有限元结构分析的层级负载均衡并行计算方法

苗新强^{1,2}, 金先龙^{1,2*}, 丁峻宏³

1. 上海交通大学机械系统与振动国家重点实验室, 上海 200240;

2. 上海交通大学机械与动力工程学院, 上海 200240;

3. 上海超级计算中心, 上海 201203

* 联系人, E-mail: jxlong@sjtu.edu.cn

2016-08-17 收稿, 2016-09-19 修回, 2016-09-20 接受, 2017-03-28 网络版发表

国家高技术研究发展计划(2012AA01A307)、国家自然科学基金(11272214, 51475287)和国家重点研发计划(2016YFB0201800)资助

摘要 由于性价比高、计算能力强, 多核机群已经成为当今高性能计算的主流工具. 然而, 多核机群环境下不同的存储机制和通信延迟特点也为高效并行算法的设计带来了挑战. 为充分利用多核机群的硬件资源获取最优性能, 本文设计了一种有限元结构分析的层级负载均衡并行计算方法. 该方法建立在对计算任务的层次性和粒度性充分挖掘的基础上. 为与多核机群的硬件拓扑体系结构相适应, 本文将计算任务划分为三个层次: 节点间并行、片间并行和核间并行. 其中, 节点间并行和片间并行采用粗粒度并行计算方法, 而核间并行采用细粒度并行计算方法. 通过将计算任务映射到多核机群的不同硬件层面执行, 该方法不仅有效实现了不同层面的负载均衡, 而且大幅度降低了系统的通信开销. 此外, 它还大幅度减少了子区域的数目, 有效提高了界面方程的数值收敛性. 为验证算法的有效性, 在“天河二号”超级计算机上进行了有限元结构线性静力分析大规模并行计算测试. 结果表明: 同传统区域分解法相比, 层级负载均衡并行计算方法能够获得较高的加速比和并行效率. 本文的研究主要集中在线性静力学问题上. 对于非线性问题或者动力学问题, 由于涉及多个迭代步, 因此可以将本文算法封装为一个子函数进行调用.

关键词 多核机群, 有限元分析, 并行计算, 负载均衡

直接法和迭代法是有限元结构分析并行计算的两种基本方法. 直接法通过排序、三角分解和回代求解等能够给定线性系统的精确解, 并且具有良好的数值稳健性^[1~3]. 然而, 它往往需要较大的内存空间^[4]. 迭代法是基于试验和误差的方法, 它通过多次迭代求解对结果进行改善以尝试收敛到问题的精确解^[5,6]. 相对直接法来说, 迭代法具有所需内存空间小的优点. 但迭代法不一定在合理的时间内收敛, 对条件数很大的病态问题也可能是不收敛的^[7]. 区域分解法可以融合直接法和迭代法求解的优点, 使并行计算既具有良好的数值稳健性, 又具有所需内存空间小的优点^[8~10]. 它首先将结构有限元网格划分为若干个子区域, 然后

利用直接法消去每个子区域的内部自由度, 再通过迭代法求解界面方程, 最后根据求得的边界位移回代各子区域内部自由度. 利用区域分解法进行有限元结构分析并行计算时, 各子区域仅在求解界面方程的过程中需要相互通信, 而其他计算过程均可在各子区域内独立完成. 因此, 在传统单核处理器构建的超级计算机上, 区域分解法能够获得较高的并行计算效率.

随着处理器和超级计算机体系结构的发展, 多核机群日益成为大规模科学计算和工程分析的主流工具^[11]. 一方面多核机群通过在单个节点内集成更多的计算核心大幅度提升了系统的计算性能, 另一方面它在节点内和节点间不同的存储机制和通信延

引用格式: 苗新强, 金先龙, 丁峻宏. 有限元结构分析的层级负载均衡并行计算方法. 科学通报, 2017, 62: 1430–1438

Miao X Q, Jin X L, Ding J H. A hierarchical load balancing parallel computing approach for finite element structural analysis (in Chinese). Chin Sci Bull, 2017, 62: 1430–1438, doi: 10.1360/N972016-00607

迟差异也为高效并行算法的设计带来了挑战^[12]。传统区域分解法在进行区域分割时没有考虑多核机群的体系结构特点,它采用直接剖分的方法实现负载均衡,即直接根据参与并行计算的处理器核的数目将结构剖分为与之相等的若干个子区域。由于多核机群单个节点内处理器核数目的成倍增加,这就导致子区域的数目会随着处理器核数目的增加而大幅度增加。子区域数目的大幅度增加一方面会导致界面方程的规模和条件数急剧扩大,从而降低系统的数值收敛性;另一方面会导致参与并行计算的进程数目大幅度增加,从而加剧对网络端口和网络带宽的竞争。此外,传统区域分解法没有考虑多核机群节点内和节点间不同的存储机制和通信延迟差异,造成系统的通信开销较大。系统数值收敛性的降低和网络通信开销的增加严重影响了界面方程的求解效率,并大幅度降低了区域分解法的整体并行效率。为充分利用多核机群的硬件资源获取最优性能,本文设计了一种有限元结构分析的层级负载均衡并行计算方法。它在区域分解法的基础上,通过将并行算法与多核机群的硬件拓扑体系结构融合在一起有效提高了系统的并行计算效率。

1 多核机群与区域分解法

1.1 多核机群

如图1所示,多核机群由多核计算节点通过高速网络相互连接构成^[13]。它的每个计算节点都配置有两个或多个多核处理器。节点内的内存和高速缓存基于树形层次化结构进行组织,存储器的大小从叶节点到根节点逐渐增大。每个计算核心都配置有尺

寸较小的二级缓存,位于相同处理器芯片上的所有计算核心共享尺寸较大的三级缓存,位于相同节点内的所有计算核心共享尺寸最大的系统主存。在各个节点间,多核机群采用分布式存储的体系结构,要访问其他节点的内存必须通过消息进行传递。可见,多核机群具有两级存储机制:节点内共享存储和节点间分布式存储。

从通信层面看,多核机群具有3层通信模式:片内通信、片间通信和节点间通信^[14]。相同芯片上不同计算核心间的通信称为片内通信;相同节点内不同芯片的计算核心间的通信称为片间通信;不同节点的通信称为节点间通信。由于通信通道和机制的不同,多核机群上的通信延迟是非均匀的:片内通信延迟最小,片间通信延迟稍大,而节点间通信延迟最大。因此,要提高多核机群的并行计算效率就要设法减少不同模式下通信延迟差异对应用程序性能造成的影响。

1.2 区域分解法

区域分解法是一种粗粒度的并行计算方法。它既适合运行在具有分布式存储体系结构的超级计算机上,又适合运行在具有共享存储体系结构的超级计算机上。利用区域分解法进行有限元结构分析并行计算的基本原理如下^[15]。

首先将结构有限元网格划分为若干个子区域,然后按照先内部自由度后边界自由度的编号原则同时独立形成每个子区域的系统方程:

$$\begin{bmatrix} K_{II} & K_{IB} \\ K_{BI} & K_{BB} \end{bmatrix} \begin{Bmatrix} x_I \\ x_B \end{Bmatrix} = \begin{Bmatrix} P_I \\ P_B \end{Bmatrix}, \quad (1)$$

式中, x_I , x_B , P_I , P_B 分别为内部自由度和边界自由度对

图1 多核机群体系结构

Figure 1 Architecture of multi-core cluster

应的位移和外部载荷向量; K_{**} 为刚度矩阵的分块矩阵, 其中下标I和B分别代表内部自由度和边界自由度.

接着通过缩聚同时独立消去各子区域内部自由度, 得到只含边界自由度未知量的界面方程:

$$\tilde{K}x_B = \tilde{P}, \quad (2)$$

式中, 有效刚度矩阵

$$\tilde{K} = K_{BB} - K_{BI}(K_{II})^{-1}K_{IB}. \quad (3)$$

有效载荷向量

$$\tilde{P} = P_B - K_{BI}(K_{II})^{-1}P_I. \quad (4)$$

然后将所有子区域的界面方程联立求解, 得到各子区域边界位移 x_B . 本文采用并行预条件共轭梯度算法求解界面方程^[16]. 求得边界位移后, 各子区域内部位移 x_I 可由下式回代求解:

$$x_I = (K_{II})^{-1}(P_I - K_{IB}U_B). \quad (5)$$

最后同时独立计算各子区域的应变和应力, 并输出计算结果.

由以上过程可见, 区域分解法仅在求解界面方程的过程中需要通信, 而其他计算过程如形成子区域系统方程、缩聚和回代内部自由度等均可在各子区域内部独立完成. 因此, 当子区域数目较少时, 界面方程的求解开销较小, 系统能够获得较高的并行计算效率. 然而, 利用多核机群进行大规模并行计算时, 子区域的数目会随着处理器核数目的成倍增加而大幅度增加, 从而严重影响界面方程的求解效率, 并大幅度降低系统整体的并行效率.

2 层级负载均衡并行计算方法

从硬件体系结构看, 多核机群由多核、多处理器和多节点的层次性元素构成. 如1.1节所述, 位于不同层次元素在数据存储和通信延迟上都有很大差异. 因此, 要提高多核机群的并行计算效率就要实现不同层面的负载均衡和最小化系统的通信开销. 本文在区域分解法的基础上将并行算法与多核机群的硬件拓扑体系结构融合在一起, 提出了一种有限元结构分析的层级负载均衡并行计算方法.

层级负载均衡并行计算方法建立在对计算任务的层次性和粒度性充分挖掘的基础上. 为与多核机群的硬件拓扑体系结构相适应, 本文将有限元结构分析的计算任务划分为3个层次: 节点间并行、片间并行和核间并行. 其中, 节点间并行和片间并行采用粗粒度并行计算方法, 而核间并行采用细粒度并行计算方法. 如图2所示, 通过将计算任务映射到多核机群的不同硬件层面执行, 层级负载均衡并行计算方法不仅有效实现了不同层面的负载均衡, 而且大幅度降低了系统的通信开销. 此外, 它还大幅度减少了子区域的数目, 有效提高了界面方程的数值收敛性. 因此, 它能够充分利用多核机群的硬件资源获取最优性能.

2.1 节点间并行

考虑到多核机群在节点间采用分布式存储体系结构, 并且节点间的通信延迟远远大于节点内的通

图2 任务映射

Figure 2 Task mapping

信延迟,因此节点间适合采用粗粒度并行计算方法.这样一方面能够实现大规模数据的分布式存储,另一方面能够有效减少系统的通信开销.

如图2所示,节点间并行基于区域分解法实现:首先将结构有限元网格均等剖分为 M 个一级子区域,然后将每个一级子区域分别分配给一个节点机单独处理.其中, M 为参与并行计算的节点机总数.由于各一级子区域的规模都相同,并且各节点机具有相同的计算性能,这就实现了节点间的负载均衡和并行计算.

2.2 片间并行

为提高数据访问速度,现代超级计算机一般都配置了二级和三级缓存.由于位于不同处理器芯片上的三级缓存是相对独立的,并且片间通信开销比片内通信开销稍大,因此片间也适合采用粗粒度并行计算方法.为尽量将不同处理器芯片间的通信限制在同一节点内,应将相邻的子区域分配在一起.

如图2所示,片间并行建立在上文一级分区的基础上:首先将每个一级子区域进一步均等剖分为 N 个二级子区域,然后将派生于相同一级子区域的各二级子区域分别分配给相同节点的不同处理器芯片单独处理.其中, N 为单个节点机内处理器芯片的总数.由于位于相同节点的各二级子区域均由同一个一级子区域派生而来,这就将相邻的子区域分配在了一起,从而将不同处理器芯片间的通信尽量限制在了同一节点内.

通过两级分区就将直接参与并行计算的子区域的数目降低为传统区域分解法的 $1/T$.其中, T 为单个处理器芯片上的计算核心总数.子区域数目的大幅度降低一方面有利于减少界面方程的规模和条件数,从而有效提高系统的数值收敛性;另一方面有利于大幅度减少参与并行计算的进程数目,从而减少通信对有限的网络端口和带宽的竞争,进而提高系统的通信效率.

2.3 核间并行

随着多核处理器的发展,在单个芯片上集成的计算核心数目也越来越多.若采用粗粒度并行计算方法利用这些多核资源,难免会面临因分区过多而导致系统数值收敛性降低和通信开销增加的问题.因此,核间并行适合采用细粒度并行计算方法.这样

一方面有利于提高系统的数值收敛性,另一方面可以利用单个芯片上所有计算核心共享三级缓存的特点有效提高数据的访问速度和通信效率.

如图2所示,核间并行涉及最底层的计算模块,是对每个二级子区域计算任务的进一步分解.基于区域分解法进行有限元结构分析并行计算的主要步骤包括:组集子区域系统平衡方程、缩聚、求解界面方程、回代内部自由度以及计算子区域应变和应力.核间并行要做的工作就是针对二级子区域的每个计算步骤寻找出里面的热点计算程序,即大的循环结构,将其分解为大量互不相干且能够独立执行的子任务,然后将每个子任务分配给多核处理器的一个计算核心去执行.下面以缩聚为例介绍核间并行的实现.

缩聚是利用区域分解法进行并行计算的一个核心环节,它通常需要消耗大量的计算时间.为了充分利用多核资源缩短计算时间,本文将缩聚计算转换为一系列相互独立的线性方程组的求解过程,并通过OpenMP进行并行化处理以将每个子任务分配给多核处理器的一个计算核心去执行.

由式(3)可见,缩聚刚度矩阵 $\tilde{K}$ 的计算需要用到 $(K_{II})^{-1}K_{IB}$ 的结果.为了得到这两个矩阵相乘的每一列的值,可将 K_{IB} 的元素逐列提取出来并赋值给向量 b ,然后再通过稀疏直接求解器依次求解线性方程组 $K_{II}t_1=b$ 的解.在 t_1 的基础上, $K_{BI}(K_{II})^{-1}K_{IB}$ 每一列的结果可通过计算 $t_2=K_{BI}t_1$ 得到.最后,将 K_{BB} 相应列的元素提取出来并与 t_2 相减就可得到缩聚刚度矩阵 $\tilde{K}$ 对应列的计算结果.考虑到缩聚刚度矩阵的列数与子区域边界自由度的个数 z 相等,因此上述操作过程共需重复执行 z 次.对每次操作来说,它都是一个能够独立执行的子任务,因此可以通过OpenMP实现对这些子任务的并行化处理.同理,缩聚外部载荷向量 $\tilde{P}$ 也可采取同样的策略求解.缩聚外部载荷向量只有1列,在进程里操作1次即可.本文设计的缩聚算法的伪代码如图3所示,图中2~9行为通过OpenMP利用多核资源进行并行计算的部分.图中, $K_{IB}(:,i)$, $\tilde{K}(:,i)$ 和 $K_{BB}(:,i)$ 分别表示相应矩阵的第 i 列元素构成的列向量; z 为二级子区域边界自由度的个数; b , t_1 , t_2 均为临时列向量.

2.4 层级负载均衡并行计算流程

有限元结构分析的层级负载均衡并行计算方法在编程模式上采用MPI+OpenMP混合编程模式.整

1. //calculate the condensed stiffness matrix
2. !\$omp parallel do
3. do $i = 1, z$
4. $\hat{b} = K_m(:, i)$
5. solve $K_{\alpha} t_1 = \hat{b}$ with a direct sparse solver
6. compute $t_2 = K_m t_1$
7. compute $\hat{K}(:, i) = K_m(:, i) - t_2$
8. end do
9. !\$omp end parallel do
10. //calculate the condensed load vector
11. $\hat{b} = P_i$
12. solve $K_{\alpha} t_1 = \hat{b}$ with a direct sparse solver
13. compute $t_2 = K_m t_1$
14. compute $\hat{P} = P_n - t_2$
15. //solve interface equations
16. solve $\hat{K} x_n = \hat{P}$ with the parallel PCG algorithm
17. //calculate internal degrees of freedom
18. $\hat{b} = P_i - K_m x_n$
19. solve $K_{\alpha} x_i = \hat{b}$ with a direct sparse solver

图3 缩聚算法

Figure 3 Condensation algorithm

个并行计算的顶层基于MPI进程构建, 其中每个MPI进程负责控制1个二级子区域的处理. 并行计算的底层基于OpenMP线程实现, 通过在每个MPI进程内部派生大量线程以充分发挥多核处理器高度并发计算的优势. 如图4所示, 有限元结构分析的层级负载均衡并行计算流程为:

第1步, 通过两级分区生成并行计算所需的数据文件, 包括各子区域的单元、节点、载荷、边界条件和相邻分区信息等;

第2步, 同时在每个节点机内启动 N 个MPI进程, 其中每个进程负责1个二级子区域数据文件的读入;

第3步, 每个MPI进程内部分别派生出 T 个线程, 利用多核资源完成相应子区域总体刚度矩阵和外部载荷向量的计算. 其中, T 为单个处理器芯片上的计算核心总数;

第4步, 每个MPI进程内部分别派生出 T 个线程, 利用多核资源完成缩聚计算;

第5步, 各进程共同利用并行预条件共轭梯度算法求解界面方程, 得到各子区域边界节点位移后再回代求解内部位移;

第6步, 每个MPI进程内部分别派生出 T 个线程, 利用多核资源计算相应子区域应变/应力;

第7步, 若需要进一步迭代则跳转至第2步重新开始执行, 否则结束.

综上所述, 层级负载均衡并行计算方法一方面

图4 层级负载均衡并行计算流程

Figure 4 Flowchart of parallel computing based on hierarchical load balancing approach

通过两级分区大幅度降低了子区域的数目, 有效提高了界面方程的数值收敛性; 另一方面, 它通过将计算任务映射到多核机群的不同硬件层面执行, 不仅有效实现了不同层面的负载均衡, 而且大幅度降低了系统的通信开销. 因此, 它能够充分利用多核机群的硬件资源有效提高有限元结构分析大规模并行计算的效率.

3 数值算例

为了验证本文算法的有效性, 在广州超算中心的“天河二号”超级计算机上进行了测试. 并行算法评估的数值模型如图5和6所示. 图5为左端固定, 右端受竖直向下载荷作用的悬臂梁计算模型. 梁长5 m, 宽0.6 m, 高0.8 m. 采用四面体单元进行网格剖分后, 该模型具有35898476单元, 6846069节点, 20538207自由度.

图6为某拱桥计算模型, 主要分析该模型在重力和车载作用下的变形和应力. 模型总长16 m, 总宽10 m, 总高5.4 m. 采用四面体单元进行网格剖分后, 该模型具有44081196单元, 8461996节点, 25385988自由度.

图5 悬臂梁计算模型
Figure 5 Computational model of cantilever

图6 拱桥计算模型
Figure 6 Computational model of arch bridge

3.1 并行计算环境

“天河二号”超级计算机采用目前国际主流的机群架构构建。它由16000个计算节点组成，每个节点配置2颗基于Ive Bridge-E Xeon E5 2692多核处理器芯片和64 GB内存。每颗多核处理器芯片具有12个计算核心，运作时钟频率为2.2 GHz，峰值性能为0.2112 TFLOPS。每个计算核心都配置有独立的256 KB二级缓存，位于相同处理器芯片上的所有计算核心共享30 MB三级缓存。在节点内不同处理器核间通过片上网络或高速缓存互联，通信速度很快。而各节点间采用TH Express-2主干拓扑结构网络联接，通信速度相对较慢。

3.2 计算数据准备

并行计算前首先需要通过两级分区准备数据。根据前面2.1和2.2节的内容知，为与多核机群的硬件

拓扑体系结构相适应两级分区时一级子区域总数应等于每次启动的节点机总数，并且每个一级子区域的派生二级子区域总数应等于单个节点内处理器芯片的总数。本次测试中，每次启动的节点机总数依次为50, 100, 150, 200，因此经一级剖分得到的一级子区域总数也应依次为50, 100, 150, 200。由于“天河二号”超级计算机每个节点内配置2颗多核处理器芯片，因此二级剖分时每个一级子区域将被进一步剖分为2个二级子区域。

3.3 并行算法性能评估

对上述各计算模型分别采用传统区域分解法和本文层级负载均衡并行算法进行求解，得到的测试结果如表1和2所示。并行计算总时间从读取数据文件开始到计算各子区域应变/应力结束；缩聚时间指消去各子区域内部自由度的时间；求解界面方程时间指利用并行预条件共轭梯度算法求得各子区域边界位移的时间。

由表1和2可见，相对传统区域分解法本文提出的层级负载均衡并行计算方法能够获得较高的加速比和并行效率。采用传统区域分解法利用大量处理器内核进行并行计算时，随着子区域数目的增多界面方程求解时间也大幅度增加，从而严重影响了系统总体的并行效率。例如在表2中，核数由1200增加到4800的过程中，传统区域分解法的界面方程求解时间由1373 s迅速增加到了2202 s，增加幅度高达829 s。这就在很大程度上直接造成了系统总体并行效率由100%下跌到了17.34%。界面方程求解时间的大幅度增加主要由两个因素导致：首先随着子区域数目的大幅度增加，界面方程的规模和条件数也急剧增大从而降低了系统的数值收敛性；其次，传统区域分解法没有考虑多核机群节点内和节点间不同的存储机制和通信延迟差异，造成系统的通信开销较大。

表1 悬臂梁模型并行计算时间和性能统计

Table 1 Statistics of time and performance of parallel computing for cantilever model

核数	传统区域分解法					层级负载均衡并行算法				
	缩聚(s)	求解 界面方程(s)	总时间(s)	加速比	并行效率 (%)	缩聚(s)	求解 界面方程(s)	总时间(s)	加速比	并行效率(%)
1200	112	714	837	1	100	565	171	809	1	100
2400	30	951	986	0.85	42.44	218	165	417	1.94	97.00
3600	11	1159	1174	0.71	23.76	123	176	316	2.56	85.34
4800	6	1260	1268	0.66	16.50	84	187	281	2.88	71.98

表 2 拱桥模型并行计算时间和性能统计

Table 2 Statistics of time and performance of parallel computing for arch bridge model

核数	传统区域分解法					层级负载均衡并行计算法				
	缩聚(s)	求解 界面方程(s)	总时间(s)	加速比	并行效率 (%)	缩聚(s)	求解 界面方程(s)	总时间(s)	加速比	并行效率(%)
1200	153	1373	1535	1	100	895	318	1303	1	100
2400	40	1631	1677	0.92	45.77	332	286	659	1.98	98.86
3600	18	2053	2074	0.74	24.67	172	294	491	2.65	88.46
4800	9	2202	2213	0.69	17.34	95	321	428	3.04	76.11

采用本文层级负载均衡并行计算方法进行有限元结构分析大规模并行计算时能够大幅度缩减界面方程求解时间,并取得良好的加速比和并行效率.例如在表2中,核数为3600时传统区域分解法和层级负载均衡并行计算方法的界面方程求解时间分别为2053和294 s,后者比前者减少了1759 s.此时,传统区域分解法的加速比和并行效率仅为0.74%和24.67%,而层级负载均衡并行计算方法的加速比和并行效率则高达2.65%和88.46%.这主要归功于两个方面的因素.(1)同传统区域分解法相比,本文设计的两级分区方法大幅度减少了子区域的数目.子区域数目的大幅度降低一方面有利于减少界面方程的规模和条件数,从而有效提高系统的数值收敛性;另一方面有利于大幅度减少参与并行计算的进程数目,从而减少通信对有限的网络端口和带宽的竞争,进而提高系统的通信效率.(2)本文算法通过将计算任务映射到多核机群的不同硬件层面执行不但有效实现了不同层面的负载均衡,而且大幅度降低了系统的通信开销.因此,层级负载均衡并行计算方法能够大幅度减少界面方程求解时间以及系统求解总时间,从而有效提高系统总体的并行计算效率.

本文算法具有广泛的通用性,它能够有效求解各种有限元结构分析问题,包括线性静力分析、线性动力分析、非线性静力分析和非线性动力分析等.本文的研究主要集中在线性弹性力学问题分析上.对于非线性问题,由于涉及多个迭代步求解,因此可以将本文算法封装为一个子函数进行调用.在每个迭代步的求解过程中,同传统区域分解法相比本文算

法都能大幅度减少界面方程求解时间,有效提高系统的加速比和并行效率.因此,经过多个迭代步求解之后,本文算法累计削减的界面方程求解时间会更加明显,本文算法相对传统区域分解法的优势也将更加突出.下一步的工作致力于将本文算法推广应用到求解非线性问题上.

4 结论

随着处理器和超级计算机体系结构的发展,多核机群日益成为大规模科学计算和工程分析的主流工具.然而,多核机群节点内和节点间不同的存储机制和通信延迟差异为高效并行算法的设计带来了挑战.本文根据多核机群的硬件拓扑体系结构特点,设计了一种有限元结构分析的层级负载均衡并行计算方法以提高系统的并行效率.

该方法将计算任务划分为3个层次以与多核机群的硬件拓扑体系结构特点相适应:节点间并行、片间并行和核间并行.其中,节点间并行和片间并行采用粗粒度并行计算方法,而核间并行采用细粒度并行计算方法.通过将计算任务映射到多核机群的不同硬件层面执行,层级负载均衡并行计算方法不仅有效实现了不同层面的负载均衡,而且大幅度降低了系统的通信开销.此外,它还大幅度减少了子区域的数目,有效提高了界面方程的数值收敛性.为验证算法的有效性,在“天河二号”超级计算机上进行了有限元结构分析大规模并行计算的测试.结果表明:同传统区域分解法相比,层级负载均衡并行计算方法能够获得较高的加速比和并行效率.

参考文献

- 1 Paszynski M, Pardo D, Calo V M. A direct solver with reutilization of LU factorizations for h-adaptive finite element grids with point singularities. *Comput Math Appl*, 2013, 65: 1140–1151

- 2 Wang S, Xia J L, de Hoop M V, et al. Massively parallel structured direct solver for equations describing time-harmonic qP-polarized waves in TTI media. *Geophysics*, 2012, 77: 69–82
- 3 Gupta A. A shared- and distributed-memory parallel general sparse direct solver. *Algebra Eng Commun Comput*, 2007, 18: 263–277
- 4 Paszyński M, Pardo D, Torres-Verdín C, et al. A parallel direct solver for the self-adaptive hp finite element method. *J Parallel Distr Com*, 2010, 70: 270–281
- 5 Ullmann E, Elman H, Ernst O. Efficient iterative solvers for stochastic Galerkin discretizations of log-transformed random diffusion problems. *SIAM J Sci Comput*, 2012, 34: 659–682
- 6 Yao J. An easy method to accelerate an iterative algebraic equation solver. *J Comput Phys*, 2014, 267: 139–145
- 7 Mitin I, Kalinkin A, Laevsky Y. A parallel iterative solver for positive-definite systems with hybrid MPI-OpenMP parallelization for multi-core clusters. *J Comput Sci*, 2012, 3: 463–468
- 8 Wang J W, Jin X L, Cao Y. Parallel computing for the train-structure dynamic coupling analysis (in Chinese). *Chin J Comput Mech*, 2012, 29: 352–356 [王建伟, 金先龙, 曹源. 列车与结构动态耦合分析的并行计算方法. *计算力学学报*, 2012, 29: 352–356]
- 9 Xicheng W, Baggio P, Schrefler B. A multi-level frontal algorithm for finite element analysis and its implementation on parallel computation. *Eng Comput*, 1999, 16: 405–427
- 10 Yang Y S, Hsieh S H, Asce M, et al. Improving parallel substructuring efficiency by using a multilevel approach. *J Comput Civ Eng*, 2012, 26: 457–464
- 11 Zavala-Díaz J C, Pérez-Ortega J, Salazar-Reséndiz E, et al. Matrix multiplication with a hypercube algorithm on multi-core processor cluster. *Dyna*, 2015, 82: 240–246
- 12 Dongarra J, Faverge M, Herault T, et al. Hierarchical QR factorization algorithms for multi-core clusters. *Parallel Comput*, 2013, 39: 212–232
- 13 Seelam S, Fong L, Tantawi A, et al. Extreme scale computing: Modeling the impact of system noise in multi-core clustered systems. *J Parallel Distr Com*, 2013, 73: 898–910
- 14 Hamid N, Walters R, Wills G. Understanding the impact of the interconnection network performance of multi-core cluster architectures. *J Comput*, 2016, 11: 132–139
- 15 Antoine X, Lorin E, Bandrauk A D. Domain decomposition method and high-order absorbing boundary conditions for the numerical simulation of the time dependent Schrödinger equation with ionization and recombination by intense electric field. *J Sci Comput*, 2015, 64: 620–646
- 16 Rao A. MPI-based parallel finite element approaches for implicit nonlinear dynamic analysis employing sparse PCG solvers. *Adv Eng Softw*, 2005, 36: 181–198

Summary for “有限元结构分析的层级负载均衡并行计算方法”

A hierarchical load balancing parallel computing approach for finite element structural analysis

MIAO XinQiang^{1,2}, JIN XianLong^{1,2*} & DING JunHong³

¹ State Key Laboratory of Mechanical System and Vibration, Shanghai Jiaotong University, Shanghai 200240, China;

² School of Mechanical Engineering, Shanghai Jiaotong University, Shanghai 200240, China;

³ Shanghai Supercomputer Center, Shanghai 201203, China

* Corresponding author, E-mail: jxlong@sjtu.edu.cn

Multi-core clusters have become primary tools for high performance computing due to their great computing power and cost-to-performance effectiveness in nowadays. However, it introduces new challenges for the design of efficient parallel algorithms because of the different storage mechanisms and non-uniform communication latencies on these machines. The traditional domain decomposition methods use the direct partition method to achieve load balancing, which directly divides the structure into a number of subdomains with equal according to the number of processing cores involved in parallel computing. As the number of processing cores in a single node of multi-core clusters increases exponentially, the number of subdomains will increase dramatically as well. A substantial increase in the number of subdomains leads to the rapid expansion of the size and the condition number of interface equations, thereby reducing the numerical convergence of the system. In addition, it leads to a considerable increase in the number of processes involved in parallel computing, thereby increasing contention for the limited network ports and bandwidth. The decrease of the numerical convergence and the increase of network communication overheads seriously affect the solution efficiency of interface equations, and greatly reduce the overall parallel efficiency of the domain decomposition method. In order to make full use of the computing power of multi-core clusters to improve the parallel efficiency of large-scale finite element structural analysis, a hierarchical load balancing approach is proposed in the paper. The proposed approach is based on the full mining of computational tasks. In order to adapt to the hardware topology architecture of multi-core clusters, the computational tasks of finite element structural analysis are divided into three layers: inter-node parallelism, inter-chip parallelism and inter-core parallelism. The coarse grain parallel computing method is utilized in inter-node parallelism and inter-chip parallelism, and the fine grain parallel computing method is used in inter-core parallelism. Through mapping computing tasks to different hardware layers of multi-core clusters, the proposed method not only efficiently achieves the load balancing at different layers, but also greatly reduces the communication overheads of the system. Furthermore, it considerably reduces the number of subdomains and significantly improves the numerical convergence of the interface equations. In order to verify the effectiveness of the algorithms, two numerical experiments about finite element structural linear static analysis for large-scale parallel computing were conducted on “Tianhe 2” supercomputer. For each model, both the traditional domain decomposition method and the proposed hierarchical load balancing approach were employed for numerical simulation utilizing 50, 100, 150, and 200 nodes, respectively. Test results show that the proposed method could obtain higher speedup and parallel efficiency compared with the conventional domain decomposition method. The proposed approach can be widely used for solving many kinds of structural analysis problems including linear static analysis, nonlinear static analysis and nonlinear dynamic analysis and so on. In this paper, the authors' current research only focuses on the linear static analysis. For the nonlinear static or dynamic analysis and other kinds of structural analysis, the proposed method can be used as a sub-procedure because the calculations are still dominated by solutions of the same sort of equations.

multi-core cluster, finite element analysis, parallel computing, load balancing

doi: 10.1360/N972016-00607