

TECHNICAL WHITE PAPER

Deploying Vertica in Eon Mode on Pure FlashBlade®

Gain cloud-like simplicity and flexibility on-premises at scale.



Contents

Introduction	3
Executive Summary of Benefits and Capabilities	3
New Capabilities	4
Technology Overview	5
Vertica in Eon Mode	5
Pure Storage FlashBlade.....	6
Solution Overview	6
Design Topology.....	7
Core Solution Benefits	8
Scalability and Capacity	8
Operational Efficiency.....	9
Database Performance: Enterprise Mode vs. Eon Mode	10
Development, Testing, and Offload Solutions.....	11
Solution Overview.....	11
Fast CopyObject Capabilities.....	12
Creating a Fast Crash-consistent Copy of DB	13
Bringing the Database Copy Online.....	14
Disaster Recovery and Backup Solutions.....	17
FlashBlade Object Replication	17
FlashBlade as VBR Target and Source.....	18
FlashBlade Replication for Disaster Recovery	19
Vertica VBR Backup and Restore	23
Vertica Data Protection and Offload Best Practices	25
Example Automation	25
Summary	25
Additional Resources	26
Appendix: Best Practices for Deploying Vertica with FlashBlade Storage	27



Introduction

The Vertica Analytics Platform provides an advanced SQL data warehouse, enabling real-time, advanced analytics on massive datasets. Enterprises deploy Vertica as an analytical database system, writing familiar SQL queries that can instantly analyze datasets that scale into petabytes in size. Historically, the Vertica architecture has supported distributed, direct-attached storage (DAS). While this approach satisfied the business and IT demands of 10 to 15 years ago, it limits the agility you need to scale and manage your deployments in today's dynamic IT environments.

Recently introduced, Vertica in Eon Mode is a new architecture and deployment model that disaggregates compute and storage into independently scaled pools. Unlike the traditional Vertica in *Enterprise Mode*, Vertica in Eon Mode requires a reliable, scalable, AWS S3-compatible object store as communal storage for all user data. Toward that goal, Pure Storage® and Vertica have partnered to deliver Vertica in Eon Mode for Pure Storage FlashBlade. Vertica originally architected Eon Mode to support cloud-native applications. In 2019 Vertica combined it with Pure's unified fast file and object store, FlashBlade, to deliver improved scale, operational efficiency, and equal or better performance for on-premises Vertica environments.

Executive Summary of Benefits and Capabilities

This paper highlights the ease of integration between Vertica in Eon Mode and Pure FlashBlade. It includes realistic examples of the benefits of the integrated solution, new capabilities, and best practices for deployment, including:

- **Simplicity and Scalability:** Scale to multi-petabyte capacity independent of compute.
 - With the separation of storage and compute, you can expand storage without having to add nodes. This eliminates the exponential growth in complexity that comes with node proliferation while scaling with DAS-based systems.
 - Vertica's columnar compression, combined with Pure FlashBlade compression and advantages of having a centralized fault-tolerant storage system, provides twice the effective storage capacity than a similarly configured Vertica Enterprise system on DAS.
- **Operational Efficiency:** Cluster-node additions and removals are 15 times faster.
 - Vertica teams need to focus on tasks that generate business value. They want to bring in the right data sources, perform ETL, aggregate, and format the data to make it available to data and business analysts. They need to spend minimal time on upkeep and maintenance associated with adding nodes, patching, and updating them.



- You can expand or contract clusters at will without rebalancing them, which can consume time and resources. For example: Adding eight additional nodes to an existing eight-node cluster took just five minutes on Vertica Eon Mode with FlashBlade compared to 74 minutes on a similar DAS-based system.
- **Database Performance:** Achieve superior load performance and better query performance.
 - Vertica end-users expect consistent and fast query performance so they can perform the analytics they need. With an all-flash storage system, Vertica and Pure is the only solution available today that can match their expectations.
 - Our test showed that the solution took 54 minutes to load 10TB of data vs. 71 minutes on DAS-based systems.
 - When we ran the same set of queries from four concurrent user sessions, the test results showed that Vertica in Eon Mode delivered similar or better performance nodes when compared to Enterprise Mode.
- **Resilience:** Benefit from disaster recovery and space-efficient rapid-cloning.
 - The availability of accelerated workflows for backup and recovery, and space-efficient rapid cloning to provide dev/test teams with isolated work environments, makes the solution resilient to failures and errors.

When you consider the solution's performance with the scalability, simplicity, resilience, and operational efficiency of Vertica Eon Mode on Pure FlashBlade, you can focus on adding value to the business instead of keeping the lights on.



New Capabilities

With the release of Vertica 10.x and Purity 3.x, the following capabilities have been added to the solution:

- **Disaster Recovery and Backup Solutions:** With Purity//FB 3, FlashBlade now offers asynchronous object replication. Combined with the Vertica Backup and Recovery utility, this provides accelerated workflows to protect and recover Vertica databases by backing up data to local or remote sites.
- **Space Efficient Rapid Cloning—Fast CopyObject Capabilities:** Use incredibly fast metadata operations to make a space-efficient copy of all objects in a Vertica Eon mode database. Use a replica clone to offload computation, for dev/test, or to validate your disaster recovery, etc. In the experiment, we created 10 copies of a 17.2TB database (50TB uncompressed) that each took about 5.20 seconds to create without using additional storage. Compare to a similar operation on Vertica Enterprise Mode that could have taken hours or days with 20 times the storage requirements.
- **Development, Test, and Query Offload Solution:** Take advantage of FlashBlade to clone and start using very large Vertica data warehouses in minutes. Starting with Purity//FB 3, you can quickly create space-efficient clones of Vertica Eon databases for developers, data scientists, and test environments.



Technology Overview

Vertica in Eon Mode

Vertica is a column-oriented relational SQL database for analytics, originally built on a distributed shared-nothing architecture. In its original implementation, you would deploy a Vertica database with distributed DAS (DDAS) nodes called Vertica Enterprise. In Enterprise Mode, each commodity server in a cluster stores shards of data on locally attached disks.

More recently, Vertica introduced Vertica in Eon Mode, which decouples the storage and compute in the cluster into distinct pools. When running Vertica in Eon Mode, you keep the storage layer of the database in a communal, Amazon S3-compatible object storage location, separate from the compute nodes. You can optionally keep data on the nodes in the Eon Mode “depot,” an on-disk cache-like transient storage location. This intermediate layer of data storage provides a faster copy of the data that is local to each node while keeping all records consistent in the communal storage. Data frequently used by queries takes priority in the depot. You can also use the depot to store newly written data before storing it in communal storage.

With traditional DDAS architectures, storage management quickly grows with cluster and data size. As a result, management issues like maintenance upgrades and node membership can dominate operational workflows. This contrasts with a disaggregated architecture where the depot is transient. A disaggregated architecture enables you to maintain the compute nodes in a minimal state, making maintenance operations on compute nodes several orders of magnitude faster and simpler. Vertica and Pure jointly address these operational challenges of traditional DDAS architecture, enabling you to focus on customer experience while maintaining an agile infrastructure footprint.

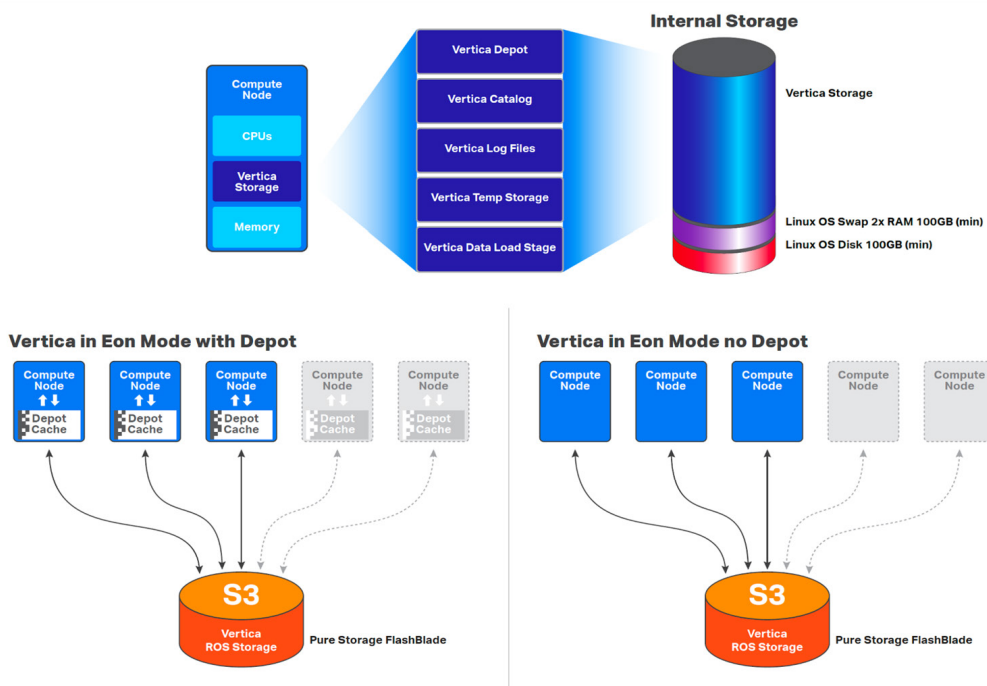


Figure 1. The architecture of a Vertica in Eon mode database

Pure Storage FlashBlade

Pure Storage developed FlashBlade as a high-performance, cloud-optimized storage platform to address the data requirements of modern applications. FlashBlade is an all-flash system, primarily optimized for storing and processing unstructured data. A FlashBlade system can simultaneously host multiple file systems and multi-tenant object stores for thousands of clients. Six key innovations enable you to scale FlashBlade for performance, capacity, and connectivity:

- **High-performance storage:** FlashBlade maximizes the advantages of an all-flash architecture by storing data in storage units and ditching crippling, high-latency storage media such as traditional spinning disks and conventional solid-state drives. Integrating scalable NVRAM into each storage unit helps performance and increases capacity scale proportionally when you add new blades to a system.
- **Unified network:** A FlashBlade system consolidates high communication traffic between clients and internal administrative hosts into a single, reliable, high-performing network that supports both IPv4 and IPv6 client access over Ethernet links up to 100Gb/s.
- **Purity Operating Environment:** Pure's symmetrical operating system runs on FlashBlade fabric modules. It minimizes workload-balancing problems by distributing all client operation requests evenly among the blades on FlashBlade storage.
- **Common media architectural design for files and objects:** FlashBlade's single underlying media architecture supports concurrent access to files via protocols including NFS, fast Amazon S3 object storage, NFS over HTTP, and SMB across the entire FlashBlade configuration.
- **Simple usability:** Purity//FB on FlashBlade storage alleviates system management headaches as it simplifies storage operations by performing routine administrative tasks autonomously. With a robust operating system, FlashBlade storage can self-tune and provide system alerts when components fail.
- **Pure1®:** The Pure1 platform provides simple cloud-based management and effortless predictive support with full-stack analytics and the AI-driven power of Pure1 Meta™. Pure1 is the cornerstone of Pure Storage's world-class NPS score.

A complete FlashBlade system configuration consists of up to 10 self-contained rack-mounted chassis interconnected by high-speed links to two external fabric modules (XFM). At the rear of each chassis are two on-board fabric modules for interconnecting the blades, other chassis, and clients using TCP/IP over high-speed Ethernet. Each of the two fabric modules is interconnected, and each contains a control processor and Ethernet switch ASIC. For reliability, each chassis is equipped with redundant power supplies and cooling fans.

The front of each chassis holds up to 15 blades for processing data operations and storage. Each blade assembly is a self-contained compute module equipped with processors, communication interfaces, and either 17TB or 52TB of flash memory for persistent data storage. You can deploy a typical FlashBlade system in the smallest possible configuration of 7 x 17TB and scale it non-disruptively to many petabytes in the highest configuration of 150 x 52TB.

Solution Overview

The solution presented here consists of the Vertica Analytics Platform on FlashBlade. The multi-dimensional performance and scalability of FlashBlade make it ideal for deploying and managing a Vertica database. We deployed the database into a single Amazon S3 bucket created on FlashBlade. The disaggregated compute and storage architecture enables you to start with the smallest possible FlashBlade configuration and scale the storage and compute layers independently based on your needs. We conducted tests to address common enterprise concerns and highlight the following:



- Capacity savings
- Operational efficiency
- ETL performance
- Query performance

Design Topology

The solution tested in our lab includes 16x Intel CPU-based servers for hosting the Vertica database nodes and a single Pure Storage FlashBlade chassis with 15 blades to host the data storage layer. We used a single SSD in each server as local storage for the Enterprise Mode tests and the same SSD for hosting the depot for the Eon Mode tests.

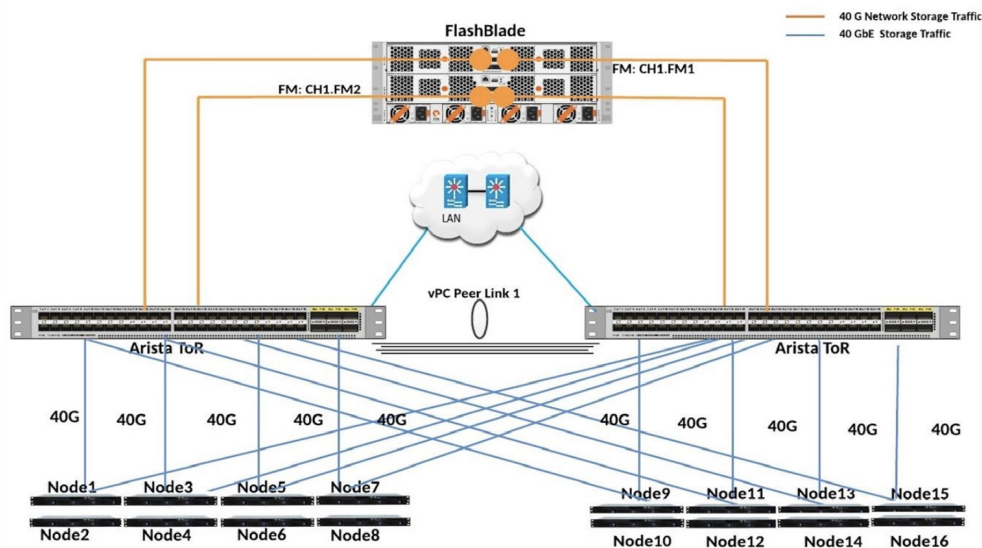


Figure 2. Test environment for Vertica in Eon Mode testing – physical topology

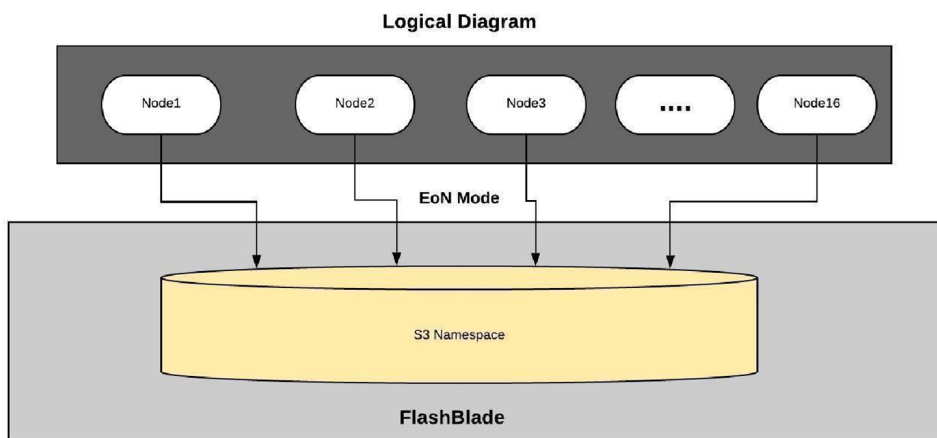


Figure 3: FlashBlade nodes.



Component	Description
FlashBlade	15 x 17TB blades
Capacity	240TB raw; 162.46TB usable (with no data reduction)
Connectivity	4 x 40 Gb/s Ethernet (data); 2 x 1 Gb/s redundant; Ethernet (management port)
Physical	4U
Software	Purity//FB 3.0

Table 1. FlashBlade Configuration

Component	Description
Vertica Nodes	16x Intel-based server each with: 2x Intel Xeon CPUs (48 cores); 512GB of memory; 1TB local SSD
Network Interface Card	2x 40 Gbps ports
Network Switch	TOR 2x Arista Ethernet Switches

Table 2. Server and Network Configuration

Core Solution Benefits

By disaggregating storage from compute, this joint solution of the Vertica Analytics Platform with Pure FlashBlade has three major areas of benefit: capacity savings, operational efficiency improvements, and increases in core database performance.

Scalability and Capacity

Vertica is a column-oriented relational SQL database that stores the data in a pre-compressed format to optimize storage capacity and performance. In an Enterprise Mode deployment of the Vertica database, you replicate the data to more than one node to protect from node failures. The number of replicas is defined by the k-safety factor of a database. The default k-safety factor is 1 (two copies of each shard), and you can use k-safety factors higher than 1 to protect from more than one simultaneous node failure.

While a higher k-safety factor can improve node reliability in an Enterprise Mode deployment model, it also increases capacity and write performance requirements on the storage side. By contrast, the disaggregated storage and compute architecture of the Vertica in Eon Mode model stores only one copy of the data on Amazon S3 storage, irrespective of the k-safety factor setting.

Figure 4 compares the storage consumed by an identical 10TB database in Eon and Enterprise Modes with a default k-safety factor of '1'. The 10TB (TPC-DS like) database gets pre-compressed to 3.8TB because of Vertica's advanced columnar compression. Eon Mode provides additional storage savings by eliminating the need for replicas. Compare this with Enterprise



Mode, which requires you to store two copies of the data—thereby increasing the storage requirement to ~8TB. This points to another important benefit of Eon Mode in some deployment scenarios: By eliminating the need to store multiple copies on the backend storage, you can significantly reduce requirements for higher write performance.

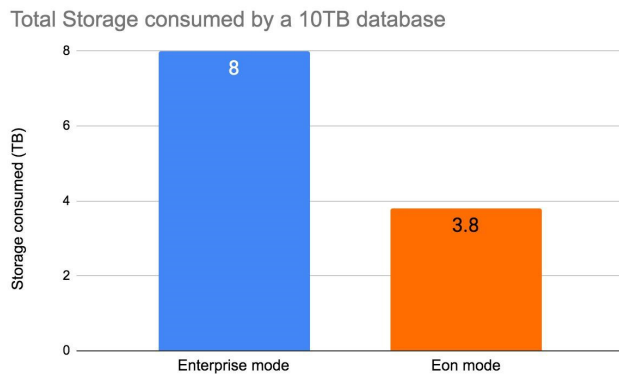


Figure 4. Capacity savings with Eon Mode

Also, with the separation of storage and compute, you can expand storage without having to add additional nodes. This eliminates the exponential growth in complexity that comes with node proliferation at scale.

Operational Efficiency

A second advantage of a disaggregated compute and storage architecture is operational efficiency. This includes how easily or quickly you can scale an environment to meet new business challenges or apply a patch in a maintenance release. Figure 5 shows the time in minutes it took to scale a Vertica database cluster from 8 compute nodes to 16 nodes on a live 10TB database. With the Enterprise Mode deployment model, you need to redistribute the data physically after adding new nodes to take advantage of additional compute. The database rebalance operation initiated after adding eight additional nodes took 74 minutes to complete on a database that is online but not running any queries. It could have taken longer with active queries running against that database.

By contrast, it took only five minutes with the Eon Mode deployment model to virtually redistribute an identically sized database. With Eon Mode, this becomes a simple logical mapping operation that doesn't involve any real data movement. The result is a 15x improvement in the time to scale the cluster environment, and we expect the time to redistribute data will scale with the size of the database and k-safety factor. For production databases, rebalancing data is likely to impact query and data loading performance. Many enterprises use Vertica to run databases that are hundreds (or thousands) of terabytes in size and have k-safety greater than 1; therefore, it may take you days of impacted performance to completely rebalance the database across the additional nodes.

The chart on the right of Figure 5 compares query performance between 8 and 16 nodes. Increasing the number of nodes to 16 shows an almost 50% reduction in query runtime as expected. The result underscores the flexibility and advantages of using disaggregated Vertica in Eon Mode with Pure Storage FlashBlade. Importantly, note that the time required to rebalance a database grows with larger database sizes in the Enterprise Mode deployment model, whereas it would remain constant with Eon Mode deployments. This allows you to easily scale your compute based on your dynamic needs. You can not only scale up your compute nodes, but also scale them down during off-peak times, thereby significantly improving the power and cooling efficiencies of your deployments.



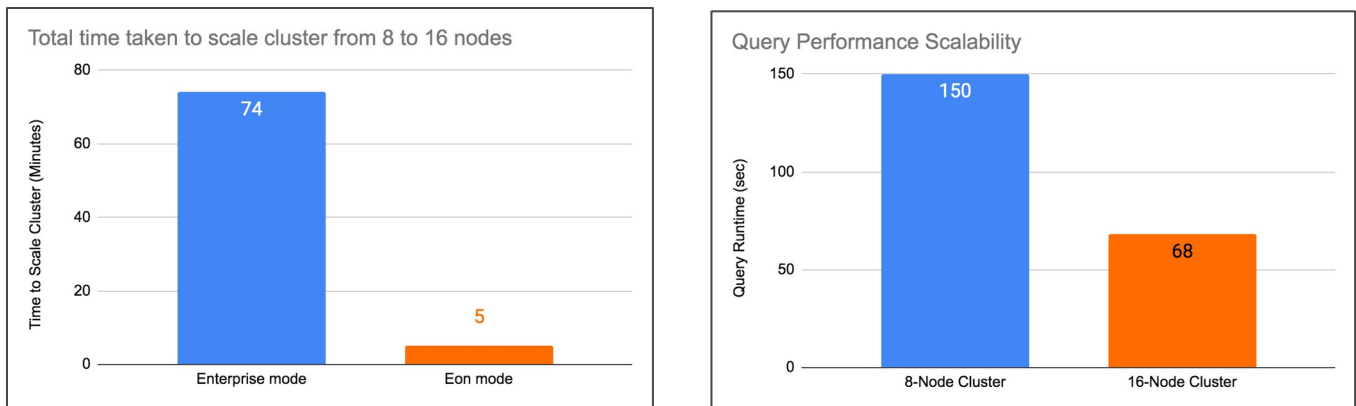


Figure 5. Enterprise vs. Eon Mode operational efficiency

Database Performance: Enterprise Mode vs. Eon Mode

Although Eon Mode delivers improvements in operational efficiency, it may cause concerns about the performance capabilities of the Amazon S3 storage protocol. Possibly you equate Amazon S3 storage to “cheap and deep” storage, but this is no longer the case. While Amazon S3 storage may have originated as a cheap and dense storage option in public clouds, it is quickly evolving as a protocol of choice for modern analytic applications.

Pure Storage FlashBlade is the industry's first Unified Fast File and Object (UFFO) storage appliance designed with modern application stacks and their requirements in mind. Pure Storage and Vertica jointly collaborated to optimize access patterns of Vertica databases to take advantage of the multi-dimensional performance and scalability of FlashBlade storage. FlashBlade storage is specially optimized for high-bandwidth applications and well-suited for Vertica's ROS (Read-Optimized-Store) data format. All our lab testing indicates that the performance of Vertica in Eon Mode databases is either on par if not better than Enterprise Mode (with appropriate tuning as outlined in [Appendix A](#)).

Data Load Time (10TB of data)

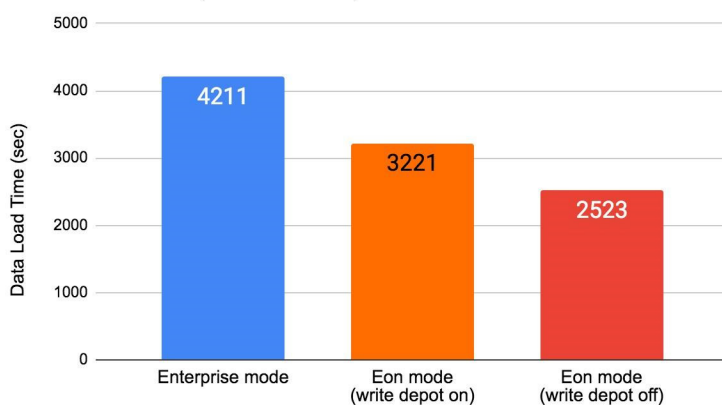


Figure 6. Database load times – typical ETL operation

Figure 6 compares data load time for a 10TB Vertica database in both Enterprise and Eon Modes. For this test, we also varied the write depot setting to quantify its impact on database loads. When UseDepotForWrites is enabled, data is written into the depot before getting flushed into Amazon S3 storage. The data from shows that Enterprise Mode took the longest time (71 minutes) to ingest the 10TB of data, as the database needs to write two copies of data at a minimum to protect from node



failures. Increasing the k-safety factor can further increase the data load times of Enterprise Mode databases. We loaded the same 10TB data in 54 and 43 minutes in Eon Mode with write depot enabled and disabled, respectively.

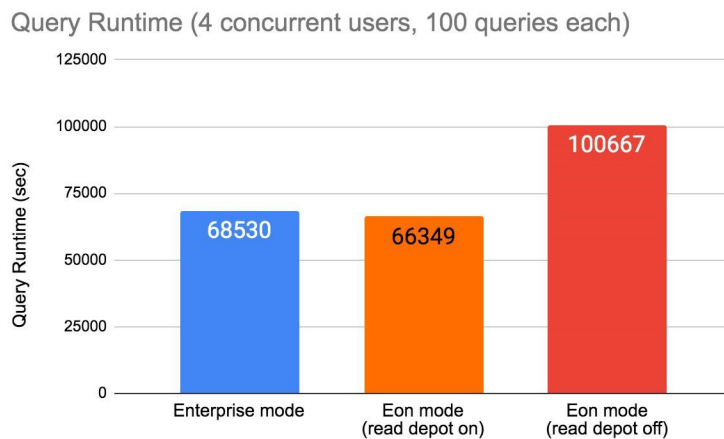


Figure 7. Database query performance – Enterprise vs. Eon Mode

Figure 7 shows database query run time on a 10TB Vertica Enterprise database and compares it to an identically sized database deployed in Eon Mode. To mimic concurrent database access, we ran the same set of queries from four concurrent user sessions. We also toggled the read depot setting in the Eon Mode tests to verify its impact on performance. The test results show that Vertica in Eon Mode with read depot enabled delivered slightly better performance than Enterprise Mode. The impact of read depot can vary significantly based on data access patterns and database size. Experiment with this setting on your databases to determine the right value for your environment.

Development, Testing, and Offload Solutions

One common challenge is having to wait many hours (or even days) to spin up a copy of the production data warehouse. What's the longest you've ever had to wait? And how much did it cost in terms of additional resources, cloud storage fees, and staff time?

Vertica in Eon mode for Pure Storage FlashBlade has an elegant solution: Starting with Purity//FB3, you can quickly create space-efficient clones of your Vertica Eon databases. This section explores how you can take advantage of FlashBlade to clone and start using very large Vertica data warehouses in minutes. These data warehouse clones are useful for developing and testing new code on the latest data. They can be used to test out transformations to schemas or data on the contents of the real database without any risk to production. They can also be used to transform a disaster recovery (DR) copy of the database into an active copy that can be used to offload additional queries to a secondary location. All without worrying about potential changes that will invalidate the DR copy.

Solution Overview

Let's take a deeper look at each of these use cases:

- **Query Offload:** Many organizations have invested in maintaining DR or secondary copies of their data warehouse. As the insights gathered become more valuable to the business, it's smart to ensure fast recovery and business continuity for the data warehouse. But having another copy of the database and a second set of cluster nodes sitting idle is a waste of



resources. And the organization also needs to be confident that its DR database is healthy. Using a clone of the DR database can safely allow both at the secondary site.

- **Developer and Data Scientist Copies of DB:** Developers and data scientists can be much more productive when they have access to a dev copy of the database holding recent, realistic data and schemas. A copy of the database that they don't have to worry about messing up; a copy where they can try running operations that modify the database as well as query it. Ideally, each user can get a copy. Since FlashBlade object clones are space-efficient and Vertica typically writes each object only once, this now becomes possible.
- **Test Copies of DB:** Today's technically mature organizations rely on automated testing, continuous integration, and continuous deployment. Testing is best done with fresh, realistic data and schemas. Because FlashBlade object clones are quick, it's possible to efficiently clone databases holding hundreds of terabytes in minutes. This fits within the time constraints of most automated testing and CI-CD frameworks and unlocks potentially higher quality results in less time.

Fast CopyObject Capabilities

FlashBlade can use incredibly fast metadata operations to make a space-efficient copy of all the objects in a Vertica Eon mode database. When the FlashBlade gets an Amazon S3 API call to PUT an object as a copy of an existing object within the same bucket, it understands that it can take a more optimal approach. Instead of copying everything associated with the source object into a new location, it can instead quickly create a new internal metadata entry that points to the data in the existing source object. The metadata copy is nearly instantaneous and has the advantage of using almost no additional space on the FlashBlade. These objects can also be overwritten by new objects, so the database copy supports updates (PUTs) as well as queries (GETs).

Dataset	Copy	s5cmd cp	s5cmd rm	s5cmd du		FlashBlade GUI			
		Seconds	Seconds	Objects	Bytes	Objects	Physical	Logical	DRR
SF=50000	0			44590	18.4T	44.59K	17.12T	18.40T	1.1
SF=50000	1	4.54	6.50	89180	36.7T	89.18K	17.12T	36.79T	2.1
SF=50000	2	4.83	6.54	133770	55.1T	133.77K	17.12T	55.18T	3.2
SF=50000	3	5.23	6.56	178360	73.4T	178.36K	17.12T	73.57T	4.3
SF=50000	4	5.28	6.65	222950	91.8T	222.95K	17.12T	93.27T	5.4
SF=50000	5	5.40	6.63	267540	110.2T	267.54K	17.12T	110.4T	6.4
SF=50000	6	5.32	6.67	312130	128.5T	312.13K	17.12T	128.52T	7.5
SF=50000	7	5.55	6.69	356720	146.9T	356.72K	17.12T	146.88T	8.6
SF=50000	8	5.03	6.77	401310	165.2T	401.31K	17.12T	165.23T	9.7
SF=50000	9	5.41	6.68	445900	183.6T	445.90K	17.12T	183.59T	10.7
SF=50000	10	5.47	6.63	490490	202.0T	490.49K	17.12T	201.95T	11.8
Average		5.20	6.63						



Table 3. Object operation time and data efficiency

We started with 50TB of raw data, which was compressed to about 18.4TB of logical storage using Vertica's advanced columnar compression. This was stored in 44,590 objects on the FlashBlade, which further compressed it to about 17.12TB of physical storage. That works out to a data reduction ratio (DRR) of about 1.074:1. In the experiment, we created 10 copies of the database. Each took an average of about 5.20 seconds to create, which is an effective copy rate averaging about 3.54TB/s in terms of logical storage and about 9.62TB/s in terms of the original raw data. This highlights the speed at which FlashBlade can clone Vertica databases.

The second factor to notice is that the physical storage used on the FlashBlade stayed steady at 17.12TB. With additional copies of the database, the object count, logical capacity, and DRR increased in predictable increments. This highlights the efficiency of FlashBlade, allowing many working copies of the same database for offload, development, and testing.

The final factor is the speed of cleaning up these database clones. If they're to be used for automated testing and validation, we need to be able to quickly get rid of cloned databases and move on to the next step. Removing all the cloned database objects took an average of 6.63 seconds. This translates to an effective rate of about 2.78TB/s logical storage (7.54TB/s raw data). At these rates, even petabyte-scale databases can be cloned and cleaned up in minutes.

Creating a Fast Crash-consistent Copy of DB

Creating a crash-consistent copy of a Vertica Eon database is as simple as copying all the underlying objects. It helps if the database is rooted in a specific prefix rather than the base of the bucket. As a best practice, configure the communal storage path for the database to be like `s3://vertica/prod/dw01` instead of just `s3://vertica/dw01`. This makes it simple to copy it into another prefix (e.g., `s3://vertica/test23/dw01`) within the same bucket.

There are several tools available for performing storage operations on AWS buckets. The tool we use most often is [s5cmd](#). It's [very fast](#) and provides a flexible syntax that makes it possible to perform [advanced tasks using simple scripts](#). Using `s5cmd`, the command to clone a Vertica database looks like this:

```
s5cmd --endpoint-url=10.1.2.3 --log=error cp 's3://vertica/prod/dw01/*' s3://vertica/test23/dw01/
```

You also need to define the credentials for the FlashBlade bucket in `$HOME/.aws/credentials`. The single quotes around the source path are to ensure the shell doesn't try to resolve the wildcard before passing the parameters to `s5cmd`. The default logging level is `info`, which will generate a line of status for every object copied. Setting `--log=error` suppresses those messages. To clean up the copy after it's no longer needed, the command looks like:

```
s5cmd --endpoint-url=10.1.2.3 --log=error rm 's3://vertica/test23/dw01/*'
```

Ensuring Database Consistency

The Vertica Eon database in S3 communal storage should always be crash-consistent and able to revive from some previous epoch. However, it may be useful to synchronize the catalog and write out recent changes to communal storage prior to cloning the database. To do that, use the following commands:



```
vsq1 -qt -c "SELECT hurry_service('System','TxnLogSyncTask'); SELECT sync_catalog();"
vsq1 -qt -c "SELECT count(*) from system_services where service_name='TxnLogSyncTask' and
last_run_end is null;" | sed -e '/^$/d' -e 's/ //g'
```

The result of the last command should be 0 (zero), showing that the sync tasks were completed across all nodes.

Caveats for Fast CopyObject

FlashBlade optimizes the AWS S3 `CopyObject` API whenever it's used to request an object copy in the same bucket as the original object. Full copies only result when copying objects across buckets, when copying objects using multipart-copy (via the `UploadPartCopy` API), or when uploading a copy of the object from a host. FlashBlade does not have the same 5GB object size restrictions for `CopyObject` as AWS. We can copy objects of any supported size (currently up to 64TB) using a single `CopyObject` API call.

The `s5cmd cp` command includes `--concurrency (-c)` and `--part-size (-p)` parameters that can be used to help ensure the use of the `CopyObject` API. If concerned about object sizes, you can modify the command used to clone the database prefix by setting `--concurrency` to 1, and `--part-size` to something very large like 67108864 (64TB):

```
s5cmd --endpoint-url=10.1.2.3 --log=error cp --concurrency=1 --part-size=67108864
's3://vertica/prod/dw01/*' s3://vertica/test23/dw01/
```

Different tools for working with Amazon S3 buckets will likely have similar options, so check the usage for the tool of your choice. You can also set the [multipart_threshold parameter](#) within the `~/.aws/config` file. This should be picked up by any tools that use the AWS S3 SDKs.

Bringing the Database Copy Online

To bring the copied database online, simply revive it on another cluster and set the communal storage path to point to the copy. It's simple, but there are a few caveats:

1. The destination cluster needs to have the same number of nodes as the source cluster.
2. The copied objects will include `cluster_config.json`, which includes a cluster lease timestamp up to 15 minutes in the future. Revive command will fail until the current time is past the recorded lease time.
3. The `s5cmd` command will use many TCP connections to copy the objects. By default, many of those connections might end up getting stuck in `time_wait` state. If the client host or user isn't able to open and reuse the connections quickly enough, the copy operation will slow down and may even generate errors.
4. By default, objects larger than 5GB need to be copied using the Multipart Upload API, which currently isn't optimized. That can result in slower copies that use additional storage.

Cluster Node Count

Vertica requires a cluster to be revived using the same number of nodes indicated in the communal storage. This simplifies mapping shards to nodes and helps ensure there are sufficient resources in the revived cluster. Ideally, users have the same number of nodes in their secondary/dev/test clusters. If that's not possible, there are a few ways to meet the requirement:



- **Shrink the source cluster:** Before cloning the database, shrink the source cluster to have the same number of nodes as the destination cluster. After the cloning is complete, add the nodes back into the cluster. This is relatively simple with Vertica Eon mode since all the data is in Amazon S3 communal storage and the nodes are mostly stateless. It's even simpler if the nodes being removed are all contained within secondary subclusters that won't impact quorum. The whole process can be scheduled for a slow period and automated. However, the operation will likely take several minutes to complete. This limits the frequency with which clones can be created and refreshed. Vertica performance and SLAs can also be impacted during these operations and while the node count is reduced.
- **Virtualize the destination cluster:** Usually, the challenge to having an equal number of nodes at the destination cluster is simply cost. It's too expensive to purchase and operate the same number of servers at a secondary location. While Vertica requires the same number of nodes, those nodes do not need to be equally powerful. It should be acceptable to virtualize a smaller collection of physical servers into a sufficient number of Vertica nodes. There will be an impact on performance compared to the source cluster and likely virtualization and management overhead. But it's often a viable option when the primary challenge is the cost and reduced performance at the secondary site is acceptable.

Cluster Lease Timeout

Vertica keeps the cluster configuration in a communal storage metadata object (`cluster_config.json`). This object has all the required information to define and revive the cluster. One detail tracked is the *Cluster Lease Expiration*. Each Vertica database in communal storage is "leased" to a particular cluster for some time in the future. This is a protection mechanism that keeps more than one cluster from using the same database. The object is periodically updated with the latest state, and the lease is extended for another period (e.g., 15 minutes by default in Vertica 10.0). If something happens to the cluster, the lease will eventually expire, and the database can be revived on another cluster.

The `cluster_config.json` object will be cloned along with all the other database objects, and the original cluster lease will prevent the cloned database from being revived until the lease expires. Usually, this isn't a problem. If the clone is used for a database backup or DR, it likely won't be revived until the lease is already expired. But if the purpose of cloning is to rapidly run some automated tests, this will slow down the process and limit how many tests can be done per hour.

The simplest solution is to stop the database before cloning it and start it again once the clone operation completes. When the database is stopped, the lease time is updated with the current time and the database can be revived immediately. If the database can be stopped with a minor impact to operations, this is the preferred and supported method. If making multiple clones, the database only needs to be stopped for the first clone. Additional clones can be made by cloning the first clone.

If stopping the database isn't practical, one work-around is to use a script to rewrite the cluster lease timestamp in the clone path. For example, the following simple awk code will take the original object as input and write a version with the modified timestamp to `stdout`:

```
#!/bin/awk -f
{
    switch ($0) {
        case / "ClusterLeaseExpiration" /:
            print strftime("  \"ClusterLeaseExpiration\" : \"%F %T.000000\",", systime(), 1)
            break
        default:
            print $0
    }
}
```



```
}
}
```

If the code is in an executable file named `grab_lease_now` (in the path), then you can use it like:

```
[dbadmin@node01]$ s5cmd --endpoint-url=10.1.2.3 cat
s3://vertica/test23/dw01/metadata/verticadb/cluster_config.json | grab_lease_now > /tmp/cfg_new
[dbadmin@node01]$ s5cmd --endpoint-url=10.1.2.3 cp /tmp/cfg_new
s3://vertica/test23/dw01/metadata/verticadb/cluster_config.json
```

The cloned database can then be revived immediately. Something like the above would need to be included as part of the test automation. In our experiments with a 50TB Vertica Eon database, we used an Ansible playbook to clone the database, revive it on another cluster, start it, stop it, and drop it. The entire cycle took only several minutes, with most of the time spent on database operations rather than replication and cloning.

Sufficient TCP/IP Connections

Amazon S3 operations are essentially REST API calls over HTTP. Many of these calls require opening and closing a TCP/IP connection for each operation. This happens very quickly, but by default, a closed TCP/IP connection will go into a `time_wait` state to make sure any delayed packets are matched to the right connection and handled properly. The default `time_wait` interval varies, but it's usually at least 60 seconds. This tends to be too conservative for modern data center networks. Each connection also requires a port on the host, and there's a finite number of ports available. When trying to clone databases with several hundred thousand objects, it's possible that all ports would have been used at least once and be stuck in the `time_wait` state. If that happens, the cloning speed will be limited by the available ports how quickly they are released.

Fortunately, modern operating systems allow the various relevant parameters to be tuned. The following tunings worked well for our testing using Centos 7.7:

```
net.ipv4.tcp_fin_timeout=20
net.ipv4.tcp_tw_reuse=1
net.ipv4.ip_local_port_range=16384 65535
```

These are typically added to the `/etc/sysctl.conf` file and then enabled by running `sysctl -p` (both as `sudo` or `root`). The `net.ipv4.tcp_fin_timeout` setting lowers the `time_wait` period to 20 seconds. The `net.ipv4.tcp_tw_reuse` setting allows connections still stuck in `time_wait` state to be reused by new connections that need them. Finally, the `net.ipv4.ip_local_port_range` setting expands the number of available ports by lowering the starting port to 16384. You may need to experiment with your particular OS and network to find the optimal settings.

These tunings should be *applied to the host/VM used to clone databases* in the S3 bucket. They don't need to be applied to all the Vertica nodes, although it's likely safe to do so. Just be aware if you're operating across a WAN with occasionally severe congestion, QoS throttles, or non-trivial packet loss. In those cases, the parameters may need to be left at the default or evaluated more carefully.



Using the CopyObject API for Objects Larger than 5GB

AWS S3 supports CopyObject API calls for objects that are 5GB or smaller and requires Multipart Copy for larger objects. FlashBlade supports object sizes up to 64TB and does not have the same CopyObject API size limit. Various S3 tools may assume the size limit and use the non-optimized API.

Fortunately, the multipart size threshold can be configured in the ~/.aws/config file. Add the following values to the profiles for the FlashBlades used with Vertica:

```
[default]
s3 =
    multipart_threshold = 64TB
```

This should ensure that operations on the FlashBlade use the larger object limits. As of v1.2.1, s5cmd doesn't require this tunable to use the CopyObject API with larger objects.

Disaster Recovery and Backup Solutions

Continuity of operations in the face of problems is part of any organization's operational maturity. Traditionally, data warehouses were not a top priority for disaster recovery. But organizations have evolved to drive more operational decisions based on analytics and metrics. Having access to data warehouses and analytical databases are increasingly mission-critical.

Vertica includes powerful native tools (e.g., VBR and copy cluster) for backing up, restoring, and copying databases and clusters. Vertica in Eon mode with Pure Storage FlashBlade includes several additional options and optimizations. These leverage FlashBlade's native replication capabilities and performance. FlashBlade can natively replicate Amazon S3 buckets used as communal storage between two FlashBlades. It can also replicate the buckets to AWS S3. Any updates to databases in the buckets are continuously propagated from the source to the target bucket. FlashBlade can also write backup streams at many gigabytes per second and restore data even faster. That makes it an ideal backup target for VBR.

The rest of this section explores DR and backup solutions that combine FlashBlade capabilities with Vertica in Eon mode. We will first review the core capabilities of FlashBlade Object Replication and using FlashBlade with VBR. Then, we'll explore using FlashBlade as part of a DR solution leveraging crash-consistent copies of the Vertica database. Finally, we'll show how to combine VBR with FlashBlade to manage consistent, point-in-time backups at local and remote sites.

FlashBlade Object Replication

With Purity//FB 3, FlashBlade now offers [asynchronous object replication](#), which enhances its fast-object capabilities and leverages fast-object potential along with cloud benefits. Users configure replication links between source buckets on a FlashBlade, and destination buckets either on another FlashBlade or an AWS S3 bucket in the cloud.



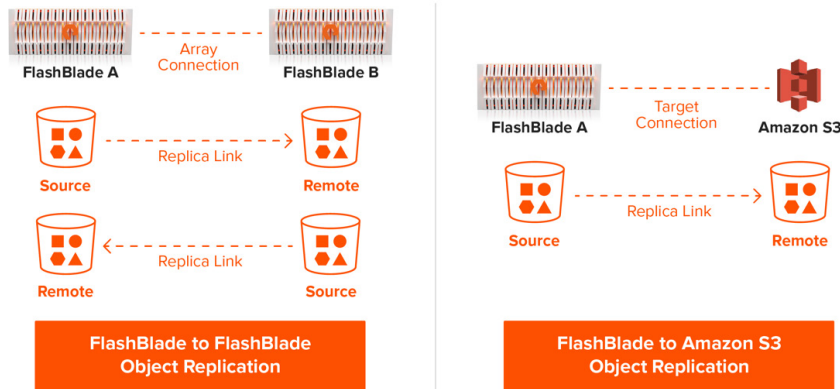


Figure 8. FlashBlade Object Replication options

Once the link is established, objects written to the source bucket are quickly and continuously replicated to the destination bucket. This allows the data to be continuously protected, span multiple locations to survive disasters, and be securely made available in the cloud. Like all FlashBlade features, this capability is part of the product, simple to manage, and doesn't require licenses or special gateways. Also, FlashBlade replication is integrated with Pure1®, our AI-driven cloud-based management platform which delivers simple fleet-wide monitoring and management.

FlashBlade as VBR Target and Source

Vertica includes a powerful utility called VBR for managing backup and restore tasks. In the latest version of Vertica (10.0), VBR allows users to:

- Create a full backup of a Vertica database.
- Restore a complete database or individual objects from backup.
- Back up specific objects (like schemas and tables) from a database (Enterprise Mode only).
- Copy a database to another cluster. For example, to promote a test cluster to production (Enterprise Mode only).
- Replicate individual objects to another cluster (*Enterprise Mode only*).

List available backups.

The VBR utility supports backing up and restoring both Eon and Enterprise mode databases to Amazon S3 locations.

FlashBlade is an ideal [backup and restore](#) target and works smoothly with VBR. [Configuration is simple](#). First set a number of environment variables in the shell running the VBR utility that define endpoint and credentials for the communal storage source and backup destination:

```
export VBR_COMMUNAL_STORAGE_ENDPOINT_URL=http://snfb-17-rr:80/
export VBR_COMMUNAL_STORAGE_ACCESS_KEY_ID="PS...JP"
export VBR_COMMUNAL_STORAGE_SECRET_ACCESS_KEY="84...b2/e8...DF"
export VBR_BACKUP_STORAGE_ENDPOINT_URL="http://snfb-11-rr:80/"
export VBR_BACKUP_STORAGE_ACCESS_KEY_ID="PS...EC"
export VBR_BACKUP_STORAGE_SECRET_ACCESS_KEY="0B...c0/af...CK"
```



Next, edit the VBR configuration .ini file to include the path to the Amazon S3 backup location and a local node path to hold locks during the backup operations:

```
s3_backup_path = s3://sfo-bkup/prod/dw02-1T/
s3_backup_file_system_path = [ ]:/home/dbadmin/backup_locks_dir/
```

Finally, use the VBR utility to initialize the Amazon S3 location before the first backup. Then start the backup process:

```
vbr -t init -c eon_backup_restore.ini --s3-force-init
vbr -t backup -c eon_backup_restore.ini
```

We will explore VBR as part of the data protection workflow for Vertica in Eon mode for Pure FlashBlade later in the paper.

FlashBlade Replication for Disaster Recovery

Vertica Eon mode simplifies DR from crash-consistent copies of the database in Amazon S3 storage. Most of the objects (data and catalog) are written only once and never updated. Older objects are rarely updated. Newer objects may be consolidated (merged), but those operations don't impact the consistency of the database. And the `revive_db` operation will automatically move back in time to find the most recent epoch where the database is in a consistent state. With these architectural features and Vertica defaults, it takes little effort to deliver a recovery-point objective (RPO) below 30 minutes. Let's take a look at what this can look like when combined with FlashBlade capabilities.

Local Crash-Consistent Disaster Recovery Copy

Sometimes, it's useful to have a local DR copy of the database that can be used for rapid recovery in case the database is somehow corrupted. With FlashBlade, we can use the same fast, space-efficient cloning technique to create local DR copies. Since the copies are space-efficient, only objects that get deleted in the original database path will take up space in the DR copies. If the DR copies are periodically recycled, those deleted objects will eventually be purged from the FlashBlade. The database can be quickly recovered using the `revive_db` command and any missing data reloaded from original sources. Creating the clone is a simple `s5cmd` command:

```
s5cmd --endpoint-url=10.1.2.3 --log=error cp 's3://vertica/prod/dw01/*'
s3://vertica/snapshot/20200801142419/dw01/
```

Also consider using the additional `vsq1` commands described in the [Ensuring Database Consistency](#) section prior to cloning the database. When the clone is no longer needed for DR, just delete the objects using `s5cmd`:

```
s5cmd --endpoint-url=10.1.2.3 --log=error rm 's3://vertica/snapshot/20200801142419/dw01/*'
```

These operations can be combined in a simple `cron` script that maintains the desired number of clones according to whatever schedule is optimal. Follow these steps if the database needs to be recovered:



1. Stop and drop the current (corrupted) database
2. Copy the objects from the DR clone path back to the production database path
3. Revive the database from the production database path
4. Reload any data that had been loaded since the revived database epoch

Depending on the database size and change rate, this workflow may be significantly faster than alternatives for local DR.

Remote Crash-Consistent Disaster Recovery Copy

Local copies of the database are great for a subset of disaster recovery scenarios. But what about real site-level disasters? It doesn't have to be a fire, flood, or major earthquake. Even a regional power or network outage that lasts for hours might have a disastrous impact on operations. For those scenarios, the only option is having a ready copy of the infrastructure and database at a remote site.

This is where FlashBlade object replication comes in. Administrators can protect their Vertica databases from disaster by setting up a FlashBlade object replication relationship with another bucket at a secondary site. Figure 9 shows the replication of a bucket called "bos" to a bucket called "bos-dr" on a FlashBlade in another region. Once the replication relationship is established, any changes to the source bucket will be replicated to the destination bucket.

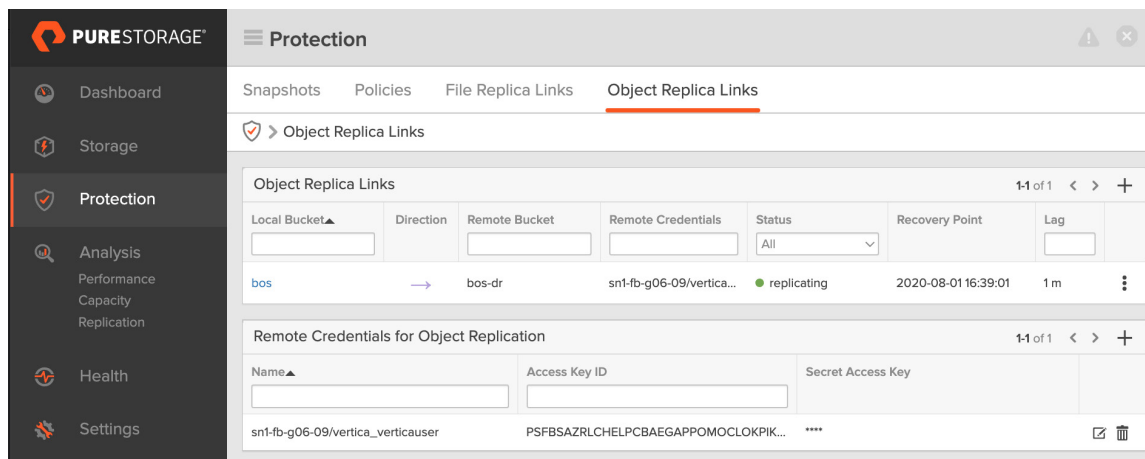


Figure 9. FlashBlade Object Replica Links

Replication is continuous: New objects are detected in seconds and scheduled for replication. If the link is idle, the objects will be replicated right away. The replication rate is as quick as the network between FlashBlades can support and the destination FlashBlade can write. It's common to see deployments transfer data at multiple gigabytes per second (Figure 10).



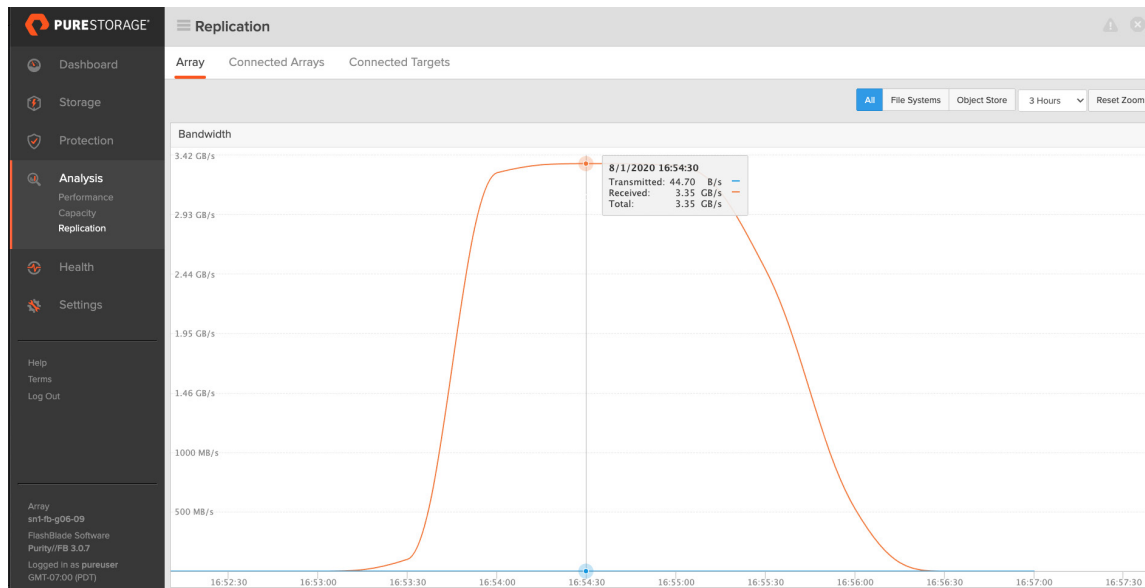


Figure 10. Example of Object Replication write bandwidth at the destination FlashBlade

Baseline Replication to the DR Bucket

The replication link will only transfer new and modified objects after it has been established. Ideally, the replication link is established before the database is created and loaded. But if an existing database needs to be replicated, Purity//FB3.0 requires a manual *baseline* replication. This can be done using this seven-step process:

1. Get the object count and size of the data in the source bucket that needs to be baselined:


```
s5cmd --endpoint-url=10.1.2.3 du -H 's3://vertica/*'
```
2. Clone the contents of the bucket into a special temporary prefix within the source bucket. (We suggest using the "!" character as the temporary prefix name since it has a low ASCII value that will be at the top of any output. But you need to quote it so that the shell won't try to interpret it as a reference to previous commands.):


```
s5cmd --endpoint-url=10.1.2.3 --log=error cp 's3://vertica/*' 's3://vertica/!/'
```
3. Replication will detect the object clones as new and replicate them to the target bucket. Wait for the replication to complete. You can use the s5cmd du command to check that the object count and sizes are identical at both sites:


```
s5cmd --endpoint-url=10.4.5.6 du -H 's3://vertica-dr/!/*'
```
4. Clone objects in the temporary prefix within the target bucket back to the root of the bucket. Use the --no-clobber option for s5cmd cp to prevent any older versions of the objects from overwriting newer versions:


```
s5cmd --endpoint-url=10.4.5.6 --log=error cp --no-clobber 's3://vertica-dr/!/*' s3://vertica-dr/
```
5. Delete the temporary prefix in the source bucket:


```
s5cmd --endpoint-url=10.1.2.3 rm 's3://vertica/!/*'
```
6. Delete the temporary prefix in the target bucket:


```
s5cmd --endpoint-url=10.4.5.6 rm 's3://vertica-dr/!/*'
```
7. Validate that the two buckets are now nearly identical (there may be changes to the source that haven't yet propagated to the target):


```
s5cmd --endpoint-url=10.1.2.3 du -H 's3://vertica/*'
```



```
s5cmd --endpoint-url=10.4.5.6 du -H 's3://vertica-dr/*'
```

Pure Support is also able to help with baseline replication for buckets that already contain data.

Disaster Recovery Operations at the Secondary Site

Once the database objects have been successfully replicated to the secondary site, how can we use them? The answer again takes advantage of Fast CopyObject. If you're most familiar with snapshot-based filesystem replication, there's an important difference: the destination bucket is writable. We want to avoid making any changes to the bucket prefix where the production database is sending updates. But we can easily clone that prefix into another prefix within the same bucket. Then use the database copy in that clone prefix for our operations. There are three scenarios where this comes in handy:

- Validating that the replicated copy of the database is consistent and can be used for disaster recovery
- Putting the DR copy of the database into production at the secondary site in case of disaster

Using the DR copy of the database to offload some ongoing operations to the secondary site

Verifying DR Database Consistency

This follows the same workflow as described in the previous sections:

1. Hurry the transaction log sync service and sync the catalog on the source database side
2. Wait for the replication lag between the source and destination FlashBlades to replicate the flushed data
3. Clone the DR database to a temporary prefix path (s5cmd cp)
4. Revive the database from the temporary path (admintools -t revive_db)
5. Start the database (admintools -t start_db)
6. Perform some sample queries to validate that it's working correctly (vsqll)
7. Stop the database (admintools -t stop_db)
8. Drop the database (admintools -t drop_db)
9. Remove the cloned objects in the temporary prefix path (s5cmd rm)

Putting the DR Database into Production

This workflow starts similarly to the Verification workflow but then shifts to indefinite operation:

1. Prevent updates from the source FlashBlade by disabling the destination bucket access key used for replication
2. Clone the DR database to an Activated prefix path (s5cmd cp)
3. Revive the database from the Activated path (admintools -t revive_db)
4. Start the database (admintools -t start_db)
5. Perform some sample queries to validate that it's working correctly (vsqll)
6. Redirect load balancers and DNS servers to the secondary site
7. Operate the main production from the secondary site until ready to fail back to the primary site



Using the DR Copy to Offload Operations

As mentioned previously, one powerful way to use all the infrastructure at the secondary site is to offload some of the primary site operations. For example, using the secondary site for development and QA during normal operations, or running less critical queries and reports that don't require up-to-the-second updates.

The workflows are similar. The one exception is if you'd like multiple independent copies of the database for different engineers. In that case, first clone the DR copy of the database once into a template prefix. The template can then be modified as needed (e.g., anonymized or redacted). Finally, the template can be cloned repeatedly to create multiple independent copies of the database.

Failing Back to the Primary Site

Once the disaster is remediated, operations at the primary site can be resumed by reversing the replication relationship and copying the latest version of the database from the secondary to the primary site. This approach is simple and reliable. As of Purity//FB 3.0, this is the main supported failback mechanism.

The downside is that there may be a huge amount of data to replicate. If the disaster wasn't destructive and most of that data still exists at the primary site, there may be more efficient ways to resume operation at the primary site. The idea would be to identify any objects that have been created or overwritten since the failover occurred and then copy them to the correct paths at the primary site. If the DR path was cloned before reviving, the secondary site would have a local record of which objects existed at the time of the failover. However, it's beyond the scope of this paper to detail how to implement this approach.

Vertica VBR Backup and Restore

As mentioned in the *FlashBlade as VBR Target and Source* section above, the VBR utility provides a powerful mechanism for backing up and restoring Vertica databases. It's capable of using Amazon S3 endpoints as the backup destination and source. It also works with both Enterprise and Eon mode databases. Combined with FlashBlade, it enables accelerated workflows to protect and recover Vertica databases.

Most of the guidelines and procedures are either covered in the [Vertica documentation](#) or the section above. In the rest of this section, we will simply present potential ways to employ VBR as part of a Vertica in Eon mode for Pure Storage FlashBlade deployment.

Local VBR Backup

The simplest backup option is to have a VBR backup of the database on the local FlashBlade. This makes it easy to restore specific Vertica objects (like tables and schemas) if they're corrupted and a rollback of the database doesn't make sense. We suggest creating a separate bucket for the VBR backups. That bucket can then be replicated to a FlashBlade in another region as part of a DR strategy. It can also be replicated to AWS S3 as part of a more comprehensive backup and archiving strategy. Local VBR copies have the advantage that:

- Local backups are not dependent on WAN bandwidth and can often be completed more quickly.
- VBR Snapshots only back up information that has changed since the last backup.
- Completed backups are always consistent and provide for object-level recovery.
- When the VBR destination bucket is the source of a FlashBlade replication, replication can be somewhat more efficient. The bucket is replicated in a consistent form and doesn't replicate temporary objects that will later be deleted or merged.



We tested VBR backup and restore rates using a four-node cluster and a 1TB database (Table 4). The FlashBlade was relatively idle, as were the cluster nodes. Most of the time was spent in process synchronization within the VBR utility. The default setting of 10 streams produced the best performance for both backup and restore.

Streams	Duration (seconds)		Throughput			
	Backup	Restore	Backup MB/s	Backup Obj/s	Restore MB/s	Restore Obj/sec
1	1126.5	1104.2	331.2	34.9	337.9	35.6
4	647.4	609	576.3	60.7	612.6	64.6
10 (default)	614.4	601.1	607.2	64.0	620.7	65.4
64	629.7	615	592.5	62.4	606.6	63.9
128	640	623.4	582.9	61.4	598.5	63.1

Table 4. VBR Backup and Restore Rates

As mentioned previously, an alternative to local VBR backups is to use FlashBlade Fast CopyObject to quickly create a space-efficient clone of the database. This can often be both faster and more space-efficient than an incremental VBR backup. If large volumes of data need to be restored, reviving the database from one of the historical clone paths may also be faster. However, there are a few downsides:

- There is no simple mechanism to recover specific Vertica objects; it's effectively the whole database or nothing
- The crash-consistent clone of the database may need to be further rolled back to find a consistent epoch
- The copy would need to be in the same bucket as the database, complicating replication. Replicated objects are not space-efficient and will use both bandwidth and capacity at the destination.

Consider the pluses and minuses of both approaches when selecting a data protection strategy.

Remote VBR Backup

Storing the VBR backup in a remote location can be part of an overall data protection and disaster recovery strategy. We previously mentioned two options: replicating a local FlashBlade VBR bucket to either another FlashBlade or AWS S3. A third option is simply to specify a remote FlashBlade array as the target in the VBR .ini configuration file:

- The configuration process is identical to using a local FlashBlade
- The network between sites needs to be able to route HTTP/HTTPS requests to the remote location
- QoS bandwidth limits should accommodate backup windows and data change rates
- If HTTP proxies exist on the network, you may need to whitelist the FlashBlade to avoid the proxies

Keep in mind Vertica's requirements for restoring a database from the VBR backup. If the goal is to restore at a remote site (especially quickly in case of a disaster), you may need to develop some automation to stand up the destination database and cluster to restore into.



Vertica Data Protection and Offload Best Practices

- Create your Eon mode databases using additional prefix paths in the bucket (e.g., s3://vertica/prod/dw01/ instead of s3://vertica/dw01/). This makes it simple to use Fast CopyObject to clone the database into additional prefixes (e.g., test, dev, etc.).
- Use separate buckets for VBR backup and replication targets (e.g., s3://bos-dr/prod/dw01/ or s3://oak-bkup/prod/dw02/). This makes it simpler to replicate just the information you need or clone the databases in the replica bucket.
- Create a separate user for replication in the replication target account and use that user's keys to set up the replication relationship. In case of disaster, you can (temporarily) disable the key at the destination site to prevent updates from the primary site. This is important in case the source FlashBlade can still intermittently communicate with the destination site.

Example Automation

We've mentioned automating many of these operations using Ansible throughout this paper. We've created a [Vertica repository](#) in the [PureStorage-OpenConnect space](#) on GitHub with example Ansible playbooks that you can clone, modify, or simply examine for code snippets and ideas. The playbooks also automate interactions with the FlashBlade, which can be helpful for fully automating your Vertica Eon mode infrastructure.

There are two main subsections:

1. Vertica PoC ([vertica-poc](#)) — code for quickly building out Vertica PoC environments for demos and tests. Primarily consists of Ansible playbooks and some setup bash scripts.
2. Vertica Cloning ([vertica-cloning](#)) — example Ansible playbooks with code related to cloning Vertica databases for Test/Dev and DR validation scenarios.

See the detailed README file within each subsection for how to use the code.

Summary

The Vertica Analytics Platform is a powerful tool for high-performance, large-scale SQL analytics. Eon Mode is a new storage deployment model for Vertica databases leveraging shared fast-object storage like FlashBlade storage. Vertica in Eon Mode significantly improves operational efficiency while maintaining (if not improving) database performance. Deploying and running Vertica in Eon Mode databases on Pure FlashBlade systems provides you with cloud-like simplicity and flexibility, but in a more secure private cloud environment of your own with predictable costs.

Vertica includes powerful native tools (e.g., VBR and copy cluster) for backing up, restoring, and copying databases and clusters. Vertica in Eon mode with Pure Storage FlashBlade includes additional options and optimizations. Starting with Purity//FB3, you can replicate FlashBlade Amazon S3 buckets to another FlashBlade or AWS S3. Any updates to databases in the buckets are continuously propagated from the source to the target bucket. FlashBlade can also write backup streams at many gigabytes per second and restore data even faster. That makes it an ideal backup target for VBR.

You can also quickly create space-efficient clones of Petabyte-scale Vertica Eon databases in minutes. These data warehouse clones are useful for developing and testing new code on the latest data. They can be used to test out transformations to schemas or data on the contents of the real database but without any risk to production. They can also be used to transform a



DR copy of the database into an active copy that you can use to offload additional queries to a secondary location. All without worrying about potential changes that will invalidate the DR copy.

Additional Resources

- [Vertica Documentation](#)
- [Solution Brief: Pure Storage and Vertica](#)
- [Pure Storage and Vertica Partnership](#)
- [Blog Post: "Why Modern Analytics Need a Modern Infrastructure"](#)



Appendix: Best Practices for Deploying Vertica with FlashBlade Storage

We recommend the following Best Practices to get the benefits we discussed and achieve a successful Vertica Eon mode for Pure Storage FlashBlade implementation:

1. Use prefix naming in the buckets to simplify working with Fast CopyObject. If you have an existing database at the root of the bucket, consider cloning the stopped database into a prefix and reviving from the new path.

2. Disable AWS Streaming since it's not needed with FlashBlade (via `vsq1` prompt):

```
select set_config_parameter('AWSStreamingConnectionPercentage', 0);
```

3. Keep local depot enabled for reads (the default). If you have a large data set with a working set that doesn't fit in local cache, turning off Depot for Reads *may* result in similar performance with simpler operation. It's recommended that at some point customers experiment with the read depot setting to determine an ideal value for their deployment. Via `vsq1` prompt:

```
select set_config_parameter('UseDepotForReads', 1);
```

4. Disable local depot for write as it improves database ETL operations (via `vsq1` prompt):

```
select set_config_parameter('UseDepotForWrites', 0);
```

5. Adjust per-server TCP connection on Vertica via VSQ1 prompt to 128 connections (via `vsq1` prompt):

```
select set_config_parameter('AWSConnectionPoolSize', 128);
```

©2021 Pure Storage, the Pure P Logo, and the marks on the Pure Trademark List at <https://www.purestorage.com/legal/productenduserinfo.html> are trademarks of Pure Storage, Inc. Other names are trademarks of their respective owners. Use of Pure Storage Products and Programs are covered by End User Agreements, IP, and other terms, available at: <https://www.purestorage.com/legal/productenduserinfo.html> and <https://www.purestorage.com/patents>

The Pure Storage products and programs described in this documentation are distributed under a license agreement restricting the use, copying, distribution, and decompilation/reverse engineering of the products. No part of this documentation may be reproduced in any form by any means without prior written authorization from Pure Storage, Inc. and its licensors, if any. Pure Storage may make improvements and/or changes in the Pure Storage products and/or the programs described in this documentation at any time without notice.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

Pure Storage, Inc.
650 Castro Street, #400
Mountain View, CA 94041