计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(10):20−28 [doi: 10.15888/j.cnki.csa.005512]

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

面向桌面环境的索引实时更新方法①

杨凯飞[1,2], 李文波[1], 柯　川[1,2]

[1](中国科学院 软件研究所 基础软件国家工程研究中心, 北京 100190)
[2](中国科学院大学, 北京 100049)

摘　要: 在桌面计算环境中, 文件和目录频繁发生新建、删除、修改、重命名、移动、复制等变化, 这对桌面索引更新的实时性和性能提出更高要求, 而传统的桌面索引更新方法完全或部分依赖周期性全盘扫描, 往往需要大规模索引重建, 导致索引生成延迟大、系统资源占用高. 针对这些弊端, 本文提出了一种基于文件系统事件监听的桌面索引实时更新方法, 并实现了相应的桌面索引实时更新系统. 实验表明: 本文提出的索引更新方法延迟低、系统资源占用低.

关键词: 桌面搜索; 文件系统监控; 索引更新; 文本抽取; inotify

引用格式: 杨凯飞,李文波,柯川.面向桌面环境的索引实时更新方法.计算机系统应用,2017,26(10):20–28. http://www.c-s-a.org.cn/1003-3254/5512.html

Index Real-Time Updating Method for Desktop Environment

YANG Kai-Fei[1,2], LI Wen-Bo[1], KE Chuan[1,2]

[1](National Engineering Research Center of Fundamental Software, Institute of Software, CAS, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In the desktop computing environment, files and directories are frequently changed, such as being created, deleted, modified, renamed, moved, and copied, which has higher demands on the real-time updating of desktop index and its efficiency. However,the traditional desktop index updating methods are fully or partially dependent on periodically full scan, which inevitably requires a large-scale index reconstruction and leads to long delay of index generation and high usage of system resources. In order to overcome these drawbacks, this paper puts forward a new desktop index real-time updating method based on a new file system events monitoring, and implements corresponding desktop index real-time updating system. Experimental results show that the proposed index updating method provides lower latency and lower usage of system resources.

Key words: desktop search; file system monitoring; index update; text extraction; inotify

随着用户所掌握的数据量迅速增加和 PC 存储容量的大幅增长, PC 中存储的文档、图片、视频等越来越多, 这些数据量大且存储无序、不规则, 管理、查找较为困难. 从如此大规模的文档集中寻找所需耗时费力, 桌面搜索引擎的重要性逐渐凸显. 桌面搜索引擎用于索引和检索 PC 中的文档信息, Google、Yahoo、Microsoft、百度等先后推出桌面搜索引擎.

类似于网络搜索引擎, 桌面搜索引擎多采用先为桌面文档建立索引再提供查询的方式. 这极大地提升了查询速度, 但也带来了桌面索引更新的问题, 桌面索引需及时更新, 才能保证其与文档内容一致. 因为桌面文档频繁发生新增、删除、修改、重命名、移动、复制等变化, 桌面索引更新的实时性和性能要求更为严苛.

① 基金项目: 国家高技术研究发展计划 (863 计划)(2013AA01A603)
收稿时间: 2016-04-06; 采用时间: 2016-04-27

桌面索引更新的基本方式有两种: 1) 周期性扫描所有文件和目录并检测其变化, 为发生变化的文件或目录更新索引. 该方法需周期性遍历所有文件和目录, 不仅性能差, 而且延迟时间长, 无法保证索引更新的实时性; 2) 基于操作系统或第三方的文件系统事件监听方法, 实时监听文件系统事件并更新索引. 这种方法实时性好、效率高. 现有桌面索引更新系统多采用方式 1) 或以其为主[1], 往往需要周期性重建索引, 导致索引更新延迟大、系统资源占用高. 方式 2) 虽然高效、实时, 但 Linux 内核文件系统变化通知机制存在无法监听子目录、冗余信息未过滤等典型缺点, 依靠其实现的现有监听方法存在监控效率低、稳定性差、维护复杂等缺点. 这也是方式 2) 未被桌面索引更新系统单独和广泛采用的重要原因.

针对桌面环境的特性和现有桌面索引更新方法的弊端, 本文提出了一种基于文件系统事件监听的桌面索引实时更新方法, 无需周期性扫描全盘即可保证索引内容与桌面文档完全一致. 本方法能够实现对文件系统事件的实时监听以及快速更新索引内容. 我们研究了现有 Linux 文件系统事件监听方法, 在此基础上, 提出了一种 Linux 文件系统事件实时监听方法并实现, 克服了现有方法的典型缺点. 最后, 在 Linux 操作系统上实现了桌面索引实时更新系统, 采用了多种性能优化策略提升其索引速度. 实验表明: 本文提出的桌面索引实时更新方法延迟时间短、系统资源占用低.

1 相关工作

桌面搜索引擎采用预先为桌面文档建立索引的方式来提升用户检索信息的速度, 但由于桌面文档频繁发生变化, 桌面索引需要频繁并且实时更新. 桌面索引实时更新的核心是及时获取到文件的变化, 现有桌面索引更新方法完全或部分依赖周期性全盘扫描的方法获取发生变化的文件和目录, 这种方法延迟大、效率低. 采用实时监听文件系统事件的方式可以大幅提升索引更新的实时性和效率, 所以本文首先研究了现有 Linux 文件系统事件监听方法.

除了需要实时监听文件系统事件, 桌面文档频繁变化还带来了桌面索引更新策略如何选择的问题, 尤其是目录发生变化时如何更高效地更新桌面索引的内容. 桌面索引中一般会存储文档的绝对路径等元数据信息, 目录的变化会导致其所包含的所有文件均需要更新, 表 1 列举了目录发生不同类型的变化时对桌面索引的影响. 在不同的变化类型下, 选择不同的更新策略会对系统性能产生较大影响. 例如, 若目录发生重命名事件, 一种策略是先从索引中删除该目录下所有文件的索引信息, 再从新目录中抽取所有文件信息并为其重建索引. 第二种策略是在倒排索引中定位所有需要更改的信息, 修改部分元数据信息即可. 若目录中包含大量文件, 则第二种策略在性能上明显占优. 相应的, 目录删除、移动、复制等变化同样需要采取优化的索引更新策略.

表 1 目录的变化对桌面索引的影响

类型	桌面索引的变化
新建	添加目录下所有文件的信息
修改	更新目录下所有文件的信息
删除	删除目录下所有文件的信息
重命名	更新目录下所有文件的信息
移动	删除原目录下所有文件信息, 添加新目录下所有文件信息
复制	添加复制后的目录下所有文件的信息

1.1 Linux 内核文件系统变化通知机制

Linux 内核先后引入了 dnotify、inotify、fanotify 三种文件系统变化通知机制. dnotify 从 Linux 2.4.0 内核开始引入, 是三者中最早引入的, 但是存在很多缺陷. inotify 从 Linux 2.6.13 内核开始引入, 完全替代了 dnotify, 也是三者中目前使用最广泛的[2-5]. fanotify 从 Linux 2.6.36 内核开始引入, 与 inotify 相比各有所长.

dnotify 可以监控文件系统中目录的变化事件, 目录中某个文件发生访问、新建、删除、更改、属性修改等变化时均会发出通知. 但是 dnotify 只能监控某个目录中发生的变化, 而无法及时获取发生变化的具体文件, 需要进一步获取信息, 而且它需要对每一个被监控目录打开一个文件描述符, 若文件所在磁盘需要卸载, 会受到影响. 此外, 它返回的事件信息较少.

inotify 监控粒度更细, 可获取发生变化的文件信息, 而且监控的事件类型很多, 涵盖文件新建、删除、移动等多种类型事件. 但是, inotify 的一个监视 (watch) 只能监控一个目录, 无法递归监控其子目录, 而且单个用户可创建的 inotify 实例数目和每个 inotify 实例可关联的监视数目均有限制, 两者的默认值分别为 128 和 8192. 在 Linux 环境下, 可分别通过修改配置文件 “/proc/sys/fs/inotify/max_user_instances”和“/proc/

sys/fs/inotify/max_user_watches”修改两者的默认值[6,7].

fanotify 既可以通知文件系统变化, 也可以拦截文件系统变化, 应用程序使用 fanotify 的访问控制 (Access Decision) 功能可以决定是否允许其它应用程序对文件的操作[8]. fanotify 提供三种文件系统变化监控模式: 全文件系统监控 (Global)、单个挂载点监控 (per-mount)、单个对象监控 (Directed). fanotify 不仅可以监控其它程序对文档和目录的操作, 还能决定是否允许其操作. 表 2 比较了三种机制监听文件系统事件的特性.

表 2 Linux 内核文件系统变化通知机制比较

机制	特性			
	最小监控粒度	事件通知方式	调用接口	监听子目录内事件
dnotify	目录	信号	fcntl()	否
inotify	文件	read()	inotify_init()	否
fanotify	文件	read()	fanotify_init()	否

在桌面搜索场景中, Linux 内核文件系统变化通知机制有两点主要不足:

1) 单个监视 (watch) 无法监听子目录内部的事件. 例如, 对 inotify 而言, 单个监视只能监听到目录中文件和直接子目录的变化, 而子目录内事件的监听需由新的监视负责. fanotify 有三种模式, 虽然其 Global 模式和 Per-mount 模式可以分别实现对全盘和某个挂载点的全部文件系统事件的监听, 但其依然无法实现在监控某一目录对象时监听包括其子目录在内的所有文件系统事件, 而且 fanotify 监听的事件类型过少, 根本无法满足桌面索引实时更新系统的需求.

2) 存在冗余信息. 例如, 为保证事件无遗漏, inotify 未过滤临时文件的变化信息, 这为桌面索引更新带来不便.

1.2 第三方的文件系统事件监听库

通过封装 Linux 内核提供的文件系统变化通知机制, 第三方的文件系统事件监听库函数也可为应用程序提供对文件系统事件的实时监听. 下面对两个提供 Java API 的监听库进行重点研究.

1) JNotify 是一套 Java API, 可以为 Java 应用程序提供监听文件系统事件的功能, 这些事件包括文件和目录新建、删除、修改和重命名等. 在 Linux 操作系统环境下, JNotify 实际上是 Linux Inotify API 的简单封装. Inotify 的单个监视不支持递归监控子目录, JNotify 通过为监控目录下的每个子目录递归创建一个监视实现了这个功能, 但这一过程的时间消耗也随着子目录层数的增加而呈现指数增长, 同时也带来了系统资源需求的增加[9,10];

2) JDK 自 1.7 版本开始, 提供了 WatchService API 供应用程序监听文件系统事件, 可监听事件包括文件和目录新建、删除和修改事件.

1.3 现有 Linux 文件系统事件监听方法的不足

Linux 内核文件系统变化通知机制中, dnotify 由于存在诸多不足, fanotify 由于支持的事件类型太少, 均无法满足需要. 所以, 现有 Linux 文件系统事件监听方法一般基于 inotify 实现.

在桌面环境中, 文件和目录频繁发生变化, 桌面索引的实时更新对文件系统事件的监听过程提出了实时、高效、无遗漏的要求. 由于 inotify 的单个监视无法监听子目录内部的事件, 所以若要监听某一目录范围内所有支持的文件系统事件, 需为其所有子目录添加监视. 这意味着桌面索引更新系统若要无遗漏的监控文件系统事件, 需要为所有子目录添加监视. 因为真正发生变化的子目录数目相对较少, 所以这种方式必然会添加大量无效的监视. 例如, Linux 中/home 目录的子目录个数可能达到上万个, 这意味着若要监听/home 目录范围所有文件系统事件, 添加的无效监视可能达到数千甚至上万个. 而且, 若监视超过默认值, 需要修改系统配置文件, 这也带来不必要的麻烦. JNotify 采用的就是这种方式, 这种方式导致监视的数目会随子目录层数的增长呈指数增长. 监视数目的增长会带来两方面问题: 1) 大量的监视会导致初始化耗时增加、监控效率下降; 2) 每个监视与子目录相关联, 子目录更改后需要增删监视, 而大量监视的维护与频繁增删操作过程复杂.

此外, 现有 Linux 文件系统事件监听方法的事件通知中存在很多冗余信息, 这会导致桌面索引更新系统进行无效的更新操作或导致索引更新失败. 例如, 当用户打开一个文档进行修改时, 编辑器一般会新建一个临时文件, 而这个临时文件的内容是不应该被索引的, 索引它会导致桌面索引中存在无效内容.

综上, 现有 Linux 文件系统事件监听方法无法满足桌面索引实时更新的性能和功能需要, 所以本文提出了一个 Linux 文件系统事件实时监听方法, 克服了现有方法的典型缺点.

2 一种 Linux 文件系统事件实时监听方法

文件系统事件监听方法是索引实时更新的核心, 针对现有 Linux 文件系统事件监听方法的缺点, 基于 Linux 内核文件系统变化通知机制, 提出一种 Linux 文件系统事件实时监听方法并实现.

2.1 方法的执行流程

现有 Linux 文件系统事件监听方法为所有子目录添加监视, 需要维护的监视数目随子目录层数的增加而成指数增长, 大量的监视提升了初始化耗时、维护成本等. 相比而言, Linux 文件系统事件实时监听方法对用户操作子目录行为进行监控, 只为可能发生文件变化的子目录添加监视. 既能够无遗漏地监听指定目录范围中所有支持的文件系统事件, 也大大减少了对无关子目录的监控, 只需维护极少数目的监视. 另外, 根据异常文件名称、文件是否存在、特殊符号等规则过滤了冗余信息.

Linux 文件系统事件实时监听方法的输入信息: 待监控目录、禁止监控的子目录, 输出信息: 发生变化的文件、变化类型 (新增、删除、修改、重命名).

如图 1, 描述了 Linux 文件系统事件实时监听方法的工作流程, 主要包括三步.

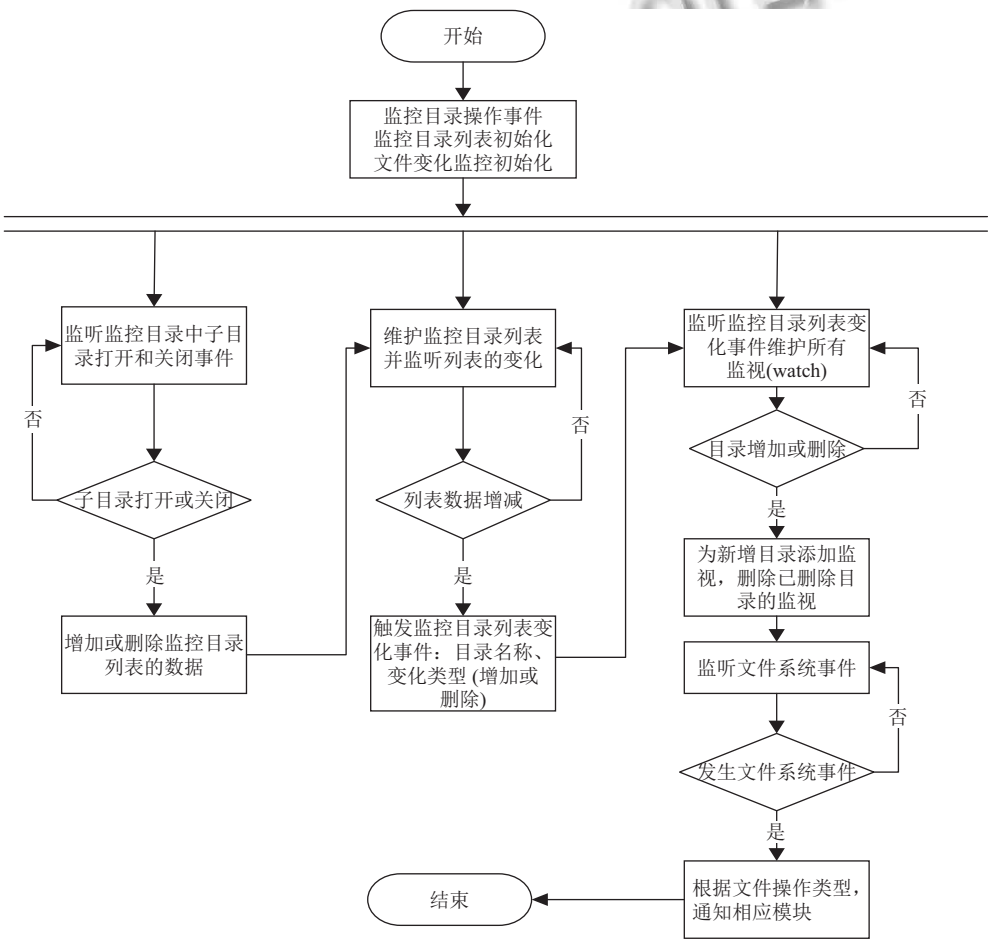

图 1 Linux 文件系统事件实时监听方法流程图

步骤 1. 对待监控目录的子目录打开和关闭事件进行监听, 基于规则过滤冗余信息. 监听到子目录打开或关闭事件后, 将打开的子目录信息添加到监控目录列表中, 并从监控目录列表中删除关闭了的子目录信息. 同时触发监控目录列表变化事件, 该事件中包含两个关键信息: 目录名称、操作类型 (添加或删除). 其中, 监控列表的长度由系统设定, 当达到最大长度时采用 LRU 算法替换列表中的数据.

步骤 2. 监听监控目录列表变化事件. 监听到事件后, 根据操作类型和目录名称添加或删除某个子目录的监视.

步骤 3. 每个监视监听目录中文件或直接子目录的

新建、删除、修改、重命名四种事件. 监听到事件后, 首先根据文件名称是否合法、隐藏属性、文件是否存在等规则过滤临时文件、缓存文件等引发的事件通知, 并通知外部应用.

2.2 监听系统实现

基于 Linux 文件系统事件实时监听方法设计并实现了相应系统. 系统包括三个模块: 用户目录操作行为监控、监控目录存储和文件系统事件监听, 如图 2 所示.

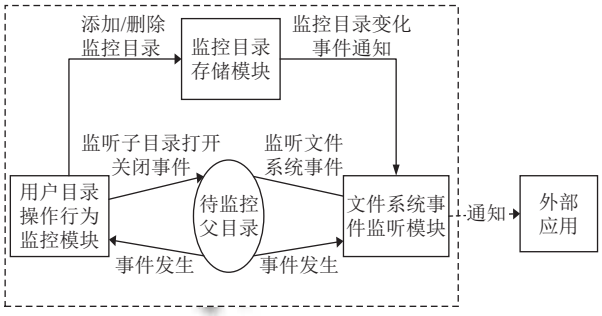

图 2 实时监听系统架构图

用户目录操作行为监控模块负责监听待监控目录中所有子目录打开和关闭事件, 并依据设置的禁止监控子目录过滤事件. 该模块基于 fanotify 实现.

监控目录存储模块负责维护监控目录列表, 它根据用户目录操作行为监控模块监听到的子目录打开和关闭事件增加或删除监控目录列表中的数据并发送事件通知给文件系统事件监听模块. 该模块基于 Java LinkedHashMap 实现.

文件系统事件监听模块负责根据监听到的监控目录列表变化事件添加或删除子目录的监视, 每个监视负责监听目录中文件和直接子目录新增、删除、修改、重命名事件. 当事件发生时, 文件系统事件监听模块通知使用该机制的外部应用. 该模块基于对 inotify 的 Java 封装 API 实现.

3 一种桌面索引实时更新方法

桌面索引更新过程可以细分为三个步骤: 1) 及时获取发生变化的文件或目录; 2) 基于文件系统事件类型采取合适的策略更新索引; 3) 快速获取发生变化的文件或目录的信息. 本文提出一种基于文件系统事件监控的桌面索引实时更新方法, 针对以上三个步骤分别给出了方案: 1) 基于 Linux 文件系统事件实时监听方法及时获取文件和目录的变化; 2) 根据发生变化的是文件或目录以及变化类型选择优化的索引更新策略, 避免目录发生变化时大规模更新甚至重建索引; 3) 在索引更新系统的实现中, 提出并采用了多种性能优化策略来提高文件信息提取速度.

方案 1) 的具体实现已在上一节详述, 方案 3) 中具体的优化策略将在下一节详述, 下面重点叙述方案 2).

在基于周期性遍历的索引更新方法中, 无法监听文件系统事件, 所以也就无法根据文件系统事件类型优化索引更新策略. 在基于文件系统事件监听的桌面索引更新方法中, 当监听到文件系统事件时, 根据发生事件的主体是文件或目录以及事件类型选择合适的索引更新策略. 在 Lucene 等开源搜索引擎的倒排索引结构中, 数据存储在逻辑上有 Document、Field 等概念, 分别可以类比为关系数据库中的一行记录和列. 在本方法中, 一个文件的信息在桌面索引中对应一个 Document, 每个 Document 都有一个唯一 ID, 我们将其设置为文件的绝对路径, 具体的索引更新策略如表 3 所示.

表 3 针对不同事件主体和类型的索引更新策略

事件类型	文件	目录
新建	新增Document	为目录信息新增Document
重命名	删除并新增Document	在ID字段通过前缀匹配确定所有相关Document并修改相应字段
修改	更新相应Document	同重命名事件
删除	删除相应Document	在ID字段通过前缀匹配确定所有相关Document, 批量删除
移动复制	转化为新建、修改、删除、重命名事件并处理	

图 3 为基于文件系统监控的桌面索引实时更新方法执行流程, 包括如下步骤:

步骤 1. 读取配置文件并判断其是否合法.

步骤 2. 获取用户配置的可索引目录, 监听该目录范围所有文件系统事件. 若目录新增事件发生, 跳转到步骤 3. 若文件新增、修改或重命名事件发生, 跳转到步骤 4. 若文件删除事件发生, 则跳转到步骤 7. 若目录删除、修改或重命名事件发生, 跳转到步骤 8.

步骤 3. 扫描所有文件, 获取格式符合要求的所有文档.

步骤 4. 抽取文件的内容, 同时获取文件的元数据:

文件名、文件格式、绝对路径. 其中, 若文件格式不属于用户规定的可抽取格式, 则将其文件内容字段置为空.

步骤 5. 对步骤 4 获得的文件内容和元数据进行分词, 扩充部分字段.

步骤 6. 以文件绝对路径作为 ID, 在索引中添加文档, 结束.

步骤 7. 获取文件绝对路径, 匹配索引中的 ID, 删除相应文档, 结束.

步骤 8. 若为目录删除事件, 获取目录绝对路径, 以前缀匹配方式获取索引中所有相关文档并批量删除. 若为目录修改或重命名事件, 获取目录绝对路径, 以前缀匹配方式获取索引中所有相关文档更新其 ID 和绝对路径字段, 结束.

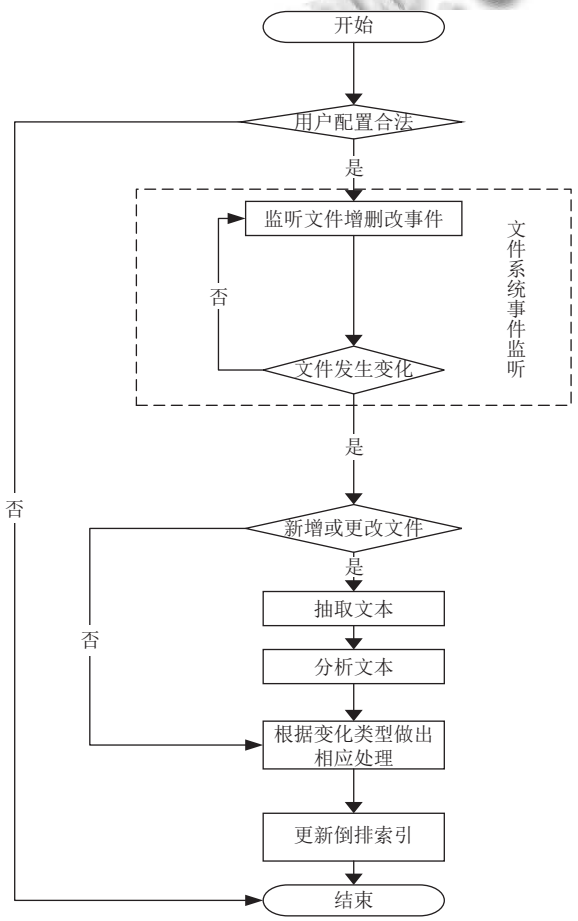

图 3 桌面索引实时更新方法流程图

4 桌面索引实时更新系统设计与实现

基于本文提出的桌面索引实时更新方法, 设计和实现了桌面索引实时更新系统.

4.1 系统结构设计

如图 4 所示, 本系统由日志记录、配置管理、事件感知、文本抽取、文本分析、索引更新六个模块组成. 其中, 所有模块均调用日志记录模块, 在图 4 中省略日志记录模块.

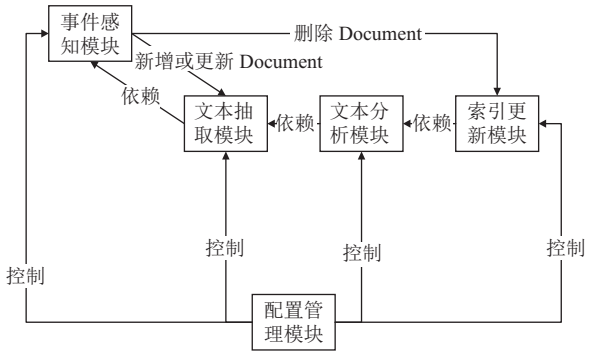

图 4 桌面索引实时更新系统框架图

日志记录模块负责记录系统索引文档的过程信息及可能出现的错误, 便于系统异常时查错; 配置管理模块负责管理用户配置信息; 事件感知模块负责监听可索引目录中文件系统事件; 文本抽取模块负责抽取文件内容和元数据; 文本分析模块负责为文件内容和元数据做分词等后续处理; 索引更新模块负责更新倒排索引.

4.2 系统实现

本系统使用 C、Java 开发, 在 Linux 操作系统实现.

日志记录模块记录如下信息: 被索引文件元数据信息、索引过程信息、系统异常信息. 根据日志信息的级别将其输出至日志文件或运行窗口. 该模块基于 Apache log4j 库实现.

配置管理模块将所有配置信息以键值对形式存储在程序根目录的“config.properties”文件中, 用户可修改该文件以灵活定制索引选项. 配置管理模块在系统启动时首先判断配置文件是否存在并检测配置信息是否合理, 然后获取配置信息并提供给其他模块. 用户可配置的选项包括: 可索引目录、禁止索引目录、可索引文件格式、可抽取文件格式.

事件感知模块基于对用户操作目录行为的监控, 只为可能发生文件变化的子目录添加监视, 每个监视负责监听单个目录中文件和直接子目录的新增、删除、修改、重命名事件, 但不包括其子目录中的文件系统事件. 该模块基于 Linux 文件系统事件实时监听

方法实现.

文本抽取模块抽取了 Word、Pdf、Txt、PPT、JPG 等常见格式文件的内容和元数据, 抽取的信息包括: 文件名称、绝对路径、文件格式、文件内容. 该模块基于 Apache Tika[11]实现, Tika 在搜索引擎、文本分析、文本翻译等场景中应用广泛[12-14].

文本分析模块基于 IkAnalyzer[15]分词工具实现, 对抽取模块得到的部分字段进行分词, 同时扩充部分字段.

索引更新模块基于 Apache Lucene[16]实现, 负责基于桌面文档的变化更新倒排索引. Lucene 将复杂的索引和检索过程以简单的接口呈现给应用[17], 在桌面搜索场景有广泛应用[18].

4.3 系统性能优化

桌面索引实时更新既需要及时获取文件系统变化, 也需要快速完成文档索引. 分析和测试了索引过程各个阶段的耗时, 提出并应用了多种优化策略以加快索引速度.

混合抽取策略. Tika 作为抽取框架, 为不同格式文件提供了统一的抽取接口. Tika 抽取文件内容包括三步: 文件类型检测、文件解析器选择、内容抽取. 若跳过前两步, 直接抽取文档将提升文本抽取速度. 因为 PDF 文件抽取相对较慢, 基于加快抽取速度和保证不同文件格式抽取接口统一性的考虑, 决定对 PDF 文件的抽取速度进行重点优化. 首先, 通过实验比较了不同数据集下 Tika 和 Pdfbox 抽取 PDF 的速度.

表 4 抽取测试数据集

编号	文件数目(个)	文件平均大小(KB)
1	1304	281
2	245	2335

表 5 pdf 抽取测试结果

工具	数据集1耗时(ms)	数据集2耗时(ms)
Tika	100	83
pdfbox	31	190

通过以上实验发现, 对小文件的抽取, Pdfbox 快于 Tika. 而对于大文件, Pdfbox 慢于 Tika. 基于以上结果, 系统采取如下混合抽取策略:

1) 若文件格式为 PDF, 且大小小于 1 MB, 直接调用 Pdfbox 抽取.

2) 其余文件调用 org.apache.tika.parser.Parser 类抽取.

多线程处理. 文本抽取和分词耗时较长, 所以采用多线程方式处理. 为兼顾速度与系统资源占用, 多线程基于以下原则: 1) 单个线程处理的文件数目固定; 2) 线程数目由文件总数决定, 但不超过最大值.

在系统实现中, 使用 StringBuffer 代替 String 进行字符串拼接等操作.

综合使用以上优化方法, 索引速度得到了极大提升, 其中, 混合抽取策略的提速效果最为突出.

5 实验

实验测试包括四项: 测试 Linux 文件系统事件实时监听方法的延迟; 测试桌面索引实时更新方法的延迟; 测试桌面索引实时更新系统的索引速度并与 DocFetcher、Recoll 的索引系统进行比较; 测试并比较桌面索引实时更新方法与基于周期性全盘扫描的索引更新方法的内存占用.

5.1 实验准备

四项实验测试均基于 Eclipse 开发环境, Ubuntu 64 位操作系统, 采用 Intel(R) Core(TM)i7-2600 CPU、14 GB 内存和 1 TB 硬盘的硬件平台.

第一项和第二项实验均通过连续复制单个文件到指定目录触发文件新建事件以及桌面索引更新, 模拟发生文件系统事件时的事件通知和索引更新过程.

第三项实验, 即索引速度对比实验选取的对比系统为 DocFetcher 1.17 和 Recoll 1.21.5, 均为这两款桌面搜索引擎的最新版本. Linux 平台上优秀的桌面搜索引擎有 Google Desktop Search、Beagle、DocFetcher、Recoll 等, 但前两者均已停止更新较长时间. DocFetcher 是一款开源、跨平台的桌面全文搜索软件, 支持多种格式文档的快速索引和检索[19]. Recoll 是一款基于 Xapian 开源搜索引擎的桌面全文桌面搜索工具, 它提供了强大的文本抽取层和完整、易用的基于 Qt 的界面[20]. 第三项实验中使用的数据集信息如表 6 所示.

表 6 实验数据集信息

编号	文件数量(千个)	文件大小(GB)	文件类型
1	1	0.651	pdf、txt、ppt、word、jpg
2	5	2.84	
3	10	4.85	
4	30	12.2	

5.2 实验结果和分析

第一项实验测试了在发生不同数目的文件系统事

件时, 本文提出的 Linux 文件系统事件实时监听方法的平均延迟时间. 如图 5, 随着文件系统事件的增多, 本方法监听到单个事件的平均延迟时间稳定在 1 ms 左右, 没有出现大幅增加或减少. 相比现有方法, 本方法仅需维护数目极少的监视即可实现对指定目录范围发生的文件系统事件的无遗漏监听, 降低了监视维护的复杂性及耗时, 能够稳定、实时地持续监听文件系统事件.

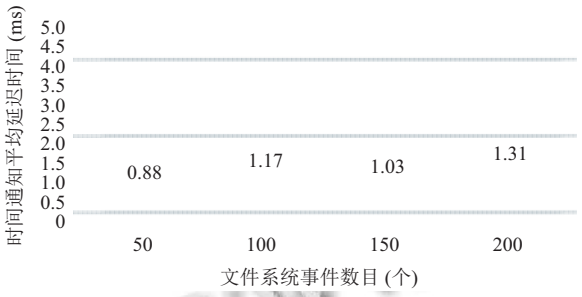

图 5 文件系统事件实时监听方法平均延迟时间

第二项实验测试了在发生不同数目的文件系统事件时, 本文提出的桌面索引实时更新方法平均延迟时间. 图 6 显示, 随着文件系统事件的增多, 桌面索引更新的平均延迟时间稳定在 90 ms 内, 延迟低.

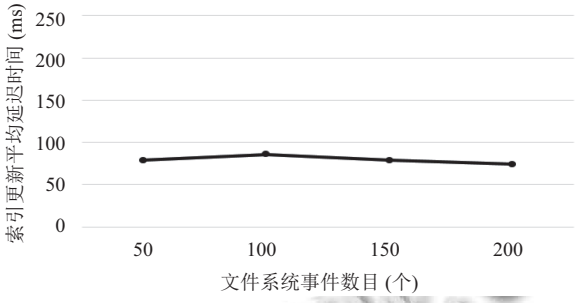

图 6 桌面索引实时更新方法平均延迟时间

第三项实验测试了本文实现的桌面索引实时更新系统的索引速度并与其它系统对比. 如图 7, 随着文件数目的增加, 三个系统耗时均不断增加, 其中 Recoll 随着文件数目的增加性能下降最为严重, 本系统的耗时远低于 DocFetcher 和 Recoll. 需要说明的是, 具体的测试数据与使用的数据集和硬件性能相关.

第四项实验对比本文提出的桌面索引实时更新方法与周期性索引更新方法的内存占用, 如图 8 所示. 可以发现, 本方法内存占用少, 而且较为稳定. 需要说明的是, 内存占用的具体数值与实验的条件有关. 本实验通过每五秒钟新建一个文件来模拟文档集的变化, 周期性扫描的时间间隔为 20 秒, 每隔一秒采集一次内存信息, 持续采集 150 秒.

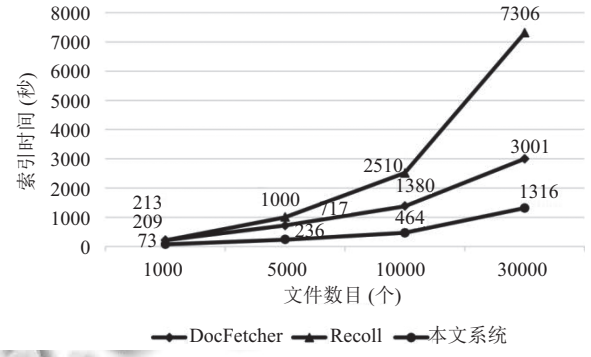

图 7 索引速度比较图

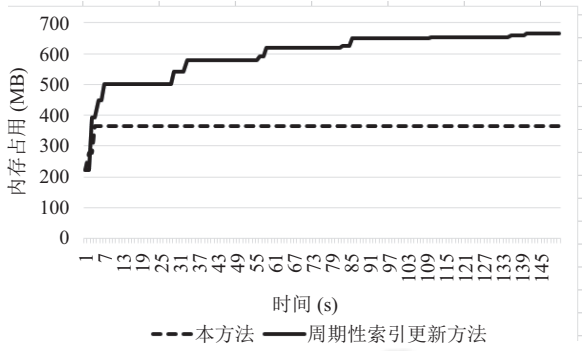

图 8 内存占用对比图

6 结语

现有桌面索引更新方法采用的周期性索引更新策略存在延迟大、系统资源占用高等缺点. 针对这些弊端, 本文提出并实现了一种基于文件系统事件监听的桌面索引实时更新方法, 无需定时扫描全盘即可保证桌面文档与索引内容完全一致. 为了实现高效、低延迟地监控文件系统事件, 研究了现有 Linux 文件系统事件监听方法, 针对其不足, 提出并实现了一种 Linux 文件系统事件实时监听方法, 仅需维护极少数目的监视即可实现对文件系统事件的无遗漏监听, 而且也对冗余的事件通知信息进行了有效过滤.

在处理目录删除、重命名事件时, 采取了批量更新倒排索引的策略, 避免了大规模索引更新或重建. 在系统的实现中, 综合应用了多种优化策略提升索引速度. 实验结果表明, 本文提出的方法在索引更新的延迟时间、系统资源占用方面均有较大提升. 本文实现的

系统能够监听和高效处理文件和目录的新建、删除、修改、重命名事件, 目录移动和复制等事件均转化为新建、删除、修改事件处理. 如何从高层语义上监听到移动和复制等事件并针对其优化索引更新策略需要进一步研究.

参考文献

1 Cen ZW, Xu JG, Sun J. SoDesktop: A desktop search engine. Proc. of the 2012 International Conference on Communication Systems and Network Technologies (CSNT). Rajkot, India. 2012. 463–467.

2 Morgan S, Mortazavi M, Palani G, *et al*. Metadata index search in a file system: U.S. Patent 20160063021. 2016-03-03.

3 Xu L, Jiang H, Tian L, *et al*. Propeller: A scalable real-time file-search service in distributed systems. Proc. of the 34th International Conference on Distributed Computing Systems (ICDCS). Madrid, Spain. 2014. 378–388.

4 Takata M, Sutoh A. Event-notification-based inactive file search for large-scale file systems. Proc. of the 2012 Digest APMRC. Singapore. 2012. 1–7.

5 Onyisi P. Event-driven messaging for offline data quality monitoring at ATLAS. Proc. of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015). Okinawa, Japan. 2015.

6 Shields I. Monitor Linux file system events with inotify. http://www.ibm.com/developerworks/ linux/library/l-inotify/. [2010-09-10].

7 McCutchan J, Love R, Griffis A. Inotify-get your file system supervised. http://inotify.aiken.cz/.

8 Paris E. Fanotify: The fscking all notification system. https://lwn.net/Articles/339253/. [2009-06-29].

9 J Notify File system events library for Java. http://jnotify.sourceforge.net/linux.html.

10 黄江平. 基于 Lucene 的桌面搜索引擎的研究与应用[硕士学位论文]. 杭州: 浙江理工大学, 2012.

11 ASF. Apache Tika-a content analysis toolkit. http://tika.apache.org/.

12 Mattmann CA, Zitting JL. Tika in Action. Shelter Island, NY: Manning Publications Co., 2011: 28–75.

13 Burgess AB, Mattmann CA. Automatically classifying and interpreting polar datasets with Apache Tika. Proc. of the 15th International Conference on Information Reuse and Integration (IRI). Redwood City, CA, USA. 2014. 863–867.

14 王旭仁, 郑秋辉, 何发镁, 等. 基于 Tika 和 Lucene 的桌面搜索引擎研究与实现. 计算机工程与设计, 2014, 35(1): 310–314.

15 IK Analyzer. http://git.oschina.net/wltea/IK-Analyzer-2012FF.

16 Lucene. http://lucene.apache.org/.

17 McCandless M, Hatcher E, Gospodnetic O. Lucene in Action: Covers Apache Lucene 3.0. 2nd ed. Shelter Island, NY: Manning Publications Co., 2010: 3–4.

18 刘艳, 杨奇龙, 蔡燕冬. FileFinder: 桌面搜索引擎的设计与实现. 计算机工程与设计, 2013, 34(7): 2627–2631.

19 DocFetch. https://sourceforge.net/projects/docfetcher/.

20 Recoll. http://www.lesbonscomptes.com/recoll/.