Performance Variability on Xeon Phi

Brandon Cook, Thorsten Kurth, Brian Austin, Samuel Williams, Jack Deslippe

June 22, 2017
Who cares?

• **Application developers**
  – understanding performance
  – reason effectively about optimizations
  – sound advice to application users

• **Users**
  – Efficient use of CPU allocations
  – Wasted cycles on terminated jobs
  – Correct estimates of campaign costs

• **Facilities**
  – System health
  – Advice for users
  – Utilization – scheduler efficiency
Cori at NERSC

- **2388 Haswell**
  - 2x 16 core @ 2.3 GHz
  - 40 MB shared L3
  - 128 GB DDR

- **Cray Aries Interconnect**
  - dragonfly topology

- **9688 Xeon Phi (KNL) nodes**
  - 68 cores @ 1.4 GHz
  - 34 MB distributed L2
  - 96 GB DDR
  - 16 GB MCDRAM (on-package)
Cori at NERSC

- **2388 Haswell**
  - 2x 16 core @ 2.3 GHz
  - 40 MB shared L3
  - 128 GB DDR

- **Cray Aries Interconnect**
  - dragonfly topology

- **9688 Xeon Phi (KNL) nodes**
  - 68 cores @ 1.4 GHz
  - 34 MB distributed L2
  - 96 GB DDR
  - 16 GB MCDRAM (on-package)
KNL is highly configurable

Cluster modes
- all-to-all
- quadrant
- SNC2/4

Memory modes
- flat
- cache
- hybrid
MCDRAM cache mode

- 16GB MCDRAM cache
- single NUMA
- No code modification
- No NUMA programing or affinity issues (e.g. numactl)
- but?

Cluster modes
- all-to-all
- quadrant
- SNC2/4

Memory modes
- flat
- cache
- hybrid
Variability in cache mode

![Graph showing variability in cache mode.](image)

**Notes:**
- MiniFE Total CG GFlops
- ZS=off Iter=1
- ZS=off Iter=2

**Legend:**
- MinFE Total CG GFlops
- ZS=off Iter=1
- ZS=off Iter=2
Brief introduction to caches

KNL’s MCDRAM cache is direct-mapped.

https://en.wikipedia.org/wiki/CPU_cache
Direct-Mapped Caches: Thrashing

```
COMMON //A(8192), B(8192)
DO I=1,N
    PROD = PROD + A(I)*B(I)
ENDDO
```

(Virtual) memory

```
32 KB

A(1)
A(2)

A(8191)
A(8192)
B(1)

B(8191)
B(8192)
```

Direct mapped cache (32 KB)

```
Cache line: 4 words

A(1)  A(2)  A(3)  A(4)
A(5)  A(6)  A(7)  A(8)

A(8185)  A(8186)  A(8187)  A(8188)
A(8189)  A(8190)  A(8191)  A(8192)
```

Registers in the CPU

Thrashing: every memory reference results in a cache miss

Location in the cache:
(memory-address) mod (cache-size)
in this case loc(A(1)) mod 32KB = loc(B(1)) mod 32KB
[because B(1) = A(1) + 8192; 8192*4B mod 32KB = 0]

http://sc.tamu.edu/help/power/powerlearn/html/ScalarOptnw/sld015.htm
Misses depend on free page list

- OS stores a list of free memory pages.
- Allocations are made from the top of the list.
- The free page list gets scrambled if memory is not freed in the order it was allocated.

Consider a 16kB cache...

//Initial page list is sorted
A = malloc(16 kB) //4 pages
B = malloc( 4 kB) //Capacity conflicts
free(A)
free(B) //Scrambled!
A = malloc( 4 kB)
B = malloc( 4 kB) //Conflict!

<table>
<thead>
<tr>
<th>OS View</th>
<th>Application View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free page list</td>
<td>A</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

//Initial page list is sorted
A = malloc(16 kB) //4 pages
B = malloc( 4 kB) //Capacity conflicts
free(A)
free(B) //Scrambled!
A = malloc( 4 kB)
B = malloc( 4 kB) //Conflict!
Solution: sort the free page list*

- zonesort: kernel module provided by Intel
- At NERSC
  - called immediately before application launch
benefit of zonesort for MiniFE

zonesort off

zonesort on
effect of zonesort for HPGMG

High Performance Geometric Multi-Grid

Highly instrumented

Perfectly load balanced problem

“smooth time”

256^3 grid per rank
Job placement
A Xeon Phi issue?
## A Xeon Phi issue?

<table>
<thead>
<tr>
<th></th>
<th>Haswell</th>
<th>Xeon Phi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flops</td>
<td>1.2 TFlops</td>
<td>3 TFlops</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>~100 GB/s</td>
<td>~400 GB/s</td>
</tr>
<tr>
<td>Memory Capacity</td>
<td>128 GB</td>
<td>96 GB</td>
</tr>
<tr>
<td>Capacity / bandwidth</td>
<td>1.28 s</td>
<td>0.24 s</td>
</tr>
</tbody>
</table>

More flops & lower memory capacity / bandwidth + same network = more pressure on network!
Aries topology

~386 nodes per group
sbatch --switches=<count>[@<max-time>]

<count> = # of groups

<max-time> = time to wait for constraint
Impact of # of groups

OSU MPI_Alltoall 32 ranks on 32 nodes

- ~1500 us
- ~3000 us
Job Placement

1 Aries Group

nodes allocated to job
Chroma HMC – 256 nodes

![Graph showing BICGStab GFLOPS/s vs solver sequence number for different jobs.](image)

- **(b) job 3724518**
- **(c) job 3731220**
- **(d) job 3828179**
Summary

• **MCDRAM Cache**
  – direct map cache
  – leads to cache conflicts
  – Intel zonesort

• **Job placement**
  – Bad placement introduces extra hops for data
  – Bad placement increases potential for interference
  – SLURM topology control helps (# of nodes < 350)

• **Not covered in this talk**
  – IO! (burst buffer on compute fabric)
  – Identification of network “Aggressors”
  – frequency scaling (DVFS)