流处理系统中的性能优化：面向 Kafka Streams 的实验驱动的配置调优

David Chen 林茨约翰·开普勒大学，奥地利
chendavidyiwen@gmail.com

Sören Henning Dynatrace 研究部 林茨，奥地利
soeren.henning@dynatrace.com

Kassiano Matteussi JKU/Dynatrace 协同创新实验室，林茨约翰·开普勒大学 林茨，奥地利 kassiano.matteussi@jku.at

Rick Rabiser LIT CPS 实验室，林茨约翰·开普勒大学 林茨，奥地利
rick.rabiser@jku.at

arXiv:2603.04027v1 [cs.PF] 4 Mar 2026

摘要

为流处理系统配置高效性能，尤其是在云原生部署中，是一项具有挑战性且在很大程度上依赖人工的工作。我们提出了一种实验驱动方法，用于自动配置优化，该方法结合了三个阶段：通过拉丁超立方抽样进行初始探索、通过模拟退火进行引导式随机搜索，以及通过爬山算法进行局部优化。该工作流与云原生基准测试框架 Theodolite 集成，可在 Kubernetes 上自动编排实验，并提前终止表现不佳的配置。在使用 Kafka Streams 和基于 Kubernetes 的云测试平台进行的实验评估中，我们的方法找到了吞吐量较默认配置最高提升 23% 的配置。结果表明，结合提前终止的拉丁超立方抽样和模拟退火在探索配置空间方面尤为有效，而通过爬山算法进行额外微调的收益有限。

CCS 概念

• **软件及其工程** → **软件性能**； •**计算机系统组织** → **云计算**； •**信息系统** → **流管理**。

关键词

流处理、性能、配置调优

ACM 参考格式：
David Chen、Sören Henning、Kassiano Matteussi 和 Rick Rabiser。2026。《流处理系统中的性能优化：面向 Kafka Streams 的实验驱动的配置调优》。载于第*17*届 *ACM/SPEC* 国际性能工程会议论文集（*ICPE Companion '26*），*2026*年*5*月*4*日至*8*日，意大利佛罗伦萨。ACM，美国纽约州纽约，6页。https: //doi.org/10.1145/3777911.3800636

本作品采用 CreativeCommons Attribution4.0 InternationalLicense 许可。*ICPECompanion '26, Florence,Italy*© 2026 版权所有归所有者/作者所有。ACM
ISBN 979-8-4007-2326-1/2026/05
https://doi.org/10.1145/3777911.3800636

1 引言

流处理系统在金融、电子商务、物联网和软件监控等各个领域实现近实时数据分析方面发挥着至关重要的作用 [11]。此类系统持续对数据流进行筛选、转换和聚合，吞吐量高，延迟低，通常低于一秒 [29]。随着数据量不断增长，以及云中可扩展计算资源日益充足，流处理系统通常以分布式系统的形式实现，因此必须解决可扩展性、容错、资源效率、状态管理和数据分区等关键问题。

Apache Kafka Streams [32], 、Apache Flink [5], 和 Apache Spark [2] 等开源流处理框架内置了对这类需求的支持。然而，这些框架提供了大量配置参数，例如与内存管理、缓冲、批处理或缓存相关的参数，这些参数会显著影响延迟、吞吐量、可扩展性和可靠性等性能指标 [4, 8, 10, 12, 18, 21, 27, 33]。因此，要在容器化云环境中部署和运行此类系统并实现高性能颇具挑战。参数数量庞大且相互依赖关系复杂，实际中不可能穷举所有配置组合。因此，实践者往往依赖默认设置、昂贵的专家知识或临时的反复试错调优，导致大量性能潜力未能得到发掘。

本文提出了一种实验驱动方法，利用基于搜索的软件工程中的启发式方法，探索流处理系统庞大的配置空间，并找出能够提升性能的配置。我们使用最受欢迎的流处理框架之一 Kafka Streams 对该方法进行评估。借助 ShuffleBench [15]基准测试，我们在基于 Kubernetes 的云测试平台部署环境中评估了 265 种不同配置，发现其吞吐量较默认配置最高提升了 23%。

我们的贡献主要包括三个方面：(i) 一种结合拉丁超立方抽样、模拟退火和爬山算法的自动化配置优化方法；(ii) 与 Theodolite [13] 基准测试框架的云原生集成，用于在 Kubernetes 上自动编排实验；(iii) 一项实验性试点评估，深入分析各个优化组件的有效性。

2 背景与相关工作

关于流处理性能评估的相关研究表明，框架配置会对性能产生显著影响。例如，内存相关参数会影响 Spark [21] 中的吞吐量，而将逻辑组件映射到物理资源可能在 Flink [33], 中引发资源争用、倾斜和性能减缓，甚至会增加数据源与云服务提供商之间的延迟 [19]。基于这些研究成果，我们首先讨论文献中有关流处理系统配置优化的关键方法，然后介绍本文采用的三种优化技术。

2.1 参数优化方法

Herodotou 对大数据处理系统的自动参数优化方法进行了分类，[17] 对其中实验驱动方法与自适应方法之间的区别与本文尤为相关。

实验驱动方法。实验驱动方法依赖于在不同参数设置下反复执行应用程序，并由基于搜索的算法和实际运行结果提供指导。例如，Fischer 等人 [10]和 Jamshidi 和 Casale[18] 研究了利用高斯过程进行贝叶斯优化，以实现流处理系统的自动配置调优。Fischer 等人在 Apache Storm [26] 上评估了该方法，并将其与并行线性（最速）上升优化器进行比较，发现两者在小型和大型部署中的性能相当。Jamshidi 和 Casale 通过使用拉丁超立方设计初始化搜索，对这一思路进行了拓展；他们在 Apache Storm 基准测试中表明，与其他方法相比，这种做法进一步提高了优化效果。Bilal 和 Canini [4] 提出了一种混合方法，将经过修改的爬山算法与针对离散配置空间改进的拉丁超立方抽样技术以及一个基于规则的组件相结合。该方法交替进行基于采样的探索和局部搜索；在 Apache Storm 上开展的实验表明，其效果优于基线爬山算法。

自适应与基于学习的方法。文献中讨论了多种流处理系统的运行时自适应方法 [6, 28]。例如，Vysotska 等人 [31] 提出了一种整体自适应优化技术，将机器学习融入其中，以持续监控系统行为并在运行时调整配置参数，旨在提升异构流处理架构的端到端性能，而不是孤立地调优单个组件。与此类似，Bash- tovyi 和 Fechan [3] 认为，Kafka Streams 等现代框架中的静态配置机制会限制系统在可变工作负载下的效率。他们提出了一个自适应配置模块，在检测到性能偏差时调整参数：对于明确的情况，结合基于阈值的规则；对于模糊的情况，则通过微调过的大语言模型来选择配置值。

讨论。 虽然基于机器学习等技术的自适应方法能够减少人工干预，并在运行时响应工作负载动态变化，但它们不太适合确定稳定的部署时配置 [17]。相反，我们采用实验驱动的方法，尤其适用于缺乏先验配置信息、且需要在定义明确的工作负载下进行受控且可复现执行的场景 [17]。与前文提及的相关研究不同，我们的方法结合了拉丁超立方抽样、模拟退火和爬山算法等搜索技术（见下文）。此外，我们面向云原生 Kubernetes 环境，并聚焦于 Kafka Streams——这一特别适用于云原生应用（如事件驱动微服务）中流处理的框架 [14, 20]。

2.2 采用的搜索技术

尽管存在许多基于搜索的优化技术，我们的方法基于以下三种技术：

拉丁超立方抽样是一种统计技术，用于在高维参数空间中生成具有代表性的配置集合。每个参数范围都被划分为概率相等的区间，并构造样本，使得对于每个参数，每个区间在所有样本中恰好使用一次，从而以相对较少的样本生成覆盖空间的配置集合 [22]。

模拟退火是一种元启发式算法，它通过反复对当前配置进行小幅修改，并以概率决定是否接受这些修改，从而探索配置空间。逐渐降低的温度控制这一概率：较高的温度最初允许频繁接受较差的配置，以便进行探索；而较低的温度会减少此类接受，并使搜索集中于改进 [1]。

爬山算法是一种类似的技术，它会反复提出相邻配置，但只接受能够改进目标的配置 [4, 24]。

3 性能优化方法

我们提出了一种方法，可自动探索流处理应用的配置空间，以确定能够优化吞吐量等性能指标的设置。该方法的输入包括：(i) 一个流处理应用或基准测试；以及 (ii) 针对该应用或底层流处理框架的配置参数规范。

优化以迭代方式进行：每一步中，该方法都会提出一个新配置，在该配置下运行应用或基准测试，并记录性能指标。为平衡探索与利用，搜索分为三个阶段，并采用互补的优化技术：使用拉丁超立方抽样进行初始的空间填充式探索，使用模拟退火进行引导式随机搜索，并使用爬山算法在有潜力的配置附近进行局部优化。整个工作流与 Theodolite 基准测试框架集成 [13], ，该框架负责协调实验的自动执行和测量数据的收集。

3.1 阶段1：拉丁超立方抽样

初始阶段的目标是找出一组多样化的配置，使其性能达到或超过默认配置。为此，该方法首先根据预定义的参数

规范，使用最大最小拉丁超立方抽样（LHS）实现生成一组候选配置。每个样本都是一个 n维点，其中每个维度对应一个配置参数。拉丁超立方抽样过程会为每个维度生成范围为 $[0, 1]$ 的归一化值。随后，根据参数定义中指定的分布（例如线性缩放或对数缩放），将这些归一化值映射到实际参数范围。然后执行每个生成的配置并测量其性能。最后，将每个采样配置的测量性能与默认配置进行比较，并选出性能最佳的配置，作为后续优化阶段的起点。

3.2 阶段2：模拟退火

第二阶段采用模拟退火，从阶段1中最有前景的配置邻域进行探索。在每次迭代中，算法会对可配置参数的一个子集进行小幅调整，并评估所得配置。如果新配置能够带来更高的性能，则将其作为新的当前解。否则，算法会以一定概率接受该配置，该概率取决于性能差异以及温度参数；该参数从某个分布中采样。此过程会重复执行，直到达到可配置的迭代次数。较高的初始温度会增加接受较差配置的可能性，从而在逐步聚焦于更优区域之前，改善对配置空间的探索。

3.3 阶段3：爬山算法

阶段2结束后，对于每个初始样本，将从模拟退火中获得的性能最佳配置作为最终、以利用为重点阶段的起点。该阶段采用爬山算法探索局部邻域，并减少每次迭代中要更改的参数数量。与模拟退火不同，算法只接受能够提升性能的配置，从而推动搜索过程围绕已知最佳配置趋向局部最优解。

3.4 云原生集成

所提出的方法以扩展形式[1] 集成到云原生 Theodolite 基准测试框架中，以自动化在 Kubernetes 上部署和执行实验，

如图 1 所示。对于每个待评估的配置，我们的优化工作流都会生成一份采用 Theodolite 自定义资源格式的基准测试执行清单 [13]，并将其提交到 Kubernetes 集群。随后，Theodoliteoperator 会获取这些清单，并负责调度相应的基准测试运行。为了降低资源消耗、云成本和总体实验时间，我们还集成了提前停止机制：该机制根据可配置的指标识别明显表现不佳的配置，并在其执行完成前将其终止。

4 实验性试点评估

我们通过在受控但有限的云环境中对 KafkaStreams 基准测试应用优化方法，开展了试点评估。虽然这些结果并非旨在构成

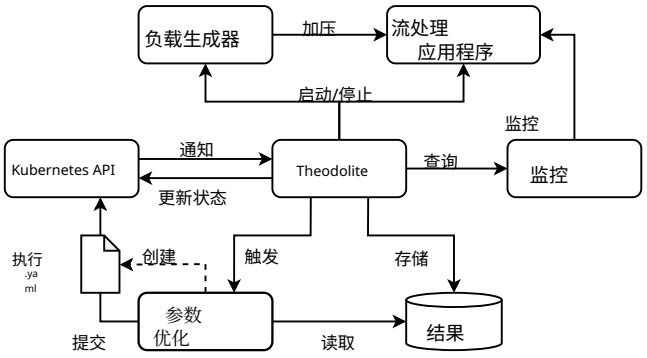

图 1：我们的优化方法与 Theodolite 的云原生架构集成 [13]。

一项全面或完全可推广的研究，但它们为三个优化阶段如何相互作用，以及它们能为 KafkaStreams 应用程序带来哪些性能提升，提供了初步见解。

4.1 方法与设置

评估在通过 CloudLab 测试平台 [9], 配置的专用 Kubernetes 集群上进行，该集群由三台 *m510* 实例组成。与公有云环境 [16] 相比，这种设置降低了实验间的变异性。我们使用 ShuffleBench [15] 基准测试及其 Kafka Streams [23, 32] 实现作为工作负载。在 Kubernetes 中，我们部署了三个 Kafka Streams 实例，每个实例限制为一个 CPU 核心和 16 GiB 内存。基准测试配置沿用我们先前的工作 [15, 16], , 但使用 1000 万个消费者和 128 字节的记录，以将负载从 Kafka 消息层转移到 Kafka Streams 框架。每种配置都在一项 8 分钟的实验中进行测试（其中包括 3 分钟预热），并根据这一时段内的平均吞吐量优化性能。

在选择 Kafka Streams 配置参数时，我们参考了厂商白皮书 [34] 和 Kafka Streams 官方文档。我们总共确定了九个参数（包括一个 RocksDB 参数、三个 Kafka 消费者参数和两个 Kafka 生产者参数），列于表1。对于这九个参数中的七个，我们在从归一化配置空间映射到实际参数范围时采用指数缩放。

4.2 优化阶段评估

我们分析了各优化阶段对整体搜索过程的贡献，以及由此实现的性能提升。[2]

4.2.1 拉丁超立方抽样。在第一阶段，我们使用 maximin拉丁超立方采样进行五次迭代，生成一组初始配置，共得到 30 个样本。作为基线，我们还加入了默认配置。每种配置均执行三次。为筛除存在故障、不稳定或性能明显较差的配置，我们采用提前终止策略：如果吞吐量连续 90 秒低于基线的 30%，或连续 5 分钟低于基线的 50%，则停止执行。

[1]https://github.com/DavidChenY/theodolite-paropt

[2]实验的原始数据作为补充材料提供 [7]。

表1：各阶段后性能最佳的配置，按吞吐量排序。配置$c_{x,y,z}$表示拉丁超立方抽样阶段的第 x个样本、模拟退火的第 y次迭代以及爬山算法的第 z次迭代。

	Kafka Streams 配置参数										结果	
标识符	cache.max.bytes.buffering	buffered.records.per.partition	consumer.fetch.min.bytes	commit.interval.ms	producer.linger.ms	consumer.max.partition.fetch.bytes	producer.batch.size	consumer.max.poll.records	write.buffer.size (RocksDB)	吞吐量（记录/秒）	+/− to previous phase
c_{29}	179.4 KB	2.9 M	2.1 KB	5 258	55	2.3 MB	73 B	1.3 G	652.1 MB	11 216 (−39.4%)	
c_{12}	2.1 KB	1.2 G	127.8 KB	4 359	121	227.3 KB	148 B	165.5 M	50.6 MB	11 630 (−37.2%)	
c_{20}	766.7 KB	1 835	503 B	580	166	220.0 MB	335 B	24	37.7 MB	12 118 (−34.5%)	
c_{18}	12 B	24.0 M	113.5 MB	2 326	198	459.6 KB	4.6 KB	2	255.4 MB	16 003 (−13.5%)	
c_{24}	9.2 MB	9 223	3.2 KB	856	48	7.8 MB	52.4 KB	517	1.3 GB	17 070 (−7.8%)	
c_{28}	519 字节	485.1 千	30.4 兆字节	5 137	67	746.2 千字节	106.3 千字节	271	1.4 GB	17 351 (−6.2%)	
c_{19}	150 B	95.8 K	16.4 KB	3 475	102	6.4 KB	30.0 KB	95.1 M	80.2 MB	17 672 (−4.5%)	
c_{default}	10.5 MB	1 000	1	5 000	0	1 MB	16 KB	500	4 MB	18 506 (±0.0%)	
c_{10}	262.5 MB	44.8 M	8.6 MB	6 621	97	932.6 KB	4.0 KB	57.3 K	69.0 MB	19 305 (+4.3%)	
c_{23}	377.5 KB	3 089	1.1 MB	1 400	189	143.6 KB	709.4 KB	219.2 K	1.9 GB	19 668 (+6.3%)	
c_{11}	63.1 KB	4 485	874 B	1 646	145	25.0 MB	202.5 KB	2 621	101.4 MB	20 788 (+12.3%)	
c_{1}	72.6 MB	69.3 M	143.3 MB	4 195	209	4.8 MB	122.1 KB	10.2 K	1.1 GB	21 156 (+14.3%)	
c_{27}	2 B	4	5.3 MB	3 716	216	510.1 MB	766.7 KB	2.4 M	183.0 MB	21 211 (+14.6%)	
$c_{10,18}$	2.3 MB	10.4 M	8.6 MB	6 788	172	2.0 MB	4.5 KB	23.8 K	66.6 MB	19 585 (+5.8%)	+1.5%
$c_{19,17}$	24 B	5 937	16.4 KB	3 968	338	6.4 KB	39.1 KB	136.2 M	110.1 MB	20 422 (+10.3%)	+15.6%
$c_{23,\ 14}$	54.8 KB	24.6 K	1.1 MB	2 013	381	239.4 KB	425.8 KB	8.2 M	1.7 GB	21 782 (+17.7%)	+10.7%
$c_{27,\ 20}$	2 B	32	5.3 MB	2 598	310	512.0 MB	1.0 MB	41.8 K	167.5 MB	22 028 (+19.0%)	+3.8%
$c_{11,7}$	13.0 KB	2 440	874 B	1 176	312	33.6 MB	202.5 KB	25	112.7 MB	22 172 (+19.8%)	+6.6%
$c_{1,\ 13}$	71.2 MB	28.7 M	143.3 MB	3 124	442	22.7 MB	122.6 KB	14.2 K	1.1 GB	22 800 (+23.2%)	+7.8%
$c_{19,17,16}$	38 B	5 937	16.4 KB	3 968	451	6.4 KB	39.1 KB	136.2 M	147.0 MB	21 170 (+14.4%)	+3.7%
$c_{23,10,4}$	113.6 KB	35.4 K	1.1 MB	1 436	381	239.4 KB	547.0 KB	8.2 M	1.6 GB	21 767 (+17.6%)	+1.2%
$c_{27,20,15}$	2 B	32	5.3 MB	2 598	310	512.0 MB	1.0 MB	102.0 K	167.5 MB	22 010 (+18.9%)	−0.1%
$c_{11,7,11}$	13.0 KB	2 440	874 B	495	312	33.6 MB	202.5 KB	25	112.7 MB	22 246 (+20.2%)	+0.3%
$c_{1,\ 13,\ 8}$	71.2 MB	28.7 M	143.3 MB	3 051	442	22.7 MB	122.6 KB	14.2 K	1.1 GB	22 712 (+22.7%)	−0.4%

0 5k 10k 15k 20k

在 30 个生成的配置中，有 18 个被提前终止。在 12 个完成执行的配置中，只有五个的吞吐量高于基线，如表1所示。最差的配置使吞吐量下降约 39%，而表现最佳的配置则.....

性能约提升了 15%，超越基线。进一步检查发现，消费者单分区最大抓取字节数的值特别低时，往往会导致提前终止。

对于完整执行的 12 种配置，我们对配置参数与测得的吞吐量进行相关性分析，如图 2 所示。我们发现，生产者批次大小与吞吐量呈非常强的正相关，生产者等待时间（毫秒）和消费者最小抓取字节数则呈中等偏强的正相关。此外，我们发现每个分区的缓冲记录数与吞吐量呈弱负相关。

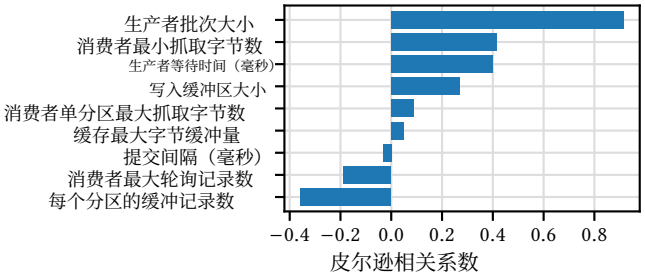

图2：参数与吞吐量的相关性。

4.2.2 模拟退火。在模拟退火阶段，我们继续使用六种表现与默认配置相近或更好的配置。对于每种配置，我们运行 25 次迭代，共评估 150 种配置。我们根据指定的吞吐量损失 2500 记录/秒（比基线吞吐量低约 14%），利用玻尔兹曼方程推导出初始温度，以便在开始时仍以 75% 的概率接受这一损失；同时采用广泛使用的指数冷却函数，冷却速率为 0.95。由于这一阶段有意侧重于探索而非利用，我们为较差的配置选择了较高的接受率，并且不进行重复运行。每次迭代中，两个参数都会在归一化参数尺度上按 [−10% +10%] 的范围进行调整。

在所有模拟退火运行中，仅有七次执行被提前终止，其中五次来自阶段1中性能最低的配置（c_{19}）。图3展示了观测到的吞吐量如何随迭代演变。较高的接受率（以及固有的性能波动）体现在吞吐量的频繁下降上。尽管如此，六个初始配置中有四个的吞吐量从第一次迭代到最后一次迭代有所提升。总体而言，配置 c_1在第13次迭代中达到了最高吞吐量，相较基线配置提升约23%，相较本阶段自身起点提升约8%（再次参见表1）。

4.2.3 爬山算法。在第三阶段，我们将重点从探索转向利用，并使用爬山算法围绕模拟退火得出的最佳结果微调配置。由于没有任何

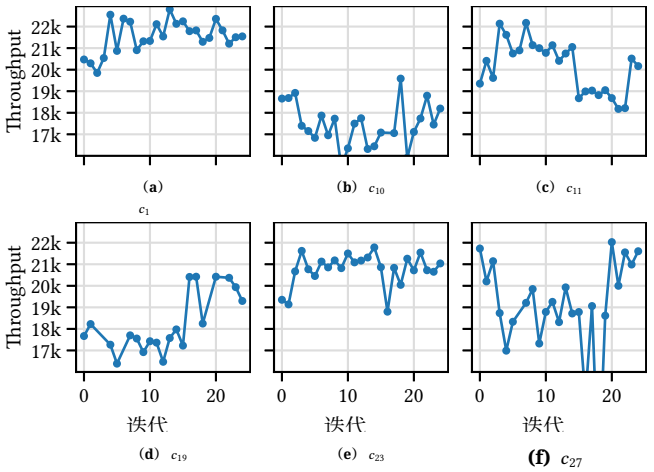

图3：不同初始配置下，吞吐量（记录/秒）随模拟退火迭代的演变。

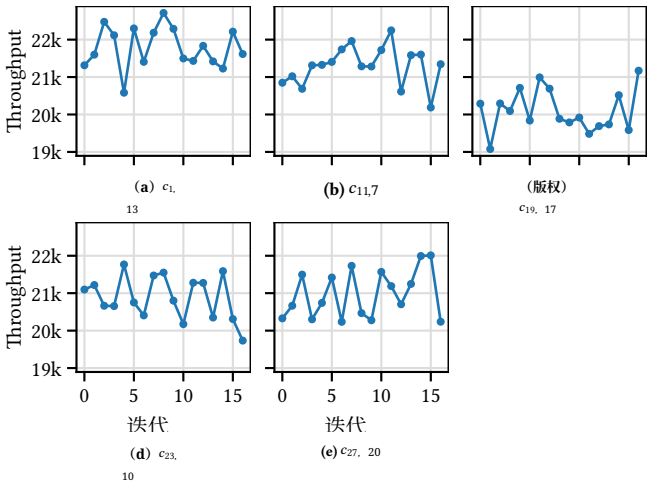

图4：不同初始配置下，吞吐量（记录/秒）随爬山法迭代的演变。

在 c_{10} 上的迭代达到 20 000 记录/秒的吞吐量，因此这些迭代不再纳入后续微调。为体现对局部优化的侧重，我们将迭代次数减少到 17。每次迭代中，都会在归一化参数尺度上，将单个参数调整 [−10% +10%] 范围内的一个值。

图4总结了爬山算法阶段的结果。四个最佳配置均源自$c_{1,13}$，该配置在模拟退火阶段已是表现最佳的配置。然而，在所有起点上，我们均未观察到显著改进（另见表1）。尽管爬山算法被设计为只接受更优的配置，吞吐量在各次迭代中仍略有上下波动，表明总体结果变异性占主导地位。将初始配置和最佳爬山算法配置各重复执行三次，所得差异仅为 −4.2%至 +2.9 %，证实表面上的增益处于基准测试的自然变异范围内。

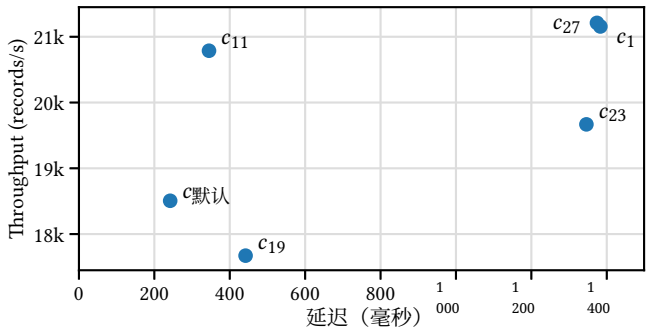

图5：所选配置的延迟与吞吐量。

4.3 讨论

我们的试点评估表明，所提出的方法能够显著提高 Kafka Streams 应用程序的吞吐量，相较默认配置约提升 23%。具体而言，拉丁超立方抽样与模拟退火的组合在识别并优化配置空间中有前景的区域方面被证明是有效的，而最后使用爬山算法进行的微调步骤并未带来超出基准测试自然变异范围的明显额外改进。

实验的执行时间是影响实际适用性的一个关键因素。在这方面，提前终止机制非常有益：在第一阶段的实验中，约 60 % 的配置被提前终止，使总体执行时间缩短了 50% 以上。我们最近量化了同一基准测试在公有云中的性能变异性 [16], , 这为将变异性信息纳入搜索过程创造了机会。

我们所展示的试点评估聚焦于吞吐量，将其作为首要目标。然而，在实际场景中，延迟和容错等其他属性也同样重要。尽管容错实验相对复杂， [25, 30],我们对部分配置进行了探索性延迟分析，如图 5 所示。虽然所有吞吐量高于默认配置的配置也都会增加延迟，但我们并未发现绝对吞吐量与延迟之间存在明确相关性，这表明未来工作可通过多目标优化明确权衡这些取舍。

5 结论与展望

本文介绍了我们针对流处理系统配置优化所采用的自动化实验驱动方法，尤其关注 Kubernetes 等云原生部署。在实验评估中，与默认配置相比，我们的方法使 Kafka Streams 基准测试的吞吐量最高提升了 23%。大部分性能提升来自拉丁超立方抽样与模拟退火算法的结合，以及对表现不佳或存在故障的配置进行提前终止。通过爬山算法进一步微调仅带来了微乎其微的改进。未来，我们计划在公有云环境中开展更广泛的评估，将该方法扩展到多目标优化（例如吞吐量和延迟），并将其应用于其他流处理框架。

致谢

我们谨此感谢林茨约翰·开普勒大学和 Dynatrace 共同资助本研究。

参考文献

[1] Emile Aarts、Jan Korst 和 Wil Michiels。2005。模拟退火。《搜索方法学：优化与决策支持技术入门教程》（2005）。https://doi.org/10.1007/0-387-28356-0_7[2] Michael Armbrust、Tathagata Das、Joseph Torres、Burak Yavuz、Shixiong Zhu、Reynold Xin、Ali Ghodsi、Ion Stoica 和 Matei Zaharia。2018。结构化流处理：Apache Spark 中面向实时应用的声明式 API。载于*2018*年数据管理国际会议论文集（*SIGMOD '18*）。ACM， 13 10 1145 3183713 3190664 页。https://doi.org/ . / . Artem Bashtovyi 和 Andrii Fechan。 . 利用系统指标开发数据流处理系统的自适应重新配置方法。《技术审计与生产储备期刊》6（2025）。https://doi.org/10.15587/2706- 5448.2025.344185[4] Muhammad Bilal 和 Marco Canini。2017。迈向流处理系统的自动参数调优。载于*2017*年云计算研讨会论文集（*SoCC '17*）。计算机协会，189–200。https: //doi.org/10.1145/3127479.3127492[5] Paris Carbone、Asterios Katsifodimos、Stephan Ewen、Volker Markl、Seif Haridi 和 Kostas Tzoumas。2015。Apache Flink：在单一引擎中实现流处理与批处理。*IEEE*计算机学会数据工程技术委员会通报36，4（2015）。[6] Valeria Cardellini、Francesco Lo Presti、Matteo Nardelli 和 Gabriele Russo Russo。2022。数据流处理系统的运行时自适应：最新进展。*ACM*《计算调查》 54, 11 2022 10 1145 3514496 s（ ）。https://doi.org/ . / David Chen。 .《流处理系统自动参数优化》复现包。https://doi.org/ 10 5281 14901857 . /zenodo。Paulo Ricardo Rodrigues De Souza、Kassiano J. Matteussi、Alexandre Da Silva Veith、Breno F. Zanchetta、Valderi R. Q. Leithardt、Álvaro L. Murciego、Edison Pignaton De Freitas、Julio C. S. Dos Anjos 和 Claudio F. R. Geyer。2020。借助 BurstFlow 提升云端大数据流处理应用的性能。《*IEEE Access*》期刊8（2020）。https://doi.org/10.1109/ACCESS.2020.3042739[9] Dmitry Duplyakin、Robert Ricci、Aleksander Maricq、Gary Wong、Jonathon Duerig、Eric Eide、Leigh Stoller、Mike Hibler、David Johnson、Kirk Webb、Aditya Akella、Kuangching Wang、Glenn Ricart、Larry Landweber、Chip Elliott、Michael Zink、Emmanuel Cecchet、Snigdhaswin Kar 和 Prabodh Mishra。2019。cloudlab 的设计与运行。载于*2019*年*USENIX*年度技术会议论文集（*USENIX ATC '19*）。USENIX 协会，14页。[10] Lorenz Fischer、Shen Gao 和 Abraham Bernstein。2015。机器调优机器：利用贝叶斯优化配置分布式流处理器。载于*2015*年*IEEE*集群计算国际会议论文集（*CLUSTER '* *15* 9 10 1109 2015 13 ）。页。https://doi.org/ . / CLUSTER. . Marios Fragkoulis、Paris Carbone、Vasiliki Kalavri 和 Asterios Katsifodimos。2024。流处理系统演进综述。《*VLDB*期刊》 33, 2 2024 10 1007 00778 023 00819 8 ( )。https://doi.org/ . /s - - - Sören Henning 和 Wilhelm Hasselbring。 . Theodolite：微服务架构中分布式流处理引擎的可扩展性基准测试。《大数据研究》 25 2021 10 1016 2021 100209 ( )。https://doi.org/ . / j.bdr. . Sören Henning 和 Wilhelm Hasselbring。 . 一种用于基准测试云原生应用可扩展性的可配置方法。《实证软件工程期刊》 27, 6 2022 10 1007 10664 022 10162 1 ( )。https://doi.org/ . /s - - - Sören Henning 和 Wilhelm Hasselbring。 . 在云中以微服务形式部署的流处理框架的可扩展性基准测试。《系统与软件期刊》 208 2024 10 1016 2023 111879 ( )。https://doi.org/ . /j.jss. . Sören Henning、Adriano Vogel、Michael Leichtfried、Otmar Ertl 和 Rick Rabiser。2024。ShuffleBench：分布式流处理框架大规模数据洗牌操作的基准测试。载于第*15*届 *ACM/SPEC* 国际性能工程会议论文集（*ICPE '24*）。ACM，12页。https://doi.org/ 10 1145 3629526 3645036 . / . Sören Henning、Adriano Vogel、Esteban Perez-Wohlfeil、Otmar Ertl 和 Rick Rabiser。2025。我应该何时运行应用程序基准测试？以流处理应用为例研究云性能变化。载于第*33*届 *ACM* 软件工程基础国际会议论文集（*FSE Companion '25*）。ACM，11页。https: // doi.org/10.1145/3696630.3728563 [17] Herodotos Herodotou、Yuxing Chen 和 Jiaheng Lu。2020。大数据处理系统自动参数调优综述。*ACM*《计算调查》53，2（ 2020 10 1145 3381027 ）。https://doi.org/ . / Pooyan Jamshidi 和 Giuliano Casale。2016。面向流处理系统最优配置的不确定性感知方法。载于第*24*届 *IEEE* 计算机与电信系统建模、分析与仿真国际研讨会论文集（*MASCOTS '16*）。IEEE，9页。https://doi.org/ 10.1109/MASCOTS.2016.17[19] Styliani Kyrama 和 Anastasios Gounaris。2026。复杂事件处理：当前状况及边缘部署考量。《未来一代计算机系统》 177 2026 10 1016 2025 108247 ( )。https://doi.org/ . /j.future. . Rodrigo Laigner、Ana Carolina Almeida、Wesley K. G. Assunção 和 Yongluan Zhou。2025。微服务架构中事件管理挑战的实证研究。*ACM*《软件工程与方法学汇刊》（2025）。https: // doi.org/ 10 1145 3776581 . / Kassiano J. Matteussi、Julio C. S. dos Anjos、Valderi R. Q. Leithardt 和 Claudio F. R. Geyer。2022。面向数据密集型流水线的 Spark Streaming 反压性能评估分析。《传感器期刊》22，13（2022）。https://doi.org/ 10 3390 22134756 . /s Matthieu Petelet、Bertrand Iooss、Olivier Asserin 和 Alexandre Loredo。2010。带不等式约束的拉丁超立方抽样。《*AStA*统计分析进展》94（2010）。https://doi.org/10.1007/s10182-010-0144-z [23] Matthias J. Sax、Guozhang Wang、Matthias Weidlich 和 Johann-Christoph Freytag。2018。流与表：同一事物的两面。载于实时商业智能与分析国际研讨会论文集（*BIRTE '* *18* 10 10 1145 3242153 3242155 ）。ACM，页。https://doi.org/ . / . Bart Selman 和 Carla P Gomes。 . 爬山搜索。John Wiley & Sons, Ltd. https://doi.org/ 10 1002 0470018860 00015 . / .s Jawad Tahir、Ruben Mayer、Christoph Doblander 和 Hans-Arno Jacobsen。2024。流有多可靠？流处理系统的端到端处理保证验证与性能基准测试。载于*2024*年*VLDB*基金会论文集（*VLDB Endow. '24*）。ACM，14页。https://doi.org/10. 14778/3712221.3712227[26] Ankit Toshniwal、Siddarth Taneja、Amit Shukla、Karthik Ramasamy、Jignesh M. Patel、Sanjeev Kulkarni、Jason Jackson、Krishna Gade、Maosong Fu、Jake Donham、Nikunj Bhagat、Sailesh Mittal 和 Dmitriy Ryaboy。2014。Storm@twitter。载于*2014*年 *ACM SIGMOD* 数据管理国际会议论文集（*SIGMOD '* *14* 10 10 1145 2588555 2595641 ）。ACM，页。https://doi.org/ . / . Giselle van Dongen 和 Dirk van den Poel。 . 流处理框架评估。*IEEE*《并行与分布式系统汇刊》31，8（2020）。https://doi.org/ 10 1109 2020 2978480 . /TPDS. . Adriano Vogel、Dalvan Griebler、Marco Danelutto 和 Luiz Gustavo Fernandes。2022。并行流处理的自适应：系统综述。《并发与计算：实践与经验》34，6（2022）。https://doi.org/ 10 1002 6759 . /cpe. Adriano Vogel、Sören Henning、Otmar Ertl 和 Rick Rabiser。2023。分布式流处理系统性能的系统映射研究。载于*Euromicro*软件工程与高级应用会议论文集（*SEAA '* *23* 7 10 1109 60479 2023 00052 ）。IEEE，页。https://doi.org/ . /SEAA . . Adriano Vogel、Sören Henning、Esteban Perez-Wohlfeil、Otmar Ertl 和 Rick Rabiser。2024。流处理框架故障恢复的综合基准测试分析。载于第*18*届 *ACM*分布式与事件驱动系统国际会议论文集（*DEBS '24*）。ACM，12页。https://doi.org/ 10 1145 3629104 3666040 . / . Victoria Vysotska、Iryna Kyrychenko、Vadym Demchuk 和 Iryna Gruzdo。2024。分布式数据流处理系统的整体自适应优化技术。载于第*8*届计算语言学与智能系统国际会议论文集（*COLINS '24*）。Ceur，13页。https: // doi.org/10.31110/COLINS/2024-2/009[32] Guozhang Wang、Lei Chen、Ayusman Dikshit、Jason Gustafson、Boyang Chen、Matthias J. Sax、John Roesler、Sophie Blee-Goldman、Bruno Cadonna、Apurva Mehta、Varun Madan 和 Jun Rao。2021。一致性与完整性：重新思考 Apache Kafka 中的分布式流处理。载于*2021*年数据管理国际会议论文集（*SIGMOD/PODS '* *21*）。ACM， 12 10 1145 3448016 3457556 页。https://doi.org/ . / . Yuanli Wang、Lei Huang、Zikun Wang、Vasiliki Kalavri 和 Ibrahim Matta。2025。CAPSys：面向数据流处理的争用感知任务放置。载于第*20*届欧洲计算机系统会议论文集（*EuroSys '25*）。ACM。https://doi.org/ 10 1145 3689031 3696085 . / . Byzek Yewa。 . 在 Confluent Cloud 上开发 Apache Kafka® 应用程序的最佳实践。 h t t p s:// www.confluent.io/resources/ebook/recommendations-developers-using-confluent-cloud/