Author : Saurav Haloi
Source : http://www.slideshare.net/sauravhaloi/introduction-to-apache-zookeeper
☞ This is translation for Korean language (also add comments to origininal presentation)

Apache ZooKeeper™ 소개

http://zookeeper.apache.org/

Sunny Kwak

sunykwak@daum.net

Agenda

분산 시스템이란?

“분산 시스템은 복수의 컴퓨터가 네트워크를 통해 통신하며 하나의 목적(목표)를 위해 서로 간에 상호작용 하는 것이다. 달리 말해 다수의 컴퓨터가 마치 하나인 것처럼 동작하는 시스템이다.”

분산 컴퓨팅에 대한 착각

- 네트워크는 신뢰할 수 있다 (reliable).
- 지연(latency)은 없다.
- 대역폭(bandwidth)은 무한하다.
- 네트워크는 안전하다 (secure).
- 토폴로지(topology)는 변경되지 않는다.
- 단 한 사람의 관리자만 있다 (administrator).
- 전송(transport) 비용은 공짜다.
- 네트워크 유형은 동일하다(homogeneous).

☞ *http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing*

분산 컴퓨팅에서의 조율

- *조율(Coordination)*
  : 다양한 노드가 함께 동작하도록 만드는 행위

- 예시 :
  - 그룹 멤버쉽 (Group Membership)
  - 잠금 제어 (locking)
  - 공급/구독 (Publisher/Subscriber)
  - 리더 선정 (Leader Election)
  - 동기화 (Synchronization)

- 노드(node)들을 조율(조정)하는 것은 매우 어렵다!

ZooKeeper 소개

> ZooKeeper 는 분산된 프로세스들이 공유된 계층적 데이터 레지스터들을 통해 조화롭게 수행될 수 있게끔 한다. - Zookeeper WiKi

ZooKeeper 는 분산 락(lock) 서버 이상이다.

ZooKeeper 란 무엇인가?

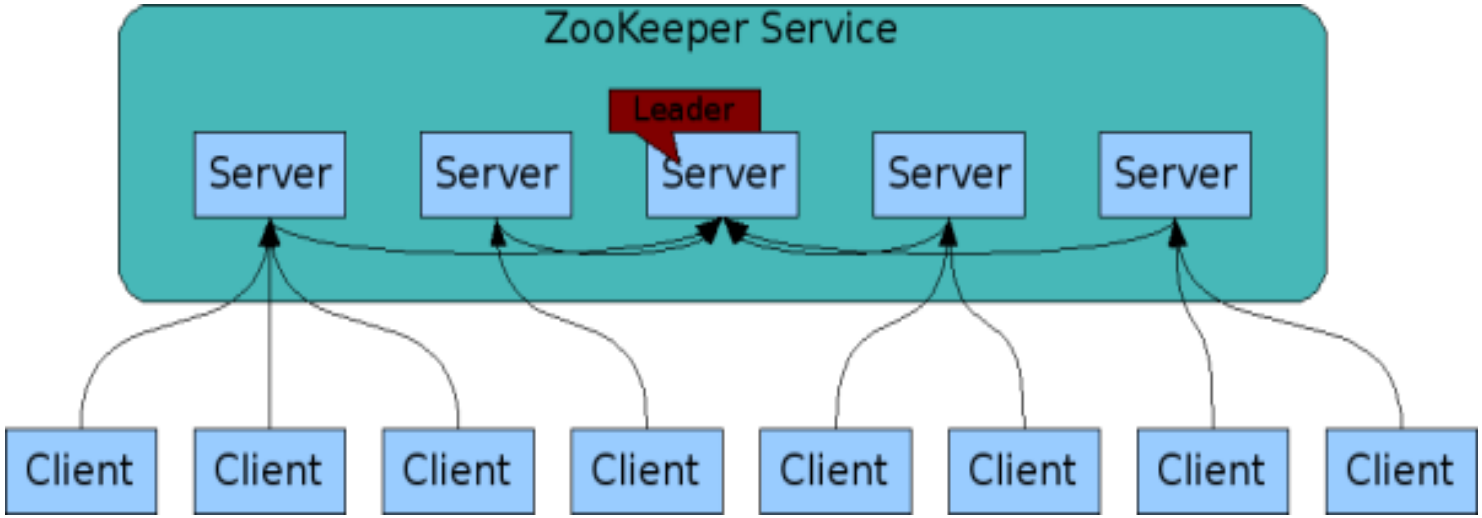
- 분산 어플리케이션들을 위한 오픈 소스, 고성능 조정자 (coordination) 서비스.
- 단순한 인터페이스를 통해 공통 서비스를 제공.
  - 명명 (naming)
  - 설정 관리 (configuration management)
  - 잠금 및 동기화 (locks & synchronization)
  - 그룹 서비스 (group services)

  ... 개발자는 기초 수준부터 코드를 작성할 필요가 없다.
- 필요에 따라, 원하는 기능을 구축(개발)할 수 있다.

ZooKeeper 활용 방안

- 설정 관리 (Configuration Management)
  - Cluster member nodes bootstrapping configuration from a centralized source in unattended way
  - Easier, simpler deployment/provisioning
- 분산 클러스터 관리 (Distributed Cluster Management)
  - Node join / leave
  - Node statuses in real time
- 명명 서비스 (Naming service) – e.g. DNS
- 분산 동기화 (Distributed synchronization) - locks, barriers, queues
- 분산 시스템에서 리더 선출 (Leader election in a distributed system).
- 중앙집중형 신뢰성 잇는 데이터 저장소
  (Centralized and highly reliable (simple) data registry)

ZooKeeper 서비스



- ZooKeeper 서비스는 복수의 서버에 복제된다.
- 모든 서버 장비는 데이터의 사본(copy)을 메모리에 저장한다.
- 서비스 기동(startup) 시 리더가 선출된다.
- 클라이언트들은 하나의 ZooKeeper 서버에 TCP/IP 로 연결을 실행하고 유지한다.
- 클라이언트는 모든 ZooKeeper 서버에서 읽을 수 있으며,
  리더를 통해 쓸 수 있되 과반수 서버의 승인(합의)이 필요하다.

Image ☞ https://cwiki.apache.org/confluence/display/ZOOKEEPER/ProjectDescription

ZooKeeper 데이터 모델

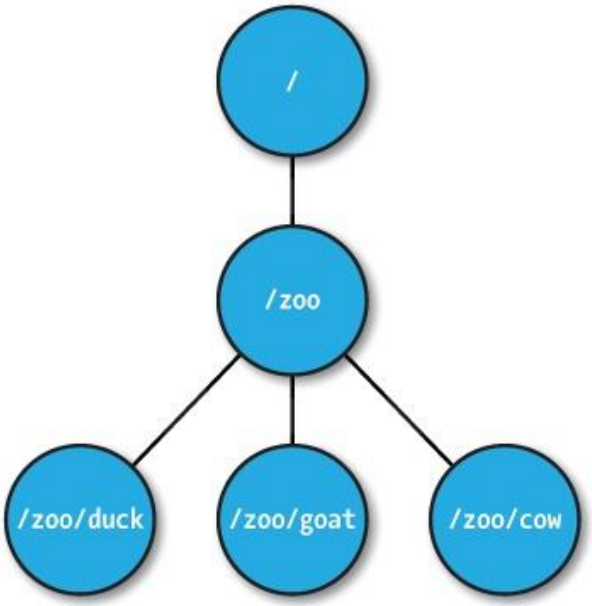
- ZooKeeper 는 계층적인 네임스페이스(namespace)를 제공한다.
- 네임스페이스 내에 존재하는 개별 노드를 Znode 라고 부른다.
- 모든 Znode 는 데이터 (바이트 배열)를 가질 수 있으며,
  자식을 가질 수도 있다.

  부모 : "foo"
  |-- 자식1 : "bar"
  |-- 자식2 : "spam"
  `-- 자식3 : "eggs"
  `-- 손자1 : "42"

- ZNode 경로 :
  - 절대경로, '/' 로 구분됨
  - 상대 참조가 없음.
  - 명칭에 유니코드 문자가 포함될 수 있음

ZNode

- Maintain a stat structure with version numbers for data changes, ACL changes and timestamps.
- 변경이 발생하면 버전 번호가 증가한다.
- 데이터는 항상 전체를 읽고 쓴다.



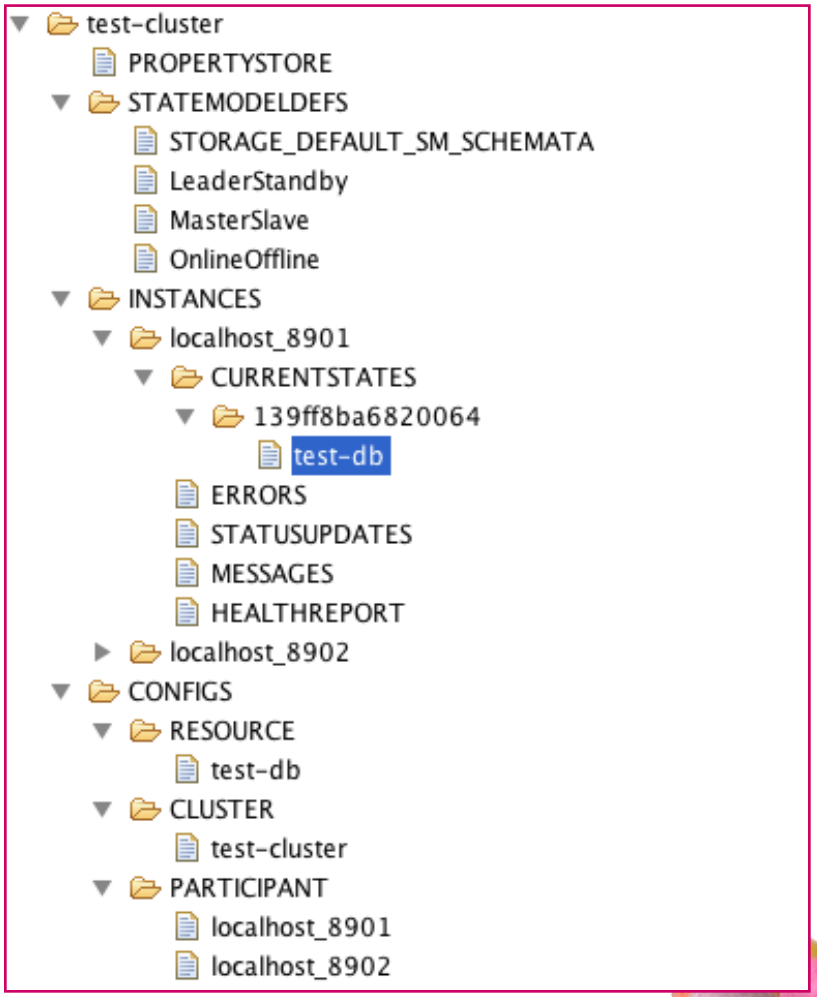


Image: http://helix.incubator.apache.org/Architecture.html

ZNode 유형

- 영구 노드 (Persistent Nodes)
  - 명시적으로 삭제되기 전까지 존재함.
- 임시 노드 (Ephemeral Nodes)
  - 세션이 유지되는 동안 활성 (세션이 종료되면 삭제됨)
  - 자식 노드를 가질 수 없음.
- 순차 노드 (Sequence Nodes)
  - 경로의 끝에 일정하게 증가하는 카운터 추가
  - 영구 및 임시 노드 모두에 적용 가능.

Znode 연산

Operation	Type
create	Write
delete	Write
exists	Read
getChildren	Read
getData	Read
setData	Write
getACL	Read
setACL	Write
sync	Read

Znodes 는 프로그래머가 접근(제어)하는 핵심 객체이다.

ZooKeeper Shell

```
[zk: localhost:2181(CONNECTED) 0] help
ZooKeeper -server host:port cmd args
        connect host:port
        get path [watch]
        ls path [watch]
        set path data [version]
        rmr path
        delquota [-n|-b] path
        quit
        printwatches on|off
        create [-s] [-e] path data acl
        stat path [watch]
        close
        ls2 path [watch]
        history
        listquota path
        setAcl path acl
        getAcl path
        sync path
        redo cmdno
        addauth scheme auth
        delete path [version]
        setquota -n|-b val path
```

```
[zk: localhost:2181(CONNECTED) 1] ls /
[hbase, zookeeper]

[zk: localhost:2181(CONNECTED) 2] ls2 /zookeeper
[quota]
cZxid = 0x0
ctime = Tue Jan 01 05:30:00 IST 2013
mZxid = 0x0
mtime = Tue Jan 01 05:30:00 IST 2013
pZxid = 0x0
cversion = -1
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 0
numChildren = 1

[zk: localhost:2181(CONNECTED) 3] create /test-znode HelloWorld
Created /test-znode
[zk: localhost:2181(CONNECTED) 4] ls /
[test-znode, hbase, zookeeper]
[zk: localhost:2181(CONNECTED) 5] get /test-znode
HelloWorld
```

ZooKeeper 감시

- 클라이언트는 Znode 에 감시(watch)를 설정할 수 있다 :
  - 노드의 자식이 변경된 경우 (NodeChildrenChanged)
  - 노드가 생성된 경우 (NodeCreated)
  - 노드의 데이터가 변경되는 경우 (NodeDataChanged)
  - 노드가 삭제된 경우 (NodeDeleted)
- ZNode가 변경되면 감시 이벤트가 발생하고, 변경사항이 클라이언트로 통지된다.
- 감시(watch)는 1회성 트리거(trigger) 이다.
- 감시(watch)는 언제나 순서대로 실행된다.
- 클라이언트 새로운 노드 데이터가 생성되기 전에 감시 이벤트를 받는다.
- 클라이언트는 이벤트 수신 및 새로운 감시 요청을 전송하는 중 발생할 수 있는 지연에 대비해야 한다.

API 동기 / 비동기

- API 메소드는 동기(sync) 뿐 아니라 비동기 방식(async)으로 동작한다.

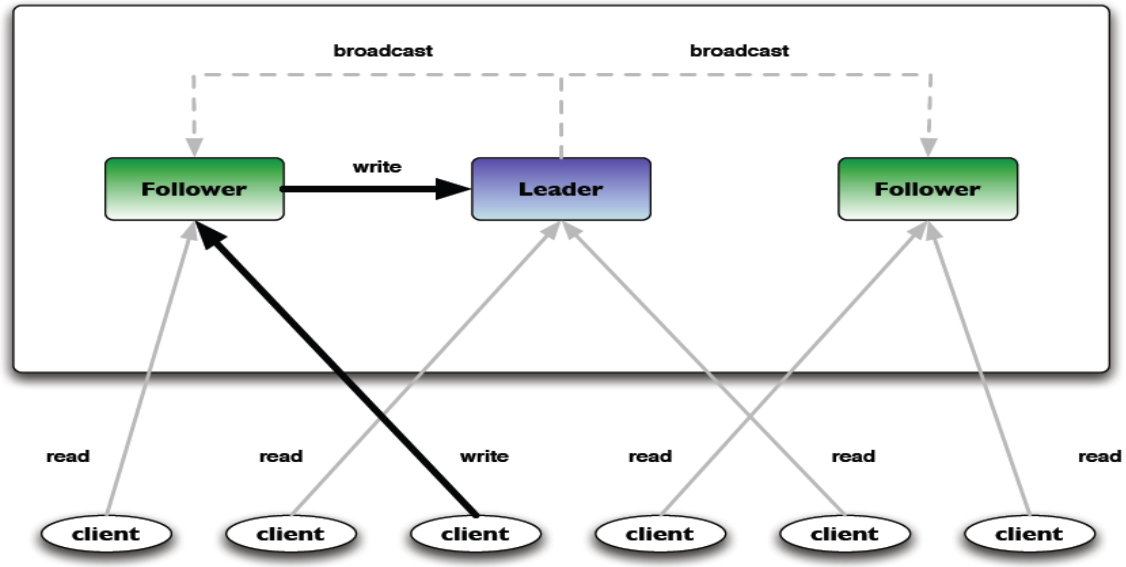
- 동기:

```
exists("/test-cluster/CONFIGS", null);
```

- 비동기:

```
exists("/test-cluster/CONFIGS", null, new StatCallback() {
        @Override
        public processResult(int rc, String path, Object ctx, Stat stat)        {
                //process result when called back later
        }
}, null);
```

Znode 읽기 / 쓰기



- 조회 요청은 클라이언트가 연결한 ZooKeeper 서버 내에서 처리된다.
- 쓰기 요청은 리더로 전달되며, 클라이언트로 정상 응답하기 전에 과반수 이상의 서버에서 쓰기가 완료되어야 한다.

Image: http://www.slideshare.net/scottleber/apache-zookeeper

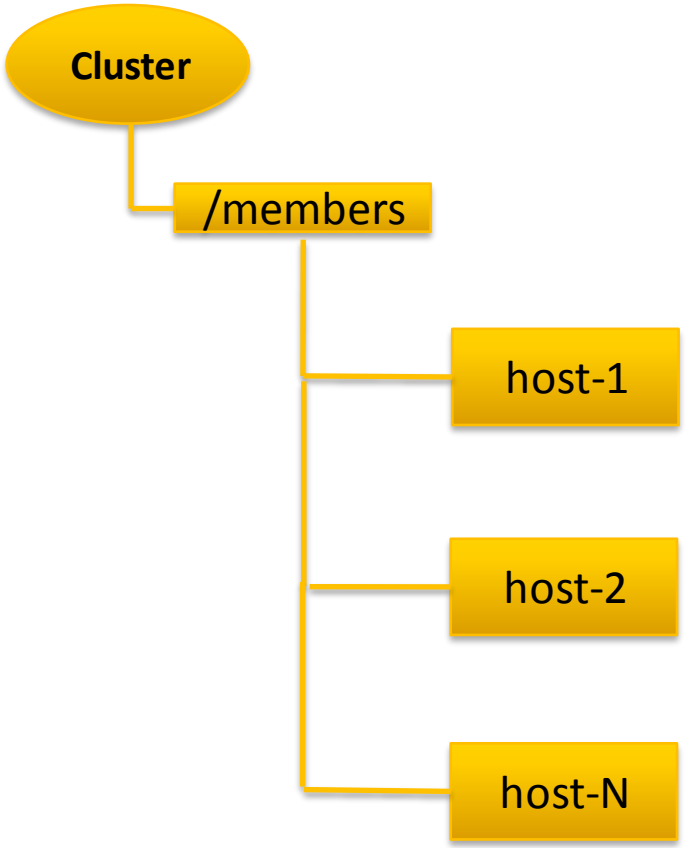
일관성 보장

- **순차 일관성 (Sequential Consistency)**
  - 변경은 요청한 순서대로 반영 된다.
- **원자성 (Atomicity)**
  - 변경은 확실히 성공하거나, 실패한다.
- **단순한 시스템 형상 (Single System Image)**
  - 클라이언트가 어떤 ZooKeeper 서버에 연결하건 간에 동일한 뷰(view)를 조회할 수 있어야 한다.
- **신뢰성 (Reliability)**
  - 변경은 적용된 이후에 동일한 클라이언트에 의해 덮어써지기 전까지는 지속(유지)되어야 한다.
- **시기적절성 (Timeliness)**
  - 클라이언트가 보는 뷰(view)의 데이터는 특정 간격 내에서는 최신 정보임을 보장해야 한다. (Eventual Consistency)

사례 #1 : 클러스터 관리

클라이언트 호스트 i, i:=1 .. N

1. /members 노드 감식
2. /members/host-${i} 임시 노드들을 생성
3. 노드 가입/탈퇴 시 이벤트 발생
4. 주기적으로 /members/host-${i} 노드들의 상태를 변경 (load, memory, CPU etc.)



사례 #2 : 리더 선출

1. "/svc/election-path" ZNode 생성
2. 선출 과정에 참여하는 모든 참가 프로세스들은 동일한 선거 경로에 임시 노드를 생성한다.
3. **가장 작은 순번에 해당하는 서버가 리더가 된다.**
4. 각각의 "follower' 노드는 자신보다 다음 낮은 순번의 노드를 감시(listen)한다.
5. 리더가 'election-path'에서 제거되면 새로운 리더를 선출하거나, 아니면 다음 낮은 순번의 노드가 리더가 된다.
6. 세션이 만료 시, 상태를 검사하고 필요하면 리더를 선출할 수 있다.

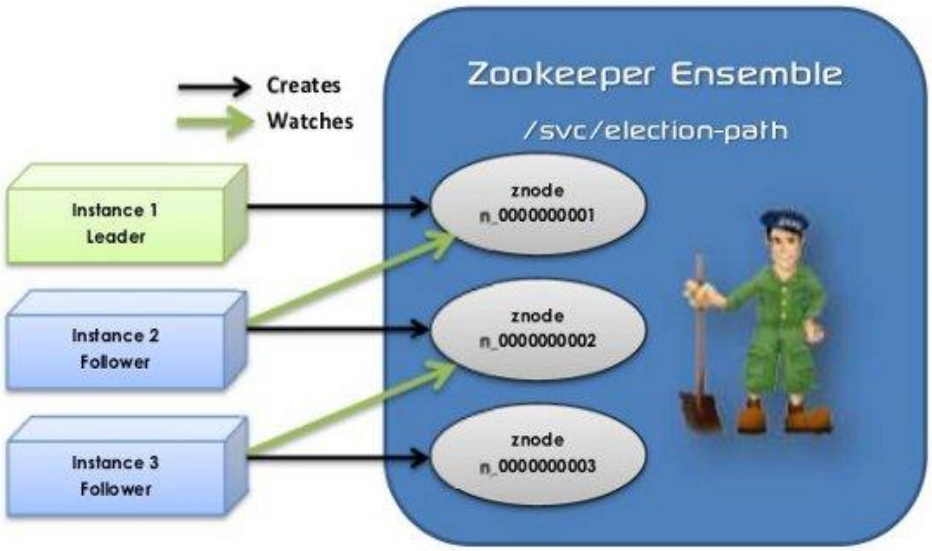


Image: http://techblog.outbrain.com/2011/07/leader-election-with-zookeeper/

사례 #3 : 분산 배타적 잠금

N개의 클라이언트가 잠금(lock)을 소유하려고 시도한다고 가정.

- 클라이언트들을 임시, 순차 Znode를 '/Cluster/_locknode_' 아래에 생성한다.
- 클라이언트들은 잠금 Znode (i.e. _locknode_) 하위의 자식 리스트를 요청한다.
- ***가장 낮은 ID를 가진 클라이언트가 잠금(lock)을 소유한다.***
- 그외의 클라이언트들은 감시(watch)를 수행한다.
- 통지가 발생할 때마다 잠금을 확인한다.
- 잠금을 해제하고 싶은 클라이언트는 노드를 삭제하고, 다음 클라이언트가 잠금을 획득하게 된다.

```
ZK
|---Cluster
    +---config
    +---memberships
    +---_locknode_
        +---host1-3278451
        +---host2-3278452
        +---host3-3278453
        +--- ...
        ₩---hostN-3278XXX
```

언어 지원

- ZooKeeper 클라이언트 라이브러리 지원 언어 :
  - Java
  - C
  - Perl
  - Python

- 커뮤니티 지원
  : Scala, C#, Node.js, Ruby, Erlang, Go, Haskell

https://cwiki.apache.org/ZOOKEEPER/zkclientbindings.html

몇가지 기억해야 할 점

- 감시(watches)는 일회성 트리거이다.
  - Znode를 계속적으로 감시하기 위해서는 이벤트/트리거가 발생할 때마다 재설정해야 한다.
- 많은 감시를 znode에 설정하면 'herd effect'가 발생한다.
  - 트래픽이 폭주하고, 확장성을 떨어뜨린다.
- Znode 에 대한 이벤트를 수신하고, 감시를 다시 설정하는 동작 znode 가 계속 변경된다면 신중하게 제어해야 한다.
- 세션 만료 시간은 가급적 길게 설정해서 가비지 컬렉션으로 인한 '정지 시간'을 줄여야 한다.
- Swapping 이 발생하지 않도록 자바 최대 heap 사이즈를 적절하게 설정해야 한다.
- ZooKeeper 트랜잭션 로그 기록을 위한 전용 디스크를 설정해야 한다.