doi:10.3772/j.issn.1002-0470.2020.03.005

基于消息总线的高性能计算环境系统软件优化设计与实现①

吴 璨②*** 王小宁③* 肖海力* 和 荣* 赵一宁* 迟学斌***

(*中国科学院计算机网络信息中心 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 针对国家高性能计算环境系统软件在传输大量资源信息时遇到性能瓶颈的问题,本文提出基于消息总线的、支持多数据中心的高性能计算环境系统软件优化结构SCE2.0。SCE2.0 使用 Kafka 作为消息通信中间件,提供异步编程接口,并增加了身份认证、权限管理和双层异地数据备份的可靠传输机制。SCE2.0 并行吞吐量达到51 000 条/s消息,同时可以减少信息传输的时间开销,缩短用户响应时间,降低系统负载,为用户带来更好的体验,实现了高效、高可扩展和高可靠的目标。

关键词 高性能计算环境; 消息总线; 多数据中心; SCE2.0; 资源信息服务

0 引 言

国家高性能计算基础设施建设自 1998 年以来,先后得到“863 计划”、“国家重点研发计划”的持续支持[1]。历经20 多年的发展,我国在高性能计算机领域已走在世界前列。由我国自主研发制造的高性能计算机“天河二号”[2]、“神威·太湖之光”[3]连续5 年10 届(2013 年-2017 年)位居全球超级计算机TOP500[4]排行榜榜首;由中国科学院计算机网络信息中心牵头建设的国家高性能计算环境(原名中国国家网格)[5]整体技术水平处于世界领先行列。截至目前,根据国家高性能计算环境运行管理中心的统计数据[6],国家高性能计算环境已接入来自国家超级计算中心、科研机构和重点高校的19 家中国国家网格结点单位,聚合了“神威·太湖之光”、“神威·蓝光”、“天河二号”、“天河一号”等国际顶级高性能计算资源,聚合计算资源超过200PFLOPS,总存储资源超过167 PB,总内存资源超过104 TB。

国家高性能计算环境的核心软件是由中国科学院计算机网络信息中心自主研发的超级计算环境中间件(super computing environment, SCE)[7,8]。SCE屏蔽了作业管理系统、接入方式、管理制度等的异构性,面向用户提供具有统一访问入口、使用方法和用户技术支持的高性能计算服务[9]。目前所使用的SCE 是1.0 版本,SCE1.0 使用单个数据中心集中处理资源信息,来自全国各地的高性能计算机的资源信息汇报至位于北京的数据中心进行统一管理。虽然 SCE1.0 已稳定运行多年,为用户提供持续的、便捷的高性能计算服务,但是步入大数据时代之后,单个数据中心的系统软件结构遇到了性能瓶颈。SCE1.0 在处理大量资源信息时,时间开销较大、系统负载较高。

随着更多高性能计算资源的接入,高性能计算环境中的用户量不断增加、作业量不断增大,资源信息也越来越多。据国家各超级计算中心统计数据显示,广州超级计算中心“天河二号”的累计用户数已经达到2 700 家,天津超级计算中心的用户数已超过1 600家,无锡超级计算中心的“神威·太湖之光”年有效工作机时达到8 760 h,与日俱增的用户信息、作业信息等大量资源信息传输对高性能计算环

① 国家重点研发计划(2018YFB0204001)和中国科学院青年创新促进会项目(2017216)资助。

② 女,1992 年生,博士生;研究方向:高性能计算,可视化与网格技术;E-mail: wucan@sccas.cn

③ 通信作者,E-mail: wxn@sccas.cn

(收稿日期:2019-04-11)

境的系统软件提出以下挑战:

(1)高性能计算环境系统软件应具备更好的性能、更高的吞吐量,具有 E 级(百亿亿次级)承载能力;

(2)高性能计算环境系统软件在处理分布式环境下的大量资源信息传输时,应具有更小的时间开销、更低的系统负载;

(3)在单个数据中心出现服务器故障、停电、机房搬迁等情况时,高性能计算环境系统软件应该可以正常提供服务。

为了迎接新形势下的更多挑战,满足大数据环境下的更多需求,本文提出基于消息总线的、支持多数据中心的国家高性能计算环境系统软件的优化设计。本文首先介绍了高性能计算环境系统软件优化设计的总体结构;然后对消息总线及基于消息总线开发的资源信息服务的详细设计和具体实现进行介绍;并且从多维度对资源信息服务进行性能评估;最后总结了本文工作,并提出下一步工作的研究方向。

1　总体结构设计

本文提出基于消息总线的、支持多中心的高性能计算环境系统软件优化设计(SCE2.0)。SCE2.0 的层次架构如图 1 所示,是由应用层、服务层、消息总线和资源层组成的 4 层架构。

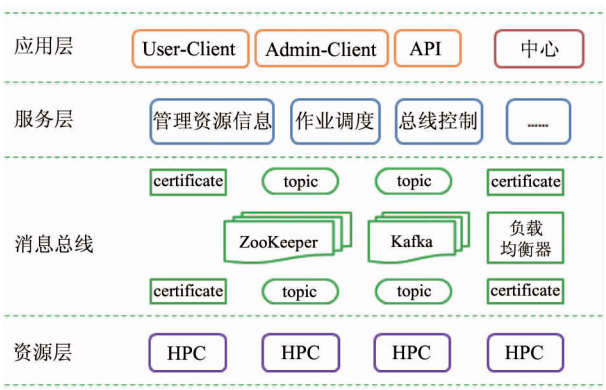


图 1　SCE2.0 层次架构

客户端(User-Client 和 Admin-Client)、应用编程接口(application programming interface,API)和数据中心位于应用层。客户端面向用户提供统一的访问入口,用户可以通过命令行或 Web Portal 使用高性能计算服务。API 面向开发人员提供了访问高性能计算环境的统一访问接口,降低了开发复杂性,缩短了系统开发周期。数据中心存储环境资源信息,为整体环境提供全局的作业视图、信息服务和元调度等。

服务层使用从消息总线获取的资源信息,为应用层提供个性化服务,例如管理资源信息、处理服务请求、作业调度、管理总线等。

消息总线用于分布式环境下的资源信息传输与管理,是支持多个数据中心同时提供服务的核心模块。消息总线将消息中间件 Kafka[10-12] 作为消息传输工具,使用 ZooKeeper[13-15] 管理消息主题(topic)和分布式集群。消息总线使用负载均衡器分配服务请求至不同的数据中心,提高系统的可扩展性;提供异步编程接口,提高信息服务的效率;使用可靠传输机制,对客户端进行权限管理和身份验证,将数据进行异地备份,提高系统的可靠性。

资源层位于最底层,由分布在全国各地的、异构的高性能计算机组成。高性能计算机(high performance computer,HPC)上部署了编译、调试工具及各领域应用软件,所有作业均提交到高性能计算机上执行。

SCE2.0 的资源层和应用层与 SCE1.0 结构相同,本文的工作重点是消息总线和服务层的设计与实现。服务层的各服务设计和实现流程具有相似性,本文仅详细介绍资源信息服务的设计和具体实现。

2　消息总线的设计与实现

2.1　消息主题管理

消息总线以消息主题作为信息处理单元,本文采用树形结构规范消息主题名称。消息主题名称使用 Zookeeper 管理,保证消息主题名称不重复。

环境中的信息按照类别进行区分,每一个类别通过一个消息主题传输数据,消息主题名称存储结构如图 2 所示。环境中的信息可以分为资源信息(resource)、监控信息(monitor)、日志信息(log)和请求信息(request)。每一类信息又可细分为若干小

类。在与消息总线通信时，可以指明具体的消息主题，如“SCE _ resource _ site1 _ hpc1 _ queue”，即收发 site1 的 hpc1 的 queue 消息；也可以仅指明父消息主题，如“SCE _ resource _ site1 _ hpc1”，即收发 site1 的 hpc1 的所有资源信息，包括 queue、job、usermap、app、node、account 和 disk。消息主题有分区数和副本数 2 个属性，分区数表示消息被分为几部分处理；副本数表示消息被复制几份。分区数越多，消息处理效率越高，副本数越多，消息安全系数越高，分区数和副本数受集群中服务器个数限制。

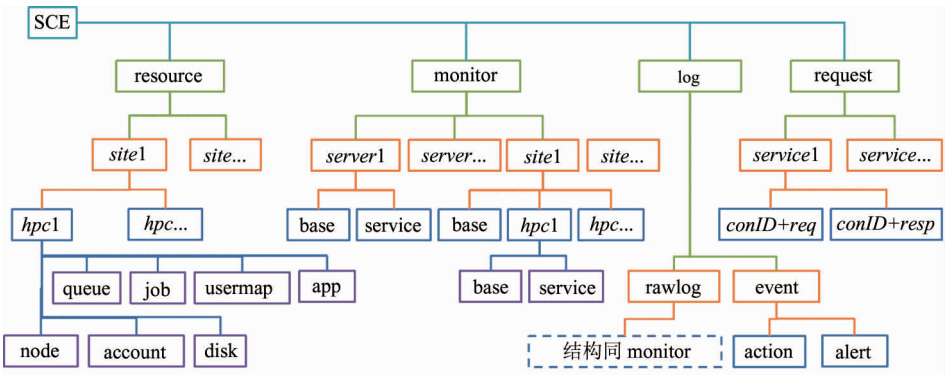


图 2　消息主题名称存储结构

2.2　消息总线接口设计

消息总线提供统一的、简单的异步编程接口，有效地提高了信息服务效率。异步编程接口实现了事件驱动架构（event-driven architecture，EDA）[16]，采用“发布-订阅”模式[17]解除服务逻辑与消息内容的紧耦合关系，使各服务可以异步处理消息。接口设计如表 1 所示。

表 1　异步编程接口

接口名称	参数	备注
put-msg	String msg ArrayList < String > topic	发布消息至一个 topic
put-msgs	String msg String path	发布消息至一组 topic
put-msg-th	int threads String msg ArrayList < String > topic	并行地发布消息
get-msg	ArrayList < String > topic	从一个 topic 接收消息
get-msgs	String path	从一组 topic 接收消息
get-msg-time	long time ArrayList < String > topic	根据时间查找消息
get-msg-th	int threads ArrayList < String > topic	并行地接收消息
add-service	String servicename String confname	扩展新服务
add-site	String sitename String confname	接入新结点
add-balancer	String balancername String confname	接入新的负载均衡策略

由于接口隐藏了访问的复杂性，因此开发新的服务和应用程序更加方便；由于事件屏蔽了消息的细节，开发服务只需针对事件编程，因此开发过程更加快捷。异步编程接口的使用，使新结点、新服务接入国家高性能计算环境时，不需要修改任何程序代码或停止现有的服务，使系统更加高效可用。

2.3　消息总线部署结构

消息总线采用集群部署方式，其部署结构如图 3所示。部署的 ZooKeeper 数量、Kafka 数量由集群规模决定，一般情况下，ZooKeeper 数量为奇数。本文共部署了 3 个 ZooKeeper 和 3 个 Kafka，构成消息总线集群。ZooKeeper 内部通信使用 2888 端口和 3888 端口，对外通信使用 2181 端口。Kafka 对外通信使用 9092 端口。客户端通过访问 2181 端口和 9092 端口与消息总线进行信息传输。

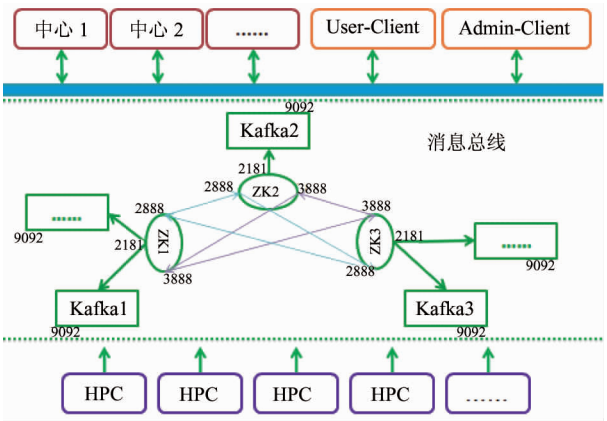


图 3　消息总线部署结构

2.4　可扩展性设计

消息总线的可扩展性包括数据中心可扩展、结点可扩展和服务可扩展。在增加数据中心、接入新结点、开发新服务时,无需修改已有代码或终止当前服务,使用消息总线提供的编程接口和已定义的编程框架即可快速实现。

增加数据中心时,新的数据中心从现有的数据中心获取基础资源信息。新的数据中心随即订阅现有数据中心所订阅的消息主题,以获得相同的消息,保持各数据中心信息一致。

接入新结点时,只需按照结点配置模板设置结点的配置信息。资源信息服务即可自动检测新结点的配置信息,将其增添至资源信息数据库。

开发新服务时,使用消息总线提供的异步编程接口,根据需求从消息总线中获取所需信息。将新服务接入消息总线时,在消息总线中创建新的消息主题即可。由于消息总线异步处理消息,不会影响当前服务。

消息总线的负载均衡策略如图 4 所示。本文在消息总线中构建了负载均衡器来均衡用户请求。用户将作业请求提交到消息总线,并创建一个名为作业 ID 的消息主题,以接收来自数据中心的响应。根据负载均衡算法,所有用户请求都通过负载均衡器分配给不同的数据中心。数据中心接收请求并通过名为作业 ID 的消息主题直接响应用户。用于传输作业请求的消息主题是临时主题。收到响应后,用户客户端将删除消息主题。每个数据中心都可以处理用户请求,因此响应时间将随着数据中心的增加而减少。

消息总线中的负载均衡器是集成不同负载均衡算法的框架,其定义了负载均衡算法接入的输入/输出标准,并提供了接入接口。本项目中的另一个团队正致力于研究负载均衡算法,旨在未来的工作中集成更多高效的负载均衡算法。

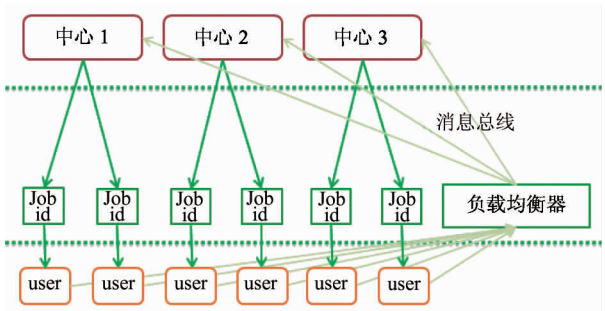


图 4　消息总线的负载均衡策略

2.5　可靠性设计

消息总线从 2 个方面考虑系统的可靠性:系统应足够安全,不允许没有注册的用户非法访问;当服务器发生故障时,数据不会丢失。针对上述 2 种情况,本文提出了可靠传输机制,即身份认证、权限控制和双层异地数据备份,其结构设计如图 5 所示。

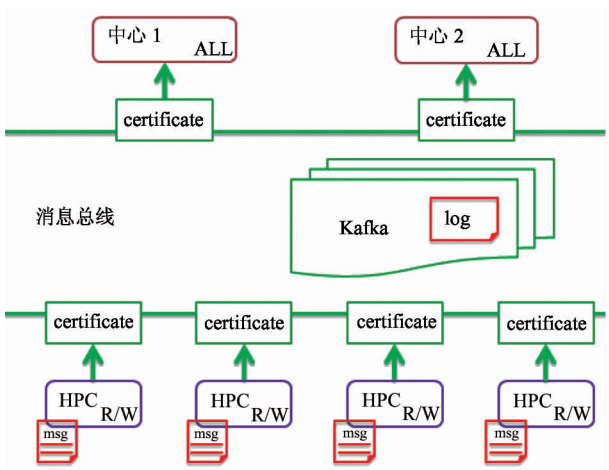


图 5　消息总线的可靠性设计

2.5.1　身份认证

消息总线为每个客户端设置一组用户名和密码,密码由 12 位数字和字母组合构成,存储在客户端的配置文件中。当客户端访问消息总线时,消息总线先验证客户端的身份。如果客户端提供的用户和密码正确,则验证通过;否则,消息总线将拒绝该

访问请求。若一个客户端连续提供了 5 次错误的用户名或密码,系统将冻结该账户并向系统管理员发出报警。

身份验证有效地阻止了非法客户端的恶意连接,从而提高了系统的可靠性。

2.5.2　权限管理

消息总线可以为不同的客户端分配不同的消息主题的权限。消息主题的权限包括 READ、WRITE、DELETE、CREATE、ALTER、DESCRIBE 和 Cluster Action[18]。不同的客户端功能不同,因此为其分配不同的权限,以防止普通客户端更改系统重要信息。消息总线为数据中心分配 READ、WRITE、DELETE、CREATE 和 ALTER 权限,为普通客户端仅分配 READ 和 WRITE 权限。

本文在国家高性能计算环境中定义了一个新角色,称为消息总线管理员。消息总线管理员可以添加、删除消息主题,修改消息主题的分区数、副本数,为客户端分配消息主题的权限等。

消息总线的权限控制限制了普通客户端的权限,减少了普通客户端的误操作对系统产生的影响。

2.5.3　双层异地数据备份

在分布式环境中,有时网络会发生中断或变得不稳定,从而导致高性能计算机、消息总线和数据中心之间的信息传输中断。本文提出了双层异地数据备份机制以应对此类情况,提高系统数据可靠性。

在资源层,使用名为 msg-file 的备份文件来记录需要传输至消息总线的信息。如图 5 所示,从 HPC 提取的资源信息被写入 msg-file 文件中。每个 msg-file 文件有一个 pub-point 属性,用于记录 msg-file 文件中已经发布的最后一条信息的位置。正常情况下,pub-point 会在信息发布完成后更新到 msg-file 文件的最后一行。但是当网络中断或不稳定而导致信息无法成功发布时,pub-point 不更新,新提取的消息写在 pub-point 之后的位置。当网络连接恢复稳定后,pub-point 后面的信息将继续被发送至消息总线。

在消息总线层,使用消息总线的日志(log)来备份数据。设置消息总线的配置,使数据在日志中保存若干天。在此期间,可以根据时间从日志中恢复数据。若出现系统异常,也可以在日志中追溯消息轨迹。消息总线管理员可以通过设置消息主题的副本数来控制数据的异地备份数;可以修改消息总线的配置,设置日志保存的天数。

消息总线的双层异地数据备份机制极大地提高了消息总线的容错能力,当发生网络中断时,数据可在网络连接恢复后自动传输;当数据由于某些客观因素丢失时,可以从集群中的其他数据中心的备份数据中恢复数据。

3　资源信息服务的设计与实现

本文对基于消息总线开发的资源信息服务的设计与实现进行详细介绍。多个数据中心的资源信息服务由聚合信息模块和同步信息模块 2 部分构成。所有数据中心通过聚合信息模块获取从 HPC 提取的资源信息;当一个数据中心的资源信息发生变化时,通过同步信息模块同步到其他数据中心。

资源信息服务的信息传输结构如图 6 所示,由聚合信息模块和同步信息模块组成。聚合信息模块和同步信息模块分别有自己的消息主题“request _ informationservice _ gathered”和“request _ informationservice _ synched”,用于传输不同模块的信息。

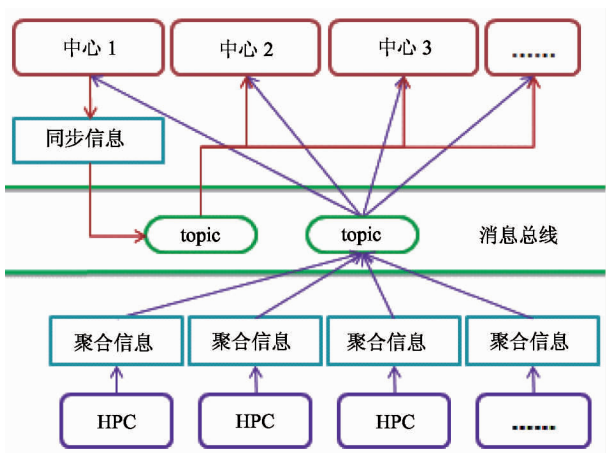


图 6　资源信息传输结构图

3.1　聚合资源信息

各数据中心通过聚合信息模块从资源层的高性能计算机上收集资源信息,包括作业信息、队列信息、用户信息和应用信息。聚合信息模块信息传输

过程如图6中“聚合信息”所示。所有数据中心同时获取消息总线中消息主题名为“request _ informationservice _ gathered”的信息,因此所有数据中心获得的信息是相同的。由于传输时间开销的差异,资源信息到达各数据中心的时间不完全相同,但是,经过一段时间后,各数据中心的数据终会保持一致。

聚合信息模块定期从高性能计算机上提取资源信息。由于高性能计算环境中的资源信息量复杂且庞大,所以从高性能计算机中提取的信息含有大量的冗余信息。为了降低信息传输的时间开销,减少不必要的网络资源浪费,本文定义了各类资源信息的标准格式。聚合信息模块在传输信息之前,将资源信息转换为简单的、标准的信息格式。各类资源信息的标准格式如表2所示。

表2　资源信息标准化格式

信息类型	信息的标准化格式
作业信息	{“JobID”:“*jobid*”,“username”:“*username*”,“status”:“*status*”,“queue”:“*queue*”,“subtime”:“*subtime*”,“starttime”:“*starttime*”,“endtime”:“*endtime*”,“cputime”:“*cputime*”,“mem”:“*mem*”,“swap”:“*swap*”}
队列信息	{“ID”:“*ID*”,“hpcname”:“*hpcname*”,“njobs”:“*njobs*”,“pendjobs”:“*pendjobs*”,“runjobs”:“*runjobs*”,“status”:“*status*”}
应用信息	{“ID”:“*ID*”,“hpcname”:“*hpcname*”,“applicationname”:“*applicationname*”,“version”:“*version*”,“description”:“*description*”}
用户信息	{“username”:“*username*”,“hpcname”:“*hpcname*”,“password”:“*password*”}

此外,高性能计算机上部署的作业管理系统各异,不同的作业管理系统中的作业状态标识符不同。为了将作业状态统一,本文定义了6个作业状态标识符:PEND、RUN、DONE、EXIT、ERROR和HOLD,将LSF[19]、PBS[20]和SLURM[21]作业管理系统中的作业状态标准化。不同作业管理系统中的作业状态与定义的标识符的对应关系如表3所示。

聚合信息模块从高性能计算机上提取资源信息后,与缓存中的历史信息对比,将发生变化的资源信息发送至消息总线中名为“request _ informationservice _ gathered”的消息主题中。所有数据中心从消息总线中获取资源信息,并存储至本地数据库或文件。聚合信息模块可以确保所有数据中心以高效率、低系统负载获得相同的资源信息。

表3　作业状态标识符

作业管理系统	作业状态	标识符
LSF	PEND,PSUSP,WAIT	PEND
	RUN, USUSP, SSUSP	RUN
	DONE	DONE
	EXIT, UNKWN, ZOMBI	EXIT
	*	ERROR
PBS	Q	PEND
	R	RUN
	C, E	DONE
	H	HOLD
	*	ERROR
SLURM	PD, CF	PEND
	R, CG	RUN
	CD	DONE
	CA, F, NF, PR, TO	EXIT
	S	HOLD
	*	ERROR

3.2　同步资源信息

同步信息模块用于将一个数据中心更改后的信息同步到其他数据中心。同步信息模块传输过程如图6中“同步信息”所示。当任务请求提交到某一个数据中心时,此数据中心的资源信息将发生变化,变化的信息应该及时同步至其他数据中心。同步信息模块提取发生变化的资源信息,并将其发送至消息总线中名为“request _ informationservice _ synched”的消息主题中。其他数据中心从该消息主题中获取信息,并更新本地数据库或文件。资源信息的标准格式与聚合信息模块中的格式相同。

每个数据中心维护一个“data-point”,用于记录已经获得的消息的位置。如果某个数据中心与消息总线在一段时间内断开连接,“data-point”将不会更改。网络服务恢复正常后,数据中心可以根据“data-point”继续从消息总线上获取数据。虽然数据一致在时间上有所差异,但是所有数据中心最终会具有相同的资源信息。

3.3 并行传输信息

为了提高消息的发送、接收效率,本文提出并行传输消息的设计方案。信息服务根据实时系统状态,计算可并发的线程数目,并行地处理消息。计算并行发送消息和并行接收消息的并发线程数目的方法是不同的,本文对此两种情况分别进行考虑。

发送消息时,并发线程的数目可以根据系统状态计算。计算机存储设备的输入/输出性能是衡量系统状态的考虑因素之一;线程的数目不应超过系统核数。同时,时间开销与数据量也和系统负载相关,所以数据量和系统负载也是考虑因素。由此,本文根据计算机存储设备的输入/输出性能、系统负载、数据量和系统核数,提出了一个自动计算并发线程数目的公式,公式如下:

$$x = [(1 - \frac{I \times S}{\alpha W}) \times C]$$

其中,x 为并发线程数目;I 是计算机存储设备的输入/输出性能,可以用 Linux 命令“iostat-x”获取,其本质是输入/输出操作的时间占比,取值范围在 0% 和 100% 之间;S 是 1 min 内的系统平均负载,可以用 Linux 命令“uptime”获取,单个 CPU 的取值范围在 0 到 1 之间;W 为工作负载,即消息的大小,以兆字节(MB)为单位;C 是系统核数,可以用 Linux 命令“iostat”获取;α 是调节因子,默认值为 0.0001。

以 40 核的服务器为例,当工作负载为1 000 MB 时,系统 1 min 内的平均负载为 0.1,计算机存储设备的输入/输出操作的时间占比为 10%。所以 C 是 40,S 是 0.1,I 是 0.1,W 是 1000。α 取默认值 0.0001。用上述公式计算,结果为 36。因此,系统自动创建 36 个线程发送信息。

接收消息时,并发线程数目与消息主题的分区数有关。并发线程数目不应超过消息主题的分区数,当并发线程数目与消息主题的分区数相等时,信息传输效率最高。

4 实验结果与分析

4.1 实验数据与实验环境

本文实验中所使用的数据包括真实数据和测试数据。

真实数据是从国家高性能计算环境工作负载库(Chinese Supercomputer Workloads Archive, CSWA)[22]下载的环境结点的公开真实工作负载。包括 4 家单位的工作负载:中国科学技术大学超级计算中心(USTC)的曙光 TC4600 百万亿次超级计算系统的工作负载;上海超级计算中心(SSC)的曙光 5000A 系统的工作负载;国家超级计算无锡中心(WXSC)的高性能计算机“神威·太湖之光”的工作负载;上海交通大学高性能计算中心(SJTU)的高性能计算机“π”的工作负载。各工作负载的详细信息如表 4 所示。

表 4 CSWA 工作负载详细信息

中心名称	服务器名称	TOP500 排名	位置	数据量
USTC	曙光 TC4600		合肥	41 MB
SSC	曙光 5000A	2018 年 11 月第 10 名	上海	55 MB
WXSC	神威·太湖之光	2018 年 11 月第 3 名	无锡	29 MB
SJTU	π	2013 年 6 月第 8 名	上海	98.7 MB

因为 CSWA 中目前只公开了 4 家单位的真实工作负载,数据规模不够大。所以本文在实验过程中,根据真实数据的格式,模拟了固定数据量的测试数据。

实验环境由 5 台服务器搭建,使用 3 台服务器构建消息总线,在其上分别部署 ZooKeeper 集群和 Kafka 集群;使用 1 台服务器作为数据中心;另外 1 台服务器作为高性能计算机。用于传输消息的消息主题有 3 个分区、3 个副本。

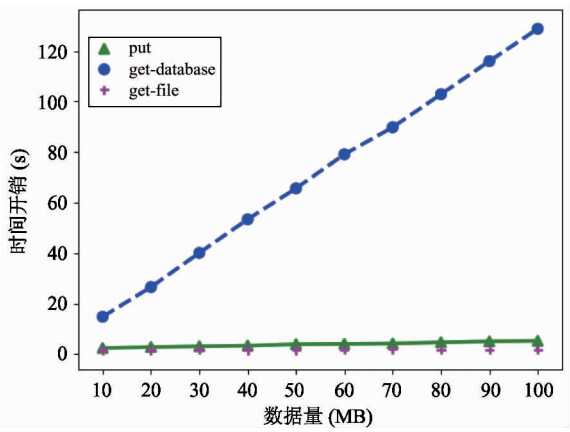
4.2 实验结果

4.2.1 资源信息服务的时间开销和系统负载

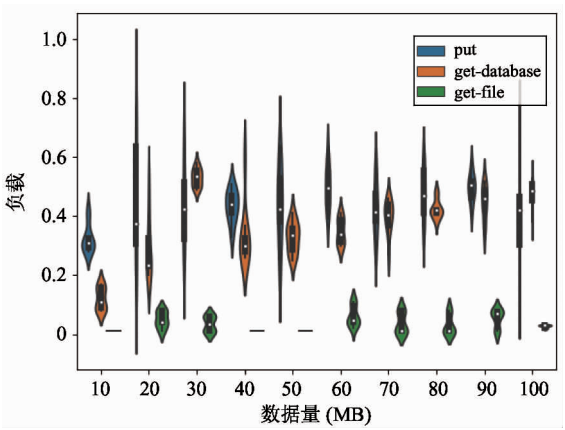
资源信息服务从数据传输的时间开销和所产生的系统负载 2 个维度来进行性能评估。实验中使用数据量分别为 10 MB,20 MB,30 MB,40 MB,50 MB,60 MB,70 MB,80 MB,90 MB,100 MB。聚合资源信息模块从高性能计算机中提取资源信息并放入消息总线记为“put”操作;数据中心从消息总线获取信息,将其存储到数据库记为“get-database”操作;存储至文件记为“get-file”操作。

本实验评估了 SCE2.0 的 put、get-file 和 get-database 3 种操作的时间开销和系统负载。此外,本实验评估了单个数据中心的 SCE1.0 的信息服务的性能,并与其进行对比,分析性能优势。

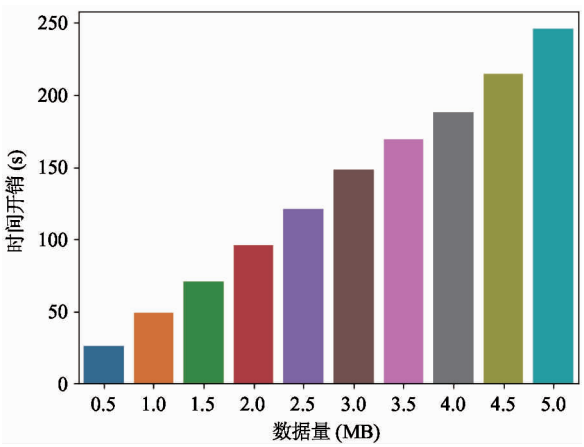
SCE2.0 中 put、get-file 和 get-database 操作的时间开销与数据量的关系如图 7(a)所示。put、get-file 和 get-database 操作的时间开销均随着数据量的增长而增加。当数据量达到 100 MB 时,put 操作时间开销不超过 5 s,get-file 操作的时间开销不超过 2 s,这是极其高效的。由于数据库连接操作的时间开销较大,get-database 操作的时间开销要长得多。因此,当环境中资源信息量巨大时,可以考虑将资源信息存储到文件。



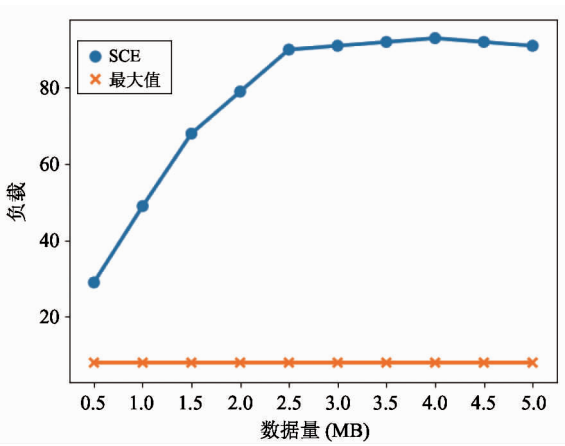
(a) SCE2.0 时间开销



(b) SCE2.0 系统负载



(c) SCE1.0 时间开销



(d) SCE1.0 系统负载

图 7　系统操作的时间开销与系统负载

SCE2.0 中 put、get-file 和 get-database 操作引起的系统负载与数据量的关系如图 7(b)所示。系统 1 min 内的平均负载随着数据量的增加而变化,但不是完全规则的。整体来看,get-file 操作引起的系统负载是最稳定的,并且远低于 put 和 get-database 操作引起的系统负载。实验中所用服务器有 8 核,正常的系统平均负载值不应超过 8。上述实验结果中,put、get-file 和 get-database 操作引起的系统负载均低于 1,远远低于上限。

图 7(c)和图 7(d)分别展示了 SCE1.0 中信息传输的时间开销和系统负载与数据量的关系。当消息大小达到 5 MB 时,时间开销接近 250 s;系统负载大于 90,远远高于上限。SCE2.0 在处理 5 MB 的消息时,时间开销不超过 2 s,系统负载不高于0.5,性能远远好于 SCE1.0。

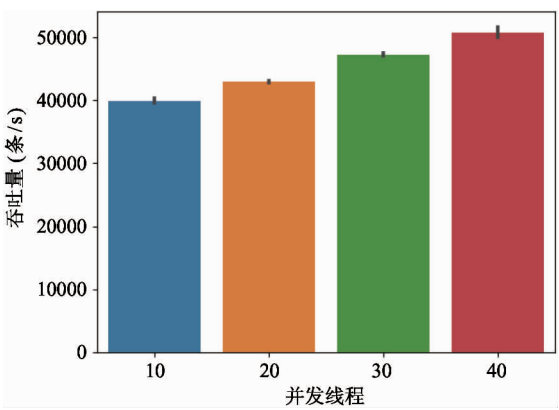
综上所述,单个数据中心的 SCE1.0 在处理数

据量较大的资源信息时，遇到了性能瓶颈。而多个数据中心的 SCE2.0 比 SCE1.0 更加高效，能够轻松处理大量数据。SCE2.0 资源信息服务的较小的时间开销有利于提升用户体验，较低的系统负载可以减轻服务器的压力。

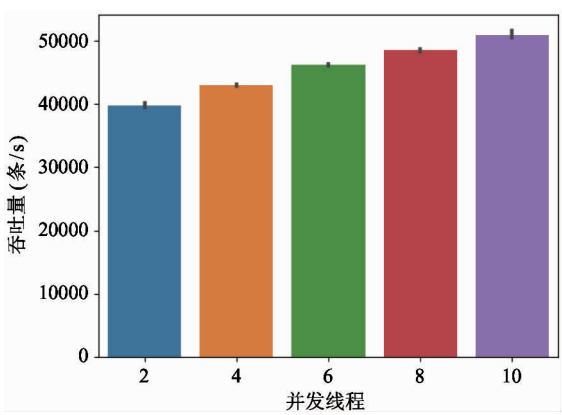
4.2.2　消息总线的吞吐量

本实验对并行 put 和 get-file 操作的吞吐量与线程数目的关系进行评估。执行 put 操作的服务器有 40 核，并且系统中没有其他进程影响系统状态，因此执行 put 操作的最大并发线程数目为 40。实验中所用的消息主题的分区数为 10，因此 get-file 操作的最大并发线程数目为 10。实验使用测试数据，数据量为 1 000 MB。

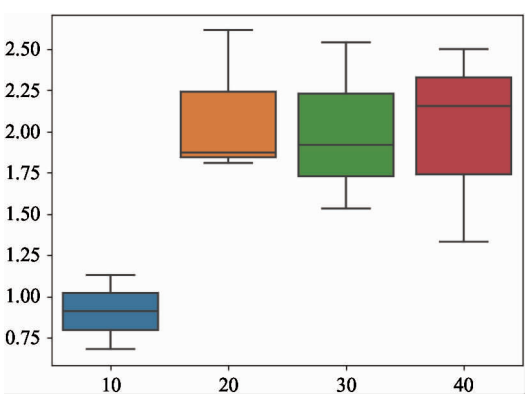
put 和 get-file 操作的吞吐量和系统负载随线程数目的变化情况如图 8 所示。从实验结果可以看出，随着线程数目的增加，put 和 get-file 操作的吞吐量均线性增加。当 put 操作的并发线程数目为 40 时，吞吐量可达 51 581 条/s 信息；当 get-file 操作的并发线程数目为 10 时，吞吐量可达51 564 条/s信息。执行 put 操作的服务器有 40 核，系统负载上限为 40；执行 get-file 操作的服务器有 8 核，系统负载上限为 8。由 put 操作引起的系统负载均小于 3，远低于上限 40；由 get-file 操作引起的系统负载均小于 2，远低于上限 8。由此得出结论，消息总线并行传输信息具有较高的吞吐量，且不引起过高的系统负载。当环境中信息量巨大时，可以使用多线程并发地处理资源信息。



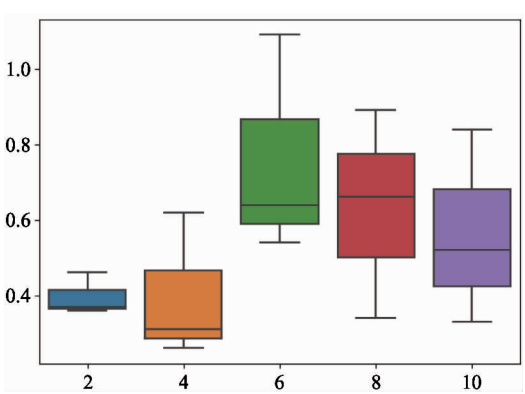
（a）put 吞吐量



（b）get-file 吞吐量



（c）put 系统负载



（d）get-file 系统负载

图 8　put 操作和 get-file 操作的吞吐量与系统负载

4.2.3　消息总线的吞吐量与 Kafka 的吞吐量对比

本实验使用真实数据进行测试，将消息总线的吞吐量和 Kafka 的吞吐量进行对比。

实验结果如图 9 所示。消息总线的吞吐量和

Kafka 的吞吐量非常接近,对 Kafka 的封装并没有降低其性能。

国家重点研发计划“高性能计算”重点专项的总体目标是突破 E 级计算机核心技术,研制满足应用需求的 E 级高性能计算机系统[1]。本文提出的消息总线及基于消息总线开发的资源信息服务具有高吞吐量和低负载处理能力,可以满足 E 级高性能计算机系统对环境的高效处理作业请求、高效传输资源信息的需求。

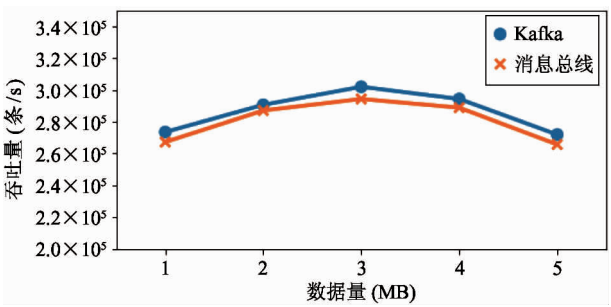


图 9　消息总线和 Kafka 的吞吐量对比

5　结　论

本文提出了基于消息总线的、支持多数据中心的高性能计算环境系统软件的优化设计 SCE2.0;实现了消息总线及基于消息总线开发的资源信息服务;提出了根据实时系统状态自动计算并行线程数目的公式,显著提高了系统的吞吐量;采用“发布-订阅”模式,提供异步编程接口,简化服务开发流程;完成了消息总线及资源信息服务的性能评估实验,实验结果表明,SCE2.0 使用多线程并行处理消息时的吞吐量已达到 51 000 条/s,同时缩短了用户响应时间,降低了系统负载。

基于消息总线的、支持多数据中心的 SCE2.0 打破了现有的单个数据中心的 SCE1.0 的性能瓶颈,实现了高效、高可扩展和高可靠的目标。支持多个数据中心同时提供服务,某个中心出现异常,环境仍可以正常提供服务,具有高可用性。目前,SCE2.0 的资源信息服务已在测试环境中上线。在未来的工作中,资源信息服务将被部署于正式环境进行多个数据中心的测试,测试无误后正式上线使用。同时基于消息总线开发更多服务,逐步替换 SCE1.0。

参考文献

[1] 迟学斌等. 国家高性能计算环境发展报告[M]. 北京: 科学出版社, 2018:17-25

[2] Liao X, Xiao L, Yang C, et al. Milky way-2 supercomputer: system and application[J]. *Frontiers of Computer Science*, 2014, 8(3): 345-356

[3] Fu H, Liao J, Yang J, et al. The sunway Taihu light supercomputer: system and applications[J]. *Science China Information Sciences*, 2016, 59(7):1-16

[4] TOP500. TOP500[EB/OL]. https://www.top500.org/:TOP500.org, 2019

[5] Xu Z, Chi X, Xiao N. High-performance computing environment: a review of twenty years of experiments in China[J]. *National Science Review*, 2016, 3(1):36

[6] 中国国家网格. 中国国家网格[EB/OL]. http://www.cngrid.org/:中国国家网格运行管理中心, 2019

[7] 戴志辉, 肖海力, 曹荣强, 等. 三层架构超级计算环境容错框架[J]. 计算机应用研究, 2011, 28(7): 2576-2579

[8] Dai Z, Wu L, Xiao H, et al. A lightweight grid middleware based on OPENSSH-SCE[C]//International Conference on Grid and Cooperative Computing, Los Alamitos, USA, 2007:387-394

[9] 吴璨, 王小宁, 肖海力, 等. 高性能计算环境中间件的优化设计与实现[J]. 计算机应用研究, 2019, 36(1):169-173

[10] Kreps J, Narkhede N, Rao J. Kafka: a distributed messaging system for log processing[C]//Proceedings of the ACM SIGMOD Workshop on Networking Meets Databases, Athens, Greece, 2011: 1-7

[11] Le Noac'H P, Costan A, Bougé L. A performance evaluation of apache Kafka in support of big data streaming applications[C]//IEEE International Conference on Big Data, Boston, USA, 2017: 4803-4806

[12] Esmaili K S, Esmaili K S. Kafka versus rabbitMQ: a comparative study of two industry reference publish/subscribe implementations: industry paper[C]//ACM International Conference on Distributed and Event-Based Systems, Barcelona, Spain, 2017: 227-238

[13] Junqueira F, Reed B. ZooKeeper: Distributed Process Coordination[M]. Sebastopol: O'Reilly Media, Inc. 2013

[14] Haloi S. Apache ZooKeeper Essentials[M]. Birming-

ham: Packt Publishing Ltd, 2015

[15] Meseguer J, Bobba R B, Skeirik S. Formal analysis of fault-tolerant group key management using ZooKeeper[C]//Proceedings of the 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2013), Delft, Nethelands, 2013:636-641

[16] 季云峰. 基于 SOA 的 EDA 的研究和实现[J]. 软件, 2012(7):74-76

[17] Liu X J, Yue M A, Dong Y U. Analysis and evaluation of real-time performance of publish/subscribe communication mode[J]. *Computer Engineering*, 2010, 36(20): 229-231

[18] 牟大恩. Kafka 入门与实践[M]. 北京: 人民邮电出版社, 2017: 212-215

[19] 弋瑞录, 胡飞, 奚水清, 等. 基于 LSF 集群系统的分布式并行计算[J]. 测控技术, 2006, 25(7):53-55

[20] 童端, 董小社, 李纪云, 等. 基于 OpenPBS 的机群作业管理系统的设计与实现[J]. 计算机工程与应用, 2004, 40(13):123-125

[21] Yoo A B, Jette M A, Grondona M. SLURM: simple Linux utility for resource management[J]. *Lecture Notes in Computer Science*, 2003, 2862(2862):44-60

[22] Shen Yu. 国家高性能计算环境工作负载[EB/OL]. https://git.lug.ustc.edu.cn/yshen/CSWA: Shen Yu, 2019

Design and implement of middleware for high performance computing environment based on message bus

Wu Can[*,**], Wang Xiaoning[*], Xiao Haili[*], He Rong[*], Zhao Yining[*], Chi Xuebin[*,**]

([*]Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190)

([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

In order to resolve the bottleneck occurring when high performance computing environment middleware is transferring massive amount of resource information, this paper proposes an optimized middleware based on message bus called SCE2.0, which is a multi-data center system. SCE2.0 uses Kafka to transfer information, and provides reliable transmission mechanism consisting of identify authentication, authority control and double-layer data backup. The throughput of SCE2.0 has reached 51 000 requests per second, which is extremely efficient. In particular, the multi-data center information service could reduce information transfer time cost, shorten user response time and reach eventual consistency at lower overhead and finally offer users better experience. The multi-data center information service has achieved the goals of efficiency, scalablity and reliablity.

Key words: high performance computing environment, message bus, multi-data center, SCE2.0, information service