Proxy 구현 - Part 3
캐싱 웹 오브젝트 동작 과정

정글 10기-34 박도현

캐싱의 목적은 같은 URI 재요청 시 원서버 없이 즉시 응답하는 것

1. 캐시된 URI 요청 → **캐시 HIT 발생**
ex) http://h:port/path

3. 프록시 선에서 처리하기 때문에
서버는 요청이 왔는지도 모름

2. proxy는 캐시 리스트에서 HIT를 확인하고 응답 데이터를 즉시 반환
ex) HTTP/1.0 200 OK\r\n...

→ 응답 속도가 훨씬 빨라진다

캐시 미스(MISS)가 발생한다면?

1. 캐시되지 않은 요청 → **캐시 MISS 발생**

2. server에게서 응답 데이터를 받아옴

4. client에게 응답 데이터 전송

3. 응답 데이터의 크기를 확인

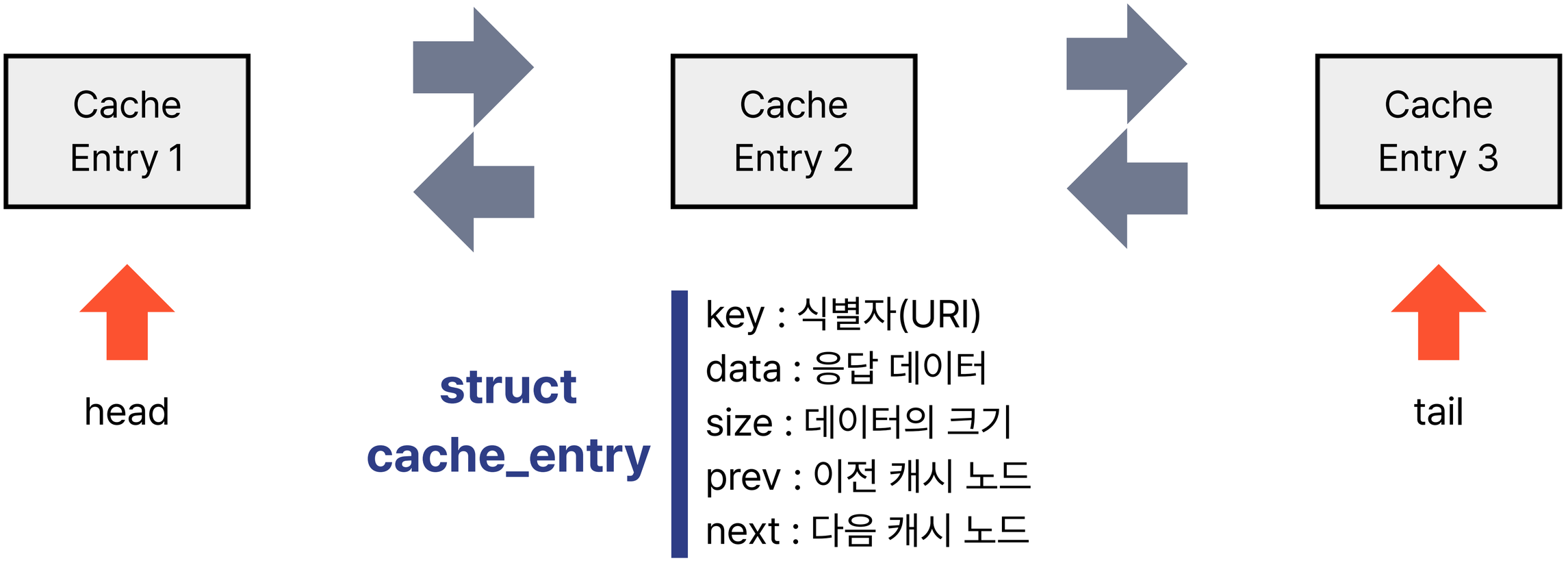
3.1. 만약 적당한 크기라면? → 캐시 리스트에 연결

3.2. 캐시 공간이 부족하다면? → 공간을 확보 후 연결

→ 이후 캐시된 요청이 오면 캐시를 통해 빠른 응답

각 캐시 엔트리를 "논리적"으로 모아두기

각 **캐시 엔트리**들은 하나의 데이터 블록으로, 메모리 힙 영역에 동적 할당되어 **독립적으로 존재**함
→ 그 캐시 엔트리들을 양방향 LinkedList로 연결시켜놓은 것 뿐



캐시와 캐시 리스트의 구조 - Code

```c
// 캐시 엔트리 구조체
typedef struct cache_entry {
    char *key;                  // 식별자(URI)
    char *data;                 // 응답 원본 데이터(헤더 포함)
    size_t size;                // 데이터 바이트 수
    struct cache_entry *prev; // LRU 리스트 이전 노드
    struct cache_entry *next; // LRU 리스트 다음 노드
} cache_entry_t;

// 전역 캐시 상태
static cache_entry_t *head = NULL; // LRU의 처음 : 가장 최근 사용 캐시
static cache_entry_t *tail = NULL; // LRU의 마지막 : 가장 오래된 캐시
static size_t current_size = 0;    // 현재 저장된 캐시 크기의 총 합
```

캐시 관리 정책 : LRU(Least Recently Used)

가장 오랫동안 사용하지 않은 캐시부터 제거한다

```c
// LRU 리스트에서 가장 오래된 항목 제거(필요 시 반복)
static void remove_tail(size_t need) {
    // 1. 현재 크기 + 필요 크기 > 최대 크기 이고,
    // 2. 제거할 대상이 있을 때(tail 존재) 반복
    while ((current_size + need > MAX_CACHE_SIZE) && tail) {
        // 가장 마지막 캐시 데이터부터 제거
        cache_entry_t *entry = tail;
        list_remove(entry);
        current_size -= entry->size; // 현재 크기에서 제거될 엔트리 크기만큼 차감
        free(entry->key);            // key 해제
        free(entry->data);           // 데이터 해제
        free(entry);                 // 엔트리 구조체 자체를 해제
    }
}
```

캐시를 찾아서 반환할 때 - 최신 사용 갱신

```c
// LRU 리스트를 head부터 순회하며 key를 통해 캐시 엔트리를 찾는다
static cache_entry_t *find_cache(const char *key) {
    for (cache_entry_t *entry = head; entry; entry = entry->next) {
        if (strcmp(entry->key, key) == 0)
            return entry;
    }
    return NULL;
}
```

```c
// 방금 쓴 캐시를 찾아서 "최신 사용"으로 갱신하기
cache_entry_t *used_entry = find_cache(key);
if (used_entry) {
    list_remove(used_entry); // 잠깐 지우고
    insert_head(used_entry); // 다시 앞에 넣는다
}
```

정리

(기존) 모든 클라이언트의 요청을 서버까지 전달하여 받아오는 방식

1. Proxy Server의 캐시 기능을 추가

Cache HIT시 Proxy 선에서 즉시 반환

2. LRU 정책

시간 지역성을 적극 활용

응답 속도와 처리량 향상 → 더욱 효율적인 구조가 됨