### 国内外关于本选题的研究现状、水平和发展趋势：

日志是分布式计算系统开发和维护的重要依据，随着现代化分布式系统规模的不断扩大和执行任务的复杂化，系统在运行过程中会产生海量的日志，发生许多难以定位和预测的故障问题。传统人工分析日志已经变得不切实际。基于日志的自动化异常检测模型，已经成为开发运维人员分析定位故障问题的重要工具，如基于聚类f方法的日志异常检测系统LogCluster[1]已经由微软实现并在部署在了大规模的服务中。

在进行日志分析前通常要将日志解析为结构化数据，并进行特征提取和分类等操作。为了实现日志的自动解析，目前提出的方法包括：基于词频对日志进行聚类(LogCluster)[1]；使用最长公共子序列找到公共模板进行分类(spell)[2]；使用前缀树对日志进行聚类(drain)[3]。但这些解析仍然要求用户对日志文本具备较全面的了解，比如drain需要人工提前设置日志中变量的正则表达式，才能识别出变量。

目前的研究主要通过识别异常的日志模式来检测异常，包括利用日志序列的IDF等频率特征进行聚类识别异常[1]；基于自动机、流程图模型识别工作流的执行异常[4][6]；通过学习日志的共现模式发现不常见的异常模式。随着深度学习技术的发展，更多研究开始将自然语言处理中的许多方法引入日志的异常检测，比如deeplog论文使用lstm的方法[5]，将序列中出现概率较低的日志判定为异常。

对日志中异常模式检测的研究将会朝着对不同软件系统日志的无差别处理，自动化程度不断提高的方向发展，并且会加强对日志语义信息的提取，从而提高在日志格式改变时模型的鲁棒性。

选题研究的目的和意义：

日志是一种介于自然语言和结构化语言之间的数据形式。日志条目打印的顺序、间隔时间等信息，可以反映任务执行状态和系统负载情况。日志异常检测的模型通常由以下三部分组成：（1）日志解析（2）离线的模型训练（3）对实时产生日志进行异常检测。

关于日志的异常检测目前的研究主要存在以下的问题：

1）在日志解析步骤，需要人工基于日志打印格式，设计日志模板和特征提取的规则，对日志进行分类，要求设计者对特定应用框架有深入了解。

2）同时日志格式会随着软件的迭代更新而发生变化，使得原本基于日志格式设计的异常检测算法效果会变差。

3）日志由并发的线程任务同时产生，乱序打印的日志加大了异常检测和定位的难度。以往的研究主要是通过taskId、blockId等标识符从乱序的日志中识别出独立的工作流[6][7]，在缺少有效标识符的日志系统中，提取出单一任务打印出的日志成为了一个难题。

为了解决以上的问题，本项目从自然语言处理得到启发，计划实现一种鲁棒的基于深度学习的计算框架日志异常检测工具。该工具可以从并发任务乱序打印的日志中，提取出独立的工作流日志序列，最终模型同时具备识别可能出现的程序执行异常和系统性能异常，最终以插件的形式为日志管理系统提供异常检测服务。

本项目研究的日志异常主要包括以下两种类型 1. 序列异常：性能扰动或局部异常[8]，可能对任务的正常执行造成干扰，具体表现为日志条目的打印顺序出错，这类异常通常不会直接导致任务失败，不会显式地在日志中告警，但具有一定几率引发任务执行故障。 2. 时延异常：系统负载过高、资源消耗过多等性能异常，影响任务的正常执行，具体表现为日志打印时延变长，可以反映系统性能状况的变化。

为了充分利用日志具有的自然语言与结构化数据的特征，本项目结合了 1）传统机器学习领域的决策树和聚类等算法 2）深度学习的自然语言处理领域的语义提取的预训练模型和循环神经网络。试图从乱序生成的日志中提取单一任务的日志，并学习日志序列分别在两种异常发生时的变化规律。最终实现对以上两种异常的有效识别，对系统故障提前发出告警。

选题研究的计划进度及可行性论述：

1. 日志解析部分：本研究主要针对当前方法存在的两个问题进行改进

（1）以往日志解析需要人工根据日志结构，定义变量的正则表达式进行解析，比如ip、block\_id其它执行任务变化的变量。存在的问题包括：需要对日志中变量类型有先验知识；当日志结构或变量格式改变后，方法可能失效

（2）基于最长公共子序列方法训练的解析器，对日志模板和变量部分的提取精度较高，但时间复杂度高且具有难以动态扩展的缺点；而基于前缀树的解析器能够提取变量部分的精度较低，且反映的日志信息也比较少。

本论文对drain进行改进，drain是使用前缀树的方法来解析日志[3]，树的节点会由一个token(单词)标记，前缀树会将日志当前位置的token匹配到对应的节点上，并继续搜索；若无匹配的节点或当前token是变量，则匹配到“\*”节点("\*"表示匹配任意token)。

经观察，drain构建的前缀树如果有遗漏的变量未事先定义，那么大量的解析步骤token都会匹配到"\*"表示的节点，分类效果会非常差；同时原本不同的变量类型，可能共用一个节点，这样的话难以区分不同类型的变量。

本篇论文通过根据一定策略合并相似子树的方法，可以自动识别日志条目中的变量部分。由于同类日志的模板相同，由于变量不同而生成的节点下的子树仍然相同，定期合并相似子树，不仅可以识别出日志模板中的变量部分，也可以提高树的分类效果。

2. 日志聚类和编码部分：存在两种聚类的方式：（1）根据日志的格式结构特征进行聚类，即直接使用步骤1中前缀树中搜索得到的叶节点标签作为分类结果；（2）根据日志的语义特征进行聚类，使用自然语言处理领域的语义提取预训练模型bert，对日志进行编码，并对编码结果进行聚类。

本篇论文将使用基于语义进行分类的方法，引入日志的实际语义信息，提高后面检测的准确性和可解释性，同时提高模型的鲁棒性，在应用迭代更新导致日志格式发生变化时，模型仍然有效。

由于bert编码执行的迭代计算有巨大的时间开销，本论文通过建立从日志类别到bert编码的映射缓存，提高模型实时计算效率。当产生一条新日志时，前缀树可以迅速解析日志条目，得到对应的类别，再通过类别到bert语义编码的映射，可以得到相应的语义编码信息，可以避免通过bert神经网络运算的巨大开销。

3. 并发任务日志分离和异常检测部分：

目前许多系统日志都是由并发执行的任务异步打印的，即一段日志中包括了多个任务交叉打印的日志，无法直接从日志中提取出单一任务的执行路径。

过往的办法主要是通过相同的标识符将单一任务的执行日志关联起来，比如根据taskid、线程id、block\_id等，但对于缺乏这些标识符的日志段，则无法判断日志条目属于哪个任务。

本论文计划采用lstm网络和密度聚类的方法。主要是基于以下规律，属于同一任务的执行日志，出现在同一段日志中概率更大；对不同任务打印的日志，出现在同一段日志中概率要小得多，存在很大随机性。

借鉴自然语言处理中利用lstm模型进行单词预测的思想，将bert编码后的日志序列作为输入，通过预测下一条日志分布的概率，可以挖掘出日志中潜在的规律。最终以模型输出的概率作为依据，筛选出属于当前任务的执行日志。同时，该模型已经学习了常见的日志模式，因此当输入的日志出现异常时，该模型也可以检测出来。

毕业论文（设计）撰写提纲：

论文题目：基于计算框架日志的异常检测模型

一、引言

1. 介绍日志在软件开发和维护中的重要作用

2. 设计日志异常检测系统的目的和作用

- 实时生成海量日志，人工处理变得不切实际

- 借助日益发展的深度学习技术，实现对日志更全面地分析，提高异常识别准确率

- 无须人工介入，实时监测任务执行状态，实现系统的自我诊断，为故障自愈提供支持

3. 日志异常检测技术的研究和应用现状

二、相关研究和可提升部分

1. 相关研究介绍

- 建立自动机、流程图[4][6]等模型模拟任务执行过程，识别异常日志序列

- 借助密度聚类、层次聚类[10][11]等传统机器学习方法，发现异常日志段

- 结合深度学习[5][9]和自然语言处理[6]等较前沿技术，预测日志异常概率

2. 可提升部分

- 日志解析步骤的自动化和自适应程度不够高，需要模型中减少人工设计日志模板变量提取规则的工作

- 日志格式发生变化，导致模型失效，需要基于日志的语义信息进行异常检测的鲁棒模型。

三、日志异常检测模型设计

1. 日志解析器

2. 日志类别到bert语义编码映射缓存

3. LSTM序列异常检测模型

4. 实现及部署

四、结论

1. 模型的主要创新点

- 动态维护日志解析前缀树

- 语义聚类

2. 对研究结果作出初步评价

- 在真实软件应用中的效果表现

- 存在的问题和进一步工作的设想

参考文献

[1] Vaarandi R , Pihelgas M . LogCluster - A data clustering and pattern mining algorithm for event logs[C]// 2015 11th International Conference on Network and Service Management (CNSM). IEEE, 2015.

[2] Du M , Li F . Spell: Streaming Parsing of System Event Logs[C]// 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016.

[3] He P , Zhu J , Zheng Z , et al. Drain: An Online Log Parsing Approach with Fixed Depth Tree[C]// 2017 IEEE International Conference on Web Services (ICWS). IEEE, 2017.

[4] Yu X , Joshi P , Xu J , et al. CloudSeer: Workflow Monitoring of Cloud Infrastructures via Interleaved Logs[C]// the Twenty-First International Conference. 2016.

[5] Du M , Li F , Zheng G , et al. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning[C]// Acm Sigsac Conference on Computer & Communications Security. ACM, 2017.

[6] Pi A , Chen W , Wang S , et al. Semantic-aware Workflow Construction and Analysis for Distributed Data Analytics Systems[C]// the 28th International Symposium. 2019.

[7] Zhou P , Wang Y , Li Z , et al. LogChain: Cloud Workflow Reconstruction & Troubleshooting with Unstructured Logs[J]. Computer Networks, 2020, 175:107279.

[8] C. Lou, P. Huang, and S. Smith, “Understanding, detecting and localizing partial failures in large system software,” in NSDI’20: Proc. of the 17th USENIX Symposium on Networked Systems Design and Implementation, 2020, pp. 559–574.

[9] 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 International Symposium. 2018.

[10] Xu W , Huang L , Fox A , et al. Detecting Large-Scale System Problems by Mining Console Logs[C]// Acm Sigops Symposium on Operating Systems Principles. ACM, 2009:117.

[11] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R. Lyu, and Dongmei Zhang. 2018. Identifying impactful service system problems via log analysis. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (*ESEC/FSE 2018*). Association for Computing Machinery, New York, NY, USA, 60–70. DOI:<https://doi.org/10.1145/3236024.3236083>

He, Shilin & Lin, Qingwei & Lou, Jian-Guang & Zhang, Hongyu & Lyu, Michael & Zhang, Dongmei. (2018). Identifying impactful service system problems via log analysis. 60-70. 10.1145/3236024.3236083.