시스템소프트웨어(CB2400108-059) - HW4:Cachelab

정보컴퓨터공학부 202255513 김대욱

시스템소프트웨어 과목의 마지막 과제인 'Cache LAB'은 두 가지 항목으로 나뉜다. part A, part B로 구성되어 있는데, part A에서는 Cache Simulator를 C언어를 이용하여 cache memory가 실제로 어떤 방식으로 구현되는 지를 프로그래밍하고, part B에서는 함수를 이용하여 전치행렬을 구현하는 것이다.

앞서 진행한 Bomb Lab이나 Attack Lab처럼 개인별로 phase의 구성이 다른 것은 아니여서, Data Lab과 같이 학과 서버(CSEDELL)에 'cachelab-handout.tar' 파일을 업로드 하고, **tar xvf cachelab-handout.tar** 명령어를 입력하여 압축 파일을 풀 수 있다. 학과 서버에서 압축을 해제하지 않고, window 시스템에서 압축을 해제할 경우 문제가 발생할 수 있으므로 이 점을 유의하여 진행하였다.

[part A]

먼저, Cache를 구성하고 있는 구조에 대해 살펴보면 아래와 같다.

```
10  typedef struct
11  {
12      unsigned tag;
13      unsigned usedtime;
14  } block;
15
16  block *cache;
```

Cache는 struct 구조로 구현하였는데, typedef를 활용하여 해당 struct를 **block** 이라는 이름으로 사용하게 된다. 16줄의 코드를 보면, struct block을 활용하여 cache를 만드는 것을 확인할 수 있다.

'usedtime' 변수는 LRU 캐시의 cache line을 결정하는 변수로 사용된다. 초깃값으로 0을 두게 되면, 아직 사용하지 않았다는 의미이므로 유효하지 않은 값을 뜻하게 된다. 만약, 0이 아닌 다른 값을 가진다면 usedtime의 값이 작을 수록 캐시를 적게 사용했다는 것으로 해석해야 한다.

그리고, 아래의 getOpt 함수를 활용하여, command line의 인자 값을 받아올 수 있다. C 언어를 사용하기 때문에 해당 함수의 인자를 살펴보면 int argc와 char **argv를 통해 코드를 구성한 것을 알 수 있다.

```
int getOpt(int argc, char **argv, int *s, int *E, int *b, int *verbose, char *tracefile)
{
    int oc;
    while((oc=getopt(argc, argv, "hvs:E:b:t:")) != -1) {
        switch(oc) {
            case 'h': printHelpMenu(); break;
            case 'v': *verbose=1; break;
            case 's': *s = atoi(optarg); break;
            case 'E': *E = atoi(optarg); break;
            case 'b': *b = atoi(optarg); break;
            case 't': strcpy(tracefile,optarg); break;
            default : printf("input error\n"); break;
        }
    }
    return 0;
}
```

그리고 아래의 main 함수 내의 코드를 통해 Cache 값을 초기화한다.

```
42      cache = (block *)malloc(sizeof(block)* E<<s);
43      memset(cache, 0, sizeof(block)* E<<s);
```

main 함수 내의 명령어를 통해 file의 argument와 주소 값을 가져온 후 데이터 접근 과정을 거쳐야 한다.

```
45      fp = fopen (tracefile,"r");
46      while(fscanf(fp,"%s%x,%d\n", op, &addr, &size) > 0) {
47          if(verbose)
48              printf("%s %x,%d ",op,addr,size);
49          switch(op[0]) {
50              case 'M': hit++;
51              case 'L':
52              case 'S': find(op[0],addr,size,++t);
53          }
54      }
```

아래는 find 함수 내의 일부분을 가져온 것인데, tag와 set_index를 만족하는 set을 찾고 65~66줄 명령어를 통해 data를 검색하게 된다. (eviction_block은 쿼리 프로세스 동안의 LRU의 cache line이다.)

```
63      unsigned tag = addr >>b >>s ;
64      unsigned set_index = addr >> b &((1<<s) -1);
65      block *cache_set = cache + E * set_index ;
66      block *eviction_block = cache_set;
```

0이 아닌 가장 작은 값을 가지고 있는 cache line은 set에서 가로지르게 된다. 따라서 usedtime != 0이고, tag가 matching 되면 **hit**라고 할 수 있다. 만약, usedtime == 0이면 block이 비어있게 되는데, 이 상태를 **miss**라고 할 수 있다.

마지막으로, usedtime != 0이고 tag가 matching 되지 않는다면, eviction_block.usedtime과 비교하여 시간이 짧으면 update를 진행하게 된다. 함수에서 Loop가 끝나게 되면, set의 모든 캐시라인이 찼다는 것을 의미하고, LRU cache line이 교체된다. **make** 명령어를 입력한 후, **./test-csim** 명령어를 입력하게 되면, 자동 채점 프로그램에 의해 아래와 같은 결과가 나오게 되는 것을 확인할 수 있다.

```
202255513@CSEDell:~/HW4/cachelab-handout$ ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     3 (1,1,1)       9       8       6       9       8       6  traces/yi2.trace
     3 (4,2,4)       4       5       2       4       5       2  traces/yi.trace
     3 (2,1,4)       2       3       1       2       3       1  traces/dave.trace
     3 (2,1,3)     167      71      67     167      71      67  traces/trans.trace
     3 (2,2,3)     201      37      29     201      37      29  traces/trans.trace
     3 (2,4,3)     212      26      10     212      26      10  traces/trans.trace
     3 (5,1,5)     231       7       0     231       7       0  traces/trans.trace
     6 (5,1,5)  265189   21775   21743  265189   21775   21743  traces/long.trace
    27

TEST_CSIM_RESULTS=27
```

[part B]

파트 B는 캐시 미스의 숫자를 최대한 줄이는 것이 목표이고, 전치 행렬 함수를 최적화 해야한다. 주어진 상황에서는 32 X 32, 64 X 64, 67 X 61 세 가지의 행렬 구조가 있다. 각 경우에 따라서 풀이 방법을 다르게 해서 문제를 해결하는 것이 도움이 된다.

* 32 X 32 Matrix Code

```
if (N == 32 && M==32) {
    for (BlockCol = 0; BlockCol < M; BlockCol += BlockSize2) {
        for (BlockRow = 0; BlockRow < N; BlockRow += BlockSize2) {
            for (Row_Idx = BlockRow; Row_Idx < BlockRow + BlockSize2; Row_Idx++) {
                for (Col_Idx = BlockCol; Col_Idx < BlockCol + BlockSize2; Col_Idx++) {
                    if (Row_Idx != Col_Idx)
                        B[Col_Idx][Row_Idx] = A[Row_Idx][Col_Idx];
                    else {
                        tmp0 = A[Row_Idx][Col_Idx];
                        tmp1 = Row_Idx;
                    }
                }
                if (BlockRow == BlockCol)
                    B[tmp1][tmp1] = tmp0;

            }
        }
    }
}
```

* 64 X 64 Matrix Code

```
else if (N == 64 && M ==64) {
    for (BlockCol = 0; BlockCol < M; BlockCol += BlockSize2) {
        for (BlockRow = 0; BlockRow < N; BlockRow += BlockSize2) {
            for (Row_Idx = BlockRow; Row_Idx < BlockRow + BlockSize1; Row_Idx++) {

                tmp0 = A[Row_Idx][BlockCol];
                tmp1 = A[Row_Idx][BlockCol + 1];
                tmp2 = A[Row_Idx][BlockCol + 2];
                tmp3 = A[Row_Idx][BlockCol + 3];
                tmp4 = A[Row_Idx][BlockCol + 4];
                tmp5 = A[Row_Idx][BlockCol + 5];
                tmp6 = A[Row_Idx][BlockCol + 6];
                tmp7 = A[Row_Idx][BlockCol + 7];

                B[BlockCol][Row_Idx] = tmp0;
                B[BlockCol + 1][Row_Idx] = tmp1;
                B[BlockCol + 2][Row_Idx] = tmp2;
                B[BlockCol + 3][Row_Idx] = tmp3;

                B[BlockCol][Row_Idx + 4] = tmp4;
                B[BlockCol + 1][Row_Idx + 4] = tmp5;
                B[BlockCol + 2][Row_Idx + 4] = tmp6;
                B[BlockCol + 3][Row_Idx + 4] = tmp7;
            }
```

```
            for (Col_Idx = BlockCol; Col_Idx < BlockCol + BlockSize1 ; Col_Idx++) {

                tmp4 = A[BlockRow + 4][Col_Idx];
                tmp5 = A[BlockRow + 5][Col_Idx];
                tmp6 = A[BlockRow + 6][Col_Idx];
                tmp7 = A[BlockRow + 7][Col_Idx];

                tmp0 = B[Col_Idx][BlockRow + 4];
                tmp1 = B[Col_Idx][BlockRow + 5];
                tmp2 = B[Col_Idx][BlockRow + 6];
                tmp3 = B[Col_Idx][BlockRow + 7];

                B[Col_Idx][BlockRow + 4] = tmp4;
                B[Col_Idx][BlockRow + 5] = tmp5;
                B[Col_Idx][BlockRow + 6] = tmp6;
                B[Col_Idx][BlockRow + 7] = tmp7;

                B[Col_Idx + 4][BlockRow] = tmp0;
                B[Col_Idx + 4][BlockRow + 1] = tmp1;
                B[Col_Idx + 4][BlockRow + 2] = tmp2;
                B[Col_Idx + 4][BlockRow + 3] = tmp3;

                B[Col_Idx + 4][BlockRow + 4] = A[BlockRow + 4][Col_Idx + 4];
                B[Col_Idx + 4][BlockRow + 5] = A[BlockRow + 5][Col_Idx + 4];
                B[Col_Idx + 4][BlockRow + 6] = A[BlockRow + 6][Col_Idx + 4];
                B[Col_Idx + 4][BlockRow + 7] = A[BlockRow + 7][Col_Idx + 4];
            }
        }
    }
}
```

* 67 X 61 Matrix Code

```
else if(N ==67 && M ==61) {
    for (BlockCol = 0; BlockCol < M; BlockCol += BlockSize3) {
        for (BlockRow = 0; BlockRow < N; BlockRow += BlockSize3) {
            for (Row_Idx = BlockRow; (Row_Idx < N) && (Row_Idx < BlockRow + BlockSize3
                for (Col_Idx = BlockCol; (Col_Idx < M) && (Col_Idx < BlockCol + BlockS
                    if (Row_Idx != Col_Idx)
                        B[Col_Idx][Row_Idx] = A[Row_Idx][Col_Idx];
                    else {
                        tmp0 = A[Row_Idx][Col_Idx];
                        tmp1 = Row_Idx;
                    }
                }
                if (BlockRow == BlockCol)
                    B[tmp1][tmp1] = tmp0;
            }
        }
    }
}
```

make 명령어와 **./test-trans -M 32 -N 32** 명령어를 입력하여 준비를 한다.

```
202255513@CSEDell:~/HW4/cachelab-handout$ make
gcc -g -Wall -Werror -std=c99 -m64 -O0 -c trans.c
gcc -g -Wall -Werror -std=c99 -m64 -o test-trans test-trans.c cachelab.c trans.o
gcc -g -Wall -Werror -std=c99 -m64 -O0 -o tracegen tracegen.c trans.o cachelab.c
# Generate a handin tar file each time you compile
tar -cvf 202255513-handin.tar  csim.c trans.c
csim.c
trans.c
202255513@CSEDell:~/HW4/cachelab-handout$ ./test-trans -M 32 -N 32

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:1766, misses:287, evictions:255

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple Row_Idx-wise scan transpose): hits:870, misses:1183, evictions:1151

Summary for official submission (func 0): correctness=1 misses=287

TEST_TRANS_RESULTS=1:287
```

그리고, **./driver.py** 명령어를 사용하여 채점을 하게 되면, <u>total points</u>가 <u>53점</u>이 된다.

```
202255513@CSEDell:~/HW4/cachelab-handout$ ./driver.py
Part A: Testing cache simulator
Running ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     3 (1,1,1)       9       8       6       9       8       6  traces/yi2.trace
     3 (4,2,4)       4       5       2       4       5       2  traces/yi.trace
     3 (2,1,4)       2       3       1       2       3       1  traces/dave.trace
     3 (2,1,3)     167      71      67     167      71      67  traces/trans.trace
     3 (2,2,3)     201      37      29     201      37      29  traces/trans.trace
     3 (2,4,3)     212      26      10     212      26      10  traces/trans.trace
     3 (5,1,5)     231       7       0     231       7       0  traces/trans.trace
     6 (5,1,5)  265189   21775   21743  265189   21775   21743  traces/long.trace
    27


Part B: Testing transpose function
Running ./test-trans -M 32 -N 32
Running ./test-trans -M 64 -N 64
Running ./test-trans -M 61 -N 67

Cache Lab summary:
                        Points   Max pts      Misses
Csim correctness          27.0        27
Trans perf 32x32           8.0         8         287
Trans perf 64x64           8.0         8        1171
Trans perf 61x67          10.0        10        1809
          Total points    53.0        53
```