GIT

Reachability bitmap performance improvement

Name - Shubham Mishra

Email - shivam828787@gmail.com

IRC Nick - shubham828

Location - Ghaziabad, Uttar Pradesh , India , UTC+5:30

Proposal Title - Reachability bitmap performance improvement.

About Me

I am Shubham, I am currently working as a Software Engineer at Microsoft India. I am a 2021 graduate from Delhi University. I am passionate about core engineering and backend technologies. I love to see beyond all abstractions and how things really work under the hood. So, I can work from their roots and make things better. I feel engineering is all about the tradeoffs that we make and I am trying to learn them to become a better Engineer.

I am passionate about open source technologies and have quite a good amount of contribution to them, I participated in **GSoC 2020** with KDE, did Internship with Linux Foundation - HDV, Season of KDE - report, and I am doing voluntary contributions to VSperf, BoostC++ and some other open-source projects.

Motivation for Proposal

I have been using Git for the last 3 years now, and I always find myself curious about how it manages a lot of files seamlessly to make developer collaboration so smooth.

I was listening to one of Derrick Stolee's podcasts, where I felt git contributors really do cool stuff, So I also wanted to be one. By becoming a regular contributor of git, I can give my contribution back to it as well as I can show off it to my friends :) . I love to study advanced data structures and Algorithms that's the reason I chose a bitmap related project. I will get a chance to learn different compression algorithms and analyze their performance.

Project Abstract

During repository clones, the Git server needs to find out all the objects which clients do not have and need to be sent to the client.
To make the process faster, Git uses bitmaps to quickly find all the related objects from an object. Bitmap approach is a performance optimization over the legacy "Counting Objects" - the process in which the git server used to iterate through the graph from branch tips to the beginning of history to list down all objects that need to be sent.

bitmap made reachability faster but uncompressed bitmaps can cost a lot of extra storage. Git uses a C ported version of EWAHBoolArray to compress bitmaps which get stored in the ".bitmap" file with the same prefix "sha" as ".pack" and ".idx".

The aim of the project is to design a performance test suite as well as do the necessary changes to improve bitmap performance by trying out a new compression scheme that can make read operations along with other common operations like intersect, union and negate faster.

Me & Git:

Microproject:

I worked on the microproject "Avoid pipes in git related commands in test scripts", the patches for it has been merged to master now

- https://public-inbox.org/git/20220224054720.23996-3-shivam828787@gmail.com
- https://public-inbox.org/git/20220224054720.23996-3-shivam828787@gmail.co/

I run a pattern matching grep to find all git commands on LHS of pipes and fix all of them from file t001-t050.

As an outcome of this process, I got to learn the code review process at git work, which is quite cool and different from other organization's I contributed to before.

I learned about building source code, running tests, using email to send patches, communicating with reviewers and sending the next patch version process.

Current understanding:

- I have gone through git internals, and I well understood about the pack files as well as the difference between git objects (tree, blob, commit).
- I have gone through some documentations - "MyFirstObjectWalk", etc. it was a good hands-on to get some glimpse of general object related tasks.
- I understand how bitmap works in general, I have got some idea how EWAH compression works and also I have gone through the research paper on roaring run.
- I played with commands of pack-object - "git pack-objects dir --progress < obj_lists.txt" and read the code of related files "pack-bitmap.c" and parts of "pack-object.c"
- I checked the general documentation of Croaring as one of the potential alternatives to EWAH.

Execution plan:

I am interested in keeping my primary focus on "building a performance suite and improving bitmaps performance by finding a better compression scheme" project and if I finish this early or even after the GSoC timeline, I will be happy to contribute to other tasks too.

From the idea page, I got some sense that decompressing a bitmap for reading bits or doing operations like intersection, negation and union makes them slow and can be improved.

Roaring + Run was the suggested alternative to explore. It divides the data into chunks of 2^16, which allows you to check for the presence of any one value faster. As a result, Roaring can compute many operations much faster than run-length-encoded formats like WAH, EWAH, Concise. After getting a high level understanding of algorithms, I explored a bit of the Croaring library which is a C implementation of roaring bitmap. It provides a lot of useful functions to do all general operations (find cardinality, and, or, copy, equals). Which I think we will be using in the "pack-bitmap.c" and "pack-bitmap-write.c" files as a replacement of ewah/bitmap functions. I do not have enough knowledge yet to figure out how compatible croaring is with the current .bitmap format. We might need to make changes in the current .bitmap format accordingly.

Steps I will be following to accomplish the task-

1. Get a better understanding of bitmap related functionalities/ codebase, EWAH, techniques.
2. Build a performance suite to validate our assumption on using different compression schemes can make bitmap faster. We can create benchmarks for reading (decompression), writing (compression) and memory to know if new techniques are really useful or not.
3. I will investigate and find out if we need a new .bitmap format, keeping in mind some of the future projects related to bitmap.
4. Try to build out an initial draft version implementing only minimal required core changes, I will try to get a review on it from a wider audience (including mentors).
5. Make changes according to the comment and repeat the review process.
6. If performance improves, I will be writing the rest of the required code changes to use the Croaring including perf tests for them.
7. Until this time, I will also get a good understanding of the bitmap related projects, so if we will be able to make good progress on roaring+run. I can start picking other subprojects too like 'table of contents' for the .bitmap file where past work - https://lore.kernel.org/git/YNuiM8TR5evSeNsN@nand.local/ can act as a good reference to me or/and 'append-only bitmap generation' subproject.

I feel for any mentoring program, Communication is the key to success. I will be utilizing the community bonding periods to figure out a process for being in regular touch with mentors, which is essential to make sure I am going in the right direction.

Timeline

I am available to dedicate around 30-35 hours every week to the project.

Community bonding periods -

1. Exploring code base mostly related to bitmaps - pack-bitmap-writer.c, midx.c, pack-objects.c.
2. research on other bitmap compression techniques
3. reading technical documents
4. interacting with mentors to understand them and project in more detail.

12 June - 25 July:

1. Write performance test suite for bitmap
2. Finding out if we need a new a .bitmap format, it might adjust rest of the schedule accordingly.
3. Writing the first version with the new compression technique

25 July - 4 sept

1. Get the initial version reviewed by reviewers and make changes accordingly.
2. if tests result well, extending the above functionality to completely move to a new technique.
3. start picking up other tasks if time left

Sept 5 - Sept 12

1. I will make sure to get all changes merged before this week including tests
2. if not, make a decision with mentor on extending the project

Other proposals

No, This is the only proposal I am making for GSoC 2022.

Blog

I want to make blog writing a habit so I planned to publish biweekly blogs at https://shubham828.github.io/ during this GSoC and after that. This is something I started during my last GSoC too but unfortunately couldn't continue post-gsoc. This GSoC gives me another opportunity to become a regular blogger.

After GSoC

I would love to be a regular contributor of Git. After GSoC, I can pick any left out subprojects of bitmap reachability and I would also be happy to extend my knowledge beyond bitmaps to learn and contribute to other parts of git too.