文章编号：2095-2163(2022)10-0169-07 中图分类号：TP311.1,TP311.5 文献标志码：A

基于八叉树的地震数据分布式存储与计算

彭 成

(中国石油化工股份有限公司 石油勘探开发研究院，北京 100083)

摘 要：在参考谷歌文件系统分布式处理技术基础上,利用三维空间下八叉树结构与编码的快速空间定位机制,实现对三维地震数据的分块存储;采用中间文件形式进行子块切分与传输,减少本地时间开销;同时实现了基于地震道的一级缓存和基于子块的二级缓存,提升了数据访问效率。进一步基于分布式八叉树结构,设计实现了地震属性的分布式映射归并计算方法。研究方法为大数据背景下三维数据体的高效存储与处理分析提供了技术支持。

关键词：分布式；八叉树；地震数据；映射归并；多级缓存

Distributed storage and calculation of seismic data based on Octree

PENG Cheng

(Petroleum Exploration and Production Research Institute, SINOPEC, Beijing 100083, China)

【Abstract】 Based on the distributed processing technology of Google file system, the fast spatial positioning mechanism of Octree structure and coding in three-dimensional space is used to realize the block storage of three-dimensional seismic data; the sub block is segmented and transmitted in the form of intermediate file to reduce the local time overhead; at the same time, the primary cache based on seismic channel and the secondary cache based on sub block are realized to improve the efficiency of data access. Further, based on the distributed Octree structure, the distributed mapping and merging calculation method of seismic attributes is designed and implemented. The research method provides technical support for efficient storage, processing and analysis of 3D data volume under the background of big data.

【Key words】 distributed; Octree; seismic data; Map-Reduce; multilevel cache

0 引 言

随着地震采集及电子扫描技术的发展,产生了海量的地震数据。地震数据的特点是单一文件数据量大,经常达到 TB 级别,因此需要采用合适的存储技术来提升访问效率[1-3]。常用的技术、如分布式存储减少了本地的空间占用,通过将地震数据分块,一方面对局部区域的获取提升了效率,另一方面分块小数据更加灵活,可以在不同存储节点间进行迁移和备份等,相比于单一文件存储更加可靠。

在地震数据的获取和使用上,由于地震数据的文件数据量大,缓冲技术的使用必不可少[4]。地震数据常用的访问方式为按照线道方式的剖面获取,遍历整体地震数据体的属性计算和反演计算,以及任意方向三维切片显示等[5-6]。对于不同的数据获取需求,最适合的缓冲策略也不同,如何设计较为均衡的缓冲策略以支持多种不同的使用场景,是提升地震数据存取效率亟需解决的研究问题。

在高效的分布式存储和缓冲策略前提下,需要将其真正应用到地震反演等处理中,就还要配套的分布式计算框架[7]。目前主流的分布式并行处理框架包括谷歌分布式文件系统等,是比较易于扩展、并能应用到不同领域中的架构[8]。基于现有的并行编程模型,开发出适合地震数据文件特点以及符合地震计算流程需求的计算框架,才能充分利用分布式子块形式存储的地震数据。不同编程模型的取舍及实现的复杂度主要体现在—合适的并行粒度需要根据计算量、通信量、计算速度、通信速度进行综合平衡,同时设法加大计算时间相对于通信时间的比重,减少通信次数、甚至以计算换通信等。不管选用何种并行编程模型,均会涉及到任务划分、通信分析、任务组合及处理器映射等关键环节[9-10]。

本文在参考谷歌文件系统基础上,利用三维空间下八叉树结构与编码的快速空间定位机制,实现

基金项目：国家页岩油重大专项“陆相页岩油资源和选区评价技术与软件实现”(2017ZX05049-001-007)；中科院 A 类战略性先导科技专项“深层油气勘探定量评价软件平台”(XDAXX010405)。

作者简介：彭 成(1990-),男,硕士,高级工程师,主要研究方向:油气勘探专业软件研发。

收稿日期：2022-07-12

对三维大数据体的结构分块存储,同时设计了基于地震道的一级缓存和基于子块的二级缓存结构,提升了数据访问效率。进一步,设计实现了地震属性的分布式映射归并计算方法,为大数据背景下三维数据体的高效存储与处理分析提供了技术支持。

1 八叉树编码与分块存储

八叉树结构适用于对三维数据的分块,广泛应用于三维图像领域,即把三维数据立方体分割为 8 个子块,每个子块进一步切分为 8 个,直至达到合适的粒度。通过读取子块代替读取整个文件,可以提升图像的查询显示效率。

1.1 地震分块配置

对地震数据按照设定的分块大小进行八叉树切分,子块采用线性莫顿(Morton)编码。莫顿码按照大小排序得到子块自然数编码(Tile ID)。在利用八叉树子块来读取地震数据时,通过输入的主测线号,联络线号和深度范围得到其在源地震数据立方体中相应的空间范围,进而转换成一组对应的 Morton 编码,再转换为 Tile ID,定位到在文件中的存储位置。

地震子块文件命名采用随机 64 位哈希编码(uuid),每个子块文件命名为"XXX(uuid).afs"。子块有 Morton、Tile ID、uuid 三种码,与子块一一对应,从莫顿码和子块大小可以推导出对应的空间位置,从而实现编码和位置信息的关联。在分配存储节点时确定一个子块需要传输到哪些存储节点中,采用的是一致性哈希算法。

分布式存储的结构包括本地(local)、存储管理服务(master)、子块存储服务(chunk)三个类型的对象,其中 local 存放了待切分的源地震数据,master 中存放切分和 chunk 节点的参数配置,chunk 节点中存放子块及地震测网信息。master 及 chunk 节点基于远程调用框架 (Remote Call Framework,RCF),运行数据存取服务程序。子块平均分配传输到各个 chunk 节点中,实现分布式存储。

1.2 子块切分与传输

为提升切分生成子块的速度,减少读写文件对象切换及尽量顺序读写文件,本文中子块的传输通过中间文件进行周转。将源地震数据按照存储节点中逐个节点切割一份中间文件。中间文件数据全部属于对应的存储节点,存储节点的数据也全部来自于此中间文件。此后将中间文件传输到对应的存储节点中。如图 1 左半侧可以看到,本地工作包括源文件的顺序读取与中间文件的顺序写入。

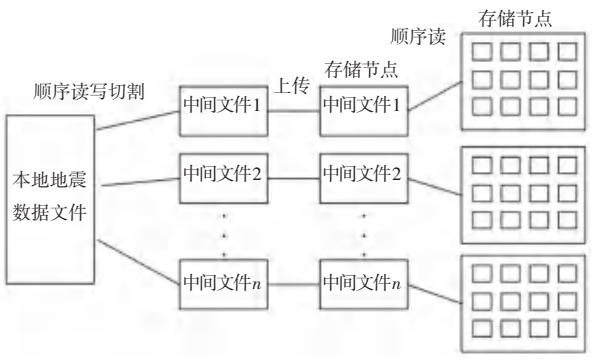

图 1 地震数据切块流程

Fig. 1 Seismic data segmentation process

每次读取源数据若干个地震道,默认读取数据量为 100 MB,并顺序遍历各个存储节点,将这批地震道数据中属于当前存储节点的数据写入其对应的中间文件。数据与存储节点的对应关系通过八叉树的切分方法来计算得到。在存储节点中,对接收到的中间文件,每次顺序读取 100 MB 左右的数据,将数据写入到各个子块中。如图 1 右半部分所示,遍历这 100 MB 左右的数据,每次读取最小单元的子块号和在子块中的位置,再将最小单元数据写入到对应的子块文件中。通过中间文件实现了源文件的顺序读、中间文件的顺序写,且减少了写对象切换的频率。读取地震数据与切分相反,也通过中间文件进行,实现源文件的顺序写、中间文件的顺序读。

2 多级缓存建立及利用

构成地震数据文件的基本单元是地震道,一级缓存就是在内存中存放一组地震道,每次查询请求传过来时,查找对应的地震道。通过命中次数计数,优先删除使用次数少的结果。

二级缓存是对子块的缓存,当一级缓存无法命中,就会继续从二级缓存去寻找。每个二级缓存相当于将一个子块文件加载到内存中,数据结构包括 Morton 码、Tile ID、命中次数等。

2.1 一级缓存的生成及利用

以地震道为单元的一级内存缓冲,缓存的是不同时窗范围、多次读取子块地震道数据合并后的地震道数据。一次具体的生成和利用如图 2 中左半部分所示:

(1)对于输入的主测线号、联络线号、时间范围,在地震道缓存中寻找相同主测线号、联络线号的缓存,如果找到,则判断缓存的时间范围是否能覆盖输入的时间范围;如果无法命中,继续查找二级缓存。

(2)当缓存道时间起止范围无法覆盖查询道的

时间起止范围时，缺少的部分从二级子块缓存中查找，而后将得到的数据与当前缓存中的地震道拼接；如果缓存时间范围可以覆盖用户查询请求，则直接使用此缓存地震道并增加缓存命中次数。

(3)将得到的地震道缓存按照用户查询请求的时间范围进行截断，生成地震数据并返回。

(4)当缓存道数量超过上限时，根据各个缓存道命中次数的多少，删除命中较少的缓存道，剩余所有缓存道的命中次数统一减去一个数值，数值大小为所删除的所有缓存道中命中次数最多的值。

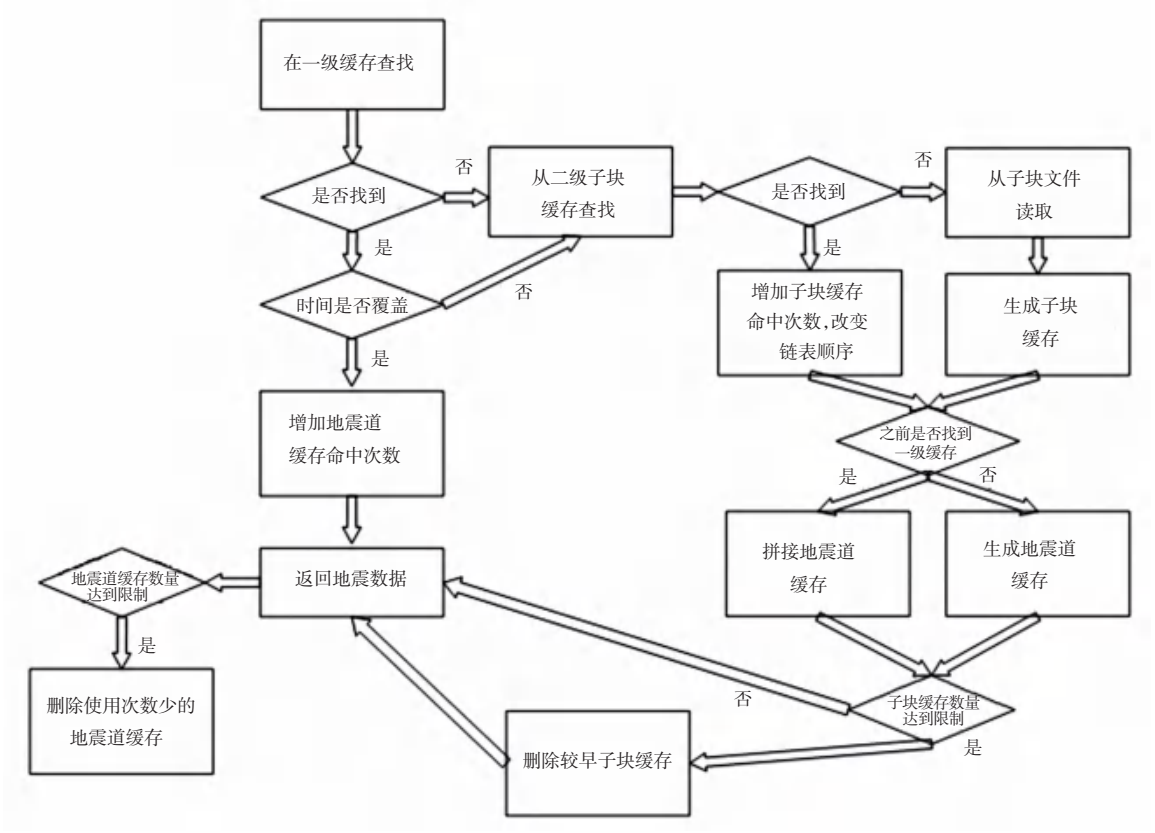

图 2　分级缓存读取地震道逻辑图

Fig. 2　Hierarchical cache reading seismis channel logic diagram

2.2　二级缓存的生成及利用

一级内存缓冲未能覆盖被请求的地震道数据时，根据主测线、联络线及时窗范围，转换为八叉树的莫顿码及对应的子块文件存储位置，询问以八叉树子块为存储单元的二级缓存。具体的流程见图 2 右半部分：

(1)如果找到二级缓存子块，命中计数加一；如果没有找到，则读取分布式子块数据并建立子块缓存。

(2)如果前期找到一级地震道缓存，可将子块数据拼接到地震道缓存中，如果没有找到则新建一个地震道缓存，时间范围为用户查询请求的范围，将子块数据填充到地震道缓存中。子块数据拼接和填充的流程方法为：根据子块数据所代表的时间范围，与地震道缓存的时间范围比较得到子块数据相对地震道缓存数据的具体位置，并替换已有位置上的数据或者填充到已有位置。子块缓存拼接、填充或替换地震道缓存如图 3 所示。由图 3 可知，对于缓存地震道时间起止范围内的情况，采用填充或者替换的方法；对于时间起止范围外的数据，采用拼接的方法。

(3)当子块缓存数量超过上限时，删去较早且使用次数较少的子块缓存，同时其他所有子块缓存的命中次数统一减去此批删去子块缓存的命中次数。

(4)将得到的地震道缓存按照用户查询请求的时间范围进行截断，生成地震数据并返回。

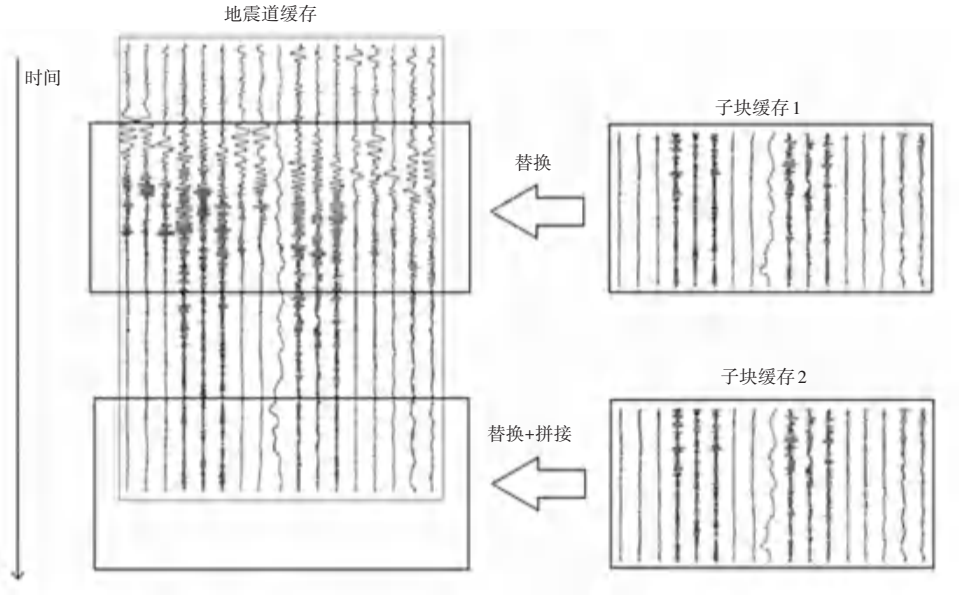

图 3 子块缓存拼接、填充或替换地震道缓存

Fig. 3 Subblock cache splicing, filling, or replacing the seismic channel cache

3 分布式映射归并计算

对批处理编程模型而言,任务划分与高效执行是建立在合理的数据粒度切分基础上。本文中的 Map-Reduce 是面向地震数据分析的服务,可执行常见的切片、地震属性分析及反演等算法,每一个映射服务(*mapper*)或归并服务(*reducer*)本身又是一个多线程执行框架。

3.1 部署映射归并服务

映射归并服务的结构包括任务管理服务(*Jobservant*)、映射服务(*mapper*)、归并服务(*reducer*),配置包括 *ip* 和端口、工作目录,服务之间的数据传输基于 *RCF* 的开源代码实现,实现心跳、消息传递、广播、事件服务等多种异步调用机制调度宕机的任务服务及发现新加入的任务服务,并合理安排计算过程。

地震数据属性计算模块参数配置包括计算的测网及深度范围,不同的地震属性计算有各自特有的计算参数,输出包括若干地震数据体或若干地震数据切片。在进行映射归并计算时,运行流程如图 4 所示。由图 4 可知,首先将用户配置好的属性计算参数传给 *mapper*、*reducer*,并从分布式文件数据块按照约定的计算数据体大小获取所有计算单元(*MapStart*),通过遍历分布式地震数据子块(*Map*),提取数据。然后遍历每个子块中的数据,以键值对(*key - value*)为单位进行计算。键值对是地震数据与对应空间范围对应关系的结构数据,对其进行属性计算后将结果写入 *mapper*、并传给 *reducer*。

所有 *mapper* 算得中间分析结果后,通知 *reducer* 进行归并,遵循键值对提取同一类别中间计算结果(*ReduceStart*),遍历并根据约定的算法归并最终数据块文件(*Reduce*),形成及建立分布式结果存储文件结果(*ReduceChunk*),得到最终的计算结果,再上传到 *chunk* 节点中,完成分布式计算。所有 *mapper* 和 *reducer* 内部计算基于多线程执行。

Map - Reduce 计算的关键步骤如图 5 所示。由图 5 可知,*mapper* 从不同的 *chunk* 获取待处理的粗粒度数据文件并读取键值对,执行匹配的地震属性计算函数,中间产生的结果存储到圆形内存缓冲区(先进先出类型),达到阈值后写入本地硬盘、且初步归并(*Combine*)形成面向不同 *reducer* 的子块文件。*Jobservant* 通知 *reducer* 所属的子块文件都在哪些 *mapper* 里并进行下载。*reducer* 获取到所有中间键值对后,就按照键值对的键进行归并排序形成中间临时文件。对于相同键的一批键值对,将其数据全部传给 Reduce 算法函数执行,将结果写入最终的输出文件。

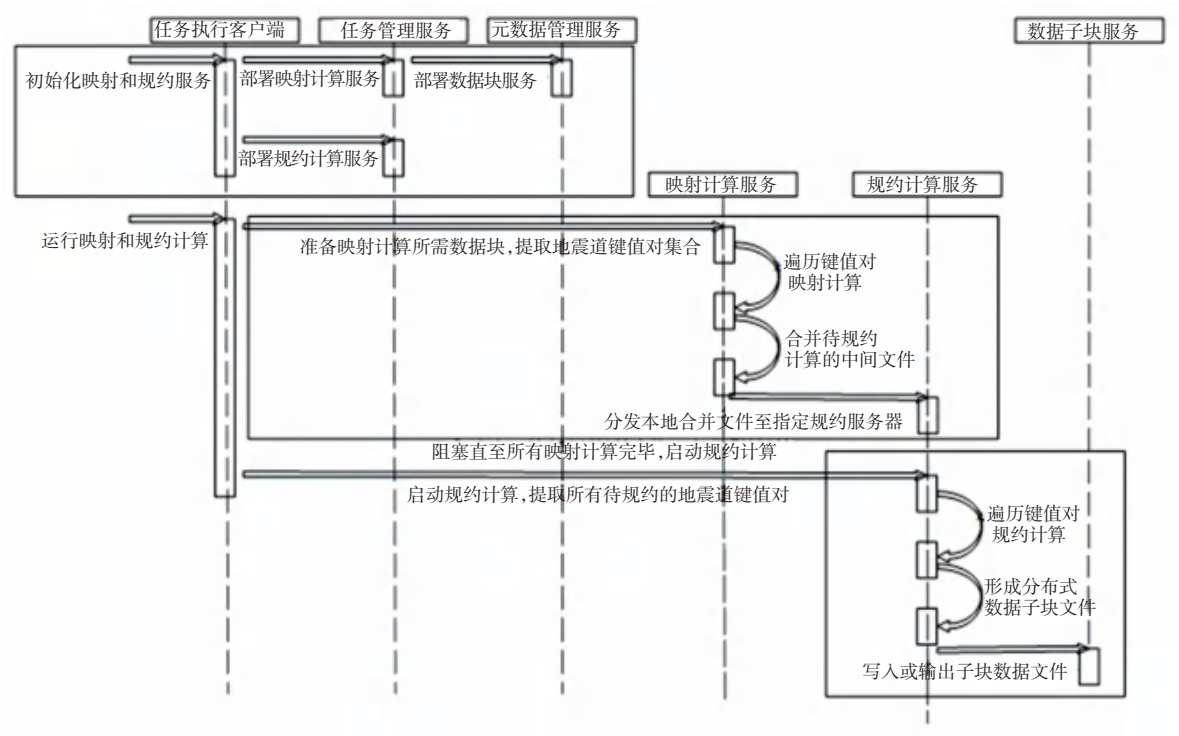

图 4　Map-Reduce 计算框架时序图

Fig. 4　Map-Reduce computing framework sequence diagram

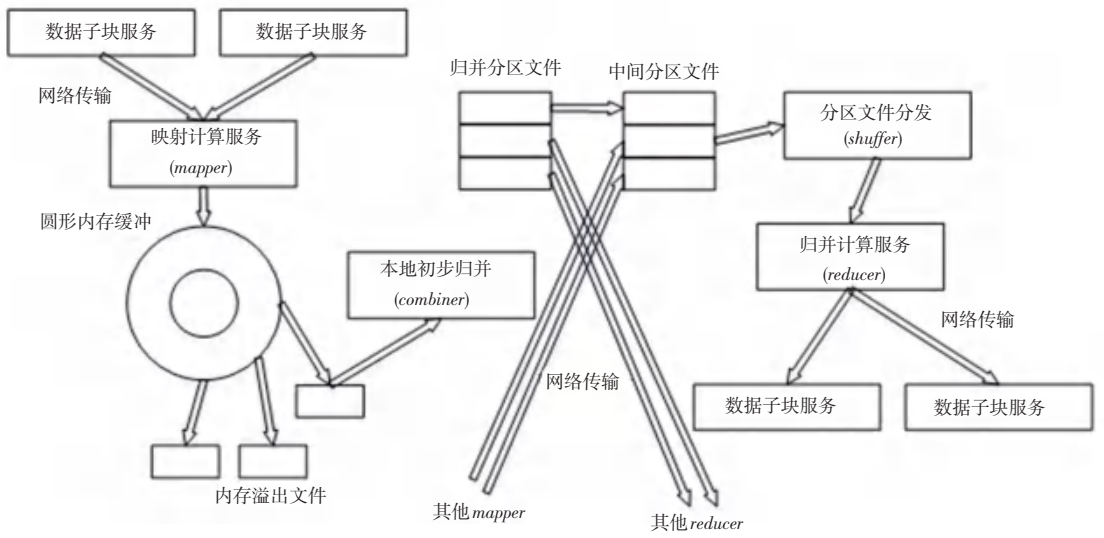

图 5　Map-Reduce 关键环节示意图

Fig. 5　Map-Reduce key link diagram

3.2　分布式映射计算

分布式计算流程如图 6 所示。由图 6 可知，首先通过 *Jobservant* 获取到 *mapper* 和 *reducer* 的地址并初始化与其连接的接口服务，接下来 *Jobservant* 会为其分配各自负责的子块文件。*mapper* 和 *reducer* 分配时，在平均分配子块的基础上，另需考虑避免超过剩余存储空间。各个 *mapper* 和 *reducer* 所负责的子块配置好后，本地将给其传送地震属性计算参数，*mapper* 开始进行计算，计算完成后再将计算结果发送过去。*reducer* 归并生成最终计算结果，此后传回本地及 *chunk* 节点，实现结果的分布式存储。

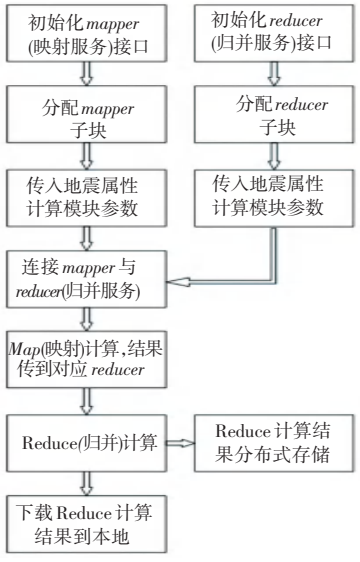

图 6 Map-Reduce 计算流程

Fig. 6 Map-Reduce calculation process

Map 的具体流程如图 7 所示,主要分为以下几个步骤:

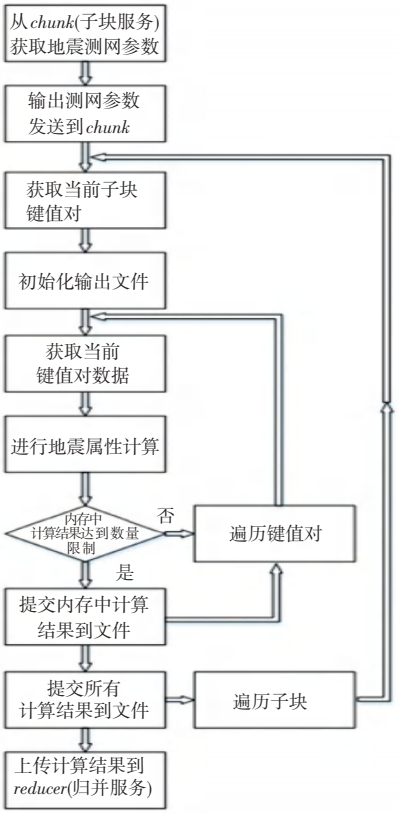

图 7 Map 计算流程

Fig. 7 Map calculation process

(1)首先从 *chunk* 节点下载源地震数据的索引文件获取测网参数,然后结合传送的地震属性计算参数,得到输出的测网范围,写成一个参数文件发送到 *chunk* 节点中,发送过去的路径即用户配置的分布式结果存放路径。

(2)通过 *master* 向对应 *chunk* 节点下载负责的子块文件,同时生成子块所属的键值对。遍历子块键值对进行属性计算,将输出写回键值对。

(3)将键值对写入输出文件,当一个子块计算完毕后,提交当前内存中的所有键值对到对应的结果文件中。

(4)发送结果文件到 *reducer* 节点,并向 *Jobservant* 获取各个 *reducer* 所负责的子块编号,再将这个子块传到上层督管负责的 *reducer* 中。

3.3 分布式归并计算

Reduce 的具体流程如图 8 所示,主要包括以下几个步骤:

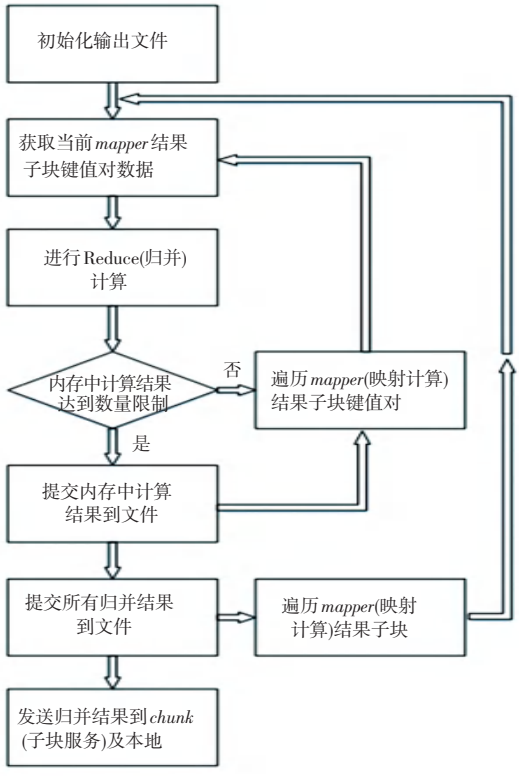

图 8 Reduce 计算流程

Fig. 8 Reduce calculation process

(1)初始化输出文件,分为 2 部分。一部分为单一的一个文件“reduceresult.afs”,存储所有计算结果;另一部分为按子块结构组织的一批文件,存放各个子块的计算结果,而后会上传到 *chunk* 节点,实现结果的分布式存储。

(2)遍历由其负责的所有子块,对每个子块遍历这些键值对,进行 Reduce 计算,形成八叉树子块。对涉及到面元计算的,需要跨键值对周围数据进行共同处理得到计算结果。

(3)将键值对写入输出文件,当一个子块计算

完毕后，提交当前内存中的所有键值对到相应的结果文件中。

(4)将这些结果文件发送到 *chunk* 节点中，并向 *master* 查获各个 *chunk* 存储的子块编号，同时将子块传输到上层督管负责的 *chunk* 中；本地下载“reduceresult.afs”读取其中的键值对，按照键值对的位置信息写入到新建的输出数据体或切片中，完成分布式结果的下载。

4　结束语

本文实现了一种基于八叉树的地震数据多级缓存方法，采用基于地震道的一级缓存和基于子块的二级缓存，分别提升了查询请求响应速度和子块读取响应速度。

采用缓存访问频次记录，避免数据块在内存中的频繁迁移，并且可以优先剔除出利用次数少的缓存对象。

基于分布式八叉树实现了地震数据分布式属性计算及结果分布式存储，减少单机计算工作量，提升了计算效率。

参考文献

[1] 赵辉. 地震监测数据的 Hadoop 存储解决方案[J]. 华南地震，2020，40(03)：70-75.

[2] 陈柯宇，孙韵，张恩莉，等. 石油勘探海量地震数据存储管理研究[J]. 电子技术与软件工程，2020(13)：148-149.

[3] 龚明明，叶伦强. 地震信息网络数据的动态存储方法研究[J]. 地震工程学报，2020，42(04)：1043-1048.

[4] 陈晓琳，李盛乐，刘坚，等. 分布式数据库 Greenplum 在地震前兆数据存储中的应用[J]. 地震研究，2020，43(02)：412-416，418.

[5] 张秀萍，李君，袁林，等. 基于 Datist 的地震目录资料自动下载与数据存储[J]. 防灾减灾学报，2020，36(01)：87-91.

[6] 朱飞鸿，柴旭超，王文青，等. 模拟地震图纸数字化存储信息录入的设计与实现[J]. 地震学报，2020，42(01)：101-108，121.

[7] 单维锋，滕云田，刘海军，等. 大数据环境下地震观测数据存储方案研究[J]. 中国地震，2019，35(03)：558-564.

[8] 唐伟，刘鲁宁，王丽英. 物探地震光磁介质地质资料管理实务探析[J]. 山东档案，2019(04)：81-82.

[9] 于盛. 地震资料处理中的并行存储技术研究与应用[J]. 电脑知识与技术，2019，15(21)：266-267.

[10] 李正，黎斌，董非非，等. 地震观测数据分布式存储的研究与应用[J]. 科技视界，2019(20)：107-108.

(上接第 168 页)

由表 3 可知，相对其他模型，引入注意力机制的双层 LSTM 模型的预测准确率最高，为 71.8%。

4　结束语

针对比赛实时预测问题，本文采用引入注意力机制的双层 LSTM 模型模型，该模型以比赛中的经验、金币及阵容信息作为模型输入，预测比赛的最终结果，在比赛进行到第 15 min 时，预测准确率为 71.8%。较传统方法相比，预测准确率提高了 1.5% 以上，得到了较优的效果。在 MOBA 类游戏中，选手的水平同样影响着比赛的胜负，而本文暂未考虑选手的因素，在未来的研究中，可以弥补这一不足，从而进一步提升模型预测结果的准确性。

参考文献

[1] KINKADE N, JOLLA L, LIM K. Dota 2 win prediction[R]. Stanford, CA, USA: Stanford University, 2015.

[2] WANG Nanzhi, LI Lin, XIAO Linlong, et al. Outcome prediction of dota2 using machine learning methods [C]//International Conference on Mathematics and Artificial Intelligence. Chengdu, China: ACM, 2018: 61-67.

[3] WANG Kaixiang, SHANG Wenqian. Outcome prediction of DOTA2 based on Naïve Bayes classifier[C]// 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). Wuhan, China: IEEE, 2017: 591-593.

[4] SONG Kuangyan, ZHANG Tianyi, MA Chao. Predicting the winning side of DotA2 [R]. Stanford, CA, USA: Stanford University, 2015.

[5] LIN L. League of legends match outcome prediction [R]. Stanford, CA, USA: Stanford University, 2016.

[6] CLEGHERN Z, LAHIRI S, ÖZALTIN O, et al. Predicting future states in dota 2 using value-split models of time series attribute data [C]//Conference on the Foundations of Digital Games. Hyannis, MA, USA: ACM, 2017: 1-10.

[7] SILVA A L C, PAPPA G L, CHAIMOWICZ L. Continuous outcome prediction of league of legends competitive matches using recurrent neural networks[C]//SBC-Proceedings of SBC Games. [S.l.]: IEEE, 2018: 2179-2259.

[8] GRUTZIK P, HIGGINS J, TRAN L. Predicting outcomes of professional DotA 2 matches[R]. Stanford, California: Stanford University, 2017.

[9] 许晨波. 基于深度学习的 Dota2 阵容推荐与胜率预测系统的研究与实现[D]. 开封：河南大学，2019.

[10] YANG Yifan, QIN Tian, LEI Yuheng. Real-time esports match result prediction[J]. arXiv preprint arXiv: 1701.03162, 2016.

[11] HODGE V, DEVLIN S, SEPHTON N, et al. Win prediction in multi-player esports: Live professional match prediction [J]. IEEE Transactions on Games , 2021, 13(4): 368-379.

[12] YANG Zelong, PAN Zhufeng, WANG Yan, et al. Interpretable real-time win prediction for honor of Kings-A popular mobile MOBA esport[J]. arXiv preprint arXiv: 2008.06313, 2020.

[13] 李康维，田佳，曹啸博，等. 基于序列到序列结构的 MOBA 游戏局势趋势预测模型[J/OL]. 控制与决策：1-7[2014-08-01]. https://doi.org/10.13195/j.kzyjc.2021.0903.

[14] 开源的 Dota2 数据平台. OpenDOTA[EB/OL]. [2014-08-01]. https://www.opendota.com.