### 用于数据分析的异构云存储配置

摘要

数据分析是公有云服务上一类重要的数据密集型工作负载。然而，为这些应用选择正确的计算与存储配置是很困难的，因为可用选项空间很大和选项之间的交互非常复杂。此外，分析型工作负载访问的不同数据流具有不同的特征，这些特征可以被不同的存储设备更好地服务。

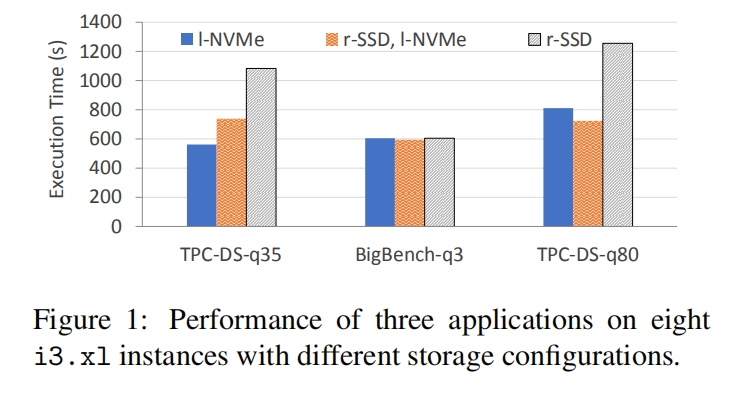
我们介绍selecta，这是一个为数据分析工作负载推荐近最优的云计算和存储资源配置的工具。Selecta使用潜在因素协同过滤来预测应用在不同配置下的性能，基于分析训练工作负载收集的稀疏数据。我们使用上百个SQL和ML应用对selecta进行评估，结果显示selecta选择近最优的性能配置（最优10%内）的概率为94%，近最优的成本配置的概率为80%。我们还使用selecta得出关于云存储系统的重要见解，包括NVME闪存设备的性能-成本效率，云存储关于支持细粒度容量和带宽分配的需求，端到端存储优化的动机。

#### 介绍

随着亚马逊AWS、谷歌云平台和微软AZure把它们的工作负载转移到平台上，公共云市场正经历着前所未有的增长。除了高弹性之外，公有云还承诺减少所有租户的总成本，因为资源是被租户共享的。然而，实现性能和成本效率要求为每一个给定的应用选择合适的配置。不幸的是，大量可用的实力类型和配置选项使为应用选择合适的资源非常困难。

存储选择通常是必要的，特别是对数据密集型分析的云部署。云厂商提供了各种各样的存储选项，包括对象、文件和块存储。块存储可以由硬盘、固态驱动器和高带宽、低延迟的NVMe组成。上述设备可能在运行应用程序的云实例本地或者远程。仅这些选项就会导致存储配置选项在吞吐量、延迟和位成本方面出现数量级的差异。随着基于3D-X poind 的新兴技术的出现，云存储领域变得越来越多样化。

选择正确的云存储配置对性能和成本来说都很重要。考虑SPARK SQL在两张128GB的表上等价查询的结果，我们发现8个EC2节点集群中的实例访问r-HDD比l-NVME查询花费时间要长8.7倍。这与最近的研究形成对比。与之前的spark相比，发现更快的存储只能将任务执行时间中位数最多提高19%。尽管NVMe单位时间成本更高，但NVMe的性能优势使该查询的执行成本降低8倍。如果我们还考虑核数和内存实例的一些选项，性能最好和性能最差VM存储配置之间的性能差距超过30倍。



为分析型应用确定云配置具有挑战性。即使我们将自己限制为单个实例类型并专注于优化性能，为特定应用程序选择存储配置仍然很重要。图一比较了三个spark应用程序的性能，这三个程序分别运行在存储设备分别为I-NVMe、r-SSD和混合(r-SSD用于输入/输出数据，l-NVMe用于中间数据)的i3.8xlarge EC2实例。第一个应用程序是I / O绑定的，并且受益于NVMe Flash的高吞吐量。第二个应用程序有CPU瓶颈，因此对三个存储选项配置表现性能一样。第三个应用程序是I/O绑定的，并且在混合存储选项表现性能最好，因为它最大限度的减少了读写I/O之间的干扰，读写I/Os在Flash上的性能是不对称的。这个结果不应该令人惊讶。分析工作负载访问多个数据流，包括输入和输出文件、日志和中间数据（例如混洗和广播）。每个数据流在访问频率、访问方式和数据生存时间方面都有不同的特征，这些特征使得不同的数据流更适合不同的存储设备。例如表1中的TPC-DS query 80应用程序，输入数据存储在r-SDD和中间数据存储在NVMe闪存比所有数据存储在NVMe表现得更好，因为它隔离了流并且消除了干扰。

我们介绍selecta，一种学习近最优VM和存储配置的工具，用于用户指定的性能成本目标的分析型应用程序。Selecta的目标是对新到达的数据进行频繁或定期重新运行的分析工作[1,25,55]。配置由云实例类型（核心计数和内存容量）以及用户输入/输出数据和中间数据的存储类型和容量定义。Selecta使用稀疏性能数据训练在各种云配置上配置的应用程序，以及仅在两个配置上配置的目标应用程序的性能度量。为了预测不同配置的应用性能，selecta使用潜因子协同过滤，这是一种普遍应用在推荐系统中的机器学习方法[10,57,11,22,23]。Selecta将稀疏性能数据用于配置在各种云配置上的训练应用程序，以及仅在两种配置上配置的目标应用程序的性能测量。与穷举搜索相比，selecta利用稀疏的训练数据来显著更快、更经济高效地学习。该方法还对CherryPick和Ernest等最新的系统进行了改进，这些系统的性能预测模型需要更多关于目标应用程序的信息，因此需要更多应用程序才能收敛[3,69]。更多的是，过去的工作没有考虑异构云存储选项或者每个应用程序中不同数据流的不同偏好[71]。

我们使用超过一百个Spark SQL和ML工作负载来评估selecta，每个工作负载都有两种不同的数据集缩放因子。我们证明selecta选择近最优性能配置的概率为94%（在最优的10%以内）和近最优的成本配置的概率为80%。我们还分析了selecta对各种参数的影响性，例如训练工作负载或者目标应用程序的可用信息量。

我们工作的关键贡献是我们通过分析型工作负载对云存储系统及其使用情况进行了分析，这带来了一些重要的见解。我们发现除了提供最佳性能之外，基于NVMe的的配置还为各种应用程序提供较低的执行成本。我们观察支持容量和带宽的细粒度分配的云存储选项的需求，类似于无服务器云服务提供的计算和内存资源的细粒度分配[7]。分解的NVMe闪存可以为这种灵活的云存储选项提供基础。最后，我们展示了对端到端云存储优化的需求，包括应用程序框架、操作系统和云服务，由于存储堆栈效率低下，几种存储配置没有发挥它们的潜力。

#### 动机与背景

我们讨论用于选择云存储配置的当前方法，并解释了所涉及的挑战。

##### 2.1当前方法

常规方法：用于数据分析作业的输入/输出传统上存在分布式文件系统，例如HDFS或者亚马逊的S3对象存储系统[62.6]。中间数据通常从每个节点上的专用本地块存储卷（例如l-ssd或者l-nvme）并在需要额外容量情况下溢出到r-HDD。在典型的Spark即服务云部署中，每个实例默认提供两个远程存储卷：一个用户实例根卷，一个用于日志[19]。

存在工具：最近的工作聚焦在在云中自动地选择一个最优的虚拟机配置[71,69,3]。然而这些工具往往忽略云存储选项的异构性。在下一节中，我们将讨论云存储配置空间的范围。

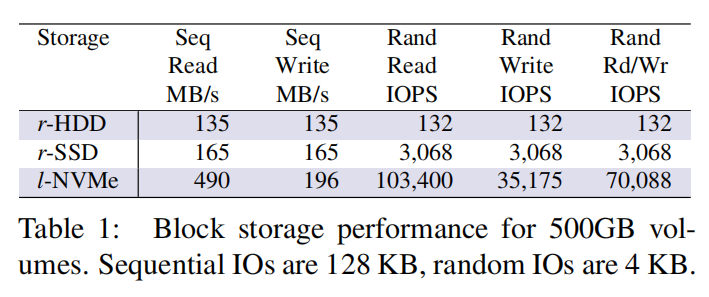
##### 2.2挑战

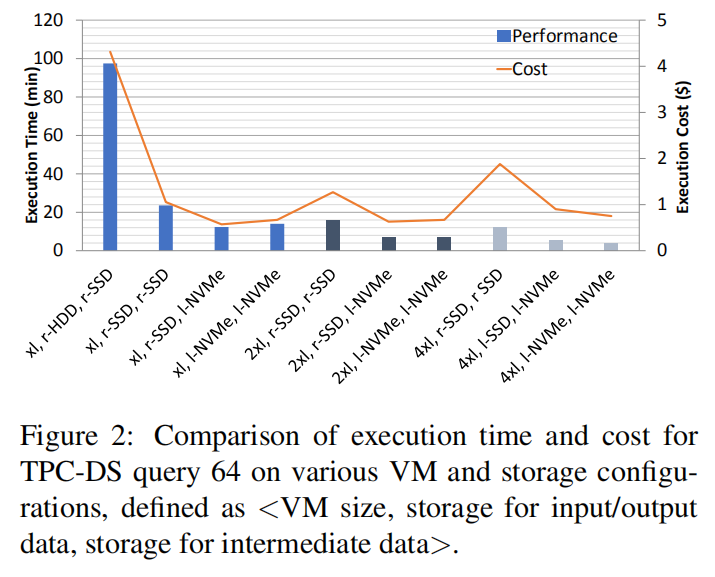
复杂配置空间：云存储有多种形式：对象存储（例如亚马孙S3[6]），文件存储（例如Azure Files[45]）和块存储（例如谷歌计算引擎持久化磁盘[29]）。块和对象存储最常用于数据分析。块存储进一步被细分为硬件选项：冷或吞吐量优化的硬盘，SAS SSD,或者Nvme闪存。块存储对实例来说可以是本地的（直接依附）或者远程的（通过网络）。本地块存储是临时的：数据存在当且仅当实例在运行。远程卷会一直存在直到用户明确删除。

表1比较了在AWS中三个可用的块存储选项。每个存储选项提供一个不同性能、成本和灵活性的权衡。例如，l-NVMe存储以较高的单位成本提供高吞吐量和低延迟。当前，云提供商通常以固定单位容量提供Nvme，该容量直接连接到选择的实例中，按每秒或每小时收费。与没有Nvme的实例相比较，对于有475GBNvme闪存的实例，AWS每小时多收费0.023美元。相比之下，S3的费用是基于容量（每月每GB0.023美元）和带宽（每10KGET请求）使用。除了存储配置外，用户还必须从各种各样 的VM类型中进行选择，以确定正确的CPU和内存数量，VM数量极其网络带宽。这些选择往往影响存储，必须一起考虑。例如，在拥有1GB/S网络带宽的实例中，网络限制了表1中的r-HDD和r-SSD存储卷可实现的序列吞吐量。

性能-成本目标：虽然拥有最多CPU核心、最多内存和最快存储的配置通常提供最高性能，但是其优化运行时间成本也要困难得多。设计用于优化特定目标（例如预测是性能最大化或者成本最小化的配置）的系统并不足以对复杂目标（例如预测在一定预算内使运行时间最小化的配置）提出建议。通过在候选配置上预测应用程序执行时间，我们的方法仍然通用。除非另有说明，否则我们把成本成为执行应用程序的成本。

异构应用数据：我们将分布式数据分析框架管理的数据主要分为两类：通常长期存储的输入输出数据和存活在作业执行期间的中间数据。





中间数据的样例包括在mapper和reduce之间交换的混洗数据、广播变量和从内存溢出的缓存数据集分区。这些数据流通常有不同的访问频率、数据生存时间、访问方式（随机访问、有序访问）和I/O大小。例如，输入/输出数据通常是长期存在的并且是顺序访问的，相反中间数据是短期存在的和大多数访问是随机的。

复杂的存储决策：为作业选择正确的配置可以显著地减少执行时间和成本，如图2所示，该图比较了在8个节点的集群中各种VM存储配置上的Spark SQL查询（TPC-DS query 64）。我们考虑在EC2(X1,2X1,4X1)中3种VM实例大小以及用于输入/输出数据和中间数据的存储选项。最低性能配置的执行时间是最好性能配置的24倍。在r-SSD存储输入/输出数据和在l-nvme存储中间数据（最低成本配置）的成本比在r-HDD存储输入/输出数据和在r-SSD存储中间数据低7.5倍。

#### selecta 设计

##### 3.1概述

Selecta是一个可以自动预测目标应用在一系列候选配置集合中的性能的工具。如图3所示，selecta使用以下作为输入：i）一组应用程序在几种配置上的执行时间；ii）目标应用在两个参考配置上的执行时间；iii）目标应用的性能-成本目标。配置由节点（VM实例）数量，CPU核心和每个节点内存以及用于输入/输出数据和中间数据的存储类型和容量定义。Selecta使用潜因子协同过滤（参阅3.2节）来预测目标应用在其余配置（非参考）上的性能。通过这些性能预测以及各种VM实例和存储选项的单位时间成本，selecta可以为用户的性能-成本目标推荐正确的配置。例如，selecta可以推荐最小化执行时间，最小化成本或者在一定预算内最小化执行时间的配置。

随着新应用的不断启动，这些性能测量会成为selecta不断增长的训练数据集的一部分并且精确度得到提高（见4.4节）。目标应用在selecta推荐的配置上运行结束之后，我们还会反馈性能测量结果-这有助于降低测量噪声和提高精度。由于selecta要花费约1分钟的时间来生成新的预测集合（确切的时间取决于训练矩阵大小），用户可以在使用新的数据集重新启动目标应用的时候来重新启动selecta去获取更精确的推荐。在我们的实验，对每个目标应用的推荐结果在两次反馈迭代后收敛。训练集不断增长的能力还为selecta提供了一种扩展其考虑的配置集的机制。最初，selecta评估的配置空间是原始训练集中的一组配置。当新的配置可用时，selecta会接受在此配置上的应用的分析数据，该工具将会开始预测此配置上所有的应用程序的性能。

##### 3.2性能预测

预测方法：selecta使用协同过滤来预测目标应用在候选配置上的性能。我们选择协同过滤，因为它与所使用的数据分析框架无关（例如spark、storm），它允许我们利用跨应用和配置收集的稀疏训练数据[56]。然而例如CherryPick[3]和Ernest[69]等系统仅仅基于训练数据为目标应用建立预测模型。Selecta的目标是利用来自多个应用可用的训练数据，通过仅对运行两次目标应用的概要分析来收敛得到准确的建议。我们讨论协同过滤可替代的方法来解释我们的选择。基于内容的方法（例如线性回归、随机森林和神经网络模型）根据应用特征（例如混洗数据和读写数据）和配置特征（带宽或者每个虚拟机的核心数量）建立模型。我们发现除非在模型中使用输入特征（例如目标应用在目标配置上的平均CPU利用率），否则基于内容的预测器将没有足够的信息来了解应用的的计算和IO需求并降低准确性。要求在所有候选配置上运行目标应用来收集特征数据的方法是不切实际的。

另一个可替代的方法是基于分析框架的结构来建立性能预测模型，例如spark中的map，shuffle，reduce阶段的具体信息[36,75]。这导致特定于框架的模型和随着框架实现结构的演变可能要求重新调整甚至重新建模（例如随着序列化操作的CPU顺序提高）。

潜因子协同过滤：selecta的协同过滤模型把应用和配置转换为潜在因子空间[10]。这个空间以潜在特征（例如隐藏）描述了应用和配置。从训练应用的性能测量中自动推断这些特征[56]。我们对潜在因子模型使用称为奇异值分解（SVD）的矩阵分解技术。SVD把输入矩阵P分解成三个矩阵U,λ,V的乘积，其中行代表应用，列代表配置。P矩阵中每个元素Pij代表应用i在配置j上的标准化性能。潜在特征由对角矩阵λ中的奇异值表示，并按下降幅度排序。矩阵U表示矩阵P的行与λ中的潜在特征的相关强度。矩阵V表示矩阵P的列和λ的潜在特征的相关强度。尽管模型没有告诉我们潜在特征物理上表示什么，但是一个潜在特征的假设例子是随机I/O吞吐量。例如，selacta可以推断出应用程序性能在多大程度上取决于随机I/O吞吐量和配置可以提供多少随机I/O吞吐量。运行SVD的一个挑战是矩阵P是稀疏的，因为我们仅在特定配置下对应用进行了性能测量。由于执行SVD矩阵分解需要一个完全填充的输入矩阵P，我们首先通过随机初始化缺失条目开始，然后运行随机梯度下降来更新未知的条目（使用目标函数最小化矩阵已知项的均方误差）【13】。直觉是以最小化已知条目的误差的方式来迭代分解和更新矩阵P，这个技术还使用准确的预测来更新未知条目。Selecta对SVD使用Python sci-kit Surprise库【33】。

##### 3.3使用selecta

**新的目标应用**：第一次将应用呈现给selecta时，将其配置在两个参考配置上，这些参考配置最好在计算和存储资源属性上差距甚远。Selecta要求在所有应用中参考配置必须保持固定，因为在运行SVD之前性能测量倍标准化为一个参考配置。分析应用程序性能涉及运行应用至完成并记录执行时间和一段时间内的CPU平均利用率（包括IO等待时间）。

**定义性能-成本目标**：预测所有配置的应用程序性能之后，selecta基于用户定义的排序函数推荐配置。例如，为了最小化运行时间成本，排序函数是min（运行时间\*成本/每小时）。当选择存储技术时（例如SSD vs。Nvme Flash），selecta还必须考虑应用程序的存储容量要求。Selecta利用Spark监视日志中可用的分析运行的统计信息来确定中间（混洗）数据和输入输出容量【63】。

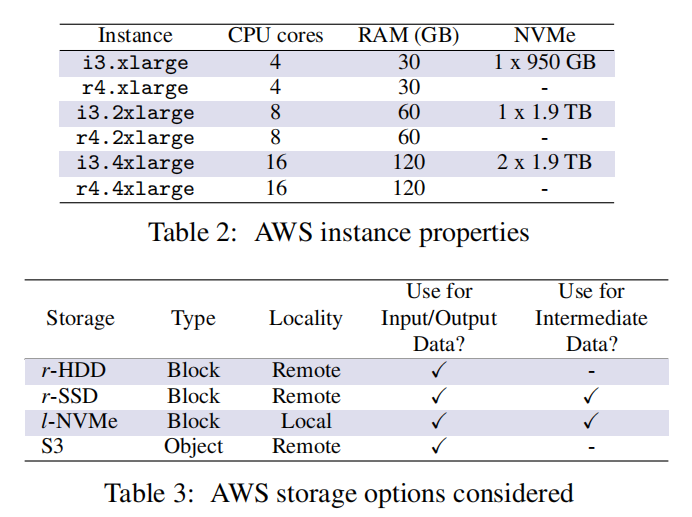
**适应变化**：重复作业极其输入数据集可能会发生变化。为了检测可能会影响最佳配置选择的应用特征的变化，selecta可能依赖于来自初始应用程序分析和后续执行回合的CPU利用率信息。当应用首次呈现给系统，selecta分配唯一ID去存储应用的特定信息（例如IO等待，CPU利用率）。每当应用重新执行的，selecta会将现在的IO等待时间与存储的配置相比较。根据IO等待时间的差异，selecta将基于可用的度量计算精确的预测或者将工作负载视为新的应用程序，开始新的性能分析运行。

**处理云中的噪音**:在公共云平台上，推荐最优配置的另一个挑战是噪音。噪音是由其他租户的干扰、硬件异构或者其他因素产生的【59】。为了解决噪声问题，selecta依赖于性能和CPU利用率的反馈。刚开始，只有少数分析程序运行，selecta的性能预测受噪音影响。当更多的度量被输进系统，selecta平均性能和CPU利用率，使用存储采样来避免异常值的高倾斜。Seelcta对在应用-配置矩阵中每一条项目都保留一定可配置数量的样本点（eg.three），以检测上述描述应用的变化。如果一个特定的应用被噪音严重影响，例如计算和I/O瓶颈和之前运行显著不同，selecta的检测应用变化机制会识别异常值。

#### 4 selecta 评价

Selecta的协同过滤方法对应用和配置的选择是不可知的。我们评估Selecta在一部分云配置空间上的数据分析工作量，目的是了解如何为数据分析配置云存储。

##### 4.1一套方法



**云配置：**我们在亚马逊EC2上部署selecta并且考虑表2和表3所展示的实例和存储选项的配置。在可能的VM和存储组合中，我们考虑17种候选配置。我们调整空间来保持研究预算不变，并且专注于最有可能发现关于分析型云存储配置的有趣见解的实验。我们选择EC2实例家族，这些实例支持数据库和流行的spark即服务提供商【18】。I3是目前唯一一个可用NVMe闪存的实例家族，r4允许对存储选项进行公平的比较，因为它们具有相同的内存和计算比率。我们仅考虑中间数据存储IOPS等于或大于输入输出数据存储IOPS的配置，因为中间数据有更多的随机读取。因为我们发现大多数应用都是r-HDD的IO绑定，所以我们仅考虑具有最少数量核心的实例大小的r-HDD。我们把分析工作限制在r-HDD上，因为我们的应用数据集达到1TB，然而在AWS上具有l-HDD的实例最少具有6TB的存储大小，这将不利于容量的有效利用。我们没有考虑本地的 SAS/SATA SSDs，因为对于大多数spark工作负载 ，它们的存储容量与CPU的核心比率太低。我们使用弹性块存储(EBS)作为远程块存储【5】。

我们使用9个节点的集群用于我们的评价。这个集群由1个Master节点和8个executor节点组成。Master节点运行spark驱动和Yarn资源管理。除非输入输出数据存储在S3，我们还在master上运行一个HDFS namenode。我们根据spark调整稚嫩配置框架相关参数，例如JVM头大小，executors数量，并将执行任务的数量与VM的CPU内核进行匹配【15，14】。

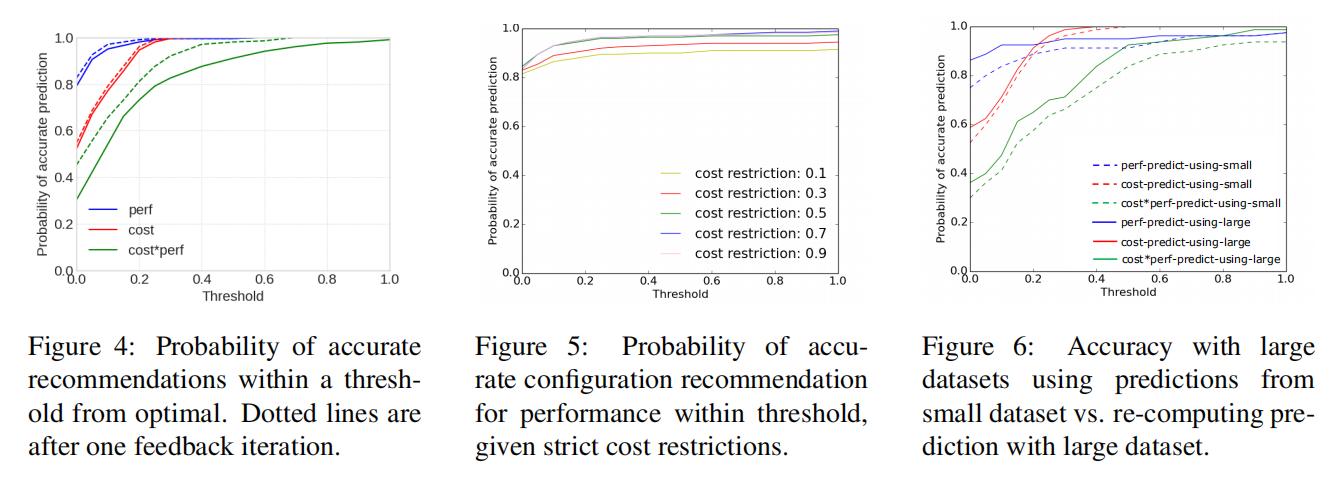
应用：与以前的研究工作相似【50，68，3】，我们认为spark是一个代表性的数据分析框架。我们使用spark v2.1.0与hadoop v 2.7.3的HDFS。我们使用超过一百个SPARK SQL和ML应用来评价selecta，每个应用具有两种不同的数据集规模，总共204个工作负载。我们的应用程序集包括92个比例因子为300和1000GB的TPC-DS 基准的查询【67】。我们对来自TPC-BB基准的Spark SQL和ML查询使用相同的比例因子，该基准在零售行业领域具有结构化、非结构化和半结构化数据模型【27】。由于大多数bigbench查询是CPU绑定的，我们专注于具有大量IO请求的查询，查询 3，8，14，16, 21, 26, 28, 29。我们也运行100和400GB的分类任务【52】。最后，我们在两个分别具有16M和32M行和4KB条目的表上运行SQL同等连接查询。对于所有的输入和输出文件，我们使用未压缩的Parquet数据形式【26】。

**实验方法:**为了获取每一个应用最真实的性能和最优配置选择，我们在所有候选配置上运行每一个程序。为了解决云中的噪音，我们每一个实验运行三次（每一个配置上的每一个应用）并且在评估中使用每次运行的平均值。两次运行是连续的，一次运行是在一天的不同时间。我们还通过使用一次运行的数据作为selecta的输入，每次运行的平均性能作为基本事实来验证结论。为了训练和测试selecta，我们使用留一法交叉验证【58】，这意味着一次将一个工作负载用作目标应用程序，而其余工作负载用于训练。我们假设应用程序已经在所有候选配置上被分析，除了第4.4节中的敏感性分析除外，我们在4.4节中调查了训练矩阵密度要求以进行准确的预测。

指标：我们使用两个指标来衡量selecta的预测质量。首先我们报告相对均方根误差(RMSE)，一个推荐系统中常见的指标。第二个以及更多关于selecta的相关指标是得到一个**准确配置建议的可能性**。我们认为建议是准确的，如果配置在应用程序的真正最佳配置的阈值T内满足用户目标。例如对T=10%的最小化成本目标，**准确预测的可能性**是selecta建议的百分比（在所有测试应用程序中），这些建议的实际成本在真正的最佳成本配置的10%内。使用阈值对噪声的鲁棒性更强和允许我们做出更多有意义的关于selecta准确性的结论，因为第二好的配置可能性能与最好的配置相似或者更差。我们的性能指标是执行时间，成本是美元。

##### 4.2预测精度

我们提供一个204行的矩阵作为seelcta的输入，其中每行（应用）被指定为目标应用在每轮测试中。我们运行selecta204次，每次把不同的应用作为目标。现在，我们假设矩阵中所有剩余训练数据行是密集的，这意味着用户已经在所有候选配置上分析应用程序。单个目标应用行是稀疏的，仅包含两个条目，每个配置在参考配置上运行。



Selecta预测所有应用程序的平均性能的相对RMSE为36%。为了理解selecta的性能预测是怎样转变成建议的，我们在图4画出了性能、成本、性能-成本目标精度图。该图显示了接近最佳建议的概率与阈值T的关系，该阈值定义了最佳阈值的百分比被认为足够接近。当查找最优性能配置，selecta有94%的概率推荐一种配置，该配置与最优配置误差在10%以内。对于最小成本目标，seelcta有80%的概率推荐一种配置，该配置与最优配置误差在10%以内。预测性能-成本目标更具有挑战性，在selecta相对运行时间中，对一个应用在所有候选配置的预测的误差是平方的：性能-成本=（执行时间）平方\*每小时配置成本。

图4中的虚线展示了经过一轮反馈之后精度是怎样提高的。这里，我们假设目标应用程序在反馈回合中拥有相同的数据集。这为目标应用程序行提供了额外的训练输入（如果建议配置不是参考配置，则为新项目；如果建议的配置为参考配置，则为新样本以求平均现有数据）。对于性能-成本目标，近最优的推荐可能性显著增加，反馈之后从52%增加到65%，阈值T=10%。图5显示了对目标（形式为：给定一个固定成本限制C，选择最好性能配置）给出准确建议的可能性。为此目的，我们认为selecta的建议是准确的，如果它的成本少于或等于预算，它的性能在目标真是最佳配置阈值之内。在图5中，假设T=10%，对于成本限制，selecta取得83%~94%的准确率。长尾是由于性能预测误差，这些误差导致selecta低估了一小部分配置的执行成本（例如，selecta推荐一个实际上超过预算的配置）。

图7，我们将selecta精度与四条基线作比较。第一条基线是一个随机森林预测其，与PARIS使用的方法相似【71】。我们使用以下特征：CPU核心数，配置提供的磁盘IOPS和磁盘传输，应用存储中间数据和输入输出数据的容量，CPU利用率、性能和在两个参考配置中运行应用程序时测量的磁盘总的吞吐量。虽然随机森林预测其比selecta利用了更多的特征，但它精确度更低。协调过滤更适合训练数据的稀疏性质。我们发现在随机森林模型中最重要的特征都与IO相关（例如，当在参考配置运行应用的时候测量的IO吞吐量，用于中间数据数据的存储所支持的读写IOPS），这些特征强调了选择正确存储的重要性。

图7中的第二条基线使用默认的推荐配置，该配置记录在Databricks的工程博客文章中：用于中间数据的l-Nvme和用于输入输出数据的S3【19，21，20】。最大单位时间成本基线使用一种简单的试探法，即总是选择单位时间最贵实例。最小单位时间成本基线选择单位时间最便宜的实例。Seelcta优于这些所以启发式策略，确认了需要工具自动进行配置选择。

##### 4.3演变的数据集

我们使用4.1节描述的大小数据集研究数据集大小对应用性能和selecat预测的影响。我们使用102个工作负载和小规模数据集来训练selecta，然后对同样的工作负载使用大规模数据集来评估selecta的预测精度。仅根据分析具有较小数据集的应用程序为具有较大数据集推荐配置时，图6虚线绘制了seelcta的精确度。实线表示的是当selecta重新分析具有大规模数据集的应用程序来进行预测的精度。对于大约8%的应用，使用小规模数据集进行分析不足以表明使用大规模数据集的性能。

我们发现在某些场合，使用小数据集的性能不能表示使用大数据的性能，应用的计算和I/O密集度的关系受数据集大小影响。正如3.3节描述，seelcta通过比较小规模和大规模数据集运行时的CPU利用率统计信息来检测这些情况。图8展示了一个工作负载案例，对于该工作负载，较小的数据集性能不能表示较大的数据集性能。我们使用因特尔性能分析工具记录和画出CPU利用率【34】。当运行期间的平均iowait百分比在参考配置的大型和小型配置运行之间发生显着变化时，通常在参考配置上对应用程序进行分析，并将其视为新应用。

##### 4.4敏感性分析

我们执行敏感性分析，以确定输入矩阵密度要求，以进行准确的预测。我们观察矩阵的行密度（分析训练应用程序的候选配置的百分比）和列密度（训练应用程序数量）。我们还讨论对参考配置选择的敏感性。

图9a展示了selecta的性能、成本和性能-成本目标如何随输入矩阵密度的函数变化而变化。假设随着时间的推移，系统已经积累了203个训练应用。我们发现，评价每个目标应用程序，行仅需要大约20%~30%密度就可以使selecta实现足够的精度。这意味着在稳定状态下，用户可以在20%~30%的候选配置上分析训练应用（包括参考配置）。分析额外的配置会减少收益。

接下来，我们考虑冷启动这种情况，在这种情况下，用户希望通过在所有候选配置上分析限量训练应用程序来快速启动系统。图9b显示了达到预期精度所要求的训练应用数量。这里，对于每一轮目标应用测试，我们采用我们拥有的203个训练应用程序，并随机删除了一部分行（训练应用）。**我们确保删除与目标应用程序的不同数据集比例因子运行相对应的，**以确保selecta的精度不取决于与目标应用程序相关的训练应用。由于达到预期精度所需的训练应用数量用户希望探索的配置空间的大小，因此图9b中的x轴表示训练应用程序的数量与候选配置的数量之比R。我们发现要从冷启动中使用密集的训练数据来快速启动selecta，为了达到预期精度，用户应该提供比候选配置数量多2.5倍的训练应用程序。**在我们的例子中，使用超过43倍的在17种配置上分析的训练应用来快速启动selecta达到收益递减目的。**

最后，我们研究冷启动是否需要在所有配置中分析训练应用。我们使用R=2.5，17种候选配置对应于使用43个训练应用程序。图9绘制了当我们改变候选配置（应用程序在这些配置上被分析）的百分比的精确性（包括参考配置，我们假设该配置总是被配置）。对于冷启动，该图表明，这对用户来说是足够的在40%~60%的候选配置上分析训练应用。当selecta继续运行和累计更多应用程序，用户需要为应用程序配置的配置的百分比下降到20%~30%（这是图9a的稳定状态结果）。

我们使用不同的参考配置对selecta进行实验。我们发现精度对参考配置的选择不是很敏感。我们看到使用不同的VM和存储类型的参考会有一点好处。尽管一个参考配置必须在所有应用程序运行种保持不变，因为它用于标准化性能，我们发现用于第二次分析运行的参考配置可以改变，因为它对seelcta的精度没有显著影响。

#### 5云存储结论

我们对数据分析的云存储配置的分析揭示了一些关于云存储配置的几个结论。我们讨论关键要点及其对存储系统未来的研究意义。

对数据分析来说，Nvme存储具有高性能和成本效益。我们发现Nvme不仅易于提供高性能，更令人惊讶的是，成本也很低。尽管Nvme是每GB/小时最贵的存储类型，它的高带宽允许应用允许更快，减少整体工作执行成本。

平均每个应用程序，我们发现，与r-SSD相比，l-Nvme减少应用作业总时间大约27%，与r-HDD相比，大约75%。尽管，我们在评估种没有考虑l-SSD和l-HDD，我们验证得知本地和远程存取HDD、SDD实现类似的性能，因为我们实例有足够的带宽（达到10GB/S）以及现代网络额外增加很少开销，除了HDD和SDD存取延迟【8】。相反，Ousterhout等人先前对spark应用程序的研究，结论是，优化或者消除磁盘访问仅仅最多能减少作业完成时间的19%【50】。我们认为存储对端到端应用性能影响增加的主要原因是Spark版本更新（我们在研究中使用spark 2.0到2.1）。Spark已经和许多目标为CPU效率的优化发展，例如可识别缓存的计算，用于表达式评估的代码生成和序列化【17】。接下来进行的工作主要是优化CPU周期（用于数据分析计算），例如优化I/O处理路径【66】，我们希望存储选择更重要。

灵活的容量和带宽分配需求：调配存储涉及选择正确的容量、带宽和延迟。Selecta使用来自Spark Log的统计信息来决定容量要求和应用协同过滤来探索性能-成本目标。然而，云提供商施加的众多的限制限制了所选存储配置的成本效益。例如，对于远程块存储卷，云提供商强加最小容量限制（AWS上的r-HDD至少500GB）以及决定卷中的数据是怎样与物理设备映射的，这直接影响了存储吞吐量（HDD数量与主轴数成正比）。对本地存储有更重要的限制，例如l-Nvme，仅在连接到指定实例类型的固定容量中可用。云提供商施加的计算、内存与存储资源之间的比率没有为许多我们研究的应用提供合理的资源平衡。例如在两张64GB的表上的SQL等价查询会使i3.x1实例上500GBNvme设备的IOPS饱和，但是有一般的容量未使用。进一步来说，本地存储是短暂的，这意味着必须保留实例来保留本地设备的数据。 因此，尽管我们展示了任务期间输入输出和中间数据存储在l-Nvme是具有成本效益的，但是与使用远程存储卷或者例如S3的对象存储系统相比，在l-nvme上长期存储输入输出文件将会显著增加成本。

我们主张在云中使用快速灵活的存储选项。云计算的新兴趋势，例如AWS Lambda，Google Cloud Functions和Azure Functions等无服务器计算产品，提供对计算和内存资源按使用量的细粒度访问【31，7，28，46】。当前，没有选项允许对具有低延迟和高带宽特性的云存储的细粒度的容量和带宽分配【41】。尽管S3提供了高可扩展、高可用和相对高带宽的按使用量存储。我们认为数据分析应用可以从更高的吞吐量受益（例如，Nvme闪存）。S3还导致高延迟，我们认为这是短期运行SQL查询（仅读取几兆数据）的主要瓶颈。

分解的Nvme是快速灵活云存储有希望的选择：通过实现在网络上高效访问资源来分解Nvme闪存，对快速灵活的云存储来说是很有前途的选择。最近硬件辅助【49，44】和软件专用【40】技术的发展实现在各种网络选项上低延迟开销访问远程Nvme设备，包括使用TCP/IP协议的商用以太网。这些技术使我们能够建立分类的闪存存储，该存储为分析型工作负载提供细粒度的容量和IOPS分配以及独立的存储和计算资源扩展。应用程序根据需要从大量可远程访问的Nvme设备中分配容量和带宽。在此设置下，selecta可以帮助分析型工作负载中的每一个数据流预测正确的容量和吞吐量要求，以此指导从分解的Nvme闪存中分配资源。

实现基于分解闪存的灵活云存储存在几个挑战。首先，网络要求非常高。当前，AWS上的设备可以实现500MB/S至4GB/S顺序读取带宽，取决于容量。写吞吐量和随机访问带宽也很高。云系统网络设施必须能够支持大量可以远程访问Nvme的实例，并且用于爆发出存储设备最大吞吐量的能力。分享远程存储设备的另一个挑战是来自不同租户读和写的请求干扰【40，61】。我们观察几种情况，分别在r-SSD和l-NVME上分离输入/输出数据和中间数据，相应地使得性能更好（成本更低）比所有数据存储在l-Nvme上。这发生在大量输入数据读取与大量混洗写入重叠的作业中，例如，图1展示的TPC-DS 80查询。一个分解存储系统必须使用他们的调度方法【40，47，61，51，60】或者设备级隔离机制【12，54，38】解决干扰问题。最后，在用于显示分类的分解闪存（例如，块存储，键值存储，分布式文件系统，或者其他）的接口中，有一些有趣的折衷的选择。

端到端的最优化需求：在我们的实验中，尽管它长期存储输入/输出数据具有成本效益和有能力匹配SSD提供的顺序带宽，但远程HDD性能表现很差。使用Linux上的blktrace tool 【37】来分析块设备层的IO请求，我们发现尽管每个spark任务顺序读取/写入输入/输出数据，但是来自运行在不同核心的多个任务的流在块设备层交错。因此，一个远程HDD卷看到的访问流由大约60%的随机IO操作组成，与完全顺序IO相比，性能大大降低。这使得随机访问（例如使用多种HDDs设备或者闪存）的高吞吐量的解决方案更适合实现数据分析的高性能。增加随机IO性能会导致更高的每单位时间成本。除了构建更高的存储系统，当这些访问在应用级别可用时，我们试图优化整个顺序访问堆栈。当然，总有一些具有固有随机访问模式的工作负载，这些访问模式不会从这种优化中受益。

#### 6 结论

我们的工作专注于根据性能和成本选择存储配置。其他重要的考虑因素包括持久性，可用性和一致性，特别是对长期的输入/输出数据存储【42】。开发者可能更喜欢一个特定的存储API（例如，POSIX的文件和对象接口）。用户可以使用这些定性约束来限制selecta认为的存储空间。用户还可以选择不同的存储系统来进行高性能处理，而不是长期存储重要数据。

我们的研究表明，分离输入/输出数据可以发现更丰富的配置空间，以及根据应用程序要求提供更好的存储资源配置。我们可以进一步把中间数据划分为更细粒度的流，例如混洗数据，广播数据和从内存溢出的缓存RDD。了解这些细粒度流的特性以及如何将它们映射到云中的存储选项可能会揭示更多的好处。

压缩方案在处理、网络和存储之间提供了有趣的折衷方案。