Operating Systems Lab
Implementation of LRU Page Replacement in xv6

September 16, 2025

Aim

To understand virtual memory management in `xv6` and implement the Least Recently Used (LRU) page replacement policy for handling page faults when the number of allocated frames exceeds the limit.

Step 1: Modifications in `proc.h`

We extend the `proc` structure to store per-process page replacement information.

New `frameinfo` structure

```
struct frameinfo {
  uint va;          // Virtual address of page
  pte_t *pte;       // Page table entry
  int ref;          // Reference (optional for CLOCK)
  uint last_used;   // Time of last access (for LRU)
};
```

Added fields in `struct proc`

```
struct frameinfo frames[16]; // Track resident pages (up to 16)
int framecount;              // Number of frames used
```

Step 2: Modifications in `vm.c`

We integrate LRU handling into the kernel's page allocation path.

Add global counter variable at top

```
extern uint ticks;   // global tick counter from trap.c
```

Add Helper Function: Update LRU Access

```
void update_lru_access(struct proc *p, uint va) {
  for(int i = 0; i < p->framecount; i++) {
    if(p->frames[i].va == va) {
      p->frames[i].last_used = ticks; // update on access
      break;
    }
  }
}
```

Replace existing `allocuvm()` from new function

```
int allocuvm(pde_t *pgdir, uint oldsz, uint newsz) {
  char *mem;
  uint a;

  if(newsz >= KERNBASE) return 0;
  if(newsz < oldsz) return oldsz;

  a = PGROUNDUP(oldsz);
  for(; a < newsz; a += PGSIZE) {
    mem = kalloc();
    if(mem == 0) {
      cprintf("allocuvm out of memory\n");
      deallocuvm(pgdir, newsz, oldsz);
      return 0;
    }
    memset(mem, 0, PGSIZE);
    if(mappages(pgdir, (char*)a, PGSIZE, V2P(mem), PTE_W|PTE_U) < 0) {
      cprintf("allocuvm out of memory (2)\n");
      deallocuvm(pgdir, newsz, oldsz);
      kfree(mem);
      return 0;
    }

    // ---------------- LRU Update ----------------
    struct proc *curproc = myproc();
    if(curproc) {
      if(curproc->framecount < 16) {
        curproc->frames[curproc->framecount].va = a;
        curproc->frames[curproc->framecount].pte = walkpgdir(pgdir, (
            void*)a, 0);
        curproc->frames[curproc->framecount].last_used = ticks;
        curproc->framecount++;
      } else {
        // Select victim using LRU
        int victim_index = 0;
        for(int i = 1; i < curproc->framecount; i++) {
          if(curproc->frames[i].last_used < curproc->frames[
              victim_index].last_used) {
            victim_index = i;
          }
        }
        // Free victim
        pte_t *vpte = curproc->frames[victim_index].pte;
        uint vpa = PTE_ADDR(*vpte);
```

```
        kfree(P2V(vpa));
        *vpte = 0;

        // Replace with new
        curproc->frames[victim_index].va = a;
        curproc->frames[victim_index].pte = walkpgdir(pgdir, (void*)a,
            0);
        curproc->frames[victim_index].last_used = ticks;
      }
    }
  }
  return newsz;
}
```

Step 3: User Program `mytest.c`

This program simulates LRU page replacement at user level.

```
#include "types.h"
#include "user.h"

#define MAX_PAGES 5
#define TOTAL_ACCESSES 15

int main(int argc, char *argv[]) {
  int lru[MAX_PAGES];
  int last_used[MAX_PAGES];
  int i, j, page, hit = 0, miss = 0, time = 0;

  for(i = 0; i < MAX_PAGES; i++) {
    lru[i] = -1;
    last_used[i] = -1;
  }

  printf(1, "Starting LRU page replacement simulation...\n");
  int accesses[TOTAL_ACCESSES] = {0,1,2,3,4,1,5,0,6,1,2,7,3,8,4};

  for(i = 0; i < TOTAL_ACCESSES; i++) {
    page = accesses[i];
    time++;
    int found = 0;

    for(j = 0; j < MAX_PAGES; j++) {
      if(lru[j] == page) {
        found = 1;
        last_used[j] = time;
        break;
      }
    }

    if(found) {
      hit++;
      printf(1, "Access page %d: HIT\n", page);
    } else {
      miss++;
      int replaced = -1;
      for(j = 0; j < MAX_PAGES; j++) {
```

```
        if(lru[j] == -1) {
          lru[j] = page;
          last_used[j] = time;
          replaced = j;
          break;
        }
      }
      if(replaced == -1) {
        int lru_index = 0, min_time = last_used[0];
        for(j = 1; j < MAX_PAGES; j++) {
          if(last_used[j] < min_time) {
            min_time = last_used[j];
            lru_index = j;
          }
        }
        printf(1, "Access page %d: MISS, replacing page %d\n", page,
           lru[lru_index]);
        lru[lru_index] = page;
        last_used[lru_index] = time;
      } else {
        printf(1, "Access page %d: MISS, placed in free frame\n", page)
           ;
      }
    }
  }

  printf(1, "LRU simulation completed.\n");
  printf(1, "Total hits: %d\n", hit);
  printf(1, "Total misses: %d\n", miss);
  exit();
}
```

Step 4: Makefile Update

Add `mytest.c` to the `UPROGS` list in `Makefile`:

```
UPROGS= ... _mytest\
```

Conclusion

We successfully implemented the Least Recently Used (LRU) page replacement policy in xv6, modified kernel structures, and verified its behavior with a simulation program.