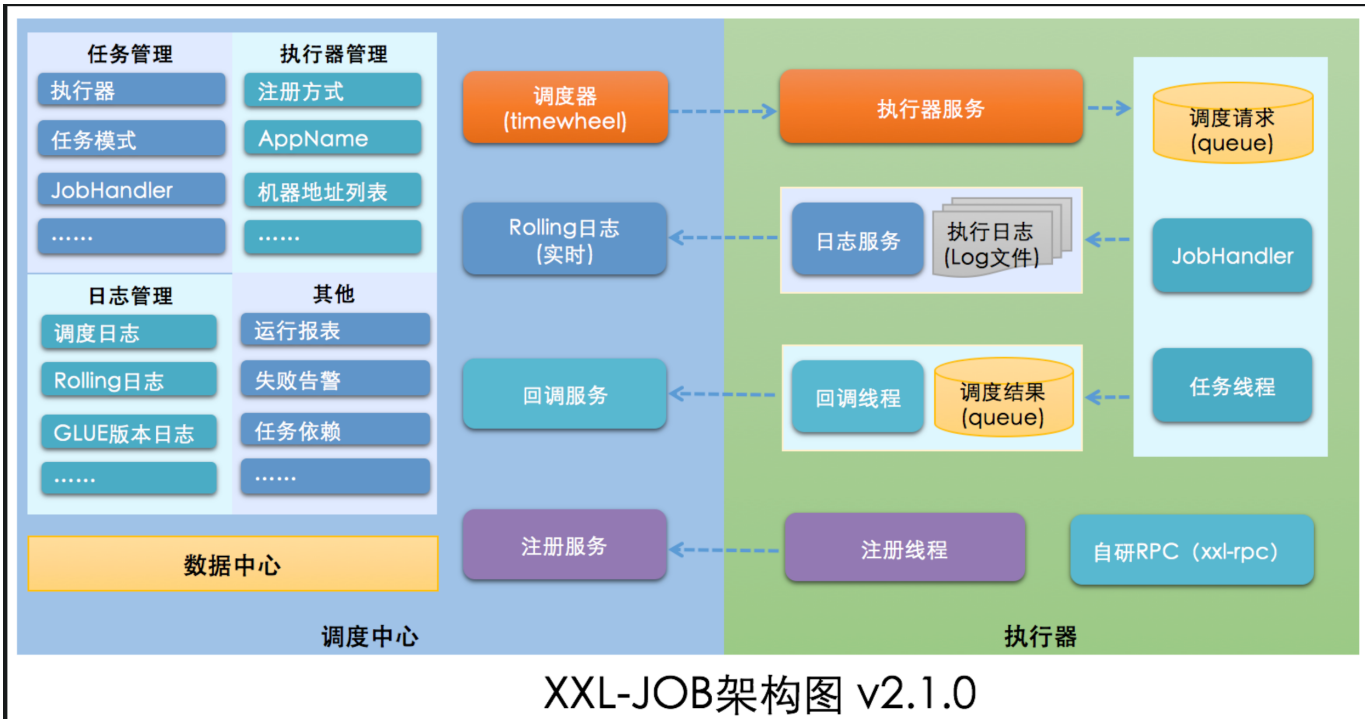
1.xxl-job

概念

xxl-job将调度行为抽象形成“调度中心”公共平台，而平台自身并不承担业务逻辑，“调度中心”负责发起调度请求。 将任务抽象成分散的JobHandler，交由“执行器”统一管理，“执行器”负责接收调度请求并执行对应的JobHandler中业务逻辑。

架构图

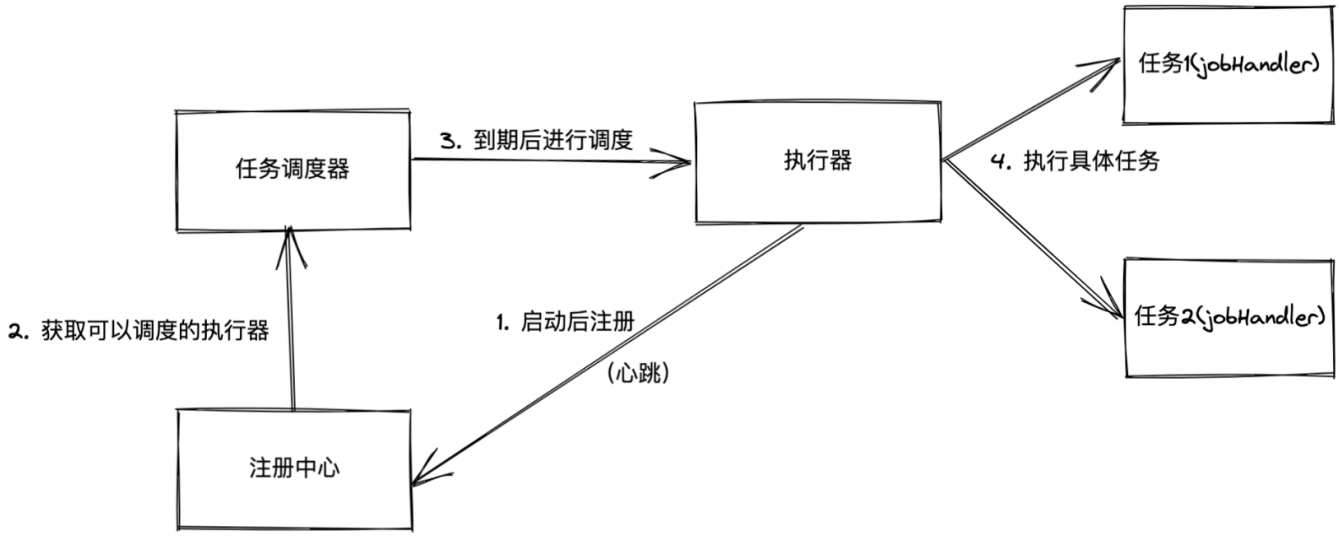


XXL-JOB架构图 v2.1.0

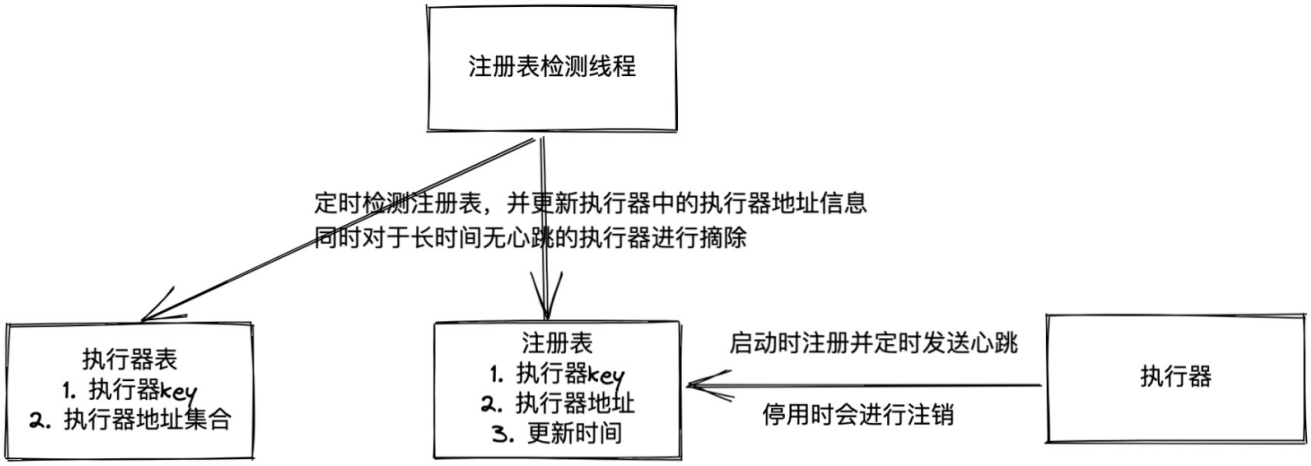
系统组成

- 调度模块（调度中心）： 负责管理调度信息，按照调度配置发出调度请求，自身不承担业务代码。调度系统与任务解耦，提高了系统可用性和稳定性，同时调度系统性能不再受限于任务模块； 支持可视化、简单且动态的管理调度信息，包括任务新建，更新，删除，GLUE开发和任务报警等，所有上述操作都会实时生效，同时支持监控调度结果以及执行日志，支持执行器Failover。
- 执行模块（执行器）： 负责接收调度请求并执行任务逻辑。任务模块专注于任务的执行等操作，开发和维护更加简单和高效； 接收“调度中心”的执行请求、终止请求和日志请求等。

调度流程



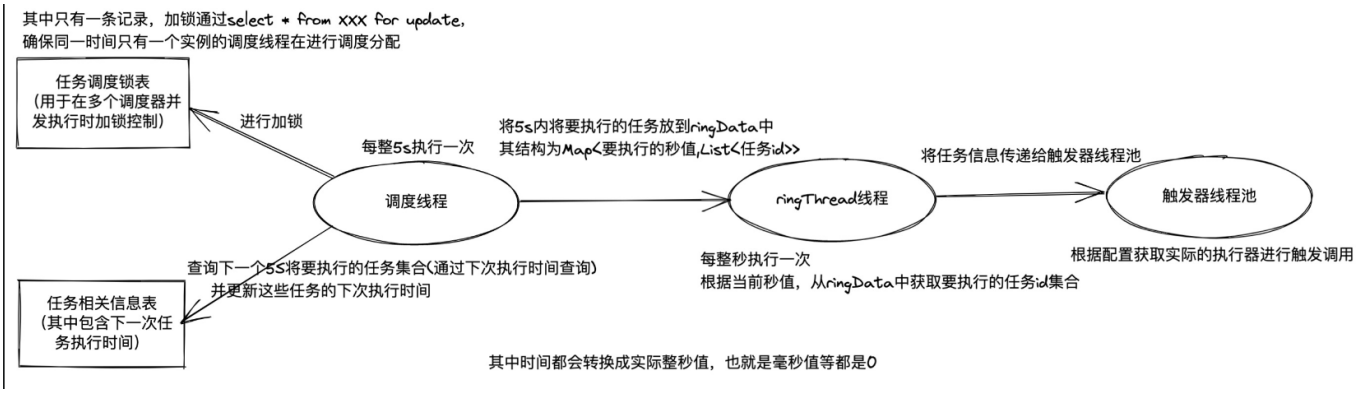
执行器流程



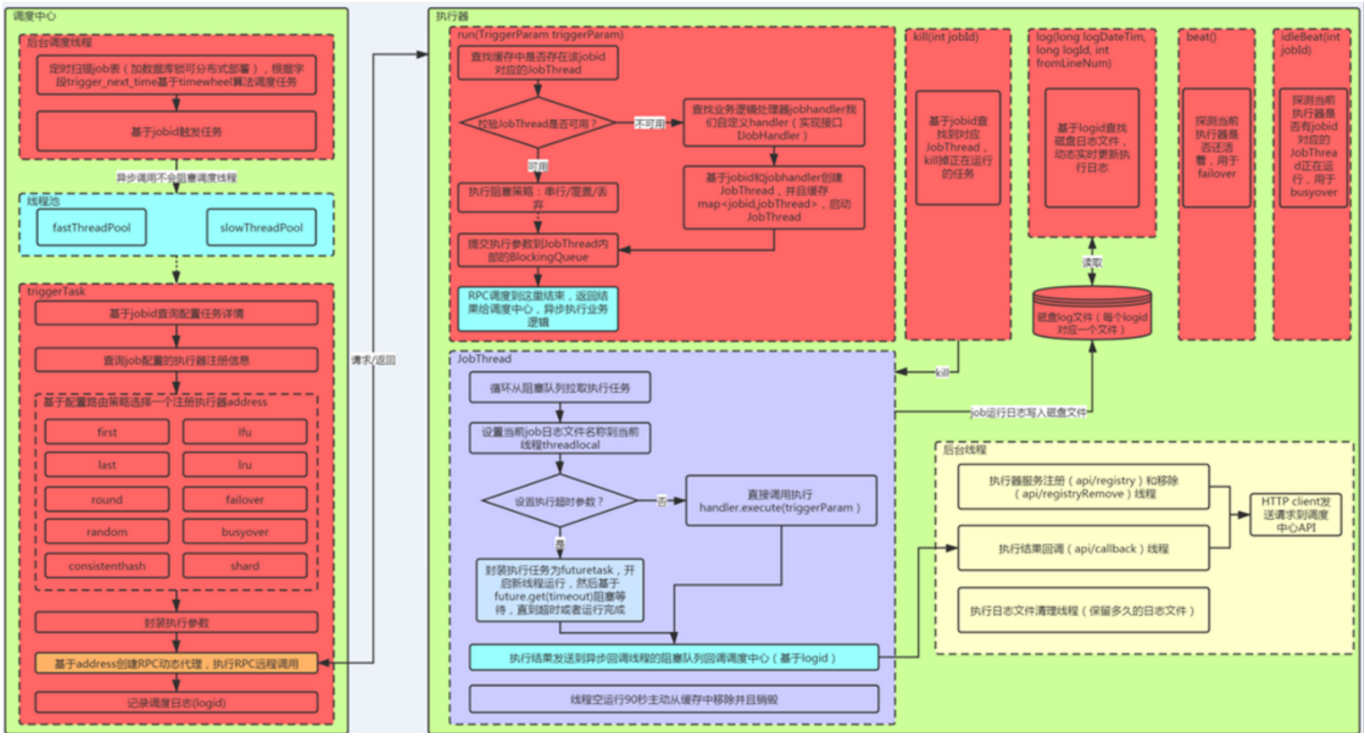
1.在执行器启动时，执行器会根据配置的调用中心地址，调用对应的接口了实现执行器的注册（以便后续调用器进行调度），注册后每隔30S还会进行一次心跳，证明自己还活着，在关闭时也会调用接口将自己摘除 2.xxl-job整体没有使用zookeeper之类的注册中心，在接收到执行器的注册请求后，会新增（更新）执行器的信息到对应的MySQL表（xxl_job_registry）中，同时在接收到心跳请求时更新其中的update_time字段 3.调用中心内部有一个检测线程，每隔30S检测一次，当发现有执行器3次都没有心跳（当前时间 - 最后一次更新时间 > 30S）时，认为这个执行器已下线，此时删除对应的记录（同时也会更新执行器记录中的执行器地址信息）

> xxl-job 任务调度中心会定时检查任务表的变化情况，这个时间间隔默认为 3 秒。如果任务表有新的任务添加或者任务状态发生变化，任务调度中心会根据任务的执行计划和优先级，更新任务队列中的任务执行状态，并将任务分发到相应的执行器上执行。这样，就保证了任务的及时性和准确性。

调度中心流程



调用流程图



任务模式

1.bean模式 原理：每个Bean模式任务都是一个Spring的Bean类实例，它被维护在“执行器”项目的Spring容器中。任务类需要加“@JobHandler(value=”名称”)”注解，因为“执行器”会根据该注解识别Spring容器中的任务。任务类需要继承统一接口“IJobHandler”，任务逻辑在execute方法中开发，因为“执行器”在接收到调度中心的调度请求时，将会调用“IJobHandler”的execute方法，执行任务逻辑。

注册发现

1.通过db来实现轻量化的注册发现 没有采用Zookeeper作为注册中心，采用DB方式进行任务注册发现； 2.注册表: 见"xxl_job_registry"表, "执行器" 在进行任务注册时将会周期性维护一条注册记录，即机器地址和AppName的绑定关系; "调度中心" 从而可以动态感知每个AppName在线的机器列表; 3.执行器注册: 任务注册Beat周期默认30s; 执行器以一倍Beat进行执行器注册, 调度中心以一倍Beat进行动态任务发现; 注册信息的失效时间为三倍Beat; 4.执行器注册摘除：执行器销毁时，将会主动上报调度中心并摘除对应的执行器机器信息，提高心跳注册的实时性；

分片

1.集群模式下 执行器集群部署时，任务路由策略选择"分片广播"情况下，一次任务调度将会广播触发对应集群中所有执行器执行一次任务，同时系统自动传递分片参数；可根据分片参数开发分片任务；

"分片广播" 以执行器为维度进行分片，支持动态扩容执行器集群从而动态增加分片数量，协同进行业务处理；在进行大数据量业务操作时可显著提升任务处理能力和速度。

> 1、分片任务场景：10个执行器的集群来处理10w条数据，每台机器只需要处理1w条数据，耗时降低10倍； 2、广播任务场景：广播执行器机器运行shell脚本、广播集群节点进行缓存更新等

故障转移 & 失败重试

1."故障转移"发生在调度阶段，在执行器集群部署时，如果某一台执行器发生故障，该策略支持自动进行Failover切换到一台正常的执行器机器并且完成调度请求流程。 2."失败重试"发生在"调度 + 执行"两个阶段，支持通过自定义任务失败重试次数，当任务失败时将会按照预设的失败重试次数主动进行重试；

properties配置模板

```
# xxl-job配置：执行器任务状态日志持久化方式：0=Log4j；1=Logback；2=Slf4j；
3=JdkLog；4=Console；5=File；6=无日志记录
xxl.job.executor.log-style=2
# xxl-job配置：执行器注册地址，多个注册中心逗号分隔，如："http://address" 或
"http://address01,http://address02"
xxl.job.executor.registry-list=http://127.0.0.1:8848/registry
# xxl-job配置：执行器获取任务地址，多个执行器逗号分隔，如："http://address" 或
"http://address01,http://address02"
xxl.job.executor.address=http://127.0.0.1:9999/xxl-job-executor-sample
# xxl-job配置：执行器AppName，服务端注册的AppName保持一致
xxl.job.executor.appname=xxl-job-executor-sample
# xxl-job配置：执行器IP地址
xxl.job.executor.ip=
# xxl-job配置：执行器端口号
xxl.job.executor.port=9999
# xxl-job配置：执行器日志缓存保存天数，超过限制自动清除。限制：大于等于1；等于-1则表示
不清除缓存（将缓存保存在内存中，重启即失效）
xxl.job.executor.logretentiondays=-1
# xxl-job配置：执行器扫描间隔秒数，配置为小于等于0 则不启动扫描，默认值:3
xxl.job.executor.scan-interval=3
# xxl-job配置：执行器日志文件目录，默认在跟目录下 logs/xxl-job/jobhandler(lower
case) 目录下
xxl.job.executor.logpath=./logs/xxl-job/jobhandler
# xxl-job配置：执行器日志文件保存天数，默认保存30天
xxl.job.executor.logretentiondays=30
# xxl-job配置：执行器心跳间隔秒数，默认值:30
xxl.job.executor.heart-beat-rate=30
# xxl-job配置：执行器异步任务线程池大小，用于执行异步任务，如：任务超时，任务重试 负载
等。默认为2，线程池的最大线程数可通过 executor.async.worker.thread.max 参数进行调整
xxl.job.executor.async.worker.thread.size=2
```

开源地址：https://github.com/xuxueli/xxl-job/

https://www.cnblogs.com/zhou-yuan/p/14750218.html