

分布式数据库控制协调体系结构的研究与实现

李海荣^{1,2}, 方中纯²

(1 北京科技大学 计算机与通信工程学院, 北京 100080; 2 内蒙古科技大学 工程训练中心, 内蒙古 包头 014000)

摘 要: 应用协调学相关理论, 针对分布式数据库系统的控制协调, 给出一种基于内存缓存、面向执行实体的协调模型 EOCC(Executing Entity Oriented memory Cache based Coordination model). 该协调模型支持用户配置和自定义规则, 有更好的灵活性、动态性、自适应性和可配置性, 并实现了基于该模型的协调软件开发框架 CSF(Coordinator Software Framework), 从而验证了模型的正确性. 最后, 给出系统更新协调过程的实例及与其他协调模型的比较.

关键词: 分布式数据库; 实体; 软件框架

中图分类号: TP311.5

文献标识码: A

文章编号: 1000—7180(2016)09—0114—05

Research and Implementation on the Control-Coordination Architecture of Distributed Database

LI Hai-rong^{1,2}, FANG Zhong-chun²

(1 School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing 100080, China; 2 Engineering Training Center, Inner Mongolia University of Science & Technology, Baotou 014000, China)

Abstract: Aiming at coordination of executing entity of distributed database, a coordination model named EOCC (Executing Entity Oriented memory Cache based Coordination model) is put forward based on coordination theory. This model supports user configuration and customized rules, which is more flexible, dynamic, adaptive and configurable. And CSF (Coordinator Software Framework) is realized based on the model so that the correctness of the model is verified. At last, an example which shows how to update and coordinate and the comparison with other models are put forward.

Key words: scoordination; distributed database; entity; software Framework

1 引言

协调是复杂大系统的核心问题,也是计算机科学、工程技术、社会经济以及生物生态等领域持续发展的普遍需求与共性问题^[1]. 尤其是在得到广泛应用的分布式大数据库系统中,对协调的要求更高,协调模型、协调方法以及协调算法也更加复杂.

在分布式数据库系统广泛使用的今天,如何实现各个系统或者地区的数据库之间很好的协调,是提升整个数据库系统性能的重要研究课题. 目前,虽然分布式数据库系统在协调方面的研究取得了较大

的进展,但总的来说,存在着智能化程度不高、人机交互不够丰富、协调单元的粒度及协调控制的灵活性存在局限等不足,因此,本文提出一种基于内存缓存、面向执行实体的协调模型 EOCC. 该协调模型支持用户配置和自定义规则,具有灵活性、动态性、自适应性和可配置性特点.

2 协调的概念

涂序彦认为^[1]: 所谓协调,就是在大系统内通过控制、调节使各局部小系统既互相制约,又互相配合、互相促进,以实现全局最优化. Coates 认为^[2]:

协调是一种活动,用来管理协作智体之间的依赖与可能出现的冲突,以达到和谐的结果。

近年来,对协调的需求也越来越复杂,国内外对协调领域的研究更加活跃了,并在体系结构、设计与实现方法、协调活动的表示层面等方面取得了较大发展^[3-7]。

3 EOCC 模型

3.1 相关概念介绍

(1) 执行实体

执行实体是指计算机中能够运行的程序组、进程、线程、服务、Agent等。执行实体协调是一种控制协调,是协调实体对整个分布式数据库系统中的功能执行实体的协调。协调实体根据用户定义的或者系统自动学到的规则,对功能执行实体进行控制。具体的协调方法有:递阶协调、导引协调、分组协调、全息协调等^[1],本文采用递阶协调,即用一台数据库节点作为系统的协调控制器,通过这个协调控制器来协调整个分布式系统的功能执行实体。

(2) 数据库节点

数据库节点(Database Node)是一个三元组 $\langle \text{CEE}, \text{FEE}, \text{CDS} \rangle$ 。

其中,CEE(Coordination Executing Entities)是协调执行实体集合,一个数据库节点包括多个用于充当协调功能的执行实体;FEE(Function Executing Entities)是功能执行实体集合,一个数据库节点包括多个执行功能的执行实体;CDS(Coordination Data Spaces)是用于保存协调数据、协调规则以及用户数据等的的数据空间。

(3) 协调管理系统

本文把完成分布式数据库系统协调功能的所有组成元素的总和称为协调管理系统。元素分为协调执行实体和协调数据,其中协调执行实体主要包括进程组、进程、线程、服务、Agent;协调数据主要包括协调需要的配置信息、协调过程中产生的中间数据。

协调管理系统(Coordination Management System)被定义为一个四元组 $\langle \text{CE}, \text{LA}, \text{CD}, \text{CM} \rangle$ 。

CE(Coordination Entities)是参与协调实体的集合,本文主要是进程组、进程、线程和Agent。

LA(Link Architecture)是联系协调部件的链路构成的体系结构。

CD(Coordination Data)是协调需要的数据,主要包括配置数据、协调中间数据等。

CM(Coordination Model)是协调模型,需要协调语言来体现。

3.2 EOCC 模型的体系结构

基于Memory Cache、面向执行实体的协调模型EOCC支持用户配置和自定义规则,具有可扩展性、自适应性、动态性和可配置性等特点。

EOCC利用内存缓存(Memory Cache)进行数据交互,Memory Cache之间通过通讯协调器在后台自动通过网络或者管道同步数据。其体系结构如图1所示。

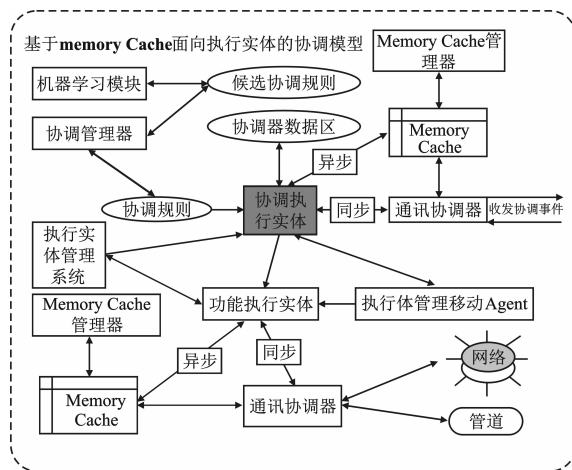


图1 EOCC 体系结构图

(1) 协调执行实体

协调执行实体(Coordination Execute Entity)是一个三元组 $\langle \text{SO}, \text{CI}, \text{KO} \rangle$ 。

SO(Status Operating)是状态操作功能模块,当协调执行实体的运行状态改变时,该模块自动把状态写入协调数据空间中。或者当协调执行实体恢复工作时,从协调数据空间中获取状态。

CI(Communication Interaction)是通讯交互,用于同本机或者网络的其它协调执行实体进行数据通讯。

KO(Knowledge Operating)是协调执行实体的知识操作功能模块,用于从协调数据空间中获取协调知识或者更新协调知识。

(2) 功能执行实体

功能执行实体(Function Executing Entity)是一个三元组 $\langle \text{HCI}, \text{CI}, \text{T} \rangle$ 。

HCI(Human Computer Interaction)是人机交互界面,用于人工对系统进行管理和维护。有的执行实体没有人机交互界面。

CI(Communication Interaction)是通讯交互,用于和本机或者网络的其他功能执行实体进行数据

通讯.

T(Task)是功能执行实体的任务.

系统利用协调执行实体对功能执行实体进行协调. 功能执行实体主要完成对数据库的操作、数据库活动的监视、网络环境的检测、数据库访问日志的规则挖掘等功能,并把数据库操作和监视结果保存于 Memory Cache 中. 在保存数据时,功能执行实体需要把待传送的数据封装成 Memory Cache 包,然后放入到 Memory Cache 中. 为了完成通讯需要一个 Memory Cache 结构体,其定义如下:

```
typedef struct _memory_cache_data
{
    STRINGsrc_work_entity_key;//源功能执行实体标志关键字
    STRINGdst_work_entity_key;//目的功能执行实体标志关键字
    STRINGtrans_type;//数据传输的类型
    STRINGipaddress;//执行实体所在机器的 IP 地址
    STRINGinteract_property;//需要交互的属性名称
    STRINGinteract_value;//需要交互的属性值
}MEMORY_CACHE_DATA, * PMEMORY_CACHE_DATA;
```

下面 5 个原语用于执行实体从 Cache 中检索数据:

out(d): 把 Cache 数据结构的元素 d 放入 Cache 中;

tout(ds): 多个数据以事务的方式放入到 Cache 中,保证这批数据的原子性,对批量数据交互传输提供了一种便利的方法;

in(d): 从 Cache 中取出一个元素放入 Cache 数据结构变量 d 中;

rd(d): 从 Cache 中复制一个元素放入 Cache 数据结构变量 d 中;

registerEvent(d,event_name): 注册一个监视 Cache 的回调函数,当满足条件的数据到达时就会触发这个回调函数,这个原语为高效异步通讯提供了便利条件.

其中,in 和 rd 原语从 Cache 中获取数据可以采用含参数的模式匹配,即可以传入功能执行实体关键字、IP 地址、属性名称等获取指定值. 这种方法便于功能执行实体之间进行高效的、目的性强的交互.

(3)通讯协调器

图 1 中包含两个通讯协调器,与协调执行实体对应的通讯协调器和与功能执行实体对应的通讯协调器.

与协调执行实体对应的通讯协调器传送的主要是协调事件,即通知其他协调执行实体触发某个事件,事件的结构定义如下:

```
typedef struct _coordination_event_data
{
    STRINGcoordination_entity_key;//协调实体的关键字
    STRINGipaddress; //协调实体所在机器的 IP 地址
    STRINGdst_ipaddress;//目标机器的 IP 地址
    STRINGevent_name;//接收该数据需要触发的事件
    STRINGparam;//事件需要传送的参数数据
    STRINGwork_entity_type;//功能执行实体的类型
    STRINGtrigger_flag;//事件的触发标志,包括:// run (启动)、stop(停止)、suspend(挂起)及 resume(恢复).
}COORDINATION_EVENT_DATA, * PCOORDINATION_EVENT_DATA;
```

当协调执行实体接收到协调事件数据时,它就根据事件的名称、参数和功能执行实体的类型来启动或者停止功能执行实体,功能执行实体就完成相应的任务.

与功能执行实体对应的通讯协调器从 Memory Cache 中取出数据,根据数据传输的类型选择是通过管道传输还是通过网络传输. 如果通过管道传输,那么就建立或者查询管道,然后进行数据传输;如果通过网络传输,那么就建立或者查询 Socket,然后进行数据传输.

(4)Memory Cache 管理器

在 EOCC 协调模型中,执行实体之间的交互是依靠 Memory Cache、采用异步通讯方式来进行的. 这部分内存的管理完全交给 Memory Cache 管理器进行管理,包括 Cache 的建立、扩展、回收 Cache 空间等操作. 在 Cache 中的数据都会有有效期,当数据过期后,Cache 管理器自动收回空间.

(5)机器学习模块

机器学习模块通过对数据库节点、网络环境、网络可靠性、数据节点可靠性和执行实体的执行环境的监视,获取一些协调规则作为候选协调规则,系统将根据这些规则进行协调.

(6)协调管理器

协调管理器用于维护协调规则,即通过该工具可以设置各个功能执行实体之间的合作关系. 用户只要在配置文件中描述被协调执行实体的状态及它们的交互关系,协调进程就能根据配置文件自动调度执行实体. 配置文件的编写实际上是一种面向执行实体的编程思想(Executing Entity Oriented Programming,EEOP). 系统利用 Memory Cache 作为

执行实体之间联系的端口,通过这种方法把各个功能执行实体有效联系起来,使它们能够协同配合、分工协作、动态平衡、相互适应,最终达到协调和谐的状态.

(7)执行实体管理系统

执行实体管理系统用于手工管理数据库节点的执行实体,主要包括启动、停止、挂起、删除执行体文件、更新、注册等操作.

(8)执行实体管理移动 Agent

执行实体管理移动 Agent 用于对执行实体进行远程管理,这使得对整个分布式数据库系统的管理非常有效.例如:当一个数据节点加入智能分布式数据库系统中后,只需要在这个数据库节点上部署移动 Agent 平台,其他的执行实体就能通过移动 Agent 进行管理.移动 Agent 负责把在线更新系统的客户端下载到数据节点并启动它,这样,更新客户端就能自动从网络上获取最新版本的执行实体文件、规则数据和配置文件,并部署到该数据库节点上.

EOCC 模型中,根据协调执行实体之间协调命令和功能执行实体之间的通讯方式把协调分为同步通讯协调和异步通讯协调.

同步通讯协调要求协调命令或数据的发送者必

须等待接收者对收到的命令或者数据作出响应后,执行实体才可以继续执行下去.如果系统中的所有通讯都采用同步通讯,那么系统的执行效率将会很低.因此,在系统设计的时候尽量要减少执行实体间的依赖性,尽量使用异步通讯方式进行协调.

4 基于 EOCC 的软件框架

基于 EOCC,本文设计并开发了一个分布式系统的控制协调软件框架 CSF(Coordinator Software Framework).CSF 中内置常用协调的脚本解析功能,能够解析基本的面向执行实体的编程脚本,开发人员可以基于该框架快速开发出系统的控制执行实体,从而动态灵活地对各种执行实体进行协调.并能在框架的基础上扩展脚本解析功能,从而完成更复杂的协调系统,这也充分体现了 EOCC 的灵活性.

该框架具备的主要接口和类有:协调框架公共类(CSFPublic)、规则文件管理类(CSFRuleFile)、规则解析接口(ICSFRuleParse)、规则解析类(CSFRuleParse)、通讯适配器类(CSFCommunicationAdapter)、执行实体管理类(CSFEEntityManger)、协调执行实体类(CSFCoorEEntity).其类图如图 2 所示.

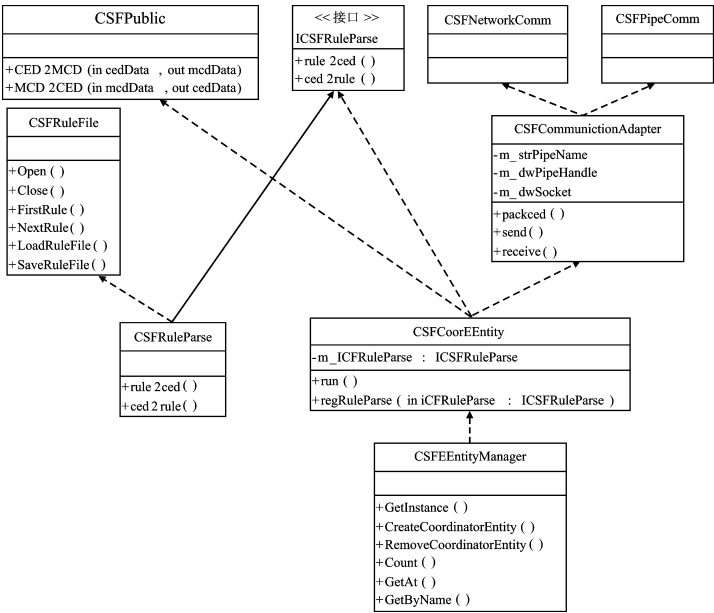


图 2 协调软件框架类图

下面给出应用 CSF 开发协调执行实体的实例代码.

```
VOID ENTITY_MAIN()
{
//实体管理类是一个单件类,系统中有且只能有一个
```

实例

```
CSFEEntityManager em = CSFEEntityManager. GetIn-
stance();
CSFCoorEEntity cee = em. CreateCoordinatorEntity
("test");
cee. regRuleParse (iRuleParse);//注册自己的规则解
```

析类

```
cee.run();//启动协调实体
//不能退出程序,因为协调实体有自己的线程,退出就
结束协调线程
while(TRUE);
}
```

有了 CSF,用户就不需要考虑通讯细节、协调细节和命令处理细节,只需要把精力放到规则的解析算法上.用户只需要开发一个规则解析类,该类实现规则解析接口,也可以继承框架自带的规则解析类,在自己的规则解析类中实现新规则的解析,增强协调执行实体处理规则的功能.

本文的规则是以 XML 格式书写的,该软件框架已经支持的标记主要有:<COOR_RULE>、<LOCATION>、<EE_NAME>、<COMM_STYLE>、<SYS_UPDATE>、<SITE_RANGE

>.开发人员也可以重载 RuleParse 类的规则解析方法,让它具备更好的规则解析功能,从而支持更多的协调规则.

需要说明的是,可以不用 CSF 框架开发协调执行实体,用户完全可以自己编写全部的代码.但这样的话,用户就需要自己解析规则文件,然后根据规则文件的语义执行协调事件和传送协调事件到其他计算机上,也就是要编写与通讯和系统协调相关的代码,因此不仅会增加开发工作量,而且系统的稳定性也不能得到很好的保障.

5 与其他协调模型的比较

从协调对象、交互方式、动态性和宿主语言的依赖性方面,对 EOCC、Manifold 和 LGI 进行比较.如表 1 所示.

表 1 协调模型的比较

特点	Manifold	LGI	EOCC
协调对象	对进程进行协调	对 Agent 进行协调	执行实体(进程、线程、服务、Agent 等)
交互方式	通过相应事件传送命令,通过流来传输数据	通过消息进行交互	协调命令或协调数据通过管道或网络
动态性	用 Manifold 语言写协调进程,对不同的工作进程进行协调	Agent 可以动态加入组中	通过配置文件可以动态配置执行实体的个数、执行情况和交互关系,可以进行 EOCC 编程.
宿主语言	对进程的宿主语言没有要求,任何语言写的进程都能参与协调	主要针对 Java 写的 Agent	对宿主语言没有要求,任何语言编写的执行体都可以添加到该协调模型中

从表 1 中可以看出,EOCC 不仅能够对更多种类的对象进行协调,而且对实现对象的语言没有要求,从而体现出更好的可扩展性和开放性;EOCC 支持用户配置和自定义协调规则,体现了更好的灵活性和智能性.

6 结束语

本文从分布式数据库系统的控制协调方面展开研究,主要研究执行实体间的协调,提出 EOCC 模型,旨在实现整个系统的协调运行和数据共享,达到提高系统的协调性与智能性的目的.

基于 EOCC,本文给出一个软件开发框架,开发人员可以基于这个框架开发控制协调程序,达到更好的协调效果.

最后,应用 EOCC 及 CSF,给出系统更新的协调实例,从而验证了模型及框架的正确性.

参考文献:

[1] 涂序彦,韩力群,马忠贵.协调学[M].北京:科学出版社,2012.

[2] Coates,Graham. Agent Co-ordination aided distributed

computational engineering design [J]. Expert Systems with Applications, 2006,31(6):776-786.

[3] 王成耀,郑雪峰,涂序彦.软件系统协调技术研究[J].计算机科学 2009,36(6):4-9.

[4] 肖正.多 Agent 系统中合作与协调机制的研究[D].上海:复旦大学,2009.

[5] Agostino Dovier, Andrea Formisano, Enrico Pontelli. Autonomous agents coordination: action languages meet CLP (FD) and linda[J]. Theory and Practice of Logic Programming, 2013,13(2):149-173.

[6] Cao Yongcan, Yu Wenwu, Ren Wei, et al. An overview of recent progress in the study of distributed multi-agent[J]. IEEE Transactions on Industrial Informatics, 2013,1(9):427-438.

[7] 闵海波,刘源,王仕成,等.多个体协调控制问题综述[J].自动化学报,2012,38(10):1557-1570.

作者简介:

李海荣 女,(1976-),博士研究生.研究方向为云计算.

方中纯(通讯作者) 男,(1975-),博士.研究方向为人工智能. E-mail:xiaofang611@126.com.