# 대용량 데이터의 내용 기반 검색을 위한 분산 고차원 색인 구조

# (A Distributed High Dimensional Indexing Structure for Content-based Retrieval of Large Scale Data)

최 현 화 † 이 미 영 †† 김 영 창 † 장 재 우 ††† 이 규 철 ††††

(Hyun-Hwa Choi) (Mi-Young Lee) (Young-Chang Kim) (Jae-Woo Chang) (Kyu-Chul Lee)

**요 약** 고차원 데이터에 대한 다양한 색인 구조가 제안되어 왔음에도 불구하고, 인터넷 서비스로서 이미지 및 동영상의 내용 기반 검색을 지원하기 위해서는 고확장성 지원 및 k-최근접점 검색 성능 향상을 지원하는 새로운 고차원 데이터의 색인 구조가 절실히 요구된다. 이에 우리는 다중 컴퓨팅 노드를 바탕으로 구축되는 분산 색인 구조로 분산 벡터 근사 트리(Distributed Vector Approximation-tree)를 제안한다. 분산 벡터 근사 트리는 대용량의 고차원 데이터로부터 추출한 샘플 데이터를 바탕으로 hybrid spill-tree를 구축하고, hybrid spill-tree의 말단 노드 각각에 분산 컴퓨팅 노드를 매핑하여 VA-file을 구축하는 두 레벨의 분산 색인 구조이다. 우리는 다중 컴퓨팅 노드들 상에 구축된 분산 벡터 근사 트리를 바탕으로 병렬 k-최근접점 검색을 수행함으로써 검색 성능을 향상시킨다. 본 논문에서는 서로 다른 분포의 데이터 집합을 바탕으로 한 성능 시험 결과를 통하여, 분산 벡터 근사 트리가 기존의 고확장성을 지원하는 색인 구조와 비교하여 검색 정확도에 대한 손실 없이 더 빠른 k-최근접점 검색을 수행함을 보인다.

키워드 : 고차원 데이터, 분산 색인 구조, k-최근접점 검색, 내용 기반 검색

*Abstract* Although conventional index structures provide various nearest-neighbor search algorithms for high-dimensional data, there are additional requirements to increase search performances as well as to support index scalability for large scale data. To support these requirements, we propose a distributed high-dimensional indexing structure based on cluster systems, called a Distributed Vector Approximation-tree (DVA-tree), which is a two-level structure consisting of a hybrid spill-tree and VA-files. We also describe the algorithms used for constructing the DVA-tree over multiple machines and performing distributed k-nearest neighbors (NN) searches. To evaluate the performance of the DVA-tree, we conduct an experimental study using both real and synthetic datasets. The results show that our proposed method contributes to significant performance advantages over existing index structures on difference kinds of datasets.

Key words : High Dimensional Data, Distributed Indexing Structure, k-NN Search, Content-based Retrieval

· 본 연구는 정보통신부 및 정보통신연구진흥원의 IT 신성장동력핵심기술개발사업의 일환으로 수행하였음{2007-S-016-1, 저비용 대규모 글로벌 인터넷 서비스 솔루션}

† 비 회 원 : 한국전자통신연구원 데이터베이스연구팀 선임연구원
hyunwha@etri.re.kr
zerowin@etri.re.kr

†† 종신회원 : 한국전자통신연구원 데이터베이스연구팀 팀장
mylee@etri.re.kr

††† 종신회원 : 전북대학교 컴퓨터시스템공학과 교수
jwchang@chonbuk.ac.kr

†††† 종신회원 : 충남대학교 컴퓨터공학과 교수
kclee@cnu.ac.kr
(Corresponding author임)

논문접수 : 2010년 8월 23일
심사완료 : 2010년 8월 26일



정보과학회논문지: 데이타베이스 제37권 제5호(2010.10)

## 1. 서 론

최근, 대용량 데이터 관리를 위한 색인 구조에 있어서 고확장성 지원은 클라우드 컴퓨팅 서비스 도래와 더불어 산업계는 물론 학계에서 중요한 이슈 중의 하나이다. 특히, CAD, 지리학, 분자학 및 생물학 등의 분야에서는 대용량의 고차원 데이터에 대한 빠른 내용 기반 검색을 필요로 하고 있다. 여기서 고차원 데이터는 객체의 모양, 색깔 및 질감과 같은 중요한 특징을 고차원의 벡터 공간의 점으로 변환한 것으로 소위 특징 벡터라 불린다. 한편, 고차원 데이터에 대한 내용 기반 검색은 유사성 검색(similarity search)을 바탕으로 이뤄진다. 유사성 검색은 특징 벡터 공간 상의 하나의 점이 주어지면 그 질의 점에 가까운 점들을 찾아내는 것을 말한다. 기본적으로 유사성 검색은 범위 검색(ε-range search)과 k-최근접점 검색(k-nearest neighbors search, k-nn search) 두가지 타입이 있다. 범위 검색은 사용자로부터 질의 점 p와 임의의 거리 값 ε을 입력으로 받아, 질의 점 p로부터의 거리가 ε와 같거나 혹은 작은 모든 점을 찾아내는 것이다. 반면, k-최근접점 검색은 사용자로부터 질의 점 p와 찾고자 하는 최근접점 개수 k를 입력 받아, 질의 점 p로부터 k개의 가장 가까운 점들을 찾아낸다.

대용량의 고차원 데이터에 대한 유사성 검색을 지원하기 위하여, 많은 분산 색인 구조가 제안되어 왔다. 그러나 제안된 색인 구조는 P2P(Peer-to-Peer) 환경 [1-5]에서 운영됨을 가정하거나 범위 질의를 지원하는 방법에 대한 연구가 대부분이다. 클라우드 컴퓨팅 서비스 혹은 인터넷 서비스로서 대용량의 고차원 데이터에 대한 내용 기반 검색을 지원하기 위해서는 클러스터 환경에서 동작하는 분산 색인 구조를 효과적으로 관리하는 방법과 이를 바탕으로 빠른 k-최근접점 검색을 수행하는 방법에 대한 연구가 필요하다. 유사성 검색에서 k- 최근접점 검색이 선호되는 이유는 범위 검색은 객체 간 거리 값과 같은 사용자 입력 정보가 데이터의 사전 지식을 바탕으로 이뤄졌을 때 효율적인 반면, k-최근접점 검색은 사용자가 원하는 가장 유사한 객체의 개수 지정으로 검색이 효과적으로 이뤄질 수 있기 때문이다.

클러스터 환경에서 대용량의 고차원 데이터를 바탕으로 k-최근접점 검색을 효율적으로 수행하기 위하여, 색인 구조가 가져야 할 특징은 아래와 같다.

- 색인 구조는 클러스터 환경 하의 다중 컴퓨팅 노드들을 바탕으로 구축될 수 있어야 한다.
- 고차원 특징 벡터 공간 상의 객체들은 대체적으로 균일하지 않은 분포를 가지는 것이 특징이다. 그러므로, 색인 정보를 저장한 각 노드들은 되도록 같은 수의 객체들을 담당하도록 관리되어야 한다.
- 대용량 고차원 데이터에 대한 빠른 k-최근접점 검색을 지원하기 위한 병렬 알고리즘을 지원할 수 있는 구조이어야 한다.
- 다중의 독립적인 질의에 대한 효과적인 병렬 수행을 지원하기 위하여, 연관되거나 혹은 유사한 객체에 관한 색인 정보는 되도록 적은 수의 노드들에서 관리되어야 한다.

본 논문에서 우리는 클러스터 환경 하에서 운영되는 분산 벡터 근사 트리(Distributed Vector Approximation-tree)라는 고차원 색인 구조를 제안한다. 분산 벡터 근사 트리는 기본적으로 검색 공간을 분할하는 이진 균형 트리(binary balanced tree) 구조를 가지며, 트리의 말단 노드마다 분산된 컴퓨팅 노드를 매핑한다. 매핑된 분산 컴퓨팅 노드 각각은 지역 색인 구조로써 VA-file [6]을 관리한다. 즉, 우리는 검색 성능 향상 및 색인 구조의 확장성 지원을 위하여, 대용량 고차원 데이터에 대한 VA-file를 효율적으로 분산시키는 방법을 제안한다.

또한, 우리는 분할 기반 접근의 트리와 필터링 기반 접근인 VA-file를 접목한 새로운 분산 색인 구조를 바탕으로 분산 k-최근접점 질의를 처리하는 방법을 설명하고, 균일한 분포의 데이터와 편향된 분포를 가지는 데이터를 바탕으로 k-최근접점 검색 시험을 수행한다.

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 관련 연구를 검토하고 3장에서는 hybrid spill-tree와 VA-file에 관한 배경 지식을 설명한다. 4장에서는 다중 노드를 바탕으로 분산 벡터 근사 트리를 구축하는 방법과 이를 바탕으로 병렬 k-최근접점 검색을 수행하는 방법에 대해서 자세히 기술한다. 5장에서는 분산 벡터 근사 트리에 대한 성능 시험 및 그 결과에 대해서 설명한다. 그리고 마지막으로 6장에서 결론을 맺는다.

## 2. 관련 연구

지금까지 고차원 데이터에 대한 k-최근접점 문제를 해결하기 위한 많은 연구가 있어 왔다. 먼저 트리 구조를 바탕으로 데이터를 관리하는 분할 기반의 접근은 크게 공간 분할 기법과 데이터 분할 기법으로 분류할 수 있다. 공간 분할 기법은 데이터의 분포와 상관없이 미리 정의된 초평면(hyper planes)에 따라 공간을 분할해 나가는 것으로 K-D-B tree[7]와 hB-tree[8]가 대표적이다. 반면, 데이터의 분포에 따라 데이터 공간의 중복을 허용하여 분할하는 데이터 분할 기법에는 R*-tree[9], X-tree[10], M-tree[11], spill-tree와 hybrid spill-tree [12] 등이 있다. 이러한 분할 기반 접근은 데이터의 차원이 증가할수록 검색 성능이 기하급수적으로 감소하는 문제점이 있다[13].

그리하여, 전체 데이터를 순차적으로 스캔하는 필터링

기반 접근이 제기되었으며, VA-file[6]이 그것이다. VA-file은 실수의 특징 벡터를 비트 문자열로 압축한 근사 정보를 바탕으로 순차 스캔의 시간을 줄여 검색 성능을 향상시키는 방법을 제안하였다. VA-file를 바탕으로 검색 성능을 향상시키고자 하는 다양한 연구는 꾸준히 진행 되고 있다[14,15].

한편, 빠른 검색을 위하여 정확도에 대한 요구사항을 낮출 수 있는 응용이 발생하면서, 데이터를 해싱하여 관리하는 해쉬 기반 접근이 제안되었다[16,17]. 그러나 해쉬 기반 접근은 편향된 분포를 가지는 데이터의 경우 검색 정확도가 많이 떨어지고, 해쉬 테이블 구축 시 최적의 성능을 위한 파라미터의 값 선정이 어렵다는 단점이 있다.

단일 컴퓨팅 노드에서 수행됨을 가정한 상기의 색인 방법들을 바탕으로 대용량의 데이터 검색을 위한 분산 색인 구조 연구가 최근 많이 이뤄지고 있다. 먼저, Mayank et al.[4]과 Parisa et al.[5]은 P2P 환경에서 LSH(Local Sensitive Hashing)을 바탕으로 근사 k-최근접점 검색을 제공하는 방법을 제안하였다. 그러나, 해쉬 함수는 결정된 저장소에 충분한 근접점이 없을 경우 이를 확장할 방법이 없기에 k-최근접점 검색에는 적합지 않다. SkipIndex[1]는 데이터 공간 분할의 k-d-tree를 바탕으로 데이터 공간을 skip graph에 매핑하는데 있어 데이터 값을 하나의 키로 압축하여 사용한다. Dist[2]와 VBI-tree[3] 역시 트리를 바탕으로 한 접근 방법이나 고차원 데이터로 잘 확장되지 못한다는 문제점이 있다.

한편 Ting et al.[18]은 클러스터 환경 기반 분산 색인 구조를 제안하였다. 이는 hybrid spill-tree를 다중 컴퓨팅 노드들에 분산하는 방법으로 특징 벡터의 샘플을 기반으로 상위 트리(top-tree)를 구성하고, 상위 트리의 말단 노드별로 hybrid spill-tree를 구축한 분산 컴퓨팅 노드를 매핑하여 다중 컴퓨팅 노드들에 걸친 하나의 hybrid spill-tree를 구축한다. 트리의 분할 기반 접근이 고차원 데이터 검색에 있어 필터링 기반 접근보다 성능이 좋지 않음을 감안하면, 인터넷 서비스 혹은 클라우드 컴퓨팅 서비스 지원을 위하여 검색 성능을 향상시키는 새로운 분산 색인 구조 연구가 필요하다 하겠다.

## 3. 배경 지식

### 3.1 Hybrid spill-tree

Hybrid spill-tree[12]는 빠른 검색 성능을 가지는 spill-tree와 높은 검색 정확도를 가지는 M-tree를 결합한 트리 구조이다. Hybrid spill-tree는 그림 1에서 보는 것과 같이 자식 노드로서 공유 데이터 공간을 가지는 spill-tree 노드 혹은 공유 공간을 가지지 않는 M-tree 노드를 가질 수 있다. 먼저, 부모 노드 내 가장 먼 거리를 가지는 두 점의 이등분 점을 지나는 수직 평면에 의해 두 자식 노드를 위한 데이터 공간이 분할된다. 이때 하나의 자식 노드에 속하는 점들의 개수가 부모 노드에 속한 전체 점 개수의 특정 임계치 비율보다 많은 경우 부모 노드는 비공유 노드, 즉 M-tree 노드로 결정된다. 반대의 경우에 부모 노드는 공유 노드인 spill-tree 노드가 된다. 보통 임계치로는 70%가 사용된다.

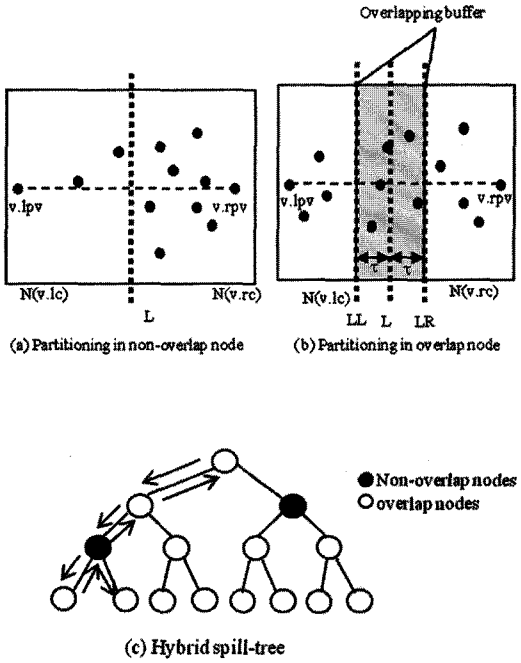



그림 1 Hybrid spill-tree 구조

Hybrid spill-tree에서 k-최근접점 검색은 비공유 노드에서 역추적(back-tracking)을 수행하는 MT-DFS (M-tree Depth First Search) 방식을, 공유 노드에서는 역추적을 수행하지 않는 defeatist search 방식을 사용하여 수행된다.

### 3.2 VA-file

VA-file[6]은 데이터의 각 차원 별로 데이터 공간을 여러 구간으로 나누고, 나누어진 각 영역에 비트(bit)를 할당하여 근사값을 생성한다. 즉, VA-file에서 각 특징 벡터는 비트 문자열의 근사값이 가리키는 특정 하나의 셀을 가리키게 된다. 예를 들어, 2차원의 데이터 공간에서 특징 벡터와 그것의 근사값을 가지는 VA-file은 그림 2와 같다.

특징 벡터의 근사값인 비트 문자열의 수는 데이터 차원 수와 각 차원 별로 할당된 비트의 개수에 의해 결정되며, 이들의 관계는 그림 3과 같다.

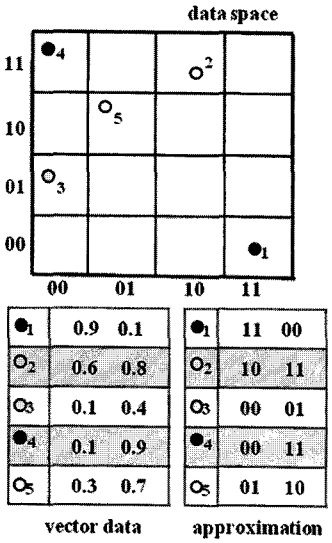



그림 2 VA-file 구조

$$n = \sum_{i=1}^{d} b_i$$

그림 3 근사값의 비트 수(n), 특징 벡터 차원 수(d), 차원별 할당 비트 수($b_i$)의 관계

## 4. 분산 벡터 근사 트리

k-최근접점 질의 처리 성능은 색인 데이터의 크기가 커질수록 감소한다. 그리하여, 한 컴퓨팅 노드에서 감당할 수 없을 만큼 색인 데이터가 많은 경우 이를 여러 노드에 분산시켜 병렬 처리하는 것이 검색 성능을 향상시키는데 있어 중요하다. 본 장에서는 글로벌 파일 시스템을 바탕으로 한 클러스터 환경 하에서 근사 k-최근접점 검색을 빠르게 수행하기 위한 색인 구조와 검색 알고리즘을 소개한다.

### 4.1 분산 벡터 근사 트리 구축

분산 벡터 근사 트리는 클러스터 환경에서 대용량의 VA-file을 여러 컴퓨팅 노드들에 효과적으로 분산시키고자 하는 목적에서 고안되었다. VA-file을 효과적으로 분산시키기 위하여, 특징 벡터들에 대한 클러스터링(clustering)을 수행한다. 클러스터링 기법으로 기존 고차원 데이터의 색인 구조 중의 하나인 hybrid spill-tree를 사용한 분산 벡터 근사 트리 구조의 모습은 그림 4와 같다. 우리는 특징 벡터의 클러스터 정보를 바탕으로 k-최근접 검색 시에 연관성이 적은 특징 벡터 클러스터들의 접근을 생략할 수 있다.

분산 벡터 근사 트리는 그림 4에서 보는 것과 같이 두 레벨의 색인 구조를 가진다. 먼저 상위 레벨의 글로

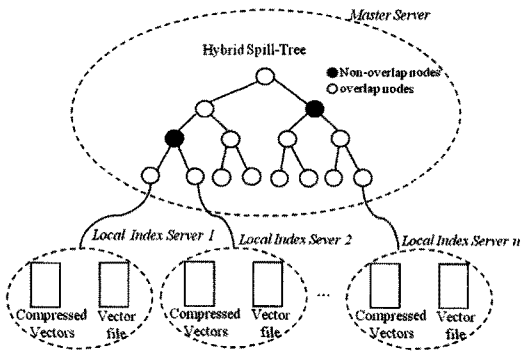



그림 4 분산 벡터 근사 트리 구조

벌 색인인 hybrid spill-tree는 한 노드에 구축되며 마스터 서버에 의해 관리된다. 한편, hybrid spill-tree의 각 말단 노드에 매핑된 분산 지역 색인 서버 각각은 지역 색인으로 VA-file을 구축한다. 사용자의 내용 기반 검색을 위한 유사성 검색은 먼저 검색할 VA-file들을 결정하기 위하여, 사용자 질의 특징 벡터를 바탕으로 글로벌 색인을 탐색한다. 글로벌 색인의 말단 노드 각각은 매핑된 분산 지역 색인 서버의 정보 및 해당 서버에서 관리하는 특징 벡터들에 대한 범위 정보를 포함한다. 글로벌 색인 탐색을 통해 결정된 분산 지역 색인 서버들에게 질의 특징 벡터가 전달되고, 각 분산 지역 색인 서버들은 자신이 관리하는 VA-file을 바탕으로 k-최근접점 검색을 수행한다.

분산 벡터 근사 트리를 구축하기 위하여, 먼저 대용량의 특징 벡터로부터 하나의 컴퓨팅 노드에서 감당할 수 있을 정도의 특징 벡터 집합을 구성한다. 이때 전체 특징 벡터의 클러스터 정보를 잃어버리지 않기 위하여, 우리는 랜덤 샘플링 기법을 사용한다. 랜덤 샘플링은 단순할 뿐만 아니라 데이터의 차원과 독립적으로 수행되며, 균일하거나 혹은 균일하지 않은 분포를 가지는 데이터를 모델링하는 데 가장 많이 사용되는 기법이다. 이렇게 랜덤 샘플링 기법을 통해 선정된 특징 벡터의 샘플 데이터를 바탕으로 hybrid spill-tree를 메모리 상에 구축한다. 한 컴퓨팅 노드에서 감당할 수 있는 정도의 샘플 특징 벡터를 바탕으로 트리를 구축하는 이유는 트리 구조가 여러 분산 컴퓨팅 노드에서 운영하기 어려울 뿐만 아니라, 병렬적으로 트리 탐색을 수행하기도 어렵기 때문이다. 한편, Hybrid spill-tree가 특징 벡터의 서브 집합을 바탕으로 구축되기 때문에 트리의 말단 노드의 페이지 크기에 대한 조절이 필요하다. 즉, 전체 데이터의 1/10 크기의 샘플링 데이터를 바탕으로 hybrid spill-tree를 구축하였다면, 트리의 말단 노드의 페이지 크기는 원래 크기보다 1/10 크기로 감소되어야 함을 의미한다.

샘플링 기반 hybrid spill-tree가 전체 데이터 기반 hybrid spill-tree와 얼마나 비슷한가를 나타내는 트리의 정확도는 데이터의 샘플링 비율에 의존한다. 샘플 데이터가 크면 클수록 클러스터 정밀도에 있어서 트리의 정확도는 증가한다. 그러나 샘플 데이터의 크기는 hybrid spill-tree의 말단 노드의 수에도 영향을 미치기 때문에, 주어진 특징 벡터 집합에서 적절한 샘플 데이터의 크기를 결정하는 것은 쉬운 일이 아니다. 그러나 우리는 Yamane[19]에 의해 제안된 그림 5의 공식에 따라 주어진 데이터를 바탕으로 최소한의 적절한 샘플 크기를 예측할 수 있다.

$$n \geq \frac{N}{N * e^2 + 1}$$

그림 5 데이터 모집단 크기(N), 표본 오차(e), 샘플 데이터 크기(n)와의 관계

결론적으로, 우리는 한 컴퓨팅 노드의 메모리 크기를 초과하지 않는 범위 내에서, 상기 그림 5의 공식으로부터 계산된 최소한의 샘플 크기를 만족하거나 큰 샘플 데이터를 사용한다.

우리는 [11]과 [12]에서 기술된 표준화된 트리 구축 알고리즘을 사용하여 메모리 상에 샘플 데이터를 바탕으로 hybrid spill-tree를 구축한다. 그리고, 구축된 hybrid spill-tree의 각 말단 노드가 가지는 특정 파티션(partition)에 해당하는 특징 벡터를 저장할 하나의 분산 지역 색인 서버를 매핑시킨다. 매핑이 완료되면, 전체 특징 벡터들을 hybrid spill-tree 탐색에 따라 결정되는 분산 지역 색인 서버에 저장한다. 이때 각 분산 지역 색인 서버들은 입력된 특징 벡터를 파일에 저장하고, 특징 벡터의 근사값을 생성하여 메모리에 저장한다. 여기서, 분산 지역 색인들의 메모리 크기는 상위 트리인 hybrid spill-tree의 말단 노드의 페이지 크기 결정을 위한 하나의 요소로 고려될 수 있다. 우리는 기본적으로 분산 벡터 근사 트리 구축 후 새로운 데이터의 삽입을 감안하여, 구축된 hybrid spill-tree의 말단 노드와 매핑된 분산 지역 색인 서버에 저장될 특징 벡터 근사값의 크기보다 1/2 혹은 그보다 작은 값을 페이지 크기로 선정한다. 한편, 분산 벡터 근사 트리 구축 시간을 단축하기 위하여, 전체 특징 벡터의 저장을 병렬적으로 수행하여 VA-file이 다중 분산 지역 색인 서버 상에 빠르게 구축될 수 있도록 한다. 지금까지 설명한 구축 알고리즘을 요약하면 그림 6과 같다.

### 4.2 K-최근접점 검색

전체 분산 벡터 근사 트리는 여러 컴퓨팅 노드들에 확장된 하나의 hybrid spill-tree라 할 수 있다. 분산 벡

```
Algorithm M_buildTree (dataSet)
Input: the high dimensional dataset
Output: DVA-tree

1. sample = sampling(dataSet);
2. root = buildHybridSpillTree(sample);
3. bindTreeLeafNodesWithServers(root);
4. rangeList = calculateRangeForParallelInsertion(dataSize, p);
5. call classifyData(root, dataSet, rangeList)
   in p-parallel executions;
6. call L_insertVectorData(classifiedDataSet)
   of all the bound local index servers in parallel;

classifyData(root, dataSet, rangeList)
1. dataRange = getDataRange (dataSet, rangeList);
2. for each vector data v ∈ dataRange
3.   serverList = searchHybridSpillTree(root, v);
4.   for local index server s ∈ serverList
5.     store v in a vector file of s;

Alogrithm L_insertVectorData (classifiedDataSet)
Input: the high dimensional vector data set
Output: vector approximation set
1.  va = getVectorApproximation(classifiedDataSet);
2.  addVA (VAListInMemory, va);
```

그림 6 분산 벡터 근사 트리 구축 알고리즘

터 근사 트리의 노드는 3 종류의 노드로 구성된다. 먼저, 한 컴퓨팅 노드에 구축된 hybrid spill-tree의 내부 노드인 라우팅 노드, hybrid spill-tree의 말단 노드인 서버 라우팅 노드, 마지막으로 말단 노드에 매핑된 분산 컴퓨팅 노드를 가리키는 데이터 노드가 그것이다. 데이터 노드는 특징 벡터와 이에 해당하는 근사값을 관리하여, VA-file을 기반으로 k-최근접점 질의를 처리한다.

분산 벡터 근사 트리를 기반으로 k-최근접점 질의 처리는 그림 7과 같이 3단계에 걸쳐 수행된다.

먼저, 사용자로부터 k-최근접점 검색 요청이 들어오면, 질의 특징 벡터를 바탕으로 마스터 서버의 hybrid spill-tree를 탐색하여, VA-file기반 검색을 수행할 분산 지역 색인 서버를 결정한다. 여기서, hybrid spill-tree 탐색은 k-최근접점 질의를 범위 질의로 변경 수행함으

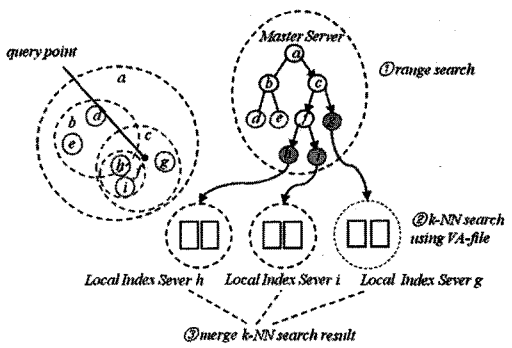



그림 7 k-최근접점 질의 처리

로써 비공유 노드 상에서 발생하는 역추적을 제거하여 탐색 시간을 단축한다. 특히, k-최근접점 검색을 범위 질의로 변경하는데 있어서, 거리 임계치 값은 샘플 데이터를 바탕으로 hybrid spill-tree를 구축 시에 미리 계산해 놓은 데이터 간 평균 k 번째 거리를 사용한다. 마스터 서버는 hybrid spill-tree의 범위 검색을 통해 결정된 여러 분산 지역 색인 서버에 특징 벡터를 병렬로 전달한다. 두번째 단계로, 특징 벡터를 받은 각 분산 지역 색인 서버들은 VA-file를 바탕으로 k-최근접점 검색을 병렬로 수행한다. 그리고 검색 결과로서 k개의 특징 벡터와 질의 특징 벡터와의 거리를 마스터 서버에 전달한다. 마스터 서버는 여러 지역 색인 서버들로부터 얻은 결과들을 통합하여 질의 특징 벡터에서 가장 가까운 k개의 특징 벡터를 추출하여 k-최근접점 결과로 사용자에게 전달한다. 여기서 여러 지역 색인 서버의 k-최근접점 검색 결과를 통합하는 것은 마스터 서버가 아닌 제 3의 서버에서 수행될 수 있다. 분산 벡터 근사 트리의 k-최근접점 검색의 성능 향상을 위한 주요 요인은 여러 지역 색인 서버에서의 VA-file 기반 k-최근접점 검색을 병렬로 수행하는데 있다. 지금까지 설명한 k-최근접점 검색에 대한 알고리즘은 그림 8과 같다.

```
Alogrithm M_KNNSearch (query, k)
Input: high dimensional query point,
       number of desired results
Output: k nearest neighbors

1. serverList = rangeSearchHybridSpillTree(treeRoot, query);
2. call L_KNNSearch(query, k) of all local index servers
   of serverList in parallel;
3. knn = mergeResult();

Algorithm L_KNNSearch (query, k)
Input: high dimensional query point
       number of desired results
Output: k nearest neighbors

1. va = getVectorApproximation(query);
2. candidates = pruneVectorData(va);
3. knn = refineCandidates(candidates, query);
```

그림 8 분산 벡터 근사 트리의 k-최근접점 검색 알고리즘

## 5. 성능 평가

본 장에서는 분산 벡터 근사 트리의 성능을 평가하기 위하여 수행한 시험 및 그 결과에 대해서 설명한다. 분산 벡터 근사 트리의 성능은 클러스터 환경을 바탕으로 한 분산 색인 구조 중에서 가장 최근 구글에 의해 제안된 분산 hybrid spill-tree[18]의 성능과 비교 분석하도록 한다. 분산 hybrid spill-tree와 분산 벡터 근사 트리는 M-tree C++ 패키지[20]를 사용하여 개발하였다.

표 1 실 세계 시험 데이터 정보

| 데이터 | 개수 | 차원수 | 분포 유형 |
|---|---|---|---|
| Aerial40 | 270,000 | 61 | skewed |
| Corel_UCI | 66,619 | 65 | uniform |

우리는 실 세계 데이터와 실 세계 데이터를 바탕으로 합성하여 만든 데이터를 기반으로 성능 시험을 수행하였다. 우리가 사용한 실 세계 데이터는 Aerial40과 Corel-UCI[21]로 데이터의 특징을 요약하면 표 1과 같다.

시험 성능은 100개의 다른 질의 특징 벡터를 사용하여 k-최근접점 검색을 수행한 평균 실행 시간과 평균 검색 정확도를 바탕으로 평가하였다. 모든 시험은 리눅스 기반의 8대 컴퓨팅 노드에서 수행되었으며, 모든 컴퓨팅 노드들은 하나의 글로벌 파일 시스템을 가진다. 시험에 참가한 컴퓨팅 노드들은 3.40 GHz Pentium® D CPU 프로세서와 2.4GB의 메모리의 동일 스펙을 가진다. 분산 hybrid spill-tree와 분산 벡터 근사 트리는 하나의 마스터 서버와 6 개의 지역 색인 서버, 그리고 1개의 결과 병합 서버를 가진다. 이러한 구조는 분산 hybrid spill-tree에서 제안한 MapReduce 실행 환경과 동일한 환경을 만들기 위함이다. 한편, 시험을 하는 과정에서 6개 이상의 지역 색인 서버를 가지는 실행 설정을 위하여, 하나의 컴퓨팅 노드에서 여러 개의 지역 색인 서버가 운영될 수 있도록 하였다.

### 5.1 실 세계 데이터 시험

많은 응용 데이터는 대부분 균일하지 않은 분포를 가지기 쉽다. 이에 우리는 먼저 균일하지 않은 데이터 분포를 가지는 Aerial40을 바탕으로 분산 벡터 근사 트리와 분산 hybrid spill-tree의 성능 시험을 수행하였다. Aerial40은 61차원으로 270,000개의 특징 벡터를 가지는 고차원 데이터 집합이다.

분산 hybrid spill-tree와 분산 벡터 근사 트리의 공정한 성능 평가를 위하여, 두 색인 구조의 상위 트리는 같은 샘플 데이터를 바탕으로 구축하였다.

그림 9는 최근접점 개수 k의 증가에 따른 k-최근접점 검색의 성능을 나타낸다. 분산 벡터 근사 트리의 k-최근접점 검색 시간은 평균 분산 hybrid spill-tree의 검색 시간보다 2배 가까이 빠르다. 게다가 요구되는 최근접점의 개수가 많아질수록 분산 벡터 근사 트리와 분산 hybrid spill-tree의 검색 성능의 차이가 꾸준히 증가함을 알 수 있다. 이는 분산 hybrid spill-tree와 분산 벡터 근사 트리의 상위 트리 탐색 시간은 k 값에 따라 비슷한 양상을 가지는 반면, 분산된 지역 색인 트리 검색의 경우 VA-file은 hybrid spill-tree에 비해 k 값 증가에 민감하지 않다는데 그 이유를 찾을 수 있다. 즉, hybrid spill-tree가 k 값이 증가할수록 트리의 디렉토

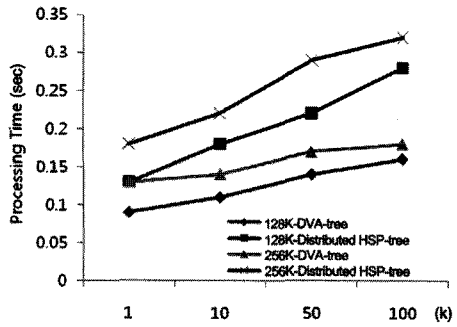
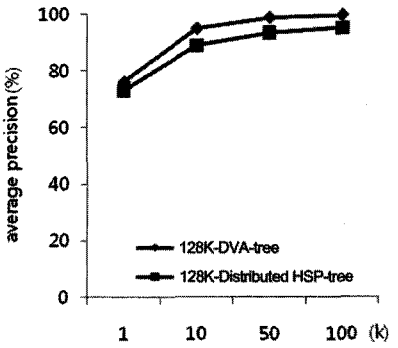
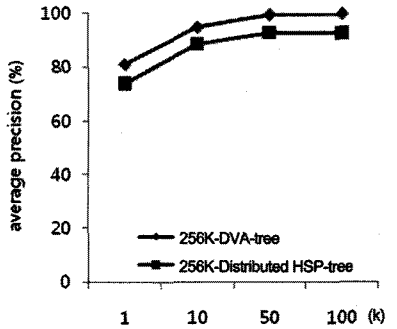



그림 9 skewed data의 k-최근접점 검색 시간




(a) The top tree with 128K leaf pages




(b) The top tree with 256K leaf pages

그림 10 skewed data의 k-최근접점 검색 정확도

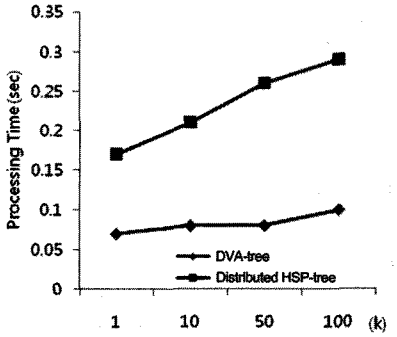



그림 11 uniform data의 k-최근접점 검색 시간

리 처리를 위한 부하가 커지는 반면, VA-file은 메모리 기반으로 특징 벡터의 근사값을 바탕으로 필터링된 아주 적은 수의 특징 벡터를 기반으로 파일 기반 랜덤 검색을 수행하기 때문이다.

한편, 분산 벡터 근사 트리와 분산 hybrid spill-tree는 상위 트리 내 말단 노드의 페이지 크기를 적게 할수록 더 좋은 성능을 보인다. 이는 같은 샘플 데이터를 기반으로 트리를 구축할 때 페이지 크기를 작게할수록 더 많은 말단 노드가 생성되기 때문이다. 말단 노드에 매핑된 분산 지역 색인 서버들은 병렬로 k-최근접점 검색을 수행하기 때문에, 페이지 크기가 작은 상위 트리의 경우 더 많은 수의 지역 색인 서버가 병렬적으로 검색을 수행하게 된다. 하지만 분명한 것은 분산 벡터 근사 트리가 분산 hybrid spill-tree와 비교하여 노드의 페이지 크기와 상관없이 k-최근접점 검색 성능이 좋다는 것이다.

그림 10은 상위 트리의 노드 페이지 크기를 128K에서 256K로 달리하였을 경우에 k-최근접점 검색 정확도를 나타낸다. 분산 벡터 근사 트리는 분산 hybrid spill-tree와 비교하여 노드 페이지 크기가 같은 경우 더 좋은 검색 정확도를 나타냄을 알 수 있다. 이는 분산 벡터 근사 트리의 지역 색인 구조인 VA-file이 hybrid spill-tree와 달리 정확한 k-최근접점 검색을 지원하기 때문이다.

우리는 또한 균일한 분포를 가지는 실 세계 데이터인 Corel_UCI를 바탕으로 성능 시험을 수행하였다. Corel_UCI는 66,619개의 65차원을 가지는 특징 벡터를 포함한다.

그림 11은 균일한 분포를 가지는 데이터를 기반으로 k-최근접점 검색을 수행했을 때의 검색 성능을 나타낸다. 분산 벡터 근사 트리는 그림 11에서 보는 것과 같이 분산 hybrid spill-tree과 비교하여 최대 3배의 검색 성능을 보인다. 한편, 균일한 분포를 가지는 데이터 집합에서 두 색인 구조의 검색 성능 차이가 편향된 분포를 가지는 데이터 집합의 검색보다 더 크게 나타난다. 이는 편향된 데이터의 경우, 분산 벡터 근사 트리의 VA-file 검색 시에 1차 근사값에 의해 필터링되는 데이터의 개수가 적어 균일 분포 데이터의 경우보다 좀 더 많은 특징 벡터를 대상으로 2차 필터링을 위한 파일의 랜덤 검색이 수행되기 때문이다.

그림 12는 균일한 분포를 가지는 데이터를 기반으로 한 k-최근접점 검색 정확도에 대한 결과이다. 분산 벡터 근사 트리와 분산 hybrid spill-tree가 거의 비슷한 검색 정확도를 보임을 알 수 있다. 이는 분산 hybrid spill-tree가 편향된 분포보다는 균일한 분포의 데이터검색에서 좀 더 좋은 정확도를 가지기 때문이다.

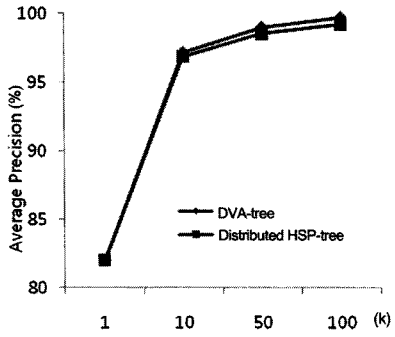



그림 12 uniform data의 k-최근접점 검색 정확도

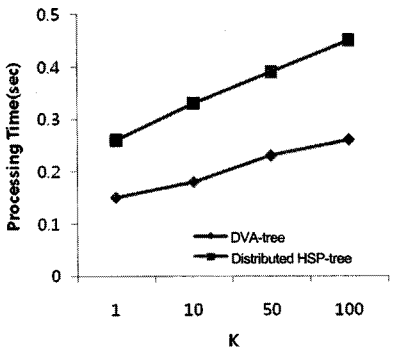



그림 14 k 값에 따른 k-최근접점 검색 시간

상기 기술한 성능 시험 결과를 통해서, 우리는 분산 벡터 근사 트리가 데이터의 분포와 상관없이 k-최근접점 검색에 있어서 좋은 성능을 제공함을 보였다. 뿐만 아니라, 분산 벡터 근사 트리는 빠른 검색을 위하여 검색 정확도를 손실하지 않음도 함께 보였다. 특히, 분산 벡터 근사 트리는 편향된 데이터에서 기존 분산 색인 구조보다 더 빠른 검색을 지원하고, 더 좋은 정확도를 보임으로써 실 세계 데이터 기반 인터넷 서비스 혹은 클라우드 컴퓨팅 서비스에 훨씬 적합한 색인 구조라 하겠다.

### 5.2 합성 데이터 시험

우리는 실 세계 데이터를 바탕으로 61차원의 편향된 분포를 가지는 합성 데이터를 생성하여 성능 시험을 수행하였다.

그림 13은 데이터의 개수를 20만개에서 100만개까지 그 수를 달리하여 k-최근접점 검색 시간을 나타낸다. 그림 13에서 알 수 있듯이, 그 결과는 상기 실 세계 데이터의 결과와 같이 분산 벡터 근사 트리가 더 좋은 성능을 보인다. 또한, 분산 벡터 근사 트리와 분산 hybrid spill-tree 간의 성능 차이는 데이터 개수가 많을수록 커진다. 이러한 결과로부터 우리는 더 많은 차원을 가지는 데이터 혹은 더 많은 수의 데이터 집합을 대상으로 k-최근접점 검색을 수행하였을 때에도 분산 벡터 근사 트리가 좋은 성능을 보임을 유추할 수 있다.

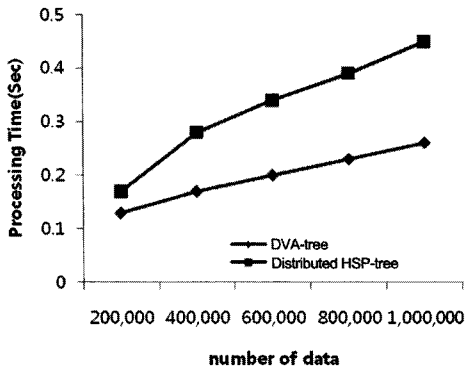



그림 13 데이터 크기별 k-최근접점 검색 시간

마지막 시험으로 우리는 100만건의 데이터를 바탕으로 최근접점 개수 k를 달리한 시험을 수행하였다. 그 결과는 그림 14와 같다. 분산 벡터 근사 트리와 분산 hybrid spill-tree 모두 최근접점 개수가 증가할수록 그 검색 시간이 선형적으로 증가한다. 하지만, 분산 벡터 근사 트리는 분산 hybrid spill-tree와 비교하여 소요되는 검색 시간의 증가가 좀 더 느림을 알 수 있다.

## 6. 향후 연구 및 결론

본 논문에서 우리는 클러스터 환경 하에서 대용량의 고차원 데이터를 바탕으로 k-최근접점 검색의 성능을 향상시키기 위하여, 다중 컴퓨팅 노드를 바탕으로 운영되는 분산 벡터 근사 트리라는 새로운 분산 색인 구조를 제안하였다. 우리는 VA-file을 분산시키기 위하여, 계층적인 클러스터링 기법으로 hybrid spill-tree를 채택하였다. Hybrid spill-tree는 대용량 고차원 데이터로부터 추출한 작은 샘플 데이터를 바탕으로 하나의 컴퓨팅 노드에 구축된다. 우리는 샘플 데이터의 크기 및 데이터 추출 기준에 대해서도 기술하였다. 한편, 구축된 hybrid spill-tree의 말단 노드는 각각 분산 컴퓨팅 노드와 매핑되며, 매핑된 분산 컴퓨팅 노드 각각은 VA-file을 구축하여 k-최근접점 검색이 병렬로 수행될 수 있도록 하였다. 우리는 여러 분산 컴퓨팅 노드들을 바탕으로 빠르게 분산 벡터 근사 트리를 구축하는 방법과 구축된 분산 벡터 근사 트리를 바탕으로 k-최근접점 검색을 위한 병렬 처리 알고리즘에 대해서 자세히 설명하였다.

한편, 우리가 제안하는 분산 벡터 근사 트리의 성능을 평가하기 위하여 다양한 분포의 데이터를 바탕으로 성능 시험을 수행하였다. 성능 시험 결과를 통하여, 우리는 분산 벡터 근사 트리가 다양한 분포의 데이터에서 좋은 검색 정확도를 가지면서 빠른 k-최근접점 검색을 수행함을 보였다.

앞으로, 우리는 분산 벡터 근사 트리의 구축 후에 발생하는 새로운 특징 벡터의 삽입 혹은 특징 벡터의 삭제에 따른 색인을 관리하는 방법 대한 연구와 더 많은 데이터를 바탕으로 한 성능 시험을 수행할 계획이다.

## 참 고 문 헌


[1] C. Zhang, A. Krishnamurthy, and R.Y. Wang, "SkipIndex: Towards a Scalable Peer-to-Peer Index Service for High Dimensional Data," *Technical Report TR-703-04*, Princeton University, 2004.

[2] B. Nam and A. Sussman, "DiST: Fully Decentralized Indexing for Querying Distributed Multidimensional Datasets," *Technical Report CS-TR-4720 and UMIACS-TR-2005-28*, Maryland University, 2005.

[3] H.V. Jagadish, B.C. Ooi, Q. H. Vu, et al., "VBI-Tree: A Peer-to-Peer Framework for Supporting Multi-Dimensional Indexing Schemes," *Proc. ICDE*, 2006.

[4] M. Bawa, T. Condie, and P. Ganesan, "LSH Forest: Self-Tuning Indexes for Similarity Search," *Proc. WWW*, 2005.

[5] P. Haghani, S. Michel, P. Cudré-Mauroux, et al., "LSH At Large-Distributed KNN Search in High Dimensions," *Proc. WebDB*, 2008.

[6] R. Weber, H.J. Schek and S. Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," *Proc. VLDB*, pp.194-205, 1998.

[7] J.T. Robinson, "The K-D-B Tree: A Search Structure for Large Multidimensional Dynamic Indexes," *Proc. SIGMOD*, 1981.

[8] D.B. Lomet and B. Salzberg, "A Robust Multi-Attribute Search Structure," *Proc. IEEE Data Engineering*, pp.296-304, 1989.

[9] N. Beckmann, H.P. Kriegel, R. Schneider, et al., "The R*-tree: An Efficient and Robust Access Method for Point and Rectangles," *Proc. ACM SIGMOD*, pp.322-331, 1990.

[10] S. Berchtold, D.A. Keim, and H.-P. Kriegel, "The X-tree: An Index Structure for High-Dimensional Data," *Proc. VLDB*, pp.28-39, 1996.

[11] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An Efficient Access Method for Similarity Search in Metric Spaces," *Proc. VLDB*, pp.426-435, 1997.

[12] T. Liu, A.W. Moore, and A. Gray, "An Investigation of Practical Approximate Nearest Neighbor Algorithms," *Proc. ANIPS*, 2004.

[13] C. Bőhm and H.P. Kriegel, "Dynamically Optimizing High-Dimensional Index Structures," *Proc. EBDT*, 2000.

[14] G.H. Cha, X. Zhu, D. Petkovic, et al., "An Efficient Indexing Method for Nearest Neighbor Searches in High-Dimensional Image Databases," *IEEE Transaction on Multimedia*, vol.4, no.1, pp.76-87, 2002.

[15] S.G. Han and J.W. Chang, "A New High-Dimensional Index Structure Using a Cell-Based Filtering Technique," *Proc. DASFAA, LNCS*, vol.1884, pp.79-92, 2000.

[16] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via hashing," *Proc. VLDB*, 1999.

[17] E. Cohen, M. Datar, S. Fujiwara, et al., "Finding Interesting Associations without Support Pruning," *Proc. ICDE*, 2000.

[18] T. Liu, C. Rosenberg, and H.A. Rowley, "Clustering Billions of Images with Large Scale Nearest Neighbor Search," *Proc. IEEE WACV*, 2007.

[19] T. Yamane, Statistics, *An Introductory Analysis*, 2nd Ed., 1976.

[20] http://www-deis.unibo.it/research/Mtree

[21] http://www.autonlab.org/autonweb/15960.html

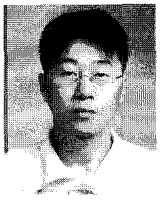


최 현 화

2000년 충남대학교 컴퓨터공학과 공학사 2002년 포항공과대학교 대학원 컴퓨터공학과 공학석사. 2002년~현재 한국전자통신연구원 선임연구원. 관심분야는 질의 처리, 분산 병렬 처리, 이벤트 스트림 처리

이 미 영

1981년 서울대학교 식품영향학과 이학사 1983년 서울대학교 대학원 계산통계학과 이학석사. 1983년~1985년 한국전자통신연구원 연구원. 1988년~현재 한국전자통신연구원 책임연구원. 관심분야는 데이터베이스 시스템, 이벤트 스트림 처리, 분산병렬 처리



김 영 창

2001년 전북대학교 컴퓨터공학과 공학사 2003년 전북대학교 대학원 컴퓨터공학과 공학석사. 2009년 전북대학교 대학원 컴퓨터공학과 공학박사. 2009년~현재 한국전자통신연구원 선임연구원. 관심분야는 공간 네트워크 데이터베이스 관리, 질의 처리, 이벤트 스트림 처리

장 재 우

1984년 서울대학교 전자계산기공학과 공학사. 1986년 한국과학기술원 전산학과 공학석사. 1991년 한국과학기술원 전산학과 공학박사. 1996년~1997년 Univ. of Minnesota, Post-Doc. 2003년~2004년 Penn State Univ. 방문연구교수. 1991년~현재 전북대학교 전기전자컴퓨터공학부 교수. 관심분야는 공간 네트워크 데이터베이스 관리, 상황 인식, 하부저장 구조

이 규 철

1984년 서울대학교 컴퓨터공학과 공학사 1986년 서울대학교 컴퓨터공학과 공학석사. 1990년 서울대학교 컴퓨터공학과 공학박사. 1989년~1994년 IBM Almaden Research Center, 초빙 연구원. 1995년~1996년 Syracuse University, CASE Center, 초빙 교수. 1997년~1998년 교육부 학술진흥재단 부설 첨단학술센터, 파견 교수. 1989년~현재 충남대학교 컴퓨터공학과 교수. 관심분야는 XML, 정보통합, 유비쿼터스 웹 서비스