计算机科学
COMPUTER SCIENCE

面向超级计算系统的节点故障异常预测方法

赵一宁, 王小宁, 牛铁, 赵毅, 肖海力

引用本文

赵一宁, 王小宁, 牛铁, 赵毅, 肖海力. 面向超级计算系统的节点故障异常预测方法[J]. 计算机科学, 2025, 52(9): 128-136.

ZHAO Yining, WANG Xiaoning, NIU Tie, ZHAO Yi, XIAO Haili. Node Failure and Anomaly Prediction Method for Supercomputing Systems [J]. Computer Science, 2025, 52(9): 128-136.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240700171

面向超级计算系统的节点故障异常预测方法

赵一宁　王小宁　牛　铁　赵　毅　肖海力
中国科学院计算机网络信息中心　北京 100190

摘　要　随着超级计算系统的规模不断扩大,其计算节点发生故障和异常的概率也随之上升,严重影响了计算系统的运行稳定性。传统的故障应对方法多采用事后响应和补救措施,只能一定程度地挽回损失,而对故障和异常进行事前预测则能够提供更多的反应和处理时间,因此逐渐成为故障响应方法的研究热点之一。对此,提出了一种面向超级计算系统的节点故障异常预测方法,旨在提升系统运行稳定性,减少计算资源的浪费。该方法首先分析系统历史运行数据,并通过无监督结合少量人工辅助的方法标记异常,基于这些异常在原始运行数据中发现关联的前置运行特征,随后基于机器学习方法建立节点故障异常的预测模型。该预测方法通过在原数据集上交叉验证获得了78%的精度和约90%的召回率,同时也保证了充分的提前时间。验证中使用的数据集是来自真实的超级计算系统的原始运行数据,证明了该方法具有可应用性。

关键词:数据分析;异常预测;运行特征;预测模型
中图分类号　TP311

Node Failure and Anomaly Prediction Method for Supercomputing Systems

ZHAO Yining,WANG Xiaoning,NIU Tie,ZHAO Yi and XIAO Haili
Computer Network Information Center,Chinese Academy of Sciences,Beijing 100190,China

Abstract　As the scale of supercomputing systems continues to expand,the probability of computing node failures and anomalies also increases,seriously affecting the stability of systems. Traditional fault response methods mainly apply post event response and remedial policies,which can only partially recover the wastage. Predicting node failures and anomalies in advance can provide more response and processing time,thus has become a research hotspot. This paper proposes a node failure and anomaly prediction method for improving the stability of supercomputing systems and reducing the waste of computing resources. The method analyzes the historical running data of the system,and marks anomalies through unsupervised methods plus a small amount of manual assistance. These anomalies are used to find correlating pre-running features are discovered in the original dataset. Prediction models are then established using machine learning methods. This prediction method achieves the precision over 78% and the recall around 90% through cross validation over the original dataset,and it also ensures sufficient lead time. The dataset used in the evaluation comes from the raw running data of a real supercomputing system,proving the applicability of the proposed method.

Keywords　Data analysis,Anomaly prediction,Running feature,Prediction model

1　引言

随着超级计算产业进入E级(每秒百亿亿次浮点运算)时代,超级计算机的系统规模持续扩大、计算节点数量不断增加,其发生故障和异常的概率也在同步上升。当节点故障异常发生时,被调度到该节点的计算任务也会受到影响,轻则运行速度减缓、完成时间延后,重则任务中止,需要从头开始重新计算。这些情况严重影响超算系统的稳定性,造成运行效率低下和计算资源浪费。

面对这个问题,以往的解决方案更多倾向于被动响应和事后补救,通常包括两种策略:1)内容恢复,即当节点故障发生后,尝试通过内存、磁盘临时文件等任务残余内容对任务的中间状态进行重现;2)存档续算,即基于检查点技术定期对计算任务进行快照保存或增量记录,并在节点发生故障、任务被迫中止时根据存档内容在正常节点继续运行。这两种方法相比重新开始计算任务,虽然可以一定程度上减少对计算资源的浪费,但仍需考虑内容恢复和定期存档的难度和额外开销。

目前,针对节点故障的主动响应和事前发现的方法也逐渐成为研究热点之一,这类方法尝试通过基于预测的前处理方式提前发现计算节点的故障异常现象,并启动如任务存档、迁移等前处理方案,从而避免因为任务中断造成的机时浪费及额外开销。相比事后和被动响应,前处理的优势在于,其可以实时关注超算系统的运行状态,及时发现系统性能下降、组

到稿日期:2024-07-25　返修日期:2024-10-18
基金项目:中国科学院战略性先导科技专项(XDB0500103)
This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences(XDB0500103).
通信作者:赵一宁(zhaoyn@sccas.cn)

件工作异常等现象，在形成较为重大的失效、故障之前就启动应对措施，从而尽量将损失最小化。

本文提出了一种面向超级计算系统的节点故障异常预测方法，基于无监督学习及少量人工辅助的方式提取超算系统运行数据中的异常信息，并据此进行关联分析，锁定与故障有关的关键因素，随后进一步建立系统运行特征向量来建立故障异常预测模型。本文采集了某大规模超算系统的运行数据作为研究对象，并测试了本预测方法的运行效果。

本文工作的贡献点主要体现在：

1）实现了从数据收集、异常标记、特征提取到模型建立与验证的完整工作流程，使其具有更好的可应用性；

2）使用基于无监督学习结合少量人工辅助的方式实现异常标记和特征关联，在极大减少人工工作量的同时也保证了标记的准确性；

3）本方法研究和测试所采用的数据全部来自超算系统运行期间产生的真实数据，确保了所提方法可以应对实际的生产环境；

4）给出了部分影响节点稳定运行的主要因素的观察结果，对后续超算系统运行维护等操作提供了方向指引和支持。

本文第 2 章具体介绍了节点故障异常预测方法的背景和关键点，第 3 章详细阐述了预测方法的工作思路和步骤流程，第 4 章讲述了模型建立与测试的过程与结果以及部分观察结论，第 5 章介绍了与节点故障相关的一些研究工作。

2 研究背景

超级计算是一种通过高性能计算机系统来实现对大规模计算任务的加速运算的重要方法，它对于众多科研和工业领域的计算任务都有着显著的推动作用，可以大幅度缩短复杂计算任务所耗费的时间，从而提升工作效率，因此超级计算对于国家科技发展、工业和经济建设、民生及健康保障都有着极其重要的意义。

现代超级计算系统的规模持续扩大，计算节点的数量也在快速提升。在实际运行中，计算节点不可避免地会发生异常、错误、故障等不同等级的问题，影响系统的稳定运行和超级计算应用程序的顺利执行。发生故障的节点通常会手动或自动重启，并导致被调度到该节点的计算任务被迫中止、中间的计算进程全部丢失，调度系统不得不再次为该任务分配其他计算节点并重新开始执行计算。计算任务的中断导致了多余的计算时间以及重复消耗相应的计算资源，同时对于提交作业的用户造成负面影响，特别是当计算任务规模较大时，因其被迫中断造成的多余消耗可能造成沉重的成本损耗。面对这个问题，研究人员提出了一些解决方案，包括将运行在潜在故障节点上的作业进行备份、状态存档，或者迁移到其他计算节点上继续运行等前处理方法。为了实现这一点，需要提前预知节点是否存在故障的可能。因此，节点故障异常预测成为超级计算系统运行领域的一个主要的研究方向，可以通过分析超级计算系统的实时运行数据来给出计算节点发生故障的判定，从而使得针对故障异常的主动响应方法成为可能。

为了实现节点故障异常预测功能，需要探索超算系统的常时运行数据，并从中发现与节点故障异常有关联的运行特征，通过对这些运行特征进行监控来发现潜在的异常先兆，推断节点故障发生的可能性。

本文所提出方法的目标是对超算系统内的计算节点的故障异常进行预测，这里的“故障异常”有两个含义：1）故障指计算节点因非人工的原因发生的系统错误、卡死，进而导致节点下线和重启，此时节点上的作业未经保存的状态全部丢失，造成较大的资源浪费；2）异常指计算节点发生慢性的或突发的状态变更，导致节点产生较明显的性能波动使得其无法正常、高效地工作，降低了系统计算资源的使用效率。其中故障也可以算作异常的一种，但其产生的后果更加严重，在方法的设计和测试过程中需要增加故障部分的权重，或者进一步设置独立考量、处理机制和统计方法。

在基于真实的超级计算系统运行数据进行预测方法研究的过程中，我们遇到了两个需要解决的主要问题：1）原始数据并不存在明确的节点故障的标记，即无法得知系统中的哪些节点在哪些时间点发生过故障，这意味着无法直接应用有监督的分类方法；2）原始数据种类繁杂、特征多样，不同计算系统往往采用不同的数据采集方式和表现形式，可能并不能支持现存的一些预测方法所采用的数据特征属性，因此较难应用和借鉴这些现存方法。第 3 章将具体阐述如何解决这些问题。

3 节点故障异常预测方法

本章叙述节点故障异常预测方法的各部分细节。节点故障异常预测方法旨在通过数据分析来探索与节点故障异常有关联的前期征兆，从而在实际运行过程中针对先兆事件发出故障异常的预警。本文方法基于机器学习技术，通过标记节点故障异常信息、发现与其相关的运行特征来实现预测。如图 1 所示，本方法由 4 个主要部分组成，分别为数据准备、异常信息提取与标记、运行特征选取与抽象、运行特征向量及预测模型的建立。异常信息提取和运行特征选取这两个部分虽然呈并行进行，但其中有部分信息和方法采用了互通的处理手段。

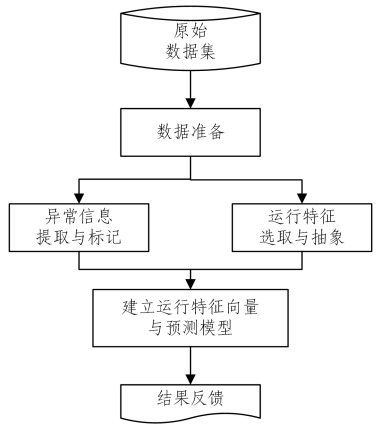

图 1 节点故障异常预测方法工作流程

Fig. 1 Process of node failure and anomaly prediction

3.1 数据准备

为探索可能的节点故障异常先兆事件，我们获取了中国科学院某大规模先进超级计算系统部分节点的实际运行

数据，以期通过数据分析发现潜在的关联性。该系统采用Slurm作为集群管理和任务调度系统。数据内容主要包括以下种类：各节点的Linux操作系统的Syslog系统日志、集群管理和调度系统Slurm的主服务工作日志、集群InfiniBand的网络管理器OpenSM的工作日志、各节点的系统运行监控指标数据等。其中Syslog日志主要包含重要的系统行为、网络访问等信息，Slurm日志主要包含调度主节点与各计算节点的通信情况、节点可用性、超级计算作业的分发与完成情况等信息，OpenSM日志包括网络通信状况、路由变化、节点描述及状态变化等信息，系统监控指标主要指对于系统内的处理器及内存占用情况、磁盘读写和网络通信等各项指标进行定时监测和记录得到的数据。

日志属于文本类数据，其内容人类可读，但并不利于机器来处理。但数量级的数据对于人工操作而言过于庞大，因此机器处理仍然是日志解析的关键环节。为了使日志数据能够通过程序简化处理过程，本文采用了日志分类方法[1]来获取日志模板，并以此为单位进行进一步的统计分析工作，例如提取日志事件中的变量信息、统计各日志事件在某段时间内的发生次数等。分类后的日志模板情况如表1所列。

表1 数据集日志模板提炼结果

Table 1 Log templates in the dataset

日志来源	模板数量	简述
Syslog	812	节点操作系统rsyslog服务产生的日志
Slurm	197	Slurm软件生成的调度日志
OpenSM	30	节点Infiniband网络管理日志

系统监控指标数据属于数值类数据，由目标超算系统各节点的系统监控工具实时扫描系统指标状况，并设定每半分钟进行一次采样记录。监控对象包括处理器、磁盘容量、内存占用、磁盘读写、网络传输和InfiniBand状况六大类共59种指标。数值类数据的内容本身是机器可读的数值，但仍需要做一些转换操作，如从时间和指标等不同维度建立数据的索引和读取方式。

3.2 异常信息提取与标记

为了探索与故障异常相关联的先兆事件，首先必须找到数据中故障异常本身的表现方式。为了解决数据缺乏故障异常标记的问题，首先通过调查前述步骤获得的日志模板集合，发现其中故障往往在日志数据中被记录或有所体现，而异常则大多隐藏在监控数据中。

通过检查所有被提取的日志模板，寻找其中是否包含节点关闭、重启等信息。在Slurm日志中发现了模板“error: Nodes 〈nodeID〉 not responding”(节点无响应)。部分发生该事件的节点在短时间后出现了“Node 〈nodeID〉 now responding”(节点恢复响应)的事件日志，标志着该节点只是临时性的应答或通信失误并且已经恢复正常。然而其余未恢复的节点无响应事件则都代表了一次节点故障，这一点也从系统日志Syslog中得到了佐证：在Slurm日志中发生节点无响应事件且未在随后恢复响应的节点，其Syslog日志在相邻的时刻也大多发生了节点重启事件(包含服务初始化、启动各种基础进程等一系列系统启动操作)，或者从该时刻附近开始长时间没有新日志出现直到重启事件发生(代表该节点进入宕机状态但没有立即重启)。遇到这种情况，可以确定地将其记录为一次节点故障，并记录下发生该故障的时间点以备后续分析应用。

对于监控指标这种数值类数据，为了区分常态和异常的情况，需要找到一种方法将每个节点的每个指标的监控值进行分类。本文采用了一种类DBSCAN的基于密度的聚类算法来实现这一功能，因为DBSCAN算法具有无须预设簇数以及可识别噪声点的优点，比较适合当前场景需求。

为避免人工设定参数造成陷入经验误区的结果，本文设定了一种基于累积分布函数(Cumulative Distribution Function，CDF)和斜率的自动计算参数的方法。累积分布函数$CDF(x)$定位为当自变量取值范围为0到x时满足条件的元素个数，在本文背景中，元素即为一次监控记录。接下来计算给定自变量值x在累积分布函数中的斜率S。由于x是一个连续数值，本文采用采样的方式将其转换为离散数值，例如将x的最大值分为n份(n即为采样样本数)，则有$x_i=\max(x)\times i/n, 0\leqslant i\leqslant n$。自变量值$x$的斜率$S(x)(x>0)$的计算式为：

$$S(x)=(CDF(x)-CDF(x-1))/n\times 100 \tag{1}$$

基于上述计算方法，监控数值的具体聚类流程如下。

1)获取某节点N指定监控指标m在数据集范围内全部时间的监控值集合$V_m=\{v_1,v_2,\cdots,v_n\}$，并统计其中的最大值$v_{\max}$和最小值$v_{\min}$。

2)建立针对指标值v的累积分布函数$CDF(v)$，v的取值范围为$(v_{\min},v_{\max})$，采样样本数$n=100$。计算获得两个值v_b和v_t，满足$S(v_b-1)\leqslant 1, S(v_b)>1, S(v_t-1)>1, S(v_t)\leqslant 1$。

3)设定聚类密度半径$r=(v_t-v_b)/2$。

4)建立空聚类集合$CS=\varnothing$，其每个内部元素均为监控值的子集合$C_j(C_j\subseteq V_m)$。

5)扫描V_m内所有元素v_i，比对CS内每个元素聚类C_j中的最大值$v_{C_j_\max}$和最小值$v_{C_j_\min}$。如果$v_{C_j_\min}-r\leqslant v_i\leqslant v_{C_j_\max}+r$，则将$vi$加入$C_j$中；如果$v_i$无法加入任何$C_j$，则为$v_i$建立新的聚类$C_n$，并将$C_n$加入CS中。

6)完成所有监控值v_i的聚类后，计算CS中每个元素聚类C_j内的监控值数量$|C_j|$和平均值$avg(C_j)$。

在本方法中，v_b和v_t是斜率最接近1的两个点，即累积分布函数曲线的陡峭与平滑部分的分界。通过计算v_b和v_t可以自动获取监控指标数值的最大分布区间，并据此确定类DBSCAN算法的半径。

完成聚类工作后，对于CS中的所有聚类，选择其中监控值数量最多的一类并将其定义为常态类C_{norm}，将其平均值$avg(C_{\text{norm}})$定义为该节点N的监控指标m的常态值v_{norm}。对于其他任意聚类C_j，如果其元素数量$|C_j|>|C_{\text{norm}}|/20$，则定义$C_j$为其他类；而不属于常态类和所有其他类的监控值元素，则被定义为异常类。在本文采用的数据集中，所有节点监控指标的常态类的元素数量均超过该指标所有元素数量的50%，而占比最高的常态类元素数量超过该指标所有元素数量的99%。

尽管通过聚类可以获取节点在某时刻的每个指标是否异常，但节点整体状态异常不能由单一指标异常来代表，仍需要对所有指标进行整体判断。为公平地衡量所有指标的异常状

况，本文引入了“偏移值”这一概念，来代表每个指标监控值相比其常态值的偏移程度。对于节点 N 的监控指标 m，其常态值为 v_{norm}，监控值集合 $V_m=\{v_1,v_2,\cdots,v_n\}$。首先找到与 v_{norm} 的最大差值 $d_{\max}=\max(\text{abs}(v_i-v_{\text{norm}}))$，$i\in\{1,2,\cdots,n\}$，则对于任意时间点 x，其监控值 $v_x\in V_m$，其偏移值 $b_m(v_x)=\text{abs}(v_x-v_{\text{norm}})/d_{\max}$。

当完成监控数值的偏移值计算后，就可以采用关键监控指标（见 3.3 节）的平均偏移值来代表系统运行在监控层面的状态和稳定性，即节点 N 在时间点 x 的平均偏移值 ab_N 为：

$$ab_N=\left(\sum_{i=1}^{n}b_{mi}(v_x)\right)/n \tag{2}$$

其中，n 为关键监控指标数量。

本文设置了一个阈值系数 t_a，并计算过往一段时间（如 6 小时）的平均偏移值 pab_N，如果有

$$ab_N>pab_N\times t_a \tag{3}$$

则将该时间点标记为异常。通过采用前述的自动计算参数的方法为 t_a 赋值，将 t_a 作为累积分布函数的自变量，采样间隔为 0.01。假设监控值总数量为 m，则当 t_a 同时满足

1）$CDF(t_a)/m>0.8$

2）$S(t_a)\leqslant 1$

3）$S(t_a-0.01)>1$

时，将其设为异常标记的阈值。

3.3 运行特征选取与抽象

超算运行系统的原始运行数据种类繁杂、数量庞大，如何选取其中的一部分与故障异常存在关联关系的信息作为运行特征的代表因素，是本类预测方法的关键。本文尝试从已经获得的超算系统的各类运行数据中提取相关运行特征，其中：Syslog 系统日志包含了节点的重启信息，证明确实有节点错误发生，但在故障发生前未找到异常信息；OpenSM 网络日志与节点故障也没有找到直接关联；Slurm 调度日志中可以找到节点错误事件，但同样缺乏错误的先兆信息；节点监控指标数据则包含了许多异常信息，其中有部分信息与节点错误呈现明显的关联性，需要进一步分析。

将每个监控指标绘制成时间-数值的折线图，并将节点故障的时间点也标记在图中。经观察发现，有部分监控指标在节点故障发生前一段时间内曾经出现了明显的波动，这种监控指标有较大可能与节点故障存在关联关系，因此可以作为运行特征的选取对象。图 2 展示了一个与节点故障具有明显关联性的监控指标的例子，图中箭头代表了在该时间段内的两次节点故障记录，可以看到在这两次故障记录的短时间之前，该指标（磁盘平均等待时间）都有大幅提升。

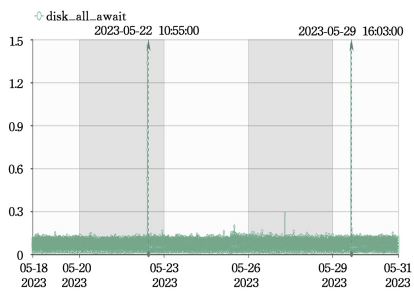

图 2 与故障高度关联的监控指标示例

Fig. 2 Illustration of a metric highly-raleted to node falures

为了确定哪些监控指标可以被用于实现节点故障的关联预测，首先设定了关联跨度窗口的时长，即发生在故障前多久的指标异常可以被认为与故障有关联。该关联跨度窗口的时长计算方法为故障前指标异常间隔的平均值与标准差之和，假设故障集合 $F=\{f_1,f_2,\cdots,f_n\}$，每个故障与前一次监控值异常的时间间隔集合 $GAP=\{g_{f1},g_{f2},\cdots,g_{fn}\}$，则关联窗口的时长 $\Delta t=avg(GAP)+StDev(GAP)$。其次，统计每个指标在所有节点故障前的关联窗口内发生过异常的次数，即假设故障集合 F，节点故障 $f_i\in F$，其发生时间为 t_i，关联窗口时长为 Δt，则当指标 m 在该故障所在节点的（$t_i-\Delta t$）到 t_i 期间内发生过异常时，该计数值 C_m 增加 1。统计故障集合中所有故障 f_1 到 f_n 的指标 m 异常状况，最终得到指标 m 的故障先兆率 $P_m=C_m/n$。当指标 m 的故障先兆率 P_m 大于阈值 t_h 时，就可以将 m 列入关键监控指标中。如表 2 所列，在实验中通过该筛选方法，从全部 59 个监控指标中选出了 27 个关键监控指标（$t_h=0.1$），这些关键监控指标将用于建立运行特征向量。

表 2 与节点故障异常相关的关键监控指标

Table 2 Key metrics related to node failures

指标大类	具体指标
处理器	idle，iowait，sys，user
磁盘	disk_all_avgqu，disk_all_await，disk_all_read，disk_all_write，disk_all_svctm，disk_all_tps，disk_all_util
InfiniBand	multicast_xmit_packets，port_rcv_data，port_rcv_packets，port_xmit_data，port_xmit_discards，port_xmit_packets，unicast_rcv_packets，unicast_xmit_packets
内存	mem_free，mem_ratio，mem_used
网络	net_all_recv，net_all_send，rcv_bytes，snd_bytes，snd_pkts

在监控指标以外，本文还引入了节点作业负载作为运行特征的一部分因素。节点作业负载定义为，在某时刻一个节点上同时运行的作业数量。作业信息可以从 Slurm 调度日志中获取，首先提取了所有与作业调度相关的日志模板，获取其“作业号”变量位置，然后扫描原始数据集，以作业号为单位统计所有作业的生命周期：记录所有包含该作业号的日志，并记录其起始时间点（首次出现的日志时间）和终结时间点（最后一次出现的日志时间），同时针对其中的节点分配情况做出记录。当建立运行特征向量时，可以根据目标节点所有作业的生命周期是否包含所查询的时间点来统计该时间点的作业负载。

3.4 运行特征向量及预测模型的建立

为了实现使用机器学习方法建立故障异常预测模型，需要将运行特征数据转换成为向量的表现形式，每个向量代表一次监控记录时间点的节点状态。首先，要建立运行特征向量的纲要，以统一的格式设置向量每个位置数值的含义。根据前述运行特征选取结果，向量纲要将基于节点关键监控指标和节点作业负载建立。对于每个关键监控指标，除了当前监控值以外，还需要对其过往一段时间的状态进行表述，因为瞬时异常是计算机系统中较常见的一种情况，可能很快就完成自我修复并恢复正常工作状态，但如果监控值异常呈现持续状态，则其影响节点运行的可能性会明显提升。因此，在纲要中对每个关键监控指标加入当前监控值的偏移值、过往 120 个时间点（即 1 个小时）中该指标的异常时间点比例（即

异常次数/120)、这些异常时间点的平均偏移值、当前异常持续时间(如果当前值非异常则清零)这 4 个特征。此外,纲要中还加入了监控值的整体异常情况,包括当前 27 个关键监控指标中的异常指标比例和这些异常指标的平均偏移值这两个特征。节点作业负载方面,则加入了当前节点作业负载和过往 120 个时间点中的平均负载这两个特征。以 120 个时间点作为统计窗口容量是因为根据统计所有指标在所有节点故障前的最近一次被归为异常值的时刻,其与故障发生时间约 90%都在 1 小时以内,因此近似取整至 120 个时间点以便理解和操作。整体的节点运行特征向量纲要如图 3 所示。

```
{
  时间,
  节点 ID,
  [
  指标 i 当前偏移值,
  指标 i 过往异常比,
  指标 i 过往偏移均值,
  指标 i 异常持续时间
  ],( i=1~27 )
  异常指标比例,
  异常偏移均值,
  当前作业负载,
  过往平均作业负载
}
```

图 3 节点运行特征向量纲要

Fig. 3 Outline of node runtime feature vectors

随后,以运行特征向量纲要为准绳,将所有相关的运行数据转换成运行特征向量,并与异常标记结果按时间相对应,形成用于建模和测试的向量数据集。转换运行特征向量的基本工作流程如下。

1)对给定节点 N 的数据,建立容量为 120×27 的过往偏移值记录表,建立容量为 120 的过往节点作业负载记录表。

2)按序扫描每个监控数据生成时间点的关键指标监控值,计算纲要中所有需要的运行特征值。其中过往数据由步骤 1)中记录表内数据统计得出。

3)加入时间点、节点信息,输入该行向量。

4)对于每个关键监控指标,若当前偏移值为异常值,则将其加入过往偏移值记录表对应行的尾部,如果当前偏移值不属于异常值,则将 0 加入记录表尾部。若记录表数量超过容量,则移除最早加入表内的数据。

5)对节点 N 的每个时间点重复步骤 2)—步骤 4),直至该节点所有时间点数据处理完毕。

6)将该过程应用到每个目标节点,直至所有节点数据都已生成完毕。

同时对于异常标记,标记规则如下:

1)对于故障前的先兆信息,如果该时间点在故障前的关联窗口时长(Δt)以内,且该时间点存在异常指标,则将该时间点标记为 1。

2)对于任意时间点,如果其平均偏移值超过过往平均偏移值与阈值系数的乘积(见式(3)),则将该时间点标记为 1。

3)其他情况则标记为 0。

通过时间点可以将运行特征向量与标记结果一一对应。运行特征向量将用于训练模型的输入,异常标记将作为模型的输出。本文采用多种机器学习方法建立正常与异常分类的故障异常预测模型,并比较其结果选择更优的模型。

获得了预测模型后,就可以通过监控程序在线获取运行监控数据,并基于运行特征向量纲要生成实时运行特征向量,在与模型进行匹配后得到该时刻的异常判定结果。当结果倾向于存在可能的后续故障时,可以将相关节点信息通过记录到文件或直接与故障响应机制连通的形式发送警报,从而起到辅助主动响应的目的。

4 实验与观察

本章将介绍节点故障异常预测方法的实验过程、结果以及一些观察结论。需要注意的是,由于预测模型的输入向量是以每次监控时间点为单位进行预测和结果比对的,但从应用角度来看,系统运行维护人员更关心的是故障异常是否被发现,故而在此处还需要一个将结果从每时间点的视角转换成每次故障异常的视角。这部分内容将在 4.2 节中详述。

4.1 实验设置

本次实验采集了中国科学院某超级计算系统约 100 个计算节点的时长约半个月的运行数据,为了进一步聚焦节点故障异常,在测试准备阶段,筛选出了其中 34 个在这段时间内发生过节点错误的节点,并以这些节点的运行数据(每半分钟生成一次数据记录)建立运行特征向量集合,数据条目总数量为 1 051 296 条。这些运行特征向量按节点名依次排列,相同节点内的向量按时间顺序排列。

在建模阶段,将向量的时间和节点信息取出单独存储,以备后续的结果转换和验证,而将纯数值部分作为模型的输入数据,将故障及异常标记部分作为模型的输出比对目标。所有向量数据按照其总量分成数量相近的 3 组,并采用同一模型运行 3 次,每次使用其中 2 组数据作为训练集,用剩余的 1 组作为测试集(即每次的训练集和测试集数据量比例约为 67∶33)。测试结果将以 3 次运行结果的总和统计得出。

本文引入了几种不同的典型的机器学习方法来建立预测模型,以期作为对比并从中选择最优的预测模型和结果。所选方法包括朴素贝叶斯分类器(Naïve Bayes,NB)、支持向量机(Support Vector Machine,SVM)、随机森林算法(Random Forest,RF)、长短期记忆网络(Long Short-Term Memory,LSTM)。其中前 3 种是作为分类器使用,其预测结果只有 0 和 1,而 LSTM 的预测结果是一个介于 0 和 1 之间的小数数值,因此在处理预测结果时,需要做相应的定义。设模型预测结果为 p,而实际异常标记结果为 l,则模型预测正确性表述如表 3 所列。

表 3 各建模方法预测结果判定条件

Table 3 Judging conditions of prediction results for modeling methods

指标	NB	SVM	RF	LSTM
真阳性(TP)	$p=1\&l=1$	$p=1\&l=1$	$p=1\&l=1$	$p>0.3\&l=1$
假阳性(FP)	$p=1\&l=0$	$p=1\&l=0$	$p=1\&l=0$	$p>0.3\&l=0$
假阴性(FN)	$p=0\&l=1$	$p=0\&l=1$	$p=0\&l=1$	$p\leqslant 0.3\&l=1$
真阴性(TN)	$p=0\&l=0$	$p=0\&l=0$	$p=0\&l=0$	$p\leqslant 0.3\&l=0$

4.2 结果转换与评估

在建模和测试阶段，故障异常预测是基于时间点来实现的，即根据每个时间点的运行特征执行一次异常的判断。然而在实际工作中，系统运行管理人员往往不需要关注每次监控时间点的异常判别和预测结果，只需要当故障异常可能发生时能够获得真实报警，并且不会因为虚假报警过多而响应不敏感即可。因此在计算预测方法的准确率等相关结果时，需要先将预测结果从以时间点为单位的形式转换成以每次预测阳性（即发现故障异常）为单位的形式，之后再计算其准确率等方面的性能指标。

首先，对于预测模型生成的 TP，FP，FN，TN 这 4 种结果，需要对其进行缩减和合并。本文将间隔时间不超过 5 分钟的相同预测结果称为一次连续的预测，并将其合并，最终只保留这次预测的起始时间、终结时间以及期间内符合该预测结果的时间点数量。假设模型生成的预测结果如表 4 所列，则经过缩减合并的预测结果如表 5 所列。

表 4 模型生成预测结果示例

Table 4 Example for prediction results

时间	预测结果	时间	预测结果
15:01:10	TN	15:07:10	TN
15:01:40	TN	15:07:40	TN
15:02:10	TN	15:08:10	TN
15:02:40	FN	15:08:40	TN
15:03:10	TP	15:09:10	TN
15:03:40	TP	15:09:40	TN
15:04:10	TP	15:10:10	TN
15:04:40	FN	15:10:40	TN
15:05:10	TP	15:11:10	FN
15:05:40	TP	15:11:40	TP
15:06:10	FP	15:12:10	TP
15:06:40	FP	15:12:40	TP

表 5 合并后的预测结果示例

Table 5 Example for merged results

序号	预测结果	起始时间	结束时间	发生次数
1	TN	15:01:10	15:02:10	3
2	FN	15:02:40	15:04:40	2
3	TP	15:03:10	15:05:40	5
4	FP	15:06:10	15:06:40	2
5	TN	15:07:10	15:10:40	8
6	FN	15:11:10	15:11:10	1
7	TP	15:11:40	15:12:40	3

合并预测结果后，还需要对其进行修剪和校对。比较常见的需要修建的情况是在 TP 前的短暂 FN 以及 TP 后的短暂 FP。前者的短暂 FN 通常是异常情况刚刚出现，持续时间这一特征还很短，可能会被认为是短时波动，因此没有立刻判定为异常，只要后续预测中发现并正确实现了 TP 的预测，就可以将此短暂 FN 预测修剪掉。而后者的短暂 FP 通常是前面 TP 的延续，预测受到之前一段时间异常的影响，由于其紧邻 TP，可以视作同一个阳性预测结果，因此也可将其修剪。如表 5 中的例子，其中第 2，4，6 行均符合修剪规则，则可以去除。经过结果合并后的预测效果示例如图 4 所示，图 4 表示数据集中某一节点在某段时间的部分运行特征与预测结果的对比情况，其中红色折线代表该节点所有关键监控指标的平均偏移值（使用左侧纵坐标轴单位），蓝色折线代表该节点的作业负载（使用右侧纵坐标轴单位），虚线箭头表示发生节点故障的时刻，浅红色区域代表在合并预测结果中预测为阳性（TP 或 FP）的时间段。可以看到，在对该节点的预测结果中比较准确地捕捉到了几次监控值的异常波动和一次节点故障。

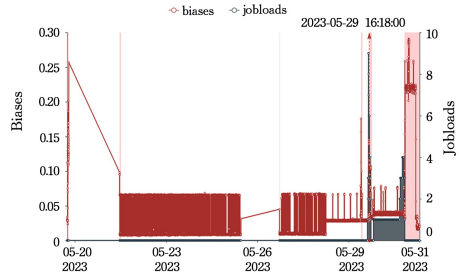

图 4 结果合并后预测效果示例（电子版为彩图）

Fig. 4 Examples of prediction effects after results merging

经过结果转换后，就可以重新计算预测模型的各项性能指标。本文选择了 3 个主要指标衡量预测结果：1）总体异常预测精度（AnomalyPrecision），即预测到的异常中有多大比例是真实发生的异常；2）总体异常召回率（AnomalyRecall），即所有真实异常中有多大比例被预测模型预测到；3）节点故障召回率（FailureRecall），即在发生过的节点故障中有多大比例在发生前的关联窗口时长内曾被模型预测到一次异常。目前，由于本文预测模型并未区分异常和故障的结果，因此无法统计预测节点故障的精度。

各个模型的测试结果如图 5 所示。其中在当前数据集中，随机森林算法的效果最佳，在 3 项指标中都表现出了最高的性能。LSTM 也给出了较好的预测表现，并且由于 LSTM 网络可以输入非整数型结果，因此具有更广阔的潜在应用范围。其中异常预测精度相对来说还不够理想，这可能是由于部分关键监控指标对于运行特征的表达不够精准，如其中一些指标表现出不定期的波动而不仅限于故障及异常发生时（见 4.3 节），对于这部分指标的处理方法有待进一步探索。

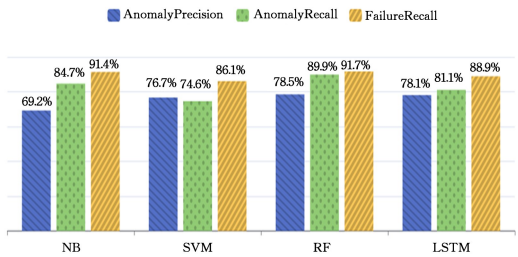

图 5 节点故障异常预测方法 3 项指标评测结果

Fig. 5 Evaluation results of three indicators for the node failure and anomaly prediction method

提前时间（Lead Time）是预测算法的另一项重要指标，它代表了目标事件能够提前多久被预测到。以超算节点故障和任务迁移的背景来说，如果提前时间不够长，将无法完成计算作业的状态保存和向其他节点迁移等工作流程，则失去了预测的意义。在本评估方法中，单次故障提前时间的计算方法为故障实际发生时间与在故障发生前的关联时间窗口之内成功进行 TP 预测的起始时间的时间差。即假设故障 f 的发生时间为 t_f，关联窗口时长为 Δt，且故障 f 的 TP 预测时间 t_{tp}

满足 $t_f-\Delta t < t_{tp} < t_f$，则成功预测故障 f 的提前时间 $l_{tf}=t_f-t_{tp}$。

对所有满足条件的故障预测的提前时间求平均值，就可以得到在测试数据集中成功被预测到的节点故障的平均提前时间，其结果如表 6 所列。

表 6 提前时间评测结果

Table 6 Evaluation result for lead time

(s)

method	故障平均提前时间	平均提前时间标准差
NB	2011.64	1008.36
SVM	1982.41	1011.95
RF	1875.18	1122.59
LSTM	1989.14	1062.07

总体上，各个方法的平均提前时间（越长代表性能越好）相差较小，4 种建模方法的平均提前时间都在半个小时以上。该时长对于故障预测的后处理来说完全足够，因此在建模方法选择时仍然倾向于以准确率作为衡量性能的主要标准。

4.3 观察

由于本方法的主要目标是针对节点的故障异常进行预测，因此在实践过程中，也获取到了与节点故障异常的关联指标及观察结果。这些观察结果可以对超算系统的问题溯源和解决提供帮助和支持，从而提升系统稳定性和运行效率。

观察 1 与节点故障异常关联度最高的指标属于网络类。

在与节点故障异常的关联中，有几项网络指标是在节点故障之前短时间内发生异常的比例最高的，大约占总数的 53%，其主要表现为网络发送、网络接收、发送字节、接收字节这几项指标的监控值突然升高。网络指标的异常可能是因为 Slurm 主进程与节点进程之间的大量重复性的通信。此类指标的异常可能由两种因素引发：一是节点本身确实发生了故障，使其无法正常接收和运行作业，因而造成无效的重复作业调度；二是网络通信组件发生异常，使得 Slurm 主进程与节点无法正常通信，因而引发反复尝试发送通信消息。

观察 2 磁盘读写异常也是引发节点故障的主要原因。

在基于本次实验目标数据集的统计中，磁盘类指标中的每一项都被选为关键监控指标，其中与节点故障关联度最高的指标包括磁盘读取时间、磁盘写入时间、磁盘等待时间、处理请求平均时间。磁盘类指标的特点是其发生异常的情况与常态情况相差极大，且其发生异常后伴随节点故障的比例超过 80%（相比之下，网络指标的异常次数更多，但会发生节点故障的比例相对较低），因此其成为节点故障预测的重点关注对象。

观察 3 突发性的作业负载增大也可能导致节点故障。

通过比对节点作业负载情况和故障发生时间发现，有一部分节点故障发生在突然的作业负载增加之后。大多数情况下，单个计算节点同时运行作业的数量在 0 到 2，但在某些情况下节点会同时被分配到 5 至 10 个作业。这种情况下的作业多是同一批次提交的一个较大计算任务的众多子作业，单个作业的规模较小，但同时提交到同一节点上可能会对其造成一些资源分配方面的压力，从而引发节点的不稳定运行，进而导致节点失效。

观察 4 处理器和内存问题难以用于判断故障异常。

尽管在关键监控指标中包含了一些处理器和内存指标，但观察结果表明，它们与节点故障的关联性较低。处理器指标本身波动比较频繁，被归为异常的时间点较多，因此异常引发节点故障的比例较低。而内存指标则相对稳定，其变化往往是缓慢且连续的，比如可用内存持续降低直到耗尽等，这种情况下无法通过某个时间点来界定其进入异常状态或预示故障。故而虽然节点故障往往伴随着一些处理器或内存监控指标上的变化，但很难反向从指标变化推断故障异常。

5 相关工作

大规模计算集群的节点故障是当前众多计算中心、数据中心普遍需要面对的问题，因此目前也已经有了一些应对方法，主要包括在故障前进行预测，以及当预测到故障时进行的处理。

在超级计算系统的故障异常预测领域，研究人员尝试根据所获得的运行数据选取不同的运行特征去实现预测。Gainaru 等[2]提出了一种将数据挖掘与信号分析相结合的方法，通过将系统监控值转换成二进制信号并计算不同信号组间的相似度和延迟来预测超算系统和应用的故障，他们还进一步讨论了故障预测的研究现状与挑战[3]。Das 等在超算系统节点故障预测方面进行了持续的研究工作，他们在 2018 年提出了 Desh[4]和 Doomsday[5]两种预测方法，其中 Desh 主要使用了节点的系统日志，将其转换成事件模板的序列并输入 LSTM 网络中进行故障预测，Doomsday 基于变量，将系统日志和作业调度日志结合形成许多文档，再通过提取文档关键词来建立预测模型输入向量。Das 基于 Desh 方法，将其进一步扩展为在线故障预测框架 Aarohi[6]。Alharthi 等[7]建立了一种基于日志的自监督预测模型，命名为 Time Machine。Frank 等[8]则重点关注减少节点故障的误报，以提升预测精度。Mohammed 等[9]建立了一个节点故障预测框架，并比较了多种机器学习方法的预测准确率。以上节点故障预测方法主要采用了事件类历史日志数据作为预测模型的运行数据特征采集目标，其中很多特征在本文采用的测试数据集中有所缺失，因此无法直接采用；还有一些基于故障前后文进行预测的方法虽然可以运行，但由于在本数据集中历史日志数据可能并不存在相关联的事件记录，因此预测效果很差，难以与本文方法做可行的对比。

此外，一些研究工作从节点负载的角度进行了故障的分析和预测。Li 等[10]探索了在数据中心由于节点工作负载造成的故障并实现了预测，Banjongkan 等[11]则进行了在作业不同阶段发生失败的预测方法研究。

除了节点故障外，一些研究人员也针对系统异常进行了判别和预测的工作。Chuah 等[12]总结了基于日志的关联分析工具，并比较分析了它们的异同以及在系统故障诊断及预测等方面的应用。Tan[13]进行了关于复杂分布式系统的在线性能异常预测的深入研究。Shen 等[14]提出了一种基于模糊系统的软件可靠性预测模型。Wang 等[15]建立了一种基于丰富特征的深度神经网络来实现磁盘故障预测，以提升系统存储稳定性。Jia 等[16]系统性地整理了基于日志数据的系统故

障诊断技术。Du 等[17]提出了基于深度学习的系统日志异常检测与诊断工具 DeepLog。Peng 等[18]提出的自动挖掘日志种类关联性的方法还使用了数据可视化来辅助对于结果的表现。Gao 等[19]通过在运行时库中增加检查和追踪等代码，实现了运行时故障定位的功能。Aksar 等[20]建立了基于半监督神经网络方法的高性能计算系统异常诊断框架 Proctor。Huang 等[21]提出了一种层级式转换的系统日志异常检测方法 HitAnomaly。系统异常判定可以作为节点故障预测的一种辅助手段，为预测提供不同角度的印证和关联性分析。

数据处理与标记是开展异常分析、故障预测等工作的重要前置任务。在这方面，Zhang 等[22]对日志解析的相关研究工作进行了详尽的梳理。Wittkopp 等[23]提出了一种面向日志的弱监督分类方法，可以自动对异常事件进行评估和标记。

对于故障节点上的中断作业的处理问题，尽管这不属于与本文方法类似的方法，但仍与故障预测的总体目标有重要关系。检查点技术是恢复中断作业的一种有效方法，它通过记录进程运行状态或关键参数来实现作业的复现和续算。ULFM 及其扩展 CPPC[24]是面向 MPI 的检查点恢复工具，主要应用于高性能计算并行程序。Benoit 等[25]尝试使用检查点与复制相结合的方法来应对节点故障和异常。任务迁移也是应对故障异常的主要方法之一，可以将任务从风险节点转移到正常节点中，而容器技术的发展为任务迁移提供了更便捷的应用模式。Ma 等[26]实现了基于 Docker 容器迁移的边缘计算系统的高效业务切换。Zhao 等[27]提出了基于容器容错池的迁移机制来应对云计算集群中的节点故障问题。Luo 等[28]提出了一种预测网络带宽优化传输的方法来提升边缘服务容器迁移的性能。

结束语 本文提出了一种超算系统节点故障异常预测方法，针对计算节点的故障风险增加、影响系统稳定性和运行效率的问题，该方法基于无监督学习及少量人工辅助的方式，通过数据整理、异常标记、特征选取、向量建立等步骤建立预测模型，实现了对目标超算系统节点故障异常的识别和预测功能。测试结果表明，该方法取得了 78.5%的识别精度和接近 90%的召回率，而节点故障召回率达到了 91%。同时，得到了部分观察结论：在目标数据集中，节点故障是由于网络和磁盘性能异常造成的概率更高，同时也有部分由作业负载引发，而受处理器和内存影响造成故障的可能性很小。

在未来工作中，我们希望能够开展对于故障和异常的精细化区分、优化特征选取方法等研究，以进一步提高预测精度，并更细致地解析引发节点故障的原因，例如网络、磁盘等不同种类指标对于故障的独立影响，从而加深对超算系统节点故障的理解。同时，我们将尝试建立在线节点故障预测结合任务迁移的工具，实现在故障发生前将计算任务转移至正常节点的功能，以减少机时浪费，提升超算资源的使用效率。

参 考 文 献

[1] WANG X D, ZHAO Y N, XIAO H L, et al. LTmatch: A Method to Abstract Pattern from Unstructured Log[J]. Applied Sciences, 2021, 11(11): 5302.

[2] GAINARU A, CAPPELLO F, SNIR M, et al. Fault prediction under the microscope: A closer look into HPC systems [C]// Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12). 2012.

[3] GAINARU A, CAPPELLO F, SNIR M, et al. Failure prediction for HPC systems and applications: Current situation and open issues [J]. International Journal of High Performance Computing Applications, 2013, 27(3): 273-282.

[4] DAS A, MUELLER F, SIEGEL C, et al. Desh: Deep Learning for System Health Prediction of Lead Times to Failure in HPC [C]// The 27th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC'18). 2018.

[5] DAS A, MUELLER F, HARGROVE P, et al. Doomsday: Predicting Which Node Will Fail When on Supercomputers [C]// Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'18). 2018.

[6] DAS A, MUELLER F, ROUNTREE B. Aarohi: Making Real-Time Node Failure Prediction Feasible [C]// The 34th IEEE International Parallel & Distributed Processing Symposium (IPDPS2020). 2020.

[7] ALHARTHI K A, JHUMKA A, DI S, et al. Time Machine: Generative Real-Time Model for Failure (and Lead Time) Prediction in HPC Systems [C]// The 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2023.

[8] FRANK A, YANG D, BRINKMANN A, et al. Reducing False Node Failure Predictions in HPC [C]// The IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC). 2019.

[9] MOHAMMED B, AWAN I, UGAIL H, et al. Failure prediction using machine learning in a virtualised HPC system and application [J]. Cluster computing, 2019, 22: 471-485.

[10] LI J, WANG R, ALI G, et al. Workload Failure Prediction for Data Centers [C]// The IEEE 16th International Conference on Cloud Computing (CLOUD). USA, 2023.

[11] BANJONGKAN A, PONGSENA W, KERDPRASOP N, et al. A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms [J]. Journal of Advances in Information Technology, 2021, 12(2): 84-92.

[12] CHUAH E, JHUMKA A, MALEK M, et al. A Survey of Log-Correlation Tools for Failure Diagnosis and Prediction in Cluster Systems [J]. IEEE Access, 2022, 10: 133487-133503.

[13] TAN Y M. Online Performance Anomaly Prediction and Prevention for Complex Distributed Systems [D]. North Carolina: North Carolina State University, 2012.

[14] SHEN Q, LOU J G, ZHANG X T, et al. Failure prediction by regularized fuzzy learning with intelligent parameters selection [J]. Applied Soft Computing Journal, 2021, 100: 106952.

[15] WANG L F, LI D J. SFFDD: Deep Neural Network with Enriched Features for Failure Prediction with Its Application to Computer Disk Driver [J]. arXiv: 2109.09856, 2021.

[16] JIA T, LI Y, WU Z H. Survey of State-of-the-art Log-based Failure Diagnosis[J]. Journal of Software, 2020, 31(7): 1997-2018.

[17] DU M, LI F, ZHENG G, et al. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning[C]// Computer and Communications Security. ACM, 2017.

[18] PENG W, LI T, MA S. Mining logs files for data-driven system management[J]. ACM SIGKDD Explorations Newsletter, 2005, 7(1): 44-51.

[19] GAO J G, ZHENG Y, YU K, et al. Runtime Fault Location Method for Sunway Supercomputer[J]. Journal of Computer Research and Development, 2024, 61(1): 86-97.

[20] AKSAR B, ZHANG Y J, ATES E, et al. Proctor: A Semi-Supervised Performance Anomaly Diagnosis Framework for Production HPC Systems[C]// International Conference on High Performance Computing. Cham: Springer, 2021: 195-214.

[21] HUANG S H, LIU Y, FUNG C, et al. HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log[J]. IEEE Transactions on Network and Service Management, 2020, 17(4): 2064-2076.

[22] ZHANG T Z, QIU H, CASTELLANO G, et al. System Log Parsing: A Survey [J]. arXiv: 2212. 14277, 2022.

[23] WITTKOPP T, ACKER A, KAO O. Progressing from Anomaly Detection to Automated Log Labeling and Pioneering Root Cause Analysis [J]. arXiv: 2312. 14748, 2023.

[24] LOSADA N, CORES I, MARTÍN M J, et al. Resilient MPI applications using an application-level checkpointing framework and ULFM[J]. Journal of Supercomputing, 2017(73): 100-113.

[25] BENOIT A, CAVELAN A, CAPPELLO F, et al. Coping with silent and fail-stop errors at scale by combining replication and checkpointing [J]. Journal of Parallel and Distributed Computing, 2018, 122: 209-225.

[26] MA L L, YI S H, LI Q. Efficient service handoff across edge servers via docker container migration [C]// Proceedings of the Second ACM/IEEE Symposium on Edge Computing. ACM, 2017: 1-13.

[27] ZHAO Q, XIE S Q, HAN K, et al. Container Migration Based on Combination of Remote Direct Memory Access and Check Point[J]. Journal of Frontiers of Computer Science and Technology, 2019, 13(12): 1995-2007.

[28] LUO C, CUI Y, LIN Y S. Container Migration Method Based on Bandwidth Prediction and Adaptive Compression[J]. Computer Engineering, 2022, 48(5): 200-207.

ZHAO Yining, born in 1983, Ph. D, senior engineer. His main research interests include data analysis, high-performance computing and distributed systems.

(责任编辑:李亚辉)