The Multics Virtual Memory: Concepts and Design


Presented by: Nuno Santos

Advanced Topics in Computer Systems (I)
Fall 2007 - 1-Oct-07
Outlook

• Introduction and Historical Context
• Presentation of Paper
• Analysis of Paper
• Discussion Topics
Introduction and Historical Context
A Note on Terminology

• Paper is 35 years old. Terminology has evolved.
• I’ll use modern terminology.
  – Some examples of modern names and corresponding Multics names:
    • Main memory – core memory
    • Address space of process - Core image of process
    • Kernel – Supervisor.
      – Although supervisor still used...
    • Memory-mapped files – not explicitly named on paper
Short Multics History

• Development started in 1962
  • Commercial project by General Electrics and MIT
    • GE computer division later acquire by Honeywell
  • Inspiration for Unix.
    • Developers of Unix came from Multics.

• Goals of project
  • 24x7 operation - High reliability
  • Secure by design

• Hardware: GE 645 computer
  – Supports multiple processors, disk and memory units.
  – On-line reconfiguration
  – 36-bit word.
Context/State of the Art (60s)

• First multiprogrammed computers
  – Large, bulky, expensive and slow
  – Memory was very limited
  – Multi-user
  – Remote access
Virtual Memory Before Multics

• The basic mechanisms of virtual Memory already existed
  – Simulate a large address space on top of limited memory
    • B5000 computer – Segmentation
    • Atlas – Demand Paging
Presentation of Paper
Multics Virtual Memory Challenge Addressed

• The problem:
  – Experience shows that “many applications require the rapid but controlled sharing of information stored on-line”

• But existing systems had limited support for sharing
  – In most systems: sharing via file system – save shared information on a file, read to memory of each user, write back to file
    • Slow and complex.
  – Non-segmented systems
    • Uniform address spaces, no access control information
    • Even if sharing is implemented, access control is hard.
Multics Virtual Memory
Design Goals

• **Sharing and protection** of information in a transparent way.
  – All information stored in memory must be directly addressable by any process
  – Access control is performed at each reference to memory
Multics Virtual Memory Concepts
Segmentation

- **Segment** – Unit of sharing and protection
  - Address space of processes divided into segments
    - **Sharing** - Can be mapped to the address space of several processes
    - **Protection** – associated access rights. Checked by hardware at each reference

- **Attributes**
  - Base address and length
  - Name
  - Owner
  - Access rights – list of (user, access rights) pairs.
Multics Virtual Memory Concepts

Segmentation on Multics

• **Large number of segment descriptors** available to each process
  – Enough for all needs, managed transparently by OS.
  – Previous systems were limited, required explicit management by programmers.

• **Segmentation closely integrated with filesystem**
  – **One-to-one mapping between files and segments**
    • Segment names are pathnames into the directory system
  – Memory-mapped files – To access a file, the system maps it into a segment.
    • No other mechanism for accessing files. i.e., no direct access.
Multics Virtual Memory Concepts
Paging

- Motivation for paging: provide large virtual memory using swapping
- Swapping with segments?
  - Possible, but not practical - Segments are variable sized
    - Complex to manage on backing store
    - Poor granularity with large segments.
      - Locality of access is not exploited
      - Limits number of segments that fit on memory
- Paging – break down segments into equal-size parts
  - Some internal fragmentation
  - But much simpler and more efficient
    - Simplified space allocation on backing store
    - Only the referenced pages of a segment need to be in memory
    - Demand paging – load into memory only the parts of a segment needed
    - A segment can be larger than the physical memory
Multics Hardware
The Honeywell 645 processor - Segmentation

• Segmentation and paging need hardware support
• Memory address: \([s, i]\)
  – s – Segment number (Max 256K segments)
  – i – index within segment (Max 256K words)
  – Max size of segment – Approx 1MB (36 bits words)

• **Descriptor Segment (DS) table** – In main memory. List of **Segment Descriptors Words (SDW)**.
  – Each SDW contains
    • absolute base address of page table of segment
    • length of segment
    • access rights
    • missing segment flag
Multics Hardware
The Honeywell 645 processor - Paging

• Each segment is paged
  – Page size – 1024 words = 4KB

• Page Table (PT) – per segment table. Array of Page Table Words (PTW).
  – Each PTW contains:
    • Absolute address of page
    • Missing flag

• Processor’s TLB (Translation Lookaside Buffer) - cache PTW and SDW entries in fast associative memory
Multics Kernel

• Segmented and partially paged
  – Use same conventions as for user programs
  – Parts of kernel do not need to be in memory
    • Note: the Multics kernel was large for the systems of the time.

• Kernel is shared between processes
  – Mapped to the address space of each process
    • A process can call the kernel directly
    • At the time, most kernels ran in a separate process or address space
  – Kernel protected by a ring protection mechanism
    • Not discussed on the article.
Segment Attributes Management

- Information on segments is stored on the filesystem
  - Each segment is a file. Each file is a segment.
- Segment attributes on filesystem
  - Symbolic Name – a pathname
  - Length
  - Memory address
  - ACLs
  - Creation date and time
- Segment creation
  - Users provide name and ACLs.
  - Segment initially inactive – no page table, no storage allocated
Segment Lifecycle

- Processes refer to segments by symbolic names
  - Must be mapped to segment numbers on first reference

- **Known Segment Table (KST)**
  - Per-process table
  - KST(segment number) = symbolic name

- On first reference to segment by a process
  - Search for segment name on directories
  - Assign an unused number s in the process
  - Add entry to KST of process
Activation and Connection of Segments

- **Activation**
  - A segment is active if it has a page table
  - **Active Segment Table (AST)** – system wide table.
    - Contains Page Table address for each segment.
  - Activation creates a page table for the segment

- **Connection**
  - Connect an active segment to a process
  - Segment information added to the **Segment Descriptor Word** of process.

- **Note on life cycle of a segment**
  - Segment “swapping” - A segment can be deactivated and disconnected while still referenced by a process.
  - Later, it can be reactivated and reconnected.
  - Used for resource management
Managing Limited Memory Swapping

- **Page Multiplexing**
  - Physical memory not enough to hold all pages
  - Page swapping done on fault handling:
    - If number of free frames is below a threshold
  - Replacement algorithm: *least-recently-used.*
    - With hardware support: used bit.

- **Active Segments Multiplexing**
  - AST is not large enough to hold all possible segments.
  - Replacement policy: *least-recently-used segment*
    - No pages in memory for the longest time
  - Victim is deactivate and disconnected
Structure of Kernel

• Divided into three modules
  – Directory Control (DC) – operations on segment attributes.
  – Segment Control (SC) - segment fault handling.
  – Page Control (PC) – page fault handling.

• Paging of kernel
  – PC is always resident
  – Parts of DC and SC can swapped
Conclusions of Paper

• Multics Virtual Memory main features
  – Sharing and protection using segments
  – Segments are paged and subject to swapping
  – Tight integration with file system: segments and files are equivalent

• Multics Kernel
  – Also segmented
  – Mapped to address space of processes
Analysis of Paper
Overview

• Good points
  – Clear and detailed description of mechanisms

• Bad points of paper
  – No experimental validation
    • What is the overhead of the Multics virtual memory implementation?
    • Was the hardware of the time powerful enough to support these mechanisms?
    • How effective is the security?
    • Users reaction to sharing mechanism?
Contributions

• Easy to identify:
  – 35 years of research and experience with operating systems make it easy
  – Use modern OSs for comparison
    • What ideas are still used?
    • What ideas were abandoned?
Contributions of the Paper

• Segmentation and paging
  – Still the mechanisms used on modern operating systems for sharing and protection
  – Current implementations differ somewhat, but are conceptually similar

• Memory-mapped files
  – Widely used by modern operating systems.
Where Multics got it Wrong

• Integration between filesystem and memory management
  – Multics went too far
  – Segment size (1MB) was a limit to file size
    • Multi-segment files added later to Multics. An ugly hack
  – Some applications require direct access to files
    • Later added to Multics
Discussion Topics
Relation between Paging and Segmentation

• On Multics, paging is done on a per-segment basis
• Modern OSs do paging below segmentation:
  – Large virtual address using paging
  – Segmentation on top of virtual address
• Which approach is better?
  – Probably the one of modern operating system
  – But why?
• Multics approach
  – Page tables are small. No need to use hierarchical page tables.
  – But segments must be activated/deactivated
Alternatives to Paging and Segmentation

• These mechanisms are widespread
• Are they the ultimate solution for sharing and protection?
• I don’t have any idea...
Operating Systems
35 years from now

• It’s surprising how much the kernel of modern OSs resemble Multics.
  – Although much has changed above the kernel

• Will OSs in 35 years have the same structure as current OSs?
Why Multics failed?

• Multics is dead.
  – Last installation was stopped in 2000
  – No direct descendent

• Technical reasons?
  – Too complex?
  – Too big?
  – Too slow?
  – Ahead of its time?
  – Did the Multics VM design had any influence on Multics death?

• Or perhaps Multics didn’t fail
  – Inspiration for Unix