WPS Configuration for Hadoop

Version: 4.2.4

www.worldprogramming.com

Contents

Introduction

What is Hadoop?

Hadoop is a scalable, fault-tolerant, open source software framework for the distributed storage and distributed processing of very large datasets on computer clusters. It is distributed under the Apache license.

For those of you who are new to Hadoop, you are advised to refer initially to *Hadoop basics* (page 7).

Advantages afforded by the integration of Hadoop into WPS

- With Hadoop integration, WPS extends its data integration capabilities across dozens of database engines.
- Data sharing between WPS and HDFS (Hadoop Distributed File System) offers data-level interoperability between the two environments. Whilst it is not transparent, it is straightforward: Hadoop data can be imported into WPS for (structured) analysis, and, if desired, subsequently sent back to HDFS.
- WPS users can invoke Hadoop functionality from the familiar surroundings of the WPS Workbench user interface
- Users can create and edit new Hadoop operations using a SQL-like language - they do not have to know Java.

Scope of this document

This document gives an overview of the implementation of WPS and Hadoop, and also covers the configuration of Kerberos where this applies.

WPS/Hadoop integration summary

The following currently implemented integrations use `filename`, `libname` and `PROC HADOOP` extensions to WPS:

- Connect to Hive using standard SQL
- Connect to Impala using standard SQL
- Connect to Hive using passthrough SQL
- Connect to Impala using passthrough SQL
- Issue HDFS commands and execute Pig scripts

WPS' Hadoop integration has been certified against Cloudera 5 and tested against other Hadoop distributions that remain close to the Apache standard. Several code samples relating to integration (page 17) are given at the end of the document.

Prerequisites

Hadoop is a complex, multipart technology stack. Before being integrated with WPS, it needs to be installed and configured correctly. The following preparatory steps should be performed and double-checked:

1. Obtain the correct set of `.jar` files that correspond to your Hadoop installation.

   Note:

   When using Apache Hive as part of your Hadoop installation with WPS, you must use Apache Hive version 0.12 or higher.

2. Set up the configuration XML files according to your specific cluster environment (IP addresses, ports, and so on).
3. Establish whether your distribution of Hadoop includes or mandates support for Kerberos. If so, confirm that Kerberos authentication against your server works, that the principal has been correctly configured, and so on. Regardless of whether or not Kerberos is being used, please complete the remaining steps of the Prerequisites.
4. Establish that the cluster is functioning correctly, perhaps by consulting with your cluster administrator who should have access to the administrative dashboards.
5. Once it has been established that the cluster is functioning correctly, establish that Hadoop-related tasks can be submitted independently of WPS.

Kerberos

Establishing identity with strong authentication is the basis for secure access in Hadoop, with users needing to be able to identify themselves so that they can access resources, and Hadoop cluster resources needing to be individually authenticated to avoid malicious systems potentially 'posing as' part of the cluster to gain access to data. To create this secure communication among its various components, Hadoop can use Kerberos, which is a third party authentication mechanism, whereby users and services that users want to access rely on the Kerberos server to handle authentication.

Note:

Some Hadoop distributions include (or even mandate) support for Kerberos. The specifics of Kerberos server configuration often vary according to distribution type and version, and are beyond the scope of this document. Refer to the distribution-specific configuration information provided with your Hadoop software. See *Configuring Kerberos and Hadoop on the client* (page 16) for how to configure Kerberos and Hadoop on the client side.

Hadoop basics

In traditional analytics environments, data is fed into an RDBMS via an initial ETL (Extract, Transform, Load) process. Unstructured data is prepared and loaded into the database, acquiring a schema along the way. Once loaded, it becomes amenable to a host of well established analytics techniques.

For large quantities of data, however, this workflow has some problems:

1. If the time taken to process a day's data reaches a point where you cannot economically complete it prior to the following day, you need another approach. Large-scale ETL puts a massive pressure on the underlying infrastructure.
2. As data gets older, it is often eventually archived. However, it is extremely expensive to retrieve archived data in volume (from tape, blu-ray, and so on). In addition, once it has been archived, you no longer have convenient and economical access to it.
3. The ETL process is an abstraction process – data becomes aggregated and normalised and its original high-fidelity form is lost. If the business subsequently asks a new kind of question of the data, it is often not possible to provide an answer without a costly exercise which involves changing the ETL logic, fixing the database schema, and reloading.

Hadoop was designed to provide:

- Scalability over computing and data, eliminating the ETL bottleneck.
- Improved economics for keeping data alive - and on primary storage - for longer.
- The flexibility to go back and ask new questions of the original high-fidelity data.

Comparing RDBMS and Hadoop

From an analytics perspective, the main differences between an RDBMS and Hadoop are as shown below.

Table 1. Key differences between RDBMS and Hadoop

RDBMS	Hadoop
The schema must be created before any data can be loaded.	Data is simply copied to the file store, and no transformation is needed.
An explicit ETL operation has to take place, transforming the data to the database's own internal structure.	A serialiser/deserialiser is applied at read time to extract the required columns.
New columns must be added explicitly before new data for such columns can be loaded.	New data can start flowing at any time, and will appear retrospectively once the serialiser/deserialiser is updated to parse it.

The schema-orientation of conventional RDBMS implementations provide some key benefits that have led to their widespread adoption:

- Optimisations, indexes, partitioning, and so on, become possible, allowing very fast reads for certain operations such as joins, multi-table joins, and so on.
- A common, organisation-wide schema means that different groups in a company can talk to each other using a common vocabulary.

On the other hand, RDBMS implementations lose out when it comes to flexibility – the ability to grow data at the speed at which it is evolving. With Hadoop, structure is only imposed on the data at read time, via a serialiser/deserialiser, and consequently, there is no ETL phase – files are simply copied into the system. Fundamentally, Hadoop is not a conventional database in the normal sense, given its ACID (Atomicity, Consistency, Isolation, Durability) properties, and even if it were, it would probably be too slow to drive most interactive applications.

Both technologies can augment each other, both can have a place in the IT organisation - it is simply a matter of choosing the right tool for the right job.

Table 2. RDBMS vs Hadoop: Key use cases

When to use RDBMS	When to use Hadoop
Interactive OLAP - sub-second response time.	When you need to manage both structured and unstructured data.
When you need to support multi-step ACID transactions on record-based data (e.g. ATMs, etc).	When scalability of storage and/or compute is required.
When 100% SQL compliance is required.	When you have complex data processing needs with very large volumes of data.

Hadoop architecture

Two key concepts lie at the core of Hadoop:

- The HDFS (Hadoop Distributed File System) – a Java-based file system that provides scalable and reliable data storage spanning large clusters of commodity servers.
- MapReduce – a programming model that simplifies the task of writing programs that work in a parallel computing environment.

An operational Hadoop cluster has many other sub-systems, but HDFS and MapReduce are central to the processing model.

HDFS

HDFS is a distributed, scalable and portable file system written in Java. HDFS stores large files (typically in the range of gigabytes to terabytes) across multiple machines. It achieves reliability by replicating the data across multiple hosts. By default, data blocks are stored (replicated) on three nodes – two on the same rack and one on a different rack (a 3X overhead compared to non-replicated storage). Data nodes can talk to each other to rebalance data, move copies around, and keep the replication of data high.

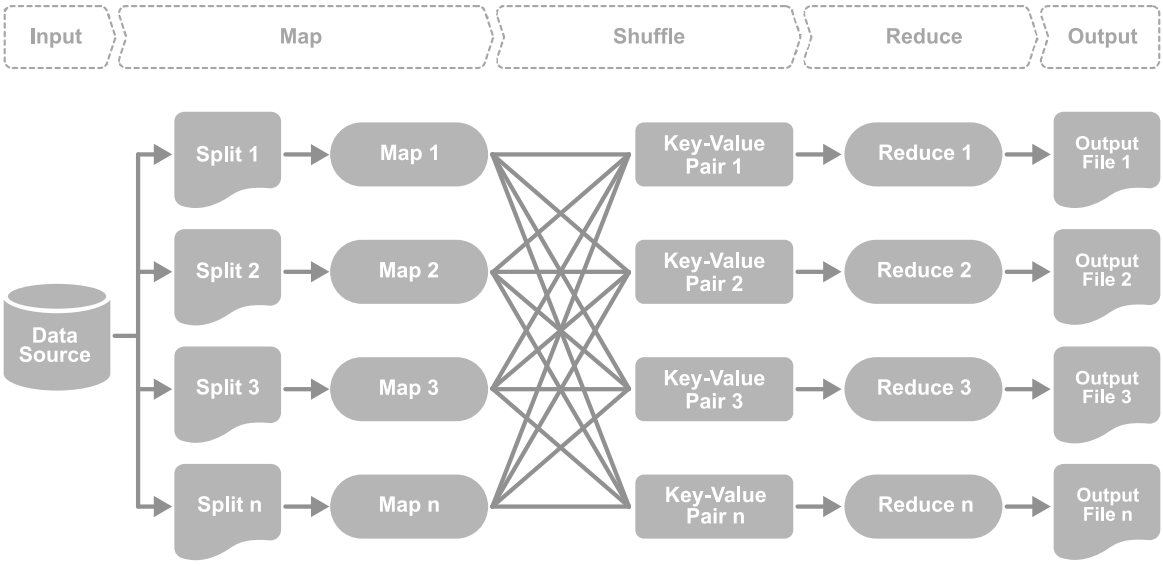
HDFS is not a fully Posix-compliant file system and is optimised for throughput. Certain atomic file operations are either prohibited or slow. You cannot, for example, insert new data in the middle of a file, although you can append it.

MapReduce

MapReduce is a programming framework which, if followed, removes complexity from the task of programming in massively parallel environments.

A programmer typically has to write two functions – a Map function and a Reduce function – and other components in the Hadoop framework will take care of fault tolerance, distribution, aggregation, sorting, and so on. The usually cited example is the problem of producing an aggregated word frequency count across a large number of documents. The following steps are used:

1. The system splits the input data between a number of nodes called mappers. Here, the programmer writes a function that counts each word in a file and how many times they occur. This is the Map function, the output of which is a set of key-value pairs that include a word and word count. Each mapper does this to its own set of input documents, so that, in aggregate, many mappers produce many sets of key-value pairs for the next stage.
2. The shuffle phase occurs – a consistent hashing function is applied to the key-value pairs and the output data is redistributed to the reducers, in such a way that all key-value pairs with the same key go to the same reducer.
3. The programmer has written a Reduce function that, in this case, simply sums the word occurrences from the incoming streams of key-value pairs, writing the totals to an output file:



This process isolates the programmer from scalability issues as the cluster grows. Part of the Hadoop system itself looks after the marshalling and execution of resources - this part is YARN if the version of MapReduce is 2.0 or greater.

There is no guarantee that this whole process is faster than any kind of alternative system (although, in practice, it is faster for certain kinds of problem sets and large volumes of data). The main benefit of this programming model is the ability to exploit the often-optimised shuffle operation while only having to write the map and reduce parts of the program.

The Hadoop ecosystem

There are several ways to interact with an Hadoop cluster.

Java MapReduce

This is the most flexible and best-performing access method, although, given that this is the assembly language of Hadoop, there can be an involved development cycle.

Streaming MapReduce

This allows development in Hadoop in any chosen programming language, at the cost of slight to modest reductions in performance and flexibility. It still depends upon the MapReduce model, but it expands the set of available programming languages.

Crunch

This is a library for multi-stage MapReduce pipelines in Java, modelled on Google's FlumeJava. It offers a Java API for tasks such as joining and data aggregation that are tedious to implement on plain MapReduce.

Pig Latin

A high-level language (often referred to as just 'Pig') which is suitable for batch data flow workloads. With Pig, there is no need to think in terms of MapReduce at all. It opens up the system to non-Java programmers and provides common operations such as join, group, filter and sort.

Hive

A (non-compliant) SQL interpreter which includes a metastore that can map files to their schemas and associated serialisers/deserialisers. Because Hive is SQL-based, ODBC and JDBC drivers enable access to standard business intelligence tools such as Excel.

Oozie

A PDL XML workflow engine that enables you to create a workflow of jobs composed of any of the above.

HBase

Apache HBase is targeted at the hosting of very large tables - billions of rows and millions of columns - atop clusters of commodity hardware. Modelled on Google's Bigtable, HBase provides Bigtable-like capabilities on top of Hadoop and HDFS.

Zookeeper

Apache Zookeeper is an effort to develop and maintain an open-source server which enables highly reliable distributed co-ordination. Zookeeper is a centralised service for maintaining configuration information and naming, and for providing distributed synchronisation and group services.

Implementing WPS and Hadoop on Windows x64

Installing WPS on Windows x64

1. Before starting to install WPS, ensure that your copy of Windows has the latest updates and service packs applied.
2. Windows workstation and server installations both use the same WPS software - usage is controlled by means of a license key applied using the `setinit` procedure.
3. The WPS installation file for Windows can be downloaded from the World Programming website. You will require a username and password to access the download section of the site.
4. Once the installation (.msi) file is downloaded, you simply double-click the file, read and accept the EULA, and follow the on-screen instructions.
5. Once the WPS software is installed, you will need to apply the license key. The license key will have been emailed to you when you purchased the WPS software. The easiest way to apply the license key is by running the WPS Workbench as a user with administrative access to the system, and following the instructions.
6. This concludes WPS configuration.

Configuring Hadoop on Windows x64

Installing Hadoop

If you have not already done so, please install Hadoop, referring as required to the documentation supplied for your particular distribution (for example, Cloudera). Once Hadoop has been installed, you should proceed with the configuration details outlined below.

Note:

Provided that you have a distribution which works in the standard Apache Hadoop way, then the configuration details should apply, even if your distribution is not Cloudera. Distributions that switch off or change the standard Apache Hadoop features are not supported.

Configuration files

All calls to Cloudera 5 Hadoop are done via Java and JNI. The Cloudera Hadoop client `.jar` files will need to be obtained and downloaded to the local machine. The following files contain URLs for various Hadoop services and need to be configured to match the current Hadoop installation:

- core-site.xml
- hdfs-site.xml
- mapred-site.xml

Note:

If you are using a Windows client against a Linux cluster, this last file needs to set the configuration parameter `mapreduce.app-submission.cross-platform` to be `true`.

Please refer to the Hadoop documentation for more information.

The CLASSPATH environment variable

The environment variable `CLASSPATH` needs to be set up to point to the Cloudera Java client files. This will vary depending on your client configuration and specific machine, but a fictitious example might resemble:

```
c:\Cloudera5\conf;c:\Cloudera5\*.jar
```

The HADOOP_HOME environment variable

On Windows, the environment variable `HADOOP_HOME` needs to be set up to point to the Cloudera Java client files. For the example above, it should be set to: `C:\Cloudera5`.

Configuring Kerberos on Windows x64

If your distribution of Hadoop includes or mandates support for Kerberos, please proceed to *Configuring Kerberos and Hadoop on the client* (page 16).

Implementing WPS and Hadoop on Linux x64

Installing WPS on Linux x64

1. WPS is supported on any distribution of Linux that is compliant with LSB (Linux Standard Base) 3.0 or above. WPS is supported on Linux running on x86 x86_64 and IBM System z, including IFL (Integrated Facility for Linux).
2. If you have a 64-bit linux distribution installed, you have the choice of using 32- or 64- bit WPS. It should be noted that some 64-bit linux distributions only install 64-bit system libraries by default. Using 64-bit WPS on these distributions will work out of the box. However, if you choose to use 32-bit WPS, you will first need to install the 32-bit system libraries. Please consult your Linux distribution documentation for directions on how to accomplish this.
3. WPS for Linux is currently available as a compressed tar archive file only. A native RPM-based platform installer will be made available in future.
4. The WPS archive for Linux is supplied in gzipped tar (`.tar.gz`) format and can be downloaded from the World Programming website. You will require a username and password to access the download section of the site.
5. To install WPS, extract the files from the archive using gunzip and tar as follows. Choose a suitable installation location to which you have write access and change (`cd`) to that directory. The archive is completely self-contained and can be unpacked anywhere. The installation location can be somewhere that requires root access, such as `/usr/local` if installing for all users, or it can be in your home directory.
6. Unzip and untar the installation file by typing: `tar -xzof <wps-installation-file>.tar.gz` or: `gzip -cd <wps-installation-file>.tar.gz | tar xvf -`
7. You will need a license key in order to run WPS. It can be applied either from the graphical user interface or from the command line by launching either application as follows.
   a. To launch the WPS Workbench Graphical User Interface issue the following command: `<wps-@product-version-full-short@-installation-dir>/eclipse/workbench`. The system will open a dialog box where you can import your license key.
   b. To launch WPS from the command line, issue the following command: `<wps-@product-version-full-short@-installation-dir>/bin/wps -stdio -setinit < <wps-key-file>`. A message will confirm that the license had been applied successfully.
8. This concludes WPS configuration.

Configuring Hadoop on Linux x64

Installing Hadoop

If you have not already done so, please install Hadoop, referring as required to the documentation supplied for your particular distribution (for example, Cloudera). Once Hadoop has been installed, you should proceed with the configuration details outlined below.

Note:

Provided that you have a distribution which works in the standard Apache Hadoop way, then the configuration details should apply, even if your distribution is not Cloudera. Distributions that switch off or change the standard Apache Hadoop features are not supported.

Configuration files

All calls to Cloudera 5 Hadoop are done via Java and JNI. The Cloudera Hadoop client `.jar` files will need to be obtained and downloaded to the local machine. The following files contain URLs for various Hadoop services and need to be configured to match the current Hadoop installation:

- core-site.xml
- hdfs-site.xml
- mapred-site.xml

Please refer to the Hadoop documentation for more information.

The CLASSPATH environment variable

The environment variable `CLASSPATH` needs to be set up to point to the Cloudera Java client files. For a fictitious example, the following lines might be added to the user profile (such as `.bash_profile`):

```
CLASSPATH=/opt/cloudera5/conf:/opt/cloudera5/*.jar
```

```
EXPORT CLASSPATH
```

Configuring Kerberos on Linux x64

If your distribution of Hadoop includes or mandates support for Kerberos, please proceed to *Configuring Kerberos and Hadoop on the client* (page 16).

Configuring Kerberos and Hadoop on the client

On both Windows and Linux, you may need to run the `kinit` command first, and enter your password at the prompt. This may be either the OS implementation of `kinit` (on Linux) or the `kinit` binary in the JRE directory within WPS.

On Windows:

- You need to be logged on as an active directory user, not a local machine user
- Your user can not be a local administrator on the machine
- You need to set a registry key to enable Windows to allow Java access to the TGT session key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos\Parameters
Value Name: allowtgtsessionkey
Value Type: REG_DWORD
Value: 0x01
```

- The JCE (Java Cryptography Extension) Unlimited Strength Jurisdicton Policy files need to be installed in your JRE (that is to say, the JRE within the WPS installation directory).

You will then need to set up the various Kerberos principals in the Hadoop XML configuration files. With Cloudera, these are available via Cloudera Manager. The list of configuration files includes:

- `dfs.namenode.kerberos.principal`
- `dfs.namenode.kerberos.internal.spnego.principal`
- `dfs.datanode.kerberos.principal`
- `yarn.resourcemanager.principal`
- `yarn.resourcemanager.principal`

Note:

The above list is not exhaustive and can often be site-specific: `libname` declarations further require that the `hive_principal` parameter is set to the `hive_principal` of the Kerberos cluster.

Code samples relating to integration

Connecting to Hive using standard SQL

```
libname lib hadoop schema=default server="clouderademo" user=demo
password=demo;

proc sql;
 drop table lib.people;
run;

data people1;
 infile 'd:\testdata.csv' dlm=',' dsd;
 input id $ hair $ eyes $ sex $ age dob :date9. tob :time8.;
run;

proc print data=people1;
 format dob mmddyy8. tob time8.;
run;

data lib.people;
  set people1;
run;

data people2;
  set lib.people;
run;

proc contents data=people2;
run;

proc print data=people2;
 format dob mmddyy8. tob time8.;
run;

proc means data=lib.people;
 by hair;
  where hair = 'Black';
run;
```

Connecting to Impala using standard SQL

```
libname lib hadoop schema=default server="clouderademo" user=demo
password=demo port=21050 hive_principal=nosasl;

proc sql;
 drop table lib.peopleimpala;
run;
```

```
data people1;
 infile 'd:\testdata.csv' dlm=',' dsd;
 input id $ hair $ eyes $ sex $ age dob :date9. tob :time8.;
run;

proc print data=people1;
 format dob mmddyy8. tob time8.;
run;

data lib.peopleimpala;
  set people1;
run;

data people2;
  set lib.peopleimpala;
run;

proc contents data=people2;
run;

proc print data=people2;
 format dob mmddyy8. tob time8.;
run;

proc means data=lib.peopleimpala;
 by hair;
  where hair = 'Black';
run;
```

Connecting to Hive using passthrough SQL

```
proc sql;
connect to hadoop as lib (schema=default server="clouderademo" user=demo
 password=demo);
    execute (create database if not exists mydb) by lib;
    execute (drop table if exists mydb.peopledata) by lib;
    execute (CREATE EXTERNAL TABLE mydb.peopledata(id STRING, hair STRING, eye
 STRING, sex STRING, age INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES
 TERMINATED BY '\n' STORED AS TEXTFILE LOCATION '/user/demo/test') by lib;
 select * from connection to lib (select * from mydb.peopledata);
  disconnect from lib;
quit;

/* options sastrace=,,,d; */
libname lib2 hadoop schema=mydb server="clouderademo" user=demo password=demo;
data mypeopledata;
  set lib2.peopledata;
run;

proc print data=mypeopledata;
run;
```

Connecting to Impala using passthrough SQL

```
proc sql;
  connect to hadoop as lib (schema=default server="clouderademo" user=demo
 password=demo port=21050 hive_principal=nosasl);
    execute (create database if not exists mydb) by lib;
```

```
    execute (drop table if exists mydb.peopledataimpala) by lib;
    execute (CREATE EXTERNAL TABLE mydb.peopledataimpala(id STRING, hair STRING, eye
 STRING, sex STRING, age INT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LINES
 TERMINATED BY '\n' STORED AS TEXTFILE LOCATION '/user/demo/test') by lib;
 select * from connection to lib (select * from mydb.peopledataimpala);
  disconnect from lib;
quit;

libname lib2 hadoop schema=mydb server="clouderademo" user=demo password=demo
 port=21050 hive_principal=nosasl;
data mypeopledata;
  set lib2.peopledataimpala;
run;

proc print data=mypeopledata;
run;
```

Execution of HDFS commands and Pig scripts via WPS

Example WPS code

```
filename script 'd:\pig.txt';
proc hadoop options='d:\hadoop.xml' username = 'hdfs' verbose;
       hdfs delete='/user/demo/testdataout' recursive;
run;

proc hadoop options='d:\hadoop.xml' username = 'demo' verbose;
       pig code = script;
run;

proc hadoop options='d:\hadoop.xml' username = 'demo';
       hdfs copytolocal='/user/demo/testdataout/part-r-00000' out='d:\output.txt'
 overwrite;
run;

data output;
 infile "d:\output.txt" delimiter='09'x;
 input field1 field2 $;
run;

proc print data=output;
run;
```

Example Pig code

```
input_lines = LOAD '/user/demo/test/testdata.csv' AS (line:chararray);
-- Extract words from each line and put them into a pig bag
-- datatype, then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;

-- filter out any words that are just white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';

-- create a group for each word
word_groups = GROUP filtered_words BY word;

-- count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT(filtered_words) AS count, group AS
 word;
```

```
-- order the records by count
ordered_word_count = ORDER word_count BY count DESC;
STORE ordered_word_count INTO '/user/demo/testdataout';
```

Using WPS with Hadoop Streaming

Hadoop streaming is a utility that comes with the Hadoop distribution. The utility allows you to create and run MapReduce jobs with any executable or script as a mapper and/or a reducer.

The outline syntax is as follows:

```
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \
    -input myInputDirs \
    -output myOutputDir \
    -mapper /bin/cat \
    -reducer /bin/wc
```

Mappers and reducers receive their input and output on `stdin` and `stdout`. The data view is line-oriented and each line is processed as a key-value pair separated by the 'tab' character.

You can use Hadoop streaming to harness the power of WPS in order to distribute programs written in the language of SAS across many computers in a Hadoop cluster, as in the example MapReduce job given below.

Note:

Because of the wide distribution of programs, any example necessarily uses a non-traditional approach for the language of SAS, in that each mapper and reducer only sees a limited subset of the data.

Before proceeding, ensure that you are familiar with the HDFS and MapReduce concepts in *Hadoop architecture* (page 8).

The following example shows the creation and running of a MapReduce job to produce counts of words that appear in the text files in the directory that is provided as input to the MapReduce job. Each individual count of a word is processed as the key-value pair `<word><tab>1`.

1. Ensure that the `input` directory has been set up on the HDFS, for example using `hadoop fs -mkdir /user/rw/input`, and that the text files containing the words to be counted have been added to the directory. Each cluster can see this directory.
2. Ensure that WPS has been installed in the same location on each node of the cluster, so that it can be called by any of the mappers and reducers.

3. Create a mapper program called **map.sas**:

```
options nonotes;

data map;
 infile stdin firstobs=1 lrecl=32767 encoding='utf8' missover dsd;
 informat line $32767.;
 input line;
 do i=1 by 1 while(scan(line, i, ' ') ^= '');
  key = scan(line, i,' ');
  value = 1;
  drop i line;
  output;
 end;
run;

proc export data=map outfile=stdout dbms=tab replace;
 putnames=no;
run;
```

4. Create a script called **map.sh** to call **map.sas**:

```
#!/bin/bash
/opt/wps/bin/wps /home/rw/map.sas
```

5. Create a reducer program called **reduce.sas**:

```
options nonotes;

data reduce;
 infile stdin delimiter='09'x firstobs=1 lrecl=32767 missover dsd;
 informat key $45.;
 informat value best12.;
 input key value;
run;

proc sql;
 create table result as select key as word, sum(value) as total from reduce
 group by key order by total desc;
quit;

proc export data=result outfile=stdout dbms=tab replace;
 putnames=no;
run;
```

6. Create a script called **reduce.sh** to call **reduce.sas**:

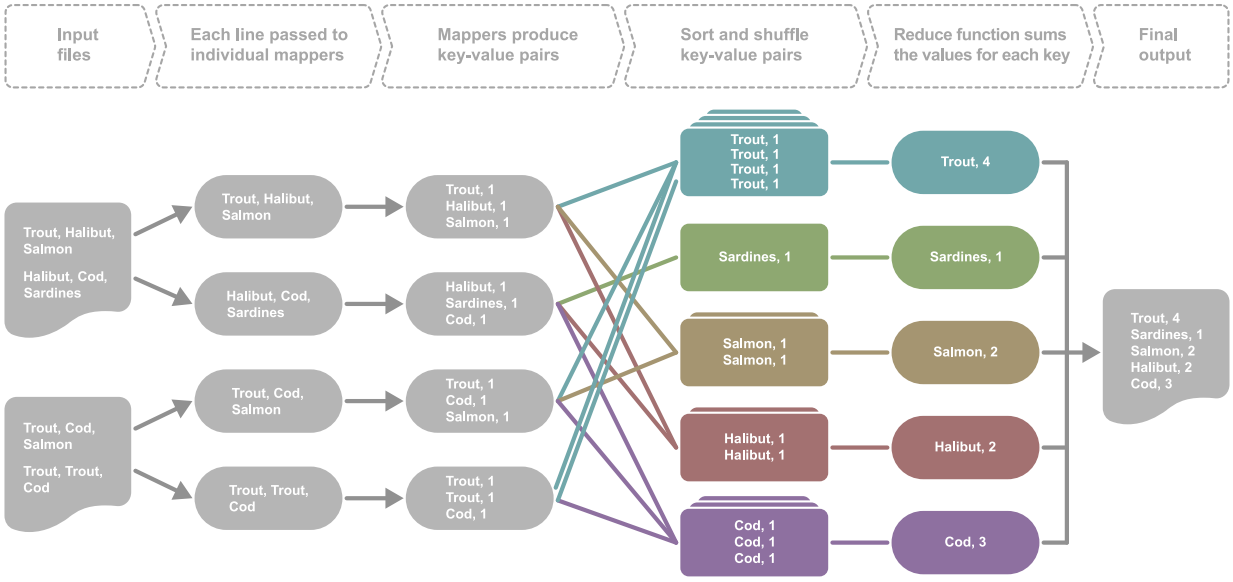
```
#!/bin/bash
/opt/wps/bin/wps /home/rw/reduce.sas
```

7. Ensure that **map.sh**, **map.sas**, **reduce.sh** and **reduce.sas** are copied to the same location of each node of the cluster, so that the mappers and reducers can run whenever required.
8. Ensure that the environment variable `CLASSPATH` has been set up on the client machine for your operating system, in accordance with *Configuring Hadoop on Windows x64* (page 12) or *Configuring Hadoop on Linux x64* (page 15).

9. Run the following command line from a client machine with a Hadoop client installed, adjusting the version numbers as appropriate:

```
hadoop jar hadoop-streaming-2.5.0-cdh5.3.2.jar -input input -output output -
mapper "/home/rw/map.sh" -reducer "/home/rw/reduce.sh"
```

Running the command has the effect of launching the MapReduce job on the particular cluster. Each instance of a mapper (where this is the **map.sh** script on a particular node invoking **map.sas**) produces a set of key-value pairs that each consist of a word and a count of 1. The shuffle phase then occurs with the key-value pairs with the same key going to the same reducer. Each instance of a reducer (where this is the **reduce.sh** script on a particular node invoking **reduce.sas**) sums the word occurrences for its particular key to an output file. The resulting output is a series of words and associated counts. This can be illustrated as follows:



Note:

The final output may end up split into more than one file inside the output directory, depending on the cluster configuration.

Reference

Railroad syntax diagrams are notations that help to explain the syntax of programming languages, and they are used in this guide to describe the language syntax.

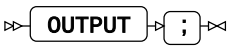
How to read railroad syntax diagrams

Railroad diagrams are a graphical syntax notation that accompanies significant language structures such as procedures, statements and so on.

The description of each language concept commences with its syntax diagram.

Entering text

Text that should be entered exactly as displayed is shown in a typewriter font :



This example describes a fragment of syntax in which the keyword `OUTPUT` is followed by a semi-colon character: `;`. The syntax diagram form is: 

.

Generally the case of the text is not significant, but in this reference, it is the convention to use upper-case for keywords.

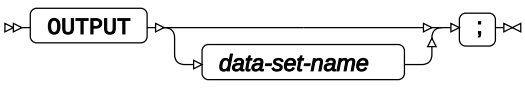
Placeholder items

Placeholders that should be substituted with relevant, context-dependent text are rendered in a lower-case, italic font :

Here, the keyword `OUTPUT` should be entered literally, but *data-set-name* should be replaced by something appropriate to the program – in this case, the name of a dataset to add an observation to.

Optionality

When items are optional, they appear on a branch below the main line in railroad diagrams. Optionality is represented by an alternative unimpeded path through the diagram:

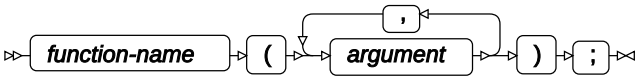


Repetition

In syntax diagrams, repetition is depicted with a return loop that optionally specifies the separator that should be placed between multiple instances.

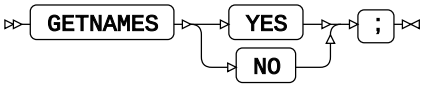
Above, the keyword `OUTPUT` should be entered literally, and it should be followed by one or more repetitions of *data-set-name* - in this case, no separator other than a space has been asked for.

The example below shows the use of a separator.



Choices

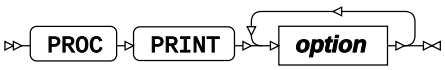
In syntax diagrams, the choice is shown by several parallel branches.



In the above example, the keyword `GETNAMES` should be entered literally, and then either the keyword `YES` or the keyword `NO`.

Fragments

When the syntax is too complicated to fit in one definition, it might be broken into fragments:



option

Above, the whole syntax is split into separate syntax diagram fragments. The first indicates that `PROC PRINT` should be followed by one or more instances of an *option*, each of which must adhere to the syntax given in the second diagram.

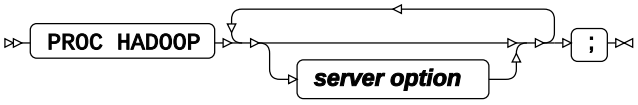
HADOOP Procedure

Supported statements

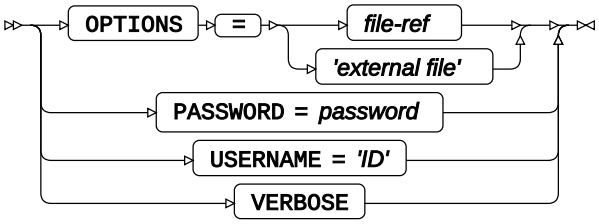
- *PROC HADOOP* (page 26)
- *HDFS* (page 26)
- *MAPREDUCE* (page 27)
- *PIG* (page 28)

PROC HADOOP

Accesses Hadoop through WPS.

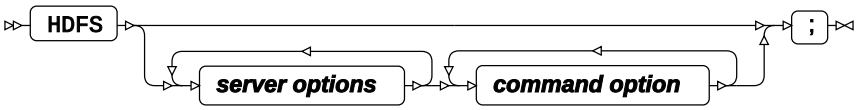


server option

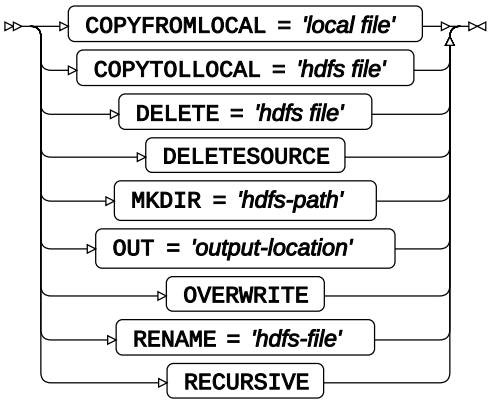


HDFS

Specifies the Hadoop distributed file system to use.

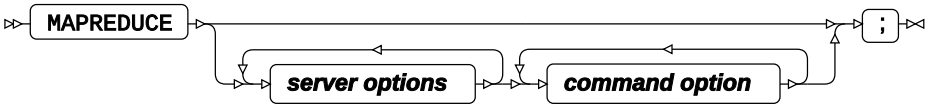


command option

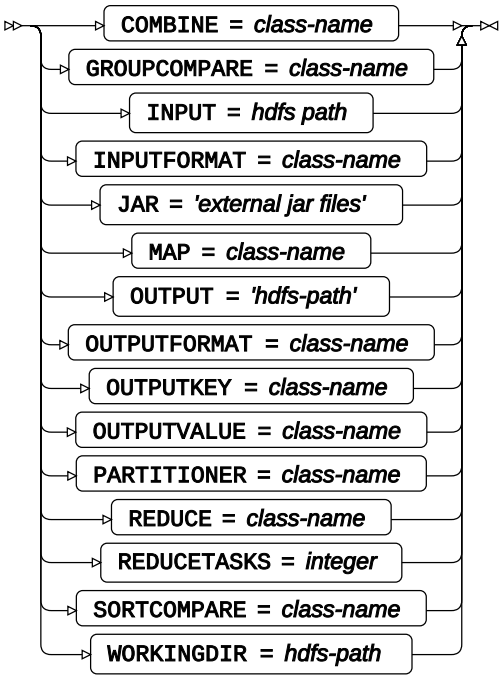


MAPREDUCE

Launches MapReduce jobs.

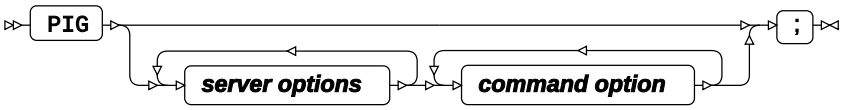


command option

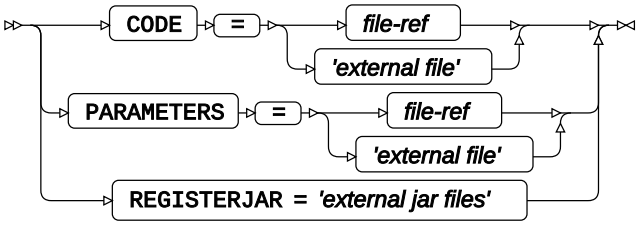


PIG

Enables external files to be submitted to a cluster.



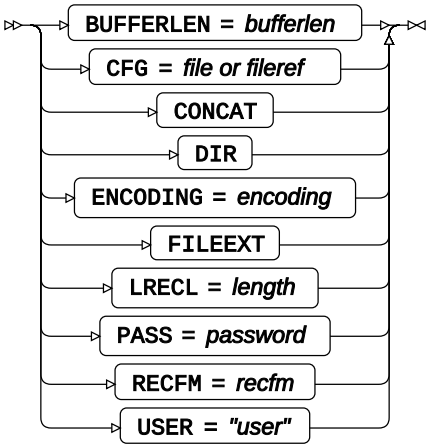
command option



Global Statements

FILENAME, HADOOP Access Method

hadoop-option



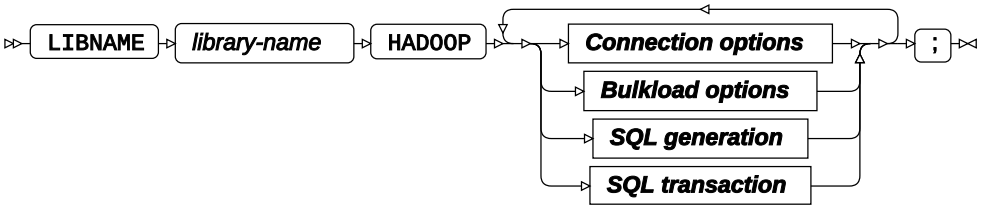
WPS Engine for Hadoop

The WPS Engine for Hadoop provides connectivity to a Hadoop database.

You can connect a SAS language program to a database server by specifying an engine name to the LIBNAME library connection statement. The engine name for the library connection statement is HADOOP. You can specify options to the LIBNAME statement that determine how SAS language programs interact with the database.

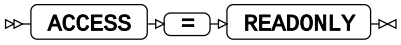
The supported LIBNAME statement options are listed in this reference.

HADOOP

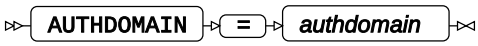


Connection options

ACCESS

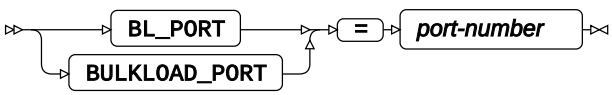


AUTHDOMAIN



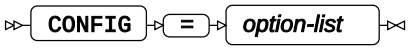
Type: String

BL_PORT



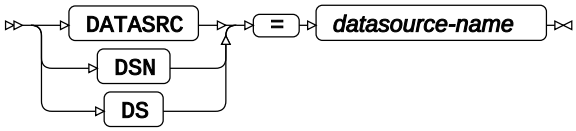
Type: Numeric

CONFIG



Type: String

DATASRC

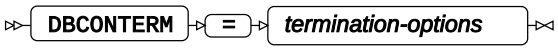


Type: String

DBCONINIT

Type: String

DBCONTERM

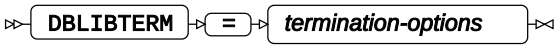


Type: String

DBLIBINIT

Type: String

DBLIBTERM



Type: String

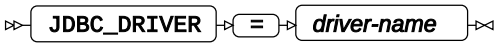
HIVE_PRINCIPAL

Type: String

JDBC_CONNECTION_STRING

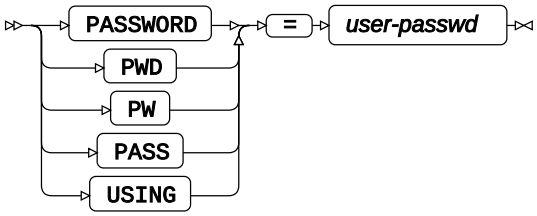
Type: String

JDBC_DRIVER



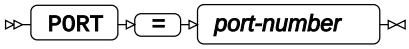
Type: String

PASSWORD



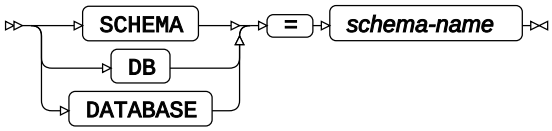
Type: String

PORT



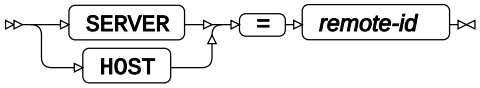
Type: Numeric

SCHEMA



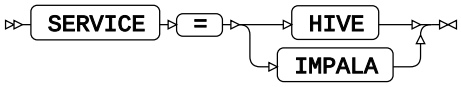
Type: String

SERVER

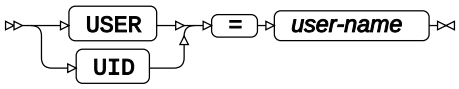


Type: String

SERVICE



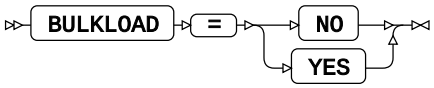
USER



Type: String

Bulkload options

BULKLOAD

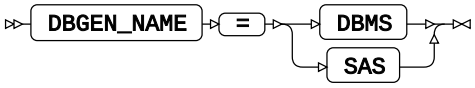


SQL generation

DBCREATE_TABLE_OPTS

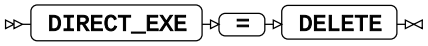
Type: String

DBGEN_NAME

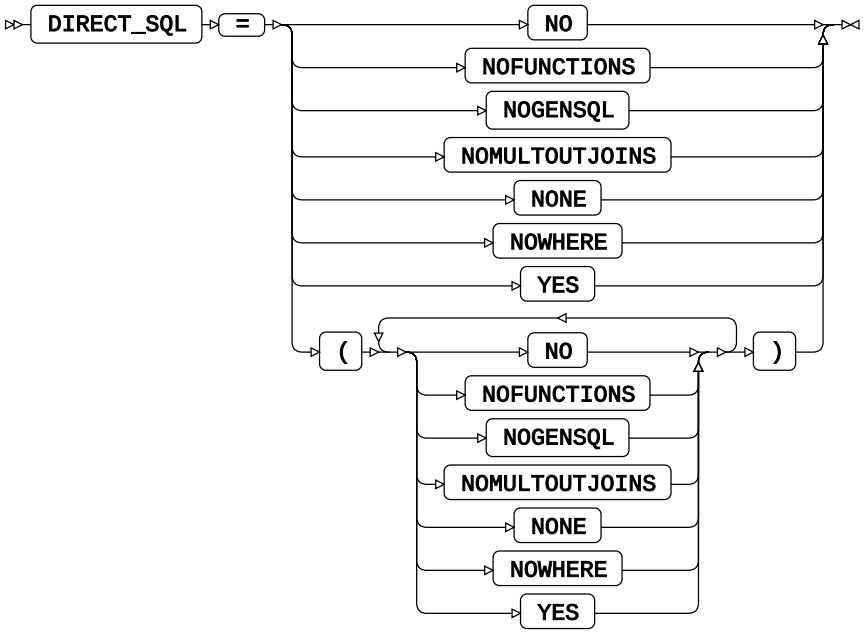


Default value: `SAS`

DIRECT_EXE

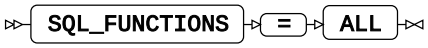


DIRECT_SQL

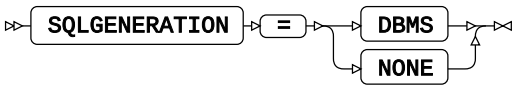


Default value: `YES`

SQL_FUNCTIONS

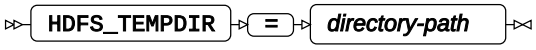


SQLGENERATION



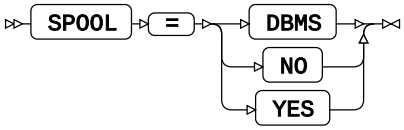
SQL transaction

HDFS_TEMPDIR



Type: String

SPOOL



Legal Notices

Trademarks

WPS and World Programming are registered trademarks or trademarks of World Programming Limited in the European Union and other countries. (r) or ® indicates a Community trademark.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

All other trademarks are the property of their respective owner.

General Notices

World Programming Limited is not associated in any way with the SAS Institute.

WPS is not the SAS System.

The phrases "SAS", "SAS language", and "language of SAS" used in this document are used to refer to the computer programming language often referred to in any of these ways.

The phrases "program", "SAS program", and "SAS language program" used in this document are used to refer to programs written in the SAS language. These may also be referred to as "scripts", "SAS scripts", or "SAS language scripts".

The phrases "IML", "IML language", "IML syntax", "Interactive Matrix Language", and "language of IML" used in this document are used to refer to the computer programming language often referred to in any of these ways.

WPS includes software developed by third parties. More information can be found in the THANKS or acknowledgments.txt file included in the WPS installation.