基于时间序列分析的杀手级任务在线识别方法

唐红艳[1] 李 影[1,2] 贾 统[1] 袁小雍[1]
(北京大学软件与微电子学院 北京 100871)[1] (北京大学软件工程国家工程研究中心 北京 100871)[2]

摘 要 通过分析Google集群中任务的失效次数和失效模式,找到具有高失效频次和连续失效特征的杀手级任务。杀手级任务不仅影响云计算系统上应用运行的可靠性与可用性,而且会浪费大量资源并显著增加调度负载。在杀手级任务资源使用模式的基础上,提出一种基于时间序列的在线识别方法,以利用资源使用时间序列在失效早期准确识别出杀手级任务并通知云计算系统采取前摄性失效恢复措施,从而避免不必要的重复调度和资源浪费。实验结果表明,该方法能够以98.5%的准确率在平均3%的失效时间内识别出杀手级任务,同时节约96.75%的系统资源。

关键词 云计算系统,杀手级任务,在线识别,时间序列,资源使用模式,失效频率
中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.010

Time Series Based Killer Task Online Recognition Approach

TANG Hong-yan[1] LI Ying[1,2] JIA Tong[1] YUAN Xiao-yong[1]
(School of Software and Microelectronics, Peking University, Beijing 100871, China)[1]
(National Engineering Center of Software Engineering, Peking University, Beijing 100871, China)[2]

Abstract By analyzing failure frequency and failure patterns in Google cluster dataset, this paper fond what are called as killer tasks that suffer from frequent and continuous failure. Killer task is a big concern of cloud system as it causes unnecessary resource wasting and significant increase of scheduling overhead. In this paper, an online recognition approach was proposed to make use of the resource usage time series to recognize killer tasks precisely at the very early stage of their occurrence so that proactive actions can be taken to avoid rescheduling and resource wasting. The experiment results show that the proposed approach performs a 98.5% precision in recognizing killer tasks at 3% of failure duration, with a 96.75% resource saving for the cloud system averagely.

Keywords Cloud system, Killer tasks, Online recognition, Time series, Resource usage pattern, Failure frequency

1 引言

云计算系统大规模、分布式的特性,使得各类失效频发[2],其中任务失效是最普遍的失效类型。为提高可用性,目前处理任务失效的主要方法是重新调度[4]。然而,云计算系统中存在某些任务不仅在重新调度后无法快速恢复,还会反复失效[3],这不仅浪费了系统资源,还增加了调度器负载,这类任务被称为"杀手级任务"。如何准确、实时地识别出杀手级任务,为系统保留足够的时间以采取前摄性措施,对动态变化的云计算系统是一个技术挑战。

本文首先对Google集群数据集[1]中任务的失效频次进行分析,发现其中有7386个频繁且连续失效的杀手级任务。之后,本文深入挖掘对比杀手级任务和非杀手级任务的特征。基于分析结果,提出一种基于时间序列的杀手级任务在线识别方法。实验结果表明,本文方法以98.5%的准确率和平均97%的提前时间识别出杀手级任务,为云计算系统平均保留410min以采取有效措施,并节约96.75%的系统资源。

本文第2节介绍相关工作;第3节对比分析杀手级任务和非杀手级任务的资源使用模式及失效频率特征;第4节提出基于时间序列的杀手级任务在线识别方法;第5节分析实验结果;最后总结全文。

2 相关工作

2.1 Google数据集分析工作

自Google发布数据集以来,不少学者对其展开了分析。文献[7]对集群中机器和任务的异构性和动态性进行了分析,并预测了任务资源。文献[8]刻画了任务的资源使用特征并提出一种基于k-means的任务分类算法,发现少量运行时间长的任务消耗了系统中大部分的CPU和内存资源。文献[9]分析发现,应用程序的资源使用量和不同类型的应用程序数量均遵循帕累托分布。这些研究主要通过分析提取资源使用模式,但均未考虑任务或作业失效。

到稿日期:2015-11-30 返修日期:2016-02-29 本文受深圳市科技计划重点项目(JSGG20140516162852628)资助。
唐红艳(1993—),女,硕士生,主要研究方向为云计算系统、大数据分析,E-mail:hytang.nancy@foxmail.com;贾 统(1993—),男,博士生,主要研究方向为云计算与大数据系统可用性,E-mail:jia.tong@foxmail.com;袁小雍(1990—),男,博士生,主要研究方向为云计算和深度学习系统,E-mail:xiaoyong.yuan@foxmail.com。

近年来，一些研究人员从失效的角度对 Google 集群数据集加以分析。文献[6]对机器和任务失效的统计分布进行分析，同时研究了机器和任务的平均失效时间(MTTF)和平均修复时间(MTTR)。文献[10]从资源使用、调度约束和用户特征的角度分析任务失效，并基于失效相关的特征对用户进行聚类。文献[11]以任务属性和资源使用为特征，利用递归神经网络模型对任务失效进行预测。基于以上研究工作，本文从任务资源使用的时间序列特性出发，从任务失效的角度对 Google 集群数据集进行全面而深入的分析，并基于分析结果提出在线杀手级任务识别方法。

2.2 在线失效预测

在线失效预测是失效管理的重要研究领域，许多学者在这一领域进行了大量研究。通过分析作业失效与资源利用率的关联，文献[12]利用简化的贝叶斯网络模型预测网格计算系统中的作业失效。文献[13]分别利用决策树和频繁项挖掘算法提取出主故障和伴随故障特征日志，从而实现高精度的故障检测。文献[14]从日志中抽取消息模式并将其与失效相关联，进而用贝叶斯方法进行失效预测。以上研究工作大多使用模式匹配进行失效预测，而目前基于时间序列模型的失效预测方法因其高准确率、高敏感度和低计算成本的优点逐渐成为研究热点。

在基于时间序列的失效预测方面，文献[15]提出一种基于时间序列和故障树分析的失效预测方法，该方法首先利用 ARMA 模型预测未来的资源使用量，之后通过构造故障树利用预测数据预测系统的可靠性。文献[16]利用 Volterra 序列预测监控数据的到达时间，进而用决策树模型预测系统未来的状态。与这些研究工作不同，本文聚焦于任务失效而不是整个系统的失效。此外，与模拟实验环境相比，本文使用的是 Google 集群真实运行数据集。

3 杀手级任务分析

3.1 杀手级任务

本文首先分析 Google 集群中任务的失效频次，发现在 25424731 个任务中共有 872796 个任务发生过一次以上失效，其中 27141 个(3%)任务经历过 200 次以上失效，失效次数远大于其他任务，且失效总频次大于其余 97%的任务。因此，本文以这 27141 个高失效频次的任务作为分析目标并进一步分析它们的失效模式，发现其中有 7386 个任务失效状态连续，两次失效间几乎不存在运行状态，这意味着这些任务无法通过重启进行快速恢复。虽然它们会在每次失效后被立即重新调度，但这不仅无法从根本上解决失效，还会造成系统资源的大量浪费，给调度器带来不必要的负载。本文将此类任务定义为杀手级任务(Killer Tasks)，剩余的 19755 个高失效频次但失效相对不连续的任务则为非杀手级任务(Non-Killer Tasks)。定义如式(1)所示：

$$Label(i)=\begin{cases}\text{killer task}, FF_i\geqslant 200 \text{ and } (\sum_{t=2}^{T_i} f(t)*f(t-1)\geqslant \frac{\sum_{t=1}^{T_i} f(t)}{2}) \\ \text{non-killer task}, FF_i\geqslant 200 \text{ and } (\sum_{t=2}^{T_i} f(t)*f(t-1)< \frac{\sum_{t=1}^{T_i} f(t)}{2})\end{cases}, f(t)=\begin{cases}1, & State_t=\text{Fail} \\ 0, & \text{otherwise}\end{cases} \tag{1}$$

其中，FF_i 和 T_i 分别为任务 i 的失效频次和驻留时间，$f(t)$ 是一个分段函数，若时刻 t 为失效状态，则函数值为 1。

3.2 资源使用模式

表 1 列出了杀手级任务的状态迁移模式。“运行-失效”表示任务运行一段时间后失效。“失效-运行”则表示任务初始为失效状态，后通过重新调度恢复。类似地，“运行-失效-运行”表示任务起初正常运行，经历一段失效期后恢复。“失效-运行-失效”表示任务自提交便开始失效，期间经历数次失效到运行的转换，但最终为失效状态。“失效”表示任务始终为失效状态。结果表明，99.2%的杀手级任务无法通过重新调度从失效中恢复，甚至 38.3%的任务自提交进入系统便始终失效。而非杀手级任务常呈现出“运行-失效”反复循环的模式，且运行时长远大于失效时长。这表明，重新调度能使非杀手级任务从短暂失效中快速修复。

表 1 杀手级任务状态迁移模式

状态迁移模式	任务数量	百分比/%
运行-失效-运行	62	0.839
运行-失效	4493	60.831
失效-运行	1	0.014
失效-运行-失效	2	0.027
失效	2828	38.289

为进一步揭示杀手级任务的资源使用特征，本文分析了其 CPU 使用情况，如图 1 所示。

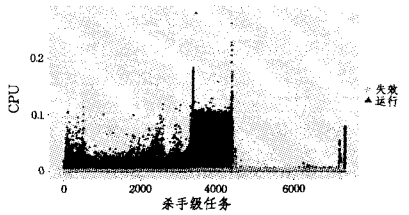

图 1 杀手级任务的资源使用情况

分析结果表明，大部分杀手级任务在失效时刻的 CPU 使用量与正常时刻存在差异。根据统计结果，在经历过失效和运行的杀手级任务中，99.1%的任务在失效时段和运行时段的 CPU 使用量不同，甚至有 14 个任务在失效时的 CPU 消耗量为 0。此外，本文还对杀手级任务在失效和运行时段的 CPU 使用量进行 Mann-Whitney U 检验。检验结果表明，除始终失效和 CPU 使用量始终为 0 的任务无法进行假设检验外，99.1%的杀手级任务的 p 值小于 0.05，这说明大部分杀手级任务在运行和失效时的 CPU 消耗量存在显著差异。又由于杀手级任务的失效和运行均连续，因此其资源使用量存在明显分段趋势。相反，大部分非杀手级任务在运行和失效时刻的资源使用虽存在差异，但未呈现出特定增大或减小的趋势，即一个任务可能在某失效时刻消耗的资源大于运行时刻，却在另一失效时刻消耗小于正常值的资源。又由于非杀手级任务失效不连续，因此它们的资源使用无明显分段趋势。

3.3 失效频率分析

为度量任务的失效频率特征，本文对杀手级和非杀手级任务在长度为 5 分钟的时间窗口内的失效频次进行了分析，结果如图 2 所示。从图中可以看出，大部分杀手级任务在某些窗口内发生过多次失效，37.8%的杀手级任务至少在一个窗口内失效 8 次以上，甚至有一个任务的最大失效次数为 16 次，意味着该任务在某些窗口内平均每分钟失效 3 次以上。相反，除 0.03%的非杀手级任务在某个窗口内发生过 4 次失效外，几乎所有非杀手级任务在所有窗口的失效次数均不超过 2 次，这与杀手级任务形成了鲜明对比。

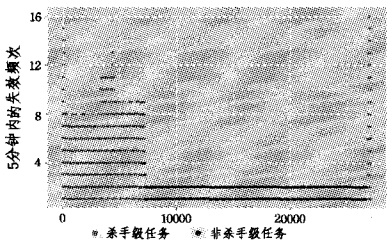

图 2　失效频率模式

4 杀手级任务在线识别

根据上文分析，杀手级任务和非杀手级任务存在不同的资源使用模式。本文提出一种基于时间序列的杀手级任务在线识别算法，其利用早期资源的使用特征提前识别出杀手级任务并通知云计算系统采取前摄性措施。本文方法分为失效频率阈值学习和杀手级任务识别两个阶段。

4.1 失效频率阈值学习算法

假设任务 i 的驻留时间为 T_i，$F_{i,t}$ 和 $R_{i,t}$ 分别表示任务 i 在时刻 t 的失效次数和资源使用量。失效频率阈值学习算法从非杀手级任务中学习失效频率阈值作为杀手级任务和非杀手级任务的分界。算法分为以下 3 个步骤。

(1)划分资源异常窗口

根据上文分析结果，杀手级任务的资源使用存在明显分段趋势，因此可将任务的生命周期划分为若干个资源异常窗口。资源异常窗口指资源使用量与其他窗口差异显著的时段。本文用资源使用量的变动程度来判断其是否异常。若某时刻资源使用变动量大于阈值，则该时刻属于另一窗口；否则该时刻与上一时刻属于同一窗口。判断某一时刻是否为资源异常点的方式如下：

$$\frac{|R_{i,t}-R_{i,t-1}|}{p(R_i)}>V_{thre} \tag{2}$$

其中，函数 p 用于求取该任务的资源使用数量级，V_{thre} 表示资源变动阈值。

(2)计算失效频率

将每个任务的生命周期划分为若干窗口后，需计算每个窗口内的失效频率。失效频率等于窗口内的总失效次数除以该窗口的时长。由于每个任务存在多个资源异常窗口，因此本文将所有窗口内失效频率的最大值作为该任务的失效频率。由上文分析可知，资源异常时段可能是失效的体现，也可能是正常运行时的资源波动。对于在两个失效时段内夹杂少许运行时刻的情况，仅采用失效时段内的失效频率无法获得正确的失效频率，从而导致无法正确区分杀手级任务和非杀手级任务，因此本文使用资源异常窗口而不是失效时段内的失效频率进行计算。

(3)学习失效频率阈值

首先拟合出所有任务的失效频率的累积分布函数。之后算法将满足累积分布函数 $F(x)$ 值为 0.9 的 x 作为失效频率阈值。这一失效频率阈值满足 90%的训练任务的失效频率，也即它在 90%的置信度上能够代表训练任务。不采用所有任务中的最大失效频率作为失效频率阈值的原因是，在训练任务中存在一些任务的失效频率远远大于其他任务，甚至可能接近于杀手级任务，若考虑这些任务的失效频率，可能会导致识别结果出现偏置。

4.2 杀手级任务在线识别算法

杀手级任务识别是一个在线过程，需要利用任务的资源使用情况和失效数据判断任务是否为杀手级任务。当输入任务 i 在时刻 t 的资源使用量和失效次数时，算法执行如下步骤：

1)根据式(2)判断时刻 t 是否处于当前资源异常窗口，若是，执行步骤 3)；否则，执行步骤 2)。

2)开启一个新的资源异常窗口，并将窗口内的失效次数和失效频率初始化为 0，随后执行步骤 3)。

3)将时刻 t 加入当前资源异常窗口并计算窗口内的失效频率，其值等于该窗口内的失效总次数除以窗口长度。若失效频率大于失效频率阈值，则该任务是杀手级任务，算法结束；否则，等待下一时刻数据输入并执行步骤 1)。

5 实验分析

5.1 实验设置

在 Google 数据集中的 7 种资源使用指标[5]中，本文采用平均 CPU 使用量进行实验。除准确率、召回率、正确率和误报率外，本文还引入前置时间(LT)和资源节约(RS)作为评价指标。这两个指标基于杀手级任务会在识别后被立即杀死的假设。若任务 i 在时刻 t_{ir} 被识别，则其前置时间等于时刻 t_{ir} 后它在系统中的驻留时间，$LT=T_i-t_{ir}$；节约资源等于它在时刻 t_{ir} 后消耗的总资源，$RS=\sum_{t=t_{ir}}^{T_i}R_{i,t}$。

5.2 实验结果

5.2.1 识别结果分析

为对比识别结果，本文选用不同的数据进行两组实验。实验 1 采用本文对杀手级任务的定义，即高频次连续失效，因此共 7386 个杀手级任务和 19755 个非杀手级任务。实验 2 仅考虑失效频次，因此选择 1151 个失效超过 500 次的任务作为杀手级任务，从非杀手级任务中随机抽样 5146 个失效超过 200 次的任务进行实验。对这两组实验，均从非杀手级任务中随机抽取 70%进行训练，剩余 30%的非杀手级任务和所有杀手级任务进行测试。如表 2 所列，本文方法在杀手级任务识别问题上性能优异，正确率和召回率均大于 90%。另外，实验 1 的识别效果优于实验 2，尤其是在误报率和准确率方面。这表明对杀手级任务更细化的定义能拉大杀手级任务和非杀手级任务间的差距，进而降低误报率，提升准确率。

表 2　识别结果对比/%

	准确率	召回率	正确率	误报率
实验 1	98.53	99.75	99.04	1.84
实验 2	85.70	96.40	91.60	11.98

5.2.2 识别效果分析

图 3 示出了算法对杀手级任务的预识别程度，用识别前后的失效次数表征。如图 3 所示，识别前的失效次数远少于识别后的失效次数，表明算法在发生少量失效时便能够识别杀手级任务并通知云计算系统，从而避免大量不必要的失效。如某失效 40393 次的杀手级任务在第 11 次失效时便被识别，有效避免了 40382 次失效。

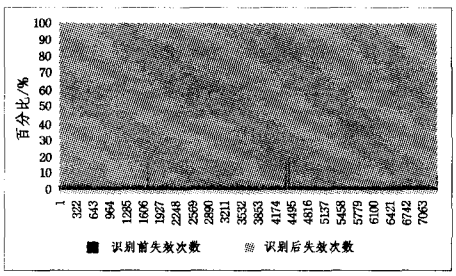

图 3 识别前后的失效次数

算法在资源节约和前置时间方面的性能如图 4 所示。前置时间百分比指任务的前置时间占其失效总时间的百分比，资源节约百分比为任务资源节约量占其失效后资源总消耗的百分比。结果表明，大部分任务的前置时间接近于失效总时长，且算法平均在 3%的失效时长内识别杀手级任务，为系统平均保留 410min 以采取有效措施。其中某个任务的前置时间为 28.99 天，即在首次失效后的第 0.24 小时内便被识别。此外，98.9%的任务的资源节约百分比大于 90%，所有任务的平均资源节约百分比为 96.75%。

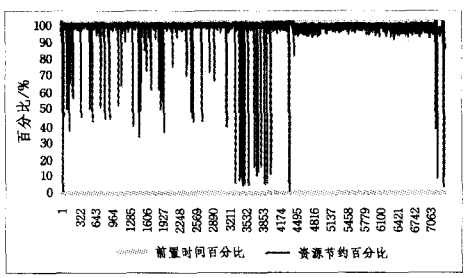

图 4 资源节约和前置时间

结束语 具有高失效频次和连续失效特征的杀手级任务不仅影响云计算系统的可靠性与可用性，还会造成大量的资源浪费并显著增加调度负载。本文从时间序列的角度对杀手级任务的资源使用模式进行探究，并在此基础上提出一种基于资源使用时间序列的杀手级任务在线识别方法。实验结果表明，本文方法能够以 98.5%的准确率识别出杀手级任务，平均节约 96.75%的系统资源。今后将通过引入更多资源使用指标(如内存使用量)以及利用机器学习模型来优化算法。此外，也考虑将方法应用于各种不同类型的云计算系统。

参 考 文 献

[1] Google Cluster Data[OL]. https://code.google.com/p/google-clusterdata/wiki/ClusterData2011_2.

[2] WANG Y J, SUN W D, ZHOU S, et al. Key Technologies of Distributed Storage for Cloud Computing[J]. Journal of Software, 2012, 23(4): 962-986. (in Chinese)
王意洁，孙伟东，周松，等. 云计算环境下的分布存储关键技术[J]. 软件学报，2012，23(4)：962-986.

[3] REISS C, TUMANOV A, GANGER G R, et al. Towards understanding heterogeneous clouds at scale: Google trace analysis: Technical Report ISTC-CC-TR-12-101[R]. Intel Science and Technology Center for Cloud Computing, 2012: 84.

[4] SOUALHIA M, KHOMH F, TAHAR S. Predicting Scheduling Failures in the Cloud[J]. arXiv preprint arXiv: 1507.03562, 2015.

[5] REISS C, WILKES J, HELLERSTEIN J L. Google cluster-usage traces: format + schema[R]. Google Inc., Mountain View, CA, USA, 2011.

[6] GARRAGHAN P, TOWNEND P, XU J. An empirical failure-analysis of a large-scale cloud computing environment[C]// 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering (HASE). IEEE, 2014: 113-120.

[7] REISS C, TUMANOV A, GANGER G R, et al. Heterogeneity and dynamicity of clouds at scale: Google trace analysis[C]// Proceedings of the Third ACM Symposium on Cloud Computing. ACM, 2012: 7.

[8] MISHRA A K, HELLERSTEIN J L, CIRNE W, et al. Towards characterizing cloud backend workloads: insights from Google compute clusters[J]. Acm Sigmetrics Performance Evaluation Review, 2010, 37(4): 34-41.

[9] DI S, KONDO D, CAPPELLO F. Characterizing Cloud Applications on a Google Data Center[C]// 2013 42nd International Conference on Parallel Processing (ICPP). IEEE, 2013: 468-473.

[10] CHEN X, LU C D, PATTABIRAMAN K. Failure analysis of jobs in compute clouds: A google cluster case study[C]// 2014 IEEE 25th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2014: 167-177.

[11] CHEN X, LU C D, PATTABIRAMAN K. Failure Prediction of Jobs in Compute Clouds: A Google Cluster Case Study[C]// 2014 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2014: 341-346.

[12] FADISHEI H, SAADATFAR H, DELDARI H. Job failure prediction in grid environment based on workload characteristics [C]// 14th International CSI Computer Conference, 2009 (CSICC 2009). IEEE, 2009: 329-334.

[13] RAO X, WANG H M, CHEN Z B, et al. Detecting Faults by Tracing Companion States in Cloud Computing Systems[J]. Journal of Computers, 2012, 35(5): 856-870. (in Chinese)
饶翔，王怀民，陈振邦，等. 云计算系统中基于伴随状态追踪的故障检测机制[J]. 计算机学报，2012，35(5)：856-870.

[14] WATABABE Y, OTSUKA H, SONODA M, et al. Online failure prediction in cloud datacenters by real-time message pattern learning[C]// 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom). IEEE, 2012: 504-511.

[15] CHALERMARREWONG T, ACHALAKUL T, SEE S C W. Failure Prediction of Data Centers Using Time Series and Fault Tree Analysis[C]// 2012 IEEE 18th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2012: 794-799.

[16] LIN R, WU B, YANG F, et al. An efficient adaptive failure detection mechanism for cloud platform based on volterra series [J]. China Communications, 2014, 11(4): 1-12.