arXiv:2602.05792v1 [cs.NI] 5 Feb 2026

云虚拟化实验数据分析

预印本

Pedro R. X. do Carmo
信息学中心（CIn）
网络与电信研究组（GPRT）
伯南布哥联邦大学（UFPE）
累西腓，巴西
pedro.carmo@gprt.ufpe.br

Eduardo Freitas
信息学中心（CIn）
网络与电信研究组（GPRT）
伯南布哥联邦大学（UFPE）
累西腓，巴西
eduardo.freitas@gprt.ufpe.br

Assis T. de Oliveira Filho
信息学中心（CIn）
网络与电信研究组（GPRT）
伯南布哥联邦大学（UFPE）
累西腓，巴西
assis.tiago@gprt.ufpe.br

Judith Kelner
信息学中心（CIn）
网络与电信研究组（GPRT）
伯南布哥联邦大学（UFPE）
累西腓，巴西
jk@gprt.ufpe.br

Djamel Sadok信息学中心（CIn）网络与电信研究小组（GPRT）伯南布哥联邦大学（UFPE）累西腓，巴西 j
amel@cin.ufpe.br

2026年2月6日

摘要

云计算范式对数据中心和电信基础设施的设计有着重要影响。它广泛利用虚拟化技术，将硬件和软件资源划分为更小的软件单元，从而提高操作灵活性。鉴于虚拟化带来的诸多优势，出现了多种处理和通信开销各异的虚拟化形式，包括全虚拟化和操作系统虚拟化。因此，由于中央处理器、输入/输出和网络资源上存在额外的虚拟化开销，预测数据平面的数据包吞吐量变得更具挑战性。本研究提供了一个主动网络测量数据集，这些数据是在改变各种网络参数时收集的，参数包括 CPU 亲和性、回显数据包注入频率、虚拟网络驱动程序类型、中央处理器、输入/输出或网络负载的使用情况，以及并发虚拟机的数量。研究中使用的虚拟化技术包括 KVM、LXC 和 Docker。本文考察了这些技术对一项关键网络指标——端到端延迟——的影响。此外，本文还构建数据模型，用于评估云计算环境对数据包往返时间的影响。为探索数据可视化，研究对数据集进行了预处理、相关性分析、降维和聚类。此外，本文还对数据集进行了简要分析，展示了该数据集在开发用于管理员决策的基于机器学习的系统方面的用途。

1 引言

近年来，云计算的广泛应用推动了面向服务的架构（SOA）的采用趋势。它支持按需弹性分配和共享硬件与软件资源。云计算提供可重复使用的分布式计算基础设施，从而降低投资和基础设施成本 [1]。总而言之，云计算能够提供对可配置计算资源的访问，这些资源可以快速、轻松地提供，更重要的是，可以按需提供。

虚拟化是驱动云计算数据中心、网络和电信系统基础设施的引擎。防火墙、路由器和交换机等网络设备，都有使用虚拟网络功能（VNF）实现的软件对应形式。这些功能基于软件构建，旨在运行于商用现成硬件之上。例如，与早期移动架构不同，5G采用由虚拟网络功能（VNF）组成的基于服务的架构（SBA）。换言之，虚拟化已成为新型处理与通信架构中无处不在的技术。尽管虚拟化带来了诸多益处，但它也增加了处理开销，影响网络性能。额外的虚拟化层（包括虚拟机本身、虚拟网络驱动程序、与主机操作系统的交互方式以及磁盘访问）都会影响数据包处理和传输所需的时间。因此，网络行为有时会出现不稳定的变化，而这些变化可能源于虚拟化的影响。本文将探讨造成此类性能不佳的因素，并分析它们对端到端数据包时延的影响。

众所周知，云系统承载着广泛的服务，例如数据库、Web 服务和网络功能。云应用的行为各不相同。有些可能依赖密集的 CPU 资源，而另一些则主要消耗磁盘 I/O 或网络资源。由于虚拟化在所承载的服务之间共享此类资源，而数据包处理需要中央处理器、磁盘 I/O 和网络资源，因此它们的网络性能会受到相应影响。此外，在创建虚拟化应用时可以使用多种虚拟化技术，每种技术采用不同的方法，因而最终对网络性能产生不同的影响，这一点将在我们的实验中加以展示。因此，云管理员在决定使用哪种类型的虚拟化技术时必须考虑这些因素。未经规划的配置设置可能会使数据平面上的数据包处理遭受严重的性能损失。

在本研究中，我们描述了一个从实验中提取的数据集；这些实验通过在网络上进行主动测量，并改变若干相关参数，以模拟真实的云计算场景及其应用。我们搭建了一个专用测试平台，用以评估给定虚拟化技术对网络性能的影响。随后，从运行三种不同虚拟化技术的测试平台基础设施中收集了与网络性能相关的数据：基于内核的虚拟机（KVM）、Linux 容器（LXC）和 Docker 容器。我们的目标是评估它们的数据包处理开销，并衡量其对端到端服务时延的影响。首先，我们描述所采用的测试平台。接下来，我们展示所收集的数据集，并使用统计方法和数据科学方法对其进行分析；最后，利用这些数据构建机器学习模型，以协助管理员做出决策，包括根据场景的实际特征与需求选择合适的虚拟化技术。

本文其余部分安排如下：第2节介绍与本文相关的研究。第3节介绍所采用的测试平台以及数据的收集方式。第4节描述我们的场景，详细介绍数据集，并说明其处理过程。第5节介绍数据分析，并展示数据用例的示例；具体而言，还将介绍基于这些数据构建的回归模型。最后，第6节总结本文并讨论未来工作。

2 相关研究

本文首次提供了一个数据集，其中包含来自采用不同虚拟化技术构建的测试平台的实验数据。该数据集涵盖多种虚拟化场景，结合了若干网络参数，并测量了端到端延迟和 CPU 使用率等性能指标。本节回顾了一些同样开展实验的研究，以及与我们研究方法类似的工作。

文献 [2] 对比了传统虚拟机实现与容器的性能；具体而言，作者考察了 KVM 和 Docker 技术的使用情况。结果表明，在几乎所有已进行的测试中，Docker 虚拟化的表现相当或更优。然而，人们认为其采用的对比场景有限，只考察了数量较少的配置及其组合。这些作者也没有提供所使用的数据，只提供了脚本形式的代码，供有意开展相关实验的读者使用。

[3] 中介绍的工作在NFV基础设施背景下采用了数据科学分析方法。该论文描述了一个基于Kubernetes的测试平台，并收集基础设施即服务（IaaS）异常数据，以构建面向NFV应用的异常数据库。作者从三类异常中收集数据并对其进行标注。他们还

应用机器学习算法来发现离群点。他们构建了该数据集，并向研究界公开提供。我们的研究采用了与他们类似的方法。但与之不同的是，我们关注虚拟化类型及其在特定场景下的表现，并通过往返时间（RTT）和 CPU 使用率指标分析性能。

最后，在先前的一项研究中 [4], ，我们使用 KVM、LXC 和 Docker，并基于本文所介绍的数据集进行了实验评估。该研究通过组合多个参数来比较不同的虚拟化类型，例如 CPU 亲和性、测量回显数据包的注入频率、虚拟网络驱动程序的类型、中央处理器、输入/输出或网络过载，以及后台虚拟机的数量。通过分析获得的数据，我们发现虚拟化会影响 RTT 的准确性。在具有密集输入/输出特征的数据中心中，使用 VirtIO 网络驱动程序的 KVM 虚拟机表现良好，RTT 开销接近于零。在网络负载较高的数据中心，Docker 技术优于其他技术，与其他虚拟化类型相比，表现出的波动较小。该研究指出，CPU 亲和性并不会显著提升网络性能。此外，将网络负载施加到接收服务器上会导致 RTT 方面的结果更差。KVM 与 VirtIO 驱动程序结合时表现最佳，端到端延迟方面也是如此。最后，在几乎所有采用的场景中，Docker 虚拟化都优于 LXC。在所引用的文章中，通过实证分析和生成图表来分析原始数据，而统计方法则着重于场景分析。尽管已经得出了许多结果，本文仍在此先前实验分析的基础上，使用相同的数据集开展更深入的评估。我们的主要目标是运用统计分析和机器学习等数据科学概念得出结论，同时公开数据集并讨论相关数据，从中提炼见解和发现，供其他研究人员在各自的使用场景中参考。尽管如此，我们也强调，两项研究使用的是相同的测试台和数据。

3 测试平台与数据收集

本节介绍用于开展计划中实验的测试平台。该平台模拟云计算环境，由通过 P2P 拓扑连接的两台服务器主机构成。我们将虚拟机分为两类：主虚拟机和后台虚拟机。两台服务器各自承载：a) 用于测量网络性能的主虚拟机（VM）；以及 b) 用于产生资源开销的后台虚拟机。后者模拟云计算环境，其中多个虚拟机同时运行、争夺资源并进行高强度的资源利用。

实验首先启动“主虚拟机”，随后启动“后台虚拟机”。接着采集评估指标，包括以往返时间（RTT）表示的端到端延迟，以及 CPU 使用率。RTT 通过在“主虚拟机”上运行 ping 工具进行测量：“发送端主机”发送 ping 数据包，“响应主机”接收并响应这些数据包。ping 数据包返回发送端虚拟机时，即计算 RTT。为确保统计显著性，每项实验共获取 1000 个 RTT 样本。CPU 使用率通过在两台服务器上运行*mpstat* 工具进行测量。图 1 展示了测试平台的示意图。
实验分为两部分。初始实验使用 KVM 虚拟机监控程序，第二组实验则使用基于容器的虚拟机，即 LXC 和 Docker。此外，我们对各种系统配置和场景因素进行组合和调整，以构建全面且具有代表性的数据集。后台虚拟机的数量在 0 至 128 之间变化，其中 0 表示不存在资源竞争的基准场景。这些工作负载虚拟机竞争磁盘 I/O、中央处理器或网络资源；负载位置可以是“发送方服务器”或“响应方服务器”。用于主动测量延迟的回显数据包发送速率设为每秒 1、1,000 或 1,000,000 个数据包（即 1、1 kHz 和 1 MHz）。有关实验中使用的参数及其各自取值范围的更多详情，请参见表2。

实验结束时，生成了若干文本（.txt）文件，其中包含所有已发送 ping 数据包的往返时间信息，以及每项实验的 CPU 使用率信息。这些数据为网络管理员分析数据、提炼有意义的见解，并就其虚拟化环境的最佳配置及最适合其运营的配置做出明智决策奠定了基础。下一节将介绍实际的数据处理和数据集构建过程。

4 场景与数据集

本研究使用开源工具 tcpdump 和 mpstat，从 Linux 操作系统收集数据。收集到的原始数据以扩展名为“.txt”的文本文件形式保存。我们使用 Python 脚本从这些文件中提取数据并创建 CSV 表格，以供预处理、创建预测模型和分组使用。该数据集可公开获取，访问地址为 [5]。表1展示了该数据集的部分样本行。

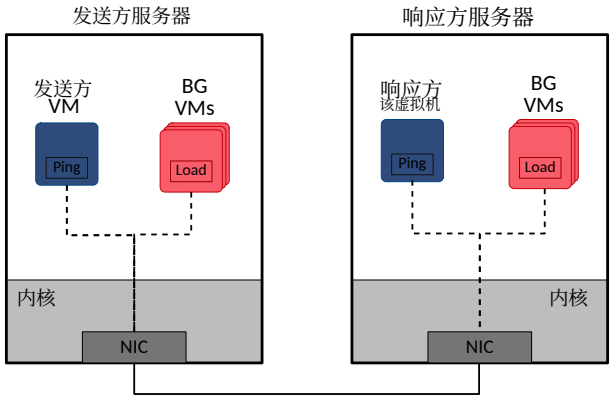


图 1：测试平台。改编自 [4]。

行_id	虚拟机编号_	CPU 亲和性	负载位置	负载类型	频率	驱动程序	虚拟化类型	中央处理器_HS_使用率	中央处理器_HR_使用率	往返时间_Ping
0	4	on	HS	NET	1000000.0	RTL8139	kvm	2.30	2.33	1.300
0	4	on	HS	NET	1000000.0	RTL8139	kvm	2.30	2.33	0.862
...	...	...	...	...	...	...	...	...	...	...
1609463	128	off	HR	IO	1000.0	0	Docker	9.35	99.50	0.575

表1：数据集

4.1 数据集

在测试平台上开展的实验生成了一个包含约 1,609,463 条（即超过一百万条）记录的数据集。该数据集包含七列，用于表示特定网络场景的特征。其中一列测量 RTT 性能指标，以变量“RTT_ping”表示。另有两列分别显示实验期间发送方和响应主机的 CPU 使用率（百分比）。数据集还包含以整数、浮点数、布尔值或字符串编码的参数。表2列出了每个参数的类型、描述和可能取值。在将这些数据用作机器学习算法的输入之前，必须进行预处理。图2展示了所采用的数据处理流程图。

参数	描述	数据类型	取值范围
虚拟机编号_	后台虚拟机数量	整数	0 1 2 4 8 16 32,64,128
CPU 亲和性	标识中央处理器是否已被分配用于处理数据包	布尔值	开/关
负载位置	每个后台虚拟机所执行的工作负载的影响也会根据其是否正在运行而有所不同于发送方服务器 (HS) 还是响应方服务器 (HR)。	字符串	“HS”或“HR"
负载类型	每个后台虚拟机执行的工作负载针对不同的硬件资源。这些包括中央处理器 ，网络或磁盘 I/O 负载	字符串	“中央处理器”、“输入输出”或“网络”
频率	以 pps（每秒数据包数）为单位的数据包发送频率	浮点数	1.0、1000.0、1000000.0
驱动程序	与 KVM 配合使用的虚拟网络驱动程序	字符串	“RTL8139" ，“E1000" ，“VirtIO", “VMXNET3" 和“0"（如果虚拟化类型不是 KVM")
虚拟化类型	所使用的虚拟化类型	字符串	“LXC”，“KVM”，“DOCKER”
中央处理器_HS_使用率	发送端主机上的 CPU 使用率（百分比）	浮点数	0.0 至 100.0
中央处理器_HR_使用率	响应主机上的 CPU 使用率（百分比）	浮点数	0.0 至 100.0
往返时间_Ping	往返时间	浮点数	任意非负浮点数

表2：数据集参数

4.2 预处理

预处理步骤对于确保分析结果的准确性和质量至关重要。错误的数据会导致结果失真，因为它会使模式检测变得困难，并导致模型表现不佳 [6,7]。此步骤包括转换和编码变量、提取特征，以及删除无用或不完整的数据。

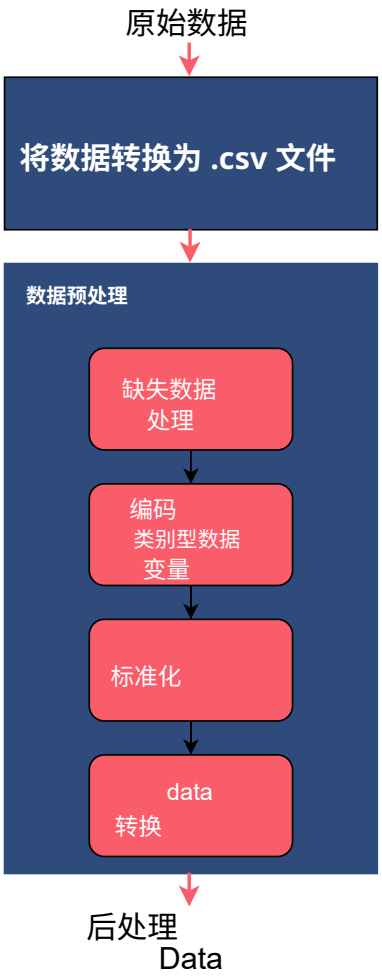


图2：数据处理步骤

首先，需要处理缺失数据。只有当“VirtType”列的值为“KVM”时，“驱动程序”参数列才包含非零值。为了解决这个问题，我们将“驱动程序”列与“VirtType”列合并，以创建一个新特征，可能的取值为“KVM_virtio”、“KVM_rtl8139”、“KVM_e1000”、“KVM_vmxnet3”、“LXC”和“DOCKER”。

接下来，使用独热编码 [8], 对数据进行编码。这种技术将类别数据转换为二进制表示。该方法会为类别变量的每个可能取值创建一列，并在相应列中填入 1。数据集中的类别列 LoadLocation、LoadType、Driver 和 VirtType 均使用独热编码进行编码。此步骤生成的新列包括 CPU、IO、NET、HS、HR、DOCKER、LXC、KVM_E1000、KVM_VIRTIO、KVM_RTL8139、KVM_VMXNET3，取值均为 1 或 0，其中 1 表示参数存在，0 表示参数不存在。

此外，使用 z分数标准化方法对非类别列进行归一化处理，将数据转换为均值为 0、标准差为 1 的分布。采用此方法是为了避免对某些算法的性能产生不利影响，并防止合并不同尺度的数值时出现问题，例如取值范围为 1 到 10^6 的“频率”列，以及取值范围为 0 到 128 的虚拟机_数量 列。数值根据下方所示的公式 1 计算：

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

其中， x 是原始值， μ 是数据的均值， σ 是标准差。

...	...	**CPU**	**IO**	**NET**
...	...	1	0	0
...	...	...	...	...
...	...	0	1	0
...	...	0	0	1

表 3：独热编码示例

完成数据转换步骤后，为便于分析和实验，会调整变量类型。所有独热编码列都设为整数值。在 CPU_Affinity 列中，值“on”转换为 1，值“off”转换为 0，二者均为整数。RTT_ping、CPU_HR_usage、CPU_HS_usage 和频率列仍编码为浮点数，而 VM_number 列仍采用整数类型。

5 数据分析与用例

在本节中，我们分析该数据集并讨论其潜在应用。

5.1 统计分析

本研究中进行的数据分析基于前一预处理阶段的输出。初步分析采用斯皮尔曼等级相关，即皮尔逊相关的非参数版本，如 [9] 所述。该相关性衡量两个排序变量之间关联的强度和方向，取值范围为 -1 到 1。负值表示负相关，正值表示正相关，0 表示不存在相关性。定义场景的场景变量与用于评估场景性能的性能变量之间观察到的相关性如图 3 所示。

讨论 初始测试结果得出了若干结论。如图 3 所示，后台运行的虚拟机数量会显著影响发送方和响应主机的 CPU 使用率。另一方面，其他变量对 CPU 消耗的影响很小。就 CPU 使用率而言，使用 Docker 会降低响应主机的 CPU 使用率。此外，研究发现，Stress NET（增加用于生成网络背景流量的虚拟机数量？）与往返时间（RTT）的增加密切相关。使用 VirtIO 驱动程序可显著提升 KVM 虚拟化的性能，而 rtl8139 驱动程序则会降低其效率。这是因为 VirtIO 驱动程序可让虚拟机（VM）直接（通过半虚拟化方式）访问设备和外围设备，因此速度快于较慢的仿真驱动程序。此外，Docker 虚拟化技术在 CPU 使用率和延迟指标方面均表现最佳。有趣的是，我们的分析表明，随着使用 ping 回显数据包进行测量的频率提高，RTT 会降低。这是因为 CPU 唤醒时间缩短，而且系统倾向于采用基于中断的传递方式；对于频率较高的数据包，这种方式效率更高 [4]。此外，后台虚拟机的数量不会影响 RTT 指标，而使用 CPU 亲和性也无法改善这两项指标。

5.2 聚类与 PCA

本节介绍一种用于进一步分析观测测量值的策略。主成分分析（PCA）是一种广受欢迎的降维技术，在相关文献中得到了广泛应用 [10]。它使我们能够将高维数据集投影到低维空间，从而降低数据的维度，使其更易于可视化。在本研究中，我们对数据集应用 PCA，将其降至二维，如图 4 所示。数据点根据我们测试的三种不同虚拟化技术进行标记：KVM、LXC 和 Docker。尽管这三种技术之间没有明显的区分，但显然，KVM 和 LXC 倾向于占据与 Docker 所占据空间不同的区域。

作为数据分析的延续，我们接下来应用一种聚类技术，以可视化前文所述主成分分析（PCA）的主要成分。聚类是一种机器学习技术，属于无监督学习方法。其目标是根据相似性将数据划分为若干组。在本研究中，我们使用 KMeans算法 [11]将数据划分为三组，分别对应三个虚拟化组：Docker、LXC 和 KVM。

KMeans算法根据数据到 K 质心的距离，将数据划分为 K 个簇，质心代表每个簇的中心。质心的初始位置是随机确定的，然后将每个数据点分配给

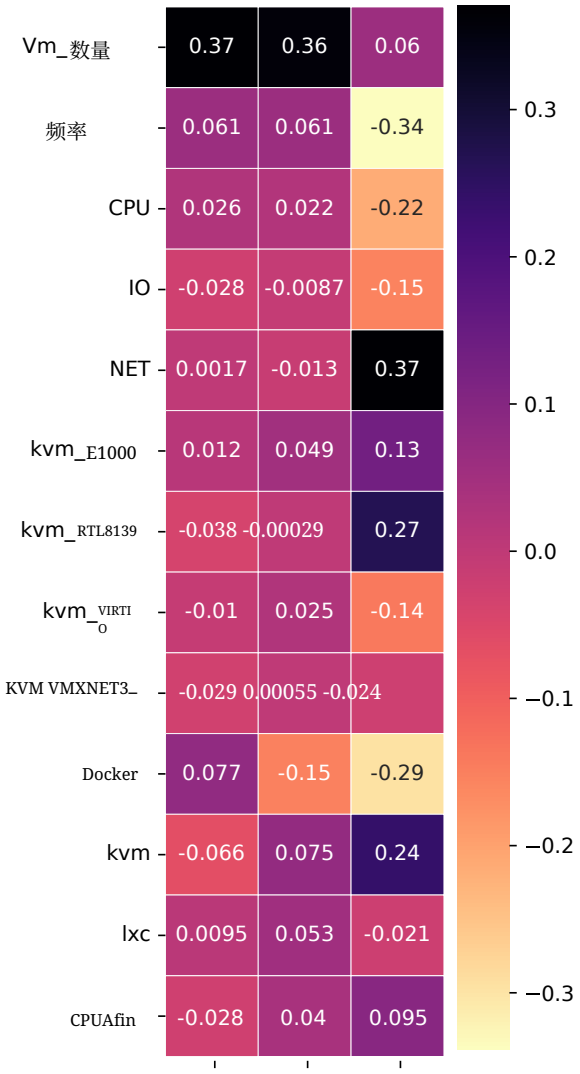


图 3：相关性

与最近的 第 t 个质心。最后，通过计算数据的均值重新确定质心的位置 a
每个簇中的点。 在我们的案例中，我们设置 $K = 3$。

聚类算法的结果如图5所示。图中的大点代表数据的质心。显然，可以识别出 Docker、LXC 和 KVM 这三组。这使我们得出结论：结合使用 PCA 和 K均值可以有效地可视化我们的高维数据。

5.3 基于机器学习的回归方法的应用

对于本文所述的场景，机器学习技术可以为网络管理员提供有价值的见解，帮助他们做出架构决策，例如选择最合适的虚拟化技术，或估算特定场景下的预期平均延迟。

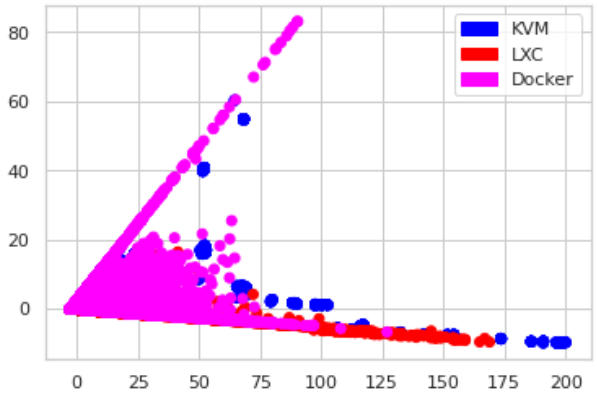


图 4：主成分分析图。X轴表示第一主成分，Y轴表示第二主成分。数据点表示正在分析的样本。向量指示最大方差的方向。每种颜色代表一种经过分析的技术：Docker、KVM 和 LXC。

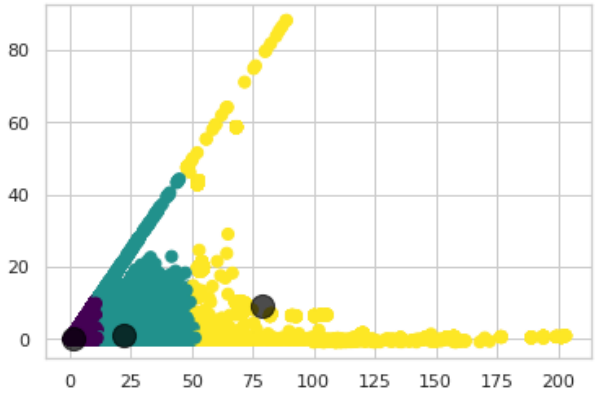


图5：K 均值聚类图。X轴和Y轴显示两个主成分的值，每个数据点根据其在这两个主成分上的值进行定位。数据点被分为 3 个簇，并以不同颜色表示。3 个簇的质心以各簇中心处的一个大点表示。

我们展示了使用该数据集构建的两个模型示例，分别根据场景变量估算响应主机的 CPU 使用率（CPU 模型）和平均往返时间（RTT 模型）。这些模型有助于预测数据集中未包含的场景的行为，例如频率范围介于 1Kpps 和 1Mpps 之间的场景，或虚拟机数量不同于表2所列数量的场景。借助这些模型，网络管理员可以根据设计目标，明智地选择最适合的虚拟化技术。我们使用回归技术构建这些模型，并借助 PyCaret 库 [12]。构建模型时所用的变量如表4所示。

在构建这些模型时，我们使用经过后处理的数据，这些数据已通过独热编码进行归一化。这一步对于模型有效地学习数据中的模式至关重要。

5.3.1 用例方法论

建立了一种方法，用于基于该数据集构建两个模型。首先，将数据集划分为训练集（80%）、测试集（10%）和验证集（10%）。使用 PyCaret 库，对训练集应用多达 19 种回归技术来构建模型。根据多项评估指标选出最佳技术，包括平均绝对误差（MAE）、均方误差（MSE）、均方根误差（RMSE）、决定系数（R^2）、均方根对数误差（RMSLE）、平均绝对百分比误差（MAPE）和训练时间（TT）。随后，使用五折交叉验证重新训练选定的技术，并重点进行超参数调优。

	输入变量	输出变量
CPU 型号	频率，Vm_数量，CPU 亲和性，中央处理器，输入输出，NET, HS, HR, DOCKER ，LXC, KVM_E1000 ，KVM_VIRTIO ，KVM_RTL8139 ，KVM_VMXNET3	CPU_HR_使用率
RTT 模型	频率，虚拟机_数量，CPU 亲和性、中央处理器、输入输出，NET, HS, HR, Docker ，LXC, KVM_E1000 ，KVM_VIRTIO ，KVM_RTL8139 ，KVM_VMXNET3	往返时间_Ping

表4：模型

RTT 模型结果 RTT 模型用于根据表4中所述的输入变量，估算给定场景下的平均 RTT。第一阶段训练使用了 17 种回归模型，结果见表5。从这些结果可以看出，基于决策树的模型优于其他模型。因此，下一阶段训练选用了基于随机森林技术的模型，因为它在三项误差指标——平均绝对误差（MAE）、均方误差（MSE）和均方根误差（RMSE）方面表现更优。

RTT 模型							
模型	MAE	MSE	均方根误差 决定系数		均方根对数误差 平均绝对百分比误差		训练时间（秒）
决策树回归器	**1.1343**	**14.8711**	**3.8560**	**0.7037**	**0.2978**	**0.1771**	**3.3290**
极端随机树回归器	**1.1343**	**14.8711**	**3.8560**	**0.7037**	**0.2978**	**0.1771**	128.8010
随机森林回归器	**1.1342**	**14.8699**	**3.8558**	**0.7037**	**0.2978**	**0.1771**	132.3980
CatBoost 回归器	1.1454	15.2269	3.9018	0.6966	0.2989	0.1780	164.1080
极端梯度提升	1.1490	15.2950	3.9106	0.6952	0.2989	0.1781	123.2300
轻量梯度提升机	1.1831	17.1895	4.1445	0.6575	0.3038	0.1791	10.0270
AdaBoost 回归器	1.8861	42.1433	6.4909	0.1606	0.5289	0.4500	40.0160
线性回归	2.2643	50.1294	7.0793	0.0015	0.7203	0.6261	6.5000
贝叶斯岭回归	2.2646	50.1469	7.0806	0.0011	0.7206	0.6253	2.1730
岭回归	2.2646	50.1467	7.0806	0.0011	0.7206	0.6253	1.2770
最小角回归	2.2646	50.1468	7.0806	0.0011	0.7206	0.6253	1.4100
Huber 回归器	2.2510	50.1597	7.0815	0.0008	0.7288	0.6099	9.1410
正交匹配追踪	2.4137	53.1145	7.2872	-0.0581	0.8037	0.7681	1.3610
套索回归	2.4605	55.0244	7.4171	-0.0962	0.8813	0.7342	1.4120
弹性网络	2.4605	55.0244	7.4171	-0.0962	0.8813	0.7342	1.4010
Lasso 最小角回归	2.4605	55.0244	7.4171	-0.0962	0.8813	0.7342	1.4910
虚拟回归器	2.4605	55.0244	7.4171	-0.0962	0.8813	0.7342	0.8080

表5：RTT 模型

以下是 RTT 模型调优步骤中选定的超参数：

随机森林(ccp_alpha=0.0,
准则='均方误差',
max_特征='自动' ,
最小_不纯度_下降=0.0,
最小_样本_叶节点=1,
最小_样本_分割=2,
最小_权重_比例_叶节点=0.0,
n_估计器数量=100，
n_作业=-1, 袋外_评分=False,
随机_状态=123)

CPU模型							
模型	MAE	MSE	RMSE	R2	均方根对数误差 平均绝对百分比误差		训练时间（秒）
决策树回归器	**0.3032**	**0.9062**	**0.9519**	**0.9992**	**0.0840**	**0.0391**	**3.4690**
极端随机树回归器	**0.3032**	**0.9062**	**0.9519**	**0.9992**	**0.0840**	**0.0391**	107.3600
随机森林回归器	**0.3032**	0.9064	0.9520	**0.9992**	**0.0840**	**0.0391**	94.8720
轻量梯度提升机	1.0115	4.1813	2.0445	0.9963	0.1142	0.0797	4.2710
CatBoost 回归器	0.9073	5.7624	2.3985	0.9949	0.0946	0.0585	106.5100
极端梯度提升	0.9535	6.1402	2.4767	0.9946	0.0965	0.0603	59.6990
AdaBoost 回归器	11.1936	441.8921	21.0114	0.6117	0.5464	0.5635	39.3820
Huber 回归器	12.8150	511.1350	22.6082	0.5509	0.7587	0.9763	6.6470
线性回归	14.1959	645.4639	25.4059	0.4329	0.7886	0.8921	2.3390
贝叶斯岭回归	14.1991	645.9363	25.4152	0.4325	0.7887	0.8917	2.5530
岭回归	14.1991	645.9339	25.4152	0.4325	0.7887	0.8917	1.4190
最小角回归	14.1998	645.9888	25.4162	0.4324	0.7887	0.8917	1.1970
梯度提升回归器	5.5745	680.9593	26.0342	0.4017	0.2841	0.2415	45.6180
正交匹配追踪	17.9799	1082.1288	32.8956	0.0492	0.9987	1.2008	1.0770
套索回归	20.8854	1438.2111	37.9236	-0.2636	1.3929	1.4613	1.2280
弹性网络	20.8854	1438.2111	37.9236	-0.2636	1.3929	1.4613	1.2450
Lasso 最小角回归	20.8854	1438.2111	37.9236	-0.2636	1.3929	1.4613	1.3330
虚拟回归器	20.8854	1438.2111	37.9236	-0.2636	1.3929	1.4613	0.6770
被动攻击回归器	35.1884	85362.3373	199.6070	-73.7373	1.0533	1.5621	2.0520

表 6：CPU模型

表7展示了所选模型第二个训练步骤的结果。读者可以看到，大多数指标均有所提升。借助该模型，管理员可以估算给定对象的往返时间

随机森林						
Fold	**MAE**	**MSE**	**RMSE**	**R2**	均方根对数误差 平均绝对百分比误差	
0	1.1527	15.0791	3.8832	0.7015	0.2992	0.1765
1	1.1366	15.1332	3.8901	0.6993	0.2967	0.1763
2	1.1459	15.1237	3.8889	0.7098	0.2980	0.1780
3	1.1209	14.6488	3.8274	0.7007	0.2960	0.1749
4	1.1232	14.1276	3.7587	0.7006	0.2994	0.1772
Mean	**1.1359**	**14.8225**	**3.8497**	**0.7024**	**0,2979**	**0.1766**
SD	0.0139	0.4377	0.0572	0.0042	0.0015	0.0012

表7：基于随机森林的 RTT 模型

场景，并确定他将使用哪种类型的虚拟化或驱动程序。

CPU 型号结果 构建的第二个模型是 CPU 型号。其目的是根据表4中列出的变量，针对给定场景估算 CPU 使用率的百分比。如表 6 所示，训练过程的第一阶段使用了 19 个回归模型。结果表明，基于决策树的模型优于其他模型。由于决策树技术相较于极端随机树回归器表现更优，其平均绝对误差、均方误差和均方根误差更低，因此被选用于下一步。此外，如训练时间（秒）所示，其训练时间比极端随机树回归器低 35 倍，因此在训练过程的下一阶段是更高效的选择。

以下是 hyp 在 CPU 型号调整中选择的参数 ng 步骤:

决策树回归器
(ccp_alpha=0.0, criterion='均方误差',
min_杂质_减少=0.0,
最小_不纯度_分割=无,
min_样本_leaf=1,
最小_样本_分割=2,
最小_权重_比例_叶节点=0.0,
随机 _状态=123,
分割器='最佳')

表8展示了所选模型第二个训练步骤的结果。可以看出，大多数指标均有所提升。

决策树						
fold	**MAE**	**MSE**	**RMSE**	**R2**	均方根对数误差	平均绝对百分比误差
0	0.3036	0.9071	0.9524	0.9992	0.0846	0.0391
1	0.3022	0.8969	0.9470	0.9992	0.0832	0.0393
2	0.3058	0.9244	0.9614	0.9992	0.0847	0.0394
3	0.3015	0.8934	0.9452	0.9992	0.0837	0.0389
4	0.3030	0.9103	0.9541	0.9992	0.0836	0.0387
Mean	**0.3032**	**0.9064**	**0.9520**	**0.9992**	**0.0840**	**0.0391**
SD	0.0015	0.0109	0.0057	0.0000	0.0006	0.0003

表8：基于决策树的往返时间模型

讨论 这些结果表明，在云虚拟化场景中，管理员可以使用这些模型来估算 CPU 或往返时间的使用率。这类洞见有助于选择虚拟化类型或驱动技术，并理解不同配置之间的权衡。即使场景采用的数据集未包含的往返时间测量频率（例如每秒 10 万个数据包），模型仍能准确估算此类场景的 CPU 使用率和往返时间 ping。这对于决策十分重要，因为无需再次执行实验。数据预处理所采用的方法使数据能够转换为适合机器学习的格式，从而让所应用的算法生成有意义的诊断结果。我们使用现有的后处理数据，通过各种回归技术创建模型。表现最佳的模型采用了基于决策树的方法，尤其是结合多个决策树来确定结果的方法。这主要是因为这些算法适用于结构化数据 [13]。这些算法使用多个基学习器，使其能够学习训练所用数据库中各项资源之间的复杂关系。

请注意，可以基于本研究提供的数据开发其他模型。可以应用分类模型；如有必要，用户可以将其他场景和组合的信息追加（扩充）到数据库中，从而开展进一步分析。

6 结论与未来工作

本研究探讨了在云计算环境中，虚拟化对性能指标、往返时间（RTT）和中央处理器使用率的影响。研究搭建了一个测试平台并开展实验，以获取与 RTT 和“中央处理器使用率”相关的数据集，并将其与场景的技术特征联系起来，例如虚拟化技术、驱动程序类型、CPU 亲和性、RTT 测量数据包的频率、负载位置（发送方或响应方）以及后台运行的虚拟机数量。

对所获得的数据进行了预处理，并运用多种数据科学技术（包括统计和数据分析）得出结论并观察潜在模式。结论表明，在响应主机上使用 Docker 会降低中央处理器使用率。与其他驱动程序相比，rtl8139 驱动程序会降低 KVM 虚拟化的性能。相反，VirtIO 驱动程序能够提升性能；此外，提高 RTT 测量数据包的频率意味着端到端延迟更低。

该数据集用于创建两个基于决策树的回归模型，根据网络的特征和参数估算往返时间（RTT）和 CPU 使用率。这些模型可帮助云管理员根据具体场景及其配置，决定何时使用虚拟化技术或驱动程序。例如，当管理员需要

估算 I/O 密集型负载应用的 CPU 使用率和往返时间时，管理员可以将场景参数输入模型，以判断结果是否符合限制条件，或者是否需要测试其他配置。

用于构建测试平台和开展实验的方法已公开，因此可以复现，并通过其他场景获取新数据。此外，本研究获取并使用的数据集已公开发布于 [5]，任何希望开展新评估或开发其他模型的人都可以使用。

最后，未来的工作将采用本研究未考虑的虚拟化技术（如 VMware、XEN 和 OpenVZ）生成其他场景，以获取更多数据。

参考文献

[1]S. A. Bello, L. O. Oyedele, O. O. Akinade, M. Bilal, J. M. Davila Delgado, L. A. Akanbi, A. O. Ajayi, and H. A. Owolabi, “建筑行业中的云计算：应用案例、优势与挑战”，*Automation in Construction*，第 122 卷，文章编号 103441，2021 年。 [在线]。获取地址：https://www.sciencedirect.com/science/article/pii/S0926580520310219[2] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio，“虚拟机与 Linux 容器性能比较的最新研究”，载于*2015 IEEE 系统与软件性能分析国际研讨会（ISPASS）*，2015 年，第 171–172 页。[3] Q. Du, Y. He, T. Xie, K. Yin, and J. Qiu，“一种收集 NFV 基础设施性能异常数据集的方法”，载于并行处理算法与体系结构，J. Vaidya 和 J. Li 编。Cham：Springer International Publishing，2018 年，第 59–71 页。[4] A. T. de Oliveira Filho, E. Freitas, P. R. X. do Carmo, D. H. Sadok, and J. Kelner，“往返时间与虚拟化的实验研究”，*Computer Communications*，第 184 卷，第 73–85 页，2022 年。 [在线]。获取地址：https://www.sciencedirect.com/science/article/pii/S0140366421004722[5] P. R. X. do Carmo, E. Freitas, A. T. de Oliveira Filho, and D. F. H. Sadok，“往返时间与虚拟化数据集”，2022 年 4 月。 [在线]。获取地址：https://doi.org/10.5281/zenodo.6459234[6] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala，“机器学习任务中数据质量的概述及其重要性”，载于第 *26* 届 *ACM SIGKDD* 知识发现与数据挖掘国际会议论文集，2020 年，第 3561–3562 页。[7] A. Géron，动手实践机器学习：使用 *Scikit-Learn*、*Keras* 和 *TensorFlow* 构建智能系统的概念、工具与技术。“O’Reilly Media, Inc.”，2019 年。[8] P. Cohen, S. G. West, and L. S. Aiken，行为科学应用多元回归/相关分析。Psychology Press，2014 年。[9] J. H. Zar，“斯皮尔曼秩相关：概述”，*WileyStatsRef: Statistics Reference Online*，2014 年。[10] H. Abdi and L. J. Williams，“主成分分析”，*Wiley interdisciplinary reviews: computationalstatistics*，第 2 卷，第 4 期，第 433–459 页，2010 年。[11] M. Ahmed, R. Seraj, and S. M. S. Islam，“k 均值算法：全面综述与性能评估”，*Electronics*，第 9 卷，第 8 期，2020 年。 [在线]。获取地址：https://www.mdpi.com/2079-9292/9/8/1295[12] M. Ali，*PyCaret*：*Python* 中一个开源、低代码的机器学习库，2020 年 4 月，pyCaret 1.0 版。[在线]。获取地址：https://www.pycaret.org[13] T. Chen and C. Guestrin，“XGBoost：一种可扩展的树提升系统”，载于第 *22* 届 *ACM SIGKDD* 知识发现与数据挖掘国际会议论文集，KDD ’16 系列。美国纽约州纽约市：ACM，2016 年，第 785–794 页。 [在线]。获取地址：http://doi.acm.org/10.1145/2939672.2939785