Tarantool, Sharding, Nginx: скорость, простота, масштабируемость

Сошников Василий
Дроздов Андрей

Tarantool

- NoSQL база данных
- Lua application server
- Клиент-сервер протокол базируется на MessagePack
- Два движка для хранения данных
  - 100% in-memory с опциональной персистентностью
  - дисковый 2 уровненный B-tree
- Асинхронная репликация master-master
- Контроль доступа

Tarantool Nginx module

- Объединяет возможности Nginx и Tarantool

Tarantool Nginx module

- Объединяет возможности Nginx и Tarantool
- Возможность вызывать хранимые процедуры с помощью HTTP POST

Tarantool Nginx module

- Объединяет возможности Nginx и Tarantool
- Возможность вызывать хранимые процедуры с помощью HTTP POST
- Гибкое конфигурирование

Tarantool Nginx module

- Объединяет возможности Nginx и Tarantool
- Возможность вызывать хранимые процедуры с помощью HTTP POST
- Гибкое конфигурирование
- Минимум оверхэдов

Tarantool Nginx module

- Объединяет возможности Nginx и Tarantool
- Возможность вызывать хранимые процедуры с помощью HTTP POST
- Гибкое конфигурирование
- Минимум оверхэдов
- Fault tolerance

Tarantool Nginx module

```
upstream tarantool_backend {
   server tarantool_host    max_fails=1 fail_timeout=30s;
   server tarantool_host.1 max_fails=1 fail_timeout=30s;

   server tarantool_host.backup backup;
 }

 server {
     location = /tarantool {
       tnt_pass tarantool_backend;
     }
 }
```

Tarantool Nginx module

- Хранимая процедура на tarantool_host*

```
function echo(a)
  return {a, “world!”}
end
```

Tarantool Nginx module

- Хранимая процедура на tarantool_host*

```
function echo(a)
  return {a, “world!”}
end
```

- Вызываем метод ‘echo’

Tarantool Nginx module

- Хранимая процедура на tarantool_host*

```
function echo(a)
  return {a, “world!”}
end
```

- Вызываем метод ‘echo’

—> { “method”: “echo”, “params”: [“hello”], “id”: 0 }

Tarantool Nginx module

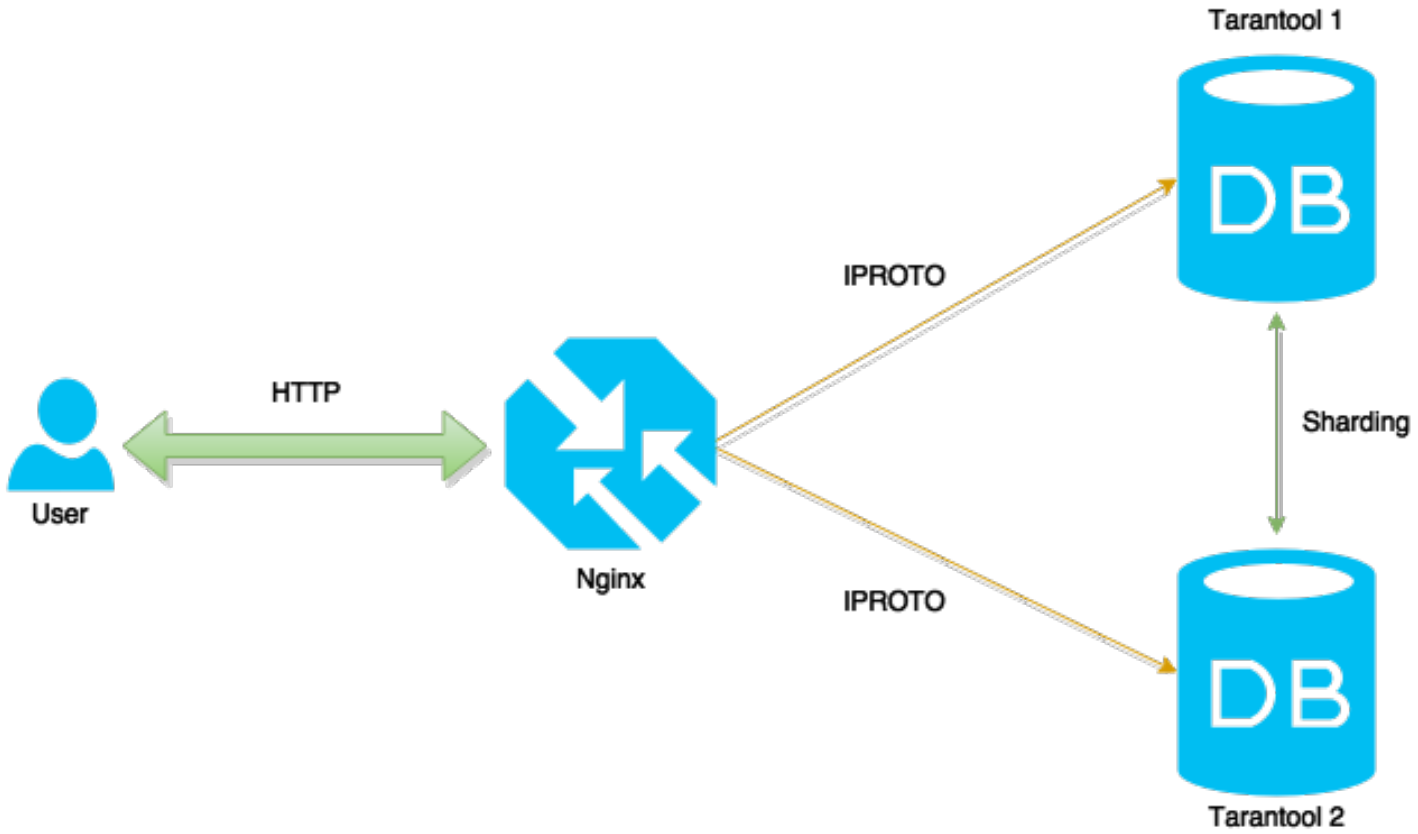
- Хранимая процедура на tarantool_host*

```
function echo(a)
  return {a, “world!”}
end
```

- Вызываем метод ‘echo’

—> { “method”: “echo”, “params”: [“hello”], “id”: 0 }
<— { “result”: [[“hello”, “world!”]], “id”: 0 }

Tarantool Nginx module



Практическое применение

- Какую задачу решить!

Практическое применение

- Какую задачу решить!
- Супер идея: загрузим всю Википедию в Tarantool!

Практическое применение

- Какую задачу решить!
- Супер идея: загрузим всю Википедию в Tarantool!
- Реальность: Нет столько машин :(

Практическое применение

- Какую задачу решить!
- Супер идея: загрузим всю Википедию в Tarantool!
- Реальность: Нет столько машин :(
- Откорректированная ‘Супер идея’:
  загрузим только граф категорий

От слов к делу

- Категория Википедии это
  - древовидная связь терминов/статей/...

От слов к делу

- Категория Википедии это
  - древовидная связь терминов/статей/…
  - типичный случай: начали читать о ‘ложке’, закончили ‘египетским фараоном’

От слов к делу

- Категория Википедии это
  - древовидная связь терминов/статей/…
  - типичный случай: начали читать о ‘ложке’, закончили ‘египетским фараоном’
  - термины/статьи/… и связи хранятся в mysql, модель хранения ‘один к многим’

SQL dump в Tarantool

- Преобразуем из SQL

ID | TITLE | CATEGORY
1 | Ложка | Ложки
1 | Ложка | Столовые приборы
...
2 | Вилка | Столовые приборы

SQL dump в Tarantool

- В json, ID | TITLE | CATEGORY

```
{
“method”: “wiki.load”,
“params”: [
    1, …,
   “Ложка”,
   [ “Ложки”,” Столовые приборы”,… ] ], …
}
```

SQL dump в Tarantool

- Загружаем в Tarantool
  стандартным HTTP конектором

SQL dump в Tarantool

- Загружаем в Tarantool
  стандартным HTTP конектором
- Вызываем wiki.compute

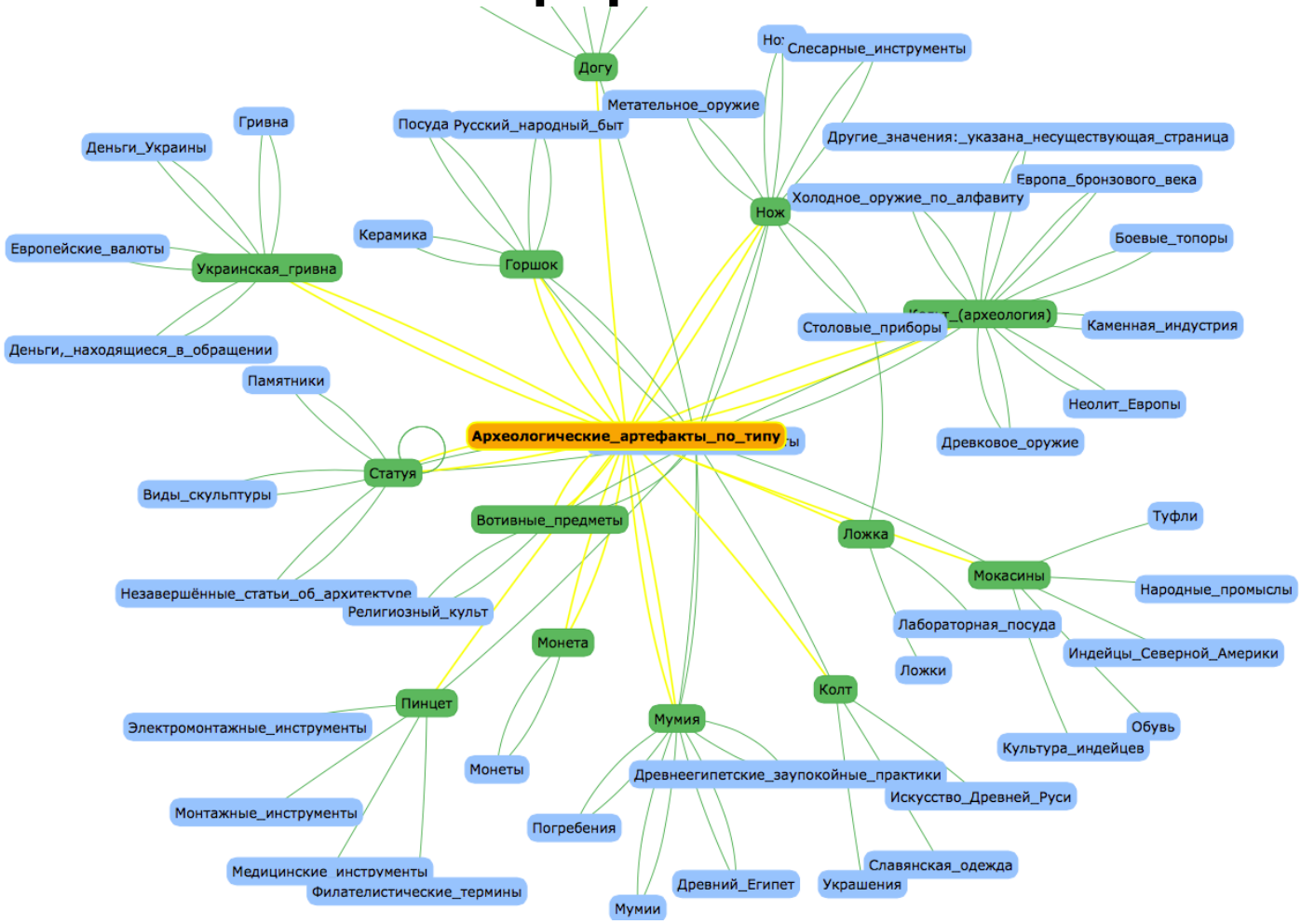
SQL dump в Tarantool

- Загружаем в Tarantool
  стандартным HTTP конектором
- Вызываем wiki.compute
- Делаем простой GUI на javascript/jquery/…
  для поиска используем jquery.post:
  $.post(‘/tarantool', { “method”: “wiki.lookup”,… })

SQL dump в Tarantool

- Загружаем в Tarantool
  стандартным HTTP конектором
- Вызываем wiki.compute
- Делаем простой GUI на javascript/jquery/…
  для поиска используем jquery.post:
  $.post('/tarantool', { "method": "wiki.lookup",… })
- Результат: http://wiki.build.tarantool.org

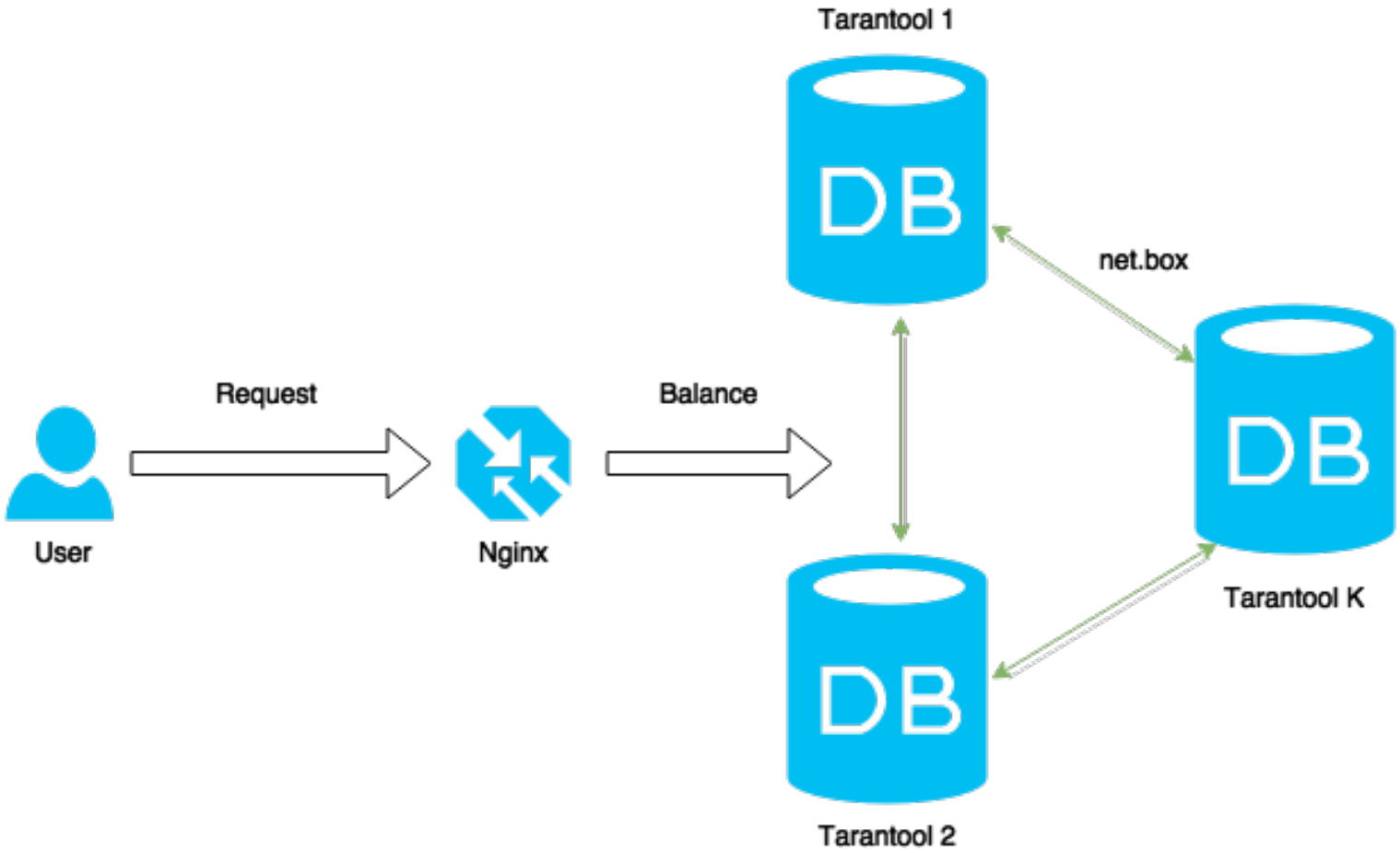
Так как связана ‘Ложка’ и ‘Египетский фараон’?



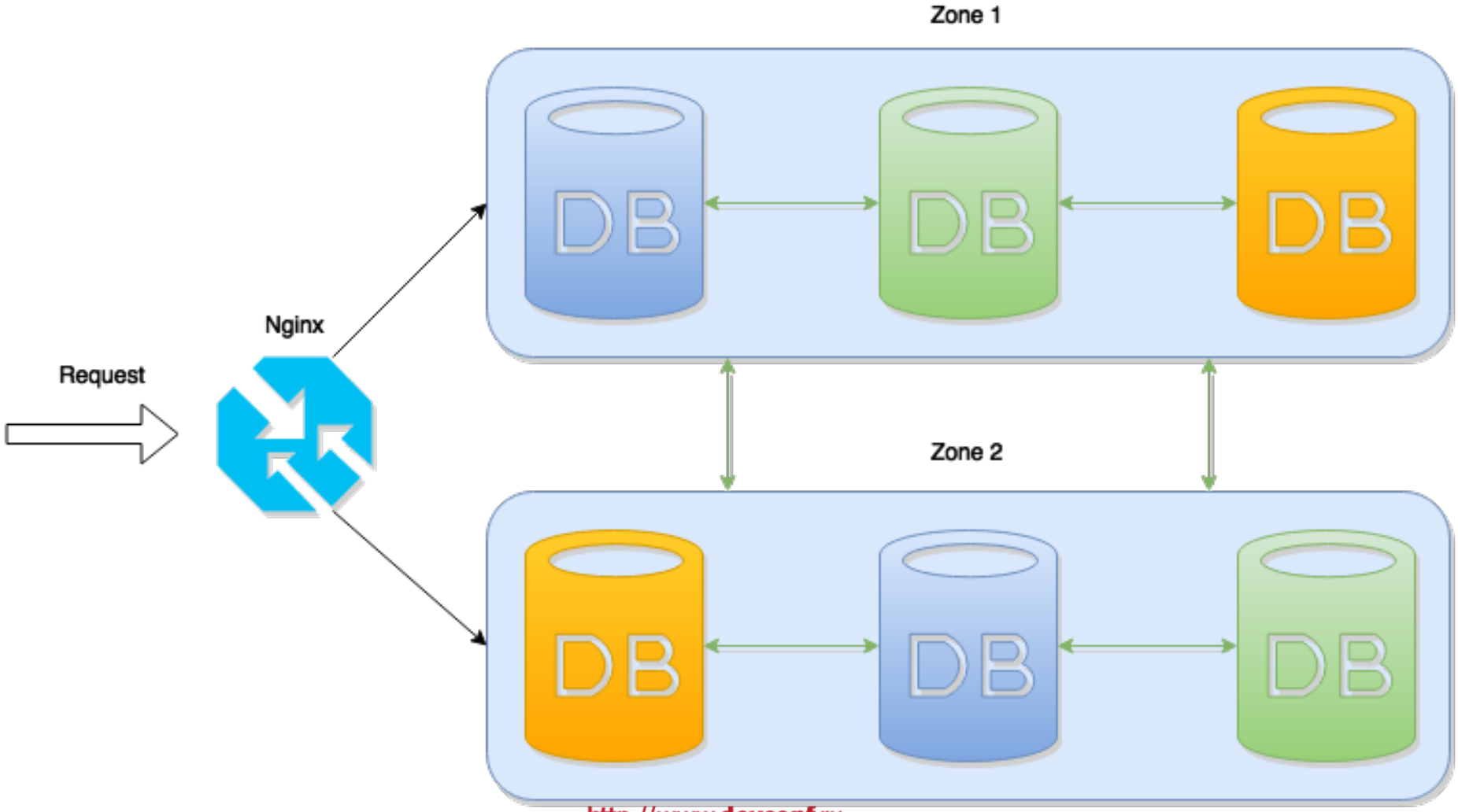
Подведем итоги

- Решить такую задачу просто
- Легко масштабируется
- Высокая скорость работы
- Fault tolerance

Как устроен шардинг?



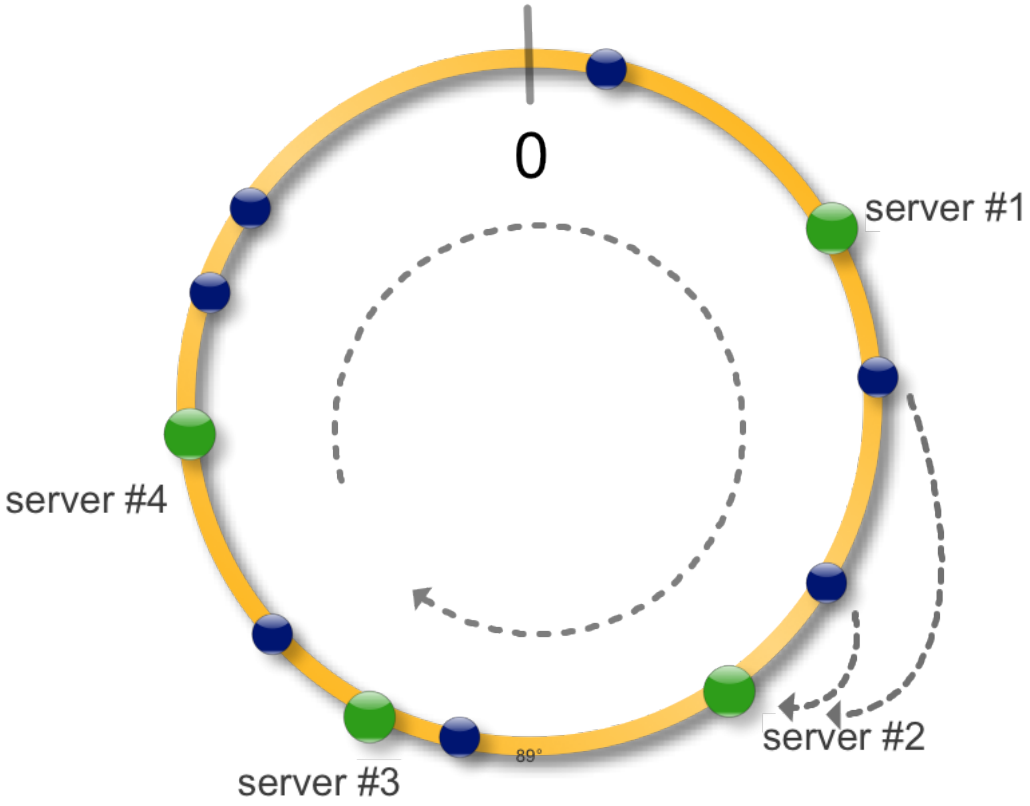
Архитектура: зоны и избыточность



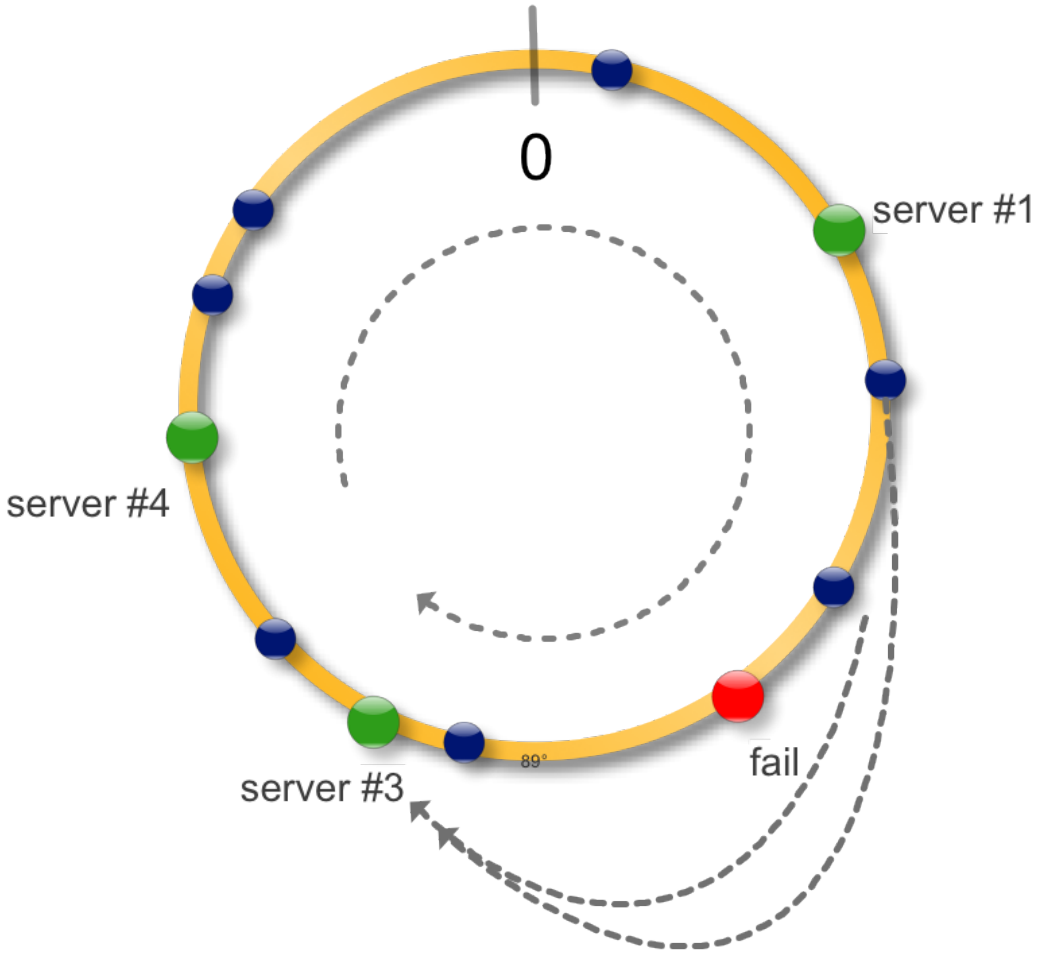
Пример конфигурации

```
local cfg = {
    servers = {  -- сервера для шардинга
        { uri = 'localhost:33130', zone = 'z1' };
        { uri = 'localhost:33131', zone = 'z2' };
    };
    login = 'my_user';
    password = 'i_love_bananas';
    redundancy = 2; -- избыточность
    binary = 33130; -- порт
}
```

Шард-функция

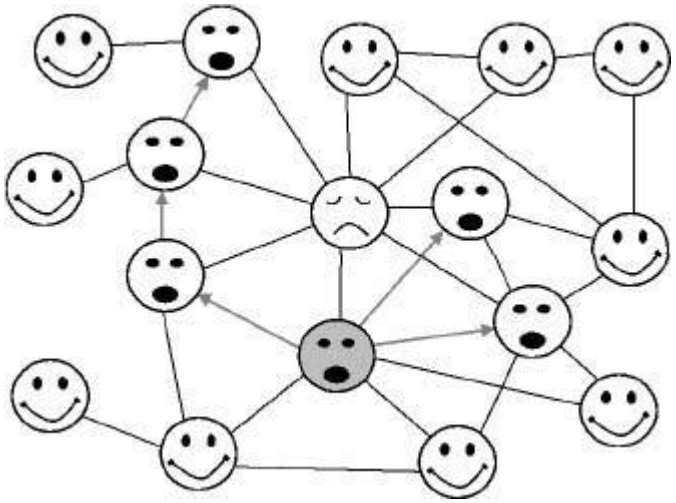


Шард-функция



Мониторинг

- Connection pool
- Gossip-like мониторинг
- Таблица шардинга



Мониторинг: инициализация

```
Node2:
  Node2: {'try': 0, 'ts': -1}
  Node1: {'try': 0, 'ts': -1}
Node1:
  Node2: {'try': 0, 'ts': -1}
  Node1: {'try': 0, 'ts': 1434368336}
...
```

Мониторинг: штатная работа

Node2:
  Node2: {'try': 0, 'ts': 1434368337}
  Node1: {'try': 0, 'ts': 1434368337}
Node1:
  Node2: {'try': 0, 'ts': 1434368338}
  Node1: {'try': 0, 'ts': 1434368339}
...

Мониторинг: ошибка в сети

Node2 connection failed

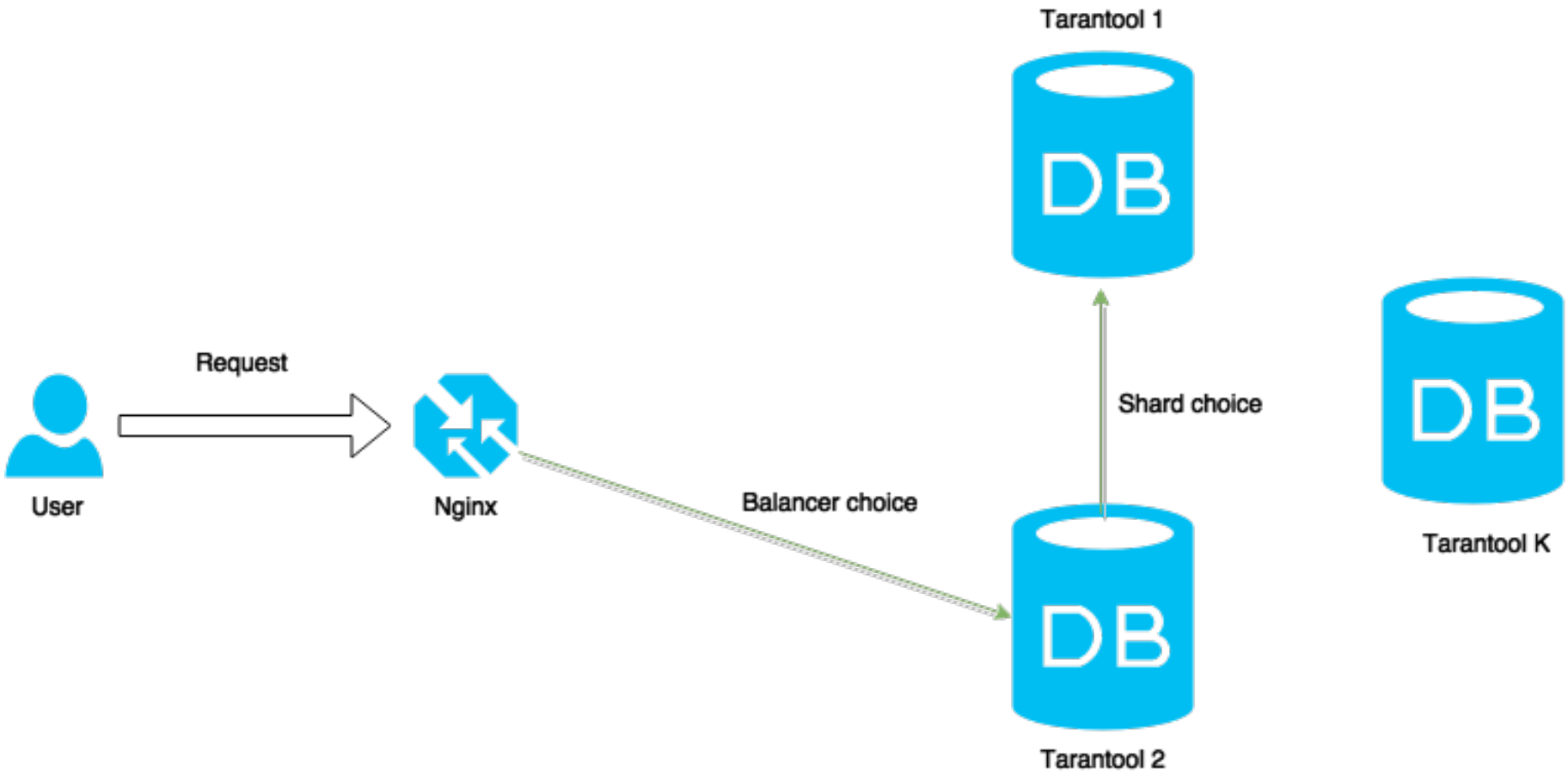
Node2:
  Node2: {'try': 0, 'ts': 1434368354}
  Node1: {'try': 0, 'ts': 1434368353}
Node1:
  Node2: {'try': 6, 'ts': 1434368358}
  Node1: {'try': 0, 'ts': 1434368357}
...

Мониторинг: исключение

kill Node2 by dead timeout
zone 1 has no active connections

Node1:
  Node2: {'try': 10, 'ts': 1434368360}
  Node1: {'try': 0, 'ts': 1434368360}
...

Однофазные операции

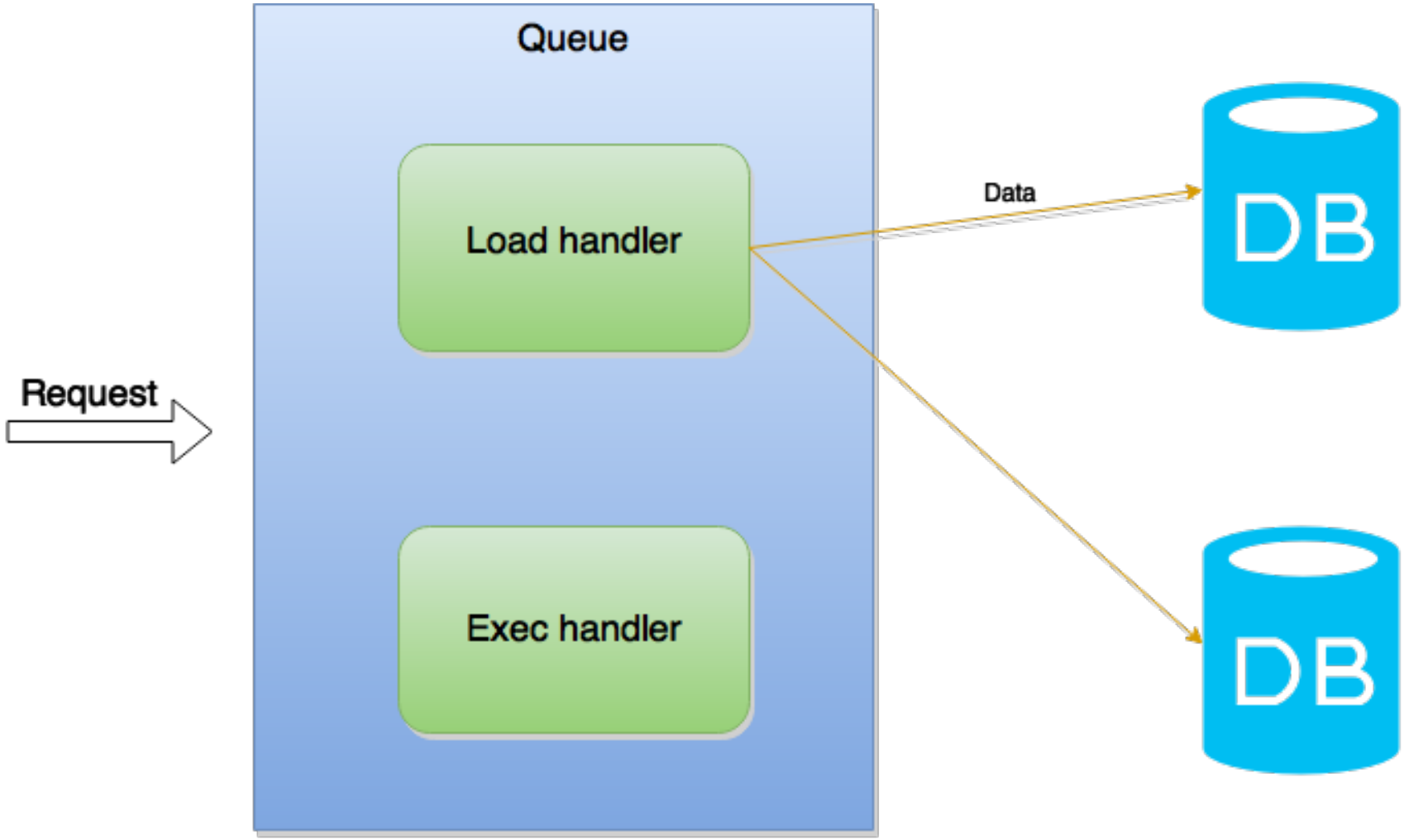


Однофазные операции

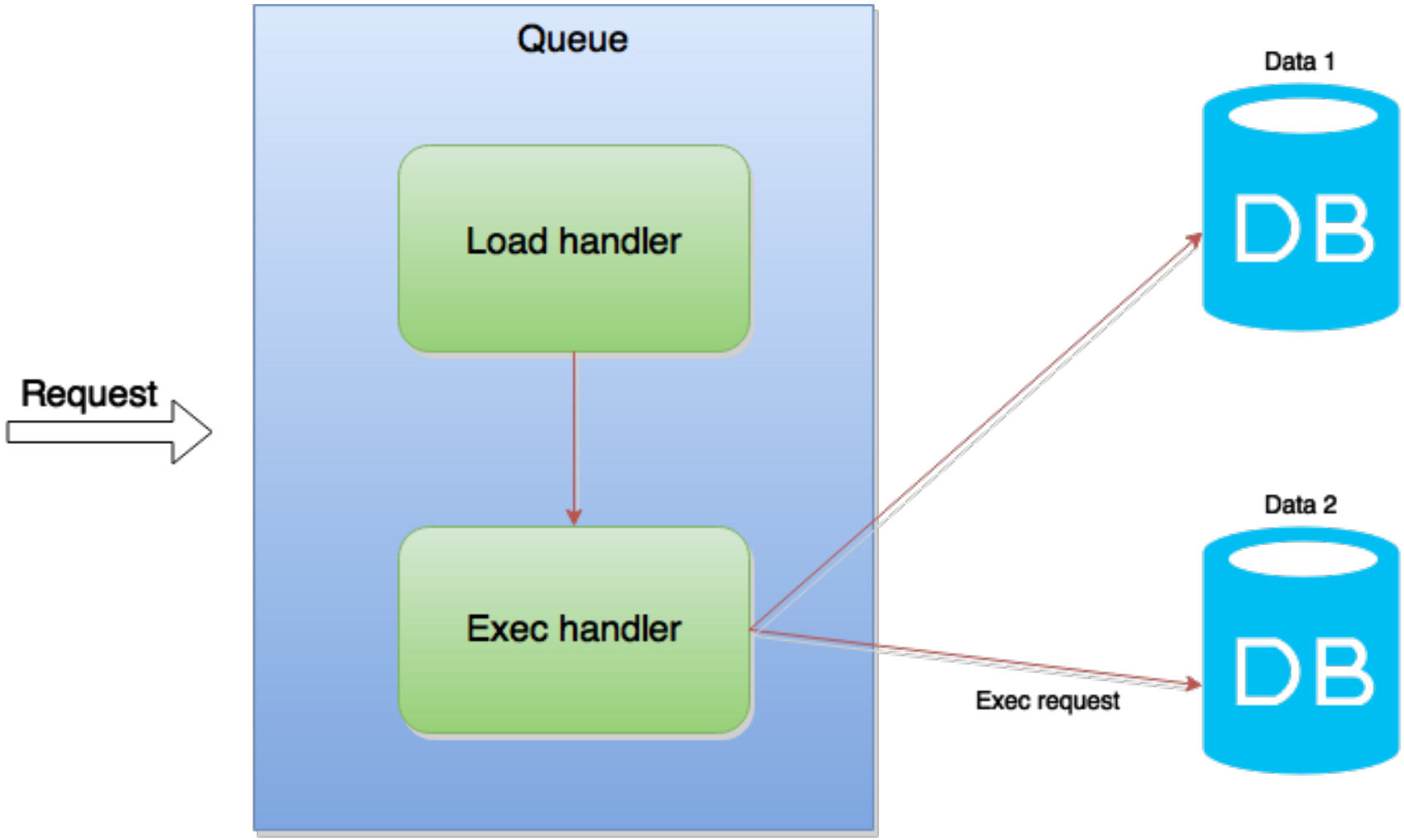
```
shard.demo:insert{1, 'test'}
shard.demo:replace{1, 'test2'}
shard.demo:update(1, {{'=', 2, 'test3'}})
shard.demo:insert{2, 'test4'}
shard.demo:insert{3, 'test5'}
shard.demo:delete(3)
```

Как быть?

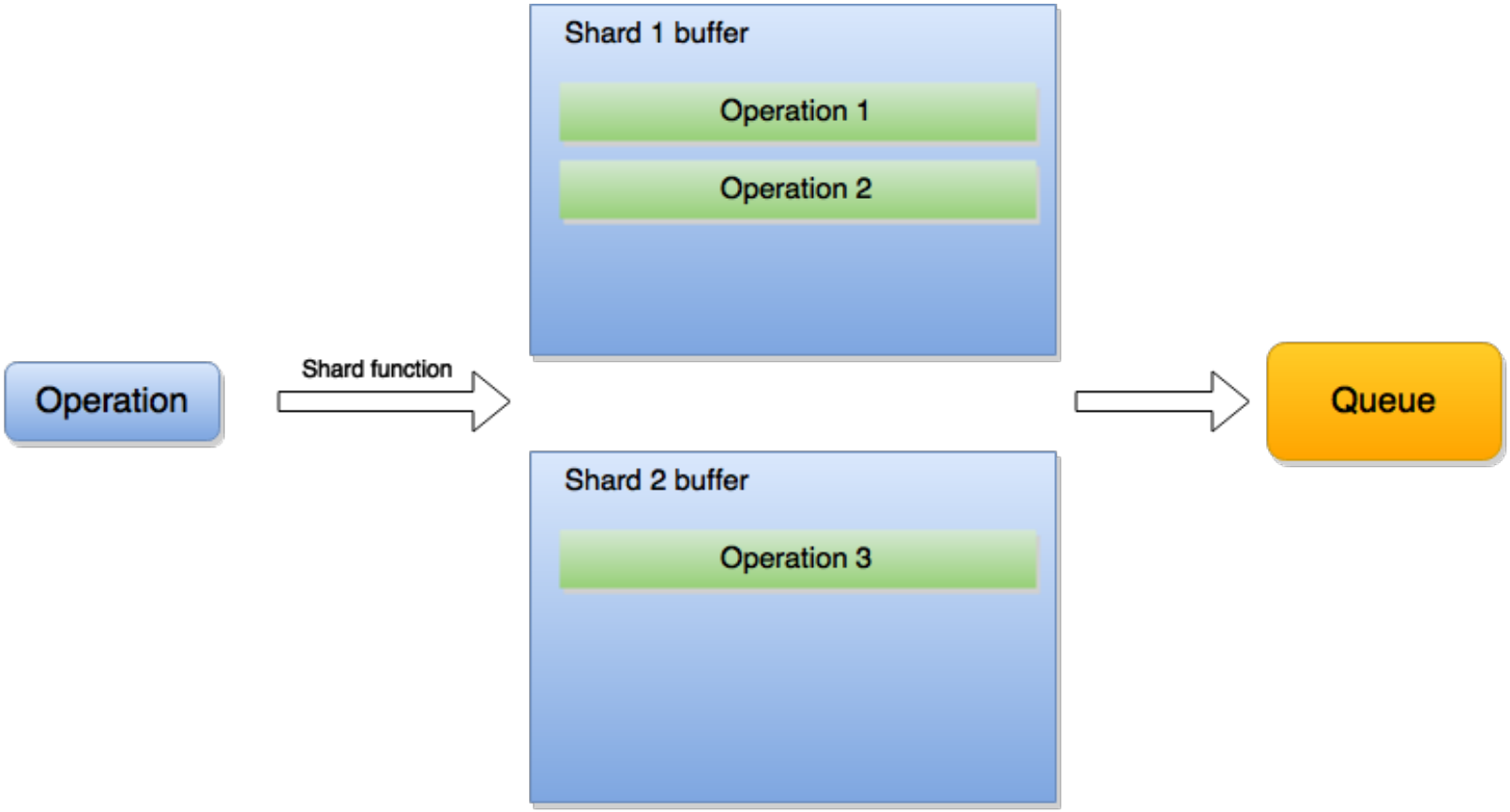
Двухфазный протокол. Рассылка



Двухфазный протокол. Исполнение



Батчинг


Shard 1 buffer
Operation 1
Operation 2
Operation
Shard function
Queue
Shard 2 buffer
Operation 3

Двухфазный протокол

```
batch = shard.q_begin()
batch.demo:q_insert(1, {0, 'test'})
batch.demo:q_replace(2, {0, 'test2'})
batch.demo:q_update(3, 0, {{'=', 2, 'test3'}})
batch.demo:q_insert(4, {1, 'test4'})
batch.demo:q_insert(5, {2, 'test_to_delete'})
batch.demo:q_delete(6, 2)
batch:q_end()
```

Спасибо за внимание!
Вопросы?

- Официальный сайт проекта
  http://tarantool.org
- Shard
  https://github.com/tarantool/shard
- Connection pool
  https://github.com/tarantool/connection-pool
- Nginx module
  https://github.com/tarantool/nginx_upstream_module
- Исходники Wiki demo
  https://github.com/Sulverus/tarantool-wiki-lookup
- Wiki demo
  http://wiki.build.tarantool.org/