第 37 卷 第 5 期
Vol.37 No.5

计 算 机 工 程
Computer Engineering

2011 年 3 月
March 2011

• 开发研究与设计技术 • 文章编号：1000—3428(2011)05—0261—03 文献标识码：A 中图分类号：TP301.6

复杂地形环境下的电磁覆盖范围并行算法

周 鋆，朱 承，张维明，黄金才，刘 忠

(国防科技大学信息系统工程重点实验室，长沙 410073)

摘 要：复杂地形环境下的电磁覆盖范围的仿真计算需要强大的计算能力作为支撑，搭建基于消息传递接口的机群实行并行计算能够解决该问题。基于此，建立复杂地形环境下电磁覆盖范围的并行算法模型，设计并行过程中任务粒度选择和并行性能实验。在实验室多机环境下，仿真计算速度得到提高，为类似并行计算问题的任务粒度选择提供了参考。

关键词：并行计算；电磁覆盖范围；消息传递接口；加速比；任务粒度

Parallel Algorithm of Electromagnetic Wave Propagation in Complex Terrain Environment

ZHOU Yun, ZHU Cheng, ZHANG Wei-ming, HUANG Jin-cai, LIU Zhong

(Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China)

【Abstract】Simulation of electromagnetic wave propagation in complex terrain environment needs powerful computing ability. Parallel computing using cluster which is based on MPI can solve this problem commendably. This paper puts forward a parallel algorithm of electromagnetic wave propagation in complex terrain environment, and designs experiments considering task granularity and capability in parallel process. Simulation speed is improved in a cluster environment in laboratory. Methods of choose an appropriate task granularity for similar parallel problems is proposed.

【Key words】parallel computing; electromagnetic wave propagation; Message Passing Interface(MPI); speed ratio; task granularity

DOI: 10.3969/j.issn.1000-3428.2011.05.089

1 概述

电磁覆盖范围的仿真计算对于雷达探测范围的仿真以及电子对抗方案的评估具有十分重要的意义[1-2]。由于受到地形环境、接收机参数、多路径传播环境等诸多不确定因素的综合影响，对于目标区域电磁覆盖范围的仿真计算，需要对各个目标点逐一进行电磁传播模型的匹配和传播损耗计算，以面积为 1 296 km^2 的战场区域为例，按照航天飞机雷达地形测绘任务(SRTM)得到的 90 m 分辨率数据进行划分，将有 16 万个点需要计算，总计算量会非常庞大。

并行计算方法由于能够解决大规模计算的问题，已被成功地应用于天气预报、石油勘探、航空航天等领域，在电磁覆盖仿真计算领域，并行计算也有很多应用[3]，但这些讨论的都是关于单个电磁传播模型的并行算法。本文考虑复杂地形环境下的电磁传播覆盖范围仿真的问题，根据环境参数不同将会调用不同的电磁传播模型，因此，需要构建适用于本文应用背景的并行算法，减少计算时间以提高效率。

2 复杂地形环境下的电磁传播覆盖范围计算

在求解计算信号发射点到目标区域的电磁传播覆盖范围时，可以将该区域划分为规则的二维数字网格 $\pounds^{m\times n}$，相当于求解信号到所有网格节点 $E=\left[e_{ij}\right]\in\pounds^{m\times n}$ 的传输损耗。然而在复杂的地形环境中，信号到各点的传输损耗会受到很多因素的影响。这里建立影响第 ij 点 $(i=1,2,\mathrm{L},m;j=1,2,\mathrm{L},n)$ 探测能力的因素集：

$$e_{ij}=<e_{ij1},e_{ij2},e_{ij3}>$$

其中，该点的地形环境 e_{ij1} 是影响电磁传播的重要因素，电磁波会绕过传播路径上的障碍物，从而产生光滑球面绕射损耗、刃峰绕射损耗、不规则地面绕射损耗等；该点的接收机参数 e_{ij2} 是指电磁波信号接收天线的参数指标，包括方向性系数、增益系数、输入阻抗等；该点所处的多路径传播环境 e_{ij3} 对损耗计算也有重要影响，在建筑物高而密集的城市地区，干扰信号可以通过多种途径到达接收机，而在开阔的郊区，干扰信号的作用相对较小。

设信号发射点到目标区域内任意一点 e_{ij} 的传输损耗集为 $C\in¡^{m\times n}$，则有 $f:E\to C$，那么电磁传输损耗的计算可以表示为：$c_{ij}=f(e_{ij}),\ i=1,2,\mathrm{L},m;\ j=1,2,\mathrm{L},n$。

由于自由空间传播模型适用于不发生反射、折射、散射和吸收现象，而只存在电磁波能量扩散引起传输损耗的情况；Okumura-Hata 模型是以准平滑地形的市区作基准，在不同校正因子下传输损耗的计算方式不同；COST231-Hata 是在 Hata 模型的基础上修正得出的；单刃峰绕射模型则是考虑了电波传播路径上存在单个刃形障碍物阻挡而发生绕射的情况；Bullington 模型则将多个刃峰逐次等效，最后成为等效单刃峰，从而计算出总的传输损耗[4]。

因此，电磁传输损耗的计算流程应该是先根据信号发射机的坐标位置和探测点位置确定所需计算的链路之后，再对链路上的所有环境信息进行综合分析，并选择合适的电磁传播模型进行损耗计算，最后得到损耗后再判断是否可探测，可以抽象为图 1 所示的 3 个流程。

基金项目：国防科技大学优秀研究生创新基金资助项目(S100501)

作者简介：周 鋆(1987－)，男，硕士研究生，主研方向：并行计算；朱 承，副教授、博士；张维明、黄金才、刘 忠，教授、博士

收稿日期：2010-07-22 **E-mail：**zhouyun@nudt.edu.cn

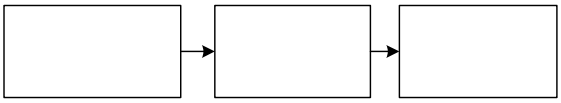

图 1 电磁传输损耗计算流程

在实际的计算过程中，由于探测点数量庞大，且每个点的计算需要根据地形来匹配合适的传播模型而变得复杂，这种串行的逐点计算模式需要耗费很多的时间，因此需要并行化处理。

3 并行算法分析与设计

3.1 并行算法基本框架

对于流程 F1，由于数据读取及数据预处理的任务量较小，如果将这个流程并行化，那么将会花费大量时间在任务分配和通信传输上，不能达到预期的目的。因为该流程任务量小，同时为了降低通信量，可以在每个处理节点上都串行执行该流程。

在关键的流程 F2、F3 中，需要对每个结点所处的环境进行分析，并进行相应的损耗计算，这样会产生相当大的计算量。同时由于这些计算是建立在 F1 的基础上的，如果实现并行，处理节点在 F2、F3 的计算中，并不需要相互通信，只是在计算完毕后，需要通信来返回结果，这里加入流程 F4，如图 2 所示。则可以通过功能分解的方式对任务进行划分[5-6]，使得不同的结点同时被不同的处理节点计算，实现并行以减少整个运行时间。

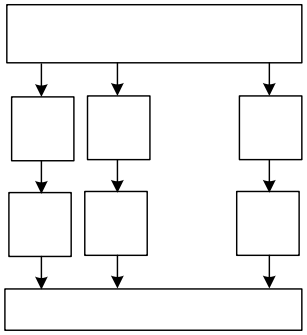

图 2 并行计算流程

在不同的处理节点中需要有一个 Master 处理节点，和多个 Slave 处理节点。其中，Master 用于计算中总的控制和协调，Slave 则用于完成 Master 分配的任务并返回结果。对于并行过程中的任务分派方式，这里有 2 种策略：(1)采用静态分派的方式，Master 在计算开始前，就将任务划分到指定的 Slave 上，当某一 Slave 先计算完毕后，不会再被分派任务；(2)采用动态分派的方式，即 Master 在整个计算始终，不断向 Slave 分派任务，直到所有任务计算完毕。由于各个 Slave 的处理速度不会完全相同，而且不同的损耗计算模型的复杂度也是不同的，为了达到负载平衡，提高各个 Slave 的利用率，降低总的运行时间，这里采用动态分派的方式。

3.2 并行算法性能分析

在电磁传输损耗计算的并行化过程中，Slave 处理节点一次计算所承担的并行任务量，即任务粒度会直接影响并行算法的性能。假设任务粒度为 ∂，总的并行任务量为 W，通信次数为 n，处理机数目为 p，这里有：

$$n=(W/\partial)\times 2 \tag{1}$$

式(1)说明任务粒度 ∂ 越小，通信次数 n 越大，而总的通信传输量是不变的，这样频繁的建立通信会额外的增加时间开销；而当任务粒度特别大时：

$$\partial_M = W/p \tag{2}$$

假设为式(2)所得到的任务粒度，显然通信次数为 $2p$，就是说计算开始后 Master 与各 Slave 通信一次用以分配任务，待各 Slave 计算完毕后再通信一次用以返回结果。由于受软硬件、网络环境等多种因素的影响，各 Slave 之间的计算能力不可能一样，从而导致负载不平衡，也会增加时间开销。因此，在实际计算过程中，由于数据点数目十分庞大，需要选用较大的任务粒度，即 Master 将待计算的数据点按块分配给各个 Slave，但同时也要考虑到负载平衡的问题。

另外假设串行算法在单台处理机上的执行时间为 T_S，并行算法在 p 台处理机上的执行时间为 T_P。针对算法总的处理部分 M 不变的情况，这里定义算法中需要串行处理的部分 M_S(F1、F4)，及算法中需要并行处理的部分 M_P(F2、F3)，则有 $M=M_S+M_P$；f 是串行处理部分所占比例，即 $f=M_S/M$，则 $(1-f)$ 为并行处理部分所占比例。则加速比 S_P 可以写为：

$$S_P=T_S/T_P=\frac{M_S+M_P}{M_S+M_P/p}=\frac{p}{1+f(p-1)} \tag{3}$$

通过分析知，当 $p\to\infty$ 时，$S_p=1/f$，它表明随着处理机数目的增加，加速比不会一直增加，而是有上限的，上限为 $1/f$。另外，如果考虑额外的开销 M_E(如通信、同步、I/O 及等待)，式(3)可以修正为：

$$S_P=T_S/T_P=\frac{M_S+M_P}{M_S+M_P/p+M_E}=\frac{p}{1+f(p-1)+M_Ep/M} \tag{4}$$

当 $p\to\infty$ 时，$S_P=1/(f+M_E/M)$，表明串行处理部分越多，并行额外开销越大，加速比越小。

3.3 基于 MPI 的并行算法实现

通过上面的分析，Master 处理节点采用任务动态分配的方式，向各 Slave 处理节点分配要计算的数据块序号，直到所有的数据块都被计算完毕，然后 Master 再负责收集计算结果并输出。基于 MPI 的并行算法流程如图 3 所示。

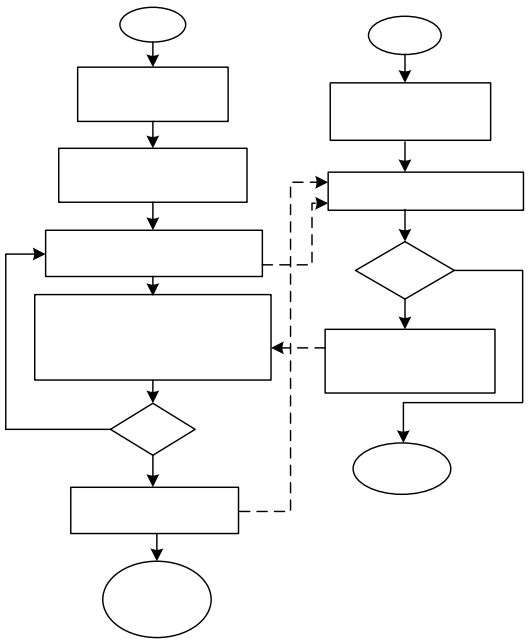

图 3 基于 MPI 的并行算法流程

MPI 基于消息传递模式实现并行，是目前广泛使用的并行程序设计模式，具有可移植性好、功能强大、效率高等许多优点，这里利用它的一种实现 MPICH 绑定 C 语言进行开

发，程序伪代码如下：

```
Begin(Master)
for(i=0; i< M;i++)
{
  MPI_Send(将要计算的数据块序号发送出去);
}
for(i=0; i<M ;i++)
{
  MPI_Recv(依次接收 Slave 计算的损耗结果);
  if(还有其他的数据块没有被计算，则继续发送)
  {
    MPI_Send(数据块序号);
  }
  else 如果计算完毕，则向 Slave 发送 terminator_tag 结束标识
  {
    MPI_Send(terminator_tag);
  }
} End
Begin(Slave)
MPI_Recv(接收 Master 发送过来的消息);
while(如果不是结束标识)
{
  for(j=0; j<N ;j++)
  {
    Closs(综合分析并计算损耗);
  }
  MPI_Send(向 Master 返回进算结果);
  MPI_Recv(继续接收 Master 发送消息);
}
计算完毕结束
End
```

4　实验设计及结果分析

本文在实验室环境下搭建 5 台双核计算机组成的集群系统(共 10 个 CPU)，操作系统均采用 Windows XP Professional SP3，节点机处理器为 Intel Core 2 Duo E7200(2.53 GHz)，网络环境为 100 Mb/s 局域网。实验以 400×400 的表示目标点高程的 SRTM 格式 DEM 数据为基础，建立二维目标点网格，覆盖范围为 1 296 km^2，并设计 3 种任务粒度 $\partial_1,\partial_2,\partial_3$，其中，

$$\partial_1=\left\{e_i\left|e_i=\sum_{j=1}^{400}e_{ij},i=1,2,\mathrm{L}\ ,400,e_{ij}\in\ ^{400\times400}\right.\right\}$$

表示 Master 每次为 Slave 分配 1 行 400 个目标点；∂_2 为 ∂_1 的 2 倍；∂_3 为 ∂_1 的 4 倍。

算法在 3 种任务粒度 $\partial_1,\partial_2,\partial_3$ 下的计算时间如表 1 所示。

表 1　3 种任务粒度下的计算时间

并行节点数	任务粒度		
	∂_1 /s	∂_2 /s	∂_3 /s
2	21.149	20.918	21.044
3	12.244	12.014	12.214
4	8.438	8.381	8.442
5	6.563	6.527	6.559
6	5.074	5.036	5.154
7	4.352	4.336	4.412
8	3.836	3.774	3.820
9	3.507	3.475	3.598
10	3.112	3.097	3.277

通过实验数据可以发现，不管在那种任务粒度下，并行计算的最少时间(分别为 3.112 s、3.097 s、3.277 s)都比算法串行执行所需计算时间 20.472 s 少(时间分别减少 17.360 s、17.375 s、17.195 s)，这说明并行算法有效地减少了运行时间。

另外，在同样计算量、任务分配策略的前提下，任务粒度的不同会导致实验结果出现差异，任务粒度 ∂_2 的计算时间相比其他任务粒度情况要少。计算时间比较结果如表 2 所示。

表 2　3 种任务粒度下的计算时间比较

并行节点数	计算时间/s	相比任务粒度为 ∂_1 时减少		相比任务粒度为 ∂_3 时减少	
		时间/s	比例/(%)	时间/s	比例/(%)
2	20.918	0.231	1.09	0.126	0.60
3	12.014	0.230	1.88	0.200	1.64
4	8.381	0.057	0.68	0.061	0.72
5	6.527	0.036	0.55	0.032	0.49
6	5.036	0.038	0.75	0.118	2.29
7	4.336	0.016	0.37	0.076	1.72
8	3.774	0.062	1.62	0.046	1.20
9	3.475	0.032	0.91	0.123	3.42
10	3.097	0.015	0.48	0.180	5.49

由表 2 可以看出，在各节点数情况下，任务粒度 ∂_2 比 ∂_1 的计算时间要少，这说明通过增加 Slave 每次的计算量，减少 Master 与 Slave 之间的通信次数，可以使算法性能有所提高；然而任务粒度 ∂_3 的计算时间却相比 ∂_2 增加了，这说明由于任务粒度过大导致的负载不平衡性加剧，使得算法的性能降低。因此，在确定任务粒度时，必须综合考虑通信和负载这 2 个因素的影响。

下面进一步讨论任务粒度 ∂_2 下的算法并行性能，计算得到不同并行节点数下的加速比和并行效率如表 3 所示。

表 3　任务粒度 ∂_2 下的加速比和并行效率

并行节点数	计算时间/s	加速比	并行效率/(%)
2	20.918	0.979	48.93
3	12.014	1.704	56.80
4	8.381	2.443	61.07
5	6.527	3.137	62.73
6	5.036	4.065	67.75
7	4.336	4.721	67.45
8	3.774	5.424	67.81
9	3.475	5.891	65.46
10	3.097	6.610	66.10

从表 3 可以看出，在 2 个节点进行实验时，计算时间为 20.918 s，比串行计算时间 20.472 s 还增加 0.446 s，这是由于在设计并行算法时，采用主从的任务分配模式，需要一个节点来组织任务的分配而不参与传输损耗的计算，因此该实验并没有实际的计算能力的提高，反而会带来一定的通信开销。另外并行计算时间从最初的 2 个节点并行 20.918 s 减少到 10 个节点并行 3.097 s，在节点机数目大于 7 以后，计算时间减少并不十分明显，这说明随着并行计算数目的增加，并行计算时间的减少会越来越不明显。

加速比随并行节点数变化的曲线如图 4 所示。并行效率随并行节点数变化的结果如图 5 所示。

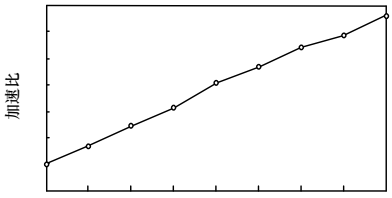

图 4　加速比随并行节点数变化的曲线

(下转第 266 页)