Contents

• Processor Architecture
• Cluster Specification
• Connection
• Compilation & Run
  – Simple app
  – MKL
  – Pure MPI
  – Hybrid
• Slurm
Introducing Intel® Xeon Phi Coprocessors

- Up to 61 IA cores/1.1 GHz/ 244 Threads
- Up to 8GB memory with up to 352 GB/s bandwidth
- 512-bit SIMD instructions
- Linux operating system, IP addressable
- Standard programming languages and tools
- Over 1 TeraFlop/s double precision peak performance
- 60(61) cores
- 4 HW threads/core
- 8 memory controllers 16 Channel GDDR5
- High-speed bi-directional ring interconnect Fully Coherent L2 Cache
### RSC PETASTREAM (Standard configuration)

<table>
<thead>
<tr>
<th>Type</th>
<th>Massively Parallel Supercomputer [RSC PetaStream]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Intel Many Integrated Core (MIC)</td>
</tr>
<tr>
<td>Performance</td>
<td>1,2 PFlops per rack (peak)</td>
</tr>
<tr>
<td>Compute resources</td>
<td>1024x Intel® Xeon Phi™ 7120D based nodes (total resources: 61.5K cores, 246K threads)</td>
</tr>
<tr>
<td>Memory</td>
<td>16 TB GDDR5</td>
</tr>
<tr>
<td>Interconnect</td>
<td>Infiniband FDR 56Gbps / Intel Truescale Infiniband / 10 GigE</td>
</tr>
<tr>
<td>Local Data Storage</td>
<td>Up to 640x Intel® DC S3500/S3700 SSDs with total capacity up to 0.5PB</td>
</tr>
<tr>
<td>Operating System</td>
<td>Linux (RedHat, SUSE, Debian, CentOS, Scientific Linux), MS Windows</td>
</tr>
<tr>
<td>Job management</td>
<td>SLURM, PBS Pro, Moab, Platform LSF</td>
</tr>
<tr>
<td>Libraries, Compilers and Tools</td>
<td>Intel® Cluster Studio XE 2015</td>
</tr>
<tr>
<td>Form factor</td>
<td>Dual side access rack</td>
</tr>
<tr>
<td>Cooling</td>
<td>RSC direct liquid cooling system: up to 400kW per Cabinet;</td>
</tr>
<tr>
<td>Specification</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>Performance</td>
<td>0.291 PFlops per rack (peak) / 0.165 PFlops (LINPACK)</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel® Xeon Phi™ 5120D based nodes</td>
</tr>
<tr>
<td>Node/Processor/</td>
<td>288/288/17280/69120</td>
</tr>
<tr>
<td>Core/Thread</td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>2.3 TB GDDR5</td>
</tr>
<tr>
<td>Interconnect</td>
<td>Infiniband FDR 56Gbps</td>
</tr>
<tr>
<td>Total Area</td>
<td>1 m²</td>
</tr>
</tbody>
</table>
Connection

- ssh user@hpc.spbstu.ru

Type your login and password.

:-)

2. Run putty.exe

3. Enter the ip address **hpc.spbstu.ru**, port **22** and connection name.

4. Click “Save” button

5. Click “Open” button and run ssh session

6. Type your login and password in terminal
WinSCP - edit and transfer files

• Download WinSCP from home page http://winscp.net/eng/download.php

• Add new connection Connection -> New connection.... :
  – Use host name hpc.spbstu.ru
  – Use protocol SFTP
  – Use port 22
  – Use your own user name and password

• Connect to your new connection, and input your password if required.
WinSCP - edit and transfer files

After a successful connection to one of the panels appear user's home directory on a cluster. You can copy files between any directory on your machine, and home directory on a cluster.

Any file in the home directory can be edited by F4 command, be sure that favorite editor selected.
Resource allocation

- Allocate Xeon Phi
  
  ```
  # salloc -N 1 --partition=mic
  salloc: Granted job allocation 83350
  ```

- Check allocation
  
  ```
  # squeue
  JOBID PARTITION  NAME    USER  ST  TIME NODES NODELIST (REASON)
  5094  mic     bash  user2  R   0:03  1 host-mic0
  ```

- Run MPI program from host
  
  ```
  # mpiexec.hydra -n 64 ./myapp.MIC
  ```

- Connect to mic
  
  ```
  # ssh 'host'-mic0
  # 'run some app'
  # exit
  ```

- Cancel resources allocation
  
  ```
  # scancel 83350
  ```
Compilation & Run: simple application

• Build app on Host
  #cd ./Workshop2016/src/
  #icc -mmic -openmp omp_native.cpp

• Connect to allocated mic
  #ssh 'hostname-mic0'

• Select directory with binaries
  #cd ./Workshop2016/src

• Run app
  #./a.out 2048 64
  #export OMP_NUM_THREADS=30
  #./a.out 2048 64
  #unset OMP_NUM_THREADS
  #exit
Intel® Math Kernel Library (Intel® MKL) accelerates math processing routines that increase application performance and reduce development time. Intel® MKL includes highly vectorized and threaded Linear Algebra, Fast Fourier Transforms (FFT), Vector Math and Statistics functions.

MKL Mathematical Functionality

– Basic Linear Algebra BLAS, Sparse BLAS, PBLAS
– Linear solvers LAPACK, ScaLAPACK1, Sparse Solver routines
– Vector Mathematical & Statistical functions VML, VSL
– Fourier Transform functions
Compilation & Run: MKL

• **Build app on Host**
  
  #icc -mmic openmp mkl mCarlo.cpp

• **Connect to allocated mic**

  #ssh 'hostname-mic0'

• **Run app**

  #cd ./Workshop2016/src
  
 =./a.out
  
  #...
  
  #exit
MKL: Computes a matrix-matrix product with general matrices

- Change previous `omp_naitive.cpp` for MKL usage
  - `# cp omp_naitive.cpp mkl_naitive.cpp`
  - Open `mkl_naitive.cpp` in editor

- See `cblas_dgemm`

- Use one of the following techniques to change the number of threads to use in the Intel® Math Kernel Library (Intel® MKL):
  - Set one of the OpenMP or Intel MKL environment variables: `OMP_NUM_THREADS`, `MKL_NUM_THREADS`, `MKL_DOMAIN_NUM_THREADS`
  - Call one of the OpenMP or Intel MKL functions: `omp_set_num_threads()`, `mkl_set_num_threads()`, `mkl_domain_set_num_threads()`

- Make table with results `omp_naitive.cpp` vs `mkl_naitive.cpp`
Vectorization and SIMD Execution

- **SIMD**
  - Flynn’s Taxonomy: Single Instruction, Multiple Data
  - CPU perform the same operation on multiple data elements
- **SISD**
  - Single Instruction, Single Data
- **Vectorization**
  - In the context of Intel® Architecture Processors, the process of transforming a scalar operation (SISD), that acts on a single data element to the vector operation that act on multiple data elements at once (SIMD).
  - Assuming that setup code does not tip the balance, this can result in more compact and efficient generated code
  - For loops in “normal” or “unvectorized” code, each assembly instruction deals with the data from only the single loop iteration
SIMD Abstraction

for (i = 0; i < 15; i++)
  if (v5[i] < v6[i])
    v1[i] += v3[i];

- SIMD can simplify your code and reduce the jumps, breaks in program flow control
- Note the lack of jumps or conditional code branches

```c
for( i = 0; i < 15; i++)
  if(v5[i] < v6[i])
    v1[i] += v3[i];
```

| v5  | 0 4 7 8 3 9 2 0 6 3 8 9 4 5 0 1 |
| v6  | 9 4 8 2 0 9 4 5 5 3 4 6 9 1 3 0 |
| vcmpi_lt k7, v5, v6 |
| k7  | 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0 |
| v3  | 5 6 7 8 5 6 7 8 5 6 7 8 5 6 7 8 |
| v1  | 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 |
| vaddpi v1{k7}, v1, v3 |
| v1  | 6 1 8 1 1 1 8 9 1 1 1 1 6 1 8 1 |
Compile & Optimize: SIMD

• **Build app on Host**
  
  #icc -mmic -vec-report3 serial.cpp main.cpp
  
  Did any of the loops vectorize? Why not?

• **Look only at the “ALTERNATIVE” suggestion**
  
  # icc -mmic -guide-vec serial.cp

  • **Add `restrict` keyword**
    
    # icc -mmic -restrict -vec-report3 restrict.cpp
    
    # cat serial.optrpt
    
    # icc -mmic -restrict -opt-report restrict.cpp

  • **Add #pragma ivdep**
    
    # icc -mmic -vec-report3 pragma.cpp
    
    Did the code vectorize? Why?

  • **Add #pragma simd**
    
    # icc -mmic -vec-report3 psimd.cpp
    
    Did the code vectorize? Why?
Intel® MPI Library is a message passing library that implements the Message Passing Interface, version 3.0 (MPI-3.0) specification. Use the library to develop applications that can run on multiple cluster interconnects.

The Intel® MPI Library has the following features:

– Scalability up to 262k processes
– Low overhead, enables analyzing large amounts of data
– MPI tuning utility for accelerating your applications
– Interconnect independence and flexible runtime fabric selection
Compilation & Run: MPI

- **Build app on Host**
  
  ```shell
  # mpiicc -mmic -o test.MIC test.c
  ```

- **Set environment variable** `I_MPI_MIC`
  
  ```shell
  # export I_MPI_MIC=1
  ```

- **Load modules for mpiexec.hydra and mic:**
  
  ```shell
  # module load launcher/mpiexec mic
  ```

- **Run app**
  
  ```shell
  # mpiexec.hydra -n 2 ./test.MIC
  # mpiexec.hydra ./test.MIC
  # mpiexec.hydra -n 60 ./test.MIC
  ```
Run: MPI mapping/pinning

• Set environment variable `I_MPI_DEBUG=4` for explore mapping information
  
  ```
  # export I_MPI_DEBUG=4
  ```

• The environment variable `I_MPI_PIN_DOMAIN` controls the mapping/pinning. It splits the (logical) processors into non-overlapping domains for which this rule applies: "1 MPI process for 1 domain".
  
  ```
  # mpiexec.hydra ... -env I_MPI_PIN_DOMAIN auto
  # mpiexec.hydra ... -env I_MPI_PIN_DOMAIN omp
  # mpiexec.hydra ... -env I_MPI_PIN_DOMAIN <N>
  ```

• Run app
  
  ```
  #mpiexec.hydra -env I_MPI_PIN_DOMAIN auto -n 2 ./test.MIC
  #mpiexec.hydra -env I_MPI_PIN_DOMAIN omp -n 2 ./test.MIC
  #mpiexec.hydra -env I_MPI_PIN_DOMAIN 4 -n 4 ./test.MIC
  #mpiexec.hydra -env I_MPI_PIN_DOMAIN 8 -n 4 ./test.MIC
  ```
Compile & Run: Hybrid MPI/OpenMP (1)

- Output code added to test code
  
  # diff test.c test-openmp.c

- Compile
  
  # mpiicc -openmp -mmic -o test-openmp.MIC test-openmp.c

- Reduce output
  
  # unset I_MPI_DEBUG # to reduce the output for now

- Set path to shared libraries of OpenMP
  
  # export MIC_LD_LIBRARY_PATH=/opt/software/intel/composerxe/lib/mic

- Run
  
  # mpiexec.hydra -n 2 ./test-openmp.MIC
Compile & Run: Hybrid MPI/OpenMP (domains)

- Split logical processors on two domains, size is defined by the formula 
  size=#cpu/#proc

  ```
  # mpiexec.hydra -prepend-rank -env KMP_AFFINITY verbose -env OMP_NUM_THREADS 8 -
  env I_MPI_PIN_DOMAIN auto -n 2 ./test-openmp.MIC 2>&1 | sort
  ```

- Domain size is equal to the OMP_NUM_THREADS environment variable value.

  ```
  # mpiexec.hydra -prepend-rank -env KMP_AFFINITY verbose -env OMP_NUM_THREADS 8 -
  env I_MPI_PIN_DOMAIN omp -n 2 ./test-openmp.MIC 2>&1 | sort
  ```

- The domain size is defined by a positive decimal number <n>

  ```
  #mpiexec.hydra -prepend-rank -env KMP_AFFINITY verbose -env OMP_NUM_THREADS 8 -
  env I_MPI_PIN_DOMAIN 8 -n 2 ./test-openmp.MIC 2>&1 | sort
  ```
### Compile & Run: Hybrid MPI/OpenMP (affinity #1)

- Short description of the **KMP_AFFINITY** and **MIC_KMP_AFFINITY**
  - Set this environment variable to influence thread affinity generally
  - OpenMP programs are affected on CPU and MIC
    ```bash
    #export KMP_AFFINITY=<type>   (for CPU)
    #export MIC_KMP_AFFINITY=<type>  (for MIC)
    ```

<table>
<thead>
<tr>
<th>Type</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>compact</td>
<td>Pack threads close to each other.</td>
</tr>
<tr>
<td>explicit</td>
<td>Use the proclist modifier to pin threads</td>
</tr>
<tr>
<td>none</td>
<td>Does not pin threads.</td>
</tr>
<tr>
<td>scatter</td>
<td>Round-robin threads to cores.</td>
</tr>
<tr>
<td>balanced (Phi only)</td>
<td>Use scatter, but keep OMP thread ids consecutive.</td>
</tr>
</tbody>
</table>
#export MIC_KMP_AFFINITY=<type>

- Imagine a system with 4 cores and 4 hardware threads/core
- Placement of 8 threads is illustrated for the 3 types
- Compact type does not fully utilize all cores; not recommended
Compile & Run: Hybrid MPI/OpenMP (affinity #2)

- Use "scatter", "compact", or "balanced" (Intel® Xeon Phi™ coprocessor specific) to modify the default OpenMP affinity

```bash
mpiexec.hydra -prepend-rank -env KMP_AFFINITY verbose,granularity=thread,scatter -env OMP_NUM_THREADS 8 -env I_MPI_PIN_DOMAIN auto -n 2 ./test-openmp.MIC 2>&1 | sort

$mpiexec.hydra -prepend-rank -env KMP_AFFINITY verbose,granularity=thread,compact -env OMP_NUM_THREADS 8 -env I_MPI_PIN_DOMAIN auto -n 2 ./test-openmp.MIC 2>&1 | sort
```
Run using SLURM Batch mode

**SLURM - Simple Linux Utility for Resource Management**

**sbatch** - submit a batch script to slurm

```bash
sbatch [script.slurm]
```

*script.slurm* – script file name

The batch script may contain options preceded with “#SBATCH”

**squeue** – view job state

```bash
squeue
```

**scancel** [jobId] – cancel job

*jobId* – Id of the job from squeue output
SLURM Batch mode commands

#SBATCH -N 10 or #SBATCH --nodes=10
allocation of 10 nodes

#SBATCH --tasks-per-node=60
Specify the number of tasks to be launched per node

#SBATCH –c 1 or #SBATCH --cpus-per-task=1
Equals to default – one allocates processor per task

#SBATCH –p mic or #SBATCH --partition=mic
Specify partition “mic”

#SBATCH --reservation=reservation_name
Allocate resources for the job from the named reservation “reservation_name”
SLURM Batch mode commands

#SBATCH -J mic or #SBATCH --job-name= mic
Specify a name for the job allocation

#SBATCH -o mic-%j.out or #SBATCH --output mic-%j.out
Instruct Slurm to connect the batch script's standard output directly to the file name specified in the "filename pattern", where the "%j" is replaced by the job ID. By default both standard output and standard error are directed to the same file

#SBATCH -e mic-%j.err or #SBATCH --error mic-%j.out
Instruct Slurm to connect the batch script's standard error directly to the file name specified
#!/bin/bash
#SBATCH --nodes=10
#SBATCH --tasks-per-node=240
#SBATCH --cpus-per-task=1
#SBATCH -p mic
#SBATCH --reservation=???
#SBATCH -J mic
#SBATCH -o mic-%j.out
#SBATCH -e mic-%j.err

if [ -f /etc/profile.d/modules-basis.sh ]; then
    source /etc/profile.d/modules-basis.sh
fi

module purge
module load intel/2016.1.150 parallel/mpi.intel/5.1.2.150
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
cd /path/to/workdir
mpiexec.hydra ./myapp appparams
#!/bin/bash
#SBATCH --nodes=10
#SBATCH --tasks-per-node=28
#SBATCH --cpus-per-task=1
#SBATCH -p tornado
#SBATCH --reservation=[reservation]
#SBATCH -J tornado
#SBATCH -o tornado-%j.out
#SBATCH -e tornado-%j.err

if [ -f /etc/profile.d/modules-basis.sh ]; then
    source /etc/profile.d/modules-basis.sh
fi

module purge
module load intel/2016.1.150 parallel/mpi.intel/5.1.2.150
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

cd /path/to/workdir
mpiexec.hydra ./myapp appparams
Use `squeue` slurm commands to see submitted jobs:

```
wsdemo18@login1:~/work
$ sbatch mic.slurm
Submitted batch job 28375
wsdemo18@login1:~/work
$ squeue

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST (REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>28375</td>
<td>mic</td>
<td>mic</td>
<td>wsdemo18</td>
<td>R</td>
<td>0:02</td>
<td>1</td>
<td>nknc-mic66</td>
</tr>
</tbody>
</table>
```

wsdemo18@login1:~/work
$ squeue

```

<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST (REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

wsdemo18@login1:~/work
SLURM Locate results

```
$ sbatch mic.slurm
Submitted batch job 28395
wsdemo18@login1:~/work

$ squeue

    JOBID PARTITION  NAME      USER ST   TIME   NODES NODELIST(REASON)
    28395  mic            mic wsdemo18 R  0:03   10 nknc-mic[273-282]

wsdemo18@login1:~/work

$ squeue

    JOBID PARTITION  NAME      USER ST   TIME   NODES NODELIST(REASON)

wsdemo18@login1:~/work

$ ls -l

    total 64
    -rw-r--r-- 1 wsdemo18 wsdemo       0 Jun 20 16:21 mic-28395.err
    -rw-r--r-- 1 wsdemo18 wsdemo       490 Jun 20 16:21 mic-28395.out
```
SLURM View results

```
wsdemo18@login1:~/work
$ cat mic-28395.out
Hello world: rank 0 of 10 running on nknc-mic273
Hello world: rank 1 of 10 running on nknc-mic274
Hello world: rank 2 of 10 running on nknc-mic275
Hello world: rank 3 of 10 running on nknc-mic276
Hello world: rank 4 of 10 running on nknc-mic277
Hello world: rank 5 of 10 running on nknc-mic278
Hello world: rank 6 of 10 running on nknc-mic279
Hello world: rank 7 of 10 running on nknc-mic280
Hello world: rank 8 of 10 running on nknc-mic281
Hello world: rank 9 of 10 running on nknc-mic282
wsdemo18@login1:~/work
```
Thank you!

Sergeev Efim

efim.sergeev@singularis-lab.com

Victor Getmanskiy

victor.getmanskiy@singularis-lab.com

https://www.singularis-lab.com/
https://www.linkedin.com/company/singularis-lab-llc
http://habrahabr.ru/company/singularis
http://vk.com/singularis_lab
SINGULARIS LAB
software development, computer vision and robotics