计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200300010

ENLHS:一种基于抽样的 Kafka 自适应调优方法

谢文康[1]　樊卫北[1,2,3]　张玉杰[1,2,3]　徐　鹤[1,2,3]　李　鹏[1,2,3]
1 南京邮电大学计算机学院　南京 210023
2 国家高性能计算中心南京分中心　南京 210023
3 江苏省高性能计算与智能处理工程研究中心　南京 210023
(1017041215@njupt.edu.cn)

摘　要　Kafka 应用在生产环境中时,除机器的硬件环境和系统平台影响其性能外,Kafka 自身的配置项决定着其能否在硬件资源有限的情况下达到理想的性能,但人为修改和调优配置项的效率极差。海量数据发送到 Kafka 后,如果不针对实际资源环境进行调优,Kafka 使用默认的配置参数无法保证其在每个生产环境下的性能。因为 Kafka 自身的配置项非常大,传统的自适应算法在大规模生产系统中的性能较差。为了提高 Kafka 的自适应能力,消除系统中的复杂性,获得更好的运行性能,提出一种针对 Kafka 的自适应性能调优方法。该方法充分考虑了 Kafka 特征参数与性能的影响权值,并使用抽样的原理来提高数据集的生成效率并优化数据选取范围,提高建模的效率并降低优化方法的复杂度。实验结果显示,该算法对开源版本 Kafka 的吞吐率和时延进行了优化,提高了 Kafka 在给定的系统资源下的吞吐性能,并降低了时延。

关键词:Kafka;消息队列;性能调优;拉丁超立方抽样;弹性网络
中图法分类号　TP311.5

ENLHS:Sampling Approach to Auto Tuning Kafka Configurations

XIE Wen-kang[1],FAN Wei-bei[1,2,3],ZHANG Yu-jie[1,2,3],XU He[1,2,3] and LI Peng[1,2,3]
1 School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China
2 Nanjing Center of HPC China,Nanjing 210023,China
3 Jiangsu HPC and Intelligent Processing Engineer Research Center,Nanjing 210023,China

Abstract　When Kafka is applied in a production environment,its performance is not only limited by the machine's hardware environment and system platform. Its own configuration items are the key element to judge whether it can achieve the desired performance under the condition of limited hardware resources,but it is manually configured. The efficiency of item modification and tuning is extremely poor. If the actual resource environment is not optimized,Kafka cannot guarantee its performance in each production environment using default configuration parameters. Because Kafka's configuration bound is extremely large,the performance of traditional adaptive algorithms in large-scale production systems is poor. Therefore,in order to improve Kafka's adaptive ability,eliminate complexity in the system,and obtain better operating performance,an adaptive performance tuning method for Kafka is proposed,which fully considers the influence weights of Kafka's characteristic parameters and performance. It uses the principle of sampling to improve the efficiency of data sets generation and optimize the range of data selection,improve the efficiency of modeling and reduce the complexity of optimization methods. Experiments show that the algorithm optimizes the throughput rate and latency of the open source version Kafka,improves Kafka's throughputs under a given system resource,and reduces latency.

Keywords　Kafka,Message queue,Performance tuning,Latin hypercube sampling,Elastic net

1　引言

近年来,随着云计算、大数据分析产业的发展,以 Hadoop[1] 为首的大数据生态及其技术栈迎来了新的发展机遇。Hadoop 大数据生态及其技术栈为大数据产业提供了完整的开源解决方案,任何人都可以从中找到与自身需求相关的技

到稿日期:2020-03-02　返修日期:2020-05-28　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研发计划项目(2018YFB1003201);国家自然科学基金(61672296,61602261,61872196,61872194);江苏省科技支撑计划项目(BE2017166,BE2019740);江苏省高等学校自然科学研究重大项目(18KJA520008);江苏省六大人才高峰高层次人才项目(RJFW-111)
This work was supported by the National Key R&D Program of China(2018YFB1003201),National Natural Science Foundation of China(61672296,61602261,61872196,61872194),Scientific and Technological Support Project of Jiangsu Province(BE2017166,BE2019740),Major Natural Science Research Projects in Colleges and Universities of Jiangsu Province(18KJA520008) and Six Talent Peaks Project of Jiangsu Province(RJFW-111).
通信作者:李鹏(lipeng@njupt.edu.cn)

术栈并根据需要进行二次开发或定制，亦可以使多个框架相互配合形成稳定、完整的系统。

Hadoop 相关的开源框架都托管在 Apache 开源基金会。Apache Kafka[2]作为其中的顶级项目之一，在 Hadoop 生态中扮演着 Pub/Sub 模式[3]的消息中间件系统[4]，用于连接系统中的数据上下游，将传统的一一对应的数据架构进行解耦；数据上下游的应用程序不需要考虑线程分配、IO 阻塞、生产消费对应关系、消息容错等问题[5]，只需要使用 Kafka 提供的 API 与其进行对接即可，保证了系统的高吞吐、高可用以及高容错性。Kafka 自身根据数据管道[6]的流向，抽象出 3 个角色：生产者，消费者，Broker。生产者就是位于数据上游的发送方，发送方将数据以消息的形式发送；Kafka 接收数据，为每条消息分配唯一的偏移量，进行集群间的消息同步，并将消息持久化到本地日志等。消费者是位于 Kafka 数据下游的系统，可以是实时/离线数据处理系统（如 MapReduce/Spark[7]），也可以是数据库系统（如 MySQL[8]/Redis[9]），还可以是分布式文件系统（如 HDFS[10]等）。消费者从 Kafka 批量拉取数据，并实现自己的处理逻辑。Kafka Broker 指 Kafka 分布式集群的服务器节点，每个 Kafka Broker 提供了名为主题（Topic）的逻辑概念。生产者和消费者通过订阅主题分配数据的流向[11]，这也是发布/订阅（Pub/Sub）模式的消息系统的基本原理，如图 1 所示。

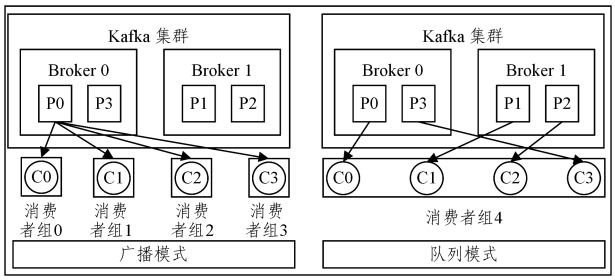

图 1 Kafka 的两种工作模式

Fig. 1 Two modes of Kafka

主题通过分区（Partition）进行具体实现，如图 2 所示。Kafka 的分区器按照一定规则将每条消息分配到不同的分区中，分区中的每条消息会按照时间顺序分配到一个单调递增的顺序编号，即偏移量（Offset）。一条消息可以通过三元组（主题，分区 ID，偏移量）确定。

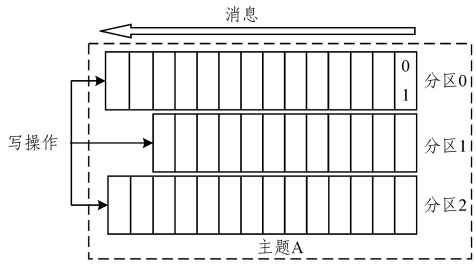

图 2 Kafka Topic 的物理结构

Fig. 2 Physics construct of Kafka Topics

为了能在不同的硬件资源条件下达到比较可观的性能表现，Kafka 提供了可以修改的配置文件，开发者可以根据产品的需要修改配置文件进行性能调优，这是传统的技术栈调优方式。但是 Kafka 本身提供了数百项的特征用于调优，开发人员只能通过自身经验和多次性能测试确定配置项，这将花费大量的时间和精力。

本文针对这一问题，研究并提出一种 Kafka 自适应性能调优方法。该方法采用抽样的机制及 ElasticNet[12]对抽样进行改进，使得 Kafka 可以自动完成性能的优化过程，节省了调优的时间，提高了调优效率。

2 相关工作

Kafka 作为高吞吐、高容错、高可用性的消息系统，被广泛应用于各种业务场景中，如图 3 所示。而 Kafka 在不同的系统资源、数据环境、部署方式下的性能是不一致的，这就需要专业运维专家根据环境的特性修改 Kafka 的配置参数。配置参数代表了 Kafka 工作模式的对外接口，Kafka 提供了多达 200 项的配置参数，给性能调优带来了非常大的困难，并且修改一次配置后需要将集群重启，还需要重新运行数据监控进行性能比对。2018 年，Bao 等将参数配置抽象为分布式消息系统配置（Configuration of Distributed Message System，CDMS）问题[13]。

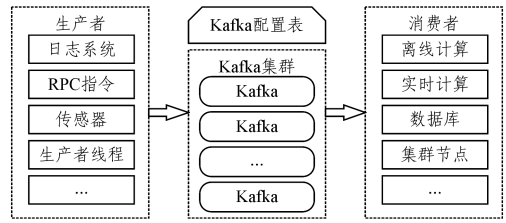

图 3 Kafka 在数据后台的运作位置

Fig. 3 Kafka working in data backend

对于 CDMS 问题，暴力的方法是列出所有配置项的组合结果，并一一部署以比较性能，但很明显这个方法的效率极低。这是因为 200 余个配置项的组合结果太过庞大，并且 Kafka 的配置项并不是单一的布尔型或离散型参数，更多的是连续的数值型参数（如 log. segment. bytes 表示 Kafka 分区中 Segment 的大小，其默认值是 1 073 741 824 字节），这种情况下进行组合结果的取值难度非常大。

对于自适应参数优化，第一种方法是基于应用程序的优化，主要通过性能监控套件对特定的框架测试性能极限，如 SPECjms2007[14]，jms2009-PS[15]，DDSBench[16]等。这种方式的弊端是对于特定的技术栈需要特定的性能测试套件，无法做到一般化；另一个弊端是这种工具往往很难获取测试对象的所有性能指标，指标单一。

第二种方法是基于搜索的优化方法[17]，其将配置问题视为黑盒，使用搜索算法进行解决，搜索的目的是评估不同参数的候选解，并最小化目标函数。搜索策略包括递归随机搜索（Recursion Random Searching，RRS）[18]、启发式算法[19]等。与其他方法相比，搜索的优化方法更为简单、通用[6]，因为其不需要了解应用程序内部的详细信息；但是这种方法必须在每次搜索迭代时重新运行应用程序，在优化大规模生产系统时非常耗时并且不切实际。

第三种方法是基于统计学习[20]的优化方法，其通过已有

的由配置项和性能组成的数据集样本进行学习并训练性能模型。该方法的精确度比较高，但是需要使用大量的训练集，而在特定时间内获取的配置项和性能数据非常少且繁琐。

2016 年，Bei 等提出 RFHOC（Random-Forest Approach to Auto-Tuning Hadoop's Configuration）算法[21]对大数据离线计算框架 MapReduce 进行自适应参数调优，其基本思想是使用随机森林和遗传算法不断学习和迭代优化配置参数。但是其学习模型只采用了 10 项配置参数，在大数据生态框架百余项的配置参数下，其学习性能和迭代性能无法满足大数据环境的要求，并且其需要修改 MapReduce 开源框架的源码，以在集群不重启的情况下使新的配置生效。

2018 年，Bao 等[13]抽象出 CDMS 问题，并提出 AutoConfig 算法，使用抽样的方法从 200 余项的配置项中抽样出 h 个参数，提高了建模的性能。该算法使用 Lasso[22]进行配置项参数的计算，缺点在于没有考虑多个配置项之间的关系，对于两个具有相关性的参数，其通常会考虑选择其一而把另一参数设置为 0。

鉴于此，目前亟需一种高效且更加精确的自适应性能调优算法来解决 CDMS 问题，使得以 Kafka 为代表的消息系统能够根据工程环境的需要对自身进行性能优化，减少人工干预并提高调优效率，在大数据环境下优化众多配置参数的学习和迭代效率。本文提出的基于抽样的 Kafka 自适应性能调优机制，将 ElasticNet 与拉丁超立方抽样[23]相结合，对配置参数和性能的关系建立数学模型，解决了 CDMS 问题，并提高了调优效率和精确度。

3 自适应性能调优算法设计

3.1 问题建模

解决 CDMS 问题的关键是找到一个配置表，使得 Kafka 的吞吐量（尤其是生产者端）在某一环境下达到最大值，其可以抽象为如下 5 个部分。

（1）应用程序，表示某一系统（生产者）需要将消息发送到另一系统（消费者），用 A 表示。应用程序可以 5-元组的形式表示：$A=(l,p,s,c,b)$。其中，l 表示单条消息的长度，单位是 kB；p 代表生产者数量；s 表示生产者的发送方式，Kafka 中将其定义为同步和异步两种模式；c 代表 Kafka Broker 确认接收到消息的方式，包括 Leader 收到就确认消息和 ISR 收到才确认消息，Kafka Broker 为 Kafka 分布式部署的节点服务器，用于存储收到的消息日志和负载均衡，Leader 表示 Kafka 某一分区的备份中的首领节点，ISR 表示数据与首领保持同步的备份节点；b 表示 Kafka Broker 的数量。

（2）运行环境，表示 Kafka 部署平台的物理资源或虚拟物理资源，用 E 表示。我们可以将 E 同样抽象为 5-元组：$E=(cr,f,m,ds,n)$。其中，cr 表示 CPU 的核心数；f 表示 CPU 的频率，单位是 GHz；m 表示机器的运行内存大小，单位是 GB；ds 表示机器硬盘的写入速率，单位为 MB/s，因为目前可购买到的硬盘空间远远大于 Kakfa 的单条数据大小，因此这里假设硬盘的空间是无限的；n 为网络带宽。

（3）配置表，表示 Kafka 运行时的配置项列表，用 C 表示，$C=(c_1,c_2,c_3,\cdots,c_n)$。Kafka 自身配置有 200 余项。

（4）吞吐速率，表示单位时间内 Kafka 收到的数据被拉取的速率，用 TP 表示。对于给定的 A,E,C，其吞吐率可以表示为 $TP(A,E,C)$。

（5）时间限制，表示在给定的时间内，算法将最后的优化结果进行输出，用 T_c 表示。

由此，CDMS 问题的抽象即为：当 A,E 给定时，找到一个 C，使得 $TP(A,E,C)$ 最大。

$$\begin{aligned}&\max\ TP(A,E,C)\\&\text{s.t. } t\leqslant T_c\end{aligned}\tag{1}$$

本文提出针对 Kafka 的自适应性能调优机制，用于解决 CDMS 问题，使用 ElasticNet[12]改进 LHS，在保持传统 LHS 进行抽样时的高效性之外，将权值加入抽样过程中，并加入降维机制，避免 LHS 在进行高维数据抽样时消耗无用的资源。ElasticNet 可以缩小特征参数集的大小，并计算出预测模型，其相对于经典的最小二乘法（Ordinary Least Squares，OLS）、岭回归和 Lasso 算法在当前场景下更加优秀。OLS 的主要问题是预测精度和模型的解释能力不足，岭回归和 Lasso 就是对这两个问题的改进；而 ElasticNet 相对于后两者，不仅考虑到特征与结果的关系，还考虑到特征与特征之间的关系，因而不会出现 Lasso 存在的两个配置项取其一的问题。

3.2 数据预处理

数据预处理包括两个部分：特征筛选和特征数据归一化处理。第 2 节提到 Kafka 自身提供了 200 余项可供调节的特征参数，但是很多特征与性能并不相关，表 1 列举了 Kafka 配置文件内的部分参数、具体功能以及对于性能的重要性等信息。本文提出的 ENLHS 算法在进行迭代时，将计算特征的权值，并进行特征的筛选，但是为了提高训练集生成的性能，可以通过官方定义、开发经验和专家推荐的特征，滤除绝大多数无用的特征。

表 1 Kafkaserver. properties 中的部分配置项

Table 1 Some configurations in Kafka server. properties

名称	定义	对性能的影响程度
broker. id	服务器的唯一 id，用于标识集群中每一台服务器的序号，Zookeeper 通过该参数辨识服务器并保持新加入的 broker 自增	低
listeners	Kafka 监听消息的端口号，是集群与外部应用对接的通道	低
num. network. threads	Kafka 服务器处理网络请求相应的线程数	高
num. partitions	Kafka 的主题默认分区数	高
log. segment. bytes	Segment 文件最大的体积	高
log. dir	log 文件的存放位置	低
message. timestamp. type	定义消息时间戳，可以设置为 CreateTime 或者 LogAppendTime	中
comparison. type	主题数据的压缩方式，包括 gzip，snappy，lz4 这 3 种方式	高

特征筛选首先根据开发经验滤除对性能完全没有意义的特征，如表 1 中的 broker. id 和 log. dir，通过描述可以知道这类特征对性能影响没有意义。经过这一步，剩余的特征集合仍然非常大，初步统计仍然包含 150 余项，这是因为其中有很多对性能影响相对较小的特征，因此特征筛选的第二步是根据官方定义和技术支持文档，将对性能影响程度特别小、对本

文研究没有意义的特征进行排除。经过这一步筛选出来的部分特征如表 2 所列。

表 2 筛选后的 Kafka 部分特征

Fig. 2 Some configurationsof Kafka after filter

名称	定义	对性能的影响程度
max. block. ms	当缓存满时 Kafka 生产者控制发送线程的阻塞时间	中
buffer. memory	Producer 的缓存大小，在 Kafka 进行阻塞控制时将阻塞消息存放到 buffer 中	高
batch. size	Producer 在进行批量发送时每一批消息的大小	高
timeout. ms	Kafka 确认消息的最大等待时长	中

经过两步的特征筛选，已经将特征缩减到数十项，大大优化了后续模型建立和迭代计算的时间，下一步将进行数据归一化。对于生成的训练集，可以计算训练集的均值 μ、方差 δ，从而使用式(1)进行归一化处理，使其满足(0,1)上的正态分布。

$$x'=\frac{x-\mu}{\delta^2} \tag{2}$$

经过上述预处理操作后，可以使实验数据转换为后续算法可直接处理的类型，满足后续计算的要求。

3.3 算法设计

本文提出的 ENLHS 算法主要分为 3 部分。第 1 部分是训练集生成，该部分使用拉丁超立方抽样(Latin Hypercube Sampling，LHS)生成一定的配置项组合数据，并将这一配置运行到 Kafka 上使其生效，获取 TP，得到初始的训练集。第 2 部分是在 T_c 允许的计算时间内进行性能模型的训练和数据的迭代，这一部分使用 ElasticNet 改进 LHS，建立配置项和性能的相关性模型，并筛选出对性能影响较大(权重)的参数，剔除对性能没有影响的参数；本部分还充分考虑了参数之间的相关性。第 3 部分是在 T_c 过期后，将得到的预期 TP' 最优的配置表部署到 Kafka，并与实际 TP 求误差，得到算法的精确度。算法的整体架构如图 4 所示。

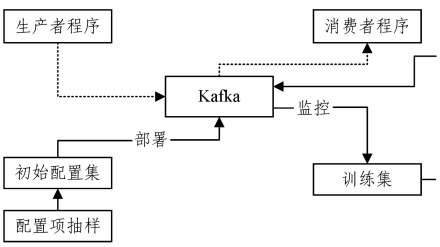

图 4 算法系统设计

Fig. 4 System design

4 自适应性能调优方法的具体实现

自适应性能调优方法使用抽样的思想，对每一个特征在一定范围内进行抽样，并加入线性回归的思想改进 LHS，优先考虑对性能影响(权重)较大的特征，抛弃绝大多数对性能没有影响和影响(权重)较小的特征。自适应性能调优算法的 3 个重点部分是抽样技术、性能预测模型的建立以及调优算法 ENLHS。

4.1 高维数据加权采样技术

在 Kafka 调优中，如果单纯地将 Kafka 的所有配置参数调高，一方面在数据量较小时，会导致系统资源的浪费；另一方面可能导致 Kafka 的消息处理粒度达不到要求，若粒度过大，则 Kafka 自身的数据治理和生产/消费的消息所带来的网络 I/O 和硬件资源也会相应增大，反而可能导致 Kafka 的吞吐性能降低。

Kafka 作为高吞吐、高可用以及高扩展的分布式消息系统，通过载入配置表生成自身的运行环境，因此调整参数表是 Kafka 性能调优的关键。修改配置表的参数必须要考虑到 Kafka 自身的数据环境以及硬件资源；此外，因为配置表中参数的类型复杂，并且多个参数之间会互相影响，所以还需要考虑单条配置参数的修改对整体性能的影响。

第 2 节列出了 3 种已经提出的配置方法，但是它们在高维数据上进行相关计算的代价可能远远高过 Kafka 本身的资源消耗，而合理的抽样方法所需的计算代价非常低，从而可以将大部分计算资源供给 Kafka 自身使用，并减少硬件资源成本，因此本文的参数筛选和计算都基于抽样的方法进行。抽样方法主要分为 4 类：系统抽样、随机抽样、分层抽样和网格抽样。

随机抽样的最大优点是简单易行，但具体到每个配置项，可以是数值型和二值型，因此对于抽样范围不定的情况，随机抽样不适用于在配置表环境下进行抽样。

网格抽样在不同网格之间的差异较大时带来的误差比较大，并且在进行自适应性能优化时所提供的初始训练集并不大，因此该抽样方法不适用。

经过调研和实验发现，分层抽样中的 LHS 对于高维参数的抽样具有非常高的性能，其运行步骤如下。

步骤 1 将每一维分成互不相交的 m 个区间，使得每一区间有相同的概率；

步骤 2 在每一维的每个区间随机抽取一个点；

步骤 3 从每一维随机抽出步骤 2 中选取的点，并组合成向量；

步骤 4 输出向量作为一组数据。

因此，LHS 对连续型数据和离散型数据都有很好的抽样性能。图 5 和图 6 展示了使用 LHS 分别在二维和三维空间中进行抽样的组合，不同形状和颜色的点表现的是不同维度的点取得的数据。图 5 是将每一维分成 5 个区间，图 6 是将每一维分成 100 个区间取得的点，灰色的点表示最终的属性 A,B,C 组合的结果 $(a_{(i)},b_{(i)},c_{(i)})$。因为 LHS 的时间代价和性能开销低，本文的自适应调优算法使用 LHS 作为初始数据集生成的底层方法。

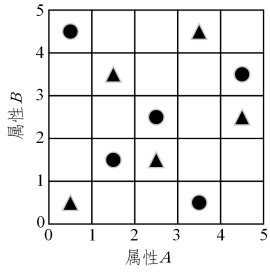

图 5 LHS 在二维平面中进行抽样

Fig. 5 LHS in 2 dimension

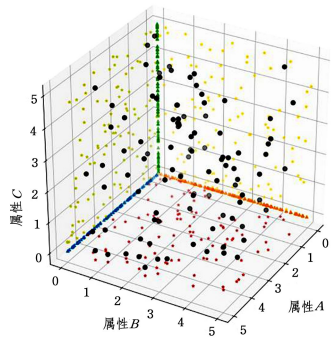

图 6 LHS 在三维空间中的抽样

Fig. 6 LHS in 3 dimension

本文将 LHS 集成到高维数据抽样模块中，作为初始训练集生成和性能模型迭代提供的底层抽样方法。高维数据抽样模块有两个作用：1）由原始的数据集采用经典 LHS 方法进行抽样，得到初始配置表数据，并依次将配置表作为 Kafka 配置文件启动 Kafka 并监测其吞吐性能，与配置表数据进行整合得到初始训练集；2）本文提出的 ENLHS 方法使用 ElasticNet 的权值改进传统 LHS，进一步提高了 LHS 效率，并且将权值代入 LHS 来改进抽样分布。

为了保证使用 ElasticNet 改进的 LHS 抽样方法能够在向量空间内均匀抽样，本文改良了 Xi 提出的密度函数，得到新的概率密度函数，如式(3)所示：

$$f(c,d,x)=d\,\mathrm{e}^{-\omega_i cx} \tag{3}$$

式(3)表示特征 i 在取值范围 $x\in[A,B]$下取值的概率密度，使密度函数维持正态分布的特性。其中，$\omega_i\in\hat{\boldsymbol{\omega}}$，表示特征 i 与性能的权值参数；参数 c 表示抽样密度函数的渐进程度；d 是标准化因子，计算方式如式(4)所示：

$$d=\frac{\omega_i c}{\mathrm{e}^{-\omega_i cA}-\mathrm{e}^{-\omega_i cB}} \tag{4}$$

我们将区间$[A,B]$划分为抽样概率相等的 K 个区间，对每个区间抽样的概率都是 $1/K$。设 z_j 为第 j 个划分点，$j=1,2,\cdots,K$，$z_0=A$，$z_K=B$，则区间$[A,z_j)$的概率为：

$$\frac{j}{K}=\int_{z_0}^{z_j} f(c,d,x)\mathrm{d}x \tag{5}$$

求解得到z_j：

$$z_j=-\frac{\log\left(\mathrm{e}^{-\omega_i cA}-\frac{\omega_i cj}{dK}\right)}{\omega_i c} \tag{6}$$

我们在任意区间取一点$\xi_j\in[z_j,z_{j+1}]$，其满足条件概率 $f(c,d,x)/h$，其中 h 是缩放参数。通过式(7)可以求解 h。

$$1=\int_{z_j}^{z_{j+1}}\frac{d\,\mathrm{e}^{-\omega_i cx}}{h}\mathrm{d}x \tag{7}$$

则 h 的求解公式为：

$$h=\frac{d(\mathrm{e}^{-\omega cz_j}-\mathrm{e}^{-\omega cz_{z+1}})}{ac} \tag{8}$$

我们使用一个随机数 $u\in[0,1]$，即：

$$u=\int_{z_j}^{\xi_j}\frac{f(c,d,x)}{h}\mathrm{d}x \tag{9}$$

将式(8)代入式(9)，求解得到：

$$\xi_j=-\frac{\log(\mathrm{e}^{-\omega_i cz_j}-u(\mathrm{e}^{-\omega_i cz_j}-\mathrm{e}^{-\omega_i cz_{j+1}}))}{ac} \tag{10}$$

至此，可以得到加权 LHS 的抽样方法，其伪代码如算法 1 所示。

算法 1 加权 LHS 抽样方法

输入：向量空间维度 N

输出：样本点

$\hat{\boldsymbol{\omega}}\leftarrow$ ElasticNet 计算权值

res$\leftarrow[\xi_1,\cdots,\xi_n]$

for ω_i in $\hat{\boldsymbol{\omega}}$:

$D_K\leftarrow$将第 i 维划分为 K 个独立不重叠的区间，划分点通过式(6)求得

$\xi_j\leftarrow$生成随机的满足式(10)的一个点

end for

return res

4.2 性能预测模型

由 3.2 节可知，尽管 Kafka 的配置表提供了数百项特征可供调节，但是每个特征的变动对性能的影响(权重)不一样，大多数参数的变动对性能完全没有影响或影响非常小，如 broker.id 用于标识每一个 Kafka Broker 服务器的唯一序号，只需要做到集群内所有序号唯一即可，对服务器的性能没有影响。此外，敏感属性对性能的影响，可以通过控制变量法进行简单的性能比较，得出敏感属性对性能的影响权重。根据这一特性，本文提出 ENLHS 算法，进一步提高参数抽样时算法的性能，该算法的核心在于对特征和性能的关系进行建模，并以此预测结果为依据改进 LHS。

在众多模型中，线性回归模型的实现相对简单并且性能较好，因此本文使用线性回归作为权重的计算模型。Kafka 的“性能与配置表”呈线性关系，其矩阵形式如式(11)所示，其中 $\boldsymbol{C}=(cb_1,cb_2,\cdots,cb_n)^{\mathrm{T}}$，$c\,\boldsymbol{b}_i=(c_1,c_2,\cdots,c_n)$，$\boldsymbol{\omega}=(\omega_1,\omega_2,\cdots,\omega_n)^{\mathrm{T}}$，$n$ 为配置项数量。

$$\boldsymbol{C\omega}=TP(A,E,\boldsymbol{C}) \tag{11}$$

建立线性回归模型后，我们使用 $\hat{\boldsymbol{y}}=(\hat{y}_1,\hat{y}_2,\cdots,\hat{y}_n)^{\mathrm{T}}$ 代表预估的性能，那么可以将式(11)改写为式(12)：

$$\hat{\boldsymbol{y}}=\boldsymbol{C}\hat{\boldsymbol{\omega}} \tag{12}$$

求解线性回归问题的关键是求解系数矩阵 $\hat{\boldsymbol{\omega}}$，使得预测值与实际值的差距最小，这一差值为损失函数 L，可以使用平方误差表示。传统求解线性回归的 OLS 代价函数如式(13)所示。但是传统 OLS 方法求解线性回归时，当特征数量非常多，并且特征之间存在相关性时，会导致方差过大，使得模型不可靠。ElasticNet 解决了这一问题，其结合岭回归和 Lasso 的特点，使用 λ 作为惩罚因子，并且设置 α 作为岭回归和 Lasso 的混合参数。使用 ElasticNet 的目标函数如式(14)所示。

$$\begin{aligned}\min L_{\mathrm{OLS}}&=\min\sum_{i=1}^{m}(y_i-\hat{\boldsymbol{y}}_i)^2\\&=\min\sum_{i=1}^{m}(y_i-c\,\boldsymbol{b}_i^{\mathrm{T}}\hat{\boldsymbol{\omega}})^2\end{aligned} \tag{13}$$

$$\min L_{\mathrm{enet}}=\min\frac{\sum_{i=1}^{m}(y_i-c\,\boldsymbol{b}_i^{\mathrm{T}}\hat{\boldsymbol{\omega}})^2}{2n}+\lambda\left(\frac{1-\sigma}{2}\sum_{j=1}^{n}\hat{\omega}_j^2+\alpha\sum_{j=1}^{n}|\hat{\boldsymbol{\omega}}|\right) \tag{14}$$

至此，Kafka 对性能与配置表的性能预测模型已建立完成，该模型可以通过求解回归系数得解。将该模型嵌入数据迭代模块中，每一次生成的数据将会被重新放回进行模型训练，以提高模型的精确性。

4.3 ENLHS 性能调优算法

基于性能预测模型和抽样方法，可以实现 ENLHS 算法，其伪代码如算法 2 所示。

算法 2 ENLHS 算法

输入：C，T_c

输出：最优特征表 C_B

1. 使用朴素 LHS 生成 h 组不同的初始配置数据
2. 将这 h 组数据部署到 Kafka 中运行，获取吞吐率性能结果，并与对应的配置数据组成初始训练集 T
3. B←T 中性能最好的 b 组数据
4. while T_c 没有超时：
5. 使用 ElasticNet 和 T 计算权值矩阵 $\hat{\boldsymbol{\omega}}$，并生成模型 M；
6. 使用 $\hat{\boldsymbol{\omega}}$ 加权 LHS 抽样 h 组配置数据，并随机抽取 b 组数据；
7. 将 b 组数据部署到 Kafka 中运行，获取吞吐率结果，并与对应配置数据结合为 EP；
8. T←T∪EP
9. 使用 T 训练 M，并修正 $\hat{\boldsymbol{\omega}}$；
10. for each $C_i \in B$：
11. EI←∅
12. C_i^h←对 C_i 中选中权值最高的 h 个特征进行加权 LHS 抽样
13. C_i^*←使用 M 对 C_i^h 进行性能预测，得到预测吞吐性能最好的配置表
14. TP_i^*←将 C_i^* 部署到 Kafka 运行，并收集吞吐性能结果
15. EI←EI∪(C_i^*，TP_i^*)
16. end for
17. B←EI∪B∪EP 中取性能最好的 b 组数据
18. T←T∪EI
19. end while
20. return config of best result in B

ENLHS 算法首先使用朴素 LHS 生成 h 组配置表 T，并取出 b 组性能最好的数据，使用 ElasticNet 和 T 训练模型并计算 $\hat{\boldsymbol{\omega}}$ 向量，然后使用加权 LHS 生成 h 组配置表并抽取 b 组放入 Kafka 中运行，以获取吞吐性能结果组成新的训练集并将其放入 T 中重新训练模型，进行模型迭代和参数修正。

5 实验与结果分析

5.1 实验环境设置

实验使用虚拟机部署 Kafka 集群，集群节点个数为 9，每个节点的硬件和软件环境如表 3 所列，其中 6 个节点用于组成 Kafka 集群，剩余的 3 个节点利用 Flink 集群消费 Kafka 数据。Flink 是纯流式的数据计算框架，在主流的三大计算框架(Apache Spark，Apache Flink，Apache Strom)中，Flink 集成了 Spark 丰富的扩展库、代码逻辑和 Storm 的低时延，引进 Flink 到流式计算中可以获得比 Spark 更好的性能。Flink 在此处的作用是即时分析 Kafka 中的数据，生成时延结果。

表 3 实验环境配置表

Table 3 Experiment environment setting

参数		配置
硬件	CPU	双核 2.4GHz 处理器
	内存	2GB
	硬盘	7200 rpm
	网络	VMWare NATMode
软件	OS	Ubuntu 16.04
	JDK 版本	1.8.151
	Kafka 版本	0.11.0
	Hadoop 版本	2.7-cdh5.7.0
	Flink 版本	1.9.0-cdh5.7.0

为了探究本文提出的自适应性能调优机制对 Kafka 的性能改变程度，本节将其与第 3 节引入的若干自适应调优算法进行性能对比，参与对比的算法有：RandomSearch，BestConfig，RFHOC 和 AutoConfig。在进行实验时，本文设置所有算法的时间限制相等。实验组以及每组的变量如表 4 所列，其中消息确认方式为：在同步模式下，$ACK=0$ 表示生产者只负责发送，不负责分区是否收到消息，这一模式下吞吐率最大，但消息可能会丢失；$ACK=1$ 表示分区 Leader 收到消息即确认；$ACK=$ all 为所有 ISR 收到消息再确认。后两者遵循 AtLeastOnce 语义，消息至少会被发送一次。

表 4 实验组信息

Table 4 Experiment information

实验组	生产者数量	消息大小/kB	确认消息方式(ACK)
1	1	0.1	all
2	1	1.0	all
3	1	0.1	1
4	1	1.0	1
5	1	0.1	0
6	1	1.0	0
7	3	0.1	all
8	3	1.0	all
9	3	0.1	1
10	3	1.0	1
11	3	0.1	0
12	3	1.0	0

5.2 实验结果分析

本节实验所要探究的是自适应性能调优机制对 Kafka 的性能影响情况。对于 DMS 系统来说，性能包括多个方面，但是本文的目的是使 Kafka 在实际生产应用中具有低时延、高吞吐的性能，因此将吞吐率，特别是生产者端的速率作为衡量指标。此外，本节将探究配置项对数据时延的影响，开源 Kafka 自带时延测试脚本，因此相关测试将使用这一脚本进行。

最后，对于本文提出的性能预测模型，同样需要对其准确度进行评判，即预测性能和实际性能的差距。模型的优劣可以使用均方根误差进行评判，其计算公式如式(15)所示。

$$RMSE=\sqrt{\frac{\sum_{i=1}^{n}TP_i^{\text{real}}-TP_i^{\text{pred}})^2)}{n}} \tag{15}$$

首先进行性能实验，实验中除了进行算法的比较之外，还

加入 Kafka 默认配置下的对比，实验结果如图 7 所示。

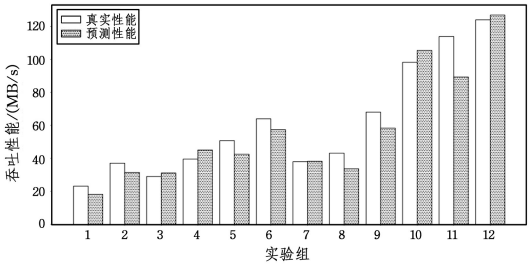

图 7 吞吐率的实验结果

Fig. 7 Experiment result of throughput

通过图 7 可以看出，Kafka 生产者端 ACK 模式在极大程度上影响着 Kafka 的消息接收速率，其原因是 Leader 和 ISR 之间进行数据的同步操作占用了时间。Kafka 自身对于 ISR 同步由于采用了批量拉取的模式，在一定程度上提高了 Kafka 的性能，但是实际情况下，需要根据应用场景在性能和安全性上进行取舍。本文提出的调优机制在进行配置表部署后相对于默认和其他算法都有了较大的性能提升，相对于默认模式性能提升了数倍，相对于其他算法性能提高了 0.3～1 倍。对于时延，本文采用开源版本 Kafka 的原生测试脚本进行测试，测试脚本为 $KAFKA_HOME/bin/目录下的 kafka-producer-perf-test.sh，其需要传入测试参数，如表 5 所列。

表 5 kafka-producer-perf-test.sh 脚本待传入参数

Table 5 Arguments passed to Script kafka-producer-perf-test.sh

参数名	参数意义
topic	主题
num-records	消息总条数
record-size	单条消息的大小
producer-props	生产者属性

本次设置 5 组实验，根据消息大小从 10 kB 到 100 kB 划分为 5 组。消息总条数为 1 073 741 824，producer-props 只传递 bootstrap-server 集群地址，发布订阅的主题的分区数为 10，备份数为 3。根据 kafka-producer-perf-test.sh 的输出结果求平均，得到如图 8 所示的实验结果。

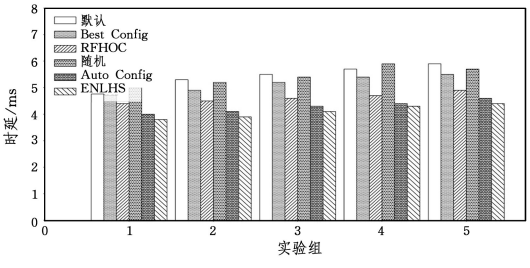

图 8 时延情况的对比

Fig. 8 Latency comparison

结合图 7 和图 8 可知，本文提出的自适应调优方法相对于几种对比方法，在相同的时间 T_c 内，可以得到更好的吞吐性能以及更低的时延。

对于模型的拟合程度实验，本文将吞吐率实验中的实验组配置项导出，并与实际吞吐率进行比较，得到如图 9 所示的结果。

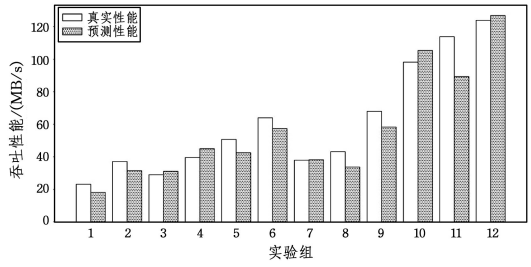

图 9 模型误差的测试结果

Fig. 9 Model error results

结合图 9 和式(14)，可以计算出本节实验统计出的模型的 $RMSE \approx 9.34$，与图 8 中的 Random Search，Best Config，RFHOC 和 AutoConfig 等算法相比，这一数值更优，因此本文所提性能预测模型所预测的结果在可接受的误差范围内。

结束语 本文针对由于 CDMS 问题的特征数量多、种类复杂、性能反馈慢所带来的 Kafka 性能瓶颈和人工调优困难的现象，经过调研分析众多自适应调优方案(如 RFHOC 及基于抽样的 AutoConfig)的不足后，提出了基于 Kafka 的自适应性能调优机制。该机制充分考虑了抽样机制的优势，引入 ElasticNet 计算特征与性能的影响权值并得到预测模型，该模型与 LHS 结合后进一步提高了抽样的效率。实验结果表明，与其他主流自适应调优方案相比，基于 LHS 算法的机制在配置项众多、种类复杂的环境下的 CDMS 问题中能够获得更好的调优效果，Kafka 能获得更好的性能表现。

参 考 文 献

[1] DEAN J，GHEMAWAT S. MapReduce：Simplified Data Processing on Large Clusters[C]// Sixth Symposium on Operating System Design & Implementation. USENIX Association，2004：107-117.

[2] HIRAMAN B R，CHAPTE V M，ABHIJEET C K. A Study of Apache Kafka in Big Data Stream Processing[C]// 2018 International Conference on Information，Communication，Engineering and Technology(ICICET). 2018：1-20.

[3] DOBBELAERE P，ESMAILI K S. Kafka versus RabbitMQ：A comparative study of two industry reference publish/subscribe implementations：Industry Paper[C]// Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems(DEBS'17). Barcelona，Spain：ACM Press，2017：227-238.

[4] DELAMER I M，MARTINEZ LASTRA J L，PEREZ O. An evolutionary algorithm for optimization of XML publish/subscribe middleware in electronics production[C]// Proceedings 2006 IEEE International Conference on Robotics and Automation，2006(ICRA 2006). 2006：681-688.

[5] BANG J，SON S，KIM H，et al. Design and implementation of a load shedding engine for solving starvation problems in Apache Kafka[C]// NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. Taipei：IEEE，2018：1-4.

[6] JAVED M H，LU X，PANDA D K. Characterization of Big Data Stream Processing Pipeline：A Case Study using Flink and Kafka

[C]// Proceedings of the Fourth IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT'17). Austin, Texas, USA: ACM Press, 2017: 1-10.

[7] D'SILVA G M, KHAN A, GAURAV, et al. Real-time processing of IoT events with historic data using Apache Kafka and Apache Spark with dashing framework[C]// 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). Bangalore: IEEE, 2017: 1804-1809.

[8] HECHT R, JABLONSKI S. NoSQL evaluation: A use case oriented survey[C]// 2011 International Conference on Cloud and Service Computing. 2011: 336-341.

[9] KAUR J. In-Memory Data processing using Redis Database[J]. International Journal of Computer Applications, 2018, 180(25): 26-31.

[10] HAO X, JIN P, YUE L. Efficient Storage of Multi-Sensor Object-Tracking Data[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(10): 2881-2894.

[11] WANG Y, WANG C. A Design of Reliable Consumer Based on Kafka[J]. Software, 2016, 37(1): 61-66.

[12] ZOU H, HASTIE T. Regularization and variable selection via the elastic net[J]. Journal of the Royal Statistical Society: Series B(Statistical Methodology), 2005, 67(2): 301-320.

[13] BAO L, LIU X, XU Z, et al. AutoConfig: automatic configuration tuning for distributed message systems[C]// Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018). Montpellier, France: ACM Press, 2018: 29-40.

[14] ALEJAN DRO A P. Workload Characterization of the SPECjms-2007 Benchmark[C]// Formal Methods & Stochastic Models for Performance Evaluation, Fourth European Performance Engineering Workshop. Epew, Berlin, Germany, 2007: 228-244.

[15] BUCHMANN A. Benchmarking of message-oriented middleware[C]// Proc of the Debs. 2009: 1-2.

[16] ESPOSITO C, RUSSO S, CRESCENZO D D. Performance assessment of OMG compliant data distribution middleware [C]// IEEE International Parallel & Distributed Processing Symposium. 2008: 1-8.

[17] HENARD C, PAPADAKIS M, HARMAN M, et al. Combining Multi-Objective Search and Constraint Solvin[C]// 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. 2015: 517-528.

[18] SHANGGUAN B, YUE P, WU Z. A stream computing based apporach for updating waterlogging infomation on remote sensing images[C]// 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). Fort Worth, TX: IEEE, 2017: 373-375.

[19] HAN J, MICHELINE K. Data mining: concepts and techniques [J]. Data Mining Concepts Models Methods & Algorithms Second Edition, 2006, 5(4): 1-18.

[20] MAGERMAN D M. Statistical decision-tree models for parsing [C]// Proceedings of the 33rd annual meeting on Association for Computational Linguistics. Cambridge, Massachusetts: Association for Computational Linguistics, 1995: 276-283.

[21] BEI Z, YU Z, ZHANG H, et al. RFHOC: A Random-Forest Approach to Auto-Tuning Hadoop's Configuration [J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(5): 1470-1483.

[22] TIBSHIRANI R. Regression Shrinkage and Selection Via the Lasso[J]. Journal of the Royal Statistical Society, 1996, 58(1): 267-288.

[23] HELTON J C, DAVIS F J. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems[J]. Reliability Engineering & System Safety, 2003, 81(1): 23-69.

XIE Wen-kang, born in 1995, postgraduate. His main research interests include cloud computing, big data frameworks and data processing.

LI Peng, born in 1979, Ph.D, professor, master supervisor, is a member of China Computer Federation. His main research interests include computer communication networks, cloud computing, and information security.