P2Cache: An Application-Directed Page Cache for Improving Performance of Data-Intensive Applications

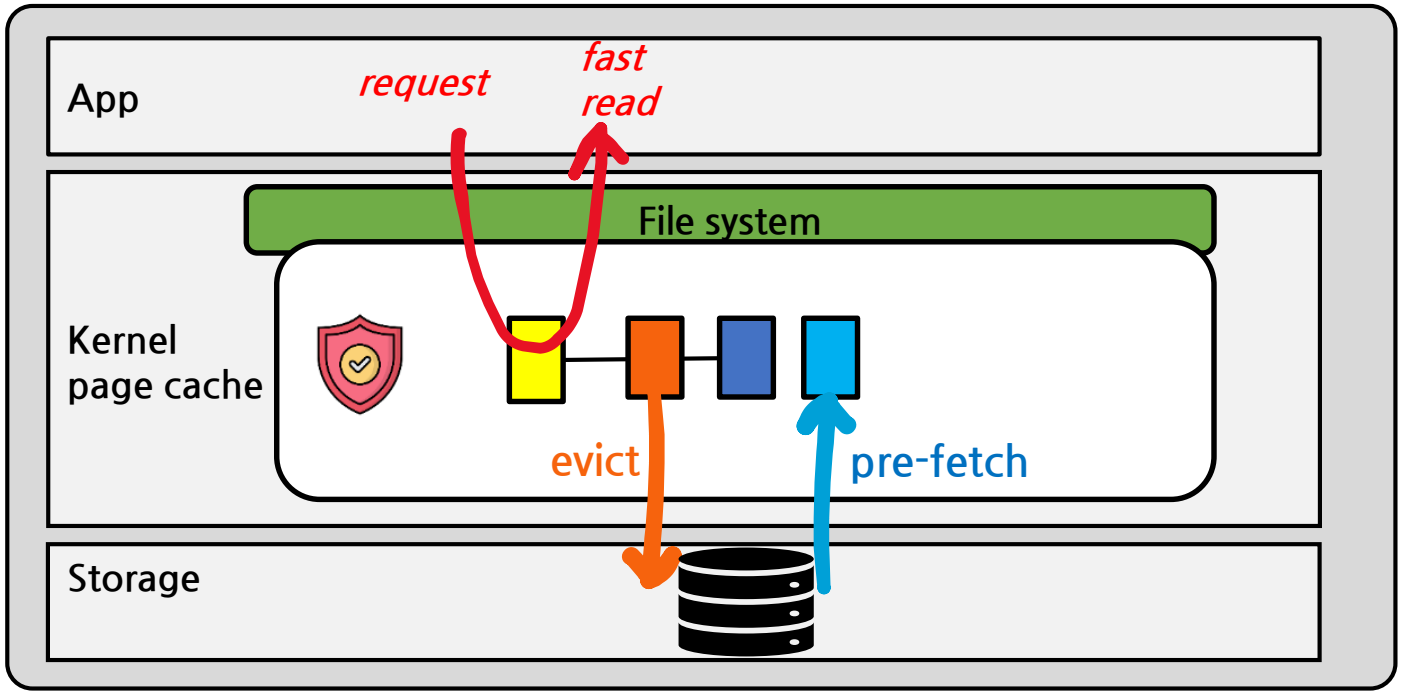
Dusol Lee[1], Inhyuk Choi[1], Chanyoung Lee[1], Sungjin Lee[2], and Jihong Kim[1]

Seoul National University[1], DGIST[2]

The 15th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage'23)

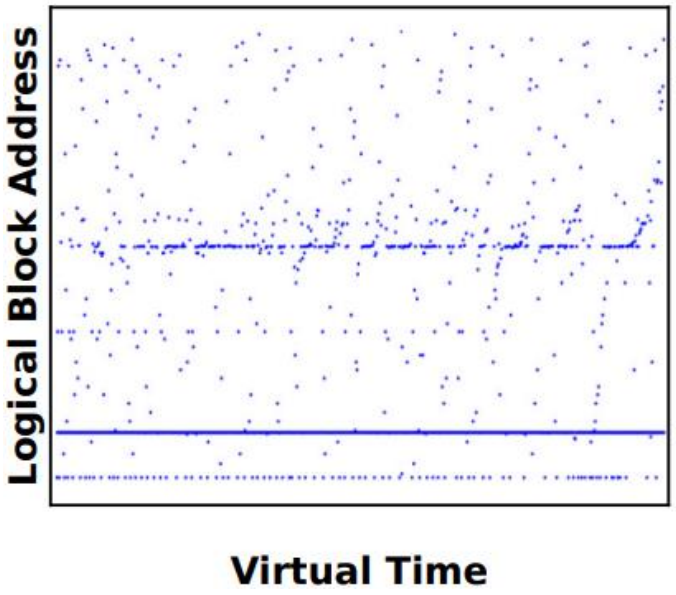
OS-level Page Caches for Data-Intensive Applications

- **A kernel-level page cache** plays a crucial role in the performance of **I/O-intensive applications**.
  - Effective in reducing the amount of data transfers between an SSD and the host DRAM.
- Page cache **supports safe caching mechanism** with policies
  - Policies: page eviction, pre-fetch, swap,..

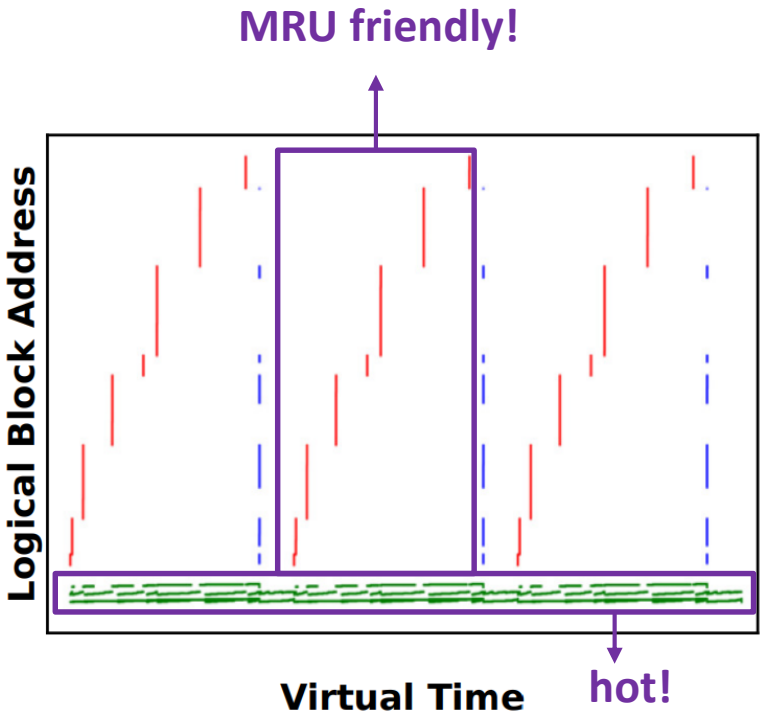


Limitations of OS-level Page Caches

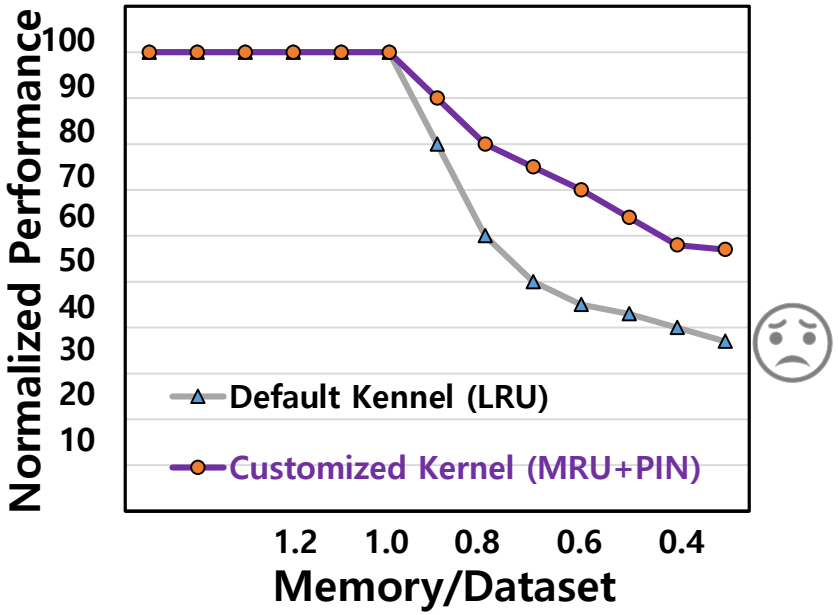
- Key Weakness: One Size Doesn't Fit All
  - **Highly customized algorithms** in data-intensive apps**.**
  - Common-case-based policies mismatches with application I/O characteristics.



I/O patterns of GraphWalker



I/O patterns of Lumos



I/O performance with varying memory sizes

Outline

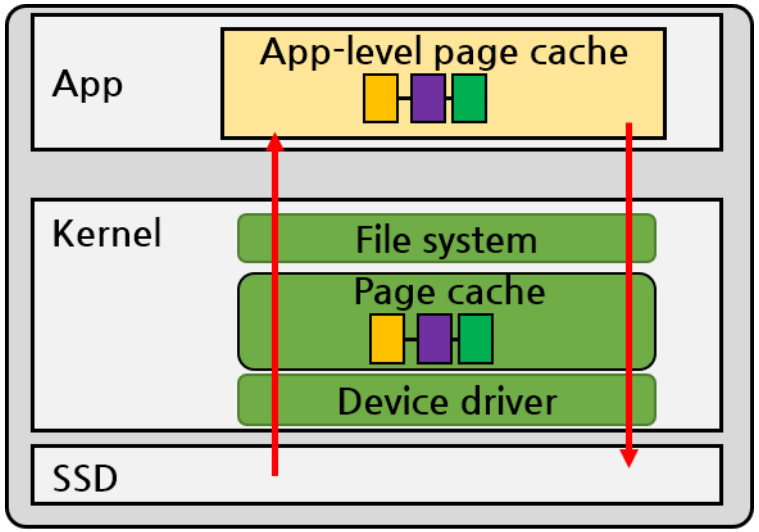
001 **Existing Solutions for Data-Intensive Applications and Limitations**

002 **P2Cache: An Application-Directed Page Cache**

003 **Evaluation & Conclusion**

Previous Solution: Application-Level Page Cache

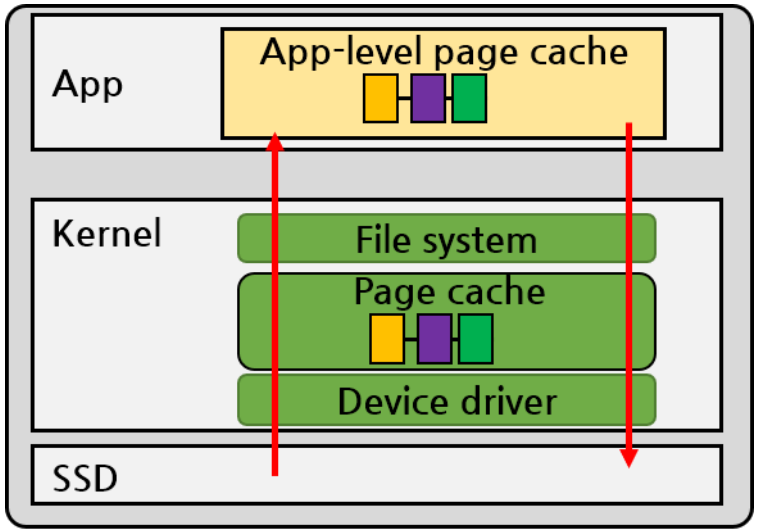
- Applications often **implement their own page caches** at the application level.
  - With kernel page cache: Can utilize kernel-supported functions such as for **data protection or consistency.**



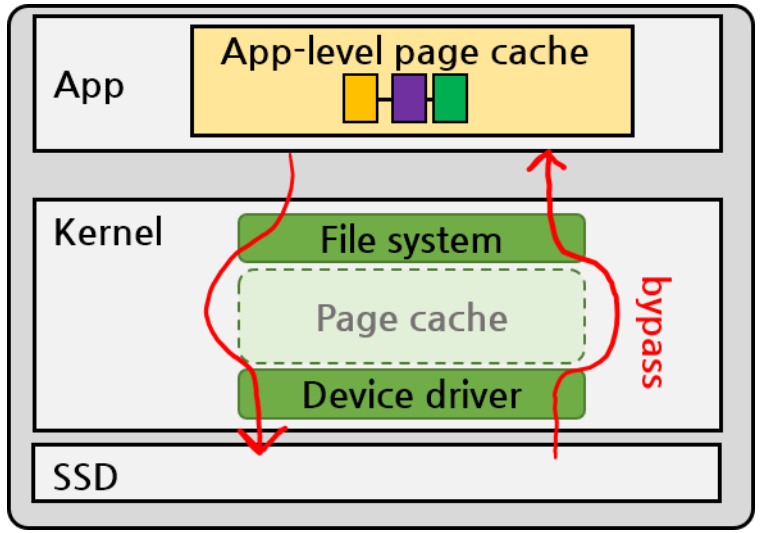
App-level page cache with kernel page cache

Previous Solution: Application-Level Page Cache

- Applications often **implement their own page caches** at the application level.
  - With kernel page cache: Can utilize kernel-supported functions such as for **data protection or consistency.**
  - Without kernel page cache: Can manage pages directly **without kernel intervention.**



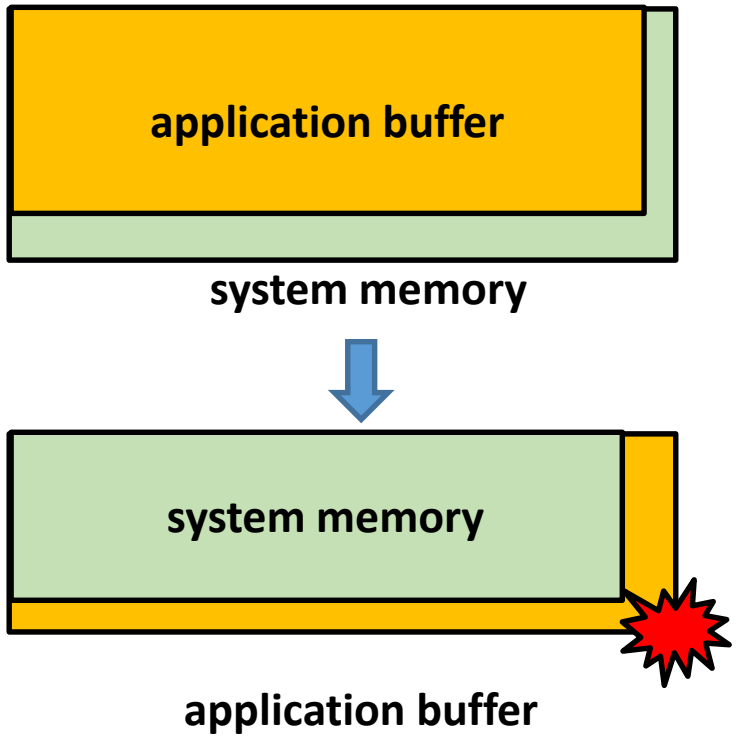
App-level page cache with kernel page cache

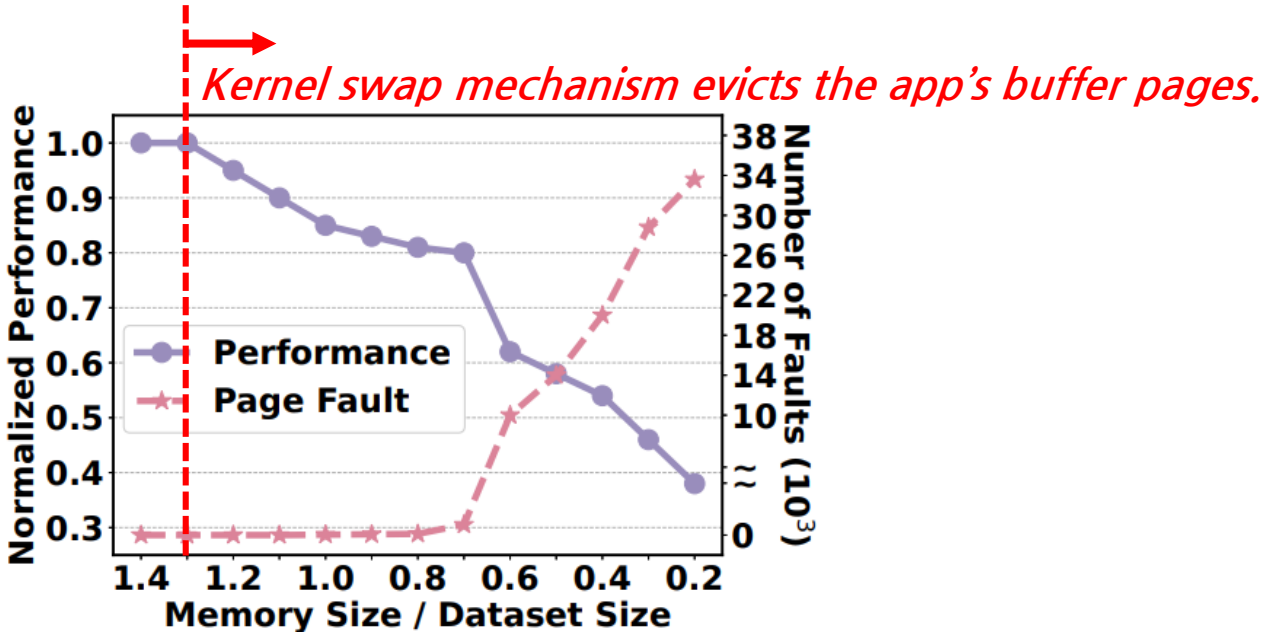


App-level page cache without kernel page cache

Limitation of Application-Level Page Cache

- Suffers from **interference** by the kernel's page cache.
  - When dataset exceeds the system memory, the kernel evicts application's pages **without app's intention**.



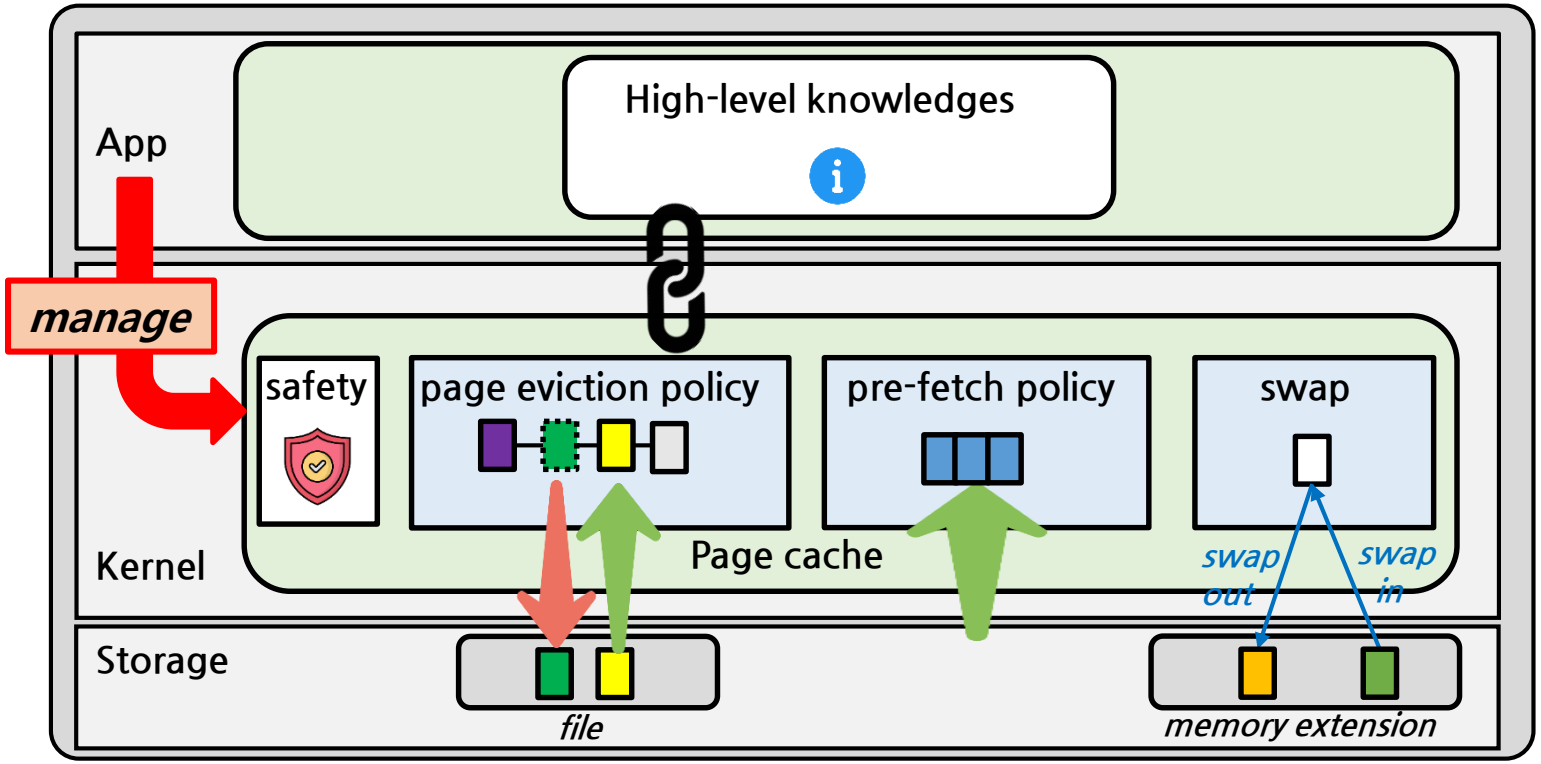


I/O performance trends with varying memory size (GraphWalker)

- **Re-implementation needed** when significant changes in either the SSD or host memory system.
- Cannot utilize kernel-supported functions such as for **data protection/consistency/share**.

Ideal Application-level Page Caches

- Need to support:
  - **Kernel-level mechanism** for page management
  - **Application-level policy** for page management



Outline

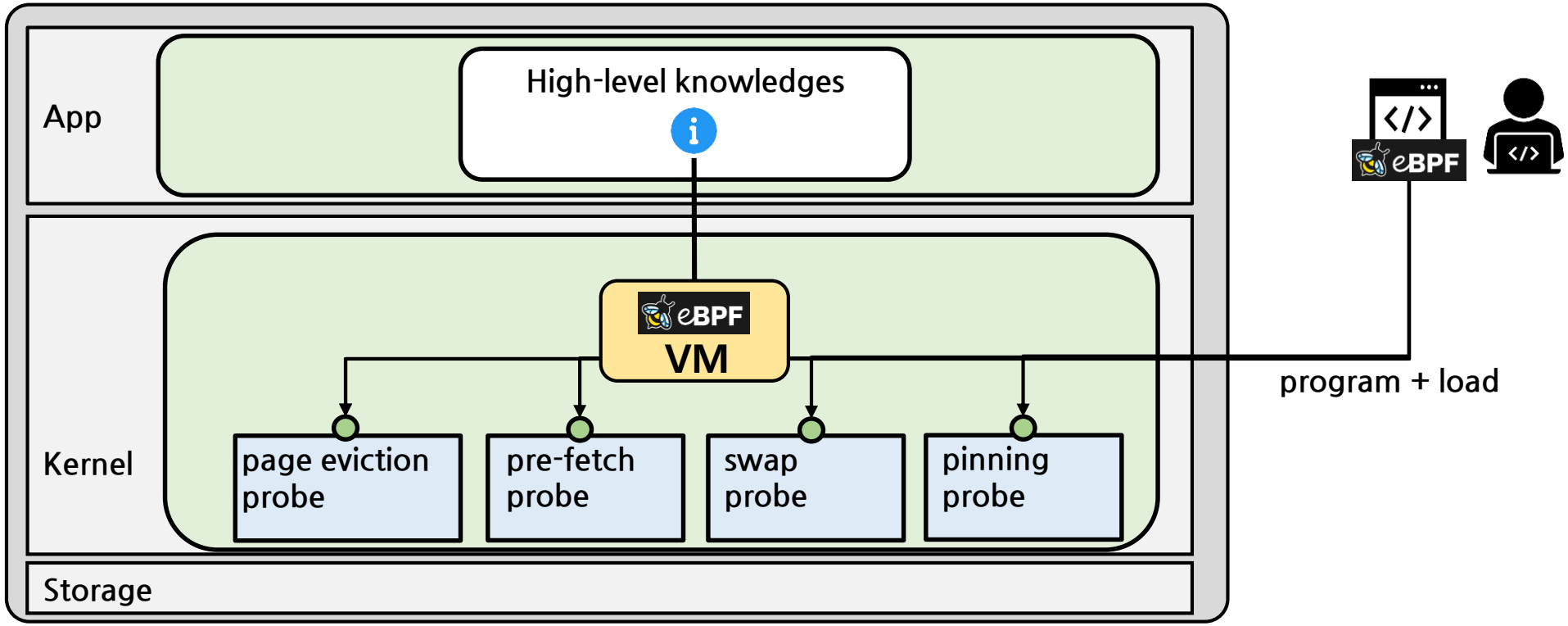
001 Existing Solutions for Data-Intensive Applications and Limitations

002 **P2Cache: An Application-Directed Page Cache**

003 Evaluation & Conclusion

P2Cache: Application-Directed Page Cache

- **An extended Linux page cache with:**
  - **Added 4 new probe points within a Linux kernel** for page cache customization.
  - **P2Cache API** for an application-level development
  - With **eBPF**, a user-defined custom policy can be run within the kernel runtime safely.



P2Cache Implementations

- **Implementaion 1: P2C API**
  - **Application developers can create customized kernel-level page caches.**

Group	Function name
Configuration	move_data_to_kernel (data, ..)
	bpf_prog_load (program, probe, ..)
	bpf_prog_unload (program, probe, ..)
Policy Implementation	get_page_list (application_name)
	get_page_data (page)
	set_eviction_list (page)
	⋮

❶ Move application data to the kernel-protected memory

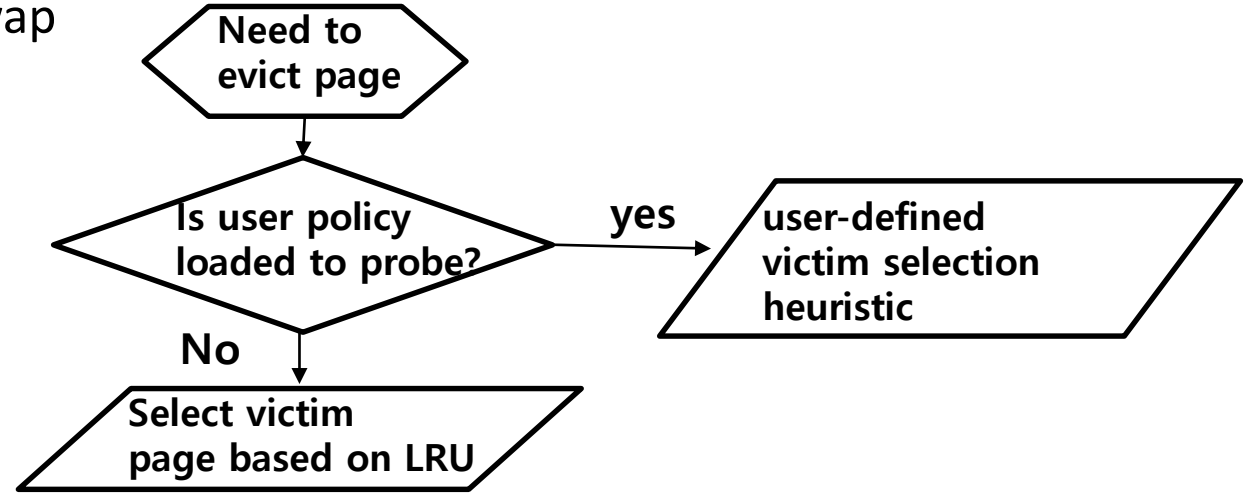
❸ Load/Unload eBPF program into the probe point

❷ eBPF program development API

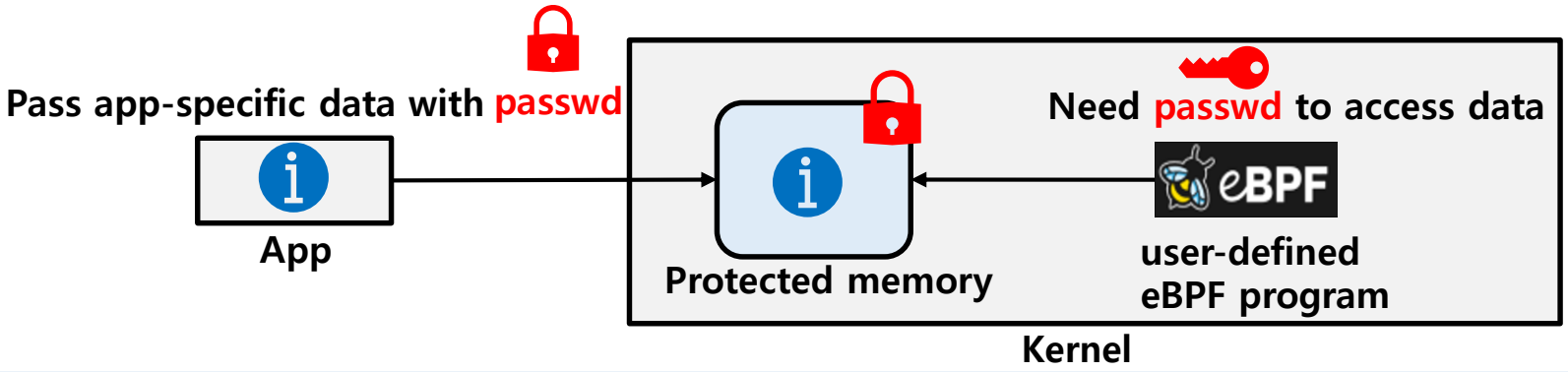
Functions in P2C API

P2Cache Implementations

- **Implementaion 2: Probe Points for Page Cache**
  - **Define 4 new probe points** which re-configure default kernel-page cache decisions to custom page cache.
    - eviction, pinning, pre-fetch, and swap

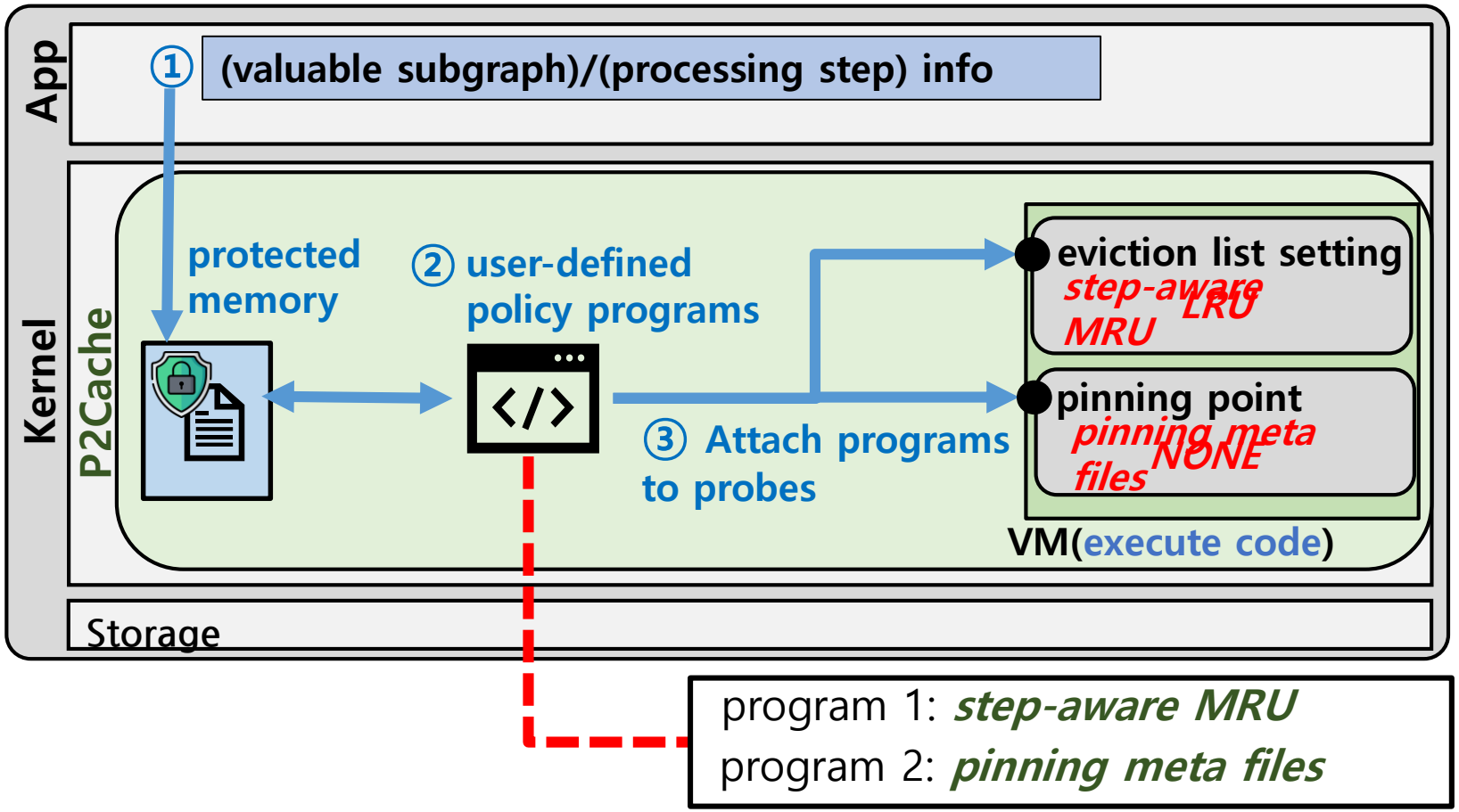


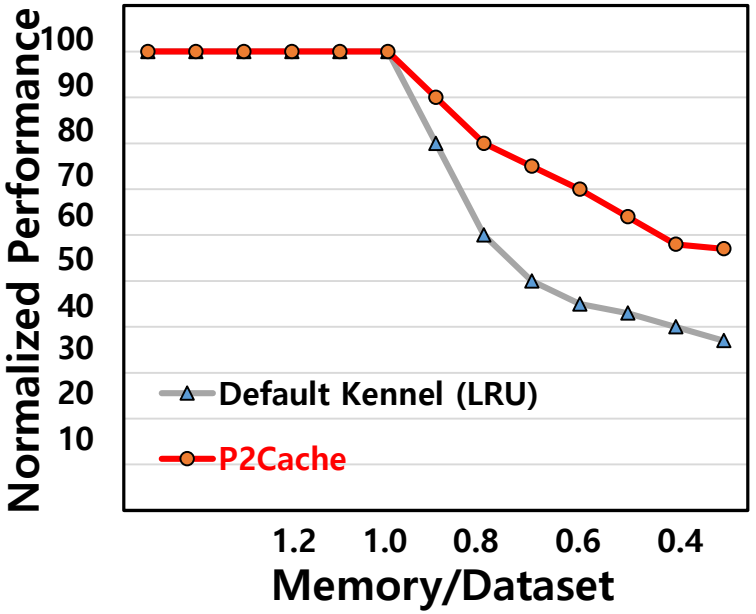
- **Implementaion 3: Per-App Kernel Access Isolation**
  - **Prevent unauthorized eBPF program** from accessing other applications' data stored in kernel.



Case Study: Custom Page Cache for Graph Processing

- Example Custom Page Cache for Graph Processing Application ‘Lumos’
  - Two customizations in **page replacement** and **pinning**





Outline

001 Existing Solutions for Data-Intensive Applications and Limitations

002 P2Cache: An Application-Directed Page Cache

003 **Evaluation & Conclusion**

Experimental Setup

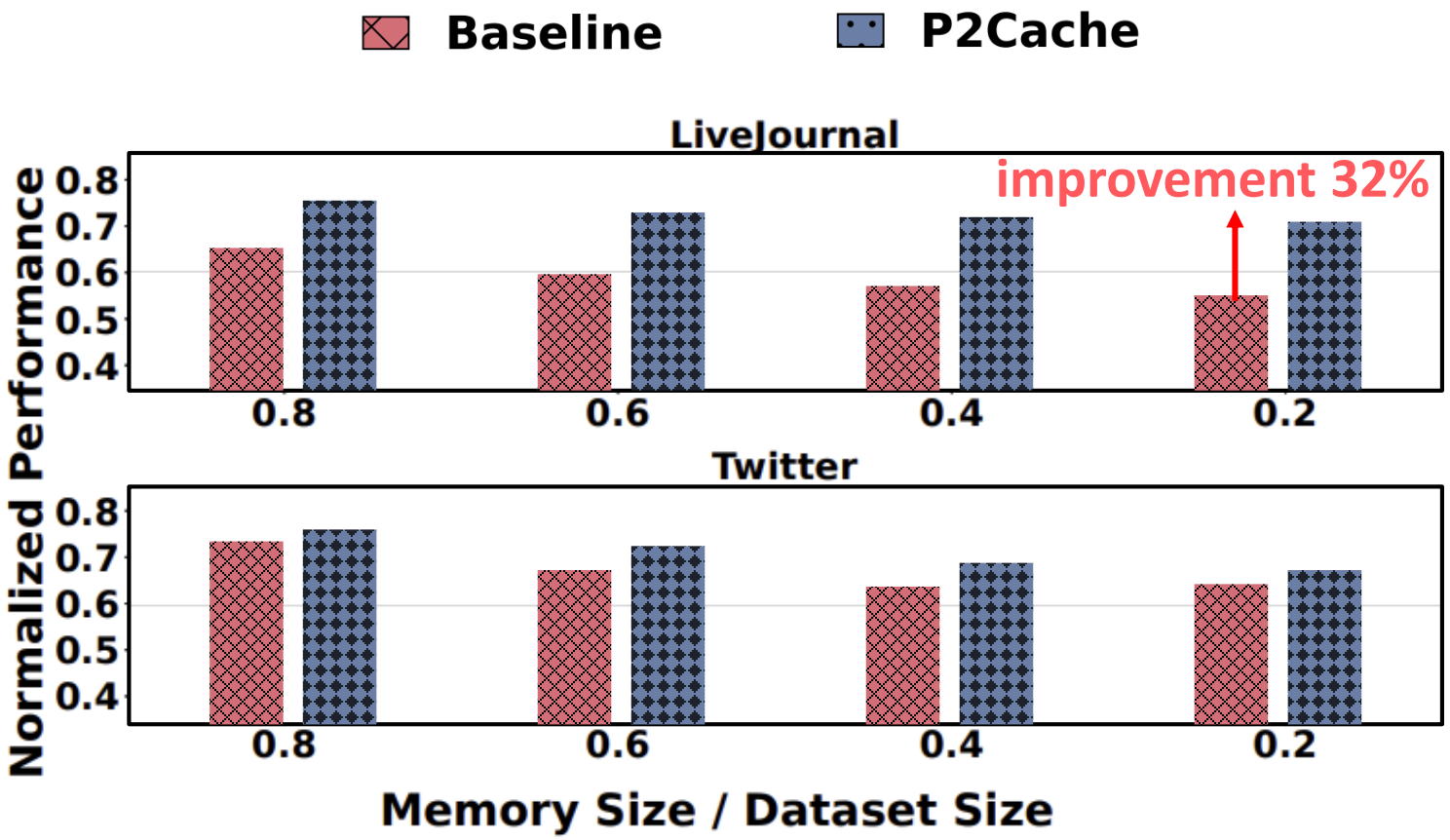
- Evaluation Platform
  - Linux Server (Kernel 5.8)
  - High-performance 2-TB NVMe SSD

- Workloads
  - Graph processing engine
    - ➢ Lumos, GraphWalker
  - Dataset
    - ➢ Live-Journal, Twitter-net

- Comparison schemes
  - **Baseline**: Default page cache
  - **P2Cache**: Customization with P2Cache

Result 1: Performance Over Kernel Page Cache

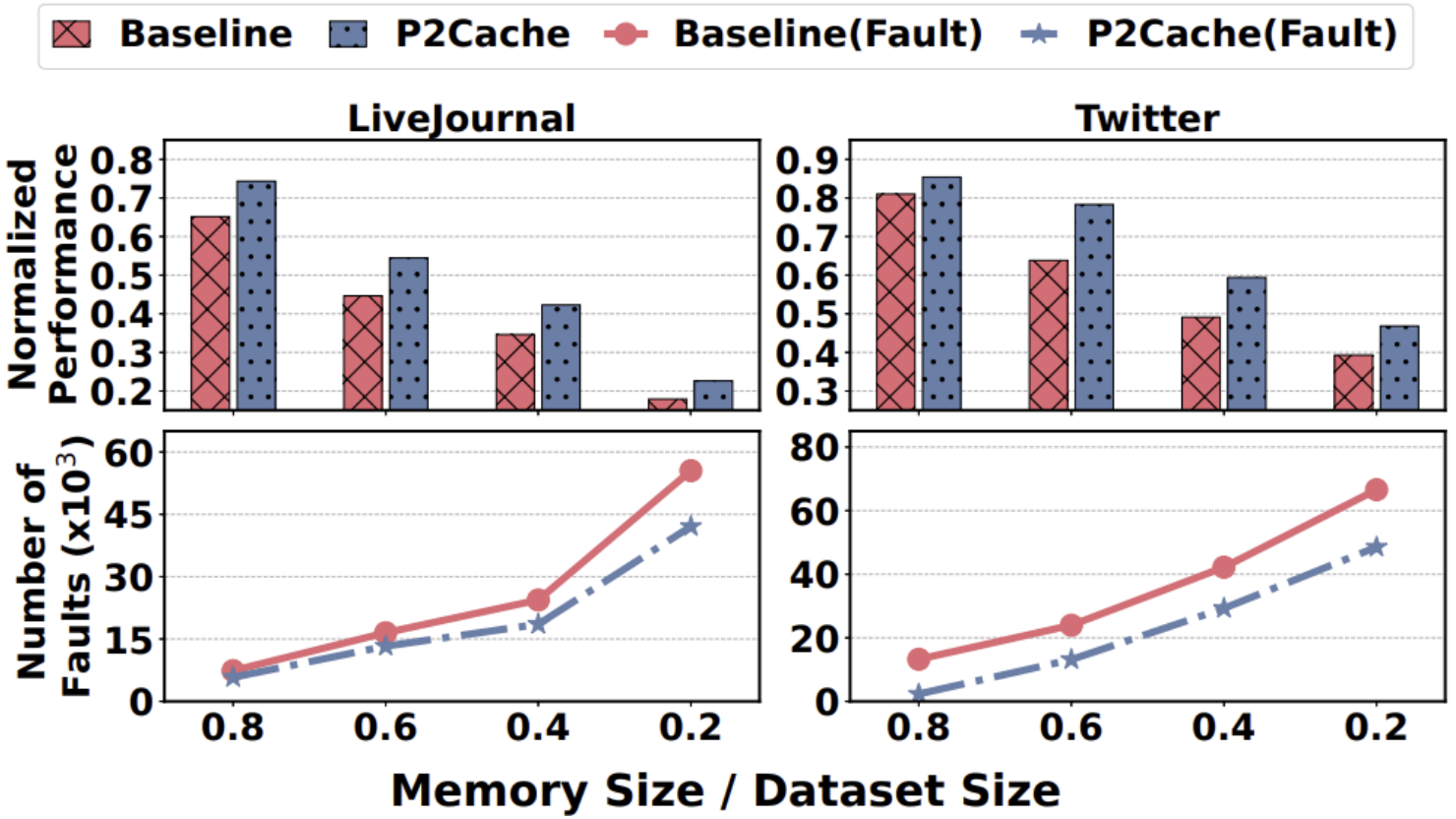
- Lumos performance comparisons



Observation: 32% improvement compared to Baseline

Result 2: Performance Over Application Cache

- GraphWalker performance comparisons



Observation: 14% improvement compared to Baseline

Conclusion

- Investigated a new design of page cache that improves the limitations of the existing OS/application page caches.
- Presented P2Cache which enables application developers to create custom kernel page cache.
- Demonstrated that the P2Cache is very promising in that it improves the limitations of the existing OS/application page caches.

Thank you!

P2Cache: An Application-Directed Page Cache for Improving Performance of Data-Intensive Applications

The 15th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage'23)