Octopus: фабрика NoSQL

Вострико́в Юрий

March 22, 2016

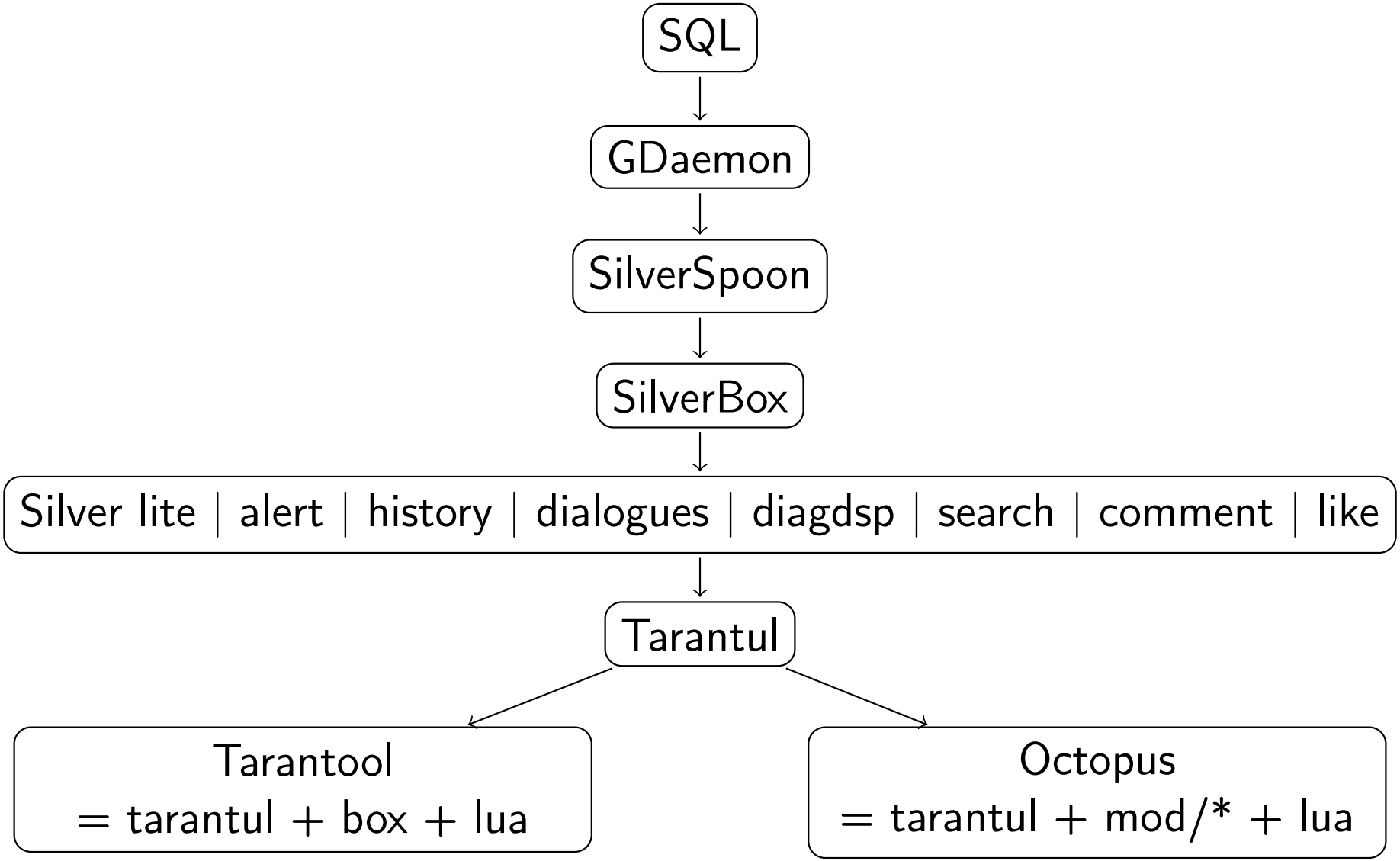
О чем этот доклад

- Octopus — фреймворк для NoSQL
- История появления NoSQL в МоемМире
- Устройство Octopus
- примеры использования в МоемМире
- пример написания игрушечного модуля

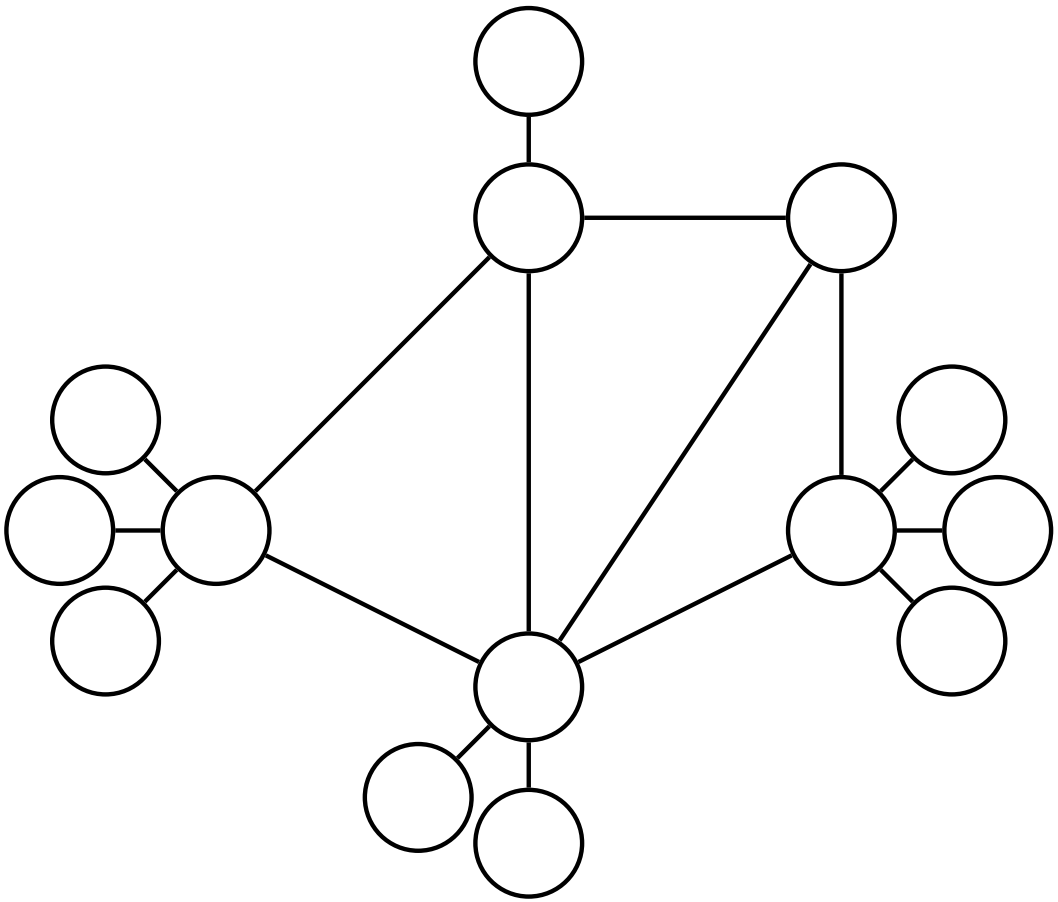
Производительность vs надежность + удобство

- Удобство / выразительность
- ACID
  - Atomicity relax
  - Isolation levels
  - Durability adjustment
- Производительность
  - задержка (latency)
  - пропускная способность (throughput)

История развития NoSQL в МоемМире



Граф дружб



Характеристики:

Кол-во связей	10^9
сейчас	$2.3 \cdot 10^9$
Размер ребра	6 байт
Среднее кол-во	20

Жизнь с SQL

- overhead: на 8 байт полезных данных приходится 50 служебных.

  Расходы на хранение (int4, int4):

PostgreSQL	36 байт (58 байт вместе с индексом)
MySQL (innodb)	46 байт

- страничный (!) кэш

	overhead
одна строка горячая	1000-кратный
все строчки горячие	8-кратный

- Надежность $\Longrightarrow$ 100%

  Ее ~~нельзя~~ сложно разменять на производительность.

GDaemon

Причины написания:

- возросла нагрузка
- появились «неудобные» запросы
  - Friends of Friends
  - FriendPath
- граф сложно пошардить

Устройство:

- самодельный демон на C
- Загрузка с SQL бакапа
- паралельная запись из perl кода: и в gdaemon и в SQL

Проблемы:
Тормозит и падает, долго загружается. Данные в GDaemon и в SQL со временем начинают расходится. Парсить текстовые дампы довольно дорого, когда их гигабайты.

Дружелюбный демон: SilverSpoon

- 2008 год
- сильно патченный memcached
  Унаследовано:
  - сетевая подсистема (FSM)
  - *a la* slab аллокатор
- Snapshot + WAL
- бинарный протокол
- почти MVC
- стукнулись об $O(n^2)$
  bmtree (Федя Сигаев):

$count(elem) < 8096$	отсортированный массив
иначе	страничное b+ дерево

Оказался весьма успешным. 100k+ RPS с одной машины.

SELECT UID FROM USERDISPATCH WHERE EMAIL

uid	email	shard	
10	dima@mail.ru	10.0.0.1	...
11	vova@mail.ru	10.0.1.2	...
12	lena@list.ru	10.0.0.1	...

Надо *два* индекса: по uid и по email. Если хранить два соответсвтия uid $\rightarrow$ email и email $\rightarrow$ uid, то удваивается расход памяти и нужны транзакции. У Redis нет было WAL.

SilverBox

- 2009
- патченный SilverSpoon
- кортежи и несколько индексов
- нет схемы (хорошо), нет типов (плохо)

Silver + *

spoon	граф связей
lite	облегченный граф связей
box	кортежи
alert	*n* последних нотификаций с вытеснением
history	лента активности
dialogues	сообщения (диалоги)
diagdsp	диспетчер для диалогов
search	перекрестный поиск
comment	комментарии к записям
like	«мне нравится»
queued	очереди с таймаутами

У alert, history, dialogues, comment, like есть вытеснение на диск.

ifdef hell

```
#ifdef NEED_RETURN_CODE
  add_iov(&cc->ret_code, sizeof(cc->ret_code));
#endif
#ifdef SILVERSPOON_SS
  smart_suggests_dispach(cc->msg, (uint32_t *)(cc->rbu
#elif SILVERBOX
  cc->ret_code = box_dispach(&(cc->txn), cc->msg, cc->
#elif SILVERALERT
  alert_dispatch(cc->msg, (char *)cc->rbuf + 12, cc->
#elif SILVERSEARCH
  search_dispatch(&cc->txn, cc->mc, SearchStorage,
                  cc->rbuf + 12 + QUERYOFFSET,
                  cc->req_len - QUERYOFFSET - 1 /* tr
#elif SILVERHISTORY
  history_dispatch_request(&(cc->txn), cc->msg, (char
#elif SILVERDIAGDSP
  diagdsp_dispatch(&(cc->txn), cc->msg, (char *)cc->rb
#else
   friends_graph_dispach(cc->msg, (uint32_t *)(cc->rbu
#endif
```

cb hell

```
void cb(int fd) {
        struct cb2_arg *arg = malloc();
        *cb2_arg = (struct cb2_arg){ .buf = buf };
        read_from_net(fd, (void *)cb2, cb2_arg);
}
void cb2(struct cb2_arg *arg) {
        parse_end_exec(arg->buf, &wal, &resp);
        free(arg);
        struct cb3_arg *arg = malloc();
        *cb3_arg = (struct cb3_arg){ .resp = resp };
        write_to_disk(x, wal, (void *)cb3, cb3_arg);
}
void cb3(struct cb3_arg *arg){
        write_to_net(x, arg->resp, (void *)cb4, arg);
}
void cb4(struct cb4_arg *arg){
        free(arg->...);
        free(arg);
}
```

Octopus

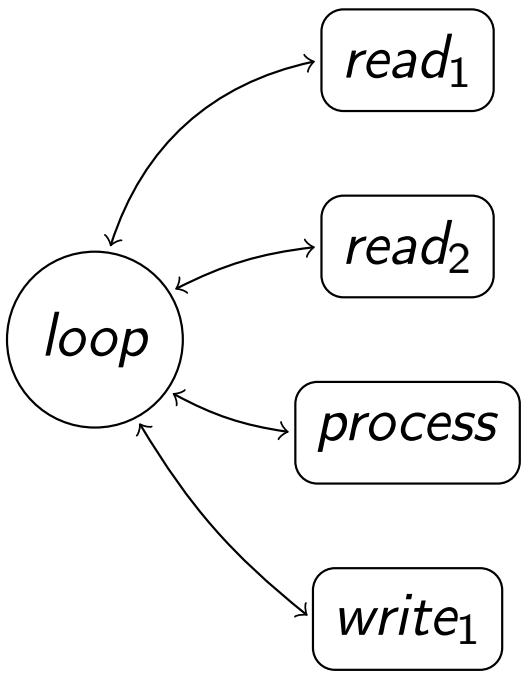
в девичестве Tarantul

- кооперативная многозадачность: fiber = libcoro + libev
- специализированные аллокаторы: slab + pool
- Snapshot + WAL
- репликация
- индексы
- экспорт статистики в Graphite
- μsharding (β)
- PAXOS (β)
- *низкоуровневые* модули
  - специализированные форматы данных
  - специализированные индексы
  - нестандартные запросы
  - «хитрые» кэши

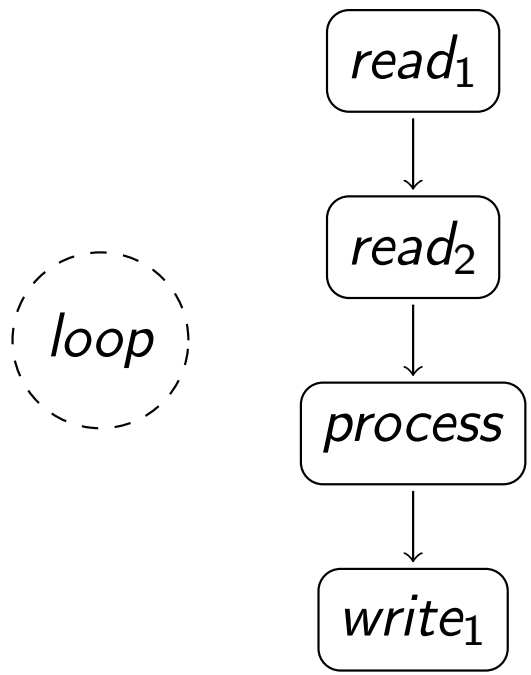
Массаракш: код наизнанку

Fibers: concurrency, not parallelism.

FSM, libev only



Fiber, libev+libcoro



cb hell

```
void cb(int fd) {
        struct cb2_arg *arg = malloc();
        *cb2_arg = (struct cb2_arg){ .buf = buf };
        read_from_net(fd, (void *)cb2, cb2_arg);
}
void cb2(struct cb2_arg *arg) {
        parse_end_exec(arg->buf, &wal, &resp);
        free(arg);
        struct cb3_arg *arg = malloc();
        *cb3_arg = (struct cb3_arg){ .resp = resp };
        write_to_disk(x, wal, (void *)cb3, cb3_arg);
}
void cb3(struct cb3_arg *arg){
        write_to_net(x, arg->resp, (void *)cb4, arg);
}
void cb4(struct cb4_arg *arg){
        free(arg->...);
        free(arg);
}
```

fiber heaven

```
char *buf, *wal, *resp;
read_from_net(fd, buf);
parse_end_exec(buf, &wal, &resp);
write_to_disk(wal);
write_to_net(x, resp);
free(buf, wal, resp);
```

The malloc(3) Is Not Enough

Два типа аллокаций: объекты и временные данные. Объекты живут долго их очень много. Временные данные живут мало их не много.

- slab alloc
  - для объектов (кортежей)
  - предсказуемая задержка
  - контролируемая фрагментация
  - быстрее чем tcmalloc на 20-30%
- palloc
  - для временных данных
  - очень дешев: в удачном случае всего 14 инструкций
  - удалить можно только все объекы сразу

Помимо МоегоМира используются в Почте.
https://gitlab.corp.mail.ru/octopus/util

Индексы

MHash	инкрементальный рехешинг, указатели 48 бит
SPTree	указатели 16-48 бит
TWLTree	B+ дерево
NIHTree	B+ дерево с рейтингом
BMTree	только в SilverSpoon/SilverLite

Расходы памяти на один элемент индекса:

glibc tree	48
MHash	17 / 29
SPTree	23.2
NIHTree	8.6
TWLTree	7.4

Все деревья есть в https://gitlab.corp.mail.ru/octopus/util

Lua: moonbox

Расширить протокол: новые структуры данных, нетривиальные трансформации. Пусть Lua процедуры пишут perl программисты. Дадим им доступ к FFI для повышенной гибкости.

Проблемы: слабопредсказуемая производительность, LuaJIT надоел автору.

В trace компиляторе много эвристик, которые зависят от данных. На тестах все компилируется, в продакшене все интерпретируется.

Octopus это модули

box	SilverBox
carbide	агрегатор/медиатор для Graphite
diagdsp	диспетчер для диалогов
diagdsp-counters	«хитрый» фидер
feeder	фидер
goldenspoon	SilverSpoon Mk.II
hash_tag	#теги
iproxy	IProto proxy & tee
kava	SilverBox с *вытеснением на диск*
like_comment	лайки комментариев
memcached	memcached + snapshot + WAL
rlimit	централизованные рейтлимиты
silversearch	перекрестный поиск
colander	IProto агрегатор
example	игрушечный пример

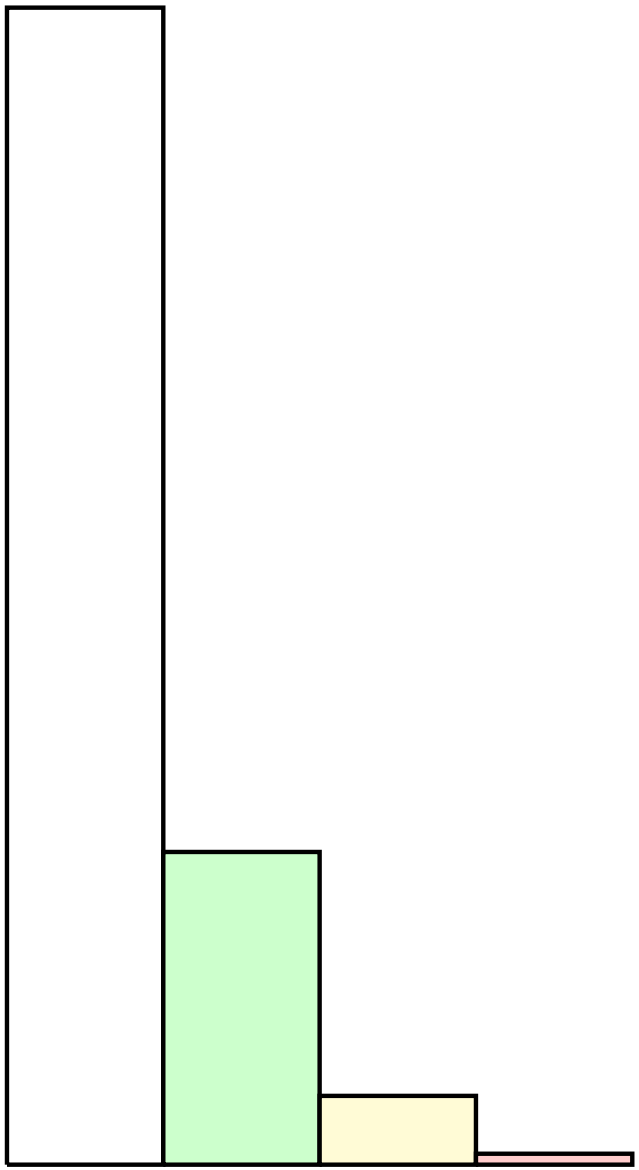
SilverDialogues: диспетчер

Пользователь $\rightarrow$ список диалогов $\rightarrow$ список сообщений.

У большинства пользователей немного диалогов. У общительных пользователей (спамеров) — много. При показе списка диалогов надо показать кол-во (новых) сообщений.

- Разный формат хранения: для маленьких — плотноупакованный, для больших — list of hashes.
- $N \times M$ «репликация»

SilverDialogues: сторадж



Вся база	74ГБ
Теплые диалоги	20ГБ
Горячие диалоги	4.4ГБ
Горячие сообщения	0.7ГБ

горячего — 1%

теплые — показ только последнего сообщения

Централизованные рейтлимиты: RLimit

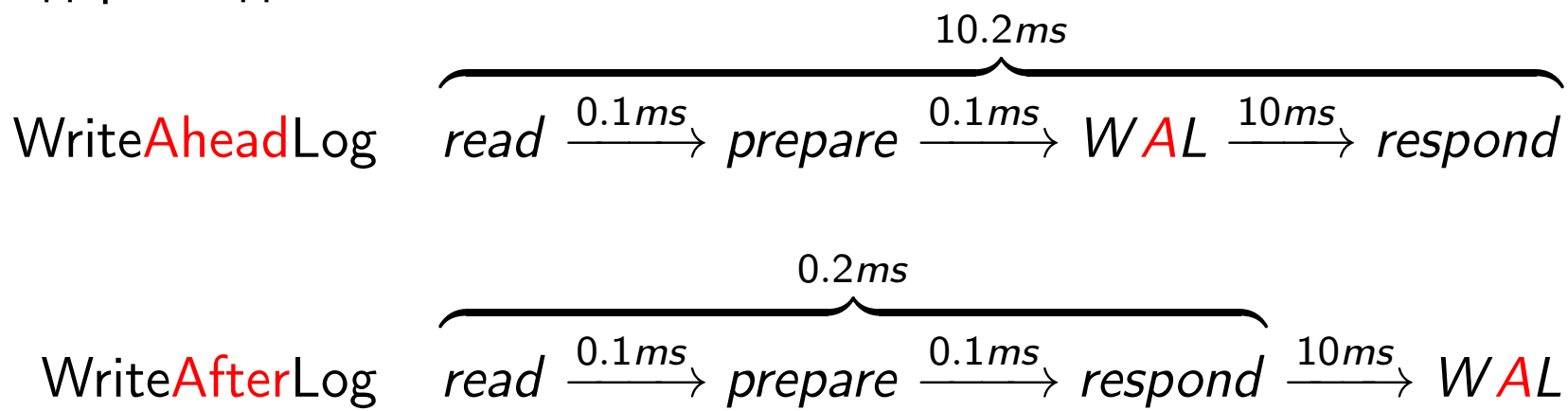
Централизованная база с рейтлимитами и толпа фронт-ngninx которые к ней ходят на каждый хит и проверяют лимит. При каждом обращении надо пересчитывать счетчики. Потеря расчетов не опасна — не приводит к отказу обслуживания т.к. рейтлимиты лишь ослабляются.

Модуль для nginx + модуль для octopus

Производительность любой ценой

Иногда нужно разменять задержку на надежность. Фильтрация плохих не должна наказывать хороших.

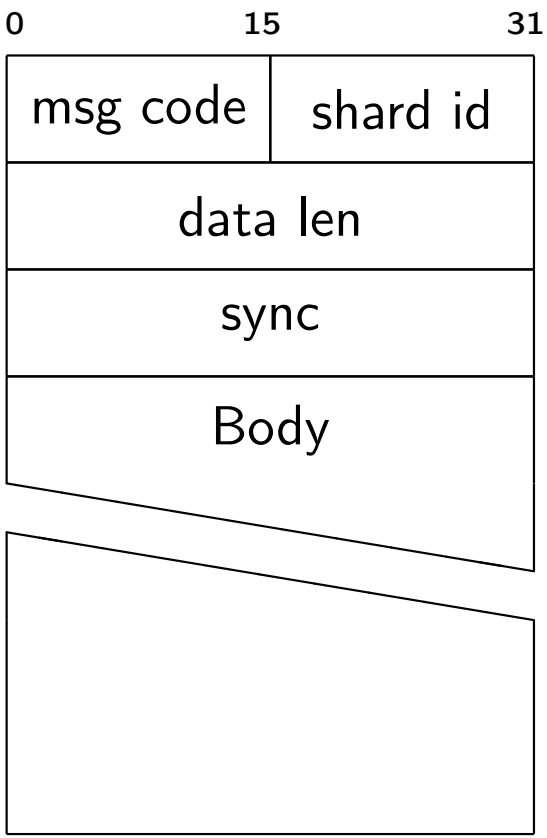
Задержка для клиента:



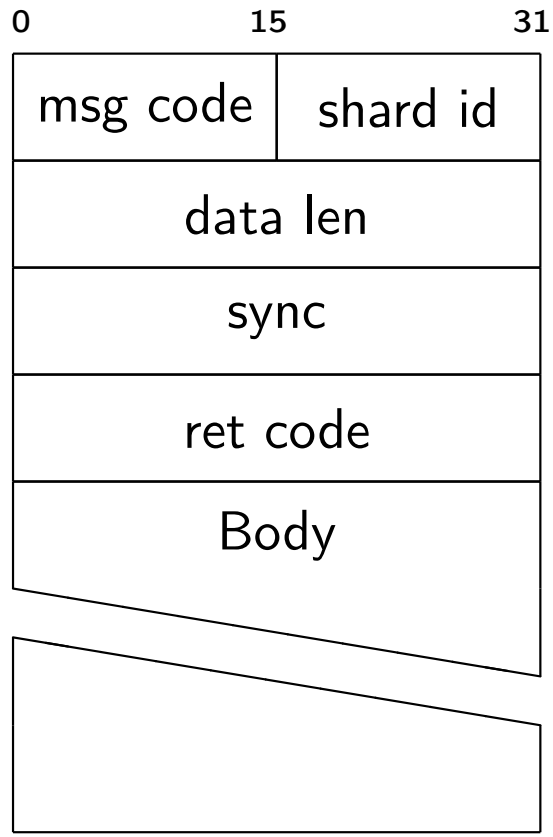
Клиентские библиотеки

- client/libiproto/libiproto.h,iproto.c
- mod/box/client/ruby/iproto.rb
- библиотеки от tarantool 1.5
- *очень* простой протокол

Запрос



Ответ



Пример модуля: u32 → u32 map

Цель: хранение маппинга из u32 в u32.

Поддерживаемые запросы:

- UPSERT
- DELETE
- QUERY

около 130 строк кода включая репликацию.

обработка UPSERT и DELETE

```
enum messages { UPSERT = 31, DELETE = 42, ... };
void modify(struct *wbuf, struct iproto *req,
            int msg_code, int len, void *data)
{
  if (len != 8)
    return iproto_error(wbuf, req, 0x102, "bad req");

  /* WAL write */
  if ([writer submit:data len:8 tag:msg_code] != 1)
    return iproto_error(wbuf, req, 0x201, "io err");

  /* modify */
  if (msg_code == UPSERT)
    twltree_insert(self->index, data, true);
  else
    twltree_delete(self->index, data);

  iproto_reply_small(wbuf, req, 0 /* OK */);
}
```

Сохранение и восстановление

```
void snapshot_write_rows(XLog *fd)
{
  struct pair *pair;
  twliterator_t it;
  twltree_iter_init(self->index, &it, forward);
  while ((pair = twltree_iter_next(&it)) != NULL)
    [fd append_row:pair len:8 tag:UPSERT];
}
void apply(const void *data, u16 tag)
{
  switch (tag) {
  case UPSERT:
    twltree_insert(self->index, data, true);
    break;
  case DELETE:
    twltree_delete(self->index, data);
    break;
  }
}
```

Сравнение

Тест: загрузка 50M *случайных* пар {a: u32, b: u32}. В левой колонке 8.3M уникальных значений, в правой 50M. Иными словами, на каждое уникальное значение «a» приходится около 6-ти значений «b»

	Разм. ГБ	Insert	Select
Octopus/Example	0.58	1:28	0:45
Octopus/Box	1.22	3:05	1:33
Tarantool	1.97	3:02	2:23
MySQL	2.3	24:18	—
PostgreSQL	3.0	4:40	—
Redis[1]	5.1	1:30	1:05
Octopus/Box[2]	1.7	1.12	0:20

[1] sadd lhs, rhs; lhs и ehs 4-байтные строчки

[2] hash index

Контакты/Где взять

- vostrikov@corp.mail.ru
- https://github.com/delamonpansie/octopus или https://gitlab.corp.mail.ru/octopus/octopus модули — отдельные ветки: octopus/master сам фреймворк, octopus/mod_example — учебный модуль.