Jin-Soo Kim
(*jinsoo.kim@snu.ac.kr*)

Systems Software &
Architecture Lab.

Seoul National University

Spring 2026

Opal

(Jeffrey S. Chase et al., ACM TOCS 1994)

The 64-bit Address Space

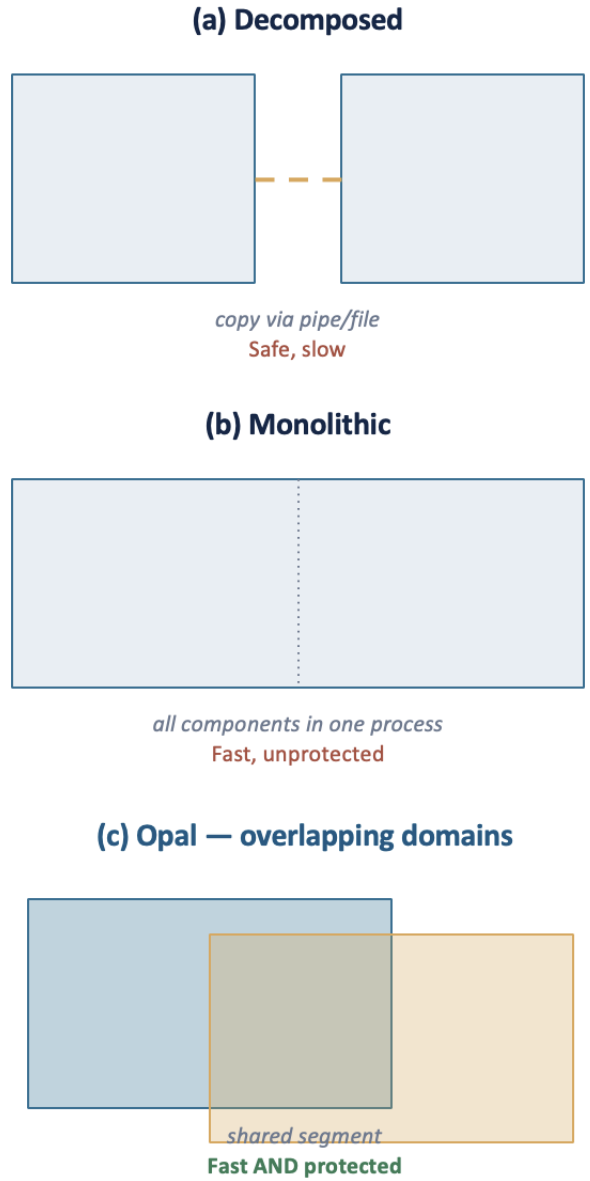
- MIPS R4000 (1991), DEC Alpha (1992), Sun UltraSPARC (1995), …
  - _____ years to fill a 64-bit address space at 1 GB/s allocation
  - Address space is no longer a scarce resource to recycle
  - Each address can have a globally unique, unambiguous meaning
  - Opportunity: revisit assumptions inherited from Unix-like OS design

- What if every process saw the same 64-bit address space?
  - Simpler sharing
  - Cheapter protection
  - More flexible cross-process data structures
  - Target workloads: integrated, pointer-rich applications (CAD/CASE, OODBs, etc.)

Private Address Space

- Pros
  - Amplify available address space across many processes
  - Strong hardware-enforced protection boundaries
  - Automatic cleanup on process exit

- Cons
  - Pointers lose meaning across process boundaries
  - Communication via pipes, files, or messages:
    – slow due to serialization and copying overhead
    – loss of pointer semantics and sharing structure
  - mmap-based sharing requires a priori address agreement
  - Putting everything into one process sacrifices protection



Sharing Patterns

- Read-only sharing
  - Pointer-rich data structures shared at wire speed without copy
  - Writers isolated in a separate domain
- Asymmetric trust
  - Producer untrusted by consumer – but producer's output visible read-only to consumer through a shared segment
- Hand-off
  - Pass a segment sequentially from domain to domain; re-attach/detach moves rights without copies
- Intra-app isolation
  - Split one app. into several domains to contain bugs or enforce per-user privileges

Opal

- A Single-address-space OS
  - Virtual addresses are context-independent
  - A virtual address resolves to the same data, no matter who dereference it
  - A thread has the right to access only a subset of those addresses
  - Addressability ≠ Access

- Approaches
  - Separate addressing from protection
  - Segments as the unit of sharing/protection
  - Protection domains and capabilities for protection and isolation
  - Portals for protected cross-domain calls

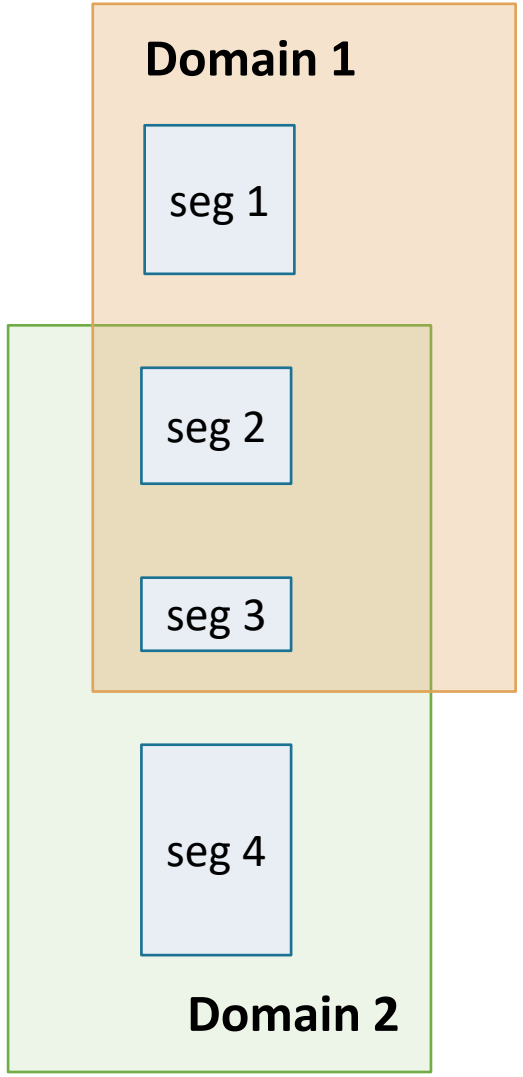
Segment

- Contiguous extent of virtual pages (min. 1 page)
- Unit of memory allocation and protection
- Permanent virtual address assigned at creation
- May be persistent
  - Backed by storage, survives restart

- Interfaces
  - Attach (≈ mmap)
  - Detach (≈ munmap)
  - Segments can be attached transparently on address faults

Protection Domain

- The subject of memory access control
  - An execution context for threads
  - A thread executes in exactly one domain
  - Many threads may execute in the same domain
- A set of attached segments with per-segment rights
  - Domains may overlap on segments
  - May threads may share a domain
- Enforced by standard page-based protection hardware
- No notion of "running a program"
  - New domain creation + Attach segments + call to portal



Portal

- Named entry point to a protection domain
  - Unique 64-bit portalID
  - Parents create children domains and install portals to call in selectively
  - A thread enters the target domain by calling through the portal
  - Portal invocation switches protection domain, not address meaning
  - Arguments can be passed as ordinary pointers in the shared address space
  - Can be used to implement system calls

- Building block for protected RPC (remote procedure call) across domains

Capability

- An unforgeable token for naming an object and carrying access rights
  - 256-bit token that names an object and carries access rights for specific operations

  - An object can be a segment, domain, thread, portal, or resource group
  - Randomized check field verifies authority to operate on the named object: prevents forgery and supports revocation

- Globally meaningful and freely shareable
  - Passed directly through shared memory – no kernel context needed

Issues

- UNIX fork()?
- Hardware support?
- Virtual contiguity?
- Address recycling?
- Copy-on-write?
- Private writable static data?
- Address randomization?
- …