**1 概念**

分布式调度，指的是在分布式条件下，定时任务的执行和管理

定时任务是一个很常见的场景，比如：

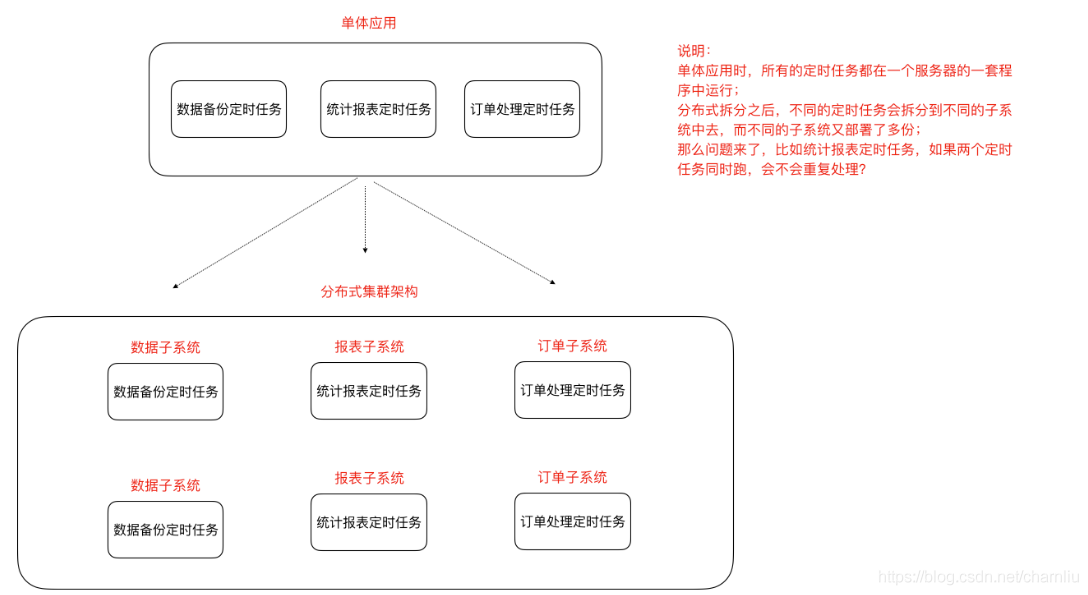
* 订单审核、出库
* 订单超时⾃动取消、⽀付退款
* 礼券同步、⽣成、发放作业
* 物流信息推送、抓取作业、退换货处理作业
* 数据积压监控、⽇志监控、服务可⽤性探测作业
* 定时备份数据
* ⾦融系统每天的定时结算 数据归档、清理作业 报表、离线数据分析作业

**2 分布式调度的特性。**

主要有两个特性

1. 运⾏在分布式集群环境下的调度任务，同⼀个定时任务程序可以部署多份，但是只应该有⼀个定时任务在执行
2. 可以把⼀个⼤的作业任务拆分为多个⼩的作 业任务，同时执⾏

特性一： 可以部署多份，只有一份执行

参见下图，部署多份只有一份执行是很有必要的，既可以节约不必要的资源浪费，也可以避免重复执行导致的潜在性错误。  


特性二： 拆分为小任务执行

当某个定时任务比较重的时候，如果能够利用分布式特性，将这个任务拆分出去，给多台机器执行，那么肯定是比较符合我们的实际需求的。

**3 分布式调度的实现**

如果说要让我来实现分布式调度，我可能第一时间想到消息队列。

既然说到消息队列，可能有个疑问就是定时任务与消息队列有什么区别和联系呢？

**3.1 定时任务与消息队列的联系**

**3.1.1 共同点**

1. 异步处理  
   ⽐如注册、下单事件 应⽤解耦，不管定时任务作业还是MQ都可以作为两个应⽤之间的⻮轮实现应⽤
2. 解耦  
   这个⻮轮可以中转 数据，当然单体服务不需要考虑这些，服务拆分的时候往往都会考虑
3. 流量削峰  
   双⼗⼀的时候，任务作业和MQ都可以⽤来扛流量，后端系统根据服务能⼒定时处理订单或者 从MQ抓取订单抓取到⼀个订单到来事件的话触发处理，对于前端⽤户来说看到的结果是已经 下单成功了，下单是不受任何影响的

**3.1.2区别**

* 定时任务作业是时间驱动，⽽MQ是事件驱动；
* 时间驱动是不可代替的，⽐如⾦融系统每⽇的利息结算，不是说利息来⼀条（利息到来事件）就算 ⼀下，⽽往往是通过定时任务批量计算； 所以，定时任务作业更倾向于批处理，MQ倾向于逐条处理；

**3.2 分布式调度框架Elastic-Job**

一般来说，我们不用自己实现一个分布式框架。如果要使用分布式调度，可以利用一些已经存在的框架。

回忆平时，使用定时任务的一个常见情况是linux的crontab，使⽤时间表达式（包括：秒、分、时、⽇、周、年） 可以在本机配置定时任务什么时间去执⾏。

Elastic-Job是当当⽹开源的⼀个分布式调度解决⽅案，也可以用时间表达式完成任务调度。

它是基于Quartz⼆次开发的，由两个相互独⽴的⼦项 ⽬Elastic-Job-Lite和Elastic-Job-Cloud组成。

* Elastic-Job-Lite 定位为轻量级⽆中⼼ 化解决⽅案，使⽤Jar包的形式提供分布式任务的协调服务
* Elastic-Job-Cloud 需要结合Mesos 以及Docker在云环境下使⽤。

Elastic-Job的github地址：https://github.com/elasticjob

**3.2.1 主要功能**

1. 分布式调度协调 在分布式环境中，任务能够按指定的调度策略执⾏，并且能够避免同⼀任务多实例重复执⾏
2. 丰富的调度策略 基于成熟的定时任务作业框架Quartz cron表达式执⾏定时任务
3. 弹性扩容缩容 当集群中增加某⼀个实例，它应当也能够被选举并执⾏任务；当集群减少⼀个实例 时，它所执⾏的任务能被转移到别的实例来执⾏。
4. 失效转移 某实例在任务执⾏失败后，会被转移到其他实例执⾏
5. 错过执⾏作业重触发 若因某种原因导致作业错过执⾏，⾃动记录错过执⾏的作业，并在上次作业 完成后⾃动触发。
6. ⽀持并⾏调度 ⽀持任务分⽚，任务分⽚是指将⼀个任务分为多个⼩任务项在多个实例同时执⾏。
7. 作业分⽚⼀致性 当任务被分⽚后，保证同⼀分⽚在分布式环境中仅⼀个执⾏实例

**3.2.2 安装**

Elastic-Job依赖于Zookeeper进⾏分布式协调，所以需要安装Zookeeper软件（3.4.6版本以上），关于 Zookeeper，此处我们不做详解，只需要明⽩Zookeeper的本质功能： 存储+通知。

安装Zookeeper

* 1 在linux平台解压下载的zookeeper-3.4.10.tar.gz
* 2 进⼊conf⽬录，cp zoo\_sample.cfg zoo.cfg
* 3 进⼊bin⽬录，启动zk服务 启动 ./zkServer.sh start 停⽌ ./zkServer.sh stop 查看状态 ./zkServer.sh status

工程中引入jar包

<!-- https://mvnrepository.com/artifact/com.dangdang/elastic-job-lite-core -->

<dependency>

<groupId>com.dangdang</groupId>

<artifactId>elastic-job-lite-core</artifactId>

<version>2.1.5</version>

</dependency>

* 1
* 2
* 3
* 4
* 5
* 6

继承 SimpleJob 类， 实现方法 ，然后是主类方法:

public static void main(String[] args) {

// 配置注册中⼼zookeeper，zookeeper协调调度，不能让任务重复执⾏，通过命名空间分类管理任务，对应到zookeeper的⽬录

ZookeeperConfiguration zookeeperConfiguration = new ZookeeperConfiguration("localhost:2181","data-archive-job");

CoordinatorRegistryCenter coordinatorRegistryCenter = new ZookeeperRegistryCenter(zookeeperConfiguration);

coordinatorRegistryCenter.init();

// 配置任务 JobCoreConfiguration jobCoreConfiguration =

JobCoreConfiguration.newBuilder("archive-job","\*/2 \* \* \* \* ?",1).build();

SimpleJobConfiguration simpleJobConfiguration = new SimpleJobConfiguration(jobCoreConfiguration,BackupJob.class.getName());

// 启动任务 new JobScheduler(coordinatorRegistryCenter,

LiteJobConfiguration.newBuilder(simpleJobConfiguration).build()).init();

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

**3.2.3 测试**

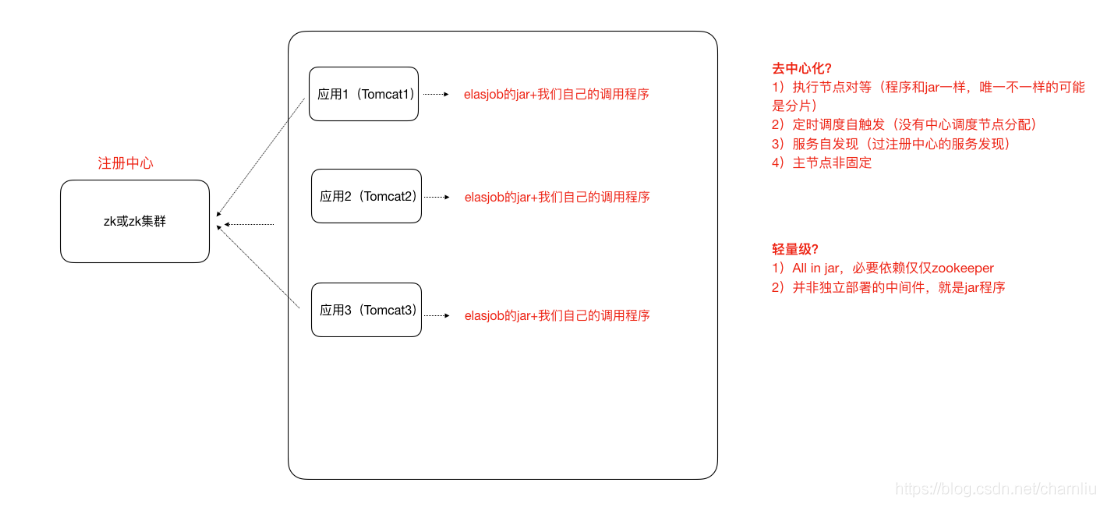
1. 可先启动⼀个进程，然后再启动⼀个进程（两个进程模拟分布式环境下，通⼀个定时任务 部署了两份在⼯作）
2. 两个进程逐个启动，观察现象
3. 关闭其中执⾏的进程，观察现象

**3.2.4 Leader节点选举机制**

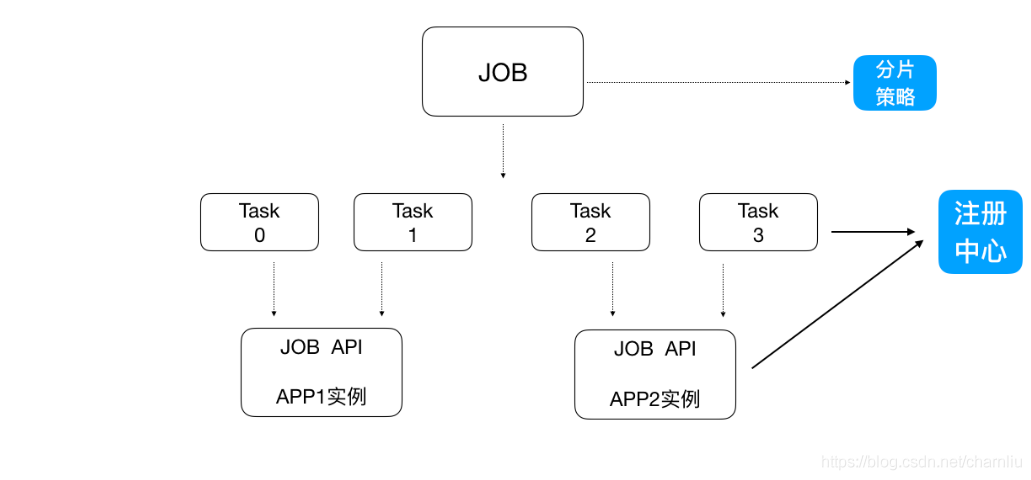
每个Elastic-Job的任务执⾏实例App作为Zookeeper的客户端来操作ZooKeeper的znode

1. 多个实例同时创建/leader节点
2. /leader节点只能创建⼀个，后创建的会失败，创建成功的实例会被选为leader节点， 执⾏任务

**3.2.5 Elastic-Job-Lite轻量级去中⼼化的特点**



**3.2.6 任务分片**

⼀个⼤的⾮常耗时的作业Job，⽐如：⼀次要处理⼀亿的数据，那这⼀亿的数据存储在数据库中，如果 ⽤⼀个作业节点处理⼀亿数据要很久，在互联⽹领域是不太能接受的，互联⽹领域更希望机器的增加去 横向扩展处理能⼒。所以，ElasticJob可以把作业分为多个的task（每⼀个task就是⼀个任务分⽚），每 ⼀个task交给具体的⼀个机器实例去处理（⼀个机器实例是可以处理多个task的），但是具体每个task 执⾏什么逻辑由我们⾃⼰来指定。参见下面的分片示意图：

Strategy策略定义这些分⽚项怎么去分配到各个机器上去，默认是平均去分，可以定制，⽐如某⼀个机 器负载 ⽐较⾼或者预配置⽐较⾼，那么就可以写策略。分⽚和作业本身是通过⼀个注册中⼼协调的，因 为在分布式环境下，状态数据肯定集中到⼀点，才可以在分布式中沟通。

**3.2.7 弹性扩容**

新增加⼀个运⾏实例app3，它会⾃动注册到注册中⼼，注册中⼼发现新的服务上线，注册中⼼会通知 ElasticJob 进⾏重新分⽚，那么总得分⽚项有多少，那么就可以搞多少个实例机器，⽐如完全可以分 1000⽚，那么就可以搞1000台机器⼀起执⾏作业

注意：

1. 分⽚项也是⼀个JOB配置，修改配置，重新分⽚，在下⼀次定时运⾏之前会重新调⽤分⽚算法，那么 这个分⽚算法的结果就是：哪台机器运⾏哪⼀个⼀⽚，这个结果存储到zk中的，主节点会把分⽚给分好 放到注册中⼼去，然后执⾏节点从注册中⼼获取信息(执⾏节点在定时任务开启的时候获取相应的分片
2. 如果所有的节点挂掉值剩下⼀个节点，所有分⽚都会指向剩下的⼀个节点，这也是ElasticJob的⾼可 ⽤。