第 38 卷 第 4 期
Vol.38 No.4

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)04—0079—03　　文献标识码：A　　中图分类号：TP393

一种基于自解析报文协议的系统分层方法

罗 毅[a]，吴产乐[a,b,c]，熊伟成[a]

(武汉大学 a. 计算机学院；b. 软件工程国家重点实验室；c. 国家多媒体软件工程技术研究中心，武汉 430072)

摘 要：将自描述的结构体信息与数据信息分离，定义一种自解析报文协议 HML。该协议通过限制结构体信息的层数，在保持自描述特性的同时，提高报文解析效率，解决高并发的大规模处理系统层次之间通信问题。通过与 XML、ISO8583 等业界通用协议的解析性能对比，证明该协议的高效性。

关键词：系统分层；松耦合；自解析；结构体；数据区

System Hierarchy Method Based on Self-resolving Message Protocol

LUO Yi[a], WU Chan-le[a,b,c], XIONG Wei-cheng[a]

(a. School of Computer; b. State Key Laboratory of Software Engineering; c. National Engineering Research Center for Multimedia Software, Wuhan University, Wuhan 430072, China)

【Abstract】This paper defines a new self-resolving message protocol called HML by departing self-description area and data area. This protocol has limitations of the layers of description area. It can raise the efficiency of resolving and solve the problems between the lays of large scale processing system. The feature is proved to be true compared with XML and ISO8583, and the protocol can be used in communication between layers in high concurrent process system.

【Key words】system hierarchy; loose coupling; self-resolving; structure; data area

DOI: 10.3969/j.issn.1000-3428.2012.04.026

1 概述

现代软件系统的设计通常较复杂，在投入大量人力和资源后，其软件系统的开发周期和质量往往也不理想。按照软件体系结构研究进展和 CBSD(Component Based Software Development)开发方法[1]，可以通过提供领域开发人员参考的架构设计模型，同时通过该模型实现系统的层次化，降低整体系统设计的复杂性。为此，本文提出一种基于自解析报文协议的系统分层方法。

2 单一系统分层的效率和灵活性

在系统分层后，层与层之间都是相对独立的实体，每层都具有相关独立的处理逻辑和封装完好的数据对象。层间的通信语义表达是首先要解决的问题，结合业界的实践，XML 是解决跨层语义表达的第一候选。然而 XML 由于本身是一种自定义的树形结构标记语言，对 XML 的传递和解析将面对必须解决的效率问题[2]，尽管近年来在 XML 的传递和解析上有很多研究[3-4]，但对于高并发的大规模处理系统而言，这些方法所带来的开销仍然是一笔不小的负担。

在单一系统的分层设计中，由于各层都运行在同一系统环境中，跨平台的语义表达并不迫切，相反地，由于这些应用对客户服务的高并发和快速响应的需求，对层间的通信效率提出了很高要求。传统的进程间通信采取预先定义好的固定长度的通信区，通信区中每个字段及其类型都是预先定义好的，一旦发生变动，调用者和被调用者都必须同步修改，否则容易带来由于通信区不匹配而引起的系统调用失败。这种方式和 XML 相比效率比较高，但不够灵活，难以适应层与层之间高内聚、低耦合的特点。

如何在效率与灵活性方面求得一定的平衡，业界一直在寻找可行的解决方案，在金融领域被广泛使用的 ISO8583[5] 及其改进就是其中一个例子。ISO8583 将整个通信区分为位元区和数据区，位元区由 128 个 BIT 的标识组成，第 N 位标识为 1 即表示后面的数据区中存在第 N 个数据字段。数据区用来定义整个数据字段，每个数据字段都包括了该数据的自描述属性和对应的值。自描述的属性包括数据字段名称、数据类型和数据长度。ISO8583 协议实现了通信数据的自描述，同时具备较高的解析效率，因而在金融领域得到广泛应用。

而对系统的分层而言，单一层次线性扩展的 ISO8583 协议格式并不能满足复杂通信区数据的传输，如结构和数组。同时位元区的定义有 128 个字段的限制，只能使用在某一相对局限的业务领域，不适合具有通用性质的系统分层通信。

3 一种自解析报文协议

大规模处理的单一系统中，进程间的通信往往并非平面的字段并列，而是将某一类字段组合起来形成逻辑上有意义的结构体，例如功能逻辑输入结构体、处理调用返回结果结构体等；在有些查询功能调用时还会在通信区中用到结构体数组。本文提出的自解析报文协议延续了将描述区和数据区分开的思想，但在描述区的设计上考虑了对结构体和数组的实现，将 XML 的多层树状描述简化为两层的自描述区。将这种层次化的自解析协议称为分层标记语言(Hierarchical Marked Language, HML)。

3.1 自解析协议的层次结构

HML 的报文结构如图 1 所示。整个报文分为 4 个区：报头，结构体描述区，字段描述区和数据区。

作者简介：罗 毅(1975－)，男，博士研究生，主研方向：网格计算，数据库技术；吴产乐，教授、博士生导师；熊伟成，博士研究生

收稿日期：2011-08-10　　**E-mail：**luoyi9999@hotmail.com

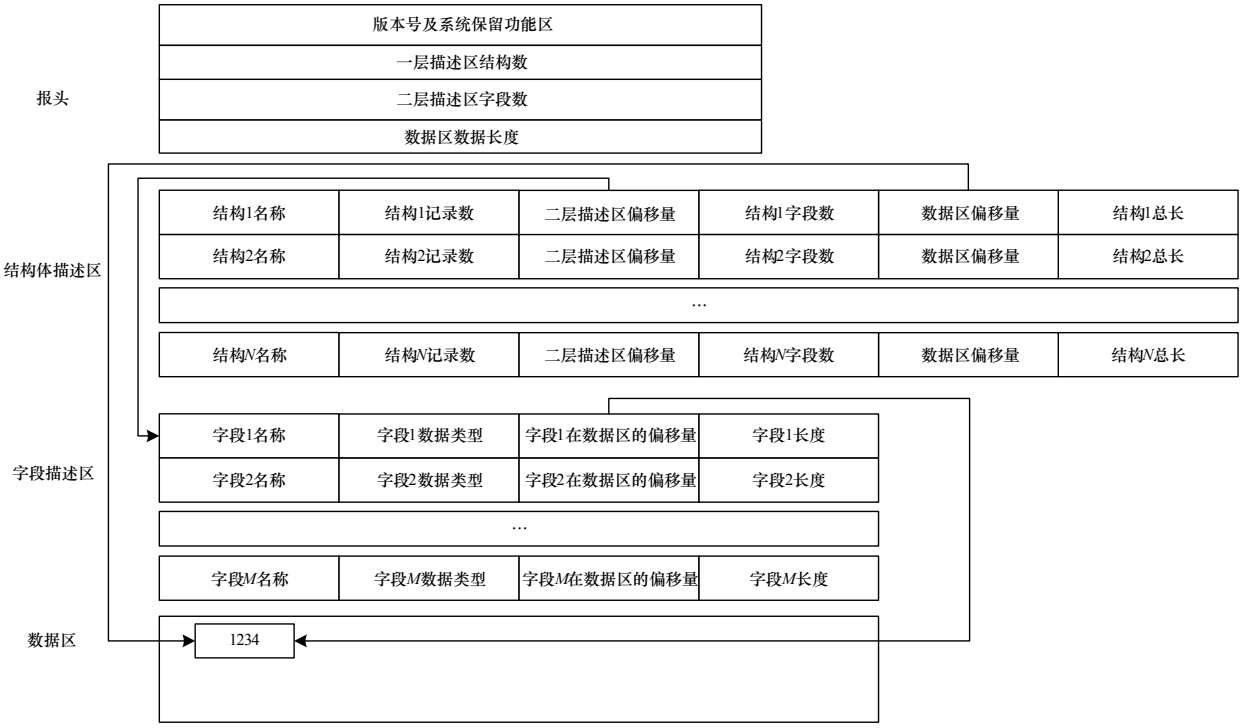

图 1 HML 报文协议结构

报头是对报文的整体解释，版本号及系统保留功能区字段提供了对报文协议版本的解释，并为某些与系统调用相关的特殊数据提供保留。报头中还提供了对结构体描述区结构体个数、字段描述区字段个数以及数据区总长度的描述，用于对整体报文合法性的校验。

结构体描述区实际是对通信区中常见的结构体的描述，每个结构体都提供了结构体名称、结构体记录数(如果是结构体数组则是数组下标，否则都是 1)、结构体包括的连续字段集合在字段描述区中的偏移位置、结构体包括的字段个数、结构体包括的连续字段数值在数据区中偏移位置、结构体在数据区中总长度等 6 个属性信息。其中结构体字段和字段数值在字段层描述区和数据区中的偏移位置为支持快速的结构字段查找提供了指针。

字段描述区提供的是结构体下所有字段的描述信息。每个字段的描述信息包括字段名称、字段数据类型、字段在数据区的偏移量、字段长度等信息。其中字段的类型延续了 ISO8583 中对字段类型的定义。可以支持常见的数据类型及其组合。字段在数据区的偏移位置为从字段描述区查找数据区中的字段数值提供了指针。

数据区则是连续的所有字段数值，它通过前两层描述区来进行定位和查找。

总体来说，查找 A 结构体中 B 字段的数值通过如下过程实现，首先略过固定长度的报头，在结构体描述区中查找到 A 名称的结构体描述，从而得到 A 结构体在所有字段描述在字段描述区的位置。根据该位置查找字段描述区，查找 B 字段的字段描述，从而得到该字段在数据区中的位置。根据该位置查找数据区，从而取得该字段的数值。可以看出，最多通过两层查找即可准确找到某字段的描述和数值信息，该机制是实现 HML 高效解析的重要方法。

为了防止 HML 过长导致通信传输的额外开销，结构体描述区最多可描述 300 个结构体，字段描述区最多可描述 900 个字段，报文总长不超过 32×10^3。

3.2 自解析协议的使用方法

由于系统分层前进程间的调用都是使用传统的通信区模式，应用程序的开发也是基于传统的通信区结构定义方法，因此在使用 HML 报文协议对系统进行分层后，上层程序在调用下层程序前首先使用 PACK 函数对原有传统的结构通信区进行打包，生成 HML 的报文格式。PACK 函数打包时，必须知道原由通信区结构的描述信息，并以结构体描述区、字段描述区的结构组织起来。因此，在 PACK 函数前首先提供 PARSE 函数扫描已由的静态通信区结构，生成结构体和字段描述区。

HML 报文生成后传递给下层的被调用程序，下层程序使用 UNPACK 函数对 HML 报文解包。下层程序需要告诉 UNPACK 函数希望解析出的结构体和字段名称，考虑到已经使用的通信区结构，也可以使用 PARSE 函数分析已有的通信区，生产结构体和字段描述区，传递给 UNPACK 函数。

通过打包和解包函数的使用，上层调用者可以将 A 结构的通信区打包成 HML，而下层被调用者可以以 A 结构的子集 B 结构来解析 HML。这样通信区结构变动时不会影响被调用的程序，从而实现层次之间的松藕合。

显而易见，HML 的结构可以通过 XML 来实现，当系统需要和异构系统进行通信时，可以使用 HML2XML 函数将 HML 报文转换为 XML 文本，同样接收到异构系统传递的 XML 文本时，也可调用 XML2HML 函数将其转化为 HML 报文。

4 自解析协议在分层系统中的应用

需要分层的单一系统往往具有非常复杂的应用逻辑，需要和若干个异构系统进行通信。一般来说，可以将单一系统中与外部系统进行通信或屏蔽某种外部属性的部分抽象出来，形成独立的接口处理层。还可以将单一系统中所有处理都需要使用的功能抽象为核心处理层，作为整个系统运行的基础和核心部分。剩下的若干功能可以按照不同的功能特性进行分组，形成不同的功能处理模块，由此形成接口处理层、

功能处理层和核心处理层相对分离的三层结构，如图 2 所示。

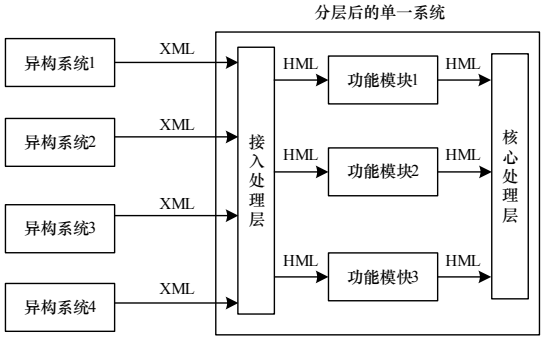

图 2 HML 在系统分层后的应用

层间使用 HML 进行通信。由于 HML 结构相对简单，最多只能支持两层结构定义，因此能在打包和解包时保持较高效率。接口处理层专门处理与外部异构系统的连接与通信，通过调用 XML2HML、HML2XML 实现外部通用、更灵活的 XML 和系统内部使用、更高效的 HML 之间的转换。

分层后的系统每层功能相对简单，能集中与本层应处理的事务，从而提高了内聚度。同时分层带来的开销，特别是 HML 的解析开销相对可控，没有带来系统性能的大规模下降，因而具有很好的使用前景。

5 协议的具体实现与性能测试

为了验证 HML 的可行性和有效性，基于大规模事务处理的大型机环境来实现 HML、ISO8583 和 XML，操作系统为 MVS6.1，事务处理中间件为 CICS7.0，使用 COBOL 和汇编程序分别实现 3 种通信协议。图 3 给出了 XML、ISO8583 和 HML 在该环境下实现的性能测试情况，包括整体程序的响应时间和 CPU 的开销。

程序资源消耗/μs		单一程序运行	10 进程并发	20 进程并发
XML	GETSTR1 响应时间/CPU 时间	131/20	130.7/7.2	1 595/12
	GETSTR2 响应时间/CPU 时间	192/17	259.4/10.1	2 699/32
	GETSTR3 响应时间/CPU 时间	178/16	2 902/68.9	4 990/22
ISO 8583	PACK 响应时间/CPU 时间	2.75/1.54	7.01/1.65	14.55/1.77
	UNPACK 响应时间/CPU 时间	2.60/1.61	6.58/1.58	12.36/2.01
HML	PACK 响应时间	2.82	7.31	14.78
	PACK CPU 消耗	2.37	2.57	2.56
	UNPACK 响应时间	2.67	7.15	14.72

图 3 XML、ISO8583 和 HML 的性能测试结果

为评价在高并发情况下的性能表现，分别安排 10 进程并发和 20 进程并发的测试。对于 XML，分别测试了树型结构下，1 层解析、2 层解析、3 层解析下的性能情况，即 GETSTR1、GETSTR2、GETSTR3。对 ISO8583 和 HML 分别测试了打包和解包情况下的性能表现。

从表 1 的性能分析对比可以看出，采用 XML 方式的解析程序随着解析的层数越深，对 CPU 的响应时间呈逐步增长趋势，而且随着并发量的加大，整体响应时间和 CPU 消耗都呈非线性增长趋势。ISO8583 的打包和解包性能相对开销最小，且随并发量的增长其开销也线性增长。HML 的性能表现仅次于 ISO8583，远优于 XML，其 CPU 消耗基本稳定，程序响应时间随并发量的增加而线性增长。

以系统中并发量适中的某常用功能计算，其使用到约 350 个字段通信区传递。在该功能使用 XML、ISO8585 和 HML 后，其 CPU 时间消耗估算分别增长 24%、2.5%、6%。考虑到 HML 为系统分层所带来的便利性支持，其相对而言的较小的系统开销，证明了其具备很好的实用性。

6 结束语

本文从系统分层的性能问题入手，分析 XML、ISO8583、传统进程调用等层次间通信传递的方法，并提出一种能适应结构体和数组传递、具备更好通用性的自解析报文协议 HML。通过该协议，对复杂的单一系统可以按其处理的不同功能分成不同的层次，层与层之间实现松散藕合。针对具体的行业应用，可以基于该协议探究适宜的系统分层模型。在性能分析方面，虽然测试显示 HML 的高性能，但如何提高在系统高并发情况下的调优，与应用程序结合如何最大限度降低系统的额外开销仍然是需要探讨的问题。在 XML 和 HML 的转换方面，如何寻求高效的转换方法也需要进一步研究。

参考文献

[1] 梅 宏, 申峻嵘. 软件体系结构研究进展[J]. 软件学报, 2006, 17(6): 1257-1275.

[2] 刘 芳, 肖铁军. XML 应用的基石: XML 解析技术[J]. 计算机工程与设计, 2005, 26(10): 2823-2824.

[3] 冯 进, 丁 博, 史殿习, 等. XML 解析技术研究[J]. 计算机工程与科学, 2009, 31(2): 120-124.

[4] 曹小冲, 胡运发, 陶晓鹏, 等. XML 数据的数值对象化及其转化算法[J]. 计算机工程, 2010, 36(6): 45-48.

[5] International Organization for Standardization. ISO 8583 Financial Transaction Card Originated Messages[S]. 1993.

编辑 陈 文

(上接第 78 页)

参考文献

[1] Studer R, Benjamins V R, Fensel D. Knowledge Engineering: Principles and Methods[J]. Data and Knowledge Engineering, 1998, 25(1): 161-197.

[2] McGuinness D, van Harmelen F. Owl Web Ontology Language Overview[EB/OL]. [2011-06-15]. http://www.w3.org/TR/owl-feathers/.

[3] 彭 涛, 张 力. 基于本体和 XML 的数据交换研究[J]. 计算机工程, 2006, 32(1): 90-92.

[4] 乔 卫. 基于领域本体的 XML 语义信息抽取的研究与实现[D]. 武汉: 武汉理工大学, 2009.

[5] 刘造新. 基于本体的 XML 关联规则挖掘方法[J]. 计算机应用, 2008, 28(9): 2318-2320.

[6] 吕 律. 一种提高本体映射精确度的方法[J]. 计算机工程, 2010, 36(7): 73-75.

编辑 顾姣健