刘凡鸣，郭瑞强，李永庆，边鹏飞，2015．基于 MapReduce 的地震波形数据并行解压缩算法研究．震灾防御技术，**10**（2）：344—352．doi：10.11899/zzfy20150214

基于 MapReduce 的地震波形数据并行解压缩算法研究[1]

刘凡鸣[1)] 郭瑞强[1)] 李永庆[2)] 边鹏飞[2)]

1）河北师范大学数学与信息科学学院，石家庄 050024
2）河北省地震局，石家庄 050021

摘要 近年来各省级地震台网 SEED 文件数据量急增。在数据处理过程中，利用原有的串行解压缩算法批量解压缩地震波形数据时存在操作繁琐、耗时较长的问题。本文引入了 MapReduce 并行编程模型，根据该编程模型思想结合原有串行解压缩算法，提出了一种并行解压缩地震波形数据的算法，并给出了算法的设计与实现。本文从正确性、运行效率以及可扩展性三个方面进行了对比实验，验证了使用并行算法解压缩数据的效率较高，并且能够一次实现批量地震波形数据的解压缩，且操作简单。

关键词：地震波形数据 解压缩 并行 MapReduce

引言

目前，中国数字地震监测网络的测震台站数量已达 1000 多个，它们为测震台网中心提供了大量的波形数据。各省级测震台网中心在完成地震速报和编目处理后，将这些波形数据归档成 SEED（The Standard for the Exchange of Earthquake Data，地震数据交换标准）（中国地震局，2003）格式用于地震科研。单是省级测震台网中心就已经积累了庞大的波形数据，以河北省地震局为例，目前已经积累了 10TB 左右的波形数据，而且还以约 0.6TB/年的速率增长。其中，“台站卷”归档了单个台站的连续波形，“事件卷”归档了多个台站对同一地震事件的记录波形。随着数字地震波形的广泛使用，对地震精定位、波形互相关分析、重复地震、波速比、地脉动噪声成像、震动图快速计算、震源机制解、震源破裂过程反演等方面的研究越加深入。在进行上述研究时，首先会将压缩格式的 SEED 波形还原成数字序列，因此需要处理的台站数量也越来越多。而原有的 SEED 解压算法属于单文件、单线程操作，极大制约了数据处理工作的效率。

Hadoop 是 Apache 软件基金会旗下的一个开源的分布式计算平台（White，2012），其核

1 **基金项目** 河北省重点地区壳幔结构及地震监测预报关键技术研究（13275407D）、河北省教育厅自然科学研究项目（QN20131141）和河北师范大学应用开发基金项目（L2012K01）联合资助
［**收稿日期**］ 2014-11-03
［**作者简介**］ 刘凡鸣，女，生于 1990 年。河北师范大学数学与信息科学学院硕士研究生。主要研究方向：数据挖掘、分布式计算。E-mail：mingl.0219@163.com
［**通讯作者**］ 郭瑞强，男，生于 1974 年。河北师范大学数学与信息科学学院副教授，博士，硕士生导师，中国计算机学会（CCF）会员（E200017546M）。主要研究方向：数据挖掘、WEB 智能系统。E-mail：rqguo@126.com

心组件包括分布式文件系统（HDFS）和 MapReduce 编程模型。其中 HDFS 具有可靠、可扩展等优点，因此用户可以将多台廉价的硬件部署成并行处理集群（Ghemawat 等，2003）。而 MapReduce 是一个可以在集群上处理大规模数据的并行编程模型，它借鉴了函数式编程思想，其中分布式系统底层细节对用户是透明的，因此用户只需编写函数式程序就可以进行并行程序的开发（Dean 等，2008）。利用 Hadoop 架构提供的 MapReduce 编程模型，可将单文件、单线程的 SEED 解压缩工作转变为计算机集群上多文件、多线程的并行处理，因此可以极大地提高解压缩效率。特别是针对测震台网数据服务中心在多用户环境下提供数据服务时，这种效率的提高更加明显。

目前 Hadoop 技术在地震相关领域已经有了初步使用，文必龙等（2014）提出了非结构化地震数据在 Hadoop 分布式平台上的存取设计方案，该方案采用了混合索引查询方法进行统一访问，提高了数据的存储效率。赵长海等（2012）探讨了 MapReduce 对石油勘探领域应用算法的适用性，并采用 MapReduce 处理地震数据用以获取地下构造，从而实现石油勘探。由于地震波形数据文件属于半结构化文件，而处理半结构化文件又是 MapReduce 的优势之一，所以 MapReduce 比较适合以批处理的方式解决问题。本文引入了 MapReduce 的编程模型（李闯等，2010），同时根据编程模型和原有串行解压缩算法，提出了一种并行解压缩地震波形数据的算法（以下简称 PDSWD 算法），并给出了算法的设计与实现。笔者从正确性、运行效率以及可扩展性三个方面对算法进行了对比实验，结果表明使用该算法解压缩数据时效率较高，并且能够一次实现批量地震波形数据的解压缩，且操作简单、方便实用。

1　地震波形数据的串行解压缩算法

中华人民共和国地震行业标准《地震波形数据交换格式（DB/T 2-2003）》（中国地震局，2003）中规定了 SEED 的格式，因此本文不再介绍。SEED 文件采用 Steim2 压缩算法（Mauro 等，2006），这种算法既能节省存储空间，又能保证数据信息不丢失。而地震波形数据的解压缩算法是 Steim2 压缩算法的逆过程（王洪体等，2004），首先需要计算样本个数，然后提取样本序列的第一个值并获取编码方式，利用差值序列重建原始的 32 位数据样本序列。

2　基于 MapReduce 的地震波形数据并行解压缩算法

2.1　MapReduce 模型

MapReduce 是一个可以在集群上处理大规模数据的并行编程模型，其分布式系统底层细节对用户是透明的。MapReduce 主要包括 Map 和 Reduce 两个阶段，用户只需根据自己的需求编写相应的 Map 和 Reduce 程序，就可以进行并行程序的开发。

输入数据可以看成是若干个<key，value>对的集合，MapReduce 的工作原理体现了分治思想，将输入数据切分成若干片，然后交由集群内的不同节点同步处理，以此实现程序的并行化，其处理流程如图 1 所示。在 Map 阶段，MapReduce 可根据用户自定义的 Map 函数，让每个 Map 节点处理由若干个<key，value>对构成的分片，而输出的中间结果是新的<key，value>对集合，处理过程可表示为：

$$\text{Map: } (k1, v1) \rightarrow \text{List } (k2, v2)$$

之后混洗操作将 key 值相同的<key，value>对聚集到一起传递给 Reduce。各个 Reduce 节点再根据用户自定义的 Reduce 函数，处理具有相同 key 值的 value 集合，最后生成新的<key，value>

对集合输出，处理过程可表示为：

$$\text{Reduce: (k2，List(v2))}\rightarrow\text{List (k3，v3)}$$

图 1 是 MapReduce 的数据流程。

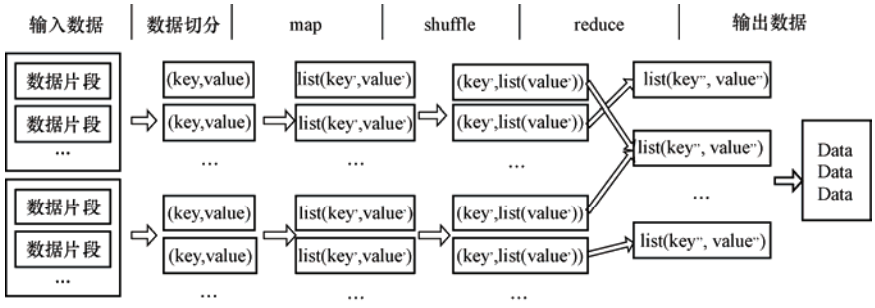

图 1 MapReduce 数据流

Fig. 1 MapReduce data flow

2.2 并行解压缩算法

由于输入数据中每个通道中的各条数据记录相互独立，在解压缩过程中互不影响，因此 SEED 文件适合并行化处理，可以将解压缩算法移植到 MapReduce 上使用。并行解压缩地震波形数据的基本思路是：利用 MapReduce 编程模型的分治思想，Map 阶段解压缩每个分片中的数据记录，然后将中间结果传递给 Reduce 阶段进行规约，拼接各通道数据得到最终结果。解压缩过程如图 2 所示。

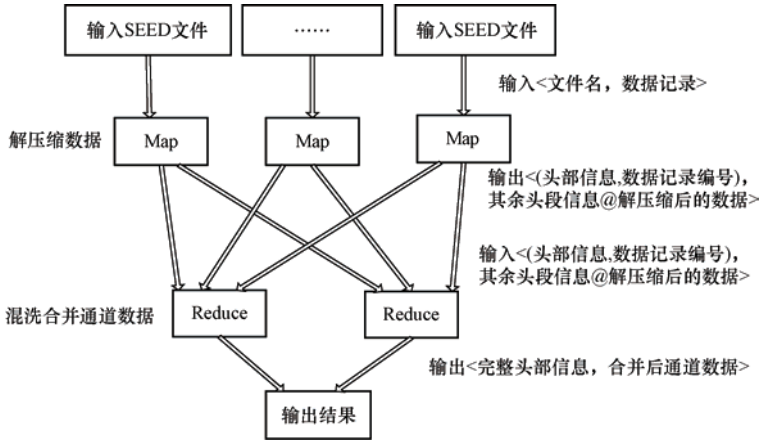

图 2 并行解压缩过程

Fig. 2 Flowchart of parallel decompression process

以下是 MapReduce 各阶段的流程描述：

（1）输入：地震事件波形 SEED 文件，包括台站、位置、台网、通道、时间、样本数据、采样率以及压缩数据等。

（2）Map 阶段：并行解压缩每个分片中的各条数据记录，得到原始样本数据，根据输入文件读取每条数据记录所属文件的文件名，以及台站、位置、台网、通道的信息，将这些信

息作为中间结果的 key，将从每条数据记录中读取的其余头部信息和解压缩后得到的数据作为中间结果的 value。

（3）Reduce 阶段：远程拷贝 Map 阶段输出的中间结果，把 key 相同的值对聚集到一起，并按照数据记录序号进行排序，然后将数据传递给 Reduce 节点，根据用户自定义的 Reduce 函数，将解压缩后得到的各通道数据按照时间的先后顺序进行拼接。

（4）输出：输出文件中包含每个台站的各个通道解压缩得到的原始样本，并按采样顺序排列。

2.3　Map 阶段处理过程

2.3.1　自定义 Inputformat

Inputformat 是 MapReduce 的一个重要接口，其中包含输入数据切片方法以及每个从分片读取键值对的方法。通常系统默认的是按行提取键值对，即<key，value>中的 key 代表行偏移量，value 代表该行内容。但是由于 SEED 文件中既有 ASCII 格式数据，又有二进制格式数据，而二进制数据中没有换行的概念，因此现有的按行提取键值对的方法不能满足需求，需要自定义适合地震数据处理的 Inputformat。因为解压缩操作使用的是 SEED 文件的数据块部分，所以在读取键值对的方法中设置跳过文件的 4 个控制头段，将数据块的第一个字节处作为起始位置，每 4096 个字节为一条记录，每条记录作为 value，该条记录所在文件名作为 key，将分片解析成<文件名，数据记录>这样的<key，value>对格式作为输入。

2.3.2　Map 函数

Map 函数接收从分片中读取的<文件名，数据记录>对，并将其作为输入，通过用户自定义的 Map 函数解压缩每条数据记录，并读取数据记录中的属性信息。因为每个通道包含的数据记录不止一条，所以需要根据时间的先后，将各通道内数据记录解压缩后的结果进行拼接。同一通道内数据记录的编号可以反映出时间的先后，编号越小则记录的时间越早，因此把数据记录的编号作为排序依据。

MapReduce 中 Map 的输出只会按照 key 排序，不会根据 value 进行排序。而本文中不仅需要按照 key 排序，在拼接数据时还需要按照数据记录编号进行排序，但数据记录编号在 value 中保存，因此就需要使用二次排序。使用二次排序时需要定义一个组合 key，在本文中组合 key 包括需要首先排序的头部信息，即原始 key，以及之后需要排序的 value 中的数据记录编号。

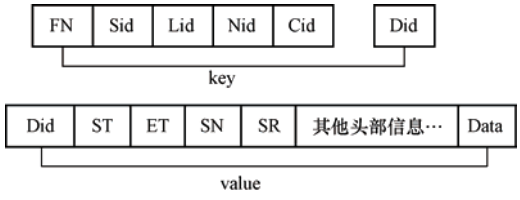

图 3　Map 输出格式

Fig. 3　The format of Map output

因此 Map 阶段输出的中间结果格式为<(头部信息，数据记录编号)，其余头部信息@解压缩后数据>。Map 阶段输出结构如图 3 所示，FN-Sid-Lid-Nid-Cid 和 Did 形成组合 key，FN-Sid-Lid-Nid-Cid 作为组合 key 的第一个值，每个编码间用分隔符“-”隔开，Did 作为第二个值。其中 FN 为文件名称，Sid 为台站编码，Lid 为位置编码，Nid 为台网编码，Cid 为通道编码，Did 为数据记录编号。其余头部信息@解压缩后数据作为 value 输出，将其余头部信息和解压缩后数据之间用分隔符“@”隔开，其余头部信息各编码之间用分隔符“-”隔开。其中 ST 为采样开始时间，ET 为采样结束时间，SN 为样本数目，SR 为采样率，由于剩余的头部信息与本文算法关联不大，这里就不再一一展开介绍。Data

为单条数据记录解压后数据。因此 Map 输出结果为<(FN-Sid-Lid-Nid-Cid,Did), Did-ST-ET-SN-SR-剩余头部信息@解压缩后数据>。

以下是 PDSWD Mapper 算法：

输入：<文件名，数据记录>

输出：<(FN-Sid-Lid-Nid-Cid,Did),Did-ST-ET-SN-SR-剩余头部信息@解压缩后数据>

步骤：

（1）读取 value 值；

（2）获取分片所属文件的文件名 FN；

（3）读取台站编码 Sid，位置编码 Lid，台网编码 Nid，通道编码 Cid，数据记录编号 Did；

（4）计算数据记录中包含的样本个数 SN，然后解压缩数据记录，得到原始数据样本 data=decode(d,SN,false)；

（5）获取数据记录的开始时间 ST，采样率 SR，并根据这两个值计算结束时间 ET；

（6）输出中间结果 Context.write((FN-Sid-Lid-Nid-Cid,Did)，Did-ST-ET-SN-SR-剩余头部信息@解压缩后数据)。

2.4 Reduce 阶段处理过程

Map 阶段后会将具有相同 key 值的键值对分发到同一个 Reduce 节点，在 Reduce 节点上拼接属于同一通道的数据记录解压后得到的数据。由于地震数据在采样过程中可能会发生异常（如两条数据记录间出现时间重叠或者时间间隔的情况），因此不能直接按照数据记录编号从小到大的顺序直接拼接，需要先比较上一条数据记录的结束时间和本条数据记录的开始时间。如果两个时间相同，则按数据记录编号从小到大的顺序直接拼接两条记录中解压出的数据。如果上一条的结束时间和本条的开始时间之间存在时间间隔，则首先计算出时间间隔，并根据采样率计算在该时间间隔内能采集到的样本个数，再在两条数据记录间补充相同个数的 null。如果本条的开始时间比上条的结束时间还早，应计算出两者重叠的时间段，再根据采样率计算出在该时间段内能采集到的样本个数，然后在上一条数据记录尾部去除相同个数的数据，最后再拼接本条数据记录中解压缩得到的数据。输入键值对为<(FN-Sid-Lid-Nid-Cid, Did), Did-ST-ET-SN-SR-剩余头部信息@解压缩后数据>，输出键值对为<全部头段信息，按台站通道拼接后得到的原始样本数据>。

以下是 PDSWD Reducer 算法：

输入：<(FN-Sid-Lid-Nid-Cid,Did),Did-ST-ET-SN-SR-剩余头部信息@解压缩后数据>

输出：<全部头段信息，按台站通道拼接后得到的原始样本数据>

步骤：

（1）读取组合键中的第一个 key 值以及 value 中除解压缩后数据以外的所有头部信息，将两者合并得到全部头段信息；

（2）读取 value 集合中第一个 value 值，获取结束时间 ET 以及解压后数据；

（3）读取集合中下一个 value，获取开始时间 ST、结束时间 ET 以及解压后数据，通过比较本条的开始时间与上一条的结束时间来决定数据拼接方法；

（4）重复步骤（3），直到读完所有 value 值，得到按通道拼接好的原始样本数据；

（5）输出最终结果 Context.write(<全部头段信息，按台站通道拼接后得到的原始样本数据>)。

3　实验结果及分析

3.1　实验环境及数据

由于条件限制，实验使用由 6 台 PC 机搭建的集群环境，其中 1 台机器作为主 NameNode 节点和 ResourceManager 节点，1 台机器作为备份 NameNode 节点，另外 4 台机器均作为 DataNode 节点和 NodeManager 节点。每台机器的配置相同，操作系统为 SUSE Linux Enterprise 11SP3，CPU 型号为 Intel 双核 6600、2.40GHz，内存为 4GB，硬盘容量为 1T，Hadoop 版本为 2.2.0，开发环境 eclipse+Hadoop plugin。

实验数据采用河北省地震局提供的 2013 年 9 月部分 SEED 事件波形数据，共包含文件 140 个，文件总大小 2.7G。

3.2　结果分析

3.2.1　解压正确性验证

为验证本文方法的正确性，使用 PDSWD 算法和原有解压缩算法分别解压缩相同的输入数据，PDSWD 算法运行在 Hadoop 集群上。每次都将使用 PDSWD 算法得到的输出样本值和原有方法解压缩得到的输出样本值做比较，结果表明两组样本值均完全相同，这就验证了 PDSWD 算法的正确性。但由于解压缩后数据样本较多，为便于可视化显示，笔者随机选取了一个开始位置，从开始位置处连续选中 30 个样本数据，并将这两组数据样本分别绘图后得到了如图 4 所示的曲线，进一步验证了 PDSWD 算法的正确性。

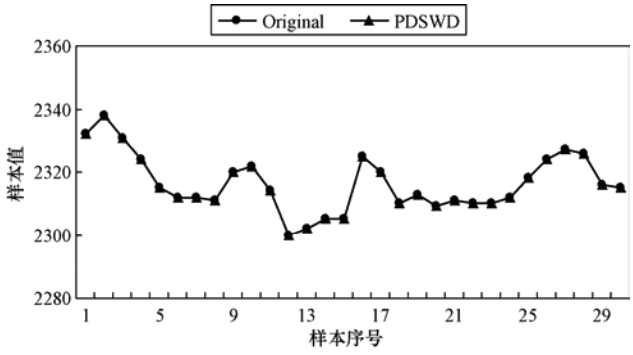

图 4　正确性验证

Fig. 4　Correctness verification

3.2.2　运行效率对比实验

为了证明本文提出的并行解压算法的效率，笔者将 PDSWD 算法与原有的串行解压缩算法的运行效率进行了对比。PDSWD 算法运行在由 PC 机搭建的分布式集群上，原有串行解压算法运行在单台机器上，分别选取 19M（1 个文件）、125.6M（5 个文件）、454.5M（20 个文件）、1126.4M（60 个文件）、2764.8M（140 个文件）数据作为输入，观察运行时间变化。图 5 是它们的运行时间对比，其中 PDSWD-3 表示在 3 节点的集群中使用 PDSWD 算法，PDSWD-6 表示在 6 节点的集群中使用 PDSWD 算法，Original 表示原有串行解压缩算法。当输入数据较小时，原有串行解压算法和 PDSWD 算法所用时间基本相同，因为作业的启动和交互需要消耗较多资源；但随着输入数据的增大，并行解压缩算法的工作效率逐渐高于串行

解压缩算法的工作效率。

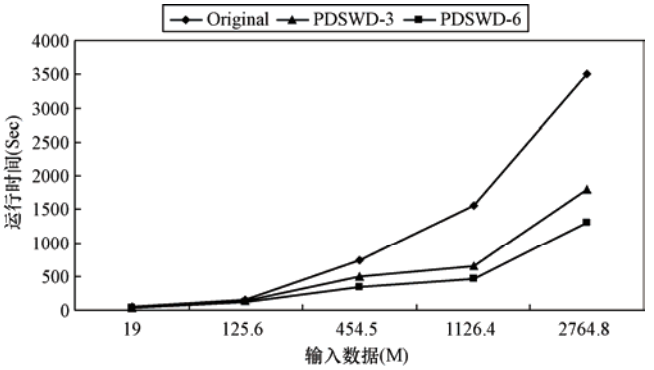

图 5 运行时间的对比

Fig. 5 Comparison of running time

3.2.3 可扩展性实验

通过改变集群规模，可观察不同节点数量对并行解压算法效率的影响。为此笔者在伪分布式集群和 3 台机器搭建的 Hadoop 集群上进行了实验，条件是只有机器数量的不同，其余配置均保持一致。实验分别解压缩了 19M（1 个文件）、125.6M（5 个文件）、454.5M（20 个文件）、1126.4M（60 个文件）的输入数据并记录解压时间。对于不同大小的数据输入，均测试 10 次运行时间并求出平均值，然后画出趋势图进行对比。在输入数据集的大小相同的情况下，可以利用加速比来衡量增加集群节点个数对并行解压算法效率的影响（陆秋等，2012），即：

$$S_1 = T_a / T_b \ (a = 1, b = 2, 3, \cdots) \tag{1}$$

式中，S_1 表示加速比；T_a 指的是伪分布式集群运行解压缩程序的执行时间；T_b 指的是多台机器搭建的 Hadoop 集群运行解压程序的执行时间，本文中 T_b 指的是 3 节点集群的运行时间。

图 6 展示了运行结果。当输入数据较小时，增加节点数量后并没有明显的变化；随着输入数据的不断增大，加速比的变化愈加明显。因此当输入数据量较大时，增加集群节点个数可以得到更好的效率。

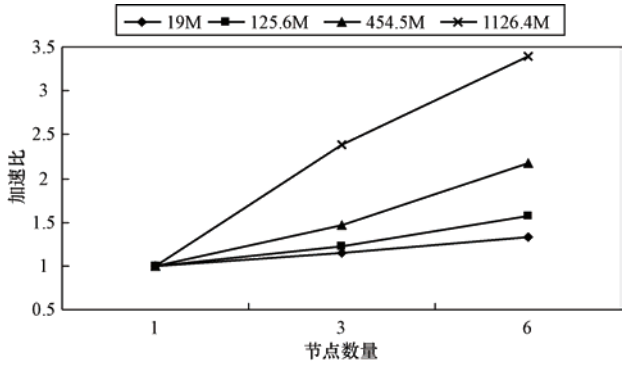

图 6 加速比

Fig. 6 Plot of speedup

4　结语

本文尝试将 MapReduce 编程模型引入到地震波形数据处理，提出了基于 MapReduce 的并行解压缩地震波形数据的算法 PDSWD，并给出了较详细的算法实现。与传统的串行解压缩算法相比，利用 MapReduce 并行解压缩处理数据具有更高的加速比，而且可以一次性解压缩批量文件。下一阶段笔者将尝试使用更大的集群规模和更大的测试数据集，并对 MapReduce 并行解压缩算法进行优化，扩展其功能，获得更好的工作效率。

参考文献

李闯，赵长海，晏海华，2010．基于 MapReduce 的菲涅耳带地震层析成像并行算法．见：2010 年全国高性能计算学术年会（HPC China）论文集，90—96．

陆秋，程小辉，2012．基于 MapReduce 的决策树算法并行化．计算机应用，**32**（9）：2463—2465，2469．

王洪体，陈阳，庄灿涛，2004．SEED 格式 STEIM2 数据压缩算法在实时地震数据传输中的应用．地震地磁观测与研究，**25**（4）：14—19．

文必龙，冯翔，左春雪等，2014．地震资料分布式存取的效率优化设计．计算机与数字工程，**42**（8）：1386—1389．

赵长海，晏海华，刘晓朋，熊登，史晓华，2012．以实际算法为例评估 MapReduce 在石油勘探中的应用．通信学报，（Z2）：81—89．

中国地震局，2003．地震波形数据交换格式（DB/T 2-2003）．北京：地震出版社．

Dean J., Ghemawat S., 2008. MapReduce: Simplified data processing on large clusters. Comunications of the ACM, **51** (1): 107—113.

Ghemawat S., Gobioff H., Leung S.T., 2003. The Google file system. ACM SIGOPS Operating Systems Review, **37** (5): 29—43.

Mauro M., Terje U., 2006. Mini SEED for LISS and data compression using Steim1 and Steim2. Norweigian National Seismic Network Technical Report.

White T., 2012. Hadoop: The definitive guide. CA Sebastopol: O'Reilly Media, Inc, 2012.

Research on Parallel Decompressing Algorithm for Seismic Waveform Data Based on MapReduce

Liu Fanming[1)], Guo Ruiqiang[1)], Li Yongqing[2)] and Bian Pengfei[2)]

1) College of Mathematics and Information Science, Hebei Normal University, Shijiazhuang 050024, China
2) Earthquake Administration of Hebei Province, Shijiazhuang 050021, China

Abstract In recent years, the number of SEED files was growing rapidly. In data processing, original algorithm of decompression batch seismic waveform data operated complicatedly and cost much time. In this paper, MapReduce programming model was introduced and a new parallel algorithm based on the thoughts of programming model and original decompression algorithm was presented. Also the design and implementation of this algorithm were given. Comparative experiments were carried out in terms of correctness, efficiency and extensibility. The results showed that the original algorithm spent more time compared to parallel algorithm which implementing decompression rapidly for a large number of seismic waveform data files. Using this method can decompress bulk of seismic waveform data and operate easily.

Key words: Seismic waveform data; Decompress; Parallel; MapReduce