TraceAnomaly

论文题目	Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks	分类	故障检测、根因定位
发表位置	2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)	等级	2类贡献度/B类会议
发表时间	2020		
解决问题	检测轨迹异常 （1）底层微服务数量众多 （2）服务之间调用关系复杂 （3）响应时间和调用路径之间相互依赖 目前业界部署的异常检测方法仍然是基于人工规则的，所以提出了TraceAnomaly，可以自动学习轨迹的模式		
现状（趋势）不足	有哪几类研究，分别的不足是什么 单一数据进行检测 根据消耗时间判断是否异常		
研究内容	提出了。。。。具体每一步是怎么操作的+整体架构图 核心思想：利用机器学习在周期性离线训练过程中自动学习轨迹的正常模式；在在线异常检测中，一个异常分数小的轨迹被认为是异常的 训练方式：无监督学习 网络：后验贝叶斯网络 TraceAnomaly A：整体结构；提出了TraceAnomaly，自动学习轨迹的模式 （1）以一种可解释的方式统一轨迹的响应时间和调用路径-可解释性 每个轨迹视为一个训练样本，为每一个服务（不是微服务）训练一个模型 将轨迹的响应时间和调用路径编码为一个向量（STV） （2）无监督学习体系-将每个轨迹作为一个整体处理 B：从call Path中提取轨迹模式和时间模式 （s,call path）,call path代表从		

	start到当前微服务s的路径 按照发送请求的顺序排序 **C：STV，为每一个调用轨迹设定一个STV，包含了服务实例和时间信息** STV的每一个维度对应一个（s,call path），其value为响应时间 D：root cause HSTV（同质服务轨迹向量） 针对一条异常的轨迹，找到HSTV，找到异常的维度 将最长的调用链和下一调用的微服务作为根因 Anomaly detection design A：VAE B：Architecture 是为了干什么？统一形式？ C：Training 无监督，训练数据往往包含很少的异常 D：Anomaly detection 通过计算STV与模型间的对数似然性判断是否发生异常 通过学习一个正常的分布而不是一个手动设定的阈值		
创新点/贡献	STV、与产业界结合密切		
实验设置	用了哪些对比试验，从哪几个方面开展 一天中某一条调用路径出现次数足够多，第二天再训练是被视为new normals 最终输出不是一个可能根因的排名，因此只算了precision@1		
数据集	TrainTicket，40个高峰期和大约380000条正常的轨迹信息模拟一天的流量 测试集：异常注入方式，运行24小时，30356条正常的轨迹、2699条响应时间异常轨迹和2380条调用轨迹异常轨迹		
对比试验	对比算法，大概反应当前热门的方法 MonitorRank、RCSF、MEPFL、DeepLog、OmniAnomaly、Multimodal LSTM、CPD、CFD		
数据类型	轨迹信息（响应时间信息和调用路径信息）	定位粒度	将最长的调用链和下一调用的微服务作为根因
未来改进	不能区分异常到底是响应时间异常还是调用路径异常 文中提出：未来计划分析日志、配置、源代码等信息		

TraceAnomaly架构图：

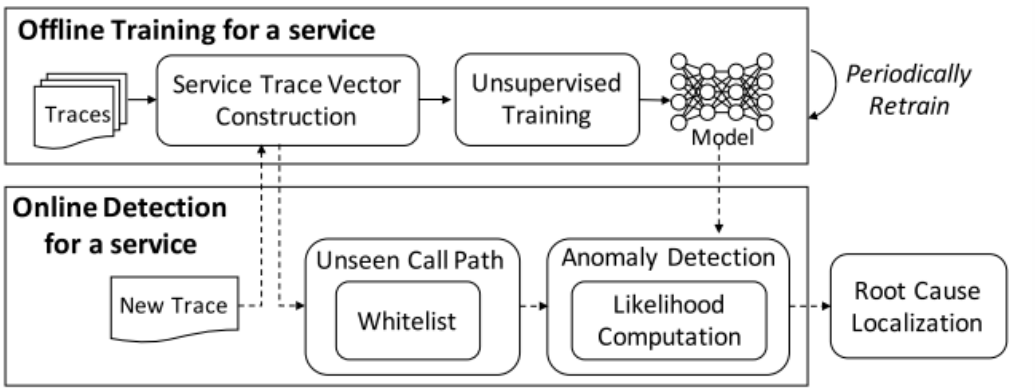

Fig. 3: The architecture of TraceAnomaly. Solid lines denote offline flow, and dashed lines denote online flow.

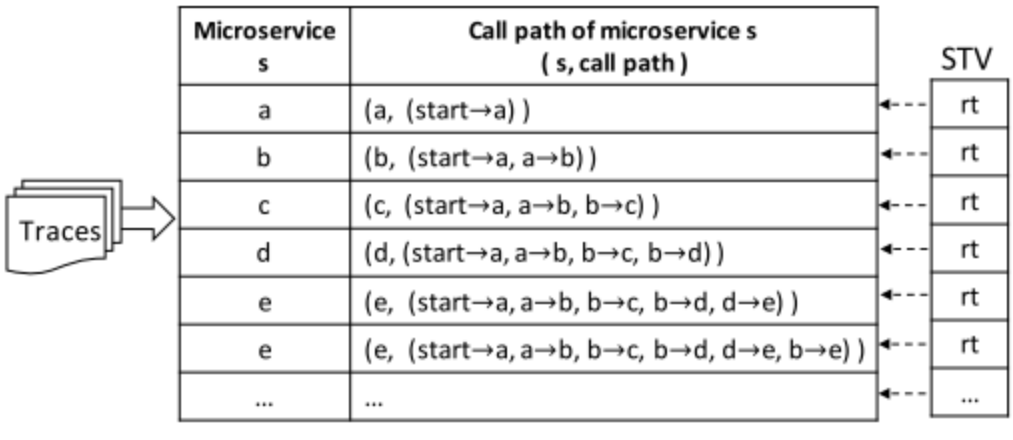

Fig. 5: Process of STV construction.

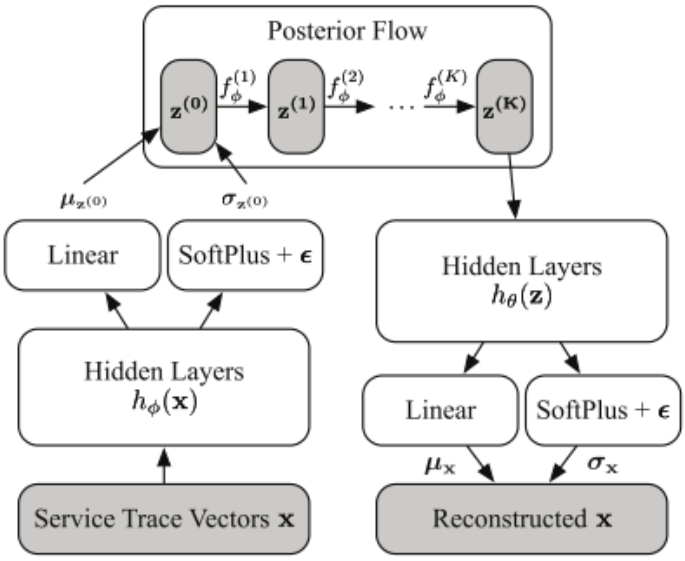

Fig. 8: The architecture of our model.