SRI-2021

Making Huge Pages Effective

Mentor - Prof. Jay Prakash Lalchandani

Team members

1. Aayush Desai - 201801060
2. Rutwa Rami - 201801205
3. Mitesh koradia - 201801017

What we have done

- High page allocation is not successful In long term due to fragmentation in many senario.
- Also due to high degree of fragmentation, a large amount of time is wasted in compaction algorithm.
- So we researched and implemented an algorithm in C++ which take care of the hybrid blocks (having both movable and unmovable pages).
- Our implementation improves the success of huge page allocation and optimises the time of compaction algorithm in high load condition.

Efficient address translation with huge pages

- TLB misses overhead are more expensive in virtualized systems.
- Huge pages allow to keep more data in to the memory.
- Huge pages reduce TLB misses .
- Thus Huge pages have great impact on the performance.

Two way Classification of Memory

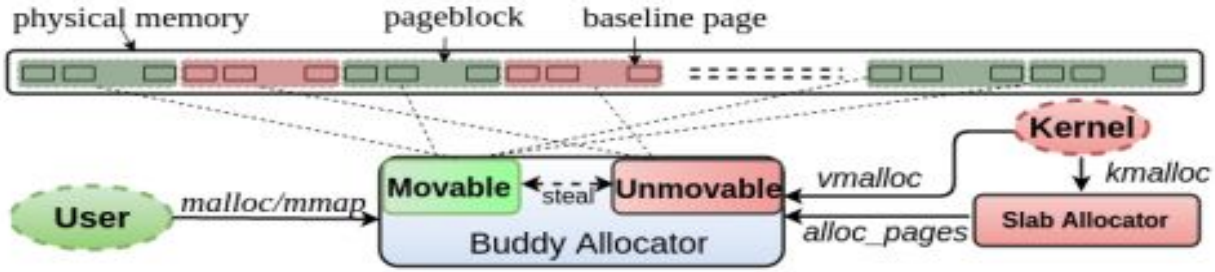
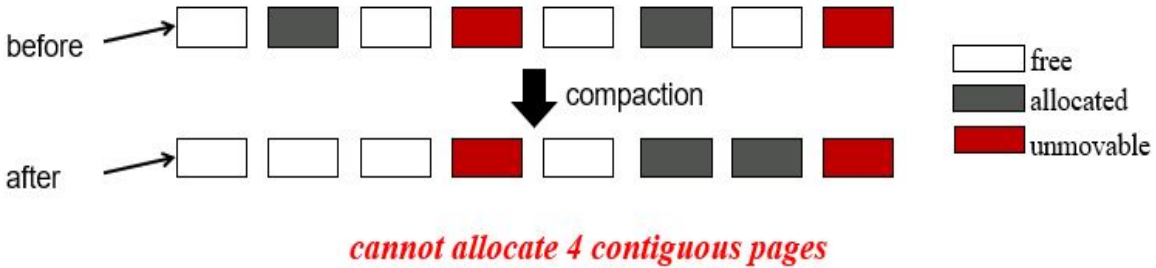


Figure 2. Physical memory management in Linux. The slab allocator allocates kernel objects while the buddy allocator serves pages to the slab allocator (from the unmovable region) and to the user space (from the movable region).

Source : Ashish Panwar, Aravinda Prasad, and K. Gopinath. 2018. Making Huge Pages Actually Useful. In ASPLOS '18: 2018 Architectural Support for Programming Languages and Operating Systems, March 24–28, 2018, Williamsburg, VA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3173162.3173203

Why aren't huge pages effective (yet)?

- It requires a large contiguous memory.
- As the time passes system becomes fragmented.
- Unmovable pages are sparsely distributed in the memory.



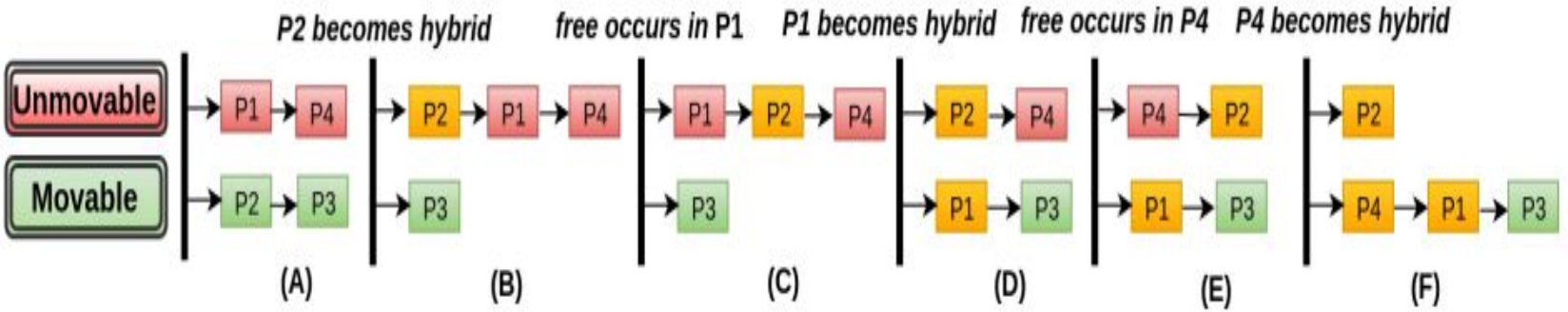
Unmovable Pages

- The pages allocated by kernel are unmovable pages.
- User space pages are referenced by page table entry.
- But kernel pages are directly mapped (location=base register + Offset)
- This direct mapping makes kernel pages unmovable.
- This is the main reason for fragmentation via pollution.

Effects of Huge Pages

1. Fragmentation-via-pollution

- With time the unmovable pages spread in entire memory. It leads to permanent fragmentation.
- Due to this buddy allocator is unable assign huge pages.



Source : Ashish Panwar, Aravinda Prasad, and K. Gopinath. 2018. Making Huge Pages Actually Useful. In ASPLOS '18: 2018 Architectural Support for Programming Languages and Operating Systems, March 24–28, 2018, Williamsburg, VA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3173162.3173203

2. Latency-inducing unsuccessful

- It occurs when kernel unnecessarily migrates pages from the polluted regions while attempting to allocate huge pages.
- It induces latency by increasing the cost of recovering from fragmentation.

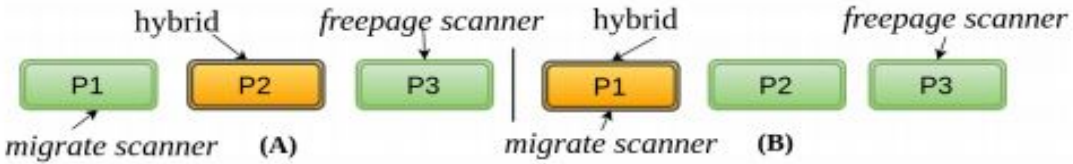


Figure 7. The location of unmovable pages affects the outcome of compaction in the two-way classification approach. In this case, Linux can allocate a huge page only in scenario A while Illuminator can allocate huge page in both A and B.

Source : Ashish Panwar, Aravinda Prasad, and K. Gopinath. 2018. Making Huge Pages Actually Useful. In ASPLOS '18: 2018 Architectural Support for Programming Languages and Operating Systems, March 24–28, 2018, Williamsburg, VA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3173162.3173203

Proposed Solution: Illuminator

Three Way classification

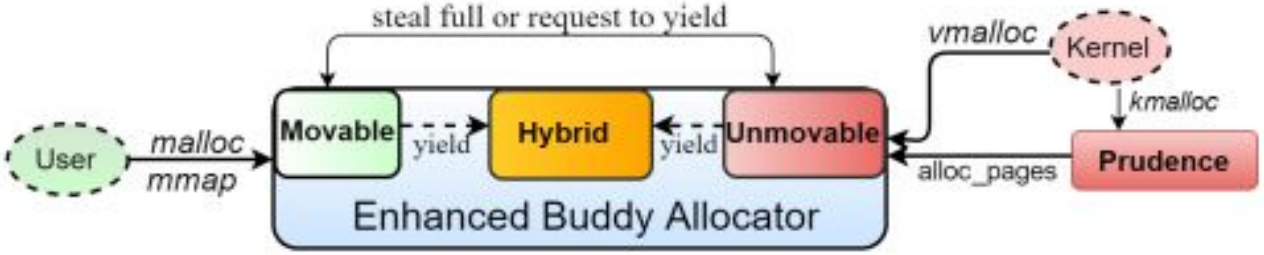
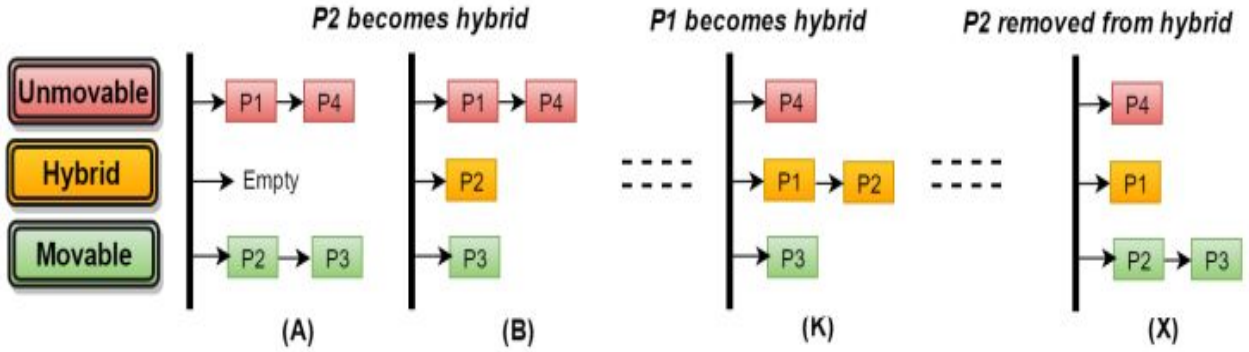


Figure 6. Illuminator explicitly manages hybrid pageblocks in a different region to prevent fragmentation via pollution. Prudence helps Illuminator by minimizing callbacks to `alloc_pages` with timely reclamation of deferred objects.

Source : Ashish Panwar, Aravinda Prasad, and K. Gopinath. 2018. Making Huge Pages Actually Useful. In ASPLOS '18: 2018 Architectural Support for Programming Languages and Operating Systems, March 24–28, 2018, Williamsburg, VA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3173162.3173203

Preventing Fragmentation via Pollution

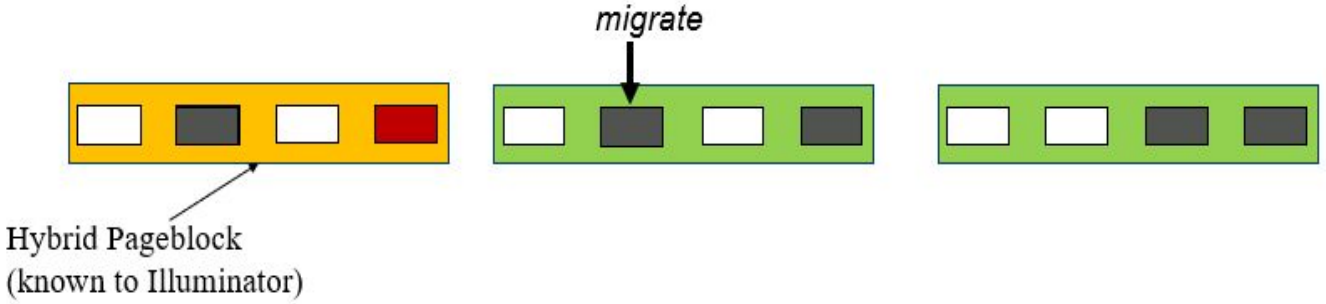


Illuminator also provides a mechanism to reclaim pageblocks from the hybrid region. In this example, P2 is added back to the movable region in state X.

Figure 5. Illuminator improves page clustering by explicitly managing hybrid pageblocks. Once P2 is yielded to the hybrid region during A→B, it is reused until state K. P1 is added to the hybrid region only when P2 fails to allocate memory.

Source : Ashish Panwar, Aravinda Prasad, and K. Gopinath. 2018. Making Huge Pages Actually Useful. In ASPLOS '18: 2018 Architectural Support for Programming Languages and Operating Systems, March 24–28, 2018, Williamsburg, VA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3173162.3173203

Eliminating LIU migration



Source : Ashish Panwar, Aravinda Prasad, and K. Gopinath. 2018. Making Huge Pages Actually Useful. In ASPLOS '18: 2018 Architectural Support for Programming Languages and Operating Systems, March 24–28, 2018, Williamsburg, VA, USA. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3173162.3173203

How Illuminator works

- In this we have kept the track of hybrid blocks which have both movable and unmovable page.
- Whenever there is a fallback then we first try to use hybrid blocks before stealing from the other memory type.
- Whenever the number of movable or unmovable pages become zero in an hybrid block we return that block to respective memory type.
- In compaction algorithm we just skip the hybrid blocks as they will not help to produce memory for huge page.

Results

- Our algorithm gives following results :
  - When there is a 70% to 80% of requests for huge page allocation then this solution of three way classification gives higher success rate than the algorithm used in kernel.
  - Also under the normal working load this solution gives no less performance than the previous algorithm which is two way classification.
  - The time which compaction algorithm used was much less because we just skipped hybrid blocks during compaction algorithm.

What we learned

- We got insight of how the memory is handled by the OS.
- We came to know about how the memory is freed in the kernel.
- We got the knowledge about how compaction algorithm works in the kernel.
- We also learned about how buddy allocator is efficiently used to allocate memory and also combine small chunks of memory to make bigger space.

Thank you