An overview of Intel MIC - technology, hardware and software

"ATLAS Future Software Technologies Forum"

August 22nd 2012

Andrzej Nowak, CERN openlab

Based on the work of Sverre Jarp, Alfio Lazzaro, Julien Leduc, Andrzej Nowak

CERN openlab

DISCLAIMER

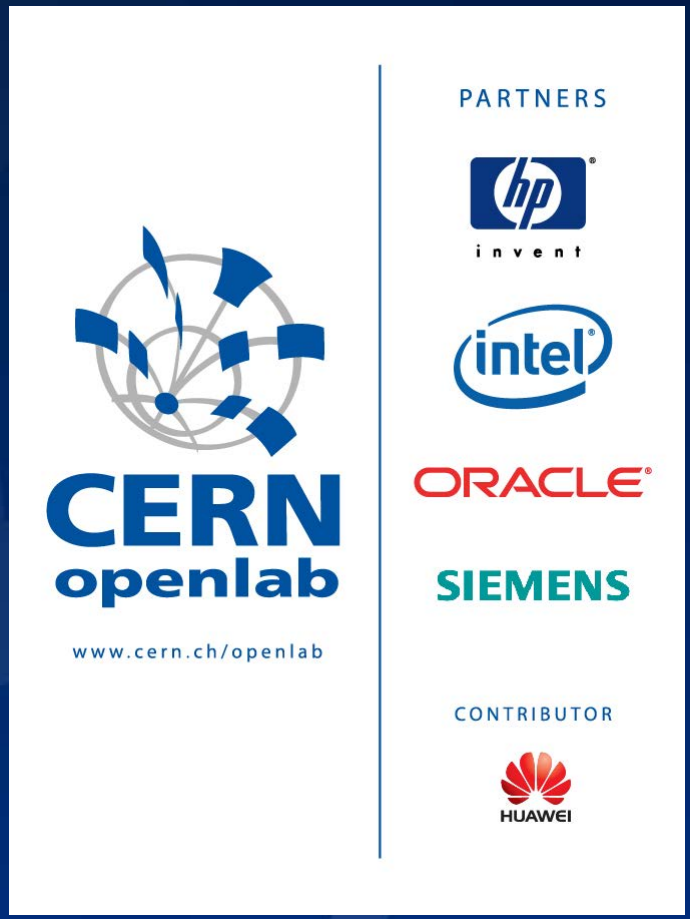
This presentation is non-NDA and may contain speculative statements. No guarantees are given.

Overview

1. Brief on openlab and its activities
2. The Intel MIC program – an overview
3. MIC hardware
4. Related software
5. Future directions

CERN openlab

- CERN openlab is a framework for evaluating and integrating cutting-edge IT technologies or services in partnership with industry
- The Platform Competence Center (PCC) has worked closely with Intel for the past decade and focuses on:
  - many-core scalability
  - performance tuning and optimization
  - benchmarking and thermal optimization
  - teaching



openlab hardware

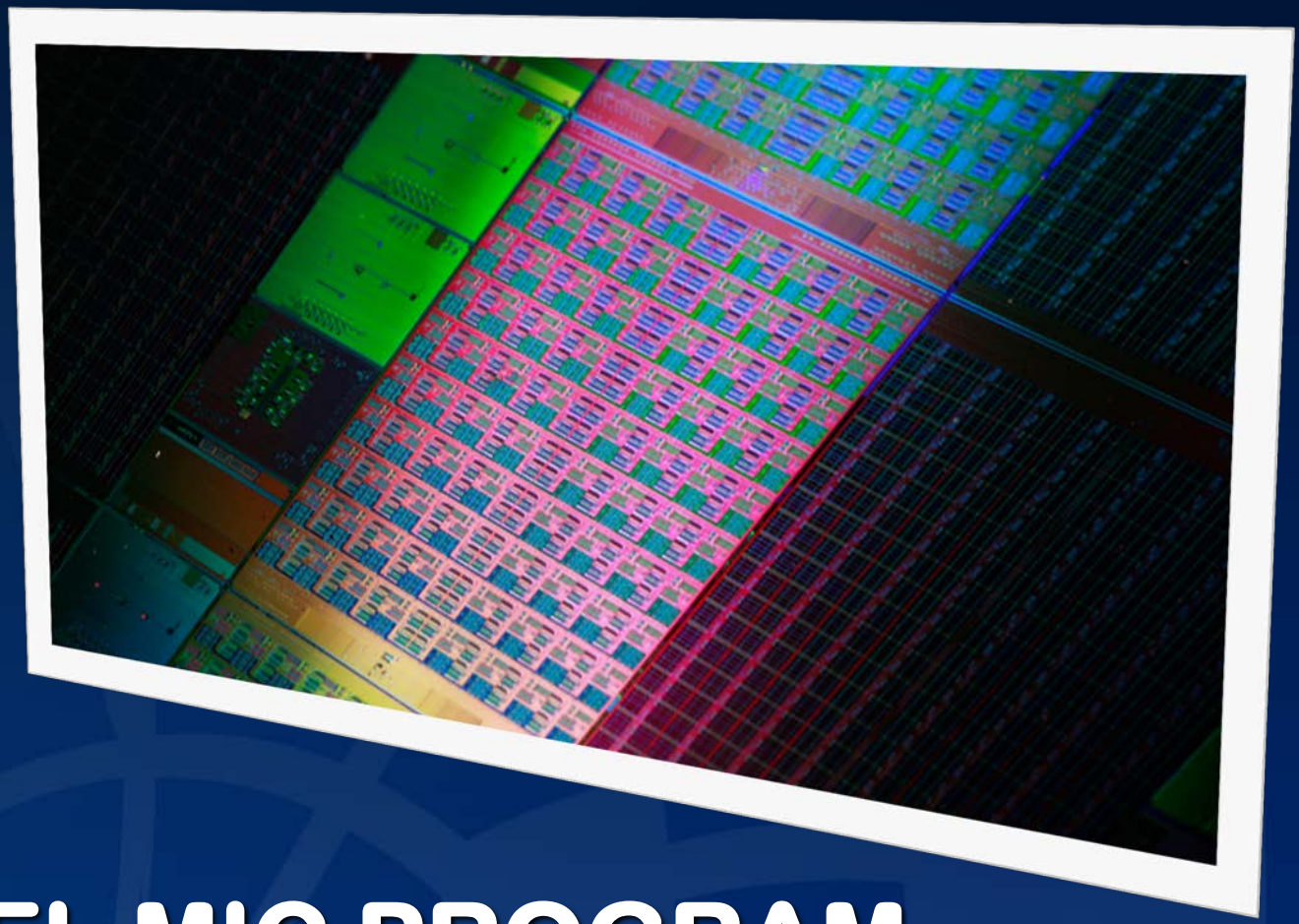
- **"Rolling" group of recent dual socket servers**
  - Nehalem-EP (8 cores)
  - Westmere-EP (12 cores)
  - In delivery: Sandy Bridge-EP (12 cores)
- **A group of several dozen "unique" test machines**
  - Alpha and Beta Intel systems
  - Various SKUs: high performance, standard, low voltage
  - Desktops, Nettops
  - Blades, Microservers
  - Quad-socket systems (Intel -EX family and AMD)
  - Accelerators (KNF, KNC and GPUs)
- **CPU families covered (and corresponding platforms):**
  - Itanium, Pentium 4, Woodcrest, Penryn, Nehalem, Westmere, Sandy Bridge, Ivy Bridge, Atom, ARM, AMD Magny-Cours

Scalability testing

- Set of major “standard” benchmarks covering various corners of HEP

Benchmark	Threaded	Vectorized
HEPSPEC06		
ROOT (MLFit)	☑	☑
Geant4 (FullCMS)	☑	
ALICE HLT	☑	☑

- GCC and the Intel compiler used



1: INTEL MIC PROGRAM

The Intel MIC program

- **Project "Larrabee" was an x86 processor with wide vectors destined for graphics**
- **Adapted into a "throughput computing" solution– the "Knights" family**
  - select Intel collaborators working in the program – CERN openlab amongst them
  - software opened up and supported for general purpose compute
  - meant strictly as a software development vehicle
- **Successor(s) foreseen**

How much is behind it?

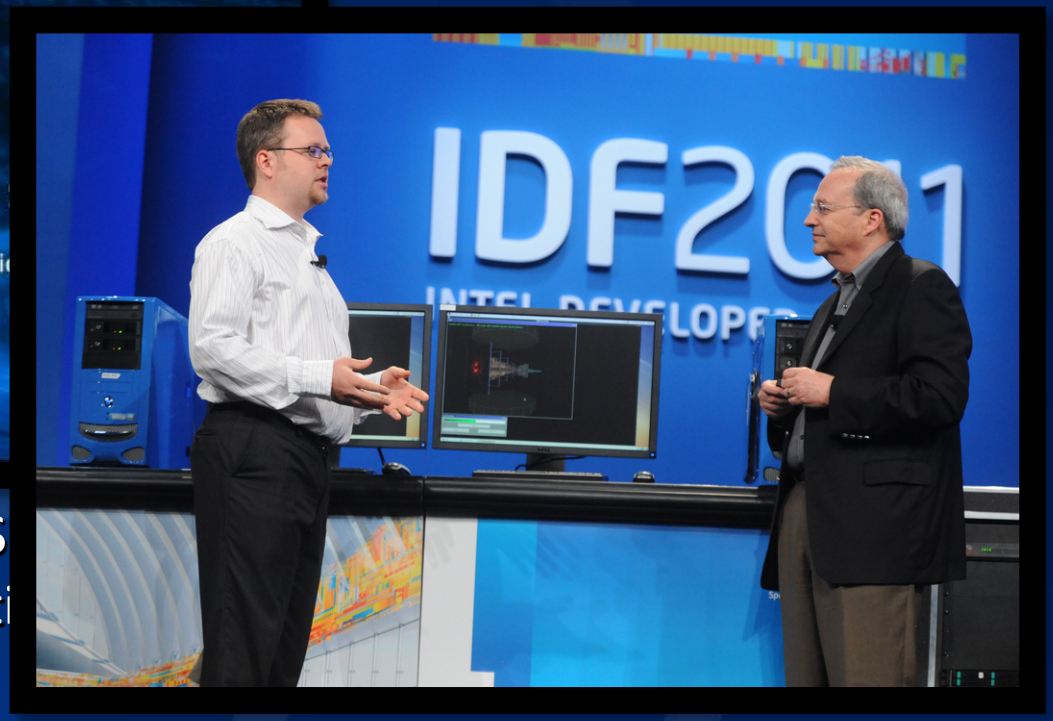
- **Intel is serious - more than a research project**
  - past: POLARIS (80 core research chip)
  - past: SCC (48 core research chip)
- **Texas Advanced Computing Center (TACC) to use Knights processors in its 10 PFLOP machine in 2013**
  - 8 out of 10 PFLOP in “Stampede” to be provided by Knights
  - Expansion to 15 PFLOP foreseen with Knights Landing
- **SGI to build HPC products based on Knights**
- **Dozens of Intel collaborators ported their workloads**
  - Many of which claim their workload will not run on GPUs because of cost and complexity


TOP500®
SUPERCOMPUTER SITES
PROJECT
LISTS
STATISTICS
RESOURCES
NEWS
Discovery - Intel Cluster, Xeon E5-2670 8C 2.600GHz, Infiniband FDR, Intel MIC
Site: Intel
System URL:
Manufacturer: Intel
Cores: 9800
Power: 100.80 kW
Memory: 8960 GB
Interconnect: Infiniband FDR
Operating System: Linux

Intel MIC and openlab

openlab contributions to the MIC program

- Ported and optimized 3 large representative benchmarks
- Feedback on the ISA



- ISC'10 w/ Intel VP Kirk S
- IDF'11 w/ Intel CTO Justi

2: HARDWARE

MIC hardware – Knights Ferry

- **PCIe card form factor, a "PC in a PC"**
  - Linux OS on board
  - PCIe power envelope - ~300W
  - Limited on-board memory (up to 2GB)
  - 32 cores @ 1.2 GHz
- **x86 architecture**
  - 64-bit
  - P54C core: In-order, Superscalar
  - 8MB shared coherent cache
  - New ISA with new vector instructions
- **Floating point support through vector units**
  - up to 1 TFLOP SP
  - 512-bit wide
- **4-wide hardware threading**
  - 128 threads in total

MIC hardware – Knights Corner

- Possible frequency bump
- More on-board memory
- >50 cores
- 22nm process
- Same x86 architecture as KNF, but with improvements
- 1 TFLOP DP
  - Still with the programmability of a Xeon

KNF hardware architecture

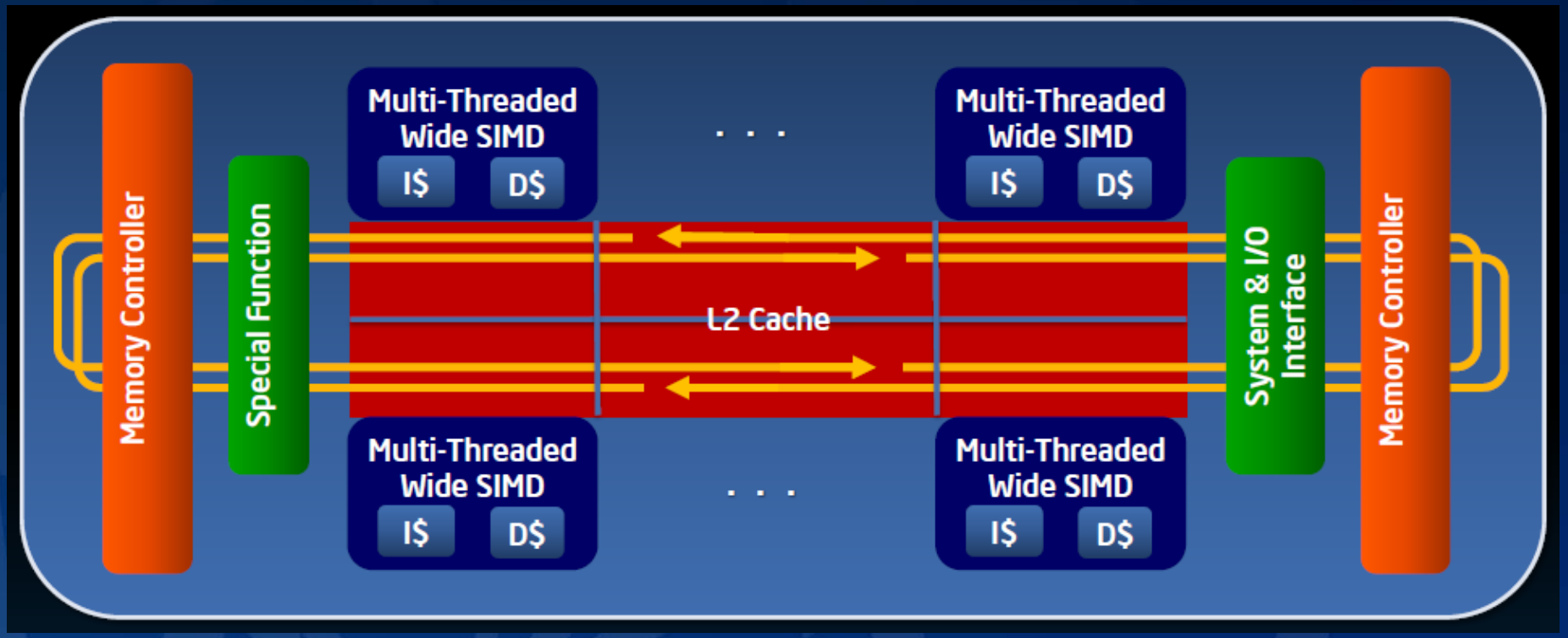
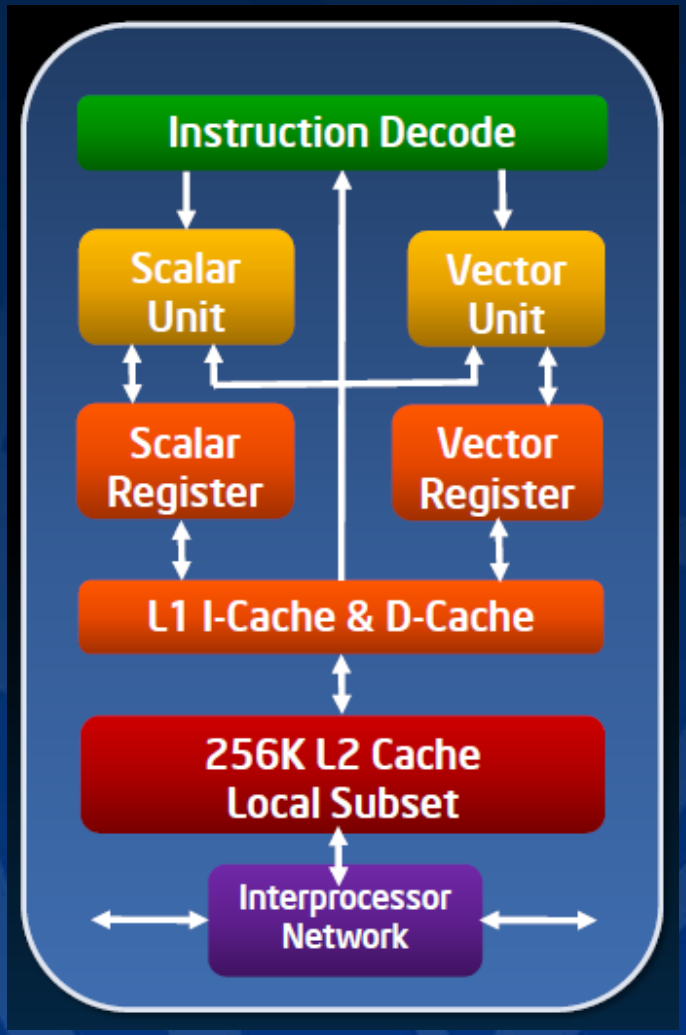


Image: Intel

Aubrey Isle core architecture

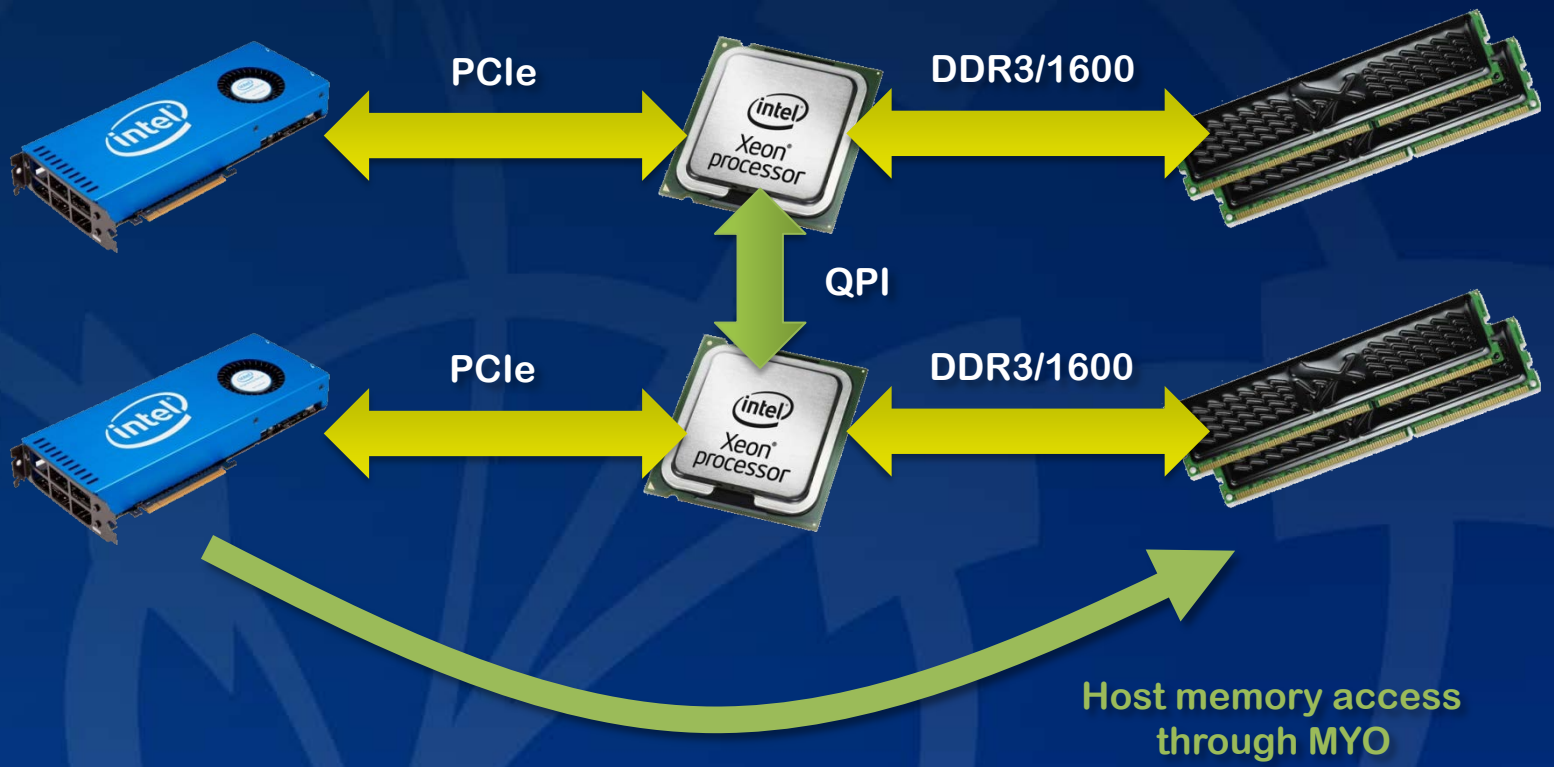


- Superscalar
- 32k L1 I/D cache
- 256k Coherent L2 caches
- 2x512-bit ring bus inter-CPU network
- 4-wide SIMD with separate register sets

Image: Intel

Possible MYO-like scenario

- MYO stands for “Mine Yours Ours”



Aubrey Isle VPU

- 512-bit 1-cycle vectors
- 32 registers + masking registers
- Data types:
  - 8x DOUBLE64
  - 16x FLOAT32
  - 16x INT32
- IEEE compliant FMA support
- Broadcasts, shuffles, swizzles, gather, scatter
- Predication and masking

3: SOFTWARE

MIC software

- **Extensive focus on programmability and interoperability with the Xeon**
  - PCI becomes transparent
- **Card OS is Linux**
- **Future GNU support possible**
- **Broad software support; standard Intel software available, same as on Xeon:**
  - C/C++/Fortran Compiler, VTune Amplifier profiler, various other
- **Various standard libraries supported**
- **Special APIs under development/consideration**
  - Host memory access
  - Low level communication
- **Various programming models considered**
- **"Familiar territory" approach – Intel wants to keep this "as x86 as possible", with all the pros and cons**

Programmability (1)

Summing vector elements in C using OpenMP – openmp.org

```
#pragma omp parallel for reduction(+: s)
for (int i = 0; i < n; i++) {
  s += x[i];
}
```

Per element multiply in C++ using Intel® Array Building Blocks – intel.com/go/arbb

```
void products( const arbb::dense<arbb::f32>& a,
               const arbb::dense<arbb::f32>& b,
               arbb::dense<arbb::f32>& c) {
  c = a * b;
}
```

Dot product in Fortran using OpenMP – openmp.org

```
!$omp parallel do reduction ( + : adotb )
  do j = 1, n
    adotb = adotb + a(j) * b(j)
  end do
!$omp end parallel do
```

Sum in Fortran, using co-array feature – intel.com/software/products

```
REAL SUM[*]
CALL SYNC_ALL( WAIT=1 )
DO IMG= 2,NUM_IMAGES()
  IF (IMG==THIS_IMAGE()) THEN
    SUM = SUM + SUM[IMG-1]
  ENDIF
  CALL SYNC_ALL( WAIT=IMG )
ENDDO
```

Parallel function invocation in C using Intel® Cilk™ Plus – cilk.org

```
cilk_for (int i=0; i<n; ++i) {
  Foo(a[i]);
}
```

Parallel function invocation in C++ using Intel® Threading Building Blocks – threadingbuildingblocks.org

```
parallel_for (0, n,
  [=](int i) { Foo(a[i]); }
);
```

Programmability (2)

MPI code in C for clusters – intel.com/go/mpi

```
for (d=1; d<ntasks; d++) {
  rows = (d <= extra) ? avrow+1 : avrow;
  printf(" sending %d rows to task %d\n", rows, dest);
  MPI_Send(&offset, 1, MPI_INT, d, mtype, MPI_COMM_WORLD);
  MPI_Send(&rows, 1, MPI_INT, d, mtype, MPI_COMM_WORLD);
  MPI_Send(&a[offset][0], rows*NCA, MPI_DOUBLE, d, mtype, MPI_COMM_WORLD);
  MPI_Send(&b, NCA*NCB, MPI_DOUBLE, d, mtype, MPI_COMM_WORLD);
  offset = offset + rows;
}
```

Per element multiply in C using OpenCL – intel.com/go/opencl

```
kernel void
  dotprod( global const float *a,
           global const float *b,
           global float *c) {
    int myid = get_global_id(0);
    c[myid] = a[myid] * b[myid];
}
```

Matrix Multiply in Fortran using Intel® Math Kernel Library – intel.com/software/products

```
      call DGEMM(transa,transb,m,n,k,alpha,a,lda,b,ldb,beta,c,ldc)
```

MIC software - scenarios

MIC – porting and writing software

- Ideal situation: just add a compiler switch and recompile
- Less-than-ideal: minor adaptations, including GCC/ICC differences if any + above step
- More likely: write parallel code or parallelize existing code + above steps
- Numerous libraries available: OpenMP, MPI, TBB, Cilk, MKL etc
- Vectorization (data parallelism) is key to achieve full performance

MIC vs. GPU – SW point of view

	GPU	MIC / KNC
Architecture	Proprietary	x86
Language	Limited to vendor API Derived	Standard C, C++, Fortran
Programmability	Limited to vendor API	Mainstream techs
Xeon optimization payoff	No	Yes
Direct access	No	Yes
IEEE 754 compliance	Evolving	Standard
SSE compatibility	No	Yes
Binary FP compat. w/ Xeon	No	?

MIC vs. GPU – HW point of view

	GPU	MIC / KNC
Architecture	Proprietary	x86
Attachment	PCI	PCI
Host memory access	Possible	Possible
SIMD elements	>Hundreds	>800
Control threads	Limited	>200
Power envelope	PCI	PCI
Memory	GDDR5, ECC possible	GDDR5, ECC via interposer
Performance	Known	To be established
Price	Acceptable	To be established

WRAP-UP: Possible future trends

- **Growing # cores**
  - Xeon and other: max # cores still grows geometrically
  - Xeon-EP: typical # cores grows arithmetically
- **Cache coherency will be hard to maintain**
- **Silicon process continues to scale for some time, but what about dark silicon?**
- **Departure from a homogeneous x86 core**
- **If the above is true, what about ISA convergence?**
- **Energy efficiency still king**

WRAP-UP: How could ATLAS and HEP benefit?

- **Pros:**
  - Ultimate programmability
  - Porting is a breeze (if porting at all)
  - Wide vectors, many threads
  - Cheap communication
  - Promise of good performance, especially for HPC-like workloads
  - Multiple cards in a system supported
  - Xeon optimization payoff
- **Cons:**
  - Host attachment currently only PCIe
  - Limited amount of on board memory
  - Latency guarantees unknown
  - **Real performance (incl. floating point) still unknown**

THANK YOU

Questions? Andrzej.Nowak@cern.ch