Programming Assignment#4

Due : 6th DEC. (Thur),5:59 PM

1. Introduction

이번 과제를 통해 Thread 를 이용하여 병렬적으로 work 를 수행하는 KV 데이터베이스를 구현한다

2. Problem specification

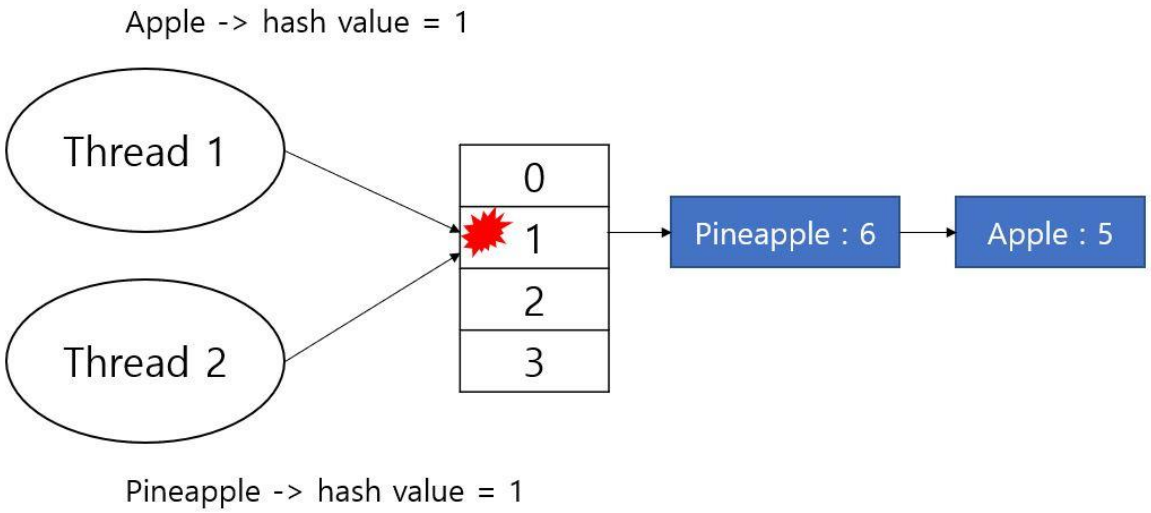
Key-Value(KV) 데이터베이스는 간단한 Key-Value 함수를 사용하여 데이터를 저장하는 비관계형 데이터베이스이다. 이전 과제에서는 하나의 프로세스를 이용하여 Hash table 및 파일 관리를 모두 수행하였다. 이 때문에 모든 과정이 Serialization 됨으로써 여러가지 일을 동시에 처리할 수 없는 비효율적인 디자인을 갖게 되었다.

이번 과제에서는 thread 를 이용하여 동시다발적으로 명령을 병렬적으로 수행할 수 있도록 KV 데이터베이스를 수정한다.

서로 다른 `input` 을 동시에 수행할 `thread` 를 생성해야 하므로, 필요에 따라 `db.c`, `db.h` 뿐만 아니라 **`main.c` 도 수정할 수 있다.**

2-1. Threads

- 이번 과제에서는 새롭게 입력해야 할 argument 가 추가된다. 데이터베이스의 Hash table 크기에 더해, 사용할 thread 의 개수를 입력한다.
  Ex) ./wordcount 1024 4
  위 예시와 같이 입력하면, size 가 1024 인 해시 테이블을 4 개의 worker thread 가 사용한다.
- thread 마다 데이터를 GET, PUT 할 시에 thread 간 synchronization 에 유의해야 한다. 그림과 같이 여러 thread 가 같은 data 에 접근할 시, race condition 이 발생하여 결과적으로 잘못된 결과를 return 할 수 있다. 따라서 기존의 구현하였던 db_put, db_get 함수를 thread-safe 하게 보완하여야 한다.
- Thread 를 사용할 경우 중간과정에서 출력될 명령어의 순서가 달라질 수 있다. 따라서 본 과제에서는 최종적으로 데이터베이스에 저장된 모든 KEY 와 VALUE 쌍의 제대로 저장되어 있으면 정답으로 간주한다.



- hash table 의 data 를 File 로 내리는 시점은 hash table 에 있는 KV 쌍의 수가 size 와 같을 때로, 이전 과제와 동일하다.
- 단, 기존 방식과 다르게 최종적으로 생성되는 파일의 개수는 제한되지 않는다.

2-2. Data compaction

- 이번 과제에서는 데이터들의 compaction 을 수행하도록 한다.
- Data compaction 은 파일들이 가지고 있는 invalid entry 를 삭제하고 흩어져 있는 valid entry 들을 모으는 작업을 의미한다. 이를 통해 불필요한 저장공간 소모를 방지하고 데이터 검색 시간을 단축시킬 수 있다.
- compaction 을 진행하는 방식 (파일 형식, 개수, 파일 이름, 기준 등등)은 자유롭게 구현 가능하나, 보고서에 그 과정을 기록하여야 한다.

  Ex)

File 6	Banana : 9	Golem : 1	Chicken : 4	Yogurt : 9
File 5	Apple : 5	Banana : 8	Caramel : 17	Deer : 7
File 4	Chicken : 3	Banana : 7	Yogurt : 8	Knee : 10
File 3	Eagle : 10	Banana : 6	Horse : 4	Deer : 6
File 2	Flower : 2	Apple : 4	Chicken : 3	Eagle : 9
File 1	Deer : 5	Eagle : 9	Caramel : 16	Apple : 3

File 7	Flower : 2	Horse : 4	Eagle : 10	Knee : 10
File 6	Banana : 9	Golem : 1	Chicken : 4	Yogurt : 9
File 5	Apple : 5		Caramel : 17	Deer : 7

2-3. Iterative execution

- **어플리케이션을 종료한 후 다시 실행할 경우 이전에 실행한 결과로 파일이 남아 있다면 해당 파일을 참고해 work 를 수행할 수 있어야 한다.**

Ex)

```
$ ./wordcount 100 4
DB opened
A
GET [A] [NULL]
PUT [A] [1]
DB Closed
$ ./wordcount 1000 2
DB opened
A
GET [A] [1]
PUT [A] [2]
```

2-4. Restriction

- 직접적으로 KV data 의 정보를 얻기 위해 처음 생성하는 hash table 이외 다른 구조체를 임의로 사용하는 것은 금지된다. 즉, hash table 외 임의의 구조체를 사용할 시 해당 구조체에서 KV data 의 정보를 얻을 수 없다.
- 이전 과제와 마찬가지로, Performance 에 따라 추가점수를 획득할 수 있다.
- DB 에서 생성하는 파일의 용량에 따라 추가점수를 획득할 수 있다.
- Thread 의 생성 및 관리는 Pthread API 를 이용한다. 다른 thread 관련 API 및 Process 를 이용한 구현은 채점대상에서 제외된다.
- Synchronization 을 수행할 지점은 자유롭게 선정하되, thread-safe 함을 유지해야 한다. 또한, 보고서에도 구현한 디자인이 thread-safe 함을 설명해야 한다.
- 보고서에는 각 Thread 의 작동 원리를 명확히 기록해야 한다.
- 파일의 입출력은 open(), read(), write(), close(), lseek() 등의 system call 을 사용하도록 한다.

3. Hand in instructions

✓ 작성한 프로그램 코드 상단에 이름과 학번을 적는다.

✓ 과제는 제출 시 "학번.tar.gz"로 압축한다.

➢ 압축 파일은 Makefile, main.c, db.c, db.h, README.pdf 로 이루어져 있어야 하며,

압축파일의 이름과 확장자는 학번.tar.gz 여야 한다.

- ✓ 프로그램 코드와 별도로, 구현에 대한 내용을 담은 보고서를 함께 제출한다. 보고서의 파일 포멧은 pdf 로 제한하며, 형식에는 제한이 없다. 제목은 README.pdf 로 한다.
- ✓ 과제는 icampus 의 과제란에 제출한다.
- ✓ 과제 제출 시간은 icampus 제출 시간을 기준으로 하며, 기한 이후엔 10%씩 감점되고, 과제 제출은 추가 제출기간이 마감되기 전까지 제출 가능합니다.
- ✓ 본 과제는 혼자서 한다.
- ✓ GNU `make` 도구는 큰 프로그램을 만들 때 유용하게 사용되는데, 프로그램을 제작하기 위해 어떤 코드를 (재)컴파일해야 하는지 결정해준다. 일단 `Makefile` 이 준비되면, 어떤 소스 코드를 변경하던지 셸에서 단순히 `make` 란 명령을 실행시키는 것으로 재 컴파일이 필요한 모든 파일을 알아서 찾아 다시 컴파일한다.
- ✓ **Copy 할 경우, 연구실 자체 규정에 따라 처벌하며, 상당한 불이익이 있을 수 있다.**

Have fun!

컴퓨터시스템연구실