Myths about KNL
Have to use lots of threading on node
Nekbone Hybrid Performance on Eight Nodes

Best performance
256 MPI Tasks/8 Threads
32 MPI Tasks/node

Very Good
OpenMP Implementation with first touch

Product of MPI and OpenMP Threads = 64 on node

Number of threads

Number of MPI Tasks across 8 nodes

32  64  128  256  512  1024  2048
S3D Scaling up to 512 nodes

Colors indicate the number of MPI tasks/node
- Blue: 16
- Red: 32
- Green: 64
- Purple: 128

Good
OpenMP Implementation
Does not use consistent
Parallelization or first
touch

Lower is Better

Product of MPI and
OpenMP
Threads = 64 on node
VH1 scaling up to 512 nodes

Good OpenMP Implementation
Does not use consistent Parallelization or first touch

Lower is Better

Product of MPI and OpenMP Threads = 64 on node

Colors Indicate number of MPI Tasks/Node
- 16
- 32
- 64
- 128
Himeno Scaling up to 512 nodes

Colors indicate the number of MPI Tasks/node

Higher is Better

Product of MPI and OpenMP Threads = 64 on node

Good OpenMP Implementation
Uses consistent Parallelization and first touch
Lots of usable Options for MCDRAM
Only two reasonable uses of MCDRAM

- If code and executable fits within 16 Gbytes use `numctl=1`
  - Numctl preferred is not useful
  - Why do you have all that DDR memory???
- If code and executable is larger than 16 Gbytes use Cache
  - Watch out for Direct Mapped Cache
  - Have not found any example where using MCDRAM as separate memory either all of partial beat using MCDRAM as CACHE
### Important Data for Haswell and KNL

<table>
<thead>
<tr>
<th></th>
<th>Latency Haswell Nanosec (clocks)</th>
<th>KNL Size /core</th>
<th>Latency KNL Nanosec (clocks)</th>
<th>KNL Size /core</th>
<th>Bandwidth Haswell GB/sec Stream Triad</th>
<th>Bandwidth KNL GB/sec Stream Triad</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Cache</td>
<td>2.7 (6)</td>
<td>32KB</td>
<td>2.9 (4)</td>
<td>32KB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2 Cache</td>
<td>8.11 (19)</td>
<td>256KB</td>
<td>13.6 (19)</td>
<td>512KB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCDRAM (Cache)</td>
<td>24.35 (56) (Level 3)</td>
<td>2.5MB</td>
<td>173.5 (243)</td>
<td>242MB</td>
<td>329</td>
<td></td>
</tr>
<tr>
<td>MCDRAM (Flat)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>486</td>
</tr>
<tr>
<td>DDR (Flat)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>102</td>
<td>90</td>
</tr>
<tr>
<td>DDR (Cache)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
</tbody>
</table>
What to do with all those Clustering modes
Cluster – Quad

- Allows a collection of cores within a quadrant to get all of the memory bandwidth
  - Could be important for a bandwidth sensitive, load imbalanced mostly-MPI application.
    - Mostly MPI = greater than/or equal to 16 MPI tasks per node
- Introduces additional NUMA affects when threading across more than one tile
Cluster Mode: Quadrant

Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. SW Transparent.

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return
Conclusions

- KNL is very easy to port to; however, performance can always be improved
- For KNL to beat Xeon you must
  - VECTORIZE
  - Use as many cores as possible – At least 16 MPI tasks – rest threads
    - OpenMP must be good OpenMP
  - Heavily use MCDRAM

- Have seen applications that do all three and can be blocked for MCDRAM-Cache run 3-4 times faster than Xeon
  - These usually cannot be blocked effectively for L3 cache on Xeon