

大规模线性问题求解算法的高可扩展性研究^{*1)}

吴学淞 曹建文 王宇鹏

(中国科学院软件研究所, 并行软件与计算科学实验室, 北京 100190)

摘 要

针对大规模线性问题的迭代求解算法在 100PF 乃至 E 级超级计算机上的高可扩展性问题进行研究. 首先分析了数理模型、数值方法、体系结构的耦合和非均衡特性, 将高可扩展性问题归结到的权重图的剖分问题上. 然后对若干图剖分算法与软件包进行分析, 基于 Petal 剖分算法实现了全局通信开销近似优化的大规模聚类分组算法. 与基于舒尔补架构的线性问题求解方法相结合, 给出了具有高可扩展性的线性问题求解架构 (Schur-KS). 最后通过大规模环境下的高可扩展性测试和中小规模环境下针对典型实际应用问题用例的强可扩展性测试, 分别验证了基于 Schur-KS 架构的求解算法的高可扩展性以及典型实际应用问题的求解性能.

关键词: 高可扩展性; 通信开销极小化; 图剖分; 权重连通图; 迭代求解

MR (2010) 主题分类: 34M03; 97K30

RESEARCH ON HIGH SCALABILITY OF ITERATIVE METHOD FOR LARGE-SCALE LINEAR SYSTEM

Wu Xuesong Cao Jianwen Wang Yupeng

(Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract

The iterative solution algorithm for large-scale linear problems is studied on the high scalability problem of 100PF and even E-class supercomputers. Firstly, the coupling and non-equilibrium characteristics of the mathematical model, numerical method and computer architecture are analyzed, and the high scalability problem is attributed to the partitioning problem of the weighted graph. Then, some graph partitioning algorithms and software packages are analyzed. Based on Petal-decomposition algorithm, the algorithm of large-scale clustering with approximate optimization of global communication overhead is given. Combined with the solving method of linear problem based on Schur complement framework, a solving framework of linear system (named Schur-KS) with high scalability is designed. Finally, through the high scalability test in large-scale parallel computing environment and the strong scalability test of typical practical application problem cases in small and medium scale environment, the high scalability of the algorithm based on Schur-KS framework and the performance of typical practical application problems are verified respectively.

* 2018 年 12 月 24 日收到.

¹⁾ 基金项目: 国家自然科学基金 (重大研究计划 91430214, 面上项目 61876175) 资助.

Keywords: high scalability; minimum communication cost; graph-decomposition; weighted connected graph; iterative method

2010 Mathematics Subject Classification: 34M03; 97K30

1. 引言

科学计算的兴起是 20 世纪后半叶最重要的科技进步之一, 计算与理论及实验相并列, 已经成为当今世界科学活动的第三种手段, 在众多领域, 如在机械制造、材料加工、航空航天、汽车船舶、土木建筑、国防军工、科学研究等领域, 是必不可少的工作. 随着科技的进步、所求解问题规模的增大, 应用问题的模型结构越来越复杂且对计算精度的要求日益提高, 使得问题的规模急剧增加. 在气候环境预报、核聚变、石油开发、飞机设计等领域, 多物理量耦合的偏微分方程问题求解, 其精细模拟可以达到的问题规模严重受制于计算机体系结构和求解算法的可扩展性, 主要瓶颈之一为复杂的全局通信^[1, 2].

近年来, 随着计算机集群系统的发展, 处理核心数量的急剧增加, 计算能力的大大提高, 尤其是 100PF(100P flops, 即每秒十亿亿次浮点运算) 级超级计算机“神威·太湖之光”的问世, 大大推动了我国大规模科学计算研究的进展. 国家 863 计划重大项目高效能计算机及应用服务环境的技术报告显示, 基于 PF 级计算机的偏微分方程问题隐格式求解应用程序, 其有效可扩展性总体上达到了万核量级. 国际上, 使用 PDE 问题的传统求解算法库 (如 PETSc, HyPre, Sundials, SAMRAI 等), 隐式求解 CFD 离散化形成的大规模线性代数方程组, 可扩展性一般可达到 1 万 -10 万核量级. 更多处理器核的并行计算, 往往会导致并行效率的严重下降. 即使考虑到使用特别定制、极低延迟的高速网络, 国内外的现有通用数值算法库, 在 100PF 级乃至 E 级计算机上也存在可扩展性瓶颈.

在科学与工程应用的实际问题中, 数理模型、数值方法、体系结构等领域具有自身的特点: 特定的分层耦合和非均衡特性: 数理模型通常在纵向上呈现多尺度的分层耦合特性、横向上呈现非线性系数在空间分布上的非均衡特性; 超级计算机上的大规模数值求解方法, 通常在纵向上呈现非线性 - 线性 - 预处理 - 迭代循环的分层耦合特性、横向上呈现非均匀扭曲、自适应加密等导致的网格离散在空间分布上的非均衡特性; 体系结构 - 异构的硬件系统, 通常在纵向上呈现系统 - 超节点 - 节点 - 核的分层耦合特性、横向上呈现 CPU 核连通图的权重在空间分布上的非均衡特性.

三个领域 (数理模型、数值求解方法、体系结构) 的分层耦合与空间非均衡性, 为发展高可扩展性数值计算算法带来了挑战和机遇. 从国内外文献可以看出, 三个领域在不同的发展阶段, 有着不同程度的交叉与耦合. 数理模型与数值方法的相互作用, 体现在计算物理与计算数学的交叉上, 甚至达到了相对系统地研究阶段; 数值方法与体系结构的相互作用, 目前正处于逐步成型的阶段, 从美国能源部 SciDAC 项目的数值算法库可以看到这方面的初步成果 (如 Metis 中的四叉树^[3]); 数理模型与体系结构的相互作用, 还处于起步阶段, 国际上也存在一些研究与关注 (如蔡小川的非线性预条件子^[4], 考虑了非线性的空间非均衡). 三个领域之间的交叉与耦合, 是设计和实现高效数值计算算法的重点. 本文基于计算机体系结构和数值方法的耦合, 对大规模线性问题的迭代求解算法高可扩展性问题进行研究.

对于超级计算机体系结构, 特点体现为大规模、多层和异构, CPU 核之间的通信性能并不相等, 这形成空间分布非均衡的关联图, 其中节点内 CPU 核紧密耦合, 超节点形成强连通

分量, 而超节点间是相对松散的弱连通; 通信网络表现为非完全图特征, 在几何拓扑上构成带权重的连通图. CPU 核为图节点, CPU 核之间的网络性能, 由边的权重函数定量给出; 权重函数的倒数, 定义了边的长度, 由此给出了路径长度, 以及任意两个 CPU 核间的距离, 进而给出了生成树的节点间的拉伸强度 (Stretch) 函数. 这样, 硬件平台异构特性导致的应用程序网格剖分、处理器剖分、处理器核映像、负载均衡等优化问题, 就能够归结为体系结构关联图的特定剖分问题, 从而得到合理而有效的解决.

对于大规模数值求解方法, 其对应的矩阵结构需要考虑网格离散导致的空间非均衡性. 在引入矩阵重排序、把系数矩阵视为宏观上的分块矩阵后, 主对角分块代表了输入的局地节点, 非主对角分块的元素代表了输入节点之间的关联或相互作用. 在这个视角之下, 矩阵的结构与图结构形成了一一对应的关系, 矩阵的对角分块对应着图的超节点, 矩阵的非对角元素对应着超节点的边及权重. 这样能够针对其形成的权重关联图, 引入图论的有效工具和快速算法对矩阵分块问题进行有效的解决.

大规模线性问题的迭代并行算法^[2], 其高可扩展性瓶颈, 主要体现在全局通信开销和全局通信频次上. 基准通信开销 (每个迭代步的通信开销) 与逆矩阵的非零元个数成正比; 全局通信频次, 与处理器群组构成的连通子图之间的割边 (Edge Separators) 数量成正比; 算法所需要的总迭代步数, 与逼近精度密切相关, 算法的总体计算开销, 与预处理系统的近似求解的计算代价密切相关. 高可扩展性的实现, 一般有三种途径: 其一是降低求解器的整体迭代次数, 这意味着需要选取高逼近精度的预条件子 (这部分工作可以参见本课题组的论文^[6]); 其二是降低基准通信开销, 这意味着需要选取通信极小化的处理器分组、矩阵分块策略; 其三是改进 Krylov 子空间迭代算法的实现手段.

本文通过第二个途径 - 降低基准通信开销, 开展理论分析和算法研究. 我们判断, 如果数值算法能够获得并行计算机体系结构自身存在的分层聚类的空间分布非均衡信息, 据此指导求解算法进行系数矩阵的分块策略, 至少会有正面的帮助. 因此, 本文针对大规模线性问题的迭代求解算法高可扩展性瓶颈进行了理论分析与算法研究, 基于 Petal- 剖分给出了大尺度聚类分组算法与程序实现, 以近似线性的时间代价 $O(n(\log n)^\beta)$ (n 为稀疏的权重图的节点个数, $0 < \beta < 1$)^[5] 实现了通信开销的近似优化, 将图剖分导致的 Edge Separator 的规模由传统意义下的 $O(n^{1+\epsilon})$ ($0 < \epsilon < 1$)^[6] 降低为 $O(n(\log n)^\alpha)$ ($0 < \alpha < 1$)^[5]. 其意义在于, 当处理器节点的个数达到 $10^6 - 10^7$ 量级时, 基于区域分解的并行计算, 其通信开销可以下降 1-2 个数量级, 这会带来并行计算可扩展性的明显提升. 基于 Petal- 剖分和舒尔补 (Schur-complement) 方法, 我们给出了具有高可扩展性的线性问题求解架构 (我们将其命名为 Schur-KS) 及其程序模块的实现, 在 100PF 级超级计算机神威·太湖之光上对其进行了测试与分析.

本文结构如下: 第 2 节介绍了基于 Petal- 剖分的大规模聚类分组算法; 第 3 节介绍了基于 Petal- 剖分、舒尔补架构的大规模线性问题求解算法的设计; 第 4 节在神威·太湖之光上的对基于 Schur-KS 架构的求解算法进行高可扩展性与强可扩展性测试与结果分析.

2. 基于 Petal 剖分的大尺度聚类分组算法

定义 1. 割边集^[6]

给定权重连通图 $G(V, E, l)$, 子集 $S \subseteq V$, 剖分 $(S, V - S)$. 剖分所对应的割边集 (Cut

edges) 定义为

$$E(S, \bar{S}) = \{e := (u, v) \in E | u \in S, v \in V - S\};$$


定义 2. 割边集的代价函数^[6]

给定权重连通图 $G(V, E, l)$, 边 e 的代价函数一般选择为其长度 $l(e)$ 的正相关函数, 在图论算法中, 我们使用 $\text{cost}(e) := l(e)$. 这样, 割边集 $E(S, S)$ 的代价函数就可以表示为:

$$\text{cost}(S, \bar{S}) \equiv \text{cost}(E(S, \bar{S})) = \sum_{e \in E(S, \bar{S})} l(e) = \sum_{e \in E(S, \bar{S})} \frac{1}{w(e)}.$$

剖分子集之间的割边集小^[7], 意味着全局通信的代价低廉. 由于我们需要能够在超大规模计算区域 (例如, 未知量的个数达到 $O(10^{12})$) 上使用大量处理器 ($O(10^6)$ - $O(10^7)$) 进行并行计算. 因此, 我们设计的高可扩展性的求解架构, 其并行剖分是分级进行的, 第一级是大尺度的聚类分组, 将现有的空闲处理器资源分解为若干个处理器群组 (clusters), 将现有的计算区域分解为若干个计算子区域 (clusters); 第二级是中小尺度级别的并行分解, 在每个处理器群组 and 计算子区域的内部, 进行中小粒度的处理器分组和计算区域的分解. 在这种并行模式下, 第一级的聚类分组不需要做负载均衡上的考虑, 它所要实现的是聚类分组的“通信极小化”. 在图论中归结为 Partitioning(剖分) 计算 (它是 NP-hard 问题) 的 Minimum Cut 优化求解算法. 在足够多的未知量和处理器的情况下, 负载均衡问题应该放在第二级剖分中实现, 通信极小化问题在第一级剖分中对算法整体的可扩展性影响最大, 应该作为唯一的约束, 进行剖分算法的近似优化实现.

第一级的剖分, 即关联图剖分的全局通信代价极小化问题, 归结为图的最小割边剖分问题, 这属于分解理论方面的研究. 伴随着高性能计算机的长足进步, 以及大规模科学与工程计算对区域剖分的需求日益增强, 出现了一系列的区域剖分算法和相应的软件包, 如 hMETIS^[8]、ParMETIS^[9]、Metis^[3]、PTScotch^[10] 等. 这些高水平的图剖分算法库难以直接调用的原因在于: 在 hMETIS 这些区域剖分软件包中, 剖分算法均考虑了通信极小化与负载均衡的取舍, 负载均衡的地位在通信极小化的地位之上, 因此, 它们不能适用于我们需要的具有“通信极小化”特点的关联图剖分算法的要求. 其中, hMETIS 采用了面向逐级粗化的超图 (hypergraph) 的二分剖分 (bisection) 算法, 二分法在负载均衡的约束下, 通信复杂性损失过大; ParMETIS 为 Metis 的并行版, 采用了分层逐级粗化 (Multilevel) 的二分或者 k-way 剖分算法, k-way 剖分需要事先输入计算区域和处理器群组的聚类数目, 在并行剖分程序的自适应性方面存在不足; PTScotch 采用了混合的不完全嵌套分解的二分剖分策略, 通信复杂度同样损失过大. 我们希望得到通信开销极小化的图剖分, 需要不预先设定 k 值, 从而实现剖分的高可扩展性, 因此我们在第一级的剖分中没有采用上述算法.

理论研究表明, 权重连通图的剖分问题, 其优化算法是 NP-hard^[11] 问题的精确解的计算, 其计算复杂性往往是指指数增长. 为了在逼近精度和时间代价上取得  我们需要在多项式时间复杂性的近似算法中进行选择. 从逼近精度的视角来看, 截止到 2017 年, 最佳近似算法的割集理论逼近精度可达到 $O(n \log^{\frac{1}{2}} n)$, 其时间代价的开销为 $O(n^{4.5})$ ^[12, 13]. 在大规模高性能计算机上, 该最佳近似算法的可扩展性很差, 不适合在高可扩展的算法模块中使用. 我们对基于 Ball-growing 的图剖分算法、基于 Star- 分解的图剖分算法、基于 Petal- 分解的图剖分算法等进行分析, 实现了基于 Petal- 剖分的大尺度聚类分组算法, 以近似线性的时间代价 ($O(n(\log n)^\beta)$) 实现了通信开销的近似优化 (逼近精度为 $O(n(\log n)^\alpha)$).

2.1. 基于 Ball-growing 的图剖分算法

Alon 提出一种基于 Ball-growing 的图剖分算法^[14], 该算法通过寻找具有 Low cost 的稀疏割边集, 将图剖分为一些 Low radius 的诱导子图. Ball-growing 策略的聚类分解思想描述如下:

从给定的源节点出发, 利用 BFS 算法沿径向逐层寻找邻居节点, 形成一系列球壳形状的子区域, 直到 $\frac{\sum_{e \in E(C, V-C)} \ell(e)}{|E(C)|}$ 小于给定的参数 θ , 则将该子区域视作一个 cluster.

给定剖分子区域 $S := B(v, r)$ 和它的诱导子图 $G[S]$, 其体积 $\text{vol}(S)$ 的计算公式表示为:

$$\text{vol}(S) := \sum_{e \in E(S)} |E(S)| + \sum_{(u,v) \in E(S, V-S), v \in S} \frac{r - d(u, v)}{\ell(u, v)}.$$

定理 1. Ball-growing 剖分的代价函数及其效果估计^[14]

给定权重连通图 $G(V, E, \ell)$, $B(v, r)$ 是以 v 为中心, r 为半径的球对应的诱导子图, 那么对任意实数 $0 \leq r_1 < r_2$, 存在 $r \in [r_1, r_2]$ 及常数 c , 满足

$$\text{cost}(B(u, v), V - B(u, v)) \leq c \cdot \frac{\text{vol}(B(v, r))}{r_2 - r_1} \cdot \log \left(\frac{\text{vol}(B(v, r_2))}{\text{vol}(B(v, r_1))} \right),$$

其基本原理为:

a) 通过寻找割边集将 $G(V, E, \omega)$ 分解成半径不超过 $R = O(x \log n)$ 的剖分区域, x 是用户指定参数;

b) 通过选取恰当的参数 x , 使得割边集的规模不超过 $|E(G)|$ 的 $1/x$, 从而保证割边集的代价函数具有 Low cost 特性.

给定 $n = |V|$, $m = |E|$, 当参数 $x = O(e^{\sqrt{\log n \log \log n}})$, $m = O(n)$ 时, Ball-growing 剖分算法的割边集的代价函数为 $O(ne^{\sqrt{\log n \log \log n}})$, 它是 $O(n^{1+\epsilon})$ 量级, 时间开销为 $O(m^2)$. Ball-growing 剖分算法导致的割边集的规模达到 $O(n^{1+\epsilon})$, 时间开销为 $O(m^2)$. 当处理器节点的个数达到 $O(10^6)$ - $O(10^7)$ 量级时, 基于 Ball-growing 剖分的并行计算, 通信开销很高, 这是我们没有采用 Ball-growing 剖分的原因.

2.2. 基于 Star- 分解的图剖分算法

Elkin 等通过引入锥 (cone) 的概念, 将 Alon 的基于球增长的聚类技术推广, 给出了基于锥增长技术的聚类算法^[15], 进而提出了 Star 剖分算法.

定理 2. 具有低代价函数特性的 Star- 分解

给定权重连通图 $G(V, E, \ell)$, $n := |V|$, $m := |E|$, 指定节点 $x_0 \in V$. $0 < \epsilon < \frac{1}{2}$, 存在一个 $(\frac{1}{3}, \epsilon)$ -Star- 分解将 V 剖分为 (V_0, V_1, \dots, V_k) . Star- 分解的时间开销为 $O(m + n \log n)$, 其割边集的代价函数为

$$\text{cost}(E(V_0, \dots, V_k)) \leq \frac{6 \cdot m \log_2(m+1)}{\text{rad}_G(x_0)\epsilon}.$$

Star- 分解可以描述为: 它是根节点簇基于球集、叶节点簇基于锥集的聚类方法, 通过寻找满足近似优化条件 (上述代价函数) 的割边集, 将 $G(V, E, \omega)$ 的节点集 V 剖分为 $k+1$ 个

分划 (V_0, V_1, \dots, V_k) , 然后将球集 V_0 与锥集 $V_i (i > 0)$ 以星型的树状结构连接. 当 $m = O(n)$ 时, Star- 分解算法与 Ball-growing 分解算法相比, 其割边集的代价函数由 $O(n^{1+\epsilon})$ 降低至 $O(n \log^2 n)$, 时间开销减小至 $O(m \log n + n \log^2 n)$.

2.3. 基于 Petal- 分解的图剖分算法

Ofer Neiman 等基于锥增长技术的聚类算法, 通过寻找近似的稀疏割集将图分为多个分划, 其中分划之间的连接具有花瓣式 (petal) 结构, 给出了一种 petal 剖分算法^[5]. 与 Star- 分解相比, 该算法在寻找具有低代价函数的割边集时, 增加了控制参数 p 及约束条件:

$$\text{vol}(W_r) \leq E(X) \cdot y.$$

这里 y 是与 m, p 有关的表达式. 基于 Petal- 分解的图剖分算法的基本过程描述如下:

先寻找满足上述约束条件的子集 W_r , 然后通过设置函数 y 得到满足 Low cost 近似优化

$$\text{cost}(E(P_r, V - P_r)) \leq \frac{\text{vol}(P_r) \ln y}{r_{j-1} - r_j}$$

的割边集, 从而得到剖分分划 $P_r \subset W_r$. Petal- 剖分将图剖分导致的 Edge Separator 由传统的 $O(n^{1+\epsilon})$ 降低为 $O(n \log n \log \log n)$, 时间开销减少至 $O(m \log n \log \log n)$, 实现了拟线性时间复杂度的图分割的近似算法, 从而能够以近似线性的时间代价实现通信开销的近似优化. 该算法的意义在于, 当处理器节点的个数达到 $O(10^6) - O(10^7)$ 量级时, 基于这种具有近似优化特性的区域分解的并行计算, 其通信开销可以近似下降 1-2 个数量级, 这会带来并行计算可扩展性的明显提升.

命题 1. 给定权重连通图 $G(V, E, \ell)$, 点 $x_0 \in V$. 设 $a = (1 + (p - 1)/L)R/2$, $b = (1 + p/L)R/2$, $L = \log n \log \log n$, p 是满足 $1 \leq p \leq L$ 的正整数, 那么存在 $r \in [a, b]$ 使得 Petal- 剖分可以形成半径为 r 的分划 P_r 满足

$$\text{cost}(E(P_r, V - P_r)) \leq E(P_r) \cdot \frac{8L \ln \chi}{R},$$

其中 $E(P_r, V - P_r)$ 表示分划 P_r 的边集, $\chi = \frac{|E|}{|E(P_r)|}$ ^[5].

在通信极小化的区域剖分策略设计中, 我们综合时间代价和割集精度考虑, 选取并实现了基于 Petal- 剖分的剖分算法^[5], 它主要作用于第一级的大尺度聚类分组, 可以成功地实现通信开销的近似优化.

2.4. Petal- 剖分算法的并行化设计

Petal- 剖分算法的主要步骤是: 首先搜索权重图中心点 x_0 , 记录半径 r_0 ; 获得以 x_0 为终点, 最短路径长度为 r_0 的点 x_k 以及最短路径上的所有点 x_k, x_{k-1}, \dots, x_0 , 其中最短路径距离 $d(x_k, x_0) > d(x_{k-1}, x_0) > \dots > d(x_1, x_0)$; 对于每个点 $x_i \in \{x_i | d(x_i, x_0) \geq r_0/2, k \geq i \geq 0\}$, 以 R 为半径, 对点 $y_i \in \{y_i | d(y_i, x_i) \leq R\}$ 计算加入当前剖分子块 X_1 的条件, 并获得剖分子块 X_1 . 重复上述类似的过程, 直至获得相应的剖分子块序列 X_j, X_{j-1}, \dots, X_1 , 最后剩余的点集为子块 X_0 . 上述过程中的变量定义、算法细节和伪代码可以参阅^[5].

针对大规模稀疏权重图 $G(V, E, \ell)$, m 为边的数量, n 为点的数量, 在 Petal- 剖分算法的并行化设计, 我们采用了分布式存储的图数据格式和并行化的 dijkstra 最短路径算法、分布式的广度优先搜索算法. 首先在 N 个处理器上分别存储 n/N 个节点 $(G_i, i = 1, 2, \dots, N)$ 的数据, 相应节点的数据包括: 节点的全局 id、邻接点的全局 id、邻接边的权重、所属的剖分子块编号、距离中心点的最短路径距离、最短路径上指向中心节点方向的前一个节点的全局 id. 为了减少获取邻接点数据的通信次数, 我们在当前处理器 N_i 上额外存储了与当前处理器上的节点相邻、但是不属于 G_i 的节点的数据. 这样设计的分布式图数据结构, 在全局数据更新时, 仅需更新当前处理器的相邻节点数据, 减少了更新数据所需的通信.

在搜索权重图中心点 x_0 过程中, 需要多次调用 dijkstra 最短路径算法, 为了提高算法的计算效率, 我们采用了 [16] 中的 dijkstra 算法的并行实现; 在获取所有点 $y_i \in \{y_i | d(y_i, x_i) \leq R\}$ 的过程中, 需要多次调用广度优先搜索算法, 我们采用了分布式的广度优先搜索算法的程序实现 [17]. 基于开源的图论算法软件包 Parallel Boost Graph Library [17], 我们实现了具有全局通信极小化特点的权重连通图的聚类分组的并行程序. 基于 Petal- 剖分的聚类分组算法, 与数据挖掘领域的 MeanShift 聚类算法相比较, 其不要求处理器结点的分布各向同性; 与当前主流的 Metis 剖分算法相比较, 其不要求事先指定处理器群组 (Clusters) 的个数 (k-way Partition), 就可以实现通信开销的近似优化.

3. 基于 Schur-KS 架构的线性问题求解算法

3.1. 基于舒尔补方法的线性问题求解架构

近年来, 基于舒尔补架构的求解算法 [18, 19] 是用于研究大规模线性问题求解算法的热点之一, 它通过图剖分的策略 (nested dissection algorithm、Recursive Hypergraph Bisection (RHB) algorithm 等 [20]), 将原始的线性系统 $Ax = b$ 重排序为如下分块结构:

$$\begin{pmatrix} D_1 & & & E_1 \\ & D_2 & & E_2 \\ & & \ddots & \vdots \\ & & & D_k & E_k \\ F_1 & F_2 & \cdots & F_k & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \\ y \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_k \\ g \end{pmatrix}, \quad (3.1)$$

其中 D_i 代表第 i 个 interior 子区域的分块矩阵, $i = 1, 2, \dots, k$, C 为 separators 分块矩阵, E_i 和 F_i 代表 E_i 与 C 之间的关联. 为了求解原始线性系统的解, 我们首先通过求解舒尔补子系统 $Sy = \hat{g}$ 来求解 y , 其中 S 定义为 $S = C - \sum_{i=1}^k F_i D_i^{-1} E_i$, $\hat{g} = g - \sum_{i=1}^k F_i D_i^{-1} f_i$, 然后对 u_i 进行求解: $D_i u_i = f_i - E_i y$. 其中, 舒尔补 S 一般具有较小的条件数 (condition number), 但是它比原始矩阵 A 更加稠密 [21]. S 和 D_i 往往需要采用高可扩展性的预条件子技术来求解 [6]. 在并行计算过程中, 矩阵主对角子块在相应的处理器群组中进行运算, 其存放于相应处理器的局地内存中, 非主对角的矩阵子块构成了不同处理器之间的相互作用 (interaction), 需要调用 MPI 通信库函数进行消息传递. 在这个视角下, 舒尔补方法形成了权重连通图的聚类剖分.

3.2. Schur-KS 架构的设计

基于大尺度聚类分组算法和舒尔补求解框架, 我们设计了大规模线性问题求解架构 (Schur-KS). 我们首先给出一个简单的例子来说明 Schur-KS 求解架构, 然后给出基于 Schur-KS 架构的线性系统求解流程.

以 10 万处理器的并行计算为例: 在求解过程中, 针对 10 万处理器之间的连接关系与通信时长形成权重图, 调用 Petal- 剖分算法形成 4 个“通信极小化”处理器群组, 每个群组拥有 2.5 万核, 并选出 ROOT. 系数矩阵采用同样的剖分算法, 形成 4 个主对角矩阵子块, 每个子块分配给相应数量的群组. 主对角子块之间的关联, 作为一个整体, 配置到各自群组的 ROOT 中. 对每个群组内的区域分解, 调用当前的数值计算算法库 (如 PETSc^[22]).

算法 1 基于 Schur-KS 架构的线性系统求解算法

输入: 线性方程组 $Ax = b$

输出: 解向量 x

- 1: 对于原始的线性方程组 $Ax = b$, 参照舒尔补方法, 调用基于 Petal- 剖分的大规模聚类分组算法, 将系数矩阵 A 分解为 (3.1) 中的结构, 包括 k 个内部子区域 ($D_i, i = 1, 2, \dots, k$), 以及相应的 C, E_i, F_i . 每个内部子区域的大小和分布, 遵从于 Petal- 剖分自然形成的聚类分组. 相应地, b 分解为 f_i 和 g , 记录 x_i 和 u_i 的对应关系.
 - 2: 对于超级计算机上的处理器之间的连接关系与通信时长形成权重图, 采用 Petal- 剖分算法进行聚类分组, 将处理器群组分解为 m 个子区域 ($g_j, j = 1, 2, \dots, m$, 这里调用 Petal- 剖分算法进行多级聚类分组, 使得 $m \geq k$). 每个子区域的大小和分布, 遵从于 Petal- 剖分自然形成的聚类分组.
 - 3: 在 m 个处理器子区域中选择 k 个子区域, 将子区域中的处理器资源分配到系数矩阵的子区域 D_i 上, 选择过程中保持子区域中的处理器数量与系数矩阵子区域的规模为一定比例.
 - 4: 舒尔补子系统的处理器群组 (g_s) 分配策略, 同样基于通信代价极小化的考虑, 依据 Petal- 剖分策略, 从处理器群组 g_j 中抽取若干处理器构成群组 g_s , 使得处理器网络关联图的通信开销, 能够达到近似优化的量级.
 - 5: 计算 S 和 \hat{g} , 其中 $S = C - \sum_{i=1}^k F_i D_i^{-1} E_i$, $\hat{g} = g - \sum_{i=1}^k F_i D_i^{-1} f_i$.
 - 6: 求解舒尔补子系统 $Sy = \hat{g}$. 我们采用了 PETSc 的 FGMRES 并行求解模块对 $Sy = \hat{g}$ 进行求解, 预条件子采用了 SuperLU_DIST 算法模块.
 - 7: 在每个内部子区域 (D_i) 的范围内, 调用传统的数值计算软件包中的并行程序对 u_k 进行求解: $D_i u_i = f_i - E_i y$, 这里我们调用 PETSc 的 Preconditioned GMRES 并行求解模块.
 - 8: 根据 x_i 和 u_i 的对应关系, 获得原始方程组的解 x_i .
-

4. 数值实验

为了验证 Schur-KS 架构的求解算法的性能, 我们在国家超级计算无锡中心的“神威·太湖之光”^[23] 超级计算机上进行了高可扩展性测试和针对典型实际应用的强可扩展性测试. “神威·太湖之光”为世界上首台运算速度超过十亿亿次的超级计算机, 它采用申威 26010 众

核处理器, 每个处理器由 4 个核组构成, 每个核组包括 1 个主核 (运算控制核心) 和 64 个从核 (核心阵列), 每个处理器芯片配置 32GB 内存, 每核组内存为 8GB. Schur-KS 求解器的编译使用了“神威·太湖之光”上的 MPI 环境下 C 语言编译器、Fortran 语言编译器, 优化选项为 -O2. 所需的 PETSc、BLAS 和 LAPACK 等数学库采用了“神威·太湖之光”提供的高性能数学库. 基于 Schur-KS 架构的线性问题求解算法的测试分为两组, 一组为大规模环境下的高可扩展性测试, 另一组为中小规模环境下的强可扩展性测试, 详细情况如下所示:

4.1. 大规模环境下的高可扩展性测试

为了验证 Schur-KS 架构的求解算法在大规模并行计算环境下的高可扩展性, 我们在超级计算机“神威·太湖之光”上的百万核环境, 针对典型应用的稀疏矩阵结构的测试用例进行了高可扩展性测试, 测试数据的应用背景是美国宇航局航天飞机火箭助推器的应力分析模型 (简称为 Nasasrb)^[24]. 由于原始模型的数据量不足, 我们采用网格加密的方式, 获得了一系列不同规模的数据 (非零元数量分别为 67 万、133 万、267 万、535 万、1070 万、2141 万、4283 万、8567 万), 分别在 (0.8、1.6、3.3、6.6、13.3、26.6、53.2、106.5) 万核环境下调用 Schur-KS 求解器进行数值求解, 统计迭代求解的时间开销. 为了排除偶然误差, 我们将迭代求解的迭代次数固定为 500 次, 多次进行测试以获得计算结果的平均值. 其中迭代求解的时间如下所示:

表 1 Nasasrb 算例迭代求解的时间

核数 (万核)	0.8	1.6	3.3	6.6	13.3	26.6	53.2	106.5
MPI 进程数 (万)	0.8	1.6	3.3	6.6	13.3	26.6	53.2	106.5
非零元数 (万)	67	133	267	535	1070	2141	4283	8567
迭代时间 (秒)	140.4	169.1	193.8	219.1	288.4	313.3	533.0	858.9

由测试结果可以看出, 随着计算任务规模与进程数等比例增加时, 计算耗时逐渐增大. 主要原因是随着进程数的增加, 并行计算所花费的通信时间逐渐增加. 从图表中可以看出, 相对于 0.8 万核环境时的测试结果, 在 1.6、3.3、6.6、13.3 万核时, 迭代求解的时间缓慢增长至 2 倍左右; 在 26.6 万核参与计算时, 处理器资源与矩阵数据规模均扩大到 32 倍, 迭代求解的时间增长至 2.23 倍, 仍然具有较好的性能; 在 53.2 万核参与计算时, 核组与矩阵规模均扩大到 64 倍, 迭代求解的时间增长至 3.8 倍; 当 106.5 万核参与计算时, 核组与矩阵规模均扩大了 128 倍, 迭代求解的时间增长至 6.12 倍. 测试结果说明了大规模并行计算时, Schur-KS 求解器针对该典型算例具有高可扩展性.

4.2. 中小规模环境下的强可扩展性测试

为了验证 Schur-KS 架构的求解算法对于实际应用问题的求解性能, 我们针对五组典型应用问题的稀疏矩阵结构的测试用例, 在“神威·太湖之光”的中小规模环境下 (256、512、1024、2048、4096 核), 进行了强可扩展性测试. 我们首先调用 Schur-KS 架构的求解方法进行测试, 并以 256 核的测试结果为准, 计算并行效率. 然后, 将 Schur-KS 求解器中的大尺度聚类分组、舒尔补等模块关闭后进行测试, 此时相当于直接在整个计算区域上调用 Petsc 求解器进行求解, 计算相应的并行效率. 将两类并行效率进行比较. 在测试过程中, 对 Krylov 子空间迭

代次数、迭代求解的重启参数等进行了固定. 为排除偶然误差, 不同核组下的测试均进行若干次, 取平均数值作为最终结果.

五组典型应用问题的背景:

1) Maxwell 方程在微波领域中的应用^[25,26], 该问题源自于共面波导 (Coplanar Waveguide) 问题中的寄生 (parasitic) 效应的分析, 描述了具有理想电导体 (perfect electric conductor) 边界条件的 2 维麦克斯韦方程的旋度计算 (the curl-curl-operator of a second-order Maxwell's equations with PEC boundary conditions). 数据集简称: Maxwell 方程.

2) Helmholtz 方程在数值天气预报和大气模型中的应用^[27,28], 相似结构的数据还常常出现在半隐式三维欧拉方程组、大气科学中的 nonhydrostatic 模型、hydrostatic 模型或 primitive 模型. 数据集简称: Helmholtz 方程.

3) 基本的运动方程 - 扩散对流反应方程^[29], 描述河流污染、大气污染、核废物污染中污染物质的分布、流体的流动和流体中热的传导等众多物理现象. 数据集简称: 扩散对流反应方程.

4) 基于有限元方法的钢法兰受力分析^[30], 用于模拟连接电缆的钢法兰的受力分析, 采用六面体有限元方法进行离散. 数据名称: Flange

5) 基于 3 维 Reynolds-Averaged Navier-Stokes 方法, 模拟发动机风扇工作时的湍流流动计算, 对风扇的设计进行评估^[31]. 数据名称: Enginefan

在强可扩展性测试中, 并行效率的结果如下所示:

表 2 强可扩展性测试的并行效率结果

数据名称	非零元数 (万)		核数				
			256	512	1024	2048	4096
			MPI 进程数				
			256	512	1024	2048	4096
		求解方法	并行效率 (%)				
Maxwell 方程	494	Schur+Petsc	100	99.9	99.4	90.7	81.3
		直接调用 Petsc	100	106.4	98.2	79.8	56.2
Helmholtz 方程	367	Schur+Petsc	100	99.6	99.6	95.3	95.0
		直接调用 Petsc	100	106.3	92.3	81.7	75.5
扩散对流反应方程	367	Schur+Petsc	100	97.6	96.0	87.0	81.5
		直接调用 Petsc	100	97.2	88.7	78.1	63.3
Flange	234	Schur+Petsc	100	95.2	90.4	79.9	70.2
		直接调用 Petsc	100	101.3	89.8	66.0	44.6
Enginefan	430	Schur+Petsc	100	97.5	97.8	84.5	73.9
		直接调用 Petsc	100	92.6	83.1	68.0	39.7

从上述测试结果可以看出, 在 512、1024 核时 Schur-KS 的求解方式与 Petsc 的并行效率十分接近. Petsc 求解器在 512 核时出现了一些超线性加速比效果, 原因在于处理器内部有部分高速缓存 (Cache), CPU 对高速缓存的读取速度要远高于对内存或硬盘中读取速度. 计算任务分配给各进程后, 有较多的数据或全部数据位于高速缓存内, 高速缓存效应可降低计算耗时. 由高速缓存减少的计算耗时弥补了因数据通信、负载均衡等所造成的额外时间开销.

随着核数的进一步增加,受到通信量等因素的影响,直接调用 Petsc 的求解方式的并行效率下降较多.通过 Schur-KS 架构调用 Petsc 的求解方式比直接调用 Petsc 的求解方式,其并行效率方面有较为明显的效果提升,其强可扩展性也相对较好,随着核数的增加,并行效率的差异逐渐明显.在 4096 核下,通过 Schur-KS 架构调用 Petsc 的求解算法针对五类典型实际应用问题的强可扩展性分别达到了 81.3%、95.0%、81.5%、70.2%、73.9%,比直接调用 Petsc 求解器进行求解的并行效率分别增加了 25.1%、19.5%、18.2%、25.6%、34.2%.测试结果验证了 Schur-KS 架构的求解算法针对上述实际应用问题的求解性能.

求解器性能提升的原因主要有:采用了具有通信极小化特点的大尺度聚类分组、中尺度分解兼顾负载均衡的新型多级剖分策略,使得处理器群组的剖分的整体通信开销达到近似极小化;聚类分组的核心模块 (Petal- 剖分) 具有近似线性的时间复杂度,提高了算法的可扩展性.这些特点使得通过 Schur-KS 架构调用 Petsc 的求解方式比直接调用 Petsc 的求解方式提高了可扩展性.

5. 结 论

当前,超级计算机的计算能力已达 100PF, E 级超算原型机目前已经投入使用,体系结构日趋复杂.数值计算算法要有效使用数百万乃至上千万异构处理器核,对计算进行加速,面临着可扩展性瓶颈.大规模线性问题求解算法能够在 100PF 级计算机上取得更好的可扩展性,除了计算方法层面上的改进和算法计算复杂性的改进之外,除了并程序优化方面的改进和底层调用高效库函数的改进之外,还存在着第三条可扩展性提升的途径:如何更加灵活地、更加有效地利用数值算法的自身特性和并行计算机体系结构的自身特性,以求进一步提升求解算法的可扩展性.第三条途径的实现,需要求解算法能够与并行计算机体系结构之间进行相互耦合、交流与通信.

我们判断,如果大规模线性问题求解算法能够获得并行计算机体系结构自身存在的分层聚类的空间分布非均衡信息,据此指导求解算法进行系数矩阵分块,至少会有正面的帮助.因此,本文针对大规模线性问题迭代求解算法在 100PF 乃至 E 级超级计算机上的可扩展性问题开展研究工作.首先,针对大规模数值模拟求解、异构的硬件系统的特定分层耦合和非均衡特性,以及目前主流的数值计算算法库及其衍生的并程序的特点给出描述,并给出提高大规模线性问题迭代求解算法的可扩展性的策略.其次,针对通讯代价极小化方法进行了理论分析与算法研究,基于 Petal- 剖分技术,给出了大规模聚类分组算法,以近似线性的代价实现了通信开销的近似优化.然后,基于大规模聚类分组算法、舒尔补架构,给出了具有高可扩展性的线性问题求解架构.最后在超级计算机神威·太湖之光上,开展了大规模环境下的高可扩展性测试、中小规模环境下的强可扩展性测试.测试结果验证了基于 Schur-KS 架构的求解算法在大规模环境 (百万核) 下针对稀疏矩阵结构的测试用例的高可扩展性,验证了其在中小规模 (256-4096 核) 下针对五组典型应用问题的强可扩展性.下一步拟将相关的理论应用至大规模分布式深度学习求解架构的设计中.

致谢: 本文受到国家自然科学基金重大研究计划项目重点支持课题 - 面向 100PF 级计算机的三类共性算法研究及高效实现 (91430214) 和国家自然科学基金面上项目 - 面向新模型的深度神经网络求解器的共性组件关键技术研究: 算法与性能提升 (61876175) 资助,并感谢 University of Florida 提供的数据集资源.

参 考 文 献

- [1] 汲培文, 江松, 张平文. 可计算建模 [J]. 中国科学, 2012, 42(6).
- [2] Saad Y. Iterative methods for sparse linear systems (2nd edition)[M]. SIAM, 2003.
- [3] Metis[CP]. <http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>.
- [4] Cai Xiao-Chuan. David E. Keyes. Nonlinearly Preconditioned Inexact Newton Algorithms[J]. SIAM J. Sci. Comput., 2002, 24(1): 183–200.
- [5] Abraham I, Bartal Y and Neiman O. Using petal-decompositions to build a low stretch spanning tree[J]. in In Proceedings of the 44th Annual ACM Symposium on the Theory of Computing STOC, 2012, 395–406.
- [6] 张慧荣. 大型稀疏线性方程组的新型预条件算法研究 [D]. 中国科学院大学, 2017.
- [7] Pal Andras Papp, Low-Stretch Spanning Trees[D]. Eotvos Lorand University, 2014.
- [8] hmetis[CP]. <http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview>.
- [9] parmetis[CP]. <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>.
- [10] SCOTCH[CP]. <http://www.labri.fr/perso/pelegrin/scotch>.
- [11] Tom Leighton, Satish Rao, Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms[J]. J. ACM, 1999, 46(6): 787–832.
- [12] Arora S, Rao S, Vazirani U. Expander flows, geometric embeddings and graph partitioning[J]. J. ACM, 2009, 56(2): 5:1–5:37.
- [13] Sherman S. Breaking the multicommodity flow barrier for $o(\log n)$ -approximations to sparsest cut[J]. in Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science, ser. FOCS '09. Washington, DC, USA: IEEE Computer Society, 2009, 363–372.
- [14] Alon N, Karp R M, Peleg D and West D. A graph theoretic game and its application to the k server problem[J]. In Proceedings of the 44th Annual ACM Symposium on the Theory of Computing STOC, 1995, 24: 78–100.
- [15] Elkin M, Emek Y, Spielman D A and Teng S H. Lower-stretch spanning trees[J]. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005, 494–503.
- [16] Andreas Crauser, Kurt Mehlhorn, Ulrich Meyer and Peter Sanders. A Parallelization of Dijkstra's Shortest Path Algorithm. Lecture Notes in Computer Science[J]. 1998, 1450: 722–731.
- [17] Parallel Boost Graph Library[CP]. https://www.boost.org/doc/libs/1.67.0/libs/graph_parallel/doc/html/index.html.
- [18] Yamazaki I, Li X S. On techniques to improve robustness and scalability of the Schur complement method[J]. Proc. of VECPAR 2010, 2011: 421–434.
- [19] Giraud L, Haidar A, Watson L T. Parallel scalability study of hybrid preconditioners in three dimensions[J]. Parallel Computing, 2008, 34: 363–379.
- [20] Dominique LaSalle, George Karypis. Efficient Nested Dissection for Multicore Architectures[J]. Euro-Par 2015: Parallel Processing, 2015, 467–478.
- [21] Ichitaro Yamazaki, Xiaoye S. Li, Francois-Henry Rouet, Bora Ucar. On Partitioning and Reordering Problems in a Hierarchically Parallel Hybrid Linear Solver[C]. 2013 IEEE 27th International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), May 2013, Cambridge, MA, United States. IEEE Computer Society.
- [22] Petsc[CP]. <http://www.mcs.anl.gov/petsc/>.
- [23] 神威太湖之光 [OL]. <http://www.nscwx.cn/>.
- [24] Duc Thai Nguyen. Parallel-Vector Equation Solvers for Finite Element Engineering Applica-

- tions[M]. Springer Science, Business Media, 2012.
- [25] Bai. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems[J]. *Applied Numerical Mathematics*, 2002, 43: 9–44.
- [26] Hiptmair R. Finite elements in computational electromagnetism[J]. *Acta Numerica*, Cambridge University Press, 2002, 237–339.
- [27] J. SteppelerR. HessU. SchättlerL. Bonaventura. Review of numerical methods for nonhydrostatic weather prediction models[J]. *Meteorology and Atmospheric Physics*, 2003, 82: 287–301.
- [28] Yeh K S, Cote J, Gravel S, Methot A, Patoine A, Roch M, Staniforth A. The CMC-MRB global environmental multiscale (GEM) model. Part III: Nonhydrostatic formulation[J]. *Monthly Weather Review*, 2002, 130: 339–356.
- [29] João Pedro Lopes. Convection, Diffusion and Reaction: Bridging Scales in Science and Engineering[D]. UNIVERSITY OF PORTO, 2011.
- [30] Janna C, Comerlati A, Gambolati G. A comparison of projective and direct solvers for finite elements in elastostatics[J]. *Advances in Engineering Software*, 2009, 40(8): 675–685.
- [31] Pacull F, Aubert S, Buisson M. A Study of ILU Factorization for Schwartz Preconditioners with Application to Computational Fluid Dynamics[C]. *Proceedings of the 2nd Intl Conf on Parallel, Distributed, Grid, and Cloud Computing for Engineering*, 2011, Paper 39.