Eden: Developer-friendly User-space Far Memory

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

UCSDCSE Computer Science and Engineering

vmware

Far memory systems

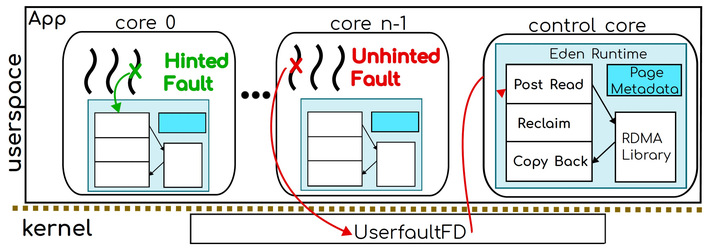
extend available memory on servers with (remote) memory from other servers.

Current approaches

- Kernel-based systems build on OS swapping to remote memory. They suffer from the *paging overheads and rigid policies* e.g., Fastswap [Eurosys'20]
- Runtime-based systems enable flexible, app-specific memory management policies but require *significant application changes* e.g., AIFM [OSDI'21]

Eden charts a middle-ground!

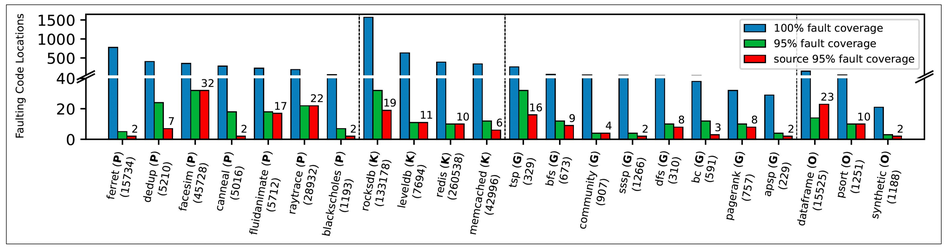
- Keeps the page-based virtual memory interface
- Manages pages from Userspace for flexibility
- Expects just a few "hints" for page faults



Eden's user-space runtime based on `userfaultfd` and Shenango [NSDI'19]

Key result 1: Only a handful of hints needed!

For most apps, a small number of locations cover 95% of faults, even after the library code is excluded. Further results show that the set of locations is robust across different degrees of memory pressure.



Hinting page faults

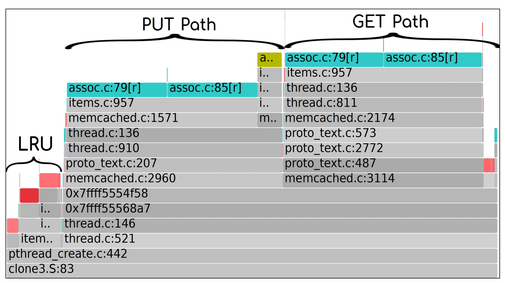
Eden exposes a simple API below for hinting impending page faults

- Eden *requires* at least basic hints
- Extended hints allow application-specific info like **reclaim priority** or **read-ahead**

```
/* basic hints */
hint_page_fault(address, mut = false);
hint_range_fault(address, size, mut = false);

/* extended hints */
hint_(page|range)_fault(
  address, /* faulting address or page */
  size,  /* size of the region (for a range fault) */
  mut = false, /* map read-only or writable */
  rdahead = 0, /* positive or negative read-ahead */
  ev_prio = 0, /* eviction priority */
  seq = false, /* sequential access */);
```

Step 1: Run the application binary with our tool `fltrace` to get faulting code locations

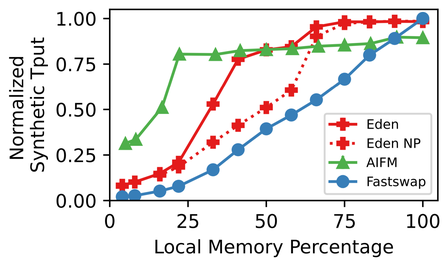


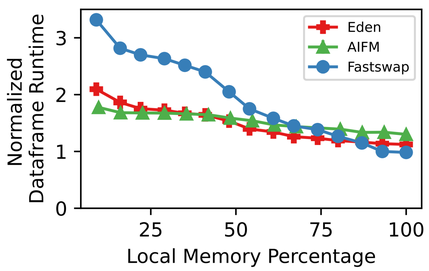
Step 2: Add hints at major faulting locations e.g., `assoc.c:85` above

```
82  item *ret = NULL;
83  int depth = 0;
84  while (it) {
85    hint_page_fault(&it->nkey);
86    if ((nkey == it->nkey) && (memcmp(key, ITEM_key(it), nkey) == 0)) {
87      ret = it;
88      break;
89    }
90    it = it->h_next;
```

Key result 2: Comparable performance to AIFM but with significantly (50x) less effort

For two benchmarks taken from AIFM, Eden performs significantly better than Fastswap and stays close to AIFM.





Synthetic Web Service Frontend
Eden required only **three** hints v. AIFM's data structure refactoring

Pandas-like C++ DataFrame
Eden needed **25** hints v. **1,195** LOC changes for running on AIFM