一种中尺度数值预报模式数据并行传输应用研究

赵磊　高松　吴铖　杜钦
（重庆市气象科学研究所，重庆 401147）

摘要：以重庆中尺度数值预报模式为例，设计开发一种针对中尺度数值预报模式海量数据的并行传输程序，提高数据传输的效率。重庆中尺度数值预报模式每天输出的大量数据需传输到数据管理服务器处理和备份。目前模式的数据传输任务是以scp单核串行程序为主，数据传输时间长，不能充分利用处理器核和网络带宽，未有效满足重庆天气预报业务对模式数据及时使用的需求；设计开发的并行传输程序以进程池为主要架构，采用Rsync代替scp提高模式数据传输速率；同时，基于输出的模式数据的特征实现多种数据分治策略，对比选择最佳的数据分治策略；结果表明基于最佳数据分治策略的并行传输程序能够极大减少数据传输时间，极大利用空闲的处理器核和网络带宽资源。

关键词：数值预报模式，并行传输，数据分治策略，Rsync

DOI：10.3969/j.issn.2095-1973.2019.06.003

Parallel Transmission of Data from a Mesoscale Numerical Prediction Model: An Application Study

Zhao Lei, Gao Song, Wu Zheng, Du Qin
(Chongqing Institute of Meteorological Sciences, Chongqing 401147)

Abstract: Taking the Chongqing Mesoscale Numerical Prediction model as an example, the authors designed and developed a parallel transmission program for large data of the mesoscale numerical prediction model to improve transmission efficiency. The large dataset the Chongqing mesoscale model produced has been transferred to the data management server for every day processing and analysis. The one-core serial program with scp was currently used to complete the task. Because it took more time and could not make full use of cpu cores and network bandwidth, this program could not meet the demand of timely use of Chongqing Weather Forecasting. The parallel transmission program takes process pool as main architecture, and adopts Rsync instead of scp to improve the data transmission speed. Meanwhile the model data is divided on the basis of data features, and the best data partition strategy would be chosen by comparison. The results showed that the parallel transmission program with the best data partition strategy can greatly reduce the data transmission time, and effectively improve the availability of cpu cores and network bandwidth.

Keywords: numerical prediction model, parallel transmission, data partition strategy, Rsync

0　引言

目前中尺度数值预报模式是局部区域天气预报业务系统中重要的组成部分，为天气预报业务提供技术支撑。重庆中尺度数值预报模式是以WRF（Weather Research and Forecasting Model）为核心模块开发的一种数值天气预报模式，是重庆精细化数值天气预报系统的重要组成部分，为重庆本地区域复杂天气变化过程预测预报提供依据和参考。重庆中尺度数值预报模式的运行平台是SGI Altix uv 1000系列大型机，该模式一天运行2次（00\12 UTC），预报时效96 h，该模式输出可用的业务数据是NetCDF格式的平面二维（2 Dimension）数据，大小13 GB左右/次，一天数据量约26 GB。根据业务和科研需求，输出的平面二维数据需及时传输到数据管理服务器上处理及归档，该数据管理服务器运行了多种数据处理串行程序，数据交换频次较高，但由于很多程序设计上的缺陷，仍有部分网络带宽资源和CPU核资源未被有效利用。目前，该类型数据的传输工作是采用基于SCP技术的单核串行程序实现，该程序能够长期稳定运行，传输单次中尺度模式数据实际耗时约2 h，传输延时长，造成数据

收稿日期：2018 年 5 月 21 日，修回日期：2019 年 7 月 16 日
第一作者：赵磊（1984—），Email：zhaoleicuit@126.com
资助信息：重庆市气象局业务技术攻关重点 / 团队 - 数值预报应用技术团队项目（YWGGTD-201716）；高性能计算机系统与数值预报系统建设项目（2016-2019）

无法及时被气象业务和科研人员参考和使用；且该程序也不能充分利用空闲的网络带宽和数据管理服务器资源。

因此，针对现有数据传输程序的缺点，本文设计开发一种新的并行数据传输程序，并对数值天气预报模式的数据有效分治划分，在不影响数据管理服务器中其它数据传输和处理程序的实际环境下，实现一种可应用业务的基于最佳数据分治策略的并行传输程序，有效提高中尺度数值预报模式数据的传输效率。

1 数据并行传输程序

1.1 并行传输网络架构

重庆中尺度数值预报模式的相关程序在以SGI大型机和曙光服务器节点所组成的高性能集群平台上运行；模式数据通过该集群千兆以太网络传输到数据管理服务器（图1），SGI大型机和数据管理服务器之间通过网络交换机连接，工作模式为全双工，网络整体带宽为1000Mbps。数据管理服务器是一台曙光高性能服务器节点，管理约37种数值预报相关业务数据资料的传输和处理工作，也管理着科研用户的天气过程试验资料；服务器架构是对称多处理器（Symmetrical Multi-Processing，SMP）[1-2]，具备16个处理器核（即CPU Core）。服务器节点之间基于基础的TCP/IP协议实现数据传输交互。

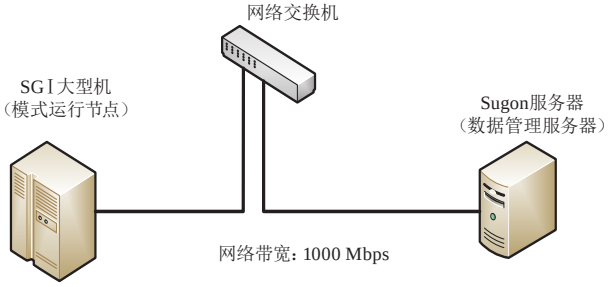

图1 并行传输网络拓扑

Fig. 1 The network topology of parallel transmission

1.2 并行传输程序

基于现有有限的网络和数据管理服务器资源，构建的并行传输主程序按照预先设计的参数构建一进程池，该进程池中每一进程只占用一个处理器核，数据传输任务由主程序依次分配给进程池中的进程去执行，进程池中各个进程间相互独立，在执行任务时它们之间没有信息交互规程，是一种易并发执行过程[3-4]；传输任务完成后的进程处于空闲态并返回进程池中，进程池将空闲态的进程信息反馈给主程序，主程序检查传输任务列表，并将新的传输任务分配给空闲态的进程执行；当任务列表中无数据传输任务，且所有传输任务的进程都执行完毕，进程池将所有空闲态的进程信息反馈给主程序，由主程序回收所有进程并关闭进程池。

该并行传输程序采用Python语言和Rsync同步技术[5-6]实现数据传输流程。Python是目前较流行的一种高级解释型编程语言，运行该语言编写的程序调试运行的基础是在相应的操作系统平台安装的Python解释器[7]；该语言拥有众多开源的软件包，非常适合多领域的编程开发和科研人员使用，本程序采用multiprocessing包[8]，通过multiprocessing.Pool函数构建一个含有多个进程的进程池；Rsync是Linux系统中非常流行的数据同步传输和备份技术，拥有SSH（Secure SHELL）和C/S架构两种配置应用方式，能实现数据的快速复制、远程复制、文件传输和安全文件传输等功能，以及相同局域网内多个服务器之间的数据快速传输及同步功能，Rsync使用一种远程更新协议，该远程更新协议采用一个有效的总和检查搜索算法，该算法允许Rsync通过网络链接仅传输两端之间数据的不同部分，且可以压缩和校验数据[5-6, 9]；本程序的子功能模块函数采用第一种配置应用方式，即在两台服务器之间采用SSH协议[6]实现身份验证，身份验证通过后，调用Rsync实现服务器端到端的数据传输。

该并行传输主程序的流程（图2）在启动初始化后，根据试验测试环境要求灵活配置处理器核数和划分分治数据包大小。因每一进程占用一个处理器核，配置处理器核数可理解为设置进程池中进程数目。从并行传输主程序的源码可看出，在初始化和配置相关变量后，process_nn为处理器核数，dip_n为分治的数据包大小，主程序首先通过create_list 函数创建dip_n分治数据包大小的传输任务列表，接着通过Python的Pool函数创建一个进程数为process_nn的进程池；传输任务列表中任务由主程序分配给具体的进程，此过程由apply_async函数完成。因数据管理服务器的处理器核资源有限，划分分治后的数据传输任务数目均会大于进程池中进程数目，进程池中进程并发执行后，具体执行数据传输的函数是rsy_2d_list，在该函数中采用Rsync检测和传输数值预报模式的输出数据；主程序一直等待进程池中出现空闲态进程，若传输任务列表里仍有待执行的任务，程序及时再将列表里的传输任务分配给进程池中的空闲态进程；并行传输主程序循环反复检测传输任务列表和进程池中进程态，直至传输任务全部执行且进程池中进程均为空闲态后，并行程序关闭进程池和回收进程资源，整体的传输程序

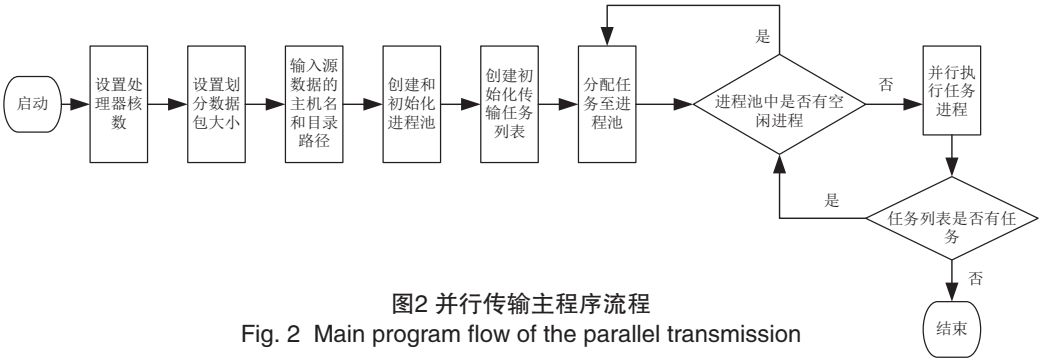

图2 并行传输主程序流程
Fig. 2 Main program flow of the parallel transmission

流程结束。

```
#并行传输主程序的源码
if __name__ == '__main__':
 startt=int(time.time())
 dip_n=sys.argv[1]  # 设置模式2d数据分治数据包大小值
 process_nn=sys.argv[2]  # 设置并行程序的CPU资源数
 nn=string.atoi(dip_n)
 process_n=string.atoi(process_nn)
 # filename存储2d数据包含的气象要素种类，包含温、压、湿、风以及降水等
 list_result=create_list(filename,nn) # 根据分治数据包大小值，通过该函数将2d数据按照气象要素种类进行划分，划分后返回一个列表对象，供下面使用
 print list_result  #输出列表，检查2d数据划分是否准确
 log_file=log_rsync_data2d.log  # 指定数据传输的日志记录文件
 len_result=len(list_result)-1 # 确定列表最后一个元素的下标，作为任务分配终止逻辑判断条件参数
 pool=multiprocessing.Pool(processes=process_n) # 利用python的multiprocessing创建进程池，该进程池包含了prpcess_n个进程
 print "start allocate processes to job\n"

 while len_result >= 0:  # 未到达划分分治列表的最后时，循环向进程池分配数据传输任务
  print "----rsync_2d_list-----"
  pool.apply_async(rsy_2d_list,(list_result[len_result],))  # 将划分列表中的元素传参给传输任务，将传输任务分配到进程池中的一个进程，并开始启动进程
  len_result=len_result-1
  print list_result[len_result]
 pool.close()  # 当所有的数据传输任务进程完成后，关闭进程池
 pool.join()  # 释放进程池所占资源，回调到主程序中，继续执行主程序下面代码
 endtt=int(time.time())
 print "sub processes done and exit\n"
 print "total run: %d seconds\n"%(endtt-startt)
 if os.path.isfile(log_file):
  output_f=open(log_file,'a')
  output_f.write("%d core  %d elment %d\n"%(process_n,nn,endtt-startt))  # 计算并行的运行时间，并将时间写入日志记录中
  output_f.close()
```

2 数据划分与分析

数据及问题的划分分治在并行程序设计中占有重要地位，是并行程序开发过程中首要解决的问题[3]，合理的数据划分分治将大幅提高并行程序的工作效率，本节重点开展针对重庆中尺度数值预报模式数据划分分治及对比分析。

2.1 划分分治

重庆中尺度数值预报模式输出的数据格式为NC格式二进制的2d（2 dimension）数据文件，该模式一次输出的数据量约13 GB，整体数据由18444左右个小文件组成，文件数量庞大而无序，每一小文件约700 K左右，表示不同时空分辨率下的一种气象要素场（如温、压湿、风、降水）数据，该数据共包含气象要素场100多种，包含业务需求的两种空间分辨率（27 km和3 km）的数据，27 km类型的预报时间分辨率为3 h，3 km类型的时间分辨率为1 h，因此，针对同一种气象要素场，空间分辨率3 km的数据量要大于空间分辨率27 km的数据量。根据上述的数据特征，本文曾依照空间分辨率划分数据，划分后的数据包粒度太粗，且3 km分辨率数据传输时间始终会大于27 km分辨率数据传输时间，实际并行程序时间以

3 km分辨率传输时间为准，程序整体运行负载不均衡；按照文件总数进行划分，又无法快速识别业务上优先级别高的气象要素场数据是否完整可用。最终寻求将数据的气象要素场种类作为划分的依据开展数据的划分分治。

数据分治划分流程（图3）和划分程序源代码详细阐述了模式数据的划分的过程。其中基础的是获取模式2d数据气象要素场的类型文件，此文件包含了传输的模式数据的所有要素场名称。划分分治的功能函数是create_list，该函数需设置两个参数，模式数据的气象要素场种类文件参数filename和划分分治的数据包大小参数*n*；在初始化列表变量后，读取和处理气象要素场类型文件中的要素场，生成一个初始的列表存放在内存，按照变量*n*值的大小，计算划分分治数icount和分治后的余数rest_n，通过初始列表的下标值nex_i和pre_i，尽可能均等划分初始列表中的气象要素场，并将划分后的结果依次存储于传输任务列表中，最后将划分分治后的传输任务列表返回给主程序。

```
#数值模式数据划分功能——划分程序源代码
def create_list(filename,n):
 # 初始化该程序中所要使用的变量。
L_re=[]
 new_L_re=[]
 new_L_re01=[]
if os.path.isfile(filename):  # 判断2d数据要素文件是否存在。filename提取和存储模式2d数据包含的所有的气象要素。
  print "the file is exist\n"
  # 读取气象要素的数据到内存中，通过两个循环将要素数据有效存储于初始列表中，去除异常字符，便于划分操作
  ffile=open(filename,'r')
  for line in ffile:
   line=line.strip("\n").split(",")
   L_re.append(line)
  ffile.close()
  for list_e in L_re:
   str_e=str(list_e).strip("[]").strip("'")
   new_L_re.append(str_e)

  icount=len(new_L_re)/n   # 根据设置的划分数据包大小，确定划分分治数
  rest_n=len(new_L_re)%n  # 划分后，确定最后一项数据包大小
  while icount > 0:
   if rest_n == 0:    # 当余数为0时，2d数据会被均等划分
    nex_i=icount*n
    pre_i=(icount-1)*n
    new_L_re01.append(new_L_re[pre_i:nex_i])
   if rest_n != 0:  #当余数不为0时，最后一项数据包大小值不等于n，而是rest_n,但rest_n只能小于n，所以整体不会延长数据并行传输时间。
    pre_i=icount*n
    nex_i=pre_i+rest_n
    new_L_re01.append(new_L_re[pre_i:nex_i])
    rest_n=0
   icount=icount-1
  else:
   print "the file not exist"
  del L_re
  del new_L_re
 return new_L_re01  # 返回根据n划分好的传输任务列表
```

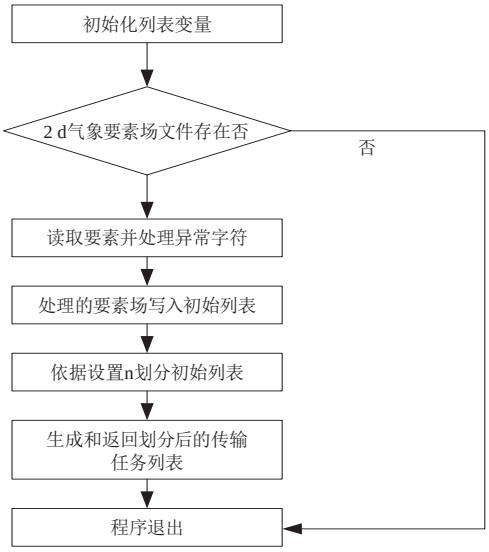

图3 数据划分流程

Fig. 3 The flow of data partition

依据模式数据气象要素场的类型，划分原则为一方面划分的数据包包含的气象要素场类型不能太多，避免数据包粒度变粗，影响传输程序执行效率；另一方面尽可能实现均等划分，确保传输并行程序负载均衡；基于上述原则本文将模式数据划分为4种不同粗细粒度的分治数据包，形成4种数据分治策略（表1），其中分治数据包大小是指每个分治策略下分治数据包包含的气象要素场类型数目，划分分治数是指每个分治策略下总共划分多少个分治数据包，4种分治策略

中最细粒度的是分治数据包含1种气象要素场，最粗力度的分治数据包包含6种气象要素场；下面将对比分析这4种不同数据分治策略在不同的处理器核数下的运行趋势。

表1 数据分治策略

Table 1 The data partition strategy

分治策略	数据分治策略1	数据分治策略2	数据分治策略3	数据分治策略4
分治数据包大小	1	2	3	6
划分分治数	174	87	58	29

串行单核的数值模式传输程序平均运行时间约7810s，数据传输时间很长，这主要是因为现有数据管理服务器从一个物理网络接口每天传输交互约37种数值预报模式相关的数据，物理网络接口负载过大的导致的，但网络带宽并没有最大化的利用。本文将4种不同数据分治策略应用到并行传输程序中，通过运行时间、加速比和并行效率三种指标对比分析，综合选择最佳分治策略；加速比（S）是衡量并行传输程序一个重要指标，是指单核运行时间（Ts）和多核并行运行时间（Tp）的比值，该值越大表明并行运行时间越短[2, 10-11]；并行效率E是在加速比上的基础上引入了每次并行使用的处理器核数n，实质上是单核运行时间和多核并行运行时间乘以核数n的比值，再乘以100%，主要描述并行程序运行过程中系统开销时间的占比[4, 11]。

2.2 试验和对比分析

试验测试环境为实际的业务应用环境，在不影响数据管理服务器其他数据传输交互业务的实际情况下，极大化的占用空闲态的处理器核资源和网络带宽。在实际应用环境下测试，发现当并发执行进程数超过8时，数据传输会出现大量丢包现象，进程数设置超过10时，数据传输完全中断失败，这说明进程数设置为8以上的值时，便无服务器资源和网络带宽可供分配使用，且逐渐开始占用其他数据传输处理程序的资源，所以为了实现业务应用环境下本并行传输程序改善的实际效果，设置的进程数的阈值为8。服务器节点操作系统均为SUSE 11 Linux x86_64，服务器节点均安装Python，并实现了SSH身份验证，试验分为三种类型，为了保证试验效果和试验数据的可信度，试验过程中处理器核数递增间隔为2；对比分析过程中，各个指标的对比分析图中将离散试验数据用不同的类型曲线连接，以便直观展示各分治策略的变化趋势。

不同数据分治策略下的并行运行时间均优于单核串行程序（图4）。各分治策略随着核数增加，并行运行时间逐渐缩小，当处理器核数从2到4时，运行时间降幅较大，运行时间差2000多秒，从4增至8时，并行运行时间降幅度明显缩小，时间差均在300 s左右，从各分治策略的转折线可看出，核数的增加并不意味着运行时间会以相同的速度减少，主要是随着核数的增加，系统在分配任务的进程数增多到一定程度，进程间通信开销大幅度增长，但系统总线只有一条，所以才导致并行运行时间降低力度缩小，运行效率开始渐渐降低；分治策略1、2、4运行时间趋势一致且几乎重合，在超过4个处理器核后，趋势线的斜率有明显一致的变化。分治策略3有点不同，在处理器核数由2增至6时，趋势线斜率变化相对而言变化不大，增至到8时，趋势线斜率和其他策略保持一致，这说明随着核数增加，分治策略3的运行时间降低力度明显，且运行时间开始逐渐优于其他分治策略。

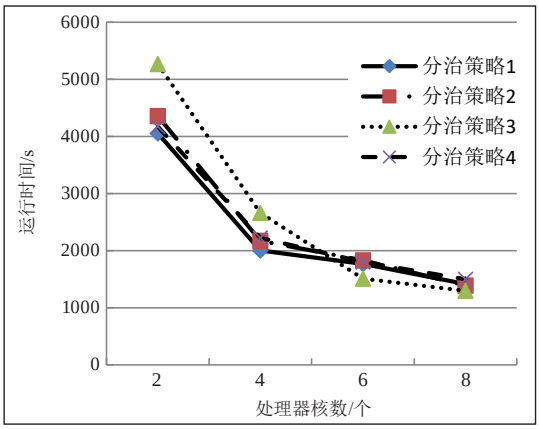

图4 数据分治策略运行时间变化趋势

Fig. 4 The data partition strategies runtime versus number of cores

分治策略的加速比和并行时间变化趋势有一定的关联（图5）。随着处理器核数的增加，各分治策略下的加速比均呈现非线性的上升趋势。从图3中折线的斜率可看出，分治策略1、2、4的处理器核数从2增至4时，加速比变化幅度较大，数据传输效率提升效果明显，核数从4增加到8时，加速比变化趋势发生了转折，加速比变化幅度相对减小；分治策略3的加速比变化趋势有点不同，当处理器核数增加到6时，加速比变化趋势最大，核数从6增至8时，加速比趋势趋于平缓；分治策略3使用处理器核数增加到4时，加速比值均低于其他分治策略，随着处理器核数不断增加，加速比值逐渐提高，并最终优于其他分治策略，在超过4个处理器核数后，该分治策略对数据的传输效率改善效果最大。加速比趋势变化和上述的并行运行时间的趋势变化类似，说明并行运行时间变化趋势决定加速比变化趋势。

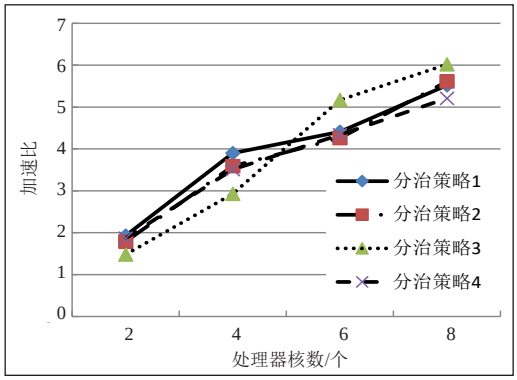

图5 数据分治策略加速比变化趋势

Fig. 5 The data partition strategies acceleration ratio versus number of cores

最后，对比分析一下不同分治策略的并行效率（图6），随着处理器核数增加，分治策略的并行效率均呈现不同的下降趋势。主要因为即随着处理器核数资源占用越来越多，进程数越多，并行程序需耗费更多系统开销时间来完成资源调度分配，导致数据并行传输处理时间所占比重降低；分治策略1、2、4呈现的趋势较为一致，并行效率从90%左右逐渐降低至70%以下，且是非线性的变化趋势，处理器核数增至4个以上时，并行效率降低明显；而分治策略3却呈现另外一种变化趋势，它的并行效率虽然整体上也是逐渐降低，但在核数增至到6时，并行效率开始大幅提升，这一点和其策略其它两个指标有一定的关联，该分治策略前两个指标可以看出，当核数从4增至6时，该分治策略发生较大变化，指标值从最差变为最佳，系统开销时间占比相对于整体并行传输时间最低；从整体并行效率变化趋势看，分治策略3并行化效率变化范围控制在70%～85%，整体变化范围比较稳定，其他分治策略并行效率的变化范围和降低力度相对较大。

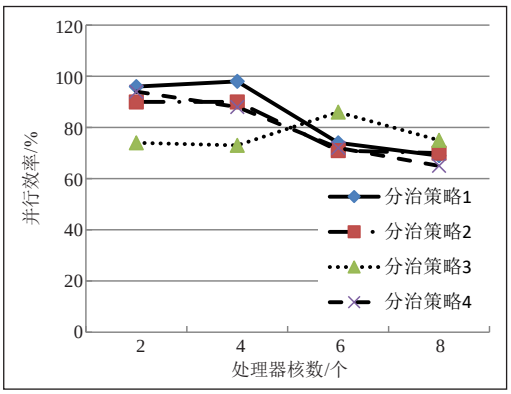

图6 数据分治策略并行效率变化趋势

Fig. 6 The data partition strategies parallel efficiency versus number of cores

为充分利用现有空闲网络带宽和服务器处理器核资源，实现数据传输运行时间最短和并行传输程序稳定和最佳并行效率，综合对比上述并行运行时间、加速比和并行效率三个指标，分治策略3是并行数据传输应用的最佳策略，基于该分治策略的并行传输程序能够能将传输时间从7810 s缩小至1297 s，加速比高达6.02，网络带宽利用率提高了6倍左右，且并行效率维持在75%左右，数据管理服务器的处理器核利用率达到50%。

3 结论

以重庆中尺度数值天气预报模式为例，中尺度数值预报模式的数据并行传输程序在现有网络和计算机资源基础上，采用Python和Rsync数据传输同步技术构建和实现；根据模式数据的特征和实际业务应用环境，将模式数据划分为4种分治策略，对比分析4种不同的分治策略，综合选择最佳的数据分治策略，最终实现最优的模式数据并行传输。最优数据并行传输一方面极大减少模式数据并行传输时间，提高模式数据传输效率，另一方面也较充分利用数据管理服务器中空闲的处理器核和网络带宽，为中尺度数值预报模式海量数据高效率传输备份提供一种解决方案。但重庆精细化数值天气预报系统不仅包含重庆中尺度数值预报模式，按照建设要求还将包含其他不同尺度多类型数值天气预报模式，传输的数据量也将大幅增大，未来将进一步研究基于MPI的进程组间无阻塞通信方式的更加通用的并行传输应用技术，以满足重庆精细化数值天气预报系统的更大容量数据高效率传输需求。

参考文献

[1] Michael R K. 李德龙, 译. 精通UNIX shell 脚本编程(第二版). 北京: 清华大学出版社, 2010.
[2] Wilkinson B. 陆鑫达, 译. 并行程序设计(第二版). 北京: 机械工业出版社, 2006.
[3] Wesley J C. 宋吉广, 译. Python 核心编程(第二版). 北京: 人民邮电出版社, 2007.
[4] 张云泉. 并行计算模型与算法. 北京: 机械工业出版社, 2016.
[5] 金之雁, 颜宏. 并行环境中的有限区域数值预报模式并行计算试验. 高原气象, 1996, 15(1): 21-27.
[6] 何骞, 卓碧华. 一种远程文件同步方法. 计算机应用, 2012, (2): 566-568.
[7] 杨健才, 张华, 金之雁. 高分辨率下数值预报模式并行计算方法研究. 甘肃气象, 1999, 17(1): 16-18.
[8] 喻炜, 颜宏, 金之雁. 并行效率初步研究. 应用气象学报, 1996, 7(1): 61-68.
[9] 周昆, 王东勇, 朱红芳, 等. 中尺度数值模式在IBMP690上的并行测试. 气象科学, 2006, 26(6): 651-654.
[10] The Python Software Foundation. multiprocessing---Process-based “threading” interface [EB/OL]. [2014-08-17]. https://docs.python.org/2/library/multiprocessing.html
[11] 林玉成, 赵瑞, 罗兵. 基于Rsync的中央气象台数据备份机制及优化设计. 高原山地气象研究, 2014, 34(1): 81-85.