拟/物理节点上,运行环境十分复杂.分布式软件系统内部的缺陷与外部的复杂环境导致分布式软件系统频繁地发生故障.这些故障可能由资源竞争、配置错误、软件缺陷、硬件失效等原因导致,在系统运行时则表现为分布式软件系统中的服务请求执行过程出现异常甚至执行失败.分布式软件系统中的故障诊断主要回答 3 个问题:(1) 分布式软件系统中是否出现故障,即异常检测问题;(2) 故障出现在分布式软件系统的什么位置,即故障定位问题;(3) 分布式软件系统中出现故障的原因,即故障根因诊断问题.

请求执行路径精准刻画了分布式软件系统因处理请求产生的一系列行为,因此,基于请求执行路径,可以检测出更细粒度的请求执行过程的异常,继而检测出使用传统的基于监控和基于日志的故障诊断方法检测不到的系统的故障.当系统发生故障时,根据请求执行路径精准地定位故障的位置,诊断故障的根因.基于请求执行路径的故障诊断主要解决 3 个问题:(1) 异常检测,检测分布式软件系统中的请求是否正常执行;(2) 故障定位,在分布式软件系统发生故障之后,找到引起故障的事件/日志/代码片段/组件等;(3) 根因诊断,对故障的根因进行解释.

在异常检测方面,文献[73]在文献[18]的基础上通过侵入 JVM 实现了其分布式追踪系统,利用分布式追踪系统构建的请求执行路径(组件之间的调用树)对所有的请求执行路径构建一个概率上下文无关文法(probabilistic context free grammar).在行为异常检测阶段,计算待检测的请求执行路径在当前概率上下文无关文法模型中生成的概率,当概率小于某阈值时,认为请求执行过程出现了行为异常,系统发生了故障.我们在 TPC-W WIPSo 应用中通过人工注入故障,应用并对比了文献[73]提出的基于概率上下文无关文法的模型检测路径结构异常和基于 HTTP 错误代码检测系统故障两种方法,进而发现在请求执行路径中应用其模型检测系统故障的方法的准确率达到 90%,远远高于基于 HTTP 的错误代码检测系统故障的准确率.文献[56]中请求执行路径为日志事件的序列,通过提取日志的模板,将请求执行路径转化成一个日志模板的有向图(message flow graph,简称 MFG).在行为异常检测阶段,采用其所提出的图的编辑距离计算算法,计算由待检测请求执行路径构建的 MFG 与由正常请求执行路径构建的 MFG 的相似度,当相似度小于某阈值时,认为待检测的请求执行路径中存在行为异常,系统发生了故障.文献[51]从日志消息中构建请求的执行路径,并对每一种服务请求的正常的执行路径构建一个自动机(automaton),在在线检测阶段,将待检测的请求执行路径输入构建的自动机检测请求执行路径是否出现异常.文献[43]与文献[51]类似,不过其从正常的请求执行路径(日志模板序列)中构建一个有限状态自动机(finite state automaton),用于异常检测.文献[74]提出了一种基于贝叶斯方法从请求执行路径中检测其中是否存在慢性故障的方法.

在故障定位方面,文献[38]从分布式软件系统的日志中推断生成请求执行路径,将请求执行路径表示为日志模板的序列,然后利用他们提出的一种日志对齐算法,对比存在故障的请求执行路径和正常的请求执行路径,定位出导致故障的日志模板及其在代码中的位置.文献[56]从日志中生成请求执行路径的碎片,为每一个任务生成一个 MFG(message flow graph)作为请求的执行路径,然后使用基于图编辑距离的算法进行离群点(与其他任务的 MFG 不同的任务)分析,并提取出离群点中特有的日志模板序列,将故障定位至提取出的少量的日志模板中.而文献[75]则利用从日志中推断出的正常的以日志模板为节点的请求执行路径,检测出待检测的请求的执行路径的异常节点,从而发现故障的日志模板,并通过扫描程序的源代码,建立日志模板与日志打印语句之间的映射关系,从而将故障定位至源代码中产生故障日志的日志打印语句.文献[18]则使用层级聚类算法 UPGMA 检测异常的请求执行路径,并通过雅卡尔相似系数(Jaccard similarity coefficient)将故障定位至具体的组件.

在根因诊断方面,文献[30]基于文献[31]提出的分布式追踪技术,在分布式软件系统中进行大量的故障注入实验并收集故障下的请求执行路径,建立故障与请求执行路径之间的关联关系.在发生故障时将待检测的请求执行路径与数据库中的请求执行路径进行相似度匹配,选择相似度最高的几个有故障的请求执行路径,进而对故障根因做出诊断.

分布式追踪技术已在故障诊断领域得到了大量应用,分布式追踪技术产生的请求执行路径能够帮助运维人员更准确地检测出分布式软件系统中的异常行为[73],提高故障定位的粒度[38],并自动诊断故障的根因[30].基于请求执行路径的异常检测通常从正常的请求执行路径中构建图模型(上下文无关文法、有限状态机

等),将待检测的请求执行路径输入构建的图模型中进行检验.基于请求执行路径的故障定位在异常检测方法的基础上,将图上的每个顶点赋予一定的位置属性(如组件、日志打印代码等).当前,基于请求执行路径的根因诊断的相关研究工作不多,主要是通过建立故障与请求执行路径之间的关联关系,当发现待检测的请求执行路径与已知的存在故障的请求执行路径相匹配时,自动报告故障的根因.

3.3 基于请求执行路径的性能分析

分布式软件系统的性能问题是近年来倍受关注的问题,性能问题往往会影响用户的体验,导致资源浪费与客户流失.但是,由于分布式软件系统往往分布在大量的节点上,组件之间的交互比较复杂,导致对分布式软件系统进行性能分析成为一个难题.传统的性能分析方法通常采用几类资源的监控数据(如 CPU、内存的利用率、响应时间、吞吐量等)检测分布式软件系统的性能问题.但是由于监控数据通常是以单个节点/进程/线程为中心,无法精准地判断涉及大量节点与进程的分布式软件系统的性能问题.

通过将分布式追踪技术产生的请求执行路径与相关的分布式系统的性能指标相结合,如响应延迟、CPU 消耗等,可以在分布式软件系统中进行精准的性能分析[76].基于请求执行路径的性能分析主要需解决两个问题:(1) 请求响应延迟异常检测;(2) 资源消耗异常检测.前者使用响应时间正常的请求作为参照,基于这类请求在正常执行情况下的响应延迟,细粒度地检测请求的响应时间是否出现了不符合预期响应时间的状况,并在请求执行路径中定位引起响应延迟的具体位置;后者结合请求执行路径与分布式软件系统的资源消耗信息(CPU/Memory/Disk 等),构建某类请求在特定资源上的资源消耗模式,当该请求的某类资源消耗与所构建的资源消耗模式不符时,则认为该请求出现了资源消耗异常,并诊断异常资源消耗的原因.

在请求响应延迟异常检测方面,文献[15]提出了一种基于请求执行路径检测请求响应延迟异常的方法.首先,根据请求类型对所有的请求执行路径进行聚类.基于假设:含有请求响应延迟异常的一类请求,其响应延迟的分布会比较分散,文献[15]使用变异系数 CV(coefficient of variation)来衡量每一类请求的响应延迟分布的分散程度,当 CV 大于某阈值时,则认为此类请求中出现了延迟异常.文献[13]使用 KS-检验(Kolmogorov-Smirnov test)检测请求的响应延迟是否符合正常的请求执行的响应延迟的分布.文献[55]基于 Dapper 生成的请求执行路径,根据每次 RPC 调用所耗时间与资源消耗推断本次请求的执行时间,当请求的执行时间与推断的请求执行时间的偏差超过一定阈值时,检测出性能异常并判断性能异常是由请求执行结构变化还是子 RPC 性能异常导致.文献[77]则提出了一种将时间序列化的监控数据与请求的事件序列相关联,从而帮助运维人员诊断性能故障根因的方法.文献[11,27,78,79]同样基于分布式追踪产生的请求执行路径,建立请求正常执行下的请求响应延迟分布的模型,并根据构建的模型进行请求响应延迟异常检测.

在资源消耗异常检测方面,文献[14]将请求执行路径与每个节点上的资源消耗通过时间相关联并进行可视化,辅助运维人员查找故障根因.文献[24]则可以监控每个请求在执行过程中实时的资源消耗,通过可视化发现资源的异常消耗.

在分布式软件系统的性能分析领域,请求执行路径精准刻画分布式软件系统的执行逻辑的特点,使得准确地分析分布式软件系统的性能故障与瓶颈成为可能.对于分布式软件系统,当前的相关研究工作利用请求执行路径作为框架,将请求执行的时间与资源消耗信息嵌入请求执行路径,使用可视化或者统计检验方法来进行分布式软件系统的性能分析,相信未来会有更多研究工作提出结合请求执行路径与时间和资源消耗信息的性能分析方法.

4 分布式追踪技术研究展望

分布式追踪技术是分布式软件系统领域一项重要的技术,不仅在学术界与工业界出现了大量的分布式追踪技术研究工作与分布式追踪系统,也出现了大量的研究工作探索将分布式追踪技术应用到不同的运维场景中,如理解分布式软件系统的架构与行为[65],精准诊断分布式软件系统中的故障[73],检测分布式软件系统中的性能问题[76].通过本文对具体的分布式追踪技术与系统的分析可以看出,已有的分布式追踪技术的研究工作主要集中在以下两个方面:(1) 系统日志及内核事件日志之间的因果关系推断问题,相关研究工作提出不同的推

断方法判断系统日志之间以及内核事件日志之间的因果关系;(2) 基于代码侵入的追踪数据获取问题,相关研究工作主要研究在不同层次软件堆栈中如何设置追踪点并设计请求上下文的传播机制,但在一些问题上的研究仍处于初级阶段,需要研究者进一步的探索与研究,主要包括:(1) 数据读写依赖问题;(2) 通用性问题;(3) 评价问题.因此,本节将从分布式追踪技术中的数据读写依赖问题、通用性问题、评价问题这 3 个方面分析当前分布式追踪技术的不足并探讨未来的研究方向.

4.1 分布式追踪技术中的数据读写依赖问题

随着分布式软件系统的不断发展,为提高分布式软件系统的性能与吞吐量,降低分布式软件系统的耦合度,消息队列尤其是消息中间件在大型分布式软件系统中得到了大量的应用,典型的消息中间件如 RabbitMQ、ActiveMQ、Kafka.在分布式软件系统中,不同的组件将消息发送到消息中间件的消息队列中,其他的组件从消息中间件中接收所订阅的消息并进行处理.消息队列也不仅仅存在于消息中间件中,很多分布式系统自身采用了消息队列来提高性能与吞吐量,如在 Hadoop 的 RPC 通信中,RPC 客户端发送的消息首先会被 RPC 服务端置入消息队列,由 Handler 读取队列中的 RPC 请求并进行真正的方法调用且将调用返回的结果置入另一个消息队列中,由 Responser 将 RPC 调用的结果返回给 RPC 客户端.因此,在分布式软件系统中一个请求的完整处理过程可能会多次经过消息队列.如果分布式追踪技术无法准确捕获对同一消息的读写产生的事件之间的因果关系,则分布式追踪技术将会对一个请求产生多个请求执行路径碎片,从而无法刻画请求在分布式软件系统中完整、准确的处理过程,称其为数据读写依赖问题.随着消息队列在分布式软件系统中越来越广泛的应用,数据读写依赖问题使得分布式追踪技术面临着新的挑战.

数据读写依赖问题已经引起了研究者的关注,Chanda[60]等人提出了一种判断对共享的内存区域进行读写的事件之间因果关系的判断方法,Zhang[80]等人在移动平台中提出了根据对同一事件的读写判定事件之间因果关系判断的方法,Wu[59]等人则基于对遵循一定标准的消息中间件的消息的解析,提取每个消息读取与写入事件中的消息 ID,进而判断同一消息的读写事件的因果关系.但已有的研究工作仅能在特定应用场景中判断特定类型的消息读写之间的因果关系,分布式追踪技术中面临的数据读写依赖问题并没有得到很好的解决.如何处理分布式软件系统中复杂的数据读写依赖问题,进而生成完整、准确的请求执行路径,是分布式追踪技术所面临的一个重要的问题,也将是分布式追踪技术的重要研究方向之一.

4.2 分布式追踪技术的通用性问题

当前,大量的分布式追踪技术仅能追踪特定分布式软件系统中的服务请求[3,81],或者处理基于特定运行时环境[18,19,22,23]、特定系统框架[63,68]及特定企业公共链接库[9,24]的分布式软件系统.而随着开源软件的发展及微服务架构的兴起,分布式软件系统的结构变得更加复杂,一个分布式软件系统可能会由多个不同团队使用不同语言开发的不同的服务与组件组成,导致对一个请求在分布式软件系统的追踪可能需要多个不同分布式追踪技术的共同协作才能完整追踪请求的处理过程[24],但不同分布式追踪技术产生的请求执行路径在粒度、数据结构、因果关系、路径表示方面都有巨大的不同,给分布式追踪技术的应用带来了巨大的挑战.

针对通用型的分布式追踪系统,有的研究者从系统日志入手[8,51],从系统日志中使用统计方法构建请求的执行路径;有的研究者从操作系统层级利用分布式软件系统的系统调用[1]或者产生的内核事件[46]提出分布式追踪技术.从系统日志中推断请求执行路径面临着请求路径的准确性不高的问题,而从操作系统层级构建分布式追踪系统则面临着缺乏上层应用信息、请求执行路径难以理解的问题.如何结合两类提高分布式追踪技术通用性的方法,提出了一种通用的分布式追踪技术将是未来的研究趋势之一.

针对不同分布式追踪技术产生的执行路径的粒度、数据结构、因果关系、路径表示方面的不兼容的问题,Alawneh[82]倡议对其进行标准化,而开源的分布式追踪系统 Zipkin[25]与 Opentracing[26]则基于 Dapper 提出了各自的数据模型,对事件类型、因果关系与请求执行路径提出了规范.但当前的分布式追踪领域仍缺少一种通用的数据模型,能够适用于不同粒度、不通场景下的分布式追踪技术.一个通用的数据模型能够提高不同分布式追踪技术之间的兼容性,且能够加速分布式软件系统中基于请求执行路径的故障诊断、性能分析、系统理解等

各项运维任务的算法的提出,从而发挥分布式追踪技术的价值.

4.3 分布式追踪技术的评价问题

由于当前分布式追踪技术的评价指标不够完善,相关研究工作评价分布式追踪技术主要从该技术所引入的系统性能的开销[9]以及基于请求执行路径的运维任务,如故障诊断[1]的效果等方面来进行,无法完整地评价一个分布式追踪技术,制约了分布式追踪技术的发展.基于对相关研究工作的分析与总结,分布式追踪技术自身的评价指标应该包括以下几个方面.

(1) 开销.分布式追踪技术所引入的开销有 3 个方面:① 将分布式追踪技术应用于分布式软件系统后,分布式软件系统中请求响应时间的增加;② 将分布式追踪技术应用于分布式软件系统后,分布式软件系统的吞吐量(通常是指分布式软件系统所能并发处理的请求的数量)的降低;③ 为实现分布式追踪,所需要的修改分布式软件系统的代码的工作量.

(2) 通用性.分布式追踪系统的通用性代表了分布式追踪技术满足不同类型、不同架构、不同语言的分布式软件系统的追踪要求的能力.

(3) 准确性.准确性是指分布式追踪技术产生的请求路径中事件之间的正确因果关系占所有捕获的事件之间因果关系的百分比.

(4) 完整性.完整性描述了分布式追踪技术产生的请求执行路径完整刻画请求的端到端处理过程的能力.

相关的研究工作对如何评价请求响应时间开销[19]、系统吞吐量开销[9]以及代码修改的开销[20]提供了可量化的评价方法,但如何对分布式追踪技术的通用性、准确性与完整性进行评价则缺少相关的研究工作与标准.文献[31]虽然对其分布式追踪技术的准确性与完整性进行了评价,但其评价方法依赖于专家所进行的人工判断,无法应用到其他分布式追踪技术的评价中.因此,一个用于评价分布式追踪技术的基准(benchmark)系统以及在相应基准系统上对分布式追踪技术的开销、通用性、准确性及完整性进行评价的指标体系对于促进分布式追踪技术的发展是必要的.相应的研究工作也会对分布式追踪技术的完善与应用产生积极的影响.

5 结束语

分布式追踪技术是精准刻画分布式软件系统的行为与状态的重要手段,对满足分布式软件系统中的故障检测、性能调优、系统理解、资源审计等一系列运维任务有着重要的意义,其研究受到了工业界和学术界的广泛关注.但当前分布式追踪技术仍面临着新的挑战,如无法准确地判断因数据读写依赖导致的事件之间的因果关系,分布式追踪技术的通用性差,缺乏有效的对分布式追踪技术的评价指标等,这些都需要进一步的研究.

本文从分布式追踪技术的基本概念出发,提出了分布式追踪技术的研究框架,并在该框架下详细地分析并总结了已有的分布式追踪技术的研究工作.通过整理总结已有的分布式追踪技术及其应用的相关工作,进一步分析了分布式追踪技术当前所面临的问题,并对未来的研究方向进行了展望,为相关研究人员开展下一步研究工作提供了一些思路.

致谢 在此,我们向对本文工作给予支持和建议的同行表示衷心的感谢.

References:

[1] Yong Y, Long W, Jing G, Ying L. Transparently capturing execution path of service/job request processing. In: Proc. of the Int'l Conf. on Service-oriented Computing. Springer-Verlag, 2018. 879–887.

[2] Zhao X, Kirk R, Yu L, *et al*. Log20: Fully automated optimal placement of log printing statements under specified overhead threshold. In: Proc. of the 26th Symp. on Operating Systems Principles. ACM, 2017. 565–581.

[3] Thereska E, Salmon B, Strunk J, Wachs M, Abd-El-Malek M, Lopez J, Ganger GR. Stardust: Tracking activity in a distributed storage system. ACM SIGMETRICS Performance Evaluation Review, 2006,34(1):3–14.

[4] Cantrill MB, Michael WS, Adam HL. Dynamic instrumentation of production systems. In: Proc. of the USENIX Annual Technical Conf. USENIX. 2004. 15–28.

[5] Gupta M, Anindya N, Manoj KA, Gautam K. Discovering dynamic dependencies in enterprise environments for problem determination. In: Proc. of the Int'l Workshop on Distributed Systems: Operations and Management. Springer-Verlag, 2003. 221–233.

[6] Jonathan, M. End-to-end tracing: Adoption and use cases, survey. Brown University, 2017. http://cs.brown.edu/people/jcmace/papers/mace2017survey.pdf

[7] Lamport L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 1978,21(7):558–565.

[8] Chow M, Meisner D, Flinn J, Peek D, Wenisch TF. The mystery machine: End-to-end performance analysis of large-scale internet services. In: Proc. of the 11th {USENIX} Symp. on Operating Systems Design and Implementation ({OSDI} 2014). 2014. 217–231.

[9] Sigelman BH, Barroso LA, Burrows M, Mike B, Pat S, Manoj P, Donald B, Saul J, Chandan S. Dapper, a large-scale distributed systems tracing infrastructure. Google Technical Report, Google Inv., 2010.

[10] Kitajima S, Matsuoka N. Inferring calling relationship based on external observation for microservice architecture. In: Proc. of the Int'l Conf. on Service-oriented Computing. Springer-Verlag, 2017. 229–237.

[11] Zhang Z, Zhan J, Li Y, Lei W, Dan M, Bo S. Precise request tracing and performance debugging for multi-tier services of black boxes. In: Proc. of the IEEE/IFIP Int'l Conf. on Dependable Systems & Networks. IEEE, 2009. 337–346.

[12] Bahl P, Chandra R, Greenberg A, Srikanth K, David AM, Ming Z. Towards highly reliable enterprise network services via inference of multi-level dependencies. ACM SIGCOMM Computer Communication Review, 2007,37(4):13–24.

[13] Sambasivan RR, Zheng AX, De RM, Elie K, Spencer W, Mchael S, William W, Lianghong X, Grrgory RG. Diagnosing performance changes by comparing request flows. In: Proc. of the Symp. on Networked Systems Design and Implementation (NSDI). USENIX, 2011,5:1.1–5.8.

[14] Lai CA, Kimball J, Zhu T, Qingyang Q, Claton P. MilliScope: A fine-grained monitoring framework for performance debugging of n-tier Web services. In: Proc. of the 37th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS). IEEE, 2017. 92–102.

[15] Mi H, Wang H, Zhou Y, Micharl RTL, Hua C. Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems. IEEE Trans. on Parallel and Distributed Systems, 2013,24(6):1245–1255.

[16] Reynolds P, Killian CE, Wiener JL, Jeffrey CM, Mehul AS, Amin V. Pip: Detecting the unexpected in distributed systems. In: Proc. of the Symp. on Networked Systems Design and Implementation (NSDI). USENIX, 2006,6:9–9.

[17] Chen M, Emre K, Anthony A, Armondo F, Eric B. Using runtime paths for macroanalysis. In: Proc. of the USENIX Workshop on Hot Topics in Operating Systems (HotOS). USENIX, 2003. 79–84.

[18] Chen MY, Kiciman E, Fratkin E, Armondo F, Eric B. Pinpoint: Problem determination in large, dynamic internet services. In: Proc. Int'l Conf. on Dependable Systems and Networks. IEEE, 2002. 595–604.

[19] Mace J, Roelke R, Fonseca R. Pivot tracing: Dynamic causal monitoring for distributed systems. ACM Trans. on Computer Systems (TOCS), 2018,35(4):11.

[20] Mace J, Roelke R, Fonseca R. Pivot tracing: Dynamic causal monitoring for distributed systems. In: Proc. of the 25th Symp. on Operating Systems Principles (SOSP). ACM, 2015. 378–393.

[21] Barham P, Donnelly A, Isaacs R, Richad. Using magpie for request extraction and workload modelling. In: Proc. of the Symp. on Operating Systems Principles (SOSP). USENIX, 2004,4:18–18

[22] Barham P, Rebecca I, Richard M, Dushyanth N. Magpie: Online modelling and performance-aware systems. In: Proc. of the USENIX Workshop on Hot Topics in Operating Systems (HotOS). USENIX, 2003. 85–90.

[23] Li D, Mickens J, Nath S, Lenin R. Domino: Understanding wide-area, asynchronous event causality in Web applications. In: Proc. of the 6th ACM Symp. on Cloud Computing. ACM, 2015. 182–188.

[24] Kaldor J, Mace J, Bejda M, Edison G, Wiktor K, Joe O, Kian WO, Bill S, Pingjia S, Brendan V, Vinod V, Kaushik V, Yee JS. Canopy: An end-to-end performance tracing and analysis system. In: Proc. of the 26th Symp. on Operating Systems Principles (SOSP). ACM, 2017. 34–50.

[25] https://zipkin.io/

[26] https://opentracing.io/

[27] Fonseca R, Freedman MJ, Porter G. Experiences with tracing causality in networked services. In: Proc. of the Internet Network Management Workshop/Workshop on Research on Enterprise Networking (INM/WREN). USENIX, 2010,10:10–10.

[28] Fonseca R, Porter G, Katz R H, Scott S, Ion S. X-trace: A pervasive network tracing framework. In: Proc. of the 4th USENIX Conf. on Networked systems design & implementation (NSDI). USENIX Association, 2007. 20.

[29] Attariyan M, Chow M, Flinn J. X-ray: Automating root-cause diagnosis of performance anomalies in production software. In: Proc. of the 10th Symp. on Operating Systems Design and Implementation (OSDI). USENIX, 2012. 307–320.

[30] Pham C, Wang L, Tak BC, Salman B, Chunqiang T, Zbigniew K, Ravishankar KI. Failure diagnosis for distributed systems using targeted fault injection. IEEE Trans. on Parallel and Distributed Systems, 2017,28(2):503–516.

[31] Tak BC, Tang C, Zhang C, Sriram G, Bhuvan U, Rong NC. vPath: Precise discovery of request processing paths from black-box observations of thread and network activities. In: Proc. of the USENIX Annual technical conference (ATC). USENIX, 2009.

[32] Koskinen E, Jannotti J. Borderpatrol: Isolating events for black-box tracing. ACM SIGOPS Operating Systems Review, 2008,42(4): 191–203.

[33] Reynolds P, Wiener JL, Mogul JC, Marcos KA, Amin V. WAP5: Black-box performance debugging for wide-area systems. In: Proc. of the 15th Int'l Conf. on World Wide Web. ACM, 2006. 347–356.

[34] Neves F, Machado N, Jose P. Falcon: A practical log-based analysis tool for distributed systems. In: Proc. of the 48th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN). IEEE, 2018. 534–541.

[35] Aguilera MK, Mogul JC, Wiener JL, Patrick R, Athicha M. Performance debugging for distributed systems of black boxes. ACM SIGOPS Operating Systems Review, 2003,37(5):74–89.

[36] Du M, Feifei L, Guineng Z, Vivek S. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proc. of the 2017 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2017. 1285–1298.

[37] Zhao X, Rodrigues K, Luo Y, Ding Y, Michael S. Non-intrusive performance profiling for entire software stacks based on the flow reconstruction principle. In: Proc. of the Symp. on Operating Systems Design and Implementation (OSDI). 2016. 603–618.

[38] Tak BC, Tao S, Yang L, Chao Z, Yaoping R. LOGAN: Problem diagnosis in the cloud using log-based reference models. In: Proc. of the 2016 IEEE Int'l Conf. on Cloud Engineering (IC2E). IEEE, 2016. 62–67.

[39] Abrahamson J, Beschastnikh I, Brun Y, Michael DE. Shedding light on distributed system executions. In: Companion Proc. of the 36th Int'l Conf. on Software Engineering. ACM, 2014. 598–599.

[40] Zhao X, Zhang Y, Lion D, Muhammud F, Yu L, Ding Y, Micheal S. Lprof: A non-intrusive request flow profiler for distributed systems. In: Proc. of the Symp. on Operating Systems Design and Implementation (OSDI). USENIX, 2014. 629–644.

[41] Ivan B, Yuriy B, Micheal DE, Arvind K. Inferring models of concurrent systems from logs of their behavior with CSight. In: Proc. of the 36th Int'l Conf. on Software Engineering (ICSE). ACM, 2014. 468–479.

[42] Tan J, Kavulya S, Gandhi R, Priya N. Visual, log-based causal tracing for performance debugging of mapreduce systems. In: Proc. of the 30th IEEE Int'l Conf. on Distributed Computing Systems. IEEE, 2010. 795–806.

[43] Fu Q, Lou JG, Wang Y, Jiang L. Execution anomaly detection in distributed systems through unstructured log analysis. In: Proc. of the 9th IEEE Int'l Conf. on Data Mining. IEEE, 2009. 149–158.

[44] Anandkumar A, Bisdikian C, Agrawal D. Tracking in a Spaghetti Bowl: Monitoring transactions using footprints. ACM SIGMETRICS Performance Evaluation Review, 2008,36(1):133–144.

[45] Xu H, Ning X, Zhang H, Junghwan R, Guofei J. Pinfer: Learning to infer concurrent request paths from system kernel events. In: Proc. of the IEEE Int'l Conf. on Autonomic Computing (ICAC). IEEE, 2016. 199–208.

[46] Zhang H, Rhee J, Arora N, Sanhan G, Guofei J, Kenji Y, Dongyan X. CLUE: System trace analytics for cloud service performance diagnosis. In: Proc. of the IEEE Network Operations and Management Symp. (NOMS). IEEE, 2014. 1–9.

[47] Erlingsson Ú, Peinado M, Peter S, Mihai B, Gloria MR. Fay: Extensible distributed tracing from kernels to clusters. ACM Trans. on Computer Systems (TOCS), 2012,30(4):13.

[48] Mace, J, Peter B, Rodrigo F, Madanlal M. Retro: Targeted resource management in multi-tenant distributed systems. In: Proc. of the 12th USENIX Symp. on Networked Systems Design and Implementation (NSDI). USENIX, 2015. 589–603.

[49] Gschwind T, Eshghi K, Garg PK, Klaus W. Webmon: A performance profiler for Web transactions. In: Proc. of the 4th IEEE Int'l Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002). IEEE, 2002. 171–176.

[50] Barham P, Donnelly A, Isaacs R, Richard M. Using magpie for request extraction and workload modelling. In: Proc. of the Symp. on Operating Systems Design and Implementation (OSDI). USENIX, 2004,4:18–18.

[51] Yu X, Joshi P, Xu J, Guoliang J. Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs. ACM SIGPLAN Notices, 2016,51(4):489–502.

[52] Tan J, Pan X, Kavulya S, Gandhi R, Narasimhan P. SALSA: Analyzing logs as state machines. In: Proc. of the USENIX Workshop on the Analysis of System Logs (WASL). USENIX, 2008. 6.

[53] Wang T, Perng C, Tao T, *et al*. A temporal data-mining approach for discovering end-to-end transaction flows. In: Proc. of the IEEE Int'l Conf. on Web Services. IEEE, 2008. 37–44.

[54] Mi HB, Wang HM, Cai H, *et al*. P-Tracer: Path-based performance profiling in cloud computing systems. In: Proc. of the 36th IEEE Annual Computer Software and Applications Conf. IEEE, 2012. 509–514.

[55] Ostrowski K, Mann G, Sandler M. Diagnosing latency in multi-tier black-box services. In: Proc. of the 5th Workshop on Large Scale Distributed Systems and Middleware (LADIS). ACM, 2011.

[56] Kc K, Gu XH. ELT: Efficient log-based troubleshooting system for cloud computing infrastructures. In: Proc. of the 30th IEEE Int'l Symp. on Reliable Distributed Systems. IEEE, 2011. 11–20.

[57] Tak BC, Tang C, Zhang C, Sriram G, Bhuvan U, Rong NC. vPath: Precise discovery of request processing paths from black-box observations of thread and network activities. In: Proc. of the USENIX Annual Technical Conf (ATC). USENIX, 2009.

[58] Cai H, Douglas T. Distea: Efficient dynamic impact analysis for distributed systems. arXiv Preprint, arXiv:1604.0463, 2016.

[59] Wu LJ, Li HW, Cheng YJ, *et al*. Application dependency tracing for message-oriented middleware. In: Proc. of the 16th Asia-Pacific Network Operations and Management Symp. IEEE, 2014. 1–6.

[60] Chanda A, Cox AL, Zwaenepoel W. Whodunit: Transactional profiling for multi-tier applications. ACM SIGOPS Operating Systems Review, 2007,41(3):17–30.

[61] Kobayashi S, Kensuke F, Hiroshi E. Mining causes of network events in log data with causal inference. In: Proc. of the IFIP/IEEE Symp. on Integrated Network and Service Management (IM). IEEE, 2017. 45–53.

[62] Kanuparthy P, Dai Y, Pathak S, Sambit S, Theophilus B, Mojgan G, Narayan PPS. YTrace: End-to-end performance diagnosis in large cloud and content providers. arXiv Preprint, arXiv:1602.03273, 2016.

[63] Mann G, Sandler M, Krushevskaja D, Sudipto G, Eyar E. Modeling the parallel execution of black-box services. In: Proc. of the USENIX Workshop on Hot Topics in Cloud Computing (HotCloud). USENIX, 2011.

[64] Guo Z, Zhou D, Lin HX, *et al*. G2: A graph processing system for diagnosing distributed systems. In: Proc. of the USENIX Annual Technical Conf (ATC). USENIX, 2011.

[65] Gu J, Wang L, Yang Y, *et al*. KEREP: Experience in extracting knowledge on distributed system behavior through request execution path. In: Proc. of the IEEE Int'l Symp. on Software Reliability Engineering Workshops (ISSREW). IEEE, 2018. 30–35.

[66] Israr T, Murray W, Greg F. Interaction tree algorithms to extract effective architecture and layered performance models from traces. Journal of Systems and Software, 2007,80(4):474–492.

[67] Abdelwahab HL, Timothy L. Summarizing the content of large traces to facilitate the understanding of the behaviour of a software system. In: Proc. of the 14th IEEE Int'l Conf. on Program Comprehension (ICPC). IEEE, 2006. 181–190.

[68] Moc J, David AC. Understanding distributed systems via execution trace data. In: Proc. of the 9th Int'l Workshop on Program Comprehension (IWPC). IEEE, 2001. 60–67.

[69] Kuhlenkamp J, Markus K. Costradamus: A cost-tracing system for cloud-based software services. In: Proc. of the Int'l Conf. on Service-oriented Computing. Springer-Verlag, 2017. 657–672.

[70] Fonseca R, Dutta P, Phillip L, Ion S. Quanto: Tracking energy in networked embedded systems. In: Proc. of the Symp. on Operating Systems Design and Implementation (OSDI). USENIX, 2008,8:323–338.

[71] Enck W, Gilbert P, Han S, Vasant T, Byung-gon C, Landon PC, Jaeyeon J, Patrick M, Anmol NS. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. ACM Trans. on Computer Systems (TOCS), 2014,32(2):5.

[72] Sambasivan RR, Shafer I, Mazurek ML, Gregory RG. Visualizing request-flow comparison to aid performance diagnosis in distributed systems. IEEE Trans. on Visualization and Computer Graphics, 2013,19(12):2466–2475.

[73] Chen MY, Accardi A, Kiciman E, Jim L, Dave P, Armondo F, Eric B. Path-based faliure and evolution management. In: Proc. of the 1st Conf. on Symp. on Networked Systems Design and Implementation. USENIX Association, 2004. 23.

[74] Kavulya SP, Daniels S, Joshi K, Matti H, Rajeev G, Priya N. Draco: Statistical diagnosis of chronic problems in large distributed systems. In: Proc. of the IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN 2012). IEEE, 2012. 1–12.

[75] Yuan D, Mai HH, Xiong WW, *et al*. SherLog: Error diagnosis by connecting clues from run-time logs. ACM SIGARCH Computer Architecture News, 2010, 143–154.

[76] Wang C, Kavulya SP, Tan J, Liting H, Mahendra K, Mike K, Karsten S, Priya N, Rajeev G. Performance troubleshooting in data centers: An annotated bibliography. ACM SIGOPS Operating Systems Review, 2013,47(3):50–62.

[77] Luo C, Lou JG, Lin Q, Qiang F, Rui D, Dongmei Z, Zhe W. Correlating events with time series for incident diagnosis. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM, 2014. 1583–1592.

[78] Chen P, Qi Y, Hou D. CauseInfer: Automated end-to-end performance diagnosis with hierarchical causality graph in cloud environment. IEEE Trans. on Services Computing, 2016,12(2):214–230.

[79] Chen P, Qi Y, Zheng P, Di H. Causeinfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In: Proc. of the IEEE INFOCOM & IEEE Conf. on Computer Communications. IEEE, 2014. 1887–1895.

[80] Zhang L, Bild DR, Dick RP, Mao ZM, Peter D. Panappticon: Event-based tracing to measure mobile application and platform performance. In: Proc. of the Int'l Conf. on Hardware/Software Codesign and System Synthesis (CODES+ISSS). IEEE, 2013. 1–10.

[81] http://incubator.apache.org/projects/htrace.html

[82] Alawneh L, Hamou-Lhadj A. Execution traces: A new domain that requires the creation of a standard metamodel. In: Proc. of the Int'l Conf. on Advanced Software Engineering and Its Applications. Springer-Verlag, 2009. 253–263.

杨勇(1993－),男,博士生,主要研究领域为分布式系统,云计算,分布式追踪.

吴中海(1968－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为大数据技术,系统安全,嵌入式软件.

李影(1975－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为分布式计算,可信计算.