2000, 17(11)
1-3

Ninf：全局计算通信库

伍　红，黄传河，刘晓明，江　贝
(武汉大学 计算机技术科学学院，湖北 武汉 430072)

摘　要：介绍了一个高性能的通信库：Ninf。Ninf是一个正在不断发展中的全局网络计算基本设施工程。它允许用户获取包括硬件、软件和科学数据在内的计算资源。这些资源分布于广大网络区域中。当Ninf远程库在远程Ninf服务器上可执行时，计算资源可被共享。用户可用Ninf远程过程调用(RPC)调用库建立一个应用。为了位置透明性和网络并行性，Ninf元服务器(MetaServer)保存计算服务器和数据库的全局资源信息，为达到全局负载平衡分配和调度粗粒度计算。

关键词：全局计算；远程过程调用；编程接口；高性能计算
中图分类号：TP393　　文献标识码：A　　文章编号：1001-3695(2000)11-0001-03

1　引言

最近几年，两个重要的趋势：全局计算和集群计算，已经吸引了很多人在高性能网络计算方面的兴趣。

计算机网络技术的显著发展，使通过国际互联网获取各种信息服务成为可能。大多数现存的全球网络服务，例如E_mail、文件检索和WWW，被限制在共享数据资源。只要提供一个计算环境去共享计算资源，包括CPU和磁盘存储器，就能更好地利用整个网络。千兆信息高速公路的到来，使世界范围内全局计算资源能利用大量的计算机。为了全局网络计算，库和系统如Legion，NetSolve，远程计算系统(RCS)已经被提出。

作为科学计算在世界范围内全局计算的基础设施，Ninf的研究正在进行。Ninf是基于网络的高性能计算信息库。Ninf的目标是为全局科学计算提供一个平台，计算资源分布在世界范围内的全球网络。基本Ninf系统支持基于客户-服务器的计算。在远程计算宿主机上的远程库可通过全局网络被程序员的客户端程序调用时，可得到计算资源。程序用现有的语言像Fortran、C或C++编写，使用Ninf远程过程调用(RPC)。程序员可以建成一个全局计算系统，使用Ninf远程库作为系统的一部分，不需要知道复杂的和混乱的网络编程。

2　Ninf：基于网络全球全局计算基本设施的信息库

2.1　Ninf 概述

基本Ninf系统应用于客户-服务器模式。服务器和客户端可通过局域网或国际互联网相连。机器可以是异构的：通讯中数据被翻译成通用网络数据格式。

Ninf服务器进程在Ninf计算服务宿主机上运行。Ninf远程库由网络桩(Stub)例程作为主例程的可执行程序组成，Ninf远程库被服务器进程管理。我们称这些可执行程序为Ninf可执行程序。当库被一个客户程序调用时，Ninf服务器搜索与这个名字有关的Ninf可执行程序，执行找到的可执行程序，建立与客户端恰当的通信。Stub例程处理Ninf服务器和客户端的通信，包括参数列举。可执行程序可用任何现有科学语言编写，如Fortran、C等，只要它能在宿主机上执行。

Ninf的优点如下：

(1)客户可以在多个远程高性能计算机上执行程序中大部分耗时的部分，例如向量超级计算机和MPPs(大规模并行处理)，客户端不需要任何专门的硬件或操作系统。如果这个超级计算机可通过高速网络访问，应用程序自然运行得相当快。它也提供了对各种超级计算机的统一存取访问。

(2)Ninf编程接口极易使用，外观上很像现有语言Fortran、C和C++。用户不需要知道网络编程知识，就可以调用远程库，很容易使用流行的数据库(如：LAPACK)转换已有的应用。

(3)Ninf RPC可以是异步、自动的。对于并行应用来说，一组Ninf元服务器(MetaServer)在网络上以分布的方式保存Ninf服务器的信息，为达到负载平衡自动的在恰当的服务器上动态分配远程库调用，并为网络并行性分配多个调用。Ninf提供了一个事务系统，Ninf调用中的数据相关性自动被元服务器检测和调度。Ninf元服务器可以被看作网络代理，它根据客户端请求和网络资源状态分配恰当的服务器。

(4)Ninf网络数据库服务器提供科学计算中所需的精确常量数据库查询，例如：物理和化学中重要常数。这样做，用户再不需从已印出的图表中输入准确的常数，也再也不怕输错。而且，目前正在研制一个框架，用户在计算中，可以直接通过网络引用研究者实验得到的报表数据值。

通过开发远程超级计算机服务器的能力和网络并行性，Ninf可以加速对应用的服务。这对快速获取某

收稿日期：2000-06-07

个特定数学函数的结果和值，是一个方便的工具。已经设计了WWW接口，被称为NinfCalc。它作为前台，登录在计算服务器上的Ninf库可被分类、浏览和通过Web浏览器访问。

从用户的观点，Ninf通过全局网络提供另一种共享资源的方式。现在已有各种已投入使用的网络基本设施和工具。例如，我们可以使用已有的文件传输服务(如：ftp)去远程访问数值库，程序在本地机上编译，并在远程库执行。除了本地有一台超级计算机，在这种情况下，Ninf很有优势。安全性和所有权问题自然是一个明显的主题；另一个实际的问题是在异构计算环境中获取、编译和维护代码。Ninf通过集中力量在计算服务器上维护库，而不是在每台机器上维护库，来减少维护代价。因此，甚至在没有速度优势可言的慢网络上，Ninf仍有优势。

2.2 编程接口

Ninf_call( )是对Ninf计算和数据库服务器的单用户接口。用一个例子来说明编程接口，考虑用C编写的简单矩阵乘法例程，接口如下：

```
double A[N][N], B[N][N], C[N][N];
/* 声明 */....
dmmul(N,A,B,C);
/* 调用矩阵乘法，C=A*B */
```

当Ninf服务器上的dmmul例程可执行时，客户程序使用Ninf_call调用远程库，用下面的方式：

```
Ninf_call ("dmmul", A, B, C);
/* 调用远程Ninf库 */
```

这里，dmmul是库名，它是Ninf登录在服务器上的可执行文件，A，B，C，N是参数。从这可以看出，客户端用户只需像本地函数调用那样指定函数名。Ninf_call( )自动确定函数参数的个数和每个参数的类型，恰当地排列这些参数，远程调用服务器，获得返回值，把返回值放到恰当的参数中，返回给客户端。用这种方式，Ninf RPC给用户一个错觉，好像参数被客户端和服务器端共享。Ninf服务器的物理地址被保存在环境变量或安装文件中。

为了实现这种简单的客户端编程接口，设计了Ninf RPC。客户端函数调用可以从服务器上获得关于正在运行的被调用库函数的所有接口信息。如图1所示，客户端函数调用给Ninf服务器发调用函数接口信息的请求，Ninf服务器轮流返回已登录的Ninf可执行程序的接口信息给客户端。使用这些信息，Ninf RPC自动完成参数排列和生成发送数据给服务器(或从服务器上接收数据)的序列。客户端库产生中断，根据提供的信息在堆栈中重排这些参数。对可变大小的矩阵参数，IDL(接口描述语言)必须指定一个表达式，包括输入的标量参数，于是矩阵的大小可被计算。在上面的例子中，客户端函数调用发送输入矩阵A和B，它们的大小被参数N计算。Ninf服务器触发dmmul库的Ninf可执行程序，并提交输入数据。

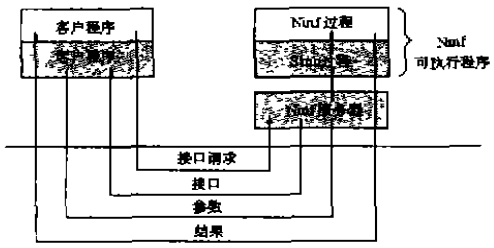

图1 Ninf RPC

这种设计不同于传统的RPCs，传统的RPCs在客户端编译时产生Stub程序。作为动态接口请求的结果，Ninf RPC不需这种编译时间，从而减少用户维护代码的时间。Nakada等人详细描述了接口信息的结构和协议。虽然这将花费额外的网络循环时间，但是典型的科学应用都是计算和数据密集型的，上面的费用显得微不足道。

Ninf RPC也可以异步运行，用以开发网络并行性。我们可以发请求给Ninf服务器，然后继续另一个计算，以后再询问请求的结果，也可以对不同服务器发多个RPC请求。因为这个原因，异步Ninf RPC是并行编程中的一个重要特性。

因为Ninf客户端编程接口对语言无依赖性，所以Ninf客户程序可用各种编程语言编写。通常很容易设计一个对Ninf的客户端接口，只要这种语言支持对C的标准外部函数接口。有人已经用C、Fortran、Java和Lisp设计和实现了客户端Ninf_call函数。客户端和远程库可以用完全不同的语言编写。

2.3 Ninf IDL(接口描述语言)

一个库提供者，给整个网络提供数值库和计算资源，可通过以下步骤产生Ninf可执行程序：(1)用Ninf IDL(接口描述语言)编写每个库函数必需的接口描述；(2)运行Ninf IDL编译器，产生必需的头文件和Stub程序代码；(3)用编写库的语言编译器编译库，然后(4)链接Ninf RPC库，最后(5)从运行宿主机登录到Ninf服务器。这些步骤之后，在网络上就可以通过Ninf RPC，用透明的方式使用这些库。一些已有的库，如：LAPACK，已经用这种方式Ninf化了。

例如：第2.2节矩阵乘法的接口描述如下：

```
Define dmmul(long mode_in int n,
 mode_in double A[n][n],
 mode_in double B[n][n],
 mode_out double C[n][n])
 "..描述 "
Required "libxxx.o"
/* 包含这个例程的库 */
Calls "C" dmmul (n, A, B, C);
/* 使用C调用惯例 */
```

mode_in和mode_out用于指明参数是读还是写。为了指明每个参数的大小，另一个mode_in参数用来形成大小的表达式。在这个例子中，n用来计算矩阵参数A，B，C的大小。既然，Ninf系统被设计成数值应用，Ninf IDL支持的数据类型正是为了这个目的。如：数据类型被限制为标量和那些多维矩阵。Ninf接口生成器编译接口描述，为接口信息中描述的每个库函数生成Stub程序。接口生成器也自动输出一个生成文件，通过链接Stub程序和库函数，建立Ninf可执行程序。

2.4 Ninf元服务器

Ninf元服务器是一个代理，收集一系列关于Ninf服务器的网络信息，自动或半自动地帮助客户选择合适的Ninf服务器。图2表明了元服务器的结构。整个Ninf基本设施包括一系列服务器和元服务器，它们分布于一个网络中。元服务器定期互相交换信息。例如：当一个新Ninf服务器加入元服务器时，新服务器的地址在元服务器网络中广播。当一个元服务器没有特定信息可用时(如：使用某个库)，在整个元服务器网络进行查询，结果被有问题的元服务器得到，它轮流为客户委托信息或自动作决定。

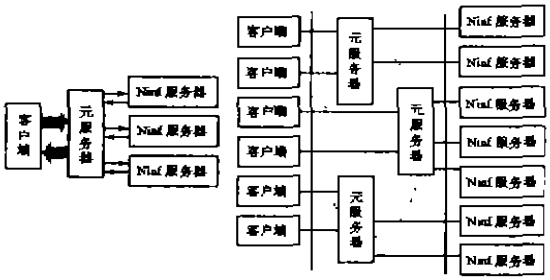

图2 Ninf元服务器和元服务器网络

Ninf元服务器跟踪每个Ninf服务器的如下信息：(1)位置(地址和端口数)；(2)服务器上已登录函数的列表；(3)(考虑通信带宽)到服务器的距离；(4)机器的计算能力(时钟周期，Flop/s等)；(5)服务器的状态，包括负载平均值。

目前的元服务器是用Java实现的，是为了开发它的可移植性、网络编程和多线程。速度并不是绝对因素。

Ninf元服务器通过收集状态的元信息和服务器的可用资源，缓和上述情况，也使Ninf服务器地址对用户透明。例如：如果许多用户共享同一服务器，其他用户的计算请求将大大地影响计算时间。这些动态信息不断用分布的方式探察和授权给其它元服务器。

为了支持元服务器的并行编程，客户端通过Ninf_call_async()异步提交多个请求。每个未完成的请求被元服务器分配给最好的服务器。客户端可以继续它的本地计算，用Ninf_call_wait函数等待结果的返回。当客户端异步执行Ninf_call，用户可提交请求给元服务器，元服务器依次选择恰当的服务器，然后委托请求。用户可以直接调用元服务器去找到最好的服务器执行被请求的库，而不是直接调用服务器。这样，整个网络多个服务器就有很好的负载平衡。

并行编程的另一个编程接口是Ninf并行事务。并行事务是一个被Ninf_check_in( )和Ninf_check_out()包起来的程序区。当Ninf_check_in( )被执行时，Ninf调用并不自发提交实际的请求，但是调用信息被保存，代表远程计算数据相关性的数据流程图被建立。当执行到Ninf_check_out( )时，数据流程图发送给Ninf元服务器。元服务器根据流程图和所给的信息，在多个Ninf服务器上调度计算，有效地执行它们。

3 总结

本文展现了高级网络计算中的一个重要的通信库：广域网上全局计算的Ninf。Ninf方案正在不断发展完善。在实际的应用中，将获得更多的经验。通过广泛使用，系统变得更成熟。目前，在日本Tsukuba中子技术实验室，正在运行一组Ninf服务器。是以Cray J-90，Sun UltraSparcs和一个32处理器DEC Alpha工作站集群为平台。在这些服务器上，通过定义Ninf IDL和登录Ninf服务器，加载了已有的数学库。库包括线性代数包，如LAPACK，Cray's LibSci和其它专门的数学库。物理和化学的科学常数数据库正在设计和建造中。

参考文献：

[1] P Arbenz, W Gander, M Oettli The Remote Computational System[J]. in: High-Performance Computation and Network. Lecture Notes in Computer Science, Springer Berlin, 1996, 1067: 662-667.

[2] N J Boden, D Cohen, R E Felderman, A E Kulawik, C L Seitz, J N Seizovic, W -K Su. Myrinet - A Gigabit-per-Second Local-Area Network[J]. IEEE MICRO 1995, 15(1): 29-36.

[3] H Casanova, J Dongarra. NetSolve: A Network Server for Solving Computational Science Problems[R]. Technical Report, University of Tennessee, 1996.

[4] A S Grimshaw, W A Wulf, J C French, A C Weaver, P F Reynolds Jr. Legion: The Next Logical Step toward a Nationwide Virtual Computer[R]. Technical Report, University of Virginia, 1994.

[5] A Hori, H Tezuka, Y Ishikawa, N Soda, H Konaka, M Maeda Implementation of Gang-scheduling on Workstation Cluster[J] in: D G Feitelson, L Rudolph (Eds.), IPPS'96 Workshop on Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, Springer, Berlin, 1996, 1162: 76-83.

[6] Y Ishikawa. Multi Thread Template Library-MPC++ version 2.0 Level 0 Document[R]. Technical Report TR-96012, RWC, 1996.

[7] Y Ishikawa, A Hori, H Tezuka, M Matsuda, H Konaka, M Maeda, T Tomokiyo, J Nolte. MPC++[J]. in: G V Wilson, Paul Lu (Eds.), Parallel Programming Using C++, MIT press. Cambrige, 1996. 429-464.