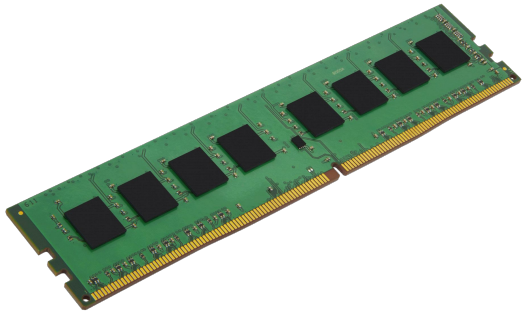
PageFlex: Flexible and Efficient User-space Delegation of Linux Paging Policies with eBPF

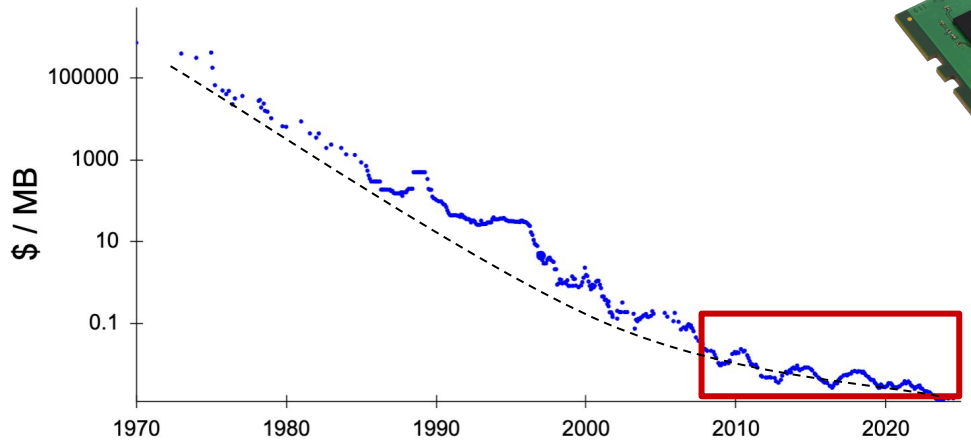
Anil Yelam | Kan Wu | Zhiyuan Guo | Suli Yang | Rajath Shashidhara | Wei Xu
Stanko Novakovic | Alex C. Snoeren | Kimberly Keeton

Google UC San Diego

Rising DRAM cost in data centers

- DRAM cost scaling has nearly flattened
- **Majority of the server cost** in data centers

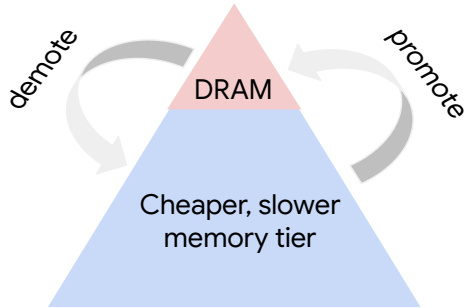




https://jcmit.net/memoryprice.htm

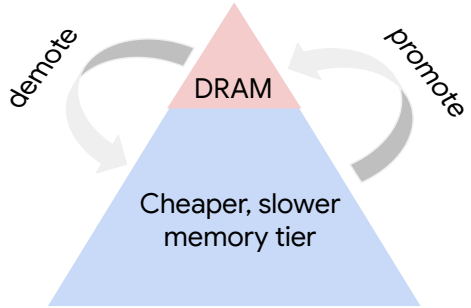
Cost saving: Offload cold data

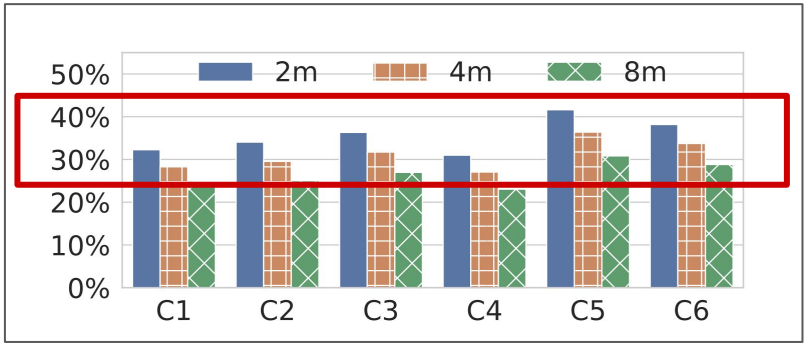
- Cheaper substrates such as a compressed DRAM tier (zswap) or SSD
- But much (5-50x) slower → Offload infrequently accessed or “cold” data



Cost saving: Offload cold data

- Cheaper substrates such as a compressed DRAM tier (zswap) or SSD
- But much (5-50x) slower ⟶ Offload infrequently accessed or “cold” data
- **20–40% of the data is very cold!**





Cold memory % across Google’s clusters [ASPLOS’ 19]

Production-scale DRAM offloading

Google Software-defined Far Memory (g-swap) [ASPLOS ’19]

Meta Transparent Memory Offloading [ASPLOS ‘22]

Production-scale DRAM offloading

Google Software-defined Far Memory (g-swap) [ASPLOS '19]

Meta Transparent Memory Offloading [ASPLOS '22]

- With **OS paging** and **LRU**, they show **20-30%** memory savings

Production-scale DRAM offloading

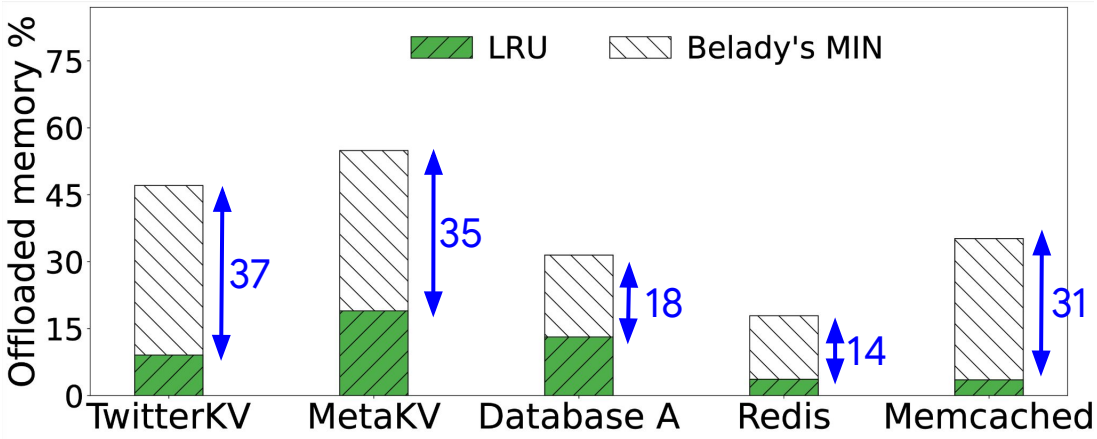
Google Software-defined Far Memory (g-swap) [ASPLOS '19]

Meta Transparent Memory Offloading [ASPLOS '22]

- With **OS paging** and **LRU**, they show **20-30%** memory savings
- LRU is generally good but we can do better

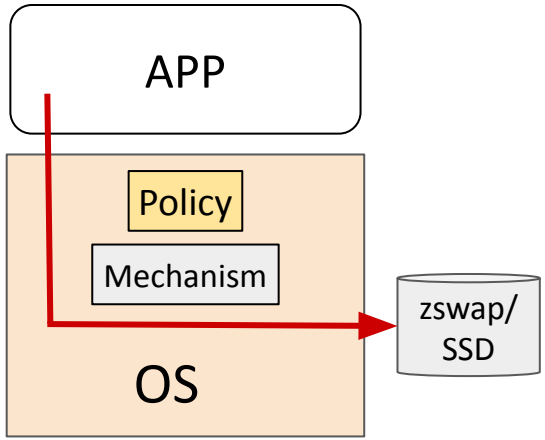
Examples
LRU-K
LIRS
ARC
Hyperbolic [ATC '17]
LHD [NSDI '18]
LRB [NSDI '20]

There is more room for policy improvement



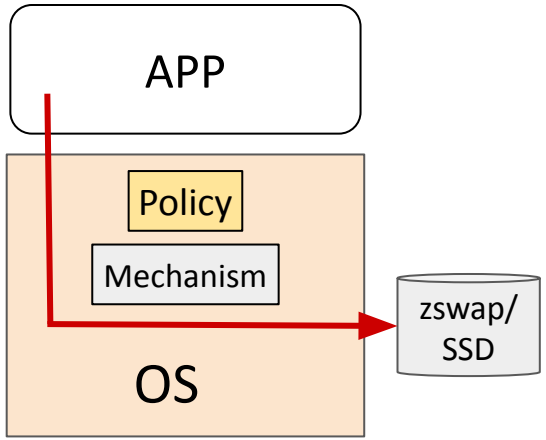
Simulated Optimal-LRU gap for real workload traces at **<1%** miss rate.

Challenge: OS-based paging policy



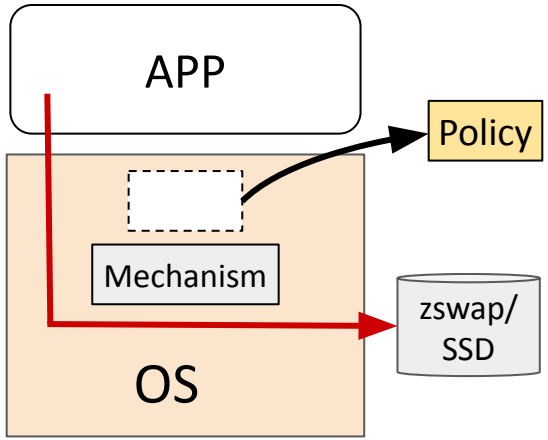
	OS
No app changes	✅
Re-use swap stack	✅

Challenge: OS-based paging policy



	OS
Fungible policy	❌
No app changes	✅
Re-use swap stack	✅

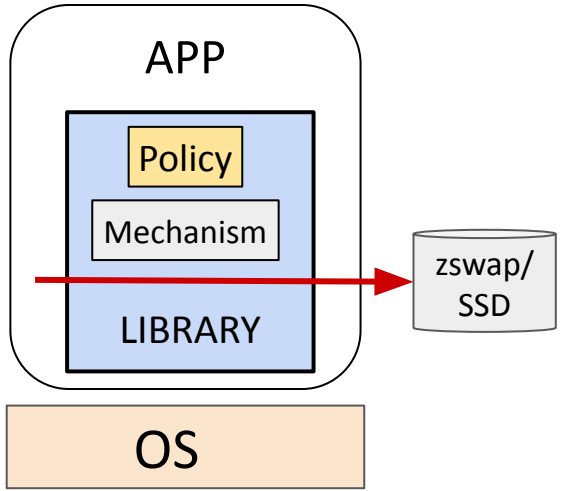
Challenge: OS-based paging policy



	OS
Fungible policy	❌
No app changes	✅
Re-use swap stack	✅

Solution: Bring policy of OS (code)

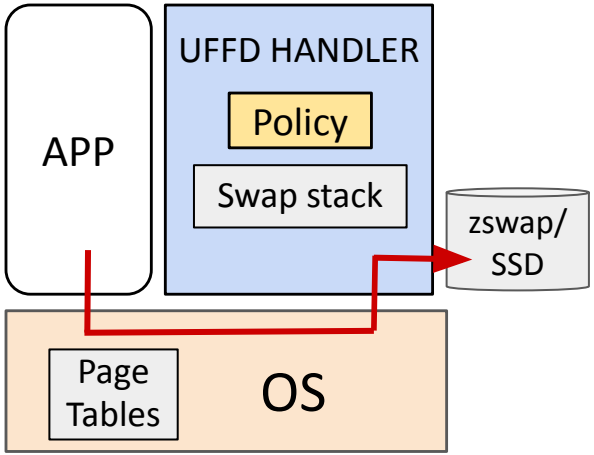
1. Custom libraries are not practical (yet)



	OS	LIB
Fungible policy	❌	✅
No app changes	✅	❌
Re-use swap stack	✅	

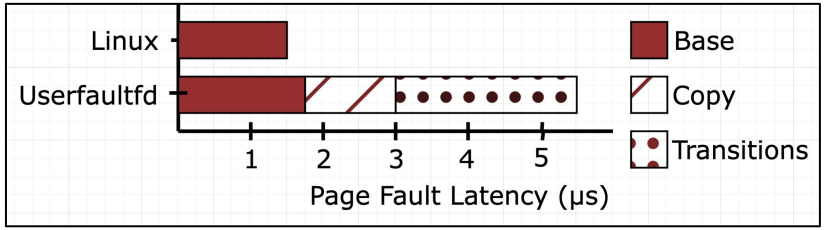
AIFM [OSDI '20]
Mira [OSDI '20]
TrackFM [ASPLOS '24]

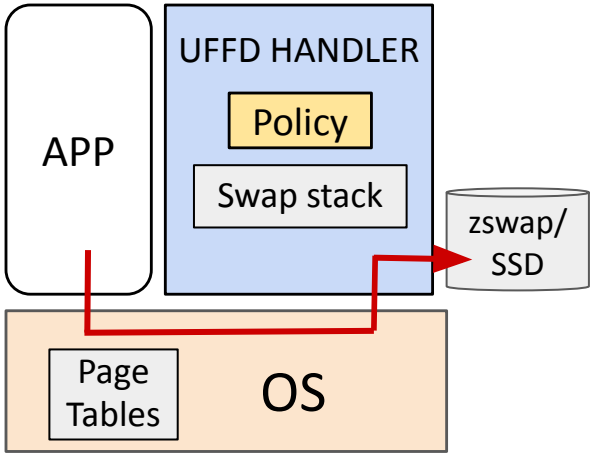
2. User-level paging (userfaultfd)



	OS	LIB	UFFD
Fungible policy	❌	✅	✅
No app changes	✅	❌	✅
Re-use swap stack	✅		❌

2. User-level paging (userfaultfd)

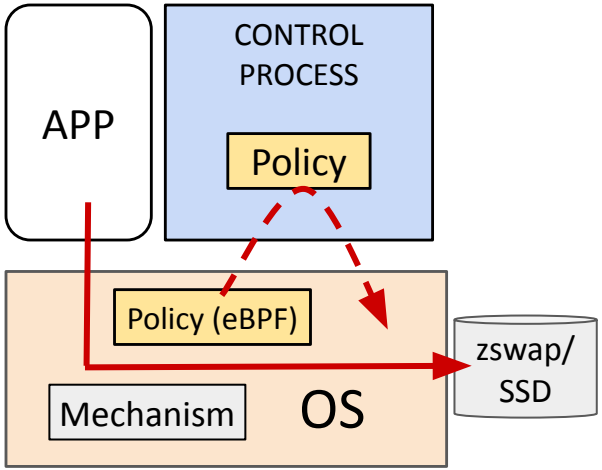




	OS	LIB	UFFD
Fungible policy	❌	✅	✅
No app changes	✅	❌	✅
Re-use swap stack	✅		❌
No overhead	✅		❌

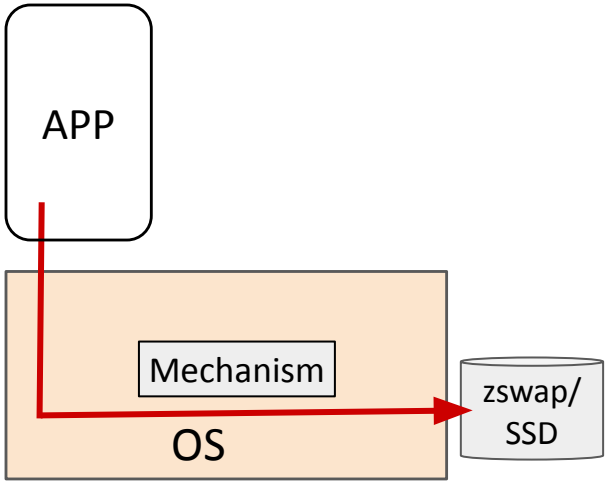
Can we externalize policy while still keeping the mechanisms intact?

Our answer: PageFlex

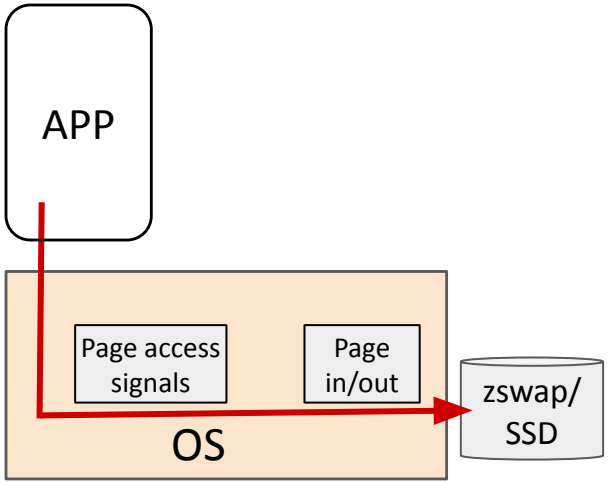


	OS	LIB	UFFD	PageFlex
Fungible policy	❌	✅	✅	✅
No app changes	✅	❌	✅	✅
Re-use swap stack	✅		❌	✅
No overhead	✅		❌	✅

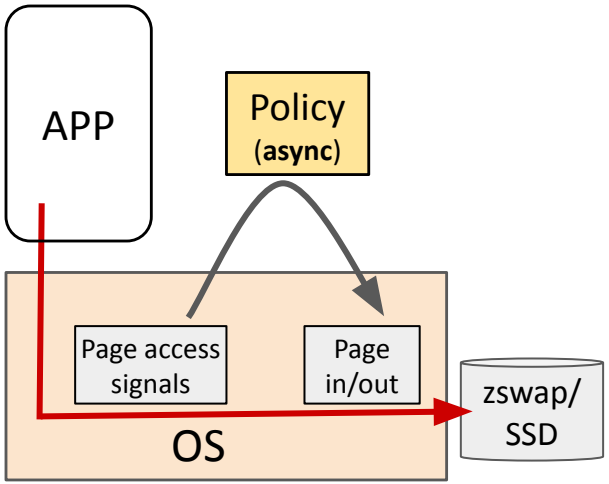
Observation #1



Observation #1

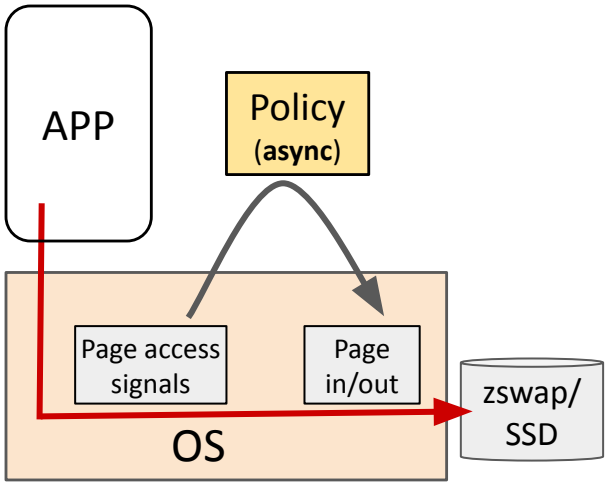


Observation #1



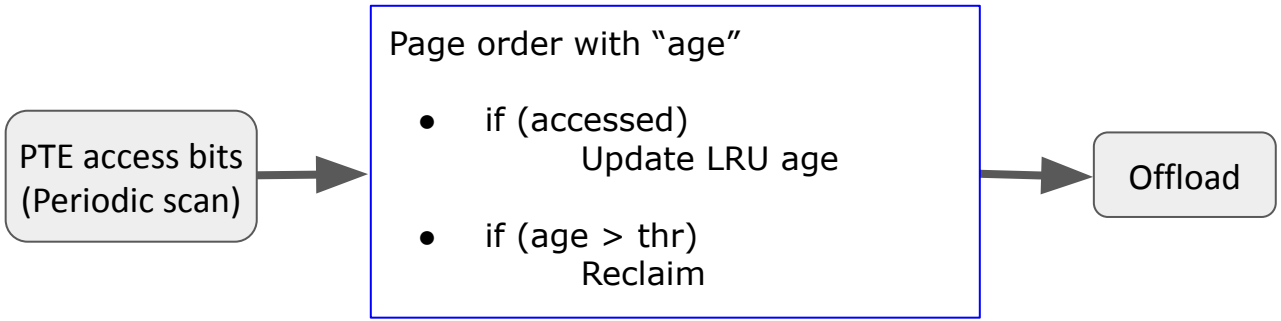
1. Only delegate asynchronous policy
   a. On-demand policy intact but rarely used
   b. Proactive offloading and Prefetching

Observation #1



1. Only delegate asynchronous policy
   a. On-demand policy intact but rarely used
   b. Proactive offloading and Prefetching
2. What do most policies need?

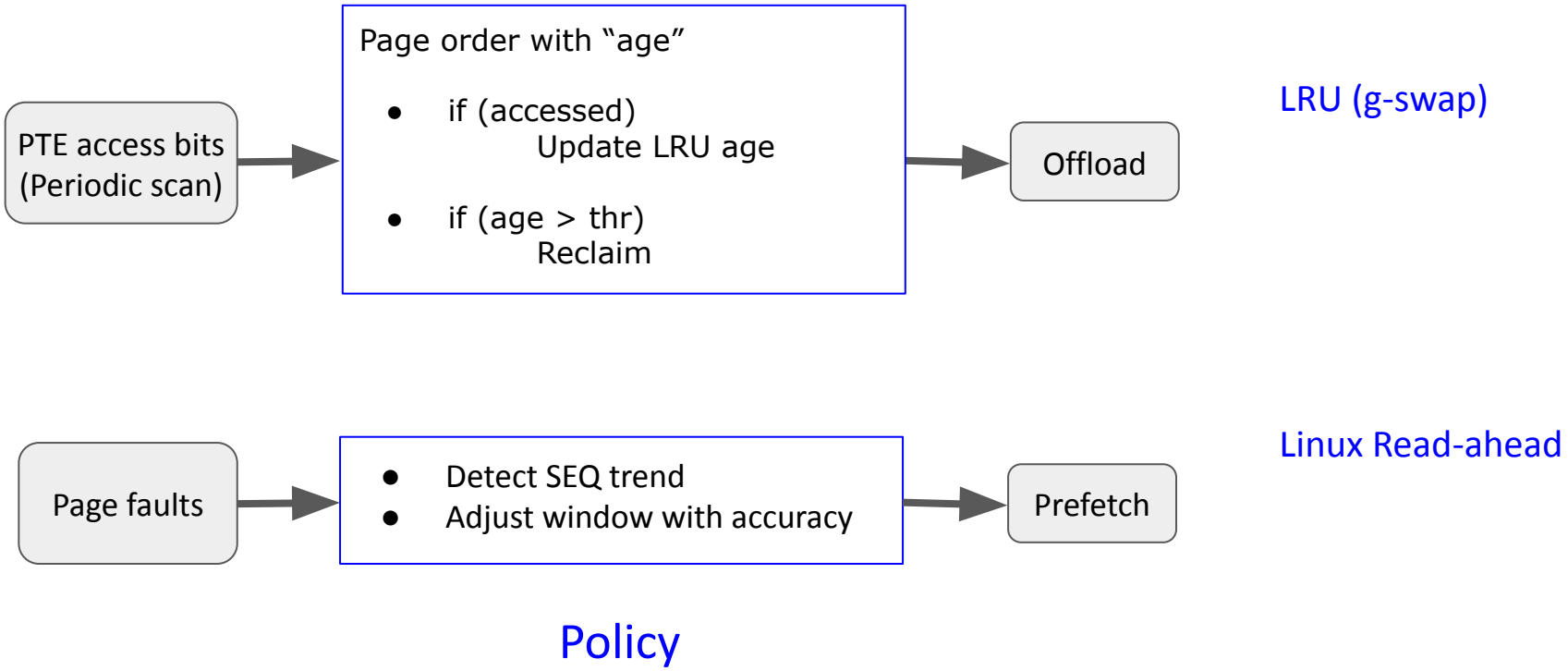
How current policies work



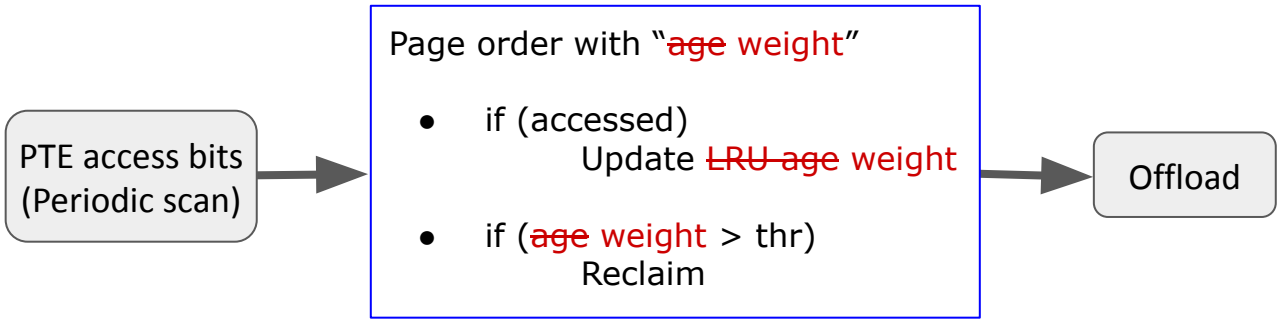
Policy

LRU (g-swap)

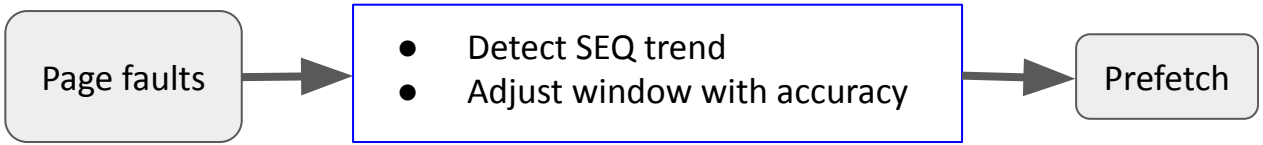
How current policies work



How ~~current~~ most policies work



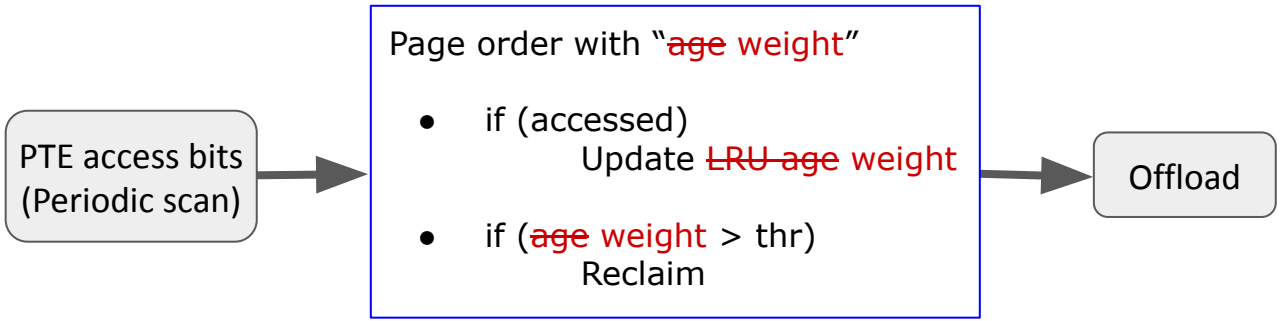
LRU (g-swap)
LFU
Hyperbolic [ATC '17]



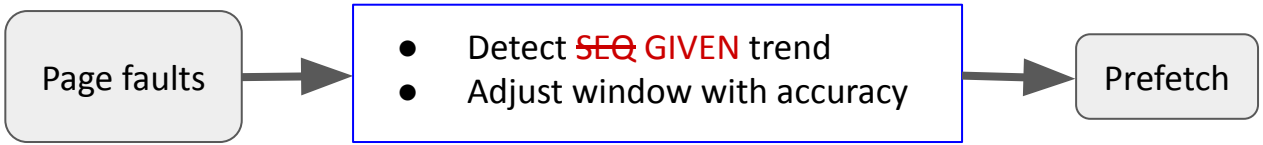
Linux Read-ahead

Policy

How ~~current~~ most policies work



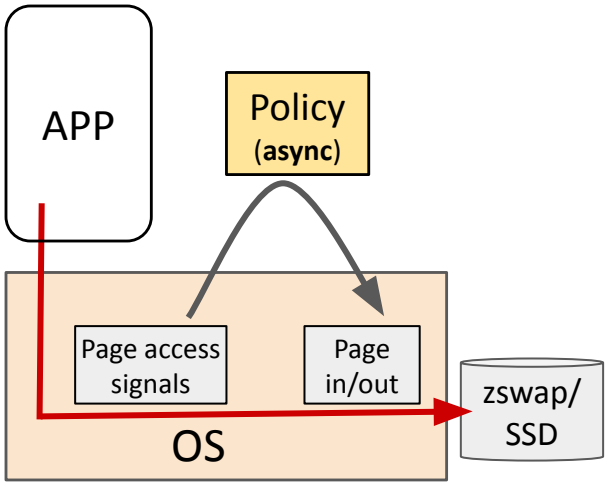
LRU (g-swap)
LFU
Hyperbolic [ATC '17]



Linux Read-ahead
Leap [ATC '20]

Policy

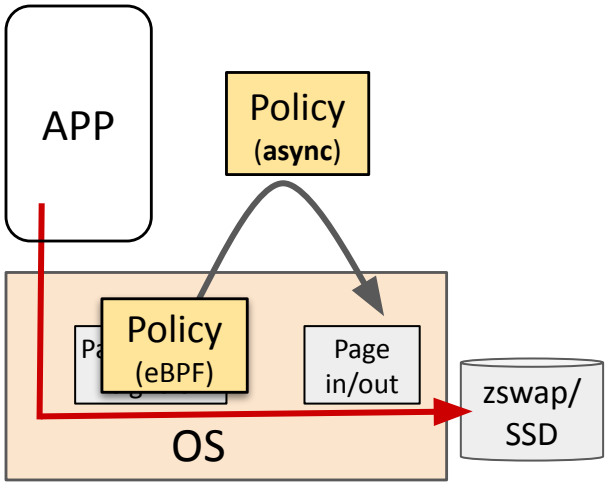
Observation #2



1. Only delegate asynchronous policy
   a. On-demand policy intact but rarely used
   b. Proactive offloading and Prefetching

2. What do most policies need?
   a. Customization at a few well-defined events
   b. With some per-page state

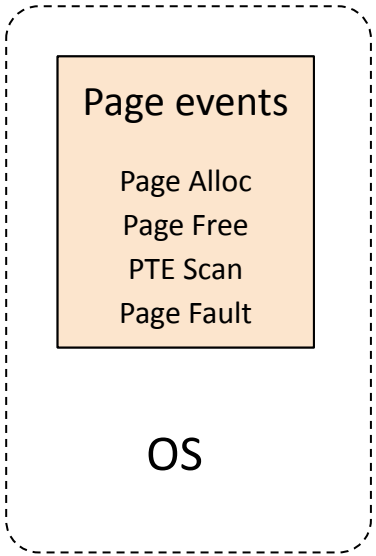
Observation #2



1. Only delegate asynchronous policy
   a. On-demand policy intact but rarely used
   b. Proactive offloading and Prefetching
2. What do most policies need?
   a. Customization at a few well-defined events
   b. With some per-page state

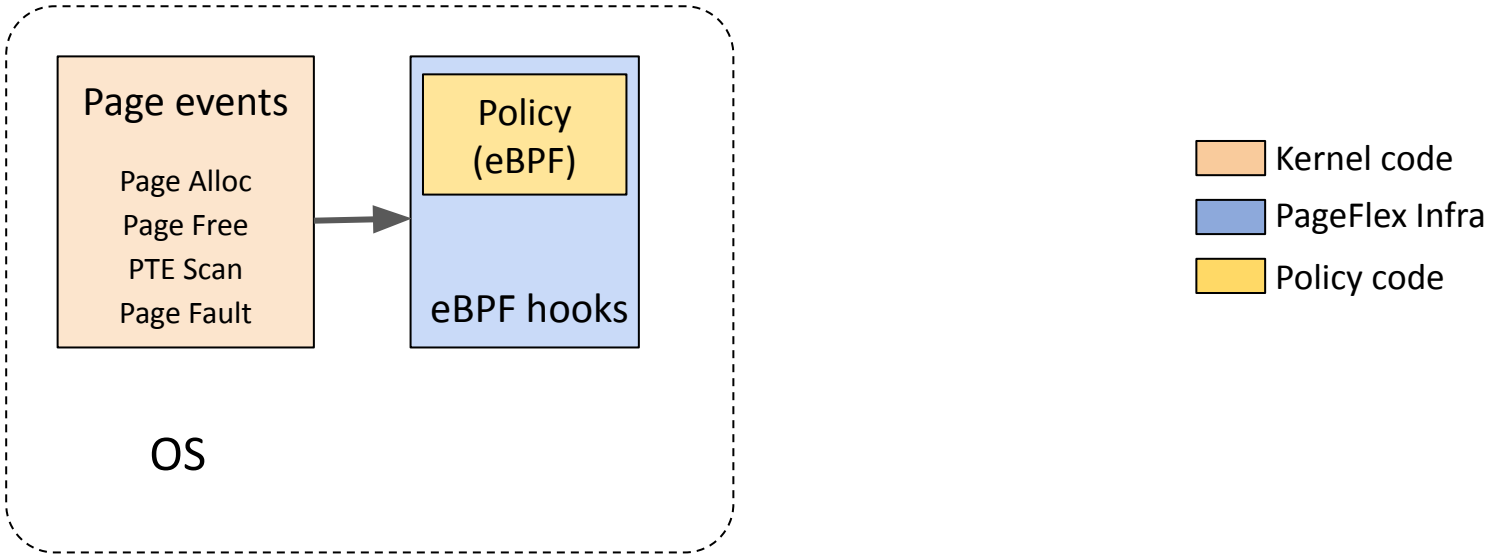
⟶ **Low-overhead customization with eBPF**

System overview

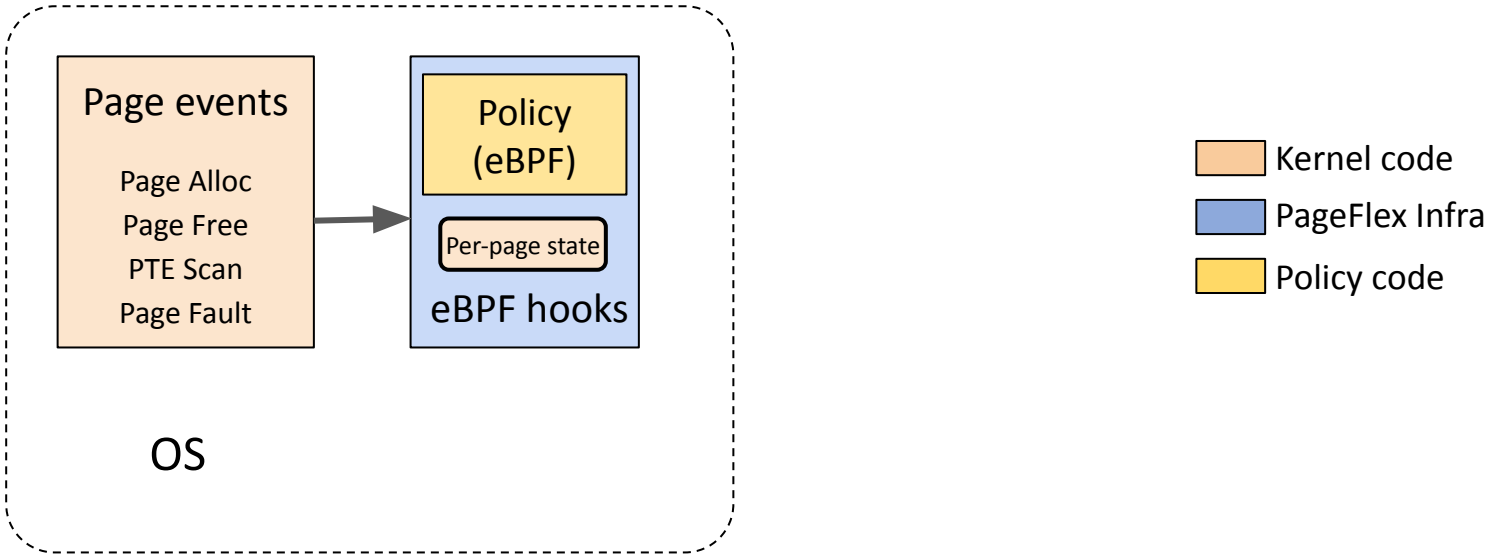


Kernel code

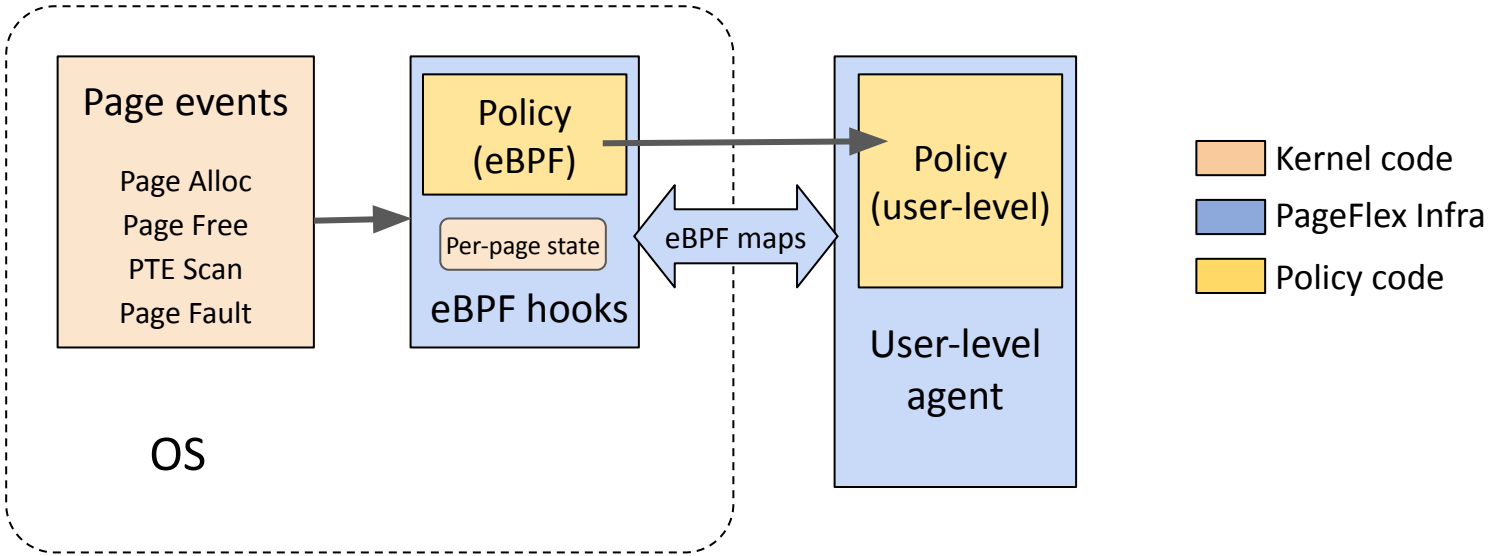
System overview



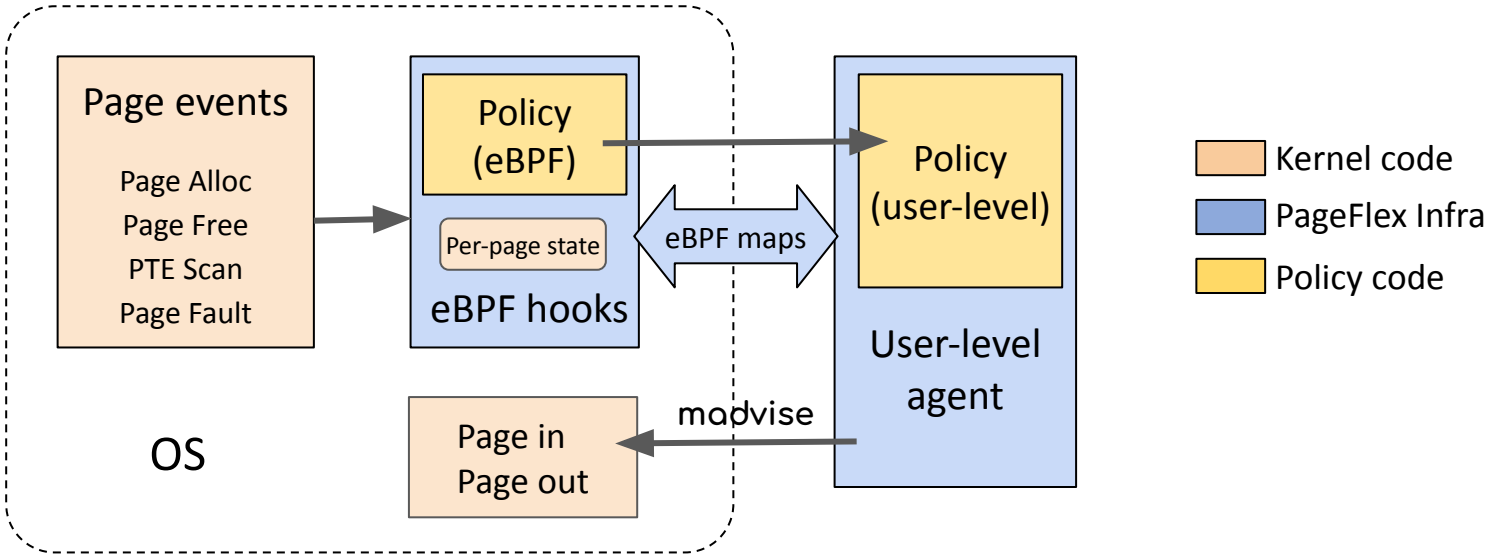
System overview



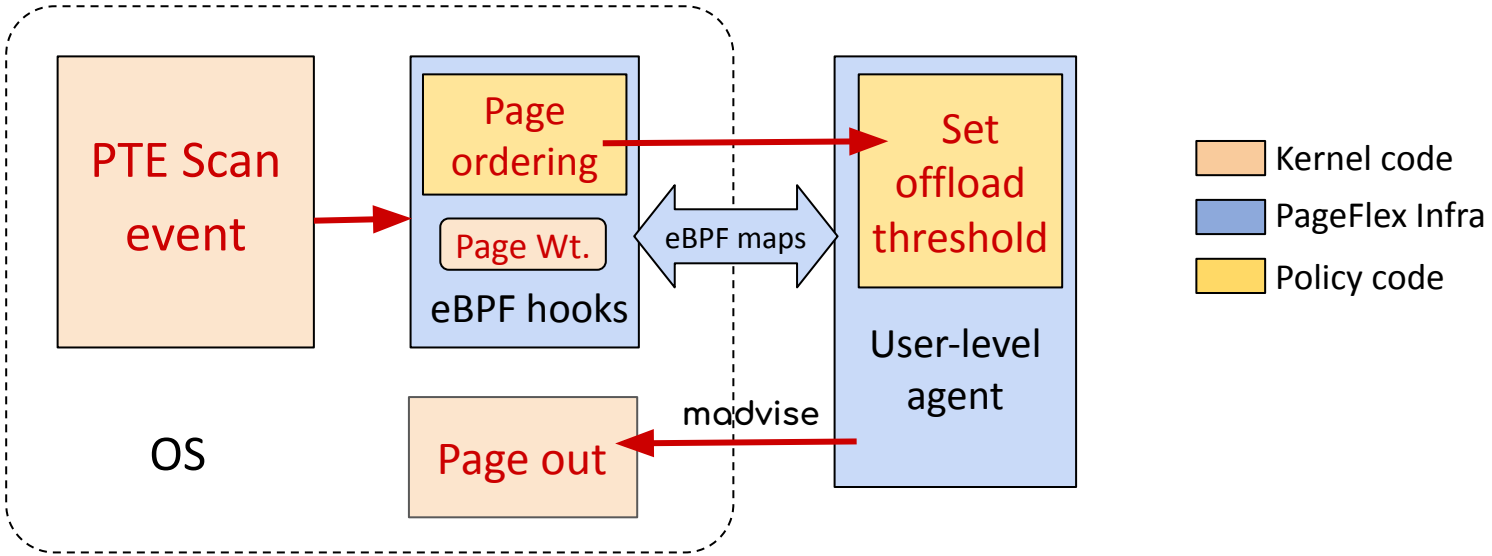
System overview



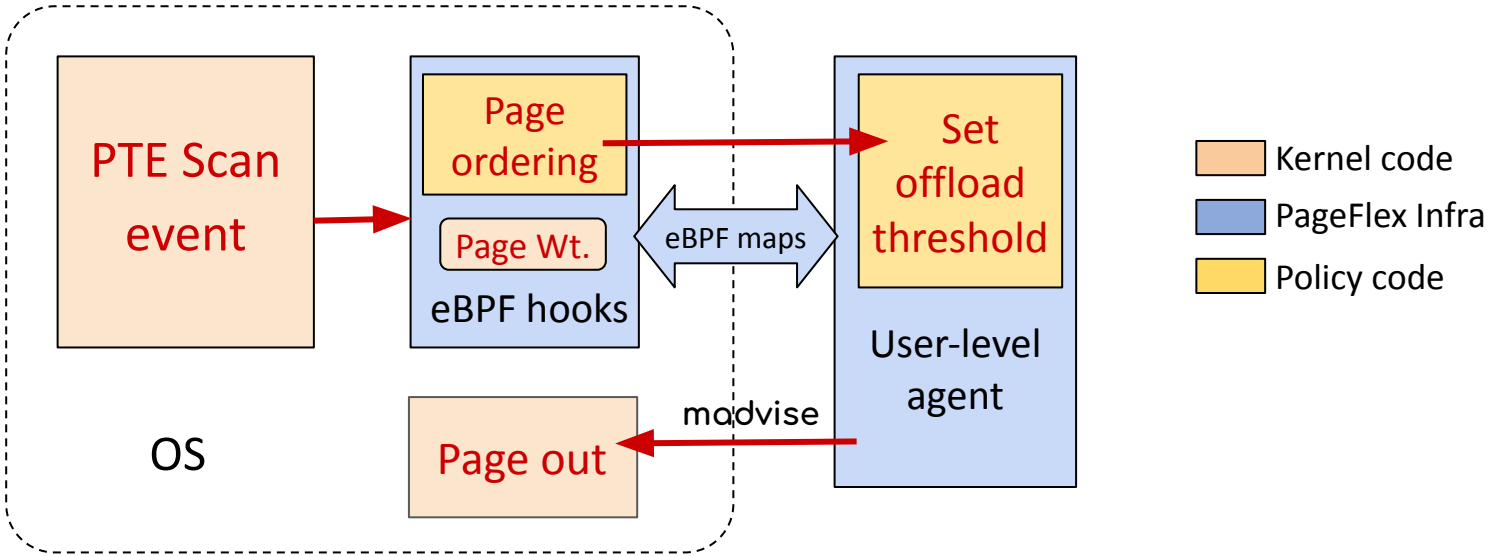
System overview



General data offload policy

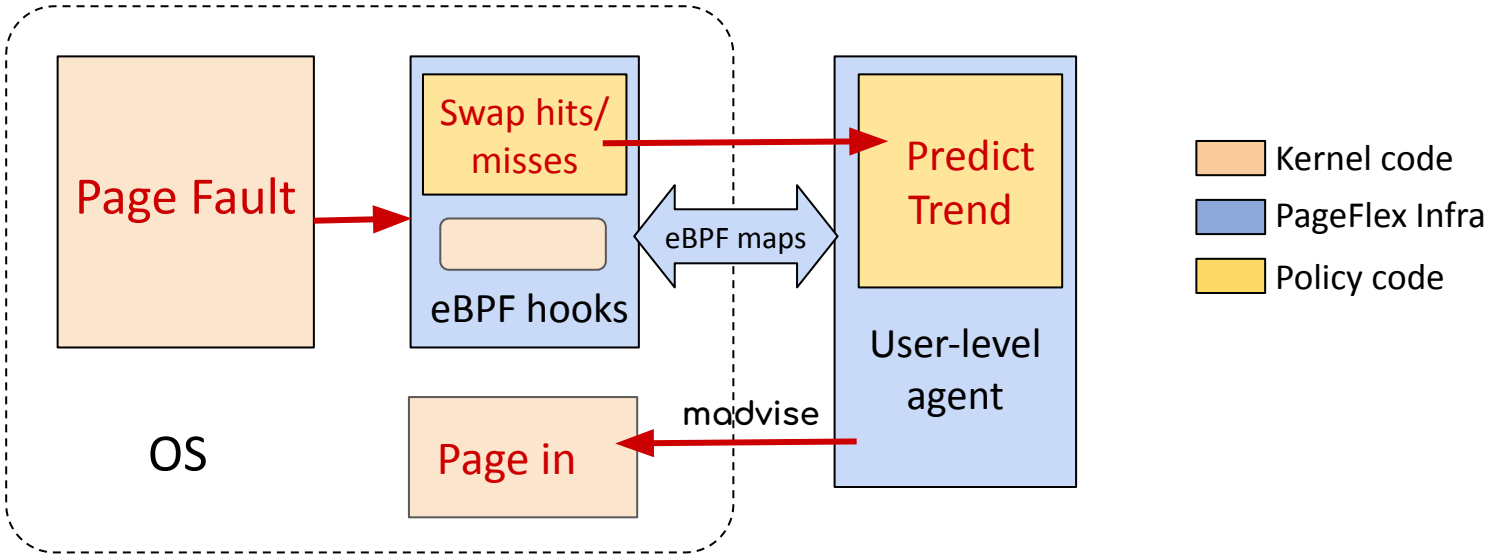


General data offload policy

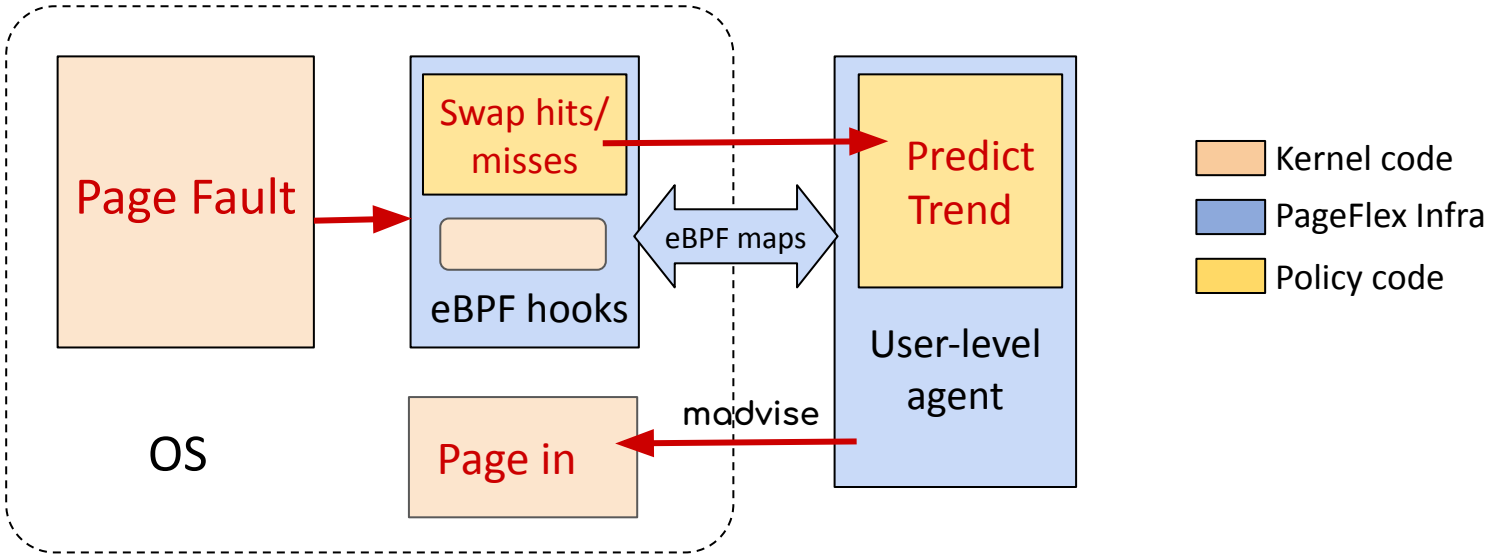


LRU, LFU, Hyperbolic [ATC'17] in ~10 lines

General prefetching policy

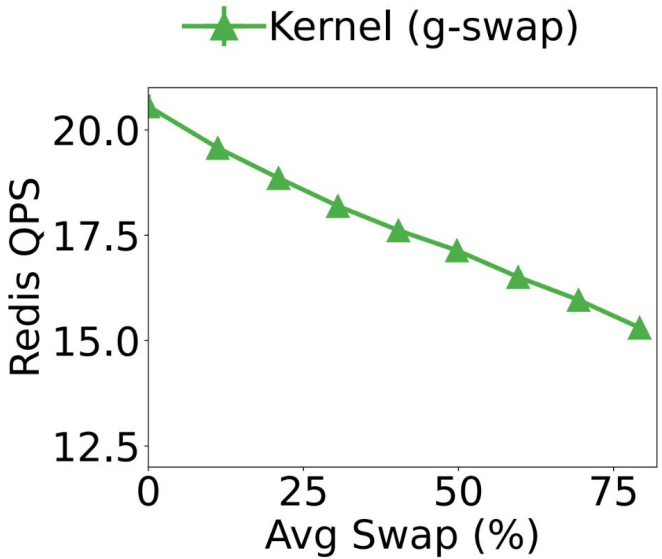


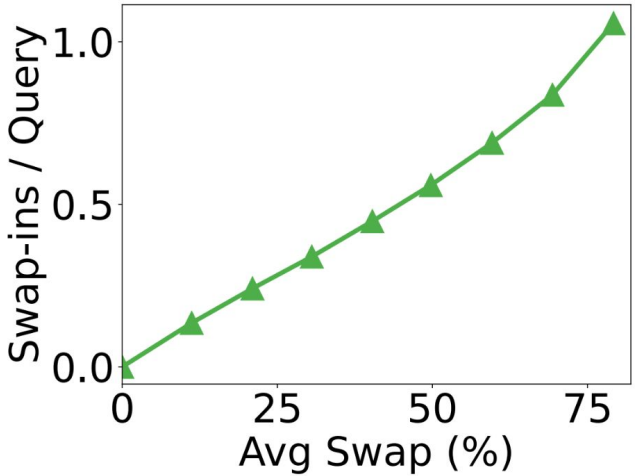
General prefetching policy



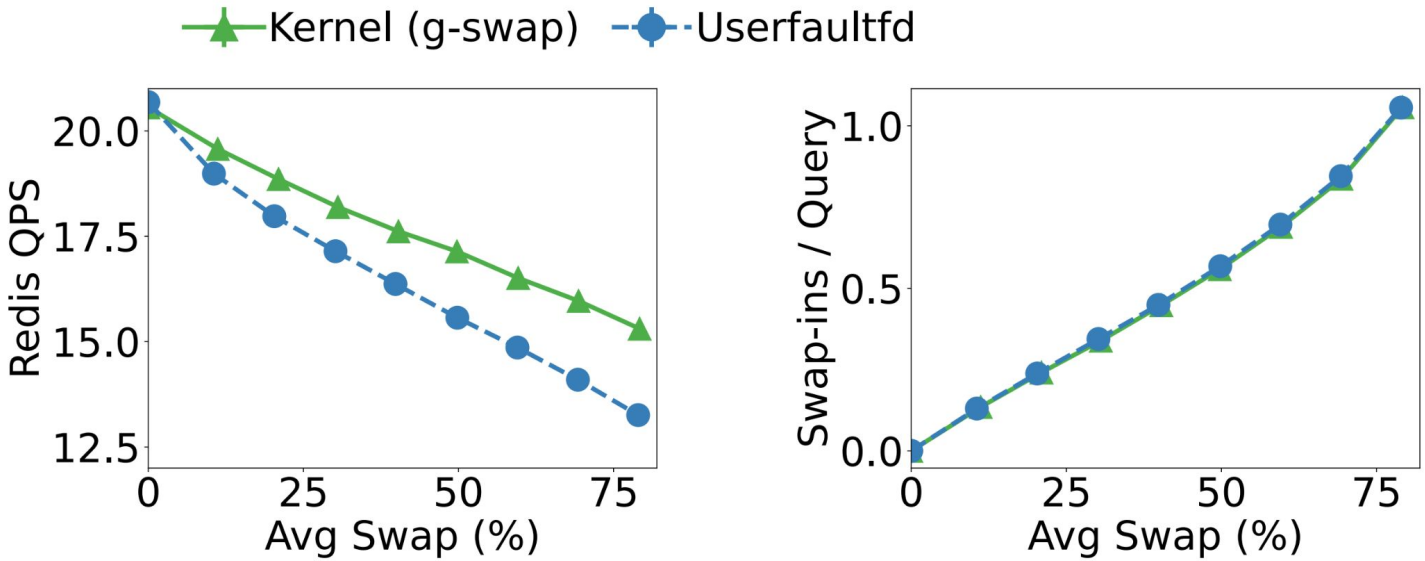
Leap [ATC '20] in ~180 lines (160 original)

Performance overhead (with LRU)

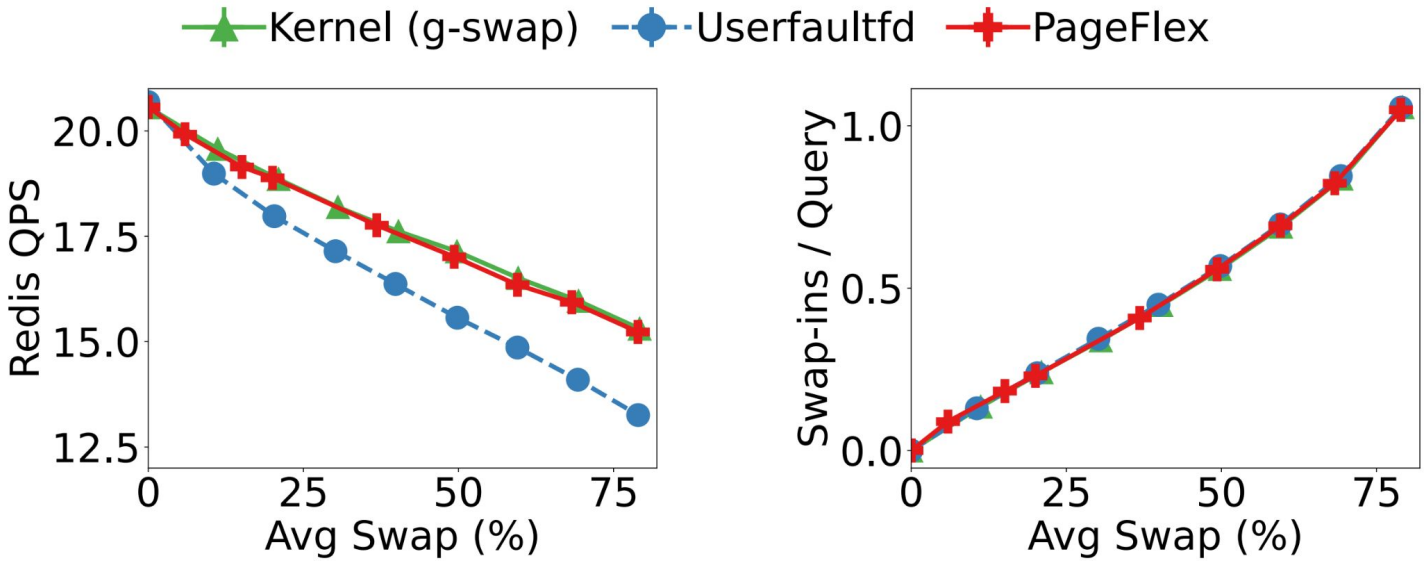




Performance overhead (with LRU)



Performance overhead (with LRU)

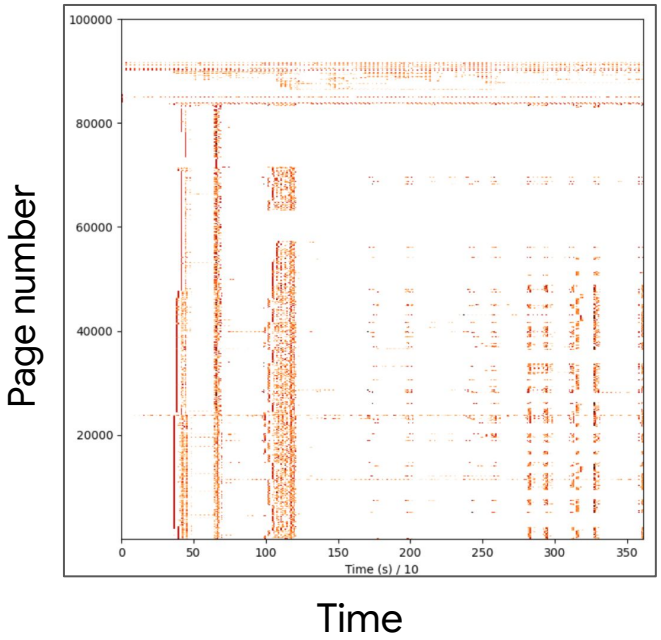


PageFlex adds very little (<1%) overhead!

Further policy specialization

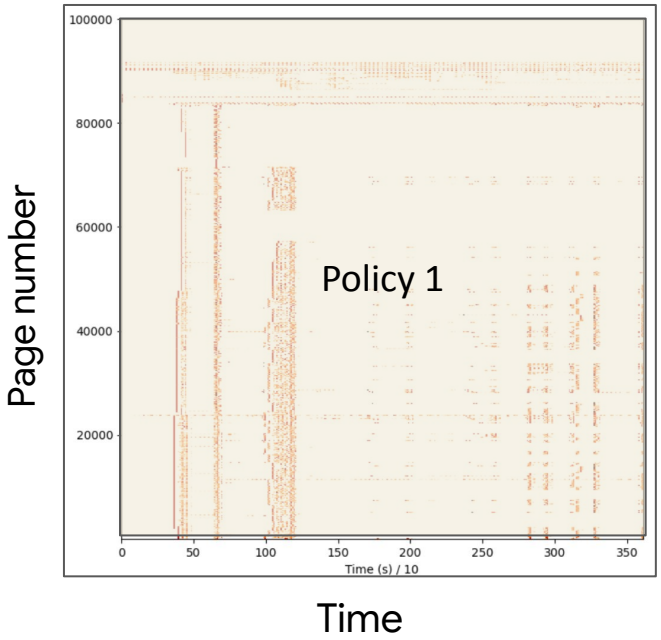
Further policy specialization

Key Idea:
Access patterns vary over space and time.



Further policy specialization

Key Idea:
Access patterns vary over space and time.
And so can benefit from different sub-policies.

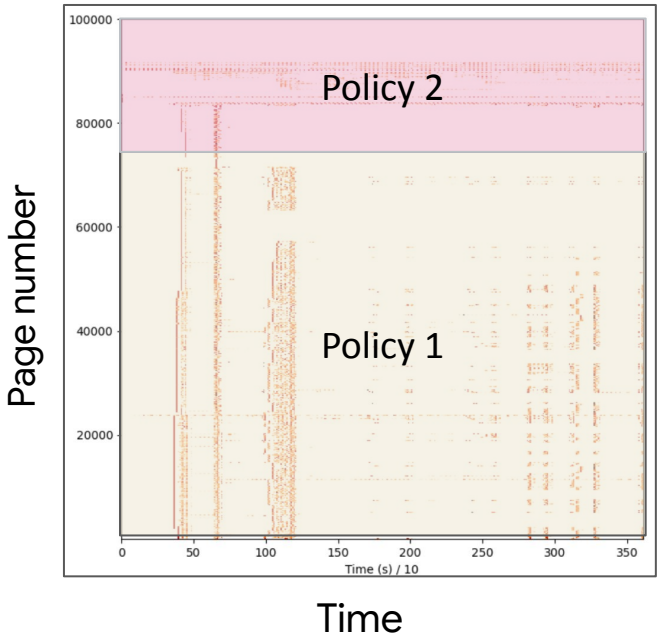


Further policy specialization

Key Idea:
Access patterns vary over space and time.
And so can benefit from different sub-policies.

Examples:

- Region-aware:
  Single LRU threshold ⟶ Per-region threshold



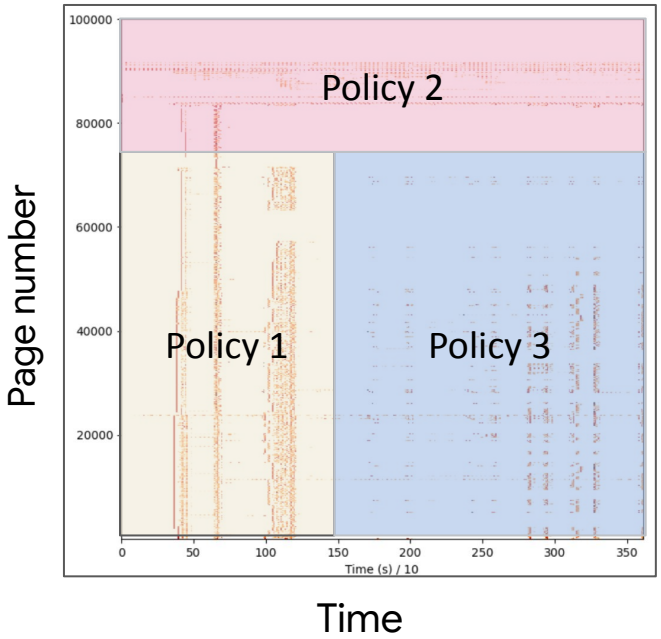
Further policy specialization

Key Idea:
Access patterns vary over space and time.
And so can benefit from different sub-policies.

Examples:

- Region-aware:
  Single LRU threshold ⟶ Per-region threshold

- Phase-aware:
  LRU always ⟶ Switch LRU/MRU

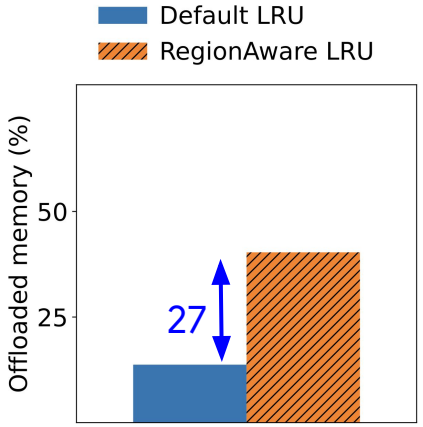


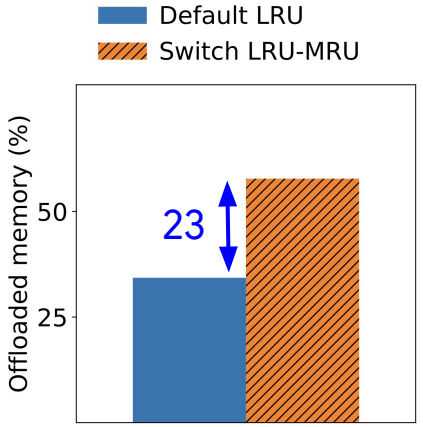
Further policy specialization

Key Idea:
Access patterns vary over space and time.
And so can benefit from different sub-policies.

Examples:

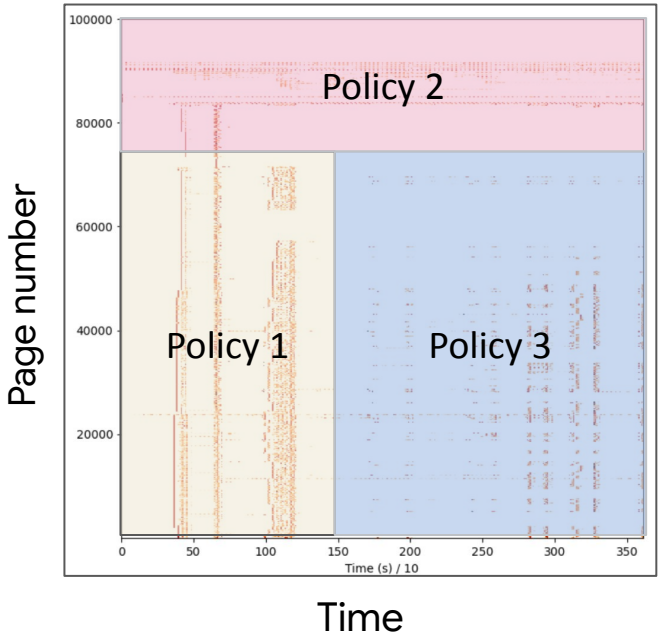
- Region-aware:
  Single LRU threshold ⟶ Per-region threshold
- Phase-aware:
  LRU always ⟶ Switch LRU/MRU





Conclusion

- Paging policies dictate memory savings
- OS-based policies are hard to evolve
- **PageFlex cleanly separates policy**
  - Fully-compatible (with existing deployments)
  - Low-overhead



Google

UC San Diego