**Tracing PostgreSQL Performance & Health Markers**

Key Health Indicators (KHI) for PostgreSQL:

1. **Response Time for the Queries**: the time required for displaying the results should be lower than required and also the reply should be consistent (on entering the same query again and again, the result should not differ, it should be the same).

(Reference: <http://www.slideshare.net/PGExperts/database-health-check-15833445>)

Good: 0 ~ 0.02 seconds

Bad: 0.28 ~ 0.3 seconds

1. **Server Load**: the number of processes which are in waiting state and ready to access the processors. The number of processes in the waiting queue should be less compared to ready queue.

(Reference: <http://www.slideshare.net/PGExperts/database-health-check-15833445>)

1. **Connection Limitations**: the number of active connections should not exceed the limit for the number of connections allowed for that database. It can be represented as two major categories: warning (in which the connection number is about to reach the limit) and critical (in which an action must be taken immediately to control the situation).

(Reference: <http://demo.mindarraysystems.com/>)

Good: Anything less than 100 (default number of connections allowed in PostgreSQL).

Bad: Exceeding 100 will cause database to request more shared memory.

**According to Heroku Postgres Legacy Plans**: Max. 20 connections for starter (Basic Usage)

Max. 500 connections for production (Application Level)

(Reference: https://devcenter.heroku.com/articles/heroku-postgres-legacy-plans)

1. **Idle Connections**: due to poor session management, the number of idle connections with uncommitted transactions increases that leads to slower response times. Although they are not of much concern but they tend to interfere with the transactions happening.

(Reference: <http://dba.stackexchange.com/questions/39697/could-too-many-idle-connections-affect-postgresql-9-2-performance>)

Good: Idle connections are not a worry until they are associated with uncommitted transactions.

Bad: Even 1 Idle connection with uncommitted transaction is bad for the database.

1. **Longest Queries**: queries that takes up a lot of time to execute. This situation can sometimes lead to starvation for other pending queries.

(Reference: <http://demo.opm.io/server/1>)

Key Performance Indicators (KPI) for PostgreSQL:

1. **Size of the Database**: the size of database and tables should be checked regularly as the size grows faster than expected. Larger sizes leads to database bloat.

Elements:-

* Value: Open PostgreSQL Monitoring suggests that size should be less than 50GiB. If it exceeds this value, it is better to go for partitions.
* Goal: do not let the size increase above 50GiB.
* Status: 0 (medium priority).
* Trend: user data tends to increase the size of the database largely.

(Reference: <http://demo.opm.io/graphs/showservice/dalibo_epoisses/PGSQL%20-%20Database%20size> )

Reference: <http://www.postgresql.org/about/>

Max. DB Size: Unlimited but only databases up to 4TB are reported to exist. (Attachment)

Max. Table Size: 32TB

Max. Row Size: 1.6TB

Max. Rows per Table: Unlimited

1. **Lock Statistics**: the number of locks held at a particular time.

Elements:-

* Value: An upper limit should be set for exclusive locks (write) whereas no such limit is required for shared locks (read).
* Goal: adhere to the set limits for both types of locks.
* Status: high priority.
* Trend: the number of transactions increasing per second leads to increasing number of locks.

(Reference: [https://www.manageengine.com/products/applications\_manager/postgresql-monitoring.html Image 1](https://www.manageengine.com/products/applications_manager/postgresql-monitoring.html%20Image%201))

There is no good or bad threshold for shared locks but only exclusive locks can be bad. “The more the locks the worse the database health”. Even, if there are few locks or no locks, the also it is bad as it depicts nothing is happening inside the database.

Reference: <https://www.compose.io/articles/common-misconceptions-about-locking-in-postgresql/>

1. **Temporary Files Generation**: queries that generate most (in number) and largest (in size) temporary files on their execution.

Elements:-

* Value: a track of number of written files and recycled files should be kept.
* Goal: keep the number of written files and recycled files nearly the same.
* Status: low priority.
* Trend: increasing creation of views and tables lead to temp files generation.

(Reference: <http://dalibo.github.io/pgbadger/> )

Good: Ideally, it should be less than 1GB.

Bad: Should not exceed 50GB.

1. **Cache Hit Ratio**: the large number of references out of total references to the cache should result in a hit and there should be fewer memory references, in case of a miss.

Elements:-

* Value: effective cache size can be set manually in PostgreSQL.
* Goal: minimize total main memory references.
* Status: high priority.
* Trend: increasing cache size solves this problem to a larger extent.

(Reference: <http://www.postgresql.org/docs/9.1/static/runtime-config-query.html> )

Good: The hit percentage should be greater than 99%.

Bad: Anything lower than 99% should be reported and increasing the cache size should be considered.

Reference: <http://www.craigkerstiens.com/2012/10/01/understanding-postgres-performance/>

1. **Memory Performance**: a track of total memory used, memory free, memory used in cache and fragmentation details should be recorded.

(Reference: <http://demo.opm.io/graphs/showservice/dalibo_epoisses/SYSTM%20-%20Memory> )

PostgreSQL uses much less memory (RAM) than it looks like.

Reference: <http://www.depesz.com/2012/06/09/how-much-ram-is-postgresql-using/>

1. **Transactions Rate**: the count of number transactions successfully taking place per second determines the performance to a larger extent. This also includes the number of commits and rollbacks happening at the time of transactions. The number of commits should be more than the number of rollbacks to declare that the database performing up to the mark. A rollback gives an impression of failed transactions.

(Reference: <http://dba.stackexchange.com/questions/35940/how-many-queries-per-second-is-my-postgres-executing> )

Good: There is no upper limit to the number of transactions performed per second, it is the success rate that matters.

Bad: Number of failed transactions (uncommitted and rollbacks) exceeds 5-10% of total transactions.

1. **Query Performance**: the number of updated, deleted and inserted rows per second determines the performance of the query entered in the database.

(Reference: <http://demo.mindarraysystems.com/loadHomePage#bmF2aWdhdGlvbj1tb25pdG9yfn5tb25pdG9ySWQ9OA> )

This again does not require any upper limit as more the operations per second are performed, healthier the database is.

Bad threshold is inconsistent and erroneous operations that can be measured by the time it is taking to execute.

**Note 1**: There is one parameter that I found is used in most of the database monitoring models, “**Streaming Replication Lag**”, but I couldn’t understand what it is about and how it is related to databases.

**Note 2**: Couldn’t determine PostgreSQL specific threshold for “Server Load” and “Longest Queries”. The latter one can be combined with “Response Time for Query” because the longer the query is taking to execute, the more is the response time.