구조적 피어-투-피어 시스템에서 동적 피어 참여 기반의 효율적인 로드 밸런싱

송진우° 최연오 양성봉
연세대학교 컴퓨터과학과
{fantaros°, yoenoz, yang}@cs.yonsei.ac.kr

An Efficient Load Balancing with the Dynamic Participation of Peers in the Structured Peer-to-Peer System

Jinwoo Song°, Yeonoh Choi, Sungbong Yang
Dept of Computer Science, Yonsei University

요 약

최근 분산 해쉬 테이블(Distributed Hash Table)을 이용한 구조적 P2P(peer-to-peer) 시스템에서 로드 밸런싱에 관한 연구가 활발하게 진행되고 있다. 일반적으로 P2P 시스템에서의 피어들은 빈번하게 참여하는 동시에 탈퇴하기도 하기 때문에 로드의 불균형이 심화되는 문제점이 있다. 본 논문에서는 구조적 P2P 시스템에서 피어의 참여와 탈퇴 시 발생하는 가상서버의 이동을 효과적으로 관리하는 기법인 PALB(Peer Activity-aware Load Balancing)를 제안한다. 제안하는 PALB는 피어의 참여, 탈퇴 시에 발생하는 로드 불균형을 해결하기 위하여 적절한 수의 가상서버를 가장 적합한 피어에게 분산시킨다. 또한 이전 연구에서 제안한 OLAB(Object Lifetime-aware Load Balancing)와 함께 사용하여 로드 밸런싱 성능을 향상시킬 수 있다. PALB의 성능 평가를 위하여 동적 P2P 시스템 환경을 구성하였고, 피어와 오브젝트의 랜덤 데이터 셋을 이용하여 시뮬레이션을 수행하였다. 시뮬레이션 결과 PALB가 기존의 시스템에 비하여 더 적은 비용으로 더 나은 로드 밸런싱을 수행함을 확인하였다. 또한 OLAB와 동시에 사용하였을 경우, 매우 뛰어난 성능을 보임을 확인하였다.

1. 서 론

최근 분산 해쉬 테이블(Distributed Hash Table)을 이용한 구조적 P2P(peer-to-peer) 시스템에서 로드 밸런싱에 관한 연구가 활발하게 진행되고 있다. 구조적 P2P 시스템들(Chord[1], Pastry[2], Tapestry[3], CAN[4]) 등은 분산 해쉬 테이블 개념을 사용한다. 분산 해쉬 테이블을 사용한 P2P 시스템들에서 각 오브젝트와 피어는 해쉬 함수에 의해 고유한 식별자(identifier)를 갖게 된다.

이러한 시스템들은 여러 가지 이유로 각 피어들이 서로 다른 로드(load)를 갖는 상황을 맞이하게 된다. 첫 번째로, 각 피어와 오브젝트의 식별자가 시스템이 유지하는 식별자 공간에 랜덤하게 배치되어 식별자 공간이 불균일하게 분할되기 때문이다[5]. 두 번째로, 피어와 오브젝트의 식별자가 균일하게 배치되어도 각 피어의 능력(저장 공간이나 네트워크 대역 등)이 다르다면 시스템이 불균형하게 된다. 세 번째로, P2P 시스템의 특성상 피어들이 빈번하게 참여하고 탈퇴하기 때문에 로드 불균형이 심화 될 수 있다.

본 논문에서는 구조적 P2P 시스템에서 피어의 참여와 탈퇴 시 발생하는 가상 서버(virtual server)의 이동을 효과적으로 관리하는 기법인 PALB(Peer Activity-aware Load Balancing)를 제안한다. 제안하는 PALB는 피어의 참여, 탈퇴 시에 발생하는 로드 불균형을 해결하기 위하여 적절한 수의 가상서버를 가장 적합한 피어에게 분산시킨다. 또한 이전 연구[6]에서 제안한 OLAB(Object Lifetime-aware Load Balancing)와 함께 사용할 수 있어서 로드 밸런싱의 성능을 향상시킬 수 있다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하며, 3장에서는 OLAB 및 새롭게 제안하는 PALB에 대하여 기술한다. 4장에서는 실험을 통한 성능 평가 결과를 설명하고 5장에서는 결론 및 향후 과제를 제시한다.

2. 관련 연구

P2P 로드 밸런싱에는 크게 두 가지 방법이 있다. 첫 번째로 가상 서버 개념을 로드 밸런싱에 응용한 연구가 있다[7]. 로드 밸런싱은 가상 서버들을 로드가 초과된 물리적 피어로부터 로드의 여유가 있는 피어에게 분산시킴으로써 수행된다. 두 번째 로드 밸런싱 방법으로 "Power of two choices" 패러다임을 이용한 연구가 있다[8]. 각 오브젝트들은 d개의 다른 해쉬 함수를 사용하여 d개의 서로 다른 식별자를 할당받고, 그 중 가장 적은 로드를 가지는 피어에 오브젝트의 키를 저장한다.

본 논문에서는 첫 번째 방법인 가상 서버 기법을 사용한 P2P 로드 밸런싱을 이용한다. 현재 많은 로드 밸런싱 알고리즘들이 제안되었지만 대부분 정적인 시스템 환경을 가정한다[1][5][8]. [6]에서는 오브젝트들은 동적으로 참여하고 탈퇴하지만 피어들의 참여와 탈퇴는 없는 환경을 가정한다. [9]는 피어와 오브젝트가 모두 동적으로 참여하고 탈퇴하는 환경을 가정하였으나 주기적인(periodic) 로드 밸런싱과 긴급(emergency) 로드 밸런싱을 병행한다.

3. 제안하는 로드 밸런싱 알고리즘

3.1 동적 피어의 참여와 탈퇴

분산 해쉬 테이블을 이용하는 구조적 P2P 시스템에서 피어의 참여와 탈퇴는 피어가 책임지게 되는 로드의 이동을 발생시킨다. 어떤 피어가 시스템에서 탈퇴할 때는 자신이 가지고 있던 로드를 다른 피어에게 넘겨주어야 한다. 시스템에 참여할 시에도 피어는 고유한 식별자를 할당 받게 되고 대응하는 로드를 다른 피어로부터 넘겨받게 된다. 이러한 현상은 가상 서버를 사용하는 시스템에서 필수적으로 발생한다. 임의의 피어가 탈퇴할 경우 가지고 있던 모든 가상서버를 탈퇴시켜야 하고 참여할 때는 시스템이 정한 수만큼의 가상서버를 생성하여 그 가상서버들을 가상 서버 레벨에서 참여시키게 된다.

문제는 이러한 로드의 이동이 시스템을 구성하는 피어들의 로드의 균형을 무너뜨리는 요인이 된다는 점이다. 탈퇴 시에는 소유하고 있던 로드가 그것을 받게 되는 피어의 로드 상황과는 무관하게 이동되며 참여 시에는 로드를 주는 피어의 상황을 고려하지 않고 무작위의 피어로부터 받게 된다.

[1]의 경우, 피어가 탈퇴할 때 자신의 후행하는 식별자를 가지는 피어에게 자신의 로드를 이동시키다. 반대로 피어가 참여할 때는 필요한 가상 서버 수만큼의 무작위의 식별자를 만들어 각각 가상 서버에게 대응시켜 참여시킨 후 선행하는 식별자를 가지는 피어로부터 로드를 받는다.

이처럼 탈퇴 시에 자신의 로드가 이미 로드가 초과된 피어에게 이동되고 참여 시에는 아직 로드가 여유 있는 피어로부터 로드를 받는 경우가 발생할 수 있다. 즉 로드가 초과된 피어는 더 많은 로드를 갖게 되고 로드가 여유 있는 피어는 더 적은 로드를 갖게 될 수 있다. 이와 같은 피어의 참여, 탈퇴에 의한 로드 불균형의 심각성은 4장의 실험결과에서 자세히 알 수 있다.

3.2 PALB

PALB는 피어의 참여, 탈퇴 시에 효율적인 피어를 선택하여 로드의 이동을 하게 한다. 피어가 탈퇴 시에는 로드를 받게 될 피어를 가능한 로드가 여유 있는 피어로 선택하고 참여 시에는 되도록 로드가 초과된 피어로부터 로드를 받도록 한다.

피어가 탈퇴하는 과정에서 로드가 초과된 피어가 발생하지 않도록 하기 위해서는 기본적으로 가상 서버를 탈퇴시키지 않고 로드가 여유 있는 피어에게 이동시키면 된다. 즉, 탈퇴하는 피어가 *일대다 방식*[5]으로 로드 밸런싱을 수행하면 된다. 그러나 이렇게 하기 위해서는 로드가 여유 있는 피어들에 대한 정보가 항상 유지되어야 하는 부담이 생기고 피어의 참여와 탈퇴가 일어날 때마다 네트워크에 가상 서버의 수가 증가해서 검색의 지연이 일어난다[1].

따라서 PALB는 가급적이면 단순히 탈퇴시켜도 로드의 초과가 발생하지 않는 가상 서버는 탈퇴시키고 나머지에 대해서만 로드가 여유 있는 피어들에게 이동시킨다. 이에 대한 의사코드는 다음과 같다.

```
Join()
  Information ← RandomInformation();
  Get the excessive number of virtual servers from the
  Information;
  numNode ←the number of virtual server to create;
  Participate numNode virtual servers into the system;
  Inform numNode to the Information;
  For (TH_PROBE)
      peer ← RandomPeer();
      HeavyToLight(peer, this);
```

그림1. PALB 피어 참여 의사코드

```
Leave()
  For (virtual server v in this peer)
     If (Other peer's load is not overloaded by v's
     leave)
        v leaves the system;
  For (TH_PROBE)
        peer ← RandomPeer();
        HeavyToLight(this, peer);
  Leave all remaining virtual servers;
  Information ← RandomInformation();
  Inform the number of moved virtual servers to the
  Information;
```

그림2. PALB 피어 탈퇴 의사코드

위의 의사 코드에서 RandomInformation() 과 RandomPeer()는 각각 랜덤 *Information*과 피어를 선택하는 함수이며 HeavyToLight(*h*, *l*)는 로드가 초과된 피어 *h*에서 로드가 여유 있는 피어 *l*에게 가상 서버를 이동시키는 함수이다. TH_PROBE는 탐색 횟수를 정하는 임계값이다.

4. 실험 및 분석

4.1 성능 평가 기준

로드 밸런싱의 성능을 평가하기 위해서 다음 두 가지의 평가 기준을 사용한다.

1) 이동된 로드: 주기적 로드 밸런싱 수행 시 이동된 가상 서버의 로드의 총합이다. 로드 밸런싱에 드는 비용은 이동된 로드에 비례한다. 알고리즘의 효율적인 면을 측정하기 위해 필요한 기준이다. 이 값이 작을수록 효율적인 알고리즘이라 할 수 있다.

2) 99.9 피어 활용도: 실험을 수행하는 시간동안 임의의 시각 t에 모든 피어의 활용도 $Utility_p$를 측정해서 가장 높은 값을 가지는 피어의 활용도이다. 이 값이 작을수록 로드 밸런싱이 잘 되었다고 할 수 있다. 피어 p의 활용도 $Utility_p$는 $Load_p / Capacity_p$이다. (이때 $Load_p$, $Capacity_p$는 각각 피어 p의 로드와 용량이다.)

4.2 실험 환경

실험 환경은 [6]과 동일하며 피어의 P2P 시스템 참여, 탈퇴의 빈도를 조절하는 피어 활동 임계값을 추가하였다. 임계값 시간동안 피어가 아무런 행동을 하지 않으면 P2P 시스템을 탈퇴한 것으로 설정하는 인자로, 값이 클수록 더 많은 피어들이 자주 참여, 탈퇴하게 된다. 본 논문에서는 가상 서버를 이동시켜서 로드 밸런싱을 하는 방법 중 *다대다 방식*[5](Many-to-many, MM)을 구현하여 사용한다. 다대다 방식은 d개의 인포메이션(*Information*)과 한 개의 전역 풀(global pool)을 사용한다.

4.3 실험 결과

그림 3과 4는 피어 활동 임계값을 변화시킴으로써 피어들이 참여, 탈퇴하는 빈도수를 조절하고, 로드 밸런싱 알고리즘들의 성능을 비교한다. 기본 다대다 방식과 OLAB는 피어의 동적 활동이 활발할수록 99.9 피어 활동도 누적과 이동된 로드 누적 측면에서 성능이 급격히 떨어진다. 그 이유는 피어 참여, 탈퇴에 따른 가상 서버의 이동에 의해서 로드 불균형이 심화되고, 그러한 상태에서의 로드 밸런싱은 더 많은 로드 이동이 필요해지기 때문이다.

PALB 방식을 적용할 경우 피어 활동 임계값의 변화에 따라 99.9 피어 활동도 측면에서 최대 25%, 이동된 로드 측면에서

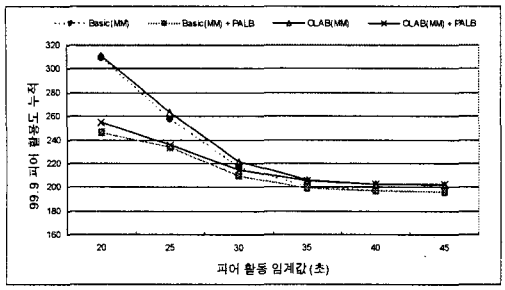

그림 3. 피어 활동 임계값에 따른 99.9 피어 활용도 누적

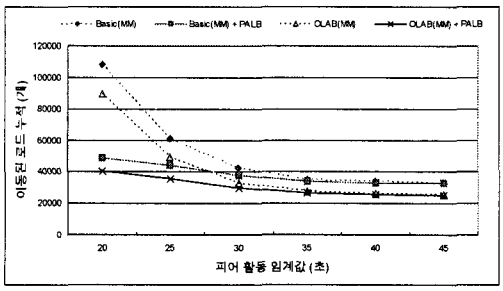

그림4. 피어 활동 임계값에 따른 이동된 로드 누적

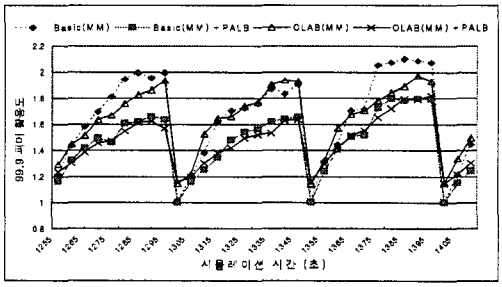

그림5. 시뮬레이션 시간에 따른 99.9 피어 활용도
(피어 활동 임계값 = 25)

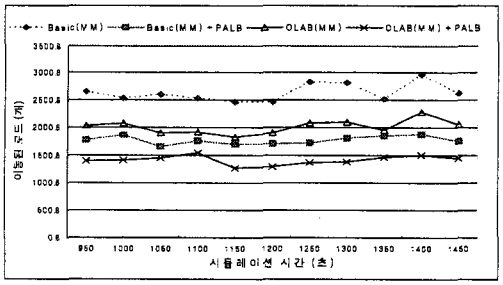

그림5. 시뮬레이션 시간에 따른 이동된 로드
(피어 활동 임계값 = 25)

는 최대 121%의 성능 향상이 있었다. OLAB와 PALB를 같이 적용한 경우에는 각각 최대 21%, 168%의 성능 향상이 있었다.

그림 4과 5는 피어 활동 임계값이 25일 때의 시스템이 안정화 된 이후의 성능을 보여준다.

5. 결 론

본 논문에서는 피어가 동적으로 참여하고 탈퇴하는 환경에서 가상서버의 이동을 효과적으로 관리하여 로드 밸런싱 성능을 향상시키는 PALB 기법을 제안하였다. 제안한 기법은 선택적으로 가상 서버를 생성 및 제거하고, 적절한 피어에게 가상 서버를 이동시킴으로써 로드 밸런싱의 성능을 높였다. 또한 OLAB와 사용하였을 경우 이동된 로드의 수를 대폭 감소시킬 수 있었다. 향후 심화된 동적 환경과 다양한 데이터 분포에 대해서 실험을 확장 할 계획이다.

6. 참고문헌

[1] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. ACM SIGCOMM, pp.149-160, 2001.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems," Proc. International Conference on Distributed Systems Platforms, pp.329-350, 2001.

[3] B. Zhao, J. Kubiatowicz, A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide Area Location and Routing," Technical Report UCB/CSD-01-1141, Computer Science Division, Univ. of California, Berkeley, 2001.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in Proc. ACM SIGCOMM, 2001.

[5] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," Proc. Second Int'l Workshop Peer-to-Peer Systems, pp.68-79, 2003.

[6] 최연오, 송진우, 양성봉, "구조화된 동적 P2P 시스템에서의 로드 변화 예측을 통한 효율적인 로드 밸런싱," 한국정보과학회 추계학술대회, Vol 32, No 2, pp.250-252, 2005

[7] F. Dabek, M. Frans Kaashoek, D. Karger, R. Morris, I. Stoica, "Wide-area Cooperative Storage with CFS," Proc. 18th ACM Symp. Operating Systems Principles (SOSP), pp.202-215, 2001.

[8] J. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables," Proc. Second Int'l Workshop Peer-to-Peer Systems, pp.80-87, 2003.

[9] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," Proc. IEEE INFOCOM, Vol. 4, pp.2253-2262, 2004.