如何快速写好
第一篇学术论文

余广坝
中山大学
2020.12.18

八股取士与科研论文写作

- 破题
- 承题
- 起讲
- 起股
- 中股
- 后股
- 束股
- 大结

破题立意

行文格式相对固定

取悦审稿人

- Title
- Abstract
- Introduction
- Background & Motivation
- Approach
- Experiment
- Related Work
- Conclusion

八股作为一种文章的形式，本身并无善恶可言。

Title

- 先问问老师吧 hhh

In this paper we present PARIS, a **P**erformance-**A**ware **R**esource **I**nference **S**ystem, which estimates the

We present PRFL, a lightweight technique that boosts spectrum-based fault localization by differentiating tests using PageRank algorithm. Given the original program spectrum information, PRFL

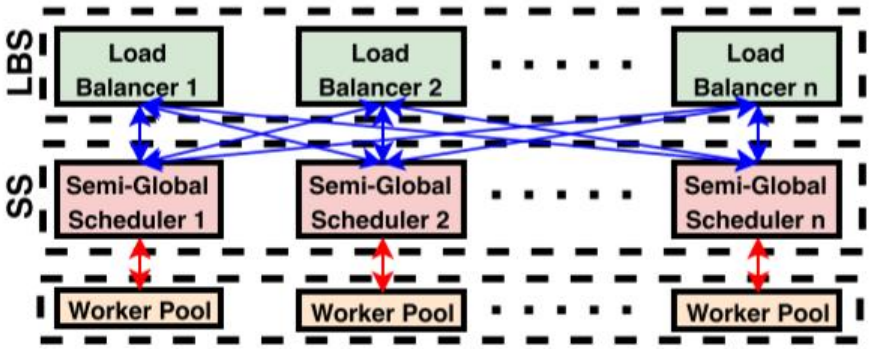


Figure 3. Archipelago Architecture. Core services include load balancing service and a scheduling service consisting of semi-global schedulers that manage their own worker pool.

Abstract

Abstract

Modern user-facing, latency-sensitive web services include numerous distributed, intercommunicating microservices that promise to simplify software development and operation. However, multiplexing compute-resources across microservices is still challenging in production because contention for shared resources can cause latency spikes that violate the service-level objectives (SLOs) of user requests. This paper presents *FIRM*, an intelligent fine-grained resource management framework for predictable sharing of resources across microservices to drive up overall utilization. FIRM leverages online telemetry data and machine-learning methods to adaptively (a) detect/localize microservices that cause SLO-violations, (b) identify low-level resources in contention, and (c) take actions to mitigate SLO-violations by dynamic reprovisioning. Experiments across four microservice benchmarks demonstrate that FIRM reduces SLO violations by up to 16.7× while reducing the overall requested CPU limit by up to 62.3%. Moreover, FIRM improves performance predictability by reducing tail latencies by up to 11.5×.

XXX 被广泛应用

XXX 的出现
带来了什么问题

简要介绍本文提出的方法

展示实验结果
突出数字

Introduction

- 介绍 XXX 被广泛应用的大背景
- XXX 的出现带来了什么问题
- 不解决这个问题会造成什么（解决了有何收益）
- 解决这个问题有哪些挑战
- 之前研究的局限性
- 为了克服之前研究的缺陷，本文提出了XXX
- 展示实验结果
- 贡献

Introduction

- 介绍 XXX 被广泛应用大背景

Modern user-facing, latency-sensitive web services, like those at Netflix [64], Google [72], and Amazon [82], are increasingly built as microservices executing on shared/multi-tenant compute resources either as virtual machines (VMs) or as containers (with containers gaining significant popularity of

Tips: 引用几个大公司增加可信性

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Introduction

- XXX 的出现带来了什么问题
- 不解决这个问题会造成什么（解决问题的Motivation）

late). These microservices must handle diverse load characteristics while efficiently multiplexing shared resources in order to maintain service level objectives (SLOs) like end-to-end latency. SLO violations occur when one or more "critical" microservice instances (defined in §2) experience load spikes (due to diurnal or unpredictable workload patterns) or shared-resource contention, both of which lead to longer than expected time to process requests, i.e., latency spikes [3, 9, 18, 26, 31, 48, 53, 54, 90, 91]. Thus, it is critical to efficiently multiplex shared resources among microservices to reduce SLO violations.

Tips: 突出问题的重要性

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Introduction

- 解决这个问题有哪些挑战

- **Complex network dependencies**. With a microservice architecture, an application is decoupled into many fine-grained components with an extraordinarily complex network topology. Moreover, to connect different micro services wrapped in a container, an overlay network such as *flannel* is always adopted, which further increases the complexity of performance diagnosis.
- **Continuous integration and delivery.** A microservice system is evolving all the time with continuous integration and delivery technologies. According to a DevOps report from Puppet [1], one excellent enterprise may have 1600 updates a year. That means the anomaly detection and root cause diagnosis procedure should adapt to these changes in order to achieve better results.
- **Dynamic run-time environment.** A microservice system always runs in a containerized environment where the states of containers change frequently. The highly dynamic environment exacerbates the difficulty of performance diagnosis.

Tips: 分点叙述

Lin, et.al, Microscope: Pinpoint the Abnormal Services with Causal Graphs in Micro-service Environments

Introduction

- 之前的方法是怎么做的，有哪些局限性

Over the years, many approaches have been proposed to locate root causes in large distributed systems [4, 9, 20, 26, 31, 34]. CauseInfer [4] and Sieve [34] build causality graphs of system components, then they pinpoint root causes with these graphs, but their methods cannot get the exact direction of dependency between two service instances. Roots [9] automatically identifies the root causes of performance anomalies in web applications deployed in PaaS clouds, but it is not natively designed for microservice systems. MicroScope [20] maintains a service dependency graph via PC-algorithm. However, MicroScope only considers abnormal traces and leaves out the information provided by normal traces.

Tips: 突出的是局限性，不是否定前人的工作

Yu, et.al, MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments

Introduction

- 为了克服之前研究的缺陷，本文提出了XXX
  - 比摘要稍微详细介绍论文提出的方法

To address the drawbacks of existing work and new challenges in microservice systems, this paper introduces a novel approach, named *MicroRank*, which is based on extended spectrum analysis to identify the latency issues in microservice systems. It primarily comprises four procedures including *Anomaly Detector*, *Data Preparator*, *PageRank Scorer* and *Weighted Spectrum Ranker*. Our method is compatible with traces of multiple widely-used microservice trace standards (e.g., OpenTracing [3], OpenCensus [4] and OpenTelemetry [5]). Once a latency issue is detected by the *Anomaly Detector* in *MicroRank*, the cause localization procedure is triggered. *Data Preparator* first distinguishes which traces are abnormal and mind the relations of service instances and traces. Then, *MicroRank*'s

Yu, et.al, MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments

Introduction

- 展示实验结果

Results. FIRM significantly outperforms state-of-the-art RM frameworks like Kubernetes autoscaling [16, 50] and additive increase multiplicative decrease (AIMD)-based methods [34, 93].

- It reduces overall SLO violations by up to 16.7× compared with Kubernetes autoscaling, and 9.8× compared with AIMD-based method, while reducing the overall requested CPU by as much as 62.3%.

Tips: 使用倍数、百分数突出效果

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Introduction

- 贡献

level ML models. Our main contributions are:

1. *SVM-based SLO Violation Localization:* We present (in §3.2 and §3.3) an efficient way of localizing microservice instances responsible for SLO violations by extracting critical paths and detecting anomaly instances in near-real time using telemetry data.
2. *RL-based Mitigation:* We present (in §3.4) an RL-based resource contention mitigation mechanism that (a) addresses the large state space problem and (b) is capable of tuning

Tips: 必须要写，分点突出显示

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Introduction

- 下文组织结构

The rest of this paper is organized as follows. Section 2 presents the system overview and formulation. Section 3 elaborates the details of Microscope. In Section 4, we will evaluate Microscope in the controlled environment. And in Section 5 we will compare our work with previous work. We discuss the advantages of Microscope and Section 6 concludes this paper.

Tips: 没啥特别大用处，个人觉得会议论文可不写

Lin, et.al, Microscope: Pinpoint the Abnormal Services with Causal Graphs in Micro-service Environments

Introdution

- 突出逻辑
  - 介绍 XXX 被广泛应用的大背景
  - XXX 的出现带来了什么问题
  - 不解决这个问题会造成什么（解决了有何收益）
  - 解决这个问题有哪些挑战
  - 之前研究的局限性
  - 为了克服之前研究的缺陷，本文提出了XXX

Background

- 集中介绍一两个非此领域难理解的背景
  - 做一些专业名词的定义

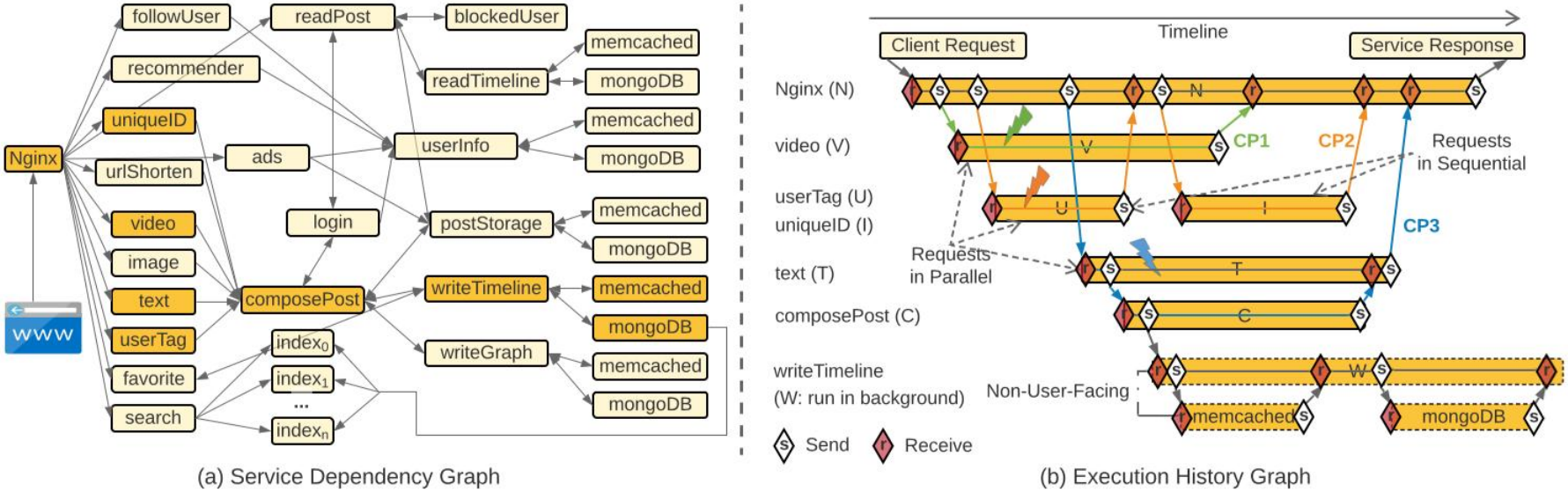


Figure 2: Microservices overview: (a) Service dependency graph of *Social Network* from the DeathStarBench [30] benchmark; (b) Execution history graph of a `post-compose` request in the same microservice.

Tips: Background 个人认为不太适合介绍论文使用方法的背景

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Motivation

- Insight + 解释 + 引出方法 Motivation

Insight 2: Microservices with Larger Latency Are Not Necessarily Root Causes of SLO Violations. It is important

belled "Text" and "Compose"). Hence, scaling microservices with higher variance provides better performance gain.

实验观察出延迟最高的不一定是根因，方差最大的更有可能是根因，为后面的方法把方差作为模型输入做铺垫

Tips: 如果是一个领域方法在另外一个领域应用，尽量写

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Approach

- 问题定义
- 方法框架
- 模块1
- 模块2
- 模块3

Approach

- 问题定义

Based on the above information, the formal description of the latency issue locating problem in microservice systems is as follows. Given a collection of traces in a time window, namely $\mathbf{T} = \{T_1, \cdots, T_n\}$, where $T_i = \{O_1^i, \cdots, O_m^i\}$, and O_j^i demotes one operation in trace i, we get the end-to-end latency of these traces, namely $\mathbf{L} = \{L_1, \cdots, L_n\}$ and the coverage graph $G = (V, E)$, where V is the collection of service instance operations, namely $O_j^i \in V$ and E is the collection of calling relations. The problem is to find out normal traces $\mathbf{T}^n = \{T_1^n, \cdots, T_k^n\}$ and abnormal traces $\mathbf{T}^a = \{T_1^a, \cdots, T_{n-k}^a\}$. Moreover, based on $\mathbf{T}^n$, $\mathbf{T}^a$, and G, finding the root cause service instances related to O_j^i. Our target is to rank service instances directly relevant to the root cause higher than those unrelated to it.

Tips: 利用数学符号定义问题；明确问题输入和输出

Yu, et.al, MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments

Approach

- 方法框架 overview

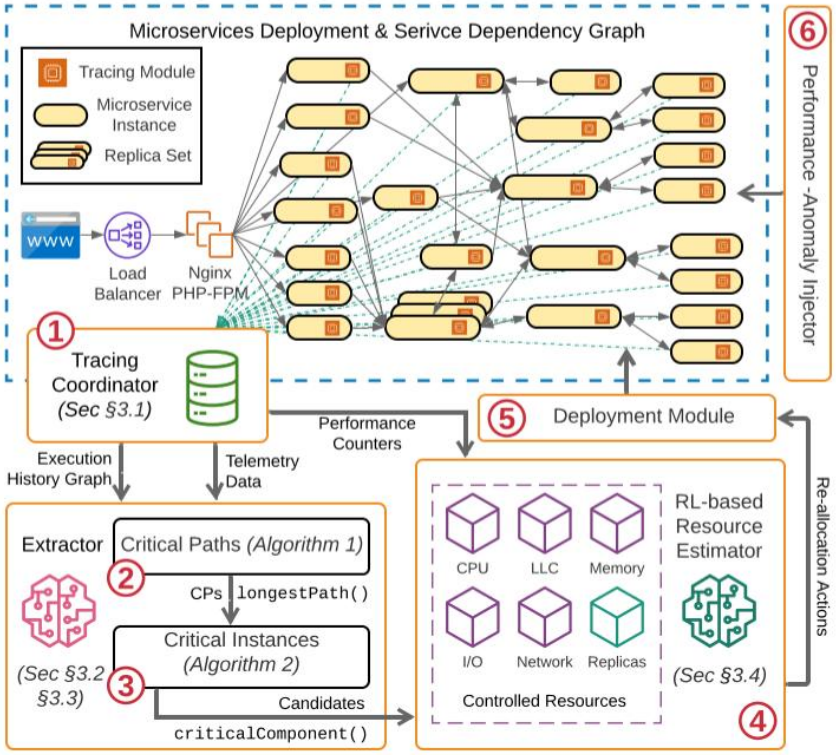


Figure 6: FIRM architecture overview.

critical microservice instances that are likely causes of SLO violations (illustrated as ③ and described in §3.3).

3. Using the telemetry data collected in ① and the critical instances identified in ③, FIRM makes mitigation decisions to scale and reprovision resources for the critical instances (illustrated as ④). The policy used to make

Tips: 利用序号来介绍框架

Tips: 框架图一定要漂亮！ 图上明确模块输入和输出

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Approach

- 模块 n
  - 方法简介
  - 为什么用这个方法
  - 在本文中是怎么用的

Tips: 简单的模块可以一两段，核心模块尽量使用这个框架

Approach

- 模块 n
  - 方法简介

FIRM leverages reinforcement learning (RL) to optimize resource management policies for long-term reward in dynamic microservice environments. In particular, FIRM utilizes the deep deterministic policy gradient (DDPG) algorithm [55], which is a *model-free*, *actor-critic* RL framework (shown in Fig. 7).

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Approach

- 模块 n
  - 为什么用这个方法

Why RL? Existing performance-modeling-based [19, 34, 35, 51, 87, 93, 117] or heuristic-based approaches [4, 5, 33, 61, 78] suffer from model reconstruction and retraining problems because they do not tackle with dynamic system status. Moreover, they require expert knowledge with significant effort to devise, implement and validate their understanding of the microservice workloads as well as the underlying infrastructure. RL on the other hand is well suited for learning resource

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Approach

- 模块 n
  - 在本文中是怎么用的

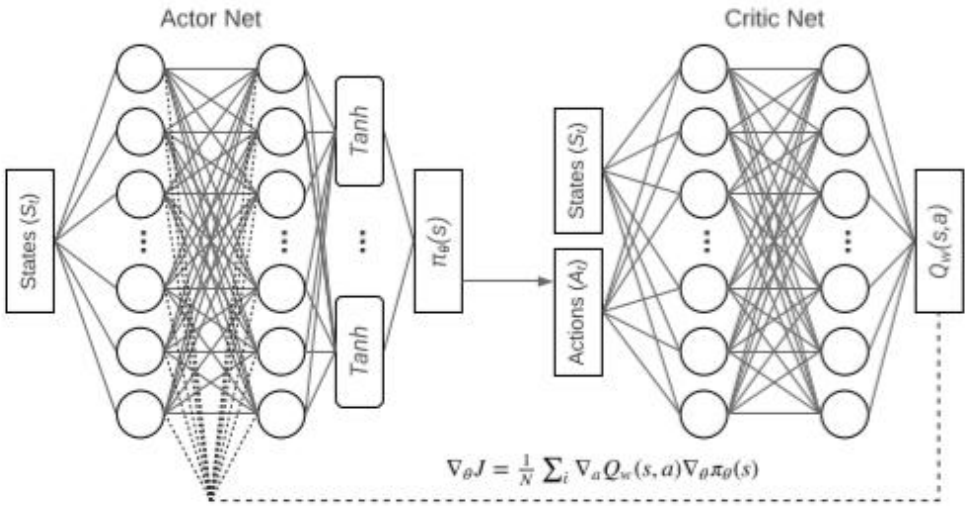


Figure 8: Architecture of actor-critic nets.

Table 3: State-Action Space of the RL-agent.

State (s_t)
SLO Violation Ratio (SV_t), Workload Changes (WC_t), Request Composition (RC_t), Resource Utilization (RU_t)
Action Space (a_t)
Resource Limits $RLT_i(t), i \in \{\text{CPU, Mem, LLC, IO, Net}\}$

$$\mathcal{L}(w) = \frac{1}{N}\sum_i (r_i + \gamma Q'_{w'}(s_{i+1}, \pi'_{\theta'}(s_{i+1})) - Q_w(s_i, a_i))^2.$$

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Evaluation

- 实验设计
- 各模块实验表现
- 参数影响
- 对比与消融实验
- 讨论

Evaluation

- 实验设计
  - 实验平台
  - 数据集/Benchmark
  - 评价指标

System Setup. We validate our design by implementing a prototype of FIRM using Kubernetes [16] as the underlying container orchestration framework. We deploy FIRM on a cluster of 15 two-socket physical nodes. Each server consists of 56–192 CPU cores and RAM varying from 500GB–1000GB. Nine of the servers use Intel `x86` Xeon E5s and E7s, while the remaining use IBM `ppc64` Power8 and Power9. All machines run Ubuntu 18.04.3 LTS. The four microservice benchmarks are deployed and orchestrated using Kubernetes.

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Evaluation

- 各模块实验表现

Table 4: Latency issues localization results on dataset $\mathcal{A}$

DataSet	Formula	R@1			R@3			R@5			Exam Score				
		PR	SP	MR	PR	SP	MR	PR	SP	MR	PR	SP	MR	IMP*	IMP†
$\mathcal{A}$	Dstar2	78	88	94	78	94	96	92	96	96	1.16	0.56	0.42	63.79%	25.00%
	Goodman	78	86	88	86	92	92	92	92	92	1.16	0.88	0.84	27.59%	4.45%
	Jaccard	78	86	88	86	92	92	90	92	92	1.28	0.88	0.84	34.38%	4.45%
	M2	78	90	94	86	94	96	92	94	96	1.16	0.60	0.42	63.79%	30.00%
	Ochiai	78	88	94	86	94	96	92	96	96	1.16	0.56	0.42	63.79%	25.00%
	Sørensen	78	86	88	86	90	92	92	92	92	1.16	0.88	0.84	34.38%	4.45%
	RussellRao	78	86	92	86	92	96	92	96	96	1.28	0.64	0.44	65.63%	31.25%
	Dice	78	86	88	86	92	92	90	92	92	1.32	0.88	0.84	36.36%	4.45%

[∗] denotes the improvement between PR and MR
[†] denotes the improvement between SP and MR

Yu, et.al, MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments

Evaluation

- 参数影响

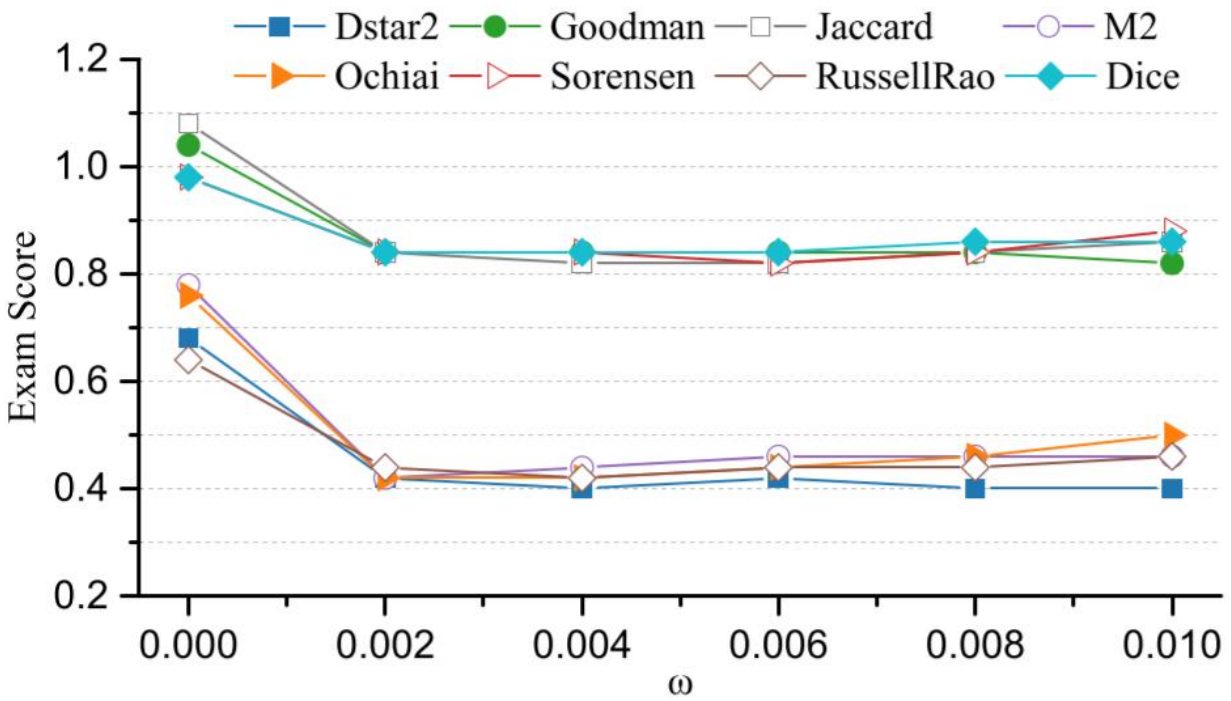


Figure 13: Impact of ω on Exam Score

Yu, et.al, MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments

Evaluation

- 消融实验: 去掉论文中某个模块后结果对比
- 对比实验: 与其他论文方法对比

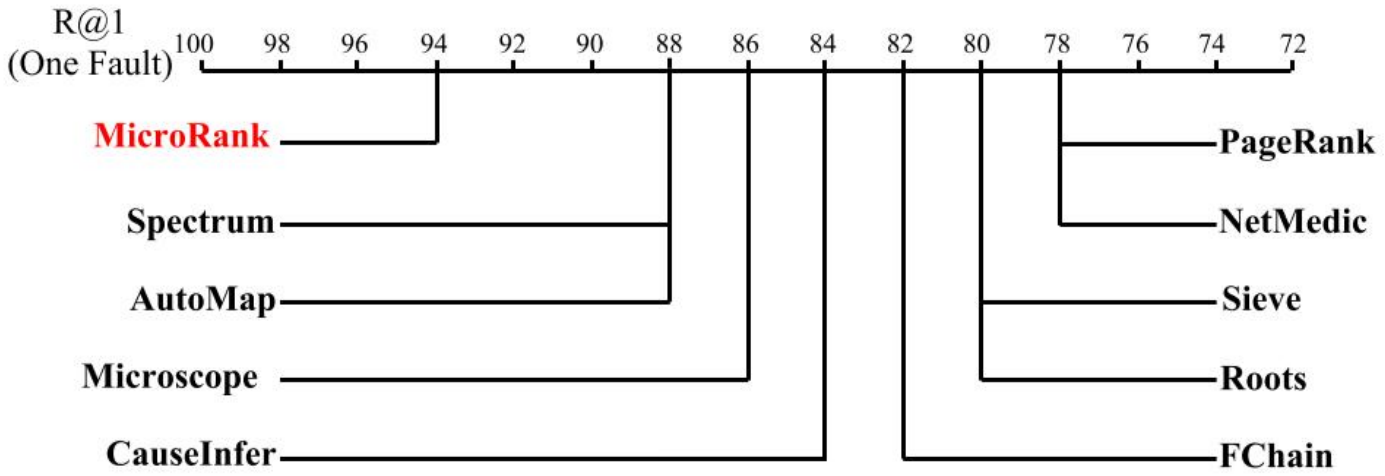


Figure 16: *R@1* **results among different RCA methods on** $\mathcal{A}$

Tips: 解释为何新方法比以前的方法好非常重要

Yu, et.al, MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments

Evaluation

- 讨论
  - 模块开销 (运行设备，CPU、内存)
  - 可扩展性：更大规模系统/数据集
  - 方法局限性（Limitation）

Table 7: Overhead statistics on benchmark experiment.

System Module	Overhead
Instrumented client	2% ± 1% CPU utilization(Single core)
Anomaly Detector	8% ± 4% CPU utilization(Single core)
Data Preparator	60% ± 5% CPU utilization(Single core)
PageRank Scorer	20% ± 5% CPU utilization(Single core)
Spectrum Scorer	2% ± 1% CPU utilization(Single core)
Anomaly Detector	0.8 second (Single physical node)
Data Preparator	21.5 second (Single physical node)
PageRank Scorer	9.5 second (Single physical node)
Spectrum Scorer	0.1 seconds (Single physical node)

Tips: Limitation 最好是要写一下

Related work

- 分点叙述
  - 参考领域 Survey 和其他相关文章分点

Root Cause Analysis. A large body of work [13,31,45,47, 57,59,86,100,111,114] provides promising examples that data-driven diagnostics help detect performance anomalies and analyze root causes. For example, Sieve [100] leverages Granger causality to correlate performance anomaly data series with particular metrics as potential root causes. Microscope [57] and MicroRCA [114] are both designed to identify abnormal services by constructing service causal graphs that model anomaly propagation and by inferring causes using graph traversal or ranking algorithms [46]. Seer [31] leverages deep learning to learn spatial and temporal patterns that translate to SLO violations. However, none of these approaches addresses the dynamic nature of microservice environments (i.e., frequent microservice updates and deployment changes), which require costly model reconstruction or retraining.

Tips: 不可小觑 Related Work, 论文评审会有专门一块评分是否引用完备，突出本文工作的不同，

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

Conclusion

- 本文提出了 XXX 方法解决了什么问题
- 取得什么样的结果
- 未来方向

7 Conclusion

We proposed *FIRM*, an ML-based, fine-grained, resource management framework that addresses SLO violations and resource under-utilization in microservices. FIRM uses a two-level ML model, one for identifying microservices responsible

Qiu, et.al. FIRM: An Intelligent Fine-Grained Resource Management Framework for SLO-Oriented Microservices

其他 tips

- 多用分点阐述

the following questions: (1) how many sandboxes of each function must be setup proactively? (2) how should these sandboxes be placed on its worker pool? (3) when/how should these sandboxes be evicted from the proactive memory pool?

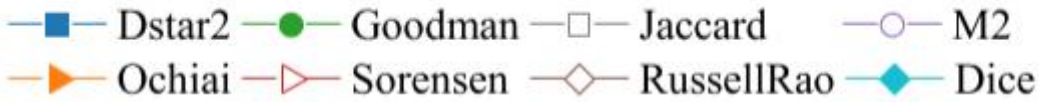
- **Complex network dependencies**. With a decoupled into many fine-grained component topology. Moreover, to connect different micr lay network such as *flannel* is always adopte performance diagnosis.
- **Continuous integration and delivery.** A with continuous integration and delivery techr Puppet [1], one excellent enterprise may have 1 detection and root cause diagnosis procedure achieve better results.

Workload. We consider four different classes of DAGs: **(i) C1** consists of DAGs that have a single function, short execution times and tight deadlines. These DAGs represent user-facing functions. **(ii) C2** consists of DAGs that have a single function, short execution times, and less strict deadlines. These DAGs represent non-critical user-facing functions (such as updating a metrics dashboard). **(iii) C3** consists of DAGs that have

utilization trade-off space. For example, it can avoid (a) scheduling delays due to time-sharing and (b) high-level cache pollution, by

其他 tips

- 图表类
  - 首页有图，抓住眼球
  - 图表要多，一图胜千言
  - 图要好看，颜值就是战斗力
  - 全文图表数量接近（有图有表）
  - 图标题在下，表标题在上
  - 作图时考虑黑白打印（不能仅使用颜色区分）



  - 图表解释写在标题里（可选）
  - 矢量图（eps, svg）

其他 tips

- 算法类
  - 行数
  - 关键词加粗
  - 明确输入输出

Algorithm 2 The *cause inference* algorithm

Input: An original abnormal service instance, rootNode; A causality graph DAG, G;
Output: A ordered list of root cause candidates

```
// Find root cause candidates
stack ← Stack(); candidates ← List()
stack.push(rootNode)
while stack is not empty do
    node ← stack.pop()
    // adj(G, X) represents the neighbors which are adjacent to X in G.
    if adj(G, node) is empty then
        candidates.append(node)
        continue
    end if
```

其他tips

- References
  - 文字与引用中间有个空格 Kubernetes [14], Mesos [31] and Swarm [11]
  - 统一格式（标题首个单词首字母大写，或所有单词首字母大小）
  - 会议缩写/不缩写

[8] J. Lin, P. Chen, and Z. Zheng, *Microscope: Pinpoint Performance Issues with Causal Graphs in Micro-service Environments: 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings*. Springer, Cham, 2018.

[9] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Acm Sigcomm Conference on Data Communication*, 2009.

[5] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software*

[18] R. Fonseca, G. Porter, R. H. Katz, and S. Shenker, "X-trace: A pervasive network tracing framework," in *4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 07)*. Cambridge, MA: USENIX Association, Apr.

其他 tips

- 工具类
  - Latex
  - Mathpix Sniping (截图生成Latex公式)
  - Excel2Latex (Excel 表格转 Latex)
  - Origin (实验结果图)
  - 亿图图示（框架图）
  - 查找引用 Bibtex 文件
    - dblp （无需下载）
    - scopus
  - 易搜搭（常用词替换）
  - 微软爱写作/Grammaly (语法检查)

总结

- 多想想 Reviewer 喜欢看什么样的论文
- 注重行文逻辑
- Motivation 非常重要
- 每个缺失的模块都可能会被怼
- 多举例子
  - 贯穿全文的大例子
  - 小例子
- 注意细节

Q & A

Thanks