doi:10.11887/j.cn.202205008 http://journal.nudt.edu.cn

面向混部云失败批处理作业的预测算法*

林伟伟[1,2],石 方[1],李毓睿[1],刘发贵[1],刘 捷[1],彭绍亮[3],王子骏[4]
(1. 华南理工大学 计算机科学与工程学院,广东 广州 510006;2. 鹏程实验室,广东 深圳 518066;
3. 湖南大学 信息科学与工程学院,湖南 长沙 410082;4. 克莱姆森大学 计算学院,美国 克莱姆森 29634)

摘 要:为了降低混部云失败批处理作业的风险,使用*K*-means聚类算法将批处理作业分为四类,在分类的基础上提出了二层嵌套分类模型(two-layer nested classification model, TLNM),实现了基于TLNM的预测算法。基于Ali Trace 2018数据集上的实验结果表明,该算法的接受者操作特性(receiver operating characteristic, ROC)曲线明显优于其他常用分类器,ROC曲线下面积(即AUC)可以达到0.978,表明该算法具有良好的分类性能。同时召回率可以达到0.951,通过混淆矩阵可以看出TLNM算法能够准确预测出执行失败的批处理作业。

关键词:云计算;混部技术;作业失败预测;资源利用率
中图分类号:TP181 **文献标志码**:A **文章编号**:1001-2486(2022)05-071-09

Prediction algorithm for failed batch jobs in co-located cloud

LIN Weiwei[1,2], *SHI Fang*[1], *LI Yurui*[1], *LIU Fagui*[1], *LIU Jie*[1], *PENG Shaoliang*[3], *WANG James Z.*[4]
(1. School of Computer Science & Engineering, South China University of Technology, Guangzhou 510006, China;
2. Peng Cheng Laboratory, Shenzhen 518066, China;
3. College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China;
4. School of Computing, Clemson University, Clemson 29634, USA)

Abstract: In order to reduce the risk of failed batch jobs in co-located cloud, the *K*-means algorithm was used to divide batch jobs into four categories. On the basis of classification, the TLNM (two-layer nested classification model) was proposed and the prediction algorithm based on TLNM was implemented. Experiment results based on Ali Trace 2018 data set show that the ROC(receiver operating characteristic) curve of this algorithm is significantly better than other commonly used classifiers, and the area under the ROC curve (i.e. AUC) can reach 0.978, indicating that this algorithm has good classification performance. At the same time, the recall rate can reach 0.951. Through the confusion matrix, it can be seen that the TLNM algorithm can accurately predict the failed batch jobs.

Keywords: cloud computing; co-location; failed job prediction; resource utilization

2015年,Google发表论文介绍了Borg系统,其中就提到了在线服务作业与批处理作业的混合运行,也就是混部技术[1]。阿里巴巴作为全球最大的几家互联网公司之一,使用混部技术把集群混合起来,将不同类型的任务调度到相同的物理资源上,充分利用资源能力,极大降低成本。为了让业界深入了解混部集群的特征,阿里巴巴在2018年发布了混部集群的负载数据集cluster-trace-v2018(Ali Trace 2018)[2]。

混部集群最主要的问题是:在同一硬件上分配的工作负载越多,每个工作负载受到彼此性能干扰的可能性越大。阿里巴巴的混部生产集群作为一个多租户和多用途数据中心,其中的批处理作业具有高度分散的特点,由Fuxi调度器统一调度管理[3],批处理作业执行不成功的情况时有发生,这些执行失败的批作业将在集群中重复执行,被标记为重调度作业。这些失败作业消耗了不可忽视的资源,可能造成严重的性能障碍。因此,为了提高混部集群的可靠性和容错性,预测失败作业的发生是十分有必要的。为此,本文提出了一种用于预测失败批处理作业的算法。本文的主要贡献包括:

* **收稿日期**:2020-11-23
基金项目:国家自然科学基金面上基金资助项目(62072187, 61872084);广东省重点领域研发计划资助项目(2021B0101420002);鹏程实验室重大任务资助项目(PCL2021A09);广东省基础与应用基础研究基金资助项目(2019B030302002);广州市开发区国际合作资助项目(2020GH10,2020GH10)
作者简介:林伟伟(1980—),男,江西武宁人,教授,博士,博士生导师,E-mail:linww@scut.edu.cn

1)对失败作业的重调度现象进行了深入分析,发现失败作业不仅浪费了大量资源,而且会影响整个集群的工作状态,预测失败作业的发生十分有必要。

2)使用 *K*-means 聚类算法对批处理作业进行了聚类分析,发现不同资源负载类别的批处理作业发生失败现象的风险不同。

3)提出了一个针对混部集群失败作业预测算法,该算法包括特征工程以及基于 LightGBM[4] 的二层嵌套分类模型(two-layer nested classification model, TLNM)。首先通过分类预测 LightGBM 模型(classification prediction lightGBM-model, CPLM)将批处理作业分为四类,然后使用三种不同的异常预测 LightGBM 模型(anomaly prediction lightGBM-model, APLM)分别对三类批处理作业进行二分类预测,即预测该批处理作业是否会发生失败。实验表明,该模型能够准确地预测出失败批处理作业,且在时间性能方面优于其他模型。

1 相关工作

混部集群在提升资源利用率的同时,也带来了许多问题[5-8],例如,批处理作业长期受到在线服务容器的资源挤压,导致两种不同类型的任务在资源分配上的不平衡问题。Liu 等[9]研究了半集装箱云的弹性和塑性。Chen 等[10]在 2018 年对 Ali Trace 2017 进行了进一步的分析,采用 *K*-means 聚类算法对运行在阿里云集群中的负载进行了聚类,并研究了得出类别的属性的共同特征,同时给出了哪几类簇通常会分配到一台物理机执行。Guo[11]等的研究发现:在阿里巴巴的混部集群中,内存似乎成了新的瓶颈,限制了资源使用效率的进一步提升。

在云数据中心的失败任务预测方面,Rosà[12-14]等分析了 Google 大数据集群中失败任务带来的影响,并且提出了一个神经网络分类器来预测这些失败任务。此外,他们还开发了三种在线预测模型,这些模型可以将作业和事件在开始时分为四类。文献[15]则着重于早期阶段云任务的故障预测,提出了一种基于相似作业之间的关联关系的故障预测方法,可以在较早的阶段联合预测任务的终止状态。文献[16]则使用极限学习机对 Google 集群的云作业进行失败预测,可以根据作业到达的顺序收集实时数据,预测作业状态,并根据这些数据更新模型。Hemmat 等[17]使用随机森林分类器对云中违反服务等级协议的作业进行了预测。Shetty 等[18]分析了谷歌的生产集群中的任务资源利用率并且基于 XGBoost 分类器对集群中的失败作业进行了预测。Chen 等[19]介绍了 Google 集群中失败作业和失败任务的统计特征,并尝试将它们与调度约束、节点操作以及云中用户的属性相关联,此外,他们还探索了早期故障预测和作业异常检测的潜力。

2 研究背景

2.1 阿里云混部集群介绍

图 1 描述了阿里巴巴公司的混部集群体系结构。托管的任务由两个调度程序协调管理:Sigma 用于在线服务,而 Fuxi 用于批处理工作负载。其中在线服务作业运行于容器(Container)当中,而批处理作业则被 Fuxi 调度器拆解为批处理实例(Batch Instances),这些实例直接运行于物理机之中。

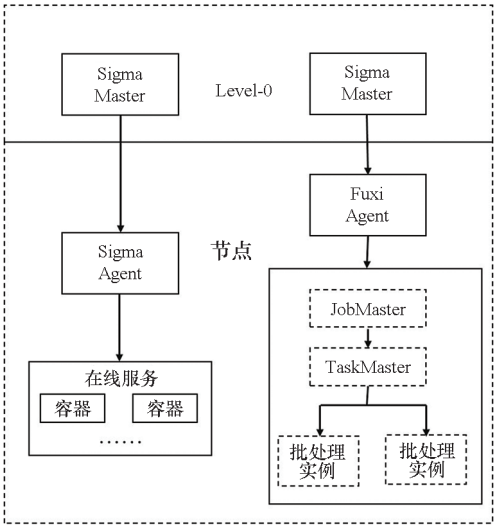


图 1 阿里巴巴混部集群体系结构

Fig. 1 Architecture of Alibaba′s co-located datacenter

2.2 失败实例的重调度现象分析

在混部集群中,由于对共享资源的不可预测的干扰,当出现任何性能峰值时,出于对在线服务容器的资源保护,批处理作业可能会被延迟或驱逐。在阿里巴巴的集群中,当在线服务的性能受到影响时,Fuxi 的代理将首先暂停批处理实例,如果仍然存在干扰,则将这些实例重新安排到另一个主机上。失败的实例将重试多次,直到成功完成为止。虽然批处理作业的失败率只有 0.081%,但失败实例数达到了 100 万个,且 20% 的失败实例出现了多次重调度。此外,61% 的失败实例在运行一段时间后重新调度,38% 的失败实例在未运行的情况下重新调度,3% 的失败实例

需要很长时间才能完成有向无环图(directed acyclic graph, DAG)任务[11]。

表 1 列出了正常实例与失败实例在平均中央处理器(central processing unit, CPU)消耗、平均内存消耗和平均耗时之间的差异。通过对比表格中的数据可以发现,失败实例的 CPU 消耗高出正常实例约 20%,内存消耗高出正常实例约 150%,平均耗时约是正常实例的 13 倍。失败实例的资源浪费情况不容忽视。

表 1　正常实例与失败实例对比

Tab. 1　Normal instances versus failed instances

评价标准	正常实例	失败实例
平均 CPU 消耗(标准化)	62.902 830	76.451 136
平均耗时/s	49.87	679
平均内存消耗(标准化)	0.109 017	0.270 008

实例被重调度所浪费的 CPU 和内存时间可以假设如下:

1)设 T 为 instance_A 被重调度浪费的时间。

2)设(Mem, Cpu)为 instance_A 运行成功时平均内存、CPU 消耗。

记 $T_{Mem} = T \times Mem$, $T_{Cpu} = T \times Cpu$,那么向量(T_{Mem}, T_{Cpu})为失败实例所浪费的资源。

以一个特定的实例(数据集中的 ins_63835109)为例,图 2 为 ins_63835109 失败实例的生命周期示意。它首先在服务器 m_1718 上以时间戳 373 421 进行调度,然后在没有完成的情况下运行 609 s。之后,它被重新调度到服务器 m_177,并在时间戳 374 031 处开始运行,执行时间为1 196 s。那么 374 031 − 373 421 = 610 s 即为失败实例浪费的时间。ins_63835109 的(Mem, Cpu) = (0.31,58),此实例由于重新调度而延迟了约10 min,所浪费的资源向量(T_{Mem}, T_{Cpu}) = (189.1,35 380)。

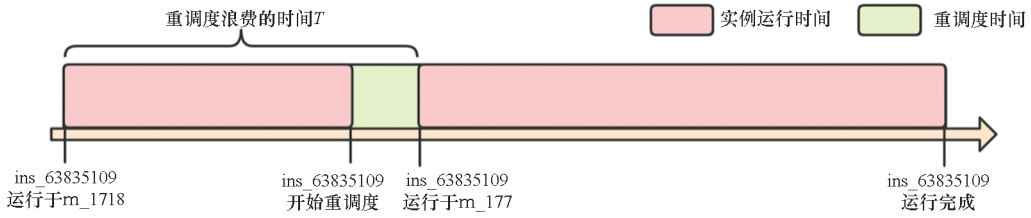


图 2　失败实例的生命周期示意(以 ins_63835109 为例)

Fig. 2　Life of a failed instance (take ins_63835109 as an example)

图 3 和图 4 代表了失败实例所浪费的内存和 CPU 资源的累积分布函数(cumulative distribution function, CDF)图。由图可以观察到,大约超过一半的失败实例的内存资源浪费在 100 以上、CPU 资源浪费在 20 000 以上(内存资源浪费量和 CPU 资源浪费量为标准化数值,每 100 内存表示消耗了一整台服务器的内存总量,每 100CPU 表示消耗了 1 个 CPU 核心),并且大多数失败实例浪费了大量资源。这种资源浪费在超大型数据中心是不容忽视的。虽然重调度这种动态机制保护了在线服务作业的性能,但它牺牲了批量作业的性能,间接地限制了资源效率。

2.3　不同类别批处理作业的失败现象分析

作为数据分析的一部分,本文对 Ali Trace 2018 中的批处理作业进行基于 K-means 的聚类与分析。本文发现了四种不同类型的批处理作业资源特征形式,而且不同类别批处理作业运行失败的风险也不同。

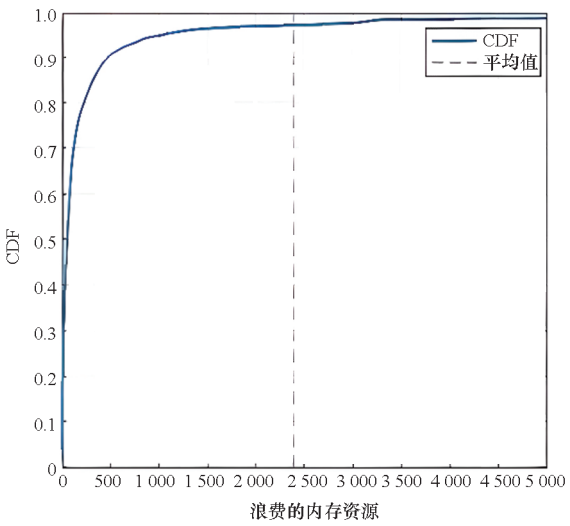


图 3　内存资源浪费 CDF 图

Fig. 3　CDF of the memory wasted

如表 2 所示,将批处理作业的平均耗时、CPU 消耗、内存消耗与实例数量作为批处理作业聚类特征。在 K = 2,…,10 中,取 K = 4 时轮廓系数最

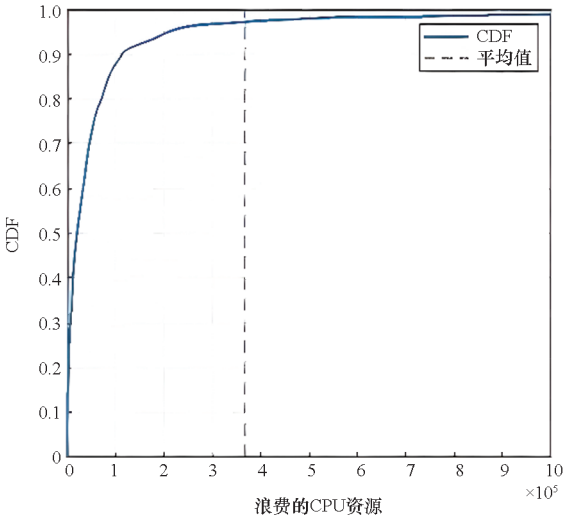


图 4　CPU 资源浪费 CDF 图

Fig. 4　CDF of the CPU wasted

高，为 0.99。因此将批处理任务数据归为 4 类，每类批处理作业的资源负载特征如表 2 所示。

表 2　不同类别批处理作业资源消耗特征

Tab. 2　Resource consumption characteristics of different class batch jobs

K-means 聚类类别	实例数量	平均 CPU 消耗	平均内存消耗	平均耗时/s
类 - 0	4 035 946	57.21	0.088	95.09
类 - 1	28 562	98.69	0.132	3 412.21
类 - 2	4 960	79.06	14.820	22.14
类 - 3	3 385	60.29	0.064	1 257.31

表 3 中列出了批处理作业各个资源画像类别中的失败作业数量与占比（失败作业表示该作业

表 3　不同资源画像批处理作业的失败现象

Tab. 3　Failure phenomenon of different resource portrait batch jobs

作业画像	作业数量	失败作业数量	失败作业占比/%
类 - 0 批处理作业占比最多的中量级作业	4 035 946	20 179	0.5
类 - 1 批处理作业中的重量级作业	28 562	942	3.3
类 - 2 短时间运行的轻量级批处理作业	4 960	0	0
类 - 3 长时间运行的轻量级批处理作业	3 385	1 590	47
总计	4 072 853	22 711	0.56

中有实例发生了重调度）。在 4 类批处理作业画像中，短时间运行的轻量级作业（类 - 2）是不会发生失败的；类 - 0 由于簇内作业数量多，发生的失败作业数量也多，但是失败作业占比不到 1%；类 - 1 属于重量型作业，运行时间久、资源消耗多、实例数量多，因此约有 3.3% 的作业存在失败的情况；类 - 3 属于长时间运行的轻量级作业，约有 47% 的作业存在失败的情况。

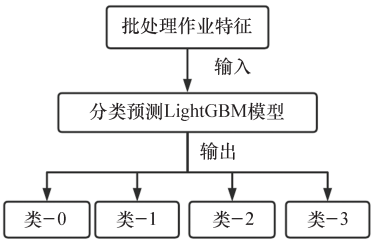
3　TLNM 与基于 TLNM 的预测算法

为了提高混部云的可靠性与错误容忍性，本文提出了一个基于 LightGBM 的 TLNM，并将 TLNM 应用在混部云的失败批处理作业预测中，实现了一个基于 TLNM 的混部云的失败批处理作业预测算法。

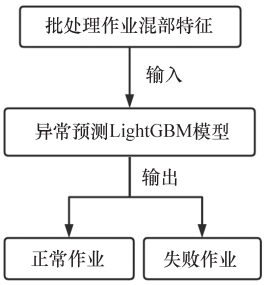
3.1　TLNM 核心思想

机器学习技术被广泛应用于根据数据特征对异常点进行分类。一方面，线性分类器如支持向量机（support vector machine，SVM）或逻辑回归（logistic regression，LR）适合于表现（近似）线性关系的系统；另一方面，决策树（decision tree，DT）、随机森林（random forest，RF）和 LightGBM 等非线性分类器可以更好地对复杂系统建模。由于系统的高度非线性和不稳定性，本文开发了 TLNM 来提高分类精度。本文选择批处理作业进行失败预测，主要出于以下三个考虑：①批处理实例数据量较大，训练模型耗时过长，且不具有代表性；②通过数据分析发现，失败批处理作业中的某个实例一旦发生重调度现象，将会导致该批处理作业中其他实例也发生重调度；③批处理作业作为混部集群中各个实例的代表，可以由 Fuxi 调度器进行实时数据采集，可以设计实时预测系统。

前文第 2 节讨论了不同类别的批处理作业所表现出的失败现象，其中短时间运行的轻量级作业中并没有出现失败作业，而其他画像中的失败作业比例有很大不同。因此，TLNM 的核心思想是对批处理作业进行多分类后，再进行失败预测。TLNM 中有四个子模型，分别是一个 CPLM 和三个 APLM，如图 5 和图 6 所示。其中 CPLM 和 APLM 独立工作：首先将批处理作业按照画像不同进行分类，这是一个多分类问题；然后按照不同的资源画像类别调用不同的 APLM 进行第二次预测，这是一个二分类问题。为了避免过度拟合偏差，通过一个由 20% 的作业证组成的独立验证集来交叉验证预测的准确性。



(a) CPLM



(b) APLM

图 5　CPLM 和 APLM 组织架构

Fig. 5　Architectures of CPLM and APLM

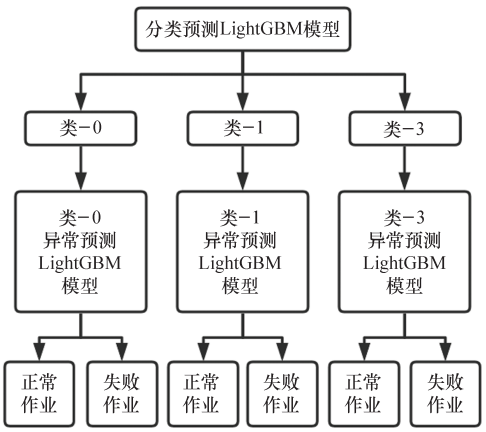


图 6　TLNM 组织架构

Fig. 6　Architecture of TLNM

3.2　基于 TLNM 的预测算法

由于混部云中存在混部干扰,而混部云的数据集无法充分暴露出混部干扰情况,因此需要构造一个完整的特征数据集来配合 TLNM 进行失败任务预测。本小节提出一个针对阿里云失败作业的预测算法,包括针对阿里云的混部特征提取算法(特征工程)和 TLNM 预测。算法 1 是混部特征提取算法的伪代码描述。以 Ali Trace 2018 为例,首先根据 batch_instance 表格中的 machine_id 字段找到该实例所运行的物理机 ID;再根据该 machine_id 匹配 machine_usage 表格和 container_usage 表格,从这两个表格中找到本文需要的特征数据;然后将本文找到的物理机状态数据与容器的运行状态数据结合,形成当前实例的混部特征数据;最后再根据当前实例的混部特征数据中的 job_name 字段,将实例的混部特征数据整理成批处理作业的混部特征数据。该算法的时间复杂度为 $O(t \cdot N)$,t 为每个实例所需要的处理时间。

算法 1　混部特征提取算法

Alg. 1　Extraction algorithm of co-location dataset

输入:原始实例数据集 D_{instance},原始在线容器数据集 $D_{\text{container}}$,原始物理机数据集 D_{machine}

输出:混部特征数据集 $D_{\text{co-location}}$

1. 读取 D_{instance},获取实例数量 N
2. 对每个实例进行如下处理:
   for $i = 1$ to N
   搜索对应物理机数据 $Data_{\text{machine}}$
   搜索对应容器数据 $Data_{\text{container}}$
   返回当前实例的混部特征 $D_i = concat(Data_{\text{machine}}, Data_{\text{container}})$
   end for
3. 所有实例的混部特征 $F_i = concat(D_1, \cdots, D_n)$
4. $D_{\text{co-location}} = F_i.\ GroupBy(job_{\text{ID}})$
5. 返回 $D_{\text{co-location}}$

如图 7 所示,本文首先对混部云数据集进行混部特征提取,制作出混部特征数据集作为 TLNM 的输入数据集,然后对混部特征数据集进

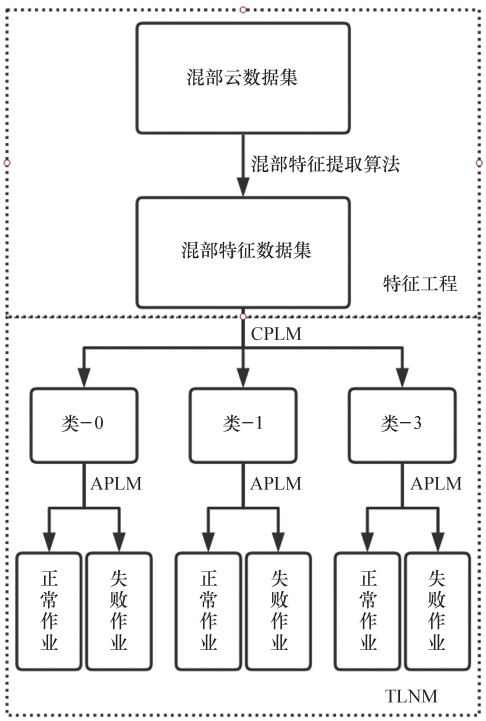


图 7　基于 TLNM 的预测

Fig. 7　TLNM based prediction

行基于TLNM预测算法的分类预测。算法2是基于TLNM的预测算法伪代码描述,TLNM首先使用混部特征数据集中的部分字段(批处理作业的平均耗时、CPU消耗、内存消耗与实例数量)进行批处理作业分类预测,然后再对分类后的批处理作业分别进行异常预测,得到预测结果,即失败作业集。该算法的时间消耗主要在模型的训练上,因此该算法的时间复杂度为 $O(n\log n \cdot m \cdot d)$,其中 n 是训练样本个数,m 是特征个数,d 为梯度提升迭代决策树(LightGBM也是其中一种)的深度。

算法2　TLNM预测算法

Alg. 2　Prediction algorithm of TLNM

输入:混部特征数据集 $D_{\text{co-location}}$,分类预测LightGBM模型 $CPLM$,类-0异常预测LightGBM模型 $APLM_0$,类-1异常预测LightGBM模型 $APLM_1$,类-3异常预测LightGBM模型 $APLM_3$

输出:失败作业集 J

1. 分类预测:
   $CPLM.predict(D_{\text{co-location}}) = D_0, D_1, D_2, D_3$;//$D_0, D_1, D_2, D_3$ 分别为类-0,类-1,类-2,类-3混部特征数据集
2. 异常预测:
   if class = 0 **then**
   $APLM_0.predict(D_0) = J_0$;//预测出类-0中的失败作业集 J_0
   else if class = 1 **then**
   $APLM_1.predict(D_1) = J_1$;// 预测出类-1中的失败作业集 J_1
   else if class = 3 then
   $APLM_3.predict(D_3) = J_3$;// 预测出类-3中的失败作业集 J_3
   end if
3. $J = concat(J_0, J_1, J_3)$
4. 返回 J

4　实验评估

4.1　主要评估指标

接受者操作特性(receiver operating characteristic, ROC)分析是一种直观评估一个或多个分类器准确性的方法,并且不受不平衡数据集的影响。如表4的混淆矩阵所示,如果进行预测,会出现四种情况,即:实例是正类并且也被预测成正类,即为真正类TP(true positive);实例是负类而被预测成正类,称之为假正类FP(false positive);实例是负类且被预测成负类,称之为真负类TN(true negative);实例是正类而被预测成负类,称之为假负类FN(false negative)。

表4　混淆矩阵

Tab. 4　Confusion matrix

实际值	预测值	
	0	1
0	TN	FN
1	FP	TP

1) TPR = TP/(TP + FN)表示当前分到正样本中真实的正样本占所有实际正样本的比例,即召回率;

2) FPR = FP/(FP + TN)表示当前被错误分到正样本类别中真实的实际负样本占所有负样本总数的比例;

3)精确率 = TP/(TP + FN),它的含义是在所有被预测为正的样本中实际为正的样本的概率。

对于ROC来说,横坐标就是FPR,纵坐标就是TPR,当TPR越大,而FPR越小时,说明分类结果是较好的。在样本不均衡的情况下,ROC曲线仍然能较好地评价分类器的性能,这是ROC的一个优良特性,也是一般ROC曲线使用更多的原因。除了ROC曲线的图形化评价外,还可以通过测量曲线下的面积(area under curve, AUC)来量化分类器的准确性。从直观上看,AUC越接近1,分类器就越准确,一个完美分类器的AUC等于1。

在本节的实验评估中,本文主要参考ROC曲线、AUC和召回率来证明TLNM在对比的分类器中具有最好的性能。

4.2　混部特征提取算法实验

为了说明特征工程的重要性,本文评估了TLNM算法在不同的特征数据集上的表现。本次实验使用的两个数据集分别为表2所示的批处理作业四个特征(平均CPU消耗、平均内存消耗,平均耗时和实例数量)构成的数据集与混部特征数据集。表5所示为经过混部特征提取算法的数据集与普通批处理作业数据集实验效果对比结果。通过混部特征提取算法之后,将预测的AUC值从0.62提升到0.98,召回率从0.37提高到0.95。

表 5　TLNM 算法在不同数据集上的表现

Tab. 5　Performance of TLNM algorithm on different datasets

评价指标	批处理作业特征数据集	批处理作业混部特征数据集
召回率	0.37	0.95
AUC	0.62	0.98

除了对比 TLNM 算法在不同数据集上的表现之外,本文还使用 SHAP[20] 工具来对特征重要度进行了进一步解释。SHAP 是基于 Python 开发的一个“模型解释”包,可以解释任何机器学习算法的输出。其名称来源于 SHapley Additive exPlanation,作者在合作博弈论的启发下构建了一个加性的解释模型,所有的特征都视为“贡献者”。对于每个预测样本,模型都产生一个预测值,SHAP value 就是该样本中每个特征所分配到的数值。SHAP value 最大的优势是能反映出样本中每一个特征值对预测结果的影响力,而且还表现出影响的正负性。

图 8 所示为混部特征数据集中各个特征的 SHAP value。其中对模型预测结果影响最大的特征为批处理作业的运行时间(run_time),影响最低的两个特征分别为容器数量(container_num)和物理机内存利用率百分比(machine_mem_util_percent)。出于篇幅考虑,在此并未对每个特征的含义加以说明。图 9 绘制了各个特征对算法输出的影响,由点构成,每个点有三个含义:①点的竖直坐标是说明它属于哪个特征;②点的颜色代表了这个特征的数值是高还是低;③水平位置代表了这个特征在某一行数据里是提高预测值还是降低预测值。比如 run_time 行中最右侧的红点,说明了

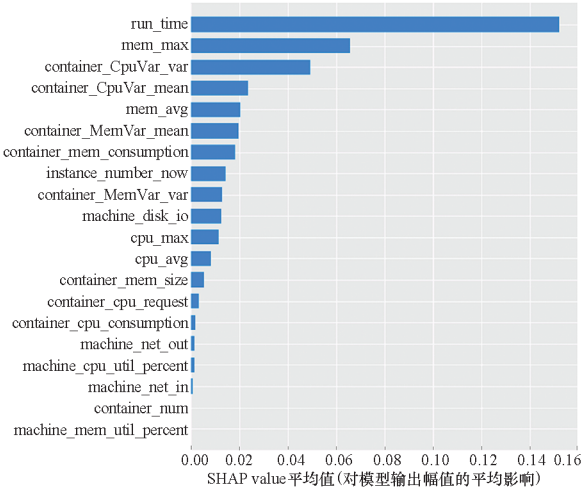


图 8　模型中各个特征的 SHAP value

Fig. 8　SHAP values for each feature in model

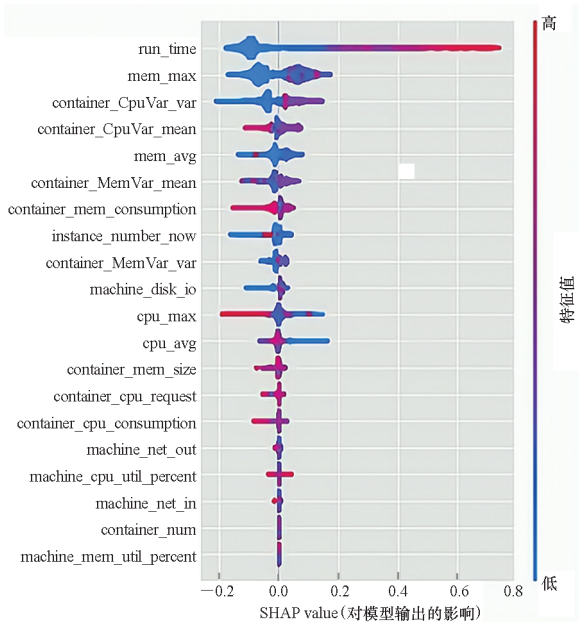


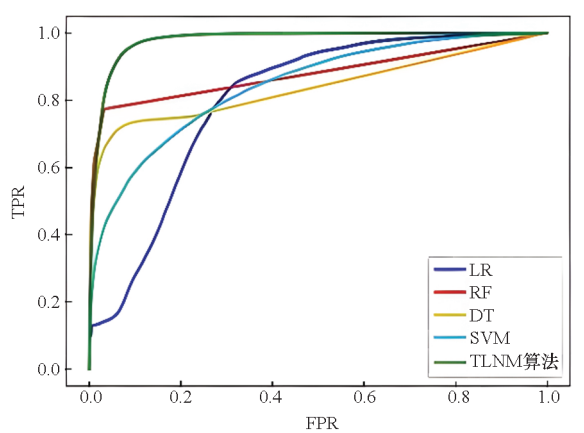
图 9　每个特征对模型输出的影响

Fig. 9　Impact of each feature on the model output

运行时间长的某个特征点,预测效果的 SHAP value 为 0.75 左右,提高了预测值。通过 SHAP 分析可以发现,对算法影响最大的是 run_time 这个特征,值的大小与预测的结果呈正相关性。

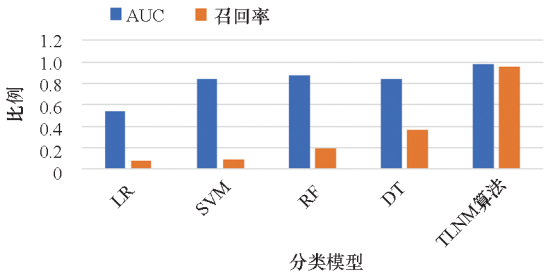
4.3　TLNM 算法与其他分类器的对比

在本小节中,通过比较 TLNM 算法和其他常用分类器(RF、SVM、LR、DT)在混部特征数据集上的表现,以说明 TLNM 算法的优势所在。从图 10可以清楚地看出,TLNM 算法的 ROC 曲线明显高于其他分类器,且 AUC 最高,不仅 AUC 接近 1(取值 0.978),而且与其他分类器之间的 AUC 也存在明显的差距。除了 TLNM 算法之外,第二好的分类器是 RF,AUC 指标为 0.878,其次是 DT。DT 与 RF 的 AUC 相近,两者具有可比性。



(a) TLNM 算法与其他分类器 ROC 曲线

(a) TLNM algorithm versus different classifiers on ROC curves



(b) TLNM 算法与其他分类器的评分对比

(b) TLNM algorithm versus different classifiers on score

图 10 TLNM 算法与不同分类器对比

Fig. 10 TLNM algorithm versus different classifiers

召回率方面,TLNM 的召回率明显高于其他分类器,在高度不平衡的数据集中精确预测出所有可能运行失败的批处理作业。TLNM 算法在测试数据集上预测结果的混淆矩阵如表 6 所示,在 4 153 个失败作业中准确预测出了 3 948 个失败作业,但是错误地把正常的 48 859 个批处理作业也当成了失败作业。

表 6 TLNM 算法的混淆矩阵

Tab. 6 Confusion matrix of TLNM algorithm

实际值	预测值	
	0	1
0	554 764	48 859
1	205	3 948

表 7 列出了 TLNM 算法与不同分类器在训练时的时间性能,所有训练均在同一台服务器(Intel(R) Xeon(R) CPU E5 - 2620 v2 @ 2.10 GHz, 112 GB 内存)上进行。在时间性能方面,TLNM 算法中所有的模型训练时间总计为 72.36 s,优于其他所有分类器。DT 的训练时间为158.19 s,是时间性能表现第二佳的模型。花费最高的模型是 SVM,训练时间达到 1 381.72 s。

表 7 不同分类器的时间性能对比

Tab. 7 TLNM algorithm versus other classifiers on time performance

分类模型	时间性能/s
LR	308.31
RF	271.65
DT	158.19
SVM	1 381.72
TLNM 算法	72.36

5 结论

在一个超大型数据中心中,作业运行失败的现象时有发生,本文对批处理作业的失败现象进行了深入分析,结果发现重调度现象会造成大量资源浪费。此外,为了更准确地预测出混部集群中的失败现象,对批处理作业进行了基于 *K*-means 的聚类分析,将批处理作业分为 4 个类别,发现不同类别的批处理作业发生失败现象的风险不同。在此基础上,提出了一种用于预测混部集群中批处理作业失败现象的二层嵌套分类模型 TLNM,并开发了基于 TLNM 的混部云失败作业预测算法。评估结果表明,与其他分类器(LR、SVM、DT、RF)相比,TLNM 算法的分类结果更优,AUC 值达到最高的 0.978,召回率也达到最高的 0.951。总的来说,本文的研究有助于提高混部集群的资源利用率和错误容忍性,本文的预测算法允许调度系统做出主动决策。例如,基于本文预测算法的资源感知调度可以进行资源优化,通过在预测服务器上标记出可能发生失败现象的批处理作业,可以提前调整批处理作业的优先级或者暂停该作业,以优化超大规模混部集群的资源配置。

参考文献(References)

[1] VERMA A, PEDROSA L, KORUPOLU M, et al. Large-scale cluster management at Google with Borg [C]// Proceedings of the Tenth European Conference on Computer Systems, 2015.

[2] Alibaba Group. Alibaba production cluster data v2018 [DB/OL]. (2018 - 12 - 03) [2020 - 11 - 20]. https://github.com/alibaba/clusterdata.

[3] ZHANG Z, LI C, TAO Y Y, et al. Fuxi: a fault-tolerant resource management and job scheduling system at Internet scale[C]//Proceedings of the 40th International Conference on Very Large Data Bases, 2014.

[4] KE G L, MENG Q, Finley T, et al. LightGBM: a highly efficient gradient boosting decision tree[C]//Proceedings of the 31st Conference on Neural Information Processing Systems, 2017.

[5] LU C Z, YE K J, XU G Y, et al. Imbalance in the cloud: an analysis on Alibaba cluster trace[C]//Proceedings of IEEE International Conference on Big Data (Big Data), 2017.

[6] CHENG Y, CHAI Z, ANWAR A. Characterizing co-located datacenter workloads: an Alibaba case study[EB/OL]. (2018 - 08 - 15) [2020 - 11 - 20]. https://arxiv.org/abs/1808.02919.

[7] CHEN S, DELIMITROU C, MARTÍNEZ J F. PARTIES: QoS-aware resource partitioning for multiple interactive services [C]//Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems, 2019: 107 - 120.

[8] REN R, LI J H, WANG L, et al. Anomaly analysis for co-located datacenter workloads in the Alibaba cluster[J]. arXiv: Distributed, Parallel, and Cluster Computing[EB/OL]. (2018 - 11 - 14)[2020 - 11 - 21]. https://arxiv.org/abs/1811.06901.

[9] LIU Q X, YU Z B. The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from Alibaba trace[C]//Proceedings of the ACM Symposium on Cloud Computing, 2018.

[10] CHEN W Y, YE K J, WANG Y, et al. How does the workload look like in production cloud? analysis and clustering of workloads on Alibaba cluster trace[C]//Proceedings of IEEE 24th International Conference on Parallel and Distributed Systems, 2018.

[11] GUO J, CHANG Z H, WANG S, et al. Who limits the resource efficiency of my datacenter: an analysis of Alibaba datacenter traces[C]//Proceedings of the International Symposium on Quality of Service, 2019.

[12] ROSÀ A, CHEN L Y, BINDER W. Catching failures of failures at big-data clusters: a two-level neural network approach[C]//Proceedings of IEEE 23rd International Symposium on Quality of Service, 2015.

[13] ROSÀ A, CHEN L Y, BINDER W. Understanding the dark side of big data clusters: an analysis beyond failures[C]//Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015.

[14] ROSÀ A, CHEN L Y, BINDER W. Failure analysis and prediction for big-data systems[J]. IEEE Transactions on Services Computing, 2017, 10(6): 984 - 998.

[15] LIU C H, DAI L P, LAI Y, et al. Failure prediction of tasks in the cloud at an earlier stage: a solution based on domain information mining[J]. Computing, 2020, 102(9): 2001 - 2023.

[16] LIU C H, HAN J J, SHANG Y L, et al. Predicting of job failure in compute cloud based on online extreme learning machine: a comparative study[J]. IEEE Access, 2017, 5: 9359 - 9368.

[17] HEMMAT R A, HAFID A. SLA violation prediction in cloud computing: a machine learning perspective[EB/OL]. (2016 - 09 - 30)[2020 - 11 - 21]. https://arxiv.org/abs/1611.10338.

[18] SHETTY J, SAJJAN R, SHOBHA G. Task resource usage analysis and failure prediction in cloud[C]//Proceedings of the 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2019.

[19] CHEN X, LU C D, PATTABIRAMAN K. Failure analysis of jobs in compute clouds: a google cluster case study[C]//Proceedings of IEEE 25th International Symposium on Software Reliability Engineering, 2014.

[20] LUNDBERG S, LEE S I. A unified approach to interpreting model predictions[EB/OL]. (2017 - 11 - 25)[2020 - 11 - 21]. https://arxiv.org/abs/1705.07874.