计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(1):231−235 [doi: 10.15888/j.cnki.csa.007239]

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

基于弹性并发的文件校验模型①

阮晓龙[1], 于冠军[2]

[1](河南中医药大学 信息技术学院, 郑州 450046)
[2](郑州祺石信息技术有限公司, 郑州 450008)
通讯作者: 于冠军, E-mail: yuguanjun@yeework.cn

摘　要: 随着数字资源不断发展, 面对大容量的数字资源校验时, 现有的文件校验方法相比于传统文件校验在执行效率、服务器资源利用率方面已有极大的提高, 但仍存在不灵活、服务器资源浪费、文件校验存在限制等特性. 本文提出一种基于服务器资源弹性并发处理的文件校验算法模型, 利用多并发和分块读取的方式, 在保证文件Hash 计算理论复杂度不变的情况下, 合理利用服务器资源利用率, 提升文件校验效率.

关键词: 文件校验; 文件完整性; 弹性并发; 多线程

引用格式: 阮晓龙,于冠军.基于弹性并发的文件校验模型.计算机系统应用,2020,29(1):231–235. http://www.c-s-a.org.cn/1003-3254/7239.html

File Verification Model Based on Flexible Concurrency

RUAN Xiao-Long[1], YU Guan-Jun[2]

[1](School of Information Technology, Henan University of Chinese Medicine, Zhengzhou 450026, China)
[2](QISHI Corporation, Zhengzhou 450008, China)

Abstract: With the continuous development of digital resources, facing large-capacity digital resource verification, the existing file verification methods have greatly improved compared with the traditional file verification in terms of execution efficiency and utilization of server resources, but there are still some characteristics such as inflexibility, waste of server resources, limitations of file verification, etc. Therefore, this study proposes a new method based on server resource flexibility and concurrency. By using multi-concurrent and block-reading methods, we can rationally utilize the resource utilization of servers to improve the efficiency of file verification under the condition that the theoretical complexity of file Hash calculation remains unchanged.

Key words: hashtab; file integrity; flexible concurrency; multithreading

互联网+时代下, 数字资源的数量和种类呈现出爆发性的增长趋势, 数字资源的存储规模日渐巨大[1]. 数字资源获取过程中的漏洞、通信传播过程中的数据丢失、黑客的恶意攻击等也随之增多, 数字资源的存储普遍存在质量问题[2], 如不能快速验证数字资源的完整性, 将极大的降低资源服务质量, 甚至带来灾难性的后果.

内容服务商早期在进行文件校验时, 通常直接使用 Hash 算法对文件进行计算, 从而判断其内容是否改变. 但随着数字资源的不断增大、网络的快速发展, 其效率低下、校验存在限制的特性越来越突出.

目前针对解决数字资源问题的相关技术手段越来越多, 比如: 文献[3]中 Long Wang 等人针对流媒体服务提出采样验证、本地文件格式检查、特定编码协议语法验证 3 种文件完整性验证方法; 文献[4,5]中针对 P2P 文件提出了延时计算、基于 Merkel 校验的文件验证方法; 文献[6]中方燕飞等人提出基于多层 MD5 消息

① 收稿时间: 2019-06-25; 修改时间: 2019-07-16; 采用时间: 2019-07-19; csa 在线出版时间: 2019-12-27

摘要, 并行计算的文件完整性校验方法; 文献[7,8]中张永棠等人提出基于粒度抽样的文件校验方法, 在舍弃一定准确性将校验安全性能控制在可接受范围内的情况下, 大幅提升文件校验效率; 文献[9]中李昊宇等人提出基于 B+树技术, 在云存储环境下多副本数据的完整性验证方案.

以上的研究在校验效率方面均有有较好的提升, 但并没有与校验准确性、服务器性能做到完全兼顾, 也没有充分考虑到在不同服务器下的实际应用. 针对现有方案存在的未能兼容服务器环境、准确性与效率等问题, 本文提出一种弹性快速的文件校验方法, 对存储的数字资源进行评估, 从而解决数字资源的质量问题, 提升文件校验速率, 保障数字资源服务者的效益.

1 文件校验分析

1.1 分析模型

在介绍本文提出的文件校验模型之前, 首先需要清楚文件校验耗时的组成, 具体如式 (1) 所示.

$$t = t' + t'' \tag{1}$$

对于文件来说, 一次校验耗费时间由文件读取耗费的总时间t'以及文件 Hash 计算耗费的总时间t''组成, 两者综合起来就是决定文件校验耗费时间的主要因素.

在对文件进行 Hash 计算时, CPU 核心处理能力越好、CPU 使用率越高, 计算耗费时间也就越短. 但在进行 Hash 计算的同时要充分考虑到服务器的磁盘 I/O 效率, 一般来说主要通过磁盘 I/O、内存等方式实现文件的读取, 毫无疑问放在内存中进行处理是最优的选择, 然而对于大文件来说, 无法通过内存映射文件的方式将其直接放置, 这时就需要将文件进行切分处理, 从而使并发计算成为可能.

1.2 分布式校验支持

对于分布式存储系统来说, 其存放资源可能是碎片化、重复性的, 通过调配系统将这些资源进行集中调配, 为文件校验处理引擎添加校验任务; 文件校验引擎根据派发的任务, 按照校验模型进行文件校验, 并将结果回写至调配系统, 从而完成文件的校验. 本文主要阐明文件校验的算法模型, 关于分布式多系统调配的业务逻辑在此不再做进一步的介绍. 本文全部实验均是在单台服务器环境下进行, 通过优化单个节点的执行效率, 从而提升整个系统的运行性能.

2 文件完整性校验方法

2.1 文件校验原理

基于上述文件完整性校验方法, 综合其优缺点, 本文提出基于弹性并发的文件校验模型: 对于待校验文件 f, 将其分为 m 个数据块, 对 m 个数据块同时进行多并发切割计算, 然后判断切割后的数据块是否超出限制长度 l, 如果超出则继续进行递归切割计算, 否则直接进行 Hash 计算, 最终得到结果值为 $a1$; 比对 $a1$、文件隐藏属性记录值 $a2$、数据库记录值 $a3$ 三方是否完全相同, 若相同则说明文件正确, 不同则表示该文件已被篡改, 其算法流程如图 1 所示.

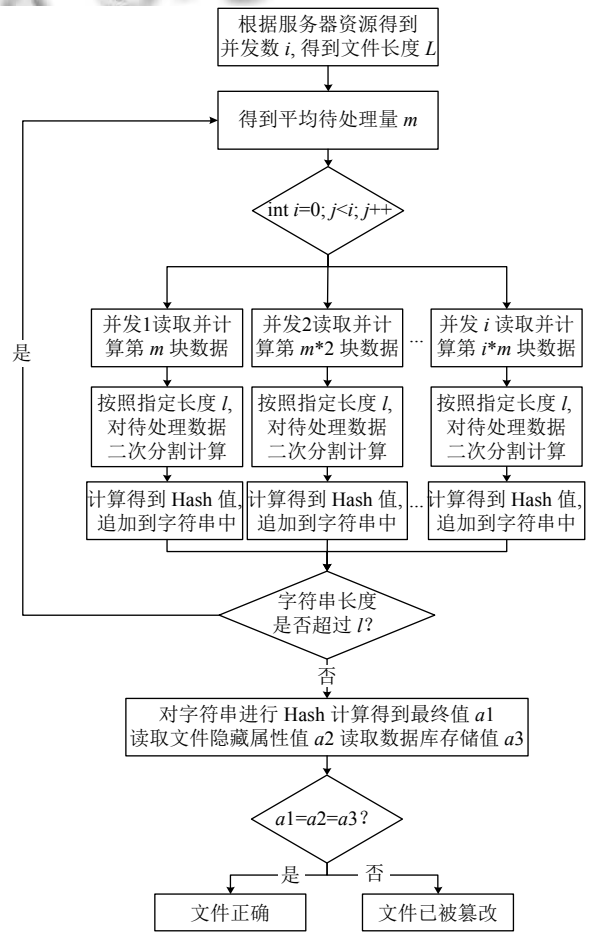

图 1 并发文件校验算法流程示意图

本模型使用弹性并发计算资源文件, 其耗费时间主要由并发读取文件耗费最长时间t'、计算耗费最长时间t''所决定, 其具体耗费时间计算如式 (2)、式 (3) 所示.

$$t' = \sum_{r=1}^{n} tr \tag{2}$$

$$t'' = \sum_{j=1}^{n} tj \tag{3}$$

其中, n 为递归次数, tr 为每次递归切割读取文件块所耗费的最长时间, tj 为每次递归切割计算文件块所耗费的最长时间. 因此, 并发校验文件耗费总时间的计算公式如式 (4) 所示.

$$t = \sum_{r=1}^{n} tr + t'r \tag{4}$$

本文进行文件校验时, 通过指针对待处理文件进行读取, 每次读取指定长度 l, 然后将其转存至内存中进行计算, 计算完毕则自动释放内存, 直至全部计算结束, 因此内存占用会维持在 l*并发数 (以下定义为 i) 左右.

在操作系统中进程是其管理单位, 线程则是进程的管理单位. 不管是在单线程还是多线程中, 每个线程都有一个程序计数器、一组寄存器、堆栈. 进程进行切换的实质就是被中止运行进程与待运行进程上下文的切换, 在进程未占用处理器时, 进程的上下文存储在进程的私有堆栈中[10]. 这就像多个同学要分时间使用同一张实验桌, 所谓要回收正在使用同学的使用权, 就是让其把属于自己的东西拿走; 而赋予某个同学实验桌使用权, 只不过就是让他把自己的东西放到实验桌上罢了[11].

与进程切换不同, 线程的切换开销则小很多, 只需要保存和恢复线程的寄存器内存, 栈的切换也是通过寄存器的切换来完成的. 由此可见线程的切换比进程的切换代价要低很多, 因此本文使用单进程多线程的方式进行文件并发校验.

2.2 最优并发数量

当进程、线程的乘积大于等于设备总 CPU 核心数时, 其处理效率是相对最优的. 这是因为进程需要一些资源 (CPU 使用时间、存储器、I/O 设备等) 才能进行工作, 且为依序逐一进行, 也就是说每个 CPU 核心任何时间仅能运行一项进程. 因此并发数为 CPU 核心数时, 即可达到最优的程序执行效率. 继续增加并发数, 执行效率会持续提升, 多余的并发线程将排队等待被执行. 由于本文读取文件会占用一定的时间, 为了使 CPU 始终处于工作状态, 并发线程数最好是 CPU 核心数的两倍, 最大并发处理量每次应不超过磁盘 I/O 的瓶颈, 同时应充分考虑服务器内存容量等资源.

本文使用 MD5 计算算法以 512 bit 为最小单位, 因此 CPU 核心的每个计算量应为 512 bit 的倍数, 综合考虑服务器磁盘 I/O、计算损耗, 每次切割计算的文件大小不易设置过大, 应根据业务处理实际情况进行设置, 本文将其初始值设定为 10 MB. 设定读取长度为 l (10 MB), CPU 核心数为 c, 内存容量为 m, 读取长度 l、并发数 i 的初始值计算方法如式 (5)、式 (6) 所示.

$$l = 10 \tag{5}$$

$$i = \begin{cases} c*2, m > l*2*c \\ c, c*l \le m \le l*2*c \\ \left\lfloor \frac{m}{l} \right\rfloor, m < c*l \end{cases} \tag{6}$$

当内存容量 m 大于 2*c*l 时, 并发数 i 为 CPU 核心数的两倍; 当 m 大于等于 c*l 且小于等于 2*c*l 时, 并发数 i 为 c; 当 m 小于 c*l 时, 并发数 i 为 m 除以 l 的向下整数商值. 因此在进行并发计算时首先应得到待校验文件大小, 当其大小不超过 10 MB 时直接进行 Hash 计算, 超过时先得到服务器 CPU 核心数量、内存容量, 然后按照公式创建对应数量的并发执行文件校验. 对于 CPU 来说每颗核心都要处理 10 MB 左右大小的数据, 一直递归执行直到全部计算完毕为止.

3 实验结果与分析

3.1 实验环境

本次实验服务器资源配置如表 1 所示.

表 1 服务器配置信息

类型	详细描述
处理器	型号 Intel(R) Core(TM) i5-7500, 主频 3.4 GHZ, 1 处理器 4 核心
内存	8 GB
操作系统	Windows 10 专业版

为排除其它因素对本次实验的影响, 此次实验仅保留原生的操作系统, 不安装任何软件/服务/组件 (测试程序除外), 最大程度的减少实验影响因素.

3.2 实验结果

本文提出的文件校验算法模型可根据服务器环境自动弹性设置参数, 为验证其灵活性及通用性, 本次实验采用不同存储规模的文件进行验证, 一是使用小文件 (1 GB 以下) 进行算法模型验证, 二是使用大文件 (10 GB) 进行验证.

通过实验, 得出 MD5 计算耗时是线性变化的, 文件越大耗费时间越长, 使用传统 MD5 校验与本文校验模型对比测试, 其实验结果如图 2 所示.

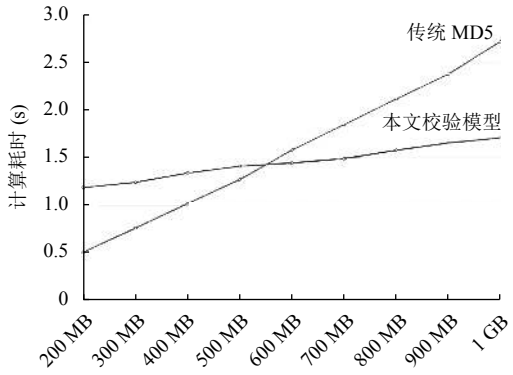

图 2 传统 MD5 与本文校验模型计算耗时对比图

从图 2 可以看出传统 MD5 计算文件大小与耗费时间呈正比关系, 耗费时间随着文件的增大而线性增加, 由于 MD5 自身以及服务器内存等实验环境因素的影响, 当待校验文件超过 1 GB 之后耗费时间递增, 因此以下实验进行对比时使用其理论值做比较, 实验结果如图 3 所示.

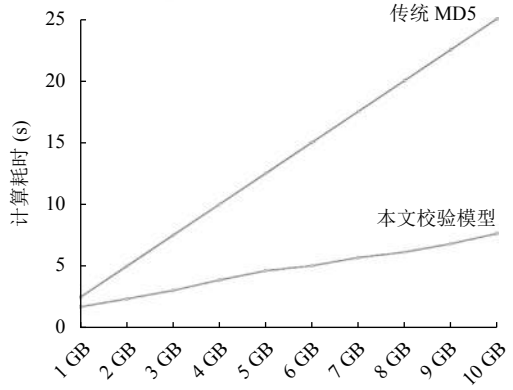

图 3 传统 MD5 理论值与本文校验模型计算耗时对比图

从上述实验结果来看, 当待校验文件大小不超过 600 MB 时, 传统 MD5 校验方式比本文校验模型耗时要短, 这是因为实验时本文校验模型包含了文件切割、内存转存等开销. 当待校验文件超过 600 MB 后, 本文校验模型与传统的 MD5 串行计算对比, 其计算效率明显提高.

下面从服务器资源占用方面对其进行分析, 进行文件校验时本文校验模型服务器资源占用情况如表 2 所示.

从表 2 中可以看出 CPU 使用率逐步增高直至稳定, 内存占用大小从 $c*l$ 左右直至升高到 $c*l*2$ 左右, 其内存占用率很低 CPU 利用率很高, 而待校验文件比较小时 CPU 利用率比较低, 这是由于文件切割耗时较长, 导致了整个校验时间耗时较长.

表 2 本文校验模型服务器资源占用情况

文件规模 (GB)	CPU 使用率 (%)	内存占用大小 (MB)
1	22.4	41.5
2	41.1	54.1
3	60.8	42.9
4	81.4	54.2
5	97.2	85.0
6	97.9	95.1
7	98.8	117.2
8	98.1	126.9
9	97.4	116.8
10	98.1	106.0

在小文件处理上本文校验模型相比于传统 MD5 校验耗时更长, 但在大文件校验中计算效率十分明显. 由于实验环境条件影响, 并无法做到在上万台服务器上运行本文校验模型, 即便如此, 通过在实验室 CPU 内存 2 核 4 GB、4 核 8 GB、8 核 16 GB、16 核 16 GB 等几种当下常见服务器配置上的运行, 也能够很好的印证上面的结论.

3.3 实验分析

对于文件校验来说其优劣主要由结果的正确性、时间成本、适应性、可移植性、鲁棒性这 5 个方面决定, 通过这些判断条件能够很好的评估文件校验算法模型是否准确高效.

本文提出的校验模型基于 MD5 算法进行优化, 其本质相当于对待校验文件进行了一次完整的 MD5 计算, 得到最终结果与事实复杂度完全相符. 在保持计算复杂度不变的前提下, 使用并发计算能够大大节约时间成本. 通过自适应弹性并发, 可以变相调节服务器磁盘 I/O 占用时间以及 CPU 核心占用时间, 其几乎能适应所有规模的文件校验以及各样服务器环境. 通过采用小批量循环计算、自适应参数的方式, 确保了服务器资源占用始终处在正常的范围内, 不会因磁盘 I/O 占用时间过长或者 CPU 占用时间过长导致系统出现宕机或假死情况.

为更好的分析本文校验模型, 将其与传统 MD5、普通多线程、抽样校验进行对比, 对比结果如表 3 所示.

本文校验模型相对于抽样校验, 其执行效率并不算高, 但其准确性更高, 可将其准确率等同于传统 MD5 校验准确率. 相对于普通多线程校验其最大的特

点就是灵活适配服务器环境, 有效提升文件校验执行效率的同时, 兼容各种不同容量规模的数字资源. 无论是流式碎片还是虚拟机导出的超大服务器备份文件, 即便是在服务器资源配置十分低下的情况下, 也能保证高效的执行效率, 降低服务器故障、业务无法正常运行的风险.

表 3 不同文件校验模型对比

	传统 MD5	普通多线程	本文校验模型	抽样校验模型
执行耗时	较长	较短	较短	极短
资源耗用	较低	较高	较高	较高
超大文件处理	耗时长, 执行程序有崩溃的风险	并发数需手动设置, 校验存在瓶颈	并发数弹性适应, 兼容执行效率	校验快速
准确性	较高	较高	较高	较低

综合以上实验及分析, 可以认为本文校验模型自适应能力强、能够有效的提升执行效率.

4 结论与展望

本文提出的一种基于弹性并发的高效文件校验算法模型, 综合服务器磁盘 I/O 效率、服务器 CPU 处理能力, 在将服务器资源占用控制在正常范围内的同时, 充分利用 CPU 核心计算资源提高校验效率, 对于存储的资源文件能够快速进行校验, 确保其完整性. 同时对于超大文件, 比如 1 TB 大小的文件, 利用小批量循环切割、读取、计算的方式极大的降低磁盘 I/O 和内存的占用.

本文校验模型可适应于各种文件校验场景, 假如对文件校验效率要求比较高, 则可以使用抽样校验+本文校验模型, 在舍弃一定准确性的前提下, 对抽样文件弹性并发校验, 更进一步地提升其计算效率; 如果对文件准确性要求比较高,则可以使用本文校验模型+MD5 结构改造, 由于MD5算法本身就具备一定的碰撞特性[12], 通过对 MD 结构基础上的改造, 在牺牲一部分处理时间的前提下, 能够有效规避 MD5 碰撞问题, 保障文件校验的准确性.

参考文献

1 McKinsey Global Institute. Big data: The next frontier for innovation, competition, and productivity. McKinsey & Company, 2011. https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/big-data-the-next-frontier-for-innovation.

2 郭志懋, 周傲英. 数据质量和数据清洗研究综述. 软件学报, 2002, 13(11): 2076–2082.

3 Wang L, Wang BH, Xie S. The research and application of integrity technology in acquisition of streaming media search engine. Information Engineering Research Institute, USA. Proceedings of 2012 2nd International Conference on Smart Materials and Nanotechnology in Engineering (SMNE 2012). 2012. 561–566.

4 李添杰, 刘述, 高强. 基于 Merkle 树的 P2P 流媒体内容完整性校验. 计算机工程与设计, 2015, 36(7): 1712–1715, 1731.

5 贺鹏程, 王劲林, 邓浩江, 等. P2P 文件完整性校验延迟隐藏算法. 计算机工程, 2010, 36(15): 29–31. [doi: 10.3969/j.issn.1000-3428.2010.15.011]

6 方燕飞, 王俊, 何王全. 基于多层 MD5 消息摘要的文件完整性实时检测技术. 计算机应用与软件, 2015, 32(1): 20–23. [doi: 10.3969/j.issn.1000-386x.2015.01.006]

7 张永棠, 丑佳文. 基于粒度抽取的 ELF 文件完整性校验方法. 信息网络安全, 2016, (9): 134–138. [doi: 10.3969/j.issn.1671-1122.2016.09.027]

8 于美丽. 云存储数据完整性校验中数据抽样算法的研究[硕士学位论文]. 上海: 东华大学, 2015.

9 李昊宇, 张龙军, 李庆鹏. 云存储环境的多副本数据完整性验证方案. 中国科技论文, 2017, 12(14): 1659–1663. [doi: 10.3969/j.issn.2095-2783.2017.14.017]

10 Tanenbaum AS, Bos H. 现代操作系统. 北京: 机械工业出版社, 2017.

11 周晓慧, 王建中, 李强, 等. 现代操作系统中的进程、线程及在 Windows OS 下的应用. 计算机应用研究, 2002, 19(2): 123–125. [doi: 10.3969/j.issn.1001-3695.2002.02.042]

12 毛熠, 陈娜. MD5 算法的研究与改进. 计算机工程, 2012, 38(24): 111–114, 118. [doi: 10.3969/j.issn.1000-3428.2012.24.027]