**基于akka技术栈的任务调度引擎功能与实现技术介绍**

目 录

[1 大数据平台 3](#_Toc43129087)

[1.1 Optimus Manager 3](#_Toc43129088)

[1.2 NTP 3](#_Toc43129089)

[1.3 HDFS 3](#_Toc43129090)

[1.4 Yarn 3](#_Toc43129091)

[1.5 Spark2 3](#_Toc43129092)

[1.6 Zookeeper 3](#_Toc43129093)

[1.7 FulltextSearch 4](#_Toc43129094)

[1.8 Hue 4](#_Toc43129095)

[2 调度系统 4](#_Toc43129096)

[2.1 Optimus Scheduler 4](#_Toc43129097)

# 简介

基于akka的任务调度引擎是一个响应式，完全去中心化，可动态缩扩容，支持跨集群任务调度，指定节点执行任务等功能的调度引擎，目前正在开发中。已可以实现工作流的正常执行、停止、重试；工作流定时调度，暂停，继续，停止；状态查看等功能。

# 功能整理

## 去中心化，可动态缩扩容

### 实现方案

任务调度引擎底层基于Akka Cluster搭建，该组件有一整套比较完善的集群实现方案，可以保证去中心化，动态缩扩容，保证集群的高可用性

### 实现情况

已完成

## 跨集群任务调度，指定节点执行任务

### 实现方案

任务调度引擎支持Akka的发布订阅系统，每一个节点都会订阅一个或多个topic。当提交任务时，任务会根据自己的topic来随机找到一个拥有相同topic的节点执行任务。用户可以根据业务需求来配置需要的topic，比如：

* 对于一个拥有多个集群节点的调度引擎，各个集群的节点都订阅一个各自的topic，这样就可以实现跨集群的任务调度。
* 每个节点都订阅一个独立的topic，这样就可以已节点为粒度来执行任务，不过在这种情况下若某节点故障可能会导致工作流执行失败。

### 实现情况

已完成

## 定时触发工作流执行

工作流需要周期性的执行，周期性执行的策略可能是每隔一段时间执行一次，也可能是监听某个文件目录出现变化就执行一次等等。

### 实现方案

调度引擎提供com.oceanum.trigger.Trigger接口来供开发者实现不同的触发策略，接口包含启动、停止、暂停、继续四个方法。

### 实现情况

已实现基于cron表达式的Quartz触发器，可以正常运行。

其他触发器正在开发，比如HDFS目录监听触发器等。

## 工作流构建与执行

### 实现方案

* **构建：**

基于Akka Stream Graph构建，工作流DAG图包含了以下几种算子：

|  |  |  |  |
| --- | --- | --- | --- |
| 算子名称 | 输入接口 | 输出接口 | 功能说明 |
| start | 0 | 1 | 启动一个动作流 |
| end | 1 | 0 | 结束一个工作流 |
| task | 1 | 1 | 执行一个任务 |
| fork | 1 | n | 并行的通知下游算子执行 |
| decision | 1 | n | 根据算子的条件表达式选择一个下游算子执行 |
| join | n | 1 | 收集上游的算子执行情况，当所有上游算子都执行完后，通知下游算子执行。用在fork算子的下游 |
| converge | n | 1 | 当上游的其中一个算子执行完后，就会通知下游的算子执行。  用在decision算子的下游。 |

通过以上的算子即可构建可运行的工作流。

* **运行：**

当工作流构建好后，需要执行工作流。只需要在工作流的输入端提交一个GraphMeta对象，GraphMeta对象内包含了工作流的运行参数和运行时的状态信息。当Task算子获取GraphMeta对象后会生成一个任务，提交给集群的某个节点执行，并监听任务的执行状态。当任务完成后会返回一个TaskMeta对象。GraphMeta和TaskMeta结合生成一个新的GraphMeta并通知到下游，

下游的Task算子会继续这个过程。

### 实现情况

已完成

## 任务执行和优先级管理

### 实现方案

* **任务执行：**

当需要执行一个任务时，任务执行客户端会根据任务的topic查找是否有带该topic的节点，如果有则会创建一个ClientInstance，节点也会创建一个ExecutionInstance。

* + ClientInstance负责监听状态信息和提交kill命令；
  + ExecutionInstance负责向ClientInstance推送任务的状态信息并向**RunnerManager(执行管理器)**提交任务，RunnerManager会返回一个hook用来kill掉任务。
  + RunnerManager(执行管理器)实际上是一个拥有固定线程的线程池，线程会根据任务配置封装成一个TaskRunner(执行器)对象来执行任务，TaskRunner每次执行一个任务都会通过java的Process类创建一个子进程，在这个子进程中执行任务。这样能够保证资源隔离，并且可以使用不同的系统用户或kerberos租户来执行任务，java任务也能使用jvm参数。
* 优先级管理：

每个任务都带有一个优先级值，当任务提交进RunnerManager时首先会进入一个优先级队列会根据优先级排序，TaskRunner会从队列中拉取任务并执行，从而实现按优先级运行。

### 实现情况

完成

## 任务控制台输出管理

需要查看任务的执行日志。

### 实现方案

调度引擎提供com.oceanum.exec.StdHandlerFactory接口来供开发者实现不同的任务控制台输出管理方案，接口方法包括对stdout和stderr输出的存储和查看。

### 实现情况

实现基于HDFS和LocalFile的控制台输出存储，查看输出还未实现。

## 任务算子类型

需要支持多种算子类型，以及自定义段子

### 实现方案

所有的任务算子类型都是shell算子的包装，实际上都是TaskRunner创建的子进程中执行的一个shell命令。TaskRunner有两个子类，分别是：

* ProcessRunner:用来执行shell，java，scala，python任务，子进程正在运行表示任务正在运行，进程结束后使用exitcode来判断任务成功还是失败；
* PluggableProcessRunner:部分任务会携带有外部系统的状态信息，比如Spark，Mapreduce的Yarn应用信息，这些任务使用该执行器执行。任务的运行状态和基本信息使用Akka Remote传输。任务需要实现com.oceanum.pluggable.Executor接口。该接口也用来实现自定义的任务。

### 实现情况

已实现ProcessRunner和shell，java，scala，python算子，

PluggableProcessRunner还在开发中。

## 持久化和查询工具

下面的几类对象需要进行持久化，并且需要方便查询：

* 定时任务和工作流的配置；
* 工作流的运行记录；
* 算子任务的运行记录；

### 实现方案

调度引擎提供com.oceanum.persistence.RepositoryFactory接口来供开发者实现不同数据库下的持久化方案，推荐使用json格式存储的数据库。

### 实现情况

已实现基于elasticsearch的EsRepositoryFactory。

其他正在开发，比如Mongodb。

## 文件管理工具

任务算子的执行文件需要存放在一个方便管理、上传、下载的地方

### 实现方案

调度引擎提供com.oceanum.file.FileSystem接口来供开发者实现不同的文件管理工具。

### 实现情况

已实现基于hdfs的HadoopFileSystem。

其他正在开发，比如s3。