Non-uniform memory access (NUMA)

• Memory access between processor core to main memory is not uniform.
• Memory resides in separate regions called “NUMA domains.”
• For highest performance, cores should only access memory in its nearest NUMA domain.
Memory buses and controllers

- Pre-2008: Front side bus
- Post-2008: QuickPath interconnect
Front Side Bus (FSB) and Multicore

- FSB was used up to the Core microarch. (2008)
- Memory access by many cores or sockets across the FSB is a bottleneck
Hypertransport & QuickPath Interconnect

• AMD HyperTransport (2003)
  Intel QuickPath Interconnect (2008)
• Memory controller now on the CPU
• Memory access is now non-uniform
UMA = Uniform Memory Access
SMP = Symmetric MultiProcessor
NUMA = Non-Uniform Memory Access
Non-Uniform Memory Access (NUMA)

- NUMA architectures support higher aggregate bandwidth to memory than UMA architectures.
- “Trade-off” is non-uniform memory access.
- Can NUMA effects be observed?
- Locality domain: a set of processor cores and its locally connected memory (a.k.a. locality “node”).

- View the NUMA structure (on Linux):
  
  `numactl --hardware`

  `numactl` can also be used to control NUMA behavior.
NUMA

• How does the OS decide where to put data?
• How does OS decide where to place the threads?

• Ref: Hager and Wellein, Sec 4.2, Chap 8, App A
• Ref: Xeon Phi book, Sec 4.4.5
Page placement by first touch

• A page is placed in the locality region of the processor that first touches it (not when memory is allocated)

• If there is no memory in that locality domain, then another region is used

• Other policies are possible (e.g., set preferred locality domain, or round-robin, using numactl)
numactl – set memory affinity

numactl [--interleave=nodes] [--preferred=node] [--membind=nodes] [--localalloc] <command> [args] ...

numactl --hardware
numactl --show

numactl -interleave=0,1 <command>
Thread affinity

• Threads can be pinned to cores (otherwise they may move to any core)
  – A thread that migrates to a different core usually will no longer have its data in cache (note, some subset of cores may share some levels of cache)
  – A thread that migrates to a different locality region may have its data in a different locality region

• For programs compiled with Intel compilers, specify affinity at runtime using KMP_AFFINITY environment variable
KMP_AFFINITY environment variable

2 sockets, 4 cores/socket

compact

scatter

Might not use all sockets
KMP_AFFINITY environment variable

2 sockets, 4 cores/socket, 2-way hyperthreading

**granularity=core,compact**

- 0,1
- 2,3
- 4,5
- 6,7
- 8,9
- 10,11
- 12,13
- 14,15

**granularity=core,scatter**

- 0,8
- 2,10
- 4,12
- 6,14
- 1,9
- 3,11
- 5,13
- 7,15

Thread can only float between hardware contexts on a core
KMP_AFFINITY environment variable

2 sockets, 4 cores/socket, 2-way hyperthreading

**granularity=fine, compact**

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>
```

**granularity=fine, scatter**

```
<table>
<thead>
<tr>
<th>0</th>
<th>8</th>
<th>2</th>
<th>10</th>
<th>4</th>
<th>12</th>
<th>6</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>13</td>
<td>7</td>
<td>15</td>
</tr>
</tbody>
</table>
```

Usually no need to bind threads to hardware thread contexts
KMP_AFFINITY on Intel Xeon Phi

Examples
KMP_AFFINITY="verbose,none" (print affinity information)
KMP_AFFINITY="balanced"
KMP_AFFINITY="granularity=fine,balanced" (bind to single hardware thread)
KMP_AFFINITY="granularity=core,balanced"
  (bind to any hardware thread on core; core is default granularity)

4 cores and 4 HW threads per core.
Note: “balanced” is only for Intel Xeon Phi.
How to choose different options

• For BW-bound codes, the optimum number of threads is often less than the number of cores. Distribute and choose number of threads to reduce contention and fully utilize the memory controllers. (Use scatter.)

• For compute-bound codes, use hyperthreading and all available hyperthreads. Threads operating on adjacent data should be placed on the same core (or nearby) in order to share cache. (Use compact.)

• On Intel Xeon Phi, “balanced” often works best. (Probably should have this option on CPUs!)

• Example: bench-dgemm on MIC
Intel Xeon Phi

Technically UMA, but access to L2 cache is nonuniform
Intel compiler optimization reports

- icc ... -qopt-report=[0-5]
  icc ... -qopt-report-phase=vec

compiler will output an optimization report in the file xx.optrpt
Backup Figures
Ring Bus to Connect Cores and Caches/DRAM

All current Intel CPUs use a ring bus (alternative is an all-to-all switch)

Intel Xeon Phi
Ring Bus on Haswell
Front Side Bus (FSB) vs. QuickPath Interconnect (QPI)
QuickPath Interconnect

NUMA: Non-Uniform Memory Access