공유 디스크 클러스터에서 친화도 기반 동적 트랜잭션 라우팅

온경오° 이상호 조행래
영 남 대 학 교 컴 퓨 터 공 학 과
{ondal, comman35}@cse.yu.ac.kr hrcho@yu.ac.kr

Affinity-based Dynamic Transaction Routing in Shared Disks Clusters

Kyungoh Ohn° Sangho Lee Haengrae Cho
Dept. of Computer Engineering, Yeungnam University

요 약

공유 디스크(Shared Disks: SD) 클러스터는 온라인 트랜잭션 처리를 위해 다수 개의 컴퓨터를 연동하는 방식으로, 각 노드들은 디스크 계층에서 데이타베이스를 공유한다. SD 클러스터에서 트랜잭션 라우팅은 사용자 트랜잭션이 요청될 경우 이를 실행할 노드를 결정하는 것을 의미한다. 이때, 동일한 클래스에 속하는 트랜잭션들을 가급적 동일한 노드에서 실행시킴으로써 캐쉬 무효화 오버헤드를 최소화할 수 있으며, 이러한 기법을 친화도 기반 트랜잭션 라우팅이라 한다. 한편, 트랜잭션 클래스의 발생빈도는 동적으로 변할 수 있으며, 특정 트랜잭션 클래스가 폭주할 경우 정적인 친화도 기반 트랜잭션 라우팅 정책만으로는 한계가 있다. 본 논문에서는 참조 지역성을 고려하여 동적인 트랜잭션 클래스의 부하를 SD 클러스터의 모든 노드들에 균등히 분배하는 동적 트랜잭션 라우팅 기법을 제안한다.

1. 서론

클러스터는 다수 개의 연결된 노드들이 트랜잭션 처리를 위해 하나의 노드처럼 협력하는 시스템이며, 현재 온라인 트랜잭션 처리, 전자 상거래, 그리고 병렬 데이타베이스 시스템 등의 다양한 분야에 적용되고 있다[5]. 클러스터의 구성방식은 데이타를 액세스하는 방법에 따라 모든 노드에서 디스크를 공유(shared disks: SD)하거나 버퍼와 디스크를 공유하지 않는 방식(shared nothing: SN)으로 분류할 수 있다. SN 클러스터에서 각 노드는 할당된 데이타베이스 분할에 대해서만 직접 액세스가 가능하다. 하지만, SD 클러스터에서는 각 노드들이 디스크 계층에서 전체 데이타베이스를 공유함으로써, 동적 부하분산이 용이하고 새로운 노드의 추가로 인한 확장성의 장점을 가진다. 뿐만 아니라, 급속하게 발전하는 SAN(storage area networks) 기술은 SD 클러스터에 높은 가용성과 데이타 액세스의 융통성을 지원한다[4].

SD 클러스터에서 각 노드들은 최근에 액세스한 페이지를 자신의 지역 버퍼에 캐싱함으로써 디스크 액세스 수와 노드들 간의 메시지 양을 줄여 성능을 높일 수 있다. 그러나 여러 노드에서 항상 최신의 페이지를 사용할 수 있기 위해서 각 노드에 캐싱되어 있는 페이지들 사이의 일관성을 유지해야 한다. 즉, 변경된 페이지는 그 이전 버전을 캐싱하고 있는 다른 노드들의 버퍼에서 무효화를 시켜야 한다[1].

사용자가 요청한 트랜잭션은 트랜잭션 라우터에 의해 선택된 노드에 할당되어 실행된다. 친화도 기반 라우팅(affinity-based routing)은 유사한 데이타 액세스 형태를 가진 트랜잭션들을 동일한 노드에 할당함으로써 지역 버퍼의 히트율을 높이고, 노드들 간의 버퍼 무효화와 로크 동기화에 의한 노드들 사이의 충돌을 줄일 수 있다[3, 6]. 그러나 순수한 친화도 기반 라우팅은 트랜잭션 클래스의 부하가 정적일 때만 그 효과를 기대할 수 있으며, 트랜잭션 클래스의 부하가 동적으로 변할 때는 특정 노드에서 과부하가 발생할 수 있다.

이러한 관점에서 본 논문에서는 트랜잭션을 실행할 노드를 선택하기 위한 동적 트랜잭션 라우팅 기법을 제안한다. 제안한 기법은 기본적으로 친화도 기반 트랜잭션 라우팅 기법을 수행하고, 버퍼 히트율을 고려한 동적 부하분산을 수행한다.

본 논문의 구성은 다음과 같다. 2절에서는 기존에 제안된 트랜잭션 라우팅 기법을 살펴보고, 3절에서는 본 논문에서 제안한 동적 트랜잭션 라우팅 기법을 설명한다. 마지막으로 4절에서 결론 및 앞으로의 연구방향에 대해 논의하기로 한다.

2. 관련 연구

SD 클러스터에서 친화도 기반 라우팅 기법은 우선 트랜잭션들을 데이타베이스 액세스 유형에 따른 트랜잭션 클래스들로 분류하고, 각 트랜잭션 클래스에 노드를 할당한다. 이후 트랜잭션이 요청되면 트랜잭션 라우터는 각 트랜잭션 클래스에 해당된 노드에 트랜잭션을 할당하여 실행한다. 그 결과 이전 트랜잭션에 의해 캐싱된 데이타를 지역 버퍼에서 액세스할 가능성이 높기 때문에,

* 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2002-002-D00151)

트랜잭션의 응답시간을 줄일 수 있다. 그러나, 친화도 기반 라우팅 기법은 시스템의 부하가 정적일 때만 효과를 기대할 수 있다. 즉, 시스템의 부하가 실행 중에 동적으로 변화될 경우 적합하지 않다. 예를 들어, 특정 데이타베이스 분할을 액세스하는 트랜잭션 클래스가 폭주할 경우, 그 트랜잭션 클래스에 할당된 노드는 과부하 상태가 되지만 다른 노드들은 유휴 상태가 된다. 그러므로, 전체 노드의 부하 편차를 최소로 유지해야 하는 클러스터의 부하 분산과 트랜잭션 라우팅 정책의 목표를 만족시키지 못한다[2].

SD 클러스터에서 친화도 기반 트랜잭션 라우팅 정책의 이러한 문제를 해결하기 위하여 트랜잭션 클래스에 동적으로 노드를 할당하는 친화도 기반 동적 부하 분산 기법들이 제안되었다[3]. 동적 트랜잭션 라우팅은 시스템의 부하가 균등히 분산될 경우 친화도 기반 라우팅을 수행하고, 트랜잭션 클래스의 부하가 변할 때 부하를 다른 노드로 분산시킨다. 부하를 분산하는 방법으로는 폭주하는 트랜잭션을 클러스터의 모든 노드에 분산하는 방법[7]과 클러스터의 최소 부하를 가진 노드에 트랜잭션을 보내는 방법이 있다[2]. 그러나, 이러한 방법들은 동적으로 부하를 분산할 수는 있지만, 낮은 버퍼 히트율과 노드들 간의 빈번한 버퍼 무효화로 인해 성능 개선에 한계가 있다. 이런 문제점은 유사한 데이타베이스 분할을 액세스하는 트랜잭션 클래스를 고려하지 않고 전체 시스템에 부하를 분산시킴으로써 발생한다.

Oracle 9*i* Real Application Cluster의 동적 자원 재소유 기법은 특정 데이타베이스 분할을 가장 빈번히 액세스하는 노드가 해당 분할의 로크 정보를 유지하여 지역적으로 로크 연산를 처리함으로써 로킹 오버헤드를 줄일 수 있다[4]. 그러나 동적 자원 재소유는 친화도 기반 트랜잭션 라우팅의 장점을 이용하는 기법으로 트랜잭션의 라우팅 정책은 아니다.

3. 친화도 기반의 동적 트랜잭션 라우팅

기존의 친화도 기반 동적 트랜잭션 라우팅 정책들의 문제점은 특정 트랜잭션 클래스가 폭주할 때 친화도를 고려하지 않고 시스템에서 부하가 가장 적은 노드에 할당하기 때문에 발생한다.

본 절에서는 친화도 기반 동적 트랜잭션 라우팅 알고리즘인 DACA(Dynamic Affinity Cluster Allocation)를 제안한다. DACA의 목표는 두 가지로 요약할 수 있다. 첫째, 버퍼 히트율이 급격하게 감소하지 않는 부하 분산을 수행한다. 둘째, 특정 데이타베이스 분할과 높은 친화도를 가진 여러 트랜잭션 클래스들을 하나의 친화도 클러스터(affinity cluster: AC)[6]로 정의한 후, AC들의 부하 분산을 고려한 라우팅 정책을 제안함으로써 라우팅 오버헤드를 감소시킨다.

3.1 시스템 모델

표 1은 DACA의 라우팅 정책을 위한 매개변수들이다. AC들간의 부하 편차를 최소하기 위해 #AC가 #N보다 작거나 같다고 가정한다[6]. 트랜잭션 라우터는 라우팅 매개변수들을 유지하여 사용자 트랜잭션들을 여러 노드에

표 1. 라우팅 매개변수

매개변수	설명
$\#AC$	AC의 수
$\#N$	노드의 수 ($\#N \geq \#AC$)
$Mem(N_p)$	노드 N_p의 메모리 크기
$HotSet(AC_q)$	AC_q에 액세스되는 핫셋의 크기
$\#T(N_p)$	N_p에서 실행되는 트랜잭션의 수
$\#T(AC_q)$	AC_q의 실행되고 있는 트랜잭션의 수
$R(AC_q)$	AC_q에 할당된 노드의 집합
$Card(R(AC_q))$	AC_q에 할당된 노드의 수
$R^{-1}(N_p)$	N_p에 할당된 AC의 집합
$Card(R^{-1}(N_p))$	N_p에 할당된 AC의 수
$\overline{L}(AC)$	AC의 평균 부하 ($\sum_{i=1}^{\#AC}(\#T(AC_i)/Card(R(AC_i))) / \#AC$)
$\overline{L}(N)$	노드의 평균 부하 ($\sum_{i=1}^{\#N}(\#T(N_i))/\#N$)

할당한다. AC_i의 트랜잭션이 트랜잭션 라우터에 의해 노드 N_j에 할당될 때 $\#T(AC_i)$와 $\#T(N_j)$값을 증가하고, 트랜잭션이 완료되어 라우터에 알려지면 이 값들을 감소한다.

트랜잭션 라우터는 트랜잭션 라우팅을 위해 라우팅 함수(**R**)를 사용하며, **R**은 각 AC에 할당된 노드들의 집합을 의미한다. $\mathbf{R}(AC_i)$가 여러 노드들을 포함할 경우, AC_i에 해당되는 트랜잭션들은 라운드 로빈 방식으로 관련 노드들에게 라우팅 되어 진다. $\mathbf{R}^{-1}$은 **R**의 역함수로 각 노드에 할당된 AC를 의미한다. 초기에 $\mathbf{R}(AC_i) = \{N_i\}$로 초기화되며, 시스템 부하에 많은 부분을 차지할 것으로 예상되는 AC에 대해서 더 많은 노드들을 할당함으로써 초기 설정을 최적화 할 수 있다.

동적 라우팅을 위해 DACA는 과부하를 AC 과부하와 노드 과부하로 구분한다. AC 과부하는 특정 AC의 트랜잭션들이 폭주하는 것을 의미하고, 노드 과부하는 특정 노드가 여러 개의 AC들에 할당되었을 때 노드에서 실행되는 트랜잭션 수가 평균을 초과한 상태를 의미한다. DACA는 AC와 노드의 상태에 따른 라우팅 정책을 수행하며 라우팅 정책을 변경해야 할 AC와 노드의 상태에 대해서 다음과 같이 정의한다.

정의 1. AC 과부하: $Card(\mathbf{R}(AC_q)) < \#N$ 이고, $\#T(AC_q)/(Card(\mathbf{R}(AC_q))+1) \geq \overline{L}(AC)$ 일 경우 AC_q는 AC 과부하

정의 2. 노드 과부하: $Card(\mathbf{R}^{-1}(N_p)) > 1$ 이고, $\#T(N_p) \geq \overline{L}(N) \times \alpha\ (\alpha \geq 1)$ 일 경우 N_p는 노드 과부하

정의 3. AC 부하미달: $Card(\mathbf{R}(AC_q)) > 1$ 이고, $\#T(AC_q)/(Card(\mathbf{R}(AC_q))-1) < \overline{L}(AC)$ 일 경우 AC_q는 AC 부하미달

3.2 동적 트랜잭션 라우팅

DACA는 과부하 상태에 따라 각 노드의 부하를 분산시킨다. 만약 AC_q가 과부하라면, AC_q에 노드를 추가 할

당하여 $\mathbf{R}(AC_q)$를 확장하며 이 정책을 노드 확장(node expansion)이라 한다. 만약 AC 과부하가 없고 노드 N_p가 과부하일 경우 AC 분산(AC distribution)을 수행하여 N_p에 할당된 일부 AC를 다른 노드로 분산시킨다. 마지막으로 AC_q가 부하미달이 되면 노드 축소(node reduction)에 따라 AC_q에 할당된 노드를 축소한다.

3.2.1 노드 확장

AC_q가 과부하 상태이고 $\mathbf{R}(AC_q)$가 $\{N_i\}$라고 가정했을 때, AC_q의 새로운 트랜잭션을 N_i에 라우팅 한다면 응답 시간이 증가하므로, 트랜잭션 라우터는 $\mathbf{R}(AC_q)$에 노드를 추가시켜 N_i의 부하를 분산시킨다. 여기서 추가될 후보 노드로 클러스터에서 최소 부하 노드가 선택된다. 최소 부하 노드를 N_k라고 하면 $\mathbf{R}(AC_q)$는 $\{N_i,N_k\}$로 확장되고, AC_q의 트랜잭션들은 라운드 로빈 방식으로 N_i 또는 N_k에 라우팅된다. 만약 N_k에 이미 할당된 AC_k가 있을 경우, AC_k를 다른 노드에 할당한다. AC_k가 부하미달이면 노드 축소 정책에 의해 $\mathbf{R}(AC_k)$에서 N_k를 제외시키고, N_k를 AC_q에 할당한다. 노드 축소 정책은 3.2.3에서 설명한다. 만약 AC_k가 부하미달이 아니라면 3.2.2에서 설명하는 AC 분산 정책과 유사한 방식으로 AC_k에 다른 노드를 할당한다. 결국 AC_k의 부하미달 여부에 관계없이 N_k는 AC_q에 할당되고 AC_k는 다른 노드에 할당된다.

DACA의 노드 확장에서 중요한 점은 각 노드의 부하 편차가 급격하지 않은 범위에서 AC에 할당되는 노드의 수를 최소로 유지하는 것이다. 이것은 과부화된 AC를 즉시 모든 노드에 할당하는 기존의 라우팅 정책과는 구분된다[2, 7]. AC의 트랜잭션들이 AC에 할당된 여러 개의 노드에서 라운드 로빈 방식으로 라우팅되므로 각 지역 버퍼는 비슷한 페이지를 캐싱하게 되고, 그 결과 노드들간의 많은 페이지의 중복과 많은 버퍼 무효화가 발생한다. AC의 트랜잭션이 더 많은 노드에서 실행된다면, 버퍼 무효화는 동적 부하 분산의 한계 요소로 작용할 것이다.

3.2.2 AC 분산

$\mathbf{R}^{-1}(N_p)$에 속하는 AC들 중에서 노드 N_p가 과부하 상태이고 $\mathbf{R}^{-1}(N_p)$가 2이상일 때, AC_s와 AC_l을 각각 $\#T(AC_s)$가 최소, $\#T(AC_l)$이 최대값인 AC라고 가정한다. 만약 AC가 할당되지 않은 노드 N_k가 있다면, 트랜잭션 라우터는 $\mathbf{R}(AC_l)$를 $\{N_k\}$로 수정한다. N_k에 할당된 AC_k가 부하미달인 경우에도 $\mathbf{R}(AC_l)$를 $\{N_k\}$로 수정한 후, AC_k에 대해 노드 축소 정책을 수행한다.

모든 노드에 AC가 할당되어 있고 부하미달인 AC도 존재하지 않을 경우, 트래잭션 라우터는 과부하가 아닌 노드 N_k에 AC_s를 할당한다. AC_k가 N_k에 이미 할당된 AC라고 가정할 때, N_k는 AC_k와 AC_s에 액세스되는 페이지들을 지역 버퍼에 캐싱하게 된다. 다른 데이타베이스 분할을 액세스하는 AC들이 하나의 노드에서 실행될 경우 지역 버퍼가 두 AC들의 HotSet을 캐싱할 만큼 크지 않다면 빈번한 캐쉬 미스가 발생한다. 따라서, 빈번한 캐쉬 미스를 줄이기 위해 AC_s는 $HotSet(AC_k) + HotSet(AC_s) < Mem(N_k)$를 만족하는 N_k에 할당되어야 한다. 만약 이것을 만족하는 노드가 두 개 이상 존재한다면 트랜잭션 라우터는 AC_s를 그들 중 최소 부하 노드에 할당한다. 조건을 만족하는 노드가 존재하지 않는다면, 전체 노드 중 최소 부하 노드에 AC_s를 할당한다.

3.2.3 노드 축소

특정 AC가 부하미달일 경우 AC에 할당된 노드를 축소시켜 다른 AC에 노드를 할당할 수 있도록 한다. AC_q가 부하미달이고 $\mathbf{R}(AC_q)$가 $\{N_i,N_k\}$라고 가정했을 때, 트랜잭션 라우터가 $\mathbf{R}(AC_q)$로부터 제외될 노드를 선택한다. 이때, 다른 AC에 중복 할당된 노드가 존재한다면 그 노드를 제외하고, 그렇지 않을 경우 임의의 노드를 제외한다.

노드 축소 정책은 노드 확장 정책과 같이 AC 과부하가 되지않는 최소 노드를 AC에 할당함으로써 DACA의 첫 번째 목표를 만족시킨다.

4. 결론 및 향후과제

SD 클러스터에서 부하의 폭주, 트랜잭션 비율의 변화 혹은 노드의 고장으로 인해 부하가 동적으로 변할 때, 순수한 친화도 기반 트랜잭션 라우팅 정책보다 부하를 고려한 동적 트랜잭션 라우팅 정책이 필요하다. 본 논문에서 제안한 DACA는 부하가 동적으로 변화될 때 기존의 친화도 기반 동적 트랜잭션 라우팅 알고리즘보다 성능이 뛰어나다. 이것은 AC 과부하가 되지 않는 범위에서 최소의 노드를 할당함으로써 참조 지역성을 높이고, 로크 요청과 버퍼 무효화의 오버헤드를 줄임으로써 가능하다. 본 논문의 향후과제는 제안한 알고리즘의 성능을 정량적으로 분석하는 것이다.

5. 참고문헌

[1] H. Cho, "Cache Coherency and Concurrency Control in a Multisystem Data Sharing Environment," *IEICE Trans. Information and Syst.* E82-D(6) (1999) 1042-1050.

[2] S. Haldar and D.K. Subramanian, "An Affinity-based Dynamic Load Balancing Protocol for Distributed Transaction Processing Systems," *Performance Evaluation* 17(1) (1993) 53-71.

[3] C.N. Nikolaou, M. Marazakis, and G. Georgiannakis, "Transaction Routing for Distributed OLTP Systems: Survey and Recent Results," *Info. Sciences* 97(1-2) (1997) 45-82.

[4] *Oracle 9i Real Application Cluster - Concepts*, Oracle Part No. A89867-02 (2001).

[5] M. Yousif, "Shared-Storage Clusters," *Cluster Comp.* 2(4) (1999) 249-257.

[6] P. Yu and A. Dan, "Performance Analysis Clustering on Transaction Processing Coupling Architecture," *IEEE Trans. Knowledge and Data Eng.* 6(5) (1994) 764-786.

[7] P. Yu and A. Dan, "Performance Evaluation of Transaction Processing Coupling Architectures for Handling System Dynamics," *IEEE Trans. Parallel and Distributed Syst.* 5(2) (1994) 139-153.