Rubix Documentation

Release 0.2.12

Qubole

Feb 09, 2022

Contents:

RubiX is a light-weight data caching framework that can be used by Big-Data engines. RubiX can be extended to support any engine that accesses data in cloud stores using Hadoop FileSystem interface via plugins. Using the same plugins, RubiX can also be extended to be used with any cloud store

CHAPTER 1

RubiX

RubiX is a light-weight data caching framework that can be used by Big-Data engines. RubiX can be extended to support any engine that accesses data in cloud stores using Hadoop FileSystem interface via plugins. Using the same plugins, RubiX can also be extended to be used with any cloud store

1.1 Usecase

RubiX provides disk or in-memory caching of data, which would otherwise be accessed over network when it resides in cloud store, thereby improving performance.

1.2 Supported Engines and Cloud Stores

- Presto: Amazon S3
- Spark: Amazon S3
- Any engine using hadoop-2 or hadoop-1, e.g. Hive can utilize RubiX. Amazon S3 is supported

CHAPTER 2

Installation Guide

This section provides instructions to install Rubix and use it with Presto or Hive. Instructions are tested on an EMR 5.X cluster. If you want help to install on-prem or on other distributions, please contact the community with your questions or issues.

The instructions require RubiX Admin

2.1 Start and connect to master

2.1.1 Start an EMR Cluster

Create an EMR cluster, use 'advanced option' to create a cluster with the required spec.

1. Specify at least 30GB space for Root device EBS volume.
2. To login into cluster, choose a EC2 key pair.

2.1.2 Connect to Master Node

Log into master and all worker nodes as "hadoop" user.

```
ssh <key> hadoop@master-public-ip and install required libs.
```

Setup and check passwordless SSH between cluster machines

```
ssh hadoop@localhost
ssh hadoop@worker-public-ip
```

2.2 Install and start RubiX

2.2.1 Install RubiX Admin

```
pip install rubix_admin
```

2.2.2 Update Config File

```
rubix_admin -h
```

This will create rubix-admin config file at ~/.radminrc with the follwoing format

```
hosts:
  - localhost
  - worker-ip1
  - worker-ip2
  ..
remote_packages_path: /tmp/rubix_rpms
```

2.2.3 Install RubiX

```
rubix_admin installer install
```

This will install the latest version of RubiX. To install a specific version of Rubix,

```
rubix_admin installer install --rpm-version <RubiX Version>
```

To install from a rpm file,

```
rubix_admin installer install --rpm <path-to-rubix-rpm>
```

To enable debugging and see the rubix activity, create /usr/lib/presto/etc/log.properties file with bellow config. com.qubole=DEBUG

2.2.4 Start RubiX Daemons

```
rubix_admin daemon start --debug
# To verfiy the daemons are up
   verfiy process ids for both
   BookKeeperServer and LocalDiscoveryServer.
sudo jps -m
```

2.3 Presto

2.3.1 Restart Presto Server

```
sudo restart presto-server
```

2.4 Apache Hive

2.4.1 Add RubiX jars to Hadoop ClassPath

Copy jars to Hadoop Lib

```
cp /usr/lib/rubix/lib/rubix-* /usr/lib/hadoop/lib
```

OR

Add Rubix Jars

```
add jar /usr/lib/rubix/lib/rubix-bookkeeper.jar
add jar /usr/lib/rubix/lib/rubix-core.jar
add jar /usr/lib/rubix/lib/rubix-hadoop2.jar
```

2.4.2 Restart Hive Metastore Server

```
hive --service metastore --stop
hive --service metastore --start
```

2.4.3 Configure Apache Hive to use RubiX FileSystem

```
hive --hiveconf \
    fs.rubix.impl=com.qubole.rubix.hadoop2.CachingNativeS3FileSystem \
    fs.rubix.awsAccessKeyId=<AWS ACCESS KEY> \
    fs.rubix.awsSecretAccessKey=<AWS SECRET ACCESS KEY>
```

2.4.4 (Advanced) Configure Apache Hive to use RubiX FileSystem for S3 and S3A schemes

If you use this option, all tables with their location in AWS S3 will automatically start using RubiX.

```
hive --hiveconf \
    fs.s3n.impl=com.qubole.rubix.hadoop2.CachingNativeS3FileSystem
        fs.s3.impl=com.qubole.rubix.hadoop2.CachingNativeS3FileSystem
        fs.s3a.impl=com.qubole.rubix.hadoop2.CachingS3AFileSystem
```

2.5 Run your first query using RubiX

2.5.1 Start Hive Client

```
hive --hiveconf hive.metastore.uris="" --hiveconf fs.rubix.impl=com.qubole.rubix.
↪hadoop2.CachingNativeS3FileSystem
```

2.5.2 Create External Table

```
CREATE EXTERNAL TABLE wikistats_orc_rubix
(language STRING, page_title STRING,
hits BIGINT, retrived_size BIGINT)
STORED AS ORC
LOCATION 'rubix://emr.presto.airpal/wikistats/orc';
```

2.5.3 Run Query (Presto or Hive CLI)

```
SELECT language, page_title, AVG(hits) AS avg_hits
FROM default.wikistats_orc_rubix
WHERE language = 'en'
AND page_title NOT IN ('Main_Page',  '404_error/')
AND page_title NOT LIKE '%index%'
AND page_title NOT LIKE '%Search%'
GROUP BY language, page_title
ORDER BY avg_hits DESC
LIMIT 10;
```

2.5.4 RubiX Stats (supported on Presto only)

The cache statistics are pushed to MBean named rubix:name=stats. To check the stats, execute

```
SELECT Node, CachedReads,
ROUND(extrareadfromremote,2) as ExtraReadFromRemote,
ROUND(hitrate,2) as HitRate,
ROUND(missrate,2) as MissRate,
ROUND(nonlocaldataread,2) as NonLocalDataRead,
NonLocalReads,
ROUND(readfromcache,2) as ReadFromCache,
ROUND(readfromremote, 2) as ReadFromRemote,
RemoteReads
FROM jmx.current."rubix:name=stats";
```

CHAPTER 3

Metrics

These are the metrics currently available for RubiX.

RubiX Metric	Description
rubix.bookkeeper.live_workers.gauge	The number of workers currently reporting to the master node.
rubix.bookkeeper.cache_eviction.count	The number of entries evicted from the local cache.
rubix.bookkeeper.cache_hit_rate.gauge	The percentage of cache hits for the local cache.
rubix.bookkeeper.cache_miss_rate.gauge	The percentage of cache misses for the local cache.
rubix.bookkeeper.cache_size.gauge	The current size of the local cache in MB.
rubix.bookkeeper.local_request.count	The number of requests made for data cached locally.
rubix.bookkeeper.nonlocal_request.count	The number of requests made for data cached on another node.
rubix.bookkeeper.remote_request.count	The number of requests made for data not currently cached.

CHAPTER 4

Contribution Guidelines

This section provides guidelines to contribute to the project through code, issues and documentation.

4.1 Developer Environment

Rubix is a Maven project and uses Java 8. It uses JUnit as the testing framework. Ensure that you have a development environment that support the above configuration.

- Fork your own copy of RubiX into your github account by clicking on the "Fork" button
- Navigate to your account and clone that copy to your development box

  git clone https://github.com/<username>/rubix
- Run tests in the RubiX root directory.

  mvn test
- Add Qubole RubiX as upstream

  git remote add upstream https://github.com/qubole/rubix.git git fetch upstream

4.2 How to contribute code on Github

4.2.1 1. Create a branch and start working on your change.

```
cd rubix
git checkout -b new_rubix_branch
```

4.2.2 2. Code

- Adhere to code standards.
- Include tests and ensure they pass.

4.2.3 3. Commit

For every commit please write a short (max 72 characters) summary in the first line followed with a blank line and then more detailed descriptions of the change.

Don't forget a prefix!

More details in Commit Guidelines

4.2.4 4. Update your branch

```
git fetch upstream
git rebase upstream/master
```

4.2.5 5. Push to remote

```
git push -u origin new_rubix_branch
```

4.2.6 6. Issue a Pull Request

- Navigate to the Rubix repository you just pushed to (e.g. https://github.com/your-user-name/rubix)
- Click *Pull Request*.
- Write your branch name in the branch field (this is filled with *master* by default)
- Click *Update Commit Range*.
- Ensure the changesets you introduced are included in the *Commits* tab.
- Ensure that the *Files Changed* incorporate all of your changes.
- Fill in some details about your potential patch including a meaningful title.
- Click *Send pull request*.

4.2.7 7. Respond to feedback

The RubiX team may recommend adjustments to your code. Part of interacting with a healthy open-source community requires you to be open to learning new techniques and strategies; *don't get discouraged!* Remember: if the RubiX team suggest changes to your code, they care enough about your work that they want to include it, and hope that you can assist by implementing those revisions on your own.

4.2.8 8. Postscript

Once all the changes are approved, one contributor will push the change to the upstream code.

4.3 Coding conventions

- two spaces, no tabs
- no trailing whitespaces, blank lines should have no spaces
- Do not mix multiple fixes into a single commit.
- Add comments for your future selves and for your current/future peers
- Do not make whitespace changes as part of your regular/feature commits.
- If you feel whitespace issues need to be fixed, please push a separate commit for the same. It will be approved quickly without any discussion.

4.4 Commit Message

Commits are used as a source of truth for various reports. A couple of examples are:

- Release Notes
- Issues resolved for QA to plan the QA cycle.

To be able to generate these reports, uniform commit messages are required. All your commits should follow the following convention:

For every commit please write a short (max 72 characters) summary in the first line followed with a blank line and then more detailed descriptions of the change.

Format of summary:

```
ACTION: AUDIENCE: COMMIT_MSG
```

Description:

```
ACTION is one of 'chg', 'fix', 'new'
Is WHAT the change is about.
'chg' is for refactor, small improvement, cosmetic changes...
'fix' is for bug fixes
'new' is for new features, big improvement

AUDIENCE is one of 'dev', 'usr', 'pkg', 'test', 'doc'
Is WHO is concerned by the change.
'dev' is for developers (API changes, refactors...)
'usr' is for final users
```

You will use your environment's default editor (EDITOR=vilemacs) to compose the commit message. Do NOT use the command line git commit -m "my mesg" as this only allows you to write a single line that most of the times turns out to be useless to others reading or reviewing your commit.

4.4.1 Example

```
new: dev: #124: report liveness metric for BookKeeper daemon (#139)

Add a liveness gauge that the daemon is up & alive. Right now, this
is a simple check that a thread (reporter to be added in a subsequent
```

(continues on next page)

(continued from previous page)

```
commit) is alive. In the future, this simple framework will be used
to add more comprehensive health checks. Ref: #140
```

The above example shows the commit summary is:

- a single line composed of four columns
- column 1 tells us the nature of the change or ACTION: new
- a short one-line summary of WHAT the commit is doing

The description or the body of the commit message delves into more detail that is intended to serve as a history for developers on the team on how the code is evolving. There are more immediate uses of this description however. When you raise pull requests to make your contributions into the project, your commit descriptions serve as explanations of WHY you fixed an issue. HOW you fixed an issue is explained by code already. This is also the place where the peer-reviewers will begin understanding your code. An unclear commit message is the source of a lot of back and forth resulting in frustration between reviewers and committers.

Reference: http://chris.beams.io/posts/git-commit/

4.5 How to report issues

A bug report means something is broken, preventing normal/typical use of Rubix.

Make sure the bug isn't already resolved. Search for similar issues.

Make sure you have clear instructions to reproduce your problem.

If possible, submit a Pull Request with a failing test, or;if you'd rather take matters into your own hands, try fix the bug yourselfMake a report of everything you know about the bug so far by opening an issue about it.When the bug is fixed, you can usually expect to see an update posted on the reporting issue.

4.6 Documentation Style Guide

- Documentation uses Sphinx documentation generator.
- Documentation is hosted on ReadTheDocs
- File issues if you notice bugs in documentation or to request more information. Label issues with `doc`
- Contributions to documentation is accepted as a Pull Request.
- Choose Markdown if you will add new pages
- Choose Rich Structured Text (rst) for indexes or if the documentation needs tables.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search