Technical Report

Performance Report of SharePoint 2007 EBS Provider for NetApp Storage System

Alex Jauch, Reena Gupta, NetApp
Nidhi Kansal, Microsoft
October 2010 | TR-3874

SCALABILITY WITH SHAREPOINT 2007 SP1 EBS

SharePoint® 2007 SP1 introduces a unique feature called External BLOB Storage (EBS), which provides the ability to store the large binary objects on external SMB shares instead of the normal BLOB store inside of SQL Server®. In support of this feature, NetApp conducted a joint testing engagement with Microsoft at the Microsoft Technology Center (MTC). The objective of this joint testing was to evaluate the performance impact of EBS as compared to native SQL Database storage.

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

Test results reveal that, in almost all cases, EBS outperforms native SQL Server storage for serving data stored in SharePoint. Key benefits of using NetApp's EBS provider for SharePoint 2007 compared to Native SQL Server storage are:

- It relieves pressure on SQL Server significantly in terms of CPU usage, SQL Server lock wait times, and disk I/O and thus improving scalability and reliability of SQL Server.
- EBS increases the scalability of SharePoint deployments as measured by the number of requests it can process per second (rps), which means the same SharePoint infrastructure will be able to support more users.
- EBS significantly improves the write latency (test time) and some read latency in a mixed workload, improving the end user perceived latency (performance).
- EBS in SnapManager 5.0 for MOSS has proven to be a very cost effective solution with the majority of data being stored on the cheaper SATA disks and with a small (approximately 10–16%) amount of metadata stored on relatively more expensive FC or SAS disks.

Note: EBS has not shown improvements in latency for read only operations, which has to be investigated further. The next version of SMMOSS would be improved to show improvement for read only performance.

2 BACKGROUND

Microsoft® Office SharePoint Server (MOSS) 2007 is the third major generation of SharePoint. Like previous versions, MOSS 2007 stores all the user-entered content in a site collection in a SQL Server database table as binary large objects (BLOBs). This architecture works well for traditional Web environments where the content is limited. Collaboration scenarios where users could store hundreds of thousands of large documents in one document library without any planning could potentially have scalability issues.

With SharePoint 2007 SP1, a new capability called External Blob Storage (EBS) was introduced to allow customers to externalize BLOB storage. NetApp supports EBS with SnapManager® MOSS 5.0 (SMMOSS). The detailed architecture of SnapManager MOSS 5.0 (SMMOSS) is described in the next section. This development raises the question of performance impact of external blog storage for NetApp customers who are considering deploying EBS on their existing and planned MOSS 2007 implementations. In order to answer these questions, NetApp conducted a proof of concept (POC) at the Microsoft Technology Center (MTC) in Mountain View, California. With the assistance of the MTC, NetApp's Microsoft Product Integration Team (MPIT) designed and implemented an EBS testing platform. This white paper relates our POC findings at the MTC.

3 SNAPMANAGER 5.0 FOR MOSS (SMMOSS) ARCHITECTURE

SnapManager for Microsoft Office SharePoint Server is an enterprise content management, backup, and recovery solution for Microsoft SharePoint versions, including Windows® SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007. In SMMOSS 5.0, there are three main areas of functions:

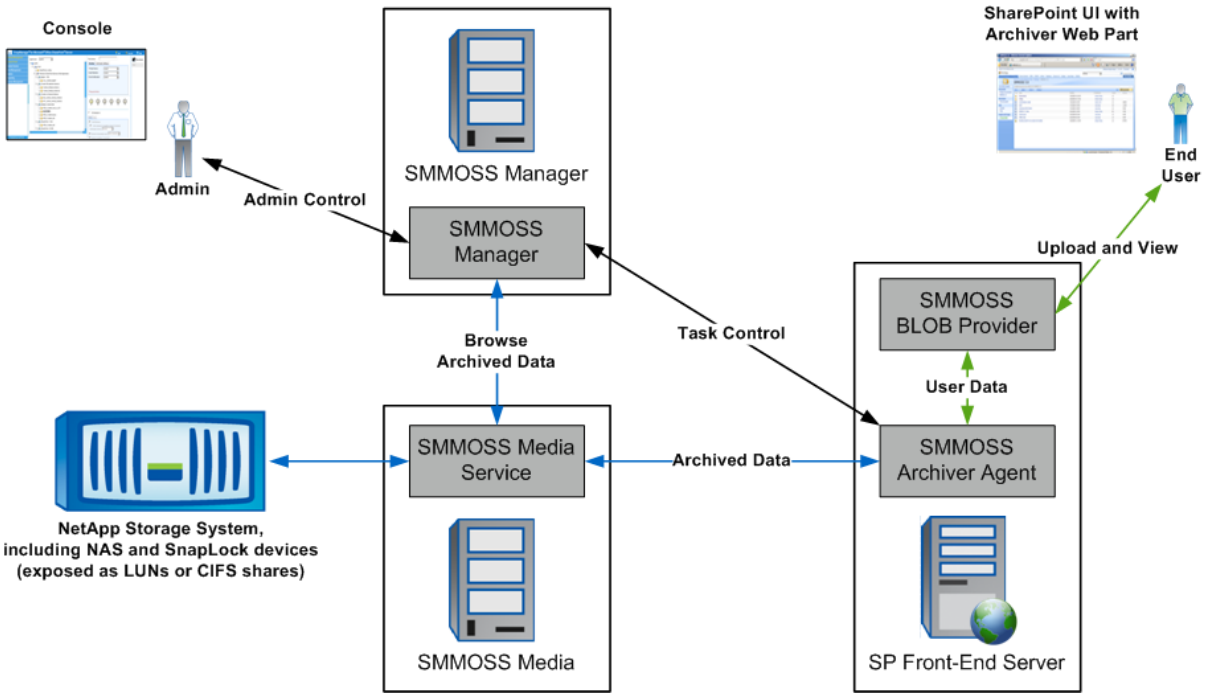
STORAGE OPTIMIZATION

Many documents (or attachments) in SharePoint are large files, which take large amount of valuable SQL Server storage, especially when SharePoint has been in use for a long period of time with old and less important large files.

SMMOSS offers two approaches to address this issue:

- **Archiver.** Business rules can be set and content meeting these rules can be moved out from SharePoint SQL Server storage that is expensive and high performing out into SATA-based cheaper storage. From user's perspective, they can still access SharePoint the same way as before. SnapLock® integration is also available to help keeping the archived content in SnapLock volumes to meet compliance regulations.
- **Extender.** SMMOSS can also be used as primary SharePoint storage where user content is uploaded into SATA-based cheaper storage directly without going through SQL Server first. This will help customers where their average content size is big.

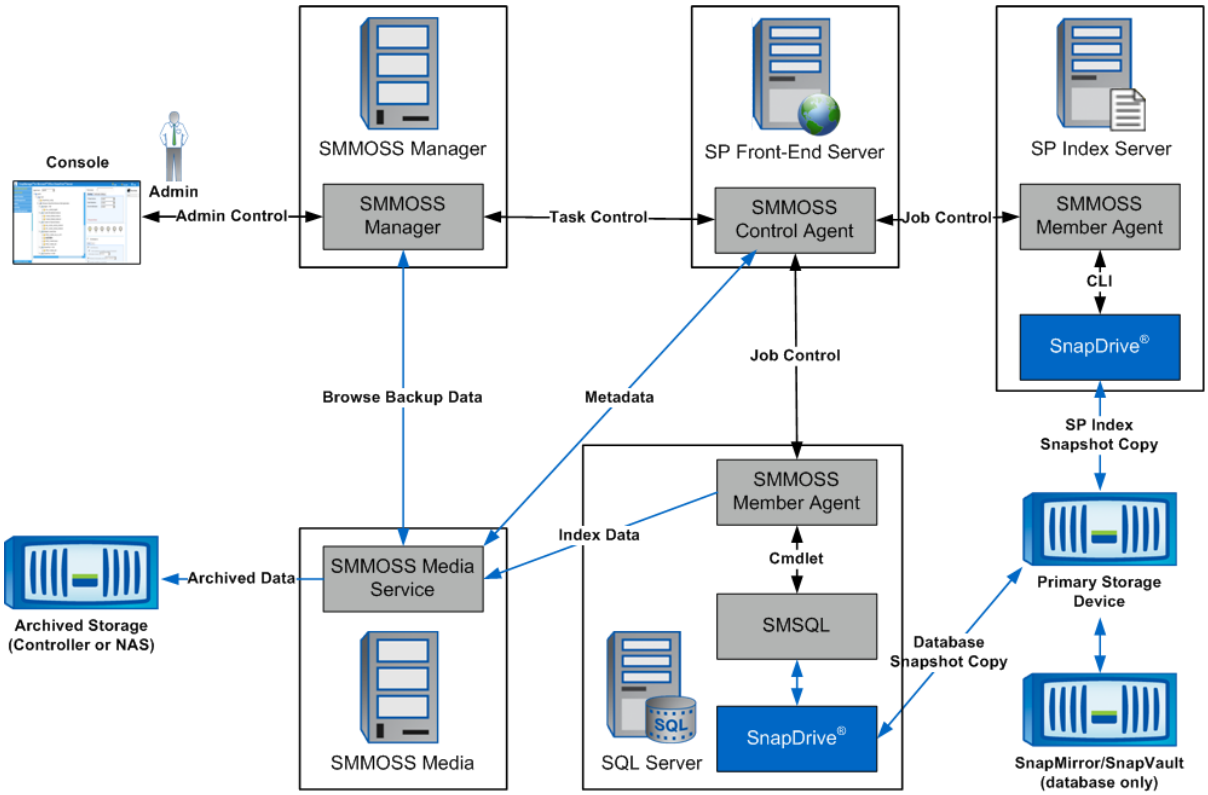
Figure 1) Storage optimization module architecture.



DATA PROTECTION

SharePoint is a complex system. A typical SharePoint deployment consists of multiple servers and very flexible content and design structure. The data protection function of SMMOSS is targeting on fast backup of SharePoint data with flexible restore granularity, all under an integrated interface. Typical use cases include fast restore of documents with full fidelity metadata, security, and version history and fast restore of content database, Web application, or even whole farm. SnapMirror® and SnapVault® integration is also available for integrated replication and long-term backup protection.

Figure 2) Data protection module architecture.



MIGRATION

As SharePoint gains popularity, more and more customers are switching to SharePoint to host and manage their data. SMMOSS 5.0 supports two important migration paths: from SMB file system and Microsoft Exchange public folders while preserving the metadata and security using defined mapping configurations.

3.1 SMMOSS 5.0 COMPONENTS DESCRIPTIONS

Following is a description of the components mentioned in above diagrams and how they interact. All these are logical components, and they could all be located on the same physical server or distributed on multiple servers.

SMMOSS MANAGER

SMMOSS Manager is the main control component. It has two internal subcomponents:

- Control Service: manages configurations such as users, security, plans, schedule, job control, and other activities in SMMOSS
- Web Service: provides an interface to Web browser-based SMMOSS Console. It gets data from Control Service.

SMMOSS Manager communicates with other SMMOSS components (including agents or Media Server) using TCP socket. SMMOSS Console connects to SMMOSS Manager using HTTP/HTTPS. Control Service uses default port of 12000; Web Service uses default port of 8080 (HTTP) or 8443 (HTTPS). Both are public ports.

SMMOSS Manager maintains all the configurations within SMMOSS, examples are users/groups defined in SMMOSS, plans and policies defined for data protection, storage optimization and migration, schedules, device settings, SMMOSS topology, and so on. These settings are stored in a Derby internal database in SMMOSS Manager Installation folder. SMMOSS System Backup can be used to back them up periodically.

SMMOSS MEDIA SERVICE

SMMOSS Media Service is responsible for managing data within SMMOSS, including job metadata with indexing of backup data, archived data, as well as data retention. Data is transferred to media server directly from related agents, and SMMOSS Media Service interacts with underlying storage to store data, using NTFS/CIFS interface. For archiving data, NetApp® storage system is required (either storage appliance or NAS devices). It can also interact with SnapLock CIFS share for compliance retention.

SMMOSS Media Service communicates with SMMOSS Manager and Agents using TCP socket. The default port is 12001, a public port.

SMMOSS CONTROL AGENT

SMMOSS Control Agent is part of SMMOSS Data Protection module. It's responsible for coordinating backup and restore jobs. It receives task control from SMMOSS Manager, analyzes SharePoint farm structure, and sends the job control to related SMMOSS Member Agents in the farm to complete the operations. SMMOSS control agent is installed on SharePoint front-end server.

SMMOSS Control Agent communicates with SMMOSS Manager and other Agents using TCP/IP. It uses default port of 10103, a public port.

SMMOSS MEMBER AGENT

SMMOSS Member Agent is part of the SMMOSS Data Protection module. Its main purpose is to interact with SharePoint related components to perform backup/restore operations. There are three types of member agents:

- SQL Server Member Agent: installed on SQL Server. It uses SMSQL cmdlet to back up and restore databases. It also generates backup index from database Snapshot™ copy and sends to SMMOSS Media Service.
- Index Member Agent: installed on SharePoint Index Server. It uses SnapDrive® CLI to create Snapshot backups of index LUNs.
- Front-End Member Agent: installed on SharePoint front-end server. It can back up IIS settings, SharePoint 12 hive, and resources in file system. Backup data is transferred to SMMOSS Media Service.

SMMOSS Member Agent communicates with Control Agent and Media Service using TCP socket. It uses default port of 10103, a public port.

SMMOSS ARCHIVER AGENT

SMMOSS Archiver Agent is the main component for SMMOSS Storage Optimization module. It should be installed on ALL front-end servers in SharePoint farm. The SharePoint version should be WSS 3.0/MOSS 2007 SP1 or above. It has the following main functions:

- Execute predefined business rules to archive existing SharePoint content to SMMOSS Media Service, and optionally change them into stubs (Archiver only)
- Retrieve data for user stub access
- Process end-user archiving requests
- Stores end-user upload data to SMMOSS Media Service (Extender only)
- Restore back original content to SharePoint at user's request

SMMOSS Archiver Agent communicates with SMMOSS Manager and Media Service using TCP socket. It uses default port of 10107, a local port only used by other components on the same server.

Followings are some subcomponents related to storage optimization.

SMMOSS BLOB PROVIDER

In WSS 3 and MOSS 2007 SP1, External BLOB Storage (EBS) interface was added to allow offloading SharePoint content from SQL Server storage. SMMOSS Storage Optimization module utilizes this interface to achieve its function. SMMOSS BLOB Provider interacts with SharePoint for stub-related operations:

- When user accesses a stub, SharePoint will ask BLOB Provider for the data stream. BLOB Provider will get data from Media Service using SMMOSS Archiver Agent.
- (Extender Only) When user uploads a file, SharePoint will pass the data stream to BLOB Provider, which in turn sends data to SMMOSS Media Service using Archiver Agent.

SMMOSS BLOB Provider is loaded by SharePoint IIS process at runtime. It communicates with SMMOSS Archiver Agent using local TCP socket.

The reason BLOB Provider does not directly interact with Media Service is that Archiver Agent can centralize the queuing and caching to help performance. This also helps to keep BLOB Provider simple and avoid affecting SharePoint performance. Data stream transfer is asynchronous between each hop.

END-USER ARCHIVING WEB PART

A SharePoint Web part is provided to give user ability to selectively archive SharePoint content. The Web Part will send user archiving request to SMMOSS Archiver Agent, which in turn queues the request, archives the content, and changes it into stubs. The Web Part communicates with Archiver Agent over local TCP socket.

SMMOSS MIGRATION AGENTS

SMMOSS FILE SYSTEM MIGRATION AGENT

SMMOSS File System Migration Agent has two parts: one is installed on the source file server; one is installed on SharePoint front-end server. Data is transferred from source migration agent to destination migration agent to store in SharePoint. These agents use TCP socket to communicate. It uses default port of 10103, a public port.

END-USER FILE SYSTEM MIGRATION WEB PART

A SharePoint Web part is provided to give user ability to selectively migrate file system content to SharePoint. To give end user rich upload experience, ActiveX control is used in the Web part. It uses TCP socket to communicate with destination migration agent.

SMMOSS PUBLIC FOLDER MIGRATION AGENT

SMMOSS Public Folder Migration Agent is installed on the destination SharePoint front-end server. It uses WebDAV to get data from Exchange server, and stores data into SharePoint.

4 TESTING METHODOLOGY

4.1 TEST CASES

In order to replicate realistic load onto MOSS, we used Microsoft's Planning and Architecture for Office SharePoint Server 2007 white paper as a baseline. In that document, Microsoft gives the following sample usage profile:

Table 1) Sample usage profile from Microsoft.

Operation	Percentage of Throughput
Get home page	15.00
Get cached document	15.00
Get static document	15.00
Get list page (HTML)	10.00
Get list page (grid)	10.00
Get list form	7.00
404 errors	5.00
Insert list item	2.00
Edit list item	2.00
Delete list item	2.00
Insert document	2.00
Synchronize with Outlook	2.00
Delete document	2.00
List URLs	2.00
DAV open document for edit	1.00
DAV save document	1.00
FPRPC open document for edit	1.00
FPRPC Save document	1.00
Short-term check-out	1.00
Incoming e-mail	1.00
RSS (Really Simple Syndication)	1.00
Start workflow	0.75
Workflow task completion	0.75
Add/remove user	0.50

In looking at this profile, we determined that we could easily simulate about 87% of the workload, including all of the workload associated with documents. Therefore, we decided to automate the following use cases:

Table 2) Automated use cases in the tests.

Use Case	Description
DeleteListItem	Delete a random item from a random list.
ListEdit	Edit a random item from a random list.
ListReadCoded	Read a random item from a random list.
LoadHomePageCoded	Open the home page of the site.
NewListItem	Create a new list item in a random list.
Read100k	Read a random 100K file.
Read100Meg	Read a random 100MB file.
Read10Meg	Read a random 10MB file.
Read1Gig	Read a random 1GB file.
Read1Meg	Read a random 1MB file.
Upload100K	Upload a new 100K file to a random library.
Upload100Meg	Upload a new 100MB file to a random library.
Upload10Meg	Upload a new 10MB file to a random library.
Upload1Meg	Upload a new 1MB file to a random library.

This does not represent an exhaustive list of SharePoint features, which was not our intent. The goal here was to create a representative sample set. We feel we have achieved this goal.

After we had our use cases developed, we needed to develop a “base case” and determine the correct weighting of each of these individual tests. In order to do this, we used the weightings above and then broke down the file uploads and downloads assuming a larger number of small files and a smaller number for larger files to come up with an optimal ratio of 75% reads and 25% writes.

Table 3) Use case percentages in the tests.

Use Case	% of Cases
DeleteListItem	2.6%
ListEdit	2.6%
ListReadCoded	21.8%
LoadHomePageCoded	19.2%
NewListItem	2.6%
Read100k	23.6%
Read100Meg	2.0%

Use Case	% of Cases
Read10Meg	2.0%
Read1Gig	2.0%
Read1Meg	8.8%
Upload100K	5.8%
Upload100Meg	2.0%
Upload10Meg	2.0%
Upload1Meg	3.0%

Note that in order to make sure of a minimum sample size, we set a minimum floor of 2%. A lower distribution would not allow enough samples for a reliable result. We also ran a number of variations of this breakdown and our overall results, and conclusions are consistent with all the variations we tested. Naturally, your results will vary. We strongly encourage customers to repeat this test using their own use cases and data.

Details on the test are in Appendix A, and we will provide the sample code upon request.

4.2 SHAREPOINT CORPUS

The next step was to develop a corpus of documents to run the test against. We decided to build a variety of document sizes with the following breakdown:

Table 4) Test documents in the Corpus.

Num Sites	File Size (MB)	Libraries	Files/Lib	Total Size	Numfiles	Files Per Site
5	0.1	400	1,000	200,000	2,000,000	400,000
5	1	40	1,000	200,000	200,000	40,000
5	10	40	100	200,000	20,000	4,000
5	100	40	10	200,000	2,000	400
5	1000	4	10	200,000	200	40

Per the farm configuration detailed starting section 6, we created 2 separate Web applications: “SQLNative” Web application configured for native SQL Server storage and “EBS” Web application configured for EBS . To make sure the test was equal, the EBS and SQLNative sites were created with the exact same tool using the same scripts.

For each of the Web applications, we had a corpus size of about 1TB as listed above and around 2 million files for each of our environments (EBS and SQLNative). This is a large enough dataset to make sure that we are simulating a real-world environment but is not beyond the scope of a reasonable enterprise class MOSS implementation. In addition, we then created two groups of 1000 lists with 1000 items per list to simulate a real-world scenario. These lists were created to support the list use cases and were spread evenly across the five sites for each environment.

There were several parameters that helped us decide our base load/testing methodology:

- The page load time (PLT) beyond 5 seconds is not acceptable for most customers.
- As WFEs are added to a SharePoint farm, there will be an increase in rps, up to a point. Our goal was to figure out the sweet spot for where rps was the max before it starts to fall.
- With Native SQL Server configuration, our tests revealed that 300 concurrent threads was a good starting point to meet the above mentioned criteria, and that's what we chose as our starting point.
- Post that, we used constant load tests and kept increasing the thread count by 100.
- At about 500, we started to see a majority of the tests fail due to time out errors and the PLT was way beyond 5 seconds, so 500 thread counts was the maximum we tested with.

In the results sections below, "threads" refers to the number of concurrent Visual Studio Test threads running. The number of threads running is not meant to imply a number of concurrent users. In all cases, tests were conducted against identically prepared libraries for native SQL Server storage and EBS storage. Since the libraries were prepared using identical load scripts, we feel that the comparison results are valid comparisons of EBS vs. Native SQL Server BLOB storage performance given the workload that we have chosen for this test.

4.3 ASSUMPTIONS

There were certain assumptions made during this testing as listed below:

- The scenario simulated during testing assumes a larger number of small files and a smaller number for larger files to come up with an optimal ratio of 75% reads and 25% writes.
- We used the OOB collaboration template for team sites hosting the SharePoint content.
- We did not vary the test cases to run under different user/different role. If there are many different SharePoint roles (SharePoint groups) in your organization with fine grained permissions, you should consider testing with different user profiles. We used images in documents to make it a 1MB doc. The documents were generated using scripts, and similar-sized documents had the same content.
- We did not test with document libraries configured for rights management.
- Initially we had setup the Network Load Balancing (NLB) with the Web Front End servers, but we had to switch it to the DNS round robin, because NLB won't allow any files larger than 100MB to be uploaded into SharePoint.
- The Index Services were started in the SharePoint Web farm, but no indexing/searching tests were included in our test methodology, mostly based on assumption that index crawling would be generally scheduled in off hours or weekends in a production environment.

5 KEY FINDINGS

5.1 EBS REDUCES READ WRITE TEST TIME FOR A MIXED WORKLOAD

For the baseline mixed workload test cases with the ratio of 75% reads and 25% writes, we noticed significant reduction in the duration of the write test for all file sizes as demonstrated in the graphs below. The read time was much better for the smaller size files, but for the larger files sizes of 100MB or above, the read time became higher as compared to native SQL Server. At this time, we noticed SQL Server lock Wait time counter values were really high, which leads us to believe that the reason behind this is SQL Server locking. SQL Server locks are caused when SQL Server locks a table page during a write operation and a succeeding operation has to wait until the lock is removed.

Figure 3 and Figure 4 show the write and read test time behavior for EBS as compared to native SQL Server for a 500 thread operation. The test duration in the chart indicates the total test time for reading/writing of all the files in the corpus with a specific file size; not to get confused, it's not for a single file test.

Figure 3) Write test duration with mixed workload.

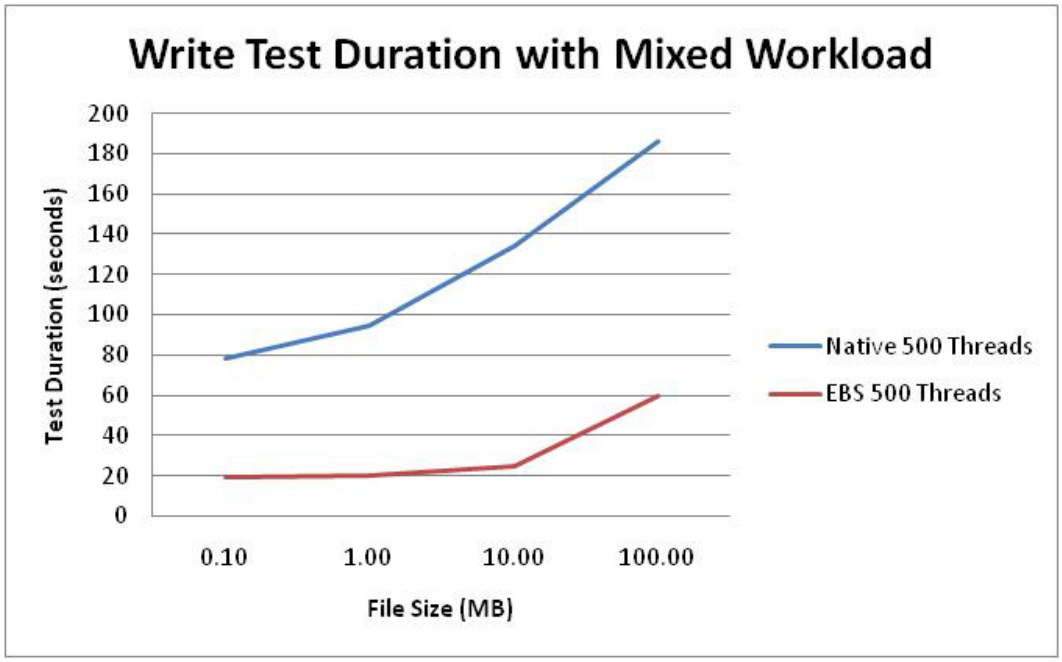
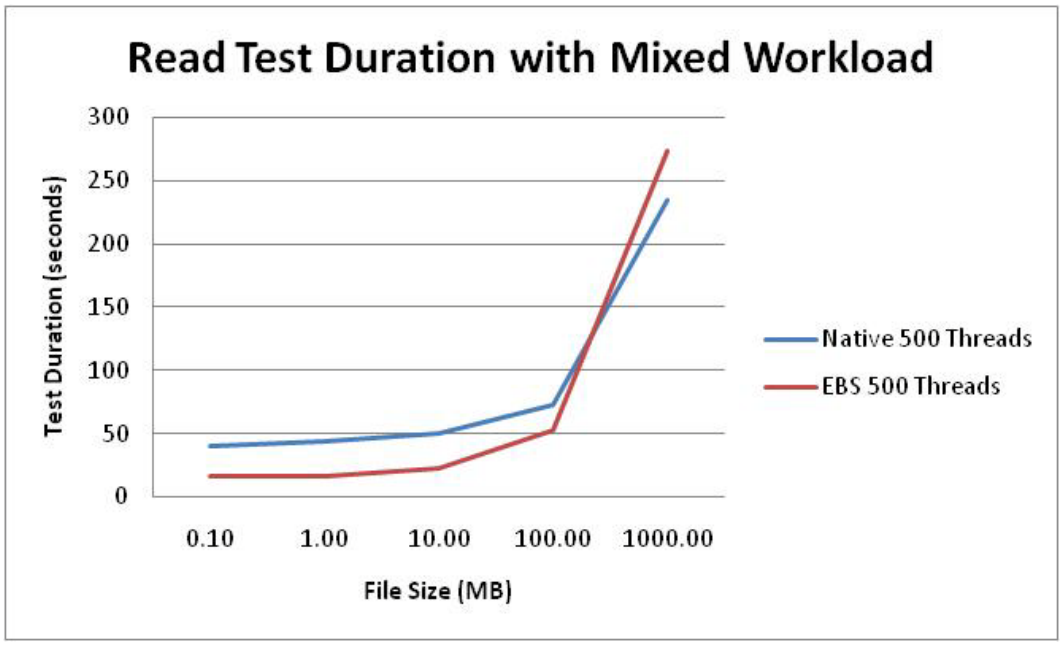


Figure 4) Read test duration with mixed workload.

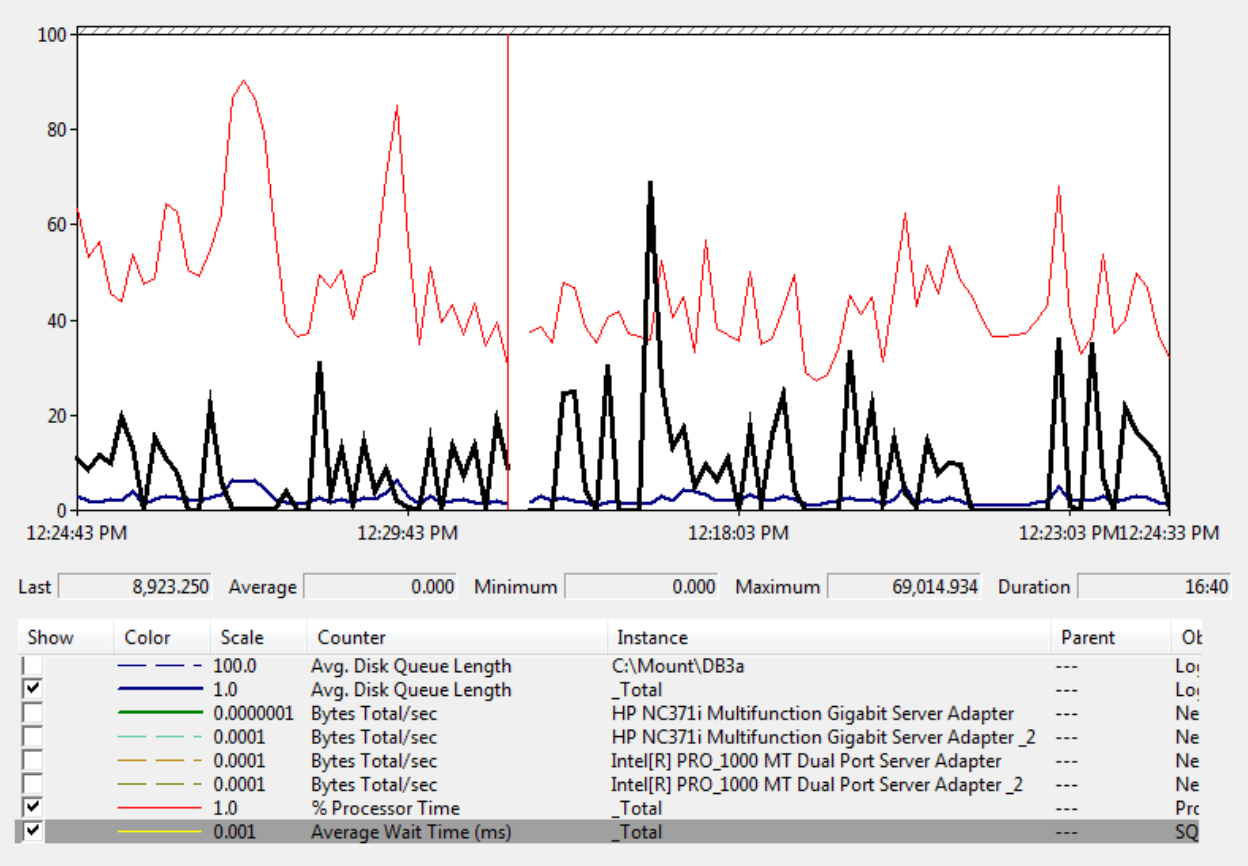


5.2 EBS RELIEVES SQL SERVER LOCK PRESSURE

Since MOSS writes all documents for a given content database to a single table, we see significant SQL Server Wait time due to locking in workloads that include concurrent document uploads. Our base case assumes a 75%/25% read/write ratio. Customers with lower write ratios will see less locking than customers with higher write profiles.

In the screenshot below, we see a sample Perfmon chart showing Average Wait time for SQL Server when supporting our baseline use case.

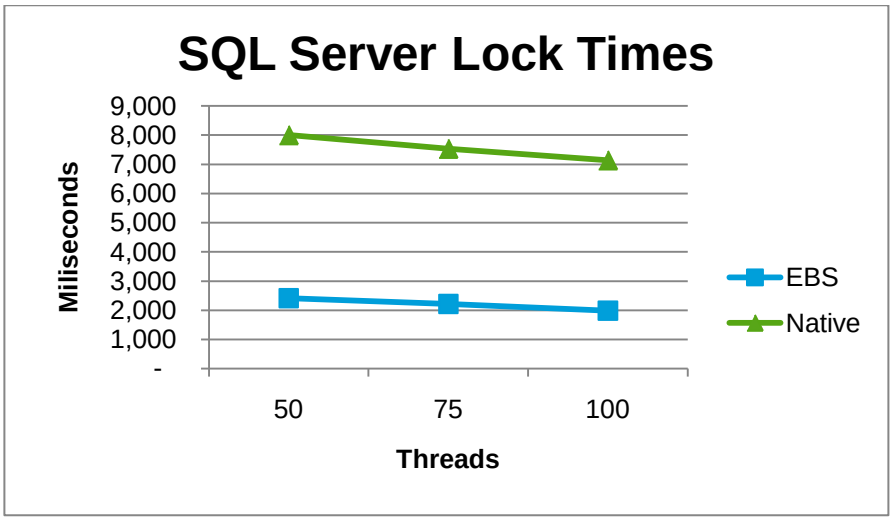
Figure 5) SQL Server locks average wait time with native SQL Server.



As you can see from the screenshot, SQL Server Locks Average Wait Time can be quite high, in this case spiking above 1 minute. This is significant wait time for collaboration workloads and SQL Server workload in general. We saw this pattern repeatedly in use cases that included significant levels of write activity (above 5%). In many cases, customers who are seeing significant SQL Server lock waits should see significant performance improvement after deploying EBS.

As you can see in the chart below, we saw significant reduction in SQL Server wait times across various load profiles on an EBS enabled document library as compared to a native SQL Server BLOB.

Figure 6) SQL Server lock times.

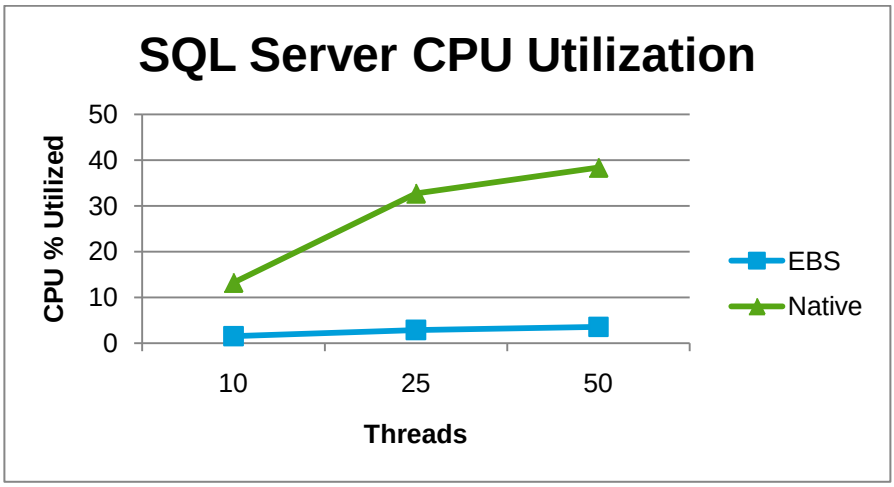


The numbers on the X-axis refer to the number of concurrent testing threads, and the numbers on the Y access refer to SQL Server wait times in milliseconds.

5.3 EBS REDUCES SQL SERVER CPU USAGE

As EBS removes some I/O operations from SQL Server, the overall CPU load that SQL Server generates with BLOBs going directly to EBS is significantly less than the same workload running against a native SQL Server BLOB store.

Figure 7) SQL Server CPU utilization.

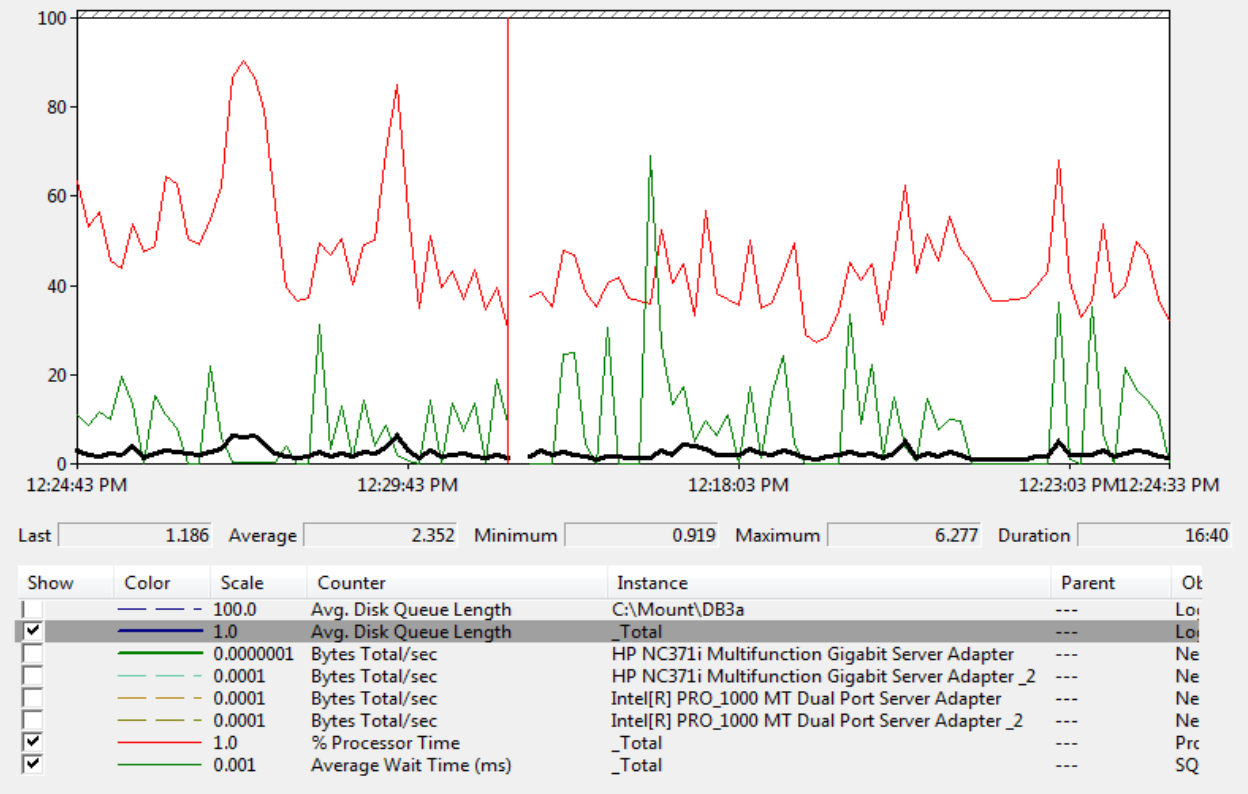


The numbers on the X-axis refer to the number of concurrent testing threads, and the numbers on the Y access refer to % CPU utilized on SQL Server. As you can see in the chart above, SQL Server CPU utilization was much higher for Native than for EBS even under relatively light loads. Again, this trend was seen across all workloads.

5.4 EBS REDUCES DISK I/O ON THE SQL SERVER

This is fairly obvious. Since EBS redirects BLOB read/write activity from SQL Server to the SMB share on the NetApp storage, there is a significant reduction in SQL Server disk I/O. SQL Servers that are disk bound will see a significant performance improvement after EBS is enabled.

Figure 8) SQL Server average disk queue length with native SQL Server.



The screenshot above shows a sample perfmon trace for a SQL Server that is experiencing disk performance pressure.

Notice that disk Queue Length is averaging 2.352 and shows a maximum of 6.277. While Disk Queue spikes are normal as SQL Server flushes cache or commits transactions from the transaction log, sustained disk queues or high disk latency as shown by perfmon indicate that SQL Server is under disk pressure.

In addition, the removal of SQL Server BLOBs makes the overall SQL Server database much smaller, as you can imagine. In our tests, the EBS databases were about 16% of the size of Native BLOB databases with the same configuration and load. As the database size becomes smaller, the time to back up or restore the SQL Server database goes down proportionally and reduces the overall TCO of the solution.

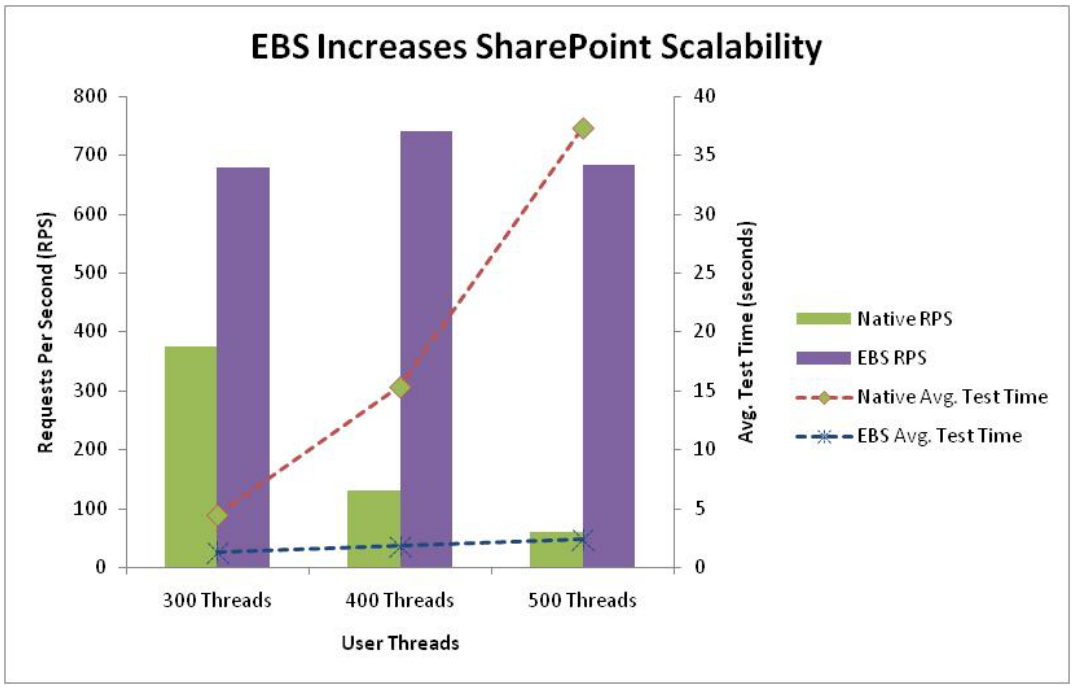
5.5 EBS INCREASES SHAREPOINT SCALABILITY

At very high levels of throughput, EBS enabled sites scale to much higher Requests Per Second (RPS) than native SQL Server BLOB storage.

As evident from Figure 9, the EBS-enabled site was able to serve more than twice the number of RPSs compared to native BLOB storage in our baseline test case. The more concurrent load that was added, the more pronounced this became. In addition, EBS sites were able to perform the tests with less latency. The chart shows that as we increased load on the native SQL Server sites, the native SQL Server

throughput kept going down. In contrast, EBS sites sustained much higher throughput; almost 1000% higher at 500 user threads with 94% lower average test time.

Figure 9) EBS results in higher RPS and lower average test time.



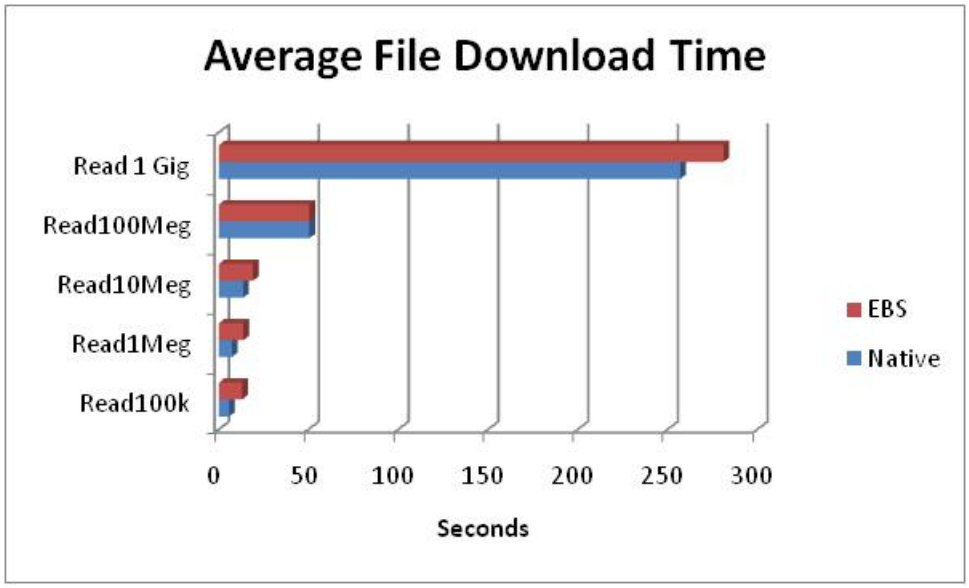
Again, this is due to the fact that there are high levels of SQL Server locks when uploading documents, resulting in the large number of writes to the same table. Workloads with lower write percentages will see lower levels of improvement, while use cases with higher levels of write activity will see even larger gains.

Translating RPS into concurrent users is difficult. Microsoft provides official guidance here. To summarize, Microsoft projects that 1 RPS = 1000 concurrent users. However, based on field experience with SharePoint implementations, we feel that this guidance is optimistic. We feel that this number should be much lower, on the order of 100 concurrent users per RPS. Bottom line is that the actual number of Requests per Second generated by users will vary depending on workload, use pattern, and the applications being used. In any case, the more RPS a given configuration can support, the more cost effective the solution is.

5.6 EBS INCURS SIGNIFICANT LATENCY ON READ OPERATIONS

When issues such as write locking in SQL Server are factored out, we saw significant latency incurred in read intensive workloads. For example, when testing a 100% read-only workload, we saw read latency as much as 2x for EBS compared to what we saw with native SQL Server BLOB performance. While some increased latency is expected, we also saw increasing latency penalties with larger files. In theory, larger BLOBs should perform better on EBS than on native SQL Server, but this is not the result we saw in our testing. We have requested that the product group look into this abnormality.

Figure 10) EBS impact on read-only operations.



In Figure 10, we see the results from a 100% read only workload. In mixed workloads, overall performance for EBS was better, but poor performance on read workloads needs to be improved.

6 DETAILED TEST RESULTS

During the course of the MTC engagement, we ran over 100 tests using various scenarios and test cases. The following section documents the key scenarios under test in detail.

6.1 BASELINE

The Baseline test uses the test case mix suggested by Microsoft and attempts to recreate a "realistic" SharePoint environment. Our expectation is that this test case is closest to actual customer experience.

Table 5) Request per second comparison for baseline use cases.

Web Front End (RPS)	Native 300 Threads	Native 400 Threads (4 WFEs)	Native 400 Threads (5 WFEs)	Native 500 Threads	Native 500 Threads	EBS 300 Threads (SMB1)	EBS 400 Threads (SMB1)	EBS 400 Threads (SMB2)	EBS 500 Threads (SMB2)	EBS 500 Threads (SMB2)
WFE 1 RPS	111.4	46.1	49	21.9	21.9	198.3	216.3	132.5	175	112.962
WFE 2 RPS	77.3	27.6	46.6	16.23	128.865	170	169.6	188.6	172	146.585
WFE 3 RPS	105	22.4	73.7	10.8	13	148	178.2	187.5	169	157.524
WFE 4 RPS	81	34.1	53.232	10.1	11	162	177.4	191.6	168	168.022
WFE 5 RPS			37.4		9.2			158		97.736
Total RPS	**374.7**	**130.2**	**259.932**	**59.03**	**183.965**	**678.3**	**741.5**	**858.2**	**684**	**682.829**
Concurrent Users	**22,482**	**7,812**	**15,596**	**3,542**	**11,038**	**40,698**	**44,490**	**51,492**	**41,040**	**40,970**
Avg. Test Time	**4.42**	**15.3**	**7.94**	**37.3**	**28.3**	**1.23**	**1.74**	**1.54**	**2.36**	**2.38**

Table 6) Test duration time comparison for baseline use cases.

Average Use Case Duration	Native 300 Threads	Native 400 Threads	Native 400 Threads	Native 500 Threads	Native 500 Threads	EBS 300 Threads (SMB1)	EBS 400 Threads (SMB1)	EBS 400 Threads (SMB2)	EBS 500 Threads (SMB2)	EBS 500 Threads (SMB2)
DeleteListItem	16.3	51.4	35.1	92.8	92.2	2.05	2.44	2.99	3.08	3.77
ListEdit	12.9	41.6	23.1	72.4	61.6	1.37	1.57	2.03	2.03	2.49
ListReadCoded	1.55	6.96	3.99	22.5	16.3	0.95	1.36	1.18	1.7	1.61
LoadHomePageCoded	5.77	15.9	6.81	30.4	29.4	0.42	0.6	0.55	0.76	0.83
NewListItem	10.6	35.1	22.5	73.7	64	1.29	1.56	1.85	1.97	2.38
Read100k	6.71	18.8	9.22	39.7	36.8	6.64	12.1	8.82	24.7	16
Read100Meg	30	45.5	46.4	72.2	71.8	41.7	48.1	50	62.9	69.9
Read10Meg	15	26.6	28.3	49.4	49.2	13.7	18.2	16.9	32	25.7
Read1Gig	169	202	206	235	231	237	283	257	275	271
Read1Meg	7.88	22.6	11.4	43.1	39.9	7.12	12.1	9.7	23.1	17.5
Upload100K	13.7	61.6	24.9	78.6	76.6	10.4	14.9	4.45	25.3	7.05
Upload100Meg	54.1	132	53	186	6.52	65.8	68.1	41.8	85.7	16
Upload10Meg	21.5	82.5	57.1	134	121	17.8	20.5	15.9	33.8	15.8
Upload1Meg	17.4	74.1	29.7	94.7	95.8	11.6	16.2	5.42	25.9	8.41
Average (All Tests)	**4.42**	**15.3**	**7.94**	**37.3**	**28.3**	**1.23**	**1.74**	**1.54**	**2.36**	**2.38**

As discussed previously, we see a significant impact on performance as concurrency rises. The more concurrent write operations that are performed, the better EBS does. This is consistent across all use cases and all scenarios.

Table 7) Baseline RPS comparison with lower thread counts.

Web Front End (RPS)	Native 50 Threads	Native 25 Threads	Native 10 Threads	EBS 50 Threads	EBS 25 Threads	EBS 10 Threads
WFE 1 RPS	0.785	2.013	0.692	1.307	2.112	0.192
WFE 2 RPS	2.53	3.013	2.169	2.878	2.796	1.132
WFE 3 RPS	2.866	2.459	1.232	2.669	2.761	1.094
WFE 4 RPS	2.262	2.526	1.587	3.297	3.557	1.508
WFE 5 RPS	4.824	6.872	0.859	7.233	5.357	1.827
Total RPS	**13.267**	**16.883**	**6.539**	**17.384**	**16.583**	**5.753**
Concurrent Users	**1,327**	**1,688**	**654**	**1,738**	**1,658**	**575**
Avg. Test Time	**15**	**6.43**	**5.95**	**8.5**	**5.23**	**4.65**

6.2 READ ONLY

This test case included only the read test cases.

Table 8) RPS and test duration time comparison for read-only test cases.

Web Front End (RPS)	Native 200	EBS 200	Native 400	EBS 400
WFE 1 RPS	71.008	80.351	19.32	99.593
WFE 2 RPS	83.096	158.339	39.422	132.244
WFE 3 RPS	116.873	160.35	36.363	131.647
WFE 4 RPS	126.522	148.872	42.672	140.659
WFE 5 RPS	138.209	134.548	52.589	112.499
Total RPS	**535.708**	**682.46**	**190.366**	**616.642**
Concurrent Users	**5,357**	**6,825**	**1,904**	**6,166**
Avg. Test Time	**1.14**	**0.48**	**6.55**	**1.02**
Average Use Case Duration	Native 200	EBS 200	Native 400	EBS 400
ListReadCoded	1.31	0.47	7.83	1.01
LoadHomePageCoded	0.77	0.17	4.95	0.35
Read100k	1.01	4.41	5.63	12.9
Read100Meg	23.4	57.1	50.3	50.3
Read10Meg	5.21	11	13.5	18.9
Read 1 Gig			257	281
Read1Meg	1.71	5.18	7.16	13.6

Again, with all write activity removed, the read latency shows up clearly. Note that non-EBS activities such as ListReadCoded are much faster in the EBS cases due to the reduced load placed on SQL Server.

6.3 WRITE ONLY

This case is the opposite of the one above. We removed all read cases and only ran the write cases. Again, this resulted in a significant performance advantage for EBS. Similarly, write latency is much reduced. In all cases, EBS was significantly faster to write the same file as native SQL Server BLOB storage.

Table 9) RPS and average test duration time comparison for write-only test cases.

Web Front End (RPS)	Native 100	EBS 100	Native 75	EBS 75	Native 50	EBS 50
WFE 1 RPS	0.404	2.3	1.025	4.289	0.55	2.229
WFE 2 RPS	2.457	8.003	0.953	6.266	1.951	4.551
WFE 3 RPS	2.104	5.122	1.159	4.63	1.683	4.032
WFE 4 RPS	2.137	12.067	1.036	4.33	0.958	5.985

Web Front End (RPS)	Native 100	EBS 100	Native 75	EBS 75	Native 50	EBS 50
WFE 5 RPS	3.067	15.593	2.292	14.611	2.213	14.384
Total RPS	**10.169**	**43.085**	**6.465**	**34.126**	**7.355**	**31.181**
Concurrent Users	**102**	**431**	**65**	**341**	**74**	**312**
Avg. Test Time	**19.5**	**3.66**	**22.8**	**3.42**	**13**	**3.01**
SQL Server: CPU	24.732	19.033	23.651	19.674	22.777	17.254
SQL Server: Disk Queue	4.573	2.732	3.414	3.072	3.831	2.778
SQL Server: Lock Wait Time (Avg)	7,138	1,990	10,826	2,217	7,999	2,417
SQL Server: Lock Wait Time (Max)	25,608	7,504	41,114	5,808	37,529	6,680
-						
Upload 100K	17.7	3.11	21.4	2.62	12	2.33
Upload 100MG	32.6	15.5	36	14.2	26.5	14
Upload 10MG	20	6.56	22.7	5	13	4.17
Upload 1MG	19.6	3.01	20	3.05	11.4	3

6.4 FLASH CACHE

In addition to our EBS testing, we also ran the same baseline case with a Flash Cache[1] card enabled on the EBS NAS storage side (FAS3170B controller). Because of the nature of large cache performance, we decided to test this scenario multiple times with long run times. The expectation is that performance should slowly improve as the cache is warmed, which did not happen. We did not see significant performance for a Flash Cache enabled storage controller as compared to a standard controller without Flash Cache.

Table 10) RPS performance and Flash Cache card enabled.

Web Front End (RPS)	EBS 300 Threads Run1	EBS 300 Threads Run 2	EBS 300 Threads Run 3	EBS 300 Threads Run 4	EBS 300 Threads Run 5
WFE 1 RPS	70.922	79.879	139.067	87.015	120.077
WFE 2 RPS	158.477	145.588	164.826	180.583	178.376
WFE 3 RPS	138.843	142.718	161.045	172.708	177.834
WFE 4 RPS	146.285	138.065	186.09	171.252	169.343
WFE 5 RPS	112.245	104.056	139.267	134.881	142.471
Total RPS	**626.772**	**610.306**	**790.295**	**746.439**	**788.101**
Concurrent Users	**37,606**	**36,618**	**47,418**	**44,786**	**47,286**
Avg. Test Time	**1.48**	**1.3**	**1.37**	**1.52**	**1.31**

[1] Previously “PAM II.”

We did not see any noticeable improvement in the latency (test case duration) either. See Table 11 for the latency (test case duration) observed.

Table 11) Average test duration time observed with Flash Cache card enabled.

Average Use Case Duration	EBS 300 Threads Run1	EBS 300 Threads Run 2	EBS 300 Threads Run 3	EBS 300 Threads Run 4	EBS 300 Threads Run 5
DeleteListItem	3.76	3.1	3.17	3.46	2.93
ListEdit	2.64	2.15	2.14	2.31	1.89
ListReadCoded	0.8	0.7	0.71	0.77	0.74
LoadHomePageCoded	0.63	0.54	0.6	0.7	0.54
NewListItem	2.48	2.02	2.09	2.26	1.91
Read100k	6.54	7.2	7.09	6.82	7.06
Read100Meg	46.8	45	41.4	40.2	46.7
Read10Meg	13.7	14.4	15.4	12.8	15.3
Read1Gig	255	248	259	236	240
Read1Meg	7.32	7.93	7.78	7.67	7.89
Upload100K	4.96	4.15	4.34	4.31	4.16
Upload100Meg	57.5	64.9	64.4	61.5	69.7
Upload10Meg	13.5	15.5	14.4	13.8	16.4
Upload1Meg	6.13	5.68	5.86	5.58	5.59
Average (All Tests)	1.48	1.3	1.37	1.52	1.31

Overall, we do not feel that Flash Cache significantly improves or harms performance with these workloads. However, it should be noted that these test cases were completely random and involved very large file libraries. Smaller libraries or less random distribution of reads might result in improved performance. We did not test “hot” sectors, pages, or documents in this engagement.

7 APPENDIX A: TEST ENVIRONMENT

7.1 STORAGE CONFIGURATION

On the SAN side configuration, all the LUNs were mounted on the Windows hosts using FCP protocol. Details for the hardware configurations and Data Layout are described in the following sections.

On the NAS side, SMB shares were used for the EBS data to create the media devices in SMMOSS Manager with the underlying protocol set to SMB 2.0. All the SMB traffic used cifs.tcp_windows_size as 64240 and a 64K read/write buffer size. By default, cifs oplocks were enabled on the storage system R8-FAS3170B.

The RAID configuration on the storage was configured to have the same RAID group size on both the SAN and NAS side. We configured a 13 disk RAID group size on the SAN storage for the SQL Server metadata with 15000 rpm FC disks and the same RAID group size on the NAS storage for the EBS shares with 7200 rpm SATA disks.

External BLOB Storage (EBS) was configured to externalize any file larger than 1KB, by using a rule with the EBS provider in the SnapManager 5.0 for SharePoint. This is not the recommended setting but it was

purposely done in order to extend all of our test documents with the lowest possible document size directly into the EBS shares.

7.2 LAB CONFIGURATION

LAB LAYOUT

The servers and the storage system layout are as described in Figure 11.

Figure 11) Lab layout.

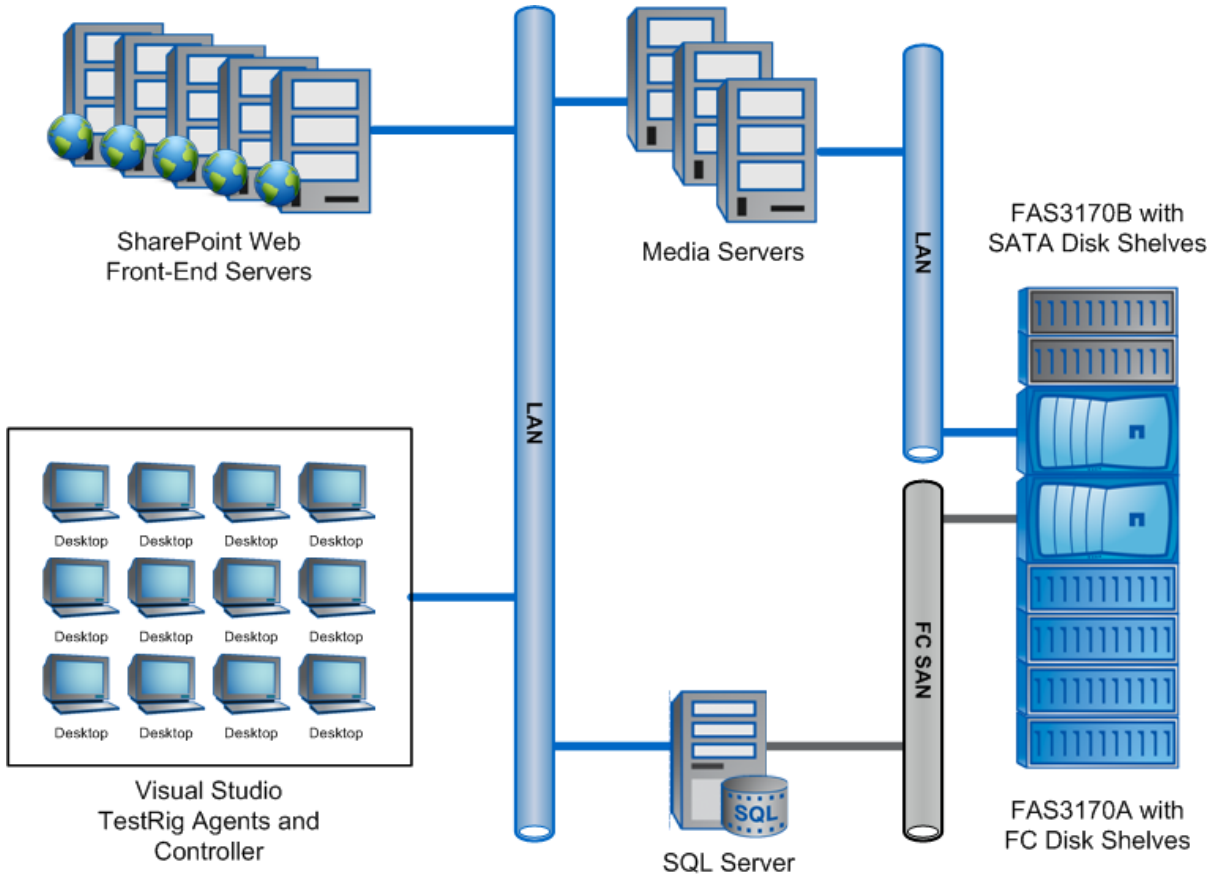


Table 12) Server roles.

Servers	Hostname	Roles	Software	Ports/comments
Presentation Tier				
Web Front Ends	R19-BL4601	Archiver, Extender, Control, Member	SharePoint 2007 SP2	All Web apps were configured on port 80 using host headers
	R19-BL4602	Archiver, Extender, Control, Member	SharePoint 2007 SP2	
	R19-BL4603	Archiver, Extender, Control, Member	SharePoint 2007 SP2	
	R19-BL4604	Archiver, Extender, Control, Member	SharePoint 2007 SP2	
	R19-BL4808	Archiver, Extender, Control, Member	SharePoint 2007 SP2	
Middle Tier				
Central Admin Server	R19-BL4807	Control Service Agent, Media Service	SharePoint 2007 SP2, SMMOSS Manager, SnapDrive 6.2,	12000 12001
Media Servers*	R19-Bl4805, R19-BL4806	Media Service	SnapDrive 6.2	12001
Data Tier (SQL Server)				
Content Databases	R17-DL5851	Member Agent	SQL Server 2008 Enterprise, SMSQL 5.0R1, SnapDrive 6.2	
Visual Studio 2008 Testrig Database	R17-DL5852	Testrig Database	SQL Server 2008 Enterprise	
On the Storage Tier, the configuration was as follows:				
FAS3170	R8-FAS3170A	Storage for SQL Server database with FC disks	Data ONTAP 7.3.3	51 x 300GB FC disks with RG Size 13
	R8-FAS3170B	Storage for EBS data with SATA disks	Data ONTAP 7.3.3	23 x 500GB SATA disks with RG Size 13

Visual Studio Test Rig Configuration: We used Visual Studio 2008 Test Edition for generating the load. We set up 1 central controller and 12 test agents as a part of our test rig. All the test agents/controller were running Windows 7 workstations. The reason to test with 12 test agents was to make sure that the agent machines were not the bottleneck. At no time did our test agents register more than 20% CPU load.

DATA LAYOUT

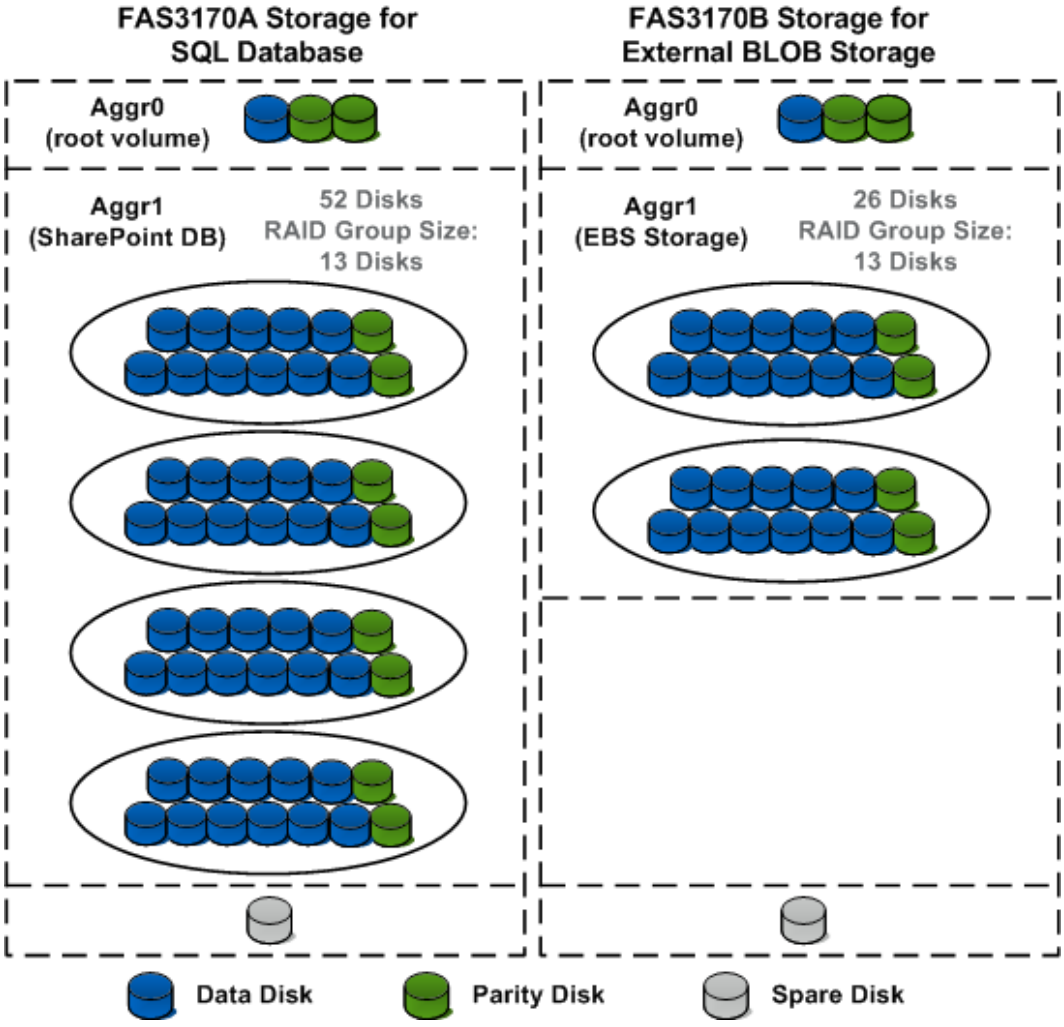
Corpus: 1TB total size in each SQL Server content database

Table 13) Data layout.

Databases	LUNs	SAN Volumes (Size)	CIFS Shares/Volumes
SQL Server Native1	Lundb1 - 300GB, Lunlog1 – 60GB	Sqlvol1 (3500GB)	
SQL Server Native2	Lundb2 - 300GB, Lunlog2 – 60GB	Sqlvol1 (3500GB)	
SQL Server Native3	Lundb3 - 300GB, Lunlog3 – 60GB	Sqlvol1 (3500GB)	
SQL Server Native4	Lundb4 - 300GB, Lunlog4 – 60GB	Sqlvol1 (3500GB)	
SQL Server Native5	Lundb5 - 300GB, Lunlog5 – 60GB	Sqlvol1 (3500GB)	
EBS1	Ebsdb1 – 200GB, Ebslog1 – 40GB, Index1 – 150GB	Sqlvol2 (2500GB)	Ebsvol1 (2000GB)
EBS2	Ebsdb2 – 200GB, Ebslog2 – 40GB, Index1 – 150GB	Sqlvol2 (2500GB)	Ebsvol1 (2000GB)
EBS3	Ebsdb3 – 200GB, Ebslog3 – 40GB, Index2 – 150GB	Sqlvol2 (2500GB)	Ebsvol2 (2000GB)
EBS4	Ebsdb4 – 200GB, Ebslog4 – 40GB, Index2 – 150GB	Sqlvol2 (2500GB)	Ebsvol2 (2000GB)
EBS5	Ebsdb5 – 200GB, Ebslog5 – 40GB, Index3 – 150GB	Sqlvol2 (2500GB)	Ebsvol3 (1000GB)
Index Search **	SSP_Index – 100GB SSP_Search_Index – 250GB SSP_Config_log – 50GB SSP_Search_log – 100GB	Spindex (1000GB)	

** Index Search Servers: The Index Services were started in the SharePoint Web farm, but no indexing/searching tests were included in our test methodology, mostly based on assumption that index crawling would be generally scheduled in off hours or weekends in a production environment.

Figure 12) Storage aggregate layout.



HARDWARE CONFIGURATION

Table 14) Server hardware configuration.

Machine Name	Server Type	Processor(s)	Memory	Hard Drives	FC HBA	OS Installed	NIC ports (on-board)
R17-DL5851	HP DL585 G2	(4) AMD Opteron 8356 (2.3 GHz QC, 2M L3)	64GB	C: (2) 73G 10K (RAID 1)	HP A8003A DP (Emulex)	Windows 2008 R2 Enterprise	2 GBE
R17-DL5852	HP DL585 G2	(4) AMD Opteron 8356 (2.3 GHz QC, 2M L3)	64GB	C: (2) 73G 10K (RAID 1)	HP A8003A DP (Emulex)	Windows 2008 R2 Enterprise	2 GBE
R19-BL4601	HP BL460 G5 Blade Server	(2) Intel® Xeon™ 5140 (2.33GHz DC)	8GB	C: (2) 73G 10K (RAID 0)	Qlogic QMH 2462 DP	Windows 2008 SP2 X64 Enterprise	2 GBE
R19-BL4602	HP BL460 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	8GB	C: (2) 73G 10K (RAID 0)	Qlogic QMH 2462 DP	Windows 2008 SP2 X64 Enterprise	2 GBE

Machine Name	Server Type	Processor(s)	Memory	Hard Drives	FC HBA	OS Installed	NIC ports (on-board)
R19-BL4603	HP BL460 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	8GB	C: (2) 73G 10K (RAID 0)	Qlogic QMH 2462 DP	Windows 2008 SP2 X64 Enterprise	2 GBE
R19-BL4604	HP BL460 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	8GB	C: (2) 73G 10K (RAID 0)	None	Windows 2008 SP2 X64 Enterprise	2 GBE
R19-BL4805	HP BL480 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	12GB	C: (4) 73G 10K (RAID 1)	Emulex LPe1105HP DP	Windows 2008 SP2 X64 Enterprise	4 GBE
R19-BL4806	HP BL480 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	12GB	C: (4) 73G 10K (RAID 1)	Emulex LPe1105HP DP	Windows 2008 SP2 X64 Enterprise	4 GBE
R19-BL4807	HP BL480 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	12GB	C: (4) 73G 10K (RAID 1)	Emulex LPe1105HP DP	Windows 2008 SP2 X64 Enterprise	4 GBE
R19-BL4808	HP BL480 G5 Blade Server	(2) Intel Xeon 5140 (2.33GHz DC)	12GB	C: (4) 73G 10K (RAID 1)	Emulex LPe1105HP DP	Windows 2008 SP2 X64 Enterprise	4 GBE

Note: All servers were patched to current levels including updates from Windows Update. Hardware updates are provided using the Hewlett Packard PSP update process. We used the latest version of the Proliant Support Pack, V8.40.

Table 15) Switches.

Switch	Description
Fibre Channel	Brocade 48000 Director SAN Switch
Ethernet	HP Procure 5412zl
	HP Procure 2900-48G
	HP Procure 3500yl

Note:

- All Fibre Channel connections are made directly and zoned to the SAN from within the switch (no ISL connections).
- Connections between the Ethernet switches was using a 10Gb SC interface. Each switch has a separate link between the 2900/3500 and one of the 5412 switches to reduce the hop count between switches.