基于EMD-LSTM的分子谱线数据预处理容器负载分组预测算法

冶鑫晨[1,2,3]，张海龙[1,2,3,4]，王 杰[1,3]，李大磊[1]，张 萌[5]，张亚州[1,2]，杜 旭[1,2]，李 嘉[1]，王万琼[1]

（1. 中国科学院 新疆天文台，乌鲁木齐 830011；2. 中国科学院大学，北京 100049；3. 国家天文科学数据中心，北京 100101；4. 中国科学院 射电天文重点实验室，南京 210008；5. 曲阜师范大学 网络空间安全学院，山东 曲阜 273165）

摘 要：集群环境中容器资源分配不均衡是当前亟待解决的问题，针对容器负载预测及资源分配策略，本文设计了基于经验模态分解-长短期记忆网络的天文数据处理容器负载分组预测算法，提出了基于预测负载信息的自适应推荐值生成算法，可根据负载波动程度自动分配容器计算资源。利用模拟数据及真实天文观测数据进行了负载预测准确度验证，实验结果表明本文提出的算法相较于三重指数方法和单一长短期记忆网络模型具有更高的预测准确度。在天文数据实时预处理测试中，相较于默认策略，本文提出的推荐值生成算法可有效提升计算资源的使用效率。

关键词：天文信息技术；容器；负载预测；长短期记忆网络；负载均衡

中图分类号：TP301.6 **文献标志码**：A **文章编号**：1671-5497(2025)04-1374-10

DOI：10.13229/j.cnki.jdxbgxb.20230690

EMD-LSTM-based group prediction algorithm of container resource load in preprocessing molecular spectral line data

YE Xin-chen[1,2,3], ZHANG Hai-long[1,2,3,4], WANG Jie[1,3], LI Da-lei[1], ZHANG Meng[5], ZHANG Ya-zhou[1,2], DU Xu[1,2], LI Jia[1], WANG Wan-qiong[1]

(1. *Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi* 830011, *China*; 2. *University of Chinese Academy of Sciences, Beijing* 100049, *China*; 3. *National Astronomical Data Center, Beijing* 100101, *China*; 4. *Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing* 210008, *China*; 5. *School of Cyber Science and Engineering, Qufu Normal University, Qufu* 273165, *China*)

Abstract: Unequal allocation of container resources in a cluster environment is currently a pressing issue. In

收稿日期：2023-07-03.

基金项目：国家重点研发计划项目(2021YFC2203502, 2022YFF0711502)；国家自然科学基金项目(12173077, 12003062)；中国科学院“西部之光”人才培养引进计划项目(xbzg-zdsys-202410)；“天山英才培养”计划项目(2022TSYCCX0095, 2023TSYCCX0112)；中国科学院科研仪器设备研制项目(PTYQ2022YZZD01)；国家天文科学数据中心项目；中国科学院天文台站设备更新及重大仪器设备运行专项项目；新疆维吾尔自治区自然科学基金项目(2022D01A360).

作者简介：冶鑫晨(1991-)，男，博士研究生. 研究方向：天文大数据. E-mail: yexinchen@xao.ac.cn

通信作者：张海龙(1980-)，男，正高级工程师，博士. 研究方向：天文大数据. E-mail: zhanghailong@xao.ac.cn

response to container load prediction and resource allocation strategies, this paper proposes an empirical mode decomposition-long short-term memory (EMD-LSTM)-based algorithm for predicting container resource load groups in the preprocessing of astronomical data. An adaptive recommendation value generation algorithm based on load prediction information is introduced, which automatically allocates container computing resources according to the degree of load fluctuation. The accuracy of load prediction was verified using simulated data and real astronomical observation data. Experimental results demonstrate that the proposed algorithm outperforms the triple exponential smoothing method and a single LSTM network model in terms of prediction accuracy. In real-time preprocessing of astronomical data, compared to the default strategy, the recommended value generation algorithm proposed in this paper effectively improves the utilization efficiency of computing resources.

Key words: astroinformatics; container; load prediction; long short-term memory; load balancing

0　引　言

由于具有灵活、一致、可重复的特性，容器技术已经在科学数据处理中得到广泛应用。随着信息技术的飞速发展，大数据处理所使用的软件日趋复杂，研究人员广泛使用容器技术实现数据处理环境部署。陈红松等[1]针对物联网大规模分布式拒绝服务攻击检测难题，基于Docker虚拟化容器技术搭建了物联网流量仿真平台。Morris等[2]在容器环境中测试了不同天文软件运行情况，研究表明容器技术可以帮助科研人员重新配置数据处理环境并生成测试结果。Molenaar等[3]创建了基于Docker的容器框架(Kliko)，用于运行一个或多个相关的计算作业，并实现了基于Web的容器调度器和专门用于天文数据输出的可视化工具。Herwig等[4]基于Jupyter和Docker开发了CyberHubs系统，应用在恒星流体动力学模拟、恒星演化、银河系化学演化的数据处理中。

集群环境中利用容器技术进行大数据处理存在资源调度滞后问题。为实现集群中容器高效管理与资源分配，Swarm、Mesos、Kubernetes等容器编排管理系统应运而生。Kubernetes是目前主流的容器管理工具[5]，其通过横向自动伸缩(horizontal pod autoscaling, HPA)与纵向自动伸缩(vertical pod autoscaler, VPA)解决负载波动问题，以提高集群的资源使用效率。利用容器技术进行的科学计算多为一次性提交作业，容器在完成当前分配的计算任务后终止运行，但对于天文观测数据预处理、暂现源搜寻[6,7]、射频干扰消除[8]等任务，负责数据处理的容器需要长期运行，并会因观测计划、终端、任务目标等因素导致负载变化。

VPA算法适用于稳定的工作负载，当更新频繁时，对有状态和性能敏感的应用程序可能造成程序中断、执行出错等问题[9]。通过利用特定的预测算法预测资源需求，提前进行资源分配优化，可以提高资源利用率和系统稳定性[10]。Rzadca等[11]利用所开发的Autopilot自动配置资源，利用滑动窗口和机器学习预测负载情况，从而调整作业中的并发任务数量(水平扩展)和单个任务的CPU/内存限制(垂直扩展)。Shanmugam[12]利用Amazon AWS的CPU利用率数据训练ARIMA模型，解决了Docker容器的CPU利用率预测问题。对于时间序列预测的问题，Bandara等[13]给出了新的研究思路——通过聚类提高全局模型的准确性，对时间序列先聚类再进行预测，可以得到更好的预测效果。然而在实时处理的情况下，需要在短时间内进行预测，聚类操作需要对整个数据集进行分析和计算，无法满足实时预测需求。

本文通过K-means对天文数据处理负载信息序列进行聚类，建立了Fits元数据与负载分组关系。对需要预测的负载信息序列利用关系方程进行分组，使用经验模态分解(Empirical mode decomposition, EMD)对负载信息进行分解，筛选分量生成样本集，利用长短期记忆神经网络(Long short-term memory, LSTM)进行预测。利用容器负载的预测信息，设计了自适应的VPA推荐值生成算法，并将其应用于容器资源分配。

1　VPA与预测算法

1.1　Kubernetes VPA

Kubernetes VPA可以根据容器的资源使用

情况自动设置CPU和内存的限值,既可以在资源过剩时收缩容器的使用资源,也可以在资源不足时提升资源限制,VPA的架构如图1所示。指标服务器收集Pod的负载信息并存储。Recommender组件获取绑定Pod的CPU与内存的负载信息,结合历史信息给出容器的资源使用量推荐值。Update组件负责监控推荐值的变化,当推荐值与Pod正在使用的值差距过大时,Update组件会删除当前Pod。而Admission Controller组件则负责Pod的重建。

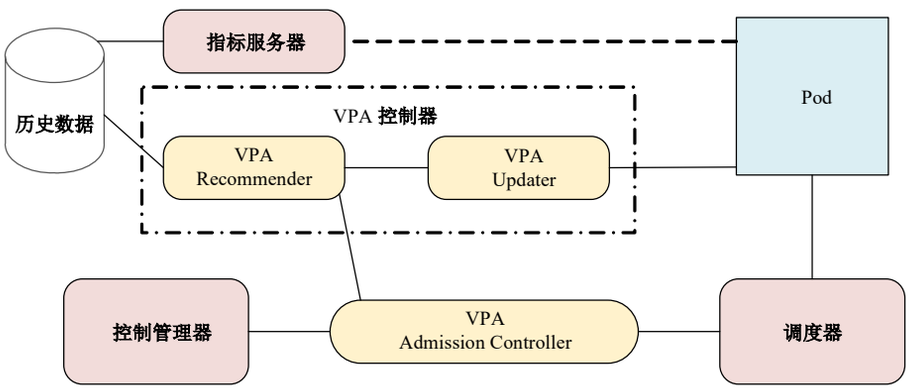

图1 VPA架构图

Fig. 1 VPA architecture diagram

控制容器资源的关键在于Recommender组件所生成的推荐值,Recommender设计理念是假设内存和CPU利用率是独立的随机变量,根据应用当前的资源使用情况以及历史的资源使用情况,计算接下来可能的资源使用阈值。

(1)对于CPU,目标是将容器使用率超过请求时间的部分保持在某个阈值以下。

(2)对于内存,目标是将特定时间窗口内容器使用率超过请求的概率保持在某个阈值以下。

这种阈值控制的方法使VPA在周期性变化的工作负载上通常表现不佳。

1.2 指数平滑算法

指数平滑算法[14]是基于移动平均算法改进的一种时间序列分析预测方法。指数平滑法有几种不同形式:一次指数平滑法用于预测没有趋势和季节性的时间序列;二次指数平滑法用于预测有趋势但没有季节性的序列;三次指数平滑法用于有趋势也有季节性的序列。

一次指数平滑预测公式为:

$$y'_{t+1}=\alpha y_t+(1-\alpha)y'_t \tag{1}$$

式中:y'_{t+1}为$t+1$周期的预测值;y_t为t周期的实际值;y'_t为t周期的预测值;α为平滑系数。先前预测值会对当前的预测值产生影响,其作用随α的增大而减小。

二次指数平滑预测公式为:

$$S_t^{(2)}=\alpha S_t^{(1)}+(1-\alpha)S_{t-1}^{(2)} \tag{2}$$

式中:$S_t^{(2)}$为第t周期的二次指数平滑值;$S_t^{(1)}$为第t周期的一次指数平滑值;$S_{t-1}^{(2)}$为第$t-1$周期的二次指数平滑值;α为平滑系数。二次指数平滑法不能单独预测,需与一次指数平滑法配合,建立预测的数据模型。

三次指数平滑预测公式为:

$$S_t^{(3)}=\alpha S_t^{(2)}+(1-\alpha)S_{t-1}^{(3)} \tag{3}$$

与二次指数平滑预测方法相似,三次指数平滑是在二次指数平滑的基础上再进行一次平滑。

1.3 LSTM

LSTM是一种具有长期记忆能力的循环神经网络(Recurrent neural network, RNN),其网络中具有多个遗忘/记忆功能的单元组成。RNN由于会存储之前时刻所有信息,因此会存在消失梯度等问题,而LSTM拥有门控模块,可以选择性地存储信息。一般的LSTM单元如图2所示,过程分为4个步骤。

步骤1 决定遗忘的记忆内容。

$$f_t=\sigma\left(W_f\cdot\left[h_{t-1},x_t\right]+b_f\right) \tag{4}$$

步骤2 决定添加的记忆内容。

$$i_t=\sigma\left(W_i\cdot\left[h_{t-1},x_t\right]+b_i\right) \tag{5}$$

$$\tilde{C}_t=\tanh\left(W_C\cdot\left[h_{t-1},x_t\right]+b_C\right) \tag{6}$$

步骤3 更新当前记忆内容。

$$C_t=f_t\times C_{t-1}+i_t\times\tilde{C}_t$$

步骤4 输出。

$$o_t=\sigma\left(W_o\cdot\left[h_{t-1},x_t\right]+b_o\right) \tag{7}$$

$$h_t=o_t*\tanh\left(C_t\right) \tag{8}$$

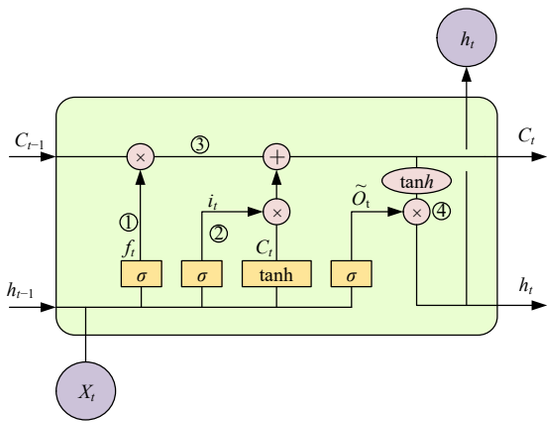

图 2　LSTM 单元

Fig. 2　LSTM unit

1.4　EMD

EMD是一种数据分解方法，常用于信号处理和数据分析领域。EMD的目标是将复杂的信号分解成一组称为本征模式函数（Intrinsic mode functions，IMFs）的基本成分，这些IMFs具有不同的频率和振幅特性，从而能够更好地揭示数据的内在结构和特征。

将EMD与预测算法结合可以用于预测非线性和非平稳数据，并提高时间序列预测的准确性。

2　基于EMD-LSTM的分组预测算法

2.1　分组预测算法架构设计

本文所设计的预测算法分为两个部分：分组和预测，算法整体架构设计如图3所示。分组部分对历史负载信息进行聚类，通过Fits元数据生成分组算法。当前负载信息需要进行预测时，通过分组算法进行分组，在分组内匹配对应的历史负载信息。对完成分组的负载数据利用EMD进行分解，利用LSTM预测每个分量的下一周期值，得到预测结果。本文所调用的Python库如表1所示。

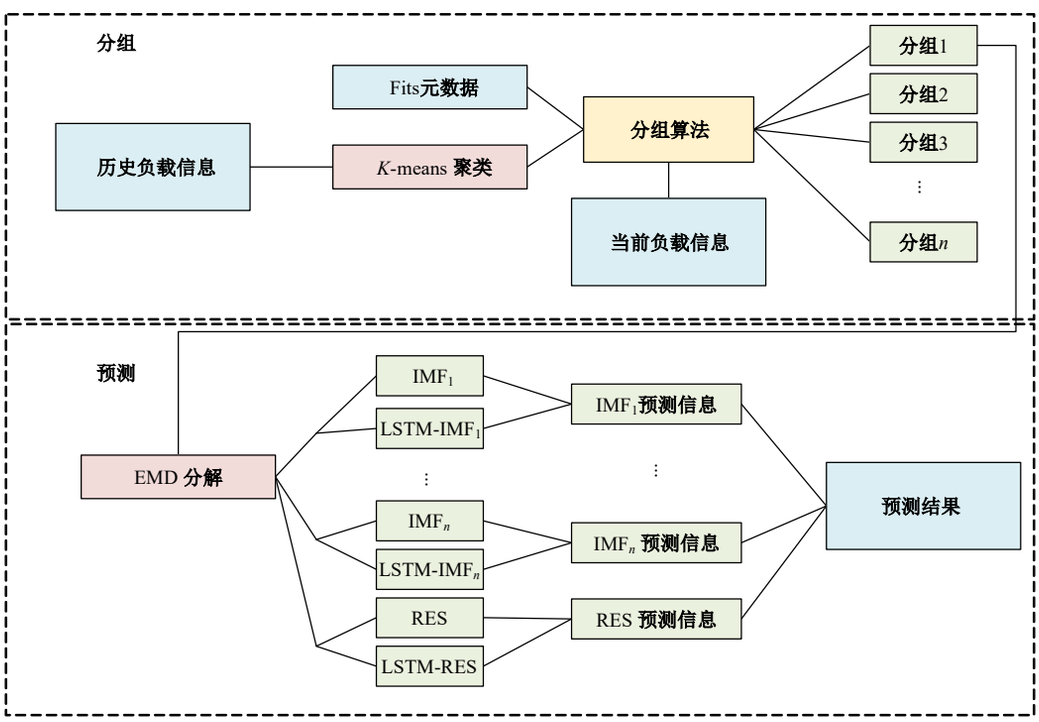

图 3　EMD-LSTM 分组预测算法架构

Fig. 3　EMD-LSTM group prediction algorithm architecture

表 1　Python 库信息

Table 1　Python library information

编号	库名称	功能
1	Numpy	数据处理
2	PyEMD	EMD分解
3	Keras	构建LSTM预测模块

2.2　分组预测算法实现

对历史负载信息序列进行聚类分组，可以在一定程序上提高预测准确度。但是容器的负载信息预测是动态过程，在指定的时间间隔中需要完成下一周期的负载预测，而每次预测都将新增的负载序列与历史负载序列进行聚类，无法满足实时要求。

利用Fits元数据与负载的潜在关系实现对新增负载序列的分组,匹配对应的历史负载再进行负载信息的预测,具体分组算法如下:

(1)利用K-means进行负载信息的分组,通过计算不同聚类数的评分值M_k,确定选取的聚类数K:

$$M_k=\sum_{i=1}^{n}\left(\frac{S_i^1+S_i^2+\cdots+S_i^j}{N_{\text{Meta}}}\right),3\leqslant k\leqslant 20 \tag{9}$$

式中:N_{Meta}为数据总个数;S_i为第i个Fits元数据条目相同且数量占比超过总数5%的数据个数,满足条件的分组为j,取最大评分值$\max(M_k)$对应的聚类数为最终选定的聚类数。

(2)新增负载信息将通过C_k值判定分组:

$$C_k=\sum_{i=1}^{n}P_i\times T_i\begin{cases}T_i=(0,1)\\ P_i=N_{\text{Meta}}^i\times\left(\dfrac{S_i^t}{\sum\limits_{m=1}^{n}S_m^t}+1\right)^2\\ S_i^t=S_i^1+S_i^2+\cdots+S_i^j\end{cases} \tag{10}$$

计算分组历史数据的第i个Fits元数据条目与新增数据相同的占比,如果超过动态阈值则T_i为1,否则为0;N_{Meta}^i为分组中第i个Fits元数据条目与新增数据相同的数据个数;将新增数据放入分组历史数据中重新计算S_i值,当前Fits元数据条目S_i^t值的占比会改变P_i值的大小;对新增的数据C_k值累加,选择$\max(\sum C_k)$所对应的分组为新增负载信息分组,将负载信息新增在所在分组历史负载信息中。分组之后的负载序列利用EMD将负载序列进行分解,EMD算法无须设定基函数,可直接根据数据在时间尺度上的特征进行分解。

将分组的负载信息时间序列分解为若干个单一频率的本征模函数IMF与残余分量(Residual,RES)。在分解过程中CPU负载和内存将转化为节点资源的占比,将IMF分量和RES残差序列分别作为LSTM模型的输入,进行训练和接收预测值。最终将各个IMF分量和RES残差序列预测结果进行合并,得到预测结果。

2.3 网络训练

(1)数据准备

分别对分组后时序负载数据的IMF分量与RES残差序列进行训练,选取非线性、非平稳负载。

(2)划分训练集和测试集

根据分组结果,将一定比例的数据作为训练集和测试集。

(3)创建输入特征与目标序列

选取时间步长$L=20$作为初始值,通过目标优化调整步长值。

(4)构建LSTM模型

使用Keras库构建LSTM模型,指定LSTM层神经元数量,并添加输出层用于产生输出结果。

(5)训练模型

设置训练周期数N与批次大小,对模型进行训练。

(6)模型评估

使用均方误差作为损失函数:

$$\text{MSE}=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2 \tag{11}$$

式中:y_i为实际值;$\hat{y}_i$为模型预估值;n为样本数量。将$\min\limits_{N}\text{MSE}$作为优化目标,更新网络权重。

3 推荐值生成与自动纵向伸缩架构

3.1 推荐值生成算法

Kubernetes默认通过历史负载实现自动纵向伸缩的资源分配,这种方法具有一定的滞后性,周期性变化的工作负载上通常表现不佳。负载出现剧烈抖动时会导致容器频繁启停,造成计算资源的浪费。

针对历史负载及预测负载信息,本文设计了推荐值生成算法,推荐值计算公式如下:

$$\text{Limit}=\frac{\text{Pre}_{n+1}}{2\times\text{Load}_n}(\text{Pre}_{n+1}+\text{Pre}_{n+2})+\left|1-\frac{\text{Pre}_{n+1}}{\text{Load}_n}\right|\text{His}_{\text{rec}} \tag{12}$$

式中:Pre_{n+1}为$n+1$的预测值;Pre_{n+2}为$n+2$的预测值;Load_n为当前时刻的实际负载值;His_{rec}为上一个周期内通过历史负载得到的推荐值。生成推荐值的权重由负载波动决定,即当负载波动较大时,推荐值的生成倚重预测负载;当负载波动平缓时,推荐值的生成倚重历史负载。

3.2 自动纵向伸缩架构设计

如图4所示,将VPA基于阈值的判定方法改为本文所设计的基于EMD-LSTM的分组预测算法。从Metrics Server获取的历史信息中对负

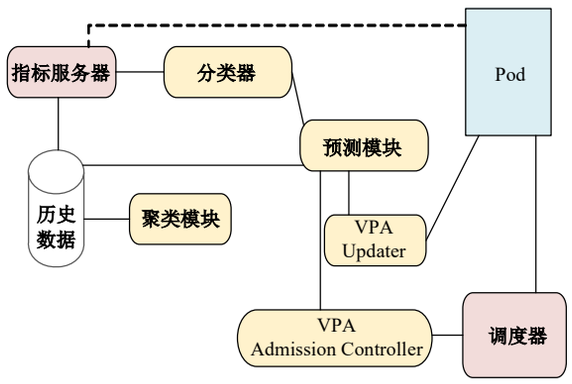

图4　重构后VPA架构图

Fig. 4　Revised VPA architecture diagram

载信息时间序列进行分组,按照所设定的监控周期获取当前周期的负载信息时间序列进行负载预测。

通过历史负载与预测值生成的新推荐值发送给VPA Updater和VPA Admission Controller,剩下的工作由Kubernetes VPA完成。

4　测试与分析

4.1　模拟负载信息预测

4.1.1　模拟负载信息生成

为测试分组预测算法的有效性,构造了由多个三角函数组成的模拟CPU负载生成函数,负载公式的通用形式为:

$$\mathrm{load}(t)=A_1\times\sin\left(\frac{2\pi\times t}{\tau_1}\right)+A_2\times\cos\left(\frac{2\pi\times t}{\tau_2}\right)+\cdots+A_n\times\cos\left(\frac{2\pi\times t}{\tau_n}\right)+C+K \tag{13}$$

式中:τ_n为不同周期;A_n为不同周期的振幅;C为负载的基线值;K为随机负载噪声。各负载公式的周期、振幅组合不同。

每一时刻的CPU负载由多个负载公式加权生成,即:

$$\begin{cases}\mathrm{load}(t)_{\mathrm{sim}}=\sum_{i=1}^{n}W_i\times\mathrm{load}(t)_i\\ \sum_{i=1}^{n}W_i=1\end{cases} \tag{14}$$

随机生成权值W_n,模拟24 h CPU负载的预测测试,权值每半小时更新一次。对生成的模拟CPU负载抽取不同权重的序列生成训练集,在预测中选取权重信息最相近的序列进行预测。

4.1.2　模拟负载信息预测对比

利用三重指数平滑方法、LSTM方法、分组预测方法分别对生成的24 h模拟CPU负载进行预测。窗口大小由交叉验证法确定,初始窗口设置为120 min,时间间隔设置为1 min,每次预测当前时刻后两个时间间隔的负载。

预测情况如图5所示,蓝色线为模拟CPU负载信息,红色线为预测信息。第一行为三重指数平滑方法预测情况,第二行为单一LSTM方法预测情况,第三行为基于LSTM的分组方法预测的情况。相关系数是一种用于衡量两组数据之间关联程度的统计指标,利用皮尔逊相关系数,计算预测数据和原始数据的线性相关性,如表2所示。

在模拟数据测试中,基于LSTM的分组预测方法预测准确度优于三重指数与单一LSTM方法,由于模拟数据负载信息复杂程度不高,基于分组预测的优势不明显。

表2　预测信息与原始数据相关系数

Table 2　Correlation coefficients between predicted information and original data

编号	方法	相关系数
1	三重指数平滑	0.988 59
2	LSTM	0.992 00
3	基于LSTM的分组	0.997 34

4.2　天文数据处理负载信息预测

4.2.1　天文数据处理测试pipeline

利用数据预处理pipeline校准新疆天文台南山25 m射电望远镜(NSRT)观测得到的甲醛分子谱线数据。数据预处理pipeline读取由数字终端XFB记录的原始数据,校准谱线数据的强度和速度。强度校准考虑了Tsys、增益曲线和大气透明度等因素,速度校准使用astropy软件包,将观测到的速度校准为本地静止标准参考系速度,校准后的谱线数据封装成Fits格式文件。

4.2.2　测试方法

将48 h甲醛分子谱线数据缩短观测间隔,模拟6 h较高负载数据实时处理,利用三重指数平滑方法、单一LSTM方法、基于EMD-LSTM的分组预测方法对实时预处理pipeline负载进行预测结果对比。将负载信息按照10 min为间隔进行分割,进行聚类并构造分组算法。进行预测时初始窗口设置为60,时间间隔设置为5 min,训练数据集由窗口内数据叠加所在组历史数据构成,每次预测当前时刻后两个时间节点的负载。

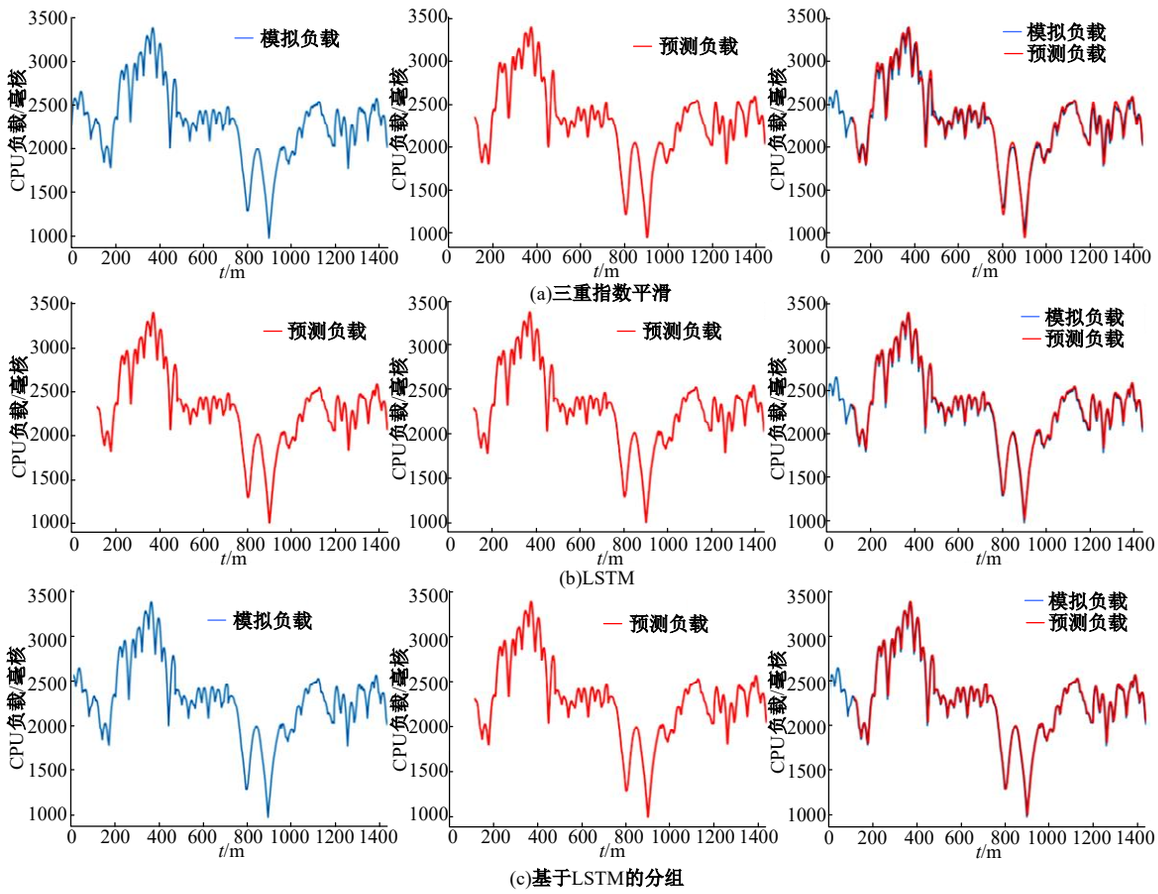

图5 模拟负载数据预测对比

Fig. 5 Comparison of predicted simulated load data

4.2.3 测试环境

测试所使用的计算节点硬件配置如表3所示。

数据处理软件封装在基于debian 11.6的Docker容器当中,单个Pod中运行一个容器,单节点中运行单个Pod。使用Kubernetes作为容器管理框架,运行的Pod默认开启VPA。

表3 节点硬件配置

Table 3 Node hardware configuration

名称	信息
中央处理器	Intel i9-10 900 K
内存	64 GB
软件	TEMPO2 2020.04.1
OS	Debian 10.10.0
网络接口	10 GbE
硬盘	1 TB NVMe SSD

4.2.4 数据集

本文使用南山25 m射电望远镜的甲醛分子谱线数据作为测试数据集,数据总规模为1 988个,数据总量为597 GB。数据具体信息如表4所示。

4.2.5 负载信息预测对比

利用三重指数平滑方法、单一LSTM方法、

表4 NSRT甲醛分子巡天数据信息

Table 4 NSRT hydrogen molecule sky survey data information

参数	值
望远镜	NSRT
文件格式	Fits
VEGAS终端模式	MODE9
基带带宽	450 MHz
量化精度	8 bits

基于EMD-LSTM的分组预测方法在单节点上对甲醛分子谱线数据实时预处理pipeline负载进行了预测结果对比,CPU预测结果如图6所示,内存预测结果如图7所示,蓝色线为实际负载信息,红色线为预测信息。第一行为三重指数平滑方法预测情况,第二行为传统LSTM方法预测情况,第三行为基于EMD-LSTM的分组方法预测情况。预测信息与实际负载相关系数如表5、表6所示。

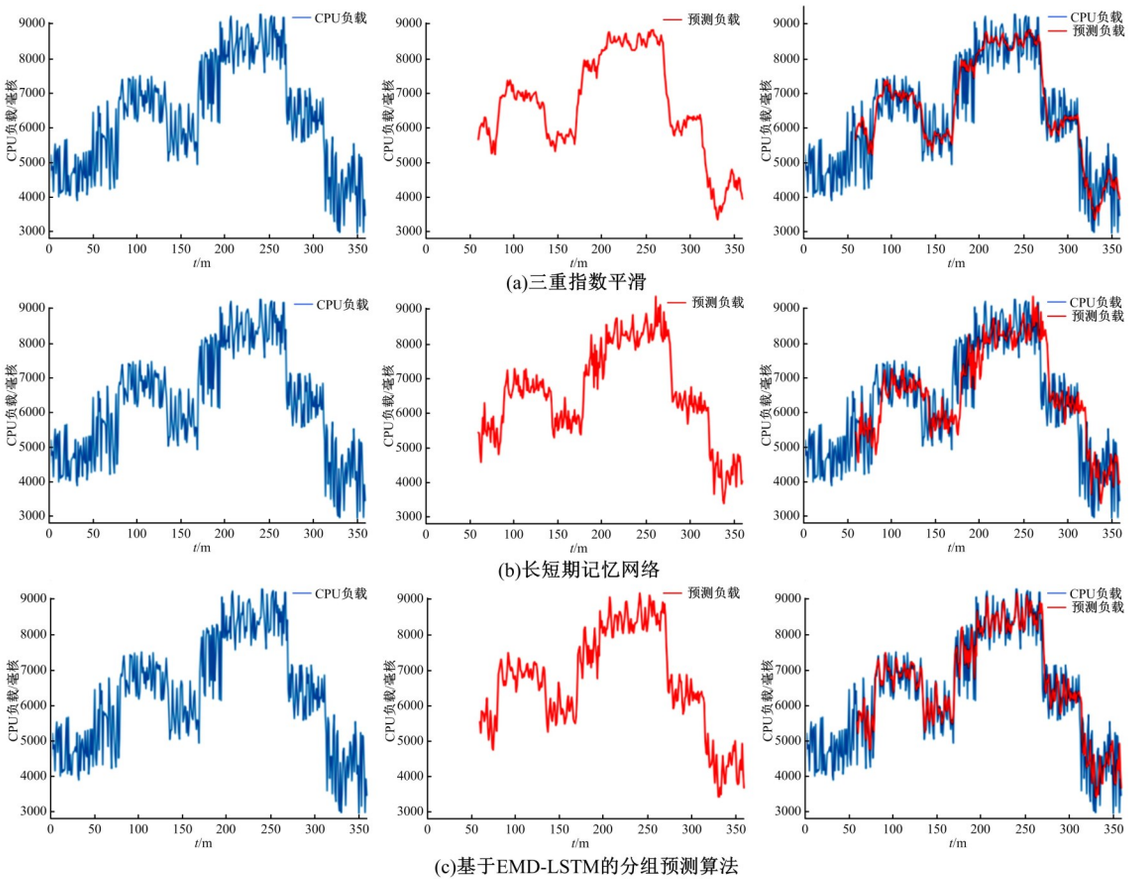

图 6　CPU 负载数据预测对比

Fig. 6　Comparison of predicted CPU load data

表 5　CPU 负载预测信息与实际负载相关系数

Table 5　Correlation coefficients between CPU load predicted information and actual load

编号	方法	相关系数
1	二重指数平滑	0.819 87
2	LSTM	0.838 94
3	基于EMD-LSTM的分组	0.850 09

表 6　内存负载预测信息与实际负载相关系数

Table 6　Correlation coefficients between memory load predicted information and actual load

编号	方法	相关系数
1	二重指数平滑	0.604 30
2	LSTM	0.654 67
3	基于EMD-LSTM的分组	0.721 10

通过测试分析可以看出相较于指数平滑、单一LSTM,在负载序列波动较大的情况下基于EMD-LSTM的分组预测算法预测效果更优。

4.3　纵向自动伸缩测试

4.3.1　测试方法

在安装VPA组件的计算节点上运行3.2节构建的天文数据处理测试pipeline,对比Recommender默认推荐值与本文的基于预测信息的自适应推荐值的CPU、内存资源的自动伸缩情况。为保证天文数据处理pipeline的正常运行,实验中均将Kubernetes VPA的updatePeriodSeconds参数设置为300 s。基于EMD-LSTM分组预测算法所需的60 min训练数据集使用VPA默认推荐值。

4.3.2　伸缩情况对比

使用本文预测算法的CPU资源伸缩情况如图8所示,内存资源伸缩情况如图9所示。蓝色的线显示了在不使用VPA进行资源调整时CPU与内存资源的使用情况,红色虚线显示了VPA默认策略的资源分配情况,黄线显示了本文提出的利用推荐值生成算法时资源分配情况。

相较于VPA默认的伸缩方案,基于预测算法的资源分配更贴合负载的波动情况。在实时观测数据处理的情况下,对于相同数据量的甲醛分子谱线数据,本文的预测算法可以减少3.34%的CPU使用和12.43%的内存使用。天文数据处理对实时性要求较低,可以容忍轻微的任务排队,自

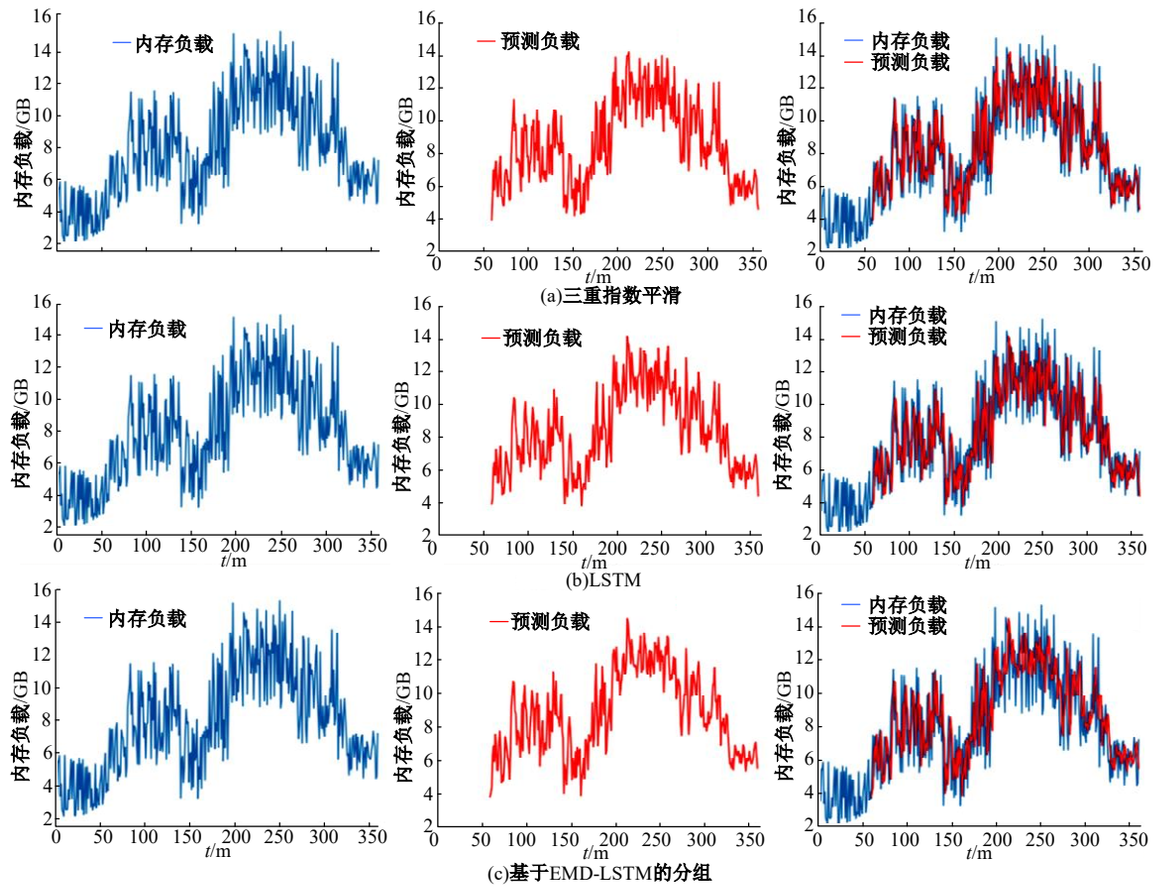

图7 内存负载数据预测对比

Fig. 7 Comparison of predicted memory load data

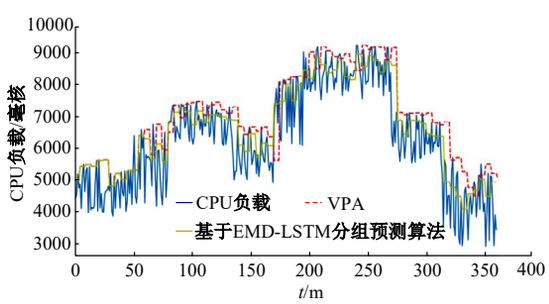

图8 CPU资源伸缩情况对比

Fig. 8 Comparison of CPU resource scaling

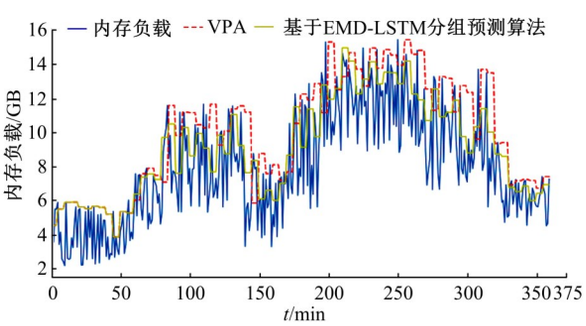

图9 内存资源伸缩情况对比

Fig. 9 Comparison of memory resource scaling

适应的推荐值生成算法可以减少Kubernetes VPA重启Pods的次数,减少数据处理的中断,降低自动伸缩造成的资源消耗。

5 结束语

本文针对Kubernetes VPA资源调整中存在的调度信息滞后问题,根据天文数据处理的内在特性,进行了基于分组的负载预测研究。通过使用K-means对负载信息进行聚类,设计并实现了负载信息与Fits元数据关系的分组算法,对分组后的负载信息使用EMD-LSTM算法进行预测。基于预测信息,设计了自适应VPA推荐值生成算法。在模拟负载数据预测测试中,相较于三重指数平滑法和单一LSTM方法,本文提出的分组预测算法展现出更高的预测准确度。对于天文观测数据实时预处理的复杂负载信息,基于EMD-LSTM的分组预测算法的优势更加明显。与VPA默认设置相比,本文设计的算法能够降低3.34%CPU和12.43%内存资源的使用率。准确

的负载预测在一定程度上减少了重启Pods的次数,降低了自动伸缩本身带来的资源消耗。本文提出的基于预测算法的资源分配策略可有效提高天文数据处理中的资源使用效率。

参考文献:

[1] 陈红松, 陈京九. 基于统计的物联网分布式拒绝服务攻击检测[J]. 吉林大学学报: 工学版, 2020, 50(5): 1894-1904.
Chen Hong-song, Chen Jing-jiu. Statistical based distributed denial of service attack detection research in internet of things[J]. Journal of Jilin University (Engineering and Technology Edition), 2020, 50(5): 1894-1904.

[2] Morris D, Voutsinas S, Hambly N C, et al. Use of Docker for deployment and testing of astronomy software[J]. Astronomy and Computing, 2017, 20: 105-119.

[3] Molenaar G, Makhathini S, Girard J N, et al. Kliko—the scientific compute container format[J]. Astronomy and Computing, 2018, 25: 1-9.

[4] Herwig F, Andrassy R, Annau N, et al. Cyberhubs: virtual research environments for astronomy[J]. The Astrophysical Journal Supplement Series, 2018, 236(1): 2.

[5] Truyen E, van Landuyt D, Preuveneers D, et al. A comprehensive feature comparison study of open-source container orchestration frameworks[J]. Applied Sciences, 2019, 9(5): 931.

[6] Niu J R, Zhu W W, Zhang B, et al. FAST observations of an extremely active episode of FRB 20201124A. Ⅳ. Spin Period Search[J]. Research in Astronomy and Astrophysics, 2022, 22(12): No. 124004.

[7] Wang Y B, Wen Z G, Yuen R, et al. The multiple images of the plasma lensing FRB[J]. Research in Astronomy and Astrophysics, 2022, 22(6): No. 065017.

[8] Chen Z H, You S P, Yu X H, et al. An RFI mitigation pipeline for CRAFTS multi-beam data based on signal cross-correlation function and sum threshold algorithm[J]. Research in Astronomy and Astrophysics, 2023, 23(5): No. 055014.

[9] Dai V D, Kim Y H. Predictive approach for vertical autoscaling in Kubernetes[J]. Proceedings of the Korean Society of Telecommunications, Korean , 2021: 896-897.

[10] Xie Y, Jin M, Zou Z, et al. Real-time prediction of docker container resource load based on a hybrid model of ARIMA and triple exponential smoothing [J]. IEEE Transactions on Cloud Computing, 2020, 10(2): 1386-1401.

[11] Rzadca K, Findeisen P, Swiderski J, et al. Autopilot: workload autoscaling at google[C]//Proceedings of the Fifteenth European Conference on Computer Systems, Heraklion, Greece, 2020: 1-16.

[12] Shanmugam A S. Docker container reactive scalability and prediction of CPU utilization based on proactive modelling[D]. Dublin: National College of Ireland, 2017.

[13] Bandara K, Bergmeir C, Smyl S. Forecasting across time series databases using recurrent neural networks on groups of similar series: a clustering approach[J]. Expert Systems with Applications, 2020, 140: No. 112896.

[14] 冯金巧, 杨兆升, 张林, 等. 一种自适应指数平滑动态预测模型[J]. 吉林大学学报: 工学版, 2007, 37(6): 1284-1287.
Feng Jin-qiao, Yang Zhao-sheng, Zhao Lin, et al. Adaptive exponential smoothing model for dynamic prediction[J]. Journal of Jilin University(Engineering and Technology Edition), 2007, 37(6): 1284-1287.