第 4 章 Zookeeper 实战

4.1 分布式安装部署

0）集群规划

在 hadoop102、hadoop103 和 hadoop104 三个节点上部署 Zookeeper。

1）解压安装

（1）解压 zookeeper 安装包到/opt/module/目录下

[atguigu@hadoop102 software]$ tar -zxvf zookeeper-3.4.10.tar.gz -C /opt/module/

（2）在/opt/module/zookeeper-3.4.10/这个目录下创建 zkData

mkdir -p zkData

（3）重命名/opt/module/zookeeper-3.4.10/conf 这个目录下的 zoo_sample.cfg 为 zoo.cfg

mv zoo_sample.cfg zoo.cfg

2）配置 zoo.cfg 文件

（1）具体配置

dataDir=/opt/module/zookeeper-3.4.10/zkData

增加如下配置

#########################cluster##########################

server.2=hadoop102:2888:3888

server.3=hadoop103:2888:3888

server.4=hadoop104:2888:3888

（2）配置参数解读

Server.A=B:C:D。

A 是一个数字，表示这个是第几号服务器；

B 是这个服务器的 ip 地址；

C 是这个服务器与集群中的 Leader 服务器交换信息的端口；

D 是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口。

集群模式下配置一个文件 myid，这个文件在 dataDir 目录下，这个文件里面有一个数据就是 A 的值，Zookeeper 启动时读取此文件，拿到里面的数据与 zoo.cfg 里面的配置信息比

较从而判断到底是哪个 server。

3）集群操作

（1）在/opt/module/zookeeper-3.4.10/zkData 目录下创建一个 myid 的文件

touch myid

添加 myid 文件，注意一定要在 linux 里面创建，在 notepad++里面很可能乱码

（2）编辑 myid 文件

vi myid

在文件中添加与 server 对应的编号：如 2

（3）拷贝配置好的 zookeeper 到其他机器上

scp -r zookeeper-3.4.10/ root@hadoop103.atguigu.com:/opt/app/

scp -r zookeeper-3.4.10/ root@hadoop104.atguigu.com:/opt/app/

并分别修改 myid 文件中内容为 3、4

（4）分别启动 zookeeper

[root@hadoop102 zookeeper-3.4.10]# bin/zkServer.sh start

[root@hadoop103 zookeeper-3.4.10]# bin/zkServer.sh start

[root@hadoop104 zookeeper-3.4.10]# bin/zkServer.sh start

（5）查看状态

[root@hadoop102 zookeeper-3.4.10]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: follower

[root@hadoop103 zookeeper-3.4.10]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: leader

[root@hadoop104 zookeeper-3.4.5]# bin/zkServer.sh status

JMX enabled by default

Using config: /opt/module/zookeeper-3.4.10/bin/../conf/zoo.cfg

Mode: follower

4.2 客户端命令行操作

命令基本语法	功能描述
help	显示所有操作命令
ls path [watch]	使用 ls 命令来查看当前znode中所包含的内容
ls2 path [watch]	查看当前节点数据并能看到更新次数等数据
create	普通创建 -s 含有序列 -e 临时（重启或者超时消失）
get path [watch]	获得节点的值
set	设置节点的具体值
stat	查看节点状态
delete	删除节点
rmr	递归删除节点

1）启动客户端

[atguigu@hadoop103 zookeeper-3.4.10]$ bin/zkCli.sh

2）显示所有操作命令

[zk: localhost:2181(CONNECTED) 1] help

3）查看当前 znode 中所包含的内容

[zk: localhost:2181(CONNECTED) 0] ls /

[zookeeper]

4）查看当前节点数据并能看到更新次数等数据

[zk: localhost:2181(CONNECTED) 1] ls2 /

[zookeeper]

cZxid = 0x0

ctime = Thu Jan 01 08:00:00 CST 1970

mZxid = 0x0

mtime = Thu Jan 01 08:00:00 CST 1970

pZxid = 0x0

cversion = -1

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 0

numChildren = 1

5）创建普通节点

[zk: localhost:2181(CONNECTED) 2] create /app1 "hello app1"

Created /app1

[zk: localhost:2181(CONNECTED) 4] create /app1/server101 "192.168.1.101"

Created /app1/server101

6）获得节点的值

[zk: localhost:2181(CONNECTED) 6] get /app1

hello app1

cZxid = 0x20000000a

ctime = Mon Jul 17 16:08:35 CST 2017

mZxid = 0x20000000a

mtime = Mon Jul 17 16:08:35 CST 2017

pZxid = 0x20000000b

cversion = 1

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 10

numChildren = 1

[zk: localhost:2181(CONNECTED) 8] get /app1/server101

192.168.1.101

cZxid = 0x20000000b

ctime = Mon Jul 17 16:11:04 CST 2017

mZxid = 0x20000000b

mtime = Mon Jul 17 16:11:04 CST 2017

pZxid = 0x20000000b

cversion = 0

dataVersion = 0

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 13

numChildren = 0

7）创建短暂节点

[zk: localhost:2181(CONNECTED) 9] create -e /app-emphemeral 8888

（1）在当前客户端是能查看到的

[zk: localhost:2181(CONNECTED) 10] ls /

[app1, app-emphemeral, zookeeper]

（2）退出当前客户端然后再重启客户端

[zk: localhost:2181(CONNECTED) 12] quit

[atguigu@hadoop104 zookeeper-3.4.10]$ bin/zkCli.sh

（3）再次查看根目录下短暂节点已经删除

[zk: localhost:2181(CONNECTED) 0] ls /

[app1, zookeeper]

8）创建带序号的节点

（1）先创建一个普通的根节点 app2

[zk: localhost:2181(CONNECTED) 11] create /app2 "app2"

（2）创建带序号的节点

[zk: localhost:2181(CONNECTED) 13] create -s /app2/aa 888

Created /app2/aa0000000000

[zk: localhost:2181(CONNECTED) 14] create -s /app2/bb 888

Created /app2/bb0000000001

[zk: localhost:2181(CONNECTED) 15] create -s /app2/cc 888

Created /app2/cc0000000002

如果原节点下有 1 个节点，则再排序时从 1 开始，以此类推。

[zk: localhost:2181(CONNECTED) 16] create -s /app1/aa 888

Created /app1/aa0000000001

9）修改节点数据值

[zk: localhost:2181(CONNECTED) 2] set /app1 999

10）节点的值变化监听

（1）在 104 主机上注册监听/app1 节点数据变化

[zk: localhost:2181(CONNECTED) 26] get /app1 watch

（2）在 103 主机上修改/app1 节点的数据

[zk: localhost:2181(CONNECTED) 5] set /app1 777

（3）观察 104 主机收到数据变化的监听

WATCHER::

WatchedEvent state:SyncConnected type:NodeDataChanged path:/app1

11）节点的子节点变化监听（路径变化）

（1）在 104 主机上注册监听/app1 节点的子节点变化

[zk: localhost:2181(CONNECTED) 1] ls /app1 watch

[aa0000000001, server101]

（2）在 103 主机/app1 节点上创建子节点

[zk: localhost:2181(CONNECTED) 6] create /app1/bb 666

Created /app1/bb

（3）观察 104 主机收到子节点变化的监听

WATCHER::

WatchedEvent state:SyncConnected type:NodeChildrenChanged path:/app1

12）删除节点

[zk: localhost:2181(CONNECTED) 4] delete /app1/bb

13）递归删除节点

[zk: localhost:2181(CONNECTED) 7] rmr /app2

14）查看节点状态

[zk: localhost:2181(CONNECTED) 12] stat /app1

cZxid = 0x20000000a

ctime = Mon Jul 17 16:08:35 CST 2017

mZxid = 0x200000018

mtime = Mon Jul 17 16:54:38 CST 2017

pZxid = 0x20000001c

cversion = 4

dataVersion = 2

aclVersion = 0

ephemeralOwner = 0x0

dataLength = 3

numChildren = 2

4.3 API 应用

4.3.1 eclipse 环境搭建

1）创建一个 Maven 工程

2）添加 pom 文件

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.8.2</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.apache.zookeeper/zookeeper -->
    <dependency>
        <groupId>org.apache.zookeeper</groupId>
        <artifactId>zookeeper</artifactId>
        <version>3.4.10</version>
    </dependency>
</dependencies>
```

3）拷贝 log4j.properties 文件到项目根目录

log4j.properties

4.3.2 创建 ZooKeeper 客户端

```
private static String connectString = "hadoop102:2181,hadoop103:2181,hadoop104:2181";
```

```
    private static int sessionTimeout = 2000;
    private ZooKeeper zkClient = null;

    @Before
    public void init() throws Exception {

    zkClient = new ZooKeeper(connectString, sessionTimeout, new Watcher() {
            @Override
            public void process(WatchedEvent event) {
                // 收到事件通知后的回调函数（用户的业务逻辑）
                System.out.println(event.getType() + "--" + event.getPath());

                // 再次启动监听
                try {
                    zkClient.getChildren("/", true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

4.3.3 创建子节点

```
    // 创建子节点
    @Test
    public void create() throws Exception {
        // 数据的增删改查
        // 参数 1：要创建的节点的路径； 参数 2：节点数据 ； 参数 3：节点权限 ；参数 4：节点的类型
        String nodeCreated = zkClient.create("/eclipse", "hello zk".getBytes(), Ids.OPEN_ACL_UNSAFE,CreateMode.PERSISTENT);
    }
```

4.3.4 获取子节点并监听

```
    // 获取子节点
    @Test
    public void getChildren() throws Exception {
        List<String> children = zkClient.getChildren("/", true);

        for (String child : children) {
            System.out.println(child);
        }

        // 延时阻塞
```

```
        Thread.sleep(Long.MAX_VALUE);
    }
```

4.3.5 判断 znode 是否存在

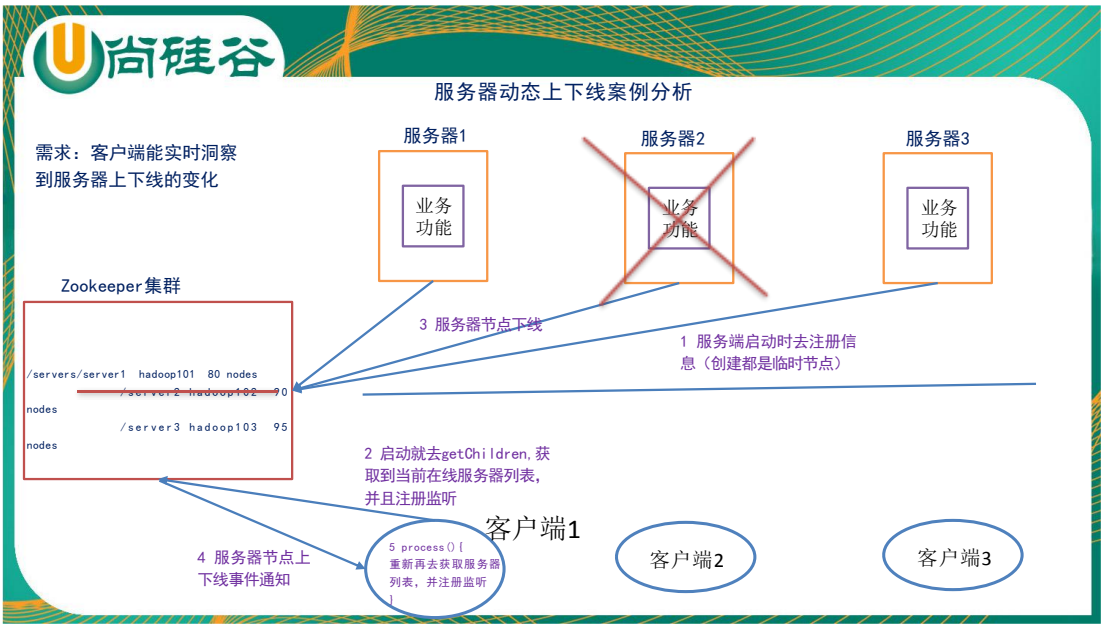
```
    // 判断 znode 是否存在
    @Test
    public void exist() throws Exception {
        Stat stat = zkClient.exists("/eclipse", false);

        System.out.println(stat == null ? "not exist" : "exist");
    }
```

4.4 案例实战

监听服务器节点动态上下线案例

1）需求：某分布式系统中，主节点可以有多台，可以动态上下线，任意一台客户端都能实时感知到主节点服务器的上下线

2）需求分析



3）具体实现：

（0）现在集群上创建/servers 节点

```
[zk: localhost:2181(CONNECTED) 10] create /servers "servers"
Created /servers
```

（1）服务器端代码

```
package com.atguigu.zkcase;
```

```
import java.io.IOException;
import org.apache.zookeeper.CreateMode;
import org.apache.zookeeper.WatchedEvent;
import org.apache.zookeeper.Watcher;
import org.apache.zookeeper.ZooKeeper;
import org.apache.zookeeper.ZooDefs.Ids;

public class DistributeServer {
	private static String connectString = "hadoop102:2181,hadoop103:2181,hadoop104:2181";
	private static int sessionTimeout = 2000;
	private ZooKeeper zk = null;
	private String parentNode = "/servers";

	// 创建到 zk 的客户端连接
	public void getConnect() throws IOException{

		zk = new ZooKeeper(connectString, sessionTimeout, new Watcher() {

			@Override
			public void process(WatchedEvent event) {

			}
		});
	}

	// 注册服务器
	public void registServer(String hostname) throws Exception{
		String create = zk.create(parentNode + "/server", hostname.getBytes(), Ids.OPEN_ACL_UNSAFE, CreateMode.EPHEMERAL_SEQUENTIAL);

		System.out.println(hostname +" is noline "+ create);
	}

	// 业务功能
	public void business(String hostname) throws Exception{
		System.out.println(hostname+" is working ...");

		Thread.sleep(Long.MAX_VALUE);
	}

	public static void main(String[] args) throws Exception {
		// 获取 zk 连接
		DistributeServer server = new DistributeServer();
```

```
            server.getConnect();

            // 利用 zk 连接注册服务器信息
            server.registServer(args[0]);

            // 启动业务功能
            server.business(args[0]);
        }
}
```

（2）客户端代码

```
package com.atguigu.zkcase;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.zookeeper.WatchedEvent;
import org.apache.zookeeper.Watcher;
import org.apache.zookeeper.ZooKeeper;

public class DistributeClient {
    private static String connectString = "hadoop102:2181,hadoop103:2181,hadoop104:2181";
    private static int sessionTimeout = 2000;
    private ZooKeeper zk = null;
    private String parentNode = "/servers";
    private volatile ArrayList<String> serversList = new ArrayList<>();

    // 创建到 zk 的客户端连接
    public void getConnect() throws IOException {
        zk = new ZooKeeper(connectString, sessionTimeout, new Watcher() {

            @Override
            public void process(WatchedEvent event) {

                // 再次启动监听
                try {
                    getServerList();
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
```

```
    //
    public void getServerList() throws Exception {

        // 获取服务器子节点信息，并且对父节点进行监听
        List<String> children = zk.getChildren(parentNode, true);
        ArrayList<String> servers = new ArrayList<>();

        for (String child : children) {
            byte[] data = zk.getData(parentNode + "/" + child, false, null);

            servers.add(new String(data));
        }

        // 把 servers 赋值给成员 serverList，已提供给各业务线程使用
        serversList = servers;

        System.out.println(serversList);
    }

    // 业务功能
    public void business() throws Exception {
        System.out.println("client is working ...");
        Thread.sleep(Long.MAX_VALUE);
    }

    public static void main(String[] args) throws Exception {

        // 获取 zk 连接
        DistributeClient client = new DistributeClient();
        client.getConnect();

        // 获取 servers 的子节点信息，从中获取服务器信息列表
        client.getServerList();

        // 业务进程启动
        client.business();
    }
}
```