Computer Science and Application 计算机科学与应用, 2025, 15(8), 34-40
Published Online August 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.158195

Hans汉斯

一种基于文件内存映射的海量数据快速存取方法研究

刘　茵，解　菁，李　荣，蔚文婧，万春旭*

北京农业职业学院智慧农业工程学院，北京

收稿日期：2025年7月1日；录用日期：2025年7月30日；发布日期：2025年8月6日

摘　要

随着大数据技术的迅猛发展，海量数据的高效存取已成为网络开发领域亟待解决的关键问题。传统的数据存取方式主要包括直接文件读写和数据库存取，二者在性能表现上各有优劣。直接文件读写方式以其极高的写入速度和简便性著称，但在数据检索方面存在复杂度高、效率低下的问题；而数据库系统虽然能够提供快速且高效的数据检索能力，但其创建和写入过程相对复杂，写入速度较直接文件写入方式明显滞后。本文提出了一种基于文件内存映射的海量数据存取方法。该方法通过将文件内容映射到内存空间，充分利用内存的高速读写特性，显著提升数据检索效率。同时，结合直接文件读写的优势，确保海量数据的存取过程在保持高效写入速度的同时，实现数据检索效率的大幅提升，从而在数据保存与读取效率之间达到良好的平衡。实验结果表明，该方法在检索速度和写入速度上均优于传统方法，具有较高的实用性和应用前景。

关键词

大数据，海量数据存取，数据库，文件内存映射，多线程

Research on a Fast Access Method for Massive Data Based on File Memory Mapping

Yin Liu, Jing Xie, Rong Li, Wenjing Wei, Chunxu Wan*

School of Smart Agriculture Engineering, Beijing Vocational College of Agriculture, Beijing

Received: Jul. 1st, 2025; accepted: Jul. 30th, 2025; published: Aug. 6th, 2025

*通讯作者。

文章引用: 刘茵, 解菁, 李荣, 蔚文婧, 万春旭. 一种基于文件内存映射的海量数据快速存取方法研究[J]. 计算机科学与应用, 2025, 15(8): 34-40. DOI: 10.12677/csa.2025.158195

Abstract

With the rapid development of big data technology, efficient access to massive data has become a key issue that urgently needs to be addressed in the field of network development. The traditional data access methods mainly include direct file read and write and database access, both of which have their own advantages and disadvantages in performance. The direct file reading and writing method is known for its extremely high writing speed and simplicity, but there are problems of high complexity and low efficiency in data retrieval; Although database systems can provide fast and efficient data retrieval capabilities, their creation and writing processes are relatively complex, and their writing speed lags significantly behind direct file writing methods. This article proposes a massive data access method based on file memory mapping. This method maps file content to memory space, fully utilizing the high-speed read and write characteristics of memory, significantly improving data retrieval efficiency. At the same time, by combining the advantages of direct file read and write, it ensures that the access process of massive data maintains efficient write speed while significantly improving data retrieval efficiency, thereby achieving a good balance between data storage and read efficiency. The experimental results show that this method is superior to traditional methods in terms of retrieval speed and writing speed, and has high practicality and application prospects.

Keywords

Big Data, Massive Data Access, Database, File Memory Mapping, Multi-Threading

1. 引言

当今社会已步入大数据时代，各领域数据量激增。全球每日产生数据量高达数万亿字节，这对数据的存储与读取带来巨大挑战。传统数据处理读写部分主要采用文件 I/O 读写和数据库读写这两种方法。文件 I/O 读写优点是写入速度快。处理包含 100 万条简单记录的数据集，在普通 PC 上仅需数秒完成。这是因为操作只需按格式将数据顺序写入磁盘连续空间，无需构建复杂数据结构或索引。缺点是检索效率低。文件本身缺乏索引机制，查找特定记录需从文件头逐行读取比对，对海量数据而言效率极低，一次检索可能耗时数分钟甚至更长。数据库读写优点是检索效率高。利用精心设计的数据结构和索引算法，数据库能快速定位数据。在同样百万条记录的表中，一次精确查询通常仅需几毫秒即可返回结果。缺点是写入速度慢且创建复杂。创建复杂数据库需前期规划、配置(定义架构、表结构、设置索引等)，过程繁琐耗时。写入时为保证一致性、完整性及事务处理需执行复杂操作，导致速度远低于文件写入。插入百万条记录可能耗时数十秒至数分钟，具体取决于配置和硬件。

为解决海量数据存储(写入)与读取(检索)效率的矛盾，满足大数据应用对快速存取的迫切需求，本文提出了一种基于文件内存映射的数据存取方法。该方法在继承文件 I/O 读写在写入速度上的高效性同时。通过引入文件内存映射技术及优化数据存储格式，显著提升数据检索效率，在数据保存和读取效率之间实现良好平衡，为大数据处理提供一种创新实用的解决方案。

2. 文件内存映射

文件内存映射是一种强大的操作系统机制，它允许程序将文件的内容直接映射到进程的虚拟地址空

间中。通过这种映射，进程对文件的访问就如同对内存的访问一样，可以直接通过指针操作来读取和修改文件数据，而无需使用传统的文件 I/O 系统调用(如 read 和 write 函数) [1] [2]。

文件内存映射具有诸多优点。首先，它能极大地提高 I/O 效率。传统的文件 I/O 操作需要在用户空间和内核空间之间进行多次数据拷贝，而内存映射文件机制使得进程可以直接操作内存，避免了这些额外的数据拷贝开销。例如，在读取一个 1 GB 的大文件时，使用传统的 read 函数可能需要数秒时间，而采用内存映射方式，读取时间可能缩短至几百毫秒[3]。其次，内存映射文件能够实现进程间的高效通信。多个进程可以将同一个文件映射到各自的虚拟地址空间，从而共享文件中的数据，实现数据的交互和同步。再者，文件内存映射提供了一种简洁的文件访问方式，简化了程序代码[4]。开发人员无需编写复杂的文件读写逻辑，直接对映射后的内存区域进行操作即可完成对文件数据的访问和修改。此外，内存映射文件还支持“懒加载”特性，即操作系统不会一次性将整个文件加载到内存，而是在进程实际访问文件的某一部分时，才将该部分数据从磁盘加载到内存，这在处理大文件时能够有效地节省内存空间[5]。

3. 基于文件内存映射的海量数据存取优化方案

3.1. 数据存储格式设计

在基于文件内存映射的数据存取方法中，合理设计数据存储格式对于提高数据检索效率至关重要。本文采用一种结构化与索引相结合的存储格式，该存储格式分为文件头、数据区、索引区三部分，各部分具体设计如下表 1：

Table 1. Data storage format
表 1. 数据存储格式

文件整体布局		
文件头 (128 字节)	**数据区 (可变大小)**	**索引区 (可变大小)**
FileHeader -file_size -data_versI/On -record_offset -record_num -record_size -index_offset -index_end	DataRecord 1 DataRecord 2 DataRecord 3 … DataRecord N	IndexNode 1 IndexNode 2 IndexNode 3 … IndexNode M

1) 文件头设计

文件头固定大小(128 字节)，主要包括文件大小、数据版本号、记录开始位置、记录数量、记录的大小、节点开始位置、节点结束位置等。通过文件头能够快速定位索引区节点的位置以及数据区的位置。

2) 数据区设计

数据区在文件头之后，对于每一条数据记录，将其划分为多个固定长度的字段，数据记录直接追加到数据区末尾。例如，对于一个包含用户信息的数据记录，可能包含用户 ID (设为 8 字节的整数类型)、用户名(设为 32 字节的字符串类型)、用户年龄(设为 2 字节的整数类型)等字段。通过固定字段长度，可以方便地在文件中定位和读取每一条记录。

3) 索引区设计

索引采用 B+树结构存储在数据区之后。B+树是一种平衡的多路搜索树，具有高效的范围查询和精确查询性能[6]。在构建索引时，每一个索引节点包含若干个键值对以及指向子节点的指针。键值即为关键

字段的值，通过将键值按照升序排列存储在节点中，使得在进行查询时，可以快速定位到目标键值所在的节点范围，进而通过节点中的指针找到对应的数据记录在文件中的偏移位置。需要说明的是数据区和索引区均为可变大小，在实现时应考虑避免索引区和数据区的重叠问题，在本方案中，在数据区更新前，事务管理模块将索引读取到内存中，在内存中维护索引，索引更新提交时一次性写入到文件中，同时更新文件头中的索引区起始位置。

3.2. 多线程读写分离架构

为了充分利用系统资源，提高数据存取的并发性能，本方法采用多线程读写分离架构来实现文件的读写操作。在本方案中，将读操作和写操作分别放在不同的线程中独立执行可减少阻塞，提高系统吞吐量，优化响应时间，读写操作流程如下图 1。

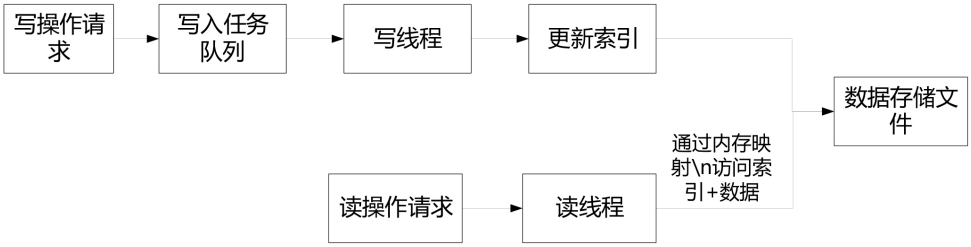

Figure 1. Reading and writing operation flow
图 1. 读写操作流

3.3. 数据更新与一致性维护

为了确保数据的一致性，本方案引入了版本控制机制。在文件头记录一个全局的数据版本号。每一次对文件进行数据更新操作时，将版本号加 1。同时，在每一条数据记录中，也记录该记录的版本号。当读线程读取数据时，如果发现版本号发生变化，说明该数据记录可能在读取过程中被其他线程更新过，此时应用程序可以根据具体业务需求选择重新读取数据或者采取其他处理方式，从而保证数据的一致性和准确性。

4. 方案实现

针对以上方案，采用 C++语言实现，其实现框图如图 2 所示。

框图采用垂直分层结构，从上至下体现模块的逻辑依赖关系，客户端接收数据写入和查询请求，并将请求分发给相应的线程模块。数据写入请求分发给写线程，数据查询请求分发给读线程，读写线程分别完成数据的读写操作。图 3 和图 4 分别是读写操作的实现流程图。

5. 方案验证

5.1. 实验条件

1) 实验环境配置：操作系统为 Windows 10 专业版，处理器为 Intel (R) Core (TM) i7-9750H CPU @ 2.60 GHz 2.59 GHz，内存为 16 GB DDR4 3200 MHz，硬盘为三星 980 PRO 1TB NVMe M.2 SSD。实验使用的编程语言为 C++，数据库采用 MySQL 8.0。

2) 实验数据：实验数据集包含 1000 万条模拟的用户信息记录，每条记录包含用户 ID (8 字节整数)、用户名(32 字节字符串)、用户年龄(2 字节整数)、用户地址(64 字节字符串)等字段，数据集总大小约为 1 GB。

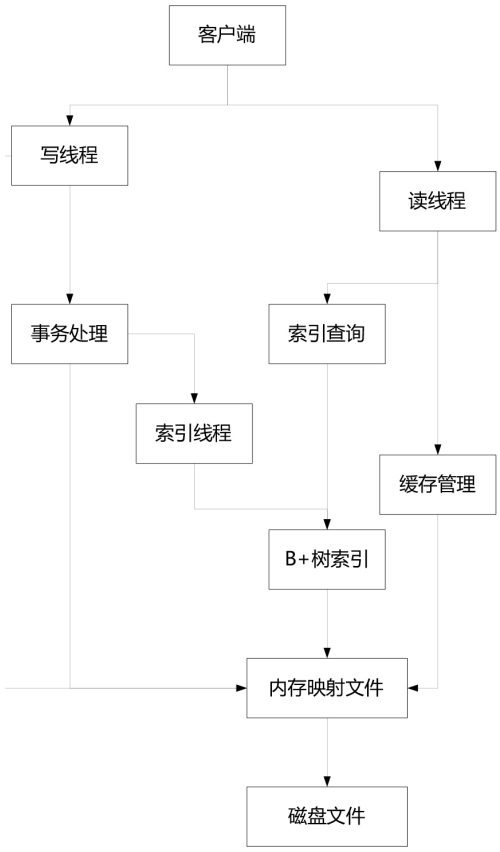

Figure 2. Block diagram of the system structure
图 2. 系统结构框图

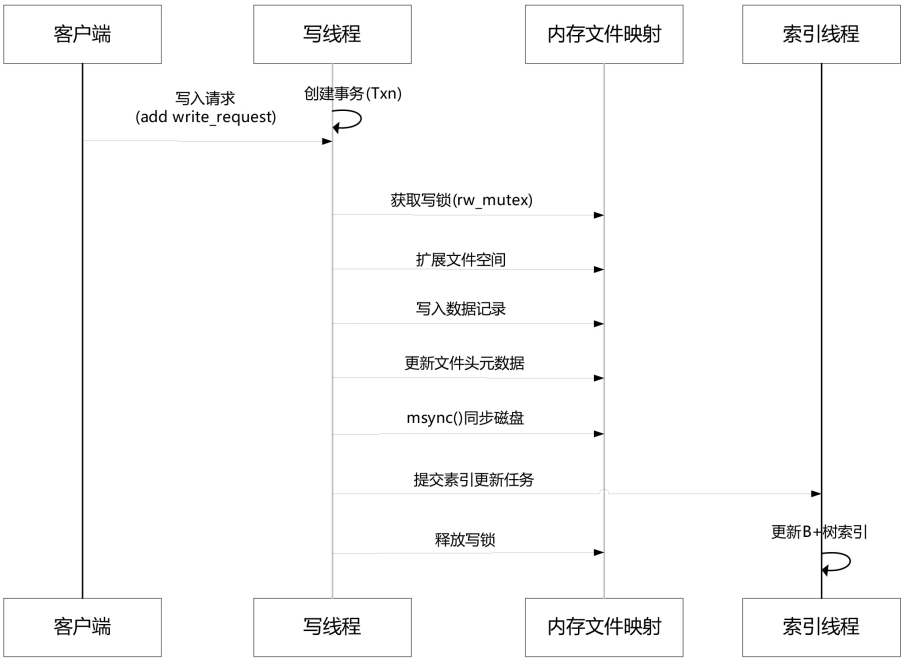

Figure 3. Flowchart of write operation
图 3. 写操作流程图

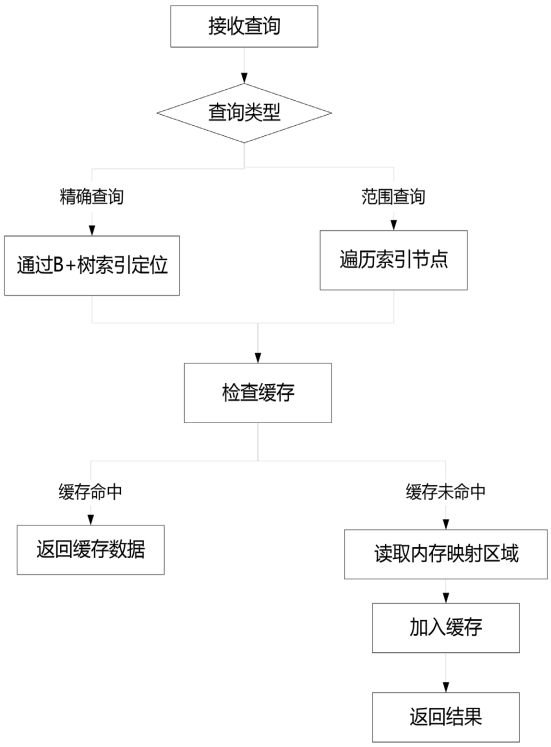

Figure 4. Flowchart of read operation
图 4. 读操作流程图

5.2. 检索速度对比

检索速度对比实验中分别测试了直接读写文件、使用本文方案以及使用数据库(MySQL)在不同查询条件下的检索速度。对于直接读写文件，采用逐行读取比对的方式进行数据检索；对于本文方案，利用构建的索引和内存映射机制进行检索；对于数据库，利用其自身的索引机制进行检索。测试结果如下表2所示：

Table 2. Comparison of retrieval speed
表 2. 检索速度对比

查询方式	精确查询(ms)	范围查询(ms)
直接读写文件	20,000~30,000	50,000~80,000
本文方案	5~10	20~50
数据库(MySQL)	3~8	15~30

从表中数据可以看出，在精确查询和范围查询场景下，本文方案的检索速度明显优于直接读写文件，虽然略逊于数据库，但差距较小。而在实际应用中，数据库的创建和维护成本较高，本文方案在保证较高检索效率的同时，具有更低的成本和更好的灵活性。

5.3. 写数据速度对比

在写数据速度对比实验中测试了文件 I/O 读写、使用本文方案以及向数据库(MySQL)写入数据的速度。测试结果如下表3所示：

Table 3. Comparison of data writing speed
表 3. 写数据速度对比

写入方式	写入 100 万条数据(s)	写入 1000 万条数据(s)
文件 I/O 读写	5~8	40~60
本文方案	2~4.8	25~36
数据库(MySQL)	30~50	300~500

从表中数据可以看出，在写入大量数据时，本文方案速度最快。综合检索速度和写数据速度来看，本文方案在数据保存和读取效率上实现了较好的平衡。

6. 结论

本文提出的基于文件内存映射的海量数据存取方法，为解决大数据场景下的高效存取问题提供了一种创新且实用的解决方案。通过技术整合与机制优化，该方案在性能、成本与灵活性之间取得了良好平衡，具有重要的工程应用价值。未来研究将围绕分布式扩展、索引优化与安全性增强等方向展开，进一步提升方案的普适性与可靠性，推动其在更多大数据场景中的落地应用。

基金项目

北京农业职业学院 2025 年度校级科研创新团队项目，项目编号 XY-TD-25-06，项目名称《日光温室智能控制系统与装备技术研究》；北京农业职业学院 2025 年度校级教育教学改革研究项目一般课题，项目编号 NZJGC202508，项目名称《高职大数据技术专业〈农业数据分析〉课程活页式新型教材建设研究》。

参考文献

[1] 刘平, 贾林林. 内存映射技术在大数据实时存储中的应用[J]. 河南科技, 2017, 607(5): 39-41.

[2] 孙文庆, 刘秉权, 肖镜辉. 基于内存映射文件的数据共享技术研究与应用[J]. 微计算机应用, 2005(2): 192-194.

[3] 段小芳, 刘丹. 内存映射技术在大数据存储应用中的研究[J]. 通信技术, 2020, 53(5): 1174-1178.

[4] 贾琴勇, 郭庆平. 内存映射文件在大型数据文件中的实现及其优越性[J]. 电脑知识与技术(学术交流), 2007(17): 1352-1353.

[5] 黄向平, 彭明田, 杨永凯. 基于内存映射文件的高性能库存缓存系统[J]. 电子技术应用, 2020, 46(7): 113-117+126.

[6] 邹驰. 基于 B+树索引的结构化数据文件的并发操作和安全加密研究[D]: [硕士学位论文]. 武汉: 华中科技大学, 2024.