Eden: Developer-friendly Application-integrated Far Memory

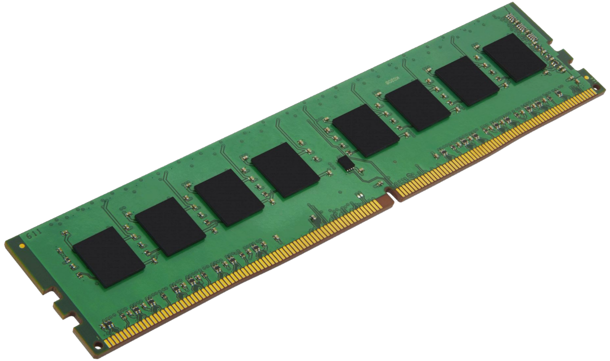
Anil Yelam, Stewart Grant, Saarth Deshpande, Nadav Amit,
Radhika Niranjan Mysore, Amy Ousterhout, Marcos K. Aguilera, Alex C. Snoeren

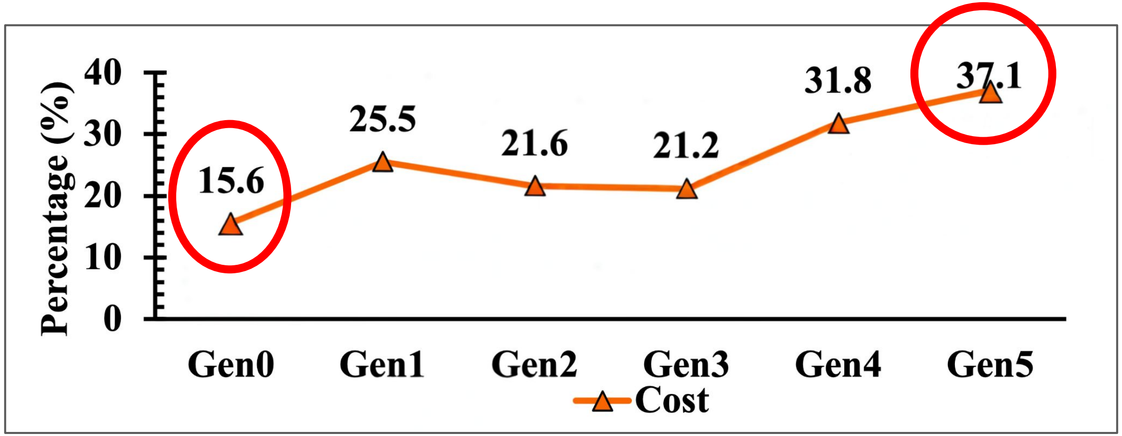
UC San Diego

vmware® by Broadcom

Rising DRAM cost in data centers

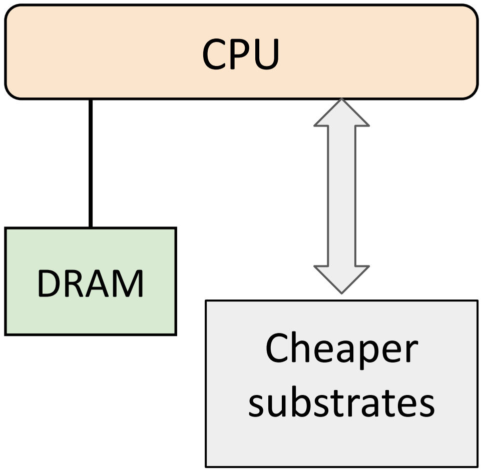
Reaching 40–50% of the server cost!



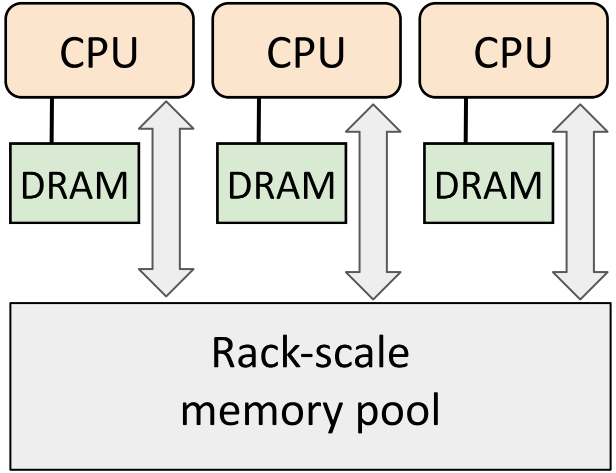


DRAM cost in Meta's data centers [Maruf et al. ASPLOS' 23]

Cost saving efforts



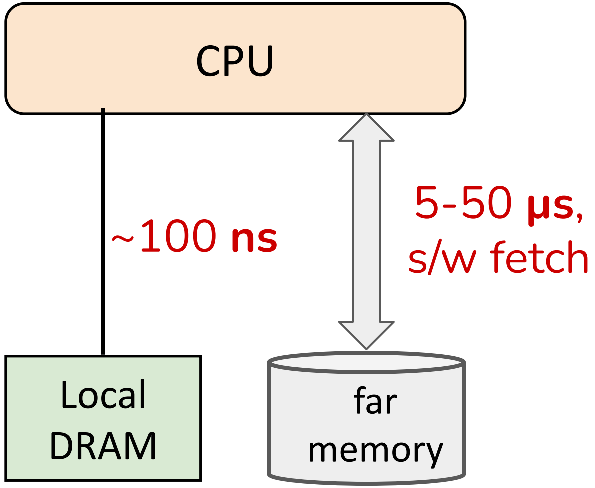
Google’s Software-defined Far Memory
Meta’s Transparent Memory Offloading



Fastswap, LegoOS, AIFM, Kona

Far memory

A cost-effective but slower memory extension.

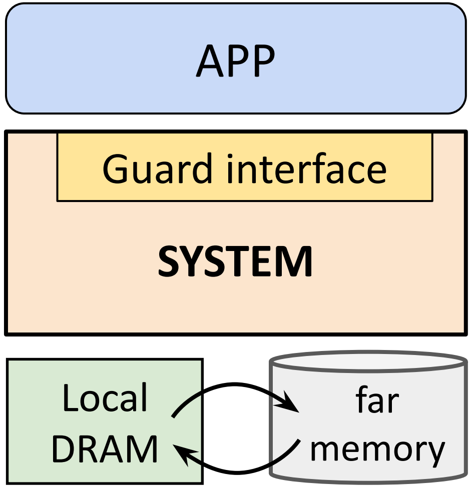


Compared to DRAM

- Slower
- Need guards

Far memory systems

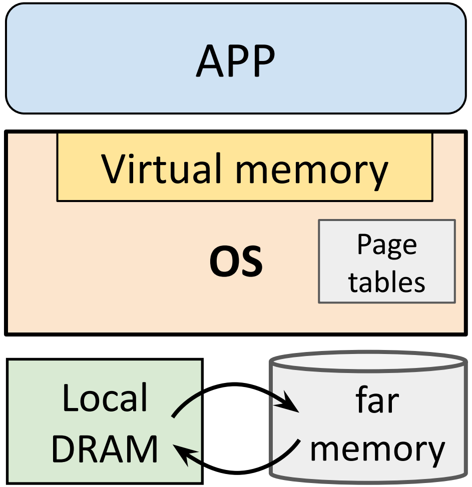
A cost-effective but slower memory extension.



Compared to DRAM	**Goal is to avoid:**
● Slower	⟶ Performance hit
● Need guards	⟶ Application changes

Hardware guards offer transparency

Use OS paging/hardware to support far memory.

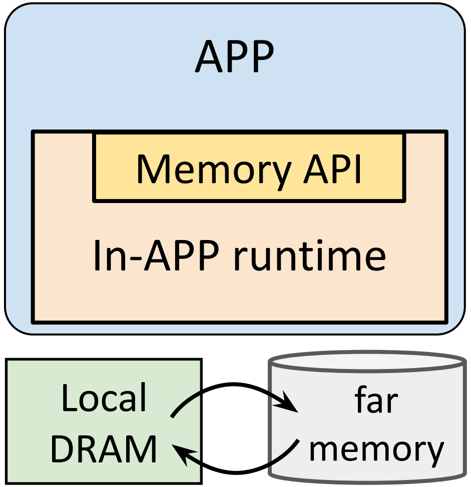


✅ No application changes

❌ Performance

Software guards enable better performance

Use a custom API and annotate every access.

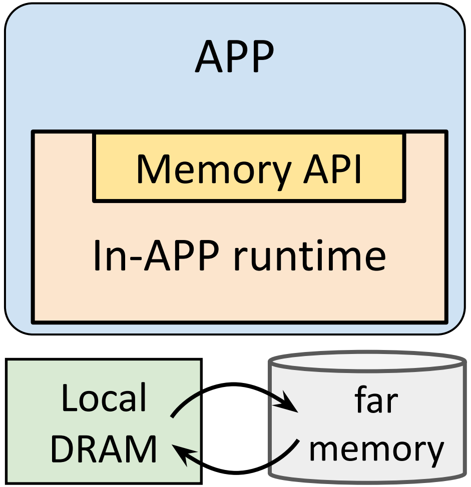


✅ Performance

❌ Significant porting effort

Software guards enable better performance

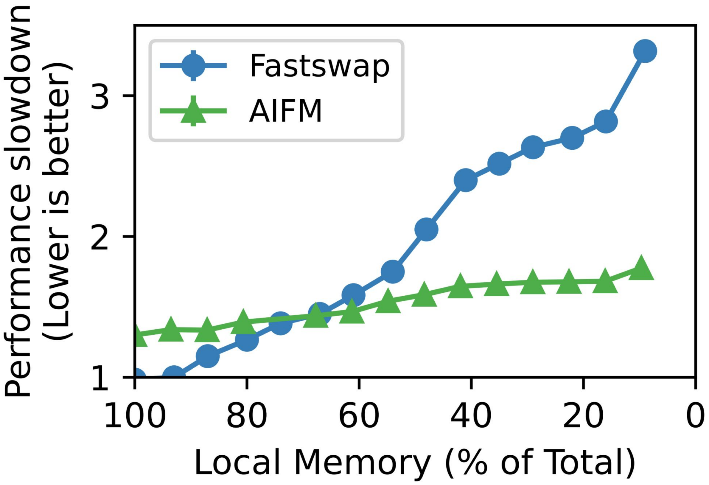
Use a custom API and annotate every access.



- ✅ Performance
- ❌ Significant porting effort
- ❌ Overhead for local accesses

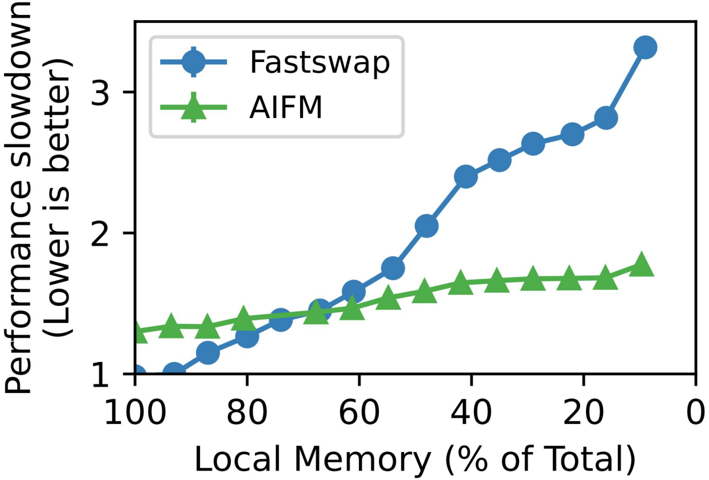
Example: Hardware vs Software guards

Performance



Example: Hardware vs Software guards

Performance



Code changes

Fastswap: 0

AIFM: >1000

Example: Hardware vs Software guards

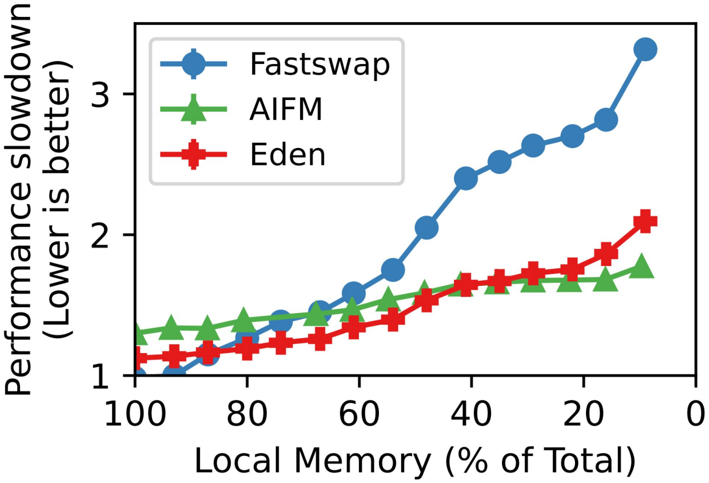
Performance

Code changes

Can we balance the performance benefits of software guards with the transparency of hardware guards?

Our answer is Eden

Performance



Code changes

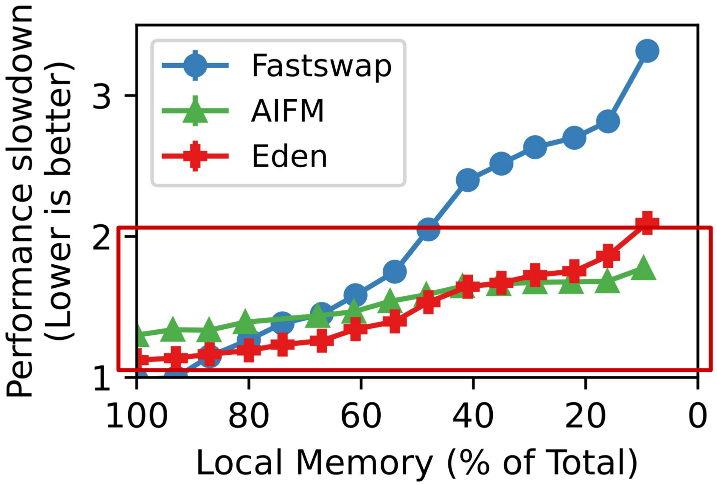
Fastswap: 0

AIFM: >1000

Eden: **10**

Our answer is Eden

Performance



Code changes

Fastswap: 0

AIFM: >1000

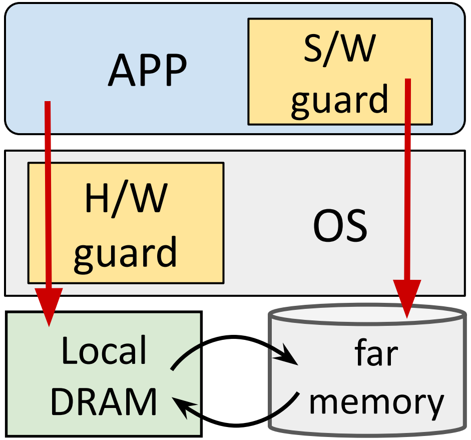
Eden: **10**

Eden supports both guards

Choice for each memory access.

- Hardware guards (default)
- Software guards (where beneficial)

Beneficial → Only far memory accesses
But that could be everywhere in the code?

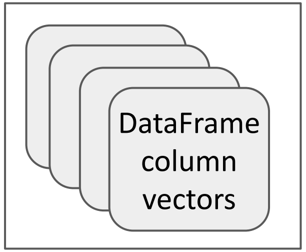


How do applications access far memory?

DataFrame example at 10% local memory.

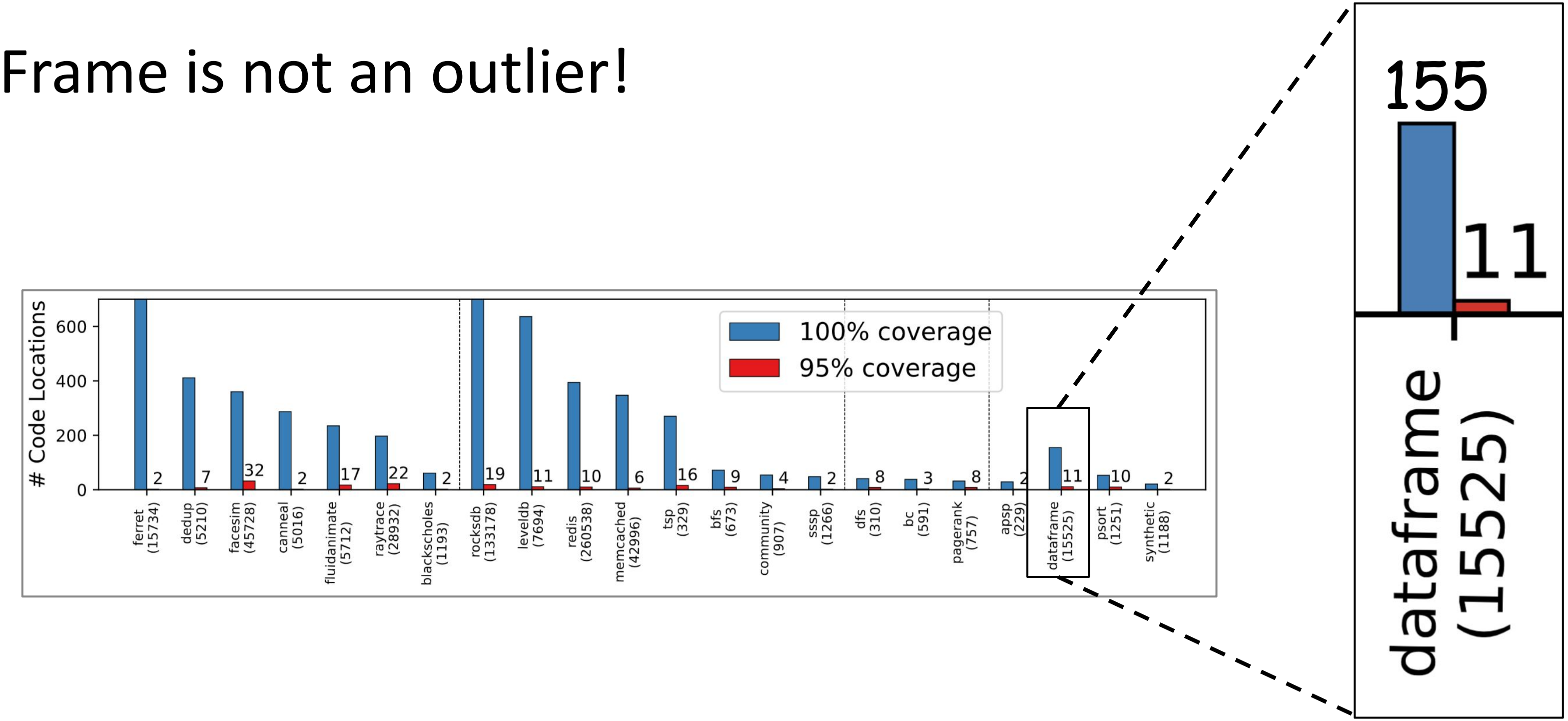
Of **15000** total lines of code:

- Only **155** → at least one far memory access
- Top **11** → 95% of all accesses!



DataFrame library

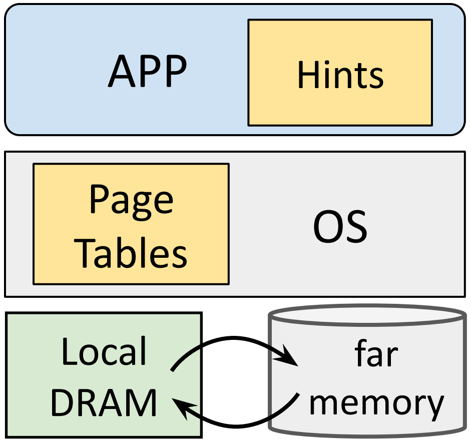
DataFrame is not an outlier!



Max 32 code locations—12 at median—see 95% far memory accesses

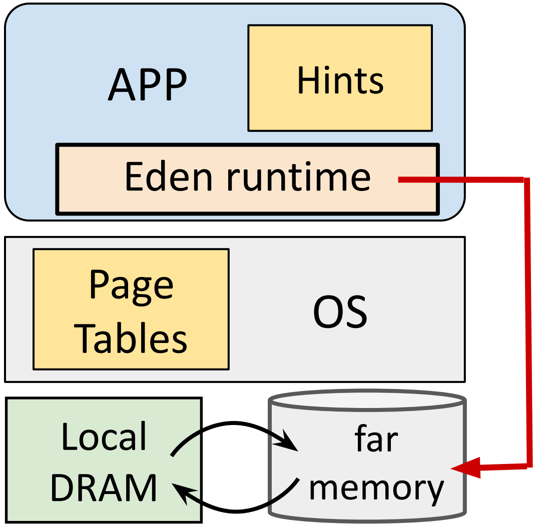
Eden overview

- **Software guards ⟶ Hints in code**
  Hardware guards ⟶ Default



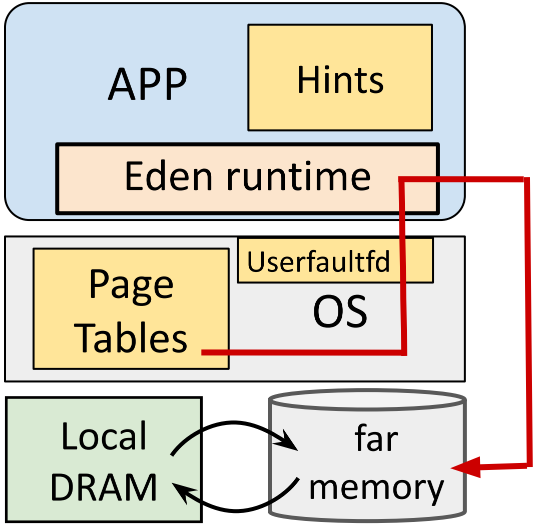
Eden overview

- Software guards → Hints in code
  Hardware guards → Default
- **User-level runtime**



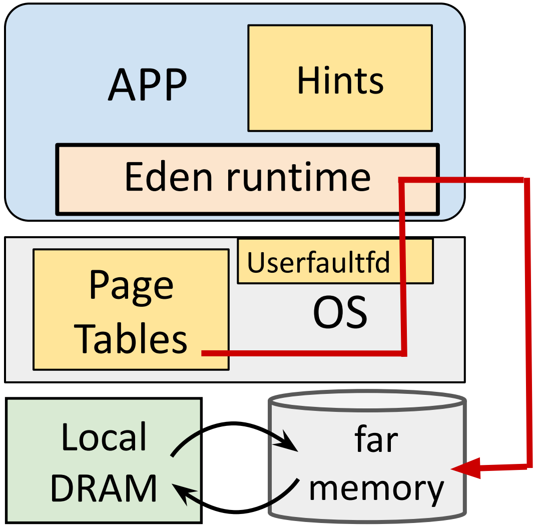
Eden overview

- Software guards → Hints in code
  Hardware guards → Default
- User-level runtime
- **Userfaultfd to keep hardware guards**
  - Slower but rare



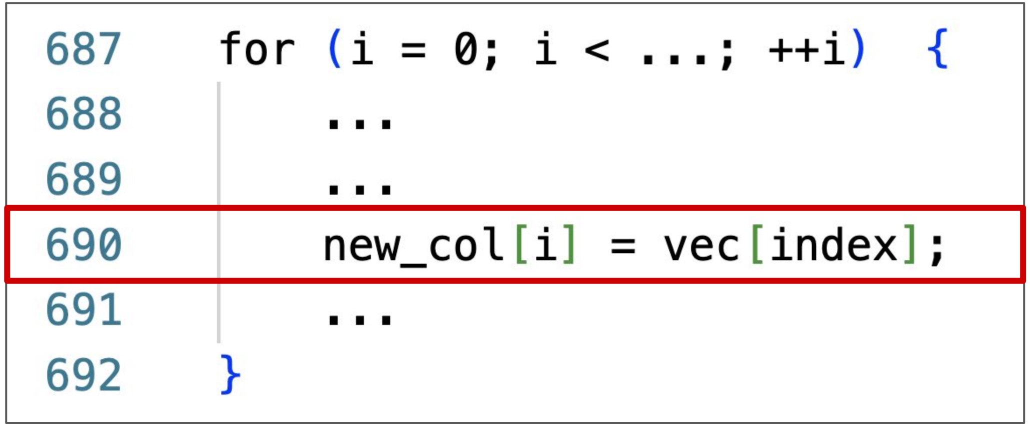
Eden overview

- Software guards → Hints in code
  Hardware guards → Default
- User-level runtime
- Userfaultfd to keep hardware guards
  - Slower but rare
- **Lightweight threads e.g., Shenango**

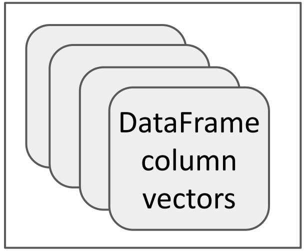


Step 1: Eden points out top locations

e.g., Top **two** locations for DataFrame on Line 690.



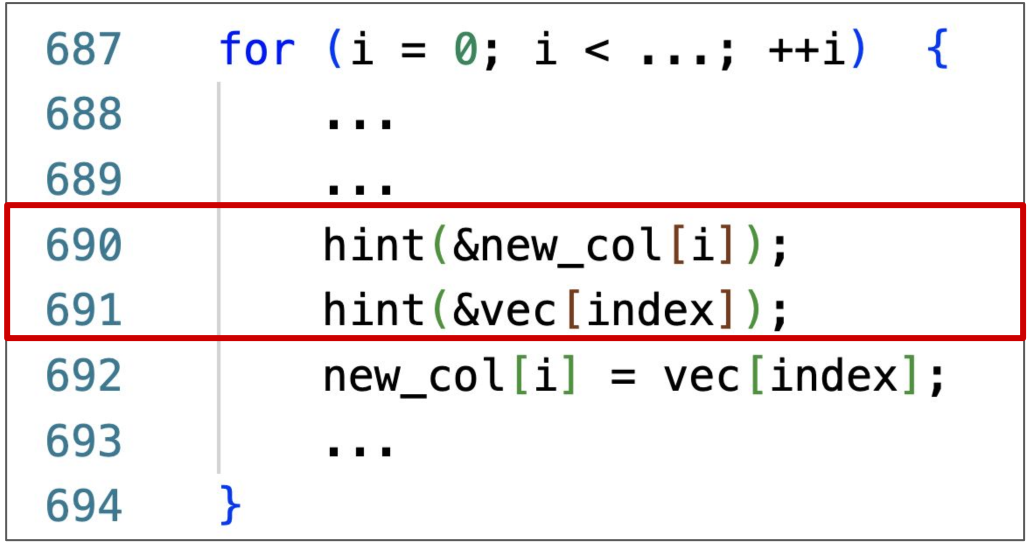
```
687    for (i = 0; i < ...; ++i)  {
688        ...
689        ...
690        new_col[i] = vec[index];
691        ...
692    }
```



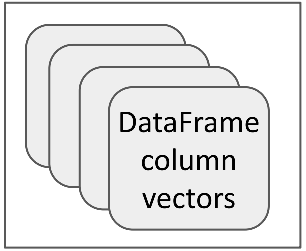
DataFrame library

Step 2: Add basic hints

Let Eden know what data to guard in software. Only for performance, not correctness!



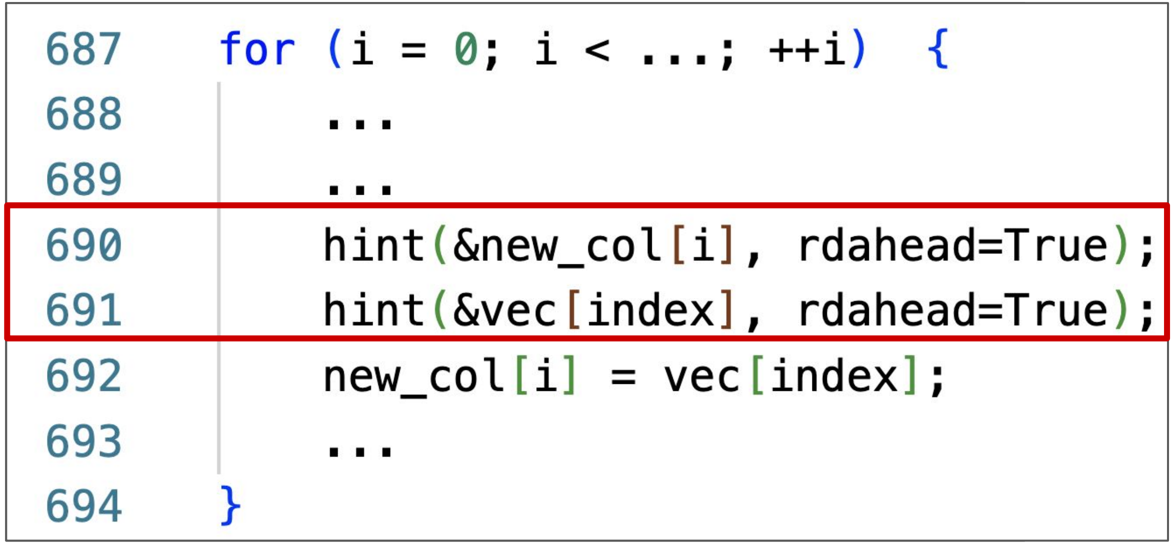
```
for (i = 0; i < ...; ++i)  {
    ...
    ...
    hint(&new_col[i]);
    hint(&vec[index]);
    new_col[i] = vec[index];
    ...
}
```



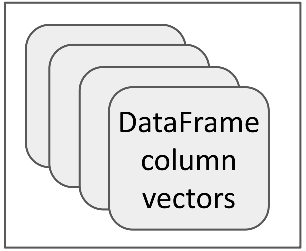
DataFrame library

Step 3: Pass additional info where helpful

DataFrame benefits from prefetching.



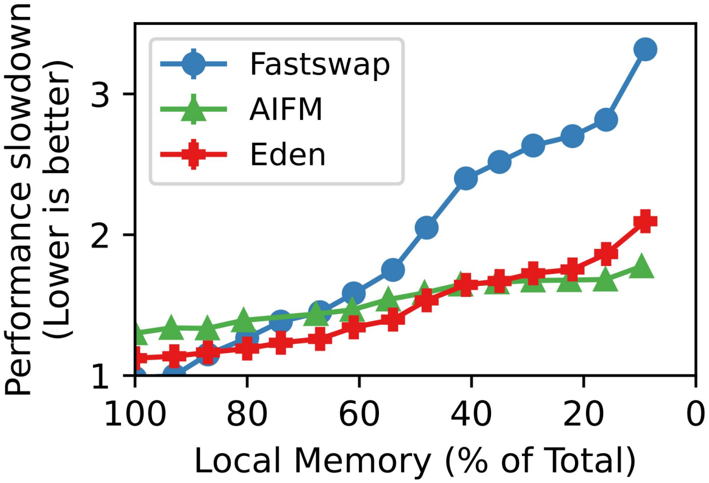
```
for (i = 0; i < ...; ++i)  {
    ...
    ...
    hint(&new_col[i], rdahead=True);
    hint(&vec[index], rdahead=True);
    new_col[i] = vec[index];
    ...
}
```



DataFrame library

Eden result for DataFrame

Performance



Code changes

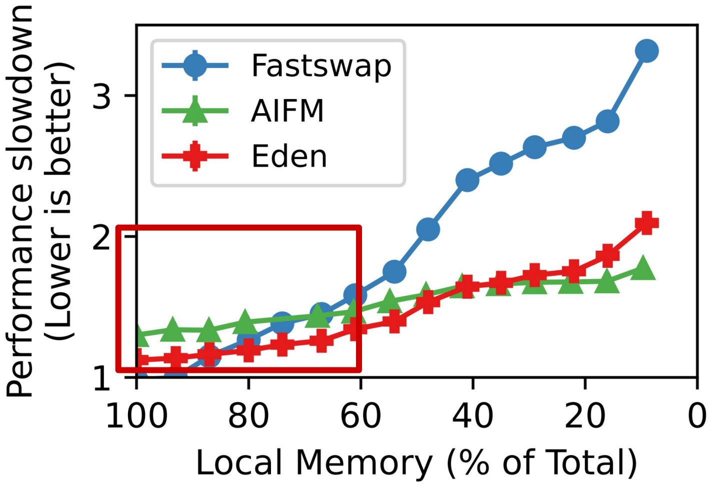
Fastswap: 0

AIFM: >1000

Eden: **10**

More local accesses → Exploit hardware guards

Performance

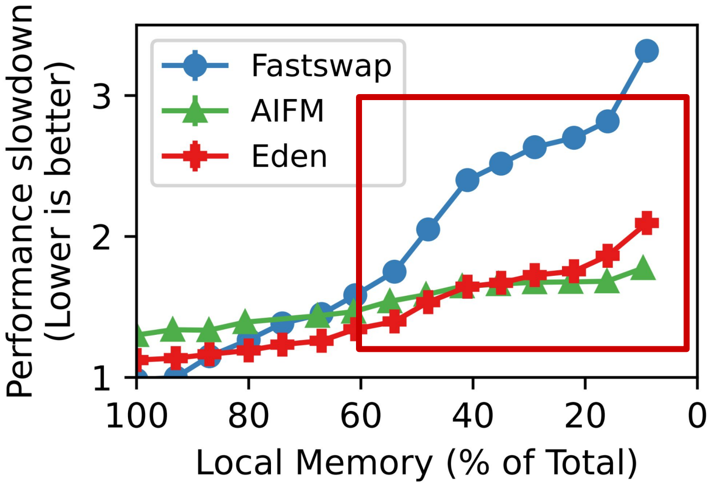


Code changes

Fastswap:	0
AIFM:	>1000
Eden:	**10**

More far accesses → Exploit software guards

Performance

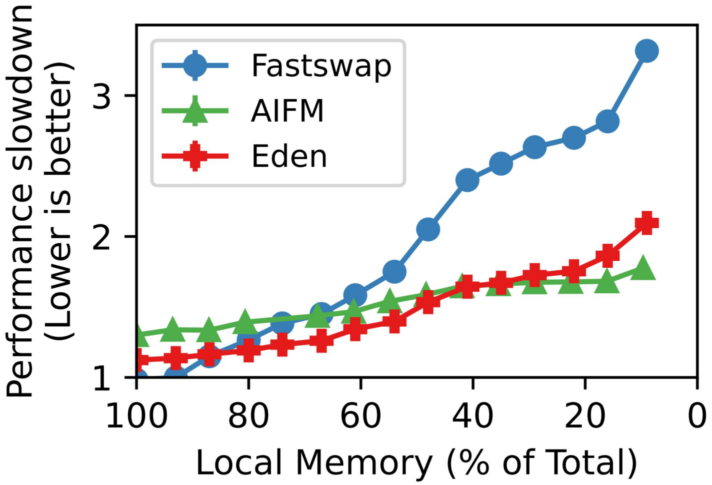


Code changes

Fastswap:	0
AIFM:	>1000
Eden:	**10**

Deliberate use of software guards

Performance

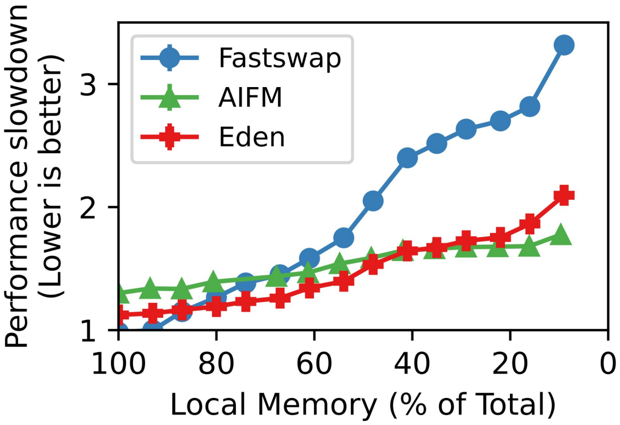


Code changes

Fastswap:	0
AIFM:	>1000
Eden:	**10**

Eden summary

- Applications only access far memory at **very few** code locations.
- Eden exploits this insight to **combine software and hardware guards**.
- Thus, Eden avoids hard bargain between performance and programmer effort.



AUTHORS: Anil Yelam, Stewart Grant, Saarth Deshpande, Nadav Amit, Radhika Niranjan Mysore, Amy Ousterhout, Marcos K. Aguilera, Alex C. Snoeren

UC San Diego

vmware by Broadcom