Technical White Paper

Everpure

Fast Backup and Recovery for SQL Server at Scale

Native SQL Server backup and restore with FlashBlade

Everpure

Contents

Introduction 3

Everpure FlashBlade 3

Solution architecture 3

Background 4

SQL Server backup/restore: Threads and parallelism 4

Using Windows file shares (SMB) 5

Using object storage (S3) 6

Validation environment and specifications 7

Results 8

Single-instance validation 9

Multi-instance validation 10

Conclusion 12

Additional resources 12

Introduction

SQL Server estates face a fundamental challenge: backup windows need to shrink while instance counts and data volumes continue to grow. When recovery time matters, traditional backup targets become bottlenecks. The constraint isn't SQL Server's native backup capability—it's the storage underneath.

Everpure™ FlashBlade® removes that constraint. As a scale-out file and object storage platform, FlashBlade delivers the rapid backup and recovery throughput SQL Server needs to meet aggressive recovery time objectives (RTOs) while also enabling consolidation of fragmented backup infrastructure onto a single high-performance platform.

This technical white paper explores how FlashBlade solves large-scale SQL Server data protection challenges. It provides baseline performance data and practical guidance for integrating FlashBlade with SQL Server–native backup using Transact-SQL (T-SQL).

Everpure FlashBlade

FlashBlade delivers the throughput SQL Server needs for rapid backup and recovery at scale. Its scale-out architecture means performance grows with capacity, avoiding the single-controller bottleneck that limits traditional backup targets.

FlashBlade supports both Server Message Block (SMB) file shares and Simple Storage Service (S3) object storage as native backup targets. SafeMode™ Snapshots provide immutable backup copies. Replication extends protection to a second FlashBlade or other S3-compatible storage. This paper focuses on the following FlashBlade models for large-scale SQL Server data protection:

- **FlashBlade//S200**: high performance with deep compression; balanced performance and efficiency
- **FlashBlade//S100**: entry-level scale-out; single chassis
- **FlashBlade//E™**: disk-comparable acquisition cost; high capacity, 40% lower total cost of ownership (TCO) than disk

Solution architecture

This solution architecture validates the performance capabilities of SQL Server–native backup with FlashBlade using SMB and S3 as backup targets.

Using SMB as a backup target provides a versatile and widely supported protocol, enabling seamless integration with various operating systems. It allows for straightforward configuration, making it user-friendly for administrators. SMB's user-level authentication enhances security, ensuring that backup access is controlled and authenticated.

Using S3-compatible object storage as a backup target provides a scalable and versatile solution. It allows seamless integration with systems supporting the S3 protocol, offering greater scalability and flexibility. The compatibility ensures a straightforward transition for users familiar with S3, enabling efficient backup and recovery processes. Additionally, S3-compatible storage inherits the durability and reliability associated with the S3 standard, making it a reliable choice for secure and scalable backup storage.

The following outcomes can be realized when combining SQL Server–native T-SQL backup and recovery with FlashBlade file and object storage:

- FlashBlade single chassis provides rapid restore for SQL Server workloads, achieving a data recovery performance of 113TB/hr or more for scalable, parallel operations.
- Multi-protocol support for file and object storage enables flexible data protection and disaster recovery strategies.
- File share and object replication capabilities provide greater resiliency by enabling data to be replicated to another FlashBlade or other S3-compatible storage with disaster recovery capabilities.

Background

Traditional data protection methodologies employed for Microsoft SQL Server are ill-equipped to cope with rapidly growing capacity and performance needs. A close look at these scenarios reveals limitations that act as strong barriers to success, including:

- **Extended recovery times**: Conventional backup systems focus on rapid data protection, often overlooking recovery. With the acceleration of digital transformation, organizations now concentrate intensely on minimizing downtime for business-critical applications to maximize the value derived from their data. Slow recovery adversely affects business operations, introducing a bottleneck that hampers efficiency and responsiveness.
- **Complicated and inefficient storage**: The absence of data reduction in traditional methods creates storage management issues, resulting in inefficient use of storage space and other related problems. This not only wastes storage resources but also increases operational costs, posing significant challenges in terms of scalability and resource optimization.
- **Limited data services**: Traditional storage solutions fall short in providing performance storage solutions with comprehensive data services such as instant recovery, multisite replication, cloud integration, compression, and granular application recovery, which exposes data to heightened vulnerability in the face of unforeseen events.

SQL Server backup/restore: Threads and parallelism

Thread usage is an important and complex topic when tuning SQL Server backup and restore. Using multiple threads allows SQL Server to parallelize the operation, maximizing potential throughput. Two main factors can control this parallelism:

- Number of volumes SQL Server data files reside on for the database being backed up/restored
- Number of backup target files (when striping backups across multiple target files)

If SQL Server is running on a Windows server and the backup target is SMB, throughput can also potentially be increased by creating multiple SMB hostnames via domain name system (DNS) names for the SMB host when connecting to SMB target shares.

Note: Multiple DNS names will not provide additional performance when backing up to object (S3) storage.

Figure 1 shows how reader and writer threads are allocated during a backup operation based on data volume layout and number of backup target files. It also illustrates how Windows handles SMB sessions in scenarios when multiple SMB DNS hostnames are used.

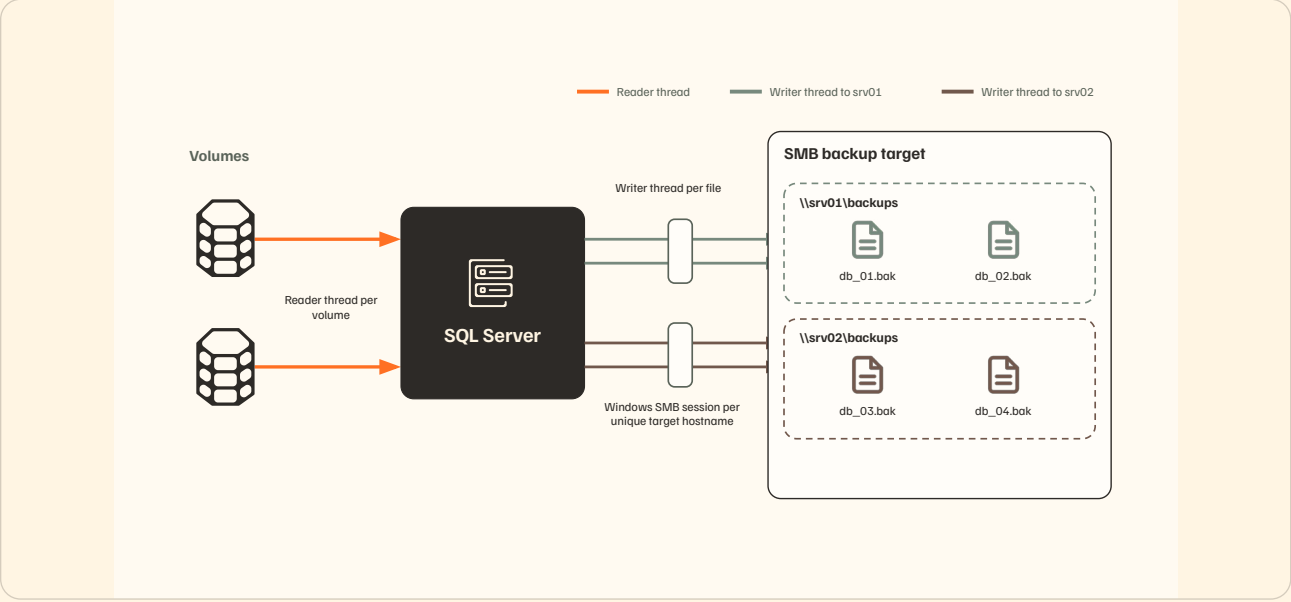


FIGURE 1 SQL Server backup thread parallelism

When SQL Server runs a **backup**, it allocates worker threads based on both the data layout and the backup layout as follows:

- **Reader threads**: One thread is allocated for each volume the data files reside on.
- **Writer threads**: One thread is allocated for each backup target file.

On Windows, SMB throughput is also affected by how SMB sessions are established. Specifically:

- Windows creates one SMB session per unique SMB hostname.
- If you point all backup files at a single SMB hostname, all backup traffic flows through a single SMB session.
- If you instead use multiple SMB hostnames, each with a unique hostname but all mapped to the same SMB share, Windows is forced to create multiple SMB sessions, which typically increases SMB parallelism and aggregate throughput.

When SQL Server executes a **restore**, the thread allocation and SMB session behaviors are similar but in reverse:

- **Reader threads**: One thread is allocated per backup target file.
- **Writer threads**: One thread is allocated for each volume the data files reside on.
- Multiple SMB sessions are created, providing more potential throughput from the backup target to the SQL Server instance.

Deciding how many volumes, backups target files, and SMB hostnames are right for a specific environment requires thorough testing and validation to ensure maximum throughput can be achieved. Based on performance testing in the validation environment, all backup and restore operations were completed using the following parameters:

- Eight database files spread across eight dedicated volumes
- One log file on a dedicated volume
- 24 backup files
- Eight SMB hostnames when writing to an SMB target

Using Windows file shares (SMB)

SQL Server–native backups to SMB are simple, well-tested, and well-supported across all modern versions of SQL Server. SMB shares via FlashBlade file services can be configured and ready in a matter of minutes. For additional information on creating SMB shares on FlashBlade, see the knowledge article Creating an SMB share on FlashBlade.

As stated previously, when backing up to SMB, the choice between a single SMB hostname and multiple data SMB hostnames can impact performance. Choices around how to configure this depends on factors such as management simplicity, throughput requirements, and concurrency goals. Multiple SMB hostnames are often favored when seeking to optimize performance, but the number used should be tested and tuned per environment.

The following T-SQL command can be used to create a full database backup to an SMB file share using multiple SMB hostnames:

```
BACKUP DATABASE <database_name> TO
  DISK = '\\data1\<fileshare_name>\db_01.Bak',
  DISK = '\\data2\<fileshare_name>\db_02.Bak',
  DISK = '\\data3\<fileshare_name>\db_03.Bak',
  DISK = '\\data4\<fileshare_name>\db_04.Bak',
  DISK = '\\data5\<fileshare_name>\db_05.Bak',
  DISK = '\\data6\<fileshare_name>\db_06.Bak',
  DISK = '\\data7\<fileshare_name>\db_07.Bak',
  DISK = '\\data8\<fileshare_name>\db_08.Bak'
WITH INIT, [NO_COMPRESSION | COMPRESSION, MAXTRANSFERSIZE=<maxtransfersize>, BUFFERCOUNT=<buffer_count>]
```

The following T-SQL command can be used to restore a database from a full backup located on an SMB file share using multiple data SMB hostnames:

```
RESTORE DATABASE <database_name> FROM
   DISK = '\\data1\<fileshare_name>\db_01.Bak',
   DISK = '\\data2\<fileshare_name>\db_02.Bak',
   DISK = '\\data3\<fileshare_name>\db_03.Bak',
   DISK = '\\data4\<fileshare_name>\db_04.Bak',
   DISK = '\\data5\<fileshare_name>\db_05.Bak',
   DISK = '\\data6\<fileshare_name>\db_06.Bak',
   DISK = '\\data7\<fileshare_name>\db_07.Bak',
   DISK = '\\data8\<fileshare_name>\db_08.Bak'
WITH REPLACE
```

Microsoft SQL Server documentation provides additional details on backup parameters to further tune backup and restore operations.

Using object storage (S3)

SQL Server 2022 introduced support for S3-compatible object storage, facilitating backup and recovery workflows through a URL syntax that leverages the S3 REST API to integrate with any S3-compatible object storage provider. For additional information on creating object buckets on FlashBlade, refer to the knowledge article Creating an S3 Bucket in FlashBlade.

Note: SQL Server requires a valid and trusted certificate to use object storage on FlashBlade as a backup target. This either requires adding a valid and trusted certificate to the FlashBlade or exporting the FlashBlade certificate and importing it to your SQL Server hosts.

For large databases, configuring FlashBlade object store as a backup target for SQL Server is simple and enables seamless communication and secure data transfer over HTTPS. It improves performance by dividing the backup into multiple files, allowing concurrent operations. This capability is linked to endpoints (URLs) associated with the object store.

The following T-SQL command can be used to create a full database backup in an object bucket:

```
BACKUP DATABASE <database_name> TO
    URL= 's3://data/<bucket_name>/<folder_name>/db_01.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_02.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_03.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_04.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_05.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_06.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_07.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_08.Bak'
WITH INIT, [NO_COMPRESSION | COMPRESSION, MAXTRANSFERSIZE=<maxtransfersize>]
```

The following T-SQL command can be used to restore a full database backup located in an object bucket:

```
RESTORE DATABASE <database_name> FROM
    URL= 's3://data/<bucket_name>/<folder_name>/db_01.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_02.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_03.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_04.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_05.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_06.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_07.Bak',
    URL= 's3://data/<bucket_name>/<folder_name>/db_08.Bak'
WITH REPLACE
```

Microsoft SQL Server documentation provides additional details on backup parameters to further tune backup and restore operations.

Limitations

While S3 offers a flexible method to back up your SQL Server databases, the SQL Server S3 connector (in conjunction with the S3 API) comes with a few limitations:

- Individual backup files have a maximum size of 195GB. When a backup is created, each file can be split into a maximum of 10,000 parts of the size **MAXTRANSFERSIZE** (default of 10MB, maximum of 20MB).
- A backup set can be a maximum of 12.7TB when backing up to URL. For very large databases, compression can help keep the backup set under this limit.
- If using a nondefault **MAXTRANSFERSIZE** parameter when backing up to URL, the use of compression is **required**.

For full details, see the Microsoft SQL Server documentation on backing up to URL.

Validation environment and specifications

Database backup and restore performance can be impacted by a large number of factors, including but not limited to:

- Host-specific bottlenecks such as CPU and memory overhead
- Storage bottlenecks such as array configuration and resource limits
- Network bottlenecks such as number of hops between the SQL Server instances and the backup target and bandwidth
- SQL Server features and backup settings such as compression and encryption

The validation environment was designed with this in mind, trying to reflect real-world configurations while also ensuring as few bottlenecks as possible. Figure 2 provides a high-level overview of the validation environment.

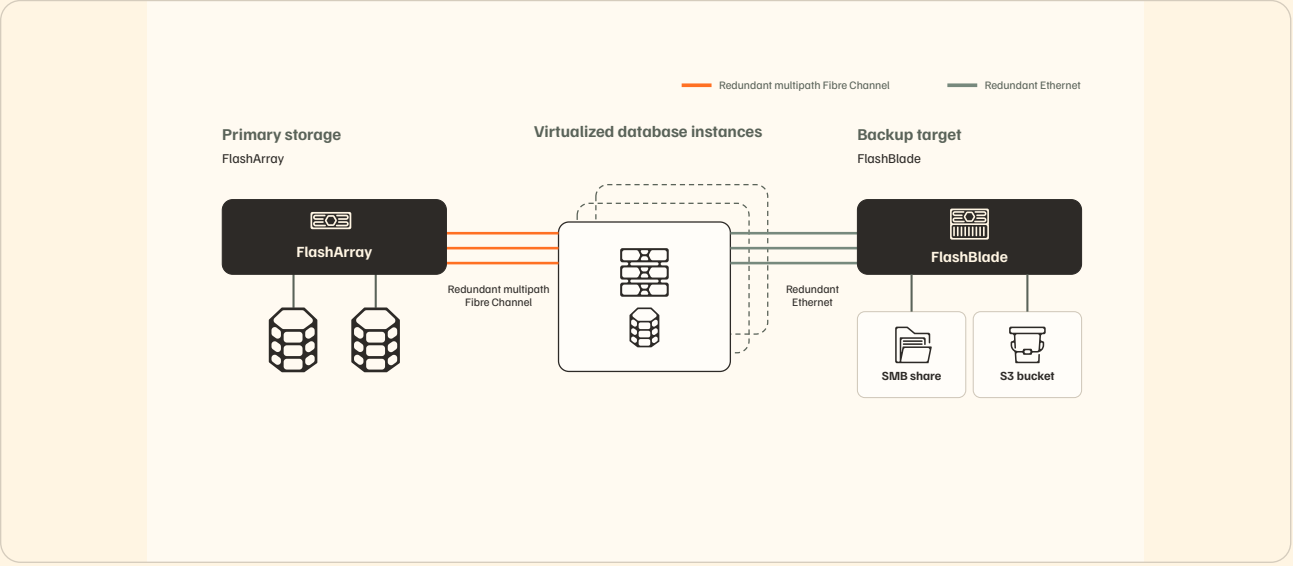


FIGURE 2 Solution architecture overview

Primary storage

Primary data storage was allocated across four FlashArray™ devices. Each FlashArray was connected to the virtual SQL Server hosts via high-speed Fibre Channel. All volumes were formatted using the New Technology File System (NTFS) with a 64K allocation unit size per Microsoft best practices.

Backup target(s)

Single-chassis, fully configured (10x4) FlashBlade devices (FlashBlade//S100, FlashBlade//S200, and FlashBlade//E) were used as backup targets. Each FlashBlade was configured with a single SMB share, with eight SMB hostnames referencing it, and a single object storage bucket. All backup and restore traffic was sent over a dedicated 100GbE network connecting the hosts to the FlashBlade devices.

Server configuration

Each virtual database host had 48 vCPUs and 128GB of memory, running SQL Server 2025 Enterprise Edition on Windows Server 2025. For an optimal CPU configuration on the SQL Server host, it is advised to have a number of cores greater than the count of both data volumes and backup files. As shown in Figure 1, reader threads are allocated per volume and writer threads are allocated per backup file. In order to reduce CPU contention, there has to be enough CPU cores to handle the number of reader and writer threads created for a backup/restore process.

Database configuration/layout: Per Figure 1 and the accompanying discussion of threads and parallelism, all instances were configured with nine 500GB volumes. Eight data files were spread across eight volumes, and one volume was dedicated to the log file.

Backup configuration

The backup and restore operations were carried out using a PowerShell testing harness that executes T-SQL scripts. The testing harness generates backup/restore scripts based on defined parameters, records backup timings and size, and coordinates concurrent backups/restores across the validation instances. All backups were run with the following parameters:

- Based on the limitations of object/S3 backups discussed previously, COMPRESSION is used for all backups to object storage in these validations.
- **MAXTRANSFERSIZE = 20M (for S3) and 4M (for SMB).**
- **BUFFERCOUNT = 512.**

MAXTRANSFERSIZE and **BUFFERCOUNT** settings were chosen to encourage maximum throughput to FlashBlade backup target(s). These settings are environment-specific and should be thoroughly tested to balance database instance resource usage and backup throughput. To learn more about how these settings can impact performance, refer to Microsoft SQL Server documentation.

Results

This section provides a backup and restore throughput comparison obtained from validations performed on database(s) using SMB and object storage as backup targets on FlashBlade//E, FlashBlade//S100, and FlashBlade//S200. Validations were conducted in both single- and multi-instance scenarios, as well as with and without native SQL Server backup compression.

Throughput figures are based on total capacity used by the database and the total duration of the validation from start to finish, not simply the peak throughput during the operation. These numbers represent real time to backup/restore. Compressed backups typically have a higher throughput since less data is moving between the SQL Server instances and the FlashBlade.

The following test configurations outline the environments and conditions used during the backup and restore validations:

- **Single instance**: A single instance with 48 cores hosting one database using the full recovery model, with eight data files and one log file distributed across nine volumes to ensure ample concurrency during database backup and restore operations.
- **Multi-instance**: To showcase FlashBlade concurrency, efficiency, and performance, eight instances were deployed in this solution, each with the same configuration used in the single-instance validations.
- **Compression**: SQL Server backup compression is a widely adopted practice to minimize storage needs and/or decrease network load. Although backup compression helps save storage space, it does incur host CPU cycle costs. For example, the increased CPU demands become a bottleneck in some cases, limiting backup and restore throughput. Thus, finding the right balance is a trade-off based on environment specifics, and there isn't a one-size-fits-all recommendation.

All compressed validations were completed using the newer Zstandard (ZSTD) compression available in SQL Server 2025 with the default LOW compression level. This compression has the advantage of lower CPU usage on the SQL Server instance while still providing good

compression. During initial validation testing, MS_XPRESS compression consumed approximately twice the CPU resources compared to ZSTD, with little difference in throughput. Using ZSTD ensures that CPU is not a bottleneck during validations.

It is also important to note that object testing was only performed with compression enabled since compression must be used when changing the **MAXTRANSFERSIZE** setting.

Single-instance validation

Single-instance performance is closely tied to specific bottlenecks within an environment. While our validations were designed to focus on the outstanding concurrent backup and restore capacity of the FlashBlade, single-instance validations were also run to understand bottlenecks in the validation environment.

Findings

Overall, performance increased as expected across the various FlashBlade models during validation. When running validations against the FlashBlade//S200, the maximum throughput of the individual hosts was uncovered. Even for single-instance backup and restore tasks, the FlashBlade performs consistently when compared to the multi-instance validations—it is able to restore a 1TB database in about three minutes.

During single-instance validations, some final settings were determined for maximum performance in this environment:

- Striping database backups across 24 files gave the best backup/restore performance.
- ZSTD compression gave the best host CPU/compression balance over standard compression (as noted previously, MS_XPRESS used approximately twice the CPU resources compared to ZSTD).
- Adding more data SMB hostnames for SMB backups increased performance with diminishing returns. Eight worked best to balance performance and real-world manageability, but determining the correct number requires thorough testing in each environment.

Figures 3 and 4 show the throughput numbers with compression enabled.

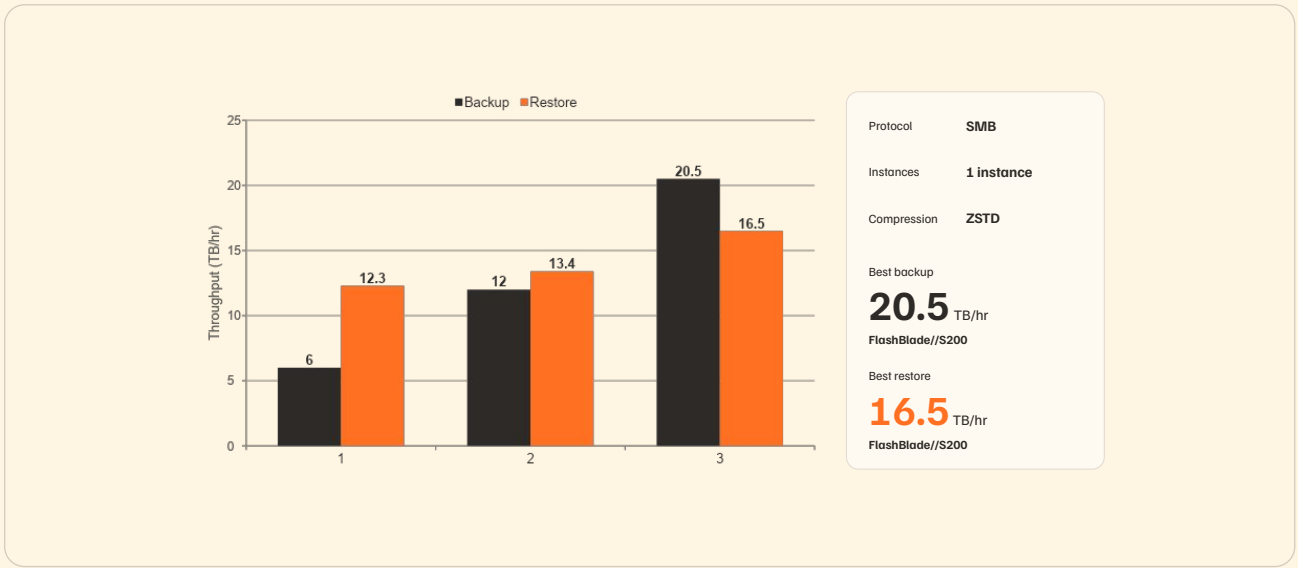
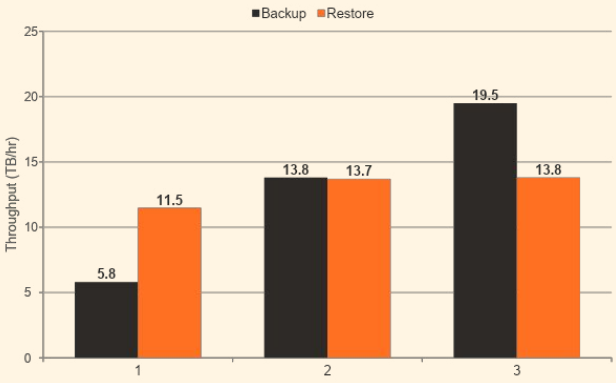


FIGURE 3 3Single-instance backup and restore performance via SMB



Protocol **Object (S3)**

Instances **1 instance**

Compression **ZSTD**

Best backup

19.5 TB/hr

FlashBlade//S200

Best restore

13.8 TB/hr

FlashBlade//S200

FIGURE 4 4Single-instance backup and restore performance via object storage

Multi-instance validation

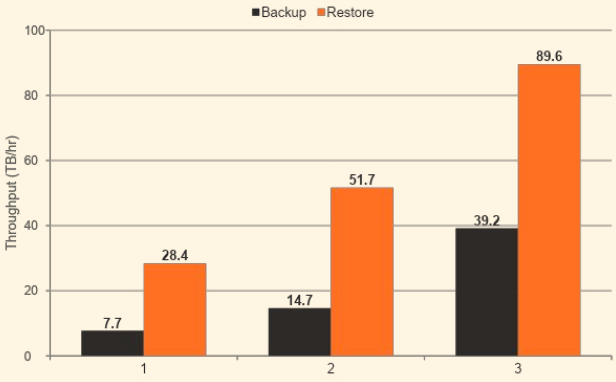
During the multi-instance validation phase, simultaneous backup and restore operations were conducted across eight instances, each hosting a single 1TB database. All the tuning techniques acquired from the single database validation were applied during this process.

The multi-instance validation clearly showcases the ability of FlashBlade to provide high performance and scalability, handling simultaneous backup and restore operations while maintaining predictable and consistent performance.

Figures 5–7 provide a comparison between the results obtained from multi-instance validations using both SMB and object storage as backup targets, as well as with and without SQL Server–native backup compression.

Backup/restore with SMB

Figures 5 and 6 illustrate the multi-instance validation results using SMB as a backup target.



Protocol **SMB**

Instances **8 instances**

Compression **None**

Best backup

39.2 TB/hr

FlashBlade//S200

Best restore

89.6 TB/hr

FlashBlade//S200

FIGURE 5 Figure 5. Summary of multi-instance validation using SMB **without** compression

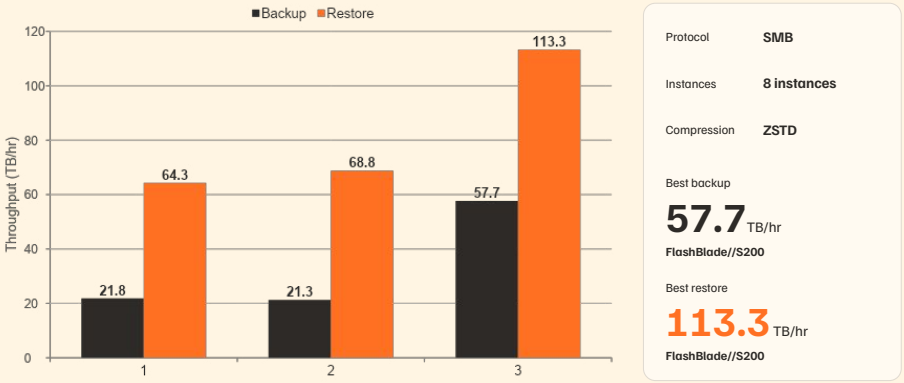


FIGURE 6 Summary of multi-instance validation using SMB **with** compression

Backup/restore with object (S3)

Figure 7 illustrates the multi-instance validation results using object storage as a backup target.

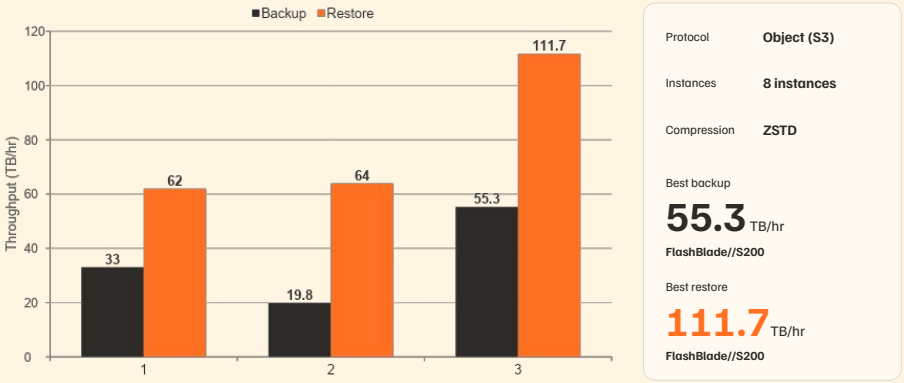


FIGURE 7 7Summary of multi-instance validation using object storage with compression

Conclusion

While each FlashBlade model provides exceptionally fast database restores, each comes with its own performance characteristics and capacities. From a data protection standpoint, this ensures that there is a FlashBlade model to meet the needs of any enterprise using SQL Server in its mission-critical applications.

FlashBlade//S200 provides enough horsepower and capacity to handle the data protection needs of very large SQL Server estates. In validations, we were able to reach 113TB/hr restore rates across eight SQL Server instances. While that number is impressive, it is limited by the Fibre Channel connectivity to the restore targets, not the FlashBlade throughput capacity. Scaling from four instances to eight had no impact on the restore time per instance, meaning that 113TB/hr is by no means the performance ceiling of this robust appliance.

FlashBlade//S100 is suitable for mid-sized SQL Server environments with lower capacity and performance requirements than would justify a FlashBlade//S200. While it is lower performance than the FlashBlade//S200, a restore rate of 69TB/hr is still an impressive number, putting average restore time of a 1TB database at around six minutes.

FlashBlade//E is optimized for capacity, topping out at an impressive 60PB of raw capacity per cluster. Despite scoring slightly lower than the FlashBlade//S100 in our validations, it would still make a solid primary backup target for a smaller organization that can take advantage of its capacity for other purposes, or as a cold storage appliance for offloading data from the more performant FlashBlade//S™ series appliances.

Regardless of the model, FlashBlade offers the flexibility of using native SMB, object/S3, or both in an organization's data protection strategy. Combined with other features like SMB replication (FlashBlade to FlashBlade), object replication to S3-compatible storage providers (including FlashBlade to FlashBlade), SafeMode immutable snapshots, and policy-driven retention, FlashBlade enables resilient, multi-tier SQL Server data protection architectures that span on-premises and cloud environments. This combination of protocol flexibility, massive concurrency, and robust data services empowers organizations to meet recovery point objectives (RPOs) and RTOs today while retaining the freedom to scale, modernize, and evolve their SQL Server estates without constantly rearchitecting their backup infrastructure.

Additional resources

- Explore Best Practices for Microsoft SQL Server on FlashArray.
- Read more about SQL Server backup to URL for S3-compatible object storage.

Learn More About FlashBlade

Visit Our Website | **800.379.PURE**

©2026 Everpure, the Everpure P Logo, Pure Storage, FlashArray, FlashBlade, FlashBlade//E, FlashBlade//S, SafeMode Snapshots, and the marks in the Everpure Trademark List are trademarks or registered trademarks of Everpure, Inc. or its licensed subsidiaries in the U.S. and/or other countries. The Trademark List can be found at everpuredata.com/trademarks. Other names may be trademarks of their respective owners.

ID56535-01-en-11/26