Paper 368-2012

Practice on the SAS® Scalable Performance Data Server Configuration for Maximum Performance

Stephen Bogacz, Danni Luo, Greg Dorfner, David Bianchi, Prime Therapeutics LLC, Eagan, MN, USA

ABSTRACT

To ensure the SAS data ware house environment to be architected to meet the need of volume increases anticipated in the coming years, and to support our daily activities of analytical-oriented exploration, querying, data mining and reporting, we rebuilt our SAS Data Warehouse (SAS DW) in 2010 and 2011. In this project, we constructed six SPDS environments (Dev, QA, UAT, PROD, DR and BI), designed new file systems in each of them. We tuned SPD server parameters, conducted performance test and finally completed data migration. The way SPD Server configured can significantly affect performance. This paper describes our practice on production SPD Server configuration, demonstrates our test result in optimizing performance.

INTRODUCTION

SAS Scalable Performance Data Server (SPD Server) distributes SAS datasets into a number of data partition files (DPFs) across multiple data paths. By striping disks at the file system level true parallelism is achievable at the disk I/O level. When SPD Server is optimally configured, it will use multiple data access threads resulting in high-performance throughput. This paper focuses on the building of our production SPD Server system and the techniques used to configure it. To ensure success we formed a team consisting of two SAS Data Warehouse Administrators, a SAS Administrator, a Unix Administrator and a SAN Administrator.

CONSTRUCT UNIX FILE SYSTEMS FOR SPD SERVER

When we undertook this exercise our SAS Data Warehouse was housed on a Unix AIX server with 4GB fiber channels, 8 CPUs with 64 GB of memory, and SAS9.1 and SPD Server 4.4 installed. The main SAS interface for our users was SAS Enterprise Guide 4.1.

In designing the new SPD Server file systems, we started by determining the amount of space currently used and the estimated data growth per week over the next three years. The resulting estimate was 2.88 TB of disk space needed. Based on the number of CPU's available on the server we settled on 16 independent file systems (data paths) based on the rule-of-thumb of two data paths for each available CPU. The 2.88 TB of space was allocated evenly across the 16 file systems. We also settled on one file system for the Index (325 GB), and one for the Metadata (30 GB).

It is important to create an SPD Server work area for temporary utility work files created during SPD Server sorts, index creation or parallel group by actions. We allocated two file systems of 177 GB each for this work area (/spdswork01 and /spdswork02). These file systems are large enough to accommodate all the concurrent processes that create temp files. SPD Server supports the use of multiple work paths allowing parallel processes to evenly distribute work files and thus allowing the system to scale as user load increases.

One design question we dealt with was whether to put the SPD Server data onto its own Unix server and the users onto another. The upside would be less I/O and CPU contention between user processes and ETL processes accessing SPD Server data. The downside would be increased network traffic between servers where SAS and SAS EG processes were running connecting to the Unix server where SPD Server was running. Ultimately we decided to leave everything on the same server and managed our SAS ETL jobs to run only on weekends and after work hours thus avoiding read/write and I/O conflict with the users.

DISK STRIPING

Per SAS documentation, disk volumes should be hardware striped. Disk striping is a process that carves a disk volume into pieces across multiple disk drive partitions. A stripe size of 64K is commonly used. When striping is performed at the RAID level, the pieces of a space volume from a number of disks within a RAID are striped together. Data can be written in an amount of 64K to these pieces one after another across these disks allowing read/write heads to access a portion of the data on each disk at the same time rather than one read/write head accessing all the data written in a single location on one disk. If a RAID consists of five disks, striping can boost I/O throughput to 64k x 5= 320K. Performance significantly improves when parallelism is achievable at the disk I/O level. To test the concept, we built two file systems in the new server before we laid out the new SPD Server File system—one was disk striped and one was not.

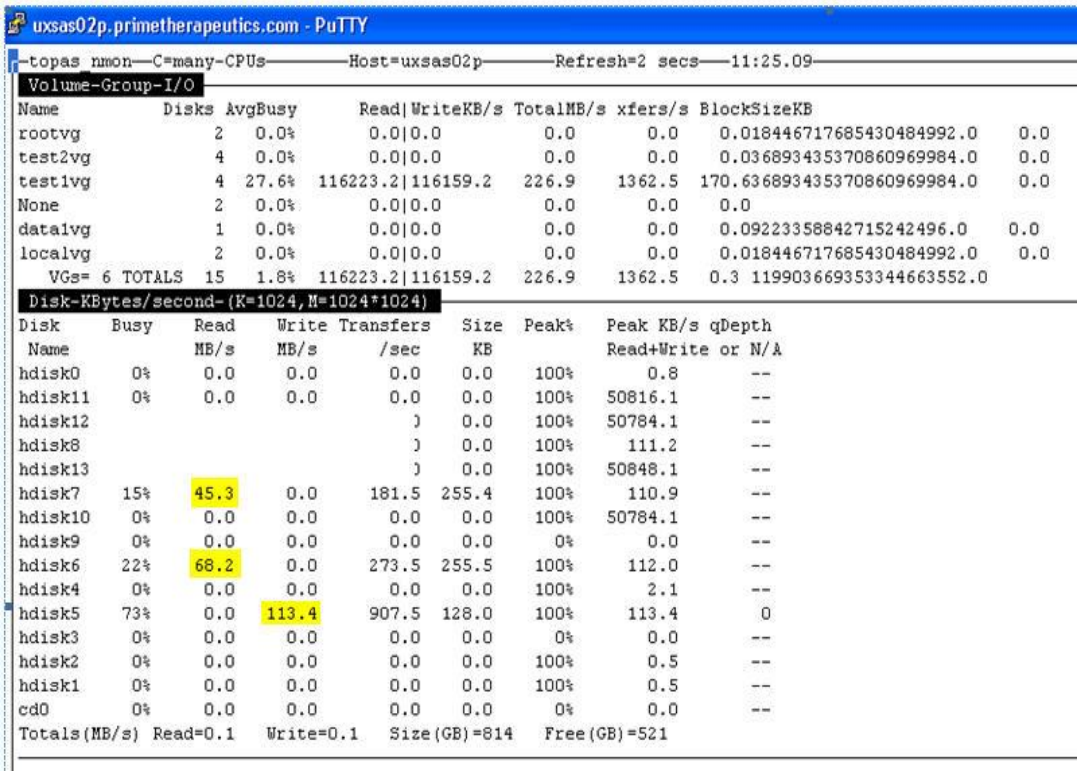
Both tests used a 30GB file with five simultaneous jobs reading one file and writing to five others. Since the data volume and number of jobs to run were small, we did not see a significant difference in run-time. However, we noticed an important difference between the two configurations. In the disk striped file system, Reads/Writes were equally distributed among disks.

TESTVG1 – WITHOUT DISK STRIPING VS TESTVG2 – WITH DISK STRIPING

Testvg1 – nohup sas /home/prodops/stest/stest_testvg1.sas &

```
NOTE: Libref TEST1 was successfully assigned as follows:
      Engine:        V9
      Physical Name: /test1
6
7          proc sql;
8            create table test1.member_elig_x as
9            select * from test1.member_elig;
NOTE: Table TEST1.MEMBER_ELIG_X created, with 123078824 rows and 29 columns.
10           drop table test1.member_elig_x;
NOTE: Table TEST1.MEMBER_ELIG_X has been dropped.
11         quit;
NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           5:00.00
      user cpu time       1:45.15
      system cpu time     3:05.36
      Memory                              2410k
      Page Faults                         437
      Page Reclaims                       63248
      Page Swaps                          0
      Voluntary Context Switches          3599
      Involuntary Context Switches        11797
      Block Input Operations              0
      Block Output Operations             0
```

One disk is read while data from another is pre-fetched with only one disk written to.



uxsas02p.primetherapeutics.com - PuTTY

topas nmon—C=many-CPUs——Host=uxsas02p——Refresh=2 secs——11:25.09

Volume-Group-I/O

Name	Disks	AvgBusy	Read|Write KB/s	TotalMB/s	xfers/s	BlockSizeKB	
rootvg	2	0.0%	0.0|0.0	0.0	0.0	0.018446717685430484992.0	0.0
test2vg	4	0.0%	0.0|0.0	0.0	0.0	0.036893435370860969984.0	0.0
test1vg	4	27.6%	116223.2|116159.2	226.9	1362.5	170.636893435370860969984.0	0.0
None	2	0.0%	0.0|0.0	0.0	0.0	0.0	
datalvg	1	0.0%	0.0|0.0	0.0	0.0	0.09223358842715242496.0	0.0
localvg	2	0.0%	0.0|0.0	0.0	0.0	0.018446717685430484992.0	0.0
VGs= 6 TOTALS	15	1.8%	116223.2|116159.2	226.9	1362.5	0.3 1199036693533446635520	

Disk-KBytes/second-(K=1024,M=1024*1024)

Disk Name	Busy	Read MB/s	Write MB/s	Transfers /sec	Size KB	Peak%	Peak KB/s Read+Write	qDepth or N/A
hdisk0	0%	0.0	0.0	0.0	0.0	100%	0.8	--
hdisk11	0%	0.0	0.0	0.0	0.0	100%	50816.1	--
hdisk12				)	0.0	100%	50784.1	--
hdisk8				)	0.0	100%	111.2	--
hdisk13				)	0.0	100%	50848.1	--
hdisk7	15%	45.3	0.0	181.5	255.4	100%	110.9	--
hdisk10	0%	0.0	0.0	0.0	0.0	100%	50784.1	--
hdisk9	0%	0.0	0.0	0.0	0.0	0%	0.0	--
hdisk6	22%	68.2	0.0	273.5	255.5	100%	112.0	--
hdisk4	0%	0.0	0.0	0.0	0.0	100%	2.1	--
hdisk5	73%	0.0	113.4	907.5	128.0	100%	113.4	0
hdisk3	0%	0.0	0.0	0.0	0.0	0%	0.0	--
hdisk2	0%	0.0	0.0	0.0	0.0	100%	0.5	--
hdisk1	0%	0.0	0.0	0.0	0.0	100%	0.5	--
cd0	0%	0.0	0.0	0.0	0.0	0%	0.0	--

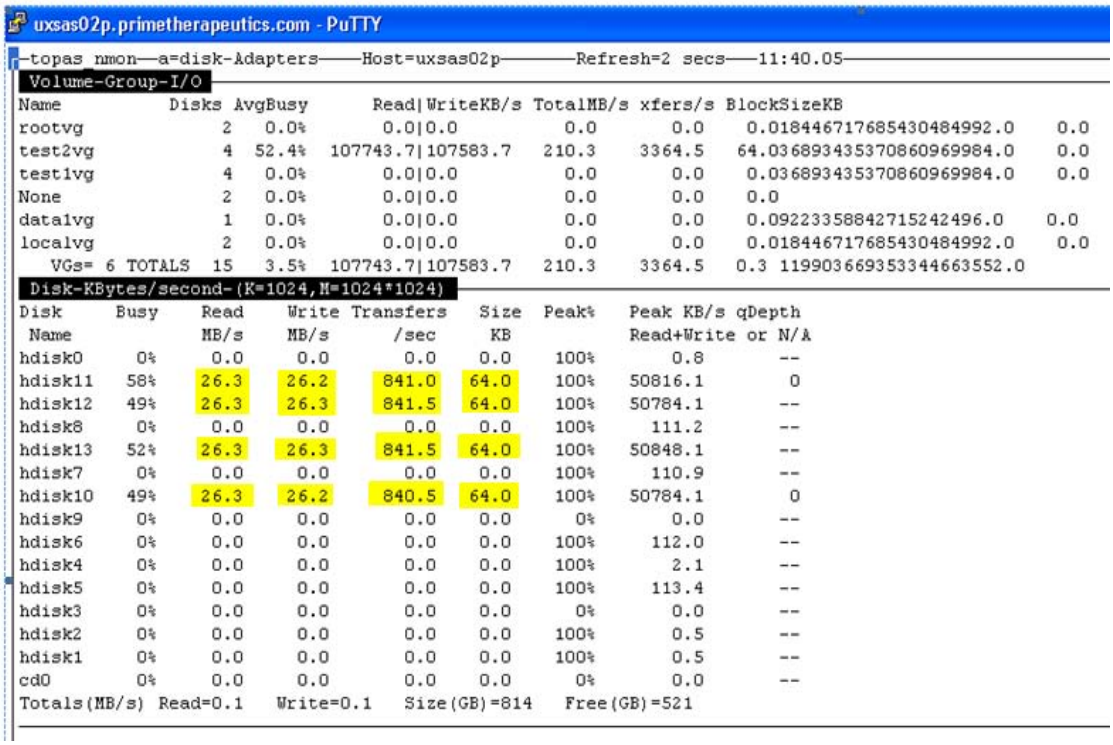
Totals(MB/s) Read=0.1 Write=0.1 Size(GB)=814 Free(GB)=521

Figure 1. Without Disk striping

Testvg2 – nohup sas /home/prodops/stest/stest_testvg2.sas &

```
NOTE: Libref TEST2 was successfully assigned as follows:
      Engine:        V9
      Physical Name: /test2
7          proc sql;
8                   table test2.member_elig_x as
9             select * from test2.member_elig;
NOTE: Table TEST2.MEMBER_ELIG_X created, with 123078824 rows and 29 columns.
10          drop table test2.member_elig_x;
NOTE: Table TEST2.MEMBER_ELIG_X has been dropped.
11         quit;
NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           5:09.00
      user cpu time       1:46.46
      system cpu time     3:11.00
      Memory                              2410k
      Page Faults                         534
      Page Reclaims                       63153
      Page Swaps                          0
      Voluntary Context Switches          3726
      Involuntary Context Switches        20769
      Block Input Operations              0
      Block Output Operations             0
```

Read/Writes equally distributed among disks. True parallelism is achieved at the disk I/O level.



uxsas02p.primetherapeutics.com - PuTTY

topas nmon—a=disk-Adapters——Host=uxsas02p————Refresh=2 secs——11:40.05

Volume-Group-I/O

Name	Disks	AvgBusy	Read|WriteKB/s	TotalMB/s	xfers/s	BlockSizeKB	
rootvg	2	0.0%	0.0|0.0	0.0	0.0	0.018446717685430484992.0	0.0
test2vg	4	52.4%	107743.7|107583.7	210.3	3364.5	64.036893435370860969984.0	0.0
test1vg	4	0.0%	0.0|0.0	0.0	0.0	0.036893435370860969984.0	0.0
None	2	0.0%	0.0|0.0	0.0	0.0	0.0	
data1vg	1	0.0%	0.0|0.0	0.0	0.0	0.09223358842715242496.0	0.0
localvg	2	0.0%	0.0|0.0	0.0	0.0	0.018446717685430484992.0	0.0
VGs= 6 TOTALS	15	3.5%	107743.7|107583.7	210.3	3364.5	0.3 119903669353344663552.0	

Disk-KBytes/second-(K=1024,M=1024*1024)

Disk Name	Busy	Read MB/s	Write MB/s	Transfers /sec	Size KB	Peak%	Peak KB/s Read+Write	qDepth or N/A
hdisk0	0%	0.0	0.0	0.0	0.0	100%	0.8	--
hdisk11	58%	26.3	26.2	841.0	64.0	100%	50816.1	0
hdisk12	49%	26.3	26.3	841.5	64.0	100%	50784.1	--
hdisk8	0%	0.0	0.0	0.0	0.0	100%	111.2	--
hdisk13	52%	26.3	26.3	841.5	64.0	100%	50848.1	--
hdisk7	0%	0.0	0.0	0.0	0.0	100%	110.9	--
hdisk10	49%	26.3	26.2	840.5	64.0	100%	50784.1	0
hdisk9	0%	0.0	0.0	0.0	0.0	0%	0.0	--
hdisk6	0%	0.0	0.0	0.0	0.0	100%	112.0	--
hdisk4	0%	0.0	0.0	0.0	0.0	100%	2.1	--
hdisk5	0%	0.0	0.0	0.0	0.0	100%	113.4	--
hdisk3	0%	0.0	0.0	0.0	0.0	0%	0.0	--
hdisk2	0%	0.0	0.0	0.0	0.0	100%	0.5	--
hdisk1	0%	0.0	0.0	0.0	0.0	100%	0.5	--
cd0	0%	0.0	0.0	0.0	0.0	0%	0.0	--
Totals(MB/s)	Read=0.1		Write=0.1	Size(GB)=814		Free(GB)=521		

Figure 2. With Disk striping

SPD SERVER PARAMETER CONFIGURATION

In building the new SPD Server environment, we reevaluated and reset a number of parameters in SPDSSERV.PARM. The SPDSSERV.PARM file controls performance by defining parameter values which are used by all SAS jobs accessing SPD Server data. SPD Server parameters should be tuned to achieve the users' expectations. Listed below is the list of parameters in the SPDSSERV.PARM file.

```
SORTSIZE=1024M;
INDEX_SORTSIZE=1024M;
GRPBYROWCACHE=1024M;
BINBUFSIZE=32K;
INDEX_MAXMEMORY=30M;
WORKPATH="('/spdswork01' '/spdswork02')";
NOCOREFILE;
SEQIOBUFMIN=64K;
RANIOBUFMIN=4K;
MAXWHTHREADS=8;
MAXSEGRATIO=75;
WHERECOSTING
RANDOMPLACEDPF;
MINPARTSIZE=1024M;
TMPDOMAIN=TMP;
SQLOPTS="reset pllljoin" ;
WHEREAUDIT;
```

PARAMETERS THAT CONTROL MEMORY USAGE

Three parameters (SORTSIZE, INDEX_SORTSIXE and GRPBYROWCACHE) control memory usage. We increased their size to 1024M based on the memory size of the Server and the number of jobs running concurrently. We have

30-50 users daily. Assuming all their jobs are doing any of these types of sorting, each of them consuming 1024MB memory, the memory of the server (64GB) can manage these jobs without memory overflow. The goal is to avoid memory swapping while allowing the max usage for the user jobs. The memory size of 1024MB has met the needs of most users since the implementation of the new SPD Server. There were a few jobs that occasionally consumed memory > 2GB. For those jobs, we directed the users to override the SPD Server parameters in their programs.

SCALE THREADS

A thread is a path of execution through a program. Single threaded programs have one path of execution, and multi-threaded programs have two or more paths of execution. Single threaded programs can perform only one task at a time, and have to finish each task in sequence before they can start another. Multiple threads can be executed in parallel in Symmetric-Multiprocessing machines. Applications that require parallel data access or processing are all potential for threaded applications. Threads can add substantial performance improvements to these applications.

In SPDSSERV.PARM, MAXWHTHREADS defines the number of threads a single process can create when performing a task in parallel. Creating too many threads can often cause performance degradation due to “thread starvation”, and SAS Support recommended performing a scalability test to determine the optimal MAXWHTHREAD setting. See Etri Abdel (2009) for the SAS step to create data table and %macro IOSCALE.

We performed the scalability test to determine the optimal MAXWHTHREDS value for our environment. We collected the run time for threads of 1,2,4,6,8,16. Test result showed that the curve became flat after 8. Thus we defined MAXWHTHREAD=8.

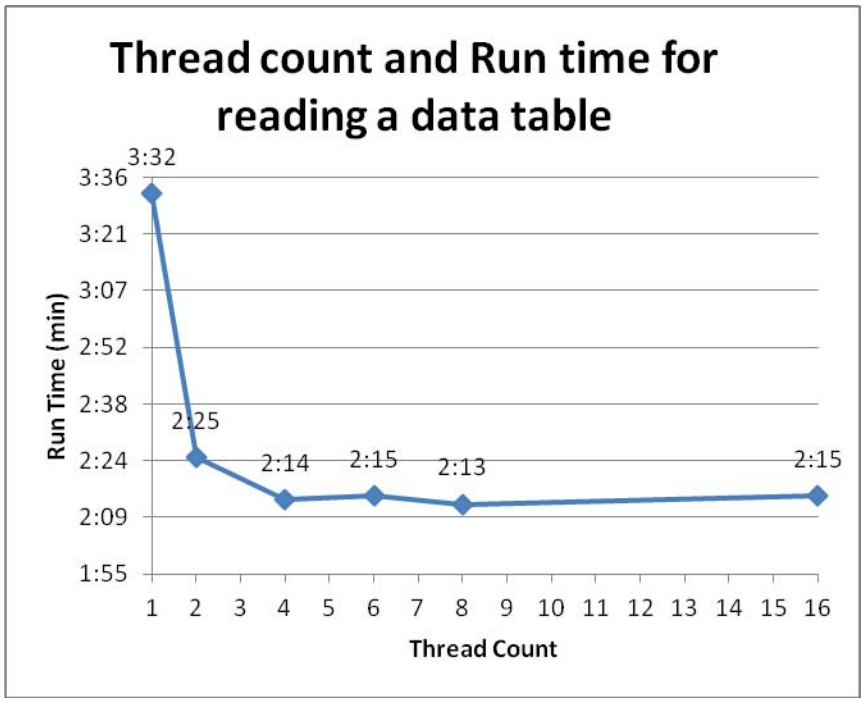


Figure 3. Thread count and run time for reading a data table

PARTITION SIZE

We adjusted MINPARTSIZE from 256MB to 1024MB to maximize throughput. This default part size applies to all tables stored in SPD Server. Part size determines the number of DPF files that are created when a table is written to the file system. For example, a table of 280GB will be distributed into 273 DPFs based on a MINPARTSIZE of1024MB. With our 16 file systems, there are 17 DPF files per file system. When CPUs process these DPF files

across all file systems simultaneously, the processing time is significantly reduced.

To reduce the number of DPF files you change the value of PARTSIZE. In our tests, we calculated and reset PARTSIZE for a number of big tables. If PARTSIZE is set with a higher MB value for big tables fewer DPF files will be created. Too few DPF files, however, may reduce parallel accessing and impact performance if there are not enough to be distributed across all file systems. On the other hand, too many DPF files may cause overhead. When too may open files are requested it can cause I/O contention.

Sample code for redefining partsize:

```
data client.Online_RxClaims (partsize=5820);
   set _temp.Online_RxClaims;
run;
```

BENCH MARK AND PERFORMANCE

We had two major purposes in building the new SAS SPD Server Data Warehouse. The first was to expand space to handle growing data volumes; the second was to improve SPD Server performance to meet users' expectations. Through our efforts mentioned in this paper we achieved that goal.

To validate that the new configuration was successful, we documented existing performance issues in the old configuration, specifically focusing on long running jobs, cases when Enterprise Guide projects would hang, significant difference between CPU time and real run time, and possible I/O contention. In addition to the documentation, we conducted performance tests in the old configuration to get benchmarks.

Performance Tests

We ran a test of 10 jobs all running at once and a second test of 20 jobs all running at once. Each of the jobs joined two SPD Server tables and included "case when" and "order by". A return of 3.38 million records was expected.

In the 10 concurrent job test, the run time was consistently 9-12 minutes in the old configuration and 7-9 minutes in the new configuration. In the 20 concurrent job test, the jobs hung after 10 minutes in the old configuration whereas the run time was 19-21 minutes in the new configuration. We submitted 30 jobs concurrently in the new configuration and all of them completed within 30-40 minutes.

The new configuration has been running for almost two years and there has not yet been a performance related issue.

SPD SERVER INDEX ANALYSIS

Our SAS Data Warehouse has a Star Schema design with a large fact table. In order to determine the fields for indexing we turned on SPD Server Audit logging for a month to document frequently used fields in "query where clause" accessing this table. Based on this index analysis, we identified the 14 most frequently used fields. We further checked the distribution of the proposed index fields for discriminate distribution. Fields with frequencies that fell in the 1-15% range were defined as good index variables. Ultimately we settled on 12 of the 14 fields as candidates for indexes. After creating indexes on these 12 fields we saw significantly improved query performance.

CONCLUSION

There are a number of things to look at when tuning SPD Server for performance. How file systems are laid out across disk, correct SPD Server parameter settings, and proper field indexing can significantly affect performance. The techniques covered in this paper will help you understand the approach we took in finding the right balance in all of these key areas and the type of performance improvements we saw after implementation.

REFERENCES

Crevar, Margaret A., Sober, Steven. 2009. *Best Practices for configuring your IO Subsystem for SAS Scalable Performance Data Server Tables*. support.sas.com/resources/papers/

Etri, Abdel. 2009. *The SAS Scalable Performance Data Server –Controlling the Beast!* SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc.

Sargent, Daniel. 2008. *Managing Large Data with SAS SPD Server®.* SAS Global Forum 2008 Conference. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Thanks to Paul Oberle and Jeff Miller from Prime Therapeutics LLC for their contribution of knowledge, skills, and support.

A special thanks to Abdel Etri of the SAS Institute for his expertise and knowledge. His valuable suggestions helped make our new SPD Server configuration a success.

CONTACT INFORMATION

Stephen Bogacz
sbogacz@primetherapeutics.com

Danni Luo
dluo@primetherapeutics.com

Greg Dorfner
gdorfner@primetherapeutics.com

David Bianchi
dbianchi@primetherapeutics.com