More Sparse Index Integrations

Personal Info

Full name: Shaoxuan Yuan

E-mail: shaoxuan.yuan02@gmail.com
Tel: (+86)189-9835-1089
Alternative Tel: (+1)949-981-8651

Education: University of California, Irvine
Year: Rising Sophomore (currently in a gap year between Freshman and Sophomore)
Major: Computer Science and Engineering

GitHub: https://github.com/ffyuanda
Website: https://ffyuanda.github.io

Before GSoC

Synopsis

I'm picking the project idea "More Sparse Index Integrations" from the SoC 2022 Ideas page. This idea aims to integrate the experimental "sparse-index" feature and "sparse-checkout" command with existing Git commands. Its difficulty should be medium, and the expected project size takes somewhere between 175 hours to 350 hours.

"sparse-checkout" command is still experimental, it allows user to "restrict their working directory to only the files they care about", this is especially useful for the users who only "need to modify a small fraction of the files available" (Bring your monorepo down to size with sparse-checkout, Stolee).

"sparse-index" is a feature that scales down your working directory's index (a key structure on how Git stores everything, a stored version of your working tree) to work with "sparse-checkout".

If "sparse-checkout" is something that makes your working directory "looks" small, then "sparse-index" is something even better: it actually makes your working directory "feels" small, everything e.g. "git-status" or "git-add" is literally faster.

Since the “sparse-checkout” and “sparse-index” may potentially influence other Git commands’ logics and the internal data structure of Git, some work needs to be done to optimize the compatibility and user experience. And that is what my selected idea proposed to do.

Benefits to Community

By joining the community and working on this idea, I can work together with my mentors and community fellows to bring a better user experience to people who are working on large monorepo with “sparse-index” and “sparse-checkout”. Moreover, I have a strong intention to stick around after GSoC, not only keep contributing to the community but also share my experience with or even mentor future potential newcomers.

Microproject

Modernize a test script
Status: merged into master
Description: Some scripts in the Git test suite have outdated styles: misused quotation marks, indentation, etc. Modernize the style of a test script.
Remarks: Although this patch is sort of a “noise patch”, that it makes change for the sake of making change, it serves as an effective intro lesson to familiarize me with the community’s working flow.

Other Patches

Other than the required microproject, I’ve submitted a few other patches when I stumbled upon bugs/potential modifications, these patches are:

t0001: replace "test [-d|-f]" with test_path_is_* functions
Status*:* merged into master
Description*:* There are preferred wrapper functions written in Git’s test library around shell commands like “test -d” or “test -f”. Replace the naked “test [-d|-f]” usages in a t0001 test script with these wrapper functions.

builtin/diff.c: fix "git-diff" usage string typo
Status: merged into master
Description: A simple typo fix. Discovered when I was reading “git-diff” usage.

After finishing the patches above and getting more familiar with the codebase, I started to make tentative patches on the proposed idea:

An RFC patch about “sparse-index” integration with “mv”
Status: suspended
Description: This is a tentative patch on my selected idea, trying to integrate “sparse-index” with “mv” command.
Remarks: During the development and discussion with the community, we found that before integrating with “sparse-index”, there are still unsettled issues with “sparse-checkout”. Since the integration with “sparse-checkout” serves as a prerequisite to enable “sparse-index”, we have to suspend this patch now and start to optimize the “sparse-checkout” integration with “mv”.

A WIP patch to optimize user experience when using “mv” with “sparse-checkout”
Status: WIP
Description: This patch modifies the existing “mv” command’s logic to work better with “sparse-checkout”, especially under “cone mode”.
Remarks: This patch gives me much more background knowledge about “sparse-checkout”, e.g. its APIs and a detailed demonstration of how it works. This patch is currently WIP.

Documentation/git-sparse-checkout.txt: add an OPTIONS section
Status: WIP
Description: This patch modifies the “sparse-checkout” documentation man page by adding an “OPTIONS” section to the doc. This change will better illustrate the usage of “sparse-checkout” subcommands and options.

Related Work

Prior works about the idea have been done by my mentors and other community members, and these works form a good approximation of the approach I’m going to take. Some previous example commits:

Integration with “clean”
Integration with “blame”

In GSoC

Plan

The proposed idea is relatively easy to understand: make more “sparse-index” integrations. However, as I have experienced a few patches, I realized that the obvious idea concept leads to much more covert difficulties. For example, before enabling “sparse-index”, we have to make sure that “sparse-checkout” is decently compatible with the target Git command.In order to do this, modifications in the original command logic need to be made, which could potentially lead to many other unforeseen issues. Therefore, I added two steps in the plan below (point 1 and point 2) to better accommodate potential issues. Notice that points 3-7 are from SoC 2022 Ideas, proposed by the community and mentors.

With this idea’s clear spirit to cover as many command integrations as possible, and also with the clear steps to take to finish each integration, I have arranged a general plan as below:

1. Investigation around a Git command and see if it behaves correctly with sparse-checkout. [Approx. 3 - 7 days]

2. Modify the Git command’s logic so that it works better with sparse-checkout. Add corresponding tests. [Approx. 7 days - 15 days]

3. Add tests to t1092-sparse-checkout-compatibility.sh for the builtin, with a focus on what happens for paths outside of the sparse-checkout cone.

4. Disable the command_requires_full_index setting in the builtin and ensure the tests pass.

5. If the tests do not pass, then alter the logic to work with the sparse index.

6. Add tests to check that a sparse index stays sparse.

7. Add performance tests to demonstrate speedup.

8. Update the Git command’s documentation accordingly, if any change happens to change its behavior (though usually this should not happen).

[points 3-8 added together should take Approx. 7 days - 15 days]

In summary, each integration will have a similar schedule stated above. So, without extending the timeline, it is expected to finish 3 - 4 integrations during the GSoC program period.

Timeline

I’m confident that I can start the project as early as the start of the Community Bonding Period (May 20 - June 12). This is because I have made contacts with my mentors, and have read the related documentation, though it should not be comprehensive. Added the prior experience with the idea, I believe I’ll be ready to get up to speed to work on the project by then.

The exact time arrangement of each integration is hard to determine, but the general pace, from my estimation, should roughly be doing **an integration per month**, starting May 20.

Here is a proposed integration schedule:

1. “git-mv” (we are currently working on this one and the expected time of completion is before May 20.)
2. “git-rm”
3. “git-grep”
4. “git-rev-parse”
5. “git-fsck”

This schedule is based upon the proposed integration schedule from SoC 2022 Ideas. As I am working on “git-mv” with the mentors, we realized that this schedule, as proposed by Derrick to be “generally organized in order of least-difficult to most-difficult”, can have surprising difficulties. Some of these difficulties I encountered are, for example, “git-mv” does not work quite ideally with “sparse-checkout” yet, so the “sparse-index” integration has to wait, reference the conversation from this patch.

With that being said, this proposed integration schedule could be deceptive. And as I dig deeper into each command, suitable modifications to the schedule could ensue. Conclusion: each integration should take a month on average.

Availability

My summertime is reserved for GSoC, so I expect that I can work 5 days per week, 6 - 8 hours per day, and that is 30 - 40 hours a week. Certainly, I may want to go for a, say, motorcycle trip, sometimes, but I will try my best to keep this time commitment and be always available through the community’s mailing list

After GSoC

I realize that it will be much more helpful if our GSoC participants can stick around after the event. And that's exactly what I intend to do. I think doing open-source, especially with the community that powers a ubiquitous dev tool is simply cool. And in such a community, besides being cool, I can learn more and grow with it and possibly see myself making more important contributions if I keep participating.

When I first joined the community three months ago, the ancient way of collaborating through a mailing list by sending diff patches was really puzzling (GitHub was the only means that I knew about for open-source collaboration). But I was lucky that folk from last year's GSoC gave me a helping hand. I also want to offer my helping hand to the new people who may be as confused as I was, like passing a torch.

I'm also thinking about writing some serious blogs about my development around Git. I can share my insights as a student participant developer and different technical details about the experimental "sparse-checkout" command, or even Git in general: its internals, daily tricks, etc. In this way, I can also spread the influence of GSoC, Git, and open-source in a positive way.

Some Credits to Myself

I've contributed to other open-source projects, though still a beginner, I'm generally familiar with the process of contribution. The related experiences are all in the contribution graph on my GitHub profile page. In the Casbin community, **I've made over 50 PRs and resolved over 30 issues**.

I came to participate in the Git community fairly early this year (around January). I got myself rather comfortable with the contribution process by writing, replying, and auditing different sorts of patches in the community.

With the patches done so far, I'm getting more familiar with the Git internals, project structures, commonly used APIs, test suites, required tech stacks, and coding guidelines. **For understanding better about Git, I read and reread the documentation a few times, including 'MyFirstContribution.txt', 'MyFirstObjectWalk.txt', 'sparse-index.txt', and 'Hacking Git'.** The book Pro Git also helps me to understand the Git internals better. Other than that, **I also fully read and referenced the blogs Make your monorepo feel small with Git's sparse index and Bring your monorepo down to size with sparse-checkout**

written by mentor Derrick. The best part is that the prior knowledge and experience with my proposed project idea make me well-prepared for the upcoming challenges.

Closing remarks

I’m currently in a gap year between Freshman and Sophomore, so I will have sufficient time to work on the project, without distraction from schoolwork.

I want to say that I’m a genuinely enthusiastic open-source newbie, and I’m truly looking forward to getting this opportunity. With this opportunity, I can see myself empowering Git’s development and even reinvigorating the open-source ecosystem, with my greatest endeavor possible.

In the very end, I want to say that I truly appreciate the support from the community and especially the mentors: Victoria and Derrick. They did and are doing an incredible job to maintain and empower the Git open-source community; they also provided and are providing the most needed and warm support to a new contributor like me.

Thanks & Regards,
Shaoxuan