Metrics Preview

July, 2023

Beta release for testing and feedback.

Revision History
12.3.0

Abstract

The following document describes functionality available in VoltDB 12.3.0 and later. This is a beta release designed specifically to solicit feedback. The new capabilities are believed to work as described. However, further testing and real-world evaluation may identify further changes or improvements needed to optimize the functionality and/or syntax.

Your feedback concerning new functionality is critical to improving VoltDB. Please send all comments, questions, and bug reports to *betasupport@voltactivedata.com*.

Metrics

VoltDB V12.3 includes the beta release of a new reporting system aimed at providing a simpler, more consistent interface for information about the database, its status, and performance. The new system has two primary goals:

- Provide a better, more seamless integration with the industry-standard Prometheus/Grafana monitoring and reporting ecosystem.
- Rationalize, and ultimately replace, multiple distinct reporting interfaces in the Volt product.

The initial goal for the new system is to replace the current external Prometheus agent, which will be deprecated. The new metrics system provides a number of advantages over the existing Prometheus agent. Although they both offer Prometheus-compliant interfaces the new metrics system is built into the server, so the metrics API starts and stops with the database unlike the external agent. The new system also eliminates the overhead associated with the external agent making multiple calls to the database to retrieve and restructure statistics in Prometheus format. Finally, the built-in services reports data on a per-server basis, which is more consistent with the way other cluster services report information to Prometheus. It also reduces the intra-cluster coordination needed to report cluster-wide statistics.

Using the New Metrics System

To use the new metrics system, you must first enable it in the configuration file when initializing the database. You do this by adding the <metrics> element to the configuration, like so:

```
<deployment>
```

```
   <cluster kfactor="1"/>
   <metrics enabled="true"/>
</deployment>
```

Once enabled, metrics can be retrieved from each server in the cluster through the metrics port, which defaults to 11781. (You can specify an alternate port and/or network interface using the **--metrics** qualifier on the **voltdb start** command.)

On Kubernetes, metrics are enabled by setting the appropriate per pod metrics properties. For example:

```
--set cluster.config.deployment.metrics.enabled=true     \
--set cluster.serviceSpec.perpod.metrics.enabled=true    \
--set cluster.serviceSpec.service.metrics.type=ClusterIP
```

One advantage of using the metrics system with Prometheus on Kubernetes is that the new metrics ports are identified automatically and by the Prometheus infrastructure. On other platforms, you should add the metrics port as a data source to your Prometheus server's configuration.

For custom metrics collection and reporting, there is also a Volt native system procedure for retrieving the same information that is available from the metrics port. The @Metrics system procedure (described below) returns the same information formatted in a sequence of VoltTable structures.

Known Limitations for the Beta Release

The following are the known limitations to this preview of metrics.

1.1. New @Metrics system procedure

The new @Metrics system procedure (described below) is still under development and the order, number, and/or names of the columns and rows may change before the final availability for production use.

Reference Documents

The following reference documentation describes the @Metrics system procedure.

@Metrics

@Metrics — Returns information about the performance and current status of the database server

Syntax

```
@Metrics
```

Description

The @Metrics system procedure returns information about the status and performance of the database server. This information is only available if metrics are enabled when the database is started by including the `<metrics enabled="true">` element in the configuration file.

Metrics are normally accessed through the metrics port via a network request. The format and request method when using the metrics port are compatible with Prometheus, an industry standard metrics collection and reporting system. The @Metrics system procedure returns the same information, however, structured as a series of VoltTables, appropriate for a client application.

Return Values

The procedure returns one VoltTable summarizing the amount of data being returned, then a series of VoltTables grouping the metrics by topic, with specific columns and number of rows dependent on the information required by each topic.

Example

The following program example filters the results by the metric name MEMORY_TUPLEDATA and prints out the total number of bytes currently used to store tuples.

```
try {
  VoltTable[] results = client.callProcedure("@Metrics").getResults();
  String metric = "MEMORY_TUPLEDATA";
  int count = 0;
  int tables = 0;
  for (VoltTable node : results) {
    if (tables++ ==1) continue; // Skip summary row
    node.resetRowPosition();
    while(node.advanceRow()) {
      if (node.getString("NAME").equals(metric)) {
      count += node.getLong("VALUE");
    }
  }
  System.out.println("Tuple Memory in use: " + String.valueOf(count));
} catch (Exception e) { e.printStackTrace(); }
```