文章编号:1671-4598(2020)06-0119-06 DOI:10.16526/j.cnki.11—4762/tp.2020.06.025 中图分类号:TP274.2 文献标识码:A

基于 Actor 模型的浮式保障平台数据采集系统设计

陆俊杰[1]，周志刚[2]，李晓峰[2]
(1. 江南大学 物联网工程学院，江苏 无锡 214000；2. 中国船舶科学研究中心，江苏 无锡 21400)

摘要：面对工程项目中需要采集的数据量越来越庞大的挑战，现有的数据采集系统主要采用C#委托和事件的观察者模式实现数据的传输，具有代码花销大、耦合度高的弊端；对 Actor 模型的异步解耦特性的研究下，提出基于 Actor 模型的浮式保障平台数据采集系统的设计和实现；根据浮式保障平台数据采集系统的需求和性能制定设计方案，将 Actor 模型的层级性与设计方案相结合制定出特殊的 Actor 系统；通过浮式保障平台数据采集系统下各个子系统进行的数据分类传输测试，实验结果验证了设计方案的可行性，且满足系统的高并发性、低耦合性以及加快数据的处理性能；因而 Actor 模型能高效替代原有的数据传输方式，减少内存消耗以及便于后期的升级维护。

关键词：数据采集；Actor 模型；异步解耦；高并发性；消息传递

Design of Data Acquisition System for Floating Support Platform Based on Actor Model

Lu Junjie[1], Zhou Zhigang[2], Li Xiaofeng[2]
(1. School of Internet of Things Engineering, Jiangnan University, Wuxi 214000, China;
2. China Ship Scientific Research Center, Wuxi 214000, China)

Abstract: Facing the challenge of increasing the amount of data that needs to be collected in engineering projects, the existing data acquisition system mainly uses C# delegate and event observer mode to realize data transmission, which has the disadvantages of large code cost and high coupling degree. Based on the research of asynchronous decoupling characteristics of Actor model, the design and implementation of data acquisition system based on Actor model for floating support platform is proposed. According to the requirements and performance of the data acquisition system of the floating support platform, the design scheme is combined, and the actor model's hierarchical nature and design scheme are combined to develop a special Actor system. Through the data classification transmission test of each subsystem under the data acquisition system of floating support platform, the experimental results verify the feasibility of the design scheme and meet the high concurrency, low coupling and speed of data processing performance. Therefore, the Actor model can effectively replace the original data transmission mode, reduce memory consumption and facilitate post—upgrade maintenance.

Keywords: data collection; actor model; asynchronous decoupling; high concurrency; messaging

0 引言

以试验平台为载体搭载各种试验系统与设备，对试验平台的关键共性技术与专有技术进行集中的试验验证，对其经济性、可行性、适用性、安全性、可靠性等方面进行检验及评估，为特定产品的设计、建造和运行维护提供坚实的技术基础及指导依据，形成技术研究与工程示范相结合的重大成果。

浮式保障平台的验证项目涉及到众多内容和技术门类，根据研究内容，可将平台试验验证子系统划分为如下内容：腐蚀监检测系统、浮式防波堤测量系统、波浪能发电性能测试系统、结构安全监测系统、环境监测系统和系统总体概览。根据各个子系统的验证要求需要采集的数据参数种类及数目繁多，并且数据无法实现互联互通，由此数据采集系统的设计目标就是要实现数据参数的集中显示和管理功能。

现如今有许多种不同的数据采集系统的设计但在各方各面都存在着缺点。在基于多线程的航空发动机数据采集系统中，采用多线程的方式结合 MFC 进行系统设计，其缺点是：1）与 Actor 模型自身高并发的特性相比较，会产生大量的代码花销占用内存[1]；2）相较于采用 WPF，MFC 这款编程框架无论在运行速度、开发成本等性能上都不能比拟。在基于虚拟仪器的风洞数据采集系统中，其缺点是系统耦合度高导致容错率低不易维护[2]。另外，在相同条件下使用基于C#的委托和事件的观察者模式，与 Actor 模型相比较同样存在系统耦合度高、不易升级维护的缺陷。从上述各种采集系统的设计发现大多数系统不能有效地解决内存花销大以及降低解耦度的问题，而利用 Actor 模型可以解耦系统，增加复用性，避免堵塞，尤其是像这种分布

收稿日期:2019-10-09； 修回日期:2019-11-28。

作者简介:陆俊杰(1995-),男,江苏无锡人,硕士,工程师,主要从事试验室数据采集和存储软件方向的开发和应用。

式系统可以提高并发性能[3]，易于升级和维护。

本文将介绍基于 Actor 模型的浮式保障平台数据采集系统的设计与实现，对系统中各个功能点的测试验证设计方案的可行性并且达到预期的设计目标。浮式保障平台数据采集系统将采用 Akka. Net 框架来设计派生自基类 Actor 的各类子系统模块。在各子系统模块内自定义 Actor 类的消息类型用以传递各类子系统的参数数据，通过 Actor 类与控件类的关联实现数据的实时显示以及不同控件类的交互。

1 浮式保障平台数据采集系统结构与原理

Actor 模型最早是由 Carl Hewitt 于 1973 提出[4]来的一个关于并发和分配问题的开创性概念。通过引用单个原语称之为 Actor，并用于并发和分布式实体，这种方法提供的高级抽象结合其灵活性和效率使其对当今的弹性多核系统以及在 Internet 规模上分发的任务具有极大的吸引力[5]。

Actor 模型采用万物皆可建模为 Actor 的理念[6]，所有逻辑或者模块均可被看作 Actor，它们彼此之间可以直接发送消息，不需要通过什么中介，并且是异步发送和处理的。因而通过不同 Actor 之间的消息传递实现模块之间的通信和交互。Actor 模型具有许多特性，例如它的状态是本地的，无法通过外部访问；所有的 Actor 必须有响应消息传递来通信；异步非阻塞的事件驱动编程模型等等。最值得称赞的是 Actor 模型在实现过程中将十分轻量，可以快速的批量创建和销毁，“几十万甚至上百万进程同时并行运行十分常见，而且经常仅仅占用很少的内存”[6]。正是由于 Actor 具有如此多的特性和优点，因而能够克服传统面向对象编程程序的局限性，高度满足并发和分布式系统的挑战。

浮式保障平台数据采集系统的软件部分在数据流的本质上是符合 Actor 模型的[7]，这类系统都是通过相关配置信息的计算与处理来实现所需物理量的监测和采集，从而根据各类控件实现数据的显示以及后续利用配置信息与采集数据的运算和分析存储。因此整个系统可以视作是一个流处理的系统[8]，在每一步的环节都可以建立 Actor 模型来处理和传递数据，提高各类专业化的数据采集子系统的解耦性，并且易于升级改造和维护。整个流处理系统被 Actor 强制组织成一个树状的层次结构，根据模型的监督特性和传递消息的特殊性而减少出错以及避免线程的锁定和堵塞，从而保障整个系统的安全可靠并且降低升级维护的难度。

这里要详细描述一下浮式保障平台数据采集系统的业务架构，从而可以利用 Actor 模型去搭建完整的 Actor 系统。系统完整的业务架构如图 1 所示。

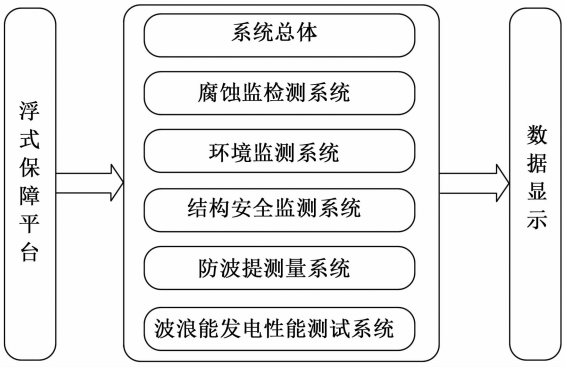

图 1 系统业务架构

正如上文引言介绍，浮式保障平台有 6 个子系统功能，接下来将一一介绍业务架构中各子系统的功能。

1）系统总体作为子系统其主要功能是显示各功能子系统下主要的数据，达到概览系统总体的功能。

2）环境监测系统最主要的功能是采取波浪骑士以及常用环境传感器来收集平台周围的温湿度以及有关波浪的参数。

3）结构安全监测系统的功能是验证监测系统性能，收集近岛礁的实测参数为后续的验证研究提供支撑。

4）腐蚀建监测系统用于配合海上试验平台进行腐蚀防护的试验验证，检验腐蚀防护设计的合理性，为保障平台的安全运行提供可靠的腐蚀监测数据。

5）防波提测量系统用于配合试验平台进行防波堤的海上试验验证，进一步检验所设计的防波堤构型和消浪机理的合理性，掌握浮式防波堤运动响应与消浪特点，为保障演示验证平台的安全运行提供可靠的环境测量数据。

6）波浪能发电性能测试系统主要用于实时监测浮式防波堤波浪能发电装置内外部运行环境参数，采集发电装置电功率参数，为保障波浪能装置及其配套设备的安全运行提供可靠的环境测量数据。

通过以上 6 个子系统的详细介绍可以发现每一个子系统都需要将大量的参数数据进行监控显示，为了方便集中化地监控查看从而需要设计一套数据采集系统可以全面详细的显示、存储参数数据，在结合以往的工程应用的基础上，采用基于 Actor 模型的数据采集系统以达到所需设计要求。

2 基于 Actor 模型的系统设计

本文采用 Akka. Net 开源库来编写实现浮式保障平台数据采集系统中的 Actor 模型。Actor 是封装状态和行为的对象，它的基本元素是演员（Actor）和消息（message）[9]，它们只是通过交换放入收件人邮箱的消息进行通信[10]，并且具有严格的等级制度，从而达到其监督特性。因此在创建之前需要初始化一个 Actor 系统，用来层层创建和管理各子 Actor。

创建 Actor 模型最核心的是使用开源库中的 Actor 类，它维护着单个 Actor 的状态，其中主要功能为自定义创建子 Actor 的数量、根据消息类型做出不同的响应、选择相同 Actor 系统下其他 Actor 的传递对象。因而我们采用 Prop（配置类，封装自定义类型 Actor 实例所需的所有信息）配方创建 Actor。

对于每一个 Actor 都有其独有的命名，但是它们的类型

只有两种即 UntypedActor 和 ReceiveActor。这两种类型的差异就是对消息处理方式的不同。换种说法，Actor 在 Akka.Net 开源库中主要依赖于类似模式匹配的概念（能够根据其 .Net 类型和/或值有选择性的处理消息），在设计上 UntypedActor 和 ReceiveActor 类型能够分别处理简单和更为复杂的消息匹配。UntypedActor 类通过 onReceive 方法实现消息的接收，其中消息作为参数，这是一种较为简单的方法；而 ReceiveActor 类内部可以为每一种要处理的消息类型提供调用，优点在于处理较为复杂的消息处理并且具有与其他类交互的性能。这两种类的不同决定着在实际应用中采取哪一种类来进行 Actor 的创建。

在创建的每一个 Actor 类中我们可以继而创建新的 Actor，两者之间形成父子关系，这么做的目的就回归到为什么一个 Actor 系统是层级结构的？原因有两个，其一是将任务层层分解分配直至最低级以便一体化处理，也就是通过最小的代码占用最容易的处理。另一个原因是监督特性，层级结构可以将整个 Actor 系统的风险降到最低，当某一层级的某一个 Actor 出现问题，其父 Actor 会接收到消息并关闭这一分支，从而不影响其他的运行，提高整个系统的安全性。

在 Actor 模型中使用 Tell（）方法来发送消息，括号内自定义消息类型和内容从而达到万物皆可传递的特性。与此同时，如何能够实现一对一或者一对多传递等情况十分重要。因而需要 Akka.Net 中的 ActorSelection 类的帮助，通过查找路径的方式可以最直接地与同一系统下任何层级的任何 Actor 进行通信，并且 ActorSelection 类只是获取句柄的方式来完成查找降级代码的占用。

图 2 描述了一个名叫 ActorSystem 的简易 Actor 系统，里面有最顶级即最上层的 Actor，它的名字叫 A，Actor 的类名为 ParentActor，并且创建了子 Actor 取名为 B，Actor 的类名为 ChildActor。在这里要介绍一下 Actor 的重要组成—邮箱。Actor 之间的通信并不是直接发送给 Actor，而是发送给该 Actor 内的邮箱，也可以描述为消息队列，按照先后顺序在 Actor 有时间的情况下一一传递给它来处理。如果要实现向 B 传递消息则需要查找传递路径，根据图中的关系可以得到“akka：//ActorSystem/user/A/B”这一路径就是我们需要的，可以通过路径直接与 B 通信，以这种方式去通信是最快捷有效的方式，只要在同一个 Actor 系统下无论是在哪个分支都可以通信。

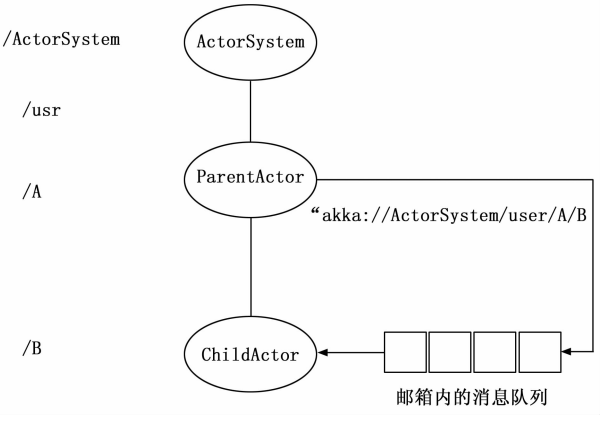

图 2　简易 Actor 系统的创建和消息传递

3　系统的实现

本章将根据上述的系统架构来搭建应用于浮式保障平台的数据采集系统。具体的系统架构如图 3 所示。

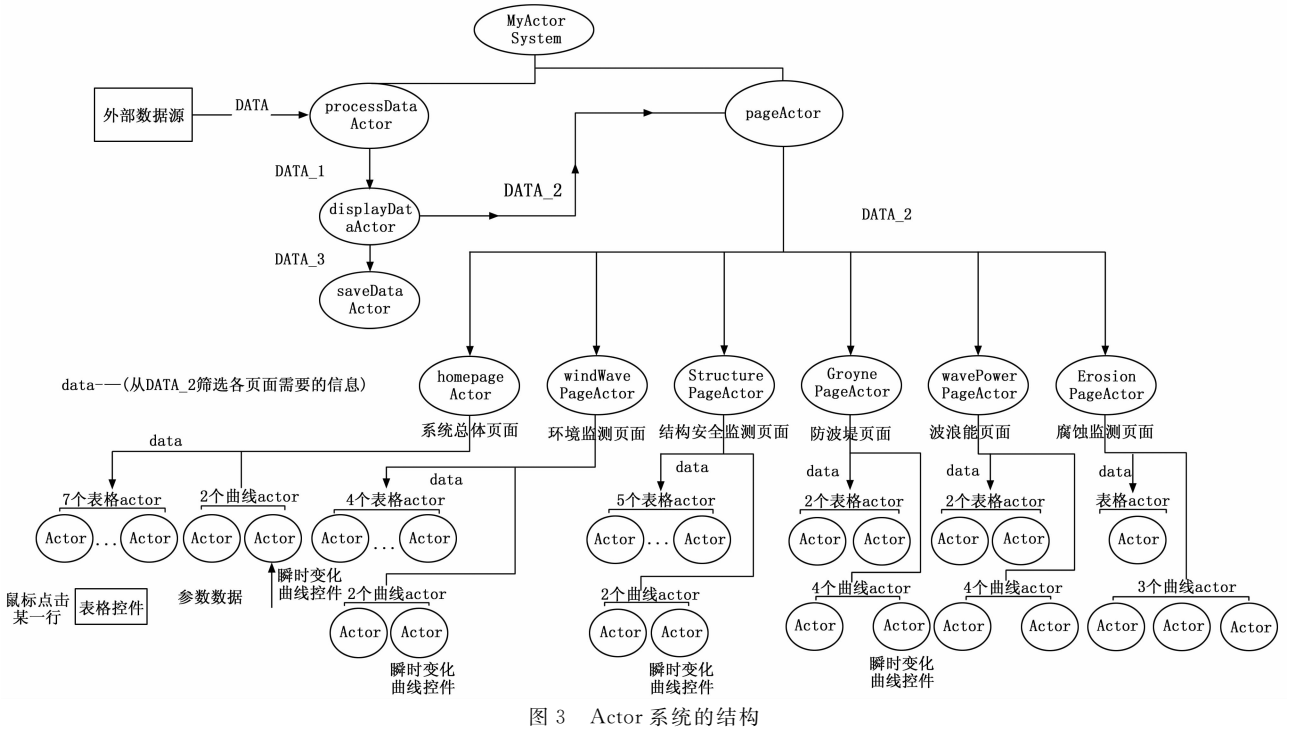

图 3　Actor 系统的结构

和上文所述的系统架构有所不同的是，在实际创建过程中数据采集 Actor 与总体界面 Actor 为兄弟关系，都是根 Actor，但这不影响整个系统，反而可以使层次架构更为清晰，一个专门负责处理最原始的数据，另一个专门用来作为中转和分发站，向所有子系统 Actor 传递数据。

在数据采集 Actor 中有 processDataActor 类和 displayDataActor 类两个父子 Actor 类，先后分别用来处理不同来源的原始数据和将相同子系统的参数数据归类放入一个存放数据容器内，最终将完整的数据参数集通过 Actor 路径传递给总体页面 Actor。

```
_subscriptions = new HashSet<IActorRef>();
IActorRef actor1 = Context.ActorOf(Props.Create(() => new WindWavePageActor()), "windWavePageActor");
IActorRef actor2 = Context.ActorOf(Props.Create(() => new HomePageActor()), "homePageActor");
IActorRef actor3 = Context.ActorOf(Props.Create(() => new StructurePageActor()), "structurePageActor");
IActorRef actor4 = Context.ActorOf(Props.Create(() => new GroynePageActor()), "groynePageActor");
IActorRef actor5 = Context.ActorOf(Props.Create(() => new ErosionPageActor()), "erosionPageActor");
IActorRef actor6 = Context.ActorOf(Props.Create(() => new WavePowerPageActor()), "wavePowerPageActor");
_subscriptions.Add(actor1);
_subscriptions.Add(actor2);
_subscriptions.Add(actor3);
_subscriptions.Add(actor4);
_subscriptions.Add(actor5);
_subscriptions.Add(actor6);
```

pageActor 类作为总体页面 Actor，最主要的任务就是作为消息中转站，不断地将接受到的消息遍历传递给所有的子 Actor。以下为创建子系统 Actor 的过程并且逐一放入 HashSet<T>存储集合中用来一对多形式地传递消息。

在每一个子系统 Actor 内，都将会根据收到的消息来创建控件 Actor、传递参数数据，因而举例详细介绍一下子系统 homePageActor 类及子 Actor 的消息传递过程，并且在命名系统主页子系统 Actor 时名字和类名一样，所以下文所有出现的 homePageActor 即是 Actor 的名字，否则会有特殊说明。homePageActor 与系统主页 homePage 类相互关联，在页面初始化过程中创建的每一个控件都将会给 homePageActor 发送一个自定义的创建消息。其中内容包含譬如控件名称描述的字符串、控件自身的类以及次序数字等，针对这种包含种类多较为复杂的消息类型，在 homePageActor 类中只能自定义 message 消息类，将例如字符串、值以及其他类在消息类中进行实例化。因而只需要将所需要传递的内容一一放入自定义的消息类中，随即将整个消息类的实例作为 message 传递给 homePageActor，就能实现复杂的通信过程并且进行控件 Actor 的创建。图 4 就是创建控件 Actor 的过程。

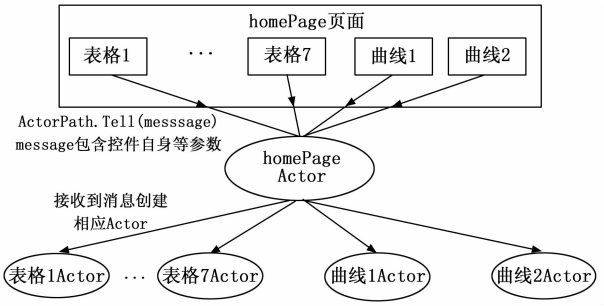

图 4 创建控件 Actor 的过程

在创建完所有的控件 Actor 之后就将进行另一功能的运行。homePageActor 会将接收到的来自父 Actor 也就是 pageActor 发送的消息进行过滤，选取符合自身子系统的参数数据继而进行下一层级 Actor 的传递，从而起到一个中转过滤的作用。在这里需要重点介绍的不是消息的传递，而是一种构成方法，即将 Actor 类与控件类进行交互也就是创建一种将控件类对象作为参数的 Actor 类，而这种构成方法只能在 ReceiveActor 类生效，采用 ReceiveActor 类而不是 UntypedActor 类的优势在于可以对有关数据调用控件类的方法处理并且直接在控件类的参数中赋值修改，减少了许多额外的代码占用。一旦控件 Actor 接收到消息，根据消息类型判断为显示数据时，就能将数据传入控件类的事件进行处理继而存放入控件的动态集合中进行刷新显示最终呈现出来。

图 5 展示了表格控件通过 Actor 显示数据的效果。其中第一列为参数名称，第三列为单位，这两列是固定不变的显示，不会被刷新变化；而第二列是参数数值以及第四列为数值状态描述，从控件 Actor 传递到控件内的数据主要就是刷新这两列的数值显示。在这表格显示的基础上，增加了表格控件和曲线控件的交互，其功能为通过鼠标点击表格的单元格实现曲线控件显示选中单元格内参数数值随时间变化的曲线图。这和表格显示数据的相似之处在于利用 Actor 传递消息的方法来取代 C# 语法中传统的异步委托事件的方法。这两者之间的比较不难看出，使用委托一旦绑定的事件越多，内存占用就越多，对性能的影响越大，在这方面 Actor 模型的并行和分布式特点使得占用内存会少很多，因而在面对高吞吐和低延迟的系统要求下，Actor 模型要更加的出色。

图 5 表格控件效果图

表格和曲线控件的交互流程如图 6 所示。在创建控件 Actor 时，曲线 Actor 将自己的名字通过字符串的消息类型发送给表格 Actor，然后在表格控件内保存有曲线 Actor 的路径参数。当在表格的鼠标点击响应事件激活时，将参数集合作为消息发送给曲线 Actor，同样的原理下，曲线控件接收到数据利用一系列的处理实现显示。

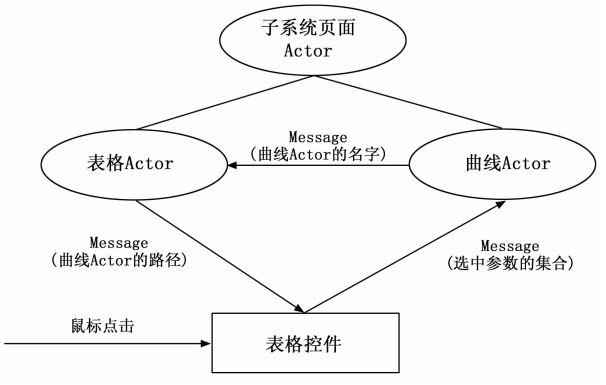

图 6　表格与曲线的交互

4　试验结果与分析

本文所设计的系统最终部署在浮动平台上，根据对各系统所上传数据的频率，实现大部分数据的刷新速率为 1 秒，少部分达到 50 毫秒。经过测试，软件能够正常接收并显示各子系统的实时数据，不存在数据丢失的情况，曲线的绘制效果良好，表格与曲线的交互并无延迟，即点即绘制，与此同时软件的内存占用从原有的 1 G 多降低至 600～700 MB。软件运行效果如图 7 所示。

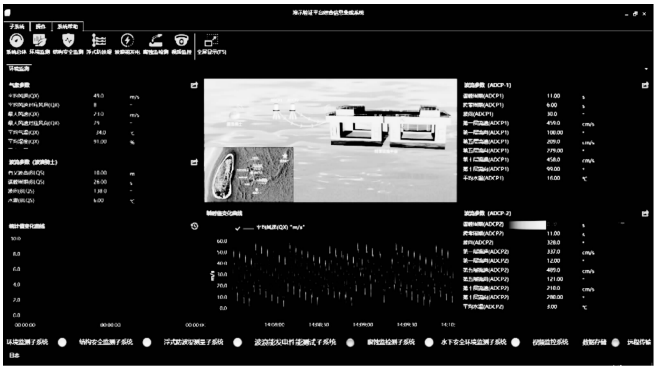

图 7　数据采集系统的显示效果

5　结束语

本文介绍了浮式保障平台项目的内容以及对比各种数据采集系统的优劣特性，结合 Actor 模型以及相关技术，设计并实现了基于 Actor 模型的浮式保障平台数据采集系统，具体的工作内容如下：

1）详细介绍 Actor 模型以及 Akka.NET 开发库中的 Actor 类。

2）详细介绍浮式保障平台数据采集系统的业务架构。

3）详细介绍整体系统的实现过程并展示实现效果。

在系统的实际运行中可以发现采用 Actor 模型能够模块化地建立一条数据采集和传输的通路，最明显的优势在于提高容错率，易于维护，层级化的 Actor 模块之间不会因为一条通信线路的问题导致全部失效。在开发过程中使用 Actor 模型较以往减少了代码占用，并且可以更加便捷地修改和升级，提高开发效率。在利用 Actor 模型作为载体传递不间断的数据时可以发现在高并发和分布式的系统中，Actor 模型之间的传递速度十分优异并且多线程、异步发送消息的特性使得系统真正意义上的成为分布式的系统。通过 Actor 模型的传递，数据参数最终将在表格控件中显示，可以清晰的看到控件内数据值以初始设定的采样速率不断地刷新，达到预计的采集和显示效果。在整个过程中，浮式保障平台数据采集系统稳定高效地完成数据的采集和显示功能，并且相较于传统的委托传值的方法降低了计算机内存的消耗，证明了基于 Actor 模型的数据采集系统的可行性。

Actor 模型现在已经广泛成为高并发事物的首要解决方案，与此同时并发分布式的数据采集系统以往的设计思路并不能很好地解决性能上的缺陷，因而 Actor 模型与数据采集系统的结合是必然。在此次设计与实现中发现系统仍有改进优化的方面，因而未来基于 Actor 模型的数据采集系统的工作可以包括以下几个方面：

1）由于 Actor 模型的消息通信的机制，在万物皆是 Actor 的理念下，有更多类与类之间的交互可以通过 Actor 模型来实现；Actor 具有延时发送的功能，在数据传输的开发中可以根据特定的情况利用这一功能实现连续间隔的发送接收。

2）目前基于 Actor 模型的数据采集系统使用的只是本地的数据传输，由于 Actor 模型具有远程传输的功能模块，可以将这方面的功能结合现有技术开发来优化数据的采集传输，在数据采集系统上会有广阔的前景。

3）Actor 模型随着发展已经具有多种语言的支持，因而在追求高速、低耗和强稳定的性能指标下，将测试基于 c＋＋、Erlang 等不同种语言下的 Actor 的性能，寻求最适用的一种。

参考文献：

[1] 王　晶，黄丽娟. 基于多线程的航空发动机数据采集系统软件设计 [J]. 测控技术，2017 (5)：119－123.

[2] 高　倩. 基于虚拟仪器的风洞数据采集系统软件设计 [D]. 南京：南京理工大学，2007.

[3] 黄善柒. 基于 Actor 模型的数据分发系统的设计与实现 [D]. 北京：北京邮电大学，2016.

[4] Hewitt C，Bishop P，Steiger R. A universal modular actor formalism for artificial intelligence [A]. Proc. of the 3rd International Joint Conference on Artificial Intelligence [C]. Morgan Kaufmann Publishers Inc. 1973：235－245.

[5] Charousset, Dominik, Hiesgen R, et al. CAF－the C＋＋ actor framework for scalable and resource－efficient applications [A]. Proceedings of the 4th International Workshop on Programming based on Actors Agents & Decentralized Control [C]. Portland Oregon, USA. 2014, 15－28.

[6] 周志刚，李晓峰，施小勇. 基于 Actor 模型的高速空泡水筒数据采集与分析系统 [A]. 航空发动机试验与测试工程技术论坛 [C]. 绵阳，2018.

[7] Akka. NET project. What is the actor model [EB/OL]. https://getakka.net/, 2013.

[8] 朱雨晴. 基于流式处理的数据采集系统的设计与实现 [D]. 北京：北京邮电大学，2017.

[9] 李春雷. 基于 Actor 模型的软总线设计与实现 [J]. 计算机工程，2019 (5)：77－83.

[10] 高宇翔. Actor 模型在工业流水巷控制系统的应用 [D]. 广州：华南理工大学，2012.

[11] Cesarini F, Tompson S. Erlang Programming [M]. Sebastopol, CA: O' Reilly Media, Inc, 2009.

[12] Snijders T A B, et al. Introduction to stochastic actor－based models for network dynamics [Z]. Social Networks, 2010.

[13] 詹杭龙. 一种基于 Actor 模型的弹性可伸缩的流处理框架 [J]. 计算机研究与发展，2017 (5)：1086－1096.

[14] 李 侃. Actor 模型的继承机制研究 [J]. 计算机工程与科学，1995 (1)：12－14.

[15] Agha G. Actors: a model of concurrent computation in distributed systems [M]. Technical Report 844, MIT, Cambridge, MA, USA, 1986.

[16] Agha G, Mason I A, Smith S, et al. Towards a Theory of Actor Computation [A]. Proceedings of CONCUR, volume 630 of LNCS [C]. Heidelberg, Springer－Verlag. 1992：565－579.

(上接第 118 页)

[17] Wei L, Cui X N, et al. Blind image deconvolution based on robust stable edge prediction [A]. Proc of International Conference on Biomedical and Health Informatics (BHI) [C]. Las Vegas: IEEE, 2016：66－69.

[18] Zhang Y, Hirakawa K. Blind deblurring and denoising of images corrupted by unidirectional object motion blur and sensor noise [J]. IEEE Trans Image Process, 2016, 25 (9)：4129－4144.

[19] Lei X, Gregson J, et al. Stochastic Blind Motion Deblurring [J]. IEEE Transactions on Image Processing, 2015, 24 (10)：3071－3085.

[20] 吴梦婷. 双框架卷积神经网络用于运动模糊图像盲复原 [D]. 重庆：重庆大学，2018.

[21] Lin H Y, Li K J, Chang C H. Vehicle speed detection from a single motion blurred image [J]. Image and Vision Computing, 2008, 26 (10)：1327－1337.

[22] Bae H, Fowlkes C C, Chou P H. Accurate motion deblurring using camera motion tracking and scene depth [A]. Applications of Computer Vision [C]. IEEE, 2013.

[23] Cannon M. Blind deconvolution of spatially invariant image blurs with phase [J]. IEEE Transactions on Acoustics Speech and Signal Processing, 1976, 24 (1)：58－63.

[24] Lokhande R, Arya K V, Gupta P. Identification of parameters and restoration of motion blurred images [A]. Proc ACM Symposium on Applied Computing [C]. 2006.

[25] 邓泽峰. 图像复原技术研究及应用 [D]. 武汉：华中科技大学，2007.

[26] Deshpande A M, Patnaik S. A novel modified cepstral based technique for blind estimation of motion blur [J]. Light and Electron Optics, 2014, 125 (2)：606－615.

[27] Fahmy M F, Abdel Raheem G M. Fast iterative blind image restoration algorithm [Z]. In NRSC, 2011：1－8.

[28] Kundur D, Hatzinakos D. Blind image deconvolution [J]. IEEE Signal Processing Magazine, 2013, 13 (3)：43－64.

[29] 屈志毅，任志宏，沃 焱. 基于交替迭代和神经网络的盲目图像恢复 [J]. 计算机学报，2000 (04)：410－413.

[30] 张雯雯，韩裕生，黄勤超，等. 基于多尺度卷积稀疏编码的红外图像快速超分辨率 [J]. 计算机辅助设计与图形学学报，2018, 30 (10)：1935－1942.

[31] 孙胜永，耿 志，胡双演，等. 一种改进的 NAS－RIF 红外图像盲复原算法 [J]. 激光与红外，2014, 44 (11)：1268－1273.

[32] Gao W Z, Zou J H, Xu R, et al. An improved NAS－RIF algorithm for image restoration [P]. SPIE/COS Photonics Asia, 2016.

[33] 杨佳煜. 基于改进 NAS－RIF 的 OCT 图像盲复原方法研究 [D]. 北京：北京化工大学，2018.

[34] Rudin L I, Osher S, Fatemi E. Nonlinear total variation based noise removal algorithms [J]. Physica D, 1992, 60 (1/2/3/4)：259－268.

[35] Bredies K, Kunisch K, Pock T. Total generalized variation [J]. SIAM Journal on Imaging Sciences, 2010, 3 (3)：492－526.

[36] Shama M G, Huang T Z, Liu J, et al. A convex total generalized variation regularized model for multiplicative noise and blur removal [J]. Applied Mathematics and Computation, 2016, 276：109－121.

[37] 梅金金. 基于正则化方法的图像复原与融合研究 [D]. 成都：电子科技大学，2017.

[38] Viswanath S, Ghulyani M, De Bec S, et al. Image Restoration by combined order regularization with optimal spatial adaptation [J]. IEEE Transaction on Image Processing, 2020：1－1.

[39] Pan J, Liu R, Su Z, et al. Kernel estimation from salient structure for robust motion deblurring [J]. Signal Processing: Image Communication, 2013, 28 (9)：1156－1170.