

# 1. Мониторинг технических метрик с Zabbix.

Мониторинг с Zabbix настроен следующим образом:

- В docker-compose подняты zabbix-server, zabbix-web и postgres DB.
- Приложение запущено локально.
- Локально же установлен zabbix-агент для сбора технических метрик.

Как оказалось, сам zabbix предоставляет уже довольно широкий набор технических метрик для мониторинга. По сути требовалось только настроить правильное взаимодействие. На скриншоте показаны самые простые метрики — утилизация CPU и использование памяти. Всплески по CPU связаны с тем, что утилитой HEY в 20 параллельных запросов осуществлялось нагрузочное тестирование. Очевидно, оно повлияло на CPU, но не затронуло использование памяти, из чего делаем вывод, что мониторинг работает верно.

```
eugene@IdeaPad-Eugene:~$ hey -c 20 -n 1000000 -m GET 'http://localhost:8083/user/search?first_name=%D0%98%D0%B2%D0%B0%D0%BD&last_name=%D0%90%D0%BD%D1%82%D0%BE%D0%BD%D0%BE'
AC
Summary:
  Total:      244.0721 secs
  Slowest:    0.8324 secs
  Fastest:    0.1322 secs
  Average:    0.2985 secs
  Requests/sec: 66.9761

  Total data: 269120661 bytes
  Size/request: 16463 bytes

Response time histogram:
  0.132 [1] |
  0.202 [1207] |
  0.272 [4584] |
  0.342 [7084] |
  0.412 [2502] |
  0.482 [700] |
  0.552 [180] |
  0.622 [65] |
  0.692 [16] |
  0.762 [5] |
  0.832 [3] |

Latency distribution:
  10% in 0.2221 secs
  25% in 0.2574 secs
  50% in 0.2916 secs
  75% in 0.3330 secs
  90% in 0.3839 secs
  95% in 0.4213 secs
  99% in 0.5122 secs

Details (average, fastest, slowest):
DNS+ dialup: 0.0000 secs, 0.1322 secs, 0.8324 secs
DNS lookup: 0.0000 secs, 0.0000 secs, 0.0021 secs
req write: 0.0000 secs, 0.0000 secs, 0.0060 secs
resp wait: 0.2983 secs, 0.1321 secs, 0.8323 secs
resp read: 0.0001 secs, 0.0000 secs, 0.0087 secs

Status code distribution:
  [200] 16347 responses
```

Рис. 1. Нагрузка приложения



Рис. 2. Мониторинг в Zabbix

## 2. Мониторинг в Prometheus.

По заданию предполагается мониторинг по принципу RED, что означает сбор и отображение метрик по Rate, Error и Duration.

Мониторинг реализован следующим образом:

- в docker-compose вместе с кластером БД postgresql (который планируется мониторить) поднимаются pg-exporters, по одному на каждый инстанс БД (всего 3), которые собирают метрики и готовы отдавать скраперу prometheus.
- Сам prometheus, как и grafana для визуализации, установлены локально как systemd-сервисы.

PG-экспортеры предоставляют довольно широкий спектр метрик. Также отсюда (<https://grafana.com/oss/prometheus/exporters/postgres-exporter/?tab=dashboards#metrics-usage>) взят пример дашборда для postgres. Здесь QPS, Number of connections и Rows отвечают за Rate, а Conflicts/Deadlocks - за Errors.

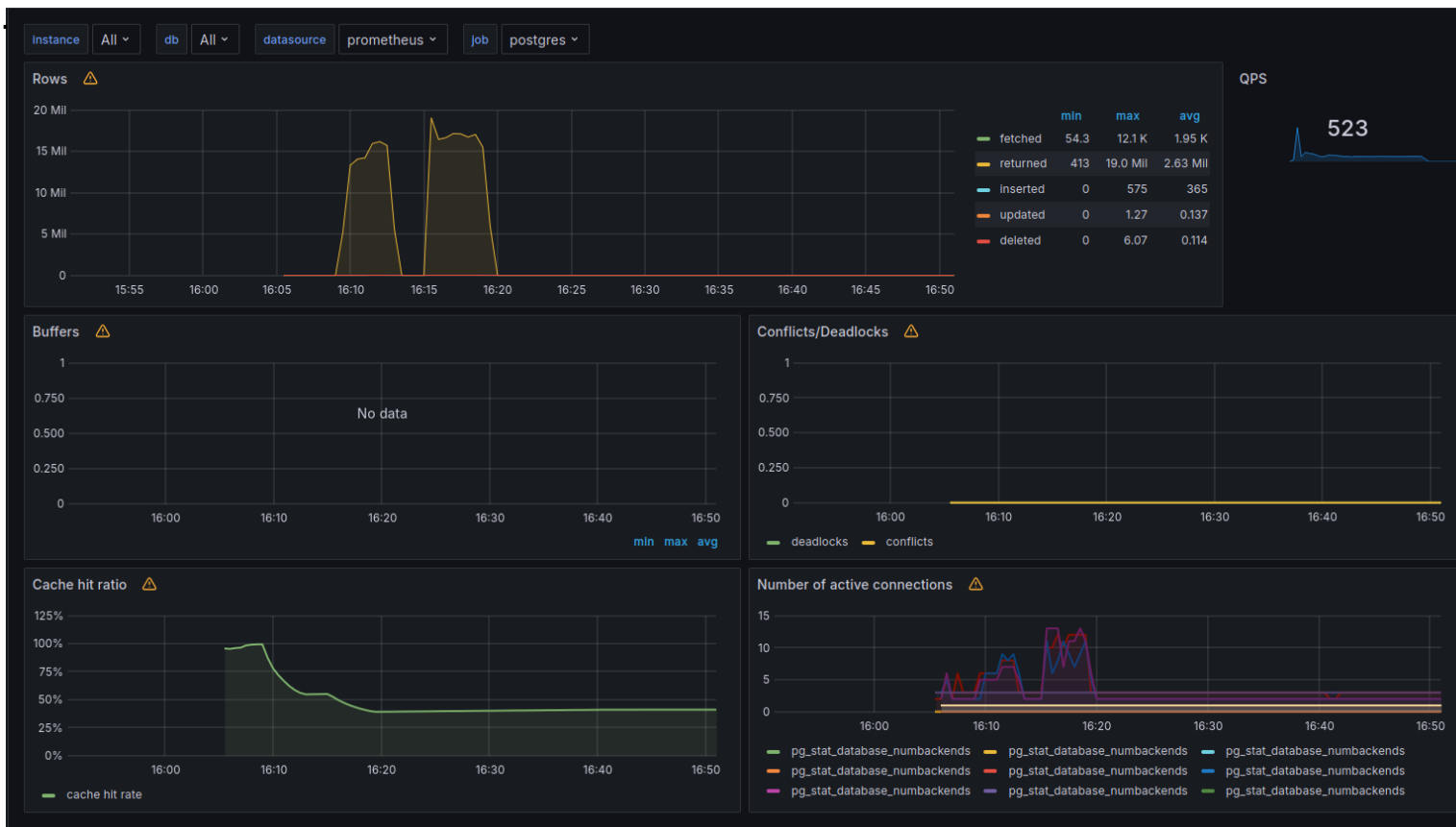


Рис. 3. Дашборд для Postgres

На предыдущем дашборде не хватает метрик, отвечающих за Duration из принципа RED. Поэтому на отдельной дашборде выводим:

- `sum(pg_stat_activity_max_tx_duration{datname="social_net", state="active"}) by (server)`
- и метрику, косвенно связанную с duration - количество локов.



Рис. 4. Еще метрики.

Совсем логично тут было бы использовать `total_time`, `mean_time` из `pg_stat_statements` (запросы для них есть в репозитории в `queries.yaml`), но их сбор не получилось настроить.