软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.007280]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于组合负载预测模型的多租户数据库弹性伸缩方法 *

徐海洋 [1,2,3], 刘海龙 [1,2], 陈先 [1,2], 王磊 [1,2], 金轲 [1,2], 侯舒峰 [1,2], 李战怀 [1,2]

[1](西北工业大学 计算机学院，陕西 西安 710100)

[2](西北工业大学 大数据存储与管理工业和信息化部重点实验室，陕西 西安 710100)

[3](西安中兴新软件有限责任公司, 陕西 西安 710100)

通讯作者: 刘海龙, E-mail: liuhailong@nwpu.edu.cn

摘 要: 云环境下的多租户数据库重要特性之一是可伸缩性，然而大部分的弹性伸缩技术难以针对复杂变化的负载进行有效的伸缩决策.若能提前预测负载变化，则能够准确调整资源供给.鉴于此，本文提出了基于内存负载预测的多租户数据库弹性伸缩方法,包括一种组合负载预测模型和一种弹性伸缩策略.组合负载预测模型融合了卷积神经网络、长短期记忆网络和门控循环单元的优势，可以比较精确地预测数据库集群内存负载需求；弹性伸缩策略基于需求预测结果，调整虚拟机数目，保证资源供应处于合理范围.与现有方法对比，本文模型预测误差降低 8.7%-21.8%，预测拟合度提高 4.6%；在此基础上，本文改进了贝叶斯优化算法用于本模型超参数调优，解决了贝叶斯优化在离散解、连续解的组合域中效果较差的问题，误差进一步降低 7.6%，拟合度进一步提高 1.04%.实验结果表明，与 kubernetes 中应用最广泛的伸缩策略相比，本文弹性伸缩方法避免了弹性伸缩的滞后性与资源浪费，响应时间降低 8.12%，延迟降低 9.56%.

关键词: 多租户数据库;资源管理;资源预测;弹性伸缩

中图法分类号: TP311.13

中文引用格式: 徐海洋,刘海龙,陈先,王磊,金轲,侯舒峰,李战怀. 基于组合负载预测模型的多租户数据库弹性伸缩方法. 软件学报. http://www. jos.org.cn/1000-9825/7280.htm

英文引用格式: Xu HY, Liu HL, Chen X, Wang L, Jin K, Hou SF, Li ZH. Elastic Scaling Method for Multi-tenant Databases Based on the Hybrid Workload Prediction Model. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7280.htm

Elastic Scaling Method for Multi-tenant Databases Based on the Hybrid Workload Prediction Model

XU Hai-Yang [1,2], LIU Hai-Long [1,2], CHEN Xian [1,2], WANG Lei[1,2], JIN Ke[1,2], HOU Shu-Feng[1,2], LI Zhan-Huai[1,2]

[1](School of Computer science, Northwestern Polytechnical University, Xi'an 710100, China)

[2](Key Laboratory of Big Data Storage and Management, Northwestern Polytechnical University, Ministry of Industry and Information Technology, Xi'an 710100, China)

Abstract: One of the important features of multi-tenant databases in cloud environments is scalability. However, most elastic scaling techniques struggle to make effective scaling decisions for complex workload variations. If workload changes can be predicted in advance, resources can be accurately adjusted. In this paper, we propose a memory load-based elastic scaling method for multi-tenant databases, including a cluster-level load prediction model and an elastic scaling strategy. The load prediction model integrates the advantages of convolutional neural networks, long short-term memory networks, and gated recurrent units to accurately forecast the memory load requirements of the database cluster. The elastic scaling strategy, based on the demand prediction results, precisely adjusts the number of virtual machines to ensure that resource provisioning remains within a reasonable range. Compared to existing methods, this model

基金项目: 基金项目:国家重点研发计划(2023YFB4503600)、国家自然科学基金（62172335）、CCF-华为胡杨林基金(CCF-HuaweiDBIR0004B).

收稿时间: 2024-05-27; 修改时间: 2024-07-16, 2024-08-19; 采用时间: 2024-08-29; jos 在线出版时间: 2024-09-13

reduces prediction errors by 8.7% to 21.8% and improves prediction fitting by 4.6%. Additionally, this paper improves the Bayesian optimization algorithm for hyperparameter tuning of this model. It addresses the issue of poor performance of Bayesian optimization in combined domains of discrete and continuous solutions, further reducing errors by 7.6% and improving fitting by 1.04%. Experimental results indicate that compared to the most widely used scaling strategy in Kubernetes, the elastic scaling method proposed in this paper avoids the latency and resource waste associated with elastic scaling. Response time is reduced by 8.12%, latency by 9.56%.
Key words: Multi-tenant databases; Resource management; Resource prediction; Elastic scaling

前言

为了提高利润和降低成本，越来越多的组织或个人选择将业务置于云环境下的多租户数据库.云环境提供的重要特性之一是可伸缩性，即按需扩展或缩减已分配的计算资源的能力.可伸缩性特性允许用户以一种弹性的方式运行他们的业务应用，只使用他们需要的计算资源，并且只为他们使用的计算资源付费.资源的弹性伸缩能力对于优化云数据库的运营成本和服务质量至关重要[1].

弹性伸缩技术主要包括基于响应阈值的被动式方法与基于预测的主动式方法两大类型. 在基于响应阈值的弹性伸缩方案中，租户必须为不同的应用指定资源伸缩的阈值，只有在应用使用的资源超过阈值后才会触发资源的伸缩. 从使用者的角度出发，这种模型的可定制性最强，但是它存在如下不足：

（1）决策延迟. 基于响应阈值的被动式弹性伸缩参照当前系统性能和负载进行决策，存在一定调整延迟，可能导致系统无法及时满足用户负载的资源需求，导致服务阻塞甚至服务中断.

（2）过度配置风险.负载突然下降时，可能导致过度配置的资源，引发资源浪费和增加成本.

（3）最佳阈值确定困难.资源的伸缩决策依赖最佳阈值的确定. 而确定最佳阈值本身也并非是一件简单的任务，需要深入了解应用的特性和需求.

（4）策略制定复杂.要准确地确定资源弹性伸缩的时机及数量难度大.不合理的策略会导致“资源震荡性调整”.

针对上述不足，基于预测的主动式弹性伸缩方法基于租户业务历史负载来预测未来的资源需求、及时改变资源的规模，以便在负载或资源利用率达到一定水平时已经可以获得足够的资源.然而，为了实现精准高效的主动式弹性伸缩，基于预测的主动式弹性伸缩方法存在以下三方面的挑战：

（1）预测模型的准确性直接影响着弹性伸缩的效果，需要构建和训练具有良好泛化能力的预测模型.

（2）不同租户业务负载的动态变化可能导致预测失准，预测模型需要适应不断变化的工作负载.

（3 主动式弹性伸缩需要在预测结果的基础上制定合理的资源伸缩策略，避免过度或不足的资源分配.

LSTM、GRU、CNN 等模型在特征提取、模型的泛化能力等方面都或多或少存在缺陷，很难独立应对上述挑战. 结合多个模型的组合模型可以充分发挥每个模型的优点、克服其缺点，并实现更准确、更可靠的时序预测.所以本文设计了一种组合预测模型，使用 CNN 来提取局部特征并实现快速计算，在预处理后的数据上使用 LSTM 来学习和处理长序列的时间序列数据，并且能够捕捉到序列中的时间依赖性，最后在 LSTM 的输出上使用 GRU 来进一步加速模型训练，并实现更准确的预测.另外，因为资源之间的关联性，多维资源的预测和调整本身是一件复杂的任务. 为了简化问题，本文主要研究多租户数据库内存资源的弹性伸缩方法. 本文基于一种组合预测模型，根据集群级内存负载的资源需求预测结果精确调整集群中虚拟机的数目，保证资源的供应处于合理的范围内.

论文的主要贡献如下：

（1）为了解决单模型预测的不足，本文设计了一种 CNN-LSTM-GRU 组合模型，融合了卷积神经网络善于提取时序数据深度特征、长短期记忆网络预测精度高和门控循环单元预测时间短的优势，旨在提高预测的准确性和健壮性.该组合模型用于多租户数据库集群级别的负载需求预测.实验标明：本文模型预测误差降低 8.7%-21.8%，预测拟合度提高 4.6%

（2）在组合预测模型中包括的大量离散和连续的超参数.本文改进了贝叶斯优化算法，在同时包括离散、连续参数的组合域中寻找最优参数解，可以更高效、准确地找到最优的组合预测模型的超参数配置. 实验表

明，使用本文的超参数求解方法后，预测误差降低了 7.6%，拟合度提高了 1.04%.

（3）基于组合预测模型，实现了一种基于预测的主动式内存资源弹性伸缩策略，能够比较精确地根据集群级内存负载的资源需求调整集群中虚拟机的数目，保证资源的供应处于合理的范围内，响应时间降低 8.12%，延迟降低 9.56%.

本文第 1 节介绍弹性伸缩的相关方法和研究现状.第 2 节介绍本文算法框架、负载预测模型和弹性伸缩策略.第 3 节通过对比实验验证了所提方法的有效性.最后总结全文.

1 相关工作

基于阈值规则的方案中被占用的资源到达事先设定的阈值时就进行资源的自动调整.例如，Kubernetes[2]采用基于阈值规则的方法进行资源弹性伸缩.它提供了一个控制平台来周期性地收集、计算和提取各种类型的指标，包括 CPU 或者内存的使用率，当超出某一个阈值时或某一指标过低时，便对 Pod 的数量进行增加或减少.单独以资源阈值规则作为缺点明显：调整策略过于被动，无法针对多租户复杂的情况进行灵活更新，在资源震荡状态下反复伸缩，造成很多不必要的操作，增加额外开销.文献[3]基于控制论，开发了自适应水平弹性控制器，用于控制缩放决策和防止资源振荡.弹性控制器的输出是添加或删除虚拟机的数量，用以确保服务请求的数量等于资源所需的数量且符合服务水平协议，同时保持虚拟机的数量最低.基于控制论的方法试图建立一个模型、通过定义一个控制器来为动态地调整资源，此类方法的实用性主要依赖于选择的控制模型类型以及系统的动态性，另外，建立一个与输入输出参数相匹配的模型较为困难.

与基于规则的方法不同，一些文献尝试了利用强化学习模型来指导多租户系统的弹性伸缩.文献[4]为 Kubernetes 研制了一款基于强化学习模型的开源弹性伸缩工具，支持 Kubernetes 集群横向弹性伸缩，但在实际的应用场景中，集群横向的资源扩展粒度还是过大，无法很好地满足现在场景细粒度的需求.文献[5]提出了一种基于强化学习的非集中式自动缩放方法，用于调整 Web 应用作业的资源规模.通过周期性收集服务器的相关指标，该强化学习模型在无监督下持续学习服务器状态，从而实现对服务器状态的预测.这使得可以根据需要增加或减少服务器数量进行扩缩容，但这种方法不能在单个服务器上进行伸缩操作.相似地，文献[6-10]均使用 Q-学习算法指导弹性伸缩，Q-学习算法用于代理学习并决策何时添加或删除虚拟机缩放资源，它平衡了响应时间和平均资源利用率，即在尽量满足 SLA 的同时，使资源利用率尽可能的高，通过学习最佳资源利用率阈值，在遵守 SLA 的同时给出最优策略，实现资源利用率最大化.文献[11]提出了一个基于异常检测的深度强化学习资源伸缩框架 ADRL，ADRL 定时收集每台虚拟机的资源使用数据，基于此数据根据前馈神经网络进行预测，与传统的直接根据预测结果进行资源伸缩不同，ADRL 通过检测异常的预测值来触发强化学习模型做出伸缩决策.为了减少资源震荡，ADRL 只有当接收到至少连续 N 个异常时才会触发伸缩决策.文献[12]认为强化学习的方法相对于基于阈值的方法以及时间序列的方法能够更有效地应对系统的动态性.但是，强化学习的扩展性较差，其搜索空间随状态变量呈指数级增长，由于缺乏相关的领域知识，强化学习在初始的性能可能非常差，它需要相当多的初期探索行动.

2 基于组合负载预测模型的多租户数据库弹性伸缩方法

2.1 基于预测模型的多租户弹性伸缩方法框架

本文期望通过准确预测动态负载需求，精确调整集群中虚拟机的数目来解决实现精准高效的主动式弹性伸缩的三个方面的挑战.其整体框架如图 1 所示.其中组合负载预测模型负责根据从多租户数据库集群获取的历史负载的内存使用状态的时序数据预测负载未来的内存资源需求.弹性伸缩模块策略模块根据组合负载预测模型预测的资源需求选择合适的弹性伸缩策略、形成最终的弹性伸缩决策，进行资源伸缩.

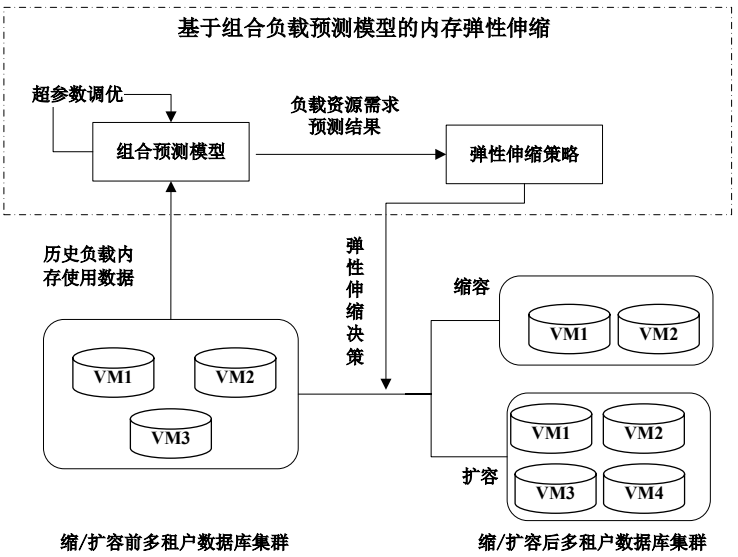

图 1 弹性伸缩方法整体框架

2.2 CNN-LSTM-GRU组合预测模型

单一预测模型有其局限性，例如：卷积神经网络（CNN）对于处理长序列的时间序列数据可能不太适用；长短期记忆网络（LSTM）尽管可以很好地学习和处理长序列的时间序列数据并有效解决 RNN 的梯度爆炸或者消失问题[13]，但需要更多的参数和计算资源，训练时间较长；门控循环单元（GRU）则比 LSTM 更加简洁，同时能够捕获到时间序列中的依赖性，缺点是 GRU 可能会丢失信息导致预测精度下降.为了弥补了单一预测模型的局限性，本文设计并实现了一种基于 CNN-LSTM-GRU 的组合预测模型对时序负载进行预测，模型的整体结构如图 2 所示. CNN 模型在特征提取与特征选择领域的能力已经在现有研究中得到证明[14-15].由于 LSTM 与 GRU 模型在训练中难以捕获时间序列中极值点的特征，因此本文利用 CNN 将特征从低维空间扩展到高维空间，以获得更丰富的特征语义，由此使得组合模型的预测能力大大提高.组合模型的第一阶段是由 CNN 对于输入的时序负载数据进行深度特征提取，通过神经网络的特征工程方法赋予时间序列高维特征，CNN 的卷积层和池化层用于平衡特征维数和提高泛化能力.第二阶段，通过 LSTM-GRU 模型对 CNN 处理的特征进行训练和预测，该组合模型同时结合了 LSTM 预测精度高和 GRU 训练速度快的优势.

图 2 中的 LSTM-GRU 部分一共有三层网络结构，具体如图 3 所示.前两层网络设置为 LSTM 模型，LSTM 模型与其他单一预测模型相比预测效果更好，并且与仅一层 LSTM 相比，两层 LSTM 预测准确率更高.但 LSTM 的三门机制使得结构变得较为复杂，从而增加了参数数量.而 GRU，作为 LSTM 的变体，整合了遗忘门与输入门为单一的更新门，这使参数更少，从而降低了网络的复杂性，并减小了矩阵运算量，提高了收敛速度.为了提高训练效率，本文决定在第三层采用 GRU 设计.每一层记忆单元之间的横向箭头代表历史数据的传输，层与层之间的纵向箭头代表对应时刻每个记忆单元的输入.在上述三层网络训练过程中，一个关键问题是防止过拟合.本文使用 dropout 方法优化三层网络用以防止过拟合.在训练过程中，会从网络中随机丢掉一些记忆单元，这些记忆单元不会参与到后续的训练过程，这个过程在每一个训练批次都是不同的，因此每个批次的网络结构都不相同，从而可以提高模型的泛化能力.此方法在图 3 中体现为层与层之间的记忆单元的“随机”数据传输，用虚线箭头表示.

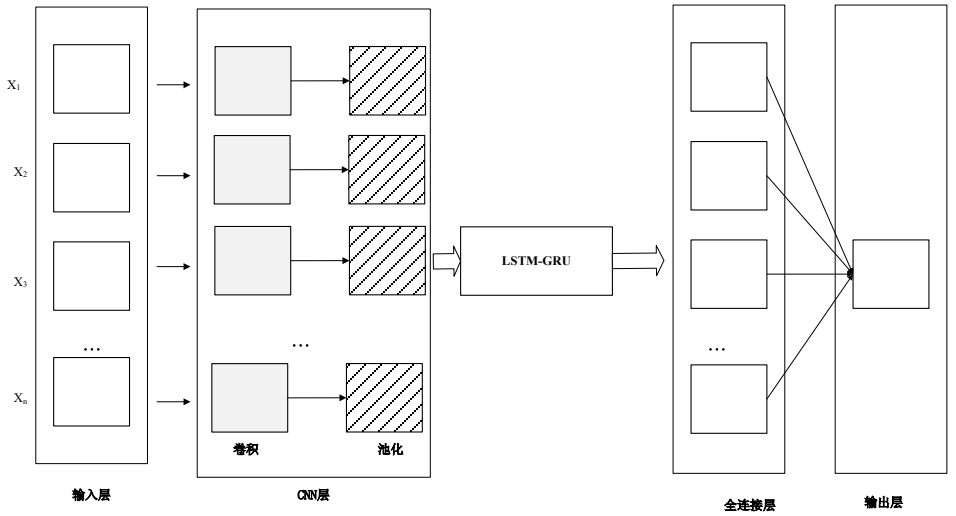

图 2 CNN-LSTM-GRU 组合模型

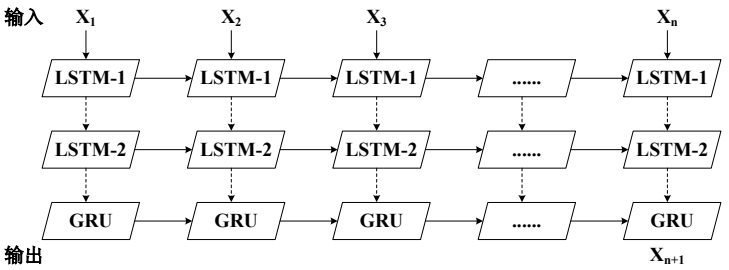

图 3 LSTM-GRU 三层网络模型

最后，利用全连接层将 LSTM-GRU 输出的隐藏状态映射为最终的预测结果.

CNN-LSTM-GRU 模型的形式化描述如下：

设输入的时序负载数据$X = (x_1, x_2, \ldots, x_t, \ldots, x_n)$，其中$x_n$表示时间步$n$的输入数据，模型使用时间步长为$t$的数据预测$t+1$时刻的负载数据，即输出$Y_{t+1} = f([x_1, x_2, \ldots, x_t])$.

对于 CNN 部分，考虑到时序负载为单变量数据，本文使用滑动窗口技术创建多维数据，为序列中的每个时间点创建一个窗口，窗口中包含当前时间点之前的t个连续时间点的值，即上文所提到的$[x_1, x_2, \ldots, x_t]$，由此将特征从低维空间扩展到高维空间，将其看作是输入到 CNN 模型的多维数据.定义卷积操作为F_{conv}，激活函数为f_{ReLU}，设定模型中的参数集合为θ，包含卷积核大小、输出通道数等参数，对前t时刻的时序负载数据进行卷积操作，用H_{CNN}表示输出的深度特征，则 CNN 特征提取阶段可以具体表达为：

$$H_{CNN} = f_{ReLU}(F_{conv}([x_1, x_2, \ldots, x_t], \theta)) \qquad (1)$$

对于 LSTM-GRU 部分，定义 LSTM 操作为F_{lstm}，定义 GRU 操作为F_{gru}，这两个操作分别对应于 LSTM 和 GRU 模型的前向传播过程.设定 LSTM 和 GRU 的隐藏单元数分别为h_{lstm}和h_{gru}，该部分采用 dropout 方法，

其中$p_{dropout}$表示 dropout 的概率.用$H_{LSTM-GRU}$表示该过程输出的隐藏状态，则对前t时刻的深度特征进行 LSTM-GRU 模型训练和预测过程具体表达为：

$$H_{LSTM-GRU} = F_{gru}(F_{dropout}(F_{lstm}(F_{dropout}(F_{lstm}(H_{CNN}))), p_{dropout}), h_{lstm}, h_{gru}) \quad (2)$$

在最后的输出层，定义全连接操作的权重矩阵W_{fc}，偏置为b_{fc}，利用 LSTM-GRU 模型输出的隐藏状态进行全连接层映射，得到$t+1$时刻的预测结果：

$$Y_{t+1} = W_{fc} \cdot H_{LSTM-GRU} + b_{fc} \quad (3)$$

本文 CNN-LSTM-GRU 组合模型的预测流程如图 4 所示. 图 4 中的实线数据流展示了模型的训练过程.首先，训练数据被输入到 CNN 模型中进行深度特征提取.该模型的输出分为两部分：深度训练数据（形如$[x_1, x_2, \ldots, x_t]$）和深度训练标签（形如x_{t+1}），它们作为 LSTM-GRU 部分的训练输入，t表示时间步长，使用前tt时刻的负载预测$t+1$时刻的负载数据.训练完成后的模型被用于对测试数据进行测试. 图 4 中的虚线数据流表示了模型的测试过程.标准化后的测试数据通过已训练的 CNN-LSTM-GRU 模型生成预测结果.随后，对预测结果进行反标准化，得到具体的预测数值.将预测结果与真实结果进行对比，并计算具体的评价指标，这些指标用于评估模型的性能优劣.

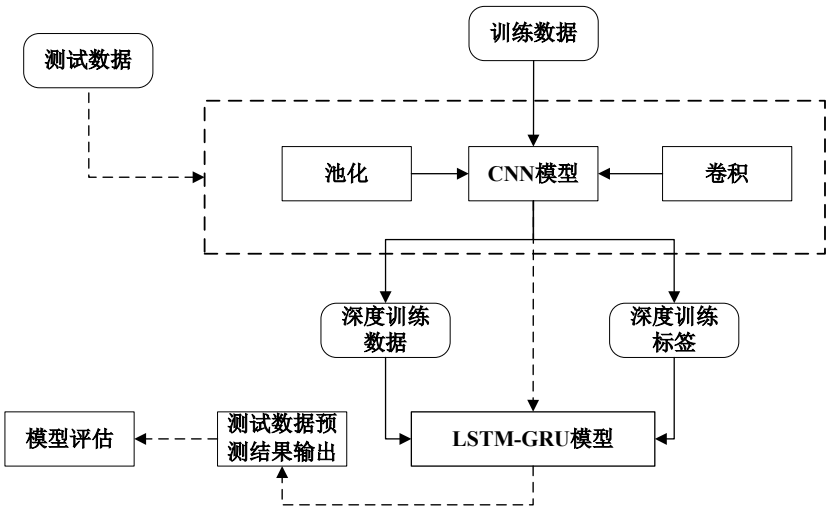

图 4 组合模型预测流程

2.3 组合预测模型超参数调优

本文提出的组合预测模型依赖于超参数的最优取值以提升其预测精准性.然而，在实际应用的多租户数据库中，由于内存负载数据的高度随机性和波动，手动设置的模型超参数往往难以达到最佳状态，影响着预测模型的性能.考虑到需要基于负载数据特性和数据量确定最佳的网络结构超参数，本文采用了参数调优算法来为该组合预测模型寻找最佳的参数组合.

虽然贝叶斯优化已经在实践中成为一种快速给出高效配置的优化算法，然而，大多数基于高斯过程的贝叶斯优化算法明确地假设解空间为一个连续空间，这就导致了其在离散解、连续解的组合域中扩展性较差.而本文需要基于贝叶斯优化算法对组合模型的超参数集合进行优化，其中，训练次数（epochs）、dropout 层系数等参数为连续数据，而隐藏层单元数（unit）、批处理大小、学习率等参数为离散数据.贝叶斯优化不适用于组合域的主要挑战在于难以在组合域上定义统一的协方差函数和距离度量以解释变量之间复杂的相互作

用.鉴于此，本文提出的解决方法是分别对于离散解空间和连续解空间定义协方差函数，并基于一定的规则进行组合，本文将改进后的贝叶斯优化算法命名为 HBO 算法（Hybrid Bayesian Optimization, HBO）.

首先定义组合域中的贝叶斯优化：本文将学习率、dropout 层系数等n个连续参数定义为$x = [x_1, x_2, \ldots, x_n]$，将隐藏层单元数、批处理大小、学习率等$m$个离散参数定义为$h = [h_1, h_2, \ldots, h_m]$，则参数解空间可以表示为$z = [h, x]$，则组合域中的贝叶斯优化可以表示为形如公式 4.

$$z* = [h*, x*] = arg\max_{z} f(z) \quad (4)$$

对于离散参数h，本文将其对应的协方差函数定义如公式 5.

$$k_h(h, h') = \frac{\sum_{i=1}^{n} II(h_i - h_i')}{n} \quad (5)$$

其中，$h_i = h_i$'时$II(h_i - h_i')$取值为 1，反之为 0.

对于连续参数x，其协方差函数仍选择平方指数协方差函数.

本文同时使用协方差和与协方差积来组合上述两个协方差函数，组合来自两种类别的数据的信息，提高了其建模性能，如公式 6 所示.

$$\begin{aligned} k_z(z, z') = (1-\lambda)(k_h(h, h') + k_x(x, x')) \\ +\lambda k_x(x, x') k_h(h, h') \end{aligned} \quad (6)$$

最后，使用最大化对数边缘似然法确定k_x中参数l取值为 1，k_z中参数λ取值为 0.5.优化后的 HBO 算法中，采集函数选择为期望改进函数（EI）.

采用 HBO 进行超参数调优的过程如下：

步骤 1：目标函数定义.本文选择预测模型中的时间窗口步长（Length，L）、学习率（Learing Rate，LR）、dropout 比例（D）、隐藏单元数（Hidden size，H）、批处理大小（Batchsize，B）、核大小（Kernel size，K）等六个超参数作为调优对象.六个参数中，时间窗口步长、学习率、dropout 比例为连续参数；隐藏单元数、批处理大小、核大小为离散参数.定义目标函数$f(z)$，其中，$z = [h, x]$是组合模型的超参数集，包括连续参数$x = (L, LR, D)$和离散参数$h = (H, B, K)$.目标函数是通过计算均方误差（MSE）来评估组合模型在给定超参数下的性能.

步骤 2：先验信念建模.引入先验分布$P(f)$来表示对目标函数的先验信念，先验分布由替代模型高斯过程给出.高斯过程中的协方差函数，选择公式(6)中设计的混合协方差函数$k_z(z, z')$.

步骤 3：选择采样点：在超参数空间中选择一个采样点z_i.对于初始采样点，则设置为$z_0 = [h_0, x_0]$，$x_0 = (L_0, LR_0, D_0)$，$h_0 = (H_0, B_0, K_0)$.

步骤 4：目标函数评估.计算目标函数在选择的超参数点z_i处的观测值y_i，计算方法为在训练集上计算模型的 MSE 值.

步骤 4：后验更新.基于新的观测值更新后验分布$P(f|z_i, y_i)$，其中z_i和y_i分别表示先前的超参数采样点和

观测值.

步骤 5：寻找最优点.在后验分布中找到最大化期望改善（EI 采集函数）的超参数点$z*$.

步骤 6：更新模型.将新的观测值y_i和对应的超参数z_i点加入训练集，重新训练超参数模型，得到更新后的先验分布.

步骤 7：迭代.重复步骤 3 到步骤 6，直到达到预定的迭代次数，并输出最优超参数解$z*$.

最终将 CNN-LSTM-GRU 组合模型的超参数调整为上述步骤得到的最优超参数解，训练数据流与测试数据流保持不变.

2.4 基于组合预测模型的弹性伸缩策略

本文所提出的基于内存负载预测的弹性伸缩方法中的弹性伸缩策略如算法 1 所示.

本文利用上文的基于 HBO 优化的 CNN-LSTM-GRU 预测模型$f()$，根据前t时刻的负载数据$L(1),L(2),\dots,L(t)$来预测$t+1$时刻的负载需求$P_n(t+1)$，如公式 7 所示.

$$P_n(t+1)=f(L(1),L(2),\dots,L(t)) \quad (7)$$

多租户数据库服务提供商将总的资源分布在虚拟机中供租户使用，因此本文的弹性伸缩策略以虚拟机为基本单位.本文以"当前分配资源的百分比阈值"指代当前所有虚拟机所能提供的总资源的一定百分比.例如，如果资源池有 10 个活跃的虚拟机，每个虚拟机提供 10GB 的内存，那么总共有 100GB 的内存资源.如果设定的伸缩策略上限阈值是 70%，当预测的需求超过这个阈值时，即 70GB，策略认为需要增加虚拟机.

具体的弹性伸缩策略如下：

不采取操作：当预测的负载$P_n(t+1)$处于$[\beta,\alpha]$区间中，说明资源利用较为平稳，不需要进行伸缩操作.

扩容：当预测的负载$P_n(t+1)$超过了当前分配资源的α阈值，需要增加虚拟机来满足未来需求.增加的虚拟机数量使用公式 8 计算.

$$\Delta R_{increase}(t)=ceil(\frac{P_n(t+1)}{\alpha*P_u})-R(t) \quad (8)$$

缩容：当预测的负载$P_n(t+1)$低于了当前分配资源的β阈值，需要减少虚拟机来节约资源.减少的虚拟机数量使用公式 9 计算.

$$\Delta R_{decrease}(t)=R(t)-ceil(\frac{P_n(t+1)}{\beta*P_u}) \quad (9)$$

公式 8 与 9 中，$\Delta R_{increase}(t)$与$\Delta R_{decrease}(t)$分别代表当前时刻应该增加或减少的虚拟机数目，$R(t)$表示在时间t的活跃虚拟机数量，P_u表示每个虚拟机所能提供的固定内存资源量，$ceil()$表示向上取整函数.α、β分别表示阈值上下限，通过上述两个公式可以保证整个多租户数据库资源池内存利用率维持在$[\beta,\alpha]$这个合理区间中.

算法 1：弹性伸缩策略

输入：

历史负载数据$L(1), L(2), \ldots, L(t)$

单个虚拟机提供的内存资源P_u

调整阈值上限α、调整阈值下限β

当前时刻虚拟机数目$R(t)$

组合预测模型$f()$

输出：

$t+1$时刻的虚拟机数目$R(t+1)$

$P_n(t+1) = f(L(1), L(2), \ldots, L(t))$

if $P_n(t+1) > \alpha * R(t) * P_u$ then

    $\Delta R_{increase} = ceil(P_n(t+1)/（\alpha * P_u)) - R(t)$

    $R(t+1) = R(t) + \Delta R_{increase}$

else if $P_n(t+1) < \beta * R(t) * P_u$ then

    $\Delta R_{decrease} = R(t) - ceil(P_n(t+1)/(\beta * P_u))$

    $R(t+1) = R(t) - \Delta R_{decrease}$

else

    $R(t+1) = R(t)$

end if

3 实验分析

3.1 实验平台

本文基于 CloudSim 开发多租户数据库集群环境，评估本文弹性伸缩方法在面对租户业务负载时的性能表现.尽管 CloudSim 提供了丰富的计算功能，但在处理动态负载、自动资源调整等弹性伸缩方面的支持相对有限.当前版本的 CloudSim 在其基础框架中并未内置专门用于弹性伸缩的方法，只有简单的手动增加或减少虚拟机的接口和机制，为了在多租户集群环境中更全面地实现和研究弹性伸缩策略，本文对 CloudSim 开发了额外的方法和接口来支持多租户数据库集群的弹性伸缩机制.

本文弹性伸缩策略的实现分为监控、分析、计划、执行四个模块.监控模块被设计用于收集多租户数据库集群中的多种资源指标，监控虚拟机使用情况并收集历史信息，保存相关数据至特定数据文件中，用于后续分析模块基于该数据进行预测.分析模块的主要作用之一是使用本文 2.2 和 2.3 节所设计的基于 HBO 优化的组合预测模型预测得到后续时间序列点的内存需求结果，之后将需求数据存储在预测结果文件中；另一方面的作用是根据监控模块对于虚拟机集群的性能指标的监控数据计算整合出在整个弹性伸缩过程中，虚拟机集群的性能变化情况，例如响应时间与延迟时间的变化，用于评估弹性伸缩策略的优劣.计划模块的主要作用是进行决策，计划模块将读取新的预测结果，并根据本文 2.4 节设计的基于预测的弹性伸缩策略判断是否需要进行弹性伸缩，并根据相应的公式决策出具体需要增加几台虚拟机、或减少几台虚拟机.如果本集群需要进行

弹性伸缩，执行模块会收到具体的扩容或缩容指令，包括具体的扩缩容虚拟机数目，进而调用 CloudSim 自身的接口完成增减虚拟机操作.

实验服务器配置为： Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz 处理器、48 核心，2 块 RAM 16G DDR4@2400MHz 内存，1 块 NVIDIA Corporation GP104 [GeForce GTX 1070 Ti] 图像处理器。

本文训练数据、测试数据均每 5 秒设置一个采样点或者预测点，即每 5 秒一个数据点.图 5~图 9 中时间序列表示是数据点的序列，例如：100 表示第 100 个数据点.

3.2 CNN-LSTM-GRU预测模型实验结果与分析

本实验所采用的数据集源自 Bit-brain 云服务提供商，包含 1750 个虚拟机的性能指标.数据集文件以逐行组织的形式呈现，每行包括有关 CPU、内存、磁盘、网络等资源的分配与利用率等指标（数据集链接 http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains）.在本文的工作中，选取该数据集中的‘Memory usage’一列作为负载预测模型的训练与测试数据，该列反映了以 MB 为单位的活跃内存使用情况.该数据集中内存负载利用率主要分布在 60%至 80%的范围内，与本文研究的弹性伸缩应用环境相契合.

对于模型的性能评价指标，本文选择均方误差(Mean Squared Error, MSE)、平均绝对百分比误差(Mean Absolute Percentage Error, MAPE)、均方根误差(Root Mean Squared Error, RMSE)、平均绝对误差(Mean Absolute Error, MAE)、决定系数R^2五个指标.

本文根据表 1 所设置的超参数对预测模型进行训练.

表 1 模型的超参数设置

超参数	参数取值
时间窗口、步长	8
CNN 核大小	2
第一层 LSTM 隐藏单元数	64
第二层 LSTM 隐藏单元数	64
第三层 GRU 隐藏单元数	64
dropout 比例	0.4
学习率	0.001

本文的 CNN-LSTM-GRU 组合模型对于时序负载的预测结果如图 5 所示.

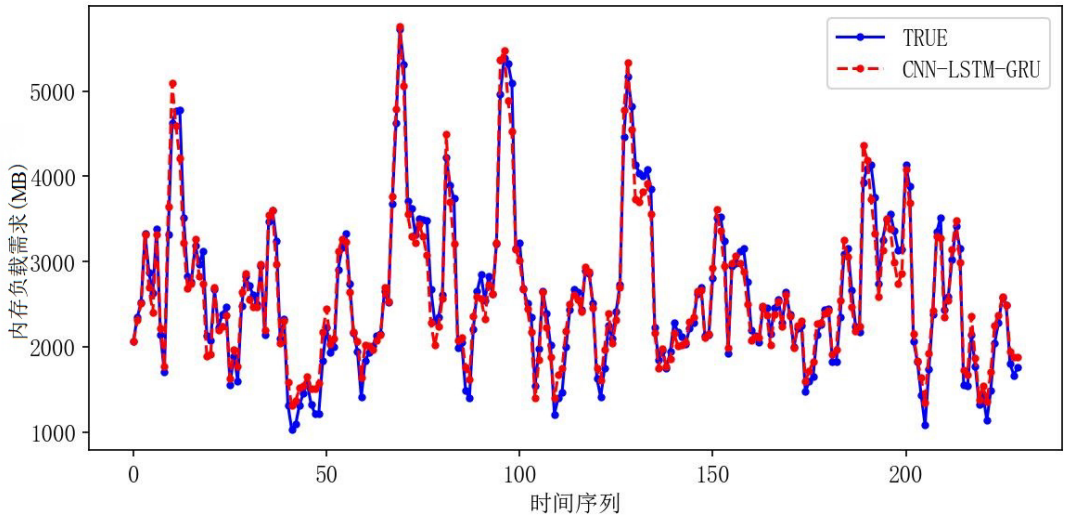

图 5 CNN-LSTM-GRU 组合模型负载内存需求预测结果

在图中以红色折线展示了本文提出的 CNN-LSTM-GRU 组合模型在时序负载预测任务中的表现，而蓝色折线则呈现了测试集的真实资源需求，图中横坐标表示测试集中的时序负载样本，纵坐标表示内存负载的需求值.数据集中真实值呈现迅速变化态势，上升和下降趋势频繁而急速，具备显著的波动性，因而预测挑战相对较高.根据实验结果，红色折线的跟随性与蓝色折线的变化趋势基本保持一致，表明本文模型在捕捉时序负载特征与变化方面效果显著.

为了验证 CNN-LSTM-GRU 组合模型的预测效果，本文实现了长短时记忆网络(LSTM)和门控循环单元(GRU)预测模型，并与这些单一模型的效果进行了对比.文献[16]中设计了一种融合了卷积神经网络和长短时记忆网络的组合模型 CNN-LSTM，旨在进行与本研究相似的时序负载预测任务.本文也实现了该模型，并使用本文构建的数据集进行了训练和测试.

根据实验结果数据，分别计算四组实验的 MSE、MAER、MSE、MAPE、R^2 等五个指标，并且记录每组实验的训练时间，结果如表 2 所示.

表 2　模型评价指标对比

模型	MSE	MAE	RMSE	MAPE(%)	R2	时间成本(s)
LSTM	12.425	2.853	3.525	8.299	0.938	61
GRU	16.587	3.012	4.073	8.096	0.918	40
CNN-LSTM	9.131	3.313	3.022	6.419	0.955	51
本文模型	**8.337**	**2.232**	**2.887**	**5.992**	**0.960**	48

根据表 2，相较于独立的 LSTM 与 GRU 模型，本文设计的 CNN-LSTM-GRU 组合模型在四个维度的指标上均展现出更高的预测精度.此外，本组合模型还表现出更为优越的拟合程度（R2），相对于单一模型的预测结果更加贴近真实数据.尽管本模型在时间成本上略高于独立的 GRU 模型，这是因为 GRU 采用了更为简化的网络结构，简化了计算过程以换取更快的训练速度，但本文模型在精度上的提升有效弥补了这一短板.与文献[16]中提出的组合模型 CNN-LSTM 进行对比，本文的模型在拟合度方面与其相近，然而在误差指标上却表现得更为优越.这一优势可归因于本文模型引入了 GRU 模块，从而在一定程度上提升了模型的学习能力，并且实现了更迅速的训练时间.综上所述，本文设计的 CNN-LSTM-GRU 组合模型充分融合了 CNN 在提取时序数

据深度特征方面的优越性、LSTM 在高预测精度上的长处，以及 GRU 在快速训练方面的特点.相较于独立模型和当前已有的组合模型，该模型的性能基本上更为出色，其中，预测误差降低了 8.7%至 21.8%，而预测拟合度提高了 4.6%.所以，本文模型可以胜任基于负载预测的弹性伸缩策略中的时序负载预测工作.

3.3 基于参数调优的预测模型实验结果与分析

使用改进后的贝叶斯优化算法 HBO 对 CNN-LSTM-GRU 组合模型寻找最优超参数，超参数包含：（1）连续超参数：时间窗口步长、学习率、dropout 比例；（2）离散超参数：隐藏单元数、批处理大小、核大小.. 六个超参数具体的范围和初始值如表 3 所示.

表 **3** 调优前模型参数初值

超参数	初始取值	取值范围/空间
时间窗口、步长	8	[5,20]
学习率	0.001	[1e-5,1e-1]
dropout 比例	0.4	[0.1,0.5]
隐藏单元数	64	（32,64,128,256）
批处理大小	16	（16,32,64,128）
核大小	2	（2，3，5）

在超参数解空间中，本文设定寻找最优解的最大迭代次数为 50.在每一轮迭代中，针对当前轮次的参数解（采样点），先将其应用于本文提出的 CNN-LSTM-GRU 组合模型，并进行 20 次训练，随后计算当前解对应的性能指标均方误差（MSE），以评估当前解的优劣.为了进行实验对比，本文选择文献[17]中未经改善的贝叶斯优化算法 BO 作为对比.经过 50 次迭代之后，两种调优算法所给出的最优解如表 4 所示.

表 **4** 最优超参数解对比

超参数	HBO 最优解	BO 最优解
时间窗口、步长	**19**	11
学习率	**0.001**	0.0087
dropout 比例	**0.41**	0.46
隐藏单元数	**256**	64
批处理大小	**32**	32
核大小	**3**	2

为了验证本文所改进的贝叶斯优化算法（HBO）对组合模型有更优的调参效果，应用上述两组超参数解构建 CNN-LSTM-GRU 模型，进行训练和负载预测.所采用的训练集和测试集与 3.2 节相一致.负载预测结果的对比参见图 6.模型评价标准的统计结果，包括均方误差（MSE）、平均绝对误差（MAE）、均方根误差（RMSE）、平均绝对百分比误差（MAPE）、以及决定系数（R^2），如表 5 所示.

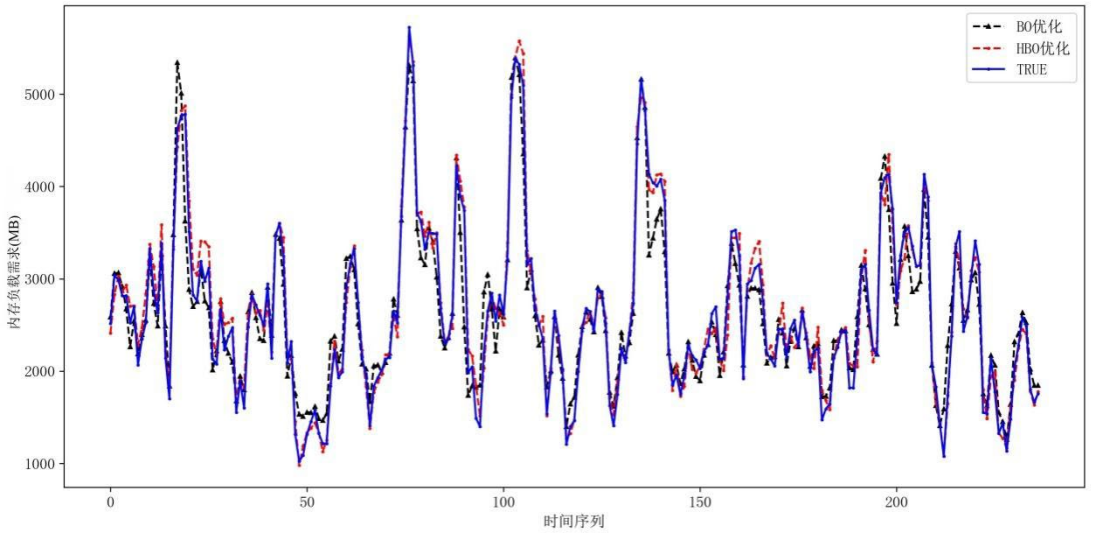

图 6 HBO 优化方法与 BO 优化方法调参效果对比

表 5 评价指标对比

模型	MSE	MAE	RMSE	MAPE(%)	R^2
CNN-LSTM-GRU	8.337	2.232	2.887	5.992	0.960
BO 调优	8.358	2.932	2.890	6.860	0.918
HBO 调优	**5.006**	**1.742**	**2.237**	**4.544**	0.97

结合表 5 中的数据分析，未改进的贝叶斯优化算法在与初始模型相比的调优效果方面表现较差.MSE 与 RMSE 两个误差指标与初始模型相近，而 MAE 与 MAPE 指标则呈现出更高的误差水平，此外，对于训练集的拟合程度也减少了 4.4%.这些数据证明了原始的贝叶斯优化算法在同时处理离散解和连续解的组合解空间方面存在局限性.并且，“隐藏单元数”与“核大小”这两个离散参数的最终取值未发生改变，这也更加证明了原始的贝叶斯优化算法难以有效应对本文场景.

与之相对，本文所设计的改进贝叶斯优化算法（HBO）更为有效地解决了复杂参数空间的问题.通过表 5 中的数据分析，HBO 所提供的最优参数解明显优化了四种误差指标，其预测精度均高于初始模型及 BO 调优的结果，使得误差再次降低 7.6%，并且拟合度提高 1.04%.从图 6 中直观的结果来看，HBO 优化在测试集上所得的预测结果（红色折线）呈现出精准的拟合结果，与真实测试集趋势变化及极值变化（蓝色折线）高度契合.这充分证实了使用优化算法对组合模型参数进行调优能够全面提升模型的预测精度.

3.4 基于内存负载预测的弹性伸缩方法实验结果与分析

本部分实验同样选用 Bit-brain 云服务提供商的负载数据集，选取该数据集中的一系列‘Memory usage’数据作为内存需求时序负载，这一部分的数据反映了以 MB 为单位的虚拟机集群中的活跃内存使用情况.作为弹性伸缩实验所选择的数据与用来训练与测试组合预测模型的数据没有重合部分，以此更具说服力的验证本文所提出的基于负载预测的弹性伸缩策略的效果，增强实验结果的可信度.

实验所依赖的组合预测模型提前被训练完毕，并且由本文改进的 HBO 参数调优算法确定最优超参数解集.伸缩决策基于 2.4 节所设计的弹性伸缩策略，其中α与β的取值分别为 90%、70%，目的是尽可能保证内存资源处于较高的利用率.

本文选用了文献[2]中设计的弹性伸缩策略作为对比实验的基准，以评估本文提出的基于负载预测的弹性

伸缩方法的性能.该策略采用基于阈值的响应式伸缩方法，目前已广泛应用于 Kubernetes 系统，具有相当的代表性.同样地，本文在 CloudSim 中重新实现了该策略，并将其负载适配为本文所采用的内存负载，以确保实验的一致性和可比性.

在实验设置方面，本实验设定 CloudSim 集群中一个的虚拟机可以提供 512MB 的内存资源，根据负载数据分布，一共创建了 10 个可供伸缩的虚拟机.初始状态下，虚拟机数量设定为 9 个.本实验实施了对虚拟机数量、集群响应时间以及集群延迟的监控与收集，作为本实验的评价指标，其中，响应时间被定义为从提交租户任务到完成所需的时间，延迟是指所有处理过的租户任务的期望响应时间与实际响应时间之间的差异.

为集群添加负载并分别执行本文的弹性伸缩策略与文献[2]的策略进行对比，所得到的两组实验结果如图 7 和图 8 所示.

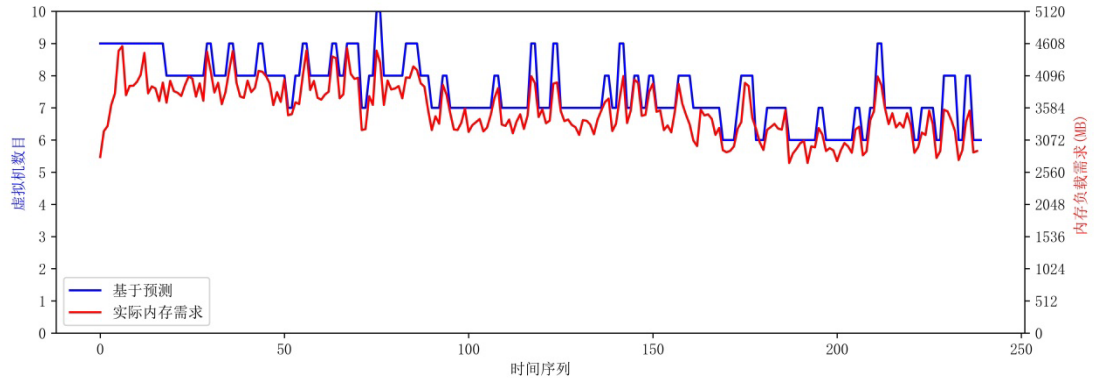

图 7 本文伸缩策略实验结果

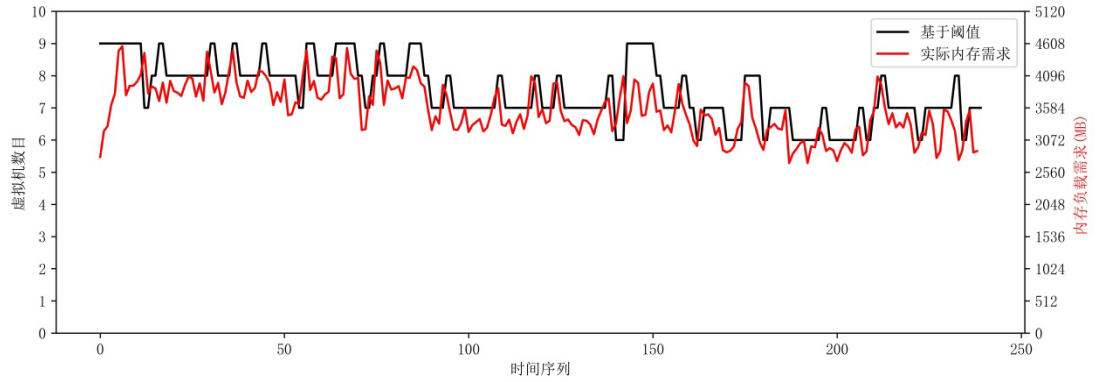

图 8 文献[2]伸缩策略实验结果

图 7 展示了本文弹性伸缩策略对负载变化的应对情况.红色折线表示内存负载随时间的变化，对应图中右纵轴，蓝色折线表示集群中虚拟机的数量，对应左纵轴.在本文弹性伸缩策略所依赖的组合预测模型中，预测窗口的时间步长被设置为 19，即基于前 19 个时间点的负载数据预测下一时刻的负载需求，故本弹性伸缩策略在第 19 个时间点正式启动，在此之前一直保持初始的 9 个虚拟机数目不变，呈现出未感知负载的状态.本文弹性伸缩策略自启动至结束，展现出对负载变化趋势的较为准确预测，并能精确地调整集群中虚拟机的数量，可以准确保证整个集群中的内存资源处于较为合理的利用区间中.

文献[2]基于阈值的弹性伸缩策略对负载变化的应对情况如图 8 所展示.相似地，图中黑色折线表示集群中虚拟机数目变化，对应左纵轴，红色折线表示负载变化，对应右纵轴.基于阈值的策略不需要“启动时间”，故首次调整虚拟机数目的时间点早于本文策略.但是该策略难以提前预知负载的变化，采取增减虚拟机的时刻

总是落后于负载变化趋势，甚至会在负载需求较低的情况下仍存在过多的可用资源.相较于本文提出的负载预测策略，基于阈值的策略在应对负载波动方面存在一定的滞后性和资源利用效率不足.

为了更进一步的比较本文基于预测的策略与对比实验基于阈值的策略在性能方面的优劣，本文记录了虚拟机集群在弹性伸缩过程中响应时间和延迟的变化结果，如图 9 所示.

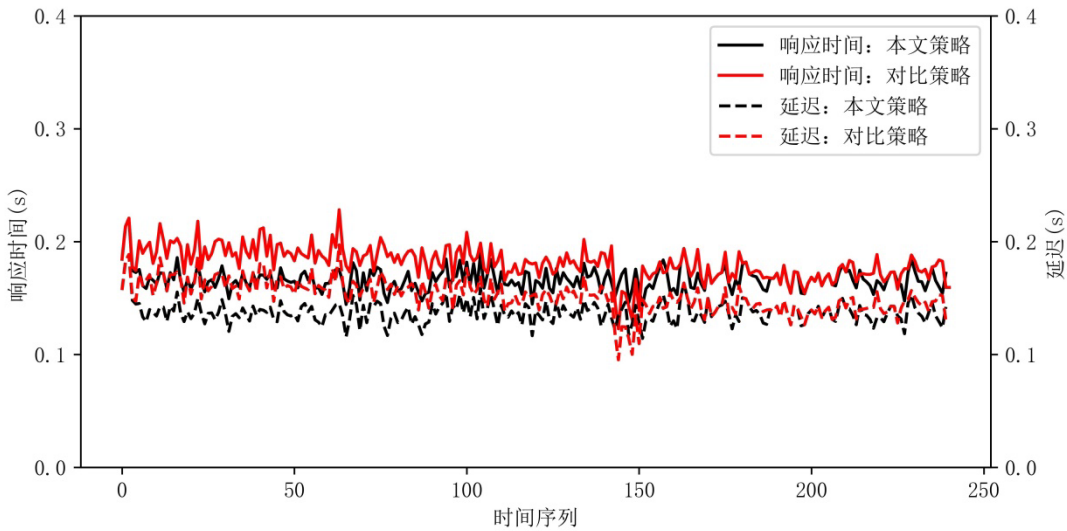

图 9 本文算法、文献[2]策略响应时间、延迟对比

图 9 中黑色实线和黑色虚线分别表示本文策略的响应时间和延迟，而红色实线和虚线则代表对比策略的响应时间和延迟，两个指标分别对应于左、右纵轴.结合如图 7 和图 8 来看，本文策略对于虚拟机数量的调整更为精准，更能与负载的变化相匹配，从而使资源得到及时和合理的利用.因此在图 9 中，两条黑色折线所代表的响应时间与延迟均处于较低水平.从数据层面的分析来看，对于本文策略，平均响应时间为 0.1664 秒，平均延迟为 0.1372 秒；而对于对比策略，其平均响应时间为 0.1811 秒，平均延迟为 0.1517 秒.本文策略的响应时间降低 8.12%，延迟降低 9.56%.

4 总 结

本文首先提出了一种 CNN-LSTM-GRU 组合模型，结合 CNN 善于提取时序数据深度特征、LSTM 预测精度高、GRU 预测时间短的优点对负载进行预测.并针对贝叶斯优化算法难以处理离散、连续混合参数的调优问题，改进了的贝叶斯优化算法中的协方差函数，基于改进后的贝叶斯算法对组合模型进行超参数调优.

实验证明本文提出的组合预测模型、基于参数调优的预测模型以及基于内存负载预测的弹性伸缩方法均具有良好的性能与较好的优越性，具体如下：

（1）相较于现有预测模型，本文 CNN-LSTM-GRU 组合预测模型预测误差降低 8.7%-21.8%，预测拟合度提高 4.6%；

（2）本文提出的 HBO 优化算法能更迅速找到最优参数解，调优后的预测模型误差进一步降低 7.6%，拟合度提高 1.04%；

（3）与 Kubernetes 中应用最广泛的伸缩策略相比，本文方法避免了弹性伸缩的滞后性与资源浪费，同时响应时间降低 8.12%，延迟降低 9.56%.

References:

[1]Zhao H, Lim H, Hanif M, Lee C. Predictive container auto-scaling for cloud-native applications. 2019 International Conference on Information and Communication Technology Convergence (ICTC). Jeju Island, Korea (South): IEEE, 2019: 1280 – 1282. [doi:10.1109/ICTC46691.2019.8939932].

[2]Nguyen T-T, Yeom Y-J, Kim T, Park D-H, Kim S. Horizontal pod autoscaling in kubernetes for elastic container orchestration. Sensors, 2020, 20(16): 4621. [doi:10.3390/s20164621].

[3]Sellami W, Hadj Kacem H, Hadj Kacem A. Dynamic provisioning of service composition in a multi-tenant saas environment. Journal of Network and Systems Management, 2020, 28(2): 367 – 397. [doi:10.1007/s10922-019-09510-2].

[4]Zafeiropoulos A, Fotopoulou E, Filinis N, Papavassiliou S. Reinforcement learning-assisted autoscaling mechanisms for serverless computing platforms. Simulation Modelling Practice and Theory, 2022, 116: 102461. [doi:10.1016/j.simpat.2021.102461].

[5]Nouri SMR, Li H, Venugopal S, Guo W, He M, Tian W. Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. Future Generation Computer Systems, 2019, 94: 765 – 780. [doi:10.1016/j.future.2018.11.049].

[6]Ghobaei-Arani M, Jabbehdari S, Pourmina MA. An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach. Future Generation Computer Systems, 2018, 78: 191 – 210. [doi:10.1016/j.future.2017.02.022].

[7]Horovitz S, Arian Y. Efficient cloud auto-scaling with sla objective using q-learning. 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud). Barcelona, Spain: IEEE, 2018: 85 – 92. [doi:10.1109/FiCloud.2018.00020].

[8]Wei Y, Kudenko D, Liu S, Pan L, Wu L, Meng X. A reinforcement learning based auto - scaling approach for saas providers in dynamic cloud environment. Mathematical Problems in Engineering, 2019, 2019(1): 5080647. [doi:10.1155/2019/5080647].

[9]Wu X, Zhang C, Yuan S, et al. Blended elastic scaling method for cloud resources following reinforcement learning. Journal of Xi'an Jiaotong University, 2022, 56(1): 142-150.[doi:10.7652/xjtuxb202201016].(in Chinese with English abstract)

[10]Song S, Pan L, Liu S. A q-learning based auto-scaling approach for provisioning big data analysis services in cloud environments. Future Generation Computer Systems, 2024, 154: 140–150. [doi:10.1016/j.future.2024.01.003].

[11]Kardani-Moghaddam S, Buyya R, Ramamohanarao K. ADRL: a hybrid anomaly-aware deep reinforcement learning-based resource scaling in clouds. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(3): 514 – 526. [doi:10.1109/TPDS.2020.3025914].

[12]Garí Y, Monge DA, Pacini E, Mateos C, García Garino C. Reinforcement learning-based application autoscaling in the cloud: a survey. Engineering Applications of Artificial Intelligence, 2021, 102: 104288. [doi:10.1016/j.engappai.2021.104288].

[13]Yu Y, Si X, Hu C, Zhang J. A review of recurrent neural networks: lstm cells and network architectures. Neural Computation, 2019, 31(7): 1235 – 1270. [doi:10.1162/neco_a_01199].

[14]Hoseinzade E, Haratizadeh S. CNNpred: cnn-based stock market prediction using a diverse set of variables. Expert Systems with Applications, 2019, 129: 273 – 285. [doi:10.1016/j.eswa.2019.03.029].

[15]Vidal A, Kristjanpoller W. Gold volatility prediction using a cnn-lstm approach. Expert Systems with Applications, 2020, 157: 113481. [doi:10.1016/j.eswa.2020.113481].

[16]Ouhame S, Hadi Y, Ullah A. An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model. Neural Computing and Applications, 2021, 33(16): 10043 – 10055. [doi:10.1007/s00521-021-05770-9].

[17]Du L, Gao R, Suganthan PN, Wang DZW. Bayesian optimization based dynamic ensemble for time series forecasting. Information Sciences, 2022, 591: 155 – 175. [doi:10.1016/j.ins.2022.01.010].

附中文参考文献:

[9]吴晓军,张成,原盛,等. 基于强化学习的云资源混合式弹性伸缩算法. 西安交通大学学报,2022,56(1):142-150.[doi:10.7652/xjtuxb202201016].