阿里云
专有云企业版
企业级分布式应用服务 EDAS

SchedulerX技术白皮书

产品版本：V3.12.0

文档版本：20200622

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告：** 重启操作将导致业务中断，恢复业务时间约十分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 **注意：** 权重设置为0，该服务器不会再接受新请求。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明：** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	单击**设置** > **网络** > **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令。	执行cd /d C:/window命令，进入Windows系统文件夹。
斜体	表示参数、变量。	bae log list --instanceid Instance_ID
[]或者[a|b]	表示可选项，至多选择一个。	ipconfig [-all|-t]
{}或者{a|b}	表示必选项，至多选择一个。	switch {active|stand}

目录

1 什么是分布式任务调度 SchedulerX 2.0

分布式任务调度 SchedulerX 2.0 是阿里巴巴基于 Akka 架构自研的的新一代分布式任务调度平台，提供定时、工作流任务编排、分布式批量调度等功能，具有高可靠、海量任务、秒级调度能力。

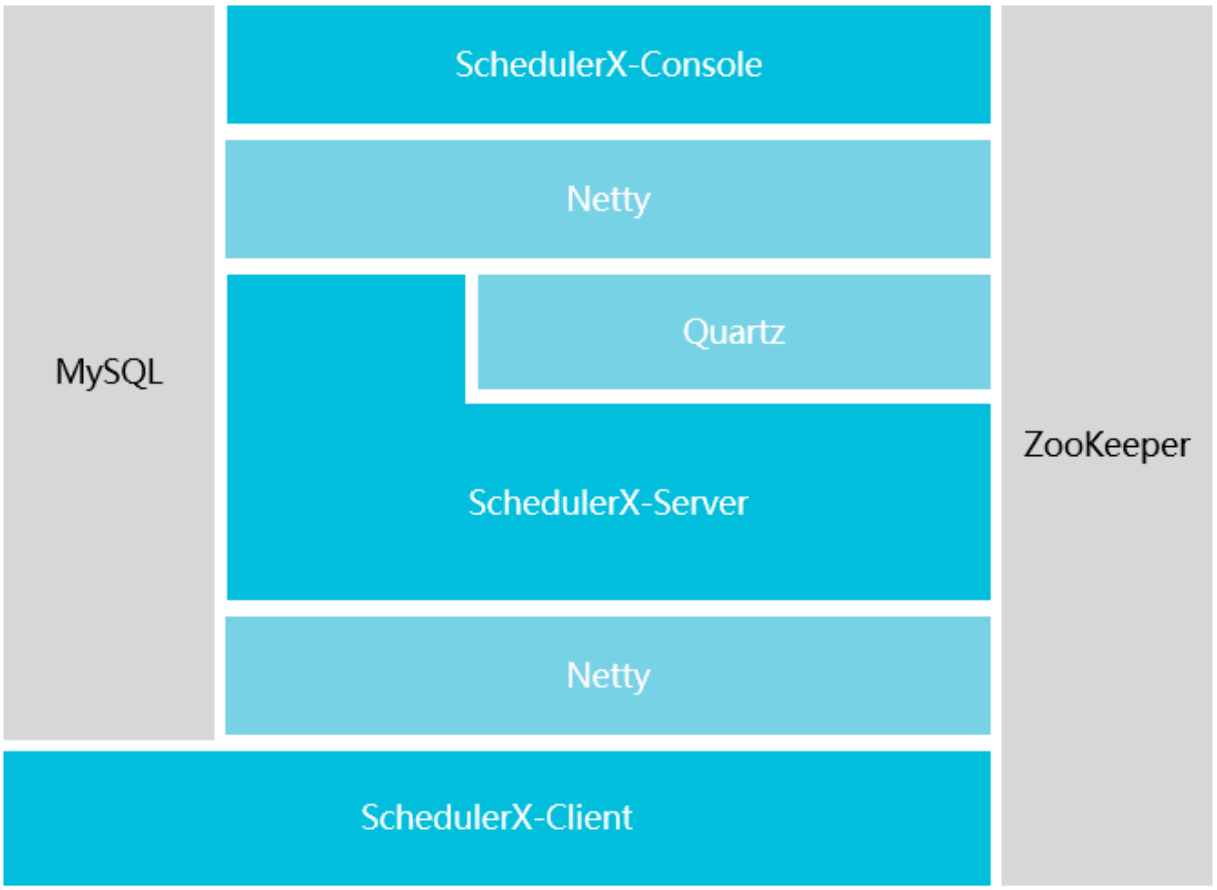
您可以在控制台配置、管理您的定时调度任务、查询任务执行记录和运行日志，还可以通过工作流进行任务编排和数据传递。SchedulerX 2.0 提供了简单易用的分布式编程模型，简单几行代码就可以将海量数据分发到多台机器上执行。

2 产品架构

本文档介绍SchedulerX的产品架构，以及各模块的作用。

SchedulerX的架构如图：

图 2-1: SchedulerX产品架构



SchedulerX各模块的作用如下：

- **SchedulerX-Server**：SchedulerX 的调度中枢，负责 Job 的触发、状态统计等。
- **ScheudlerX-Console**：SchedulerX 的 Web 控制台，是人机交互的主要入口。用户登录 SchedulerX-Console，可以进行 Job 的日常管控，包括 Job 维护，执行状态查看等一系列操作。同时，客户端通过 SchedulerX-Console，可以发现 SchedulerX-Server 的 IP 列表，客户端基于此找到需要连接的 Server 并建立连接。
- **SchedulerX-Client**：作为客户端嵌入到用户自己的应用中，是 Job 真正运行的地方。Server 发起触发命令，Client 收到命令后，执行用户自定义的业务逻辑，并上报执行结果给 Server。
- **ZooKeeper**：作为 Server 列表和 Console 列表的注册中心，同时作为多 Server Job 触发的协调中心。

- **MySQL**：作为 Job 相关元数据配置的存储中心，同时保存着中间运行时状态，包括 Job 触发实例、子任务等数据。
- **Quartz**：一个完全由Java编写的开源调度框架。
- **Netty**：Netty 是各模块之间通信的网络底层框架。

3 功能特性

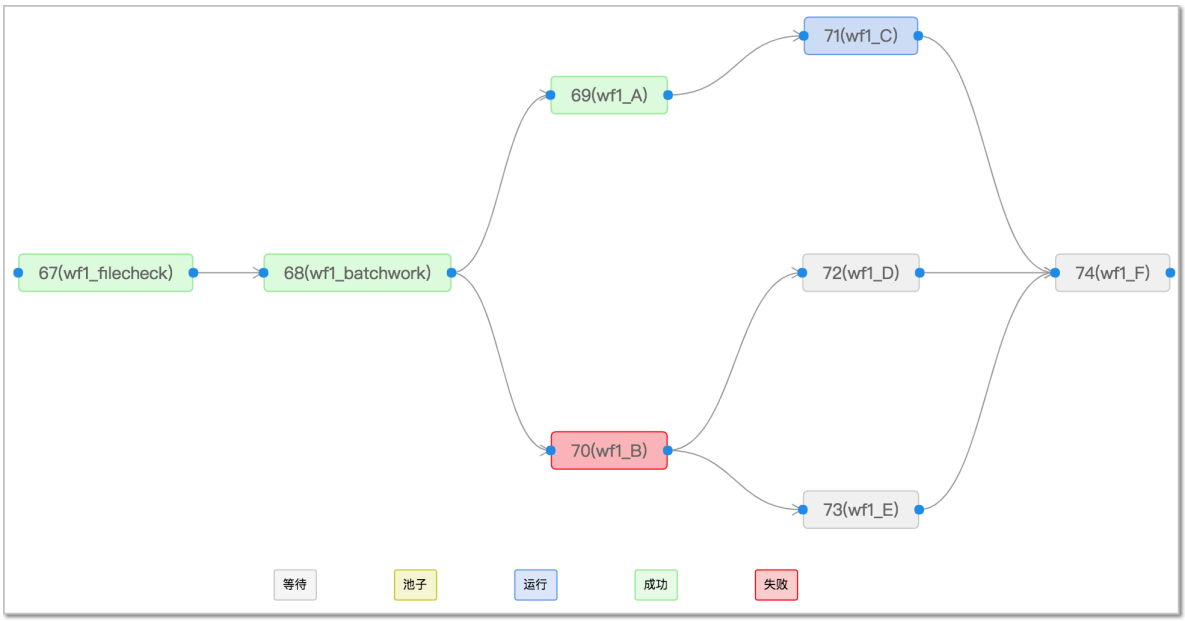
SchedulerX 2.0 主要提供调度、执行和运维三方面的功能。

调度

SchedulerX 2.0 支持定时（多种定时表达式）和工作流调度。

- 多种表达式的定时调度
  - Crontab：支持 Unix Crontab 表达式，详情请参见 Quartz Cron表达式。不支持秒级别。
  - Fixed rate：Crontab 必须被 60 整除，不支持其它数量级时间间隔的任务，如每隔 40 分钟的定时任务。Fixed rate 专门用来做定期轮询，可以弥补 Crontab 的不足，且表达式简单。不支持秒级别。
  - Second delay：适用于对实时性要求比较高的业务，例如执行间隔为 10 秒的定时调度任务。支持秒级别。
  - 日历：支持多种日历，还可以自定义导入日历。适用于金融业务，如需要在每个交易日执行定时任务。
  - 时区：适用于跨国业务，如需要在每个国家所在时区执行定时任务。
- 工作流调度

  使用有向无环图 DAG（Directed Acyclic Graph）进行任务编排，操作简单，前端直接拖拖拽拽即可。详细的任务状态图能直观的看到下游任务为什么没执行。



在定时调度和工作流调度中支持基于多语言的多种任务类型。

- Java：可以在用户进程中执行，也可以通过上传 JAR 包动态加载。

- Shell：前端直接写 Shell 脚本。
- Python：前端直接写 Python 脚本，需要 Python 环境。
- Go：前端直接写 Go 脚本，需要 Go 环境。
- HTTP：HTTP 任务无需依赖 Client，在控制台配置完即可使用。
- 自定义：可以自定义任务类型，然后实现一个 Plugin 即可。

执行

SchedulerX 2.0 支持多种执行方式和主流的分布式编程模型。

- 多种执行方式
  - 单机：随机挑选一台机器执行。
  - 广播：所有机器同时执行且等待全部结束。
  - 并行计算：map/mapreduce 模型，1~300 个子任务，有子任务列表。
  - 内存网格：map/mapreduce 模型，100,000 以下子任务，无子任务列表，基于内存计算，比网格计算快。
  - 网格计算：map/mapreduce 模型，1000,000 以下子任务，无子任务列表，基于文件 H2 计算。
  - 分片运行：类似 elastic-job 模型，控制台配置分片参数，可以将分片平均分给多个客户端执行。支持多语言版本。
- 分布式编程模型
  - Map 模型：类似于 Hadoop MapReduce 里的 Map。只要实现一个 Map 方法，简单几行代码就可以将海量数据分发到多台机器上执行。
  - MapReduce 模型：MapReduce 模型是 Map 模型的扩展，废弃了 postProcess 方法，新增 Reduce 接口，需要实现 MapReduceJobProcessor。

运维

SchedulerX 2.0 支持数据大盘、报警监控、日志搜集、失败重试，数据时间和重刷数据等运维能力。

- 数据大盘

  控制台提供了执行记录大盘和执行列表，可以看到每个任务的执行历史，并提供操作。

- 查看日志

  每次执行的调度任务都可以在详情中查看运行日志。如果任务执行失败，前端直接就能看到错误日志，非常方便。

- 原地重跑

  任务失败，修改完代码发布后，可以立即重新执行。

- 标记成功

  任务失败，如果后台把数据处理修正了，重新执行又需要几个小时，可以直接将任务标记为成功。

- 停止调度任务

  实现 JobProcessor 的 kill() 接口，您就可以在前端停止正在运行的任务，甚至子任务。

- 数据时间

  SchedulerX 2.0 可以处理有数据状态的任务，在创建任务的时候设置调度时间，而实际上处理的数据时间可能和任务执行时间不一致，可以配置时间偏移，调度时间 + 时间偏移即数据时间。例如一个任务是每天 00:30 运行，但是实际上要处理前一天的数据，就可以向前偏移一个小时。调度时间不变，执行的时候通过 context.getDataTime() 获得的就是前一天 23:30。

- 重刷数据

  既然任务具有了数据时间，就会用到重刷数据。例如一个工作流最终产生一个报表，但是业务发生变更（新增一个字段）或者发现上一个月的数据有错误，那么就需要重刷过去一个月的数据。通过重刷数据功能，可以重刷某些任务/工作流的数据（只支持天级别），每个实例都是不同的数据时间。

- 失败自动重试
  - 实例失败自动重试：在任务管理的高级配置中，可以配置实例失败重试次数和重试间隔，例如重试 3 次，每次间隔 30 秒。如果重试 3 次仍旧失败，该实例状态才会变为失败，并发送报警。
  - 子任务失败自动重试：如果是分布式任务（并行计算/内网网格/网格计算），子任务也支持失败自动重试和重试间隔，同样可以通过任务管理的高级配置进行配置。

- 报警监控
  - 失败报警
  - 超时报警
  - 报警方式：短信

4 产品优势

SchedulerX 2.0 具备高性能、高可靠、丰富的场景和简单易用等优势。

- 高性能：通过分布式的架构和 Akka 异步特性，支持海量任务和秒级别调度。
- 高可靠：通过主备机制、消息 At-least-once delivery、定期轮检等多种手段，保证任务调度和运行的高可靠。
- 丰富的调度和计算场景：支持定时调度、API 调度、任务编排；支持单机、广播、分布式计算多种计算模型。
- 简单易用：接入简单，提供很多易用的运维工具，如前端可以查看执行记录和运行日志，支持原地重跑、重刷数据等操作。

5 应用场景

SchedulerX的应用场景包含定时触发、定期触发和手动触发调度任务。

SchedulerX 包含以下应用场景。

- **固定时间点触发的任务**

  例如：2016年11月11日0点执行的一次任务。

- **周期性触发的任务**

  例如：每秒钟（或者每小时、每天、每星期、每月等）执行一次的任务。

- **通过控制台手动触发的任务**

  例如：可以通过控制台手工触发任务的调度执行。任务触发执行后，由用户实现的 Job 处理器接口中的代码决定具体要完成的业务逻辑功能（例如扫表、触发 RPC 调用、入库、执行本地脚本等）。