University of Science and Technology of China

GBFS*图文件系统
详细设计报告

丁峰　牛田　谢灵江　张立夫 †

2018 年 7 月 2 日

*Graph Based File System

†丁峰：PB16110386　牛田：PB15010419　谢灵江：PB16111096　张立夫：PB15020718

目录

1 成员介绍

小组成员介绍:

丁峰 计算机学院大二学生，熟练使用 google scholar 查找文献，学习过 C 语言、数据结构、计算机系统概论，初步了解 linux 内核编译

牛田 学习过 c 语言，以及所有计算机学院前期的基础课程。初学 java、python、html。

谢灵江 计算机学院大二学生，学习过 c 语言，以及所有计算机学院前期的基础课程

张立夫 掌握编程语言：C C++ Python HTML CSS JavaScript，参与过前端开发

总体来说，组员掌握的技能都比较基础，后期项目实现过程都进行了大量相关知识的学习。

2 立项依据

如今，我们的生活越来越离不开我们的个人电脑。然而，电脑给我们的生活带来了极大的便利的同时，也给我们增加了不少烦恼。随着我们越来越依赖自己的电脑，我们的电脑也存储了越来越多的文件。毫不夸张地说，一个人一台电脑里就有至少几个 G 甚至几十个 G 的数据。而相应的，一个普通用户缺少高效管理数量如此庞大的文件办法。到目前为止，手工处理文件分类仍然是许多用户的唯一选择。然而，手工对文件分类的弊端十分明显。第一，人的记忆力有限。如果说寻找几天前存放的文件还是一件很简单的事情，那么要寻找一个几年前存放的文件，在已经忘记了文件的具体名字和它的路径的情况下，寻找它犹如大海捞针。第二，手工分类时遇到分类难题。例如，究竟是按照创建时间分类各个文件，还是按照文件内容分类，还是按照文件的功能性分类，还是按照文件的所有者分类，不同人对这几个分类标准的喜好不同，这几个分类标准运用场景也不同，很难选择一个都认可的分类方法。就算确定了一个详细的分类标准，将其严格执行也要花费不少

精力。而且，万一在归类文档时出现错误，那么这个文档就很可能难以被发现。第三，文件路径的限制。在目前流行的文件系统中，一个文件基本上只有一个路径，导致文件与文件之间没有很强的逻辑关系，因此常常会出现打开了一个文件之后，它的相关文件缺失的情况。更加糟糕的是，多重的文件层次导致文件绝对路径特别长，难以记忆和传输。

事实上，许多操作系统厂商已经意识到了这一个问题。Apple's spotlight,Linux's KDE Baloo, Windows Desktop Search 都提供了本地文件的搜索引擎。然而，在用户完全忘记了文件名的情况下，这些工具就毫无用武之地。而且用户对其搜索结果并不满意，因为它们的搜索结果是基于字符串匹配而非被搜索文件的内容上的含义，即不符合语义搜索的要求。并且，这些搜索引擎不提供对用户文件分类的服务。

基于上述几点不足，小组成员打算开发一种可视化的图文件系统。该文件系统基于知识图谱的相关知识对计算机的个人文件数据进行显示分类。这项工作主要的目的在于弥补传统文件系统无法表示的复杂逻辑关系的缺陷。图文件系统使用图表示文件间的联系，从知识图谱中获取相关的信息去构造文件间的联系。

3 项目架构

3.1 总体架构

总体架构如下图所示，通过 FUSE 来实现文件的基本操作，将相关操作发送至 Server ，由 Server 对相关指令解码后对图数据库 Neo4j 急性相关节点的操作。用户可以通过 GBShell 来实现对数据库的检索与修改。具体部件功能介绍如下

3.2 FUSE

主要功能:

- 为整个图文件系统提供一个底层文件系统，挂载点与 ext4 中某个文件夹有镜像关系。
- 为了保证 Neo4j 中的数据总是最新的，用户对文件的操作后，所产生的数据需要传送给 Neo4j。这一功能由 fuse 来完成。

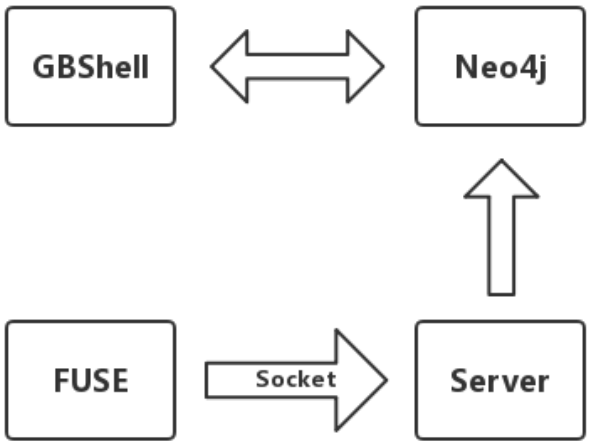

图 1: 总体架构框图

- 根据文件名及其后缀，把不重要的文件进行筛除，只发送给 Neo4j 一些比较重要或者用户经常使用的文件。例如，.dll，.swp 后缀名文件就会被筛出，而对.pdf，.ppt，.doc 的文件的操作，fuse 就会把数据发送到 Neo4j 中。
- 引入推荐模块（Recommend），对服务端 (Server) 发出推荐请求
- 通信方式——socket 信道。在用户对文件系统中的文件进行创建、删除、重命名、打开、更改权限等操作时，文件系统会把更新的信息传给 Neo4j，以使 Neo4j 中的数据保持最新。先在本地和 Neo4j Server 连接，再使用 socket.send 把数据传入 Neo4j 和 Server。

3.3 知识图谱

知识图谱本是 google 的一个知识库的名称，现在被用来泛指各种大规模知识库。其特点是以 RDF 三元组的模式存储数据，与图等价，能迅速有效地查找到与一个数据相连结的关联内容。利用知识图谱可以对文件系统中的文件作出一些初始的属性判断，在生成或放入文件的同时调用知识图谱，从其名称中推测出可能发生关联的方向以及类型。从文件名中提取出的

属性联系将会成为除了用户手动操作以外的最主要的关联依据。此次图文件系统中，考虑到贴近于当前生活的情况，我们使用的知识图谱为复旦大学提供的中文知识库 CN-DBpedia，其中有较为丰富全面的中文的数据信息。

3.4 Neo4j 与 Py2Neo 介绍

Neo4j 是我们为了实现图文件系统对文件的“图”逻辑关系存储而选择的图数据库，或者也可以称为“关系数据库”。顾名思义，Neo4j 数据库中有两种基本的元素：节点（Node）和关系（Relationship）。节点和关系都可以被赋予多个标签（Label）以及属性（Property），关系连接着图中的节点，这中设计使得我们可以给存储的数据赋予“图”的逻辑结构，并对“图”中的节点和关系进行操作，并实现常规数据库无法实现的图相关的算法（如 Dijkstra 算法）。在本项目中，我们将文件作为节点，文件的标签可以来源有两种：

- GetProperty 模块从知识图谱中搜索而来；
- Shell 模块提供了添加标签的操作，使用户可以给文件添加自定义的标签。

文件的属性是由 FUSE 给出的，如文件名，文件拓展名、UID、访问时间等等。而文件节点之间的关系我们是这样连接的——如果两个文件节点有相同的标签（不妨记为 label1），那么就给这两个节点之间赋予一个关系，关系的标签为 label1，属性为这两个文件之间的关联度（即带权图中边的权值，weight），默认值为 1。我们采用的 Neo4j 的接口为 Py2Neo，Py2Neo 是一个 python 语言操作 Neo4j 的工具包，提供了很多高级的 API，方便了我们在 python 程序和命令行中使用 Neo4j，我们的程序中对于 Neo4j 的操作全部是采用 Py2Neo 实现的。

4 模块具体设计

4.1 FUSE

使用 python3.6 实现整个 FUSE。在 github 库中，对应的文件是 GBFS_1.py。首先，定义一个类，里面实现了各种操作系统级别的方法。整个文件系统事实上是一个传递式的文件系统，它仅仅是接受一个目录，并在挂载点将其显示出来，使任何在挂载点的修改都会镜像到源数据中。所以，它应该表现的

与原有的文件系统尽可能一致。因此，这段代码的大部分方法只是对 os 模块的一些简单封装。比如，可以直接给它们赋值，但是在这里，我们需要修改一下路径，以及实现与 Server 进行通信等。前文已经提到，利用 Socket 信道与 Server 保持通信是为了保证 Neo4j 能自动更新里面的数据。具体编写时，需要考虑到以下几点：

- 保证通过 Socket 发送的包的字节数是一致的，我们小组规定，一个 Socket 包只传递 100B 的数据，这样会极大方便 Server 的接收工作。为了严格保证每个包只传递 100B 的数据，fuse 会在发送数据前计算此次通信需要多少个包，并在数据量不足 100B 时，补上 0 将包填满。
- 在每次通信的过程中，Socket 会包含一些文件操作的基本信息：此次通信需要传递的包的数量，操作符命令，文件的绝对路径，文件名，文件名后缀，文件的 atime，ctime，mtime，文件 UID。这些信息都是用逗号隔开的。操作符命令有 Create，Read，Unlink，Rename，Chown，Open。举个例子，当我们打开一个 Neo4j.pdf 时，fuse 会通过 Socket 发送：0,Read,/home/xxx/root/Neo4j.pdf,Mon Jul 2 15:22:59 2018,000000000
- 并不是所有的文件都要加入到 Neo4j 数据库中。如前文所述，后缀名不符合要求的文件不会被放入 Neo4j 中。因此，FUSE 需要实现筛选功能。对文件后缀名符合要求的（例如：.txt，.pdf，.c，.ppt，.doc，.xsl，.py 等等）文件进行操作时，才 FUSE 会将数据发送给 Neo4j。
- FUSE 与预读调用。根据前文，fuse 在打开一个文件时，会调用 Recommend 模块，然后得到 3 个与其相关联的文件，在 Server 端对其进行预读处理。

4.2 Server

考虑到直接从 FUSE 模块对 Neo4j 进行更新操作会使得打开文件以及文件读写操作的延迟提高，降低使用效率。为减少这种延迟，我们采用了通过 Socket 由 FUSE 向 Server 进行通信的方式，来由 Server 模块进行数据库的更新等操作。通过 Socket 从 FUSE 模块获得操作指令，将其进行分词，以获得对应的具体操作参数。根据开头数字标示选择从 Socket 中读取几个包来获取到完整的一个操作指令，然后通过调用 py2neo 库来实现对 Neo4j 数据库的更新。对于 Server 部分有如下图中几种操作：

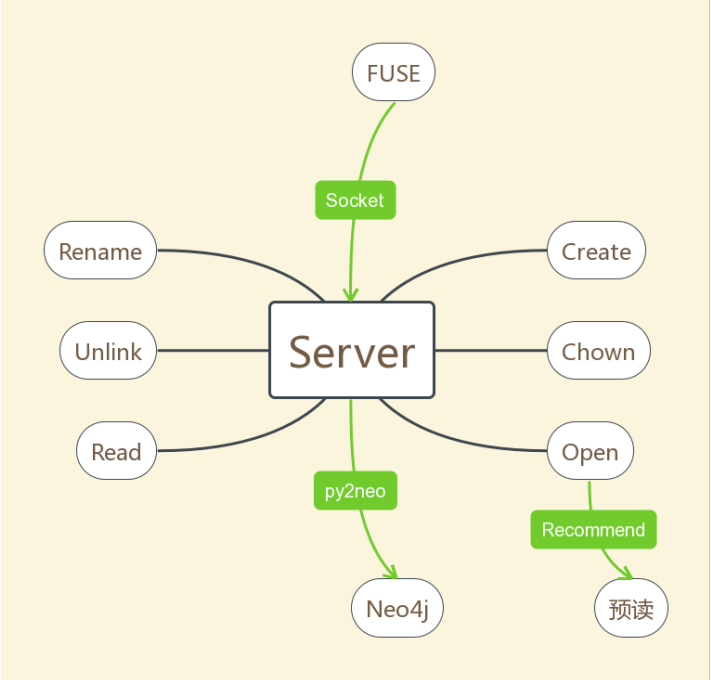

图 2: Server 功能框图

- Create：创建一个新的节点，并在 Server 中调用 GetProperty 模块来获取与新建文件名相关的三个标签，并将其作为标签添加到新创建的节点中，正是因为 GetProperty 从知识图谱中获取标签需要一定时间，所以才会将该部分的操作放至 Server 端。除此之外还会添加文件相关的固有属性，如文件名，文件路径，创建修改时间以及文件后缀名等。
- Open：对数据库中对应的所打开文件的节点的相关信息进行修改。除此之外，还在这一部分实现预读操作：通过调用 Recommend 模块，获取到三个与该文件最相关的文件路径，将这三个文件打开并且读取 4MB 的大小到内存中，之后关闭文件。由于真正实现预读操作需要对操作系统的底层进行操作，所以我们仅是将近期可能被打开的文件先打开一遍，由于时间局部性，实现二次打开的提速。
- Unlink：删除对应文件节点，并将该节点所相连的边进行删除。
- Rename & Read & Chown：根据文件所被修改的属性，会对该文件节点的相应属性进行修改。

4.3 GBShell

通过 GBShell 模块实现了一个简单的命令行操作功能。具体参数与对应操作如下图所示：

- -a：给一个或多个文件节点添加自定 label，支持通配符 *，会给具有相同 label 的所有节点之间添加边相连。
- -s：显示一个或多个文件节点的全部 label 与属性，支持通配符 *。
- -l：显示一个或多个文件节点的所有相邻文件节点，会显示相邻文件路径，支持通配符 *。
- -r：对一个文件进行相关推荐，调用 Recommend 模块，获取得到五个相关文件，并提供相关文件选择并打开操作。
- -f：根据 label 与属性对文件节点进行搜索，可以同时交叉搜索多个 label 与属性，并可以对搜索到的文件进行选择并由默认程序打开。
- -d：删除一个或多个文件节点的一个或多个 label，支持通配符 *，由于 label 为自定属性，所以可以通过该参数进行删除，删除一个 label 的同时会删除所有与该节点相连的具有该 label 的边。

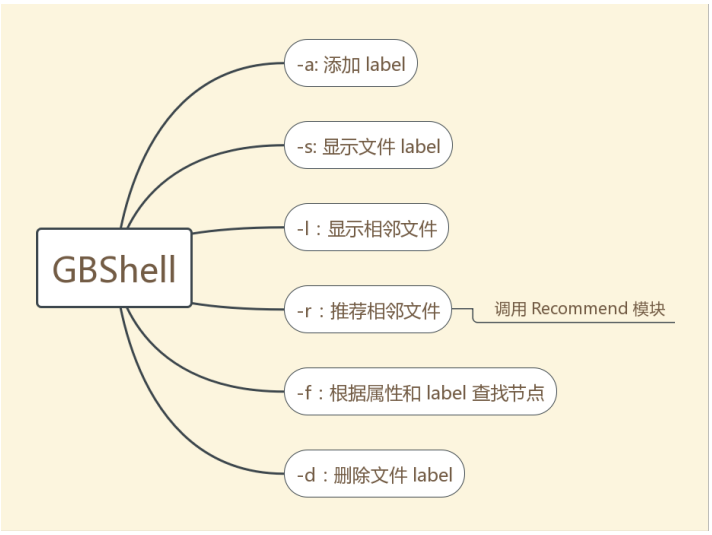

图 3: GBShell 功能框图

4.4 Recommend

Recommend 模块的具体功能是为用户提供文件推荐服务，当用户输入一个文件时，GBShell 模块会调用 Recommend 模块，列出与被输入文件关联度最大的 5 个文件。用户只需从这 5 个文件中进行选择即可由默认程序进行打开。推荐功能是基于图数据库——Neo4j 来实现的。在项目结构中我们已经简要介绍了 Neo4j 是如何存储我们的文件节点以及之间的关系的，Recommend 模块将会通过 py2neo 对 Neo4j 进行操作，来选取与输入的文件节点关联度最大的 5 个文件节点，并按照关联度从大到小的顺序进行排序，最终列出来供用户选择。用户可以选择自己想要的文件，Recommend 程序将工作目录跳转到该文件所在的文件夹，用户也可以选择留在当前文件夹进行工作。之后，Recommend 程序会在 Neo4j 数据库中更新文件之间关系的关联度（weight）。如我们之前提到的，我们将每个关系的关联度默认设为 1，当一个文件所关联的文件被选择后，对应的关系的关联度（or 权值）将会增加 1，增加的上限为 10，到达上限后权值不再增加。而未被选择的文件所对应的关系的权值将会与我们设定的阈值比较，如果权值 >=5，则权值减 1，如果权值小于 5，则不再减少。通过以上的三个步骤：1. 权值排序 2. 用户选择 3. 更新权值，我们完成了一次推荐的操作。由于时间有限，我们实现的推荐算法比较简单，通过后续的工作可以进一步提出更优化

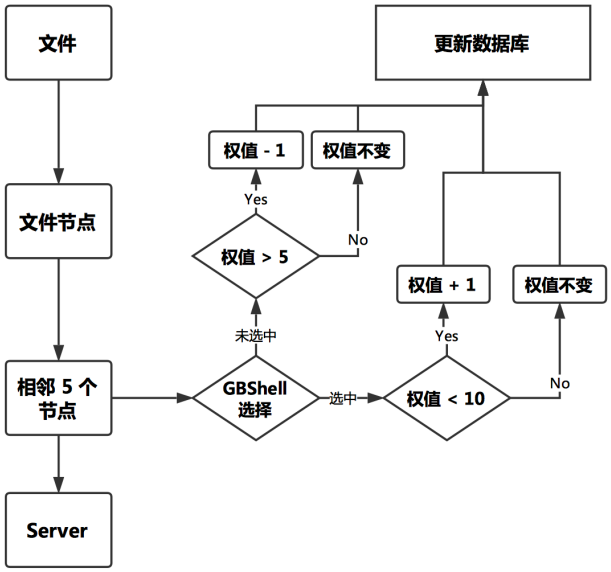

图 4: Recommend 流程图

的推荐算法，给用户带来更好的使用体验。

4.5 GetProperty

GetProperty 模块通过数据库（复旦大学 CN-DBpedia）对新增的文件添加一些基础的标签。当一个新的文件被添加时，Server 模块调用 GetProperty 模块，利用 CN-DBpedia 数据库提供的 API 接口，通过网络来获取相关信息。首先使用文件的名称，在数据库之中查找到可能性权重较大的对应词条，成功后再将词条的信息取出（前三条相关信息，若与词条相关内容不足三个则全部取出），返回给 Server。值得一提的是，在知识图谱中，数据是三元组，但本系统调用时得到的信息是作为标签存储的，即为了操作的便利放弃了关系。例如，当【OS-中文名称-操作系统】被取出返回时，“中文名称”会被舍弃，仅把“操作系统”作为文件“OS”的标签进行添加。

5 实现情况

5.1 FUSE 实现情况

fuse 的筛选功能实现完成，可以筛选出需要的文件，有.pdf .txt .ppt .mp3 .avi .doc .xsl .c .py .cpp 等后缀名文件可以被放入到 Neo4j。与 Server 的通信方法实现完成，Socket 包严格按照一个包 100B 传输。但依然存在一个问题，在 fuse 中，挂载点与另一个文件夹有镜像关系。这样设计的好处是可以实现硬盘空间的对应，但不足之处是，在与 shell 的对接过程中，fuse 内部的绝对路径与操作系统的绝对路径不是等价的，经过修改后不会出现问题，但却要求用户必须挂载在一个名为 mountpoint/的挂载点上。GetProperty 实现了文件名称的可能的对应知识词条的搜索，以及将词条的相关数据作为 label 加入文件之中的功能。比较不足的是因为接口的问题并不能对其数据权重有较为有效的判断，只能取靠前的三元组作为有效的值。

5.2 Recommend 实现情况

Recommend 模块的推荐功能在编写过程中遇到了很多问题，由于负责编写的同学第一次接触 python，以及编写时沟通不足，编写完成后出现了很多意想不到的 bug。bug 的原因有以下几类：

- while 循环未更新变量的值导致死循环；
- 未考虑两个文件节点之间有多个相同标签的情况，导致推荐时可能推荐的 5 个关联最大的文件有重复的情形；
- Py2Neo 文档不详细导致实现某些函数时，必须手动实现，而不能调用 Py2Neo 中的 API，然而手动实现导致了一些问题：比如文件节点的文件名不能包含空格的问题；
- 未考虑文件系统中文件数量不足 5 个的情况；
- 其他由于代码编写不仔细导致的问题。

5.3 GBShell 实现情况

实现了在模块具体设计中的所有计划功能，并且对通配符 * 和对文件夹内文件统一操作的支持，对搜索和推荐功能得到的文件可以直接通过选

择使用系统默认程序进行打开。

5.4 可扩展性

由于采用 Socket + Server 方式进行对数据库的更新，所以可方便的移植到分布式系统中，采用一个服务端进行数据库的更新。对于推荐系统，可以采用更为合适的推荐算法，以提高推荐效果。对于知识图谱部分，可根据不同使用方向，采取其专业领域的知识图谱，甚至采用自然语言处理的方式进行相关标签的提供，以此可以提高标签准确率。预读部分则需要通过深入底层来实现真正意义的预读，而这个则需要操作系统方面的配合与实现。以上都是可以在未来进行加强优化的部分，并且可以通过以上优化来得到比较好的使用体验。

6 分工安排

丁峰 前期负责 FUSE 实现的调研，后续 FUSE 的编写实现以及 FUSE 与 Server 的 Socket 通信工作。参与调研报告、可行性报告以及详细设计报告的编写。

谢灵江 负责知识图谱方面的资料查询，功能应用。完成 GetProperty 部分的编写实现，辅助完成少量 GBShell 部分的输入输出提示工作。参与调研报告、可行性报告以及详细设计报告的编写。

牛田 前期调研时负责 Neo4j 图数据库，和 Py2Neo 的调研、学习以及使用。写代码时负责 Recommend.py 模块的编写以及完善，Recommend 模块包含 shell 以及 server 两个部分，分别与 GBshell 以及 Server 连接。同时也参与了可行性报告、详细设计报告的编写。

张立夫 组织小组成员进行每周讨论，对项目整体进度进行把控与任务分配。前期进行相关文献调研与调研报告及可行性报告的编写与排版编译。后期设计项目整体架构，编写 Server 与 GBShell 模块，实现各个模块之间的整合与测试。编写与编译详细设计报告。

参考文献

[1] David Gifford, Pierre Jouvelot, Mark Sheldon, James O'Toole. Semantic File Systems. ACM Operating Systems Review, Oct. 1991, pp 16-25

[2] Daniele Di Sarli,Filippo Geraci.GFS: a Graph-based File System Enhanced with Semantic Features.

[3] Michael Schuhmacher,Simone Paolo Ponzetto. Knowledge-based Graph Document Modeling

[4] Wei Jin,Rohini K. Srihari. Graph-based Text Representation and Knowledge Discovery

[5] Stephan Bloehdorn,Max Völkel.TagFS —Tag Semantics for Hierarchical File Systems

[6] Adrian Sampson Design for a Tag-Structured Filesystem

[7] Py2Neo v4 handbook http://py2neo.org/v4/index.html

[8] Neo4j Tutorial https://neo4j.com/developer/get-started/

[9] https://github.com/skorokithakis/python-fuse-sample

[10] 参考 2017 Spring OS(H) 课程中学长们相关的工作实现