北京理工大学

本科生毕业设计（论文）开题报告

学　　院： 计算机学院

专　　业： 计算机科学与技术

班　　级： 07111906

姓　　名： 陈林峰

指导教师： 陆慧梅

二〇二三年一月六日

毕业设计（论文）开题报告评审表

<table>
<tr><td colspan="2">姓名</td><td></td><td>学号</td><td></td><td>班级</td><td></td><td>专业</td><td></td></tr>
<tr><td colspan="2">导师</td><td></td><td colspan="2">校外导师（职称）</td><td colspan="2"></td><td>校外导师单位</td><td></td></tr>
<tr><td rowspan="3">论文选题</td><td>题目名称</td><td colspan="7"></td></tr>
<tr><td>题目性质</td><td colspan="7">软件开发（　　）　　理论研究（　　）
工程设计（　　）　　技术科学研究与工程技术研究（　　）</td></tr>
<tr><td>题目来源</td><td colspan="7">结合科研　　（　　）　　结合生产实际（　　）
结合实验室建设（　　）　　自拟题目　　（　　）</td></tr>
<tr><td rowspan="4">评审组成员</td><td>姓　名</td><td colspan="2">职　　称</td><td colspan="3">工作单位及职务</td><td colspan="2">签　　字</td></tr>
<tr><td></td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2"></td><td colspan="3"></td><td colspan="2"></td></tr>
<tr><td>评审意见</td><td colspan="8">（含：选题意义；选题是否满足毕业要求；技术方案是否可行；进度安排是否合理等）</td></tr>
<tr><td>成　绩</td><td colspan="8"></td></tr>
<tr><td colspan="9">评审组长签字：　　年　　月　　日</td></tr>
</table>

注：成绩以“合格”“不合格”记；评审组长为高级职称人员。

1. 毕业设计（论文）选题的内容

本课题主要研究基于 key-value 数据库的文件系统设计与实现。目前主流操作系统上的文件系统诸如 linux 上的 Ext 系列、brtfs、zfs，windows 的 NTFS 等都在底层的存储设备上要求具备一定的数据格式，这些数据格式根据文件系统种类的不同而具有较大差异。本课题通过使用 key-value 数据库作为底层数据结构管理存储设备，向上提供数据库接口，使用提供的数据库接口，实现一个包含常用 POSIX 接口的文件系统，最后，将实现的文件系统与其它已有的文件系统进行对比，分析基于数据库的文件系统实现带来的优缺点以及其使用场景。

2. 研究方案

2.1 研究背景

计算机的文件系统是一种存储和组织计算机数据的方法，它使得对数据的访问和查找变得容易，文件系统使用文件和树形目录的抽象逻辑概念代替了硬盘和光盘等物理设备使用数据块的概念，用户使用文件系统来保存数据而不必关心数据实际保存在硬盘（或者光盘）的地址为多少的数据块上，只需要记住这个文件的所属目录和文件名。在写入新数据之前，用户不必关心硬盘上的哪个块地址没有被使用，硬盘上的存储空间管理（分配和释放）功能由文件系统自动完成，用户只需要记住数据被写入到了哪个文件中。随着时间的推移，存储需求不断变化，数据量不断增加，文件系统必须是可靠的、持久的、安全的、高效的、容错的和可扩展的[1], 为了实现这些特性并跟上不断变化的计算需求和存储需求，不同的技术和文件系统随着时间的推移而被引入到操作系统中，但是绝大多数的文件系统实现仍然采用的是树状层次结构，提供给用户的接口仍然是 open,read/write 等。

在文件系统快速发展的同时，数据库系统同样也在随着需求的不断变化而快速迭代。数据库是以电子方式存储的系统数据集合，它可以包含任何类型的数据，包括文字、数字、图像、视频和文件。这些数据按一定的数据模型组织、描述和存储，具有较小冗余度、较高数据独立性和易扩展性，并可为各种用户共享。通常，数据库的实现以文件系统为基础，所有的数据库操作，如增删改查、备份等最后都会转换为对磁

盘上文件的操作，但有时数据库并不想受到文件系统实现带来的影响，一个广泛的现象是许多数据库系统的实现都会包含一个缓存模块，以此来提高其性能和可靠性。除了一直占据主要地位的大型关系型数据库系统，如 Mysql、SQLserver 等，近年来也出现了一些嵌入式数据库和许多 nosql 数据库，这些数据库与传统的数据库相比，其对系统资源的占用更小，如 sqlite，其与应用程序运行在同一个地址空间中，而不是像 Mysql 那样存在服务器端和客户端的区别，同时，新兴的 nosql 数据库虽然功能不如传统的大型关系型数据库完善，但根据需求的变化，这些 nosql 数据库在一些专有领域也有着不错的表现。

从数据库和文件系统的发展来看，两者是在互相改进，共同发展，从两者的功能来看，他们都是存储数据的一种管理方式。既然两者的功能如此相像，那一个值得思考的问题就是为什么大多数的数据库系统实现以文件系统为基础，而不能反其道而行之？文件系统的实现者是否可以重用数据库的基础设施以获得数据库提供的更多特性？更近一步地，文件系统的实现是否可以完全在一个数据库系统提供的接口上进行？

2.2 本选题的主要任务

基于数据库的文件系统实现曾经是研究热点，但由于性能以及其它原因，相关的项目最终都没有得到有效的落实。当前嵌入式数据库蓬勃发展，涌现出许多适用于不同场景的数据库实现，本文关注如何让文件系统与数据库系统相互融合，使得文件系统不仅可以在数据库之上构建，同时还能利用数据库的优势提供给用户更多接口使用。本课题主要完成以下工作:

1. 文件系统探究：分析 linux 系统中文件系统常用接口与功能
2. key-value 数据库探究: 分析使用 rust 实现的 jammdb 数据库系统，总结其相关接口与功能并做相关的移植
3. 使用 rust 编程语言实现一个基于 key-value 数据库 (jammdb) 的文件系统 (DBFS)，文件系统支持常用的 posix 接口，根据数据库的功能拓展文件系统功能
4. 将 DBFS 与其它使用 rust 实现的文件系统 (FAT32) 进行对比分析，总结使用数

据库系统构建文件系统带来的优势和不足之处。

2.3 国内外研究现状

早期，数据库系统的基础是底层的文件系统，许多管理数据的特性都只是在数据库系统上得到体现，而近年来，文件系统的设计开发中越来越借鉴了数据库相关技术的思想，二者不断融合发展。日志文件系统就是在借鉴数据库系统的日志和事务特性之后发展出的一种文件系统。在对文件系统进行修改时，需要进行很多操作，比如在扩展一个文件的大小时，需要对文件的元信息和磁盘元信息都需要修改，这些操作可能中途被打断，但是，理论上这些操作不是不可中断的。如果操作被打断，就可能造成文件系统出现不一致的状态，虽然可以使用一些工具对这些情况进行修复，如 fsck，但这些工具在修复时需要扫描整个磁盘，繁琐且速度较慢。日志文件系统在修改磁盘数据时利用日志信息记录修改过程，如果文件系统发生崩溃，重启后只需要根据日志信息就可以恢复数据从而保证数据的一致性和完整性。发展较早的日志文件系统 JFS[2]、XFS[3] 如今在某些系统上仍然广泛使用。而较新的 Btrfs[4]、Ext4[5]、NTFS 等具有日志功能的文件系统在服务器系统与桌面系统上占据着主要地位。除了对数据库的日志和简单事务的借鉴之外，许多适应新需求的文件系统实现在设计中会直接包含事务 ACID 特性的设计，exF2FS[6]是一个事务性的日志文件系统，其允许事务跨越多个文件，应用程序可以显式指定与事务关联的文件，同时其允许使用少量内存执行事务并在一个事务中封装多个更新。DurableFS[7]提供了一种受限制的事务形式：文件打开和关闭之间的操作自动形成具有原子性和持久性保证的事务。

一些文件系统的实现不再局限于在文件系统的设计上引入数据库系统的特性，而是选择直接基于数据库系统进行文件系统的实现，考虑到数据库系统提供的特性的实现复杂度，这种方案在实施难度上要远远小于重新设计文件系统。IFS[8]是一个基于关系型数据库 POSTGRES 实现的文件系统，其向用户提供事务特性以及时间旅行功能，同时其使用多张关系表来构建层级文件系统以最大限度地支持传统 POSIX 接口。Oracle iFS 是一个运行在数据库之上的文件系统，实质上，Oracle iFS 提供了文件系统和数据库之间的范式转换，同时提高了一些高级搜索和数据备份功能。国内学者[9]将 Oracle iFS 移植到 linux 的 VFS 模块中使得用户对其使用就如原有的文件系统一样，并且将基于内容分类、关联访问、基于内容的访问、版本控制等功能扩展到 VFS 中。

AMINO[10]是一个具有 ACID 语义的文件系统，其使用 Berkeley Database 作为后备存储，将一个易于使用但功能强大的嵌套事务 API 导出到用户空间。应用程序可以开始、提交和中止事务，同时其设计了一个简单的 API，使协作进程能够共享事务。使用相同的 API，单个应用程序可以支持多个并发事务。该文件系统不仅为应用程序提供了 ACID 的额外好处，在性能方面也与 ext3 相当。

2.4 技术方案的分析、选择

已经有许多学者在文件系统中引入数据库系统特性或者直接在数据库系统上构建文件系统，这些文件系统在提供基础的文件系统功能上还引入了新的特性，有效地保障了文件系统的一致性和完整性。本课题根据前人已有的实践经验，使用一个已有 key-value 数据库构建一个文件系统，并支持常见的 POISX 接口。

2.4.1 研究方法

本文采用的研究方法主要包含下面三种。

文献研究法: 通过图书馆、互联网、电子资源数据库等途径查阅大量文献，理解数据库系统、文件系统等相关知识，学习数据库实现的有关理论知识，获取前人在数据库和文件系统融合方面的研究信息，为后续的 DBFS 设计提供思路和参照。

实验研究法: 通过梳理选取的 jammdb 数据库实现，设计通用的层级文件系统在数据库中的表现形式，将 jammdb 数据库移植到 riscv 裸机平台上，并实现设计的文件系统，良好的设计能减少由于数据库限制带来的性能损失。

比较分析法: 通过提供常见的 POSIX 文件系统接口，将实现的文件系统与其它文件系统 (FAT32) 进行对比，分析 DBFS 与常见的文件系统实现性能差距与优势。在前期的调查分析基础上，可以预想到 DBFS 在文件元数据扩展，查找速度以及小文件存储上可以带来比传统文件系统更大的优势。

2.4.2 技术方案

本课题的技术方案主要包含四个阶段。

在第一阶段，主要完成的是 jammdb 数据库的分析和移植工作。jammdb 数据库

是一个使用 rust 编程语言实现的嵌入式单文件系统，是 go 语言实现的 BoltDB 的移植，支持将 key-value 存储为字节数据。jammdb 提供 ACID 特性，具有多个无锁读取器和一个并发的写入器，所有的数据被组织为一棵 B+ 树，因此随机和顺序读取非常快。jammdb 不提供完整的数据库的功能，而专注于插入值与删除值，因此在嵌入式系统中使用非常方便。jammdb 数据库的移植工作主要包含两个方面，一是分析 jammdb 的实现，理清数据库对操作系统的依赖，二是将 jammdb 数据库移植到 riscv 裸机平台上。jammdb 使用单个文件作为底层的存储，依赖操作系统的内存映射和几个文件相关的系统调用，在移植的过程中，需要将这些依赖抽象出来，使用统一的接口。目前已完成将依赖剥离的工作，为了检验正确性，使用了一个内存模拟的虚假文件进行了测试。在完成接口的抽象之后，需要将 jammdb 放到操作系统内核中，由于 linux 系统与 rust 的交互不是很方便，因此本文选择了使用一个 rust 实现的简单操作系统对 DBFS 进行测试和分析，并且这个简易的操作系统运行在 riscv 平台上，使用 qemu 进行模拟。

在第二阶段，主要完成文件系统的设计和实现。jammdb 数据库只提供了几个简单的接口，而文件系统需要提供几个较为复杂的读写访问接口，在数据库中并没有直接对应文件系统层级结构的数据结构，因此需要基于数据库的相关结构重新设计一种可以表达文件系统层级结构的存储方式。有两种方案可以进行选择，其一是基于 jammdb 提供嵌套 bucket 数据结构，其二是直接对层级目录进行哈希映射，将层级文件系统转换为一个单层的表现形式。

在第三阶段，主要完成文件系统实现的优化和扩展。在第二阶段中，我们已经完成了一个简单的文件系统的设计和实现，但是这个文件系统还有很多不足之处，其对于数据的存储和访问有很大的改进空间，并且没有提供文件元数据扩展功能。在第三阶段中，我们将对文件系统进行优化和扩展，使其能够满足一些实际的使用需求。基于嵌入式数据库的文件系统在存储一些小文件和不发生修改的文件具有较大的优势，例如小文件在传统文件系统中可能会占用许多不必要的空间，这是由于传统文件系统一般基于数据块进行存储分配，而数据库对数据的存储是紧密排列的，因此对于一块相同大小的空间，DBFS 就可以存储更多的数据。对 DBFS 的优化集中于设计文件数据在数据库中的存储方式，以及扩展文件系统接口，使得用户可以对存储的文件进行元数据扩展，并且可以根据自定义的元信息进行文件的检索。

- 提供用户接口使得用户可以增删改查文件元数据
- 直接导出数据库接口，使得用户既可以正常使用文件系统，也可以使用数据库
- 添加文件时通过参数控制文件存储方式

在第四阶段，主要完成 DBFS 与传统文件系统的对比分析，既然 DBFS 在存储小文件以及一些不发生修改的文件具有优势，那么就需要与传统的文件系统在这些方面进行对比测试，通过实现特定场景的测试程序，可以得到两种文件系统的性能数据。

2.5 实施技术方案所需的条件

此部分说明所需的软硬件等环境，建议使用表格的形式，方便表述。

表 2-1 硬件、软件环境

	指标	版本参数
硬件环境	CPU	Intel i5-8265U
	RAM	8 GB
软件环境	操作系统	Linux 5.15.79.1-microsoft-standard-WSL2
	编程语言	rustc 1.66.0-nightly
	qemu	6.1.0

2.6 存在的主要问题和技术关键

目前存在的主要问题是数据库的裸机移植与文件系统设计。

2.6.1 数据库裸机移植

目前大多数的数据库实现都依赖操作系统提供的内存管理，文件系统，线程等功能，这些功能在裸机环境下是不存在的，因此需要对数据库进行裸机移植，使其能够在裸机环境下运行。本文选择的嵌入式数据库 jammdb 依赖于操作系统的内存映射和文件系统功能，而我们的目标是要将其作为文件系统的底层存储，即数据库需要直接管理存储设备，如何将数据库依赖的内存映射和文件系统功能对应到内核功能上，是

本文实现中的一个关键问题。因为 jammdb 数据库是单文件数据库，本文当前的解决方案是将存储设备直接抽象为一个文件，并在这个基础上实现数据库需要的文件接口，比如 seek,extend 等，这样对数据库实现的修改就会大大减少，再将内核的内存映射功能做一些修改，完成数据库需要的内存映射接口，这样就可以实现数据库的裸机移植。

2.6.2 文件系统设计

jammdb 是一个 key-value 类型的数据库，与常见的文件系统提供的接口差别很大，要使用数据库提供的接口构建文件系统接口需要对数据库的存储格式进行必要的设计，jammdb 中存在一个 bucket 的数据格式，bucket 中存储了 key-value 对，一个 key 可以对应一个 value，也可以对应一个 bucket，在这个基础上可以很直接地构建层级的文件系统，但由于 rust 语法的限制，当前这种想法并不能快速实现，需要修改数据库的实现。而另外的思路则是设计特殊的存储格式，使得数据库可以表达文件系统的层次结构。数据库没有像文件系统那样提供修改文件内容的接口，而只有删除和插入，如果每次对文件的修改都要进行一次删除和插入操作，将会导致性能低下，因此 DBFS 的实现还需要关注文件在数据库的存储形式。

2.7 预期能够达到的研究目标

2.7.1 软件成果

本文的研究和实现有助于数据库和文件系统的进一步融合，使得文件系统可以利用来自数据库的优势。同时移植使用 rust 实现的数据库以及使用 rust 实现文件系统也可以给 rust 语言生态带来更多的应用，目前有多个使用 rust 实现的桌面操作系统和嵌入式系统，本文的成果可以为这些系统提供更多的功能。

2.7.2 文章成果

完成论文一篇

3. 课题计划进度表

大致的课题计划进度如下所示。

- 2022.12-2023.1: 参与选题并完成开题报告
- 2023.1-2023.2: 分析数据库实现，将数据库移植到 riscv 裸机平台，设计文件系统实现方式，完成文件系统的基本功能
    1. 修改数据库接口，在不影响其原有功能的基础上可以自定义底层文件依赖
    2. 在 qemu 模拟的 riscv 平台上实现定义的接口，使得 jammdb 可以运行与裸机上
    3. 根据数据库的数据结构或者自定义数据结构完成层级文件系统的抽象和实现
    4. 实现 POSIX 定义的大部分文件系统调用接口
- 2023.2-2023.3: 完成文件系统的高级功能，优化文件系统的实现，提高文件系统性能
    1. 针对文件内容存储方式，优化文件读写性能
    2. 直接导出数据库接口，使得用户在同一块存储设备上既可以使用数据库也可以使用文件系统
    3. 针对数据库的 key-value 带来的便利，实现用户对文件元数据的扩展
- 2023.3-2023.4: 将基于数据库的文件系统实现与其它文件系统进行对比分析，总结使用数据库系统构建文件系统带来的优势和不足之处。
    1. 完成 jammdb 的正确性测试
    2. 编写测例完成对文件系统实现的正确性测试
    3. 编写测例测试不同文件系统的性能，分析原因
    4. 编写测例使用 DBFS 的扩展功能，对比在传统文件中实现相关功能带来的代价

- 2023.4-2023.5: 完成论文撰写，完成课题报告
- 2023.5-2023.6: 参与论文答辩

4. 参考文献

[1] IRUM I, RAZA M, SHARIF M, et al. File Systems for Various Operating Systems: A Review[J]. Research Journal of Applied Sciences, 2012, 4(17): 2934-2947.

[2] BEST S. {JFS} Log: How the Journaled File System Performs Logging[C]//4th Annual Linux Showcase & Conference (ALS 2000). 2000.

[3] SWEENEY A, DOUCETTE D, HU W, et al. Scalability in the XFS File System.[C]//USENIX Annual Technical Conference: vol. 15. 1996.

[4] RODEH O, BACIK J, MASON C. BTRFS: The Linux B-tree filesystem[J]. ACM Transactions on Storage (TOS), 2013, 9(3): 1-32.

[5] CAO M, BHATTACHARYA S, TS'O T. Ext4: The Next Generation of Ext2/3 Filesystem.[C]//LSF. 2007.

[6] OH J, JI S, KIM Y, et al. {exF2FS}: Transaction Support in {Log-Structured} Filesystem[C]//20th USENIX Conference on File and Storage Technologies (FAST 22). 2022: 345-362.

[7] KALITA C, BARUA G, SEHGAL P. DurableFS: a file system for NVRAM[J]. CSI Transactions on ICT, 2019, 7(4): 277-286.

[8] OLSON M A, et al. The Design and Implementation of the Inversion File System.[C]//USENIX Winter. 1993: 205-218.

[9] 张步忠. 基于数据库的一个文件系统的移植[D]. 苏州大学, 2005.

[10] WRIGHT C P, SPILLANE R, SIVATHANU G, et al. Extending ACID semantics to the file system[J]. ACM Transactions on Storage (TOS), 2007, 3(2): 4-es.