The experience of the HLST on Europe's biggest KNL cluster

Tamás Fehér, Serhiy Mochalskyy, Nils Moschüring
Roman Hatzky
Intel MIC Programming Workshop
LRZ
Marconi – KNL at CINECA, Bologna

- Total number of KNL nodes: 3600
- Partition dedicated to the EUROfusion community: 392 (144 flat / 248 cache mode)
  -> about 1 Pflop/s

Photo: F. Pierantoni
Overview

• Memory Bandwidth benchmarks
• Latency benchmarks
• OpenMP Benchmarks
• Code Performance
• Summary
STREAM and IMB

MEMORY BANDWIDTH
BENCHMARKS
STREAM Memory Bandwidth different architectures

STREAM Memory Bandwidth
flat mode DDR4

![Graph showing STREAM Memory Bandwidth vs. number of threads]

- triad(68) = 90.5 GB/s
STREAM Memory Bandwidth
flat mode MCDRAM

Bandwidth (GB/s)

nthreads

copy
scale
add
triad

triad(68) = 493.8 GB/s
STREAM Memory Bandwidth
flat mode MCDRAM - alignment

Alignment also important for cache mode
STREAM Memory Bandwidth

Array size

![Graph showing STREAM Memory Bandwidth vs. Total array size (MiB or GiB). The graph displays different bandwidths at various array sizes.](image-url)
STREAM Memory Bandwidth

cache versus flat

78 GB total arrays size
STREAM Memory Bandwidth over time

linpack_knl_cache_105080.r064u06s01 at 2017-04-06 23-39-26

flops

bw
STREAM Memory Bandwidth over time

linpack_knl_cache_130796.r064u06s01 at 2017-04-14_16-10-05

flops

bw

Nils Moschüring

Intel MIC Programming Workshop, June 28th, 2017
Intel MPI benchmark - bandwidth

Bandwidth from IMB benchmark (MB/s)

- Markoni KNL intra node (flat:snc2)
- Markoni KNL inter node (flat:snc2)
- Markoni Broadwell intra node
- Markoni Broadwell inter node

Message size (Byte)

0 1 10 100 1000 10000 100000 1000000 10000000
LATENCY BENCHMARKS
Latency

**Broadwell**
- 32k L1 cache ~1.7ns
- 265k L2 cache ~5ns
- 45M L3 cache ~20ns
- DDR ~ 90ns

**KNL**
- 32k L1 cache ~4ns
- 1M L2 cache ~15ns
- MCDRAM ~ 170 ns
- DDR ~ 155 ns
### IMB – Ping Pong Test - Latency

#### Intra node Marconi

<table>
<thead>
<tr>
<th></th>
<th>CPU0</th>
<th>CPU1</th>
</tr>
</thead>
<tbody>
<tr>
<td>node0</td>
<td>0.61</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.09</td>
</tr>
</tbody>
</table>

#### Intra node HELIOS

<table>
<thead>
<tr>
<th></th>
<th>CPU0</th>
<th>CPU1</th>
</tr>
</thead>
<tbody>
<tr>
<td>node0</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.64</td>
</tr>
</tbody>
</table>

#### Knights Landing

<table>
<thead>
<tr>
<th></th>
<th>KNL0</th>
</tr>
</thead>
<tbody>
<tr>
<td>node0</td>
<td></td>
</tr>
</tbody>
</table>

#### Knights Corner

<table>
<thead>
<tr>
<th></th>
<th>KNC0</th>
</tr>
</thead>
<tbody>
<tr>
<td>node0</td>
<td></td>
</tr>
</tbody>
</table>
## IMB – Ping Pong Test - Latency

### Inter node Marconi

<table>
<thead>
<tr>
<th>Broadwell</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>node0</td>
<td>CPU0</td>
<td>1.49</td>
</tr>
<tr>
<td>Latency (μs)</td>
<td></td>
<td>CPU0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>node1</td>
</tr>
</tbody>
</table>

### Inter node HELIOS

<table>
<thead>
<tr>
<th>Sandy Bridge</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>node0</td>
<td>CPU0</td>
<td>1.13</td>
</tr>
<tr>
<td>Latency (μs)</td>
<td></td>
<td>CPU0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>node1</td>
</tr>
</tbody>
</table>

### Knights Landing

| node0           | KNL0     | 3.99     |
| Latency (μs)    |          | KNL0     |
|                 |          | node1    |

### Knights Corner

| node0           | KNC0     | 6.00     |
| Latency (μs)    |          | KNC0     |
|                 |          | node1    |
OPENMP BENCHMARKS
OpenMP overhead

- KNL overhead $\approx 2x$ larger:
  - more threads
  - lower CPU frequency
- Exception: ATOMIC 5x longer, use CRITICAL instead
- Using EPCC OpenMP Microbenchmarks J.M. Bull et. al

![Graph showing overhead time (microsec) vs. number of threads for parallel_for and reduction operations.](image)
OpenMP overhead + hyper-threading

![Graph showing overhead time (microsec) vs. number of threads for different numbers of MPI tasks. The graph includes data points for 1 MPI task, 2 MPI tasks, 4 MPI tasks, and 8 MPI tasks.]
CODE PERFORMANCE
HPCG benchmark

- HPCG: sparse 3D problem with multigrid preconditioned conjugate gradient solver.
- The Intel optimized version of the HPCG benchmark was executed in one node.
Gysela execution time

- Test case: 127 x 256 x 64 x 63 (Nr x Ntheta x Nphi x Nvpar, Nmu=0)
- 1 node, 4 MPI tasks, 8 threads (Broadwell) / 16 threads (KNL)
UTL_TRIDIAG_R

- Solve a tridiagonal system
  - forward elimination
  - back substitution

- Not vectorizable

\[
\begin{bmatrix}
  b_1 & c_1 & 0 \\
  a_2 & b_2 & c_2 \\
  & a_3 & b_3 & \ddots \\
  & & \ddots & \ddots & \ddots \\
  & & & a_n & c_{n-1} & b_n \\
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_{n-1} \\
  x_n \\
\end{bmatrix}
= 
\begin{bmatrix}
  d_1 \\
  d_2 \\
  \vdots \\
  d_{n-1} \\
  d_n \\
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>instruction</th>
<th>instruction</th>
<th>Broadwell latency</th>
<th>Broadwell throughput</th>
<th>KNL latency</th>
<th>KNL throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>3N</td>
<td>FMA</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2N-1</td>
<td>DIVSD</td>
<td>10-14</td>
<td>1/5-1/4</td>
<td>42</td>
<td>1/42</td>
</tr>
<tr>
<td>2N-1</td>
<td>VDIVPD</td>
<td>19-23</td>
<td>1/16</td>
<td>32</td>
<td>1/32</td>
</tr>
</tbody>
</table>


Nils Moschuring

Intel MIC Programming Workshop, June 28th, 2017
Summary

• Good stuff:
  – MPI latency and OpenMP overhead comparable
  – KNL can match Broadwell performance without extensive tuning for most codes
  – Optimization on KNL helps on Broadwell and vice versa

• Bad Stuff:
  – Cache mode operation can be dubious
  – Peak performance hard to reach
  – Hyperthreading rarely useful
Summary

• KNL is equal to Broadwell if your code either
  – Has very good scalability (to make use of increased core count)
  – Has very good vectorization (to make use of more vector units)
  – Effectively uses only 16 GB (to make use of higher bandwidth)

• If more than one holds, you will probably get more performance than on Broadwell

• Memory mode Quadrant seems to be the best