An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

RecordService Introduction

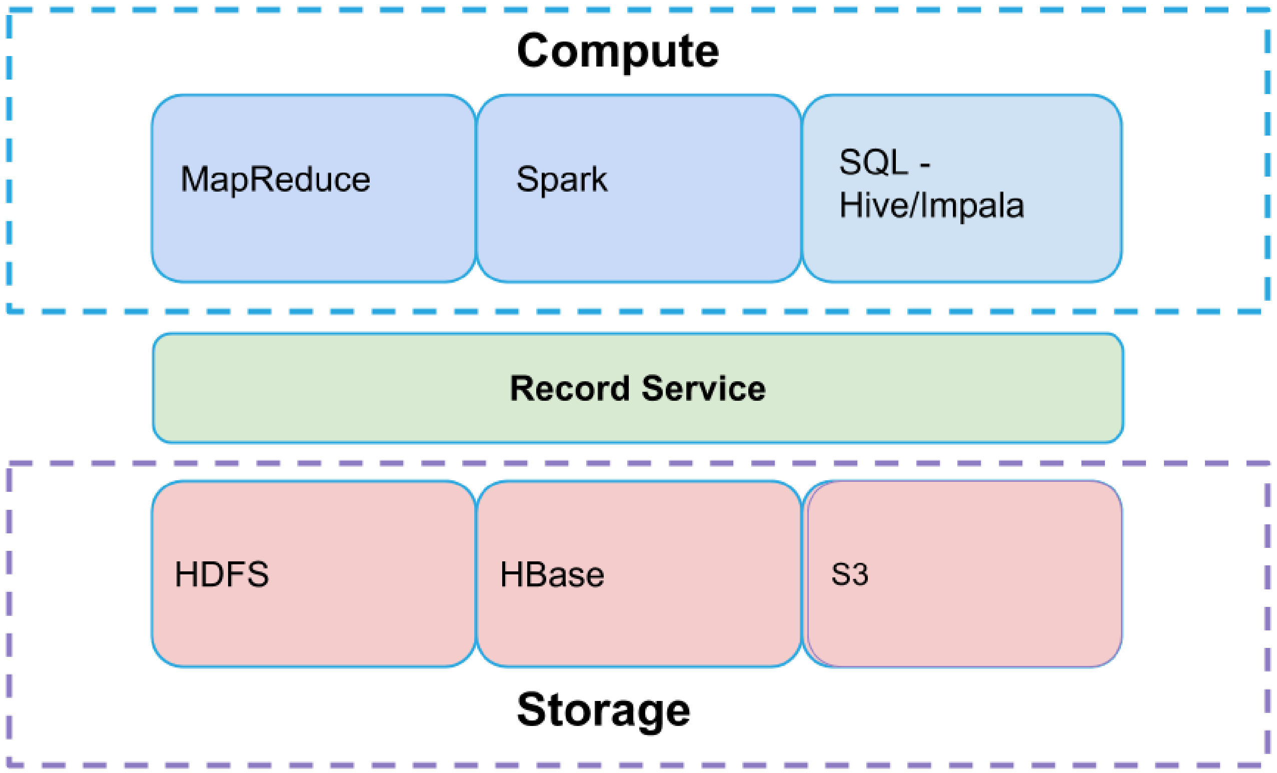
RecordService provides an abstraction layer between compute frameworks and data storage. It provides row- and column-level security, and other advantages.

The Motivation

One of the key aspects of the Hadoop ecosystem is decoupling storage managers (for example, HDFS and Apache HBase) and compute frameworks (for example, MapReduce, Impala and Apache Spark). Although this decoupling allows for far greater flexibility — pick the framework that best solves your problem — it leads to more complexity to ensure that everything works together seamlessly. Furthermore, as Hadoop becomes an increasingly critical infrastructure component for users, the expectations for compatibility, performance, and security also increase.

RecordService is a new core security layer for Hadoop that sits between the storage managers and compute frameworks to provide a unified data access path.

Unified Data Access Path



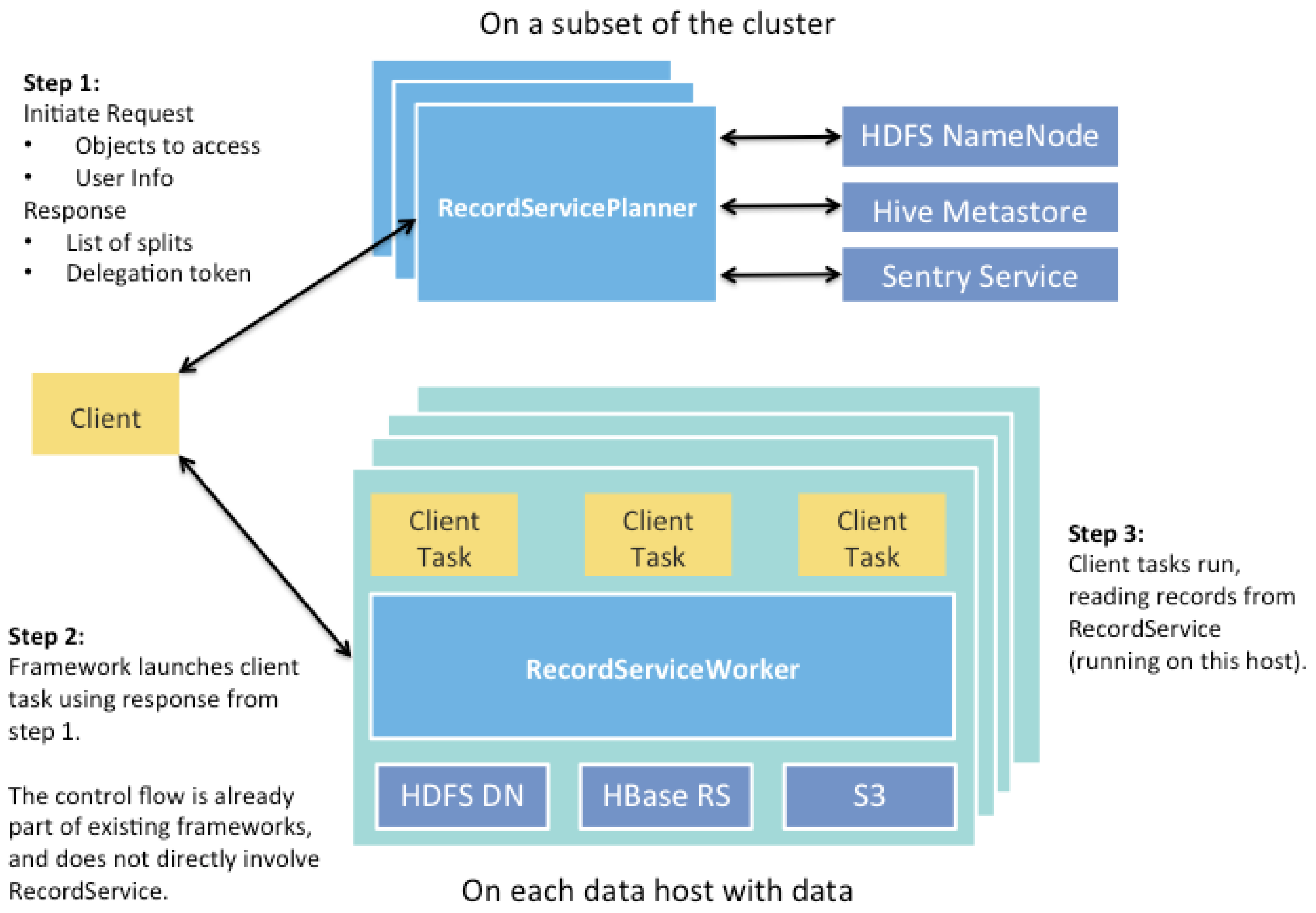
The Benefits

- RecordService provides fine-grained data permissions and enforcement across Hadoop.
- Clients work independently of on-disk storage format.
- The unified data access path provides a single place to implement and test file format–related changes.
- Components swap transparently above or below RecordService.
- Performance is improved through the optimized scanner, dynamic code generation, and Parquet implementation provided by Impala.
- Existing MapReduce and Spark jobs gain Impala-quality performance.
- RecordService can make projections over original source datasets instead of making copies or subsets.

The Design

RecordService provides the following services.

- RecordServicePlanner — Generates tasks, performs authorization checks, and handles metadata access. Called during input split generation.
- RecordServiceWorker — Executes tasks, and reads and writes to the storage layer.
    - Builds on the highly optimized I/O scheduler and file parsers provided by Impala.
    - Returns rows in a canonical format.
- Thrift APIs.
- Client Integration Libraries — Allow easy migration to RecordService.


On a subset of the cluster
Step 1:
Initiate Request
• Objects to access
• User Info
Response
• List of splits
• Delegation token
RecordServicePlanner
HDFS NameNode
Hive Metastore
Sentry Service
Client
Client Task
Client Task
Client Task
RecordServiceWorker
HDFS DN
HBase RS
S3
Step 2:
Framework launches client task using response from step 1.
The control flow is already part of existing frameworks, and does not directly involve RecordService.
Step 3:
Client tasks run, reading records from RecordService (running on this host).
On each data host with data

RecordService, Beta

Source

Mailing list
How to contribute

Examples
Release Notes

Generated on 23 October 2015

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

Getting Started with RecordService Beta

You can try RecordService beta in multiple ways. First, download the server via the virtual machine (VM) or by downloading the RecordServiceClient parcel or package.

RecordService Beta must be installed only on a test cluster, because it is beta software.

To integrate the RecordService into your existing MapReduce and Spark jobs, use the RecordService Client libraries. The client repository contains many example applications you can run right away.

See RecordService Examples.

Using the Beta VM

The RecordService beta VM provides a single-node Hadoop cluster, including Impala, MapReduce/YARN, Spark, Sentry, and RecordService. Using the VM is the easiest way to see RecordService in action, and it does not require that you use an existing Hadoop cluster.

The VM itself does not contain RecordService Client and examples: those should be run on a separate machine.

See Download and Install the RecordService VM.

Using RecordService on an Existing Hadoop Cluster

This requires a cluster running CDH 5.4.

See Download and Install RecordService.

Verify the Server is Working

Run some of the examples to see how programs can read data from RecordService, and to see how Sentry column- and row-level permissions are enforced on MapReduce and Spark jobs.

See Run a Job Using RecordService.

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

RecordService Beta Release Notes

This release of RecordService is a public beta and should not be run on production clusters. During the public beta period, RecordService is supported through the mailing list RecordService-user@googlegroups.com, not through the Cloudera Support Portal.

As you use RecordService during the public beta period, keep in mind the following:

- The RecordService team responds to beta issues as quickly as possible, but cannot commit to issue-resolution or bug-fix delivery times during the public beta period.
- There is no guarantee that a bug will be fixed in a future release.
- The RecordService team does not provide patches for beta releases, and cannot guarantee upgrades from this release to later releases.
- Although multiple releases of beta code might be planned, the contents are not guaranteed. There is no schedule for future beta code releases. Any releases are announced to the user group as they occur.

Release Notes Table of Contents

RecordService VM Requirements

RecordService VM requires VirtualBox version 4.3 or 5. You can download a free copy of VirtualBox at https://www.virtualbox.org/wiki/Downloads.

Platform and Hardware Support

RecordService supports the following software and hardware configurations when running on your own Hadoop cluster:

- CDH 5.4
- Server support: RHEL5 and RHEL6, Ubuntu LTS, SLES, and Debian
- Intel Nehalem (or later) or AMD Bulldozer (or later) processor
- 64 GB memory
- For optimal performance, run with 12 or more disks, or use SSD.

Storage and File Format Support

RecordService supports reading HDFS or S3 of the following file formats:

- Parquet
- Text
- Sequence file
- RC
- Avro

Data Type Support

RecordService supports the following data types:

- INT (8-64 bits)
- CHAR/VARCHAR
- BOOL
- FLOAT
- DOUBLE
- DECIMAL
- STRING
- TIMESTAMP

RecordService does not support the following data types:

- BLOB/CLOB
- Nested Types

Known Issues

RecordService client configurations are not properly propagated to Spark jobs

RecordService configuration options are not propagated to Spark jobs using the RecordService custom service descriptor (CSD). All configuration options must be specified in the job or through Cloudera Manager safety valves for Spark.

Workaround:

- Apply configuration options using the Spark configuration safety valve:
  Spark -> **Configuration** -> **Spark (Standalone) Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.con**

```
spark.recordservice.planner.hostports=<comma separated list of planner host:ports>
```

- If the cluster is Kerberized, also set:

```
spark.recordservice.kerberos.principal=<Kerberos principal>
```

- Save changes and deploy the client configuration.

digest-md5 library not installed on cluster, breaking delegation tokens

The digest-md5 library is not installed by default in parcel deployments.

Workaround

To install the library on RHEL 6, use the following command-line instruction:

```
sudo yum install cyrus-sasl-md5
```

Short circuit reads not enabled

Workaround

In Cloudera Manager, open the HDFS configuration page and search for *shortcircuit*. There are two configurations named **Enable HDFS Short Circuit Read**. One defaults to *true* and one to *false*. Set both values to *true*.

Path requests do not work with multiple planners

The RecordService planner creates a temporary table. The name of the table collides between RecordService planners. On a single planner, it is properly protected by a lock. On the Hive Metastore Server, collisions are likely to occur.

Workaround

For the beta release, run only one instance of the RecordService planner.

Limitations

Security Limitations

- RecordService only supports simple single-table views (no joins or aggregations).
- SSL support has not been tested.
- Oozie integration has not been tested.

Storage/File Format Limitations

- No support for write path.
- Unable to read from Kudu or HBase.

Operation and Administration Limitations

- No diagnostic bundle support.
- No metrics available in Cloudera Manager.

Application Integration Limitations

- Spark DataFrame is not well tested.

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

Download and Install RecordService

Use these instructions to install RecordService in an existing Hadoop cluster. You can also use the RecordService VM if you want to quickly install and explore the features of RecordService.

See Download and Install RecordService VM.

Download RecordService

Use these instructions to install RecordService. Cloudera recommends that you install Cloudera Manager on your cluster, so that you can download and install RecordService as an add-on service in Cloudera Manager.

Installing RecordService as an Add-on Service in Cloudera Manager

You can install RecordService as an Add-on Service in Cloudera Manager. First, install both the custom service descriptor (CSD) and the parcel in Cloudera Manager; then, add RecordService as an add-on service from the home page in Cloudera Manager.
See http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_mc_addon_services.html.

Installing the RecordService CSD

Follow these steps to install the RecordService CSD.

1. Download the CSD from http://archive.cloudera.com/beta/recordservice/csd.
2. Upload the CSD to `/opt/cloudera/csd` in the Cloudera Manager server.
3. Change the owner and group for the JAR using the following comamand-line instruction:
   `chown cloudera-scm:cloudera-scm /opt/cloudera/csd/RECORD_SERVICE-1.0.jar`
4. Update the permissions on the file using the following command-line instruction:
   `chmod 644 /opt/cloudera/csd/RECORD_SERVICE-1.0.jar`
5. Restart the Cloudera Manager server:
   1. As the root user on the Cloudera Manager server, run `service cloudera-scm-server restart`.
   2. Log in to the Cloudera Manager Admin Console and restart the Cloudera Manager Service.
6. Check whether the CSD successfully installed in `http://{cm-server}:7180/cmf/csd/refresh`. Search for the following entry:

```
{
csdName: "RECORD_SERVICE-1.0",
serviceType: "RECORD_SERVICE",
source: "/opt/cloudera/csd/RECORD_SERVICE-1.0.jar",
isInstalled: true
}
```

See http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_mc_addon_services.html.

Parcel Installation

Follow these steps to install the RecordService Parcel.

1. Go to `http://{cm-server}:7180/cmf/parcel/status`.
2. Click **Edit Settings**.
3. Add the RecordService repository url `http://archive.cloudera.com/beta/recordservice/parcels/latest` to **Remote Parcel**

Repository URLs.
4. Open the Cloudera Manager **Parcel** status page, `http://{cm-server}:7180/cmf/parcel/status`.
5. **Download** the RecordService parcel.
6. **Distribute** the parcel.
7. **Activate** the parcel. Cloudera Manager asks you to restart the entire cluster, but you only need to start/restart RecordService.

See http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_ig_parcels.html

Cloudera Manager Deployment

Follow these steps to start RecordService for a cluster from Cloudera Manager.

1. Add a service from the Cloudera Manager home page.
2. Customize role:
    - RecordService Planner and Worker: Select hosts with both the role of DN and NN.
    - RecordService Planner: Select hosts with the role of NN.
    - RecordService Worker: Select hosts with the role of DN.
3. Review and modify configuration settings, such as *log dir* and *planner port*.
    - If Sentry is enabled in the cluster, add the following configuration to the field **Configuration Snippet (Safety Valve) for sentry-site.xml**.

```
<property>
  <name>sentry.service.server.principal</name>
  <value>sentry/_HOST@principal</value>
</property>
<property>
 <name>sentry.service.security.mode</name>
 <value>kerberos</value>
</property>
<property>
 <name>sentry.service.client.server.rpc-address</name>
 <value>hostname</value>
```

```
</property>
<property>
 <name>sentry.service.client.server.rpc-port</name>
 <value>portnum</value>
</property>
<property>
 <name>hive.sentry.server</name>
 <value>server1</value>
</property>
```

4. Start the service.
5. Go to the debug page hostname:11050 to verify that the service started properly.
6. Go to the RecordService cluster page in Cloudera Manager.
7. From **Actions** choose **Deploy Client Configuration**.
8. Run the following command on the host where you run the RecordService client:

```
export HADOOP_CONF_DIR=/etc/recordservice/conf
```

See http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_mc_add_service.html?scroll=cmug_topic_5_1.

RecordService Role Deployment Recommendations

RecordService Planner: Clients (MapReduce and Spark) connect to the Planner to submit requests. The Planner performs authorization checks and generates all tasks to execute for the overall request. For the beta release, run only one Planner node located on a non-DataNode.

RecordService Worker: Read data from the underlying storage layer (HDFS/S3) and construct records. Worker roles cannot directly process client requests. Run a Worker on every DataNode in the cluster to ensure locality.

RecordService Planner Worker: Acts as both a Planner and a Worker. Provides flexibility for small clusters.

RecordService Gateway: Contains configuration information, such as the location of the RecordService Planner. Deploy to all Gateway nodes, and any other nodes that run client jobs that do not already have a RecordService Planner or Worker role deployed.

Run a Job Using RecordService

You can verify the RecordService server installation by running examples from the client JAR.

1. Download the client library and example tarball. http://archive.cloudera.com/beta/recordservice/client-dist/recordservice-client-0.1-bin.tar.gz
    - You can also build the client library yourself from the client repository. http://github.mtv.cloudera.com/CDH/RecordServiceClient.
    - Client libraries are also available directly from the Cloudera public Maven repository.
2. To verify the server installation, run client examples in your clusters.

- Log in to one of the nodes in your cluster, and load test data:

```
    > wget -q --no-clobber \
      https://s3-us-west-1.amazonaws.com/recordservice-vm/tpch.tar.gz
    > tar -xzf tpch.tar.gz
    > hadoop fs -mkdir -p /test-warehouse/tpch.nation
    > hadoop fs -put -f tpch/nation/* /test-warehouse/tpch.nation/
    > impala-shell -f create-tbls.sql
```

See https://github.com/cloudera/RecordServiceClient/blob/master/tests/create-tbls.sql.

- Run a MapReduce job for RecordCount on tpch.nation:

```
> hadoop jar /path/to/recordservice-examples-0.1.jar \
com.cloudera.recordservice.examples.mapreduce.RecordCount \
"SELECT * FROM tpch.nation" "/tmp/recordcount_output"
```

- Start spark-shell with the RecordService JAR:

```
> path/to/spark/bin/spark-shell \
--conf spark.recordservice.planner.hostports=planner_host:planner_port \
--jars /path/to/recordservice-examples-spark-0.1.jar
> scala> import com.cloudera.recordservice.spark._
import com.cloudera.recordservice.spark._
> scala> val data = sc.recordServiceRecords("select * from tpch.nation")
data: org.apache.spark.rdd.RDD[Array[org.apache.hadoop.io.Writable]] = \ RecordServiceRDD[0] at RDD at Recor
> scala> data.count()
res0: Long = 25
```

See http://github.mtv.cloudera.com/CDH/RecordServiceClient/blob/master/java/examples-spark/README.md

Package Installation (Not Recommended)

Follow these steps to install RecordService from a package.

1. Download the package from http://archive.cloudera.com/beta/recordservice.
2. Install Hadoop, Hive, Impala, Sentry, ZooKeeper, and any other application you want to use.
3. Install the RecordServicePlanner using the following command-line instruction:

```
./recordserviced -hostname=hostname -recordservice_planner_port=12050 \
-recordservice_worker_port=0 -recordservice_webserver_port=11050 \
-webserver_doc_root=path/to/package/lib/recordservice \
-log_dir=path/to/log/dir -abort_on_config_error=false \
-lineage_event_log_dir=path/to/log/dir \
-audit_event_log_dir=path/to/log/dir -profile_log_dir=path/to/log/dir \
-v=1 -mem_limit=8G
```

4. Install the worker using the following command-line instruction.

```
./recordserviced  -hostname=hostname -recordservice_planner_port=0 \
-recordservice_worker_port=13050 -recordservice_webserver_port=11050 \
-webserver_doc_root=path/to/package/lib/recordservice \
-log_dir=path/to/log/dir -abort_on_config_error=false \
-lineage_event_log_dir=path/to/log/dir \
-audit_event_log_dir=path/to/log/dir -profile_log_dir=path/to/log/dir \
-v=1 -mem_limit=8G
```

5. Install both planner and worker on one node:

```
./recordserviced -hostname=hostname -recordservice_planner_port=12050 \
-recordservice_worker_port=13050 -recordservice_webserver_port=11050 \
-webserver_doc_root=path/to/package/lib/recordservice \
-log_dir=path/to/log/dir -abort_on_config_error=false \
-lineage_event_log_dir=path/to/log/dir \
-audit_event_log_dir=path/to/log/dir -profile_log_dir=path/to/log/dir \
-v=1 -mem_limit=8G
```

6. Set additional parameters:
   - Add the following parameters if the cluster is Kerberized:
     ```
     -principal=kerberos_principle -keytab_file=the/path/to/record_service.keytab
     ```
   - Add the Sentry configuration file, if applicable:
     ```
     -sentry_config=path/to/sentry-site.xml/
     ```

RecordService, Beta

Generated on 17 November 2015

Source
Mailing list
How to contribute

▶ Examples
Release Notes

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

Configure RecordService

Client Configurations

While it should not be necessary to change the default configuration, you have the option of modifying any of these settings.

PROPERTY	DESCRIPTION	PARAMETER	VALUE	SIGNIFICANCE
FETCH_SIZE_CONF	Option for performance tuning that configures the max	recordservice.task.fetch.size		

	number of records returned when fetching results from the RecordService. If not set, server default will be used.			
MEM_LIMIT_CONF	Maximum memory the server uses per task	recordservice.task.memlimit.bytes		
RECORDS_LIMIT_CONF	Maximum number of records returned per task	recordservice.task.records.limit		
PLANNER_REQUEST_MAX_TASKS	Maximum number of tasks to generate per PlanRequest	recordservice.task.plan.maxTasks		
PLANNER_RETRY_ATTEMPTS_CONF	Maximum number of attempts to retry RecordService RPCs with Planner	recordservice.planner.retry.attempts		
PLANNER_RETRY_SLEEP_MS_CONF	Sleep between retry attempts with Planner in milliseconds	recordservice.planner.retry.sleepMs		
PLANNER_CONNECTION_TIMEOUT_MS_CONF	Timeout when	recordservice.planner.connection.timeoutMs		

	connecting to the Planner service			
PLANNER_RPC_TIMEOUT_MS_CONF	Timeout for Planner RPCs	recordservice.planner.rpc.timeoutMs		
WORKER_RETRY_ATTEMPTS_CONF	Maximum number of attempts to retry RecordService RPCs with Worker	recordservice.worker.retry.attempts		
WORKER_RETRY_SLEEP_MS_CONF	Sleep in milliseconds between retry attempts with Worker	recordservice.worker.retry.sleepMs		
WORKER_CONNECTION_TIMEOUT_MS_CONF	Timeout when connecting to the Worker service	recordservice.worker.connection.timeoutMs		
WORKER_RPC_TIMEOUT_MS_CONF	Timeout for Worker RPCs	recordservice.worker.rpc.timeoutMs		
WORKER_ENABLE_SERVER_LOGGING_CONF	Enable server logging (logging level from Log4j)	recordservice.worker.server.enableLogging		

Server Configurations

The properties listed on the Cloudera Manager RecordService Configuration page are the ones Cloudera considers the most reasonable to change. However, adjusting these values should not be necessary. Very advanced administrators might consider making minor adjustments.

Kerberos Configuration

No special configuration is required via Cloudera Manager. Enabling Kerberos on the cluster configures everything.

Sentry Table Configuration

Sentry is configured for you in the RecordService VM. This section describes how to configure Sentry in a non-VM deployment.

Prerequisite

Follow CDH documentation to install Sentry and enable it for Hive (and Impala, if applicable).

See http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/sg_sentry_service_install.html.

Configure Sentry with RecordService

1. Enable RecordService to read policy metadata from Sentry:
    1. In Cloudera Manager, navigate to the **Sentry Configuration** page.
    2. In **Admin Groups**, add the user *recordservice*.
    3. In **Allowed Connecting Users**, add the user *recordservice*.
2. Save changes.
3. Download the Cloudera Manager generated `sentry-site.xml` configuration file (same config as HiveServer2)
    - In Cloudera Manager, navigate to **Hive** -> **HiveServer2** -> **Processes** -> `sentry-site.xml`
4. Deploy configurations to cluster: copy `sentry-site.xml` to the `/etc/sentry/conf/` directory on all nodes running RecordService Planner or RecordService Planner and Worker. Nodes that are only running only the RecordService Worker role do not need this configuration option.
5. Enable Sentry for RecordService
    1. In Cloudera Manager, navigate to **RecordService Configuration**.
    2. Select the **Sentry** service.

   3. In the **Sentry Configuration File** field, enter `/etc/sentry/conf/sentry-site.xml.
6. Save changes.
7. Restart the Sentry and RecordService services.

Delegation Token Configuration

No special configuration is required with Cloudera Manager. This is enabled automatically if the cluster is kerberized.

RecordService persists state in Zookeeper, by default, under the /recordservice Zookeeper directory. If this directory is already in use, you can configure the directory with `recordservice.zookeeper.znode`. This is a Hadoop style XML configuration that you can add to the advanced service configuration snippet.

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

Download and Install RecordService VM

Downloading RecordService VM

Follow these steps to download the RecordService VM.

1. Install VirtualBox. The VM works with VirtualBox version 4.3 on Ubuntu 14.04 and VirtualBox version 5 on OSX 10.9. Download VirtualBox for free at https://www.virtualbox.org/wiki/Downloads.
2. Clone the recordservice-quickstart repository onto your local disk from https://github.com/cloudera/recordservice-quickstart.

Installing the RecordService VM

Follow these steps to install the RecordService VM.

1. In a terminal window, navigate to the root of the RSQuickstart Git repository.
2. Run the script `install.sh`.

This script downloads an `ova` file and loads it into VirtualBox. The script might ask you to enter a password, because it edits your `/etc/hosts` file to give the VM a stable IP address, `quickstart.cloudera`. When the script completes, the running VM functions as a RecordService server.

Testing Your VM Configuration

Test that the VM is running and IP forwarding is configured properly.

1. Enter the command `ssh cloudera@quickstart.cloudera`.
2. Enter the password `cloudera`.

If you cannot ssh to the VM, see Troubleshooting the VM Configuration.

Successfully connecting through ssh verifies that you can log in to the VM.

Configuring RecordService Environment Variables

1. On your host machine, navigate to the root of your RecordServiceClient repository, `$RECORD_SERVICE_HOME`.
2. Run `source config.sh`.
3. Navigate to the `recordservice-quickstart` directory.
4. Run `source vm_env.sh`.
5. Navigate to `$RECORD_SERVICE_HOME/java`.
6. Test your environment with the command `mvn test -DargLine="-Duser.name=recordservice"`

The VM is preconfigured with sample data to execute the tests. The *recordservice* user has access to the data via Sentry.

The VM is not secured with LDAP or Kerberos. If you want to change the security configuration, you can add roles in Sentry through `impala-shell`. If you have `impala-shell` on your host machine, you can connect to the VM by issuing the following command.

```
impala-shell -i quickstart.cloudera:21000 -u impala
```

You can also connect from within `impala-shell`.

```
CONNECT quickstart.cloudera:21000;
```

Troubleshooting the VM Configuration

Unable to ssh to the VM

- Ensure that the ssh daemon is running on your machine.
- Ensure that the RecordService VM is running. In your terminal, enter the following command:

```
VBoxManage list runningvms
```

You should see “rs-demo” listed as a running VM.

Verify VM is listed correctly in /etc/hosts

Check that the VM is listed correctly in your `/etc/hosts` file. If you open the file, you should see a line that lists an IP address followed by `quickstart.cloudera`. You can check the VM’s IP with the following command:

```
VBoxManage guestproperty get rs-demo /VirtualBox/GuestInfo/Net/0/V4/IP
```

Verify Known Hosts

If you’ve used a Cloudera QuickStart VM before, your known hosts file might already have an entry for `quickstart.cloudera` registered to a different key. Delete any reference to `quickstart.cloudera` from your known hosts file, which is usually found in `~/.ssh/known_hosts`.

Debugging the VM

Restarting a service

To restart a service, use the standard RHEL service model.

```
service <service-name> start|stop|restart
```

You can view all of the installed services `/etc/init.d` directory.

Debugging Via Logs

RecordService logs are in the `/var/log/recordservice` directory. You can find most service logs in the `/var/log` directory.

Dubugging Via Webpage

View the RecordService debug page on your host machine at `quickstart.cloudera:11050`.

Other Service Variables

To view the default execution environment for a service, look for its file in the `/etc/default` directory.

RecordService, Beta

Source
Mailing list
How to contribute

Examples
Release Notes

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

Application Integration

MapReduce Input Formats

RecordService is designed to integrate with existing MapReduce InputFormats. The RecordService client contains InputFormats that are common for most applications. They implement the standard MRv1 and MRv2 InputFormat API, and should be usable by any existing application or library that accepts InputFormats. The examples also contain custom InputFormats that can be built on top of the lower-level RecordService client APIs.

Unlike many standard Hadoop InputFormats, the RecordService format is independent of the underlying storage format. For example, you can use the AvroInputFormat even if the underlying data is a CSV file. The InputFormat specifies the Java object model used by the application (for example, Avro objects when using AvroInputFormat, hadoop.io.Text when using TextInputFormat, and so on).

Drop-in Replacements for TextInputFormat and AvroInputFormat

For application migration, RecordService provides implementations of TextInputFormat (com.cloudera.recordservice.mapreduce.TextInputFormat) and AvroInputFormat (com.cloudera.recordservice.avro.mapred.AvroInputFormat). These are drop-in replacements for the corresponding classes in the Hadoop

API. Several examples show that you can migrate applications by changing the import statements for those classes to RecordService package names.

Spark RDD

RecordService integrates with Spark in several ways. You can use the InputFormats discussed above through the Spark APIs (SparkContext.hadoopFile()). In some cases, RecordService achieves native Spark integration by implementing the RDD (Resilient Distributed Dataset) interface directly (as opposed to the InputFormat approach, which implements a MapReduce API that Spark can use). The SchemaRDDExample in examples-spark demonstrates this.

Spark SQL Support and DataFrames

RecordService integrates with SparkSQL (and by extension DataFrames) by implementing the SparkSQL DataSources API. RecordService supports projection and predicate pushdown.

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

RecordService Examples

RecordService Examples

You can find the source code for RecordService examples in the RecordService Client GitHub repository.

https://github.com/cloudera/RecordServiceClient/tree/master/java/examples

Instructions for running the examples are stored in the repository with the source code.

Example	**Description**
RSCat	This example shows how you can output tabular data for any dataset readable by RecordService. It demonstrates a standalone Java application built on the core libraries without using a computation framework such as MapReduce or Spark.
SumQueryBenchmark	This demonstrates running a sum over a column and pushing the scan to RecordService. It shows how you can use RecordService to accelerate scan-intensive operations.

Terasort	This is a port of the Hadoop Terasort benchmark test ported to RecordService. See README in the Terasort package for more details. This also demonstrates how to implement a custom InputFormat using the RecordService APIs.
MapredColorCount/MapreduceAgeCount/MapReduceColorCount	These examples are ported from Apache Avro. They demonstrate the steps required to port an existing Avro-based MapReduce job to use RecordService.
RecordCount/WordCount	More MapReduce applications that demonstrate some other InputFormats included in the client library, including TextInputFormat and RecordServiceInputformat. Useful for existing MapReduce jobs already using TextInputFormat.
Reading data from a View, and Enforcing Sentry Permissions (uses RecordCount)	This example demonstrates how an MR job can now read data even when the user only has permission to see part of the data in a file (table). See https://github.com/cloudera/RecordServiceClient/tree/master/java/examples#how-to-enforce-sentry-permissions-with-mapreduce
com.cloudera.recordservice.examples.avro	Unmodified from the Apache Avro examples, these utilities help you generate sample data.

RecordService Spark Examples

The following examples can be found at https://github.com/cloudera/RecordServiceClient/tree/master/java/examples-spark.

Example	**Description**
Query1/Query2	Examples that demonstrate RecordService native RDD Resilient Distributed Dataset (RDD) integration using RecordServiceRDD.
WordCount	The Hadoop WordCount example built on top of the RecordService equivalent of textFile().
SchemaRDDExample	Another example of the native RDD integration, this time using SchemaRecordServiceRDD.
TeraChecksum	This example uses hadoopFile() with the RecordService InputFormats. This is a port of the TeraChecksum MapReduce job, written in Spark.
TpcdsBenchmark	This demonstrates the SparkSQL integration running a portion of the tpcds benchmark.

DataFrameExample	An example that demonstrates DataFrames and RecordService working together.
How to use RecordService with Spark shell	Examples of using spark-shell to interact with RecordService in a variety of ways.
Reading Data from a View and Enforcing Sentry Permissions	This example demonstrates how an MR job may now read data even when the user only has permission to see part of the data in a file (table). See ReadMe.md

RecordService, Beta

Source
Mailing list
How to contribute

Examples
Release Notes

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

RecordService FAQ

Q: Where and how are permissions managed?

A: Sentry handles policy metadata. See Sentry Configuration.

In the current version of Sentry, permissions on views allow for fine-grained access control — restricting access by column and row.

Q: What kind of data can we use RecordService for?

A: Hive Metastore Tables only. Support might be added in the future for other schema sources, depending on customer demand.

Q: What happens if you try to access data controlled by RecordService without using RecordService?

A: Sentry's HDFS Sync feature ensures that the files are locked such that only users with full access to all of the values in a file (with Sentry permissions for the entire table) are allowed to read the files directly. See Configuring the Sentry Service.

Q: Do I need to be running Sentry to evaluate the RecordService?

A: No. You can deploy RecordService without Sentry, in which case no authorization checks will happen. This can be useful to evaluate functionality and performance.

Q: Are there any APIs to discover the permissions that are set?

A: Hue can show this, as well as SHOW commands in Hive or Impala CLI.

Q: What is the RecordService security model? Can you purposely restrict views into the data?

A: Yes, you can control permissions per view. Setting the active role, is not currently supported.

Q: Why does RecordService implement its own schema (which seems to be a copy of Hive's schema)?

A: The client API is layered so that it does not have to pull in all dependencies. As you move higher in the client API, you get access to more standard Hadoop objects. In this case, you get a recordservice-hive JAR that returns Hive Schema objects.

Q: Can you list tables through RecordService, or do you use the Hive metastore to get tables, and then ask RecordService for the schema?

A: You cannot list tables through RecordService. You have to use the Hive Metastore, or use a tool such as Hue.

You need only the fully qualified table name to read from a table, so the client does not need to pass the table metadata to RS.

Q: How does accessing a path directly compare to querying the Hive metastore?

A: RecordService infers the schema. If the path contains a self-describing file, such as Avro or Parquet, it uses that. For files like CSV, RecordService defaults to a STRING schema.
In a future release RecordService might infer schema from files, but the security rules for paths are still under consideration. For now, RecordService only supports reading from tables defined in the Hive Metastore.

RecordService, Beta

Source
Mailing list
How to contribute

Examples
Release Notes

Generated on 29 October 2015

An abstraction layer between storage managers (for example, HDFS, HBase, Kudu) and compute frameworks (for example, MapReduce, Spark, Hive).

Getting Involved

You are welcome to comment and contribute to the RecordService project in any of these ways.

- Mailing list: recordservice-user@googlegroups.com
- Discussion forum: http://community.cloudera.com/t5/Beta-Releases/bd-p/Beta
- Contributions: http://github.com/cloudera/RecordServiceClient/
- Documentation: http://cloudera.github.io/RecordServiceClient/
- Bug Reporting: Open Github Issue
- Server Implementation: http://github.com/cloudera/RecordService/

RecordService, Beta

Source
Mailing list
Examples

How to contribute

Release Notes

Generated on 19 October 2015