Masaryk University
Faculty of Informatics

The effect of aging on filesystems performance

Bachelor's Thesis

Samuel Petrovic

Brno, Spring 2016

Replace this page with a copy of the official signed thesis assignment and the copy of the Statement of an Author.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Samuel Petrovic

Advisor: Adam Rambousek

Acknowledgement

This is the acknowledgement for my thesis, which can span multiple paragraphs.

Abstract

This is the abstract of my thesis, which can span multiple paragraphs.

Keywords

filesystem, xfs, IO operation, aging, fragmentation ...

Contents

List of Tables

List of Figures

1 Introduction

Theses are rumoured to be the capstones of education, so I decided to write one of my own. If all goes well, I will soon have a diploma under my belt. Wish me luck! Possible other chapters to explain terms: journaling filesystem, allocation groups, B+ trees

2 Targeted filesystems

2.1 XFS

XFS is a 64-bit journaling file system created by Silicon Graphics, Inc(SGI) in 1993. It is known for excellence in execution of paralel I/O operations, because of its architecture based on allocation groups.

Allocation groups are euqally sized linear regions within file system. Each allocation group manages its own inodes and free space, therefore increasing parallelism. Architecture of this design enables for significant scalability of bandwidth, threads, and size of filesystem, as well as its files, simply because multiple processes and threads can access the file system simultaneously.

XFS allocates space as extents stored in pairs of B+ trees, each pair for each allocation group (imrpoving performance especially when handling large files). One of the B+ trees is indexed by the length of the free extents, while the other is indexed by the starting block of the free extents. This dual indexing scheme allows for the highly efficient location of free extents for file system operations. Extent describes one or more contiguous blocks, which can considerably shorten list of blocks.

Metadata journaling ensures consistency of data in case of emergency situations as f.e. system crash.

Prevention of file system fragmentation consist mainly of *delayed allocation* feature as well as online defragmentation(*xfs_fsr*), that can.

Delayed allocation, also called *allocate-on-flush* is a feature that, when a file is written to the buffer cache, substracts space from the free-space counter, but won't allocate the free-space bitmap. The data is held in memory, instead, until it must be flushed (to storage) because of memory pressure when calling Unix *sync*, or when flushing dirty buffers. This approach improves the chance, that the file will be written in a contiguous group of blocks, avoiding fragmentation and reducing CPU usage as well.

Pridat nieco na zaver o XFS?

2.2 EXT4

Ext4, also called *fourth extended filesystem* is a 48-bit journaling file system, developed as successor of ext3 for linux kernel, improving reliability and performance features.

Traditionally, ext* systems use an indirect block mapping sheme. Such an approach is generaly inefficient for large files, on operations like deleting or truncating. Ext4 use modern approach of *extents*, which positively affect performance and encourage continuous layouts.

When allocating, ext4 use multiblock allocation, which is more efficient way than one block allocation at time, which is present in earlier ext* file systems. Multiblock allocation has far better performance, particulary when in use with delayed allocation and extents.

Similary as xfs, ext4 use delayed allocation design, to increase performance, especially when in use with multiblock allocation and extent-based approach, also reducing fragmentation on the device. For cases of fragmentation that still occur, ext4 provide support for online defragmentation and *e4defrag* tool to defragment either single file, or whole filesystem.

3 Journaling file systems

4 Methodology

4.1 My aging script

My aging tool is a simple approach to write and remove many files of random sized files.

The tool consist of three scripts and one common library called *functions*. The scripts are named *filesystem_ager.py*, *fio_config_generator.py* and *random_deletor.py*.

The workflow consist of calling filesystem_ager, with desired parameters. Script manages triggering fio_config_generator, calling fio tool on generated config and triggering random deletor. These three actions are repeated given number of times. Parameters of filesystem_ager are:

1. Total desired size do be written in one cycle
2. Denominator of total desired size (Total desired size will be divided by this number)
3. Range of size of written files
4. Number of cycles

Although FIO tool has some parameters to randomize the size of files which are written, the management of file sizes and randomisation, as well as naming of files is handled by fio_config_generator instead, to provide more control over those qualities. Parameters of this script are:

1. Total deisred size to be written
2. Range of size of written files

The script will generate global settings of a workload, then proceeds to generate jobs for every file that will be written. File size is always the name of that file, and these are gathered to a list, then list of generated files is returned and script ends. Including file size in its name, as well as indexation of files will help effectively search and delete files in the random deletion process, without need to search for files on the disk and examine them for size. Simplistic approach in fio config will hopefully result in compatibility and reliability in use with any fio version.

After config file is generated, filesystem_ager will run fio tool on generated config and therefore, files are written on the device.

The removing of files is handled by random_deletor script. Its parameters are:

1. Total written size
2. Denominator of total size
3. Range of size of written files
4. Number of existent files

If denominator equals zero, random_deletor wont remove any files and will return empty list. Otherwise, desired range of deletion is estimated. Random_deletor then proceeds to remove files while desired volume is not deleted. Files are randomly selected through choosing random integer from zero to number of existent files. This step may seem inefficient, but with large amounts of generated files, the time to perform succesfull selection will not change dramatically. Selected file name is then parsed for size information, and if it fits into desired volume to remove, it is deleted, through subprocess command. Names of removed files are gathered in a list and returned.

Number of deleted files is substracted from number of existent files. filesystem_ager then sums up deleted volume, log it as well as other information and triggers the cycle again.

Here is a sequence diagram to show the structure of filesystem_ager.

4.2 Aging using fs-drift

fs-drift is a workload aging test written by Ben England. It relies on randomly mixed requests generated according to options. These requests can be writes, reads, creates, appends or deletes.

At the beginning of run time, the top directory is empty, and therefore *create* requests success the most, other requests, such as *read* or *delete*, will fail because of lack of files and small probability of randomly choosing existing one.

Over time, as the filesystem grows, *create* requests began to fail and other requests succede more. Finally, filesystem will reach a state of equilibrium, when requests are equaly likely to execute. From this point, the filesystem will not grow anymore, and the test will run until one of the *STOP* conditions are met (specified with parameters).

4.3 fs-drift_matrix

To determine which fs-drift settings will be the most fitting for purposes of this thesis, I wrote a small python script fs-drift_matrix.

It is capable of taking matrix of possible fs-drift paramteres from *json* file and then run it on a device.

After every run, histograms-generating scripts are triggered to store histograms of *free space* and *used space* fragmentation. Also outputs of the *fs-drift* script and *df* command are logged.

As the creator states in README, to fill up a filesystem, maximum number of files and mean size of file should be defined such that the product is greater than the available space.

For the purpose of this thesis, desired usage is 60%-100% with enough fragmentation to consider the device aged.

5 Age measurement

For determining some overall idea about an extent to which is the filesystem aged or dirty, I wrote scripts that generate histograms representing fragmentation of used space as well as fragmentation of free space. Both scripts use common linux tools and pyplot to generate the graphics. Both scripts can display linear or logarhytmic Y scale.

5.1 extent distribution

Script extent_distribution.py makes use of xfs_io fiemap tool, which is a tool to display extent distribution of a given file and works correctly even for ext* filesystems.

The script will first recursively crawls the whole filesystem from given top folder and makes a list of all files. Fiemap is then run over every file separately.

The only data, that are then parsed from the output, is how many non-contigous extents does the file have. These integers are aggregated to a single list, from which are then counted, and final histogram is made.

5.2 free space fragmentation

Script free_space_fragmentation.py use the tool e2freefrag, which runs over a device, and outputs the histogram of free space fragmentation in texutal form. Script will store this output and then easily parse the histogram and aggregate the data into a graphic form.

6 Measuring performance

6.1 recipe_fio

Measuring a performance is done by an internal tool I developed, recipe_fio. Similar to filesytem_ager, recipe_fio use fio tool to handle needed IO operations, but instead of focusing on filling the filesystem, the script use measurement features of fio, which consist of performing IO operations and reporting results.

The main script receives slightly enhanced fio config, enriched of some non-fio parameters, which are used by test only. These parameters are:

1. used filesystem
2. number of test repetitions. For statistical stability, we run the test several times under same conditions.
3. flag which represents whether or not to rsync data to some result-storing server

After compiling, tool parses the parameters and gathers information about system, which consist of: version of kernel, time and date, hostname, RHEL compose, memory, kernel, mount, system info, system variables and fio version.

Then it proceeds to set environment for testing by:

1. Installing fio tool
2. Creating directory for results

Python script run_tests.py, manages to parse recipe parameter, resulting in creating one or several fio configs in the directory, further adding logging parameters to the configs. Then for every created config, directory on the desired medium is created and fio tool is triggered. If the script succesfully ends, file OK is created in the results directory.

When the testing is over, bash script generate the name of results, which consist of:

1. time
2. date
3. used filesystem
4. version of kernel
5. version of RHEL compose

Then proceeds to tar the result directory into a tar file with generated name, and according to argument will or will not rsync the result onto the data server.

6.2 Measuring management

Measuring performance of aged filesystem happen in a loop as follows:

1. inspecting device, logging
2. randomly removing volume needed for test to execute
3. running test
4. cleanup

Inspection of device is just generating histograms of extent distribution and free space fragmentation.

Volume is randomly removed using random_delete_volume.py script. This scriptt globs all files in the filesystem, retrieves information about used volume as well as it's overall size. Then proceeds to randomly choosing files to delete and stops when desired volume is freed. The approach of recursively globbing all files may be inefficient, but this way, we can be sure, that volume is deleted from whole device evenly.

Generaly, in our measurements, we focus on iterating through different thead count, while keeping size and blocksize the same. I also test for all 4

6.3 Generating results

The output of result generator is a htlm report summarising all information about system, links to raw data and charts of measured values. *talk about highcharts and how do you represent data

7 Floats and references

The logo of the Masaryk University is shown in Figure 7.1 and Figure 7.2 at pages 15 and 16. The weather forecast is shown in Table 7.1 at page 16. The following chapter is Chapter 8 and starts at page 17. Items 3, 3b, and 3(c)iv are starred in the following list:

1. some text
2. some other text
3. ⋆
   (a) some text
   (b) ⋆
   (c) some other text
      i. some text
      ii. some other text
      iii. yet another piece of text
      iv. ⋆
   (d) yet another piece of text
4. yet another piece of text

If your reference points to a place that has not yet been typeset, the `\ref` command will expand to **??** during the first run of `pdflatex thesis.tex` and a second run is going to be needed for the references to resolve. With online services – such as Overleaf – this is performed automatically.

Figure 7.1: The logo of the Masaryk University at 40 mm

Figure 7.2: The logo of the Masaryk University at $\frac{2}{3}$ and $\frac{1}{3}$ of text width

Day	Min Temp	Max Temp	Summary
Monday	13°C	21°C	A clear day with low wind and no adverse current advisories.
Tuesday	11°C	17°C	A trough of low pressure will come from the northwest.
Wednesday	10°C	21°C	Rain will spread to all parts during the morning.

Table 7.1: A weather forecast

8 Mathematical equations

TEX comes pre-packed with the ability to typeset inline equations, such as $e^{ix} = \cos x + i \sin x$, and display equations, such as

$$\mathbf{A}^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

LATEX defines the automatically numbered `equation` environment:

$$\gamma Px = PAx = PAP^{-1}Px. \tag{8.1}$$

The package amsmath provides several additional environments that can be used to typeset complex equations:

1. An equation can be spread over multiple lines using the `multline` environment:

$$\begin{aligned} a + b + c + d + e + f + b + c + d + e + f + b + c + d + e + f \\ + f + g + h + i + j + k + l + m + n + o + p + q \end{aligned} \tag{8.2}$$

2. Several aligned equations can be typeset using the `align` environment:

$$a + b = c + d \tag{8.3}$$
$$u = v + w + x \tag{8.4}$$
$$i + j + k + l = m \tag{8.5}$$

3. The `alignat` environment is similar to `align`, but it doesn't insert horizontal spaces between the individual columns:

$$a + b + c + d = 0 \tag{8.6}$$
$$e + f + g = 5 \tag{8.7}$$

4. Much like chapter, sections, tables, figures, or list items, equations – such as (8.8) and (My equation) – can also be labeled and referenced:

$$b_{11}x_1 + b_{12}x_2 + b_{13}x_3 = y_1, \tag{8.8}$$
$$b_{21}x_1 + b_{22}x_2 + b_{24}x_4 = y_2. \tag{My equation}$$

5. The `gather` environment makes it possible to typeset several equations without any alignment:

$$\psi = \psi\psi, \tag{8.9}$$
$$\eta = \eta\eta\eta\eta\eta, \tag{8.10}$$
$$\theta = \theta. \tag{8.11}$$

6. Several cases can be typeset using the `cases` environment:

$$|y| = \begin{cases} y & \text{if } z \geq 0, \\ -y & \text{otherwise.} \end{cases} \tag{8.12}$$

For the complete list of environments and commands, consult the amsmath package manual[1].

1. See `http://mirrors.ctan.org/macros/latex/required/amslatex/math/amsldoc.pdf`. The `\url` command is provided by the package url.

9 We have several fonts *at* **disposal**

The serified roman font is used for the main body of the text. *Italics are typically used to denote emphasis or quotations.* `The teletype font is typically used for source code listings.` The **bold**, SMALL-CAPS and sans-serif variants of the base roman font can be used to denote specific types of information.

We can also change the font size, although it is usually not necessary.

A wide variety of mathematical fonts is also available, such as:

$$\mathrm{ABC}, \mathcal{ABC}, \mathbf{ABC}, \mathsf{ABC}, \mathit{ABC}, \mathtt{ABC}$$

By loading the amsfonts packages, several additional fonts will become available:

$$\mathfrak{ABC}, \mathbb{ABC}$$

Many other mathematical fonts are available[1].

1. See `http://tex.stackexchange.com/a/58124/70941`.

10 Inserting the bibliography

After loading the `biblatex` package and linking a bibliography database file to the document using the `\addbibresource` command, you can start citing the entries. This is just dummy text [**inbook-full**] lightly sprinkled with citations [**incollection-full**]. Several sources can be cited at once [**whole-collection**, **manual-minimal**, **manual-full**]. **inbook-full** was written by **inbook-full** in **inbook-full** We can also produce **inbook-full** or (**inbook-full**, **inbook-full**). The full bibliographic citation is: **inbook-full** . We can easily insert a bibliographic citation into the footnote[1].

The `\nocite` command will not generate any output, but it will insert its argument into the bibliography. The `\nocite{*}` command will insert all the records in the bibliography database file into the bibliography. Try uncommenting the command and watch the bibliography section come apart at the seams.

When typesetting the document for the first time, citing a `work` will expand to [**work**] and the `\printbibliography` command will produce no output. It is now necessary to generate the bibliography by running `biber thesis.bcf` from the command line and then by typesetting the document again twice. During the first run, the bibliography section and the citations will be typeset, and in the second run, the bibliography section will appear in the table of contents.

The `biber` command needs to be executed from within the directory, where the LaTeX source file is located. In Windows, the command line can be opened in a directory by holding down the Shift key and by clicking the right mouse button while hovering the cursor over a directory. Select the Open Command Window Here option in the context menu that opens shortly afterwards.

With online services – such as Overleaf – all commands are executed automatically.

1. **inbook-full**.

11 Inserting the index

After using the `\makeindex` macro and loading the `makeidx` package that provides additional indexing commands, index entries can be created by issuing the `\index` command. It is possible to create ranged index entries, which will encompass a span of text. To insert complex typographic material – such as α or TEX – into the index, you need to specify a text string, which will determine how the entry will be sorted. It is also possible to create hierarchal entries.

After typesetting the document, it is necessary to generate the index by running

`texindy -I latex -C utf8 -L` ⟨*locale*⟩ `thesis.idx`

from the command line, where ⟨*locale*⟩ corresponds to the main locale of your thesis – such as `english`, and then typesetting the document again.

The `texindy` command needs to be executed from within the directory, where the LATEX source file is located. In Windows, the command line can be opened in a directory by holding down the Shift key and by clicking the right mouse button while hovering the cursor over a directory. Select the Open Command Window Here option in the context menu that opens shortly afterwards.

With online services – such as Overleaf – the commands are executed automatically, although the locale may be erroneously detected, or the `makeindex` tool (which is only able to sort entries that contain digits and letters of the English alphabet) may be used instead of `texindy`. In either case, the index will be ill-sorted.

A An appendix

Here you can insert the appendices of your thesis.