Programming the new KNL Cluster at LRZ
KNL MCDRAM and KNC Offloading

Dr. Volker Weinberg (LRZ)

January, 24-25, 2018, LRZ
KNL Cluster and Memory Modes
MCDRAM

with material from Intel, John Cazes et al. (TACC) and Adrian Jackson (EPCC)
Cores are grouped in pairs (tiles)
- 36 possible tiles
- 2D mesh interconnect
- 2 DDR memory controllers
  - 6 channels DDR4
  - Up to 90 GB/s
- 16 GB MCDRAM
  - Up to 475 GB/s
Programming the new KNL Cluster at LRZ
Tile

- Basic unit for replication
- Each tile consists of 2 cores, 2 vector-processing units (VPU) per core, a 1 MB L2 Cache shared between the 2 cores
- **CHA** (caching/home agent)
  - Serves as the point where the tile connects to the mesh
  - Holds a portion of the distributed tag directory structure
Tiles are connected by a cache-coherent, **2D mesh interconnect**

Provides a more scalable way to connect the tiles by providing higher bandwidth and lower latency compared to KNC **1D ring interconnect**

**MESIF** (Modified, Exclusive, Shared, Invalid, Forward) cache-coherent protocol

Cache lines present in L2 caches are tracked using a **distributed tag directory structure**

Around **700 GB/s total aggregate bandwidth**

Mesh is organized into rows and columns of half rings that fold upon themselves at the endpoints

Mesh enforces a **YX routing rule**

Mesh at fixed frequency of 1.7 GHz

Single hop: X-direction 2 clocks, Y-direction: 1 clock
2 Memory Types
- MCDRAM (16 GB)
- DDR4 (96 GB)

3 Memory Modes
- Cache
- Flat
- Hybrid
  - 25% (4 GB)
  - 50% (8 GB)
  - 75% (12 GB)
KNL Memory: Overview

- **Memory hierarchy on KNL:**
  - DDR4 (96 GB)
  - MCDRAM (16 GB)
  - Tile L2 (1 MB)
  - Core L1 (32 KB)

- **Tile**: set of 2 cores sharing a 1MB L2 cache and connectivity on the mesh

- **Quadrant**: virtual concept, not a hardware property. Way to divide the tiles at a logical level.

- **Tag Directory**: tracks cache line locations in all L2 caches. It provides the block of data or (if not available in L2) a memory address to the memory controller.
KNL Memory: MCDRAM

- **High-bandwidth** memory integrated on-package
- **8 MCDRAM** devices on KNL, each with 2 GB capacity -> **total 16 GB**
- Connected to EDC memory controller via **proprietary on-package I/O: OPIO**
- Each device has a **separate read and write bus** connecting it to its EDC
- **Aggregate Stream Triads Bandwidth** for the 8 MCDRAMS is **over 450 GB/s**
- Slighter higher latency than main memory (~10% slower)
KNL Memory: DDR4

- High-capacity memory off-package
- KNL has direct access to all of main memory
- 2 DDR4 memory controllers on opposite sides of the chip, each controlling 3 DDR4 channels
- Maximum total capacity is 384 GB
- Aggregate Stream Triads Bandwidth from all 6 DDR4 channels is around 90 GB/s
KNL Memory Modes

- **Cache:**
  - MCDRAM serves as cache for transactions to DDR4 memory
  - Direct-mapped memory-side cache with 64-byte cache-lines
  - Inclusive of all modified lines in L2 cache
  - Completely transparent to the user

- **Flat**
  - Flat address space
  - Different NUMA nodes for DDR4 and MCDRAM
  - `numactl` or `memkind` library can be used for allocation

- **Hybrid**
  - 20% / 50% / 75% of MCDRAM set up as cache
  - Potentially useful for some applications
KNL Memory Modes

(a) Cache Mode
- 16GB cache
- MCDRAM 16GB
- 96GB RAM
- DDR4 96GB
- 68 cores

(b) Flat Mode
- DDR4 96GB
- MCDRAM 16GB
- 68 cores
- 112GB RAM

(c) Hybrid Mode
- DDR4 96GB
- MCDRAM
- 68 cores

24.-25.1.2018 Programming the new KNL Cluster at LRZ
## Memory Modes: Comparison

<table>
<thead>
<tr>
<th>Memory Mode</th>
<th>MCDRAM</th>
<th>DDR4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat</td>
<td>100% Memory</td>
<td>100% Memory</td>
</tr>
<tr>
<td>Hybrid 25%</td>
<td>75% Memory 25% Cache</td>
<td>100% Memory</td>
</tr>
<tr>
<td>Hybrid 50%</td>
<td>50% Memory 50% Cache</td>
<td>100% Memory</td>
</tr>
<tr>
<td>Hybrid 75%</td>
<td>25% Memory 75% Cache</td>
<td>100% Memory</td>
</tr>
<tr>
<td>Cache</td>
<td>0% Memory 100% Cache</td>
<td>100% Memory</td>
</tr>
</tbody>
</table>
Cluster Modes

● Cluster Modes modify the distance that L2 coherency traffic flows go through the mesh

● 5 Cluster Modes supported:
  – All-to-all
  – Quadrant / Hemisphere
  – 2 Sub-NUMA Cluster modes: SNC-4 / SNC-2

● Regardless of the cluster mode selected, all memory (all MCDRAM and all DDR4) is available to all cores, and all memory is fully cache-coherent.

● What differs between the modes is whether the view of MCDRAM or DDR is UMA (Uniform Memory Access) or NUMA.
Cluster Modes: Overview

Cluster modes modify the distance that coherency traffic flows through mesh!

(a) All-to-All (no communication localized)
(b) Quadrant (some communication localized)
(c) Sub-NUMA-4 (SNC-4) (all communication localized)
Cluster Modes

- **A quadrant** is a virtual concept and not a hardware property. Divides tiles into 4 groups at a logical level.
- **A tag directory** is used to track L2 cache line locations and status (which tile and if valid).
- Tag directory is distributed over tiles:
  - Each directory component is responsible for an exclusive portion of address space.
  - Directory indicates where cache line is: a certain tile’s L2 cache or in memory.
Cluster Modes

- A tile’s L2 cache can hold any data
- Tag directory tracks if data is in L2 and which tile’s L2 has data
- Tag directory is distributed across all tiles
  - Each tile has an equal portion of the address space
  - Portion of tag directory in a tile not related to L2 cache in that tile
- Every tile has a Caching-Home Agent (CHA)
  - services queries about it’s portion of the tag directory
Cluster-Modes: UMA vs. NUMA

- **Quadrant-Mode:**
  - Each memory type is UMA
  - Latency from any given core to any memory location in the same memory type (MCDRAM or DDR) is essentially the same

- **SNC-4 (2) Mode:**
  - Each memory type is NUMA
  - Cores and memory divided into 4 quadrants (2 halves) with lower latency for near memory accesses (within the same quadrant (half)) and higher latency for far memory accesses (within a different quadrant (half))
Cluster-Modes: NUMA Domains

- **Flat-All2All/Quadrant/Hemisphere Mode:**
  - 1 DDR NUMA Domain
  - 1 MCDRAM NUMA Domain

- **Flat-SNC-2:**
  - 2 DDR NUMA Domain
  - 2 MCDRAM NUMA Domain

- **Flat-SNC-4:**
  - 4 DDR NUMA Domain
  - 4 MCDRAM NUMA Domain

- **Cache Mode:**
  - 1 DDR NUMA Domain
  - 0 MCDRAM NUMA Domain

Memory interleaving (technique to spread out consecutive memory access across multiple memory channels in order to parallelise the accesses) differs among the various modes.
Cache Coherency Protocol

- For memory loads/stores:
  - Core (requestor) looks in local L2 cache
- If not there it queries Distributed Tag Directory (DTD) for it:
  - Sends message to tile (CHA) containing DTD entry for that memory address (tag owner) to check if any other tile on the chip has that address in its caches
- If it’s not in any cache then data fetched from memory
  - DTD updates with requestor information
- If it’s in a tile’s L2 cache then:
  - Tag owner sends message to tile where data is (resident)
  - Resident sends data to requestor
Cluster Modes: All to All

- No affinity between tile and tag directory:
  When there is an L2 miss in a tile the directory tag may be anywhere on the chip
- No affinity between tag directory and memory:
  Data associated to a directory tag may be anywhere on the chip

1. L2 Miss: data not in local L2
2. Directory access: look for tag in directory
3. Memory access: look for data in memory
4. Data: send data to original tile
Cluster Modes: Quadrant

- **No affinity between tile and tag directory:**
  When there is an L2 miss in a tile the directory tag may be anywhere on the chip

- **Affinity between tag directory and memory:**
  Data associated to a directory tag will be in the same quadrant that the directory tag is located

1. L2 Miss: data not in local L2
2. Directory access: look for tag in directory
3. Memory access: look for data in memory
4. Data: send data to original tile
Cluster Modes: SNC-4

1. L2 Miss: data not in local L2
2. Directory access: look for tag in directory
3. Memory access: look for data in memory
4. Data: send data to original tile

- **Affinity between tile and tag directory:**
  When there is an L2 miss in a tile the directory tag will be in the same quadrant

- **Affinity between tag directory and memory:**
  Data associated to a directory tag will be in the same quadrant that the directory tag is located
Cluster Modes: Some more remarks

- **All-to-All Mode:**
  - Most general mode. Lower performance than other modes, ignore
  - Only mode that can be used when DDR DIMMS have not identical capacity

- **Quadrant-Mode:**
  - Lower latency and higher bandwidth than all-to-all.
  - Will always give reasonable performance
  - SW transparent, no special NUMA optimisation
    - 1 NUMA region for MCDRAM
    - 1 NUMA region for DDR
  - Specially well suited for MPI applications with 1 rank per KNL

- **SNC-4:**
  - Each Quadrant exposed as a separate NUMA domain (like 4-Socket Xeon)
  - Well suited for MPI applications with 4 or n*4 ranks per KNL
  - SW needs to NUMA optimise to get benefit
  - Good for NUMA-aware code

24.-25.1.2018
Programming the new KNL Cluster at LRZ
Cluster-Modes & Memory Modes Combinations

● 5 Flat Memory Mode Variants:
  – Flat-A2A
  – Flat-Quadrant
  – Flat-Hemisphere
  – Flat-SNC4
  – Flat-SNC2

● 5 Cache Memory Mode Variants
  – Cache-A2A
  – Cache-Quadrant
  – Cache-Hemisphere
  – Cache-SNC4
  – Cache-SNC2

● 5 x 3 = 15 Hybrid Variants
Using numactl

- use only DDR (default)
  
  `numactl --membind=0 ./a.out`

- use only MCDRAM in flat-quadrant mode
  
  `numactl --membind=1 ./a.out`

- use MCDRAM if possible in flat-quadrant mode; else DDR
  
  `numactl --preferred=1 ./a.out`

- show numactl settings
  
  `numactl --hardware`

- list available numactl options
  
  `numactl --help`
Using numactl in various modes

- **Flat-quadrant mode**: use only MCDRAM
  `numactl --membind=1 ./a.out`

- **Flat-SNC2 mode**: use only MCDRAM
  `numactl --membind=2,3 ./a.out`

- **Flat-SNC4 mode**: use only MCDRAM
  `numactl --membind=4,5,6,7 ./a.out`
Changing of Memory and Cluster Modes

- KNL = single chip solution that can change the design of a machine at a level that has traditionally unchangeable
- Operating Systems and applications are not prepared for dynamically changing NUMA distances or changing memory and caching structures
- → Changing either cluster mode or memory mode requires a rebuild of tag directories
  - Requires reboot
  - Takes 15-20 minutes
Selection of Memory Modes in BIOS

Uncore Configuration

Uncore Status
- MMIO P2P Disable: <No>
- Cluster Mode: <On>
- Memory Mode: <Flat>
- OPIO Parallel Training: <En>
- OPIO Parallel Training: <On>
- Channel Count: <Flat>
- MCDRAM Throttling Mode: <Disable>
- MCDRAM Repair: <Enable>
- MCDRAM Diagnostics: <Enable>
- MCDRAM Firmware Update: <Enable>
- MCDRAM Data in NURAM: <Store>
- EDC Demand Scrub: <Enable>

Select Memory Mode:
Auto will try to set Cache mode if supported.
Selection of Cluster Modes in BIOS
for memorymode in cache flat hybrid; do
  for clustermode in a2a quad hemi snc2 snc4; do
    #SBATCH --constraint=$memorymode,$clustermode
    salloc --constraint=$memorymode,$clustermode ...
  done
done
Following the suggestions of the Intel experts, we finally adopted one configuration only for all the KNL racks serving the knldebug and knlprod (academic) queues, namely: cache/quadrant.

The queues serving the Marconi FUSION partition allow instead the use of nodes in flat/quadrant or cache/quadrant modes.
Using the memkind library

- https://github.com/memkind/memkind

- The *memkind* library is a **user extensible heap manager** built on top of *jemalloc* (http://jemalloc.net/) which enables control of memory characteristics and a **partitioning of the heap between kinds of memory**.

- The *memkind* library delivers two interfaces:
  - `hbwmalloc.h` - recommended for high-bandwidth memory use cases (stable) → `man memkind`
  - `memkind.h` - generic interface for more complex use cases (partially unstable) → `man hbwmalloc`
Using the memkind library

SYNOPSIS

```
#include <hbwmalloc.h>
int hbw_check_available(void);
void* hbw_malloc(size_t size);
void* hbw_calloc(size_t nmemb, size_t size);
void* hbw_realloc (void *ptr, size_t size);
void hbw_free(void *ptr);
int hbw_posix_memalign(void **memptr, size_t alignment, size_t size);
int hbw_posix_memalign_psize(void **memptr, size_t alignment, size_t size,
                              hbw_pagesize_t pagesize);

hbw_policy_t hbw_get_policy(void);
int hbw_set_policy(hbw_policy_t mode);
int hbw_verify_memory_region(void *addr, size_t size, int flags);
```

Details: man hbwmalloc

24.-25.1.2018 Programming the new KNL Cluster at LRZ
Using the memkind library: Policies

`hbw_set_policy()` sets the current fallback policy. The policy can be modified only once in the lifetime of an application and before calling `hbw_*alloc()` or `hbw_posix_memalign*()` function. Note: If the policy is not set, than HBW_POLICY_PREFERRED will be used by default.

**HBW_POLICY_BIND**

If insufficient high bandwidth memory from the nearest NUMA node is available to satisfy a request, the allocated pointer is set to NULL and `errno` is set to ENOMEM. If insufficient high bandwidth memory pages are available at fault time the Out Of Memory (OOM) killer is triggered. Note that pages are faulted exclusively from the high bandwidth NUMA node nearest at time of allocation, not at time of fault.

**HBW_POLICY_PREFERRED**

If insufficient memory is available from the high bandwidth NUMA node closest at allocation time, fall back to standard memory (default) with the smallest NUMA distance.

**HBW_POLICY_INTERLEAVE**

Interleave faulted pages from across all high bandwidth NUMA nodes using standard size pages (the Transparent Huge Page feature is disabled).
Using the memkind library: alloc

Traditional:
```
#include <stdlib.h>
...
double *A;
A = (double*) malloc(sizeof(double) * N);
...
Free(A);
```

Memkind library:
```
#include <hbwmalloc.h>
...
double *A;
A = (double*) hbw_malloc(sizeof(double) * N);
...
hbw_free(A);
```
Using the memkind library: posix_memalign

**Traditional:**

```c
#include <stdlib.h>
...
int ret; double *A;
ret = posix_memalign((void *)A, 64, sizeof(double)*N);
if (ret!=0) //error
...
free (A);
```

**Memkind library:**

```c
#include <hbwmalloc.h>
...
int ret; double *A;
ret = hbw_posix_memalign((void*) A, 64, sizeof(double)*N);
if (ret!=0) //error
...
hbw_free (A);
```
Intel Fortran Extensions for MCDRAM

**Traditional:**

```fortran
real, allocatable :: A(:)
...
ALLOCATE (A(1:1024))
```

**MCDRAM:**

```fortran
real, allocatable :: A(:)
!DIR$ ATTRIBUTES FASTMEM :: A
!DIR$ ATTRIBUTES FASTMEM, ALIGN:64 :: A  ! Alternative for alignment
...
ALLOCATE (A(1:1024))
```

**Alternativ:**

```fortran
real, allocatable :: A(:)
!dir# FASTMEM
ALLOCATE (A(1:1024))
```

FOR_SET_FASTMEM_POLICY(...) to change policy.
OpenMP 5.0 MCDRAM Support

- Memory Management Support for OpenMP 5.0
- Support for new types of memory: High Bandwidth Memory, Non-volatile memory etc.


- This Technical Report augments the OpenMP TR 4 document with language **features for managing memory on systems with heterogeneous memories**.
- To be released approx. Nov. 2018
The following slides contain material obtained on the Stampede2 Supercomputer at Texas Advanced Computing Center, The University of Texas at Austin

https://portal.tacc.utexas.edu/user-guides/stampede2

Stampede2 is the flagship supercomputer at the Texas Advanced Computing Center (TACC). It will enter full production in the Fall 2017 as an 18-petaflop national resource that builds on the successes of the original Stampede system it replaces.

Thanks to TACC for access to Stempede2 during an ISC‘17 Tutorial.
Stampede2 Supercomputer

24.-25.1.2018
Programming the new KNL Cluster at LRZ
Stampede2 Supercomputer

Model: Intel Xeon Phi 7250
Total cores per KNL node: 68 cores on a single socket
Hardware threads per core: 4
Hardware threads per node: 68 x 4 = 272
Clock rate: 1.4GHz

RAM: 96GB DDR4 plus 16GB high-speed MCDRAM. Configurable in two important ways; see Programming and Performance for more info.

All but 508 KNL nodes have a 132GB /tmp partition on a 200GB Solid State Drive (SSD). The 508 KNLs originally installed as the

Local storage: Stampede1 KNL sub-system each have a 58GB /tmp partition on 112GB SSDs. The latter nodes currently make up the flat-quadrant and flat-snc4 queues.
Useful commands and system files

- /proc/cpuinfo
- /proc/meminfo
- numastat -H
- numastat -m *(includes huge page info, too)*
- numastat -p pid
- /sys/devices/system/node/node*/meminfo
  and other files
- /usr/bin/memkind-hbw-nodes
/proc/cpuinfo

processor : 271
vendor_id : GenuineIntel
cpu family : 6
model : 87
model name : Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz
stepping : 1
microcode : 0x1ac
cpu MHz : 1543.445
cache size : 1024 KB
physical id : 0
siblings : 272
core id : 73
cpu cores : 68
apicid : 295
initial apicid : 295
fpu : yes
fpu_exception : yes
cpuid level : 13
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 cflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl est tm2 ssse3 fma cx16 xtpr pdcm sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer xsave avx f16c rdrand lahf_lm abm 3dnowprefetch ida arat epb pln pts dtherm fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms avx512f rdseed adx avx512pf avx512er avx512cd xsaveopt
bogomips : 2793.63
ciflush size : 64
cache_alignment : 64
address sizes : 46 bits physical, 48 bits virtual
power management:

c403-003.stampede2(9)
/proc/meminfo

- **Flat-Quadrant Mode:**
  MemTotal: 115218908 kB
  MemFree: 108756608 kB
  MemAvailable: 108562240 kB

- **Cache-Quadrant Mode:**
  MemTotal: 98696336 kB
  MemFree: 92462428 kB
  MemAvailable: 92282108 kB

- **Flat-SNC-4 Mode:**
  MemTotal: 115217380 kB
  MemFree: 109216500 kB
  MemAvailable: 108983732 kB
numactl -H in Flat-Quadrant Mode

available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
node 0 size: 98207 MB
node 0 free: 90483 MB
node 1 cpus:
node 1 size: 16384 MB
node 1 free: 15723 MB
node distances:
node   0    1
  0:  10  31
  1:  31  10
numactl -H in Cache Mode

available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107
108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128
129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191
192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212
213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254
255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271

node 0 size: 98199 MB
node 0 free: 90294 MB
node distances:

node  0
  0:  10
**numactl** -H in Flat-SNC4 Mode

available: 8 nodes (0-7)

node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 68 69 70 71 72 73 74 75 76 77 78
79 80 81 82 83 84 85 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152
153 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
node 0 size: 24479 MB
node 0 free: 21852 MB

node 1 cpus: 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 86 87 88 89 90 91 92 93
94 95 96 97 98 99 100 101 102 103 154 155 156 157 158 159 160 161 162 163 164 165 166 167
168 169 170 171 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
node 1 size: 24576 MB
node 1 free: 22867 MB

node 2 cpus: 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 104 105 106 107 108 109 110
111 112 113 114 115 116 117 118 119 120 121 122 123 124 240 241 242 243 244 245 246 247
248 249 250 251 252 253 254 255
node 2 size: 24576 MB
node 2 free: 22887 MB

node 3 cpus: 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 120 121 122 123 124 125 126
127 128 129 130 131 132 133 134 135 188 189 190 191 192 193 194 195 196 197 198 199 200 201
202 203 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
node 3 size: 24576 MB
node 3 free: 23144 MB

24.-25.1.2018  Programming the new KNL Cluster at LRZ
numactl -H in Flat-SNC4 Mode contnd.

node 4 cpus:
node 4 size: 4096 MB
node 4 free: 3968 MB
node 5 cpus:
node 5 size: 4096 MB
node 5 free: 3976 MB
node 6 cpus:
node 6 size: 4096 MB
node 6 free: 3976 MB
node 7 cpus:
node 7 size: 4096 MB
node 7 free: 3975 MB
node distances:

Distances:
10 „near“ DDR
21 „far“ DDR
31 „near“ MCDRAM
41 „far“ MCDRAM

Affinitization of DDR and MCDRAM to the divisions of the KNL!
/usr/bin/memkind-hbw-nodes

**Flat-Quadrant:**
c455-001.stampede2(17)$ memkind-hbw-nodes
1
c455-001.stampede2(18)$

**Flat-SNC4:**
c463-001.stampede2(1)$ memkind-hbw-nodes
4,5,6,7
c463-001.stampede2(2)$

**Cache:**
c403-001.stampede2(2)$ memkind-hbw-nodes
c403-001.stampede2(3)$
**STREAM Benchmark in Cache Mode**

```
export OMP_NUM_THREADS=1
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           17843.4     0.009051     0.008967     0.009453
Scale:          14305.0     0.011287     0.011185     0.011523
Add:            15736.8     0.015452     0.015251     0.015739
Triad:          15622.9     0.015512     0.015362     0.015851
```

```
export OMP_NUM_THREADS=68
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:          263585.5     0.000624     0.000607     0.000640
Scale:         269297.2     0.000612     0.000594     0.000631
Add:           325244.9     0.000772     0.000738     0.000798
Triad:         308499.2     0.000803     0.000778     0.000872
```
#### STREAM Benchmark in Flat-Quadrant Mode

```bash
c455-001.stampede2(27)$ export OMP_NUM_THREADS=1
c455-001.stampede2(30)$ ./stream
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>18344.3</td>
<td>0.008848</td>
<td>0.008722</td>
<td>0.009068</td>
</tr>
<tr>
<td>Scale:</td>
<td>15184.7</td>
<td>0.010682</td>
<td>0.010537</td>
<td>0.011006</td>
</tr>
<tr>
<td>Add:</td>
<td>16788.1</td>
<td>0.014392</td>
<td>0.014296</td>
<td>0.014504</td>
</tr>
<tr>
<td>Triad:</td>
<td>16712.0</td>
<td>0.014462</td>
<td>0.014361</td>
<td>0.014548</td>
</tr>
</tbody>
</table>

```bash
c455-001.stampede2(21)$ numactl -m 0 ./stream
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>18377.9</td>
<td>0.008787</td>
<td>0.008706</td>
<td>0.008876</td>
</tr>
<tr>
<td>Scale:</td>
<td>15207.8</td>
<td>0.010655</td>
<td>0.010521</td>
<td>0.010875</td>
</tr>
<tr>
<td>Add:</td>
<td>16677.2</td>
<td>0.014519</td>
<td>0.014391</td>
<td>0.014714</td>
</tr>
<tr>
<td>Triad:</td>
<td>16724.8</td>
<td>0.014447</td>
<td>0.014350</td>
<td>0.014541</td>
</tr>
</tbody>
</table>

```bash
c455-001.stampede2(24)$ numactl -m 1 ./stream
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>17999.9</td>
<td>0.009024</td>
<td>0.008906</td>
<td>0.009474</td>
</tr>
<tr>
<td>Scale:</td>
<td>14455.0</td>
<td>0.011269</td>
<td>0.011069</td>
<td>0.011690</td>
</tr>
<tr>
<td>Add:</td>
<td>15846.7</td>
<td>0.015331</td>
<td>0.015145</td>
<td>0.016039</td>
</tr>
<tr>
<td>Triad:</td>
<td>15850.0</td>
<td>0.015344</td>
<td>0.015142</td>
<td>0.015800</td>
</tr>
</tbody>
</table>
```

---

24.-25.1.2018

Programming the new KNL Cluster at LRZ
STREAM Benchmark in Flat-Quadrant Mode

```bash
c455-001.stampede2(27)$  export OMP_NUM_THREADS=68
c455-001.stampede2(30)$  ./stream
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>82768.7</td>
<td>0.001939</td>
<td>0.001933</td>
<td>0.001945</td>
</tr>
<tr>
<td>Scale:</td>
<td>82901.6</td>
<td>0.001937</td>
<td>0.001930</td>
<td>0.001947</td>
</tr>
<tr>
<td>Add:</td>
<td>88332.1</td>
<td>0.002731</td>
<td>0.002717</td>
<td>0.002741</td>
</tr>
<tr>
<td>Triad:</td>
<td>88425.2</td>
<td>0.002732</td>
<td>0.002714</td>
<td>0.002757</td>
</tr>
</tbody>
</table>

```bash
ac455-001.stampede2(28)$  numactl -m 0 ./stream
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>83334.0</td>
<td>0.001937</td>
<td>0.001920</td>
<td>0.001961</td>
</tr>
<tr>
<td>Scale:</td>
<td>82768.7</td>
<td>0.001938</td>
<td>0.001933</td>
<td>0.001950</td>
</tr>
<tr>
<td>Add:</td>
<td>88131.1</td>
<td>0.002742</td>
<td>0.002723</td>
<td>0.002769</td>
</tr>
<tr>
<td>Triad:</td>
<td>88138.8</td>
<td>0.002745</td>
<td>0.002723</td>
<td>0.002754</td>
</tr>
</tbody>
</table>

```bash
ac455-001.stampede2(29)$  numactl -m 1 ./stream
```

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>397093.9</td>
<td>0.000434</td>
<td>0.000403</td>
<td>0.000473</td>
</tr>
<tr>
<td>Scale:</td>
<td>426901.2</td>
<td>0.000400</td>
<td>0.000375</td>
<td>0.000428</td>
</tr>
<tr>
<td>Add:</td>
<td>433146.7</td>
<td>0.000578</td>
<td>0.000554</td>
<td>0.000602</td>
</tr>
<tr>
<td>Triad:</td>
<td>345328.6</td>
<td>0.000835</td>
<td>0.000695</td>
<td>0.000944</td>
</tr>
</tbody>
</table>

24.-25.1.2018 Programming the new KNL Cluster at LRZ
STREAM Benchmark in Flat-SNC4 Mode

```
c455-001.stampede2(27)$ export OMP_NUM_THREADS=68
c455-001.stampede2(30)$  ./stream

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>74773.1</td>
<td>0.002148</td>
<td>0.002140</td>
<td>0.002189</td>
</tr>
<tr>
<td>Scale:</td>
<td>78970.2</td>
<td>0.002031</td>
<td>0.002026</td>
<td>0.002038</td>
</tr>
<tr>
<td>Add:</td>
<td>79156.5</td>
<td>0.003040</td>
<td>0.003032</td>
<td>0.003066</td>
</tr>
<tr>
<td>Triad:</td>
<td>79337.4</td>
<td>0.003032</td>
<td>0.003025</td>
<td>0.003050</td>
</tr>
</tbody>
</table>

c455-001.stampede2(30)$  numactl -m 0 ./stream

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>36012.3</td>
<td>0.004460</td>
<td>0.004443</td>
<td>0.004475</td>
</tr>
<tr>
<td>Scale:</td>
<td>36093.6</td>
<td>0.004448</td>
<td>0.004433</td>
<td>0.004468</td>
</tr>
<tr>
<td>Add:</td>
<td>38903.7</td>
<td>0.006185</td>
<td>0.006169</td>
<td>0.006212</td>
</tr>
<tr>
<td>Triad:</td>
<td>38942.8</td>
<td>0.006176</td>
<td>0.006163</td>
<td>0.006206</td>
</tr>
</tbody>
</table>

--

c455-001.stampede2(30)$  numactl -m 1 ./stream

<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>36381.3</td>
<td>0.004488</td>
<td>0.004398</td>
<td>0.004804</td>
</tr>
<tr>
<td>Scale:</td>
<td>36430.6</td>
<td>0.004543</td>
<td>0.004392</td>
<td>0.005053</td>
</tr>
<tr>
<td>Add:</td>
<td>39525.4</td>
<td>0.006552</td>
<td>0.006072</td>
<td>0.007996</td>
</tr>
<tr>
<td>Triad:</td>
<td>39519.2</td>
<td>0.006190</td>
<td>0.006073</td>
<td>0.006780</td>
</tr>
</tbody>
</table>
```

24.-25.1.2018

Programming the new KNL Cluster at LRZ
**STREAM Benchmark in Flat-SNC4 Mode**

```
c455-001.stampede2(27)$ export OMP_NUM_THREADS=68
c455-001.stampede2(27)$ numactl -m 2 ./scratch
Function        Best Rate MB/s  Avg time     Min time     Max time
Copy:            34669.0     0.004625     0.004615     0.004635
Scale:           34692.3     0.004621     0.004612     0.004630
Add:             38199.5     0.006288     0.006283     0.006297
Triad:           38215.4     0.006287     0.006280     0.006298

c455-001.stampede2(27)$ numactl -m 3 ./scratch
Function        Best Rate MB/s  Avg time     Min time     Max time
Copy:            34535.2     0.004650     0.004633     0.004665
Scale:           34601.1     0.004644     0.004624     0.004664
Add:             38161.8     0.006322     0.006289     0.006349
Triad:           38198.0     0.006324     0.006283     0.006346

c455-001.stampede2(27)$ numactl -m 4 ./scratch
Function        Best Rate MB/s  Avg time     Min time     Max time
Copy:           105186.3     0.001539     0.001521     0.001560
Scale:          105334.9     0.001541     0.001519     0.001558
Add:            101516.0     0.002378     0.002364     0.002406
Triad:          102040.8     0.002371     0.002352     0.002398
```
STREAM Benchmark in Flat-SNC4 Mode

```bash
c455-001.stampede2(27)$ export OMP_NUM_THREADS=68

c455-001.stampede2(27)$ numactl -m 5 ./scratch
<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>107442.9</td>
<td>0.001497</td>
<td>0.001489</td>
<td>0.001504</td>
</tr>
<tr>
<td>Scale:</td>
<td>107236.9</td>
<td>0.001496</td>
<td>0.001492</td>
<td>0.001503</td>
</tr>
<tr>
<td>Add:</td>
<td>104206.3</td>
<td>0.002312</td>
<td>0.002303</td>
<td>0.002325</td>
</tr>
<tr>
<td>Triad:</td>
<td>104672.2</td>
<td>0.002295</td>
<td>0.002293</td>
<td>0.002304</td>
</tr>
</tbody>
</table>

c455-001.stampede2(27)$ numactl -m 6 ./scratch
<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>98834.9</td>
<td>0.001627</td>
<td>0.001619</td>
<td>0.001633</td>
</tr>
<tr>
<td>Scale:</td>
<td>98762.1</td>
<td>0.001628</td>
<td>0.001620</td>
<td>0.001635</td>
</tr>
<tr>
<td>Add:</td>
<td>93823.6</td>
<td>0.002569</td>
<td>0.002558</td>
<td>0.002586</td>
</tr>
<tr>
<td>Triad:</td>
<td>94086.6</td>
<td>0.002561</td>
<td>0.002551</td>
<td>0.002572</td>
</tr>
</tbody>
</table>

c455-001.stampede2(27)$ numactl -m 7 ./scratch
<table>
<thead>
<tr>
<th>Function</th>
<th>Best Rate MB/s</th>
<th>Avg time</th>
<th>Min time</th>
<th>Max time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy:</td>
<td>99938.7</td>
<td>0.001608</td>
<td>0.001601</td>
<td>0.001615</td>
</tr>
<tr>
<td>Scale:</td>
<td>99317.5</td>
<td>0.001614</td>
<td>0.001611</td>
<td>0.001620</td>
</tr>
<tr>
<td>Add:</td>
<td>94599.5</td>
<td>0.002545</td>
<td>0.002537</td>
<td>0.002550</td>
</tr>
<tr>
<td>Triad:</td>
<td>94929.6</td>
<td>0.002535</td>
<td>0.002528</td>
<td>0.002543</td>
</tr>
</tbody>
</table>
```

24.-25.1.2018 Programming the new KNL Cluster at LRZ
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 6000000000

int main(int argc, char **argv)
{
    float *list;
    long long int i;

    list = (float *) malloc(N * sizeof(float));
    if(list != NULL) {printf("memory is reserved\n");
    } else {printf("No memory free.\n");}

    printf("Sizeof float: %i Bytes\n",sizeof(float));
    printf("Sizeof list: %lli Bytes\n",sizeof(float)*N);
    printf("Sizeof list: %lli GB\n",sizeof(float)*N/1024/1024/1024);

    for(i=0; i<N; i++) list[i] = i;
    return 0;
}
Oversubscription of MCDRAM

c455-002.stampede2(14)$ time numactl -m 0 ./alloc
memory is reserved
Sizeof float:  4 Bytes
Sizeof list:  24000000000 Bytes
Sizeof list:  22 GB

real    1m38.960s
user    1m35.385s
sys     0m3.571s
c455-002.stampede2(15)$
Oversubscription of MCDRAM

c455-002.stampede2(11)$ time numactl -m 1 ./alloc
memory is reserved
Sizeof float:  4 Bytes
Sizeof list:  24000000000 Bytes
Sizeof list:  22 GB
Killed
real    1m11.186s
user    1m4.327s
sys     0m6.472s
c455-002.stampede2(12)$

24.-25.1.2018
Programming the new KNL Cluster at LRZ
Oversubscription of MCDRAM

- Using the memkind library:

```
#include <hbwmalloc.h>
...
list = (float *) hbw_malloc(N * sizeof(float));
```

c455-001.stampede2(6)$ icc -O0 alloc2.c -lmemkind -o alloc2
c455-001.stampede2(7)$ ./alloc2

```
hbw_check_available=0
memory is reserved
Sizeof float: 4 Bytes
Sizeof list: 24000000000 Bytes
Sizeof list: 22 GB
c455-001.stampede2(8)$
```

As default fallback policy is HBW_POLICY_PREFERRED!
Performance

Cluster and Memory Modes Performance Comparison

From TACC ISC’17 Tutorial, Baseline = Flat-A2A DDR4

24.-25.1.2018

Programming the new KNL Cluster at LRZ
References


● **Tutorial**: ”*Introduction to Manycore Programming*”, Texas Advanced Computing Center, 2017. Available under a Creative Commons Attribution Non-Commercial 3.0 Unported License. https://creativecommons.org/licenses/by-nc/3.0/

● **Tutorial**: “*Introduction to KNL and the ARCHER KNL Cluster*” , 2017, Adrian Jackson, EPCC, licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US
Acknowledgements

- **IT4Innovation**, Ostrava.
- Partnership for Advanced Computing in Europe (**PRACE**)
- **Intel**
- **BMBF** (Federal Ministry of Education and Research)
- **Dr. Karl Fürlinger** (LMU)
- **J. Cazes, R. Evans, K. Milfeld, C. Proctor** (TACC)
- **Adrian Jackson** (EPCC)
Thank you for your participation!