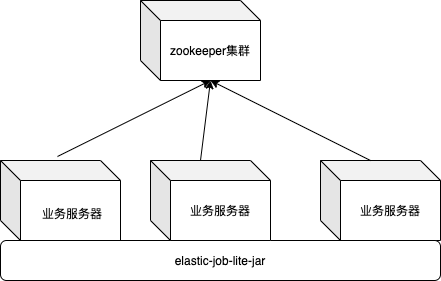
以elastic-job-lite为例，elastice-job-lite作为jar包集成到业务系统当中, 每个业务系统同时也是一个任务执行器，通过zk来协调。



如上图所示，elastic-job-lite通过zk实现各服务的注册、控制及协调：

* 第一台服务器上线触发主服务器选举。主服务器一旦下线，则重新触发选举，选举过程中阻塞，只有主服务器选举完成，才会执行其他任务。
* 某服务节点(引入elastic-job-lite-jar包)上线时会自动将服务器信息注册到注册中心，下线时会自动更新服务器状态。
* 主节点选举，服务器上下线，分片总数变更均更新重新分片标记。
* 定时任务触发时，如需重新分片，则通过主服务器分片，分片过程中阻塞，分片结束后才可执行任务。如分片过程中主服务器下线，则先选举主服务器，再分片。
* 通过上一项说明可知，为了维持作业运行时的稳定性，运行过程中只会标记分片状态，不会重新分片。分片仅可能发生在下次任务触发前。
* 每次分片都会按服务器IP排序，保证分片结果不会产生较大波动。
* 实现失效转移功能，在某台服务器执行完毕后主动抓取未分配的分片，并且在某台服务器下线后主动寻找可用的服务器执行任务。

## 调度问题的复杂性如何解决

* 弹性扩容  
  每个业务服务节点同时也是一个定时任务执行者, 各任务执行者之间不会直接通信，而是通过zookeeper协调, 定时任务状态存储在zk, 去中心化部署。  
  因此可以很方便地增加或者减少服务节点。
* 失效转移  
  如果在任务执行过程中有一个执行实例挂了，那么之前被分配到这个实例的任务（或者分片）会在下次任务执行之前被重新分配到其他正常节点实例上执行。
* 任务分片  
  elastic-job-lite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行中的服务节点，开发者需要自行处理分片项与真实数据的对应关系。