
The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

About

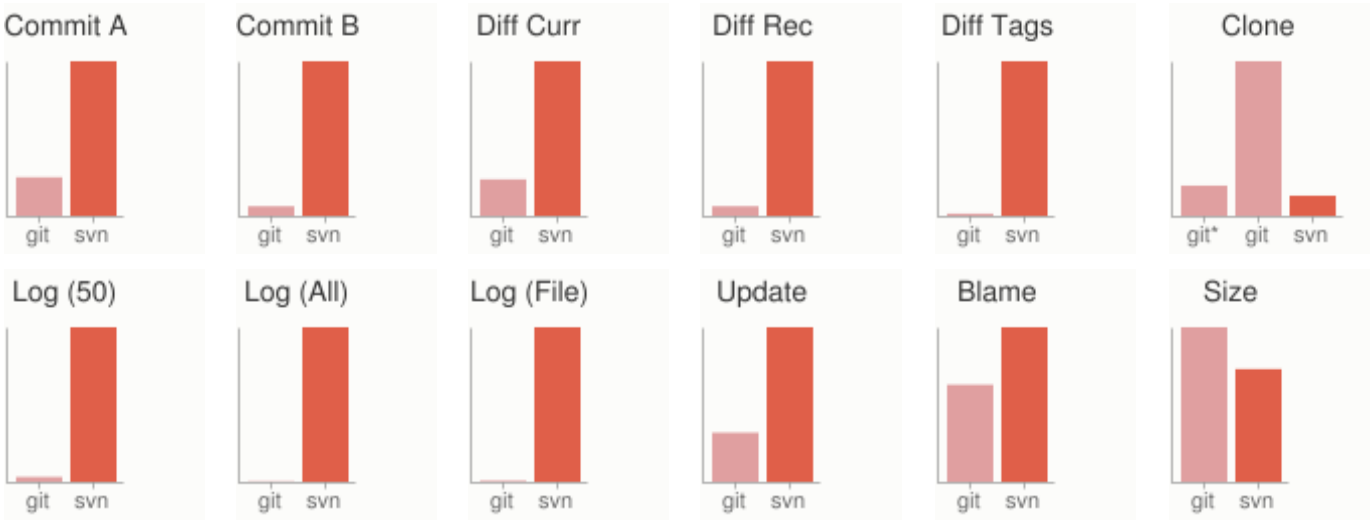
Small and Fast

Git is fast. With Git, nearly all operations are performed locally, giving it a huge speed advantage on centralized systems that constantly have to communicate with a server somewhere.

Git was built to work on the Linux kernel, meaning that it has had to effectively handle large repositories from day one. Git is written in C, reducing the overhead of runtimes associated with higher-level languages. Speed and performance has been a primary design goal of the Git from the start.

Benchmarks

Let's see how common operations stack up against Subversion, a common centralized version control system that is similar to CVS or Perforce. *Smaller is faster.*



For testing, large AWS instances were set up in the same availability zone. Git and SVN were installed on both machines, the Ruby repository was copied to both Git and SVN servers, and common operations were performed on both.

In some cases the commands don't match up exactly. Here, matching on the lowest common denominator was attempted. For example, the 'commit' tests also include the time to push for Git, though most of the time you would not actually be pushing to the server immediately after a commit where the two commands cannot be separated in SVN.

All of these times are in seconds.

Operation		Git	SVN	
Commit Files (A)	Add, commit and push 113 modified files (2164+, 2259-)	0.64	2.60	4x
Commit Images (B)	Add, commit and push a thousand 1 kB images	1.53	24.70	16x
Diff Current	Diff 187 changed files (1664+, 4859-) against last commit	0.25	1.09	4x
Diff Recent	Diff against 4 commits back (269 changed/3609+,6898-)	0.25	3.99	16x
Diff Tags	Diff two tags against each other (v1.9.1.0/v1.9.3.0)	1.17	83.57	71x
Log (50)	Log of the last 50 commits (19 kB of output)	0.01	0.38	31x
Log (All)	Log of all commits (26,056 commits – 9.4 MB of output)	0.52	169.20	325x
Log (File)	Log of the history of a single file (array.c – 483 revs)	0.60	82.84	138x
Update	Pull of Commit A scenario (113 files changed, 2164+, 2259-)	0.90	2.82	3x
Blame	Line annotation of a single file (array.c)	1.91	3.04	1x

Note that this is the best case scenario for SVN—a server with no load with a gigabit connection to the client machine. Nearly all of these times would be even worse for SVN if that connection was slower, while many of the Git times would not be affected.

Clearly, in many of these common version control operations, **Git is one or two orders of magnitude faster than SVN**, even under ideal conditions for SVN.

One place where Git is slower is in the initial clone operation. Here, Git is downloading the entire history rather than only the latest version. As seen in the above charts, it's not considerably slower for an operation that is only performed once.

Operation		Git*	Git	SVN
Clone	Clone and shallow clone(*) in Git vs checkout in SVN	21.0	107.5	14.0
Size (MB)	Size of total client side data and files after clone/checkout (in MB)		181.0	132.0

It's also interesting to note that the size of the data on the client side is very similar even though Git also has every version of every file for the entire history of the project. This illustrates how efficient it is at

compressing and storing data on the client side.

← Branching and Merging Distributed →
About this site
Patches, suggestions, and comments are welcome.
Git is a member of Software Freedom Conservancy