面向多任务的报表管理系统运行器①

艾 飞[1,2], 王 宁[2], 赵 奎[2]

[1](中国科学院大学, 北京 100049)
[2](中国科学院 沈阳计算技术研究所, 沈阳 110168)

摘 要: 在“辽河流域水环境管理技术综合示范”项目中, 许多的业务需求需要以报表的形式呈现数据. 传统的方式是把每张报表的产生过程固定到每一个业务系统中, 报表需求发生变化时需要修改源程序才可以应对多变的需求, 此外报表的样式比较单一, 不能更加直观的显示数据. 现实中报表需要以多种不同的样式呈现数据, 并且可以应对需求多变的报表数据请求. 为了灵活的制作和使用报表, 我们把所有报表的需求抽离出来, 抽离的部分形成报表管理系统. 报表管理系统以服务的方式满足业务系统的需求, 只要业务系统按照规定的格式请求数据, 报表管理系统就能按照需求输出报表. 报表管理系统以服务的形式呈现给业务系统, 不仅可以支持单系统访问, 同时还可以支持并发的多系统访问, 任何按照报表管理系统标准接口编写的程序都可以使用报表管理系统提供的服务.

关键词: 报表; 报表管理系统; 服务; 并发; 运行器

Statement Management System’s Operation Device for Multiple Tasks

AI Fei[1,2], WANG Ning[2], ZHAO Kui[2]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology of Chinese Academy of Sciences, Shenyang 110168, China)

Abstract: In the project of "Comprehensive Demonstration of Water Environment Management about LiaoHe River Basin", many business requirements need to present the data in the form of statements. The traditional way of processing statements is fixed in each business system, and the style is single. In reality, statements need to be present in a variety of different styles, and can respond to changing data request. In order to satisfy the demand of business system, we put the production process of statements out which formats the Statement Management System. Statement Management System is provided to the systems as the service. As long as the systems in accordance with the provisions request data, Statement Management System will be in accordance with the requirements of output statements. Statement Management System can not only support single system, also support multiple systems concurrently. Any business system can use Statement Management System if only they comply with provisions.

Key words: statement; statement management system; service; concurrently; operating device

传统的报表管理系统大致分为两种形式: 第一种形式是作为一个独立的软件运行, 经过一系列的配置和操作后可以得到所需要的报表, 但是很难直接集成在其他的业务系统当中; 另一种则是把报表管理系统所用到的接口进行封装, 提供给程序员使用, 这样被封装的接口以插件的形式集成到每个系统中, 这样的方式需要程序员熟悉接口的使用, 并没有摆脱通过修改程序来应对多变的报表需求的困扰, 每当重新开发一个业务系统, 必须重新加载插件, 配置工程, 这样就会有很多重复的工作, 浪费人力、财力, 而且业务系统与报表模块之间有很高的耦合度, 报表得不到统一的维护. 参考以往两种不同的报表形式, 以及面向多

① 基金项目:国家水体污染控制与治理科技重大专项(2012ZX07505)
收稿时间:2013-05-15;收到修改稿时间:2013-06-21

任务思想的启发[1], 我们希望达到一个目标: 报表管理系统在使用简单的基础上, 也方便与业务系统集成, 可以为现如今大多数网络业务系统提供报表服务, 直接作为它们的一个功能模块, 简化其结构和开发周期, 为灵活多变的企业级报表提供有力的软件支持. 最终报表管理系统要与“辽河流域水环境管理技术综合示范”项目无缝的连结, 除此之外, 报表管理系统也要灵活的与以后的业务系统对接, 支持多系统并发访问.

1 面向多任务研究

我们所设计的报表管理系统目的是把它作为一项服务, 任何授权的系统都可以直接访问, 试着想象一下, 在一个公司里会有几个甚至十几个系统会有报表的需求, 那它们可以按照规定的格式向报表管理系统发出请求就可以得到自己的数据了, 而且报表样式多样. 该系统也可以运行在互联网上, 为任何接入的系统服务. 图 1 展示了访问方式转换后的对比, 左边部分是传统的插件方式, 右边部分是把报表相关操作抽离后的系统结构.

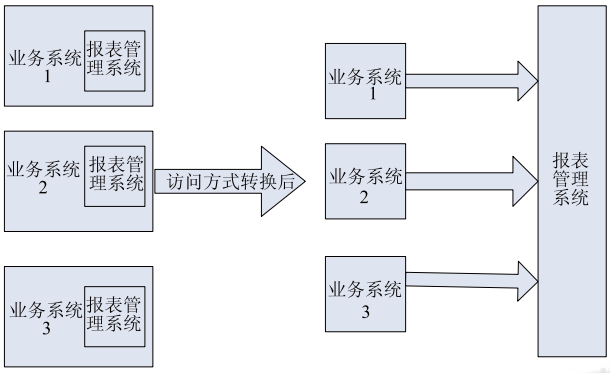

图 1 报表管理系统的访问模式转换对比

访问方式转换后, 每个业务系统的体积明显变小, 它们的报表需求可以更加统一的管理. 以后任何符合标准的业务系统都可以挂接在报表管理系统上.

2 报表管理系统运行器

2.1 运行器简介

设计运行器的是为了达到报表管理系统可以面向多任务的目的. 运行器的功能主要包括: 组织业务系统发来的请求, 管理报表管理系统提供的服务, 根据请求执行服务返回结果. 运行器可以认为是充当了业务系统和传统报表管理系统之间桥梁的角色. 如图 2 所示为报表管理系统的结构图[2].

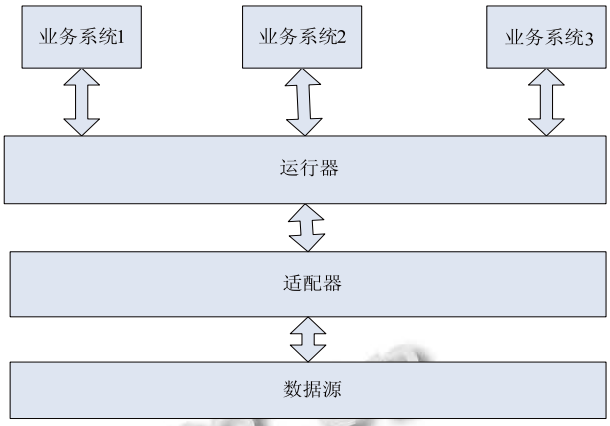

图 2 报表管理系统结构图

适配器提供了服务执行需要的基本组件, 组件描述了对不同数据源的操作, 屏蔽了不同数据源之间的差异, 包括了数据的查询统计等基本功能.

数据源是原始数据存储的地方, 数据源主要涉及到不同的数据库, 甚至包括了普通的文本文件.

2.2 运行器结构设计

我们对报表管理系统的运行器做了分层设计[3], 报表管理系统运行器主要包括请求处理, 服务调度, 服务三层.

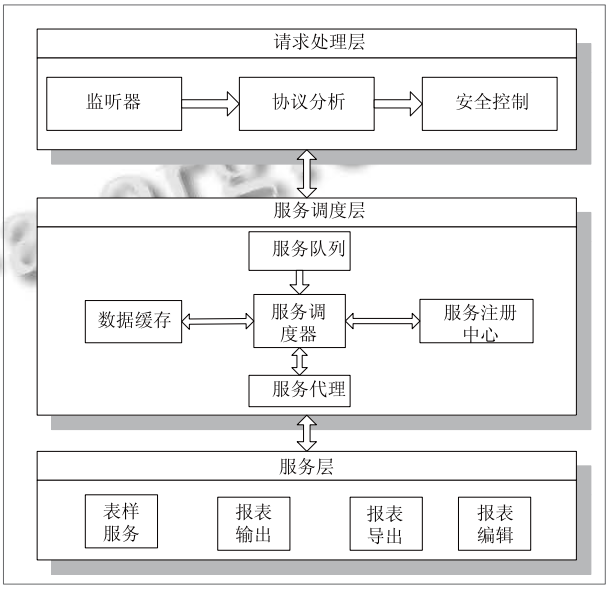

图 3 运行器层次结构图

2.3 各个部分功能

2.3.1 请求处理层

请求处理层主要分为三部分: 监听器, 协议分析, 安全控制.

报表管理系统对外提供统一的接口, 有报表需求的业务系统, 按照接口的标准向报表管理系统发出请求. 一个报表的请求过程需要业务系统发送两次请求: 第一次是表样的请求, 报表管理系统响应第一次请求返回表样, 用户看到表样后, 为表样填充参数, 接着发送第二次请求, 最终返回数据.

其中监听器负责监听请求, 协议分析负责从请求中提取数据, 安全控制负责安全验证.

例如, 当用户的发送的请求是 HTTP 报文请求时, 监听器监听到新的 HTTP 请求到来; 协议分析从监听到的请求中提取 HTTP 版本号、请求方法、身份验证参数、表样参数等数据, 反序列化为系统可以处理的对象; 安全控制进一步进行身份认证, 登记日志等工作.

2.3.2 服务调度层

服务调度器并行的调度服务队列中的请求, 查询服务注册中心, 看是否存在该服务, 如果存在该服务则需要继续查询执行该服务所需要的资源, 把查询结果返回服务调度器, 服务调度器把服务执行的资源传给服务代理, 服务代理负责执行服务.

2.3.3 服务层

服务代理将使用服务层的资源. 服务层描述的是粗粒度的服务, 主要包括处理表样请求、报表请求、报表导出等, 当有新的服务出现时, 可以直接在服务层添加服务, 然后修改配置文件即可, 但是这部分一般是相对固定的, 新的服务更多的是指细粒度的服务, 直接由服务注册中心管理.

3 部分功能模块的设计

3.1 监听器

报表请求过来后会经过协议过滤栈的过滤, 判断是什么样的协议请求, 针对不同的协议请求做预处理, 预处理后会被存入监听队列, 监听队列里存储的信息是最原始的信息. 这些最原始的信息会经过协议分析反序列化为系统可以处理的对象, 之后为了系统安全还需要对每一个已经反序列化的对象进行身份认证, 登记日志等操作.

报表管理系统是面向多任务的, 系统必须做到及时处理不同业务系统发送过来的请求, 所以需要做并行的处理, 但是又需要考虑到系统的资源的实际情况, 保证系统的正常运行, 我们需要限制并发的用户. 我们给出两个属性: ListeningThreadCount 和 ListenedQueueLength, ListeningThreadCount 是协议处理器的并发的线程数, 用来监听业务系统的请求, 当 ListeningThreadCount 到达上限后, 后续来的请求将会被直接丢弃, 直到有新的线程空出. ListenedQueueLength 表示监听到的等待处理的请求的最大数目, 这样保证系统中待处理的服务数目保持在小于 ListeningThreadCount+ListenedQueueCount 的条件下.

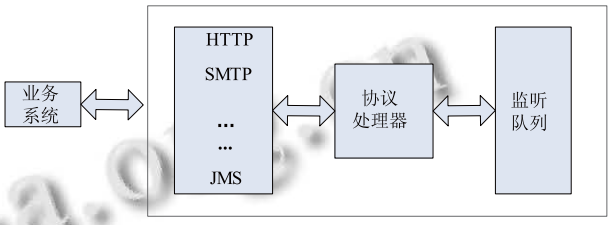

图 4　监听器结构图

3.2 服务注册中心

服务注册中心提供服务的查询, 注册, 删除功能. 服务注册中心描述的服务是细粒度的, 它面向每个业务系统, 业务系统每个报表的请求对应一个配置项, 配置项描述了服务执行所需要的资源,如下所示.

```
<services>
    <service>
            <service_id>1</service_id>
            <template>report1.jrxml</template>
            <action>
              <action_name>
                ProduceReport
                </action_name>
                <url>
              RunningMachine\\ProduceReport
                </url>
        </action>
    </service>
    .....
    </services>
```

我们利用“service_id”元素找到数据模版和服务执行的类. 返回给服务调度器, 服务调度器调用服务代理, 利用 java 反射原理初始化对象, 为对象的属性赋值, 执行对象的相应的动作, 最后返回结果.

3.3 数据缓存

有一部分的报表结果的产生需要较长的时间, 如

果每一次都重新计算的话, 会影响用户的体验和系统的性能, 因此我们设计了数据缓存, 数据缓存必须解决读写阻塞的问题, 为此采用了多版本的并发控制[4], 在读数据时, 读取符合自己要求的数据, 写数据时则产生一个新的版本, 把旧版本的数据维护起来. 这种并发是基于时间戳并发控制的一个变种. 其中服务队列设计如表 1 所示, 数据缓存表设计如表 2 所示.

表 1 服务队列

字段解释	字段名称	备注
服务编号	service_id	
服务类型	service_type	“1”表示数据请求
服务运行参数	request_parameter	
是否允许系统缓存数据	allow_cache	True 表示可以缓存
是否重复计算标示	computer_repeat	True 表示重新计算
系统标识	system_id	

表 2 数据缓存表

字段解释	字段名称	备注
服务编号	service_id	
创建时间	create_time	
文件的使用数	read_count	读文件的进程数目
访问标记	access_flag	True 为可访问，当有重新计算后值为 False
数据	concrete_data	

基于表 1 表 2 的设计, 服务调度器借鉴了计算机操作系统请求分页中的原理[5], 设计了服务调度器中关于数据缓存的访问策略, 具体伪代码如下:

```
if(service_type=1&&allow_cache=true){
if(是否重复计算标示=false){
        if(数据缓存中存在 service_id 对应的数据&&数据没有被禁止访问){
          read_count++;
          读取数据返回给用户;
          read_count--;
        }
        else{
          服务代理重新计算数据;
          返回数据给用户;
          在数据缓存中添加新项, 存入数据;
        }
    }
    else{
        设置 access_flag=false,禁止新来的读请求访问数据;
        服务代理重新计算数据;
        返回数据给用户;
        在数据缓存中添加新项, 存入数据;
        设置 access_flag=true;
        当 read_count=0 时, 删除旧版本;
    }
}
```

我们知道在计算机中, 如果内存过大, 要耗费大量的时间去查询对应的数据是否存在, 这样也严重影响系统的性能,因此数据缓存不是无限大的, 假设数据缓存可以存放 N 条数据, 那么 N 必须满足:

$N*Time1<Time2$

Time1 表示在数据缓存中查询一条数据的平均时间, Time2 是计算复杂报表所需的平均时间. 在不知道数据缓存命中率的情况下, 上面这个公式是保证系统性能的必要条件. 当数据缓存到达上限的时候, 我们简单的采取“最近最久未使用置换”算法[5]删除部分陈旧的数据.

4 结语

本文中设计的运行器实现了面向多任务多业务系统的需要, 有效的为业务系统提供了报表支持, 同时简化了以后业务系统的开发, 降低了业务系统与报表模块之间的耦合度. 但是此报表管理系统作为一个通用的系统, 对于业务系统比较特殊的需求可能就会无法兼顾.

参考文献

1 刘卫宁,刘波,孙棣华.面向多任务的制造云服务组合.计算机集成制造系统,2013(1):199–209.
2 杜万雅.基于 ESB 的 SOA 框架的设计与实现[硕士学位论文].北京:北京交通大学,2008.
3 龙丽萍.ESB_SOA 架构在企业应用集成中的应用和研究[硕士学位论文].长沙:中南大学,2009.
4 萧美阳,叶晓俊.并发控制实现方法的比较研究.计算机应用研究,2006,(6):19–22.
5 汤小丹,梁红兵,哲风屏,汤子瀛.计算机操作系统.西安:西安电子科技大学出版社.2007:144–152.