**Computer Architecture HW1 – Cache implementation**

**201822011 – Yumin Won**

1. 실행 환경 및 프로그램 실행 방법
2. 실행 환경

OS : ubuntu 16.04 (Linux)

Program Language : C

gcc version : 5.4.0

1. 실행 방법

압축파일 해제 >> config, main.c, Makefile

config 파일 설정 (init Capacity: 16, way : 4 block size :16)

$make >> main.c 컴파일 하여 cache 실행 파일 생성

$./cache >> 실행 파일 수행

Policly 선택 (1. PURE\_LRU 2. PSEUDO\_LRU 3. Exit)

정책에 따라Workload 파일 6개을 전부 실행한 뒤 각 workload xx.out 결과 파일들을 생성한 뒤 (**PSEUDO\_LRU의 경우 xx\_p.out 결과 파일 생성**), 다시 policy 수행 여부를 묻는다.

\*\* 제출된 코드의 구현 상태는 위에 언급된 상태로 구현된 상태입니다. ping\_trace.out에 대한 결과를 확인하려면 code내에 file\_name[num] -> trace\_name, file\_print() 인자로 file\_name[num] -> p\_output\_name 변경 후, main안에 while 문에 break를 작성하면 됩니다. (주석으로 표시해두었습니다.)

\*\* 실험한 결과 파일들은 액셀 및 cache\_data 폴더에 넣어 두었습니다.

1. Cache Analysis

* Cache 분석을 위해 case 3가지의 경우와 마지막 결론을 위한 실험을 수행
* 그래프의 y축은 miss rate를 나타낸다.

**CASE 1. Capacity change (16, 32, 64) – Fix way : 4 , block\_size : 16**

1. Pure LRU vs Pseudo LRU (for Read)

1. Pure LRU vs Pseudo LRU (for Write)

* Case1 Analysis
* Cache capacity를 늘릴 경우 Cache에 더 많은 Data block을 저장할 수 있기 때문에 miss율은 Cache capacity가 증가함에 따라 낮아지게 된다. 하지만 workload의 특성에 따라 miss rate의 정도가 다르게 나타나는 것으로 추축된다. 다른 workload들의 경우 Cache capacity가 증가 함에 따라 miss rate가 낮아지지만 Soplex와 libquantum의 workload의 경우 spatial locality 높은 request들의 영향 때문에 Cache capacity의 영향이 크게 보이지 않는 것으로 추측된다.

**CASE 2. Way change (4, 8, 16) – Fix Capacity : 16, block\_size : 16**

1. Pure LRU vs Pseudo LRU (for Read)
2. Pure LRU vs Pseudo LRU (for Write)

* Case2 analysis
* Way 개수를 증가시킨다면, Cache의 한 set안에 들어갈 수 있는 Data block 수가 많아지게 되므로 Cache hit 높아질 것이라 예측했다. 하지만 way가 증가함에 따라 set의 개수가 감소하기 때문에 결과가 크게 달라지는 점은 없었다.

**CASE 3. Block size change (16, 32, 64) – Fix Capacity : 16, way : 4**

1. Pure LRU vs Pseudo LRU (for Read)
2. Pure LRU vs Pseudo LRU (for Write)

* Case3 analysis
* Cache가 담을 수 있는 block size를 늘리면 Cache가 한번에 더 많은 양의 Data Block을 읽기 때문에 spatial locality의 정도가 높아지기 때문에 Cache hit가 발생할 확률이 더 높다. 따라서 위의 그래프에서 보이는 것처럼 povray workload를 제외한 대부분의 workload들에대해서 miss rate가 낮아지는 것을 볼 수 있다. 또한 Case1번과 Case2번에서 추측했던 soplex와 libquantum workload들이 spatial locality가 높은 request라는 사실이 맞다는 것이 관측 되었다. 그래서 Cache의 block size를 높임에 따라 두 workload들의 miss rate가 낮아지는 것을 확인할 수 있었다.

**CASE 4. Cache (16, 4, 16) vs Cache(64, 4, 64)**

1. Pure LRU

* Cache(16, 4, 16) (Capacity, way, block size)
* Cache(64, 4, 64) (Capacity, way, block size)

1. Pseudo LRU

* Cache(16, 4, 16) (Capacity, way, block size)
* Cache(64, 4, 64) (Capacity, way, block size)
* Case 4 Analysis
* 따라서 모든 워크로드들에 대한 miss rate를 줄이기 위해서는 Cache capacity 뿐만 아니라 Cache block size에 대해서도 고려를 해야함을 알 수 있었다.