페이지 실행시간 동기화를 이용한 다중 해쉬 결합에서 결합률에 따른 효율적인 프로세서 할당 기법

(Efficient Processor Allocation based on Join Selectivity in Multiple Hash Joins using Synchronization of Page Execution Time)

이 규 옥 † 홍 만 표 ††
(Kyu-Ock Lee) (Man-Pyo Hong)

요 약 다중 결합 질의에 포함된 다수의 결합 연산자를 효율적으로 처리하기 위해 서는 효율적인 병렬 알고리즘이 필요하다. 최근 다중 해쉬 결합 질의의 처리를 위해 할당 트리를 이용한 방법이 가장 우수한 것으로 알려져 있다. 그러나 이 방법은 실제 결합 시에 할당 트리의 각 노드에서 필연적인 지연이 발생되는 데 이는 튜플-시험 단계에서 외부 릴레이션을 디스크로부터 페이지 단위로 읽는 비용과 이미 읽는 페이지에 대한 해쉬 결합 비용간의 차이에 의해 발생하게 된다. 이들 사이의 실행시간을 가급적 일치시키기 위한 '페이지 실행시간 동기화' 기법이 제안되었고 이를 통해 할당 트리 한 노드 실행에 있어서의 지연시간을 줄일 수 있었다. 하지만 지연 시간을 최소화하기 위해 할당되어질 프로세서의 수 즉, 페이지 실행시간 동기화 계수(k)는 실제 결합 시의 결합률에 따라 상당한 차이를 보이게 되고 결국, 이 차이를 고려하지 않은 다중 해쉬 결합은 성능 면에서 크게 저하될 수밖에 없다. 본 논문에서는 결합 이전에 어느 정도의 결합률을 예측할 수 있다는 전제하에 다중 해쉬 결합 실행 시에 발생할 수 있는 지연 시간을 최소화할 수 있도록 결합률에 따라 최적의 프로세서들을 노드에 할당함으로서 다중 해쉬 결합의 실행 성능을 개선하였다. 그리고 분석적 비용 모형을 세워 기존 방식과의 다양한 성능 분석을 통해 비용 모형의 타당성을 입증하였다.

Abstract In the relational database systems, the join operation is one of the most time-consuming query operations. Many parallel join algorithms have been developed to reduce the execution time. Multiple hash join algorithm using allocation tree is one of the most efficient ones. However, it may have some delay on the processing each node of allocation tree, which is occurred in tuple-probing phase by the difference between one page reading time of outer relation and the processing time of already read one. This delay problem had been solved by using the concept of 'synchronization of page execution time' which we had proposed. However, The number of processors being allocated to minimize the delay, the coefficient of synchronization of page execution time(k), can be changed by join selectivities being applied when the joins are executed actually. In the result, multiple hash joins which are not sufficiently considered the changes can lead to degrade the performance of it. In this paper, we improved the performance of multiple hash joins by allocating the optimal number of processors which could minimize the delay being occurred when the joins are executed actually. Finally, we analyze the performance by building the analytical cost model and verify the validity of it by various performance comparison with previous method.

· 이 논문은 2000년도 두뇌한국 21 사업에 의하여 지원되었음
† 정 회 원 : 한국기계연구원 연구원
kolee@kimm.re.kr
†† 종신회원 : 아주대학교 컴퓨터공학과 교수
mphong@madang.ajou.ac.kr
논문접수 : 2000년 10월 11일
심사완료 : 2001년 1월 10일

1. 서 론

최근 데이타베이스의 규모가 대용량화되고 데이타베이스 연산도 매우 복잡해짐에 따라 이 분야에 대한 연구가 더욱 활발히 이루어지고 있다. 관계형 데이타베이스 연산 중에서 결합 연산은 다른 연산에 비해 월등히

많은 비용을 요구하는 연산으로서 데이타베이스의 규모가 커지고 복잡도가 높아짐에 따라 그 비용은 더욱 증가하게 된다.

다양한 결합 방법 중에서 해쉬 결합은 다른 결합 방법에 비해 우수한 성능을 보이며 특히, 다수의 해쉬 결합이 우향 트리 형태로 구성된 다중 해쉬 결합일 경우, 파이프라인으로 처리할 수 있다[2,5,6,7]. 하나의 파이프라인으로 구성된 다중 해쉬 결합은 여러 단계로 이루어져 있으며, 각각의 단계는 여러 개의 프로세서에 의해서 병렬로 실행될 수 있는 하나의 결합 연산으로 이루어져 있다.

최근까지는 주로 선형 실행 트리에 대한 연구가 진행된 반면, 하드웨어의 급속한 발전과 점차 복잡해지는 질의로 인해 현재에는 bushy 트리에 대한 연구가 활발히 이루어지고 있다. 또한 연산자간 병렬화와 연산자내 병렬화의 통합적 접근이라든지 결합 순서 스케줄링 및 프로세서 할당 등을 다루는 연구들이 제안되었다[4]. 최근의 연구결과들 중에서 다중 해쉬 결합 질의 처리를 위한 효율적인 방법으로 할당 트리를 이용한 정적 프로세서 할당 방법이 제안되었는데, 이 방법은 파이프라인이 가능한 bushy 트리의 노드들을 그룹핑하여 할당 트리를 생성하고 기본 릴레이션의 초기 정보, 즉 릴레이션의 카디널리티와 속성 값의 도메인 카디널리티를 이용하여 상향식(bottom-up) 방법으로 누적 실행 비용을 계산 한 후 다시 하향식(top-down) 방법에 의해 프로세서를 할당하는 방법이다[1]. 특히, 이 방법은 동기실행시간(synchronous execution time) 개념을 이용해 할당 트리 내에서 동일한 수준의 노드들이 가급적 동시에 실행이 완료 되도록 함으로서 전체적으로 지연시간을 줄이고자 하였다[4].

그러나 이 방법에서는 할당 트리의 각 노드에 할당된 프로세서들에 대해서는 각 노드 내에서의 해쉬 결합들을 수행하는 비용 모형에 대한 충분한 분석 및 평가가 고려되지 않음으로서 각 노드 내에서 필연적으로 발생될 수 있는 지연시간에 대한 처리를 할 수 없었다.

이 문제는 페이지 실행시간 동기화를 이용한 파이프라인 해쉬 결합 알고리즘을 제안하여 어느 정도 해결하였다[9,10]. 하지만 여기에 존재할 수 있는 또 다른 문제점은 실제 다중 해쉬 결합의 실행 시의 결합률(join selectivity)에 따라서 결합률을 고려한 페이지 실행시간 동기화 기법의 성능이 오히려 실제 결합률을 충분히 고려하지 않고 결합률 50%를 기준으로 페이지 실행시간 동기화 기법을 사용했던 기존 방식 보다 저하될 수 있다는 점이다.

이러한 현상은 페이지 실행시간 동기화를 위해 디스크로부터 동시에 읽어 들여야할 S 릴레이션의 페이지 수만큼의 프로세서들이 그만큼의 페이지들을 디스크로부터 동시에 읽는 시간과 이전에 읽은 그 수만큼의 페이지에 대한 실제 결합시간을 일치시키고야 말겠다는 단순한 노력에 의해 발생하게 된다. 따라서, 페이지 실행시간 동기화 기법을 통한 다중 해쉬 결합에서도 페이지 실행기간 동기화를 위해 할당되는 프로세서의 수를 실제 결합률에 따라 유연하게 할당할 수 있어야 한다.

본 논문에서는 결합 이전에 어느 정도의 결합률을 예측할 수 있다는 전제하에 다중 해쉬 결합 실행 시에 발생할 수 있는 지연 시간을 최소화할 수 있도록 결합률에 따라 최적의 프로세서들을 노드에 할당함으로서 다중 파이프라인 해쉬 결합의 실행 성능을 개선하였다. 또한 분석적 비용 모형을 세워 결합률을 고려한 방식과 그렇지 않은 방식에 대해 다양한 성능 분석을 수행하였다.

논문의 구성은, 2장에서 관련 연구로서 할당 트리 한 노드의 실행을 페이지 실행시간 동기화 기법을 이용해 수행하는 내용을 살펴보고, 3장에서는 할당 트리 한 노드에서 깊이 별로 결합률에 따른 페이지 실행시간 동기화 계수(k)의 변화와 그것이 그 노드에서의 실행 성능에 어떠한 영향을 미치는지를 살펴본다. 4장에서는 제안한 방법에 대한 성능 평가를 위한 비용 모형을 제시한다. 5장에서는 제시한 비용 모형을 통해 다양한 방법으로 성능을 평가를 하며 결합률을 고려한 방법과 그렇지 않은 방법을 비교한다. 마지막으로 6장에서는 결론으로서 본 논문의 결과를 정리한다.

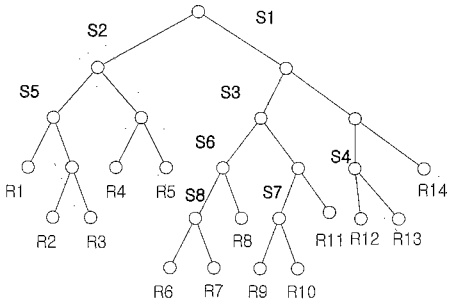
2. 페이지 실행시간 동기화를 이용한 다중 파이프라인 해쉬 결합

2.1 할당 트리를 이용한 프로세서 할당 기법

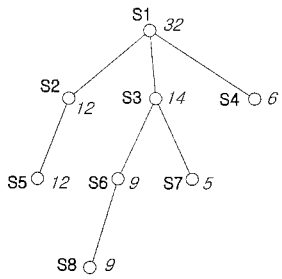
Bushy 트리로 표현된 다중 해쉬 결합 질의를 처리하기 위하여 bushy 트리에서 파이프라인이 가능한 노드들을 그룹핑하여 할당 트리로 변환한 후 질의를 처리하는 방법이 제안되었다. Bushy 트리 형태로 표현된 다중 해쉬 결합 질의를 할당 트리의 형태로 변환하고 각 노드에 프로세서를 할당하는 방법에 대해 살펴본다.

할당 트리를 생성하는 방법은 먼저 bushy 트리로 표현된 질의에서 파이프라인이 가능한 노드들을 찾는다. 그리고 파이프라인이 가능한 노드들을 그룹핑하여 할당 트리의 한 노드로 변환한다. 그림 1은 bushy 트리에서 파이프라인이 가능한 노드들을 식별하여 할당 트리로 변환하는 과정을 보여준다. (a)에서 각각의 파이프라인

그룹들은 (b)의 한 노드로 변환됨을 알 수 있다.



(a)



(b)

그림 1 Bushy 트리의 할당트리 변환

Bushy 트리를 할당 트리로 변환한 후, 상향식(bottom-up) 방법에 의해 누적실행비용(cumulative execution cost)을 계산하고 다시 하향식(top-down) 방법에 의해 각 노드에 프로세서를 할당하게 된다. 이 방법은 하나의 부(parent) 노드의 실행 전까지 모든 자식(child) 노드들이 거의 동시에 실행을 마칠 수 있도록 동기실행시간(synchronous execution time) 개념을 이용하며, 결합 연산을 수행하기 전에 단지 기본 릴레이션들의 카디널리티와 속성값이 가질 수 있는 도메인의 카디널리티만을 고려하여 누적 실행 비용을 계산하고, 그 값에 의해 프로세서를 할당하는 프로세서 할당 기법을 이용한다.

이 프로세서 할당 기법의 문제점은 기본 릴레이션의 카디널리티와 속성값이 가질 수 있는 도메인의 카디널리티 만을 고려하여 누적 실행 비용을 계산하고 그 값에 따라 프로세서들을 할당하기 때문에 각각의 결합 결과로 생성되는 중간 릴레이션(intermediate relations)들의 크기를 전혀 예측할 수 없고 그로 인해 각 노드에 프로세서를 할당을 하기 위한 보다 정확한 workload를 예측하기 어렵다는데 있다. 따라서, 이러한 문제점을 해결하기 위한 연구들도 활발히 진행되고 있다.

할당 트리의 한 노드는 그림 2와 같은 하나 이상의 해쉬 결합을 파이프라인 방식으로 처리할 수 있는 우향 트리로 구성되어 있다. 여기서 내부 릴레이션들(R)에 대한 해쉬-표의 구축은 동시에 이루어질 수 있으며 모든 해쉬-표의 구축이 완료되면 외부 릴레이션(S)은 디스크로부터 페이지 단위로 입력되어 해쉬-표를 구축할 때 사용했던 동일한 해쉬 함수를 적용하여 한 튜플씩 해쉬-표에서 비교한다. 비교한 결과, 일치하는 튜플은 결합되어 상위-수준의 결합으로 흘러가며 일치하지 않는 튜플은 무시하게 된다. 일치하여 상위-수준의 결합으로 흘러온 튜플들은 같은 방식으로 처리되어 최상위-수준의 결합에 도달하게 되며 여기서 결합에 성공한 튜플들은 최종적으로 페이지 단위로 디스크에 저장하게 된다.

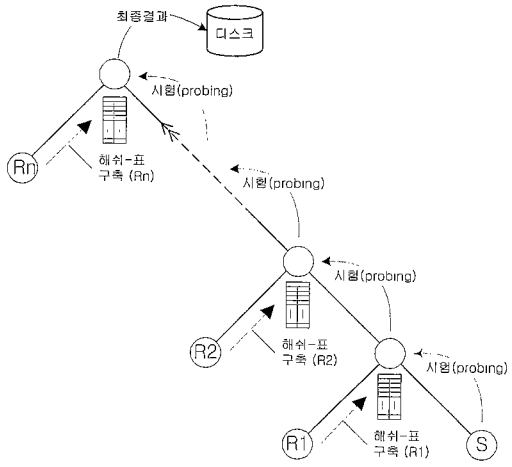


그림 2 할당 트리의 한 노드

할당 트리 한 노드에서의 해쉬 결합의 위치 및 각 해쉬 결합 내에서의 기능에 따라 각 프로세서들이 처리해야할 비용은 많은 차이를 갖게 되며 이는 프로세서들의 불가피한 지연을 야기할 수 있으며 나아가 전체 시스템의 성능을 저하시킬 수 있다. 이 같은 문제점을 방지하기 위해서는 정확하고도 보다 정밀한 비용 모형을 세우고 다양한 성능 분석을 통해 효과적인 프로세서의 할당 방법을 마련하는 것이 필수적이다.

할당 트리의 한 노드, 즉 다중 파이프라인 해쉬 결합

에서 외부 릴레이션 한 페이지에 대한 결합 실행 모형을 보면 그림 3과 같다. 여기서는 한 노드에 할당된 프로세서의 수가 그 노드내의 내부 릴레이션(R)의 수 보다 많거나 같고, 각 프로세서는 I/O 전담 프로세서가 있으며, 프로세서들 사이의 통신지연시간(communication latency)은 없다고 가정한다. 또한 프로세서 구조는 분산 메모리와 공유 디스크를 갖는 다중 프로세서 구조라고 가정한다.

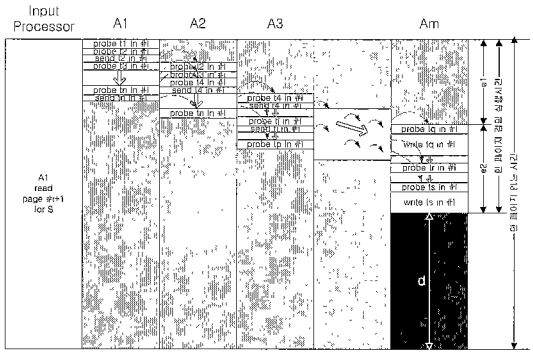


그림 3 한 페이지에 대한 다중 파이프라인 해쉬 결합 실행 모형

먼저, 할당 트리의 한 노드에 할당된 프로세서(A1, ..,Am) 전체는 그 노드에 속한 모든 내부 릴레이션들(R)을 디스크로부터 읽어서 각각의 해쉬-표를 구축한다. 구축이 완료되면, 그림 3에서와 같이 할당된 프로세서들 중 하나의 프로세서(A1)는 외부 릴레이션(S)을 디스크로부터 페이지 단위로 읽어서 해쉬 결합을 진행하여 그 결과를 상위 해쉬 결합에 할당된 프로세서(A2)에 전송한다. 이 같은 해쉬 결합은 상향식으로 이루어져 그 결과가 최상위 해쉬 결합에 할당된 프로세서(Am)에 전송되며 최상위 해쉬 결합에 할당된 프로세서들은 최종적으로 결합을 수행한 후 그 결과를 디스크에 저장하게 된다. 이 작업은 외부 릴레이션(S)의 크기를 페이지 크기로 나눈 수 만큼 파이프라인 방식으로 실행되며 마지막으로 최종적으로 입력된 페이지에 대한 해쉬 결합을 상기의 절차로 수행하게 되면 할당 트리 한 노드의 처리가 완료된다.

그러나 그림 3에서 보는 바와 같이 외부 릴레이션(S)의 한 페이지를 디스크로부터 읽는 시간에 비해 이미 읽은 한 페이지에 대한 해쉬 결합 실행시간이 현저히 차이(그림 3의 d)가 나는 것을 볼 수 있다. 이 차이는 트리의 깊이(depth)가 클수록 더욱 커지며 따라서, 결합에 참여하는 프로세서들의 지연시간이 길어지고 그 영향이 전체 질의 실행시간에 미치게 된다. 또한 일반적으로 질의 최적화 과정을 통해 내부 릴레이션의 크기보다 외부 릴레이션의 크기가 상대적으로 크기 때문에 외부 릴레이션(S)의 크기가 커질수록 전체 질의 실행시간에 미치는 영향은 더욱 커지게 된다.

따라서 외부 릴레이션(S)의 한 페이지를 디스크로부터 읽는 시간과 이미 읽은 한 페이지에 대한 해쉬 결합 실행시간과의 차이를 줄임으로서 할당 트리 내의 한 노드에서의 실행시간을 최소화할 수 있는 페이지 실행시간 동기화 기법이 제안되었다.

2.2 페이지 실행시간의 동기화

할당 트리의 한 노드 즉, 우향 트리로 표현된 다중 해쉬 결합을 파이프라인 형식으로 실행함에 있어 프로세서들의 지연시간을 최소화하고 나아가 전체 시스템의 성능을 향상시키기 위해서는 프로세서가 튜플-시험 단계에서 외부 릴레이션(S)을 디스크로부터 한 페이지 읽는 비용과 이미 읽은 페이지에 대한 처리시간 사이의 차이를 최대한 줄여야 한다. 이러한 차이를 줄이기 위해서는 외부 릴레이션(S)의 한 페이지를 읽는 시간에 다수의 페이지를 다수의 프로세서에서 동시에 읽도록 하고 그 시간 내에 이미 읽은 다수의 페이지들을 처리하도록 함으로서 페이지들을 동시에 읽는 시간과 이미 읽은 페이지들에 대한 처리가 가급적 동시에 완료될 수 있도록 하는 것이다. 이는 위에서 언급한 차이에 인한 프로세서들의 지연을 최소화할 수 있게 되는데 이 개념을 '페이지 실행시간 동기화' 라고 하였다[9,10].

페이지 실행시간 동기화는 외부 릴레이션(S)의 한 페이지를 읽는데 소요되는 시간을 이미 읽은 외부 릴레이션(S)의 한 페이지에 대한 해쉬 결합 실행시간으로 나누고, 그 값을 페이지 실행시간 동기화 계수(k)라고 할 때, 가능하다면 k개의 프로세서를 외부 릴레이션(S)의 페이지를 디스크로부터 읽는 작업에 할당하는 방법이다. 따라서, k개의 프로세서들은 외부 릴레이션(S)에서 k개의 페이지를 동시에 읽을 수 있으며, 읽혀진 k개의 페이지들에 대한 해쉬 결합 실행은 나머지 프로세서들에서 파이프라인 방식으로 실행하도록 한다. 결국, k개 단위의 페이지 입력과 이미 읽은 k개 페이지에 대한 다중 해쉬 결합 작업이 파이프라인 방식으로 처리될 수 있다. 따라서, 전체 실행시간에 많은 영향을 미치고 있는 외부 릴레이션(S)의 페이지 단위 입력 비용을 k개의 프로세서에서 동시에 실행할 수 있도록 함으로서 질의에 대한 전체 실행시간을 줄일 수 있다. 단, 여기서의 전제는 할당 트리 한 노드에 할당된 프로세서의 수는 그 노드의 내부 릴레이션(R)의 수보다는 커야 하고, 하나의 해쉬

결합에는 적어도 하나 이상의 프로세서를 할당하여야 한다는 것을 전제로 한다. 또한 프로세서 구조는 분산 메모리와 공유 디스크를 갖는 다중 프로세서 구조라고 가정한다.

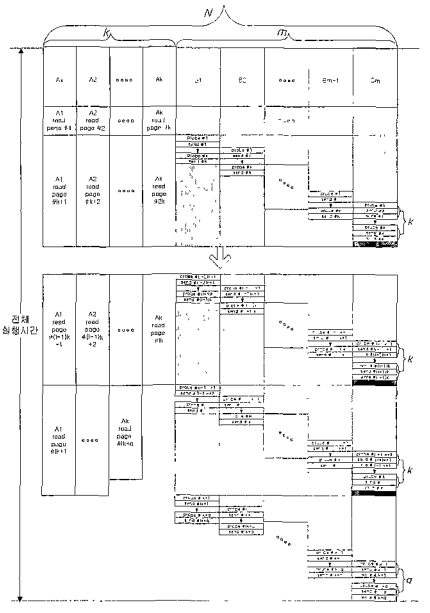


그림 4 페이지 실행시간 동기화를 이용한 다중 해쉬 결합 실행 과정

그림 4는 할당 트리의 한 노드에서 페이지 실행시간 동기화를 이용한 다중 파이프라인 해쉬 결합 의 실행 과정을 보여 준다. 여기서, k는 외부 릴레이션(S)의 페이지를 읽는 작업에 할당된 프로세서의 수이며, m은 해쉬 결합을 수행하는 프로세서의 수로서, 그 노드에 포함된 내부 릴레이션(R)의 수 즉, 해쉬 결합의 수보다는 크거나 같아야한다. 따라서, 그 노드에 할당된 전체 프로세서의 수를 N이라고 할 때, $N=k+m$ 이 된다. k값은 그림 3에서의 d 값을 줄일 수 있는 최적의 값이며 프로세서 수가 충분하다고 더 큰 값을 k값으로 부여한다고 해도 해쉬 결합 쪽에서 그 시간 안에 k개의 페이지를 처리할 수 없으므로 페이지를 읽는 프로세서들이 오히려 지연될 수 있다. 그러므로 충분한 프로세서들이 할당되었을 경우에도 k개의 프로세서만을 외부 릴레이션(S)을 페이지 단위로 읽는데 할당하고, 나머지 프로세서들은 모두 해쉬 결합을 수행하는 데 할당함으로서 결합에 소요되는 시간을 줄이는데 활용하는 것이 좋다.

3. 결합률을 고려한 페이지 실행시간 동기화

할당 트리로 표현된 다중 해쉬 결합의 실행은 각 노드에 할당된 프로세서들에 의해 실행된다. 트리 전체의 실행은 동기실행시간 개념에 입각해서 실행되며 각각의 노드는 페이지 실행시간 동기화 기법을 이용해 실행된다. 2장에서 언급한 바와 같이 할당 트리의 한 노드를 실행함에 있어 필연적으로 발생할 수 있는 지연시간을 최소화하기 위해 페이지 실행시간 동기화 기법이 제안되었다. 본 연구에서는 이 지연시간을 최소화할 수 있는 페이지 실행시간 동기화 계수(k)가 실제 다중 해쉬 결합시의 결합률, 릴레이션의 크기 그리고 디스크로부터 입/출력시에 사용되는 페이지 크기와의 상관 관계에 따라 어떠한 영향을 받는지에 대해 분석하였다.

그림 5는 할당 트리 한 노드를 구성하고 있는 다중 파이프라인 해쉬 결합에서 결합률 변화에 대해 깊이별로 분석된 페이지 실행시간 동기화 계수(k)의 변화 양상을 보여주고 있다.

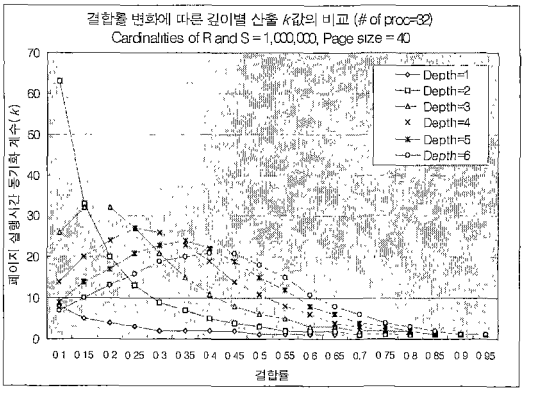


그림 5 결합률 변화에 따른 깊이별 산출 k값

그림 5에서 보는 바와 같이 할당 트리의 한 노드를 구성하고 있는 다중 파이프라인 해쉬 결합은 깊이에 관계없이 대체적으로 결합률이 증가함에 따라서 정도의 차이는 있지만 대체적으로 페이지 실행시간 동기화 계수(k)가 작아짐을 보였다. 그 이유는 파이프라인 해쉬 결합을 위해 디스크로부터 입력된 한 페이지의 결합률이 높다면 그 만큼 결합에 성공하여 디스크로 다시 저장될 튜플들이 많아지게 되고 이것이 디스크에 저장을 위한 비용을 증가시키기 때문에 페이지 실행시간 동기화를 위해서 동시에 입력시킬 S 릴레이션의 페이지 수 즉, 페이지 실행시간 동기화 계수(k)는 감소하게 되는

것이다. 또한 파이프라인 해쉬 결합의 깊이가 커질수록 일반적으로 페이지 실행시간 동기화 계수(k)의 크기가 커지는 이유는 깊이가 커지면 한 페이지의 결합 성공 튜플 수가 결합률에 따라 차츰 줄어들기 때문이다.

그림 5에서 나타난 또 하나의 주목할 점은 깊이가 3보다 큰 다중 파이프라인 해쉬 결합의 경우, 보통 결합률이 증가할수록 페이지 실행시간 동기화 계수(k)가 감소하는 추세를 보이는 반면에 결합률이 50% 미만인 경우에는 계수가 오히려 증가하는 추세를 보이는 부분이 생기는 것을 알 수 있다.

이는 한 노드로 구성된 다중 파이프라인 해쉬 결합에서 한 페이지의 결합 비용은 그림 3에서 보는바와 같이 크게 두 부분으로 이루어진다. 하나는 *e1*으로서 파이프라인의 마지막 단계를 제외한 각 단계에서 최초로 결합에 성공한 튜플이 발생하여 다음 단계로 그 튜플을 전송하게되는 데 소요되는 비용들의 합이며, 또 다른 하나는 *e2*로서 마지막 단계에서 최종적으로 결합에 성공한 튜플들을 페이지 단위로 디스크에 저장하는 비용이다. *e1*과 *e2*의 결합률 변화에 따른 깊이별 변화 추이는 각각 그림 6과 그림 7에서 보여주고 있다

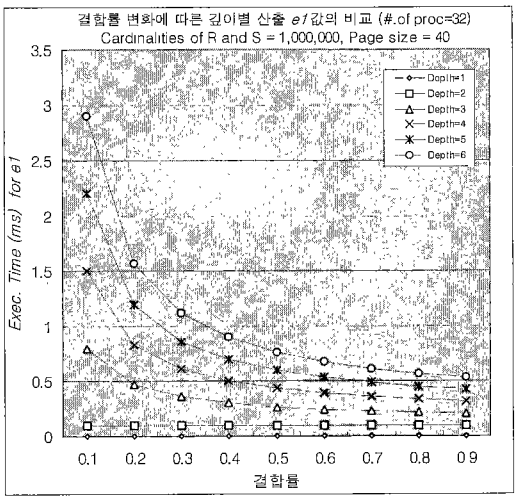


그림 6 결합률 변화에 따른 깊이별 *e1*값 추이

그림 6에서는 먼저, *e1*은 *e2*에 비해 상대적으로 매우 작은 비용이며 깊이가 커질수록 *e1*의 값은 커진다. 또한 깊이가 커질수록 결합률이 증가함에 따라 *e1*값의 감소폭도 커진다.

그림 7에서는 그림 6과는 반대로 깊이가 커질수록 *e2* 값은 작아진다. 하지만 깊이가 커질수록 결합률 변화에

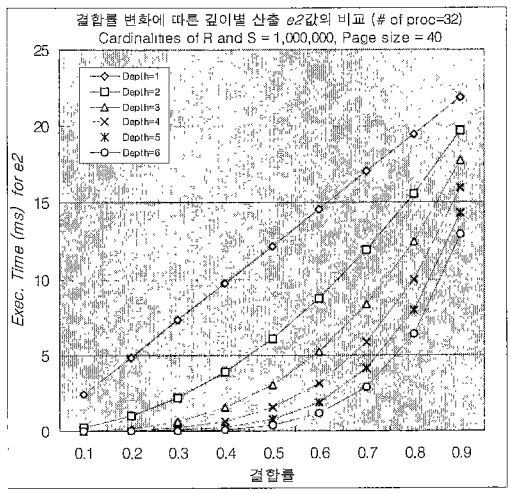


그림 7 결합률 변화에 따른 깊이별 *e2*값 추이

대한 *e2*값의 증가폭은 커짐을 알 수 있다.

*e1*과 *e2*의 결합률에 따른 변화 추이에 따라 이들의 합과 한 페이지를 디스크로부터 입력하는 비용과 사이의 관계를 토대로 계산되는 페이지 실행시간 동기화 계수(k)의 변화는 그림 5와 같은 결과를 보이게 된다.

또한 분석 결과, 릴레이션들의 크기는 페이지 실행기간 동기화 계수(k)에 영향을 미치지 않았으며, 또한, 한 페이지 크기와 페이지 실행기간 동기화 계수(k)와의 관계는 페이지 크기에 따른 관련된 각종 파라미터들의 값들이 영향을 받기 때문에 객관적인 성능분석이 불가능하여 생략하였다.

4. 비용 모형

본 장에서는 할당 트리 한 노드의 실행을 위한 다양한 비용 모형을 제시한다. 먼저 할당 트리의 임의의 한 노드에서 다중 파이프라인 해쉬 결합을 수행할 때, 한 페이지에 대한 결합 비용 모델을 제시하고 이를 통한 페이지 실행시간 동기화를 위한 동기화 계수(k) 모형을 제시한다. 또한 결정된 동기화 계수(k) 값에 따라 페이지 실행시간 동기화를 이용한 한 노드에서의 다중 파이프라인 해쉬 결합의 비용모형을 제시한다. 본 연구에서 제시한 비용 모형의 기본 가정은 다음과 같다.

(1) 컴퓨터 구조는 분산 메모리와 공유 디스크를 갖는 다중 프로세서 구조이다.

(2) 각각의 프로세서는 하나의 내부 릴레이션에 대한 해쉬-표를 저장하기에 충분한 크기를 메모리를 갖는다.

(3) 프로세서들 사이에 통신지연시간 (communica-

tion latency)은 없다.

(4) Equi-join을 근간으로 하며 유일키(primary key)를 이용한 결합이다.

본 연구의 비용 모형을 위한 매개변수 및 환경 특성 값은 표 1과 같다.

표 1 비용 모형과 성능 분석을 위한 매개변수와 특성 값

매개변수	의 미	특성값
N_P	한 노드에 할당된 전체 프로세서 수	
$\|\|R\|\|$	내부 릴레이션(R)의 카디널리티	1,000,000
$\|\|S\|\|$	외부 릴레이션(S)의 카디널리티	1,000,000
P_{speed}	프로세서 처리 속도	3 MIPS
t_{pio}	페이지 당 디스크 서비스 시간	20 ms
I_{read}	디스크로부터 한 페이지 읽는데 필요한 명령어 수	5,000
I_{write}	디스크에 한 페이지 쓰는데 필요한 명령어 수	5,000
I_{build}	해쉬-표 구축 단계에서 한 튜플 처리를 위한 명령어 수	100
I_{probe}	튜플-시험 단계에서 한 튜플 처리를 위한 명령어 수	200
I_{send}	한 튜플 전송에 필요한 명령어 수	100
P_{size}	페이지 당 튜플 수	40 tuples
ρ_i	한 노드 내의 i-번째 결합 단계에서의 결합률	
d	외부 릴레이션(S) 한페이지 읽는 시간과 한 페이지 결합 시간과의 차이	
k	d값을 줄이기 위해 할당한 외부 릴레이션(S) 입력 전담 프로세서 수	

4.1 한 페이지의 다중 파이프라인 해쉬 결합

다중 파이프라인 해쉬 결합에서 발생된는 지연시간을 줄이기 위한 페이지 실행시간 동기화 계수(k)는 외부 릴레이션(S)의 한 페이지를 입력하는 비용과 이미 입력된 한 페이지에 대한 결합 비용과의 관계에 따라 결정되게 되는데, 그 식은 (1)과 같다.

$$k = \left\lfloor \frac{\text{외부릴레이션}(S)\text{의한페이지입력비용}(\beta)}{\text{한페이지해쉬결합비용}(\alpha)} \right\rfloor \quad (1)$$

여기서, 이미 입력된 한 페이지에 대한 파이프라인 해쉬 결합 비용을 α라 했을 때, α는 3장에서 언급한 $e1$과 $e2$의 합($\alpha = e1+e2$)으로 구성된다. $e1$과 $e2$는 각각 (2)와 (3)식으로 표현된다.

$$e1 = \sum_{i=1}^{m-1} \frac{\frac{P_{size}^{i-1}}{P_{size}^{i-1} \cdot \rho_i} \cdot I_{probe} + I_{send}}{P_{speed}} \quad (2)$$

여기서, m은 다중 파이프라인 해쉬 결합의 파이프라인 단계(stage) 수 즉, 해쉬 결합의 수를 나타낸다.

$$e2 = \left(P_{size}^{m-1} \cdot \frac{I_{probe}}{P_{speed}} + \frac{\frac{P_{size}^{m}}{P_{size}} \cdot I_{write}}{P_{speed}} + \frac{P_{size}^{m}}{P_{size}} \cdot t_{pio} \right) \quad (3)$$

외부 릴레이션(S)의 한페이지 입력 비용을 β라 했을 때, β는 (4)식과 같다.

$$\beta = \frac{I_{read}}{P_{speed}} + t_{pio} \quad (4)$$

4.2 한 노드의 다중 파이프라인 해쉬 결합

할당 트리 한 노드(S)를 페이지 실행시간 동기화 기법을 이용해 그 노드에 할당된 프로세서들을 통해 실행할 때의 결합 비용을 $Join_{time}^{SyncPET}(S)$라 할 때, $Join_{time}^{SyncPET}(S)$는 (5)식과 같다.

$$Join_{time}^{SyncPET}(S) = \frac{\sum_{i=1}^{m} \frac{||R_i||}{P_{size}} \cdot \left(\frac{I_{read} + I_{build}}{P_{speed}} + t_{pio} \right)}{N_P} + \frac{\left(\frac{||S||}{P_{size}} - 1 \right)}{k} \cdot MAX\left(\frac{I_{read}}{P_{speed}} + t_{pio}, e1 + e2 \right) + \sum_{i=1}^{m-1} \frac{\frac{P_{size}^{i-1}}{P_{size}^{i-1} \cdot \rho_i} \cdot I_{probe} + I_{send}}{P_{speed}} + \left(P_{size}^{m-1} \cdot \frac{I_{probe}}{P_{speed}} + \frac{\frac{P_{size}^{m}}{P_{size}} \cdot I_{write}}{P_{speed}} + \frac{P_{size}^{m}}{P_{size}} \cdot t_{pio} \right) \quad (5)$$

$Join_{time}^{SyncPET}(S)$는 3개 항목의 합으로 구성되는데, 첫 번째 항목은 한 노드 내의 모든 내부 릴레이션들을 디스크로부터 읽어 들여 각각의 해쉬-표를 구축하는 비용이고, 둘째는, 외부 릴레이션을 k로 나눈 수에 디스크로부터 페이지 단위로 읽어 들이는 시간과 이미 입력된 페이지에 대한 결합시간 즉, $e1$과 $e2$의 합 중에서 큰 값을 곱한 비용이 된다. 이것은 실제 결합시의 결합률에 따라 한 페이지에 대한 결합 비용이 한 페이지를 디스크로부터 입력 비용 보다 커질 수 있기 때문이다. 셋째 항목은, 마지막으로 읽어 들인 페이지들에 대한 해쉬 결합 실행 비용이다.

5. 성능 분석

본 장에서는 4장에서 설정한 비용 모형들을 근거로 할당 트리 한 노드 즉, 다중 파이프라인 해쉬 결합에서 결합률의 변화에 따른 깊이별 페이지 실행시간 동기화 계수(k)를 구하고 결합률을 고려한 k값을 이용하여 실제 할당 트리 한 노드의 실행 성능을 기존 결합률 50%를 기준으로 실행하던 방법과 비교 분석하였다.

성능 분석을 위한 매개 변수는 표 1과 같다. 동일한 환경에서의 성능분석을 위해 내부 릴레이션들(R)과 외

부 릴레이션(S)의 카디널리티를 각각 1,000,000 튜플 정도로 설정하였고, 할당 트리의 각 노드에 포함된 각각의 해쉬 결합에서의 결합률은 난수(random number)를 생성하여 결합에 이용하였으며 결과의 신뢰성을 위해 1,000회 이상 반복 수행하였다.

할당 트리 한 노드 즉, 다중 파이프라인 해쉬 결합에서 결합률의 변화에 따른 깊이별 페이지 실행시간 동기화 계수(k)를 구하고 실제 할당 트리 한 노드의 실행 성능을 분석하여 기존의 결합률 50%를 기준으로 실행한 방법과 비교 분석한 결과가 깊이 별로 그림 8부터 그림 13에 나와 있다.

그림 8에서 그림 13까지에서 보는 바와 같이 깊이에

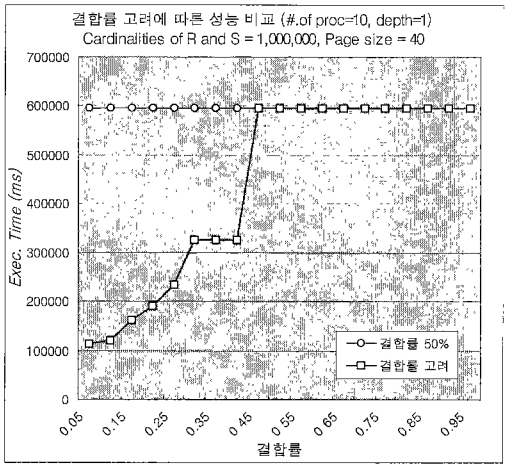


그림 8 깊이가 1인 경우

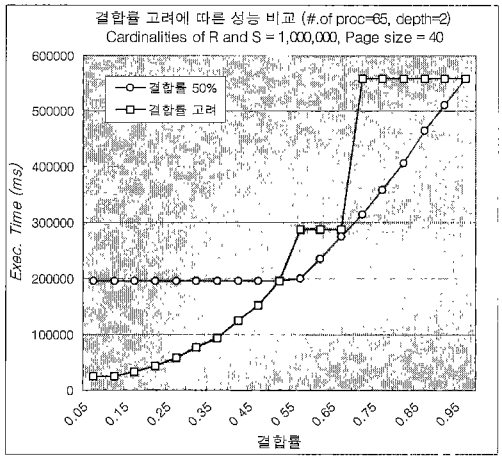


그림 9 깊이가 2인 경우

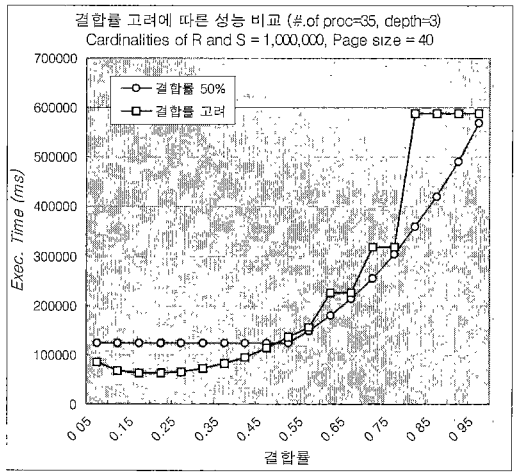


그림 10 깊이가 3인 경우

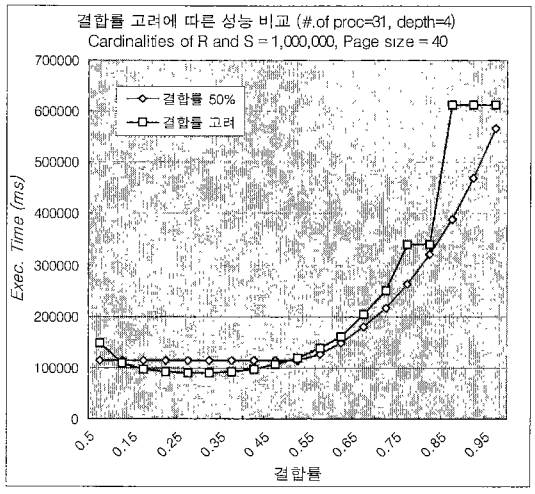


그림 11 깊이가 4인 경우

따라 다소 정도의 차이는 보이지만 깊이가 3 이하인 경우, 결합률이 50% 미만일 경우에는 기존의 결합률 50%를 기준으로 한 기존의 방법 보다 결합률을 정확히 고려하여 실행한 다중 파이프라인 해쉬 결합의 성능이 우수함을 보였다. 하지만 깊이가 4 이상일 경우에는 깊이가 커질수록 결합률을 고려한 방식의 성능이 결합률 50%를 기준으로 한 방식 보다 우수한 구간이 좁아짐을 알 수 있다.

한편, 결합률이 50%를 초과할 경우에는 깊이가 1인 경우를 제외하면 결합률을 50% 기준으로 실행하는 페이지 실행시간 동기화 방법이 오히려 우수한 성능을 보

였다. 그 이유를 살펴 보면, 결합률이 50%를 초과할 경우, 결합에 성공하여 디스크로 다시 저장되어야 하는 튜플들이 증가하게 되어 이미 입력된 k개의 페이지들에 대한 결합 비용이 오히려 k개의 페이지를 동시에 입력하는 비용을 초과하게 되고 이것은 당초 페이지 실행시간 동기화의 기본 개념과 배반되는 것이 될 수도 있기 때문이다. 하지만, 이 초과되는 비용을 방지하기 위하여 (1)식에 의해 산출된 k값을 고수하게 되면, 반대로 그만큼 외부 릴레이션을 디스크로부터 입력하는 비용이 커지게 되며, 이 입력 비용이 해쉬 결합에서 초과되는 비용을 능가하게 되어 한 노드의 전체적인 실행 성능면에서는 결합률 50%를 기준으로 하여 산출된 k값을 이용하여 실행하는 기존의 방법보다 나빠지게 되는 이유가 된다.

따라서, 페이지 실행시간 동기화를 위해 디스크로부터 동시에 읽어 들여야할 S 릴레이션의 페이지 수만큼의 프로세서들이 그만큼의 페이지들을 디스크로부터 동시에 읽는 시간과 이전에 읽은 그 수만큼의 페이지에 대한 실제 결합시간을 일치시키고야 말겠다는 단순한 노력에서 탈피하여 페이지 실행기간 동기화를 위해 할당되는 프로세서의 수를 실제 결합률에 따라 유연하게 할당할 수 있어야 한다.

본 연구에서는 페이지 실행시간 동기화 기법을 이용한 다중 파이프라인 해쉬 결합을 성능 분석한 결과 결합률이 50%를 초과하는 경우의 페이지 실행시간 동기화 계수(k)를 가능하면 결합률이 50%일 때의 동기화 계수(k)를 이용하는 것이 가장 성능이 우수함을 알 수 있었다.

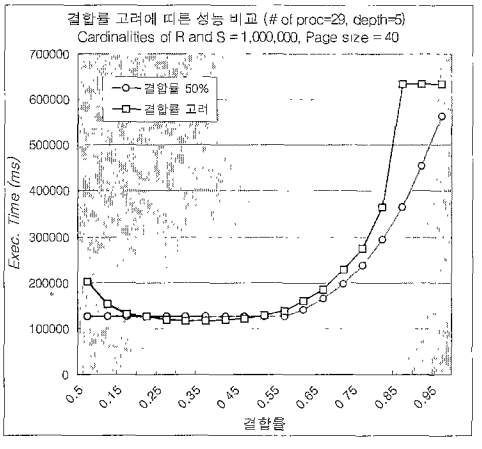


그림 12 깊이가 5인 경우

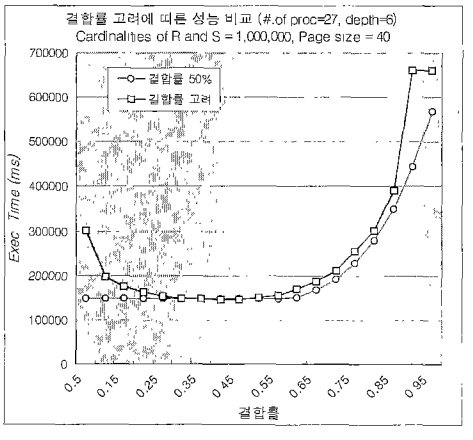


그림 13 깊이가 6인 경우

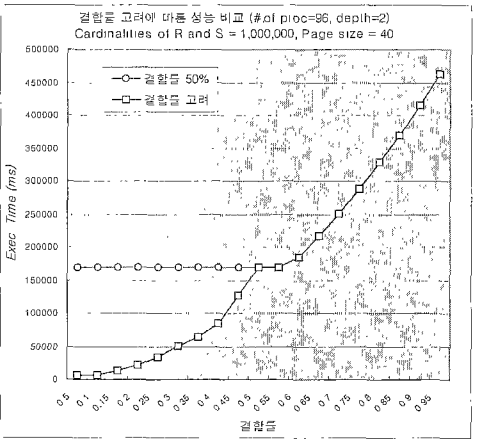


그림 14 깊이가 2인 경우

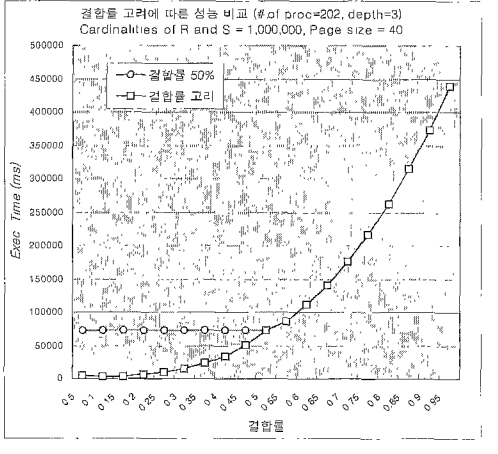


그림 15 깊이가 3인 경우

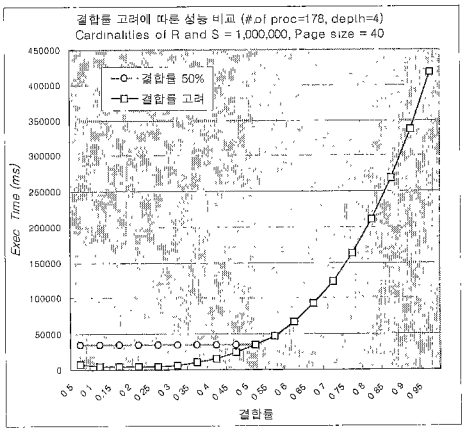


그림 16 깊이가 4인 경우

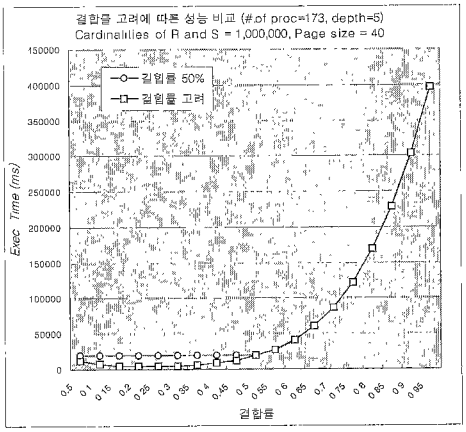


그림 17 깊이가 5인 경우

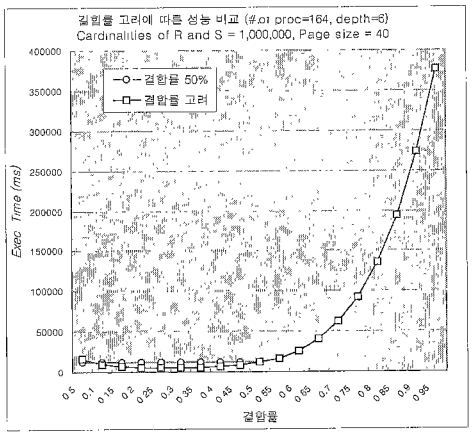


그림 18 깊이가 6인 경우

그림 14에서 그림 18까지는 깊이가 2 이상인 다중 파이프라인 해쉬 결합의 경우 결합률이 50%를 초과할 때 결합률이 50%일 때의 동기화 계수(k)를 적용하여 성능 분석한 결과를 기존의 방법과 비교한 결과를 보여준다.

그림 14에서 그림 18까지의 결과에서 보는 바와 같이 다중 파이프라인 해쉬 결합의 깊이에 관계없이 결합률이 50% 미만의 경우에는 본 논문에서 제안한 결합률을 고려한 결합 방식이 결합률을 50%로 고정하여 산출된 동기화 계수(k)를 실제 결합률에 상관없이 적용하는 기존 방식에 비해 성능이 우수함을 보였고, 특히 깊이가 낮을수록 더욱 성능이 기존 방식에 비해 월등해짐을 볼 수 있었다. 또한 결합률이 50% 이상일 경우에는 결합률이 50%일 때의 동기화 계수(k)를 동일하게 적용함으로서 성능의 저하를 방지할 수 있었다.

6. 결 론

다중 파이프라인 해쉬 결합을 실행하기 위해 할당 트리의 한 노드에 할당된 프로세서들이 가능별로 상대적인 비용의 차이를 보이게 되고, 이로 인한 불가피한 프로세서들의 지연이 다중 해쉬 결합 질의의 성능을 저하시키는 문제는 페이지 실행시간 동기화를 이용해 어느 정도 해결할 수 있었다[9,10].

또한 다중 파이프라인 해쉬 결합에서는 결합률에 따라 페이지 실행시간 동기화 계수(k)가 달라질 수 있고 이로 인한 노드 전체의 성능이 크게 달라질 수 있다.

따라서, 결합 이전에 어느 정도의 결합률을 예측할 수 있다면 페이지 실행시간 동기화 기법을 이용한 다중 파이프라인 해쉬 결합의 보다 효율적 실행을 위해서 반드시 결합률을 고려하여야 할 필요가 있다.

본 연구에서는 다중 파이프라인 해쉬 결합의 효율적 실행을 위해 실제 결합 시에 적용되는 결합률이 실행 성능에 어떠한 영향을 미치는가에 대해 분석하였다. 그리고 결합률을 충분히 고려한 페이지 실행시간 동기화 기법을 통해 보다 효율적으로 다중 파이프라인 해쉬 결합을 수행할 수 있는 기법을 제안하였고 또한 분석적 비용 모형을 세워 다양한 성능 분석을 수행하였다. 본 연구에서 제안한 결합률을 고려한 페이지 실행시간 동기화 기법을 사용하여 다중 파이프라인 해쉬 결합을 실행했을 경우, 결합률을 50%로 고정하여 실행하는 기존의 방식에 비해 실제 결합시의 결합률이 50% 미만일 경우에는 월등히 우수한 성능을 보였으며 50% 이상일 경우에는 결합률 50%일 때의 페이지 실행시간 동기화 계수(k)를 적용함으로서 성능의 저하를 방지할 수 있었다.

결론적으로, 페이지 실행시간 동기화 기법을 통한 다중 파이프라인 해쉬 결합에서도 페이지 실행기간 동기화를 위해 할당되는 프로세서의 수를 실제 결합률에 따라 유연하게 할당할 수 있어야 노드의 실행 성능을 향상시킬 수 있고 나아가 전체 할당 트리의 성능을 향상시킬 수 있게 된다.

향후 연구과제로서는 해쉬 필터(hash filter)와의 연계(interleave)를 통한 다중 파이프라인 해쉬 결합 알고리즘의 확장, 그리고 특정 프로세서 구조에서의 다중 해쉬 결합 알고리즘 개발도 과제로서 의미가 있을 것이다.

참 고 문 헌

[1] Hui-I Hsiao, Ming-Syan Chen, "Parallel Execution of Hash Joins in Parallel Databases," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 8, pp.872-883, Aug. 1997.

[2] Hui-I Hsiao, Ming-Syan Chen, and Philip S. Yu, "On Parallel Execution of Multiple Pipelined Hash Jois," *Proc. ACM SIGMOD*, pp.185-196, May 1994.

[3] Ming-Syan Chen, Ming-Ling Lo, Philip S. Yu, and Honesty C. Young, "Using Segmented Right-Deep Trees for the Execution of Pipelined Hash Joins," *18th International Conference on VLDB*, pp.15-26, August 1992.

[4] Ming-Syan Chen, P.S. Yu, and K.L. Wu, "Scheduling and Processor Allocation for Parallel Execution of Multi-Join Queries," *Proc. 8th International Conf. Data Engineering*, pp.58-67, Feb. 1992.

[5] Ming-Syan Chen, Mingling Lo, Philip S. Yu, and Honesty C. Young, "Applying Segmented Right-Deep Trees to Pipelining Hash Joins," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 7, No. 4, August 1995.

[6] Donovan A. Schneider and D.J. DeWitt, "Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines," *Proceedings of the 16th VLDB Conference*, pp.469-480, August 1990.

[7] Mingling Lo, Ming-Syan Chen, C. V. Ravishankar, and Philip S. Yu, "On Optimal Processor Allocation to Support Pipelined Hash Joins," *Proc. ACM SIGMOD*, pp.69-78, May 1993.

[8] D.J.DeWitt, and J. Gray, "Parallel Database System: The Future of High Performance Database System," *Comm. of ACM*, pp.85-98, June 1992.

[9] 이규옥, 원영선, 홍만표, "페이지 실행시간 동기화 기법을 이용한 다중 파이프라인 해쉬 결합", 정보과학회논문지:시스템 및 이론, 제27권, 제 7호, pp. 639-649, 2000.

[10] Kyuock Lee, Youngsun Weon, and Manpyo Hong, "Multiple Pipelined Hash Joins Using Synchronization of Page Execution Time," *Int'l Conf. PDPTA'2000*, Vol. V, pp. 2863-2869, June, 26-29, 2000.

이 규 옥
정보과학회논문지 : 시스템 및 이론
제 28 권 제 1 호 참조

홍 만 표
정보과학회논문지 : 시스템 및 이론
제 28 권 제 1 호 참조