eBPF-mm: Userspace-guided memory management with eBPF

Konstantinos Mores, Stratos Psomadakis, Georgios Goumas

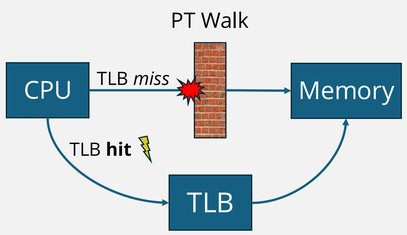
National Technical University of Athens

Introduction

Research Problem

- Virtual Memory Overhead.
- Address Translation Wall [1].
- Translation Lookaside Buffers hit ratio.
- Huge Page management.

Address Translation



Motivation

- Armv8-A and RISC-V support <u>additional</u> huge page sizes.
  - ➢ *64KiB* and *32MiB*.
  - ➢ Coalesced PTE and PMD entries in a single TLB entry.

Linux THP

- No fine-grained control over huge page size selection.
- Greedy, cost-unaware huge page allocation policies.

<u>Our Proposal</u>

eBPF programs capable of:
- Attaching to huge page management key decision points.
- Fine-grained control over huge page size selection.

Using eBPF implement a *cost-benefit* policy:
- ❖ Inspired by CBMM [2].
- ❖ Determines the most beneficial huge page size.

Benefit Estimation

- DAMON for access frequency.
- HW-based TLB miss sampling with armv8-A SPE.

Memory Region Classification [3]

- TLB-friendly.
- High-Reuse TLB-Sensitive.
- Low-Reuse TLB-Sensitive.

Cost Calculation

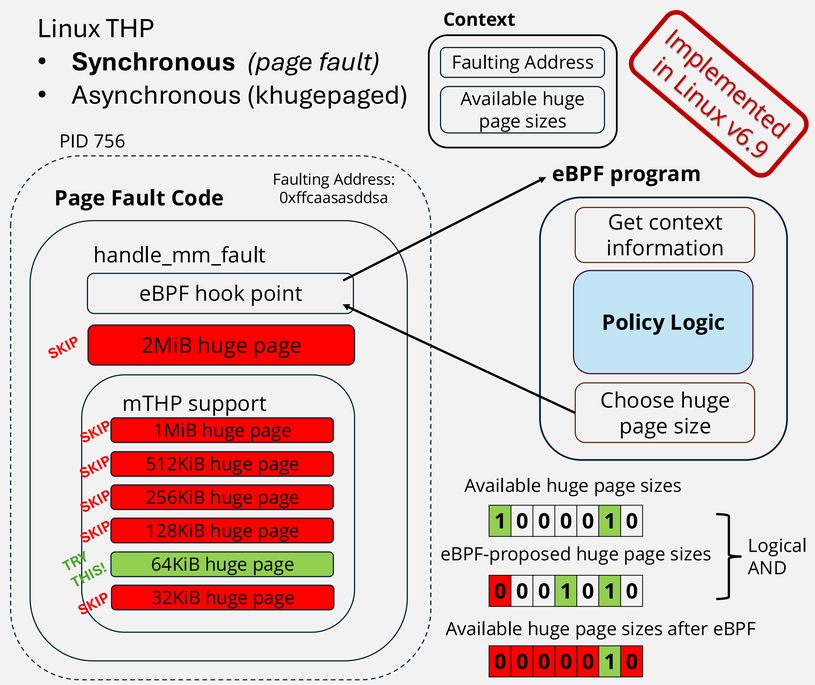
Costs:
- Prepare memory area.
- Find contiguous physical memory.

Calculate costs with *tracepoints*.

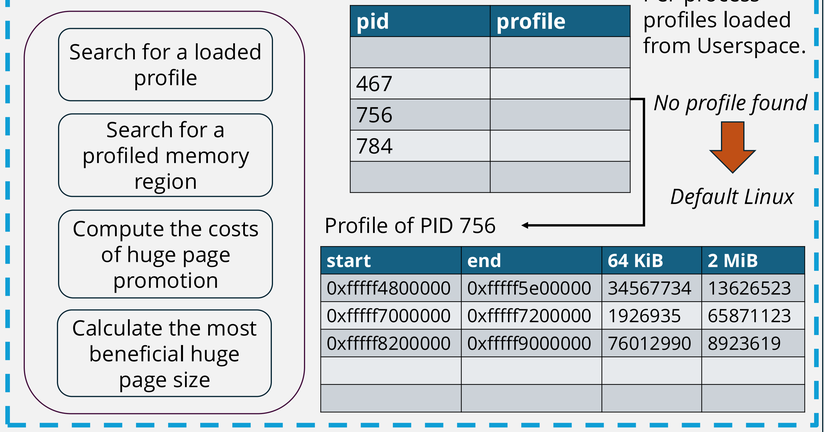
Larger page size
Higher setup cost

Implementation

Linux THP
- **Synchronous** *(page fault)*
- Asynchronous (khugepaged)



Policy Logic: *Cost-Benefit*

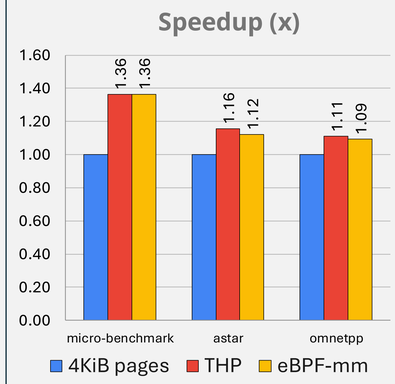


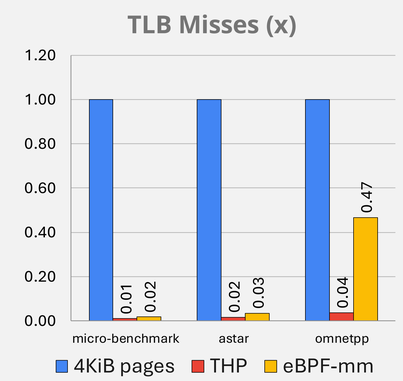
Cost = Allocation + Preparation

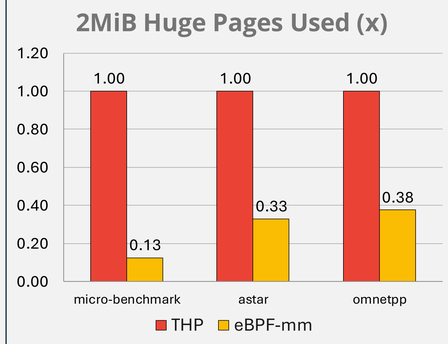
<u>Allocation</u>
- Zero if abundant contiguous memory.
- Else, cost of memory compaction.
- New eBPF helper function informs about memory contiguity.

<u>Preparation</u>
- Memory zeroing *(anonymous)*.
- Secondary Storage IO *(file-backed)*.

Results







Observations

eBPF-mm achieves similar performance with Linux THP:
- Uses only a fraction of 2MiB huge pages.
- Reduces TLB misses adequately.
- Utilizes intermediate 64KiB translation sizes.

Future Work

Implement more policies regarding huge page allocation:
- ➢ Fair distribution of huge pages among processes.

Expand eBPF programs to other parts of the memory management subsystem:
- ➢ Page placement in memory-tiered systems
- ➢ Victim-page selection for reclamation.

Conclusions

- ✓ Less memory bloat generated by avoiding internal fragmentation.
- ✓ More flexible to use under fragmentation.
- ✓ Possibly allow memory contiguity to be consumed by other applications.
- ✓ Intermediate sizes balance the trade-offs of huge pages.

References

1. Bhattacharjee, **"Preserving Virtual Memory by Mitigating the Address Translation Wall"**, In IEEE Micro 2017.
2. M. Mansi, **"CBMM: Financial Advice for Kernel Memory Managers"**, In USENIX ATC 2022.
3. Aninda Manocha, **"Architectural Support for Optimizing Huge Page Selection Within the OS"**, In MICRO 2023.