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。

功能列表：

* 分布式调度协调
* 弹性扩容缩容
* 失效转移
* 错过执行作业重触发
* 作业分片一致性，保证同一分片在分布式环境中仅一个执行实例
* 自诊断并修复分布式不稳定造成的问题
* 支持并行调度
* 支持作业生命周期操作
* 丰富的作业类型
* Spring整合以及命名空间提供
* 运维平台

### 简介

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，外部依赖仅Zookeeper。

### 基本概念

#### 1、分片概念

任务的分布式执行，需要将一个任务拆分为多个独立的任务项，然后由分布式的服务器分别执行某一个或几个分片项。

例如：有一个遍历数据库某张表的作业，现有2台服务器。为了快速的执行作业，那么每台服务器应执行作业的50%。 为满足此需求，可将作业分成2片，每台服务器执行1片。作业遍历数据的逻辑应为：服务器A遍历ID以奇数结尾的数据；服务器B遍历ID以偶数结尾的数据。 如果分成10片，则作业遍历数据的逻辑应为：每片分到的分片项应为ID%10，而服务器A被分配到分片项0,1,2,3,4；服务器B被分配到分片项5,6,7,8,9，直接的结果就是服务器A遍历ID以0-4结尾的数据；服务器B遍历ID以5-9结尾的数据。

#### 2、分片项与业务处理解耦

Elastic-Job并不直接提供数据处理的功能，框架只会将分片项分配至各个运作的作业服务器，开发者需要自行处理分片项与真实数据的对应关系。

#### 3、个性化参数的适用场景

个性化参数即shardingItemParameter，可以和分片项匹配对应关系，用于将分片项的数字转换为更加可读的业务代码。

例如：按照地区水平拆分数据库，数据库A是北京的数据；数据库B是上海的数据；数据库C是广州的数据。 如果仅按照分片项配置，开发者需要了解0表示北京；1表示上海；2表示广州。 合理使用个性化参数可以让代码更可读，如果配置为0=北京,1=上海,2=广州，那么代码中直接使用北京，上海，广州的枚举值即可完成分片项和业务逻辑的对应关系。

### 核心理念

#### 1、分布式调度

Elastic-Job-Lite并无作业调度中心节点，而是基于部署作业框架的程序在到达相应时间点时各自触发调度。

注册中心仅用于作业注册和监控信息存储。而主作业节点仅用于处理分片和清理等功能。

#### 2、作业高可用

Elastic-Job-Lite提供最安全的方式执行作业。将分片总数设置为1，并使用多于1台的服务器执行作业，作业将会以1主n从的方式执行。

一旦执行作业的服务器崩溃，等待执行的服务器将会在下次作业启动时替补执行。开启失效转移功能效果更好，可以保证在本次作业执行时崩溃，备机立即启动替补执行。

#### 3、最大限度利用资源

Elastic-Job-Lite也提供最灵活的方式，最大限度的提高执行作业的吞吐量。将分片项设置为大于服务器的数量，最好是大于服务器倍数的数量，作业将会合理的利用分布式资源，动态的分配分片项。

例如：3台服务器，分成10片，则分片项分配结果为服务器A=0,1,2;服务器B=3,4,5;服务器C=6,7,8,9。 如果服务器C崩溃，则分片项分配结果为服务器A=0,1,2,3,4;服务器B=5,6,7,8,9。在不丢失分片项的情况下，最大限度的利用现有资源提高吞吐量。

### 整体架构图



## 一、起航

如果在使用中遇到什么问题，请首先在[FAQ](http://elasticjob.io/docs/elastic-job-lite/01-start/faq)中寻找解决问题的答案。

### 开发指南

#### 1、环境要求

* Java，JDK1.7及以上
* Zookeeper，3.4.6及以上
* Maven，3.0.4及以上