# *Speed up your Vagrant NFS shares with cachefilesd* *09 Mar 2014*

Most web based tech startups are deploying to cloud hosted Linux machines. At the same time, only a small percentage of engineers actually run Linux on their development desktops. Enter Vagrant, an easy way to provision local Linux development environments in a virtual machine.

At my current startup, we're using Vagrant with our regular Chef recipes so that the configuration of development matches production as closely as possible. The primary difference is the way the code is loaded. Instead of being saved on the VM itself, it's mounted via a shared directory with the host. That way, the 1/4 or so of the engineers on the team that use a graphical IDE as their editor get the super fast file access they require.

In that setup, although we're editing files on the host system, we run git inside the guest VM. This is done to support git hooks that rely on our dependencies that are only installed inside the VM.

## *VirtualBox Shared Folders are Slow*

Right at the start, we noticed considerable lag running git commands in this setup. At this point we were using the default synced folder mechanism, VirtualBox shared folders. Our `Vagrantfile` had a section like this:

```
config.vm.synced_folder "~/projects", "/home/vagrant/projects"
```

After a little research, we found that there were known issues with VirtualBox shared folders being very slow for large folders. The recommended solution was to use NFS.

## *Using NFS with Vagrant*

Vagrant makes switching to NFS fairly easy. It's all pretty seamless with a simple configuration change, except for the fact that folder permissions owner and group can only be set to the vagrant user. For us, this involved some chef recipe tweaking; as we normally place the files in the developer's home directory.

```
config.vm.synced_folder "~/projects", "/home/vagrant/projects", type: "nfs"
```

You will need to do a `vagrant reload` to update the config.

*Note: don't try to use NFS4 with `mount_options: ['vers=4']` on a Mac host. The Mac NFS4 implementation is not ready for primetime..*

# Futher Performance Improvements with cachefilesd

Git commands were still not running instantly, which is part of the benefit of using git in the first place. Enter cachefilesd, a Linux service that caches NFS file access.

You can install it manually in your Vagrant instance, just to see what the performance difference is.

```
sudo apt-get install cachefilesd
sudo echo "RUN=yes" > /etc/default/cachefilesd
```

Then, update your `Vagrantfile` as follows:

```
config.vm.synced_folder "~/projects", "/home/vagrant/projects", type: "nfs", moun
t_options: ['rw', 'vers=3', 'tcp', 'fsc']  # the fsc is for cachedfilesd
```

You will need to do a `vagrant reload` to update the config. You can check that cachefilesd is actually working by listing the cache directory inside the VM:

```
cd /var/cache/fscache/
sudo du -sh
```

If you are using Chef, you can install cachefilesd with something like the following:

```
package "cachefilesd" do
  action :install
end

file "/etc/default/cachefilesd" do
  content <<-EOS
RUN=yes
  EOS
  action :create
  mode 0755
end
```

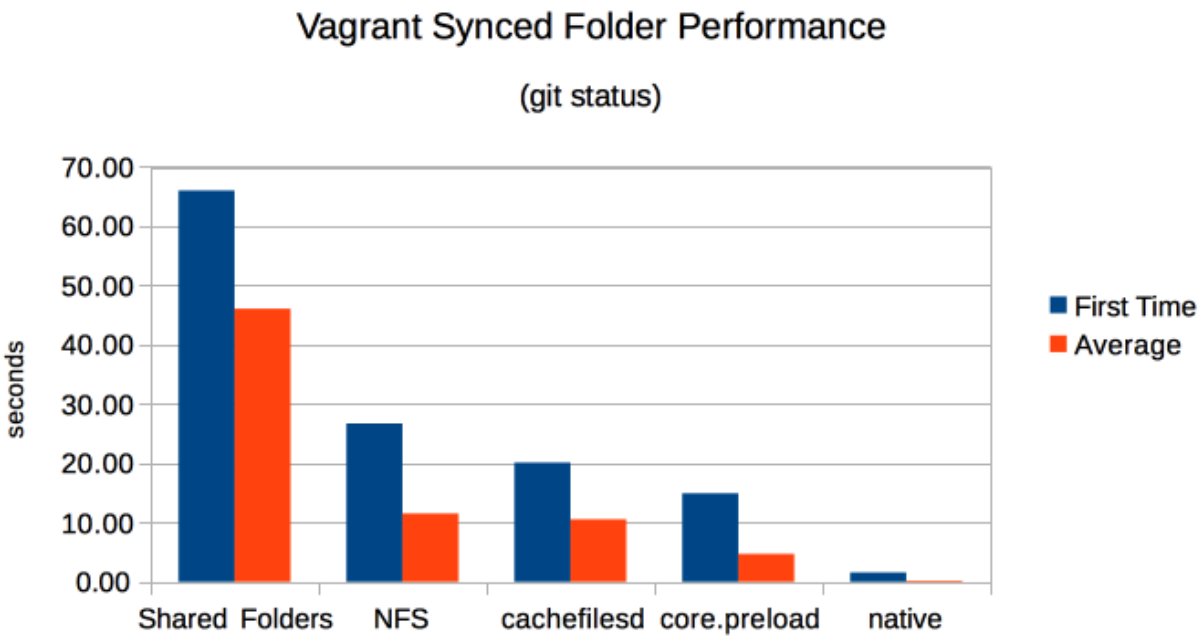
# Leveraging Git's preloadindex setting

I also found a git setting specifically designed to increase peformance over NFS. Hopefully this will be made a default git setting soon. Simply run the following.

```
git config core.preloadindex true
```

# Measuring the Performance Difference

Here are the stats for our git repository. This was in a repo with approximately 33,000 files. The host machine is OSX 10.9 with an SSD. The guest VM was configured with 2GB RAM and 2 CPUs. My methodology was to run `git status` four times for each configuration. The first time is noted separately, and the last three times are averaged. Time was measured using the `time` command.




# Still looking for a better solution

Git over NFS is not ideal, even when everything is "local". The main problem seems to be very large repos (tens of thousands of files or more). Smaller repos still perform in the hundreds of milliseconds.

Other stuff to try:

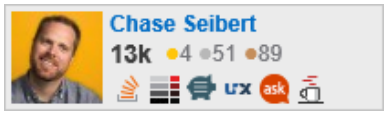
- Git 1.7 added sparse checkouts
- Git also has an --assume-unchanged that can be used to exlcude directories that don't often change.

I'm currently working at NerdWallet, a startup in San Francisco trying to bring clarity to all of life's financial decisions. We're hiring like crazy. Hit me up on Twitter, I would love to talk.

Follow @chase_seibert on Twitter

## Chase Seibert

Facts, hacks and attacks from my life as a web application developer




### Tags

- vagrant

### My Favorites

- Posts for new team members
- Recommended third party articles

### Related Posts

- React Native Six Months In
- Minimizing Impact of Interruptions on Engineers
- Best Practices for Meetings

### Social Links

- LinkedIn
- Facebook
- Twitter
- Resume