Usability of the Cache Aware Roofline Model on Knight Landings

Nicolas Denoyelle
nicolas.denoyelle@inria.fr

28 mars 2017
Figure – Machine with 2 components
Roofline Model

**Figure** – Roofline Model of an hypothetical system with one memory and one compute unit
Figure – Hypothetical NUMA system with a memory hierarchy and one Core
Cache Aware Roofline Model

**Figure** – Cache Aware Roofline Model of hypothetical NUMA system with a memory hierarchy and one Core.
Several modes, several performances (Cf Ian Masliah Talk), one model.
Several modes, several performances (Cf Ian Masliah Talk), one model.
Several modes, several performances (Cf Ian Masliah Talk), one model.
Several modes, several performances (Cf Ian Masliah Talk), one model.
### KNL benchmarks (64 threads)

#### Bandwidths

<table>
<thead>
<tr>
<th>Obj</th>
<th>Type</th>
<th>GByte.s</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Load</td>
<td>8364</td>
<td>785</td>
</tr>
<tr>
<td>L1</td>
<td>Intel</td>
<td>4030</td>
<td>NA</td>
</tr>
<tr>
<td>L2</td>
<td>Load</td>
<td>2293</td>
<td>370</td>
</tr>
<tr>
<td>L2</td>
<td>Intel</td>
<td>984</td>
<td>NA</td>
</tr>
<tr>
<td>MCDRAM</td>
<td>Load</td>
<td>356</td>
<td>7.61</td>
</tr>
<tr>
<td>MCDRAM</td>
<td>Store(nt)</td>
<td>267</td>
<td>0.40</td>
</tr>
<tr>
<td>MCDRAM</td>
<td>Intel</td>
<td>463.7</td>
<td>NA</td>
</tr>
<tr>
<td>DRAM</td>
<td>Load</td>
<td>86.7</td>
<td>0.29</td>
</tr>
<tr>
<td>DRAM</td>
<td>Store(nt)</td>
<td>50.9</td>
<td>0.04</td>
</tr>
<tr>
<td>DRAM</td>
<td>Intel</td>
<td>83.1</td>
<td>NA</td>
</tr>
</tbody>
</table>

Throughput max $\approx 1.56$

Instructions/cycle (per core)

#### Performance peaks

<table>
<thead>
<tr>
<th>Type</th>
<th>GFlop.s</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>1299</td>
<td>8.75</td>
</tr>
<tr>
<td>Intel ADD</td>
<td>528</td>
<td>NA</td>
</tr>
<tr>
<td>MUL</td>
<td>1299</td>
<td>5.82</td>
</tr>
<tr>
<td>FMA</td>
<td>2597</td>
<td>12.6</td>
</tr>
<tr>
<td>Intel FMA</td>
<td>1055</td>
<td>NA</td>
</tr>
</tbody>
</table>

Throughput max $\approx 1.70$

Instructions/cycle (per core)
Core & VPU

- 2x 64B Load & 1 64B Store ports in Dcache.
TABLE – Per-cluster load bandwidth (GByte/s) matrix of the KNL.
Compile test code with -g

module load compiler/intel/64/2017_update2-knl

source /cm/shared/apps/intel/composer_xe/2017_update2-knl/advisor_2017.1.2.501009/advixe-vars.sh

advixe-gui

2 runs:
- Instrument code and collect functions, loops: Flops, Bytes.
- Run normal code and collect functions, loops: runtime.
Streaming Benchmarks

- ddot $dot += a[i] \times b[i]$ (2LD + 2 FLOPS)
- scale $a[i] = scalar \times b[i]$ (1LD + 1ST + 1 FLOPS)
- triad $c[i] = a[i] + scalar \times b[i]$ (2LD + 1ST + 2 FLOPS)

**KNL_flat_load_STREAM**

**NUMANODE:0**

Bandwidth-bound benchmarks below the roof?!
Streaming Benchmarks

- ddot dot += a[i] * b[i] (2LD + 2 FLOPS)
- scale a[i] = scalar*b[i] (1LD + 1ST + 1 FLOPS)
- triad c[i] = a[i]+scalar*b[i] (2LD + 1ST + 2 FLOPS)

Bandwidth-bound benchmarks below the roof?!
Analytical model

- $s$ the size of $b$ array and $a$ array, respectively stored and loaded in the process,
- $t$ the wall time,
- $B_s$ the memory store bandwidth,
- $B_l$ the memory load bandwidth,

$$t = \frac{s}{B_s} + \frac{s}{B_l}$$  \hspace{1cm} (1)

Actually more bandwidths: load, store, store(nt) for each Cluster, for each DRAM and MCDRAM.

$$t = \sum_{\{(s_k, B_l)\}} \frac{s_i}{B_j}$$  \hspace{1cm} (2)
Check with DRAM, and MCDRAM. We allocate 1GB buffer with a slice in DRAM and another in MCDRAM.
Future Works

- CARM for locality, with automatic building and validation.
- Energy CARM.
- Extended analytical model for memory partitioning.
- What about latency? Only 10% improvement on lulesh proxy-application between DRAM and MCDRAM.
Merci