THE POWER TO KNOW.

SAS® Scalable Performance Data Server 5.3: Processing Data in Hadoop

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS® Scalable Performance Data Server 5.3: Processing Data in Hadoop*. Cary, NC: SAS Institute Inc.

SAS® Scalable Performance Data Server 5.3: Processing Data in Hadoop

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

October 2016

P1:spdshadoopug

Contents

What's New in SPD Server 5.3 with Hadoop

Overview

SPD Server 5.3 now runs on the Microsoft Windows x64 operating system, has expanded the supported Hadoop distributions, supports distributed locking, supports parallel processing for Write operations, and has expanded optimized WHERE processing functionality.

In addition, the documentation for both users and administrators of SPD Server with Hadoop is now contained in one document.

Operating System Support

To use SPD Server with a Hadoop cluster, SPD Server 5.3 now runs on the Microsoft Windows x64 operating system as well as the Linux x64 operating system. In addition, when SPD Server is running as a Microsoft Windows service, you can specify a user ID on the Hadoop cluster. For more information, see "Running SPD Server as a Microsoft Windows Service" on page 22.

Hadoop Distribution Support

SPD Server 5.3 has expanded the supported Hadoop distributions. For the list of supported Hadoop distributions see "What Is Required to Run SPD Server with Hadoop?" on page 2.

SPD Server Distributed Locking

With SPD Server 5.3, you can request distributed locking for data stored in HDFS. Distributed locking provides synchronization and group coordination services to clients over a network connection. To request distributed locking, you must include the new ZKPR_PORT= and ZKPR_QUORUM= parameter file options in the spdsserv.parm parameter file. In addition, there are several parameter file options to change default values. For more information, see "SPD Server Distributed Locking" on page 19.

Improved Performance of Writing Data to HDFS

SPD Server 5.3 supports parallel processing for all Write operations in addition to Read operations in HDFS. For more information, see "Parallel Processing for Data in HDFS" on page 21.

Optimized WHERE Processing

Optimized WHERE processing for SPD Server 5.3 now supports SPD Server dynamic cluster tables. For more information, see "WHERE Processing Optimization with MapReduce" on page 26.

Chapter 1

Introduction to Using SPD Server with Hadoop

Understanding SPD Server with Hadoop

SPD Server provides a multi-user, high-performance data delivery environment that enables you to interact with Hadoop through the Hadoop Distributed File System (HDFS). Using SPD Server with SAS applications, you can read, write, and update tables in HDFS.

SPD Server provides the following benefits when used with Hadoop:

- SPD Server organizes data into a streamlined file format that has advantages for a distributed file system like HDFS. Data is separate from the metadata, and the file format partitions the data.
- SPD Server supports parallel processing. The server reads data stored in HDFS by running multiple threads in parallel.
- SPD Server provides a multi-user environment.
- SPD Server is a full 64-bit server that supports up to two billion columns and (for all practical purposes) unlimited rows of data. SPD Server tables are stored on disk in a format that enhances access and supports any large table requirements for SAS 9.4. SPD Server cluster tables are a unique design feature of SPD Server that further enhances managing large tables by enabling the user to create a virtual table that consists of several SPD Server tables.
- SPD Server uses access control lists (ACLs) and SPD Server user IDs to secure domain resources.
- If the Hadoop cluster supports Kerberos, SPD Server honors Kerberos ticket, cache-based, logon authentication and authorization as long as the Hadoop cluster configuration files are accessible.

Understanding Apache Hadoop

Apache Hadoop is an open-source software framework that provides massive data storage and distributed processing of large amounts of data. The Hadoop framework provides the tools needed to develop and run software applications.

Hadoop runs on a Linux operating system. Hadoop is available from either the Apache Software Foundation or from vendors that offer their own commercial Hadoop distributions such as Cloudera and Hortonworks.

You use Hadoop to store and process big data in a distributed fashion on large clusters of commodity hardware. Hadoop is an attractive technology for a number of reasons:

- Hadoop provides a low-cost alternative for data storage.
- HDFS is well suited for distributed storage and processing using commodity hardware. It is fault-tolerant, scalable, and simple to expand. HDFS manages files as blocks of equal size, which are replicated across the machines in a Hadoop cluster to provide fault tolerance.
- Data replication provides failure resiliency.

Hadoop consists of a family of related components that are referred to as the Hadoop ecosystem. Hadoop provides many components such as the core components HDFS and MapReduce. In addition, Hadoop software and services providers (such as Cloudera and Hortonworks) provide additional proprietary software.

What Is Required to Run SPD Server with Hadoop?

- You must license the SAS Scalable Performance Data Server 5.2 or later.
- You must install SPD Server by following the installation instructions in *SAS Scalable Performance Data Server: Administrator's Guide*.
- To use SPD Server 5.3 with a Hadoop cluster, SPD Server must be on a Linux x64 or a Microsoft Windows x64 operating system. (Note that to use SPD Server 5.2 with a Hadoop cluster, SPD Server must be on a Linux x64 operating system.)
- SPD Server 5.3 supports the following Hadoop distributions, with or without Kerberos:
  - Cloudera CDH 5.7
  - Hortonworks HDP 2.4
  - MapR 5.1
- To access a Hadoop cluster, the SPD Server administrator must specify Hadoop parameter file options in the libnames.parm parameter file and the spdsserv.parm parameter file and update the rc.spds script. The Hadoop parameter file options specify that the domain can access data in HDFS and make the Hadoop cluster configuration files and Hadoop distribution JAR files available to SPD Server. For more information, see Chapter 2, "Enabling SPD Server with Hadoop," on page 3.

Chapter 2

Enabling SPD Server with Hadoop

Overview: Enabling SPD Server with Hadoop

This chapter provides the information for the SPD Server administrator to enable SPD Server with Hadoop. To enable SPD Server with Hadoop requires only changes by the administrator. The client programs do not need to change.

To enable SPD Server with Hadoop:

1. Verify your Hadoop environment. For more information, see “Checklist to Verify the Hadoop Environment” on page 4.
2. Obtain the Hadoop distribution JAR and cluster configuration files from the Hadoop cluster. For more information, see “Obtaining Hadoop Distribution JAR and Configuration Files from Hadoop Cluster” on page 5.
3. Specify the Hadoop parameter file options in the libnames.parm file and the spdsserv.parm file to make the Hadoop cluster configuration files and the Hadoop distribution JAR files available to SPD Server. For more information, see “Specifying the Hadoop Parameter File Options” on page 9.
4. For a MapR Hadoop distribution, you must add information to the spdsserv.parm parameter file. On a Microsoft Windows operating system, you must add properties to configuration files. For more information, see “Additional Configuration for a MapR Hadoop Distribution” on page 7.
5. On Microsoft Windows, to perform WHERE processing optimization, you must add a property to the mapred-site.xml configuration file. For more information, see “Additional Configuration for Optimized WHERE Processing on Microsoft Windows” on page 8.
6. Update the rc.spds script. For more information, see “Updating the rc.spds Script” on page 11.
7. Run basic tests to confirm that your Hadoop connections are working. For more information, see “Validating the SPD Server to Hadoop Connection” on page 11.

Checklist to Verify the Hadoop Environment

A good understanding of your Hadoop environment is critical to a successful connection between SPD Server and Hadoop. It is recommended that you verify your Hadoop environment by becoming familiar with the following items:

- Gain working knowledge of the Hadoop distribution that you are using (for example, Cloudera). You will also need working knowledge of HDFS and services for MapReduce 1, MapReduce 2, and YARN. For more information, see the Apache website or the vendor’s website.
- Ensure that the HDFS, MapReduce, and YARN services are running on the Hadoop cluster.
- Know the location of the MapReduce home.
- Know the host name of the NameNode.
- Determine where the HDFS cluster is running.
- Understand and verify your Hadoop user authentication.
- Understand and verify your security setup. It is recommended that you enable Kerberos for data security.
- Verify that you can connect to the Hadoop cluster from your client machine outside of SPD Server with your defined security protocol.

Obtaining Hadoop Distribution JAR and Configuration Files from Hadoop Cluster

Overview

To use SPD Server to access files on a Hadoop server, a set of Hadoop JAR and configuration files must be available to the SPD Server machine. To make the required JAR and configuration files available, you must obtain these files from the Hadoop cluster, copy them to the SPD Server machine, and specify the Hadoop parameter file options.

There are two methods to obtain the JAR and configuration files:

- If you license SAS/ACCESS Interface to Hadoop, use the SAS Deployment Manager.
- Use the Hadoop tracer script hadooptracer.py in Python that is provided by SAS.

Note: Gathering the JAR and configuration files is a one-time process (unless you are updating your cluster or changing Hadoop vendors). If you have already gathered the Hadoop JAR and configuration files for another SAS component using the SAS Deployment Manager or the Hadoop tracer script, you do not need to do it again.

Using SAS Deployment Manager to Obtain the Hadoop JAR and Configuration Files

If you license SAS/ACCESS Interface to Hadoop, you should use SAS Deployment Manager to obtain and make required Hadoop JAR and configuration files available to the SPD Server machine. For more information about using SAS Deployment Manager for SAS/ACCESS Interface to Hadoop, see *SAS Hadoop Configuration Guide for Base SAS and SAS/ACCESS*.

If you do not license SAS/ACCESS Interface to Hadoop, follow the instructions in "Using the Hadoop Tracer Script to Obtain the Hadoop JAR and Configuration Files" on page 5.

Using the Hadoop Tracer Script to Obtain the Hadoop JAR and Configuration Files

Prerequisites for Using the Hadoop Tracer Script

These are prerequisites to ensure that the Hadoop tracer script runs successfully:

- Ensure that Python and strace are installed. Contact your system administrator if these packages are not installed on the system.
- Ensure that the user running the script has authorization to issue HDFS and Hive commands.
- If Hadoop is secured with Kerberos, obtain a Kerberos ticket for the user before running the script.

Obtaining and Running the Hadoop Tracer Script

To obtain and run the Hadoop tracer script, follow these steps:

1. On the Hadoop server, create a temporary directory to hold a ZIP file that you will download. An example is **/opt/sas/hadoopfiles/temp**.
2. Download the hadooptracer.zip file from the following FTP site to the directory that you created in Step 1: ftp://ftp.sas.com/techsup/download/blind/access/hadooptracer.zip.
3. Using a method of your choice (for example, PSFTP, SFTP, SCP, or FTP), transfer the ZIP file to the Hive node on your Hadoop cluster.
4. Unzip the file.

   The hadooptracer.py file is included in this ZIP file.
5. Change permissions on the file to have EXECUTE permission.

   ```
   chmod 755 hadooptracer.py
   ```
6. Run the tracer script.

   ```
   python ./hadooptracer.py --filterby latest
   ```

   Note: The **filterby latest** option ensures that if duplicate JAR or configuration files exist, the latest version is selected.

   This script performs the following tasks:

   - Pulls the necessary Hadoop JAR and configuration files and places them in the **/tmp/jars** directory and the **/tmp/sitexmls** directory, respectively.
   - Creates the hadooptrace.json file in the **/tmp** directory. If you need a custom filename or path for the JSON output file, use this command instead:

     ```
     python ./hadooptracer.py -f /your-path/your-filename.json
     ```
   - Creates a log in the **/tmp/hadooptracer.log** directory.

     If a problem occurs and you need to contact SAS Technical Support, please include this log file with your tech support track.

     Note: Some error messages in the console output for hadooptracer.py are normal and do not necessarily indicate a problem with the JAR file and configuration file collection process. However, if the files are not collected as expected or if you experience problems connecting to Hadoop with the collected files, contact SAS Technical Support for assistance and include the /tmp/hadooptracer.log file.
7. On the SPD Server machine, create two directories to hold the JAR and configuration files. An example would be **/opt/sas/hadoopfiles/jars** and **/opt/sas/hadoopfiles/configs**.
8. Using a method of your choice (for example, PSFTP, SFTP, SCP, or FTP), copy the files in the **/tmp/jars** and the **/tmp/jars** and the **/tmp/sitexmls** directories on the Hadoop server to directories on the SPD Server machine that you created in Step 7, such as **/opt/sas/hadoopfiles/jars** and **/opt/sas/hadoopfiles/configs**, respectively.
9. If you are using Hortonworks or MapR, additional configuration is needed. For more information, see the following topics:

   - “Additional Configuration for Hortonworks” on page 7
   - “Additional Configuration for a MapR Hadoop Distribution” on page 7

Additional Configuration for Hortonworks

If you run the Hadoop tracer script on Hortonworks, there are two additional configuration items.

- You must manually revise all occurrences of **${hdp.version}** in the mapred-site.xml property file on the SPD Server machine. Otherwise, an error occurs when you submit a program to Hadoop.

  Use the **hadoop version** command to determine the exact version number of your distribution to use in place of **${hdp.version}**. This example assumes that the current Hortonworks version is 2.2.0.0-2041 and replaces **${hdp.version}** in the mapreduce.application.framework.path property.

  Change the following code:

  ```
  <property>
     <name>mapreduce.application.framework.path</name>
     <value>/hdp/apps/${hdp.version}/mapreduce/mapreduce.tar.gz#mr-framework</value>
  </property>
  ```

  The code above should be replaced with this code:

  ```
  <property>
     <name>mapreduce.application.framework.path </name>
     <value>/hdp/apps/2.2.0.0-2041/mapreduce/mapreduce.tar.gz#mr-framework</value>
  </property>
  ```

- If you are running on a Windows client, you must manually add the following property to the mapred-site.xml file on the SAS client side. Otherwise, an error occurs when you submit a program to Hadoop.

  ```
  <property>
     <name>mapreduce.app-submission.cross-platform</name>
     <value>true</value>
  </property>
  ```

Additional Configuration for a MapR Hadoop Distribution

For a MapR Hadoop distribution, you must do the following:

- If you run the Hadoop tracer script on MapR and are running on a Windows client, you must manually add the following property to the mapred-site.xml file on the SPD Server machine. Otherwise, an error occurs when you submit a program to Hadoop.

  ```
  <property>
     <name>mapreduce.app-submission.cross-platform</name>
     <value>true</value>
  </property>
  ```

- In the spdsserv.parm parameter file, modify the **java.security.auth.login** property in JREOPTIONS to point to the **mapr.login.conf** file. Normally, the **mapr.login.conf** file is installed in the **MAPR_HOME/conf** directory.

  Here is an example:

  ```
  -jreoptions (-Djava.security.auth.login.config=/opt/mapr/conf/mapr.login.conf)
  ```

- On the Microsoft Windows operating system, you must add the group identifier and the user identifier properties to the mapred-site.xml and yarn-site.xml Hadoop cluster configuration files. You do this to describe the account that is responsible for SPD Server. Here is an example. Replace the property attributes in italics with the correct information at your site.

  ```
  <property>
      <name>hadoop.spoofed.user.uid</name>
      <value>4658</value>
    </property>
    <property>
      <name>hadoop.spoofed.user.gid</name>
      <value>100</value>
    </property>
    <property>
      <name>hadoop.spoofed.user.username</name>
      <value>spdsmgr</value>
    </property>
  ```

- On the Linux operating system if the Hadoop cluster supports Kerberos, you must add the following commands to the rc.spds script, which is located in the **InstallDir/site** directory:

  ```
  MAPR_HOME=/opt/mapr
  export MAPR_HOME
  ```

- If the Hadoop cluster supports Kerberos, the MapR ticket expiration and renewal expiration times must be at least as long as the Kerberos ticket expiration and renewal expiration times. SPD Server uses the Kerberos ticket expiration and renewal expiration times to manage MapR tickets. They are renewed and re-created at the same time as the Kerberos ticket.

Additional Configuration for Optimized WHERE Processing on Microsoft Windows

On Microsoft Windows, to perform WHERE processing in the Hadoop cluster with MapReduce, you must add the cross-platform property to the mapred-site.xml cluster configuration file. The property specifies that the platform neutral CLASSPATH should be used. Here is an example.

```
<property>
   <name>mapreduce.app-submission.cross-platform</name>
   <value>true</value>
</property>
```

Specifying the Hadoop Parameter File Options

Making the Hadoop Cluster Configuration Files and Hadoop Distribution JAR Files Available to SPD Server

To make the Hadoop cluster configuration files and the Hadoop distribution JAR files available to SPD Server, you must specify Hadoop parameter file options in either the libnames.parm file (as part of an individual SPD Server domain definition) or in the spdsserv.parm file.

Settings that you specify in the libnames.parm file take precedence over settings in the spdsserv.parm file. This enables the SPD Server administrator to define common Hadoop settings in the spdsserv.parm file and to use the libnames.parm file to override settings when needed. For more information about the libnames.parm file and the spdsserv.parm file, see *SAS Scalable Performance Data Server: Administrator's Guide*.

The following parameter file options are needed to access Hadoop:

- HADOOP=YES on page 33 in the libnames.parm file to specify that an SPD Server domain can access data in HDFS.
- HADOOPCFG= on page 36 to specify the location of the Hadoop cluster configuration files in either the libnames.parm file or the spdsserv.parm file. For example, if the cluster configuration files are copied to the location **/u/hadoop/hdist/cdh/confdir**, then the following syntax sets the location appropriately:

  ```
  HADOOPCFG=/u/hadoop/hdist/cdh/confdir;
  ```

- HADOOPJAR= on page 36 to specify the location of the Hadoop distribution JAR files in either the libnames.parm file or the spdsserv.parm file. For example, if the JAR files are copied to the location **/u/hadoop/hdist/cdh/cdh54**, then the following syntax sets the location appropriately:

  ```
  HADOOPJAR=/u/hadoop/hdist/cdh/cdh54;
  ```

If you specify the HADOOPCFG= and HADOOPJAR= parameter file options in the libnames.parm file, SPD Server automatically assumes that the domain is a Hadoop domain. You do not have to specify HADOOP=YES.

If you specify the HADOOPCFG= and HADOOPJAR= parameter file options in the spdsserv.parm file, the settings become the default settings for the domain. YIn the libnames.parm file, you must specify HADOOP=YES for each domain definition that you want to be a Hadoop domain.

Determining the JRE Version

The Java Runtime Environment (JRE) version for the Hadoop cluster must be a major Java version. The default is 1.7. If the JRE version for the Hadoop cluster is not 1.7, include the HADOOPACCELJVER= parameter file option on page 34 in the spdsserv.parm file to specify the version.

The JRE version of the JAR files in the JAR file directory must match the JRE version of the Hadoop environment. If you have multiple Hadoop servers that are running different JRE versions, then you must create a different JAR file directory for each JRE version.

Kerberos Security

To access a Hadoop cluster that is secured with Kerberos, use the following information:

- If you are running on Linux, a valid Kerberos keytab file is required for the HDFS user ID for your Hadoop cluster. In either the libnames.parm file or the spdsserv.parm file, you must include these parameter file options:
  - HADOOPKEYTAB= on page 37 to specify the path to the Kerberos keytab file
  - HADOOPREALM= on page 38 to specify the Kerberos realm to use
  - HADOOPUSER= on page 38 to specify an authorized Kerberos user ID

  Including the parameter file options in the libnames.parm file specifies that you want the domain to be treated as a domain that is secured with Kerberos. Including the parameter file options in the spdserv.parm file specifies that any domain that is defined in the libnames.parm file is treated as a domain that is secured with Kerberos.

  TIP If you have a domain that you do not want to be treated as a domain that is secured with Kerberos, you can specify HADOOPKERBEROS=NO in the libnames.parm file. For more information, see "HADOOPKERBEROS= Parameter File Option" on page 37.

- If you are running on Microsoft Windows, a keytab file is not required for Kerberos security if your Windows realm trusts your Hadoop realm. The HADOOPKEYTAB=, HADOOPREALM=, and HADOOPUSER= parameter file options should not be used in this case. The single sign-on to the Windows desktop provides the Kerberos TGT to authenticate to Hadoop. However, the Microsoft Active Directory stores the credentials cache in memory. To allow access to the credentials cache, you must add the registry key **AllowTGTSessionKey** and set the value to 1. For more information, see Registry Key to Allow Session Keys to Be Sent in Kerberos Ticket-Granting-Ticket.

- In the spdsserv.parm parameter file, modify the **`java.security.krb5.conf`** property in JREOPTIONS to specify your Kerberos configuration file. Here is an example:

  ```
  -jreoptions
     (-Djava.security.krb4.config=/u/fedadmin/krb5/krb5_PDE.KRB.SAS.COM.ini)
  ```

- If you are using Advanced Encryption Standard (AES) encryption with Kerberos, you must manually add the Java Cryptography Extension local_policy.jar file in every place that JAVA Home resides on the cluster. If you are outside the United States, you must also manually add the US_export_policy.jar file. The addition of these files is governed by the United States import control restrictions.

  These two JAR files also need to replace the existing local_policy.jar and US_export_policy.jar files in the SAS JRE location that is the **`<install-dir>/SASPrivateJavaRuntimeEnvironment/9.4/jre/lib/security/`** directory. It is recommended to back up the existing local_policy.jar and US_export_policy.jar files first in case they need to be restored.

  These files can be obtained from the IBM or Oracle website.

- When SPD Server is secured with Kerberos and is running as a Microsoft Windows service, you must change the Log On account for the three SPD Server services, which include **SPD 5.3 Data Server**, **SPD 5.3 Name Server**, and **SPD 5.3 Snet Server**. The account should match the authorized Hadoop account for your Hadoop cluster. Follow these steps:

1. Access the Services (Local) window.
2. Right-click the service and select **Properties**.
3. Select the **Log On** tab.
4. Select **This Account**, specify the authorized and password, and click **OK**.

SPD Server ACLs with Hadoop Domains

Specify the default location for SPD Server Hadoop ACLs with the ACLDIR= start-up option in the rc.spds script. If you define a domain that contains the Hadoop parameter file option HADOOP=YES, SPD Server creates a directory using the following schema: ***hadoop-acl-path*/HADOOPACLS/*domain_name***.

ACLs for SPD Server resources are typically created in the root pathname of each domain. Storing ACLs in this location does not work with Hadoop domains for the following reasons:

- ACLs are small and are updated frequently. Updating data in HDFS is very slow and can, therefore, significantly degrade performance.
- HDFS does not support the type of locking required for ACL processing.

Because of the restrictions, ACLs with Hadoop domains must be on a local file. The SPD Server parameter file option HADOOPACLPATH= on page 35 specifies a local file system directory where ACLs for the Hadoop environment will be created and stored.

Updating the rc.spds Script

This line must be specified in the rc.spds script that starts an SPD Server:

```
export ELSOPT="-delay proxy"
```

Validating the SPD Server to Hadoop Connection

To validate that SPD Server is correctly configured to access the Hadoop cluster:

- Ensure that you specified values for the HADOOPCFG= and HADOOPJAR= parameter options in either the spdsserv.parm file or in the libnames.parm file.
- If you want one or more domains to access Hadoop data that is secured with Kerberos, make sure that you specified values for the HADOOPKEYTAB=, HADOOPREALM, and HADOOPUSER= parameter file options for the domains in either the spdsserv.parm file or the libnames.parm file.
- After you start SPD Server, check the SPD Server log file to verify that the domains were added to the Name Server. Hadoop option information is logged when each domain is added to the Name Server. The log entry should resemble the following:

```
libname KERB added to Name Server (HADOOP=yes, KERBEROS)
   HADOOPCFG=/u/fedadmin/sdm/hadoopcfg/hdp23k1/prod
   HADOOPJAR=/bigdisk/lax/xxxxxx/spds53/hw23_hadoopjar
```

```
HADOOPUSER=spdsmgr
HADOOPREALM=NA.SAS.COM
HADOOPKEYTAB=/u/fedadmin/keytab/spdsmgr.keytab
HADOOPACLPATH=/bigdisk/lax/xxxxxx/spds53/acls.lax/HADOOPACLS/KERB
```

- Submit a SASSPDS engine LIBNAME statement to a domain, such as the following example:

  ```
  libname foo sasspds 'public' server=myhost.5400 user="anonymous";
  ```

  If the LIBNAME statement is successful, then your Hadoop connection has been made.

- Add MYDOMAIN to your libnames.parm file as a domain libref with a path located in your Hadoop file system.

The following SPD Server log shows the output for a LIBNAME statement and two PROC steps. The log results verify that the Hadoop environment is set up correctly.

Output 2.1 *SPD Server Log*

```
1       libname foo sasspds "public" host="lax94t01.unx.sas.com" service="14503" user="anonymous";
NOTE: This is a SPD 5.3  Engine
      executing SAS (r) 9.4 (TS1M3) on the Linux platform.
NOTE: User anonymous(ACL Group ) connected to SPD(LAX) 5.3 server at 10.24.8.107.
NOTE: Libref FOO was successfully assigned as follows:
      Engine:        SASSPDS
      Physical Name: :5763/user/spdsmgr/sasxxx/test_domains/lax94t01/public/
2          data foo.atable;
3            x=1;
4          run;

NOTE: The data set FOO.ATABLE has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           1.42 seconds
      cpu time            0.04 seconds

2                                                The SAS System

5          proc datasets lib=foo;
                                                                Directory

                  Libref               FOO
                  Engine               SASSPDS
                  Physical Name        :5763/user/spdsmgr/sasxxx/test_domains/lax94t01/public/
                  Local Host Name      lax94t01
                  Local Host IP addr   10.24.8.107
                  Server Hostname      lax94t01.unx.sas.com
                  Server IP addr       .
                  Server Portno        37615
                  Free Space (Kbytes)  9.0071993E15
                  Metapath             '/user/spdsmgr/sasxxx/test_domains/lax94t01/public/'
                  Indexpath            '/user/spdsmgr/sasxxx/test_domains/lax94t01/public/'
                  Datapath             '/user/spdsmgr/sasxxx/test_domains/lax94t01/public/'
                  Hadoop               YES


                                                                Member
                                                          #  Name    Type

                                                          1  ATABLE  DATA
6          run;


NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.62 seconds
      cpu time            0.09 seconds
```

```
7          proc contents data=foo.atable;
8          run;

                                          The SAS System                              1

                                               The CONTENTS Procedure

                              Data Set Name        FOO.ATABLE              Observations          1
                              Member Type          DATA                    Variables             1
                              Engine               SASSPDS                 Indexes               0
                              Created              05/03/2016 12:06:33     Observation Length    8
                              Last Modified        05/03/2016 12:06:33     Deleted Observations  0
                              Protection                                   Compressed            NO
                              Data Set Type                                Sorted                NO
                              Label
                              Data Representation  Default
                              Encoding             latin1  Western (ISO)


                                            Engine/Host Dependent Information

                                          Blocking Factor (obs/block)  131072
                                          ACL Entry                    NO
                                          ACL User Access(R,W,A,C)     (Y,Y,Y,Y)
                                          ACL UserName                 ANONYMOU
                                          ACL OwnerName                ANONYMOU
                                          Data set is Ranged           NO
                                          Data set is a Cluster        NO


                                        Alphabetic List of Variables and Attributes

                                               #    Variable    Type    Len

                                               1    x           Num       8
NOTE: PROCEDURE CONTENTS used (Total process time):
      real time           0.89 seconds
      cpu time            0.03 seconds

NOTE: The PROCEDURE CONTENTS printed page 1.

9          proc print data=foo.atable;
10         run;

                                          The SAS System

                                                      Obs    x

                                                       1     1
NOTE: There were 1 observations read from the data set FOO.ATABLE.
NOTE: The PROCEDURE PRINT printed page 2.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.16 seconds
      cpu time            0.00 seconds
```

SPD Server with Hadoop Configuration Examples

Using the libnames.parm File

One way to access a Hadoop cluster (without Kerberos) is to specify a domain in the libnames.parm file, and then add the HADOOPCFG= and HADOOPJAR= parameter file options.

Here is an example of the libnames.parm file that specifies the HADOOPCFG= and HADOOPJAR= parameter options:

```
libname=foo
   pathname=/user/userlname/mydomain
   hadoopcfg=/u/hadoop/hdist/cdh/confdir/hdp20p1
   hadoopjar=/u/hadoop/hdist/cdh/sas_cdh20u;
```

Using the spdsserv.parm File

Another way to access a Hadoop cluster (without Kerberos) is to specify the HADOOPCFG= and HADOOPJAR= options in the spdsserv.parm file. Then, you can specify multiple domains in your libnames.parm file. Each of those domains gets its Hadoop cluster configuration files and Hadoop distribution JAR files from the spdsserv.parm file.

Here is an example of the HADOOPCFG= and HADOOPJAR= parameter options specified in the spdsserv.parm file:

```
hadoopcfg=/u/hadoop/hdist/cdh/confdir/cdh54d1;
hadoopjar=/u/hadoop/hdist/cdh/cdh54;
```

Then, in the libnames.parm file, you only need to specify a libref, a Hadoop directory, and the parameter option HADOOP=YES.

```
libname=Stuff1 pathname=/user/userlname/mydomain1 hadoop=yes;
libname=Stuff2 pathname=/user/userlname/mydomain2 hadoop=yes;
```

To use the domains that are defined in the libnames.parm file, specify them in the same way as any other SPD Server domain in client code:

```
libname Stuff1 sasspds 'Stuff1' server=lax94d01.14526 user="anonymous";
```

Chapter 3

Functionality for SPD Server with Hadoop

Overview: Functionality for SPD Server with Hadoop

SPD Server reads, writes, and updates data in HDFS. Specific SPD Server functionality is supported for Hadoop storage and is explained in this document.

For more information about SPD Server and its functionality that is not specific to Hadoop storage, see *SAS Scalable Performance Data Server: User's Guide* and *SAS Scalable Performance Data Server: Administrator's Guide*.

Understanding the SPD Server Table File Format

SPD Server organizes data into a streamlined file format that has advantages for a distributed file system like HDFS. The advantages of the SPD Server file format include the following:

- Data is separate from the metadata. The file format consists of separate files: one for data, one for metadata, and two for indexes. Each type of file has an identifying file extension. The extensions are .dpf for data, .mdf for metadata, and .hbx and .idx for indexes.
- The SPD Server file format partitions the data by spreading it across multiple files based on a partition size. Each partition is stored as a separate physical file with the extension .dpf. Depending on the amount of data and the partition size, the data can consist of one or more physical files, but is referenced as one logical file.

  The default partition size is 128 megabytes. You can specify a different partition size with the SPDSSIZE= macro variable or the PARTSIZE= table option. For more information, see the "SPDSSIZE= Macro Variable" on page 32 and the "PARTSIZE= Table Option" on page 48.

SPD Server Restrictions on Hadoop

COPY and LOAD Statements in SQL Explicit Pass-Through

The SPD Server COPY and LOAD statements in SQL explicit pass-through are not supported in a Hadoop domain. The commands require local direct access to the source and destination tables from the machine on which the server is running.

Implicit Pass-Through

An implicit pass-through using the CREATE TABLE AS SELECT statement to SPD Server does not support creating a Hadoop table that already exists. That is, a table on a Hadoop cluster cannot be replaced using implicit pass-through. If a table exists, SPD Server fails a query with a Member Lock error. This results in PROC SQL planning and executing the query, which causes the selected rows to be read from SPD Server to SAS. The selected rows are returned to SPD Server to create the table.

A best practice is to drop the table before creating it.

User-Defined Formats

SPD Server does not support user-defined formats in a Hadoop domain. User-defined formats require a record-level locking proxy, but the Hadoop file system locking mechanisms do not support this operation.

Unsupported SPD Server Options

The following SPD Server options are not supported for a Hadoop domain:

- LIBNAME statement option LOCKING=YES, which enables record-level locking
- libnames.parm parameter file option BACKUP=YES, which specifies to back up or restore objects in the domain using the SPD Server Backup and Restore utilities
- libnames.parm parameter file option DYNLOCK=YES, which enables dynamic locking

SPD Server User Password File

By default, the SPD Server user password file is located in the **`InstallDir/site`** directory. If you do not keep the file in this location, you must specify the location using the ACLDIR start-up option, which is explained in the *SAS Scalable Performance Data Server: Administrator's Guide*. The location for the user password file must be on the local file system, not HDFS. This restriction exists for the following reasons:

- SPD Server password files are small, and are updated frequently. Updating data in HDFS is very slow and can, therefore, significantly degrade performance.
- HDFS does not support the type of locking required for SPD Server password file processing.

SPD Server Distributed Locking

Overview: SPD Server Distributed Locking

SPD Server 5.3 supports distributed locking for data stored in HDFS. Distributed locking provides synchronization and group coordination services to clients over a network connection. For the service provider, SPD Server uses the Apache ZooKeeper coordination service—specifically the implementation of the recipe for Shared Lock that is provided by Apache Curator.

Distributed locking provides the following benefits:

- The lock server maintains the lock state information in memory and does not require Write permission to any client or data library disk storage locations.
- A process requesting a lock on a table that is not available (because the table is already locked) can choose to wait for the table to become available, rather than have the lock request fail immediately.

- If a process abnormally terminates while holding locks on tables, the lock server automatically drops all locks that the client was holding, which eliminates the possibility of leftover lock files.

Understanding the Service Provider

Apache ZooKeeper is an open-source distributed server that enables reliable distributed coordination to distributed client applications over a network. ZooKeeper safely coordinates access to shared resources with other applications or processes. At its core, ZooKeeper is a fault-tolerant, multi-machine server that maintains a virtual hierarchy of data nodes that store coordination data. For more information about ZooKeeper and the ZooKeeper data nodes, see Apache ZooKeeper.

Apache Curator is a high-level API that simplifies using ZooKeeper. Curator adds many features that build on ZooKeeper and handle the complexity of managing connections to the ZooKeeper cluster. For more information about Curator, see Curator.

SPD Server accesses the Curator API to provide the locking services.

Requirements for SPD Server Distributed Locking

SPD Server distributed locking has the following requirements:

- ZooKeeper 3.4.0 or later must be downloaded, installed, and running on the Hadoop cluster. The zookeeper JAR file is required.
- Curator 2.7.0 or later must be downloaded on the Hadoop cluster. The following Curator JAR files are required:
  - curator-client
  - curator-framework
  - curator-recipes

Requesting Distributed Locking

By default, SPD Server uses the standard SPD Server member-level locking. To request distributed locking, you must include parameter file options in the spdsserv.parm parameter file. You must specify the following parameter file options to provide the information so that SPD Server can communicate with ZooKeeper:

- ZKPR_QUORUM= on page 41 to specify the list of quorum machines.
- ZKPR_PORT= on page 41 to specify the I/O port to service requests.

In addition, these parameter file options can be included in the spdsserv.parm parameter file to change default values:

- ZKPR_CTIMEOUT= on page 39 to specify the connection time out.
- ZKPR_LTIMEOUT= on page 40 to specify the lock wait time out.
- ZKPR_MAXRETRY= on page 40 to specify the number of times that Curator attempts to connect to ZooKeeper before failing.
- ZKPR_RETRYSLEEP= on page 42 to specify the amount of time that Curator sleeps between attempts to connect to ZooKeeper.
- ZKPR_RPRTHRESH= on page 42 to specify the wait time before deleting an empty ZooKeeper server node.

- ZKPR_STIMEOUT= on page 43 to specify the wait time before the session is considered expired.

Parallel Processing for Data in HDFS

Overview: Parallel Processing for Data in HDFS

Parallel processing uses multiple threads that run in parallel so that a large operation is divided into multiple smaller ones that are executed simultaneously. SPD Server supports parallel processing to improve the performance of reading and writing data that is stored in HDFS.

By default, SPD Server performs parallel processing only if a Read operation includes WHERE processing. If the Read operation does not include WHERE processing, the Read operation is performed by a single thread. To request parallel processing for all Read and Write operations, use these options:

- SPDSHPRD= macro variable on page 30 to request parallel processing for all Read operations.
- PARALLELREAD= table option on page 46 to request parallel Read processing for the specific table.
- SPDSHPWR= macro variable on page 31 to request parallel Write processing for all Write operations.
- PARALLELWRITE= table option on page 47 to request parallel Write processing for the specific table.

Here is an example of the SPDSHPRD= macro variable to request parallel processing for all Read operations:

```
%let spdshprd=yes;
```

In this example, the PARALLELREAD= table option requests parallel processing for all Read operations for the Class.StudentID table:

```
libname class sasspds 'mydomain' server=myhost.5400 user="anonymous";

proc freq data=class.StudentID (parallelread=yes);
   tables age;
run;
```

Here is an example of the SPDSHPWR= macro variable to request parallel processing for all Write operations and to use eight threads. By specifying the SPDSHPWR= macro variable, parallel processing is performed for all Write operations:

```
%let spdshpwr=8;
```

In this example, the PARALLELWRITE= table option requests parallel processing for all Write operations and to use eight threads:

```
libname class sasspds 'mydomain' server=myhost.5400 user="anonymous";

proc append base=class.StudentID data=class.Ages (parallelwrite=8);
run;
```

Note: For parallel processing for Read operations, SPD Server determines the number of threads to use based on the MAXWHTHREADS= parameter file option value, which specifies the maximum number of threads that SPD Server uses for processing. For parallel processing for Write operations, you must specify the number of threads to use. The maximum number of threads is determined by the MAXWHTHREADS= parameter file option value.

Parallel Processing Considerations

The following are considerations for requesting parallel processing:

- For some environments, parallel processing might not improve the performance. The availability of network bandwidth and the number of CPUs on the SAS client machine determine the performance improvement. It is recommended that you set up a test in your environment to measure performance with and without parallel processing.
- When parallel Read processing occurs, the order in which the rows are returned might not be in the physical order of the rows in the table. Some applications require that rows be returned in the physical order. For example, the COMPARE procedure expects that rows are read from the table in the same order that they were written. Also, legacy code that uses the DATA step or the OBS= table option might rely on physical order to produce the expected results.
- When parallel Write processing occurs, each parallel write thread works on its own data partition file. This can result in multiple partitions that do not contain the maximum partition size for the data file. The number of parallel write threads should be carefully chosen so as not to result in too many small partitions as this will result in poor performance for subsequent reads and writes.

Tuning Parallel Processing Performance

To tune the performance of parallel processing, consider these SPD Server options:

- The SPD Server MAXWHTHREADS= parameter file option specifies the maximum number of threads that SPD Server uses for processing. For more information, see the MAXWHTHREADS= parameter file option in the *SAS Scalable Performance Data Server: Administrator's Guide*.
- The SPD Server THREADNUM= table option specifies the number of threads to be used for processing. For more information, see the THREADNUM= table option in the *SAS Scalable Performance Data Server: User's Guide*.

Running SPD Server as a Microsoft Windows Service

A Microsoft Windows service is a computer program that operates in the background. Windows services can be configured to start when the operating system is started and to run in the background as long as Windows is running. You can also run services in the security context of a specific user account that is different from the logged-on user or the default computer account.

When SPD Server is running as a Microsoft Windows service, you can specify an authorized user ID on the Hadoop cluster. By default, the user ID is SYSTEM.

To specify a user ID, you must change the Log On account for the three SPD Server services, which include **SPD 5.3 Data Server**, **SPD 5.3 Name Server**, and **SPD 5.3 Snet Server**. Follow these steps:

1. Access the Services (Local) window.
2. Right-click the service and select **Properties**.
3. Select the **Log On** tab.
4. Select **This account**, specify the authorized user ID and password, and click **OK**.

SPDSBASE Process

About SPD Server SPDSBASE Process

The SPDSBASE process is responsible for accessing or creating SPD resources for SPD Server users. Several SPDSBASE processes can be active simultaneously in an SPD Server installation, handling work requests for different users or different SAS sessions. The SPDSBASE process can take on the role of either an SPD Server user proxy, an SPD Server SQL proxy, or an SPD Server SQL user proxy.

For information about configuring and managing SPDSBASE processes, see *SAS Scalable Performance Data Server: Administrator's Guide*.

Restrictions for Sharing an SPDSBASE Process in Hadoop Domains

For a Hadoop domain, there are restrictions in order to share SPDSBASE processes.

To access the Hadoop cluster, the SPD Server user proxy forks and executes the SAS Java proxy (JProxy) process. The SAS Java proxy process provides the Java Virtual Machine (JVM) environment that is required to communicate with the Hadoop cluster. A Java proxy is configured to access only one Hadoop cluster.

For Hadoop domains to share the same user proxy, the following parameter file options must have the same values for the domains:

- HADOOPCFG= and HADOOPJAR= parameter file options
- HADOOPUSER=, HADOOPKEYTAB=, and HADOOPREALM= parameter file options, if specified

Using the Directory Cleanup Utility with Hadoop

Overview: Using the Directory Cleanup Utility with Hadoop

The directory cleanup utility performs routine maintenance functions on directories that are used by SPD Server. To execute the utility, you submit the spdsclean command, which supports a simple command-line interface. You control the level of cleanup and the behavior of the utility by specifying command options. For more information about

the directory cleanup utility, including the spdsclean command options, see *SAS Scalable Performance Data Server: Administrator's Guide*.

For a Hadoop domain, you can specify any of the standard spdsclean command options, with the addition of some Hadoop options. The Hadoop options define parameter values for SPD Server environment variables such as PATH, LD_LIBRARY_PATH, TKPATH, and JREOPTIONS.

Providing Hadoop Configuration Files and JAR Files Information

To use the directory cleanup utility with a Hadoop domain, you must provide the Hadoop cluster configuration files path and the Hadoop distribution JAR files path. There are several methods that you can use.

- Use the -hadoopcfg and the -hadoopjar options in the spdsclean command to specify the locations of the configuration files and the JAR files:

```
spdsclean
   -hadoopcfg /u/fedadmin/hadoopcfg/cdh54p1
   -hadoopjar /u/fedadmin/hadoopjars/cdh54
   other-options
```

- Specify the location of the configuration files and the JAR files in either the libnames.parm or spdsserv.parm parameter file, and then reference the parameter file in the spdsclean command with either the -libnamefile or the -parmfile options. Here is an spdsclean command that references the configuration and JAR files that are in the libnames.parm parameter file:

```
spdsclean -libnamefile /opt/spds/site/libnames.parm other-options
```

  Here is the content of the libnames.parm file:

```
libname=Stuff1
   pathname=/user/userlname
   hadoopcfg=/u/fedadmin/hadoopcfg/cdh54p1
   hadoopjar=/u/fedadmin/hadoopjars/cdh54
   hadoop=yes;
```

- You can use SAS environment variables to specify the location of the configuration files and the JAR files. Specify the environment variables using a UNIX command prompt before submitting the spdsclean command.

```
export SAS_HADOOP_CONFIG_PATH=/u/fedadmin/hadoopcfg/cdh54p1
export SAS_HADOOP_JAR_PATH=/u/fedadmin/hadoopjars/cdh54
```

Using the Directory Cleanup Utility on Hadoop ACL Files

To use the directory cleanup utility on Hadoop domain ACL files, you must specify the path to the directory that contains the ACL files. Here are two methods that you can use:

- Use the -hadoopaclpath option in the spdsclean command to specify the path to the ACL files. In addition, you must include the -all or the -acl option. The -hadoopaclpath option overrides any path specified with the HADOOPACLPATH= parameter file option in the spdsserv.parm parameter file. Here is an example of the spdsclean command:

```
spdsclean -libnamefile /opt/spds53/site/libnames.parm
   -hadoopaclpath /opt/spds53/site/acls
   -domains concur
   -acl
   -verbose
```

- Use the HADOOPACLPATH= parameter file option in the spdsserv.parm parameter file to specify the path to the ACL files, and use the -parmfile option in the spdsclean command to specify the location of the spdsserv.parm parameter file. Here is an example of the spdsclean command:

```
spdsclean -libnamefile /opt/spds53/site/libnames.parm
   -parmfile /opt/spds53/site/spdsserv.parm
   -domains concur
   -acl
   -verbose
```

Using the Directory Cleanup Utility on Hadoop Domains Secured with Kerberos

To use the directory cleanup utility on Hadoop domains that are secured with Kerberos, submit the following KINIT command before submitting the spdsclean command:

```
$ kinit -kt full–path–to–keytab–file UserID;
```

Updating Data in HDFS

HDFS does not support updating data. However, because traditional SAS processing involves updating data, SPD Server supports SAS Update operations for data stored in HDFS.

To update data in HDFS, SPD Server uses an approach that replaces the table's data partition file for each row that is updated. When an update is requested, SPD Server re-creates the data partition file in its entirety (including all replications), and then inserts the updated data into the new data partition file. Because the data partition file is replaced for each row that is updated, the greater the number of rows to be updated, the longer the process.

For general-purpose data storage, the ability to perform small, infrequent updates can be beneficial. However, updating data in HDFS is intended for situations when the time it takes to complete the update outweighs the alternatives.

Here are some best practices for Update operations using SPD Server:

- It is recommended that you set up a test in your environment to measure Update operation performance. For example, update a small number of rows to gauge how long updates take in your environment. Then, project the test results to a larger number of rows to determine whether updating is realistic.
- It is recommended that you do not use the SQL procedure to update data in HDFS because of how PROC SQL opens, updates, and closes a file. There are other SAS methods that provide better performance such as the DATA step UPDATE statement and MODIFY statement.

- The performance of appending a table can be slower if the table has a unique index. Test case results show that appending a table to another table without a unique index is significantly faster than appending the same table to another table with a unique index.

WHERE Clause Planner for Hadoop

The SPD Server WHERE clause planner does the tuning work for you by automatically costing the different approaches to index evaluation. The SPD Server WHERE clause planner avoids computation-intensive operations and uses simple computations where possible.

WHERE clause EVAL strategies SPD Server indexing keeps track of the cardinality ratio and distribution of variable values in a table and uses them to calculate the cost of a WHERE clause. The WHERE clause planner uses four evaluation strategies to determine the number of rows that will be required to execute a given query.

EVAL 6 emulates the behavior of EVAL 2. EVAL 6 means that the query is a candidate for Hadoop WHERE processing. If the Hadoop WHERE processing fails, EVAL 6 reverts to EVAL 2 operation. EVAL 6 takes true rows as determined by EVAL 1, EVAL 3, or EVAL 4, and then eliminates any rows shown to be false, leaving a table that contains only true rows. EVAL 2 processes all rows of a table when no index evaluation is possible. For example, no index evaluation is possible when an index is not present or when some predecessor function performs an operation that invalidates the index.

For more information about the WHERE clause planner and WHERE clause EVAL strategies, see *SAS Scalable Performance Data Server: User's Guide*.

WHERE Processing Optimization with MapReduce

Overview: WHERE Processing Optimization with MapReduce

WHERE processing enables you to conditionally select a subset of rows, so that the software processes only the rows that meet specified conditions. To optimize the performance of WHERE processing, you can request that data subsetting be performed in the Hadoop cluster, which can be an SPD Server dynamic cluster table. Then, when you submit SPD Server code that includes a WHERE expression (which defines the condition that selected rows must satisfy), SPD Server instantiates the WHERE expression as a Java class. SPD Server submits the Java class to the Hadoop cluster as a component in a MapReduce program.

By requesting that data subsetting be performed in the Hadoop cluster, performance might be improved by taking advantage of the filtering and ordering capabilities of the MapReduce framework. As a result, only the subset of the data is returned to the SPD Server client. Performance is often improved with large tables when the WHERE expression qualifies only a relatively small subset.

WHERE Processing Optimization Settings

By default, WHERE processing is performed by SPD Server on the SPD Server host. Data subsetting is not performed in the Hadoop cluster. Here are the WHERE processing optimization settings that you can specify:

- In the default state, you can request optimized WHERE processing in specific cases by using the ACCELWHERE= table option on page 45 or the SPDSACWH= macro variable on page 29.
- To request global optimized WHERE processing, you can include the HADOOPACCELWH parameter file option on page 34 in the spdsserv.parm parameter file. All data subsetting is performed in the Hadoop cluster. With HADOOPACCELWH declared, you can use the ACCELWHERE= table option or the SPDSACWH= macro variable to turn off WHERE processing optimization.
- To specify that WHERE processing optimization cannot be requested, you can include the NOHADOOPACCELWH parameter file option on page 34 in the spdsserv.parm parameter file. With NOHADOOPACCELWH declared, you cannot use the ACCELWHERE= table option or the SPDSACWH= macro variable to request WHERE processing optimization.
- On Microsoft Windows, to perform WHERE processing optimization, you must add a property to the mapred-site.xml configuration file. For more information, see "Additional Configuration for Optimized WHERE Processing on Microsoft Windows" on page 8.

Using the HADOOPACCELWH Parameter File Option

To request global WHERE processing optimization so that all data subsetting is performed in the Hadoop cluster:

- Include the HADOOPACCELWH parameter file option in the spdsserv.parm parameter file.
- Use the HADOOPACCELJVER= parameter file option in the spdsserv.parm parameter file to specify the Java Runtime Environment version running on the Hadoop cluster. The default Java version is 1.6.
- Use the HADOOPWORKPATH= parameter file option in the spdsserv.parm parameter file to specify the path to the HDFS directory that stores the temporary results of the MapReduce output. If the parameter is not specified, the files are written to **`/tmp`**. The specified path must exist. If you specify a path that does not exist, the MapReduce job fails.
- Use the SPDSACWH= macro variable or the ACCELWHERE= table option to turn off WHERE processing optimization if needed.

Note: You can include the NOHADOOPACCELWH parameter file option in the spdsserv.parm parameter file to specify that WHERE processing optimization with MapReduce cannot be requested. With the NOHADOOPACCELWH setting, you cannot use the SPDSACWH= macro variable or the ACCELWHERE= table option to request WHERE processing optimization.

WHERE Processing Optimization Requirements

To perform the data subsetting in the Hadoop cluster, the following requirements must be met. If any of these requirements are not met, the subsetting is performed by the SPD Server host, not by a MapReduce program in the Hadoop cluster.

- The table cannot be encrypted.
- The table cannot be compressed.
- The table must be larger than the HDFS block size.
- The submitted SAS code cannot request BY-group processing.
- The submitted SAS code cannot include the STARTOBS= or ENDOBS= options.

The submitted WHERE expression cannot include any of the following syntax:

- a variable as an operand, such as **`where lastname;`**
- variable-to-variable comparison
- SAS functions, such as SUBSTR, TODAY, UPCASE, and PUT
- arithmetic operators *, / , + , – , and **
- IS NULL or IS MISSING and IS NOT NULL or IS NOT MISSING operators
- concatenation operators, such as || or !!
- negative prefix operator, such as **`where z = -(x+y);`**
- pattern matching operators LIKE and CONTAINS
- sounds-like operator SOUNDEX (=*)
- truncated comparison operator using the colon modifier (:), such as **`where lastname=: 'S';`**

Note: For additional details about WHERE processing optimization, include the macro variable SPDSWDEB=YES in your code to determine whether the optimization occurred. For more information, see the SPDSWDEB= macro variable in *SAS Scalable Performance Data Server: User's Guide*.

Chapter 4

Macro Variables Reference

Dictionary

SPDSACWH= Macro Variable

Specifies whether to perform WHERE processing optimization by subsetting the data in the Hadoop cluster with MapReduce.

Valid in: SPD Server

Requirement: To perform data subsetting in the Hadoop cluster, there are table and SAS code requirements. For more information, see "WHERE Processing Optimization with MapReduce" on page 26.

Interaction: The status of the HADOOPACCELWH parameter file option determines the functionality of the SPDSACWH= macro variable. See "HADOOPACCELWH Parameter File Option" on page 34.

Syntax

SPDSACWH=YES | NO

Required Arguments

YES
turns on WHERE processing optimization when the HADOOPACCELWH parameter option is not included (undeclared) in the spdsserv.parm file, which is the default state. If the NOHADOOPACCELWH parameter option is included in the spdsserv.parm file, SPDSACWH=YES is ignored, and WHERE processing optimization is not performed.

NO
turns off WHERE processing optimization when the HADOOPACCELWH parameter option is included (declared) in the spdsserv.parm file.

Details

By default, WHERE processing optimization is not enabled. That is, the spdsserv.parm file does not include the HADOOPACCELWH parameter file option, which requests that data subsetting be performed in the Hadoop cluster with MapReduce. In the default state, you can use the SPDSACWH=YES macro variable to turn on WHERE processing optimization in specific cases.

If global WHERE processing optimization is enabled with the HADOOPACCELWH parameter file option, you can use the SPDSACWH=NO macro variable to turn off WHERE processing optimization in specific cases.

If WHERE processing optimization is disabled with the NOHADOOPACCELWH parameter file option, the SPDSACWH= macro variable setting is ignored.

The ACCELWHERE= table option overrides the SPDSACWH= macro variable. For information about the ACCELWHERE= table option, see "ACCELWHERE= Table Option" on page 45.

Example

```
%let spdsacwh=yes;
```

Note: Because SPD Server is able to operate in select Hadoop environments, you might want to determine whether a libref resides in a Hadoop domain. For example, to determine whether the libref My_Lib is in a Hadoop domain, submit the LIBNAME LIST statement **`libname my_lib list;`**. If the queried libref is in a Hadoop domain, SPD Server returns HADOOP=YES.

SPDSHPRD= Macro Variable

Determines when SPD Server uses parallel processing to read data stored in HDFS.

Valid in: SPD Server

Default: NO

Interaction: The PARALLELREAD= table option setting overrides the SPDSHPRD= macro variable. For more information, see "PARALLELREAD= Table Option" on page 46.

See: "Parallel Processing for Data in HDFS" on page 21

Syntax

SPDSHPRD=NO | YES

Required Arguments

NO
specifies that parallel processing occurs only if a Read operation includes WHERE processing. This is the default behavior.

YES
requests parallel processing for all Read operations.

Example

```
%let spdshprd=yes;
```

Note: Because SPD Server must be enabled to operate in select Hadoop environments, you might want to determine whether a libref is in a Hadoop domain. For example, to determine whether the libref My_Lib is in a Hadoop domain, submit the LIBNAME LIST statement **`libname my_lib list;`**. If the queried libref is in a Hadoop domain, SPD Server returns HADOOP=YES.

SPDSHPWR= Macro Variable

Determines whether SPD Server uses parallel processing to write data in HDFS.

Valid in: SPD Server

Default: 0, which means no parallel processing for Write operations.

Interactions: The PARALLELWRITE= table option setting overrides the SPDSHPWR= macro variable setting. For more information, see "PARALLELWRITE= Table Option" on page 47.

When parallel Write processing occurs, the order in which the rows are written is unpredictable. The order in which the rows are returned cannot be determined unless the application imposes ordering criteria.

See: "Parallel Processing for Data in HDFS" on page 21

Syntax

SPDSHPWR=*threads*

Required Argument

threads
specifies the number of threads to use for Write operations. Specifying 1 thread is the same as the default 0, which means no parallel processing for Write operations.

Interaction The SPD Server MAXWHTHREADS= parameter file option, which is specified in the spdsserv.parm parameter file, determines the maximum number of threads that SPD Server uses for processing. For more information, see the MAXWHTHREADS= parameter file option in the *SAS Scalable Performance Data Server: Administrator's Guide*.

Note Negative numbers are invalid and will result in an error.

Example

```
%let spdshpwr=8;
```

Note: Because SPD Server must be enabled to operate in select Hadoop environments, you might want to determine whether a libref is in a Hadoop domain. For example, to determine whether the libref My_Lib is in a Hadoop domain, submit the

LIBNAME LIST statement **`libname my_lib list;`**. If the queried libref is in a Hadoop domain, SPD Server returns HADOOP=YES.

SPDSSIZE= Macro Variable

Specifies the size of the SPD Server data partition file.

Valid in: SPD Server

Default: 128 megabytes

Range: 16 to 8,796,093,022,207 megabytes

Restriction: Specify a data partition file size only when creating a new table.

Interaction: The PARTSIZE= table option overrides the SPDSSIZE= macro variable. For more information, see "PARTSIZE= Table Option" on page 48.

Tip: To update data, a smaller partition size provides the best performance. For example, when you update a value, SPD Server locates the appropriate partition, modifies the value, and rewrites all replications of the partition. Because each update requires that the partition be rewritten, it is recommended that you perform updates only occasionally or set a small partition size if you are planning to update the data frequently.

Syntax

SPDSSIZE=*n*

Required Argument

n
is the size of the data partition file in megabytes.

Details

Each partition is stored as a separate file with the file extension .dpf. Depending on the amount of data and the partition size, the data can consist of one or more physical files, but is referenced as one logical file.

The SPDSSIZE= specification is limited by the MINPARTSIZE= parameter file option, which specifies the minimum data partition size. MINPARTSIZE= ensures that a SAS user does not create small partitions, thereby generating a large number of physical files. The default for MINPARTSIZE= is 128 megabytes.

For a Hadoop domain, the value of SPDSSIZE= must be larger than the greater of the value declared for MINPARTSIZE= or 128 megabytes, in order to have any effect.

Example

```
%let spdssize=250;
```

Chapter 5

Parameter File Options Reference

Dictionary

HADOOP= Parameter File Option

Specifies whether a domain can access data in HDFS.

Valid in: libnames.parm file

Default: NO

See: “Making the Hadoop Cluster Configuration Files and Hadoop Distribution JAR Files Available to SPD Server” on page 9

Syntax

HADOOP=NO | YES;

Required Arguments

NO
specifies that a domain cannot access data in HDFS. This is the default setting.

YES
specifies that a domain can access data in HDFS.

HADOOPACCELJVER= Parameter File Option

Specifies the JRE version used in the Hadoop environment when performing WHERE processing optimization with MapReduce.

Valid in: spdsserv.parm file

Default: 1.7

Interaction: To request that data subsetting be performed in the Hadoop cluster, use the HADOOPACCELWH parameter file option. By default, data subsetting is performed by SPD Server on the SAS client.

Syntax

HADOOPACCELJVER=*version*;

Required Argument

version
specifies the JRE version. The value must be a major Java version. For the version, specify only the numbers that appear before the hyphen or underscore.

```
hadoopacceljver=1.6;
```

HADOOPACCELWH Parameter File Option

Specifies whether to perform WHERE processing optimization by subsetting the data in the Hadoop cluster with MapReduce.

Valid in: spdsserv.parm file

Default: Data subsetting is performed on the SPD Server host.

Interactions: The HADOOPACCELWH parameter option affects only domains that specify HADOOP=YES or domains that specify any other Hadoop option.

Requesting that data subsetting be performed in the Hadoop cluste might improve performance by taking advantage of the filtering and ordering capabilities of the MapReduce framework. As a result, only the subset of the data is returned to the SPD Server client. Performance is often improved with large tables when the WHERE expression qualifies only a relatively small subset.

Syntax

HADOOPACCELWH | **NOHADOOPACCELWH**;

Required Arguments

HADOOPACCELWH
specifies that data subsetting is performed in the Hadoop cluster with a MapReduce program. Here is an example:

```
hadoopaccelwh;
```

Requirements To perform data subsetting in the Hadoop cluster, there are requirements. See "WHERE Processing Optimization with MapReduce" on page 26.

To submit the MapReduce program to the Hadoop cluster, the Hadoop configuration file must include the properties to run MapReduce (MR1) or MapReduce 2 (MR2) and YARN.

Interaction With HADOOPACCELWH specified in the spdsserv.parm file, you can use the SPDSACWH= macro variable or the ACCELWHERE= table option to turn off WHERE processing optimization.

NOHADOOPACCELWH
specifies that WHERE processing optimization with MapReduce cannot be requested.

Interaction With NOHADOOPACCELWH specified in the spdsserv.parm file, you cannot use the SPDSACWH= macro variable or the ACCELWHERE= table option to request WHERE processing optimization.

HADOOPACLPATH= Parameter File Option

Specifies the location of the Hadoop ACL files.

Valid in: spdsserv.parm file

Default: The ACL files are created in the location that is identified in the ACLDIR= start-up option in the rc.spds script. This is the same location as the psmgr database.

Interaction: If you specify a value for the HADOOPACLPATH= option with the SPDSCLEAN utility, that value overrides any other HADOOPACLPATH= settings that are specified in the spdsserv.parm file.

See: "SPD Server ACLs with Hadoop Domains" on page 11

Syntax

HADOOPACLPATH=*pathname*;

Required Argument

pathname
specifies the directory path for the Hadoop ACL files.

HADOOPCFG= Parameter File Option

Specifies the location of the Hadoop cluster configuration files.

Valid in: libnames.parm file

spdsserv.parm file

Interactions: If you specify this option in the spdsserv.parm file, it affects only domains that specify HADOOP=YES or domains that specify other Hadoop domain options.

If you specify any Hadoop domain option, SPD Server assumes that the HADOOP=YES parameter option is set for that domain.

See: "Making the Hadoop Cluster Configuration Files and Hadoop Distribution JAR Files Available to SPD Server" on page 9

Syntax

HADOOPCFG=*pathname*;

Required Argument

pathname

specifies the directory path for the Hadoop cluster configuration files. Here is an example:

```
hadoopcfg=/u/hadoop/hdist/cdh/confdir;
```

HADOOPJAR= Parameter File Option

Specifies the location of the Hadoop distribution JAR files.

Valid in: libnames.parm file

spdsserv.parm file

Interactions: If you specify this option in the spdsserv.parm file, it affects only domains that specify HADOOP=YES or domains that specify other Hadoop domain options.

If you specify any Hadoop domain option, SPD Server assumes that the HADOOP=YES option is set for that domain.

See: "Making the Hadoop Cluster Configuration Files and Hadoop Distribution JAR Files Available to SPD Server" on page 9

Syntax

HADOOPJAR=*pathname*;

Required Argument

pathname

specifies the directory path for the Hadoop distribution JAR files. Here is an example:

```
hadoopjar=/u/hadoop/hdist/cdh/cdh54;
```

TIP If multiple paths are required, you can use the colon (:) to concatenate pathnames. Here is a UNIX example:

```
HADOOPJAR=/u/hadoop/hdist/cdh/cdh54:/u/myjars/cdh/cdh54;
```

HADOOPKERBEROS= Parameter File Option

Specifies whether a Hadoop cluster requires authentication using Kerberos.

Valid in: libnames.parm file

Default: NO

Requirement: If you are running on Linux, to access a Hadoop cluster that is secured with Kerberos, the HADOOPKEYTAB=, HADOOPREALM=, and HADOOPUSER= parameter file options must be specified in either the libnames.parm file or the spdsserv.parm parameter file.

Interaction: If the HADOOPKEYTAB=, HADOOPREALM=, and HADOOPUSER= parameter file options are specified in the spdsserv.parm file, then a domain that has HADOOP=YES specified is secured by Kerberos unless HADOOPKERBEROS=NO is specified.

See: "Kerberos Security" on page 10

Syntax

HADOOPKERBEROS=NO | YES;

Required Arguments

NO
specifies that a Hadoop cluster does not require authentication using Kerberos.

YES
specifies that a Hadoop cluster requires authentication using Kerberos.

HADOOPKEYTAB= Parameter File Option

When running on Linux, specifies the location of the Kerberos keytab file when accessing a Hadoop cluster that is secured with Kerberos.

Valid in: libnames.parm file

spdsserv.parm file

Interactions: If you specify any Hadoop domain option, SPD Server assumes that HADOOP=YES is set for that domain.

If you specify this option in the spdsserv.parm file, it affects only domains that specify HADOOP=YES or domains that specify other Hadoop options.

See: "Kerberos Security" on page 10

Syntax

HADOOPKEYTAB=*pathname*;

Required Argument

pathname
specifies the directory path for the Kerberos keytab file. A Kerberos keytab file contains pairs of Kerberos principals and an encrypted copy of that principal's key. The keytab file enables you to access distinct Kerberos Services without being prompted for a password.

HADOOPREALM= Parameter File Option

When running on Linux, specifies the Kerberos realm to use in order to access a Hadoop cluster that is secured with Kerberos.

Valid in: libnames.parm file
spdserv.parm file

Interactions: If you specify this option in the spdsserv.parm file, it affects only domains that specify HADOOP=YES or domains that specify other Hadoop options.
If you specify any Hadoop parameter file option, SPD Server assumes that HADOOP=YES is set for that domain.

See: "Kerberos Security" on page 10

Syntax

HADOOPREALM=*Kerberos-realm*;

Required Argument

Kerberos-realm
specifies the Kerberos realm. A Kerberos realm is an authentication administrative domain. A realm establishes the boundaries within which an authentication server has the authority to authenticate a user, host, or service.

HADOOPUSER= Parameter File Option

When running on Linux, specifies an authorized Kerberos user ID for access to the Hadoop cluster.

Valid in: libnames.parm file
spdsserv.parm file

Default: The default is the user ID of the SPD Server process owner.

Interaction: If you specify this option in the spdsserv.parm file, it affects only domains that specify HADOOP=YES or domains that specify other Hadoop options.

See: "Kerberos Security" on page 10

Syntax

HADOOPUSER=*ID*;

Required Argument

ID
is an authorized Kerberos user ID on the Hadoop cluster.

HADOOPWORKPATH= Parameter File Option

Specifies the HDFS directory that stores the temporary results of the MapReduce output.

Valid in: spdsserv.parm file

Default: /tmp

Interaction: This option affects only domains that specify HADOOP=YES or domains that specify any other Hadoop option.

Syntax

HADOOPWORKPATH=*pathname*;

Required Argument

pathname
specifies the directory path to an HDFS directory.

MINPARTSIZE= Parameter File Option

Specifies the minimum data partition size for an SPD Server table.

Valid in: spdsserv.parm file

Default: 128 megabytes

Syntax

MINPARTSIZE=*n*;

Required Argument

n
is the minimum data partition file size in megabytes. Setting a minimum data partition size ensures that large tables cannot be created with an arbitrarily small partition size. Large tables with small partition sizes create an excessive number of physical files, which increases clutter and degrades I/O performance. Here is an example:

```
minpartsize=256;
```

ZKPR_CTIMEOUT= Parameter File Option

For SPD Server distributed locking, specifies the connection time-out.

Valid in: spdsserv.parm file

Default: 30000

See: "SPD Server Distributed Locking" on page 19

Syntax

ZKPR_CTIMEOUT=*n*;

Required Argument

n

is the number of milliseconds that Curator and the ZooKeeper client wait for a communication from the ZooKeeper server before considering the server connection to be expired (for example, **`zkpr_ctimeout=10000;`**). When operating normally, the client establishes a connection to the server and communicates with it over that connection. If the connection is non-responsive for more than the specified value, it is considered expired and is dropped, followed by an attempt to establish a new connection. Values less than or equal to zero are ignored.

ZKPR_LTIMEOUT= Parameter File Option

For SPD Server distributed locking, specifies the lock wait time-out.

Valid in: spdsserv.parm file

Default: 90000

See: "SPD Server Distributed Locking" on page 19

Syntax

ZKPR_CTIMEOUT=*n*;

Required Argument

n

is the number of milliseconds that the ZooKeeper server waits for a lock to become available before declaring that a lock request has failed and returning control to the client (for example, **`zkpr_ltimeout=10000;`**). Values less than zero are ignored. A value of zero is valid.

ZKPR_MAXRETRY= Parameter File Option

For SPD Server distributed locking, specifies the number of times that Curator attempts to connect to ZooKeeper before failing.

Valid in: spdsserv.parm file

Default: 3

See: "SPD Server Distributed Locking" on page 19

Syntax

ZKPR_MAXRETRY=*n*;

Required Argument

n
is the number of times that Curator attempts to connect to ZooKeeper before failing (for example, `zkpr_maxretry=5;`). Values less than or equal to zero are ignored.

ZKPR_PORT= Parameter File Option

For SPD Server distributed locking, specifies the I/O port to service requests.

Valid in: spdsserv.parm file

Requirement: To request distributed locking, you must include ZKPR_PORT= and ZKPR_QUORUM= on page 41 parameter file options in the spdsserv.parm file.

See: “SPD Server Distributed Locking” on page 19

Syntax

ZKPR_PORT='*port*';

Required Argument

port
specifies the I/O port on which the quorum machines that are listed in the ZKPR_QUORUM= parameter file option are configured to service requests (for example, `zkpr_port='2181';`).

ZKPR_QUORUM= Parameter File Option

For SPD Server distributed locking, specifies the list of quorum machines.

Valid in: spdsserv.parm file

Requirements: To request distributed locking, you must include ZKPR_QUORUM= and ZKPR_PORT= on page 41 parameter file options in the spdsserv.parm file.

The listed machines must be running a ZooKeeper server and servicing requests on the port that is specified in the ZKPR_PORT= parameter file option.

Note: As a best practice, the ZooKeeper machines should be on the Hadoop cluster that is being accessed.

See: “SPD Server Distributed Locking” on page 19

Syntax

ZKPR_QUORUM='*quorum*';

Required Argument

quorum
specifies a comma-separated list of quorum machines that are configured to work together as a single server (for example, `zkpr_quorum='abcdef07.unx.sas.com,abcdef08.unx.sas.com,abcdef06.unx.sas.com';`).

ZKPR_RETRYSLEEP= Parameter File Option

For SPD Server distributed locking, specifies the amount of time that Curator sleeps between attempts to connect to ZooKeeper.

Valid in: spdsserv.parm file

Default: 1000

See: "SPD Server Distributed Locking" on page 19

Syntax

ZKPR_RETRYSLEEP=*n*;

Required Argument

n
is the number of milliseconds that Curator sleeps between attempts to connect to ZooKeeper (for example, `zkpr_retrysleep=2000;`). The sleep time starts with this setting, but increases between each attempt. Values less than or equal to zero are ignored.

ZKPR_RPRTHRESH= Parameter File Option

For SPD Server distributed locking, specifies the wait time before deleting an empty ZooKeeper server node.

Valid in: spdsserv.parm file

Default: 3000

See: "SPD Server Distributed Locking" on page 19

Syntax

ZKPR_RPRTHRESH=*n*;

Required Argument

n
is the number of milliseconds that the ZooKeeper server waits before deleting an empty ZooKeeper server node (for example, `zkpr_rprthresh=4000;`).

ZKPR_STIMEOUT= Parameter File Option

For SPD Server distributed locking, specifies the wait time before the session is considered expired.

Valid in: spdsserv.parm file

Default: 180000

See: “SPD Server Distributed Locking” on page 19

Syntax

ZKPR_STIMEOUT=*n*;

Required Argument

n

is the number of milliseconds that Curator and the ZooKeeper client wait for a communication from the ZooKeeper server before considering the client session to be expired (for example, **`zkpr_stimeout=150000;`**). When operating normally, the client establishes a connection to the server and communicates with it over that connection. The connection might be dropped and re-established as the network or server nodes experience faults. But the client session continues to exist for the duration of these interruptions. If an interruption persists for more than the specified value, the client session is considered expired and is terminated. No reconnection is possible after that. Values less than or equal to zero are ignored.

Chapter 6

Table Options Reference

Dictionary

ACCELWHERE= Table Option

Specifies whether to perform WHERE processing optimization by subsetting the data in the Hadoop cluster with MapReduce.

Valid in: SPD Server
DATA step and PROC step

Requirement: To perform data subsetting in the Hadoop cluster, there are table and SAS code requirements. For more information, see "WHERE Processing Optimization with MapReduce" on page 26.

Interaction: The status of the HADOOPACCELWH parameter file option determines the functionality of the ACCELWHERE= table option. See "HADOOPACCELWH Parameter File Option" on page 34 .

Syntax

ACCELWHERE=YES | NO

Required Arguments

YES
turns on WHERE processing optimization when the HADOOPACCELWH parameter option is not included (undeclared) in the spdsserv.parm file, which is the default state. If the NOHADOOPACCELWH parameter option is included in the spdsserv.parm file, ACCELWHERE=YES is ignored and WHERE processing optimization is not performed.

NO
turns off WHERE processing optimization when the HADOOPACCELWH parameter option is included (declared) in the spdsserv.parm file.

Details

By default, WHERE processing optimization is not enabled. That is, the spdsserv.parm file does not include the HADOOPACCELWH parameter file option, which requests that data subsetting be performed in the Hadoop cluster with MapReduce. In the default state, you can use the ACCELWHERE=YES table option to turn on WHERE processing optimization in specific cases.

If global WHERE processing optimization is enabled with the HADOOPACCELWH parameter file option, you can use the ACCELWHERE=NO table option to turn off WHERE processing optimization in specific cases.

If WHERE processing optimization is disabled with the NOHADOOPACCELWH parameter file option, the ACCELWHERE= table option setting is ignored.

The ACCELWHERE= table option overrides the SPDSACWH= macro variable. For information about the SPDSACWH= macro variable, see "SPDSACWH= Macro Variable" on page 29.

Example

```
proc print data=my_lib.my_table (accelwhere=yes)
   where x=1;
run;
```

PARALLELREAD= Table Option

Determines when SPD Server uses parallel processing to read data stored in HDFS.

Valid in: SPD Server
DATA step and PROC step

Default: NO

Syntax

PARALLELREAD=NO | YES

Required Arguments

NO
specifies that parallel processing occurs only if a Read operation includes WHERE processing. This is the default behavior.

YES
requests parallel processing for all Read operations for the specific table.

Details

The PARALLELREAD= table option setting overrides the SPDSHPRD= macro variable. For more information about the SPDSHPRD= macro variable, see "SPDSHPRD= Macro Variable" on page 30.

Example

```
libname class sasspds 'mydomain' server=myhost.5400 user="anonymous";

proc freq data=class.StudentID (parallelread=yes);
   tables age;
run;
```

Note: Because SPD Server must be enabled to operate in select Hadoop environments, you might want to determine whether a libref is in a Hadoop domain. For example, to determine whether the libref Class is in a Hadoop domain, submit the LIBNAME LIST statement **`libname class list;`**. If the queried libref is in a Hadoop domain, SPD Server returns HADOOP=YES.

PARALLELWRITE= Table Option

Determines whether SPD Server uses parallel processing to write data in HDFS.

Valid in: SPD Server
DATA step and PROC step

Default: 0, which means no parallel processing for Write operations.

Interactions: The PARALLELWRITE= table option setting overrides the SPDSHPWR= macro variable setting. For more information, see "SPDSHPWR= Macro Variable" on page 31.

When parallel Write processing occurs, the order in which the rows are written is unpredictable. The order in which the rows are returned cannot be determined unless the application imposes ordering criteria.

See: "Parallel Processing for Data in HDFS" on page 21

Syntax

PARALLELWRITE=*threads*

Required Argument

threads
specifies the number of threads to use for Write operations. Specifying 1 thread is the same as the default 0, which means no parallel processing for Write operations.

Interaction The SPD Server MAXWHTHREADS= parameter file option, which is specified in the spdsserv.parm parameter file, determines the maximum number of threads that SPD Server uses for parallel processing. For more information, see the MAXWHTHREADS= parameter file option in the *SAS Scalable Performance Data Server: Administrator's Guide*.

Note Negative numbers are invalid and will result in an error.

Example

```
libname class sasspds 'mydomain' server=myhost.5400 user="anonymous";

proc append base=class.StudentID data=class.Ages (parallelwrite=8);
run;
```

Note: Because SPD Server must be enabled to operate in select Hadoop environments, you might want to determine whether a libref is in a Hadoop domain. For example, to determine whether the libref Class is in a Hadoop domain, submit the LIBNAME LIST statement **`libname class list;`**. If the queried libref is in a Hadoop domain, SPD Server returns HADOOP=YES.

PARTSIZE= Table Option

Specifies the size of the SPD Server data partition file.

Valid in: SPD Server
DATA step and PROC step

Default: 128 megabytes

Range: 16 to 8,796,093,022,207 megabytes

Restriction: Specify a data partition file size only when creating a new table.

Interaction: The PARTSIZE= table option overrides the SPDSIZE= macro variable. For more information, see “SPDSSIZE= Macro Variable” on page 32.

Tip: When you update data, a smaller partition size provides the best performance. For example, when you update a value, SPD Server locates the appropriate partition, modifies the value, and rewrites all replications of the partition. Because each update requires that the partition be rewritten, it is recommended that you perform updates only occasionally or set a small partition size if you are planning to update the data frequently.

Syntax

PARTSIZE=*n*

Required Argument

n
is the size of the data partition file in megabytes.

Details

Each partition is stored as a separate file with the file extension .dpf. Depending on the amount of data and the partition size, the data can consist of one or more physical files, but is referenced as one logical file.

The PARTSIZE= specification is limited by the MINPARTSIZE= parameter file option. The MINPARTSIZE= parameter file option specifies the minimum data partition size.

MINPARTSIZE= ensures that a SAS user does not create small partitions, thereby generating a large number of physical files. The default for MINPARTSIZE= is 128 megabytes.

For a Hadoop domain, the value of PARTSIZE= must be larger than the greater of the value declared for MINPARTSIZE= or 128 megabytes, in order to have any effect.

Example

```
proc sql;
   create table spdscen.hr80spds (partsize=250)
      .
      .
      .;
quit;
```

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Scalable Performance Data Server: Administrator's Guide*
- *SAS Scalable Performance Data Server: User's Guide*
- *SAS FedSQL Language Reference*
- *SAS and Hadoop Technology: Overview*

For a complete list of SAS publications, go to sas.com/store/books. If you have questions about which titles you need, please contact a SAS Representative:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
Phone: 1-800-727-0025
Fax: 1-919-677-4444
Email: sasbook@sas.com
Web address: sas.com/store/books

Glossary

access control list (ACL)
a list of users and permission types that each user has for a data resource such as a file, directory, or table.

ACL
See access control list.

Apache Hadoop (Hadoop)
an open-source framework that enables the distributed processing of large data sets, across clusters of computers, using a simple programming model.

authentication
See client authentication.

Base SAS
the core product that is part of SAS Foundation and is installed with every deployment of SAS software. Base SAS provides an information delivery system for accessing, managing, analyzing, and presenting data.

big data
information (both structured and unstructured) of a size, complexity, variability, and velocity that challenges or exceeds the capacity of an organization to handle, store, and analyze it.

block
a group of observations in a data set. By using blocks, thread-enabled applications can read, write, and process the observations faster than if they are delivered as individual observations.

Certificate Revocation List (CRL)
a list of revoked digital certificates. CRLs are published by Certification Authorities (CAs), and a CRL contains only the revoked digital certificates that were issued by a specific CA.

client authentication (authentication)
the process of verifying the identity of a person or process for security purposes. Authentication is commonly used in providing access to software, and to data that contains sensitive information.

cluster
See computer cluster.

component file
any of several file types in a logical file structure that is tracked and indexed as a single table. Each SPD Server table includes a metadata file (.mdf), at least one data file (.dpf), and might also include index files (.hbx or .idx).

compound WHERE expression
a WHERE expression that contains more than one operator, as in WHERE X=1 and Y>3. *See also* WHERE expression.

computer cluster (cluster)
a set of connected nodes (computers that are used as servers) in a centralized, cohesive system that shares computing tasks across the system for fast, reliable processing. A computer cluster can be as simple as two machines connected in a network, but more often refers to a large network of computers. A cluster can be established to achieve higher levels of performance and load distribution, or to increase reliability through redundancy.

controller
a computer component that manages the interaction between the computer and a peripheral device such as a disk or a RAID. For example, a controller manages data I/O between a CPU and a disk drive. A computer can contain many controllers. A single CPU can command more than one controller, and a single controller can command multiple disks.

CPU-bound application
an application whose performance is constrained by the speed at which computations can be performed on the data. Multiple CPUs and threading technology can alleviate this problem.

CRL
See Certificate Revocation List.

data partition
a physical file that contains data and which is part of a collection of physical files that comprise the data component of a table. *See also* partition.

data resource
any of a collection of domains, tables, catalogs and other types of data that users (with permissions) can access with SPD Server.

directory cleanup utility (spdsclean)
a component of SPD Server that performs routine maintenance functions on directories.

distinguished name (DN)
a unique identifier of an entry in an LDAP network directory. In effect, a distinguished name is the path to the object in the directory information tree.

distributed data
data that is divided and stored across multiple connected computers.

distributed locking
provides synchronization and group coordination services to clients over a network connection. The service provider is the Apache ZooKeeper coordination service, specifically the implementation of the recipe for Shared Lock that is provided by Apache Curator.

DN
See distinguished name.

domain
for SPD Server, a specific directory of file storage locations. The SPD Server Administrator defines the domain in the libnames.parm parameter file and assigns a name. Users connect to the SPD Server domain by specifying the domain name, for example, in the LIBNAME statement for the SASSPDS engine.

dynamic cluster table
two or more SPD Server tables that are virtually concatenated into a single entity, using metadata that is managed by SPD Server.

dynamic locking
provides multiple users concurrent access to tables. Users can perform read and write functions, and the integrity of the table contents is preserved. Clients that use dynamic locking connect to a separate SPD user proxy process for each connection in the domain.

explicit pass-through
a form of the SQL pass-through facility that passes the user-written, DBMS-specific SQL query code directly to a particular DBMS for processing. *See also* implicit pass-through.

firewall
a set of related programs that protect the resources of a private network from users from other networks. A firewall can also control which outside resources the internal users are able to access.

format
See SAS format.

function
See SAS function.

Hadoop
See Apache Hadoop.

Hadoop configuration file
a file that defines how a system connects to the Hadoop cluster, and provides system information.

Hadoop Distributed File System (HDFS)
a portable, scalable framework, written in Java, for managing large files as blocks of equal size. The files are replicated across multiple host machines in a Hadoop cluster in order to provide fault tolerance.

Hadoop distribution
a collection of Hadoop components such as HDFS, Hive, and MapReduce. A commercial Hadoop distribution is provided by a vendor such as Cloudera and Hortonworks.

HDFS
See Hadoop Distributed File System.

I/O-bound application
an application whose performance is constrained by the speed at which data can be delivered for processing. Multiple CPUs, partitioned I/O, threading technology, RAID (redundant array of independent disks) technology, or a combination of these can alleviate this problem.

implicit pass-through
a form of the SQL pass-through facility that translates SAS SQL query code to the DBMS-specific SQL code, enabling the translated code to be passed to a particular DBMS for processing. *See also* explicit pass-through.

informat
See SAS informat.

JAR (Java Archive)
the name of a package file format that is typically used to aggregate many Java class files and associated metadata and resources (text, images, and so on) into one file to distribute application software or libraries on the Java platform.

Java Archive
See JAR.

Java Database Connectivity (JDBC)
a standard interface for accessing SQL databases. JDBC provides uniform access to a wide range of relational databases. It also provides a common base on which higher-level tools and interfaces can be built.

JDBC
See Java Database Connectivity.

Kerberos authentication
a protocol for network authentication, and mutual authentication, between a client and a server, or between one server and another server. Kerberos uses 'tickets' to allow nodes in a non-secure network to prove their identity to one another in a secure manner, protecting against eavesdropping and network attacks.

LDAP (Lightweight Directory Access Protocol)
a protocol that is used for accessing directories or folders. LDAP is based on the X.500 standard, but it is simpler and, unlike X.500, it supports TCP/IP.

libnames.parm file
an SPD Server parameter file that defines the domains by establishing the names and file storage locations for data resources. The file also serves as a tool for controlling access to the domains and for managing storage of SPD Server files.

library reference
See libref.

libref (library reference)
a SAS name that is associated with the location of a SAS library. For example, in the name MYLIB.MYFILE, MYLIB is the libref, and MYFILE is a file in the SAS library.

light-weight process thread
a single-threaded subprocess that is created and controlled independently, usually with operating system calls. Multiple light-weight process threads can be active at one time on symmetric multiprocessing (SMP) hardware or in thread-enabled operating systems.

Lightweight Directory Access Protocol
See LDAP.

macro variable (symbolic variable)
a variable that is part of the SAS macro programming language. The value of a macro variable is a string that remains constant until you change it.

MapReduce
a component of Apache Hadoop, a parallel programming model for distributed processing of large data sets. The Map phase performs operations such as filtering, transforming and sorting. The Reduce phase aggregates the output.

name server
an SPD Server server process that converts domain names to data storage locations.

national language support (NLS)
the set of features that enable a software product to function properly in every global market for which the product is targeted.

NLS
See national language support.

ODBC
See Open Database Connectivity.

Open Database Connectivity (ODBC)
an interface standard that provides a common application programming interface (API) for accessing data. Many software products that run in the Windows operating environment comply with this standard so that you can access data that was created using other software products.

parallel execution
See parallel processing.

parallel I/O
a method of input and output that takes advantage of multiple CPUs and multiple controllers, with multiple disks per controller to read or write data in independent threads.

parallel processing (parallel execution)
a method of processing that divides a large job into multiple smaller jobs that can be executed simultaneously on multiple CPUs. *See also* threading.

parameter file
a document that contains the data that SPD Server needs to perform its functionality. SPD Server uses a libnames.parm parameter file and an spdsserv.parm parameter file.

partition
part or all of a logical file that spans devices or directories. A partition is one physical file. Data files, index files, and metadata files can all be partitioned, resulting in data partitions, index partitions, and metadata partitions, respectively. Partitioning a file can improve performance for very large tables. *See also* data partition.

pass-through facility
See SQL pass-through facility.

password database
registers users to enable access to SPD Server. The database stores each user ID and password, a user's ACL group memberships, authorization level, performance class, account expiration information, and server access records.

password database utility
a component that creates and manages the password database, and that enables users to access SPD Server. It is an interactive command-line utility that begins with the psmgr command.

primary path
the location in which metadata files are stored, and the default location for other component files. Typically, the other component files (data files and index files) are stored in separate storage paths in order to take advantage of the performance boost of multiple CPUs.

RAID (redundant array of independent disks)
a type of interleaved storage system that comprises multiple disks to store large amounts of data inexpensively. RAIDs can have several levels. For example, a level-0 RAID combines two or more hard drives into one logical disk drive. Various RAID levels provide different amounts of redundancy and storage capability. Also, because the same data is stored in different places, I/O operations can overlap, which can result in improved performance. *See also* redundancy.

record-level locking
locking at the record level in a table or data set. The user who owns the lock has exclusive access to a single record, while other users can access other records in the same table or data set.

redundancy
a characteristic of computing systems in which multiple interchangeable components are provided in order to minimize the effects of failures, errors, or both. For example, if data is stored redundantly (in a RAID, for example), then if one disk is lost, the data is still available on another disk.

redundant array of independent disks
See RAID.

RLS
See row-level security.

row-level security (RLS)
a security feature that controls access to rows in a table in order to prevent users from accessing restricted data.

SAS format (format)
a type of SAS language element that is used to write or display data values according to the data type: numeric, character, date, time, or timestamp.

SAS function (function)
a type of SAS language element that is used to process one or more arguments and then to return a result that can be used in either an assignment statement or an expression.

SAS informat (informat)
a type of SAS language element that is used to read data values according to the data's type: numeric, character, date, time, or timestamp.

SAS Management Console
a Java application that provides a single user interface for performing SAS administrative tasks.

SAS Metadata Server
a multi-user server that enables users to read metadata from or write metadata to one or more SAS Metadata Repositories.

SAS Scalable Performance Data Server (SPD Server)
a server that restructures data in order to enable multiple threads, running in parallel, to read and write massive amounts of data efficiently.

sasroot
a representation of the name for the directory or folder in which SAS is installed at a site or a computer.

SASSPDS
the SAS engine that provides access to SPD Server.

scalability
the ability of a software application to function well and with minimal loss of performance, despite changing computing environments, and despite changes in the volume of computations, users, or data. Scalable software is able to take full advantage of increases in computing capability such as those that are provided by the use of SMP hardware and threaded processing. *See also* scalable software, server scalability.

scalable software
software that responds to increased computing capability on SMP hardware in the expected way. For example, if the number of CPUs is increased, the time to solution for a CPU-bound problem decreases by a proportionate amount. And if the throughput of the I/O system is increased, the time to solution for an I/O-bound problem decreases by a proportionate amount.

Secure Sockets Layer
See SSL.

serde
an interface that enables serialization or deserialization of one or more file formats.

server scalability
the ability of a server to take advantage of SMP hardware and threaded processing in order to process multiple client requests simultaneously. That is, the increase in computing capacity that SMP hardware provides increases proportionately the number of transactions that can be processed per unit of time. *See also* threaded processing.

session
a single period during which a software application is in use, from the time the application is invoked until its execution is terminated.

SMP (symmetric multiprocessing)
a type of hardware and software architecture that can improve the speed of I/O and processing. An SMP machine has multiple CPUs and a thread-enabled operating system. An SMP machine is usually configured with multiple controllers and with multiple disk drives per controller.

sort indicator
an attribute of a data file that indicates whether a data set is sorted, how it was sorted, and whether the sort was validated. Specifically, the sort indicator attribute indicates the following information: 1) the BY variable(s) that were used in the sort; 2) the character set that was used for the character variables; 3) the collating sequence of character variables that was used; 4) whether the sort information has been validated. This attribute is stored in the data file descriptor information. Any SAS procedure that requires data to be sorted as a part of its process uses the sort indicator.

spawn
to start a process or a process thread such as a light-weight process thread (LWPT). *See also* thread.

SPD Server
See SAS Scalable Performance Data Server.

SPD Server STARJOIN Facility (STARJOIN Facility)
a component of the SPD Server SQL Planner that optimizes N-way star schema joins for qualified SPD Server tables.

SPDO procedure
the operator interface for SPD Server. The procedure defines and manages SPD Server ACLs, defines row-level security for tables, manages proxies, defines and manages cluster tables, refreshes server parameters and domains, performs table management functions such as truncating tables, and executes SPD Server utilities from a central point.

SPDSBASE process
accesses or creates SPD Server resources for users. Several SPDSBASE processes can be active simultaneously in an SPD Server installation, handling work requests for different users or different SAS sessions. The SPDSBASE process can take on the role of either an SPD Server user proxy, an SPD Server SQL proxy, or an SPD Server SQL user proxy.

spdsclean
See directory cleanup utility.

spdsserv.parm file
an SPD Server parameter file that defines the server configuration and performance parameters that control processing behavior and use of resources.

SQL pass-through facility (pass-through facility)
the technology that enables SQL query code to be passed to a particular DBMS for processing. *See also* record-level locking.

SQL query rewrite facility
examines SQL queries to optimize processing performance. When an SPD Server user submits SQL statements that contain subexpressions, the SQL query rewrite facility optimizes the SQL query when possible.

SSL (Secure Sockets Layer)
an encryption protocol for securing client/server communication. *See also* Transport Layer Security.

star schema
tables in a database in which a single fact table is connected to multiple dimension tables. This is visually represented in a star pattern. SAS OLAP cubes can be created from a star schema.

STARJOIN Facility
See SPD Server STARJOIN Facility.

symbolic variable
See macro variable.

symmetric multiprocessing
See SMP.

thread
the smallest unit of processing that can be scheduled by an operating system.

thread-enabled operating system
an operating system that can coordinate symmetric access by multiple CPUs to a shared main memory space. This coordinated access enables threads from the same process to share data very efficiently.

thread-enabled procedure
a SAS procedure that supports threaded I/O or threaded processing.

threaded I/O
I/O that is performed by multiple threads in order to increase its speed. In order for threaded I/O to improve performance significantly, the application that is performing the I/O must be capable of processing the data rapidly as well. *See also* I/O-bound application, thread.

threaded processing
processing that is performed in multiple threads in order to improve the speed of CPU-bound applications. *See also* CPU-bound application.

threading
a high-performance technology for either data processing or data I/O in which a task is divided into threads that are executed concurrently on multiple cores on one or more CPUs.

time to solution
the elapsed time that is required for completing a task. Time-to-solution measurements are used to compare the performance of software applications in different computing environments. In other words, they can be used to measure scalability. *See also* scalability.

TLS
See Transport Layer Security.

Transport Layer Security (TLS)
the successor to Secure Sockets Layer (SSL), a cryptographic protocol that is designed to provide communication security. TLS uses asymmetric cryptography for authentication and confidentiality of the key exchange, symmetric encryption for data/message confidentiality, and message authentication codes for message integrity. SAS uses TLS to secure communication between SAS clients and servers.

WHERE clause
a syntax string that includes the keyword WHERE, followed by one or more WHERE expressions. A WHERE clause defines the conditions to be used for selecting observations in a data set. *See also* WHERE expression.

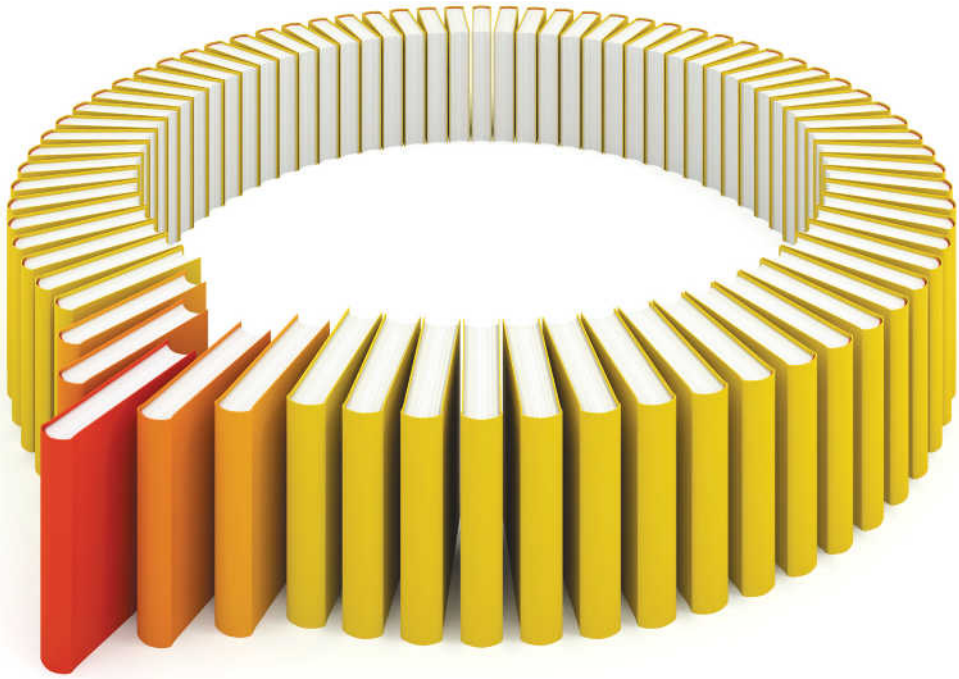
WHERE clause planner
uses factors of cardinality and distribution to calculate relative processor costs of various WHERE clause options. The SPD Server WHERE clause planner avoids computation-intensive operations and uses simple computations where possible.

WHERE expression
is a syntax string within a WHERE clause that defines the criteria for selecting observations. For example, in a membership database, the expression "WHERE member_type=Senior" returns all senior members. *See also* compound WHERE expression, WHERE processing.

WHERE processing
a method of conditionally selecting rows for processing by using a WHERE expression. *See also* WHERE expression.

workspace tables
a list of paths that contain temporary SPD Server work tables and temporary intermediate files that are associated with the declared domain.

Index



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.