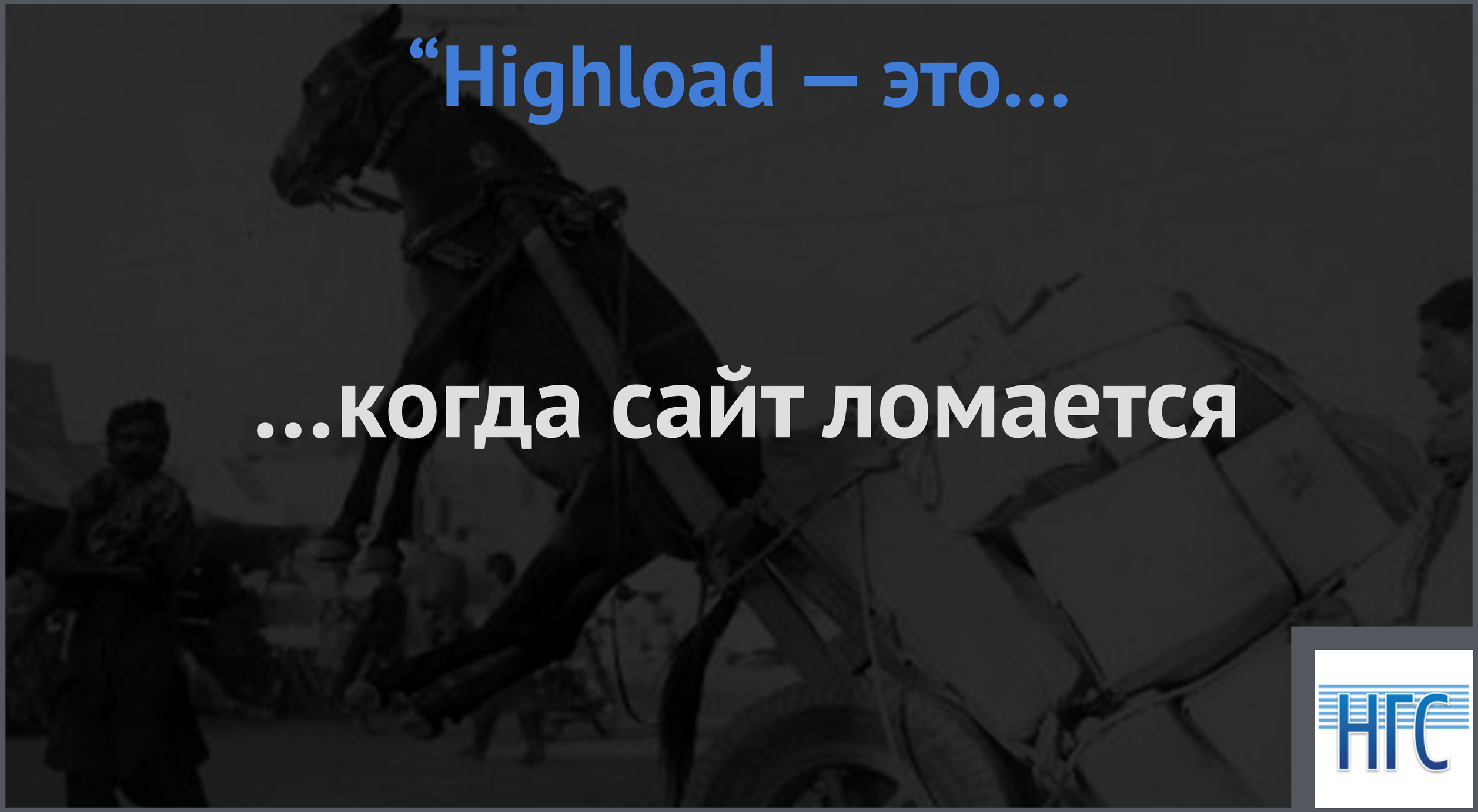
Что такое Highload?

Тайны высоконагруженных систем.
Скандалы. Интриги.
Расследования.

НГС

“Highload — это...


“Highload — это...
Ваш вариант?



“Highload — это...

...когда сайт ломается

НГС

“Highload — это...

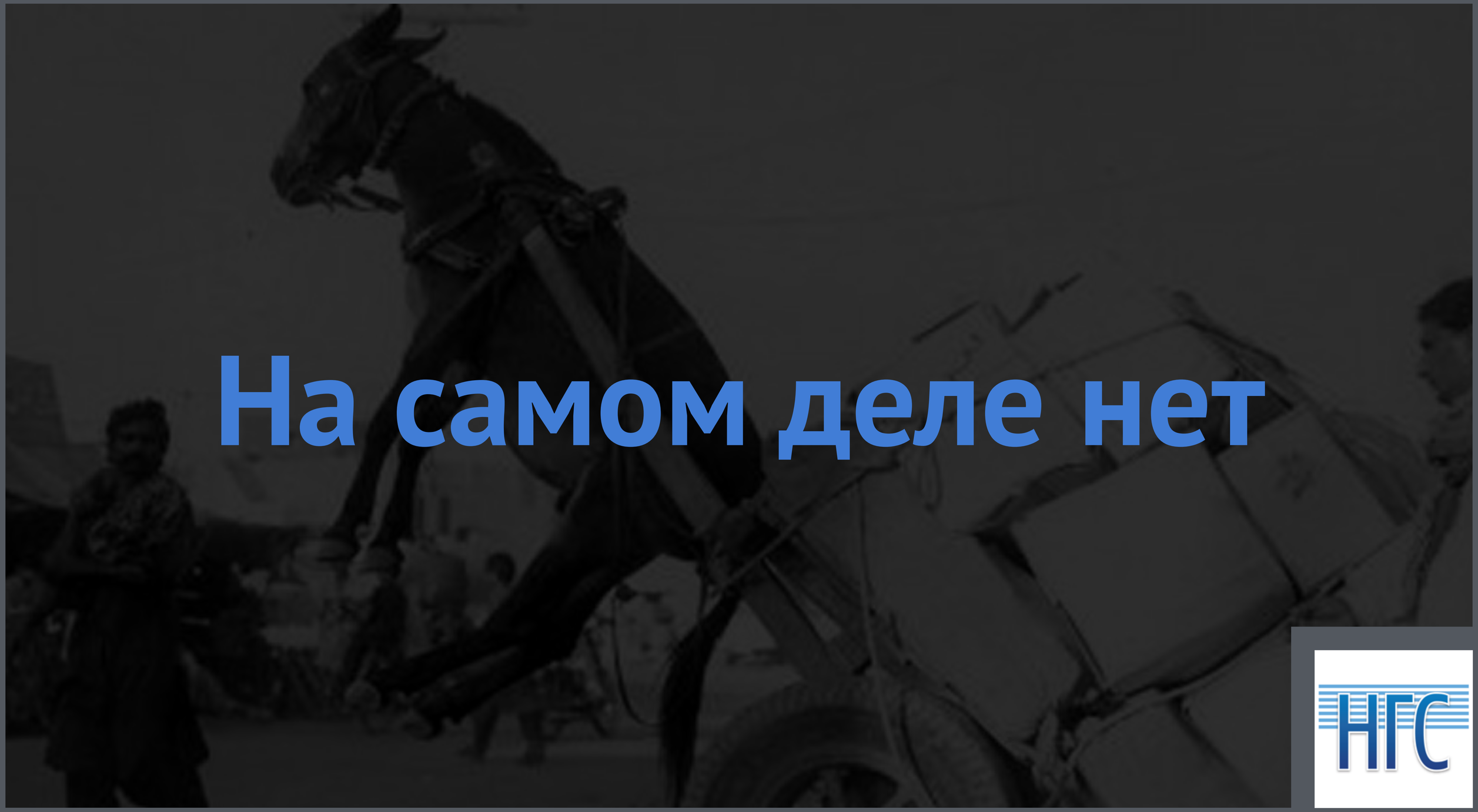
...когда больше 10 тысяч пользователей в день

“Highload — это...

...когда больше 100 тысяч пользователей в день

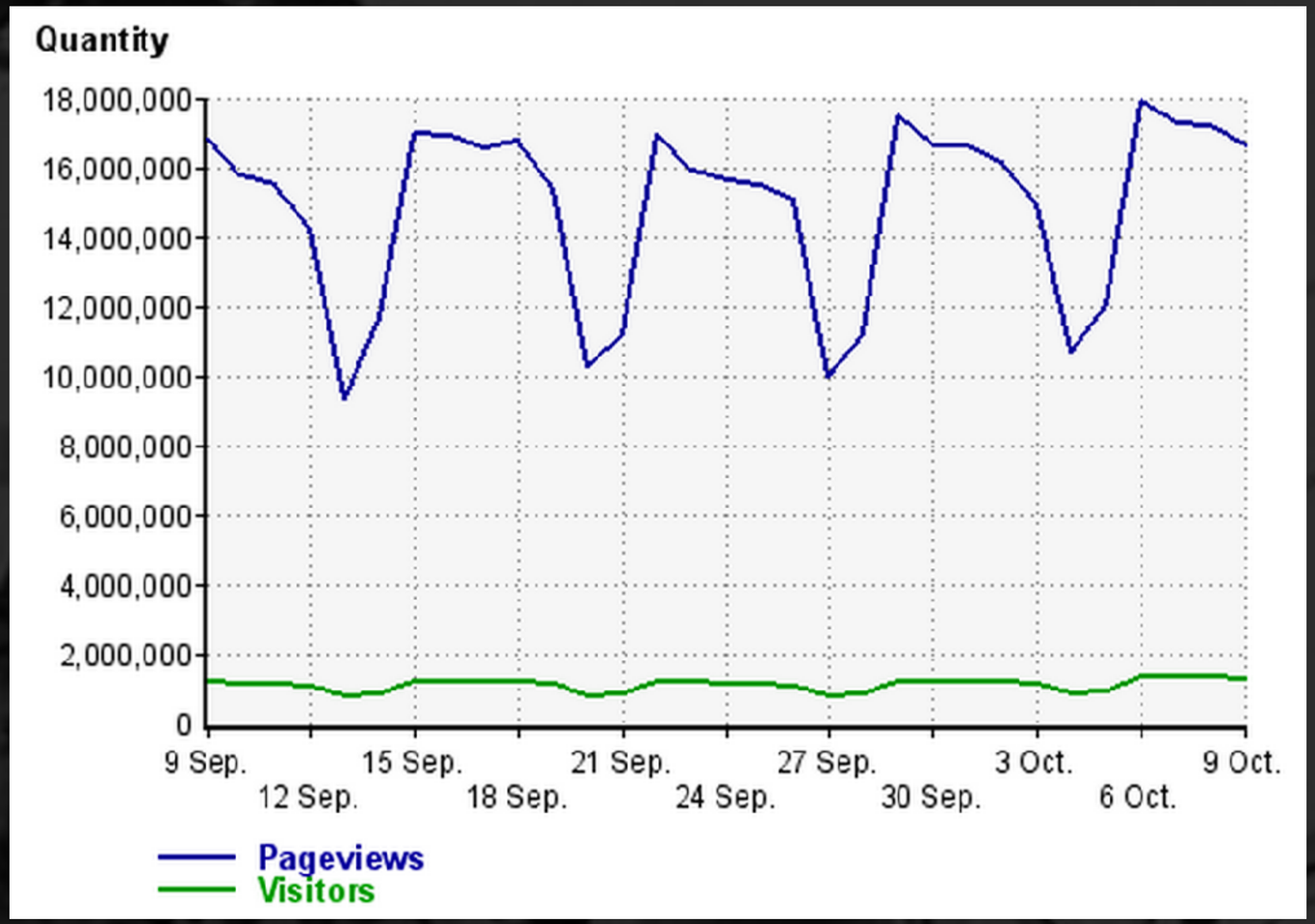
“Highload — это...

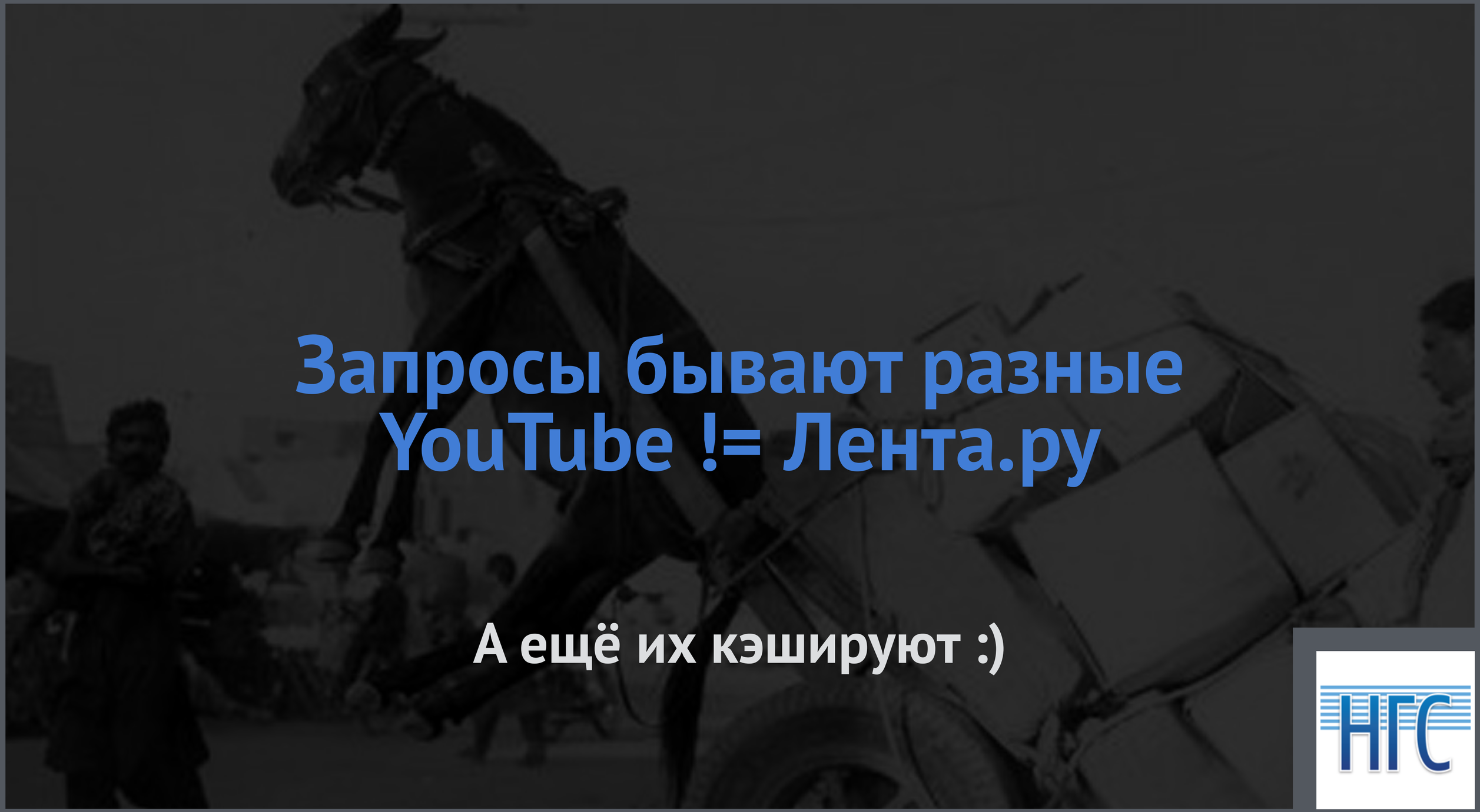
...когда больше миллиона пользователей в час!



На самом деле нет

Пользователи != запросы





Запросы бывают разные
YouTube != Лента.ру

А ещё их кэшируют :)

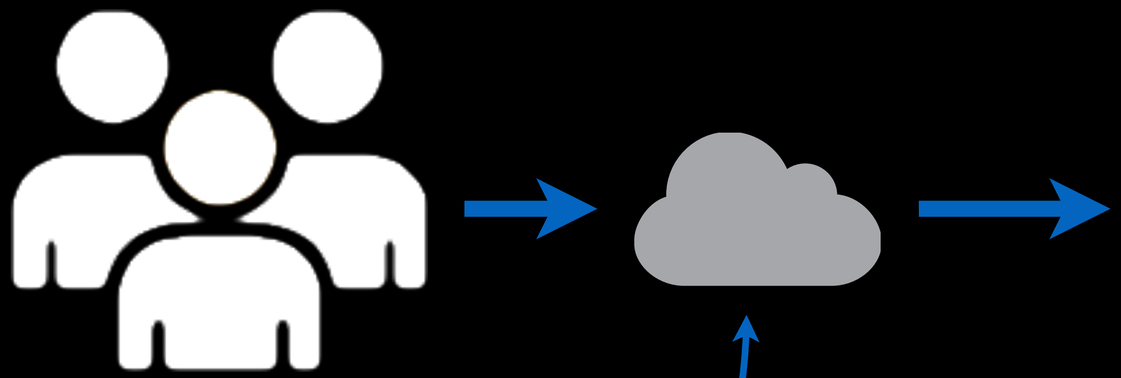
НГС

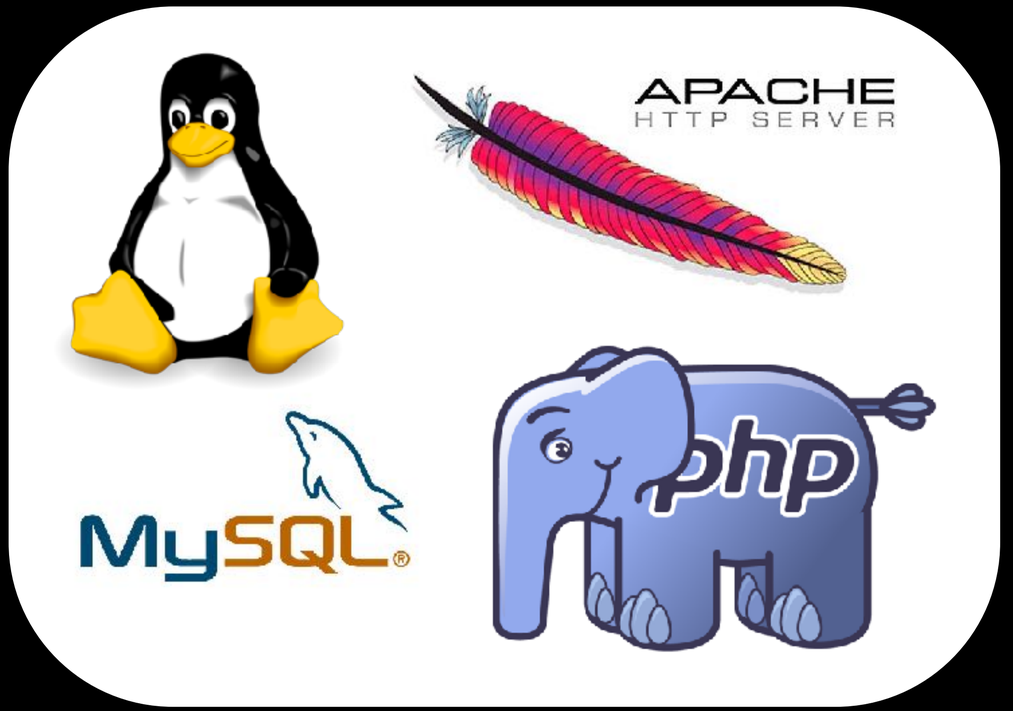
Highload — это когда ресурсов одного сервера уже мало

Секреты высоконагруженных систем

НГС

LAMP’овый пример

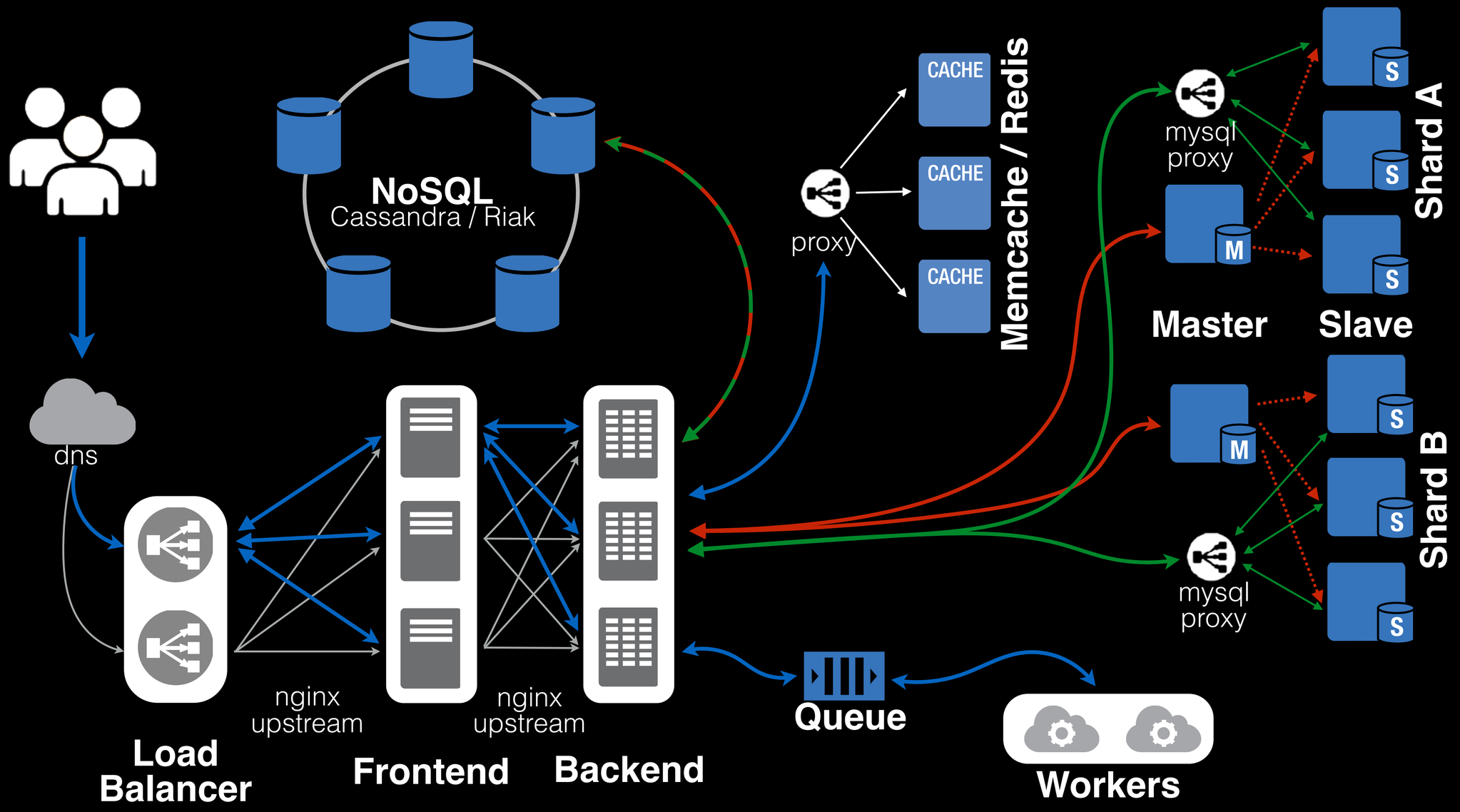




*это интернет

НГС

И что получим из него

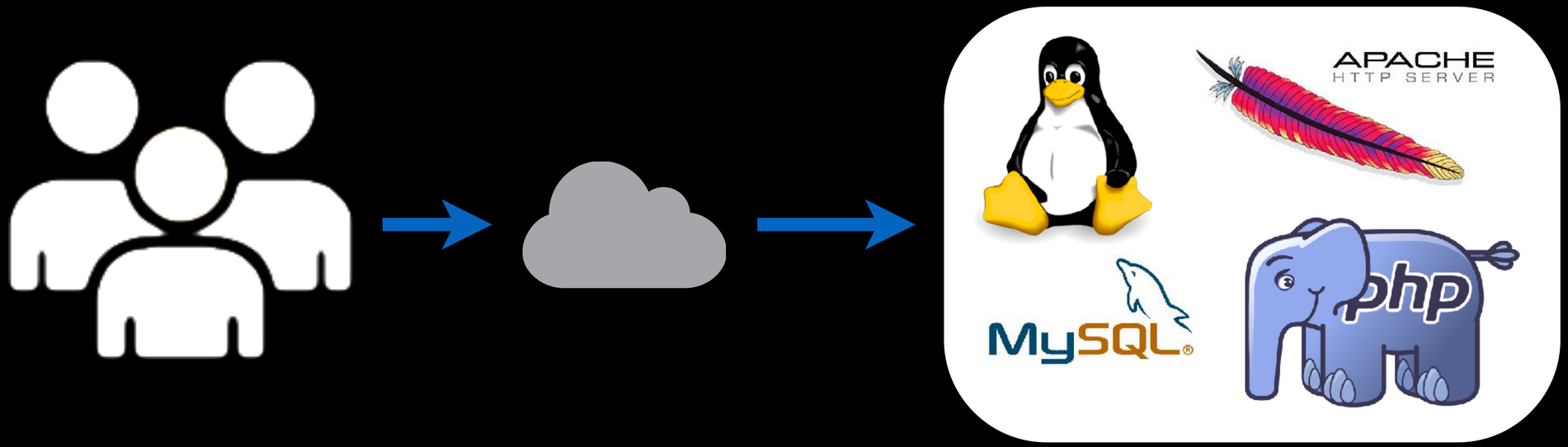


LAMP - Linux, Apache, MySQL, PHP

VPS

(200Mz, 256Mb, 10Gb)

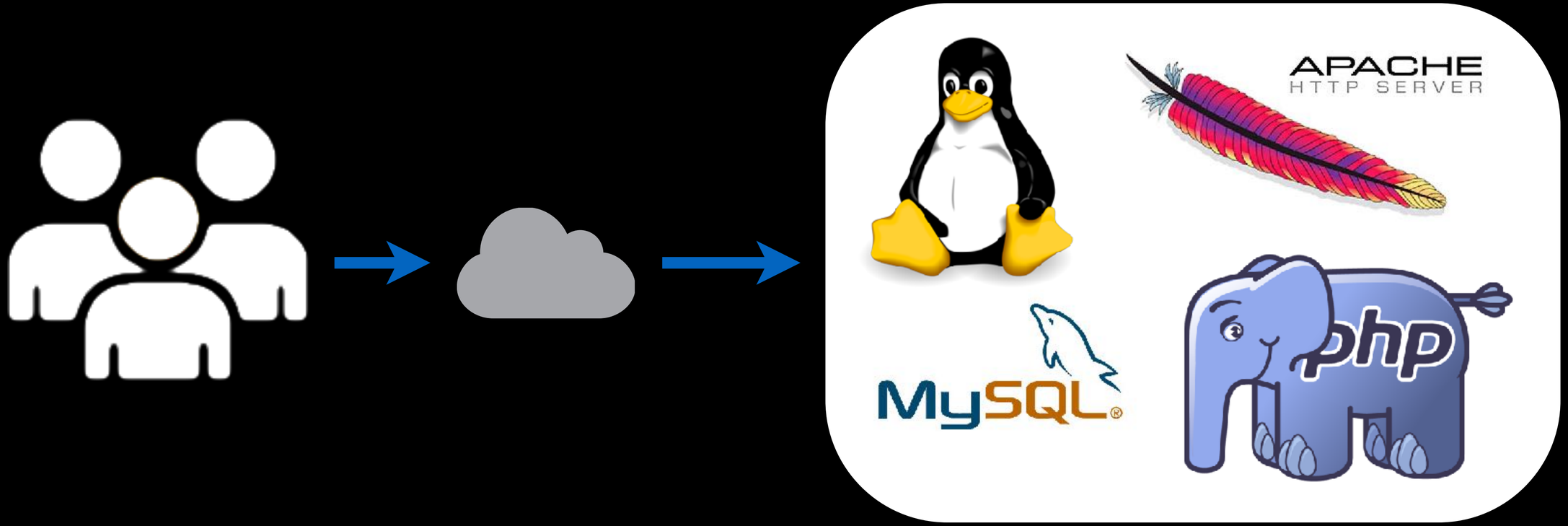
LAMP - Linux, Apache, MySQL, PHP



VDS

(1Gz, 1Gb, 100+Gb)

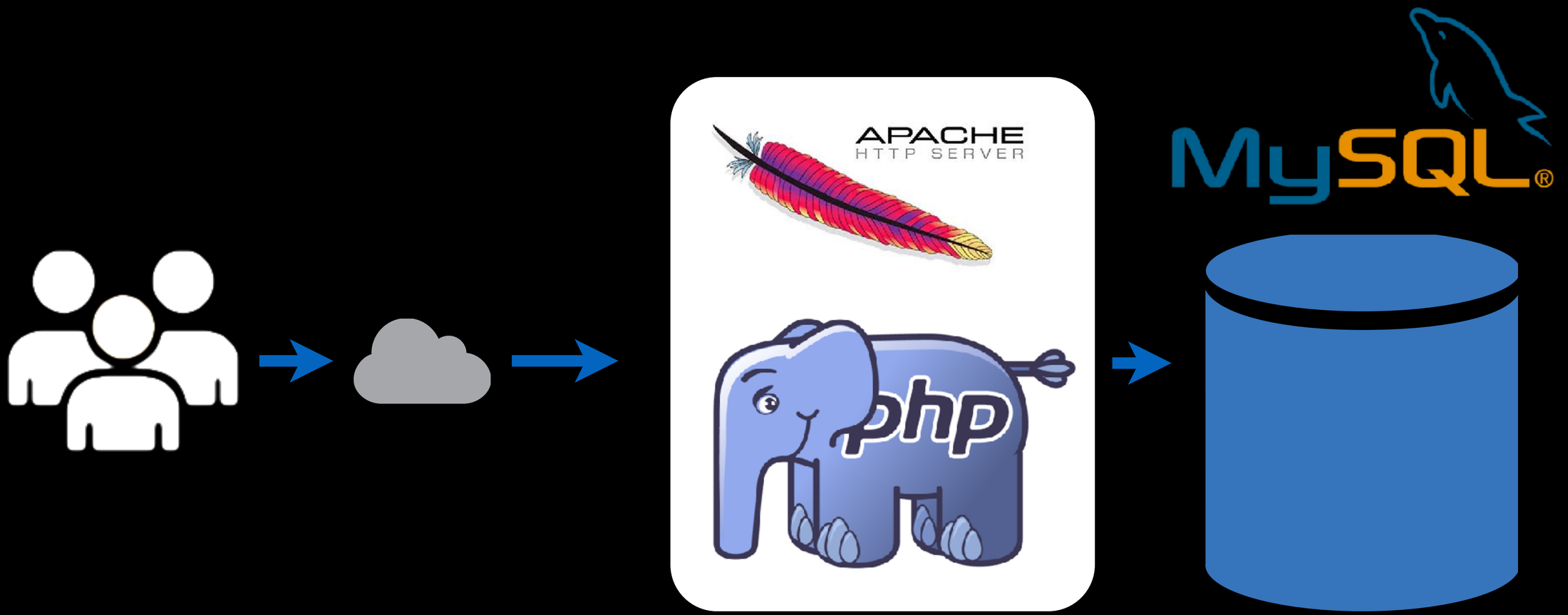
LAMP - Linux, Apache, MySQL, PHP



dedicated server

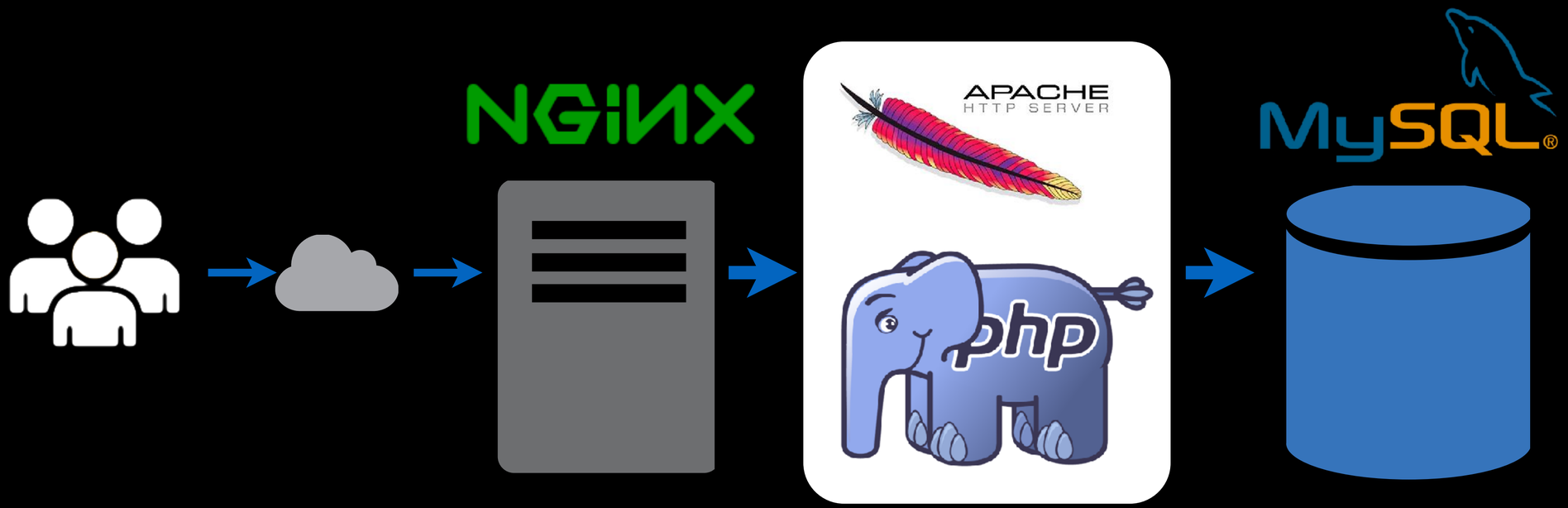
(2+Gz, 16+Gb, 100+Gb)

Выносим базу



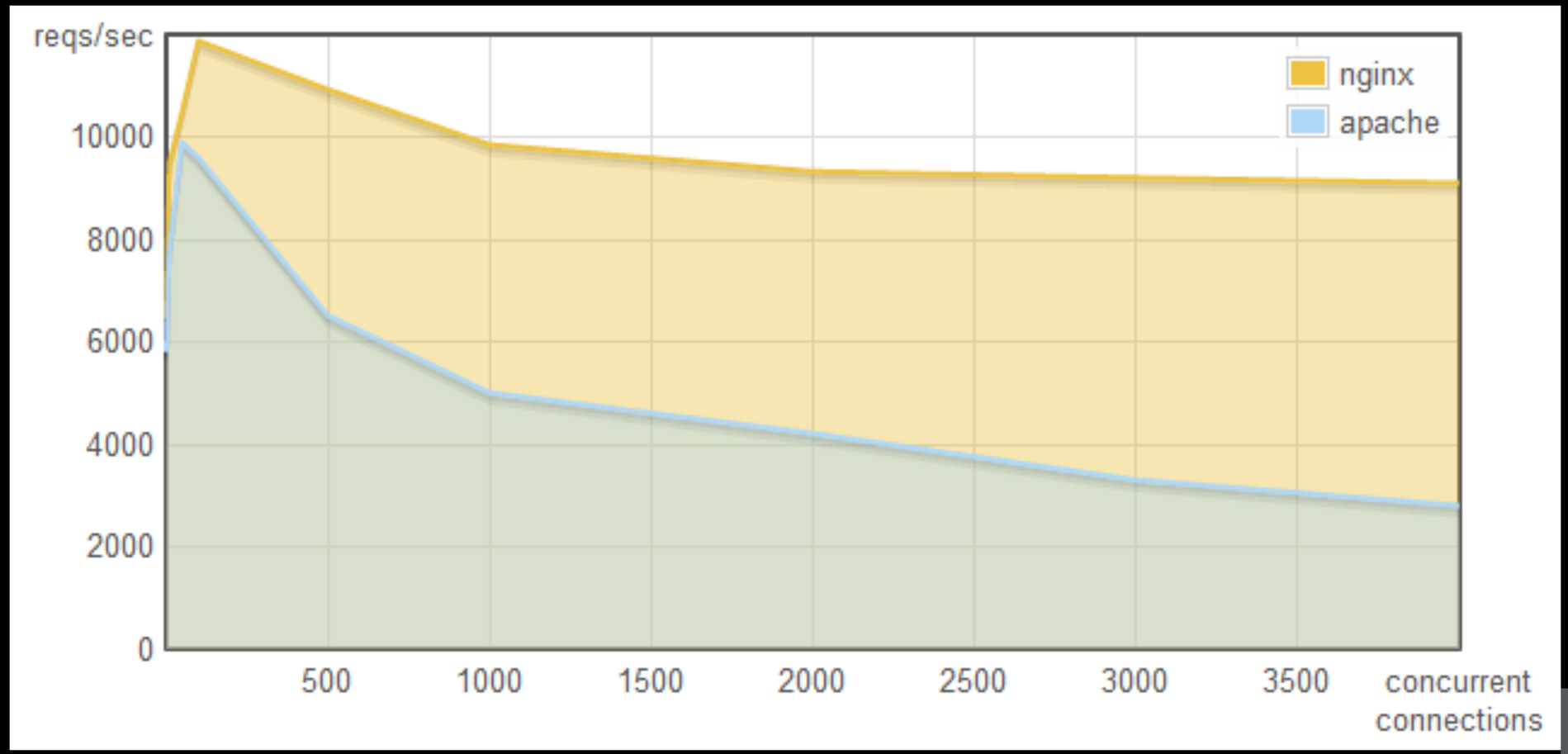
Гуглить:
n-tier arcitecture, mysql configuration

nginx перед apache



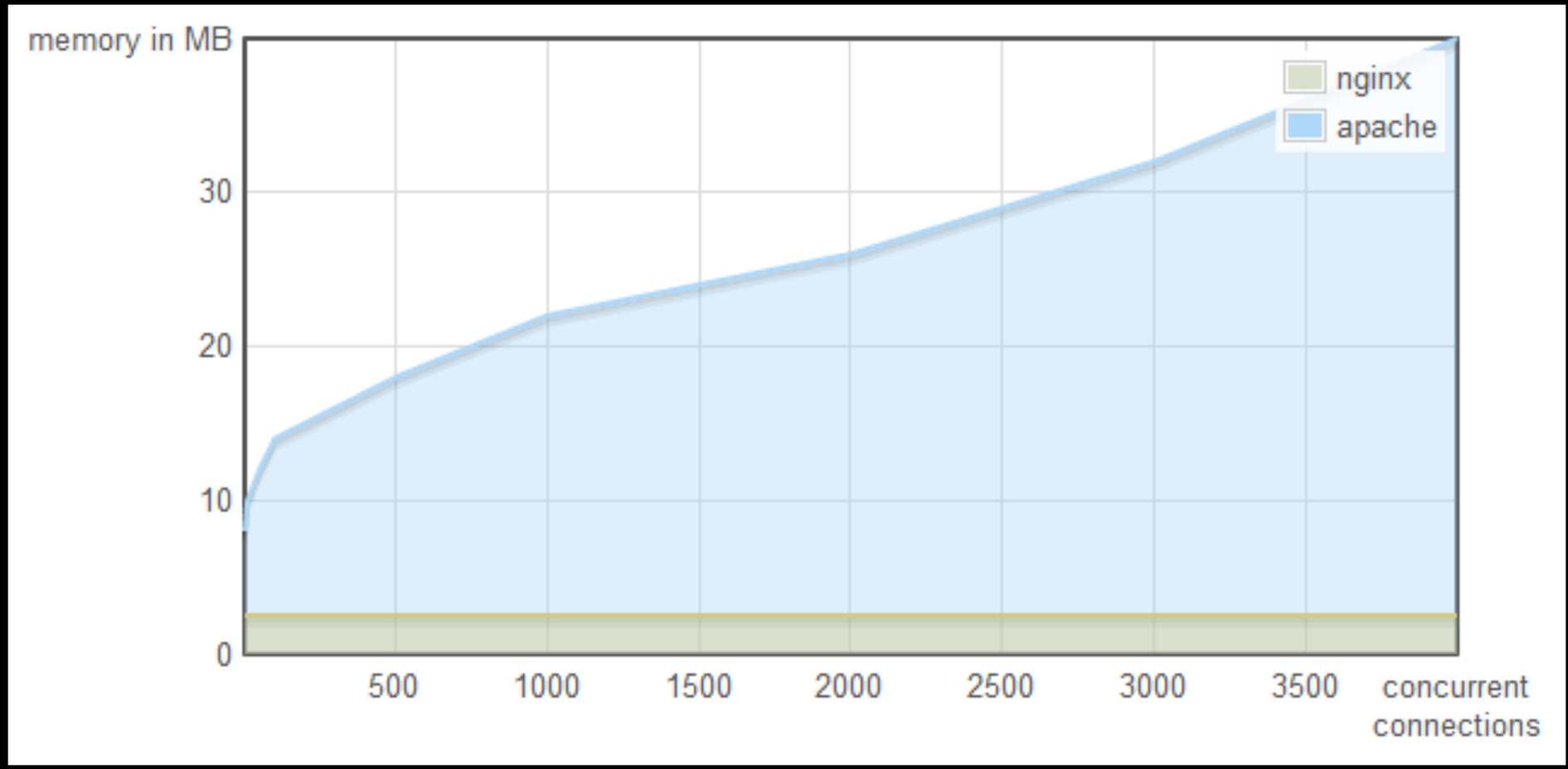
Гуглить:
nginx, libevent, libev, nginx vs apache

nginx перед apache



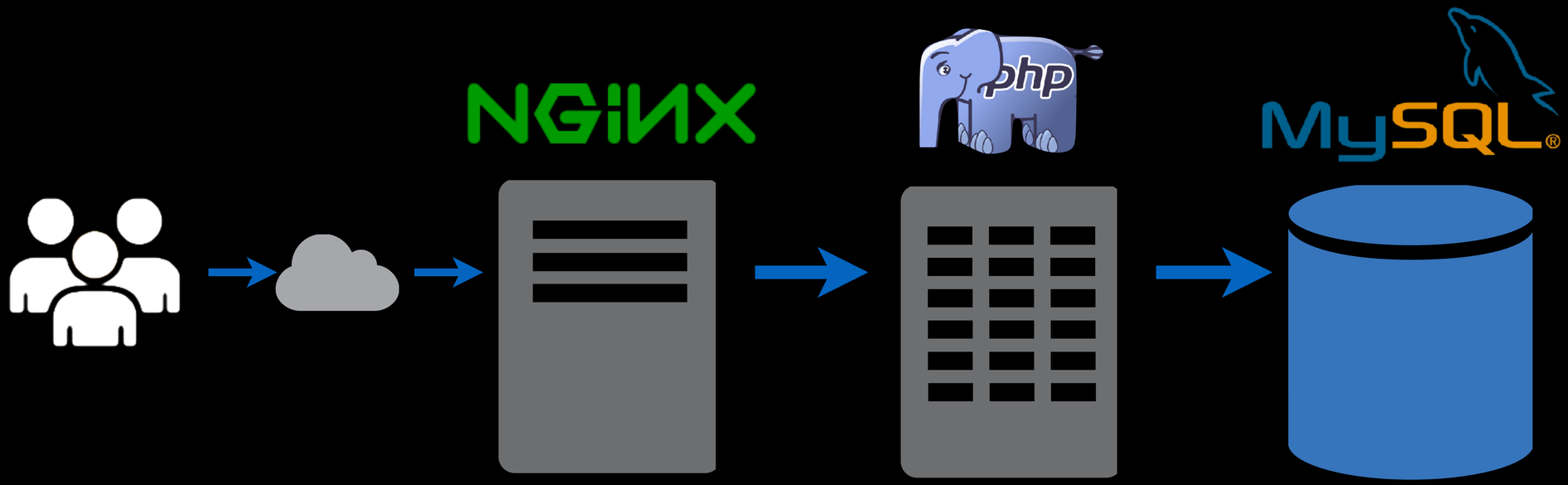
Гуглить:
nginx, libevent, libev, nginx vs apache

nginx перед apache



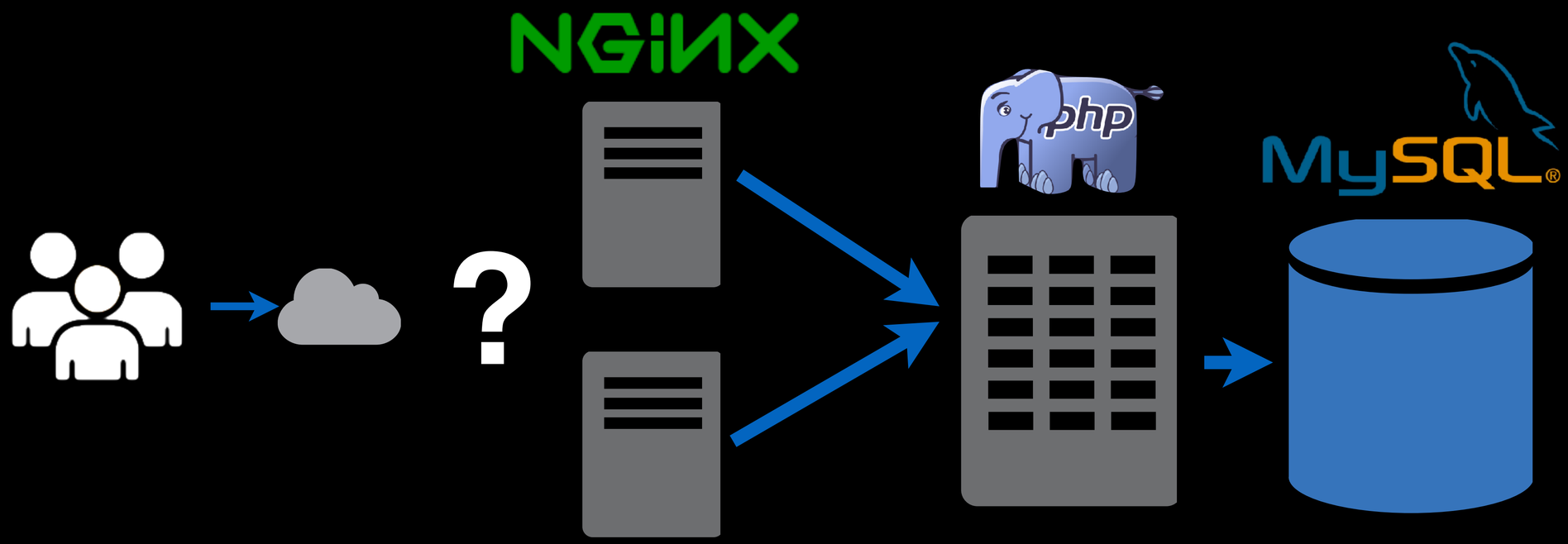
Гуглить:
nginx, libevent, libev, nginx vs apache

А зачем нам apache?



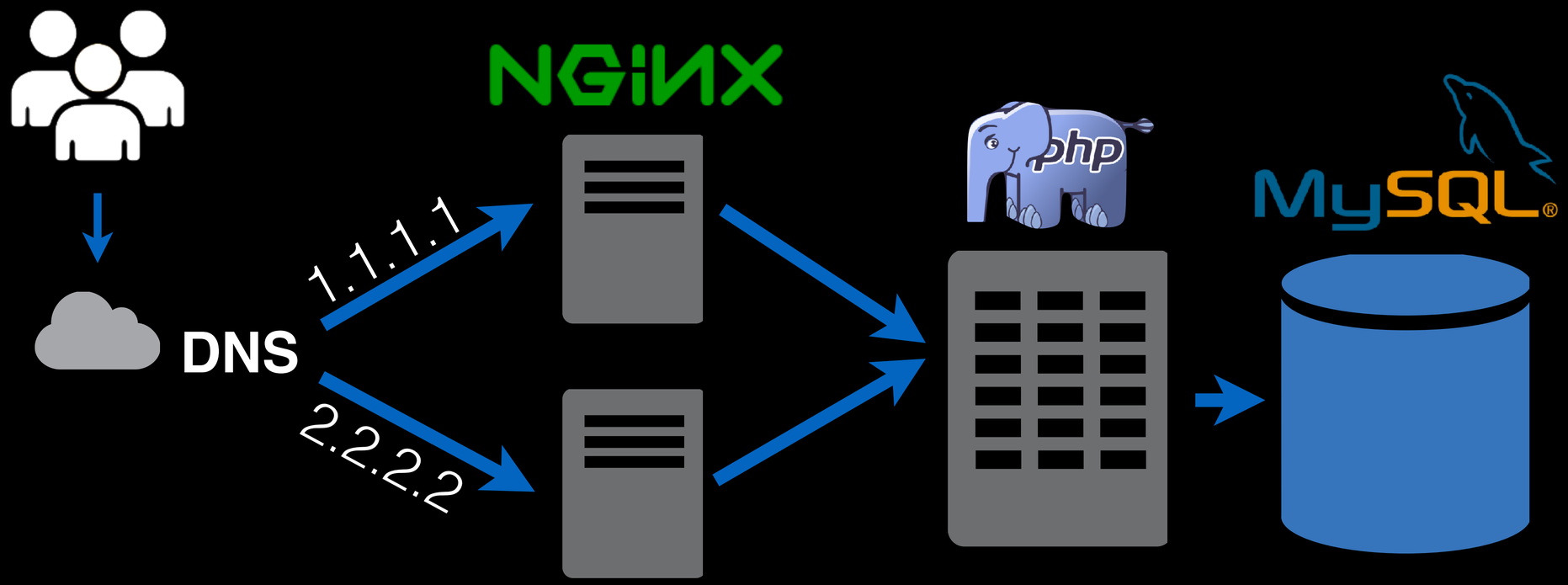
Гуглить:
php-fpm, fastcgi, wsgi

А если у нас много видео?



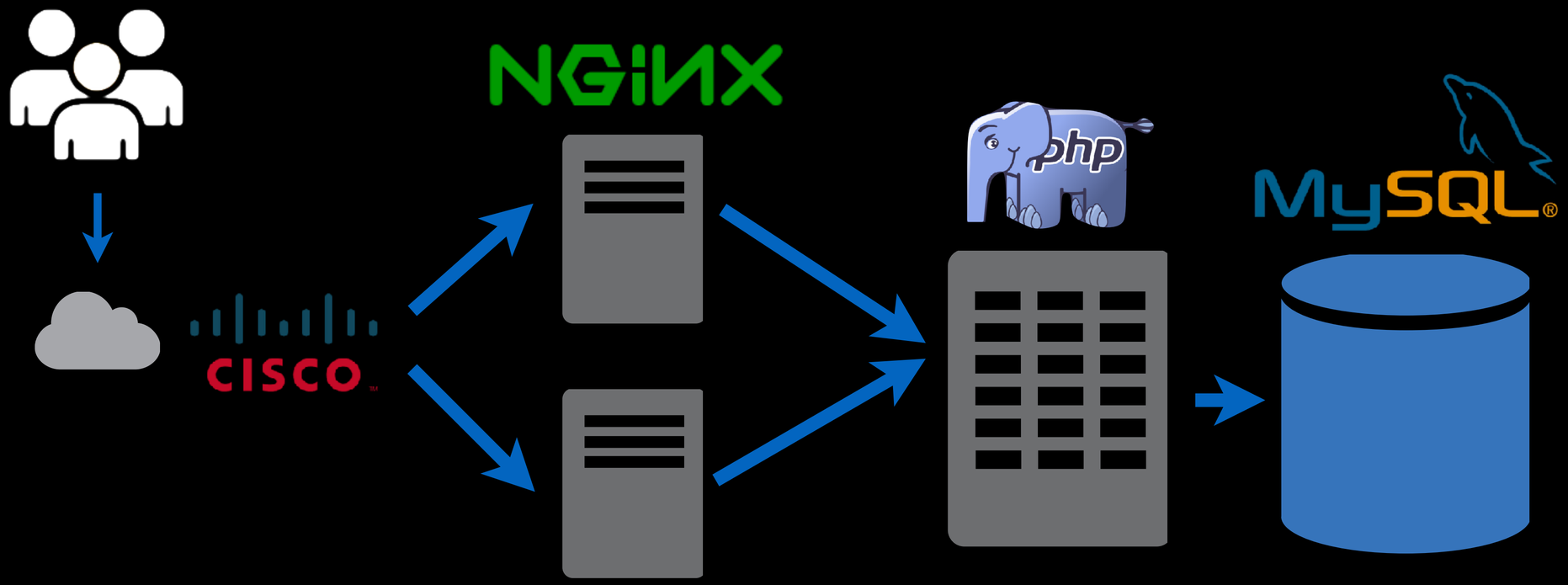
Гуглить:
http load balancer

Балансировка запросов



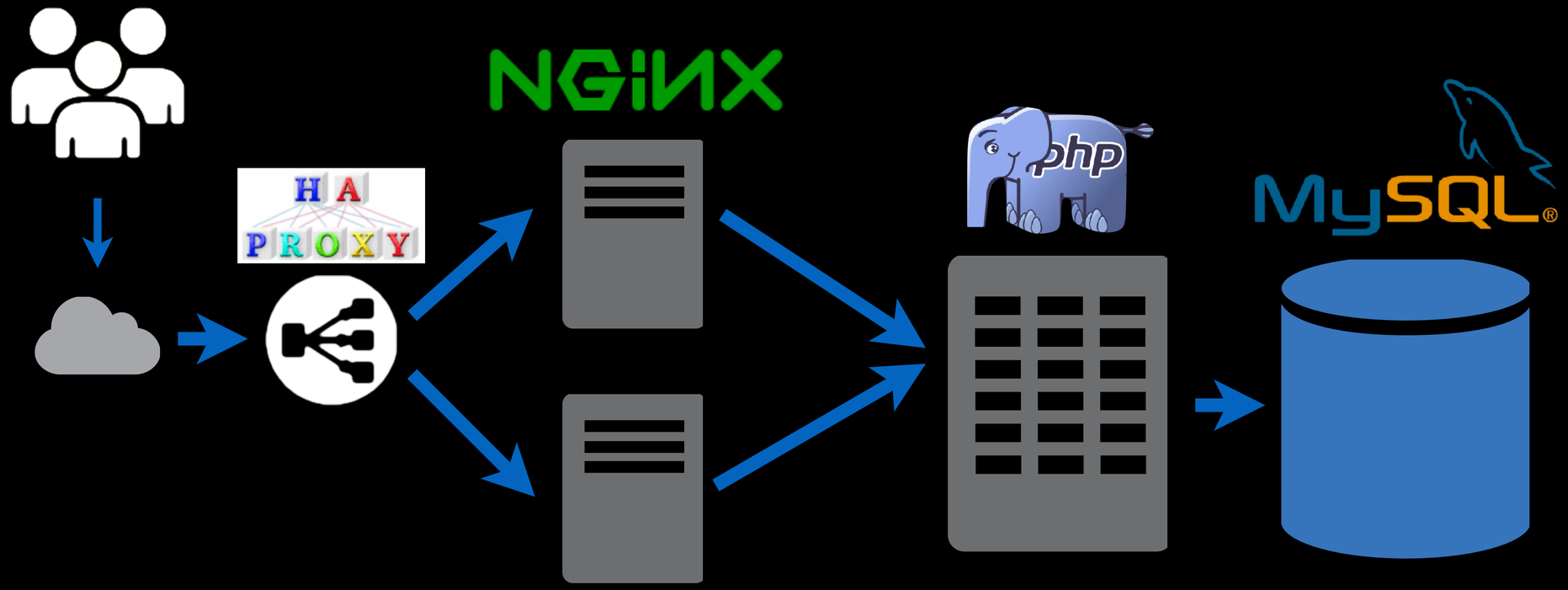
Гуглить:
http load balancer, DNS round-robin, Anycast DNS

Балансировка запросов



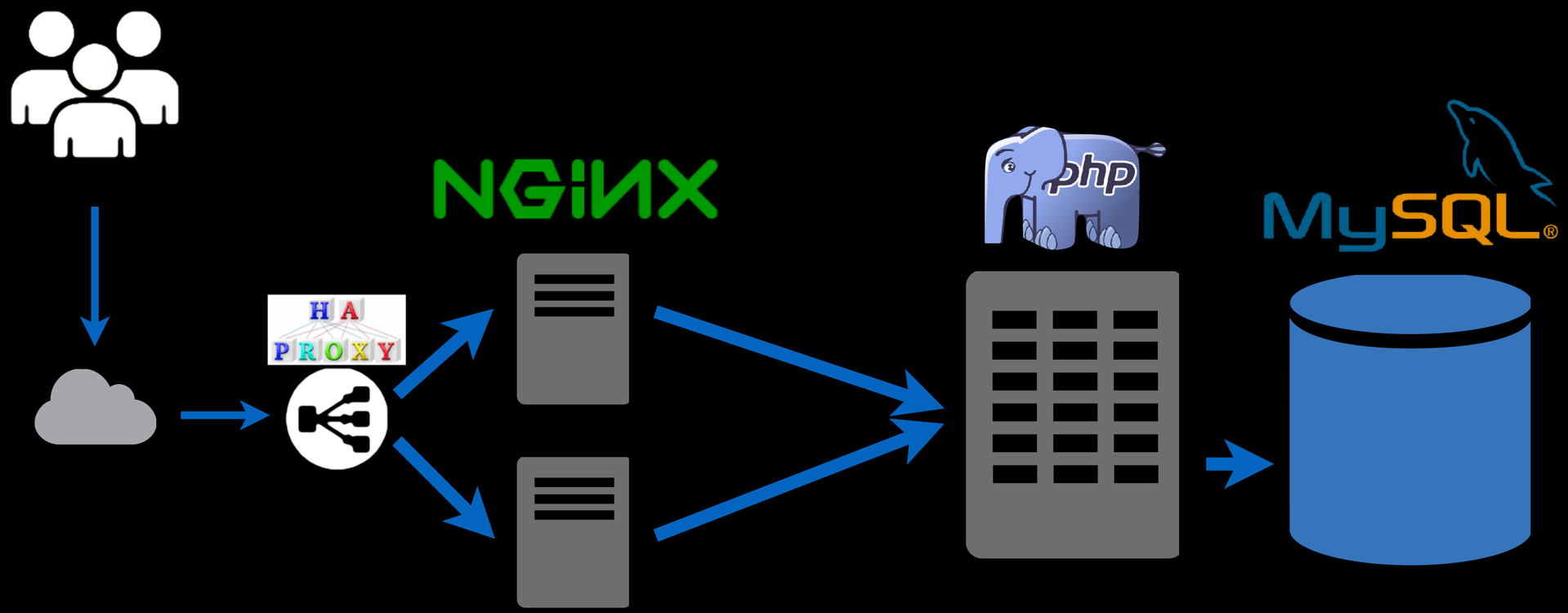
Гуглить:
load balancer, F5, Cisco ACE

Балансировка запросов

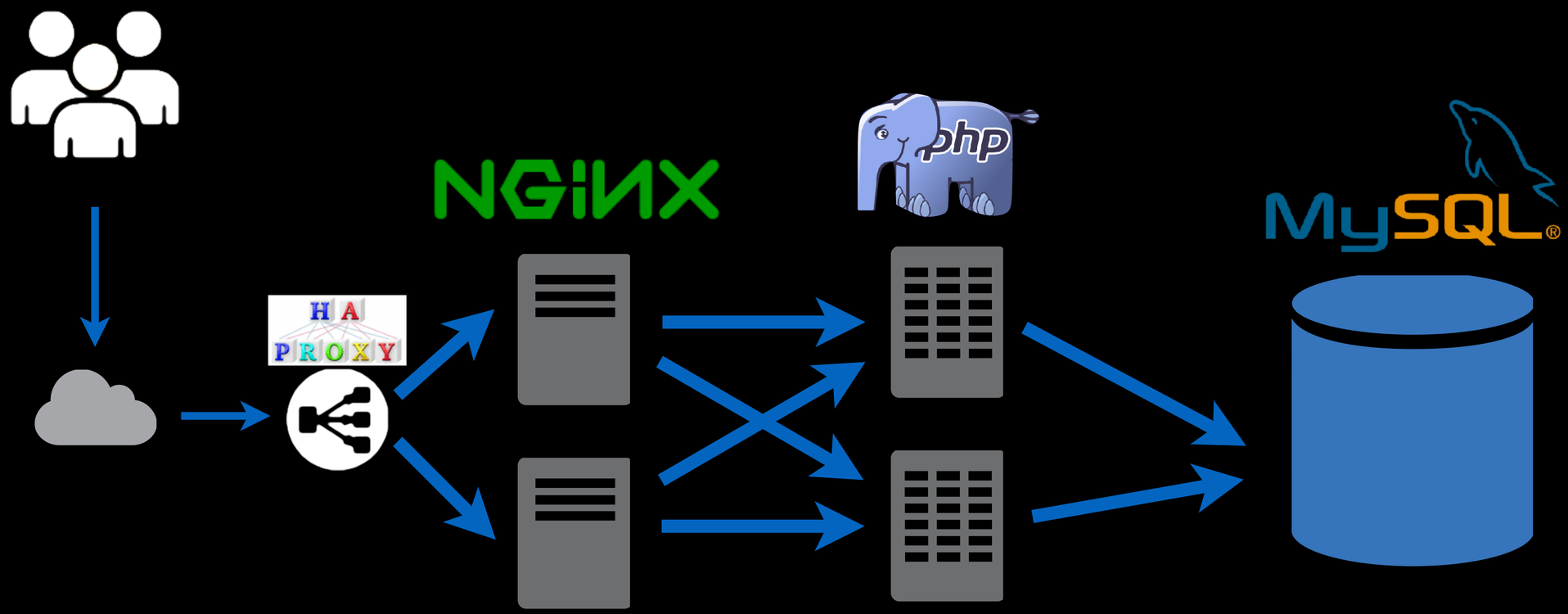


Гуглить:
tcp load balancer, haproxy, keepalived

Промежуточный итог



Масштабируем backend



Гуглить:
nginx upstream, nginx fastcgi_pass, nginx proxy_pass

НГС

Масштабируем backend

- Самое простое, поставить рядом второй сервер
- Меньше запросов к backend'у - меньше нагрузка
- Кэшируем всё!

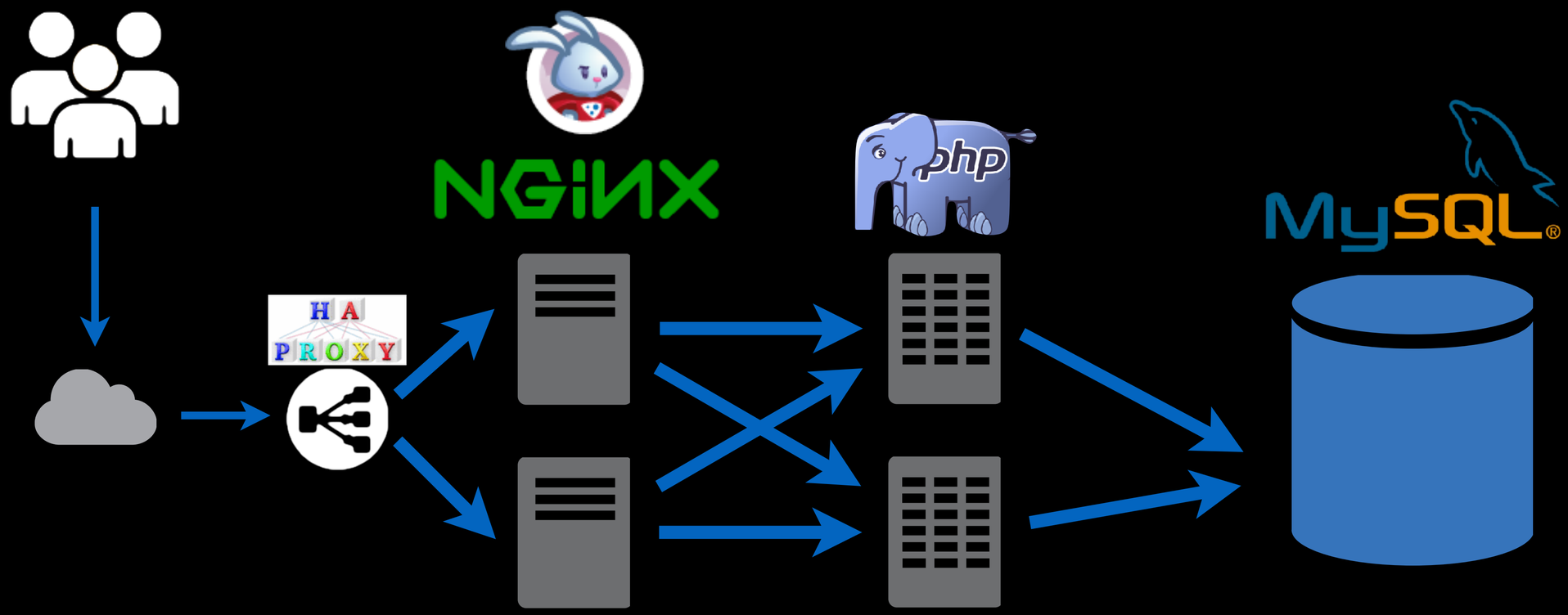
Гуглить:
Nginx proxy_cache, Nginx fastcgi_cache,
Varnish, Apache Traffic Server

Масштабируем backend

- Самое простое, поставить рядом второй сервер
- Меньше запросов к backend'у - меньше нагрузка
- Кэшируем всё!

Гуглить:
Nginx proxy_cache, Nginx fastcgi_cache,
Varnish, Apache Traffic Server

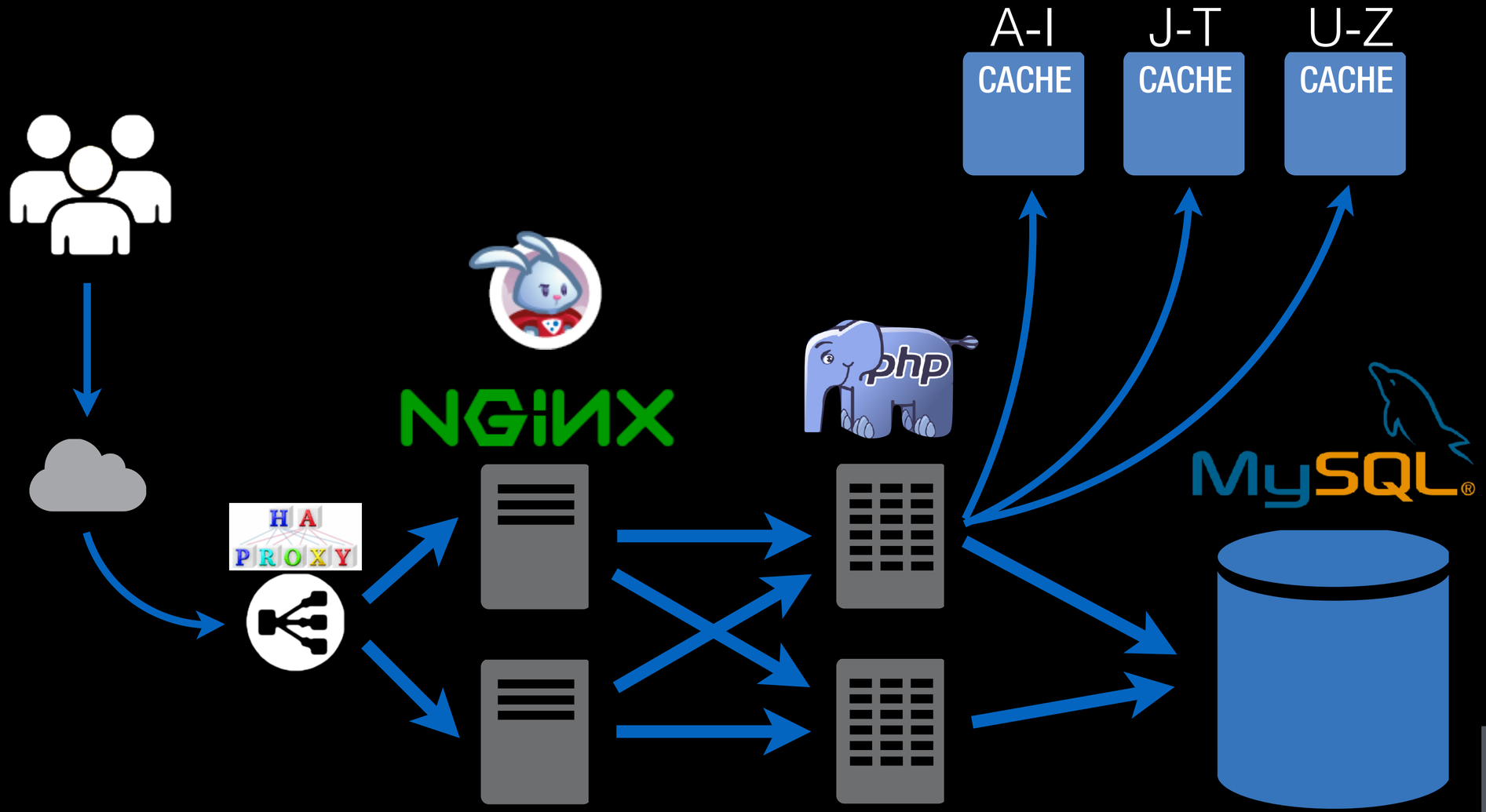
Добавим кэша



Гуглить:
Nginx proxy_cache, Nginx fastcgi_cache,
Varnish, Apache Traffic Server

НГС

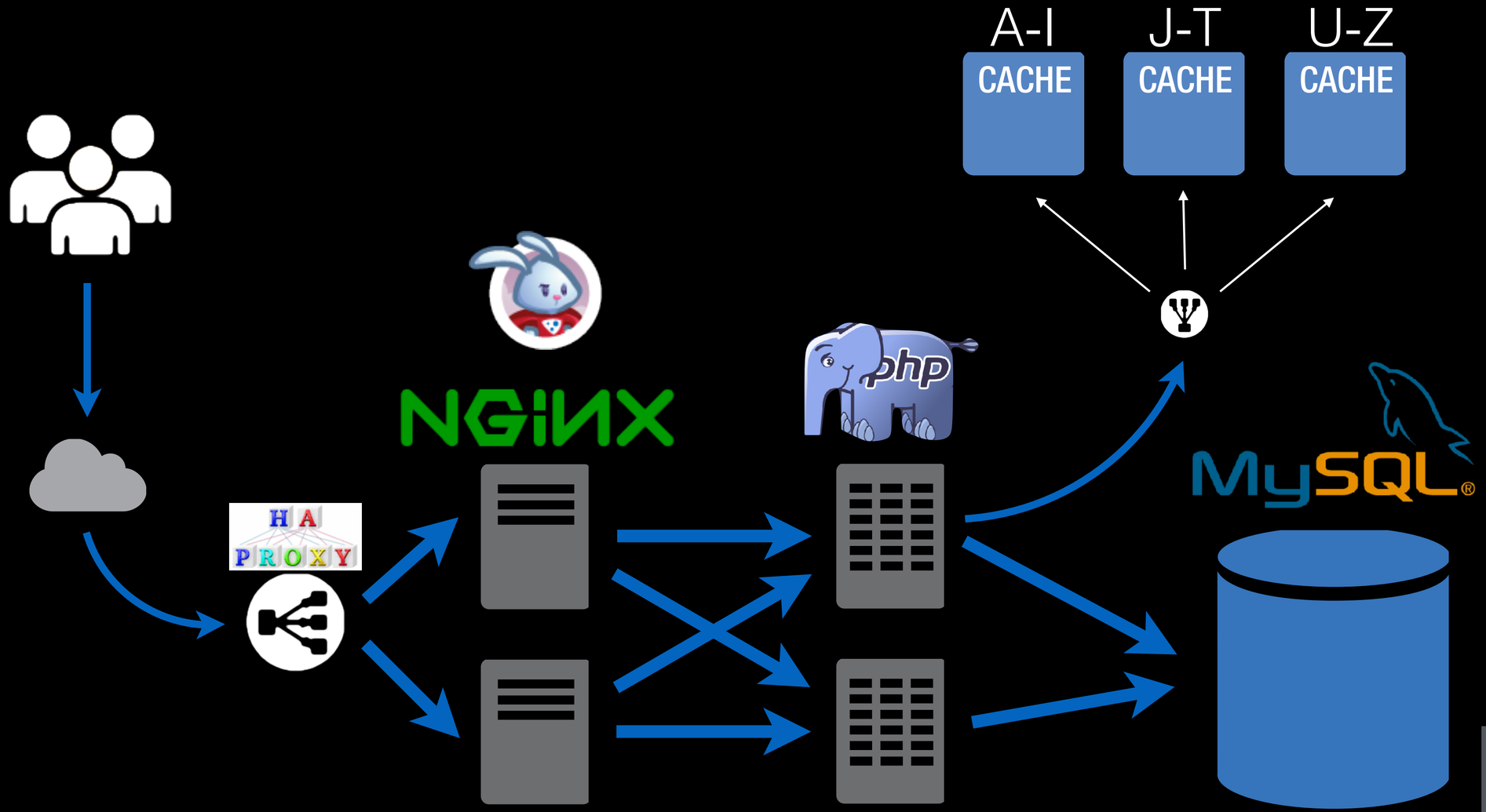
Добавим кэша



Гуглить:
memcache, redis, key-value store, libketama, consistent hashing

НГС

Добавим кэша

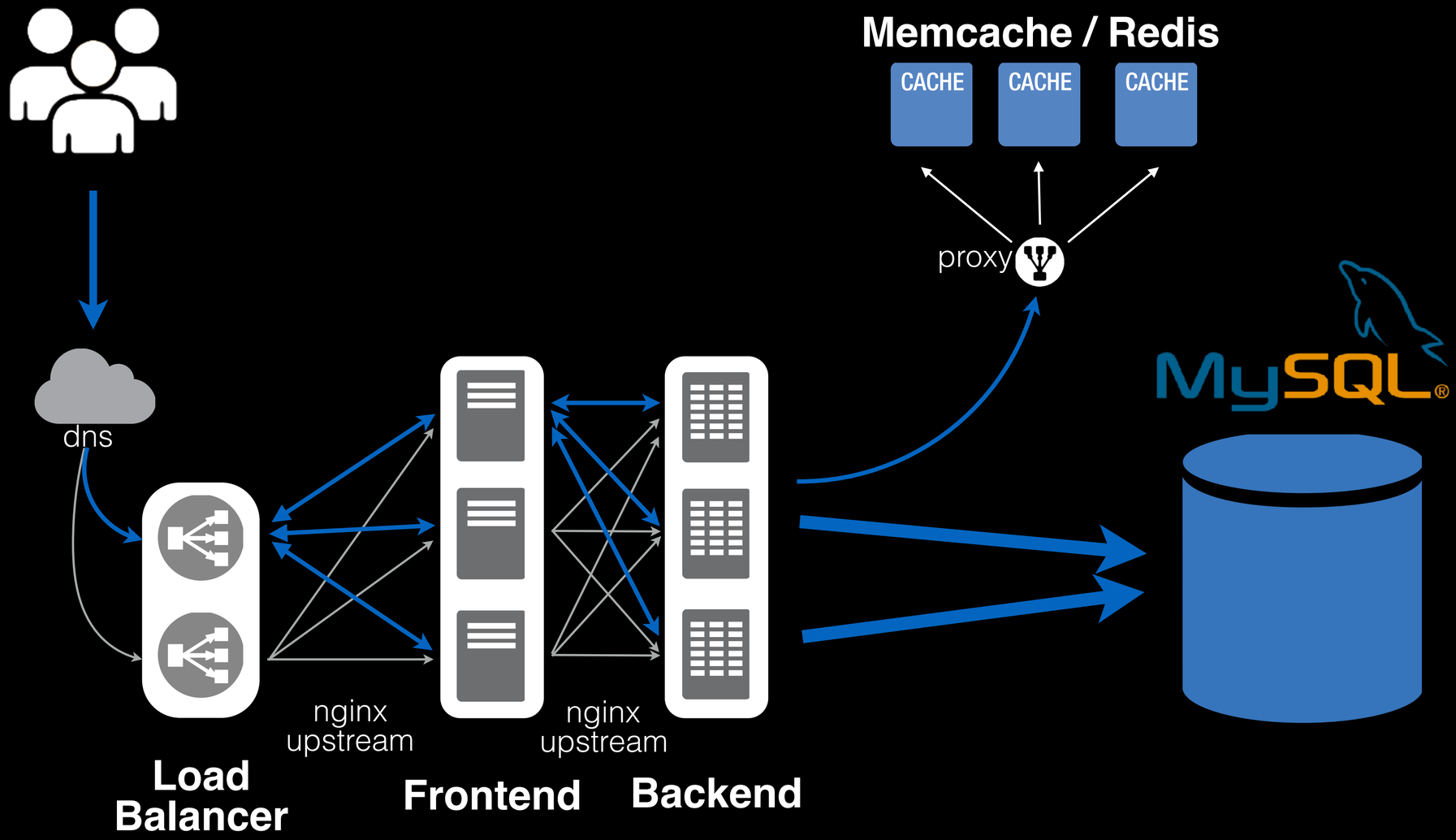


Гуглить:
twemproxy, mcrouter, haproxy

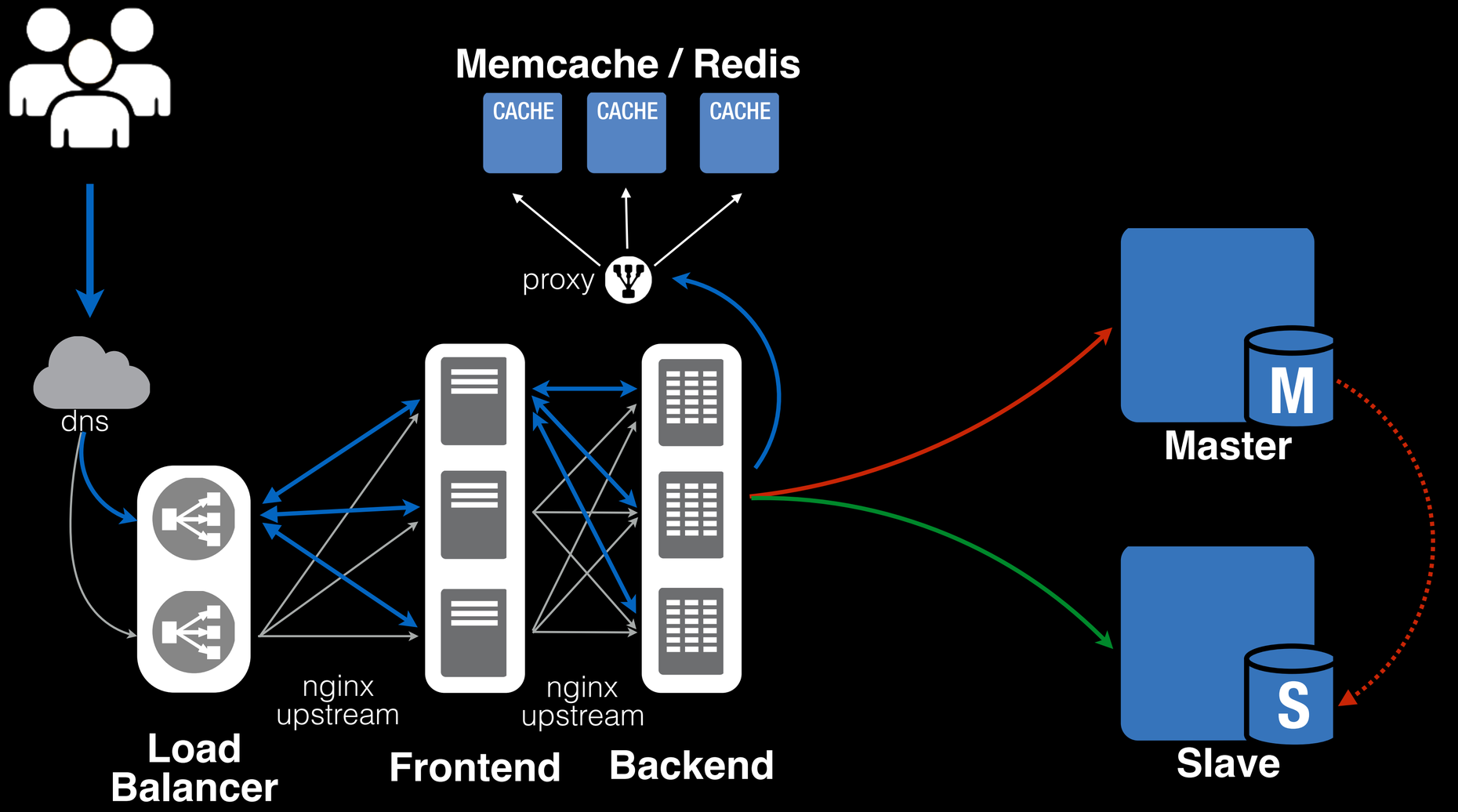
Кэширование

- Кэши не страшно потерять, поэтому применяем шардинг
- Для лучшей отказоустойчивости используем специализированные прокси - twemproxy, mcrouter
- Помним про race-condition
- Экспериментально определяем размер кэша
- Обязательно мониторить hit/miss

Что получилось



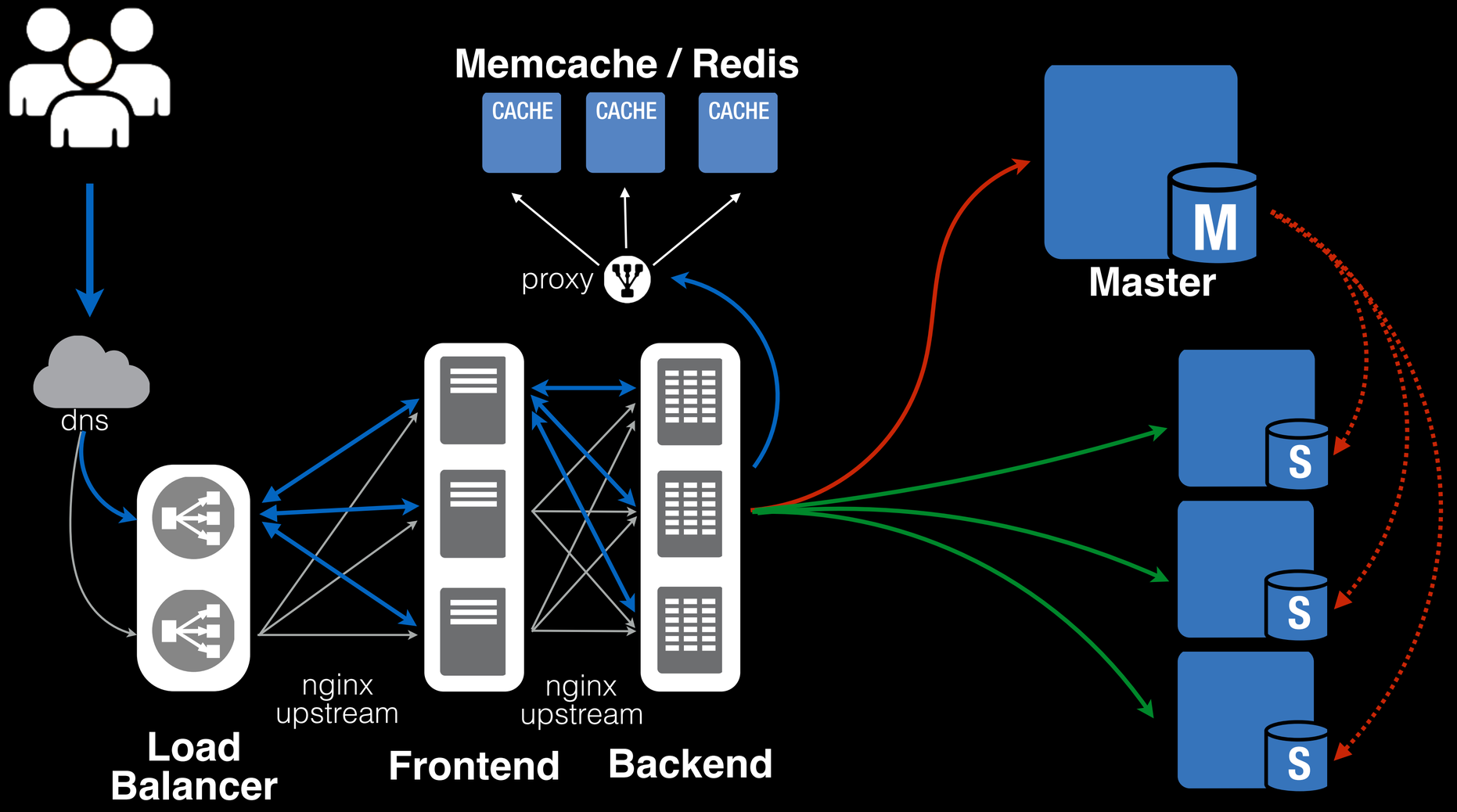
Репликация



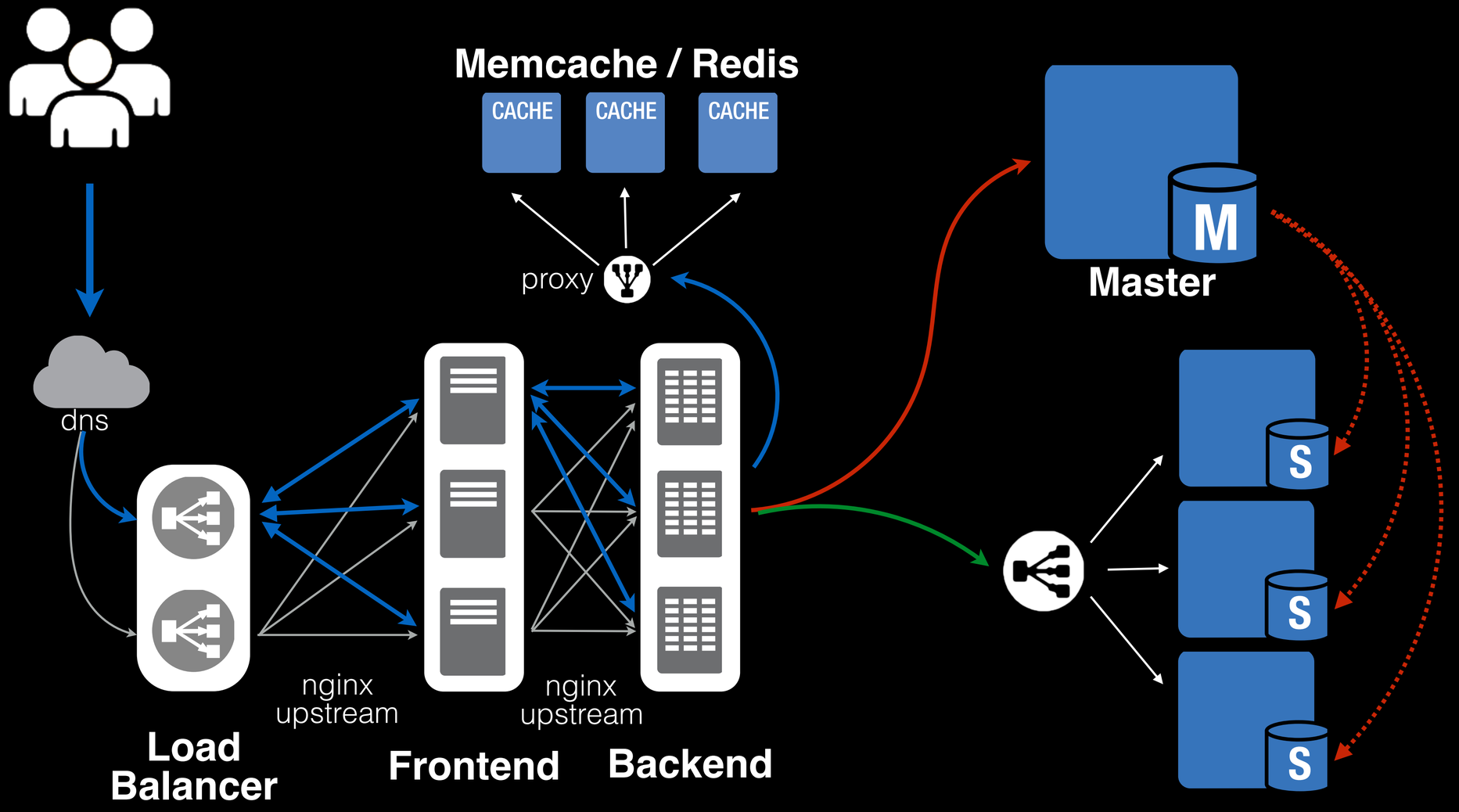
Гуглить:

mysql binlog, row-based/statement-based replication

Репликация


Memcache / Redis
CACHE
CACHE
CACHE
proxy
M
Master
dns
S
S
S
nginx
upstream
nginx
upstream
Load
Balancer
Frontend
Backend
Slave

Репликация

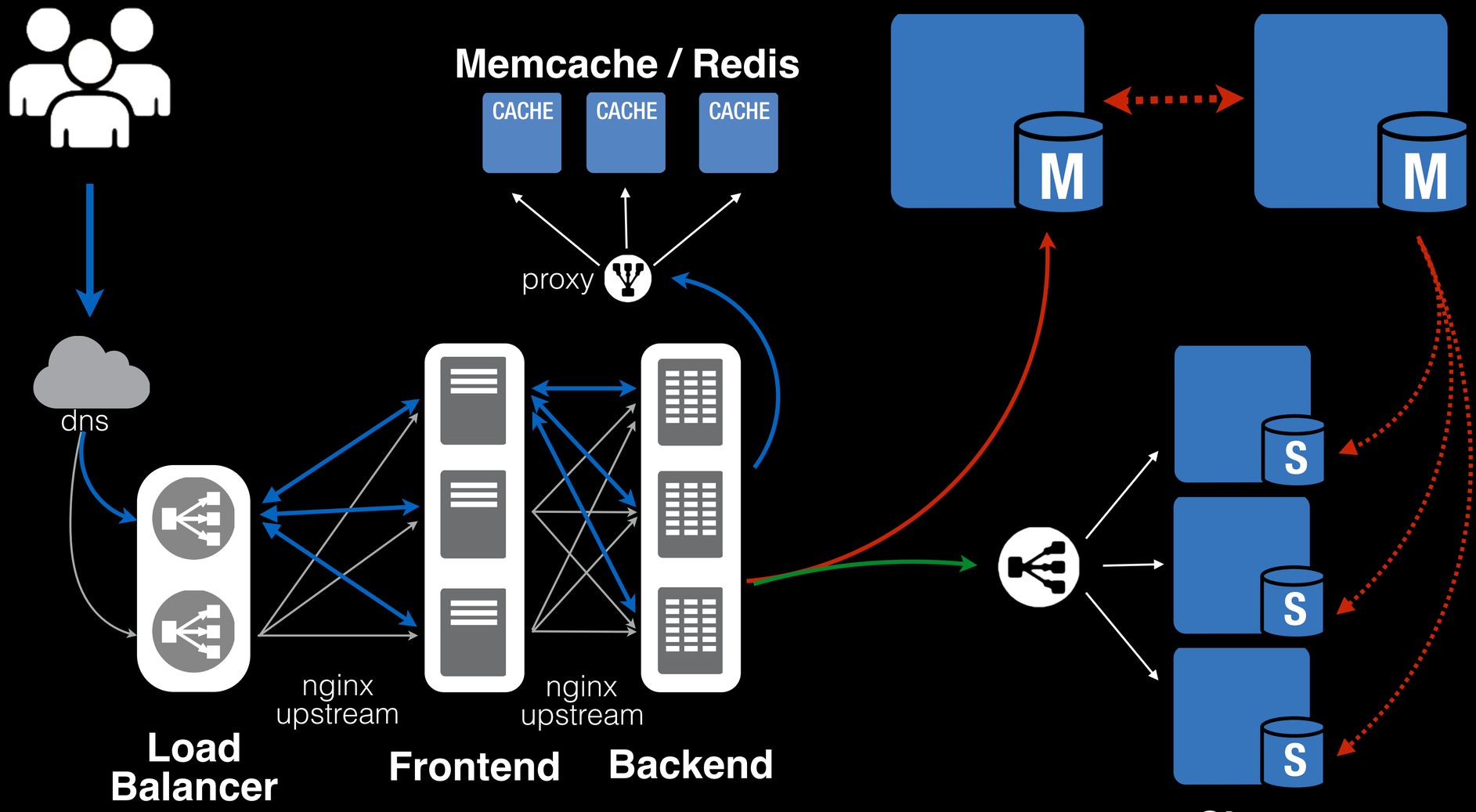


Гуглить:
haproxy, mysql-proxy, pgpool

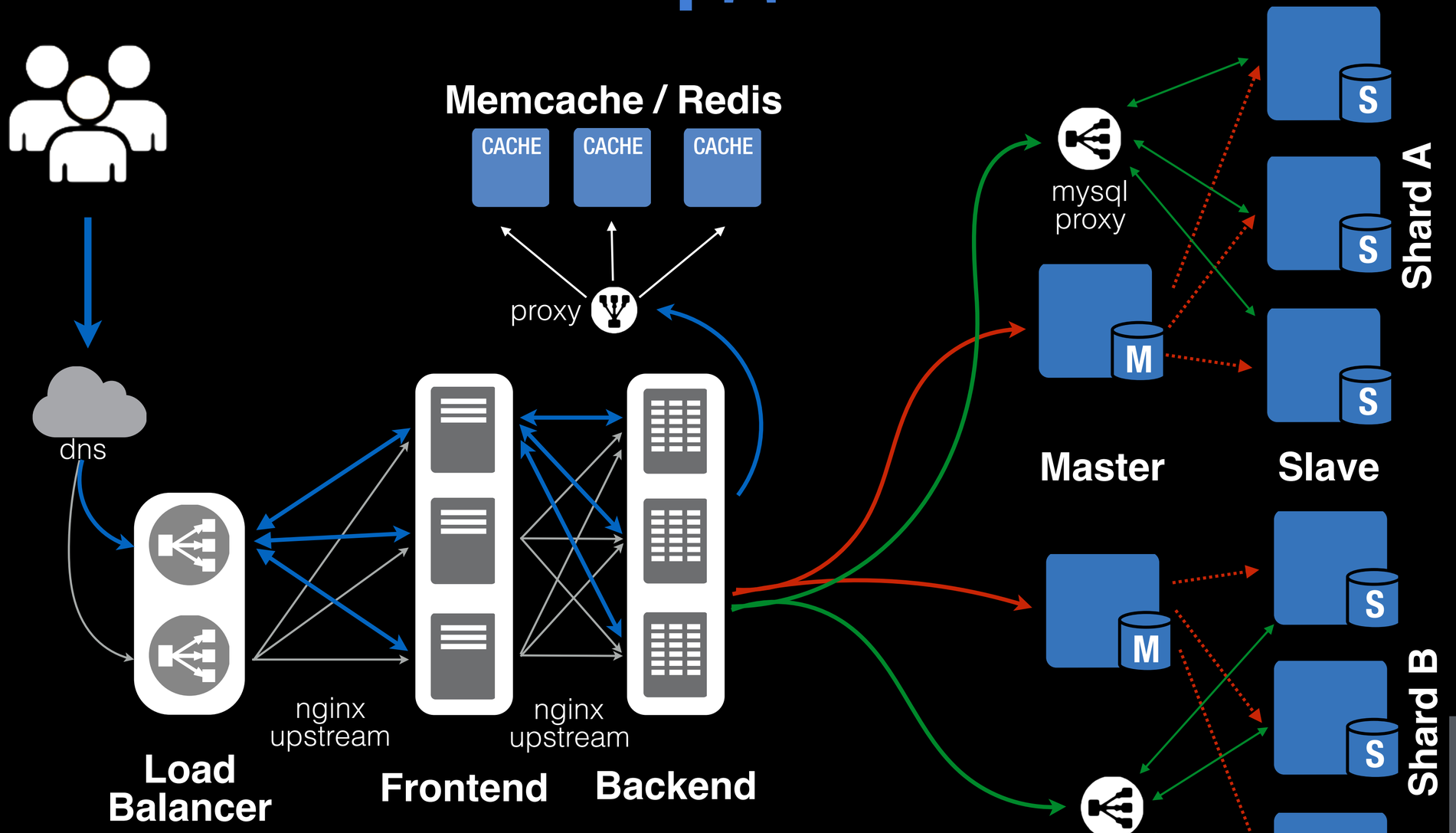
Нюансы репликации (MySQL)

- Репликация может лагать
- Нужно быть аккуратным с недетерминированными запросами (NOW(), RAND() и т.д.)
- В случае падения мастера нужна ручная перестройка схемы
- Стоит регулярно смотреть в slowlog
- Масштабирует только чтение

master-master репликация



Шардинг



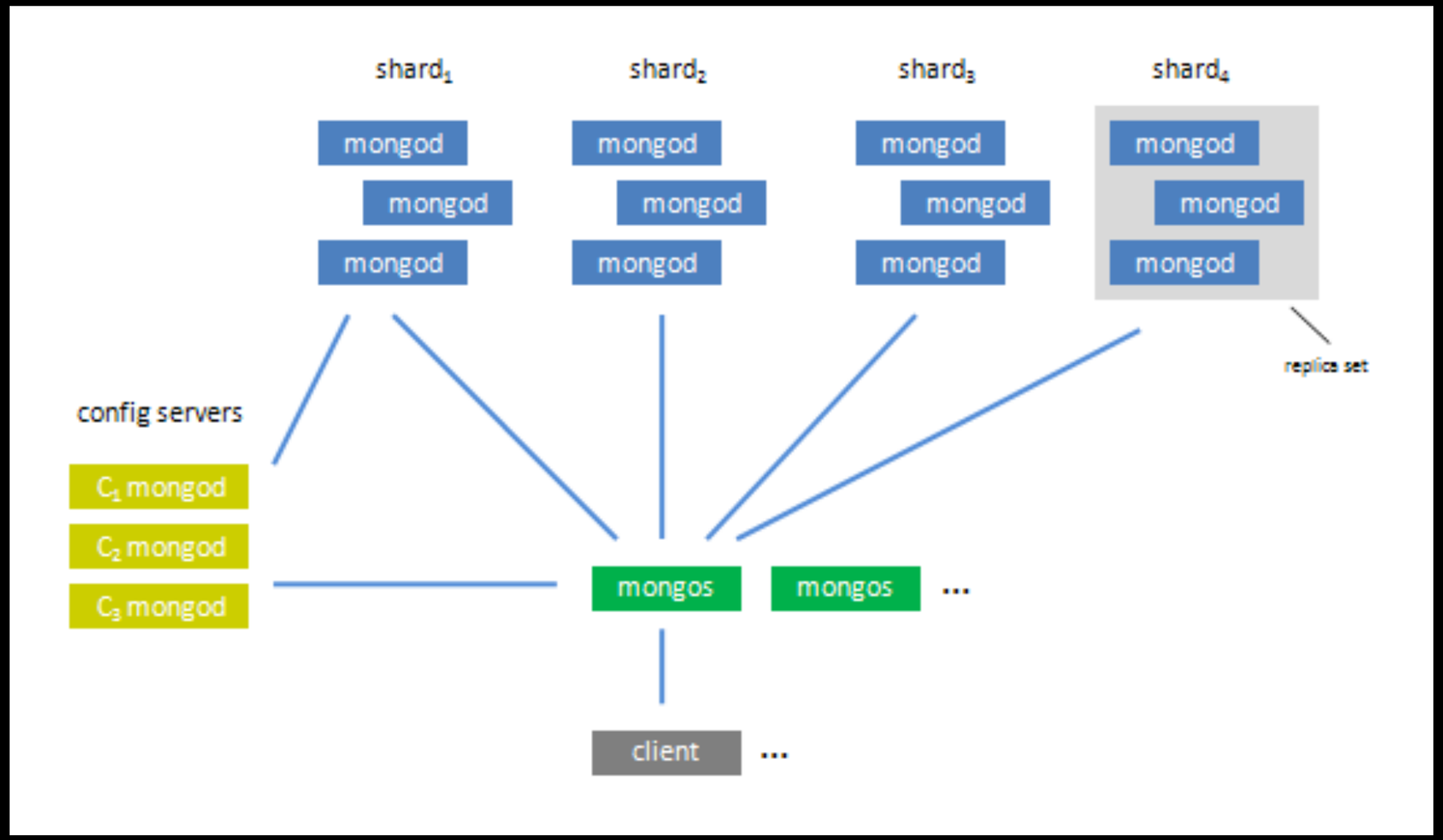
Гуглить:
mysql cluster, mysql fabric

Шардинг

- Очень важно выбрать правильный ключ
- В случае с транзакциями боль
- Насколько прост ввод/вывод шарда
- Записи масштабируются, но это требует много ресурсов
- Ещё куча нюансов :)

Но не забываем про NoSQL :)

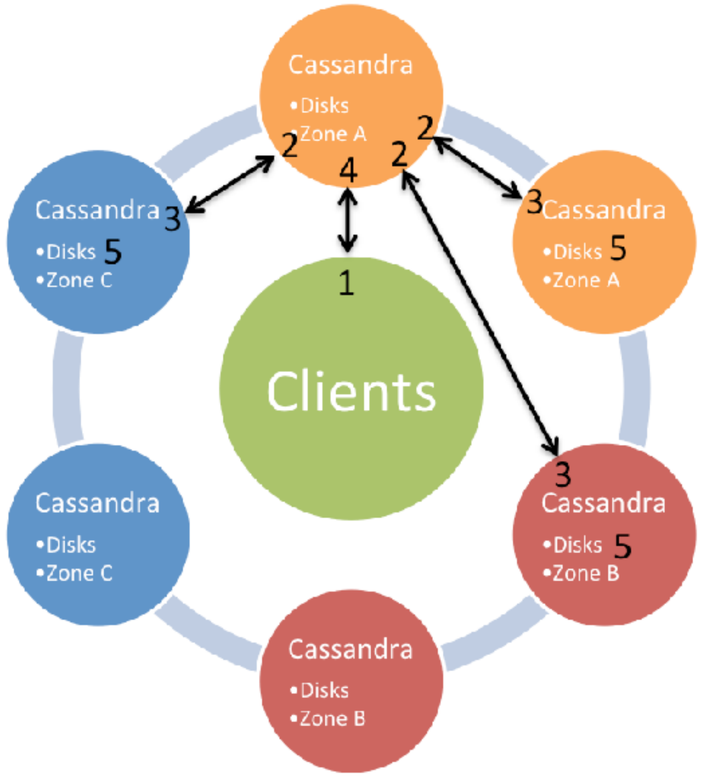
Шардинг в MongoDB



Гуглить:
mongodb, mongodb sharding

Cassandra

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk



If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

SSTable disk writes and compactions occur asynchronously

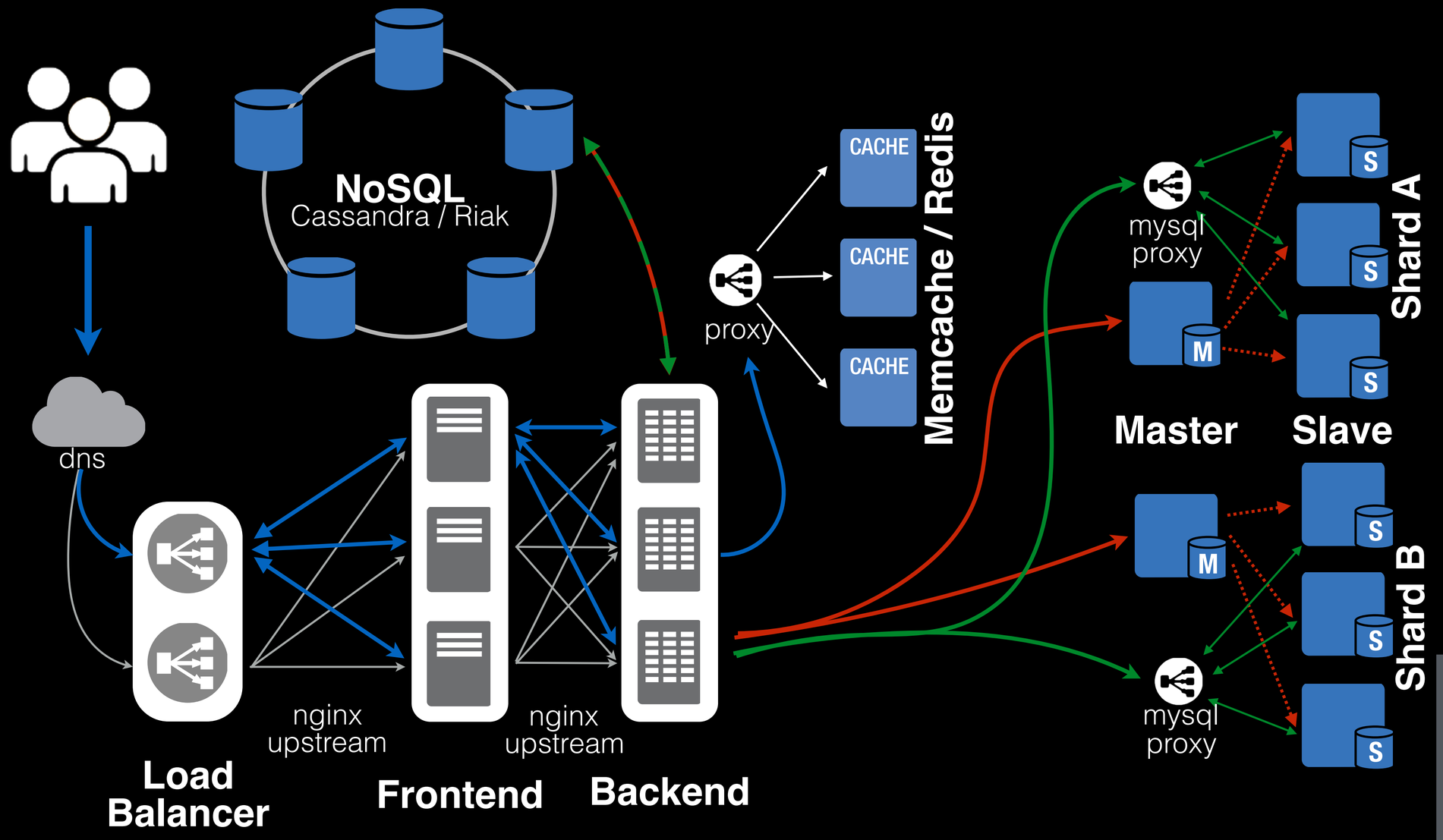
Гуглить:
cassandra, google bigtable, hbase, amazon dynamo, riak

НГС

NoSQL

- Много хороших решений
- Надо уметь готовить
- Другая схема данных
- Другой язык запросов
- Другие приоритеты в CAP-теореме

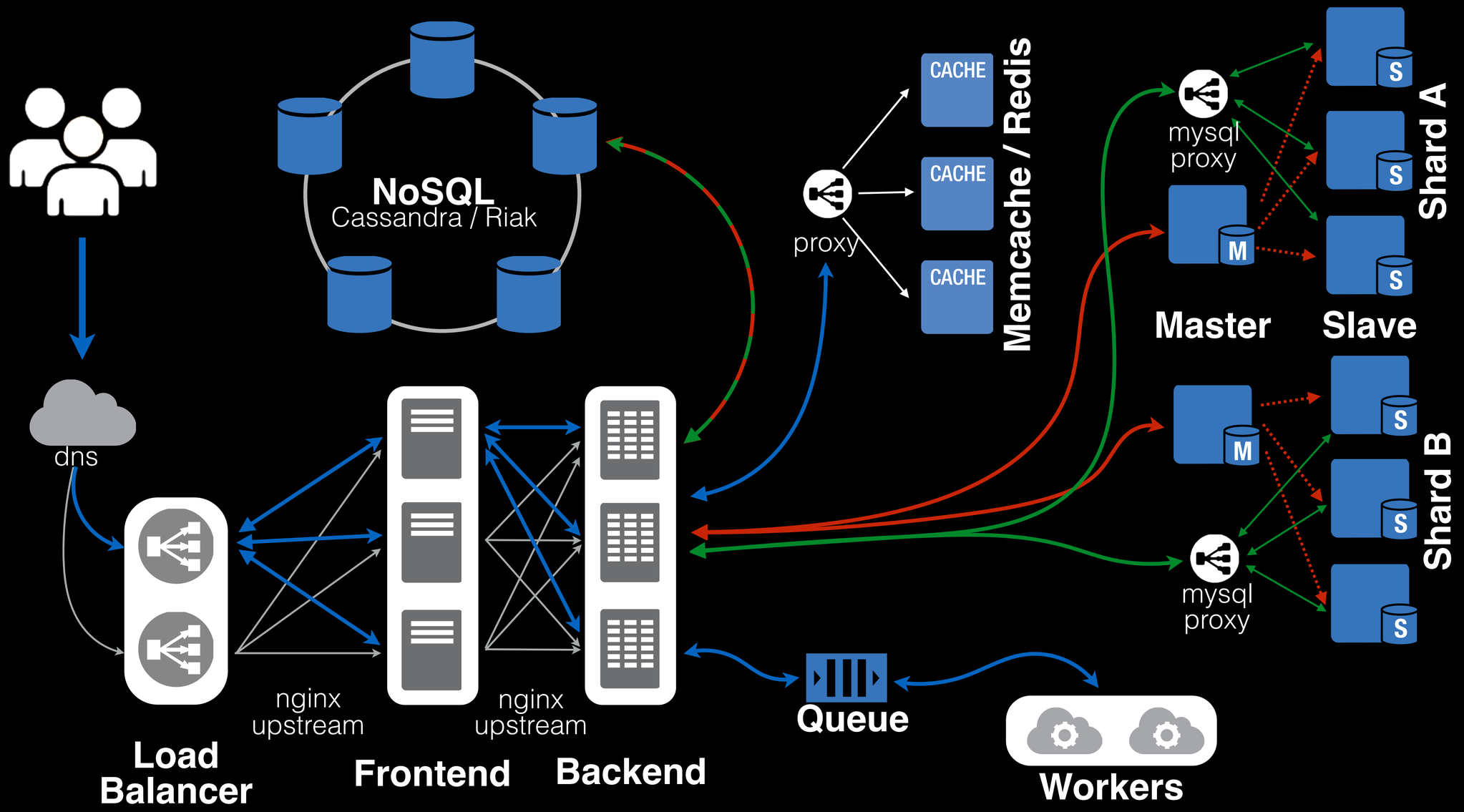
Теперь с NoSQL



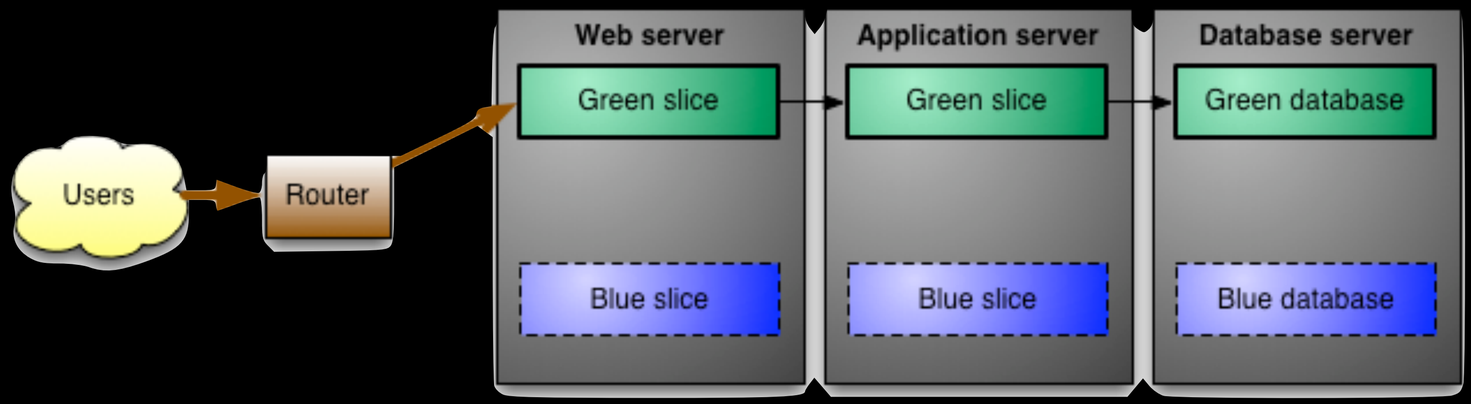
Масштабирование записи

- Мастер-мастер даст отказоустойчивость
- Шардинг даст пропорциональный прирост записи, но есть нюансы
- Не стоит забывать про “NoSQL” решения, есть очень хорошие базы данных
- Иногда полезно совмещать RDBMS и NoSQL решения

Итоговая архитектура



Релиз новой версии кода



Гуглить:
blue-green deployment, continuous delivery, continuous deployment

Мониторинг

- Графики должны быть на все изменяющиеся значения
- Даже если значение не изменяется, на него должен быть график (вдруг изменится)
- Как минимум следить за превышением пороговых значений
- В идеале отслеживание трендов и “умное” детектирование аномалий

Гуглить:
zabbix, graphite, pinba, statsd, etsy skyline, opentsdb, influxdb

Общие нюансы

- Всегда помните о проблеме больших чисел
- Помните о блокировках в разных частях системы
- Отказоустойчивость очень важна
- Загружать сервера на 100% опасно
- SOA и микросервисы

Гуглить:
fault-tolerance, failover, soa, microservice arcitecture

Секреты высоконагруженных систем

- Отделяем мух от котлет для полноценной утилизации ресурсов и независимой масштабируемости
- Все сервисы масштабируются по-разному, но подходы похожие
- Но помни про нюансы (лаги, консистентность и т.д.)
- Быстрый и автоматический failover залог здорового сна
- Использовать php+mysql для всех задач можно, но есть специализированные сервисы для многих задач

Вопросы?

@olegfedoseev
o.fedoseev@office.ngs.ru
vk.com/devngs