

# spark standalone-HA mode

Le cluster Spark Standalone est un modèle de cluster de l'architecture Master-slave.

Problème :

Master = Point de défaillance du cluster .

Solution : Spark propose deux solutions :

1) Récupération à nœud unique avec système de fichiers local

ne peut être utilisée que dans des environnements de développement ou de test.

2) Standby Masters basés sur Zookeeper (Standby Masters avec ZooKeeper) - peuvent être utilisés dans des environnements de production.

# SPARK HA

## ZOOKEEPER

@192.168.1.10

Master-Alive

Master-StandBy

Master-StandBy

@192.168.1.20

@192.168.1.30

W1

W2

W3

@192.168.1.40

@192.168.1.50

@192.168.1.60

# Configuration cluster SPARK HA avec ZK

## Fichier /opt/zookeeper/conf/zoo.cfg

```
server.1=192.168.1.10:2888:3888  
server.2=192.168.1.20:2888:3888  
server.3=192.168.1.30:2888:3888  
server.4=192.168.1.40:2888:3888  
server.5=192.168.1.50:2888:3888
```

## Fichier /opt/spark/conf/spark-env.sh

```
export SPARK_DAEMON_JAVA_OPTS="  
-Dspark.deploy.recoveryMode=ZOOKEEPER  
-Dspark.deploy.zookeeper.url=192.168.1.10:2181,192.168.1.20:2181, 192.168.1.30:2181, 192.168.1.40:2181,192.168.1.50:2181  
-Dspark.deploy.zookeeper.dir=/opt/spark/spark_recovery"
```

## Créer un fichiers myid dans le répertoire data.

Ce fichier ne contient uniquement une ligne qui est le numéro de l'instance Zookeeper qui utilise ce répertoire de donnée

## zookeeper quorum hosts

Le fichier **zoo.cfg** contient les propriétés suivantes qui sont intéressantes dans un contexte de détection d'échec : \*\*

**tickTime=2000**

**initLimit=5**

**syncLimit=2**

La propriété **initLimit** définit le nombre de graduations d'attente pendant les tentatives de connexion au noeud principal ZooKeeper.

Par exemple, si la propriété **initLimit** est définie sur 5 et que la propriété **tickTime** est de 2000 millisecondes,

un noeud ZooKeeper attend  $5 \times 2000$  millisecondes avant d'abandonner la connexion et de décider de choisir un nouveau leader.

Il est recommandé de conserver la valeur par défaut de la propriété **tickTime** et de régler la propriété **initLimit**

en fonction de votre environnement et de votre déploiement afin de réduire l'impact des échecs de noeud.

**SyncLimit** : limite à quel point un serveur peut être obsolète par rapport à un leader.

\*\* <https://zookeeper.apache.org/doc/r3.1.2/zookeeperStarted.html>

\*\* <https://zookeeper.apache.org/doc/r3.4.14/zookeeperStarted.html>

## zookeeper quorum hosts

Si l'un des serveurs ZooKeeper est en panne, ZK en utilisera un autre de la liste.

Tant que la majorité des serveurs ZooKeeper sont opérationnels, le service sera disponible.

Parce que Zookeeper nécessite une majorité, il est préférable d'utiliser un nombre impair de machines. Généralement 3 ou 5.

Par exemple, avec quatre machines, ZooKeeper ne peut gérer que la panne d'une seule machine ;

si deux machines échouent, les deux machines restantes ne constituent pas la majorité.

Cependant, avec cinq machines, ZooKeeper peut gérer la défaillance de deux machines.



## zookeeper quorum hosts

ZooKeeper fournit un mécanisme d'élection de leader.

Ce mécanisme garantit que bien qu'il y ait plusieurs maîtres dans le cluster, un seul est actif et les autres sont en StandBy.

Lorsque l'Actif Master échoue, un autre Standby Master sera élu.

Étant donné que les informations du cluster, y compris les informations des workers du driver et de l'application, ont été conservées dans le système de fichiers,

**L'ensemble du processus de récupération (à partir du moment où le premier leader tombe) devrait prendre entre 1 et 2 minutes.**

**Notez que ce délai n'affecte que la planification de nouvelles applications –**

**les applications qui étaient déjà en cours d'exécution pendant le basculement principal ne sont pas affectées.** \*\*

\*\* <https://spark.apache.org/docs/latest/spark-standalone.html#standby-masters-with-zookeeper>

zookeeper quorum hosts

## SCÉNARIOS DE TEST

## TEST 1 :

Étape 1 : lancement cluster ZK

\*\$ zkServer.sh start #sur chaque nœud du cluster

Étape 2 : lancement cluster SPARK

\*\$ start-master.sh #sur chaque master

## Résultat ÉTAPES 1 + 2 :

un seul master est élu **ALIVE** et les 4 autres sont en **StandBy**

Étape 3 : démarrer tous les workers

\$ start-worker.sh spark://192.168.1.10:7077, 192.168.1.20:7077,  
192.168.1.30:7077, 192.168.1.40:7077, 192.168.1.50:7077

## Résultat final :

Tous les workers se connectent au même master (qui est en **ALIVE**)



## Spark Master at spark://localhost:7077

URL: spark://localhost:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: STANDBY

192.168.1.30



## Spark Master at spark://localhost:7077

URL: spark://localhost:7077  
Alive Workers: 2  
Cores in use: 5 Total, 0 Used  
Memory in use: 10.0 GiB Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

192.168.1.10

### Workers (2)

Worker Id
worker-20210623095930-192.168.1.60-43139
worker-20210623095954-192.168.1.87-34199

## Les WORKERS



## Spark Master at spark://localhost:7077

URL: spark://localhost:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: STANDBY

192.168.1.20

### Workers (0)

Worker Id	Address	State

### Running Applications (0)

## SCÉNARIOS DE TEST

### TEST 2 :

Mêmes étapes (1+2+3) que le test 1 :

- Lancement cluster ZK
- lancement cluster SPARK
- Lancement des workers

Étape 4 : arrêter le master en cours (qui est en ALIVE)

#### Résultat:

- un nouveau master est élu :  
il passe en mode RECOVERING (figure page suivante ) et puis en mode ALIVE
- l'un des workers se connecte au nouveau master et un autre passe en status unknown et puis dead

**URL:** spark://localhost:7077

**Alive Workers:** 1

**Cores in use:** 4 Total, 0 Used

**Memory in use:** 6.6 GiB Total, 0.0 B Used

**Resources in use:**

**Applications:** 0 Running, 0 Completed

**Drivers:** 0 Running, 0 Completed

**Status:** RECOVERING

▼ **Workers (2)**

Worker Id	Address
worker-20210623095930-192.168.1.60-43139	192.168.1.60:43139
worker-20210623095954-192.168.1.87-34199	192.168.1.87:34199

## 2) l'un des workers se met en DEAD après avoir arrêter un master

(Les autres workers continuent à exister et se connectent au nouveau master)

```
==> spark-noureddine-org.apache.spark.deploy.worker.Worker-1-ASUS-PC.out <==  
at io.netty.channel.nio.AbstractNioChannel$AbstractNioUnsafe.finishConnect(AbstractNioChannel.java:334)  
at io.netty.channel.nio.NioEventLoop.processSelectedKey(NioEventLoop.java:702)  
at io.netty.channel.nio.NioEventLoop.processSelectedKeysOptimized(NioEventLoop.java:650)  
at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:576)  
at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:493)  
at io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:989)  
at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)  
at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)  
at java.lang.Thread.run(Thread.java:748)  
21/06/23 10:56:31 INFO Worker: Master has changed, new master is at spark://localhost:7077  
  
2023-06-21T10:53:56.000+00:00 [Worker-1] INFO org.apache.spark.SparkException - Exception thrown in awaitResult:  
    at org.apache.spark.util.ThreadUtils$.awaitResult(Thredutils.scala:301)  
    at org.apache.spark.rpc.RpcTimeout.awaitResult(RpcTimeout.scala:75)  
    at org.apache.spark.rpc.RpcEnv.setupEndpointRefByURI(RpcEnv.scala:101)  
    at org.apache.spark.rpc.RpcEnv.setupEndpointRef(RpcEnv.scala:109)  
    at org.apache.spark.deploy.worker.Worker$$anon$2.run(Worker.scala:355)  
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)  
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)  
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)  
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)  
    at java.lang.Thread.run(Thread.java:748)  
Caused by: java.io.IOException: Failed to connect to localhost/192.168.1.60:7077  
    at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:287)  
        at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:218)  
        at org.apache.spark.network.client.TransportClientFactory.createClient(TransportClientFactory.java:230)  
        at org.apache.spark.network.client.NettyRpcEnv$$anon$1.call(Outbox.scala:202)  
        at org.apache.spark.network.client.NettyRpcEnv$$anon$1.call(Outbox.scala:198)  
        at org.apache.spark.util.Thredutils$.awaitResult(Thredutils.scala:301)  
        at org.apache.spark.rpc.RpcTimeout.awaitResult(RpcTimeout.scala:75)  
        at org.apache.spark.rpc.RpcEnv.setupEndpointRefByURI(RpcEnv.scala:101)  
        at org.apache.spark.rpc.RpcEnv.setupEndpointRef(RpcEnv.scala:109)  
        at org.apache.spark.deploy.worker.Worker$$anon$2.run(Worker.scala:355)  
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)  
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)  
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)  
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)  
        at java.lang.Thread.run(Thread.java:748)  
Caused by: io.netty.channel.AbstractChannel$AnnotatedConnectException: Connection refused: localhost/192.168.1.60:7077  
    at io.netty.channel.AbstractChannel$AnnotatedConnectException.(AbstractChannel.java:716)  
    at io.netty.channel.nio.NioEventLoop.doFinishConnect(NioSocketChannel.java:330)  
    at io.netty.channel.nio.NioEventLoop.finishConnect(AbstractNioChannel.java:334)  
    at io.netty.channel.nio.NioEventLoop.processSelectedKeyOptimized(NioEventLoop.java:650)  
    at io.netty.channel.nio.NioEventLoop.processSelectedKeys(NioEventLoop.java:576)  
    at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:493)  
    at io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:989)  
    at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)  
    at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)  
    at java.lang.Thread.run(Thread.java:748)  
21/06/23 10:54:32 ERROR Worker: RECEIVED SIGNAL TERM
```

Problème résolu (temporairement) après l'arrêt du Firewall, qui bloque les ports

# Problèmes rencontrés

1-tous les serveurs sont en mode Alive.

Dans un cas normal un master doit être en mode Alive et StandBy pour les masters en écoute.

Solution :

- .Refaire la configuration sur tous les masters (spark-env.sh + zoo.cfg)
- .Même liste de serveurs
- .Même dossier de récupération (/opt/spark/spark\_recovery)

## **Could not find or load main class org.apache.zookeeper.server.quorum.QuorumPeerMain**

Cela se produit en raison de l'indisponibilité du répertoire

lib qui contient le fichier zookeeper-3.5.6.jar.

Au lieu d'extraire/installer apache-zookeeper-3.5.6.tar.gz

sur chaque nœud du cluster, nous devons procéder

avec apache-zookeeper-3.5.6-**bin**.tar.gz. qui contient tous les fichiers jar requis dans le répertoire lib.



# paxos algorithm

<https://zookeeper.apache.org/doc/r3.4.13/zookeeperInternals.html>

Chaque serveur commence dans l'état LOOKING, où il doit

- \* soit élire un nouveau leader,

- \* soit trouver celui qui existe.

**Si un leader existe déjà :** d'autres serveurs indiquent au nouveau quel serveur est le leader.

À ce stade, le nouveau serveur se connecte au leader

et s'assure que son propre état est cohérent avec l'état du leader.

**Si un ensemble de serveurs**, cependant, sont tous dans l'état LOOKING,

ils doivent communiquer pour élire un leader. Ils échangent des messages pour converger vers un choix commun pour le leader.

Le serveur qui remporte cette élection passe à l'état LEADING, tandis que les autres serveurs de l'ensemble passent à l'état FOLLOWING.

Les messages d'élection de chef sont appelés notifications d'élection de chef, ou simplement notifications.

Classe JAVA pour l'élection du leader:

Le protocole est extrêmement simple. Lorsqu'un <https://zookeeper.apache.org/doc/r3.5.5/api/org/apache/zookeeper/server/quorum/Leader.html> de l'en-

Le message d'élection du leader est également appelé avis d'élection de leader.

Lorsqu'un nœud entre dans l'état LOOKING, il envoie une notification à chacun des autres nœuds.

La notification contient son élection actuelle (vote), consistant en sid (Id de serveur) et zxid (ID de transaction zookeeper).

Par exemple, (1, 5) le représentant de notification est envoyé par le nœud avec l'identifiant de serveur de 1 et le dernier zxid de 5.

Lorsqu'une notification de vote est reçue, le nœud modifiera son vote selon les règles suivantes :

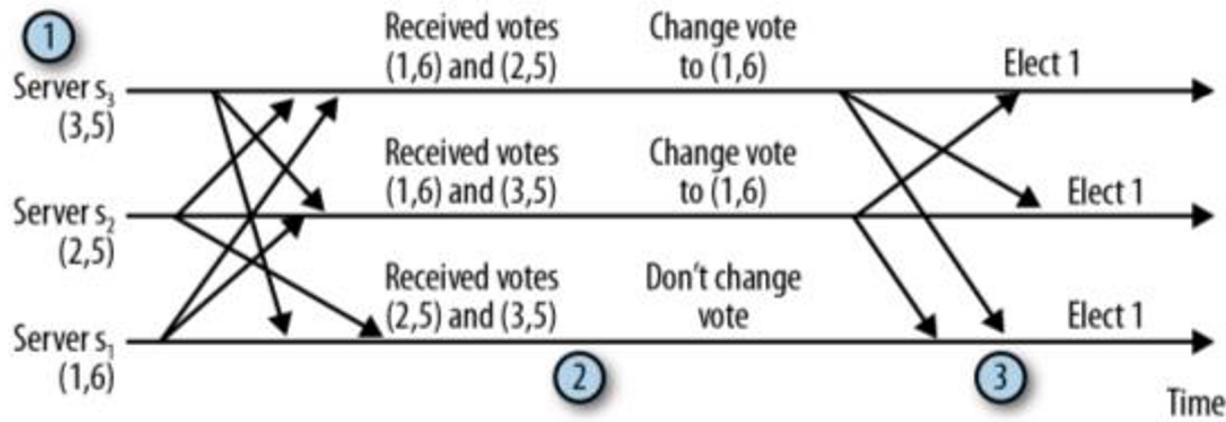
- 1) Définir voteld et voteZxid pour représenter l'ID de vote du nœud actuel, myZxid et mySid représentent la valeur du nœud actuel.
- 2) si  $\text{voteZxid} > \text{myZxid}$  ou ( $\text{voteZxid} == \text{myZxid}$  et  $\text{voteld} > \text{mySid}$ ), le récepteur conservera le vote en cours, c'est-à-dire (mySid, myZxid).
- 3) Sinon, le récepteur changera son propre vote, le vote est (mySid, myZxid), ce qui signifie que je ne suis pas d'accord.

En gros, le vote récemment lancé l'emportera, qui a un grand zxid, qui gagne, si le zxid est le même, on ne peut comparer

Ensuite, si le récepteur accepte l'élection reçue, il enverra la notification aux autres nœuds.

Une fois qu'un nœud reçoit plus de la moitié des nœuds avec le même vote (sid, zxid), le nœud déclare LEADER élu.

Si LEADER est le nœud lui-même, il assumera la fonctionnalité du rôle de leader. Sinon, il deviendra un Follower, et se con-



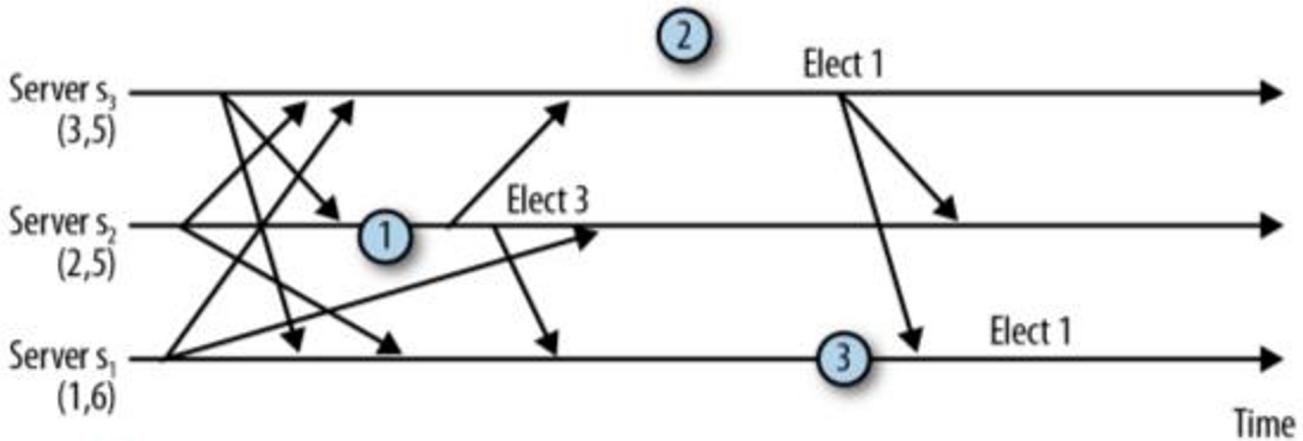
- ① Server  $s_1$  starts with vote (1,6), server  $s_2$  with (2,5) and server  $s_3$  with (3,5).
- ② Servers  $s_2$  and  $s_3$  change their vote to (1,6) and send a new batch of notifications.
- ③ All three servers receive the same vote from a quorum and elect server  $s_1$ .

Étape 1 : les trois nœuds enverront leur vote électoral respectif à d'autres nœuds ; S1(1,6), S2(2,5), S3(3,5)

Étape 2 : Grâce aux règles dont nous venons de parler, les zxids de S2 et S3 sont relativement petits, do

Étape 3 : les trois nœuds reçoivent plus de la moitié des (1,6) élections. S1 a donc été choisi comme leader

Tous les processus électoraux ne sont pas aussi fluides que le processus ci-dessus, souvent en raison de retards du réseau.



- ① Server  $s_2$  receives vote  $(3,5)$  and changes its vote, forming a quorum. It elects server  $s_3$ .
- ② Server  $s_3$  receives vote  $(1,6)$ , but it takes some time to send a new batch of notifications.
- ③ Server  $s_1$  elects itself leader once it receives the vote of server  $s_3$ .

Étape 1 : en raison du retard du réseau S1 à S2, lorsque S2 est soumis au vote de S3 (3, 5) et est d'accord avec (3, 5).

Étape 2 : S3 est soumis à (1,6), mais en raison du retard du réseau, il envoie une notification à S1 est relativement le

Étape 3 : S1 reçoit le vote de S3, S1 a également plus de la moitié des votes, et S1 est également sélectionné comme

Dans l'étape 2, S3 ne répond pas à S2 et propose de se choisir comme Leader.

Étant donné que le vote est passé à (1, 6), S3 ne demandera pas à S2 les exigences du chef.

Tous les S2 attendront un délai d'attente et S1 finira par devenir un leader.



# TESTS

# Si pas quorum pas cluster Zookeeper

```
[node1@node1PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node1@node1PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Error contacting service. It is probably not running.  
[node1@node1PC ~]$
```

Pas de Quorum

```
[node2@node2PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node2@node2PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Error contacting service. It is probably not running.  
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC ~]$
```

```
[node1@node1PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node1@node1PC ~]$
```

Quorum OK

```
[node3@node3PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node3@node3PC ~]$
```

```
[node2@node2PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: leader  
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC ~]$
```

Disable this terminal from "MultiExec" mode

**Q :**

**Voir s'il y a un ordre dans lequel le choix des masters se fait**

**Maching entre le fichier Myid et l'ordre des élections des masters**

```
[node1@node1PC ~]$ zkServer.sh start | grep Starting  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node1@node1PC ~]$ zkServer.sh status | grep Mode:  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Mode: follower  
[node1@node1PC ~]$ zkServer.sh stop | grep Stopping  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Stopping zookeeper ... STOPPED  
[node1@node1PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC ~]$ zkServer.sh start | grep Starting  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node3@node3PC ~]$ zkServer.sh status | grep Mode:  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Mode: follower  
[node3@node3PC ~]$ zkServer.sh stop | grep Stopping  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Stopping zookeeper ... STOPPED  
[node3@node3PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC ~]$ zkServer.sh start | grep Starting  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node5@node5PC ~]$ zkServer.sh status | grep Mode:  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Mode: leader  
[node5@node5PC ~]$ zkServer.sh stop | grep Stopping  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Stopping zookeeper ... STOPPED  
[node5@node5PC ~]$
```

```
[node2@node2PC ~]$ zkServer.sh start | grep Starting  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node2@node2PC ~]$ zkServer.sh status | grep Mode:  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Mode: follower  
[node2@node2PC ~]$ zkServer.sh stop | grep Stopping  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Stopping zookeeper ... STOPPED  
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC ~]$ zkServer.sh start | grep Starting  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node4@node4PC ~]$ zkServer.sh status | grep Mode:  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Mode: follower  
[node4@node4PC ~]$ zkServer.sh stop | grep Stopping  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Stopping zookeeper ... STOPPED  
[node4@node4PC ~]$
```

Disable this terminal from "MultiExec" mode

Démarrage du cluster avant le changement du f  
node1 => id =1  
node2 => id =2  
...  
node5 => id=5  
Résultat = node5 est toujours le leader

```
[node1@node1PC ~]$ cat /opt/zookeeper/data/myid
4
[node1@node1PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node2@node2PC ~]$ cat /opt/zookeeper/data/myid
5
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC ~]$ cat /opt/zookeeper/data/myid
1
[node3@node3PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC ~]$ cat /opt/zookeeper/data/myid
3
[node4@node4PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC ~]$ cat /opt/zookeeper/data/myid
2
[node5@node5PC ~]$
```

**Changements du fichier Myid dans chaque machine**  
**/opt/zookeeper/data/myid**

```
Error contacting service. It is probably not running.
```

```
[node1@node1PC ~]$ zkServer.sh start
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Starting zookeeper ... STARTED
```

```
[node1@node1PC ~]$ zkServer.sh status
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Client port found: 2181. Client address: localhost. Client SSL: false.
```

```
Error contacting service. It is probably not running.
```

```
[node1@node1PC ~]$
```

Disable this terminal from "MultiExec" mode

```
Error contacting service. It is probably not running.
```

```
[node3@node3PC ~]$ zkServer.sh start
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Starting zookeeper ... STARTED
```

```
[node3@node3PC ~]$ zkServer.sh status
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Client port found: 2181. Client address: localhost. Client SSL: false.
```

```
Error contacting service. It is probably not running.
```

```
[node3@node3PC ~]$
```

Disable this terminal from "MultiExec" mode

```
Error contacting service. It is probably not running.
```

```
[node5@node5PC ~]$ zkServer.sh start
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Starting zookeeper ... STARTED
```

```
[node5@node5PC ~]$ zkServer.sh status
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Client port found: 2181. Client address: localhost. Client SSL: false.
```

```
Error contacting service. It is probably not running.
```

```
[node5@node5PC ~]$
```

```
Error contacting service. It is probably not running.
```

```
[node2@node2PC ~]$ zkServer.sh start
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Starting zookeeper ... STARTED
```

```
[node2@node2PC ~]$ zkServer.sh status
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Client port found: 2181. Client address: localhost. Client SSL: false.
```

```
Error contacting service. It is probably not running.
```

```
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
Error contacting service. It is probably not running.
```

```
[node4@node4PC ~]$ zkServer.sh start
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Starting zookeeper ... STARTED
```

```
[node4@node4PC ~]$ zkServer.sh status
```

```
/usr/bin/java
```

```
ZooKeeper JMX enabled by default
```

```
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
```

```
Client port found: 2181. Client address: localhost. Client SSL: false.
```

```
Error contacting service. It is probably not running.
```

```
[node4@node4PC ~]$
```

1) Contenu myid changé

2) impossible de démarrer le cluster ZK

3) il faut changer le fichier zoo.cfg pour qu'il correspon

192.168.1.10 => 4

192.168.1.20 => 5

... voir capture précédente

```
[node1@node1PC data]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node1@node1PC data]$ cat myid
4
[node1@node1PC data]$
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC data]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node3@node3PC data]$ cat myid
1
[node3@node3PC data]$
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC data]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node5@node5PC data]$ cat myid
2
[node5@node5PC data]$
```

```
[node2@node2PC data]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
[node2@node2PC data]$ cat myid
5
[node2@node2PC data]$
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC data]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node4@node4PC data]$ cat myid
3
[node4@node4PC data]$
```

Disable this terminal from "MultiExec" mode

Démarrage du cluster (tous les nœuds ont été démarrés)  
Résultat de plusieurs tests :  
Le node2 est toujours leader  
(id = 5) le plus grand id

```
[node1@node1PC ~]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node1@node1PC ~]$
[node1@node1PC ~]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: leader
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC ~]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node3@node3PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC ~]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node5@node5PC ~]$
```

```
[node2@node2PC ~]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Error contacting service. It is probably not running.
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC ~]$ zkServer.sh status
/usr/bin/java
ZooKeeper JMX enabled by default
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg
Client port found: 2181. Client address: localhost. Client SSL: false.
Mode: follower
[node4@node4PC ~]$
```

**Node2 arrêté (id= 5)**  
**Le node1 devient leader (id = 4)**  
**Le plus gros ID du reste du cluster**

```
Mode: leader  
[node1@node1PC ~]$ zkServer.sh stop  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Stopping zookeeper ... STOPPED  
[node1@node1PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Error contacting service. It is probably not running.  
[node1@node1PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node3@node3PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node3@node3PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node5@node5PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node5@node5PC ~]$
```

```
[node2@node2PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Error contacting service. It is probably not running.  
[node2@node2PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Error contacting service. It is probably not running.  
[node2@node2PC ~]$
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node4@node4PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: leader  
[node4@node4PC ~]$
```

Disable this terminal from "MultiExec" mode

**Node1 arrêté (id= 4)**  
**Le node4 devient leader (id = 3)**  
**Le plus gros ID du reste du cluster**

Question:

Est ce que le leader ZK et le même leader  
(master Alive) dans spark?

```
[node1@node1PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node1@node1PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node1@node1PC ~]$ █
```

Disable this terminal from "MultiExec" mode

```
[node3@node3PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node3@node3PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: leader  
[node3@node3PC ~]$ █
```

Disable this terminal from "MultiExec" mode

```
[node5@node5PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node5@node5PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node5@node5PC ~]$ █
```

```
[node2@node2PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node2@node2PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node2@node2PC ~]$ █
```

Disable this terminal from "MultiExec" mode

```
[node4@node4PC ~]$ zkServer.sh start  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Starting zookeeper ... STARTED  
[node4@node4PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/..../conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node4@node4PC ~]$ █
```

Disable this terminal from "MultiExec" mode

**Test 1 :**  
**Node 3 > leader ZK**

The image shows four browser windows displaying the Apache Spark master interface. Each window has a header bar with the Apache logo and version 3.1.1.

- Node 3 (Leftmost):** Spark Master at spark://node3PC:7077. Status: STANDBY.
- Node 1 (Centered):** Spark Master at spark://node1PC:7077. Status: ALIVE. This window is circled in green and highlighted with a yellow overlay containing the text "Node1 MASTER ALIVE (node3 leader Zookeeper)".
- Node 2 (Second from Right):** Spark Master at spark://node2PC:7077. Status: STANDBY.
- Node 4 (Rightmost):** Spark Master at spark://node4PC:7077. Status: STANDBY.

**Node 1 Master Details:**

- URL:** spark://node1PC:7077
- Alive Workers:** 0
- Cores in use:** 0 Total, 0 Used
- Memory in use:** 0.0 B Total, 0.0 B Used
- Resources in use:**
- Applications:** 0 Running, 0 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

**Node 2 Standby Details:**

- URL:** spark://node2PC:7077
- Alive Workers:** 0
- Cores in use:** 0 Total, 0 Used
- Memory in use:** 0.0 B Total, 0.0 B Used
- Resources in use:**
- Applications:** 0 Running, 0 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** STANDBY

**Node 3 Standby Details:**

- URL:** spark://node3PC:7077
- Alive Workers:** 0
- Cores in use:** 0 Total, 0 Used
- Memory in use:** 0.0 B Total, 0.0 B Used
- Resources in use:**
- Applications:** 0 Running, 0 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** STANDBY

**Node 4 Standby Details:**

- URL:** spark://node4PC:7077
- Alive Workers:** 0
- Cores in use:** 0 Total, 0 Used
- Memory in use:** 0.0 B Total, 0.0 B Used
- Resources in use:**
- Applications:** 0 Running, 0 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** STANDBY

```
[node1@node1PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/./conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node1@node1PC ~]$
```

```
[node2@node2PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/./conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node2@node2PC ~]$
```

```
[node3@node3PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/./conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Error contacting service. It is probably not running.  
[node3@node3PC ~]$
```

```
[node4@node4PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/./conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: follower  
[node4@node4PC ~]$
```

```
[node5@node5PC ~]$ zkServer.sh status  
/usr/bin/java  
ZooKeeper JMX enabled by default  
Using config: /opt/zookeeper/bin/./conf/zoo.cfg  
Client port found: 2181. Client address: localhost. Client SSL: false.  
Mode: leader  
[node5@node5PC ~]$
```

**Node3, leader zookeeper est arrêté**

Après l'arrêt du leader ZK (node3), le master node1 (spark) s'est arrêté également. Le node5 est devenu le Master Alive (spark)

Apache Spark 3.1.1

## Spark Master at spark://node3PC:7077

URL: spark://node3PC:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: STANDBY

Workers (0)

Apache Spark 3.1.1

## Spark Master at spark://node2PC:7077

URL: spark://node2PC:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: STANDBY

Réessayer

Apache Spark 3.1.1

## Spark Master at spark://node2PC:7077

URL: spark://node2PC:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: STANDBY

Apache Spark 3.1.1

## Spark Master at spark://node4PC:7077

URL: spark://node4PC:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: STANDBY

Workers (0)

Apache Spark 3.1.1

## Spark Master at spark://node5PC:7077

URL: spark://node5PC:7077  
Alive Workers: 0  
Cores in use: 0 Total, 0 Used  
Memory in use: 0.0 B Total, 0.0 B Used  
Resources in use:  
Applications: 0 Running, 0 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

Workers (0)

Remarque:

Un nœud qui est un Master spark peut ne pas être un membre du cluster zookeeper, mais il pourra être choisi comme MASTER ALIVE par NB : il faut rajouter les liens du cluster ZK dans spark-env.sh

## Conclusion :

Le nœud élu leader dans un cluster Zookeeper n'est pas forcement le même noeud Master Alive de spark, en revanche la disponibilité d'un leader ZK et d'un Master Alive spark est liée, si le leader est stopper le master Alive sera stoppé automatiquement.



ZooKeeper est simple.

Il

permet aux processus distribués de se coordonner via un espace de noms hiérarchique partagé qui est organisé de la même manière qu'un système de fichiers standard.

L'espace de noms se compose de registres de données - appelés znodes, dans le jargon ZooKeeper - et ceux-ci sont similaires aux fichiers et aux répertoires.

Contrairement à un système de fichiers typique, conçu pour le stockage, les données ZooKeeper sont conservées en mémoire, ce qui signifie que ZooKeeper peut atteindre un débit élevé et des nombres de latence faibles.

La mise en œuvre de ZooKeeper met l'accent sur un accès hautement performant, hautement disponible et strictement ordonné.

Les aspects de performance de ZooKeeper signifient qu'il peut être utilisé dans de grands systèmes distribués.

Les aspects de fiabilité l'empêchent d'être un point de défaillance unique.

L'ordre strict signifie que des primitives de synchronisation sophistiquées peuvent être implémentées chez le client.

### **3 - Voir comment zookeeper stocke les metadata et de quelle maniere.**

Zookeeper conserve toutes ces données dans des fichiers journaux sur disque. Le chemin est spécifié par le paramètre dataDir dans zookeeper.properties. Par défaut avec Kafka, c'est /tmp/zookeeper.

L'emplacement pour stocker les snapshots de la base de données en mémoire et,

sauf indication contraire, le journal des transactions des mises à jour de la base de données. \*\*

ZooKeeper stocke ses données dans un répertoire de données et son journal des transactions dans un répertoire du journal des transactions.

Par défaut, ces deux répertoires sont identiques.

Le serveur peut (et doit) être configuré pour stocker les fichiers journaux des transactions dans un répertoire distinct de celui des fichiers de données.

Le débit augmente et la latence diminue lorsque les journaux de transactions résident sur des périphériques de journal dédiés. \*\*\*

\*\* <https://zookeeper.apache.org/doc/r3.3.3/zookeeperStarted.html>

\*\*\*<https://zookeeper.apache.org/doc/r3.1.2/zookeeperAdmin.html>