scalar

Scalar ======

Scalar is a repository management tool that optimizes Git for use in large repositories. It accomplishes this by helping users to take advantage of advanced performance features in Git. Unlike most other Git built-in commands, Scalar is not executed as a subcommand of ’git’; rather, it is built as a separate executable containing its own series of subcommands.

Background ----------

Scalar was originally designed as an add-on to Git and implemented as a .NET Core application. It was created based on the learnings from the VFS for Git project (another application aimed at improving the experience of working with large repositories). As part of its initial implementation, Scalar relied on custom features in the Microsoft fork of Git that have since been integrated into core Git:

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

With the requisite Git functionality in place and a desire to bring the benefits of Scalar to the larger Git community, the Scalar application itself was ported from C# to C and integrated upstream.

Features --------

Scalar is comprised of two major pieces of functionality: automatically configuring built-in Git performance features and managing repository enlistments.

The Git performance features configured by Scalar (see "Background" for examples) confer substantial performance benefits to large repositories, but are either too experimental to enable for all of Git yet, or only benefit large repositories. As new features are introduced, Scalar should be updated accordingly to incorporate them. This will prevent the tool from becoming stale while also providing a path for more easily bringing features to the appropriate users.

Enlistments are how Scalar knows which repositories on a user’s system should utilize Scalar-configured features. This allows it to update performance settings when new ones are added to the tool, as well as centrally manage repository maintenance. The enlistment structure - a root directory with a `src/` subdirectory containing the cloned repository itself - is designed to encourage users to route build outputs outside of the repository to avoid the performance-limiting

overhead of ignoring those files in Git.

Design ------

Scalar is implemented in C and interacts with Git via a mix of child process invocations of Git and direct usage of `libgit.a`. Internally, it is structured much like other built-ins with subcommands (e.g., `git stash`), containing a `cmd_<subcommand>()` function for each subcommand, routed through a `cmd_main()` function. Most options are unique to each subcommand, with `scalar` respecting some "global" `git` options (e.g., `-c` and `-C`).

Because `scalar` is not invoked as a Git subcommand (like `git scalar`), it is built and installed as its own executable in the `bin/` directory, alongside `git`, `git-gui`, etc.

□□□ #2
□ □□□ □□□ 2026-10-31 20:17:14 UTC
□ □□□ □□□ 2026-10-31 20:17:15 UTC