文章编号:1001 - 9081(2010)04 - 1110 - 04

发布订阅模式数据交换中间件设计与实现

高德宏,张新家,陈春雷,刘维宇
(西北工业大学 自动化学院,西安 710072)
(gaodehong_1984@163.com)

摘 要:为了满足当前数据交换中发布订阅需求,以及实时性和复杂性要求,设计了一种基于优先级任务的数据交换模型,并利用该模型开发 TTS 数据交换平台的核心模块。给出数据交换平台的整体框架以及数据集交换流程,并详细介绍任务泵、线程池等模块的实现。实际的应用测试表明,该平台能够实现复杂数据发布订阅应用,满足系统对实时性要求。

关键词:中间件;交换数据集;任务泵;线程池
中图分类号: TP393 **文献标志码**:A

Design and implementation of data exchanging middleware with publish/subscribe model

GAO De-hong, ZHANG Xin-jia, CHEN Chun-lei, LIU Wei-yu
(*College of Automation, Northwestern Polytechnical University, Xi' an Shaanxi* 710072, *China*)

Abstract: In order to satisfy the demands of data subscribe/publish, real-time and complexity in present data exchanging, a data exchanging model based on task-priority model was designed. Using this model the core modules of TopTang Software (TTS) exchanging platform were developed. This paper proposed the framework of TTS platform, discussed all the processes of dataset exchanging, and then, illustrated the implementation of the core modules, including task-pumper and thread-pool. The practical application test demonstrates TTS can satisfy the requirements of publish/subscribe and real-time.

Key words: middleware; exchanging dataset; task-pumper; thread-pool

为了解决分布异构数据库集成问题,数据集成中间件的研究应运而生,传统中间件一般采用消息、队列、广播等方式进行数据通信[1]。但是这些方式难以满足某些数据发布、数据订阅请求、实时消息,及复杂数据交换应用要求。本文针对上述问题设计并实现一种具有优先级的交换任务模型。有专门的交换任务泵激活满足条件的交换任务;同时,系统采用线程池进行实际数据交换,提高整体交换性能。最后将该模型添加到数据交换中间件盛唐软件(TopTang Software,TTS)系统中进行应用测试。

1 中间件数据交换应用框架

TTS 系统是面向企业综合业务数据的交换平台,可以解决大型企业分布异构数据库的数据集和数据交换应用。中间件在数据交换方面的功能主要有数据订阅请求和数据发布应用,如图 1 所示。

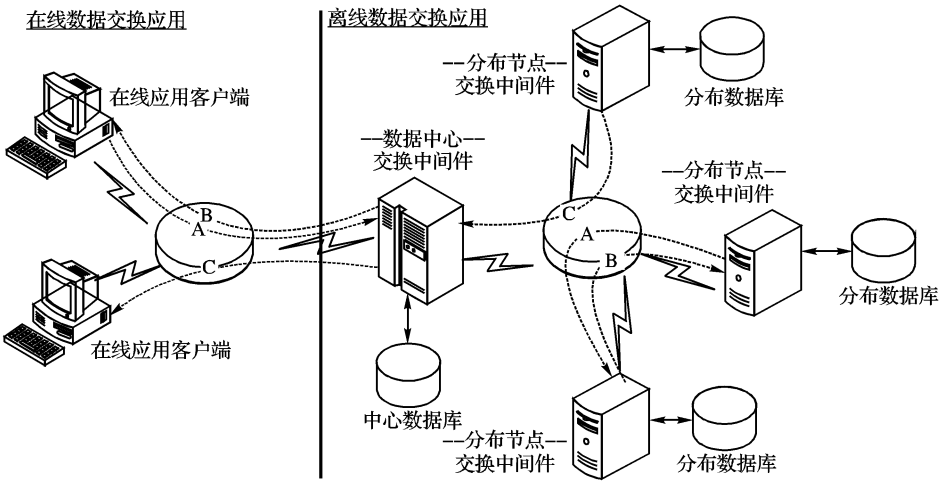


图 1 中间件数据交换应用

1)数据订阅请求:是指本地节点向异地节点发送相关参数,请求异地节点数据库的相关数据,如图 1 中箭头 A,B。实

收稿日期:2009 - 10 - 09;**修回日期**:2009 - 12 - 30。

作者简介:高德宏(1984 -),男,江苏睢宁人,硕士研究生,主要研究方向:网络嵌入式系统; 张新家(1961 -),男,湖北人,副教授,博士,主要研究方向:网络与信息安全; 陈春雷(1983 -),男,山东潍坊人,主要研究方向:计算机网络、软件重构; 刘维宇(1986 -),女,湖北应城人,硕士研究生,主要研究方向:计算机网络、软件重构。

际应用环境中,订阅请求又分为可配置订阅和随机订阅。如图 1 右侧所示,分布节点中间件可以根据业务需求采用立即激活、循环或者定时请求等条件向异地节点发送数据请求,这种应用可以通过应用配置程序在应用系统发布前配置到中间件交换任务列表中,该应用称为可配置订阅。另外,如图 1 左侧所示,系统支持应用客户端程序通过系统提供的在线应用客户端接入程序随机访问数据中心,该种应用称为随机订阅。

2) 数据发布:是指本地节点主动向异地节点发送本地数据库数据, 如图 1 中箭头 C 所示。实际应用环境中,数据发布应用可以分成订阅后发布和自动发布。订阅后发布应用是指数据的发布需要异地节点相关参数激活交换任务;而自动发布应用是指分布节点配置的交换条件主动向异地节点发布本地数据。

在中间件的设计过程中我们采用模块化的思想,将其分成不同的功能的模块实现,根据节点配置信息选择不同的模块组合实现相应的功能。如图 2 所示,根据功能主要分为数据接收模块(ExServer)、数据解析模块(ExParser)、任务泵模块(ExPumper)、线程池(ExPool)。

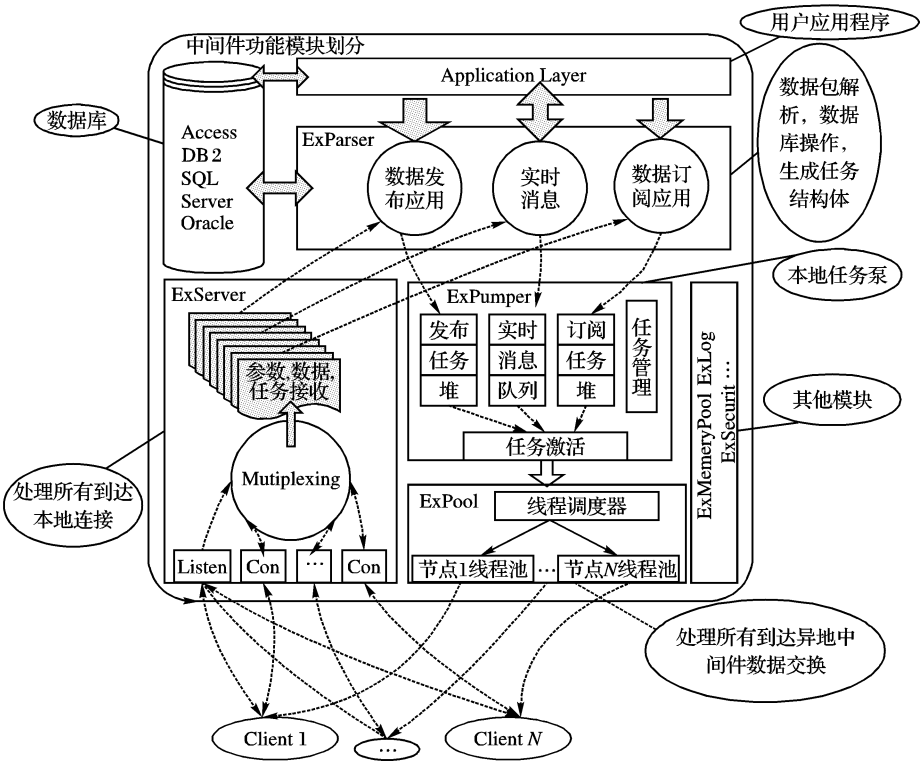


图 2　交换中间件功能模块

2　中间件设计和实现

2.1　交换数据集产生

中间件每次交换的数据称为一个交换集,数据集中包含了一次查询的全部或者部分数据。虽然不同应用的数据集产生方式不尽相同,但是我们使用统一的数据结构 CExTask 结构体。如图 3CExTask 类图所示,任务结构体包含四部分信息:任务标识信息、任务激活条件描述信息、交换数据描述信息、任务内存结构信息。

数据集的产生过程如下:首先如图 2 数据解析模块所示,它负责解析所有应用程序、数据接收模块的数据包[2],并根据数据包类型进行相应的操作,如数据库存储、任务合成等。如果数据包为任务数据包,则根据相应 XML 描述生成交换任务;然后如图 2 任务泵模块所示,该模块按任务优先级顺序判断 CExTask 结构体中条件字段信息,并将满足条件的任务转交给线程池模块;最后线程池模块根据 CExTask 结构体中交换数据描述信息,产生相应数据库查询语句,将查询的结果封装到固定格式的 XML 字符串中形成如下所示的交换数据集。

```
< Action AT = "DSetLoad" ActID = "20" Transact = "1"
   SQL = "select…." Success = "yes" >
    < Records RecordID = "1102" ExID = "192" >  < Record >
       < Col Index = "0" ColName = "ID" DataType = "16" > 12
       </Col >
       < Col Index = "1" ColName = "Name" DataType = "15" >
       xiaozhao </Col >
       </Record >  < Record >
       < Col Index = "0" ColName = "ID" DataType = "16" > 13
       </Col >
       < Col Index = "1" ColName = "Name" DataType = "16" >
       xiaogao </Col >
       </Record >
    </Records >
</Action >
```

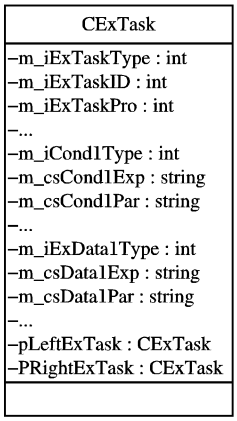


图 3　CExTask 类图

2.2　交换任务的激活发送

交换任务的激活发送操作由图 2 中任务泵模块和线程池

模块。当某交换任务满足条件后,任务泵模块首先激活该交换任务,并向线程池模块发送 WMC_NEWTASK 消息,最后由线程池模块根据任务配置生成交换数据集发送到异地中间件。

2.2.1 任务泵模块设计

任务泵模块设计的目的是对所有交换任务进行分类,按优先级排序,并按优先级顺序激活满足条件的任务。ExPumper 是中间件的核心,其按功能主要分为以下 3 个部分。

1)应用分类器。应用分类器负责将交换任务分类为数据发布任务(PublicTask)/数据订阅任务(SubscribeTask)/实时消息(RTMsg),并将任务添加到对应的内存结构中,同时计算任务的优先级。由于发布任务和订阅任务需要在运行时动态改变优先级,且在大型应用中各个分布节点的任务量较大,需要考虑任务运行时插入删除排序的效率。这里采用最大堆结构。由于实时消息实时性要求高,但任务量相对较少,因此采用优先级链表,每次处理时都将所有实时消息处理完毕后,才处理订阅发布任务。这样既可以满足实时性要求,同时减少排序等过程所浪费的时间。

任务优先级计算公式:

$$P_j = \sum_{i=0}^{m-1} e_i \cdot \boldsymbol{Vec}_{ji};\ i = 0,\cdots,m-1;j = 0,\cdots,n-1 \quad (1)$$

根据式(1),可以计算出每个任务的优先级 P_j,并根据该值为任务排序。其中 $\boldsymbol{Vec}_j$ 是 m 维优先级向量,向量值表示任务 j 特征,如任务的类型、激活时间、激活类型、消息类型和条件判断类型等。e 是对应优先级项的加权因子,其中 $\sum_{i=0}^{m-1} e_i = 1$。假设对于每一个任务 T_i 有两个时间,任务预激活时间 T_{Pj} 和任务实际被处理时间 T_{Ji}(包含判断激活,数据库查询时间等)。假设系统激活任务扫描时间 T_S(该时间是系统间隔一定时间定时扫描系统中任务的时间间隔),则在任务堆中任务 i 被激活时间为:

$$T_i = \sum_{j=0}^{j<i} T_{Ji} + T_S \quad (2)$$

则任务 i 被延迟时间:

$$T_{\omega i} = \begin{cases} 0, & T_i \leqslant T_{Pj} \\ T_i - T_{Pj}, & \text{其他} \end{cases} \quad (3)$$

根据式(2)、(3)可以推出系统总的任务延时为:

$$\sum_{i=0}^{n-1} [(N-i) \cdot T_{Ji}] + N \cdot T_S - \sum_{i=0}^{n-1} T_{Pi} \quad (4)$$

为了提高系统的实时性,使式(4)达到最小,要尽量减少任务扫描时间间隔 T_S,但是该值较小时会造成系统频繁地进行线程之间的切换,具体 T_S 的确定见任务激活器相关叙述。同时,根据式(4)提高系统实时性可以降低处理时间较长任务的优先级。在程序中通过改变加权因子 e 的值,降低处理时间较长任务的优先级,增强系统实时性。

2)任务管理器。任务管理器负责对任务堆和任务链表管理,例如任务的插入、删除、更新操作。同时,需要根据当前状态调整任务优先级,最后选取合理的任务激活策略,避免产生任务“饥饿”现象,同时在该层中实现任务过滤功能。任务管理器设计流程如图 4 所示。

当数据解析模块产生新交换任务,会发送相应的消息通知任务管理器。任务管理器捕捉到该消息后,根据消息类型首先对相应的任务堆或队列进行排序;然后调整相应的任务激活指针。如图 4 所示,各种应用的优先级顺序为:实时消息、数据订阅应用和数据发布应用。每次任务管理器对任务堆或者任务队列排序后,激活指针都指向堆的根节点或者队列的头节点。

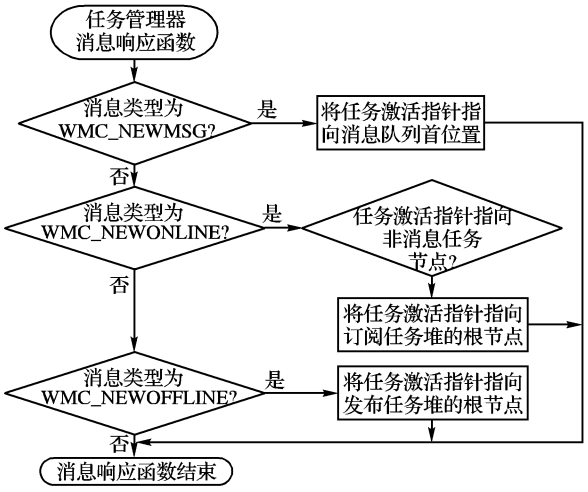


图 4 任务管理器流程

任务调度可能发生任务饥饿现象,即当任务的优先级较低时任务很难被激活,所以当任务被激活后,采用立即降低被激活任务优先级方法避免任务饥饿[3],待任务处理完毕后恢复任务优先级。任务管理器的另一个功能是进行任务过滤,根据任务类型和预先的配置规则修改过滤任务的参数、或者禁止某些任务的调度。

3)任务激活器。任务激活器负责依据任务优先级顺序,将满足条件的任务提交给线程池。任务激活的难点在于:任务扫描间隔时间 T_S 的确定(该值的确定如图 5)。保证应用实时性,要求系统间隔一定时间去扫描任务堆或任务链表,但是频繁激活测试操作会造成系统反复进行线程切换[4],大大降低系统有效操作效率。为此设定系统进行切换的最小阈值 δ,以及响应超前值 ε。在任务添加或者删除过程中,不断修改系统激活时间 T_p,使系统既能尽力保持任务实时性,同时能保持较高的处理效率。

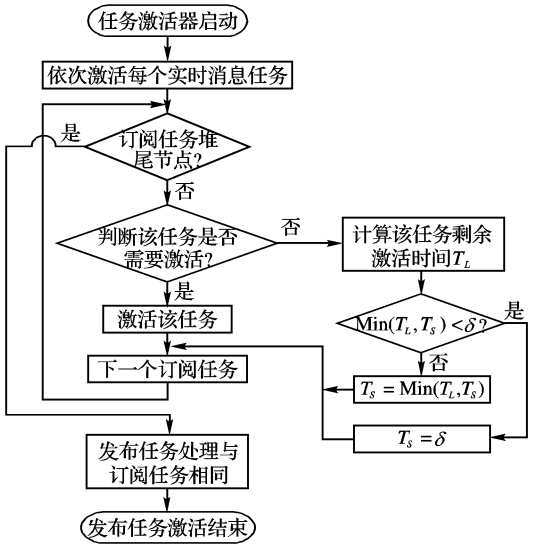


图 5 任务激活器扫描间隔时间 T_S 的确定

T_S 的确定是根据所有未激活的任务剩余间隔时间的最小值,和线程切换最小阈值δ的最小值。若 T_S 小于δ,则激活任务间隔时间 T_S 为 δ,反之为其本身。

2.2.2 线程池模块

线程池模块负责发送激活任务所产生数据集或消息。线程池采用静态生成、动态分配的策略[2],并通过信号量控制

线程间同步。交换线程空闲时处于阻塞状态(线程池状态图见图 6)。

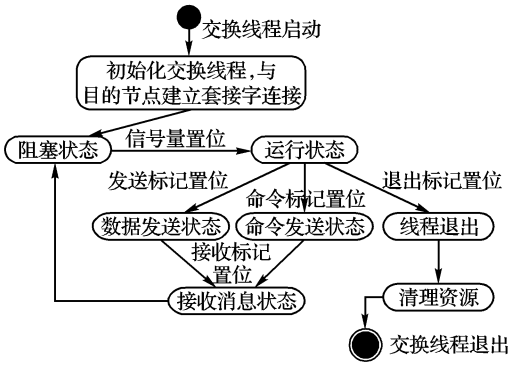


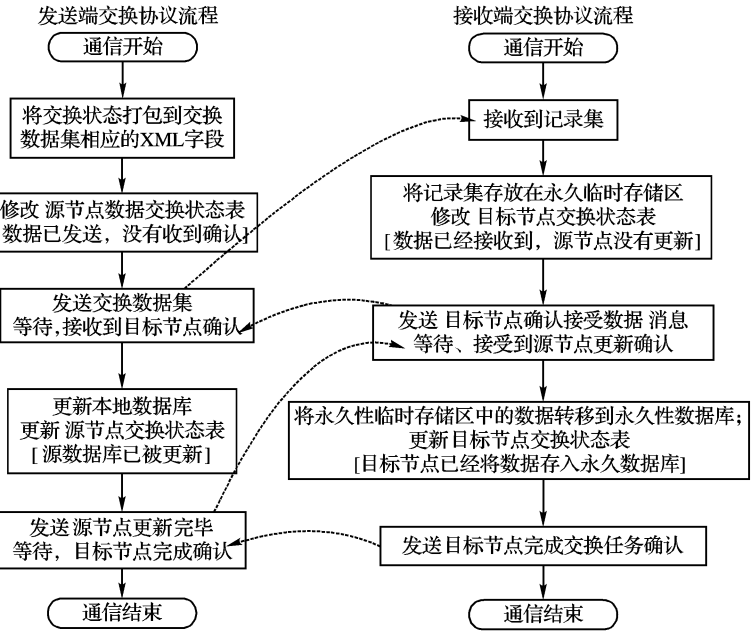
图 6　交换线程状态

有激活任务到达时,线程调度器根据负载平衡原则选择合适的空闲线程,并将激活任务挂载到该线程的处理指针上,同时释放该线程信号量[3]。线程获得信号量后会立即处理发送指针上的任务。线程根据任务结构体中交换数据描述信息生成相应的 SQL 查询语句,并进行相应数据库查询,生成相应的交换数据集。线程使用 Window 自恢复的信号量,当交换任务处理完毕后,该线程的信号量也恢复成占用状态,使该线程阻塞,等待下次调度[2]。

2.3　数据集接收

实际数据交换过程中需要考虑数据集可靠接收问题,防止因网络异常,系统错误而产生的数据库数据不一致问题。图 6 显示系统采用的交换策略主要框架,并且在交换过程中向交换数据集添加交换状态字段传递数据交换状态(如图 7 中交换数据集格式所示)。系统启动和重启阶段根据数据集交换状态进行相应的处理。

如图 7 所示,接收端接收到交换数据集后,1)接收端将数据集存储到缓存到特定数据表中,并在同一数据库事务中记录数据集交换状态;2)等待发送端更新数据库,及其更新数据库确认;3)接收端接收到更新确认后,读取缓存数据表的交换数据集,将解析的结果存入到交换任务实际的交换数据表中,并同一数据库事务操作中更新数据集交换状态;4)接收端发送交换结束确认,至此数据集交换完成。




```
交换数据集XML格式示例:
<SWTPush SrcPushID="2" PushID="200">
<PushIndentify Seq="0520092420" Type="0" SrcState="0" DesState=
"0">XXX</PushIndentify>
<Records><Record>
<Col index="0" DataType="16">Dehong</Col>
</Record></Records>
</SWTPush>
```

图 7　数据集交换策略

3　结语

本文根据交换中间件在数据订阅和发布实时消息等应用的需要,提出一种带有优先级任务泵的交换中间件模型,并给出数据交换的总体流程。最后,将该系统应用到智能交通信息采集系统上,测试表明该中间件能够满足数据订阅和发布应用,能够满足一般系统实时性要求,可以节约系统开发周期和成本。

参考文献:

[1]　段雪峰,张新家,戴冠中. 中间件服务性能建模与分析[J]. 计算机应用,2004,24(1):106 - 107.

[2]　王华锋,张新家. 三层结构的网络服务器设计与实现[D]. 西安:西北工业大学,2007.

[3]　张中庆, 梁雪平. Apache 源代码全景分析 I[M]. 北京: 电子工业出版社, 2009.

[4]　JOHNSON M H. Windows system programming[M]. 安娜, 吴明军,译. 北京: 机械工业出版社, 2006.

[5]　STEVENS W R. UNIX network programming [M]. 杨继张,译. 北京: 清华大学出版社, 2000.

[6]　COMER D E, STEVENS D L. Internetworking with TCP/IP Volume III [M]. 赵刚, 林瑶,译. 北京: 电子工业出版社, 2001.