# M-M/S-CD Memory Management Conceptual and System Models


Yauhen Klimiankou

klimenkov@bsuir.by

Evgeny.Klimenkov@gmail.com

Belarusian State University of Informatics and Radioelectronics


.

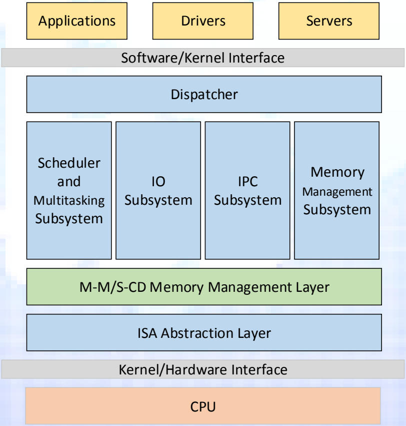
**ISPRAS Open 2017**

1th December 2017

# Introduction

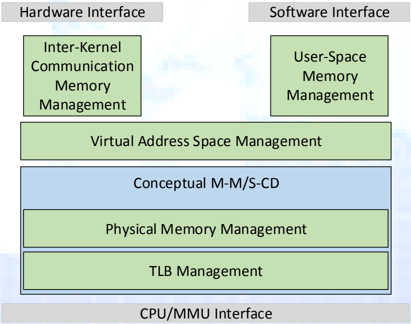
M-M/S-CD is a new approach to the memory management design.

- Designed for application in true microkernels and OS architectures based on them.
- Implemented for IA-32 platform.
- Implemented as a set of well-abstracted modules in C++.
- Goals: minimalism, simplicity, efficiency, flexibility, usability.

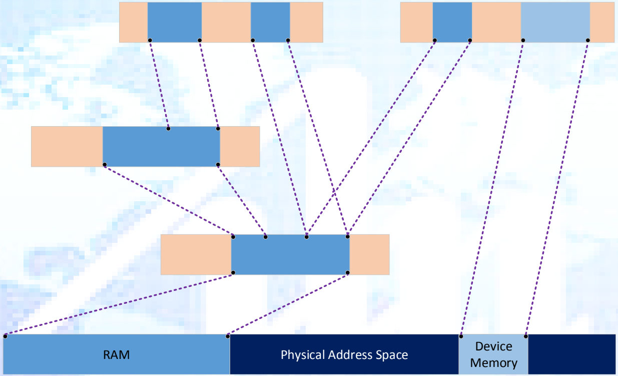


# Motivation

- There are a variety of different types of computer systems: starting from embedded computer and ending by high-performance servers in supercomputer clusters.
- We had a desire to obtain standardized, flexible and powerful memory management architecture as part of framework for kernels design.

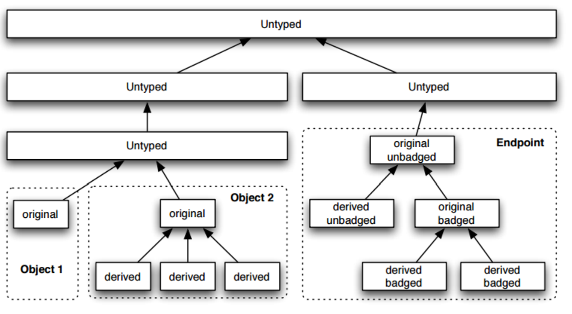


# Recursive Address Space Construction



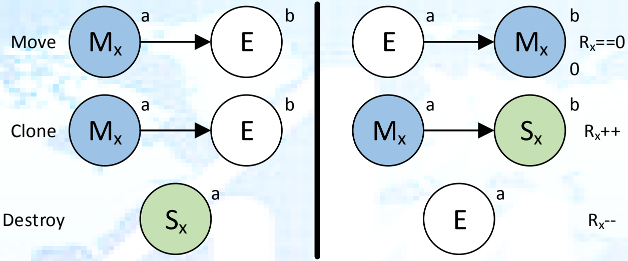

- Operations: map, grant, flush.
- Significant cost in kernel complexity and memory overhead.
- Hierarchical dynamic tree-like data structures.
- Problem of kernel memory exhaustion.
- Legacy approach.

# Capability-Based Memory Management




- Capability-mediated operations (copy, delete, retype, revoke).
- Reduced but still significant cost in kernel complexity and memory overhead.
- Still hierarchical dynamic tree-like data structures (Capability Derivative Tree + Mapping DataBase).
- Explicit kernel memory management.
- State-of-the-art approach.

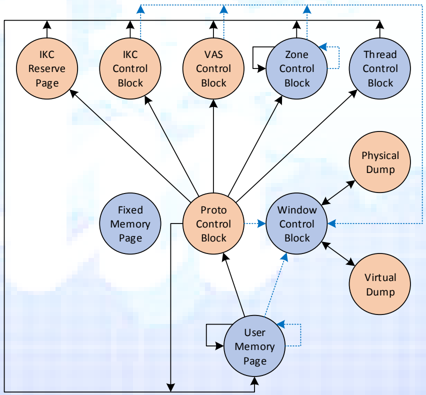
# M-M/S-CD Memory Management: Conceptual Model




- Operations: Move, Clone, Destroy.
- Strictly bounded CPU cost, in-place memory management, 1500 lines of source code that compiles into about 1KB of binary code.
- Flat model, strict roles separation, reuse of page tables used by MMU.
- Consistency guaranteed by movable-only references.
- Explicit kernel memory management.

# M-M/S-CD Memory Management: System Model

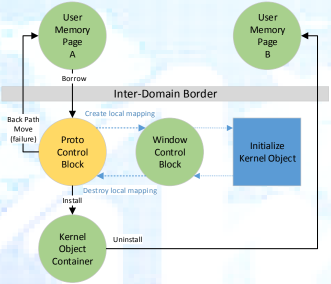
- Pure conceptual model is enough for true microkernel, but inconvenient, leads to suboptimal performance and not suitable for untraditional OS architectures.
- System model provides extended and optimized variant of M-M/S-CD.
- System model defines dynamic kernel objects and transitions between them.



# M-M/S-CD Memory Management: System Model

- Knowing the expected usage of the object we can enforce expected semantics and optimize the transitions.
- There are two families of containers: physical and virtual. Physical containers are shared with MMU, TLB management is omitted for virtual containers.
- User Memory Page fully reproduces semantic of the container of conceptual M-M/S-CD model.
- In contrast, Fixed Memory Page Prohibits all operations on container.

# Proto Control Block




- Assures isolation between kernel space and user space.
- Eliminates intra-kernel race conditions.
- Proto Control Block is a system invocation local.
- Modified semantics: move_in&&lock(UMP) + move_out&&unlock(UMP) + move_to(kernel object container)
- Destroyed kernel objects move back to user space without clean up (information can be returned from the kernel).

# Basic Dynamic Kernel Objects

- There are three basic kernel objects: thread, virtual address space and zone.
- Thread Control Blocks support only move semantic and support container ownership. Thread Control Blocks are an accountable system resource, which makes thread creation simple, efficient, deterministic and predictable.
- VAS Control Block has an additional operation – activation. Cloning is supported between VCBs. Collective mastering. A concept of single "main" thread is not enforced by M-M/S-CD.
- Zone presents a fixed size region of VAS. Has modified reference counting policy. Serves as a mean of delegation of memory management rights to the external pager. Paging can be performed transparently to the client.

# Auxiliary Dynamic Kernel Objects

- M-M/S-CD enforces closeness of the memory model.
- To allow dynamic inter-kernel memory exchange in multikernel OS, M-M/S-CD should be extended by a way of controlled violation of closeness.
- Slave reference counters table is not static in that case.
- Two additional types of containers are added to support injection and extraction of memory pages to and from physical memory pool managed by kernel, and for extension/shrinkage of counters table.
- Window Control Block implements a form of Thread Local Storage.
- It serves for temporal isolation of memory from the rest of the system.
- It also serves for inter-address space access to the data. (for example for asynchronous message passing)

# Evaluation and Comparison

| Memory management operation | M-M/S-CD inter-space [ticks] | M-M/S-CD intra-space [ticks] | seL4 (copy&map) [ticks] | seL4 (dup&map) [ticks] |
|---|---|---|---|---|
| Clone | 562 | 562 | 17303 | 18421 |
| Destroy | 512 | 1100 | 31171 | 31909 |
| Move | 573 | 573/1166 | 48382 | 50309 |

| System call | execution time [ticks] |
|---|---|
| seL4_CNode_Copy | 5560 |
| seL4_CNode_Delete | 17605 |
| seL4_x86_Page_Map | 11508 |
| seL4_x86_Page_Unmap | 13225 |

# Evaluation and Comparison

- Both kernels are sysenter/sysexit based.
- 1 system call in M-M/S-CD vs 2 system calls in seL4.
- Heavy dispatching in seL4 (segment registers reloading) (1600 ticks) vs fast and simple dispatching in our kernel (completely flat memory model).
- Hierarchical tree-like basis of CBMM involves high processing cost vs fixed-cost and simple processing in flat model.
- Support of deep revocation is looks like valueless, but still supported.
- At the same time advanced paging support is valuable, but still is unsupported.
- No more details... at least now. Hacking of seL4 is a pain :(.

# Conclusions

- M-M/S-CD is minimalistic and simple.
- M-M/S-CD + ISA abstraction layer + dispatcher = well abstracted framework for microkernel design.
- M-M/S-CD introduces tiny memory overhead.
- And significantly outperforms at least CBMM in terms of CPU overhead.
- Plus it provides completely predictable and deterministic operations.
- 
- Thus... It is our believe that it is a promising alternative for widely-used CBMM :(.

# Questions?