选择稀疏矩阵乘法最优存储格式的研究

李佳佳[1,2] 张秀霞[1,2] 谭光明[1] 陈明宇[1]
[1]（中国科学院计算技术研究所计算机体系结构国家重点实验室 北京 100190）
[2]（中国科学院大学 北京 100049）
(lijiajia@ict.ac.cn)

The Study of Choosing the Optimal Storage Format of Sparse Matrix Vector Multiplication

Li Jiajia[1,2], Zhang Xiuxia[1,2], Tan Guangming[1], and Chen Mingyu[1]
[1](State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)
[2](University of Chinese Academy of Sciences, Beijing 100049)

Abstract Sparse Matrix Vector Multiplication ("SpMV") is one of the most important kernels in scientific and engineering applications. It is also one of thJe most essential subprograms of sparse BLAS library. A lot of work has been dedicated in optimizing SpMV, and some has achieved significant performance improvement. Since most of the optimization methods are less of generalization and only suitable for a specific type of sparse matrices, the optimized SpMV kernels have not been widely used in real applications and numerical solvers. Besides, there are many storage formats of a sparse matrix and most of them achieve diverse performance on different SpMV kernels. In this paper, considering different sparse matrix features we presented an SpMV Auto-Tuner (SMAT) to choose the optimal storage format for a given sparse matrix on different computer architectures. The optimal storage format releasing the highest SpMV performance is helpful to enhance the performance of applications. Moreover, SMAT is also extensible to new formats, which will make full use of the achievements of SpMV optimization in literatures. We tested SMAT using 2366 sparse matrices from university of Florida sparse matrix collection. SMAT achieves 9.11 GFLOPS (single) and 2.44 GFLOPS (double) on Intel platform, and 3.36 GFLOPS (single) and 1.52 GFLOPS(double) on AMD platform. Compared with Intel MKL library, the speedup of SMAT is 1.4--1.5 times.

Key Words SpMV; auto-tuning; numerical solver; sparse matrix; SpBLAS

摘要 稀疏矩阵向量乘法（简称“SpMV”）是科学和工程领域中重要的核心子程序之一，也是稀疏 BLAS 库的重要函数. 目前很多 SpMV 的优化工作在不同程度上获得了性能提升，但大多数优化工作针对特定存储格式或一类具有特定特征的稀疏矩,缺乏通用性. 因此高性能的 SpMV 实现并没有广泛地应用于实际应用和数值解法器中. 另外，稀疏矩阵具有众多存储格式，不同存储格式的 SpMV 存在较大性能差异. 根据以上现象，提出一个 SpMV 的自动调优器（SMAT）. 对于一个给定的稀疏矩阵，SMAT 结合矩阵特征选择并返回其最优的存储格式. 应用程序通过调用 SMAT 来得到合适的存储格式从而获得性能提升，同时随着 SMAT 中存储格式的扩展更多的 SpMV 优化工作可以将性能优势在实际应用中发挥作用. 使用佛罗里达大学的 2366 个稀疏矩阵作为测试集，在 Intel 上 SMAT 分别获得 9.11GFLOPS（单精度）和 2.44GFLOPS（双精度）的最高浮点性能，在 AMD 平台上获得了 3.36GFLOPS（单精度）和 1.52GFLOPS（双精度）的最高浮点性能. 相比 Intel 的 MKL 数学库，SMAT 平均获得 1.4~1.5 倍的性能提升.

关键词 SpMV；自动调优；数值解法器；稀疏矩阵；SpBLAS

中图法分类号 TP302

收稿日期：2012-09-27；修改日期：2013-05-03
基金项目：国家自然科学基金项目（61272134，61033009, 61003062,60925009）；国家“九七三”重点基础研究计划基金项目（2011CB302502，2012CB316502）

0 引言

稀疏矩阵向量乘法（简称“SpMV”）是科学和工程领域中重要的核心子程序之一，SpMV 的性能对应用起着至关重要的作用，如激光聚变大规模数值模拟和电路模拟等应用. 由于应用中的求解线性方程组的部分通常通过调用数值解法器[1]实现，因此 SpMV 在数值解法器中的性能直接决定了 SpMV 在实际应用中的性能. 本文用数值解法器代替实际应用来评测 SpMV 的性能. 目前常见的数值解法器有 MATLAB[2], PETSc[3], Trilinos[4], hypre[5]等. 其中劳伦斯利弗莫尔国家实验室开发的高性能预处理器（hypre[5]）中，经测试 SpMV 占其中代数多重网格算法[6]运行总行时间的 89%. 由此可见，SpMV 的优化对数值解法器和实际应用都是十分必要的.

自 20 世纪 70 年代起，SpMV 的优化工作一直进行，包括存储格式的优化[7-12]和结合计算机体系结构的优化[13-18]等. 虽然许多优化方法显著提高了 SpMV 的性能，但在数值解法器和实际应用中这些优化工作并未加以很好利用. 由于压缩稀疏行（列）方法（即 CSR/CSC 格式）具有较高的压缩率且实现简单，该格式仍然是数值解法器中最通用的稀疏矩阵存储格式. 上面提到的 4 种数值解法器（MATLAB, PETSc, Trilinos 和 hypre）都使用 CSR 格式作为基本存储格式. 虽然 hypre 和 PETSc 采用分块 CSR 格式或分块的 DIA 格式作为扩展的存储格式，但相比于 SpMV 的众多优化方法，数值解法器中的 SpMV 并没有发挥出其应有的最优性能. 以 hypre 库为例，其中 SpMV 的性能与 Im 等人[14]优化的 SpMV 相比，性能差距达到 2 倍. 由此我们观察到以下现象：目前解法器和实际应用中调用的 SpMV 核心程序与 SpMV 程序本身的优化工作之间存在着不小的性能差距，该差距将对数值解法器和实际应用的性能造成极大影响.

我们分两方面详细说明产生这一差距的原因. 一方面，虽然 SpMV 的优化方法达数十种之多，但缺少通用的优化方法. 大多数优化方法针对特定的存储格式或一类具有相似特征的稀疏矩阵. 如分块 CSR 格式和分块 DIA 格式[15]针对具有稠密子块的矩阵. 当稀疏矩阵所含的稠密子块比例较大时，分块格式才会表现出明显的性能优势. 这种优化的存储格式的适用局限性使得不能将其作为解法器中的单一存储格式. 因为对于不具有稠密子块特征的稀疏矩阵，该存储格式会造成这些矩阵的性能下降. 因此，SpMV 优化方法的不通用性导致不能使用单一优化方法来提升数值解法器的性能.

另一方面，在真实应用和某些数值算法中，计算过程中稀疏矩阵并非固定不变. 如多重网格算法[6]中，当原始矩阵（第 0 层网格的矩阵算子）为五对角矩阵时，矩阵具有明显的对角线特征. 随着网格层次的增加，粗化算法使得矩阵算子的对角特征越来越弱. 在计算过程中矩阵特征的不固定性使得单一的优化方法不能满足整个数值算法和应用的需要.

综上，SpMV 优化方法的不通用性和计算过程中矩阵特征的不固定性，使得单一的优化方法并不能满足数值解法器对 SpMV 的调用要求，导致目前的数值解法器仍主要使用基本的 CSR 存储格式. 由于单一优化方法不能满足解法器和应用的需求，我们有必要对优化方法进行动态选择. 由于 SpMV 的优化方法以存储格式的优化为主，因此本文针对 SpMV 在不同存储格式之间进行动态选择.

我们构建了一个 SpMV 的自动调优器（SMAT），针对不同平台选择最优的存储格式，从而获得最优的 SpMV 实现供数值解法器调用. 本文的主要贡献如下：

- 我们使用佛罗里达大学的稀疏矩阵集[19]（共 2366 个矩阵）进行测试，总结了不同存储格式的特征并提取了稀疏矩阵的特征参数（第 2 节）.
- 我们构建了一个 SpMV 的自动调优器（SMAT），

根据不同的输入矩阵在不同平台上自动选择最优的矩阵存储格式. SMAT 中采用离线特征提取和在线搜索相结合的方法（第 3 节）.

- 在 Intel 上 SMAT 分别获得 9.11GFLOPS（单精度）和 2.44GFLOPS（双精度）的最高浮点性能，在 AMD 平台上获得了 3.36GFLOPS（单精度）和 1.52GFLOPS（双精度）的最高浮点性能. 相比 Intel 的 MKL 数学库[20]，SMAT 平均获得 1.4~1.5 倍的性能提升. 同时，SMAT 的预测时间约为 9~22 倍的 CSR-SpMV 执行时间，在调用上百次 SpMV 的应用中是可以接受的（第 4 节）.

1 背景

1.1 存储格式

为节省存储空间、减少矩阵的运算时间，稀疏矩阵通常采用压缩方法进行存储——即只存储矩阵中的非零元素，这种存储方法即为稀疏矩阵的存储格式. 存储格式发展于 20 世纪 70 年代，4 种基本存储格式被广泛接受：CSR[21], DIA[21], ELL[21], COO[21]. 根据稀疏矩阵的不同特征，基于这 4 种存储格式产生了新型存储格式，主要分为以下 3 类：分块格式（VBR[12], BCSR[15], BDIA[15]）、需要进行矩阵重排的格式（JAD[21], CSX[9]）、需要进行矩阵划分的格式（PKT[13], HYB[13], Cocktail[10]）等数十种存储格式. 本文选取 4 种基本存储格式的原因如下：

首先，这 4 种存储格式在数据结构上差异明显，适合不同特征的稀疏矩阵，并将差异性体现在 SpMV 的性能中. 其次，这 4 种格式均保留了稀疏矩阵的原始特征. 其他存储格式基于这 4 种基本格式并可能改变原始矩阵的特征. 分块格式的性能是以基本存储格式为基础，如 BCSR 和 BDIA；矩阵重排和矩阵划分改变了矩阵的特征，并且重排或划分后仍然使用现存的格式进行存储. 最后，基于这 4 种基本格式进行研究，为日后 SMAT 扩展到更多的存储格式打下良好基础.

因此，我们选择 DIA, ELL, CSR, COO 这 4 种稀疏矩阵的存储格式作为本文的关注点. 图 1 使用一个例子矩阵给出了 4 种格式的数据结构及其 SpMV 程序的基本实现. 从图 1 中，我们得到不同格式 SpMV 基本程序的特征对比，在表 1 中给出. 这些特征对 SpMV 的性能起着不同程度的影响. 存储格式对 SpMV 性能的影响将在第 3 节详细讨论.

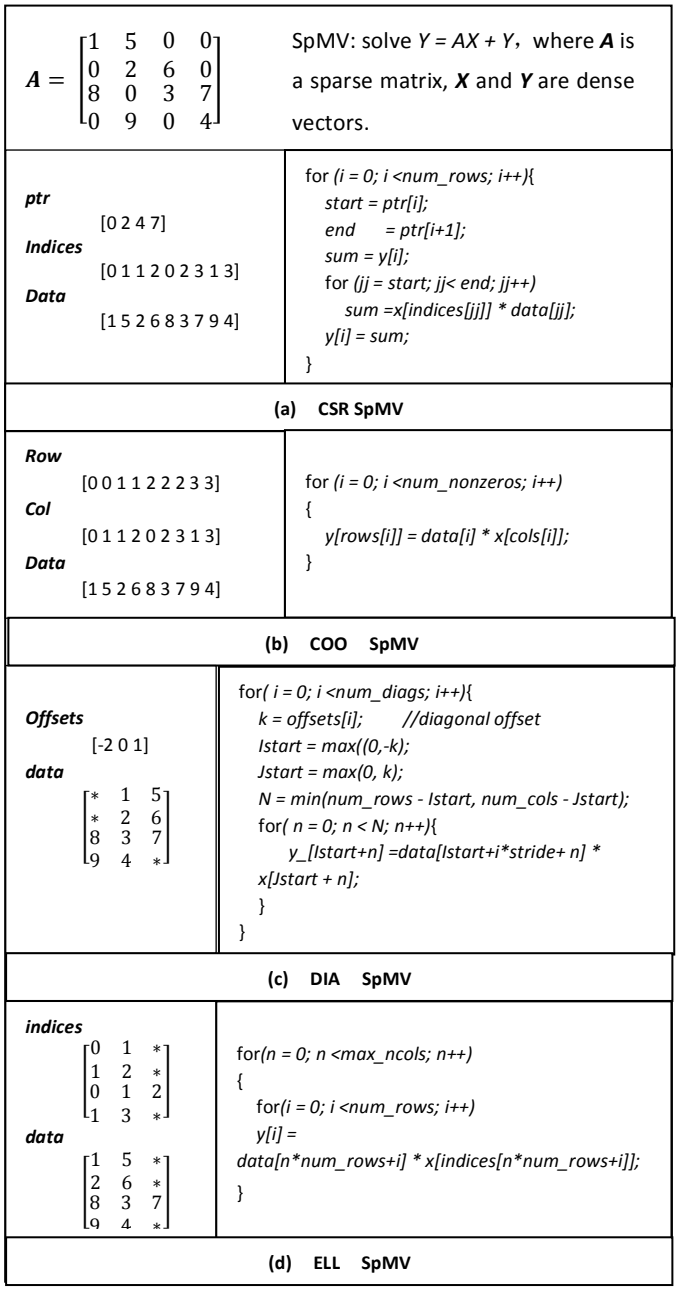


Fig.1 Data structure of four basic storage formats and their SpMV implementation.

图 1 4 种存储格式数据结构及其 SpMV 实现

Table 1 Comparison of SpMV Characteristics of Four Formats

表 1 4 种存储格式 SpMV 特征的对比

Storage Format	Characteristics of *X*Access	Length of Inner-loop	Extra Computing	Times of *Y* re-writing
DIA	Continuous	Mostly Equal	Decided by Nonzero Ratio	Number of Diagonals
ELL	Irregular	Equal	Decided by Nonzero Ratio	Maximum Number of Nonzeros per Row
CSR	Irregular	Inequal	No	1
COO	Irregular	--	No	1①

①Decided by architecture characterisitics. In cache architecture, times of writing vector ***Y*** in COO format is 1.

1.2 佛罗里达稀疏矩阵集

佛罗里达稀疏矩阵集[19]（简称 UF 矩阵集）从 1991 年开始，从实际应用中收集矩阵. 该矩阵集用来缩小计算科学家和稀疏矩阵算法开发者之间的距离. 有了这些真实应用中的矩阵，程序开发者可以利用它们真实反映算法在实际应用中将会取得的性能. 相比其他稀疏矩阵集，如 Matrix Market[22]和 Harwell-Boeing[23]集，UF 矩阵集包含了更多的应用且具有更大的矩阵规模（这两个矩阵集已包含在 UF 矩阵集中），而且此矩阵集仍在不断更新. 鉴于以上原因，我们选择 UF 矩阵集作为我们的测试集，更加真实地反映 SpMV 在真实应用中表现的性能.

UF 矩阵集有两种划分方法：1) 根据问题来源分为矩阵组（matrix group）；2) 根据应用范围分为矩阵类（matrix kind）. 我们关注矩阵类的划分，UF 矩阵集所含矩阵类在表 2 中给出. 按照矩阵所属领域，矩阵集分为 24 个矩阵类，即包含 24 个应用领域.

Table 2 The Distribution of Application Domain in UF Collection

表 2 UF 矩阵集中应用领域的分布

Application Domain	Number of Matrices
linear_programming	327
graph	323
structural	277
combinatorial	266
circuit_simulation	260
computational_fluid_dynamics	168
optimization	138
2D_3D	121
economic	71
model_reduction	70
chemical_process_simulation	64
power_network	61
theoretical_quantum_chemistry	47
electromagnetics	33
semiconductor_device	33
thermal	29
materials	26
least_squares	21
computer_graphics_vision	12
statistical_mathematical	10
counter-example	8
acoustics	7
biochemical_network	3
robotics	3

1.3 动机

不同存储格式在具有不同特征的稀疏矩阵上获得不同的性能. 我们以“Linear Programming”为例，对 4 种存储格式的 SpMV 进行性能测试，画出该矩阵类中矩阵在 4 种存储格式中的性能（图 2）. 其中 84%的矩阵在 CSR 格式取得最优性能，13%的矩阵在 COO 上取得最优性能，少数矩阵在 DIA 或 ELL 上取得最优性能. 如图 2 所示，当矩阵在 DIA 格式上取得最优性能时，其性能值远好于其他格式，因此属于同一应用领域的矩阵，其最优存储格式并不唯一，矩阵特征也不一致. 这使得在某个应用领域中仍难以简单地决定哪种格式是该领域的最优存储格式.

从图 2 表达的性能中，可知 DIA 和 ELL 格式的 SpMV 性能较高. 但并不是使用这两种格式存储的矩阵都能获得最优性能. 表 3 对 UF 矩阵集进行分类——适用矩阵集和最优矩阵集. 由于存储空间的限制，DIA 和 ELL 这两种格式都需要对矩阵进行补零操作. 我们对其零元素比例需要进行限制——一个矩阵的对角线条数或最大每行

非零元个数不能超过平均非零元个数的 20 倍，即非零元所占比例至少为存储数据的 5%. 因此 DIA 和 ELL 并不能适用于所有稀疏矩阵，我们将可使用某种存储格式（如 DIA）的矩阵集合称为“适用矩阵集”（如 *DIA_mats*），将具有同一最优格式的矩阵集合称为“最优矩阵集”（如 *good_DIA_mats*）. 每个矩阵有一个最优格式和多个可用格式.

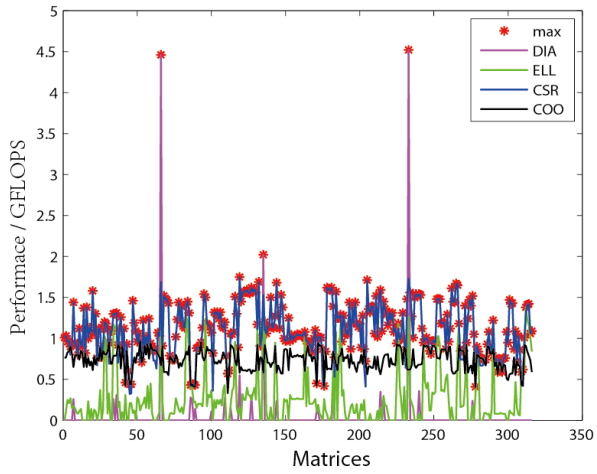


Fig.2 The ratio of matrices with the best storage format to all the matrices in the same matrix group.

图 2 最优存储格式的矩阵在各矩阵类所占比例

Table 3 Matrix Sets of the Four Storage Formats

表 3 4 种存储格式对应的矩阵集

Storage Formats	Suitable Matrix Set (Size of Matrix Set)	Best Matrix Set (Size of Matrix Set)
DIA	*DIA_mats* （458）	*good_DIA_mats* （206）
ELL	*ELL_mats* （1878）	*good_ELL_mats* (169)
CSR	*all_mats* （2366）	*good_CSR_mats* (1496)
COO	*all_mats* （2366）	*good_COO_mats* (507)

由于不同应用中稀疏矩阵特征的不一致性，详细的矩阵信息对选择存储格式从而选择有效的优化方法至关重要. 根据提取的矩阵特征，我们建立了 SpMV 自动调优器（SMAT），在不同体系结构上自动选择 SpMV 的最优存储格式. SMAT 在减少应用级程序员工作量的同时保证了 SpMV 的性能.

2 矩阵特征提取

下面我们对每个子矩阵集的特征进行分析. 为简便起见，我们用 *M, N* 表示矩阵的行数和列数，*NNZ* 表示矩阵的非零元个数. 本节中每个特征与 SpMV 性能的关系图均为对特征取值进行分段划分，观察每段区间上的矩阵分布. 下文图中使用“GOOD”表示某存储格式（如 DIA）为最优格式的矩阵所占比例，因此所有的 GOOD 矩阵集合即表 3 中的“*good_DIA_mats*”. “BAD”表示该格式未能获得最好性能的矩阵比例.“GOOD”和“BAD”矩阵的集合就是“*DIA_mats*”.

我们使用佛罗里达大学的稀疏矩阵集中的 2366 个矩阵进行不同格式的 SpMV 性能测试，通过测试统计出不同存储格式的特征.

2.1 DIA 格式

从表 1 中可以看出，对角线条数和非零元所占比例分别影响 SpMV 的额外计算量和写 ***Y*** 的次数，对 SpMV 性能造成影响. 我们用 *Ndiags* 和 *ER_DIA* 分别代表对角线条数和非零元所占比例，*ER_DIA* 的计算公式为式(1)

$$ER\_DIA = NNZ\ /(Ndiags \times M)\ . \quad (1)$$

我们对这两个参数在子矩阵集 *DIA_mats* 上测试其 SpMV 性能（图 3、图 4）.

1) 对角线条数（*Ndiags*）：DIA-SpMV 中写 ***Y*** 的次数有为 *Ndiags*，对角线条数越多，对向量 ***Y*** 的重复读写次数增加，对 SpMV 性能造成影响. 图 3 给出了 *Ndiags* 与 DIA-SpMV 性能的关系. 图 3 中横坐标为 *Ndiags* 的数目，分为 9 个取值区间；纵坐标为矩阵所占比例. 其中“GOOD”指 DIA 为最优格式的矩阵所占比例，可知所有的 GOOD 矩阵集合即表 3 中的

“*good_DIA_mats*”；而“BAD”指 DIA 未能获得最好性能的矩阵比例. 从图 3 中看出，当对角线条数大于 300 后，DIA 格式基本在绝大多数矩阵上不再获得最高性能.

结论 1. 当稀疏矩阵的对角线条数较少时，SpMV 使用 DIA 格式具有性能优势.

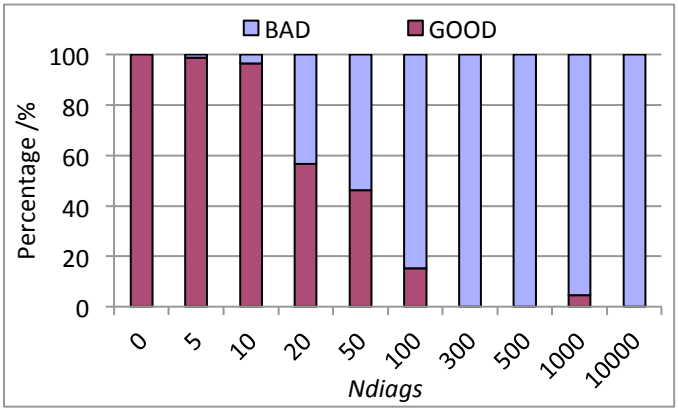


Fig.3 The influence of *Ndiags* on DIA-SpMV.

图 3 *Ndiags* 对 DIA-SpMV 性能的影响

2) DIA 格式中非零元所占比例（*ER_DIA*）：即使一条对角线上只有一个非零元，DIA 格式也需要存储整条对角线，包含存储额外的零元素. 大量的补零操作降低了非零元所占比例，增加了 SpMV 的额外计算，从而影响其性能. *ER_DIA* 与 DIA-SpMV 的性能图如图 4 所示. 可知，当矩阵中非零元所占比例过小（<20%），DIA 格式的 SpMV 不会取得较好的性能.

结论 2. 只有当非零元所占比例大于某一阈值（如 50%）时，DIA-SpMV 会取得 4 种格式中的最好性能.

Fig.4 The influence of *ER_DIA* on DIA-SpMV.

图 4 *ER_DIA* 对 DIA 格式 SpMV 性能的影响

从图 4 中我们看到，当 ER_DIA 取值在 30%~50%之间时，仍有一部分矩阵的最优格式为 DIA. 因此我们引进新的参数“*NTdiags_ratio*”来更好地提取矩阵的特征.

3) 真对角线所占比例（*NTdiags_ratio*）：我们将非零元比例大于一定值（如 50%）的对角线称为“真对角线”（*Tdiag*）. 在真对角线上使用 DIA 格式，SpMV 会取得很好的性能. *NTdiags_ratio* 用来表示真对角线在所有对角线中所占比例，计算公式如下

$$NTdiags\_ratio = 真对角线条数 / Ndiags. \quad (2)$$

从图 5 中得到如下结论：

结论 3. 当 *NTdiags_ratio* 较高（如> 40%）时，稀疏矩阵因使用 DIA 格式获得最优性能.

通过 3 个矩阵特征参数与 DIA-SpMV 的性能关系的测试，我们提取了表示 DIA 格式特征的 3 个特征参数（*Ndiags, ER_DIA, NTdiags_ratio*），并观察到当这 3 个阈值满足一定条件时，DIA 格式的 SpMV 会以较高的加速比获得最优性能.

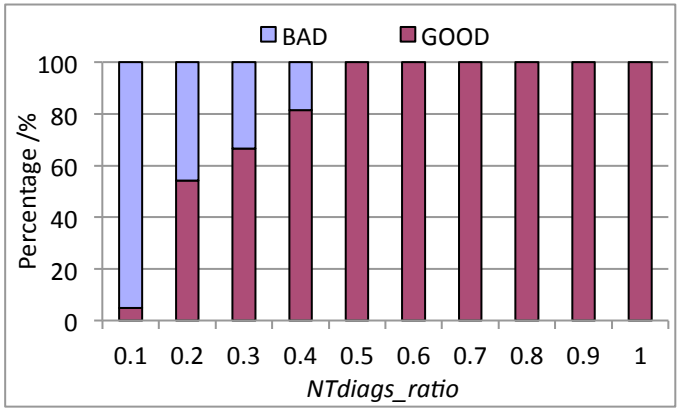


Fig.5 The influence of *NTdiags_ratio* on DIA-SpMV.

图 5 *NTdiags_ratio* 对 DIA 格式 SpMV 性能的影响

2.2 ELL 格式

我们使用类似的方法，对 ELL 矩阵进行分析. ELL 格式对于每行非零元个数不相等的情况同样需要补零操作. 为方便起见，我们将每行的非零元个数简称为“行度（*row_degree*）”. 从表 1 中可以看出，最大行非零元个数和非零元的填充率分别影响 SpMV 的额外计算量和

写 ***Y*** 的次数，因此对 SpMV 整体性能造成影响. 我们用 *max_RD* 和 *ER_ELL* 分别代表最大行非零元个数（即最大行度）和 ELL 格式中非零元所占比例，它们的计算公式为式(3)和式(4). 我们在 ELL 的适用子矩阵集 *ELL_mats* 上测试 ELL-SpMV 性能，继而分析这两个参数对 SpMV 的性能影响.

$$max\_RD = max_M\{row\_degree\}, \quad (3)$$

$$ER\_ELL = NNZ/(max\_RD \times M). \quad (4)$$

1) 矩阵最大行度（*max_RD*）：ELL-SpMV 中写向量 ***Y*** 的次数为 *max_RD*. 当最大行度增加时，对 ***Y*** 的重复写次数也增加，与 DIA 格式类似这同样会造成 ELL-SpMV 的性能降低. 图 6 中给出了 *max_RD* 和 ELL-SpMV 性能的关系. 从图 6 中得出如下结论：

结论 4. 只有最大行度较小时（如<10），ELL-SpMV 才可能取得最好性能.

Fig.6 The influence of *max_RD* on ELL-SpMV.

图 6 *max_RD* 对 ELL 格式 SpMV 性能的影响

2) ELL 格式中非零元所占比例（*ER_ELL*）：当一个稀疏矩阵每行非零元个数不一致时，ELL 以最大行非零元个数为依据，对其他行进行补零操作. 大量的补零操作给 SpMV 带来额外计算，从而影响其性能. *ER_ELL* 与 ELL-SpMV 的性能如图 7 所示. 我们得到如下结论：

结论 5. 只有当非零元所占比例大于某一阈值（如 90%）时，ELL-SpMV 会取得最好的性能.

与 *ER_DIA* 不同的是，*ER_ELL* 的阈值取值更大，这是由于 ELL-SpMV 程序的性能优势要小于 DIA-SpMV，因此对其参数取值要求也更加严格.

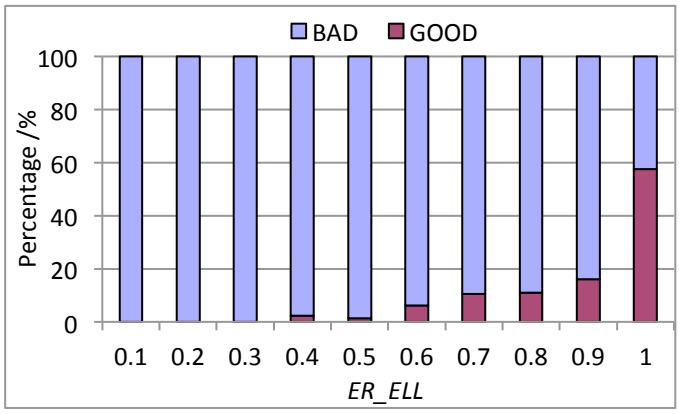


Fig.7 The influence of *ER_RD* on ELL-SpMV.

图 7 *ER_RD* 对 ELL 格式 SpMV 性能的影响\

在 ELL-SpMV 的测试过程中，我们同样观察到影响其性能的另一个因素——行度的波动. 当行度波动大时，非零元填充率相应减低，影响 SpMV 性能. 这个参数对于提取适合 ELL 格式的矩阵起到了辅助作用.

3) 行度波动值（*var_RD*）：我们用矩阵行度的方差来表示矩阵中行度的波动情况. 计算公式如下

$$var\_RD = \frac{1}{M}\sum_1^M |行度 - aver\_RD|. \quad (5)$$

var_RD 与 ELL-SpMV 性能的关系如图 8 所示. 从图 8 中得到如下结论：

结论 6. 当每行非零元个数波动很小时（如<1），ELL-SpMV 才可能取得较好的性能.

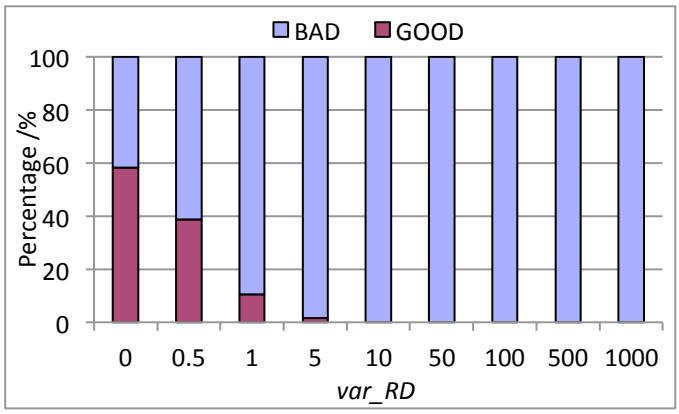


Fig.8 The influence of *var_RD* on ELL-SpMV

图 8 *var_RD* 对 ELL 格式 SpMV 性能的影响

通过 3 个矩阵特征参数与 ELL-SpMV 的性能关系的测试，我们提取了表示 ELL 格式特征的 3 个特征参数

（*max_RD, ER_ELL, var_RD*），并观察到当这 3 个阈值满足一定条件时，ELL 格式的 SpMV 就会获得最优性能，但加速比并不高.

2.3 COO 格式

以上 2 种格式存储稀疏矩阵时，都可能引入不必要的零元素填充，从而影响其 SpMV 性能. CSR 和 COO 格式则不存在这个问题，它们只存储矩阵中的非零元素，不会引入不必要的开销. 由上面对 DIA 和 ELL 的分析，我们得出了适合这两种存储格式的矩阵特征. 当一个稀疏矩阵不具有上述特征时，我们需要在 CSR 和 COO 两种格式中进行选择. 从 Yang[11] 中得知，在 GPU 上 COO-SpMV 在幂律矩阵[24]取得这 4 种格式的最优性能. 因此我们通过对幂律矩阵的测试在 CPU 上验证这一规律，并将这一规律作为 COO 的特征. 我们使用 SNAP[25] 中的部分幂律矩阵进行测试，38 个矩阵中有 20 个矩阵在 COO 格式下取得的最高性能. 可知如下结论：

结论 7. COO 格式在小世界网络矩阵中的优势在 CPU 上并没有 GPU 上体现的明显. 在 CPU 上并不是所有幂律矩阵都在 COO 格式上取得最优性能.

因此我们只使用幂律特征作为 CSR 和 COO 格式选择的一个参考标准.

4) 幂律特征值（*R*）：该参数从幂律分布[24]中得到，见式(6)

$$P(k) \sim k^{-R}, \tag{6}$$

其中 *k* 表示矩阵中不同行度值，*P(k)*表示该行度出现的频率. 我们对矩阵的分布用最小二乘法对该幂律公式中的 *R* 值进行计算. 为了确定 *R* 值的取值范围，我们对 *good_COO_mats* 中的矩阵进行测试，如图 9 所示. 由图 9 可知如下结论：

结论 8. 当 *R* 取值在[1,4][11, 24]区间内，COO-SpMV 会取得好于 CSR-SpMV 的性能.

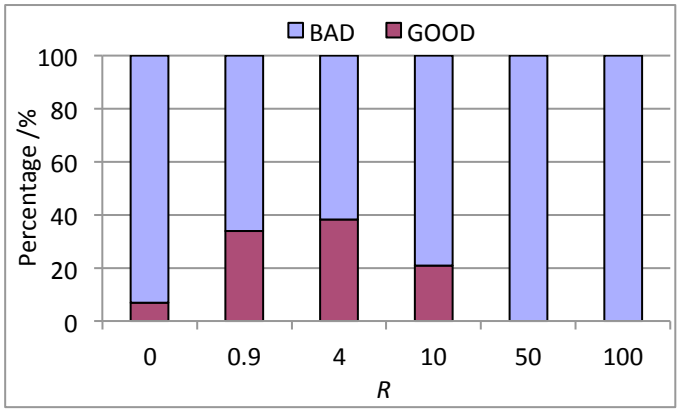


Fig.9 The influence of *R* on COO-SpMV

图 9 *R* 对 COO 格式 SpMV 性能的影响

我们通过对不同矩阵子集性能的测试，得到了影响不同存储格式的特征参数（见表 4 第 1 列），并且得到了相应结论. 从这些结论中，我们得到了每个特征参数的阈值. 参数的阈值也在表 4 中用符号进行表示，阈值的取值将在第 3 节中介绍.

Table 4 Characteristics Parameter List of Sparse Matrices, Their Meanings and the Symbols of Thresholds

表 4 稀疏矩阵的特征参数和含义及其对应的阈值表示

Parameter	Threshold	Meaning
M	-	# of Rows
N	-	# of Columns
NNZ	-	# of Nonzeros
Ndiags	*Ndiags_STH*	# of Diagonals
	Ndiags_LTH	
NTdiags_ratio	*NTdiags_ratio_STH*	Ratio of "True Diagonals"
	NTdiags_ratio_LTH	
ER_DIA	*ER_DIA_STH*	Nonzero Ratio of DIA Format
	ER_DIA_LTH	
max_RD	*max_RD_STH*	Maximum Value of "*row_degree*"
	max_RD_LTH	
var_RD	*var_RD_STH*	Variaton of "*row_degree*"
	var_RD_LTH	
ER_ELL	*ER_ELL_STH*	Nonzero Ratio of ELL Format
	ER_ELL_LTH	
R	*R_STH*	Chracteristics Value of Power Law
	R_LTH	
Best format	-	The Best Storage Format

3 SMAT 自动调优器

基于这 4 种基本存储格式，我们建立了 SpMV 自动调优器（SMAT），用来选择最优的存储格式，从而使 SpMV 获得最高性能. SMAT 架构如图 10 所示. SMAT 架构的输入为 CSR 格式存储的稀疏矩阵，输出该矩阵的最优格式. 该架构分为两大部分：离线部分和在线部分. 离线部分通过对稀疏矩阵训练集进行测试，在不同体系结构上通过在测试矩阵集上运行多种 SpMV 算法，确定特征参数阈值. 在线部分通过矩阵特征的实时提取来进行格式预测和运行反馈，从而得到输入矩阵的最优存储格式. 下面我们详细介绍每部分的构成.

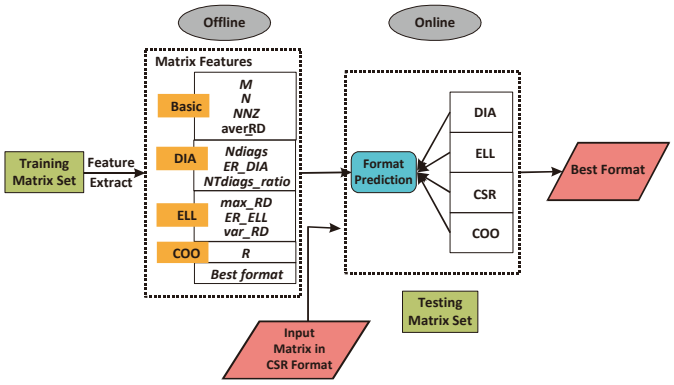


Fig.10 SMAT architecture.

图 10 SMAT 架构

3.1 离线部分

我们从 UF 矩阵集中选出 2055 个矩阵进行离线部分中每种存储格式的性能测试. 类似于第 3 节的分析，我们在表 3 中的“适用矩阵集”上测试这 4 种格式的 SpMV 性能，从而通过统计得到具有平台差异性的参数阈值.

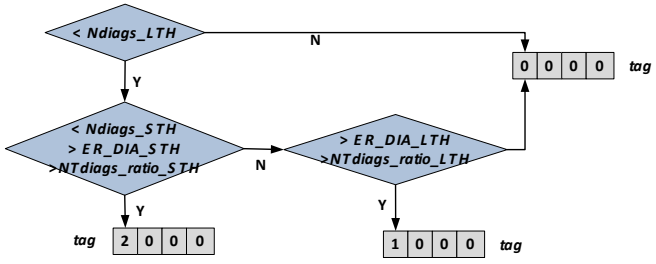
如表 4 所示，每个参数对应两个阈值，一个称为“紧阈值”. 当该参数值满足该阈值时，“适用矩阵集”中 90%的矩阵可以在某一存储格式上取得最优性能. 以 DIA 格式中的 *Ndiags* 参数为例，我们使用 *DIA_mats* 测试 DIA-SpMV 的性能. 当 *Ndiags*<20 时，90%以上的矩阵在 DIA 格式下取得最优性能. 因此 *Ndiags* 的紧阈值 *Ndiags_STH*=20. 另一个阈值为“松阈值”. 在“最优矩阵集”中，当多于 90%的矩阵该参数取值分布于大于（或小于）某一数值的范围内，我们将该数值称为松阈值. 如 DIA 的“最优矩阵集（*good_DIA_mats*）”中 90%以上的矩阵分布在 *Ndiags*<100 的范围内，因此 *Ndiags_LTH*=100. 因此，紧阈值确保了预测的正确性，而松阈值则确保了预测的少遗漏性.

这样，在 SMAT 的离线过程，我们确定了每个参数的阈值大小，从而确定了在线过程中的存储格式预测模型.

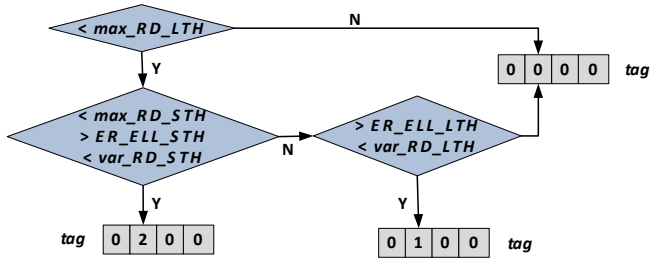
3.2 在线部分

在线部分对输入的 CSR 格式稀疏矩阵进行实时处理，通过提取其矩阵特征，得到其最优存储格式.

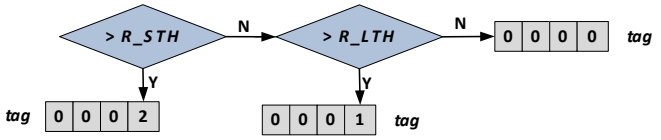
在该部分，SMAT 首先对输入的 CSR 格式矩阵进行特征提取，从而得到表 4 中每个特征参数的取值. 由于离线部分已经确定了每个参数的阈值，我们使用一个可信度数组(tag)来标识每个格式的预测情况（如图 11 所示）. 数组中的每一位对应一个存储格式，从左到右依次对应 DIA, ELL, CSR, COO 的标签. 当矩阵特征参数全部满足某一存储格式对应的紧阈值时，数组的相应位置的标签为 2，即预测该矩阵使用该存储格式*很大可能*会取得最优性能. 当矩阵特征参数不能全部满足紧阈值，而满足全部松阈值对应的条件时，数组的对应位置的标签置为 1，即该存储格式*可能*成为该矩阵的最优格式. 当特征参数不能满足全部的松阈值时，数组对应位置 0. 通过该数组的值，SMAT 的在线过程可以通过这些阈值的比较，推测该稀疏矩阵适合的存储格式. 图 11 给出了 SMAT 的在线流程图。



(a) DIA prediction procedure



(b) ELL prediction procedure



(c) COO prediction procedure

Fig.11 The prediction procedure of each format and the confirmation of confidence array value.

图 11 每种格式的判断流程图及预测可信度数组值的设定

由于提取一个矩阵的全部特征需要的时间较长，并且在进行每个存储格式的预测时，并不需要全部的特征参数，因此 SMAT 将矩阵的特征提取过程和存储格式预测两部分混合在一起. 先对矩阵进行部分特征的提取，再根据格式预测的结果决定是否对下一个存储格式进行特征提取. 举例来说，当图 11 中的可信度数组中 DIA 格式为 2，则认为找到了最优存储格式为 DIA 而停止下一个存储格式的特征提取和格式预测. 这样，在不影响 SMAT 预测准确性的前提下，节省了矩阵特征提取过程不必要的时间开销. 从 SMAT 的在线流程图（图 12）中的分支流程可以看出提取特征和格式预测的交替过程.

为了进一步确保 SMAT 预测的准确度，当 tag 数组的值都不为 2 时，我们使用运行反馈模块确保其准确性. 当可信度数组中没有标签值为 2，SMAT 使用实际运行并对比测试性能的做法得到最优存储格式. 对于数组中标签为 1 的位执行一次对应格式的 SpMV，并记录其性能. 根据性能记录，从中选择性能最大值，其对应的存储格式则为最优存储格式. 图 12 中的运行部分即为运行反馈模块. 运行及反馈使得对特征不明显的稀疏矩阵也可以准确地选择存储格式，提高了 SMAT 的预测准确性.

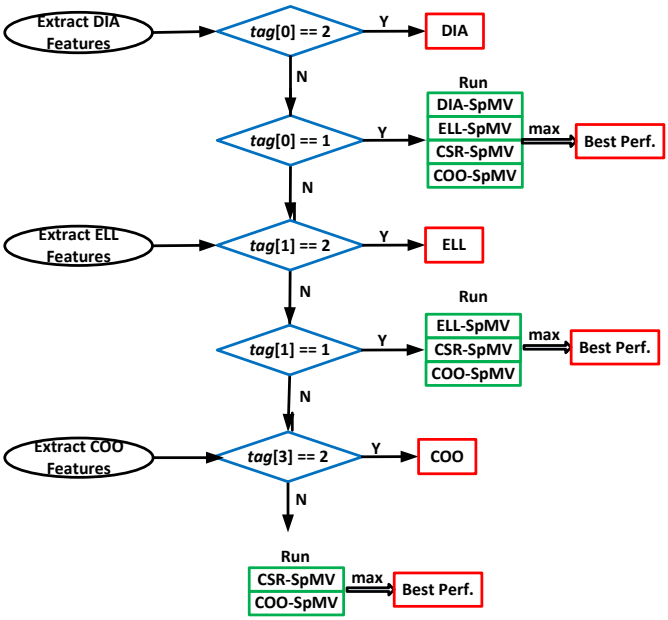


Fig.12 The online prediction procedure of SMAT.

图 12 SMAT 在线预测流程图

4 性能及分析

4.1 实验平台及数据集

我们选择 Intel 和 AMD 两个平台测试 SMAT 性能并进行分析. Intel 平台采用 Intel Xeon X5680 为 CPU，主频为 3.33 GHz. 该系统配置了 24 GB 的内存和可用磁盘容量为 384 GB. AMD 系统包含主频为 1.9 GHz 的 Opteron CPU 和 16 GB 的内存，可用磁盘容量为 130 GB.

本文我们对佛罗里达大学稀疏矩阵集进行划分为训练集和测试集. 在 Intel 平台上我们测试了将该矩阵集中的 2055 个矩阵作为训练集，311 个矩阵作为测试矩阵. 由于 AMD 平台磁盘容量的限制，我们只能测试佛罗里达矩阵集中的 1664 个矩阵，其中 1438 个矩阵组成训练集，226 个矩阵作为测试矩阵. 本节给出 SMAT 在测试矩

阵上的性能并评价了其预测准确率.

4.2 性能

我们在 Intel 和 AMD 平台上分别使用测试集矩阵测试了 SMAT 的单精度和双精度浮点性能，如图 13, 14 所示. 从图 13, 14 中可知，在单核 CPU 上，SMAT 在 Intel 平台上分别获得了高达 9.11GFLOPS（浮点效率为 68%）和 2.44GFLOPS(效率 18%)的单精度和双精度浮点性能；在 AMD 平台上分别获得了 3.36GFLOPS（效率 44%）和 1.52GFLOPS（效率 20%）的单精度和双精度浮点性能.

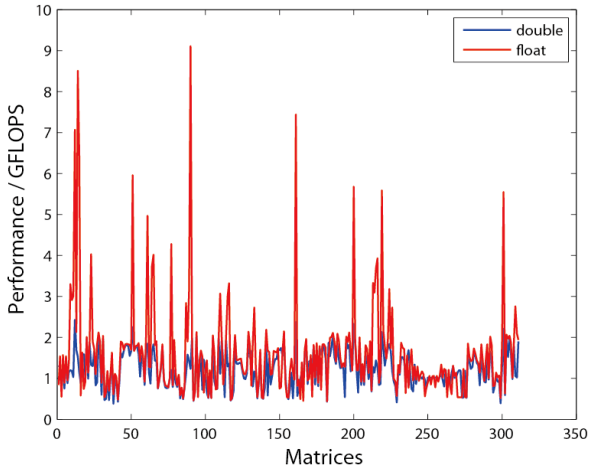


Fig.13 The performance of SMAT on Intel platform in single- and double-precision.

图 13 SMAT 在 Intel 平台上的单精度和双精度浮点性能

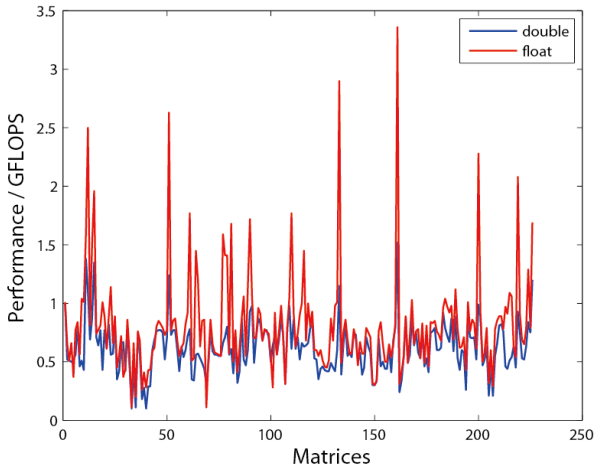
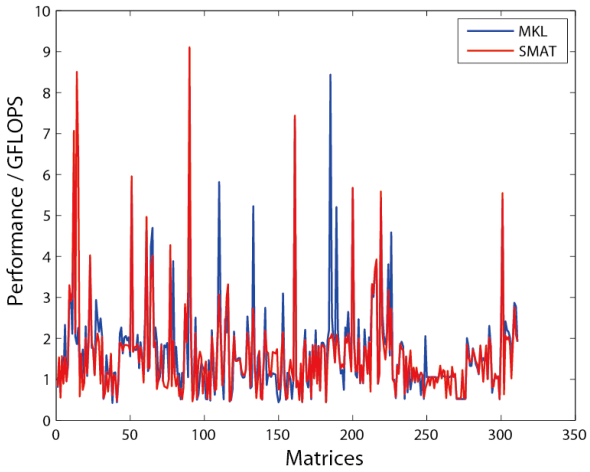


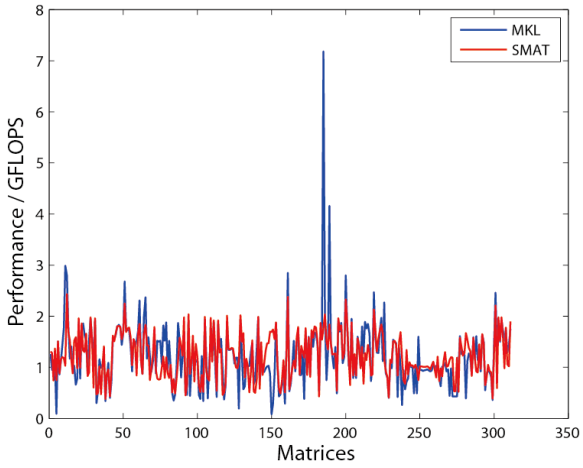
Fig.14 The performance of SMAT on AMD platform in single- and double-precision.

图 14 SMAT 在 AMD 平台上的单精度和双精度浮点性能

我们在图 15 对 SMAT 与 Intel 的 MKL 数学库进行性能对比. 根据实验得知，单精度时 SMAT 在 71%的测试矩阵上获得高于 MKL 的性能，其平均加速比达到 1.4 倍. 双精度时 SMAT 在 76%的测试矩阵上取得高于 MKL 的性能，其平均加速比为 1.5 倍. 但在某些矩阵上 MKL 仍获得较高性能，如“Meszaros/air03”上，MKL 在单精度和双精度上都获得了高于 SMAT 的性能.



(a) Single-precision



(b) Double-precision

Fig.15 Performance Comparison of SMAT and Intel MKL library.

图 15 SMAT 与 Intel MKL 数学库的性能对比

4.3 实验分析

4.3.1 准确率

我们首先分析 SMAT 的预测准确率. 通过测试，SMAT 在 Intel 平台的预测准确率为 89.34% （单精度）和 86.18%（双精度），在 AMD 平台上的准确率为 85.10%（单精度）和 82.09%（双精度）. 由于不同体系结构上预测模型的不同（即参数阈值不同），两个平台上的预测准确率也存在差别. 下面我们详细分析 SMAT 在每种存储格式的预测准确率如图 16 所示. 从图 16 中可知，SMAT 在 DIA 和 ELL 这两种格式上获得最高的预测准确率，分别为 97%和 91%，高于 SMAT 的总体准确率. 而 COO 的预测准确率较低，因此 SMAT 的整体准确率受其影响而没有达到 DIA 和 ELL 的准确度.

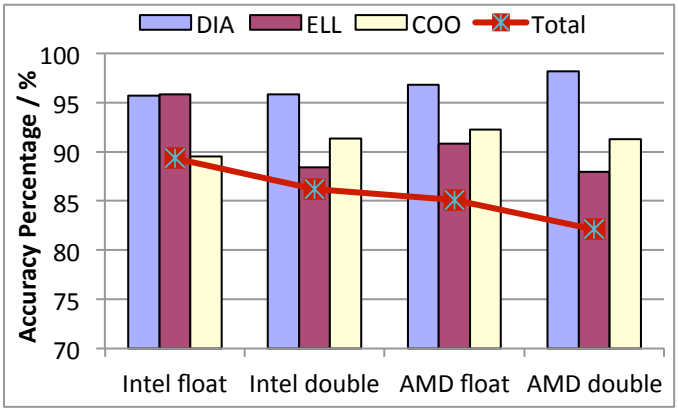


Fig.16 Prediction accuracy of the four formats.

图 16 4 种存储格式的预测准确率

4.3.2 决策时间

我们将 SMAT 中选择最优格式部分（在线矩阵特征提取、格式预测和实际运行）的时间作为时间开销. 我们测试了在线格式选择过程的执行时间，用 CSR-SpMV 执行时间的倍数来表示（表 5）. SMAT 的在线搜索时间约为 CSR-SpMV 的 9~22 倍. 对于数值代数库，很多情况下调用 SpMV 的次数为上百次（如 Jacobi 迭代和 Gauss-Seidel 迭代等），因此我们认为这些开销是可以接受的. 与蛮力搜索相比，SMAT 使用预测方法节省了不必要的时间开销和格式转换开销.

Table 5 Format Choosing Time (Use Times of CSR-SpMV to Represent)

表 5 格式选择时间（使用 CSR-SpMV 的倍数来表示）

Search Method	Intel Float	Intel Double	AMD Float	AMD Double
SMAT Prediction	16	22	9.7	9.6
“Brute-Force” Search	1868	2075	2404	2522

5 相关工作

从 20 世纪 70 年代至今，很多 SpMV 的优化工作提升了其性能. 这些优化工作大致为两类：一类是开发新的存储格式；另一类针对体系结构特征进行优化. 存储格式的优化目的在于降低访存从而提升 SpMV 性能. Im 等人[14]创建了 BCSR 格式来更好地开发稀疏矩阵中稠密子块的性能. Vuduc 等人[12]将 BCSR 格式优化为 VBR 格式，更好地开发了不同大小稠密子块的性能. Kourtis 等人[9]提出 CSX 存储格式来压缩索引数组从而开发稀疏矩阵中的子结构. 另一方面，一些工作结合体系结构特征对 SpMV 进行优化，尤其针对新型体系结构如多核 CPU、FPGA 和众核 GPU. Nagar 等人[17]在 Convey HC-1，一个由基于 Xeon 的 CPU 和基于 FPGA 的协处理器组成的混合系统，对 SpMV 进行优化. Williams 等人[16]在 5 个平台（AMD Opteron, Intel Clovertown, Sun Niagara2, STI Cell SPE）上实现并评价了不同的 SpMV 优化策略. Bell 和 Garland[13]在 NVIDIA GPU 上优化了几种不同存储格式的 SpMV，并提出了新的存储格式（HYB）. 这些 SpMV 优化工作在不同程度上提高了 SpMV 的性能.

在 SpMV 的优化工作中，一些工作利用自动调优技术提升性能的同时也提高不同平台之间的可移植性. Vuduc 等人[15]建立了 OSKI 自动调优器来调节 BCSR 和 VBR 存储格式下的分块大小. Williams 等人[16]使用自动调优技术结合层次化策略来选择最优的参数组合. Choi 等人[8]在 NVIDIA GPU 上实现了 BCSR 和 SBELL 格式，并

使用自动调优技术确定分块大小. Yang 等人[11]提出了混合存储格式，结合模型自动调节每种存储格式在矩阵中所占的比例. 以上工作都是使用自动调优方法结合体系结构特征来确定的一个存储格式的参数. 本文提出的 SMAT 是对多种存储格式进行选择，借助自动调优技术在不同存储格式中选择最优格式，从而增加了 SMAT 的扩展性和适用性.

6 结语

本文通过观察 4 种存储格式(DIA, ELL, CSR, COO)的 SpMV 性能得到一些结论，从而得出了影响矩阵性能的参数集. 我们使用该参数集提取矩阵特征，并结合观察得到的结论，构建了 SpMV 自动调优器（SMAT）. 根据输入的稀疏矩阵，SMAT 可以在不同平台上结合其特征选择并输出最优存储格式，供数值解法器和上层应用调用. SMAT 不仅提高了数值解法器和应用的性能，同时使得已有的 SpMV 优化技术可以更好的应用到实际中. 使用 UF 稀疏矩阵集中的部分矩阵进行测试，SMAT 在 Intel 平台上获得了 9.11 GFLOPS (单精度)和 2.44 GFLOPS (双精度)的性能，AMD 平台上获得了 3.36 GFLOPS (单精度)和 1.52 GFLOPS (双精度)的性能. 相比 Intel 的 MKL 数学库，SMAT 具有 1.5 倍左右的加速比. 虽然 SMAT 的在线格式预测需要 9~22 倍的 CSR-SpMV 的时间，在同一矩阵循环调用 SpMV 上百次时，SMAT 的搜索开销是可以接受的. 将来我们会把更多的存储格式加入 SMAT 中，并将 SMAT 集成到数值解法器中.

参考文献

[1] Kelley C T. Iterative Methods for Linear and Nonlinear Equations [M]. Philadelphia, PA: SIAM, 1995

[2] The MathWorks Inc. MATLAB and statistics toolbox release 2012b [OL]. 2012. [2013-03-26]. http://www.mathworks.com/products/statistics/

[3] Argonne National Laboratory. PETSc: Portable, extensible toolkit for scientific computation [OL]. 2013. [2013-03-26]. http://www.mcs.anl.gov/petsc/

[4] Sandia National Laboratories. The Trilinos project [OL]. 2013. [2013-03-26]. http://trilinos.sandia.gov

[5] Falgout R D, Yang U M. Hypre: A library of high performance preconditioners [C] //Proc of the Int Conf on Computational Science. New York: ACM, 2002: 632-641

[6] Falgout R D. An introduction to algebraic multigrid [J]. Computing in Science and Engineering, 2006, 8(6): 24-33

[7] Buluc A, Williams S, Oliker L, et al. Reduced-bandwidth multithreaded algorithms for sparse matrix-vector multiplication [C] //Proc of the 2011 IEEE Int Parallel & Distributed Processing Symp. Piscataway, NJ: IEEE, 2011: 721-733

[8] Choi J W, Singh A, Vuduc R W. Model-driven autotuning of sparse matrix-vector multiply on gpus [C] //Proc of the 15th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2010: 115-126

[9] Kourtis K, Karakasis V, Goumas G, et al. CSX: An extended compression format for spmv on shared memory systems [C] //Proc of the 16th ACM Symp on Principles and Practice of Parallel Programming. New York: ACM, 2011: 247-256

[10] Su B Y, Keutzer K. clSpMV: A cross-platform OpenCL SpMV framework on GPUs [C] //Proc of the 26th ACM Int Conf on Supercomputing. New York: ACM, 2012: 353-364

[11] Yang X, Parthasarathy S, Sadayappan P. Fast sparse matrix-vector multiplication on GPUs: Implications for graph mining [J]. Proc of the VLDB Endowment, 2011, 4(4): 231-242

[12] Vuduc R W, Moon H J. Fast sparse matrix-vector multiplication by exploiting variable block structure [C] //Proc of the 1st Int Conf on High Performance Computing and Communications. Berlin: Springer, 2005: 807–816

[13] Bell N, Garland M. Implementing sparse matrix-vector multiplication on throughput-oriented processors [C] // Proc of the Conf on High Performance Computing Networking, Storage and Analysis. New York: ACM, 2008

[14] Im E J, Yelick K, Vuduc R. Sparsity: Optimization framework for sparse matrix kernels [J]. Int Journal of High Performance Computing Applications, 2004, 18(1): 135-158

[15] Vuduc R, Demmel J W, Yelick K A. OSKI: A library of automatically tuned sparse matrix kernels [J]. Journal of Physics: Conf Series, 2005, 16(1): 521–530

[16] Williams S, Oliker L, Vuduc R, et al. Optimization of sparse matrix-vector multiplication on emerging multicore platforms [C] // Proc of the 2007 ACM/IEEE Conf on Supercomputing. New York: ACM, 2007

[17] Nagar K, Bakos J. A sparse matrix personality for the Convey HC-1 [C] //Proc of 2011 IEEE 19th Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2011: 1–8

[18] Song Qingzeng, Gu Junhua. FPGA design and implementation of sparse matrix vector multiply [J]. Computer Engineering, 2011, 37(23): 214-216 (in Chinese)（宋庆增，顾军华. 稀疏矩阵向量乘的 FPGA 设计与实现[J]. 计算机工程, 2011, 37(23): 214-216）

[19] Davis T A, Hu Y. The university of Florida sparse matrix collection [J]. ACM Trans on Mathematical Software(TOMS), 2011, 38(1):1-25

[20] Intel Corp. Math kernel library [OL]. 2013. [2012-03-26]. http://software.intel.com/en-us/intel-mkl

[21] Saad Y. Sparskit: A basic tool kit for sparse matrix computations - Version 2 [R]. Twin Cities, MN: Twin Cities - University of Minnesota, 1994

[22] Boisvert R F, Pozo R, Remington K, et al. Matrix market: A web resource for test matrix collections [C] //Proc of the IFIP TC2/WG2.5 Working Conf on Quality of Numerical Software: Assessment and Enhancement, London: Chapman & Hall, 1997: 125–137

[23] Harwell Laboratory. HBSMC: Harwell Boeing sparse matrix collection [OL]. 1992. [2013-3-26]. http://people.sc.fsu.edu/~jburkardt/datasets/hbsmc/hbsmc.html

[24] Aiello W, Chung F, Lu L. A random graph model for power law graphs [J]. Experimental Mathematics, 2000, 10(1):53–66

[25] Stanford University. SNAP: Stanford large network dataset collection [OL]. 2013. [2012-03-26]. http://snap.stanford.edu/data/

Li Jiajia, born in 1988.PhD candidate.Student member of China Computer Federation.Her main research interests include high performance computing, parallel computing and auto-tuning method.

Zhang Xiuxia, born in 1987.PhD candidate.Student member of China Computer Federation.Her main research interests include high performance computing and parallel computing.

Tan Guangming, born in 1980.Associate Professor and PhD supervisor.Member of China Computer Federation.His main research interests include Parallel algorithms and programming on Multi/Many-core architectures.

Chen Mingyu, born in 1972.Professor and PhD supervisor.Member of China Computer Federation. His main research interests include architecture, operating System and algorithm optimization for high performance computers.

文章校对负责人：李佳佳
电话：010-62601041
手机：13426202481
E-mail：lijiajia@ict.ac.cn