요약 차이를 이용한 요약화일 동적 분산 기법
(A Dynamic Signature Declustering Method using Signature Difference)

강 형 일 † 강 승 헌 † 유 재 수 †† 임 병 모 †††
(Hyung Il Kang) (Seung Heon Kang) (Jae Soo Yoo) (Byoung Mo Im)

요 약 요약화일을 병렬로 처리하기 위해서는 효과적인 요약화일 분산 기법이 요구된다. Hamming Filter에서 분산 기법으로 이용되는 선형코드분산기법(LCDM)은 대부분의 경우 우수한 분산 성능을 갖지만 정적 특성 때문에 요약이 편중될 경우 요약화일을 균등하게 분산하기 어렵다. 또한 제한된 확장성과 비결정성(non-determinism)과 같은 문제점을 가지고 있다. 본 논문에서는 LCDM의 문제점을 해결하는 새로운 요약화일 분산 기법인 내적 기법(inner- product method)을 제안한다. 내적 기법은 요약의 내적에 의해 계산되는 요약 차이(signature difference)를 기반으로 하여 요약화일을 동적으로 분산한다. 다양한 데이타 작업부하에서 모의 실험을 통해 내적 기법이 LCDM보다 우수함을 보인다.

Abstract For processing signature file in parallel, an effective signature file declustering method is needed. The Linear Code Decomposition Method(LCDM) used for the Hamming Filter may give a good performance in some cases, but due to its static property, it fails to evenly decluster signature file when signature are skewed. In addition, it has other problems such as limited scalability and non-determinism. In this paper we propose a new signature file declustering method, called Inner-product method, which overcomes those problems in the LCDM. The Inner-product method declusters signature file dynamically based on the signature difference which is computed by using signature inner product. we show through the simulation experiment that the Inner-product outperforms the LCDM under various data workloads.

1. 서 론

정보 검색 및 관리는 오랫동안 컴퓨터 시스템의 주요 연구 분야로써 주로 정형화된 데이타(formatted data)를 처리 대상으로 하였다. 그러나 최근들어 사무자동화(office automation), 멀티미디어 데이타베이스 시스템(multimedia database system), 디자인 응용 프로그램(CAD/CAM, CASE) 그리고 진보된 정보 검색 시스템(advanced information retrieval systems) 등은 새로운 종류의 데이타 형태인 비정형화된 데이타(unformatted data)의 처리를 요구하고 있다. 따라서 정형화된 데이타 뿐만아니라 비정형화된 데이타를 신속하고 효율적으로 처리하기 위해서 보다 우수한 성능의 자료 저장 구조가 요구된다. 이와같은 다양한 형태의 자료를 효과적으로 처리하기 위한 자료 저장 구조 중의 하나가 요약화일(signature file) 방법이다[1].

요약(signature)은 자료 객체의 내용을 대표할 수 있는 키워드를 데이타베이스에 저장된 객체로부터 추출한 후 해쉬함수(hashing function)를 이용하여 구성한 비트열(bit string)이다. 요약들이 저장된 요약화일은 자료 검색시 실제 자료가 저장된 데이타베이스에 접근하기 전에 검색되어 질의를 만족할 가능성이 있는 문서를 미리 판별할 수 있도록 여과 기능(filtering effect)을 제공하는 다중키 자료 접근 방법이다[2]. 일반적으로 요약화

· 본 연구는 한국과학재단 특정기초 연구(과제번호: 1999-1-303-007-3)의 지원에 의해서 수행되었음.
† 비 회 원 : 충북대학교 정보통신공학과
khi@pretty.chungbuk.ac.kr
k1000@pretty.chungbuk.ac.kr
†† 종신회원 : 충북대학교 전기전자공학부 교수
yjs@cbucc.chungbuk.ac.kr
††† 비 회 원 : 한진정보통신 연구원
bmim@hist.co.kr
논문접수 : 1999년 5월 3일
심사완료 : 2000년 1월 21일

일의 요소인 각각의 문서 요약은 단어 요약을 중첩 코딩(superimposed coding)하여 만든다. 그림 1은 중첩코딩기법을 이용하여 "Database", "Parallel", "Information"의 세 단어로 구성된 문서의 문서요약을 구성하는 것을 보여준다. 여기서 요약의 길이는 12이고 한 단어 요약에 1로 설정된 비트의 수는 2이다.

Document D = (Database, Parallel, Information)

Keywords	Word Signature
Database	0110 0000 0000
Parallel	0000 1000 0001
Information	0001 0001 0000
Document Signature	0111 1001 0001

그림 1 중첩코딩기법을 이용한 문서 요약 구성

요약화일의 크기는 데이타 화일의 크기보다 훨씬 적기 때문에 주어진 질의에 대해 대부분의 부적합한 문서들을 즉시 제거하는 여과기로서의 역할을 효과적으로 수행할 수 있는 것으로 알려져 있다[2]. 작은 크기의 데이타 화일에 대해서는 요약화일을 순차적으로 구성하더라도 좋은 성능을 갖지만 데이타 화일이 클때는 문제가 된다. 이러한 요약화일은 트리나 해싱기법을 이용하여 그 성능을 향상시킬 수 있다. 정적 환경을 위한 요약화일 기법으로 Bit-Sliced Signature File[3]과 Frame-Sliced Signature File[4]이 제안되었고 반면에 동적 환경을 위한 요약화일 기법으로 S-tree[5], Quick Filter[6,7], HS File[8]이 제안되었다.

최초의 요약화일 자료 구조는 순차적 요약화일 구조였다. 그러나, 관리할 자료 규모가 증가함에 따라 요약화일 자체가 증가하기 때문에 요약화일 자체 검색 속도를 빠르게 하고 효율적으로 관리하기 위해 트리, 해싱 등 다양한 기법을 이용한 요약화일 저장 구조가 제안되었다. 그러나, 최근 관리할 자료의 규모가 초거대화 되어가고 있는 다양한 응용 환경을 위해서는 요약화일의 신속한 검색을 위한 병렬화가 불가피하다. 요약화일의 병렬로 처리하기 위한 많은 시도가 있었다. Fragmented Signature File(FSF)[9], Key-Based Partition Method[10], Hamming Filter[11]는 요약화일을 병렬로 처리를 위해 요약화일을 분할한다. 부분 만족 질의(partial match query)에 대해 좋은 분산 성능을 보이는 Hamming Filter는 데이타 전송시 에러 검출과 정정에 사용되는 선형코드분산기법(LCDM)을 이용하여 요약화일을 분산한다[12,13]. LCDM은 대부분의 경우에 좋은 분산 성능을 가져오지만 정적 특성 때문에 요약들이 편중되어 있을 때는 요약화일을 고르게 분산하기 어렵다. 또한 제한된 확장성과 비결정성(non-determinism)과 같은 문제점을 가지고 있다.

본 논문에서는 LCDM의 문제점을 해결하는 새로운 요약화일 분산 기법인 내적 기법(inner- product method)을 제안한다. 내적 기법은 요약 내적(signature inner-product)을 이용하여 요약화일을 분산한다. 두 요약간의 요약 내적은 각 요약을 비트 스트링 벡터로 간주한 스칼라 값이다. 이 기법에서는 새로운 요약과 각 노드의 대표 요약간의 차이 정도를 비교한다. 즉, 새로운 요약과 각 노드의 대표 요약과의 내적들 중 최소값을 가진 노드에 새로운 요약을 할당한다. LCDM은 현재의 요약 배치 상태를 반영하지 못하는 정적 정보를 기반으로 한다. 반면에 내적 기법은 현재 요약 배치 상태를 기반으로 동적으로 요약화일을 분산하다. 그러므로 내적 기법은 다양한 작업부하와 환경구성(configuration)들에 대처할 수 있다.

본 논문에서는 통계적 모델링을 기반으로한 성능평가를 통해 균등분포, 정규분포, 지수분포와 같은 다양한 분포의 데이타 집합에 대해 내적 기법이 LCDM보다 우수한 검색 성능을 내타냄을 보인다. 그리고 점근적 표기법(asymptotic notation)을 사용한 요약 삽입 시간을 제시하고 이를 통해 삽입이 빈번하게 발생하는 동적 환경에서 내적 기법이 우수한 삽입 성능을 나타냄을 보인다.

본 논문은 다음과 같이 구성된다. 2절에서는 다양한 병렬 요약화일 기법들을 살펴본다. 3절에서는 내적 기법을 제안하고 4절에서는 실험결과를 보인다. 마지막으로 5절에서는 결론을 내린다.

2. 병렬처리를 위한 요약화일

대규모 트랜잭션 처리 시스템, 의사 결정 시스템, 멀티미디어 시스템 등 관리할 자료의 규모가 증가함에 따라 자료 검색 성능 향상을 위한 자료 저장 구조 병렬화가 지속적으로 요구되고 있다. 이러한 데이타베이스 시스템을 위한 병렬 처리기 구조는 자원들의 공유 정도에 따라 다음과 같이 세가지로 분류될 수 있다[14]. 먼저, 디스크와 주기억장치 모두를 모든 처리기가 공유하는 SE(Shared Everything) 병렬처리 구조 그리고 모든 처리기가 디스크 만을 공유하는 SD(Shared Disk) 병렬처리 구조, 마지막으로 처리기들이 특정 자원을 공유하지 않고 네트워크를 통한 메시지 전달만을 통하여 병

렬 처리를 하는 SN(Shared Nothing) 병렬처리 구조가 있다. 실제 데이타베이스 구현에 있어서 어느 병렬 구조가 가장 적합한지에 관한 치열한 논쟁이 있어왔지만, 일반적으로 SE와 SD는 병렬 처리할 자료를 분산할 필요가 없기 때문에 자료 처리 알고리즘이 단순하다는 장점을 안고 있지만, 값이 비싸고 공유하는 자원에서 병목현상(bottleneck)이 발생하기 때문에 처리기의 개수가 작은 병렬 시스템에 적합한 것으로 알려져 있다. 하지만, SN은 메모리, 디스크, 처리기를 독립적으로 갖는 병렬 노드를 네트워크를 통해 연결하는 모듈 디자인 시스템이기 때문에 확장성이 뛰어나며, 값이 저렴하다. 따라서 대규모 데이타베이스를 위한 병렬 처리 시스템으로 적합하다. 그러나 병렬 처리할 데이타를 모든 노드에 자료부하 및 수행부하 관점에서 균등하게 분산시키는 효과적인 분산기법을 필요로 한다. 이때 '자료부하'란 각각의 병렬 노드에 분산 되는 데이타의 개수를 의미하며, '수행부하'란 각각의 병렬 노드에서 임의의 질의에 만족되는 데이타의 개수를 의미한다. 일반적으로 모든 질의에 대한 수행부하 균등 분산 기법은 불가능하기 때문에 다양한 휴어리스틱[hueristic] 알고리즘에 관한 연구가 진행중이다[15,16]. 참고로, 본 논문에서 제안하는 요약화일 분산 기법도 요약화일 수행부하의 균등한 분산을 위한 휴어리스틱 알고리즘이다.

병렬환경에서 동작하는 요약화일을 만들기 위해 Fragmented Signature File(FSF), Key-Based Partition Method(KBPM), Hamming Filter[9,10,11]과 같은 많은 시도가 있어왔다. 이들은 요약들을 병렬처리에 참여하는 노드의 디스크로 분산시켜 요약화일의 검색속도를 향상시킨다.

표 1 병렬 요약화일 분류

Categories	Partitioning Methods	Declustering Methods
Parallel Signature File	Key-Based Partition Method	Hamming Filter
	Fragmented Signature File	

본 논문에서는 요약화일의 디스크에 분배 기법에 따라 병렬 요약화일 구성을 두 가지 범주로 분류한다. 하나는 요약화일 분할 기법(partitioning method)이고 다른 하나는 요약화일 분산 기법(declustering method)이다. 표 1은 두 가지 범주를 나타낸다. 지금부터 자료 분배 방법에 초점을 맞춰 병렬 요약화일을 살펴보고 분석한다.

2.1 분할 기법(Partitioning Method)

본 절에서는 요약화일을 부요약화일(sub-signature file) 집합이나 요약 프레임(signature frame) 집합에 분할하는 요약화일 분할 기법을 설명한다. 부요약화일은 요약화일을 수평으로 분할했을 때 요약화일의 분할(partition)이고 요약 프레임은 요약화일을 수직으로 분할했을 때 요약화일의 분할이다. 일반적으로 각 부요약화일의 요약들은 공통키(common key)를 가지고 있다. M개의 프로세서와 N개의 분할이 주어졌을 때, 만약 M과 N이 같다면 각 분할은 바로 각 처리 노드로 할당된다. 그러나 N이 M보다 크다면 추가적인 분할 기법들이 요구된다. [9]에서 세 개의 Key-Based Partition Method가 제안되었다. 이들 Key-Based Partition Method는 요약화일을 수평으로 분할한다. 이들은 동일한 요약키를 갖는 요약들을 같은 분할로 나누고 같은 노드에 할당한다. 그러나 이들은 키를 선택하는 방법에서 차이가 있다. (1)Fixed Prefix Method는 고정된 위치에서 고정길이의 키를 선택한다. (2)Extended Prefix Method는 고정된 위치에서 가변길이의 키를 선택한다. (3)Floating Key Method는 가변적인 위치에서 고정길이의 키를 선택한다. 위의 세 Key-Based Partition Method는 질의-내(intra-query) 병렬화를 위해 이용될 수도 있지만 질의-간(inter-query) 병렬화에 더 적합하다.

FSF는 혼합된 분할 방법이다. 수직 분할에 의해 요약화일을 요약화일 프레임들로 분할한다. 또 한편으로 수평 분할을 통해 요약화일을 요약 서브화일(signature sub-file)로 분할한다. 이때 각 요약 서브화일내의 모든 요약은 공통 요약키를 갖는다.

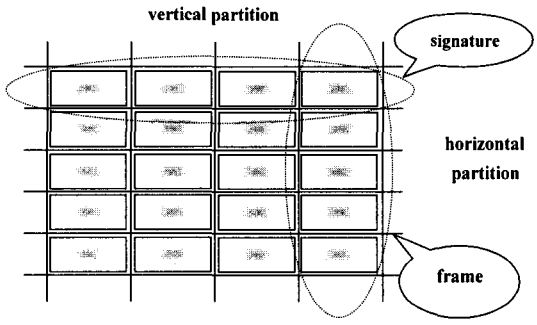

그림 2 혼합된 분할기법

그림 2는 혼합된 분할 기법을 나타낸다. 이렇게 수직과 수평으로 분리된 자료 분할들은 주어진 처리 노드에

할당된다. 이 방법은 질의-내 병렬화를 허용하지만 질의-간 병렬화에 더 적합하다. 왜냐하면 임의의 요약질의를 만족하는 요약들의 특정 분할에만 위치하기 때문이다. 따라서 임의의 요약질의에 대해서 특정 분할만이 검색되어진다. 따라서 다른 분할들은 다른 요약질의를 검색에 사용될 수 있다.

2.2 분산 기법(Declustering Method)

분할 기법들은 단순한 수평적, 수직적 또는 수평과 수직이 혼합된 요약화일 분할방법을 이용한다. 따라서 특정 질의에 대해 비활성 분할들이 존재하게 되는데 이러한 비활성 분할들은 질의-간 병렬화를 통해 다른 질의 검색에 사용될 수 있다. 분산 기법은 질의-내 병렬화를 최대화하여 수행편중(execution skew)을 없애는 것을 주된 목적으로 한다. 가장 단순한 분산기법은 무작위로 요약을 처리 노드들로 분산하는 것이다. Hamming Filter는 선형코드를 이용하여 요약화일을 분산하는 선형코드분산기법(LCDM)에 의해 수평으로 요약화일을 분산한다. 그리고 LCDM에 의해 각 지역 노드로 분산된 요약들은 Quick Filter를 사용하여 처리된다. 그러므로 Hamming Filter는 선형코드분산 원리를 이용하는 Quick Filter의 확장으로 여겨질 수 있다.

LCDM은 선형코드의 신드롬(syndrome) 특성을 이용한다[15]. 그림 3은 LCDM이 선형공간 $\{0,1\}^n$을 각각 동일한 신드롬을 갖는 $2^{(n-k)}C(n,k)$로 분산하는 것을 보여준다. 동일한 신드롬을 갖는 코드단어(codeword)는 편중되지 않는 데이타에 대해서는 균등하게 분산된다. 이것은 코드단어간의 Hamming distance가 보장되기 때문이다. Hamming Filter는 LCDM을 각 요약의 접미사에 적용함으로써 요약화일을 분산한다.

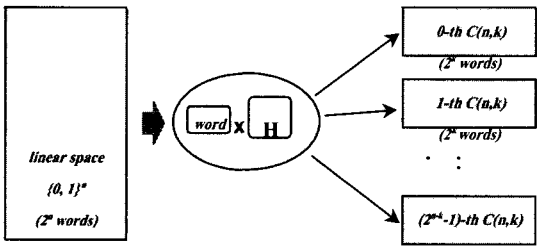

그림 3 선형코드분산 기법

질의-내 병렬화를 위해서는 자료편중(data skew)과 수행편중(execution skew)을 피할 수 있는 효과적인 분산 알고리즘이 요구된다. LCDM은 편중되지 않는 자료에 대해서는 수행부하도 편중되지 않는다. 그러나 LCDM은 동일한 접미사를 갖는 요약을 같은 노드에 할당하기 때문에 만약 많은 요약들이 동일한 접미사를 갖는다면 자료 편중을 피할 수 없다. 또한 LCDM에서는 다음과 같은 문제점들이 병렬화를 어렵게 한다. 첫째, 처리 노드 개수의 확장이 어렵다. 이 방법은 처리 노드의 수가 2^n개일 때만 정의된다. 이것은 실제 대부분의 병렬 시스템들이 2^n개만큼의 처리 노드를 갖지 않을 수 있기 때문에 심각한 제약이 된다. 둘째, 비결정적이다. 크기가 m인 접미사에 대해, LCDM이 요약화일을 분산하기 위해 m!개의 검정 행렬(check matrix)이 사용될 수 있다. 요약화일 분산에 이용될 검정 행렬의 선택이 LCDM의 분산 성능에 영향을 미칠 수 있지만, 최적의 검정 행렬 선택에 대한 적절한 정책이 제시된바 없다. 마지막으로, LCDM은 요약의 부분적인 정보 즉, 요약의 접미부분만을 사용함으로써 정보의 손실을 초래할 수도 있다.

3. 병렬 요약화일을 위한 새로운 분산 기법

본 절에서는 새로운 요약화일 분산 기법인 내적 기법(Inner-product)을 제안한다. 대용량 병렬 요약화일을 구성하기 위한 기반 플랫폼으로는 SN 병렬 구조로 가정한다. SN 병렬 구조에서 요약 질의 처리 시간을 최소화하기 위해, 데이타는 균등하고 독립적으로 모든 처리 노드에 분산될 필요가 있다. SN 병렬 환경에서 요약 질의 응답시간을 최소화하기 위해 질의-내 병렬화는 필수적이다. p개의 병렬 처리기를 가진 SN 병렬 시스템에서 한 요약 질의에 대한 응답시간은 $\mathbf{max}\{C_1, C_2, \ldots, C_p\}$으로 정의된다. 여기서 $C_i(1 \le i \le p)$는 i번째 처리 유닛의 응답시간, 즉 물리적 디스크 접근 회수이다. C_i는 그림 3.1에서와 같이 i번째 처리 유닛에서 요약 검색 시간과 false drop 처리 시간을 더한 것이다. 그러므로 목표는 가능한 모든 요약화일 분산기법들 중에서 임의의 질의에 대해 $\mathbf{max}\{C_1, C_2, \ldots, C_p\}$이 최소가 되는 분산 기법을 찾는 것이다. 이러한 분산문제는 NP-complete이다 [15,16].

제안하는 내적기법은 질의-내(intra-query) 병렬화를 최대화하여 수행편중을 없애기 위한 방법이기 때문에 질의요약을 처리하기 위해 모든 노드에서 질의를 처리한다. 각 노드에서의 질의 처리는 기존의 순차 요약화일 처리방법[1]과 같다. 예를 들어 그림 4와 같이 3개의 노드에 9개 문서들을 대표하는 요약들이 제안하는 동적 분산기법인 내적 기법을 통하여 분산되어 있다고 가정하자. 질의요약(query signature) 010000100001의 응답시간은 각 노드에서 순차적으로 질의요약을 비트별로 포함하는 문서 요약을 선택한다. 노드 1에서는 D1, 노드

3에서는 D8, D9의 요약화일이 선택되고 질의요약 010000100001의 응답시간은 실제 false drop 처리시간을 포함하며 검색된 요약의 수가 2개인 노드 3에 의존한다.

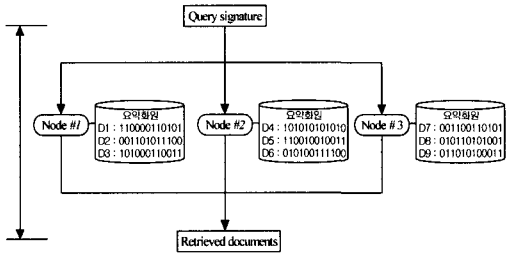

그림 4 요약 질의의 응답 시간 및 질의처리 예

요약 질의 응답시간을 최소화하기 위한 기본 기준은 너무 많은 데이타가 특정 노드에 치우치는 자료편중과 수행 시간이 특정 노드에서 과다하게 걸리는 수행편중을 피하는 것이다. 본 논문에서 제안하는 방법을 기술하기 전에 몇 가지 표기법(notation)과 정의(definition)를 소개한다.

3.1 정의 및 용어

표기법

- n, 요약 크기
- m_i, i번째 노드의 요약 개수
- $c_{ij} = \sum s_{ijk}$, $(1 \le k \le m_i)$ i번째 노드에서 요약의 j번째 수직합(vertical sum)
- s_{ijk}, i번째 노드에서 k번째 요약의 j번째 비트

정의 3.1 (completely different)

만약 두 요약의 모든 원소가 다르면 두 요약은 완전히 다르다.

정의 3.2 집계벡터(count vector)

$cv_i = <c_{i1}, c_{i2}, \ldots, c_{ij}, \ldots, c_{in}>$로 표현되는 i번째 노드의 집계벡터(count vector)는 i번째 노드에 할당된 요약의 수직 합 벡터이다. 집계벡터의 구성 절차는 그림 5와 같다.

$$
\begin{array}{rl}
 & 1\,0\,0\,1\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1 = s_{i1} \\
 & 1\,0\,1\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0 = s_{i2} \\
+ & 0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,0\,0\,1\,0\,1 = s_{i3} \\
\hline
 & <2,0,2,1,1,2,1,2,3,2,2,1,0,1,0,2> = cv_i
\end{array}
$$

그림 5 i번째 노드의 집계벡터

정의 3.3 단위요약(unit signature)

$uv_i = <u_{i1}, u_{i2}, \ldots, u_{ij}, \ldots, u_{in}>$로 표현되는 i번째 노드의 단위요약(unit signature)은 i번째 노드의 요약 할당 상태를 나타낸다. 이것은 0과 1로 정규화된 집계벡터이다. 예를 들어, c_{i1}이 cv_i의 평균보다 크면 u_{i1}은 '1'이고 그렇지 않으면 u_{i1}은 '0'이다.

정의 3.4 두 요약의 내적(inner product of two signature)

두 요약 s_i, s_i의 내적 $s_i \cdot s_i$은 두 요약간 원소쌍들의 곱을 합한 값이다. 예를 들어 s_i = (1001010111000001), s_j = (1010001111110000) 이면 $s_i \cdot s_i$ = 1*1 + 0*0 + 0*1 + 1*0 + 0*0 + 1*0 + 0*1 + 1*1 + 1*1 + 1*1 + 0*1 + 0*1 + 0*0 + 0*0 + 0*0 + 1*0 = 5 이다.

보조정리 3.1

동일한 두 요약의 내적은 요약의 구성원소의 합이다.

$$s_i \cdot s_i = \sum s_i$$

증명)

$s_i = (s_{i1}, s_{i2}, \ldots, s_{in})$이면, $s_i \cdot s_i = s_{i1}*s_{i1} + s_{i2}*s_{i2} + \ldots + s_{in}*s_{in} = s_{i1} + s_{i2} + \ldots + s_{in} = \sum s_i$이다. 왜냐하면 s_{ij}는 이진수이므로 $0 \le j \le n$인 모든 j에 대해 $s_{ij} * s_{ij} = s_{ij}$ 이기 때문이다.

보조정리 3.2

완전히 다른 두 요약의 내적은 0이다.

$$s_i \cdot s_j = 0$$

증명)

$s_i = (s_{i1}, s_{i2}, \ldots, s_{in})$, $s_j = (s_{j1}, s_{j2}, \ldots, s_{jn})$이면, $s_i \cdot s_j = s_{i1}*s_{j1} + s_{i2}*s_{j2} + \ldots + s_{in}*s_{jn} = 0 + 0 + \ldots + 0 = 0$ 이다. s_i와 s_j는 완전히 다르므로 s_{ik}, s_{jk} 중 하나는 항상 0이다. 그러므로 $0 \le k \le n$인 모든 k에 대해 $s_{ik} * s_{jk} = 0$이다.

정리 3.1

주어진 두 요약 s_i, s_j에 대해 $0 \le s_i \cdot s_j \le \mathbf{min}\{\sum s_i, \sum s_j\}$ 이다.

증명)

다음 3가지 경우가 위의 정리를 만족함을 보여 위의 정리를 증명한다.

1) s_i와 s_j가 같은 경우

보조정리 3.1에 의해 $s_i \cdot s_j = \sum s_i$이다. 그러므로 이것은 위의 정리를 만족한다.

2) s_i와 s_j가 완전히 다른 경우

보조정리 3.2에 의해 $s_i \cdot s_i = 0$이다. 그러므로 이것

은 위의 정리를 만족한다.

3) s_i와 s_j가 다른 경우

i) $0 \le s_i \cdot s_j$: 내적의 합은 0보다 크거나 같기 때문에 위의 정리를 만족한다.

ii) $s_i \cdot s_j \le \mathbf{min}\{\sum s_i, \sum s_j\}$: $s_i = (s_{i1}, s_{i2}, \ldots, s_{in})$, $s_j = (s_{j1}, s_{j2}, \ldots, s_{jn})$이라 하면 $s_i \cdot s_j = s_{i1}*s_{j1} + s_{i2}*s_{j2} + \ldots + s_{in}*s_{jn}$이다. 만약 s_i에서 1로 설정된 비트 집합 수가 s_j의 그것보다 작다면 $s_i \cdot s_j$는 s_i의 1로 설정된 비트의 수를 넘지 않는다. 반대로 만약 s_j에서 1로 설정된 비트 집합 수가 s_i의 그것보다 작다면 $s_i \cdot s_j$는 s_j의 1로 설정된 비트의 수를 넘지 않는다. 그러므로 이것은 위의 정리를 만족한다.

정리 3.2

만약 s_i와 s_j간 1의 위치 차이가 s_i와 s_k간 1의 위치 차이보다 크다면 $s_i \cdot s_j < s_i \cdot s_k$이다.

증명)

두 요약의 내적은 단지 두 요약이 동일한 위치에 1을 갖는 원소의 수가 많을 때 증가한다. s_i과 s_j가 s_i과 s_k보다 동일 위치의 1이 더 적기 때문에 $s_i \cdot s_j < s_i \cdot s_k$이다.

3.2 기본 개념

병렬 환경에서 요약 질의의 응답시간은 수행부하(execution load)가 모든 처리노드에 고르게 분산되었을 때 최소화될 수 있다. 여기서 수행부하는 요약에서 1로 설정된 비트의 수와 밀접하게 관련되어있다. 왜냐하면 요약 질의를 처리할 때 0으로 설정된 비트는 비교할 필요가 없기 때문이다. 그러므로 요약화일은 1로 설정된 비트를 기반으로 분산되어져야 한다. i번째 노드의 집계벡터 cv_i는 정의 3.2에서 정의된 것처럼 i번째 노드에 이미 할당된 요약의 수직 합이다. 그러므로 cv_i는 i번째 노드의 잠재적인 수행부하 분포를 개략적으로 나타낸다. 제안된 방법의 기본 개념은 각 처리노드의 집계벡터 구성원소들의 차이를 최소화하는 것이다. 이것으로 인해 질의를 만족할 가능성이 있는 요약들을 모든 처리 노드에 균등하게 분산할 수 있다. 본 논문에서는 집계벡터의 차이를 최소화하는 새로운 발견적 방법(heuristic method)인 내적기법을 제안한다. 내적기법은 정규화된 집계벡터인 단위요약(unit signature)을 사용한다.

3.3 내적기법(The Inner-Product Method)

본 절에서는 제안된 요약화일 분산기법인 내적기법에 대해 기술한다. 내적기법은 각 요약 할당 단계에서 최소 내적을 갖는 노드를 찾는 욕심장이 기법(greedy method)의 일종이다. 이것은 노드의 단위요약과 새로 할당되는 요약간의 요약내적을 이용하여 요약화일을 분산한다. 비트 스트링 벡터인 두 요약의 요약내적은 정의 3.4에 정의된 것처럼 스칼라 값이다. 집계벡터의 차이를 최소화하기 위해 1의 위치가 다른 요약들을 동일한 처리 노드에 할당한다. 정리 3.2에 의해 요약내적을 이용하여 요약간 차이정도를 비교할 수 있다. 내적기법에서는 이러한 원리를 이용하여 새 요약과 각 노드의 대표 요약인 단위요약간의 차이정도를 비교한다. 정규화된 집계벡터인 단위요약은 정의 3.3에 정의된 것처럼 각 처리 노드의 요약 할당 상태를 나타낸다. 내적기법은 새로운 요약과 각 노드의 단위요약과의 내적들 중 최소의 내적을 갖는 처리노드에 새로운 요약을 할당한다. 내적기법은 각 처리노드에 대해 3가지 종류의 데이타를 사용한다. 이것은 각각 집계벡터(count vector), 요약집계(signature count), 단위요약(unit signature)이다. 요약집계는 각 노드에 할당된 요약의 수이다. 제안된 내적기법의 상세한 절차는 네가지 단계로 구성된다.

- 초기화 단계(Initialization Phase)

처리 노드의 수가 p일 때 초기 상태는 다음과 같다. 여기서 sc_i, cv_i, uv_i는 각각 i번째 노드에서 요약집계, 집계벡터, 단위요약을 나타낸다.

- $sc_i = 0, (1 \le i \le p)$
- $cv_i = 0, (1 \le i \le p)$
- $uv_i = 0, (1 \le i \le p)$

- 전 계산 단계(Pre-calculation Phase)

이 단계에서는 요약 s_j를 할당하기 전에 모든 처리 노드의 내적을 계산한다. 수식은 다음과 같다.

- $uv_i{+}{+} = uv_i \cdot s_j, (1 \le i \le p, s_j$는 할당될 요약$)$

- 노드선택 단계(Node Selection Phase)

모든 노드의 내적이 계산된 후, 최소 $uv_i{+}{+}$를 갖는 처리노드 하나를 선택한다. 즉, 새로운 요약이 할당될 p_k노드는 $p_k=\mathbf{min}\{uv_i{+}{+}\}, (1 \le i \le p)$로 정의된다. 만약 두개 이상의 노드가 같은 $uv_i{+}{+}$을 갖는다면 요약집계가 낮은 노드를 선택한다. 요약집계 또한 같다면 무작위로 하나의 노드를 선택한다. 그러나 이러한 상황은 거의 발생하지 않는다.

- 노드할당 단계(Node Allocation Phase)

p_k를 노드선택 단계에 의해서 선택된 노드라고 하자. 요약 s_j을 노드 p_k에 할당한다. s_j를 할당한 후 p_k의 요약집계를 1증가시키고 p_k의 집계벡터를 변경시킨다. 그리고 p_k의 단위요약을 다음과 같이 재구성한다.

- $sc_k = sc_k + 1$
- $cv_k = cv_k + s_j$
- 만약 $cv_k > \text{mean}(cv_k)$ 이면 $uv_k = 1$ 그렇지 않으면 $uv_k = 0$

내적기법의 상세한 알고리즘은 다음과 같다.

알고리즘

입력 :
- $s\_j[n]$: 분산될 요약으로 비트 스트링의 1차원 배열. 여기서 n은 요약의 크기

결과 :
- p : 새로운 요약이 할당될 처리노드의 번호

변수 :
- PN : 처리노드의 총 개수
- $cv[PN][n]$: 집계벡터로 정수형의 2차원 배열
- $uv[PN][n]$: 단위요약으로 '0'아니면 '1'의 값을 갖는 2차원 배열
- $uv\_plus[PN]$: 임시 단위요약으로 '0'아니면 '1'의 값을 갖는 2차원 배열
- $sc[PN]$: 요약집계로 정수형의 1차원 배열

처리 :

```
/* 초기화 */
/* static은 C언어에서와 같은 의미 */
static sc [i] = 0, ( 0  i < PN )
static cv[i][j] = 0, ( 0  i < PN ,  0  j < n)
static uv [i] [j]= 0, ( 0  i < PN ,  0  j < n)
for( i = 0; i < PN; i++ ) {
     /* uv++ 구성 */
    uv_plus[i] = uv[i] · s_j;
}
/* MIN 함수는 최소값을 갖는 uv_plus의 인덱스를 리턴한다. 만
   약 둘 이상의 uv_plus가 동일한 값을 갖는다면 그들중 최소의
   요약집계를 갖는 노드의 인덱스를 리턴한다. 만약 요약집계 또
   한 같을 경우 무작위로 선택된 인덱스를 리턴한다. */
p = MIN(uv_plus[i]);
/* 요약 s_j를 p번째 처리노드에 할당한다. */
for( i = 0; i < n; i++) {
    cv[p][i] = cv[p][i] + s_j[i]
}
/*  p번째 처리노드의 요약집계를 1증가한다. */
sc[p] = sc[p] + 1;
/* p번째 처리노드의 단위요약을 재구성한다. mean 함수는 집계벡
   터의 평균을 리턴한다. */
for( i = 0; i < n; i++ ) {
    if ( cv[p][i] > mean(cvp) )
        uv[p][i] = 1;
    else
        uv[p][i] = 0;
}
return(p);                  /* 결과를 리턴한다. */
```

예제

cv_i와 cv_j를 각각 i번째 노드와 j번째 노드의 집계벡터이고 이들은 cv_i = <4, 5, 4, 7, 4, 6, 4, 6, 3, 5, 5, 7>, cv_j = <5, 4, 7, 4, 6, 4, 6, 4, 5, 3, 7, 5>라고 하자. 이때 요약의 크기 n은 12이다. 그러면 단위요약 uv_i과 uv_j은 각각 <0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1>과 <0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0>이 된다. 새로 할당해야할 요약 s_k가 <1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0>이라면 $s_k \cdot uv_i$는 0이고 $s_k \cdot uv_j$는 4이다. 그러므로 내적기법은 s_k를 i번째 노드에 할당한다. 왜냐하면 i번째 노드가 더 작은 내적을 가지고 있기 때문이다.

4. 성능평가

4.1 실험

본 절에서는 검색 시간과 삽입 시간에 의해 내적기법의 성능을 평가한다. 실험은 128Mbyte의 메인메모리를 갖춘 Sparc-20 워크스테이션에서 10,000와 100,000개의 문서에 대하여 수행하였다. 각 문서는 저자, 제목 그리고 18개의 키워드, 즉 20개의 필드로 구성된다. 입력 및 설계 매개변수는 표 2와 같다. 실험은 [18]에서 사용된 것과 동일한 합성 데이타를 기반으로 하였다. 균등, 정규, 지수 분포 데이타 집합의 3가지 합성 데이타 형태를 생성하였다. 자료 편중은 균등분포에서 지수분포로 갈수록 증가한다.

표 2 매개변수

매개변수	설 명	값
N	문서 개수	10, 100K
p	처리노드 개수	8,16
D	문서당 단어 개수	20
n	요약 크기(비트)	512
m	단어당 '1'로 설정된 비트의 수	16

4.1.1 요약화일 분산도

첫 번째 실험에서는 제안하는 방법의 요약화일 분산의 성능을 평가하기 위해 3가지 방법(내적기법, LCDM, 무작위 분산기법)에 대한 요약 검색의 응답 시간을 비교하였다. 요약 검색의 응답 시간은 처리노드에서의 요약 검색 시간과 false drop 처리시간을 합한 것이다. 질의-내 병렬화에서 응답 시간은 검색된 요약의 수가 최대인 처리노드에 의존하기 때문에 검색된 요약의 최대 개수를 각 질의에 대한 응답 시간의 척도(measure)로 사용하였다. 하나의 검색된 요약은 최소한 한 블럭의 디스크 접근을 야기한다. 왜냐하면 false drop 처리는 실제 문서와의 매칭 처리를 기반으로 하기 때문이다. i-질의에 대한 검색된 요약의 최대 개수는 $\mathbf{max}(R_{i1}, R_{i2}, ..., R_{ip})$이다. 여기에서 $R_{ij}(1 \le j \le p)$는 j번째 처리노드에서 i-질의에 대해 검색된 요약의 개수이다.

본 실험에서는 8개, 16개의 처리노드에 각각 10,000개, 100,000개의 문서를 가진 데이타 화일을 분산한다. 실제 운영중인 많은 정보검색 시스템들은 실험에서 사용된 데이타 집합보다 훨씬 많은 수백만개의 문서를 가지고 있다. 본 실험은 제한된 실험 환경 때문에 비교적 작은 실험 데이타 집합을 사용하지만 제안된 기법의 동작 특성은 실험 데이타의 크기와는 상관이 없다. 실험 집합에 대해 10개의 표본 질의를 임의로 선택하였다. 그리고 많은 검색 결과가 나오도록 하기 위해 단일어(single term) 질의어를 사용하였다. 이것은 병렬 처리 효과를 확실하게 비교할 수 있도록 하기 위함이다. 이후의 모든 그림에서는 3가지 분산 기법의 분산 효과를 비교한다. 또한 제안된 기법의 특성을 쉽게 이해할 수 있도록 하기 위해 한 개의 그래프를 추가하였는데 이 그래프는 최적으로 분산한 경우를 나타낸다. 즉, 각 질의에 대해 검색된 최대 요약 개수의 이론적 최소값으로 이것은 분산의 하한(lower bound)이다. 예를 들어 n이 검색된 요약의 총 개수이고 m이 처리노드의 개수일 때, 주어진 질의어 q에 대해 검색된 최대 요약 개수의 이론적 최소값은 $\lceil n/m \rceil$ 이다.

그림 6과 그림 7은 균등분포 데이타 집합에 대해 제안된 요약화일 분산기법의 성능을 보여준다. 내적기법과 LCDM은 무작위 분산기법보다 우수한 성능을 보인다. 그리고 내적기법은 대부분의 질의에서 LCDM보다 우수한 성능을 보인다. 그림 7은 그림 6과 비슷한 결과를 보인다. 그러므로 제안된 방법은 일반성의 손실 없이 확장될 수 있다고 말할 수 있다.

그림 8과 그림 9는 정규분포 데이타 집합에 대한 제안된 요약화일 분산 기법의 성능을 보여준다. 이 두 그림에서는 균일분포 데이타 집합에 대한 실험결과와 동일한 성능 패턴을 보인다.

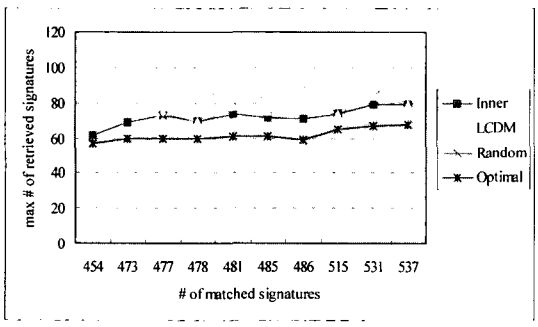

그림 6 병렬 노드 8개인 환경에서 10,000개 균등분포 문서의 분산도

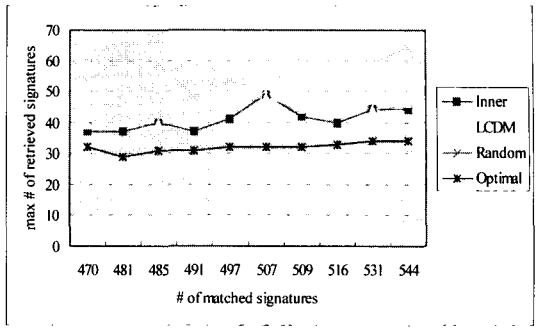

그림 7 병렬 노드 16개인 환경에서 100,000개 균등분포 문서의 분산도

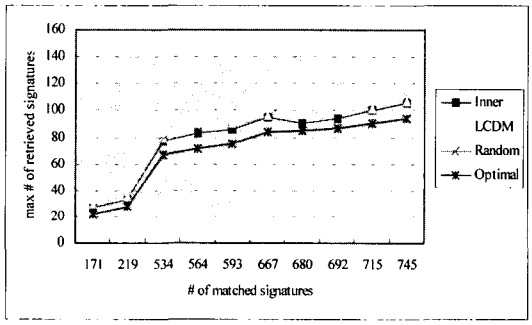

그림 8 병렬 노드 8개인 환경에서 10,000개 정규분포 문서의 분산도

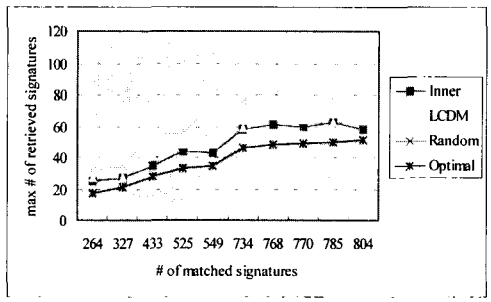

그림 9 병렬 노드 16개인 환경에서 10,0000개 정규분포 문서의 분산도

마지막으로 지수분포 데이타 집합에 대한 성능은 그림 10과 그림 11에 나타나 있다. 이 두 그래프 또한 동일한 성능 패턴을 보여준다. 그러나 대용량 데이타 집합

에 대해서는 무작위 분산기법은 내적기법에 비해 그 성능이 현저하게 떨어진다. 그 이유는 일치하는 요약의 수가 급증함에 따라 내적기법은 동적으로 축적된 정보를 이용함으로써 안정을 유지하기 때문이다.

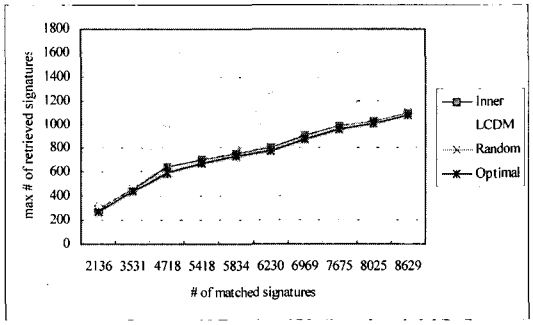

그림 10 병렬 노드 8개인 환경에서 10,0000개 지수분포 문서의 분산도

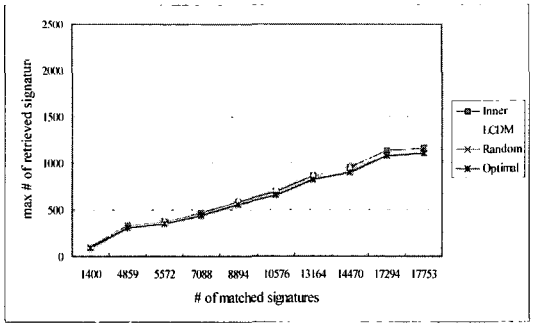

그림 11 병렬 노드 16개인 환경에서 10,0000개 지수분포 문서의 분산도

비록 대부분의 질의에 대해 LCDM이 무작위 분산기법보다 우수하지만 어떤 질의에 대해서는 무작위 분산기법과 유사하거나 떨어지는 성능을 보인다. 이것은 대량으로 일치된 요약은 동일한 접미사를 가지고 있고 그러한 요약들은 동일한 처리노드에 할당되기 때문이다. 그러나 내적기법은 그러한 문제점을 가지고 있지 않다. 왜냐하면 이전에 할당된 요약들의 통계적 정보를 기반으로 요약화일을 동적으로 분산하기 때문이다. 결과적으로 다양한 형태의 작업부하 분포에 대하여 내적기법은 LCDM과 무작위 분산기법보다 우수한 검색 성능을 갖는다. 또한 내적기법은 높은 확장성과 LCDM이 갖지 못한 결정성을 가지고 있다.

4.1.2 삽입시간

두 번째 실험에서는 점근적 표기법(asymptotic notation)의 한 종류인 *Big "Oh"*(O())를 이용하여 요약 분산 기법들의 삽입시간을 비교한다. 병렬 요약의 삽입시간은 다음과 같이 두 가지 항목으로 구성된다.

$$I_{time} = D_{time} + O_{time}$$

여기에서 I_{time}, D_{time}, O_{time}는 각각 삽입시간, 분산시간, 지역구성시간(local organization time)이다. 그러나 본 실험에서는 D_{time}에 대해서만 고려한다. 왜냐하면 O_{time}은 모든 요약화일 분산기법에 대해 동일하기 때문이다. 세 가지 분산 기법들의 계산 시간은 다음과 같이 변수 n과 p를 이용한 $O()$ 표기법으로 나타낸다. 여기에서 n은 요약의 크기, p는 처리노드의 개수이다.

- 무작위 분산기법 : p와 n에 독립적이므로 $C_1O(1)$
- LCDM = 행렬 곱 계산 시간 = $C_2O(p^2)$
- 내적기법 = uv^{++} 구성시간 + 최소 uv^{++} 검색 시간 + cv 증가 시간
  $= C_7O(pn) + C_8O(pn) + C_9O(p) + C_{10}O(pn) + C_{11}O(n) = O(pn)$

그러므로 $p<<n$일 경우 LCDM은 내적기법보다 우수하지만 $p>>n$일 경우 내적기법은 LCDM기법보다 우수하다. 결론적으로 내적기법은 삽입이 빈번하게 발생하는 동적인 대규모 병렬 데이타베이스 시스템 환경에 적합하다.

5. 결 론

본 논문에서는 Hamming Filter에서 분산 기법으로 사용되는 LCDM의 문제점을 기술하였다. LCDM은 병렬화를 어렵게 하는 문제점을 가지고 있다. 첫째, 처리노드 수의 확장이 어렵다. 이 기법은 2^n개의 처리노드를 갖는 경우만을 정의한다. 이것은 실제 대부분의 병렬 시스템들이 2^n개만큼의 처리 노드를 갖지 않을 수 있기 때문에 심각한 제약이 된다. 둘째, 비결정적이다. m크기의 접미사에 대해, LCDM이 요약화일을 분산하기 위해 m!개의 검정 행렬(check matrix)이 사용될 수 있다. 검정 행렬의 선택이 LCDM의 분산 성능에 영향을 미칠 수 있지만, 최적의 검정 행렬 선택에 대한 적절한 정책이 제시된바 없다. 마지막으로 부분적인 정보 즉, 요약의 접미부분만을 사용함으로 LCDM은 정보의 손실을 초래할 수도 있다.

본 논문에서는 LCDM의 문제점을 해결하는 새로운 요약화일 분산 기법인 내적기법(inner- product method)을 제안하였다. 이 기법은 현재의 요약 할당 상

태를 기반으로 하여 동적으로 요약화일을 분산한다. 그러므로 내적기법은 작업부하와 환경의 다양성을 극복할 수 있다. 본 논문에서는 통계적 모델링을 기반으로한 성능평가를 통하여 균등분포, 정규분포, 지수분포와 같은 다양한 분포의 데이타 집합에 대해 내적기법이 LCDM보다 우수한 요약화일 분산을 수행함을 알 수 있었다. 그리고 점근적 표기법(asymptotic notation)을 사용한 요약 삽입 시간 성능을 제시하고 이를 통해 삽입이 빈번하게 발생하는 동적 환경에서 내적 기법이 우수한 성능을 나타냄을 보였다. 앞으로의 연구과제는 병렬 환경에서 검색 성능향상을 위해 각 노드에서의 효율적인 검색방법 및 요약화일의 구조에 대한 연구를 하는 것이다.

참 고 문 헌

[1] C. Faloutsos. Signature-based text retrieval methods.*A Survey. IEEE Computer Society Technical Committee on Data Engineering*, 13(1): 25--32, 1990.

[2] C. Faloutsos and S. Christodoulakis. Design of a signature file method that accounts for non-uniform occurrence and query frequencies. In *Proc. of the 11th VLDB Conf.*, pages 165--170, Stockholm, Sweden, August 1985.

[3] C. S. Roberts. Partial match retrieval via the method of the superimposed codes. In *Proc. of IEEE 67*, pages 1624--1642, Dec. 1979.

[4] Z. Lin and C. Faloutsos. Frame-sliced signature files. *IEEE Transaction on Knowledege and Data Engineering*, 4(3):281--289, June 1992.

[5] U. Deppisch. S-tree: A dynamic balanced signature index for office retrieval. In *ACM SIGIR*, pages 77--87, 1986.

[6] F. Rabitti and P. Zezula. A dynamic signature technique for multimedia database. In *Proc. of the 13th ACM SIGIR*, pages 193--210, Brussels, Belgium, September 1990.

[7] Ciaccia P. Zezula, P. and P. Tieberio. Hamming filter: A dynamic signature file organization for parallel stores. In *Proc. of the 19th VLDB Conf.*, pages 314--327, Dublin, Ireland, 1993.

[8] Kim M. H. Lee Y. J. Yoo, J. S. and J. W. Chang. The hs file : A new dynamic signature file method for efficient information retrieval. In *Int'l Conf. On Database and Expert Systems Applications*, Athenes, Greece, Sept. 1994.

[9] Tiberio P. Grandi, F. and P. Zezula. Frame-sliced partitioned parallel signature files. In *Proc. of 15th Ann. Int'l SIGIR*, pages 286--297, Denmark, June 1992.

[10] D.L. Lee and C. Leng. A partitioned signature file structure for multiattribute and text retrieval. In *Proc. of the 6th Int'l Conf. On Data Engineering*, pages 389--397, Los Angeles, California, Feb. 1990.

[11] Rabitti F. Zezula, P. and P. Tiberio. Dynamic partitioning of signature files. *ACM TOIS*), 9(4):336--369, October 1991.

[12] C.Faloutsos and D.Metaxas. Declustering using error correcting codes. In *Proc. of 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 253--258, Philadelphia, Pennsylvania, March 1989.

[13] F.M. Reza. *An Introduction to Information Theory* McGraw-Hill, 1961.

[14] M. Mehta and D. J. DeWitt. Managing intra-operator parallelism in parallel database systems. In *Proc. of the 21th VLDB Conf.*, pages 382--394, Zurich, Swizerland, 1995.

[15] M. H. Kim and S. Pramanik. Optimizing database accesses for parallel processing of multikey range searches. *The Computer Juornal*, 35(1):45--51, 1992.

[16] Y. Y. Sung. Performance analysis of disk modulo allocation method for cartesian product files. *IEEE Transactions on Software Engineering SE-13(9)*, pages 1018--1026, 1987.

강 형 일

1996년 목포대학교 전산통계학과(이학사). 1998년 목포대학교 전산통계학과(이학석사). 1998년 ~ 현재 충북대학교 정보통신공학과 박사과정 재학 중. 관심분야는 데이타베이스 시스템, XML, 정보검색

강 승 헌

1997년 목포대학교 전산통계학과 졸업(학사). 1999년 충북대학교 정보통신공학과 졸업(공학석사). 1999년 ~ 현재 충북대학교 정보통신공학과 박사과정 재학 중. 관심분야는 데이타베이스시스템, 정보검색, XML

유 재 수

1989년 전북대학교 공과대학 컴퓨터공학과(학사). 1991년 한국과학기술원 전산학과(공학석사). 1995년 한국과학기술원 전산학과(공학박사). 1995년 ~ 1996년 목포대학교 전산통계학과 전임강사. 1996년 ~ 현재 충북대학교 공과대학 전기전자공학부 조교수. 관심분야는 데이타베이스 시스템, 정보검색, 멀티미디어 데이타베이스, 분산 객체 컴퓨팅

임 병 모

1991년 연세대학교 이과대학 전산과학과(학사). 1993년 한국과학기술원 전산학과(공학석사). 1999년 한국과학기술원 전산학과(공학박사). 1999년 ~ 현재 한진정보통신 선임연구원. 관심분야는 병렬 데이타베이스, OLTP, 분산처리 시스템, 정보검색