基于消息传递的跨编程语言异构数据计算架构研究

Research on Heterogeneous Data Computing Architecture Across Programming Languages Based on Message Passing

王建如[1]，张　强[1]，程新洲[2]，韩　斌[1]，陈丰强[1]，杨启娇[1]，贾玉玮[2]（1. 中国联通青海分公司，青海 西宁 810000；2. 中国联通研究院，北京 100048）

Wang Jianru[1]，Zhang Qiang[1]，Cheng Xinzhou[2]，Han Bin[1]，Chen Fengqiang[1]，Yang Qijiao[1]，Jia Yuwei[2]（1. China Unicom Qinghai Branch，Xining 810000，China；2. China Unicom Research Institute，Beijing 100048，China）

摘　要：

当前各大公司在数据采集、加工、分析领域仍有数据融合、数据拉通的安全保障要求，通常采用分布式计算的方式来解决，但仍然面临多方数据规范不统一、技术栈差异较大等问题。聚焦异构数据分布式计算框架研究，提出了一种基于消息传递的跨编程语言的异构数据计算架构，并在实际的数据环境中进行模拟测试，对实际应用效果进行验证，为后续再生产环境中的推广应用提供了全面的研究论述。

关键词：

数字化运营；分布式计算；异构数据计算

doi：10.12045/j.issn.1007-3043.2023.06.005

文章编号：1007-3043（2023）06-0025-05

中图分类号：TP391

文献标识码：A

开放科学（资源服务）标识码（OSID）：

Abstract：

At present，major companies still have security requirements for data fusion and data pulling through in the fields of data acquisition，processing，analysis，the common method is to solve by distributed computing，but they still have to face issues such as inconsistent data specifications from multiple parties and significant differences in technology stacks. Focusing on the research of heterogeneous data distributed computing framework，it proposes a cross-programming language heterogeneous data computing framework based on message passing，and carries out simulation tests in the actual data environment to verify the practical application effect，which provids a comprehensive research discussion for the subsequent promotion and application in the reproduction environment.

Keywords：

Digital operations；Distributed computing；Heterogeneous data computing

引用格式：王建如，张强，程新洲，等. 基于消息传递的跨编程语言异构数据计算架构研究[J]. 邮电设计技术，2023（6）：25-29.

1 概述

随着信息技术和互联网产业的蓬勃发展，传统的单机计算很难满足逐渐增大的计算能力，在这种背景下，无数的计算机从业者投身其中，提出了许多优秀的并行计算和分布式计算体系架构，在实际应用中最为灵活的一种便是基于消息传递的分布式计算架构。

采用消息传递技术[1]，使用标准输入输出流进行信息收发，可提供稳定、可靠的跨编程语言的分布式计算服务。消息传递架构主要以通信为核心，各个计算节点大多相互独立，拥有独立的内存地址空间，各节点之间通过网络收发数据进行数据共享[2]。最常见的是基于消息传递接口（MPI）的并发编程模型框架[3]。消息传递架构主要依赖网络通信（暂时不讨论单机的进程通信式并行计算），可连通同一网络内大量的计算节点，非常适用于大规模集群式的分布式计算。

目前的分布式计算框架过于依赖编程语言。虽然MPI在设计之初考虑到了跨编程语言、跨平台，但是

收稿日期：2023-04-07

从目前已有的成熟框架来看，不同的框架均有自己特定的编程语言。以MPICH为例，只可使用C/C++或Fortran进行编程[4]。对于交叉学科的研究人员来说，一旦选定了框架，即选定了编程语言，无形中增加了他们的学习成本。针对一些专业性强的小众编程语言而言，以NCL（The NCAR Command Language，是一种专门为科学数据处理以及数据可视化设计的高级语言，很适合用在气象数据的处理和可视化上[5]）为例，它在气象学和地理学中有着非凡的意义，但是其并不支持分布式计算，也没有相关的分布式计算框架。因此，想要加速一个NCL程序只有2种解决方案，一是使用其他支持分布式计算的语言重构代码，二是增强单机的性能。这在一定程度上增加了科研人员改进的难度。所以，一个能实现真正跨编程语言的分布式并行计算框架显得非常重要。

2 技术架构设计

采用Go语言作为核心的开发语言，提供了松耦合的分布式计算服务[6]，各节点之间通过KCP协议（一种可靠的UDP协议）进行协作通信，软件架构的拓扑如图1所示。该架构的核心计算部分为所有的计算节点，控制节点原则上不参与计算，只负责任务下发、结果收集、用户交互等工作。系统将代码编写、计算任务调度等相结合，使用户既可以充分利用所有节点的计算资源，又不用参与繁复的任务下发、调度节点等操作。

其中消息队列系统为整个系统的核心通信枢纽，负责系统内所有消息的识别和转发。

容器引擎[7]作为每个计算任务的运行载体，将各个计算任务之间进行隔离，做到跨编程语言，但语言环境间又互不影响。

控制节点采用前后端分离技术，使用Vue和Ajax技术与后端API接口对接，使用JSON格式进行数据交互。

3 系统整体功能结构

针对本系统的功能需求，系统整体功能结构如图2所示。系统功能模块主要包括控制模块、WebIDE管

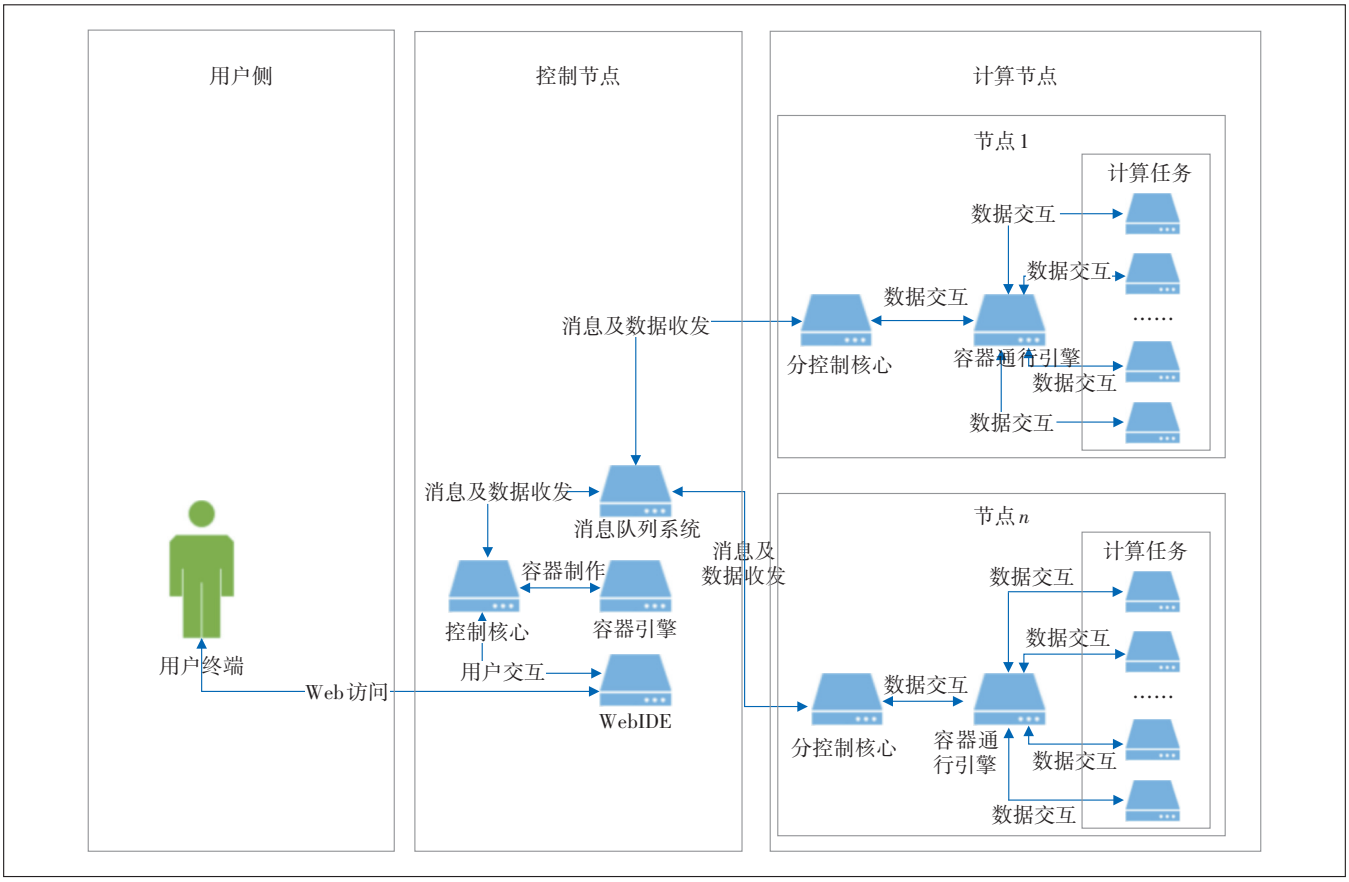

图1　技术架构拓扑图

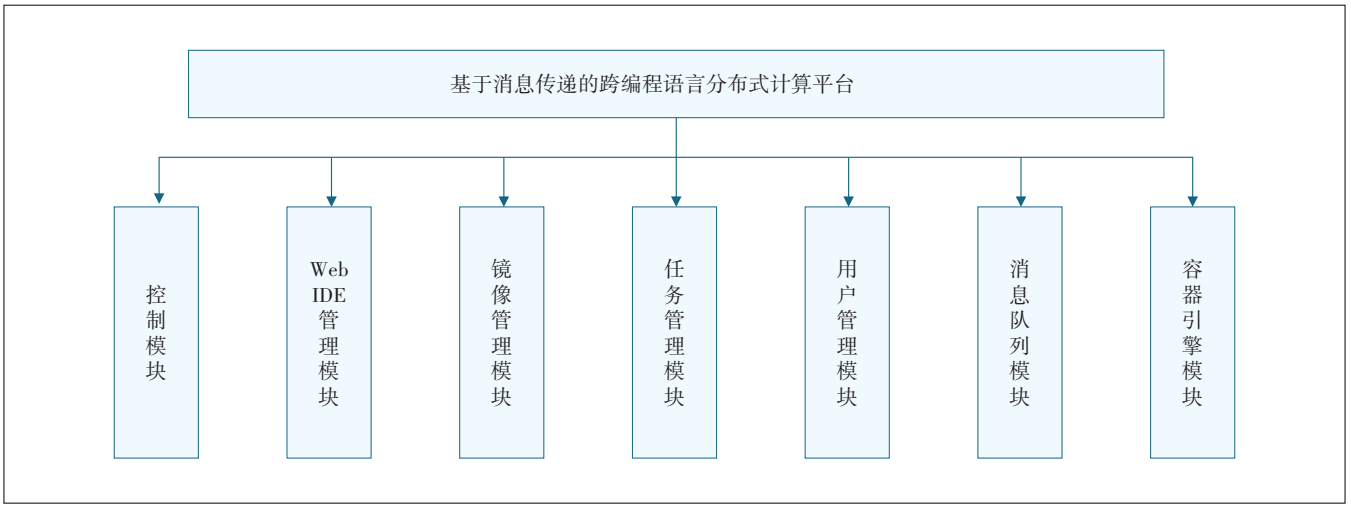

图2　系统整体功能结构

理模块、镜像管理模块、任务管理模块、用户管理模块、消息队列模块和容器引擎模块。

a）控制模块。控制模块作为本系统的核心模块，主要负责连接各个计算节点内的分控制模块，负责监控计算节点的健康状况及其内部任务的实际运行状态，同时负责计算任务下达以及计算结果的收集。

b）WebIDE管理模块。WebIDE模块主要由一个路由转发核心构成，负责将容器内启动的对应code-server（VSCode的Web版）工作端口转发映射给用户浏览器。同时负责WebIDE的新建、删除以及超时未使用回收的工作[8]。

c）镜像管理模块。镜像管理模块主要负责容器所运行的镜像的制作、删除、编辑等操作。

d）任务管理模块。任务管理模块主要为用户提供分布式计算任务的新建、启动、结果收集等操作。

e）用户管理模块。用户管理模块主要针对系统管理员，主要负责用户的新增、删除以及修改密码等操作。

f）消息队列模块。消息队列模块主要负责整个系统内的所有的消息识别、转发工作，同时为整个系统进行解耦。

g）容器引擎模块。容器引擎模块主要负责运行WebIDE及计算任务的镜像，保证权限的隔离及计算环境不冲突。

4 关键技术及系统实现

实现基于消息传递的跨编程语言分布式计算平台的关键技术主要包括高性能的消息传递、超精简的容器引擎、基于输入输出流的数据交互以及镜像的制作与打包。

4.1 高性能的消息传递

消息传递的能力和速度直接决定本系统的实际工作效率，同时消息系统也是节点间进行数据交互的重要组件，消息系统中的消息传递主要有2种模式，分别是点对点模式[9]和发布-订阅模式[10-11]。本系统基于KCP协议，使用点对点模式。通过在内存中开辟一个线程安全的Queue用于存储数据，使用Go语言协程（Go语言在语言层面支持的并发执行的“执行体”，可以使并发编程变得轻巧和安全[12]）将队列内数据进行并行的、批量的识别与转发。得益于KCP协议的低延迟和Go语言的高并发，实现了一个性能优越的消息队列。

4.2 超精简的容器引擎

因为本系统的各个计算任务是跨编程语言的，难免会在运行环境上有些冲突，为解决此问题，就需要一个容器将各个任务的运行环境进行隔离。基于LXC（Linux Container）技术[13]实现了一个超精简的容器引擎，调启容器时，只需要提供文件系统路径及所需执行的命令，即可完成一个容器的启动。

4.3 基于输入输出流的数据交互

对于各个编程语言来说，基础的输入输出功能是标配，本系统正是从此处入手，使用一个Go语言协程配合容器引擎将任务启动，并将stdout和stdin进行重定向，将获取到的输入输出进行识别和分类，转化成不同的消息进行发送。因此用户只需在其代码中按规则进行输入输出即可完成节点间的数据收发。

4.4 镜像的制作与打包

镜像从狭义上说就是容器所运行的文件系统，这就意味着每个镜像中都包含着一个操作系统的所有基础文件。当计算镜像较多时，会造成大量数据冗余。所以本系统采用AUFS（AUFS是轻量级的可堆叠文件系统，其目录是树状的结构，支持将多个文件系统目录的联合挂载[14]），将操作系统基础文件系统和用户独享的可写文件夹联合，打包镜像时只需打包用户可写文件夹并注明依赖的操作系统基础文件系统即可，以提升本系统的工作效率及硬盘利用率。

4.5 系统实现

系统实现基于Go语言搭建Web服务，采用Vue、Ajax、SQLite[15-18]、反向代理、容器、KCP协议等核心技术进行开发。

5 性能测试

本系统为一个分布式计算框架，故采用的测试方法为编写一个计算Pi的程序，将其使用本课题系统进行并行计算，最后计算出加速比及加速效率。

使用Go语言编程，使用Nilakantha级数计算方法，将n置为10 000 000 000。分别编写出了一个串行程序和一个基于系统的并行程序。

测试服务器环境及配置如下。

a）主节点：Intel 2核4G Ubuntu Linux云服务器。

b）计算节点1：Intel 1核2G Ubuntu Linux云服务器。

c）计算节点2：Intel 1核2G Ubuntu Linux云服务器。

d）计算节点3：Intel 1核2G Ubuntu Linux云服务器。

e）计算节点4：Intel 1核2G Ubuntu Linux云服务器。

串行程序计算。为保证计算时的计算机物理计算资源与接下来的并行加速时相一致，依次在计算节点上进行计算，结果见表1。

并行程序计算。按上述计算节点顺序依次增加一个计算节点进行测试，各节点都计算5次取平均值，结果见表2。

对比串行程序及单节点时的平均计算时长，可以看出串行程序略快于单节点计算时间，这里的耗时主要来自于系统自身，即系统工作耗时以及网络延迟。

使用表2中的数据，绘制出了系统对测试程序加速比增长折线图（见图3）。

表1　串行程序计算结果表

指标名称	时间/s
计算节点1计算时间	16.659 199 9
计算节点2计算时间	16.538 167 6
计算节点3计算时间	16.400 484 3
计算节点4计算时间	16.774 903 1
计算节点1计算时间	16.542 998 3
平均计算时间	16.583 150 7

表2　并行程序计算结果表

节点数	Pi值	时间/s	平均时间/s	加速效率
1	3.141 592	17.613 617	17.070 219	–
	3.141 592	16.690 537	–	–
	3.141 592	17.299 034	–	–
	3.141 592	17.053 869	–	–
	3.141 592	16.694 041	–	–
2	3.141 592	9.541 455 3	9.494 501 2	0.90
	3.141 592	9.587 701 6	–	–
	3.141 592	9.504 393 5	–	–
	3.141 592	9.517 662 8	–	–
	3.141 592	9.321 292 9	–	–
3	3.141 592	6.559 034 9	6.409 658 3	0.89
	3.141 592	6.438 477 6	–	–
	3.141 592	6.523 102 9	–	–
	3.141 592	6.251 224 9	–	–
	3.141 592	6.276 451 3	–	–
4	3.141 592	5.088 814 7	5.069 875 2	0.84
	3.141 592	4.953 106 4	–	–
	3.141 592	5.122 770 9	–	–
	3.141 592	5.092 021 5	–	–
	3.141 592	5.092 662 4	–	–

综合上述数据，可以看出本文所提出的系统的实

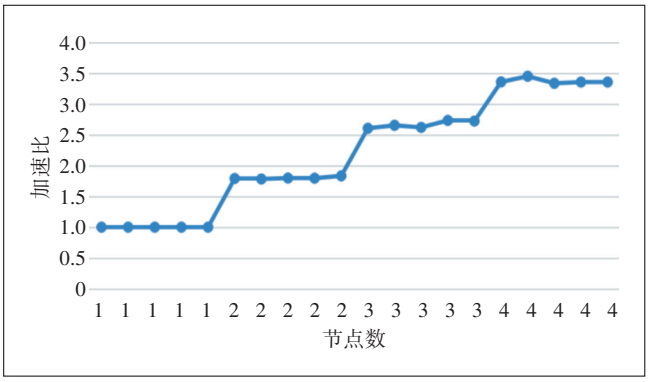

图3　加速比增长折线图

际加速效率均在84%以上，最高可以达到90%。随着更多节点的介入，加速比显著上升。在节点数为3之前，加速比呈线性增加趋势，加速效率呈线性降低趋势。当节点数达到4个时，加速效率出现断崖式下降，加速比的增加幅度也有所降低，故可以预测此时本系统对测试任务的加速效果不明显，此时影响本系统加速效果的主要因素为系统自身，其中包含网络通信耗时、交互引擎耗时、数据序列化及反序列化耗时等系统内部计算耗时。

除上述的表格外，经测试后发现，本系统单节点空载时，完成一次广播、一次接收广播数据以及一次规约操作，耗时约0.121 133 6 s。考虑到节点增加后消息队列SDK中，集中发送消息时的自动延迟技术的影响，本系统增加一个节点预计约增加0.6 s的系统工作时间。

因此可以看出，本系统适用于计算量较大的分布式计算，具有相对较好的加速效率，具有一定的分布式加速能力。

在上述实验中，所有计算节点均使用同一配置的云服务器，但来自不同服务供应商，且不在同一局域网，主节点与计算节点使用公共互联网通信，网络延迟为30~40 ms，上述所有的测试均在此网络延迟下进行。

6 结束语

分布式计算是高性能计算中最为重要的一员，有巨大的发展前景及应用价值。针对现有分布式计算框架过于依赖编程语言的问题，本文设计实现了基于消息传递的跨编程语言分布式计算平台，以牺牲少量计算性能为前提，解决了现有分布式框架无法跨编程语言的问题，同时也为部分使用小众编程语言的交叉学科提供了一种全新的分布式计算解决方案，满足分布式计算场景化应用的需求。数字经济蓬勃兴起，数据要素成为关键驱动力，大数据与各领域融合的广度和深度持续拓展，为进一步促进各类资源要素快速流动和各类市场主体加速融合，基于跨编程语言异构数据计算架构来实现跨领域、跨公司的数据安全交易要求，可以为数字经济可持续发展注入新动力，释放大数据的经济价值和赋能效应。

参考文献：

[1] 郭振，王增福，白向龙，等. 消息传递方法及其在信息融合中的应用[J]. 控制与决策，2022，37(10)：2443-2455.
[2] 权海平. 基于Linux的小型集群的研究与实现[D]. 南京：南京邮电大学，2013.
[3] 范雷，韩奕. 基于MPI的大规模并发数据处理架构设计[J]. 中国新通信，2018，20(15)：49-51.
[4] 郑波祥，殷进勇. 一种面向集群应用的MPICH2编程模型设计[J]. 工业控制计算机，2017，30(11)：9-10，13.
[5] 陈心怡. 大气环境监测预警系统的研究与开发[D]. 南京：东南大学，2018.
[6] 王伯槐，张烨. 基于Go语言的消息推送平台的设计与实现[J]. 数码设计，2017，6(2)：33-36.
[7] 朱小亮. 基于容器引擎的云平台设计与实现[D]. 北京：北京工业大学，2019.
[8] 李孟辉. 基于Web的云开发平台的研究与实现[D]. 成都：电子科技大学，2017.
[9] CLARK M P. Point-to-Point (PTP) and Point-to-Multipoint (PMP) wireless systems and antennas [M]//CLARK M P. Wireless access networks: fixed wireless access and WLL networks—design and operation. Chichester: John Wiley & Sons, Ltd, 2001: 41-56.
[10] EUGSTER P T, FELBER P A, GUERRAOUI R, et al. The many faces of publish/subscribe [J]. ACM Computing Surveys, 2003, 35 (2): 114-131.
[11] GUPTA A, SAHIN O D, AGRAWAL D, et al. Meghdoot: content-based publish/subscribe over P2P networks networks [C]//Middleware 2004. Berlin, Heidelberg: Springer, 2004: 254-273.
[12] 林荣智. GO语言的并发编程介绍[J]. 科技展望，2016，26(22)：12.
[13] 段赫. 基于LXC容器资源优化的研究与实现[D]. 广州：华南理工大学，2016.
[14] 王羿. 基于联合文件系统的异构存储融合[C]//中国新闻技术工作者联合会2017年学术年会论文集(学术论文篇). 重庆：中国新闻技术工作者联合会，2017：136-141.
[15] 陈敬静. SQLite数据库研究与可视化[D]. 南京：南京邮电大学，2020.
[16] 倪天龙，张贤高，王培. 数据库SQLite在嵌入式系统中的应用[J]. 单片机与嵌入式系统应用，2005(10)：3.
[17] 张恒喜，史争军. 基于SQLite的Android数据库编程[J]. 电脑编程技巧与维护，2011(21)：2.
[18] 艾伦. SQLite权威指南[M]. 北京：电子工业出版社，2012.

作者简介：
王建如，工程师，学士，主要从事运营商数据治理、大数据分析工作；张强，高级工程师，硕士，主要从事运营商数据顶层架构设计、大数据分析工作；程新洲，教授级高级工程师，硕士，主要从事大数据、网络智能运营研究与应用工作；韩斌，高级工程师，硕士，主要从事运营商大数据运营工作；陈丰强，学士，主要从事运营商数据治理、大数据分析工作；杨启娇，学士，主要从事运营商数据分析工作；贾玉玮，高级工程师，硕士，主要从事网络感知分析及研究、网络大数据算法模型研发等工作。