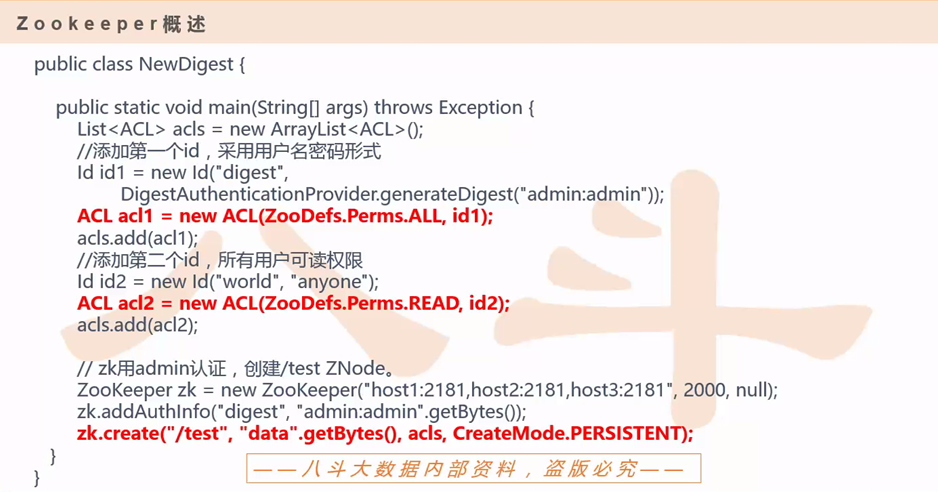
图1：



我们继续我们的内容，然后我们看一下一个比较简单的java代码，还有一个问题有一个节点是否能够绑定多个访问控制是不是？那绑定多个访问控制的话一个叫做访问控制链。

看到图1中标红的第一句代码 ACL这个相当于是创建了一个访问控制，然后名字是acl1，然后后面括号中写的ALL是什么意思呢？这个ALL代表的它拥有所有权限。（如下图2所示）

图2：



acl2拥有读的权限（括号中的READ），但是这个All这个权限和这个READ这个权限是绑定了哪个它具体模式上面呢？（如下图3所示）

图3：



我们来看一下这个acl1，acli1它其实绑定到了一个id1上，这个id1是什么意思呢？它在图1中的第六行（包括空格开始数），id1它是声明了一个digest是不是？digest刚才我们说过了，就是通过用户名和密码的方式进行认证，然后首先要指明你的模式，然后你再指明你的用户名和密码，然后你有了这个模式或者机制，有了它的那个用户并且又绑了一个权限，相当于三元组就已经凑齐了。（如下图4所示）

图4：



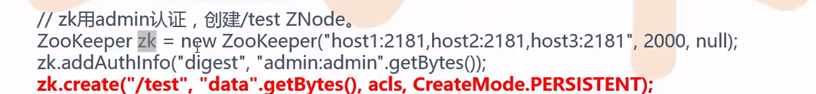
这个时候你再添加第二个ID，第二个ID是什么呢？就是world和anyone这两个，这是对所有的机器或者对所有的用户都有一个读的权限，都是只读的，然后把这两个acl1和acl2 add到了一个acls里面去，acls是一个List集合里面，它创建了一个类似于数组一样的集合（Arralist）这么一个集合，就把这个acl1和acl2就往集合里面添加进去。（如下图5所示）

图5：



然后相当于这个时候把两个访问控制就凑到了一块了，接下来怎么用呢？我们来看这一行红色的代码（如下图6所示）

图6：



ZK就是得到一个Zookeeper一个对象，然后这个对象它里面就有一个create这么一个方法，这个create就是可以拥有了它的一个创建节点的权限，就是可以用这个权限是可以创建节点的，于是它就创建了一个你可以认为在根的目录下创建了一个test这个节点，中间这个data就是字符串的意思，它的字符串就是一个数据，这个数据的内容就是一个data，然后里面的acls，就是说你通过zk对象创建了这个/test节点，数据（data）也给了，并且对这个节点也赋予了一些权限（acls），这个权限（acls）哪里来呢？权限就是由acl1和acl2来的，就是说你这个/test这个节点，可以被两类用户访问或者被两类模式访问，第一个模式可以通过digest这种方式来访问，第二个是可以通过world这种方式来访问。

然后好了，这个时候节点有了，数据有了,权限有了，最后你还要指定一下这个节点是一个什么类型的节点，这里面呢指定的是一个永久性节点(CreateMode.PERSLSTENT)。

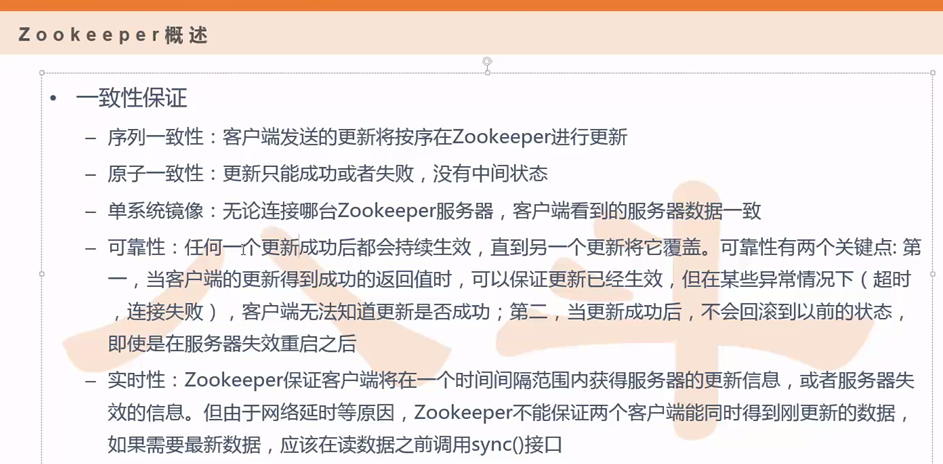
那么我们接下来看一下这个zk对象是怎么来的，这个zk对象是通过一个Zookeeper这么一个类然后创建（new）了一个对象出来，然后在这个Zookeeper这个类里面你可以往里面传一些，通过一些参数比如host1，host2和host3，就是说你这个Zookeeper的集群有很多个机器，然后每个机器上面都会有它的地址是吧？你把地址进行排列，就是这个时候它去访问你的Zookeeper的时候因为你这个Zookeeper这个集群一旦建好了之后，你去访问任何一个节点你得到的数据都是一致的，如果说你当前这个图6中的host1:2181，但是它是从前往后遍历嘛对吧？如果这个host1:2181它能够提供服务的话,那么后面那两个host2和host3就不会再访问了，就当作一个被选方案了对吧，这个host1只要能提供服务你就可以对下进行操作，如果说这个host1出问题了，那你就开始用host2的地址来访问是吧？然后host2也不行，你就用host3来访问这个意思。

那么后面的2000是什么意思呢？2000就相当于是超时时间，那么2000这个超时时间后面的null是什么意思呢？这个是监控的。

那么那个addAuthlnfo()这个是什么意思呢？就是权限认证，所以可能大家addAuthlnfo()和对这个Create这两个方法搞不清楚，首先你要去访问一个Zookeeper的时候，前提是你需要通过一个账户来去，你得注册一下,比如说你要登录一个网站，这个网站你得填入用户名和密码对不对？比如你上京东买东西你得去登录对不对？这时候相当于是你通过什么样子的方式登录，你是通过用户名密码登录还是通过扫二维码的方式登录还是其他的方式登录，这个时候你通过这种digest的用户名密码的方式去登录之后，好了你登录完进去，这个时候你就拥有了一些个权限了，这个权限你就可以对某一个节点进行创建（Create），所以addAuthlnfo（）就是已经拥有的权限而这个Create是可以赋予别人有什么权限，就是不一样的角色。

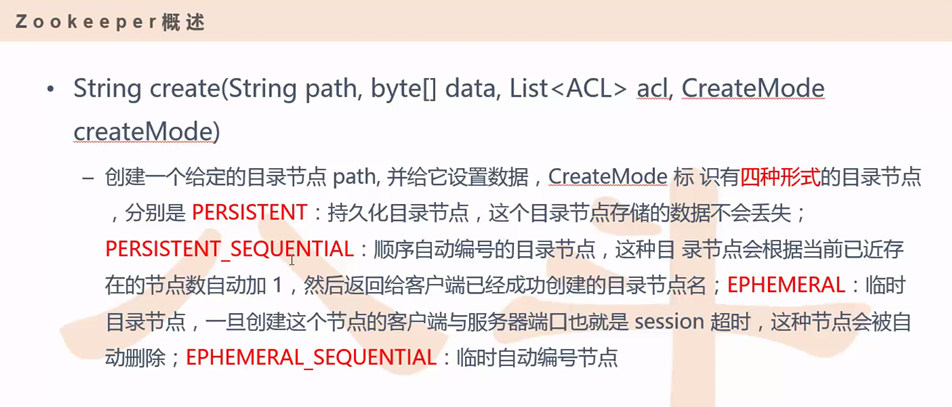
好了然后这一块就是对Zookeeper的一些概述，主要是保证一些一致性，这个大概了解一下就可以了（如下图7所示）

图7：



然后主要是这个比较重要（如下图8所示）

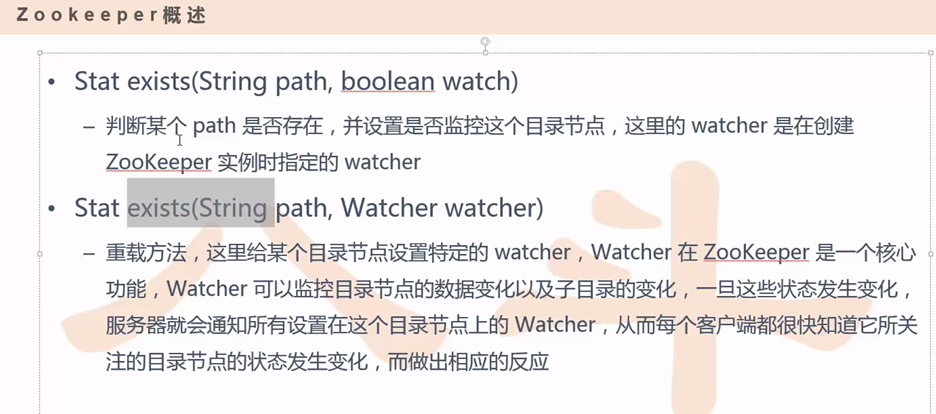
图8：



之前我们说过了这个节点呢有四种，哪四种呢？就是永久型的（PERSISTENT）,永久顺序型的（PERSISTENT\_SEQUENTLAL），临时型的（EPHEMERAL），也就是这个临时顺序型（PERSISTENT\_SEQUENTLAL）呢是不会单独领出来单独做一个节点的形式去用，所以必须和你的永久性节点和你的临时节点进行绑定的方式去用。

这个大家了解一下，这个开发的时候可能会用到（如下图9所示）

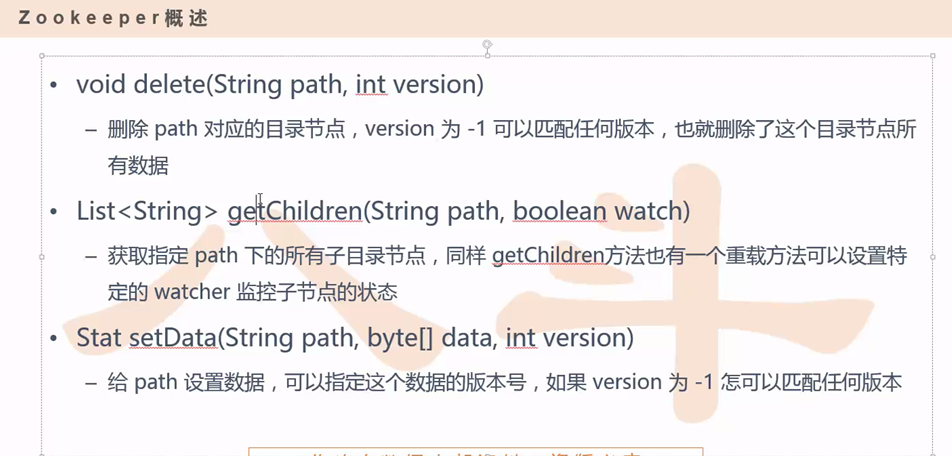
图9：



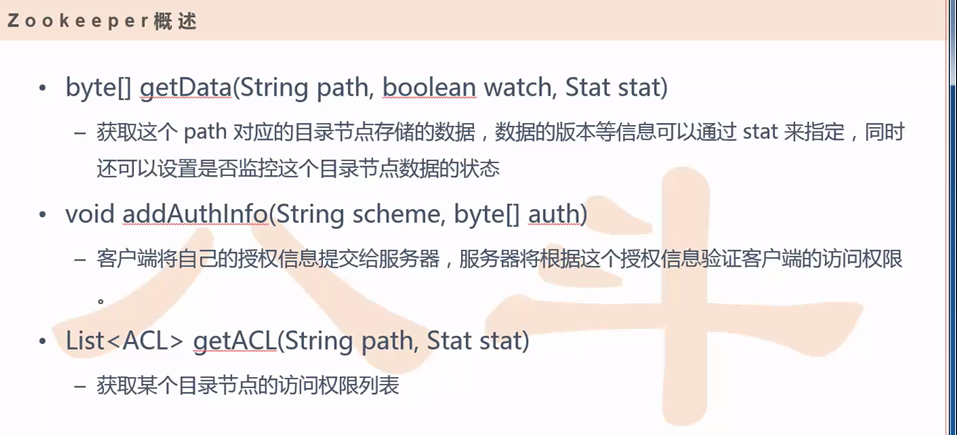
这是exists（）判断一个监控是否存在。

而下图10是delete()判断是否删除（如下图10所示）

图10：



getChildren（）是否这个子节点发生变化，setDat（）a这是进行一些设置数据，getData（）就是读数据，然后addAuthlnfo()就是注册，getACL()就是获得它有哪些权限（如下图11所示）

图11：  


然后setACL就是设置权限（如下图12）

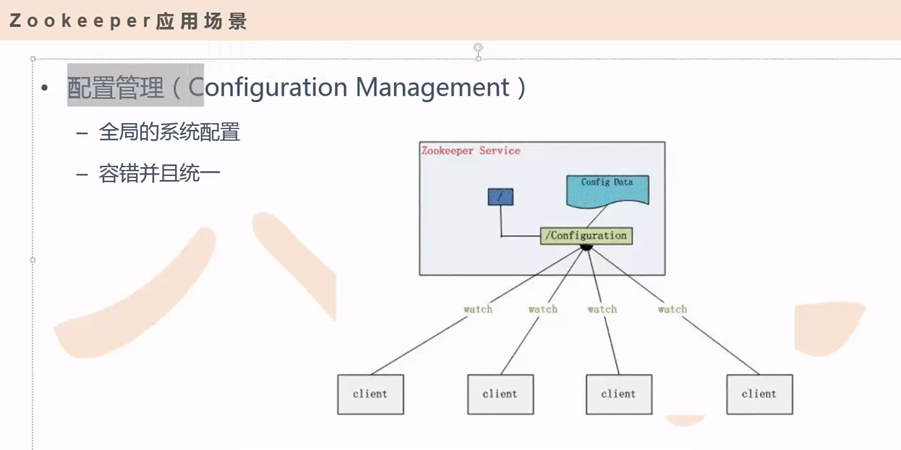
图12：  


像这几种方法就是开发的时候会在涉及。

好了然后接下来我们既然学这个Zookeeper那我们主要的目的是把它用起来对不对？然后你也知道在真正的这个项目里面或者是真正它能够以什么样的形式去用是吧？那相当于接下来就说一些主要的Zookeeper应用场景。

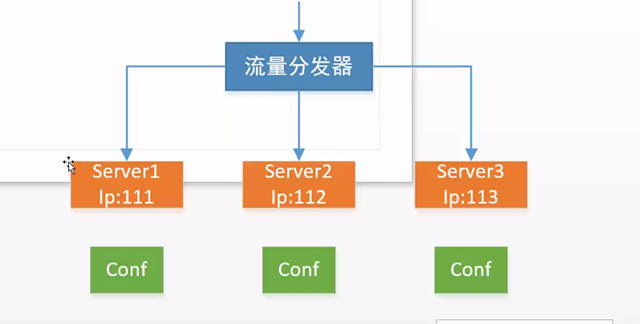
那么第一个应用场景，其实这个刚才大概说了一下，我再重复一下（如下图13所示）

图13：



首先配置管理很重要，配置管理什么意思呢？其实很简单（如下图14所示）

图14：



比如说Server1和Server2和Server3是新浪的三台机器，然后那三个Conf是三个配置。

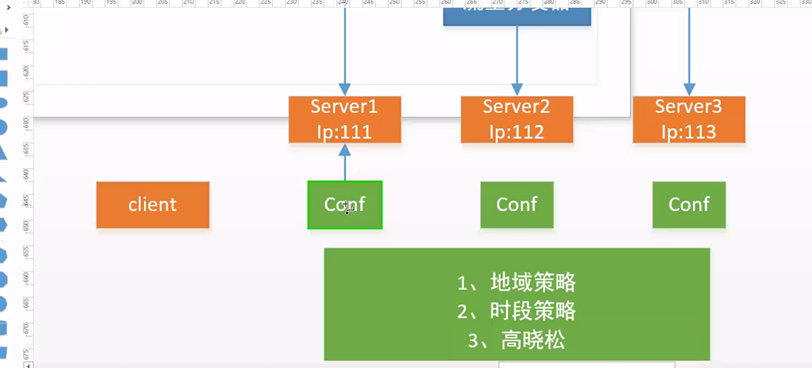
就是说比如说你在那个公司内部你是一个运维工程师，那这个运维工程师就是经常去监控一些个机器是不是？假设说这个运维工程师和开发工程师都总会遇到有这样的一个需求，这什么需求呢？比如说你开发这一套系统Server1和Server2和Server3，这一套系统可以对外提供服务了，那这套系统呢是里面很多复杂的策略对不对？比如说举个什么例子呢？比如说有一些物品，那这些物品呢，比如你是北京来的用户，我是可以看到这样一个信息的，比如说上海来的用户流量是不可以看到的，比如武汉来了一个，这个数据是可以看到的明白这意思把？相当于Conf这里面有一个什么策略呢？类似于是一个地域策略和时段策略，比如说有一些视频你白天的时候不能放，只能晚上的时候才能放，那这个时候哪些视频可以晚上放呢？这个时候就可以通过一个时间可以控制明白了把？或者有些视频你可以在上海或者是在偏远的山区你是可以看到的，但是在这个北上广这几个城市是看不到的就是类似于这样的。

就是你总会有些这么一个需求，不管是合理的也好不合理的也好总会有这样的需求。

或者我们拿高晓松的例子来举个例子。

比如说你传播这个高晓松酒驾的这些新闻都不允许上线，那么类似于这么多这么多的策略，比如说你真正做这个系统开发做项目产品的时候，像类似于这种的需求真的是特别特别多，那这么多需求通常来说通过怎样的信息来维护呢？那么通过一些配置（Conf），这个配置是通常是以文件的形式来维护的，这个文件是通过被服务器进行加载，然后服务器内部对这些需求策略进行挨个逐条去校验去过滤，然后比如说来了一个视频或者音乐，是不是应该把这个物品能够放出去呢，那就需要层层过滤是不是？（如下图15）

图15：



并且结合当前的流量的一些特征进行分发。

高晓松

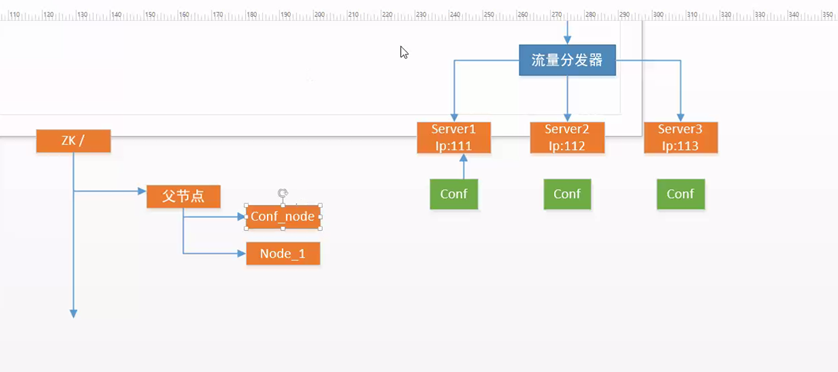
好了再举个例子，假设可能最近高晓松可能跟我们公司有合作，那我们一定对这个高晓松这个小说或者是棋坛都要进行强推荐是吧？但是呢总会把它一些负面新闻给它关联出来，那这样的话很不利于我们跟高晓松之间的一个合作把是不是？这是对整个合作是不利的，怎办呢？那我们需要对这些负面新闻信息进行一些屏蔽打压，那怎么打压呢？那我们可以通过一些临时改一些配置改一些数据，然后现在有一个黑名单，就是只要是你过滤到中间有一些不应该投放的一些信息，我就应该被这个黑名单给过滤掉明白这意思把？

然后这个时候说你一开始是没有这样的策略的，现在有这样的一个策略，这个老大非常着急，赶紧要把这个策略上线否则的话我们会面临什么什么样子的危险，那这个时候如果来了个程序员会怎样？你作为一个系统开发者，你就开始你上每台机器把这个配置（conf）去更新一遍是不是？你去ssh每台机器你觉得你会花比较少的时间，但是起码你人工操作一个配置怎样也要一分钟五分钟，起码是这种分钟级别的。

如果你三台机器也好，三分钟搞定，比如一分钟一个，比如说给你一百台机器，那你就花将近两个小时才能搞定是吧？如果发现你两个小时操作完一遍，你发现某个配置写错了怎办？那就重头来一遍，那这样是肯定不可能的，这时候你需要client客户端上写一个脚本去把这些配置(Conf)快速的过一遍是吧？那这个时候可以很快速的修改，如果你这个机器是有上百台上千台通过客户端上写一个脚本的方式还好，如果真要是机器再多一下的话你会发现这些机器都会访问你的这个客户端，这个客户端机器的压力就特别大了对不对？那这个时候这个客户端机器可能会挂掉，怎办呢？所以可以通过配置管理这样的方式来帮你这个客户端承担一些压力。

怎么承担呢？其实就很简单（如下图16所示）

图16：

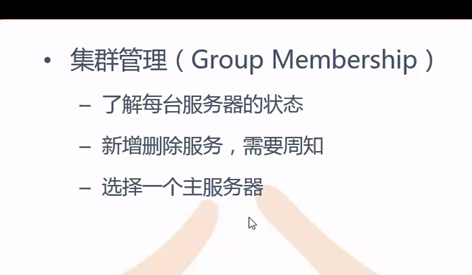


比如说图16中的Conf\_node就是一个配置节点，因为之前说了这个Conf\_node这个配置节点是可以存少量数据的，然后这个数据本身也是可以做一个配置，配置本身也是个数据对吧？配置一般来说都是一个比较短的一个文件，那如果要是再多的话，这个配置可能就不算是配置了。

那么这么搞就简单了，那所有的Server1和Server2和Server3这个机器都去监控Conf\_node这个节点明白了把？也就是说当你这个Conf\_node节点数据发生变化了就是你的配置，比如你想给大家重新做分发，然后这个配置发生变化，你这个配置数据一发送变化，好了，你所有的机器都会被通知到，一通知到说明你发现了一个数据变化这么一个事件了，那么接下来你就开始对Conf\_node这个数据进行一个传输或者是拷贝，然后变成本地配置文件然后加载都可以明白了把？那这种方式跟刚才的对比有什么好处呢？你想这个你其实对一个Zookeeper节点进行访问，相当于就是对一个集群进行访问，虽然Conf\_node这里面是一个节点，其实这个节点背后它是个集群明白这意思了把？它可以相当于把这个压力给你分发掉有这么一个特点这个大家能看懂了吗？

还有一类就是集群管理(Group Membership)(如下图17所示)

图17：



这个集群管理是什么意思呢？就是了解每台服务器的状态，可以新增删除服务，需要周知。

其实这两个我们刚才已经说过了对不对？你就是通过一个建立临时节点，然后让这个节点与这个服务器进行建立一个心跳关系然后去监控，如果是出现了一些服务的一些变化，那就是这些事件的变化你可以进行一些周知。还有一个就是选主。

我们先看这个集群状态，这个集群状态其实就是我们刚才说的，其实没什么区别，你创建一个临时节点，然后你对着干server的getChildren()这种方式进行一个遍历，如果某台服务器出现问题了出现下线了，那么对应的节点就自动删除（如下图18和19所示）

图18：

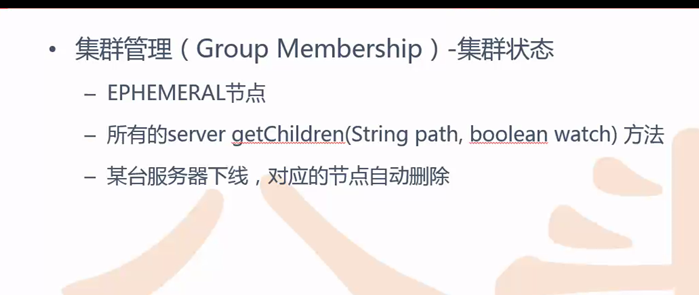
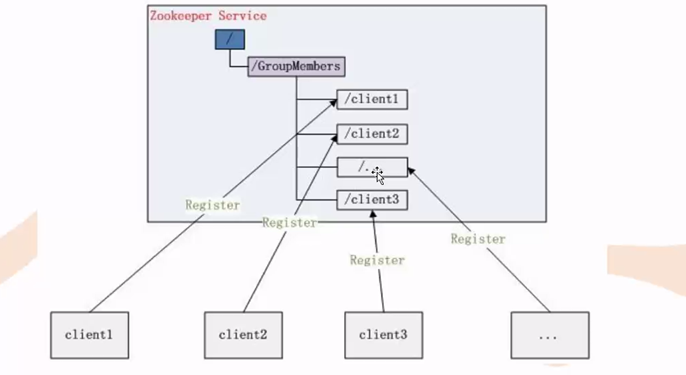


图19:



如图19所示，那个长方形里面三个点的机器出现下线了，那么它对应的节点就自动消失，然后通过它的一个父节点（GroupMombers）进行通知，这就不说了。

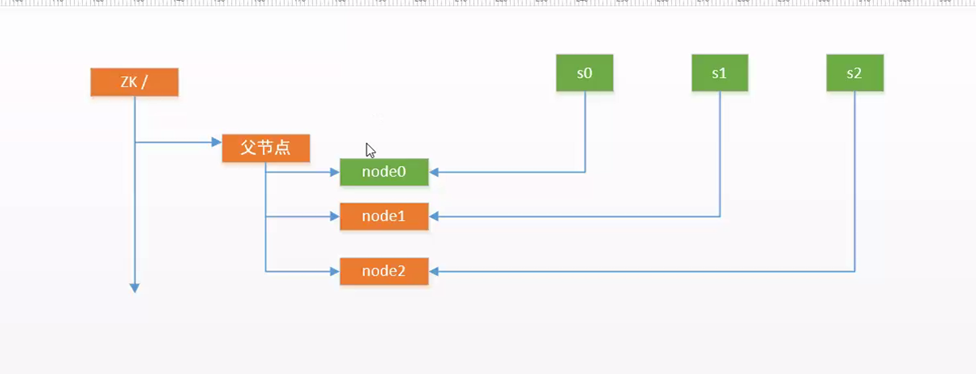
那么选主节点是什么意思呢？（如下图20所示）

图20：



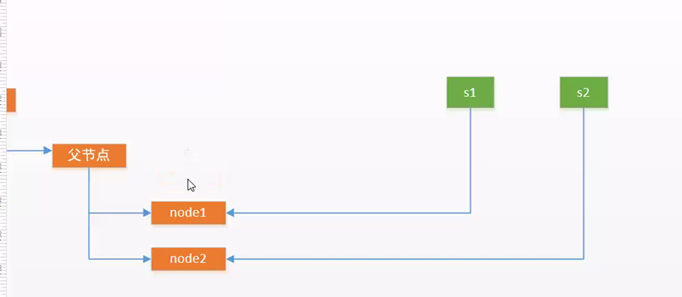
选主这个就需要通过你的临时节点（EPHEMERAL）和你的顺序节点(SEQUENTIAL)进行一个搭配，这个选主这个其实挺有意思的。（如下图21所示）

图21：



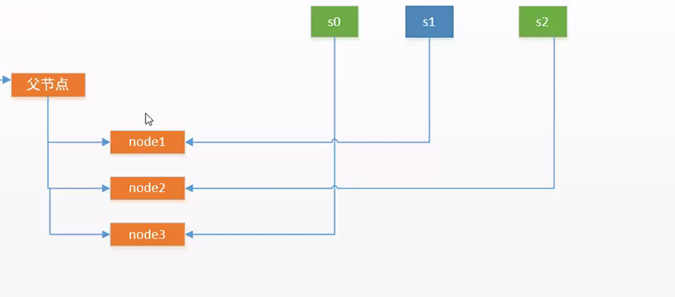
然后node0是有s0创建的，node1是有s1创建的，node2是有s2创建的，比如说这s1和s2和s3这三台机器都是主节点，比如说storm里面有多个主，那你在一个集群里面不可能存在多个主同时有两个主同时工作的这是不可能的，你这个当前只能有一个主来服务，那有这三台机器s1和s2和s3，那你选这三个主的哪个机器都可以，那具体选哪个呢？你必须要选出来一个是吧？那你要有一定的策略，一山不容二虎，那什么样的策略呢？很简单你可以通过这个图21中的数字编号最小的那个当主，那么s0最小是不是？那就是s0当主了对不对？这当主当着突然有一天出现问题了就挂掉了（如下图22所示）

图22：



那么同样s0的节点node0消失，这个时候父节点就开始告诉大家说你们之前的这个主挂掉了，现在就开始选举，选举出一个新的主出来，那么s1和s2选谁呢？一看就知道了是吧？S1比s2小，肯定选1嘛对吧？这个时候这个s1变成主了，那如果刚才那个机器s0这个机器修好了，如果这个s0还想来竞争这个主，你觉得这个s1会把这个主让给它吗？不会让给他，那这个s0也当作一个普通机器来，然后再你的父节点下面重新再创建一个节点（如下图23所示）

图23：

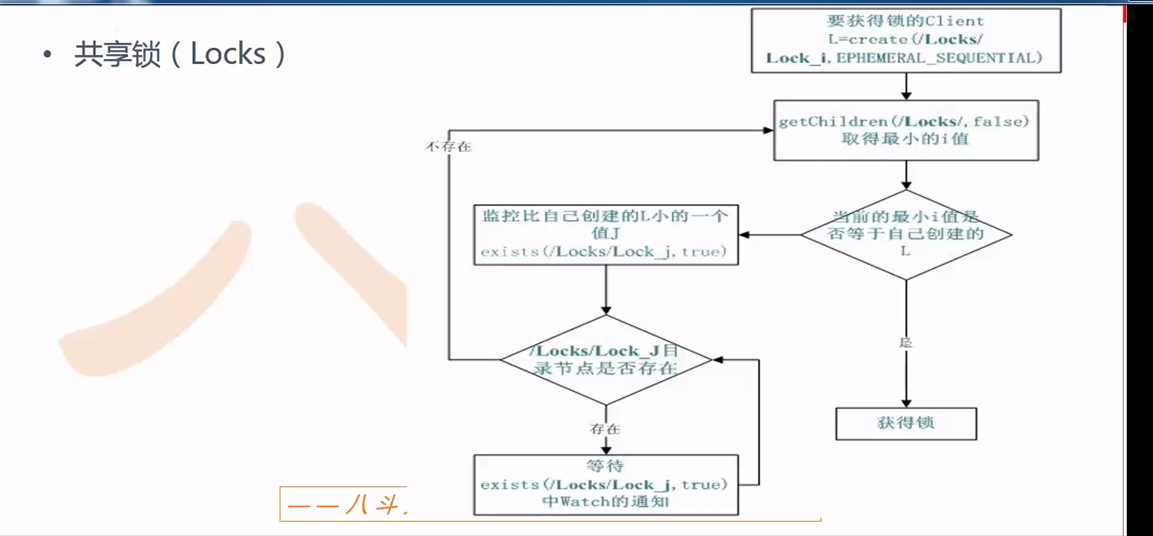


刚才那个node0就不能再分配给s0了，就是你后面的再往父节点下面再去创建节点的时候，那这个节点的后面的数字是累加的，为什么呢？是因为它的类型决定，类型是临时节点（EPHEMERAL）加顺序节点(SEQUENTIAL)明白了把？

所以选主的策略就是把最小编号那个Server挂掉之后，就开始选一个当前的情况下最小的那个节点。好了这个选主节点应该没什么问题把？

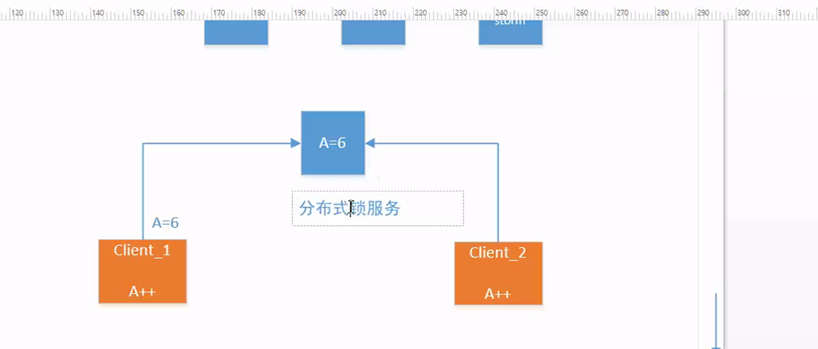
然后接下来我们看一个锁服务（如下图24所示）

图24：



这个锁服务是什么意思呢？这个锁服务是用来控制我们最一开始举得那个例子（如下图25所示）

图25：

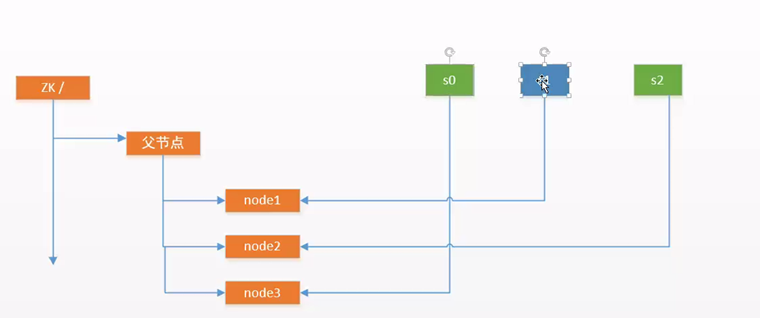


这不是锁服务嘛是不是？锁服务是来控制什么的呀？锁服务是来控制它不同节点不同角色之间的一个协同工作对吧？就是得有个顺序是吧？谁先来谁后来你必须把这个顺序理清楚是不是？那这个顺序，起码这个规则是由这个锁来控制的，就保证了你当前这A=6这里面的数据不能被乱改。

那锁呢是在单台机器会是常见的一种形式，那在分布式里面也可以通过一个锁的服务，只不过是这种锁就不可能像单机那种力度非常细，它是一种粗粒度锁，它就是不会细到内存级别。

好了这时候我们看一个这样的一个例子，就是说s0和s1和s2都想对某一个公共的数据进行修改(如下图26所示)

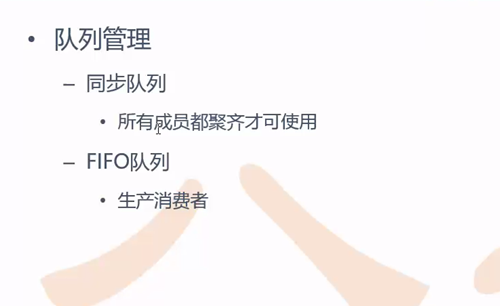
图26：



那谁先修改谁后修改，那这个时候其实跟刚才的例子一样，你就找node节点里面数字最小的那个就可以了，然后只有当这个s1处理完了之后，你就需要把它给退出掉，接下来就是node2所对应的那个进程开始进行运转是吧？那node2这个进程任务执行完之后，开始对这个node3所对应的那个进程开始运转明白这意思把？其实这个锁服务跟那个选主其实差不多。

后面还有一个队列管理（如下图27所示）

图27：

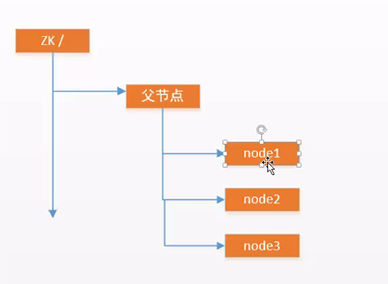


其实对这个队列管理也是比较好理解的。

分为同步队列和先入先出（FIFO）队列对不对？

什么叫同步队列呢？（如下图28所示）

图28：

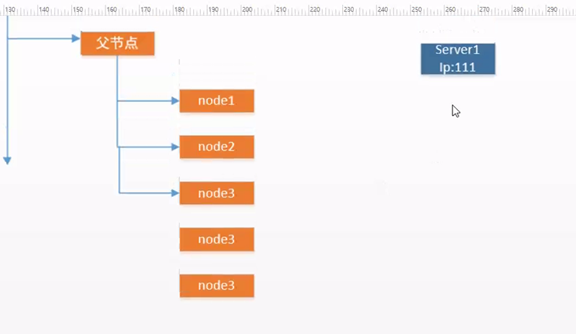


比如说一个班级要去植树，这个班级可能有50个人，比如说有50棵树，每一个人都去植树，然后谁要求的呢？是学校老师要求的，每人必须都去植树而且这个植树这个动作就是一个工作，必须要做的工作，然后要求是每人是只植自己的那颗树，不许让别人帮忙。

好了这个数据也有了，就等着大家同时到达，比如说约到下午一点我们开始植树，少一个我们都不出发，这个时候来了第49个人了，那还少一个人，那你必须等这个人，你这个人如果是不来的话我们就不出发，就接下来的工作得等你到，就相当于是个同步状态，必须把所有的资源都准备就绪了之后，那接下来的事情才能开始去做明白了把？那这个就叫做一个同步队列，就是说所有的成员都聚齐才可以用。

那这个怎么做到的呢？其实这个也比较容易。假设说我这个Server1我已经知道你父节点里面一共应该有几个节点（如下图29所示）

图29：



比如说有5个节点，我这个Server1已经知道了，如果你这5个节点都齐了之后，我这个Server1才去处理另外一个任务，其中有任何一个节点没有来，我下面的任务就无法进行。比方说每一个节点都是各自管理自己的一段数据是不是？比如你有一个节点不存在，那说明这个数据就不完整了，没人处理这个数据了对不对？所以必须要保证所有的节点都同时到齐。

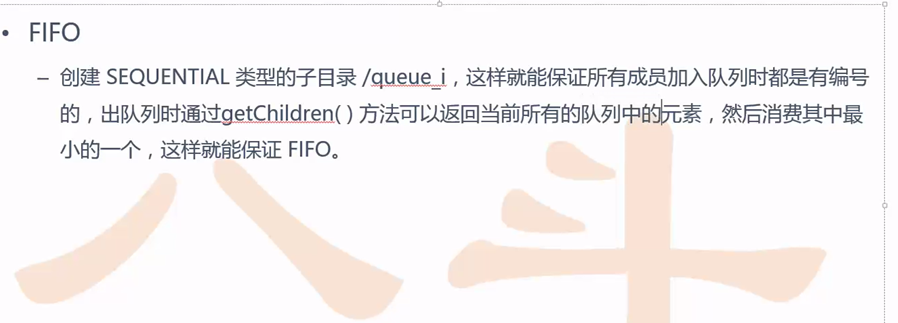
怎么保证同时到齐呢？就是通过这种getChildren()这个方式，父节点通知Server1

，就是说你这里面建了一个监控连接，就是Server1告诉你这个父节点说只要是你这个后面的这个队列一旦发生变化了你得及时的通知我，我来检查一下我来数一下人数够不够是吧？然后人数够的话，Server会对这个队列进行遍历发现确实人够了，接下来就开始去执行下一向工作了对吧？那其实这个就是一个同步队列明白这意思把？

那么先入先出队列是什么意思呢？其实这个队列跟我们刚才讲的一样，就是先来后到的意思，谁先来谁先处理，谁后来你必须排队，那么这个就不细说了哈！！

好了先入先出队列（FIFO）刚已经说过了（如下图30所示）

图30：



可以通过getChildren()去返回当前所有队列里面的元素，找到最小的那一个元素然后进行处理。

好了那么接下来呢，我们看一下Zookeeper具体怎么去用（如下图31所示）

图31：



这里就不说实践和安装部分了哈！！谢谢大家的支持！！