Amazon S3 (Simple Storage Service) is a data storage service. You are billed monthly for storage and data transfer. Transfer between S3 and AmazonEC2 is free. This makes use of S3 attractive for Hadoop users who run clusters on EC2.
Hadoop provides multiple filesystem clients for reading and writing to and from Amazon S3 or compatible service.

S3 Native FileSystem (URI scheme: s3n)

A native filesystem for reading and writing regular files on S3. The advantage of this filesystem is that you can access files on S3 that were written with other tools. Conversely, other tools can access files written using Hadoop. The disadvantage is the 5GB limit on file size imposed by S3.

S3A (URI scheme: s3a)

A successor to the S3 Native, s3n fs, the S3a: system uses Amazon's libraries to interact with S3. This allows S3a to support larger files (no more 5GB limit), higher performance operations and more. The filesystem is intended to be a replacement for/successor to S3 Native: all objects accessible from s3n:// URLs should also be accessible from s3a simply by replacing the URL schema.

S3 Block FileSystem (URI scheme: s3)

A block-based filesystem backed by S3. Files are stored as blocks, just like they are in HDFS. This permits efficient implementation of renames. This filesystem requires you to dedicate a bucket for the filesystem - you should not use an existing bucket containing files, or write other files to the same bucket. The files stored by this filesystem can be larger than 5GB, but they are not interoperable with other S3 tools.

S3 can be used as a convenient repository for data input to and output for analytics applications using either S3 filesystem. Data in S3 outlasts Hadoop clusters on EC2, so they should be where persistent data must be kept.
Note that by using S3 as an input you lose the data locality optimization, which may be significant. The general best practise is to copy in data using `distcp` at the start of a workflow, then copy it out at the end, using the transient HDFS in between.

History

- The S3 block filesystem was introduced in Hadoop 0.10.0 (HADOOP-574).
- The S3 native filesystem was introduced in Hadoop 0.18.0 (HADOOP-930) and rename support was added in Hadoop 0.19.0 (HADOOP-3361).
- The S3A filesystem was introduced in Hadoop 2.6.0. Some issues were found and fixed for later Hadoop versionsHADOOP-11571, so Hadoop-2.6.0's support of s3a must be considered an incomplete replacement for the s3n FS.

Why you cannot use S3 as a replacement for HDFS

You cannot use either of the S3 filesystem clients as a drop-in replacement for HDFS. Amazon S3 is an "object store" with

- eventual consistency: changes made by one application (creation, updates and deletions) will not be visible until some undefined time.
- s3n and s3a: non-atomic rename and delete operations. Renaming or deleting large directories takes time proportional to the number of entries -and visible to other processes during

this time, and indeed, until the eventual consistency has been resolved.

S3 is not a filesystem. The Hadoop S3 filesystem bindings make it pretend to be a filesystem, but it is not. It can act as a source of data, and as a destination -though in the latter case, you must remember that the output may not be immediately visible.

Configuring to use s3/ s3n filesystems

Edit your `core-site.xml` file to include your S3 keys

```
<property>
  <name>fs.s3.awsAccessKeyId</name>
  <value>ID</value>
</property>

<property>
  <name>fs.s3.awsSecretAccessKey</name>
  <value>SECRET</value>
</property>
```

You can then use URLs to your bucket : s3n://MYBUCKET/, or directories and files inside it.

```
s3n://BUCKET/
s3n://BUCKET/dir
s3n://BUCKET/dir/files.csv.tar.gz
s3n://BUCKET/dir/*.gz
```

Alternatively, you can put the access key ID and the secret access key into a *s3n* (or *s3*) URI as the user info:

```
  s3n://ID:SECRET@BUCKET
```

Note that since the secret access key can contain slashes, you must remember to escape them by replacing each slash `/` with the string `%2F`. Keys specified in the URI take precedence over any specified using the properties `fs.s3.awsAccessKeyId` and `fs.s3.awsSecretAccessKey`.
This option is less secure as the URLs are likely to appear in output logs and error messages, so being exposed to remote users.

Security

Your Amazon Secret Access Key is that: secret. If it gets known you have to go to the Security Credentials page and revoke it. Try and avoid printing it in logs, or checking the XML configuration files into revision control. Do not ever check it in to revision control systems.

Running bulk copies in and out of S3

Support for the S3 block filesystem was added to the `${HADOOP_HOME}/bin/hadoop distcp` tool in Hadoop 0.11.0 (See HADOOP-862). The `distcp` tool sets up a MapReduce job to run the copy. Using `distcp`, a cluster of many members can copy lots of data quickly. The number of map tasks is calculated by counting the number of files in the source: i.e. each map task is responsible for the copying one file. Source and target may refer to disparate filesystem types. For example, source might refer to the local filesystem or `hdfs` with `S3` as the target.
The `distcp` tool is useful for quickly prepping S3 for MapReduce jobs that use S3 for input or for backing up the content of `hdfs`.
Here is an example copying a nutch segment named `0070206153839-1998` at `/user/nutch` in `hdfs` to an S3

bucket named 'nutch' (Let the S3 AWS_ACCESS_KEY_ID be `123` and the S3 AWS_ACCESS_KEY_SECRET be `456`):

```
% ${HADOOP_HOME}/bin/hadoop distcp hdfs://domU-12-31-33-00-02-DF:9001/user/nutch/0070206153839-1998 s3://123:456@nutch/
```

Flip the arguments if you want to run the copy in the opposite direction.
Other schemes supported by `distcp` include `file:` (for local), and `http:`.