Anton Bobrov
Software engineer

Location: Moscow
E-mail: baverman@gmail.com
GitHub: https://github.com/baverman
Telegram: @bvrmn

Profile

I am a Python backend developer since 2007. During my career I've been built a plenty of fast HTTP APIs for mobile and desktop applications with consistent and convenient interfaces. I like simple implementations and know a couple of tricks how to investigate and fix performance issues. Also, I have a background with a desktop, web frontend, and mobile development.

I prefer straightforward code with minimum abstractions, the clean architecture (strict layers without abstraction leaks), and narrow and deep components (isolated highly cohesive pieces with an encapsulated domain model and minimal public interface).

Some of my projects to look at:

Fast TSDB backend for Graphite (https://github.com/baverman/hisser)
Python data validation with web in-mind (https://github.com/baverman/covador)
Dead simple queue (https://github.com/baverman/dsq)

Python:

py2/py3, stdlib, asyncio, Flask/FlaskAdmin, SQLAlchemy, Alembic, Django, aiohttp, requests, Celery, uWSGI, lxml, cryptography, NumPy, Pandas, Cython, CFFI, ctypes, cpu/memory profilers, memory leaks.

DB:

PostgreSQL, Redis, Mongo, SQLite, LMDB, InfluxDB, Clickhouse, Elasticsearch, MySQL, Hive.

Containers:

Docker, LXC, systemd-spawn, KVM, qemu-nbd.

Tools:

Sentry, Grafana, Graphite, Jenkins, Telegraf

Linux administration:

Ansible, Fabric, systemd, nginx, HAProxy, bash scripting, pipelines, CentOS, Ubuntu, Alpine, networking, LVM.

System programming:

Process management, sockets, TCP, UDP, non-blocking IO, low-level protocol parsing.

Clouds:

GCP, AWS, Linode, OpenNebula.

VCS:

Git/StGit, Mercurial, SVN.

Frontend:

HTML, CSS, JavaScript, jQuery, React.

Other languages:

Node, C, Go, Java, Groovy, PHP.

Notable experience

Nov 2020 — today, CloudLinux, Team Lead, remote

A lead of KernelCare python team (5 members). Product allows to update kernel and userspace processes without reboots and restarts.

The team maintains:

- KernelCare agent: glue logic, packaging for rhel-based (el6, el7, el8, RedHat, centos, RockyLinux, AlmaLinux) and deb-based (Debian >=8, 9, 10, 11, Ubuntu >=16.04) distros.
- Scanner interface: ldpreloadable library written in C, it allows to rewrite package version info to force security scanners report proper patched versions.
- Patch server: licensing, patches acceess rules, patch release process.
- ePortal: on-premise solution for patch delivery for enterprise clients.
- Internal tools for tracking CVE.
- Framework to test kernel and userland patches.

Stack: python3, uWSGI, SQLite, Clickhouse, nginx, HAProxy, TincVPN, Grafana, Sentry, Docker.

Achievements:

- Speed up agent functional tests from 1 hour to 20 minutes.
- Moved ePortal tests from real VM in cloud to docker containers, speed up: from 30 minutes to 10 minutes.
- Developed and released a new forward compatible patches meta format for userland patches. A new patchset release doesn't depend from meta from old releases.
- Fixed kernel test suite bugs to allow green runs.
- Implemented HA on patch server.
- Implemented HA on ePortal.
- Introduced package building in docker containers. This allowed to build packages for non x86-64 architectures and reduced build time from 10 minutes to 30s.
- Developed and released internal SAML2 to OAuth2 proxy to allow internal services to use simple OAuth2 clients to authorize with corporate SSO.

- Refactored scanner interface with safe C practices, increased tests coverage and fixed numerous segfaults.
- Documented release process and introduced branch policy for ePortal and Kernelcare Agent packages.

Mar 2018 — Feb 2020 (2 years), CloudLinux, Team Lead, remote

A lead of Imunify360 backend team (3 members). The main service takes care of an event stream from all client installations (thousands of events per second, gigabytes of data per day), pushes it through ETL pipeline to Clickhouse datastore, and handles notifications back to clients.

Stack: python3, asyncio, aiohttp, Redis, Mongo, Clickhouse, HAProxy, TincVPN, Grafana, Sentry, Docker.

Achievements:

- Reworked service architecture to be able to process a growing event stream.
- Implemented app-level real-time monitoring with in/out metrics on each step of ETL path. Statsd, Telegraf, Hisser, Grafana.
- 100% test coverage of whole ETL path, client proto parsing, queue handling, disk storage, ETL processors. As a result, team can deploy new releases directly to production without manual testing on a stage environment.
- Speed up functional tests from 30 to 10 minutes.
- Optimized ansible playbook. Deploy time shortened from 10 to 1 minute.
- Refactored a private admin site for main service (old unmaintainable code with numerous bugs).
- Implemented a fast reverse-ip lookup component.
- 20x speedup of event validation code.
- Reviewed and fix mongo indexes, added partial indexes to fulfill query requirements.
- Tuned HAProxy timeouts and logging, configured upstream checks.
- Moved ETL storage from Hadoop to Clickhouse and implemented Clickhouse automatic table migrations.
- Mesh VPN between production cluster nodes.
- Implemented geo-distributed RBL.

Aug 2016 — Feb 2017 (7 months), Rambler&Co, Team Lead

A lead of Rambler/News team, 6 members, 4 backend and 2 frontend devs. There were 3 services: desktop website + mobile API, mobile site and admin control panel for the content management team. Load: 800 requests per second. My main duties were: solve performance issues and reduce technical debt.

Stack: python2, PostgreSQL, Redis, Django, flask, celery, SQLAlchemy, nginx, uWSGI.

Achievements:

- 2x codebase reduction from 40k to 20k SLOC. Code was in a messy state, endpoints simultaneously used Django/DjangoORM, Django/SQLAlchemy and Flask/SQLAlchemy with duplicated functionality. I've refactored and implemented all endpoints as Flask/SQLAlchemy views.

- Performance optimization. CPU load was reduced by 4 times. The average response latency was reduced by 3 times. I tuned DB indexes, replace complex ORM code (hybrid props and multi-level relation joins which can lead to up to 300 db requests per one http request) with straightforward queries and introduced 2-level caching with uWSGI and Redis.
- Reduced deploy time from 30 to 5 minutes.
- All services were prepared for migration from FreeBSD to Linux.

May 2014 — Jun 2016 (2 years), Zvooq, Team Lead

A lead of the backend team consisting of 6 members. Zvooq is a music streaming service and our team was taking care of mobile/web API, search, editorial team tools, subscriptions, pushes, billing, 3rd party integrations, and backend code for custom ad campaigns.

Stack: python2, FlaskAdmin, PostgreSQL, Elasticsearch, LMDB, Redis, Werkzeug, Celery, SQLAlchemy, nginx, uWSGI, Sentry

Achievements:

- Performance optimization and huge architecture rework. The main service couldn't handle 20rps load before and can process 1000rps after on the same hardware.
- Application real-time monitoring.
- Reduced deploy time from 20 to 1 minute. Implemented CI/CD
- Force unit tests and full coverage. The team can deploy releases after automatic tests directly to production servers.

Other commercial experience

- An analytical system (python) with UI to create multi dimensional queries to click/view stream data in Hadoop with pre-aggregated HLL counters for non-additive data.
- A python backend for UI of one of the biggest Russian metric/counter sites.
- A python backend for postomat delivery service.
- Backend for biggest regional auction/advert service (PHP, Java).
- A backend and UI for logistic/account system for a bakery production Business.