多租户高可用并行任务调度框架①

刘一田, 刘士进

(南瑞集团公司(国网电力科学研究院), 南京 210003)

摘 要: 描述了一种多租户高可用并行任务调度框架 MTHPT 的设计思想、体系结构和实现技术, MTHPT 包括 3 部分: 任务定义与配置、异步并行任务调度模式、消息告警与监视. 任务调度引擎和任务执行组件采用分开部署、异步并行调度和快速回调的模式, 快速释放调度引擎占用的线程资源, 解决了部分任务执行周期长、定时任务无法按时执行等影响业务系统性能的问题. 任务调度配置提供了多租户应用模式. 实验分析及评估表明, MTHPT 提高了应用系统的任务调度并行调度效率和稳定性.

关键词: 多租户; 高可用; 异步调度

Multi-Tenant High Availability Parallel Task Scheduling Framework

LIU Yi-Tian, LIU Shi-Jin

(NARI Group Corporation(State Grid Electric Power Research Institute), Nanjing 210037, China)

Abstract: This paper describes design ideas, architecture and implementation techniques of a kind of multi-tenant high availability parallel task scheduling framework named MTHPT. It consists of three parts: task definition and configuration, asynchronous parallel task scheduling mode, alarm messages and monitoring. Task scheduling engine and task execution components use separate deployments, asynchronous parallel scheduling and fast callback mode, quick releasing thread resource scheduling engine, which can solve such issues as: part of the task long implementation cycle, timed task that can not be performed on time and other issues that affect the business performance of the system. Task scheduling configuration provides a multi-tenant application model. Experimental analysis and assessment show that MTHPT improves the efficiency and stability of task parallel scheduling application system.

Key words: multi-tenant; high availability; asynchronous dispatch

1 引言

在大型系统中, 任务调度是一项基础性的需求. 对于一些需要重复、定时执行或者耗时比较长的任务经常会被剥离出来单独处理, 而随着任务规模与复杂性的上升, 任务调度服务框架也就随需而生. 设计良好的任务调度框架需要具备可靠性及伸缩性, 它可以管理并监控任务的执行状态, 提供稳定可靠的调度模式调度任务, 以保证任务的正确执行.

文献[1,2]概述了多租户环境下多任务调度算法的划分, 不同的调度算法会对系统的整体性能有重要的影响, 目前在实时多任务的调度算法主要采用优先级驱动调度算法, 根据任务的优先级确定时序, 优先级驱动的调度算法分为静态调度和动态调度两种. 在静态调度算法中, 所有任务的优先级在设计时就确定下来了, 且在运行过程中不会发生变化; 在动态调度算法中, 任务的优先级则在运行过程中确定, 并可能不断地发生变化. 静态调度比较简单, 但缺乏灵活性, 不利于系统扩展. 动态调度在增加灵活性的同时增加了复杂性. 这些任务调度的调度算法常采用的时间片轮转和优先级抢占等调度策略, 任务调度方式采用单机多线程的方式调度任务资源, 每次任务调度时都会创建一个新的调度线程, 由于业务系统中部分统计类

① 基金项目:国家电网公司科技项目“基于分布式技术的业务系统架构研究”

收稿时间:2016-05-29;收到修改稿时间:2016-07-25 [doi: 10.15888/j.cnki.csa.005645]

任务本身执行耗时较长, 且调度线程采用同步方式一直等到服务调用结束后才回收, 在多数情况下, 线程资源在长时间空等而形成浪费, 个别任务执行异常也会占用线程不释放资源, 导致调度服务的线程池快速饱和, 出现无法启动新任务、任务延时执行、任务不执行的情况, 且偶尔会发生任务调度服务器宕机的情况.

一种有效的任务调度和分配算法是多核处理器获取高性能的关键因素. 因此, 本文设计了一种多租户高可用并行任务调度框架 MTHPT, 解决业务系统中出现的上述问题, 提升基础软件开发平台的通用技术支撑能力. 通过可视化界面定义任务的执行组件, 任务执行组件支持REST服务和本地服务Bean两种执行组件, 为了增加引擎的稳定性, MTHPT 默认使用 REST 服务接口定义执行组件, 即任务执行组件和调度引擎作为两个独立的应用, 调度引擎只负责触发框架任务执行组件封装器的调用, 由封装器启动应用线程执行任务调度. 任务的定义支持调度规则的配置, 支持手工任务、周期任务和 Cron 表达式任务规则. 在部署架构上支持调度引擎和执行任务分开部署, 在调度方式上采用异步并行执行部署在应用中的框架任务执行组件封装器, 通过执行组件接口约束自定义业务执行组件, 在执行具体的任务调度时, 在任务执行组件封装器中分别启动业务应用服务器线程、调用部署任务的应用线程执行任务、基于 RESTful 服务异步回调告知任务调度引擎任务的调度状态, 及时通知任务调度引擎释放任务调度服务的调度线程, 避免任务调度延迟和调度服务资源占用.

2 MTHPT框架设计

本文描述的高可用并行任务调度框架如图 1 所示, 其主要具备如下特征.

(1) 支持多租户定制个性化的调度任务, 租户可以申请任务调度服务集群中的子集为指定应用系统提供任务调度服务.

(2) 调度引擎和执行任务分开部署: 任务调度的引擎与执行任务需要分开部署, 目的是提高任务引擎的稳定性, 即使二次开发编写的任务组件不稳定, 也不影响任务引擎的稳定运行, 另外也是为了减轻任务引擎应用服务的压力.

(3) 异步并行调度: 调度引擎异步并行调度任务, 调度引擎在调用远程 RESTful 的任务组件时, 远程执行组件分发器接收到调用的消息后, 从线程池中获取一个可用的线程执行任务, 任务分发器非阻塞调用执行组件, 立即返回引擎的调用, 节约了引擎中线程池的可用线程, 当远程任务分发器执行任务完成, 则将执行结果通知给引擎, 引擎收到通知后在任务实例日志表中记录任务实例执行情况, 如任务执行时长、执行结果、执行的异常信息等.

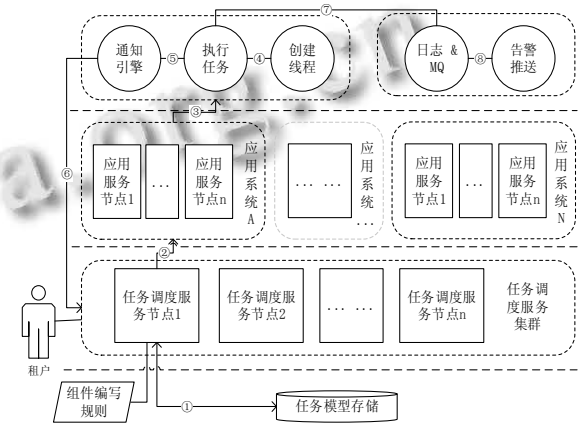

图 1 多租户高可用并行任务调度框架

(4) 任务定义模式: 通过任务执行组件接口约束自定义任务, 任务定义采用 RESTful 服务模式和本地服务组件, 但是为了服务的稳定性, 推荐使用 RESTful 服务模式, RESTful 服务组件支持异步调用, 每一个任务定义都是一个 RESTful 资源映射. 在 RESTful 资源类中分别启动本地应用线程、将启动成功或失败结果告知调度引擎, 使用创建的应用线程执行任务, 并用日志记录执行异常信息.

(5) 监控分析告警: 引擎中的任务在后台调度执行, 无法通过前台查看任务的调用和执行结果, 因此在后台记录任务实例日志及消息总线推送告警两种方式, 通过查看任务实例日志可分析任务在什么时间点被调度, 以及执行的时长、执行是否成功和执行的异常信息等. 对执行任务超时未完成、任务超时未开始等信息发送到消息总线 ActiveMQ[3], 由消息总线推送到前端进行告警消息展示.

(6) 任务之间的灵活依赖: 具有依赖关系的任务调度在传统调度策略中的研究由来已久[4], 典型的依赖任务调度模型都是建立在图的基础之上[5,6], MTHPT 可将任意一个任务作为自己的父任务进行依赖触发.

(7) 调度服务集群与失效转移: 任务引擎的集群

是多套任务引擎服务连接同一个数据库, 任务引擎服务基于数据库资源锁的机制实现任务获取, 集群中任务调度服务争抢已定义的定时任务, 任务调度服务集群中主机出现异常时, 不影响已执行调度任务的作业执行, 集群中其它任务调度服务正常接管调度服务.

高可用并行任务调度框架的执行流程任务调度分为八个步骤, 分别介绍如下.

(1) 编写任务调度业务执行组件 Bean, 配置任务调度规则. 业务执行组件根据 MTHPT 框架的执行组件编写规则, 实现 MTHPT 框架的任务执行组件接口; 完成组件定义后, 以租户身份进入任务管理视图, 在管理视图中配置租户域的业务执行组件类, 并编排任务调度执行组件的依赖关系和执行顺序. 上述过程的定义结果通过数据表的方式存储到任务定义存储中.

(2) 任务调度节点从待执行任务记录表读取记录, 根据任务调度规则在指定的时间点或周期内从数据库中读取待执行的任务, 任务调度节点之间是争抢的关系, 任务调度节点读取任务记录时锁定任务记录表, 判断当前任务是否存在父任务或后继任务, 并递归查找所有关联任务记录, 将查找结果依据排序关系放到任务队列中, 然后将这些记录的调度状态字段值修改为正在调度, 然后释放表锁, 其它任务节点在读取任务记录时, 如果发现任务调度表中记录状态为正在调度, 则放弃本次调度, 顺序查找可调度的下一个任务, 确保一个任务仅在一个任务节点上调度.

(3) 争抢到任务记录的任务调度节点启动调度服务线程, 从任务队列中依次取出任务, 异步触发部署在业务应用节点上的框架执行组件包装器 RESTful 服务并执行回调监听. MTHPT 框架定义了框架执行组件包装器组件和任务执行组件接口组件, 框架执行组件包装器组件中读入参数中的业务执行组件配置, 包装器组件通过依赖反射的方式注入业务执行组件, 在业务应用部署节点应用服务器中创建调度线程执行任务.

(4) 业务应用部署节点应用服务器线程创建成功, 则由该应用线程执行业务执行组件逻辑.

(5) 将业务应用部署节点应用服务器线程创建成功或失败的消息提示发回给任务调度节点的调度服务线程.

(6) 任务调度节点线程根据成功或失败的消息提示, 如果成功则回收当前调度服务线程, 如果失败则执行重新调度.

(7) 业务执行组件逻辑在执行业务逻辑过程中, 将执行完成或异常失败等信息记录在日志记录中, 并同时将异常信息发往消息总线, 由消息总线推送到前端, 执行完成后将任务执行状态修改为已完成.

(8) 分析监控组件根据日志进一步分析业务执行组件的执行情况, 提供监视及分析视图.

3 MTHPT框架实现方法

多租户高可用并行任务调度框架 MTHPT 采用任务执行组件接口限定自定义任务所必须实现的方法, 框架调度引擎仅对框架封装的任务执行组件包装器组件进行异步并行调用, 任务执行组件包装器组件实现了应用线程的创建、任务定义时配置的自定义业务执行组件的实例化和执行, 异步回调通知等关键封装, MTHPT 框架会根据创建线程的结果, 快速返回调度引擎此次调度是否成功. 调度引擎在确定调度结果后, 释放调度服务的调度线程到线程池中. 已启动的应用线程执行自定义任务组件, 当执行异常时用日志记录并将异常信息发到消息总线由其推送到前端. 根据上述 MTHPT 的框架结构的设计, MTHPT 包含几点主要关键技术: ①任务定义和配置. 包括任务组件编写、任务配置; ②异步并行任务调度模式; ③消息推送告警与监视.

(a) 任务定义和配置

任务的模型定义是一个四元组, 可描述为 TD=<TN, TS, TR, BE>, 其中 TN 为租户, TS 是任务服务, TR 是任务调度规则, BE 是业务执行组件. 租户申请需要使用的任务调度服务, 服务可以由任务调度服务集群中的一个或多个节点组成, 通过统一的任务调度规则配置视图入口, 配置任务的调度周期、起止时间、触发次数等规则, 在执行组件的选择上, 可以选择已实现框架约定接口的本地业务执行组件, 或者配置已实现框架约定接口的远程 RESTful 服务地址, 如图 2 所示.

(b) 异步并行任务调度模式

当任务执行组件的定义采用远程 RESTful 服务定义时, 框架的任务执行组件封装器和业务执行组件默认与业务应用服务合并部署, 作为业务应用功能的一部分. 如果任务执行组件的执行比较消耗内存等关键资源, 也可以将组件封装器和业务执行组件分开独立部署.

图 2 任务定义和配置

任务调度服务中的任务调度器从数据库中读取并实例化任务及任务调度规则, 每个任务都对应一个子任务队列, 子任务队列中的任务以同步的方式顺序执行, 任务调度器异步并行调用每个任务对应的子任务队列中的子任务, 触发子任务的执行组件封装器的 RESTful 服务接口, 等待组件封装器的执行结果, 组件封装器在创建线程后即可将创建成功或失败的消息返回给任务调度器, 任务调度器接收到成功消息后, 回收调度线程, 在接收到失败消息后, 重新执行任务. 异步并行任务调度过程如图 3 所示.

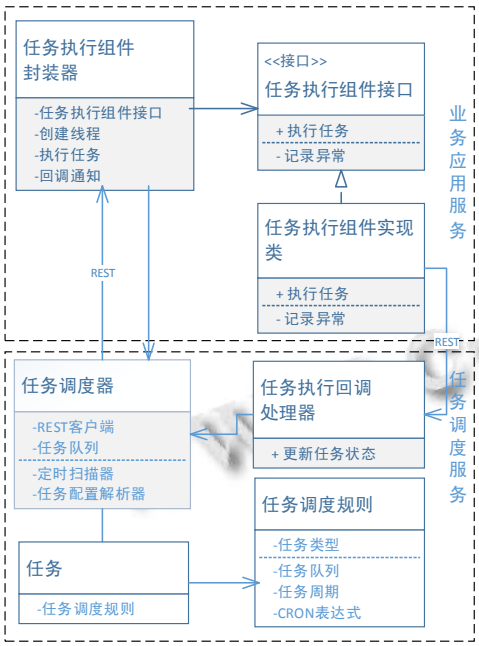

图 3 异步并行任务调度模式

为了实现任务编排功能, 任务执行组件执行完成后, 调用任务调度服务的更新任务状态方法, 更新指定指定任务的子任务队列中的子任务执行状态, 如果子任务队列中存在后续任务, 则继续触发后续任务调度执行.

(c) 消息推送告警与监视

任务执行过程出现异常时, 首先通过日志组件记录异常信息, 并将异常信息发送到消息总线 ActiveMQ 上, 消息的发送采用发布/订阅模式, 消息主题通过配置文件动态获取, 前端任务异常提醒组件以 websocket 方式保持与 ActiveMQ 的长连接, 接收指定消息主题的推送文本并解析, 将解析的结果封装后以弹窗方式提示给用户.

对异常执行的任务执行组件日志进行分析统计, 提供日志监视视图对分析结果进行展示, 能更直观发现任务调度问题.

4 实验分析与评估

实验测试环境由 8 台 8 核 8G 内存的 PC 服务器(HP DL 380 G4 378735-AA1)组成, 其中数据库、任务调度服务、应用服务主机比例为 1:4:3, 数据库和应用服务采用国家电网公司设备(资产)精益运维管理系统的测试数据库和应用服务, 任务调度服务分别以传统的同步调度和 MTHPT 框架模式的异步调度模式各部署 2 台, 任务调度服务和应用服务都通过集群方式提供服务, 调度任务的业务执行组件是对设备(资产)精益运维管理系统已有的统计任务进行修改, 实现了 MTHPT 框架的执行组件接口, 以不同租户身份登录任务调度管理视图后分别以十分钟、半小时、一小时、一天为调度周期配置调度规则, 观察统计任务的调度效率和稳定性.

经过两周的高负荷运行, 根据记录的异常日志等信息, 统计分析如表 1 所示.

表 1 传统任务调度和 MTHPT 调度模式的实验数据对比

	传统同步调度方式	MTHPT 框架调度方式
任务调度服务宕机次数	1	0
失败的任务次数	4	1
延迟的任务数	10	0
超时的任务数	4	1

表中实验数据表明, MTHPT 任务调度框架增强了任务调度服务的稳定性, 避免了任务调度延迟, 提高了调度效率. 但也发现了一些问题, 具体如下:

(1) 任务执行组件在执行结束后要调用任务回调器更新任务状态, 任务状态更新后并未及时反映到任务监视视图中.

(2) 任务中的子任务队列在发生任务调度异常时, 需要对子任务队列中的任务重新执行, 造成一定的资源浪费.

(3) 将需要实时处理的任务放入子任务队列中, 会影响实时性.

目前, 多租户高可用任务调度框架 MTHPT 已在国家电网公司统一应用平台(SG-UAP)中实现并大规模推广应用, 取得了较好的应用效果.

5 结语

本文研究了传统的任务调度策略和任务调度模式, 在此基础上, 设计了多租户高可用并行任务调度框架, 给出了任务调度定义模式和调度引擎优化方法, 阐述了该框架的架构设计以及关键实现技术. 最后, 以基于国家电网公司设备(资产)精益运维管理系统的应用运行实例为背景, 给出一个多租户多任务并行任务定义、业务执行组件接口和任务调度的实现过程. 实验结果、 效率评估及生产运行实践表明, MTHPT 提升了任务调度服务的调度效率和稳定性, 提高了生产应用信息系统的服务水平. 后续将针对遗留问题持续改进 MTHPT 框架.

参考文献

1 Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973, 20(1): 46–61.

2 蒲汛,杜嘉,卢显良.基于用户优先级的云计算任务调度策略. 计算机工程,2013,39(8):155–159.

3 The Apache Fundation. Apache ActiveMQ. http://activemq.apache.org/. [2016-04].

4 Nejad MM, Mashsyekhy L, Grosu D. Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. IEEE Trans. on Parallel and Distributed Systems, 2014, (99): 1–5.

5 Sih GC, Lee EA. A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures. IEEE Trans. on Parallel and Distributed Systems, 1993, 4(2): 175–187.

6 杜晓丽,蒋昌俊,徐国荣,等.一种基于模糊聚类的网格 DAG 任务图调度算法.软件学报,2006,17(11):2277–2288.