introduction

This document describes:

- creating an operating environment suitable for using the S3-subsetting benchmark suite developed in support of the Fornax project
- installing the suite into that environment
- basic usage instructions for the suite

an important note on scope
This is not a guide for using subsetting techniques for practical purposes. Most of the setup instructions in this document support specific needs of the benchmark suite itself, and the benchmark suite is not intended to support or demonstrate any particular science use case.

setup

step 1: make an instance

Create an EC2 instance in us-east-1 with the following parameters:

- AWS Ubuntu 22.04 (Jammy) LTS AMI
- m6i.large instance type
- 32 GB gp3 EBS root volume

Make sure that the security group you use for this instance allows SSH access from your IP.

notes

- I specify us-east-1 because all of the objects the currently-written benchmarks access are in buckets in us-east-1, so the bucket owners will incur egress costs if you access them from a different AWS Region. There is nothing else special about us-east-1. If you would like to write a different set of benchmarks that access objects in a different AWS Region, you should create an instance there instead.
- The instructions below specify use of superuser privileges for a variety of purposes. This is mostly not strictly necessary. You can install *goofys* and *wondershaper* into non-privileged directories, so long as they are in your working path. Actually making use of *wondershaper* for bandwidth-throttled tests *does* require superuser privileges, although those tests can be deactivated. Other than that, you could install and use this suite in an environment in which you do not have superuser privileges, so long as it has not been specially configured so that users cannot mount and dismount FUSE filesystems and access device information (like TIKE). However, I strongly recommend just creating an EC2 instance you control.
- I specify Ubuntu 22.04 LTS because it is the highest Ubuntu version with an officially-supported AMI at the time of writing (2022-08-16). I do not anticipate any

forward compatibility issues, so if you are using this guide after the release of a new official Ubuntu AMI, use the newer AMI instead.

- The benchmark suite will most likely work on other Linux distributions, although I have not tested it. Some paths, instructions to package managers, etc. might need to be changed, depending on the distribution. It could also be made to work on MacOS with fairly small modifications. It cannot be made compatible with Windows without a total rewrite, because it relies on POSIX tools and conventions to acquire its metrics.
- I specify m6i.large because it is an inexpensive general-purpose instance type that is a good median representative of instance types commonly used for cloud science platforms. I specify a 32 GB root volume because it will provide a comfortable amount of headroom, along with scratch space if you want to pull specific FITS files locally to investigate results. However, many other instance setups will work, with the following caveats:
    - Some benchmark results will be significantly affected by CPU characteristics (especially single-thread speed), RAM speed, network bandwidth, and probably a bunch of other stuff.
        - This also means that running these benchmarks on a variety of instances might help you select instance types for practical use cases!
    - I do not recommend using an instance with less than 4 GB of total RAM.
    - I have no particular reason to anticipate problems with ARM architectures, but have only tested on x86_64.
    - As currently written, the benchmark suite only writes logs to local storage, so you can get away with a smaller root volume if you like, although I do not recommend less than 8 GB.
        - Note that modifications to volume IOPS and I/O speed parameters should not meaningfully affect any benchmarks, unless you modify the code to make it scratch FITS files to disk, or something like that.

step 2: install support software

1. SSH to the instance and update the stock OS software:
    a. *sudo apt update && sudo apt upgrade*
2. Install and initialize your preferred version of *conda*.
    a. I like Mambaforge (https://github.com/conda-forge/miniforge) but this is not mandatory. I would not recommend Anaconda, because it will install many unnecessary packages that will take up space for no reason.
    b. make sure *conda* is accessible from your path. the installer will run *conda init* by default, but you will need to log out and back in, open a new shell, source your rcfile, or something like that in order for it to be accessible immediately after installation.
    c. note: the benchmarking suite does not actually use any *conda* tools at runtime, so if you strongly prefer to set your Python environment up some other way, go ahead, but you're on your own for that: the instructions below assume a working *conda* installation.
3. install goofys (used for FUSE mounts)

    a. *wget https://github.com/kahing/goofys/releases/latest/download/goofys*
    b. make it executable and place/link it somewhere that will be in the “ubuntu” user’s path. for example:
        i. *chmod +x goofys && sudo mv goofys /usr/bin/goofys*
4. install wondershaper (used for bandwidth throttling)
    a. *git clone https://github.com/magnific0/wondershaper.git*
    b. make the wondershaper/wondershaper file executable and put it somewhere in the “ubuntu” user's path, as in step 3
5. clone the fornax-s3-subsets repo:
    a. *git clone https://github.com/fornax-navo/fornax-s3-subsets.git*
6. Create the “fornax-bench” *conda* environment using the fornax-s3-subsets/subset/bench_environment.yml file:
    a. *cd fornax-s3-subsets/subset && mamba env create -f bench_environment.yml* (or *conda env…* if you didn’t install Mambaforge)
    b. **important**: make sure you have this environment active when you execute any code from the benchmark suite (*conda activate fornax-bench*), or else explicitly pass paths to that environment’s interpreter(s).
    c. note: this environment file installs astropy from G. Barentsen’s cloud-support branch using pip, which is why it explicitly includes a lot of astropy dependencies (it helps ensure correctness of the conda environment solver). We will likely modify this to install from trunk astropy after Barentsen’s PR is merged. Also note that the environment file includes a number of nice-to-haves (like jupyter) that are not strictly necessary for running benchmarks.
7. reboot (*sudo reboot)* and SSH back into the instance.

step 3: set up paths and credentials

1. Set up AWS client configuration
    a. **Please note that some of the currently-written settings refer to objects in a private bucket, ‘nishapur’. STScI is planning to stage some of these objects publicly; in the interim, these will not be usable unless you stage the objects themselves.** If you are using objects in a private bucket, place the relevant AWS credentials in ~/.aws/credentials. If you only want to use files in public buckets like stpubdata, you don’t need to do this. The suite does not currently support specifying multiple AWS accounts.
    b. define us-east-1 as a default AWS region in ~/.aws/config so that *fsspec* and *goofys* will be able to find the buckets. I also recommend setting the default AWS output style to JSON. Here is a minimal example:

```
[default]
region = us-east-1
output = json
```

2. Make a mountpoint for *goofys*. The execution notebook sets this to /home/ubuntu/s3 by default. If you'd like to use this default mountpoint, run:
    a. *mkdir /home/ubuntu/s3*
    b. You can change this by modifying the "mountpoint" value in the SETTINGS variable in that notebook. (It does not need to be in /home/ubuntu, so long as it is writable by the default "ubuntu" user.)
    c. note: because *goofys* uses *libfuse*, no further configuration is required to allow the default "ubuntu" user to mount an S3 bucket. FUSE enables creation of non-privileged filesystems in userspace, both networked and otherwise. (For instance, it is also commonly used to manage SD card mounts in Android.)

execution

Now you are fully prepared to use the suite to run S3-subsetting benchmarks. The intended entry point is fornax-s3-subsets/subset/execute_benchmarks.ipynb, and I recommend tunneling to a Jupyter server on your instance and executing benchmarks from that Notebook. If you prefer, however, you can extract code from that Notebook and execute it however you like.

The suite is highly configurable. You can edit the submodules of *subset.benchmark.benchmark_settings* to modify the benchmark parameter space, use different files, etc. Some settings can also be defined at runtime; execute_benchmarks.ipynb includes descriptions of all of these options. You can also define new benchmark cases by creating new submodules of *subset.benchmark.benchmark_settings*.

The benchmark suite uses file size and property information in *fileinfo.csv files contained in fornax-s3-subsets/subset/benchmark/benchmark_settings to generate derived statistics (e.g., percentage of HDU data volume transferred). If you add new FITS files, you will need to retrieve information about them. I have provided a script for this purpose: fornax-s3-subsets/subset/regenerate_benchmark_fileinfo.py. If you edit the BENCHMARKS_FOR_WHICH_TO_REGENERATE_FILEINFO variable in that script to specify your benchmarks of interest and then execute it from the command line, it will examine all files mentioned in those benchmark modules and generate new *fileinfo.csv files for each module.