# Mercury：启用用于高性能计算的远程过程调用

## 摘要

远程过程调用（RPC）是一种已被分布式服务广泛采用的技术。现在，这种技术越来越多地用于高性能计算（HPC）中，该技术允许将例程的执行委派给远程节点，这些远程节点可以放置在一边并专用于特定任务。但是，现有的RPC框架假定基于**套接字**的网络接口（通常在TCP / IP之上），不适用于HPC系统，因为此**API通常无法很好地**映射到这些系统上使用的原生(native)网络传输，从而导致降低网络性能。另外，现有的RPC框架通常**不支持处理大数据参数**，例如在读或写调用中找到的那些参数。

传统RPC的缺点：

* Socket网络接口效率低
* 不支持处理大数据参数

我们在本文中介绍了一个**异步RPC接口**，称为Mercury，该接口专门设计用于HPC系统。该接口允许参数和执行请求的异步传输，并提供对大数据参数的直接支持。 Mercury是通用的，以便允许传送任何函数调用。此外，网络实现是抽象的，可以轻松移植到未来的系统中，并**有效利用现有的原生(native)传输机制**。

## 引言

在异构环境中工作时，工程师或科学家经常发现**分发**应用程序工作流的各个步骤很有用，特别是在高性能计算中，通常会看到系统或节点嵌入了不同类型的资源和库，专用于特定任务，例如计算，存储或分析和可视化。远程过程调用（RPC）[1]是一种遵循客户端/服务器模型的技术，它允许在远程资源上透明地执行本地调用。它包括将本地功能参数序列化到内存缓冲区中，然后将该缓冲区发送到远程目标，该目标依次反序列化参数并执行相应的函数调用。可以在各种领域中找到实现此技术的库，例如使用Google Protocol Buffers [2]或Facebook Thrift [3]的Web服务或使用GridRPC [4]的网格计算等领域。 RPC也可以通过使用面向对象的方法和诸如CORBA [5]或Java RMI [6]之类的框架来实现，在这些框架中抽象对象和方法可以分布在远程资源中。

但是，在高性能计算（HPC）系统上使用这些标准和通用RPC框架存在两个主要限制：无法利用本机(native 原生？)传输机制来有效地传输数据，因为这些框架主要是在TCP / IP协议之上设计的，无法传输大量数据，因为RPC接口通常限制数据为兆大小级别。此外，即使没有强制执行限制，通常也建议不要通过RPC库传输大量数据，这主要是由于序列化和编码的开销，导致数据在到达远程节点之前被复制了很多次。

我们在本文中介绍了Mercury，这是一种为HPC系统设计的异步RPC接口，可以解决这些限制。 Mercury的主要目的是为需要在分布式环境中远程交换或处理大数据的高级框架（例如I / O转发器、远程存储系统或分析框架）奠定基础。 Mercury主要公开了进行无阻塞RPC以及支持大数据参数所需的语义。 Mercury不直接提供更高级别的功能，例如多线程执行，请求聚合和流水线操作，但可以轻松地在其上构建。

本文的组织如下。在第二部分中，我们首先讨论相关工作。然后，在第三节中，我们将讨论在其上构建接口的网络抽象层以及为有效传输大小数据而定义的体系结构。第四节概述了API，并说明了在使用流水线技术方面的优势。我们还将描述用于接口的网络传输插件的开发以及性能评估结果。第五节提出结论和未来的研究方向。

## 相关工作

RPC已被分布式服务广泛采用，并且网络文件系统（NFS）[7]是在大量数据传输中使用RPC的一个很好的示例，因此在HPC系统上更经常使用RPC。它利用XDR [8]序列化任意数据结构并创建独立于系统的描述。然后将生成的字节流发送到远程资源，该资源可以反序列化并从中获取数据。NFS的最新版本还可以利用单独的传输机制通过RDMA协议传输数据，在这种情况下，数据将在XDR流外部进行处理。我们在本文中介绍的接口遵循相似的原理，但另外直接处理批量数据。它还不限于使用XDR进行数据编码，该编码可能会对性能产生负面影响，尤其是在发送方和接收方共享通用系统体系结构时。通过提供网络抽象层，我们定义的RPC接口使用户能够使用小消息或远程内存访问（RMA）类型的传输来有效地发送小数据和大数据，这些传输完全支持最新HPC上存在的单方面语义系统。此外，该接口是非阻塞的，因此允许使用异步操作模式，从而使调用者不必在发出另一个操作之前等待前一个操作执行。

I / O转发可伸缩性层（IOFSL）[9]是另一个项目，本文提出的部分工作基于此项目。

IOFSL利用RPC专门转发I / O调用。它定义了一个称为ZOIDFS的API，该API在本地对功能参数进行序列化并将其发送到远程服务器，在该服务器中，它们又可以映射到特定于文件系统的I / O操作。扩展IOFSL的主要动机之一是不仅可以发送一组特定的调用（例如通过ZOIDFS定义的调用），还可以发送一组可以动态定义和通用定义的调用。我们注意到，IOFSL构建在并行虚拟文件系统（PVFS）[11]中使用的BMI [10]网络传输层之上。它支持动态连接以及容错能力，还定义了两种消息传递类型，即意外消息和预期消息（在第III-B节中介绍），它们可以启用异步操作模式。尽管如此，BMI的设计受到限制，因为它没有直接公开从客户端内存到服务器内存显式实现RDMA操作所需的RMA语义。

此限制可能是一个问题，并且可能会对性能产生负面影响（第III-B节中介绍了使用RMA方法的主要优点）。另外，尽管BMI不提供单方面的操作，但它确实提供了一套相对高级的网络操作。这使得将BMI移植到新的网络传输中（例如Cray Gemini互连[12]）不简单且耗时，因为在我们的上下文中实现RPC仅需要BMI提供的功能的子集。

…

## 架构

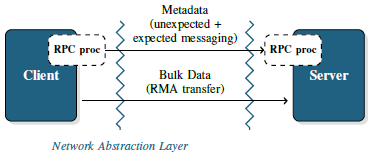
Mercury的接口依赖于三个主要组件：**网络抽象层**、能够以通用方式处理调用的**RPC接口**；以及一个**大数据接口**，该接口是RPC层的补充，旨在通过抽象内存段轻松地传输大量数据。在本节中，我们介绍了总体体系结构及其每个组件。

### 概述

RPC接口遵循客户端/服务器体系结构。如图1所示，根据与调用相关的数据大小，发出远程呼叫会导致不同的步骤。我们区分两种类型的传输：包含**典型函数参数**的传输（通常较小，称为元数据metadata）和描述**大量数据**的函数参数的参数（称为批量数据bulk data）。

通过该接口发送的每个RPC调用都会将函数参数序列化到一个内存缓冲区中（取决于内部连接方式，其大小通常限制为1 KB），然后使用网络抽象层接口将其发送到服务器。

**关键要求之一是在传输的任何阶段都限制存储副本**，尤其是在传输大量数据时。因此，如果发送的数据很小，则使用**小消息**对其进行序列化和发送；否则，将在相同的小消息内将要**传输的存储区域的描述**发送到服务器，然后服务器可以开始提取数据（如果数据是远程调用的输入）或推送数据（如果数据是远程调用的输出）。限制发送给服务器的初始RPC请求的大小有助于提高可伸缩性，因为如果大量客户端同时访问同一服务器，它将避免不必要的服务器资源消耗。根据所需的控制程度，所有这些步骤都可以由Mercury透明处理或直接暴露给用户。



**图1**：架构概述：双方都使用RPC处理器对通过接口发送的参数进行序列化和反序列化。调用具有相对较小参数的函数会导致使用网络抽象层公开的**短(小)消息**传递机制，而包含较大数据参数的函数还会使用RMA机制。

### 网络抽象层

网络抽象层的主要目的是对暴露给用户的网络协议进行抽象，从而允许**通过插件系统集成多种传输方式**。为此，提供了一种轻量级的接口，这样只需付出合理的努力即可实现新的插件。该接口本身定义了三种主要的数据传输机制：非预期的消息传递、预期的消息传递和远程内存访问；但是需要额外的设置才能在客户端和服务器之间建立连接（根据所使用的基础网络实现，动态连接可能并不总是可行的）。

非预期和预期的消息传递仅限于短(小)消息的传输，并且使用了**双向**方法。出于性能原因，最大消息大小由内部的具体连接方式决定，并且可以小到只有几千字节。**非预期消息**传递的概念在其他通信协议（例如BMI [10]）中使用，通过网络抽象层直接发送非预期消息即可，而不需要先通告（发布post）匹配的接收。通过使用此机制，客户端可以不被阻止。并且服务器可以在每次发出非预期信息的接收时，提取**已发布（post）的**新消息。

预期消息和非预期消息之间的另一个区别是，非预期消息可以从任何远程源到达，而预期消息则需要知道远程源。（有点像UDP和TCP之间的区别）

RMA接口允许访问远程内存块（连续和不连续的）。在大多数单面接口和RDMA协议中，必须先将内存**注册**到网络接口控制器（NIC），然后才能使用它。在网络抽象层中定义接口的目的是创建第一级抽象，并定义与大多数RMA协议兼容的API。将**内存段**注册到NIC通常会导致创建一个包含虚拟地址信息的句柄。必须先将创建的本地句柄传达给远程节点，然后该节点才能开始放置put或获取get操作。网络抽象负责确保可以对这些内存句柄进行**序列化**并在网络上进行传输。一旦交换了句柄，就可以发起无阻塞的put或get操作。在大多数互连方式上，put和get将映射到互连网络提供的put和get操作API。网络抽象接口的目的是允许在仅支持双面消息传递方法的网络协议（例如TCP / IP）之上模拟单面传输。

有了此网络抽象层，即可轻松移植Mercury以支持新的互连。网络抽象提供的相对有限的功能（例如，没有大小限制的双向消息）可确保接近本机的性能。

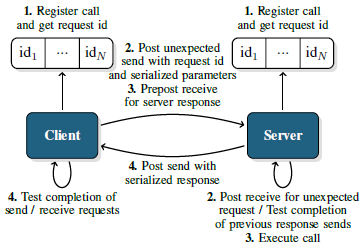
### RPC接口和元数据

只涉及少量数据的调用将使用第III-B节中定义的非预期/预期消息传递。但是，在更高级别上，向服务器发送函数调用意味着客户端在开始发送信息之前必须知道如何对输入参数进行编码，并且一旦从服务器接收到响应，就必须知道如何对输出参数进行解码。

在服务器端，服务器还必须知道收到RPC请求时要执行的操作以及如何对输入和输出参数进行解码和编码。描述函数调用和编码/解码参数的框架是我们接口操作的关键。

一个重要的标准是支持一组可以以通用方式发送到服务器的函数调用，从而避免了硬编码（hard-coded）例程集的限制。**通用框架**在图2中进行了描述。在初始化阶段，客户端和服务器通过使用**唯一函数名称**来注册编码和解码函数，每个函数名称映射到客户端和服务器共享的唯一ID。当通过函数调用接收到操作ID时，服务器还将注册需要执行的**回调**。

* 要发送不涉及批量数据传输的函数调用，客户端会将输入参数以及该操作的ID编码到缓冲区中，然后使用非阻塞的非预期消息传递协议将其发送到服务器。为了确保完全异步，客户端还预发布了用于接收来自服务器的响应的**内存缓冲区**。出于效率和资源消耗的原因，这些消息的大小通常限制在几千字节。
* 但是，如果元数据超出非预期消息的大小，则客户端将需要在单独的消息中传输元数据，透明地使用第III-D节中描述的批量数据接口，以便将其他元数据公开给服务器。



**图2**：RPC调用的异步执行流程。接收缓冲区是预先发布的，允许客户端在执行调用并将响应发送回时完成其他工作（即异步）。

当服务器接收到新的请求ID时，它将查找相应的回调（即远程调用本身吧）、对输入参数进行解码、执行函数调用、对输出参数进行编码，然后开始将响应发送回客户端。将响应发送回客户端也是无阻塞的。因此，在接收到新的函数调用时，服务器还可以测试test响应请求列表以检查其完成情况，并在操作完成时释放相应的资源。一旦客户端知道已经接收到响应（使用wait等待/test测试调用），即远程已完成了此函数调用，则客户端可以解码输出参数并释放用于传输的资源。

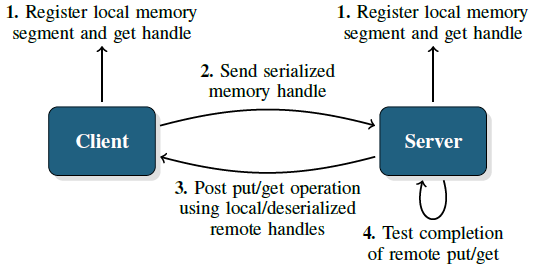
### 批量数据接口Bulk Data Interface

除了先前的接口之外，某些函数调用可能还需要传输大量数据。对于这些函数调用，使用**批量数据接口**，该接口建立在网络抽象层中定义的RMA协议的基础上。只有RPC服务器会启动单面传输，以便它可以控制数据流，以防止其内存受到并发访问。

如图3所示，批量数据传输接口使用一种**单面通信方法**。 RPC客户端通过创建批量数据描述符（其中包含虚拟内存地址信息，要公开的内存区域的大小，以及其他取决于基础网络实现的参数）。然后，可以将批量数据描述符序列化，并与RPC请求参数一起发送到RPC服务器（使用第III-C节中定义的RPC接口）。服务器解码输入参数时，将反序列化批量数据描述符，并获取必须传输的内存缓冲区的大小。

对于**使用大量数据参数**的RPC请求，RPC服务器可以分配一个需要接收的数据大小的缓冲区，通过创建批量数据块描述符来公开其本地内存区域，并启动在该内存区域上的异步read/get操作。然后，RPC服务器将等待/测试(waits/tests)操作是否完成，并在完全接收到数据后（或如果执行调用支持则部分接收）执行调用。然后（调用完成后）将响应（即调用结果）发送回RPC客户端，并释放内存句柄。

对于**产生大量数据参数**的RPC请求，RPC服务器可以分配一个将要产生的数据大小的缓冲区，通过创建大容量数据块描述符公开内存区域，执行调用，然后启动对已公开的客户端内存区域的异步write/put操作。然后，RPC服务器可以等待/测试操作是否完成，并将响应（即调用结果）发送回RPC客户端。然后可以释放内存句柄。



**图3**：执行RPC时需要将大量数据发送到服务器的其他步骤。使用传统的RPC机制，将客户端内存句柄序列化并与其他参数一起发送到服务器。

通过此过程传输数据对于用户可能是**透明**的，特别是因为RPC接口还可以将内存句柄与其他参数一起进行序列化/反序列化。**当必须传输不连续的内存段时，此功能特别重要。**

在这两种情况下，内存段都在RPC客户端上注册，并通过创建的内存句柄进行抽象。然后将内存句柄与RPC函数的参数一起序列化，而使用不连续的内存区域传输大数据将导致与上述相同的过程。但是，这一次内存句柄可能具有可变大小，因为基础网络实现可能不直接支持非连续内存段的注册。在特殊情况下，如果句柄太大而无法使用意外的消息传递接口进行交换，则Mercury可以在内部创建一个**批量数据描述符**，然后使用批量数据接口来交换该句柄。

## 评估

（参考概要文档）

## 结论与未来工作

在本文中，我们介绍了Mercury框架。 Mercury专为在高性能计算环境中提供RPC服务而设计。Mercury建立在一个易于移植的小型网络抽象层上，可提供与现代HPC网络环境功能紧密匹配的操作。与大多数其他RPC框架不同，Mercury提供直接支持来处理包含大数据参数的远程调用。 Mercury的网络协议旨在扩展到成千上万的客户。我们通过实现一个远程写入功能（包括对大数据参数的流水线操作）来展示该框架的功能。随后，我们评估了我们在两个不同的HPC系统上的实施情况，显示了单客户端性能和多客户端可伸缩性。

借助Mercury提供的高性能、可移植的通用RPC功能，可以通过用Mercury调用替换内部的硬编码IOFSL代码来简化和现代化IOFSL。由于在其上构建了Mercury的网络抽象层已经支持使用BMI进行网络连接，因此继续支持IOFSL的现有部署，同时利用Mercury的网络协议的改进的可伸缩性和性能。当前，Mercury不提供取消正在进行的RPC调用的支持。取消对于节点或网络可能发生故障的环境中的弹性很重要，将来的工作将包括对此的支持。（目前应该已经支持了）

尽管Mercury已经支持有效执行RPC调用所需的所有功能，但可以进一步减少每个调用所需的用户代码量。 Mercury的未来版本将提供一组预处理器宏，通过自动生成尽可能多的样板代码来减少用户的工作量。（应该也已经支持了）

网络抽象层当前具有BMI，MPI 2和MPI 3的插件。但是，由于MPI RMA功能很难在客户端/服务器环境中使用[24]，因此我们打算添加对InfiniBand，Cray Gemini / Aries和IBM Blue Gene / Q网络的本地支持。