计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(9):319−323 [doi: 10.15888/j.cnki.csa.008683]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向决策分析的海量气象数值预报数据快速提取①

李永生, 张金标, 张　敏, 陈冰怀

(广东省气象探测数据中心, 广州 510080)
通信作者: 李永生, E-mail: 879976993@qq.com

摘　要: 以海量半结构化的气象数值预报数据产品为研究对象, 针对传统数据抽取方法效率不高的问题, 基于多进程处理技术, 设计了一种基于精准位置寻址的快速数据块定位算法, 实现了数据块的精准定位; 设计了可按需在空间范围内进行裁剪的截取算法, 可按需根据数据的属性维度、经纬度范围等信息实现数据按需抽取; 基于上述算法实现了全流程统一控制的多进程数据读取的业务流程. 并以单平面耗时为主要考核指标, 分别采用 1 进程, 4 进程、8 进程以及 16 进程进行数据处理, 实际测试结果表明, 采用 16 进程处理比单个进程处理的速度由 257 ms 提高到 37 ms. 该方法有效的提升非结构气象数值预报产品数据的抽取效率, 已在面向城市治理等气象决策分析业务中业务化应用.

关键词: 非结构化模型; 数据抽取; 数据分析; 决策分析; 可视化分析

引用格式: 李永生,张金标,张敏,陈冰怀.面向决策分析的海量气象数值预报数据快速提取.计算机系统应用,2022,31(9):319–323. http://www.c-s-a.org.cn/1003-3254/8683.html

Fast Data Extraction for Numerical Weather Prediction Based on Decision Analysis

LI Yong-Sheng, ZHANG Jin-Biao, ZHANG Min, CHEN Bing-Huai

(Guangdong Meteorological Observation Data Center, Guangzhou 510080, China)

Abstract: Traditional data extraction methods are usually inefficient. To address this problem, we first design an exact position addressing-based algorithm with multi-processing methods to achieve the accurate positioning of data blocks by taking the massive data generated from semi-structured numerical weather prediction (NWP) products as the research object. Then, an extraction algorithm is designed to extract data in the spatial range on demand, namely, to realize on-demand data extraction according to attribute dimensions as well as the latitude and longitude of data. As a result, the multi-process data reading under unified whole-process control is achieved on the basis of the above two algorithms. For testing, the time consumption of a single data plane is taken as the main assessment index, and the single-, quad-, octo-, and 16-core processes are employed for data processing. The test results reveal that the processing with 16-core processes is faster than that of a single-core process, and the time consumption is reduced from 257 ms to 37 ms. This method can effectively improve the efficiency of data extraction for non-structural NWP products and has been put into use in decision analysis for urban governance.

Key words: unstructured model; data extraction; data analysis; decision analysis; visual analysis

1　引言

GRIB 码是世界气象组织 (WMO) 推荐使用的与计算机操作系统无关的压缩的二进制编码, 是一种用于交换和存储规则分布数据的二进制文件格式, 现行

① 基金项目: 广东省科技计划 (2018B020207012)
收稿时间: 2021-12-14; 修改时间: 2022-01-12; 采用时间: 2022-01-26; csa 在线出版时间: 2022-06-17

的 GRIB 码版本有 GRIB1 和 GRIB2 两种格式, 具有多维数据存储方便、压缩比高、支持多种压缩方式、扩展性强等特点, 尤其擅长大容量多维度的非结构化数据的交换, 广泛应用于数值天气预报产品、海洋水文等数据的存储和交换. 针对 GRIB 码进行解码工具有两种, 第一种是 wgrib 命令行工具, 可以通过命令行的方式通过不同的参数组合对 GRIB 数据进行读取操作. 第 2 种方法是利用业界 Unidata 发布的 NetCDF-java 的网格数据操作 API (支持 Bufr、Grib、NetCDF3/4、HDF 等网格数据), 具有按照 Shape 形状提取指定范围内网格数据的方法, 可以通过编程使用 get 和 set 键值操作读取数据, 其中第一种方法在国内应用比较广泛. 使用上述方法基本能够满足获取特定维度的部分数据的业务应用需求[1], 但在面向决策分析领域气象预测的应用中, 通常以融合分析应用为主, 需要对海量的数值预报产品的一次性批量数据快速读取和分析, 传统的数据读物方法存在数据读取时间效率不高, 空间资源消耗大等问题[2]. 本文即针对上述业务痛点, 面向气象预警、预报和预测业务实际需求, 以 GRIB 码存储的数值预报数据产品为研究对象, 设计一种基于精准定位, 数据提取的维度、读取经纬度的范围可按需定制的, 多进程处理的快速数据提取方法.

2 研究对象数学模型

本文的研究对象是以 GRIB 格式存储的半结构化气象数值预报产品, 现行的 GRIB 码版本有 GRIB1 和 GRIB2 两种格式, 每个 GRIB1 记录单元通常存储某个要素在某个层次的连续网格序列点的数据值. 逻辑上将记录单元划分为“段”, 每个 GRIB1 记录单元可以包含六段, 其中两个段是可选的, 具体如表 1 所示.

表 1 GRIB1 格式说明

段号	段名	内容
0段	指示段	GRIB资料长度, GRIB版本号
1段	产品定义段	段长, 编码的分析或预报产品的标识
2段	网格定义段 (可选)	段长, 网格的几何形状
3段	位图段 (可选)	段长, 一个比特位对应一个网格上的数值
4段	二进制数据段	段长, 数据值
5段	结束段	“7777”表示结束

GRIB 文件每一个“段”对应到数值预报产品中就是一个平面数据, 该平面数据是一个非结构化的网格数据, 网格点数由产品的空间分辨率决定, 平面数据 $Data[v]$由产品的要素种类 ($VarName$)、产品的生成时间 ($DataTime$)、层次高度 ($LevelName$)、预报时效 ($ForcastName$)、成员变量 ($MemberName$) 等维度信息唯一确定, 用数学表达为:

$$Data[v] = F(VarName, DataTime, LevelName, ForcastName, \cdots)$$

上述维度信息确定的平面数据$Data[v]$是网格点类的半结构化数据, 其属性信息包括经度范围、纬度范围、空间分辨率以及数据维度上的南北走向, 1 代表维度上从南到北依次排列, 0 表示维度从北到南依次排列, 其属性函数可以描述为:

$$Data[v] = M(LonNum, LatNum, SpaRes, DataDir, \cdots)$$

3 数据快速精准定位算法的设计与实现

分析传统数据读取算法的技术原理可知, 数据解码效率方面存在主要制约因素[3] 是需要一定范围内遍历查找所需提取特定维度数据所在的位置, 即数据块的定位[4]; 这在一次性批量处理大量数值预报产品时产生了一定的时间损耗; 但上述因素存在较大的提升空间, 相关的方法的创新研究也基本围绕上述制约因素展开. 针对上述问题, 本文基于精准数据定位获取数据的算法[5], 节省了传统方法中数据块查找所消耗的时间; 进而明显提高了数据的读取速度, 算法具体流程如下.

每个 GRIB 实体文件是若干平面数据$Data[v]$的组合, 每个平面数据对应一个数据块, 文件结构示意如图 1 所示.

(1) 获取元信息体: 获取整个 GRIB 文件的所有数据块的元信息, 包括每个“数据块”的起始位置, 结束位置等, 所获取的信息均是轻量的元信息, 形成元信息体. 整个过程的耗时在毫秒级别.

(2) 获取实体信息体: 算法过滤去掉每个数据块的元数据信息, 并一次性将所有 GRIB 数据解码为二进制数据, 包括数据的解压缩、解码, 一次性将所有 GRIB 数据解码为二进制数据, 称为实体数据体. 实体数据体存储的顺序和步骤 (1) 中元数据的顺序是一一对应的.

(3) 目标数据块的位置: 根据元数据体和实体数据体的对应关系, 首先对元数据体进行整理, 通过数据整理函数, 将元数据信息进行分类, 重点确定数据块的实际位置, 位置确定后, 根据属性信息计算数据的偏移量进而定位数据并对数据块进行循环读取.

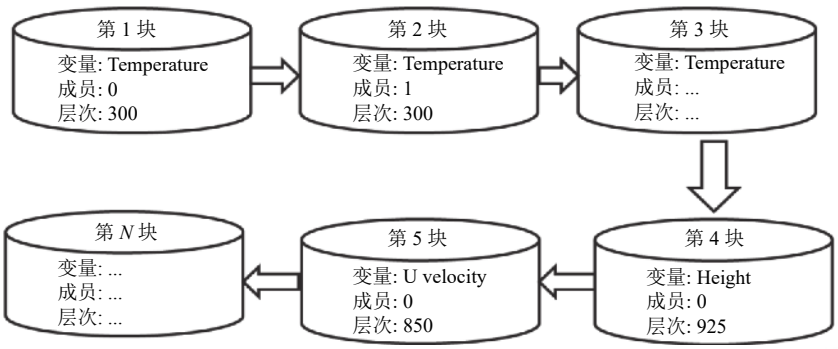

图 1 数据文件结构图

4 数据按需定制截取算法的设计与实现

获取特定数据块的精确位置后, 为了不产生计算和存储资源的浪费以及处理时间的增加. 需要对一定空间范围的数据进行按需截取, 避免总是进行整个平面数据的读取和流转处理, 从而可以按照业务需求, 根据其属性函数在空间范围内进行裁剪以获取业务所需空间范围的数据, 这样可以减少数据的传输量, 提高数据业务应用效率[6–9].

4.1 按需可定制的内容

(1) 可对平面数据进行空间范围内的按需裁剪, 允许业务用户根据所需数据的空间经纬度信息对数据块进行按需截取[10].

(2) 可根据数值预报产品的要素种类、产品的生成时间、层次高度、预报时效等信息, 定制筛选业务所需的平面数据[11,12].

4.2 空间范围截取算法

算法根据实际业务确定的空间经纬度范围信息, 对数值预报产品平面数据进行逐行剪裁, 形成业务所需空间范围的平面数据, 数据截取从低纬度到高维的方向具体的算法流程如下.

(1) 每圈读取长度的确定

根据数值预报产品原始空间经度范围信息确定, 整个经度范围:

$$R1 = Math.abs(LongStart - LongEnd) + LongInterval$$

整体经度差:

$$L1 = Math.abs(origianlLongStart - cutLongStart) + Math.abs(originalLongEnd - cutLongEnd)$$

计算单圈读取的数据长度:

$$Length = 4\times(R1 - L1)/LongInterval$$

(2) 再确定读取圈数:

整个纬度范围:

$$R2 = Math.abs(LatiStart - LatiEnd) + LatiInterval$$

整体纬度差:

$$L2 = Math.abs(origianlLatiStart - cutLatiStart) + Math.abs(originalLatiEnd - cutLatiEnd)$$

计算读取圈数:

$$Number = 4\times(R2 - L2)/LatiInterval$$

采用 for 循环依次读取每圈数据:

$$f(d) = \sum_{n=1}^{Number} Length\times n$$

算法的实现过程如算法 1.

算法 1. 格点数据按需截取算法

输入: 模型 *model* 包括分辨率值, 原始经纬度范围, 裁剪经纬度范围: {[(*longitudeInterval*, *latitudeInterval*)], [(*orignalLongitudeStart*, *orignalLatitudeStart*), (*orignalLongitudeEnd*, *orignalLatitudeEnd*)], [(*cutLongitudeStart*, *orignalLatitudeStart*), (*orignalLongitudeEnd*, *orignalLatitudeEnd*)]}

输出: 字节数组 *data*

1. 计算经纬度范围
2. double *longitudeRange*=|*orignalLongitudeStart*−*orignalLongitudeEnd*|+*longitudeInterval*
3. double *latitudeRange*=|*orignalLatitudeStart*−*orignalLatitudeEnd*|+*latitudeInterval*
4. 计算经度差值
double *longitudeSub*1=|*orignalLongitudeStart*−*cutLongitudeStart*|
double *longitudeSub*2=|*orignalLongitudeEnd*−*cutLongitudeEnd*|
5. 计算纬度差值
double *latitudeSub*1=|*orignalLatitudeStart*−*cutLatitudeStart*|
double *latitudeSub*2=|*orignalLatitudeEnd*−*cutLatitudeEnd*|
6. 计算整体的经度差值和纬度差值
double *longitudeFinal*=*longitudeSub*1+*longitudeSub*2
double *latitudeFinal*=*latitudeSub*1+*latitudeSub*2
7. 确定数据存储方向, 1 表示从南到北, 0 表示从北到南
String *sNDirection* = *model*.*getGridInfoXml*().*getsNDirection*();
String *wEDirection* = *model*.*getGridInfoXml*().*getwEDirection*();

```
8. 根据缠绕方向确定纬度裁剪大小
  double latitudeSub = 0d;
  switch (sNDirection) do
      case “1”:
          latitudeSub = latitudeSub2;
          break;
        case “0”:
          latitudeSub = latitudeSub1;
          break;
        default:
          decodeLogger.error (“---------数据南北缠绕方向定义错误, 只能为 1 或者 0!”);
          break;
      }
  end switch
9. 根据缠绕方向确定经度裁剪大小
  double longitudeSub = 0d;
  switch (wEDirection) do
      case “1”:
        longitudeSub = longitudeSub1;
        break;
    case “0”:
        longitudeSub = longitudeSub2;
        break;
    default:
        decodeLogger.error(“---------数据东西缠绕方向定义错误, 只能为 1 或者 0!”);
        break;
    }
    End switch
10. 计算纬度遍历个数
  int latiLengthValue = (int)((latitudeRange－latitudeFinal)/latitudeInterval);
11. 计算纬线长度
  double latiLength = latitudeSub/latitudeInterval;
12. 计算纬度取值长度
  double longitudePosi = 4 × (longitudeRange/longitudeInterval) × latiLength;
13. 计算沿经度纬度移动值
  int latitudePosi = 4 × (int)(longitudeSub/longitudeInterval);
14. 计算开始整体移动的个数
  long finalPosi = (long)(planeStartPos + longitudePosi + latitudePosi);
15. 计算单圈读取长度
  double length = 4 × (longitudeRange－longitudeFinal)/longitudeInterval;
16. 输出 data
  for int i = 0; i < latiLengthValue; i++ do
  byte[] oneCircleByte = gribUtil.readFileByPosiBytes(bodyPath, finalPosi, (int) length);
      data = byteJoin(data, oneCircleByte);
      finalPosi += 4 × longitudeRange/longitudeInterval;
    }
  end for
17. return data
```

5 实例测试与结果分析

本文选取一类数值预报产品进行实际测试, 该产品每时次需处理的数据文件为 66 个, 每个文件数据量约 600 M, 每个文件约 200 个平面数据. 以数据解码耗时为主要考核指标, 分别对不同的文件数均采用单个进程解码测试. 结果表明随着文件数量的增长耗时基本呈线性增长趋势, 如图 2 所示, 且 66 个文件的总耗时由传统方法的 2 h 提高到约 40 min, 解码效率大幅提升.

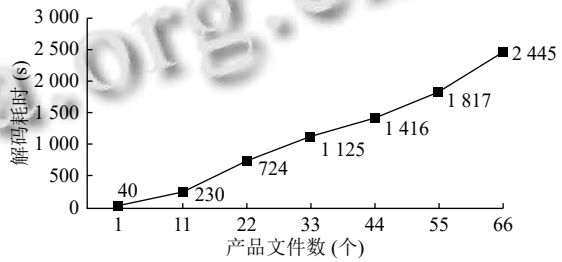

图 2 单进程多文件解码耗时统计图

另外以单平面耗时为主要考核指标, 分别采用 1 进程, 4 进程、8 进程以及 16 进程进行数据处理, 实际测试结果表明, 采用 16 进程处理的速度由单个进程的 257 ms 提高到 37 ms. 实际测试结果显示多进程技术的引入对数据处理速度提升明显. 如图 3 所示.

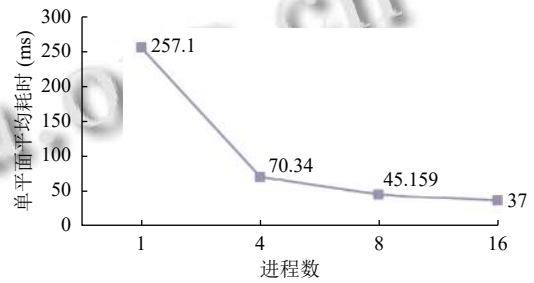

图 3 多进程单平面解码耗时统计图

本文实现的批量数据数据处理方法已经为广东省气象行业的市县版的 Gift 系统提供数据快速处理服务, 同时在广东省气象行业的预报预测和决策分析业务系统提供可视化数据服务支撑, 如图 4 所示.

6 结论

本文针对传统数据抽取方法效率不高的问题, 基于多进程处理技术, 设计了一种基于精准位置寻址的快速数据块定位算法, 实现了数据块的精准定位; 设计

了可按需在空间范围内进行裁剪的截取算法, 可按需根据数据的属性维度、经纬度范围等信息实现数据按需抽取; 基于上述算法实现了全流程统一控制的多进程数据读取的业务流程. 实际测试结果表明利用本方法来解码 GRIB 格式的数值预报产品, 可以大大提升非结构气象数值预报产品数据的抽取效率, 提高资源利用率. 为海量半结构化的气象数值预报数据产品的快速处理提供了方法参考, 具有很好的业务应用价值.

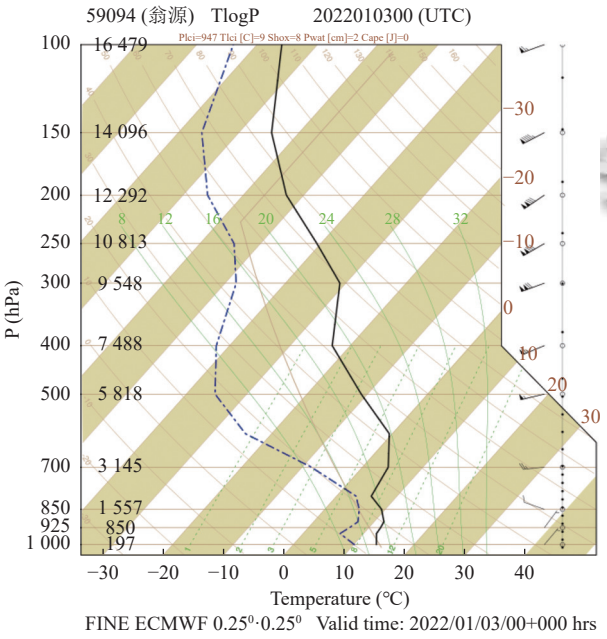

图 4 预报决策可视化业务化数据支撑图

参考文献

1 李永生, 曾沁, 杨玉红, 等. 基于大数据技术的气象算法并行化研究. 计算机技术与发展, 2016, 26(9): 47–49, 55.

2 李鸣野. 基于散列查找和多线程调度的快速提取 GRIB 数据方法. 山西师范大学学报 (自然科学版), 2019, 33(2): 10–17.

3 曾沁, 李永生. 基于分布式计算框架的风暴三维追踪方法. 计算机应用, 2017, 37(4): 941–944. [doi: 10.11772/j.issn.1001-9081.2017.04.0941]

4 李永生, 曾沁, 徐美红, 等. 基于 Hadoop 的数值预报产品服务平台设计与实现. 应用气象学报, 2015, 26(1): 122–128. [doi: 10.11898/1001-7313.20150113]

5 但玻, 冯汉中, 罗可生. ECMWF0.25*0.25 经纬网格模式资料处理及软件实现. 高原山地气象研究, 2013, 33(3): 92–96.

6 刘媛媛, 应显勋, 赵芳. GRIB2 介绍及解码初探. 气象科技, 2006, 34(S1): 61–64.

7 赵芳, 薛蕾, 刘媛媛. 表格驱动码业务试验系统设计与实现. 气象科技, 2018, 46(4): 679–684.

8 肖华东, 孙婧, 孙朝阳, 等. 中国气象局 S2S 数据归档中心设计及关键技术. 应用气象学报, 2017, 28(5): 632–640. [doi: 10.11898/1001-7313.20170511]

9 孙周军, 乔文文, 侯灵, 等. 混合架构的可视化数据调度检索模型. 计算机系统应用, 2019, 28(12): 63–71. [doi: 10.15888/j.cnki.csa.007202]

10 王兵, 李杰. 基于通用模型的 GRIB 格式数据读取技术. 航空计算技术, 2018, 48(6): 96–101. [doi: 10.3969/j.issn.1671-654X.2018.06.023]

11 张瑞聪, 任鹏程, 房凯, 等. Hadoop 环境下分布式物联网设备状态分析处理系统. 计算机系统应用, 2019, 28(12): 79–85. [doi: 10.15888/j.cnki.csa.007181]

12 胡洋. 基于深度学习的 SDN 虚拟蜜网路由优化. 计算机系统应用, 2020, 29(10): 274–279. [doi: 10.15888/j.cnki.csa.007626]

(校对责编: 孙君艳)