File system resilience against power failures

Shaozhi Ye
yeshao@google.com

Main purpose

Evaluate the reliability and integrity of Ext2 and Ext4 against power failures.

Overview

Ext4 has a lot of performance advantage over Ext2 and we are planning to make it an official file system for Google. Lots of performance evaluation has been performed here but before we make the final move, we need to examine some worst case scenarios. This project compares the reliability and integrity of ext2 and ext4 against power failures, which happen to Google servers and cause various problems.

Problem

In short, we want to evaluate the reliability and integrity of our current Ext2 in production and Ext4 file system with different configurations when power failure occurs. More specifically, the first question we want to answer is how much data will be lost with these two file systems. The *data* here include both data (file content) and meta data. The second question is that how much data can be recovered and how long the recovery process (fsck) will take.

Goal

- Understand how reliable Ext4 is against power failures.
- Provide insights for developing reliable applications on Ext4 file systems.
- Find related bugs in current Ext4 and come up with possible patches.
- Investigate how flash devices perform against power failures.

Implementation

Test environment setup

One or several clients talks to a single server via TCP or UDP. The server acknowledges after it receives/commits the data to disks. We cut off the power supply for the server during the conversation and check its file system status after reboot. We need to compare the data on the server side with what the clients believe the server has. In the case of data, we just need to check the file content, while in the case of metadata, we need to compare the directory structure.

File traffic generator

We are planning to write our own simple benchmark tool but absolutely welcome any suggestions for available tools. The requirements for this tool include:

- A client sends test configurations and data to the server.
- A server takes configurations and performs the IO operations desired by the client. The server informs the client about its IO status.
- The configuration includes IO size, acknowledgement interval setting, IO options (fsync, o_direct, fallocate, mmap, etc).
- IO operations include file write/create/delete and directory options.
- It may need to support both TCP and UDP.

Measure

The *data* in this section may refer to file content, meta data, blocks, or inodes.

- Data loss: The client thinks the server has A while the server does not.
- Data error: The client thinks the server has A while the server has B.
- Ordering error: The client commits A before B while the server does not follow this order or has B without A's presence.
- Recovery (Fsck) time: We want to find out the worst case scenario for Ext4 systems and evaluate its fsck time. We are also

interested in the error messages provided by fsck.

Ext2 vs. Ext4

- Data: A large file is sent (creation/update) to the server, the server acknowledges on receiving/committing a certain amount blocks/size.
- Meta data: A large amount of files/directories are created/moved /copied/deleted on the server upon the client's request. The server acknowledges upon completion of each operation.
- Evaluate the application layer assurance:
    - fsync
    - O_Direct
    - Write barrier
    - Acknowledgement interval
- Evaluate the file system layer assurance: Journaling vs non-journaling.
    - write-back: does not journal data
    - ordered: journals metadata and writes data before metadata
    - journal: journals both data and metadata

Fsck

xiangw has performed a set of experiments to show the improvement of Ext4 over Ext2. In short, Ext4 reduces the fsck time *a lot*. Here we want to evaluate the worst case scenario for Ext4. We will also analyze the error messages given by Fsck to see what kind of errors will be caught by fsck.

Mission 2: Disk vs. Flash

We want to know if there is anything special with flash devices comparing to disks.

Milestone schedule

- Benchmark tool develop: 3 weeks. We may add new options during the testing period.
- Testing on disks: 5 weeks. We need to setup test environment,

learn how to measure fsck time via serial console, test various configurations which will be the most time consuming part with the number of options we need to evaluate, make graphs and analysis, find bugs and provide fixes.

- Testing on elephants: 2 weeks. With our previous experiences on disks, this part will be quick.