TUHH
Technische Universität Hamburg

Leibniz Universität Hannover

LLFree: Scalable and Optionally-Persistent Page-Frame Allocation

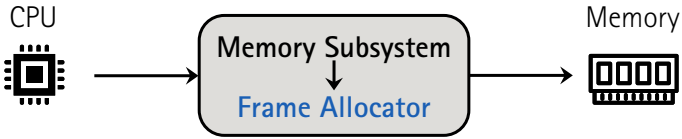
USENIX ATC'23

Lars Wrenger Florian Rommel Alexander Halbuer Christian Dietrich Daniel Lohmann

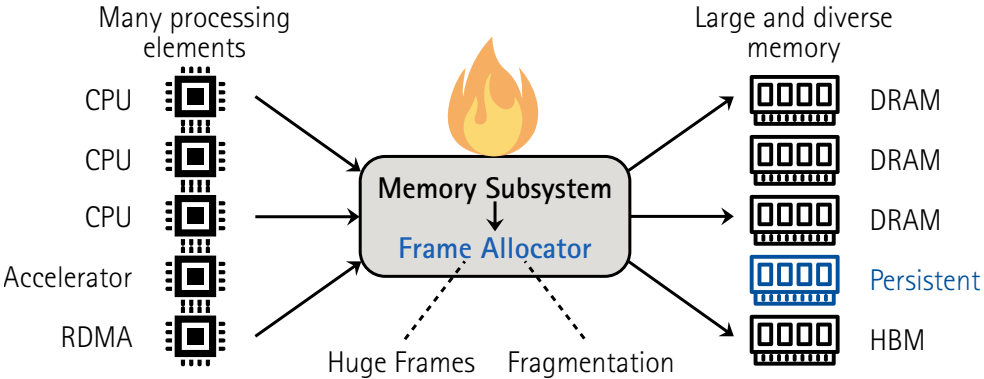
Leibniz Universität Hannover
wrenger@sra.uni-hannover.de

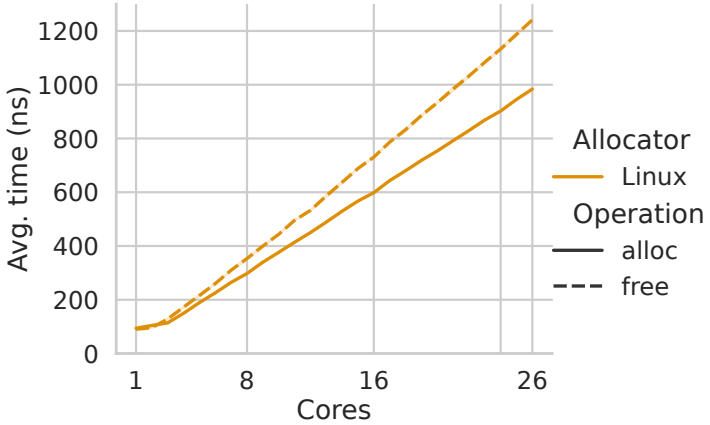
2023-07-12

Memory Management Challenges

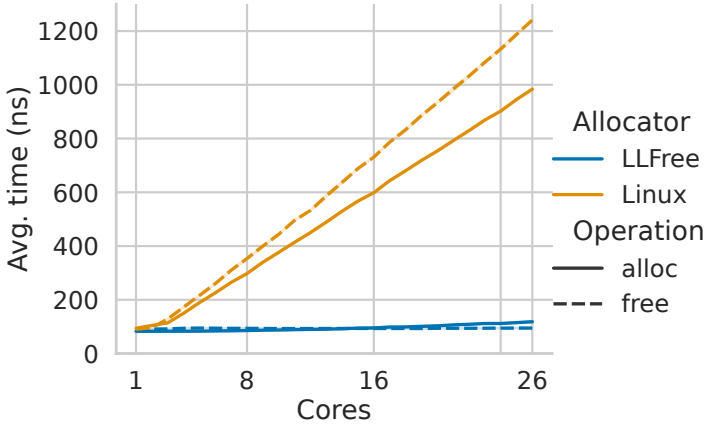


Memory Management Challenges



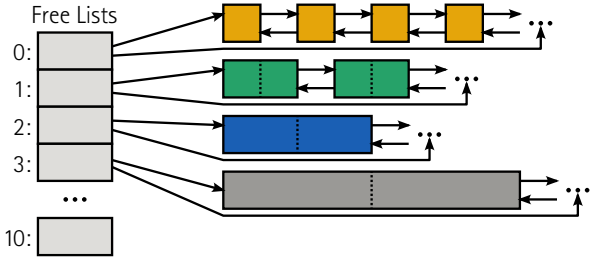


- Allocate 64 GiB as 4 KiB pages in parallel (Linux 6.0)

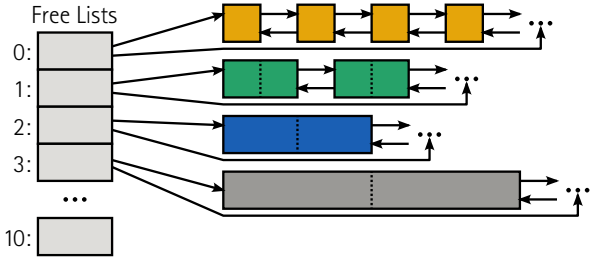


- Allocate 64 GiB as 4 KiB pages in parallel (Linux 6.0)

The Linux Page Frame Allocator

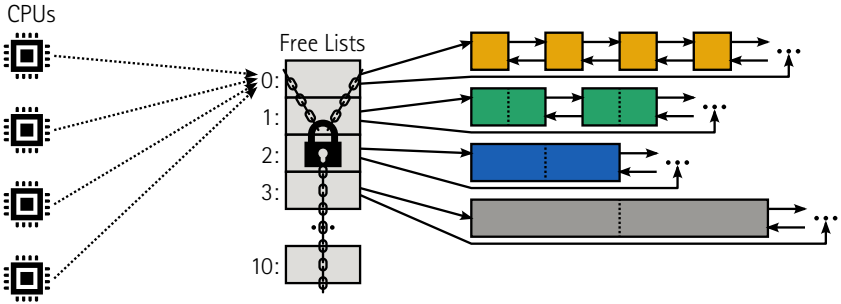


The Linux Page Frame Allocator



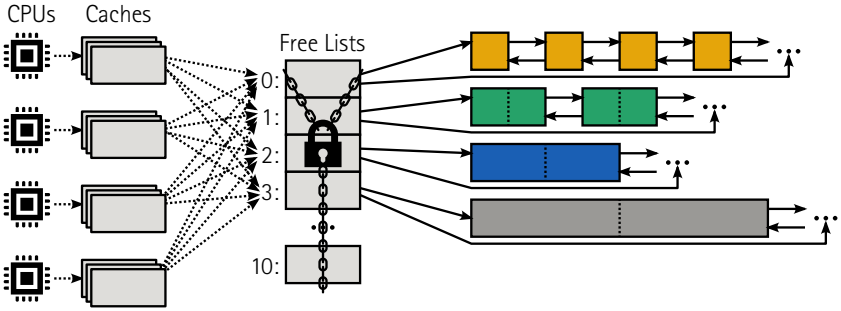
- Split/Merge Costs

The Linux Page Frame Allocator



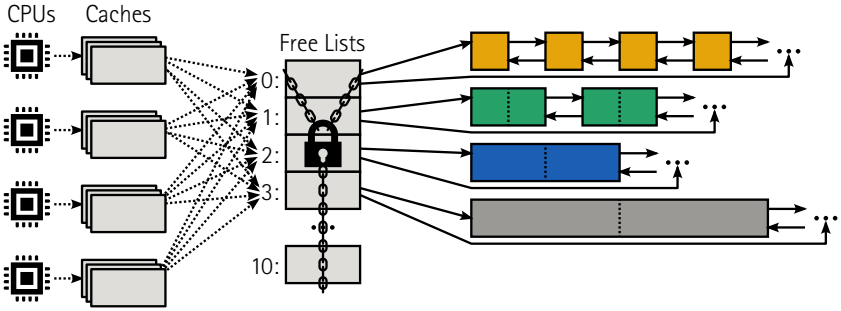
- Split/Merge Costs
- Scalability

The Linux Page Frame Allocator



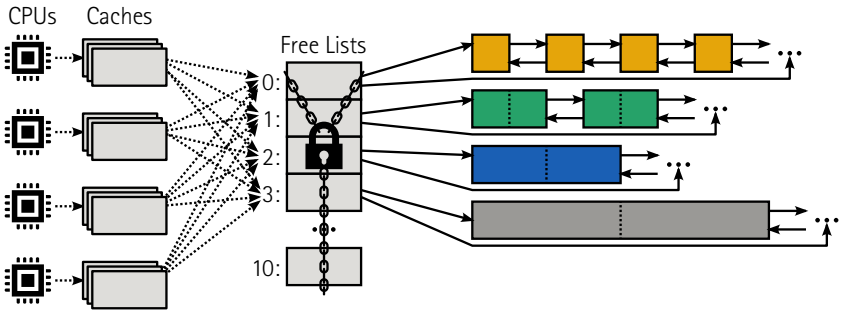
- Split/Merge Costs
- Scalability

The Linux Page Frame Allocator



- Split/Merge Costs
- Scalability
- Fragmentation

The Linux Page Frame Allocator



- Split/Merge Costs
- Scalability
- Fragmentation
- Persistency

Design Principles

Respect Hardware: hardware-specifc page sizes, cache-line granularity

Avoid Sharing: preventing memory sharing bottlenecks

Careful Redundancy: avoid synchronization problems for persistent state

The LLFree Allocator

Design Principles

Respect Hardware: hardware-specifc page sizes, cache-line granularity

Avoid Sharing: preventing memory sharing bottlenecks

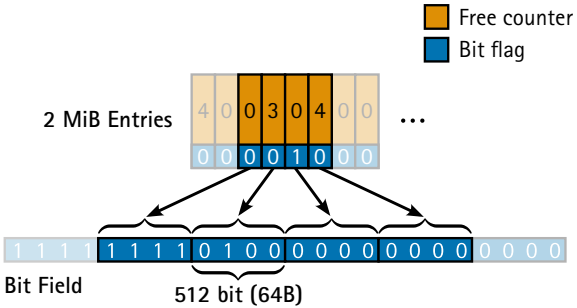
Careful Redundancy: avoid synchronization problems for persistent state

- **Lock- and log-free** → LLFree
  - Locks (Mutex): Not crash consistent!
  - Logging (ACID): Slow and intensifies write wearing!
- Atomic updates → "Jump between valid states"

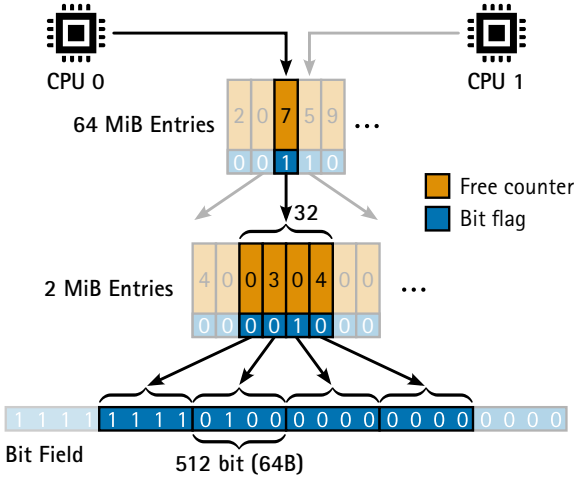
Architecture

Bit Field

Architecture

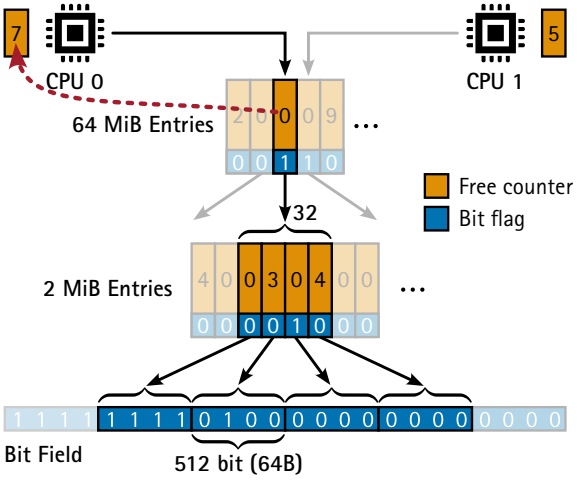


Architecture



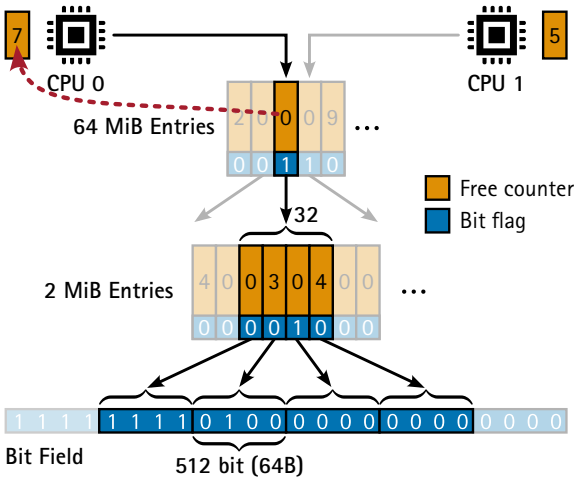
- **Avoid Sharing** in 64 MiB Chunks

Architecture



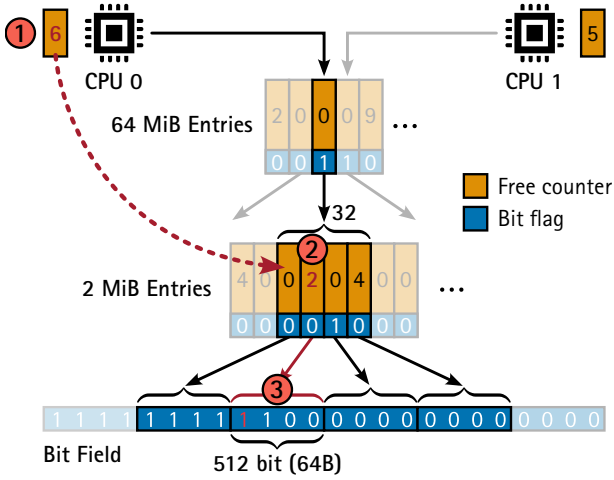
- **Avoid Sharing** in 64 MiB Chunks
  - False-sharing on the 64 MiB counters
    ⇒ Local split counters

Architecture

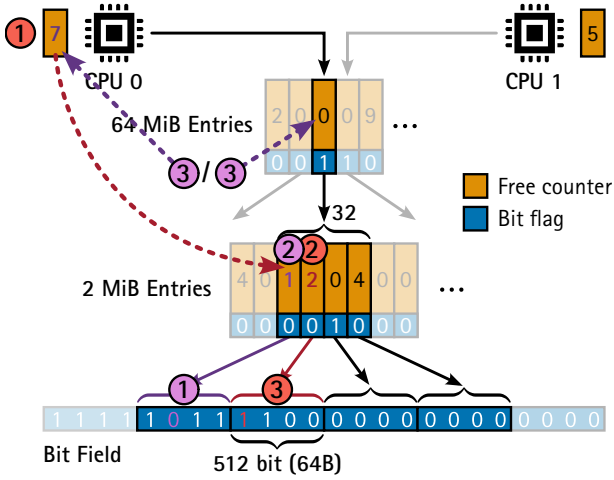


- **Avoid Sharing** in 64 MiB Chunks
  - False-sharing on the 64 MiB counters
    ⇒ Local split counters
- **Reservation policy**
  → Avoid Fragmentation
  - Prioritize already fragmented 64 MiB Areas

Allocate and Free



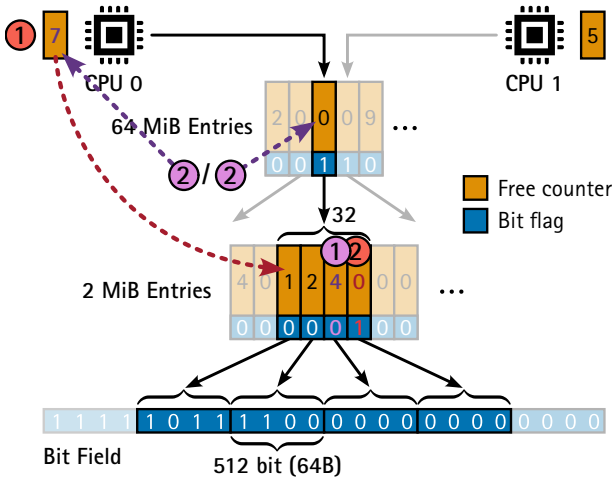
4 KiB Allocation: 3 cache lines



4 KiB Allocation: 3 cache lines

4 KiB Free: 3 cache lines

Allocate and Free



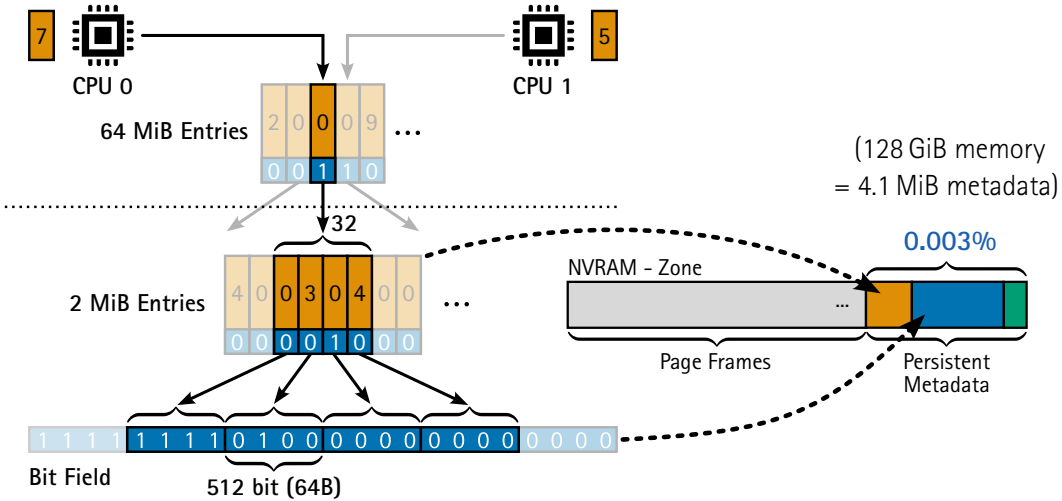
4 KiB Allocation: 3 cache lines

4 KiB Free: 3 cache lines

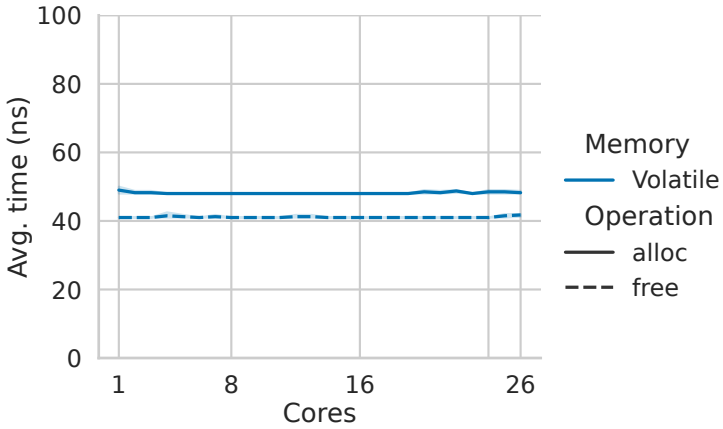
2 MiB Allocation: 2 cache lines

2 MiB Free: 2 cache lines

Persistence and Recovery

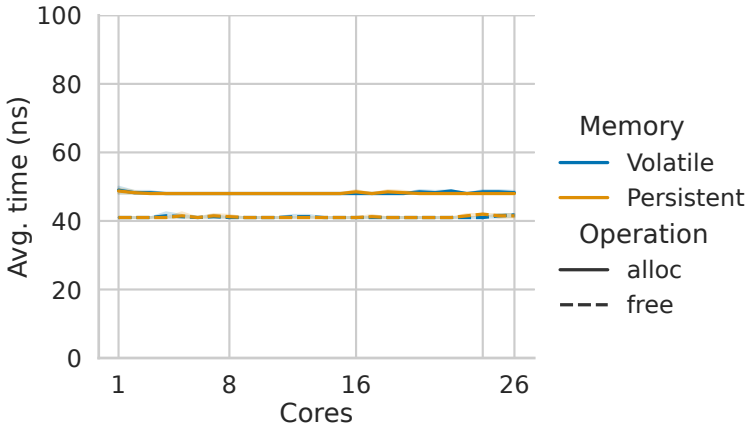


Evaluation: 4 KiB frames



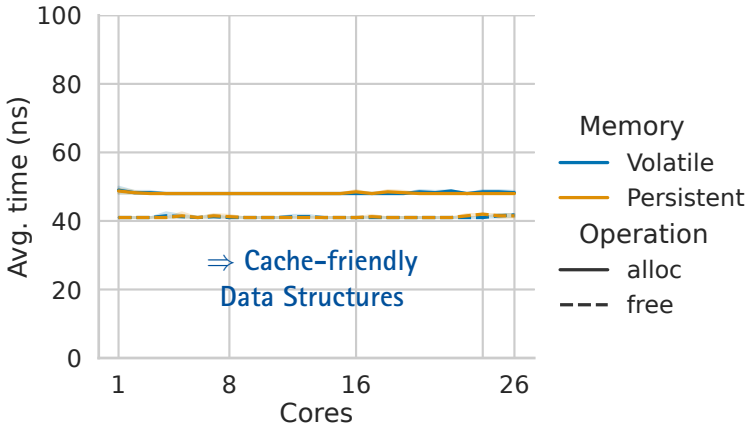
- Allocating 64 GiB as 4 KiB frames in parallel

Evaluation: 4 KiB frames



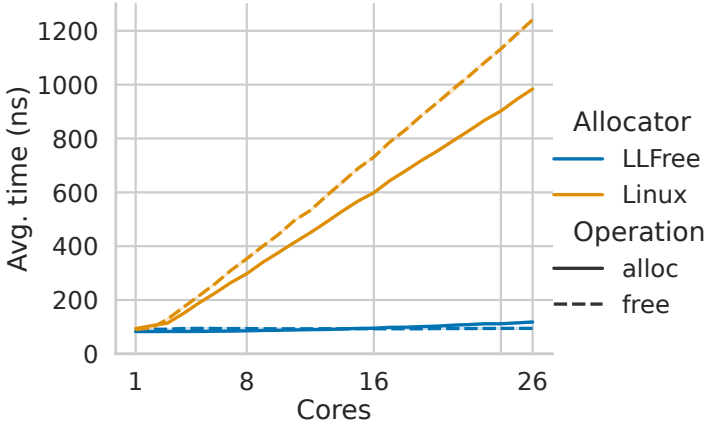
- Allocating 64 GiB as 4 KiB frames in parallel

Evaluation: 4 KiB frames



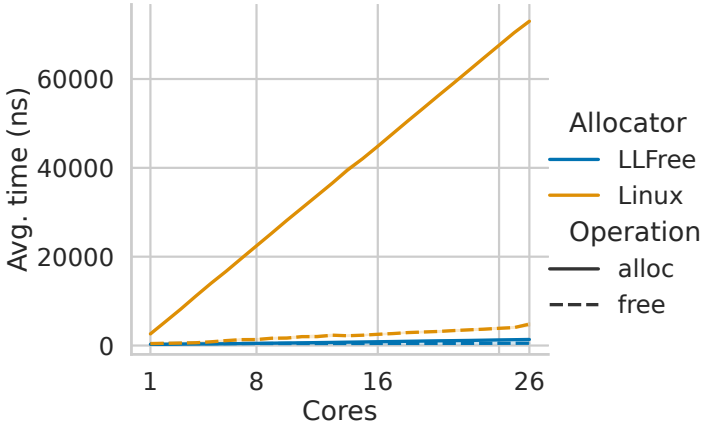
- Allocating 64 GiB as 4 KiB frames in parallel

Evaluation: 4 KiB frames



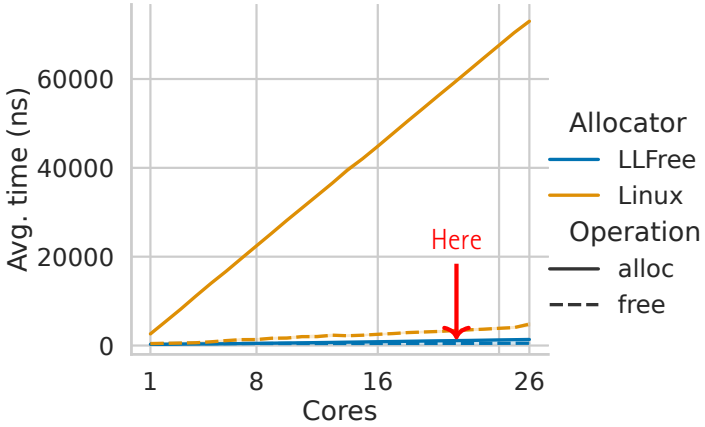
- Allocating 64 GiB as 4 KiB frames in the Linux Kernel (6.0)

Evaluation: 2 MiB frames



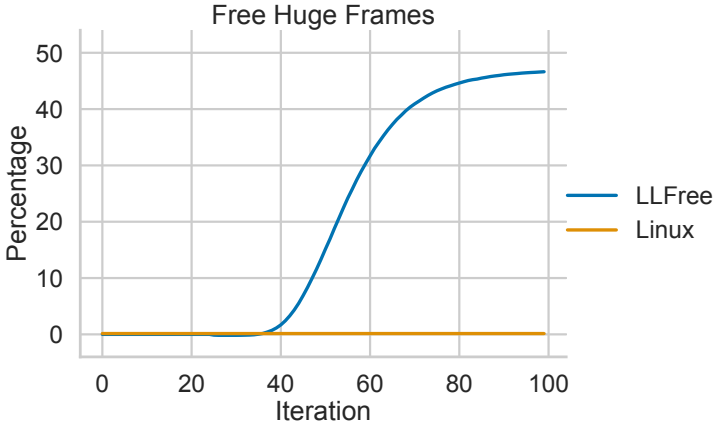
- Allocating 64 GiB as **2 MiB** frames in the Linux Kernel (6.0)
- Linux has a severe performance problem (patched in 6.4)

Evaluation: 2 MiB frames



- Allocating 64 GiB as **2 MiB** frames in the Linux Kernel (6.0)
- Linux has a severe performance problem (patched in 6.4)

Automatic Defragmentation



- Initially, half of the memory is allocated and entirely fragmented
- In every iteration, randomly reallocate 10 % of the allocated pages

Summary

- The Linux page allocator
  - Does not scale well
  - Prone to fragmentation
  - Not persistent
- LLFree
  - Lock- and log-free
  - Excellent multicore scaling
  - Over-time defragmentation
  - Persistent on NVM

Try LLFree

Open Source at
github.com/luhsra/llfree-rs

- Email: wrenger@sra.uni-hannover.de