Advanced Computing Center for Research and Education

Using Intel Xeon Phi Co-Processors

www.accre.vanderbilt.edu
• Think of the Phi as a compute device that can run mathematically intensive portions of a program faster than a traditional central processing unit (CPU)

• The Phi *can be* treated as a standalone system where a user logs in and builds/runs programs directly
  • The Phi runs its own simplified Linux micro operating system ($\mu$OS)

• Generally, the Phi is used as a co-processor alongside the CPU
  • The CPU runs serial portions of a program, and then offloads mathematically intensive portions of the program to the Phi
Intel Xeon Phi: Basics

- Physically, the Phi is a card that plugs in through a PCI express slot (much like a GPU) on the motherboard
  - Future generations (“Knight’s Landing”) will sit in their own socket on the motherboard
- The Phi has its own on-board memory
  - Data must be copied from system memory to Phi memory and back again, which introduces overhead
  - The amount of computation relative to memory transfers should be high to warrant the use of the Phi
Is the Phi just a GPU?

• While similar, there are many significant differences between the Phi and a NVIDIA GPU:
  • CPU code can be directly built for the Phi (x86 architecture)
  • Intel has invested significant effort in developing libraries (e.g. MKL) that automatically offload compute-intensive operations to the Phi (no changes to code required)
  • Many programming paradigms supported by Phi (MPI, OpenMP, auto-vectorization, Intel TBBs, Intel Cilk)
  • Future Phi cards will sit in their own socket in the motherboard
  • Phis tend to have lower peak performance than GPUs
## CPU vs. Phi

<table>
<thead>
<tr>
<th></th>
<th>2 x Intel Xeon E5-2670 CPU</th>
<th>Intel Xeon Phi (7120P)</th>
</tr>
</thead>
<tbody>
<tr>
<td># cores</td>
<td>16</td>
<td>61</td>
</tr>
<tr>
<td>hardware threads per core</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>clock rate (GHz)</td>
<td>2.60</td>
<td>1.24</td>
</tr>
<tr>
<td>RAM (GB)</td>
<td>132</td>
<td>16</td>
</tr>
<tr>
<td>memory bandwidth (GB/s)</td>
<td>102</td>
<td>352</td>
</tr>
<tr>
<td>vector processor size (bits)</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>peak performance (GFLOPs; double precision)</td>
<td>333</td>
<td>1208</td>
</tr>
<tr>
<td>power (W)</td>
<td>115</td>
<td>300</td>
</tr>
<tr>
<td>performance per watt</td>
<td>2.9</td>
<td>4.0</td>
</tr>
</tbody>
</table>

- Max theoretical speedup: 1208 / 333 ~ 3.6
Additional Information

- Phi terminology
  - Phi also commonly referred to as a MIC (Many Integrated Core) card
  - Current generation of card referred to as Knight’s Corner
  - CPU processor = host; Phi card = device

- Software support
  - Many popular scientific packages and codes already have support for the Phi
  - See: https://software.intel.com/en-us/xeonphionlinecatalog
Good Phi Applications

- Data parallel applications
  - Identical instructions being repeatedly performed on independent pieces of data
  - Often matrix and vector operations are already implemented in parallel, in which case it’s just a matter of linking against Intel’s libraries

- Memory intensive applications
  - Applications limited by memory access speed may benefit from the Phi where the peak bandwidth is over three times higher than standard system memory
  - Less common
Programming Approaches

- **Automatic Offloading**
  - Use Intel’s MKL library to automatically offload expensive operations to the Phi

- **Native Phi Programming**
  - Build executable to run completely on Phi

- **Explicit Offloading**
  - Insert directives in code to offload portions of code to the Phi

- **Symmetric MPI**
  - Run program across host and device simultaneously, where the workload is balanced between the two
Examples for these programming approaches are available in an ACCRE Github repository (includes examples for Matlab, Python, and R)

Visit: https://github.com/accre/Intel-Xeon-Phi.git

Or download examples on the cluster:

$ setpkgs -a git
$ git clone https://github.com/accre/Intel-Xeon-Phi.git
Automatic Offloading

- Intel’s Math Kernel Library (MKL) supports automatic offloading to the Phi
  - Support for many common matrix operations (matrix multiply, Choleski factorization, single value decomposition, principal components analysis, dot product, etc.)
  - The library determines whether to run the computation on the host or device
  - Computations are run symmetrically across the host and device (or multiple devices, if present)
  - Many statistical analysis and simulation techniques use matrix algebra routines “under the hood”
Automatic Offloading

- Automatic offloading enabled on the cluster for Matlab, Python, and R
  - In many cases, it can be as simple as adding a line or two to your SLURM script to activate the AO version of the software
  - Need to load the Intel-compiled version of the software (see example SLURM script below)

```sh
#SBATCH --time=8:00:00
....
setpkgs -a R_3.1.1_intel
setpkgs -a intel_cluster_studio_compiler
export MKL_MIC_ENABLE=1
Rscript --no-save bmark.R
```
Automatic Offloading

Example execution times (in seconds) in Python for 20,000 x 20,000 matrix

- Single Value Decomposition (x 10^1)
  - Phi*: 2458.4
  - Host**: 2584.7

- Matrix Inversion
  - Phi*: 297
  - Host**: 1123

- Dot Product
  - Phi*: 314
  - Host**: 1913

*2 available MIC cards in node

**Multithreaded execution, 16 cores
These lines load the appropriate Phi-supported software.

This line is useful for verifying that computation is actually being offloaded to the Phi.

These lines enable us to control the number of OpenMP threads to launch across the MIC cores (240 is the max; often 120 is a sweet spot).

```bash
#!/bin/bash
#SBATCH --mail-user=vunetid@vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --partition=mic
#SBATCH --nodes=1
#SBATCH --time=1-0:0:00
#SBATCH --mem=64G
#SBATCH --output=automatic-offload.out

# load Python and Intel environment
setpkgs -a python2.7.8_intel14
setpkgs -a intel_cluster_studio_compiler

# This single line allows MKL to decide whether to offload certain matrix operations to one or both MICs
export MKL_MIC_ENABLE=1
# This next line will generates reports what was offloaded to the MIC
export OFFLOAD_REPORT=2

# Control number of threads to offload onto MIC card(s)
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=120

# Run program
python bmark.py
```

https://github.com/accre/Intel-Xeon-Phi/blob/master/Python/automatic-offloading2/automatic-offload.slurm
Troubleshooting

• The Intel libraries will determine whether the computation is sufficiently large to justify using the Phi
  • For example, it would not be efficient to offload a 10x10 matrix operation to the Phi
• It might be that the matrix or vector operation you are performing does not have Phi support
• Analyze the offload report for any hints about performance
Automatic Offloading

- Examples for Matlab, Python, and R in ACCRE Github repository

Visit: [https://github.com/accre/Intel-Xeon-Phi.git](https://github.com/accre/Intel-Xeon-Phi.git)

Or download examples on the cluster:

```
$ setpkgs -a git
$ git clone https://github.com/accre/Intel-Xeon-Phi.git
```

- More information on automatic offloading can be found here: [https://software.intel.com/sites/default/files/11MIC42_How_to_Use_MKL_Automatic_Offload_0.pdf](https://software.intel.com/sites/default/files/11MIC42_How_to_Use_MKL_Automatic_Offload_0.pdf)
Native Phi Programming

- Rather than offloading certain operations to the Phi, you can also run an entire program directly from the Phi
  - Build from the Phi host (must be logged into a Phi node), run from Phi card; just include the -mic flag at compile-time
  - Use program called `micrun` to run native-compiled executables, or log into a MIC card and run directly
  - Convention is to give executable file a `.mic` extension (e.g. `myProg.mic`)
  - Can safely use all 61 cores but should generally only use 60 (1 core reserved for the µOS and/or offload daemon)
  - Generally use OpenMP or some other shared memory programming interface for parallelization
Native Phi Programming

- Typical compile command:

```bash
icc -o sample.mic openmp_sample.c -std=c99 -O3 -mmic -vec-report3 -openmp
```

Flags:

- `-std=c99`: Code conforms to C99 language standards
- `-O3`: Level three optimization
- `-mmic`: Build with MIC instruction set
- `-vec-report3`: Report auto-vectorization at compile-time
- `-openmp`: Build with OpenMP support
Vectorization

- Single instruction, multiple data (SIMD) instructions that enable the processor to execute multiple loop iterations in a single clock cycle
- Phis have a 512-bit wide vector processing unit, allowing 16 SP floating point, 16 integer, or 8 DP floating point computations simultaneously
- Example vectorization report (-vec-report3):

  openmp_sample.c(97): (col. 3) remark: LOOP WAS VECTORIZED
  openmp_sample.c(103): (col. 5) remark: LOOP WAS VECTORIZED
  openmp_sample.c(102): (col. 3) remark: loop was not vectorized: not inner loop
  openmp_sample.c(108): (col. 3) remark: loop was not vectorized: existence of vector dependence
  openmp_sample.c(123): (col. 7) remark: loop was not vectorized: loop was transformed to memset or memcpy

Native Phi Programming

#!/bin/bash
SBATCH --mail-user=vunetid@vanderbilt.edu
SBATCH --mail-type=ALL
SBATCH --partition=mic
SBATCH --nodes=1
SBATCH --time=1:00:00
SBATCH --mem=64G
SBATCH --output=mic-test.out

# load Intel environment and build executable
setpkgs -a intel_cluster_studio_compiler
make

# This next line will control the number of OpenMP # threads that are executed on the MIC. By default the # program will be run across 61 * 4 = 244 threads
export MIC_OMP_NUM_THREADS=120

# micrun is a command for launching native compiled # MIC binaries from the host
micrun ./sample.mic

https://github.com/accre/Intel-Xeon-Phi/blob/master/Native/native-mic.slurm
Explicit Offloading

- Amounts to including directives to the compiler for offloading expensive operations (generally loops) to the Phi
  - Generally used in combination with a multithreading interface (e.g. OpenMP) to enable offloaded instructions to be performed in parallel on the Phi
  - Can coordinate which MIC card(s) is/are used, data transfers, MIC memory management, etc.
  - Ideally, the host continues working while the MIC card is processing data (asynchronous offload)
  - “export OFFLOAD_REPORT=2” useful for monitoring offload at runtime
Explicit Offloading

```
#pragma omp parallel for
for ( i=0; i<500000; i++ ) {
    a[i] = (double)i;
}
```

```
#pragma omp parallel for
for ( i=0; i<500000; i++ ) {
    a[i] = (double)i;
}
```

```
#pragma omp parallel for
for ( i=0; i<100000; i++ ) {
    c[i] = a[i] + b[i];
    d[i] = a[i] - b[i];
    b[i] = -b[i];
}
```

Offloads for loop to any available MIC card visible to host. Runs loop in parallel on MIC using OpenMP.

Offloads for loop to MIC card 0. Runs loop in parallel on MIC using OpenMP.

Provides information to compiler about what variables need to be copied to/from the MIC card.

---

SC 2014 Conference, Intel Xeon Phi Workshop
Explicit Offloading

```
#include <stdio.h>
#include <omp.h>

int main()
{
    double sum;
    int i,n,nt;

    n=2000000000;
    sum=0.0e0;

    #pragma offload target(mic:0)
    {
        #pragma omp parallel for reduction(+:sum)
        for(i=1;i<=n;i++)
        {
            sum = sum + i;
        }
    }
    nt = omp_get_max_threads();

    #ifdef __MIC__
        printf("Hello MIC reduction %f threads: %d\n",sum,nt);
    #else
        printf("Hello CPU reduction %f threads: %d\n",sum,nt);
    #endif
}
```

```
#!/bin/bash
#SBATCH --mail-user=vunetid@vanderbilt.edu
#SBATCH --mail-type=ALL
#SBATCH --partition=mic
#SBATCH --nodes=1
#SBATCH --time=5:00
#SBATCH --mem=64G
#SBATCH --output=offload-test.out

# load Intel environment and build
setpkgs -a intel_cluster_studio_compiler
make

# These next two lines will control the number of OpenMP threads that are executed on the MIC. By default the program will be run across 60 * 4 = 240 threads
export MIC_ENV_PREFIX=MIC
export MIC_OMP_NUM_THREADS=120

# Run program
./reduce_offload_mic
```

https://github.com/accre/Intel-Xeon-Phi/tree/master/Offload
Symmetric Model

- Run MPI tasks across host and device simultaneously
- Requires building and running separate executables for the host and device
- Program tuning is essential
  - Balancing MPI tasks (and threads within tasks) between host and device to achieve optimal load balancing and performance
- Beyond the scope of this course, but it is possible!
Phi Nodes at ACCRE

- Currently four Phi nodes in ACCRE cluster for development and prototyping
  - vmp902, vmp903, vmp904, vmp905
  - Each node equipped with 2 Intel Xeon Phi co-processors (7120P model): 61 cores, 4 hardware threads/core, 1.24 GHz, 15.8 GB RAM (per card)
  - Each node also equipped with 2 Intel Xeon E5-2670 CPU processors: 8 cores each, 2 hardware threads/core, 2.60 GHz; 132 GB system RAM
Phi-Enabled Software

- Intel Cluster Studio installed on cluster; provides all the Intel compilers and libraries with Phi support
- Packages on ACCRE including Phi support:
  - Matlab 2014a and later (setpkgs -a matlab)
  - R 3.2.0 (setpkgs -a R_3.2.0)
  - Python 2.7.8 / NumPy 1.9.1 (setpkgs -a python2.7.8_intel14)
  - LAMMPS version July 2015 (setpkgs -a lammps_mic)
- Examples for all these packages available at the ACCRE Github page: https://github.com/accre/Intel-Xeon-Phi
Phi Node Policies

- Usage is free!
- New users must open a help desk ticket requesting access
  - We will enable access through SLURM so we can track usage
- For development, please only use one Phi node at a time; production benchmarks may use more than one node but if demand is high we may limit access since there are only four nodes total
- Each job gets an entire node (including access to both cards); this may change if demand is high
Submitting Jobs

• Checking Phi node availability:

$ sinfofeatures
NODELIST FEATURES AVAIL NODES(A/I)
... vmp[902-905] mic up 1/3

1 node allocated (A), 3 nodes idle (I)

• Running interactive job on Phi nodes:

$ salloc --partition=mic --account=accre_mic --time=2:00:00

• Batch job:

... #SBATCH --partition=mic
#SBATCH --account=accre_mic
#SBATCH --time=2:00:00
#SBATCH --mem=120G
...
More Resources

- Open a helpdesk ticket requesting access, assistance, or if you have any questions: http://www.accre.vanderbilt.edu/?page_id=369
- Intel Xeon Phi Developer’s Guide: Click here
- Software packages with Phi support: https://software.intel.com/en-us/xeonphionlinecatalog
- Automatic Offloading White paper: Click here
- Phi architecture, White paper: Click here
- More useful links:
  - https://www.msi.umn.edu/sites/default/files/Phi_Intro.pdf
Questions?

@ACCREVandy  facebook.com/accrevandy  github.com/accre