주기적 편중 분할에 의한 다차원 데이타 디클러스터링

(Declustering of High-dimensional Data by Cyclic Sliced Partitioning)

김 학 철 † 김 태 완 †† 이 기 준 †††
(Hak-Cheol Kim) (Tae-Wan Kim) (Ki-Joune Li)

요 약 디스크 입출력 성능에 의해서 많은 영향을 받는 대용량의 데이타를 저장하고 처리하는 시스템에서 데이타를 다수의 병렬 디스크에 분산 시켜 저장한 후 질의 처리 시 디스크 접근 시간을 감소시키기 위한 노력들이 많이 행해졌다. 대부분의 이전 연구들은 데이타 공간이 정형의 그리드 형태로 분할되어 있다는 가정 하에 각 그리드 셀에 대해서 효과적으로 디스크 번호를 할당하는 알고리즘 연구에 치중하였다. 하지만, 그리드 형태의 분할은 저차원 데이타에 대해서는 효과적이지만 고차원 데이타에 대해서는 우수한 디스크 할당 알고리즘을 적용하더라도 디클러스터링에 의한 성능 향상을 이룰 수가 없다. 그 이유는 그리드 분할 방법은 데이타 분포 비율에 관계 없이 전체 데이타 공간을 동일한 비율로 분할하기 때문이다. 고차원 데이타는 대부분 데이타 공간의 표면에 존재 한다. 본 논문에서는 이와 같은 현상을 고려하여 데이타 표면으로부터 주기적으로 편중 분할하는 알고리즘을 이용한 새로운 디클러스터링 알고리즘을 제시한다. 다양한 실험 결과에 의하면 표면으로부터 주기적으로 편중 분할하는 방법은 차원이 증가할 수록, 또한 질의 크기가 증가할 수록 그리드 형태의 분할에 비해서 질의를 만족하는 데이타 블록의 수를 현저히 감소시킬 수 있다. 본 논문에서는 분할 결과 데이타 블록들의 배치(layout)를 이용한 디스크 번호 할당 알고리즘들을 제시하였다. 우리는 제시한 알고리즘의 성능을 보이기 위해서 다양한 차원과 디스크 수에 대해서 여러 가지 실험을 하였다. 본 연구에서 제시한 디스크 할당 알고리즘은 절대 최적의 디스크 할당 방법에 비해서 추가적인 디스크 접근 횟수가 10번을 넘지 않는다. 디클러스터링 알고리즘의 응답 시간에 대해서 그리드 분할에 대해서 가장 좋은 성능을 보이는 것으로 알려져 있는 Kronecker sequence을 이용한 디스크 할당 알고리즘과 비교 하였으며 차원이 높아짐에 따라 최대 14배까지 성능이 향상된다.

키워드 : 고차원데이타, 디클러스터링, 병렬 디스크 입출력, 영역 질의

Abstract A lot of work has been done to reduce disk access time in I/O intensive systems, which store and handle massive amount of data, by distributing data across multiple disks and accessing them in parallel. Most of the previous work has focused on an efficient mapping from a grid cell to a disk number on the assumption that data space is regular grid-like partitioned. Although we can achieve good performance for low-dimensional data by grid-like partitioning, its performance becomes degenerate as grows the dimension of data even with a good disk allocation scheme. This comes from the fact that they partition entire data space equally regardless of distribution ratio of data objects. Most of the data in high-dimensional space exist around the surface of space. For that reason, we propose a new declustering algorithm based on the partitioning scheme which partition data space from the surface. With an unbalanced partitioning scheme, several experimental results show that we can remarkably reduce the number of data blocks touched by a query as grows the dimension of data and a query size. In this paper, we propose disk allocation schemes based on the layout of the resultant data blocks after partitioning. To show the performance of the proposed algorithm, we have performed several experiments with different dimensional data and for a wide range of number of disks. Our

† 학생회원 : 부산대학교 전자계산학과
hkckim@pnu.edu
†† 비 회 원 : 부산대학교 컴퓨터 및 정보통신 연구소 연구원
twkim@quantos.cs.pusan.ac.kr
††† 정 회 원 : 부산대학교 정보컴퓨터공학부 교수
lik@pusan.ac.kr
논문접수 : 2004년 6월 12일
심사완료 : 2004년 8월 14일

proposed disk allocation method gives a performance within 10 additive disk accesses compared with strictly optimal allocation scheme. We compared our algorithm with Kronecker sequence based declustering algorithm, which is reported to be the best among the grid partition and mapping function based declustering algorithms. We can improve declustering performance up to 14 times as grows dimension of data.

Key words : High-dimensional data, Declustering, Parallel disk I/O, Range query

1. 서 론

CPU 성능의 향상 속도에 비해 상대적으로 디스크 입출력 시스템의 성능 향상 정도가 낮음으로 인해 입출력 성능에 의해 많은 영향을 받는 시스템에서 이러한 병목 현상을 해결하기 위한 많은 노력들이 행해져 왔다. 이 가운데, 다수의 병렬 디스크에 데이타들을 분산 배치한 후 질의 처리 시 동시에 여러 개의 디스크에 접근함으로써 입출력 성능을 향상 시키려는 연구가 최근 많이 행해졌으며 우리는 이를 '디클러스터링(declustering)'이라고 부른다.

데이타 디클러스터링을 위해서는 데이타를 물리적 디스크 블록 크기 단위로 분할 하는 과정과 각각의 데이타 블록에 대해서 디스크 번호를 할당하는 2단계의 과정이 필요하다. 이 때, 디클러스터링 알고리즘의 성능은 분할 단계에서 결정되는 질의를 만족하는 데이타 블록의 수와 두 번째 단계에서 결정되는 디스크 접근의 병렬성에 의해서 결정된다.

하지만, 대부분의 이전 연구들은 분할 방법이 디클러스터링 성능에 미치는 영향은 간과한 채디스크 할당 알고리즘 연구에 집중 하였다. 그들은 데이타 공간을 이루는 각 차원이 겹치지 않는 일정한 크기의 구간으로 분할되어 전체 데이타 공간이 그리드 형태로 분할되어 있다는 가정 하에 각 차원의 구간 번호로 결정되는 셀에 대해서 효과적으로 디스크 번호를 할당하는 매핑 함수 연구에 집중 하였다. Disk Modulo[1,2], Fieldwise Xor [3], Error Correcting Code[4], Hilbert Curve Allocation Method[5], Cyclic Allocation Method[6,7], GDM-based[8], Golden Ratio Sequence[9], Coloring[10], Discrepancy[11], Kronecker Sequence[12] 등 다양한 매핑 함수들이 소개되고 평가되었다[13-15]. 특별한 가정을 할 경우에는 디스크 접근의 병렬성 측면에서 절대 최적인 매핑 알고리즘도 존재하지만[16], 데이타 공간을 그리드 형태로 분할하는 것은 차원이 증가함에 따라 많은 문제점들을 야기한다. 우리는 이를 3장에서 좀 더 자세히 살펴볼 것이다.

차원이 증가할수록 데이타 공간의 표면에 존재하는 데이타의 비율은 급격히 높아지는 반면에 데이타 공간 중심의 데이타 분포 비율은 낮아진다. 본 논문에서 우리는 고차원 데이타의 이러한 특성을 고려하여 [17]에서 제시한 데이타 표면으로부터 주기적으로 편중 분할하는 방법을 적용하였다.

다양한 실험 결과 본 연구에서 적용한 분할 방법은 차원이 증가할 수록 또한 질의 크기가 증가할수록 그리드 형태의 분할에 비해서 질의를 만족하는 데이타 블록의 수를 현저히 감소시킴을 알 수 있었다.

디스크 할당 알고리즘의 대부분을 차지하고 있는 매핑 함수에 의한 방법은 그리드 형태로 분할된 데이타 블록에 대해서만 적용할 수 있다. 따라서, 본 논문에서 적용한 분할 방법에 대해서는 새로운 디스크 할당 알고리즘이 요구된다. 본 연구에서 우리는 분할 후 데이타 블록들의 유형을 고려하여 질의를 동시에 만족할 가능성이 높은 데이타 블록에 대해서 서로 다른 디스크 번호를 할당하는 알고리즘들을 제시한다. 본 연구에서 제시한 디스크 할당 알고리즘은 수행한 실험 결과 절대 최적의 디스크 할당 방법에 비해서 추가 디스크 접근 횟수가 10번을 넘지 않았다.

본 연구에서는 데이타 분포 비율을 고려하여 편중 분할하는 방법을 적용함으로써 질의를 만족하는 데이타 블록의 수를 감소시켰으며 분할 결과 데이타 블록들의 유형과 배치 상태를 이용한 디스크 할당 알고리즘을 적용함으로써 디스크 접근의 병렬성을 높였다. 이러한 결과는 디클러스터링 알고리즘의 응답 시간을 크게 감소시켜 매핑 함수를 이용하는 이전 연구들 가운데 고차원 데이타에 대해서 성능이 가장 뛰어난 것으로 알려져 있는 Kronecker Sequence를 이용한 디클러스터링 방법 [12]에 비해서 최대 14배까지 성능의 향상을 보임을 알 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 디클러스터링 문제와 관련된 개념들을 설명하고 3장에서는 대부분의 이전 연구들에서 가정하고 있는 다차원 데이타 공간을 정방형의 그리드 형태로 분할하는 방법의 문제점들을 기술한다. 4장에서는 이를 해결하기 위해서 본 논문에서 적용한 데이타 표면으로부터 주기적으로 편중 분할하는 방법을 설명하고 분할 결과에 대해서 디스크 번호를 할당하는 알고리즘들을 제시한다. 5장에서는 제시한 디클러스터링 방법의 성능을 기존의 그리드 분할

후 매핑 함수에 의한 디클러스터링 방법과 비교 실험을 통하여 제시하고 마지막으로 6장에서는 본 논문의 결론 및 향후 연구 과제를 제시한다.

2. 문제 정의 및 개념 설명

2장에서는 디클러스터링 문제를 정의하고 관련된 개념 및 성능 평가 척도를 제시한다. 표 1은 본 논문에서 앞으로 사용할 기호와 그 의미를 나타낸 것이다.

표 1 기호 및 의미

기 호	의 미
d	데이타 차원
N	전체 데이타 수
M	디스크 수
π	물리적 디스크 페이지 크기
B	1개의 디스크 블록에 저장할 수 있는 최대 데이타 수
P	N개의 데이타를 저장하기 위해 사용된 디스크 블록의 수
B_Q	질의 Q를 만족하는 데이타 블록의 수
λ	그리드 분할시 각 차원의 분할 횟수
q	d 차원 영역 질의 Q의 한 변의 길이

디클러스터링의 대상인 데이타를 물리적 디스크에 저장하기 위해서는 디스크 블록 단위로 분할해야 한다. 우리는 데이타 집합 S={$p_1,p_2,...,p_N$}에 대해서 물리적 디스크 페이지 크기가 π일 때데이타 분할 $\pi(N, B)$를 다음과 같이 정의한다.

정의 1. 데이타 분할 $\pi(N, B)$

$$\pi(N,B):\{v|v\in S\}\rightarrow\{G_1^\pi,G_2^\pi,...,G_P^\pi\},$$

여기서 $|G_i^\pi|\le B$, $\sum_{i=1}^{P}|G_i^\pi|=N$ 그리고 $i\neq j$에 대해서 $G_i^\pi\cap G_j^\pi=\varnothing$ □

이를 바탕으로 M개의 디스크에 대해서 디클러스터링 알고리즘은 다음과 같이 2 단계로 정의할 수 있다.

• 단계 1. 데이타 분할 과정 :

$$\{v|v\in S\}\rightarrow\{G_1^\pi,G_2^\pi,...,G_P^\pi\}$$

• 단계 2. 디스크 할당 과정 :

$$\{G_1^\pi,G_2^\pi,...,G_P^\pi\}\rightarrow\{0,1,2,...,M-1\}$$

이 때, 데이타 집합을 분할한 후 M개의 디스크에 분산하여 저장하였을 경우 질의 Q를 처리하기 위한 디스크 접근 횟수 $DA(Q)$는 다음과 같이 결정된다.

정의 2. 질의 Q를 처리하기 위한 디스크 접근 횟수: $DA(Q)$

$$DA(Q)=\max_{i=0}^{M-1}DA_i(Q)$$

여기서, $DA_i(Q)$는 질의 Q를 처리하기 위한 i번째 디스크 접근 횟수 □

정의 2에 의하면 디클러스터링 알고리즘의 성능 향상을 위해서는 질의를 처리하기 위해서 이용 가능한 모든 디스크에 대한 접근 횟수가 동일해야 함을 알 수 있다. 즉, 질의 조건 Q를 만족하는 데이타 블록이 M개의 디스크에 균일하게 분포되어 있어야 함을 의미한다. 이를 바탕으로 절대 최적의 디클러스터링 알고리즘을 다음과 같이 정의한다.

정의 3. 절대 최적인 디클러스터링 알고리즘(strictly optimal declustering algorithm)

다음 조건을 만족하는 디클러스터링 알고리즘은 절대 최적이다.

$$\forall_Q,\ DA(Q)=\left\lceil\frac{B_Q}{M}\right\rceil$$

특별한 경우를 제외하고 모든 질의에 대해서 위의 조건을 만족하는 디클러스터링 알고리즘은 존재하지 않는다고 알려져 있으며[16] 모든 디클러스터링 알고리즘의 성능은 다음과 같이 결정된다.

$$DA(Q)=\left\lceil\frac{B_Q}{M}\right\rceil+\alpha,\ \text{여기서}\ \alpha>0 \quad (1)$$

디클러스터링 알고리즘의 성능을 나타내는 식 (1)에서 전체 성능에 영향을 주는 요소는 질의를 만족하는 데이타 블록의 수 B_Q와 추가적인 디스크 접근 횟수 α이다. 이 중에서 B_Q는 데이타 분할 방법에 의해서 영향을 받으며 α는 디스크 할당 정책에 의해 결정된다. 따라서, 전체 디클러스터링 알고리즘 성능의 향상을 위해서는 질의처리를 위해서 접근해야 하는 전체 데이타 블록의 수를 감소시켜야 하며 동시에 접근해야 하는 데이타 블록들이 전체 디스크 상에 최대한 균일하게 분포해야 한다.

3. 관련 연구 및 연구동기

디클러스터링 문제와 관련된 대부분의 이전 연구들은 질의 처리시 방문해야 하는 전체 데이타 블록의 수는 간과한 채 디스크 접근의 병렬성에 초점을 맞추었다. 즉, 그들은 디클러스터링 알고리즘의 성능을 나타내는 식 (1)의 추가적인 디스크 접근 횟수 α를 감소시키기 위한 연구에 집중하였다. 그들은 데이타 공간을 이루는 각 차원이 겹치지 않는 여러 구간으로 나누어져 전체 데이타 공간이 그리드 형태로 분할되어 있다는 가정 하에 각 차원 구간 번호로 결정되는 셀에 대해서 효과적으로 디스크 번호를 할당하는 다양한 알고리즘들을 제시하였다[1-12]. 그림 1은 8×8 형태로 분할된 2차원 데이타에 대해서 디스크 수가 4일 때 서로 다른 매핑 함수를 적용할 때 할당된 디스크 번호를 나타낸 것이다.

특별한 가정을 하였을 경우에 디스크 접근의 병렬성

3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1
1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3
3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1
1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3

(a) DM[1,2]

3	2	1	0	3	2	1	0
2	3	0	1	2	3	0	1
1	0	3	2	1	0	3	2
0	1	2	3	0	1	2	3
3	2	1	0	3	2	1	0
2	3	0	1	2	3	0	1
1	0	3	2	1	0	3	2
0	1	2	3	0	1	2	3

(b) FX[3]

3	2	1	0	3	0	3	2
0	1	2	3	2	1	0	1
3	0	3	0	1	2	3	2
2	1	2	1	0	3	0	1
1	2	1	2	3	0	3	2
0	3	0	3	2	1	0	1
3	2	1	0	1	2	3	2
0	1	2	3	0	3	0	1

(c) HCAM[5]

3	1	2	0	3	1	2	0
2	0	1	3	2	0	1	3
1	3	0	2	1	3	0	2
0	2	3	1	0	2	3	1
3	1	2	0	3	1	2	0
0	3	0	3	2	1	0	1
1	3	0	2	1	3	0	2
0	2	3	1	0	2	3	1

(d) GRS[9]

그림 1 매핑 함수에 의한 디스크 번호 할당 예: 디스크 수는 4

측면에서는 절대 최적의 성능을 보장하는 매핑 방법[16]이 존재하지만 다차원 공간에 대해서 그리드 형태로 분할하는 것은 여러 가지 문제점들을 야기한다. 3장에서는 이에 대해서 살펴 본다.

N개의 데이타를 저장하기 위해서는 적어도 $\lceil N/B \rceil$개의 데이타 블록(그리드 셀)이 필요하며 그리드 형태로 분할 할 때 d차원 데이타에 대해서 각 차원의 분할 횟수 λ는 다음과 같다.

$$\lambda = \left\lceil \sqrt[d]{\left\lceil \frac{N}{B} \right\rceil} \right\rceil \quad (2)$$

식 (2)의 분할 횟수는 차원이 증가함에 따라 급격히 감소하며 그림 2는 10^6개의 데이타에 대해서 디스크 페이지 크기는 4096 바이트, 각 차원은 float형 4바이트 값을 가진다고 가정할 때 차원의 증가에 따른 분할 횟수 λ를 나타낸 것이다.

그림 2에서 알 수 있듯이 9차원 데이타일 경우 3번 이하의 분할이 발생하며 14차원 이상의 데이타에 대해서는 2번 이하의 분할이 발생한다. 즉, 14차원 이상의 데이타에 대해서는 일부 차원에 대해서 단 1번 분할이

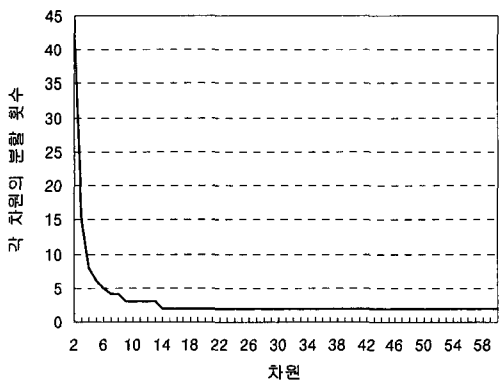


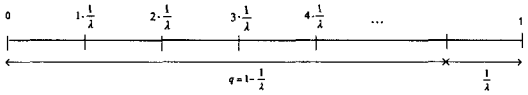
그림 2 차원의 증가에 따른 각 차원의 분할횟수

발생하게 된다. 각 차원의 분할 횟수는 질의 크기에 따라 영역 질의와 겹치는 그리드 셀의 수를 결정하며 다음과 같은 성질을 만족한다.

정리 1. $[0,1)^d$ 데이타 공간에 대해서 각 차원의 분할 횟수가 λ일 때 영역질의 Q의 한변의 길이 $q(>0)$가 다음 조건을 만족하면 질의 Q는 모든 데이타 블록(그리드 셀)과 겹친다.

$$q > 1 - \frac{1}{\lambda} \quad (3)$$

증명. 다음 그림과 같이 1차원에 대해서 증명하고 이를 d차원 데이타에 대해서 확장하도록 한다.



구간 [0,1)을 동일한 크기로 λ번 분할하면 구간 1개의 길이는 $1/\lambda$이 된다. 그림에서 알 수 있듯이 질의 변의 길이 q가 $1-1/\lambda$일 때 마지막 구간을 제외한 모든 구간과 겹치게 된다. q의 길이가 $1/\lambda$보다 큰 경우, q의 시작점은 $[1,1/\lambda)$ 사이에 존재하게 되며 시작점의 위치에 관계 없이 전체 구간과 겹치게 된다. d차원 공간에 대해서 영역 질의 한 변의 길이가 모든 차원에 대해서 위와 같은 성질을 만족하면 영역 질의 Q는 모든 그리드 셀과 겹치게 된다. □

그림 3은 그림 2의 결과에 대해서 각 차원의 분할 횟수에 의한 식 (3)의 $1/1/\lambda$의 길이와 선택률(*selectivity*)에 따른 영역질의 한 변의 길이 q의 관계를 나타낸 것이다.

그림 3에서 알 수 있듯이 그리드 형태로 분할할 경우 차원이 증가 할수록 모든 그리드 셀과 겹치게 됨을 알 수 있다. 좀 더 자세히 설명하면 선택률이 0.1%일 경우 14차원 이상, 1%일 경우 12차원 이상, 10%일 경우에는 9차원 이상의 데이타에 대해서 질의 처리를 위해서 모든 데이타 블록을 접근해야 한다. 이와 같이 질의 처리 시 모든 데이타 블록을 접근할 때에는 순차적으로 디스크 번호를 할당하는 것이 가장 최적이다. 즉, 이 때에는

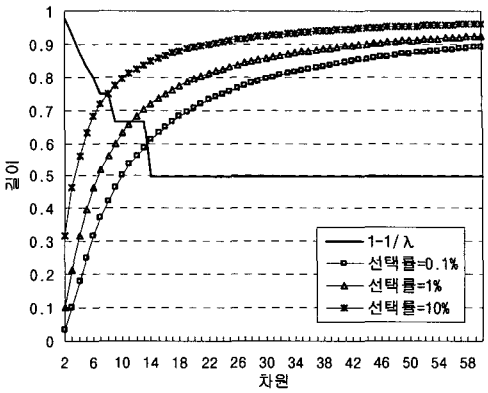


그림 3 차원의 증가에 따른 질의창의 크기 변화 및 각 차원의 분할 횟수와의 관계

지금까지 제시되었던 많은 디스크 할당 알고리즘들이 무의미해 진다. 이러한 문제점을 해결하기 위해서는 다차원 데이타에 대해서 그리드 형태와 다른 새로운 분할 방법과 그에 상응하는 디스크 할당 알고리즘이 요구된다. 다음 장에서 우리는 다차원 데이타의 특성을 고려한 새로운 분할 방법에 대해서 소개하고 분할 후 데이타 블록들의 배치 특성을 이용한 디스크 할당 알고리즘들을 제시한다.

4. 다차원 공간의 편중 분할 및 디스크 할당 알고리즘

앞 장에서 우리는 다차원 공간에 대해서 그리드 형태로 분할시 발생하는 문제점들에 대해서 살펴 보았다. 4장에서는 다차원 공간의 특성과 이를 고려하여 본 논문에서 적용한 분할 방법에 대해서 소개하고 그에 상응하는 디스크 할당 알고리즘들을 제시한다.

4.1 다차원 공간의 특성 및 편중 분할 알고리즘

4.1절에서는 다차원 공간의 중요한 특성에 대해서 살펴 보고 본 논문에서 적용한 분할 방법을 소개한다. 그림 4는 $[0,1)^d$공간에 대해서 전체 데이타 공간에 균일하

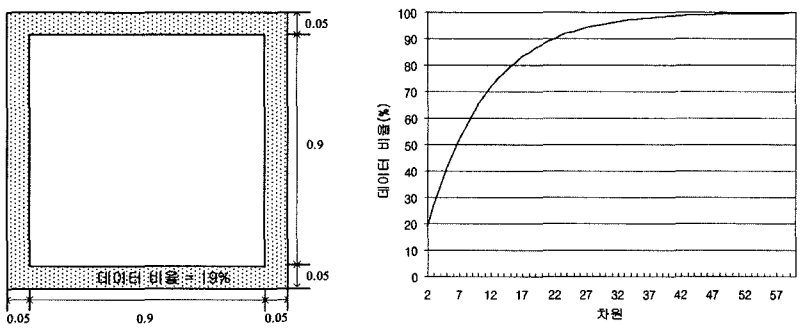


그림 4 차원의 증가에 따른 데이타 표면으로부터 0.05 거리에 존재하는 데이타 비율

게 분포하는 데이타에 대해서 차원이 증가함에 따라 표면으로부터 0.05 거리에 있는 데이타의 비율을 나타낸 것이다.

그림 4의 왼쪽에서 알 수 있듯이 2차원 데이타의 경우에는 데이타 공간 중심에 81%의 데이타가 존재하며 단지 19%의 데이타만 이 영역에 존재하지만 7차원 공간에서는 50% 이상의 데이타가 데이타 표면으로부터 5% 이내의 거리에 존재한다. 이 비율은 차원이 증가함에 따라서 급격히 증가하여 고차원 데이타일 경우에는 전체 데이타의 90% 이상이 이 영역에 존재한다.

지금까지 제시된 디클러스터링 알고리즘들은 데이타 비율에 관계 없이 전체 데이타 공간을 동일한 비율인 그리드 형태로 분할하였다. 하지만, 입출력의 병렬성을 높이기 위해서는 데이타가 집중되어 있는 영역은 최대한 서로 다른 디스크에 분산되어 저장되어야 한다. 다시 말해서 차원이 증가할수록 데이타 표면에서 많은 분할이 발생해야 한다. 고차원 데이타에 대한 이러한 특성을 고려하여 본 연구에서는 데이타 표면으로부터 주기적으로 편중 분할하는 알고리즘을 적용하였다.

우리는 [17]에서 고차원 데이타에 대해서 3가지 편중 분할 알고리즘을 제시 하였으며 본 연구에서는 이 가운데 성능이 가장 좋은 것으로 판명된 CSP(Cyclic Sliced Partition) 방법을 적용하였다.

$[0,1)^d$ 공간에 존재하는 N개의 데이타를 저장하기 위해서는 $P=\lceil N/B \rceil$개의 데이타 블록이 필요하며 각 데이타 블록의 최소 경계 입방체(MBH: *Minimum Bounding Hypercube*)의 체적은 $1/P$ 와 같다. i번째 차원의 구간(*interval*) I_i를 $(low_i, high_i)$와 같이 표현할 때 d차원 최소 경계 입방체(MBH: *Minimum Bounding Hypercube*)는 $\{I_0,I_1,...,I_{d-1}\}$과 같이 정의할 수 있다. d차원 데이타에 대한 CSP분할 알고리즘은 데이타 표면으로부터 분할 축을 순차적으로 선택하여 하한값(*low*)을 이용하여 d번 분할하고 같은 방법으로 분할 축을 선택하여 상한값(*high*)을 이용하여 d번 분할한다. 위와 같은 과정을 생성되는 전체 데이타 블록의 수가 P가 될 때까지 재귀적으로 반복한다. 그림 5는 2차원 데이타에 대해서 필요한 데이타 블록의 수가 20일 때 분할 결과를 나타낸 것이다.

그림 5에서 숫자는 데이타 블록이 생성되는 순서를 나타내며 Q_1, Q_2는 영역 질의를 의미한다. 위의 예에서 알 수 있듯이 CSP 분할 방법은 데이타 분포 비율에 비례하여 데이타 공간 중심보다 표면에서 많은 분할이 발생함을 알 수 있다.

그림 6은 본 논문에서 적용한 CSP 분할 방법을 자세히 기술한 것이다.

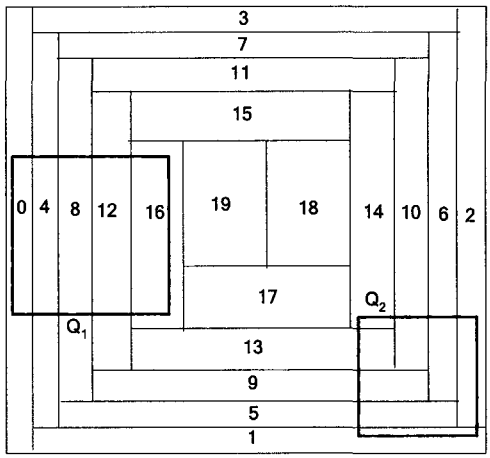


그림 5 2차원 데이타에 대한 CSP 분할 예(P=20)

Algorithm CSP(d:차원, N:데이터 수, B: 1개의 디스크 블록에 저장할 수 있는 최대 데이터 수)

{

$P \leftarrow \lceil N/B \rceil$ /* 필요한 데이터 블록의 수 계산 */

$S \leftarrow \{I_0, I_1, ..., I_{d-1}\}$, $\forall i$, $I_i.low \leftarrow 0$, $I_i.high \leftarrow 1$

MBH R[P] /* P개의 분할 결과를 저장할 변수 */

call ***Partition***(d, 0, P, $1/P$, S, R)

}

Partition(d, i, N_B, V, *MBH S*, *MBH R*[])

- d: 데이터 차원
- i: 분할 순서
- N_B: 필요한 데이터 블록의 수
- V: 데이터 블록 1개의 체적
- S: 분할할 데이터 공간
- R: 분할 결과를 저장할 배열

{

$R[i] \leftarrow S$

/* 마지막 데이터 블록일 경우 이전에 분할하고 남은 나머지 데이터 공간 전체가 데이터 블록이 되며 그 외의 경우에는 분할 축을 순차적으로 선택하여 분할 축에 대해서 구간 정보를 갱신한다. */

If($i < N_B - 1$) {

$dp \leftarrow i \bmod d$ /* 분할축 선택 */

$V'(S) \leftarrow \prod_{j=0}^{d-1}(S.I_j.high - S.I_j.low)$ /* $S.I_j$는 S의 j번째 구간이며 $j \neq dp$ */

$\delta \leftarrow \frac{V}{V'(S)}$ /* 분할 값이 계산 */

If(($i \bmod 2d$) < d) {/* 하한값(low)을 이용하여 분할 */

$S.I_{dp}.low \leftarrow S.I_{dp}.low + \delta$

$R[i].I_{dp}.high \leftarrow S.I_{dp}.low$

}

Else {/* 상한값(high)을 이용하여 분할 */

$S.I_{dp}.high \leftarrow S.I_{dp}.high - \delta$

$R[i].I_{dp}.low \leftarrow S.I_{dp}.high$

}

call **Partition**(d, i+1, N_B, V, S, R)

}

}

그림 6 CSP 분할 알고리즘

4.2 디스크 할당 알고리즘

디클러스터링 알고리즘 연구의 대부분은 디스크 할당 알고리즘에 집중 되었다. 그들은 데이타 공간이 그리드 형태로 분할되어 있다는 가정하에 각 차원의 구간 번호를 이용하여 데이타 블록(그리드 셀)에 디스크 번호를 할당하였다. 본 연구에서 적용한 분할 알고리즘의 결과

로 생성되는 데이타 블록의 모양은 그리드 형태와 다르다. 따라서, 새로운 디스크 할당 알고리즘이 요구된다.

4.2절에서는 본 연구에서 적용한 분할 방법에 적합한 디스크 할당 알고리즘을 제시한다. 먼저, 효율적인 디스크 할당을 위해서 분할 결과 생성되는 데이타 블록의 특성을 분석하도록 한다. d차원 N개의 데이타를 저장하기 위해서 필요한 데이타 블록의 수가 N_B일 때 CSP 분할 결과는 그림 7과 같이 $\left(\left\lceil N_B/2d \right\rceil \times 2d\right)$ 2차원 그리드에 매핑 시킬 수 있다. 그림 7에서 행은 분할 주기를 나타내며 열은 동일한 분할축 또는 차원에 대해서 각각 하한값(*low*)과 상한값(*high*)에서 중심으로 향한 분할 순서를 나타낸다. L_i와 H_i는 i차원에 대해서 분할이 발생함을 의미한다. 그리고 그리드 셀 내의 값은 분할 순서를 의미한다.

L_0	L_1	L_2	...	L_{d-1}	H_0	H_1	H_2	...	H_{d-1}	
0	1	2	...	d-1	d	d+1	d+2	...	2d-1	
2d										
4d										$\lceil N_B/2d \rceil$
⋮										
						$\lceil N_B/2 \rceil - 1$				

그림 7 d차원 데이타에 대한 CSP 분할 결과의 2차원 그리드 매핑 테이블

예를 들어 그림 5의 2차원 데이타에 대한 CSP 분할 결과와 영역 질의는 그림 8과 같이 5×4 그리드에 매핑 시킬 수 있다.

Q_1		Q_2	
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

그림 8 그림 5의 분할 결과에 대한 2차원 그리드 매핑 예

본 연구에서는 효과적인 디스크 할당 알고리즘을 위해서 CSP 분할의 특성을 최대한 이용하였다. 그림 4에서 알 수 있듯이 차원이 증가함에 따라 대부분의 데이타는 데이타 표면에 존재하며 CSP 분할은 데이타 표면으로부터 중심 방향으로 주기적으로 분할하며 분할 축에 대해서 분할하는 구간의 길이는 점차적으로 증가한다. 따라서, 마지막 데이타 블록(중심 데이타 블록)의 각 차원의 구간 길이는 매우 커지게 된다. 그림 9는 차원이 증가함에 따라 마지막에 생성되는 중심 데이타 블록 각 차원의 구간 평균 길이를 나타낸 것이다.

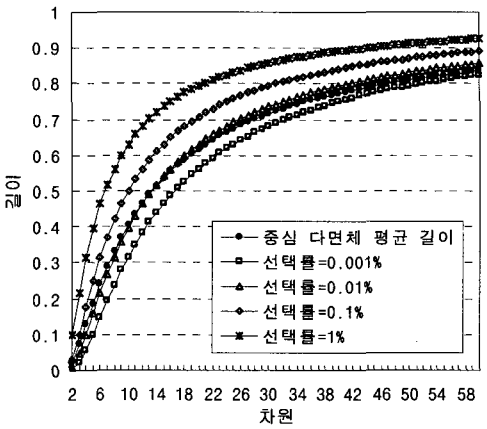


그림 9 CSP분할의 중심 다면체의 평균 길이와 선택률에 따른 질의창의 길이

그림 9에서 우리는 CSP 분할과 영역 질의 크기와의 관계를 보이기 위해서 다양한 선택률에 대해서 영역 질의 한 변의 길이를 함께 나타 내었다. CSP 분할의 특성을 나타내는 그림 9의 결과를 분석하여 본 연구에서는 2가지 디스크 할당 알고리즘을 제시한다.

CDM(Cyclic Disk Modulo): 그림 5의 0번 데이타 블록과 2번 데이타 블록은 동일한 차원(x축)에 대해서 각각 하한값(low)과 상한값(high)을 이용하여 분할한 결과이다. 그림 9의 결과에서 알 수 있듯이 마지막 중심 데이타 블록을 이루는 각 차원의 구간 길이는 매우 길어지므로 차원이 증가함에 따라 이들 데이타 블록은 영역 질의에 대해서 동시에 접근할 가능성이 낮다. 다시 말하면, 같은 분할 주기에 속하는 데이타 블록들은 동시에 접근할 가능성이 상대적으로 낮다. 이에 비해서 그림 5의 0번 블록과 4번 블록은 다른 분할 주기에 속하면서 동일한 분할 축에 대해서 하한값(low)을 이용하여 분할한 결과이다. 이러한 데이타 블록들은 동시에 접근할 가능성이 매우 높다. 즉, i번째 블록과 $(i+2kd)$(k는 정수)번째 데이타 블록은 그림 5의 영역 질의 Q_1과 같이 동시에 접근될 가능성이 매우 높다. 따라서, 분할 결과를 2차원 공간에 매핑 시킨 그림 7에서 같은 열에 있는 데이타 블록은 서로 다른 디스크에 저장되어야 한다. 본 연구에서는 이를 고려하여 분할 주기가 같은 데이타 블록(그림 7에서 같은 행에 속하는 데이타 블록)은 동일한 디스크에 저장하며 분할 주기 별로 순차적으로 서로 다

른 디스크를 할당하였다. d차원 데이타에 대한 CSP 방법의 분할 주기는 $2d$이며 디스크 수가 M일 때 CDM 방법에 의한 i번째 데이타 블록의 디스크 번호는 다음과 같이 결정 된다.

$$DiskId(i) = \left\lfloor \frac{i}{2d} \right\rfloor \bmod M$$

CSR(Cyclic Shifted Round-robin): CDM 방법은 2차원 그리드에 매핑한 결과 1개의 열(column)에 해당하는 질의(Q_1)에 대해서는 최적이지만 2개 이상의 열(column)에 해당하는 질의(Q_2)에 대해서는 성능의 저하를 초래한다. 이는 d차원 데이타에 대해서 분할 순서가 인접한 데이타 블록들은 d-1차원 공간은 공유하기 때문이다. 이러한 질의는 데이타 공간의 꼭지점 근처에 존재하는 질의이며 d차원 데이타에 대해서 2^d개의 꼭지점이 존재한다. 차원이 증가할수록 영역 질의의 크기는 매우 커지게 되며 꼭지점 근처에 존재하는 비율이 높아진다. 따라서 분할 주기가 인접한 데이타 블록들은 서로 다른 디스크에 저장되어야 한다. 이를 위해서 데이타 블록이 생성되는 순서대로 순차적으로 디스크를 할당하며 2차원 매핑 테이블에서 여러 행과 겹치는 질의를 위해서 일정한 주기별로 디스크 번호를 변화시켜 순차적으로 할당하도록 한다. d차원 데이타에 대해서 분할 주기는 $2d$이며 M개의 디스크를 가정하면 $\lceil M/2d \rceil$번의 분할 주기에 대해서 순차적으로 디스크 번호를 할당할 수 있다. 따라서, 첫 번째 $\lceil M/2d \rceil$개의 데이타 블록에 대해서 순차적으로 디스크 번호를 할당 하였다. 그런 다음 디스크 번호를 1만큼 증가시킨 후 $\lceil M/2d \rceil$개의 데이타 블록에 대해서 순차적으로 디스크 번호를 할당 하였다. 위와 같은 과정을 반복하며 d차원 i번째 데이타 블록의 디스크 번호는 다음과 같이 결정된다.

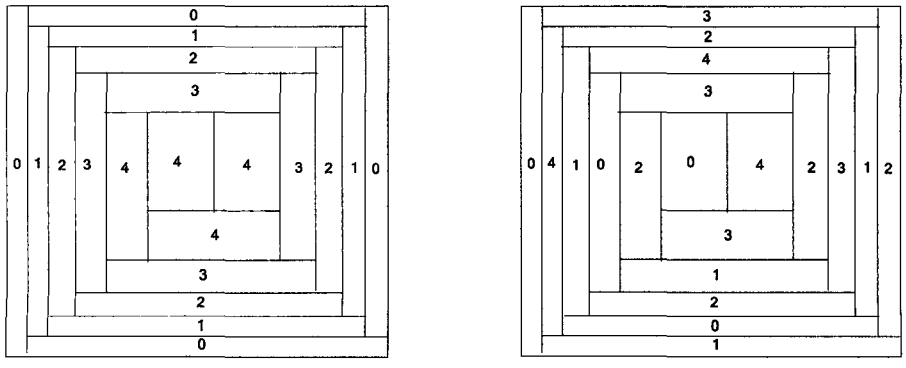
$$DiskId(i) = \left(i - \left\lfloor \frac{i}{\lceil M/2d \rceil \cdot 2d} \right\rfloor \cdot \left(\left\lceil \frac{M}{2d} \right\rceil \cdot 2d \right) + \left(\left\lfloor \frac{i}{\lceil M/2d \rceil \cdot 2d} \right\rfloor \bmod M \right) \right) \bmod M$$

그림 10은 디스크 수가 5일 때 본 논문에서 제시한 CDM 방법과 CSR방법에 의해서 그림 5의 CSP 분할 결과 데이타 블록에 할당된 디스크 번호를 나타낸 것이다.

5. 실험 결과

5장에서는 실험 결과를 보이도록 한다. 우리는 본 연구에서 제시한 새로운 디클러스터링 알고리즘의 성능을 보이기 위해서 다양한 실험을 하였다. 데이타는 $[0,1)^d$ 공간상의 10^6개의 특성 벡터이며 한 특성 벡터의 원소 값의 크기는 4 바이트로 가정한다. 따라서, 디스크 페이지 크기를 π라고 할 때 1개의 디스크 블록에 저장할 수 있는 d차원 데이타 수는 $B=\lfloor \pi/4d \rfloor$이며 10^6개의 데이타를 저장하기 위해서 적어도 $\lceil 10^6/B \rceil$개의 데이타 블록(그리드 셀)이 필요하다. 본 실험에서는 특별한 언급이 없으면 디스크 페이지 크기 π는 4096 바이트로 고정하였다.

매핑 함수를 이용하는 디클러스터링 알고리즘과의 비교를 위해서 데이타 공간을 그리드 형태로 분할해야 하며 이 때, 필요한 개수만큼의 그리드 셀을 생성하기 위한 각 차원의 분할 횟수는 식 (2)와 같이 결정된다. 그림 2에서 알 수 있듯이 차원이 증가할수록 각 차원의 분할 횟수는 급격히 감소하여 일반적으로 고차원 데이타에 대해서는 이진 분할(binary partition)을 가정한다. 즉, d차원 데이타에 대해서 필요한 데이타 블록의 수가 P개일 때 이진 분할을 가정하더라도 $\lceil \log_2 P \rceil$개의 차원에 대해서 단 1번 분할이 발생하며 나머지 차원에 대해서는 전혀 분할이 발생하지 않는다. 다차원 데이타에 대한 이진 분할은 '차원의 저주(Curse of Dimensionality)' 현상 때문에 많은 문제점을 내포한다. 이를 해결하기 위해서는 이진 분할에 필요한 차원의 수보다 적은 수의 차원을 선택해서 여러번 분할하는 것이다. 그리드



(a) CDM (b) CSR

그림 10 CSP 분할 결과에 대한 디스크 할당 예(디스크 수는 5)

분할을 위해서 선택된 차원의 수를 $d_p(\leq d)$라고 할 때, d_p개의 차원에 대한 분할 횟수는 식 (2)에 의해서 $\sqrt[d_p]{\lceil P \rceil}$이며 나머지 차원의 분할 횟수는 1이 된다. d 차원 데이타에 대해서 $\min(d, \lceil \log_2 P \rceil)$가지 분할 방법이 존재하며 본 논문에서 적용한 분할 방법과의 공정한 비교를 위하여 [18]에서 제시한 그리드 분할의 성능 예측 모델을 적용하여 이 중에서 영역 질의와 겹치는 그리드 셀의 수를 최소로 하는 분할 방법을 적용하였다. 그리드 분할 결과에 대해서 다차원 그리드에 대해서 가장 좋은 성능을 보이는 것으로 소개되고 있는 Kronecker Sequence[12]를 이용하여 디스크 번호를 할당하였으며 이를 본 논문에서 제시한 디클러스터링 알고리즘과 비교하였다.

또한 질의 크기가 디클러스터링 알고리즘의 성능에 미치는 영향을 보이기 위하여 전체 데이타 공간에 균일하게 분포하며 선택률이 $s(10^{-6} \leq s \leq 10^{-1})$일 때 한 변의 길이 q가 $\sqrt[d]{s}$인 영역 질의 10,000개를 생성 하였다.

5.1 분할 방법에 대한 성능 비교

디클러스터링 알고리즘의 성능을 나타내는 식 (1)에서 질의를 만족하는 전체 데이타 블록의 수 B_Q는 분할 방법에 의해서 결정된다. 데이타 분할을 위해서 본 연구에서 우리는 데이타 표면으로부터 주기적으로 편중 분할하는 알고리즘을 적용하였다. 5.1절에서는 분할 방법의 측면에서 본 연구에서 적용한 CSP 방법과 그리드 분할의 성능을 비교 하였다. 이를 위해서 다양한 크기의 10,000개의 영역 질의에 대해서 실제로 겹치는 데이타 블록(그리드 셀)의 수를 조사하여 평균을 계산하였다. 그림 11은 차원이 증가함에 따라서 그리드 분할에 의하여 생성된 블록들에 대하여 영역 질의와 겹치는 블록의 평균 수를 CSP 분할의 경우로 나눈 값인 비율을 나타낸 것이다.

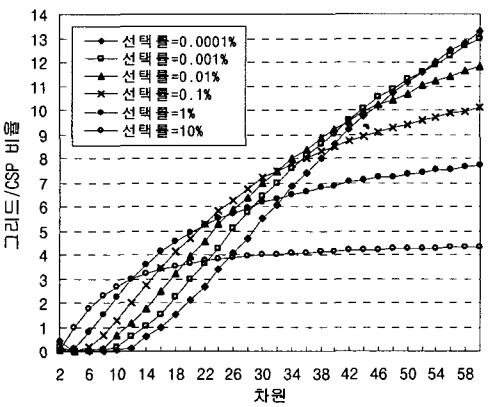


그림 11 데이타 분할 측면에서 CSP 분할의 그리드 분할에 대한 성능 향상 비율

그림 11에서 알 수 있듯이 저차원 데이타에 대해서는 성능 예측 모델에 의해서 적절한 분할 방법을 선택할 경우에는 그리드 형태의 분할도 좋은 성능을 보장한다. 하지만, 차원이 증가함에 따라 본 논문에서 적용한 데이타 표면으로부터의 편중 분할 방법이 그리드 분할에 비해서 좋은 성능을 나타내며 60차원 데이타에 대해서 선택률이 0.0001%일 때 13배 이상의 성능이 향상됨을 알 수 있다. 분할 단계에서의 이러한 성능 향상은 전체 디클러스터링 알고리즘의 성능 향상으로 귀결되며 5.3절에서 그 결과를 보이도록 한다.

일반적으로 차원이 증가할수록 성능이 향상되지만 그 비율은 질의 크기와 차원에 따라서 달라진다. 즉, 차원이 낮을 때에는 질의 크기가 증가할수록 성능이 많이 향상되며 차원이 증가하면 질의 영역이 작을 때 성능 향상의 비율은 높아진다. 이러한 이유는 그리드 방법은 전체 데이타 영역을 데이타 분포 비율에 관계없이 동일한 비율로 분할 하므로 질의 크기에 의해서 많은 영향을 받기 때문이다. 차원이 높은 경우 선택률에 따른 질의창의 크기 변화는 미세하다. 고차원의 데이타에 대해서는 극히 작은 선택률의 영역 질의도 데이타 영역의 대부분을 포함하게 된다. 그리드 분할은 데이타의 존재 비율이 극히 낮은 데이타 공간의 중심 부분에 대해서도 동일한 비율로 분할하는 반면 CSP 방법은 중심 영역은 거의 분할을 하지 않는다. 따라서, 그리드 분할은 작은 영역 질의에 대해서도 CSP 분할 방법에 비해서 많은 데이타 블록을 접근하게 되며 질의창이 커지게 되면 상대적으로 그 비율은 줄어든다.

5.2 디스크 할당 알고리즘에 대한 성능 비교

앞 절에서 우리는 본 연구에서 적용한 분할 방법과 대부분의 이전 연구에서 분할 방법으로 가정하고 있는 그리드 분할에 대해서 성능 비교를 하였다. 분할 방법과 함께 디클러스터링 알고리즘의 성능에 영향을 주는 요소는 분할 후 데이타 블록에 대해서 디스크 번호를 할당하는 정책이다. 5.2절에서는 이에 대한 성능 비교 결과를 제시한다.

이를 위해서 8~64개의 디스크와 10^{-6}~10^{-1} 선택률에 해당하는 영역 질의에 대해서 다양한 실험을 하였다. 디스크 수가 M이고 영역 질의 Q와 겹치는 데이타 블록의 수가 B_Q일 때 최적의 디스크 할당 방법에 의한 정의 2의 디스크 접근 횟수는 $\lceil B_Q/M \rceil$이다. 본 논문에서는 디스크 할당 측면에서의 성능 척도를 위해서 M개의 디스크 중에서 질의 Q를 처리하기 위한 접근 횟수가 가장 높은 디스크의 접근 횟수와 최적의 디스크 할당 경우의 디스크 접근 횟수와의 차이를 적용 하였다.

그림 12는 디스크 수가 16, 64일 때 차원이 증가함에 따라 본 논문에서 제시한 디스크 할당 알고리즘의 성능

을 나타낸 것이다.

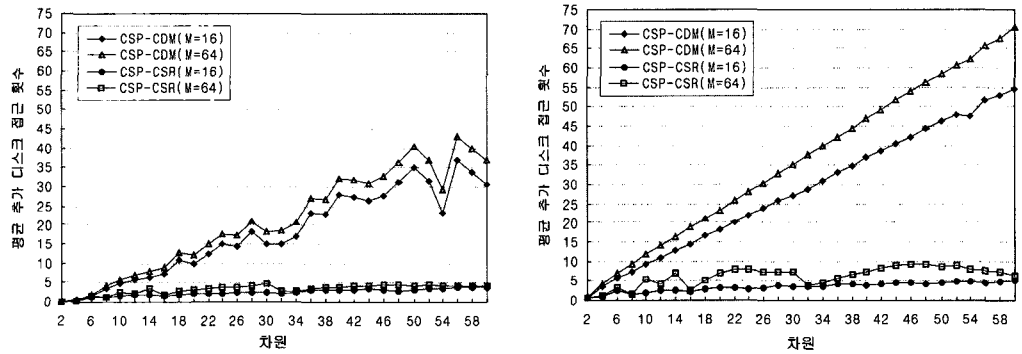
이후의 실험 결과에서 CSP-CDM과 CSP-CSR은 각각 CSP 분할 후 본 연구에서 제시한 CDM 방법과 CSR 방법을 이용하여 디스크 번호를 할당한 것을 나타내며 M은 디스크 수를 의미한다. 실험 결과에서 알 수 있듯이 분할 주기별로 순차적으로 디스크 번호를 할당하는 CDM 방법에 비해서 분할 주기를 고려하며 동시에 같은 분할 주기 내에서 인접한 데이타 블록에 대해서 다른 디스크 번호를 할당하는 CSR 방법이 성능이 뛰어남을 알 수 있다. 또한 CSR 방법은 차원에 관계없이 비슷한 성능을 보이는 반면 CDM 방법은 차원이 증가할수록 추가적인 디스크 접근 횟수가 증가한다. 이러한 이유는 차원이 증가할수록 데이타 공간 구석점의 수는 지수적으로 증가하며 이에 비례하여 구석점 근처의 영역 질의의 수도 증가하기 때문이다. CSP 방법에 의해서 분할 순서가 인접한 d차원의 데이타 블록들은 d-1차원에 대해서 공유한다. CDM 방법은 분할 주기만을 고려하여 디스크 번호를 순차적으로 할당하므로 구석점 근처의 영역 질의에 대해서 성능의 저하를 초래한다.

본 연구에서 수행한 모든 실험에 대해서 CSR 할당 방법에 의한 디스크 할당 알고리즘은 정의 3의 최적 디스크 할당 방법에 비해서 추가적인 디스크 접근 횟수가 10을 넘지 않았다.

5.3 응답 시간에 대한 성능 비교

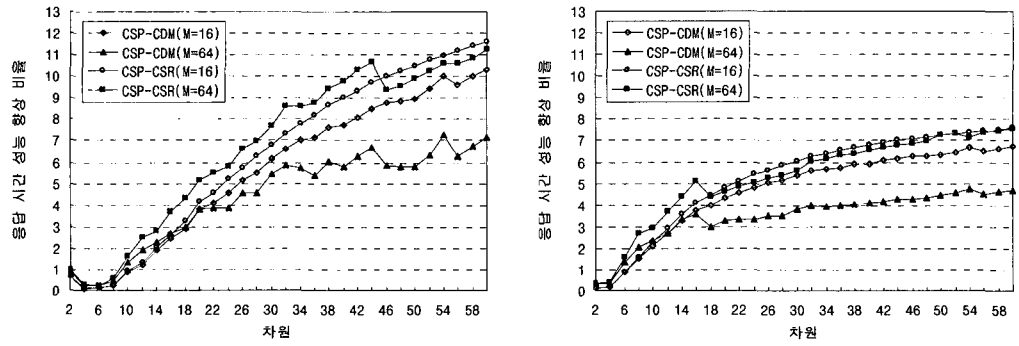
5.1절과 5.2절에서 우리는 분할 방법과 디스크 할당 측면에서 본 논문에서 제시한 디클러스터링 알고리즘의 성능 비교 결과를 제시 하였다. 5.3절에서는 디클러스터링 알고리즘의 성능을 응답 시간 측면에서 보이도록 한다. 각각의 디스크에 대해서 독립적으로 접근할 수 있고 1개의 데이타 블록을 적재하는데 걸리는 시간이 동일하다고 가정할 때 질의 Q를 처리하기 위한 응답 시간은 Q를 만족하는 블록들이 가장 많이 저장되어 있는 디스크의 블록 수로 결정된다. 따라서, 본 실험에서 응답 시간은 정의 2에 의한 디스크 접근 횟수로 가정 하였다.

그림 13은 그리드 분할 후 Kronecker sequence를 이용하는 디클러스터링 알고리즘의 본 연구에서 제시한 방법에 대한 응답 시간의 비율을 나타낸 것이다. 그림 11에서 알 수 있듯이 차원이 낮을 경우에는 전체 데이



(a) 선택률 0.01% (b) 선택률 1%

그림 12 최적 할당에 대한 평균 추가 디스크 접근 횟수



(a) 선택률 = (b) 선택률 =

그림 13 그리드 분할 후 Kronecker Sequence[9]에 의한 디스크 할당 알고리즘에 대한 성능 향상 비율

타 공간을 동일한 비율인 그리드 형태로 분할해도 영역 질의와 겹치는 데이타 블록의 수에 큰 영향을 주지 않는다. 하지만, 차원이 증가함에 따라 그리드 형태로 분할할 경우 영역 질의와 겹치는 셀의 수가 급격히 증가한다. 이는 응답 시간에 영향을 주어 그림 13에서 알 수 있듯이 동일한 선택률과 디스크 수에 대해서 본 논문에서 제시한 디클러스터링 알고리즘의 성능 향상 비율은 차원이 증가함에 따라 높아진다.

디스크 할당시 분할 주기만을 고려하는 CDM 방법에 비해서 같은 분할 주기 내에서 인접한 데이타 블록에 대해서도 다른 디스크 번호를 할당하는 CSR 방법이 좋은 성능을 보여 준다. 또한 질의 크기가 작을수록 성능 향상의 비율이 높다. 이는 그리드 분할 방법이 전체 데이타 공간에 대해서 동일한 비율로 분할함으로써 작은 영역 질의에 대해서도 접근해야 하는 데이타 블록의 수가 증가하기 때문이다. 또한 차원이 높아 질수록 디스크 수가 작을 때 성능이 더 많이 향상 되는 것도 이러한 이유 때문이다.

그림 14는 20차원 데이타에 대해서 데이타 수의 변화에 따른 응답 시간의 변화를 나타낸 것이다.

데이타 수의 증가를 반영하기 위하여 필요한 데이타 블록의 수가 2000~40000로 증가할 때 각각 디스크 수가 10, 40일 때 응답 시간을 나타 내었다. 그림 14에서 알 수 있듯이 본 논문에서 제시한 디클러스터링 알고리즘과 그리드 분할 후 Kronecker Sequence를 이용하여 디스크를 할당하는 디클러스터링 알고리즘 모두 데이타 수가 증가함에 따라 선형적으로 응답 시간이 증가함을 알 수 있다. 본 논문에서 제시한 CDM, CSR 방법에 의한 디스크 할당 알고리즘은 디스크 수가 많을 때 성능 차이가 큰 이유는 CDM 할당 방법은 디스크를 효과적으로 활용하지 못 하기 때문이다. 하지만, 본 논문에서 적용한 불균등 분할 방법에 의해서 질의를 만족하는 데이타 블록의 수를 현저히 줄일 수 있기 때문에 그리드 분할을 이용하는 디클러스터링 알고리즘에 비해서 좋은 성능을 보여 준다.

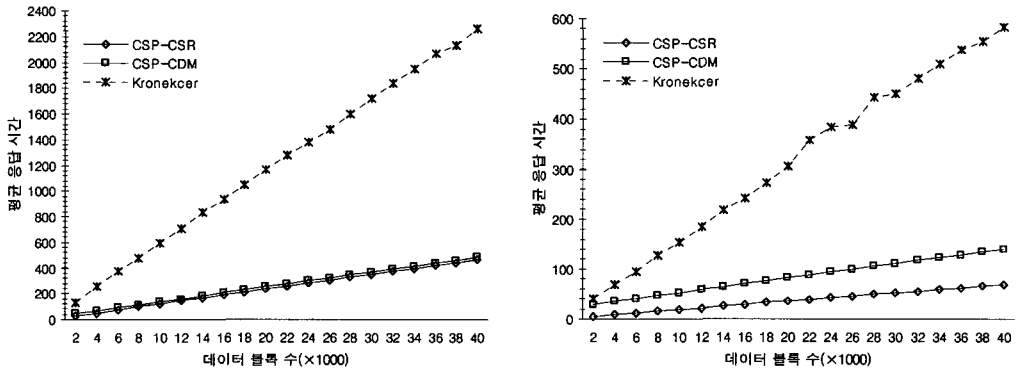
그림 15는 20차원 데이타에 대해서 디스크 페이지 크기 변화에 따른 응답 시간의 변화를 나타낸 것이다.

디스크 페이지 크기를 2배 증가시키면 필요한 데이타 블록의 수는 1/2로 감소하며 따라서 응답시간의 향상 비율은 2배가 되어야 한다. 각각의 디클러스터링 알고리즘의 디스크 페이지 크기 변화에 따른 확장성을 보이기 위하여 초기 512 바이트에 대해서 디스크 페이지 크기를 2배씩 증가시킬 때 응답 시간의 향상 비율을 나타내었다.

CSP분할 후 CDM 디스크 할당 방법에 의한 디클러스터링 알고리즘은 페이지 크기 변화에 따른 확장성이 낮은 것으로 나타났다. 이는 생성되는 데이타 블록의 수는 1/2로 감소하지만 효과적으로 디스크 번호를 할당하지 못 하기 때문이다. Kronecker Sequence에 의한 디클러스터링 알고리즘은 CDM 방법에 비해서 우수한 확장성을 보여 준다. 본 논문에서 제시한 CSP-CSR 디클러스터링 알고리즘은 디스크 수와 질의 크기에 관계없이 거의 동일한 확장성을 보여준다. 즉, 디스크 페이지 크기를 2배 증가 시킬 때 응답 시간의 향상 비율도 비례하여 2배 향상 되는 것으로 나타났다.

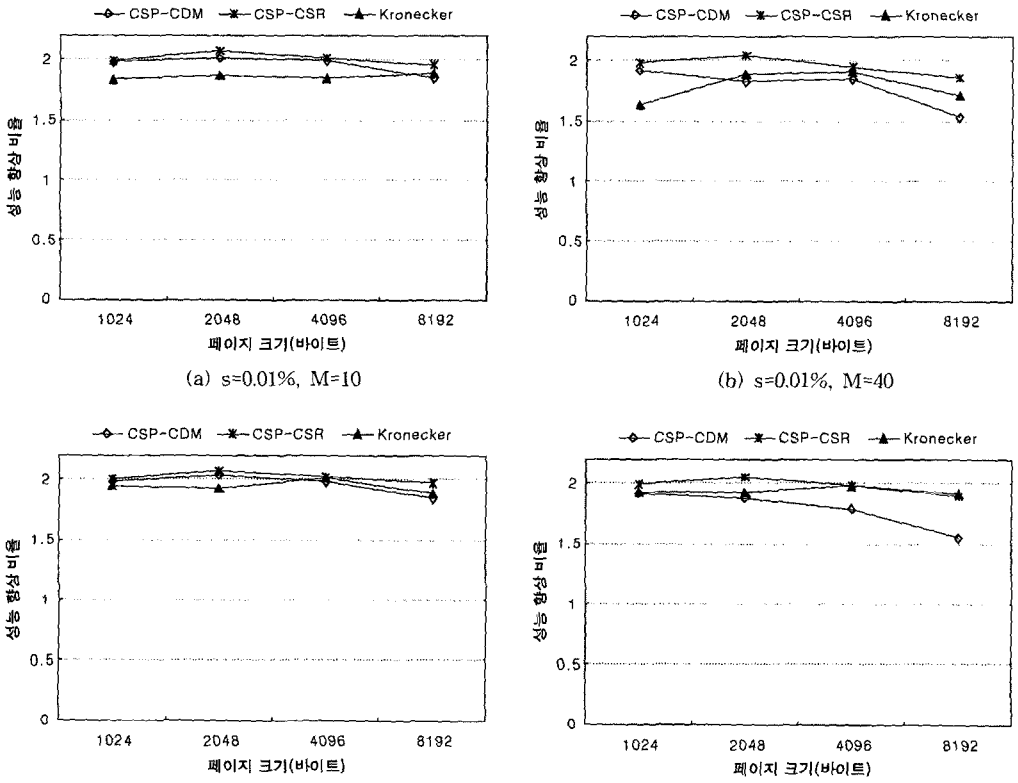
6. 결론 및 향후 연구 과제

지금까지 데이타 분산을 통한 입출력 성능 향상을 위한 많은 연구들이 행해졌다. 대부분의 이전 연구들은 데이타 공간이 그리드 형태로 분할되어 있다는 가정하에 각 차원의 구간 번호로 결정되는 그리드 셀에 대해서 효과적으로 디스크 번호를 할당하는 알고리즘 개발에



(a) 디스크 수 : 10 (b) 디스크 수 : 40

그림 14 데이타 수의 변화에 따른 응답 시간(선택률 = 1%)



(a) s=0.01%, M=10 (b) s=0.01%, M=40

(c) s=1%, M=10 (d) s=1%, M=40

그림 15 페이지 크기 증가에 따른 응답 시간 향상 비율(s: 선택률, M: 디스크 수)

집중되었다. 하지만, 그들은 데이타 분할이 전체 디클러스터링 알고리즘의 성능에 미치는 영향을 간과하였다.

본 논문에서 우리는 다차원 데이타에 대해서 그리드 형태로 분할할 때 발생하는 문제점들을 분석과 실험을 통하여 보였으며 이를 해결하기 위하여 데이타 표면으로부터 주기적으로 편중 분할하는 새로운 방법을 적용하였다. 다양한 차원과 선택률에 대해서 실험한 결과 본 논문에서 적용한 편중 분할 방법은 기존의 균등한 그리드 분할 방법에 비해서 차원이 증가할수록 영역 질의를 만족하는 데이타 블록의 수를 현저히 감소시킬 수 있다.

이전 연구에서 디스크 할당 방법으로 제시 되었던 다양한 매핑 함수는 본 논문에서 적용한 편중 분할에 의해 생성된 데이타 블록에 대해서 적용하기 어렵다. 따라서, 우리는 분할 특성을 이용한 2가지 디스크 할당 알고리즘을 제시하였다. 다양한 크기의 질의와 차원에 대한 실험 결과 본 논문에서 제시한 CSR 디스크 할당 알고리즘은 절대 최적의 디스크 할당 방법에 비해서 추가적인 디스크 접근 횟수가 10번을 넘지 않았다. 그리드 분할의 성능 예측 모델을 이용하여 효과적인 그리드 분할 후 다차원 데이타에 대해서 가장 좋은 성능을 보이는 것으로 소개되고 있는 Kronecker sequence를 이용하여 디스크 번호를 할당하는 디클러스터링 알고리즘에 비해서 응답 시간 측면에서 최대 14배까지 성능이 향상됨을 알 수 있었다.

본 연구에서 우리는 다차원 데이타에 대해서 데이타 표면으로부터 주기적으로 편중 분할하여 생성되는 데이타 블록에 대해서 2가지 디스크 할당 알고리즘을 제시하였다. 향후 연구 과제는 제시한 디스크 할당 알고리즘에 대한 분석 모델의 제시와 함께 좀 더 다양한 디스크 할당 방법의 제시와 평가이다.

참 고 문 헌

[1] H. C. Du and J. S. Sobolewski. Disk Allocation for Cartisian Files on Multiple-Disk Systems. *ACM Trans. Database Systems*, 7(1): 82-102, 1982.

[2] J. Li, J. Srivastava, and D. Rotem. CMD: A Multidimensional Declustering Method for Parallel Data Systems, *In Proc. VLDB Conf*, pages 3-14, 1992.

[3] M. H. Kim and S. Pramanik. Optimal File Distribution For Partial Match Retrieval. *In Proc.*

SIGMOD Conf, pages 173-182, 1988.
[4] C. Faloutsos and D. Metaxas. Disk Allocation Methods Using Error Correcting Codes. *IEEE Trans on Computers*, 40(8): 907-914, 1991.
[5] C. Faloutsos and P. Bhagwat. Declustering Using Fractals. *In Proc. Parallel and Distributed Information Systems Conf*, pages 18-25, 1993.
[6] S. Prabhakar, K. Abdel-Ghaffar, and A. El Abbadi. Cyclic Allocation of Two-Dimensional Data. *In Proc. ICDE Conf*, pages 94-101, 1998.
[7] S. Prabhakar, D. Agrawal and A. E. Abbadi. Disk Allocation for Fast Range and Nearest-Neighbor Queries. *Distributed and Parallel Databases* 14(2): 107-135, 2003
[8] S-W. Kuo, M. Winslett, Y. Cho, and J. Lee. New GDM-based Declustering Methods for Parallel Range Queries. *In Proc. IDEAS Symp*, pages 119-127, 1999.
[9] R. Bhatia, R. K. Sinha, and C.-M. Chen. Declustering Using Golden Ratio Sequences. *In Proc. ICDE Conf*, pages 271-280, 2000.
[10] M. J. Atallah and S. Prabhakar. (Almost) Optimal Parallel Block Access for Range Queries. *In Proc. PODS Conf*, pages 205- 215, 2000.
[11] C. M. Chen and C. T. Cheng. From Discrepancy to Declustering: Near optimal multidimensional declustering strategies for range queries. *In Proc PODS Conf*, pages 29-38, 2002.
[12] C-M. Chen, R. Bhatia, and R.K. Sinha. Multidimensional Declustering Schemes Using Golden Ratio and Kronecker Sequences. IEEE TKDE, 15(3): 659-670, 2003.
[13] B. Himatsingka and J. Srivastava. Performance Evaluation of Grid Based Multi-Attribute Record Declustering Methods. *In Proc ICDE Conf*, pages 356-365, 1994.
[14] Yuan Y. Sung. Performance analysis of disk modulo allocation method for Cartesian product files. *IEEE Trans. Softw. Eng.* 13(9): 1018-1026, 1987.
[15] B.K. Moon and J.H. Saltz. Scalability Analysis of Declustering Methods for Multidimensional Range Queries. *IEEE TKDE*. 10(2): 310-327, 1998.
[16] K. Abdel-Ghaffar and A.E. Abbadi. Optimal Allocation of Two-Dimensional Data. *In Proc ICDT*, pages 409-418, 1997.
[17] 김태완, 이기준, "고차원 데이타 패킹을 위한 주기적 편중 분할 방법", 정보과학회논문지:데이타베이스, 31권 2호, pages 122-131, 2004.
[18] T-W. Kim, H-C. Kim and K-J. Li. Analyzing the range query performance of two partitioning methods in high-dimensional space. Technical Report, Department of Computer Science, Pusan National University, 2003. http://isel.cs.pusan.ac.kr/paper/pdf/twkim_03_IPL.pdf

김 학 철
1997년 부산대학교 전자 계산학과 졸업(이학학사). 1999년 부산대학교 전자 계산학과 졸업(이학석사). 2002년 부산대학교 전자 계산학과 박사과정 수료. 2002년~현재 부산대학교 전자 계산학과 박사 학위 후보자. 관심분야는 병렬 데이타베이스 관리 시스템, GIS, 시공간데이타 베이스

김 태 완
정보과학회논문지 : 데이타베이스 제 31 권 제 2 호 참조

이 기 준
정보과학회논문지 : 데이타베이스 제 31 권 제 2 호 참조