计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(7):85−92 [doi: 10.15888/j.cnki.csa.008555]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于服务的云平台应用程序监控分析系统①

罗理机, 唐　华, 周　昕, 张　涛

(湖北工业大学 信息技术中心, 武汉 430068)
通信作者: 罗理机, E-mail: mancst2000@163.com

摘　要: 针对云平台中对应用程序的性能监控方法存在全流程收集分析异常能力不足的问题, 提出一种基于云平台服务组件的应用程序异常检测和瓶颈识别系统 (AAD-PSC), 可对多层架构云平台上的应用程序提供可自定义指标值的监控分析能力. 系统首先在前端应用服务层收集云平台服务调用数据并与异常事件相关联; 然后为应用程序适配定制化的异常检测方法, 达到最优检测效果; 最后查明由非工作负载变化引起的性能异常, 并对其进行瓶颈识别. 实验结果表明, 监控系统可快速准确检测不同类别的异常事件并识别性能瓶颈, 能够满足云平台下对应用程序的性能监控需求.

关键词: 云平台; 应用程序; 服务组件; 瓶颈识别; 性能监控; 大数据

引用格式: 罗理机,唐华,周昕,张涛.基于服务的云平台应用程序监控分析系统.计算机系统应用,2022,31(7):85–92. http://www.c-s-a.org.cn/1003-3254/8555.html

Service-based Application Monitoring and Analysis System of Cloud Platform

LUO Li-Ji, TANG Hua, ZHOU Xin, ZHANG Tao

(Information Technology Center, Hubei University of Technology, Wuhan 430068, China)

Abstract: To address the problem that the methods of cloud platforms to monitor application performance have a poor ability to collect and analyze anomalies in the whole process, this study proposes an application anomaly detection and bottleneck identification system based on cloud platform service components (AAD-PSC) that can provide monitoring and analysis characterized by customizable indicator values of applications on a cloud platform with multi-tier architecture. For this purpose, this system collects service invocation data at the front-end application service layer and correlates them with anomaly events. Then, customized anomaly detection methods are determined for the applications to achieve the optimal detection results. Finally, performance anomalies caused by non-workload changes are identified, and bottleneck identification is conducted. Experimental results show that the proposed monitoring system is able to quickly and accurately detect different types of anomaly events and identify corresponding performance bottlenecks and meets the needs of a cloud platform in application performance monitoring.

Key words: cloud platform; application; service component; bottleneck identification; performance monitoring; big data

随着云计算技术的发展与市场规模的扩大, 越来越多的企业建立了云平台为应用程序提供统一的资源池和通用的平台组件服务. 由于云平台组件的通用性, 极大地便利了应用程序的规模部署[1].

对于应用程序开发者和云平台运维人员而言, 最为关注的是应用程序与云平台的兼容性, 他们希望实时检测应用程序的运行情况并及时发现异常和定位原因, 以便于进行针对性的调整[2]. 因此, 收集云平台上应

① 基金项目: 湖北省教育厅人文社科基金 (15Q065)
收稿时间: 2021-09-19; 修改时间: 2021-10-19, 2021-10-29; 采用时间: 2021-11-07; csa 在线出版时间: 2022-03-18

用程序的运行数据和分析异常原因的能力变得尤为重要.

云平台的架构包含 3 个基本层次: 基础设施层 (infrastructure layer)、平台层 (platform layer) 和应用层 (application layer)[3]. 基础设施层是最底层, 提供标准化的硬件资源池服务. 中间的平台层为应用程序的开发、测试、部署和运行提供相关的中间件和基础服务. 应用层是最高层, 提供种类繁多的应用程序集合. 云平台为了给用户提供通用便捷的组件服务, 相比于传统的服务器架构来说底层更为封闭, 并且多层架构的云平台也成为了主流, 以上两点因素都增加了对其中应用程序运行数据的收集和分析的难度[4]. 目前, 云平台上的应用程序监控主要面临以下问题: (1) 应用程序权限限制. 大多数云平台自身不具备应用程序级别的性能监控能力, 主要依赖于应用程序自身的检测异常机制和第三方检测软件的支持. 因此, 云平台运维人员必须赋予应用程序足够的权限, 这大幅度增加了云平台的运维成本和安全风险[5]. (2) 全流程收集分析异常能力不足. 云平台的组件服务对调用它的应用程序是不透明的, 这使得应用程序很难识别造成异常的根本原因. 并且, 随着大型企业云的层次结构变得愈发复杂[6], 每一层中多种交互组件服务的存在加大了准确判断性能异常原因的难度[7]. 提高快速准确查明异常原因的能力, 需要在云平台的不同层进行收集和关联异常信息, 建立全流程收集分析异常的能力. (3) 监控程序自定义能力不足. 目前, 一些大型的云平台已经具备对在其中运行的应用程序进行监控的能力, 但这些方法大多是对应用程序的进程状态信息做简单分析, 并传送给云平台运维人员做人工巡检. 然而, 为满足不同应用程序的监控需求, 需要对监控系统的信息采集速率、采样数据量和性能异常阈值等指标进行自定义设置, 而不能仅仅依赖于操作系统进程表中的数据. 对云平台用户和运维来说, 提供对应用程序多样化的监控需求是十分迫切的, 建立适配性强的云平台应用程序性能监控系统已经成为一项重要的研究课题[8].

为了解决以上问题, 本文设计了一种针对云平台应用程序的性能监控系统, 可以跨层收集应用程序的运行时数据, 快速准确的识别导致应用程序异常的云服务组件. 系统的主要特性有: (1) 具有完备的云平台应用程序监控运维能力. 能够全层次收集并分析云平台上应用程序的运行数据, 并利用平台即服务 (platform as a service, PaaS) 提供的管理服务接口 (management service interface, MSI) 实现自动运维. (2) 具有快速准确的异常检测能力. 通过对平台服务调用的持续时间分析, 识别异常调用事件, 为不同应用程序设置自定义的异常事件指标值, 保证了异常检测的准确性. (3) 提供云平台组件层面的瓶颈识别功能. 通过对应用程序调用组件的时间值进行异常偏离判断, 系统能够识别引发瓶颈的云平台服务组件. 综上所述, 本文构建了一个在私有云平台中基于服务组件的应用程序异常检测和瓶颈识别系统, 适用于多层架构云平台上的应用程序监控分析. 提供该能力有助于增强云平台运维人员对平台资源的管控能力, 降低应用程序开发人员对程序代码检测的压力, 进一步推动了云平台资源使用的合理化.

1 相关研究

现有关于云平台系统资源或应用程序运行性能监控与分析的研究, 主要途径是通过在不连续的时间间隔内收集重要的性能指标数据, 并进行分类、聚类、统计等异常检测分析. 在此基础上建立的异常监控分析系统框架, 一定程度上保证了云平台的性能、可靠性和服务质量.

文献 [9] 中提出了一种称为云平台自动异常检测 (autonomic anomaly detection, AAD) 的系统框架, 通过特征提取和数据变换整理性能指标数据, 并对异常事件进行聚类分析, 实现了异常事件的自动监控与分析. 在此基础上, 文献 [10] 引入互信息 (mutual information, MI) 参数来降低不同性能指标之间的冗余, 使性能指标数据的特征提取更为精确. 然后采用主成分分析法 (principal component analysis, PCA) 对提取的数据降维, 最后采用决策树分析性能指标从而识别异常事件. 饶翔等[11,12] 提取云平台的日志信息构造故障特征模型, 关联时间窗口内外的异常事件, 并采用改进的决策树训练基于日志信息的故障分类器, 实现了前后异常事件的关系判断. 但是, 该模型只针对的是系统的日志信息, 要求云平台对日志进行分类并使用标准语义记录, 故障分类器需要大量的日志信息进行分析, 所需的时间成本和存储空间较大. 贾统等[13] 提出通过提取日志模板变量, 构建日志异常检测模型, 然后使用机器学习、聚类等方法识别异常数据, 并对异常事件预警.

然而, 以上方法均利用获取云平台底层运行数据

(例如虚拟机日志信息) 或检测应用程序代码来实现对云平台资源的监控与分析, 虽取得了较多成果, 但已难以适应当前部署应用程序数量剧增和底层愈加封闭的主流大型云平台 (例如 PaaS) 的要求.

现有的云服务提供商提供的资源监测服务 (例如亚马逊云 CloudWatch[14], 阿里云 CloudMonitor[15] 以及华为云监控服务[16] 等) 通过在云平台底层添加数据采集器, 监控底层云资源, 然后在开发测试环境中执行并检测应用程序代码, 实现了对云资源和应用程序的有效监控. 但是, 开发测试环境中的检测结果并不一定总是能适用于实际的生产环境, 并且云平台的底层数据隐藏在对外提供的托管服务层之下, 接口不对外开放[17,18], 适用范围有限.

在以上研究基础上, 本文设计了一种基于云平台服务组件的应用程序异常检测和瓶颈识别系统. 通过跨层关联和分析服务调用请求数据, 识别导致应用程序异常的云服务组件, 实时准确地判断异常原因.

2 云平台异常检测和瓶颈识别系统模块设计

系统设计的主体思想是利用对应用程序调用云平台组件响应时间的实时监控, 为云平台的运行维护人员提供一个集成应用程序性能检测、平台组件异常检测和异常问题定位功能的云平台监控分析整体解决方案. 系统通过嵌入云平台的内置服务收集应用程序与云平台的组件交互数据, 并且对数据进行储存和分析. 获得的分析结果根据不同需求分别反馈给云平台管理员和程序开发人员. 系统通过监控云平台组件的响应时间, 对云平台的整体性能进行分析, 并且对单个应用程序的异常响应情况做出即时反应, 避免了传统云平台监控系统利用应用程序监听器对应用程序进行监控的弊端[19]. 同时, 对不同类型云组件响应时间的分析可以快速、准确地定位产生异常问题的原因.

云平台监控分析系统以应用程序调用服务组件的响应时间为指标识别和描述应用程序的异常响应, 并以此为基础搭建整体功能框架. 首先, 设计一种自动识别异常响应的检测器, 以建立应用程序异常与监控事件分析的触发机制; 然后, 提出了应用程序工作负载变化监控方法, 以识别应用程序异常响应是由于工作负载变化还是云平台组件瓶颈引发; 最后, 对于非应用程序工作负载变化引发的异常响应, 提供了云平台组件瓶颈识别功能.

2.1 系统框架

图 1 给出了私有云环境下云平台监控分析系统的整体架构, 分为数据收集、异常检测和异常分析 3 个模块. 数据收集模块用标识符对相关联的用户请求进行标记, 并负责在云平台的每一层上收集有关应用程序调用的信息; 异常检测模块把收集到的模块存入数据库中, 进行数据分析判断是否需要启动异常分析; 异常分析模块负责分析异常事件发生的原因并识别造成性能瓶颈异常的原因. 本节余下的部分将对云平台监控分析系统架构中的每一模块的功能进行详细说明.

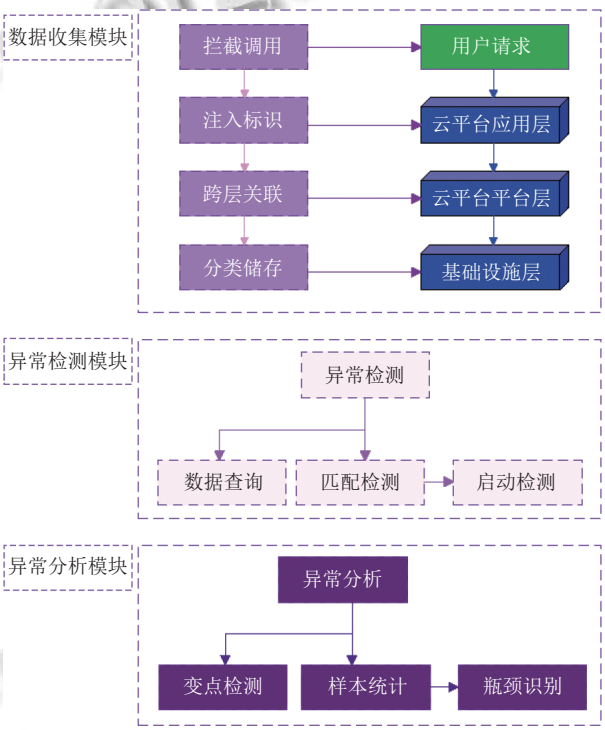

图 1 云平台监控分析系统架构

2.2 前端数据收集模块

云平台的多层架构使用户能够以按需、易扩展的方式获得所需的资源, 同时也提高了云平台的安全性. 在云平台的每一层只能收集与其本层状态变化相关的数据, 且层次之间的封装结构使其中一层无法监测到其他层的状态更改. 因为应用程序调用组件的行为跨越云平台的多个层次, 所以为了对云平台的系统资源进行整体监控, 系统必须从每个层次收集数据, 并把与同一调用行为相关联的数据进行跨层组合.

为了解决这个问题, 云平台监控分析系统在前端设置了数据收集模块. 数据收集模块通过在服务接口

的入口拦截内核调用来收集应用程序的组件调用请求数据. 具体的做法是首先在每一个 HTTP 请求的消息头中添加唯一的数据标识符, 使云平台的每一层都能够识别请求的关联对象. 然后, 把数据收集工具部署到平台内, 抓取组件调用请求并根据消息头中的数据标识符关联与之相关的应用程序和平台组件. 最后, 数据收集模块组合调用请求并记录与之相关的资源变化信息, 再根据需要对不同的信息分类保存. 组合后的信息记录使系统能够跨层监控应用程序对平台组件的请求调用, 以及与之相关联的资源变化信息, 而不需要考虑云平台各层之间的耦合关系. 同时, 不同调用请求对应的数据收集工具相对独立, 互相之间无须发送关联消息, 因此也提高了模块的扩展性.

2.3 异常检测模块

系统收集到组合信息之后, 以应用程序 ID、时间为索引将其存入数据库中, 便于长期存储和查询. 异常检测模块读取这些数据并进行定时分析. 模块对每个应用程序配置定制化的异常检测方法, 不同应用程序可以对应不同的检测方法. 不同种类应用程序调用的平台组件服务类型、频率不同, 因此其所适配的异常检测方法也不同. 为了解决应用程序和对应检测方法的适配问题, 模块还支持对同一应用程序进行并发检测, 从而比较不同检测方法的准确性, 找出与应用程序匹配度最高的检测方法进行配置.

传统方式监控云平台性能往往是直接在云平台基础设施层部署采集插件, 通过分析系统吞吐量、储存设备读写速度、CPU 时钟频率等指标达到监控云主机性能的目的[11−13], 但这种方式已不能适用于目前主流的封闭底层资源监控接口的大型云平台. 异常检测模块通过调用平台层提供的服务响应接口分析云平台对应用请求的回应速度, 来衡量云平台对请求的响应快慢, 系统无须获取云平台底层的资源监控权限即可达到监控云平台性能的目的, 提高了系统的通用性和安全性. 具体做法是当应用程序启动时, 异常检测模块同时启动检测过程, 该过程定期测量目标应用程序的响应时间T_{resp}和性能异常偏离率P_{pad}两个指标.

应用程序的响应时间用来衡量程序的性能, 响应时间越长, 性能越低, 可以用以下公式表示:

$$T_{\text{resp}} = \sum_{i=1}^{n} (t_{ri} - t_{si})/n \tag{1}$$

其中, t_{ri}和t_{si}分别为第 i 个组件调用的服务回应时间和触发时间, n 为统计持续时间内检测到的组件调用总个数.

为了衡量云组件服务对调用的响应快慢, 我们还定义了性能异常偏离率指标, 用来反映应用程序响应时间落在正常范围以外的概率. 当第 i 个组件调用的响应时间高于应用程序响应时间T_{resp}与统计持续时间内组件调用响应时间的标准差σ_{resp}之和时, 组件调用响应时间被判定为偏离正常范围. 因此, 性能异常偏离率P_{pad}可通过统计持续时间内偏离正常范围的组件调用数目N_{dev}和组件调用的总数N_{total}比例计算得到:

$$P_{\text{pad}} = \frac{N_{\text{dev}}}{N_{\text{total}}} \tag{2}$$

当P_{pad}大于 5% 时, 表示在统计持续时间内有 5% 以上的组件调用响应时间偏离了正常范围, 则此时异常检测模块认为应用程序性能发生异常.

为减少对应用程序的性能损耗, 异常检测模块间隔启动对响应时间T_{resp}和性能异常偏离率P_{pad}两项指标的计算和分析, 当P_{pad}偏离正常范围时, 就触发异常事件.

异常检测模块也能够对应用程序匹配的检测方法进行分析间隔时间和统计持续时间两个参数的配置, 其中分析间隔时间即为检测方法定时启动分析的间隔时间, 这个参数在系统测试中被设置为 15 s. 统计持续时间指检测方法提取和统计异常事件发生的时间跨度, 通常时间跨度越长则内存中存放的数据量越大, 为限制系统运行时占用的内存大小, 统计持续时间参数在系统测试中被设置为 30 min.

2.4 异常分析模块

异常分析模块对异常检测模块发出的异常事件进行分析, 首先分析异常事件的发生原因, 如果是由工作负载大小变化引起, 则不做处理, 如果是则进行瓶颈识别.

系统使用变点检测算法 (change point detection, CPD)[20] 对工作负载的变化进行检测, 算法生成一个负载变化的变点列表来表示一段时间内工作负载的重大变化. 系统利用 CPD 对负载变化点的历史数据批量分析, 实现对平台负载变化的持久跟踪, 能够准确识别平稳的大数据量请求 (非峰值请求) 造成服务器性能异常的情况.

对于因非工作负载变化而引起的性能异常, 系统对其进行瓶颈识别, 找出与之关联最大的平台组件. 系

统首先记录每一次样本程序调用平台组件的时间长度组成数组 $C[n]$ (n 为调用平台组件次数), 然后使用离群值统计法[21] 判断捕获的数据是否异常, 此方法适用于在定时采集时间样本的数据量范围内对样本的偏离情况进行测算, 计算出样本的偏离值. 系统选择与偏离值最大的时间样本相关联的组件作为瓶颈识别的结果, 此组件即为引发瓶颈的云平台服务组件.

3 云平台应用程序监控分析系统测试

系统测试在 OpenStack 私有云环境下进行, 操作系统版本为 CentOS 7.4, 测试节点配置为 8 核 CPU (时钟频率 2.6 GHz), 16 GB 内存, 200 GB 硬盘.

本次测试使用的样本应用程序是一个利用 PHP 编写的 Web 应用程序, 功能是实现单点登录并发布图文信息. 程序调用统一身份认证 (CAS) 组件来实现身份验证, 并调用云平台的数据存储组件来保存发布的信息. 用户的每次发布请求都会调用两次云平台服务组件, 一次用于检查用户的登录状态, 另一次用于在数据库相应表中写入数据.

为了全面评测云平台应用程序监控分析系统各个模块的功能, 依次测试了系统检测云平台异常事件的能力、分析服务器负载变化的能力以及识别性能瓶颈原因的能力.

3.1 异常事件检测

在测试系统的异常事件检测能力之前, 需要先确定样本程序正常运行时的响应时间. 因此, 样本程序首先在无任何外部程序干扰的情况下运行了 10 次, 每次持续 1 h, 得到正常情况下样本程序响应时间的平均值 t_{avg} 和标准差 σ_t. 响应时间的阈值 t_{max} 由以下公式获得:

$$t_{max} = t_{avg} + 3\sigma_t \tag{3}$$

利用操作系统的 Page Cache 策略, 写入 8 线程单次 4 KB 的 Buffer, 可以模拟云平台数据存储组件响应缓慢造成的性能异常. 设置此性能异常事件 30 分钟触发一次, 并在 5 s 时间内调用平台的数据存储组件使响应时间升高到 t_{max} 以上, 响应时间超过 t_{max} 则视为发生一次异常事件.

为验证第 2.3 节所述系统异常检测模块的性能, 将分析间隔时间设置为 15 s, 统计持续时间为 30 min, 样本程序的运行时间一共为 5 h. 触发性能异常事件与系统检测到异常事件的时间点如表 1 所示.

综合实验结果, 从性能异常事件发生开始到系统检测到异常事件, 平均用时 268 s, 最长的时间间隔为 299 s. 由于检测用时与分析间隔时间密切相关, 因此可以通过修改检测工具的分析间隔时间进一步对检测性能进行调优.

表 1 触发和检测到异常的间隔时间统计

序号	触发异常 (hh:mm:ss)	检测异常 (hh:mm:ss)	间隔 (s)
1	08:40:00	08:44:05	245
2	09:10:00	09:14:45	285
3	09:40:00	09:44:47	287
4	10:10:00	10:13:55	235
5	10:40:00	10:44:58	298
6	11:10:00	11:14:08	248
7	11:40:00	11:44:01	241
8	12:10:00	12:14:32	272
9	12:40:00	12:44:31	271
10	13:10:00	13:14:59	299

3.2 分析负载变化

在对系统的分析负载变化能力进行测试的过程中, 样本程序同样运行 5 个小时. 程序运行期间, 更改高频读取请求的间隔时间为随机, 以达到模拟平台负载变化的目的. 在随机高频读取请求发生期间, 请求的频率在每分钟 450–650 次之间, 这远高于平台正常运行情况下每分钟 120–150 次的请求频率, 这些读取请求大部分都无法命中数据库缓冲区, 因此会引发平台服务器缓存负载的急剧上升.

如图 2 所示, 每个负载高峰发生之后, 系统几乎都能准确的检测到异常事件发生, 纵向箭头指示检测到异常事件发生并触发瓶颈识别模块的时刻. 唯一一次例外是发生在两次高频请求短时间内集聚发生, 这是由于系统在检测到第一次异常事件之后对数据进行回收处理, 进入等待状态以收集更多数据. 在紧接着另一次发生负载高峰情况时, 系统认为异常事件的发生是由于工作负载变化引起的, 因此没有第二次触发瓶颈识别模块工作.

3.3 性能瓶颈原因识别

在第 3.2 节的实验中, 系统分析所有检测到的异常都是由云平台数据存储组件响应超时引起的. 实验结果符合实际情况, 因为实验中的每一次异常事件都是由大数据量的 Buffer 写入造成数据存储组件无法及时响应.

为了测试系统对其他原因引起性能瓶颈的识别能力, 接下来本实验修改用户验证组件的逻辑, 使应用程

序向用户验证组件发送请求时, 组件延迟返回消息, 延迟时间大于 t_{max} 时, 则产生异常事件. 在实验中设置每 30 min 触发一次验证异常, 实验共持续 5 h.

如图 3 所示, 系统在每一次的逻辑异常发生之后都检测到了性能异常事件, 其中 7 次异常被系统识别为用户验证组件超时引发的异常, 另外还有两次异常被识别为由数据储存组件超时引发. 通过手动检查云平台系统日志发现, 例外的两次异常都是由于数据库查询超时引起用户验证组件异常, 系统将异常归结为第一个出现异常的数据储存组件是准确的.

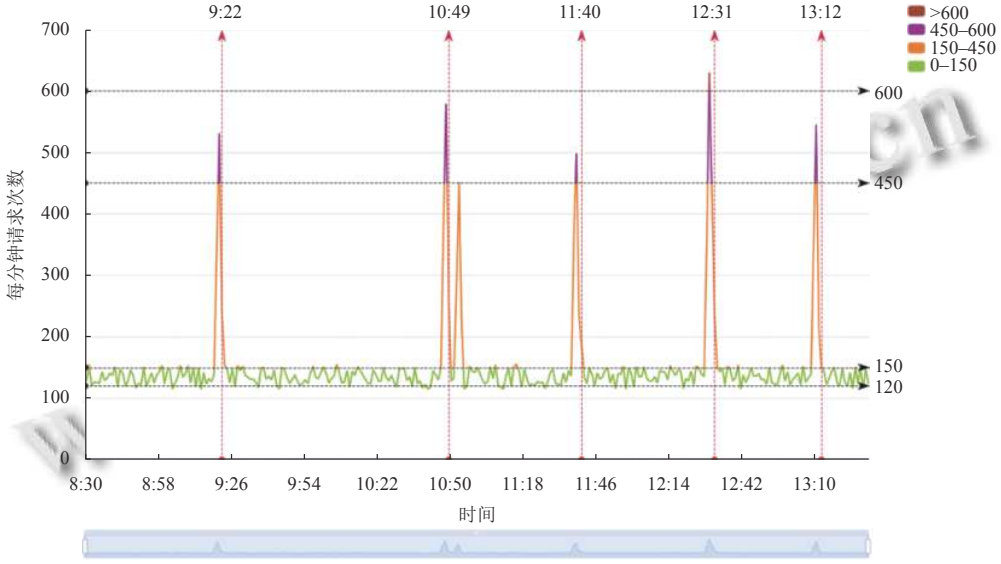

图 2 服务器负载与异常检测能力测试

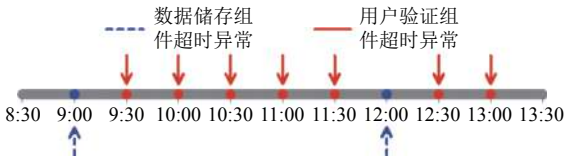

图 3 识别异常事件原因能力测试

为了在更大的样本情况下测试多种异常组合发生时瓶颈识别的能力, 系统在私有云平台上运行了 1 个星期. 通过分析这段时间内系统检出的 125 个异常, 发现除了 7 个异常事件外, 大多数情况下系统都能准确识别第一个引起异常的平台组件, 平台的瓶颈识别准确率达到了 94.4%.

3.4 实验结果分析

在异常检测和瓶颈识别实验的基础上, 测试云平台应用程序监控分析系统的通用性和准确性. 将系统的瓶颈识别准确率、是否需要平台底层监控权限、适用的云平台类型这 3 项指标作为考察系统通用性和准确性的因素, 对比算法是 ATAD[22], vPerfGuard[23], BAD[24], ANN[25]. 为了更好地说明异常识别的性能, 测试时采用了相同的引发云平台性能异常机制. 利用统一身份认证 (CAS) 组件仿真大量的随机读取请求发送至被测平台, 持续时间均为 1 个星期, 结果如表 2 所示, 可以看出 vPerfGuard 的识别准确率最高, 而 AAD-PSC 次之, 然后是 BAD, 其余算法的准确率较低. vPerfGuard 的识别准确率高的原因是直接在云平台底层获取硬件资源和工作负载指标, 避免了性能异常集聚发生时的误判, 但是 vPerfGuard 需要云平台提供底层的资源监控权限, 无法适用于 PaaS 类型的云平台, 通用性不及其他方法. BAD 构建自适应的监督学习任务, 识别准确率依赖大数据集的收集和模型拟合, AAD-PSC 是通过直接监测云平台的服务性能指标判断云平台的当前性能, 所以 AAD-PSC 比 BAD 的判断准确率更高.

表 2 触发和检测到异常的间隔时间统计

监控系统	识别准确率 (%)	是否需要底层权限	是否适用于PaaS
ATAD	85.6	否	是
vPerfGuard	100	是	否
BAD	93.3	否	是
ANN	90.8	否	是
AAD-PSC	94.4	否	是

4 小结

监控和分析云平台上托管的应用程序运行情况是云平台运维中的一项核心需求, 应用程序开发和云平台运维人员希望能够具备监控云应用程序的性能异常, 并分析其产生原因的能力. 但随着云计算技术的发展, 云平台的规模急速扩大, 多层架构日益复杂, 传统云平台应用程序监控系统存在依赖云平台赋权、异常溯源不准确、分析方式单一等问题, 已无法满足现有大规模云平台的需求. 本文设计并实现了一种面向云平台应用程序的性能监控系统. 通过标识符跨层关联应用程序请求与相关事件, 简化了信息收集过程. 采用了基于变点检测的瓶颈识别方法, 实现了快速、准确的异常事件溯源. 实验结果表明, 监控系统可以在异常事件发生后的 5 min 内准确检测不同类别的异常事件并识别性能瓶颈, 能够满足云平台下对应用程序运行性能的监控需求.

增强出版

本文附有基于服务的云平台应用程序监控分析系统演示视频, 可点击视频链接或手机扫描二维码观看.

参考文献

1 Vaquero LM, Rodero-Merino L, Caceres J, *et al*. A break in the clouds: Towards a cloud definition. ACM SIGCOMM Computer Communication Review, 2008, 39(1): 50–55. [doi: 10.1145/1496091.1496100]

2 Feller E, Rilling L, Morin C. Snooze: A scalable and autonomic virtual machine management framework for private clouds. 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Ottawa: IEEE, 2012. 482–489. [doi: 10.1109/CCGrid.2012.71]

3 Birje M N, Bulla C. Cloud monitoring system: Basics, phases and challenges. International Journal of Recent Technology and Engineering, 2019, 8(3): 4732–4746.

4 Pourmajidi W, Steinbacher J, Erwin T, *et al*. On challenges of cloud monitoring. arXiv: 1806.05914, 2018.

5 Rak M, Venticinque S, Máhr T, *et al*. Cloud application monitoring: The mOSAIC approach. 2011 IEEE 3rd International Conference on Cloud Computing Technology and Science. Athens: IEEE, 2011. 758–763. [doi: 10.1109/CloudCom.2011.117]

6 Fatema K, Emeakaroha VC, Healy PD, *et al*. A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. Journal of Parallel and Distributed Computing, 2014, 74(10): 2918–2933. [doi: 10.1016/j.jpdc.2014.06.007]

7 陈皓, 许源佳, 王焘, 等. 基于相似度匹配的微服务故障诊断方法. 计算机系统应用, 2021, 30(5): 1–11. [doi: 10.15888/j.cnki.csa.007888]

8 Prasad VK, Bhavsar M. Efficient resource monitoring and prediction techniques in an IaaS level of cloud computing: Survey. 1st International Conference on Future Internet Technologies and Trends. Surat: Springer, 2017. 47–55. [doi: 10.1007/978-3-319-73712-6_5]

9 Smith D, Guan Q, Fu S. An anomaly detection framework for autonomic management of compute cloud systems. 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops. Seoul: IEEE, 2010. 376–381. [doi: 10.1109/COMPSACW.2010.72]

10 Fu S. Performance metric selection for autonomic anomaly detection on cloud computing systems. 2011 IEEE Global Telecommunications Conference. Houston: IEEE, 2011. 1–5. [doi: 10.1109/GLOCOM.2011.6134532]

11 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. 计算机学报, 2012, 35(5): 856–870. [doi: 10.3724/SP.J.1016.2012.00856]

12 饶翔. 基于日志的大规模分布式软件系统可信保障技术研究 [博士学位论文]. 长沙: 国防科学技术大学, 2011.

13 贾统, 李影, 吴中海. 基于日志数据的分布式软件系统故障诊断综述. 软件学报, 2020, 31(7): 1997–2018. [doi: 10.13328/j.cnki.jos.006045]

14 Amazon CloudWatch. http://aws.amazon.com. [2021-11-09].

15 Aliyun CloudMonitor. https://www.aliyun.com. [2021-11-09].

16 Huawei CloudMonitoring Service. https://www.huaweicloud.com. [2021-11-09].

17 张倩, 何汉东. 基于私有云平台的云主机资源监控方案. 计算机系统应用, 2017, 26(8): 71–76. [doi: 10.15888/j.cnki.csa.005875]

18 Maenhaut PJ, Volckaert B, Ongenae V, *et al*. Resource management in a containerized cloud: Status and challenges. Journal of Network and Systems Management, 2020, 28(2): 197–246. [doi: 10.1007/s10922-019-09504-0]

19 Xiao P. Improving the scalability of cloud monitoring service by low communication overhead mechanisms. International Journal of Networking and Virtual Organisations, 2020, 23(1): 67–81. [doi: 10.1504/IJNVO.2020.107957]

20 Daly D, Brown W, Ingo H, *et al*. The use of change point detection to identify software performance regressions in a continuous integration system. Proceedings of the

ACM/SPEC International Conference on Performance Engineering. Edmonton: ACM, 2020. 67–75. [doi: 10.1145/3358960.3375791]

21 国家标准化管理委员会. GB/T 4883-2008 数据的统计处理和解释 正态样本离群值的判断和处理. 北京: 中国标准出版社, 2009.

22 Dani MC, Jollois FX, Nadif M, *et al*. Adaptive threshold for anomaly detection using time series segmentation. 22nd International Conference on Neural Information Processing. Istanbul: Springer, 2015. 82–89. [doi: 10.1007/978-3-319-26555-1_10]

23 Xiong PC, Pu C, Zhu XY, *et al*. vPerfGuard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments. Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering. Prague: ACM, 2013. 271–282. [doi: 10.1145/2479871.2479909]

24 Ibidunmoye O, Rezaie AR, Elmroth E. Adaptive anomaly detection in performance metric streams. IEEE Transactions on Network and Service Management, 2018, 15(1): 217–231. [doi: 10.1109/TNSM.2017.2750906]

25 Alnafessah A, Casale G. Artificial neural networks based techniques for anomaly detection in Apache Spark. Cluster Computing, 2020, 23(2): 1345–1360. [doi: 10.1007/s10586-019-02998-y]

(校对责编: 牛欣悦)