13th ANNUAL WORKSHOP 2017

THE RDMA SUBSYSTEM ISSUES AND THE CORE LINUX KERNEL

Christoph Lameter, Ph.D.

Jump Trading LLC

March 27th , 2017

OVERVIEW

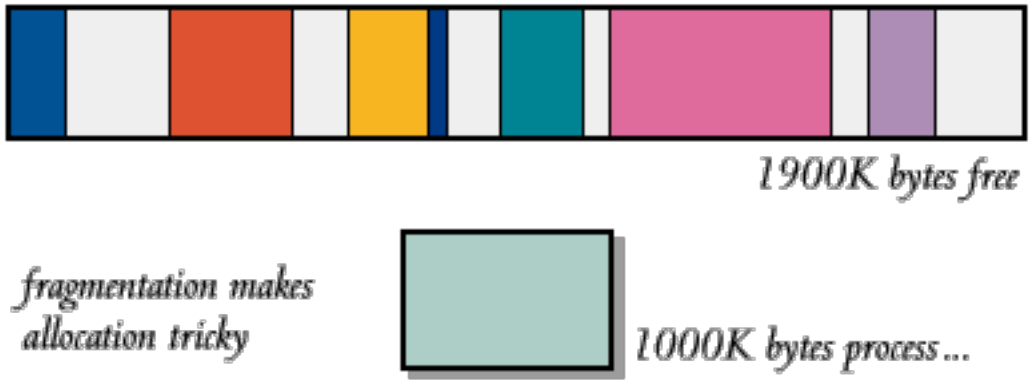
Core Kernel Issues that may impact the RDMA subsystem

- **Problems with contiguous memory**
- **Pinning / locking of memory**
- **XDP: in kernel network stack bypass**
- **DAX: Persistent memory support in the core VM**
- **Heterogeneous Memory (HMM) and Migration to device memory**
- **Defragmentation and large page availability, THP**
- **Ambient Capabilities**
- **Memory Hotplug**
- **Integration with the network stack**

CONTIGUOUS MEMORY
THE PERENNIAL KERNEL PROBLEM

CONTIGUOUS MEMORY ISSUES

- **My large kmalloc does not always work …**
- **Cannot allocate a huge page**
- **Must use vmalloc()**
- **Defragmentation does not work anymore after a couple of days**
- **Memory vanishes after a scan through large devices**



REASONS FOR THE MESS

- **Due to the fact that the kernel manages memory in 4K chunks.**
- **Ability to allocate large contiguous memory is marginal and not guaranteed.**
- **At boot large contiguous memory areas are available. As the system runs memory is broken down into smaller chunks and some of those become unmovable.**
- **After a while large allocations (huge pages f.e.) become impossible. Allocation attempts will fail.**
- **Large contiguous memory use at this point can only be optional not standard. Subsystems can temporarily exploit large contiguous allocations.**

WORKING ON A SOLUTION

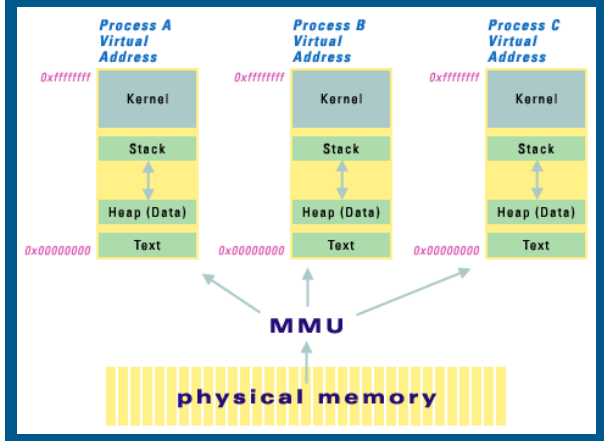
- Kernel supports defragmentation already
- Problem are the unmovable objects created for managing kernel metadata
- In particular these are created by the inodes and dentries of the filesystem.
- Work on making these object movable by equipping the allocators with the ability to recognize the object types and calling back into the respective subsystems to move objects.
- This enables:
  defragmentation,
  memory hotplug,
  object moving for performance.

- Status: Patches for basic functionality are there but the full bloom of the solution could take some time.

PINNING AND LOCKING OF MEMORY

MLOCK POSIX MEMORY LOCKING

A general way to ensure memory is not swapped out.



- This is not what memory registration does
- Mlock() system call
- POSIX semantics state that mlock'ed memory must stay in memory. It is exempt from swapping.
- There is nothing in the POSIX standard about the memory having the same physical address. We (the Linux memory management developers) interpret this that we have the freedom to change the physical location of MLOCKed memory.
- Some disagreement and confusion on this.
- Mlocked pages are subject to defragmentation, Same page elimination, page migration, compaction etc etc.
- Mlocked pages cannot be used for DMA.

PINNING MEMORY

RDMAs way of ensuring memory can be subject to remote memory access

- **Memory registration causes the RDMA subsystem to pin the memory.**
- **The physical address of memory is fixed while pinned thus the memory is suitable for DMA.**
- **Memory is pinned in 4K segments if not using huge pages (then its 2M).**
- **A 4GB memory region uses 1 Million OS descriptors of 4K segments or 2000 descriptors for huge pages.**

Memory may be pinned by multiple kernel subsystems simultanoeusly. This is usually a transient condition. Therefore the kernel may repeatedly be getting back to a memory page that is pinned to see if it can be physically moved. Large scans like that are done for compaction, duplicate page elimination, memory migration, swapping, defragmentation and so on. Attempts to move pinned memory will always fail but it may take some effort from the kernel with a page until the determination can be made that a page is truly pinned. The kernel will call on all kernel subsystems that may have pinned the page to release it before deciding that it is time to move on,

RECENT NEW INTERESTING KERNEL FUNCTIONALITY

DAX AND ZONE_DEVICE

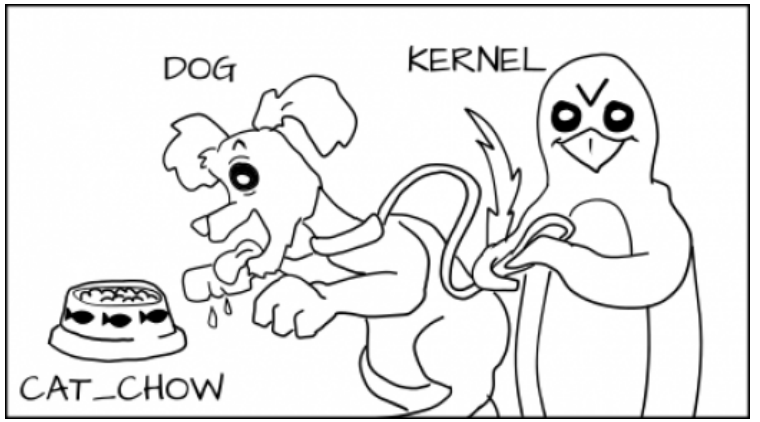
Enabling large scale memory access that may use persistent memory

- New functionality in the VM for device drivers.
- DAX allows referring to memory outside of what the kernel manages.
- File systems used to access DAX memory mapped regions.
- Data is not copied to main memory but a reference is established.
- ZONE_DEVICE used to allow basic kernel management for I/O
- ZONE_DEVICE makes it usable for the RDMA sub-system.
- Filesystems supporting DAX: ext2, ext4, xfs
- Throughput is through the roof of course since no actual transfers of memory take place.

AMBIENT CAPABILITIES

Giving processes the right to do what they need to do

- **Security limitations can prevent a userid from having the rights to exercise functionality needed for a process.**
- **Linux had a dysfunctional capabilities subsystem that was unable to give the security capabilities that we needed.**

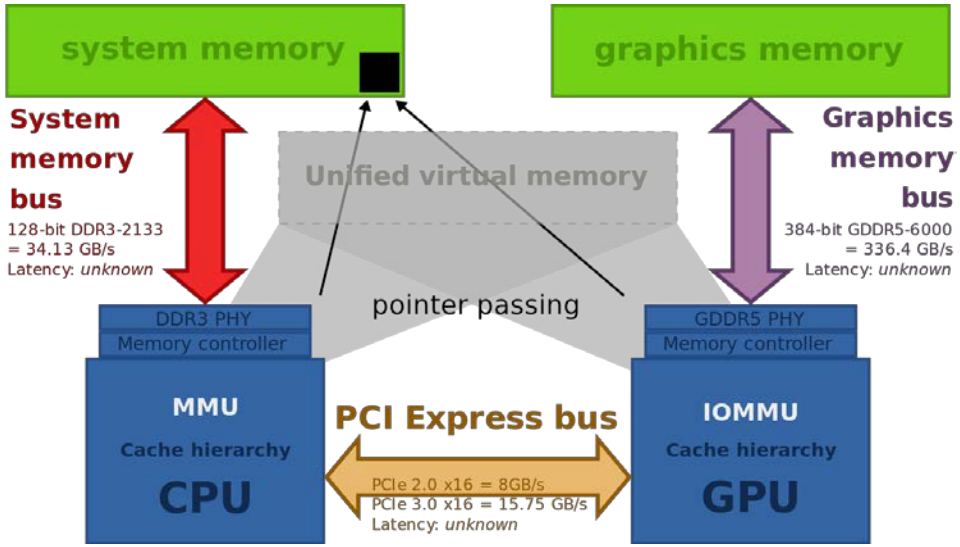


- **F.e. The case for being able to change process priority. SCHED_FIFO is intended only to be used for a short time period. This enables proper use of the OS functionality.**
- **RAW packet access**
- **Integrated in systemd and distros these days. RH 7.4 is going to have it.**
- **Can create shell with Capabilites which makes debugging easy.**

HETEROGENEOUS MEMORY AND DEVICE MEMORY OFFLOAD

Memory: More complexity to you ….

- **Managing memory in accelerators (GPUs, FPGAs, etc)**
- **HMM is realized using ZONE_DEVICE and maybe using DAX as well but not relying on a filesystem.**
- **Memory can be directly mapped from the accelerator.**
- **Device Memory offload can migrate memory between Linux and the accelerator. Shared state if both are reading. Otherwise one or the other has exclusive access to the page.**
- **Allows fast access to shared data between accelerators and Linux OS.**
- **Control of shared data set.**



MISCELLANEOUS INTERESTING STUFF

Trends and Technologies

- **XDP: in kernel network stack bypass**
- **Defragmentation and large page availability, THP**
- **Memory Hotplug**
- **Integration with the network stack**
- **Anything else?**

13th ANNUAL WORKSHOP 2017

THANK YOU

Christoph Lameter

Jump Trading LLC