文章编号:1671-4598(2020)04-0170-06 DOI:10.16526/j.cnki.11—4762/tp.2020.04.035 中图分类号:TP311.5 文献标识码:A

一种新的异构系统集成方法、框架与实现

曹建平，朱国涛，孙文柱，胡文婷
(海军航空大学 青岛校区，山东 青岛 266000)

摘要：为解决分布式异构系统集成时常见的紧耦合问题，实现可扩展、易配置和实时性的集成要求，提出了一种新的分布式系统集成方法和框架，并利用C++语言进行了实现；该框架主要采取先集中后分发的总体思路，利用数据传输只依赖数据大小而与数据内容无关的特性将与子系统紧密相关的业务数据转换为传输数据，从而使软件通过配置能柔性适应多种应用场景；基于该框架的软件实现已应用于某型飞机分队战术模拟训练系统，取得了较好的应用效果。

关键词：分布式异构系统；集成；可扩展

A Novel Distributed Heterogeneous System Integration Method, Framework and Implementation

Cao Jianping, Zhu Guotao, Sun Wenzhu, Hu Wenting
(Qingdao Branch, Naval Aeronautical Institute, Qingdao 266000, China)

Abstract: In order to solve the common tight coupling problem in the integration of distributed heterogeneous systems and realize the integration requirements of extensible, configurability and real-time performance, a novel method and framework for the integration of distributed systems is proposed, and its implementation based on C++ language is completed. The framework mainly adopts the general idea of centralized distribution, which converts business data closely related to subsystems into transmission data by using the characteristics that data transmission only depends on data size and is independent of data content, so that the software can flexibly adapt to a variety of application scenarios through configuration. The software implementation based on this framework has been applied to the Tactical Simulation Training System of a certain type of aircraft unit, and has achieved good application results.

Keywords: distributed heterogeneous system; integration; extensible

0 引言

为提高性能和费效比，大型复杂计算常常采用分布式方式进行[1]。分布式系统作为一个体系，一般包括多个异构的子系统，各子系统的多样性、异构性为集成工作带来很大挑战，因此建立统一的系统集成框架具有重要意义。该框架不但要保证系统集成达到低耦合、扩展性、易配置和实时性等要求，还要有效提高开发、升级和维护效率。

当前，对于分布式异构系统的集成方法和框架包括：应用于分布式仿真的分布式交互仿真（distributed interactive simulation，DIS）框架和高层体系结构（high level architecture，HLA）框架，目前已形成IEEE标准[2]，并有多种软件实现，在军事仿真领域应用广泛；基于XML、SOAP和WSDL技术的Web Service框架[3]，可实现异构的分布式web应用进行实时交互，在软件实现上包括微软.NET，IBM的WebSphere和Borland的JBuilder等，在电子商务、电子政务和游戏领域应用广泛；以数据为中心，采用虚拟总线的“发布-订阅”通信模式，强调高可靠性和实时性的数据分发服务（Data Distribution Service）框架[4]，通过21种QoS服务质量策略能很好支持异构设备之间数据分发和传输，包括OpenDDS、OpenSpliceDDS等软件实现，广泛应用于国防、民航和工业控制领域；在物联网领域还包括AMOP、XMPP、MQTT和COAP等框架和实现。Michael等为集成多种异构模拟器提供了一种名为Ptolemy II的集成环境；Silva T W B[5]等针对异构的硬件平台环境，基于HLA协议，在联邦大使和应用程序之间提出一种通用的松耦合的虚拟总线（Virtual Bus）计算平台，达到了提高集成效率的目的；朱晓攀[6]等根据像质处理提升仿真系统需求提出了采用数据分发服务（data distributed service，DDS）技术进行集成，实现海量图像和数据传输的按需分配、实时性、可靠性、扩展性和高吞吐率要求。

以上框架和相应的实现在各自领域都达到了系统集成框架的功能和性能要求，但在细分领域还需要进行定制开发，存在一定的开发和维护难度。为此，针对分布式飞行仿真提出了一种新的分布式异构系统集成方法和框架，并利用C++原生语言进行了实现。该框架在某型飞机分队战术模拟训练中得到了应用，应用结果表明该框架可用于互联包括视景显示计算机、飞行控制计算机、教员控制台、计算机生成兵力等多种异构实体，具有低耦合、扩展性强、易配置和实时性高的优点。该框架还可应用于其他类似的轻量级分布式实时应用的异构集成。

收稿日期:2019-07-07； **修回日期**:2019-08-26。

作者简介:曹建平(1984-),男,内蒙古通辽人,硕士研究生,讲师,主要从事分布式仿真、并行程序设计方向的研究。

1　基于集中－分发模式的异构系统集成方法

对于大型的分布式系统，系统之间的关系比较复杂，传统一般采取如图 1 所示的紧耦合方式进行集成。这种设计对系统的开发、升级和扩展都不利，尤其在增加新的系统或数据时（图 1 中的虚线）时非常困难，随着系统复杂度的增加甚至会造成整个分布式系统的崩溃。因此，采取松散耦合的方式进行系统集成已经成为共识。

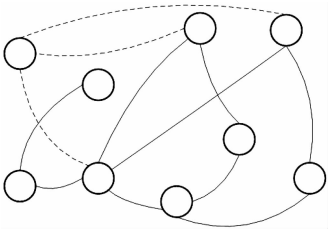

图 1　传统设计的紧耦合网络

为破除系统之间的紧耦合关系，在系统之间添加一个分发服务器，每个系统通过分发服务器进行交互，如图 2 所示。该设计的基本思想就是将各系统的数据集中到分发服务器上，同时分发服务器又实时地将数据分发给各子系统，从而达到了分布式系统数据在各子系统间的共享。各子系统都包含了一个解析子模块，负责对共享的数据进行过滤和解析，这样各子系统在升级或添加新的子系统时只需要更新解析子模块即可，从而达到了系统之间的松耦合，将这种数据集成方式称为“集中－分发”模式。

该设计中的分发服务器与 DDS 技术中的“虚拟总线”概念有所类似，都是为了达到子系统间数据共享的目的，但实际上是有区别的，主要体现为两点：一是 DDS 中的“总线”是虚拟的，实际上是通过网络中间层实现的，而本设计是存在一个确定且独立运行的分发服务器；二是 DDS 的“虚拟总线”核心是“发布－订阅”模式，而本设计是将分布式系统中的所有数据送到各子系统，由各子系统过滤解析获取需要的数据。之所以采取这种方式还是由于面向的是轻量级应用，虽然通过“发布－订阅”方式可减少数据冗余，但对数据传输效率带来不良影响且对系统的可扩展也不利。

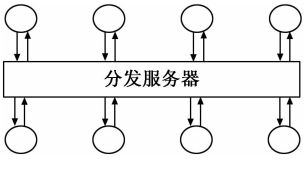

图 2　基于“集中－分发”模式的松散耦合设计

在该模式中，分发服务器处于核心位置，要实现分发服务器与所有子系统进行交互要求分发服务器功能单一，即只完成数据的分发，不涉及到数据的具体内容，分发服务器与各子系统之间不存在任何额外的协议。因此，对于各子系统而言，分发服务器实际上相当于一个快递公司，只负责邮件（数据）的托运而没有权限查看或处理邮件的内容。这是达到松散耦合的一个必要条件，但不是充分条件。原因在于，虽然分发器对传输数据的内容是无关的，但却对传输数据的大小是敏感的，因此需对分布式系统的单次传输量进行统一规定。这样，分发服务器就成为了一个特殊的快递公司，它只接收具有固定大小的邮件并将其发送到所有想接收这个邮件的客户（各子系统）手里。对于一个子系统而言，需将其单位业务数据包的大小包装成分发器传输规定的大小。要完成松散耦合，另一个必要条件是各子系统能正确过滤和解析从分发器发送过来的数据。为实现该功能，在分发服务器中设置一个缓冲区域，在该区域中各子系统数据的相对位置是固定的，对各子系统进行编号，将编号和位置建立一一映射。这样，某一个子系统就可以通过编号设置接收哪些子系统的数据，并根据相对位置找到对应的数据并进行解析。该过程通过配置文件实现，为系统的可扩展提供了可能。另外，分发服务器缓冲区域在场景运行之前就是固定好不能更改的，分发服务器中的接收和发送线程通信采用“best－effort”模式，只要缓冲区域内有数据就发送，从而可以适应不同子系统的发送接收数据频率的差异性。

为此，在分发服务器和子系统之间还需要设置一个中间件完成上述工作。该中间件具有 3 个作用：1）将本地单位业务数据包装成符合传输要求的数据；2）过滤和解析远程子系统数据；3）进行数据传输。如图 3 所示，根据中间件功能，中间件可划分为 3 个模块，分别是数据打包模块、数据解析模块和数据传输模块。整个数据流过程是：打包模块负责将子系统的数据打包成符合分发服务器规定大小的数据包并通过传输模块中的发送线程将本地数据发送到分发服务器；而传输模块中的接收线程负责接收分发服务器发送的整个分布式系统的共享数据，并由解析模块对共享数据进行过滤和解析。中间件可以以多种方式集成到子系统中，既可以打包成动态链接库由子系统调用，也可以是一个单独的进程与子系统进行数据交互。

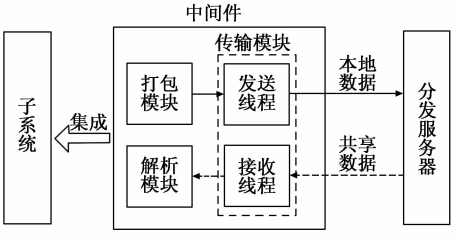

图 3　中间件的模块划分与数据流

由上可知，分发服务器是业务无关的，而子系统是通过中间件来分享分布式系统的数据，因此当一个新的子系统加入分布式系统时只需要定制中间件中的解析模块即可实现，从而实现了子系统间数据集成的松散耦合。

2　基于集中－分发模式的异构系统集成框架

面向轻量级的分布式实时应用，根据基于“集中－分发”模式的异构数据集成方法，实现了数据集成框架，总

体框架如图 4 所示。在该框架下，将分布式应用的整个运行过程称为场景，将场景中的每个子系统称为参与者，每个参与者有一个角色属性（参与者括号的内容），根据角色属性可确定其业务数据类型。根据数据流方向，将框架分为 3 个部分：由参与者组成的参与者集、分发服务器、嵌入了中间件的中间层。原生的中间件是一个动态链接库，可以采用多种方式与参与者集成，图 4 给出了常见的 4 种方式：1）参与者程序直接调用中间件动态库；2）将中间件动态库封装成为一个独立的程序，通过网络与参与者进行通信；3）参与者基于高层体系结构（HLA）开发，则将中间件封装成为一个联邦成员，通过 HLA/RTI 通信；4）参与者是一种多 Agent 体系结构，则可将中间件封装成为一个 Agent，通过多 Agent 系统中的通用黑板（GBB）进行通信。

分发服务器则有两部分组成，分别是分发模块和缓冲区。每一个参与者按顺序对应缓冲区的一个子区域，每个子区域存储对应参与者的实时状态。每个子区域又对应分发模块中的 IO 线程，IO 线程负责与对应的中间件进行网络通信，从而更新缓冲区子区域中的参与者实时状态。

场景中的参与者对数据需求是不同的，有的只生产数据，有的只消费数据，有的既生产也消费数据。因此，在分发服务器中分发模块分为只写模式、只读模式、读写模式，只写模式对应一个写线程，只读模式对应一个读线程，读写模式对应一个写线程和一个读线程。写线程从中间件读取参数者数据并存储到缓冲区对应子区域；读线程读取整个缓冲区实时状态并发送到对应参数者的中间件。缓冲区的数据结构也是可以定制的，一种是采用先进先出的链表结构以保留历史数据；另一种缓冲区空间固定，只保留实时最新状态。

该框架可柔性适应多种场景，原因就在于分发服务器和中间件是可配置的。对于分发服务器，需要配置的必要信息有场景中的参与者数目、每个参与者的 IO 模式、每个参与者的网络地址、每个参与者联接的网络协议和单次传输数据大小；对于中间件，需要配置感兴趣的参与者的顺序号及其角色、分发服务器网络地址、单次传输数据大小。从配置信息可以看出，分发服务器不关心参与者的角色信息，中间件为解析数据必须要知道对应参与者的角色信息。

该框架下，参与者之间不直接互联，而是以分发服务器为中心进行数据交互。因此可实现增量开发，即在场景中增删参与者时只需要完善配置信息并对中间件进行简单的二次定制即可实现。

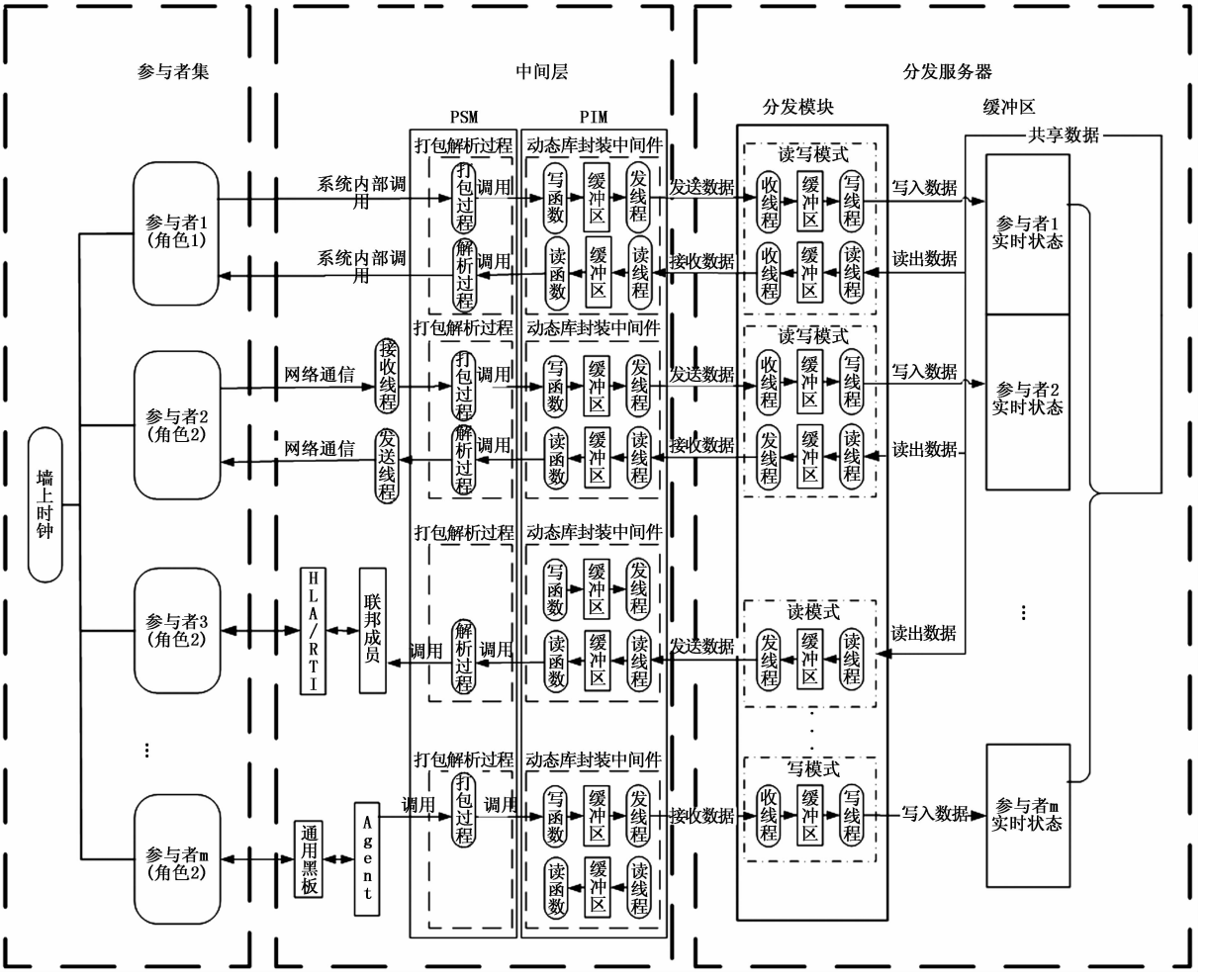

图 4 基于“集中－分发”模式的数据集成架构

3　基于集中—分发模式的异构系统集成实现

该框架包括两个软件模块，分别是分发服务器和中间件。按照本框架，这两个软件模块具有多种实现方式，本文利用 C++语言进行了实现，下面分别予以介绍。

3.1　分发服务器的实现

按照框架要求，分发服务器除了缓冲区、分发模块以外还需要有配置功能和网路传输功能，因此分发服务器还包括配置子模块和网络传输子模块，模块划分如图 5 所示。配置子模块主要用来读取配置文件，在正式运行之前为分发子模块和缓冲区子模块配置好相关参数。网络运输子模块封装了 WinSock 的一些常用函数，以动态链接库的形式给出。缓冲区子模块给出了存储缓存各参与者传输数据的数据结构。分发模块处于分发器的核心，由配置子模块配置其相关参数，并调用网络传输子模块传输数据并对缓冲区子模块进行更新，完成节点间的数据分发。

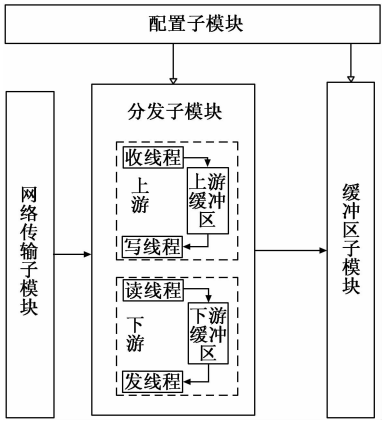

图 5　分发服务器的模块划分

3.1.1　配置子模块实现

分发服务器采用 ini 文件方式配置，程序中采用类的方式进行封装。类中的成员变量是配置参数，由于程序中需且仅需一个参数，因此都是静态变量。成员函数 _ initialConfigParams 是读取相应的配置文件为配置参数赋值。

3.1.2　网络传输子模块实现

为了重用代码，将 winsock 的一些常用函数封装为类，类继承关系如图 6 所示。

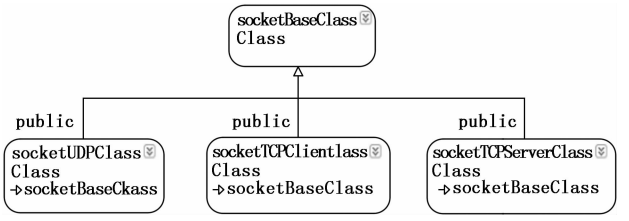

图 6　网络传输子模块类图

其中，socketBaseClass 作为基类，主要封装了本地和远程地址以及 Init、send 和 Receive 三个成员函数，在基类中 Init、Send 和 Receive 是虚函数，需要在子类中进行重写。socketUDPClass 对 UDP 协议相关的函数进行了封装，基类继承了相关变量并对 Init、Send 和 Receive 进行了重写。由于 UDP 是不分服务端和客户端的，所以只要一个类就可以实现。socketTCPClientClass 和 socketTCPServerClass 封装了 TCP 协议的相关函数。对于 TCP 而言，服务器端和客户端的初始化过程是不同的，因此封装为两个类。

3.1.3　缓冲区子模块实现

按照缓冲发送模式的要求，在分发器中设置一个缓冲区用来暂存各参与者的状态数据。如图 7 所示，当要联入 m 个参与者时，缓冲区就平均分为 m 块，每一块用来缓存对应参与者各状态的数据。

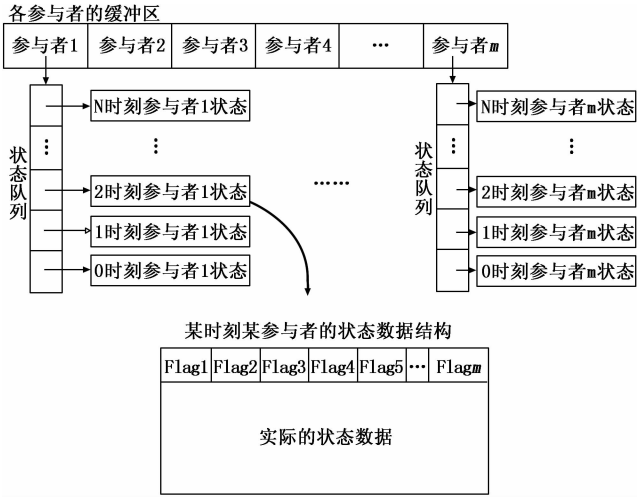

图 7　分发器缓冲区数据结构

实现时整个缓冲区作为一个数组，数组的成员又是一个先进先出的队列结构，而队列中的每个成员又是指向实际状态数据内存的指针。因此，利用标准 C++的 STL 库，将程序中的缓冲区的数据结构声明如下：

由于目前联网的参与者的数目是事先确定的，因此缓冲区数组的大小是一定的，也就是说对应参与者的队列是在联网运行前已经确定好了。在联网分发过程中变化的只有队列中的元素，也就是要对队列进行压入数据与弹出数据操作。对于压入数据是比较简单的，当某一个参与者状态数据传入时就压入缓冲区对应队列的头部；而弹出数据相对要复杂一些，原因就在于该数据要传到除自身以外的所有参与者处后才能弹出。

为了实现这个逻辑，为队列的每一个成员即参与者的状态数据结构加了一个头部，该头部作为标志位数组是由一定数目的整数类型组成的，该数组的大小与参与者数目相同，数组中的数值取 1 时表示该状态数据已发送到所在位置的参与者处，否则表示还没有发送，当标志位全部为 1 时表示数据全部发送到位，程序会自动删除该状态数据，即完成了队列状态数据的弹出操作。

3.1.4　分发子模块实现

如图 5 可知，分发子模块是分发器的核心模块，其具体负责将接收到的参与者状态数据分发到其他参与者处。分发子模块是由上游子模块和下游子模块两部分组成的，

上游子模块负责接收某一个参与者的状态数据并将其写入缓冲区内，而下游子模块负责读取其他参与者的状态数据并发送回参与者处，如图 8 所示。

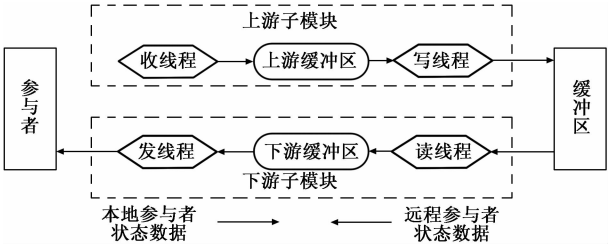

图 8 分发子模块组成

由图 8 所示，根据“生产者－消费者模式”，在上游子模块和上游子模块中都存在一个数据缓冲区，该缓冲区主要目的也是为了分离收线程与写线程或者读线程与发线程之间的耦合关系，使其不相互依赖从而达到提高效率的目的。

分发子模块中的收线程、写线程、发线程和读线程都是独立运行的线程，并且可知，在整个训练过程当中，线程数目是一定的（与场景中的参与者数目成正比），并且各个线程一旦开始运行就不会停下来，直到程序终止，也就是不会频繁地进行创建和消除，从而节省了调度时间，提高了分发效率。在程序开发过程中，涉及的开发难度主要体现在线程之间的同步，在程序实现过程当中主要采用关键代码段的方式来实现的。

在 4 个线程当中，收线程、写线程和发线程的逻辑相对比较简单，重点描述一下读线程。读线程需要额外完成上文提到的数据弹出逻辑。读线程主体是一个 while 循环，首先申请存放读取数据的内存区并将其初始化为 0，接着依次读取缓冲区成员，当某个缓冲区成员队列不为空是从队列尾部开始向前遍历读取状态数据，读取时首先判读对应标志位是否以置为 1，如果是的话就继续向前遍历直到遇到标志位为 0 的队列成员，将该队列数据存到内存区域且设置标志位为 1，最后判断该状态的标志位是否都为 1，如果是则将该状态数据弹出。其总体流程如图 9 所示。

3.2 中间件的实现

由图 4 体系结构可知，中间层主要实现参与者与分发服务器之间的数据传输、打包与解析。对于不同角色的参与者而言，由于整个场景中的单次传输数据大小是相同的，因此传输模块和打包模块是固定不变的。而参与者角色有所不同，因此解析模块是可定制的，从而实现将传输数据转化为业务数据。

3.2.1 传输模块的实现

传输模块的组成如图 10 所示，可以看到传输模块的组成与分发服务器器的分发子模块的组成是类似的，不同之处在于写线程变成了写函数，而读线程变成了读函数。参与者在集成中间层时，只需要周期性调用写函数和读函数，就可以完成数据的接收与发送。

传输模块中的发送数据区和接收数据区也是以队列的

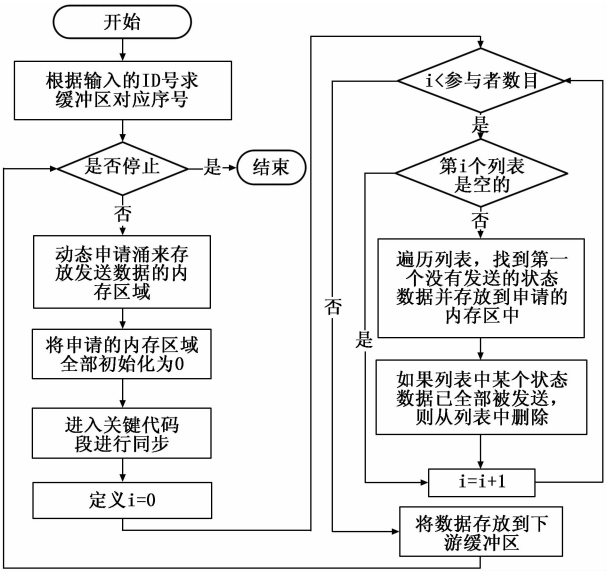

图 9 读线程逻辑流程图

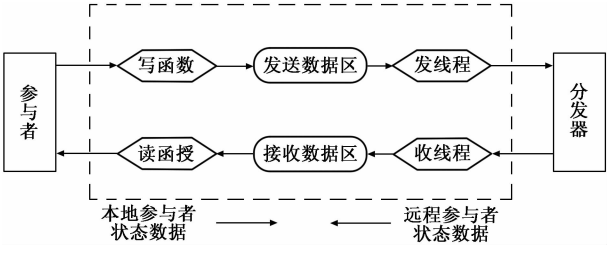

图 10 传输模块组成

数据结构，写函数、发线程、读函授和收线程之间的同步依然是通过关键代码段。

3.2.2 解析模块实现

在分发模块和传输模块中没有涉及到各参与者的业务数据，这样就非常有利于扩展参与者的角色。解析模块就负责将传输数据转换为程序能够处理的业务数据。

我们知道，在内存块中数据都是以 0、1 的形式存在的，在没有上下文的情况下是没有任何意义的，但当我们取得某一块内存的地址并将其解释为事先定义好的数据结构，那么这块内存的数据就能得到解析了，如图 11 所示。

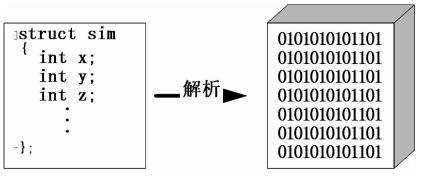

图 11 利用定义好的结构体解析内存块

传输模块类中定义的接收数据区是一大块数据，它包含了所有远程参与者的状态数据，将参与者 ID 号与接收数据区的内存块建立一一对应的关系，通过参与者的 ID 号就能找到对应的数据块。如图 12 所示，由于单次传输数据大小是一定的，因此在分发器缓冲区被均分为 N 块（N 是参与者数目），每一块的大小都是单次传输数据大小。那么分

发器内缓冲区的数据排列都是按照配置文件中参与者 ID 数组给出的顺序排列，那么数据由分发器发往某个参与者时前后顺序仍然是不变的（注意此时本地参与者数据不会发到本地参与者），那么再通过本地参与者定义的配置文件中的远程参与者数组所规定的顺序就可以找到对应 ID 远程参与者数据地址了。

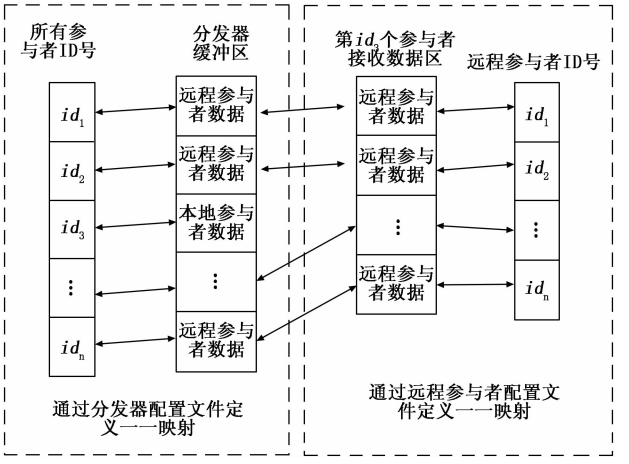

图 12　参与者 ID 号与内存块中数据一一映射关系

为便于实现解析过程，提供了一个远程参与者类，如图 13 所示。类中给出了 5 种默认的参与者角色，分别是模拟器角色、教控台角色、引导台角色、VRForce 产生的虚拟兵力角色和 MaxSim 产生的虚拟兵力角色。如果在场景中要加入新的角色，只要继承该类并在字段中添加要扩展的参与者的业务数据结构并重写 initFromIni（）函数即可。

图 13　远程参与者类结构

4　应用

某型飞机分队战术模拟训练系统是面向单机、双机、四机及以上相同或不同机型开展战术协同训练科目而开发的。该系统由多个模拟器、教员台、引导台、虚拟兵力、态势监控等多个异构子系统组成，如何将这些异构的子系统进行有效的数据集成并满足系统间数据交互的实时性和可靠性是开发分队战术模拟训练系统的一个重点和难点。采用本文提供的分发服务器和中间层软件进行了集成，集成后网络结构如图 14 所示。

从该网络结构可以看出，该系统参与者数目 $N=12$，

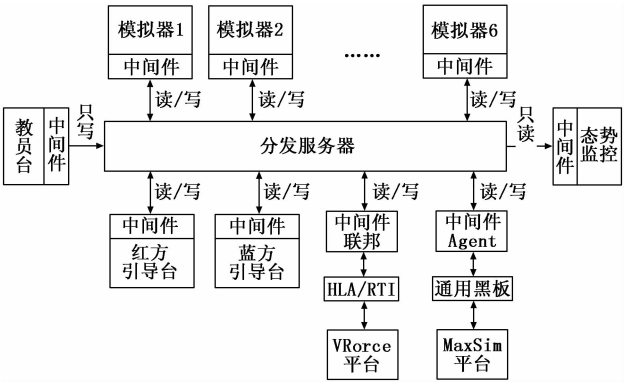

图 14　某型飞机分队战术模拟训练系统网络结构

参与者的角色有 5 种：模拟器（编号 1 到 6）、引导台（编号 7、8）、VRForce 计算机生成兵力平台（编号 9）、MaxSim 计算机生成兵力平台（编号 10）、教员台（编号 11）和态势监控（编号 12）。为每种角色的业务数据定义特定的数据结构。

按照本文给出的方法集成的某型飞机分队战术模拟训练系统已经在部队得到了初步的应用，实践表明，该方法能够将异构的子系统快速进行集成并满足通信的实时性和可靠性要求。

5　结语

本文旨在为连接节点数目不多、数据吞吐量不大，但数据交互实时性要求高的轻量级的分布式应用场景提供一种易使用的系统集成方法、框架和实现。从设计和使用看，本文给出的框架具有模块化、可扩展性和可配置性的优点。并且所有代码都是基于 C++标准库完成，因此具有跨平台的特点。下一步可进一步提高分发服务器的柔性，达到根据参与者的请求进行自动配置的功能。

参考文献：

[1] Foster I，Kesselman C，Nick J M，et al. Grid services for distributed system integration [J]. Computer，2002，35（6）：37-46.

[2] IEEE B E . IEEE standard for modeling and simulation（M&S）high level architecture（HLA）－framework and rules－redline [Z]. IEEE，2010：1-378.

[3] Baina K，Benatallah B，Casati F，et al. Model－driven web service development [J]. International Journal of Web Services Research，2004，3084（4）：42-45.

[4] Schlesselman J M，Pardo－Castellote G，Farabaugh B. OMG data－distribution service（DDS）：architectural update [A]. Military Communications Conference [C]. 2004.

[5] Silva T W B，Morais D C，Andrade H G R，et al. Environment for integration of distributed heterogeneous computing systems [J]. Journal of Internet Services & Applications，2018，9（1）：4.

[6] 朱晓攀，陈　实. 基于 DDS 的像质处理提升仿真系统设计与实现 [J]. 系统工程与电子技术，2018，40（8）：218-225.