Memory Management 101: Introduction to Memory Management in Linux

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

- Memory and processes
- Real/Virtual memory and Paging
- Machine configuration
- Processes use of memory
- Overcommit
- Knobs
- There is an advanced MM talk tomorrow called “Flavors of Memory”
Division of memory into “pages”
  ○ 1-N bytes become split at page size boundaries and become $M = \frac{N}{\text{page size}}$ pages

Refer to memory by the Page Frame Number (PFN) and an offset into the page.

Common size is 4k (Intel legacy issues)

MMU creates virtual addresses.
Basics of “paging”

- Process have virtual memory
- -> PFN
- Page Tables
- Faults
  - Major
  - Minor
- Virtual vs physical
Process memory

- Virtual memory maps to physical memory
- View of memory that is distinct for each process.
- Pages shared
- Access control
- Copy on Write
Swap, Zero pages etc.

- Swap page
- Zero page
- Read data behavior
- Write data behavior
- Anonymous vs file backed pages
Linux Basic memory information

/proc/meminfo

/sys/devices/system/
has lots of more detailed
information on hardware
(processors and
memory)

Commands:
numactl --hardware
free, top, dmesg

<table>
<thead>
<tr>
<th>Memory Information</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemTotal</td>
<td>31798552 kB</td>
</tr>
<tr>
<td>MemFree</td>
<td>25949124 kB</td>
</tr>
<tr>
<td>MemAvailable</td>
<td>30823580 kB</td>
</tr>
<tr>
<td>Buffers</td>
<td>220988 kB</td>
</tr>
<tr>
<td>Cached</td>
<td>4679188 kB</td>
</tr>
<tr>
<td>SwapCached</td>
<td>0 kB</td>
</tr>
<tr>
<td>Active</td>
<td>2803000 kB</td>
</tr>
<tr>
<td>Inactive</td>
<td>2336992 kB</td>
</tr>
<tr>
<td>Active(anon)</td>
<td>240776 kB</td>
</tr>
<tr>
<td>Inactive(anon)</td>
<td>6432 kB</td>
</tr>
<tr>
<td>Active(file)</td>
<td>2562224 kB</td>
</tr>
<tr>
<td>Inactive(file)</td>
<td>2330560 kB</td>
</tr>
<tr>
<td>Unevictable</td>
<td>0 kB</td>
</tr>
<tr>
<td>Mlocked</td>
<td>0 kB</td>
</tr>
<tr>
<td>SwapTotal</td>
<td>2097148 kB</td>
</tr>
<tr>
<td>SwapFree</td>
<td>2097148 kB</td>
</tr>
<tr>
<td>Dirty</td>
<td>48 kB</td>
</tr>
<tr>
<td>Writeback</td>
<td>0 kB</td>
</tr>
</tbody>
</table>

AnonPages: 239716 kB
Mapped: 195596 kB
Shmem: 7396 kB
Slab: 550628 kB
SReclaimable: 443040 kB
SUunreclaim: 107588 kB
KernelStack: 6840 kB
PageTables: 11176 kB
Inspecting a process memory use

/proc/<pid>/status
/proc/<pid>/*maps

(there are other files in /proc/<pid>/* with more information about the processes)

Commands:
ps, top

| Name: sshd | VmData: 1332 kB |
| VmPeak: 65772 kB | VmStk: 132 kB |
| VmSize: 65772 kB | VmExe: 492 kB |
| VmLck: 0 kB | VmLib: 8076 kB |
| VmPin: 0 kB | VmPTE: 168 kB |
| VmHWM: 6008 kB | VmSwap: 0 kB |
| VmRSS: 6008 kB | |
| RssAnon: 1216 kB | |
| RssFile: 4792 kB | |
| RssShmem: 0 kB | |
### User limit (ulimit)

- **max memory size**
- **virtual memory**
- **stack size**
- and lots of other controls.

```bash
cl@nuc-kabylake:/proc/6713$ ulimit -a
```

<table>
<thead>
<tr>
<th>Control</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>core file size</td>
<td>0 (blocks, -c)</td>
</tr>
<tr>
<td>data seg size</td>
<td>unlimited (kbytes, -d)</td>
</tr>
<tr>
<td>scheduling priority</td>
<td>0 (-e)</td>
</tr>
<tr>
<td>file size</td>
<td>unlimited (blocks, -f)</td>
</tr>
<tr>
<td>pending signals</td>
<td>123132 (-i)</td>
</tr>
<tr>
<td>max locked memory</td>
<td>16384 (kbytes, -l)</td>
</tr>
<tr>
<td>max memory size</td>
<td>unlimited (kbytes, -m)</td>
</tr>
<tr>
<td>open files</td>
<td>1024 (-n)</td>
</tr>
<tr>
<td>pipe size</td>
<td>8 (512 bytes, -p)</td>
</tr>
<tr>
<td>POSIX message queues</td>
<td>819200 (bytes, -q)</td>
</tr>
<tr>
<td>real-time priority</td>
<td>0 (-r)</td>
</tr>
<tr>
<td>stack size</td>
<td>8192 (kbytes, -s)</td>
</tr>
<tr>
<td>cpu time</td>
<td>unlimited (seconds, -t)</td>
</tr>
<tr>
<td>max user processes</td>
<td>123132 (-u)</td>
</tr>
<tr>
<td>virtual memory</td>
<td>unlimited (kbytes, -v)</td>
</tr>
<tr>
<td>file locks</td>
<td>unlimited (-x)</td>
</tr>
</tbody>
</table>
Overcommit configuration

Virtual memory use vs physical
overcommit_kbytes
overcommit_memory
  0 - overcommit. Guess if mem is available.
  1 - Overcommit. Never say there is no memory
  2 - Only allocate according to the ratio
overcommit_ratio
  total = swap + physical * ratio
Important VM control knobs

Found in /proc/sys/vm

More descriptions of these knobs in Kernel source code. [linux/Documentation/admin-guide](https://www.kernel.org/doc/Documentation/admin-guide)

- `admin_reserve_kbytes`
- `dirty_writeback_centisecs`
- `min_free_kbytes`
- `numa_zonelist_order`
- `stat_refresh`
- `block_dump`
- `drop_caches`
- `min_slab_ratio`
- `oom_dump_tasks`
- `swappiness`
- `compact_memory`
- `extfrag_threshold`
- `min_unmapped_ratio`
- `oom_kill_allocating_task`
- `user_reserve_kbytes`
- `compact_unevictable_allowed`
- `hugetlb_shm_group`
- `mmap_min_addr`
- `overcommit_kbytes`
- `vfs_cache_pressure`
- `dirty_background_bytes`
- `laptop_mode`
- `mmap_rnd_bits`
- `overcommit_memory`
- `watermark_scale_factor`
- `dirty_background_ratio`
- `legacy_va_layout`
- `mmap_rnd_compat_bits`
- `overcommit_ratio`
- `zone_reclaim_mode`
- `dirty_bytes`
- `lowmem_reserve_ratio`
- `nr_hugepages`
- `page-cluster`
- `dirty_expire_centisecs`
- `max_map_count`
- `nr_hugepages_mempolicy`
- `panic_on_oom`
- `dirty_ratio`
- `memory_failure_early_kill`
- `nr_overcommit_hugepages`
- `percpu_pagelist_fraction`
- `dirtytime_expire_seconds`
- `memory_failure_recovery`
- `numa_stat`
- `stat_interval`
Resources

- Admin Guide online
- Kernel.org has wikis and documentation ([www.kernel.org](http://www.kernel.org))
- manpages (especially for system calls and coding)
Questions / Comments

You can reach me at cl@linux.com or @qant on twitter
**“Simple” Memory Access**

- **UMA** (Uniform Memory Access)
- Any access to memory has the same characteristics (performance and latency)
- The vast majority 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
- **Aliasing**
- **Coloring**
- Cache Miss
- Trashing
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
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.
An Introduction to Linux memory management. The basics of paging. Understanding basic hardware memory management and the difference between virtual, physical and swap memory. How do determine hardware installed and how to figure out how processes use that memory. How a process uses physical and virtual memory effectively. How to control overcommit and virtual and/or physical memory limits.

Basic knobs in Linux to control memory management. System calls for a process to control its memory usage