Computer Architecture

Assignment #4

담당 교수님 : 이성원 교수님

컴퓨터정보공학부

2018202013

정영민

■ Introduction

● Goals and Objectives

이번 프로젝트는 프로세서에서 데이터를 요청할 때 해당 데이터를 memory에서 가져오는 과정에서의 AMAT를 구하여 최적의 cache 정보를 찾는 것이다. 고려하는 Cache는 L1 Cache와 L2 Cache가 있으며 L1과 L2의 가격 차이를 고려하며 여러 조건들을 바꿔 최적의 Cache를 찾는 것이 목표이다.

● Backgrounds

◆ Bubble Sorting

Bubble Sorting이란 서로 인접한 두 원소를 검사하여 정렬하는 알고리즘이다. 첫 번째 자료와 두 번째 자료를 비교하여 크기가 순서대로 되어있지 않으면 서로 교환한다. 이후 두 번째 자료와 세 번째 자료를, 세 번째 자료와 네 번째 자료를, … 이런 식으로 마지막-1 번째 자료와 마지막 자료를 비교하여 교환하면서 자료를 정렬한다. Bubble Sorting의 장점으로는 구현이 매우 간단하는 점이 있으며 단점으로는 자료의 비교, 교환 작업이 많이 일어나기 때문에 비효율적이다. 비교 횟수는 자료의 순서와 상관없이 모두 일정하며 교환 횟수의 시간 복잡도는 T(n) = O(n^2)이다.

◆ SPEC95

SPEC95 프로그램은 프로세서의 컴퓨팅 집약적 성능의 측정을 제공하기 위한 프로그램이다. 이때 메모리의 계층과 컴퓨터 시스템의 컴파일러의 구성 요소들은 비교 목적으로 사용된다. SPEC95는 CINT95 폴더 내의 Benchmarks를 사용한다. Benchmarks의 정보는 다음과 같다.

Benchmark	Application Area	Specific Task
cc1	GCC compiler	Based on the GNU C compiler version 2.5.3.
compress	Compression	Compresses large text files (about 16MB) using adaptive Lempel-Ziv coding
anagram	anagram	Performs text and numeric manipulations (anagrams/prime number factoring).

■ Result & Analysis of Lab

- **Sim 1**

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.3542	11.626	0.3827	0.2346	11.2739
128	0.2878	9.674	0.3249	0.1707	9.5302
256	0.235	8.1316	0.2691	0.1106	7.8627
512	0.1693	6.2039	0.2323	0.0746	6.8085

Sim1를 Benchmark CC1을 사용하였을 때의 표이다.

CC1의 inst access는 119208954이고 data access는 44468514이다.

Unified cache AMAT를 구하는 방식은 1.04^n+(Unified cache Miss rate * 30)으로 구하였으며 Split cache AMAT는 1.04^n+(% inst.)(inst. Miss rate * Miss penalty)+(% data)(data Miss rate * Miss penalty)의 식으로 구하였다. N은 Sets가 64일 때 0이다.

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.307	10.21	0.3226	0.1594	9.3785
128	0.1536	5.648	0.1991	0.1247	6.4206
256	0.0965	3.9766	0.0598	0.105	3.2355
512	0.0389	2.2919	0.0392	0.0879	2.6887

Sim1를 Benchmark Compress를 사용하였을 때의 표이다.

Compress의 inst access는 80432491이고 data access는 29063189이다.

AMAT 계산은 위와 같다.

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.2024	7.072	0.1573	0.2321	6.5307
128	0.1873	6.659	0.1539	0.2216	6.3917
256	0.1749	6.3286	0.146	0.2196	6.2603
512	0.1646	6.0629	0.1306	0.2186	5.9979

Sim1를 Benchmark Anagram을 사용하였을 때의 표이다.

Anagram의 inst access는 7070이고 data access는 4007이다.

이 또한 AMAT 계산은 위와 같다.

- **Sim 2**

L1I/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss rate	AMAT
8/8/1024	0.4578	0.3974	0.2512	32.4998
16/16/512	0.4255	0.3037	0.3928	40.1403
32/32/256	0.3827	0.2346	0.6089	49.9923
64/64/128	0.3249	0.1707	0.8216	53.5716
128/128/0	0.2691	0.1106	x	46.2076

Sim2를 Benchmark CC1을 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT는 L1 Hit time + ((% inst.)(Inst.Miss rate) + (% Data.)(Data.Miss rate)) * (L2 Hit time + (L2 Miss rate * L2 Miss penalty))의 식으로 구하였다.

L1I/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss rate	AMAT
8/8/1024	0.4355	0.3685	0.0433	13.4416
16/16/512	0.4035	0.2221	0.091	14.9988
32/32/256	0.3226	0.1594	0.2248	19.5142
64/64/128	0.1991	0.1247	0.4225	20.0465
128/128/0	0.0598	0.105	x	15.3595

Sim2를 Benchmark Compress를 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT 또한 위 식과 같다.

L1I/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss rate	AMAT
8/8/1024	0.1716	0.2573	0.536	26.9988
16/16/512	0.1653	0.2458	0.5989	28.4263
32/32/256	0.1573	0.2321	0.6576	29.2073
64/64/128	0.1539	0.2216	0.7185	30.5057
128/128/0	0.146	0.2196	x	35.5248

Sim2를 Benchmark Anagram을 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT 또한 위 식과 같다.

- **Sim 3**

Sets	Split Cache Miss rate / AMAT							
	1-way		2-way		4-way		8-way	
64	0.3425	11.275	0.2714	9.2028	0.2065	7.3203	0.1593	5.9727
128	0.283	9.53	0.2167	7.6042	0.1624	6.0423	0.1061	4.4245
256	0.226	7.8616	0.1639	6.0644	0.1076	4.4451	0.0493	2.7701
512	0.1895	6.8099	0.1193	4.7723	0.0593	3.0448	0.0273	2.1618
1024	0.1329	5.1569	0.0699	3.338	0.0291	2.1894	0.0142	1.8225
2048	0.0956	4.0847	0.042	2.5506	0.0162	1.8551	0.0069	1.6593

Sim3를 Benchmark CC1을 사용하였을 때의 표이다. Access수는 위와 같다.

AMAT는 $(1.04)^{n+k}(1.02)^{k}$+(% inst.)(inst. Miss rate * Miss penalty)+(% data)(data Miss rate * Miss penalty)이다. N은 sets가 64일 때 0이며 k는 Associative가 1일 때 0이다.

Sets	Split Cache Miss rate / AMAT							
	1-way		2-way		4-way		8-way	
64	0.2793	9.379	0.0937	3.8718	0.0234	1.8273	0.0209	1.8207
128	0.1794	6.422	0.0686	3.1612	0.0211	1.8033	0.018	1.7815
256	0.0718	3.2356	0.0215	1.7924	0.0181	1.7601	0.0147	1.7321
512	0.0521	2.6879	0.0183	1.7422	0.0149	1.7128	0.0119	1.6998
1024	0.0202	1.7759	0.0151	1.694	0.012	1.6764	0.0098	1.6905
2048	0.0172	1.7327	0.0125	1.6656	0.0099	1.6661	0.0079	1.6893

Sim3를 Benchmark Compress를 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT 또한 위 식과 같다.

Sets	Split Cache Miss rate / AMAT							
	1-way		2-way		4-way		8-way	
64	0.1842	11.275	0.1767	6.3618	0.1727	6.3063	0.1621	6.0567
128	0.1783	9.53	0.1718	6.2572	0.1637	6.0813	0.1477	5.6725
256	0.1726	7.8616	0.1633	6.0464	0.1491	5.6901	0.1414	5.5331
512	0.1624	6.8099	0.151	5.7233	0.142	5.5258	0.1414	5.5848
1024	0.1541	5.1569	0.1441	5.564	0.1414	5.5584	0.1414	5.6385
2048	0.1471	4.0847	0.1414	5.5326	0.1414	5.6111	0.1414	5.6943

Sim3를 Benchmark Anagram을 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT 또한 위 식과 같다.

- **Sim 4**

Block Size	Unified cache Miss rate	AMAT
16	0.1693	6.079
64	0.0396	2.2696
128	0.0203	1.7339
256	0.012	1.5299
512	0.0075	1.4417

Sim4를 Benchmark CC1을 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT는 1.04^n+(Miss rate * penalty)로 구하였다. N은 block size가 16일 때 0이다.

Block Size	Unified cache Miss rate	AMAT
16	0.0389	2.167
64	0.0126	1.4596
128	0.0111	1.4579
256	0.0042	1.2959
512	0.0032	1.3127

Sim4를 Benchmark Compress를 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT 또한 위 식과 같다.

Block Size	Unified cache Miss rate	AMAT
16	0.1646	5.938
64	0.0466	2.4796
128	0.0241	1.8479
256	0.014	1.5899
512	0.0092	1.4417

Sim4를 Benchmark Anagram을 사용하였을 때의 표이다.

Access수는 위와 같다.

AMAT 또한 위 식과 같다.

CC1의 경우 AMAT의 변화 폭이 가장 큰 부분이 Sim2이다. 따라서 가장 빠른 것을 선택하면 L1 Cache의 Sets는 16이고 L2 Cache의 Sets는 1024이다. Sim1에서 Sets가 256아래로는 Split이 유리한 것으로 보아 유추하면 Sets가 16일 경우 Split cache일 때 유리하다. Sim4에서 AMAT의 변화 폭을 보았을 때 Block size가 64인 경우 가장 폭이 컸으니 가장 적합한 Cache의 Block Size가 64라고 생각한다. Sim3에서 같은 Cache Size로 예상하였을 때 1-way와 2-way의 속도 차이가 크지 않다. 따라서 성능과 경제적 측면으로 따졌을 때 CC1의 Cache는 Split Cache이며 L1 Cache의 # of Sets가 16이고 L2 Cache의 # of Sets는 1024이고 Block size가 64이며 Associativity가 1인 경우가 가장 적합하다.

Compress의 경우는 # of Sets는 64일 때 적합하다고 생각한다. 이때 Split Cache가 빠르며 L2 Cache의 Sets는 256이다. 이때 2-way가 폭이 크므로 Associativity는 2라고 생각하며 size가 커지는 데에 비해 속도 개선이 크지 않기 때문에 적합한 Block size는 16이라고 생각한다. 따라서 성능과 경제적 측면으로 따졌을 때 Compress의 Cache는 Split Cache이며 L1 Cache의 # of Sets는 64이고 L2 Cache의 # of Sets는 256이고 Block Size는 16이며 Associativity가 2인 경우 가장 적합하다.

Anagram의 경우 # of Sets는 64일 때 적합하다고 생각한다. 이때 Split Cache가 빠르며 L2 Cache의 Sets는 256이다. Block Size는 64일 때 최적이라고 생각하고 이때 Associativity는 2일 때가 적합하다고 생각한다. 따라서 Anagram의 Cache는 Split Cache이며 L1 Cache의 # of Sets는 64이고 L2 Cache의 # of Sets는 256이고 Block Size는 64이며 Associativity가 2인 경우 가장 적합하다.

- **Sim 5**

# of Sets	Unified cache Miss rate	Unified cache AMAT	Split cache		Split cache AMAT
			Inst. Miss rate	Data Miss rate	
64	0.0276	1.828	0.0057	0.0424	1.4247
128	0.0178	1.574	0.0044	0.0329	1.369
256	0.0117	1.4326	0.0034	0.0283	1.3557
512	0.0087	1.3859	0.0017	0.026	1.3438

Unified cache와 Split Cache의 AMAT를 비교한 표이다. Split cache가 더 성능이 좋은 것을 확인할 수 있다.

L1I/L1D/L2U	Inst.Miss rate	Data.Miss rate	Unified Cache Miss rate	AMAT
8/8/1024	0.4653	0.0983	0.0164	10.2919
16/16/512	0.2739	0.0611	0.0309	7.1703
32/32/256	0.0057	0.0424	0.4137	2.5507
64/64/128	0.0044	0.0329	0.6533	2.7881
128/128/0	0.0034	0.0283	x	2.8274

L1/L2의 Size를 변경한 후 AMAT를 비교한 표이다. 32/32/256에서 제일 AMAT가 적고 감소폭이 큰 것을 확인할 수 있다.

Sets	Split Cache Miss rate / AMAT					
	1-way		2-way		4-way	
64	0.019	1.57	0.0113	1.3998	0.0104	1.4373
128	0.0147	1.481	0.0103	1.4122	0.0096	1.4583
256	0.0124	1.4536	0.0095	1.4324	0.0087	1.4781
512	0.0105	1.4399	0.0087	1.4543		
1024	0.0096	1.4579				
2048	0.0091	1.4897				

Associativity를 변경한 후 AMAT를 비교한 표이며, 유의미한 부분만을 사용하였다. 1-way 일 때는 sets가 512일 때 제일 좋은 AMAT를 나타내며 그 후로는 속도가 낮아진다. Sets가 64, 128, 256일 때까지는 2-way가 1-way보다 성능이 좋지만 512일 때부터 1-way의 성능이 2-way일 때보다 좋아진다. 4-way의 경우 해당 값에서 2-way보다 성능이 낮기 때문에 고려하지 않았다. 이는 8-way또한 마찬가지이다.

Block Size	Unified cache Miss rate	AMAT
16	0.0087	1.261
64	0.0013	1.1206
128	0.0008	1.1489
256	0	1.1699
512	0	1.2167

Block size의 크기를 변경하며 AMAT를 비교한 것이다. Block size가 64일 때 가장 빠른 AMAT를 가지며, 그 후로 커질수록 AMAT의 수가 커지는 것을 확인할 수 있다.

Lab의 결과를 보고 최적의 캐시를 생각하면 L1 Cache의 # of Sets는 64가 적절하다. 이때 L1 Cache는 Split Cache이며 L2 Cache의 # of Sets는 256이다. Associativity의 변화에 따른 AMAT 변화가 크지 않으므로 1이 가장 적절하며 Block size는 AMAT의 변화 폭이 크지 않으므로 16이 적절하다.

■ Discussion and Conclusion

Benchmark program과 Bubble sort program의 suitable cache가 다른 이유는 프로그램마다 AMAT의 변화 폭에 차이가 있으며, Cache의 가격을 생각하였을 때 가격대비 AMAT의 변화 폭이 크지 않은 경우엔 Cache의 크기를 늘릴 필요성이 없기 때문에 suitable cache가 다르다고 생각한다.

이번 프로젝트는 적합한 Cache를 찾는 것이었다. 이번 프로젝트를 하며 어려웠던 점은 Cache size가 커질 경우 가격이 얼마만큼 상승하는 지 잘 알지 못하여 적합한 Cache를 찾는 데에 어려움을 겪었다. Cache size가 커질 때 가격이 기하급수적으로 늘어난다고 생각하면 물론 size를 최대한 줄이는 것이 맞다는 판단을 하였는데, 만약 Size와 가격이 직선 그래프로 비례하여 증가한다면 그렇지 않다고 생각하였고, 또 가격이 비싸더라도 속도 향상 폭이 클 경우 size를 키울 수밖에 없는 경우도 생각하니 적합한 Cache 찾기가 어려웠다. 또, 해당 프로그램을 실행하는 데에 너무 오랜 시간이 걸려 결과를 적는 데에 많은 시간을 할애하게 되었다. Associativity의 AMAT를 구하는 식에서 Associativity가 2배가 되는 만큼 Cache Size가 2배가 되니 1.02만 곱하는 것이 아니라 추가로 1.04를 더 곱해야 한다는 것을 깨닫게 되었다. 그 외 어려움은 계산이 복잡하다는 것이 있었다.