C-DAC Four Days Technology Workshop

ON

Hybrid Computing – Coprocessors/Accelerators
Power-Aware Computing – Performance of Applications Kernels

hyPACK-2013

Mode 3 : Intel Xeon Phi Coprocessors

Lecture Topic :
Intel Xeon-Phi Prog. –
MPI-OpenMP-Intel TBB - An Overview

***Venue** : CMSD, UoHYD ; **Date** : October 15-18, 2013*

An Overview of Xeon Phi Coprocessor

Lecture Outline

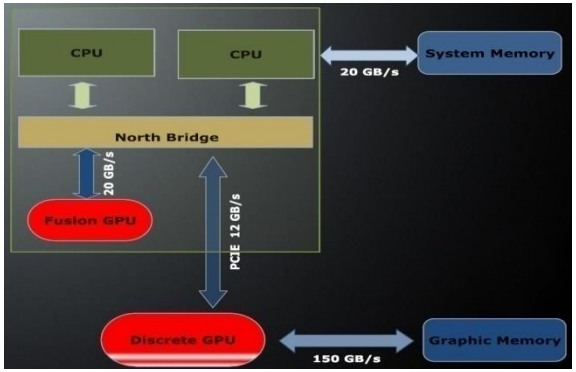
Following topics will be discussed

- Understanding of Xeon –Phi Architectures
- Programming on Xeon-Phi – Prog. MPI-OpenMP-Intel TBB
- Tuning & Performance – Software Threading

Intel Xeon-Phi
Shared Address Space Programming
OpenMP, Intel TBB
Explicit Message Passing - MPI

Systems with Accelerators

A set (one or more) of very simple execution units that can perform few operations (with respect to standard CPU) with very high efficiency. When combined with full featured CPU (CISC or RISC) can accelerate the “nominal” speed of a system.



CPU — ACC

Single thread perf. **throughput**

CPU ACC.

Physical integration

CPU & ACC

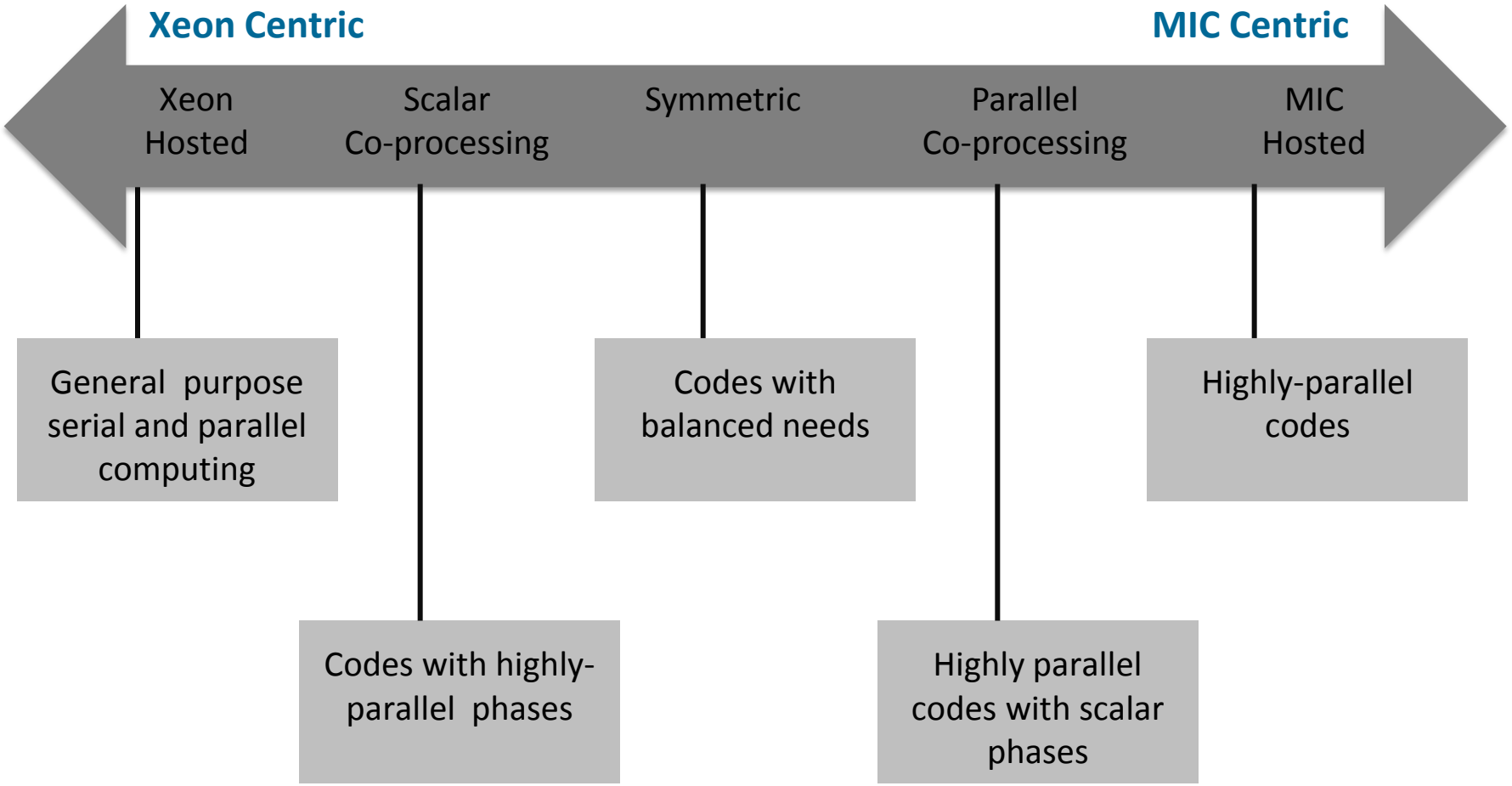
Architectural integration

CPU : Control Path & Data Path
ALU : Arithmetic and logic units
ACC: Accumulator
PC : Program counter
Micro-Instructions (Register transfer)

Source : NVIDIA, AMD, SGI, Intel, IBM Alter, Xilinux References

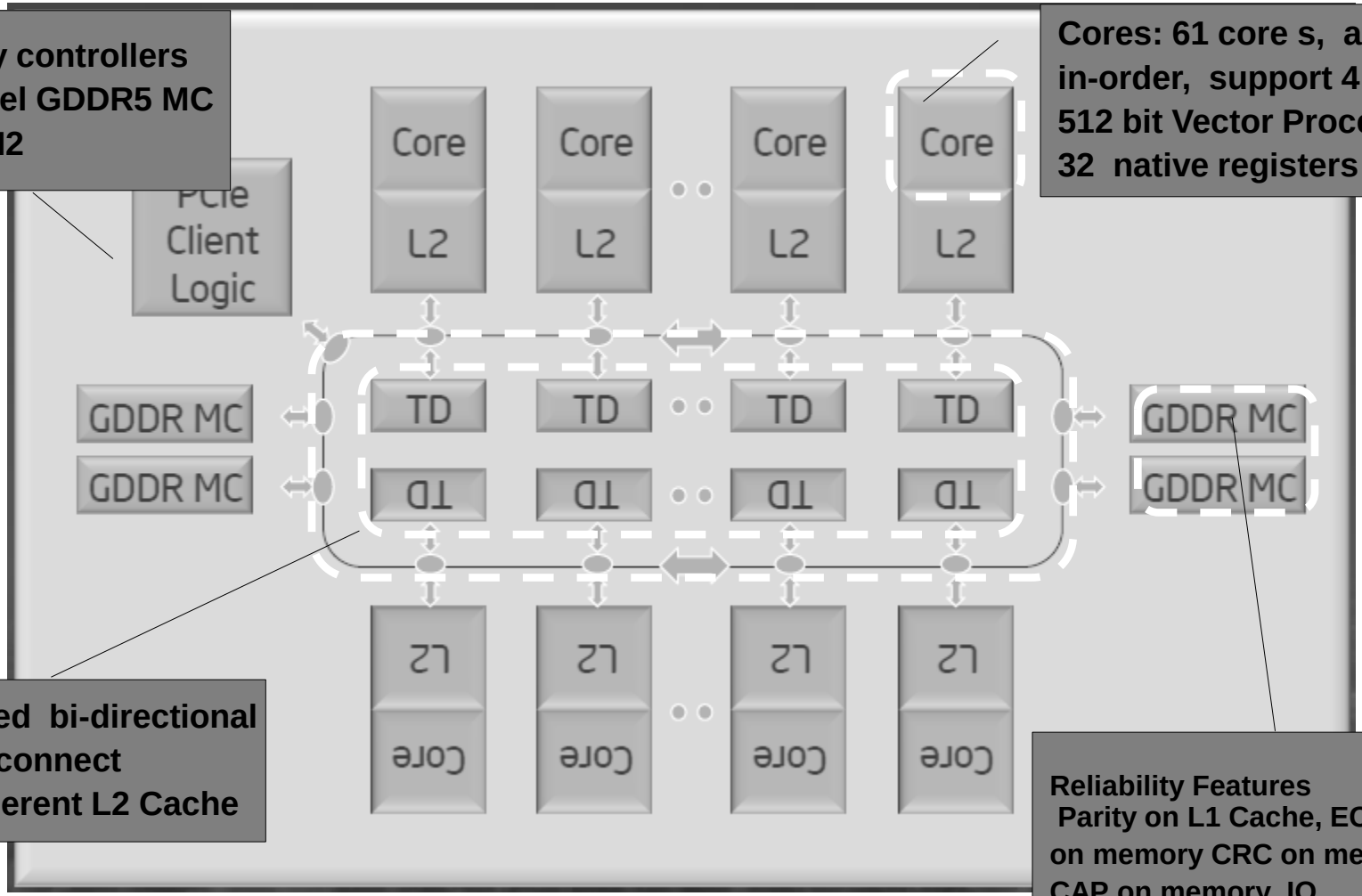
MIC Architecture, System Overview

Compute modes vision



Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel® Xeon Phi™ Architecture Overview



8 memory controllers
16 Channel GDDR5 MC
PCIe GEN2

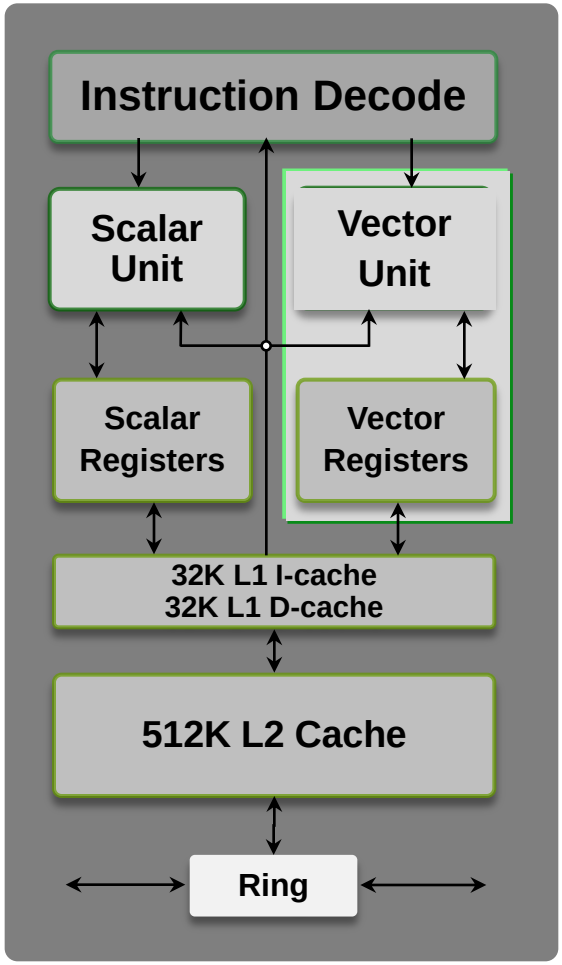
Cores: 61 core s, at 1.1 GHz
in-order, support 4 threads
512 bit Vector Processing Unit
32 native registers

High-speed bi-directional ring interconnect
Fully Coherent L2 Cache

Reliability Features
Parity on L1 Cache, ECC
on memory CRC on memory IO,
CAP on memory IO

Source : References & Intel Xeon-Phi; http://www.intel.com/

Core Architecture Overview



- 60+ in-order, low power IA cores in a ring interconnect
- Two pipelines
  - Scalar Unit based on Pentium® processors
  - Dual issue with scalar instructions
  - Pipelined one-per-clock scalar throughput
- SIMD Vector Processing Engine
- 4 hardware threads per core
  - 4 clock latency, hidden by round-robin scheduling of threads
  - Cannot issue back to back inst in same thread
- Coherent 512KB L2 Cache per core

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor architecture Overview

Quick Glance*

- The Intel Xeon Phi coprocessor Architecture Overview (Core, VPU, CRI, Ring, SBOX, GBOX, PMU)
- The Cache hierarchy (Details of L1 & L2 Cache)
- Network Configuration (MPSS) : (Obtain the information can be obtained by running the micinfo program on the host.)
- System Access

Remark : Root privileges are necessary for the destination directories (Required for availability of some library usage for codes such MKL)

(= Useful for tuning and Performance)*

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor architecture Overview

- The Intel Xeon Phi coprocessor consists of up to 61 cores connected by a high performance on-die bidirectional interconnect.
- The coprocessor runs a full service Linux operating system
- The coprocessor supports all important Intel development tools, like C/C++ and Fortran compiler, MPI and OpenMP
- To Coprocessor support s high performance libraries like MKL, debugger and tracing tools like Intel VTune Amplifier XE.

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor architecture Overview

- The Intel Xeon Phi coprocessor The coprocessor is connected to an Intel Xeon processor - the "host" - via the PCI Express (PICe) bus.
- The implementation of a virtualized TCP/IP stack allows to access the coprocessor like a network node.

Remark : Summarized information can be found In the following MIC architecture from the System Software Developers Guide and other references

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

- Details about the system startup and the network configuration can be found in Intel Xeon-Phi documentation coming with MPSS
- To start the Intel Manycore Platform Software Stack (Intel MPSS) and initialize the Xeon Phi coprocessor the following command has to be executed as root or during host system start-up:

```
hypack-root@mic-0:~> sudo service mpss start
```

Remark : The above command has to be executed as a root

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

- To start the Intel Manycore Platform Software Stack (Intel MPSS) and initialize the Xeon Phi coprocessor the following command has to be executed as root or during host system start-up:

```
hypack-root@mic-0:~> sudo service mpss start
```

Remark : The above command has to be executed as a root.

Details about the system startup and the network configuration can be found in Intel Xeon-Phi documentation coming with MPSS. For other necessary commands, refer Intel Xeon Phi documentation

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

- Deafault IP addresses ???•?? •?•??? , ???•?? •?•???, etc. are assigned to the attached Intel Xeon Phi coprocessors. The IP addresses of the attached coprocessors can be listed via the traditional **ifconfig** Linux program.

```
hypack-root@mic-0:~> /sbin/ifconfig
```

Further information can be obtained by running the **micinfo** program on the host.

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
```

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
System Info
 Host OS : Linux
 OS Version : 3.0.13-0.27-default
 Driver Version : 4346-16
 MPSS Version : 2.1.4346-16
 Host Physical Memory : 66056 MB
 ……
 Device No: 0, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……
 Version
 ……..…
 Board
 …….
```

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

```
hypack-root@mic-0:~> /sudo/opt/intel/mic/bin/micinfo
 Device No: 0, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……..
 Core
 ……..…
 Thermal
 ……..
 GGDR
 ……..
 Device No: 1, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……..
 ……..
 ……..
```

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

```
hypack-root@mic-0:~>/sudo/opt/intel/mic/bin/micinfo
 Device No: 0, Device Name: Intel(R) Xeon Phi(TM) coprocessor
 ……..
 Core
 ……..…
```

Intel Xeon-Phi Coprocessor System Access

Quick Glance:

Users can log in directly onto the Xeon Phi coprocessor via ssh. User can get basic information abbot Xeon-Phi by executing the following commands.

```
[hypack01@mic-0]$ ssh mic-0
[hypack01@mic-0]$ hostname
```

.....

```
[hypack01@mic-0]$ cat /etc/issue
Intel MIC Platform Software Stack release 2.X
```

To get further information about the cores, memory etc. can be obtained from the virtual Linux /proc or /sys filesystems:

```
[hypack01@mic-0]$ tail -n26 /proc/cpuinfo
```

.....

OpenMP

Intel Xeon-Phi : OpenMP I Prog. Model

- OpenMP parallelization on an "**Intel Xeon + Xeon Phi coprocessor machine**" can be applied in **four** different programming models.

  - **Realized with Complier Options**

Intel Xeon-Phi : OpenMP I Prog. Model

- **Four** Models with different programming models
  - Native OpenMP on the Xeon host
  - Serial Xeon host with OpenMP offload
  - OpenMP on the Xeon Host with OpenMP offload
  - Native OpenMP on the Xeon Phi coprocessor

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : OpenMP Prog. Model

❖ **Remark :**

- OpenMP threads on Xeon Host and OpenMP threads on Xeon Phi do not interface each other and when an offload/pragma section of the code is encountered

- Offloaded as a **Unit** and uses a number of threads based on available resources on **Xeon Phi** coprocessor

- Usual semantics of **OpenMP** Constructs apply on Xeon host and Xeon-Phi Coprocessor

Intel Xeon-Phi : OpenMP Prog. Model

❖ **Remark :**

- Offload to the **Xeon Phi** coprocessor can be done at any time by multiple host CPUs until the filling of the available resources.

- If there are **no** free threads, the task meant to be offloaded may be done on the **host**.

- For offload schemes, the **maximal amount** of threads that can be used on the Xeon Phi coprocessor is **4 times** the total number of cores **minus one**, because **one core** is reserved for the **OS** and its **services**.

Intel Xeon-Phi : OpenMP Prog. Model

❖ Threading and affinity

- **Important Considerations** for OpenMP threading and affinity are the total number of threads that should be utilized and the scheme for binding threads to processor cores.
- The Xeon Phi coprocessor supports 4 threads per core.
- Using more than one core is recommended.
- When running applications natively on the Xeon Phi the full amount of threads can be used.
- On Xeon host, benefit from hyper-threading exists.

Intel Xeon-Phi : OpenMP Prog. Model

Threading and affinity : Settings :

Settings	Description
OpenMP on host without HT	1 x ncore-host
OpenMP on host with HT	2 x ncore-host
OpenMP on Xeon Phi in native mode	4 x ncore-phi
OpenMP on Xeon Phi in offload mode	1 x ncore-phi-1

- If OpenMP regions exist on the **host** and on the part of the code **offloaded** to the Xeon Phi, **two** separate OpenMP runtimes exist.

Intel Xeon-Phi : OpenMP Prog. Model

❖ Threading and affinity

- Environment variables for controlling OpenMP behavior are to be set for both runtimes

For example

- the **KMP_AFFINITY** variable which can be used to assign a particular thread to a particular physical node. For
- Intel Xeon Phi it can be done like this:
- **export MIC_ENV_PREFIX=MIC**

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : OpenMPI Prog. Model

❖ Threading and affinity

```
export MIC_ENV_PREFIX=MIC

#specify affinity for all cards
export MIC_KMP_AFFINITY=...

#specify number of threads for all cards
export MIC_OMP_NUM_THREADS=120

#specify the number of threads for card #2
export MIC_2_OMP_NUM_THREADS=200

#specify number of threads and affinity for card #3
export MIC_3_ENV="OMP_NUM_THREADS=60 |
                  KMP_AFFINITY=balanced"
```

Intel Xeon-Phi : OpenMP Prog. Model

Threading and affinity

One can also use special **API calls** to set the environment for the coprocessor only, **e.g.**

```
omp_set_num_threads_target()
```

```
omp_set_nested_target()
```

Intel Xeon-Phi : OpenMP Prog. Model

Loop Scheduling

- OpenMP accepts four different kinds of loop scheduling - **`static`**, **`dynamic`**, **`guided`** & **`auto`**.

- The **`schedule`** clause can be used to set the loop scheduling at compile time.

- Another way to control this feature is to specify schedule(runtime) in your code and select the loop scheduling at runtime through setting the **`OMP_SCHEDULE`** environment variable

Intel Xeon-Phi : OpenMP Prog. Model

Scalability

- Use **`-collapse`** directive to specify how many for-loops are associated with the OpenMP loop construct

- Another way to improve scalability is to reduce barrier synchronization overheads by using the **`nowait`** directive.

- Another way to control this feature is to specify schedule(runtime) in your code and select the loop scheduling at runtime through setting the **`OMP_SCHEDULE`** environment variable

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : MPI Prog. Model

Setting Up the MPI Environment

The following commands have to be executed to set up the MPI environment:

copy MPI libraries and binaries to the card (as root)

only copying really necessary files saves memory

```
scp /opt/intel/impi/4.1.0.024/mic/lib/* mic0:/lib
scp /opt/intel/impi/4.1.0.024/mic/bin/* mic0:/bin
```

setup Intel compiler variables

```
. /opt/intel/composerxe/bin/compilervars.sh intel64
```

setup Intel MPI variables

```
. /opt/intel/impi/4.1.0.024/bin64/mpivars.sh
```

Intel Xeon-Phi : Hybrid MPI/OpenMP

Programming Models

Two Major Approaches

1. A **MPI offload** approach (MPI ranks reside on the host CPU and work is offloaded to the Xeon Phi Coprocessor

1. A **symmetric** approach in which MPI ranks reside both on the CPU and on the Xeon Phi.

AMPI program can be structured using either model

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : Hybrid MPI/OpenMP

Programming Models : Threading of MPI ranks

1. For **hybrid OpenMP/MPI** applications use the **thread safe** version of the Intel MPI Library by using the `-mt_mpi` compiler driver option.
2. A desired process pinning scheme can be set with the `I_MPI_PIN_DOMAIN` environment variable. It is recommended to use the following setting:

```
$exportI_MPI_PIN_DOMAIN = omp
```

By using this, one sets the process pinning domain size to be `OMP_NUM_THREADS.` In this way, every MPI process is able to create `$OMP_NUM_THREADS` number of threads that will run within the corresponding domain.

Intel Xeon-Phi : Hybrid MPI/OpenMP

Programming Models : Threading of MPI ranks

It is recommended to use the following setting:

```
$exportI_MPI_PIN_DOMAIN = omp
```

- Using this, one sets the process pinning domain size to be **OMP_NUM_THREADS.** Every MPI process is able to create **$OMP_NUM_THREADS** no. of threads that will run within the corresponding domain.
- If this variable is not set, each process will create a number of threads per MPI process equal to the no. of cores (treated as a separate domain.)
- To pin OpenMP threads within a particular domain, one could use the **KMP_AFFINITY** environment variable

Intel Xeon-Phi : MPI Programming Model

Setting up the MPI environment :

Details about using the Intel MPI library on Xeon Phi coprocessor systems can be found in references

The following commands have to be executed to set up the MPI environment:

copy MPI libraries and binaries to the card (as root)

only copying really necessary files saves memory

```
scp /opt/intel/impi/4.1.0.024/mic/lib/* mic0:/lib
scp /opt/intel/impi/4.1.0.024/mic/bin/* mic0:/bin
```

setup Intel compiler variables

```
. /opt/intel/composerxe/bin/compilervars.sh intel64
```

setup Intel MPI variables

```
. /opt/intel/impi/4.1.0.024/bin64/mpivars.sh
```

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi Coprocessor : MPI on Cluster

Network Fabric : The following network fabrics are available for the Intel Xeon Phi coprocessor (Refer C-DAC PARAM YUVA Cluster)

Fabric Name	Description
shm	Shared-memory
tcp	TCP/IP-capable network fabrics, such as Ethernet and InfiniBand (through IPoIB)
ofa	OFA-capable network fabric including InfiniBand (through OFED verbs)
dapl	DAPL–capable network fabrics, such as InfiniBand, iWarp, Dolphin, and XPMEM (through DAPL)

The Intel MPI library tries to automatically use the best available network fabric detected (usually shm for in-tra-node communication and InfiniBand (dapl, ofa) for inter-node communication).

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 fab-ric>. The availability is checked in the following order: shm:dapl, shm:ofa, shm:tcp.

Source : References & Intel Xeon-Phi; http://www.intel.com/

MPI- Prog. Model

Intel Xeon-Phi : MPI Programming Model

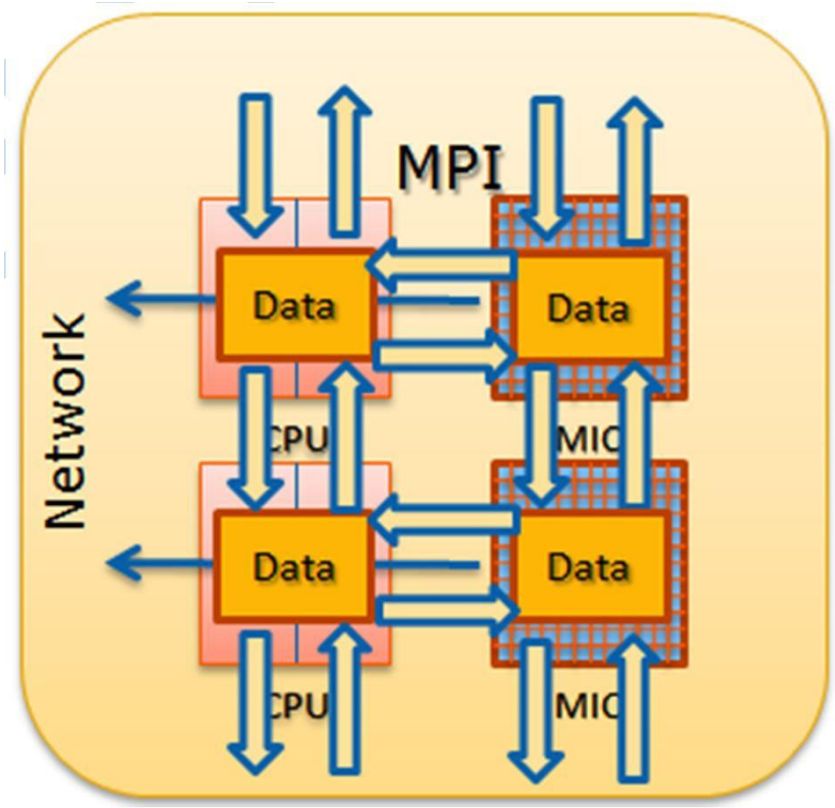
- Intel MPI for the Xeon Phi coprocessors offers various MPI programming models:
  - **Symmetric model :** The MPI ranks reside on both the host and the coprocessor. Most general MPI case.
  - **Coprocessor-only model :** All MPI ranks reside only on the coprocessors
  - **Host-only model All :** MPI ranks reside on the host. The coprocessors can be used by using offload pragmas. (Using MPI calls inside offloaded code is not supported

Intel Xeon-Phi : MPI Programming Model

- Intel MPI for the Xeon Phi coprocessors offers various MPI programming models:
  - **Symmetric model :** The MPI ranks reside on both the host and the coprocessor. Most general MPI case.
  - **Coprocessor-only model :** All MPI ranks reside only on the coprocessors
  - **Host-only model All :** MPI ranks reside on the host. The coprocessors can be used by using offload pragmas. (Using MPI calls inside offloaded code is not supported

Source : References & Intel Xeon-Phi; http://www.intel.com/

Symmetric Model



MPI on Host Devices and Co-processors

- The MPI processes reside on both the host and the MIC devices
- This model involves both the host CPUs and the co-processors into the execution of the MPI processes and the related MPI communications.
- Message passing is supported inside the co-processor, inside the host node, and between the co-processor and the host. environment variable
- Most general MPI view of an essentially heterogeneous cluster.

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : MPI Programming Model

❖ **Symmetric model** To build and run an application in **host-only** mode, the following commands have to be executed:

compile the program for the host (offloading is enabled per default

```
mpiicc –mmic -o hello.MIC hello.c
```

launch MPI jobs on the host “ycn-0”,the MPI process will offload code for acceleration

```
mpirun –host ycn-1 -n 1 ./hello
```

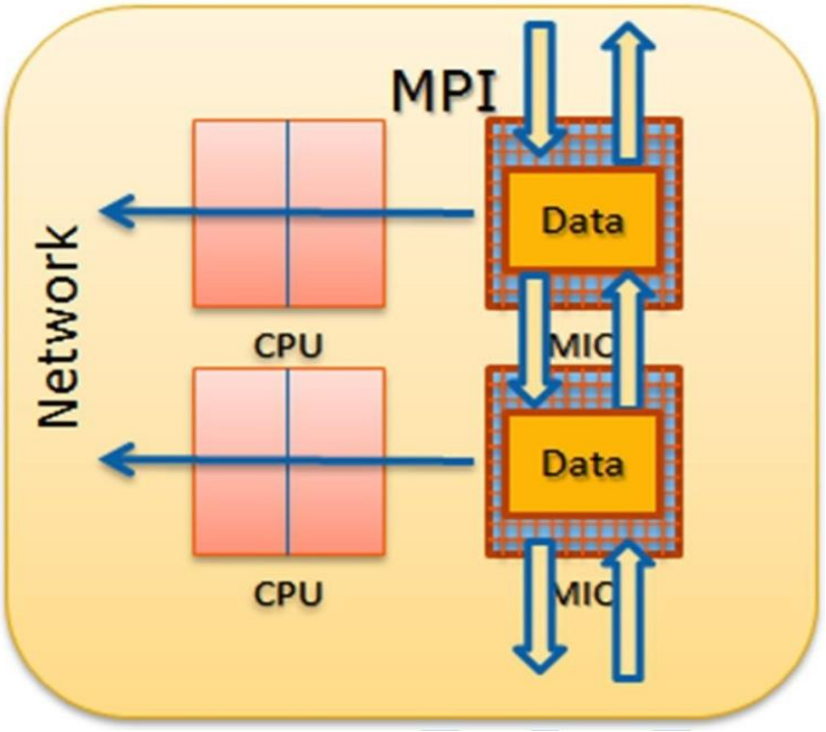
Intel Xeon-Phi : MPI Programming Model

- **Coprocessor-only model** To build and run an application in coprocessor-only mode, the following commands have to be executed:

```
# compile the program for the coprocessor (-mmic)
mpiicc -mmic -o hello.MIC hello.c

#copy the executable to the coprocessor
scp hello.MIC mic0:/tmp
#set the I_MPI_MIC variable
export I_MPI_MIC=1
#launch MPI jobs on the coprocessor mic0 from the host
#(alternatively one can login to the coprocessor and
   run mpirun there)
mpirun -host mic0 -n 2 /tmp/hello.MIC
```

Co-processor-only Model (or MPI Native)



MPI on Co-processors

- The MPI processes reside on the MIC co-processor only .
- MPI libraries, the application, and other needed libraries are uploaded to the co-processors.
- An application can be launched from the host or the co-processor.
- This can be seen as a specific case of the symmetric model

Source : References & Intel Xeon-Phi; http://www.intel.com/

MPI Prog. Models for Xeon systems with MIC

Offload

- Intel ® MIC Architecture or host CPU as an accelerator

MIC Offload (direct acceleration)

- MPI ranks on the host CPU only
- Messages into/out of the host CPU
- Intel ® MIC Architecture as an accelerator

Host Offload (reverse acceleration)

- MPI ranks on the MIC CPU only
- Messages into/out of the MIC CPU
- Host CPU as an accelerator

MPI

- MPI ranks on several co-processors and/or host nodes
- Messages to/from any core

Co-processor-only

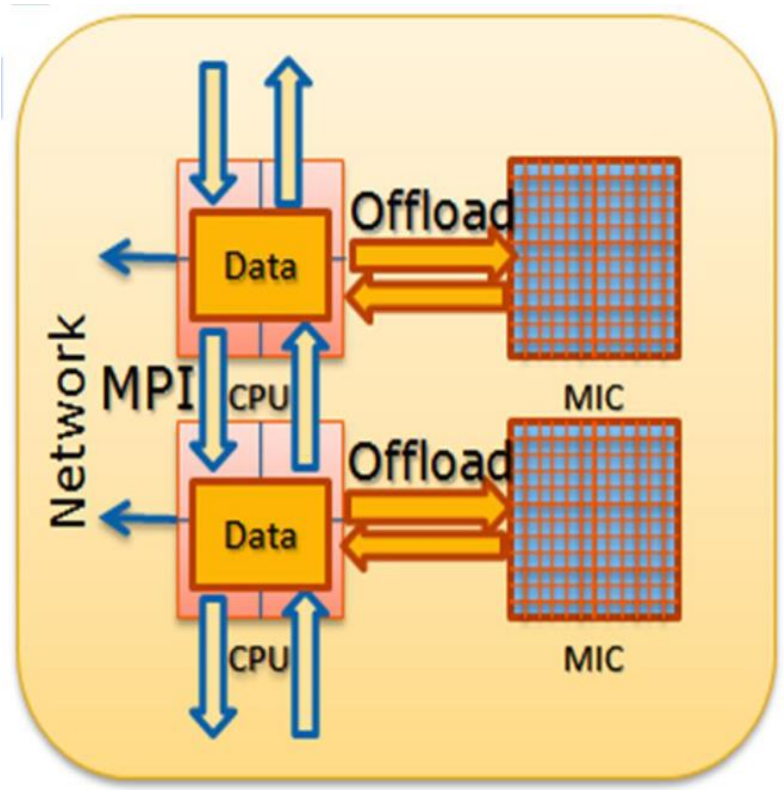
- MPI ranks on the MIC CPU only
- Messages into/out of the MIC CPU c/o host CPUs
- Threading possible

Symmetric

- MPI ranks on the MIC and host CPUs
- Messages into/ out of the MIC and host CPUs
- Threading possible

Source : References & Intel Xeon-Phi; http://www.intel.com/

Offload Model



MPI on Host Devices with Offload to Co-processors

- This model is characterized by the MPI communications taking place only between the host processors.
- The co-processors are used exclusively thru the offload capabilities of the products like Intel C, C++, and Fortran Compiler for Intel MIC Architecture, Intel Math Kernel Library (MKL), etc.
- This mode of operation is already supported by the Intel MPI Library for Linux OS

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : MPI Programming Model

❖ **Host-only model** To build and run an application in **host-only** mode, the following commands have to be executed:

compile the program for the host (offloading is enabled per default

```
mpiicc -o hello.MIC hello.c
```

launch MPI jobs on the host “ycn-0”,the MPI process will offload code for acceleration

```
mpirun –host ycn-1 -n 1 ./hello
```

Intel Xeon-Phi : MPI Programming Model

Simple way to Launch MPI jobs :

Instead of specifying the hosts and coprocessors via **`-n hostname`** one can also put the names into a **`hostfile`** and launch the jobs via

```
mpirun -f hostfile -n 4 ./hello
```

Note that the executable must have the same name on the hosts and the coprocessors in this case. If one sets **`export`**

```
I_MPI_POSTFIX=.mic
```

the **`.mic`** postfix is automatically added to the executable name by **`mpirun`**, so in the case of the example above test is launched on the host and **`hello.mic`** on the coprocessors.

Intel Xeon-Phi : MPI Programming Model

Simple way to Launch MPI jobs :

```
mpirun -f hostfile -n 4 ./hello

I_MPI_POSTFIX=.mic
```

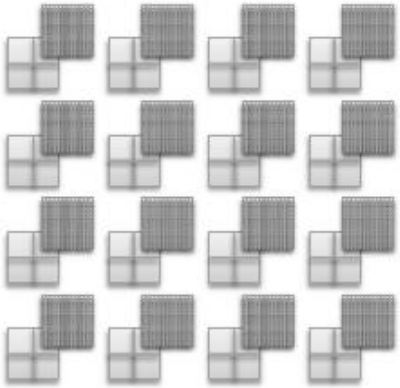
It is also possible to specify a prefix using

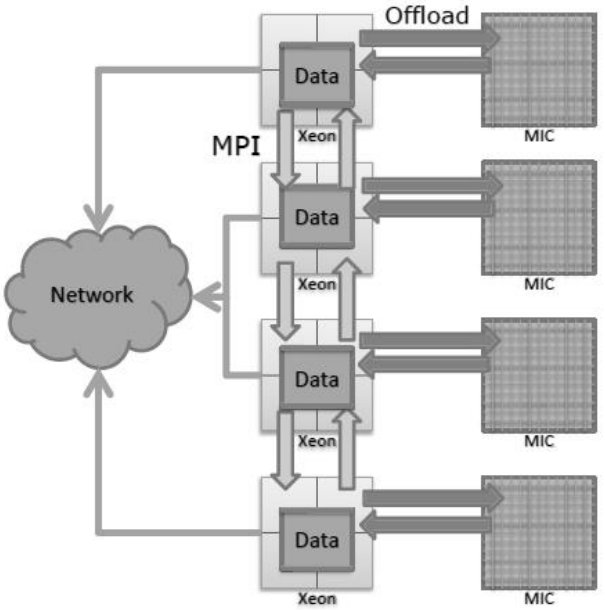
```
export I_MPI_PREFIX=./MIC/
```

In this case `./MIC/hello` will be launched on the coprocessor. This is specially useful if the host and the coprocessors share the same NFS file system

Programming Intel MIC-based Systems MPI+Offload

- MPI ranks on Intel ® Xeon ® processor (only)
- All messages into/out of processors
- Offload models used to accelerate MPI ranks
- Intel CilkTM Plus, Open MP*, Intel Threading Building Blocks, Pthreads* within Intel ®MIC
- Homogenous network of hybrid nodes:

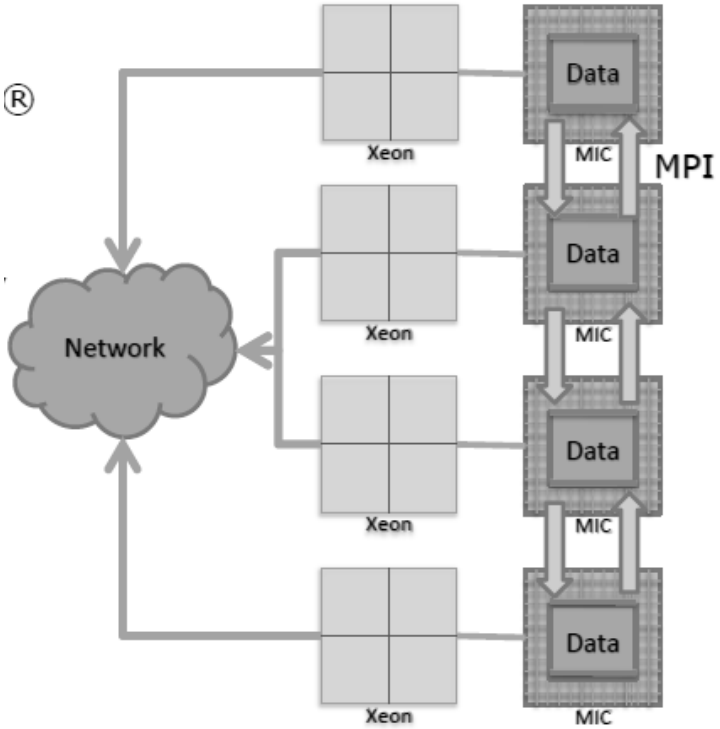


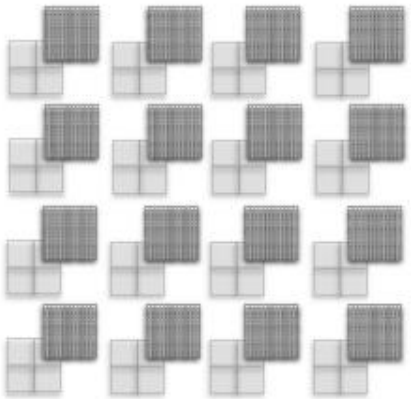


Source : References & Intel Xeon-Phi; http://www.intel.com/

Programming Intel ® MIC-based Systems *Many-core Hosted*

- MPI ranks on Intel ® MIC (only)
- All messages into/out of Intel ® MIC
- Intel ® CilkTM Plus, Open MP*, Intel ® Threading Building Blocks, Pthreads* used directly within MPI processes
- Programmed as homogenous network of many-core

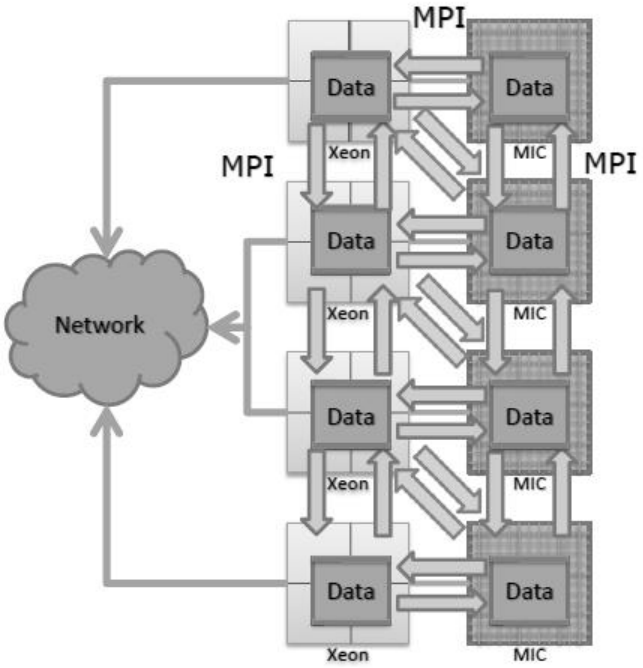


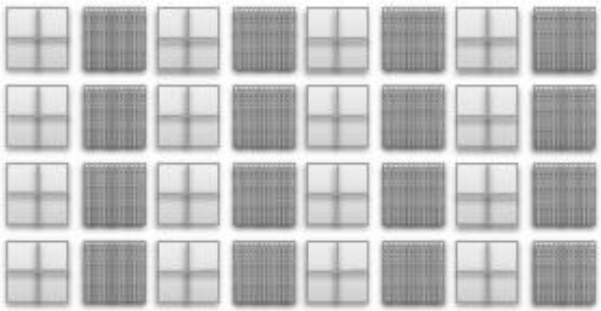


Source : References & Intel Xeon-Phi; http://www.intel.com/

Programming Intel ® MIC-based Systems *Symmetric*

- MPI ranks on Intel ® MIC Intel ® Xeon ® processors
- Messages into/out anu core
- Intel ® CilkTM Plus, Open MP*, Intel ® Threading Building Blocks, Pthreads* used directly within MPI processes
- Programmed as heteregenous network of homogenous





Source : References & Intel Xeon-Phi; http://www.intel.com/

Keys to Productive Performance on Intel ® MIC Architecture

- Choose the right Multi-core centric or Many-core centric model for your application
- Vectorize your application (today)
  - Use the Intel Vectorizing compiler
- Parallelize your application (today)
  - With MPI (or other multi-process model)
  - With threads (via Intel (R) CilkTM Plus, OpenMP*, Intel (R) Threading Buildig Blocks, Pthreads, etc.)
- Go asynchronous

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel TBB

Intel Xeon-Phi : Intel TBB Prog.

- **Rule of thumb :** An application must scale well past one hundred threads on Intel Xeon processors to profit from the possible higher parallel performance offered with e.g. the Intel Xeon Phi coprocessor.

- The scaling would profit from utilising the highly parallel capabilities of the MIC architecture, you should start to create a simple performance graph with a varying number of threads (from one up to the number of cores)

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : Intel TBB Prog.

- **Rule of thumb :** An application must scale well past one hundred threads on Intel Xeon processors to profit from the possible higher parallel performance offered with e.g. the Intel Xeon Phi coprocessor.

- The scaling would profit from utilising the highly parallel capabilities of the MIC architecture, you should start to create a simple performance graph with a varying number of threads (from one up to the number of cores)

Intel Xeon-Phi : Intel TBB Prog.

- **What we should know from programming point of view :** We treat the coprocessor as a 64-bit x86 **SMP-on-a-chip** with an high-speed bi-directional **ring** interconnect, (up to) **four** hardware threads per core and **512-bit SIMD** instructions.

- With the available number of cores, we have easily 200 hardware threads at hand on a single coprocessor.

Intel Xeon System & Xeon-Phi

About Hyper-Threading

- hyper-threading hardware threads can be switched off and can be ignored.

About Threading on Xeon-Phi Coprocessor

- The multi-threading on each core is primarily used to hide latencies that come implicitly with an in-order microarchitecture. Unlike hyper-threading these hardware threads cannot be switched off and should never be ignored.
- In general a minimum of **three** or **four** active threads per cores will be needed.

Intel Xeon-Phi : Intel TBB Prog.

Intel TBB Advantages

- Intel TBB Generic Programming
- Intel TBB is easy to start
- Intel TBB obeys to logical parallelism:
- Intel TBB is compatible with other programming models:
- The Intel TBB template-based approach – Performance gain can be achieved.

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : Intel TBB Prog.

Intel TBB : Using TBB NATIVELY

A minimal C++ TBB example looks as follows:

```
#include “tbb/task_scheduler_init.h”
#include “tbb/parallel_for.h”
#include "tbb/blocked_range.h"
   using namespace tbb;
   int main() {
   task_scheduler_init init;
   return 0;
}
```

Intel Xeon-Phi : Intel TBB Prog.

Intel TBB : Using TBB NATIVELY

- Scalable parallelism can be achieved by **parallelizing** a loop of iterations that can each run independently from each other. The **`parallel_for`** template function replaces a serial loop

- A typical example would be to apply a function **`MatAdd`** on all elements of an array over the iterations space of type **`size_t`** going from **`0`** to **`n-1`**

Source : References & Intel Xeon-Phi; http://www.intel.com/

Intel Xeon-Phi : Intel TBB Prog.

Intel TBB : Using TBB NATIVELY

```
void SerialApplyMatAdd(float a[], size_t n) {
for( size_t i=0; i!=n; ++i )
     MatAdd(a[i]);
}
```

becomes

```
void  ParallelApplyMatAdd(float  a[],size_t  n)
{
     parallel_for(size_t(0),n,[=](size_ti)
     {MatAdd(a[i]);});
}
```

Compiling programs that employ TBB constructs, link in the Intel TBB shared library with `-ltbb`.

```
icc -mmic -ltbb foo.cpp
```

Intel Xeon-Phi : Intel TBB Prog.

Intel TBB : Using TBB NATIVELY

The Intel TBB header files are not available on the Intel MIC target environment by default (the same is also true for Intel Cilk Plus). To make them available on the coprocessor the header files have to be wrapped with

#pragma offloaddirectives as demonstrated in the example below:

```
#pragma offload_attribute (push,target(mic))
#include “tbb/task_scheduler_init.h”
#include “tbb/parallel_for.h”
#include "tbb/blocked_range.h"
#pragma offload_attribute (pop)
```

Intel Xeon-Phi : Intel TBB Prog.

Intel TBB : Using TBB NATIVELY

Functions called from within the offloaded construct and global data required on the Intel Xeon Phi coprocessor

should be appended by the special function **`attribute __attribute__((target(mic)))`**.

Codes using Intel TBB with an offload should be compiled with **`-tbbflag`** instead of **`-ltbb`**.

Compiling programs that employ TBB constructs, link in the Intel TBB shared library with **`-ltbb`**.

```
icc -mmic -ltbb foo.cpp
```

An Overview of Multi-Core Processors

Conclusions

- An Overview of Xeon-Phi Architectures, Programming on based on Shared Address Space Platforms – OpenMP, MPI, Intel TBB, Performance of Software threading are discussed.

Thank You

Any questions ?