持续软件工程中软件性能的架构支持：一项系统映射研究

Romina Eramo[a], *, Michele Tuccic, Daniele Di Pompeob,Vittorio Cortellessab,Antinisca Di Marcob,Davide Taibid,e

[a]泰拉莫大学，泰拉莫，意大利[b]拉奎拉大学，拉奎拉，意大利[c]查理大学，布拉格，捷克共和国[d]奥卢大学，奥卢，奥卢，芬兰[e]坦佩雷大学，坦佩雷，芬兰

arXiv:2304.02489v1 [cs.SE] 5 Apr 2023

Abstract

持续软件工程范式在现代开发实践中日益受到欢迎，其中，设计活动与运行时活动的交错由软件系统的持续演化所促成。在此上下文中，性能评估并不容易，但近期研究表明，随着软件不断演化的架构模型能够支持这一目标。在本文中，我们开展了一项映射研究，旨在对现有的、涉及面向性能的持续软件工程架构支持的科学贡献进行分类。我们将系统映射方法应用于初始的一组215 篇潜在相关的论文，并选取了66项主要研究，通过分析这些研究来刻画和分类当前的研究现状。这一分类有助于聚焦于该领域正在考虑的主要方面，尤其是新出现的发现及其对未来研究的启示。

关键词： 软件架构、软件性能、持续软件工程、DevOps ps

1. Introduction

持续软件工程（CSE）是一种很有前景的软件过程，它将业务战略（即需求工程）、开发和运维融入一个连续的过程。其目标是打造更出色的软件产品，并实现更多成功的落地应用，以满足相关需求和约束。同样，近期对 DevOps 的重视也体现出这样一种认识：软件开发与运维交付之间的集成必须持续进行。DevOps 改善了利益相关者、开发团队和运维团队之间的端到端协作。此外，DevOps 已成功应用于安全性和测试等领域。软件性能（SP）是软件系统采用并取得成功的一项关键质量属性。研究人员和工业界从业者已认识到，及时、高效地将性能工程实践集成到持续开发流程中的重要性 [1]。然而，当前的软件性能工程方法并未针对采用 CSE 流程的环境量身定制，相关实践也滞后于发展 [2, 3]。

尽管软件性能是一种与软件所部署的平台相关的非功能属性，但在过去二十年中，性能评估主要是在设计层面通过软件架构（自适应）等方法进行估算的 [4, 5, 6]。自适应可以转换为性能模型，其指标可用于比较不同的自适应方案。此类设计时性能评估并未充分考虑目标平台特征的诸多方面。然而，这些早期阶段的比较分析揭示了不同方案结果之间的明显差异，无疑能够帮助架构师以更全面的视角了解其性能影响并做出决策。

在过去十年中，持续工程范式的兴起显著改变了软件过程。如今，软件工程通常需要在运行中的代码与设计模型之间形成持续循环，使流程的这两个方面能够相互反馈 [7]。例如，可以从执行追踪中收集运行时数据，为软件模型提供数据。软件模型旨在检查功能属性或非功能属性。在即将到来的执行场景中分析软件模型，可以提出即时重构/自适应措施，从而在这些场景发生时维持可接受的软件行为 [8, 9, 10]。在 CSE 流程中，架构模型似乎日益重要，其中一个原因是它们能够支持与性能相关的决策 [11]。

尽管人们对采用持续架构师-

*通讯作者电子邮件地址：reramo@unite.it (Romina Eramo), tucci@d3s.mff.cuni.cz (Michele Tucci), daniele.dipompeo@univaq.it (Daniele Di Pompeo), vittorio.cortellessa@univaq.it(Vittorio Cortellessa), antinisca.dimarco@univaq.it (Antinisca Di Marco), davide.taibi@oulu.fi (Davide Taibi)

方法和性能工程实践方面，关于在持续工程过程中可使用的不同性能工程技术的适用性，文献中的共识仍然很少。已有少量研究发表，这些研究从自适应系统和微服务的某些特定方面考察了持续工程 [12, 13, 14, 15, 16]。

然而，当前的CSE和 DevOps 实践侧重于快速交付、缩短新功能发布所需的时间、降低风险、提升新的效率，以及建立持续交付流水线。高效且自动化的性能工程工具至关重要，而要实现这一目标仍面临诸多相关挑战。因此，仍有必要开展一项研究，系统地调查该主题的所有关键出版物，并识别出可能适用于持续工程流程的性能工程技术。

在本研究中，我们对现有文献开展映射研究 [17]，以调查科学界针对CSE 中软件性能的架构支持所作出的贡献。此外，本研究旨在刻画并分类当前的研究现状，从而更好地构建我们对该主题的理解，并确定近期值得在这一领域开展研究的方向。

本研究的主要贡献包括：

- 一种可复用的框架，用于对专门针对持续软件工程中软件性能架构支持的解决方案、方法和技术进行分类、比较和评估；
- 一份关于 CSE 中软件性能架构支持研究现状的系统性图谱，涵盖性能领域、领域、所解决的问题以及采用的工具；
- 对新兴趋势、文献空白及其对未来研究影响的讨论。

本文其余部分的组织结构如下：在第 2 节中，我们回顾背景并比较现有文献。在第3节中，我们定义目标问题，并阐述为开展映射研究而采用的过程；在第 4—8 节中，我们描述并分析所得结果，以回答目标问题。在第9 节中，我们讨论研究有效性的威胁，最后，第 10 节给出结论性说明。

2. 背景与相关工作

本节介绍一些背景信息，并阐述相关主要概念之间的协同作用；同时也介绍了其他相关主题的研究。

2.1. 主要概念

持续软件工程（CSE）。这指的是在快速并行周期中开发、发布软件并从中学习的能力。这包括确定新功能以构建、演进和重构架构，开发功能，验证它，将其发布给客户，并收集客户的实验反馈，以指导下一轮开发 [18]。

CSE 的定义容易产生不同解读，且通常与整个软件（工程）生命周期中出现的其他持续性活动结合使用 [7]。具体而言，开发 阶段涉及的活动包括：持续集成、持续部署/发布、持续交付，以及持续验证/测试。而运维 阶段则涉及流程的末端，即启动版本交接的阶段；在此阶段，除了持续监控以观察并发现合规问题和风险外，还特别关注在初次采用这些系统后对其进行持续使用。近期兴起的 *DevOps* [19] 实践推动了开发与运维的集成，印证了这两个领域密切互动对于实现 CSE 至关重要。最后，业务管理与软件开发职能之间也有必要建立更紧密、持续的联系，以使业务规划等活动受益；*BizDev* [7] 这一现象补充了 DevOps，将业务管理与软件开发和运维职能整合起来。

软件性能（SP）。这代表软件工程活动及相关分析的整体集合，这些活动和分析贯穿整个软件开发周期，旨在满足性能需求 [20]。该领域侧重于对现代软件系统（例如，数据密集型、自主、分布式、自适应和嵌入式系统）进行定量评估，并权衡性能与其他服务质量（QoS）属性（例如，安全性、可靠性和可用性）。过去几十年中，人们开发了众多用于系统评估的性能工程方法、方法论和技术 [21]。

软件性能评估是软件工程中的一项关键任务，旨在确保新版本的软件不会损害用户感知的性能。性能下降可能以多种形式出现，例如响应时间延迟过高、吞吐量过低以及资源利用率过高。尽管这些论点表明应在每次变更时评估性能，但近期关于持续工程的研究表明，这尚未成为标准做法 [22]。

软件架构（自适应）。软件系统的自适应是系统的结构或多个结构，这些结构由软件组件、这些组件的外部可见属性以及它们之间的关系组成 [23]。自适应通常是第一个用于体现如何实现各类需求相关决策的设计制品。它展示了需求与所构建系统之间的对应关系，从而为设计决策提供依据 [24]。整体系统结构的设计，尤其是在大型和复杂系统中，是一个关键因素。例如，性能在很大程度上取决于不同组件之间所需通信、协调或协作的复杂程度，特别是在复杂分布式系统中。

对自适应评估的需求源于这样一种认识：软件开发与所有工程学科一样，是

一个持续建模和精化的过程。在大部分开发工作完成之前检测架构问题，使得重新架构活动能够及时进行，而无需返工已经完成的工作。同时，调优活动在软件生命周期中增强并维持软件性能 [25]。

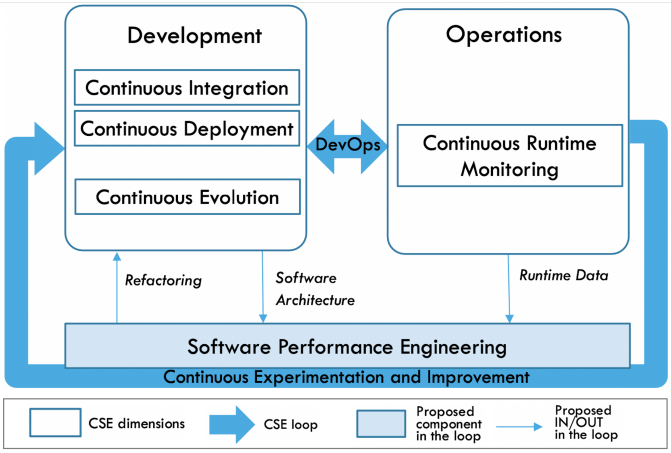


图 1：所考虑的上下文概览

CSE、软件性能和自适应之间的协同效应。 CSE 面临着诸多挑战，包括自适应的演化，以及检测和解决与软件质量属性相关的问题，例如性能 [7]。在实践中，软件开发涉及软件的持续演化，这主要是由于不断出现新的需求。自适应可能不足以嵌入新要求的功能，从而迫使软件进行繁重而复杂的演化。当原始开发人员不再可用时，情况会更加复杂。在此类情况下，系统的可维护性与其架构密切相关。

通过持续工程实践，开发人员能够更好地控制和了解缺陷，并访问质量属性状态，从而在系统开发过程中修复任何潜在问题。软件工程中的设计时与运行时交互支持动态自适应，并确保非功能属性和终端用户期望 [27]。值得注意的是，持续监控被认为有助于及早检测服务质量问题，例如性能下降 [28]。

图 1展示了本研究的上下文。我们考虑了 [7] 中提出的CSE整体视角，并通过仅关注对软件性能至关重要的持续活动（白色矩形框中）以及添加性能评估这一具体任务（浅蓝色矩形框中）来调整该图- 形。

该图还展示了 CSE与软件性能之间的桥接工件。自适应和运行时数据是CSE 的输出工件，为软件性能提供输入，而重构则进入 CSE。自适应是代表模型复杂性与表达能力之间最佳权衡的抽象，并支持对（软件）系统性能特征的评估。运行时数据代表正在运行的系统，为设置由自适应定义的性能模型提供所有参数。重构包括关于如何更改软件系统以解决或 mitigate performance degradation.

基于上述定制，我们制定了一个查询，以提取分析所涉及的文献对象。

2.2. 相关研究

在本节中，我们讨论以某种方式探讨 CSE 范式中自适应和软件性能作用的二级研究。

Koziolek [12] 开展了一项全面的文献综述，根据1999年至2009年发表的研究，对有关基于组件的软件系统的性能预测和测量方法进行了分类。这些方法引入了用于软件组件性能的专用建模语言，旨在理解所设计架构的性能，而不是进行以代码为中心的性能修复。该综述指出，软件组件运行时生命周期阶段的工具支持有限，并且对于性能建模语言尚未达成共识。事实上，由于工具支持有限，所评审的方法中没有任何一种准备好得到广泛应用；对于CSE而言，这一点尤为根本。所调查的方法仅以有限的方式支持软件组件的运行时生命周期阶段建模（即，它们仅包括运行时基于组件的系统的工作负载和使用配置建模）。对于持续的性能改进，动态软件重构至关重要。然而，[12]所评审的主要研究仅部分支持CSE在性能方面的动态和自动化机制。此外，运行时的在线性能监控尚未与建模技术充分结合，因而无法对不断变化的使用配置和部署环境作出响应。作为对Koziolek [12], 发表的综述的扩展，我们在研究中对文献进行了更新的分析，包括截至2022年2月发表的论文。不同于 [12], 我们的工作聚焦于调查可应用于CSE上下文的性能工程方法，并考虑适用于各种系统的方法，而不是将研究限制在基于组件的软件系统上。

在随后开展的一项整体性文献综述中，Becker等人 [13]专门研究了基于2004年至2011年间发表的研究、面向自适应系统的模型驱动性能工程方法。作者提出了一套全面的分类方案，并以特征图的形式呈现。他们区分了反应式和主动式自适应策略，并归纳出自适应的两大类别：设计时和运行时。

自适应是指系统能够自主决定（即无需或仅需极少的人为干预）如何进行调整，以适应其上下文和环境的变化，并管理软件部署和执行期间所处环境中的不确定性[29]。自适应之所以得以实现，是因为自适应系统会显式表示自身的结构、行为和目标 [30]。近期的研究致力于探讨自适应的动因及其在实践中的应用 [31]。在此背景下，CSE 将持续工程定义为一种工程流程，旨在快速响应市场和客户提出的新需求，即构建能够更精准地

满足动态客户需求的解决方案 [32]。多种持续性活动（如持续监控、持续集成等）都是整体 CSE 的组成部分 [33]。CSE 与（自）适应是两种不同的运行时机制：自适应是系统管理变化和不确定性的能力，而 CSE 是一种动态流程，通过持续对系统进行工程化，使其能够添加新功能、功能和能力，或以更智能的方式实现这些功能和能力。

虽然上述研究（即 [13]）一致地概述了针对模型驱动的性能方法的自适应中的性能工程，但我们寻求将兴趣领域扩展到 CSE 中运行时知识与架构模型更一般的交织，而不将研究限制于模型驱动的性能方法。

最近，不同研究已在若干语境中涵盖了 CSE [14, 15, 16]的不同方面。然而，与我们的工作不同，它们并未专门考虑软件性能工程。Pahl 等人 [14] 已对 2014 年至 2015 年发表的 21 篇论文进行了系统映射研究，以识别、分类并系统比较微服务及其在云中的应用的现有研究主体，并将其置于持续开发的语境中。Taibi 等人 [15] 已对持续架构与微服务和 DevOps 进行了系统映射研究，并在其调查中纳入了 2014 年至 2017 年发表的 23 项研究。他们提供了微服务应用程序架构风格的概述，突出了每种风格的优缺点。然而，并未考虑非功能属性，例如性能。Jabbari 等人 [16] 对 DevOps 的分类进行了系统映射研究，并纳入了 2011 年至 2016 年发表的 49 篇论文。他们调查了 DevOps 在软件开发过程中的利用方式。他们发现，很少有主要研究利用模型驱动工程技术并聚焦于质量保证。

最后，Bezemer 等人 [34] 开展了一项工业调查，以深入了解在工业 DevOps环境中如何处理性能问题。具体而言，他们调查了执行性能评估的频率、所使用的工具、所获得性能数据的粒度，以及基于模型的技术的使用情况。

与上述论文相比，在本文中，我们执行了一项系统映射研究，该研究通过提供一个能够对主要研究进行分类的分类模式，调查了在CSE上下文中如何评估性能，这些主要研究涉及研究领域、所解决的目标问题、所提供的贡献、所设计的方法论、所研究的性能指标以及所使用的数据类型。与其他相关工作不同，本研究的重点是CSE中自适应和软件性能的结合。

3. Research Method

为深入了解 CSE 中针对软件性能的架构支持方面的当前研究实践，我们依据所提出的指南，对相关文献开展了一项系统映射研究该指南由 Petersen 等人提出 [17], , 并采用 Wohlin [35] 定义的“滚雪球法”。

本研究采用的过程由五个步骤组成，如图 2 所示。在第一步中，我们定义研究问题，并确定将在后续步骤中纳入综述的范围。随后，我们开展文献检索，以获取相关出版物列表，然后在论文筛选步骤中应用纳入与排除标准对其进行筛选。筛选出的出版物是数据提取的输入，在此过程中，我们通过考察其全文对相关出版物进行分类。最终，我们获得一个分类模式，并将其作为将从论文中提取的数据映射到问题的输入。

在以下章节中，我们描述上述映射过程的五个步骤。在第 4至第 8节中，我们展示对论文进行分析和映射的结果。此外，为简化本研究的可复现性，完整的复现包已公开提供 [36]。

3.1. 定义研究问题

为了研究科学界在 CSE 中为软件性能提供架构支持方面的贡献，我们提出了以下研究问题：

研究问题*1*： 研究领域 和 目标系统 有哪些已被研究？该研究问题有两个目的：*i)* 突出专注于为该领域提供解决方案的研究领域；以及*ii)* 提取应用程序或技术旨在应用于的主体系统（我们将其称为“目标系统” [37, 38]）。

该研究问题的依据与本研究的目标密切相关，旨在界定CSE中如何以及在何种程度上运用性能工程。它还确定了所选研究中所考虑的应用程序领域。这有助于我们了解持续性能评估的成熟度，并确定性能被视为关键约束的应用程序。

研究问题*2*：什么以及如何性能问题 得到了处理？。该研究问题聚焦于识别在 CSE 流程中所针对的软件性能工程问题，以及为解决这些问题而提出的解决方案。

软件性能工程中可以处理多个问题，包括需求规范、建模、分析、预测，以及改进软件系统性能的建议。该研究问题的依据与研究人员所考虑的软件性能目标问题的识别，以及在 CSE 上下文中提出的相关贡献，具有严格的关联。

RQ3：采用了哪些工具？可以使用若干工具来解决性能问题。我们将这些工具划分为三个关键词类别：输入数据、方法/技术以及性能输出度量/指标。第一类包括用于开展性能分析的数据类型，范围从运行时数据到需求，再到软件/性能

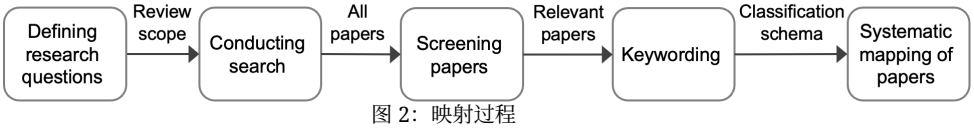


图 2：映射过程

模型。第二类的示例包括模式/反模式识别、性能预测或测试。第三类旨在确定目标指标，例如响应时间、吞吐量和网络带宽。本研究问题旨在确定在 CSE 上下文中应用了哪些工具，以及应用程度如何。

这一问题的依据与确定所提出解决方案在软件性能和 CSE 领域中的特征与局限密切相关。

RQ4：当前研究存在哪些空白 空白 以及对未来研究的启示？这一研究问题综合了前面三个 RQ 所强调的不同观点，旨在确定迄今研究最多和最少的语境。例如，在分布式系统这一目标中，作为研究领域的持续监控，围绕性能评估这一问题得到了多大程度的研究？

我们尤其关注突出强度较低的组合。我们预计，其中一些组合不会引起多少关注，而另一些则代表研究空白。此外，我们着重于确定值得在不久的将来开展研究的领域。

3.2. 开展搜索

搜索过程包括确定搜索字符串、概述最相关的文献来源和搜索词、制定纳入与排除标准，以及执行查询。

搜索字符串。我们根据问题结构中的PICO[1] 术语 [40]定义了搜索关键词。正如Kitchenham [40], 所建议的，如果研究重点是一般性调查，则比较和结果术语在软件工程中并不总是需要考虑。因此，我们从人群和干预术语中提取了关键词。

我们通过组合关键词并审查搜索结果的标题和摘要，优化了搜索术语和相关搜索字符串，以确保返回相关的研究。最终的关键词集列于表1中。然后将生成的查询调整为每个文献来源的语法。应用于不同文献来源的所有查询均报告在复现包 [36]中。

(“持续软件工程” OR “持续集成” OR “持续部署”OR “持续开发” OR “持续改进” OR DevOps OR “持续演进” OR “持续监控”) AND “软件架构” AND “软件性能”

查询 1：基线搜索字符串。

[1]PICO 要素包括：问题/患者/总体、干预/指标、比较、结果 [39]。

文献来源。我们根据Kitchenham and Charters [40], 的建议选择了一系列相关的文献来源，因为这些来源被认为是软件工程领域中最具代表性的，并且在许多综述中被使用。该列表包括：*ACM*数字图书馆、*IEEEXplore Digital Library*、*Scopus*和*Springer Link*。

纳入与排除标准。我们定义了适用于标题与摘要（T/A）或全文（All）的纳入与排除标准，如表2所示。

搜索。最后，搜索于 2022年3月1日进行，并纳入截至当时所有可用的出版物。应用搜索词后返回了 215篇论文，这是合并所考虑的书目来源中的论文后得到的结果，如图3左侧所示。移除重复论文后，我们获得了 195篇论文。

我们使用论文的“黄金集”验证了搜索字符串，以确保我们没有遗漏相关作品。黄金列表中考虑的论文是： [软件性能2], [软件性能9], [软件性能10],[软件性能12], [软件性能31],和 [软件性能49]。

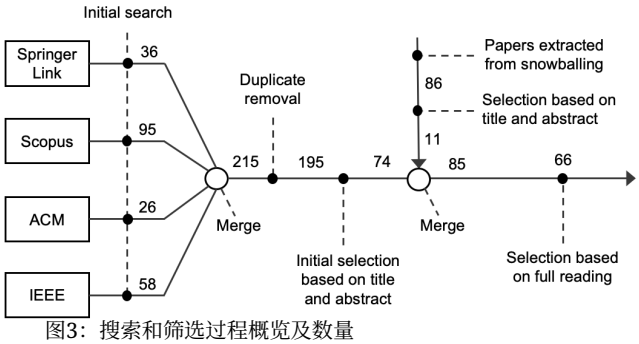


图3：搜索和筛选过程概览及数量

3.3. 论文筛选

获得初始论文集后，我们应用了本节所述的筛选过程。该过程的概览及数量如图3所示。

测试纳入与排除标 准的适用性。 在应用纳入与排除标准之前，所有作者都在一个由随机选取的 20 篇论文组成的子集上迭代测试了这些标准的适用性。基于分歧以及共同讨论，我们明确了纳入与排除标准。

将纳入与排除标准应用于标题与摘要。修订后的标准应用于剩余的195 篇论文（表2）。我们纳入符合所有纳入标准的论文，并排除符合任何排除标准的论文。每篇论文均由两位作者阅读；如有分歧，则由第三位作者协助解决分歧。对于32 篇论文，作者讨论并消除了可能存在的分歧。在最初的195 篇论文中，我们纳入了

Population	**P Terms**
Software Performance	software performance
Intervention	**I Terms**
软件架构	软件架构
持续软件工程	DevOps、持续集成、持续部署、持续开发、持续改进- 进、DevOps、持续演化、持续监控

表1：关键词定义

标准	**评估标准**	**Step**
纳入	论文涵盖软件性能工程问题	All
	论文提出了面向 CSE/DevOps 的基于模型或架构方法，或对以软件性能为目标的（自）自适应/重构作出贡献的软件性能	All
Exclusion	论文并非完全以英语撰写。	T/A
	该论文未经同行评审（即博客、论坛等）	T/A
	该论文为重复论文（仅考虑最新版本）	T/A
	该论文是一篇立场论文、书籍章节或工作计划（即，该论文未报告结果）	T/A
	该论文并不完全或部分聚焦于软件性能	All
	该论文并未完全或部分聚焦于软件架构或软件工程	All

表2：纳入与排除标准

74 篇基于标题与摘要的论文。第三位作者参与之前的评审者间一致性为0.75，使用科恩卡帕系数计算得出，这表明作者之间具有较高的一致性 [41]。

滚雪球法。我们通过滚雪球式检索过程 [35]，考虑检索到的论-文中列出的所有参考文献，并评估所有引用检索到的论-文的论文，最终得到另外一篇相关论文。我们对通过初始搜索检索到的论文采用了相同的过程。在2022年3月进行了滚雪球式检索。我们识别出 86 篇潜在论文，但只有 11 篇被纳入（在将纳入与排除标准应用于标题与摘要之后），以组成最终的出版物集合，并对其进行全文阅读和数据提取。

全文阅读。剩余论文的筛选工作由85 位作者独立完成。我们确保随机分配论文，使每位作者分配到的论文数量大致相同。此外，我们对分配进行了调整，以确保每一对作者之间的工作量均衡。

我们阅读了85篇论文全文并应用了表2中定义的标准。为了提高我们研究的可靠性 [42]，我们寻求了第三位作者的服务，针对两篇论文做出最终决定。在这种情况下，第三位作者参与之前的评分者间一致性很强（科恩卡帕系数= 0.94；几乎完全一致）。基于此过程，我们总共选择了66 篇论文进行综述。

3.4. 数据提取与分析

为确保数据提取过程的严谨性，并便于管理所提取的数据，本节对一个结构良好的分类框架进行了严谨设计，具体说明如下。

为了回答我们的研究问题，我们从66 篇入选论文中提取了一组信息。值得注意的是，我们通过遵循系统化的方法，在研究中定义了主要概念及相应的数据。

一个称为关键词标引 的过程。该过程的目标是有效地制定一个分类方案，使其适用于入选论文，并将其研究重点纳入考量 [43]。具体而言，我们通过以下两个步骤中的半自动化过程确定了编码方案的代码：

1. 论文中最常出现的关键词的自动识别。我们使用自然语言处理（NLP）技术，自动识别所选论文摘要中最频繁提及的关键词。我们首先从一个数据集中收集摘要，该数据集构成了处理所用的文本语料库。语料库分两个阶段进行预处理：噪声去除 和标准化。在噪声去除阶段，我们通过将文本转换为小写，并删除标点符号、标签、特殊字符和数字，进行了初步清理。随后，我们应用了两种标准化技术：使用词干提取 去除后缀，以及使用词形还原 将具有相同词根的词归为一组。作为最后的预处理步骤，我们删除了介词、代词和连词。这样，我们通过为文本构建词袋模型，生成了一个词频向量。在该模型中，不考虑词语顺序和语法信息，因为整个文本由其词语的多重集合表示，并可据此推导出各词的出现次数。随后，使用词频向量获取出现频率最高的50个单词，以及由两个单词（二元词组）和三个单词（三元词组）组成的组合。

2.关键词的手动细化。我们通过阅读每篇论文的摘要，细化了关键词和概念的收-集。我们将不-同论文中的关键词结合起来，以形成对研究性质和贡献的高层次理解。这帮助我们定义了一组能够代表研究问题的关键词类别。然而，论文摘要过于有限，无法定义所有有意义的关键词。

因此，我们彻底检查了论文的所有章节，以整合我们的分类方案。在第一轮评审之后，我们通过重新随机分配评审者（论文作者中的成员）进行了两轮评审。最后，在获得一组整合后的关键词后，我们重新组织了原有类别，从而得到下文所使用的最终分类。

我们为每位作者分配了一组随机选择的 10 篇论文，以验证编码方案和关键词，并确保研究人员之间有共同的理解。随后，我们讨论了编码结果和可能的不一致之处，并最终确定了该方案。

生成的分类框架如表 3 所示。它包含七个类别，并对提取的相关关键词进行了分组。每个关键词的详细描述见附录 C。每个类别通过使用下文详细描述的指标来解决相应的研究问题。

对于 RQ_1，，我们提取了关于研究领域和目标系统的信息。研究领域和目标既可以是完全研究的，也可以是已部分研究的。论文的主要目标包含在完全研究的研究领域和目标中。

在已部分研究的研究领域中，要么考虑论文的主要目标，要么它是支持主要目标的次要领域（而已部分研究的目标意味着该研究可以支持该目标系统，即使它在论文中未被描述为中心）。例如，我们考虑了诸如 [软件性能 40], 的论文，其中 持续软件工程 已部分研究，因为该论文主要关注 软件性能工程。需要注意的是，论文可能完全研究一个领域并部分研究另一个领域，因此每篇论文可能被分配多个研究领域。

关于RQ_2，入选论文所探讨的问题，我们提取了入选论文报告的主要问题、次要问题以及贡献信息。例如，对于论文 [SP58], ，我们将性能需求 视为主要问题，将服务质量 视为次要问题；而对于论文 [SP45]，我们将性能分析方法 视为主要贡献，将性能建模方法 和性能预测方法 视为次要贡献。我们也对RQ_3采用了相同的方法。

从入选论文中提取所有信息后，我们通过统计每个数据组和指标对应的论文数量来分析数据（见表3）。因此，对于（RQ_1），我们统计了已充分研究某主题、已部分研究该主题或涉及该主题（无论是充分还是部分研究）的论文数量。同样，对于（RQ_2），我们统计了涉及各项研究问题（主要、次要或两者兼有）的论文数量。按照前述方法，我们通过统计已充分评估和已部分评估的指标、所用数据及方法的数量，对（RQ_3）进行了分析。

对于前三个研究问题，我们考察了各个关键词的结果。研究问题4 旨在观察不同关键词的结果，以便找出那些

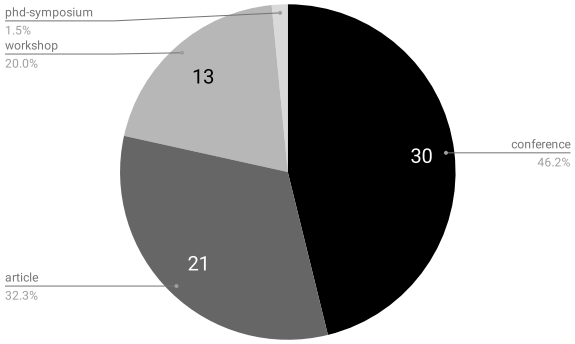


图4：入选论文类型。

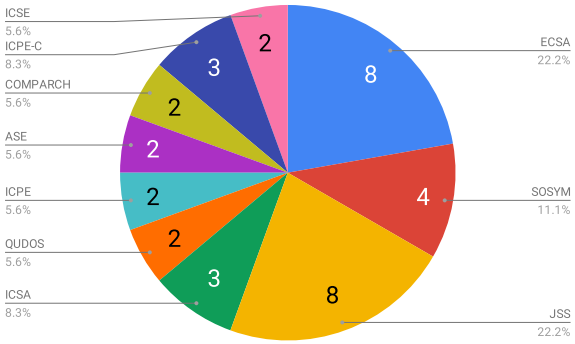


图5：各发表场所的入选论文。

鲜少受到研究的语境。为此，我们引入了气泡图，以便直观比较某一语境相较于其他语境受到研究的程度。在图中，气泡的大小表示在某个研究领域（x 轴）与领域（y 轴）的交叉处，研究特定关键词的论文数量。每个气泡旁都标注了论文的确切数量。这样，便可以直观地识别图中没有气泡或气泡较小的区域，从而确定在所考察的文献中，哪些研究领域与目标系统的组合很少涉及某个关键词。此外，我们还结合分析了入选论文中提及的前述结果、未来改进方向与研究方向，以确定一系列对未来研究的启示。

表3列出了分析所使用的完整关键词和指标清单。

4. 结果概述

我们选取了66篇同行评审出版物，其中包括21篇（32.3%）期刊论文、30篇（46.2%）会议论文，以及15篇（21.5%）研讨会（或其他附属活动）论文，如图4所示。入选论文发表于33个不同的发表场所。图5展示了至少有两篇入选论文涉及的发表场所列表。

入选论文显示，2016年到2022年间，人们对面向性能的持续软件工程的兴趣持续增长，而在2015年之前发表的出版物数量非常少，这与持续开发这一事实一致-

RQ	Categories	Keywords	Metrics
RQ1	Research area	软件性能工程，软件架构， 持续软件工程、DevOps、 持续监控，敏捷软件开发。	#已充分研究的主题 (F) #已部分研究的主题 (P) #已研究的主题 (F, P)
	目标系统	嵌入式 / CPS ，云 ，实时 ， 分布式、数据密集型 软件密集型 ， 基于组件的软件 / 微服务 / 面向服务的架构。	
RQ2	研究问题	Performance evaluation/assessment, 性能需求, 服务质量, 资源分配 / 部署, 不确定性.	#Main target problem/contributions (M) #Secondary target 问题/贡献 (S) #主要或次要目标 问题/贡献 (M, S)
	贡献	性能分析方法，领域特定语言， 持续工程框架， 性能建模方法、工具支持、 性能预测方法, 自适应.	
RQ3	Methodologies	性能模型、基于模型的工程 / 模型驱动工程（基于模型的工程/模型驱动工程），性能 反模式 / 根本原因 / 瓶颈检测, Performance prediction techniques, Performance analysis 技术, 参数依赖关系, 性能测试 / 负载测试 / 基准测试, 性能模型生成/提取，仿真， 机器学习，（多目标）优化。	#Fully evaluated 指标/数据/方法（F） #部分评估 指标/数据/方法（P） #完全或部分评估 指标/数据/方法（F，P）
	指标（输出）	响应时间、利用率、吞吐量、资源需求， 网络带宽，内存 / 内存泄漏。	
	Used Data (input)	运行时/ 受监控，工作负载，需求， 性能模型、软件模型、数据分析。	
RQ4	Research area and Target system combined with data 来自研究问题2和RQ3	研究问题1的所有关键词与研究问题2和研究问题3的特定关键词相结合。	#全面或部分解决 a specific combination 关键词（F, P）的

表3：分类框架：用于分析的数据提取、关键词和指标。蓝色关键词是在人工关键词精炼步骤中获得的。

ment和DevOps直到最近才出现 [7]。因此，我们可以得出结论：尽管 2020年 出版物的数量有所减少，但自 2015年 以来，通过持续工程解决方案支持软件性能的这一理念得到了加强。

and 2021.

主要研究的年度引用次数累计值（图6，蓝线，来源：GoogleScholar）表明，人们日益增长的兴趣不仅体现在出版物上，也体现在对数据集中研究的引用次数上。此外，自2016年以来，引用次数迅速增长。值得注意的是，整个数据集迄今累计获得了3000多次引用（即过去六年中引用次数增加了一倍以上）。这一结果表明，人们对本研究上下文的兴趣正在显著增长。

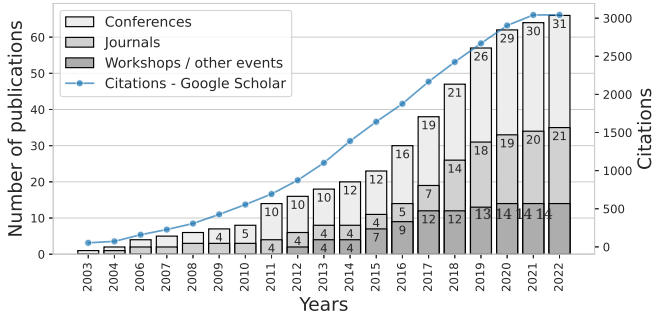


图6：各年度及按出版物类型划分的主要研究和引用的累计数量。

因此，尽管与性能工程相比，DevOps 和 CSE 领域相对年轻，但过去十年间已取得重大贡献，研究人员也日益活跃（图6）。

在以下章节中，我们呈现本研究的结果，旨在回答我们的研究问题（见第 3.1节）。对于每一条提取的信息，我们均报告定量数据以及对所得结果的解读。

5. 哪些研究领域和目标系统已被调查 (RQ1)

如第 2 节所述，本研究主题结合了多个方面，可能会引起不同学科的兴趣，也适用于不同范围的研究。为了概述这一研究主题，我们介绍了致力于提供解决方案的研究领域，以及已开发出解决方案的目标系统。

5.1. 研究领域

图 7 描绘了入选论文的主要**研究领域**，这些是其重点领域（每项研究可能贡献于多个领域）。图中的条形图比较了已识别的研究领域与论文数量。尽管我们的主题以该图所示的维度为特征，但入选论文贡献于这些特定研究领域这一事实并不明显。例如，一篇论文

可能只是使用了特定的性能评估技术，而并未在该领域做出贡献。对于每个研究领域，我们堆叠了两个不同的条形，以结合完全 和部分调查 的领域（如第 3.4 节所述）。

正如预期，入选论文主要聚焦于软件性能工程（66篇论文中的48篇）和软件架构（36篇论文）领域，这与根据发表趋势获得的发表场所信息一致。在这48篇论文中，有28篇（其中14篇完全聚焦于此）同时研究了软件性能工程和软件架构，也就是说，它们针对软件系统中基于架构的性能工程提出了具体贡献。更具体地说，在这28篇论文中，有14篇（完全或部分地）深入使用了架构/软件模型；其中一些致力于基于模型的软件工程（[SP8], [SP63], [SP11], [SP14], [SP58], [SP9], [SP12], 和 [SP29]），另一些则聚焦于模型驱动软件工程（[SP48], [SP46]，[SP53], [SP61], 和 [SP26]）。

许多研究为持续软件工程作出了贡献（38篇论文）。这些研究至少考虑了第2节介绍的一个持续维度，即持续集成（8篇； [SP23], [SP8] [SP43], [SP11], [SP60]，[SP57], [SP59], 和 [SP62]）、持续部署（5篇；[SP53], [SP59], [SP30], [SP25], 和 [SP61]）、持续开发（3篇； [SP59], [SP61], 和 [SP15]）、持续改进（17篇； [SP4], [SP8], [SP23], [SP63], [SP40], [SP43]，[SP11], [SP60], [SP57], [SP3], [SP53], [SP48], [SP22], [SP59]，[SP62], [SP32], 和 [SP36]），以及总体上聚焦于CSE（10篇； [SP48], [SP22], [SP49], [SP59], [SP62], [SP24], [SP28]，[SP37], [SP61], 和 [SP15]）。研究还特别关注持续监控（31篇论文）；因此，我们决定将其与其他持续维度分开列示。

Nineteen 论文重点关注以下领域：软件性能工程、软件架构 和持续软件工-程，这意味着它们不仅考虑了这些领域的方法论和技术，还为这些领域提供了科学贡献；因此，它们被置于这一主题的背景下。此外，15篇论文包含了软件性能工程、软件架构 和持续监控的组合。

Only *nine* 入选论文涵盖了 *DevOps* ，仅有 *five* 篇论文关注在敏捷开发 开发过程中性能工程的架构支持。其中， *three*篇论文结合了 软件性能工程、软件架构和 *DevOps* （[软件性能50], [软件性能62], 和 [软件性能61]），而*three* 篇论文结合了软件性能工程、软件架构 和 敏捷开发 （[软件性能8], [软件性能63], 和 [软件性能15]）。

对于每个领域，已充分研究与部分研究之间的比例关系较为均衡（即，有相当比例的领域已充分研究）。然而，仅就持续软件工程而言，部分研究该领域的论文数量有所增加。

5.2. 目标系统

图8描述了入选论文所针对的具体系统类型（作为案例研究）。这些关键词并不旨在

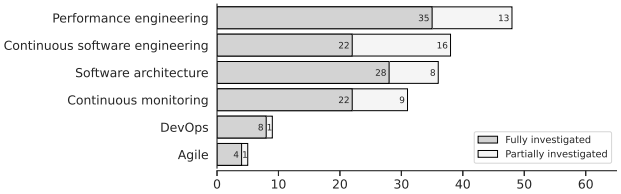


图 7：研究领域 - 结果

彼此并不互斥，即一个系统例如可以同时是分布式系统和实时系统。我们基本上确定了每篇论文中系统的主要特征，这些论文的方法/解决方案已应用于（或可能应用于）这些系统，前提是这些特征已被明确识别。主要关注点是软件密集型系统（23篇论文），例如系统之系统（SoS）（[软件性能43]和 [软件性能19]）、汽车软件（[软件性能60]）以及业务流程管理（[软件性能8]）。

同样，大量应用程序（23 篇论文）属于 基于组件的系统（*CBS*），并且有几篇论文关注面向服务的架构（SOA）（[SP12]和 [SP17]）以及微服务（[SP41]和 [SP61]）。

此外，分布式系统 （16 篇论文）引起了相当大的兴趣。然而，少数论文涵盖了 嵌入式系统 （包括 信息物理系统（*CPS*）（7 篇论文）和 实时系统（6 篇论文））。一个新兴的应用领域与大数据及其管理和分析相关（数据密集型系统；7 篇论文）。

关于 完全- 和 部分调查的 目标之间的关系，观察到较大的目标（例如 密集型软件系统）由几篇论文代表，这些论文仅部分置于该上下文中。这可能取决于每篇论文可能被分配多个目标这一事实。例如， [SP60]完全涵盖了 信息物理系统领域，特别是汽车领域，并部分涵盖密集型软件

系统)。在其他情况下，一些论文并未提供针对特定目标的解决方案，而是以更一般的方式进行归类，例如 [软件性能32]的情况。

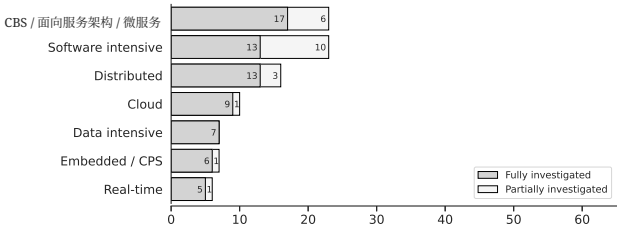


图8：目标系统 - 结果

5.3. Discussion

人们投入了大量关注于对数据进行持续监控，以实现持续开发、交付和集成，从而在自适应层面提升系统性能。由于监测数据能够支持性能分析，持续监控代表了提供 CSE 的一项基本能力。总体而言，结果证实，在自适应和软件性能领域，

CSE 是一个新兴主题，正逐渐受到研究人员和从业者的关注。现有关于DevOps的会议和系统综述表明，软件工程研究人员对这一主题有着浓厚的兴趣。尽管如此，专注于 DevOps 的论文仍然很少，这凸显了一个有待研究界解决的有趣空白。然而，关于敏捷开发的期刊论文数量有限是预期之中的结果。即使作为DevOps的先驱，敏捷开发也更加注重代码，并且产生的文档更少（例如，软件/设计模型），从而无法支持基于SA的SPE。

尽管大量入选论文已被充分确定为 CSE领域的贡献者，但部分研究这一领域的论文数量有所增加。此外，若干入选论文并未置于 CSE 或DevOps的上下文中，这可归因于这些论文并未明确将自身定位于这些领域，即便它们实际上提供了涵盖迭代开发过程多个方面的解决方案，其中更新是持- 续进行的。然而，这证实了这一新兴主题正在逐渐获得发展。

就所考虑的目标而言，基于组件的软件和密集型软件系统是研究最为广泛的对象。这些目标的特征在于存在不同的组件、服务或子系统。这些组件、服务或子系统可以彼此独立（例如，微服务），也可以像许多复杂系统中那样，彼此之间以及与环境之间具有强依赖关系和紧密关系。然而，在这两种情况下，异构组件的存在都使得有必要将这些活动集成到持续开发与维护中（例如，DevOps）。基于组件的开发范式建立在不同组件（例如，服务）内部复用这一概念之上，从而实现集成。一旦完成集成和实现，这些组件就必须进入持续维度，也就是说，需要了解并分析集成后的行为，而不仅仅是单个组件的行为；因此，它们在性能上下文中也同样重要。

接下来，结果展示了云和分布式系统的目标，这些目标以技术栈和基础设施的复杂性为特征。云节点的性能可能比其他节点差几个数量级 [44]。例如，如果在

对于关键任务服务或科学应用程序的托管，性能波动和可用性会成为关注重点，因此需要进行云监控，持续测量和评估基础设施或应用程序的行为（从性能、可靠性、功耗和安全性等方面），以便根据变化调整系统或采取纠正措施。通常，我们可以观察到，在以不受控请求为特征的开放系统中，持续工程对于支持其集成和演进至关重要，有助于识别此前未曾发现的其他问题。

最后，我们注意到，较新颖且具有创新性的目标尚未得到广泛研究。例如，对于数据密集型系统，大量使用大数据和机器学习要求对资源、性能和安全性进行高效管理。

Main findings:

- 持续监控数据受到高度重视，这代表了提供 CSE 的一项基本能力；
- CSE 和 DevOps 正在兴起：大多数研究提供了涵盖迭代开发过程多个方面的解决方案，其中更新是持续进行的；
- 软件密集型系统，尤其是基于组件的系统，是研究最多的对象：其异构组件要求在持续开发和维护中集成各项活动；
- 较新且更具创新性的目标（如云和数据密集型系统）尚未得到广泛研究。

6. 已解决哪些性能问题以及如何解决 (RQ2)

我们分类了 66 篇入选论文，通过考虑它们所解决的目标问题（我们将目标问题分为五类）及其研究贡献（我们确定了七种不同类型的贡献）。特别是，入选论文根据其研究方向和范围在主题上与至少一个目标问题和至少一项研究贡献相关联。我们详细概述了已解决的性能问题，并用示例论文加以说明。

6.1. 研究问题

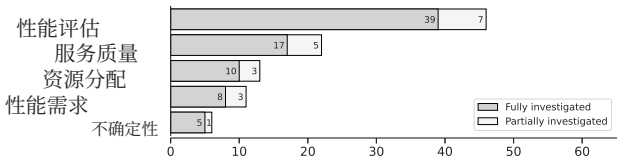


图 9：研究问题 - 结果

图 9 展示了入选论文所关注的问题（**目标问题**）。所得结果证实，大多数入选论文（46 篇论文，共 66篇）聚焦于性能评估。这些论文包括旨在评估或预测系统持续（再）工程过程中预期性能的方法和技术。例如， [软件性能63] 提出了一种在敏捷开发过程中持续集成性能模型的方法，通过优化参数依赖关系的学习来实现这一目标；而 [软件性能32] 提出了一种通过模型预测控制实现持续性能自适应的方法。

我们可以看到，22篇入选论文旨在提供服务质量，其目标是满足整体系统质量的不同需求/属性，包括保证一定水平的性能。例如，在 [软件性能37], 中，作者

提出了一种用于自动提取质量感知架构模型的方法，以探索设计时和运行时场景中的性能特性。在 [SP59], 作者提出了一种在DevOps范式中应用 QoS 感知型自适应应用程序的方法。除性能外，部分入选论文还为其他属性提供了具体支持，例如可用性([SP36]和 [SP2])、可靠性 ([SP12], [SP7], [SP44] 和 [SP13])、隐私([SP26])、可扩展性和弹性([SP41])。

此外，其中 13 篇论文考虑了资源分配。这是意料之中的结果，与我们的查询直接相关。在持续开发和运营过程中，资源分配是一项重大挑战，可能会严重影响团队快速／频繁交付的能力。例如，在 [软件性能31], 领域，研究人员使用资源配置文件检测企业应用版本的（性能）变化，并支持新部署的容量规划。在 [软件性能47], 领域，研究人员从运行数据的持续监控中提取性能模型，然后用它们模拟分布式和基于组件的应用的性能指标（资源利用率、响应时间和吞吐量）以及运行时成本。

此外，11 篇入选论文探讨了性能需求。近年来，随着系统日益复杂，高性能需求已不可避免。例如， [软件性能58] 领域提出了一种工具，用于发现满足成本和性能需求的安全架构设计方案；而 [软件性能45] 领域则提出了一种性能驱动的软件重构方法，以持续评估并采取行动，从而满足性能需求。

值得注意的是，映射研究揭示，不确定性是一个新兴的目标问题（6篇论文）。这些论文提出了有趣的解决方案，用于评估和预测系统的性能，即使是在不确定条件下也是如此。这些研究考虑了不同类型的不确定性，而这些不确定性不可避免地会影响质量评估的准确性。外部不确定性源于物理环境的复杂性和不可预测性（如在 [软件性能38]和 [软件性能36]中），以及非软件控制部分的复杂性和不可预测性（例如云网络 [软件性能13]）。其他论文（[软件性能25]和 [软件性能39]）同时提到了外部不确定性和内部不确定性，后者指的是源于

软件系统本身的复杂性（大型代码库、多个开发团队以及多样化的开发实践和标准等）。 [SP56], 中描述了对不确定性的不同表征，区分了参数、模型、代码和监测数据中的不确定性，并提供了一种在 DevOps 流程中检测性能需求违规的解决方案。

6.2. 研究贡献

图10描述了主要的**研究贡献**。根据我们的分析，我们可以说大多数研究通过性能分析、性能建模和性能预测 方法为系统的持续性能评估做出贡献。特别是，性能分析 方法，包括允许在系统持续工程期间进行性能分析的方法和工具，已在23篇论文中提出。例如，

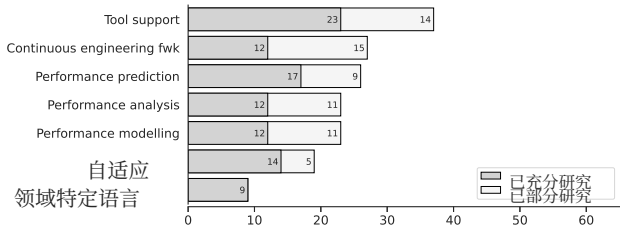


图 10：研究贡献——结果

[软件性能8] 提出了一种利用系统架构持续评估性能需求的方法。相比之下， [软件性能40] 提出了一种基于流程挖掘的性能分析方法。此外，23篇论文提出了有助于系统性能建模过程的方法（性能建模）。例如， [软件性能53] 提出了一种用于数据密集型应用程序的设计、质量评估和部署的统一建模语言配置文件，而 [软件性能34]则生成了用于数据密集型应用程序性能分析的分层排队网络。此外，26篇论文提出了性能预测方法。例如，[软件性能57]提出了一种基于模型的方法，该方法被集成到持续交付流水线中，从而能够预测内存管理和垃圾回收行为。相比之下， [软件性能32] 提出了一种通过模型预测控制进行持续评估和自适应的方法。

此外，我们还确定了 19 篇探讨自适应的论文，使持续运行的软件系统能够在变化的上下文中运行，同时满足其质量要求。大多数方法都采用性能评估，以实现软件系统中的预期服务质量目标。这些方法旨在提供例如可重新配置的软件架构（[SP5]和 [SP14]）、自适应微服务架构（[SP41]），或对运行中系统的性能参数进行自适应（[SP27]和 [SP12]）。相比之下， [SP11] 提出了一种自适应监控框架，用于研究基于组件的软件系统中的性能异常。值得注意的是，只有 [SP13] 部分解决了不确定条件下的自适应问题。具体而言，作者对云系统的行为和性能进行了建模与分析，以便在考虑资源和网络的不确定行为的情况下，自适应地处理最终用户请求。

一个新的研究方向正在兴起，它与领域特定语言有关；在所考虑的66 篇论文中，我们仅报告了9 篇入选论文。例如， [SP38] 扩展了混合架构分析与设计语言，以支持环境不确定性的建模，并针对各种性能查询进行了定量评估。在[SP50], 中，作者提出了一种领域特定模型方法，用于设计、部署和监控与大数据分析相关的应用程序的性能质量。 [SP46] 提出了一种领域特定语言，用于对基于会话的系统的工作负载规范进行建模，以开展负载测试和性能预测。

大量论文（37篇）提供了工具支持，以辅助性能工程研究。此外，相当数量的方法考虑了持续工程

框架（27篇）。工具支持和持续工程框架通常与其他研究贡献一并考虑。例如， [SP37] 提供了一种工具支持，可通过监控日志文件来提取架构性能模型；而 [SP23]则提出了一种持续工程框架，用于自动构建并参数化大型企业系统的性能模型。

6.3. 讨论

该领域的大多数研究旨在解决性能评估和评定问题。这是现代系统开发中的一个相关目标，其中日益增长的敏捷范式要求性能分析处于持续维度才能有效。

值得注意的是，发现22篇论文关注服务质量。其中，15篇论文使用了自适应方法，这意味着在18篇论文中的15篇，自适应方法针对的是服务质量而非仅仅性能。

管理系统行为的不确定性是一个新兴主题（解决此问题的论文是近期出版物），并且它横切于先前提到的目标问题。尽管拥有系统的完整模型代表理想情况，但在实践中，只有部分和有限的度量可用。因此，专门的性能分析或预测技术必须处理不确定的知识。所提出的研究及其对不同类型不确定性的讨论突出了相关问题并提供了新的研究思路。

超过一半的目标问题借助工具支持得到了解决，并且在性能预测、性能分析和性能建模之间分布均衡。其他目标问题则专注于自适应方法，其次是领域特定语言。这一结果表明，尽管研究涉及大量异构运行时数据，但很少有研究利用抽象来实现持续性能控制。提高规范的抽象层次有助于进一步提升自动化程度和互操作性。

最后，在大多数研究中（63 篇论文中的 52 篇），CSE 通过专用工具或框架得到支持。这一结果颇有意义，因为在本研究的背景下，质量与性能需求要求提供持续工程框架或专用工具的支持。

主要发现：

- 为在持续开发中提供适当支持，性能预测、性能分析和性能建模得到了进一步研究；
- 已有相当数量的自适应方法被提出，以确保服务的质量（包括性能）；
- 质量与性能需求需要持续工程框架或专用工具的支持。

7. 已采用哪些工具（RQ3）

性能分析可以采用多种技术来进行，这些技术使用不同的目标输出指标，并依托不同类型的输入数据。本节旨在识别研究背景下更常采用的工具。

7.1. 输入数据

图11(a)报告了输入**数据**。在 CSE 方面，系统会受到持续监控，并将性能指标反馈到支持预测分析的性能模型中。

总计61 篇和32 篇论文（共66篇）分别使用运行时/受监控和性能模型作为输入数据。将性能模型作为输入数据的论文数量较少，因为性能模型仅在过去几年才被纳入软件工程流程。而运行时性能评估和修复长期以来一直被视为常见做法。

使用监测数据的论文中，大多数（34 篇）还提出了持续监控性能特征的方法，并主要将其用于分析和预测（例如， [SP11], [SP62], [SP41], 和 [SP28]）。相比之下，其他研究采用了仿真（[SP38]）、系统执行建模 [45] （[SP43]），以及利用执行数据的性能模型生成器（例如， [SP57]）。然而，使用性能模型的论文主要采用统一建模语言 +MARTE（例如，[SP40]和 [SP38]）、排队网络（[SP8], [SP34]和 [SP36]）或 Palladio 组件模型（[SP46]和 [SP57]）。

在使用运行时/监测数据的61篇论文中，只有 $five$ 篇论文未提供关于它的明确信息，同时采用了性能模型。特别是， [软件性能8] 考虑了改变非功能需求，并提供了一种使用系统架构持续评估性能需求的方法。 [软件性能58] 提出了一种在架构设计中优化性能、成本和安全性的方法，并执行了静态分析（即使用污点分析）以识别架构级漏洞。 [软件性能36]分析了不同且相关的服务质量模型，从而在持续重新架构系统时减少了整体不确定性。 [软件性能54] 提出使用性能单元测试来探索性能特征，并在软件系统的整个开发过程中持续检测潜在的性能问题。 [软件性能19] 定义了一种基于系统交互和影响的事件驱动架构建模来预测系统性能的方法。

相当数量的论文（14篇）将软件模型视为流程的输入。这可能是因为表示软件模型时通常采用不同的记法，阻碍了在性能评估任务中实现全面自动化集成。大多数将软件模型作为输入的论文已充分研究了这一方面；其中一些研究（[SP61]）提出了基于模型转换的自动化技术，旨在将软件模型打造为软件工程流程中的一等工件。此处的结果还表明，数据分析在该领域尚未得到广泛关注（共9篇论文，包括 [SP50],

[SP41], 和 [SP21]）。然而，人们对数据分析的兴趣日益浓厚；因此，预计未来几年数据分析将成为主要的输入来源。对于每个输入数据点，已充分研究与已部分研究论文之间的比例大致相当（相当一部分方法和技术已得到充分研究），但上述软件模型和数据分析除外。对于后者，已部分研究的论文数量相对较多（*five*以上 *nine*），很可能是由于性能数据分析技术近期取得了进展。

7.2. 方法论和技术

图11(b)展示了入选论文中使用的方法/技术。正如预期，大多数论文聚焦于性能建模（66篇中的37篇）和性能分析（66篇中的32篇）。为了在生命周期早期解决性能问题，有必要构建并分析模型，许多论文旨在解决这一方面的问题便证实了这一点（ [SP24],[SP56], 和 [SP55]）。基于模型的软件工程和模型驱动工程技术也得到了广泛考虑（26篇论文），因为它们并未将模型的采用限制在性能领域。在若干情况下，模型也在软件工程领域中被视为（一等公民）（[SP63], [SP48],和 [SP14]）。相当数量的论文涉及性能模型提取和性能测试技术（均为23篇论文），这些技术通常用于研究现有运行软件系统中的性能问题（[SP46]和 [SP30]）。最后可以观察到，尽管在过去几年中，在CSE的上下文中，机器学习和多目标优化技术的采用已扩展到不同领域，但它们仍仅得到有限考虑，因为每种技术分别只有*six* 篇论文关注。对于每种方法/技术，已充分研究与部分研究之间的比例关系是相称的（即相当比例的方法/技术已得到充分研究）。只有在性能分析方面，部分研究该方法论的论文数量有所增加， [SP62]和 [SP32], 便是这种情况，它们涉及Palladio组件模型和排队网络。值得注意的是，尽管如此，仍然出现了这种情况，即

性能建模 已得到充分研究。在某些语境中，由于缺乏系统测量数据和参数值，深入开展性能分析可能较为困难。因此，在这种情况下，可以充分研究性能建模，但在论文的各项贡献中，分析仍处于次要地位。

7.3. 输出度量指标

图11(c)展示了目标输出**度量指标**。三项典型的性能指标，即响应时间 (40)、利用率 (32) 和吞吐量 (17)，是所考察论文中最常被关注的指标。尽管一方面这可以视为意料之中的结果，但另一方面，内存 (9)和网络带宽 (8)在此上下文中的研究明显较少，这多少令人意外。这两项度量指标在现代异构分布式软件系统的性能评估中可能发挥着至关重要的作用

因此，这一结果表明该方向缺乏研究。这些输出已充分研究。因此，无法对少数部分研究 的指标作出切实的考量。少数论文考察了多种类型的度量指标（[软件性能39],[软件性能30], [软件性能31],和 [软件性能12]）。

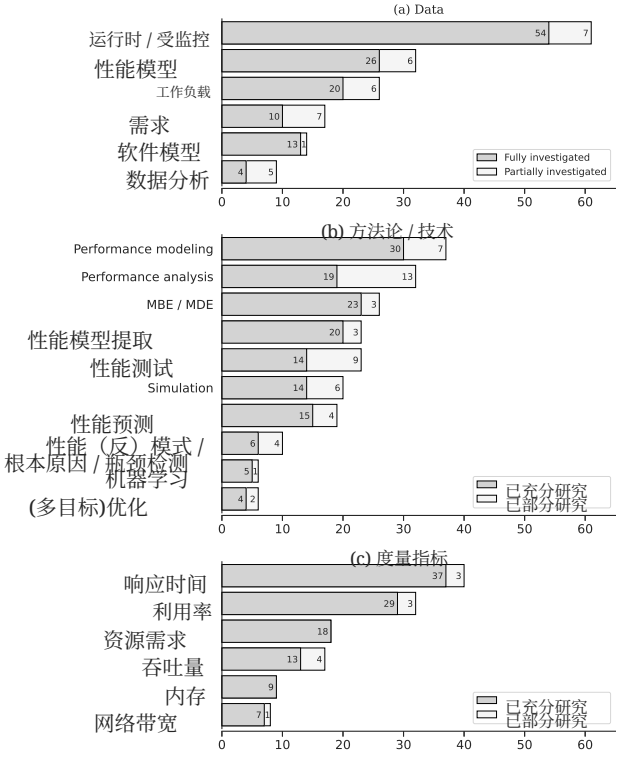


图11：数据、方法/技术、度量指标——结果

7.4. 讨论

我们分析了入选论文，并考虑到一种方法可以归类为三个要素：输入数据、方法论/技术和输出度量指标。并非所有入选论文都明确遵循这一模式，但获得的结果为这三个要素的组合频率提供了一些有趣的见解。

基于对结果的直接观察，可以看出，以运行时/受监控 数据为输入，并生成响应时间 和利用率指标作为输出的性能建模 和分析技术，迄今为止受到了最多的关注。这是对性能分析师在CSE 流程上下文中预期必须开展的工作的相关确认。然而，令人意外的是，无论采用何种技术，需求 和数据分析很少进入流程，以内存 和网络带宽为目标。这凸显了研究的缺失，尤其是在分布式异构软件系统（信息物理系统、边缘计算、IoTT 和数据密集型系统）领域，这一点至关重要。一方面，数据分析变得越来越容易获得，性能需求也越来越严格；另一方面，传统性能度量（例如响应时间和利用率）在

隔离并不能提供系统性能表现的整合视图，而系统性能可能由于内存使用不当和网络连接不佳而下降。

Main findings:

•以监测数据为输入并输出响应时间和利用率指数的性能建模和分析技术方法，是广泛使用的方法。需求和数据分析很少参与该过程，以设定内存和网络带宽目标；

• 尽管机器学习和多目标优化技术日益受到研究，但它们在 CSE 上下文中仍鲜少受到关注；

• 基于模型和模型驱动技术得到广泛关注，因为它们并不将模型的应用局限于性能领域，但在若干情况下，也会考虑软件模型。

8. 当前研究空白与未来方向（RQ4）

在本节中，我们旨在通过可视化每个感兴趣关键词对应的研究领域与目标系统交集中的论文数量，来检测潜在的研究空白。在下一节中，我们将按以下关键词类别讨论我们的发现：目标问题、研究贡献、所使用的方法论和技术、所使用的性能度量和指标以及输入数据。为呈现我们的结果，我们按照第3 节所述，绘制了气泡图。此外，我们还基于所分析的研究空白以及入选论文中描述的未来方向，讨论其对未来研究的影响。

8.1. 目标问题

图 12 和图 13 分别展示了性能评估 和不确定性 目标问题的气泡图。这些图呈现出截然不同的情况。许多研究以性能评估问题为目标，尤其是在某些领域，而很少有论文普遍考虑不确定性。造成这种情况的原因可能是，不确定性直到最近才作为一个独立的关注点出现在软件工程中，并且在持续工程实践中的应用仍然非常有限。

相比之下，由于性能评估这一一般性问题正是本研究的具体目标，因此预计会有更多论文对其进行考察。从图12可以明显看出，某些研究领域（持续监控、DevOps 和敏捷开发）从未与某些目标系统（实时系统）相交，
嵌入式系统和信息物理系统）在开展性能评估时的情况。这可能仅仅是因为 DevOps 和敏捷实践在这些系统中的采用率较低。

软件密集型系统的开发领域还存在进一步的空白，即使由于采用云计算等新技术，该领域仍在持续演进，物联网和人工智能；持续工程和 DevOps 可以借助新的性能工程解决方案，实现更广泛的软件应用（例如在智慧城市、智能制造和智慧出行领域）。

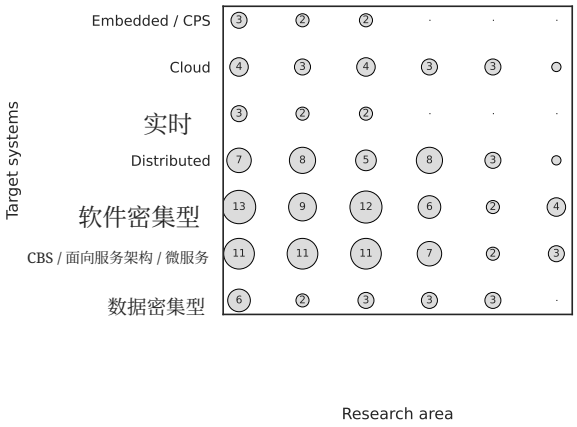


图 12：在研究领域与目标系统交汇处研究关键词 性能评估（目标问题） 的论文数量。

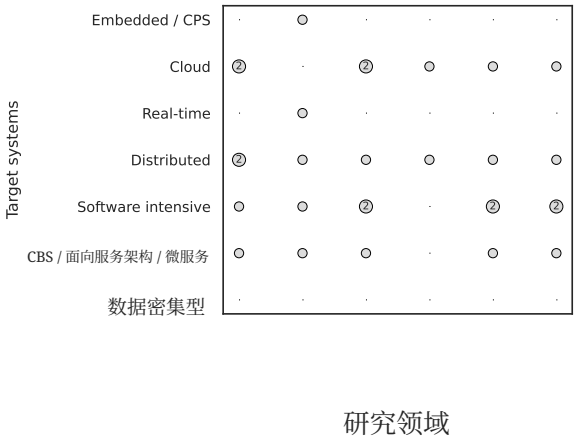


图 13：在研究领域与目标系统交汇处研究关键词 不确定性（目标问题） 的论文数量。

8.2. 研究贡献

在已确定的多项研究贡献中，我们报告了持续工程框架（图 14）和性能预测（图 15）的分布图，因为我们认为它们与本研究的目的相关。在图 14 中可以观察到，大多数提出新型持续工程框架的论文集中在图的下半部分。这些框架主要针对的目标系统是 CBS/SOA/微服务、软件密集型系统和分布式系统。可以预见，CSE 是该领域中最受关注的研究领域。然而，持续工程框架很少或从未在实时和嵌入式系统中提出。

图 15 显示，只有少数论文提出了嵌入式、实时和数据密集型系统中的性能预测方法。此外，似乎只有三篇论文同时关注 DevOps 和敏捷开发。这可能代表了一个关于

将在未来几年逐步填补，因为基于人工智能和机器学习的方法，例如 AIOps [46] 领域中的方法，正迅速涌现，成为一种性能建模与预测的新方式，而且能够更轻松地融入当前的 DevOps 实践中。

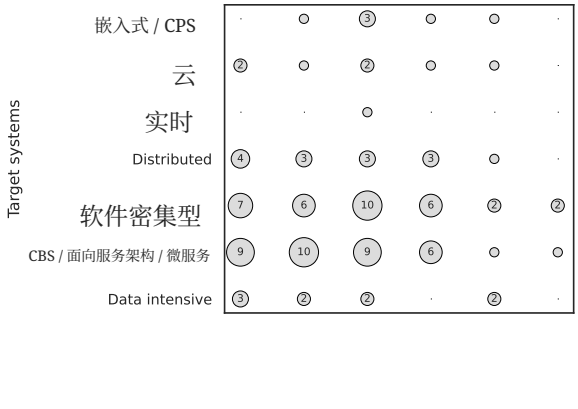


图 14：调查关键词持续工程框架（研究贡献）在研究领域与目标系统交叉处的论文数量。

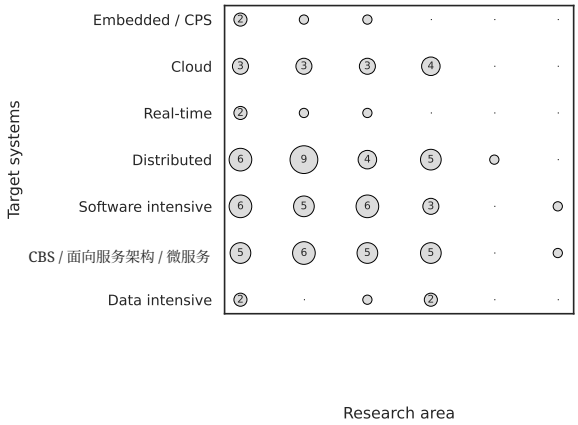


图 15：调查关键词性能预测（研究贡献）在研究领域与目标系统交叉处的论文数量。

8.3. 输入数据

用作这些方法输入的数据类型在确定一种方法可应用的情境以及启动该过程所需的信息类型方面发挥着重要作用。工作负载 （图 16）和需求（图 17）是两种数据类型，对它们的考量似乎与特定的目标系统相关。例如，虽然在云、分布式和*CBS/SOA/*微服务 系统中通常会考虑工作负载，但在嵌入式、实时和数据密集型 系统中几乎从不考虑工作负载。嵌入式系统和实时系统中未考虑工作负载是意料之中的，而在数据密集型系统中，这则提供了一个研-究机会。在考虑需求的使用时，我们面临着不同的情况。从数量上看，

在气泡图中的论文里，可以看出 *DevOps* 和敏捷开发在评估性能时并未过多强调需求；这出乎意料，因为它们在其流程中会考虑需求规范。此外，软件密集型和*CBS/*面向服务的架构*/*微服务 系统通常将需求视为性能工程方法开发的起点。

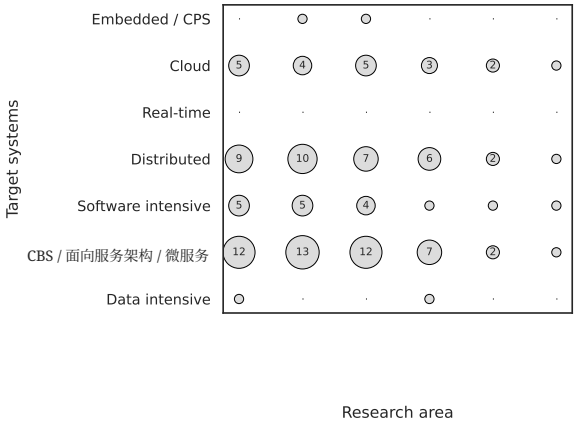


图 16：输入数据 - 工作负载

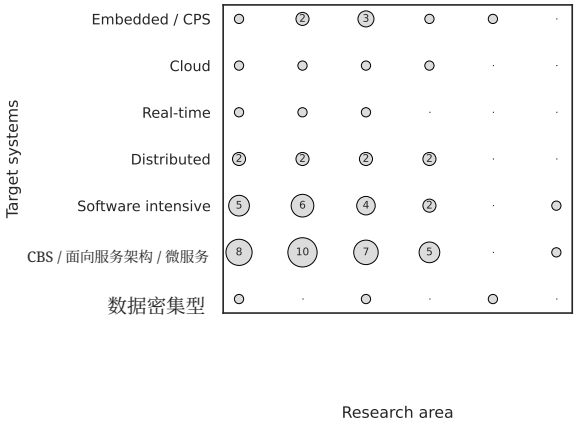


图 17：输入数据 - 需求

8.4. 方法论和技术

本研究所采用的方法论和技术为了解当前研究兴趣和发现研究空白提供了有力的信息来源。例如，考察性能模型的使用情况（图18）时可以明显看出，在论文数量方面，DevOps 和敏捷开发研究领域落后于其他领域。图19所示的性能测试则呈现出不同的情况。在这种情况下，虽然各研究领域的占比几乎相同，但实时和嵌入式系统领域对性能测试、负载测试和基准测试的采用明显缺乏关注，因为针对上述方面的实时嵌入式系统论文数量分别为0篇和3篇。性能模型的生成或提取

（图20）是采用性能模型的一种特殊用例。因此，DevOps 和敏捷开发研究领域仍只有少数论文，这并不令人意外；而分布式系统和微服务系统似乎最依赖性能模型自动生成。最后，关于仿真的使用（见图21），我们发现，与其他方法论相比，仿真已应用于多篇有关实时和嵌入式系统的论文。此外，与其他研究领域相比，DevOps 和敏捷开发领域的代表性似乎较低。

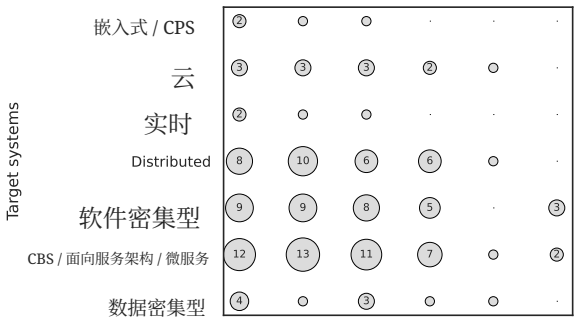


图 18: 方法论 / 技术 - 性能模型

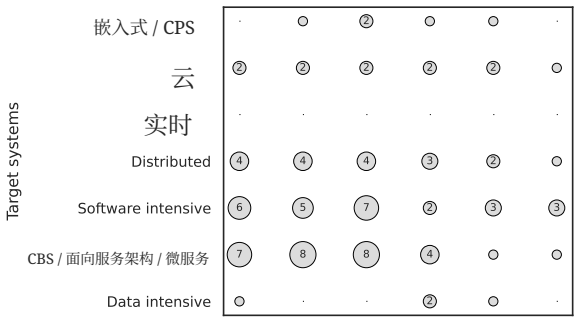


图 19: 方法论 / 技术 - 性能测试 / 负载测试 / 基准测试

8.5. 输出度量指标

在调查入选论文中作为输出目标的性能度量和指标时，我们观察到考虑关键词 响应时间 （图22）和 内存/内存泄漏（图 23 ）的论文数量存在两种截然不同的情况。在软件性能工程中，响应时间和内存是主要关注的度量之一，因为它们通常分别用于评估服务质量和运营成本。尽管如此，内存在入选论文中很少被考虑

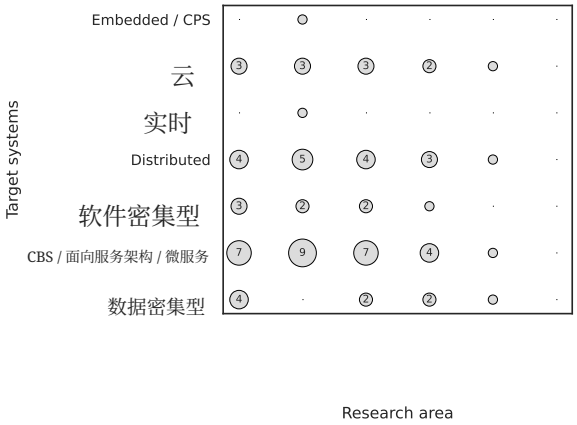


图 20: 方法论 / 技术 - 性能模型生成 / 提取

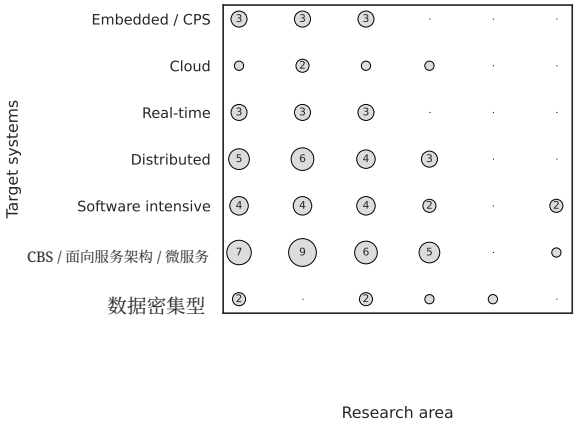


图 21：方法论 / 技术 - 仿真

本研究。更有趣的是，*DevOps* 和 敏捷 似乎在任何目标系统中都完全不考虑内存。我们预计，随着机器学习和数据密集型系统的持续增加，内存使用将变得至关重要。相比之下，响应时间在总体上被考虑得更频繁，特别是在分布式系统、软件密集型系统和*CBS*/面向服务的架构/微服务领域。这描绘了一幅图景：影响服务质量的性能度量是CSE相关研究中的首要关注点，从而导致在内存使用相关问题及其如何影响提供服务成本的调查中存在巨大差距。

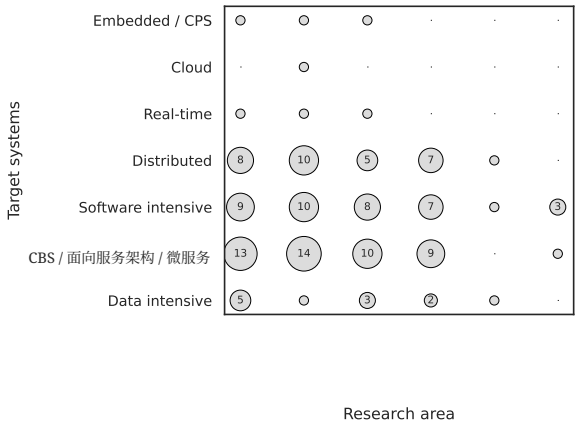


图 22: 输出度量指标 - 响应时间

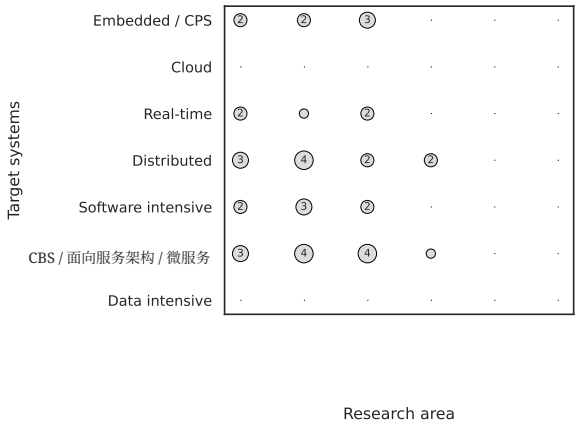


图23：输出度量指标 / 内存 / 内存泄漏

8.6. 关于研究空白的讨论

正如先前在第8节中报告的结果所清楚表明的那样，较为成熟的研究主题与新兴研究主题在覆盖范围方面存在明显差异。更为一般的性能评估主题就是一个明显的例子。它是我们研究中调查得最多的主题之一，这不仅是因为它理所当然地处于我们探究的核心位置，还因为它是一个发展历史跨越数十年的成熟研究主题。

然而，当同一主题与软件行业中较新的趋势（如敏捷开发和 DevOps）结合时，在涉及论文的数量方面，它显然远远落后。除了较新的主题可能在研究中受到的关注较少这一显而易见的原因外，更有说服力的解释可以从敏捷开发流程和实践的具体特点中找到。事实上，敏捷开发周期通常都很短，因为这类周期的主要重点可能是快速发布新功能。虽然在发布新功能之前，通常需要进行一些测试，例如产品负责人开展的验收测试，但性能测试通常并非必需，因为它们往往会耗费大量时间和资源。一种可能的解决方案是，将新需求划分为低性能优先级和高性能优先级，并考虑为性能分析和测试投入相当多的预算。

同样，云原生系统的具体需求并不总是与性能评估的要求相一致。大多数云原生系统采用持续追踪工具（Jaeger、OpenTelemetry 和AppDynamics），在高度分布式且动态的环境中记录执行追踪信息。这些工具可能有助于开展性能分析，因为它们通常会提供与性能相关的额外信息，例如远程调用的执行时间。然而，企业不会对分布式追踪信息开展性能分析，直到性能开始影响用户体验。持续评估和改进性能仍未被视为优先的质量任务，从而导致软件系统和服务的性能迅速下降 [47]。这使我们相信，开展一项考虑工业界的后续研究（多声部文献综述）可能会具有一定价值。

与敏捷开发、DevOps 和云相比，软件工程中的不确定性问题早已受到关注。即便如此，它在研究中仍是一个新兴主题，尤其是在考虑不确定环境下的性能评估时。在存在不确定性的情况下评估性能，本身就十分困难。由于诸多因素，软件系统的特征正变得更加动态：系统因迁移至云端或由第三方软件构成而更加分布式；面向更广泛的用户群体开放；更重要的是，系统需要在不断变化的环境中提供服务质量保证。在此上下文中，性能评估必须考虑波动的工作负载以及资源需求的巨大差异，这给长期预测和分析性能结果的准确性与有效性带来了更多挑战。为解决这一问题，研究人员已开始研究不确定性下的软件性能；我们的研究发现，这一研究领域尚未得到充分探索，但可能具有巨大的发展潜力。

Main findings:

- 尽管性能评估是一个成熟的研究主题，但将其与软件行业的最新趋势（如敏捷开发和 DevOps）结合起来，显然早就该做了。
- 研究人员已经开始研究不确定性下的软件性能。该研究领域尚未得到充分研究，但可能具有巨大的增长潜力。
- 没有证据表明企业在云系统上开展性能分析方面的经验。

8.7. 对未来研究的启示

在本节中，我们将讨论本研究的意义以及未来研究面临的挑战。

8.7.1. 迈向 *CSE* 中的质量文化

所得结果表明，人们对质量文化抱有兴趣，这意味着分析和验证会在持续软件工程流程中尽早开展（如DevOps 中的测试），从而通过协作式产品改进，更轻松地发现并修复缺陷。为了提升软件（架构）的质量特性并推动其持续改进（和重新架构），服务质量分析必须成为整个软件开发生命周期中的一项集成活动，这要求持续将质量特性呈现出来供分析（[SP36]和 [SP50]）。

如前所述，大多数入选研究都聚焦于少数几种性能属性。然而，在系统的持续工程过程中，需要加强对各种性能属性以及软件质量各个方面的支持。需要考虑的一些具有挑战性的质量属性包括：验证正确性属性，例如架构不匹配（ [SP50]）；评估架构运行时模型的保真度（[48, 49]）以及其对人工检查的实用性（[SP26]）；通过考虑影响因素来扩展可扩展性，例如实验中用户行为复杂度的变化 [SP61], 、支持状态管理和资源配置机制 [SP20], ，以及引入内存分配和释放运维的时间消耗，以提高模型的预测质量[SP57]。同样，其他服务质量属性，例如可靠性 [SP14], 、一致性 [SP12], 、安全性和安全性，以及可用性 [SP13] ，也可以在 CSE 框架中建模和分析。

8.7.2. DevOps 中性能工程的优势

DevOps 正在工业界得到广泛采用。然而，其快速变更、开发自动化和快速反馈循环等原则（通常依赖于动态云环境）与当前性能工程方法的复杂性相冲突 [34]。因此，应改进性能工程框架，以便在快速变化的系统中采用。现代系统经常变化，需要持续重新学习故障模型，以保持预测质量[软件性能44]。

此外，这些系统的特点是具有持续可用的数据流。性能模型应定期、增量甚至持续地构建，并由受监控环境中组件的变化触发。一旦检测到潜在问题，就可以仅使用最新数据快速重建模型，然后用于与先前的模型结果进行比较 [SP24]。

8.7.3. 数据驱动方法和机器学习

性能与负载测试会产生大量数据，可能难以分析。数据驱动方法能够为优化性能、构建新功能以及预防服务问题提供强有力的洞见，尤其适用于分布式（企业级）应用程序、内存数据库和大数据系统 [SP61]。

尽管现代软件工程技术不断发展，但目前尚无一套成熟的方法论，能够在持续开发中系统地运用性能工程和数据驱动工程。

此外，基于机器学习的性能评估可以成为持续工程流程 [软件性能52] 中不可或缺的一部分。性能模型必须能够自主学习，并在生产环境 [软件性能51] 中的系统运行期间不断自我改进。

8.7.4. 持续控制系统不确定性

正如第6节和第8节所述，本研究所选论文对不确定性的关注较少。然而，学术界和工业界都十分关注具有高度不确定性的情境。为了降低不确定性，并尽早获得关于产品/软件的反馈，尽快在短周期内检验假设和假说至关重要 [7]。

持续监控、频繁重新评估和重新架构对于降低 CSE 和 DevOps [软件性能56]中的不确定性至关重要。高效的分析方法可用于传播软件系统中不确定参数的影响，并计算性能指标的鲁棒性，从而增强应对不确定性的灵活性（[软件性能39]）。

8.7.5. 集成与抽象

一个反复出现的问题是需要将方法和工具集成到持续和DevOps流水线中。因此，有必要设计和开发能够与本研究 [软件性能29]上下文中所使用的其他工具和方法集成的性能工程方法。

持续监控是CSE中的一个基本过程。在对未来研究影响的上下文中，需要进一步调查能够自动更新系统和性能/质量模型的自适应监控和分析基础设施（[软件性能10], [软件性能11], [软件性能23], [软件性能62] 和 [软件性能12]）。

外部能力可以通过将系统表示为黑盒组件的方式进行集成，具体做法是集成

黑盒监控技术 [软件性能12], ，或创建描述特定企业应用的资源剖析（使用标准测量解决方案），而不是依赖定制解决方案来收集所需数据 [软件性能31]。通过支持收集有关被监控应用状态的任意信息，可以进一步改进这一做法；这需要动态地将相应类型集成到系统中 [软件性能17]。

使用更高抽象层次有助于减少集成工作。未来的研究可以着眼于开发一种基于模型的框架，该框架考虑（特定领域）语言和自动化机制的定义，以确保通过设计实现 CSE 中监控、分析、测试和仿真的潜力。如上文所述，机器学习支持的和数据驱动方法可用于（持续地）学习和调优模型。

8.7.6. 对从业者和研究人员的启示
从业者可以从所呈现的结果中受益，以了解软件性能工程和软件架构如何支持CSE和DevOps中的实践。此外，他们可以从适用于持续软件开发上下文中性能工程的架构支持的方法论分类中受益。

研究人员可以利用我们的结果了解研究趋势和研究空白，并更好地聚焦于未来的工作。特别是，软件性能社区可以借助这项工作了解其方法是否适用于 CSE/DevOps 上下文。此外，本研究有助于他们确定推动工具开发的需求，从而提高工业采用的可能性。相比之下，CSE/DevOps 领域的研究人员可以识别出哪些方法论和技术能够在开发周期的某个阶段帮助提升软件性能。

Main findings:

- 人们对质量文化的兴趣日益增长，其中质量属性在软件开发生命周期中持续接受分析和验证；
- 当前性能工程方法的复杂性应提高其对DevOps原则——快速变更、开发自动化和快速反馈循环——的适用性；
- 亟需建立一套方法论，以便在持续开发中系统地应用性能工程和机器学习/数据驱动工程；
- 持续监控以及频繁的重新评估和重新架构对于降低 CSE 和 DevOps 中的不确定性是必要的；
- 使用更高抽象层次（即通过基于模型的框架或（特定领域的）语言和自动化机制）有助于减少在系统中集成异构方法和组件所需的工作量。

9. 有效性威胁

系统文献综述结果可能会受到一些威胁的影响，这些威胁主要与调查的正确性和完整性有关。在本节中，我们根据Wohlin等人提出的指南确定了这些威胁 [50]：构念效度、内部效度、外部效度和结论效度威胁。此外，我们还确定了减轻这些威胁所需采取的行动。

9.1. 构念效度
构念效度与将研究执行exe-执行 [50]的结果推广到其背后的概念或理论有关。我们识别了与对所选研究进行潜在主观分析相关的威胁。正如 Kitchenham [40], 指南所建议的，数据提取由两名或更多研究人员独立完成；如有不一致，则由第三位作者参与讨论，以解决任何分歧。根据 Dybå 和 Dingsøyr [51]提出的方案，对每篇选定论文的质量进行了检查。

9.2. 内部效度
内部效度威胁与对处理和结果之间因果关系可能得出的错误结论有关 [50]。对于二手研究，内部效度体现了研究发现对文献中所报告内容的代表程度。

为应对这些威胁，我们严格定义了研究方- 案，包括数据提取表。所有作者首先通过从10篇随机入选论文中提取信息，对数据提取表进行了验证。考虑到数据分- 析过程，威胁很小，因为我们在处理定量数据时仅采用了描述性统计技术。

在考虑定性数据时，采用半自动化方法定义关键词，将其转换为定量数据。关于关键词定义，我们首先应用自然语言技术，以降低所选术语的主观性，然后以协作方式手动细化关键词。

最后，所有研究人员随机选取了10项研究，以验证结果是否一致，且不受执行提取工作的研究人员影响。在需要时，研究人员会对分歧进行讨论并协作解决。

9.3. 外部效度
外部效度威胁与推广结果 [50]的能力有关。在二次研究中，外部效度取决于所选研究的代表性。如果所选研究不具备外部效度，对其内容的综合就不会有效。在我们的研究中，我们无法评估所有纳入研究的外部效度。

为应对这一威胁，我们将搜索字符串应用于多个文献来源，包括 Springer Link、Scopus、ACM数字图书馆和 IEEE Xplore数字图书馆。使用不同的文献来源，使我们能够确保获取绝大多数论文。此外，我们还通过开展滚雪球活动来补充搜索。

仅纳入以英语撰写的论文可能会使我们的结果产生偏差。其他语言的研究也可能具有相关性。然而，我们只采用英语，因为英语是科学论文中使用最广泛的语言，因此我们认为与这一威胁相关的偏差极小。我们仅纳入同行评审论文，没有考虑灰色文献（例如技术报告、硕士论文、网络论坛等）。由于我们的目标是仅识别高质量的科学研究，因此我们认为这一威胁极小。

9.4. 结论效度

结论效度与根据结果得出的结论的可靠性有关 [50]。

其中一个威胁与可能遗漏某些研究有关。为减轻这一威胁，我们谨慎地采用了检索策略，在八个数字图书馆中进行检索，并结合滚雪球式检索过程，考察检索到的论文中列出的所有参考文献，并评估所有引用了已检索论文的论文，最终又发现了一篇相关论文。我们采用了范围较广的搜索字符串，因此获得了大量期刊论文，并能够纳入更多可能的研究结果。我们制定了纳入与排除标准，并首先将其应用于标题与摘要。然而，我们并未仅依赖标题和摘要，而是在根据标题与摘要决定接受某篇论文之前，先浏览其全文，并再次应用纳入与排除标准。

另一种可能威胁结论效度的因素与对结果的错误解读有关。为减轻这一威胁，所有作者都仔细审查了结果。不过，其他研究人员可能会给出不同的解读。

10. 结论

本文介绍了一项关于 CSE 中软件性能架构支持的映射研究。 在215 项相关研究中，我们选取了 66项主要研究，并对其进行了分析，以回答我们的研究问题。因此，我们深入探讨了研究背景，并为研究人员和开发人员应对与该主题相关的挑战提供了思路，其中包括这样一个事实：在这一背景下，知识空白和未来研究主题尚未得到充分调查。具体而言，我们通过分类框架分析了所选主要研究的发表趋势、研究领域和目标系统、目标问题与贡献，以及其他具体特征。

本研究表明，软件性能和自适应是 CSE 中得到充分考虑的方面，其中受影响最大的维度是持续监控和持续改进。本研究还表明，软件性能工程方法和方法论已经足够成熟（得益于特定框架和工具的支持），可以应用于持续实践，并且倾向于使用在运行时受监控的数据。总体而言，自适应被认为能够提供特定支持；在许多情况下，自适应模型被用作分析和预测性能以及架构参数和配置的输入。更多总体而言，分布式系统、基于组件的系统、面向服务的架构、微服务和软件密集型系统均已获得相关支持。其他语境（例如数据密集型系统或嵌入式系统）中的应用程序较少。最值得关注的研究空白出现在云系统以及需要研究不确定性的系统中。

致谢

本工作部分得到了Ulla Tuominen基金会（芬兰）的Adoms资助和芬兰科学院的MuFAno资助（资助编号349488）、ECSEL联合事业（JU）的AIDOaRt项目资助（资助编号101007350）、由Fondo Territori、Lavoro e Conoscenza CGIL CISL UIL资助的TerritoriAperti项目，以及由H2020-INFRAIA-2019年-1欧盟资助的SoBigData RI项目（资助编号871042）的支持。Daniele Di Pompeo得到欧盟—下一代欧盟—国家复苏与韧性计划（Piano Nazionaledi Ripresa e Resilienza，PNRR）项目的支持：“SoBigData.it——加强意大利研究基础设施以开展社会挖掘和大数据分析”——Prot. IR0000013——Avviso n. 3264 del 28/12/2021年。MicheleTucci得到OP RDE项目No. CZ.02.2.69/0.0/0.0/18053/0016976“查理大学研究、技术和行政工作人员的国际流动-_性”的支持。

References

[1] V.Ferme,C.Pautasso，迈向整体式持续软件性能评估，载于：第八届ACM/SPEC国际性能工程会议伴会论文集，ICPE ’17 Companion，2017年，第159–164页。doi:10.1145/3053600.3053636。 [2] C.-P. Bezemer, S. Eismann, V.Ferme, J. Grohmann, R.Heinrich, P. Jamshidi, W. Shang, A. van Hoorn, M. Villaviencio,J. Walter, F. Willnecker，DevOps中如何处理性能？一项关于工业实践的调查（2018年）。doi:10.48550/ARXIV.1808.06915。[3] G. Kudrjavets,J.Thomas, N. Nagappan，软件性能工程不断演变的格局，载于：2022年软件工程评估与测量国际会议论文集，EASE’22，2022年，第260–261页。doi:10.1145/3530019.3534977。 [4] F.Brosig,N. Huber,S.Kounev，分布式基于组件的系统的架构级性能模型自动提取，载于：第26届自动化软件工程国际会议（ASE ’11），2011年，第183–192页。[5] V. Cortellessa, A. DiMarco, P. Inverardi，基于模型的软件性能分析，Springer，2011年。 [6] A.Martens, H. Koziolek, S.Becker,R.H.Reussner，使用进化算法自动改进软件架构模型的性能、可靠性和成本，载于：第一届性能工程国际会议论文集（ICPE ’10），2010年，第105–116页。 [7] B. Fitzgerald,K.-J.Stol，持续软件工程：路线图与议程，《系统与软件杂志》123（2017年）176–189。 [8] M. Mazkatli, D. Monschein, J.Grohmann,A.Koziolek，具有参数依赖关系的架构性能模型增量校准，载于：2020年IEEE国际软件架构会议，ICSA 2020，巴西萨尔瓦多，2020年3月16–20日，IEEE，2020年，第23–34页。doi:10.1109/ICSA47634.2020.00011。[9] S.Spinner, J.Walter,S.Kounev，虚拟化环境中在线性能模型提取的参考架构，载于：A. Avritzer, A. Iosup, X. Zhu, S.Becker（编），ACM/SPEC国际性能工程会议伴会出版物，ICPE2016 Companion，荷兰代尔夫特，2016年3月12–16日，ACM，2016年，第57–62页。doi:10.1145/2859889.2859893。

[10] S. Spinner, J. Grohmann, S. Eismann, S. Kounev, 《面向自感知计算基础设施的在线模型学习》, J. Syst. Softw. 147（2019年）1–16。doi:10.1016/j.jss.2018.09.089。[11] D. Arcelli, V. Cortellessa, D. Di Pompeo, R. Eramo, M. Tucci, 《利用架构/运行时模型驱动可追溯性提升性能》, 载于: IEEE国际软件架构会议, ICSA 2019, 德国汉堡, 2019年3月25–29日, IEEE, 2019年, 第81–90页。doi:10.1109/ICSA.2019.00017。[12] H. Koziolek, 《基于组件的软件系统的性能评估: 一项综述》, Perform. Evaluation 67（2010年）634–658。[13] M. Becker, M. Luckey, S. Becker, 《自适应系统的模型驱动性能工程: 一项综述》, 载于: 第8届国际ACM SIGSOFT软件架构质量会议论文集（QoSA '12）, 2012年, 第117–122页。[14] C. Pahl, P. Jamshidi, 《微服务: 一项系统映射研究》, 载于: J. S. Cardoso, D. Ferguson, V. M. Muñoz, M. Helfert（编）, CLOSER 2016——第6届云计算与服务科学国际会议论文集, 第1卷, 意大利罗马, 2016年4月23–25日, SciTePress, 2016年, 第137–146页。doi:10.5220/0005785501370146。[15] D. Taibi, V. Lenarduzzi, C. Pahl, 《微服务与DevOps的持续架构设计: 一项系统映射研究》, CoRR abs/1908.10337（2019年）。arXiv:1908.10337。[16] R. Jabbari, N. B. Ali, K. Petersen, B. Tanveer, 《什么是DevOps? 关于定义与实践的系统映射研究》, 载于: XP2016科学研讨会论文集, 英国苏格兰爱丁堡, 2016年5月24日, ACM, 2016年, 第12页。doi:10.1145/2962695.2962707。[17] K. Petersen, S. Vakkalanka, L. Kuzniarz, 《软件工程系统映射研究指南: 更新版》, Inf. Softw. Technol. 64（2015年）1–18。[18] M. Tichy, J. Bosch, M. Goedicke, 《编者按》, 《系统与软件杂志》123（2017年）173–175。doi:https://doi.org/10.1016/j.jss.2016.09.010。URL htt ps://www.sciencedirect.com/science/article/pii/S0164121216301741[19] C. Ebert, G. Gallardo, J. Hernantes, N. Serrano, 《DevOps》, IEEE Software 33 (3)（2016年）94–100。doi:10.1109/MS.2016.68。[20] M. Woodside, G. Franks, D. C. Petriu, 《软件性能工程的未来》, 载于: 《软件工程的未来》（FOSE '07）, 2007年, 第171–187页。doi:10.1109/FOSE.2007.32。[21] J. Merseguer, W. Binder, J. Murphy, 《客座编者按: 软件性能工程自动化》, Autom. Softw. Eng. 24 (1)（2017年）71–72。doi:10.1007/s10515-016-0201-2。URL https://doi.org/ 10 1007 10515 016 0201 2 。 /s - - - C. Laaber, [22] 《持续软件性能评估: 在每次构建中检测软件库的性能问题》, 载于: 第28届ACM SIGSOFT国际软件测试与分析研讨会论文集, 2019年, 第410–414页。doi:10.1145/3293882.3338982。URL https://doi.org/ 10 1145 3293882 3338982 。 / . L. Bass, P. [23] Clements, R. Kazman, 《实践中的软件架构》, 软件工程SEI系列, Addison-Wesley, 2003年。URL http://books.google.fi/books?id= mdiIu8Kk1WMC[24] W. Hasselbring, 《软件架构: 过去、现在与未来》, Springer International Publishing, 瑞士, 2018年, 第169–184页。doi:10.1007/978-3-319-73897-0_10。[25] C. Del Rosso, 《通过软件架构评估实现持续演化: 一项案例研究》, Journal of Software Maintenance and Evolution: Research and Practice 18 (5)（2006年）351–383。doi:https://doi.org/10.1002/smr.337。URL https://onlinelibrary.wiley.com/doi/abs/ 10 1002 337 。 /smr。[26] M. Daneva, R. Bolscher, 《我们对持续交付和DevOps中的软件架构风格了解多少? 》, 载于: M. van Sinderen, L. A. Maciaszek（编）, 《软件技术》, 2020年, 第26–39页。[27] H. Brunelière, R. Eramo, A. Gómez, V. Besnard, J. Bruel, M. Gogolla, A. Kästner, A. Rutle, 《复杂系统中面向设计—运行时交互的模型驱动工程: 科学挑战与路线图——MDE@Derun 2018研讨会报告》, 载于: M. Mazzara, I. Ober, G. Salaün（编）, 《软件技术: 应用与基础——STAF 2018联合研讨会, 法国图卢兹, 2018年6月25–29日, 修订版入选论文》, 计算机科学讲义第11176卷, Springer, 2018年, 第536–543页。doi:10.1007/978-3-030-04771-9\ _40。URL https://doi.org/10.1007/978-3-030-04771-9_40

[28] W.Hasselbring、A. van Hoorn, Kieker: 用于软件工程研究的监控框架, 《软件影响》5（2020年）100019。doi: 10.1016/j.simpa.2020.100019。URLhttps://doi.org/ 10 1016 2020 100019 。/ j.simpa.。D.Weyns, [29] 《自适应系统导论: 当代软件工程视角》, 约翰·威立父子公司, 2020年。[30] R. de Lemos、D. Garlan、C. Ghezzi、H. Giese、J. Andersson、M.Litoiu、B.R. Schmerl、D. Weyns、L. Baresi、N. Benco mo、Y. Brun、J.Cámara、R.Calinescu、M. B. Cohen、A. Gorla、V.Grassi、L. Grunske、P.Inverardi、J.Jézéquel、S.Malek、R.Mirandola、M. Mori、H. A. Müller、R. Rouvoy、C.M. F. Rubira、É.Rutten、M.Shaw、G. Tamburrelli、G.Tamura、N. M. Villegas、T. Vogel、F. Zambonelli, 自适应系统的软件工程: 提供保证方面的研究挑战, 载于: 《自适应系统的软件工程》, 《计算机科学讲义》第9640卷, Springer, 2013年, 第3–30页。[31] D. Weyn、I. Gerostathopoulos、N. Abbas、J. Andersson、S. Biffl、P. Brada、T. Bures、A. D.Salle、P. Lago、A.Musil、J. Musil、P.Pelliccione, 自适应在工业界应用的调查初步结果, 载于: 2022年软件工程与自适应和自管理系统国际研讨会（SEAMS）, 2022年, 第70–76页。doi: 10.1145/3524844.3528077。[32] J. Bosch, 持续软件工程: 简介, 载于: 持续软件工程, Springer, 2014年, 第3–13页。[33] B.Fitzgerald、K.-J.Stol, 持续软件工程: 路线图与议程, 《系统与软件杂志》123（2017年）176–189。[34] C.-P. Bezemer、S. Eismann、V.Ferme、J. Grohmann、R.Heinrich、P.Jamshidi、W. Shang、A. van Hoorn、M. Villavicencio、J. Walter、F. Willnecker, DevOps如何处理性能问题? , 载于: 2019年ACM/SPEC国际性能工程会议, ICPE'19, 计算机协会, 纽约, NY, 美国, 2019年, 第45–50页。doi:10.1145/3297663.3309672。URL https://doi.org/ 10 1145 3297663 3309672 。/。[35] C.Wohlin, 系统性文献研究中的滚雪球法指南及其在软件工程中的复现, 载于: 第18届软件工程评估与评价国际会议论文集, 2014年。doi:10.1145/2601248.2601268。URLhttps://doi.org/10.1145/2601248.2601268[36] 复现包《持续软件工程中的软件性能架构支持: 系统映射研究》（2021年）。 6298843 URLhttps://zenodo.org/record/[37] D. Bjørner, 《软件工程三联篇》。软件工程3——领域、需求与软件设计, 施普林格出版社, 2006年。URL https://link.springer.com/book/ 10 1007 3 540 33653 2 。/ - - - B.R. [38] Bryant、J. Gray、M. Mernik, 领域特定软件工程, 载于: FSE/SDP未来软件工程研究研讨会论文集, FoSER '10, 2010年, 第65–68页。doi:10.1145/1882362。1882376。[39] X.Huang、J.Lin、D.Demner-Fushman, 评估PICO作为临床问题知识表示的有效性, 载于: AMIA年度研讨会论文集。2006年, 2006年, 359 63pp。–。[40] B.Kitchenham、S. Charters, 软件工程系统性文献综述实施指南（2007年）。[41] K.E.Emam, 基准测试卡帕系数: 软件过程评估中的评定者间一致性, 《实证软件工程》4（2）（1999年）113–133。doi: 10.1023/A:1009820201126。URL https://doi.org/ 10 1023 1009820201126 。/A: C. Wohlin、P. Runeson、P. A. daMota Silveira Neto、E. Engström、I. do Carmo Machado、E. S.de Almeida, 软件工程映射研究的可靠性, 《系统与软件杂志》86（10）（2013年）2594–2610。doi:https://doi.org/10.1016/j.jss.2013.04.076。URL https://www.sciencedirect.com/science/article/pii/ S0164121213001234[43] K. Petersen、R. Feldt、S. Mujtaba、M. Mattsson, 软件工程中的系统映射研究, 载于: EASE, 2008年。[44] M. Armbrust、A. Fox、R.Griffith、A.D. Joseph、R. Katz、A.Konwinski、G. Lee、D.Patterson、A.Rabkin、I. Stoica、M.Zaharia, 云之上: 云计算的伯克利视角, 技术报告, 加州大学伯克利分校（2009年）。URL http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html

[42] [45] J . Slaby, S . Baker, J . Hill, D . Schmidt, 应用系统执行模型 -

用于评估企业分布式实时与嵌入式系统服务质量的工具，载于：第12届IEEE嵌入式与实时计算系统及应用国际会议（RTCSA'06），2006年，第350–362页。 doi:10.1109/RTCSA.2006.17。[46] Y.Dang、Q.Lin、P.Huang, AIOps：现实世界的挑战与研究创新，载于：2019年IEEE/ACM第41届软件工程国际会议：配套论文集（ICSE-Companion），2019年，第4–5页。 doi:10.1109/ICSE-Companion.2019.00023。 [47] D. Ameller、X. Franch、C. Gómez、S. Martínez-Fernández、J. Araújo、S.Biffl、J. Cabot、V. Cortellessa、D. M. Fernández、A. Moreira、H. Muc- cini、A.Vallecillo、M. Wimmer、V. Amaral、W.Böhm、H. Brunelière、L. Burgueño、M.Goulão、S. Teufl、L. Berardinelli，模型驱动开发中的非功能需求处理：一项调查，《IEEE软件工程汇刊》47（4）（2021年）818–835。 doi:10.1109/TSE.2019.2904476。[48] N. Bencomo、S.Götz、H.Song, Models@run.time：研究现状与研究挑战导览，《软件与系统建模》18（5）（2019年）3049–3082。 doi:10.1007/s10270-018-00712-x。[49] A. Bennaceur、R.France、G. Tamburrelli、T. Vogel、P. J. Mosterman、W. Cazzola、F. M.Costa、A.Pierantonio、M.Tichy、M.Akşit、P. Em-manuelson、H. Gang、N. Georgantas、D. Redlich，利用自适应软件中的运行时模型，载于：N.Bencomo、R. France、B. H. Cheng、U. Assmann（编），Models@run.time，计算机科学讲义第8378卷，Springer，2014年，第19–46页。 doi:10.1007/978-3-319-08915-7_2。URL https://hal.inria.fr/hal-01018754 [50] C. Wohlin、P.Runeson、M. Höst、M. C. Ohlsson、B.Regnell，《软件工程中的实验》，Springer，2012年。 [51] T. Dybå、T.Dingsøyr，敏捷软件开发的实证研究：一项系统综述，《信息与软件技术》50（9-10）（2008年）833–859。 doi:10.1016/j.infsof.2008.01.006。URL http://dx.doi.org/10.1016/j.infsof.2008.01.006

附录 A. The SelectedPapers

[软件性能1] G. Valetto and G. Kaiser, 使用过程技术来控制和协调软件适应。国际软件工程会议, 2003年。 https://doi.org/10.1109/ICSE.2003.1201206

[软件性能2] Garlan, D.，Cheng, SW.，Huang, AC.，Schmerl, B.，Steenkiste, P.，《基于架构的自适应与可复用基础设施》。计算机学报，2004年。 https://doi.org/10.1109/MC.2004.175

[软件性能3] Del Rosso, C., 通过软件架构评估实现持续演化：一项案例研究。《Journal of Software Maintenance and Evolution》，2006年。https://doi.org/10.1002/smr.337[软件性能4] T. Tsai、K. Vaidyanathan 和 K. Gross，低开销运行时内存泄漏检测与恢复。太平洋地区可靠计算国际研讨会，2006年。 https://doi.org/10.1109/PRDC.2006.42[软件性能5] Y. Liu、I. Gorton 和 V. K. Le，面向自适应 J2EE 应用的可配置事件关联架构。澳大利亚软件工程会议，2007年。 https://doi.org/10.1109/ASWEC.2007.5[软件性能6] Del Rosso, C., 软件产品族架构的软件性能调优：实时嵌入式系统领域的两项案例研究。《系统与软件杂志》，2008年。 https://doi.org/10.1016/j.jss.2007.07.006[软件性能7] I. Epifani、C. Ghezzi、R. Mirandola 和 G. Tamburrelli，通过运行时参数适应实现模型演化。国际软件工程会议，2009年。 https://doi.org/10.1109/ICSE.2009.5070513

[软件性能8] R. J. Pooley 和 A. A. L. Abdullatif，CPASA：软件架构的持续性能评估。计算机系统工程国际会议暨研讨会，2010年。https://doi.org/10.1109/ECBS.2010.16[软件性能9]Brosig F、Huber N、Kounev S.，分布式组件系统架构级性能模型的自动提取。国际自动化软件工程会议，2011年。 https://doi.org/10.1109/ASE.2011.6100052[软件性能10] Ehlers J、van Hoorn A、Waller J、Hasselbring W.，用于性能异常定位的自适应软件系统监控。自主计算国际会议，2011年。 https://doi.org/10.1145/1998582.1998628

[软件性能11] Ehlers, Jens andHasselbring, Wilhelm, 一种用于基于组件的软件系统的自适应监控框架. 欧洲软件架构会议, 2011年. https://doi.org/10.1007/978-3-642-23798-0_30

[SP12] R. Calinescu、L. Grunske、M. Kwiatkowska、Mirandola R、G. Tamburrelli，《基于服务的系统中的动态服务质量管理与优化》。IEEE 软件工程汇刊，2011年。https://doi.org/10.1109/TSE.2010.92

[软件性能13] Rodrigo N. Calheiros、Rajiv Ranjan、Rajkumar Buyya，《云计算环境中基于分析性能和服务质量的虚拟机配置》。国际并行处理会议，2011年。 https://doi.org/10.1109/ICPP.2011.17

[软件性能14] Von Massow R、Van Hoorn A、Hasselbring W，《基于组件的软件架构的性能仿真》，运行时可重配置自主系统。欧洲软件架构会议，2011年。 https://doi.org/10.1007/978-3-642-23798-0_5

[软件性能15] Woodside M、Franks G、Petriu DC，《自适应软件性能建模》。IEEE 年度计算机软件与应用程序会议研讨会，2012年。 https://doi.org/10.1109/COMPSACW.2012.85

[软件性能16] von Laszewski、Gregor、Lee、Hyungro、Diaz、Javier、Wang、Fugang、Tanaka、Koji、Karavinkoppa、Shubhada、Fox、Geoffrey C. 和 Furlani、Tom，《面向联邦云和裸机环境的基于计费与度量的云迁移和云播种框架设计》。云服务、联邦与第八届 Open Cirrus 峰会研讨会，2012年。 https://doi.org/10.1145/2378975.2378982

[软件性能17] Reiner Jung, Robert Heinrich, EricSchmieders, 使用 Kieker 和 Palladio 进行模型驱动的插桩以预测动态应用程序。Kieker/Palladio日论文集，2013年。 http://ceur-ws.org/Vol-1083/paper11.pdf[软件性能18] Wert, Alexander，通过系统化实验进行性能问题诊断。CompArch：基于组件的软件工程与软件架构，2013年。 https://doi.org/10.1145/2465498.2465499[软件性能19] ChiprianovV,FalknerK,Szabo C,PuddyG.，分布式实时嵌入式系统之系统的架构支持：模型驱动的性能预测。欧洲软件架构会议，2014年。https://doi.org/10.1007/978-3-319-09970-5_30

[自适应20] M.S. S.Junior 和N. S. Rosa 和F. A. A. Lins，长运行工作流的执行支持。国际计算机与信息技术会议，2014年。https://doi.org/10.1109/CIT.2014.94

[软件性能21] Birngruber, Erich 和 Forai,Petar和Zauner, Aaron，《全面回顾：整体度量指标在数据密集型计算环境中的系统性能和用户体验可见性》。高性能计算用户支持工具国际研讨会，2015年。https://doi.org/10.1145/2834996.2835001

[软件性能22] Incerto,E.、Tribastone, M.和Trubiani, C.，一种基于排队网络流体分析的运行时自适应主动方法。第1届质量感知DevOps 国际研讨会，QUDOS 2015，2015年。https://doi.org/10.1145/2804371.2804375

[软件性能23] Wert, Alexander 和 Schulz,Henning和 Heger, Christoph，AIM：可适应的检测与监控，用于自动化软件性能分析。软件测试自动化国际研讨会，2015年。https://doi.org/10.1109/AST.2015.15[软件性能24] Brebner, P.，从应用程序性能管理（APM）数据自动进行性能建模：一份经验报告。性能工程国际会议，2016年。https://doi.org/10.1145/2851553.2851560[软件性能25] Gerostathopoulos, I.和 Bures, T. 和 Schmid, S. 和Horky, V.和 Prehofer, C. 和 Tuma, P.，迈向软件密集型系统之系统中的系统化实时实验。软件密集型系统之系统国际研讨会，2016年。https://doi.org/10.1145/3175731.3176175[软件性能26] HeinrichR.，动态云应用中的架构运行时模型：用于性能和隐私分析。性能评估，2016年。https://doi.org/10.1145/2897356.2897359[软件性能27] Incerto, E.和Tribastone,M.和 Trubiani, C.，符号化性能自适应。软件工程自适应与自管理系统国际研讨会，2016年。https://doi.org/10.1145/2897053.2897060[软件性能28] Keck, P. 和 Hoorn, A.V. 和 Okanovic, D.和Pitakrat, T.和 Dullmann, T.F.，基于反模式的问题注入：用于评估性能和可靠性评估技术。软件可靠性工程国际会议研讨会，2016年。https://doi.org/10.1109/ISSREW.2016.36[软件性能29] Spinner S, WalterJ，Kounev S.，虚拟化环境中的在线性能模型提取参考架构。软件开发性能方法挑战研讨会，2016年。https://doi.org/10.1145/2859889.2859893[软件性能30] Willnecker, F. 和Krcmar, H.，分布式企业应用程序部署拓扑的优化。软件架构质量国际 ACM SIGSOFT会议，2016年。https://doi.org/10.1109/QoSA.2016.11[软件性能31] BrunnertA, Krcmar H.，使用企业应用程序资源剖析进行持续性能评估和容量规划。《系统与软件杂志》，2017年。https://doi.org/10.1016/j.jss.2015.08.030

[软件性能32] Incerto, E.和 Tribastone, M.和Trubiani, C.，通过高效模型预测控制实现软件性能自适应。自动化软件工程国际会议，2017年。https://doi.org/10.1109/ASE.2017.8115660

[软件性能33] Kunz，将监控数据转换为性能模型的通用平台。国际性能工程会议，2017年。https://doi.org/10.1145/3053600.3053635

[软件性能34] Li, Chenand Altamimi, Taghreedand Zargari, ManaHassanzade- hand Casale, Giulianoand Petriu, Dorina, Tulsa：一种用于将统一建模语言转换为分层排队网络，以分析数据密集型应用程序的工具。系统定量评估，2017年。https://doi.org/10.1007/978-3-319-66335-7_18

[软件性能35] Perez-Palacin D，Ridene Y，Merseguer J.，DevOps中的质量评估：税务欺诈检测系统的自动化分析。国际质量感知DevOps研讨会，2017年。https://doi.org/10.1145/3053600.3053632[软件性能36] Trubiani, C. 和 Mirandola, R.，服务质量模型的持续重新架构：用于降低不确定性的协同分析。欧洲软件架构会议，2017年。https://doi.org/10.1007/978-3-319-65831-5_3[软件性能37] Walter, J.、Stier, C.、Koziolek, H. 和 Kounev, S.，一种用于架构性能模型的可扩展提取框架。国际性能工程会议伴随会议，2017年。https://doi.org/10.1145/3053600.3053634[软件性能38] Y. Bao、M. Chen、Q. Zhu、T. Wei、F. Mallet 和 T. Zhou，使用统计模型检验对不确定性感知的混合AADL设计进行定量性能评估。《IEEE计算机辅助设计集成电路与系统汇刊》，2017年。https://doi.org/10.1109/TCAD.2017.2681076[软件性能39] Aleti, A.、Trubiani, C.、van Hoorn, A. 和 Jamshidi, P.，一种用于稳健软件性能估算中不确定性传播的高效方法。《系统与软件杂志》，2018年。https://doi.org/10.1016/j.jss.2018.01.010[软件性能40] Bernardi, Simona、Dominguez, Juan L.、Gomez, Abel、Joubert, Christophe、Merseguer, Jose、Perez-Palacin, Diego、Requeno, Jose I. 和 Romeu, Alberto，使用流程挖掘进行性能评估的系统化方法。《实证软件工程》，2018年。https://doi.org/10.1007/s10664-018-9606-9[软件性能41] Cholomskis, Aurimasand Pozdniakova, Olesiaand Mažeika, Dalius，自动扩缩模块的云软件性能指标收集与聚合。信息与软件技术国际会议，2018年。https://doi.org/10.1007/978-3-319-99972-2_10[软件性能42] D. Bardsley、L. Ryan 和 J. Howard，无服务器性能与优化策略。国际智能云会议，2018年。https://doi.org/10.1109/SmartCloud.2018.00012[软件性能43] Falkner, Katrina、Szabo, Claudia、Chiprianov, Vanea、Puddy, Gavin、Rieckmann, Marianne、Fraser, Dan 和 Aston, Cathlyn，系统之系统的模型驱动性能预测。《软件系统建模》，2018年。https://doi.org/10.1007/s10270-016-0547-8

[软件性能44] Pitakrat, T.、Okanović, D.、van Hoorn, A. 和 Grunske, L.，Hora：架构感知在线故障预测。《系统与软件杂志》2018年。 https://doi.org/10.1016/j.jss.2017.02.041

[软件性能45] Trubiani, C. and Bran, A. 和 Van Hoorn, A. 和 Avritzer, A. 和Knoche,H.，利用负载测试和性能剖析进行性能反模式检测。信息与软件技术，2018年。 https://doi.org/10.1016/j.infsof.2017.11.016

[软件性能46] Vögele, Christianand van Hoorn, Andréand Schulz, Eikeand Has- selbring, Wilhelmand Krcmar, Helmut, 《WESSBAS：用于负载测试和性能预测的概率工作负载规范提取——一种面向基于会话的应用系统的模型驱动方法》。《软件系统建模》，2018年。https://doi.org/10.1007/s10270-016-0566-5

[软件性能47] Willnecker, F. 和Krcmar,H.，面向分布式应用的部署拓扑多目标优化。ACM互联网技术汇刊，2018年。 https://doi.org/10.1145/3106158

[自适应48] Cortellessa, V.、DiPompeo, D.、Eramo,R.、Tucci, M.，基于模型的方法：用于基于微服务的系统的持续性能工程。《系统与软件杂志》，2022年。 https://doi.org/10.1016/j.jss.2021.111084

[软件性能49] Bezemer,C.-P.、Eismann, S.、Ferme, V.、Grohmann, J.、Heinrich, R.、Jamshidi, P.、Shang, W.、Van Hoorn, A.、Villavicencio, M.、Walter, J. 和 Willnecker, F.，《DevOps 中如何处理性能？工业实践调查》。国际性能工程会议，2019年。 https://doi.org/10.1145/3297663.3309672[软件性能50] Castellanos, C. and Varela, C.A. and Correal, D.，《大数据分析应用程序中的性能质量场景测量：一种 DevOps 与领域特定模型方法》。ACM 国际会议论文集系列，2019年。 https://doi.org/10.1145/3344948.3344986[软件性能51] Grohmann, J. and Eismann, S. and Elflein, S. and Kistowski, J.V. and Kounev, S. and Mazkatli, M.，《使用特征选择技术检测性能模型中的参数依赖关系》。年度国际建模、分析与仿真研讨会（计算机与电信系统），MASCOTS，2019年。 https://doi.org/10.1109/MASCOTS.2019.00042[软件性能52] Muller, H. and Bosse, S. and Turowski, K.，《机器学习在企业应用程序服务容量管理中的效用》。国际信号图像技术与互联网系统会议，2019年。https://doi.org/10.1109/SITIS.2019.00053[软件性能53] Perez-Palacin, D. and Merseguer, J. and Requeno, J.I. and Guerriero, M. and Di Nitto, E. and Tamburri, D.A.，《面向数据密集型应用程序设计、质量评估与部署的 UML 配置文件》。《软件与系统建模》，2019年。 https://doi.org/10.1007/s10270-019-00730-3[软件性能54] R. Chatley and T. Field and D. Wei，《虚拟时间中的持续性能测试》。国际软件架构会议研讨会，2019年。 https://doi.org/10.1109/ICSA-C.2019.00027

[软件性能55] Spinnner, S. and Grohmann, J. and Eismann, S. and Kounev, S.，《面向自感知计算基础设施的在线模型学习》。《系统与软件杂志》，2019年。 https://doi.org/10.1016/j.jss.2018.09.089[软件性能56] Trubiani, C. and Jamshidi, P. and Cito, J. and Shang, W. and Jiang, Z. M. and Borg, M.，《性能问题？DevOps，请留意不确定性》。IEEE 软件，2019年。https://doi.org/10.1109/MS.2018.2875989[软件性能57] Willnecker, F. and Krcmar, H.，《基于模型的自动内存管理与垃圾回收行为预测》。Simulation Modelling Practice and Theory，2019年。 https://doi.org/10.1016/j.simpat.2018.09.014[软件性能58] Yasaweerasinghelage, R. and Staples, M. and Paik, H.-Y. and Weber, I.，《优化架构的性能、成本与安全性》。欧洲软件架构会议，2019年。https://doi.org/10.1007/978-3-030-29983-5_11[软件性能59] De Sanctis, M. and Bucchiarone, A. and Trubiani, C.，《QoS 感知型自适应应用程序的 DevOps 视角》。软件工程持续开发与软件生产和部署新范式国际研讨会，2020年。 https://doi.org/10.1007/978-3-030-39306-9_7[软件性能60] Giaimo, Federico and Berger, Christian，《以重型商用车辆日常运行场景为例的汽车软件持续实验》。欧洲软件架构会议，2020年。https://doi.org/10.1007/978-3-030-58923-3_5[软件性能61] Heinrich, R.，《用于整合运行时观测与基于组件的模型的架构运行时模型》。《系统与软件杂志》，2020年。 https://doi.org/10.1016/j.jss.2020.110722[软件性能62] Mazkatli, M. and Monschein, D. and Grohmann, J. and Koziolek, A.，《具有参数依赖关系的架构性能模型增量校准》。国际软件架构会议，2020年。 https://doi.org/10.1109/ICSA47634.2020.00011[软件性能63] Voneva, Sonya and Mazkatli, Manar and Grohmann, Johannes and Koziolek, Anne，《优化参数依赖关系以增量提取性能模型》。欧洲软件架构会议，2020年。 https://doi.org/10.1007/978-3-030-59155-7_17[软件性能64] Camilli, M., Janes, A., Russo, B.，《面向微服务系统性能模型的自动化测试式学习与验证》。《系统与软件杂志》，2022年。 https://doi.org/10. 1016/j.jss.2022.111225[软件性能65] Kugele, S., Obergfell, P., Sax, E.，《面向服务的汽车软件架构的基于模型的资源分析与综合》。《软件与系统建模》，2021年。 https://doi.org/10.1007/s10270-021-00896-9[软件性能66] Grohmann, J., Eismann, S., Bauer, A., Spinner, S., Blum, J., Herbst, N., Kounev, S.，《SARDE：持续、自适应资源需求估算框架》。ACM 自主与自适应系统汇刊，2021年。 https://doi.org/10.1145/3463369

附录 B。 数据提取

表 B.4 概述了数据提取得出的结果。具体而言，它列出了入选的

论文（软件性能）及其针对所选关键词的评估（请注意，黑色方块表示已充分研究的关键词，白色方块表示已部分研究的关键词）。在接下来的章节中，我们将报告论文分类的结果，以回答本研究的研究问题。针对每个研究问题，我们将讨论所得结果，并阐述其对从事软件架构、性能工程和持续软件工程工作的研究人员与从业者的启示。

附录 C. 关键词定义

本附录定义了我们在数据提取过程中使用的关键词。

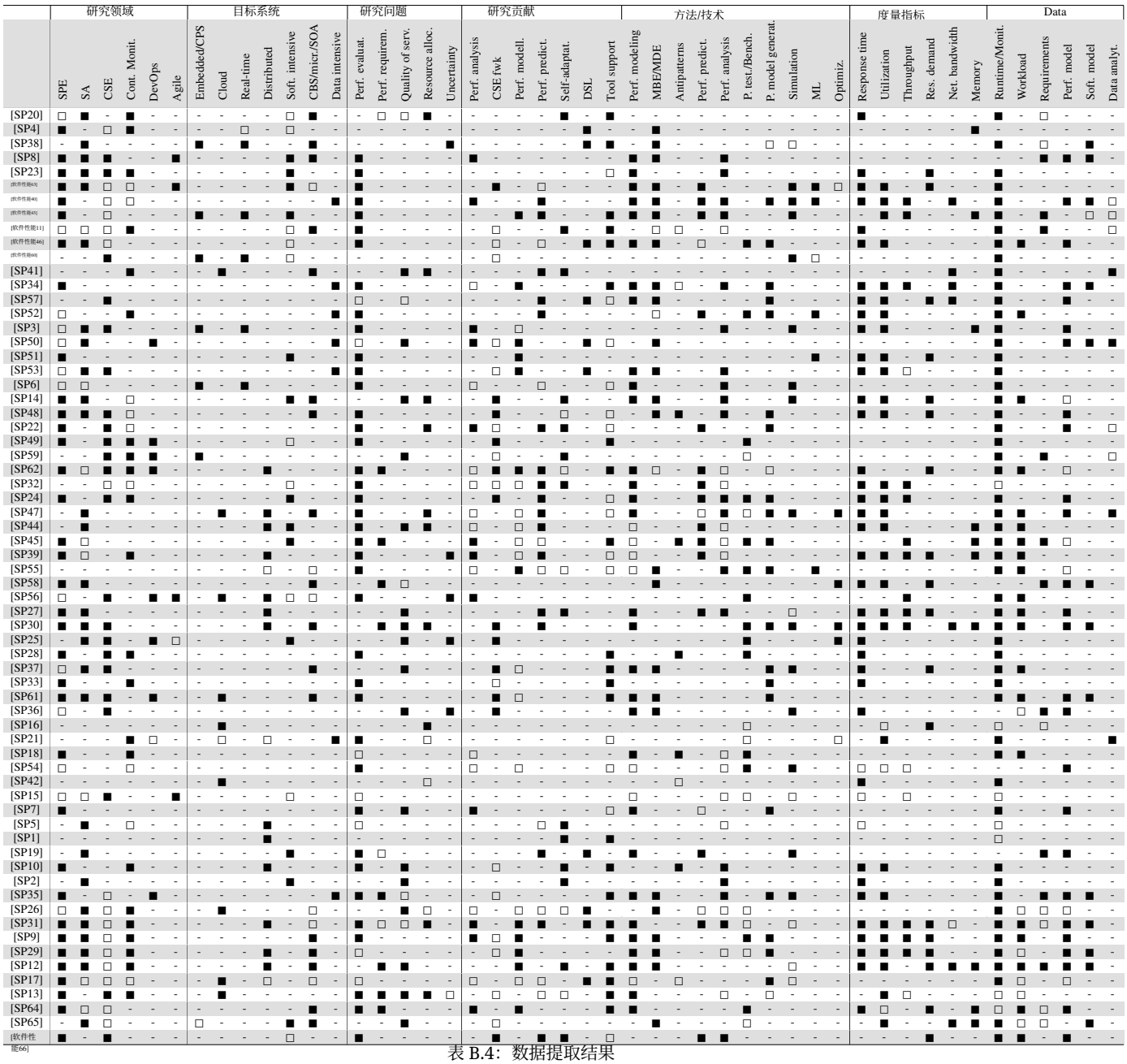

研究领域
目标系统
研究问题
研究贡献
方法/技术
度量指标
Data
SPE
SA
CSE
Cont. Monit.
DevOps
Agile
Embedded/CPS
Cloud
Real-time
Distributed
Soft. intensive
CBS/micr./SOA
Data intensive
Perf. evaluat.
Perf. requirem.
Quality of serv.
Resource alloc.
Uncertainty
Perf. analysis
CSE fwk
Perf. modell.
Perf. predict.
Self-adaptat.
DSL
Tool support
Perf. modeling
MBE/MDE
Antipatterns
Perf. predict.
Perf. analysis
P. test./Bench.
P. model generat.
Simulation
ML
Optimiz.
Response time
Utilization
Throughput
Res. demand
Net. bandwidth
Memory
Runtime/Monit.
Workload
Requirements
Perf. model
Soft. model
Data analyt.
[SP20]
[SP4]
[SP38]
[SP8]
[SP23]
[软件性能63]
[软件性能40]
[软件性能43]
[软件性能11]
[软件性能66]
[软件性能68]
[SP41]
[SP34]
[SP57]
[SP52]
[SP3]
[SP50]
[SP51]
[SP53]
[SP6]
[SP14]
[SP48]
[SP22]
[SP49]
[SP59]
[SP62]
[SP32]
[SP24]
[SP47]
[SP44]
[SP45]
[SP39]
[SP55]
[SP58]
[SP56]
[SP27]
[SP30]
[SP25]
[SP28]
[SP37]
[SP33]
[SP61]
[SP36]
[SP16]
[SP21]
[SP18]
[SP54]
[SP42]
[SP15]
[SP7]
[SP5]
[SP1]
[SP19]
[SP10]
[SP2]
[SP35]
[SP26]
[SP31]
[SP9]
[SP29]
[SP12]
[SP17]
[SP13]
[SP64]
[SP65]
[软件性能66]

表 B.4：数据提取结果

Keywords	Description
Software Performance Engineering	软件工程活动以及贯穿整个软件的相关分析 开发周期，并致力于满足性能需求。
软件架构	与整体系统结构的设计或分析相关的活动，以及 相关的开发决策。
持续软件 工程	与快速开发、发布软件并从中学习相关的持续活动 并行周期（例如，持续集成、持续交付）。
DevOps	促进开发与运维之间集成的实践。
Continuous monitoring	对系统运行时行为的持续观察；密切相关 对于CSE 和 DevOps，在许多研究中，它被视为一种独立的实践。
敏捷软件 开发	迭代式软件开发，其中需求和解决方案通过 自组织跨职能团队之间的协作而演进。

表 C.5：研究领域 - 关键词

关键词	Description
Embedded / Cyber-physical	嵌入式系统在更大型的机械或 电子系统。而在信息物理系统中，物理组件彼此深度交织。 借助软件组件彼此交互，其交互方式会随上下文而变化。
云	按需提供计算机系统资源，尤其是数据存储的可用性 和计算能力，且无需用户直接主动管理。
实时	受实时约束的硬件和软件系统，例如从 事件到系统响应。
Distributed	其组件位于不同联网计算机上的系统， 它们相互沟通并协调其行动。
Data intensive	利用基于高级的机器无关方法的系统 数据运维以及在分布式计算集群中的运行时控制。
Software intensive	在这些系统中，软件在设计、构建和 实现。此外，复杂系统行为本质上难以 由于其组成部分（以及环境）之间的依赖关系和关系，该模型。
Component-based sw. / Microservices / SOA	面向服务的架构是一种用于软件开发的企业架构方法，基于 可复用软件组件（如基于组件的软件）或服务。作为其 演进中，微服务架构构建由可独立部署的服务组成的套件。

表 C.6：目标系统 - 关键词

Keywords	Description
性能评估/ 评估	涉及在系统开发 生命周期中应用相关技术，以确保满足性能方面的非功能需求（例如 吞吐量、延迟或内存使用）。
性能需求	处理需求工程的方法，即定义和/或确保 软件系统在某些特定条件下完成给定任务的表现 （例如，速度、效率、资源消耗、吞吐量、响应时间）。
Quality of service	处理整体质量规范或测量的方法 服务（例如，性能、可用性、可扩展性和可服务性）的。。
资源分配 / 部署	确保资源高效使用的资源管理方法 基于不同工作负载
不确定性	处理不确定性的方法，包括与参数值相关的不确定性（例如， 工作负载、运行概况和资源需求）因为不可避免地影响整体 系统性能。

表 C.7：目标问题 - 关键词

Keywords	Description
Performance analysis approach	旨在提出性能分析方法和技术的途径 以解决软件开发生命周期中的性能问题。
领域特定语言	提出针对特定应用程序领域的专用（建模）语言的方法 领域
持续工程 框架	提出支持一种或多种持续软件的框架的论文 engineering dimensions
Performance modelling approach Tool support	提出用于性能分析的性能建模的方法，以及 国产成人 prediction of the system 提供支持所提出方法的已实现工具的论文
性能预测 方法	提出性能预测方法以估计一个或多个 性能因素（例如执行时间或缓存未命中）。
自适应	在设计时和运行时为软件系统赋能的方法，以 adapt to dynamic conditions

表 C.8：贡献 - 关键词

Keywords	Description
Performance modeling	用于对系统行为和工作负载进行建模以预测的。 various performance metrics.
Model based engineering / Model driven engineering	基于模型和模型驱动的方法论和技术，不限制 将模型应用于性能领域，但在若干情况下，模型也 在软件工程领域中被视为（一等公民）。
性能反模式 / 根本 原因 / 瓶颈检测	通过匹配规范来识别性能问题的特定技术 反模式，或通过识别根本原因，或通过检测性能瓶颈。
性能预测 techniques	通常依赖于预测应用程序和系统性能的方法，通常依赖于 关于机器学习和分析建模。
Performance analysis tech.	用于自动化性能分析任务的方法和技术。
Parametric dependency	利用组件的输入参数与其之间关系的方法 从性能属性到性能预测。
Performance / Load / Benchmark testing	包括多种性能测试类型的方法，例如负载测试和 benchmark testing.
性能模型 生成 / 提取	用于自动生成性能模型和提取的技术，例如基于受监控/运行时数据的 基于例如监控/运行时数据的性能模型。
Simulation	开发仿真模型或利用仿真来 评估系统的性能。
机器学习	将机器学习技术用于性能工程的方法 活动，例如预测或测试。
（多目标） 优化	使用（多目标）优化方法进行性能提升的方法 improvements.

表 C.9：方法论 - 关键词

Keywords	Description
Response time	从用户发送请求的那一刻起，到.....的时间 应用程序表明请求已完成。
Utilization	组件被占用的时间百分比，相较于 该组件可供使用的总时间。
吞吐量	响应客户端请求而传输到服务器的数据量，在给定 时间段内。
资源需求	满足各类工作需求所需的资源数量。
Network bandwidth	每单位时间内可通过.....传输的最大数据量 communication channel.
Memory / Memory Leaks	内存使用表示根据可用资源百分比得出的平均利用率 特定时刻正在使用的内存。内存泄漏是指软件 无法高效管理可用资源。

表 C.10：度量指标 - 关键词

Keywords	Description
运行时 / 受监控	系统执行期间从传感器或监控中获取的数据。
工作负载	为测试或分析而在已识别的应用场景中分配负载。
需求	系统必须满足的条件或具备的能力。
Performance model	表示系统行为并用于分析/预测的模型 performance metrics.
Software model	表示系统行为特征的抽象。
Data analytics	用于数据分析过程和分析，以发现趋势并得出 关于其中所含信息的结论。

表 C.11：数据 - 关键词