Open Systems Solutions to Large File Requirements

Tom Truscott, SAS Institute Inc., Cary, NC

ABSTRACT

Open Systems (UNIX and UNIX-like systems) have widely different levels of support for files that exceed two gigabytes. Some vendor systems support "large files", other systems do not support them at all, and yet others have special extensions which programs may use to create and operate on large files. Regardless of the level of underlying support, the SAS® System will give Open Systems users the capability to create and manage truly monstrous data sets and related files.

INTRODUCTION

UNIX systems on 32-bit CPU architectures typically have a two gigabyte limit on the maximum size of any file. This limit is becoming a serious problem as disk drive prices fall and as data warehousing, decision support, and other enterprise applications move towards UNIX. It has forced SAS users to partition their files (e.g. by geographic region or date range). The limit is particularly burdensome for users attempting to move existing mainframe data sets which already are well in excess of two gigabytes.

Numerous SAS users have requested support for "large files" on Open Systems. Most prospective users of large files have a minimum configuration of a mid-range server (such as a Digital AlphaServer 2100, IBM RS6000/I30, HP 9000-K410, or Sun SPARCserver 1000) with multiple CPUs, at least 256 megabytes of RAM, and somewhere in excess of 25 gigabytes of disk space.

The two gigabyte file size limit is due to the operating system's use of 32-bit integers to hold file size related information. The largest value that a signed 32-bit integer can represent is $2^{31} - 1$, or 2147483647. For example the stat subroutine call, which gets status information about a file, returns the size of the file in a 32-bit integer and so it cannot correctly report the size of a file that is larger than two gigabytes. Similarly the lseek subroutine call, which positions the read/write file pointer for random access I/O, is limited to accessing the first two gigabytes of a file. Several other file system subroutine calls have this type of problem and the result is that "large files" are not available on such operating systems.

UNIX system vendors have come up with two different solutions for removing the two gigabyte file size limit. One solution is to "fix" the file system subroutine calls to use 64-bit integers for file size related values. Signed 64-bit integers can represent values up to $2^{63} - 1$, or 9223372036854775807. (The disk space needed to store a file that large would cost two trillion US dollars at today's prices.) This is a particularly clean solution for the new 64-bit CPU architectures such as the DEC Alpha.

System vendors that rely on 32-bit CPU architectures need to maintain binary compatibility with existing applications and have a more difficult problem if they want to support large files. Changing the file system subroutine calls to use 64-bit integers would create an incompatibility with the existing applications that expect to interact with the file system using 32-bit integers. Consequently, system vendors have developed an extended set of file system subroutine calls that support large files and have left the existing calls alone to maintain backwards compatibility. For example the new set of calls might include stat64 and lseek64, which would behave just like stat and lseek but would use 64-bit integers for file size related values and so could be used on large files.

Unfortunately, some system vendors have chosen to do nothing about the "large file" problem. Since the system vendors have addressed this problem in three different ways, and SAS users want large file support regardless, SAS Institute has chosen to solve this problem for its customers in three different ways!

CASE 1: STANDARD OS SUPPORT

The DEC Alpha running Digital UNIX is a new 64-bit operating system that never had a two gigabyte file size limit. The SAS System has been ported to the Alpha and supports arbitrarily
large data sets. All that SAS users need to do to create large files on the Alpha is to buy enough disk drives to hold them.

Porting the SAS System to the Alpha was not a trivial task. Although the SAS System consists primarily of highly portable C language code, there were several coding mistakes that were exposed during the port. For example, there was code that relied on the integer data types int and long int being the same, which indeed is true for all other current SAS ports. But on the Alpha an int is a 32-bit integer, and a long int is a 64-bit integer. The good news is that now that the SAS system source code has been cleaned up for the DEC Alpha, ports to additional 64-bit architectures should go more smoothly.

Other system vendors such as SGI also have 64-bit operating systems but they have an even larger installed base of 32-bit systems. For backwards compatibility the 64-bit systems are able to run programs compiled for the 32-bit systems. Rather than supply separate 32-bit and 64-bit versions of the SAS System in such cases, SAS Institute instead supplies a single 32-bit version which has been "extended" to support large files.

CASE 2: "EXTENDED" OS SUPPORT

As mentioned earlier, several system vendors started to develop new file system subroutine calls that support large files and they left the existing calls alone to maintain backwards compatibility. The problem was that there was no standard for doing this, and so each vendor wrote a slightly different set of calls! This hurts portability and makes SAS Institute's job harder. To combat this problem, SAS Institute and the UnixWare Technology Group co-sponsored a series of "Large File Support" summit meetings that were attended by representatives of more than thirty system vendors. The meetings have resulted in widespread agreement to adopt a common specification for large file support on 32-bit architectures.

Besides the agreement on the names for the new calls (stat64, lseek64, truncate64, ...) there was an agreement that old binaries should be prevented from operating on large files, as otherwise they would almost certainly malfunction and in potentially serious ways. There was also agreement on a list of utilities (such as cp and cmp which copy and compare files) that are required to operate correctly on large files.

Because the Large File Support specification is just now being implemented, it is currently an experimental interface in SAS 6.11 for MIPS and Intel ABI systems. SAS users must also specify the -largefile saslf6s option to request that the large file subroutine calls be used.

CASE 3: NO OS SUPPORT

It will be several years before most of the installed base of UNIX systems will be large file capable. As an interim solution the SAS System will support large data sets by logically grouping several files together and managing them as if they were a single file. These file "partitions" will typically be spread out over several directories in different file systems. File partitioning is done in a given library for all data sets, indexes, catalogs, and views.

The syntax for a partitioned libname is:

\[\text{libname } iname\left(\text{path1}, \text{path2} \ldots\right)\]
\[\text{type=partition [partsize=#]} ;\]

The paths may be either UNIX directory names or previously declared libnames, including other partitioned libnames. If partsize (the size at which to split files) is omitted a default value will be used. If partsize is specified then type becomes optional as it will be assumed. The number specified as the partition size may have an optional suffix of k, m, or g) to indicate that the size is in units of kilobytes, megabytes, or gigabytes.

For example, suppose a system has two gigabyte disk drives mounted on the directories /disk1 and /disk2. Then the SAS System can treat them as a single library with 3.6 gigabytes of disk space by using the syntax:

\[\text{libname big\left(\text{/disk1}, \text{/disk2}\right)}\]
\[\text{type=partition partsize=1800M} ;\]

The first partition of a partitioned file resides in the first directory of the partitioned libname and has a normal SAS data set name such as x.ssd01). Subsequent partitions reside in subsequent directories and are have names of the form x.ssd01.1 where the trailing number increases with each directory.

The SAS System -work option also supports the concatenated libname syntax to indicate that
the work library is partitioned. This will usually be necessary to support large data sets (e.g. to sort them).

The default partition size for all partitioned lib-

names (including the work library) is 500M, but that may be changed with the new -partsize option.

Some vendors provide file systems that can store far more than two gigabytes, although no single file can exceed that size. In such a case the user need only specify one directory path for all file partitions.

File partitioning may also be useful on smaller systems as a way to exploit free disk space that is scattered across multiple systems. But file partitioning should be used sparingly as it complicates system administration issues such as backup and restore. Use of multiple directories increases the chances of problems with reliability and file consistency. Partitioned libnames should be carefully documented so that confusion is minimized when problems arise.

PERFORMANCE

The performance of SAS on large files does not seem to depend on which of the above methods are used for accessing them. For example, performance on a three gigabyte data set is about the same whether it is stored in a single huge file or in a partitioned libname. Of much greater importance is the speed of the disk I/O system. On busy multi-user systems (such as servers) it may also help to have multiple CPUs, and it may also be important to "tune" the file storage system. Many UNIX systems now include tools such as a "logical volume manager" which can interleave files (and file systems) across multiple disks, can "mirror" file systems to provide increased performance and reliability, and so on. If the system already has 128 megabytes of RAM then adding another 128 might result in only a minor speedup whereas adding a few gigabytes of RAM (enough that the large files reside entirely in memory) might yield a spectacular speedup. This same phenomenon also occurs with smaller files, but much less expensively. It depends on how well the operating system manages the available RAM and other factors, so your mileage may vary.

In one test of large file performance, an SGI Challenge L was used to perform various operations on partitioned 2.1 gigabyte data sets and the elapsed times required were noted. The SAS memsize was 64M, sortsize was 32M, and partsize was 1G for both the data set and work libraries.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the data set</td>
<td>18</td>
</tr>
<tr>
<td>Read the data set, write a new one and compare the two</td>
<td>160</td>
</tr>
<tr>
<td>Sort in descending order</td>
<td>130</td>
</tr>
<tr>
<td>Create an index for the data set</td>
<td>210</td>
</tr>
<tr>
<td>Subset (where clause) of the even numbered obs.</td>
<td>78</td>
</tr>
<tr>
<td>PROC MEANS, producing mean and summary of the variable</td>
<td>150</td>
</tr>
<tr>
<td>Using SAS/CONNECT, download the data set and compare to the original</td>
<td>180</td>
</tr>
<tr>
<td>Create an SQL table of the data set and compare to the original</td>
<td>190</td>
</tr>
</tbody>
</table>

Sorting the data set required 5.2 gigabytes of free space in the work library. A general rule is that sorting requires work library free space of at least 2.5 times the size of the data set being sorted. Sorting also requires data set library free space of at least the size of the sorted data set (2.1 gigabytes in this test), even if the original data set is being replaced. This is because, to guard against possible data loss, the sort writes out the entire new sorted data set before deleting the original. Creating an index may also involve a sort, as it did in this test.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the data set</td>
<td>18</td>
</tr>
<tr>
<td>PROC FREQ</td>
<td>11</td>
</tr>
</tbody>
</table>
AVAILABILITY

The SAS System for Convex OS and Digital UNIX both fully support files larger than two gigabytes. The \texttt{-largefile} extension and partitioned libname features are available as experimental interfaces in the SAS System 6.11 for MIPS and INTEL ABI operating systems. It is our intention to support these in maintenance releases on other platforms after they have been more fully tested.

CONCLUSION

SAS System support for files larger than two gigabytes permits users to more fully exploit the capabilities of their computing environment. Whether you are migrating from the mainframe, growing an existing database, or doing serious data mining, the possibilities are unlimited.

More information on the Large File Support specification is available at

\begin{center}
http://www.sas.com/standards/
\end{center}

ACKNOWLEDGEMENTS

Carl Zeigler led the Open Systems large file effort at SAS Institute, and was a co-organizer of the Large File Summit meetings. Bob Huemer ported the SAS System to the DEC Alpha. Deborah Morgan designed and implemented the support for SAS System access to large file sub-routine calls. Larry Noe designed and implemented the SAS System support for large files via partitioned libnames. Eric Coates tested the large file support on the various Open Systems platforms.

TRADEMARKS

SAS and SAS/CONNECT are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. \textregistered indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.