Technische Universität Hamburg

Leibniz Universität Hannover

Morsels: Explicit Virtual Memory Objects

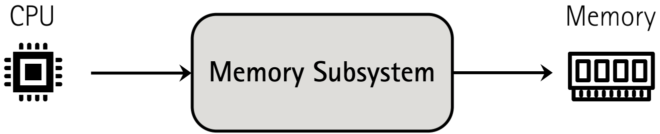
DIMES '23

Alexander Halbuer   Christian Dietrich   Florian Rommel   Daniel Lohmann

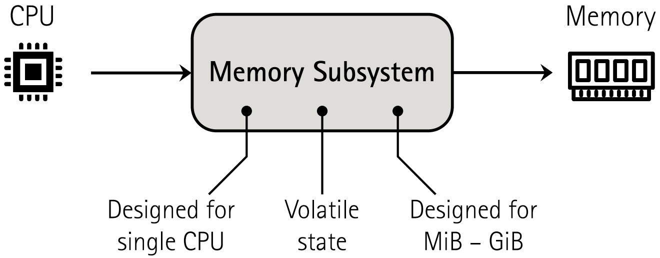
Leibniz Universität Hannover
halbuer@sra.uni-hannover.de

October 23, 2023

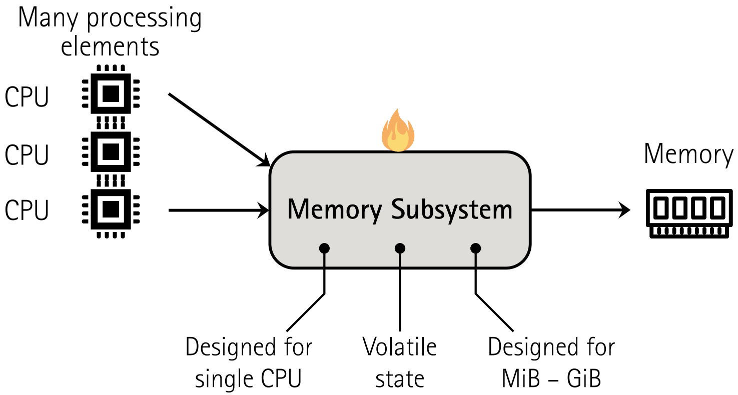
New Challenges for Virtual-Memory Management



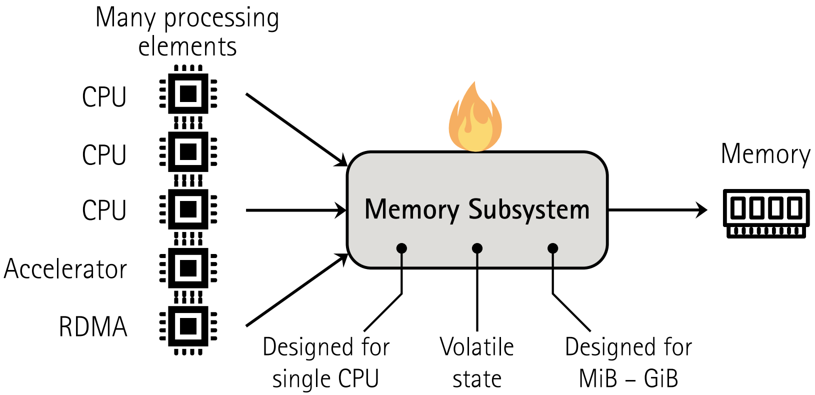
New Challenges for Virtual-Memory Management



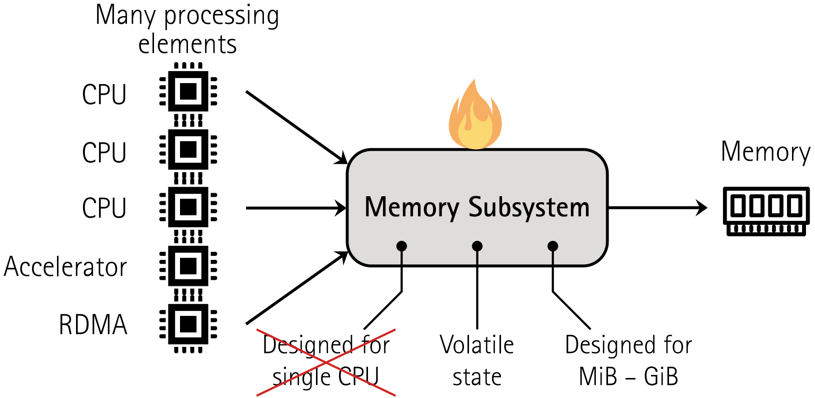
New Challenges for Virtual-Memory Management



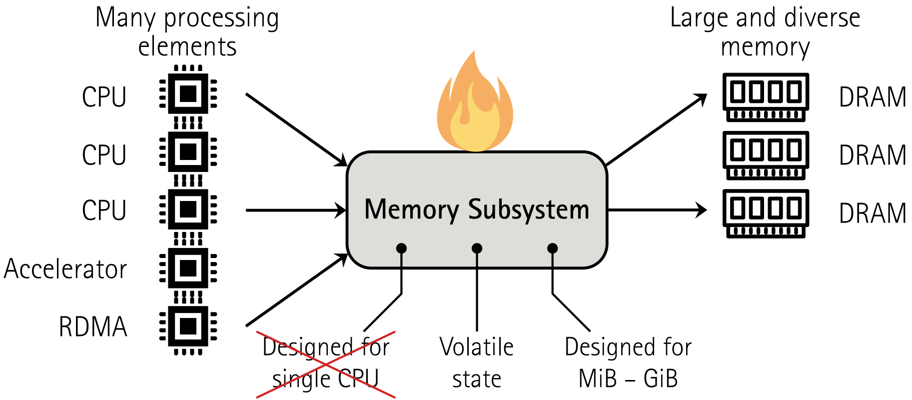
New Challenges for Virtual-Memory Management



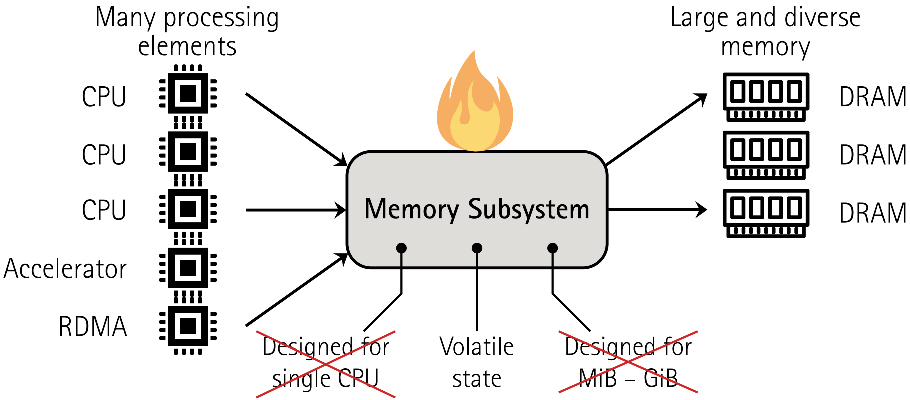
New Challenges for Virtual-Memory Management



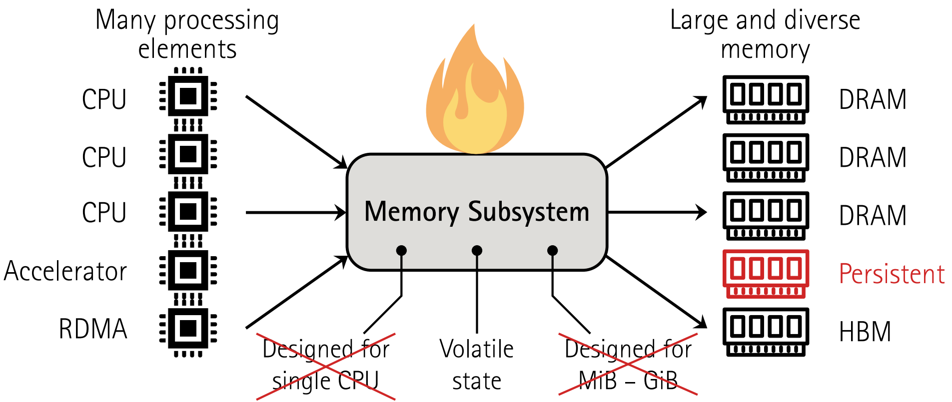
New Challenges for Virtual-Memory Management



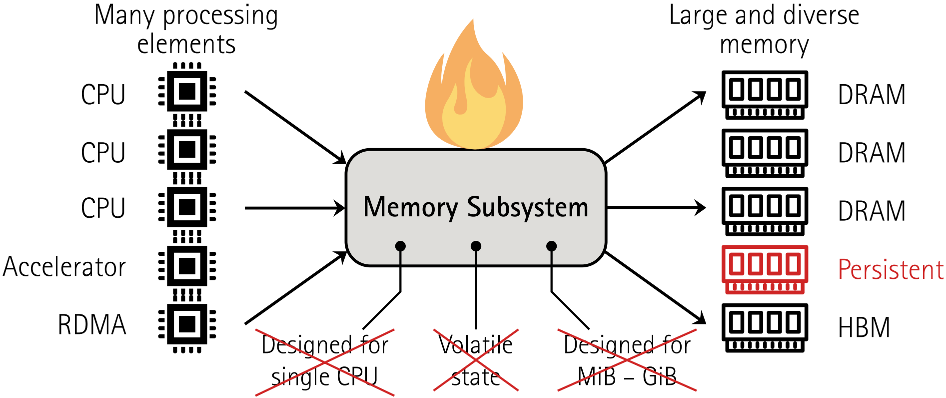
New Challenges for Virtual-Memory Management



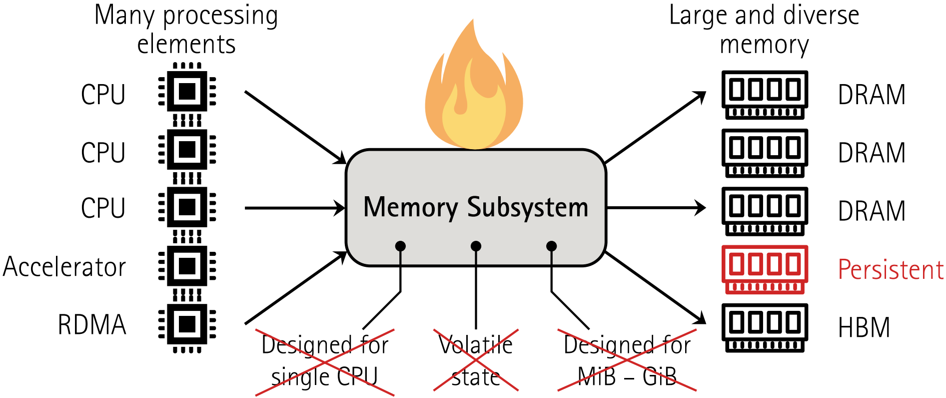
New Challenges for Virtual-Memory Management



New Challenges for Virtual-Memory Management

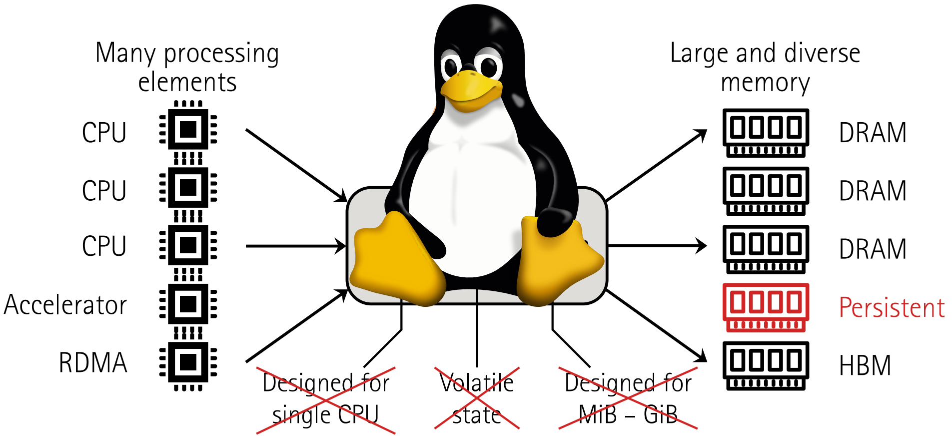


New Challenges for Virtual-Memory Management



What are efficient OS abstractions for large memory objects?

New Challenges for Virtual-Memory Management



What are efficient OS abstractions for large memory objects?

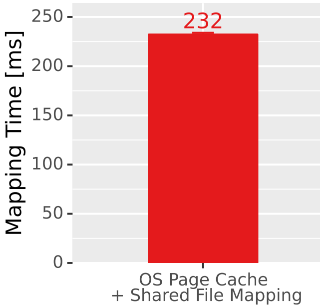
Motivation

Goal: Efficient handling of virtual memory

Motivation

Goal: Efficient handling of virtual memory

Mapping ∼7 GiB of data into an application's address space

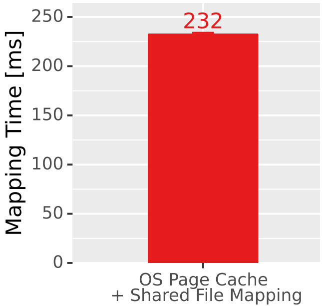


Motivation

Goal: Efficient handling of virtual memory

- Conventional file mapping (pre-faulted)
  - High setup costs
  - Create new page tables
  - Populate with pages from page cache

Mapping ∼7 GiB of data into an application's address space



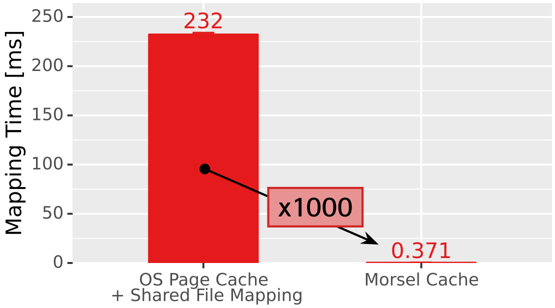
Motivation

Goal: Efficient handling of virtual memory

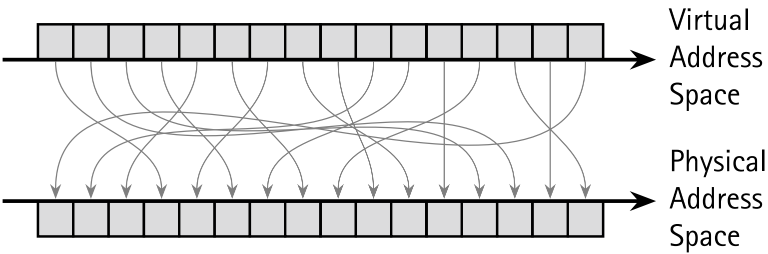
- Conventional file mapping (pre-faulted)
  - High setup costs
  - Create new page tables
  - Populate with pages from page cache

Our solution: Morsels

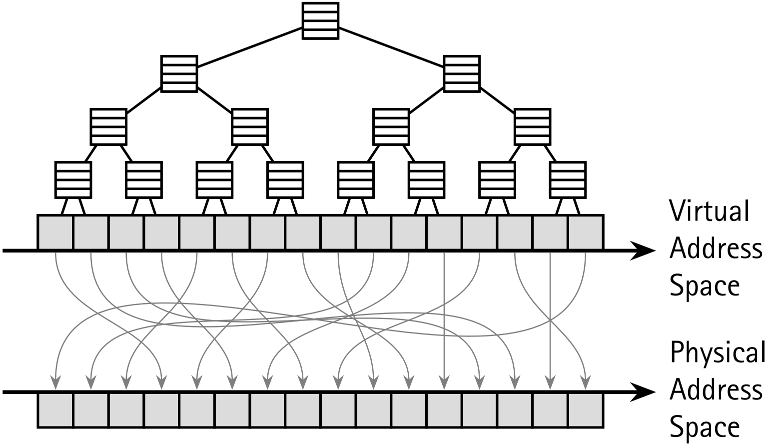
Mapping ∼7 GiB of data into an application's address space



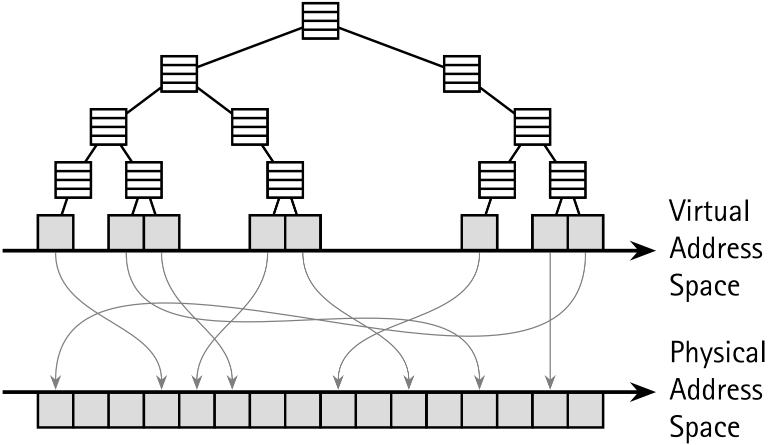
Memory Objects – Morsel Concept



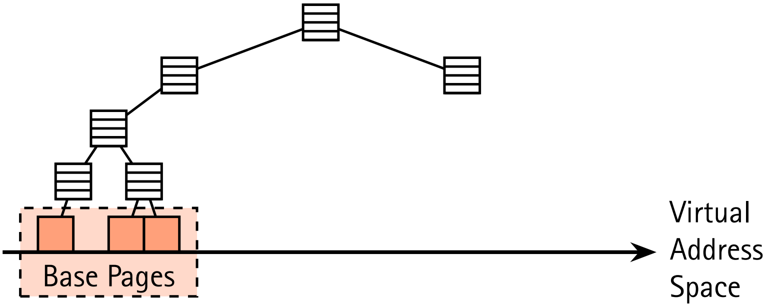
Memory Objects - Morsel Concept



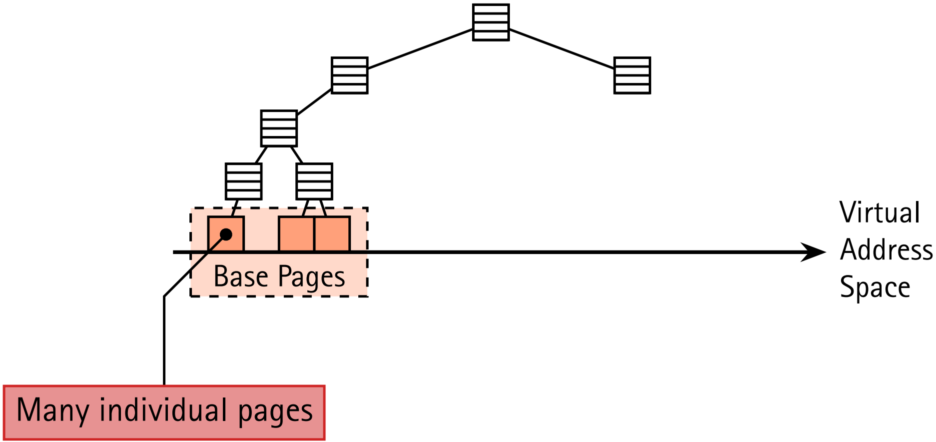
Memory Objects - Morsel Concept



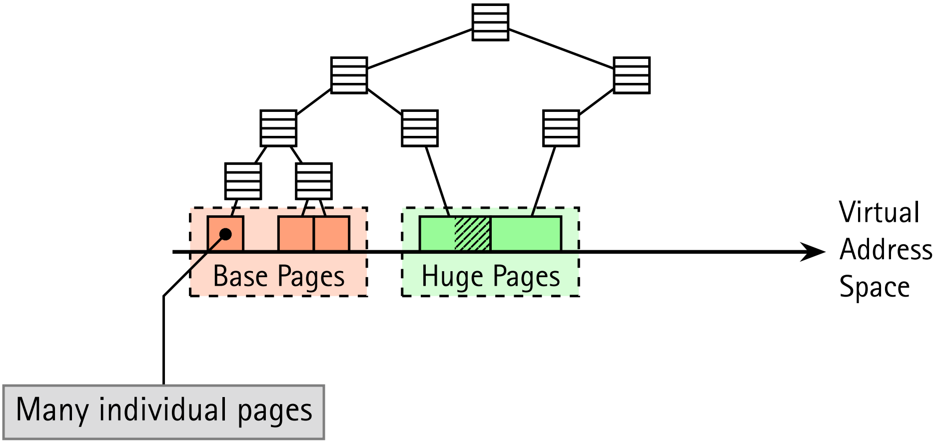
Memory Objects - Morsel Concept



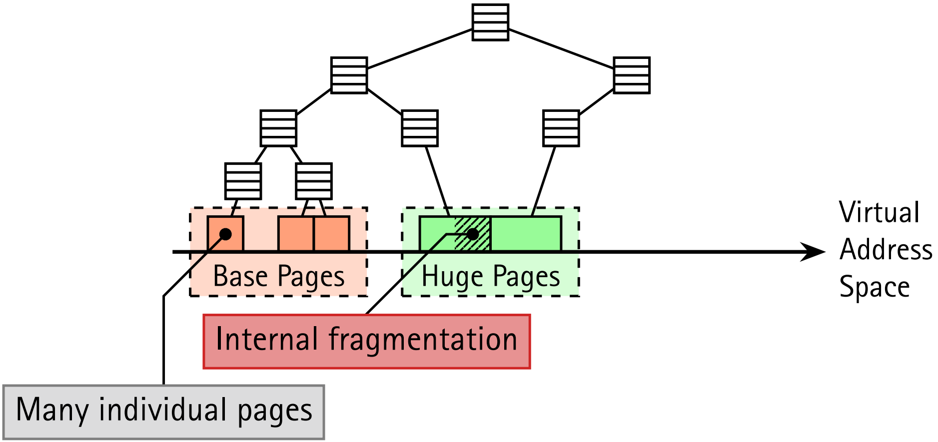
Memory Objects - Morsel Concept



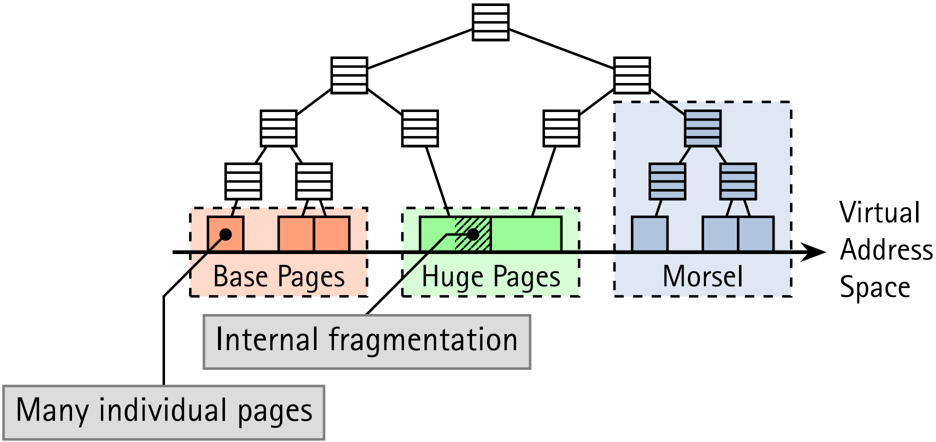
Memory Objects - Morsel Concept



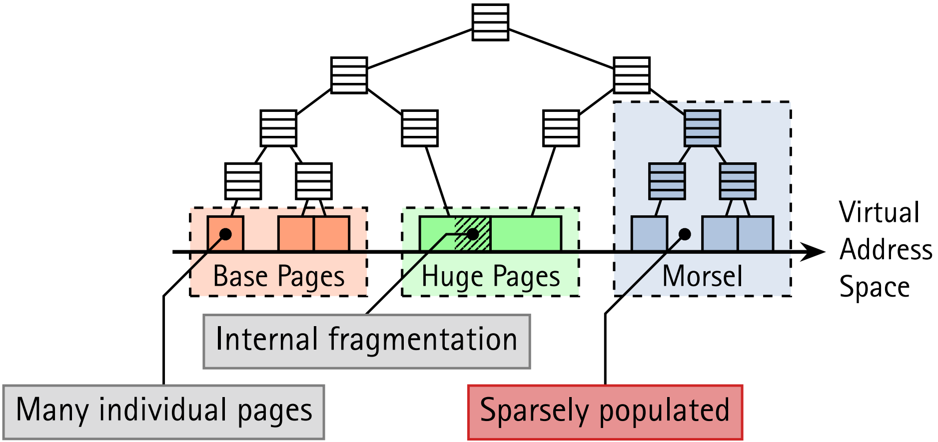
Memory Objects – Morsel Concept



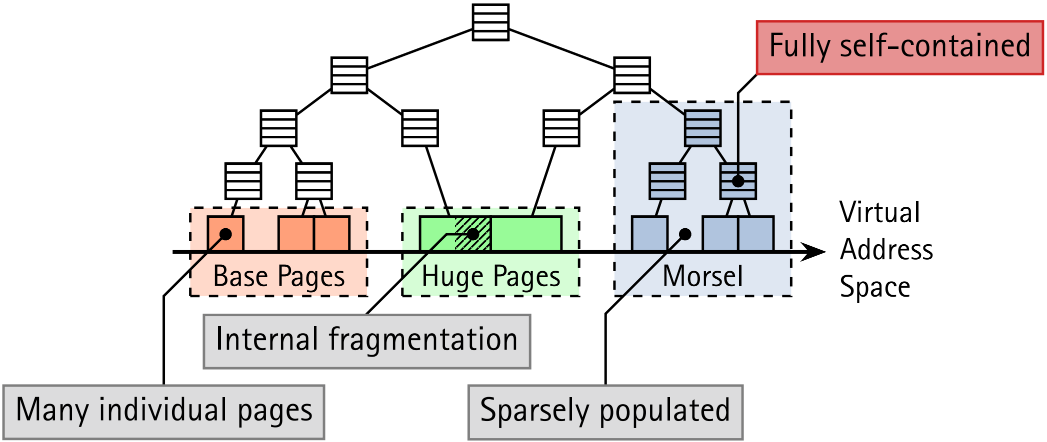
Memory Objects – Morsel Concept



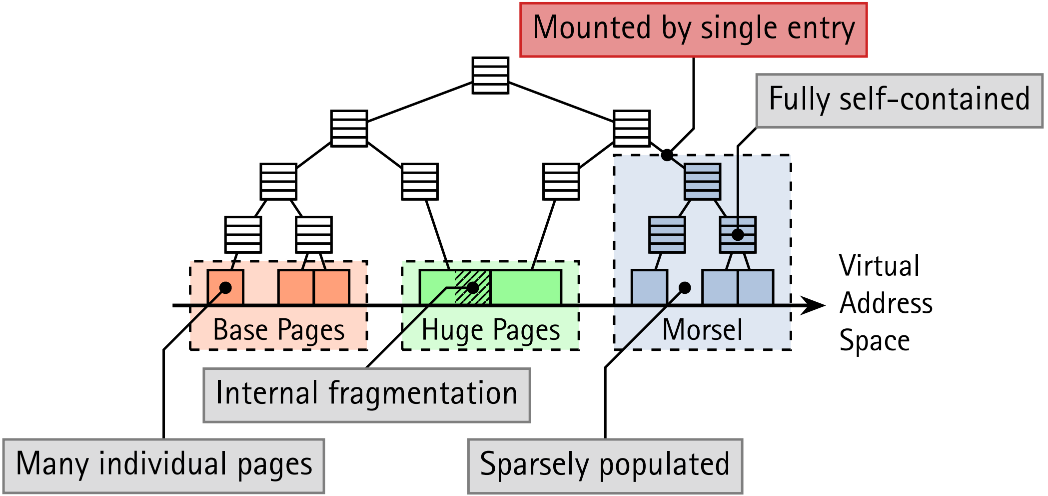
Memory Objects - Morsel Concept



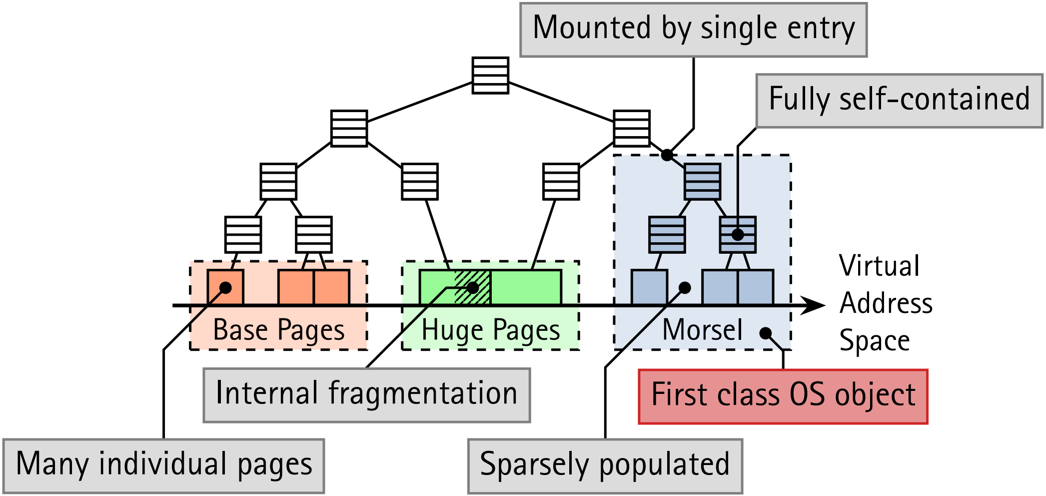
Memory Objects – Morsel Concept



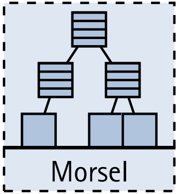
Memory Objects – Morsel Concept



Memory Objects – Morsel Concept

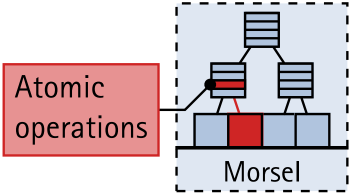


Memory Objects – Morsel Concept



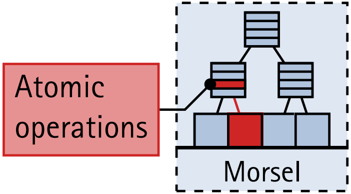
Memory Objects - Morsel Concept

- Good multi-core scalability
  - No locking required



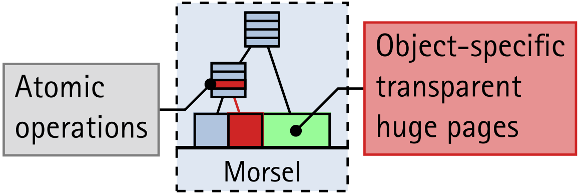
Memory Objects - Morsel Concept

- Good multi-core scalability
  - No locking required
- Persistent objects on NVM
  - Fully self-contained design
  - Crash consistent implementation



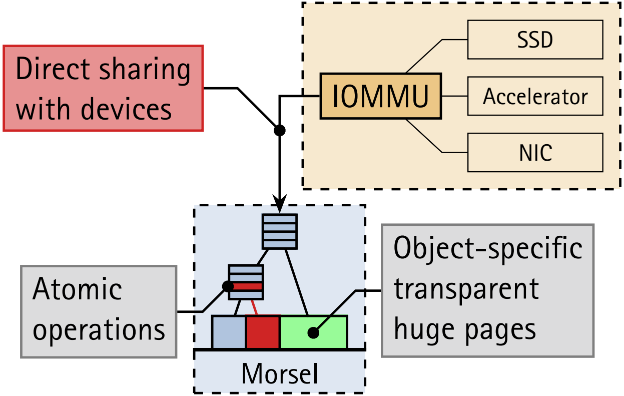
Memory Objects - Morsel Concept

- Good multi-core scalability
  - No locking required
- Persistent objects on NVM
  - Fully self-contained design
  - Crash consistent implementation
- Huge-page support
  - Increase TLB coverage
  - Transparency on object granularity



Memory Objects - Morsel Concept

- Good multi-core scalability
  - No locking required
- Persistent objects on NVM
  - Fully self-contained design
  - Crash consistent implementation
- Huge-page support
  - Increase TLB coverage
  - Transparency on object granularity
- Sharing with devices
  - Reduce management overhead for DMA



Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

- Conventional file mapping (pre-faulted)
  - Uses individual 4-KiB pages
  - Populated from the page cache
  - Time increases with mapping size

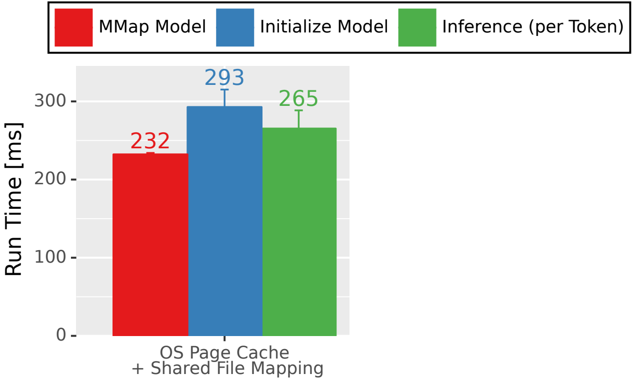
Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

- Conventional file mapping (pre-faulted)
  - Uses individual 4-KiB pages
  - Populated from the page cache
  - Time increases with mapping size

`llama.cpp` with SelFee model: 6.82 GiB



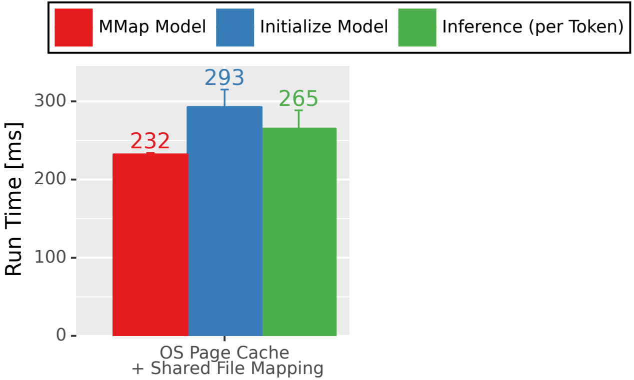
Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

- Conventional file mapping (pre-faulted)
  - Uses individual 4-KiB pages
  - Populated from the page cache
  - Time increases with mapping size
- Morsel mapping
  - Cache provides model data as Morsel
  - Single indivisible unit
  - Mapped in constant time

`llama.cpp` with SelFee model: 6.82 GiB



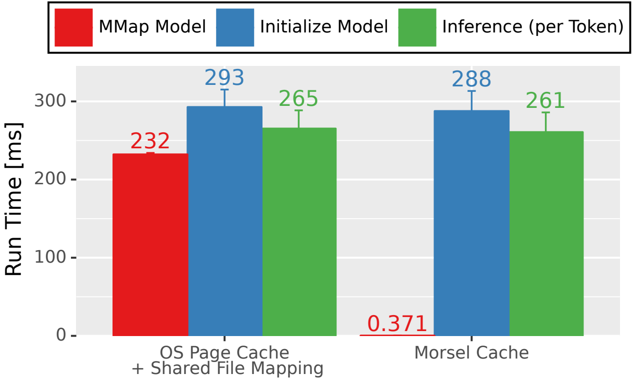
Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

- Conventional file mapping (pre-faulted)
  - Uses individual 4-KiB pages
  - Populated from the page cache
  - Time increases with mapping size
- Morsel mapping
  - Cache provides model data as Morsel
  - Single indivisible unit
  - Mapped in constant time

`llama.cpp` with SelFee model: 6.82 GiB



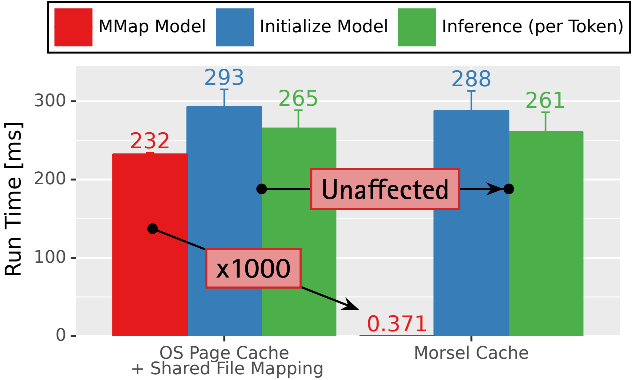
Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

- Conventional file mapping (pre-faulted)
  - Uses individual 4-KiB pages
  - Populated from the page cache
  - Time increases with mapping size
- Morsel mapping
  - Cache provides model data as Morsel
  - Single indivisible unit
  - Mapped in constant time

`llama.cpp` with SelFee model: 6.82 GiB



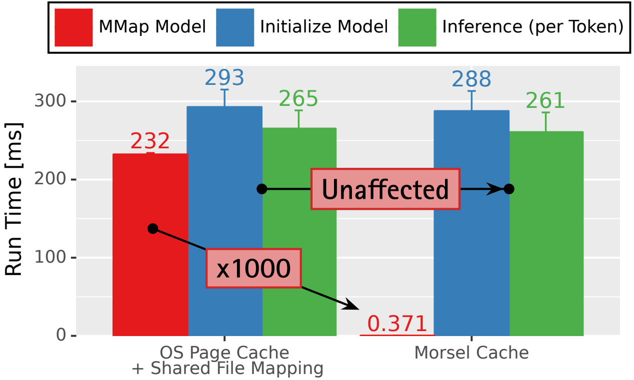
Case Study 1: User-Space Read-Only File Cache

Evaluation: Prototypic implementation in Linux

Task: Inference with a large language model

- Conventional file mapping (pre-faulted)
  - Uses individual 4-KiB pages
  - Populated from the page cache
  - Time increases with mapping size
- Morsel mapping
  - Cache provides model data as Morsel
  - Single indivisible unit
  - Mapped in constant time

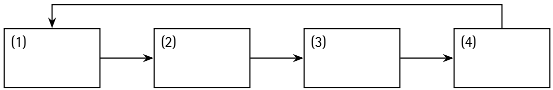
`llama.cpp` with SelFee model: 6.82 GiB



Result: 45% reduced startup time

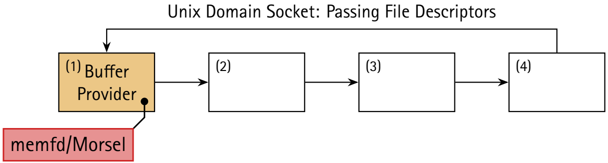
Case Study 2: Processing Pipeline

Unix Domain Socket: Passing File Descriptors



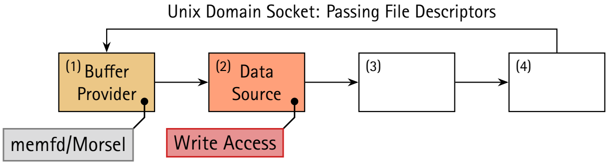
Case Study 2: Processing Pipeline

- Comparing Morsels with memfd



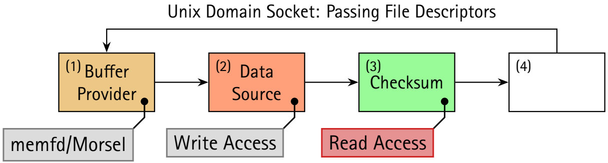
Case Study 2: Processing Pipeline

- Comparing Morsels with memfd



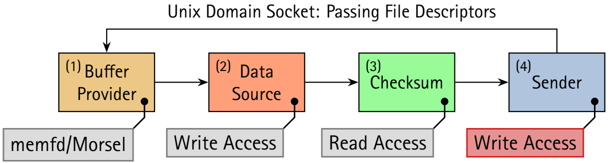
Case Study 2: Processing Pipeline

- Comparing Morsels with memfd



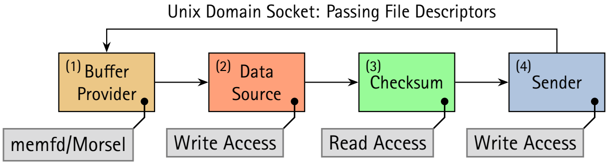
Case Study 2: Processing Pipeline

- Comparing Morsels with memfd



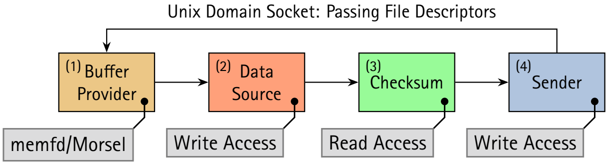
Case Study 2: Processing Pipeline

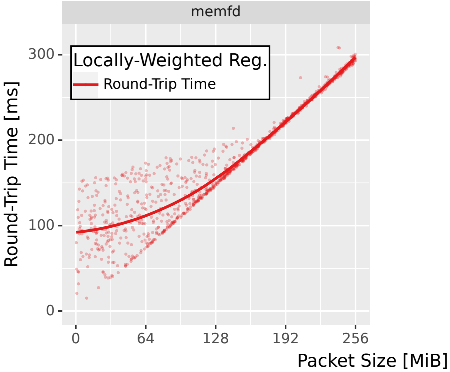
- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size



Case Study 2: Processing Pipeline

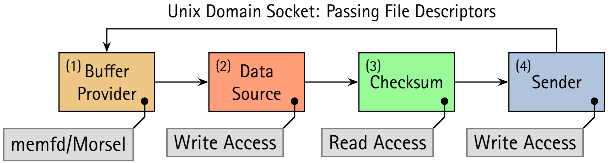
- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size

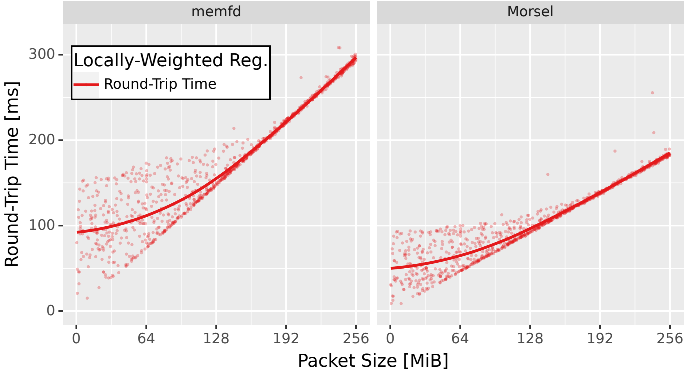




Case Study 2: Processing Pipeline

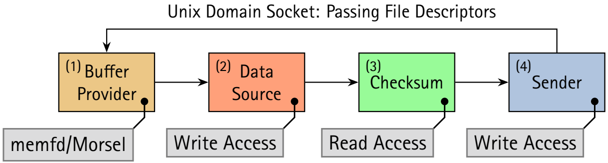
- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size

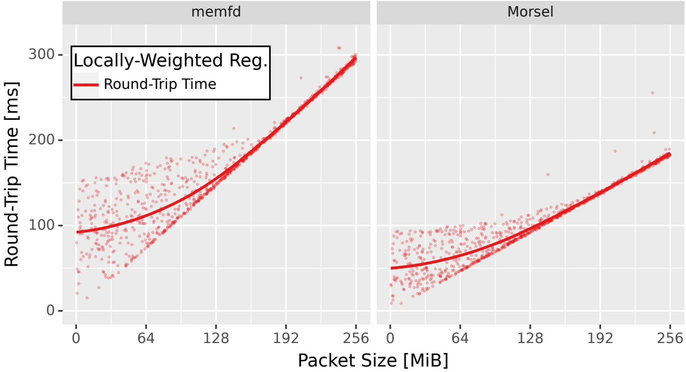




Case Study 2: Processing Pipeline

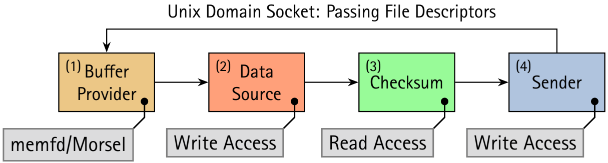
- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size
- Morsels: 39% reduced round-trip time on average

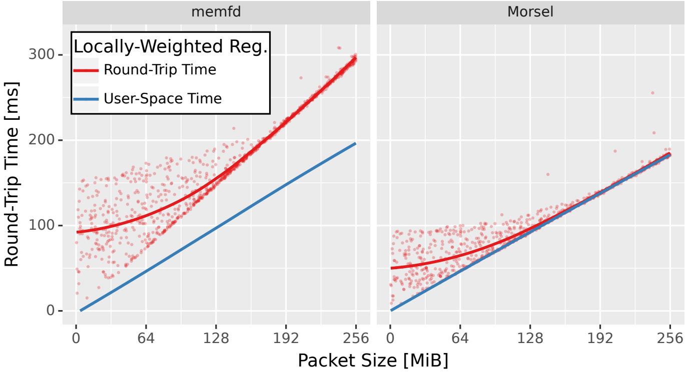




Case Study 2: Processing Pipeline

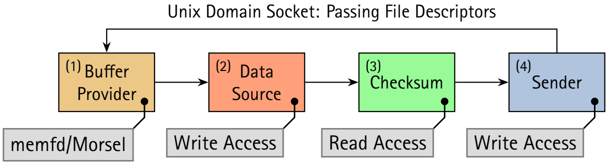
- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size
- Morsels: 39% reduced round-trip time on average

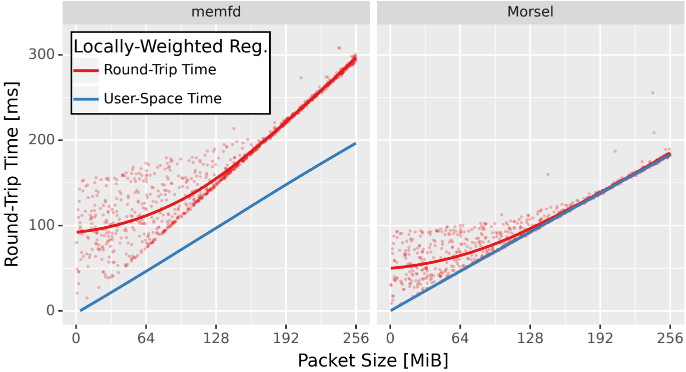




Case Study 2: Processing Pipeline

- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size
- Morsels: 39% reduced round-trip time on average
- Use-space time draws lower bound
  - \+ Waiting time
  - \+ Kernel processing
  - = Round-trip time

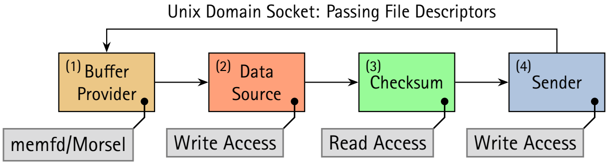


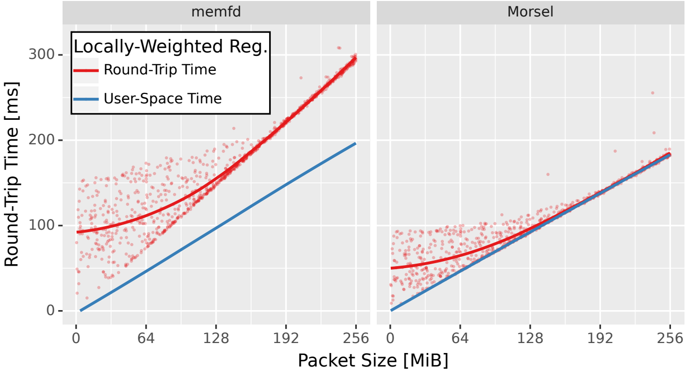


Case Study 2: Processing Pipeline

- Comparing Morsels with memfd
  - 1 000 packets per type
  - Up to 256 MiB packet size
- Morsels: 39% reduced round-trip time on average
- Use-space time draws lower bound
  - \+ Waiting time
  - \+ Kernel processing
  - = Round-trip time

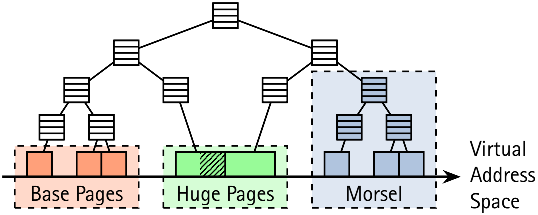
Conclusion: Morsels enable efficient handling of virtual memory!





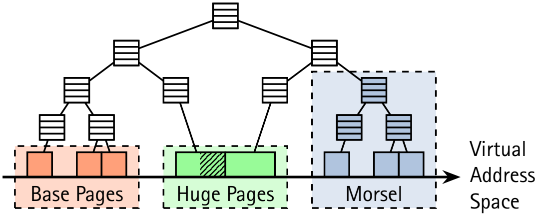
Morsels: Highly efficient handling of large memory objects

- **Problem:** Conventional memory management
  - Many individual entities (pages)
  - Huge pages induce fragmentation
  - No persistent state



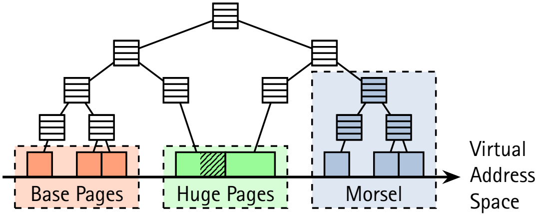
Morsels: Highly efficient handling of large memory objects

- **Problem:** Conventional memory management
  - Many individual entities (pages)
  - Huge pages induce fragmentation
  - No persistent state
- **Solution:** Morsels
  - Indivisible unit (page-table subtree)
  - Ready for persistent memory
  - Efficiently sharable

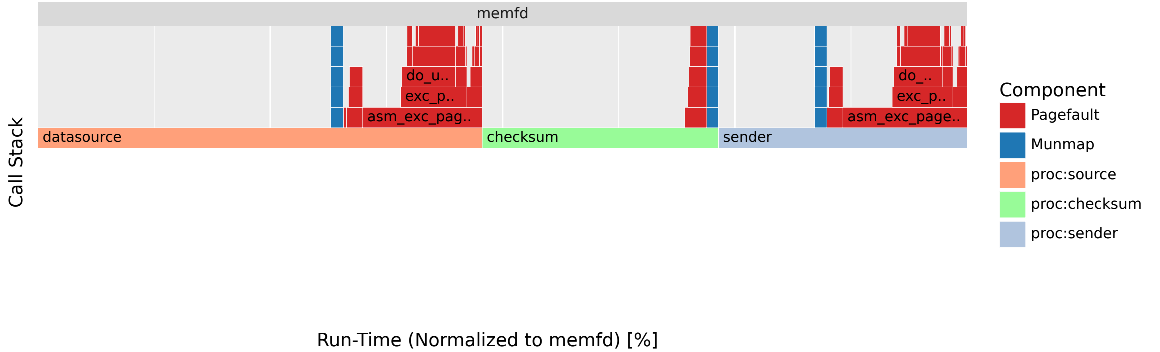


Morsels: Highly efficient handling of large memory objects

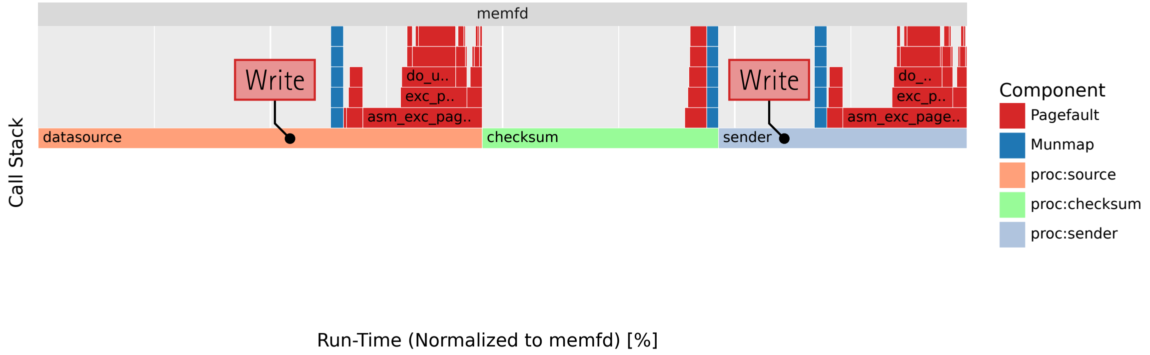
- **Problem:** Conventional memory management
  - Many individual entities (pages)
  - Huge pages induce fragmentation
  - No persistent state
- **Solution:** Morsels
  - Indivisible unit (page-table subtree)
  - Ready for persistent memory
  - Efficiently sharable
- Results
  - Constant mapping time (x1000 faster for 7 GiB)
  - 39% reduced RTT for shown pipeline



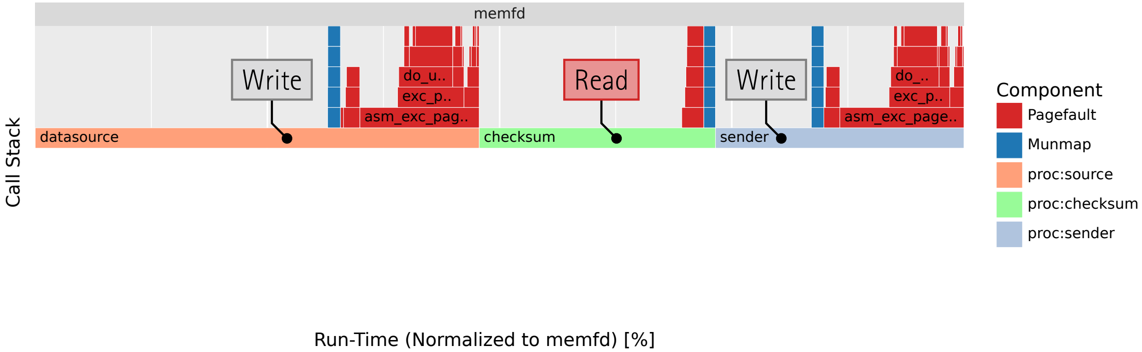
Case Study 2: Processing Pipeline



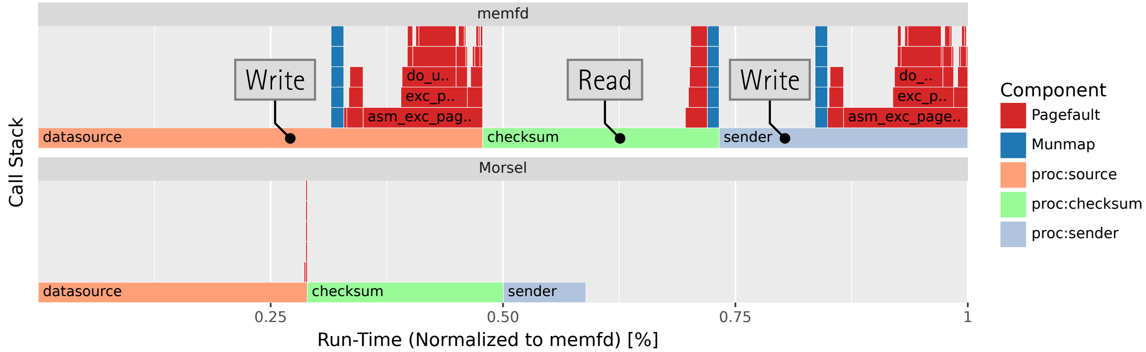
Case Study 2: Processing Pipeline



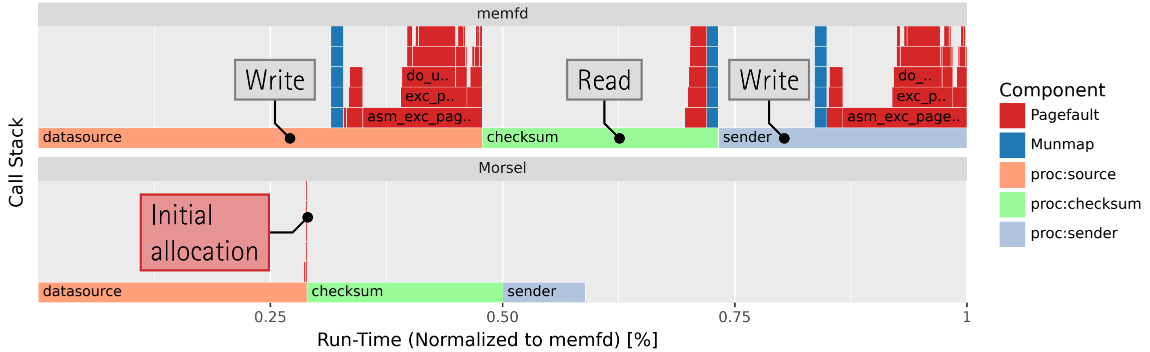
Case Study 2: Processing Pipeline



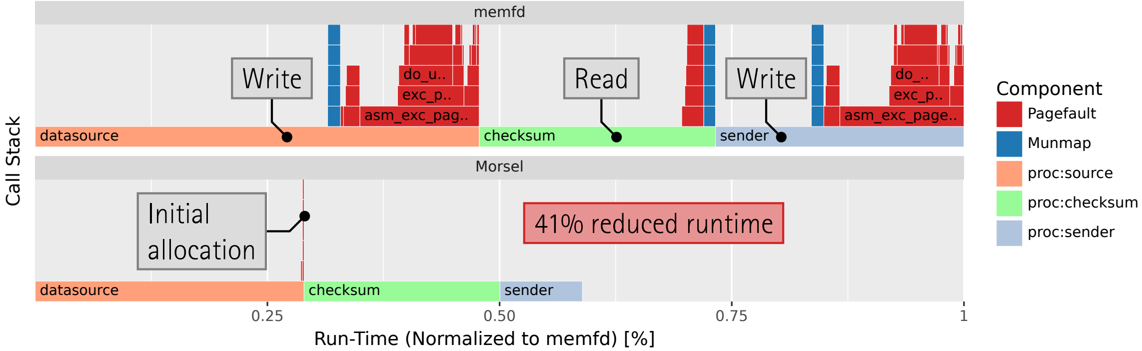
Case Study 2: Processing Pipeline



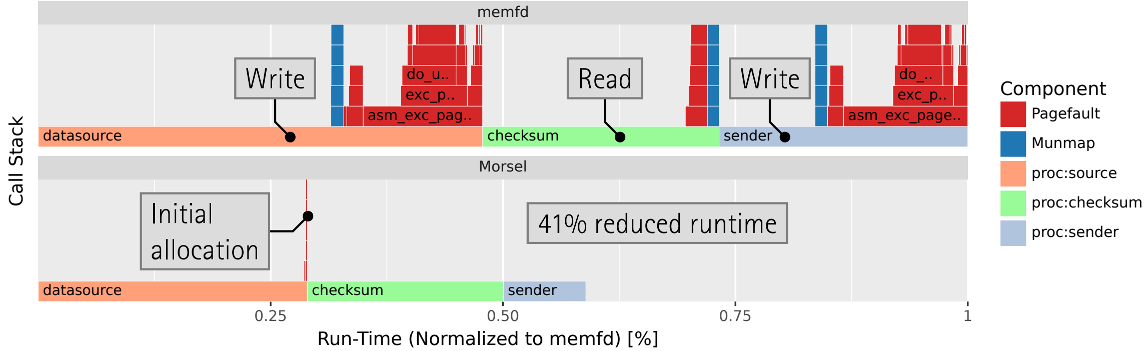
Case Study 2: Processing Pipeline



Case Study 2: Processing Pipeline



Case Study 2: Processing Pipeline



Conclusion: Morsels enable efficient handling of virtual memory!