rabbit.engr.oregonstate.edu

Mike Bailey

mjb@cs.oregonstate.edu

Oregon State University

What is *rabbit*?

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

PCIe Bus

NVIDIA Titan Black

PCIe Bus

15 SMs
2880 CUDA cores
6 GB of memory

OpenGL support
OpenCL support

PCIe Bus

31S1P Xeon Phi system

PCIe Bus

“mic0”
57 Cores
22 nm
8 GB of memory
No disk

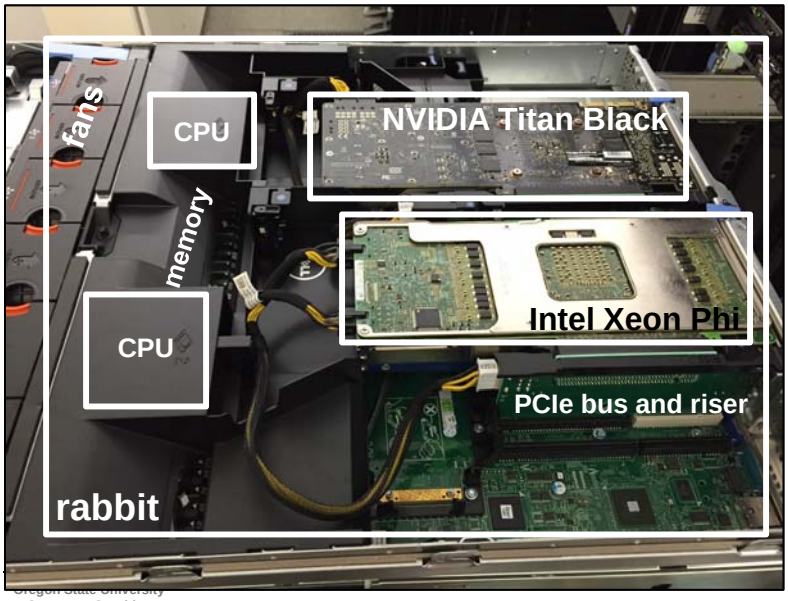
Application support

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

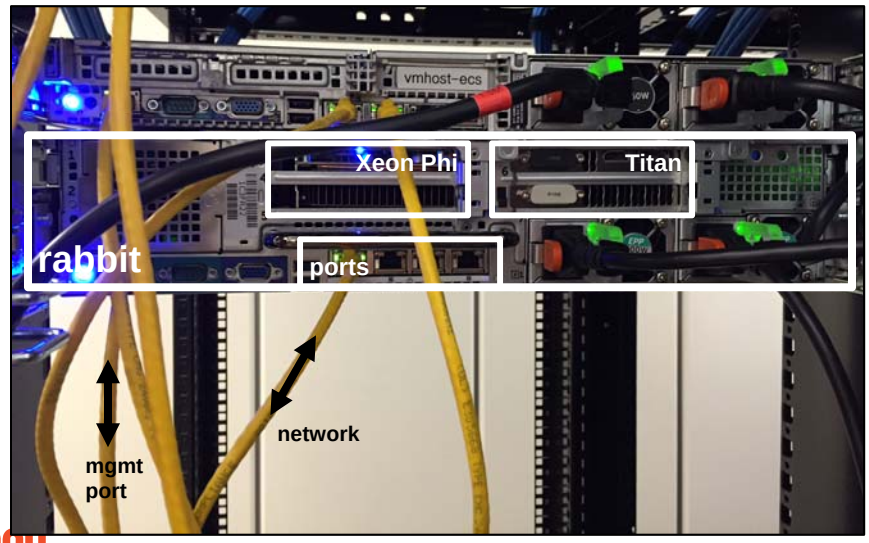
What is *rabbit*?

rabbit lives in a rack in our server room in the Kelley Engineering Center:

What is *rabbit*?



What is *rabbit*?



Getting to *rabbit* and setting up your account

Lowercase letter 'L'

To login to *rabbit*:

ssh rabbit.engr.oregonstate.edu –l yourengrusername

Put this in your *rabbit* account's .cshrc :

```
setenv INTEL_LICENSE_FILE    28518@linlic.engr.oregonstate.edu
setenv SINK_LD_LIBRARY_PATH  /nfs/guille/a2/rh80apps/intel/studio.2013-sp1/composer_xe_2015.0.090/compiler/lib/mic/
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:

source .cshrc

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

To verify that the Xeon Phi card is there:

ping mic0

To see the Xeon Phi card characteristics:

micinfo

To run some operational tests on the Xeon Phi:

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*

```
rabbit 151% 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 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 Model       : 0x01
        Coprocessor Model Ext   : 0x00
        Coprocessor Type        : 0x00
        Coprocessor Family      : 0x0b
        Coprocessor Family Ext  : 0x00
        Coprocessor 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
        GDDR Vendor             : Elpida
        GDDR Version            : 0x1
        GDDR Density            : 2048 Mb
        GDDR Size               : 7936 MB
        GDDR Technology         : GDDR5
        GDDR Speed              : 5.000000 GT/s
        GDDR Frequency          : 2500000 kHz
        GDDR 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, I*

```
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*

```
mic0 (cores):
Device Utilization: User:   0.00%,   System:   0.09%,   Idle:  99.91%
Per Core Utilization (57 cores in use)
Core #1:   User:   0.00%,   System:   0.27%,   Idle:  99.73%
Core #2:   User:   0.00%,   System:   0.27%,   Idle:  99.73%
Core #3:   User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #4:   User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #5:   User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #6:   User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #7:   User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #8:   User:   0.00%,   System:   0.27%,   Idle:  99.73%
Core #9:   User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #10:  User:   0.00%,   System:   0.27%,   Idle:  99.73%

                  . . .

Core #50:  User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #52:  User:   0.00%,   System:   0.27%,   Idle:  99.73%
Core #53:  User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #54:  User:   0.00%,   System:   0.27%,   Idle:  99.73%
Core #55:  User:   0.00%,   System:   0.00%,   Idle: 100.00%
Core #56:  User:   0.00%,   System:   0.27%,   Idle:  99.73%
Core #57:  User:   0.00%,   System:   0.54%,   Idle:  99.46%
```


Cross-compiling and running from *rabbit*

To compile on *rabbit* for *rabbit*:

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:

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*:

micnativeloadex try

To cross-compile on *rabbit* for the Xeon Phi, deliberately disabling vectorization:

icpc -mmic -o try try.cpp -O3 -lm –openmp –no-vec -align -qopt-report=3 -qopt-report-phase=vec

Gaining Access to the Cores, I

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

```
float sum = 0.;
#pragma omp parallel for reduce(+: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

Oregon State University
Computer Graphics

mjb – February 24, 2015

Gaining Access to the Cores, II

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

```
#pragma omp task
        . . .
```

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

micnativeloadex try

Oregon State University
Computer Graphics

mjb – February 24, 2015

Gaining Access to the Vector Units

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

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

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

```
icpc -mmic -o try try.cpp -O3 -lm -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

micnativeloadex try
```

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 when 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.

Compiling for OpenCL

```
printinfo:      printinfo.cpp
                icpc  -o  printinfo   printinfo.cpp  -O3   /usr/lib64/libOpenCL.so   -lm   -openmp
```


The *printinfo* Program Output

```
Number of Platforms = 1
Platform #0:
        Name    = 'NVIDIA CUDA'
        Vendor  = 'NVIDIA Corporation'
        Version = 'OpenCL 1.1 CUDA 7.0.18'
        Profile = 'FULL_PROFILE'
        Device #0:
                Type = 0x0004 = CL_DEVICE_TYPE_GPU
                Device Vendor ID = 0x10de (NVIDIA)
                Device Maximum Compute Units = 15
                Device Maximum Work Item Dimensions = 3
                Device Maximum Work Item Sizes = 1024 x 1024 x 64
                Device Maximum Work Group Size = 1024
                Device Maximum Clock Frequency = 1071 MHz

Device Extensions:
cl_khr_byte_addressable_store
cl_khr_icd
cl_khr_gl_sharing
cl_nv_compiler_options
cl_nv_device_attribute_query
cl_nv_pragma_unroll
cl_nv_copy_opts

cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_fp64
```

15*192 = 2880 CUDA cores!

Compiling for OpenGL

```
sample:     sample.cpp
            icpc -o sample sample.cpp /usr/lib64/libglut.so.3.9.0 -lGLU -lm -openmp
```

Warning: This compiles just fine, but I am still working on how to actually run the program remotely.

Reservation System

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