n-way Set Associative Cache와 Fully Associative Cache의 성능분석

조 용 훈† · 김 정 선††

요 약

본 논문에서는 n-way set associative cache와 fully associative cache의 유용성 검증을 위하여 direct mapping, 2-, 4-, 8-, 16-way set associative mapping뿐만 아니라, 32-, 64-, 128-, 256-, 512-, 1024-, 2048-, 그리고 4096-way set associative mapping을 사용하는 캐쉬의 성능을 제안된 시뮬레이터 프로그램을 실행시켜 분석한다. 일반적으로 캐쉬 메모리 내에 있는 하나의 라인번호 내에 수용 가능한 주기억장치의 라인 수 n이 커짐에 따라 그 성능이 선형적으로 개선될 것으로 기대되지만, 본 논문의 분석에 따르면 512K 이상의 대용량 캐쉬에서는 n의 변화에 따른 성능 개선이 거의 없는 상태였고, 소용량 캐쉬의 경우에도 사용된 라인사이즈가 작은 경우 그 성능개선이 미미하였으며 라인사이즈가 비교적 큰 캐쉬에서는 괄목할 만한 성능개선이 있음을 확인하였다.

Performance Analysis of n-way Set Associative Cache and Fully Associative Cache

Yong Hoon Cho† · Jung Sun Kim††

ABSTRACT

In this paper, the performance of direct mapping caches, 2-, 4-, 8-, ···, 4096-way set associative caches, and fully associative caches are analyzed by trace simulation for verifying their effectiveness. In general, it is well known that as n, the number of main memory lines to be stored into one cache line number in direct mapping cache, increases, the performance of the cache memory should get higher linearly. According to our analysis, however, it is not true on all the cache organizations. It is shown that as n increases, miss ratios get lower only when the small cache(less than 256K) using large line size is used. It is also shown that fully associative mapping achieves high performance only when small size cache using large line size is used.

1. 서 론

캐쉬 메모리는 고속 소용량의 기억장치로서 주기억장치로 부터 가까운 장래에 중앙처리장치에 의해 사용될 가능성이 높은 부분을 미리 가져다가 기억하고 있다가 고속으로 제공하여 줌으로써 평균 기억장치 접근 시간을 줄여주는 중요한 기억장치이다. 일반적으로 캐쉬 메모리의 성능은 miss ratio와 bus traffic ratio에 의해 평가되며, 그 사용된 캐쉬 메모리의 용량, 채택된 라인사이즈(line size), 사상방식(mapping pro-

† 정 회 원:경북실업전문대학 전자계산과
†† 종신회원:한국항공대학교 항공전자공학과
논문접수:1996년 9월 17일, 심사완료:1997년 1월 29일

cedure), 기록방식(write policies), 교환방식(replacement algorithims), 그리고 그 실행되는 응용 프로그램의 시·공간적 국부성의 정도 등에 따라 매우 현격한 차이를 보이고 있다.[1] 본 논문에서는 전술한 파라미터들을 계획적으로 변경시켜 가면서 그 성능을 분석해 내는 시뮬레이터(simulator)를 사용하여, set associative mapping procedure를 사용하는 캐쉬 메모리의 경우 하나의 캐쉬 라인번호에 수용할 수 있는 주기억장치의 라인 갯수인 n을 달리해 가면서 최적의 구조를 찾아내기 위한 전략을 제시한다. 이를 위하여 본 논문에서는 그 사상방식으로서 direct mapping 방식과 n-way set associative mapping(n=2, 4, 8, 16, 32, ···, 4096)을 사용함으로써 기존의 연구에서 제시하지 못한 fully associative mapping구조의 분석을 가능하게 하였으며, 기록방식으로서는 write-back policy를 사용하였다. (write allocation방식 채택, 2.4절 참조) 그리고, 교환방식으로는 LRU(Least Recently Used)를 사용하였으며, 응용 프로그램용 트레이스로서는 미국에 소재하는 MORGAN KAUFMANN PUBLISHERS, INC.에 의해 제작되어 일반에게 자기 테이프로 제공되는 4개의 독립된 트레이스 (traces : lisp.000.din, pasc.000.din, forf.003.din, macr.000.din)를 사용 하였다.[2] 각 트레이스들은 그 국부성을 비롯한 여러가지 특성에 있어서 상당한 차이가 있으므로 객관성을 부여하기 위하여 4개 트레이스들 각각의 bus traffic ratio와 miss ratio를 구한 후 이들의 산술적 평균을 구해 분석에 사용하였다. 이들 시뮬레이션 결과를 이용하여 n-way set associative cache에 대한 fully associative cache의 유용성을 검증하기 위해 각 캐쉬 메모리 용량 별로 2-way에 대한 n-way 캐쉬의 성능 개선률을 구하여 그래프로 표시하여 분석에 사용하였다.

2. 시뮬레이션 개요

일반적으로 새로운 구조를 갖는 컴퓨터의 타당성을 제안하거나 특정 컴퓨터 설계 시 어떤 새로운 성능을 제시하고자 할 때, 그러한 컴퓨터를 실제로 제작하여 보여주는 것이 가장 좋은 방식이다. 그러나, 단지 그러한 성능 상의 진전을 보여 주기 위해서 완전한 컴퓨터 시스템을 실제로 제작한다는 것은 시간적·경제적으로 효율적이지 못하다. 이와같이 새로운 시스템의 제작과정에서 많은 파라미터들의 값을 변경시켜 가면서 분석하기 위해서는 시뮬레이션 기법이 가장 효율적이다.[3] 본 논문에서는 이러한 새로운 캐쉬 구조를 제안하기 위하여 C언어를 이용하여 시뮬레이터를 제작하여 UNIX환경(α-chip workstation)하에서 실행하였다. 이 시뮬레이터는 서론에서 언급한 바와 같은 각종 파라미터들을 변경시켜 가면서 여러가지 구조의 캐쉬 메모리의 성능을 분석해 낸다.

2.1 캐쉬 메모리의 용량

시뮬레이터는 DSCS(Desired Starting Cache Size ; 시뮬레이트할 가장 소용량 캐쉬의 용량을 선정하기 위한 값)를 입력받아 해당 용량의 캐쉬로 부터 시작하여 그 용량을 2배씩 증가시켜 가면서 8가지 크기의 캐쉬를 시뮬레이트 한다. 예를 들어, DSCS=13인 경우 2^{13}=8KB 캐쉬로 부터 16KB, 32KB, 64KB, 128KB, 256KB, 512KB, 1MB 캐쉬 메모리까지 분석한다.

2.2 라인사이즈(line sizes)

캐쉬 메모리와 주기억장치간 데이터 전송의 단위인 라인 사이즈는 위 2.1절의 각 캐쉬 메모리 용량에 대하여 기본적으로 4바이트 부터 시작하여 8, 16, 32, 64, 128, 256, 512바이트까지 늘려 가면서 분석한다. 따라서 본 시뮬레이터를 운영하면 한 번에 64가지 구조의 캐쉬 메모리를 분석할 수 있다.

2.3 사상방식(mapping procedures)

시뮬레이터는 오퍼레이터의 요구에 따라 direct mapping, 혹은 set associative mapping 중 어느 하나를 시뮬레이트 한다. Set associative mapping이 요구되면 한 캐쉬 라인번호에 수용할 수 있는 주기억장치 라인수(n)를 입력받아 해당되는 한 가지 set associative cache를 2.1과 2.2절의 64가지 캐쉬 구조들에 대해 분석한다. 이 경우 주기억장치 용량, 캐쉬 사이즈, 사용된 라인사이즈에 따라 fully associative mapping의 way수(n)가 결정된다. 본 논문에 제공된 표에서 0.00으로 표시된 란과 임의의 값이 표시된 란의 경계 부근에 위치하는 값들이 해당되는 캐쉬사이즈와 라인사이즈에 따른 fully associative mapping구조를 갖는 캐쉬의 성능에 해당된다.

2.4 기록방식(write policies)

현재까지 이 시뮬레이터는 write-back policy만을 분석한다. 추후 write-through와 write-once policy도 분석할 수 있도록 보강해 나아갈 계획이다. 여기서 사용된 write-back policy의 경우에는 읽기동작 뿐만 아니라 쓰기동작의 경우에도 miss가 발생되면 주기억장치로 부터 필요한 라인을 캐쉬로 인출하는 기법(write allocation이라 알려져 있다.)이 채택되었다.[5]

2.5 교환방식(replacement algorithims)

캐쉬 메모리가 가득 찬 상태에서(cache full) 캐쉬 메모리 접근 실패(miss)가 발생하였을 때 교환 대상 캐쉬 라인을 선정하기 위해 사용할 수 있는 교환방식에는 여러가지가 있다. Random algorithm, FIFO(First In/First Out) algorithm, LFU(Least Frequently Used) algorithm, 그리고 LRU(Least Recently Used) algorithm등의 교환방식 중에서 본 시뮬레이터는 가장 효율적이라 알려진 LRU방식을 사용한다. 이는 한 라인 번호 내에 존재하는 여러개의 라인 중에서 현 시점으로 부터 가장 오랫동안 사용되지 않은 라인을 교체대상으로 삼는다.

2.6 응용 프로그램

시뮬레이터는 사용될 응용 프로그램(traces)들의 화일이름을 요구한다. 트레이스 화일의 각 레코드는 8비트 label과 32비트 가상주소를 갖는다. Label은 메모리 접근의 목적(0;데이터 인출, 1;데이터 기록, 2;명령어 인출, 4;escape record;이 경우는 캐쉬를 완전히 비움)을 제시한다.

전술한 바와 같이 본 논문에서는 이러한 트레이스로서 미국 MORGAN KAUFMANN PUBLISHERS, INC.에서 자기 테이프로 제공한 여러개의 화일들 중 lisp.000.din, pasc.000.din, forf.003.din, 그리고 macr.000.din의 4개 화일들을 그대로 사용한다.[2] 시뮬레이터는 각 트레이스의 전체 메모리 접근 횟수 중 명령어 인출, 데이터 인출, 기록동작, 그리고 system call 과 같이 cache empty를 필요로 하는 접근의 비율을 결과 화일에 출력한다. 2.4절에서 설명한 바와 같이 write allocation을 채택한 경우 일반적으로 전체 메모리 접근 횟수 중 기록동작의 비율이 높을수록 그 성능(특히 bus traffic ratio)이 떨어진다.

3. 시뮬레이션 결과 분석

3.1 캐쉬의 성능과 라인사이즈

캐쉬의 성능은 가장 기본적으로 miss ratio로 판단한다. Miss ratio란 임의의 프로그램 수행 도중 발생하는 전체 메모리 접근 횟수에 대한 캐쉬 접근 실패 횟수의 비율을 의미하는 것으로서 그 값은 작을수록 좋다. 다음으로 사용되는 성능평가 기준이 bus traffic ratio이다. Bus traffic ratio란 임의의 프로그램을 캐쉬 메모리를 사용하지 않는 컴퓨터를 이용하여 실행시킬 때 중앙처리장치와 주기억장치 간에 주고 받는 총 데이터 수에 대한, 캐쉬 메모리를 사용하는 컴퓨터를 이용하여 실행시킬 때 발생되는 총 데이터 수의 비율을 의미한다. 캐쉬 접근 실패가 많이 발생하거나 주기억장치 기록동작이 많이 필요한 경우 이 ratio는 증가한다.

일반적으로 direct mapping이나 set associative mapping을 사용하는 캐쉬 구조에서는 캐쉬와 주기억장치 간 주고 받는 데이터의 단위인 라인사이즈가 클수록 miss ratio는 적어지는 반면에 bus traffic ratio는 커진다. 그러나 소용량 캐쉬에서는 라인사이즈가 커지면 캐쉬내에 기억될 수 있는 라인의 수가 상대적으로 적어져서 오히려 miss ratio가 커지는 현상도 발생된다.

〈표 1〉 2-way set associative mapping

Name of outfile = set2ot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 363212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	23.44	15.73	10.57	7.93	6.53	5.78	5.54	5.49
16	36.92	24.75	16.25	11.78	9.25	7.67	7.14	7.01
32	64.29	42.89	27.34	19.06	14.04	10.83	9.60	9.26
64	120.11	82.38	51.59	34.26	22.89	16.11	13.37	12.42
128	252.52	168.60	102.93	65.22	39.84	25.49	19.40	16.99
256	581.77	399.81	232.99	146.53	78.27	44.66	30.47	24.19
512	1563.46	1020.67	589.52	348.38	175.77	93.29	61.14	45.76

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	9.42	6.32	4.33	3.40	3.01	2.80	2.75	2.74
16	7.28	4.85	3.21	2.41	2.05	1.82	1.76	1.75
32	6.23	4.07	2.60	1.84	1.47	1.24	1.17	1.15
64	5.67	3.78	2.35	1.57	1.12	0.87	0.80	0.77
128	5.91	3.82	2.29	1.43	0.91	0.65	0.56	0.52
256	6.64	4.49	2.59	1.57	0.84	0.52	0.41	0.36
512	8.93	5.79	3.36	1.91	0.93	0.53	0.39	0.33

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 2〉 32-way set associative mapping

Name of outfile = set32tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	19.08	11.57	8.41	7.00	5.60	5.50	5.47	5.47
16	31.40	17.92	12.14	9.97	7.53	7.08	6.98	6.98
32	51.96	31.32	19.79	14.72	11.81	9.48	9.19	9.19
64	89.50	61.38	35.92	23.85	18.65	13.35	12.28	12.25
128	174.54	118.57	70.94	44.40	29.74	21.52	16.76	16.52
256	451.23	240.24	159.48	88.53	53.62	35.81	24.49	22.53
512	0.00	634.67	348.16	195.06	107.89	59.64	40.36	31.30

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	7.57	4.59	3.46	3.06	2.75	2.74	2.73	2.73
16	6.17	3.44	2.33	2.05	1.78	1.75	1.74	1.74
32	4.99	2.93	1.84	1.44	1.27	1.16	1.15	1.15
64	4.20	2.86	1.59	1.09	0.92	0.78	0.77	0.77
128	4.02	2.68	1.53	0.95	0.69	0.57	0.52	0.52
256	5.17	2.65	1.71	0.90	0.57	0.43	0.36	0.35
512	0.00	3.47	1.83	0.98	0.54	0.33	0.27	0.24

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 3〉 64-way set associative mapping

Name of outfile = set64tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	19.02	11.56	8.36	7.01	5.58	5.51	5.47	5.47
16	31.45	17.92	12.06	9.99	7.44	7.09	6.98	6.98
32	51.63	31.02	19.67	14.65	11.85	9.49	9.20	9.19
64	89.18	61.54	35.82	23.62	18.67	13.29	12.28	12.25
128	173.14	118.18	68.28	43.64	29.52	21.54	16.72	16.52
256	0.00	240.47	160.34	88.24	53.04	35.72	24.40	22.53
512	0.00	0.00	349.05	195.08	107.07	59.18	40.37	31.30

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	7.55	4.58	3.43	3.06	2.74	2.74	2.74	2.73
16	6.18	3.44	2.37	2.06	1.77	1.75	1.75	1.74
32	4.96	2.90	1.83	1.43	1.28	1.16	1.15	1.15
64	4.18	2.87	1.59	1.08	0.92	0.78	0.77	0.77
128	3.99	2.67	1.46	0.93	0.68	0.57	0.52	0.52
256	0.00	2.65	1.73	0.89	0.56	0.43	0.36	0.35
512	0.00	0.00	1.84	0.98	0.53	0.33	0.27	0.24

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 4〉 128-way set associative mapping

Name of outfile = set128tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	18.89	11.59	8.29	7.04	5.57	5.51	5.47	5.47
16	31.51	17.91	12.07	9.99	7.39	7.08	6.99	6.98
32	51.67	31.02	19.44	14.64	11.89	9.49	9.20	9.20
64	89.07	61.75	35.91	23.53	18.65	13.25	12.28	12.26
128	0.00	117.88	63.01	43.48	29.52	21.37	16.72	16.54
256	0.00	0.00	161.13	88.25	52.86	35.76	24.48	22.57
512	0.00	0.00	0.00	195.94	105.62	58.82	40.24	31.37

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	7.49	4.60	3.41	3.07	2.74	2.74	2.74	2.74
16	6.18	3.44	2.37	2.06	1.76	1.75	1.75	1.75
32	4.96	2.90	1.81	1.43	1.28	1.16	1.15	1.15
64	4.18	2.88	1.59	1.07	0.92	0.78	0.77	0.77
128	0.00	2.66	1.45	0.93	0.68	0.57	0.52	0.52
256	0.00	0.00	1.74	0.89	0.56	0.43	0.36	0.35
512	0.00	0.00	0.00	0.98	0.52	0.33	0.27	0.25

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 5〉 256-way set associative mapping

Name of outfile = set256tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	18.92	11.61	8.28	7.03	5.57	5.51	5.47	5.47
16	31.59	17.89	12.03	9.97	7.38	7.09	6.99	6.99
32	51.52	30.13	19.33	14.60	11.91	9.49	9.20	9.20
64	0.00	60.91	35.80	23.60	18.66	13.24	12.28	12.28
128	0.00	0.00	67.87	43.78	29.54	21.38	16.72	16.55
256	0.00	0.00	0.00	88.53	52.83	35.79	24.38	22.58
512	0.00	0.00	0.00	0.00	105.74	58.80	40.34	31.41

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	7.50	4.61	3.41	3.07	2.74	2.74	2.74	2.74
16	6.20	3.43	2.36	2.06	1.76	1.75	1.75	1.75
32	4.95	2.80	1.80	1.43	1.28	1.16	1.15	1.15
64	0.00	2.83	1.58	1.07	0.92	0.78	0.77	0.77
128	0.00	0.00	1.45	0.93	0.69	0.57	0.52	0.52
256	0.00	0.00	0.00	0.90	0.56	0.43	0.36	0.35
512	0.00	0.00	0.00	0.00	0.52	0.33	0.27	0.25

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

3.2 Direct mapping, set associative mapping과 fully associative mapping

본 논문에서는 2.1, 2.2절에서 설명한 구조의 캐쉬 메모리를 direct mapping과 2-, 4-, 8-, 16-way 뿐만 아니라 4096-way 캐쉬 구조까지 분석하여 줌으로써 fully associative mapping구조까지 분석할 수 있도록 하여 준다. 일반적으로 n-way set associative 캐쉬란 하나의 캐쉬 메모리 라인번호내에 수용할 수 있는 주기억장치의 라인 수가 복수개(n)인 direct mapping 캐쉬를 의미한다. 즉, n=1이면 direct mapping, n=2이면 2-way set associative mapping, n=4이면 4-way set associative mapping 등과 같다. 여기서 n이 커지면 캐쉬 메모

〈표 6〉 512-way set associative mapping

Name of outfile = set512tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	18.88	11.62	8.28	7.04	5.56	5.51	5.47	5.47
16	31.61	17.89	11.99	9.97	7.39	7.09	6.99	6.99
32	0.00	30.11	19.33	14.55	11.96	9.49	9.20	9.20
64	0.00	0.00	35.79	23.41	18.61	13.24	12.28	12.28
128	0.00	0.00	0.00	43.57	29.34	21.41	16.72	16.55
256	0.00	0.00	0.00	0.00	53.24	35.87	24.63	22.58
512	0.00	0.00	0.00	0.00	0.00	58.89	40.47	31.41

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	7.48	4.61	3.41	3.07	2.75	2.74	2.74	2.74
16	6.21	3.43	2.36	2.06	1.76	1.75	1.75	1.75
32	0.00	2.80	1.80	1.43	1.29	1.16	1.15	1.15
64	0.00	0.00	1.58	1.07	0.92	0.78	0.77	0.77
128	0.00	0.00	0.00	0.93	0.63	0.57	0.52	0.52
256	0.00	0.00	0.00	0.00	0.56	0.43	0.36	0.35
512	0.00	0.00	0.00	0.00	0.00	0.33	0.27	0.25

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 7〉 1024-way set associative mapping

Name of outfile = set1024tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	18.82	11.59	8.28	7.05	5.52	5.48	5.48	5.47
16	0.00	17.89	12.01	9.99	7.39	7.00	6.99	6.99
32	0.00	0.00	19.42	14.57	12.01	9.29	9.21	9.20
64	0.00	0.00	0.00	23.49	18.64	13.19	12.29	12.28
128	0.00	0.00	0.00	0.00	29.31	21.78	16.74	16.55
256	0.00	0.00	0.00	0.00	0.00	35.54	24.62	22.58
512	0.00	0.00	0.00	0.00	0.00	0.00	40.09	31.41

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	7.45	4.60	3.41	3.08	2.74	2.74	2.74	2.74
16	0.00	3.43	2.36	2.06	1.77	1.75	1.75	1.75
32	0.00	0.00	1.81	1.43	1.29	1.15	1.15	1.15
64	0.00	0.00	0.00	1.07	0.92	0.79	0.77	0.77
128	0.00	0.00	0.00	0.00	0.68	0.58	0.52	0.52
256	0.00	0.00	0.00	0.00	0.00	0.43	0.37	0.35
512	0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.25

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 8〉 2048-way set associative mapping

Name of outfile = set2048tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	0.00	11.61	8.27	7.05	5.48	5.48	5.48	5.48
16	0.00	0.00	12.02	10.00	7.35	6.99	6.99	6.99
32	0.00	0.00	0.00	14.59	12.02	9.21	9.21	9.21
64	0.00	0.00	0.00	0.00	18.65	12.65	12.29	12.29
128	0.00	0.00	0.00	0.00	0.00	21.90	16.57	16.57
256	0.00	0.00	0.00	0.00	0.00	0.00	22.67	22.60
512	0.00	0.00	0.00	0.00	0.00	0.00	0.00	31.45

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	0.00	4.61	3.41	3.07	2.74	2.74	2.74	2.74
16	0.00	0.00	2.36	2.07	1.77	1.75	1.75	1.75
32	0.00	0.00	0.00	1.43	1.29	1.15	1.15	1.15
64	0.00	0.00	0.00	0.00	0.92	0.77	0.77	0.77
128	0.00	0.00	0.00	0.00	0.00	0.58	0.52	0.52
256	0.00	0.00	0.00	0.00	0.00	0.00	0.35	0.35
512	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

〈표 9〉 4096-way set associative mapping

Name of outfile = set4096tot.d
Write Policy used = Write_back
Trace Program1 used = lisp.dat, 291390
Trace Program2 used = pasc.dat, 422090
Trace Program3 used = forf.dat, 368212
Trace Program4 used = macr.dat, 342828

Bus Traffic Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	0.00	0.00	8.27	7.06	5.48	5.48	5.48	5.48
16	0.00	0.00	0.00	10.00	7.23	6.99	6.99	6.99
32	0.00	0.00	0.00	0.00	12.08	9.21	9.21	9.21
64	0.00	0.00	0.00	0.00	0.00	12.48	12.29	12.29
128	0.00	0.00	0.00	0.00	0.00	0.00	16.57	16.57
256	0.00	0.00	0.00	0.00	0.00	0.00	0.00	22.60
512	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Miss Ratio

	8K	16K	32K	64K	128K	256K	512K	1M
8	0.00	0.00	3.41	3.07	2.74	2.74	2.74	2.74
16	0.00	0.00	0.00	2.07	1.75	1.75	1.75	1.75
32	0.00	0.00	0.00	0.00	1.30	1.15	1.15	1.15
64	0.00	0.00	0.00	0.00	0.00	0.77	0.77	0.77
128	0.00	0.00	0.00	0.00	0.00	0.00	0.52	0.52
256	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.35
512	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	Trace1	Trace2	Trace3	Trace4
Fraction of Inst. fetches	0.58	0.46	0.52	0.55
Fraction of Data fetches	0.34	0.29	0.29	0.28
Fraction of Writes	0.08	0.25	0.19	0.17
Number of Cache Flushes	0	0	0	0

리 내에 생성될 수 있는 라인번호의 수는 direct mapping의 1/n로 작아지고 주기억장치에 생기는 페이지의 수는 n배 많아진다. 예를 들면 n=2인 경우 주기억장치의 페이지 수는 direct 경우의 배로 늘어나고, 캐쉬의 한 개 라인번호 상에는 서로 다른 페이지로 부터 나오는 동일한 라인번호를 갖는 2개의 주기억장치 라인이 기억될 수 있는 캐쉬의 구조를 의미한다. 이들 두 라인은 주기억장치에서 서로 최대 k(k<m, 여기서 m은 주기억장치에 생성되는 페이지의 수로서 주기억장치 용량을 캐쉬 메모리 용량의 1/2로 나눈

수)페이지 만큼 떨어져 있을 수 있다. 따라서 캐쉬 내에 존재하는 라인번호의 수는 반으로 줄어든다.

n이 커짐에 따라 캐쉬 내에 존재하는 라인번호의 개수는 점차 줄어들게 되고 결국 캐쉬 내의 라인번호의 수가 하나로 줄어 들었을 경우를 fully associative mapping이라고 한다. 예를 들어, 32M(N=25;주기억장치 주소의 총 비트 수) 주기억장치와 256K direct 캐쉬 메모리의 경우 라인사이즈가 64(N_b=6;라인내 단어의 위치를 표시하기 위해 필요한 비트 수)라 한다면 주기억장치 주소 25비트 중 태그비트의 수(N_t; Tag bits의 수, 해당 라인이 소속되어 있는 주기억장치 페이지 번호를 표시하기 위해 필요한 비트 수)는 7비트(주기억장치 내에 32M/256K=$2^{25}/2^{18}=2^7$개의 페이지가 생성되므로)이므로 캐쉬 내 라인번호를 나타내는 비트 수(N_l)는 $N_l=N-N_t-N_b$, 즉 $N_l=25-7-6=12$가 되어 캐쉬와 주기억장치 페이지 내에 존재하는 라인의 수는 $2^{12}=4096$개 라인에 다다르게 된다. 그러나, 2-way set associative mapping의 경우는 $N_t=8$, $N_l=11$이 되며, 4-way의 경우에는 $N_t=9$, $N_l=10$ 등과 같이 되어 fully associative mapping의 경우, 즉 $N_l=0$가 되면 $N_t=19$가 되며 이때 n의 값은 4096(2^{12})이 된다. 즉, 32M 주기억장치와 256K 캐쉬 메모리가 라인사이즈 64를 사용한다면 이때 fully associative mapping은, n이 4096이 되므로, 4096-way set associative mapping이 된다. (그림 1)에 32M 주기억장치와 256K 캐쉬가 라인사이즈 64를 사용할 경우 그 주기억장치 주소의 구성과 n의 변화에 따른 태그비트와 라인번호 비트의 변화를 제시하였다.

N_t	N_l	N_b
tag bits, 7	line bits, 12	word bits, 6

(a)주기억장치주소의 구성

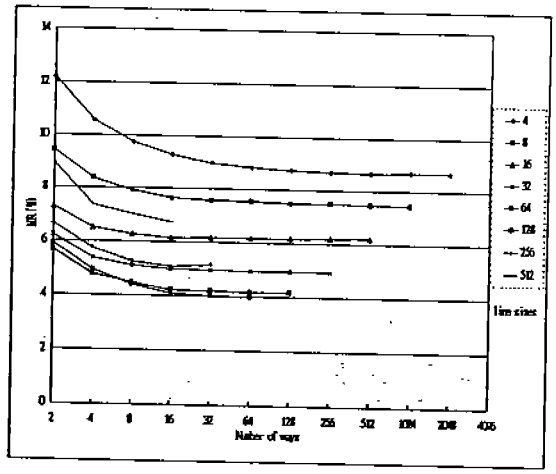
n	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}
N_l	12	11	10	9	8	7	6	5	4	3	2	1	0
N_t	7	8	9	10	11	12	13	14	15	16	17	18	19

(b) n에 따른 태그비트와 라인번호비트 수

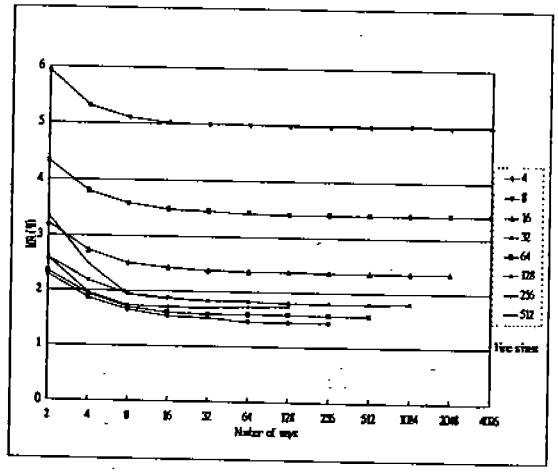
(그림 1) 32M 주기억장치와 256K 캐쉬가 라인사이즈 64 사용할 경우

이와같이 특정 용량의 주기억장치를 사용할 경우 fully associative mapping을 위한 n의 값은 캐쉬의 용량과 사용된 라인사이즈에 따라 결정된다. 예를 들면, 32M 주기억장치는 256K 캐쉬를 사용할 때 그 사용된 라인사이즈가 64이면 4096-way, 128이면 2048-way, 256이면 1024-way 등과 같고, 512K 캐쉬 사용 시 라인사이즈가 64라면 8192-way, 128이면 4096-way, 그리고 256이면 2048-way 등과 같다.

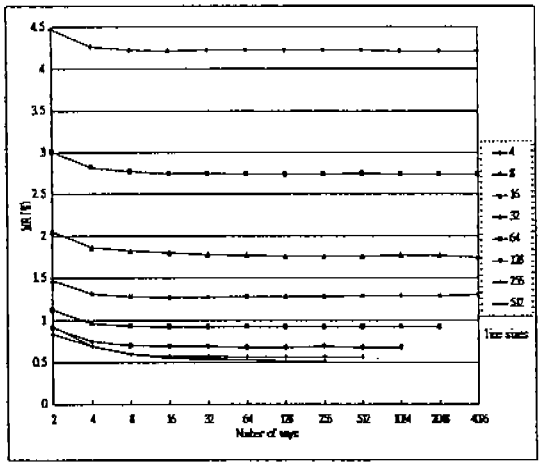
〈표 2〉에서 〈표 9〉까지에 32-way에서부터 4096-way까지의 시뮬레이션 결과를 제시하였고, (그림 2)에서 (그림 7)까지에 8K, 32K, 128K, 256K, 512K, 그리고 1M에 대한 miss ratios를 발췌하여 각 라인사이즈 별로 n값의 변화에 따른 성능 상의 개선 상태를 그래프



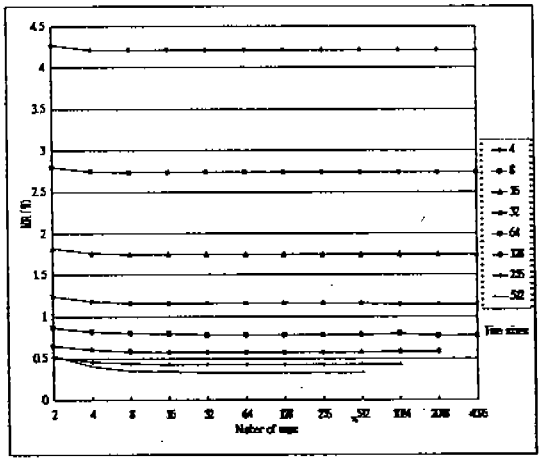
(그림 2) 8K 캐쉬의 miss ratios



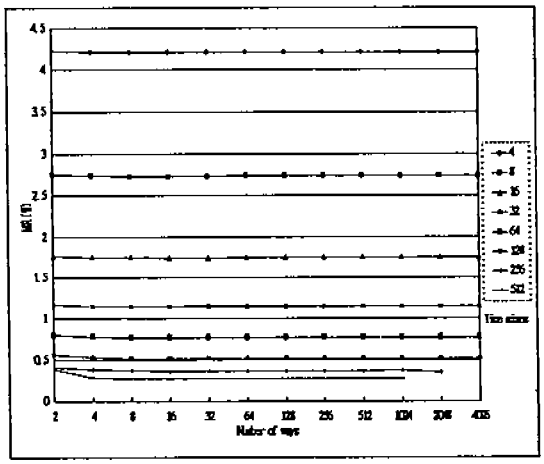
(그림 3) 32K 캐쉬의 miss ratios



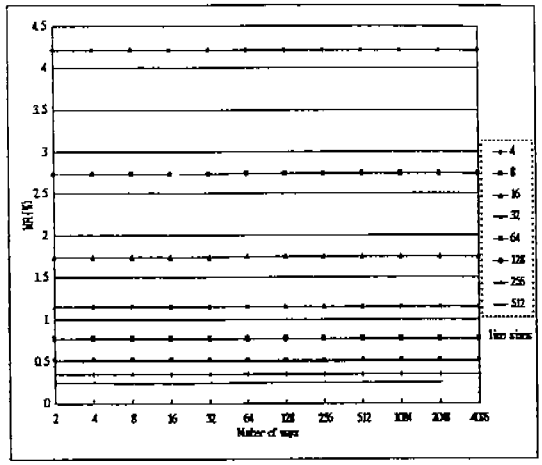
(그림 4) 128K 캐쉬의 miss ratios



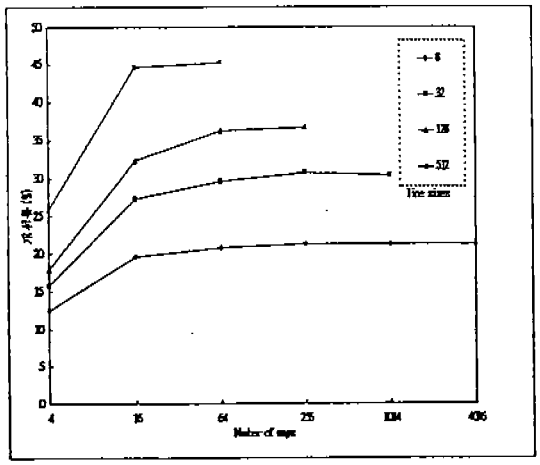
(그림 5) 256K캐쉬의 miss ratios



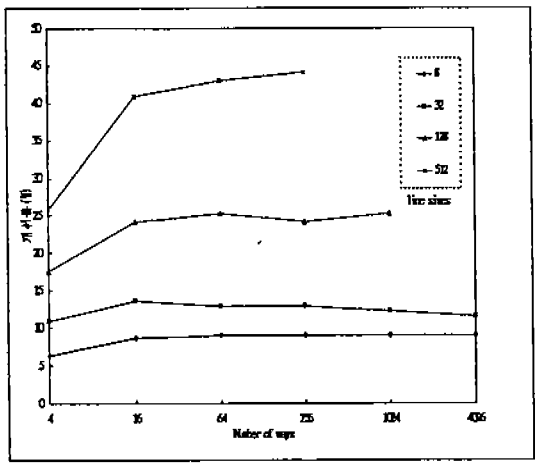
(그림 6) 512K 캐쉬의 miss ratios



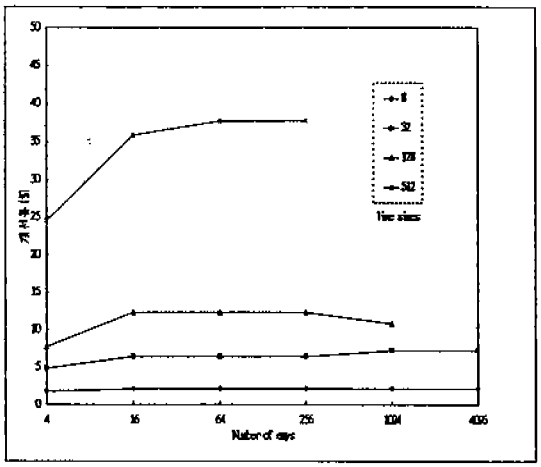
(그림 7) 1M 캐쉬의 miss ratios



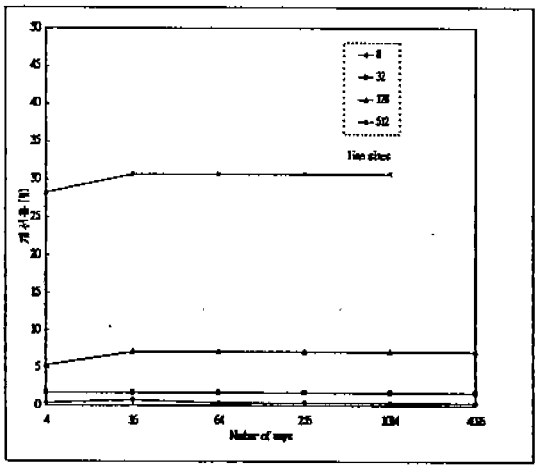
(그림 8) 32K 캐쉬의 2-way대비 개선률



(그림 9) 128K의 2-way대비 개선률



(그림 10) 256K의 2-way대비 개선률



(그림 11) 512K의 2-way대비 개선률

로 표시하였다. 위에서 언급한 표에서 0.00으로 표시된 부분과 그렇지 아니한 부분과의 경계선을 구성하는 부분과, 그림에서 각 라인사이즈 별 곡선이 끝나는 지점들이 각각 해당되는 캐쉬를 위한 fully associative mapping의 성능을 보여준다.

3.3 결과 분석

전 절에서 언급한 바와 같이 일반적으로 n이 커지면 캐쉬의 성능은 좋아지는 것으로 알려져 있다. 그러나 본 논문의 시뮬레이션 결과는 n값이 커짐에 따라 모든 구조의 캐쉬에서 선형적인 성능개선이 이루어지는 것은 아니라는 것을 보여준다.

〈표 1〉에서 〈표 9〉까지의 시뮬레이션 결과와 (그림 2)에서 7까지 제시된 도표의 이해를 돕기위해 8K, 32K, 128K, 256K, 그리고 512K 캐쉬 사용 시 n의 변화에 따른 2-way 대비 miss ratios 개선률을 (그림 8), 9, 10, 그리고 11에 나타내었다.

(그림 8)은 라인사이즈 8, 32, 128, 512 각각을 위한 32K miss ratio의 2-way 대비 4-, 16-, 64-, 256-, 1024-, 그리고 4096-way의 개선률을 보여주고 있는데, 라인사이즈 8을 사용할 경우 n이 증가함에 따라 fully associative mapping인 4096-way까지 그 성능이 꾸준히 향상됨을 보여주고 있다. 라인사이즈 512의 경우에는 n이 증가함에 따라 더욱 급격한 변화를 보여주고 있으나 64-way가 fully associative mapping에 해당되므로 더 이상 n을 증가 시킬 수가 없다. 따라서 소용량 캐쉬의 경우에는 fully associative mapping을 사용하는 것이 좋다. (그림 11)에서 라인사이즈 8의 경우 n의 변화에 따른 성능 상의 개선은 거의 없다. 라인사이즈 512의 경우에는 4-way와 16-way에서 2-way에 대해 약 30% 정도의 개선이 보이나 그 이후 거의 개선 효과가 없다.

이상의 분석으로 부터 우리는 소용량 캐쉬의 경우에는 n이 커질수록 유리해 진다는 것을 알 수 있다. 작은 n에 대해서는 그 사용된 라인사이즈가 클수록 개선효과는 더욱 커진다. 결론적으로 n이 커짐에 따라 소용량 캐쉬 일수록, 또한 사용된 라인사이즈가 클 수록 그 성능개선 효과가 크다. (그림 10)에 의하면 오늘날 흔히 사용되는 256K 캐쉬의 경우에는 라인사이즈 512 사용 시 fully associative mapping인 256-way까지 상당한 성능 개선이 있음을 보여준다.

4. 결 론

캐쉬 메모리의 성능은 그 사용된 용량, 라인사이즈 등 뿐만 아니라, 특히 실행되는 프로그램의 특성 등에 따라 커다란 차이를 보이고 있으므로 특정 프로그램 수행시 나타나는 성능만 가지고 그 시스템을 평가하기는 매우 어렵다. 따라서 본 논문에서는 복수 개의 프로그램 트레이스를 별도로 실행시킨 결과의 산술적 평균을 구해 분석에 사용함 으로써 그 결과의 신뢰도를 높혔다. 이러한 시뮬레이션 결과를 분석할 때 direct mapping 캐쉬구조 보다는 n-way set associ-

ative 캐쉬구조의 성능이 월등히 뛰어나다는 것은 자명한 일이다.([1] 참조) 이러한 n-way set associative 캐쉬 구조를 논할 때 일반적으로 n이 커지면 그 성능 또한 좋아질 것으로 기대된다. 그러나 (그림 8), 9, 10, 11에서 나타났듯이 근래에 많이 사용되고 있는 256KB 이하의 소용량 캐쉬 메모리를 갖는 컴퓨터의 경우에는 n=2로부터 4, 8, 16으로 증가시켜 감에 따라 그 성능개선이 뚜렷하게 나타나므로 fully associative mapping을 사용할 필요성이 있지만, 512K 이상 대용량 캐쉬의 경우에는 n의 증가에 따른 성능개선 효과가 거의 없으므로(특히, 작은 라인사이즈의 경우 더욱 없다.) fully associative mapping을 사용할 이유가 없을 것으로 판단된다.

향후, 가능하다면 현재 급속한 발전을 거듭하고 있는 IBM PC의 캐쉬 메모리를 집중 연구하여 본 논문의 결론과 부합되는 최적의 캐쉬 구조를 제안할 필요가 요구된다.

참 고 문 헌

[1] Y. H. Cho, J. S. Kim, "The Effects of Cache Memory on the System Bus Traffic", 한국통신학회논문지 제21권 제1호, Jan. 1996.

[2] J. L. Hennessy, D. A. Patterson, "Computer Architecture, a Quantitative Approach", Morgan Kaufmann Publishers, Inc., 1990.

[3] D. J. Lilja, "Cache Coherence in Large-Scale Shared-Memory Multiprocessors: Issues and Comparisons", ACM Computing Surveys, Vol. 25, No. 3, Sept. 1993, pp.303-312.

[4] M. D. Hill, A. J. Smith, "Evaluating Associativity in CPU Caches," IEEE Trans. on Computer, Dec. 1989.

[5] J. R. Goodman, "Cache Memory Optimization to Reduce Processor/Memory Traffic," Journal of VLSI and Computer Systems, Vol. 2, No. 1-2, 1987, pp.61-86.

조 용 훈

1979년 한국항공대학교 항공통신공학과 졸업(학사)

1985년 미국 Purdue 대학교 대학원 전기공학과(공학석사)

1996년 한국항공대학교 전자공학과 박사과정 수료

1992년 12월~1994년 2월 미국 Ball 주립대학교 교육부 교환교수

1986년~현재 경북실업 전문대학 전자계산과 부교수

관심분야: 컴퓨터 공학(병렬처리, 컴퓨터 구조)

김 정 선

1965년 한국항공대학교 항공전자공학과(공학사)

1972년 한양대학교 전자공학과(공학석사)

1983년 2월 경희대학교 전자공학과(공학박사)

1976년~현재 한국 통신학회 재무이사

1994년 6월~현재 전국전자계산소장협의회 회장

관심분야: 컴퓨터 구조학