Intel MIC Programming Workshop: Hardware Overview & Native Execution
Dr. Momme Allalen (LRZ)
June, 26-28, 2017 @ LRZ
Agenda

- Intro @ accelerators on HPC
- Architecture overview of the Intel Xeon Phi Products (MIC)
- KNL vs KNC
- KNC Programming models
- What you need to know to start your code on KNC
- Native mode KNC and KNL programming
- Hands on
Why do we need “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.

• 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
  • @high power efficiency: keep the power/core as low as possible.
Why do we need “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 options, Intel Xeon Phi (KNC) and Nvidia GPU.
- Can perform many parallel operations every clock cycle.
Intel Multi-Core Architecture

- Intel Xeon processors are for general purpose
- The current architecture
  - Haswell and Broadwell
  - Skylake is upcoming
Architectures Comparison (CPU vs GPU)

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

- Simple flow control
- More transistors for computing in parallel (up to 21 billion on Nvidia Volta GPU)
  - (SIMD)

<table>
<thead>
<tr>
<th></th>
<th>Intel Xeon CPU E5-2697v4 “Brodwell”</th>
<th>Nvidia GPU P100</th>
<th>Nvidia GPU V100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores @ Clock</td>
<td>2 x 18 cores @ ≥ 2.3 GHz</td>
<td>56SMs @ 1.4 GHz</td>
<td>80SMs@1.4 GHz</td>
</tr>
<tr>
<td>SP Perf./core</td>
<td>≥ 73.6 GFlop/s</td>
<td>up to 166 GFlop/s</td>
<td></td>
</tr>
<tr>
<td>SP peak</td>
<td>≥ 2.6 TFlop/s</td>
<td>up to 10.6 TFlop/s</td>
<td>up to 15 TFlop/s</td>
</tr>
<tr>
<td>Transistors/TDP</td>
<td>2x7 Billion /2x145W</td>
<td>14 Billion/300W</td>
<td>21 Billion/300W</td>
</tr>
<tr>
<td>BW</td>
<td>2 x 62.5 GB/s</td>
<td>510 GB/s</td>
<td>up to 900 GB/s</td>
</tr>
</tbody>
</table>

Intel MIC Programming Workshop @ LRZ
June, 26-28, 2017
allalen@lrz.de
Intel Xeon Phi Products
Intel Many Integrated Core (MIC) Architecture

• Xeon Phi CoProcessor: first product was released in 2012 named Knights Corner (KNC) which is the first architecture supporting 512 bit vectors

• Xeon Phi Processor: 2nd generation announced at ISC16 in June named Knights Landing (KNL) also support 512bit vectors with a new instruction set called Intel Advanced Vector Instructions 512 (Intel AVX-512)

Specialised Platform for high demanding computing application
Intel KNC 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 with minimal shell environment
- Full Intel software tool suite
- 8 - 16 GB GDDR5 DRAM (cached)
- Up to 61 x86 (64 bit) in-order cores
- Up to 1 GHz
- 512 bit wide vector registers
- 4 way hyper-threading
- 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

June, 26-28, 2017
<table>
<thead>
<tr>
<th>Rank</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>Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China</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>Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.20GHz, TH Express-2, Intel Xeon Phi 3120P, NUDT National Super Computer Center in Guangzhou China</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>Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect, NVIDIA Tesla P100, Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland</td>
<td>361,760</td>
<td>19,590.0</td>
<td>25,326.3</td>
<td>2,272</td>
</tr>
<tr>
<td>4</td>
<td>Titan - Cray XK7, Opteron 6274 16C 2.20GHz, Cray Gemini interconnect, NVIDIA K20x, Cray Inc. DOE/SC/Oak Ridge National Laboratory United States</td>
<td>560,640</td>
<td>17,590.0</td>
<td>27,112.5</td>
<td>8,209</td>
</tr>
<tr>
<td>5</td>
<td>Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom, IBM DOE/NNSA/LLNL United States</td>
<td>1,572,864</td>
<td>17,173.2</td>
<td>20,132.7</td>
<td>7,890</td>
</tr>
<tr>
<td>6</td>
<td>Cori - Cray XE, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect, Cray Inc. DOE/SNS/BNL/NERSC United States</td>
<td>622,336</td>
<td>14,014.7</td>
<td>27,880.7</td>
<td>3,939</td>
</tr>
<tr>
<td>7</td>
<td>Oakforest-PACS - PRIMERGY CX441 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path, Fujitsu Joint Center for Advanced High Performance Computing Japan</td>
<td>556,104</td>
<td>13,554.6</td>
<td>24,913.5</td>
<td>2,719</td>
</tr>
<tr>
<td>8</td>
<td>K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect, Fujitsu RIKEN Advanced Institute for Computational Science (AICS) Japan</td>
<td>705,024</td>
<td>10,510.0</td>
<td>11,280.4</td>
<td>12,660</td>
</tr>
</tbody>
</table>
Intel Xeon Phi **KNL** Architecture

- Bootable CPU
- Up to 72 cores based on the Intel Atom cores (Silvermont microarchitecture)
- 4HT running @ 1.3 to 1.5 GHz
- 3+ TFlop/s in DP (FMA)
- 6+ TFlop/s in SP (FMA)
- ~ 384 GB DDR4 (> 90 GB/s)
- 16 GB HBM (MCDRAM)
  > 400 GB/s
- Binary-compatible with Xeon
- Common operating system (SUSE, WINDOWS, RHEL...)

June, 26-28, 2017
Intel MIC Programming Workshop @ LRZ
allalen@lrz.de
KNC vs KNL

- Co-processor
- Binary incompatible with other architectures
- 61 In-order cores
- 1.1 GHz processor
- up to 16 GB RAM
- 22 nm process
- One 512-bit VPU
- No support for branch prediction and fast unaligned memory access

- No PCIe - Self hosted
- Binary compatible with prior Xeon architectures (no phi)
- up to 72 Out-of-order cores
- 1.4 GHz processor
- Up to 400 GB RAM (with MCDRAM)
- 14 nm process
- Tow 512-bit VPUs
- Support for branch prediction and fast unaligned memory access

The Improvement on the KNL Hardware still not good for non optimised code

June, 26-28, 2017
## 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>
Lab1: Access SuperMIC @LRZ
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 : 60
Cores
Total No of Active Cores : 60
```
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`
Interacting with Intel Xeon Phi Coprocessors

```
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.
```
Intel Fabric integrated on KNL processor

- Intel released KNL-F on Nov. 2016; KNL with a Fabric
  - Intel Omni-Path Architecture
  - High Bandwidth and low latency
  - The Omni-Path technology will allow to build Clusters like: LRZ-KNL Cluster
- Other Xeon Phi based system: Server, Workstation ..

[Link: dap.xeonphi.com]
C/C++/Fortran, Python/Java .... Porting is easy
Two parallelisation modes are required: Shared memory and vectorisation
Run multiple threads/processes and each thread issues vector instructions (SIMD)
KNL: Cores and threads

- Up to 36 tiles connected by 2D Mesh interconnect each with 2 physical cores (up to 72 cores with out of order instruction execution)
- Distributed L2 cache across a mesh interconnect
KNL Tile (Cores and threads)

- Up to 72 cores with 4 way hyper threading up to 288 logical processors

- Up to 36 MB L2 per KNL
KNL and Vector Instruction Sets

- Binary runs without recompilation
- KNC binary requires recompilation

- Conflict Detection: Improves Vectorisation
- Prefetch: Gather and Scatter Prefetch
- Exponential and Reciprocal Instructions

SNB E5-2600
- x87/MMX
- SSE*
- AVX
- AVX2

HSW E5-2600
- x87/MMX
- SSE*
- AVX
- AVX2

KNL
- x87/MMX
- SSE*
- AVX
- AVX2
- AVX-512F
- AVX-512CD
- AVX-512PF
- AVX-512ER

are going to be used in future Xeon architecture like SKX
Memory on KNL

- Two levels of memory on KNL:
  1. Main memory
     - KNL has direct access to all of main memory
     - Similar latency and bandwidth as a standard Xeon processors
     - 6 DDR channels
  2. Multi-Channels DRAM or MCDRAM
     - HBM on chip: 16GB
     - Slightly higher latency than main memory
     - 8 MCDRAM controllers, 16 channels
Using MCDRAM on KNL

- At boot time you have to choose one memory mode operation

**Flat Mode**
- MCDRAM treated as a NUMA node
- as separately addressable memory
- Users control what goes to MCDRAM

**Cache Mode**
- MCDRAM treated as a transparent Last Level Cache (LLC)
- MCDRAM is used automatically

**Hybrid Mode**
- Combination of Flat and Cache
- Ratio can be chosen in the BIOS
• Flat mode offers the best performance for applications, but require changes to the code (memory allocation) or the execution environment (NUMA node)

numactl —membind 0 ./exec # DDR4
numactl —membind 1 ./exec # MCDRAM
**Programming Models on KNC**

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

- **Offload to MIC (KNC)**
  - 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.)
Native Mode on KNC

• 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.
Native Mode on KNC

• Compile on the Host:

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

• Launch execution from the MIC (KNC):

~$ scp hello $HOST-mic0:
~$ ssh $HOST-mic0
~$ ./hello.knc
hello, world
The tool automatically transfer the code and dependent libraries and execute from the host:

```
~$ ./hello.knc
-bash: ./hello: cannot execute Binary file
~$ export SINK_LIBRARY_PATH=../intel/compiler/lib/mic
~$ micnativeloadex ./hello.knc
  hello, world
~$ micnativeloadex ./hello.knc -v
  hello, world
Remote process returned: 0
Exit reason: SHUTDOWN OK
```
Lab2: Native Mode
For the Labs:
SuperMIC System Initialisation

- Exercise Sheets + Slides online:
  https://goo.gl/IPBNmK
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
Lab3: Access KNL test system
Interacting with Intel Xeon Phi Processors

```bash
user@host~$ ssh lxlogin1.lrz.de -l Your-User-ID
user@host~$ ssh mcct03.cos.lrz.de -l Your-User-ID
user@host~$ ssh mcct04.cos.lrz.de -l Your-User-ID
user@host~$ module list or module av

user@mcct03~$ numactl -H
node 0 size: 96457 MB
node 0 free: 92947 MB
node 1 cpus:
node 1 size: 16011 MB
node 1 free: 15865 MB
node distances:
node 0 1
  0: 10 31
  1: 31 10

user@mcct04~$ numactl -H
node 0 size: 96341 MB
node 0 free: 79787 MB
node distances:
node 0
  0: 10
```