MM101: Introduction to Memory Management

Christoph Lameter <cl@linux.com>
Overview

- Memory and processes
- Real/Virtual memory and Paging
- Machine configuration
- Processes use of memory
- Overcommit
- Knobs
- Processor cache use
Division of memory into “pages”
  ○ 1-N bytes become split at page size boundaries and become
    \[ M = \frac{N}{\text{page-size}} \]
    pages

We can then refer to memory by the Page Frame Number (PFN) and an offset into the page.

Common size is 4k (Intel legacy issues)

The MMU “creates” virtual addresses.
Basics of “paging”

- Processes have virtual memory
- \( \rightarrow \) PFN
- Page Tables

- Faults
  - Major
  - Minor

- Virtual vs physical
Process Memory

- Virtual memory maps to physical memory
- A view of memory 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
Kernel 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 Type</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>

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnonPages:</td>
<td>239716 kB</td>
</tr>
<tr>
<td>Mapped:</td>
<td>195596 kB</td>
</tr>
<tr>
<td>Shmem:</td>
<td>7396 kB</td>
</tr>
<tr>
<td>Slab:</td>
<td>550628 kB</td>
</tr>
<tr>
<td>SReclaimable:</td>
<td>443040 kB</td>
</tr>
<tr>
<td>SUNreclaim:</td>
<td>107588 kB</td>
</tr>
<tr>
<td>KernelStack:</td>
<td>6840 kB</td>
</tr>
<tr>
<td>PageTables:</td>
<td>11176 kB</td>
</tr>
</tbody>
</table>
Inspecting a processes use of memory

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

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

Commands:
ps, top

<table>
<thead>
<tr>
<th>Name: sshd</th>
<th>VmData: 1332 kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>VmPeak: 65772 kB</td>
<td>VmStk: 132 kB</td>
</tr>
<tr>
<td>VmSize: 65772 kB</td>
<td>VmExe: 492 kB</td>
</tr>
<tr>
<td>VmLck: 0 kB</td>
<td>VmLib: 8076 kB</td>
</tr>
<tr>
<td>VmPin: 0 kB</td>
<td>VmPTE: 168 kB</td>
</tr>
<tr>
<td>VmHWM: 6008 kB</td>
<td>VmSwap: 0 kB</td>
</tr>
<tr>
<td>VmRSS: 6008 kB</td>
<td></td>
</tr>
<tr>
<td>RssAnon: 1216 kB</td>
<td></td>
</tr>
<tr>
<td>RssFile: 4792 kB</td>
<td></td>
</tr>
<tr>
<td>RssShmem: 0 kB</td>
<td></td>
</tr>
</tbody>
</table>
User limit (ulimit)

- Max memory size
- Virtual memory
- Stack size
- and lots of other controls.

```bash
cl@nuc-kabylake:/proc/6713$ ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 123132
max locked memory (kbytes, -l) 16384
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 8192
cpu time (seconds, -t) unlimited
max user processes (-u) 123132
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
```
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`

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_overcommithugepages  percpu_pagelist_fraction  dirtytime_expire_seconds
memory_failure_recovery  numa_stat  stat_interval
Resource

- Admin Guide online
- Kernel.org has wikis and documentation ([www.kernel.org](http://www.kernel.org))
- *manpages* (especially for system calls and coding)
“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 the 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.
For questions and feedback please reach out to me at -
cl@linux.com
http://gentwo.org/christoph