大数据应用系统的消息驱动架构

贵芳[1]，李廉[1]，杨静[1]，武永卫[2]
1. 合肥工业大学计算机与信息学院，安徽 合肥 230009；2. 清华大学计算机科学与技术系，北京 100084

摘要
基于消息驱动框架的软件开发成为大数据应用系统的重要模式之一。基于面向实体、消息驱动的开发架构，设计并实现了该架构中的消息管理模块。定义了消息基本格式，制定了消息管理规范，并且具体实现了其功能，通过一个案例显示了该架构的优点。

关键词
大数据应用系统；消息驱动架构；消息管理模块；多用户；动态演化编程

中图分类号：TP31　　文献标识码：A　　doi: 10.11959/j.issn.2096-0271.2016040

A message driven framework for big data application system

GUI Fang[1], LI Lian[1], YANG Jing[1], WU Yongwei[2]
1. School of Computer and Information, Hefei University of Technology, Hefei 230009, China
2. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Abstract
Software development based on message driven framework has become an important pattern for big data application system. The message management module which is an important part of whole framework was designed and implemented. The basic format of message and the standard of message management was defined, its function was realized, and the advantages through a case were shown.

Key words
big data application system, message driven framework, module of message management, dynamic and self developing programming

1 引言

在互联网支持的大数据应用系统中，实体行为通常通过消息交流产生相互影响，实体行为之间是并发的、独立的、自演化的。所有实体的行为决定了系统的行为，在系统的稳定态下个别实体的变异不会影响整个系统的功能，虽然每个实体的行为具有确定性，但它们所作用的系统演化结果具有不确定性。也就是说，互联网中的所有个体构成了巨大的并发计算机，每一个个体通过与其他个体进行信息交互实现各自的计算，并且得出整个系统的运行结果[1]。如果将这种在互联网世界表现出来的行为方式运用到程序架构设计中，由个体的消息来驱动程序，降低个体变异对整个系统的影响，形成一种新的开发框架，将会是程序设计的一种新思路，可以看作在程序层面模拟自然系统的行为。比如，很多大数据应用系统由于个体行为与整个系统性能有着强关联性，其编程方法和开发模式就需要解决实体的动态性、不确定性、并发性，通过模拟自然系统的行为和演化模式，实现程序的灵活性、顽健性等问题。

在大数据应用系统中，已出现很多实际问题需要采用新的程序架构来完成特定的功能要求。比如在智慧城市中，许多用户之间共用一套消息管理系统，消息管理系统可将消息主动推送给相应的用户，而用户之间很少进行联系[2,3]。在传统的面向对象的程序设计方法中，对象之间是相互通信且彼此联系的，任何一个对象的加入或者退出都需要告知其他对象，简单来说就是每个对象内部都有一套自己的消息管理系统。但是在大数据应用中，多元数据的融合却成为重要的问题，这就给程序设计带来新的问题，即动态融合的数据如何更好地应用于系统，以实现“融合、跨界、基础、突破”[4]。

针对这类大数据应用，一种新的程序开发架构正在形成，这就是面向实体、基于消息驱动[5]的开发架构——消息驱动架构（massage driven framework，MDF）。MDF在结构上没有传统的主程序，而是分为3个模块，即实体模块、消息模块和数据及显示模块。实体类似于对象，封装了数据和操作，形成独立的运行单元。但是实体之间没有直接联系，所有的实体都是基于消息控制的。MDF提供了一种在网络环境下，共同开发系统的机制，遵守事先规定好的规范，开发人员不需要有沟通联系，根据规范自行开发应用模块（实体），然后加入系统中运行，并且还可以随时退出，这些操作都无需与系统其他用户和管理人员打招呼。所有这些实体（用户）行为决定了系统的行为，个别实体的消亡或变化不影响整个系统的功能。MDF是一种在互联网环境下的编程框架，支持众多实体参与的共同开发模式，它建立一种直接模拟自然和社会系统复杂行为运行的编程方法，排除个别变异实体对整个系统功能的影响；探索一种新的应对大数据问题高并发、大流量、用户独立的解决方案，满足新的应用需求。

消息管理是MDF重要的组成部分，可以说是整个系统的中枢，所有实体的运行要靠消息管理进行协同，因此消息管理模块的设计与开发是整个系统的关键内容。这里面主要有三大挑战：一是在MDF中，所有的通信方式都被严格定义为消息，不同类型的消息有着不同的处理方式，消息管理器如何在消息数量庞大的情况下识别消息类型，尽可能提高消息的处理准确率；二是在消息维护过程中，如何通过调度策略，以优化各种不同类型消息的处理

和管理，使得在消息交互过程中尽量减少资源开销和成本；三是在消息接收和发送过程中，尤其在消息数量庞大的情况下，消息必然出现排队情况，如何设计合理算法，保证不同优先级的消息能够及时得到发送，又避免出现消息长期等待不被处理的情况。

本文中指的消息，相对于已有的事件驱动（event-driven）中的事件有所不同，事件的定义更加广泛，可以是程序本身发出的信息，也可以是终端设备发出的请求。事件的管理和处理需要有一套复杂的硬软件设备来完成。消息只是用户程序（终端应用程序）发出的一些文字序列，因此比起事件来说，在类型和内容上要简单的多。理论上说，事件驱动的技术可以用来产生消息驱动程序，但是不如另外为消息驱动重新定制一种架构，在应用系统开发中会更加有效和简单。

消息管理模块在MDF中起到了中心的作用。本文设计并实现了MDF的消息管理模块，包括以下3个方面：

- 定义了消息的基本格式，制定了消息池的语言规范，通过消息表单的配置，实现不同类型消息的区别管理；
- 利用现有的编程语言Java开发了消息维护中间件，根据消息规范的表单配置自动生成消息类型管理，实现消息的接收、维护、存储和发送等核心功能；
- 开发了中间件，接受实体对于消息管理模块的操作，包括查询、检索以及对于各种类型消息的定义变更，满足了MDF中实体运行和变化的需要。

在MDF中，实体的数量是可变的，实体的类型是多样的。所有实体发送的消息和接收的消息将遵循统一的格式，实体需要向系统声明的是它的URL地址、消息类型、消息发送要求等。URL地址提供了该实体消息发送和接收的接口。消息类型和消息发送要求提供了消息的处理方式，这种要求被消息管理模块响应，并被正确维护。一个系统中的实体类型是多种的，每一种实体类型被定义了消息格式，同一类型的实体必须按照相同的消息格式发送消息，系统的消息管理模块在逻辑上是唯一的。所以消息管理模块的设计为实现MDF的并发性、动态性、顽健性、不确定性和跨平台性等重要特性奠定了基础。

2 相关工作

随着计算机经验和软件技术的发展，计算机编程语言经历了机器语言、汇编语言、面向过程的程序设计语言以及面向对象的程序设计语言阶段[6]。具体的语言不胜枚举，见**表1**。

随着编程语言的发展，计算机程序设计的方法也主要经历了3个阶段的发展：面向机器的程序设计、面向过程的程序设计和面向对象的程序设计[7]。

人类和计算机进行交流最开始的语

表1 编程语言的发展

年份	语言	年份	语言
1957-1959年	Fortran、USP、COBOL	1991年	Python
1970年	Pascal	1993年	Ruby
1972年	C	1995年	Java
1983年	C++	1995年	PHP
1983年	Objective-C	1995年	JavaScript

言是由计算机可以直接识别的二进制指令组成的机器语言，在这个时期，计算机的程序架构就是直接描述机器操作，因此这时的程序称为面向机器的程序架构。随着高级程序语言的出现，面向过程的程序架构被提出，这种结构化的程序设计采用了“自顶向下，逐步求精”[8]的思想，将计算问题模块分化，功能分解，大的问题化解为若干个小问题，大大提高了工作效率，也利于程序的维护。当然，面向过程的设计方法也存在一定缺点，最主要的是该方法编写的程序是一系列的线性步骤，这种编程方式必须按照线性步骤从头到尾编写代码，过程枯燥且不易修改，代码的灵活性和可移植性都较差。为了克服这一困难，面向对象的程序架构应运而生，其主要思想是“注重对象，抽象成类”，不再强调过程，更侧重于对对象的操作。对象是数据和操作的封装体，与客观实体直接对应。通过封装使对象变得独立，只能通过预先设定的方法与对象交谈，在构建代码的过程中通过继承，减少冗余代码的同时又可扩展现有代码；封装可减少外界对程序的干扰；且面向对象的系统很容易切分成很多独立的部分，将系统化繁为简，同时这种方法可以使系统从小到大逐步升级[9]。但传统的面向对象架构仍存在不足，对象之间需要建立自己的通信通道，所有对象都需要了解一定的环境参数。但是在互联网环境中某些由众多用户动态参与的应用系统中，例如购物、游戏、社区等，它们的特点是用户众多且不确定，并且在大多数情况下，用户不需要了解过多的运行要求和参数，而且用户之间是透明的，也就是用户之间可能不知道对方的存在，因此也不能有直接的沟通，对于这类程序的开发就需要提供一种更为自由和开放的编程模式。从目前主要是个人和小团队开发的模式，逐步过渡到可以由彼此互不认识的众多用户共同开发的模式，让众多的用户包括非计算机专业的用户也可以参与到系统的开发和运行中，使得每一个用户既是程序的应用者也是程序的开发者，将更加能够体现大数据应用的特点[10]。

本文提出的MDF的程序设计方法的主要思想是“实体独立，消息驱动”，各个实体之间遵循相同的消息通信协议，参与设计和开发的人员只需输入框架指定的几个参数就可参与开发，因此开发人员不需要相互沟通，大家只需要根据协议和规范就可以进行各自的应用开发，最终组成一个庞大的应用系统，让更多的用户根据个性化的要求方便地使用。

3 MDF架构与消息管理模块

3.1 MDF架构

MDF没有传统意义上的主体程序，而是由实体、消息和数据显示3个部分组成。总体来说，MDF是模拟自然和社会系统的运行模式，MDF所有运行都是通过消息驱动的。任何用户都可以成为系统的组成部分，称之为实体，或者系统对象。程序开发者只发布系统规范，系统规范指明实体的类型[11]、实体的动作定义、实体的消息构成、实体消息的内容格式与解释以及实体的登录方法、实体生命周期等。一般地，规范提供实体样式。系统规范还会提供实体与数据显示部分的数据交换方式、运行结果显示的方式以及相应的显示控制软件，由数据显示部分具体执行。规范是开放的，用户可以根据系统规范开发实体，或者实体样式填好参数进行注册，进入系统运行；也可以根据规范自行开发显示模块，获取个性化的显示方式。实体的地位是平等的，实体的加入和退出完全是自由

的，虽然会影响系统的运行结果，但是一般不会影响系统的功能。消息部分是实体之间进行信息交互的中介，实体之间并不直接通信，而是借助消息部分进行转发[12]。实体的动作由消息定义，实体通过发送和接收消息来实行具体动作。实体的功能以及如何由消息定义动作在系统规范中予以说明。数据显示部分提供运行结果和过程的显示。MDF通过3个模块实现这些功能，分别是实体管理模块、消息管理模块和由数据库和输出控制组成的数据显示模块[13]。其具体架构如**图1**所示。

- 实体管理模块：是整个软件开发的主要部分。开发者编制、发布实体规范，规范提供实体描述定义、实体动作定义以及消息内容格式与解释，用户根据规范自行开发实体程序，或者根据开发者提供的实体样板注册登录。实体管理模块根据实体规范协议管理所有的实体，包括实体的登录和退出以及生存期间的运行管理。
- 消息管理模块：管理实体之间用于交互的消息。消息管理模块发布一个消息格式协议[14]，其中定义了消息格式的内容与解释。消息管理模块根据消息格式协议管理消息的接收、发行、维护以及存储等。
- 数据显示模块：提供系统运行数据的显示，包括结果显示和过程显示，数据显示模块与实体进行通信，接收实体提供的数据，通过输出控制器执行数据显示的功能。同时该模块还提供公共性数据和永久性数据的存储和查询，这些功能同样通过与实体消息交互来驱动和实现。

本文主要介绍消息管理模块的设计原理和实现，其他两个模块将另文撰写。

3.2 消息格式协议

消息管理模块最重要的是消息格式协议（message format propotol，MFP）。MFP定义消息为一个文本信息，分为4种类型，分别为：发送消息、查询消息、测试消息和名单消息，所有消息都用表示消息类型的代码开头，后面跟随消息本体。其中，A表示发送消息，即实体之间传递的消息；B表示查询消息，即实体发出的查询某个消息的请求；C表示测试消息，即实体用于测试通信链路状态的消息；D表示名单消息，即实体发出的名单，这个名单提供当前实体的变动，或者在多播状态下提供接收实体的列表。

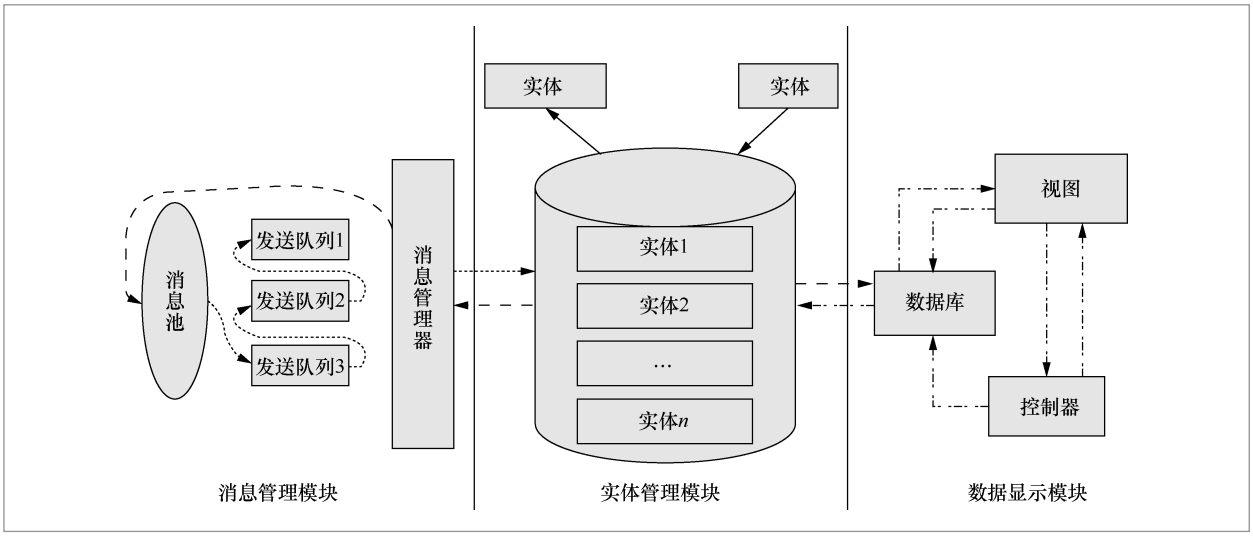


图1 MDF系统架构

MFP定义消息head部分有7个属性（见**表2**）。各部分内容如下。

● name：符号串类型，表示消息的名称。字长限制为128 byte。

● from：URL类型+时间类型，消息发送方的URL地址，后跟发送时间，中间用“+”号隔开。

● to：消息接收方地址，使用“X+P”形式表示，其中，X表示发送形式，分别是U（单播）、M（多播）和B（广播）。P表示发送对象，在单播时为URL地址，多播时为接收方的URL地址，最多为16个地址，或者是分组文件名称，这个分组文件事先存放在消息管理模块中。广播时，P为空，此时由消息管理模块根据内部存放的实体名单向各实体发送消息。X和P之间用“+”隔开。

● type：消息发送类型，使用单个字符表示，其中A表示主动发送，即该消息根据设定的时间自动发送到接收方；P表示消息只在接收方查询时发送。

● life：消息生存时间，使用“单个字符+时间类型”形式表示，单个字符表示消息销毁时间，后面是具体的时间，中间用“+”隔开。其中，单个字符L表示该消息发送后即销毁；L+time表示该消息在time这个时间点销毁；K+time表示消息在保存time时间后销毁。

● time：时间类型，表示消息发送的时间，这个参数只在主动发送时有效，在查询发送的消息中，该栏目可以为空。

● priority：整数类型，取值为1~3，表示该消息的优先等级，消息的优先等级由系统规范定义。

表2列出了发送消息name、from、to、type、life、time、priority的全部7种属性。消息管理模块会针对消息的属性描述进行不同的处理。

发送消息（以A开头）的消息内容由head和body两段组成。head定义了消息的声明部分，用以描述消息的属性；body定义了消息的内容部分，用以描述消息的正文。MFP仅对消息的head部分进行了格式要求，其属性值对消息的传输、保留、销毁、发送以及与实体之间的交互起着重要的作用。MFP对body未做要求，只是限定了body的长度。body的内容语义解释由程序开发者定义，所有实体必须按照这个定义来编排消息内容。在MFP和消息管理模块中，不涉及对于消息内容的任何读写，只是根据消息head部分的要求进行消息管理，而消息body部分的内容理解是由实体完成的。

查询消息（以B开头）由消息名称（name）（必有）、查询方的URL地址（to）以及查询内容（body）组成。查询内容是消息的属性，该模块支持属性匹配查询，消息管理模块在匹配检查完成后，将符合条件的消息根据URL地址发送给查询方。查询

表2 消息的基本格式

字段		类型	说明
head	name	符号串	消息名称，长度不超过128 byte
	from	URL+timedid	发送方地址和发送时间
	to	X +P	发送形式与接收方URL地址
	type	发送类型	主动发送或被动发送
	life	字符+时间类型	消息的生存时间
	time	时间类型	消息的发送时间
	priority	整数类型	优先级
body	text	文本类型	消息内容，长度限制为1 024 byte

内容部分可以是空，表示这部分不用匹配。在本文的版本中，暂不支持模糊查询和通配符查询，也暂不支持多条件查询。

测试消息（以C开头）是由发送方的URL地址加上任意的64 byte的文本，中间用“+”隔开。消息管理模块在收到该消息后，向发送方返回该64 byte的文本。测试消息是为了检查当前通信线路是否可用，当实体发送测试消息并正确回收发送的文本后，就可以执行正常的消息传送了。

名单消息（以D开头）有两种格式：名单定义与名单删除。名单定义用于表示实体定义的多播名单，一般情况下，实体可以随时发送多播名单，多播名单有多个，编号加以区别，该多播名单声明实体名称和多播对象URL地址。当实体提交一个多播消息时，应在属性to中指明多播名单的编号，以便于消息管理模块生成发送消息对象。名单删除表示需要删除的名单，所有名单通过名单名称进行识别。

3.3 消息管理模块

消息由实体产生，并以消息格式协议规定的形式由消息管理模块处理，其中消息名称name是消息的唯一标识。

每一个实体在登录系统时，会向消息管理模块发送一个身份登录信息，从而在消息管理模块中产生一个动态的实体名单，该名单维护当前登录实体名称。同时每一个实体需要发送多播消息时，会向消息管理模块提交多播名单编号。如果未提交多播名单，协议允许发送方在消息属性to部分临时指定不超过16个发送对象的URL地址。实体名单和多播名单都由消息管理模块维护。

图2演示了消息从接收到发送的完整流程[15]。实体1发送控制指令与通信线程1建立消息传输通道1，发送消息msg2。消息管理模块接收到msg2，经过合法性和完整性的验证，将消息存放在消息池中。如果消息属于主动发送类型，则将该消息加入发送队列，根据指定的优先级发送消息。如果是被动发送消息，则在接收到查询消息后，将该消息加入发送消息队列，根据该消息的优先级发送到实体2。

由**图2**可以看出，消息管理模块主要分为2个部分：消息池和消息数据库。

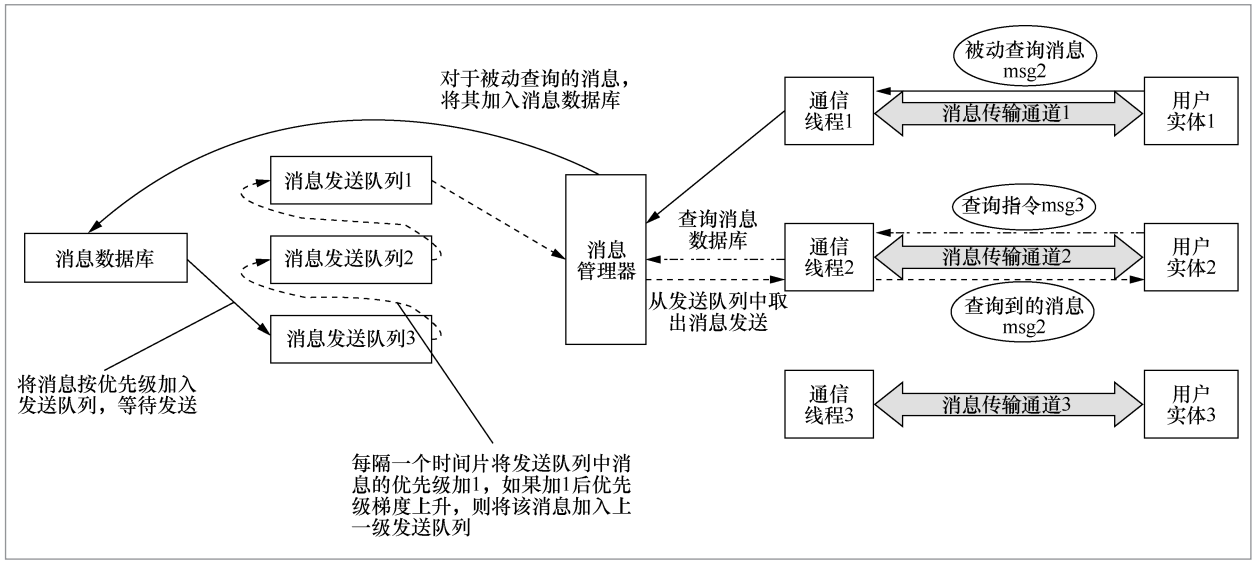


图 2 消息的流向

消息池是3个消息队列，分别对应MFP中的3个优先级。需要发送的消息根据优先级放入相应的消息队列进行排队。消息队列的管理算法在下面单独说明。

消息数据库是一个数据库，消息被接收后，经过消息管理模块的解析，如果不是即时发送，则存入相应的数据库。消息数据库随时检查消息，当某条消息达到发送条件时，即将该消息送入消息池排队。如果接收到的消息是查询消息，则进行相应的查询，并将查询的信息返回发送方。如果某条消息需要销毁，则消息数据库进行相应的操作[16]。

除了这两部分外，消息管理模块还负责消息池和消息数据库的维护，保证其正常运行和通信畅通。当实体发送测试消息时，该模块负责响应该测试消息。当实体发送名单消息时，该模块负责读取并存入相应的存储器管理，或从存储器中删除名单。

（1）消息管理

消息与实体之间的联系采取socket通信协议[17]。socket为消息管理模块和实体模块之间的通信提供了进程通信的端点，每个socket都用一个半相关的描述（协议、本地IP地址、本地端口）。系统会根据实体的URL地址和端口号为其生成一个socket号；消息管理模块向所有实体公布唯一的socket号，任何知道该socket号的实体都可以向消息管理器发出连接请求。

消息管理模块与实体之间的连接过程分为3个步骤：通信线程的监听、实体端请求和消息管理模块的连接确认。由于消息在网络中进行传输会有较大的时延，为了提高系统的实时性和吞吐量，每个实体与消息管理器之间都有一个消息传输通道。对于每一个消息传输通道，消息管理器中有一个用来收发消息的通信线程与其对应[18]。

通信线程接收到用户实体发送的消息后，首先对消息进行完整性、合法性检验，然后再对检验合格的消息进行处理。

- 如果是发送消息，则解析消息head中的各个属性，对于满足发送条件的，则根据属性中的指示，将消息加上需送达的实体URL地址以及其他一些辅助信息，根据其优先级，加入相应的发送队列等待发送。
- 如果是查询消息，根据查询消息中的查询内容在消息库中进行查询；将查询到的消息加上送达实体的URL地址以及其他一些辅助信息，并根据其优先级，加入相应的发送队列等待发送。
- 如果是测试消息，则根据协议规定，返回测试实体64 byte的文本。
- 如果是名单消息，则将名单加上发送实体的URL地址与编号作为文件名称，存入相应的存储器，以备使用。

同时，消息管理模块也支持一些控制指令，以保证系统通信的正常和完好。

（2）消息池

消息池负责管理消息的发送，主要包括多个发送队列（本文中是3个）、发送队列维护线程、消息的发送线程。消息管理模块与实体连接后，就会进行消息的传递，在消息的传送过程中，将根据优先队列进行消息发送。

消息池经常处于并发工作状态，为此在消息池中对每条消息都标记了优先级，同优先级的消息位于同一发送队列，如**图3**所示。

具体算法如下。

采用优先队列发送（priority queue sent，PQS）算法。

①读取最高优先级（priority=3）的队列消息，如果该消息信道空闲，则发送该消息；如果信道繁忙，则转向同队列的下一条消息；如果下一条消息不存在，则将下一优先级的第1个消息移入该队列，并发送该消息。

②消息发出后，等待时间间隔γ，如果收到消息发送成功信息，则在队列中删去

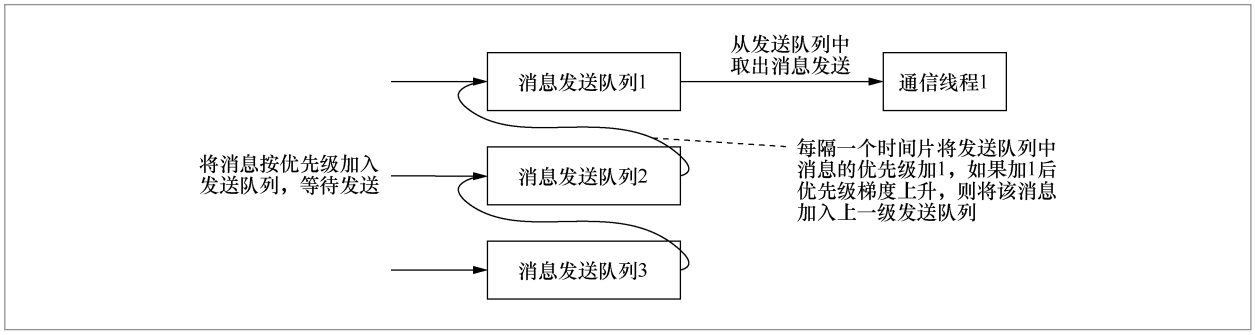


图 3 发送队列设计

此消息。将次高优先级（priority=2）和最低优先级（priority=1）的队列中的最前消息优先级加1，并排到前一优先级的队尾；返回第①步。

③如果时间间隔γ后，未收到消息发送成功信息，则再次发送该消息。

④如果再次时间间隔γ后，仍未收到消息发送成功信息，将该消息移到同一优先级的队尾，返回第①步。

该算法可以设立通信进程而支持消息的并行发送，在前一个消息发出而未收到确认信息时，后一条消息仍可以发送，而且确认信息回复的次序可以与发送次序不同，后发的消息回复可能先收到。

消息池在消息队列的内部采用FIFO（first in first out，先进先出）算法[19]；在队列间，采用动态优先权调度算法[20]，即将消息加入发送队列时，按其优先级加入相应的发送队列，消息发送总是在最高优先级队列中进行，每处理一条消息后，就将其余队列中的第1条消息的优先级加1，并按更新后的优先级将消息转至相应的发送队列队尾。梯度发送队列之间用链表实现。发送队列维护线程用于动态（按时间片）更新消息的优先级。消息的发送线程用于从优先级最高的发送队列中取出消息，并根据消息head中的to字段调用相应的通信线程发送消息。由于系统中有多个消息传输通道，等待一个消息被接收后再发送另一个消息会导致传输通道的空闲时间过长，为了提高系统的吞吐量，本文采用了多线程的消息发送模式[21]，只要待发消息的信道空闲，就发送该消息，这样的方法可以提高信道的利用率。

（3）消息库

消息库是一个关系数据库，本文使用的是MySQL。消息库存放所有未被删除的消息以及名单消息中提供的名单，并实现以下的操作。

- 接收消息后，经过解析，如果需要立即发送，则按照优先级送到相应的发送队列排队，并存入信息库；若不需要发送，则送入消息库，消息的各属性作为消息库的字段，以便于按属性查询。
- 检查消息库：如果某条消息符合发送条件，则取出该消息，送入发送队列；如果某条消息需要销毁，则销毁该消息。
- 收到查询消息后，进行按属性查询，并将查询结果返回查询实体。如果被查询消息需要发送，则送入发送队列。
- 收到名单消息后，如果是名单定义，则将名单追加到消息库；如果是名单删除，则从消息库中删除指示名称的名单。

4 实验设计

本文提供一个基于MDF开发的实

例——天文大数据模拟系统。天文科学是一个产生大数据的学科，例如在太阳与小行星的运行系统中，就有数以万计的小行星，计算这些小行星的轨迹是一个典型的大数据应用。这些小行星还会不断增加和消失，如果把小行星看作用户，这是非常有代表性的多用户动态运行系统。因此通过MDF开发一个系统，随时模拟小行星的运行，该系统支持小行星的随时更新。在本文中，笔者做了一个简化的实验性版本，用以说明原理，以太阳、地球、火星、金星4个星体为例，模拟星体的运行轨迹。星体之间由于受到引力作用，以太阳为中心做椭圆轨迹的运动。太阳系的运行是包括太阳在内的所有星体共同作用的结果，由于每个星体都是不停运动的，对其他星体引力的大小和方向在时刻变动。消息驱动框架正适宜去描述这类问题。在该模型中，每个星体都是一个实体，它们都具备相同的功能，而一个星体对另一个星体的引力影响可以认为是该星体发出了一个消息给另一个星体，星体根据消息内容做出自己的运行。在MDF方式下，每个星体只要给出自己的物理参数（质量、位置、速度），整个太阳系的运行就可以自动建立起来，而且可以随时添加新发现的星体（小行星），也可以随时删去星体。

图4显示了金星在太阳、地球和火星作用下的引力模型。金星有一个质量和初速度V_0使其能在引力作用下围绕太阳转动。

在该引力模型中，实体管理模块有两种实体类型：太阳和行星。地球、火星、金星为行星类型。实体管理模块定义太阳的位置固定不动，行星在引力的牵引下围绕太阳作公转运动。太阳实体具有质量（weight）和位置（position）两种属性，以（weight,positon）的消息格式发送并存放在消息管理模块。每个行星实体都有质量（weight）、位置（position）、速度（velocity）、时间（time）4种属性，并将这4种属性以（weight,position,velocity,time）的形式加以封装，作为消息相互传递通信。太阳的消息是被动查询类型，而每个行星实体的消息都是主动广播类型。以火星为例，火星实体向消息管理器发送一条查询消息，消息管理器解析火星实体发送来的查询消息的head部分，将查询到太阳的消息发送给火星，火星实体解析出太阳的位置和质量，又得到其他行星推送的消息，就可以计算出自己在当前位置受到的来自太阳和其他行星的引力，并计算下一步的位移。

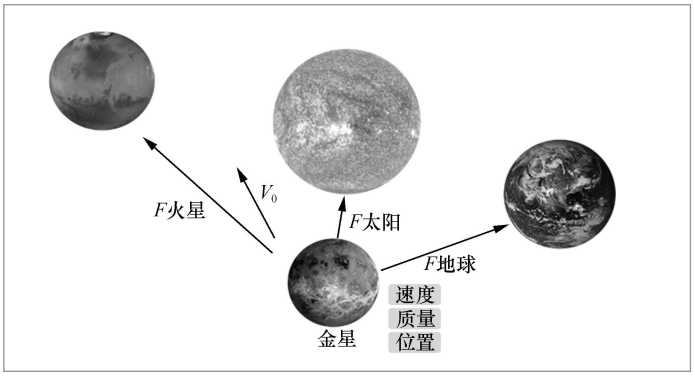


图4 三行星的运行模型

为了观察实验结果，将太阳、地球、金星和火星的运动轨迹以图像的形式展现出来，本次实例使用的质量、位置、速度数据均以目前太阳系运行模型为基准，参数设置在**表3**中，而**图5**则将**表3**的参数图形化，显示了各个星体的初始位置。

图6显示了太阳在静止不动的情况下，地球、金星和火星围绕太阳的运行轨迹。

表3 行星参数

行星	质量/kg	位置/m	速度/(m·s^{-1})
金星	$4.867\,6\times10^{24}$	$(0,\ 1.07\times10^{11})$	(−35 000, 0)
地球	5.98×10^{24}	$(1.496\times10^{11},\ 0)$	(0, 29 783)
火星	6.418×10^{23}	$(-2.496\times10^{11},\ 0)$	(0, −24 000)

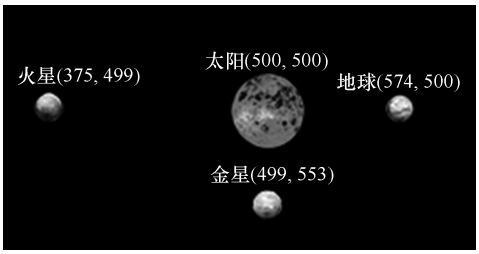


图 5 星体的初始位置

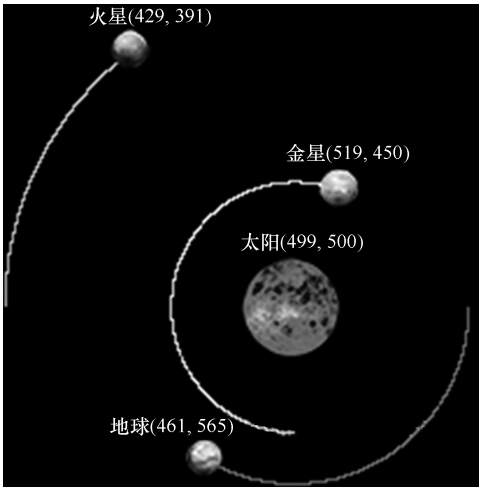


图 6 星体的运行轨迹

在该天体模型中，可以随时加入或撤销星体。当发现新的星体时，只要在实体管理模块输入星体正确的质量、速度和初始位置，就可以加入该天体模型中。新星体的加入或撤销会影响其他星体的运行轨迹，但不会导致整个模型的崩溃。

5 结束语

在大数据应用背景下，编程语言显示出与机器的相关性越来越小，与客观世界越来越贴近，程序的开发架构也越来越与计算机执行的过程脱离，而趋向于和问题结构越来越靠近，“描述就是编程”已经成为当前编程架构的关注点。编程语言已经逐渐从专业高端走向普通大众。本文研究的消息驱动框架的开发模式是一种典型的声明式编程，其中，消息管理模块是消息驱动框架的交通枢纽、神经中枢，它控制着通信唯一媒介——消息的接收、存储、发送以及维护等相关工作。这种编程框架模拟了许多自然和社会系统的行为方式，改变了人们的解决问题的方式，也将处理逻辑的重点从内部转移到外部，简化了问题求解的复杂度[22]。

在消息驱动的开发框架中，消息管理模块的定义与设计是重点，目前也只是完成初步的描述，还有很多细节和功能方面需要改进，以下几点是需要完善和考量的。

- 程序的顽健性：在消息管理模块中有大量的中间件，要保证每个中间件必须是顽健的。因此建立错误恢复机制、为开发人员提供测试和调试的接口以及提供较为明确的错误报告对消息驱动框架的发展是相当必要的。
- 消息管理：本文虽然对消息管理做了初步的约定和设计，但是在实际系统运行中，消息管理的要求是多样的，如何适应不同需求的消息管理模式，仍有待于继续探讨。
- 消息管理模块与实体模块和显示模块的对接：本文的工作主要是针对MDF的消息管理模块，实体模块以及数据管理和显示模块仍需要进一步完成和完善。

参考文献:

[1] 高志刚.自然资源系统的演化及其动力机制[J]. 资源开发与市场，2002，18(6)：14-17.
GAO Z G. Evolution and its driving mechanism of natural resources system[J]. Resource Development & Market，2002，18(6)：14-17.
[2] 洪学海，范灵俊，洪筱楠，等. 智慧城市建设

中政府大数据开放与市场化利用[J]. 大数据, 2016, 2(3): 17-26.
HONG X H, FAN L J, HONG X N, et al. Government big data opening and market utilization for smart city construction[J]. Big Data Research, 2016, 2(3): 17-26.

[3] 郑磊, 高丰. 中国开放政府数据平台研究：框架、现状与建议[J]. 电子政务, 2015(7): 8-16.
ZHENG L, GAO F. Study on the Chinese government data platform: framework, status and suggestions[J]. E-Government, 2015(7): 8-16.

[4] 马朝辉, 聂瑞华, 谭昊翔, 等. 大数据治理的数据模式与安全[J]. 大数据, 2016, 2(3): 83-95.
MA C H, NIE R H, TAN H X, et al. Research on data schema and security in data governance[J]. Big Data Research, 2016, 2(3): 83-95.

[5] MASKIT D, TAYLOR S. A message-driven programming system for fine-grain multicomputers[J]. Software: Practice and Experience, 1994, 24(10): 953-980.

[6] 邵长磊, 吾际舟. 计算机编程语言的演变[J]. 农业网络信息, 2009(11): 112-115.
SHAO C L, WU J Z. Evolution of computer programming languages[J]. Agriculture Network Information, 2009(11): 112-115.

[7] SAMMET J E. Programming languages: history and future[J]. Communications of the ACM, 1972, 15(7): 601-610.

[8] 朱树人,贺株莉. 面向对象与面向过程的程序设计方法比较[J].长沙电力学院学报, 1998, 13(2): 161-167.
ZHU S R, HE Z L. Compare the programming method of the object-oriented and the process-oriented[J]. Journal of Changsha University of Electric Power, 1998, 13(2): 161-167.

[9] 张丽霞. 面向过程的编程与面向对象的编程[J]. 赤峰学院学报, 2006, 22(5): 41-46.
ZHANG L X. Programming about process-oriented and object-oriented[J]. Journal of Chifeng College, 2006, 22(5): 41-46.

[10] BAINOMUGISHA E, CARRETON A L, CUTSEM T, et al. A survey on reactive programming[J]. ACM Computing Surveys (CSUR), 2013, 45(4): 115-123.

[11] KANTOROWITZ E, TADMOR S. Toward specification-oriented frameworks[C]//The Israeli Workshop on Programming Languages & Development Environments, July 1, 2002, Hefa, Israel. [S.l.:s.n.], 2002: 4.

[12] DE PAOLI F, TISATO F. On the complementary nature of event-driven and time-driven models[J]. Control Engineering Practice,1996, 4(6): 847-854.

[13] HE C, HE K. A role-based approach to design pattern modeling and implementation[J]. Journal of Software, 2006, 17(4): 658-669.

[14] ZHANG T, ZHANG Y, YU X F, et al. MDF based design patterns modeling and model transformation[J]. Journal of Software, 2008, 19(9): 2203-2217.

[15] PENCZEK F, HERHUT S, SCHOLZ S B, et al. Message driven programming with s-net: methodology and performance[C]// The 39th International Conference on Parallel Processing Workshops (ICPPW), September 13-16, 2010, San Diego, CA, USA. New Jersey: IEEE Press, 2010: 405-412.

[16] PHILLIPS J C, STONE J E, SCHULTEN K. Adapting a message-driven parallel application to GPU-accelerated clusters[C]// International Conference for High Performance Computing, Networking, Storage and Analysis, November 15-21, 2008, Austin, TX, USA. New Jersey: IEEE Press, 2008: 1-9.

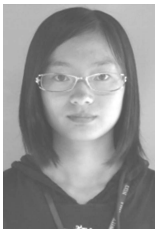
[17] 肖美华, 余立金. SOCKET通信程序模型抽取及可靠性验证[J]. 计算机科学, 2012, 39(11):102-105.
XIAO M H, YU L J. Model extraction and reliability verification on socket communication program[J]. Computer Science, 2012, 39(11):102-105.

[18] 刘邦桂, 李正凡. 用Java实现流式Socket通信[J]. 华东交通大学学报, 2007, 24(5): 110-112.
LIU B G, LI Z F. Implementation of socket stream communication in Java[J]. Journal of East China Jiaotong University, 2007, 24(5): 110-112.

[19] 于海, 樊晓桠. 基于FPGA异步FIFO的研究与实现[J]. 微电子学与计算机, 2007, 24(3): 210-213.
YU H, FAN X Y. Research and implementation of asynchronous FIFO based on FPGA[J]. Microelectronics &

Computer, 2007, 24(3): 210-213.
[20] 夏家莉, 陈辉. 一种动态优先级实时任务调度算法[J]. 计算机学报, 2012, 35(12): 2685-2695.
XIA J L, CHEN H. A real-time tasks scheduling algorithm based on dynamic priority[J]. Chinese Journal of Computers, 2012, 35(12): 2685-2695.
[21] 李建中, 张东东. 滑动窗口规模的动态调整算法[J]. 软件学报, 2004, 15(12): 1800-1814.
LI J Z, ZHANG D D. Algorithms for dynamically adjusting the sizes of sliding windows[J]. Journal of Software, 2004, 15(12): 1800-1814.
[22] KALÉ L V, BHANDARKAR M A. Structured dagger: a coordination language for message-driven programming[C]//The Second International Euro-Par Conference, August 26-29, 1996, Lyon, France. Berlin: Springer Berlin Heidelberg, 1996: 646-653.

作者简介



贵芳(1991-),女,合肥工业大学计算机与信息学院硕士生,主要研究方向为大数据应用、机器学习和程序语言等。

李廉(1951-),男,合肥工业大学计算机与信息学院教授,主要从事大数据应用、机器学习、无线传感器网络等领域的研究工作。担任教育部高等学校大学计算机课程教学指导委员会主任,中国计算机学会理论计算机专业委员会主任。先后承担或参与国家自然科学基金重点项目和面上项目,“973”计划前期研究专项,省、部级重点项目,国家科技攻关计划项目等10项;发表论文100余篇;已获得授权专利2项,国家软件著作权10件。获安徽省教学成果奖特等奖一项,国家教学成果奖二等奖一项。

杨静(1979-),女,博士,合肥工业大学计算机与信息学院副教授,主要研究方向为人工智能和数据挖掘。

武永卫(1974-),男,清华大学计算机科学与技术系教授、副系主任,主要从事并行与分布式处理、云存储和大数据系统等方面的研究工作。担任IEEE Transactions on Sustainable Computing的指导委员会委员,IEEE Cloud Computing、IEEE Transactions on Cloud Computing等国际期刊编委。在TC、TPDS等国际期刊和MICRO、FSE、ATC等国际发表学术论文100余篇,获得FSE等国际会议最佳论文3篇;承担了国家自然科学基金重点课题、“863”计划、欧盟FP6、Intel、百度等课题;申请中国发明专利20项,美国发明专利3项;获国家技术发明奖二等奖、国家科技进步奖二等奖,省部级一等奖2项,中创软件人才奖等奖励。

收稿日期: 2016-06-20
基金项目: 国家自然科学基金资助项目(No.61370219, No.61433008, No.U1435216);广东省佛山市创新团队基金资助项目(No.2015IT100095)
Foundation Items: The National Natural Science Foundation of China(No.61370219, No.61433008, No.U1435216), The Innovation Teams Foundation of Foshan, Guangdong(No. 2015IT100095)