Zookeeper 소개

최범균 (2014-05-23)

클러스터에 필요한 것

- 클러스터 구성원 관리
  - 살았니? 죽었니?
  - 누가 마스터?
- 클러스터 구성원 간 정보 공유
- 분산 서버의 동시 접근 제어

ZooKeeper 개요

- 분산 시스템을 위한 코디네이터
- 서버와 클라이언트로 구성
  - 서버 앙상블: n개 서버로 단일 주키퍼 클러스터 구성
  - 클라이언트: 앙상블에 속한 서버에 연결하여, 서비스 사용
- 서버 제공 서비스
  - 데이터 스토리지 (영속 데이터/임시 데이터)
    - 클러스터 구성원 간 데이터 공유
  - 데이터 변경 통지(Watch)
    - 클러스터 멤버십 관리
  - 시퀀스 노드
    - 마스터, 분산 락 등에 활용

ZooKeeper 데이터 모델과 주요 기능

ZNode

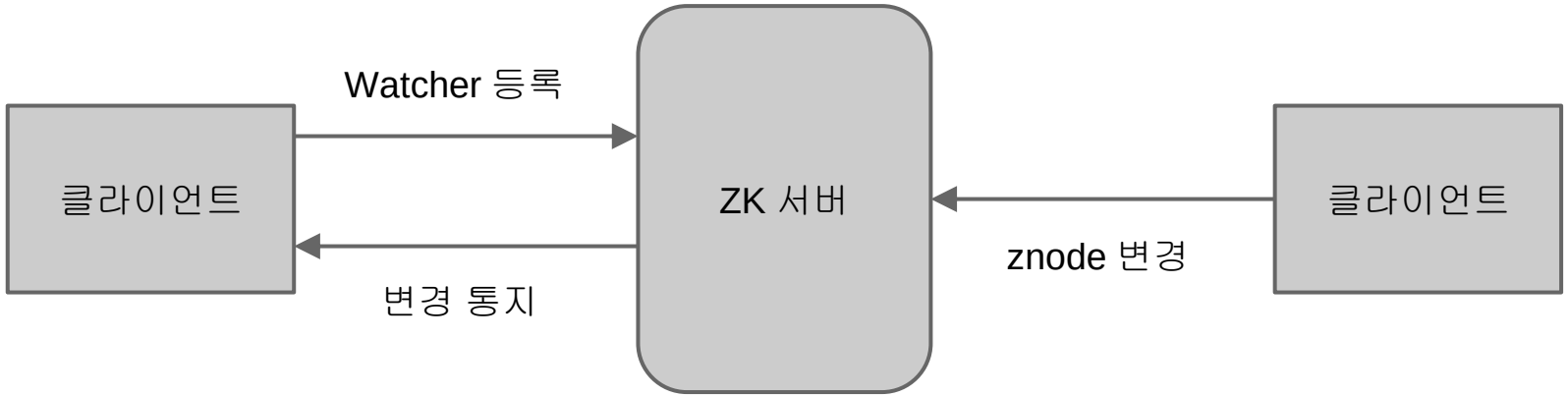
- ZNode
  - ZooKeeper는 계층형 네임스페이스를 가짐
  - 계층의 각 노드를 ZNode라고 부름
    - 계층의 각 경로는 ‘/’로 구분
  - znode는 데이터를 가질 수 있음 (byte[])
  - znode는 자식 노드를 가질 수 있음
    - 예, /svc 노드는 /svc/nodes 노드를 가질 수 있음
- 영속 종류
  - PERSISTENT: 클라이언트가 종료되도 유지
  - EPHEMERAL: znode를 생성한 클라이언트의 연결이 끊기면 삭제
    - 자식 znode를 가질 수 없음

주요 오퍼레이션

- 주요 오퍼레이션
  - create
  - delete
  - exists
  - getChildren
  - getData
  - setData
- 언어 별 바인딩 제공
  - Java, C, Python, Ruby, Scala 등
    - https://cwiki.apache.org/confluence/display/ZOOKEEPER/ZKClientBindings

ZNode Watch

- znode의 변화를 통지 받음
- 오퍼레이션 실행시 Wacher 등록
  - zk.getChildren(“/mysvc/nodes”, watcher)



ZNode Watch 이벤트 종류

Watch 생성 오퍼레이션	znode 생성	znode 삭제	자식 생성	자식 삭제	znode 값 변경
exists	NodeCreated	NodeDeleted			NodeDataChanged
getData		NodeDeleted			NodeDataChanged
getChildren		NodeDeleted	NodeChildrenChanged	NodeChildrenChanged	

Watch 특징

- 일회성 이벤트 수신
  - 다시 Watch하고 싶으면 재등록해야 함
  - 재등록 전에 발생한 변경은 수신하지 못함
- 상태 변경 요청한 클라이언트가 성공 코드를 받은 이후에, Watcher를 등록한 클라이언트에 통지 됨
- ZooKeeper의 변경 순서대로 Watch 이벤트를 수신 함

시퀀스 노드

- 자식 노드가 생성한 순서에 따라 일렬 번호를 가짐
  - 예, /svc 노드의 자식 노드로 /nodes를 시퀀스 노드로 생성하면 다음과 같은 순서로 노드 생성
    - /svc/nodes0000000001
    - /svc/nodes0000000002
  - PERSISTENT와 EPHEMERAL에 모두 적용 가능
  - 시퀀스 범위 (4 byte 정수)
    - -2,147,483,647 ~ 2,147,483,647 (-21억~21억)

ZooKeeper 몇 가지 용도

그룹 관리: 자식 ZNODE, Watch

- 그룹 멤버십 목록을 가질 부모 znode 생성
  - 예: /members

멤버십 참여
- 부모 znode에 자신을 위한 자식 znode를 ephemeral로 생성
  - 예, /members/m-hostip
- 자식 znode 데이터는 다른 구성원이 자신과 통신하는데 필요한 정보 저장 (예, ip/port 등)

멤버십 탈퇴
- 자신에 해당하는 자식 znode를 삭제

멤버십 목록, 주기적 갱신 (각 클라이언트)
- 주기적으로 getChildlen 실행해서 목록 갱신

멤버십 목록, Watch 이용 (각 클라이언트)
- 부모 znode에 getChildlen로 WATCH 등록하고, getChildlen 결과로 목록 갱신
- NodeChildrenChanged 이벤트 수신 시, 위 과정 재실행

마스터 선출: 시퀀스 노드, WATCH

- 마스터 후보를 보관할 znode 생성
  - 예: /masters

마스터 후보 등록

- 부모 znode에 자신을 위한 자식 znode를 시퀀스 & ephemeral로 생성
  - 예, /masters/m-0000000010

마스터 선출 (각 후보 클라이언트에서 진행)

- 부모 znode에 getChild로 WATCH 등록
- NodeChildrenChanged 이벤트 수신시, 현재 마스터가 없는 지 확인
- 마스터에 해당하는 자식 znode가 없으면
  - 부모 znode에 자식 znode 를 구한 뒤 시퀀스 번호가 가장 작은 znode를 마스터로 선택
- 부모 znode에 getChild로 WATCH 등록

기타 용도

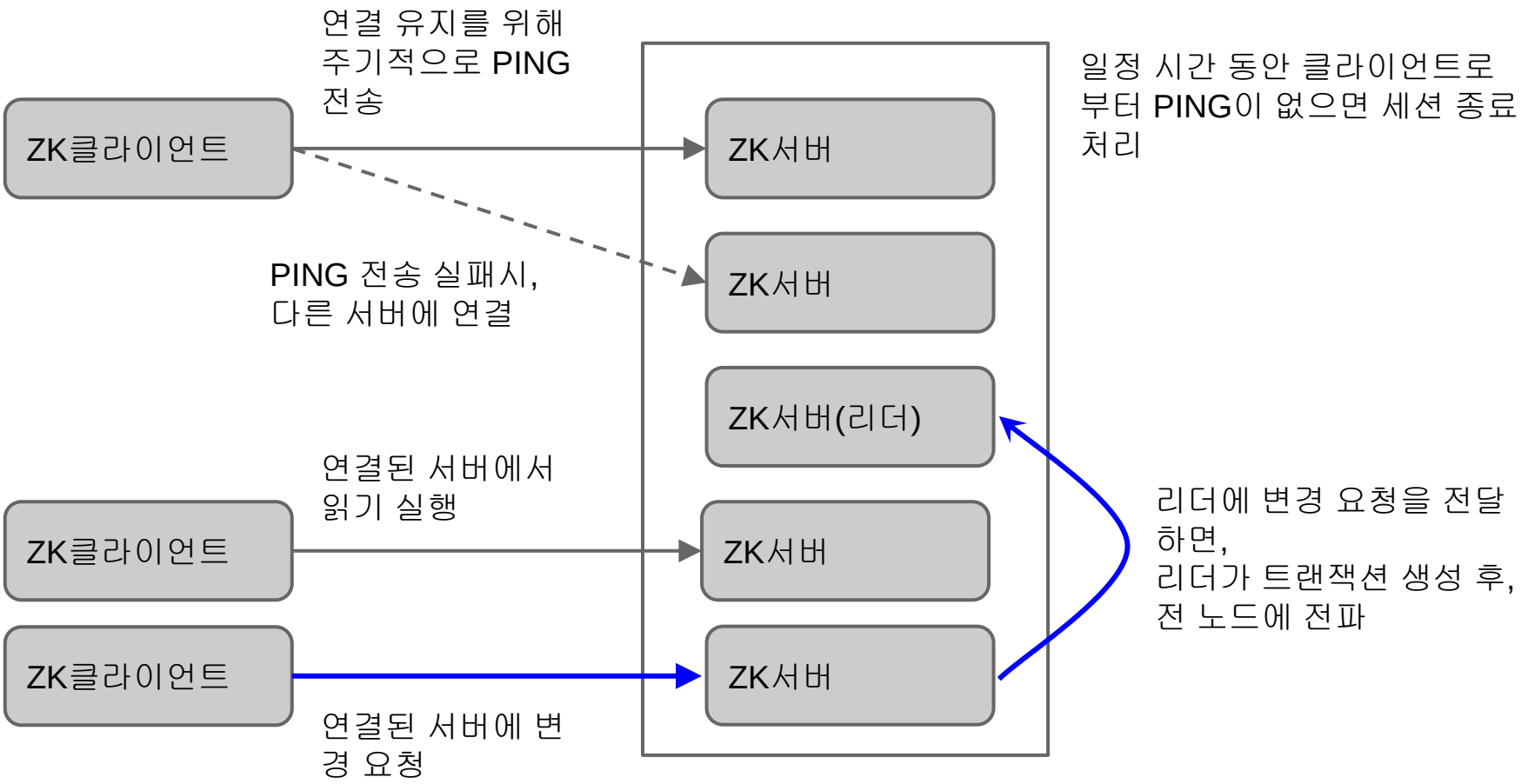
- 분산 락
  - EPHEMERAL 사용
- 이벤트 통지
  - WATCH 사용
- pub/sub
  - SEQUENCE 사용
- 분산 카운터
  - SEQUENCE 또는 znode의 version 사용
- 디렉토리 서비스
  - 활성화된 서비스 목록 등 제공

ZooKeeper 동작 방식 개요

버전

- zxid: ZooKeeper Transaction ID
  - 모든 변경은 고유 트랜잭션 ID를 가짐
  - 순서를 가짐
- version
  - znode가 변경될 때 마다 버전 값 증가
- cversion
  - 자식 znode가 변경될 때 마다 버전 값 증가

서버와 클라이언트 연결


연결 유지를 위해
주기적으로 PING
전송
ZK클라이언트
ZK서버
일정 시간 동안 클라이언트로
부터 PING이 없으면 세션 종료
처리
PING 전송 실패시,
다른 서버에 연결
ZK서버
ZK서버(리더)
연결된 서버에서
읽기 실행
ZK클라이언트
ZK서버
리더에 변경 요청을 전달
하면,
리더가 트랜잭션 생성 후,
전 노드에 전파
ZK클라이언트
ZK서버
연결된 서버에 변
경 요청

Quorum: 가용성, 데이터 변경

- 과반 정족수 이상 서버 다운시, 서비스 중지
  - 예, 2대 중 1대 다운되면 서비스 중지
  - 이런 이유로 서버 개수는 홀수로 맞춤 (3, 5 등)
- 데이터 변경시 성공 기준
  - 리더 서버가 트랜잭션 생성 후 다른 서버에 전파할 때, 과반 정족수 이상 트랜잭션 성공하면 클라이언트에 변경 성공 결과 리턴

일관성 보장

- Sequential Consistency: 변경 요청은 순서대로 적용
- Atomicity: 변경은 성공 또는 실패
- Single System Image: 클라이언트는 연결한 서버에 상관없이 동일 서비스 사용
- Reliability: 클라이언트가 성공 코드를 받으면, 서버에 반영됨을 보장
- Timeliness: 클라이언트는 일정 시간내에 최신 상태를 사용 (Eventual Consistency)

맺음말

믿고 쓸 만한 분산 코디네이터

- 다양한 오픈소스에서 채택
  - Hadoop: 마스터(Name Node) 이중화
  - HBase: 마스터 선출, 리전 데이터 공유 등에 사용
  - Storm: 서버 간 데이터 공유 등에 사용
  - Kafka: 클러스터 관리
  - Neo4j: 마스터 선출에 사용