검색 속도 향상을 위한 P2P 분산 k-ary 복제 기법

양은호° 나종근 박상하 김종권
서울대학교 컴퓨터공학부
{ehyang°, jkna, shpark, ckim}@popeye.snu.ac.kr

DkR: Distributed k-ary Replication Scheme for Peer-to-Peer System

Eunho Yang°, Jongkeun Na, Sangha Park, Chongkwon Kim
School of Computer Science and Engineering, Seoul National University

요 약

DNS나 E-mail 혹은 Instant Messaging등 메시지 애플리케이션에서 서버-클라이언트 모델은 scalability에 의한 한계가 있다. 이러한 문제를 해결할 수 있는 대안은 Peer-to-Peer 기반 접근방식이다. 하지만 이러한 환경에서는 기존의 P2P 시스템에 높은 데이터의 지속성과 Bounded Look-up Latency를 효과적으로 지원하는 방안이 필수적이다. 본 논문에서는 Replication을 통하여 Look-up Latency를 줄이기 위한 하나의 방법으로 Distributed k-ary Replication(DkR)을 제안한다. DkR은 source node와 root node 사이에 multi-path를 유지하고 Look-up Locality 특성을 이용하여 Replica를 배치함으로써 효율적으로 Look-up latency를 줄이게 된다. 마지막으로 DkR의 Replication 효과를 시뮬레이션을 통해서 확인을 해 보았다.

1. 서 론

Peer-to-Peer (P2P) 시스템은 large-scale scalability를 필요로 하는 네트워크 기반 응용에 적용될 수 있는 새로운 구조를 제안한다. P2P 시스템에서 사용자는 자신의 resource(CPU, DISK, Network Bandwidth등)를 시스템에 일부 기여하고 시스템으로부터 파일공유나 메시징과 같은 서비스를 제공받는다.

현재까지 가장 잘 알려진 P2P 애플리케이션은 Napster, KaZaA와 같은 파일공유를 목적으로 하는 것들이지만 머지않아 기존 DNS나 E-Mail 시스템, 인스턴트 메시징을 대처 할 수 있는 P2P 기반 애플리케이션이 출현될 것이다. 이러한 애플리케이션들이 모두 클라이언트-서버 시스템에 의한 scalability 문제를 갖고 있고 이를 해결할 수 있는 대안이 바로 P2P 기반 접근방식이기 때문이다. 일반적인 메시징 시스템의 경우 메시지 ID나 사용자 ID를 기반으로 메시지가 관리되므로 메시지를 찾고 관리할 수 있는 것을 보장하는 구조화된(structured) 오버레이 기법(CAN, Chord, Pastry/Tapestry 등)이 적용될 필요가 있다.

새로운 P2P 애플리케이션을 위하여 1) 데이터의 지속성(persistency) 및 이용성(availability)을 보장해야 하며, 2) DNS와 같이 지연에 민감한 애플리케이션을 고려하여 look-up latency를 최소화해야 한다. 서버가 존재하지 않는 P2P환경에서 이러한 사항을 만족시키기 위해서는 Replication이 필수적이라고 할 수 있겠다. Replication시 얼마나 많은 Replica를 만들어야 하며 어떻게 배치해야 하는지 결정하는 것은 look-up latency를 결정짓는 요소이기 때문에 매우 중요하다.

본 논문에서 제안하는 Distributed k-ary Replication (DkR) 기법은 Replication Level (k)에 따라서 Replica의 개수를 제한하고, 제한된 Replica를 효과적으로 배치시켜서 look-up시 이득을 얻는 방식이다. 구체적인 내용은 3장에 기술되어 있으며 4장에서 시뮬레이션을 통하여 DkR 기법의 replication 성능을 볼 수 있다.

2. 관련 연구

Pastry[1]는 self-organizing, highly scalable, 그리고 fault tolerant하기 위해 만들어진 구조화된 P2P overlay network이다. 여기서는 모든 node와 object는 커다란 id space에서 선택되어진 "nodeID 와 key"라고 하는 유일한 identifier를 할당 받는다. 주어진 message와 key를 가지고 Pastry는 그 key와 숫자상으로 가장 가까운 nodeID를 가진 node로 message를 전달한다. 이때 route 상에 존재하는 중간 node는 network locality를 제공하기 위하여 IP hop 수와 같은 approximation metric을 이용하여 결정이 된다.

PAST[2]는 Pastry위에 지어진 storage system이다. PAST에서 replica는 그 key와 숫자상으로 가장 가까운 nodeID를 가진 k개의 node에 저장된다. PAST는 또한, node의 addition이나 failure에 상관없이 k개의 node에 복제된 object를 똑같이 유지한다. 각 node가 해당 object를 찾고자 할 때는 Pastry의 locality에 의해 복제된 object 중 자신과 가장 가까운 node에게 먼저 도달하게 된다.

[3]에서는 look-up performance를 제공하기 위해 proactive replication framework 제안하였다. O(1) look-up performance를 성취하는 closed-form optimal solution에 기반을 두었다. 하지만 replication하는 level 단계를 많이 할수록 storage requirement가 기하급수적으로 늘어나고 look-up을 하지 않아도 수시로 새로 들어오는 node나 failure된 node를 확인하기 위하여 refresh해야 하므로 network overhead가 커지는 단점이 있다.

3. Distributed k-ary Replication 기법

3.1 개념 및 동작

우리는 이 논문에서 Distributed k-ary Replication (DkR)이라는 이름의 replication 기법을 제안하였다. Distributed K-ary Replication 기법은 Chord, Pastry, Tapestry 와 같은 prefix-routing을 사용하는 모든 DHT(Distributed Hash Table) 위에서 동작하는 framework이다. 각 node는 unique하고 random한 id를 가지고 있고, 각 node들은 circular identifier space위에 존재하게 된다. 각 object는 random id를 부여 받게 되고 object의 id와 가장 가까운 id를 가지는 node(root node라고 부르자)가 저장을 하게 된다. 본 논문에서는 General Messaging System을 가정하였고, 또한 문제의 핵심을 보다 명확하고 간단하게 하기 위하여 Single writer를 가정하여 업데이트에 대한 동시성 제어 (concurrent control)를 필요로 하지 않는다고 가정하였다. 이제 DkR의 operation에 대해서 더욱 자세하게 살펴보도록 하자.

3.2 Replication Operation

DkR은 기본적으로 Source node가 처음으로 data를 등록할 때, 그 data를 저장해줄 root node로 routing해가는 경로 상에 존재하는 모든 intermediate node가 replication을 갖는 방법을 사용한다. 이 때, 기존의 DHT의 mechanism에서는 root을 찾아가는 하나의 path만이 존재하지만 DkR에서는 replication level k개의 route를 parallel하게 search를 하면서 replication을 만들어 낸다. 이러한 mechanism의 intuition은 prefix-routing의 경우는 hop이 지나면 지날수록 routing하고자 하는 id와 일치하는 prefix의 길이가 길어지므로 routing 가능한 node의 수가 적어지게 되고, 여러 route가 합쳐지게 되므로 multi-path 상의 중간 node에 replica를 위치시키면 다른 node가 해당 object를 찾고자 할 때 replication 되어 있는 path와 만날 확률이 점점 커지게 되어 결국 replication을 찾을 수 있다는데 있다.

이 논문에서는 DHT로 Pastry를 사용하였다. k개의 parallel한 search가 가능하기 위해서는 기존의 Pastry routing의 각 entry가 k개 이상의 값을 유지하도록 수정해야 한다.

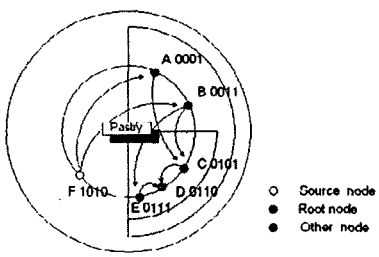

그림 1. Replication in DkR (k=2)

그림 1은 replication level k = 2이고, Id length=8, base bit length=1 인 Pastry overlay에서, node F가 id 0110인 data를 등록할 때, root를 찾고 replica를 만드는 과정이다. Pastry overlay 외각의 원, 반원, 부채꼴 영역은 search level이 커질 때 다음 hop으로 가능한 node들의 영역을 표시하고 있다. 우선, node F가 id의 첫째 자리가 0인 node 중 (반원으로 표시된 영역 내부의 node) node A, B를 routing entry로 가지고 있다고 하면, 두 node로 등록 메시지를 보낸다. 메시지를 받은 node A, B 도 각각 DkR을 사용하여 id의 둘째 자리까지 맞는 node(부채꼴로 표시된 영역 내부의 node)로 메시지를 보내게 된다. 그림을 간략하고 명확하게 하기 위해서 node B만 두 개의 multi-path로 메시지를 보내는 것으로 가정하였다. node A는 node C로, B는 node C, E로 메시지를 보내게 되고, 최종적으로는 node C, E에서 root node인 D node를 찾게 된다. 결과적으로 multi-path 상에 있는 node A에서 E까지 모든 node들이 replica을 가지게 된다.

3.3 look-up Operation

그림 2는 임의의 node가 특정 data를 search할 때의 과정을 나타낸 것이다. Search level이 증가할수록-hop 수가 증가할수록 더 많은 prefix가 일치하는 node를 routing table entry에 저장해야 하므로 상대적으로 적은 수의 가능한 node가 존재하게 된다. 하지만 그림에서 볼 수 있는 것처럼 replication을 가지고 있는 node의 비율은 점점 증가하게 된다. 그러므로 hop 수가 늘어날수록 replication을 찾을 확률이 점점 증가하게 되고, 일정 수준이 지나면 어떤 node로 routing 된다고 하더라도 거의 모든 node가 replica을 가지게 된다. 그러므로 DkR을 사용하여 data 등록 시에 한번만 multi-path를 이용하여 적정한 수준의 replica를 만들어 놓으면, 이후 look-up시에는 기존의 Pastry에서 사용하는 하나의 path를 이용한 routing을 한다고 하더라도 look-up latency를 기하급수적으로 줄일 수 있다는 장점이 생겨나게 된다.

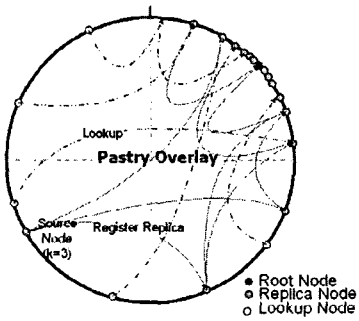

그림 2. look-up in DkR (k=3)

4. 성능 평가

4.1환경

제안된 방식인 DrK를 시뮬레이션하기 위해 FreePastry v1.3.2를 추가 확장하였다. 기존 FreePastry와는 다르게 각 라우팅 테이블의 앤트리 당 유지 될 수 있는 이웃 node의 개수를 default 1에서 8로 확장하였다.

replication 기법의 성능을 평가할 주요 metric은 space

overhead 측면에서 본 각 데이터의 replica의 수와 look-up gain 측면에서 본 hop count가 되겠다.

4.2 Replication Level vs. # of Replica

그림 3에서와 같이 k가 증가되면서 Replica의 개수가 증가됨을 확인할 수 있다. 당연히, k가 증가하면 멀티-패스의 개수가 증가할 것이고 따라서 Replication에 참여하는 node 수는 증가할 것이다. 하지만 전체 node 수에 비해 작은 수의 node 임을 알 수 있다.

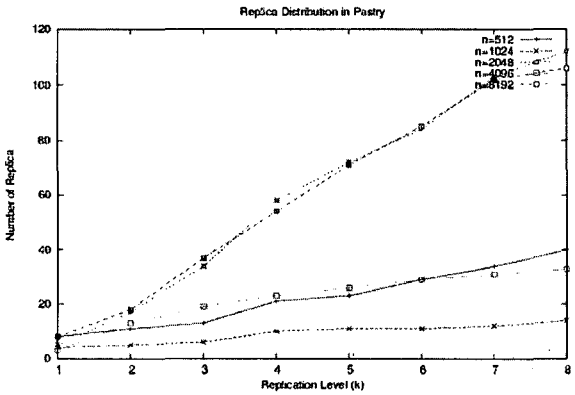

그림 3. Replication Level (k) vs. The number of Replica in Pastry with DkR 기법

4.3 Replication Level vs. Look-up Latency

전체 node 개수와는 독립적으로 Replication Level에 따라서 우리는 Look-up Latency가 줄어 드는 것을 확인할 수 있다. 그림 4에 보이듯이 node 개수를 달리한 모든 실험에서 DkR 을 적용할 경우 k를 증가시킬 경우 Look-up Latency가 감소함으로 알 수 있다. N=8192의 경우 k=1인 경우 평균 5.2 hops이 걸리는 것에 비해 k=8인 경우 평균 3.7 hops이 걸리므로 2 hops 정도의 look-up gain을 얻을 수 있다. DkR 기법을 적용하여 k에 따라 결정된 Replica 수만큼 multi-path상에 있는 node들에게 메시지를 Replication 함으로서 적은 양의 replica로 매우 큰 look-up gain를 얻었다.

그림 5를 통해서 우리는 replication level (k)에 따른 look-up시 hop count 분포를 볼 수 있다. K=8인 경우 72%가4 hops 이하임을 알 수 있다. 이에 반해, k=1인 경우 37%가 4 hops 이하를 갖는다.

5. 결론 및 향후 연구

제안된 Distributed k-ary Replication (DkR) 기법은 source와 root node사이에 존재하는 멀티패스상에 Replica를 배치하여 Look-up Latency를 줄이고자 하는 Idea에 기반한다. DkR는 Pastry의 look-up locality 특성을 이용함으로써 적은 Replica 개수로 최대의 look-up gain을 얻는다. DkR의 Replication 효과는 시뮬레이션을 통해서 부분적으로 증명이 되었다. 앞으로는 space / time 측면에서 DkR의 특성을 좀 더 수학적으로 분석할 필요가 있으며 분석 결과를 시뮬레이션을 통해서 확인하고자 한다.

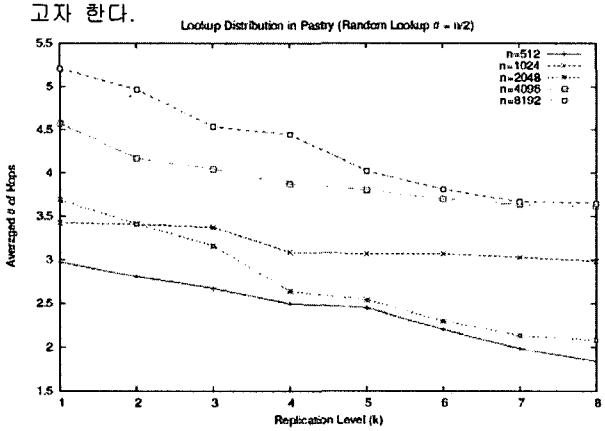

그림4. Replicaion Level vs. the Averaged number of Hops in Pastry Look-up with DkR

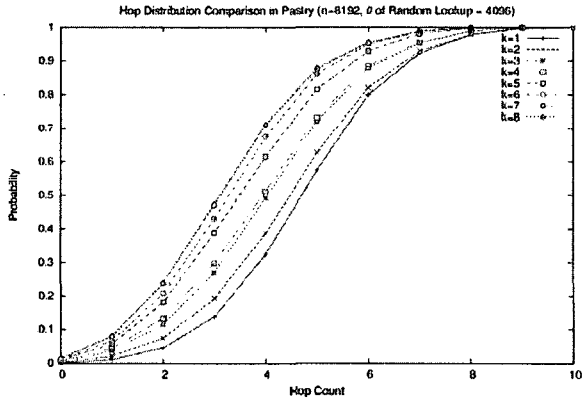

그림5. Hop distribution comparison in Pastry with DkR (전체 node 수 = 4096)

6. 참고 문헌

[1] Antony Rowstron and Peter Druschel, "Pastry : Scalable, decentralized object location and routing for large-scale peer-to-peer systems", IFIP/ACM ICDSP, 2001

[2] Antony Rowstron et al., "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", SOSP-18, 2001

[3] Venugopalan Ramasubramanian and Emin Gun Sirer, "Beehive: O(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays", NSDI, 2004

[4] John Kubiatowicz et al., "OceanStore : An Architecture for Global-Scale Persistent Storage", ASPLOS 2000

[5] Edith Cohen, Scott Shenker, "Replication Strategies for Unstructured Peer-to-Peer Networks", SIGCOMM 2002