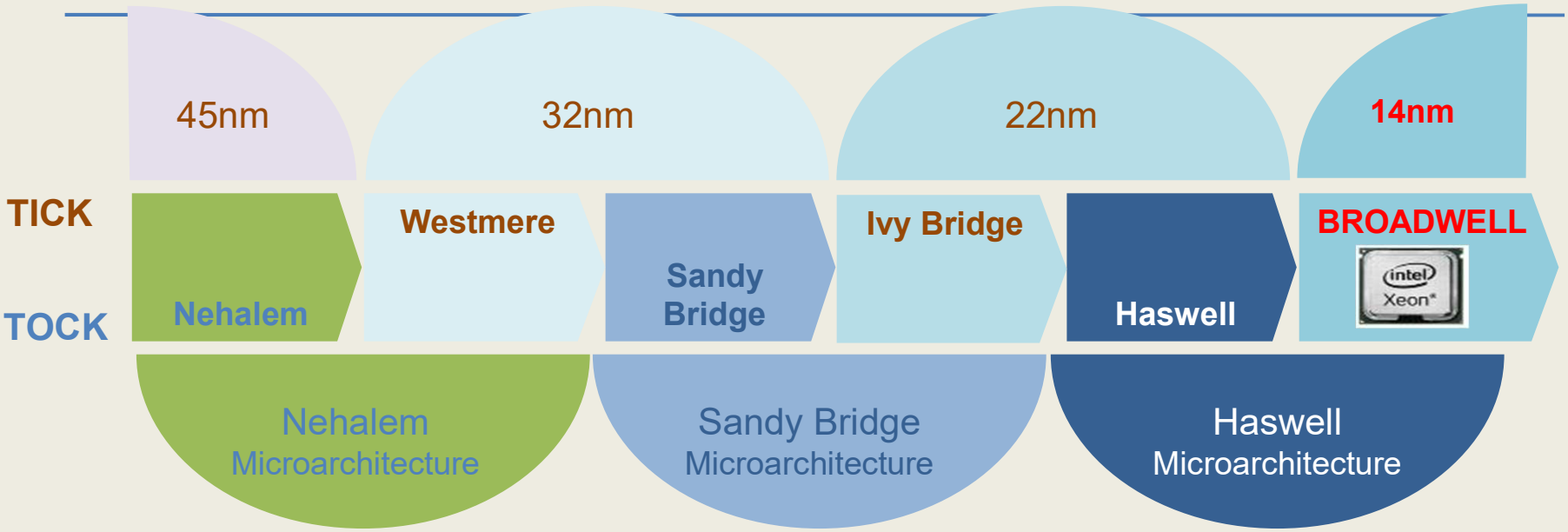
Overview of the Intel®
Xeon and Xeon Phi technologies
Broadwell and Knights Landing

Fabio Affinito (SCAI - Cineca)

Intel® Xeon Processor Architecture

Intel® Xeon® Processor E5-2600 v4 Product Family - TICK



Typically, Increases in Transistor Density Enables New Capabilities, Higher Performance Levels, and Greater Energy Efficiency

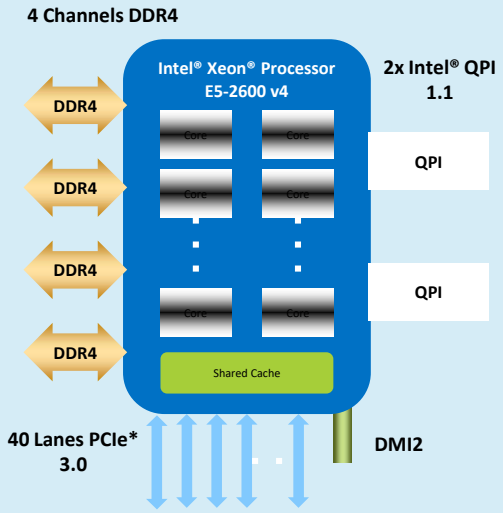
Intel® Xeon® E5-2600 v4 Product Family Overview

New Features:
- Broadwell microarchitecture
- Built on 14nm process technology
- Socket compatible◊ replacement/ upgrade on Grantley-EP platforms

New Performance Technologies:
- Optimized Intel® AVX Turbo mode
- Intel TSX instructions^

Other Enhancements:
- Virtualization speedup
- Orchestration control
- Security improvements

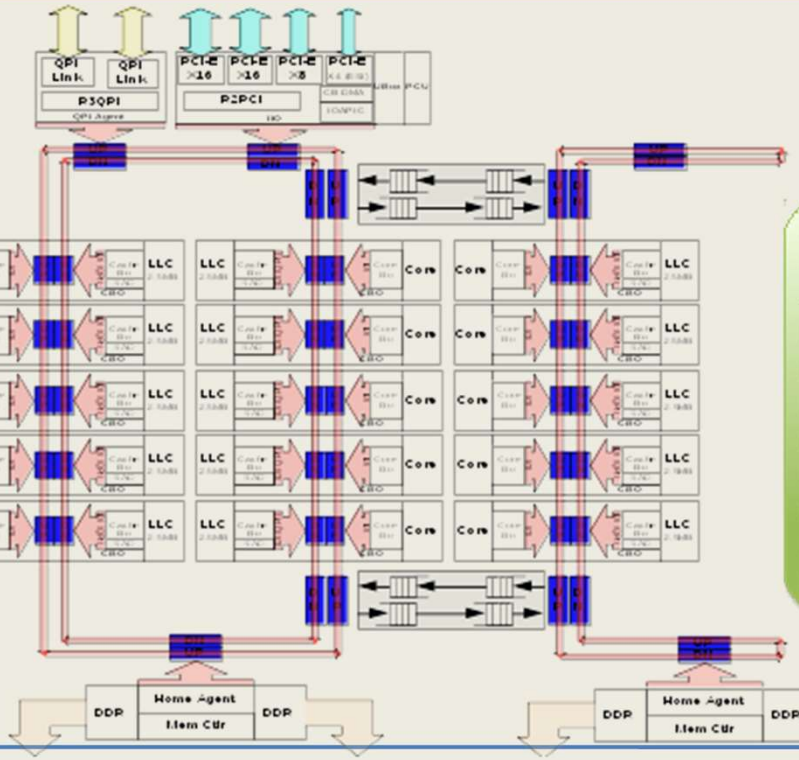
Features	Xeon E5-2600 v3 (Haswell-EP)	Xeon E5-2600 v4 (Broadwell-EP)
Cores Per Socket	Up to 18	**Up to 22**
Threads Per Socket	Up to 36 threads	**Up to 44 threads**
Last-level Cache (LLC)	Up to 45 MB	**Up to 55 MB**
QPI Speed (GT/s)	2x QPI 1.1 channels 6.4, 8.0, 9.6 GT/s	
PCIe* Lanes / Speed(GT/s)	40 / 10 / PCIe* 3.0 (2.5, 5, 8 GT/s)	
Memory Population	4 channels of up to 3 RDIMMs or 3 LRDIMMs	**+ 3DS LRDIMM†**
Memory RAS	ECC, Patrol Scrubbing, Demand Scrubbing, Sparing, Mirroring, Lockstep Mode, x4/x8 SDDC	**+ DDR4 Write CRC**
Max Memory Speed	Up to 2133	**Up to 2400**
TDP (W)	160 (Workstation only), 145, 135, 120, 105, 90, 85, 65, 55	



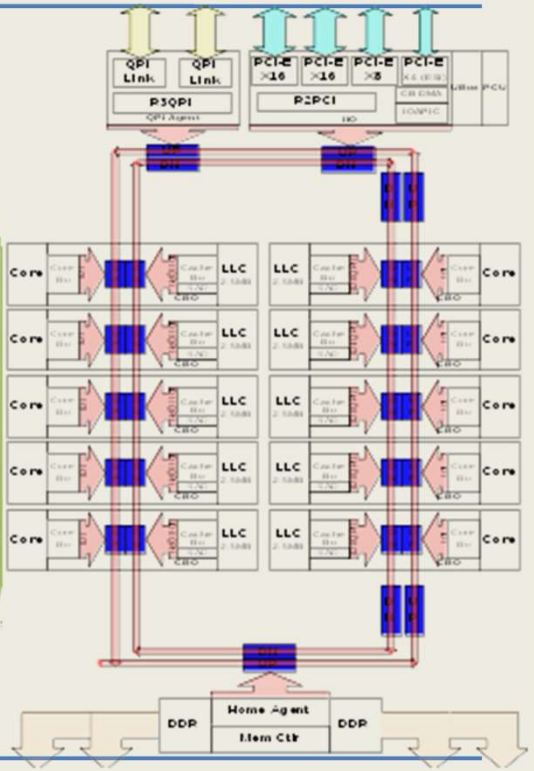
◊ Requires BIOS and firmware update; ^ not available broadly on E5-2600 v3; † Depends on market availability

Intel® Xeon® Processor E5-2600 v4 Product Family MCC/LCC

Medium Core Count (MCC) SKUs
12C to 14C
E5-2690 v4
E5-2680 v4 E5-2660 v4 E5-2650 v4 E5-2650L v4 E5-2687W v4



Low Core Count (LCC) SKUs
4C to 10C
E5-2640 v4
E5-2630 v4
E5-2620 v4
E5-2609 v4
E5-2667 v4
E5-2643 v4
E5-2637 v4
E5-2630L v4



Intel® Xeon® Processor E5-2600 v4 Product Family HCC

High core count (HCC) die configuration

- Used by SKUs with 16 to 22 cores
  - E5-2699 v4
  - E5-2698 v4
  - E5-2697 v4
  - E5-2697A v4
  - E5-2695 v4
  - E5-2683 v4
- For each core
  - 2.5M last level cache (LLC)
  - Caching agent (CBO)
- For each ring
  - Home agent (HA)
  - Memory Controller with 2 DDR4 channels

What's next …

- Broadwell (code name) E7 (4-socket server processor models)
- Skylake (code name) server (E5 and E7)
  - Micro-architecture launched in client processors Sep. 2015
  - Intel® AVX-512 (only for server)
  - Expect a lot of additional, key changes
- FPGA and Xeon server integration
- NVM (non-volatile memory) - 3D XPoint™ Technology

Intel® Many Integrated Core Architecture (Intel® MIC)
Intel® Xeon Phi™ Coprocessor

Intel® Xeon Phi™ Product Family

based on Intel® Many Integrated Core (MIC) Architecture

2013:

Intel® Xeon Phi™ Coprocessor x100 Product Family

"Knights Corner"

22 nm process

Up to 61 Cores

Up to 16GB Memory

2016:

Second Generation Intel® Xeon Phi™

"Knights Landing"

14 nm

Processor & Coprocessor

+60 cores

On Package, High-Bandwidth Memory

Future Knights:

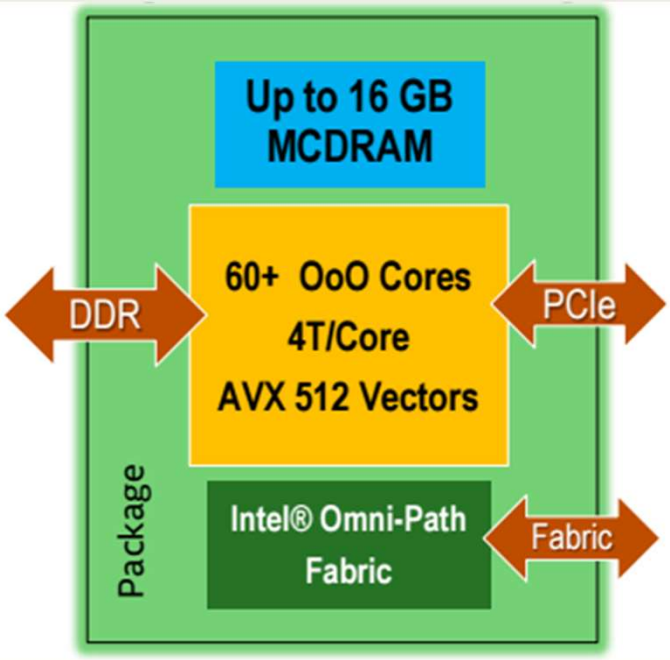
Upcoming Gen of the Intel® MIC Architecture

In planning

Continued roadmap commitment

*Per Intel's announced products or planning process for future products

Knights Landing: Next-Generation Intel® Xeon Phi™



Stand-alone, Self-boot CPU

60+ new Silvermont-based cores

4 Threads per core

AVX 512 vector units

Binary Compatible[1] with Intel® Xeon® processor

2-dimensional Mesh on-die interconnect

MCDRAM: On-Package memory: 400+ GB/s of BW[2]

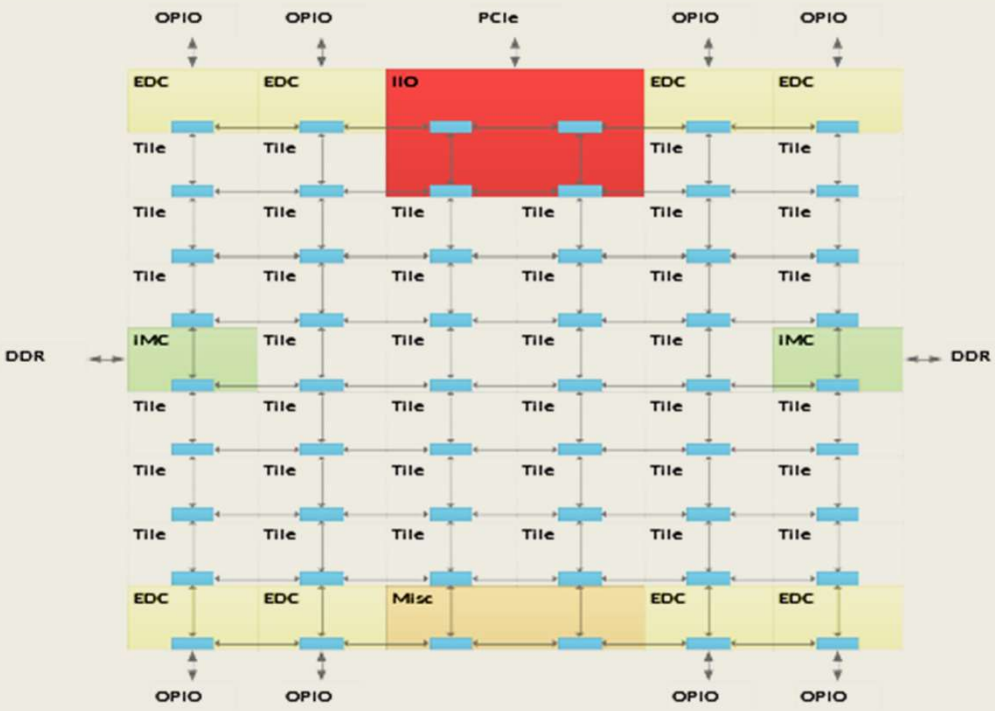
DDR memory

Intel® Omni-path Fabric

3+ TFLops (DP) peak per package

~3x ST performance over KNC

KNL Mesh Interconnect



Mesh of Rings

- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

Cache Coherent Interconnect

- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

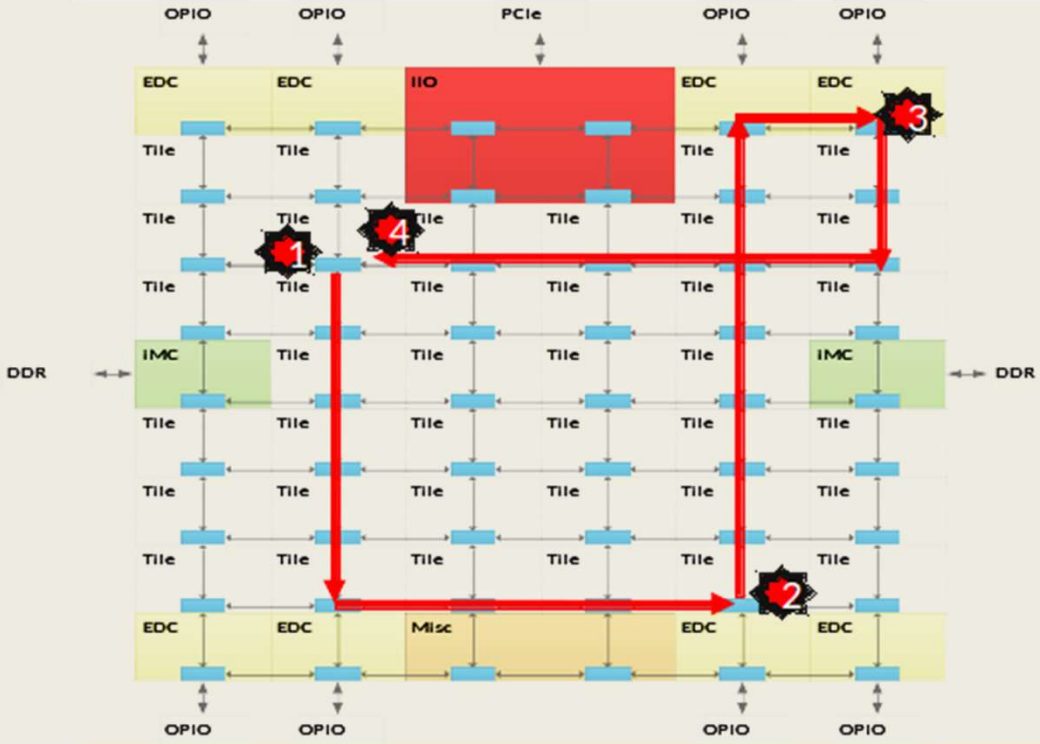
Three Cluster Modes

(1) All-to-All

(2) Quadrant

(3) Sub-NUMA Clustering (SNC)

Cluster Mode: All-to-All



Address uniformly hashed across all distributed directories

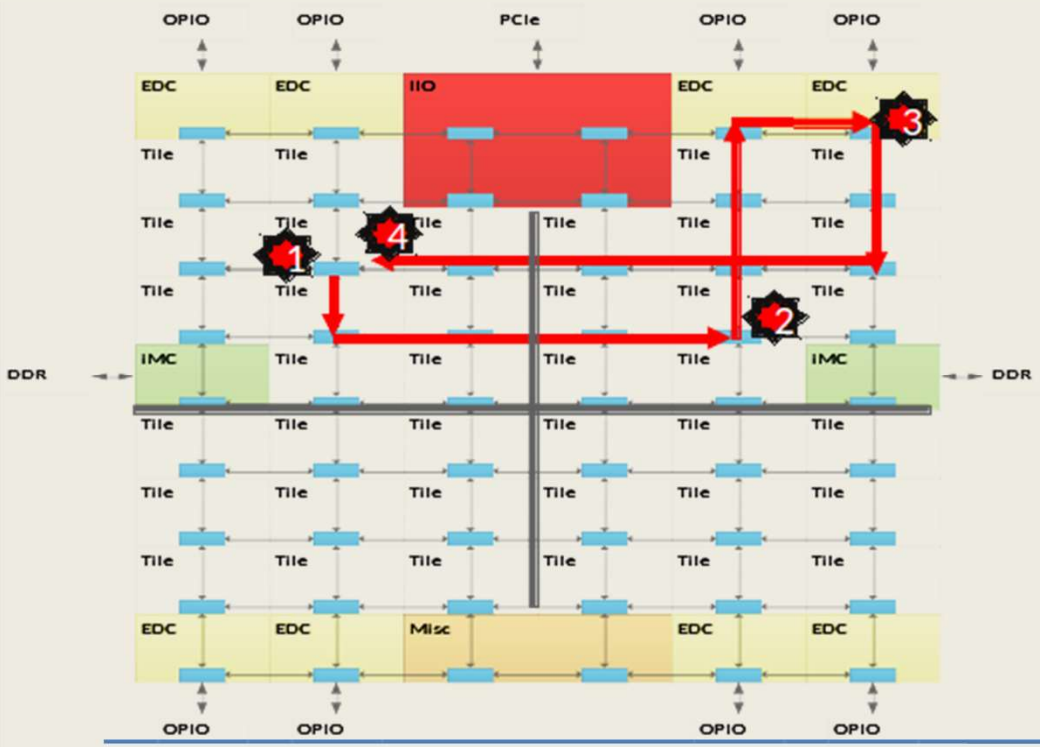
No affinity between Tile, Directory and Memory

Lower performance mode, compared to other modes. Mainly for fall-back

Typical Read L2 miss

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

Cluster Mode: Quadrant



Chip divided into four virtual Quadrants

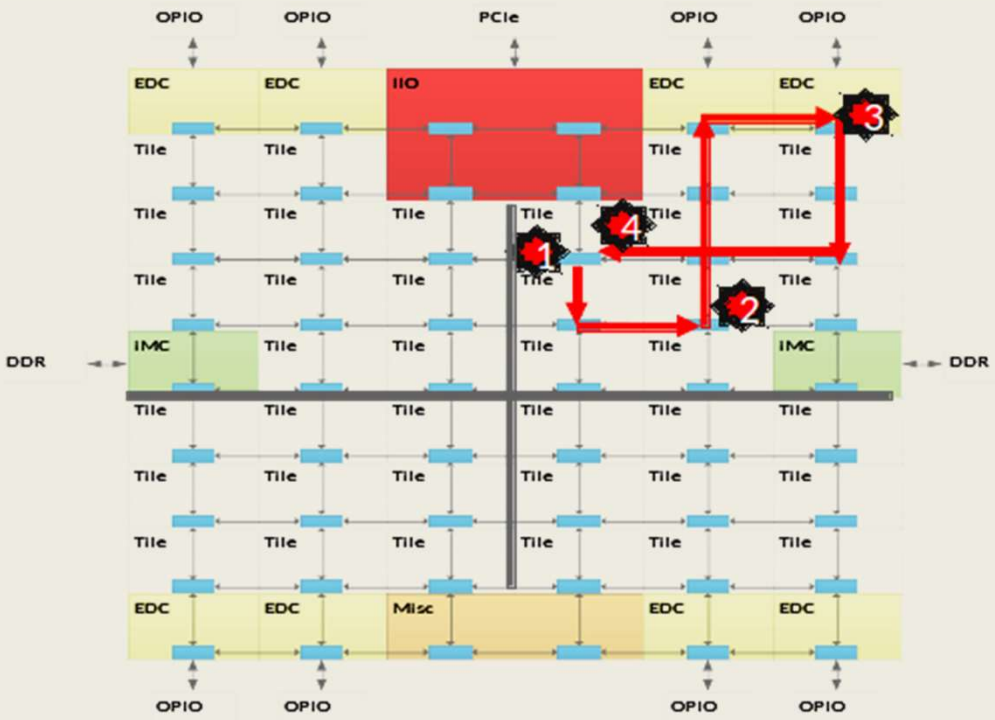
Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. Software transparent.

1. L2 miss, 2. Directory access, 3. Memory access, 4. Data return

Cluster Mode: Sub-NUMA Clustering (SNC)



Each Quadrant (Cluster) exposed as a separate NUMA domain to OS

Looks analogous to 4-Socket Xeon

Affinity between Tile, Directory and Memory

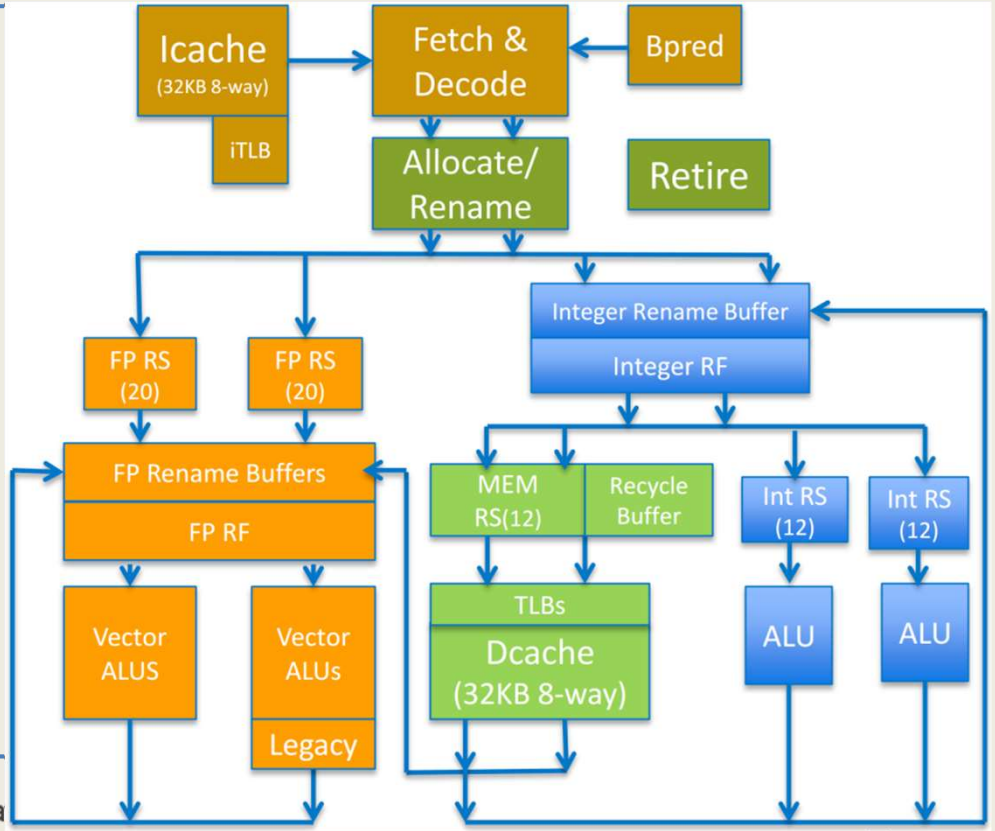
Local communication. Lowest latency of all modes

Software needs to be NUMA-aware to get benefit

1. L2 miss, 2. Directory access, 3. Memory access, 4. Data return

KNL Core and VPU

Out-of-order core w/ 4 SMT threads
VPU tightly integrated with core pipeline
2-wide decode/rename/retire
2x 64B load & 1 64B store port for D$
L1 prefetcher and L2 prefetcher
Fast unaligned and cache-line split support
Fast gather/scatter support



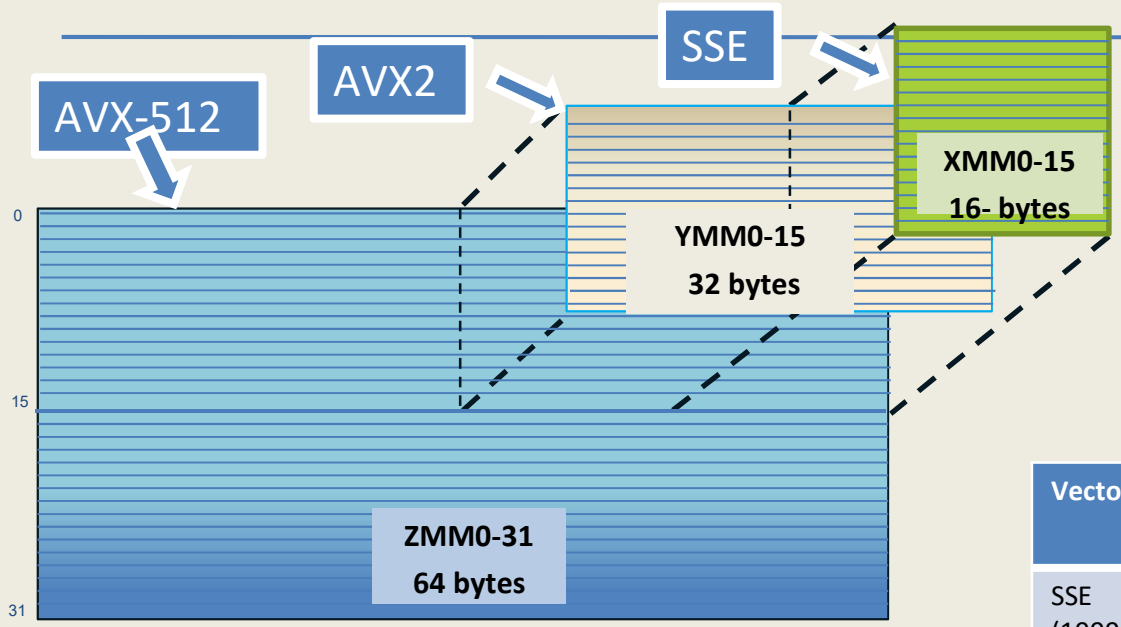
Software Adaption for KNL – Key New Features

Large impact: Intel® AVX-512 instruction set
- Slightly different from future Intel® Xeon™ architecture AVX-512 extensions
- Includes SSE, AVX, AVX-2
- Apps built for HSW and earlier can run on KNL (few exceptions like TSX)
- Incompatible with 1st Generation Intel® Xeon™ Phi (KNC)

Medium impact: New, on-chip high bandwidth memory (MCDRAM) creates heterogeneous (NUMA) memory access
- can be used transparently too however

Minor impact: Differences in floating point execution / rounding due to FMA and new HW-accelerated transcendental functions - like exp()

AVX-512 - Greatly increased Register File



Vector Registers	IA32 (32bit)	Intel64 (64bit)
SSE (1999)	8 x 128bit	16 x 128bit
AVX and AVX-2 (2011 / 2013)	8 x 256bit	16 x 256bit
AVX-512 (2014 – KNL)	8 x 512bit	32 x 512bit

The Intel® AVX-512 Subsets [1]

AVX-512 F

AVX-512 F: 512-bit Foundation instructions common between MIC and Xeon

- ❑ Comprehensive vector extension for HPC and enterprise
- ❑ All the key AVX-512 features: masking, broadcast…
- ❑ 32-bit and 64-bit integer and floating-point instructions
- ❑ Promotion of many AVX and AVX2 instructions to AVX-512
- ❑ Many new instructions added to accelerate HPC workloads

AVX-512CD

AVX-512 CD (Conflict Detection instructions)

- ❑ Allow vectorization of loops with possible address conflict
- ❑ Will show up on Xeon

AVX-512ER

AVX-512 extensions for exponential and prefetch operations

- ❑ fast (28 bit) instructions for exponential and reciprocal and transcendentals (as well as RSQRT
- ❑ New prefetch instructions: gather/scatter prefetches and PREFETCHWT1

AVX-512PR

The Intel® AVX-512 Subsets [2]

AVX-512DQ

AVX-512 Double and Quad word instructions

- ❑ All of (packed) 32bit/64 bit operations AVX-512F doesn't provide
- ❑ Close 64bit gaps like VPMULLQ : packed 64x64 ➔ 64
- ❑ Extend mask architecture to word and byte (to handle vectors)
- ❑ Packed/Scalar converts of signed/unsigned to SP/DP

AVX-512BW

AVX-512 Byte and Word instructions

- ❑ Extend packed (vector) instructions to byte and word (16 and 8 bit) datatypes
- ❑ MMX/SSE2/AVX2 re-promoted to AVX512 semantics
- ❑ Mask operations extended to 32/64 bits to adapt to number of objects in 512bit
- ❑ Permute architecture extended to words (VPERMW, VPERMI2W, …)

AVX-512VL

AVX-512 Vector Length extensions

- ❑ Vector length orthogonality
- ❑ Support for 128 and 256 bits instead of full 512 bit
- ❑ Not a new instruction set but an attribute of existing 512bit instructions

Other New Instructions

MPX

Intel® MPX – Intel Memory Protection Extension

- ❑Set of instructions to implement checking a pointer against its bounds
- ❑Pointer Checker support in HW (today a SW only solution of e.g. Intel compilers)
- ❑Debug and security features

SGX

Intel® SGX – Intel® Software Guard Extensions

- ❑ Intel® Software Guard Extensions enables applications to execute code and protect secrets fro

CLFLUSHOPT

Single instruction – Flush a cache line

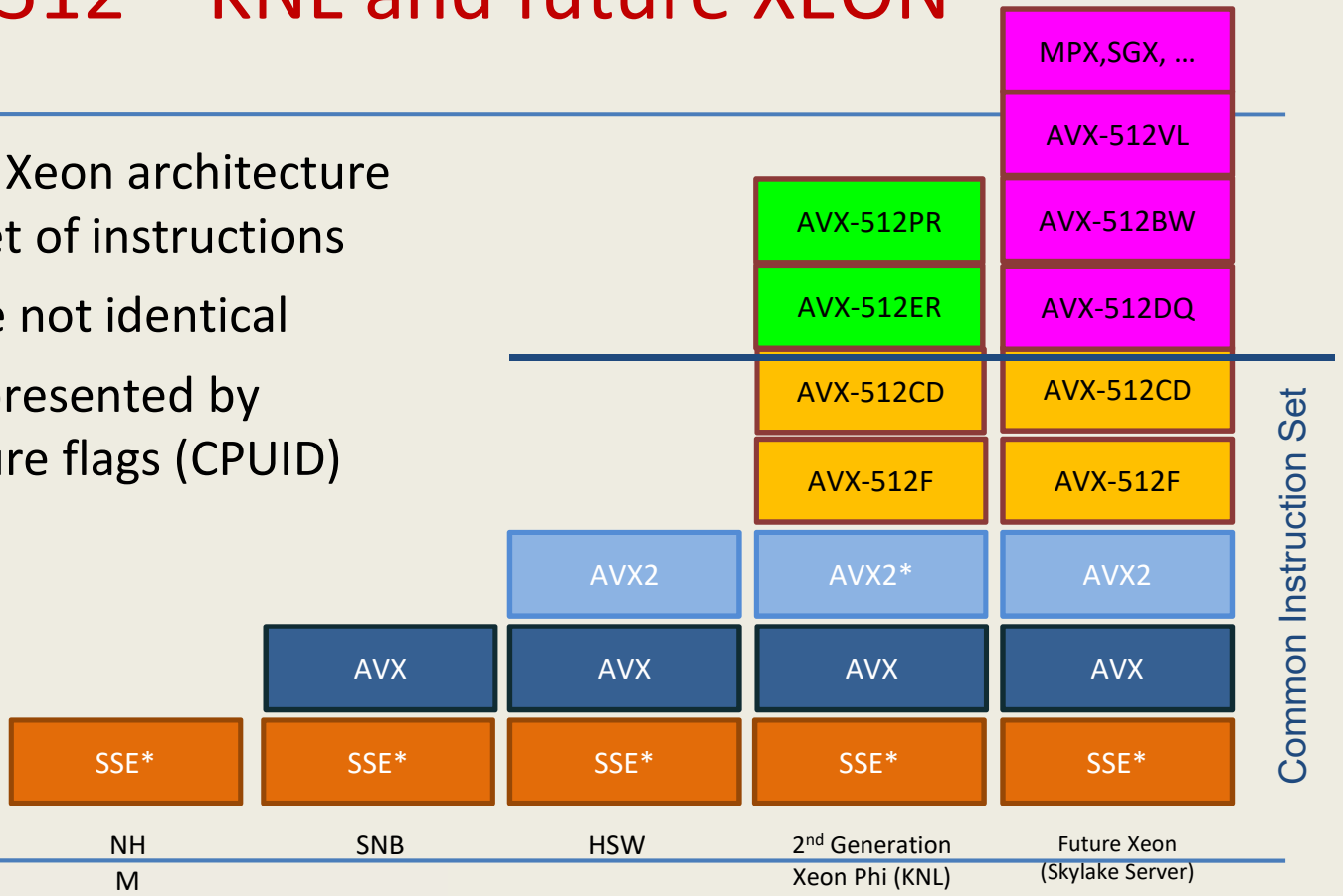
- ❑ needed for future memory technologies

XSAVE{S,C}

Save and restore extended processor state

AVX-512 – KNL and future XEON

- KNL and future Xeon architecture share a large set of instructions
  - but sets are not identical
- Subsets are represented by individual feature flags (CPUID)



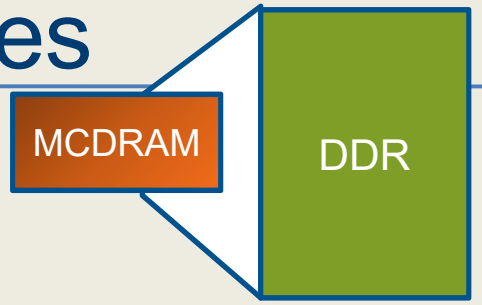
Intel® Compiler Processor Switches

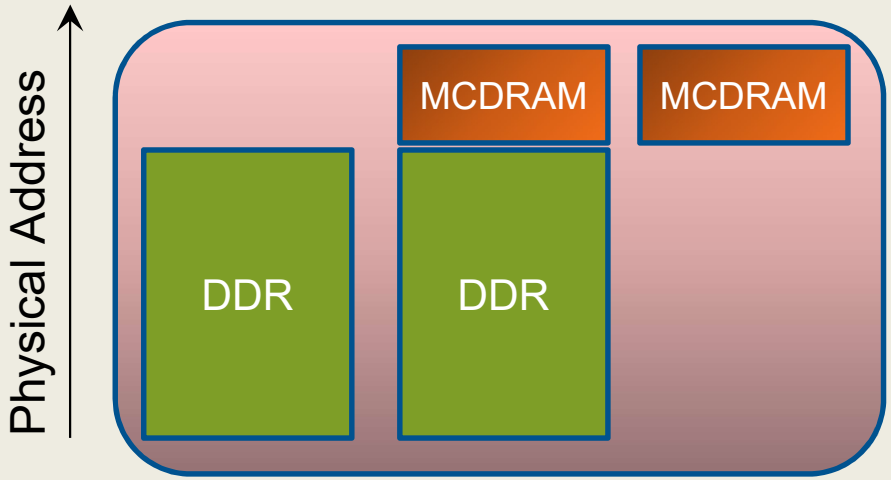
Switch	Description
`-xmic-avx512`	KNL only; already in 14.0
`-xcore-avx512`	Future XEON only, already in 15.0.1
`-xcommon-avx512`	AVX-512 subset common to both, already in 15.0.2
`-m, -march, /arch`	Not yet !
`-ax<…-avx512>`	Same as for "-x<…-avx512>"
`-mmic`	No – not for KNL

KNL Memory Modes

- Mode selected at boot
- MCDRAM-Cache covers all DDR

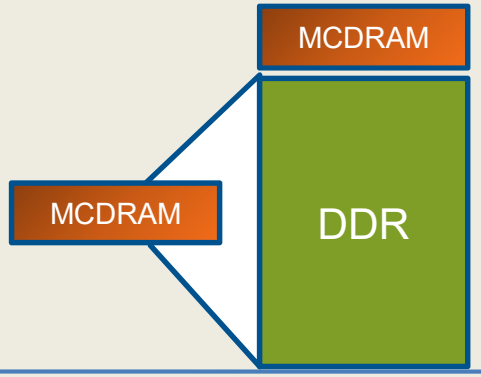
Cache Model



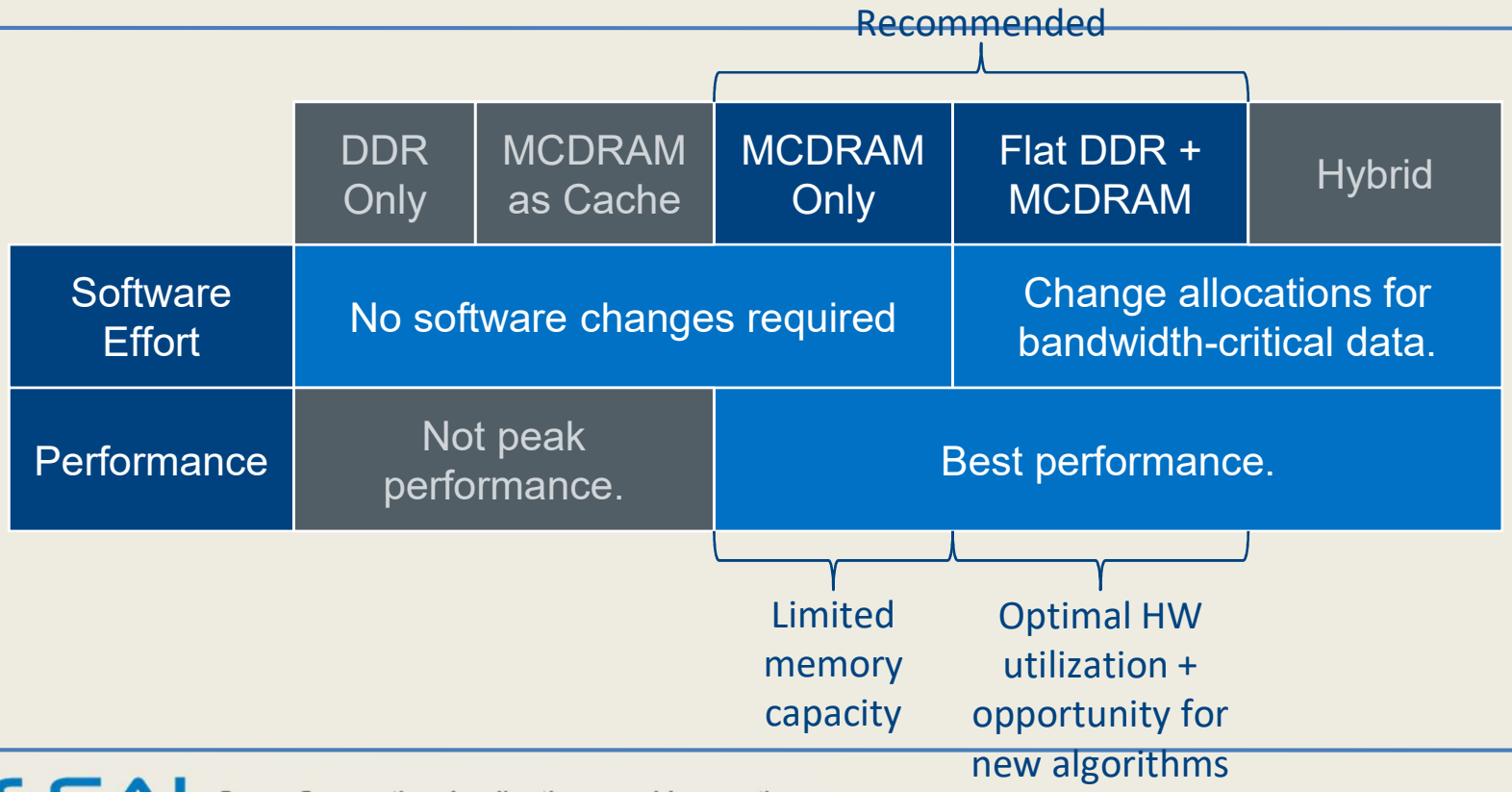


Flat Models

Hybrid Model



MCDRAM: Cache vs Flat Mode



High Bandwidth On-Chip Memory API

- API is open-sourced (BSD licenses)
  - https://github.com/memkind
  - Uses jemalloc API underneath
    - http://www.canonware.com/jemalloc/
    - https://www.facebook.com/notes/facebook-engineering/scalable-memory-allocation-using-jemalloc/480222803919

Malloc replacement:

```
#include <memkind.h>

  hbw_check_available()
  hbw_malloc, _calloc, _realloc,… (memkind_t kind, …)
  hbw_free()
  hbw_posix_memalign()
  hbw_get_size(), _psize()

ld …  -ljemalloc -lnuma -lmemkind -lpthread
```

HBW API for Fortran, C++

Fortran:

!DIR$ ATTRIBUTES FASTMEM :: data_object1, data_object2

- All Fortran data types supported
- Global, local, stack or heap; scalar, array, …
- Support in compiler 15.0 update 1 and later versions

C++:

standard allocator replacement for e.g. STL like

```
#include <hbwmalloc.h>
std::vector<int, hbwmalloc::hbw_allocator>
```

Porting codes on Knights Landing

Trends that are here to stay

Data parallelism
- Lots of threads, spent on MPI ranks or OpenMP/TBB/pthreads
- Improving support for both peak tput and modest/single thread

Bigger, better, faster memory
- High capacity, high bandwidth, low latency DRAM
- Effective caching and paging
- Increasing support for irregular memory refs, modest tuning

ISA innovation
- Increasing support for vectorization, new usages

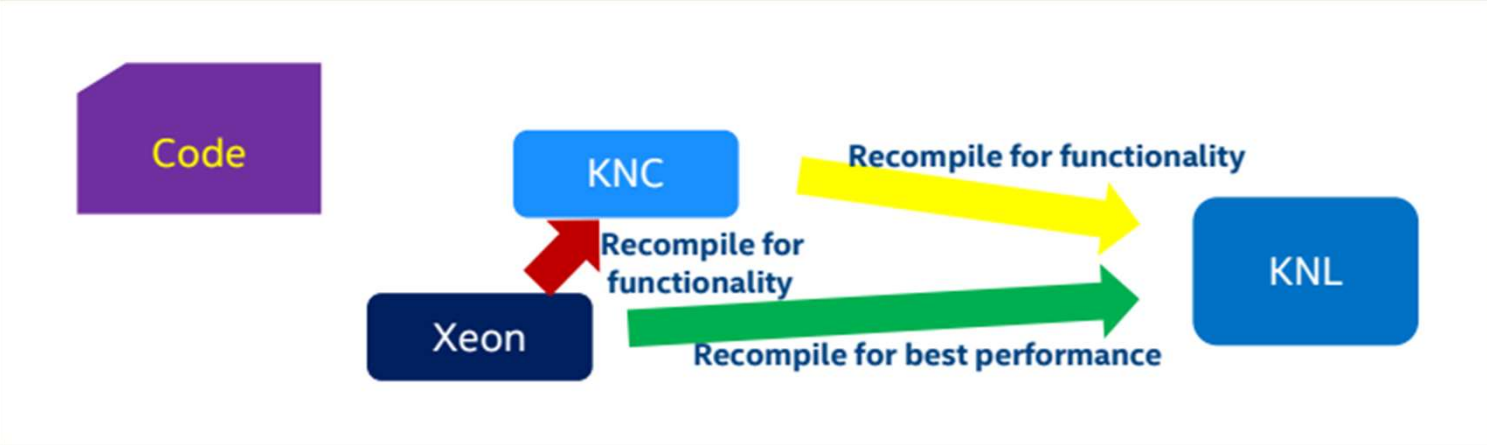
Evolution or revolution?

Incremental changes, significant gains

Parallelization – consistent strategy
- MPI vs. OpenMP – already needed to tune and tweak
- Less thread-level parallelism required
- Vectorization – more opportunity, more profitable

Enable new features with memory tuning
- Access MCDRAM with special allocation
- Blocking for MCDRAM vs. just cache

Compatibility



KNL specific enabling

- Recompilation, with –xMIC-AVX512
- Threading: more MPI ranks, 1 thread/core
- Vectorization: increased efficiency
- MCDRAM and memory tuning: tile, 1GB pages

What is needed?

- Building

Change compiler switches in make files

- Coding

Parallelization: vectorization, offload

Memory management: MCDRAM enumeration and memory allocation

- Tuning

Potentially fewer threads: more cores but less need for SMT

More memory more MPI ranks

Take aways

Keep doing what you were doing for KNC and Xeon

Some goodness comes for free with a recompile

With some extra enabling, use new MCDRAM feature

Acknowledgements

Most of the material (slides, figures, etc) showed here is courtesy of Intel

In particular, thanks for providing material and support to:
Georg Zitzlsberger, Heinrich Bockhorst, Han Benedict and CJ Newburn