Agenda

● Intro @ accelerators on HPC
● Architecture overview of the Intel Xeon Phi Products
● Programming models
● Native mode programming & hands on
Why do we Need **Coprocessors/or “Accelerators”** on HPC?

- In the past, computers got faster by increasing the **clock frequency** of the core, but this has now reached its limit mainly due to **power requirements** and heat dissipation restrictions (unmanageable problem).

- Today, processor cores are not getting any faster, but instead the **number of cores per chip** increases and registers are getting wider.

- On HPC, we need a chip that can provide higher computing performance at lower energy.
Why do we Need Coprocessors/or “Accelerators” on HPC?

- One solution is a heterogeneous system containing both CPUs and “accelerators”, plus other forms of parallelism such as vector instruction support.

- Two types of hardware acceleration options, Intel Xeon Phi and Nvidia GPU.
- Designed to accelerate a computation through massive parallelisation, and to deliver the highest performance and energy efficient computing in HPC.
<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>System</th>
<th>Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Rpeak (TFlop/s)</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputing Center in Wuxi</td>
<td>Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC</td>
<td>10,649,600</td>
<td>93,014.6</td>
<td>125,435.9</td>
<td>15,371</td>
</tr>
<tr>
<td>2</td>
<td>National Super Computer Center in Guangzhou</td>
<td>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.20GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT</td>
<td>3,120,000</td>
<td>33,862.7</td>
<td>54,902.4</td>
<td>17,808</td>
</tr>
<tr>
<td>3</td>
<td>DOE/SC/Oak Ridge</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.20GHz, Cray Gemini interconnect, NVIDIA K20x</td>
<td>560,640</td>
<td>17,590.0</td>
<td>27,112.5</td>
<td>8,209</td>
</tr>
<tr>
<td>4</td>
<td>DOE/NNSA/LLNL</td>
<td>Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM</td>
<td>1,572,864</td>
<td>17,173.2</td>
<td>20,132.7</td>
<td>7,890</td>
</tr>
<tr>
<td>5</td>
<td>DOE/SC/LBNL/NERSC</td>
<td>Cori - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries Interconnect</td>
<td>622,336</td>
<td>14,014.7</td>
<td>27,880.7</td>
<td>3,939</td>
</tr>
<tr>
<td>6</td>
<td>Joint Center for Advanced High Performance Computing</td>
<td>Oakforest-PACS - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 48C 1.4GHz, Intel Omni-Path</td>
<td>556,104</td>
<td>13,554.6</td>
<td>24,913.5</td>
<td>2,719</td>
</tr>
<tr>
<td>7</td>
<td>RIKEN Advanced Institute for Computational Science (AICS)</td>
<td>K computer, SPARC64 VIIIxf 2.0GHz, Tofu interconnected</td>
<td>705,024</td>
<td>10,510.0</td>
<td>11,280.4</td>
<td>12,660</td>
</tr>
<tr>
<td>8</td>
<td>Swiss National Supercomputing Centre (CSCS) Switzerland</td>
<td>Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100 Cray Inc.</td>
<td>206,720</td>
<td>9,779.0</td>
<td>15,988.0</td>
<td>1,312</td>
</tr>
</tbody>
</table>
Accelerators in the Top 500 List
**Green500 List for November 2016**

Listed below are the November 2016 Green500’s energy-efficient supercomputers ranked from 1 to 10.

<table>
<thead>
<tr>
<th>Rank</th>
<th>MFLOPS/W</th>
<th>Site</th>
<th>System</th>
<th>Total Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9462.1</td>
<td>NVIDIA Corporation</td>
<td>NVIDIA DGX-1, Xeon E5-2698v4 20C 2.2GHz, Infiniband EDR, NVIDIA Tesla P100</td>
<td>349.5</td>
</tr>
<tr>
<td>2</td>
<td>7453.5</td>
<td>Swiss National Supercomputing Centre [CSCS]</td>
<td>Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100</td>
<td>1312</td>
</tr>
<tr>
<td>3</td>
<td>6673.8</td>
<td>Advanced Center for Computing and Communication, RIKEN</td>
<td>ZettaScaler-1.6, Xeon E5-2618Lv3 8C 2.3GHz, Infiniband FDR, PEZY-SCnP</td>
<td>150.0</td>
</tr>
<tr>
<td>4</td>
<td>6051.3</td>
<td>National Supercomputing Center in Wuxi</td>
<td>Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway</td>
<td>15371</td>
</tr>
<tr>
<td>5</td>
<td>5806.3</td>
<td>Fujitsu Technology Solutions GmbH</td>
<td>PRIMERGY CX1640 M1, Intel Xeon Phi 7210 64C 1.3GHz, Intel Omni-Path</td>
<td>77</td>
</tr>
<tr>
<td>6</td>
<td>4995.7</td>
<td>Joint Center for Advanced High Performance Computing</td>
<td>PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path</td>
<td>2718.7</td>
</tr>
<tr>
<td>7</td>
<td>4688.9</td>
<td>DOE/SC/Argonne National Laboratory</td>
<td>Cray XC40, Intel Xeon Phi 7230 64C 1.3GHz, Aries interconnect</td>
<td>1087</td>
</tr>
<tr>
<td>8</td>
<td>4112.1</td>
<td>Stanford Research Computing Center</td>
<td>Cray CS-Storm, Intel Xeon E5-2680v2 10C 2.8GHz, Infiniband FDR, Nvidia K80</td>
<td>190</td>
</tr>
<tr>
<td>9</td>
<td>4086.8</td>
<td>Academic Center for Computing and Media Studies [ACCMS], Kyoto University</td>
<td>Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect</td>
<td>748.1</td>
</tr>
<tr>
<td>10</td>
<td>3836.6</td>
<td>Thomas Jefferson National Accelerator Facility</td>
<td>KOI Cluster, Intel Xeon Phi 7230 64C 1.3GHz, Intel Omni-Path</td>
<td>111</td>
</tr>
</tbody>
</table>

All systems in the top 10 are accelerator-based (5 using Xeon Phi)
Architectures Comparison (CPU vs GPU)

- Large cache and sophisticated flow control minimise latency for arbitrary memory access for serial process.

- Simple flow control and limited cache.
- More transistors for computing in parallel (up to 17 billion on Pascal GPU)
- (SIMD)

<table>
<thead>
<tr>
<th></th>
<th>Intel Xeon CPU E5-2697v4 “Brodwell”</th>
<th>Nvidia GPU P100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores @ Clock</td>
<td>2 x 18 cores @ ≥ 2.3 GHz</td>
<td>56SMs @ 1.3 GHz</td>
</tr>
<tr>
<td>SP Perf./core</td>
<td>≥ 73.6 GFlop/s</td>
<td>up to 166 GFlop/s</td>
</tr>
<tr>
<td>SP peak</td>
<td>≥ 2.6 TFlop/s</td>
<td>up to 9.3 TFlop/s</td>
</tr>
<tr>
<td>Transistors/TDP</td>
<td>2x7 Billion /2x145W</td>
<td>14 Billion/300W</td>
</tr>
<tr>
<td>BW</td>
<td>2 x 62.5 GB/s</td>
<td>510 GB/s</td>
</tr>
</tbody>
</table>

7-8 February 2017
Intel Xeon Phi Products

• The first product was released in 2012 named Knights Corner (KNC) which is the first architecture supporting 512 bit vectors.

• The 2\textsuperscript{nd} generation released at ISC16 in Frankfurt named Knights Landing (KNL) also support 512bit vectors with a new instruction set called Intel Advanced Vector Instructions 512 (Intel AVX-512)

• KNL has a peak performance of 6 TFLOP/s in single precision ~ 3 times what KNC had, due to 2 vector processing units (VPUs) per core, doubled compared to the KNC.
Intel MIC Architecture in common with Intel multi-core Xeon CPUs!

- X86 architecture
- C, C++ and Fortran
- Standard parallelisation libraries
- Similar optimisation methods
- Up to 22 cores/socket
- Up to 3 GHz
- Up to 1.54 TB RAM
- 256 bit AVX vectors
- 2-way hyper-threading

- PCIe bus connection
- IP-addressable
- Own Linux version OS
- Full Intel software tool suite
- 8 - 16 GB GDDR5 DRAM (cached)
- Up to 61 x86 (64 bit) cores
- Up to 1 GHz
- 512 bit wide vector registers
- 4-way hyper-threading
- 64-bit addressing
- SSE, AVX or AVX2: are not supported
- Intel Initial Many Core Instructions (IMCI).
Architectures Comparison

CPU
- General-purpose architecture

MIC
- Power-efficient Multiprocessor X86 design architecture

GPU
- Massively data parallel
Knights Corner vs Knights Landing

- **Knights Corner**
  - Co-processor
  - Binary incompatible with other architecture
  - 1.1 GHz processor
  - 8 GB RAM
  - 22 nm process
  - One 512-bit VPU
  - No support for branch prediction and fast unaligned memory access

- **Knights Landing**
  - No PCIe - Self hosted
  - Binary compatible with prior Xeon architectures (no phi)
  - 1.4 GHz processor
  - Up to 400 GB RAM (with MCDRAM)
  - 14 nm process
  - Two 512-bit VPUs
  - Support for branch prediction and fast unaligned memory access
The Intel Xeon Phi (KNC) Microarchitecture

- Bidirectional ring interconnect which connects all the cores, L2 caches through a distributed global Tag Directory (TD), PCIe client logic, GDDR5 memory controllers …etc.
Network Access

- Network access possible using TCP/IP tools like ssh.
- NFS mounts on Xeon Phi Supported.

First experiences with the Intel MIC architecture at LRZ, Volker Weinberg and Momme Allalen, inSIDE Vol. 11 No.2 Autumn 2013
## Cache Structure

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache line size</td>
<td>64B</td>
</tr>
<tr>
<td>L1 size</td>
<td>32KB  data cache, 32KB instruction code</td>
</tr>
<tr>
<td>L1 latency</td>
<td>1 cycle</td>
</tr>
<tr>
<td>L2 size</td>
<td>512 KB</td>
</tr>
<tr>
<td>L2 ways</td>
<td>8</td>
</tr>
<tr>
<td>L2 latency</td>
<td>15-30 cycles</td>
</tr>
<tr>
<td>Memory → L2 prefetching</td>
<td>hardware and software</td>
</tr>
<tr>
<td>L2 → L1 prefetching</td>
<td>software only</td>
</tr>
<tr>
<td>Translation Lookside Buffer (TLB)</td>
<td>64 pages of size 4KB (256KB coverage)</td>
</tr>
<tr>
<td>Coverage options (L1, data)</td>
<td>8 pages of size 2MB (16MB coverage)</td>
</tr>
</tbody>
</table>

Intel's Jim Jeffers and James Reinders: *Intel Xeon® Phi™ Coprocessor High Performance Programming*, the first book on how to program this HPC vector chip, is available on Amazon.
Features of the Initial Many Core Instructions (IMCI) Set

512-bit wide registers
- can pack up to eight 64-bit elements (long int, double)
- up to sixteen 32-bit elements (int, float)

Arithmetic Instructions
- Addition, subtraction and multiplication
- Fused Multiply-Add instruction (FMA)
- Division and reciprocal calculation
- Exponential functions and the power function
- Logarithms (natural, base 2 and base 10)
- Square root, inverse square root, hypothenuse value and cubic root
- Trigonometric functions (sin, cos, tan, sinh, cosh, tanh, asin, acos, atan)
Lab1: Access Salomon
For the Labs: Salomon System Initialisation

- Access: ssh -l USERID login1.salomon.it4i.cz
- Load the Intel environment on the host via:
  \textit{module load intel impi}
- \textbf{Submit an interactive job via}
  \texttt{qsub -I -A DD-16-1 -q R553892.isrv5}
  \texttt{  -l select=1:accelerator=True,walltime=06:00:00 -l}
  \texttt{qstat -a}
  \texttt{qstat -u $USER}
Interacting with Intel Xeon Phi Coprocessors

```
user@host~$ micinfo -listdevices

MicInfo Utility Log
Created Thu Jan  7 09:33:20 2016
List of Available Devices
deviceId |  domain  | bus# | pciDev# | hardwareId
------------|-------------|---------|------------|----------------
0  |              0  |      2 0  |             0  |    2 2 5 0 8 0 8 6
1 |             0 |      8b |            0 |   22508086

user@host~$ micinfo | grep -i cores
Cores
  Total No of Active Cores : 61
Cores
  Total No of Active Cores : 61
```
Useful Tools and Files on Coprocessor

- `top` - display Linux tasks
- `ps` - report a snapshot of the current processes.
- `kill` - send signals to processes, or list signals
- `ifconfig` - configure a network interface
- `traceroute` - print the route packets take to network host
- `mpiexec.hydra` – run Intel MPI natively
- `/proc/cpuinfo`
- `/proc/meminfo`
user@host~$ /sbin/lspci | grep -i "co-processor"
20:00.0 Co-processor: Intel Corporation Xeon Phi .... (rev 20)
8b:00.0 Co-processor: Intel Corporation Xeon Phi .... (rev 20)

user@host~$ cat /etc/hosts | grep mic1
user@host~$ cat /etc/hosts | grep mic1-ib0 | wc –l
user@host~$ ssh mic0 or ssh mic1

user@host-mic0~$ ls /
bin boot dev etc home init lib lib64 lrz media mnt proc root sbin sys tmp usr var
• micsmc a utility for monitoring the physical parameters of Intel Xeon Phi coprocessors: model, memory, core rail temperatures, core frequency, power usage, etc.
Parallelism on the Heterogeneous Systems

Vectorisation - SIMD
Parallelism MPI/OpenMP
ccNUMA domains
Multiple accelerators

PCIe buses
Other I/O resources

Memory Interface
Memory

Other I/O

1 2 3 4 5 6
L1D L1D L1D L1D L1D L1D
L2 L2 L2 L2 L2 L2
L3 L3 L3 L3 L3 L3
Memory
Memory Interface
Memory Interface

7-8 February 2017
MIC Programming Models

- **Native Mode**
  - Programs started on Xeon Phi
  - Cross-compilation using `–mmic`
  - User access to Xeon Phi is necessary

- **Offload to MIC**
  - Offload using OpenMP extensions
  - Automatically offload some routines using MKL
    - MKL Compiled assisted offload (CAO)
    - MKL automatic Offload (AO)

- **MPI tasks on Host and MIC**
  - Treat the coprocessor like another host
    - MPI only and MPI + X (X may be OpenMP, TBB, Cilk, OpenCL…etc.)
MIC Programming Models

Parallelisation
- Intel Cilk Plus
- OpenMP
- OpenACC
- MPI
- Intel Threading Building Blocks
- OpenCL
- Pthreads

Ease of use
Vectorisation
- MKL
- Auto Vectorisation
- OpenMP 4.0 (#pragma omp simd)
- Vectorisation Pragmas (#pragma vector, ivdep, simd,...)
- Intel Cilk Plus Array Notation, Elemental Functions
- C/C++ Vector Classes (F32vec16, F64vec8)
- Intrinsics/Assembler

Fine control
Native Mode

- First ensure that the application is suitable for native execution.
- The application runs entirely on the MIC coprocessor without offload from a host system.
- Compile the application for native execution using the flag: `-mmic`
- Build also the required libraries for native execution.
- Copy the executable and any dependencies, such as run time libraries to the coprocessor.
- Mount file shares to the coprocessor for accessing input data sets and saving output data sets.
- Login to the coprocessor via console, setup the environment and run the executable.
- You can debug the native application via a debug server running on the coprocessor.
## Invocation of the Intel MPI compiler

<table>
<thead>
<tr>
<th>Language</th>
<th>MPI Compiler</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>mpiicc</td>
<td>icc</td>
</tr>
<tr>
<td>C++</td>
<td>mpiicpc</td>
<td>icpc</td>
</tr>
<tr>
<td>Fortran</td>
<td>mpiifort</td>
<td>ifort</td>
</tr>
</tbody>
</table>
Native Mode

• Compile on the Host:

~$ $INTEL_BASE/linux/bin/compilervars.sh intel64
~$ icpc -mmic hello.c -o hello
~$ifort -mmic hello.f90 -o hello

• Launch execution from the MIC:
~$ scp hello $HOST-mic0:
~$ ssh $HOST-mic0
~$ ./hello
hello, world
Native Mode: micnativeloadex

The tool automatically transfers the code and dependent libraries and executes from the host:

```bash
~$ ./hello
-bash: ./hello: cannot execute Binary file
~$ export SINK_LIBRARY_PATH=/intel/compiler/lib/mic
~$ micnativeloadex ./hello
  hello, world
~$ micnativeloadex ./hello -v
  hello, world
  Remote process returned: 0
  Exit reason: SHUTDOWN OK
```
Native Mode

- Use the Intel compiler flag `-mmic`
- Remove assembly and unnecessary dependencies
- Use `–host="—host=x86_64"` to avoid program does not run errors
- Example, the GNU Multiple Precision Arithmetic Library (GMP):

```bash
~$host: wget http://mirror.bibleonline.ru/gnu/gmp/gmp-6.1.0.tar.bz2
~$host: tar –xf gmp-6.1.0.tar.bz2
~$host: cd gmp.6.1.0
~$host: ./configure --prefix=${gmp_build} CC=icc CFLAGS="-mmic"
    –host=x86_64 –disable-assembly --enable-cxx
~$host: make
```
Lab 2: Native Mode
For the Labs:
Salomon System Initialisation

- Load the Intel environment on the host via:
  \textit{module load intel}

- Submit an interactive job via
  \texttt{qsub -I -A DD-16-44 -q R553892.isrv5}
  \texttt{-l select=1:ncpus=24:accelerator=True:naccelerators=2}
  \texttt{-l walltime=06:00:00}

- Exercise Code:
  \texttt{cp –r /home/weinberg/MIC_Workshop/ .}

- Exercise Sheets + Slides online:
  \url{https://goo.gl/IPBNmK}
Important MPI environment variables

- Important Paths are already set by intel module, otherwise use:
  - . $ICC_BASE/bin/compilervars.sh intel64
  - . $MPI_BASE/bin64/mpivars.sh

- Recommended environment on Salomon:

```
module load intel
export I_MPI_HYDRA_BOOTSTRAP=ssh
export I_MPI_MIC=enable
export I_MPI_FABRICS=shm:dapl
export I_MPI_DAPL_PROVIDER_LIST=ofa-v2-mlx4_0-1u,ofa-v2-scif0,ofa-v2-mcm-1
export MIC_LD_LIBRARY_PATH = $MIC_LD_LIBRARY_PATH:/apps/all/impi/5.1.2.150-iccifort-2016.1.150-GCC-4.9.3-2.25/mic/lib/
depending on version
```
micinfo and _SC_NPROCESSORS_ONLN

```
~$ micinfo -listdevices
~$ micinfo | grep -i cores
~$ cat hello.c
#include <stdio.h>
#include <unistd.h>
int main(){
    printf("Hello world! I have %ld logical cores.\n",
            sysconf(_SC_NPROCESSORS_ONLN));
}
```

```
~$ icc hello.c -o hello-host && ./hello-host
~$ icc -mmic hello.c -o hello-mic
~$ micnativeloadex ./hello-mic
```

7-8 February 2017

Intel MIC Programming Workshop @ Ostrava
The following network fabrics are available for the Intel Xeon Phi coprocessor:

<table>
<thead>
<tr>
<th>Fabric</th>
<th>Network Hardware and Software used</th>
</tr>
</thead>
<tbody>
<tr>
<td>shm</td>
<td>Shared-memory</td>
</tr>
<tr>
<td>tcp</td>
<td>TCP/IP-capable network fabrics, such as Ethernet and InfiniBand (through IPoIB)</td>
</tr>
<tr>
<td>ofa</td>
<td>OFA-capable network fabrics including InfiniBand (through OFED verbs)</td>
</tr>
<tr>
<td>dapl</td>
<td>DAPL–capable network fabrics, such as InfiniBand, iWarp, Dolphin, and XPMEM (through DAPL)</td>
</tr>
<tr>
<td>ofi</td>
<td>OFA-capable network fabrics such as Intel True fabric, Intel Omni-Path, InfiniBand and Ethernet</td>
</tr>
</tbody>
</table>
The default can be changed by setting the $I\_MPI\_FABRICS$ environment variable to $I\_MPI\_FABRICS=<fabric>$ or $I\_MPI\_FABRICS= <intra-node fabric>:<inter-nodes fabric>$

- **Intranode: Shared Memory, Internode: DAPL**
  - Default on SuperMIC/MUC
  - `export I_MPI_FABRICS=shm:dapl`

- **Intranode: Shared Memory, Internode: TCP**
  - Can be used in case of Infiniband problems
  - `export I_MPI_FABRICS=shm:tcp`
Thank you.