doi:10.11887/j.cn.202002006 http://journal.nudt.edu.cn

MPI并行程序中通信等待问题的诊断方法及其应用*

武林平,景翠萍,刘 旭,田鸿运
(北京应用物理与计算数学研究所,北京 100094)

摘 要:随着并行规模的扩大,现有通信等待问题的诊断方法存在内存开销大、测量时间开销大等问题。通过对现有通信等待问题诊断方法的深入分析,同时考虑测量开销可控的实际需求,建立基于热点函数的通信等待问题诊断模型。基于上述模型,总结出一种更精简、更实用的通信等待问题诊断方法。将该诊断方法分别应用到二维LARED集成、LARED-S、LAP3D等大规模MPI并行程序的通信等待问题诊断过程,应用效果表明本诊断方法可精确定位导致通信等待问题的关键代码段,给出的优化方案及性能提升空间对于后续的程序改进具有参考价值,其中根据诊断结果优化后的LARED-S程序性能提升32%,通信等待时间减少44%。

关键词:通信等待;MPI并行程序;负载平衡;性能诊断
中图分类号:TP316.4 **文献标志码**:A **文章编号**:1001-2486(2020)02-047-08

Diagnostic methods for communication waiting in MPI parallel programs and applications

WU Linping, *JING Cuiping*, *LIU Xu*, *TIAN Hongyun*
(Institute of Applied Physics and Computational Mathematics, Beijing 100094, China)

Abstract: As the increasing of the scale of parallel systems, some problems such as large measurement cost and memory overhead exist in the diagnostic methods of communication waiting phenomenon. With the deep analysis on the existing diagnostic methods, and considering the actual demand of controllable measurement, a diagnosis model for communication waiting based on hotspot function was established, and a tidy and practical diagnostic method based on the above model was presented. The above diagnostic method was applied to the diagnostic process of the communication waiting phenomenon in the large-scale MPI parallel programs, such as the LARED integration, the LARED-S, the LAP3D. The application results show that this method can accurately identify the key code segment leading to communication waiting and the proposed optimization solution and performance improvement space has reference value for the subsequent program improvement. The optimized LARED-S program, according to the diagnostic result, can increase performance by 32% and reduce communication waiting time by 44%.

Keywords: communication waiting; MPI parallel programs; load balance; performance diagnosis

“块同步并行”(Bulk Synchronous Parallel, BSP)模式(如图1所示)的理想运行情况下,为满足负载平衡需求,任务被均匀地分配到并行程序的每个进程,从而保证每个进程都具有相同的计算性能、通信性能、访存性能,进而使得所有进程均在相同的时刻到达同步点。然而,在实际运行情况下,并行程序本身的算法特征、系统噪音的扰动等因素将破坏这种负载平衡,使得各计算进程不能在同一时刻到达同步点(并行程序的实际运行过程如图2所示),从而引起进程间的相互等待现象[1]。对于阻塞式的点对点通信而言,通信接收方的执行进度将依赖于通信发送方的进度,在这个过程中,通信接收方从启动数据接收操作到实际开始接收数据的这段时间称为通信等待时间[2]。聚合通信过程也类似,因为它的完成需要每个进程参与,当少数进程未能及时到达同一个聚合通信点时,将引起所有参与进程的相互等待现象。

相关研究表明,MPI并行程序在运行过程中,通信等待时间将构成大规模并行程序通信时间的主要部分[2]。特别是将通信密集型应用程序扩展到大规模处理器核数时,这种通信等待现象将加剧,对实现高性能的并行应用程序提出了严峻的挑战。

* **收稿日期**:2019-09-20
基金项目:国家重点研发计划资助项目(2018YFB0204003);国家自然科学基金资助项目(61672003);国家自然科学基金青年科学基金资助项目(11601034)
作者简介:武林平(1977—),男,河南焦作人,研究员,博士,E-mail:wu_linping@mail.iapcm.ac.cn;
景翠萍(通信作者),女,助理研究员,硕士,E-mail:xjtu_cs@163.com

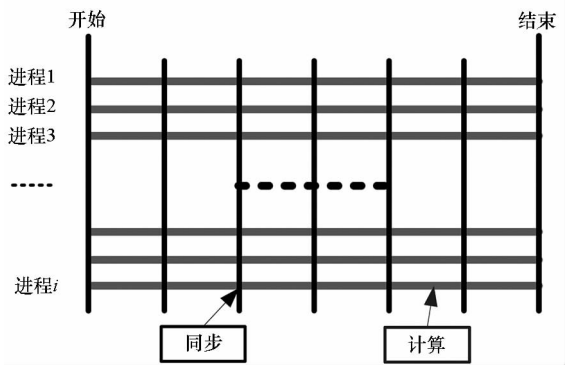


图 1 BSP 并行程序的理想运行过程

Fig. 1 Ideal state of BSP parallel program

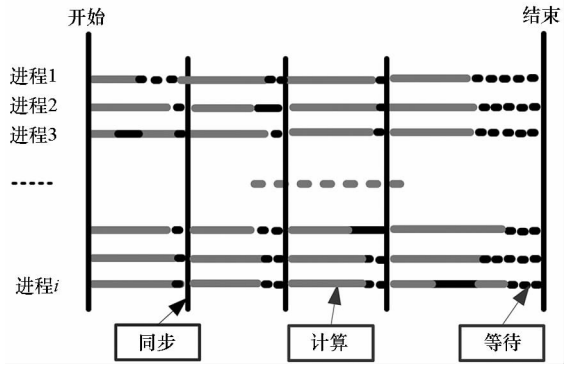


图 2 BSP 并行程序的实际运行过程

Fig. 2 Real state of BSP parallel program

1 通信等待问题的诊断方法相关研究

1.1 MPI 通信等待模式

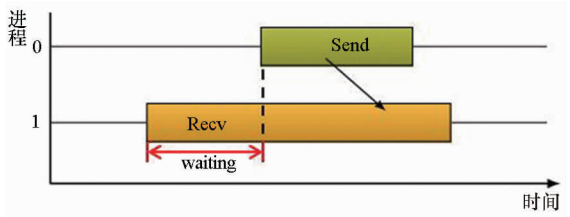
通常,MPI 并行程序中通信等待现象最初是由同步点之前的计算负载不均衡或通信不均衡等原因引发的。但各类因素之间的相互干扰及传播效应,将造成通信等待现象存在于 MPI 并行程序的全部运行过程中。这种“现象”与“原因”之间的复杂关联关系,使得定位导致通信等待问题的根本原因的诊断过程变得非常复杂。例如某个计算过程在某段时间内计算负载不均衡,但下个时刻这种负载不均衡现象可能与其他计算过程的负载不均衡现象相互抵消,这种负载不均衡现象并不会导致 MPI 通信等待问题。

因此,分析通信等待现象如何形成、在何处形成,对于快速准确地找到其根本原因从而消除通信等待现象至关重要。文献[2]详细分析了 MPI 并行程序中点对点通信、聚合通信以及同步过程中可能出现的各种通信等待模式。文献[3]提出了一种名为 FACT 的 Trace 分析技术,可在小规模系统上低开销地获取大规模并行程序的通信轨迹数据,采用人工分析方法可识别 MPI 并行程序中的通信等待模式。参考 MPI－3 标准,MPI 通信过程中主要存在以下几种常见的通信等待模式[2]:

1)点对点通信过程中:存在 2 种典型的通信等待模式,包括晚发送模式和晚接收模式,分别如图 3(a)和图 3(b)所示。

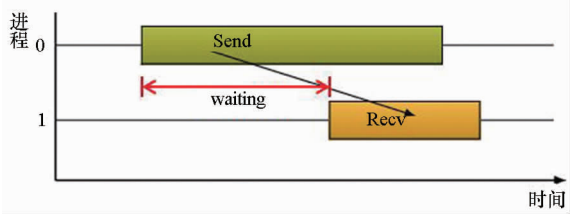
2)同步过程(MPI_Barrier)中:同步过程中出现的通信等待问题主要是因为进程不能及时到达同步点产生的,主要形成过程如图 3(c)所示,称为早同步模式。

3)聚合通信过程中:存在多种典型的通信等待模式,包括晚广播模式、早规约模式以及全局交换模式等,其中早规约模式如图 3(d)所示。在早规约模式下,进程过早参与聚合通信,导致这些进程上出现通信等待现象。以此类推,其他聚合通



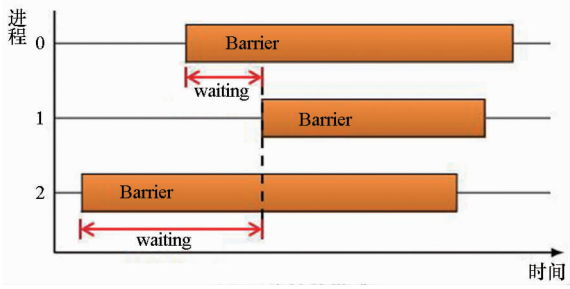
(a) 晚发送模式

(a) Late sender pattern



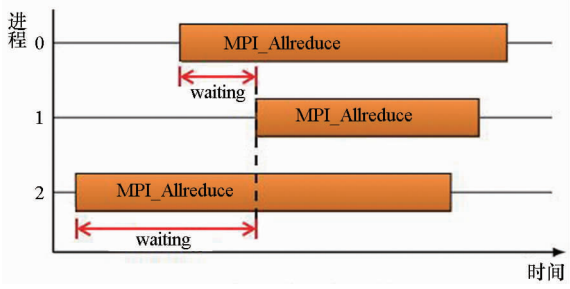
(b) 晚接收模式

(b) Late receiver pattern



(c) 早同步模式

(c) Early barrier pattern



(d) 早规约模式

(d) Early reduce pattern

图 3 常见的 MPI 通信等待模式

Fig. 3 Scheme of MPI communication waiting

信等待模式的形成过程也是类似的。

1.2　通信等待模式的量化分析方法

针对以上几种常见的通信等待模式,相关研究主要从原因定位、通信等待时间量化分析、通信等待问题诊断模型(Diagnostic Model for Communication Waiting Problem, CWP-DM)3 个方面,详细分析了 MPI 并行程序中通信等待问题形成的原因以及可能产生的影响。

为了更加清楚地阐述该问题,下面首先给出超时时间、等待时间、同步间隔这 3 个关键词的具体含义。

1)超时时间:特指造成通信等待现象的原因,例如某代码段在进程 p 上的运行时间远超平均值,超出平均值的时间即为超时时间。一般来讲,MPI 并行程序的超时时间将存在于多个代码段,这些超时时间将逐步累加,最终导致通信等待现象的发生。

2)等待时间:指进程已参与 MPI 通信但仍处于空闲等待状态的时间,是通信等待现象的表现形式。

3)同步间隔:进程 p 和进程 q 的两个相邻同步点之间的时间间隔称为一个同步间隔。在一个同步间隔内,进程 p 和进程 q 之间没有其他同步点。

1.2.1　通信等待问题形成的原因

图 4 以点对点通信等待模式为例,采用时间线图描述 MPI 并行程序中由于计算负载不均衡、通信不均衡等引发的通信等待现象[4]。其中,不同进程上执行的函数都以不同颜色的矩形方框表示,矩形方框的长度表示每个函数的执行时间,进程间带箭头的线段表示进程间的通信关系或同步关系。例如,S1 表示进程 1 上执行的 MPI_Send 函数,S2 表示进程 2 上执行的 MPI_Send 函数,R1 表示进程 2 上执行的 MPI_Recv 函数,comp 表示计算函数。

在图 4 中,进程 2 和进程 3 都出现了通信等待现象,显示为进程 2 和进程 3 上的阴影区域。MPI 并行程序中通信等待现象可分为直接等待现象和间接等待现象。由于某个进程上函数的执行时间超时直接引发另一个进程上出现的等待现象称为直接等待现象,产生的等待时间称为直接等待时间;直接等待现象进一步传播而引发的通信等待现象称为间接等待现象,由此产生的等待时间称为间接等待时间[5]。在图 4 示例中,进程 2 上的通信函数 R1 产生了直接等待时间,它是由进程 1 上执行的函数 comp 计算超时引起的。与

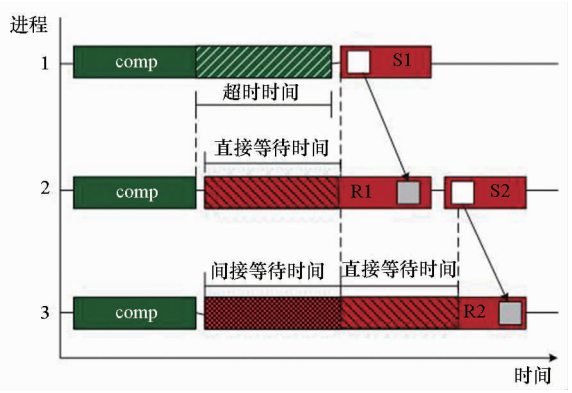


图 4　MPI 并行程序中通信等待问题的时间线图

Fig. 4　Time-line diagram of communication waiting in MPI parallel program

此同时,进程 2 上出现的直接等待现象又进一步引发进程 3 上出现间接等待现象,因此进程 3 上产生了间接等待时间。除此之外,进程 3 上还产生了直接等待时间,这是由于进程 2 上的通信函数 R1 执行时间超时引起的。

1.2.2　通信等待时间量化分析

通常采用 Trace 分析技术检测 MPI 并行程序中的通信等待时间。文献[6]针对 MPI 并行程序中点对点通信和聚合通信,提出了一种测量通信等待时间的方法:首先记录 MPI 通信和同步操作在每个进程上的执行时间,然后计算时间位移从而获取通信等待时间。文献[7]针对 MPI 单边通信,提出了一种测量通信等待时间的方法,该方法借鉴了文献[4]中的研究思路。

本文中,用二元组(p,i)来描述进程 p 上执行的函数 i。理想情况下,参与通信的进程应同时开始执行相应的通信函数。如果进程 p 上的通信函数(p,i)的完成取决于另一个进程 q 上执行的函数(q,k),并且进程 p 在进程 q 开始执行函数(q,k)之前开始执行函数(p,i),则这两个函数形成了一个同步点。因此,同步点 $S=((p,i),(q,k))$是函数(p,i)和函数(q,k)的二元组,并且同步点的形成需要满足以下条件[8]:

1)函数(p,i)的完成取决于函数(q,k);

2)函数(p,i)比函数(q,k)开始执行得早,即 $Enter(p,i)<Enter(q,k)$;

3)两个函数存在重叠的执行时间,即 $Exit(p,i)>Enter(q,k)$。

因此,对于进程 p 上执行函数(p,i)和进程 q 上执行函数(q,k)的这段时间内,进程 p 上执行的函数(p,i)出现通信等待现象。其中,等待时间可量化[9]为:

$$w(p,i)=Enter(q,k)-Enter(p,i) \qquad (1)$$

式(1)不仅适用于点对点通信等待模式,而且适用于同步等待模式和聚合通信等待模式。对于后者,进程 q 特指最晚参与同步过程或最晚参与聚合通信的进程。

1.2.3 通信等待问题诊断模型

通常基于通信等待问题诊断模型可定位导致通信等待问题的关键函数,然后人工分析这些关键函数的详细执行行为,最终定位导致通信等待问题的根本原因。文献[9]中提出了一种通信等待问题诊断模型,给出了在所有同步间隔中某个函数调用路径导致的直接等待时间和间接等待时间,见式(2):

$$\begin{cases} short\ term\ cost(q,c) \\ = \sum\limits_{\zeta=((p,i',i),(q,k',k))} \dfrac{1}{\hat{\delta}_\zeta + \hat{\omega}_\zeta} \delta_\zeta(q,c)\omega(p,i) \\ long\ term\ cost(q,c) \\ = \sum\limits_{\zeta=((p,i',i),(q,k',k))} \dfrac{1}{\hat{\delta}_\zeta + \hat{\omega}_\zeta} \delta_\zeta(q,c)\phi(p,i) \end{cases} \tag{2}$$

其中,$\hat{\delta}_\zeta = \sum\limits_{c\in C} \delta_\zeta(q,c)$,$\hat{\omega}_\zeta = \sum\limits_{l=k'+1}^{l<k} \omega(q,l)$,$C$ 表示一个同步间隔内执行的所有函数的集合,其他参数说明详见文献[9],这里不再赘述。

该诊断模型考虑了多方面因素的影响,分析了直接等待时间和间接等待时间形成的根本原因,并且需要对 MPI 并行程序中所有函数调用路径展开详细的测量与分析[10]。然而,在实际应用过程中,测量发现这种诊断模型耗费的内存开销较大,比如 LARED 集成程序[11]在 1024 核运行时单进程的内存开销高达 16 GB,测量数据见表 1。文献[8]中还指出,将上述模型应用到 Sweep3D 程序中,测量时间和内存开销随着并行规模的扩大将快速增长。除此之外,过大的测量开销将导致性能数据出现偏差,从而导致诊断算法误报率增大。对于内存受限的高性能计算机系统(例如平均单核内存少于 2 GB),上述模型引入的测量开销甚至可能造成诊断失效。

表 1 LARED 程序运行 1000 个时间步单进程的内存开销

Tab. 1 Memory overhead for LARED program running in 1000 time steps

进程数	256	512	1024
内存开销/GB	35	21	16

2 基于热点函数的通信等待问题诊断方法

2.1 诊断模型

根据以上分析,考虑测量开销可控、诊断算法误报率低的实际需求,需要精简通信等待问题诊断模型,降低测量与分析开销。对于 MPI 并行程序,定位间接等待现象产生的原因将会带来额外的开销,并且直接等待现象消除后,间接等待现象也随之消除。因此,本文建立的通信等待问题诊断模型不再考虑直接等待现象的传播对程序性能的间接影响,只定位导致直接等待现象的计算超时或通信超时函数。另外,程序的性能瓶颈大多是由于性能热点函数引发的,为了更精准定位通信等待问题,本文建立的通信等待问题诊断模型首先从热点函数引发的通信等待时间入手。例如,可选取执行时间大于 1% 的函数作为热点函数。

在一个同步间隔内,进程 p 上的函数 (p,c) 的执行时间由式(3)计算给出。为了准确反映每个函数对 MPI 并行程序的通信等待时间形成的影响,式(3)排除了直接等待现象的传播对程序性能产生的间接影响。

$$t(p,c) = Exit(p,c) - Enter(p,c) - w(p,c) \tag{3}$$

在一个同步间隔内,进程 p 上执行的函数 (p,c) 的超时时间则是由函数 c 在进程 q 和进程 p 上的执行时间之间的差值计算得出,见式(4)。在一个同步间隔内,由于执行时间差为负数的函数不会引起进程 p 出现通信等待现象,因此这部分时间忽略不计。

$$\sigma_t(q,c) = \begin{cases} t(q,c) - t(p,c) & t(q,c) > t(p,c) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

根据以上分析,针对 MPI 并行程序,建立基于热点函数的通信等待问题诊断模型(Diagnostic Model for Communication Waiting Problem based on Hotspots, H-CWP-DM),见式(5)。

$$cost(q,c) = \frac{\sigma_t(q,c)}{\sum\limits_{f\in C} \sigma_t(q,f)} \cdot w(p,i) \tag{5}$$

其中,C 表示一个同步间隔内执行的所有热点函数的集合,$\sum\limits_{f\in C} \sigma_t(q,f)$ 表示一个同步间隔内在进程 q 上执行各热点函数的超时时间之和,$w(p,i)$ 表示在一个同步间隔内进程 p 上执行的函数 (p,i) 产生的通信等待时间,$cost(q,c)$ 表示进程 q 上执行的函数 (q,c) 引发进程 p 上产生的通信等待开销。

该诊断模型通过比较一个同步间隔内各热点

函数的执行时间来定位超时时间较大的函数，从而确定导致通信等待问题的关键函数，并可给出这些关键函数产生的通信等待时间。通过该诊断模型，可定位产生通信等待时间比较大的关键函数，然后分析这些关键函数的详细执行行为，最终可以定位产生通信等待问题的根本原因。除此之外，式(5)建立的诊断模型不仅适用于点对点通信等待模式，而且适用于同步等待模式和聚合通信等待模式。对于后者，进程 q 特指最晚参与同步活动或最晚参与聚合通信的进程，并且选取最早参与同步活动或最早参与聚合通信活动的进程 p 来分析进程 p 上产生通信等待现象的根本原因。

2.2　诊断流程

基于 H-CWP-DM 诊断模型，总结出精简版、更实用的通信等待问题诊断流程，主要包括以下步骤：

步骤 1：测量并分析 MPI 并行程序运行时的性能热点函数。

步骤 2：测量 MPI 并行程序的通信等待时间，确定通信等待时间对应的同步间隔。

步骤 3：根据 H-CWP-DM 诊断模型，定位产生通信等待时间比较大的关键函数。

步骤 4：从计算、访存、通信等方面分析关键函数的详细执行行为，明确产生通信等待问题的根本原因。

3　实际应用

3.1　高性能计算平台

为了获取 H-CWP-DM 诊断模型需要的各种性能数据，在某并行机系统上基于主流性能分析工具形成了可用的并行程序运行时行为测量软件平台，如图 5 所示。该软件平台可用于测量并行程序在系统上的不同层次、不同方面的性能数据，包括性能热点函数、计算时间、通信时间及等待时间等。

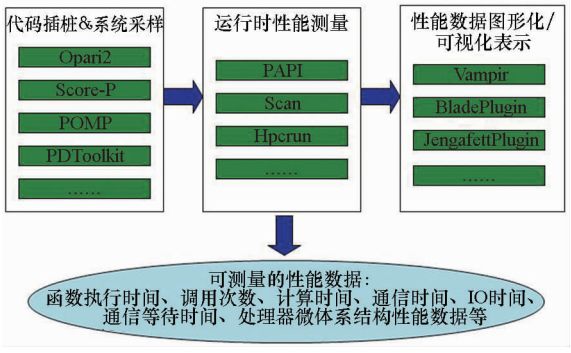


图 5　并行程序运行时行为测量软件平台

Fig. 5　Software of performance measurement for parallel applications at run time

所有测量在一台包含 1050 个双路 16 核计算节点的并行机上进行(共 33 600 个处理器核)，计算节点操作系统使用 Linux；处理器为 Intel Xeon 系列，每颗处理器包含 128 MB 的共享三级 Cache，内存控制器集成在处理器内部，配置为 NUMA 结构，单节点共享 64 GB 内存。

3.2　MPI 并行程序

上述诊断方法目前已应用到 LARED-S、二维 LARED 集成、LAP3D 等数十个实际生产性数值模拟程序的性能问题诊断过程中。这些程序都是基于 MPI 实现的大规模并行程序，可以成功扩展到上万处理器核[11]。测量发现这些程序都存在通信等待问题，采用上述方法诊断优化后，这些程序的性能都提升了 15% 以上。下面主要以 LARED-S 程序的性能问题诊断为例，详细介绍诊断过程，其他程序的诊断过程类似，这里不再详述。

3.3　LARED-S 程序的性能诊断

根据第 2.2 节给出的通信等待问题的诊断流程，LARED-S 程序的性能问题诊断过程如下：

步骤 1：测量并分析程序运行时的性能热点函数。LARED-S 程序是欧拉网格下的二维、三维并行多介质辐射流体力学模拟程序，主要用于惯性约束聚变等领域中辐射流体界面不稳定性模拟[12]。该程序采用 960 个进程运行时的性能热点函数分布如图 6 所示。其中，通信时间占程序执行时间的 49.4%，计算时间占 42.6%。

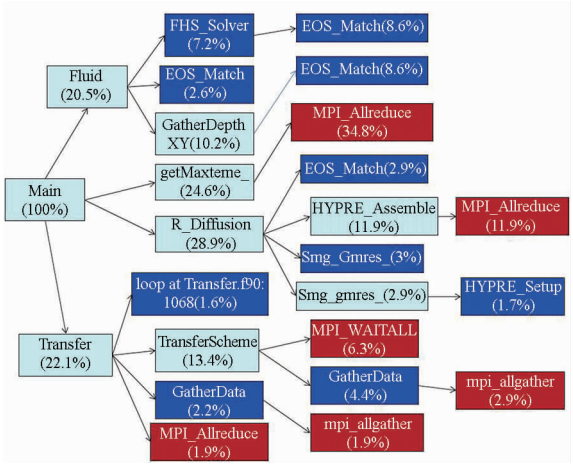


图 6　LARED-S 程序的性能热点函数分布

Fig. 6　Performance hotspots distribution of LARED-S program

步骤 2：测量 LARED-S 程序中通信等待时间，确定通信等待时间对应的同步间隔。测量发

现 MPI_Allreduce 通信过程中通信等待时间占 99%。MPI_Allreduce 的通信时间及等待时间在各进程上的分布见图 7,运行 3 个时间步的最大等待时间之和达 125 s,平均每时间步最大等待时间达 41.6 s。

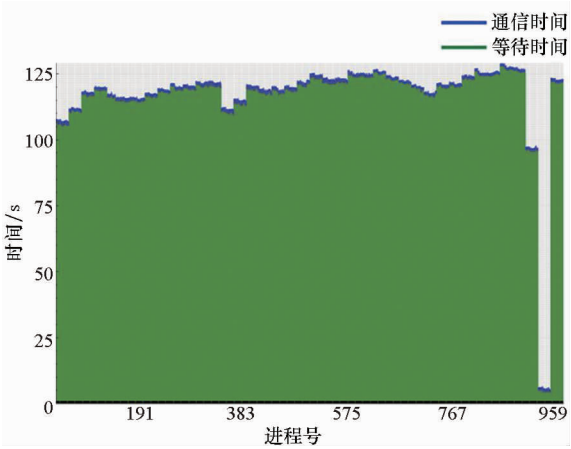


图 7 MPI_Allreduce 的通信时间及等待时间分布

Fig. 7 Communication time and waiting time distribution of MPI_Allreduce

步骤 3:根据 H-CWP-DM 诊断模型,定位产生通信等待时间比较大的关键函数。根据式(5)测量并分析性能数据,定位出导致 MPI_Allreduce 通信等待时间开销大的关键函数如图 8 所示。测量发现,在一个时间步内,EOS_Match 函数的 EOS_Match_If_970 代码段和 EOS_Match_If_946 代码段在聚合通信过程中产生的通信等待时间较大,二者等待时间之和为 30 s。在一个时间步内,优化前 MPI_Allreduce 通信过程中产生的通信等待时间为 41.6 s;如果 EOS_Match_If_970 和 EOS_Match_If_946 代码段优化负载平衡算法后,MPI_Allreduce 的执行时间最多可减少 30 s。由上述数据可知,这 2 个代码段改进后 MPI_Allreduce 的等待时间可缩短 72%(30 s/41.6 s)。

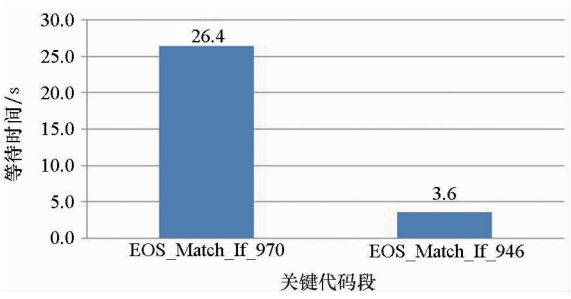
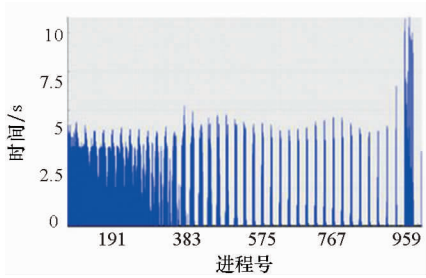


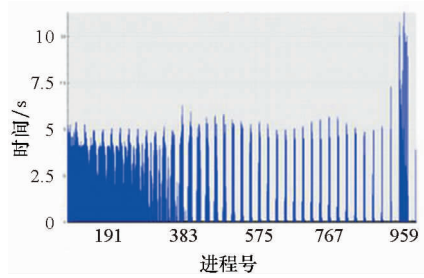
图 8 计算超时导致的等待时间

Fig. 8 Waiting time caused by excess computation

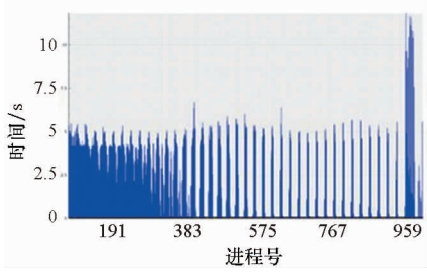
步骤 4:分析关键代码段的详细执行行为,明确产生通信等待问题的根本原因。找到关键代码段之后,需要分析 EOS_Match 函数的 2 个代码段的详细执行行为。图 9 和图 10 分别是代码段 EOS_Match_If_970 在不同时间步(*Step*)和不同进程上的执行时间和执行次数分布。其中,最大执行时间为 11 s,平均执行时间为 2 s,最小执行时间 0 s;最大执行次数为 1.3E +06,平均执行次数为 3.3E +05,最小执行次数为 0。比较图 9、图 10 发现,该代码段在时间维度、空间维度的执行时间与执行次数分布特征基本吻合。除此之外,测量发现 EOS_Match_If_946 代码段也有类似的规律,这里不再赘述。



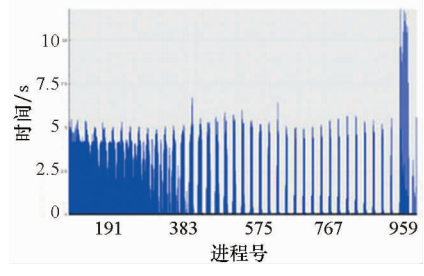
(a) *Step* = 0



(b) *Step* = 1



(c) *Step* = 99

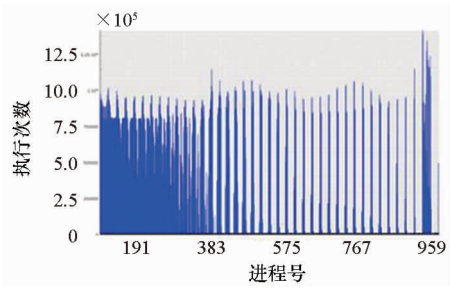


(d) *Step* = 100

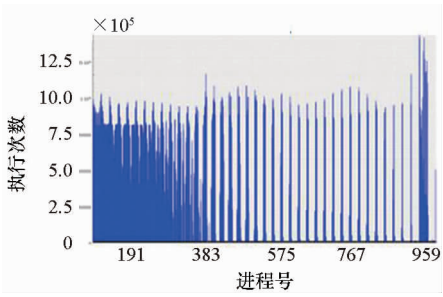
图 9 EOS_Match_If_970 代码段的执行时间分布

Fig. 9 Execution time distribution of EOS_Match_If_970

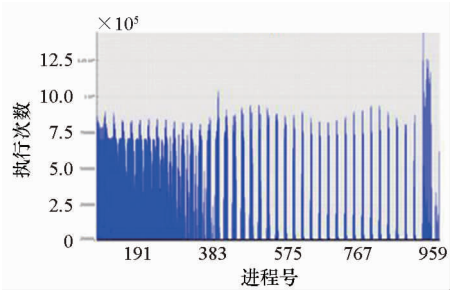
因此,可推断代码段 EOS_Match_If_970 和 EOS_Match_If_946 在不同进程上的执行次数分



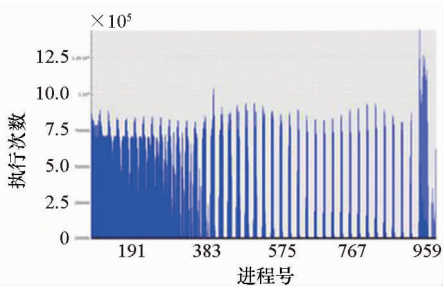
(a) *Step* = 0



(b) *Step* = 1



(c) *Step* = 99



(d) *Step* = 100

图 10　EOS_Match_If_970 代码段的执行次数分布

Fig. 10　Visits distribution of EOS_Match_If_970

布不均衡是导致聚合通信过程出现通信等待问题的根本原因。据此,开发人员对 EOS_Match 函数进行了 2 步优化:首先根据物理模拟需求精简了该函数的计算量,其次通过调整网格剖分策略实现相关代码段执行次数的均衡分布。优化后的 LARED-S 程序在相同运行条件下性能提升 32%,MPI_Allreduce 的等待时间缩短 44%(18.1s/41.6 s)。

除此之外,本文测量分析了该程序分别采用 CWP-DM 诊断模型和 H-CWP-DM 诊断模型所需的诊断开销,见表 2。相比于 CWP-DM 诊断模型,采用 H-CWP-DM 诊断模型所耗费的时间开销和内存开销都相对较低。由于测量所用的并行机的计算节点可用主内存为 64 GB,而 CWP-DM 诊断模型单进程所需的内存开销高达 115 GB,因此造成该诊断方法失效。

表 2　LARED-S 程序运行 3 个时间步的性能诊断开销

Tab. 2　Diagnostic overhead for LARED-S program running in 3 time steps

诊断模型	时间开销	内存开销
H-CWP-DM	907.2 s	933 MB
CWP-DM	诊断失效	115 GB

4　结论

通过对现有通信等待问题诊断方法的深入分析,同时考虑测量开销可控、诊断算法误报率低的实际需求,建立了 H-CWP-DM 诊断模型,并基于该诊断模型总结出精简版、更实用的通信等待问题诊断方法。将本诊断方法应用到 LARED-S、二维 LARED 集成、LAP3D 等数十个实际生产性数值模拟程序的通信等待问题诊断过程。实际应用结果表明本诊断方法可精确定位导致通信等待问题的关键代码段,给出的优化方案及性能提升空间对于后续的程序改进具有参考价值。其中,优化后的 LARED-S 程序性能提升 32%,通信等待时间缩短 44%。

除此之外,针对通信等待问题,未来的研究将立足于下列 2 个问题:

1)实现通信等待问题的自动化诊断。将本诊断方法集成到并行程序运行时行为测量软件平台中,可自动生成通信等待问题的诊断报告,降低人工测量与分析时间。

2)将本诊断方法应用到更大规模的 MPI 并行程序中,甚至是超过百万核运行规模的应用程序。

参考文献(References)

[1]　武林平, 魏勇, 徐小文, 等. 系统噪音影响的量化分析[J]. 计算机研究与发展, 2015, 52(5): 1146 – 1152.
WU Linping, WEI Yong, XU Xiaowen, et al. Impact of system noise by quantitative analysis[J]. Journal of Computer Research and Development, 2015, 52(5): 1146 – 1152. (in Chinese)

[2]　Geimer M, Wolf F, Wylie B J N, et al. The SCALASCA performance toolset architecture [J]. Concurrency and Computation Practice and Experience, 2010, 22(6): 702 – 719.

[3]　Zhai J D, Sheng T W, He J Z, et al. FACT: fast communication trace collection for parallel applications

through program slicing [C]// Proceedings of the 21th International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'09), 2009.

[4] Wylie B J N, Geimer M, Mohr B, et al. Large-scale performance analysis of Sweep3D with the Scalasca toolset[J]. Parallel Processing Letters, 2010, 20(4): 397 – 414.

[5] Szebenyi Z, Wylie B J N, Wolf F. Scalasca parallel performance analyses of PEPC [C]// Euro-Par 2008 Workshops-Parallel Processing, Springer, 2009: 305 – 314.

[6] Geimer M, Wolf F, Wylie B J N, et al. A scalable tool architecture for diagnosing wait states in massively parallel applications[J]. Parallel Computing, 2009, 35(7): 375 – 388.

[7] Hermanns M A, Miklosch M, Böhme D, et al. Understanding the formation of wait states in applications with one-sided communication[C]// Proceedings of the 20th European MPI Users' Group Meeting, ACM, 2013: 73 – 78.

[8] Böhme D, Geimer M, Arnold L, et al. Identifying the root causes of wait states in large-scale parallel applications[J]. ACM Transactions on Parallel Computing, 2016, 3(2): 1 – 24.

[9] Böhme D. Characterizing load and communication imbalance in parallel applications [D]. Germany: RWTH Aachen University, 2014.

[10] Mao G Y, Böhme D, Hermanns M A, et al. Catching idlers with ease: a lightweight wait-state profiler for MPI programs[C]// Proceedings of the 21th European MPI Users' Group Meeting, Kyoto, Japan, 2014: 103 – 108.

[11] Mo Z Y, Zhang A Q, Cao X L, et al. JASMIN: a parallel software infrastructure for scientific computing[J]. Frontiers of Computer Science in China, 2010, 4(4): 480 – 488.

[12] 范征锋, 徐小文, 孙文俊, 等. 辐射流体界面不稳定性模拟程序 LARED-S[C]//第十六届全国流体力学数值方法研讨会, 北京, 2013: 25 – 26.
FAN Zhengfeng, XU Xiaowen, SUN Wenjun, et al. Radiation fluid interface instability simulation program LARED-S[C]// The 16th National Symposium on Numerical Methods in Fluid Mechanics, Beijing, 2013: 25 – 26. (in Chinese)