Best Practices for Using the SAS® Scalable Performance Data Server® in a SAS® Grid environment

Introduction

This document describes how to set up the SAS® Scalable Performance Data Server®, SPD Server, to run in a SAS Grid environment. SPD Server can be used in a grid for both data loading and query. Adding SPD Server to a grid increases the availability of the data service and provides a scalable environment for SAS users.

Note: SAS used SPD Server and SAS Grid Manager when executing the previous two SAS® Data Integration Server world records.

Setup overview

It is recommended that you have significant experience with setting up SPD Server before configuring it to run on a SAS Grid. In setting up SPD Server to run on a SAS Grid, each grid node requires its own instance of the product. These instances share the SPD Server executables, all metadata, indexes and actual data files. Sharing is accomplished by having a clustered (shared) file system setup between all the SAS Grid nodes. SAS processes executing on each grid node communicate only with their local SPDS server installed on the same grid node.

Requirements for setup

Clustered / shared file system - all SAS Grid nodes that require access to SPD Server (via libname) require access to both the executable and the data. Clustered file systems include but are not limited to: IBM's GPFS, NFS, Linux GFS, Sun's QFS and Veritas CFS. File system logical layout must be the same on every grid node. Example: /spdsdata, which points to a shared file system, must exist on all grid nodes.

SPDS 4.3 or higher - these are the versions of SPD Server that have been tested on a grid

Homogeneous server installation - all grid nodes must be the same hardware platform and operating system type

Programs wishing to access SPD Server must use libnames with the hostname set to "localhost". SAS processes executing across the grid may not get executed on the same grid node therefore it is important that they communicate only with the local SPD Server

instance. A hostname of localhost forces communication with the local SPD Server instance to eliminate the need to talk over the network.

Each SPD Server instance will share the SPD Server executable, but each grid node will have its own SPD Server “audit” directory, "site" directory and "log" directory. Some simple pathname changes need to be made to the rc.spds file to enable this.

In the Best Practices for Configuring your IO Subsystem for SAS® Scalable Performance Data Server® Tables paper from SAS Technical Papers you will learn that SPD Server tables are similar in nature to SAS data files, but the number of file systems required to support them is very different. This paper is designed to augment the SAS Global Forum 2007 paper Best Practices for Configuring your IO Subsystem for SAS® Applications by covering additional information that pertains to SPD Server tables.

Detailed setup directions

1. Setup a shared file system between all servers in the grid. Sample file systems to share

`/shared/sas`	location of foundation SAS (SAS_9.1), SPD Server
`/shared/data1`	location of shared data for all servers
`/shared/home`	location of home directories for users
`/shared/work (optional)`	some people prefer to put saswork on the local server and not share it
`/shared/spdsdata`	spd server data directory
`/shared/spdsmeta`	spd server meta data directory
/shared/spdsindex	spd server shared index directory

The size of these file systems will depend upon your needs.

SPD Server temporary directories cannot be shared, but they could be a subdirectory on a shared file system if you want them on the SAN storage.

2. Set up SAS and SAS Grid Manager. These product binaries can be shared across all nodes.

Example:

shared/sas/SAS_9.x

3. Set up SPD Server on one of the grid servers in the shared SAS directory (example: /shared/sas/spds44). Install and test to make sure it works correctly. Use and test your shared file system as the data, index and SPD Server metadata directories. Making sure this all works on a single server first is critical.

4. Make a copy of the site directory, one for each grid node that will be running SPD Server.

For example:

gridnode0 /shared/sas/spds/site0
gridnode1 /shared/sas/spds/site1
gridnode2 /shared/sas/spds/site2
gridnode3 /shared/sas/spds/site3

5. Make a log directory for each grid node that will be running SPDS

For example:

gridnode0 /shared/sas/spds/log0
gridnode1 /shared/sas/spds/log1
gridnode2 /shared/sas/spds/log2
gridnode3 /shared/sas/spds/log3

6. Make an Audit directory for each grid node that will be running SPD Server

For example:

gridnode0 /shared/sas/spds/Audit0
gridnode1 /shared/sas/spds/Audit1
gridnode2 /shared/sas/spds/Audit2
gridnode3 /shared/sas/spds/Audit3

7 Modify the rc.spds file in each of the site directories

(a) change the LOGDIR variable as follows:

for gridnode0 LOGDIR=$INSTDIR/log0
for gridnode1 LOGDIR=$INSTDIR/log1
for gridnode2 LOGDIR=$INSTDIR/log2
for gridnode3 LOGDIR=$INSTDIR/log3

(b) use a shared libnames.parm file as this will keep you from having to edit a different libnames.parm for each gridnode

in rd.spds set the LPARM environment variable:

Example: LPARM=$INSTDIR/libnames.parm

8. Start each server with the rc.spds script. Check to see that all servers are up and running.

9. Create a master script that can remotely start and stop all SPD Server servers. This can be done with rsh if permissions for the SPD Server user allow remote execution. For example: via rhosts on UNIX).

When launching SAS programs on the grid, be sure to use a "localhost" enabled libaname:

Example: Libname saslib SASSPDS IP=YES LIBGEN=YES USER="ANONYMOUS" schema="saslib" Serv="5190" HOST="localhost";

Using localhost causes the SAS program to use the local instance of SPD Server on that grid node.

Calculating Partition Size for SPD Server Tables

In the white paper Partitioning of SAS® Scalable Performance Data Server® Tables we discuss setting and controlling the data partition size of the SPDS Server tables. Setting and controlling the partition size are the most basic functions for managing data tables that are stored with the SPD Server.

In Appendix A we provide a sas macro to calculate the correct partition size of the SPD Server table. You must set the correct partition size prior to creating the SPD Server table. Once the partition size is set, you cannot change it.

Loading SPD Server Tables

When loading an SPD Server table always sort the table by the variables used to JOIN that SPD Server table to other SPD Server tables. To improve performance of filtering data add the variables used in WHERE clauses to the sort. Put these variables after the join variables. If you have queries that do aggregates or counts add these variables to the sort after the join variables. When ordering the “where, aggregations, and counts” variables in the sort routine go from lowest cardinality to highest cardinality.

Always use compression on SPD Server domains. You can use the LIBNAME option COMPRESS=YES or the SPD Server macro %let spdsdcmp=YES to ensure all tables written to an SPD Server domain are compressed.

Always replace PROC APPEND with SPD Server Cluster tables. The white paper Scalability Solution for SAS® Dynamic Cluster Tables provides an overview of dynamic cluster tables in SPD Server 4.3 as well as enhancements that have been included in later releases. **Note:** Where the enhancements are discussed, this paper also documents their respective releases. Earlier releases of SPD Server supported two types of clustered data

tables: time-based partitioning and partition by value (in an experimental form). In SPD Server 4.3 and later, dynamic cluster tables enable both the partitioning of data based on criteria in the data and parallel loading of the cluster tables. Dynamic clusters also enhance the manageability of tables. New table options, as well as SQL planner enhancements, have been added to take advantage of these new capabilities and to improve query performance.

Always replace DATA STEP modify code with PROC APPEND using the SPD Server option (UNIQUESAVE=REP). To use PROC APPEND with (UNIQUESAVE=REP) the SPD Server table must have a unique index.

Quering SPD Server Tables

Always use the following **LIBNAME** options

- IP=YES
  - If the ANSI standard SQL submitted to SPD Server would run faster in pass-through sql, the query will be converted to pass-through sql on the fly,
- LIBGEN=YES
  - Required when SAS Marketing Automation uses SPD Server tables
  - A sql planner that is used when certain queries are submitted to SPD Server.
- DISCONNECT=YES
  - Kills the SPD Server proxy server when the LIBNAME is cleared.
- COMPRESS=YES
  - Ensures compression is used when writing to the SPD Server table.

Using SPD Server to Manage the Transient needs of SAS End-Users

SAS end-users require disk space to hold temporary data sets. To accomplish this IT personnel must pre allocate disk space per SAS end-user. Some companies allocate 100GB per SAS end-user for SASWORK. This causes IT to spend cycles maintaining symbolic links or ACLs at the operating system level per sas end-users. Another issue is that half of the sas end-users are always pushing the limits of the 100GB while the other half of the sas end-users rarely us any of their 100GB

Implementing SPD Server and moving the disk space allocated to SASWORK per SAS end-user to the SPD Server domain USER accomplishes two things. One, it provides more space to the sas end-users that were bumping up to the limit of the file system and reduces the total amount of disk space allocated to the true transient needs of SAS end-users. Two, it leaves enough space in the home directories of the SAS end-user to store their sas code, EG projects, EM projects and other SAS projects but not SAS data sets. This has a side of effect of stopping the proliferation of SAS data sets because SAS end-users cannot use their directories to store permanent sas data sets. If the end-user requires permanent sas data sets that space would be allocated to a SPD Server Domain and the ACLs on that domain would allow the SAS end-user to create permanent SPD Server tables.

Setting up the SPD Server Domain User

The transient space is allocated to the SPD Server domain called USER. To accomplish this you would modify your LIBNAMES.PARMS to have a domain called USER:

```
libname=user pathname=/spds1/metadata/user
   roptions="datapath=('/spds2/data/user'
                       '/spds3/data/user'
                       '/spds4/data/user')
             indexpath=('/spds1/indexes/user')";
```

SAS end-user example

To enable SAS end-users to use the transient space they must submit a LIBNAME statement to point to the SPD Server domain USER that has the LIBNAME option TEMP=YES. TEMP=YES will create a sub-directory in the USER domain for each SAS end-user. All transient data is stored in this newly create subdirectory. Once the SAS session ends, that sub-directory and all of it contents are deleted
Example SAS libname statement to point to SPD Server transient domain USER

```
libname USER sasspds "user”
    TEMP=yes
```

```
 IP=yes
LIBGEN=yes
HOST="eecsun5z5"
SERV="5200"
USER="anonymous";
```

All one level names used in sas code are automatically written to the SPD Server domain USER. For example:

```
data TABLENAME;
   set LIBREF.TABLENAME;
run;

proc print data=TABLENAME; RUN;
        * would run successful, SPD Server table processed is
        USER.TABLENAME;

proc print data=USER.TABLENAME; RUN;
        * would run successful,  SPD Server table processed is
        USER.TABLENAME;

proc print data=WORK.TABLENAME; RUN;
        * would FAIL because the SAS data set WORK.TABLENAME
        does not exist;
```

Conclusion

Moving the disk space allocated to SASWORK into the SPD Server domain USER has the following benefits:

a. Only one operating system id is used to write and read data from the operating systems file systems. This reduces IT costs by eliminating the need to allocate/maintain permanent and transient disk space on a per SAS end-user bases
b. A true understanding of how much transient space is required for your SAS end-users will be gained over time.
c. The SPD Server audit logs provide usage information on all tables in SPD Server.
d. The proliferation of SAS data sets will not occur.
e. Increased security because your data is locked down inside of SPD Server and can only be accessed when appropriate SPD Server credentials are provided.

Appendix A

Data Driven SPD Server Partition Size

Introduction

The macro documented in this appendix was created to provide a data driven process to calculate and set the correct partition size for a new SPD Server table.

This macro has the following parameters:

- SOURCE - Source table being loaded into SPD Server
- PARTS - SAS recommends 2 parts per file system allocated to the data component of an SPD Server domain
- FILESYSTEMS - the number of file systems dedicated to the data component of the SPD Server domain. refer to !spdsroot/site/LIBNAMES.PARM
- FILE - a temp file that is used to store the SPDS statement used to set the correct partition size
- NOBS - Data driven, the number of rows in the source table
- LRS -Data driven, the logical record length of the source table

This macro creates a report containing information about the source table and the SPD Server table.

Notes/How to

This macro has been tested on the Solaris and Windows platforms.

Code

```
* Calculate SPDS Partition Size v 1.3___________________
* Source
*       Type: String
*      Group: General
*      Label: Source table being loaded into SPD Server
*       Attr:  Modifiable, Required
*_____________________________________________________
* PARTS
*       Type: Numeric
*      Group: General
*      Label: SPD Server R&D recommends 2 parts per
*             file system allocated for the data component
*             of an SPD Server table
*       Attr:  Modifiable, Required
*_____________________________________________________
* FILESYSTEMS
*       Type: String
*      Group: General
*      Label: the number of files system dedicated to the
*             data component of the SPDS table, refer to
*             LIBNAMES.PARM
*       Attr:  Modifiable, Required
*_____________________________________________________
* FILE
*       Type: String
*      Group: General
*      Label: a temp file that is used to store the SPDS
*             statement to set the correct partition size
*       Attr:  Modifiable, Required
*_____________________________________________________
* NOBS
*       Type: Numeric
*      Group: General
*      Label: the number of rows in the source table being
*             loaded into the SPDS table
*       Attr: Data driven
*_____________________________________________________
* LRS
*       Type: Numeric
*      Group: General
*      Label: the logical record length of the source table
*             being loaded into the SPDS table.
*       Attr: Data driven
*_____________________________________________________;
%global lrs nobs partsize spdssize;
%macro
partsize(SOURCE=sashelp.voption,PARTS=2,FILESYSTEMS=10,FILE="/tmp/&sysu
serid..partsize.sas");
       proc contents
              data=&Source
              out=contents (keep=length nobs)
              noprint;
       run;

       data _null_;
```

```
                set contents end=done;
                lrs + length;
                if done then do;
                   call symput ('lrs',trim(left(put(lrs,8.))));
                   call symput ('nobs',trim(left(put(nobs,8.))));
                end;
        run;

        %if &nobs eq . %then %do;
                proc sql;
                        create table nobs as select count(*)as nobs from
&SOURCE;
                quit;
                data _null_;
                        set nobs;
                        call symput ('nobs',trim(left(put(nobs,8.))));
                run;
        %end;


        data partsize;
                file &FILE;
                format observations logical_record_length
rows_per_filesystem file_systems
               rows_per_partition parts bytes kbytes mbytes gbytes
tbytes comma24.0;
                observations=&NOBS;
                file_systems=&FILESYSTEMS;
                logical_record_length=&LRS;
                parts=&PARTS;
                rows_per_filesystem=ceil(observations/file_systems);
                rows_per_partition=rows_per_filesystem/&PARTS;
                partsize=CEIL((rows_per_filesystem *
logical_record_length) / (1024*1024));
                partsize=CEIL(partsize/&parts);
                if partsize <=16 then partsize=16;
                bytes=&nobs*&lrs;
                kbytes=bytes/1024;
                mbytes=bytes/1024**2;
                gbytes=bytes/1024**3;
                tbytes=bytes/1024**4;
                cpartsize=partsize || 'M';
                call symput
('partsize',trim(left(put(cpartsize,$30.))));
                put '%let spdssize=' cpartsize ';';
        run;
        options source2;
        %inc &FILE;
        options nosource2;
        %put For the table &Source there are &NOBS rows with a logical
record length of &LRS bytes.;
        %put Creating &parts partitions per data path, the partition
size for &Source is &PARTSIZE;
        title "SPD Server Parition Size for &Source is &PARTSIZE";
        proc print uniform Label;
                Label parts='Desired Number of Data Components per File
System per Table'
```

```
               rows_per_filesystem='Number of Rows per File System'
               rows_per_partition='Number of Rows per Partition'
                       Observations='Number of Rows in Table'
                       file_systems='Number of File Systems for SPDS
Data Component'
                       partsize='Partition size of SPD Server Table'
                       logical_record_length='Logical Record length'
                       bytes='Bytes'
                       kbytes='Kilobytes'
                       mbytes='Megabytes'
                       gbytes='Gigabytes'
                       tbytes='Terabytes';
               var partsize observations logical_record_length
file_systems parts rows_per_filesystem
            rows_per_partition bytes kbytes mbytes
            gbytes tbytes;
        run;

        title;

%mend;
%partsize(SOURCE=train.sales_fact_2004,PARTS=2,FILESYSTEMS=10,FILE="c:\
temp\&sysuserid..partsize.sas");
```