当代设备上的基于特征的 SpMV 性能分析

Panagiotis Mpakos*、Dimitrios Galanopoulos*、Petros Anastasiadis*、
Nikela Papadopoulou†、Nectarios Koziris*、Georgios Goumas*

*雅典国立技术大学计算系统实验室，希腊
†查尔姆斯理工大学，G¨oteborg，瑞典
*{pmpakos,dgal,panastas,nkoziris,goumas}@cslab.ece.ntua.gr †nikela@chalmers.se

摘要——SpMV 内核的特点是：针对不同输入矩阵和计算平台，其性能存在显著差异。虽然 GPU 曾被认为是 SpMV 的最先进平台，但随着先进多核 CPU 和低功耗现场可编程门阵列加速器的出现，我们需要重新审视其性能和能效。本文基于与内存带宽强度低、指令级并行度低、负载不均衡和内存延迟开销等常见瓶颈相关的矩阵结构特征，对 SpMV 性能进行了高层次分析。为此，我们构建了一个涵盖这些特征的广泛人工矩阵数据集，并研究了九个现代高性能计算平台上不同存储格式的性能，包括五个 CPU、三个 GPU 和一个现场可编程门阵列。利用真实世界矩阵验证所提出的方法后，我们分析了大量实验结果，并得出了关于不同目标架构在 SpMV 方面的竞争力，以及各特征/瓶颈对其性能影响的关键见解。

索引词——稀疏矩阵-向量乘法、性能分析

I. 引言

稀疏矩阵-向量乘法（SpMV）是一种关键的计算内核，广泛应用于科学计算、图处理、机器学习等领域。SpMV 是大型稀疏系统求解器的核心，实际上主导了这些求解器的执行时间。据多次报道，其性能仅达到系统峰值性能的一小部分，且在不同矩阵之间差异很大。内存带宽强度不足、指令级并行度降低、负载不均衡以及内存延迟开销等多种问题共同导致了这种不理想的性能表现 [1], [2]。影响该内核性能的确切瓶颈受矩阵属性影响，并且可能因目标架构而异。

传统上，研究工作主要集中于分析稀疏矩阵-向量乘法的性能，并提出有效的格式选择方法 [2] ，这些方法基于输入矩阵 [3]–[11], 、目标平台 [12]–[15], 或两者兼顾 [2], [16], [17]。过去几年，凭借高计算吞吐量和内存带宽，GPU 一直占据主导地位，成为加速稀疏矩阵-向量乘法的先进目标平台。尽管稀疏矩阵-向量乘法会产生间接内存访问，可能导致控制流 fi 分歧，但基于 GPU 加速的稀疏矩阵-向量乘法已得到广泛研究和优化 [18]–[22]。然而，随着 AMD EPYC 和 ARM NEON 等新型 CPU 的出现，以及核心数增加和缓存大小不断增大，CPU 再次成为颇具吸引力的选择。此外，需求对能源效率型高性能计算的需求促使人们在评估方法的有效性时考虑能耗相关指标。因此，诸如现场可编程门阵列等新兴架构因其能够提供更高的每瓦性能，如今被视为适用于大规模系统的可行替代方案。

本文旨在研究当代设备上 SpMV 在性能和能效方面的执行行为，并考察已确定的四种潜在瓶颈，即内存带宽强度、指令级并行度降低、负载不均衡和内存延迟开销。为了了解这些瓶颈的影响，我们重点关注稀疏矩阵的四个主要特征，这些特征均与至少一个瓶颈直接相关（第 III-A 节）。具体而言，我们考察与内存带宽强度相关的矩阵内存占用、与低指令级并行度相关的平均行长度、与负载不均衡相关的偏度系数，以及通过跨行相似度和行间平均邻居数描述、与内存延迟开销相关的不规则性。

本工作的主要贡献如下：

- 我们确定了一组最小矩阵特征，能够充分捕捉稀疏矩阵-向量乘法的主要性能瓶颈。（第III-A节）
- 我们实现并公开发布了[1] 一个稀疏矩阵生成器（第 III-B节）。该生成器使用所考虑的特征集来生成一个人工矩阵数据集，其规模远大于其他常用的真实世界矩阵测试套件，且涵盖的矩阵特征空间也广泛得多。
- 我们在九种目标设备上，使用多种“当前实践”和“研究型”存储格式及实现[2] 收集实验结果（第IV节），并验证了尽管使用的特征集很精简，人工矩阵在所有设备上的性能仍与具有相似特征的真实矩阵相近（第V-A节）。
- 通过分析大量评估结果，我们 fi现，尽管 GPU 仍保持性能优势，但对于中等规模的矩阵，CPU 已重返竞争行列（第 V-B 节）。现场可编程门阵列是能效最fi高的设备

[1]生成器可在 https://github.com/DimitrisGalanopoulos/artificial-matrix-generator[2]基准测试可在 https://github.com/p-anastas/SpMV-Research

用于稀疏矩阵-向量乘法，但目前其性能上限显著较低。

- 在以特征为中心的分析（第 V-C 节）中，我们发现内存占用是影响最大的特征，其次是每行非零元素的平均数量。所有设备上的大多数格式都能处理负载不均衡，而不规则性则可能损害图形处理器的性能。
- 我们分析了“研究型”格式（第 V-D 节）的性能，并观察到，尽管“实际应用主流”格式的整体性能更好，“研究型”格式在棘手场景中仍具有竞争优势，例如规模较大、负载不均衡且不规则的矩阵。
- 我们考察了人工构造的矩阵数据集规模对性能结果质量的影响（第 V-E 节），并展示了人工生成的数据集如何推动数据分析，从而更好地理解稀疏矩阵-向量乘法的性能瓶颈（第 V-F 节）。

II. 背景

本节讨论稀疏矩阵-向量乘法的常见瓶颈，以及稀疏矩阵的存储格式。

A. 稀疏矩阵-向量乘法的常见性能瓶颈

稀疏矩阵-向量乘法以峰值性能占比极低而闻名，即使在最佳情况下，这一比例也可能不超过 20% [13], [21], [23]。尽管多种因素会共同导致性能低下，包括矩阵的稀疏模式、存储格式、算法以及平台的硬件特性，但先前的研究 [2], [21], [24]归纳出稀疏矩阵-向量乘法性能中的以下四个可识别瓶颈，这些瓶颈在不同架构上均显而易见：

1）内存带宽强度：稀疏矩阵-向量乘法是一种运算强度极低的内核；由于其流式特性以及对稀疏存储格式元数据的访问，其 fl每字节运算次数比率低于 1。因此，稀疏矩阵-向量乘法深受处理器/内存性能差距的影响。该瓶颈的严重程度可能因输入矩阵的大小和平台可用的内存带宽而异，但它被认为是该内核的主要瓶颈。

2) 指令级并行度较低：当稀疏矩阵中每行的非零元素较少，且循环遍历操作占执行时间的比例显著增加时，指令级并行度会成为另一个瓶颈。在这种情况下，由于向量和索引元数据（例如行指针）的占比更高，内核的运算强度会进一步降低。同时，向量化和行级并行的机会也会受到限制。

3）负载不均衡：对于每行非零元素数量方差较大，或稀疏模式各区域差异明显的矩阵，会出现这一瓶颈。根据工作分配策略的不同，稀疏矩阵-向量乘法可能会遭遇严重的负载不均衡和资源利用不足。这一瓶颈的严重程度取决于输入矩阵的稀疏模式、存储格式、算法、工作分配方式，以及硬件平台上可用的并行度。

4）内存延迟开销：由于矩阵的稀疏模式决定了对 x向量的不规则访问模式，这一瓶颈会严重影响稀疏矩阵 - 向量乘法的实现。这些访问可能导致 CPU 上出现大量缓存未命中、图形处理器上的非对齐内存访问，以及现场可编程门阵列上的显著开销（如果 x向量 未缓存在速度最快的内存中），进而造成性能损失。这些开销的影响取决于矩阵的稀疏模式。

B. 稀疏矩阵存储格式与实现

稀疏矩阵-向量乘法中的矩阵采用稀疏存储格式，这会显著影响其性能。多年来，人们实现了各种格式，以应对特定的架构、算法或稀疏性相关瓶颈。关于如何选择最佳存储格式的丰富研究成果 [3]–[11] 表明，没有一种存储格式是完美的。在本综述中，我们考虑两类格式：厂商库采用的“业界实践”格式，以及学术研究提出的一些“研究型”格式。下面我们简要介绍所用各类格式的优势与不足；有关业界实践格式的更多详情和伪代码，请读者参阅 Filippone 等人 [21]。我们还指出，由于时间和篇幅限制以及代码不可用，无法穷尽纳入所有研究型格式。此外，这也超出了本文的范围；本文侧重于评估先进存储与优化技术为稀疏矩阵-向量乘法带来的性能提升，而非全面比较各种存储格式。

1) 坐标格式（COO）：COO 是一种直观的存储格式；它使用三个长度为 nnz 的数组存储稀疏矩阵，分别记录每个非零元素的行（行索引 RowIdx）、列（ColIdx）和值（Value）。COO 在工作线程之间均衡负载方面表现良好，但冗余的元数据信息会增加内存带宽需求。

2) 压缩稀疏行格式（CSR）：CSR 是应用最广泛的格式，它将行索引 RowIdx 压缩为行指针 RowPtr 数组，数组大小为 $m+1$，其中包含每一行首个元素的偏移量。它显著减少了矩阵的内存占用，并且根据矩阵结构，可以在稀疏矩阵-向量乘法的内存强度与指令级并行（ILP）潜力之间实现一定的平衡。

3) ELLPACK（ELL）和混合格式（HYB）：为更好地利用向量单元，ELL 格式将非零元素的列索引和值存储在稠密的$m \times max(nnzrow)$数组中$Offset$ 和$Evalue$ 分别为这两个数组的名称，并为所有较短的行添加填充。这样可以牺牲内存带宽来提升指令级并行度，使均衡矩阵获得较高性能；但对于每行非零元素数量方差较大的矩阵，由于过度填充，会导致性能严重下降。HYB 格式旨在避免这一问题，并通过与 COO 结合来减少 ELL 的内存占用。HYB 定fi义一个阈值k，该阈值可通过启发式方法确定，也可设为每行非零元素数量的平均值；对于每一行的前fik个非零元素使用 ELL 存储，其余元素则使用 COO 存储。

4) VSL（压缩稀疏列 *[21]* 变体）：VSL [25] 是赛灵思为 Alveo-U280 FPGA 专门设计的格式。它将矩阵划分为二维

分区，再将每个分区划分为 16 个部分，并通过相等数量的 HBM 通道将其馈送至 16 个执行单元；同时使用零填充，以适应双精度累加延迟。当填充过多，导致矩阵的存储需求超过用于存储该矩阵的 HBM 通道的最大容量时，此设计便会失效。

5）研究型格式及其实现：CSR5 [20]ex 通过拆分行来创建大小相近的子问题，从而扩展 CSR 以解决负载不均衡问题。它专为 fit GPU 设计，而 CSR 无法在各线程束/线程之间均衡分配工作。CSR5 的缺点是需要额外的元数据来拆分行，这会略微增加内存占用；然而，当其复杂的输入矩阵拆分并无益处时，也可能导致性能下降。Merge [26] 是 CSR 的另一种轻量级扩展，无需预处理开销。它通过为每个处理单元分配大小相同的工作块来解决负载不均衡问题。SELL-C-σ [27]是 ELL 的一种变体，其中可调参数 C（块大小）和 σ （排序范围）经过选择，以匹配底层硬件能力，同时避免增加内存延迟开销。SparseX [28] 是一个库，可自动检测稀疏矩阵中的稠密、水平、垂直、对角或块子结构。SparseX 直接针对内存带宽强度进行优化，并以最小的内存占用对每个子结构进行编码，从而实现矩阵的高度压缩表示；这在大型矩阵中尤其有利。

III. 方法论

本研究旨在深入了解当代计算设备上稀疏矩阵-向量乘法的性能表现。为此，我们需要收集大量矩阵的评估结果，并确保前述潜在瓶颈确实出现在实验中。因此，我们选择不依赖测试套件 [29], [30], 中已有的真实矩阵，原因是：a) 数量不足；b) 其中许多矩阵彼此相关（例如，同一矩阵的不同版本）；c) 它们并不总能体现性能表现，即便这些表现出现在罕见矩阵中，甚至出现在目前尚不存在的矩阵中。

为了控制实验过程，我们采用生成人工矩阵的方法。我们的方法论中的一个关键步骤是确定矩阵生成过程。我们的方法基于定义一个（或数量非常有限的）矩阵特征，这些特征与第 II-A 节讨论的至少一种性能瓶颈高度相关。下一节将讨论选定的五种特征。掌握矩阵特征后，我们便能定义一个由真实矩阵边界限定的特征空间，再创建一个生成器来遍历搜索空间，并在我们的控制下生成大量矩阵。

A. 矩阵特征

以往研究已发现矩阵特征与稀疏矩阵-向量乘法的性能之间存在相关性fi [2], [6]–[9]。研究中使用了数量相当多的特征来训练合适的预测器

表 I：用于人工生成矩阵的特征

标签	特征	描述	矩阵空间
f1	内存占用	矩阵（压缩稀疏行）大小（MB）	[4-32], [32-512], [512-2048]
f2	平均每行非零元素数	每行平均非零元素数	5、10、20、50、100、500
f3	偏斜系数	偏斜系数	0, 100, 1000, 10000
f4.a	跨行相似度	行相似度	0.05, 0.5, 0.95
f4.b	平均数量 邻居	邻居数量	0.05、0.5、0.95、1.4、1.9

以预测稀疏矩阵-向量乘法的性能。我们从以往研究中汲取灵感，重点关注影响性能的矩阵相关特征fl，但刻意限制特征数量，尤其是每个性能瓶颈只使用一个特征。这样，我们以牺牲准确性为代价，换取更简单的方法、更易于理解的结果，以及更可控的矩阵创建过程。

1）矩阵内存占用作为内存带宽强度的指标：压缩稀疏行格式中稀疏矩阵数据的内存占用（以 MBytes 为单位）对性能有很大影响。小尺寸 fit 位于存储层次结构的较低层级中（例如 CPU 中的缓存、图形处理器中的共享内存等），但也会限制并行松弛度。大尺寸可以抵消更复杂格式的开销，并利用更多工作线程，但也需要更大的内存带宽。

2）每行平均非零元数作为低 ILP 的指标：每行非零元素数显然会影响 ILP，因为对于较短的行，大量 CPU 周期会花费在循环控制指令上。然而，该指标还会对性能产生其他严重影响，因为它会降低运算强度，能够指示矩阵的密度（规则性）、向量化潜力及其均衡性（每行平均非零元数非常大时，不容易形成负载不均衡的矩阵）。

3) 偏斜系数*fi*作为负载不均衡的指标：该指标衡量矩阵各行的非零元素数有多不均衡，定义如下：

$$\text{偏斜} = \frac{\text{每行最大非零元素数} - \text{每行平均非零元素数}}{\text{每行平均非零元素数}}$$

偏斜值为 1 表示最长行的非零元素数是每行平均非零元素数的两倍。通常，偏斜值在 10 左右或更低的矩阵被视为均衡矩阵；不均衡矩阵的偏斜值可达数百甚至数千。

4) 平均邻居数和跨行相似度作为不规则性及内存延迟开销的指标：我们将“邻居”定义为非零元素所在行中，位于其左侧或右侧、列距离不超过预设最大值的所有其他元素。该指标计算列距离为 1 时的平均邻居数（因此数值介于 0 和 2 之间），用于衡量向量 x 上的非零元聚集和空间局部性。

跨行相似度表示相邻行之间相似程度的一种度量。我们定义非零元素的“跨行邻居”为下一行中位于预定最大列距离内的所有元素。我们采用的列距离为 1，这意味着同一列，或相邻的左列和右列。然后，我们将跨行相似度指标计算为：一行中至少拥有一个跨行邻居的元素所占的比例（取值范围为 0 到 1），并在所有行上取平均值。跨行相似度旨在捕捉向量 x 上的时间局部性。

表 II：测试平台特征，以及各测试平台使用的存储格式/库。

测试平台	AMD-EPYC 7402 (AMD-EPYC-24)	AMD-EPYC 7742 (AMD-EPYC-64)	ARM-NEON Ampere-Altra Q80-33	Intel Xeon Gold 5120
核心	24 个核心	64 个核心	80 个核心	14 个核心
内存	128 MB 末级缓存（三级缓存） 256 GB DDR4	256 MB 末级缓存（三级缓存） 256 GB DDR4	80 MB 末级缓存（二级缓存） 512 GB DDR4	19.25 MB 末级缓存（三级缓存） 256 GB DDR4
实测带宽	DDR4 50 GB/s，末级缓存 700 GB/s	DDR4 105 GB/s，末级缓存 878 GB/s	DDR4 102 GB/s，末级缓存 650 GB/s	DDR4 55 GB/s，末级缓存 300 GB/s
编译器	gcc 9.2.0	gcc 9.2.0	armclang 22.0.1	gcc 7.5.0
编译器标志	-O3 -mavx2 -fopenmp	-O3 -mavx2 -fopenmp	-O3 -fopenmp -armpl=lp64,parallel	-O3 -fopenmp
格式	MKL-IE 2018.1, AOCL-Sparse 3.2, 朴素 CSR，向量化CSR，压缩稀疏行5 [20], MergeCSR [26],SparseX [28],SELL-C-σ [27]	MKL-IE 2019年.1, 朴素 CSR, 压缩稀疏行5 [20]	Arm性能库 21.1.0，朴素 CSR, 朴素 CSR，合并CSR [26], SparseX [28],SELL-C-σ [27]	MKL-IE 2018.1，朴素 CSR，压缩稀疏行5 [20], 向量化CSR，MergeCSR [26], SparseX [28],SELL-C-σ [27]

测试平台	IBM POWER9 AC922	英伟达 Tesla-P100（PCIe）	英伟达 Tesla-V100（PCIe）	NVIDIA Tesla-A100（PCIe）	Alveo-U280
核心	16 个核心（2 线程/核心）	3584 个 CUDA 核心 @ 1480 MHz	5120 个 CUDA 核心 @ 1455 MHz	6912 个 CUDA 核心 @ 1412 MHz	130 万个查找表
内存	80 MB 末级缓存（三级缓存） 319 GB DDR4	12 GB CoWoS HBM2 带宽 549 GB/s	32 GB CoWoS HBM2 带宽 900 GB/s	40 GB CoWoS 高带宽内存2 带宽 1555 GB/s	8 GB 高带宽内存2 - 带宽 460GB/s 32 GB DDR4 - 带宽 38GB/s
实测带宽	DDR4 109 GB/s，末级缓存 612 GB/s	HBM2 带宽 464 GB/s	HBM2 带宽 760 GB/s	HBM2 带宽 1350 GB/s	HBM2 带宽 287.5 GB/s
编译器	gcc 8.3.1，IBM XL 16.1.1	gcc 9.2.0，CUDA-9.2 和 CUDA-11	gcc 9.2.0，CUDA-9.2 和 CUDA-11	gcc 9.2.0，CUDA-11	gcc 7.5.0，Vitis 2020.2
编译器标志	-O3 -fopenmp	-O3 -arch=sm_60	-O3 -arch=sm 70	-O3 -arch=sm 80	-O3
格式	朴素 CSR、均衡 CSR, MergeCSR [26],SparseX [28]	cuSPARSE-11 坐标格式与压缩稀疏行, cuSPARSE-9.2 HYB，压缩稀疏行5 [20]	cuSPARSE-11 坐标格式与压缩稀疏行, cuSPARSE-9.2 HYB，压缩稀疏行5 [20]	cuSPARSE-11 坐标格式与压缩稀疏行, MergeCSR [26]	Vitis 稀疏库 2021.1

显然，这两个不规则性特征都相当简单且采用蛮力方法，例如，它们不会研究超出连续行范围的关系。更复杂的方案（例如，栈距离配置文件）或许能更好地捕捉内存访问效应，但对于我们希望获得的洞见而言，也会引入不必要的复杂性。我们最初尝试为不规则性定义单一特征时，重点关注行带宽（或对其进行适当划分），但这种方法无法捕捉足够准确的行为。因此，我们转而采用用于衡量空间局部性和时间局部性的两个指标。如第III-B节所述，我们仍在生成器内部使用行带宽。

```
压缩稀疏行_矩阵 *人工生成_矩阵_生成( 行数_行数, 列数_,
```

清单1：人工矩阵生成函数

B. 矩阵生成器

通过研究文献和矩阵套件中的真实矩阵，我们定义了表 I 所示的特征空间，使其尽可能均匀地涵盖大多数真实矩阵类型。随后，我们通过下文所述的矩阵生成器生成了覆盖该特征空间的人工矩阵。生成器函数与清单1中的函数类似，能够灵活定义矩阵特征。该函数以 CSR存储格式返回人工矩阵数据，然后我们将其转换为当前测试的存储格式。

行数和每行平均非零元数隐式地fi定义了矩阵的大小。生成器使用一种随机分布（由 分布 参数指定）来确定每行的非零元数。本研究采用正态分布：$\mathcal{N}$(平均 非零元 行数；标准差 非零元 行数)。

为了实现各行非零元素数的偏斜分布，我们采用了一个指数递减函数，并从所需的最大值开始，即 $MAX \cdot e^{-C \cdot \text{row idx nr rows}}$，其中 C 是计算得出的常数，用于控制函数的形状（并用于标准差）。随后重新计算前述分布函数的平均值，使每行非零元素数的综合平均值等于所需值。

选定非零元素数后，我们通过其余参数定义它们的位置。该算法按行处理。首先，以 cross_row_sim 所指定的概率，从上一行复制列位置，以实现相邻行相似度。最后一步使用随机定位方案。bw_scaled 参数表示矩阵带宽，并按列数 ([0-1]) 的比例进行缩放。bw_scaled 取值为 0.05、0.3 和 0.6。每行的非零元素会随机放置在该行的有限空间内，使平均带宽与给定值相符。每次随机放置后，我们还会根据 avg_num_neigh 推导出的概率，反复放置相邻邻居，直到随机判定失败。这样便可模拟同行非零元素聚集。

IV. 实验设置

在实验中，我们在三种不同架构（共九台不同机器）上评估稀疏矩阵-向量乘法：三款英伟达 GPU（Tesla-V100、Tesla-P100、Tesla-A100）、 five 台单插槽 CPU 系统（AMD-EPYC 7402、AMD-EPYC 7742、ARM- NEON Ampere Altra Q80-33、Intel Xeon Gold 5120、IBM Power9 AC922），以及一块 Xilinx Alveo-U280 现场可编程门阵列。表 II 更详细地列出了测试平台的硬件，以及所使用的格式和库。AMD-EPYC-24 和 AMD-EPYC-64 分别配置为 NPS1 和 NPS4（每插槽 NUMA 节点）。IBM CPU 每个核心最多支持 4 个硬件线程。经过测试，整体最佳配置确定为每个核心 2 个线程。其他 CPU 未进行特殊配置。始终使用 OpenMP 环境变量将线程固定到核心，并以并行方式初始化矩阵，利用 Linux fi先触碰策略来确保数据局部性。除现场可编程门阵列外，我们使用 STREAM 基准测试计算所有测试平台的内存带宽；对于现场可编程门阵列，则根据所使用的 HBM2 通道数量（32 个通道中使用了 20 个）计算带宽。关于多设备执行行为（例如双 CPU/插槽）的进一步研究留待未来开展。

如第 II-B 节所述，我们针对每种设备采用公开可用的“当前实践”格式，以及“研究”格式和实现，并避免使用专为某一特定矩阵族设计的格式，以便提供一个通用的 fi凤凰大参考

分析，我们为每种设备采用公开可用的“当前实践”格式，以及“研究型”格式和实现，并避免使用针对特定{v1}矩阵族专门设计的格式。在某些情况下，研究型格式还可能涉及预处理开销或调优复杂度。由于我们关注的是纯粹的稀疏矩阵-向量乘法性能，因此不将这些因素纳入分析。需要指出的是，由于无法充分访问 AMD-EPYC-64 系统，我们无法在该系统上运行其他 CPU 系统所测试的所有格式。不过，现有格式已足以充分展现总体架构趋势，且这些趋势与规模较小的 AMD-EPYC-24 的表现一致。此外，在 Tesla-A100 上测试的研究型格式范围受到限制，因为可用的、针对 CUDA-SDK 11 更新的格式较少；而 CUDA-SDK 11 是满足 A100 计算能力 8.0 要求的最低版本。

通过第 III-A 节所述的矩阵特征，我们定义了一个搜索空间（表 I），并在其中生成了 16200 个人工矩阵。对于每种配置（测试平台/矩阵/格式），我们运行了 128 次双精度稀疏矩阵-向量乘法迭代，并记录平均性能（ GFLOP）和功耗（瓦特），取 5 次实验的算术平均值。我们采用双精度作为科学计算的标准数据类型，其他精度级别的研究留待未来工作。对于 AMD 和 Intel CPU，我们在执行期间通过 RAPL（运行平均功耗限制）收集 CPU 封装（核心和非核心组件）的能耗读数，以计算平均功耗。同样，我们通过 Altra-HWMON（硬件监视器）驱动程序收集 ARM-NEON 的能耗读数。我们无法在 Power9 系统上准确测量功耗，因此对 CPU 插槽采用较为保守的估算值，即恒定的 200W 热设计功耗。为求完整，我们也纳入了这些结果。对于 GPU 和现场可编程门阵列，我们分别使用 nvidia-smi 的 NVML 驱动程序接口和 xbutil 工具，在执行期间收集功耗读数，以估算平均功耗。

V. 实验评估

A. 人工矩阵的验证

在基于我们的人工矩阵分析稀疏矩阵-向量乘法性能之前，我们需要验证矩阵生成方法能否以合理的准确度捕捉真实矩阵的行为。我们的基本验证假设是：如果该方法有效，那么特征相似的人工矩阵和真实矩阵应具有相似的性能。我们构建了一个验证集，其中包含先前文献中使用最广泛的45个矩阵，这些矩阵涵盖了所考察特征的广泛范围，如表III所列。对于每个真实矩阵，我们使用每项特征的广泛 $[-30\%、+30\%]$ 值范围，生成额外的人工矩阵（与表I中的人工数据集分开），并将这些矩阵称为其人工“朋友”（每个矩阵大约70个）。我们报告每个平台上每个矩阵在测试格式中达到的最佳性能。

图1展示了每个测试平台上验证矩阵（黑点）的性能，以及与其对应的“朋友”人工矩阵的性能范围（箱线图）。为了更好地理解性能，我们还提供了使用压缩稀疏行存储器估算的 Roofline 上界 [31]

表III：用于验证的矩阵套件。每个规则性（f4）子特征的取值范围均匀划分为 3 个区间，分别标记为 S（小）、M（中）、L（大）（“小”表示不规则矩阵）。

id	矩阵	f1	f2	f3	f4
1	scircuit	11.63	5.61	61.95	MM
2	mac_econ_fwd500	15.36	6.17	6.14	MS
3	raefsky3	17.12	70.22	0.14	LL
4	bbmat	20.42	45.73	1.76	LM
5	conf5_4-8x8-15	22.13	39	0	LL
6	mc2depi	26.04	3.99	0	LS
7	rma10	27.35	50.69	1.86	LL
8	cop20k_A	30.5	21.65	2.74	MM
9	热力机械 dK	33.35	13.93	0.44	MM
10	webbase-1M	39.35	3.11	1512.43	LS
11	cant	46.1	64.17	0.22	LL
12	专用集成电路68万	46.91	5.67	69710.56	LM
13	pdb1HYS	49.86	119.31	0.71	LL
14	TSOPF_RS_b300_c3	50.67	104.74	1	LL
15	切比雪夫4	61.8	78.94	861.9	LL
16	consph	69.1	72.13	0.12	LL
17	com-Youtube	72.71	5.27	5460.3	MS
18	rajat30	73.13	9.59	47421.8	MM
19	辐射	88.26	34.23	101.18	SS
20	斯坦福 伯克利	89.39	11.1	7519.69	MM
21	shipsec1	89.95	55.46	0.84	LL
22	PR02R	94.29	50.82	0.81	LM
23	gupta3	106.76	555.53	25.41	LL
24	mip1	118.73	155.77	425.24	LL
25	rail4284	129.15	2633.99	20.33	SL
26	pwtk	133.98	53.39	2.37	LL
27	曲柄段 2	162.16	221.64	14.44	LL
28	Si41Ge41H72	172.5	80.86	7.19	LM
29	TSOPF_RS_b2383	185.21	424.22	1.32	LL
30	in-2004	198.88	12.23	632.78	LL
31	Ga41As41H72	212.61	68.96	9.18	LM
32	eu-2005	223.42	22.3	312.27	LM
33	维基百科-20051105	232.29	12.08	410.37	SS
34	人类 基因1	282.41	1107.11	6.17	SS
35	德劳内 n22	304	6	2.83	MS
36	sx-stackoverflow	424.58	13.93	2738.46	SS
37	dgreen	442.43	31.87	4.87	SS
38	mawi_201512012345	506.18	2.05	8006372.09	LM
39	ldoor	536.04	48.86	0.58	LL
40	dielFilterV2real	559.9	41.94	1.62	MM
41	circuit5M	702.4	10.71	120504.85	LM
42	soc-LiveJournal1	808.06	14.23	1424.81	SS
43	bone010	823.92	72.63	0.12	LL
44	audikw_1	892.25	82.28	3.19	LL
45	cage15	1154.91	19.24	1.44	LS

表 II 展示了每个验证矩阵的内存占用，以及各测试平台测得的带宽。值得注意的是，由于高带宽内存容量有限，有 10 个矩阵无法在现场可编程门阵列上运行。此外，由于实验期间无法获得相应资源，我们也未能收集 3 台设备（Tesla-P100、AMD-EPYC-64、ARM-NEON）上某些矩阵的测量数据。我们得出以下观察结果：首先，许多验证矩阵和友元矩阵都接近其对应的屋顶线界限，这表明所用的内核实现已经过良好优化。与屋顶线界限偏差最大的情况包括：a) 图形处理器上的极小矩阵，无法充分利用其资源；b) 总体而言，现场可编程门阵列测试平台上的情况，过多的填充使屋顶线性能特征分析变得复杂；以及 c) 具有问题特征的矩阵（内存占用除外）。当矩阵的某项指标较差时，我们便称其为“问题矩阵”。

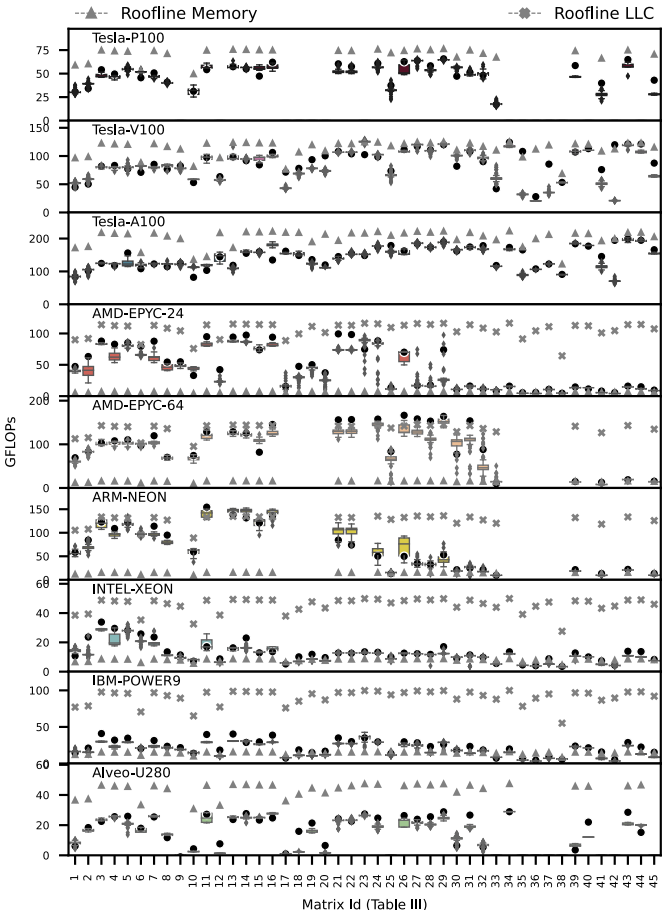


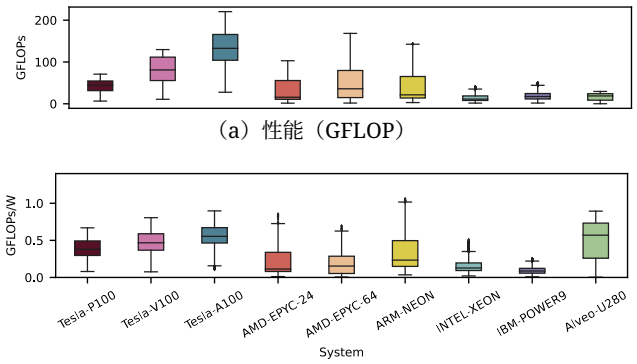
图1：表III中选定的验证矩阵与其在我们的测试平台上的“同伴”之间的性能比较。Roofline Memory 点（--Δ--）表示每个验证矩阵的内存带宽性能屋顶，而 Roofline LLC 点（--X--）表示相应的末级缓存屋顶。

表IV：比较每个验证矩阵与 1）其合成“朋友”的性能（平均绝对百分比误差 - MAPE），或 2）其最接近的“朋友”的性能（APE-best）

设备	MAPE	APE-best
Tesla-P100	10.01 %	4.57 %
Tesla-V100	18.42 %	10.15 %
Tesla-A100	9.94 %	5.19 %
AMD-EPYC-24	20.04 %	8.42 %
AMD-EPYC-64	21.81 %	6.39 %
ARM-NEON	15.65 %	4.41 %
英特尔至强	16.49 %	7.36 %
IBM-POWER9	21.77 %	14.11 %
Alveo-U280	23.49 %	16.63 %
平均值	17.51%	8.58%

性能无法仅由其内存占用解释，即它并非受内存带宽限制，而是由其他瓶颈（低指令级并行、负载不均衡、不规则性）的综合作用导致，这些因素无法通过内存屋顶线模型进行解读。

其次，在绝大多数情况下，人工生成的矩阵在所有设备上的性能趋势都能与真实矩阵保持一致。具体而言，平均绝对百分比误差



（a）性能（GFLOP）

（b）能效（GFLOP/W）

图 2：不同平台上稀疏矩阵-向量乘法的性能与能效。

（MAPE）为 17.51%，即在所有矩阵和测试平台上，每个验证矩阵的性能与其同类矩阵中位性能之间的误差。第三，某些验证矩阵区域存在问题，对 MAPE 的贡献最大，尤其是在 CPU 系统上，例如 AMD-EPYC-64 上的矩阵 30 至 33。对于每个测试平台，当验证矩阵的内存占用超过缓存大小时，便会出现相应的问题区域，导致性能下降（从末级缓存转为受内存屋顶线限制）；在此情况下，$[-30\%, +30\%]$特征搜索范围可能导致同类矩阵驻留在缓存中，而基于验证的矩阵却不在缓存中，反之亦然，从而增大 MAPE。最后，即使某些验证矩阵与其同类矩阵的中位性能差异显著，它们仍往往与至少一些人工生成的同类矩阵具有非常相似的性能。具体而言，将每个验证矩阵中性能最接近的同类矩阵定义为“最佳同类矩阵”后，每个验证矩阵与其最佳同类矩阵之间的 MAPE 仅为 8.58%。

要点 1：由内存占用、平均行大小、偏斜系数、平均邻居数和跨行相似度构成的小型特征集*f*能够以出色的准确度捕捉稀疏矩阵在所有设备上的性能表现。

B. 跨设备稀疏矩阵-向量乘法行为

本节使用人工矩阵数据集，评估所有平台上的稀疏矩阵-向量乘法性能与能效。同样，对于每个矩阵，我们展示测试格式中取得的最佳结果。

1）性能：图 2a 总结了各平台的性能（吞吐量）。主要观察结果如下：a）在所有设备上，我们的大型人工矩阵数据集中，稀疏矩阵-向量乘法性能差异显著；b）尽管 A100 的平均性能更佳，但核心数更多、缓存更大的最新一代中央处理器（AMD-EPYC-64、ARM-NEON）已成为许多情况下的有力替代方案，并能实现高性能；c）Alveo U280 的吞吐量无法与其他设备竞争。

要点 2：在性能方面，中央处理器无法取代 GPU，但仍是可靠的替代方案。

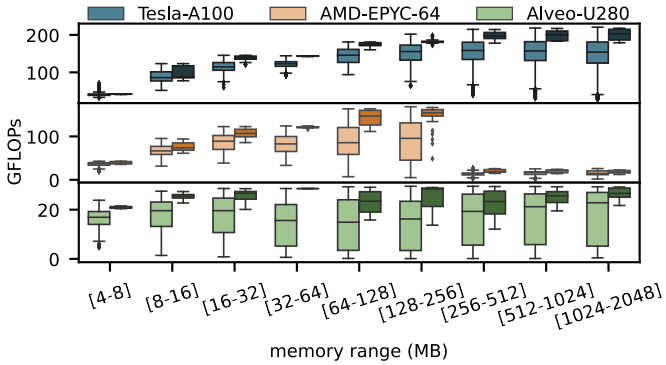


图 3：内存占用对稀疏矩阵-向量乘法性能的影响。浅色箱线图表示完整的矩阵数据集，深色箱线图表示各设备上具有有利特征的矩阵。

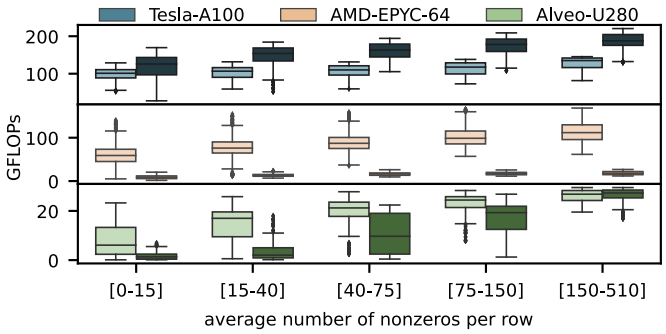


图 4：行大小对稀疏矩阵-向量乘法性能的影响。浅色箱线图表示小型矩阵，深色箱线图表示大型矩阵。所有设备的划分阈值均设为 256MB。

2）能效：除了关注纯粹的性能外，我们还关注所测试的各个平台的能效。能效可定义为系统在一段时间内完成的有效工作量与该系统在此期间消耗的能量之比 [32]。在分析中，我们采用 GFLOP/W 指标，并在图 2b 中展示结果。我们发现，尽管现场可编程门阵列的原始性能较弱，但它们仍是所测试平台中能效最高的，运行功耗也低于其他设备。图形处理器在能效方面保持竞争力，这可归因于它们所达到的高性能水平。中央处理器在这一指标上似乎明显落后，ARM-NEON 是个例外，它是唯一在功耗方面表现突出的中央处理器。

要点3：三种能效更*f*i高的设备分别沿着三条不同的路径实现能效*f*i：“低功耗”（Alveo-U280）路径、“高性能”（Tesla-A100）路径，以及“均衡架构”（ARM-NEON）路径。

C. 特征分析

在本小节中，为简化分析并节省篇幅，我们分析一个图形处理器设备（Tesla-A100）、一个 CPU 设备（AMD-EPYC-64）以及唯一可用的现场可编程门阵列的结果。我们按特征展示实验结果，以便更好地理解各项特征对每种设备上稀疏矩阵-向量乘法性能的影响。因此，尽管这些设备可用的存储格式相对较少，我们仍选择展示性能最强的图形处理器和 CPU。

1) 内存占用（矩阵大小）的影响：图 3 展示了内存占用（以 MB 为单位）对性能的影响。每种设备对应两个箱线图：一个包含所有测量结果（浅色箱线图），另一个仅包含其他三个特征均被选为直观上有利的矩阵（即规则、均衡且行较长的矩阵）。显然，内存占用（矩阵大小）是影响 GPU 和 CPU 性能的极其重要的特征，但在现场可编程门阵列上则表现得相当随机。

对于 A100，随着矩阵规模增大，设备会逐渐利用更多可用并行度，因此相比小型矩阵更偏好大型矩阵。我们注意到，小型矩阵与大型矩阵之间的性能差距可能超过 2×。这种现象归因于现代 GPU 所具备的极高并行度，这需要足够的工作量才能让数千个执行单元持续忙碌。我们还注意到，在少数大型矩阵的情况下，性能可能会骤降至远低于小型矩阵的水平，这表明其他三项特征虽不常见，却会产生重大影响。值得注意的是，对于 GPU，矩阵大小不会影响内存带宽强度，而是会影响可用并行度的水平。

CPU 的总体性能趋势相当符合预期。能够装入末级缓存的矩阵性能出色。相反，由于该内核的内存带宽强度极高，大型矩阵的性能会急剧下降；对于 AMD-EPYC-64 CPU，小型矩阵与大型矩阵之间的性能差距超过7× 。不过需要指出的是，由于 CPU 的末级缓存容量相当大，当代 CPU 似乎能够在中等规模的矩阵上实现较高性能，也就是说，矩阵包含 10-20M 个非零元素且最多有 1M 行。测试的其他 CPU 也表现出类似趋势：随着矩阵内存占用增大，性能会发生变化；不过，各设备末级缓存容量不同，因此其性能“截断”点也各不相同（AMD-EPYC-64 的截断点在 256 MB 以上）。

如前所述，所有受考察的 CPU 都无法达到 Tesla-A100 GPU 的性能水平。然而，如果我们关注内存占用在 64-256MB 范围内的矩阵，可以看到 AMD-EPYC-64 CPU 能达到 Tesla-A100 性能的 60%。同样，对于 ARM-NEON，若关注内存占用在 32-128MB 范围内的矩阵，则可达到 A100 性能的 72%。

要点 4：（接续要点 2）中央处理器在适宜的操作规模下，性能已接近高端 GPU。

现场可编程门阵列的性能分析更为复杂。我们注意到，由于 Vitis Sparse Library 和底层硬件的限制，数据集中并非所有矩阵都能在 Alveo FPGA 上运行。具体而言，行数多、稀疏度高（每行非零元素少）且不规则的矩阵会产生大量零填充，导致内存占用超过所用 HBM 通道的容量。因此，图 3 所示的大多数大型（大于 256 MB）矩阵在现场可编程门阵列上的性能更好且更稳定（箱线图更窄），直观来看更有优势

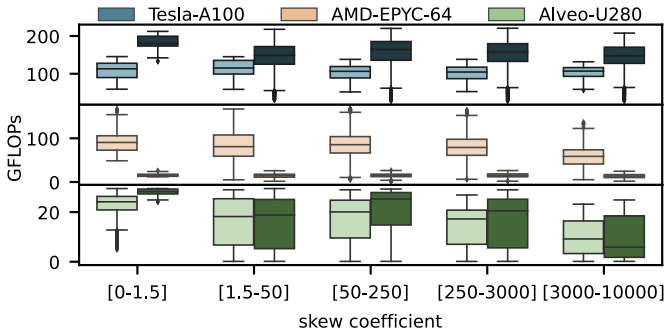


图 5：负载不均衡对 SpMV 性能的影响。偏斜度越高，矩阵的负载越不均衡。浅色箱线图表示小型矩阵，深色箱线图表示大型矩阵。所有设备的划分阈值均设为 256MB。

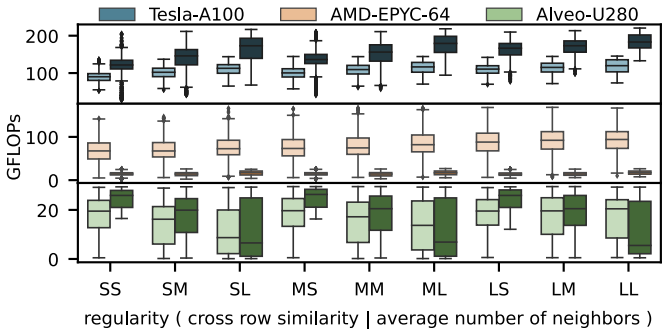


图 6：规则性对稀疏矩阵-向量乘法性能的影响。子特征的取值设为 S（小）、M（中）、L（大）。数值越高，表示矩阵越规则。浅色箱线图表示小型矩阵，深色箱线图表示大型矩阵。所有设备的划分阈值均设为 256MB。

与小型矩阵相比。总的来说，我们发现内存占用会导致性能表现相当随机，因此需要考察其他特征，以更深入地了解现场可编程门阵列上的稀疏矩阵-向量乘法性能。

2）行大小：图 4 展示了行大小对稀疏矩阵-向量乘法性能的影响。根据上一段的讨论，由于 CPU 和 GPU 在不同矩阵大小范围内的行为差异很大，因此我们将大型矩阵和小型矩阵的结果分开分析（各设备采用不同的划分界限）。我们也对现场可编程门阵列采用相同的方法，以考察矩阵大小与行大小的组合是否会产生有趣的规律。初步观察表明，这一特征也会显著影响性能；不过，对于 GPU 和 CPU，其影响在各自适宜的矩阵大小范围内更大（CPU 对应小型矩阵，GPU 对应大型矩阵）。具体而言，分别查看 GPU 和 CPU 在大型矩阵和小型矩阵上的中位数，可以发现长行与短行之间的差距约为 $2\times$。

在现场可编程门阵列上，对于大型矩阵，行长度较短导致的性能下降为 $\sim 20\times$，这验证了我们之前的说法：由于零填充，该库无法很好地处理高度稀疏的大型矩阵。对于小型矩阵，小行与大行之间的差距为 $\sim 7。5\times$。在这两种情况下，行较大的矩阵都能达到该格式在此平台上所能实现的峰值性能。

3) 负载不均衡：图 5 展示了负载不均衡对稀疏矩阵-向量乘法性能的影响。与行大小的情况一样，我们分别展示大型矩阵和小型矩阵的结果，并针对不同设备采用不同的划分阈值。偏斜系数增加会使矩阵的负载更加不均衡，其中一小部分行相较于其余行更长。我们分别展示小型矩阵和大型矩阵的结果。

对于图形处理器和 CPU，均衡矩阵与不均衡矩阵之间未观察到显著差异。CPU 不太容易受到此特性的影响，主要是因为其并行度较低：矩阵会被分配给线程处理，且分块较大，因此能够更容易地掩盖负载不均衡。另一方面，大多数图形处理器格式在设计时都考虑了工作共享和负载不均衡问题，这是由于其具有极高的并行度（例如 ELL格式中的零填充、CSR5 中的行拆分，以及 HYB 中的坐标格式）。对于图形处理器而言，高度均衡的矩阵仅能带来 1.fi 的性能提升。

相比之下，现场可编程门阵列的性能下降了 $\sim 4\times$，且大尺寸矩阵受到的性能下降更为严重。

4) 不规则性：我们考察的最后一个特征是行内邻居数（avg_num_neighbors）与相邻行相似度（cross_row_similarity）的组合，它们可作为衡量矩阵不规则性的指标。与之前相同，我们分别分析小型和大型矩阵的结果。在图 6 中，我们将每个子特征的取值范围等分为三个区间，并将各子特征标记为 S（小）、M（中）、L（大），其中“小”表示矩阵不规则。

聚焦于各平台适用的矩阵大小，我们首先观察到，图形处理器在运行大型矩阵时容易受到不规则性的影响。矩阵越规则，性能就越稳定（箱线图向上收缩）。在 CPU 上，不规则性的影响较小；矩阵变得规则时，性能会提升 $\sim 1.3\times$。我们认为，这是因为对于能够放入缓存的小型/中型矩阵，x向量也有望放入缓存，因此不会遭遇缓存未命中。对于更大的矩阵，与内存带宽限制相比，x向量缓存未命中带来的额外代价似乎较小。最后，在现场可编程门阵列上，我们观察到一些意料之外的现象。当 avg_num_neighbors 设为“大”时，无论 cross_row_similarity 的值如何，性能都会下降。对此的一种可能解释是，当 avg_num_neighbors 设为“大”时，由于应用的零填充较少，可能会生成更多矩阵。因此，avg_num_neighbors 为“大”的箱线图中包含许多大型矩阵，而这些矩阵在现场可编程门阵列上的表现不佳。

要点 5：我们的观察结果可总结如下：

- 对于 CPU 而言，影响最大的特征是矩阵的内存占用。当超过其缓存大小时，性能可能下降 $7\times$。下一个最重要的特征是行长度，较小的行长度会使性能降低 $\sim 2\times$。负载不均衡得到了充分处理，因为 CPU 中的并行度并不高，

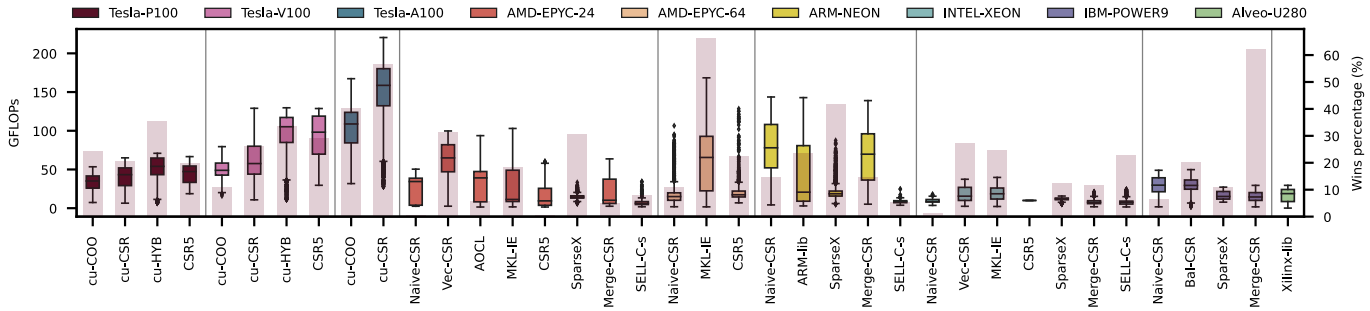


图 7：不同设备上 SpMV 格式/库的性能比较。每种格式后方的条形图表示，对于各设备而言，人工矩阵数据集中使用该格式可实现最佳性能的部分所占百分比。“Naive-CSR”是标准 CSR，“Bal-CSR”增加了非零元均衡（行分解），“Vec-CSR”还增加了操作向量化。

而不规则性也不会造成重大问题。

- 对于 GPU 而言，大型矩阵不会造成问题，与小型矩阵相比，性能最高可达 $2\times$。与 CPU 类似，行长度较小时，性能会下降 2 倍。借助性能最佳格式中的专用工作共享机制，负载不均衡不会影响性能。然而，不规则性对 GPU 来说是个问题，因为大型矩阵（适合 GPU 的矩阵大小）的性能会大幅下降，最多可达 $2\times$。
- 对于现场可编程门阵列，矩阵大小不会影响性能。另一方面，行长度至关重要，因为由于大量零填充，库无法很好地处理高度稀疏的大型矩阵。负载不均衡会使性能最多下降 $\sim 4\times$，大型矩阵尤为明显。至于不规则性，由于大量填充和内存容量有限，观察到了一种反常现象。

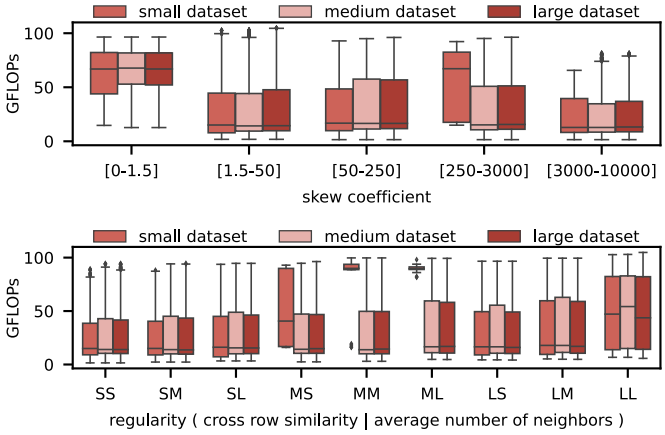


图 8：在 AMD-EPYC-24 CPU 上测试的三种不同人工数据集规模的性能变化对比。“小”“中”“大”数据集分别包含 3K、16K、27K 个矩阵。

D. 矩阵格式的影响

在本小节中，我们研究各设备所用不同格式的性能。图 7 分别总结了我们分析中测试的每种格式在各个平台上的性能。每个箱线图后方柱状图的高度表示在该特定格式表现最佳（“获胜”）的矩阵所占百分比。例如，对于 AMD-EPYC-24 系统，MKL-IE 是 $\sim$17% 的数据集所对应的最佳格式。从该图可以看出，各设备并没有明显占优的存储格式，因为我们观察到所有格式都能取得相当高比例的胜出。

我们还发现，“研究型”格式和实现有一个共同特点：尽管它们可能在许多情况下胜出，但通常无法达到很高的原始性能。具体而言：SparseX [28] 通过识别稀疏矩阵中的子模式来减少内存占用，因此主要针对无法 fi装入 CPU缓存的矩阵。SELL-C-σ [27] 旨在利用向量扩展等硬件能力来提高指令级并行，因此（与所有 ELL变体一样）更适用于规则矩阵，并能更高效地压缩规模较大的矩阵。CSR*5* [20]主要是为缓解 GPU架构上矩阵负载不均衡和不规则性的影响而设计的。最后，Merge-CSR [26] 凭借在线程间“均等分配工作负载”的方式，在高度负载不均衡的矩阵上表现出色。

结论 6：没有哪种格式/库明显胜出，稀疏矩阵-向量乘法优化仍是一个开放的 *fi*领域。

要点 7：“研究型”格式能够处理更通用的供应商提供的格式无法胜任的情况，例如规模较大、负载不均衡程度高且不规则性强的矩阵。

E. 人工数据集规模的选择

在本小节中，我们将讨论如何为人工矩阵数据集选择合适的规模。最初，我们使用一个小型数据集进行评估，该数据集约包含 3000 个矩阵，旨在模拟 SuiteSparse 稀疏矩阵集合 [30] 的总体规模。然而，在这个小型数据集上，我们观察到某些特征值出现了意料之外的行为，如图 8 所示。随后，我们在保持特征空间范围不变的同时，将“小型”数据集扩充为“中型”数据集。表 I 描述了该数据集；我们在第 V-C 节中使用它分析所有设备。之后，我们再次在保持

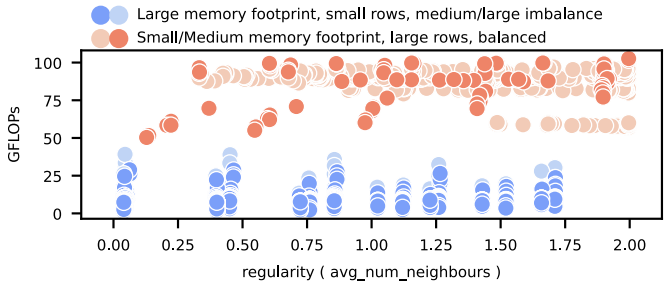


图 9：随着规则性特征增大，AMD-EPYC-24中央处理器上的性能变化。不同颜色表示不同的“固定”特征值范围组合（内存占用、行长度、负载不均衡）。

特征空间范围不变的基础上，进一步采样更多特征值，将数据集扩充为“大型”数据集。这个大型数据集包含 27000 个矩阵，并在 AMD-EPYC-24 测试平台上进行了测试，以确定增加矩阵数量是否会为我们的观察带来新的视角。如图 8 所示，与“中型”数据集相比，扩大数据集并未改变总体性能趋势。鉴于在所有设备上进行基准测试既耗时又耗电，我们得出结论：包含 16K 个矩阵的“中型”数据集规模适宜，可用于研究所有设备上的稀疏矩阵-向量乘法性能。

还需强调生成器的灵活性：用户可以在特定特征空间中生成矩阵，以便研究聚焦于特定矩阵属性或特定架构。

F. 深入数据分析的机会

最后，我们考察如何利用收集到的结果，更好地理解特定瓶颈对每种设备的影响。在图 9 中，对于 AMD-EPYC-24 设备，允许（规则性特征中的）平均邻居数子特征变化，而其他三个特征则被划分并固定在三个特征值范围内：小、中、大。我们注意到，当矩阵在这三个“固定”特征上的取值直观上不利于 CPU 执行时，无论考察规则性的何种取值，性能都不可避免地较低。在这种情况下，最高只能达到峰值性能的 40%。另一方面，对于直观上较理想的固定特征（矩阵较小或中等大小、行较长且负载不均衡程度较低），增加每行的平均邻居数确实可以将性能提升多达 1.6×。为简洁起见，我们不再提供其他图表。也可以针对其他特征和设备进行类似的性能研究，以确定每种设备的最优特征值范围。

VI. 相关工作

稀疏矩阵-向量乘法是一种研究得十分透彻的内核。早期关于 SpMV 的研究包括提出各种存储格式，其中 COO 和 CSR/CSC 不依赖矩阵的稀疏结构，而对角格式（DIA、JAD）或分块格式（BCOO、BCSR、VBR）则表示特定的结构。实际上，不同格式的 SpMV 性能既取决于矩阵的稀疏模式，也取决于目标系统的架构特征 [33]。

人们投入了大量精力来优化串行稀疏矩阵-向量乘法 [34], [35], , 重点关注寄存器级和缓存级分块。早期关于多核 CPU 的研究 [1], [23] 指出了本文讨论的瓶颈，并提出通过自动调优 [36],和压缩格式 [28], [37],[38]来缓解这些瓶颈的技术。后续研究则提出了面向众核处理器[27], [39], [40], 的格式，重点关注 SIMD 效率fi。多核 CPU 上稀疏矩阵-向量乘法的先进实践包括 CSR 稀疏矩阵-向量乘法实现，以及英特尔数学核心函数库中的检查器-执行器压缩稀疏行格式稀疏矩阵-向量乘法 [41], ; ARM 性能库中的 CSR 稀疏矩阵-向量乘法和针对矩阵结构优化的稀疏矩阵-向量乘法 [42]; 以及 AMD AOCL 库中的检查器-执行器压缩稀疏行格式稀疏矩阵-向量乘法 [43]。

GPU 上的稀疏矩阵-向量乘法优化遵循了类似的发展趋势，针对不同瓶颈采用不同格式：HYB [18] 针对 SIMD 优化和负载均衡，CSR5[20]也旨在提高 SIMD 利用率，Merge [26] 旨在改善负载均衡，而 BCCOO [19] 则应对内存限制。一项针对 GPU 稀疏矩阵-向量乘法方法的全面综述 [21],指出，目前尚无一种格式能够使所有矩阵都达到最佳性能。GPU 上稀疏矩阵-向量乘法的先进实践包括 NVIDIA cuSPARSE 库 [22],, 该库支持多种存储格式及相应的稀疏矩阵-向量乘法函数，包括分块 CSR 格式。

FPGA 设计专注于 CSR 定制化 [44]–[49], , 试图弥补 FPGA 在内存带宽方面的劣势。近期，商用 FPGA 已配备高带宽内存（HBM）通道。近期的研究 [50]–[52] 提出了基于 HBM 的稀疏矩阵-向量乘法加速器，但这些加速器仅支持 fi定点或单 fl精度浮点运算。FPGA 上稀疏矩阵-向量乘法的当前实践包括 Vitis Sparse Library [25], 以及面向 HBM 的压缩稀疏列稀疏矩阵-向量乘法加速器。

许多研究基于输入矩阵 [3]–[11] , 致力于预测多核和众核 CPU 及 GPU 上稀疏矩阵-向量乘法的最优存储格式，并在目标平台上对稀疏矩阵-向量乘法进行自动调优 [12]–[17]。我们工作中所述的人工矩阵生成器受到其中一部分研究的启发[4]–[6], [8], [13]–[15], [17],; 这些研究定义了矩阵特征，并探究或建模这些特征与预期性能之间的关联，考察因素为存储格式。

尽管所有与稀疏矩阵-向量乘法相关的研究都进行了性能和/或能效评估，但近期专门关注稀疏矩阵-向量乘法性能的研究十分有限。据我们所知，这是首个全面研究当前稀疏矩阵-向量乘法实践在多种现代架构上的性能与能效的工作。我们认为，更新对稀疏矩阵-向量乘法性能表现的认识，对于支持不断发展的优化工作至关重要，因为稀疏矩阵-向量乘法是一个极其关键的内核。

稀疏矩阵应用研究通常依赖稀疏矩阵集合 [29], [30],, 因为随机生成的矩阵在性能和数值稳定性方面并不能代表真实稀疏矩阵。

我们的稀疏矩阵生成器旨在提供能够模拟真实世界稀疏矩阵的稀疏模式，从而支持针对不同架构上矩阵稀疏模式的研究。需要指出的是，我们不考虑稀疏矩阵的数值特性。

七、结论

在本研究中，我们对现代架构上稀疏矩阵-向量乘法的行为进行了广泛研究。我们的工作基于 SpMV 已知的瓶颈，即内存带宽强度低、指令级并行度低、负载不均衡以及内存延迟开销；这些瓶颈可能出现在所有架构上，具体取决于矩阵的稀疏模式、所选存储格式以及相应的算法。我们提取了矩阵稀疏模式的 fi 个特征，并认为这些特征对性能的影响最为显著。基于这些特征，我们使用稀疏矩阵生成器创建了一个包含 16200个矩阵的数据集，并通过真实矩阵验证了该数据集对预期性能的反映能力。借助这一大规模数据集，我们在九种不同的现代架构上探索了 SpMV 的性能和能效，包括两个 AMD EPYC、一种 ARM NEON、一种 Intel Xeon 和一种 IBM Power9 中央处理器，NVIDIA P100/V100/A100 GPU，以及 Xilinx Alveo U280 现场可编程门阵列；同时采用了“当前实践”和“研究型”存储格式及 SpMV 实现。

我们的 findings揭示了在当代架构上执行稀疏矩阵-向量乘法的潜力。我们得出结论：尽管图形处理器在高性能稀疏矩阵-向量乘法中仍占据主导地位，但中央处理器在执行小型和中型矩阵时也极具竞争力，而现场可编程门阵列则是能效最高的选择。通过对各项特征进行详细分析，我们量化了不同瓶颈对各种架构的影响。矩阵规模是决定预期性能的首要因素，其次是每行非零元素数的平均值，这表明稀疏矩阵-向量乘法仍然是一种内存受限算法；但低指令级并行度，以及在行数较少的矩阵中出现的更低运算强度等瓶颈，也值得高度关注。不规则内存访问导致的内存延迟开销在图形处理器上尤为明显。另一方面，尽管现代架构的并行度有所提升，大多数存储格式仍能有效处理负载不均衡。

致谢

我们感谢匿名评审对本文提出的真知灼见和宝贵建议。我们还感谢CSLab研究组成员对这项工作提出的宝贵意见。本研究获得了欧盟“地平线2020”研究与创新计划的资助，并获得希腊发展与投资部下属希腊研究与创新总秘书处通过欧洲高性能计算联合事业（EuroHPC JU）资助协议第956416号（exaFOAM项目）和第955739号（OPTIMA项目）提供的资助。

参考文献

[1] G. Goumas、K. Kourtis、N. Anastopoulos、V. Karakasis 和 N. Koziris，《现代架构上的稀疏矩阵-向量乘法性能评估》，《超级计算杂志 2009。

[2] A. Elafrou、G. Goumas 和 N. Koziris，《现代多核与众核处理器上的稀疏矩阵-向量乘法性能分析与优化》，发表于 ICPP 2017。[3] B.-Y. Su 和 K. Keutzer，《clSpMV：面向 GPU 的跨平台 OpenCL 稀疏矩阵-向量乘法框架》，发表于 ICS 2012。[4] J. Li、G. Tan、M. Chen 和 N. Sun，《SMAT：一种输入自适应的稀疏矩阵-向量乘法自动调优器》，发表于 PLDI 2013。[5] N. Sedaghati、T. Mu、L.-N. Pouchet、S. Parthasarathy 和 P. Sadayappan，《GPU 上稀疏矩阵表示的自动选择》，发表于 ICS 2015。[6] A. Benatia、W. Ji、Y. Wang 和 F. Shi，《使用多类别 SVM 为 GPU 上的稀疏矩阵-向量乘法选择稀疏矩阵格式》，发表于 ICPP 2016。[7] ——，《使用机器学习方法预测 GPU 上稀疏矩阵-向量乘法的性能》，发表于 ICPADS 2016。[8] I. Nisa、C. Siegel、A. S. Rajam、A. Vishnu 和 P. Sadayappan，《基于机器学习的有效格式选择与 GPU 上稀疏矩阵-向量乘法性能建模》，发表于 IPDPSW 2018。[9] J. C. Pichel 和 B. Pateiro-L´opez，《基于深度学习技术的稀疏矩阵分类新方法》，发表于 CLUSTER 2018。[10] Y. Zhao、J. Li、C. Liao 和 X. Shen，《弥合深度学习与稀疏矩阵格式选择之间的差距》，发表于 PPoPP 2018。[11] G. Zhu 和 G. Agrawal，《基于采样的 GPU 稀疏格式选择》，发表于 SBAC-PAD 2021。[12] G. Xiao、Y. Chen、C. Liu 和 X. Zhou，《ahSpMV：神威架构上用于稀疏矩阵-向量乘法的自动调优混合计算方案》，IEEE物联网期刊 2019。[13] G. Tan、J. Liu 和 J. Li，《面向多核与众核架构的自适应稀疏矩阵向量乘法库的设计与实现》，TOMS 2018。[14] A. Benatia、W. Ji、Y. Wang 和 F. Shi，《BestSF：用于优化 GPU 稀疏矩阵-向量乘法的稀疏元格式》，TACO 2018。[15] K. Hou、W.-c. Feng 和 S. Che，《面向多核与众核处理器上稀疏矩阵-向量乘法并行化的自动调优策略》，发表于 IPDPSW 2017。[16] K. Li、W. Yang 和 K. Li，《基于概率建模的 GPU 稀疏矩阵-向量乘法性能分析与优化》，IEEE并行与分布式系统汇刊 2014。[17] A. Elafrou、G. Goumas 和 N. Koziris，《BASMAT：面向通用GPU的瓶颈感知稀疏矩阵-向量乘法自动调优》，发表于 PPoPP 2019。[18] N. Bell 和 M. Garland，《面向吞吐量的处理器上的稀疏矩阵-向量乘法实现》，发表于 SC’09。[19] S. Yan、C. Li、Y. Zhang 和 H. Zhou，《yaSpMV：另一个 GPU 稀疏矩阵-向量乘法框架》，ACM SIGPLAN 通知 2014。[20] W. Liu 和 B. Vinter，《CSR5：一种用于跨平台稀疏矩阵-向量乘法的高效存储格式》，发表于 ICSR 2015。[21] S. Filippone、V. Cardellini、D. Barbieri 和 A. Fanfarillo，《通用GPU上的稀疏矩阵-向量乘法》，TOMS 2017。[22] 英伟达，《NVIDIA cuSPARSE 库（2022）》。 [在线]。网址：https://bit.ly/ 3 7 5 e uoc S. Williams、[23] L. Oliker、R. Vuduc、J. Shalf、K. Yelick 和 J. Demmel，《新兴多核平台上的稀疏矩阵-向量乘法优化》，发表于 SC’07。[24] G. Goumas、K. Kourtis、N. Anastopoulos、V. Karakasis 和 N. Koziris，《理解稀疏矩阵-向量乘法的性能》，发表于 PDP 2008。[25] 赛灵思，《Vitis 稀疏库（2021）》。 [在线]。网址：https: //bit.ly/ 3 8 MgfbC D. Merrill 和 M. Garland，《基于归并的稀疏矩阵向量并行乘法》，[26] 发表于 SC’16。[27] M. Kreutzer、G. Hager、G. Wellein、H. Fehske 和 A. R. Bishop，《一种统一的稀疏矩阵数据格式，用于在配备宽 SIMD 单元的现代处理器上高效执行通用稀疏矩阵-向量乘法》，《SIAM 科学计算期刊》2014。[28] A. Elafrou、V. Karakasis、T. Gkountouvas、K. Kourtis、G. Goumas 和 N. Koziris，《SparseX：面向多核平台上高性能稀疏矩阵-向量乘法的库》，TOMS 2018。[29] R. F. Boisvert、R. Pozo、K. Remington、R. F. Barrett 和 J. J. Dongarra，《矩阵市场：测试矩阵集合的网络资源》，发表于《数值软件质量》1997。[30] T. A. Davis 和 Y. Hu，《佛罗里达大学稀疏矩阵集合》，TOMS 2011。[31] S. Williams、A. Waterman 和 D. Patterson，《Roofline：面向多核架构的深刻可视化性能模型》，《ACM 通讯》2009。

[32] H. Giefers、P. Staar、C. Bekas 和 C. Hagleitner，《异构系统上稀疏矩阵乘法的能效分析：GPU、Xeon Phi 和 FPGA 的比较研究》，发表于 ISPASS 2016年。 [33] J.-H. Byun、R. Lin、K. A. Yelick 和 J. Demmel，《针对多核的稀疏矩阵-向量乘法自动调优》，电气工程与计算机科学系，加州大学伯克利分校，2012年技术报告。 [34] E.-J. Im、K. Yelick 和 R. Vuduc，《Sparsity：稀疏矩阵内核的优化框架》，《国际高性能计算应用期刊》2004年。 [35] R. Vuduc、J. W. Demmel 和 K. A. Yelick，《Oski：自动调优稀疏矩阵内核库》，发表于《物理学杂志：会议系列》2005年。 [36] A. Jain 和 K. Yelick，《pOSKI：用于在多核架构上执行优化稀疏矩阵-向量乘法的可扩展自动调优框架》，博士论文 2008年。 [37] K. Kourtis、G. Goumas 和 N. Koziris，《利用索引和值压缩提升多线程稀疏矩阵-向量乘法的性能》，发表于 ICPP 2008。 [38] K. Kourtis、V. Karakasis、G. Goumas 和 N. Koziris，《CSX：适用于共享内存系统上稀疏矩阵-向量乘法的扩展压缩格式》，ACM SIGPLAN 通知 2011年。 [39] X. Liu、M. Smelyanskiy、E. Chow 和 P. Dubey，《基于 x86 的众核处理器上的高效稀疏矩阵-向量乘法》fi，发表于 ICS 2013。 [40] B. Xie、J. Zhan、X. Liu、W. Gao、Z. Jia、X. He 和 L. Zhang，《Cvr：x86 处理器上稀疏矩阵-向量乘法的高效向量化》fi，发表于 CGO 2018。 [41] E. Wang、Q. Zhang、B. Shen、G. Zhang、X. Lu、Q. Wu 和 Y. Wang，《Intel® Xeon PhiTM 上的高性能计算》，发表于 Springer International Publishing 2014。 [42] ARM，《ARM 性能库（2022）》。 [在线]。网址：https://bit.ly/3eab3Xu

[43] AMD，《AOCL 稀疏库（2022年）》。 [在线]。获取地址：https://bit.ly/3T3yYa0 [44] L. Zhuo 和 V. K. Prasanna，《现场可编程门阵列上的稀疏矩阵-向量乘法》，发表于 FPGA 2005。 [45] Y. Zhang、Y. H. Shalabi、R. Jain、K. K. Nagar 和 J. D. Bakos，《现场可编程门阵列与图形处理器在稀疏矩阵-向量乘法方面的比较》，发表于 FPT’09。 [46] S. Kestur、J. D. Davis 和 E. S. Chung，《面向通用的 FPGA 矩阵-向量乘法架构》，发表于 FCCM 2012。 [47] P. Grigoras、P. Burovskiy、E. Hung 和 W. Luk，《通过压缩非零值加速 FPGA 上的稀疏矩阵-向量乘法》，发表于 FCCM 2015。 [48] M. Hosseinabady 和 J. L. Nunez-Yanez，《使用高级综合实现稀疏矩阵-向量乘法的流式数据流引擎》，《IEEE 集成电路与系统计算机辅助设计汇刊》，2019年。fiP. Mpakos、N. Papadopoulou、C. Alverti、G. I. Goumas 和 N. Koziris，《FPGA 上稀疏矩阵-向量乘法的性能与能效》，发表于 PARCO 2019。 [49] Y. Hu、Y. Du、E. Ustun 和 Z. Zhang，《Graphlily：在配备高带宽内存的 FPGA 上加速图线性代数》，ICCAD 2021。fiL. Song、Y. Chi、L. Guo 和 J. Cong，《Serpens：一种基于高带宽内存的通用稀疏矩阵-向量乘法加速器》，发表于 DAC ’22。 [50] Y. Du、Y. Hu、Z. Zhou 和 Z. Zhang，《使用高级综合在配备高带宽内存的 FPGA 上实现高性能稀疏线性代数：以稀疏矩阵-向量乘法为例》，发表于 FPGA 2022。 [51] P. Sandhu、C. Verbrugge 和 L. Hendren，《对稀疏矩阵-向量乘法进行完全结构驱动的性能分析》，发表于 ICPE’20。 [52] B. Xie、Z. Jia 和 Y. Bao，《众核平台上稀疏矩阵-向量乘法方法的基准测试》，发表于《基准测试、测量与优化》，2019年。