Everything Everywhere All at Once: PostgreSQL configuration guide

DMITRII DOLGOV

15-12-2022

Red Hat

https://wiki.postgresql.org › wiki › Tuning_Your_PostgreSQL_Server

Tuning Your PostgreSQL Server - PostgreSQL wiki

Tuning Your PostgreSQL Server. by Greg Smith, Robert Treat, and Christopher Browne. PostgreSQL ships with a basic configuration tuned for wide compatibility rather than performance. Odds are good the default parameters are very undersized for your system....

https://wiki.postgresql.org › wiki › Tuning_Your_PostgreSQL_Server

Tuning Your PostgreSQL Server - PostgreSQL wiki

Tuning Your Post
PostgreSQL ship
performance. Oc

https://www.percona.com › blog › tuning-postgresql-database-parameters-to-optimize-perf...

Exploring PostgreSQL Performance Tuning Parameters

Key areas include: Configuration parameter tuning: This tuning involves altering variables such as memory allocation, disk I/O settings, and concurrent connections based on specific hardware and requirements.

https://wiki.postgresql.org › wiki › Tuning_Your_PostgreSQL_Server

Tuning Your PostgreSQL Server - PostgreSQL wiki

Tuning Your Post
PostgreSQL ship
performance. Oc

https://www.percona.com › blog › tuning-postgresql-database-parameters-to-optimize-perf...

Exploring PostgreSQL Performance Tuning Parameters

Key areas include: Configuration parameter tuning: This tuning involves altering variables such as memory allocation, disk I/O settings, and concurrent connections based on specific

https://www.enterprisedb.com › postgres-tutorials › introduction-postgresql-performance...

An Introduction to PostgreSQL Performance Tuning and Optimization

This document provides an introduction to tuning PostgreSQL and EDB Postgres Advanced Server (EPAS), versions 10 through 13. The system used is the RHEL family of linux distributions, version 8.

Red Hat

https://wiki.postgresql.org › wiki › Tuning_Your_PostgreSQL_Server

Tuning Your PostgreSQL Server - PostgreSQL wiki

Tuning Your Pos

https://www.percona.com › blog › tuning-postgresql-database-parameters-to-optimize-perf...

erformance Tuning Parameters

n parameter tuning: This tuning involves altering variables such
settings, and concurrent connections based on specific

https://www.crunchydata.com › blog › optimize-postgresql-server-performance

Optimize PostgreSQL Server Performance Through Configuration - Crunchy Data

The value of work_mem is used for complex sort operations, and defines the maximum amount of memory to be used for intermediate results, such as hash tables, and for sorting. When the value for work_mem is properly tuned, then the majority of sort actions are performed in the...

e...

An Introduction to PostgreSQL Performance Tuning and Optimization

This document provides an introduction to tuning PostgreSQL and EDB Postgres Advanced Server (EPAS), versions 10 through 13. The system used is the RHEL family of linux distributions, version 8.

https://wiki.postgresql.org › wiki › Tuning_Your_PostgreSQL_Server

Tuning Your PostgreSQL Server - PostgreSQL wiki

Tuning Your Pos

https://www.percona.com › blog › tuning-postgresql-database-parameters-to-optimize-perf...

erformance Tuning Parameters

n parameter tuning: This tuning involves altering variables such

https://www.crunchydata.com › blog › optimize-postgresql-server-performance

Optimize PostgreSQL Server Performance Through Configuration - Crunchy Data

The value of work_mem is used for complex sort operation
of memory to be used for intermediate results, such as has
value for work_mem is properly tuned, then the majority of

An Introduction to PostgreSQL Performan

This document provides an introduction to tuning Pos
Server (EPAS), versions 10 through 13. The system us
distributions, version 8.

https://rhaas.blogspot.com › 2019 › 01 › how-much-maintenanceworkmem-do-i-need.html

Robert Haas: How Much maintenance_work_mem Do I Need?

maintenance_work_mem controls the amount of memory that the system will allocate in two different cases which are basically unrelated to each other. First, it controls the maximum amount of memory that the system will use when building an index.

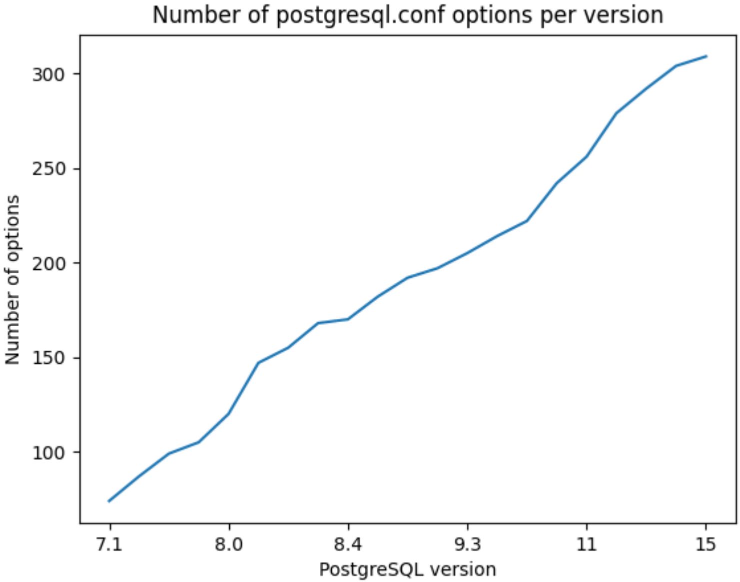
```
#------------------------------------------------------------------------------
# RESOURCE USAGE (except WAL)
#------------------------------------------------------------------------------

# - Memory -

#shared_buffers = 128MB         # min 128kB
                    # (change requires restart)
#huge_pages = try           # on, off, or try
                    # (change requires restart)
#huge_page_size = 0         # zero for system default
                    # (change requires restart)
#temp_buffers = 8MB         # min 800kB
#max_prepared_transactions = 0      # zero disables the feature
                    # (change requires restart)
# Caution: it is not advisable to set max_prepared_transactions nonzero unless
# you actively intend to use prepared transactions.
#work_mem = 4MB             # min 64kB
#hash_mem_multiplier = 2.0      # 1-1000.0 multiplier on hash table work_mem
#maintenance_work_mem = 64MB        # min 1MB
#autovacuum_work_mem = -1       # min 1MB, or -1 to use maintenance_work_mem
#logical_decoding_work_mem = 64MB   # min 64kB
#max_stack_depth = 2MB          # min 100kB
#shared_memory_type = mmap      # the default is the first option
                    # supported by the operating system:
                    #   mmap
                    #   sysv
                    #   windows
                    # (change requires restart)
#dynamic_shared_memory_type = posix # the default is usually the first option
                    # supported by the operating system:
                    #   posix
                    #   sysv
                    #   windows
                    #   mmap
                    # (change requires restart)
```

```
#------------------------------------------------------------------------------
# RESOURCE USAGE (exc
#----------------                                                       ------

# - Memory -

#shared_buffer

#huge_pages =

#huge_page_siz

#temp_buffers =
#max_prepared_t

# Caution: it is                                                          less
# you actively i
#work_mem = 4MB
#hash_mem_multipl                                                        k_mem
#maintenance_work_
#autovacuum_work_me
#logical_decoding_wo
#max_stack_depth = 2M
#shared_memory_type =
                    # s
                    # m
                    #   sysv
                    #   windows
                    # (change requires resta
#dynamic_shared_memory_type = posix # the de                      e first option
                    # supported by the opera
                    #   posix
                    #   sysv
                    #   windows
                    #   mmap
                    # (change requires restart)
```

```
#------------------------------------------------------------------------------
# RESOURCE USAGE (exc
#------------------------------------------------------------------------------

# - Memory -

#shared_buffer

#huge_pages =

#huge_page_siz

#temp_buffers =
#max_prepared_t

# Caution: it is                                                            less
# you actively i
#work_mem = 4MB
#hash_mem_multipl                                                       k_mem
#maintenance_work_
#autovacuum_work_me
#logical_decoding_wo
#max_stack_depth = 2M
#shared_memory_type =
                    # s
                    # m
                    #   sysv
                    #   windows
                    # (change requires resta
#dynamic_shared_memory_type = posix # the de                   e first option
                    # supported by the opera
                    #   posix
                    #   sysv
                    #   windows
                    #   mmap
                    # (change requires restart)
```

Types of configuration

Grand Unified Configuration



Number of postgresql.conf options per version
Number of options
300
250
200
150
100
7.1
8.0
8.4
9.3
11
15
PostgreSQL version

Postgres95 1.01 Distribution

```
/* ----------------
 *  specify the size of buffer pool
 * ----------------
 */
NBuffers = atoi(optarg);
```

[...] one can achieve a substantial portion of the performance gain from configurations generated by ML-based tuning algorithms by setting two knobs according to the DBMS's documentation. These two knobs control the amount of RAM for the buffer pool cache and the size of the redo log file on disk.

Van Aken, D., Yang, D., Brillard, S., Fiorino, A., Zhang, B., Bilien, C. and Pavlo, A., 2021. An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems. Proceedings of the VLDB Endowment, 14(7), pp.1241-1253.

initdb

```
--wal-segsize=SIZE     size of WAL segments,
                       in megabytes
--data-checksums       use data page checksums
```

”Public” code-based configuration

```
--with-blocksize=8
--with-wal-blocksize=8
```

```
// pg_config_manual.h

#define PG_CACHE_LINE_SIZE       128
#define PG_IO_ALIGN_SIZE         4096
#define NUM_SPINLOCK_SEMAPHORES 128
```

”Internal” code-based configuration

```
/*
 * When maintenance_io_concurrency is not saturated,
 * we're prepared to look ahead up to N times
 * that number of block references.
 */
#define XLOGPREFETCHER_DISTANCE_MULTIPLIER 4
```

”Internal” code-based configuration

```
/*
 * Space/time tradeoff parameters: do these need
 * to be user-tunable?
 *
 * To consider truncating the relation, we want
 * there to be at least REL_TRUNCATE_MINIMUM
 * or (relsize / REL_TRUNCATE_FRACTION).
 */
#define REL_TRUNCATE_MINIMUM        1000
#define REL_TRUNCATE_FRACTION         16
```

”Internal” code-based configuration

```
/*
 * Size of the LRU list.
 *
 * XXX: What's a good value? It should be large
 * enough to hold the maximum number of large
 * tables scanned simultaneously.  But a larger
 * value means more traversing of the LRU list
 * when starting a new scan.
 */
#define SYNC_SCAN_NELEM 20
```

”Internal” code-based configuration

```
/* TODO: Unscientifically determined threshold */
LLVMPassManagerBuilderUseInlinerWithThreshold(
        llvm_pmb, 512);
```


Red Hat

How smart PostgreSQL should be?

How many parameters to expose?

Who is the configuration consumer?

”Developer” code-based configuration

```
--enable-profiling
--enable-debug
--enable-coverage
--enable-cassert
```

”Developer” code-based configuration

```
/*
 * This assert is too expensive
 * to have on normally ...
 */
#ifdef CHECK_WRITE_VS_EXTEND
  Assert(blocknum >= mdnblocks(reln, forknum));
#endif
```

”Developer” code-based configuration

```
#ifdef CHECK_DEADLOCK_RISK
/*
 * Issue warning if we already hold a lower-level
 * lock on this object and do not hold a lock of
 * the requested level or higher. This indicates
 * a deadlock-prone coding practice.
```


Red Hat

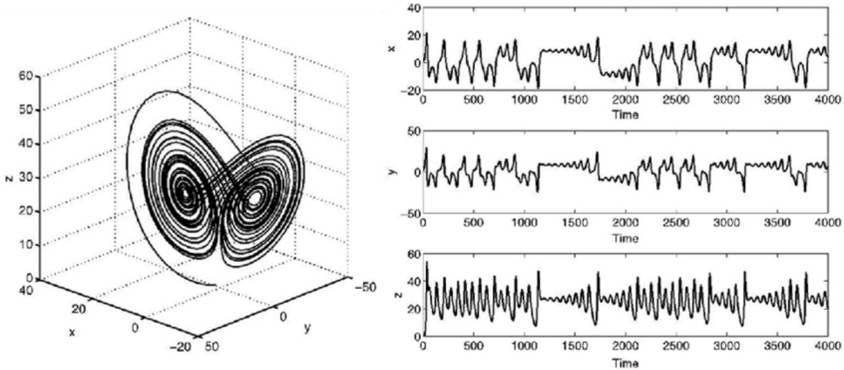
”Developer” code-based configuration

```
/* This is just to allow attaching to startup
 * process with a debugger */
#ifdef XLOG_REPLAY_DELAY
  if (ControlFile->state != DB_SHUTDOWNED)
    pg_usleep(60000000L);
#endif
```

”Developer” code-based configuration

```
/*
 * This helps detect intermittent faults caused
 * by code that reads a cache entry and then
 * performs an action that could invalidate the
 * entry, but rarely actually does so.  This can
 * spot issues that would otherwise only arise
 * with badly timed concurrent DDL, for example.
 */
#ifdef DISCARD_CACHES_ENABLED
```

System model



The phase space plot of the Lorenz attractor,

"Nonlinear time series methods for analyzing behavioral sequences"

Red Hat

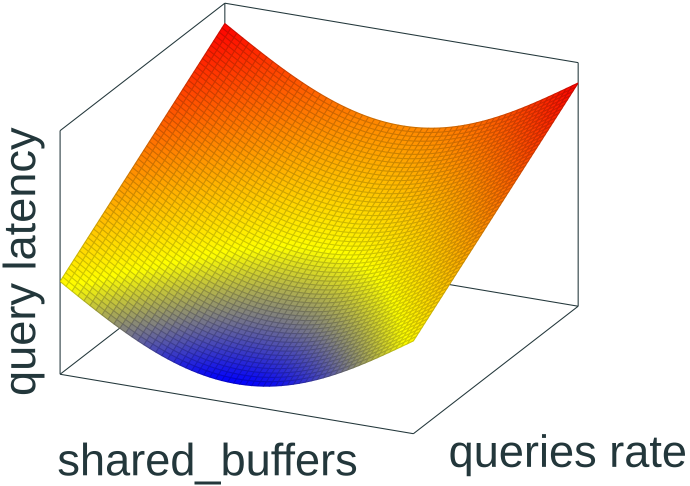
Dimentions?

DB parameters

Hardware resources

Workload parameters

Performance results


query latency
shared_buffers
queries rate

System dynamics

Full system model is unknown

Approximate with bunch of simpler models?

Explore the full model experimentally?

A simpler model: increase the value of X will lead to better performance, but more memory consumption.

A simpler model: more formalized

github.com/le0pard/pgtune

github.com/timescale/timescaledb-tune

github.com/pgconfig/api

github.com/gregs1104/pgtune

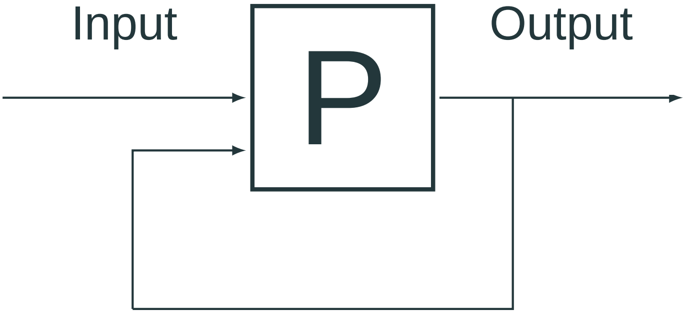
"The Thrilling Adventures of Lovelace and Babbage", Sydney Padua, 2015

Simpler model usually do not include **higher order** parameters interaction.

2^2 factorial experiment

	A	B
(1)	-	-
a	+	-
b	-	+
ab	+	+

Feedback loop



Adaptive Self-Tuning Memory in DB2

Adam J. Storm
IBM Canada
ajstorm@ca.ibm.com

Christian Garcia-Arellano
IBM Canada
cmgarcia@ca.ibm.com

Sam S. Lightstone
IBM Canada
light@ca.ibm.com

Yixin Diao
IBM TJ Watson Research Center
diao@us.ibm.com

M. Surendra
IBM TJ Watson Research Center
suren@us.ibm.com

ABSTRACT

DB2 for Linux, UNIX, and Windows Version 9.1 introduces the Self-Tuning Memory Manager (STMM), which provides adaptive self tuning of both database memory heaps and cumulative database memory allocation. This technology provides state-of-the-art memory tuning combining control theory, runtime simulation modeling, cost-benefit analysis, and operating system resource analysis. In particular, the novel use of cost-benefit analysis and control theory techniques makes STMM a breakthrough technology in database memory management. The cost-benefit analysis allows STMM to tune memory between radically different memory consumers such as compiled statement cache, sort, and buffer pools. These methods allow for the fast convergence of memory settings while also providing stability in the presence of system noise. The tuning model has been found in numerous experiments to tune memory allocation as well as expert human administrators, including OLTP, DSS, and mixed environments. We believe this is the first known use of cost-benefit analysis and control theory in database memory tuning across heterogeneous memory consumers.

1. INTRODUCTION

be relieved of the need to invest time in understanding how the database uses memory before tuning can begin.

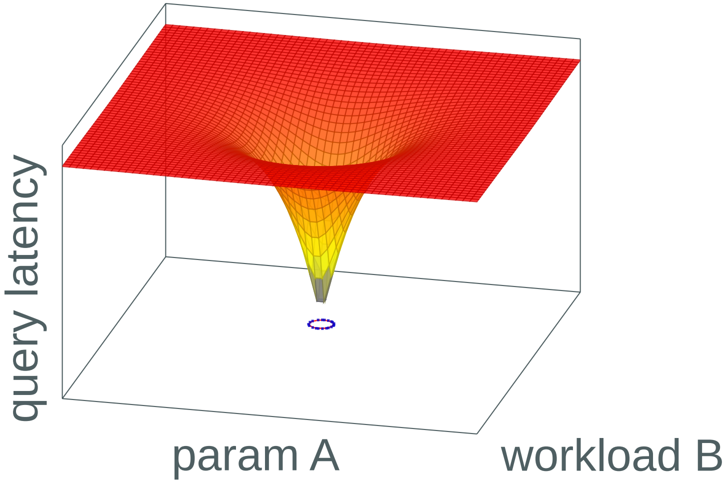
2. Uncertain memory requirements for a given workload – In some cases, even experienced DBAs can find it difficult to tune a database's memory because the workload characteristics are unknown. With the introduction of this new feature, the system will now be able to continuously monitor database memory usage and tune when necessary to optimize performance based on the workload characteristics. As a result, the user will require no knowledge of their workload for the memory to be tuned well.

3. Changing workload behavior – For many industrial workloads, no single memory configuration can provide optimal performance because, at different points in time, the workload can exhibit dramatically different memory demands. If STMM is running and the workload's memory demands shift, the system will recognize the changing needs for memory and adapt the memory allocation accordingly. As a result, the user will rarely (if ever) need to manually change the affected memory configuration parameters to enhance performance.

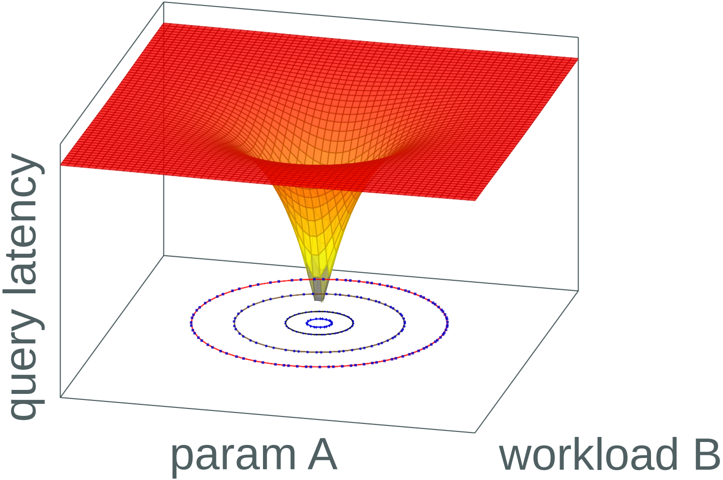
4. Performance tuning is time-consuming – Tuning a database's memory to achieve high levels of performance is extremely costly

System stability

Is highest performance enough?



Huynh, A., Chaudhari, H.A., Terzi, E. and Athanassoulis, M., 2021. Endure: A Robust Tuning Paradigm for LSM Trees Under Workload Uncertainty. arXiv preprint arXiv:2110.13801.



Huynh, A., Chaudhari, H.A., Terzi, E. and Athanassoulis, M., 2021. Endure: A Robust Tuning Paradigm for LSM Trees Under Workload Uncertainty. arXiv preprint arXiv:2110.13801.

Questions?

@erthalion@fosstodon.org

ddolgov at redhat dot com

Red Hat