WordCamp Sofia 2026

Stress testing and scaling WordPress on a $12 VPS

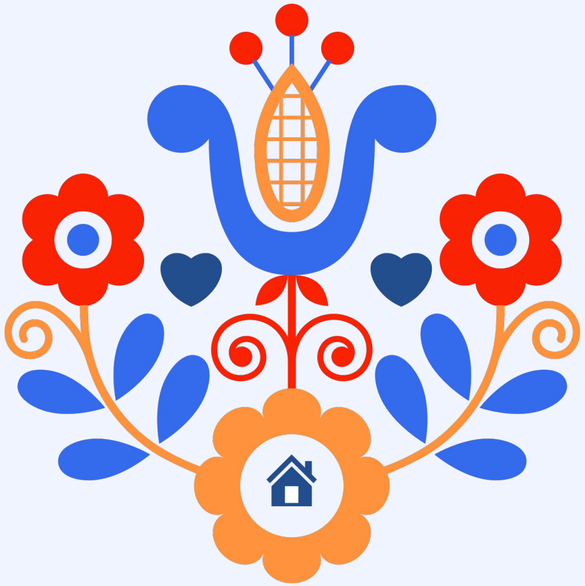
Nedko Hristov
señor DevOps @ Nemetschek Bulgaria



WordCamp Sofia 2026



Hi. I'm Nedko.





WordCamp Sofia 2026





WordCamp Sofia 2026

Senior DevOps engineer @ Nemetschek Bulgaria by day, home labber and WordPress geek by night, tech speaker



WordCamp Sofia 2026

It is Lato. Lato was created by:

Polish designer Łukasz Dziedzic





“ Talk is cheap. Show me the code. ”

Linus Torvalds

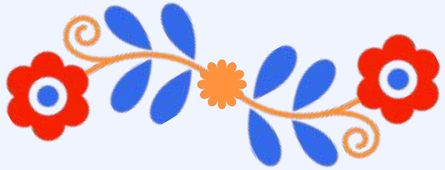
WordCamp Sofia 2026

The code is yours

Like WordPress, this entire presentation - including all code, images, dashboards, and parsers - is licensed under the GNU GPL. It is free to use, modify, study and distribute.

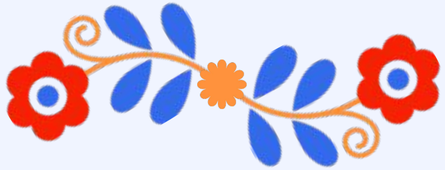
WordCamp Sofia 2026

nedko.info/wcsf2026



WordCamp Sofia 2026

At the end of the presentation you'll know how to reach those numbers



37× faster response

~3,000 ms → ~80 ms · same $12 VPS · zero cost

DB load −75%

20+ threads running → 3

200× faster for global visitors

~3,000 ms → ~15 ms · Cloudflare free tier

750× faster · PHP = 0

~3,000 ms → ~4 ms · Simply Static · pure Nginx

WordCamp Sofia 2026



Why WordPress?



WordCamp Sofia 2026

Meet marvin

He is cheap, but hardworking lives only on 1GB RAM, 1vCPU and 25GB disk



WordCamp Sofia 2026

Why $12?

- 1 vCPU
- **2 GB RAM**
- 50 GB SxSD
- 2 TB Transfer



WordCamp Sofia 2026

All the tests has been done on a **2GB docker RAM limit** to mimic the VPS spec



WordCamp Sofia 2026

WordPress data

- 2,500 posts
- 500 WooCommerce products
- ~6,000 revisions in DB
- Realistic bloat: transients, orphaned meta



What Is Virtual User?

Key concept: what is a Virtual User?

1 VU = 1 browser-like loop, running forever

```
VU #1:  GET /  → wait 1s → GET /?p=1 → wait 1s → repeat …
VU #2:  GET /  → wait 1s → GET /?p=2 → wait 1s → repeat …
   ⋮
VU #50: GET /  → wait 1s → GET /?p=5 → wait 1s → repeat …
```

50 VUs = **50 open connections** = **~50 req/s sustained**

Real-world equivalent *(think time = 15–30× longer in real life)*

k6 VUs	≈ Real simultaneous visitors
10	~150 – 300
50	**~750 – 1,500**
100	~1,500 – 3,000

k6 is deliberately **aggressive** — spikes are visible, crashes happen fast.

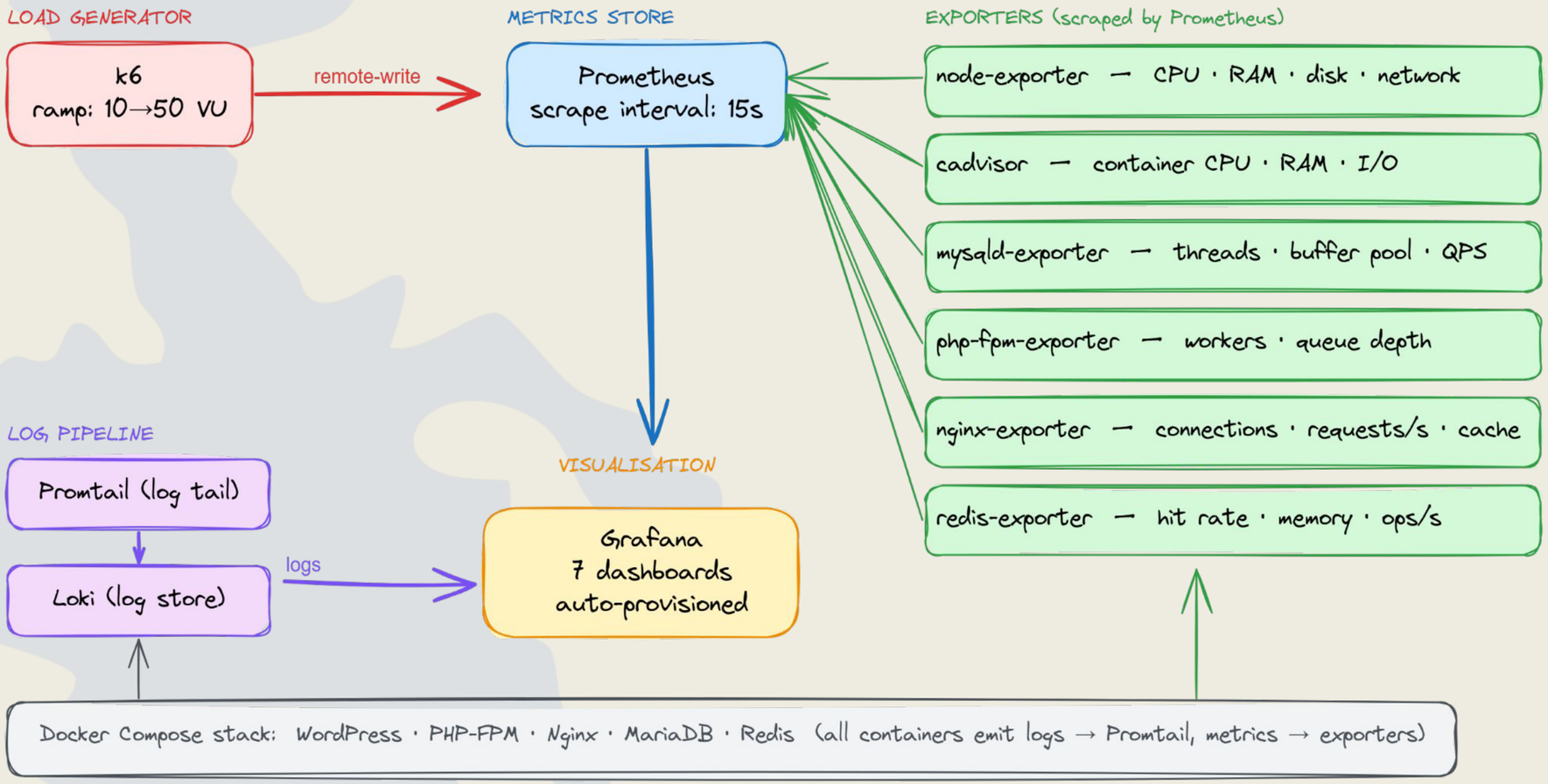
Key metric definitions

Term	What it actually means
Virtual Users	Concurrent browser-like loops running continuously - not unique visitors
requests/s	Total HTTP responses served per second across all VUs
p95 latency	95% of requests were *faster* than this - 5% were slower
p99 latency	The near-worst case - 1% of requests were slower than this
FPM workers	PHP processes *currently executing code* right now
FPM queue depth	Requests waiting for a free PHP worker - 0 = healthy, >0 = trouble
DB threads running	MySQL queries *actively executing* - not just connected
Redis hit rate	% of WordPress DB calls answered from RAM instead of MySQL
Cache offload	% of HTTP requests served by nginx without PHP running at all

WordCamp Sofia 2026

Observability stack

Every number on screen is a real metric from a real process. No synthetic benchmarks.

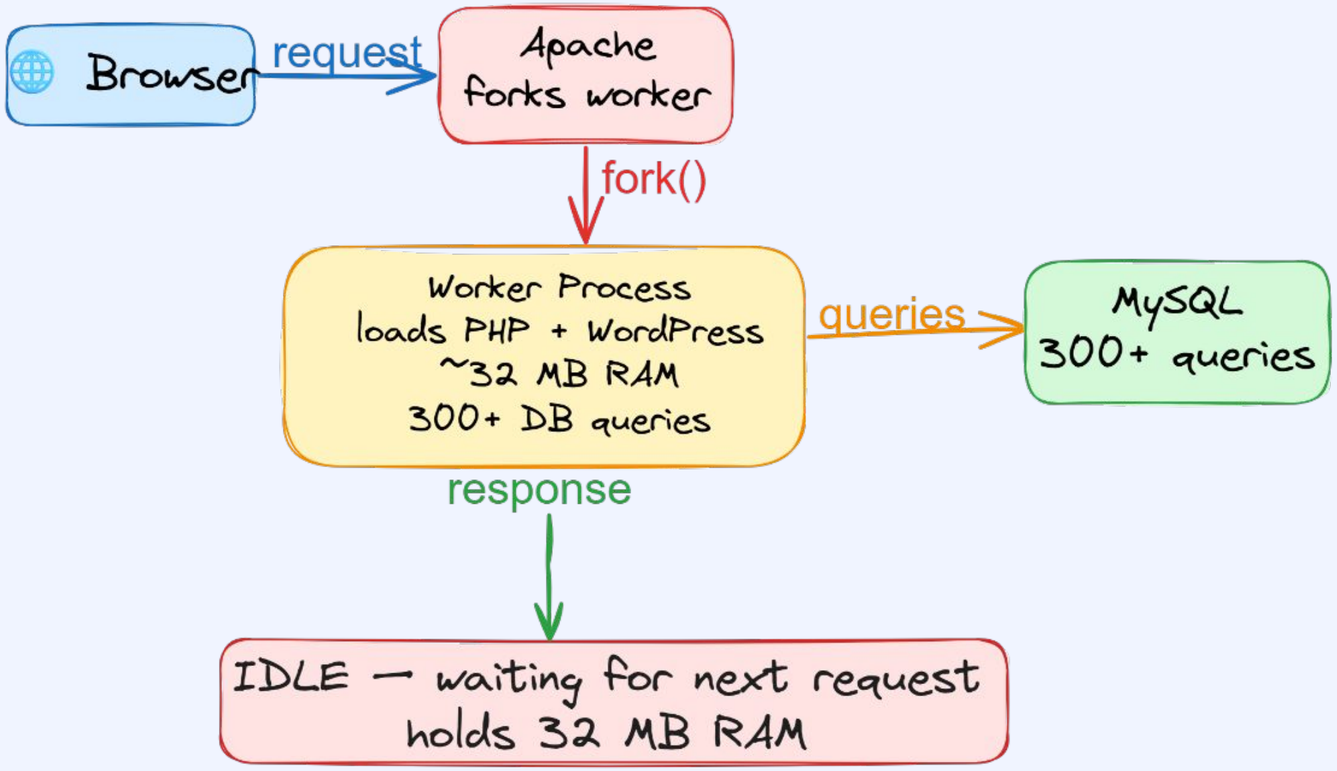


WordCamp Sofia 2026

Level 0 — Apache + mod_php

Level 0 — How Apache prefork works

Apache Prefork — 1 OS Process Per Connection



THE PROBLEM

- 2 GB RAM on VPS
- OS + DB + Redis = ~512 MB overhead
- 32 MB per idle WordPress worker
- (2048 − 512) ÷ 32 = 47 workers max

W #1 W #2 W #3 … W #47

↑ all 47 workers IDLE = 1.5 GB RAM consumed

Browser #48 → queued

Browser #200 → 503 Service Unavailable

WordCamp Sofia 2026

Baseline, 50 Virtual Users

Home
Starred
Dashboards
Explore
Alerting
Connections
Administration

Virtual Users

50

Requests Per Second

18.2 req/s

p95 Latency

3.24 s

Error Rate

0%

CPU %

90.3%

PHP-FPM Active Workers

No data

Virtual Users & Requests Per Second — Load Ramp



PHP-FPM Workers — Active / Idle / Queue Depth

No data

Latency Percentiles — p50 / p95 / p99



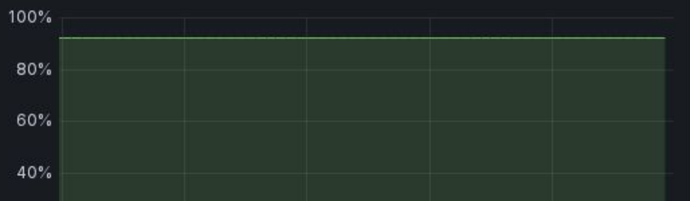
MariaDB — Threads Running



Requests: Total vs Hitting PHP-FPM vs Served from Cache

No data

Redis Object Cache — Hit Rate



WordCamp Sofia 2026

Level 0 - The crash ramp 0 -> 500 VU

Home
Starred
Dashboards
Explore
Alerting
Connections
Administration

Virtual Users

111

Requests Per Second

21.4 req/s

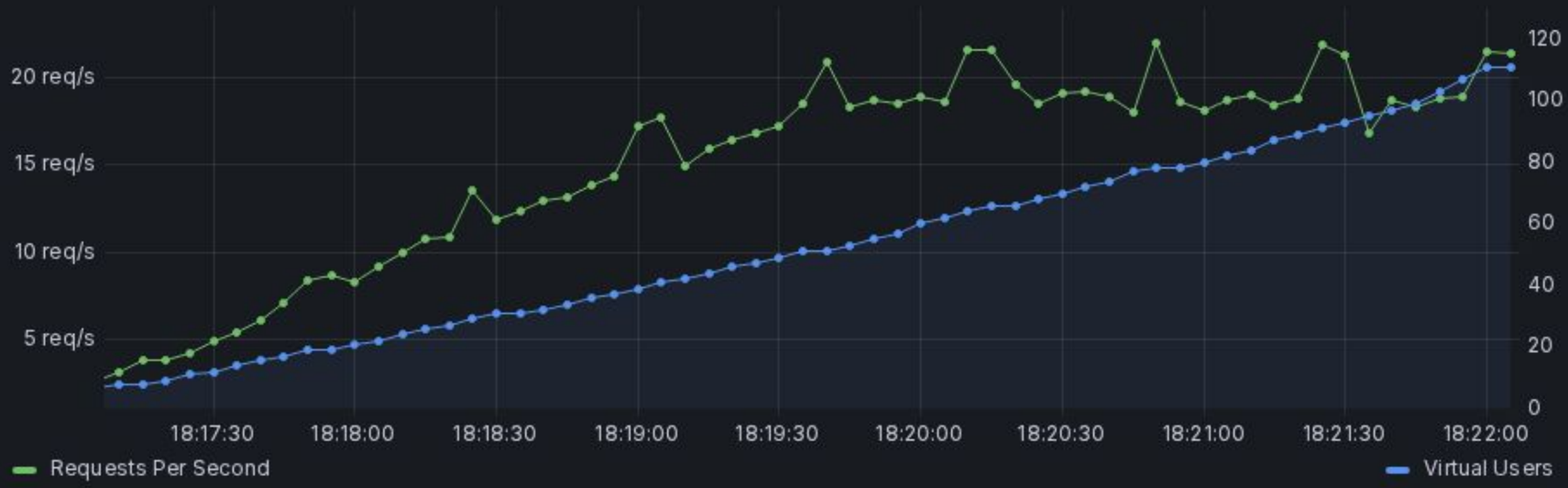
p95 Latency

5.09 s

Error Rate

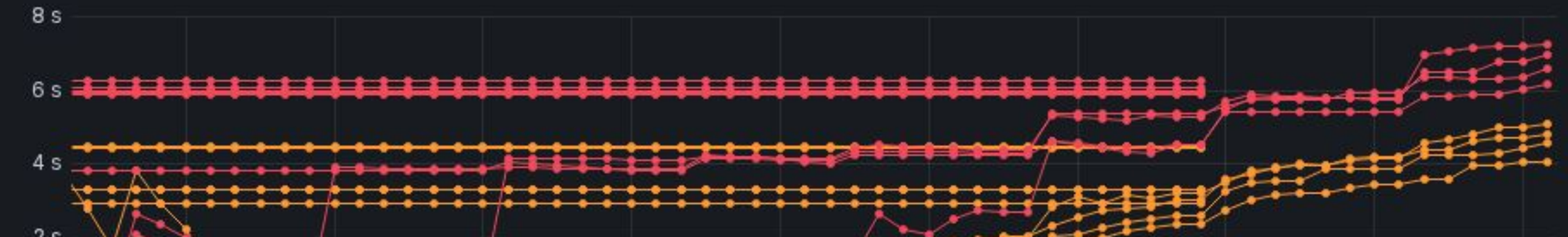
0%

Virtual Users & Requests Per Second — Load Ramp



PHP-FPM Work

Latency Percentiles — p50 / p95 / p99



MariaDB — Thr

60

40

Level 0 — reading the crash

Panel	Metric	Value	What it means
VU & req/s	Peak requests/s	~110 → collapse	Server buckled — req/s fell while VUs kept climbing
VU & req/s	req/s at 500 VU	24.5	**83% of requests failing or queued**
Latency	p95	**~3–6 s**	Many times worse than Level 1
Latency	p99	~6 s	Connections timing out at OS level
Latency	p50	~1–2 s	Even the median is unusable
PHP-FPM	Queue depth	**~250**	Catastrophic backlog — workers exhausted

Level 1 - Nginx + PHP-FPM +OPcache + Redis

1. Nginx (event-driven) One thread, 10,000+ concurrent connections. Zero RAM per idle connection

2. PHP-FPM (decoupled pool) Workers handle PHP execution only. Nginx queues connections externally

3. OPcache + Redis object cache

3.1 OPcache - compiled PHP bytecode in shared memory - zero disk reads

3.2 Redis - DB query results in RAM - **82%** of DB calls served from memory

Level 1 - Nginx + PHP-FPM +OPcache + Redis

❌ Without OPcache (Level 0)

Every single request:
disk read → parse → lex → AST → compile → opcodes → execute → response

Repeat for 300+ PHP files

✅ With OPcache (Level 1+)

First request only:
compile → store in shared RAM

Every request after:
read opcodes from RAM → execute → response

Zero disk. Zero compilation.
~98% hit rate after 30s of traffic

WordCamp Sofia 2026

Level 1 - Demo Dashboard with 500 VU

Home
Starred
Dashboards
Explore
Alerting
Connections
Administration

Virtual Users

500

Requests Per Second

42.0 req/s

p95 Latency

140 ms

Error Rate

0%

CPU %

32.6%

PHP-FPM Active Workers

5

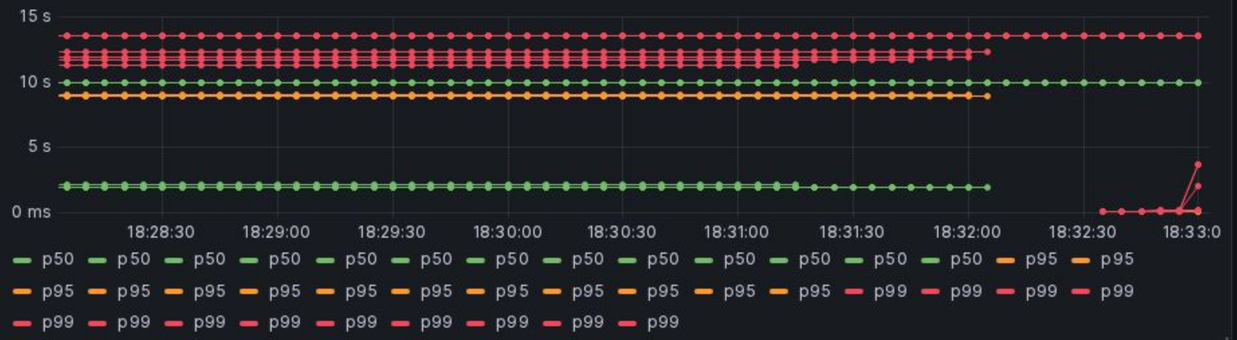
Virtual Users & Requests Per Second — Load Ramp



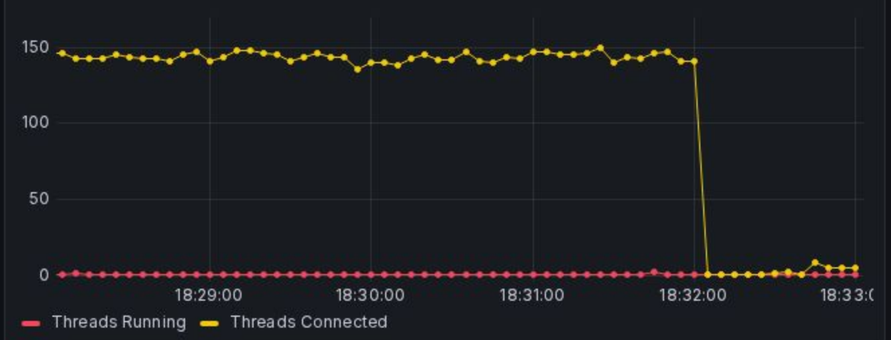
PHP-FPM Workers — Active / Idle / Queue Depth



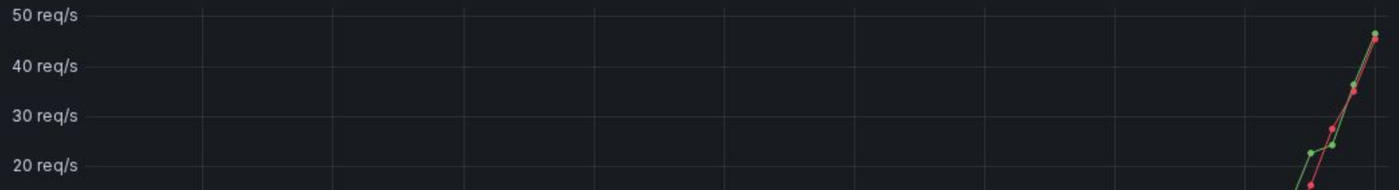
Latency Percentiles — p50 / p95 / p99



MariaDB — Threads Running



Requests: Total vs Hitting PHP-FPM vs Served from Cache



Redis Object Cache — Hit Rate



Level 1: Reading the dashboard

Panel	Metric	Value	What it means
Virtual Users	Peak	50	Steady-state load
Requests/s	**Peak**	**~65 req/s**	**Stable - no collapse**
Latency	**p95**	**~130 ms**	**Was 3+ s - now human-perceptible**
FPM Queue	**Depth**	**0**	**Crash eliminated - was ~250**
Redis	**Hit rate**	**0% → ~80%**	**Cache warming during test**

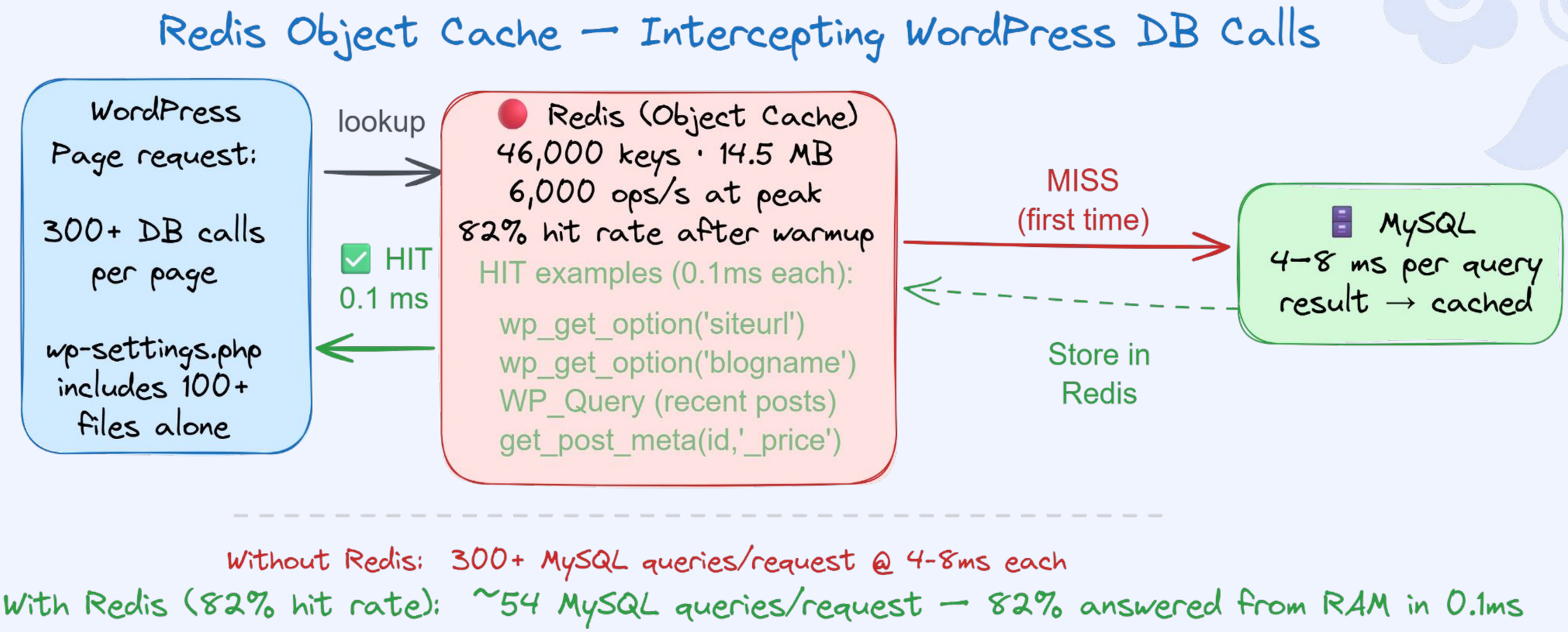
L0 → L1: the jump

Metric	Level 0	Level 1	Δ
Crash at 50 Virtual Users	server buckles	stable	crash eliminated
p95 Latency	~3 s	~130 ms	−96%
CPU @ 50 Virtual Users	~100%	~25%	−75%
FPM queue depth	~250 backlog	0	queue gone
Redis hit ratio	—	~80%	new cache layer

WordCamp Sofia 2026

Level 1 - Redis: 80% Hit Rate, 4K ops/s

Level 1 — Redis: 82% Hit Rate, 6K ops/s



Home
Starred
Dashboards
Explore
Alerting
Connections
Administration

Cache Hit Ratio

90.0%

Used Memory

8.18 MB

Total Keys

229

Evictions / sec

0

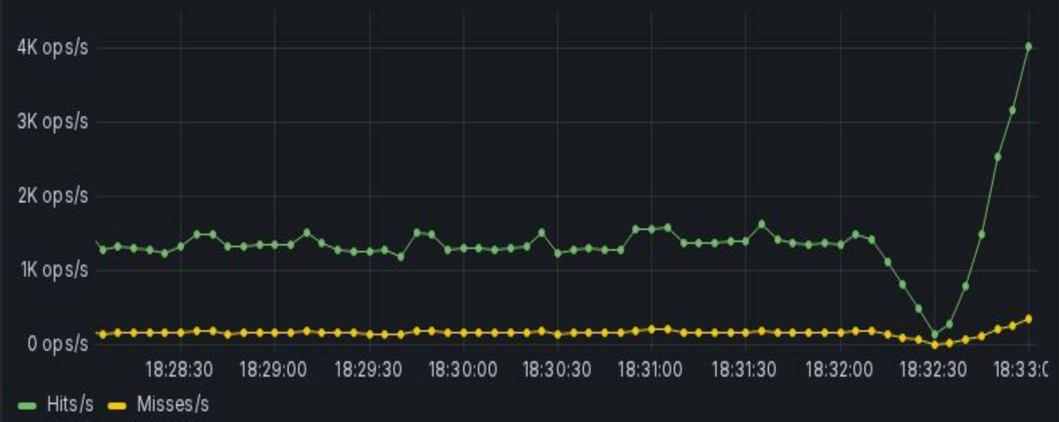
Connected Clients

10

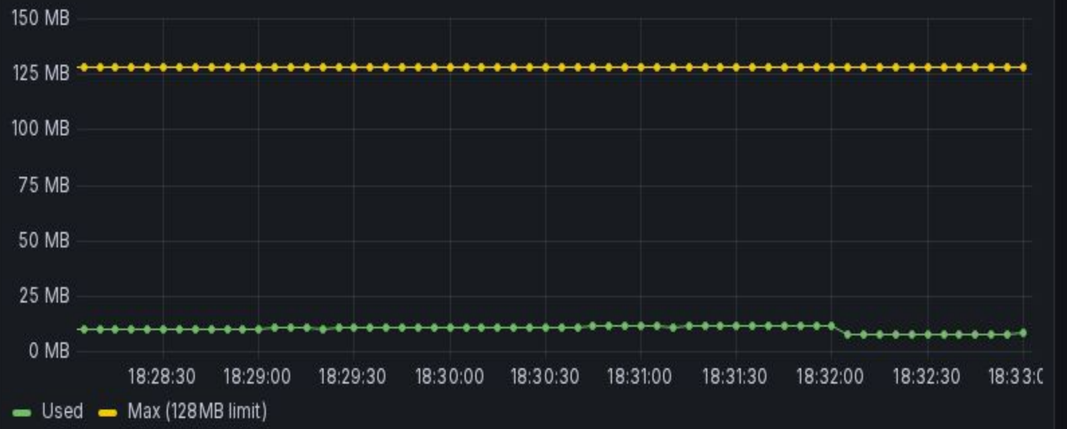
Commands / sec

4.34K ops/s

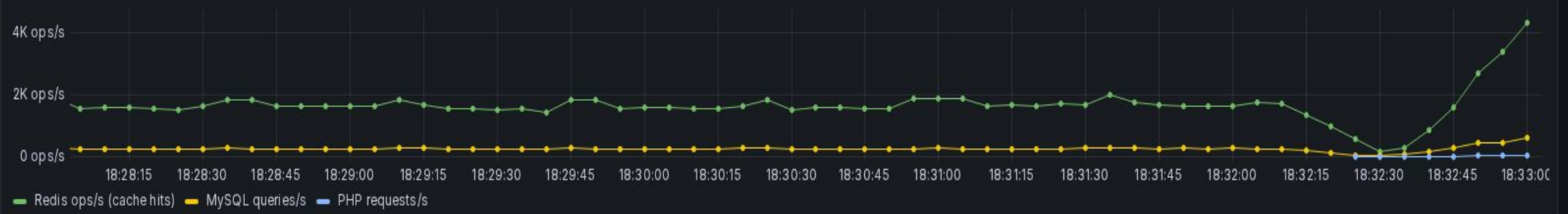
Hits vs Misses over time

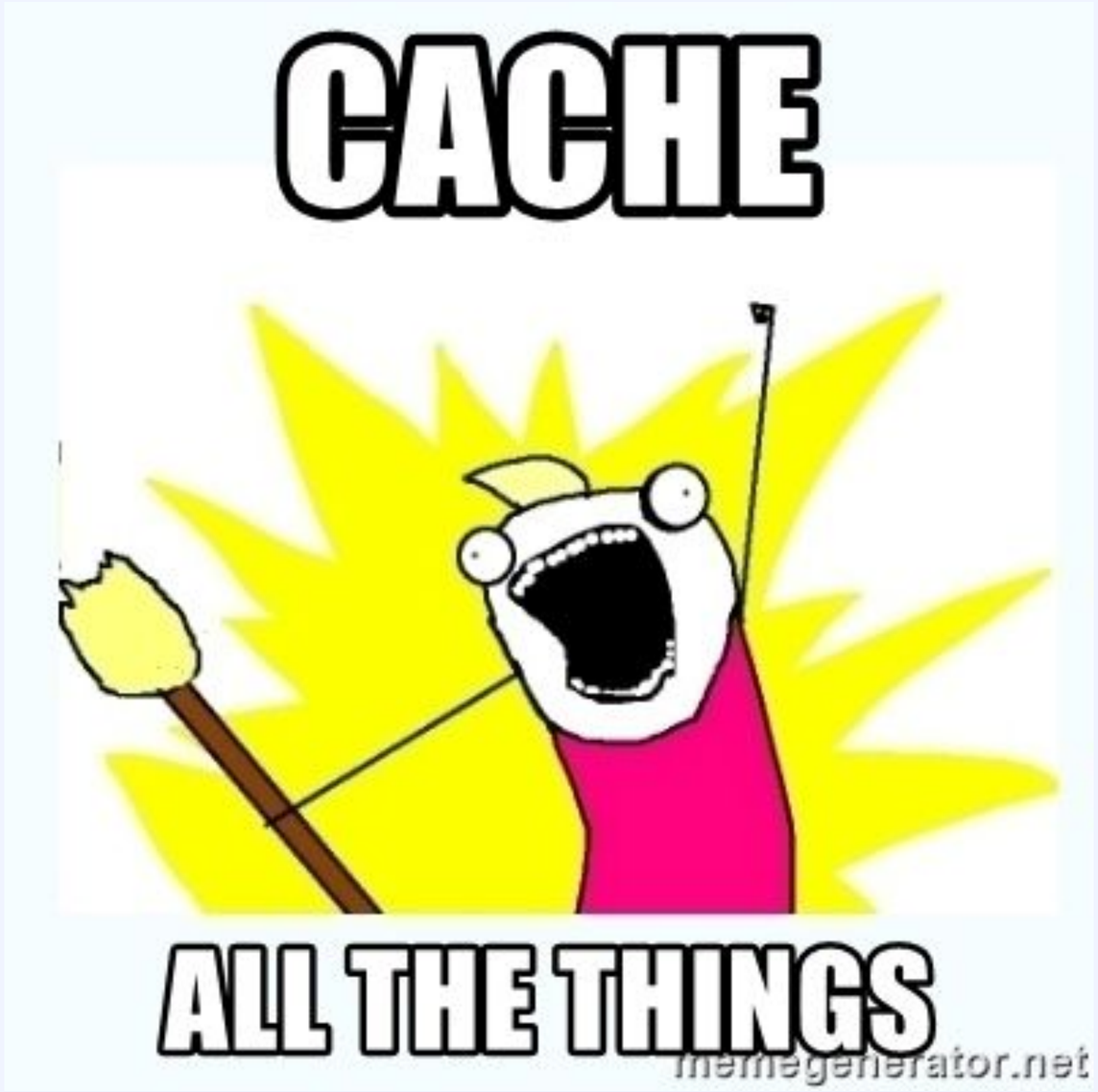


Memory Used vs Max



DB Queries Saved by Redis Object Cache



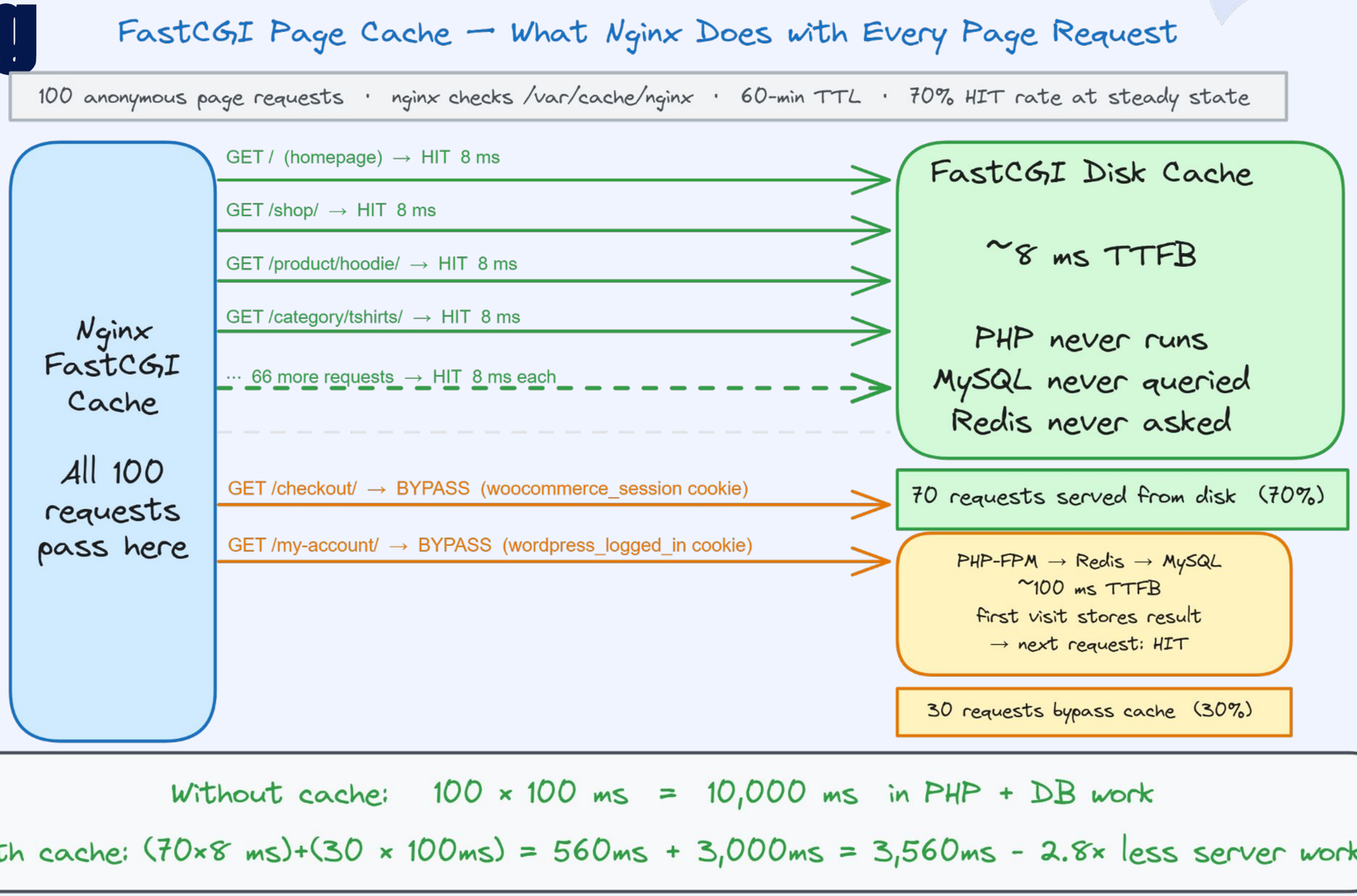

CACHE
ALL THE THINGS
memegenerator.net

WordCamp Sofia 2026

Cache Everything

Level 2 - FastCGI page cache + MariaDB tuning

Level 2 — FastCGI page cache + MariaDB tuning



WordCamp Sofia 2026

Level 2 - ramp at 360 VU

Home
Starred
Dashboards
Explore
Alerting
Connections
Administration

Virtual Users

360

Requests Per Second

62.0 req/s

p95 Latency

65.3 ms

Error Rate

0%

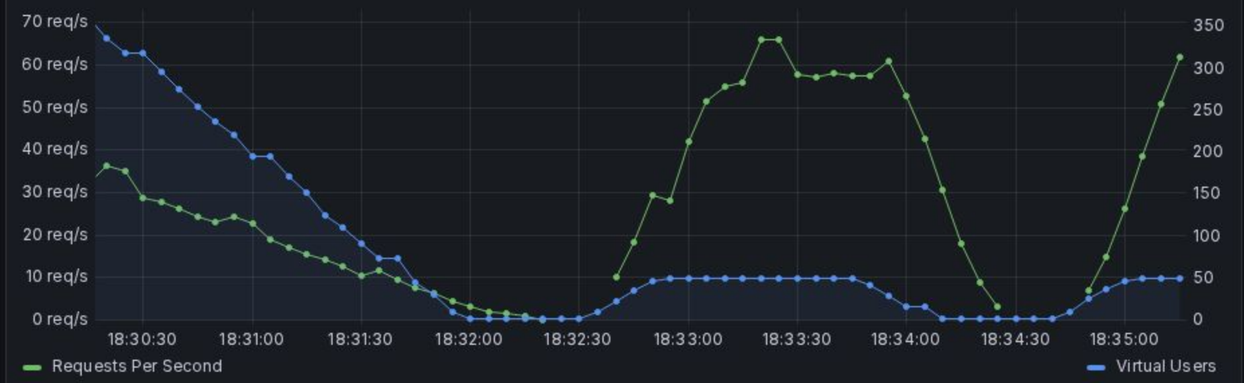
CPU %

6.13%

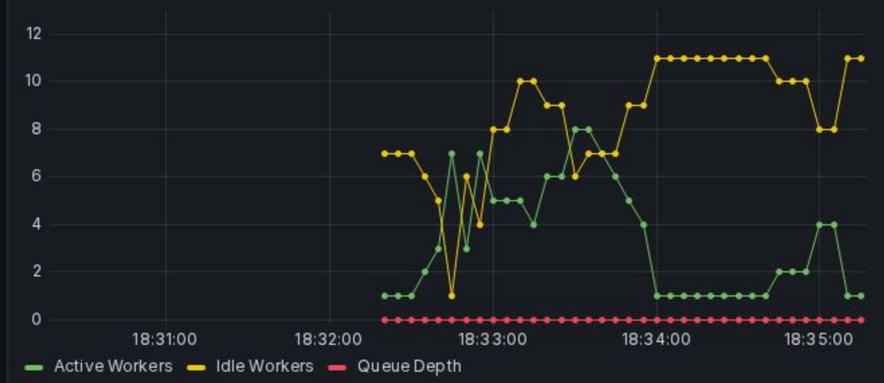
PHP-FPM Active Workers

1

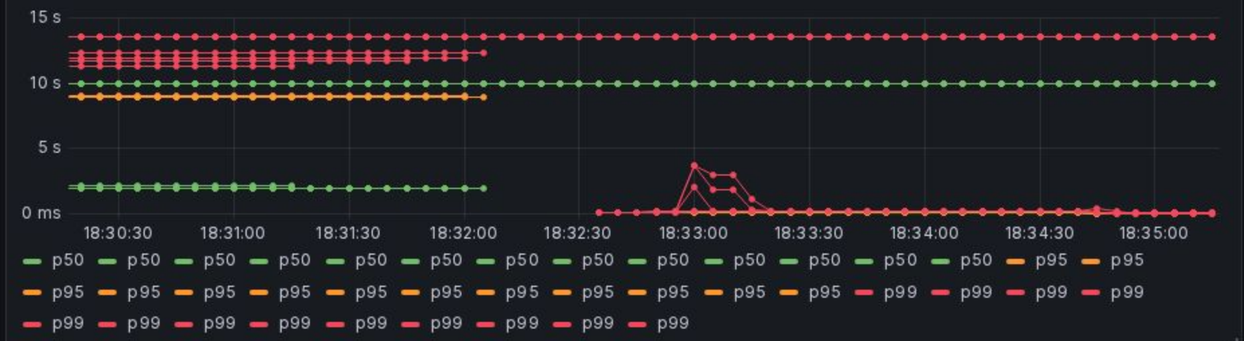
Virtual Users & Requests Per Second — Load Ramp



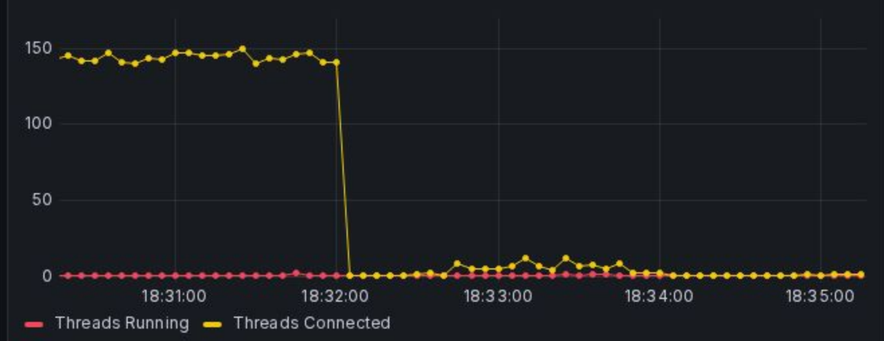
PHP-FPM Workers — Active / Idle / Queue Depth



Latency Percentiles — p50 / p95 / p99



MariaDB — Threads Running



Requests: Total vs Hitting PHP-FPM vs Served from Cache



Redis Object Cache — Hit Rate



L0 → L1 → L2: the progression

Metric	Level 0	Level 1	Level 2
Peak requests/s	**~140 →**	**~65 req/s**	**~70–75 req/s**
p95 Latency	**~3 s**	**~130 ms**	**~70–80 ms**
CPU @ 50 Virtual Users	~100%	~25%	~18%
FPM queue depth	~250	0	0
DB Threads peak	20+	~10–15	~3

WordCamp Sofia 2026

Level 2 - NgInx Cache: FPM Offload In action

Home
Starred
Dashboards
Explore
Alerting
Connections
Administration

Active Connections

51

Accepted Connections / se

1.26 c/s

Requests / sec

52.0 req/s

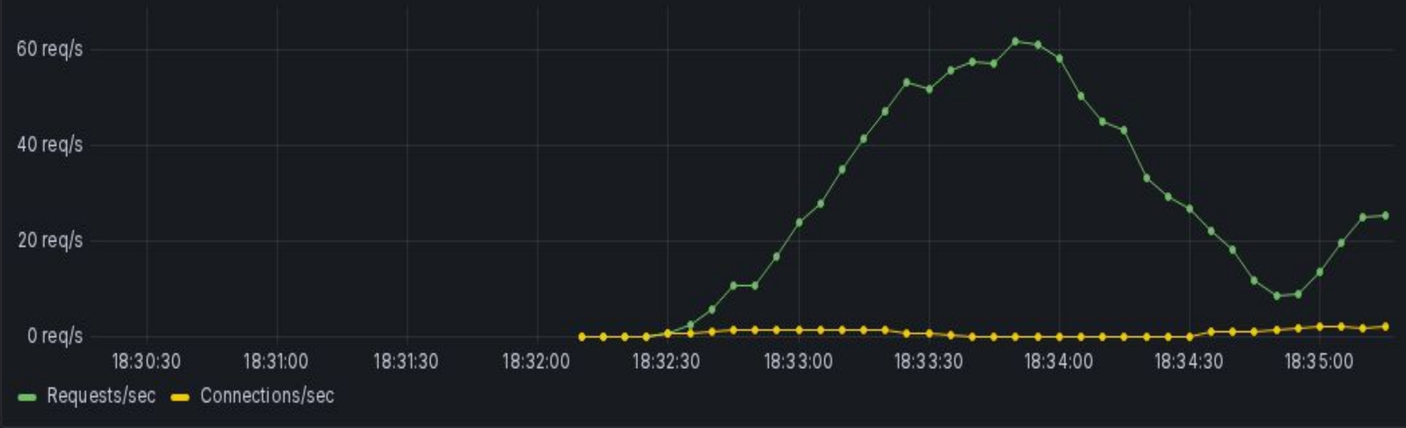
Waiting Connections

49

Total Requests (5m window)

6.42 k

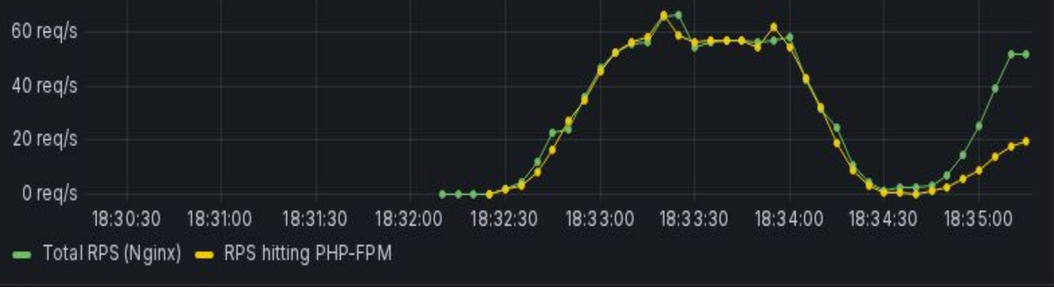
Request Rate Over Time



Current Connection State



PHP-FPM Offload — Requests NOT hitting PHP



WordCamp Sofia 2026

Level 2 - the FPM offload panel explained

Level 2 — FPM Offload Panel: Nginx vs PHP-FPM Request Rate

Grafana panel · 50 Virtual Users · 5-minute test · steady state after ~60s warmup



~70 req/s total - ~15–20 req/s to PHP = ~50 req/s answered from disk

2.8× more throughput on same hardware

WordCamp Sofia 2026

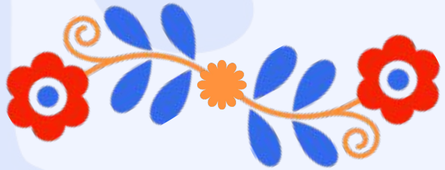
Level 1 vs Level 2 - the real numbers

Metric	Level 1	Level 2	Change
p95 Latency	~130 ms	~70–80 ms	~−40%
CPU usage	~25%	~18%	~−30%
MariaDB threads (peak)	~12–15	~3	~−75–80%
Cache offload	0%	~70%	majority served from disk
Requests in 5 min	~19,500	~21,500	~+20–25%

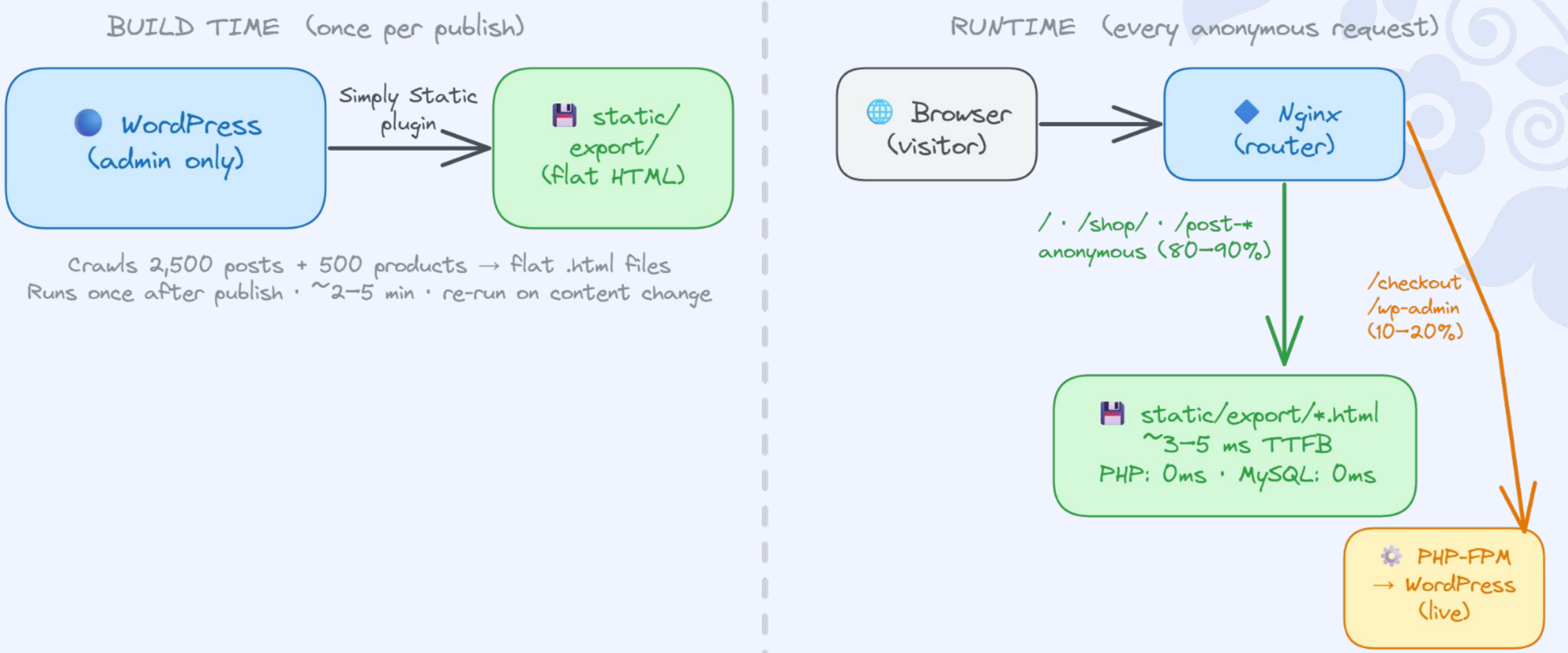
WordCamp Sofia 2026

What's next?

The Static Leap



Level 4 — WordPress as a CMS, Nginx as the Runtime



Static file serve: ~0.1ms SSD read (or <0.01ms OS page cache) · 2 vCPU → ~10,000 req/s
PHP workers: needed only for cart · checkout · admin — anonymous traffic: zero PHP

WordCamp Sofia 2026

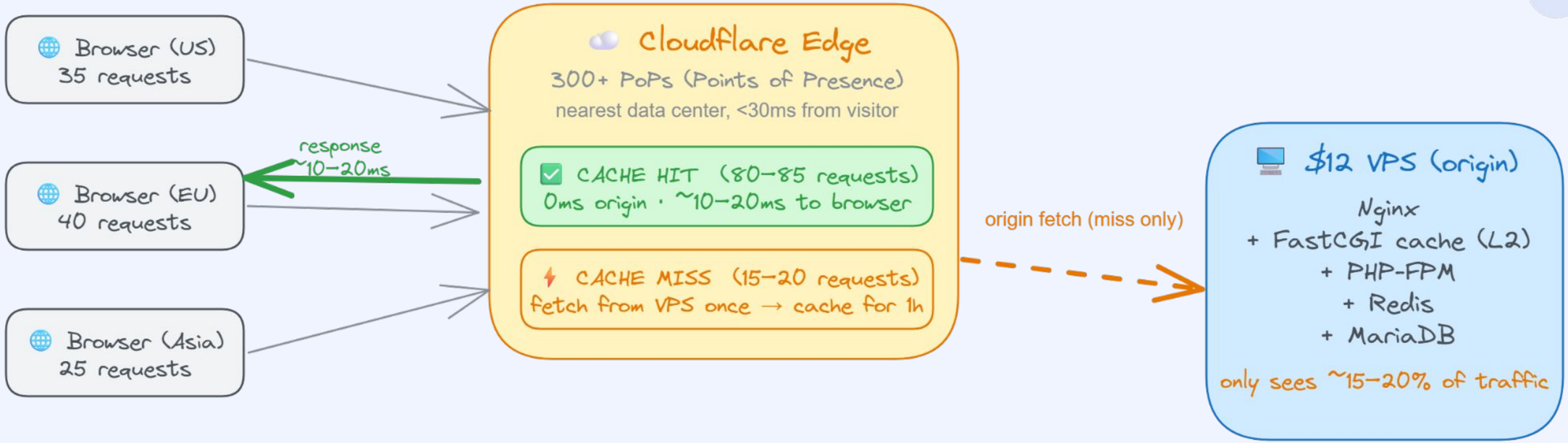
What's next?

CloudFlare

Wait? There's more?

Level 3 — Cloudflare CDN: Your VPS Only Sees Cache Misses

100 requests during a traffic spike · visitors from US, EU, Asia · same $12 VPS as origin



100 req spike → 80–85 served from Cloudflare edge (~10–20ms, zero VPS load)
15–20 reach VPS → L2 FastCGI likely HITs most → VPS barely notices the spike

CDN bonus points

Performance (free)

- Global CDN — 300+ PoPs worldwide
- Full page caching via Cache Rules
- Free SSL/TLS — automatic HTTPS
- HTTP/2 + HTTP/3 (QUIC) — automatic

Security (free)

- DDoS mitigation — unlimited, unmetered
- Bot Fight Mode — basic bot blocking
- WAF — 5 custom rules
- IP reputation — automatic bad actor blocking
- Analytics — requests, bandwidth, threats

WordCamp Sofia 2026

The full picture

Every improvement listed here costs exactly $0.

WordCamp Europe
Kraków 2026

Every Improvement llsted here costs exactly $0

Level	Stack	p95 Latency	CPU @ 50 VU	DB Threads
0	Apache + mod_php	Crash ~50 VU	100%	20–30
1	Nginx + FPM + OPcache + Redis	~100–150 ms	~25%	~10–15
2	Level 1 + FastCGI cache + MariaDB	~70–90 ms	~18%	~3
3	Level 2 + Cloudflare CDN	~20 ms (edge)	<5%	<1
4	Level 2 + Simply Static	~4 ms	<5%	<1

WordCamp Sofia 2026

The full picture

The Cache Hierarchy (full stack view)

Every layer is a PHP execution avoided. Hit the left as much as possible.

1. Browser Cache (Cache-Control: immutable)

2. CDN / Cloudflare Edge (global PoPs, R2 for media)

3. Nginx FastCGI Page Cache (RAM, 60 min TTL)

4. PHP OPcache (bytecode in shared memory)

5. Redis Object Cache

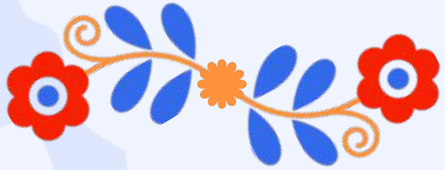
6. MariaDB



💀 Disk I/O (the floor you never want to hit)

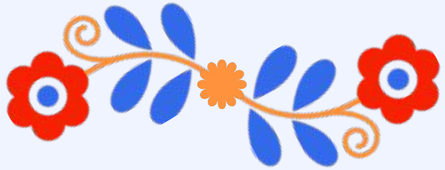
Every improvement listed here costs exactly $0.

WordPress is not slow. You setup is.
Set up Prometheus + Grafana before you need it. You cannot fix what you cannot see



WordCamp Sofia 2026

nedko.info/wcsf2026



WordCamp Sofia 2026

Благодаря!



nedko.info/wcsf2026

It is Lato. Lato was created by:

Polish designer Łukasz Dziedzic