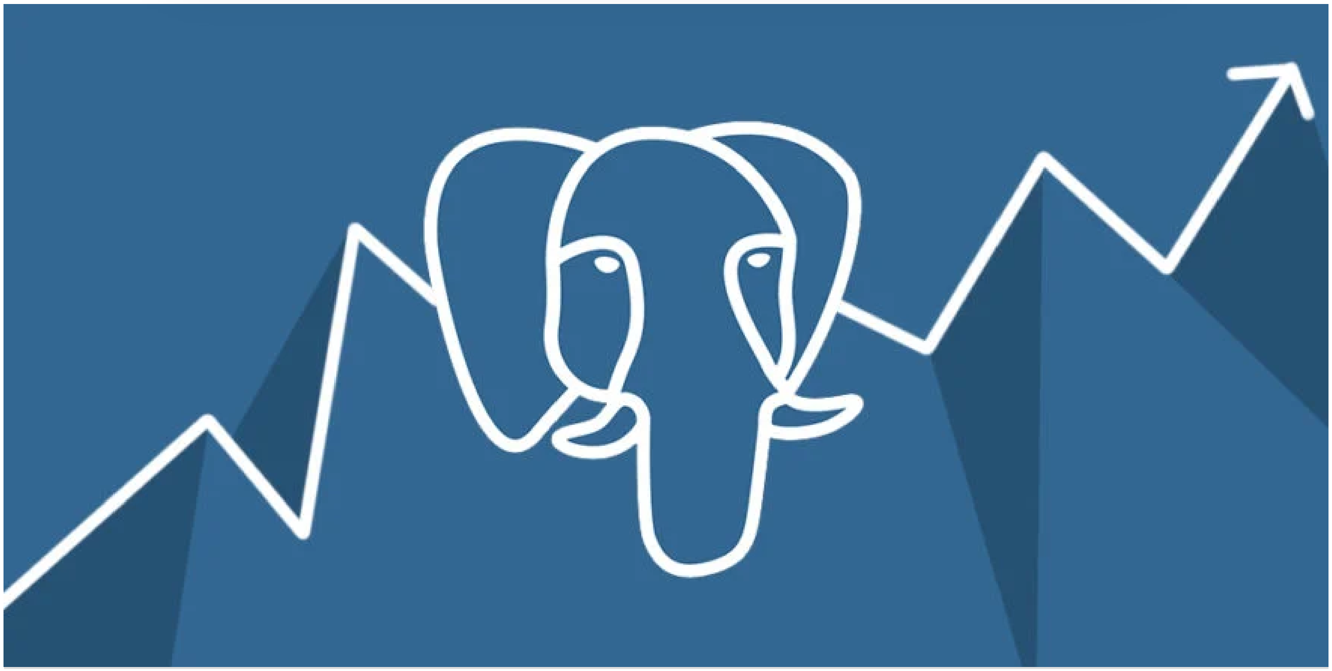


PostgreSQL Performance Tuning Tutorial

LOU BICHARD | NOVEMBER 6, 2018 |
DEVELOPER TIPS, TRICKS & RESOURCES (HTTPS://STACKIFY.COM/DEVELOPERS/)

Setting up the database correctly is just the first step in PostgreSQL performance tuning. With every table you add and every query you run, your databases will require maintenance and updates to ensure ideal PostgreSQL optimization.

PostgreSQL optimization is pretty straight-forward, however, there are some things that it needs to know from you, the database admin, in order to run effectively.

When it c on, one rule appl our database up, you can start (https://c s like Retrace (- management/). This is important because PostgreSQL performance tuning is all about trade-offs. You might accept a slow-running query in one part of your application that isn't used so often in order to get a lightning-fast response time on a frequently performed query. Download your free, two week trial of Retrace today. (https://s1.stackify.com/account/createclient?_ga=2.79824532.2081195068.1584382175-1971815645.1570122931&_gac=1.11298432.1584390051.EAIaIQobChMIven

Today I'm going to walk you through some different PostgreSQL optimizations. First, we'll go through some of the basic setup options available:

- **Hardware updates**—Changes you can make to the physical server
- **Configuration**—Modifications to the out-of-the-box PostgreSQL configuration
- **Vacuuming**—Ways in which vacuum settings can improve performance

Once your system is set up, we'll see how you can analyze and improve your schema:

- **Analyze your query performance**—How to analyze individual queries
- **Analyze your logs**—How to get information out of your system for analysis

Part 1: Best Practices and Setup

In the first part of this article, we’re going to go through how you can alter your basic setup for faster PostgreSQL performance. In part 2, I’ll cover how to optimize your system specifics, such as query optimizations. Let’s get to it!

Beefing

The obvi SQL performa he of the most memory, CPU and

When it comes to hardware updates, you should consider the following:

Updating your memory. Memory is what your system uses to effectively cache data it needs often. It's important to keep memory in mind when you're optimizing your queries later. The bigger your cache, the less your database will need to go out to disk—which can be painful for performance. PostgreSQL tries to hold the most frequently accessed data in memory to make performance improvements based on how your queries are performed and the configuration that you give it. But we'll return to memory-based performance optimization later.

Separating the application from the database. On the note of increasing memory, if you're running your database application on the same server as your application, you may want to consider isolating it. It'll be difficult to truly analyze and improve your database performance if a separate component is affecting your metrics and environment.

Database configuration: The what, why, and how

SOURCE: PIXCOVE.COM (HTTPS://WWW.PIXCOVE.COM/CONTROLS-PANELS-KNOB-DIALS-KNOBS-DIAL-SWITCHES-CHANGES-AMPLIFIER-AUDIO-LOUDSPEAKER-GAIN/)

PostgreSQL comes with a set of default configurations. This default configuration is set up for compatibility and is essentially a best guess that tries to suit all possible use cases of PostgreSQL. Luckily for you, that means there are some potential quick wins if you start to tweak the default configuration.

Before we dive in and go through some of the most common configuration optimizations you can make, it's worth pointing out that there are several PostgreSQL performance tuning tools available—such as PGTune (https://pgtune.leopard.in.ua/#/)—that try to do some of this configuration legwork for you. Their effectiveness varies, but it might be worth experimenting to the delta between the output of these tools and your current database configuration.

Database configurations in PostgreSQL are made directly in the configuration file (*postgresql.conf*), or through running an ALTER SYSTEM (https://www.postgresql.org/docs/current/static/sql-

altersyste
configura
(https://v mmand
as follow

```
SHOW ALL
```

This command will list out all of the existing configuration and their settings. It's also important to note that different configurations will apply only in certain conditions, such as a database restart. Some configuration changes will require a server restart; others will require a reload of the configuration.

When you've made configuration changes to your database, you can see details such as whether a restart is required for your configuration update by running the following command:

```
SELECT * FROM pg_settings WHERE pending_restart = true;
```

PostgreSQL performance tuning configuration changes

You will probably want to modify a lot of different configurations to get the most out of your PostgreSQL database. Let's go through some of the main configurations that you can change to get more performance from your system.

max_connections

Connections are how your applications communicate with your database. Each connection will require some communication chatter and setup in order to establish. However, once established, queries can be sent down the wire to your database. Your PostgreSQL database will come with a default number of connections, which you can alter. You want to make sure that your applications aren't unnecessarily

connecti Memory
allocatio want to
balance l ctions.

checkp

A checkpoint is a periodic action that stores information about your system. By default, a checkpoint will run after a number of segments, but depending on your system, you may want to increase this value. We'll talk later in the article about how you can log out your checkpoint data, but checkpoint configuration is important because it can be quite an expensive operation. It's often thought that the default configuration is too aggressive and performs checkpoints too often, so you might want to increase this value to make checkpoints less frequent.

work_mem

As I mentioned earlier, memory allocation and management is a big part of PostgreSQL performance tuning. If your system is doing a lot of complex sorts, increasing the sort memory can help the database optimize its configuration for your setup. This allows PostgreSQL to cache more data in memory while it performs its sorting, as opposed to making expensive calls to the disk.

random_page_cost

This setting essentially is the amount of time that your optimizer should spend reading memory before reaching out to your disk. You should alter this setting only when you've done other plan-based optimizations that we'll cover soon, such as vacuuming, indexing, or altering your queries and schema.

These ar abase configura

(https://v ml). Now that you another area for i

Use vacuuming to avoid bloat

The next area we're looking at is vacuuming. A vacuum is a scan that marks tuples as no longer needed so that they can be overwritten. Failure to do this can mean that you have dead tuples lingering in your system. These dead tuples are often called *bloat.* Bloat originates mostly from records that are being deleted, updated, or inserted.

PostgreSQL sets up vacuuming by default, but just as you can configure other settings, you can also set your vacuuming settings. You can even set vacuuming settings on a per-table basis for more fine-grained optimization.

To perform a vacuum, you simply run the command:

```
VACUUM
```

If you want to see the history of previously run vacuums, you can do so by executing the following:

```
SELECT * FROM pg_stat_user_tables
```

Generally speaking, regular vacuuming can't be done enough. More frequent vacuuming keeps bloat to a minimum and ensures that database performance stays high. Although autovacuuming is already set up, you might want to alter the settings to be more aggressive.

Part 2

We'd like to show you notifications for the latest news and updates.

Cancel Allow

Upfront s n
performa on
upfront, he data
and anal ow it's
set up, know how it's being queried, and optimize based on these use cases. PostgreSQL and the community give us some handy tools for making these optimizations.

Update Postgres algorithm data with ANALYZE

Before we continue, it's important to understand the life cycle of a query. The life cycle is what happens from an initial request to the database right up to the response it sends. PostgreSQL doesn't just dip its hand into a big bag of memory. When you have a lot of data, crude fetching of your data can lead to performance drops. If you're scanning your database sequentially (often called a table scan) for your data, your performance will scale linearly— more rows, slower performance. But we can do better than that.

So what is the life cycle of a query? Initially, there is a transmission of the query string to PostgreSQL. The query string is then parsed and a plan is created. Plans are really important. Plans are the steps that PostgreSQL is going to take to find requested data. How you set up your database configuration, your schema, and your indexes (more on indexes soon) will all affect how performant these plans are. So, it'll be important that we understand plans and how to optimize them. Finally, the database will perform the plan and retrieve the data.

A discrepancy can occur between the database plan that PostgreSQL intends to use to get your data and the way it actually fetches your data. This is because PostgreSQL bases its plan on infreque be updated e up to date. n

This is wh and updates about how to create its plans. So, if you're updating the tables or schema or adding indexes, remember to run an ANALYZE command after so the changes will take effect.

Deeply understand your query performance

One of the next obvious areas to go to is query optimization. The queries that you are running might be inefficient (https://stackify.com/writing-better-sql-queries/) for many reasons, so we'll need a way to understand what's going on with these queries—which is where the EXPLAIN command comes in.

EXPLAIN gives you a precise breakdown of how PostgreSQL will execute your query. The plan is based on statistics about the table, and it takes into account things like database indexes to find the most efficient path to your data. But EXPLAIN will only guess and give you a plan that it thinks it will execute. You can run EXPLAIN ANALYZE to get not only the information about the predicted plan, but also an update on how the query performed.

When you've got the plan that the database is trying to perform, you can then begin to analyze it. Several tools exist to make life easier, such as PEV, (http://tatiyants.com/pev/#/plans) which allows you to visualize and understand the output of your EXPLAIN command.

Logs as a source of high-quality PostgreSQL performance tuning data

Let's assu king to
debug slo nt to do
this is thr the
applicati then
gather u nt tool
like Retra get a
picture of how the system is behaving.

By default, your application doesn't log all data. This is because increases in logging also have an effect on database performance. So, while we go through how you can change log settings to analyze performance (https://stackify.com/retrace-application-performance-management/), remember that the log settings themselves can actually affect performance.

Logs are emitted to a file in your system, depending on your configuration. When you've found where your logs are at, you can use a tool such as Retrace (https://stackify.com/retrace-log-management/) to analyze these logs. Retrace will tell you stats such as the most frequently run queries, how long the queries take on average, and so on. These aggregate metrics can give you a better understanding of where you may have performance bottlenecks within your system.

So, now that we see how great logs are for understanding system performance, how do we go about setting them up or changing them?

log_line_prefix

This is the format of your database logs. For tools like PGBadger, they'll need rich data and will also need to know the data format. Your log line prefix tells PostgreSQL what format to emit its log data in. If you want to do any meaningful log analysis, you'll need to set this value to be compatible with the tooling you are using to analyze your logs.

log_sta

This is th efer to the level ant absolute tement has a few

- *ddl*, which logs only structural changes to the database
- *mod*, which logs modifications to existing data (basically everything except SELECT)
- *all*, which logs … everything

log_checkpoints

As we discussed in the configuration settings, checkpoints in PostgreSQL are periodic actions that store data about your system. These log checkpoints can, if excessive, lead to performance degradations. If you suspect this might be the case, by enabling log checkpoints you'll be able to see rich data about those checkpoints, such as how often they're running and what could be triggering them.

logging_connection

You also might want to know information about connections. If you've only got one application connecting to your database, but you're seeing many concurrent connections, something could be wrong. Too many connections flooding your database could also mean that requests are failing to reach the database and could be affecting your application's end users.

Now that we've got lots of data about our system, what tools do we have for improving the structure of our system to be more performant?

Faster queries with database indexes

In a worl d lead
to a full s ults.
With a la here
database rmation
to your d e data
you're loo an
understanding of our data and how we're trying to access it. Which is why it's important we have observability and monitoring tools, like Retrace (https://stackify.com/retrace/) setup to help us. Indexes don't come for free, however; if you updated the content of a book, you'll have to update the index each time you make a change to the content.

PostgreSQL has a few different types of index. The difference between these is that they all use a different algorithm. The indexes available are B-tree (the default index), Hash, GiST, SP-GiST, and GIN. PostgreSQL will create implicit indexes when you create a primary key or a unique key constraint. However, you will have to add the rest of your indexes manually.

The basic syntax for an INDEX is:

```
CREATE INDEX index_name ON table_name;
```

However, you shouldn't use an index in some use cases—for example, when the overhead of using the index exceeds the benefit from the algorithm, such as on a small table. But tables that have large batch updates performed might also see performance issues. It may make sense to remove indexes on these tables temporarily while it's updating, before restoring indexes.

There you have it: lightning-fast PostgreSQL performance tuning

Hopefully greSQL
performa analyze
your que t, and
modify y e able to
tweak yo t the
most out

olutions ⌄

We'd like to show you notifications for the latest news and updates.

Cancel Allow

Just remember, tuning your database will take time and practice. So be patient and stay curious to find out more about your system to get the best performance results.

Improve Your Code with Retrace APM

Stackify's APM tools are used by thousands of .NET, Java, PHP, Node.js, Python, & Ruby developers all over the world.
Explore Retrace's product features to learn more.

(/retrace-application-performance-management/)

App Performance Management (https://stackify.com/retrace-application-performance-management/)

(/retrace-code-profiling/)

Code Profiling (https://stackify.com/retrace-code-profiling/)

Error Tracking (https://stackify.com/retrace-error-monitoring/)

(/retrace-log-management/)

Centralized Logging (https://stackify.com/retrace-log-management/)

(/retrace-app-metrics/)

App & Server Metrics (https://stackify.com/retrace-app-metrics/)

Learn More
(/retrace/)

About the Author | Latest Posts

About Lou Bichard

Lou Bichard is a JavaScript full stack engineer with a passion for culture, approach, and delivery. He believes the best products emerge from high

perform nd
systems olutions

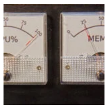
Search Stackify

Search

Topics/Keywords

- ASP.NET (https://stackify.com/?tag=asp.net,net-core)
- .NET Core (https://stackify.com/content/net-core/)
- Java (https://stackify.com/content/java/)
- Azure (https://stackify.com/content/azure/)
- AWS (https://stackify.com/content/AWS/)
- Cloud (https://stackify.com/?tag=cloud,azure,aws)
- Product Updates (https://stackify.com/stackify/)
- App Monitoring (https://stackify.com/?tag=monitoring,apm)
- App Performance Tips (https://stackify.com/?tag=performance,profiler,apm)
- Error Handling (https://stackify.com/?tag=exception,exceptions,error,errors)
- Logging Tips (https://stackify.com/?tag=logs,logging)
- DevOps (https://stackify.com/content/DevOps/)

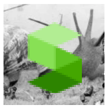
Popular Posts



How to Troubleshoot IIS Worker Process (w3wp) High CPU Usage

How to Monitor IIS Performance: From the Basics to Advanced IIS Performance Monitoring

SQL Performance Tuning: 7 Practical Tips for Developers



olutions ⌄

Look

etrace

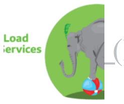
5 Aw

We'd like to show you notifications for the latest news and updates.

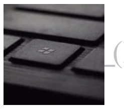
Cancel Allow

Recent Posts

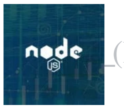
(https://stackify.com/how-to-load-test-your-php-website/) How To Load Test Your PHP Website (https://stackify.com/how-to-load-test-your-php-website/)



(https://stackify.com/best-website-load-testing-services/) Best website load testing services (https://stackify.com/best-website-load-testing-services/)



(https://stackify.com/how-to-monitor-a-windows-process/) How to Monitor a Windows Process (https://stackify.com/how-to-monitor-a-windows-process/)



(https://stackify.com/best-5-tools-for-node-js-monitoring/) Best 5 Tools for Node.js Monitoring (https://stackify.com/best-5-tools-for-node-js-monitoring/)

(https://stackify.com/how-to-monitor-website-performance/) How to monitor website performance (https://stackify.com/how-to-monitor-website-performance/)

(https://stackify.com/guest-blogging-guidelines/)

Prefix

We'd like to show you notifications for the latest news and updates.

Cancel Allow

olutions ⌄

n

.NET Monitoring

Java Monitoring

PHP Monitoring

Node.js Monitoring

Ruby Monitoring

Python Monitoring

Retrace vs New Relic

Retrace vs Application Insights

Solutions

Application Performance Management

Centralized Logging

Code Profiling

Error Tracking

Application & Server Monitoring

Real User Monitoring

For Developers

For DevOps

Resources

What is APM?

Pricing

We'd like to show you notifications for the latest news and updates.

Cancel Allow

PO Box 2159
Mission, KS 66201
816-888-5055 (tel:18168885055)

(https://www.facebook.com/Stackify/) (https://twitter.com/Stackify) (https://www.youtube.com/channel/UCnxYNQDtTr_ZYUnMjeACfCg) (https://www.linkedin.com/company/stackify/)