The Intel Xeon Phi

Mike Bailey
mjb@cs.oregonstate.edu
Oregon State University

[Diagram of the Intel Xeon Phi architecture]

[Graph showing performance metrics]

Oregon State University
Computer Graphics

OSU_Xeon_Phi.pptx mjb – March 30, 2015
Setup

Xeon system
rabbit.engr.oregonstate.edu
2 E5-2630 Xeon Processors
8 Cores
64 GB of memory
2 TB of disk

Xeon Phi support:
icc, icpc, libraries, drivers

NVIDIA Titan Black
PCIe Bus
15 SMs
2880 CUDA cores
6 GB of memory
OpenGL support
OpenCL support

31S1P Xeon Phi system
PCIe Bus
“mic0”
57 Cores
22 nm
8 GB of memory
No disk
Application support

1 core for Linux +
56 cores * 4 hyperthreads/core
= 224 hyperthreads for you

mjb – March 30, 2015
Xeon Phi Internals

Each Xeon Phi core has:

- Instruction Decode
- 32K L1 Instruction Cache
- Scalar Unit
- 32K L1 Data Cache
- Scalar Registers
- Vector Unit
- Vector Registers
- 512K L2 Cache / Core
- Ring Bus

The Xeon Phi chip contains almost 5B transistors!
Cache is 8-way set associative. Cache lines are 64 bytes.
Assembly instructions are executed in-order. Thus, threading is important!

Vector registers are 512 bits wide = 16 floats. They can perform Fused Multiply-Add (FMA).
Theoretical performance = almost 1 TFLOPS
Xeon Phi Peak Performance

Clock freq x # cores x # vector lanes x 2 FMA / 2 cycles to decode =

1.091 GHz x 56 x 16 x 2 / 2 =

0.98 TFLOPS

FMA stands for “Fused Multiply+Add”.

It allows the operation:

\[ d = a*b + c; \]

to be performed in the same amount of time as:

\[ d = a*b; \]
Getting to *rabbit* and setting up your account

To login to *rabbit*:

```bash
ssh rabbit.engr.oregonstate.edu -l yourengrusername
```

Put this in your *rabbit* account’s `.cshrc`:

```bash
setenv INTEL_LICENSE_FILE 28518@linlic.engr.oregonstate.edu
setenv ICCPATH /nfs/guille/a2/rh80apps/intel/studio.2013-sp1/composer_xe_2015/bin/
set path=( $path $ICCPATH )
source /nfs/guille/a2/rh80apps/intel/studio.2013-sp1/bin/iccvars.csh intel64
```

Then activate these values like this:

```bash
source .cshrc
```

(These will be activated automatically the next time you login.)

To verify that the Xeon Phi card is there:

```bash
ping mic0
```

To see the Xeon Phi card characteristics:

```bash
micinfo
```

To run some operational tests on the Xeon Phi:

```bash
miccheck
```
Running ping

rabbit 150% ping mic0
PING rabbit-mic0.engr.oregonstate.edu (172.31.1.1) 56(84) bytes of data.
64 bytes from rabbit-mic0.engr.oregonstate.edu (172.31.1.1): icmp_seq=1 ttl=64 time=290 ms
64 bytes from rabbit-mic0.engr.oregonstate.edu (172.31.1.1): icmp_seq=2 ttl=64 time=0.385 ms
64 bytes from rabbit-mic0.engr.oregonstate.edu (172.31.1.1): icmp_seq=3 ttl=64 time=0.242 ms
64 bytes from rabbit-mic0.engr.oregonstate.edu (172.31.1.1): icmp_seq=4 ttl=64 time=0.230 ms
64 bytes from rabbit-mic0.engr.oregonstate.edu (172.31.1.1): icmp_seq=5 ttl=64 time=0.225 ms
64 bytes from rabbit-mic0.engr.oregonstate.edu (172.31.1.1): icmp_seq=6 ttl=64 time=0.261 ms
Running *micinfo*

**MicInfo Utility Log**
Created Mon Jan 12 10:21:07 2015

### System Info
- **HOST OS**: Linux
- **OS Version**: 2.6.32-504.3.3.el6.x86_64
- **Driver Version**: 3.4.2-1
- **MPSS Version**: 3.4.2
- **Host Physical Memory**: 65859 MB

### Device Info
Device No: 0, Device Name: mic0

### Version
- **Flash Version**: 2.1.02.0390
- **SMC Firmware Version**: 1.16.5078
- **SMC Boot Loader Version**: 1.8.4326
- **uOS Version**: 2.6.38.8+mpss3.4.2
- **Device Serial Number**: ADKC31600731

### Board
- **Vendor ID**: 0x8086
- **Device ID**: 0x225e
- **Subsystem ID**: 0x2500
- **Coprocessor Stepping ID**: 3
- **PCIe Width**: Insufficient Privileges
- **PCIe Speed**: Insufficient Privileges
- **PCIe Max payload size**: Insufficient Privileges
- **PCIe Max read req size**: Insufficient Privileges

### Coprocessor Info
- **Model**: 0x01
- **Model Ext**: 0x00
- **Type**: 0x00
- **Family**: 0x0b
- **Family Ext**: 0x00
- **Stepping**: B1
- **Board SKU**: B1PRQ-31S1P
- **ECC Mode**: Enabled
- **SMC HW Revision**: Product 300W Passive CS

### Cores
- **Total No of Active Cores**: 57
- **Voltage**: 1089000 uV
- **Frequency**: 1100000 kHz

### Thermal
- **Fan Speed Control**: N/A
- **Fan RPM**: N/A
- **Fan PWM**: N/A
- **Die Temp**: 40 C

### GDDR
- **Vendor**: Elpida
- **Version**: 0x1
- **Density**: 2048 Mb
- **Size**: 7936 MB
- **Technology**: GDDR5
- **Speed**: 5.000000 GT/s
- **Percentage**: 2500000 kHz
- **Voltage**: 1501000 uV
Running *miccheck*

```
rabbit 152% miccheck

MicCheck 3.4.2-r1
Copyright 2013 Intel Corporation All Rights Reserved

Executing default tests for host
  Test 0: Check number of devices the OS sees in the system ... pass
  Test 1: Check mic driver is loaded ... pass
  Test 2: Check number of devices driver sees in the system ... pass
  Test 3: Check mpssd daemon is running ... Pass

Executing default tests for device: 0
  Test 4 (mic0): Check device is in online state and its postcode is FF ... pass
  Test 5 (mic0): Check ras daemon is available in device ... pass
  Test 6 (mic0): Check running flash version is correct ... pass
  Test 7 (mic0): Check running SMC firmware version is correct ... pass

Status: OK
```
Running micsmc, 1

rabbit 153% micsmc -a

mic0 (info):
Device Series: ........... Intel(R) Xeon Phi(TM) coprocessor x100 family
Device ID: .............. 0x225e
Number of Cores: ....... 57
OS Version: ............. 2.6.38.8+mpss3.4.2
Flash Version: .......... 2.1.02.0390
Driver Version: ......... 3.4.2-1 (root@rabbit.engr.oregonstate.edu)
Stepping: ............... 0x3
Substepping: ............ 0x0

mic0 (temp):
Cpu Temp: ............... 44.00 C
Memory Temp: ............ 28.00 C
Fan-In Temp: ............ 24.00 C
Fan-Out Temp: ........... 28.00 C
Core Rail Temp: .......... 29.00 C
Uncore Rail Temp: ....... 29.00 C
Memory Rail Temp: ....... 29.00 C

mic0 (freq):
Core Frequency: ......... 1.10 GHz
Total Power: ............. 92.00 Watts
Low Power Limit: ........ 283.00 Watts
High Power Limit: ....... 337.00 Watts
Physical Power Limit: .... 357.00 Watts

mic0 (mem):
Free Memory: ............ 7347.64 MB
Total Memory: ........... 7698.83 MB
Memory Usage: .......... 351.18 MB
Running *micsmc, II*

<table>
<thead>
<tr>
<th>Core</th>
<th>User</th>
<th>System</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>mic0 (cores):</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device Utilization: User: 0.00%, System: 0.09%, Idle: 99.91%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per Core Utilization (57 cores in use)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #1:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #2:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #3:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #4:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #5:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #6:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #7:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #8:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #9:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #10:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #50:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #52:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #53:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #54:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #55:</td>
<td>User: 0.00%, System: 0.00%, Idle: 100.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #56:</td>
<td>User: 0.00%, System: 0.27%, Idle: 99.73%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core #57:</td>
<td>User: 0.00%, System: 0.54%, Idle: 99.46%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Oregon State University
Computer Graphics

mjb – March 30, 2015
Cross-compiling and running from *rabbit*

To compile on *rabbit* for *rabbit*:

```bash
icpc -o try try.cpp -O3 -lm -openmp -align -qopt-report=3 -qopt-report-phase=vec
```

To cross-compile on *rabbit* for the Xeon Phi:

```bash
icpc -mmic -o try try.cpp -O3 -lm -openmp -align -qopt-report=3 -qopt-report-phase=vec
```

Note: the summary of vectorization success or failure is in a *.optvec* file

To execute on the Xeon Phi, type this on *rabbit*:

```bash
micnativeloadex try
```
Gaining Access to the Cores, I

```c
#pragma omp parallel for
for( int i = 0; i < N; i++ )
    C[i] = A[i] * B[i];

float sum = 0.;
#pragma omp parallel for reduction(+:sum)
for( int i = 0; i < N; i++ )
    sum += A[i] * B[i];
```

```
icpc -mmic -o try try.cpp -O3 -m-openmp -align -qopt-report=3 -qopt-report-phase=vec
micnativeloadex try
```
Gaining Access to the Cores, II

```c
#pragma omp parallel sections
#pragma omp section
  ...  
#pragma omp section
  ...  
#pragma omp task
  ...  
```

```c
icpc -mmic -o try try.cpp -O3 -m-openmp -align -qopt-report=3 -qopt-report-phase=vec
micnativeloadex try
```
Gaining Access to the Vector Units

```c
#pragma omp simd
for( int i = 0; i < N; i++ )

#pragma omp parallel for simd
for( int i = 0; i < N; i++ )

C[0:N] = A[0:N] * B[0:N] ;
```

```bash
icpc -mmic -o try try.cpp -O3 -m -openmp -align -qopt-report=3 -qopt-report-phase=vec
micnativeloadex try
```
Turning Off All Vectorization

icpc -mmic -o try try.cpp -O3 -lm -openmp -no-vec

The only reason I can think of to do this is when running benchmarks to compare vector vs. scalar array processing.

The Intel compiler does a great job of automatically vectorizing where it can. **Warning:** just because you didn’t deliberately vectorize your code doesn’t mean it didn’t end up vectorized! Use the “-no-vec” flag instead.
Vectorizing Conditionals

```c
#pragma omp simd
for( int i = 0; i < N; i++ )
{
    if( D[i] == 0 )
    else
}
```

In my tests, this was 3-4x as fast as this.

```c
#pragma omp simd
for( int i = 0; i < N; i++ )
{
}
```
Reducing a Vector

```c
float f = __sec_reduce_add( A[0:N] );
float f = __sec_reduce_mul( A[0:N] );
float f = __sec_reduce_max( A[0:N] );
float f = __sec_reduce_min( A[0:N] );

int i = __sec_reduce_max_ind( A[0:N] );
int i = __sec_reduce_min_ind( A[0:N] );

boolean b = __sec_reduce_all_zero( A[0:N] );
boolean b = __sec_reduce_all_nonzero( A[0:N] );
boolean b = __sec_reduce_any_zero( A[0:N] );
boolean b = __sec_reduce_any_nonzero( A[0:N] );
```

You must specify the array length. An argument of `A[:]` will throw a compiler error.
Reducing a Vector

```c
float sum = 0.;
for( int i = 0; i < n; i++ )
{
    sum += A[i];
}
```

In my tests, this was the same speed as this.

```c
float sum = __sec_reduce_add( A[0:N] );
```
Elemental Vector Functions

```
for( int i = 0; i < N; i++ )
{
    C[i] = A[i] * B[i];
}
```

In my tests, this was 3x as fast as this.

```
__declspec(vector)
float vmul( float x, float y )
{
    return x*y;
}
```

```
for( int i = 0; i < N; i++ )
{
    C[i] = vmul( A[i], B[i] );
}
```
Offload Mode

float *A = new float[NUMS];
float *B = new float[NUMS];
float *C = new float[NUMS];

double *dtp = new double;
double Time0 = omp_get_wtime();

#pragma offload target(mic) in(A:length(NUMS)) in(B:length(NUMS)) out(C:length(NUMS)) out(dtp:length(1))
{
    omp_set_num_threads(NUMT);
    double time0 = omp_get_wtime();

    #pragma omp parallel for simd
    for(int i = 0; i < NUMS; i++)
        C[i] = A[i] * B[i];

    double time1 = omp_get_wtime();
    *dtp = time1 - time0;
}

double Time1 = omp_get_wtime();

double overalldt = Time1 - Time0;
double offloaddt = *dtp;

printf(stderr, "%6d\t%6d\t%8.5f\t%8.5f\t%8.4f\t%8.2f\n", NUMT, NUMS, overalldt, offloaddt, 100.*offloaddt/overalldt, ((double)NUMS)/offloaddt/1000000.);
Offload Mode

You don't need to do anything special with the compile line:

```
icpc  -o try try.cpp  -O3  -lm -openmp -align -qopt-report=3 -qopt-report-phase=vec
./try
```
Offload Mode: Persistence Between Offloads

#define ALLOC alloc_if(1)
#define REUSE alloc_if(0)
#define RETAIN free_if(0)
#define FREE free_if(1)

#pragma offload target(mic) in(A:length(NUMS), ALLOC, RETAIN) out(C:length(NUMS), ALLOC, FREE )
{
   ...
}

...

#pragma offload target(mic) in(A:length(NUMS), REUSE, RETAIN) out(D:length(NUMS), ALLOC, RETAIN)
{
   ...
}

...

#pragma offload target(mic) in(A:length(NUMS), REUSE, FREE) out(D:length(NUMS), REUSE, FREE )
{
   ...
}

--------------------
Alignment

To ensure alignment, replace this

float  Temperature[NUMN];

with this:

#define ALIGN64    __declspec(align(64))

    . . .

ALIGN64  float  Temperature[NUMN];
Alignment

To ensure alignment, replace this

```c
float *A = (float *) malloc( NUMS*sizeof(float) );
float *B = (float *) malloc( NUMS*sizeof(float) );
float *C = (float *) malloc( NUMS*sizeof(float) );
```

with this

```c
float *A = (float *) _mm_malloc( NUMS*sizeof(float), 64 );
float *B = (float *) _mm_malloc( NUMS*sizeof(float), 64 );
float *C = (float *) _mm_malloc( NUMS*sizeof(float), 64 );
```

You then free memory with:

```c
_mm_free( A );
_mm_free( B );
_mm_free( C );
```
Alignment

If you want to ensure alignment, but still want to use C++’s `new` and `delete`, replace this:

```cpp
float *A = new float [NUMS];
float *B = new float [NUMS];
float *C = new float [NUMS];
```

with this

```cpp
float *pa = (float *) _mm_malloc( NUMS*sizeof(float), 64 );
float *pb = (float *) _mm_malloc( NUMS*sizeof(float), 64 );
float *pc = (float *) _mm_malloc( NUMS*sizeof(float), 64 );
float *A = new(pa) float [NUMS];
float *B = new(pb) float [NUMS];
float *C = new(pc) float [NUMS];
```

You then free memory with:

```cpp
delete [ ] A;
delete [ ] B;
delete [ ] C;
```

An advantage of using `new` and `delete` instead of `malloc` is that they allow you to use C++ constructors and destructors.
As You Create More and More Threads, On What Cores Do They End Up?

If you want them spread out onto as many cores as possible, execute this:

```c
kmp_set_defaults( "KMP_AFFINITY=scatter" );
```

If you want them packed onto the first core until it has 4, then onto the second core until it has 4, etc., execute this:

```c
kmp_set_defaults( "KMP_AFFINITY=compact" );
```

Use the scatter-mode if you want as much core-power applied to each thread as possible.

Use the compact-mode if there is an advantage to some threads sharing a core’s local memory with other threads.
Running the Volume-Integration Program
Running the Volume-Integration Program

Multicore, no vectorization
Reservation System

https://secure.engr.oregonstate.edu/engr/resources/bailey

![Reservation System](https://secure.engr.oregonstate.edu/engr/resources/bailey)

**Thursday 11 December 2014**

<table>
<thead>
<tr>
<th>Time</th>
<th>William Bailey</th>
</tr>
</thead>
<tbody>
<tr>
<td>07:00am</td>
<td></td>
</tr>
<tr>
<td>07:30am</td>
<td></td>
</tr>
<tr>
<td>08:00am</td>
<td></td>
</tr>
<tr>
<td>08:30am</td>
<td></td>
</tr>
<tr>
<td>09:00am</td>
<td></td>
</tr>
<tr>
<td>09:30am</td>
<td></td>
</tr>
<tr>
<td>10:00am</td>
<td></td>
</tr>
<tr>
<td>10:30am</td>
<td></td>
</tr>
<tr>
<td>11:00am</td>
<td></td>
</tr>
<tr>
<td>11:30am</td>
<td></td>
</tr>
<tr>
<td>12:00pm</td>
<td></td>
</tr>
<tr>
<td>12:30pm</td>
<td></td>
</tr>
<tr>
<td>01:00pm</td>
<td></td>
</tr>
<tr>
<td>01:30pm</td>
<td></td>
</tr>
<tr>
<td>02:00pm</td>
<td></td>
</tr>
<tr>
<td>02:30pm</td>
<td></td>
</tr>
</tbody>
</table>