Release Notes

Product	Volt Active Data
Version	12.3.2 LTS VoltDB Operator 2.1.2 VoltDB Helm Chart 2.1.2
Release Date	September 1, 2023 (updated September 7, 2023)

This document provides information about known issues and limitations to the current release of VoltDB. If you encounter any problems not listed below, please be sure to report them to support@voltactivedata.com. Thank you.

Upgrading From Older Versions

The process for upgrading from the recent versions of VoltDB is as follows:

1. Shutdown the database, creating a final snapshot (using **voltadmin shutdown --save**).
2. Upgrade the VoltDB software.
3. Restart the database (using **voltdb start**).

For Kubernetes, see the section on "Upgrading the VoltDB Software and Helm Charts" in the *VoltDB Kubernetes Administrator's Guide*. For DR clusters, see the section on "Upgrading VoltDB Software" in the *VoltDB Administrator's Guide* for special considerations related to DR upgrades. If you are upgrading from versions before V6.8, see the section on "Upgrading Older Versions of VoltDB Manually" in the same manual.

Finally, for all customers upgrading from earlier versions of VoltDB, please be sure to read the upgrade notes for your current and subsequent releases, including *V6*, *V7*, *V8*, *V10*, and *V11*.

Changes Since the Last Release

In addition to the major enhancements listed above, users of previous versions of VoltDB should take note of the following changes that might impact their existing applications. See the *VoltDB Operator Release Notes* for changes specific to the use of VoltDB on the Kubernetes platform.

1. Release V12.3.2 (September 1, 2023, updated September 7, 2023)

1.1. Additional improvements

The following limitations in previous versions have been resolved:

- In recent versions of VoltDB (starting with V12.2) on Kubernetes, it was possible for a pod to signal it was ready before the startup of the database server process was complete. This could cause problems when performing actions such as a rolling restart where the next pod could be taken down before the rejoin of the previous pod was finished. This issue has been resolved.

- Previously, the Helm property for specifying the replication port on per-pod DR clusters (`cluster.serviceSpec.perpod.dr.replicationPort`) was ignored. This issue has been resolved.

- There was an issue where enabling per pod XDCR NodePort services but not specifying the public DR port, the cluster incorrectly advertised port 32555 for all nodes in the cluster. If, on the other hand, you explicitly identified the port with the `cluster.clusterSpec.perpod.dr.startReplicationNodePort` property, the correct ports were advertised. This issue has been resolved and the default port setting now advertises the correct ports.

- There were issues related to the DR RESET command that could, in certain edge cases, result in unexpected behavior. In some cases, clusters could not rejoin an XDCR mesh without first having the DR ID changed. Or if the rejoin operation failed, any further attempt to rejoin the cluster to the mesh would also fail. Or the cluster might inadvertently report that the rejoin was complete before the operation actually finished. These issues have been resolved.

2. Release V12.3.1 (July 26, 2023)

2.1. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2023-2976

2.2. Additional improvements

In addition, the following limitations in previous versions have been resolved:

- An issue was found regarding export. If a server crashed unexpectedly (due to failure or a forced shutdown) transactions being processed at the time may be interrupted and rolled back, leaving the database unchanged. However, if any of those transactions included both writing to an export (or topic) target and to database tables, it was possible in rare cases, due to a race condition in export handling, for an export or topic record to slip through, resulting in it being queued and sent to the target, creating an atomicity error.

  This issue has been resolved. Now, if the transaction fails, the export or topic publishing is also canceled. Note that one consequence of this correction is that the maximum time before export records are queued to the export target, which previously was solely controlled by the flush interval, now includes the time required for the transaction to be submitted, processed, and returned by the appropriate partitions and sites.

- In V12.3 of VoltDB on Kubernetes, if the Volt Management Center is configured as a service (`vmc.enabled="true"`) with TLS/SSL enabled (`vmc.service.ssl.enabled="true"`), attempting to access VMC or the JSON API would fail with an error. This issue has been resolved.

- The recent improvement in memory management and compaction for VoltDB created an issue where, if compaction was disabled, memory usage would incrementally grow and not be recovered whenever a snapshot was taken. This issue has been resolved. However, as a consequence of the fix, the improved memory management that ensures that VoltDB transactions return query results in a deterministic order does not apply *when compaction is disabled*. See the section on determinism in the V11 documentation for an explanation of how to write deterministic procedures if you plan to disable compaction.

3. Release V12.3.0 (July 5, 2023)

3.1. Long-Term Support release

The current release, Volt Active Data V12.3, is the Long-Term Support (LTS) release for version 12. We encourage all customers using V12 to upgrade to the V12.3 release to ensure ongoing support through minor

version updates. See the chapter on Long-Term Support in the *Volt Upgrade Guide* for more information on the advantages of LTS releases and subsequent chapters on how to upgrade from previous major releases.

3.2. LDAP Integration

It is now possible to use existing LDAP users and groups to authenticate and authorize access to VoltDB databases in place of builtin VoltDB accounts. LDAP integration allows users and programs to pass LDAP credentials to the database server, which in turn passes those credentials to LDAP for authentication. VoltDB then looks up what groups the LDAP user is a member of to identify what roles they have and ultimately what permissions they are granted by the database. VoltDB LDAP integration supports both plain and secure LDAP. See the chapter on Security in the *Using VoltDB* manual for details.

3.3. Dynamic schema change in XDCR clusters without pausing

When done carefully, it is possible to add and remove DR tables from a XDCR VoltDB schema without having to pause the databases. However, previously, there was no safe way to modify the tables themselves. VoltDB V12.3 introduces dynamic schema change which, as the name implies, allows you to apply selected modifications to DR tables without pausing the database and *while continuing to process transactions for those tables*.

Dynamic schema change allows you to add or remove columns from the end of a table. You can also modify the length of VARCHAR columns. During the transition, when the schema of the clusters do not match, VoltDB follows set rules for applying the binary logs to mismatched tables. In many cases — such as if applications use the default values for new columns — the clusters can process transactions normally during the transition and the clusters' contents will match once the schema change is complete.

Of course, there is always certain amount of risk to dynamic changes if you do not manage your client applications correctly, so VoltDB keeps a count of how many binary logs are applied to mismatched tables so you can determine how much risk there is. New data columns to existing selectors and new selectors (such as DRSCHEMA and XDCR) for the @Statistics system procedure help you monitor the status of schema changes in your databases.

Since dynamic schema change does introduce some additional risk of divergence to the clusters, it is not enabled by default. Before applying schema changes without pausing, you must enable dynamic schema change in the configuration file. See the documentation on updating the schema in the chapter on Database Replication of the *Using VoltDB* manual for details. Also, if you want to use this feature in existing XDCR databases, see the section later in this document for upgrading XDCR clusters to use dynamic schema change.

3.4. Beta release of a new metrics system

VoltDB V12.3 includes the beta release of a new reporting system aimed at providing a simpler, more consistent interface to statistics about the database status and performance. The new system is built into the server (unlike the previous external Prometheus agent), meaning it starts and stops automatically as part of the overall database system. Each server reports its own status. And the system as a whole integrates seamlessly with Prometheus and Grafana

To learn more about the new metrics system, see the beta release documentation which is available online.

3.5. Support for the latest base platforms (RHEL 9 and Kubernetes 1.26)

Both Red Hat Enterprise Linux (RHEL) version 9 and Kubernetes 1.26 are now supported as platforms for production use.

3.6. Kubernetes upgrade

The VoltDB base image on Kubernetes has been upgraded to use Alpine 3.18 and Java 17.

3.7. Improved management of memory used for "undoing" transactions

VoltDB uses temporary memory to store changes in case a transaction needs to rollback. Rather than allocate and deallocate memory for each transaction, VoltDB maintains an *undo pool* that is reused for this purpose. The pool is extended if a transaction requires more undo memory. The pool then stays at the larger size assuming a future transaction is likely to need the same amount of undo memory.

Previously, the current allocation of the undo pool was not visible. A new field has been added to the @Statistics MEMORY output reporting the current allocation. In addition, the **voltadmin defrag** command now resets the undo pool as part of its memory optimization function.

3.8. New information available from @Statistics system procedure

The @Statistics system procedure returns new and enhanced data on a number of different topics:

- Memory utilization — Output from the MEMORY selector includes a new field, UNDO_POOL_SIZE, reporting the amount of memory in use for the undo pool.
- XDCR and dynamic schema change — There are new fields in the DRPRODUCER and DRCONSUMER selectors, as well as new selectors for DRSCHEMA and XDCR that help determine if dynamic schema change is allowed, if the schema are currently matching or not, and how many tuples have been received while the schema were mismatched.
- Command logs — The COMMANDLOG selector returns additional fields that help detect potential performance bottleneks, specifically around the time required to allocate and deallocate command log segments and backpressure caused by I/O latency.

See the @Statistics reference page in the *Using VoltDB* manual for more information.

3.9. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2023-2650
CVE-2022-34917

3.10. Additional improvements

In addition, the following limitations in previous versions have been resolved:

- There was an issue where TLS/SSL errors could result in the library getting caught in a loop until it ran out of memory. This issue has been resolved by upgrading the Netty version included with VoltDB.
- Previously, if a client JAR file contained additional unexpected entries, the **sqlcmd** utility would stall attempting to load information from the JAR. The utility now ignores unexpected entries, resolving this issue.
- The VoltDB Prometheus agent has a command argument (`--webserverport`) that specifies the port used to respond to data requests. The `--webserverinterface` argument has been added to allow you to specify which network interface to use as well.

4. Release V12.2.2 (April 24, 2023)

4.1. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2023-27533
CVE-2023-27534

CVE-2023-27535
CVE-2023-27536

4.2. Additional improvements

In addition, the following limitations in previous versions have been resolved:

- Using Volt Operator V2.0.0, the database pods could fail to start and the Operator report a "panic" error if TLS/SSL was configured using cert-manager. This issue has been resolved.
- In recent releases of Volt Operator (starting with V2.0.0), debug logging was turned on and the `operator.debug.enabled` ignored, so debug logging was always in effect. This issue has been resolved.
- When running a client application linked against the VoltDB Java API and using the Java 11 runtime, the application may issue warnings about an " illegal reflective access operation." The message does not impact the actual operation of the client or the database itself. You can suppress this warning using the --add-opens qualifier on the **java** command. For example:

```
$ java --add-opens=java.base/java.lang=ALL-UNNAMED  \
       --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \
       --add-opens=java.base/java.net=ALL-UNNAMED   \
       --add-opens=java.base/java.nio=ALL-UNNAME    \
       myclientapp
```

5. Release V12.2.1 (April 13, 2023)

5.1. Security update

Packages within Volt Active Data have been updated to eliminate the following known security vulnerability: CVE-2023-0464.

6. Release V12.2 (March 29, 2023)

6.1. Important! New Volt Operator for Kubernetes

The Volt Operator for Kubernetes has been updated to improve stability, extensibility, and to support Kubernetes V1.23 through V1.25. The new version of the Operator, V2.0, is required for both Volt V12.2 and Kubernetes V1.25.

The transition from V1.x to V2.0 of the Operator is the same as any other update, with one exception: when upgrading to V2.0 you must use the **kubectl replace** command to update the CRD instead of the **kubectl apply** command, as was documented in previous releases. On the other hand, use of **replace** instead of **apply** works when updating any version of the Volt software. So the documentation has been revised to recommend the use of **replace** for all software upgrades.

Because Volt Operator and Helm chart V2.0 are required for Kubernetes 1.25, customers upgrading to 1.25 must upgrade the components in the following order:

1. Update the helm repository using the **helm repo update** command
2. Update the Volt Operator custom resource configuration (CRD).
3. Upgrade the Volt Operator and Helm chart to version 2.0. If you are upgrading Volt to V12.2 you can do that at this time as well.
4. Finally, upgrade Kubernetes to V1.25.

6.2. Changes to Kubernetes pod security

In previous releases, the Volt Operator established permissions for the Kubernetes pods to provide Volt with the rights needed to function properly. This was done using Kubernetes pod security policies (PSP). Starting with the Volt Operator and charts V2.0, pod security policies are no longer enabled by default. This means the Volt pods have no specific limitations set for them.

For Kubernetes V1.23 and V1.24, you can enable PSP constraints by setting the Helm property `global.rbac.pspEnabled` to "true". However, PSP has been deprecated by Kubernetes and is no longer available in V1.25. So the preferred method for applying security constraints on pods is+ to use the built-in Kubernetes Pod Security admission controller or to use a third-party admission controller. In either case, the Volt pods require the equivalent of the *baseline* admissions to operate properly. For example, the following **kubectl** command enables the baseline permissions for the *mydatacenter* namespace:

```
$ kubectl label --override namespace mydatacenter pod-security.kuberneted.io/enforce=
```

6.3. Cloud native placement groups maximize high availability by distributing partitions across availability zones in Kubernetes

Volt Active Data now supports *cloud native placement groups*. K-Safety guarantees the database cluster can survive a certain number of node failures. Placement groups allow you to define groups of nodes that are logically connected so that Volt can avoid losing the database if an entire group goes offline. In Kubernetes, if you enable the new cloud native placement capability, Volt maximizes its availability, using knowledge of the Kubernetes regions and zones to intelligently distribute copies of the database partitions so that even if an availability zone is lost, the database can remain operational. See the section on "Configuring High Availability" in the *Volt Kubernetes Administrator's Guide* for more information.

6.4. TLS/SSL support for subject alternative names (SAN)

One feature of TLS/SSL is that the client can verify that the certificate is assigned to the same host name or address they connect to. In fact, it can allow for multiple names and even wildcards using Subject Alternative Names (SAN). By default, Volt does not utilize this feature. However, starting with Volt V12.2 the Volt Java client can enable alternative name support and verify that the certificate matches the requested address. See the note on using TLS alternative name support later in this document for details.

6.5. Required version of Python updated to V3.9 or later

Because version 3.6 is no longer supported, the minimal version of Python required by Volt Active Data has been updated to V3.9.

6.6. Kakfa security update requires new property for Volt topic producers

To resolve certain security vulnerabilities the Kafka client library has been upgraded. However, the newer library requires Kafka producers to explicitly disable idempotency when sending messages to Volt topics, which was not required before. **Important**: if you are using a Kafka producer to send data to Volt topics and do not explicitly disable idempotency, you will need to update the client to add the property `enable.idempotence` setting the property to "false" before upgrading to Volt V12.2. See the section on "Calling Topics from Consumers and Producers" in the *Using VoltDB* manual for details.

6.7. Masking passwords when configuring security on Kubernetes

When configuring security in VoltDB, you must specify usernames and passwords, which previously appeared in plain text in the YAML configuration file. You can now use the Helm **voltadmin** plugin to mask those passwords, so they are not readily viewable in the YAML. See the description of the Helm plugin **mask** command in the *Volt Kubernetes Administrator's Guide* for details.

6.8. Support for JDBC getMaxConnection method

The Volt JDBC API now supports the getMaxConnection method, which returns the integer value of zero (0) indicating the maximum is unspecified.

6.9. Ability to use separate TLS/SSL credentials for each XDCR cluster

Previously, when enabling TLS/SSL encryption for XDCR clusters in Kubernetes, all of the clusters needed to use the same certificate. Although the property `cluster.config.deployment.dr.connection.ssl` existed, it was not applied and the credentials of the local cluster were used to contact the remote clusters in the XDCR mesh.

It is now possible to specify separate TSL/SSL credentials for each XDCR cluster. When specifying the credentials of the remote XDCR cluster, use the new properties `cluster.config.deployment.dr.connection.ssl.truststore.file` and `cluster.config.deployment.dr.connection.ssl.truststore.password`, or specify a Kubernetes secret using the property `cluster.config.deployment.dr.connection.ssl.sslSecret.certSecretName` where the secret defines the keys *dr_conn_truststore_data* and *dr_conn_truststore_password*.

When upgrading an existing cluster to operator 2.0 and using the new XDCR TLS/SSL properties, be sure to remove any existing definition of the `cluster.config.deployment.dr.connection.ssl` property by setting it to an empty mapping . For example:

```
$ helm upgrade mydb voltdb/voltdb --reuse-values \
   --set cluster.config.deployment.dr.connection.ssl={}
```

6.10. Additional improvements

In addition to the new features and capabilities described above, the following limitations in previous versions have been resolved:

- The REPLICATIONROLE field of the @SystemInformation OVERVIEW selector has been changed to report the actual DR role (including XDCR roles) rather than only *replica* or *none*.
- Under normal conditions, after elastically shrinking the cluster (that is, removing nodes) the cluster saves a snapshot as a final step. If the snapshot accidentally starts before the nodes are completely removed, later attempts to shrink the cluster could fail, reporting that an elastic operation is already in progress. This issue has been resolved.
- There was a race condition where a problem pausing export connections during a schema or configuration change could result in a deadlock. This issue has been resolved.
- There was an issue in previous releases of VoltDB for Kubernetes where on OpenShift the metrics service did not start properly due to how the permissions were configured. This issue has been resolved.
- There were several issues related to running VoltDB under Java 17:
  - The web-based Volt Management Center (VMC) did not work properly, failing to load necessary resources from the server.
  - Client applications that did not explicitly include the `--add-opens` flag for the sun.nio.ch library suffered connection problems where they could not submit queries to the database.
  - The Volt loader utilities (such as csvloader and kafkaloader) had similar connection problems and did not work properly.

  All of these issues have been resolved.

- The default setting for the Kubernetes Helm property `global.createNetworkPolicy` has been changed from "true" to "false".
- Under certain rare conditions, a compaction event coincident to a multi-partition transaction could cause the transaction to stall. This issue has been resolved.
- In recent versions of Volt, attempting to create an import connector using the Kinesis importer would fail at runtime. This issue has been resolved.
- The default setting for the Helm property `global.createNetworkPolicy` has changed fro "true" to "false".
- The Helm property `cluster.clusterSpec.additionalXDCRReadiness` has been deprecated and is now ignored.

7. Release V12.1.1 (March 17, 2023)

7.1. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2022-0778
CVE-2022-1292
CVE-2021-1304
CVE-2021-3712
CVE-2022-2068
CVE-2021-23840
CVE-2022-41721
CVE-2022-41723
CVE-2022-41881

8. Release V12.1 (December 21, 2022, updated December 29, 2022)

8.1. Simplified updating of TLS/SSL certificates in Kubernetes eliminates need for cluster restart

The latest Operator release simplifies the process for updating TLS/SSL certificates and eliminates the need to stop and restart the cluster to apply the new certificate. In Kubernetes,if you store the TLS/SSL credentials using Kubernetes secrets, either created manually or through the cert-manager, the Operator now automatically updates the configuration on the running cluster once the secret changes. So, if you created the secret manually, you simply need to delete and recreate the secret with the new credentials. If you use cert-manager, cert-manager itself updates the credentials when the expiration date approaches. In either case, the Operator recognizes the change and updates the running cluster without any further operator intervention.

8.2. New @Statistics selector for information about the cluster nodes and the partitions they host

The @Statistics system procedure has a new selector, HOST, that returns information about the nodes of the cluster, including what partitions it hosts and whether it is safe to stop the node in a K-safe cluster. See the description of the @Statistics system procedure in the *Using VoltDB* manual for details.

8.3. New @SystemInformation selector for listing environment variables

Support for a new selector, ENV, has been added to the @SystemInformation system procedure. The ENV selector returns information about the environment variables defined for the database server process, one row for each variable on each host. See the description of the @SystemInformation system procedure in the *Using VoltDB* manual for details.

8.4. TLS/SSL protocols 1.0 and 1.1 are no longer supported

The TLS/SSL protocol versions 1.0 and 1.1 were deprecated and removed from most industry applications and browsers in 2020. Going forward, these obsolete protocol versions are no longer available in VoltDB either.

8.5. Additional improvements

The following limitations in previous versions have been resolved:

- The HTTP export connector has been improved to cancel all pending export messages if the connection to the export target times out. This allows the connection to be reset and the blocked requests to be resubmitted. You can set the timeout limit separately for each export target using the new `reconnect` attribute of the export `<configuration>` element in the database configuration file. You specify the reconnect limit as an integer and a single character time unit. For example, "60s" for 60 seconds or "2m" for 2 minutes.
- Changes intended to improve the management of TLS/SSL in V12.0, inadvertently removed support for certain encryption algorithms, such ECDSA. This issue has been resolved.
- Several corrections have been made to the data reported by the @Statistics system procedure. Specifically:
  - At times the @Statistics MEMORY selector would under report the tuple data memory. This issue has been resolved.
  - The @Statistics INITIATOR selector unintentionally included import processing as part of the initiator statistics, producing misleading results. This error has been corrected.
- There was a minor memory leak associated with statistics triggered by ad hoc queries. Although normally not sufficient to even be noticed, constant and very frequent ad hoc queries (for example thousands an hour for days) each creating a separate connection could eventually cause excessive memory usage, slowing down the database and, in extreme cases, ultimately blocking further transactions.This issue has been resolved.

9. Release V12.0 (October 18, 2022)

9.1. Memory management improvements

Memory management for the database table data has been rewritten and optimized to reduce certain impediments in the previous implementation. Although this change is primarily internal and transparent to you as a customer, it does have two direct benefits in terms of eliminating development and operational roadblocks:

- **No Large Compaction Events** — defragmentation of data storage is now performed incrementally on a per table and per partition basis. Since the compaction transactions are much smaller, and also partitioned, they have little or no impact on the ongoing business workload. In addition, you have control over how large those periodic compaction events are and and how often they occur. See the chapter on memory management in the *Volt Performance and Customization Guide* for more information.
- **No Hash Mismatches Due to Row Order** — All copies of a partition now always return a query's results in one, deterministic order, even if the query itself is not sorted. This means that queries without an appropriate ORDER BY clause will not cause a hash mismatch.

9.2. Support for Ubuntu 22.04 and the Rocky OS

Volt Active Data has added Ubuntu 22.04 and Rocky OS as supported platforms for VoltDB.

9.3. Support for storing TLS/SSL credentials in Kubernetes secrets

When enabling TLS/SSL in Kubernetes, you can now store your TLS/SSL credentials (including the keystore, truststore, and passwords) in a Kubernetes secret. This avoids having to specify passwords on the Helm com-

mand line and simplifies the commands needed to start and update database instances. See the section on configuring TLS/SSL in the *Volt Kubernetes Administrator's Guide* for details.

9.4. Expiration dates for user accounts

You can now specify an expiration date for user accounts in the database configuration file. Once the specified date is past, the associated account can no longer access the database, until the configuration for the user account is updated. The expiration date is optional. See the section on defining users and roles in the *Using VoltDB* manual for details.

9.5. New LAG() windowing function

The LAG() function accesses previous rows from the window results using an offset. See the section on windowing functions in the SELECT reference page for more information.

9.6. Dedicated pod for VMC and HTTP API in Kubernetes

By default, Volt in Kubernetes now creates a separate pod for the Volt Management Center (VMC) and HTTP API. This provides a single service name for accessing these resources, as well as a single instance for the entire cluster (rather than separate instances for each host). The new pod is available from the service name `{release-name}-voltdb-vmc` where *{release-name}* is the name of the Helm release for the database cluster.

9.7. License is required on the voltdb init command

Starting with V12, the **voltdb init** command must find and load a license file or the initialization of the database root directory will fail. The license file can either be specified explicitly using the `-l` or `--license` flag or it can be found in one of the three default locations (the current working directory, the user's home directory, or the `voltdb` folder where VoltDB is installed). It is still possible to specify a license file on the **voltdb start** command — in case you need to change or update the license after initialization — but a license must be specified on the **voltdb init** command first.

9.8. Kafkaloader10 is now deprecated

To support different versions of the Kafka API, two versions of the kafkaloader utility were provided in the past: kafkaloader and kafkaloader10. Now that support for older versions of Kafka has been dropped, the legacy loader, kafkaloader10, has been deprecated and will be removed in a future release.

9.9. Old deprecated methods removed from the Java client API

Several obsolete methods in the Java client API that were previously deprecated have now been removed. Those methods were `setClientAffinity`, `setSendReadsToReplicas`, and `setReconnectOnConnectionLoss`.

9.10. The voltadmin plan_upgrade command has been removed

The procedure for upgrading the VoltDB software using limited hardware is no longer supported. The associated command, **plan_upgrade**, has been removed from the **voltadmin** utility.

9.11. @Statistics DRCONSUMER column renamed

The results of the @Statistics system procedure DRCONSUMER selector have been altered slightly. Specifically, the last column of the third results table has been renamed to be more descriptive from LAST_FAILURE to LAST_FAILURE_CODE.

9.12. Additional information in @SystemInformation output

The @SystemInformation system procedure now includes additional information regarding the number of processors, both existing and available for use. This is important because in certain situations (most

notably, Kubernetes) not all processors are available to the application. The new results field is called JAVAAVAILABLEPROCESSORS.

9.13. Additional improvements

The following limitations in previous versions have been resolved:

- **Special note when upgrading VoltDB from a version earlier than V11.2 to V11.2 or later using Kubernetes:** When upgrading VoltDB on Kubernetes from a version before V11.2 to a later release, you must specify an empty string for the `cluster.config.deployment.dr.conflictretention` property when doing the upgrade. For example:

```
$ helm upgrade mydb voltdb/voltdb --reuse-values \
   --set operator.image.tag=1.9.0 \
   --set cluster.clusterSpec.image.tag=12.0.0 \
   --set cluster.config.deployment.dr.conflictretention=""
```

  You only need to specify this property when performing the upgrade. However, if you wish to use conflict retention to limit the number of conflict logs that are retained, once the upgrade is complete, you can set the `conflictretention` property to a valid time period. For example, 30 days:

```
$ helm upgrade mydb voltdb/voltdb --reuse-values \
   --set cluster.config.deployment.dr.conflictretention=30d
```

- There was an edge case when using XDCR where, if a cluster stops and rejoins the XDCR environment, then stops again before any XDCR data is exchanged, replication is broken and the cluster must be reinitialized and join the XDCR environment from scratch to reestablish communication. This issue has been resolved.
- Previously, the **sqlcmd** utility did not always display floating point (DOUBLE) numbers correctly. This was *not* a problem with how the data was stored, but in how it was displayed, where **sqlcmd** did not use the appropriate precision. This issue has been resolved.
- The DCR round trip time statistics (that is, the columns starting *DR_ROUNDTRIPTIME_* in the @Statistics DRPRODUCER selector results) intermittently reported excessively high and incorrect latency values. This issue has been resolved.
- The Log4J configuration option for deleting old log files (MaxBackupIndex) did not work as expected. This issue has been resolved.
- Previously, attempting to configure TLS/SSL encryption using a truststore containing more than one certificate would fail. This issue has been resolved.

Known Limitations

The following are known limitations to the current release of VoltDB. Workarounds are suggested where applicable. However, it is important to note that these limitations are considered temporary and are likely to be corrected in future releases of the product.

1. Command Logging

1.1. Do not use the subfolder name "segments" for the command log snapshot directory.

VoltDB reserves the subfolder "segments" under the command log directory for storing the actual command log files. Do not add, remove, or modify any files in this directory. In particular, do not set the command log snapshot directory to a subfolder "segments" of the command log directory, or else the server will hang on startup.

2. Database Replication

2.1. Some DR data may not be delivered if master database nodes fail and rejoin in rapid succession.

Because DR data is buffered on the master database and then delivered asynchronously to the replica, there is always the danger that data does not reach the replica if a master node stops. This situation is mitigated in a K-safe environment by all copies of a partition buffering on the master cluster. Then if a sending node goes down, another node on the master database can take over sending logs to the replica. However, if multiple nodes go down and rejoin in rapid succession, it is possible that some buffered DR data — from transactions when one or more nodes were down — could be lost when another node with the last copy of that buffer also goes down.

If this occurs and the replica recognizes that some binary logs are missing, DR stops and must be restarted.

To avoid this situation, especially when cycling through nodes for maintenance purposes, the key is to ensure that all buffered DR data is transmitted before stopping the next node in the cycle. You can do this using the @Statistics system procedure to make sure the last ACKed timestamp (using @Statititcs DR on the master cluster) is later than the timestamp when the previous node completed its rejoin operation.

2.2. Avoid bulk data operations within a single transaction when using database replication

Bulk operations, such as large deletes, inserts, or updates are possible within a single stored procedure. However, if the binary logs generated for DR are larger than 45MB, the operation will fail. To avoid this situation, it is best to break up large bulk operations into multiple, smaller transactions. A general rule of thumb is to multiply the size of the table schema by the number of affected rows. For deletes and inserts, this value should be under 45MB to avoid exceeding the DR binary log size limit. For updates, this number should be under 22.5MB (because the binary log contains both the starting and ending row values for updates).

2.3. Database replication ignores resource limits

There are a number of VoltDB features that help manage the database by constraining memory size and resource utilization. These features are extremely useful in avoiding crashes as a result of unexpected or unconstrained growth. However, these features could interfere with the normal operation of DR when passing data from one cluster to another, especially if the two clusters are different sizes. Therefore, as a general rule of thumb, DR overrides these features in favor of maintaining synchronization between the two clusters.

Specifically, DR ignores any resource monitor limits defined in the deployment file when applying binary logs on the consumer cluster. This means, for example, if the replica database in passive DR has less memory or fewer unique partitions than the master, it is possible that applying binary logs of transactions that succeeded on the master could cause the replica to run out of memory. Note that these resource monitor limits *are* applied on any original transactions local to the cluster (for example, transactions on the master database in passive DR).

2.4. Different cluster sizes can require additional Java heap

Database Replication (DR) now supports replication across clusters of different sizes. However, if the replica cluster is smaller than the master cluster, it may require a significantly larger Java heap setting. Specifically, if the replica has fewer unique partitions than the master, each partition on the replica must manage the incoming binary logs from more partitions on the master, which places additional pressure on the Java heap.

A simple rule of thumb is that the worst case scenario could require an additional P * R * 20MB space in the Java heap , where P is the number of sites per host on the replica server and R is the ratio of unique partitions on the master to partitions on the replica. For example, if the master cluster is 5 nodes with 10 sites per host and a K factor of 1 (i.e. 25 unique partitions) and the replica cluster is 3 nodes with 8 sites per host and a K factor of 1 (12 unique partitions), the Java heap on the replica cluster may require approximately 320MB of additional space in the heap:

Sites-per-host * master/replace ratio * 20MB

8 * 25/12 * 20 = ~ 320MB

An alternative is to reduce the size of the DR buffers on the master cluster by setting the DR_MEM_LIMIT Java property. For example, you can reduce the DR buffer size from the default 10MB to 5MB using the VOLTDB_OPTS environment variable before starting the master cluster.

```
$ export VOLTDB_OPTS="-DDR_MEM_LIMIT=5"
$ voltdb start
```

Changing the DR buffer limit on the master from 10MB to 5MB proportionally reduces the additional heap size needed. So in the previous example, the additional heap on the replica is reduced from 320MB to 160MB.

2.5. The **voltadmin status --dr** command does not work if clusters use different client ports

The **voltadmin status --dr** command provides real-time status on the state of database replication (DR). Normally, this includes the status of the current cluster as well as other clusters in the DR environment. (For example, both the master and replica in passive DR or all clusters in XDCR.) However, if the clusters are configured to use different port numbers for the client port, VoltDB cannot reach the other clusters and the command hangs until it times out waiting for a response from the other clusters.

3. Cross Datacenter Replication (XDCR)

3.1. Avoid replicating tables without a unique index.

Part of the replication process for XDCR is to verify that the record's starting and ending states match on both clusters, otherwise known as *conflict resolution*. To do that, XDCR must find the record first. Finding uniquely indexed records is efficient; finding non-unique records is not and can impact overall database performance.

To make you aware of possible performance impact, VoltDB issues a warning if you declare a table as a DR table and it does not have a unique index.

3.2. When starting XDCR for the first time, only one database can contain data.

You cannot start XDCR if both databases already have data in the DR tables. Only one of the two participating databases can have preexisting data when DR starts for the first time.

3.3. During the initial synchronization of existing data, the receiving database is paused.

When starting XDCR for the first time, where one database already contains data, a snapshot of that data is sent to the other database. While receiving and processing that snapshot, the receiving database is paused. That is, it is in read-only mode. Once the snapshot is completed and the two database are synchronized, the receiving database is automatically unpaused, resuming normal read/write operations.

3.4. A large number of multi-partition write transactions may interfere with the ability to restart XDCR after a cluster stops and recovers.

Normally, XDCR will automatically restart where it left off after one of the clusters stops and recovers from its command logs (using the **voltdb recover** command). However, if the workload is predominantly multi-partition write transactions, a failed cluster may not be able to restart XDCR after it recovers. In this case, XDCR must be restarted from scratch, using the content from one of the clusters as the source for synchronizing and recreating the other cluster (using the **voltdb create --force** command) without any content in the DR tables.

3.5. Avoid using TRUNCATE TABLE in XDCR environments.

TRUNCATE TABLE is optimized to delete all data from a table rather than deleting tuples row by row. This means that the binary log does not identify which rows are deleted. As a consequence, a TRUNCATE TABLE

statement and a simultaneous write operation to the same table can produce a conflict that the XDCR clusters cannot detect or report in the conflict log.

Therefore, do not use TRUNCATE TABLE with XDCR. Instead, explicitly delete all rows with a DELETE statement and a filter. For example, `DELETE * FROM table WHERE column=column` ensures all deleted rows are identified in the binary log and any conflicts are accurately reported. Note that `DELETE FROM table` without a WHERE clause is *not* sufficient, since its execution plan is optimized to equate to TRUNCATE TABLE.

3.6. Use of the VoltProcedure.getUniqueId method is unique to a cluster, not across clusters.

VoltDB provides a way to generate a deterministically unique ID within a stored procedure using the getUniqueId method. This method guarantees uniqueness *within the current cluster*. However, the method could generate the same ID on two distinct database clusters. Consequently, when using XDCR, you should combine the return values of VoltProcedure.getUniqueId with VoltProcedure.getClusterId, which returns the current cluster's unique DR ID, to generate IDs that are unique across all clusters in your environment.

3.7. Multi-cluster XDCR environments require command logging.

In an XDCR environment involving three or more clusters, command logging is used to ensure the durability of the XDCR "conversations" between clusters. If not, when a cluster stops, the remaining clusters can be at different stages of their conversation with the downed cluster, resulting in divergence.

For example, assume there are three clusters (A, B, and C) and cluster B is processing binary logs faster than cluster C. If cluster A stops, cluster B will have more binary logs from A than C has. You can think of B being "ahead" of C. With command logging enabled, when cluster A restarts, it will continue its XDCR conversations and cluster C will catch up with the missing binary logs. However, without command logging, when A stops, it must restart from scratch. There is no mechanism for resolving the difference in binary logs processed by clusters B and C before the failure.

This is why command logging is required to ensure the durability of XDCR conversations in a multi-cluster (that is , three or more) XDCR environment. The alternative, if not using command logging, is to restart all but one of the remaining clusters to ensure they are starting from the same base point.

3.8. Do not use **voltadmin dr reset** to remove an XDCR cluster that is still running

There are two ways to remove an cluster from an XDCR relationship: you can use **voltadmin drop** on a running cluster to remove it from the XDCR network, or you can use **voltadmin dr reset** from the remaining clusters to remove a cluster that is no longer available. But you *should not* use **voltadmin dr reset** to remove a cluster that is still running and connected to the network. Resetting a running cluster will break DR for that cluster, but will result in errors on the remaining clusters and leave their DR queues for the reset cluster in an ambiguous state. Ultimately, this can result in the removed cluster not being able to rejoin the XDCR network at a later time.

4. TTL

4.1. Use of TTL (time to live) with replicated tables and Database Replication (DR) can result in increased DR activity.

TTL, or time to live, is a feature that automatically deletes old records based on a timestamp or integer column. For replicated tables, the process of checking whether records need to be deleted is performed as a write transaction — even if no rows are deleted. As a consequence, any replicated DR table with TTL defined will generate frequent DR log entries, whether there are any changes or not, significantly increasing DR traffic.

Because of the possible performance impact this behavior can have on the database, use of TTL with replicated tables and DR is not recommended at this time.

5. Export

5.1. Synchronous export in Kafka can use up all available file descriptors and crash the database.

A bug in the Apache Kafka client can result in file descriptors being allocated but not released if the producer.type attribute is set to "sync" (which is the default). The consequence is that the system eventually runs out of file descriptors and the VoltDB server process will crash.

Until this bug is fixed, use of synchronous Kafka export is not recommended. The workaround is to set the Kafka producer.type attribute to "async" using the VoltDB export properties.

6. Import

6.1. Data may be lost if a Kafka broker stops during import.

If, while Kafka import is enabled, the Kafka broker that VoltDB is connected to stops (for example, if the server crashes or is taken down for maintenance), some messages may be lost between Kafka and VoltDB. To ensure no data is lost, we recommend you disable VoltDB import before taking down the associated Kafka broker. You can then re-enable import after the Kafka broker comes back online.

6.2. Kafka import can lose data if multiple nodes stop in succession.

There is an issue with the Kafka importer where, if multiple nodes in the cluster fail and restart, the importer can lose track of some of the data that was being processed when the nodes failed. Normally, these pending imports are replayed properly on restart. But if multiple nodes fail, it is possible for some in-flight imports to get lost. This issue will be addressed in an upcoming release.

7. SQL and Stored Procedures

7.1. Comments containing unmatched single quotes in multi-line statements can produce unexpected results.

When entering a multi-line statement at the sqlcmd prompt, if a line ends in a comment (indicated by two hyphens) and the comment contains an unmatched single quote character, the following lines of input are not interpreted correctly. Specifically, the comment is incorrectly interpreted as continuing until the next single quote character or a closing semi-colon is read. This is most likely to happen when reading in a schema file containing comments. This issue is specific to the sqlcmd utility.

A fix for this condition is planned for an upcoming point release

7.2. Do not use assertions in VoltDB stored procedures.

VoltDB currently intercepts assertions as part of its handling of stored procedures. Attempts to use assertions in stored procedures for debugging or to find programmatic errors will not work as expected.

7.3. The UPPER() and LOWER() functions currently convert ASCII characters only.

The UPPER() and LOWER() functions return a string converted to all uppercase or all lowercase letters, respectively. However, for the initial release, these functions only operate on characters in the ASCII character set. Other case-sensitive UTF-8 characters in the string are returned unchanged. Support for all case-sensitive UTF-8 characters will be included in a future release.

8. Client Interfaces

8.1. Avoid using decimal datatypes with the C++ client interface on 32-bit platforms.

There is a problem with how the math library used to build the C++ client library handles large decimal values on 32-bit operating systems. As a result, the C++ library cannot serialize and pass Decimal datatypes reliably on these systems.

Note that the C++ client interface *can* send and receive Decimal values properly on 64-bit platforms.

9. SNMP

9.1. Enabling SNMP traps can slow down database startup.

Enabling SNMP can take up to 2 minutes to complete. This delay does not always occur and can vary in length. If SNMP is enabled when the database server starts, the delay occurs after the server logs the message "Initializing SNMP" and before it attempts to connect to the cluster. If you enable SNMP while the database is running, the delay can occur when you issue the **voltadmin update** command or modify the setting in the VoltDB Management Center Admin tab. This issue results from a Java constraint related to secure random numbers used by the SNMP library.

10. TLS/SSL

10.1. Using Commercial TLS/SSL certificates with the command line utilities.

The command line utilities, such as sqlcmd, voltadmin, and the data loaders, have an `--ssl` flag to specify the truststore for user-created certificates. However, if you use commercial certificates, there is no truststore. In that case, you need to specify an empty string to the `--ssl` command qualifier to access the database. For example:

```
$ sqlcmd --ssl=""
```

11. Kubernetes

11.1. Shutting down a VoltDB cluster by setting `cluster.clusterSpec.replicas` to zero might not stop the associated pods.

Shutting down a VoltDB cluster by specifying a replica count of zero should shut down the cluster and remove the pods on which it ran. However, on very rare occasions Kubernetes does not delete the pods. As a result, the cluster cannot be restarted. This is an issue with Kubernetes. The workaround is to manually delete the pods before restarting the cluster.

11.2. Specifying invalid or misconfigured volumes in `cluster.clusterSpec.additionalVolumes` interferes with Kubernetes starting the VoltDB cluster.

The property `cluster.clusterSpec.additionalVolumes` lets you specify additional resources to include in the server classpath. However, if you specify an invalid or misconfigured volume, Helm will not be able to start the cluster and the process will stall.

11.3. Using binary data with the Helm --set-file argument can cause problems when later upgrading the cluster.

The Helm `--set-file` argument lets you set the value of a property as the contents of a file. However, if the contents of the file are binary, they can become corrupted if you try to resize the cluster with the **helm upgrade** command, using the `--reuse-values` argument. For example, this can happen if you use `--set-file` to assign a JAR file of stored procedure classes to the `cluster.config.classes` property.

This is a known issue for Kubernetes and Helm. The workaround is either to explicitly include the `--set-file` argument again on the **helm upgrade** command. Or you can include the content through a different mechanism. For example, you can include class files by mounting them on a separate volume that you then include with the `cluster.clusterSpec.additionalVolumes` property.

Implementation Notes

The following notes provide details concerning how certain VoltDB features operate. The behavior is not considered incorrect. However, this information can be important when using specific components of the VoltDB product.

1. IPv6

1.1. Support for IPv6 addresses

VoltDB works in IPv4, IPv6, and mixed network environments. Although the examples in the documentation use IPv4 addresses, you can use IPv6 when configuring your database, making connections through applications, or using the VoltDB command line utilities, such as **voltdb** and **voltadmin**. When specifying IPv6 addresses on the command line or in the configuration file, be sure to enclose the address in square brackets. If you are specifying both an IPv6 address and port number, put the colon and port number *after* the square brackets. For example:

```
voltadmin status --host=[2001:db8:85a3::8a2e:370:7334]:21211
```

2. XDCR

2.1. Upgrading existing XDCR clusters for Dynamic Schema Change

Volt Active Data V12.3 introduces a new feature, dynamic schema change for XDCR clusters. To use this feature, clusters must be configured to enable the feature and must be using the latest DR protocol. However, existing clusters that upgrade from earlier versions will not automatically use the new protocol. Instead, you must explicitly upgrade the DR protocol using the **voltadmin protocol --upgrade** command.

First, to use dynamic schema change you must enable the feature in the configuration file. This must be done when you initialize the cluster which, for existing XDCR clusters, is easiest to when performing the version upgrade. To upgrade XDCR clusters, you must drop the cluster from the XDCR environment, upgrade the software, then reinitialize and restart the cluster. While reinitializing the cluster, add the `<schemachange enabled="true"/>` element to the configuration file. For example:

```
<deployment>
  <dr id="1" role="xcdr">
    <schemachange enabled="true"/>
    <connection source="paris.mycompany.com,rome.mycompany.com"/>
  </dr>
</deployment>
```

Similarly, for Kubernetes, the YAML would be:

```
cluster:
  config:
    deployment:
      dr:
        id: 1
        role: xdcr
        schemachange:
          enabled: true
        connection:
            source: paris.mycompany.com,rome.mycompany.com
```

Once all of the clusters are upgraded to the appropriate software version, you can upgrade the DR protocol. When used by itself, the **voltadmin dr protocol** command displays information about what versions of the DR protocol the XDCR clusters support and which version they are using. When the XDCR relationship starts, the clusters use the highest supported protocol. However, when an existing XDCR group is upgraded, they do not automatically upgrade the originally agreed-upon protocol. In this case, you must go to each cluster and issue the **voltadmin dr protocol --upgrade** command to use the highest protocol that all the clusters can support. Once you issue the **voltadmin dr protocol --upgrade** command on all of the clusters, you are then ready to perform dynamic schema changes on your XDCR environment.

3. Volt Management Center

3.1. Schema updates clear the stored procedure data table in the Management Center Monitor section

Any time the database schema or stored procedures are changed, the data table showing stored procedure statistics at the bottom of the Monitor section of the VoltDB Management Center get reset. As soon as new invocations of the stored procedures occur, the statistics table will show new values based on performance after the schema update. Until invocations occur, the procedure table is blank.

3.2. TLS/SSL for the HTTPD port on Kubernetes

Prior to VoltDB V12.0, encryption for the httpd port which VMC uses was automatically enabled on Kubernetes when you enabled TLS/SSL for the server using the `cluster.config.deployment.ssl.enabled` property. You could then selectively enable SSL for other ports using the `.dr`, `.internal`, and `.external` subproperties of `cluster.config.deployment.ssl`.

Starting with V12.0, VMC on Kubernetes is run as a separate service by default and you can enable or disable TLS/SSL encryption for VMC independently using the `vmc.service.ssl...` properties. However, if you choose not to run VMC as a separate service, by setting the `vmc.enabled` property to "false", VMC encryption is managed the same way as in earlier releases using the `cluster.config.deployment.ssl...` properties as described above.

4. SQL

4.1. You cannot partition a table on a column defined as ASSUMEUNIQUE.

The ASSUMEUNIQUE attribute is designed for identifying columns in partitioned tables where the column values are known to be unique but the table is not partitioned on that column, so VoltDB cannot verify complete uniqueness across the database. Using interactive DDL, you can create a table with a column marked as ASSUMEUNIQUE, but if you try to partition the table on the ASSUMEUNIQUE column, you receive an error. The solution is to drop and add the column using the UNIQUE attribute instead of ASSUMEUNIQUE.

4.2. Adding or dropping column constraints (UNIQUE or ASSUMEUNIQUE) is not supported by the ALTER TABLE ALTER COLUMN statement.

You cannot add or remove a column constraint such as UNIQUE or ASSUMEUNIQUE using the ALTER TABLE ALTER COLUMN statement. Instead to add or remove such constraints, you must first drop then add the modified column. For example:

```
ALTER TABLE employee DROP COLUMN empID;
ALTER TABLE employee ADD COLUMN empID INTEGER UNIQUE;
```

4.3. Do not use UPDATE to change the value of a partitioning column

For partitioned tables, the value of the column used to partition the table determines what partition the row belongs to. If you use UPDATE to change this value and the new value belongs in a different partition, the UPDATE request will fail and the stored procedure will be rolled back.

Updating the partition column value may or may not cause the record to be repartitioned (depending on the old and new values). However, since you cannot determine if the update will succeed or fail, you should not use UPDATE to change the value of partitioning columns.

The workaround, if you must change the value of the partitioning column, is to use both a DELETE and an INSERT statement to explicitly remove and then re-insert the desired rows.

4.4. Ambiguous column references no longer allowed.

Starting with VoltDB 6.0, ambiguous column references are no longer allowed. For example, if both the *Customer* and *Placedorder* tables have a column named *Address*, the reference to *Address* in the following SELECT statement is ambiguous:

```
SELECT OrderNumber, Address FROM Customer, Placedorder
   . . .
```

Previously, VoltDB would select the column from the leftmost table (Customer, in this case). Ambiguous column references are no longer allowed and you must use table prefixes to disambiguate identical column names. For example, specifying the column in the preceding statement as *Customer.Address*.

A corollary to this change is that a column declared in a USING clause can now be referenced using a prefix. For example, the following statement uses the prefix *Customer.Address* to disambiguate the column selection from a possibly similarly named column belonging to the *Supplier* table:

```
SELECT OrderNumber, Vendor, Customer.Address
   FROM Customer, Placedorder Using (Address), Supplier
    . . .
```

5. Runtime

5.1. File Descriptor Limits

VoltDB opens a file descriptor for every client connection to the database. In normal operation, this use of file descriptors is transparent to the user. However, if there are an inordinate number of concurrent client connections, or clients open and close many connections in rapid succession, it is possible for VoltDB to exceed the process limit on file descriptors. When this happens, new connections may be rejected or other disk-based activities (such as snapshotting) may be disrupted.

In environments where there are likely to be an extremely large number of connections, you should consider increasing the operating system's per-process limit on file descriptors.

5.2. Use of Resources in JAR Files

There are two ways to access additional resources in a VoltDB database. You can place the resources in the `/lib` folder where VoltDB is installed on each server in the cluster or you can include the resource in a subfolder of a JAR file you add using the sqlcmd **LOAD CLASSES** directive. Adding resources via the `/lib` directory is useful for stable resources (such as third-party software libraries) that do not require updating. Including resources (such as XML files) in the JAR file is useful for resources that may need to be updated, as a single transaction, while the database is running.

LOAD CLASSES is used primarily to load classes associated with stored procedures and user-defined functions. However, it will also load any additional resource files included in subfolders of the JAR file. You can remove classes that are no longer needed using the **REMOVE CLASSES** directive. However, there is no explicit command for removing other resources.

Consequently, if you rename resources or move them to a different location and reload the JAR file, the database will end up having multiple copies. Over time, this could result in more and more unnecessary memory being used by the database. To remove obsolete resources, you must first reinitialize the database root directory, start a fresh database, reload the schema (including the new JAR files with only the needed resources) and then restore the data from a snapshot.

5.3. Servers with Multiple Network Interfaces

If a server has multiple network interfaces (and therefore multiple IP addresses) VoltDB will, by default, open ports on all available interfaces. You can limit the ports to an single interface in two ways:

- Specify which interface to use for internal and external ports, respectively, using the **--internalinterface** and **--externalinterface** arguments when starting the database process with the **voltdb start** command.

- For an individual port, specify the interface and port on the command line. For example **voltdb start --client=32.31.30.29:21212**.

Also, when using an IP address to reference a server with multiple interfaces in command line utilities (such as **voltadmin stop node)**, use the @SystemInformation system procedure to determine which IP address VoltDB has selected to identify the server. Otherwise, if you choose the wrong IP address, the command might fail.

5.4. Using VoltDB where the /tmp directory is noexec

On startup, VoltDB extracts certain native libraries into the /tmp directory before loading them. This works in all standard operating environments. However, in custom installations where the /tmp storage is mounted with the "noexec" option, VoltDB cannot extract the libraries and, in the past, refused to start.

For those cases where the /tmp directory is assigned on storage mounted with the 'noexec' option, you can assign an alternative storage for VoltDB to use for extracting and executing temporary files. This is now done automatically on Kubernetes and does not require any user intervention.

On non-Kubernetes environments, you can identify an alternative location by assigning it to `volt.tmpdir`, `org.xerial.snappy.tmpdir`, and `jna.tmpdir` in the VOLTDB_OPTS environment variable before starting the server process. The specified location must exist, must be an absolute path, and cannot be on storage mounted with the "noexec" option. For example, the following command assigns an alternate temporary directory called `/volttemp`:

```
export VOLTDB_OPTS="-Dvolt.tmpdir=/volttemp \
                    -Dorg.xerial.snappy.tempdir=/volttemp \
                    -Djna.tmpdir=/volttemp"
```

When using an alternate temporary directory, files can accumulate over time since the directory is not automatically purged on startup. So you should make sure you periodically delete old files from the directory.

6. Platforms

6.1. OpenShift and Transparent Huge Pages (THP)

For production, VoltDB requires that Transparent Huge Pages (THP) are disabled because they interfere with memory-intensive applications. However, THP may be enabled on OpenShift containers and the containers themselves not have permission to disable them. To overcome this situation, you must run the Helm chart for disabling THP from a privileged container:

```
$ helm -n kube-system install thp voltdb/transparent-hugepages \
       --set thp.securityContext.privileged=true
```

6.2. Kubernetes Compatibility

See the *Volt Kubernetes Compatibility Chart* for information on which versions of the Volt Operator and Helm charts support which version of VoltDB. See the *VoltDB Operator Release Notes* for additional information about individual releases of the VoltDB Operator.