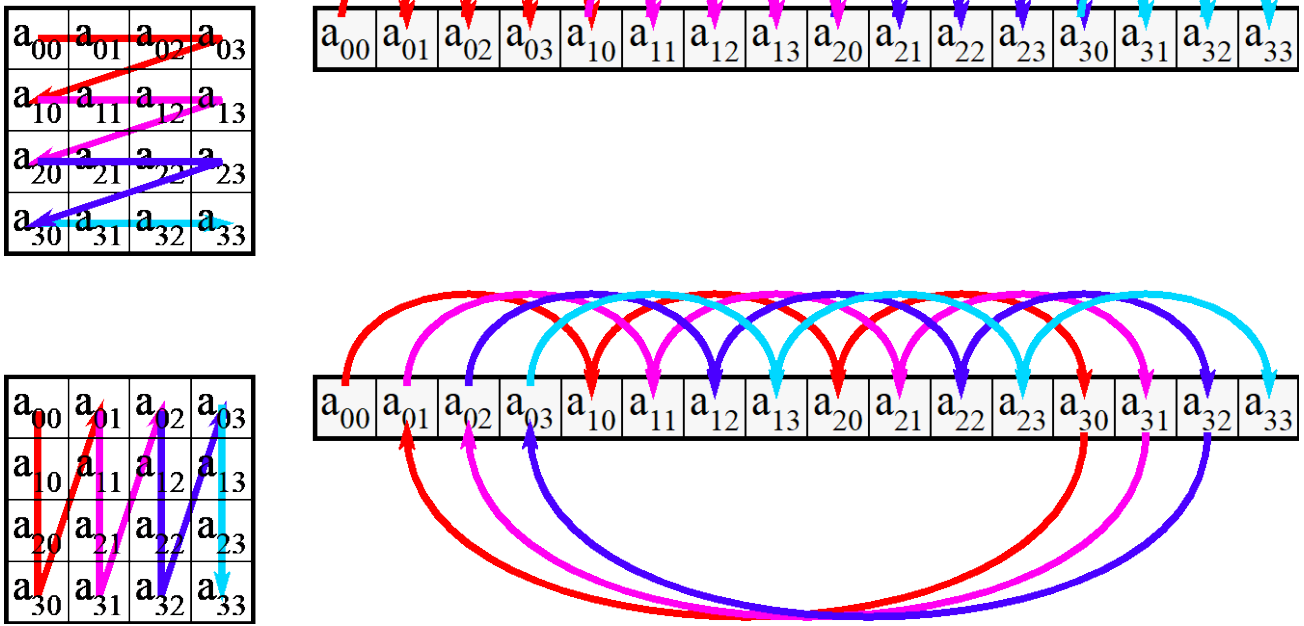
MEMORY CONTROLLERS AND THREADS



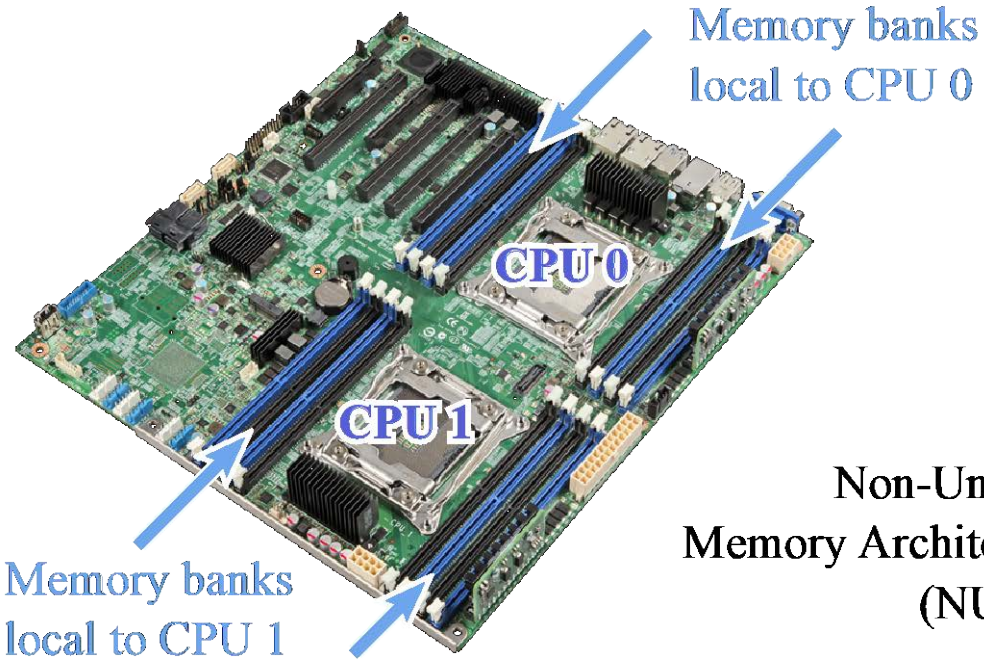
CACHE LINES AND VECTORIZATION



SEQUENTIAL ACCESS

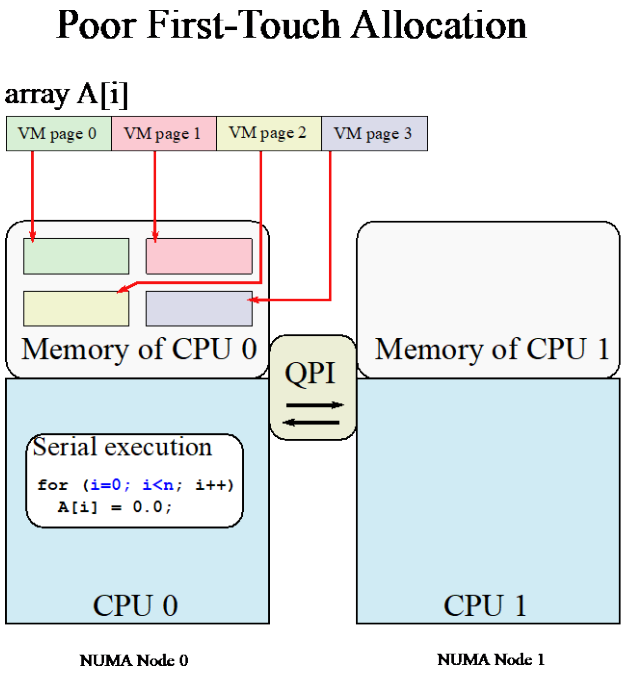


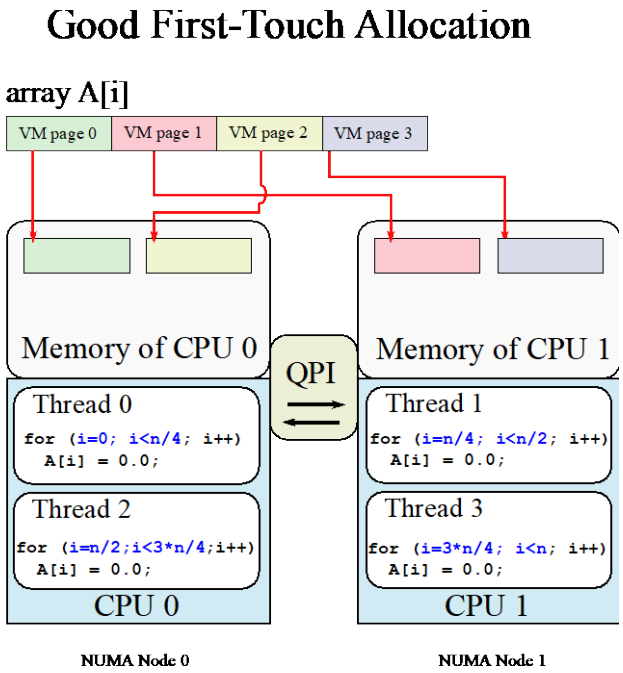
NUMA AND DATA ACCESS LOCALITY



Non-Uniform
Memory Architecture
(NUMA)

FIRST TOUCH POLICY AND ARRAY INITIALIZATION





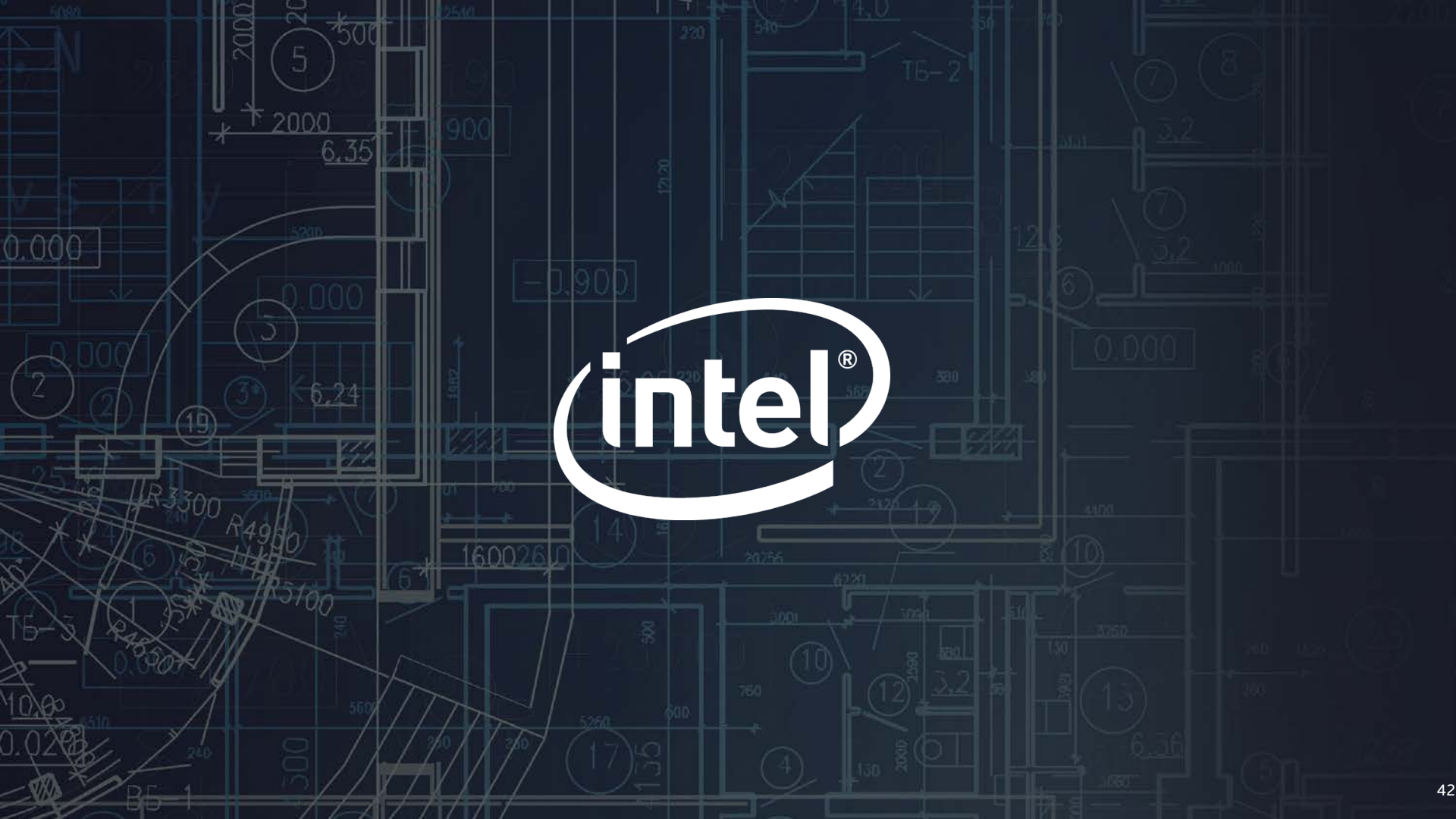
OPTIMIZATION IS …

Moving code from being

Memory Bound

to being

Compute Bound


intel®