# Overview of Intel Xeon Phi Coprocessor

Sept 20, 2013

Ritu Arora

Texas Advanced Computing Center

Email: rauta@tacc.utexas.edu

TACC

THE UNIVERSITY OF TEXAS AT AUSTIN

# This talk is only a trailer…

A comprehensive training on running and optimizing your code for Intel Xeon Phi will be held on October 4, 2013

Title of the Training: Optimize Your Code for the Intel XEON Phi

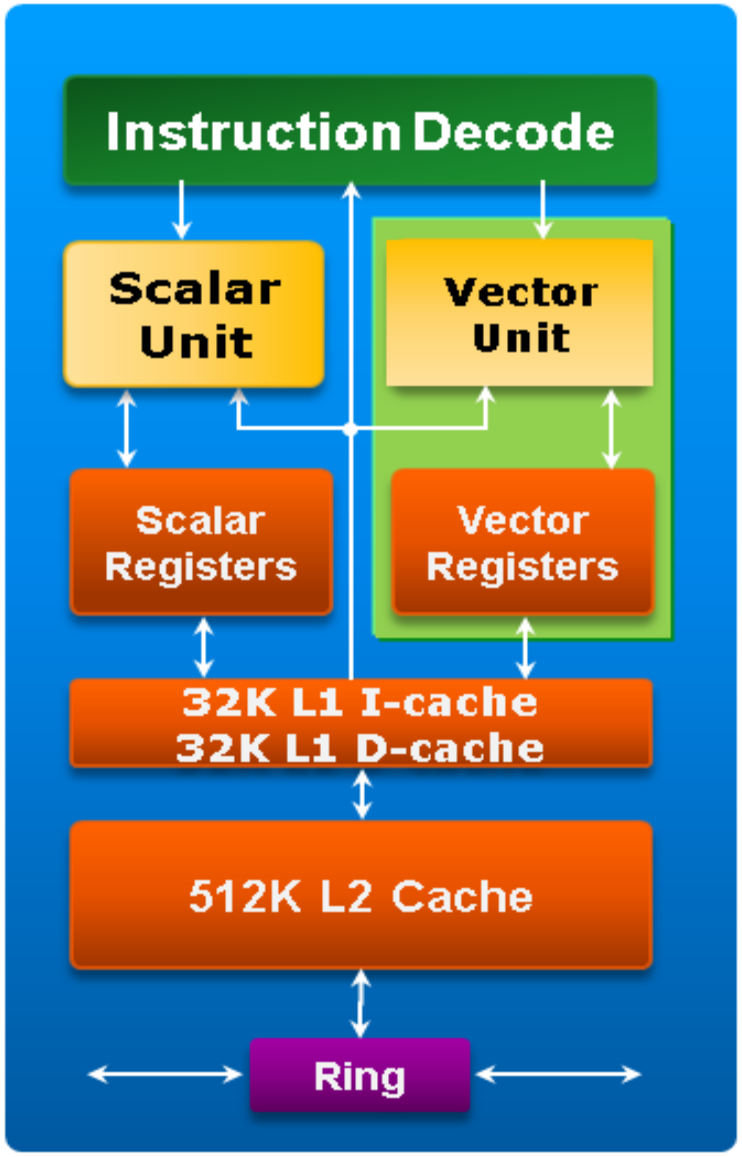
# Intel Xeon Phi Coprocessor – What is it?

- Also known as Intel Many Integrated Core (MIC) architecture, leverages the x86 architecture and is about large die with simple circuit.

- Is a highly parallel device - provides up to 61 cores, 244 threads, and 1.2 teraFLOPS of performance

- Comes in a variety of configurations to address diverse hardware, software, workload, performance, and efficiency requirements.
  - The majority of the 6400 nodes on Stampede are configured with two Xeon E5-2680 processors and one Intel Xeon Phi SE10P Coprocessor (on a PCIe card)

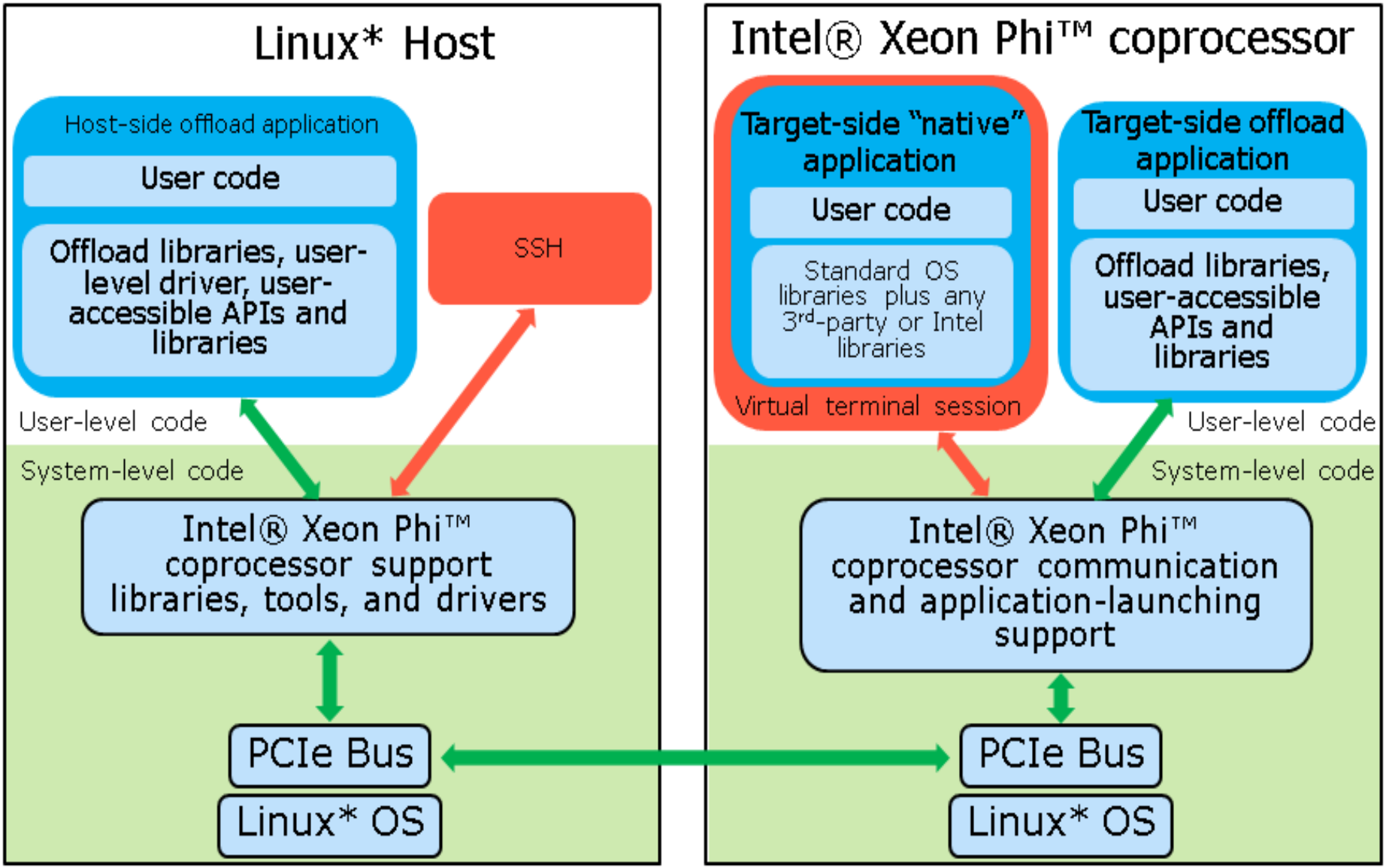
# An Intel Xeon Phi Architecture Core




- Each core includes a newly-designed Vector Processing Unit (VPU)
- The VPU is a key feature of the Intel Xeon Phi Architecture-based cores
- Each vector unit contains 32 512-bit vector registers
- Fully utilizing the vector unit is critical for best performance

Source: Reference 1

# Software Stack




Source: Reference 1

# Developing Applications for Intel Xeon Phi

- Because it is based on the x86 technology, applications for Intel Xeon Phi can be developed using existing knowledge of multi-core and SIMD programming
  - Programming and optimizing for Intel Xeon Phi coprocessors is similar to programming and optimizing for Intel Xeon processors

- Porting applications to Intel Xeon Phi
  - You can port sections of your code (written in C/C++ or FORTRAN) to run on Intel Xeon Phi using the offload language extensions
  - You can also port your entire application to Intel Xeon Phi

- Highly parallel applications that also use SIMD operations for most of their execution get best performance results on Intel Xeon Phi

# Development Environment: Available Compilers and Libraries

- Compilers for building applications that run on Intel 64 architecture and Intel Xeon Phi Architecture
  - Intel C++ Composer XE 2013
  - Intel Fortran Composer XE 2013
- Libraries packaged with the compilers include:
  - Intel Math Kernel Library (Intel MKL) optimized for the Intel Xeon Phi
  - Intel Threading Building Blocks (Intel TBB)
  - Intel Integrated Performance Primitive (Intel IPP)
- Libraries packaged separately include:
  - Intel MPI for Linux OS

# Intel Xeon Phi Execution Models

1. Offload Execution Mode

2. Coprocessor Native Execution Mode

3. Symmetric Execution Mode

# Intel Xeon Phi Execution Models: Offload Mode

- The host system (Xeon E5 processor on Stampede) offloads part or all of the computation to the coprocessor
- The application starts execution on the host
- As the computation proceeds, the host can decide to send data to the coprocessor and let that work on it
- The host and the coprocessor may or may not work in parallel
- Note: Offload code will not run on the 61$^{st}$ core of Phi as the offload daemon runs here

# Intel Xeon Phi Execution Models: Coprocessor Native Execution Mode

- The coprocessor can be viewed as another compute node and users run their applications directly on it without offload from a host system

- In order to run natively, an application has to be cross-compiled for Xeon Phi operating environment

- Intel Composer XE provides simple switch (`-mmic`) to generate cross-compiled code

- Execute the compiled code on host or `ssh` to `mic0` to run on Phi

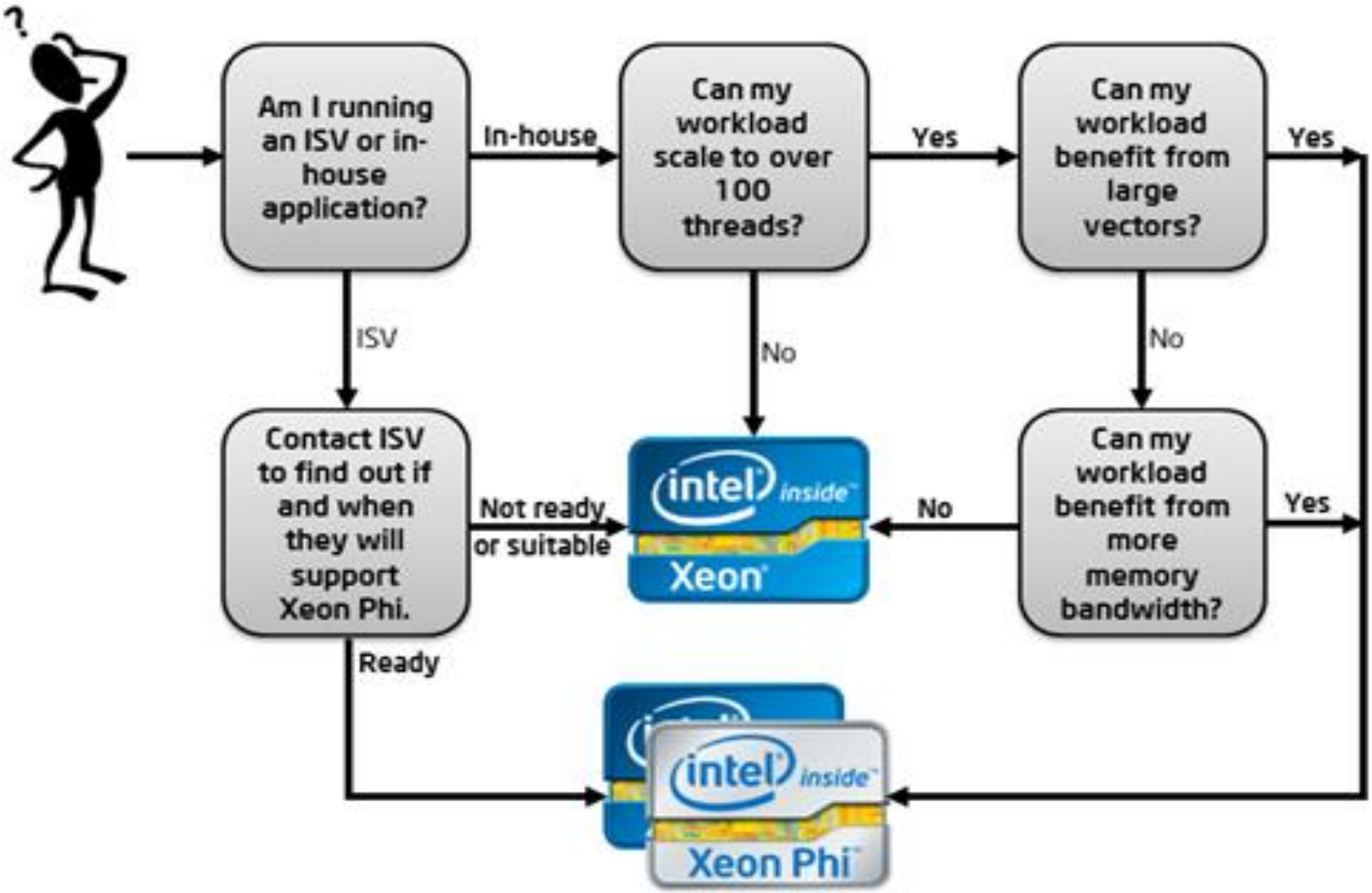
# Intel Xeon Phi Execution Models: Symmetric Execution Mode

- In this case the application processes run on both the host and the Intel Xeon Phi coprocessor
- They usually communicate through a message passing interface
- This execution environment treats Xeon Phi card as another node in a cluster in a heterogeneous cluster environment
- Note: Load-balancing between Xeon cores and Xeon Phi cores needed

# Intel MKL Automatic Offload Model

- Few of the host Intel MKL functions are Automatic Offload-aware
- Call such functions as you normally would on the host
- If you have preceded the library call with a call to `mkl_mic_enable()`, Intel MKL will automatically decide at runtime whether some or all of the work required to complete the call should be divided between the host and the Intel Xeon Phi Coprocessor
  - depending upon problem size, the load on both processors, and other metrics
- Turn this functionality off with `mkl_mic_disable()`

# Is Intel Xeon Phi architecture for me?




Source: Reference 2

# Sample Code (1)

- Sample offload code using the explicit memory copy model can be found on Stampede at the paths below:

C++

**/opt/apps/intel/13/composer_xe_2013.0.062/Samples/en_US/C++/mic_samples**

Fortran

**/opt/apps/intel/13/composer_xe_2013.0.062/Samples/en_US/Fortran/mic_samples/**

# Sample Code (2)

- Sample offload code using MKL in automatic offload mode:

**/opt/apps/intel/13/composer_xe_2013.0.062/mkl/examples/mic_samples/ao_sgemm**

**/opt/apps/intel/13/composer_xe_2013.0.062/mkl/examples/mic_samples/ao_sgemm_f**

# Native Execution Mode Example (1)

- A simple hello world program in `hello.c`

```
//Content of hello.c
int main() {
  printf("Hello world from Intel Xeon Phi\n");
  return 0;
}
```

- Steps for compiling and running the code on Intel Xeon Phi

```
login1$ icc -mmic -o hello hello.c
login1$ srun --pty -A A-ccsc -p normal-mic -t 1:00:00 -n
16 /bin/bash -l
c455-002$ scp hello mic0:
c455-002$ ssh mic0
~ $ ./hello
Hello world from Intel Xeon Phi
```

# Native Execution Mode Example (2)

- Once the code has been compiled on the host for execution on MIC, user can launch the executable on the MIC directly from the host, either implicitly or explicitly – no need to `ssh` to `mic0` (like it was shown on previous slide)

- This is accomplished with the help of native application launcher on Stampede known as `micrun`

```
c455-002$ micrun /work/01698/rauta/training/mic_training/hello
Hello world from Intel Xeon Phi
```

- The application will run on the coprocessor from the directory where it was launched on the host, will propagate the application's return value to the shell, and will automatically propagate and translate any environment variables that contain the `MIC_` prefix.

# References


1. http://software.intel.com/sites/default/files/article/335818/intel-xeon-phi-coprocessor-quick-start-developers-guide.pdf
2. http://software.intel.com/en-us/articles/is-the-intel-xeon-phi-coprocessor-right-for-me
3. http://software.intel.com/en-us/articles/intel-xeon-phi-programming-environment
4. https://www.cac.cornell.edu/vw/mic/native.aspx