Chord 시스템에서의 협력을 통한 로드 밸런싱 기법

강영상[O] 염헌영
서울대학교 전기·컴퓨터 공학부
{ewdriver[O], yeom}@dcslab.snu.ac.kr

Coordinated Load-Balancing in Chord System

Youngsang Kang[O] Heonyoung Yeom
School of Computer Science and Engineering, Seoul National University

요 약

Peer-to-Peer Network에서의 object 위치 정보의 배치 및 라우팅 방법인 Chord 시스템은 node ID와 object ID 생성에 있어서 SHA-1과 같은 해쉬 함수를 사용한다. Chord는 object 위치 정보 관리에 있어서 해쉬에 의해 저절로 load-balancing을 기대하나 실험에 의하면 노드별 load에는 상당한 편차가 있음이 나타난다. 그러므로 이 논문에서는 각 node간 협력을 통하여 보다 효과적인 load-balancing을 제공하는 CLCS 기법을 제안한다.

1. 서 론

응용 프로그램들이 넓은 범위로 대형화됨에 따라 그것이 마주치게 되는 복잡한 문제들을 해결하기 위해 분산 미들웨어 계층이 필요시 된다. 특히 분산 객체의 배치 및 라우팅 시스템에 대한 최근 연구에서는 overlay network상에서 배치 및 라우팅 방법을 결합하여 분산 오퍼레이션, 확장성, 결함 내성, 로드 밸런싱을 제공하는 방법을 연구중이다.

기존의 Peer-to-Peer overlay network에 대한 연구들(CAN[1], Pastry[2], Chord[3], Tapestry[4], ...)에서는 파일과 같은 data object들에 대한 배치와 라우팅 방법을 제공한다. 즉, object들이 저장되어 있는 위치 정보들이 overlay network상의 각 노드에 분산되기 때문에, 새로운 object의 위치 정보를 추가하기 위해서는 그것을 어떤 노드에 저장할 것인지 결정하는 방법인 **배치** 방법과, 어떤 object가 저장된 위치를 알기 위해 그것의 위치 정보를 갖고 있는 노드를 찾기 위한 **라우팅** 방법이 그것이다. 일반적으로 object의 위치 정보가 어떤 노드에 보관될 지는 object 이름의(예, 파일 이름) 해쉬값에 따라 정해지고, 해쉬의 성질에 따라 각 위치 정보들은 overlay network 상에서 균일한 분포로 pseudo-random한 위치에 유지될 것을 기대할 수 있다. 그러므로 해쉬의 성질만으로 object 위치 관리에 있어서 로드 밸런싱 효과를 얻을 수도 있다.

그러나, 실험([그림 1])에 의하면 각 노드가 관리하게 되는 위치 정보의 양은 상당한 편차를 가지고 있음을 알 수 있다. 그것의 최소양과 최대양을 비교하면 수 십 배 이상 되어 해쉬의 성질에만 의존하는 배치 및 라우팅 방법으로는 효과적인 로드 밸런싱이 이루어지기 어렵다. 그래서 Chord 시스템에서는 가상 노드 개념을 도입하여 이러한 문제를 해결하려고 하고 있으나 가상 노드로 인해 각 노드가 관리해야 하는 라우팅 테이블의 크기가 커지고, stabilize를 위한 트래픽 양이 증가하게 된다. 그러므로, 보다 효과적인 로드 밸런싱을 위해서는 노드들 간의 별도의 협력이 필요하다. 이 논문에서는 Chord 시스템의 각 노드 간의 협력을 통한 효과적인 로드 밸런싱 방법인 CLCS 기법을 제시하고 있다.

이 논문의 구조는 다음과 같다. 2절에서는 Chord 시스템에 대한 간략한 소개를 하고 있고, 3절에서는 이 논문에서 제시하는 CLCS 기법에 대한 설명을 하고 있다. 4절은 simulation을 통해 CLCS의 효과를 보여주고 있고, 5절에서 결론을 맺고 있다.

2. Chord System

2.1 Naming

Chord에서는 노드와 object에 m-bit identifier를 할당하기 위해 Consistent Hashing[5]의 변형을 사용한다. 그리고 이 object들의 위치 정보는 (K, V)와 같은 쌍으로 표현될 수 있다. 여기에서 K는 object의 이름을 나타내고, V는 object의 위치 정보를 뜻한다. 각 노드는 그것의 IP 주소를 SHA-1과 같은 해쉬 함수로 해쉬하여 만들어진 m-bit nodeID를 할당받고, objectID=hash(K)로 objectID가 정해지고, 위치 정보 (K, V)가 저장되는 위치는 objectID 값에 의해 결정된다.

2.2 Location

노드는 overlay network의 identifier 공간 상에서 자신의 nodeID에 해당하는 부분에 위치하게 된다. ID들은 ID modulo 2^m으로 원형 identifier 공간을 형성한다. 이 공간에서 노드 n_2의 identifier가 n_1의 identifier 바로 이후에 있을 때 n_2를 n_1의 successor라고 하고, n_1를 n_2의 predecessor라고 한다.

각각의 위치 정보 (K, V)는 identifier 공간에서 nodeID가 objectID와 같거나 크면서 첫 번째인 노드에 할당된다. 그것이 object K에 대한 것이라고 할 때 이러한 노드를 K의 successor 노드라 하고 successor(K)로 표현한다. 즉, 노드 n_1, n_2에 대해서 n_2가 n_1의 successor라 하고, objectID $\in (n_1.id, n_2.id]$이라고 하면 이 object의 위치 정보는 n_2에 저장된다.

3. Coordinated Load-balancing in Chord System

Chord 시스템의 각 노드는 관리해야 하는 objectID의 구간을 갖고 있다. 노드는 이 구간에 속하는 object 위치의 리스트를 유지한다. 그 구간은 노드의 ID, predecessor의 ID에 의해 결정되고, 노드 n의 predecessor가 p라고 할 때, 구간은 (p.id, n.id]이 된다. 여기에서 p.id, n.id는 각각 p과 n의 ID를 말한다. objectID와 nodeID가 SHA-1과 같은 해쉬 함수를 통해서 생성되기 때문에 그 분포가 균일하고, 각 노드가 관리해야 하는 object 위치의 개수도 노드마다 상당히 유사할 것으로 기대되나 실험에 의하면 [그림 1]과 같이 상당한 편차를 보임을 알 수 있다. 이 그림은 각 노드별로 할당된 object의 개수를 plotting한 것이다. 그러므로, 단순히 해쉬의 성질을 이용한 성능을 기대할 것이 아니라, 이러한 편차를 줄이기 위해 노드들간의 조정을 통한 직접적인 load-balancing이 필요함을 알 수 있다.

이 논문에서는 Chord system 상의 노드들의 효과적인 load-balancing을 위해 CLCS(Coordinated Load-balancing in Chord System) 기법을 제시한다. 이것은 기본적으로는 Chord system과 동일하나 Load balancing을 위한 추가적인 procedure를 갖는다. 그것은 다음 절에 설명하는 Periodic Coordinated Load-balancing이다.

CLCS 기법에서도 Chord system에서와 같은 배치 및 라우팅 방법을 사용한다. routing의 correctness를 위해 Chord에서의 invarient를 유지한다. 그리고, 라우팅의 효율을 위해서 각 노드들은 finger table을 갖고 있고, successor node에 대한 정보와 finger table이 up-to-date하도록 stabilization process 역시 주기적으로 수행된다.

CLCS에서는 다음과 같은 가정을 하고 있다.

· 가정 : node는 자신의 nodeID를 임의로 설정할 수 있다.

이것은 CLCS가 load-balancing을 위해 자신의 ID를 변경하기 때문이다. Chord 뿐만이 아니라 Pastry, Tapestry등과 같은 다른 Peer-to-Peer system에서도 nodeID를 생성하기 위해 SHA-1과 같은 해쉬 함수를 사용하는데, 이것은 nodeID들이 ID space상에서 고른 분포를 갖고 중복되는 nodeID가 생성되지 않도록 하기 위한 것이다. 필요에 의해 nodeID를 임의로 설정하는 것은 가능하다.

3.1 Periodic Coordinated Load-balancing

각 노드는 주기 P마다 자신의 successor node와 load balancing을 시도한다. 이때, 노드 n은 [표 1]의 *SendLoadBalanceREQ(s)*를 호출하고, REQ message를 받은 successor s는 *SendLoadBalanceREPLY(n)*를 실행하여 REPLY message를 n에게 보낸다.

각 node가 가지고 있는 object location의 개수를 load라고 부르자. 즉 n.load와 s.load는 n과 s가 가지고 있는 object location의 개수이다. 각각이 가지고 있는 위치 정보 리스트를 n.list, s.list라고 하자. SendLoadBalanceREQ()에서 n은 s에게 자신의 load를 알려주고, s에게 받은 object 위치 정보 리스트를 자신의 리스트에 추가한다. 그리고, 그 object의 ID 중에 마지막에 있는 것, 즉 ID space circle 상에서 가장 이후에 있는 것으로 nodeID를 재설정한다.

REQ message를 받은 successor s는 자신의 load와 n의 load를 비교하여 s.load가 n.load보다 큰 경우에는 load balancing이후 두 node의 load가 같아지도록 조절한다. 이 때 s.load와 두 load의 평균 1/2 (n.load + s.load)의 차이인 Move만큼 object location을 n에게 전달하게 된다. 그러기 위해 s가 관리하고 있는 object location list인 O_1, O_2, ..., $O_{s.load}$ 중 objectID가 선행하는 Move개를 선택하여 n에게 보내고 s.list에서 그것들을 제거한다.

이와 같이 system상의 각 node들은 주기 P마다 SendLoadBalanceREQ()를 호출함으로써 successor와 load balancing을 시도하게 되고, 시간이 지남에 따라서 편중된 load가 분산될 것을 기대할 수 있다.

4. Simulation 결과

CLCS와 Chord 시스템에서의 load-balancing 효과를 비교하기 위해 simulation을 수행 하였다. 시스템의 노드 개수는 4000개, object 개수는 200000개, CLCS 주기는 10분, 노드가 join하고 object가 전부 insert된 후로부터 2시간이 지났을 때 각 노드가 유지하는 object의 위치 정보 개수를 구하였다. [그림 1], [그림 2]에서 보는 바와 같이 original Chord에서는 각 노드에 할당된 object 개수에 많은 편차가 보이나 CLCS에서는 편차가 상당히 줄어든 것으로 나타난다.

```
SendLoadBalanceREQ(s)
  Sends REQ message to successor s
  Waits for reply
  Receives MoveSet from s
  n.list ← n.list ∪ MoveSet
  Let ID be the last element in MoveSet
  n.id ← ID

SendLoadBalanceREPLY(n)
  If n.load ≥ s.load then
    Sends to n REPLY message with ϕ where ϕ means an empty set.
  Else
    Let s.list = {O1, O2, ... Os.load}, Oi ∈ (n.id, Oi+1) where 1 ≤ i ≤ s.load
    Let Os.load+1 = s.id
    Move ← ⌈1/2 (s.load – n.load)⌉
    Sends REPLY message to n with MoveSet = {O1, O2, ... Os.Move}
    s.list ← s.list – MoveSet
  end if
```

표 1. **Load Balancing Procedure**

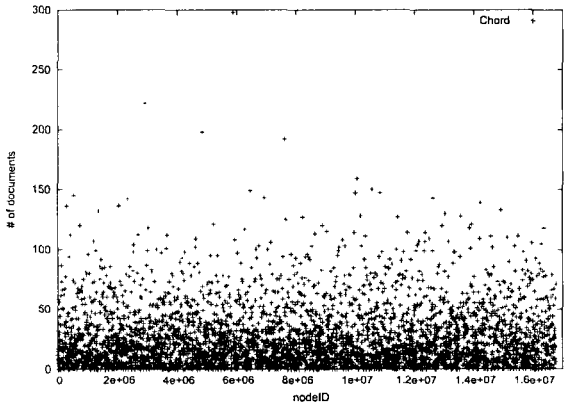

그림 1. **# of documents for each node in Chord**

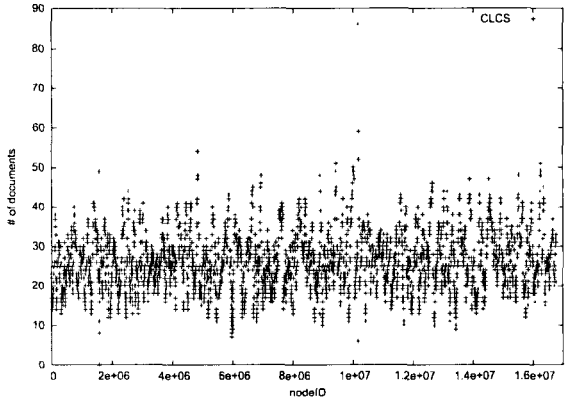

그림 2. **# of documents for each node in CLCS**

5. 결론

본 논문은 CAN, Chord, Pastry, Tapestry등과 같은 Peer-to-Peer network을 위한 여러 overlay network 시스템에서 해쉬 함수를 사용하여 nodeID와 objectID를 생성하여, 그 자체만으로 load-balancing을 기대하나 실험에 의하면 각 노드에 주어지는 load는 상당한 편차를 갖고 있음을 보였다. 그리고, 노드들간 작용을 통해서 load를 보다 효과적으로 분배할 수 있는 방법인 CLCS 기법을 제시하였고, 그 결과 노드별 load의 편차를 줄여주었음을 simulation을 통해서 증명하였다.

6. 참고 문헌

[1] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, Scott Shenker. A Scalable Content Addressable Network. *ACM SIGCOMM*, 2001.

[2] Antony Rowstron, Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. *Lecture Notes in Computer Science*, 2218, 2001.

[3] Ion Stoica, Robert Morris, David Karger, Frans Kaasho다, Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proc. of the ACM SIGCOMM Conf*, 2001.

[4] B. Y. Zhao, J. D. Kubiatowicz, A. D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, UC Berkeley, 2001.

[5] . Karger, E. Lehman, F. T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. *ACM Symposium on Theory of Computing*, 1997.