Scalar - an opinionated repository management tool

Scalar is an add-on to Git that helps users take advantage of advanced performance features in Git. Originally implemented in C# using .NET Core, based on the learnings from the VFS for Git project, most of the techniques developed by the Scalar project have been integrated into core Git already:

- partial clone,
- commit graphs,
- multi-pack index,
- sparse checkout (cone mode),
- scheduled background maintenance,
- etc

This directory contains the remaining parts of Scalar that are not (yet) in core Git.

Roadmap

The idea is to populate this directory via incremental patch series and eventually move to a top-level directory next to `gitk-git/` and to `git-gui/`. The current plan involves the following patch series:

- `scalar-the-beginning`: The initial patch series which sets up `contrib/scalar/` and populates it with a minimal `scalar` command that demonstrates the fundamental ideas.

- `scalar-c-and-C`: The `scalar` command learns about two options that can be specified before the command, `-c <key>=<value>` and `-C <directory>`.

- `scalar-diagnose`: The `scalar` command is taught the `diagnose` subcommand.

- `scalar-and-builtin-fsmonitor`: The built-in FSMonitor is enabled in `scalar register` and in `scalar clone`, for an enormous performance boost when working in large worktrees. This patch series necessarily depends on Jeff Hostetler's FSMonitor patch series to be integrated into Git.

- `scalar-gentler-config-locking`: Scalar enlistments are registered in the user's Git config. This usually does not represent any problem because it is rare for a user to register an enlistment. However, in Scalar's functional tests, Scalar enlistments are created galore, and in parallel, which can lead to lock contention. This patch series works around that problem by re-trying to lock the config file in a gentle fashion.

- `scalar-extra-docs`: Add some extensive documentation that has been written in the original Scalar project (all subject to discussion, of course).

- `optionally-install-scalar`: Now that Scalar is feature (and documentation) complete and is verified in CI builds, let's offer to install it.

- `move-scalar-to-toplevel`: Now that Scalar is complete, let's move it next to `gitk-git/` and to `git-gui/`, making it a top-level command.

The following two patch series exist in Microsoft's fork of Git and are publicly available. There is no current plan to upstream them, not because I want to withhold these patches, but because I don't think the Git community is interested in these patches.

There are some interesting ideas there, but the implementation is too specific to Azure Repos and/or VFS for Git to be of much help in general (and also: my colleagues tried to upstream some patches already and the enthusiasm for integrating things related to Azure Repos and VFS for Git can be summarized in very, very few words).

These still exist mainly because the GVFS protocol is what Azure Repos has instead of partial clone, while Git is focused on improving partial clone:

- `scalar-with-gvfs`: The primary purpose of this patch series is to support existing Scalar users whose repositories are hosted in Azure Repos (which does not support Git's partial clones, but supports its predecessor, the GVFS protocol, which is used by Scalar to emulate the partial clone).

  Since the GVFS protocol will never be supported by core Git, this patch series will remain in Microsoft's fork of Git.

- `run-scalar-functional-tests`: The Scalar project developed a quite comprehensive set of integration tests (or, "Functional Tests"). They are the sole remaining part of the original C#-based Scalar project, and this patch adds a GitHub workflow that runs them all.

  Since the tests partially depend on features that are only provided in the `scalar-with-gvfs` patch series, this patch cannot be upstreamed.