IDP 平台中轻量级业务框架①

吕明倍，李 炜

(北京邮电大学 网络与交换技术国家重点实验室，北京 100876)
(东信北邮信息技术有限公司，北京 100191)

摘 要：IDP（Integrated Data Platform，综合数据业务平台）是开放的、模块化的、基于标准的基础平台，为移动运营商提供了连接底层各种基础服务以及连接上层各种业务的通用平台。基于 JBoss 的 IDP2.0 系统在应用过程中，遇到了占用资源多，性能差，维护成本高等问题。为了解决上述问题，介绍了一种 IDP 轻量级业务框架的总体设计方案，并重点介绍了业务逻辑模块 DSP（Data-service Script Processor，数据业务脚本处理器）的设计。该框架具有高性能，易于维护，便于快速开发业务逻辑等优点。

关键词：综合数据业务平台；轻量级框架；脚本驱动；Lua

Lightweight Business Framework in IDP

LV Ming-Bei, LI Wei

(State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)
(EBUPT Information Technology Co. Ltd., Beijing 100191, China)

Abstract: IDP (Integrated Data Platform) is an open, modular, standard-based platform. It provides a common platform which joins all the basic services and the above services together. The IDP2.0 system based on JBoss having encountered a lot of resources, poor performance, high maintenance costs problems in the application process. This paper presents a general design of lightweight framework for IDP, highlighted the design of DSP (Data-service Script Processor). The framework has a high-performance, easy maintenance, facilitate rapid development of business logic and so on.

Key words: IDP; light-weight framework; script driven; Lua

1 引言

目前移动运营商正在从移动通信专家向移动信息专家转型，随着移动运营商角色策略的转变，整个增值业务市场也在转变，移动运营商已经开始对市场进行整合，同时开始自营业务的试探。按照目前的趋势，一个综合性的数据业务平台将是移动数据业务的发展方向。综合数据业务平台(Integrated Data Platform 简称 IDP)是开放的、模块化的、基于标准的基础平台[1]，向下提供与短信网关、彩信中心[2]、BOSS(Business Operation Support System，运营支撑系统)、WAP (Wireless Application Protocol，无线应用协议)网关等功能实体的通信，向上提供用户鉴权、短信 / 彩信的编辑和发送、用户管理等公共功能。其主要能力和结构建立在满足现有数据业务产品的基础上。业务开发人员不需要再关心底层与其他实体的通信，可以更关注于业务逻辑的开发。另外，此平台还集成部分基础的业务功能。

IDP2.0 平台已经在多个地区进行了部署和应用，在应用的过程中，逐渐发现了一些问题：

1) IDP2.0 使用 JBoss[3]作为 JAVAEE 容器，JBoss

① 基金项目:国家杰出青年科学基金(60525110);国家重点基础研究发展计划(973)(2007CB307100,2007CB30710);国家自然科学基金(61072057, 60902051);中央高校基本科研业务费专项资金(BUPT2009RC0505);国家科技重大专项(2011ZX03002-001-01,2011ZX03002-002-01)
收稿时间:2011-06-27;收到修改稿时间:2011-07-30

过于庞大，部署和维护都比较困难；

2) JBoss 是重量级容器，启动之后，消耗的内存和 CPU 资源较多，业务逻辑运行的效率比较低；

3) JBoss 要求数据库支持事务，无法和其他基于非事务数据库的平台共用数据库；

4) 使用 EJB3.0 作为业务逻辑开发的工具，调试不方便，开发效率低，不能快速响应业务需求，并且维护代码不方便。

为了解决 IDP2.0 系统的上述问题，需要开发一套轻量级的业务框架，这就是 IDP3.0 平台。IDP3.0 平台的具体需求如下：一个轻量级的业务框架，以实现数据业务的快速开发，同时满足性能需求。能够实现业务逻辑的动态加载和动态更新。具备与其他外部实体交互的能力和网管告警功能。

2 脚本驱动模式

针对引言中提出的问题和软件需求，我们给出一个整体的解决思路，这个思路称作脚本驱动的软件设计模式[4]。

脚本驱动的软件设计模式是一种混合的编程模式，即在软件开发过程中，使用多种编程语言共同完成开发。最常见的使用方式就是“系统编程语言 + 脚本语言”的组合进行开发。脚本语言相比传统编程语言的优势在于：

1) 没有复杂的数据类型，在运行时进行类型动态检查，减少非法操作的可能性[4]。

2) 不需要进行编译和打包，容易部署，可以通过脚本解释器批量执行。

3) 易于编辑，不需要庞大的 IDE 进行开发，用普通的文本编辑器开发即可。

4) 语法简单，关键字少，学习门槛较低。

在 IDP3.0 系统中，我们使用脚本驱动的模式进行开发，可以提高系统灵活性，实现业务逻辑的动态加载，减小模块自身的细粒度，降低模块间耦合度，最大程度的实现功能模块的复用，降低业务开发的门槛，实现快速开发。

虽然脚本驱动的开发模式具有以上等优点，但是由于该模式使用脚本语言进行业务流程的开发，所以它也有着脚本编程语言固有的缺点，即脚本语言的执行速度相对于系统编程语言来说还是相差很远。在应用中，要根据实际情况，把逻辑固定的功能模块和需要灵活多变的流程区分开，将固化的功能模块使用性能较高的系统编程语言实现，然后使用脚本语言调用和组装，来满足系统对整体性能的要求。

3 脚本语言的选择

3.1 Lua

Lua 是一个小巧的脚本语言。该语言的设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。Lua 采用 ANSI C 编写，只要系统中有 ANSI C 编译器，就可以运行 Lua，通过这种方式实现跨平台。Lua 脚本可以很容易的被 C/C++代码调用，也可以反过来调用 C/C++的函数，这使得 Lua 在应用程序中可以被广泛应用。一个完整的 Lua 解释器不过 200k，在目前所有脚本引擎中，Lua 的速度是最快的。

3.2 Squirrel

Squirrel 是一种弱类型的动态语言，语法与 C/C++很相似，它从著名的 Lua 语言继承了很多特性，适用的范围也与 Lua 语言相似。Squirrel 不是解释执行的语言，它有一个编译器和虚拟机，代码会被动态编译成字节码，然后在虚拟机上执行，虚拟机是建立在标准 libc 基础上的。

3.3 Python

Python 的特点是简单，易学，免费、开源，并且具有脚本语言中最丰富和强大的类库。Python 对于 C 和其他语言有良好支持，可以把 Python 作为一种“胶水语言”（glue language）使用，即使用 Python 将其他语言编写的程序进行集成和封装。

3.4 Lisp/scheme

LISP 全名为 List Processor，即列表处理语言，是一种基于 λ 演算的函数式编程语言。由于 Lisp 元语言的特性，很多新的编程语言的特性都可以用它来模拟和验证。

3.5 脚本语言对比选择

上面介绍的这几种语言中，Lua 和 Python 是在这种模式中使用得最多的。相对来说，Lua 体积更小，执行速度更快，对比来说，Python 的虚拟机更大，执行速度更慢。具体到 IDP3.0 的开发需求来说，我们选择了 C++和 Lua 的组合，以 C++为底层语言，以 Lua 为高层语言，使用 C++开发功能相对固化的组件，使

用 Lua 开发业务逻辑，调用底层组件。这两种语言的组合较好的满足了 IDP3.0 的需求。

4 总体设计

4.1 IDP3.0 总体结构

如图 1 所示，IDP3.0 系统主要包含四个部分：业务逻辑模块 DSP（Data-service Script Processor），消息总线模块 DBUS[5]，缓存模块 DCACHE，接口模块。系统对外通过通用消息平台 UNISMS[6]与短信网关连接，通过 MMSCGATE 模块与彩信网关连接。本文主要介绍系统中业务逻辑模块 DSP 的设计。

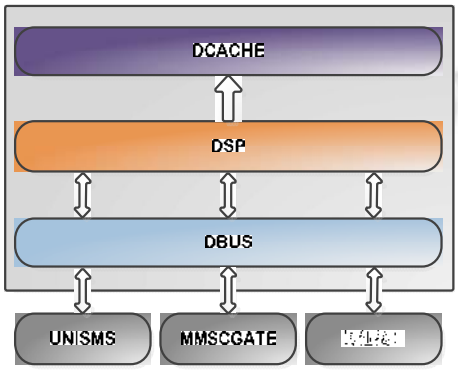

图 1 IDP3.0 总体结构图

4.2 有限状态自动机模型

有限状态自动机[7]是计算理论中的一种模型，一个自动机具有以下几个要素：

1) 状态集合。

2) 输入消息集合。

3) 状态转换函数，以自动机当前状态和输入消息为参数，结果为下一个状态。

在通信系统中，一般对时间都有着比较高的要求，任何一项操作或者信息交互都会要求在一定时间内完成，这是通信系统中的自动机模型和计算理论中的自动机最大的不同之处，我们将这种自动机模型称为有限时间有限状态自动机。

有限时间有限状态自动机可以看成是通信系统中最简单的一种模型，具有以下特点：

1) 具有有限种状态，包括唯一的开始状态，以及结束状态的集合。

2) 自动机接收外部消息作为事件，事件会触发自动机状态的转换。

3) 自动机状态转换满足两个条件，不能由其他状态转换到开始状态，不能由结束状态转换到其他状态。

4) 除开始状态以外，每一个状态都有一个对应的超时状态，如果在一定时间内没有转换到其他状态，则自动转换到超时状态。

这种自动机模型可以满足通信网中一些简单协议和业务，我们对 DSP 的业务逻辑处理建立了有限时间有限状态自动机模型，DSP 中一次业务处理的交互流程可以使用一个有限状态自动机来表示。我们以 VPMN 短号短信为例，用自动机模型来展示一个完整的业务流程。具体流程如图 2 所示：

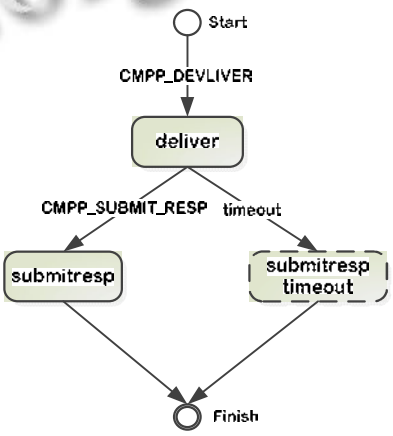

图 2 VPMN 短号短信业务流程图

1) IDP 平台接收到短信网关的上行短信 CMPP_DELIVER 消息，回复 CMPP_DELIVER_RESP 消息。

2) 对 CMPP_DELIVER 中的源号码和目的号码进行转换，目的号码从 VPMN 短号转换成真实号码，源号码从真实号码装换成 VPMN 短号。把转换后的消息通过 CMPP_SUBMIT 发送到短信网关。

3) 接收到短信网关的 CMPP_SUBMIT_RESP 消息，进入 submitresp 状态，IDP 平台处理完 CMPP_SUBMIT_RESP 消息，进入结束状态。

4) 如果等待 CMPP_SUBMIT_RESP 消息超时，即在 deliver 和 submitresp 状态之间转换超时，则进入 submitresp timeout 状态，进行异常处理后进入结束状态。

4.3 DSP 结构

IDP3.0 中使用业务逻辑模块 DSP 进行业务逻辑处理。业务逻辑模块 DSP 以有限时间有限状态自动机为

模型实现了通用的业务实例，利用 Lua 作为业务逻辑开发语言，进行上层业务逻辑的开发。业务逻辑模块 DSP 中的 Lua 脚本，是 IDP 系统中根据外部事件或内部事件触发导致系统由一个状态转换到另一个状态的过程的描述。

业务逻辑模块 DSP 的结构如图 3 所示。在 DSP 中，我们用二进制组件的方式嵌入了 Lua 的脚本引擎，这样 DSP 就获得了解释执行 Lua 脚本的能力。通用消息层使用了通用消息平台的代码，通用消息封装层作为通用消息层和 Lua 脚本引擎之间的中间层，也可以称为“胶水层”，是底层和 Lua 脚本引擎之间的桥梁，同时也是 Lua 脚本的宿主。

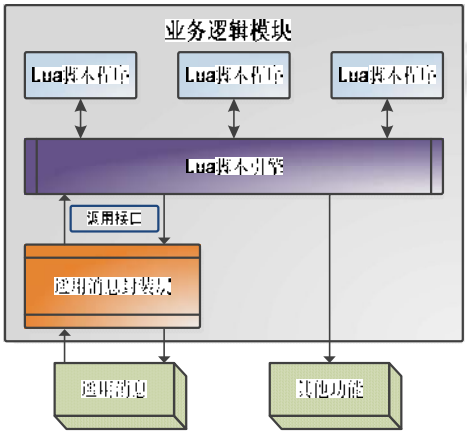

图 3　业务逻辑模块 DSP 总体结构图

4.4 DSP 接口设计

根据上面的有限时间有限状态自动机的业务模型，我们可以得知，自动机的变化由两种情况引起：1）收到外部消息触发；2）时间触发。所以我们把 DSP 的业务驱动方式也分成两种：消息驱动方式，时间驱动方式。

4.4.1 消息驱动方式的接口设计

1) LuaMsgUser 类，通用消息层的消息到上层逻辑的入口。

主要成员变量：

map<int, TMsgEntry> entrys; 各个协议对应的入口信息，通过读入配置文件获取。Key 表示协议号，value 表示入口信息。

主要成员函数：

bool compatible(TMsgItem& msg); 根据配置文件中配置的协议编号，判断参数 msg 中的协议是否能够被业务层处理。

bool onMsg(TMsgItem& msg); 对从通用消息层传来的 msg 进行处理。当消息驱动接口从其他进程收到消息时，首先判断消息类型，如果消息为会话开始消息，并且目的会话 ID 值为 0，则 DSP 创建一个新的自动机，并调用这个自动机来处理这次会话请求。如果消息不是会话开始消息，则查询现有自动机集合中会话 ID 和此消息中目的会话 ID 相等的自动机，如果查找成功，则调用该自动机对消息进行处理，如果查找不成功，则根据该消息的协议类型，调用该协议的缺省入口函数。

2) TFSM，有限时间有限状态自动机类

主要成员变量：

unsigned int fsmid; 自动机 id。

vector<TPeerAddress> sessions; 自动机中的会话，TPeerAddress 保存会话对端的通用消息会话以及会话状态等。

map<string, string> context; 自动机实例变量，以 key-value 的形式保存。

string fsmstate; 自动机下一个状态，也就是下一条消息对应的入口函数

主要成员函数：

bool setstate(string state, int timeout); 设置自动机下一个状态，以及状态转换最长时间（超过该时间，则自动机转换到下一个状态的超时状态）。

bool settimertask(string timerid, int period, int times); 设置自动机定时任务。

bool deltimertask(string timerid); 删除自动机定时任务。

bool settimer(string timerid, int period, int times); 设置通用定时器。

bool deltimer(string timerid); 取消通用定时器。

bool sendmessage(int sessionid, int msgtype, const char* message); 发送消息。

4.4.2 时间驱动方式的接口设计

TLuaTimerQueue 类：定时调用 Lua 函数的定时器队列。它在通用消息中设置了一个周期 1 秒的定时器，通用消息底层每秒调用一次 TLuaTimerQueue 的 onTimer 函数。

主要成员变量：

map<int, LuaTimer*>* luatimerqueue; 整型 key 的

定时器队列。

map<string, LuaTimer*>* luastimerqueue; 字符串 key 的定时器队列。

主要成员函数：

int addTimer(string filename, int interval, int times); 设置一个定时器。

bool delTimer(int id); 删除一个定时器。

virtual bool onTimer(); 提供给通用消息层的定时回调函数。遍历所有的定时器，给定时器流逝时间加 1，如果定时器超时，则执行对应的定时功能。

5 IDP3.0与IDP2.0测试对比

5.1 代码部署方式比较

基于 JBoss 的 IDP2.0 系统部署业务逻辑代码的步骤为：

1) 编写业务处理的 EJB3.0 代码。

2) 使用 IDE 对代码进行编译和打包。

3) 将打包后的代码上传至服务器上 JBoss 的 deploy 目录中。

4) 如需要对业务代码进行修改，则需重复步骤 1 至步骤 3 的内容。

IDP3.0 系统部署业务逻辑代码的步骤为：

1) 编写业务处理的 Lua 脚本（无需编译）。

2) 将 Lua 脚本上传至服务器，并使用 DSP 加载。

3) 如需要对业务代码进行修改，可以直接在服务器上使用文本编辑器编辑 Lua 脚本。

由以上对比可以得知，IDP3.0 系统在代码部署和修改的灵活性上都有了很大的提高。

5.2 性能比较

5.2.1 测试操作

IDP 系统处理 CMPP 消息较频繁，消息处理速率是 IDP 系统的主要关注点之一，这里选择使用一段简单的处理 CMPP 消息的业务代码测试 IDP2.0 和 IDP3.0 系统的性能。

5.2.2 测试条件

操作系统：Red Hat Enterprise Linux 5

CPU：Intel(R) Xeon(R) CPU E5504 2.00GHz

内存：4GB

数据库：MySQL5.1.55

测试模拟环境：使用 100 个线程，每个线程每秒产生 10 个 CMPP 消息，并送入消息总线 DBUS 中，测试时间 10 分钟。

5.2.3 测试结果

由表 1 中的对比测试结果可见，IDP3.0 系统在处理 CMPP 消息速率上比 IDP2.0 系统有所提升，CPU 占用率相差不多，在内存占用上有显著的下降。因此可以得知，IDP3.0 比 IDP2.0 在系统资源占用和性能上都得到了优化。

表 1 IDP 系统对比测试结果

系统	平均每秒处理消息数	CPU 占用率	内存占用
IDP3.0	251	80%	542MB
IDP2.0	183	78%	873MB

6 总结

本文总结了 IDP2.0 系统在实际应用中遇到的一些问题，并针对这些问题，提出了使用脚本驱动的开发模式构建轻量级业务框架的解决方案。该方案可以减低 IDP 系统模块间耦合度，最大程度的实现功能模块的复用，降低业务开发的门槛。并引入了有限时间有限状态自动机的业务模型，介绍了业务逻辑模块 DSP 的结构和接口设计。在性能方面，使用运行性能较高的 C++来实现基础模块，以 Lua 作为上层语言编写业务流程逻辑，来保证系统整体上拥有较高效的性能。

参考文献

1 赵贝尔.综合数据业务平台（IDP）的设计及核心功能实现.北京:北京邮电大学,2007.

2 Qi Q, Liao JX, Zhu XM, Wang C. An Adaptive Model of Task Scheduling Based on QoS Feedback Control. 3rd International Conference on Communications and Networking in China, ChinaCom 2008. Piscataway: Inst of Elec and Elec Eng Computer Society, 2008.1166–1170.

3 斯塔克.罗时飞译.JBOSS 管理与开发核心技术.第 4 版.北京:电子工业出版社,2004.30–84.

4 张锦盛.基于脚本驱动的应用系统开发方法.昆明:云南大学,2010.

5 戴俊,朱晓民.基于 ActiveMQ 的异步消息总线的设计与实现.计算机系统应用,2010,19(8):254–257.

6 贾欢欢,王纯.一种业务无关的通用协议.计算机系统应用,2011,20(2):173–177.

7 王柏,杨娟.形式语言与自动机.北京:北京邮电大学出版社,2003.30–56.