교 육 학 석 사 학 위 논 문

Skewed-Associative 캐시를 위한 선형 해시 함수의 구성

2008년 8월

부경대학교 교육대학원

수 학 교 육 전 공

정 호 선

교 육 학 석 사 학 위 논 문

Skewed-Associative 캐시를 위한 선형 해시 함수의 구성

지도교수 조 성 진

이 논문을 교육학석사 학위논문으로 제출함.

2008년 8월

부경대학교 교육대학원

수 학 교 육 전 공

정 호 선

정호선의 교육학석사 학위논문을 인준함.

2008년 8월 27일

주 심 이학박사 표 용 수 (인)

위 원 이학박사 박 진 한 (인)

위 원 이학박사 조 성 진 (인)

목 차

표 목차

그림 목차

Construction of Linear Hash Functions

for Skewed-Associative Cache

Ho Seon Jeong

Graduate School of Education

Pukyong National University

Abstract

A hash function H is a computationally efficient function that maps bitstrings of arbitrary length to bitstrings of fixed length, called hash values. Hash functions have a variety of general computational uses. In this paper, we construct new XOR-based hash functions by using the concepts of rank and null space. These are conflict-free hash functions which are different type according to the number of output bitstrings is even or odd. To apply the constructed hash functions to the skewed-associative cache, we show that the degree of interbank dispersion between two hash functions is maximal.

Ⅰ. 서론

해시 함수는 임의의 이진 수열을 고정된 길이로 출력하는 함수이다. 이러한 해시 함수를 설계함에 있어서 고려되어야 할 사항들은 다음과 같다. 주어진 이진 수열에 대한 해시 값을 계산하는 것이 쉬워야 하며 임의의 입력 값으로부터 얻어진 해시 값이 주어졌을 때 그 입력 값을 계산하는 것이 어려워야 하고 충돌회피성을 가지고 있어야 한다. 또한 해시 함수는 상호 배치된 멀티 뱅크 메모리의 대역폭을 증가시키거나 캐시 이용의 향상을 위해 사용되기도 한다. 이를 위한 해시 함수는 무엇보다 해시 값을 계산하는 것이 간편하고, 빈번하게 일어나는 패턴에 대한 해시 값이 다르도록 설계되어야 한다. 이를 충돌 회피(Conflict-free)라 한다.

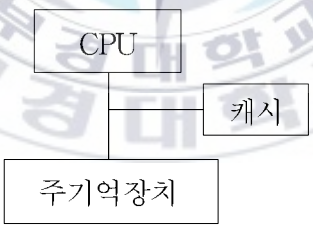
Seznec[1]은 멀티 뱅크 캐시에 대한 새로운 구성 방법으로서 skewed-associative 캐시 개념을 소개하였고, 2-way skewed-associative 캐시가 2-way set-associative 캐시와 하드웨어 복잡도는 같지만, 적중률은 같은 크기의 4-way set-associative 캐시의 적중률과 같음을 보였다[2].

본 논문에서는 주소 비트로부터 세트 인덱스를 계산하는 선형 해시 함수를 새롭게 구성한다. 이는 기존의 Frailong[3]이나 Vandierendonck[4,5] 등의 선형 해시 함수와는 다르게 주소 비트 수의 반이 짝수인 경우 뿐 아니라 홀수인 경우에 대해서도 충돌 회피가 가능하도록 각각 구성한다. 특히 같은 크기의 멀티 뱅크 캐시에서 set-associative 캐시보다 더 나은 성능을 가진 skewed-associative 캐시에 대하여, Vandierendonck 등은 skewed-associative 캐시에 대한 해시 함수의 충돌을 set-associative 캐시에 대한 해시 함수에 기저 변환을 이용 하여 충동 회피가 가능하도록 하였으나 본 논문에서는 기저 변환 없이 충돌 회피가 가능하도록 구성한다.

Ⅱ. 기반지식

2.1 캐시 메모리

캐시 메모리(Cache Memory)는 속도가 빠른 중앙처리장치(CPU)와 CPU에 비하여 상대적으로 속도가 느린 주기억장치 사이에서 원활한 정보의 교환을 위하여, 주기억장치의 정보를 일시적으로 저장하는 고속기억장치로서 그림과 같이 CPU와 주기억 장치 사이에 설치한다. 캐시 메모리는 주기억장치에 비하여 속도가 빠르기 때문에 컴퓨터의 처리속도를 빠르게 할 수 있지만, 주기억장치에 비하여 가격이 비싸기 때문에 일반적으로 수 MB의 주기억장치를 사용하는 시스템에서는 수십 KB 정도의 캐시 기억장치를 사용한다.



<그림 2.1.1 캐시 메모리의 위치>

CPU가 기억장치로부터 어떤 데이터를 읽으려고 할 때는 먼저 그 데이터가 캐시에 있는지 검사한다. 만약 있다면, 데이터는 즉시 CPU로 전달되어 액세스 시간이 크게 단축된다. 그러나 만약 없다면, 그 데이터는 주기억장치로부터 인출되어야 하기 때문에 더 오랜 시간이 걸린다. 이와 같이

CPU가 프로그램 코드나 데이터를 처음으로 액세스할 때는 주기억장치로부터 직접 읽어 와야 하지만, 일단 캐시에 적재되어 있는 프로그램 코드나 데이터가 CPU에 의해 다시 사용될 때는 캐시로부터 신속히 읽어올 수 있다.

캐시의 사용 목적은 평균 기억장치 액세스 시간을 단축하는 것이고, 이를 위해서는 가능한 한 적중률을 높여야 한다. 그러나 지역성은 각 프로그램의 특성에 따라 달라지는 것이므로 한계가 있다. 따라서 캐시 설계 과정에서는 캐시의 적중률을 높이는 것 외에도 여러 사항을 고려해야 한다. 캐시 설계에 있어서의 공통적인 목표는 캐시 적중률의 극대화, 캐시 액세스 시간의 최소화, 캐시 실패에 따른 지연시간의 최소화, 주기억장치에 캐시간의 데이터 일관성 유지 및 그에 따른 오버헤드의 최소화 등으로 요약할 수 있다.

캐시는 주기억장치보다 메모리의 크기가 훨씬 더 작기 때문에 캐시의 슬롯이 여러 개의 주기억장치 블록들에 의해 공유된다고 하였다. 그렇게 되도록 하기 위해서는 어떤 주기억장치 블록들이 어느 캐시 슬롯을 공유할 것인지를 결정해 주는 방법이 필요하다. 그 방법을 주기억장치와 캐시간의 사상방식(mapping scheme)이라고 하는데, 캐시 적중률에도 많은 영향을 미치는 주요 설계 요소이다. 캐시의 내부 조직은 사상 방식에 따라 결정된다. 가장 널리 사용되고 있는 사상 방식으로는 세 가지가 있는데 직접 사상(direct mapping), 완전-연관 사상(fully-associative mapping) 및 세트-연관 사상(set-associative mapping)방식 등이다[6].

몇 년 전만 해도, 직접 사상 캐시 체제는 마이크로프로세서의 캐시에서 최적의 체제로서 간주되어 왔다. 그러나 기술의 발전으로 첫 번째 세대인 RISC 마이크로프로세서와 관련 있었던 거대한 외부 캐시는 현재 작은 On-chip 캐시로 대체되고 있다.

2.2 해시 함수

2^n 개의 주소 집합 $\{0, 1, 2, \cdots, 2^n - 1\}$에서 2^m 개의 인덱스의 집합 $\{0, 1, 2, \cdots, 2^m - 1\}$로의 전사 함수를 해시 함수라 정의한다. 캐시 메모리는 두 가지 목적으로 해시 함수를 사용하는데, 캐시 메모리에서 주소를 찾을 때 탐색하는 시간을 제한하고 여러 개의 연결된 독립된 뱅크에서 동시에 접근을 허락하여 하드디스크 드라이브의 성능을 높이기 위하여 데이터가 인접하지 않도록 배열하는 방식을 사용하기 위한 것이다.

해시 함수에서는 상당히 간단한 연산을 사용하는데, 대부분 계산하는데 부담이 적은 XOR 연산을 이용한 함수를 사용한다. 해시 함수는 데이터를 자르고 치환하거나 위치를 바꾸는 등의 방법을 사용해 결과를 만들어 내며, 이 결과를 흔히 해시 값(hash value)이라 한다.

해싱(Hashing)은 하나의 문자열을 원래의 것을 상징하는 더 짧은 길이의 값이나 키로 변환하는 것이다. 짧은 해시키를 사용하여 항목을 찾으면 원래의 값을 이용하여 찾는 것보다 더 빠르기 때문에, 해싱은 데이터베이스 내의 항목들을 색인하고 검색하는데 사용된다. 또한 해싱은 빠른 속도의 데이터 색인 및 검색 외에도, 전자서명을 암호화하고 복호화 하는 데에도 사용된다. 전자서명은 해시 함수를 이용하여 변환된 다음, 해시 값, 즉 요약 메시지와 전자서명이 별도로 전송된다. 수신자는 송신자가 사용한 해시 함수와 같은 것을 사용하여, 서명으로부터 요약 메시지를 뽑아내어 그것을 이미 수신한 요약 메시지와 비교한다. 그리고 그 비교 결과가 같아야 전자서명을 유효한 것으로 판단한다. 해싱은 항상 한 쪽 방향으로만 연산된다.

우수한 해시 함수는 결정론적으로 작동해야 하므로 서로 다른 두 개의 입력에 대해, 동일한 해시 값을 생산해서는 안 된다. 왜냐하면 동일한 해시

값 사이에 충돌이 많이 날수록 서로 다른 데이터를 구별하기 어려워지고 데이터를 검색하는 비용을 늘리기 때문이다. 따라서 충돌 위험성이 매우 적은 해시 함수라야 훌륭한 해시 함수로 평가된다.

2.3 선형 해시 함수

n비트 주소를 $m(<n)$비트 세트 인덱스로 사상하는 선형 기반의 해시 함수는 $n\times m$ 행렬로 표현할 수 있다. $GF(2)$위에서 주소를 나타내는 벡터 a와 $n\times m$ 행렬(H)의 곱으로 해시 값인 세트 인덱스(s)를 다음과 같이 계산할 수 있다.

$$s = aH$$

여기서, 덧셈은 XOR 연산이고 곱셈은 AND 연산이다.

행렬은 선형변환이므로, 그것의 계수(rank)와 영공간(null space)에 의해 특성화될 수 있다. 행렬 H의 계수는 H의 일차독립인 열 또는 행의 개수이고, 행렬 H의 영공간은 $N(H)=\{x: xH=0\}$이다. 여기서, H가 $n\times m$ 행렬일 때 다음을 만족한다[7,8].

$$n = \mathrm{rank}(H) + \dim N(H)$$

두 행렬의 영공간에 대하여 다음 정리를 얻는다.

<정리 2.3.1> 두 $n\times m$ 행렬 A, B와 $n\times 2m$ 첨가행렬 $[A\,B]$에 대하여 다음이 성립한다.

$$N(A)\cap N(B) = N([A\,B])$$

(증명) 이 정리의 증명은 다음의 동치관계로부터 얻어진다.

$$x\in N(A)\cap N(B) \Leftrightarrow xA = xB = 0 \Leftrightarrow x\in N([A\,B]) \qquad \square$$

영공간이 포함된 패턴의 공통부분이 오직 영벡터만을 포함할 때 충돌회피 사상이 된다. 때때로 특정한 패턴들이 충돌이 없도록 보내져야 한다는 것이 알려져 있다. 그런데 그런 기능을 수행하는 해시함수를 구성해야 하는 어려운 점이 있지만, 여기서 한 예를 들어 그런 해시함수를 구성하는 방법을 제시하고자 한다.

모든 패턴은 $2^m \times 2^m$ 배열의 부분집합이고, 각각 2^m개의 세트(예컨대, 인터리브 기억장치에서 캐시의 2^m개 세트 또는 2^m개 뱅크)에 저장될 것이라 가정한다. 각 패턴은 충돌 없이 저장되어야 할 2^m개 원소로 이루어져 있다. 패턴의 원소는 m비트 행 번호 i와 m비트 열 번호 j를 $x=[i \mid j]$처럼 연이어 붙여서 해시함수의 입력으로 사용한다. 패턴의 임의의 서로 다른 두 원소 x, x'에 대하여 $xH \neq x'H$일 때 패턴이 충돌 없이 저장된다고 한다.

$$xH = x'H \Leftrightarrow 0 = xH \oplus x'H = (x \oplus x')H \Leftrightarrow x \oplus x' \in N(H)$$
$$xH \neq x'H \Leftrightarrow x \oplus x' \notin N(H)$$

그러므로 $N(H)$가 패턴에 있는 임의의 서로 다른 두 원소 x, x'의 합 $x \oplus x'$을 포함하지 않도록 해시함수 H를 구성하면 될 것이다.

2^m개 원소로 이루어진 각 패턴에서 서로 다른 두 원소 x, x'로 이루어진 쌍 $x \oplus x'$은 ${}_{2^m}C_2 = \frac{2^m(2^m-1)}{2}$ 개 이므로, ${}_{2^m}C_2 = \frac{2^m(2^m-1)}{2}$ 개 각 $x \oplus x'$에 대하여 $x \oplus x' \notin N(H)$ 인지를 계산하는 것은 아주 어렵다. 그러나 많은 경우의 패턴들이 벡터공간임이 드러났기 때문에[9], 문제가 간단해졌다. 패턴의 임의의 두 원소의 XOR가 패턴의 원소일 때 그 패턴을 벡터공간이라고

한다. 이런 패턴들에 대해서는 ${}_{2^m}C_2 = \frac{2^m(2^m-1)}{2}$개의 쌍이 벡터공간의 2^m개 원소만큼으로 줄어든다. 왜냐하면 2^m개 원소로 이루어진 벡터공간 V에서 임의의 두 원소는 XOR가 V의 원소이므로, V의 임의의 두 원소의 합으로 나타날 수 있는 경우는 2^m개 뿐 이고, 영벡터를 제외한 2^m-1개 원소들이 $N(H)$에 속하지 않는다는 것만 확인하면 충돌이 발생하지 않는다는 것을 알 수 있다.

즉,

$$xH \neq x'H \Leftrightarrow x \oplus x' \notin N(H)$$

기본패턴(basic pattern) $V([O \mid j], [i \mid O], [i \mid i], i, j \in GF(2)^{1\times m})$ 와 H의 영공간 $N(H)$와의 교집합이 $\{0\}$인 경우에 충돌이 일어나지 않았다고 한다.

<정리 2.3.2> 벡터공간 V에 대하여 $V \cap N(H) = \{0\}$일 때 충돌 없이 함수를 얻을 수 있다.

(증명) V는 벡터공간이고 $V \cap N(H) = \{0\}$이므로

$$aH = bH \Rightarrow a+b \in N(H) \Rightarrow a+b = 0 \Rightarrow a = b \qquad \square$$

2.4 2-Way Associative 캐시

1997년 set-associative 캐시를 제안한 Smith[10]는 메인 메모리의 주소에 대한 해싱(hashing)은 집합의 선택에 있다고 암시하였다. 이는 하나의 해시 함수가 사용된 그림 2.4.1에서 잘 나타나고 있다.

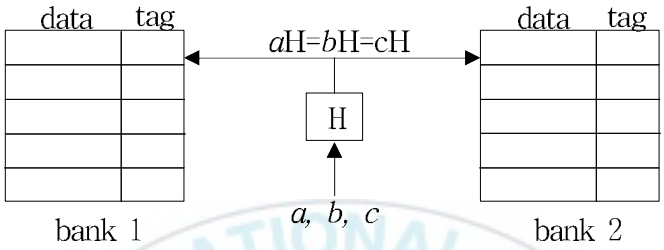
최근에 Agarwal[11]는 hash-rehash 캐시에 대해 연구하였다. 전통적인 캐시에서는, 주소 인덱스가 캐시 집합에 들어가고 만약 그곳에서 데이터를 찾게 된다면 그 데이터의 단어를 프로세서로 보내어지는데, 이런 경우를 first time hit 라 부른다. 만약 처음 캐시 집합에서 데이터를 찾지 못한다면, 주소는 다시 사용하지 않은 다른 해시 함수로 들어간다. 만약 두 번째 액세스에서도 역시 데이터를 찾지 못하다면, 데이터는 중앙처리장치에서 꺼내게 된다.

Hash-rehash 캐시는 직접 사상 캐시보다 더 나은 overall ratios을 제안한다. 그러나 Agarwal은 64KB 캐시와 hash-rehash 캐시는 2-way set-associative 캐시보다 더 긴 실행 시간을 야기한다고 비판하였다.

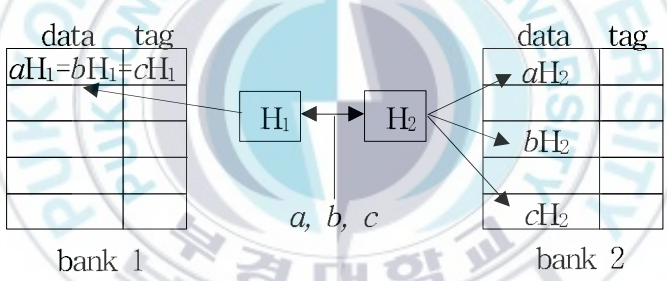
Hash-rehash 캐시는 처음의 캐시가 사용되어진 후 first time miss 이후에 두 번째 캐시를 사용하였다. 그러나 skewed-associative 캐시에서는 다른 해시 함수를 별개의 캐시 뱅크를 찾아보기 위해 동시에 사용하였다.

Skewed-associative 캐시 메모리에서는 두 개의 해시 함수가 사용된다. 즉, 2-way skewed-associative 캐시의 경우, 주소 a는 캐시 뱅크 1에서는 해시 값 aH_1에 사상되고, 캐시 뱅크 2에서는 해시 값 aH_2에 사상된다. 2-way set-associative 캐시는 그림 1에서와 같이 해시 함수가 하나이므로 주소 a, b, c에 대해 뱅크 1과 뱅크 2의 해시 값이 같지만, 2-way

skewed-associative 캐시는 그림 2에서와 같이 주소 a, b, c가 뱅크 1에서는 같은 해시 값을 가져 충돌이 발생하지만, 뱅크 2에서는 해시 값이 충돌 없이 분산된다.



<그림 2.4.1> Set-Associative 캐시



<그림 2.4.2> Skewed-Associative 캐시

이렇게 뱅크 1에서 충돌이 발생하는 주소들에 대해 뱅크 2에서 해시 값이 얼마나 충돌 없이 분산되느냐를 뱅크간의 분산정도(degree of interbank dispersion, 이하 DID)라 한다.

2-way skewed-associative 캐시는 2-way set-associative 캐시와 동일한 하드웨어 복잡성을 가지고 있으나, 시뮬레이션으로는 크기가 같은 4-way set-associative 캐시와 같은 적중률을 전형적으로 보여준다. 그렇

다면 skewed-associative 캐시가 set-associative 캐시보다 더 낫다. 그리고 2-way skewed-associative 캐시의 hit time은 직접 사상 캐시의 hit time에 매우 가깝다. 따라서 2-way skewed-associative 캐시는 오늘날의 one-chip 캐시를 가진 마이크로프로세서에서 바람직한 절충안이다.

Ⅲ. 선형 해시 함수

이 장에서는 주소 비트로부터 세트 인덱스를 계산하는 선형 해시 함수를 새롭게 구성한다. 이는 기존의 XOR기반의 선형 해시 함수와는 다르게 주소 비트 수의 반이 짝수인 경우 뿐 아니라 홀수인 경우에 대해서도 충돌 회피가 가능하도록 주소비트($2m$)의 절반(m)이 홀수인 경우와 짝수인 경우를 나누어 해시 함수를 정의한다.

<정의 3.1.1> 주소비트 수가 $2m$이라 하자. 주소비트에서 m비트 세트 인덱스로 사상하는 해시 함수를 다음과 같이 정의한다.

(i) $m=2k-1 \ (k\in\mathbb{N})$

$$H_1=\begin{pmatrix}T_1\\ I_m\end{pmatrix}, \quad H_3=\begin{pmatrix}T_3\\ I_m\end{pmatrix}$$

여기서 I_m은 $m\times m$ 단위행렬이고,

$$T_1=(t_{ij})_{m\times m}, \quad t_{ij}=\begin{cases}1,\ 1\le i\le k,\ 1\le j\le k-i+1\\ 1,\ 1\le i\le 2k-1,\ j=2k-i\\ 0,\ \text{그 외}\end{cases}$$

이고

$$T_3=(t_{ij})_{m\times m}, \quad t_{ij}=\begin{cases}1,\ i=j=1\\ 1,\ 1\le i\le 2k-1,\ j=2k-i\\ 1,\ k\le i\le 2k-1,\ 3k-i-1\le j\le 2k-1\\ 0,\ \text{그외}\end{cases}$$

이다.

(ii) $m=2k\ (k\in\mathbb{N})$

$$H_2=\begin{pmatrix}T_2\\ I_m\end{pmatrix},\qquad H_4=\begin{pmatrix}T_4\\ I_m\end{pmatrix}$$

여기서 I_m은 $m\times m$ 단위행렬이고,

$$T_2=(t_{ij})_{m\times m},\quad t_{ij}=\begin{cases}1,\ 1\le i\le k,\ i\le j\le k\\ 1,\ 1\le i\le 2k,\ j=2k-i+1\\ 0,\ \text{그 외}\end{cases}$$

이고

$$T_4=(t_{ij})_{m\times m},\quad t_{ij}=\begin{cases}1,\ k+1\le i\le 2k,\ k+1\le j\le i\\ 1,\ 1\le i\le 2k,\ j=2k-i+1\\ 0,\ \text{그 외}\end{cases}$$

이다.

예를 들어 $m=7$인 경우의 T_1, T_3와 $m=6$인 경우의 T_2, T_4는 각각 다음과 같다.

$$T_1=\begin{pmatrix}1&1&1&1&0&0&1\\1&1&1&0&0&1&0\\1&1&0&0&1&0&0\\1&0&0&1&0&0&0\\0&0&1&0&0&0&0\\0&1&0&0&0&0&0\\1&0&0&0&0&0&0\end{pmatrix},\ T_3=\begin{pmatrix}1&0&0&0&0&0&1\\0&0&0&0&0&1&0\\0&0&0&0&1&0&0\\0&0&0&1&0&0&1\\0&0&1&0&0&1&1\\0&1&0&0&1&1&1\\1&0&0&1&1&1&1\end{pmatrix},\ T_2=\begin{pmatrix}1&1&1&0&0&1\\0&1&1&0&1&0\\0&0&1&1&0&0\\0&0&1&0&0&0\\0&1&0&0&0&0\\1&0&0&0&0&0\end{pmatrix},\ T_4=\begin{pmatrix}0&0&0&0&0&1\\0&0&0&0&1&0\\0&0&0&1&0&0\\0&0&1&1&0&0\\0&1&0&1&1&0\\1&0&0&1&1&1\end{pmatrix}$$

해시 함수들은 해시 값의 계산이 간편하고, 빈번하게 일어나는 패턴에 대한 해시 값이 다르도록 설계되어야 한다. 왜냐하면 뱅크간의 충돌은 인터리브 멀티 뱅크의 대역폭을 감소시키고, 캐시 미스율을 증가시키기 때문이다.

다음 두 정리는 정의 3.1.1에서 제안된 해시 함수가 빈번하게 일어나는 패턴에 대하여 충돌이 발생하지 않음을 보장한다.

<정리 3.1.1> 정의 3.1.1의 T_i와 $T_i \oplus I$는 모두 정칙행렬이다 $(i = 1, 2, 3, 4)$.

(증명) 1) T_1의 행렬식은 여인수 전개를 통하여 $k \times k$ 소행렬식만 남게 된다. 여기서 1행부터 $k-1$행을 순차적으로 k행에 더하면

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

이 된다. 따라서 $|T_1| = 1$이므로 T_1은 정칙행렬이다.

T_2에 대한 증명도 위와 유사하다.

2) $T_1 \oplus I$의 행렬식은 $k+j$행에 $k-j$행을 더하면 $\begin{vmatrix} A & C \\ B & O \end{vmatrix}$형태가 되고, 이를 $k-2$번 여인수 전개하면 $\begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix}$이 되어 $|T_1 \oplus I| = 1$이다. 따라서 $T_1 \oplus I$는 정칙행렬이다.

$T_2 \oplus I$에 대한 증명도 위와 유사하다. □

<정리 3.1.2> $T = (t_1, t_2, \cdots, t_m)^t$ 이고 $I_m = (e_1, e_2, \cdots, e_m)^t$인 정의 3.1.1의 해시 함수 $H = \begin{pmatrix} T \\ I_m \end{pmatrix}$에 대하여 M_{ij}을 T에서 $m+1-i$행부터 i개의 행을 선택하고 I에서 $m+1-j$행부터 j개의 행을 선택하여 얻은 행렬이라 하자.

즉, $m \times m$행렬 $M_{ij} = (t_{m+1-i}, \cdots, t_{m-1}, t_m, e_{m+1-j}, \cdots, e_m)^t$ 이라 하면 $\mathrm{rank}(M_{ij}) = m$이다. 여기서, $i+j=m$, t_i는 T의 i행 벡터이고, e_i는 기본 단위 행벡터, A^t는 A의 전치행렬이다.

(증명) 1) $j=0$인 경우

$M_{ij} = T$이므로 정리 3.1.1에 의하여 $\mathrm{rank}(M_{ij}) = m$이다.

2) $1 \le j \le m-1$인 경우

$M_{ij}' = (t_m, t_{m-1}, \cdots, t_{m+1-i}, e_{m+1-j}, \cdots, e_m)^t$ 라 하자. T가 T_3 또는 T_4인 경우는 M_{ij}'이 주대각선의 성분이 모두 1인 상 삼각행렬이므로 $|M_{ij}'| = 1$이다. 따라서 M_{ij}'의 모든 행들은 일차독립이다. 그런데 M_{ij}와 M_{ij}'는 행동치 이므로 $\mathrm{rank}(M_{ij}) = \mathrm{rank}(M_{ij}') = m$이다.

같은 방법으로 T가 T_1 또는 T_2인 경우는 M_{ij}'이 주대각선의 성분이 모두 1인 하 삼각행렬이므로 $|M_{ij}'| = 1$이다. 따라서 M_{ij}'의 모든 행들은 일차독립이다. 그런데 M_{ij}, M_{ij}'는 행동치 이므로 $\mathrm{rank}(M_{ij}) = \mathrm{rank}(M_{ij}') = m$이다.

3) $j=m$인 경우

$M_{ij} = I_m$이므로 $\mathrm{rank}(M_{ij}) = m$이다. □

표 3.1.1은 $m=4$일 때 정의 3.1.1의 H_2에 의해 얻어지는 해시 값을 나타낸다. 빈번하게 발생하는 패턴(행, 열, 주대각선, 반대각선, 직사각형)[3,4]에 대하여 정리 3.1.1에 제안된 해시 함수는 행, 열, 주대각선, 반대각선의 충돌회피를, 정리 3.1.2에 제안된 해시 함수는 직사각형의 충돌회피를 만족함을 알 수 있다.

		Column Index Matrix															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Row Index Matrix	0000	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	0001	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111
	0010	0100	0101	0110	0111	0000	0001	0010	0011	1100	1101	1110	1111	1000	1001	1010	1011
	0011	1100	1101	1110	1111	1000	1001	1010	1011	0100	0101	0110	0111	0000	0001	0010	0011
	0100	0110	0111	0100	0101	0010	0011	0000	0001	1110	1111	1100	1101	1010	1011	1000	1001
	0101	1110	1111	1100	1101	1010	1011	1000	1001	0110	0111	0100	0101	0010	0011	0000	0001
	0110	0010	0011	0000	0001	0110	0111	0100	0101	1010	1011	1000	1001	1110	1111	1100	1101
	0111	1010	1011	1000	1001	1110	1111	1100	1101	0010	0011	0000	0001	0110	0111	0100	0101
	1000	1101	1100	1111	1110	1001	1000	1011	1010	0101	0100	0111	0110	0001	0000	0011	0010
	1001	0101	0100	0111	0110	0001	0000	0011	0010	1101	1100	1111	1110	1001	1000	1011	1010
	1010	1001	1000	1011	1010	1101	1100	1111	1110	0001	0000	0011	0010	0101	0100	0111	0110
	1011	0001	0000	0011	0010	0101	0100	0111	0110	1001	1000	1011	1010	1101	1100	1111	1110
	1100	1011	1010	1001	1000	1111	1110	1101	1100	0011	0010	0001	0000	0111	0110	0101	0100
	1101	0011	0010	0001	0000	0111	0110	0101	0100	1011	1010	1001	1000	1111	1110	1101	1100
	1110	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000
	1111	0111	0110	0101	0100	0011	0010	0001	0000	1111	1110	1101	1100	1011	1010	1001	1000

<표 3.1.1> H_2에 의해 해시된 해시 값

2-way skewed-associative 캐시를 위한 효과적인 해시 함수는 그림 2.4.2의 뱅크 1에서 충돌이 발생하는 주소들에 대하여 뱅크 2에서 각각 다른 해시 값을 갖도록 고안되어야 한다. 이를 위해서 DID가 최대가 되어야 한다[5]. Vandierendonck 등은 XOR 기반의 두 해시 함수의 상한과 하한, 영공간, 열벡터의 개념을 사용하여 DID를 정의하였으나, 본 논문에서는 두 해시 함수 A, B의 첨가행렬 $[AB]$의 영공간의 차원이 0일 때 위의 성질을 만족한다는 사실을 이용하여 DID를 계수를 이용하여 새롭게 정의한다.

<정의 3.1.2> XOR 기반의 두 $2m \times m$ 해시 함수 A와 B의 DID를 첨가행렬 $[AB]$를 이용하여 다음과 같이 정의한다.

$$\mathrm{DID}(A, B) = \mathrm{rank}([AB])$$

정의 3.1.2로부터 $0 \le \mathrm{DID}(A,B) \le 2m$임을 쉽게 알 수 있다. 2-way skewed-associative 캐시에 대한 두 해시 함수를 구성할 때, 두 해시 함수의 DID가 최대가 되게 구성해야 그림 2.4.2에서와 같이 한 뱅크에서 충돌이 일어나더라도 다른 뱅크에서는 해시 값이 충돌 없이 분산되게 된다. 즉, 첨가행렬 $[A\,B]$의 계수가 $2m$이어야 하고. 이는 식 $n = \mathrm{rank}(H) + \dim N(H)$에 의하여 $[A\,B]$의 영공간의 차원이 0임을 의미한다.

<예제 3.1.1> $m=3$인 경우 두 선형 해시 함수가 다음과 같다고 하자.

$$A = \begin{pmatrix} 1\,1\,1 \\ 1\,1\,0 \\ 1\,0\,0 \\ 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{pmatrix}, \quad B = \begin{pmatrix} 0\,0\,1 \\ 0\,1\,1 \\ 1\,1\,1 \\ 1\,0\,0 \\ 0\,1\,0 \\ 0\,0\,1 \end{pmatrix}$$

행렬 $[A\,B]$의 계수가 5이므로 식 $n = \mathrm{rank}(H) + \dim N(H)$에서 $[A\,B]$의 영공간의 차원은 1이고, $N([A\,B]) = (111 : 101)$이다. 따라서 주어진 두 함수는 같은 해시 값을 갖는 주소들이 모두 2개씩이므로 충돌이 발생한다. 다음의 표 3.1.2은 각 주소들에 대한 두 함수의 해시 값을 나타낸다. 표 3.1.2에 의하면 $(110 : 010)$과 $(001 : 111)$은 해시 함수 A와 B에 의하여 같은 해시 값을 가지므로 충돌이 발생한다.

		Value of A							
		000	011	101	110	001	010	100	111
Value of B	000	000:000 111:101	110:010 001:111	101:110 010:011	011:100 100:001				
	011	001:100 110:001	000:011 111:110	011:111 100:010	010:000 101:101				
	101	010:110 101:011	011:001 110:110	000:101 111:000	001:010 110:111				
	110	011:010 100:111	010:101 101:000	001:001 110:100	000:110 111:011				
	001					000:001 111:100	001:110 110:011	010:010 101:111	011:101 010:000
	010					001:101 110:000	000:010 111:111	011:110 100:011	010:001 101:100
	100					010:111 101:010	011:000 100:110	000:100 111:001	110:110 001:011
	111					011:011 100:110	010:100 101:001	001:000 110:101	000:111 111:010

<표 3.1.2> A와 B의 해시 값

다음 정리들은 정의 3.1.1과 같이 구성한 2-way skewed-associative 캐시에 대한 해시 함수들이 m이 홀수인 경우에 DID가 최대임을 보인다.

<정리 3.1.3> 정의 3.1.1에 의해 정의된 T_1과 T_3에 대하여 $m=2k-1\ (k\geq 3)$일 때 다음이 성립한다.

$$\mathrm{rank}([T_1 \oplus T_3]) = m$$

(증명) $T_1 \oplus T_3 = (t_{ij})$, $t_{ij} = \begin{cases} 1,\ 1\leq i\leq k,\ 3\leq i+j\leq 2k+1 \\ 1,\ 1\leq i\leq m,\ i+j=2k \\ 1,\ k\leq i\leq m,\ 2m-(k-1)\leq i+j\leq 2m \\ 0,\ \text{그 외} \end{cases}$ 이므로,

이에 기본 행 연산을 시행하여 행동치인 블록 행렬 $\begin{pmatrix} I_{k-1} & A \\ O & I_k \end{pmatrix}$을 얻을 수 있다. 여기서 A는 영행렬이 아닌 $(k-1)\times k$ 행렬이다. 그러므로 다음이 성립한다. $\mathrm{rank}(T_1 \oplus T_3) = \mathrm{rank}([I_{k-1}\, A]) + \mathrm{rank}([O\, I_k]) = (k-1)+k = 2k-1 = m$이다. □

<정리 3.1.4> 정의 3.1.1에서 정의된 H_1과 H_3는 $m = 2k-1\ (k \geq 3)$일 때 다음이 성립한다.

$$\mathrm{rank}([H_1 H_3]) = 2m$$

(증명) 기본 행 연산을 시행하여 주어진 블록 행렬을 다음과 같이 유도할 수 있다.

$$[H_1 H_3] = \begin{pmatrix} T_1 & T_3 \\ I_m & I_m \end{pmatrix} \Rightarrow \begin{pmatrix} I_m & I_m \\ O & T_1 \oplus T_3 \end{pmatrix} \Rightarrow \begin{pmatrix} I_m & O \\ O & T_1 \oplus T_3 \end{pmatrix}$$

정리 3.1.3에 의하여 $\mathrm{rank}(T_1 \oplus T_3) = m$이므로 $\mathrm{rank}([H_1\, H_3]) = 2m$ 이다. □

위의 두 정리는 m이 짝수인 경우에도 성립한다.

<예제 3.1.2> 정의 3.1.1에 의해서 $m = 3$인 2-way skewed-associative 캐시에 대한 두 선형 해시 함수는 다음과 같다.

$$H_1 = \begin{pmatrix} 1&1&1 \\ 1&1&0 \\ 1&0&0 \\ 1&0&0 \\ 0&1&0 \\ 0&0&1 \end{pmatrix},\ H_3 = \begin{pmatrix} 1&0&1 \\ 0&1&1 \\ 1&1&1 \\ 1&0&0 \\ 0&1&0 \\ 0&0&1 \end{pmatrix}$$

여기서, $\mathrm{DID}(H_1, H_3) = \mathrm{rank}([H_1 H_3]) = 6$ 이므로 DID가 최대이다. 따라서 다음 표 3.1.3에서와 같이 2-way skewed-associative 캐시에 대하여 정의 3.1.1에서와 같이 구성한 H_1과 H_3는, DID가 최대가 되어 하나의 뱅크에서 8개의 주소가 충돌이 발생하더라도 또 다른 뱅크에서는 모두 흩어지게 해시하므로 충돌회피가 가능하다.

예를 들어, 6-비트 주소 (101:010), (000:001)와 (001:101)는 모두 그림 2.4.1의 뱅크 1에서는 표 3.1.3의 H_1에 의해서 (001)로 해시되어 충돌이 발생하지만, 그림 2.4.2의 뱅크 2에서는 H_3에 의해서 각각 서로 다른 해시 값 (000), (001)와 (010)를 가지므로 DID가 최대가 됨을 알 수 있다.

		Value of H_1							
		000	001	010	011	100	101	110	111
Value of H_3	000	000:000	101:010	100:101	001:111	111:001	010:011	011:100	110:110
	001	101:011	000:001	001:110	100:100	010:010	111:000	110:111	011:101
	010	100:111	001:101	000:010	101:000	011:110	110:100	111:011	010:001
	011	001:100	100:110	101:001	000:011	110:101	011:111	010:000	111:010
	100	111:101	010:111	011:000	110:010	000:100	101:110	100:001	001:011
	101	010:110	111:100	110:011	011:001	101:111	000:101	001:010	100:000
	110	011:010	110:000	111:111	010:101	100:011	001:001	000:110	101:100
	111	110:001	011:011	010:100	111:110	001:000	100:010	101:101	000:111

<표 3.1.3> H_1와 H_3의 해시 값

Ⅵ. 결론

본 논문에서는 2-way skewed-associative 캐시에 대한 해시 함수가 만족해야 하는 DID를 계수를 이용하여 새롭게 정의하였고, 이를 이용하여 DID가 최대인 XOR 기반의 새로운 해시 함수를 제안하였다. 이는 기존의 선형 해시 함수와는 다르게 주소 비트 수의 반이 짝수인 경우 뿐 아니라 홀수인 경우에 대해서도 충돌 회피가 가능하도록 각각 구성한 것이다. 특히 같은 크기의 멀티 뱅크 캐시에서 set-associative 캐시보다 더 나은 성능을 가진 skewed-associative 캐시에 대하여 충돌 회피가 가능하도록 구성하여 충돌회피성에서 훨씬 개선된 방법이다.

참 고 문 헌

[1] A. Seznec, A Case for Two-Way Skewed Associative Caches, Proc. 20th Ann. Int'l Symp. Computer Architecture, pp. 169-178, 1993.

[2] F. Bodin and A. Seznec, Skewed-Associativity Improves Program Performance and Enhances Predictability, IEEE Transactions on Computers, Vol. 46, pp. 530-544, 1997.

[3] J.M. Frailong, W. Jalby and J. Lenfant, XOR-Schemes: A Flexible Data Organization in Parallel Memories, Proc. 1985 Int'l Conf. Parallel Processing, pp. 276-283, 1985.

[4] H. Vandierendonck and K.D. Bosschere, XOR-Based Hash Functions, IEEE Transactions on Computers, Vol. 54, No. 7, pp. 800-812, 2005.

[5] H. Vandierendonck and K.D. Bosschere, On Null Spaces and their Application to Model Randomisation and Interleaving in Cache Memories, ELIS Technical Report DG 02-02, 2002.

[6] W. Stallings, Computer Organization and Architecture : Designing for Performance, 7th ed., Prentice Hall, 2005

[7] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd ed., The Johus Hopkins Univ. Press, 1996.

[8] R.A. Horn and C.R. Johnson, Matrix Analysis, The Cambridge Univ. Press, 1999.

[9] D.H. Lawrie, C.R. Vora, The prime memory system for array access, IEEE Transactions on Computers, Vol. 31, pp. 435-442,

1982.

[10] A.J. Smith, A Comparative Study of Set Associative Memory Mapping Algorithms and Their Use for Cache and Main Memory, IEEE Transactions Software Engineering, 1978.

[11] A. Agarwal, Analysis of Cache Performance for Operating Systems and Multiprogramming, Kluwer Academic Publishers, 1989.