

How Much Memory Does my System Use?

Richard Weinberger

2026-10-09



Hello



Richard Weinberger

- Co-founder of sigma star gmbh
- Linux kernel developer and maintainer
- Strong focus on Linux kernel, lowlevel components, virtualization, security, code audits

sigma star gmbh

- Software Development & Security Consulting
- Main areas: Embedded Systems, Linux Kernel & Security
- Contributions to Linux Kernel and other OSS projects

Why This Talk?

- While everything is shiny nobody cares
- Embedded systems usually have little memory
  - Real embedded systems, not a Raspberry Pi
  - 256MiB, 512MiB or 1GiB RAM are common
- Common questions:
  - How much memory is really used?
  - Can I reduce DRAM by 50% in the next generation?
  - How much memory does my application use?
  - How much memory does the kernel use?

Memory

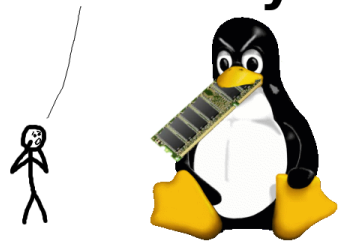
- Linux divides memory into pages
- Typical page size is 4KiB
- Page size depends on hardware
- Kernel offers various ways to allocate memory
  - Depends on the use case
- For more details see talk: *What Is Linux Doing With My RAM? - Vlastimil Babka, SUSE Labs*

free(1): Display amount of free and used memory in the system

```
rw@nailgun ~ ❯ free -m
               total        used        free      shared  buff/cache   available
Mem:           31585       16044         262        1817       17095       15540
Swap:          31574        8567       23006
rw@nailgun ~ ❯
```

www.linuxatemyram.com

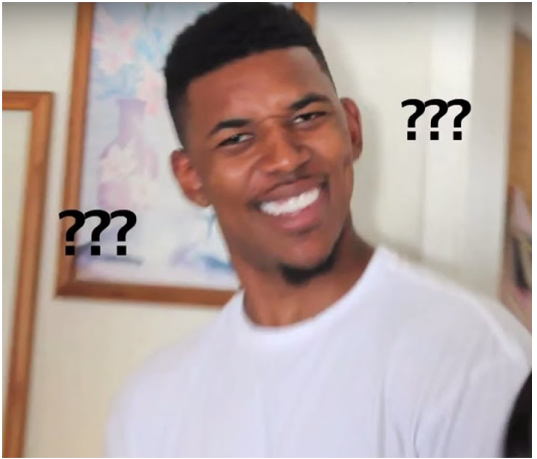
Linux ate my ram!



Don't Panic!
Your ram is fine!

Memory Metrics to the Rescue!

```
rw@nailgun ~ ❯ cat /proc/vmstat | wc -l
198
rw@nailgun ~ ❯ cat /proc/meminfo | wc -l
60
rw@nailgun ~ ❯
```



Mapping

- Memory is installed to userspace using mappings
- Mapping means both MMU and kernel know the virtual range
- While a mapping is virtually linear it is almost never physically

Types of Userspace memory

1. Anonymous Private
   - Memory is private to the process
   - Pages stay either in core (RAM) or are swapped to disk
   - Used for `malloc()`, `brk()`, user space stack
2. Anonymous Shared
   - Memory is shared between processes
   - Pages stay either in core (RAM) or are swapped to disk
   - Used for POSIX and SysV shared memory

Types of Userspace memory (cont'd)

3. File backed Private
   - Memory is private to the process
   - Reading from memory region triggers file read
   - Repeated reads, read from RAM (Page Cache)
   - Memory Pages can be removed from RAM (reclaim)
   - Used for shared libraries, programs
4. File backed Shared
   - Memory *can* be shared between processes
   - Like private file backed
   - But writes to memory region will be written back to file
   - Memory Pages can be removed from RAM (reclaim)
   - Reclaim triggers writeback

Memory Accounting Metrics

- Resident Size (RSS or RSZ): Anything from 1., 2., 3. and 4. that resides in RAM
- Text Size (TRS or CODE): Executable pages in RAM from 3.
- Data Size (DRS or DATA): Data pages in RAM from 1. and 3. without executable pages
- Shared Size (SHZ or SHM): Basically RES minus pages from 1.
- Virt Size (VSZ or VSS): Total size of all mappings
- Proportional Size (PSZ or PSS): Total amount of pages in RAM, shared pages broken up
- Unshared memory Size (USS): Total amount of pages in RAM, private

Kernel Memory: Zones

- Memory is physically into zones
- On x86_64, one zone for each NUMA domain
- One zone for 16bits devices (DMA)
- One zone for 32bits devices (DMA32)
- See `/proc/zoneinfo`

Kernel Memory: Buddy Allocator

- Page allocator
- Keeps lists of physically linear pages per zone
- Currently 11 (MAX_ORDER + 1) lists
- 2^order
- 1, 2, 4, 16, … pages
- Lists can be split up
- See `/proc/buddyinfo`

Kernel Memory: Slab Allocator

- General purpose sub page allocator
- Allocates itself from the page allocator
- Fast and per CPU
- See `/proc/slabinfo`
- Currently three different implementations:
  - SLAB: Original Slab allocator
  - SLOB: A traditional K&R allocator for small systems
  - SLUB: High performance waitfree allocator
- SLOB and SLAB gone in recent kernels
  - SLOB: RIP v6.4
  - SLAB: RIP v6.10

Kernel Memory: VMalloc Allocator

- General purpose allocator for virtually linear pages
- Allocates via page allocator and maps them together
- Useful for lager allocations, but with more overhead
- On x86_64 now also used for the kernel stack

Kernel Memory: CMA Allocator

- Contiguous memory allocator
- Maintains a set of pages and keeps them physically linear
- Can do defragmentation
- Used by device drivers which need large linear memory (GPUs, FPGAs)

Reclaim

- Linux uses a lot of memory for caching and unwritten data
- Reclaim means Linux reclaims these pages back to use them
- Cached data can be dropped
- Unwritten data need to be written (writeback)

free(1): Display amount of free and used memory in the system

```
rw@nailgun ~ ❯ free -m
               total        used        free      shared  buff/cache   available
Mem:           31585       16044         262        1817       17095       15540
Swap:          31574        8567       23006
rw@nailgun ~ ❯
```

Memory Usage per Process: Userspace

- RSS seems right, but isn't. VSS neither.
  - Pages accounted by RSS can be shared
  - Mostly due to shared libraries
- PSS it is
- Sadly PSS is not shown in `/proc/PID/status`
- `cat /proc/PID/smaps_rollup | grep -E ^Pss:`

Memory Usage per CGroup: Userspace + Kernel

- CGroups group processes
  - Don't confuse with process groups!
- A process can not leave a group
- Children inherit group membership
- *C* in CGroup stays for control
- A CGroup can have zero or more controllers
- There is also a memory controller
  - Purpose: Accounting and enforcing memory limits

Memory Usage per CGroup: Userspace + Kernel (cont'd)

- Accounts userspace and kernel memory directly used by processes
- Usecase: Containers
- systemd groups almost everything in s cgroup
  - See `systemd-cgls`
- CGroups life in `/sys/fs/cgroup/`
- e.g. `/sys/fs/cgroup/system.slice/wickedd.service`
- `memory.current` show amount of used memory in bytes

Memory Usage per CGroup: Userspace + Kernel (cont'd)

- `cat: /sys/fs/cgroup/system.slice/wickedd.service/memory.current: No such file or directory`
- memory controller not always enabled by default.
  - `DefaultMemoryAccounting` in systemd's `system.conf`
- See `/sys/fs/cgroup/cgroup.controllers`
- `echo +memory > /sys/fs/cgroup/cgroup.subtree_control`
- `echo +memory > /sys/fs/cgroup/system.slice/group.subtree_control`
- Acctounting happens only at allication time!
- Restart service

Memory Usage Inside the Kernel

- No global counter
- Use total memory usage minus sum of Pss as rough estimate
- `/proc/meminfo` has some handy metrics
- `Slab`
- `VmallocUsed`
- `CmaTotal` and `CmaFree`

Memory Usage Inside the Kernel (cont'd)

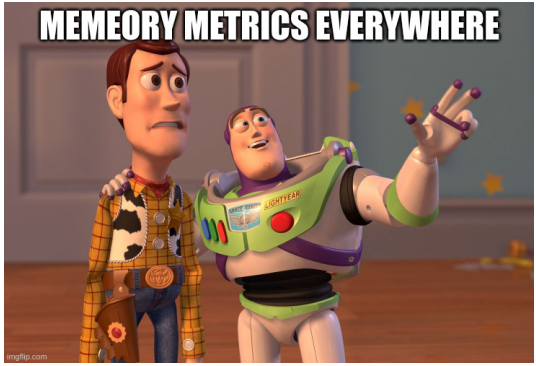
- SLAB has many instances (AKA pools)
- `/proc/slabinfo`
- Often per driver/subsystem
- Calculate `num_objs * objsize`
- Useful to locate memory leaks

Useful Tools

- smem
- slabtop
- collectd

Summary

- Don't panic
- Pss from `/proc/PID/smaps_rollup` for processes
- CGroup's `memory.current` for cgroup'ed processes
- `Slab`, `VmallocUsed` from `/proc/meminfo`



FIN



Thank you!

Questions, Comments?

Richard Weinberger
richard@sigma-star.at

https://sigma-star.at/quiz/embedded-security/