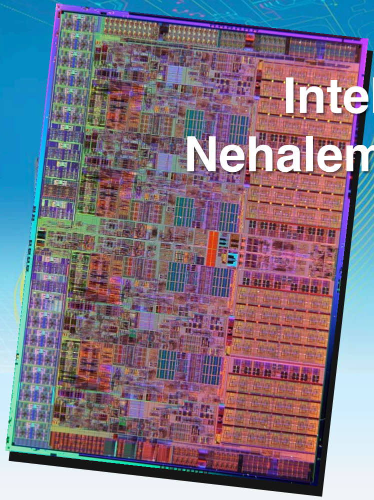


Intel® Next Generation Nehalem Microarchitecture

Andrey Semin
HPC Technology Manager
Intel Corporation, EMEA

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- 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.
- Penryn, Nehalem, Westmere, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user
- 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.
- Intel, Intel Inside, Xeon, Core 2, Core i7, Pentium, AVX and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
-

Risk Factors

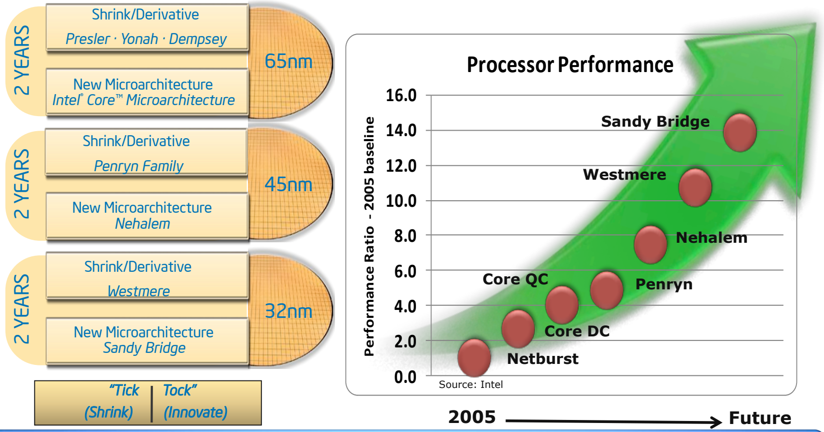
This presentation contains forward-looking statements that involve a number of risks and uncertainties. These statements do not reflect the potential impact of any mergers, acquisitions, divestitures, investments or other similar transactions that may be completed in the future. The information presented is accurate only as of today's date and will not be updated. In addition to any factors discussed in the presentation, the important factors that could cause actual results to differ materially include the following: Factors that could cause demand to be different from Intel's expectations include changes in business and economic conditions, including conditions in the credit market that could affect consumer confidence; customer acceptance of Intel's and competitors' products; changes in customer order patterns, including order cancellations; and changes in the level of inventory at customers. Intel's results could be affected by the timing of closing of acquisitions and divestitures. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Additionally, Intel is in the process of transitioning to its next generation of products on 45 nm process technology, and there could be execution issues associated with these changes, including product defects and errata along with lower than anticipated manufacturing yields. Revenue and the gross margin percentage are affected by the timing of new Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel's response to such actions; Intel's ability to respond quickly to technological developments and to incorporate new features into its products; and the availability of sufficient components from suppliers to meet demand. The gross margin percentage could vary significantly from expectations based on changes in revenue levels; product mix and pricing; capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; excess or obsolete inventory; manufacturing yields; changes in unit costs; impairments of long-lived assets, including manufacturing, assembly/test and intangible assets; and the timing and execution of the manufacturing ramp and associated costs, including start-up costs. Expenses, particularly certain marketing and compensation expenses, vary depending on the level of demand for Intel's products, the level of revenue and profits, and impairments of long-lived assets. Intel is in the midst of a structure and efficiency program that is resulting in several actions that could have an impact on expected expense levels and gross margin. Intel is also in the midst of forming Numonyx, a private, independent semiconductor company, together with STMicroelectronics N.V. and Francisco Partners L.P. A change in the financial performance of the contributed businesses could have a negative impact on our financial statements. Intel's equity proportion of the new company's results will be reflected on its financial statements below operating income and with a one quarter lag. The results could have a negative impact on Intel's overall financial results. Intel's results could be affected by the amount, type, and valuation of share-based awards granted as well as the amount of awards cancelled due to employee turnover and the timing of award exercises by employees. Intel's results could be impacted by adverse economic, social, political and physical/infrastructure conditions in the countries in which Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. A detailed discussion of these and other factors that could affect Intel's results is included in Intel's SEC filings, including the report on Form 10-Q for the quarter ended Sept. 29, 2007.

Agenda

- **Nehalem Design Philosophy**
- **Enhanced Processor Core**
- **New Instructions**
- **Optimization Guidelines and Software Tools**
- **New Platform Features**

- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- 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.
- Penryn, Nehalem, Westmere, Sandy Bridge and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user
- 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.
- Intel, Intel Inside, Xeon, Core, Pentium, AVX and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
-

Intel Tick-Tock Development Model: Delivering Leadership Multi-Core Performance



Silicon and Software Tools Unleash Performance

Nehalem Design Goals

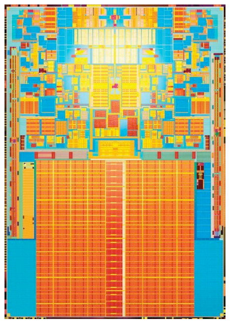
World class performance combined with superior energy efficiency – Optimized for:

A single, scalable, foundation optimized across each segment and power envelope

A Dynamic and Design Scalable Microarchitecture

Core Microarchitecture Recap

- Wide Dynamic Execution
  - 4-wide decode/rename/retire
- Advanced Digital Media Boost
  - 128-bit wide SSE execution units
- Intel HD Boost
  - New SSE4.1 Instructions
- Smart Memory Access
  - Memory Disambiguation
  - Hardware Prefetching
- Advanced Smart Cache
  - Low latency, high BW shared L2 cache



Nehalem builds on the great Core microarchitecture

Nehalem Micro-Architecture

A new dynamically scalable microarchitecture

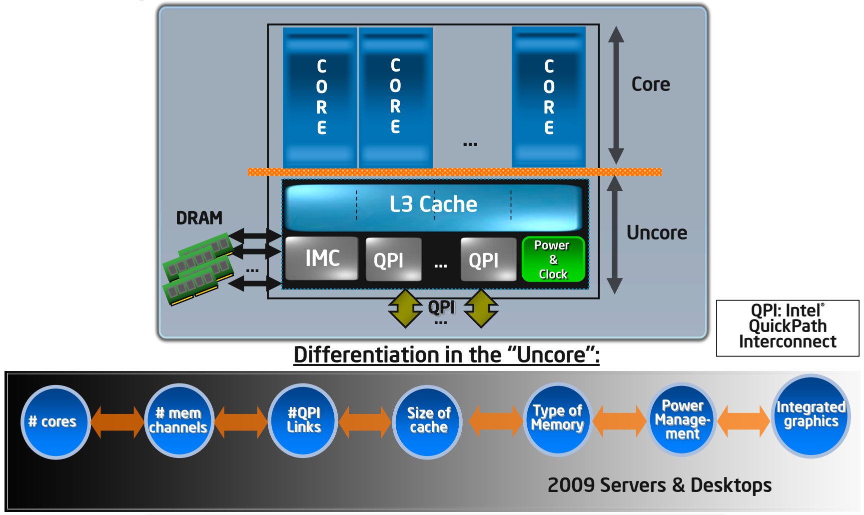
KEY FEATURES	*BENEFITS*
45nm Intel® multi-core processors *(2, 4, 8 core implementations planned)*	Energy efficient multi-core processing
Greater Instruction per clock and improved cache hierarchy	Faster Processing / core
Simultaneous Multi-Threading	More Threads / core
Dynamic Resource Scaling *Any unneeded cores automatically put into sleep mode; remaining operating cores get access to ALL cache, bandwidth and power/thermal budgets*	Lower power consumption during periods of low utilization
Turbo Mode *CPU operates at higher-than-stated frequency when operating below power and thermal design points*	Additional Processing boost during peak demand periods

FASTER cores … MORE cores/threads … DYNAMICALLY ADAPTABLE

Agenda

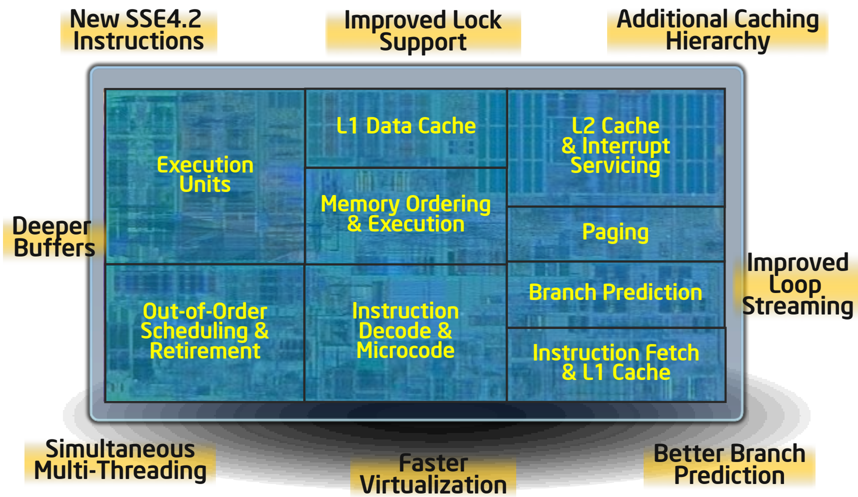
- Nehalem Design Philosophy
- **Enhanced Processor Core**
- New Instructions
- Optimization Guidelines and Software Tools
- New Platform Features

Designed For Modularity

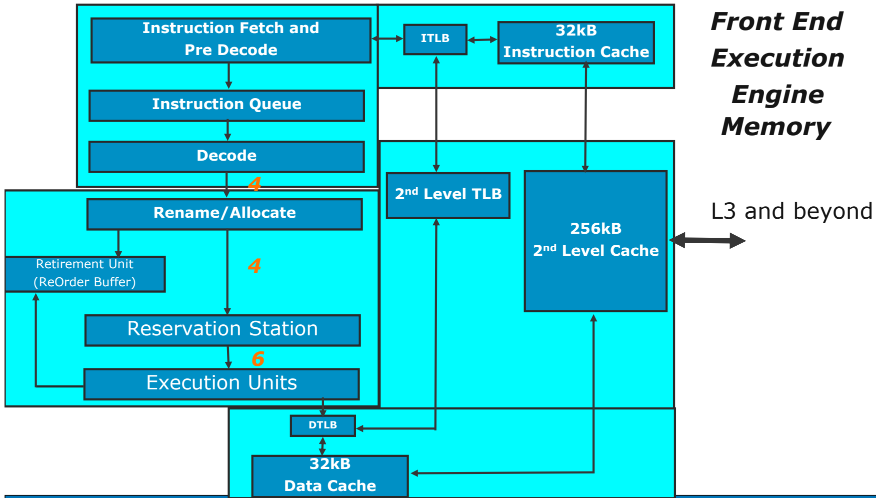


Optimal price / performance / energy efficiency for server, desktop and mobile products

Designed for Performance



Enhanced Processor Core

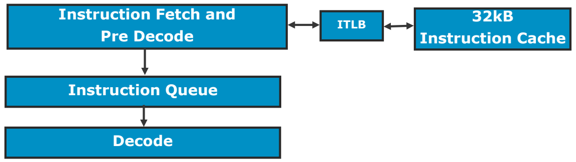


Front End
Execution Engine
Memory

L3 and beyond

Front-end

- Responsible for feeding the compute engine
  - Decode instructions
  - Branch Prediction
- Key Core 2 Features
  - 4-wide decode
  - Macrofusion
  - Loop Stream Detector



Macrofusion Recap

- Introduced in Core 2
- TEST/CMP instruction followed by a conditional branch treated as a single instruction
  - Decode as one instruction
  - Execute as one instruction
  - Retire as one instruction
- Higher ***performance***
  - Improves throughput
  - Reduces execution latency
- Improved ***power efficiency***
  - Less processing required to accomplish the same work

Nehalem Macrofusion

- Goal: Identify more macrofusion opportunities for increased ***performance*** and ***power efficiency***
- Support all the cases in Core 2 **PLUS**
  - CMP+Jcc macrofusion added for the following branch conditions
    - JL/JNGE
    - JGE/JNL
    - JLE/JNG
    - JG/JNLE
- Core 2 only supports macrofusion in 32-bit mode
  - Nehalem supports macrofusion in both 32-bit and 64-bit modes

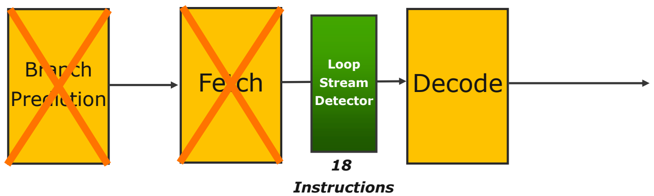
Increased macrofusion benefit on Nehalem

Front-end: Loop Stream Detector

Reminder

- Loops are very common in most software
- Take advantage of knowledge of loops in HW
  - *Decoding the same instructions over and over*
  - *Making the same branch predictions over and over*
- Loop Stream Detector identifies software loops
  - Stream from Loop Stream Detector instead of normal path
  - Disable unneeded blocks of logic for ***power savings***
  - ***Higher performance*** by removing instruction fetch limitations

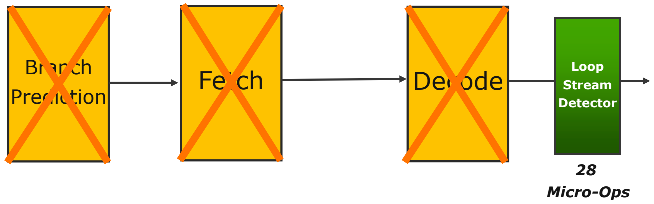
Core 2 Loop Stream Detector



Front-end: Loop Stream Detector
in Nehalem

- Same concept as in prior implementations
- ***Higher performance:*** Expand the size of the loops detected
- ***Improved power efficiency:*** Disable even more logic

Nehalem Loop Stream Detector



Branch Prediction Reminder

- Goal: Keep powerful compute engine fed
- Options:
  - Stall pipeline while determining branch direction/target
  - Predict branch direction/target and correct if wrong
- Minimize amount of time wasted correcting from incorrect branch predictions
  - ***Performance:***
    - Through higher branch prediction accuracy
    - Through faster correction when prediction is wrong
  - ***Power efficiency:*** Minimize number of speculative/incorrect micro-ops that are executed

Continued focus on branch prediction improvements

L2 Branch Predictor

- Problem: Software with a large code footprint not able to fit well in existing branch predictors
  - Example: Database applications
- Solution: Use multi-level branch prediction scheme
- Benefits:
  - Higher ***performance*** through improved branch prediction accuracy
  - Greater ***power efficiency*** through less mis-speculation

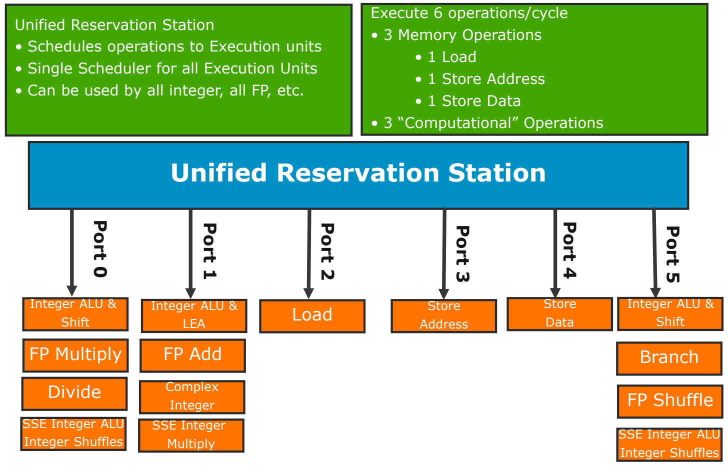
Renamed Return Stack Buffer (RSB)

- Instruction Reminder
  - CALL: Entry into functions
  - RET: Return from functions
- Classical Solution
  - Return Stack Buffer (RSB) used to predict RET
  - RSB can be corrupted by speculative path
- The ***Renamed RSB***
  - No RET mispredicts in the common case

Execution Engine

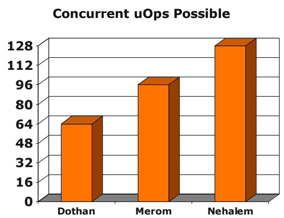
- Start with powerful Core 2 execution engine
  - Dynamic 4-wide Execution
  - Advanced Digital Media Boost
    - 128-bit wide SSE
  - HD Boost (Penryn)
    - SSE4.1 instructions
  - Super Shuffler (Penryn)
- Add Nehalem enhancements
  - Additional parallelism for higher performance

Execution Unit Overview



Increased Parallelism

- Goal: Keep powerful execution engine fed
- Nehalem increases size of out of order window by 33%
- Must also increase other corresponding structures



Structure	Merom	Nehalem	Comment
Reservation Station	32	36	Dispatches operations to execution units
Load Buffers	32	48	Tracks all load operations allocated
Store Buffers	20	32	Tracks all store operations allocated

Increased Resources for Higher Performance

Enhanced Memory Subsystem

- Start with great Core 2 Features
  - Memory Disambiguation
  - Hardware Prefetchers
  - Advanced Smart Cache
- New Nehalem Features
  - New TLB Hierarchy
  - Fast 16-Byte unaligned accesses
  - Faster Synchronization Primitives

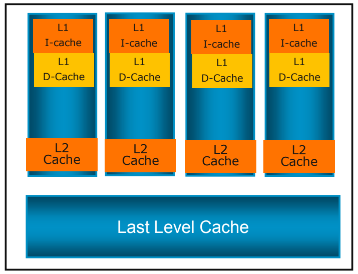
New TLB Hierarchy

- Problem: Applications continue to grow in data size
- Need to increase TLB size to keep the pace for performance
- Nehalem adds new low-latency unified 2nd level TLB

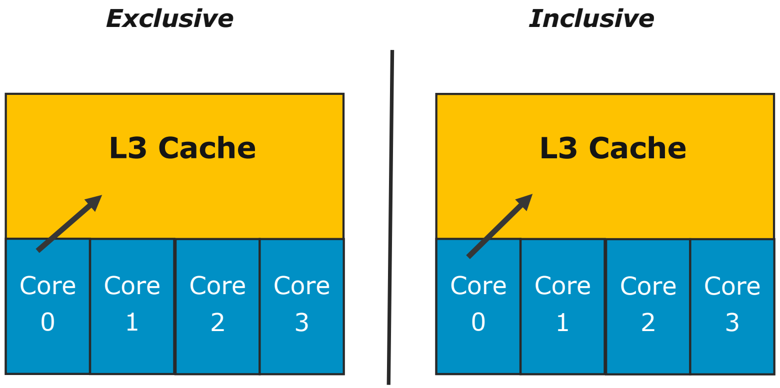
	# of Entries
1st Level Instruction TLBs	
Small Page (4k)	128
Large Page (2M/4M)	7 per thread
1st Level Data TLBs	
Small Page (4k)	64
Large Page (2M/4M)	32
New 2nd Level Unified TLB	
Small Page Only	512

Enhanced Cache Subsystem – New Memory Hierarchy

- New 3-level cache hierarchy
  - 1st level remains the same as Intel Core Microarchitecture
    - 32KB instruction cache
    - 32KB data cache
  - New L2 cache per core
    - 256 KB per core – holds data + instructions
    - Very low latency
  - New shared last level cache
    - Large size (8MB for 4-core)
    - Shared between all cores
      - ✓ Allows lightly threaded applications to use the entire cache
    - Inclusive Cache Policy
      - ✓ Minimize traffic from snoops
      - ✓ On cache miss, only check other cores if needed (data in modified state)

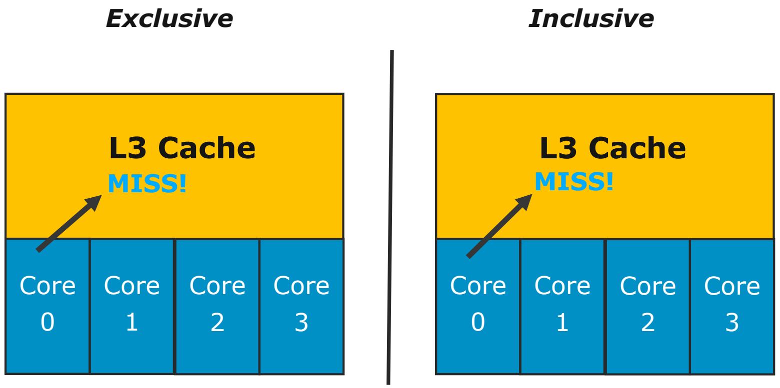


Inclusive vs. Exclusive Caches – Cache Miss



Data request from Core 0 misses Core 0's L1 and L2
Request sent to the L3 cache

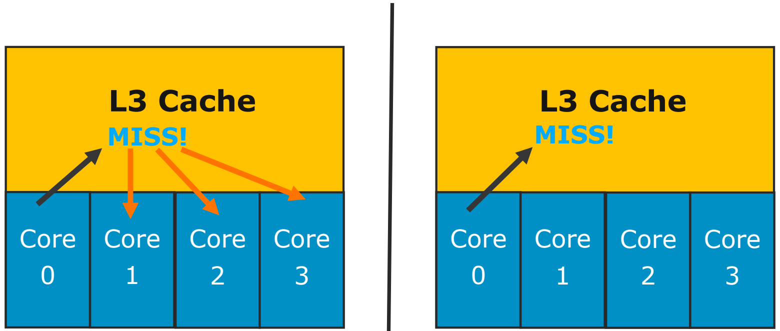
Inclusive vs. Exclusive Caches – Cache Miss



Core 0 looks up the L3 Cache
Data not in the L3 Cache

Inclusive vs. Exclusive Caches – Cache Miss

Exclusive ***Inclusive***



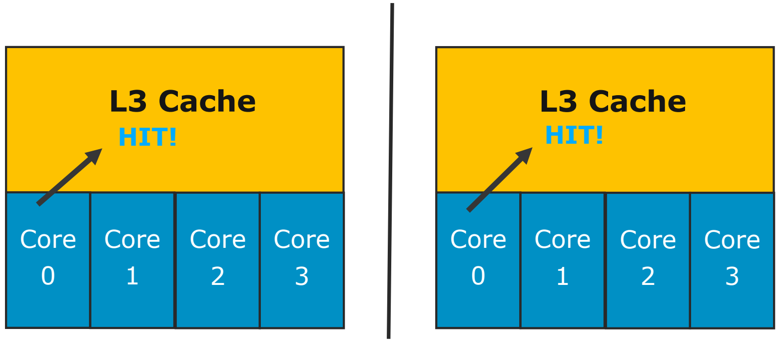
Must check other cores

Guaranteed data is not on-die

Greater ***scalability*** from inclusive approach

Inclusive vs. Exclusive Caches – Cache Hit

Exclusive ***Inclusive***



No need to check other cores

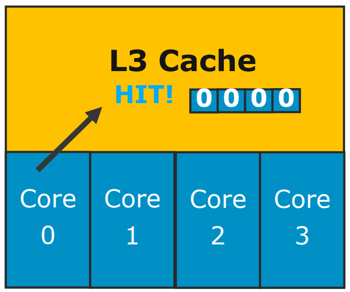
Data could be in another core

BUT Nehalem is smart…

Inclusive vs. Exclusive Caches – Cache Hit

- Maintain a set of "core valid" bits per cache line in the L3 cache
- Each bit represents a core
- If the L1/L2 of a core *may* contain the cache line, then core valid bit is set to "1"
- No snoops of cores are needed if no bits are set
- If more than 1 bit is set, line cannot be in Modified state in any core

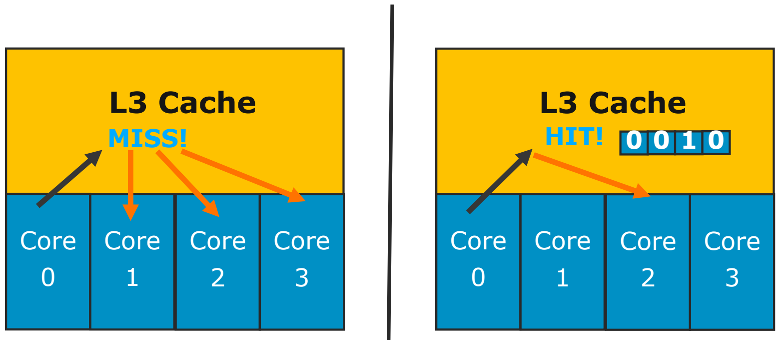
Inclusive



Core valid bits limit unnecessary snoops

Inclusive vs. Exclusive Caches – Read from other core

Exclusive ***Inclusive***

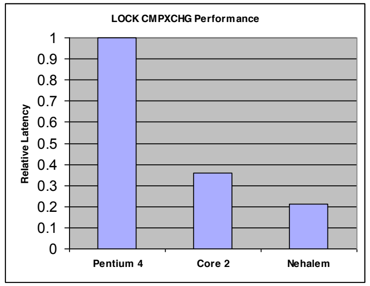


Must check all other cores

Only need to check the core whose core valid bit is set

Faster Synchronization Primitives

- Multi-threaded software becoming more prevalent
- ***Scalability*** of multi-thread applications can be limited by synchronization
- Synchronization primitives: LOCK prefix, XCHG
- Reduce synchronization latency for legacy software



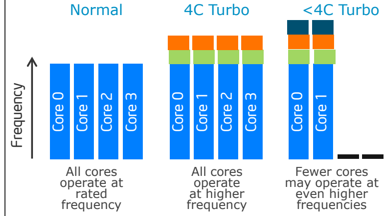
*Greater thread **scalability** with Nehalem*

Other Performance Enhancements

Intel Xeon® 5500 Series Processor (Nehalem-EP)

Intel® Turbo Boost Technology

Increases performance by increasing processor frequency and enabling faster speeds when conditions allow



Higher performance on demand

Intel® Hyper-Threading Technology

Increases performance for threaded applications delivering greater throughput and responsiveness

Higher performance for threaded workloads

† For notes and disclaimers, see performance and legal information slides at end of this presentation.

Hyper-Threading Implementation Details for Nehalem

- Multiple policies possible for implementation of SMT
- **Replicated** – Duplicate state for SMT
  - Register state
  - Renamed RSB
  - Large page ITLB
- **Partitioned** – Statically allocated between threads
  - Key buffers: Load, store, Reorder
  - Small page ITLB
- **Competitively shared** – Depends on thread's dynamic behavior
  - Reservation station
  - Caches
  - Data TLBs, 2nd level TLB
- **Unaware**
  - Execution units

Agenda

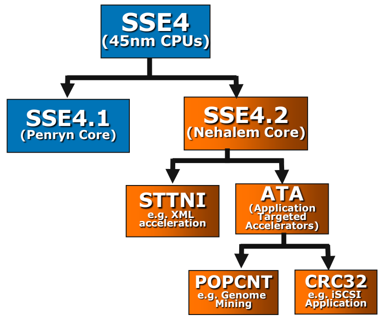
- **Nehalem Design Philosophy**
- **Enhanced Processor Core**
- **New Instructions**
- **Optimization Guidelines and Software Tools**
- **New Platform Features**

Extending Performance and Energy Efficiency
- SSE4.2 Instruction Set Architecture (ISA) Leadership

Accelerated String and Text Processing
- Faster XML parsing
- Faster search and pattern matching
- Novel parallel data matching and comparison operations

Accelerated Searching & Pattern Recognition of Large Data Sets
- Improved performance for Genome Mining, Handwriting recognition.
- Fast Hamming distance / Population count

New Communications Capabilities
- Hardware based CRC instruction
- Accelerated Network attached storage
- Improved power efficiency for Software I-SCSI, RDMA, and SCTP



What should the applications, OS and VMM vendors do?:
Understand the benefits & take advantage of new instructions in 2008.
Provide us feedback on instructions ISV would like to see for next generation of applications

STTNI - STring & Text New Instructions
Operates on strings of bytes or words (16b)

Equal Each Instruction

True for each character in Src2 if same position in Src1 is equal

```
Src1:    Test\tday
Src2:    tad tseT
Mask:    01101111
```

Equal Any Instruction

True for each character in Src2 if any character in Src1 matches

```
Src1:    Example\n
Src2:    atad tsT
Mask:    10100000
```

Ranges Instruction

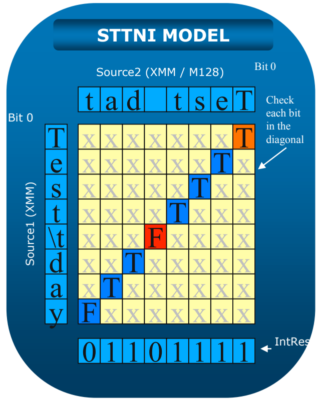
True if a character in Src2 is in at least one of up to 8 ranges in Src1

```
Src1:    AZ'0'9zzz
Src2:    taD tseT
Mask:    00100001
```

Equal Ordered Instruction

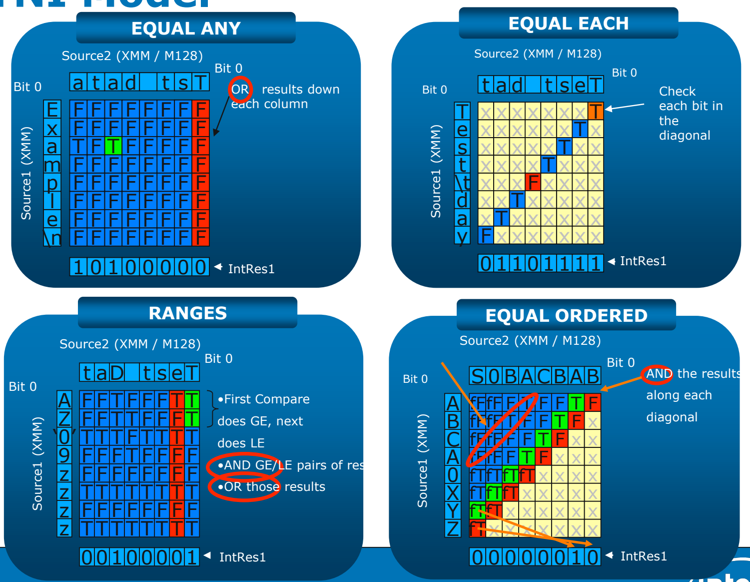
Finds the start of a substring (Src1) within another string (Src2)

```
Src1:    ABCA0XYZ
Src2:    S0BACBAB
Mask:    00000010
```



Projected 3.8x kernel speedup on XML parsing & 2.7x savings on instruction cycles

STTNI Model



Example Code For strlen()

```
string  equ     [esp + 4]
        mov     ecx,string            ; ecx -
        test    ecx,3                 ; test if
        je      short main_loop
str_misaligned:
        ; simple byte loop until string is al
        mov     al,byte ptr [ecx]
        add     ecx,1
        test    al,al
        je      short byte_3
        test    ecx,3
        jne     short str_misaligned
        add     eax,dword ptr 0       ; 5 b
        align   16                    ; should b
main_loop:
        mov     eax,dword ptr [ecx]   ; re
        mov     edx,7efefeffh
        add     edx,eax
        xor     eax,-1
        xor     eax,edx
        add     ecx,4
        test    eax,81010100h
        je      short main_loop
        ; found zero byte in the loop
        mov     eax,[ecx - 4]
        test    al,al                 ; is it byte
        je      short byte_0
        test    ah,ah                 ; is it byte
        je      short byte_1
        test    eax,00ff0000h         ; is it byte
```

```
        je      short byte_2
        test    eax,0ff000000h
; is it byte 3
        je      short byte_3
        jmp     short main_loop
; taken if bits 24-30 are clear and
; 31 is set
byte_3:
        lea     eax,[ecx - 1]
        mov     ecx,string
        sub     eax,ecx
        ret
byte_2:
        lea     eax,[ecx - 2]
        mov     ecx,string
        sub     eax,ecx
        ret
byte_1:
        lea     eax,[ecx - 3]
        mov     ecx,string
        sub     eax,ecx
        ret
byte_0:
        lea     eax,[ecx - 4]
        mov     ecx,string
        sub     eax,ecx
        ret
strlen  endp
        end
```

STTNI Version

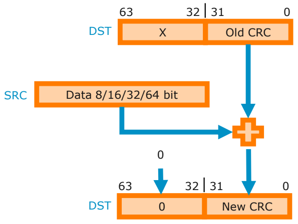
```
int sttni_strlen(const char * src)
{

char eom_vals[32] = {1, 255, 0};

__asm{

mov       eax, src

movdqu    xmm2, eom_vals

xor       ecx, ecx

topofloop:

add       eax, ecx

movdqu    xmm1, OWORD PTR[eax]

pcmpistri xmm2, xmm1, imm8

jnz       topofloop

endofstring:

add       eax, ecx

sub       eax, src
ret
```

Current Code: Minimum of 11 instructions; Inner loop processes 4 bytes with 8 instructions
STTNI Code: Minimum of 10 instructions; A single inner loop processes 16 bytes with only 4 instructions

ATA - Application Targeted Accelerators

CRC32

Accumulates a CRC32 value using the iSCSI polynomial

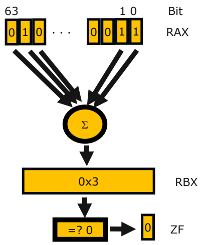


One register maintains the running CRC value as a software loop iterates over data.
Fixed CRC polynomial = 11EDC6F41h

Replaces complex instruction sequences for CRC in Upper layer data protocols:
• iSCSI, RDMA, SCTP

POPCNT

POPCNT determines the number of nonzero bits in the source.



POPCNT is useful for speeding up fast matching in data mining workloads including:
- DNA/Genome Matching
- Voice Recognition

ZFlag set if result is zero. All other flags (C,S,O,A,P) reset

Enables enterprise class data assurance with high data rates in networked storage in any user environment.

CRC32 Preliminary Performance

CRC32 optimized Code

```
crc32c_sse42_optimized_version(uint32 crc, unsigned
char const *p, size_t len)
{ // Assuming len is a multiple of 0x10
  asm("pusha");
  asm("mov %0, %%eax" :: "m" (crc));
  asm("mov %0, %%ebx" :: "m" (p));
  asm("mov %0, %%ecx" :: "m" (len));
  asm("1:");
   // Processing four byte at a time: Unrolled four times:
    asm("crc32 %eax, 0x0(%ebx)");
    asm("crc32 %eax, 0x4(%ebx)");
    asm("crc32 %eax, 0x8(%ebx)");
    asm("crc32 %eax, 0xc(%ebx)");
    asm("add $0x10, %ebx")2;
    asm("sub $0x10, %ecx");
    asm("jecxz 2f");
    asm("jmp 1b");
  asm("2:");
  asm("mov %%eax, %0" : "=m" (crc));
  asm("popa");
  return crc;
}}
```

- Preliminary tests involved Kernel code implementing CRC algorithms commonly used by iSCSI drivers.
- 32-bit and 64-bit versions of the Kernel under test
- 32-bit version processes 4 bytes of data using 1 CRC32 instruction
- 64-bit version processes 8 bytes of data using 1 CRC32 instruction
- Input strings of sizes 48 bytes and 4KB used for the test

	32 - bit	64 - bit
Input Data Size = 48 bytes	6.53 X	9.85 X
Input Data Size = 4 KB	9.3 X	18.63 X

Preliminary Results show CRC32 instruction outperforming the fastest CRC32C software algorithm by a big margin

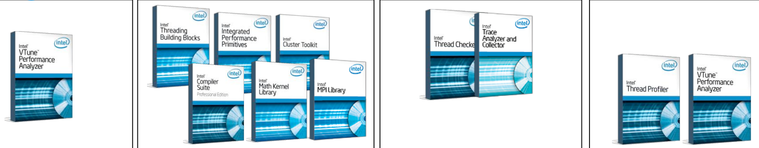
Agenda

- **Nehalem Design Philosophy**
- **Enhanced Processor Core**
- **New Instructions**
- **Optimization Guidelines and Software Tools**
- **New Platform Features**

Software Optimization Guidelines

- Most optimizations for Core microarchitecture still hold
- Examples of new optimization guidelines:
  - 16-byte unaligned loads/stores
  - Enhanced macrofusion rules
  - NUMA optimizations
- Nehalem SW Optimization Guide are published
- Intel Compiler supports settings for Nehalem optimizations (e.g. -xSSE4.2 option)

Simplified Many-core Development with Intel® Tools

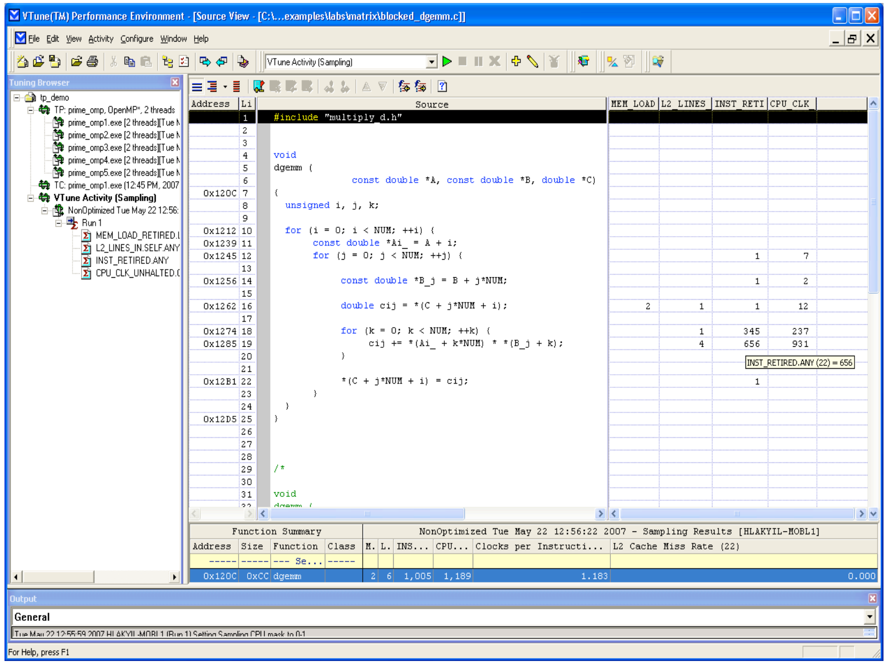


Insight	*Methods*	*Confidence*	*Performance*
• **VTune™ Analyzer** • Find the code that can benefit from threading and multicore • Find hotspots that limit performance	• **Compilers / Libraries** • MKL • TBB • IPP • **Clients** • OpenMP • Ct research • Hybrid methods • **Clusters** • MPI • Hybrid methods	• **Intel® Thread Checker** • Find deadlocks and race conditions **Intel® Trace Analyzer and Collector** • Event based tracing	• **VTune Analyzer** • Tune for performance and scalability • **Intel® Thread Profiler** • Visualize efficiency of threaded code
Architectural Analysis	**Introduce Parallelism**	**Confidence/Correctness**	**Optimize/Tune**

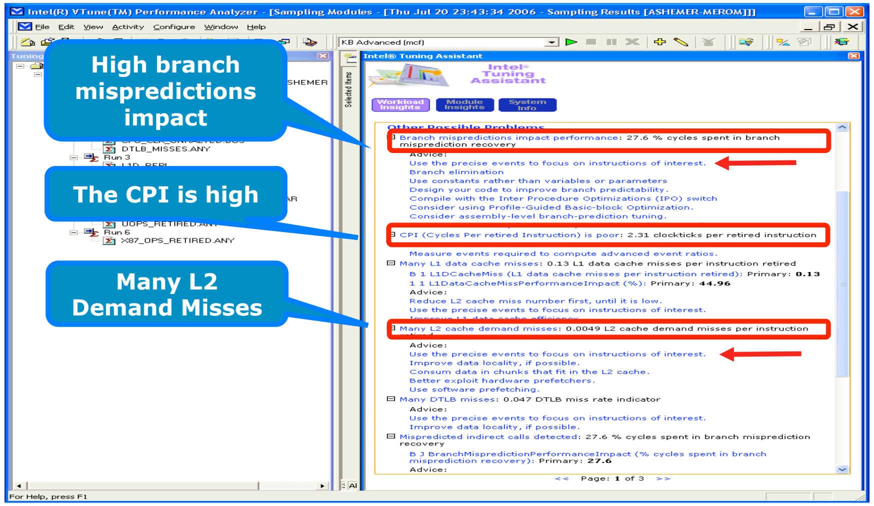
☑ **Windows; Linux; Mac OS**

Tools Support of New Instructions

- Intel Compiler 10.x+ supports the new instructions
  - SSE4.2 supported via intrinsics
  - Inline assembly supported on both IA-32 and Intel64 targets
  - Necessary to include required header files in order to access intrinsics
    - ✓ <tmmintrin.h> for Supplemental SSE3
    - ✓ <smmintrin.h> for SSE4.1
    - ✓ <nmmintrin.h> for SSE4.2
- Intel Library Support
  - XML Parser Library released in Fall '08
  - IPP is investigating possible usages of new instructions
- Microsoft Visual Studio 2008 VC++
  - SSE4.2 supported via intrinsics
  - Inline assembly supported on IA-32 only
  - Necessary to include required header files in order to access intrinsics
    - ✓ <tmmintrin.h> for Supplemental SSE3
    - ✓ <smmintrin.h> for SSE4.1
    - ✓ <nmmintrin.h> for SSE4.2
  - VC++ 2008 tools masm, msdis, and debuggers recognize the new instructions

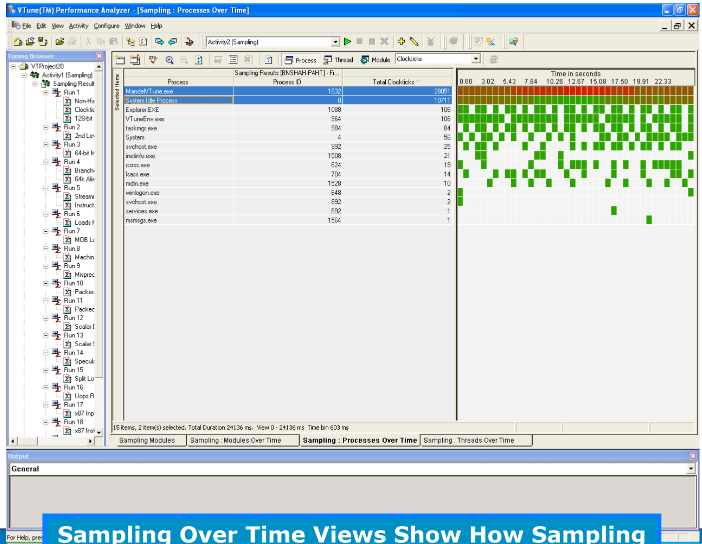


VTune Tuning Assist View



Use specific events to focus on instructions of interest.

VTune Sampling Over Time View

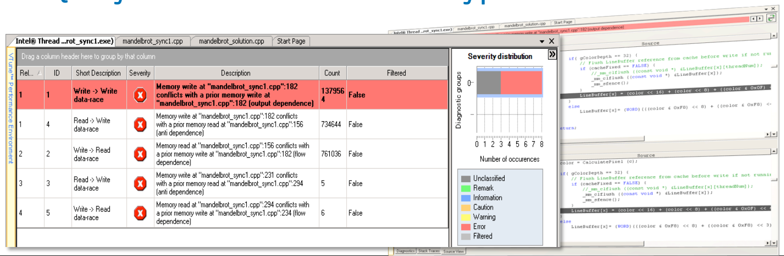


Sampling Over Time Views Show How Sampling Data Changes Over Time

Intel® Thread Checker

Deliver Multi-Threaded Optimized Code

- Detect hidden potential non-deterministic multithreading errors such as deadlocks and data races
- Analyze the results using Visual Studio* integration or a standalone graphical interface.
- Quickly drill down to the source to identify problematic lines of code



Use the Same Toolset for 32/64 bit on Windows*, Linux* and Mac OS* X

Intel® Software Development Products (● = Currently Available)		Itanium: Windows* / Visual Studio*	Itanium: Linux* / GCC*	Windows / Visual Studio	Linux / GCC	Mac OS* / Xcode*
Compilers	C++	●	●	●	●	●
	Fortran	●	●	●	●	●
Performance Analyzers	VTune® Performance Analyzer	●	●	●	●	
Performance Libraries	Integrated Performance Primitives	●	●	●	●	●
	Math Kernel Library	●	●	●	●	●
	Mobile Platform SDK			●		
Threading Analysis Tools	Thread Checker			●	●	
	Thread Profiler			●		
Cluster Tools	MPI Library	●	●	●	●	
	Trace Analyzer and Collector	●	●	●	●	
	Math Kernel Library Cluster Edition	●	●	●	●	
	Cluster Toolkit	●	●	●	●	
XML Tools**	XML Software Suite 1.0		●	●	●	

From Servers to Mobile / Wireless Computing, Intel® Software Development Products Enable Application Development Across Intel® Platforms

** Additional XML tools information can be found at www.intel.com/software/xml

Agenda

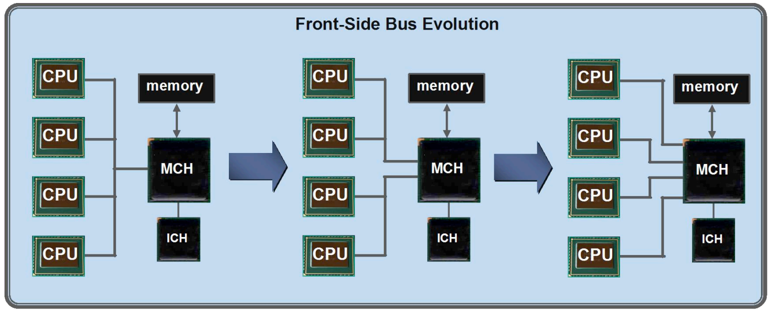
- Nehalem Design Philosophy
- Enhanced Processor Core
- New Instructions
- Optimization Guidelines and Software Tools
- **New Platform Features**

Feeding the Execution Engine

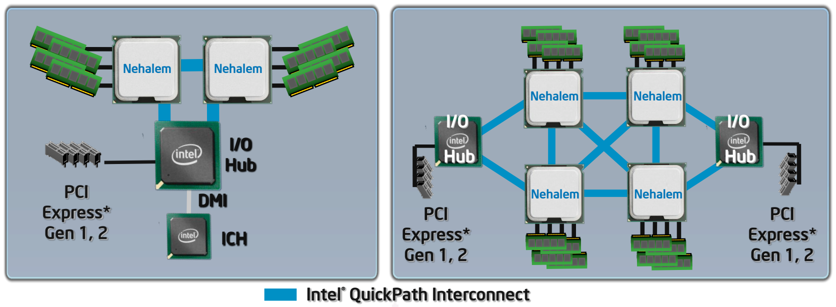
- Powerful 4-wide dynamic execution engine
- Need to keep providing fuel to the execution engine
- Nehalem Goals
  - ***Low latency*** to retrieve data
    - Keep execution engine fed w/o stalling
  - High data ***bandwidth***
    - Handle requests from multiple cores/threads seamlessly
  - ***Scalability***
    - Design for increasing core counts
- Combination of great ***cache hierarchy*** and ***new platform***

Nehalem designed to feed the execution engine

Previous Platform Architecture



Nehalem Based System Architecture

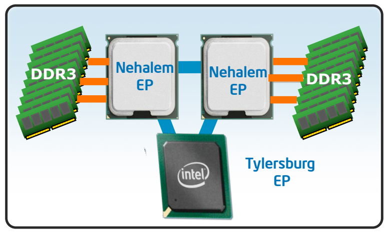


Nehalem Microarchitecture
Integrated Intel® QuickPath Memory Controller
Intel® QuickPath Interconnect
Buffered or Un-buffered Memory
PCI Express* Generation 2
Optional Integrated Graphics

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

Integrated Memory Controller (IMC)

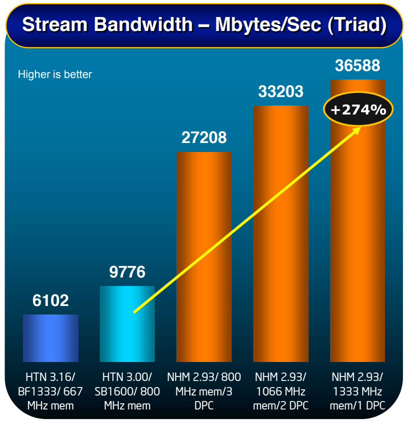
- Memory controller optimized per market segment
- Initial Nehalem products
  - Native DDR3 IMC
  - Up to 3 channels per socket
  - Speeds up to DDR3-1333
    - Massive ***memory bandwidth***
  - Designed for ***low latency***
  - Support RDIMM and UDIMM
  - RAS Features
- Future products
  - ***Scalability***
    - Vary # of memory channels
    - Increase memory speeds
    - Buffered and Non-Buffered solutions
  - Market specific needs
    - Higher memory capacity
    - Integrated graphics

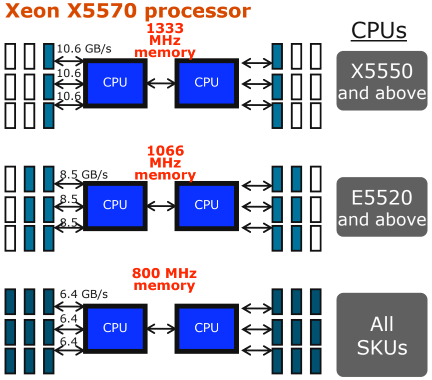


Significant performance through new IMC

Intel® Xeon® Processor 5500 series based Server platforms

Stream Bandwidth for Xeon X5570 processor





(DPC – Dimms Per Channel)

Nehalem-EP memory Bandwidth for different configuration

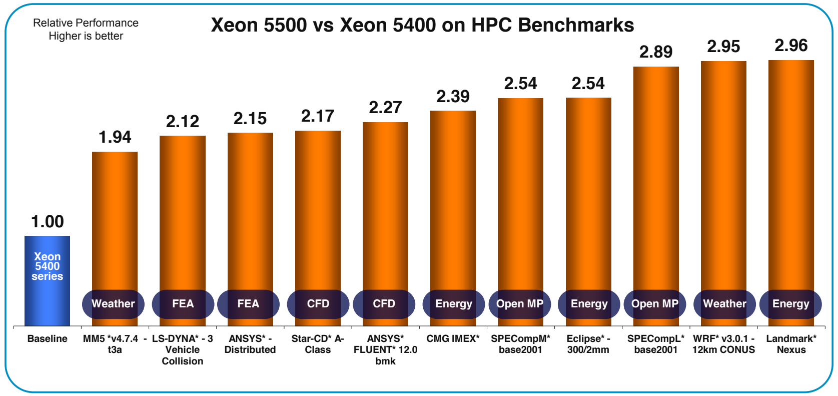
Memory speed	800 MHz			1066 MHz		1333 MHz
	1 DPC	2 DPC	3 DPC	1 DPC	2 DPC	1 DPC
Stream Triad	27748	26565	27208	33723	33203	36588

Massive Increase in Platform Bandwidth

Source: Intel internal measurement – March 2009

Intel® Xeon® Processor 5500 series based Server platforms

HPC Performance comparison to Xeon 5400 Series

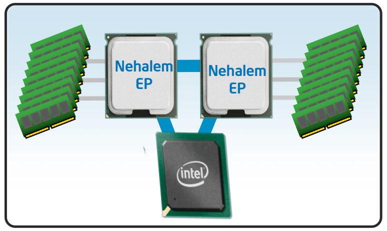


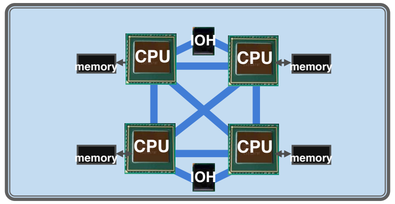
Source: Published/submitted/approved results March 30, 2009. See backup for additional details

Exceptional gains on HPC applications

QuickPath Interconnect

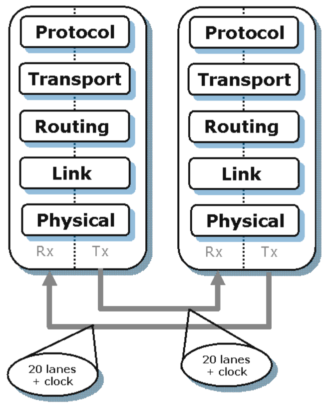
- Nehalem introduces new QuickPath Interconnect (QPI)
- ***High bandwidth***, ***low latency*** point to point interconnect
- Up to 6.4 GT/sec initially
  - 6.4 GT/sec -> 12.8 GB/sec
  - Bi-directional link -> 25.6 GB/sec per link
  - Future implementations at even higher speeds
- Highly ***scalable*** for systems with varying # of sockets





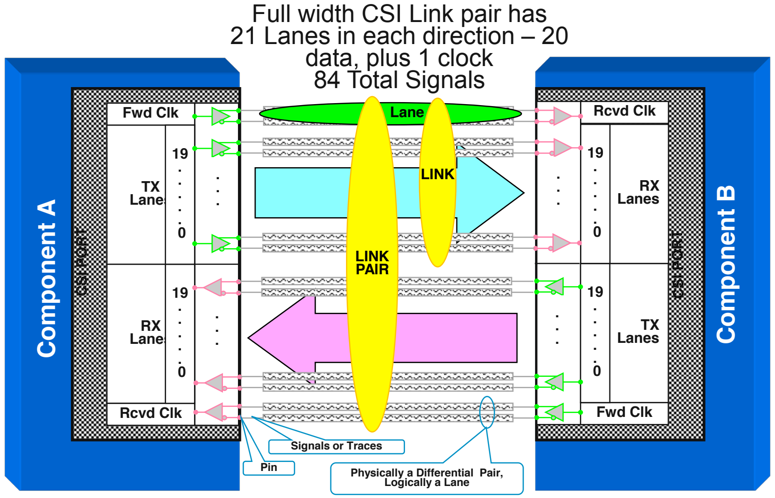
Layered Architecture

- Functionality is partitioned into five-layers, each layer performing a well-defined set of non-overlapping functions
  - Protocol Layer is the set of rules for exchanging packets between devices
  - Transport Layer provides advanced routing capability for the future*
  - Routing Layer provides framework for directing packet through the fabric
  - Link Layer is responsible for reliable transmission and flow control
  - Physical Layer carries the signals and transmission/receiver support logic



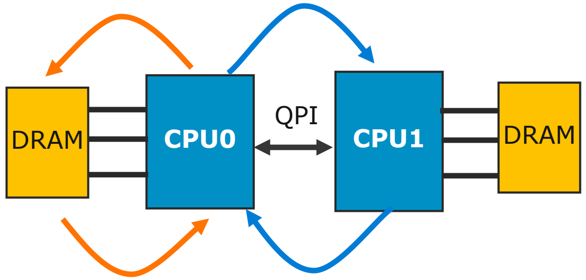
Modularity aids interconnect longevity & eases component design

QPI Link – Logical View



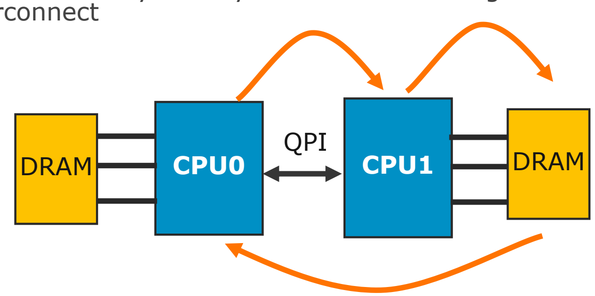
Local Memory Access

- CPU0 requests cache line X, not present in any CPU0 cache
  - CPU0 requests data from its DRAM
  - CPU0 snoops CPU1 to check if data is present
- Step 2:
  - DRAM returns data
  - CPU1 returns snoop response
- Local memory latency is the maximum latency of the two responses
- Nehalem optimized to keep key latencies close to each other



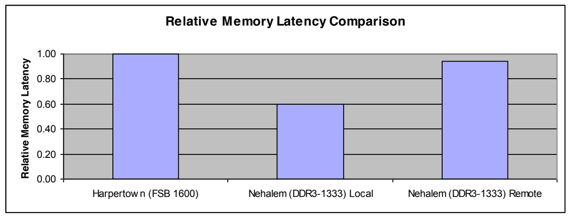
Remote Memory Access

- CPU0 requests cache line X, not present in any CPU0 cache
  - CPU0 requests data from CPU1
  - Request sent over QPI to CPU1
  - CPU1's IMC makes request to its DRAM
  - CPU1 snoops internal caches
  - Data returned to CPU0 over QPI
- Remote memory latency a function of having a low latency interconnect



Memory Latency Comparison

- ***Low memory latency*** critical to high performance
- Design integrated memory controller for low latency
- Need to optimize both local and remote memory latency
- Nehalem delivers
  - Huge reduction in local memory latency
  - Even remote memory latency is fast
- Effective memory latency depends per application/OS
  - Percentage of local vs. remote accesses
  - Nehalem has lower latency regardless of mix



Summary

- Nehalem – The 45nm Tock designed for
  - ***Power Efficiency***
  - ***Scalability***
  - ***Performance***
- Enhanced Processor Core
- Brand New Platform Architecture
- Extending x86 ISA Leadership
- Tools Available to support new processors feature and ISA

- More web based info: http://www.intel.com/technology/architecture-silicon/next-gen/index.htm