



Better Anon (Swap) Readahead

Kairui Song / Tencent

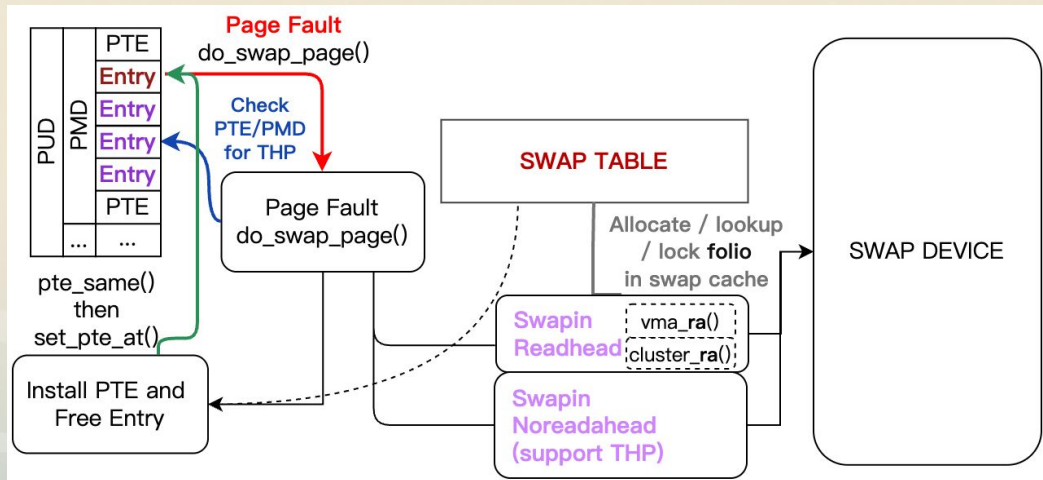


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Swapin: design maintenance issues

- Swapin
 - Mostly unified
- Except:
- SYNC: No Readahead
 - But has THP
 - Kind of “readahead”
- non-sync: Readahead
 - No THP even for large window



- We are bypassing readahead (exclude THP) for the same reason of why we were bypassing swap cache: performance



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Swapin: performance reading

We bypass readahead for SYNC devices because it's harmful for performance...

Readahead is not harmful, a bad readahead heuristic is.

Build Linux Kernel in 1G Memcg, -j16, defconfig, NVME swap:

Anon (Swap):

Traffic: R: 20.03 GB / W: 49.92 GB

- VMA Readahead: statistic counter
 - Hit ratio **~55%**
 - Wasted **~45%**
 - Using default config

File (XFS):

Traffic: R: 20.24 GB / W: 1.30 GB

- Readahead: estimated with kprobe and BCC cachestat
 - Hit ratio >93%
 - Wasted <7%
 - Could be tuned down



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Swapin: performance reading

- Swapin is driven by page fault (mostly), the application stall.
- Readahead (if done well) prevents the application from stall.
- Better performance, dramatically better P99 latency
- Combines with batching, should improve the performance by a lot.

fault entry/exit + generic mm machinery	605	11.5 %
do_swap_page specific (lookup, accounting, rmap, PTE)	482	9.2 %
VMA readahead window scan (PTE walk + window calc)	380	7.2 %
folio alloc + memcg + LRU + swap-cache insert	925	17.6 %
swap bio setup + submit	472	9.0 %
IO + completion + wakeup	2392	45.5 %



Swapin: performance reading

- Swapin is driven by page fault (mostly), the application stall.
- Readahead (if done well) prevents the application from stall.
- Better performance, dramatically better P99 latency
- Combines with batching, should improve the performance by a lot.

fault entry/exit + generic mm machinery (fault handling	620 ± 8	14.1%
`do_swap_page` specific (cache lookup, accounting, rmap, PTE)	405 ± 14	9.2%
folio alloc + memcg + LRU + swap-cache insert + swap bio setup	701 ± 32	15.9%
zram: zsmalloc/zspool read	427 ± 17	9.7%
zram: block/bio submit + complete	257 ± 4	5.8%
zram: zcomp stream get/put	139 ± 1	3.2%
LZ4 decompression (core 1272 + wrapper 133)	 1401 ± 18	 31.8%
swap book keeping (free, cluster management)	449 ± 32	10.2%



LINUX

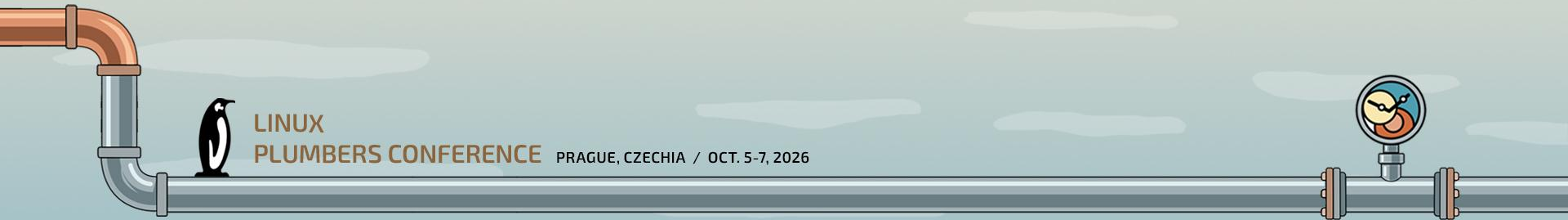
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



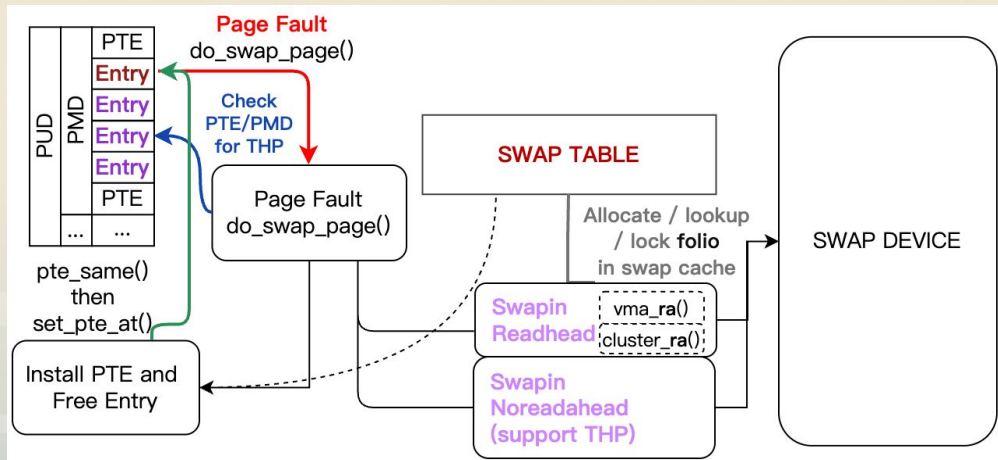
Goal

- Unify Swapin Readahead
- Combined THP with readahead
- Better policy / performance
- Unify Anon / File Readahead?



Swapin: THP and readahead

- SYNC: THP, **no readahead**
 - ZRAM, BRD
 - THP is easier, no wait on IO.
- non-SYNC: **readahead, no THP**
 - For SSD/HDD.
 - **Even if the readahead window is large.**
 - **Tricky to add THP.**



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Background: THP issues for non-SYNC readahead

- For async devices, readahead folios has to stay in swap cache, unmapped, as they are not up to dated.
- No locking.
- So we could end up having unusable **mixed folios** (needs to be split before use)
- Mostly just performance issue
 - Some also applies to SYNC THP swapin but time window is much shorter and easy to solve.
 - Tricky to handle correctly.
- Charging / Reclaim / Thrashing problems.



LINUX

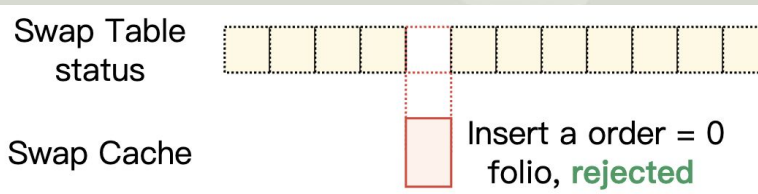
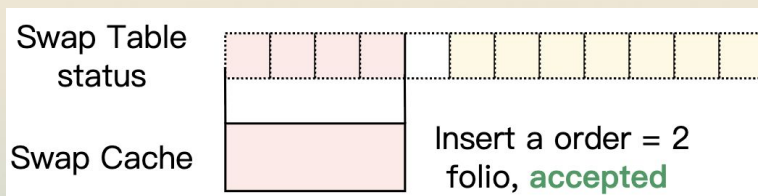
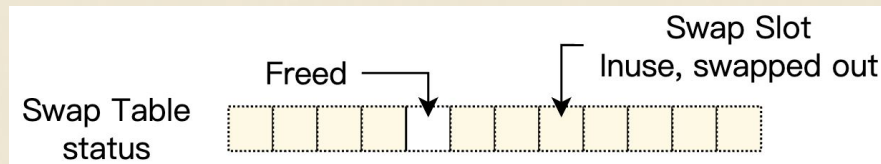
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Background: Swap Cache as synchronization layer

- Every swap in request will lock the swap cluster -> alloc & insert a folio to the cluster's table -> then IO / decomp.
- Higher order folios must occupies a continuous set of swap space (swap entry).
- A successful insert ensures the folio occupies an continuous exclusive, inuse set of swap entries / slots.



LINUX

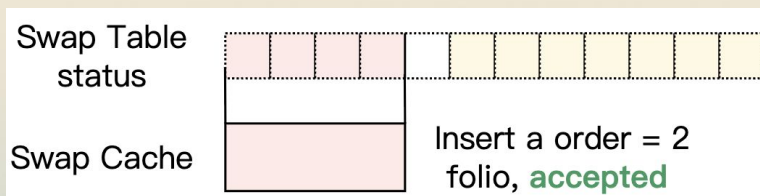
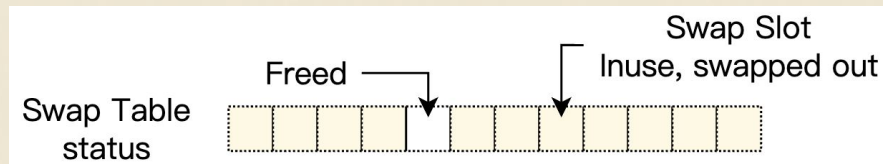
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Background: Swap Cache as synchronization layer

- Holding the **folio lock** ensures the slots / entries are exclusive to the folio, won't be used by anyone else, data is safe and stable.
- Serialize all folio based swap operations. (folio_put_swap, folio_dup_swap, folio_free_swap...)
- Helped to solve a lot of tricky problems.



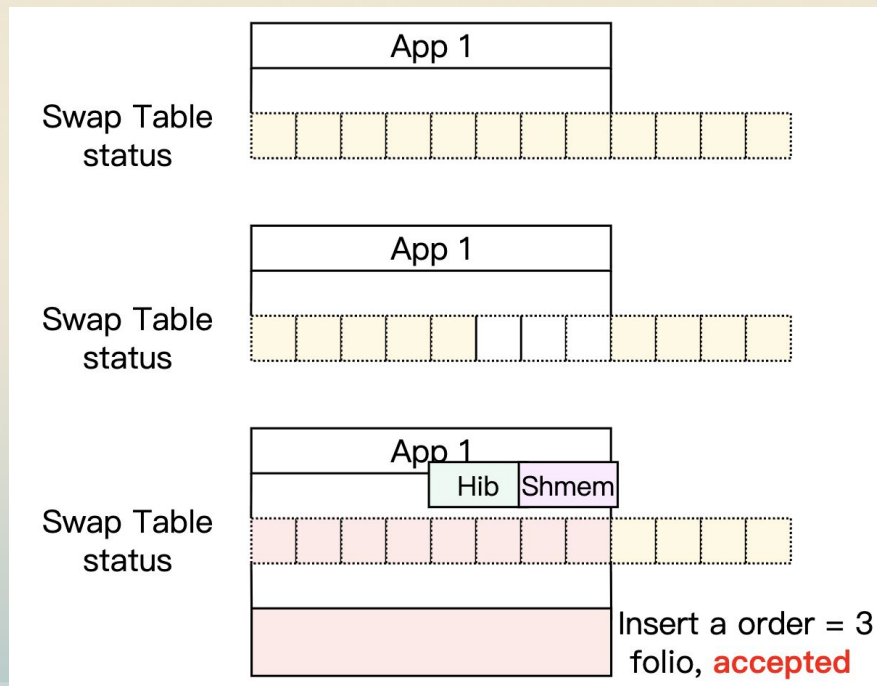
LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Mixed Folio: Swap Cache unaware of the slot type

- Readahead / THP both may see the page table / shmem mapping holds a valid THP region, try to insert a high order folio.
- During the folio allocation & **before the folio insert** start, swap slots get freed.
- Insert start, found slots still in use but they belong to a different component.
- Created a mixed folio, have to check afterwards and discard it.



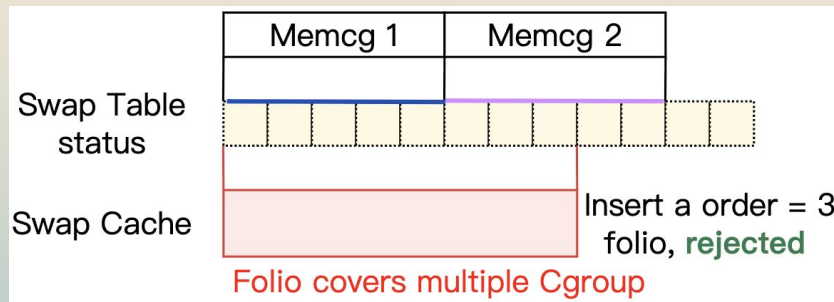
LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Mixed Folio: Swap Cache unaware of the slot type

- This one is not hard to fix, give swap table more type or special count value (SWP_TB_HIB and resurrect of SWAP_MAP_SHMEM).
- Similar for memcg, swap cache also ensures memcg is same now.
- Also zeromap.
- Could be applied to more to avoid mixed folios.
- <https://lore.kernel.org/linux-mm/20260811132209.2862708-1-youngjun.park@lge.com/>



LINUX

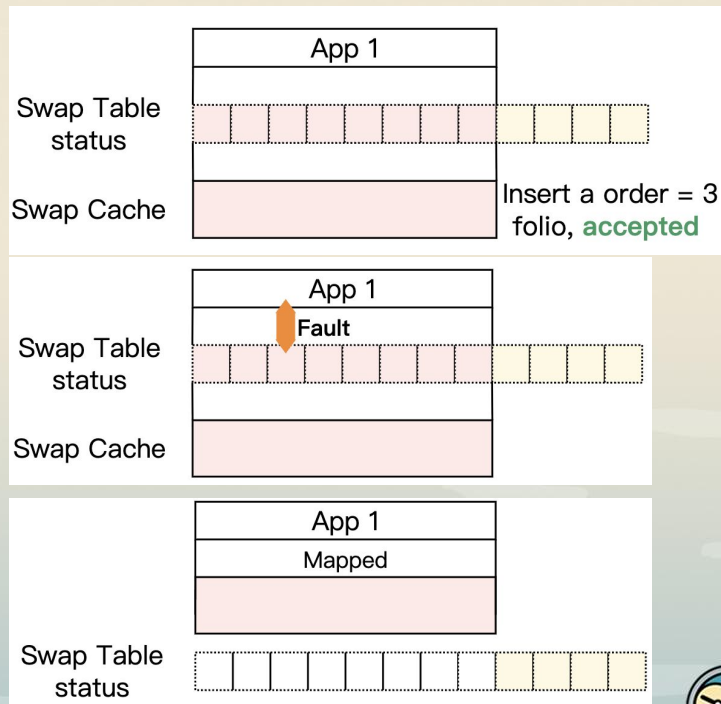
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Mixed Folio: Swap slots may gets freed afterwards

- A folio added to swap cache while the page table and swap table all looking good and suggests readahead / THP.
- Folio stay in swap cache for a while.
- Ideally, application triggers a fault that need this folio.
- Swapin use swap cache, map the folio, then it's good.



LINUX

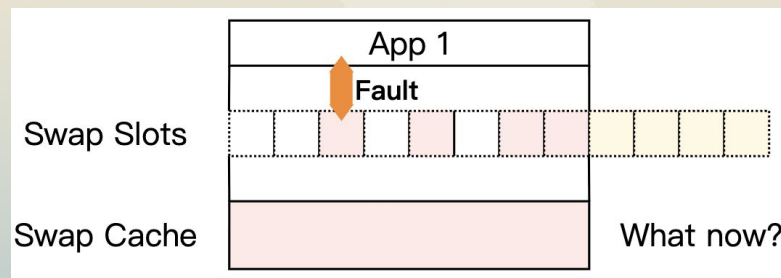
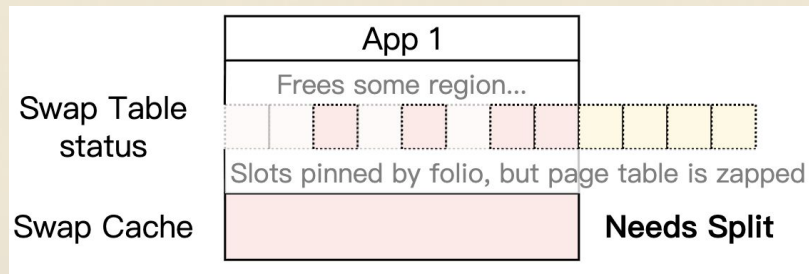
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



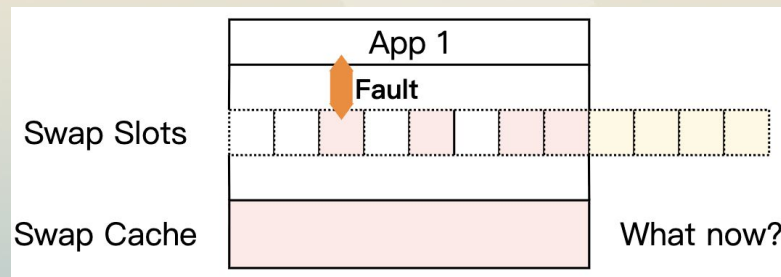
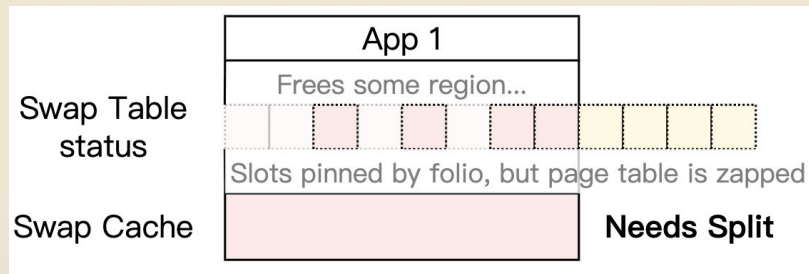
Mixed Folio: Swap slots may get freed afterwards

- Application frees some region.
- The folio staying in swap cache is now a **mappingless partial THP swap cache folio**.
- How to handle that?
- Currently, we just delete it and redo the whole allocation / IO, ugly in implementation, bad for performance.



Mixed Folio: Swap slots may get freed afterwards

- **Mappingless partial THP swap cache folio**
- Split it? Unstable.
 - More magic in swap table?
 - Or special handling in THP?
- Just map it?
 - Waste of space?
 - New rmap API?
- Always add to a mapping for readahead?
 - Cluster readahead / shmem?



Better heuristic

We bypass readahead because it's harmful for performance...

Readahead is not harmful, a bad readahead heuristic is. **RA waste is more critical for compression swap due to re-compression**

Build Linux Kernel in 1G Memcg, -j16, defconfig

Anon (Swap):

Traffic: R: 20.03 GB / W: 49.92 GB

- Readahead: statistic counter
 - Hit ratio **~55% (~52%*)**
 - Wasted **~45% (~48%*)**
- (*cluster readahead)

File (XFS):

Traffic: R: 20.24 GB / W: 1.30 GB

- Readahead: estimated with kprobe and BCC cachestat
 - Hit ratio >93%
 - Wasted <7%
 - Could be tuned down



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Better readahead policy (anon & file)

- Using shadow (Both anon and file has it, but not always true for file)
 - Using refault distance as a hint.
 - Using MGLRU's gen number as a hint.
 - Using MGLRU(FG)'s gen + refs as a hint.
- Using DAMON's info as a hint.
- Copy file readahead's policy (But we don't have a per file thing here).
- Be more memory pressure aware / Thrashing.
 - New Memcg helper or is `__GFP_NORETRY` enough?
- Readahead (ext)?



LINUX

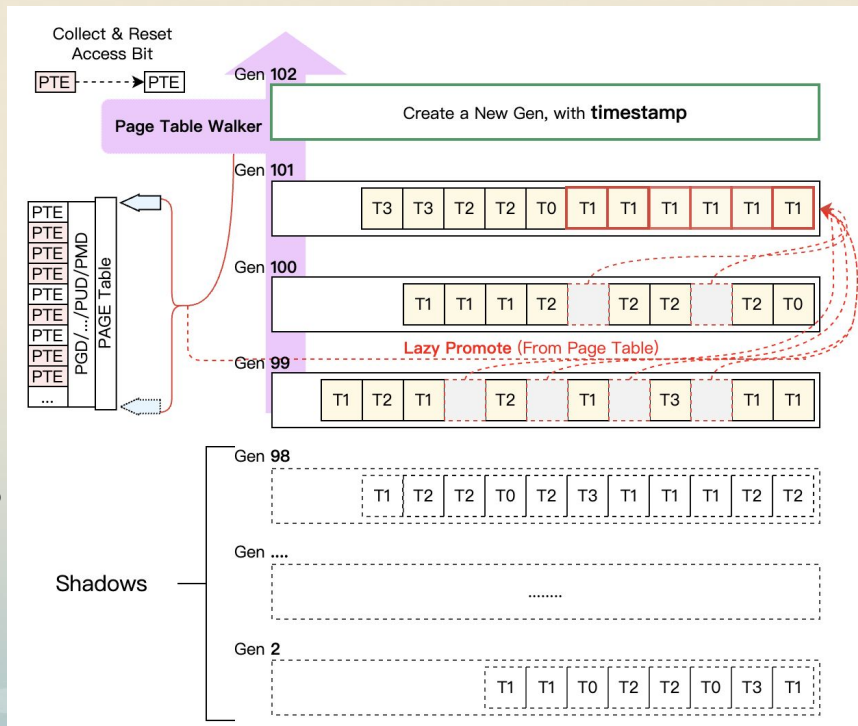
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



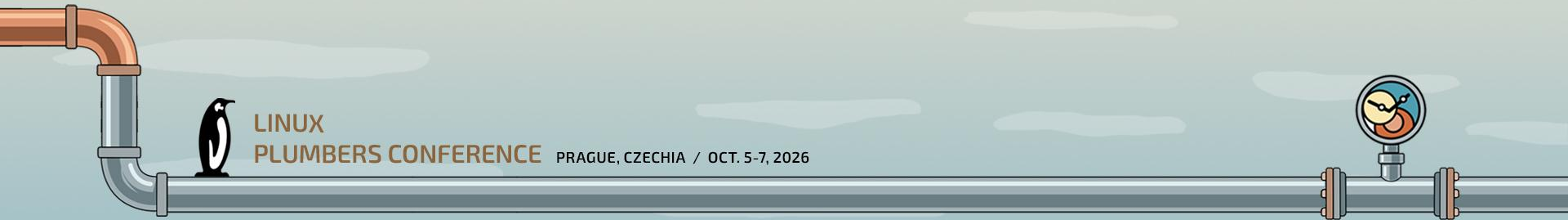
e.g. New Idea for Readahead

- Shadows
- MGLRU:
<GEN NUMBER> | <REFS>
 - If two folios were at the same gen with same refs, they are more likely to be used together.
- Doing extra shadow lookup is super fast now.



Better heuristic

- Memory Pressure Aware
- More knob
- ra_ext?



What can be unified for: Anon vs File Readahead

Anon:

- PG_readahead
 - Update per VMA / Global readahead statistic
- Driven by major fault only
- Per VMA / Global RA info
- Memcg charging:
 - Full reclaim

...

File:

- PG_readahead
 - Start next readahead batch
- Driven by maj/min fault / read
- Per-file RA info
- Memcg charging:
 - `__GFP_NORETRY`

...



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Async readahead

- Can we make better use of async readahead for every kind of devices
- SSD / HDD - Bio
- Compressed Swap - Async decompression? (something like kcompressd?)

