initMAX

Webinar

PostgreSQL Performance Tuning

all your microphones are muted

ask your questions in Q&A, not in the Chat

use Chat for discussion, networking or applause

1

The Performance Tuning Reality

initMAX

PostgreSQL Performance Tuning

This is the reality

Sysadmin tuning gains: 10-30% improvement

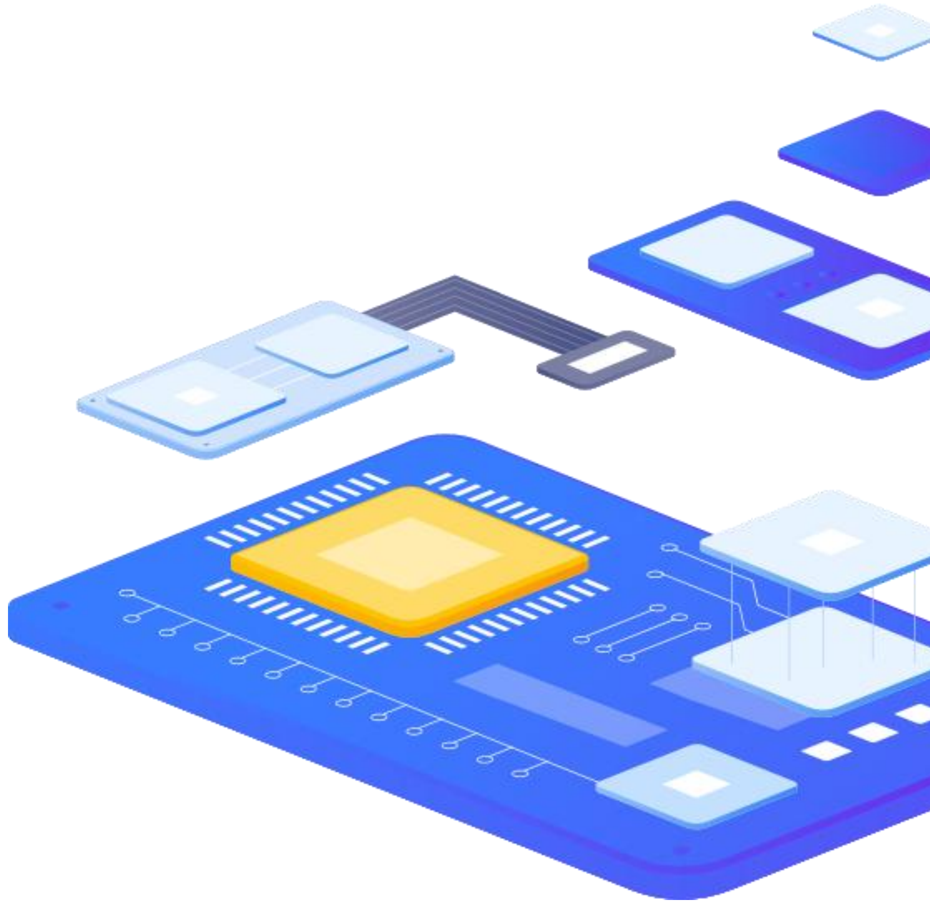
- OS optimization
- PostgreSQL parameter tuning
- Stability and monitoring focus

Developer tuning gains: 100-1000% improvement

- Missing indexes
- Query optimization
- Data model improvements

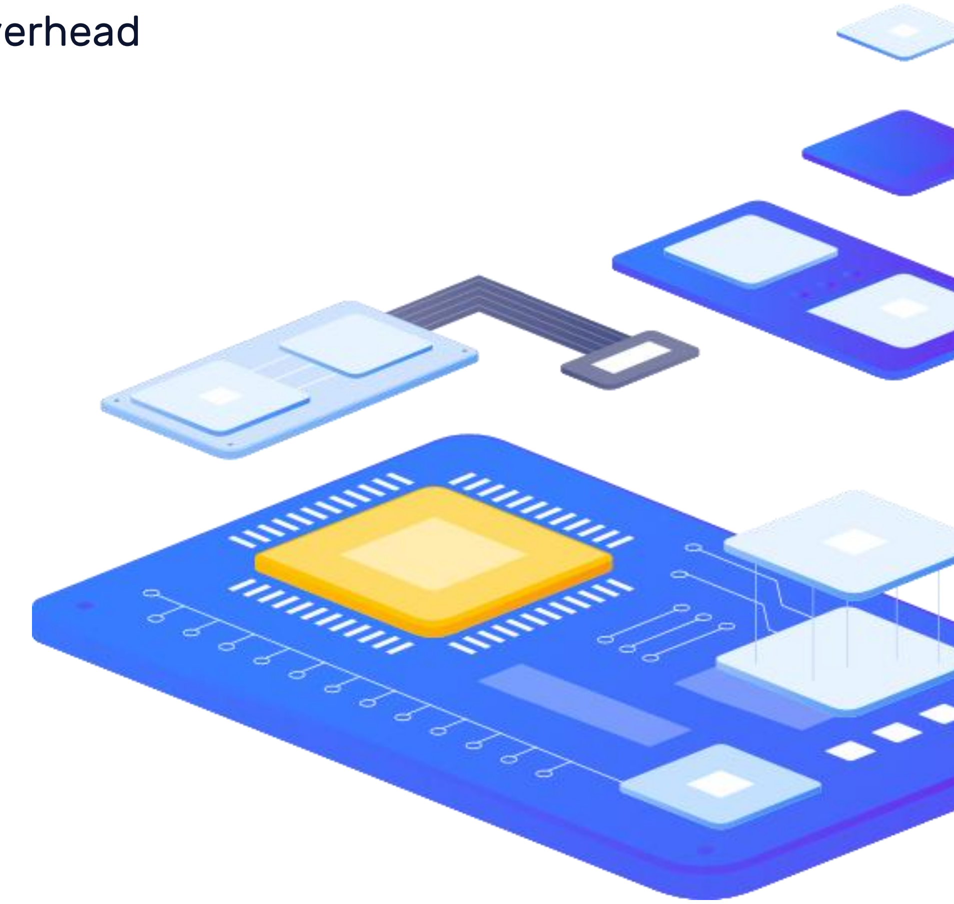
When Your Database is Down, Performance is Zero

- **Streaming replication:** 2-5% primary overhead
- **Synchronous replication:** 20-30% latency increase
- **Backup strategies:** pg_basebackup vs. pgbackrest
- **Connection failover:** Use connection pooling with health checks
- **Use proven solutions for HA:** Patroni



PostgreSQL Performance Anti-Patterns

- **Over-tuning shared_buffers:** >40% RAM causes OS cache competition
- **Ignoring connection limits:** >1000 connections = context switching overhead
- **Default autovacuum settings:** Designed for HDDs, not SSDs
- **Disabling fsync:** Never do this in production
- **Single large database:** Consider multiple smaller databases
- **Forgetting to ANALYZE:** Statistics drive query planning

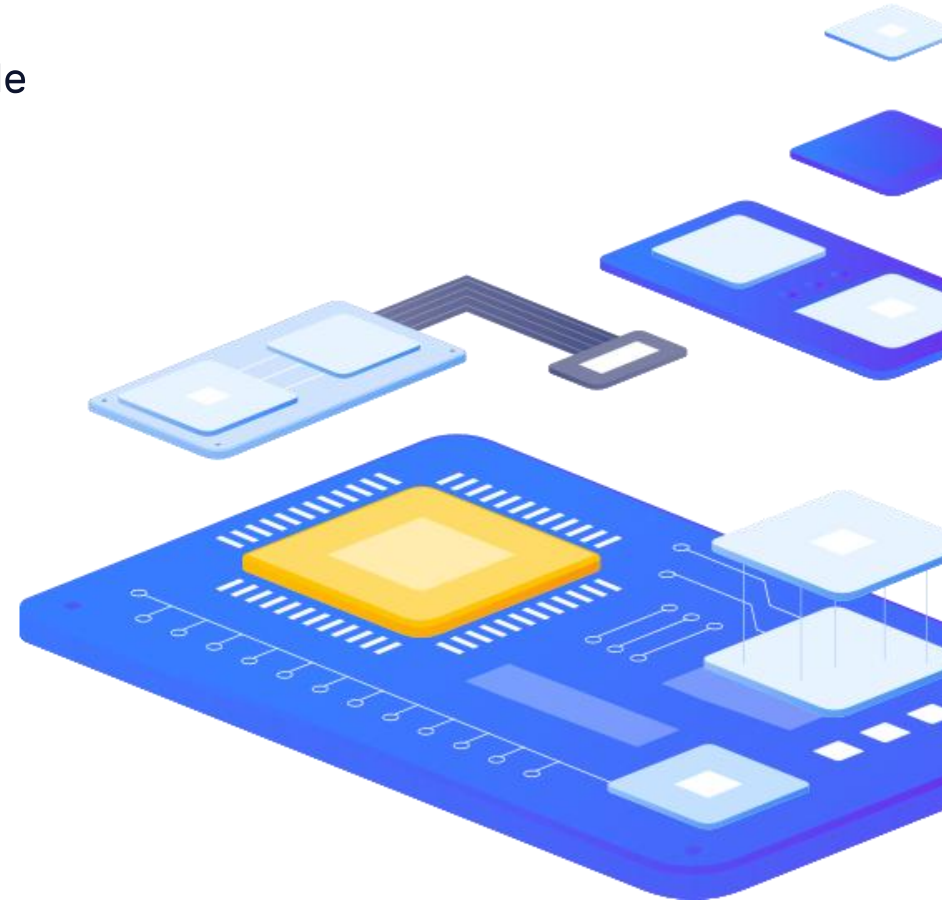


2

Sysadmin tuning options

Our Performance Tuning Scope

- **Hardware:** 8 CPU cores, 16-64GB RAM, SSD storage
- **Software:** PostgreSQL 15+, Linux VMs, On-premises
- **Workload:** Mixed OLTP/OLAP, connection pooling required
- **Monitoring:** Zabbix, pgwatch, pg_stat_statements
- **Goal:** Maximize stability while optimizing performance



Expected Performance Improvements

Typical improvements from sysadmin tuning

- **Query response time:** 15-25% faster
- **Throughput:** 20-30% increase
- **I/O reduction:** 30-40% with WAL compression
- **Connection overhead:** 60-70% reduction with pooling

OLTP vs OLAP - Different Tuning Strategies

Characteristic	OLTP	OLAP
Queries	Short, frequent	Long, complex
work_mem	64-128MB	512MB-2GB
Connections	Many (pooled)	Few (direct)
Indexes	Many small	Few large
Autovacuum	Aggressive	Moderate
Partitioning	By date/ID	By time/region

initMAX

Essential OS-Level Optimizations

Memory management

- vm.swappiness = 1
- vm.dirty_ratio = 10
- vm.dirty_background_ratio = 5
- vm.overcommit_memory = 2
- vm.nr_hugepages = 4096

Impact

15-20% I/O performance improvement

- Reduces swap usage
- Optimizes dirty page handling
- SSD-optimized I/O scheduling
- Eliminates unnecessary metadata writes

```
# Filesystem mount options
/dev/sda1 /var/lib/postgresql ext4 noatime,discard,barrier=0
```

```
# I/O scheduler for SSD
echo deadline > /sys/block/sda/queue/scheduler
```

Huge Pages - Essential for Large shared_buffers

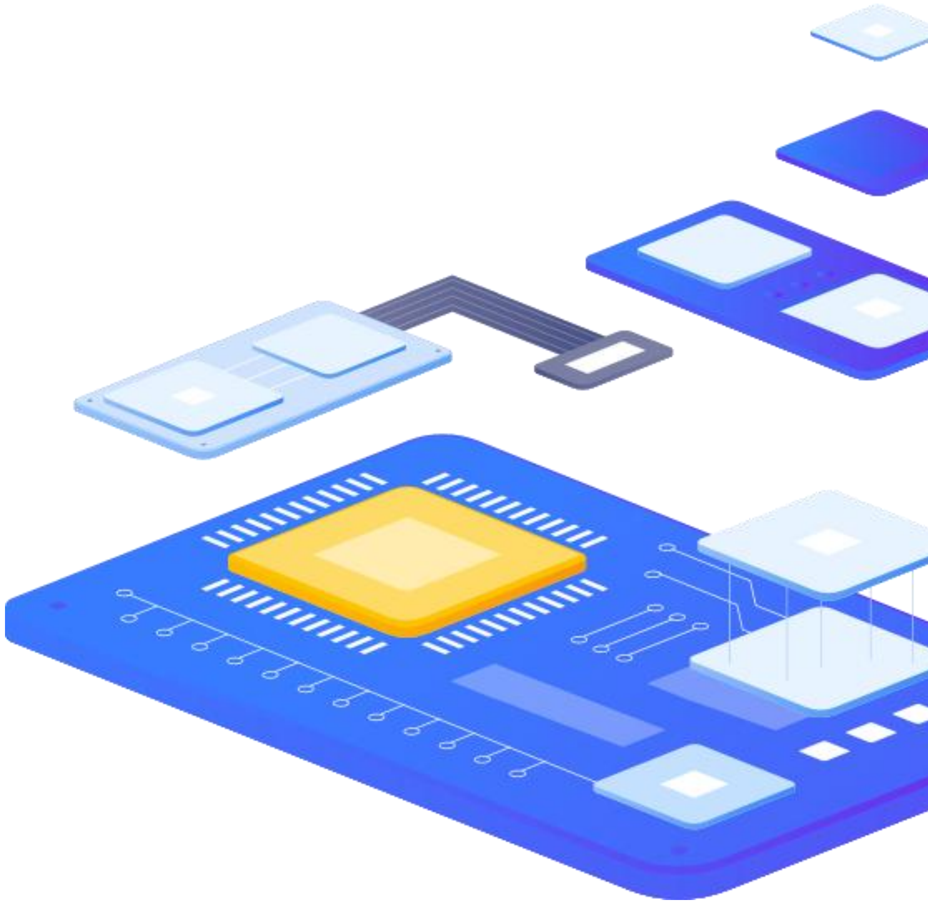
Benefits

- Reduced TLB misses
- TLB (Translation Lookaside Buffer) holds about ~1500 entries
- Lower memory overhead
- Better memory locality

Requirement: Mandatory for shared_buffers > 8GB

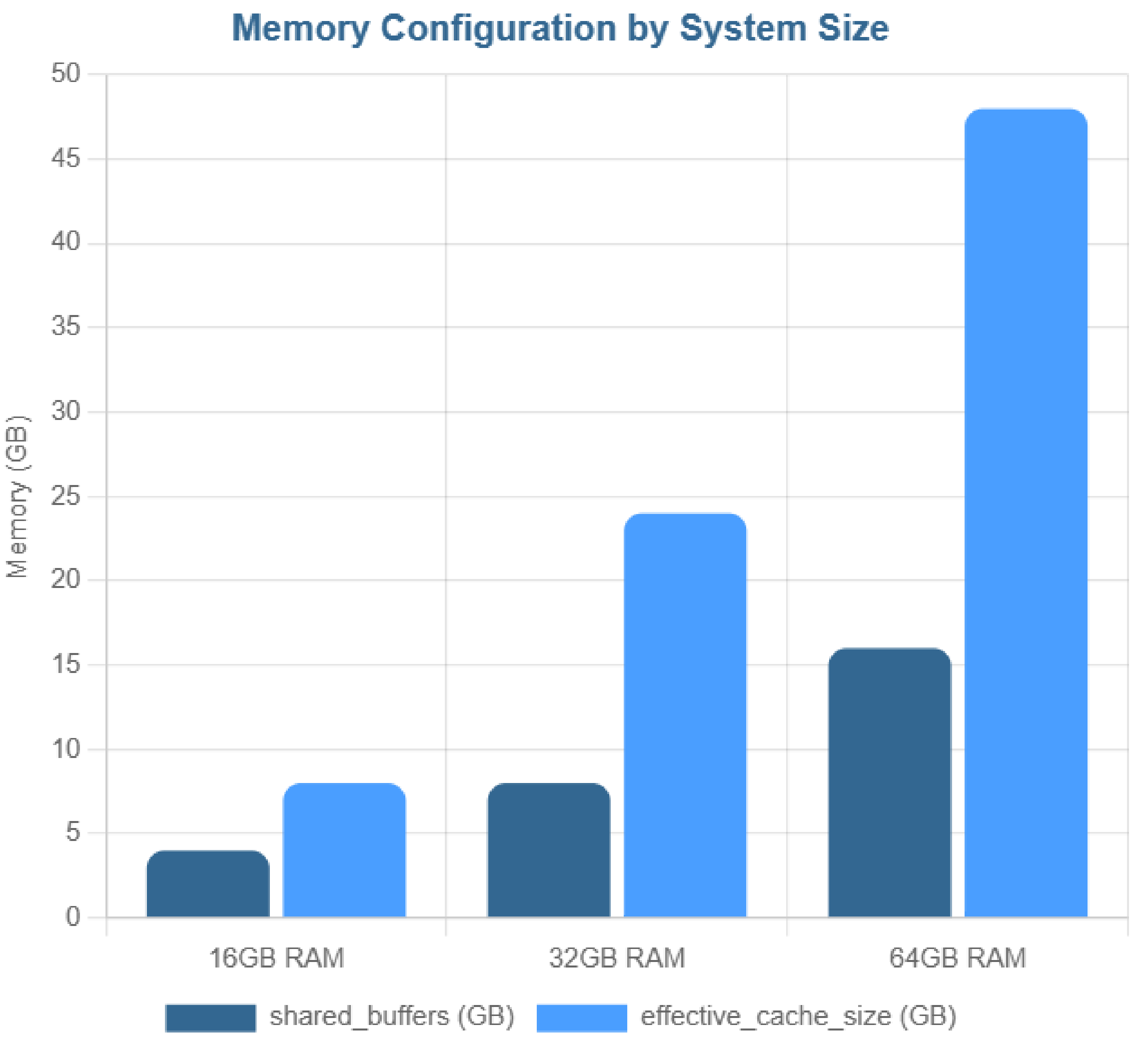
Calculate required huge pages (8GB shared_buffers)

8192MB / 2MB(default) = 4096 pages



PostgreSQL Memory Parameters

Parameter/RAM	16GB	32GB	64GB
shared_buffers	4GB	8GB	16GB
work_mem	64MB	96MB	128MB
maintenance_work_mem	1GB	2GB	4GB
effective_cache_size	8GB	24GB	48GB



Write-Ahead Log Performance Tuning

```
# Compression options in PostgreSQL 15+
wal_compression = zstd

# Buffer and size configuration
wal_buffers = 64MB
max_wal_size = 40GB
checkpoint_timeout = 60min
checkpoint_completion_target = 0.95
full_page_write = on # Torn page corruption prevention
```

Impact: 30% I/O reduction with zstd compression

Compression Comparison

- zstd: Best compression ratio
- lz4: Fastest compression
- pglz: Legacy default

Balancing Performance vs. Recovery Time

```
checkpoint_timeout = 60min                    # Longer = less I/O spikes
max_wal_size = 40GB                           # Depends on write volume
checkpoint_completion_target = 0.95           # Spread I/O over time

# Monitor checkpoint performance(17+)
SELECT num_timed, num_requested, write_time, sync_time FROM pg_stat_checkpointer;
```

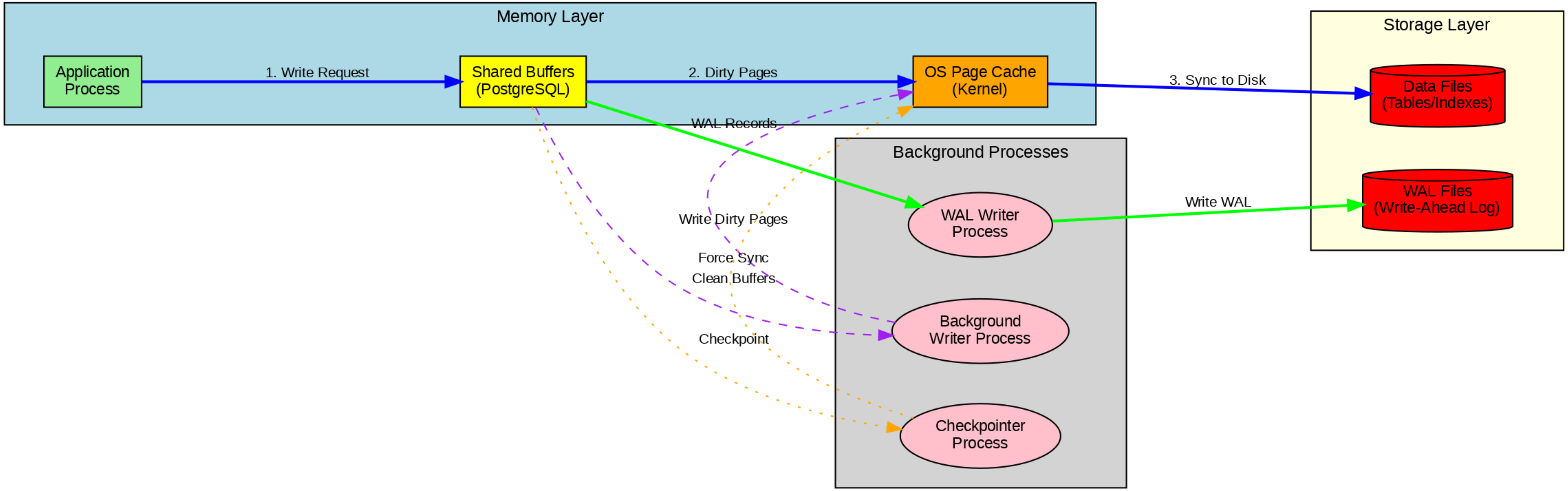
Trade-off - Longer checkpoints = longer recovery time

Tuning Goals

- Spread I/O evenly
- Avoid I/O spikes
- Balance with recovery requirements
- Monitor checkpoint frequency

PostgreSQL Performance Tuning

How PostgreSQL writes data



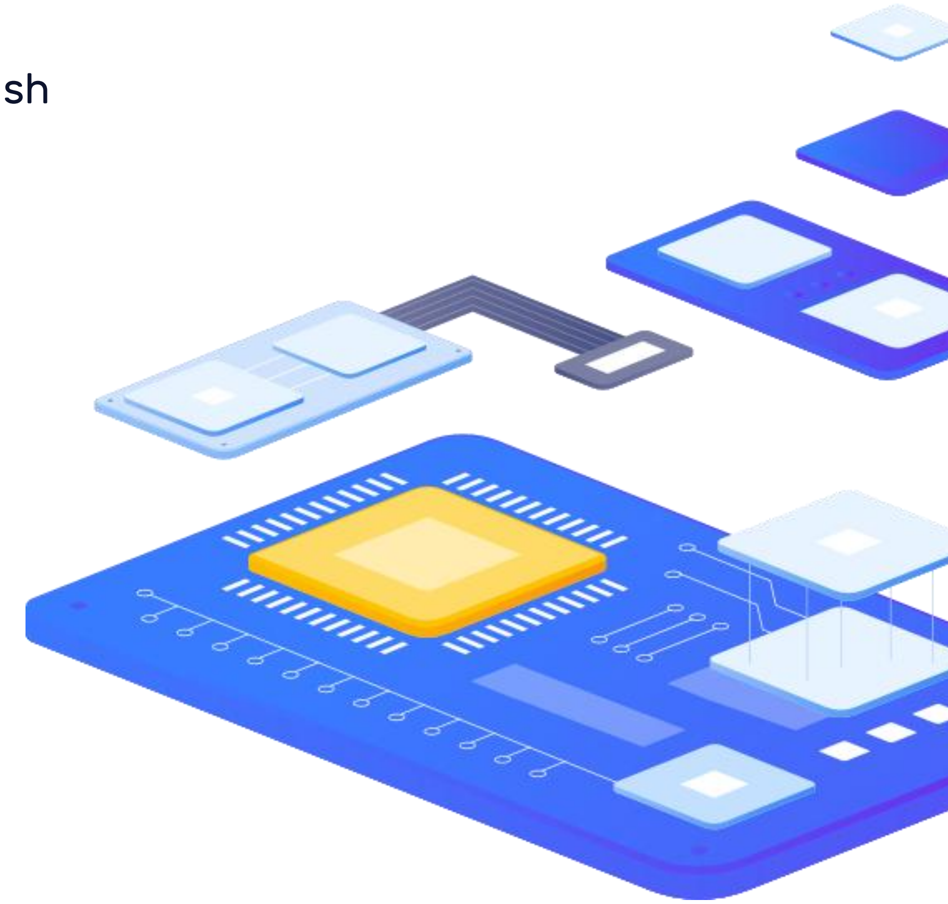
initMAX

PostgreSQL Dirty Page Flushing

Understanding Dirty Pages

Dirty pages are modified data pages in PostgreSQL's shared_buffers that haven't been written to disk yet. PostgreSQL uses three mechanisms to flush these dirty pages:

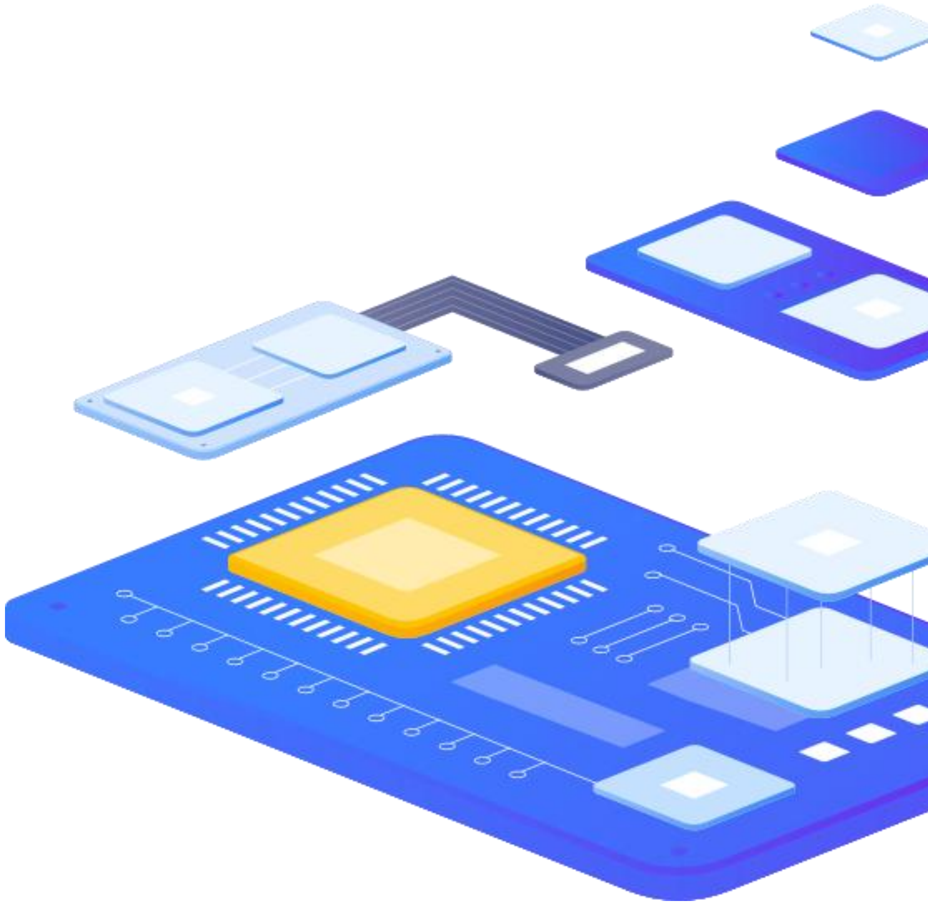
- **Background Writer (BGWriter)** - Continuous gentle flushing
- **Checkpointer** - Periodic forced synchronization
- **Backend Processes** - Emergency direct flushing (performance killer!)
    - **No clean pages available** in shared_buffers for new data
    - **BGWriter can't keep up** with dirty page generation
    - **Shared_buffers is too small** for the workload



PostgreSQL Dirty Page Flushing

Fixing Backend Direct Writes

- Increase shared_buffers (if <25% of RAM)
  - shared_buffers = 8GB
- Make BGWriter more aggressive
  - bgwriter_delay = 100ms
  - bgwriter_lru_maxpages = 200
  - bgwriter_lru_multiplier = 4.0
- Tune checkpoints to be less frequent but more spread out
  - checkpoint_timeout = 15min
  - checkpoint_completion_target = 0.9
  - max_wal_size = 10GB



PostgreSQL Dirty Page Flushing

Key Takeaways for Sysadmins

- **BGWriter is your friend** - Tune it to be more aggressive on modern hardware
- **Backend writes are the enemy** - They indicate undersized shared_buffers or overwhelmed BGWriter
- **Checkpoints should be infrequent but spread out** - Long intervals with gradual completion
- **Monitor the ratios** - Backend write percentage is the most critical metric
- **SSD changes everything** - Default settings assume spinning disks

The goal is to have BGWriter handle 90%+ of dirty page flushing, checkpointer handle periodic durability, and backend processes never have to flush pages directly.

Aggressive Autovacuum for SSD Performance

```
# For 8 CPU system
autovacuum_max_workers = 6

# 10x default for SSD
autovacuum_vacuum_cost_limit = 2000
autovacuum_vacuum_cost_delay = 10ms

# 5% instead of 20%
autovacuum_vacuum_scale_factor = 0.05

# Table-specific tuning for busy tables
ALTER TABLE busy_table SET ( autovacuum_vacuum_scale_factor = 0.01 );
```

Modern reality

Default settings designed for spinning disks

SSD Advantages

- Higher IOPS capacity
- Better random access performance
- Can handle aggressive vacuuming

PostgreSQL Performance Tuning

initMAX

HOT Updates - Heap-Only Tuples

Normal Update Process

- Create new row version (tuple)
- Update ALL indexes pointing to old tuple
- Mark old tuple as dead
- Eventually VACUUM removes dead tuple

Result: Expensive index maintenance, more I/O

HOT Update Process

- Create new row version in SAME page
- NO index updates needed
- Chain old and new tuples together
- Much faster, less bloat

Key Requirement: New tuple must fit on same page AND no indexed columns changed

HOT Update: When PostgreSQL can update a row without updating indexes

initMAX

Fillfactor - Reserving Space for Updates

What is Fillfactor?

- Fillfactor: Percentage of page filled during initial INSERT
- Default: 100% (fill pages completely)
- Range: 10-100%

Trade-offs:

- Lower fillfactor: More HOT updates, less storage efficiency
- Higher fillfactor: Better storage density, fewer HOT updates

Workload-based Recommendations:

- Read-heavy (OLAP): fillfactor = 100 (maximize storage efficiency)
- Balanced OLTP: fillfactor = 85-90
- Update-heavy: fillfactor = 70-80

Test different fillfactor values with your actual workload to find the optimal balance between storage and performance.

Verify Your Tuning Results

System checks

- Huge pages allocated and used
- I/O scheduler set to deadline/noop
- Swap disabled or minimal
- Overcommit memory disabled

PostgreSQL checks

- Cache hit ratio > 95%
- Connection utilization < 80%
- Checkpoint frequency reasonable
- No excessive temp file usage
- Autovacuum keeping up with changes

3

Developer tuning options

Recognizing Application-Level Issues

Red flags requiring developer intervention

- Sequential scans on large tables
- Queries with temp file usage > work_mem
- N+1 query patterns
- Missing foreign key indexes or generally missing indexes
- Inefficient data types (char vs varchar)
- uuid as PK (bad for indexing)

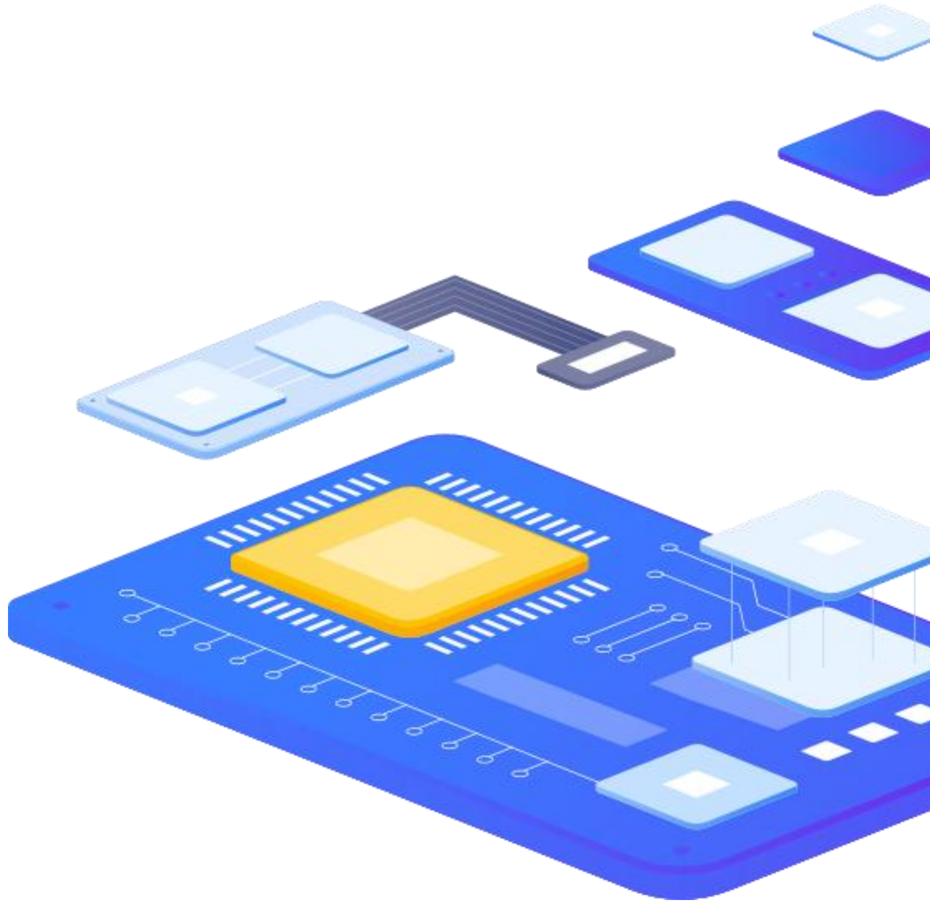
```
-- Bad: N+1 query pattern
SELECT * FROM orders WHERE customer_id = ?;  -- 1000 times

-- Good: Single query with JOIN
SELECT * FROM orders o JOIN customers c ON o.customer_id = c.id;
```

The fastest query is the one you don't have to execute

Application Logic and Caching Strategies

- Smart Data Structures in Memory
- Precomputed Values and Materialized Data
- Batch Operations Instead of Individual Queries
- Application-Level Caching (Redis/Memcached)
- Prepared statements
- Read Replicas for Query Separation
- Lazy Loading

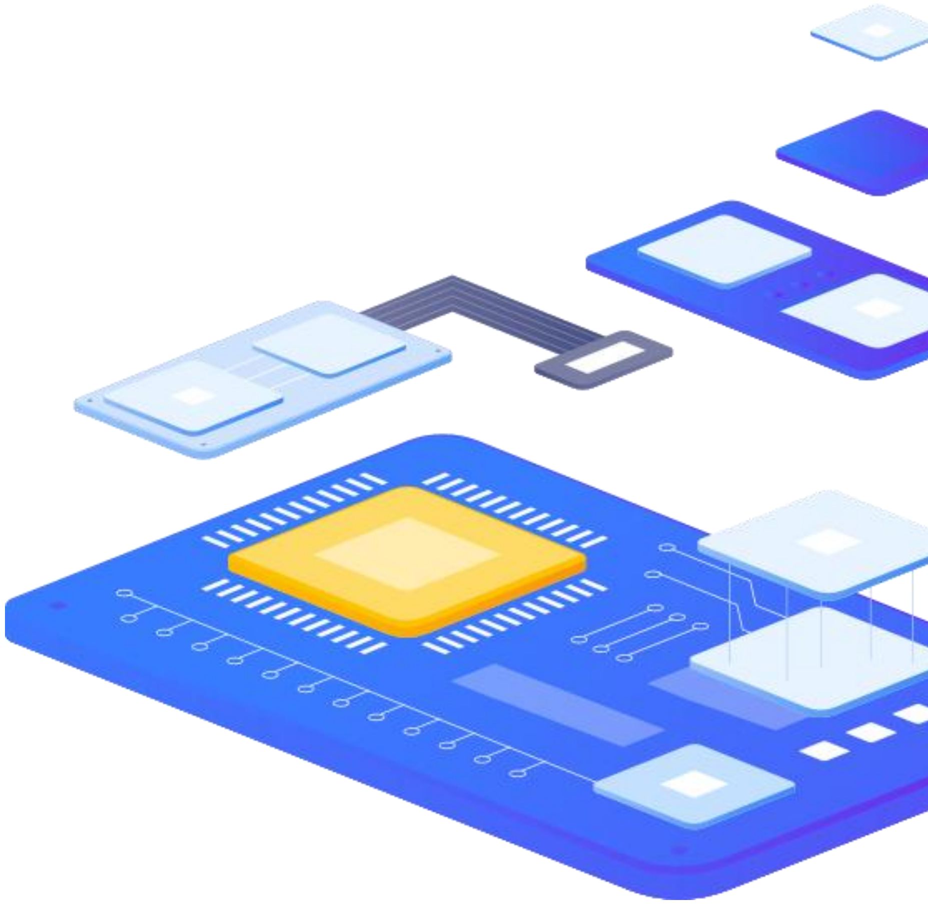


4

Monitoring / troubleshooting

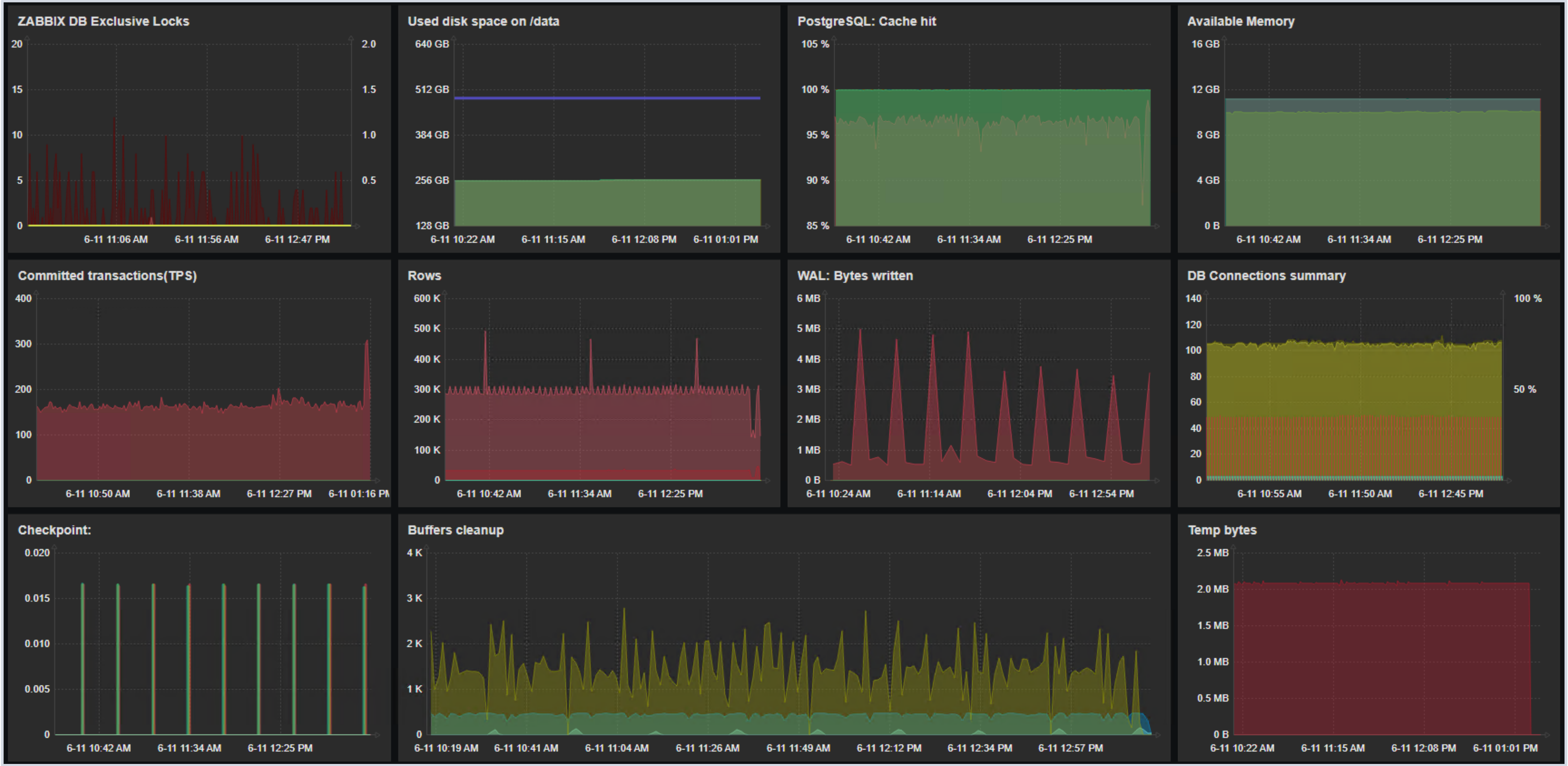
Monitoring essentials

- System metrics like CPU, RAM, HDD, etc.
- Number of connections
- Locking
- XMIN horizon age
- BgWriter / Checkpointer
- WAL rate
- Temp bytes
- Cache hit
- TPS / QPS
- Replication lag / inactive slots



PostgreSQL Performance Tuning

Zabbix



PostgreSQL Performance Tuning

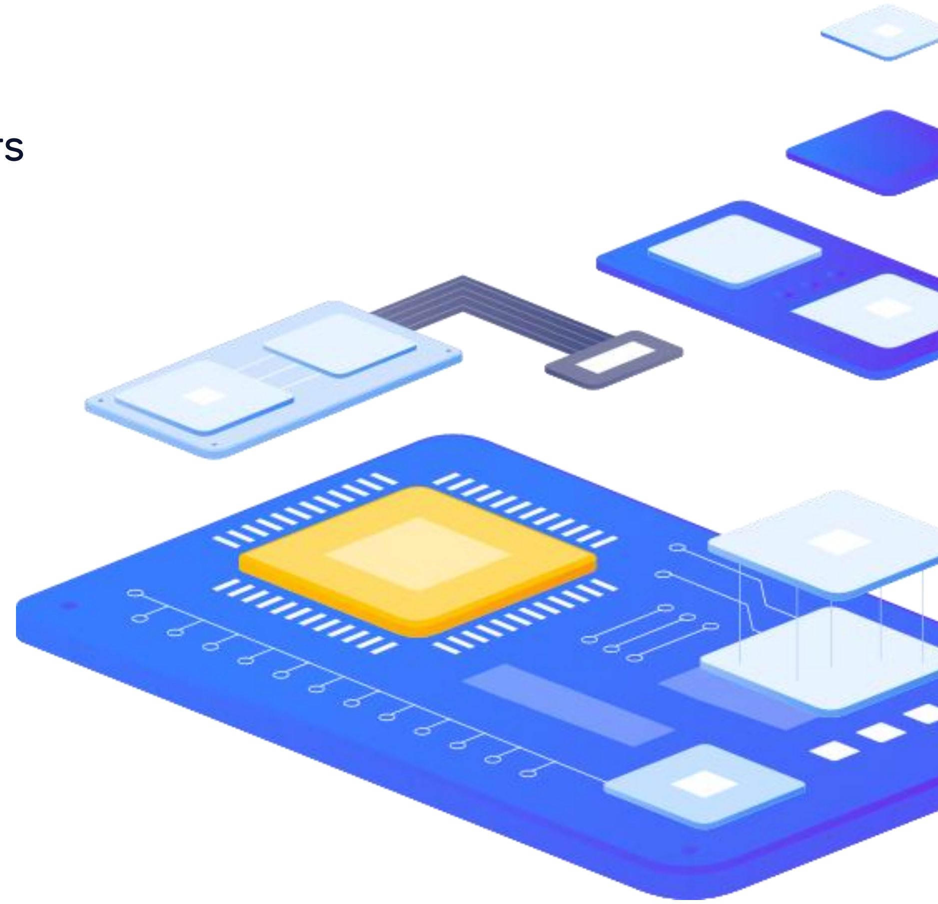
PgWatch

5

Key Takeaways for PostgreSQL Performance

Key Takeaways for PostgreSQL Performance

- **Memory is king:** 25% RAM for shared_buffers, tune work_mem
- **Connection pooling is mandatory:** Use PgBouncer or similar
- **Modern hardware needs modern settings:** SSD-optimized parameters
- **Monitor everything:** pg_stat_statements, system metrics, locks
- **Stability first:** Performance is irrelevant if the database isn't up.
- **Know when to escalate:** Some issues require developer intervention



initMAX

Questions?

PostgreSQL Performance Tuning

initMAX

Certified training and Support

Basic certified courses

SQL Basics

This training has been designed for people who want to get familiar with SQL. You will learn how SQL works and how to write proper SQL statements using practical examples that will be useful for your daily work.

4 DAYS

€ 2,000

Excluding VAT

Requirements: None
Available online: Yes
Certification: Yes

Course dates:
On request | 13.–16. 1. 25 | 12.–15. 5. 25 | 3.–6. 11. 25

REGISTER

More info and dates

Introduction to PostgreSQL

This workshop has been designed for people who want to get familiar with SQL and PostgreSQL. You will learn how to use PostgreSQL and how to write proper SQL statements.

4 DAYS

€ 2,000

Excluding VAT

Requirements: None
Available online: Yes
Certification: Yes

Course dates:
On request | 23.–26. 9. 24 | 27.–30. 1. 25 | 26.–29. 5. 25 | 24.–27. 11. 25

REGISTER

More info and dates

Advanced certified courses

PostgreSQL Professional

This course provides a deep insight into advanced PostgreSQL topics like indexing, storage parameters, optimization, replication, monitoring and many more.

3 DAYS

€ 1,500

Excluding VAT

Requirements: None
Available online: Yes
Certification: Yes

Course dates:
On request | 7.–9. 10. 24 | 10.–12. 2. 25 | 9.–11. 6. 25 | 8.–10. 12. 25

REGISTER

More info and dates

PostgreSQL – administration and performance tuning

This course is perfectly suitable for database administrators and sysadmins, dealing with topics related to administration and performance tuning.

4 DAYS

€ 2,000

Excluding VAT

Requirements: None
Available online: Yes
Certification: Yes

Course dates:
On request | 21.–24. 10. 24 | 24.–27. 2. 25 | 23.–26. 6. 25

REGISTER

More info and dates

PostgreSQL – High Availability & Patroni

This course is intended for PostgreSQL users who are interested in fully automating high availability operations. The course first gives an overview of the general high availability landscape in PostgreSQL clusters and then focuses on installation, configuration and fully automated operation of very popular (open source) cluster manager "Patroni".

3 DAYS

€ 1,500

Excluding VAT

Requirements: advanced knowledge of PostgreSQL and OS Linux
Available online: Yes
Certification: Yes

Course dates:
On request | 4.–6. 11. 24 | 10.–12. 3. 25 | 8.–10. 9. 25

REGISTER

More info and dates

Certified training and Support

Expert certified courses

PostgreSQL in Kubernetes

This course provides an introduction to Kubernetes itself and to Kubernetes resources, which are needed in order to manage PostgreSQL.

3 DAYS

€ 1,500

Excluding VAT

Requirements: None
Available online: Yes
Certification: Yes

Course dates:
On request | 18.–20. 11. 24 | 24.–26. 3. 25 | 22.–24. 9. 25

REGISTER

More info and dates

Introduction to PostGIS

This course provides an introduction to PostGIS and its most important features and capabilities. Along with gaining theoretical knowledge, participants will work with real-world datasets to deepen and reinforce their practical skills.

3 DAYS

€ 1,500

Excluding VAT

Requirements: advanced knowledge of PostgreSQL
Available online: Yes
Certification: Yes

Course dates:
On request | 2.–4. 12. 24 | 7.–9. 4. 25 | 6.–8. 10. 25

REGISTER

More info and dates

PostgreSQL – Migration tutorial

This training has been designed for people who want to switch to PostgreSQL.

4 DAYS

€ 2,000

Excluding VAT

Requirements: advanced knowledge of PostgreSQL
Available online: Yes
Certification: Yes

Course dates:
On request | 16.–19. 12. 24 | 22.–25. 4. 25 | 20.–23. 10. 25

REGISTER

More info and dates

initMAX

PostgreSQL Performance Tuning

Certified training and Support

DBA – PostgreSQL

PostgreSQL is a powerful open-source object-relational database management system (ORDBMS). It is used for application development, data warehousing, analysis and other data-intensive tasks. Key features of PostgreSQL include a powerful engine, support for advanced data types and indexing methods, and support for stored procedures and triggers written in various programming languages, including PL/pgSQL, Tcl, and Python. Furthermore, PostgreSQL supports multiversion concurrency control (MVCC), allowing multiple users to access the same data simultaneously without conflicts, and offers robust data integrity and security support.

I AM INTERESTED IN THIS SERVICE →

Precise database management backed by experience

Reliable monitoring and notification system

Stability, availability and scalability

initMAX

Contact us:

Phone:	+420 800 244 442
Web:	https://www.initmax.cz
Email:	tomas.hermanek@initmax.cz
LinkedIn:	https://www.linkedin.com/company/initmax
Twitter:	https://twitter.com/initmax
Tomáš Heřmánek:	+420 732 447 184