THE LINUX FOUNDATION
OPEN SOURCE SUMMIT

Flavors of Memory supported by Linux, their use and benefit

Christopher Lameter, Ph.D. <cl@linux.com>  Jump Trading LLC

@qant

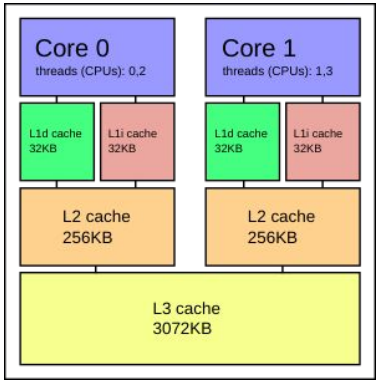
THE LINUX FOUNDATION

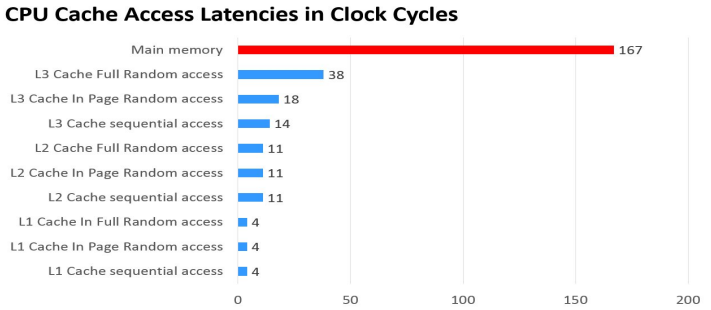
Flavors Of Memory

- The term ***computer memory*** is a simple term but there are numerous nuances to the meaning of memory within Linux
- Some new hardware types of memory
  - Non volatile memory (f.e. Intel ApachePass/3dXpoint)
  - Accelerator memory (GPU, FPGA, ManyCore) in coprocessors (heterogeneous memory management)
  - HBM (High Bandwidth Memory)
- Numerous qualifiers for regular main memory
  - NUMA, Local, Remote, Swap, Transcendent, Transparent, Huge, Zones, Node, DAX, RDMA.
- Terms used in connection with Memory Management
  - AutoNUMA, Reclaim, Swap, Migration, False Alias, Color, Fragmentation, Defragmentation, Affinity

“Simple” Memory Access

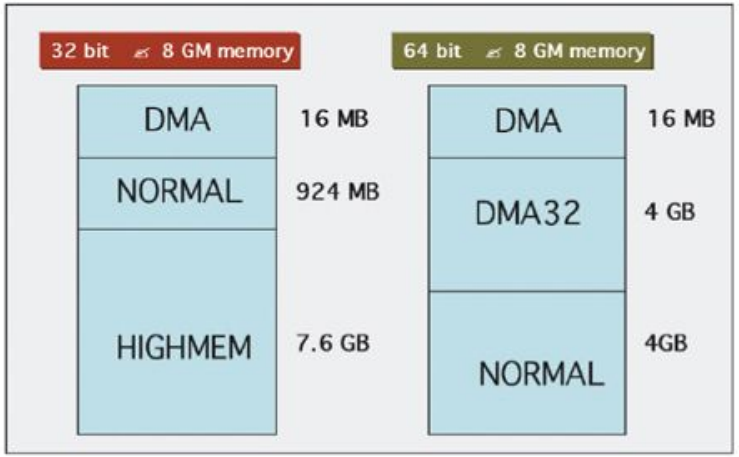
- ***UMA*** (Uniform Memory Access)
- Any access to memory has the same characteristics (performance and latency)
- The vast major of systems have only UMA.
- But there is always a processor cache hierarchy
  - The CPU is fast, memory is slow
  - Caches exist to avoid accesses to main memory
- ***Alias***ing
- ***Color***ing
- Cache Miss
- Trashing





Memory *ZONES*

- Some devices cannot write to all of memory.
  - System has allocation zones (ZONE_DMA, DMA32, NORMAL)
- Some processors cannot access all of memory
  - ZONE_HIGHMEM
- Isolate Memory that is movable for defragmentation
  - ZONE_MOVABLE
- Special I/O zone
  - ZONE_DEVICE



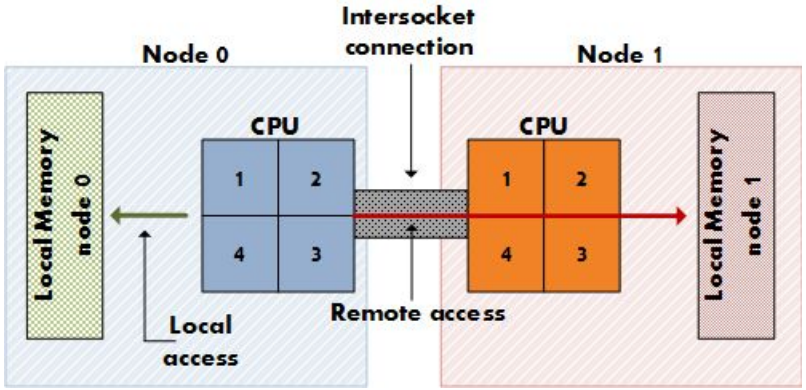
NUMA Memory

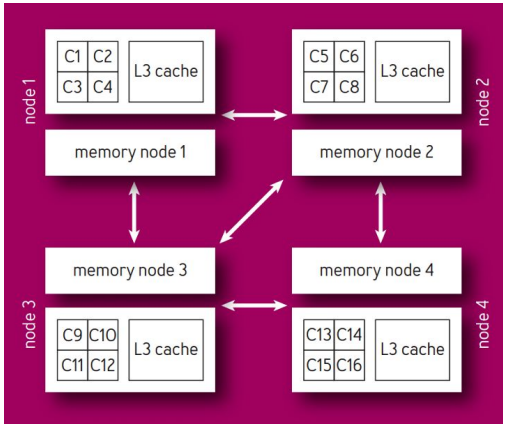
- Memory with different access characteristics

- Memory ***Affinities*** depending on where a process was started
- Control ***NUMA*** allocs with memory policies

- System Partitioning using Cpusets and Containers
- Manual memory ***migration***

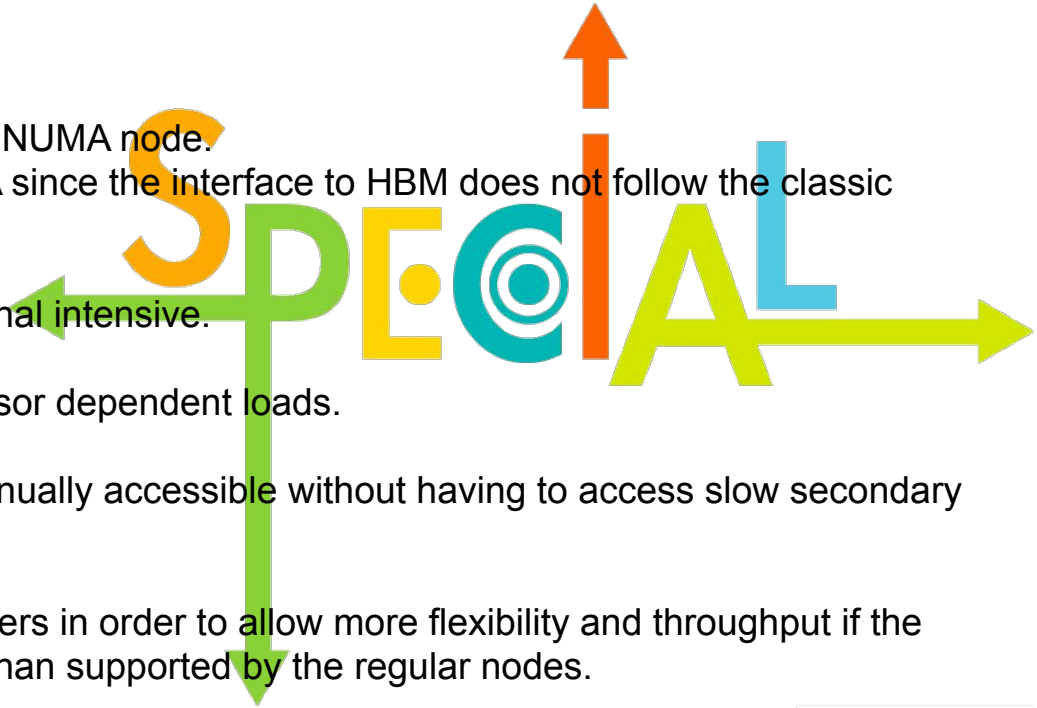
- Automatic memory migration





Special NUMA Nodes

- ***HBM* support (High Bandwidth Memory)**
  - HBM in use for GPUs etc
  - High bandwidth but also high latency
  - Supported in the Xeon Phi f.e. as a separate NUMA node.
  - Memory may behave differently under RDMA since the interface to HBM does not follow the classic memory channel design.
- **NUMA nodes with processors only**
  - Special use case where a load is computational intensive.
  - All memory access is remote.
  - Only makes sense for extremely high processor dependent loads.
- **Memory only NUMA nodes**
  - Expand memory to keep large datasets continually accessible without having to access slow secondary media.
- **Device only NUMA node**
  - Used to expand the availability of I/O expanders in order to allow more flexibility and throughput if the system requires a larger number of devices than supported by the regular nodes.
  - Rarely used.

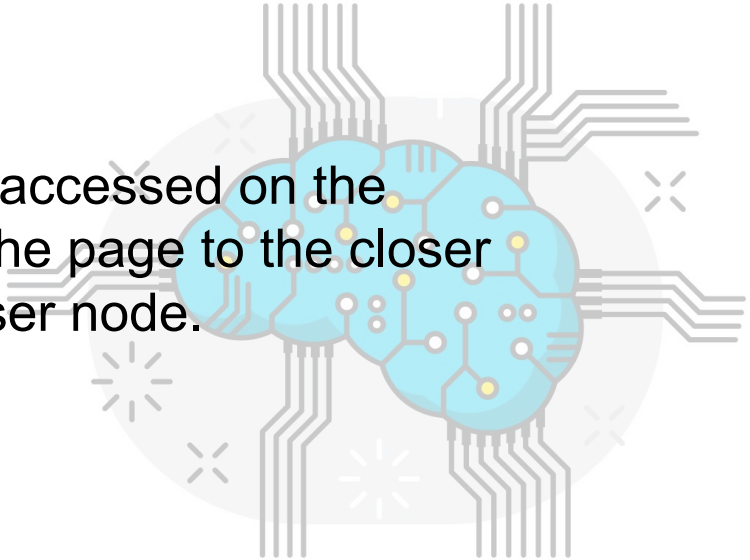


AutoNUMA: Local / Remote Memory

- Software run on NUMA system can show improved performance if *local* memory is used.

- ***AutoNUMA*** scans for memory that is accessed on the "wrong" node and then either moves the page to the closer node or moves the process to the closer node.

- A heuristic approach guessing how code will behave predicting future behavior based on past accesses



Huge Memory

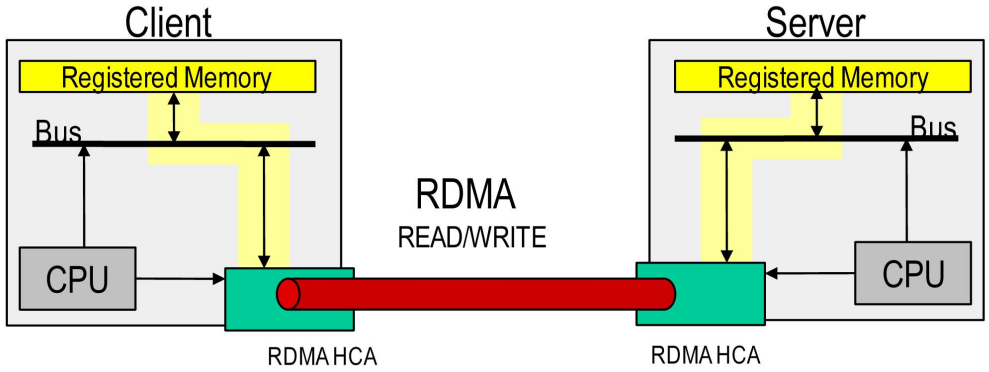
- Typical memory is handled in chunks of base page size (Intel 4k, IBM PowerX 64K, ARM 64K)
- Systems support larger memory chunks of memory called Huge pages (Intel 2M)
- Must be pre configured on boot in order to guarantee that they are available
- Required often for I/O bottlenecks on Intel.
- 4TB requires 1 billion descriptors with 4K pages. Most of this is needed to compensate for architectural problems on Intel. Intel processors have difficulties using modern SSDs and high speed devices without this.
- Large contiguous segments (I/O performance)
- Fragmentation issues
- Uses files on a special file system that must be explicitly requested by mmap operations from special files.

Transparent Huge Pages

- Allows the use of huge pages “transparently” without changes to the applications code.
- Improves performance (less TLB faults, more contiguous memory, less management by the OS)
- But less efficient than manual Huge pages
- Defragmentation requires to create more THPs
- madvise() to control THP allocations
- A potential to waste memory
- Scans memory to find contiguous pages that can be coalesced to huge pages
- ***Not supported (yet) for files on disks***

RDMA (Remote Direct Memory)

- Direct transfer data from user memory to user memory via the network
- Realized by the NIC
  - Full line rate
  - No processor use
  - No cache pollution
- Memory registration
  - Requires pinned memory
- ODP: On Demand Paging
  - Avoids pinning
  - Can register large amounts of memory (f.e. the whole process address space)
  - Memory registration can be moved out of critical loops.
  - NIC can generate faults and those faults may delay RDMA transfers
- The use of RDMA grows with memory and storage sizes.



Non Volatile Memory

- Memory contents are preserved unlike DRAM
- NVM can be addressed/handled like regular memory (usually with higher latency but without the PCI-e overhead of NVMe)
- Creates new challenges with data consistency that are still being worked out.

In a typical use case NVM is handled like a regular block device.

I.e. (output of ls)

ls -l /dev/pmem0

brw-rw---- 1 root disk 259, 3 Mar 29 14:03 /dev/pmem0

As usual the block device can be formatted and mounted

mkfs.ext4 /dev/pmem0

mount -o dax /dev/pmem0 /nvdimm

Files can be stored and retrieved from the file system like any other disk.

Files can be opened from the application with a special O_DAX option to allow direct mapping into application space.

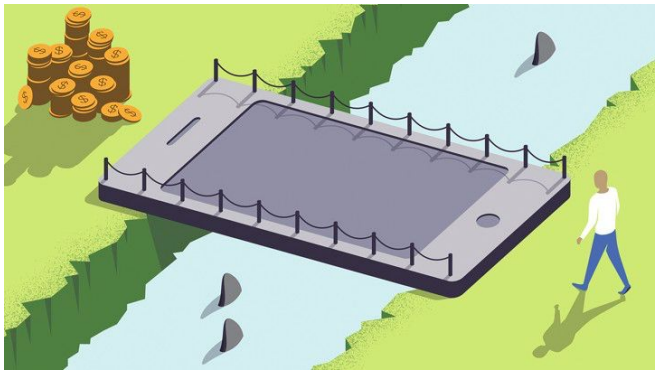
DAX (Direct Access to Device Memory)

File contents of a DAX file does not use the page cache for mmapped accesses.

Since the Nonvolatile Memory is directly accessible: Mmap directly establishes references to the data on the NVRAM module.

Contents cannot be evicted from memory thus there is no need to use the page cache and subject pages to eviction.

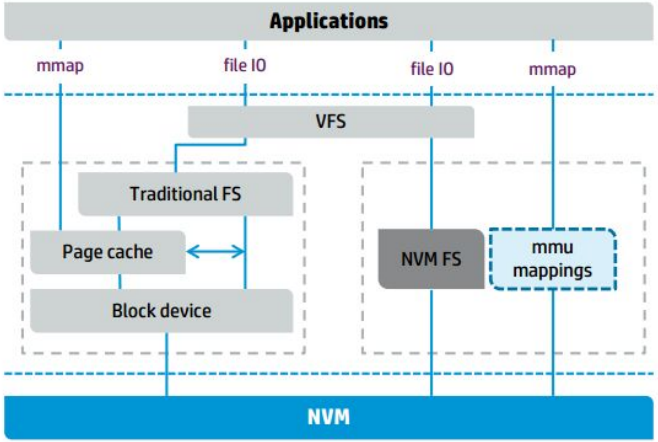
On Intel 2M and 1G mappings can exist in an NVM device. This significantly reduces OS overhead and TLB pressure.



THE LINUX FOUNDATION

NVM OS layering

- Using classic block Devices
- Additional direct DAX path via MMU
- Filesystem can decide which path is allowed
- Block device can control flushing
- Preserves POSIX file system semantics as much as possible
- Newer APIs can be used for direct access to the hardware but there are numerous conceptual/stability issues.



Nonvolatile Memory Challenges

Filesystems have something like transactions to ensure that the filesystem is always coherent. So we need something like that also on the level of NVM accesses. For that the processors cache coherency handling needs to be modified to allow transactions on NVM. Cachelines needs to be written back to NVM memory in such a way that a consistent state exists in case of failure.

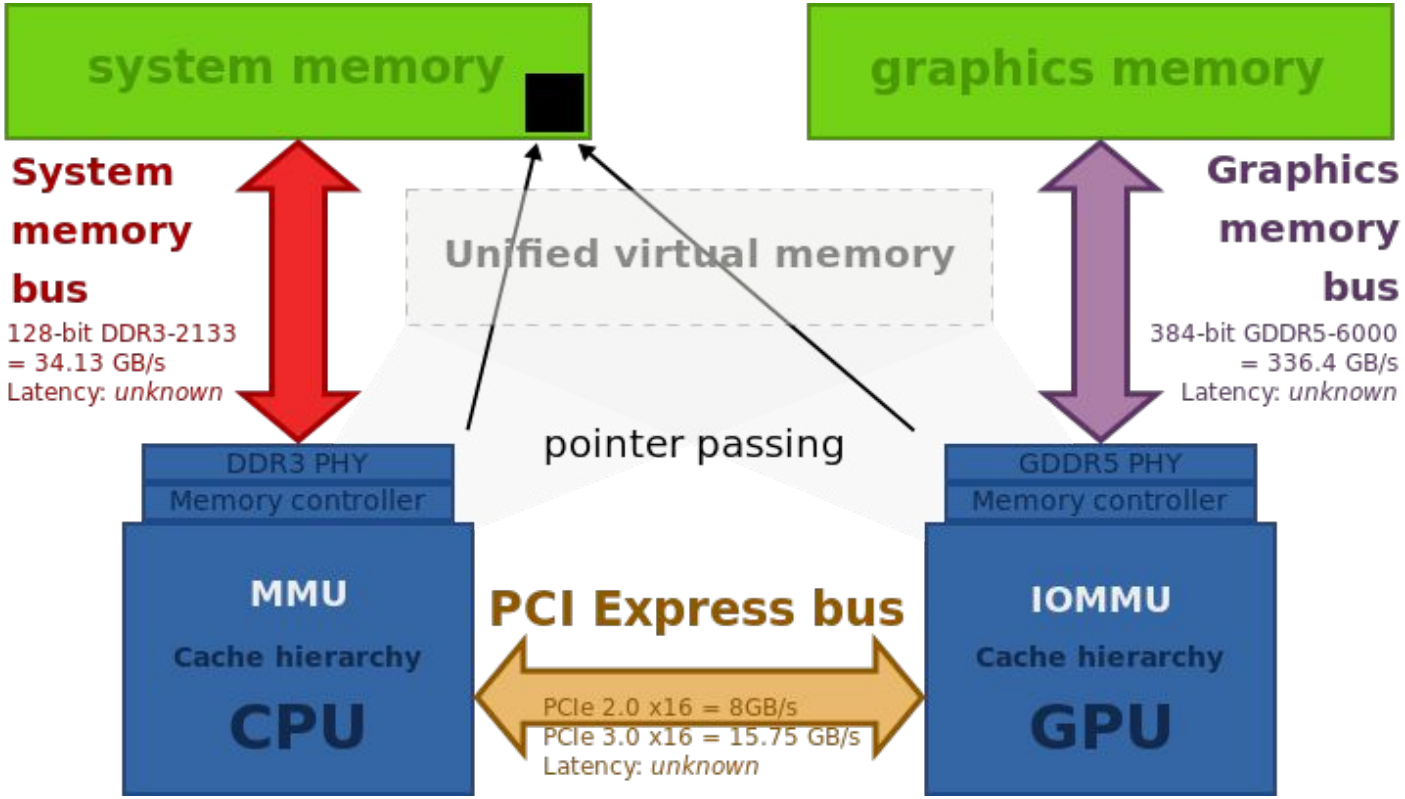
If the access to the NVM volume is through a file system then the file system coherency mechanisms can ensure file consistency within the constraints of each of their filesystems.

Non RDMA-able Non Volatile memory

Linux supports two different types of Non volatile memory. The overhead of keeping state per 4K page is about 2% of the capacity of a NVM device. NVM device have huge capacity and so we may end up dedicating significant amount of regular memory for NVM management.

ZONE_DEVICE has to be available for an NVM device in order to have metadata for 4K pages. If not the operations on NVM memory areas are limited.

Heterogeneous Memory



Heterogeneous Memory

- Enhancement of the mmu_notifiers build on the cross platform HSA (Heterogeneous System Architecture) standard.
- Managing large memory for I/O device (GPUs, coprocessors) with a virtual address space that allows the use of cross device pointers.
- External memory behind PCI-E and other device specific barriers that may be high latency and have their own coherency methods.
- HMM synchronizes memory views on both sides
- ZONE_DEVICE used to manage reference to memory that does not exist on the host side
- RDMA issues: Pages may never be mapped into main memory. Significant overhead if pages have to be copied for I/O. Maybe better to use PeerDirect?

Transcendent Memory

Allow access to memory in unusual ***transcendent*** way.

Establishes an in kernel memory API for fast memory that may be from a hypervisor, from a remote machine or locally compressed.

Common use cases

- Swap to memory through compression
- Swap to memory of another machine or the memory of the hypervisor

Future

- **RDMA squared**
  - Universal RDMAbility via a system wide ODP-like scheme?
  - Shared distributed memory via RDMA (shmem style) and process sharing
  - Remote “Persistent” Memory pools?
- **Remote RDMA for large storage arrays**
  - pNFS with RDMA layout. Access large array of NVMe memory via RDMA.
  - PeerDirect in remote nodes. Substitute device access for memory access.
  - NVM memory pools. Shared shared memory pools and filesystems that support such a shared memory pool.
- The memory hierarchy keeps expanding. Ultimately we need a multilayered generic memory infrastructure that moves data over a variety of busses to allow for optimal latency.

Questions / Comments

You can reach me at cl@linux.com or @qant on twitter

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

THE LINUX FOUNDATION