THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

# Memory Management 101: Introduction to Memory Management in Linux

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

@qant

THE LINUX FOUNDATION

## 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*"

THE LINUX FOUNDATION

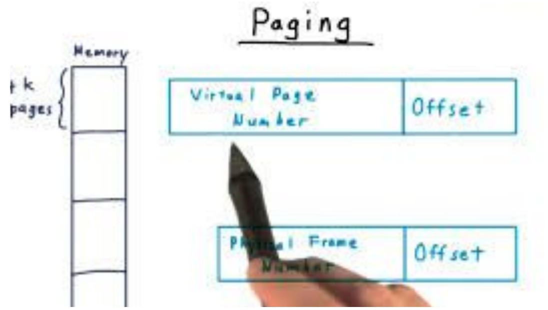
# Pages and physical page frame numbers

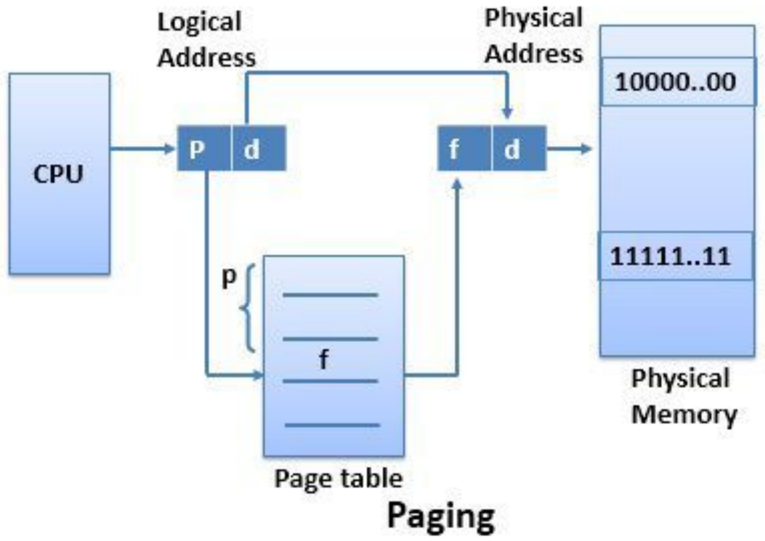
- Division of memory into “pages”
  - 1-N bytes become split at page size boundaries and become M = N/page size pages
- Refer to memory by the Page Frame Number (PFN) and an offset into the page.
- Common size is 4k (Intel legacy issues)
- MMU creates virtual addresses.

THE LINUX FOUNDATION

# Basics of “paging”

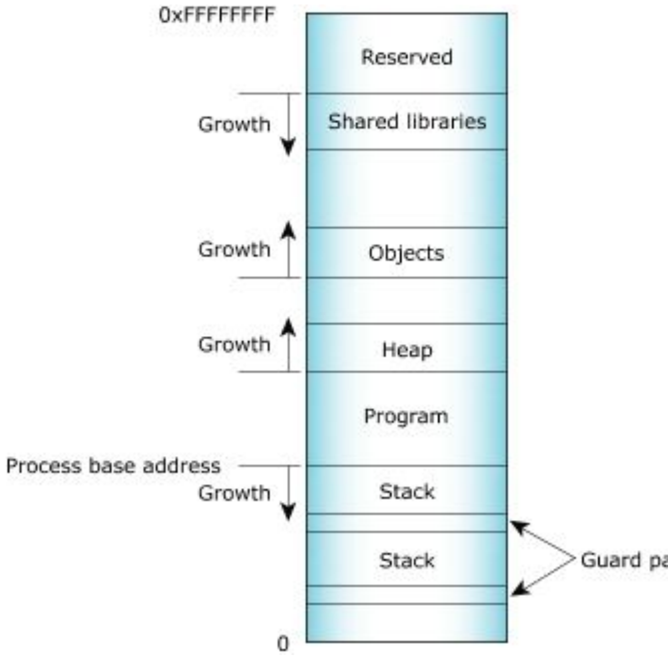
- Process have virtual memory
- -> PFN
- Page Tables
- Faults
  - Major
  - Minor
- Virtual vs physical





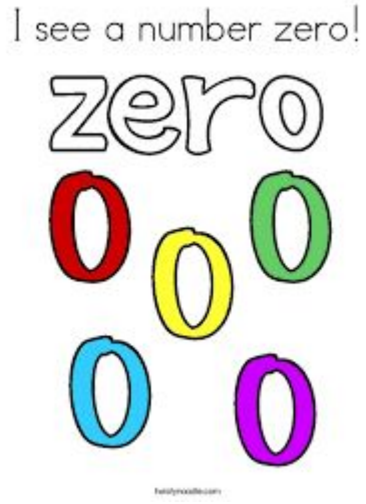
# Process memory

- 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.

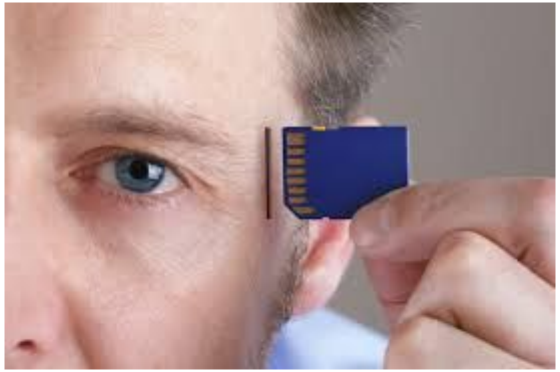


## Linux Basic memory information

/proc/meminfo

Memtotal

<go through fields>



## Process memory use

/proc/<pid>/status

<Discuss select fields>

info is shown via

ps, top and various monitoring tools.

## User limit (ulimit)

- max memory size
- virtual memory
- etc

# Overcommit configuration

Virtual memory use vs physical

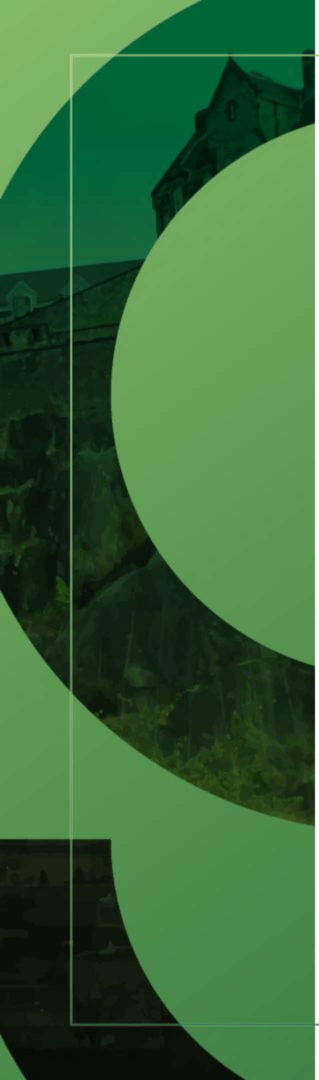
overcommit_kbytes

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



## Questions / Comments

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

THE LINUX FOUNDATION

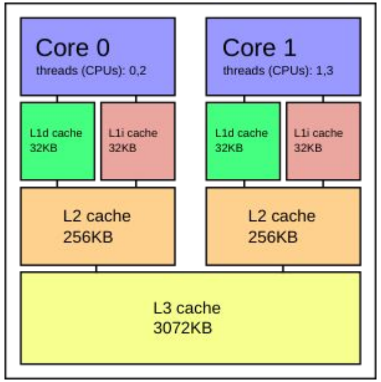
# OPEN SOURCE SUMMIT

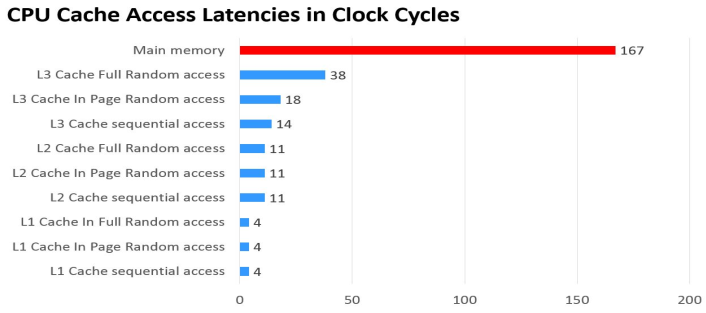
EUROPE

THE LINUX FOUNDATION

# “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





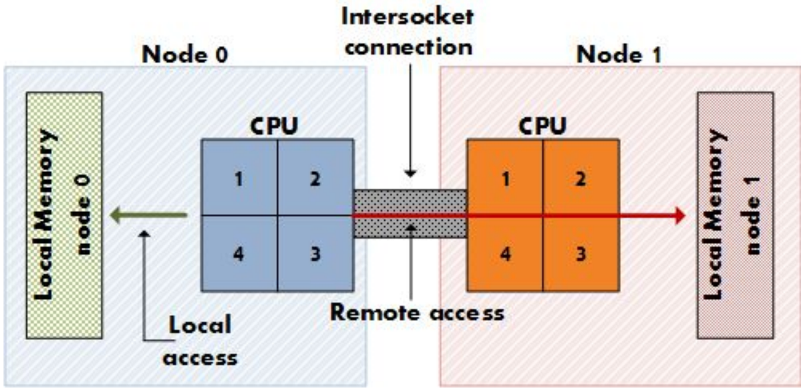
# NUMA Memory

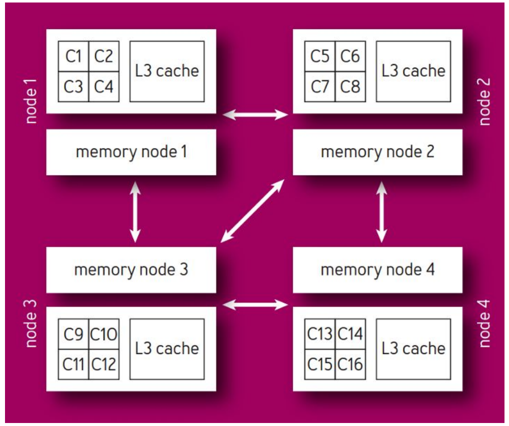
- Memory with different access characteristics

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

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

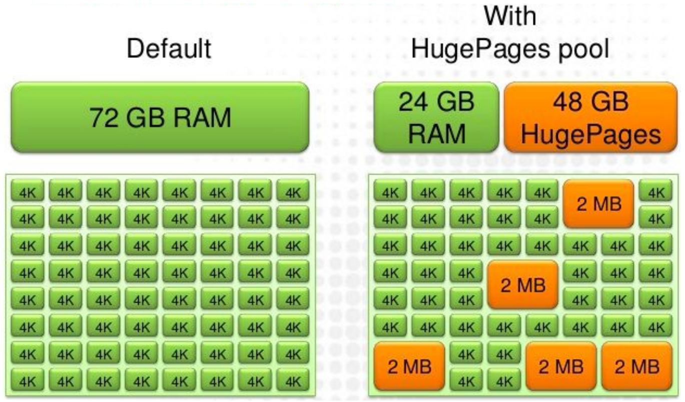
- Automatic memory migration

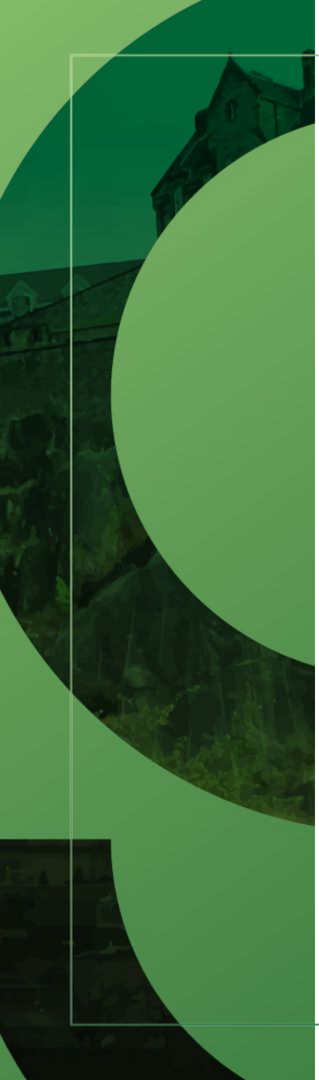




# Huge Memory

- 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