High-Load Systems: Overcoming Challenges in Social Network Development

By Alexander Kolobov

Alexander Kolobov

Developer and Team Lead

- Teams > 10 members
- Desktop and Mobile
- New users and onboarding
- Workflows, roadmaps, KPIs

Today

- What is high-load
- High-load challenges and requirements
- Technologies vs challenges

What is High-Load

High-Load or Not?

RPS
>10K

Latency
300ms

Availability
>99,99%

Resource
Utilisation
>50%

VK Social Network

MAU
100M

Posts/day
100M

Post views
per day
9B

Servers
20,000

VK Social Network

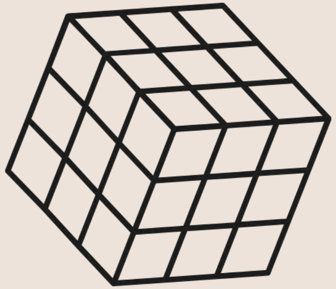
RPS
3M

Latency
120ms

Availability
>99,94%

Resource
Utilisation
>60%

High-Load Challenges



High-Load Challenges

- Performance
- Data Management
- Scalability
- Reliability
- Fault Tolerance

External Solutions Risks

- Designed for broad application
- Vulnerability
- Failures under high-loads
- Limited control
- Scalability limitations

High-Load Structure Requirements

- Downtime is unacceptable
- Zero data loss ensured by cloud services
- Linear scaling
- Ease of maintenance

Technologies vs Challenges

VK Architecture Evolution

Year	Users	Technology
2013	55 million	KPHP to C++ translator
2015	76 million	Hadoop
2017	86 million	CDN
2019-2020	97 million	Blob Storage, gRPC, microservices on Go/Java, KPHP language
2021-2022	100 million	Parallelism in KPHP, QUIC, ImageProcessor, AntiDDOS

What happened?

- Popularity growing
- Databases slowing down
- Large and slow codebase
- Increased user-generated content

Specialized Databases or Engines

- Data storage
- Microservice with an embedded database
- C/C++

Benefits of Custom Engines

- Minimal structuring
- Efficient data access
- Fast query execution
- Performance optimization
- Scalability

Heavy Caching

- Precomputed data
- Automatic code-level scaling
- Reduces load on backend

KPHP

$\text{PHP} \longrightarrow \text{C++}$

- 2-40 times faster in synthetic tests

KPHP — 10 times faster in production environments

KPHP benefits

- Development convenience
- Support for PHP 7/8

Open Source Features:

- Fast compilation
- Strict typing
- Shared memory
- Parallelization
- Coroutines
- Inlining
- NUMA support

Noverify PHP Linter

- Designed for large codebases
- Focuses on analyzing git diff pre-push
- Indexes approximately 1 million lines of code per second
- Analyzes about 100,000 lines of code per second
- Can also run on standard PHP projects

Microservices
Go/ Java/ gRPC

- Time to market
- Develop services in different languages

Bottlenecks in Content Storage and Delivery

- Every image needs to be displayed in multiple sizes
- Interface requirements
- Different platforms

Image Processor and WebP format

Results of switching from JPEG to WebP

- 40% reduction in photo size
- 15% faster delivery time (50 to 100 ms improvement)

Other highload companies — the same principles

- Netflix: caching strategies and custom data storage solutions
- Yandex: ClickHouse, in-house caching solutions, distributed systems
- LinkedIn: Espresso and caching with Apache Kafka
- Twitter: Manhattan distributed database

Key Takeaways

- High-load systems challenges: scalability limitations, reliability issues, performance bottlenecks, and integration complexity.
- Requirements: zero data loss, rapid feature deployment, and minimal downtime.
- Under high loads external solution become risky.
- Identify main bottlenecks and optimize them
- Technologies: efficient and scalable data storage with robust caching, compiled languages, distributed architecture, and advanced tooling.

DM me in Telegram

@iamaleko