CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 46 卷第 7 期 2024 年 7 月
Vol. 46, No. 7, Jul. 2024

文章编号:1007-130X(2024)07-1158-09

MiniBranRAP:极小化分支判断数的AMG粗网格矩阵计算并行算法*

杜　皓[1,2],毛润彰[1,2],邓蕴桐[1,2],黄思路[2],徐小文[2]

(1.中国工程物理研究院研究生院,北京 100094;2.北京应用物理与计算数学研究所,北京 100088)

摘　要:代数多重网格(AMG)是科学工程计算与工业仿真领域求解大规模稀疏线性代数方程组最常用的算法之一。在启动(Setup)阶段的每个网格层,AMG 需要基于限制算子 $\boldsymbol{R}$、当前细网格层矩阵 $\boldsymbol{A}$ 和插值算子 $\boldsymbol{P}$ 的稀疏矩阵乘积来计算粗网格矩阵 $\boldsymbol{A}_c=\boldsymbol{RAP}$,该过程是 AMG 并行性能的主要瓶颈。首先发现了主流 AMG 解法器中 RAP 并行算法由于分支判断的平方复杂度导致的性能瓶颈,并结合稀疏矩阵 CSR 的行主序特点,提出了具有线性复杂度分支判断数的 RAP 并行算法 MiniBranRAP。该算法集成到 JXPAMG 解法器中,并通过实际应用算例验证了算法的有效性。测试结果表明,对于 6 个来自实际应用的典型算例,相对于 Hypre 最新版本的 BoomerAMG 解法器,基于 MiniBranRAP 的 JXPAMG 解法器在 28 个进程上将 Setup 阶段的计算效率平均加速 3.3 倍、最高加速 9.3 倍。

关键词:代数多重网格(AMG);粗网格矩阵计算;分支判断;Hypre;JXPAMG

中图分类号:TP301.6　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2024.07.003

MiniBranRAP:A minimizing branch parallel algorithm of the coarse matrix computation in AMG solver

DU Hao[1,2], MAO Run-zhang[1,2], DENG Yun-tong[1,2], HUANG Si-lu[2], XU Xiao-wen[2]

(1. Graduate School of China Academy of Engineering Physics, Beijing 100094;
2. Institute of Applied Physics and Computational Mathematics, Beijing 100088, China)

Abstract: Algebraic multi-grid (AMG) is one of the most commonly used algorithms for solving large-scale sparse linear algebra equations in the field of scientific engineering computing and industrial simulation. For each grid layer in the Setup phase, AMG needs to calculate the coarse grid matrix $\boldsymbol{A}_c=\boldsymbol{RAP}$ through the product of three sparse matrix based on the restriction operator $\boldsymbol{R}$, the current fine grid matrix $\boldsymbol{A}$, and the interpolation operator $\boldsymbol{P}$, which has become the main bottleneck in the parallel performance of AMG. This paper first discovers that the performance bottleneck of the RAP parallel algorithm in mainstream AMG solvers is caused by the quadratic complexity of branch judgments. Then, utilize the row-based order characteristics of the sparse matrix format CSR, and propose a RAP parallel algorithm called MiniBranRAP with linear complexity of branch judgment counts. The algorithm is integrated into the JXPAMG solver, and the effectiveness of the algorithm is verified through practical examples. The numerical test results show that, for 6 typical examples from practical applications, compared with the latest version of Hypre's BoomerAMG solver, the JXPAMG solver based on MiniBranRAP can speed up the computation efficiency of the Setup phase by an average of 3.3 times and a maximum of 9.3 times on 28 processors.

Key words: algebraic multi-grid (AMG); coarse grid matrix computation; branch; Hypre; JXPAMG

* **收稿日期**:2023-11-07;**修回日期**:2023-12-21
基金项目:国家自然科学基金(62032023)
通信作者:徐小文(xwxu@iapcm.ac.cn)
通信地址:100088 北京市北京应用物理与计算数学研究所
Address: Institute of Applied Physics and Computational Mathematics, Beijing 100088, P. R. China

1　引言

对于大规模稀疏线性代数方程组的求解，代数多重网格 AMG（Algebraic Multi-Grid）是一种具有最优复杂性的算法，即计算量与问题规模为线性关系。其算法思路可以概括为一句话：在当前网格层磨光，在粗网格层校正。磨光步消去误差的高频部分，校正步利用粗网格求得误差的修正解，两者互补实现迭代误差的快速下降[1-3]。与基于细网格的几何信息逐层粗化得到粗网格的几何多重网格 GMG（Geometric Multi-Grid）不同，AMG 算法的网格粗化等算法组件是通过代数信息如矩阵元素大小构建的，不依赖于几何信息。因此，AMG 算法在保留了 GMG 算法的最优复杂性的同时，算法健壮性也得到明显提升，可以有效地处理很多具有复杂区域、非结构网格和非光滑系数等特征的问题，通常作为预条件技术加速 Krylov 子空间迭代法[4]。目前，AMG 算法已广泛应用于辐射流体力学[3]、结构力学[5]、工业数值模拟[6]和量子色动力学[7]等具有复杂特征的领域，解决了许多具有挑战性的实际应用问题[8]。

尽管如此，AMG 算法的并行可扩展性差是一个公认的问题，其中启动（Setup）阶段的粗网格矩阵计算是一个主要的瓶颈[3,9-12]。在 Setup 阶段的每个网格层，AMG 算法采用 Galerkin 方法通过限制算子 $\boldsymbol{R}$、当前细网格矩阵 $\boldsymbol{A}$、插值算子 $\boldsymbol{P}$ 3 个稀疏矩阵的乘积来计算粗网格矩阵 $\boldsymbol{A}_c=\boldsymbol{RAP}$，该过程称为 RAP 运算。通常，RAP 通过 2 次稀疏矩阵乘积运算 SpGEMM 实现[13]，因此 SpGEMM 运算中存在的问题，包括结果稀疏矩阵中非零元的数量事先未知、稀疏矩阵插入操作开销大等[14]，在 RAP 的实现中也存在。此外，RAP 的并行实现还涉及由于矩阵合并导致的分支判断操作。

在实际应用中发现一个关于 RAP 并行算法中分支判断的性能瓶颈问题：相对于串行计算情形，由于分支判断的影响，RAP 的并行计算时间开销急剧增加，成为瓶颈。通过分析发现，在包括 Hypre 在内的现有主流 AMG 解法器中，RAP 并行算法的矩阵合并操作存在较多分支判断，且复杂度与矩阵规模的平方呈正比，成为导致瓶颈的主要原因。针对这个瓶颈问题，本文提出一种极小化分支判断数的 RAP 算法，称之为 MiniBranRAP。（Minimized Branch for RAP）。该算法利用稀疏矩阵 CSR（Compressed Sparse Row）格式的行主序性质，通过引入标记数组，将分支判断数从平方复杂度降为线性复杂度，大幅度减少了 RAP 组件在并行计算情形中的分支判断数。来源于辐射流体力学、结构力学和航空发动机等实际应用中的 6 个典型算例的数值结果表明，MiniBranRAP 算法可以显著降低 RAP 组件的并行计算时间，相比于当前主流的 AMG 解法器 Hypre，MiniBranRAP 为 Setup 阶段带来了平均 3.3 倍、最高 9.3 倍的加速效果。

本文组织结构如下：第 2 节简要介绍 AMG 算法，第 3 节介绍 RAP 并行算法及瓶颈问题，第 4 节介绍本文提出的极小化分支判断数的 RAP 并行算法，第 5 节给出数值实验结果，第 6 节总结全文。

2　AMG 算法

对于大规模稀疏线性代数方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，$\boldsymbol{A}$ 表示稀疏系数矩阵，$\boldsymbol{x}$ 表示未知向量，$\boldsymbol{b}$ 表示右端向量。记最细层网格 $\Omega=\{\boldsymbol{x}_0,\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n\}$ 为未知向量元素的集合，以 2 层网格为例，经典 AMG 算法可以描述如算法 1[1,8,11]所示。

算法 1　AMG 算法（2 层网格情形）

Step 1　启动阶段（Setup）：构建多重网格结构。

Step 1.1　粗化：选取粗网格点 Ω_c 包含于 Ω；

Step 1.2　构建插值算子和限制算子：$\boldsymbol{P}:\Omega_c\rightarrow\Omega$，$\boldsymbol{R}:\Omega\rightarrow\Omega_c$，一般选取 $\boldsymbol{R}=\boldsymbol{P}^{\mathrm{T}}$；

Step 1.3　构造粗网格矩阵：$\boldsymbol{A}_c=\boldsymbol{RAP}$（Galerkin 方法）。

Step 2　求解阶段（Solve）：重复执行下面的循环算法直至达到收敛标准：

Step 2.1　前磨光：在细网格对 $\boldsymbol{Ax}=\boldsymbol{b}$ 进行 l_1 次光滑迭代得到近似解 $\boldsymbol{x}^*$；

Step 2.2　粗网格校正：在粗网格上求解误差方程 $\boldsymbol{A}_c\boldsymbol{e}=\boldsymbol{b}-\boldsymbol{Ax}^*$，并校正近似解 $\boldsymbol{x}^*=\boldsymbol{x}^*+\boldsymbol{e}$；

Step 2.3　后磨光：再在细网格对 $\boldsymbol{Ax}=\boldsymbol{b}$ 进行 l_2 次光滑迭代更新近似解 $\boldsymbol{x}^*$。

从算法 1 可知，经典 AMG 算法求解稀疏线性代数方程组 $\boldsymbol{Ax}=\boldsymbol{b}$ 有 2 个阶段：启动阶段（Setup）和求解阶段（Solve）。Setup 阶段基于稀疏系数矩阵 $\boldsymbol{A}$ 构造多重网格算法组件，包括粗网格点选择、粗网格矩阵构造等，形成代数形式的网格层级结构；Solve 阶段则是在代数网格层次结构上，通过 V-Cycle 或者 W-Cycle 等多重网格循环求解近似解 $\boldsymbol{x}^*$ 直至收敛。

大量实际应用表明[8]，Setup 阶段是 AMG 并

行可扩展的关键,其中粗网格矩阵的计算(算法 1 中 Step 1.3)通常采用 Galerkin 方法[15],涉及 3 个稀疏矩阵乘积 RAP 操作。一方面,3 个稀疏矩阵乘积涉及大量非规则运算;另一方面,基于 RAP 计算的粗网格矩阵无法保持与细网格矩阵相同的稀疏结构,且随着网格粗化,粗网格矩阵虽然规模减小,但稠密度增加,稀疏结构更加复杂。这 2 个方面导致 RAP 并行性能不理想,最终影响 AMG 算法的并行效率,成为主要的性能瓶颈[8,11,16],是当前实际应用中迫切需要解决的问题。

3 RAP 算法及并行性能瓶颈

本节介绍当前主流 AMG 解法器中采用的 RAP 并行算法,并指出实际应用中发现的性能瓶颈问题。

现有 AMG 解法器,例如 Hypre[17] 和 JXPAMG(Parallel AMG solver developed by Jiu suo and XTU)[18],都采用并行 CSR(ParCSR)作为稀疏矩阵的数据结构,具体如图 1 所示。稀疏矩阵按行被划分到各个进程上,该数据结构将本地矩阵分为 2 个部分,对角部分存储本地矩阵对角块部分的非零元,非对角部分存储其他非零元。在经典 AMG 算法中,限制算子通常为插值算子的转置,即 $\boldsymbol{R}=\boldsymbol{P}^{\mathrm{T}}$,因此在实现中限制算子并不单独存储,而是直接取为插值算子 $\boldsymbol{P}$ 的转置。

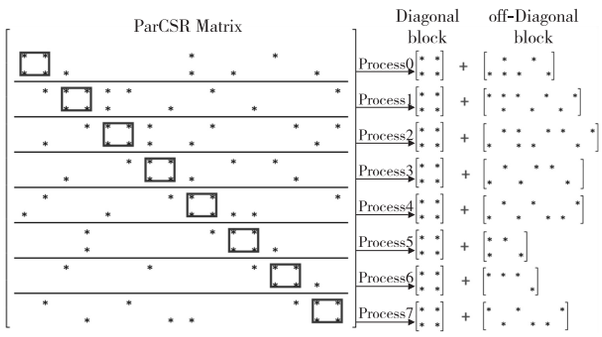

Figure 1 Diagram of ParCSR data format

图 1 ParCSR 数据格式示意图

图 2 以 3 个进程为例,给出了 0 号进程的 RAP 计算流程,主要包括以下 5 个步骤(图中上标 D 表示对角块,上标 O 表示非对角块,e 表示通信获得的矩阵,out 表示要向外发送的矩阵):

(1)将矩阵 $\boldsymbol{P}$ 转置得到 $\boldsymbol{R}$。即图 2 中步骤①,矩阵仍然以 ParCSR 格式存储。

(2)通过 MPI 通信获得 $\boldsymbol{A}\times\boldsymbol{P}$ 所需的元素。由于乘法需要用到矩阵 $\boldsymbol{P}$ 的元素,而这些元素部分位于其他进程中,所以要通信相关矩阵 $\boldsymbol{P}$ 的块。

(3)计算 $\boldsymbol{R}_{\mathrm{offd}}\times\boldsymbol{A}\times\boldsymbol{P}$。进行 1 次串行的 3 个矩阵相乘操作,该步操作是 2 次 SpGEMM 的耦

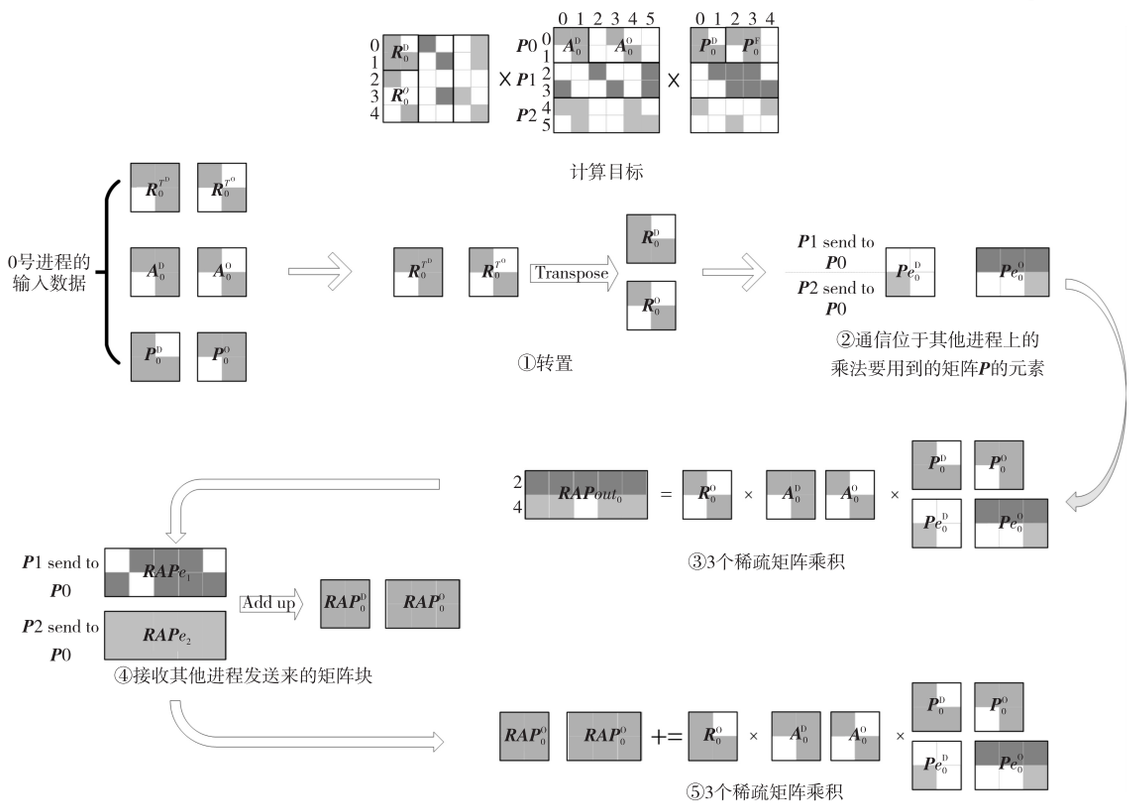

Figure 2 Algorithm flow of RAP component

图 2 RAP 组件算法流程

合,基于文献[13,19,20]的方法实现。

(4)通信 $\boldsymbol{R}_{\text{offd}} \times \boldsymbol{A} \times \boldsymbol{P}$。由于步骤(3)计算得到的矩阵块是其他进程所需的,需要将这些矩阵块发送给其他进程,且每个进程需要将接收到的矩阵块进行合并。接收到的矩阵块均以 CSR 格式存储,同时接收到的还有矩阵块各行局部索引到全局索引的映射数组。

(5)计算 $\boldsymbol{R}_{\text{diag}} \times \boldsymbol{A} \times \boldsymbol{P}$。最后,计算矩阵对角块部分的结果,并与步骤(4)得到的矩阵块相加。

下面以航空发动机应用中的 HF-C 算例(算例的详细信息见第 5.1 节)为例,采用目前主流 AMG 解法器 Hypre 的 BoomerAMG[21],测试上述 RAP 算法的并行性能。图 3 给出了该算例由串行到 2,4 和 8 个进程并行计算的时间开销。

从图 3 可知,对于实际算例 HF-C,从串行到 2 个进程并行,计算时间增加了 11.4 倍;同时可以看到,进程数大于 2 之后,RAP 计算保持了较好的并行可扩展性。但是,计算时间从串行到并行的急剧增加现象在之前的实际应用中未被关注,一个可能的原因是在实际应用中一般不会进行串行计算,因此该现象未见报道。

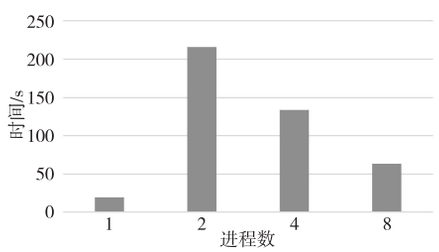

Figure 3　RAP calculation time for different number of processors in HF-C test case

图 3　HF-C 算例不同进程数 RAP 计算时间

4　MiniBranRAP:极小化分支判断数 RAP 并行算法

针对第 3 节暴露出来的并行性能瓶颈问题,本节首先开展瓶颈定位与分析,采用文献[12]的分析方法,结合性能分析工具 HPCTOOLKIT[22],发现了当前 RAP 并行算法存在的分支判断数为平方复杂度的问题。进一步利用 CSR 格式以行主序存储的特点,提出具有线性复杂度分支判断数的并行 RAP 算法,称之为 MiniBranRAP 算法。

4.1　瓶颈定位与分析

采用文献[12]中的分析方法和分析工具,逐步定位 RAP 组件的瓶颈点,发现了并行计算时矩阵合并部分(图 2 步骤④所示)时间开销相比串行计算时有明显增长。再使用性能分析工具 HPCTOOLKIT 采集性能数据,发现该部分的分支判断数有明显增长,如图 4 所示。

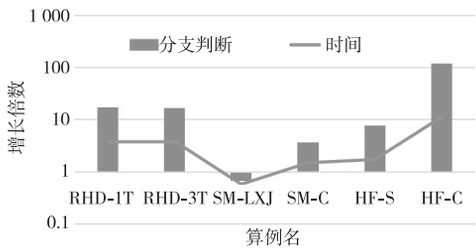

Figure 4　The increase factor of branch condition count and computation time of RAP calculation step④ with 2 processors compared to serial computing in six cases

图 4　RAP 计算步骤④在 6 个算例中 2 个进程相比于串行的分支判断数和计算时间的增长倍数

由图 4 可知,分支判断数是影响性能的因素之一。当存在大量的分支判断时,CPU 需要频繁地在不同分支之间进行切换,这会增加缓存未命中和分支预测失败的可能性,从而降低 CPU 流水线的效率,进而导致性能下降,由此可知这就是导致 RAP 组件性能问题的原因之一。下面对上述 RAP 并行算法进行分析。

在图 2 的算法流程中,步骤④的伪代码如算法 2 所示,流程如图 5 所示。可以从算法 2 的第 3～7 行看到,该算法逐行合并 $\boldsymbol{RAP}$ 与 $\boldsymbol{RAP}_j$ 得到 CSR 格式的粗网格矩阵 $\boldsymbol{A}_c$,且合并每一行的时候都会重复从 $\boldsymbol{RAP}_j$ 的首行开始比对 2 个矩阵和的行指标,导致冗余的分支判断,这里的 $\boldsymbol{RAP}_j$ 表示本进程接收的第 j 个矩阵块。如图 5 所示,假如需要合并 3 个 CSR 格式稀疏矩阵 1～3,即把矩阵 2、矩阵 3 加到矩阵 1 上,要想得到结果矩阵的最后一行(用实线框标出),必须遍历稀疏矩阵 1 的最后一行和矩阵 2、矩阵 3 的所有行(用实线框标出部分),共有 5 次分支判断。

算法 2　步骤④原始合并算法

```
for 本地矩阵 RAP 的第 i 行 do
  for 发送数据 RAP_j 到本地的进程 j do
    for 进程 j 发送数据 RAP_j 的第 k 行 do
      if 第 k 行对应本地数据第 i 行 then
        加到粗网格矩阵 A_c 的第 i 行;
      end if
    end for
  end for
end for
```

一般地,假设本地矩阵规模为 $n_{\text{loc}} = N/p$,本

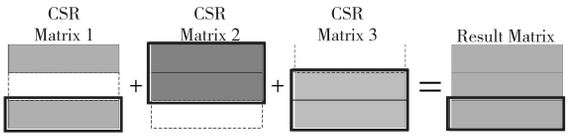

Figure 5 Illustration of the result matrix obtained by merging sparse matrices in the original algorithm

图 5 原始算法合并稀疏矩阵得到结果矩阵示意图

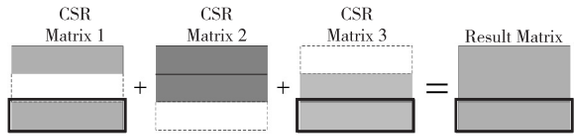

Figure 6 Illustration of the result matrix obtained by merging sparse matrices in the MiniBranRAP algorithm

图 6 MiniBranRAP 算法合并稀疏矩阵得到结果矩阵示意图

地进程平均接收进程数为 p_{recv}，接收的矩阵平均规模为 n_{recv}，其中 N 为矩阵总规模，p 为总进程数，则此时的分支判断数复杂度为 $O(n_{loc}\cdot n_{recv}\cdot p_{recv})$。通常情况下，$n_{recv}=n_{loc}=N/p$，而 p_{recv} 为常数，则复杂度为 $O((N/p)^2)$；在极端情况下，$p_{recv}=p$，此时分支判断数的复杂度为 $O(N^2/p)$，均是关于矩阵规模的平方复杂度。

4.2 MiniBranRAP 算法

分析发现，CSR 格式的矩阵以行主序存储，即行指标的顺序隐含在数据结构中，不需要重复从首行开始比对。对此，本文引入标识数组 $marker_j$ 记录 $\boldsymbol{RAP}_j$ 当前访问到的行指标，下一步比对只需从该行继续遍历即可。本文将该优化方法命名为 MiniBranRAP。如图 6 所示，要想得到结果矩阵的最后一行(用实线框标出)，仅需访问稀疏矩阵 1 和矩阵 3 的最后一行(用实线框标出部分)，共有 2 次分支判断，相比优化前减少 3 次。MiniBranRAP 的伪代码如算法 3 所示，方框中的内容是修正部分。

算法 3 MiniBranRAP 算法

```
for 发送 RAP_j 到本地的进程 j do
    marker_j ← 记录 RAP_j 第 1 行的索引;
end for
for 本地矩阵 RAP 的第 i 行 do
  for 发送数据 RAP_j 到本地的进程 j do
    while marker_j 不对应本地数据第 i 行 do
        marker_j++;
    end while
    if marker_j 对应本地数据第 i 行 then
      加到粗网格矩阵 A_c 的第 i 行;
    end if
  end for
end for
```

MiniBranRAP 算法的分支判断数的复杂度为 $O(n_{loc}\cdot p_{recv}+n_{recv})$，通常情况下为 $O(N/p)$，在极端情况下也仅有 $O(N)$，是线性复杂度。而 MiniBranRAP 仅增加了 $O(p_{recv})$ 次遍历 $\boldsymbol{RAP}_j$ 的第 1 行(算法 3 第 1～3 行)以及额外的 $O(p_{recv})$ 空间开销(数组 $marker$)。

5 数值实验

本文提出的 MiniBranRAP 算法已经集成到自主研发的并行 AMG 解法器 JXPAMG 中。本节采用几类实际应用算例，与最新版本的 Hypre 软件(Hypre-2.28.0)中的 BoomerAMG 解法器进行比较，两者均采用相同的参数，主要参数如表 1 所示。

Table 1 Parameters of JXPAMG and Hypre solver

表 1 JXPAMG 与 Hypre 解法器参数

插值类型	粗化类型	磨光类型	强连通阈值	最大 V-Cycle 数	最粗层矩阵规模阈值
修正的经典插值[23]	HIMS[24]	混合对称的 GS[21]	0.25	10	100

实验平台为某集群的一个计算结点，96 GB 内存，包含 2 个通用 CPU，共 28 个 CPU 核，单核主频为 2.60 GHz。

5.1 测试算例

本文的测试算例包含 6 个来自实际应用领域的真实算例，具体如表 2 所示，6 个算例分别来源于辐射流体力学、结构力学、航空发动机 3 个不同领域，其偏微分方程类型、离散网格类型以及离散方法都有所不同，导致离散系统的稀疏矩阵特征和非零元分布也不相同。

在这 6 个算例中，RHD-1T、RHD-3T、SM-LXJ 和 HF-C 也都被选为 SolverChallenge 竞赛题目[25]。

算例具体的代数特征和矩阵稀疏分布如表 2 和图 7 所示，从数据和图表可知：

(1)算例 RHD-1T 和 RHD-3T 使用结构网格的 7 点格式离散，稀疏元素分布较为规则，平均每行非零元数目和带宽均较小；

(2)算例 SM-LXJ 规模较小，但稠密度是所有算例中最大的；

(3)算例SM-C和HF-S非零元数目较多、平均

Table 2　Application background and basic algebraic features of test cases

表 2　测试算例应用背景与基本代数特征

算例名	应用邻域	偏微分方程	网格类型	离散方法	阶数	非零元数	平均每行非零元数	稠密度	带宽
RHD-1T	辐射流体力学流体不稳定性	三维辐射扩散方程	结构网格	有限体积7点格式	2 097 152	14 581 760	6.95	3.32×10^{6}	16 384
RHD-3T	辐射流体力学流体不稳定性	三维三温能量方程	结构网格	有限体积7点格式	6 291 456	52 133 888	8.29	1.32×10^{6}	49 152
SM-LXJ	离心机装置结构力学分析	三维线弹性方程	非结构网格	有限元	83 073	2 826 927	34.03	4.10×10^{4}	45 426
SM-C	接触问题结构力学分析	三维线弹性方程	非结构网格	有限元	12 075 090	655 371 242	54.27	4.49×10^{6}	12 075 079
HF-S	航空发动机整机静力学分析	三维线弹性方程	非结构网格	有限元	2 081 541	71 033 481	34.13	1.63×10^{5}	733 683
HF-C	航空发动机燃烧室仿真	三维压力方程	非结构网格	有限元	19 637 808	97 535 370	4.97	2.53×10^{7}	18 234 897

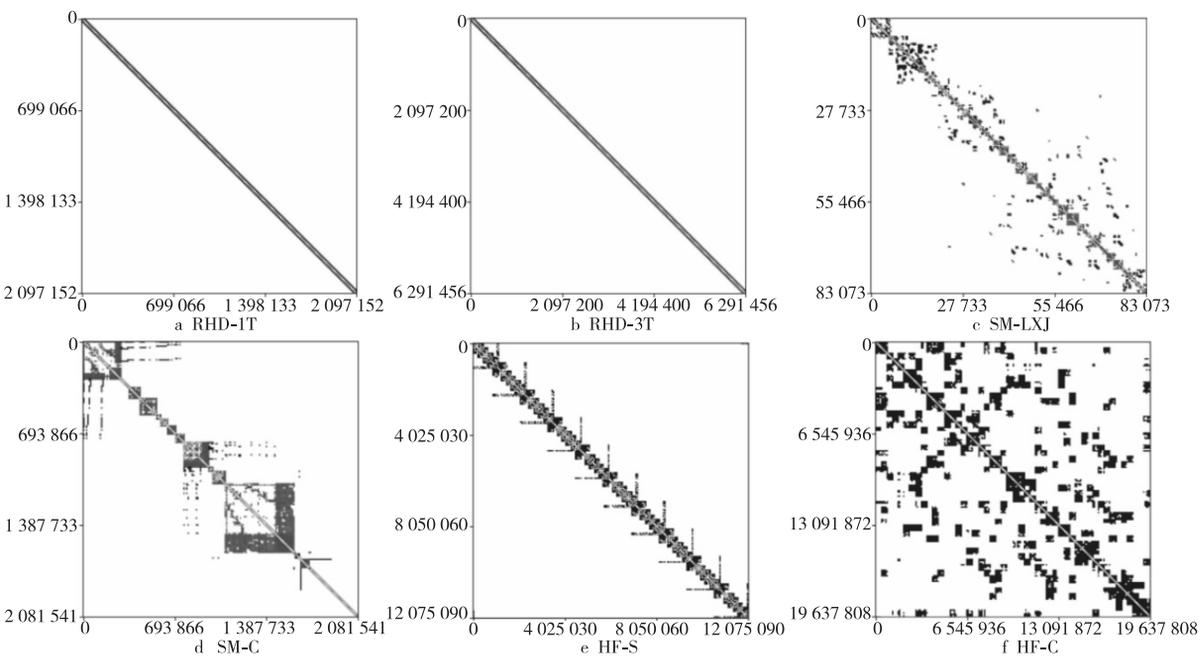

Figure 7　Distribution plot of nonzero elements in test case matrices

图 7　测试算例矩阵非零元分布图

每行非零元数目和带宽也较大;

(4)算例 HF-C 是这些算例中规模最大的,但也是最稀疏的。需要注意的是,该算例带宽最大,但平均每行非零元数目很小,这说明各行之间的非零元分布复杂。

这些具有不同特征的算例可以较全面地展示 RAP 算法的性能。

5.2　性能结果与分析

仍以 HF-C 算例为例,采用优化后的 JXPAMG 测试 RAP 算法的并行性能,图 8 给出了该算例在串行和 2,4,8 个进程并行计算时的时间开销。与 Hypre 的 BoomerAMG 相比(如图 3 所示),优化后的 JXPAMG 的 RAP 算法具有更好的并行可扩展性,且并行时的时间开销远远低于 Hypre 的。

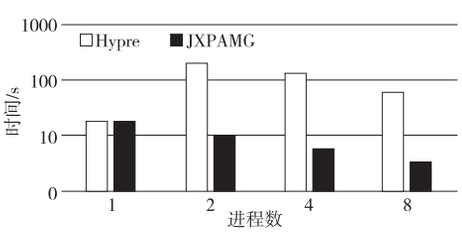

Figure 8　Comparison of RAP calculation time for different number of processors in HF-C test case: optimized JXPAMG vs Hypre

图 8　HF-C 算例不同进程数 RAP 计算时间对比:优化后 JXPAMG 与 Hypre

使用 HPCTOOLKIT 采集优化后的 JXPAMG

测试 6 个算例在 2 个进程下的分支判断数和计算时间相比于串行时的增长倍数,测试数据如表 3 所示。与 Hypre 相比(如图 4 所示),可以看出分支判断数在 2 个进程时变为串行的 0.5 倍左右,且计算时间也不再增长,基本为串行计算时间的 0.6 倍左右。

Table 3 Comparison of time increase factors for branch condition count and computation time in step④ of optimized JXPAMG and Hypre's RAP calculation, with 2 processors compared to serial computing, for six test cases

表 3 优化后的 JXPAMG 与 Hypre 的 RAP 计算步骤④在 6 个算例中 2 个进程相比于串行的分支判断数和计算时间的增长倍数对比

比较对象	解法器	RHD-1T	RHD-3T	SM-LXJ	SM-C	HF-S	HF-C
分支判断数	JXPAMG	0.5	0.5	0.5	0.5	0.5	0.5
	Hypre	17.3	16.9	0.7	3.7	7.9	120.8
时间/s	JXPAMG	0.6	0.7	0.6	0.6	0.6	0.5
	Hypre	3.8	3.9	0.6	1.5	1.7	11.4

下面通过测试 MiniBranRAP 对 Setup 阶段的影响来展示其效果。记基于原 RAP 算法的 Hypre 和 JXPAMG 解法器分别为 $Hypre_{oldRAP}$ 和 $JXPAMG_{oldRAP}$,记基于 MiniBranRAP 的 JXPAMG 为 $JXPAMG_{MiniBranRAP}$。图 9 给出了测试结果。从中可以得到如下结论:

(1)$JXPAMG_{MiniBranRAP}$ 在 Setup 阶段能取得更好的可扩展性。相比于 $Hypre_{oldRAP}$ 和 $JXPAMG_{oldRAP}$ 从串行到并行时其 Setup 阶段的时间开销会有增长,$JXPAMG_{MiniBranRAP}$ 消除了这一现象,随着进程数增加时间开销会单调下降。

(2)$JXPAMG_{MiniBranRAP}$ 的并行性能有明显提升。尽管由于在串行时 $JXPAMG_{MiniBranRAP}$ 引入了额外开销,相比于 $JXPAMG_{oldRAP}$ 的时间开销有所增长,但在并行时都可以获得一定的加速效果。

(3)$JXPAMG_{MiniBranRAP}$ 性能优于 $Hypre_{oldRAP}$ 的。虽然在串行时 $JXPAMG_{MiniBranRAP}$ 略有劣势,但在并行时 $JXPAMG_{MiniBranRAP}$ 的 Setup 阶段的时间开销都小于 $Hypre_{oldRAP}$ 的,并且取得了平均 3.3 倍、最高 9.3 倍的加速效果。

(4)对于稀疏分布更复杂、规模更大的矩阵优化效果更加明显。可以看到,对于稀疏分布最复杂、规模最大的算例 HF-C,$JXPAMG_{MiniBranRAP}$ 取得了 6 个算例中最好的优化效果,并行计算时平均加速 7.6 倍。同时对于稀疏分布最简单、规模最小的算例 SM-LXJ 优化效果最差。

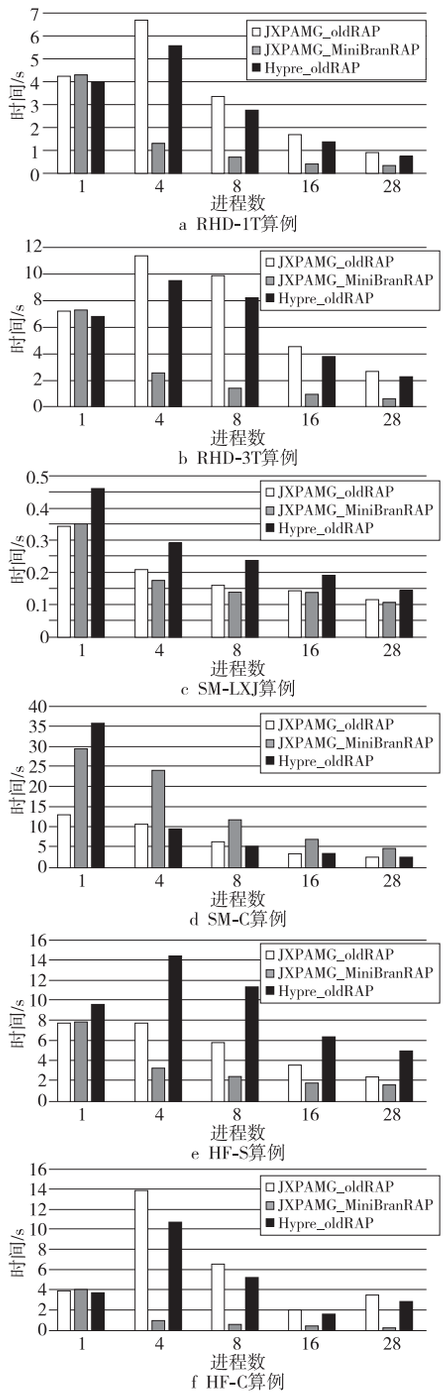

Figure 9 Comparison of time costs in Setup phase for six test cases

图 9 6 个算例 Setup 阶段时间开销对比

为进一步分析优化效果来源,本文又比较了在 4 个进程时 JXPAMG 优化前后构建多重网格前 5 层粗网格矩阵的时间开销,统计结果如表 4 所示。由于时间主要集中在前 5 层,表 4 中仅展示前 5 层的计算时间,忽略更高层级的数据。可以看到,对构建第 1 层粗网格矩阵的优化效果最为明显,平均有 15.9 倍加速、最高有 44.4 倍加速,这是性能提升最主要的来源。这进一步说明矩阵的规模越大,

本文的优化算法取得的优化效果越好,也与理论分析的分支判断复杂度相符。

Table 4　Calculation time of coarse grid matrices for six test cases with 4 processors

表 4　6 个算例 4 个进程下计算各层粗网格矩阵时间

算例名		层级				
		1	2	3	4	5
RHD-1T	优化前	5.43	0.381	0.088 2	0.018 4	3.11e−3
	优化后	0.232	0.232	0.080 6	0.018 3	3.12e−3
	加速比	23.40	1.64	1.09	1.01	1.00
RHD-3T	优化前	8.89	0.790	0.149	0.031 5	5.61e−3
	优化后	0.564	0.439	0.141	0.031 2	5.78e−3
	加速比	15.80	1.80	1.06	1.01	0.97
SM-LXJ	优化前	0.067 5	0.033 9	0.012 8	2.67e−3	4.17e−4
	优化后	0.040 2	0.031 6	0.012 4	2.64e−3	4.14e−4
	加速比	1.68	1.07	1.03	1.01	1.01
SM-C	优化前	19.3	1.04	0.131	0.014 4	1.18e−3
	优化后	3.80	0.692	0.105	0.013 1	1.17e−3
	加速比	5.08	1.50	1.25	1.10	1.01
HF-S	优化前	5.24	0.778	0.111	0.015 4	2.83e−3
	优化后	1.04	0.393	0.090 7	0.014 1	2.72e−3
	加速比	5.04	1.98	1.22	1.09	1.04
HF-C	优化前	119	12.9	1.39	0.202	0.029 1
	优化后	2.68	2.16	0.810	0.179	0.028 4
	加速比	44.40	5.97	1.72	1.13	1.02

6　结束语

本文揭示了实际应用中之前未曾注意到的由于分支判断数过多导致的 AMG 粗网格矩阵并行性能瓶颈问题,通过瓶颈定位与分析,发现当前主流 AMG 解法器的 3 个稀疏矩阵乘积并行算法涉及平方复杂度的分支判断操作。利用 CSR 格式行主序性质,本文设计了具有线性复杂度分支判断的并行算法 MiniBranRAP,消除了 RAP 操作串行到并行的瓶颈现象,并通过实际应用典型算例,验证了新算法的有效性。由于 AMG 算法流程的复杂性,且性能与具体的体系结构特征和应用特征密切相关,AMG 的性能仍然存在很大的优化空间。如何进一步挖掘计算机和应用特征对 AMG 性能的影响,开展特征驱动的 AMG 性能优化将是未来需要持续研究的课题。

参考文献:

[1] Ruge J W, Stüben K. Algebraic multigrid [M] // Multigrid Methods. Philadelphia: Society for Industrial and Applied Mathematics, 1987: 73-130.

[2] Stüben K. A review of algebraic multigrid [J]. Journal of Computational and Applied Mathematics, 2001, 128 (1-2): 281-309.

[3] 徐小文,莫则尧,安恒斌. 求解大规模稀疏线性代数方程组序列的自适应 AMG 预条件策略[J]. 中国科学:信息科学, 2016, 46(10): 1411-1420.
Xu Xiao-wen, Mo Ze-yao, An Heng-bin. An adaptive AMG preconditioning strategy for solving large-scale sparse linear systems[J]. Scientia Sinica Informations, 2016, 46(10): 1411-1420.

[4] Saad Y. Iterative methods for sparse linear systems [M]. Philadelphia: Society for Industrial and Applied Mathematics, 2003.

[5] Tian R, Zhou M Z, Wang J T, et al. A challenging dam structural analysis: Large-scale implicit thermo-mechanical coupled contact simulation on Tianhe-II[J]. Computational Mechanics, 2019, 63(1): 99-119.

[6] Griebel M, Schüller A, Schweitzer M A. Scientific computing and algorithms in industrial simulations: Projects and products of fraunhofer SCAI[M]. Berlin: Springer, 2017.

[7] Brannick J, Kahl K. Bootstrap algebraic multigrid for the 2D Wilson Dirac system[J]. SIAM Journal on Scientific Computing, 2014, 36(3): B321-B347.

[8] 徐小文. 并行代数多重网格算法:大规模计算应用现状与挑战[J]. 数值计算与计算机应用, 2019, 40(4): 243-260.
Xu Xiao-wen. Parallel algebraic multigrid methods: state-of-the art and challenges for extreme-scale applications [J]. Journal on Numerical Methods and Computer Applications, 2019, 40(4): 243-260.

[9] Baker A H, Gamblin T, Schulz M, et al. Challenges of scaling algebraic multigrid across modern multicore architectures[C] // Proc of 2011 IEEE International Parallel & Distributed Processing Symposium, 2011: 275-286.

[10] Yang U M. Parallel algebraic multigrid methods—high performance preconditioners[M] // Numerical Solution of Partial Differential Equations on Parallel Computers. Berlin, Heidelberg: Springer, 2006: 209-236.

[11] Falgout R D, Schroder J B. Non-Galerkin coarse grids for algebraic multigrid[J]. SIAM Journal on Scientific Computing, 2014, 36(3): C309-C334.

[12] 毛润彰,杜皓,田鸿运,等. 几类典型应用的代数多重网格算法并行可扩展瓶颈分析[J/OL]. 计算物理, 1-16[2023-11-24]. http://kns.cnki.net/kcms/detail/11.2011.O4.20230506.1706.002.html.
Mao Run-zhang, Du Hao, Tian Hong-yun, et al. Analysis of parallel scalability bottleneck for algebraic multigrid in typical real applications[J/OL]. Chinese Journal of Computational Physics, 1-16[2023-11-24]. http://kns.cnki.net/kcms/detail/11.2011/O4.20230506.1706.002.html.

[13] Park J, Smelyanskiy M, Yang U M, et al. High-performance algebraic multigrid solver optimized for multi-core based

distributed parallel systems[C]//Proc of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2015: 1-12.

[14] Liu W F, Vinter B. An efficient GPU general sparse matrix-matrix multiplication for irregular data[C]//Proc of 2014 IEEE the 28th International Parallel and Distributed Processing Symposium, 2014: 370-381.

[15] Hemker P W. A note on defect correction processes with an approximate inverse of deficient rank[J]. Journal of Computational and Applied Mathematics, 1982, 8(2): 137-139.

[16] Pichahi S M R. Improving the performance and scalability of algebraic multigrid[D]. Utah: The University of Utah, 2021.

[17] Falgout R D, Yang U M. Hypre: A library of high performance preconditioners[C]//Proc of International Conference on Computational Science, 2002: 632-641.

[18] Xu X W, Yue X Q, Mao R Z, et al. JXPAMG: A parallel algebraic multigrid solver for extreme-scale numerical simulations[J]. CCF Transactions on High Performance Computing, 2023, 5(1): 72-83.

[19] Gustavson F G. Two fast algorithms for sparse matrices: Multiplication and permuted transposition[J]. ACM Transactions on Mathematical Software, 1978, 4(3): 250-269.

[20] Patwary M M A, Satish N R, Sundaram N, et al. Parallel efficient sparse matrix-matrix multiplication on multicore platforms[C]//Proc of International Conference on High Performance Computing, 2015: 48-57.

[21] Yang U M, Henson V E. BoomerAMG: A parallel algebraic multigrid solver and preconditioner[J]. Applied Numerical Mathematics, 2002, 41(1): 155-177.

[22] Adhianto L, Banerjee S, Fagan M, et al. HPCTOOLKIT: Tools for performance analysis of optimized parallel programs[J]. Concurrency and Computation: Practice and Experience, 2010, 22(6): 685-701.

[23] De Sterck H, Falgout R D, Nolting J W, et al. Distance-two interpolation for parallel algebraic multigrid[J]. Numerical Linear Algebra with Applications, 2008, 15(2-3): 115-139.

[24] De Sterck H, Yang U M, Heys J J. Reducing complexity in parallel algebraic multigrid preconditioners[J]. SIAM Journal on Matrix Analysis and Applications, 2006, 27(4): 1019-1039.

[25] SOLVER 会议组委会. 第二届线性解法器算法与性能优化竞赛 [EB/OL]. [2023-11-24]. https://www.solver-conference.cn/solverchallenge22/.

作者简介:

杜皓(2001-),男,山东东营人,硕士生,研究方向为并行算法和高性能计算。**E-mail**:duhao23@gscaep.ac.cn

DU Hao, born in 2001, MS candidate, his research interests include parallel algorithm and high performance computing.

毛润彰(1998-),男,湖南岳阳人,博士生,研究方向为高性能并行计算和线性代数解法器。**E-mail**:mrz@nudt.edu.cn

MAO Run-zhang, born in 1998, PhD candidate, his research interests include high performance parallel computing and linear algebra solver.

邓蕴桐(1999-),女,广东肇庆人,硕士生,研究方向为并行数值算法。**E-mail**:dengyuntong21@gscaep.ac.cn

DENG Yun-tong, born in 1999, MS candidate, her research interest includes parallel numerical algorithm.

黄思路(1992-),女,吉林龙井人,博士,助理研究员,研究方向为高性能计算。**E-mail**:huang_silu@iapcm.ac.cn

HUANG Si-lu, born in 1992, PhD, assistant research fellow, her research interest includes high performance computing.

徐小文(1978-),男,湖南郴州人,博士,研究员,研究方向为高性能科学与工程计算、大规模并行计算和并行数值算法。**E-mail**:xwxu@iapcm.ac.cn

XU Xiao-wen, born in 1978, PhD, research fellow, his research interests include high performance scientific and engineering computing, large-scale parallel computing, and parallel numerical algorithm.