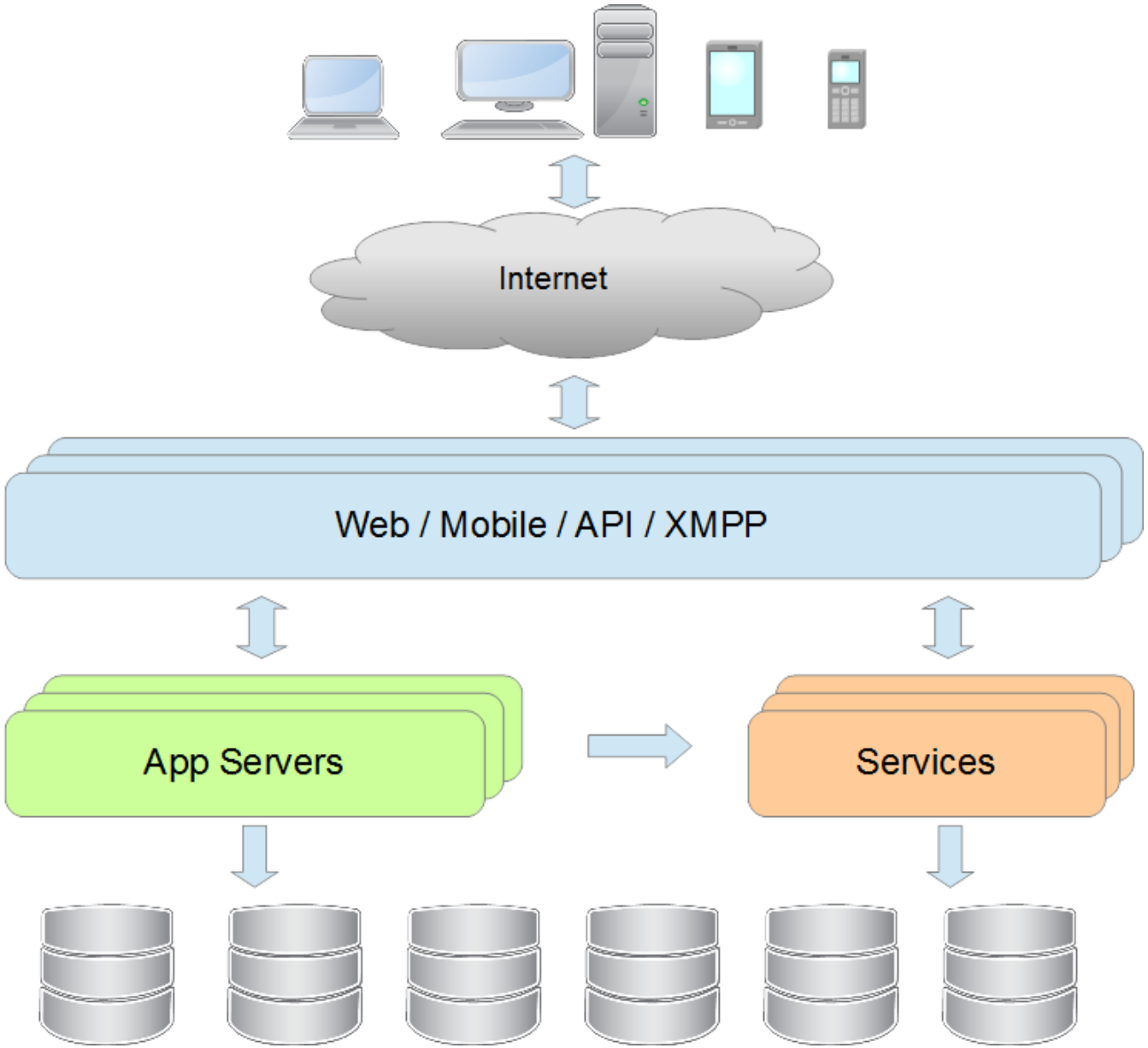
Франкенштейнизация Voldemort или key-value данные в Одноклассниках

Роман Антипин

инженер-программист
проекта Одноклассники


Internet
Web / Mobile / API / XMPP
App Servers
Services

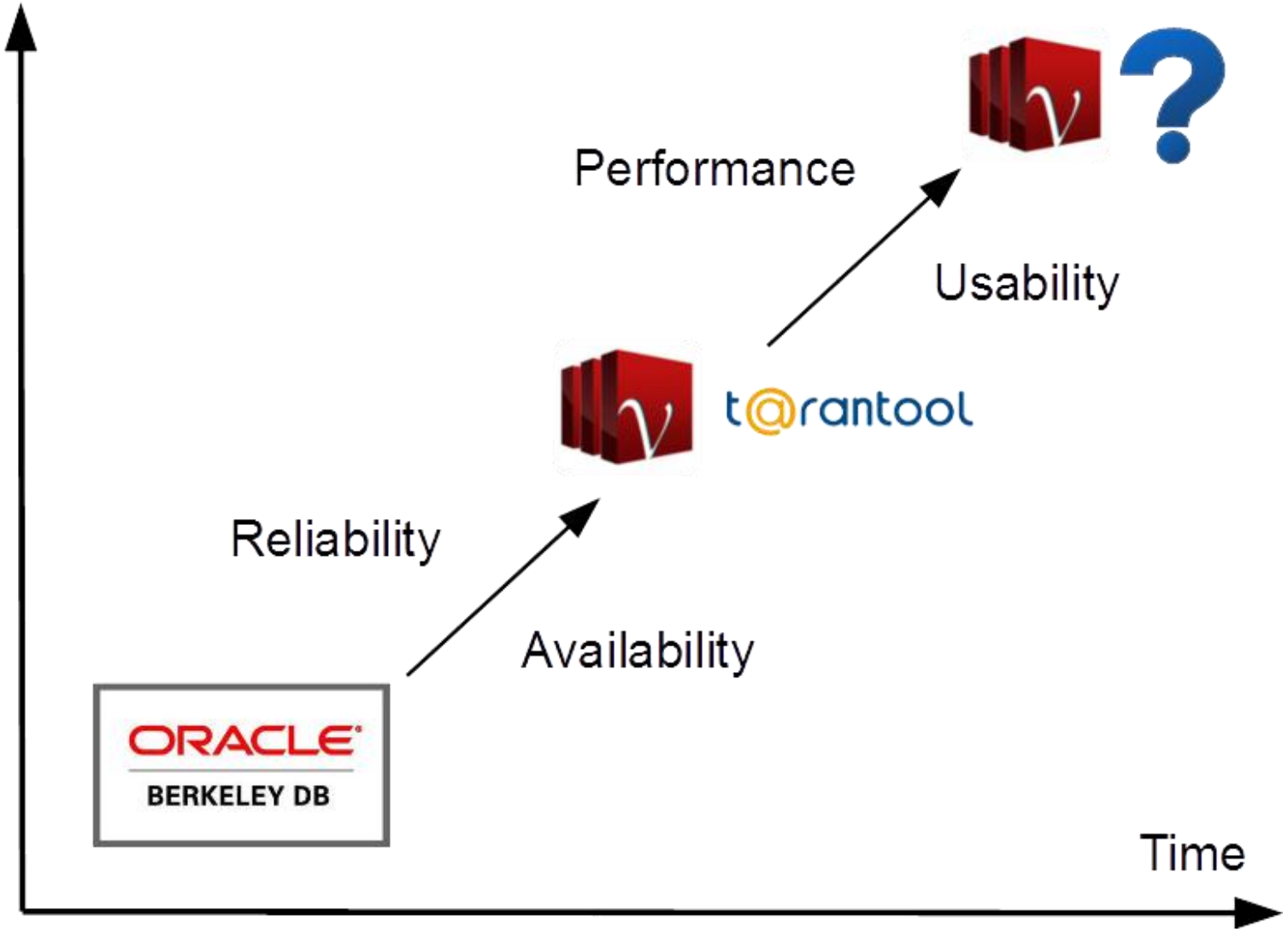
Данные в Одноклассниках

- SQL (MSSQL): ~ 330 серверов без учета backup, ~28 TB
- Blob storage (OBS): ~ 460 серверов, ~1.5 PB
- Column Family (Cassandra): ~ 410 серверов, ~70 TB
- Key-Value (Voldemort): ~ 220 серверов, ~2.2 TB
- Key-Value, Queue (Berkeley DB): ~100 серверов, ~1 TB

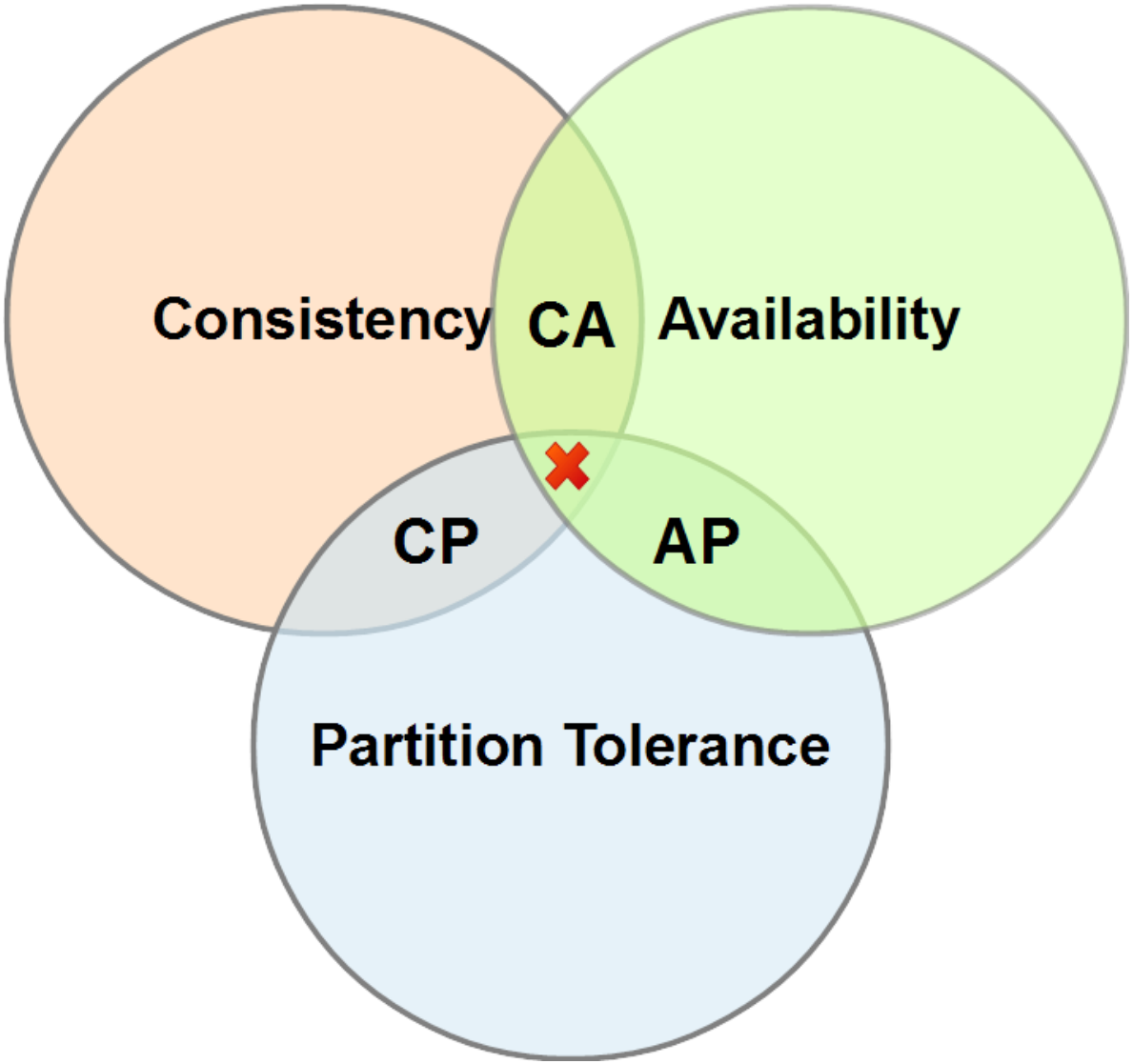
3 года назад:

- SQL (MSSQL)
- Key-Value, Queue (Berkeley DB)

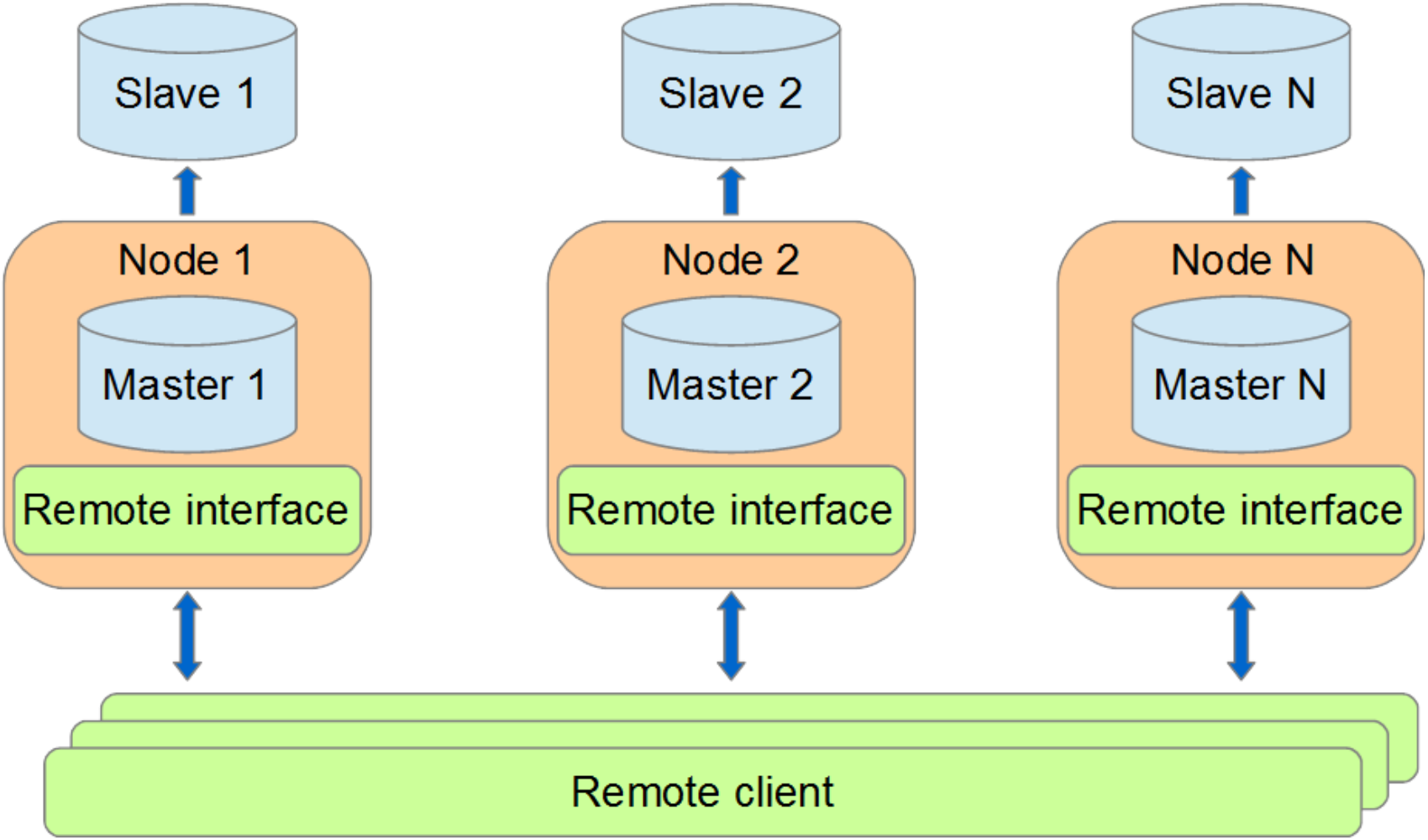
Развитие Key-Value хранилищ



CAP теорема



Решение на основе Berkley DB



Недостатки решения с Berkley DB

- Низкая отказоустойчивость (data corruption)
- Ограничение в масштабируемости
- Сложное обслуживание (ручная работа по восстановлению)
- Высокая стоимость (много update => slave == backup)
- Berkley DB 4.5 (Си) необходимо использовать JNI
  - Пробовали Berkley DB 5.0 (Си) – снижение производительности

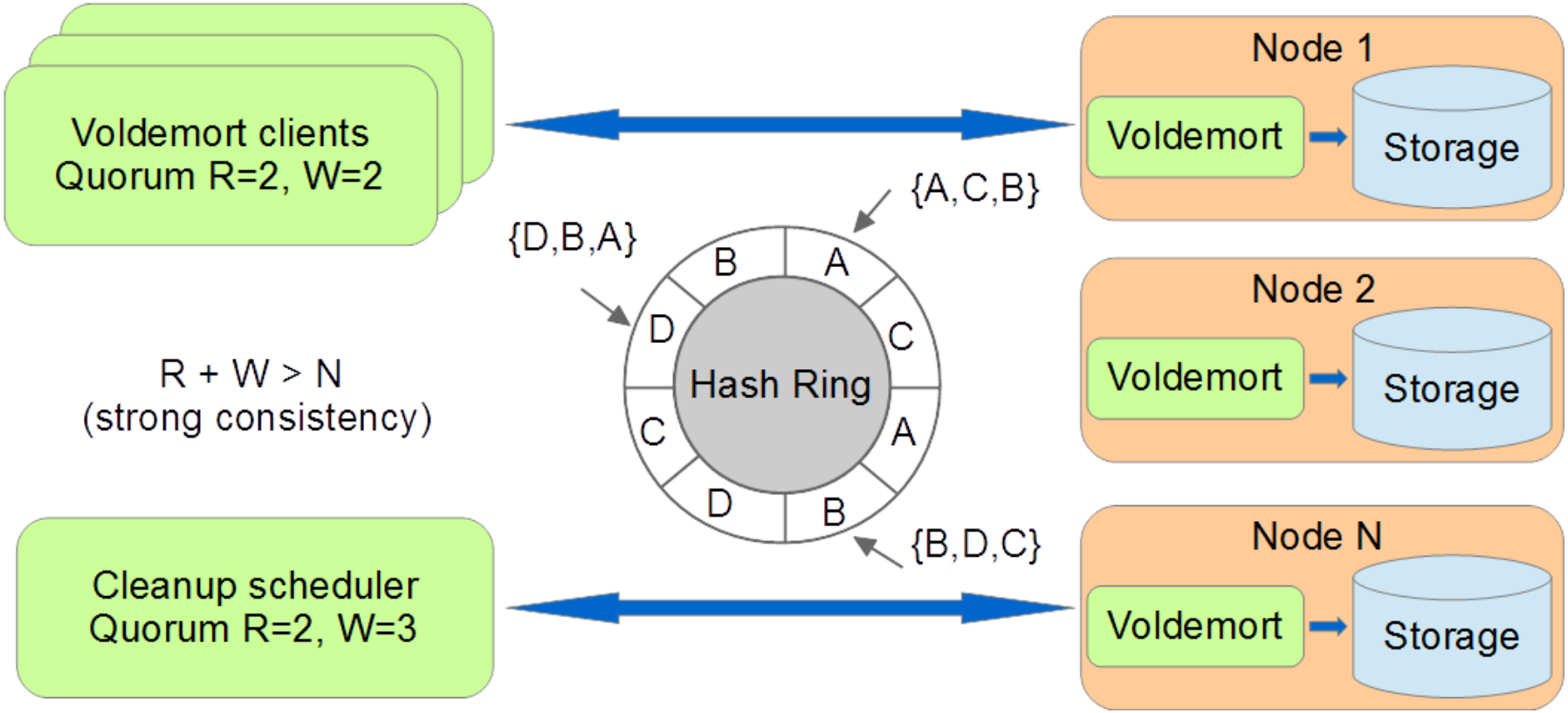
Требования к key-value хранилищу. Итерация 1

- Open source
- Высокая доступность, отказоустойчивость (data corruption)
- Неограниченная масштабируемость без down time
- Высокая производительность для отношения 3:R / 1:W
- Возможность использования дешевого железа
- Простота обслуживания (минимум ручной работы, только для восстановления персистентных отказов оборудования)

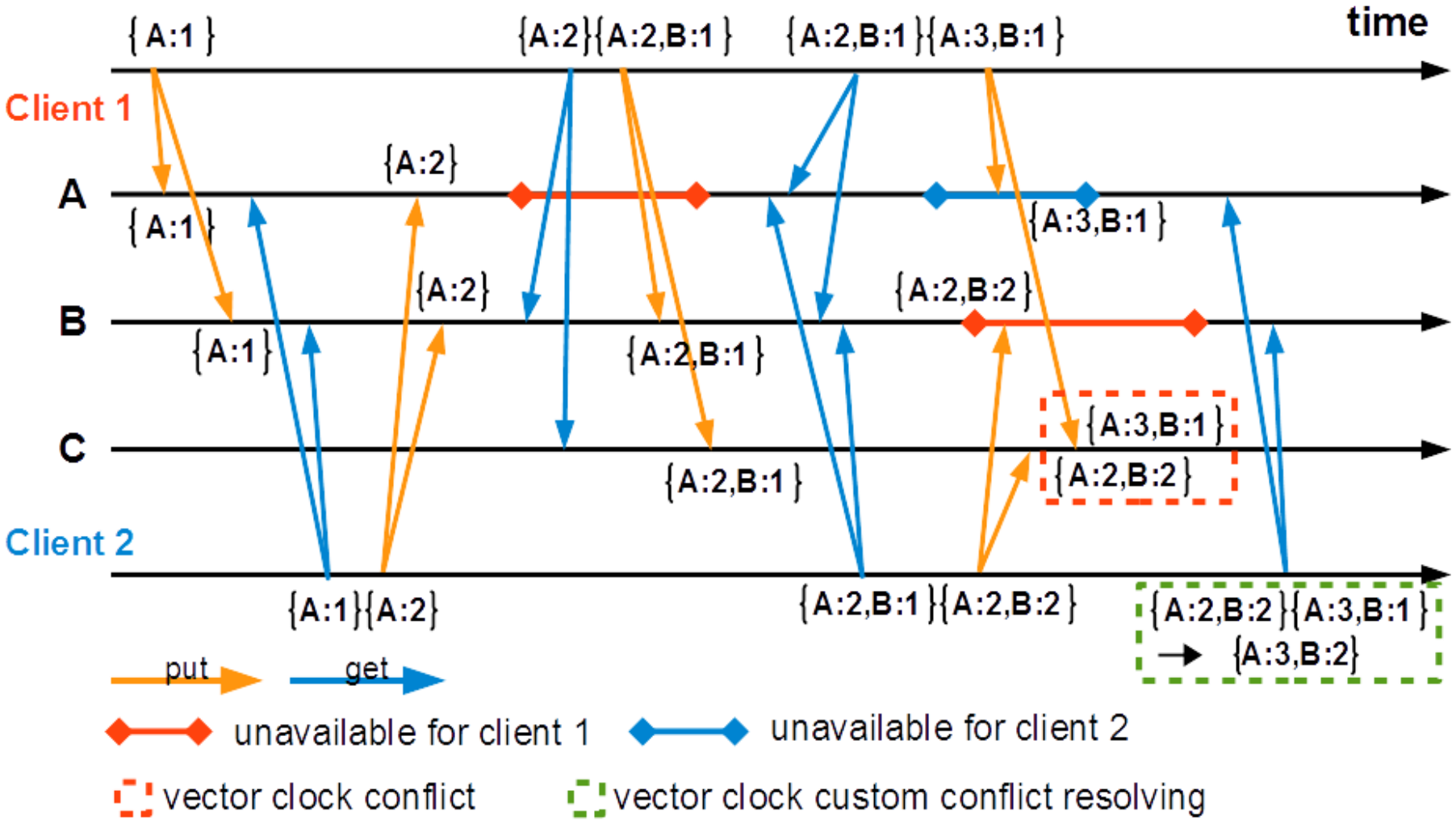
Анализ существующих решений (2010 г.)

- Cassandra 0.6.4
  - Column Family
  - Поддерживает Hinted Handoff
  - Java
  - Нет механизма определения конфликтов (timestamp)
- Voldemort 0.8.1
  - Pluggable backend (default Berkley DB)
  - Механизм определения конфликтов (Vector Clock)
  - Java
  - Отсутствие Hinted Handoff
- Riak
  - Сопоставим с Voldemort
  - Erlang

Архитектура Voldemort



Vector clock



Tarantool как back end для Voldemort

- In-Memory NoSQL database
  - Высокая производительность без деградации
  - Ограничение по RAM
- Персистентность за счет commit logs и snapshots
- Поддержка вторичных индексов
- Поддержка хранимых процедур на lua
- Опыт использования в @mail.ru
- Кооперативная многозадачность
- Отсутствие embedded решения
- Максимальный размер кортежа 1Мб

Исследование производительности

Тесты для кластера 3 ноды с replication factor 3, read/write quorum 2
Сервера SuperMicro 2 CPU(8 cores), RAM 32 Gb, 500 Gb SATA 7200.
Размер данных ~1-5Kb, объем данных 25Gb, read 50%, write 50%

- Voldemort 0.8.1 (BerkeleyDB java edition)
  - Read: ~32K req/sec
  - Write: ~7.5K req/sec
- Voldemort 0.8.1 (Tarantool)
  - Read: ~36K req/sec
  - Write: ~9K req/sec

Результаты использования Voldemort+Tarantool

- Нет повреждений данных (data corruption)
- Высокая доступность (устойчивость к падению любой ноды)
- Масштабируемость без down time
- Время старта кластера ~20 минут (объем ноды ~80G)
- Необходимость ручной работы только в случае полного восстановления ноды или расширения кластера
- Увеличение пропускной способности (throughput) в ~2 раза
- Увеличение среднего времени операций (latency) на ~50%
- Наличие небольшого количества повторных изменений

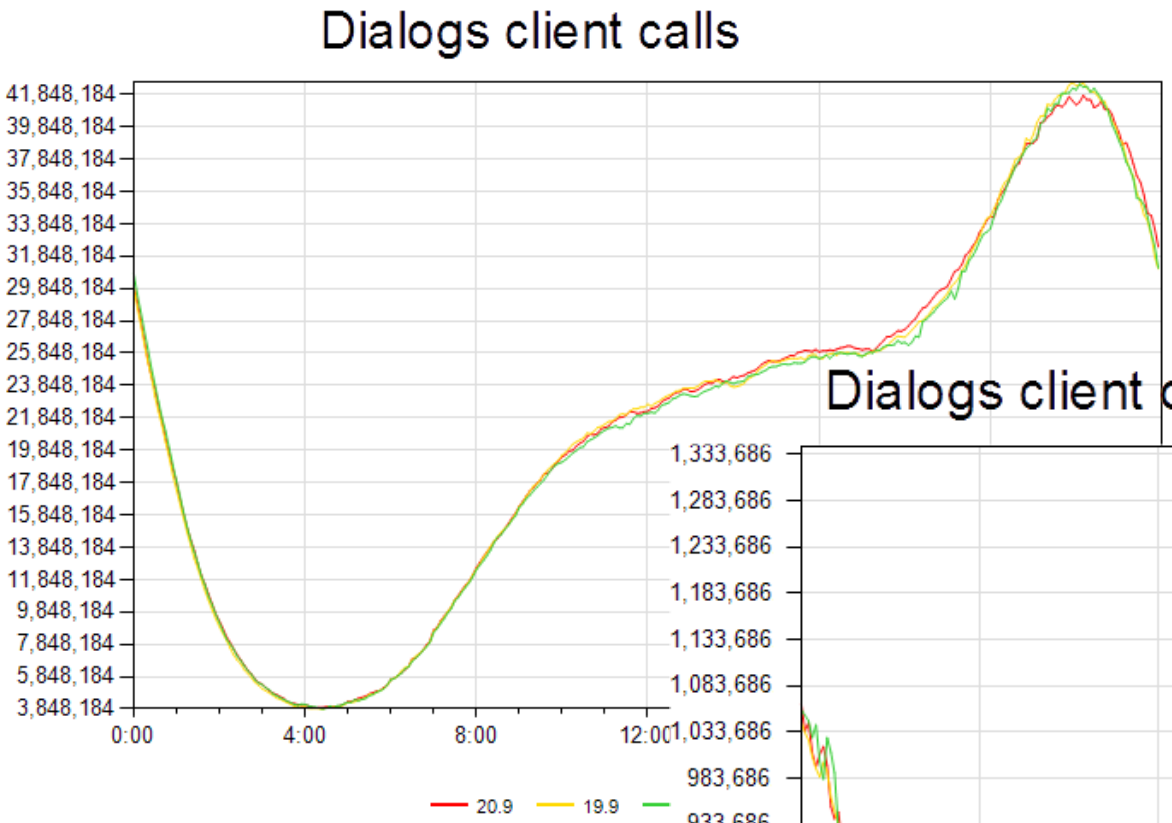
Примеры использования

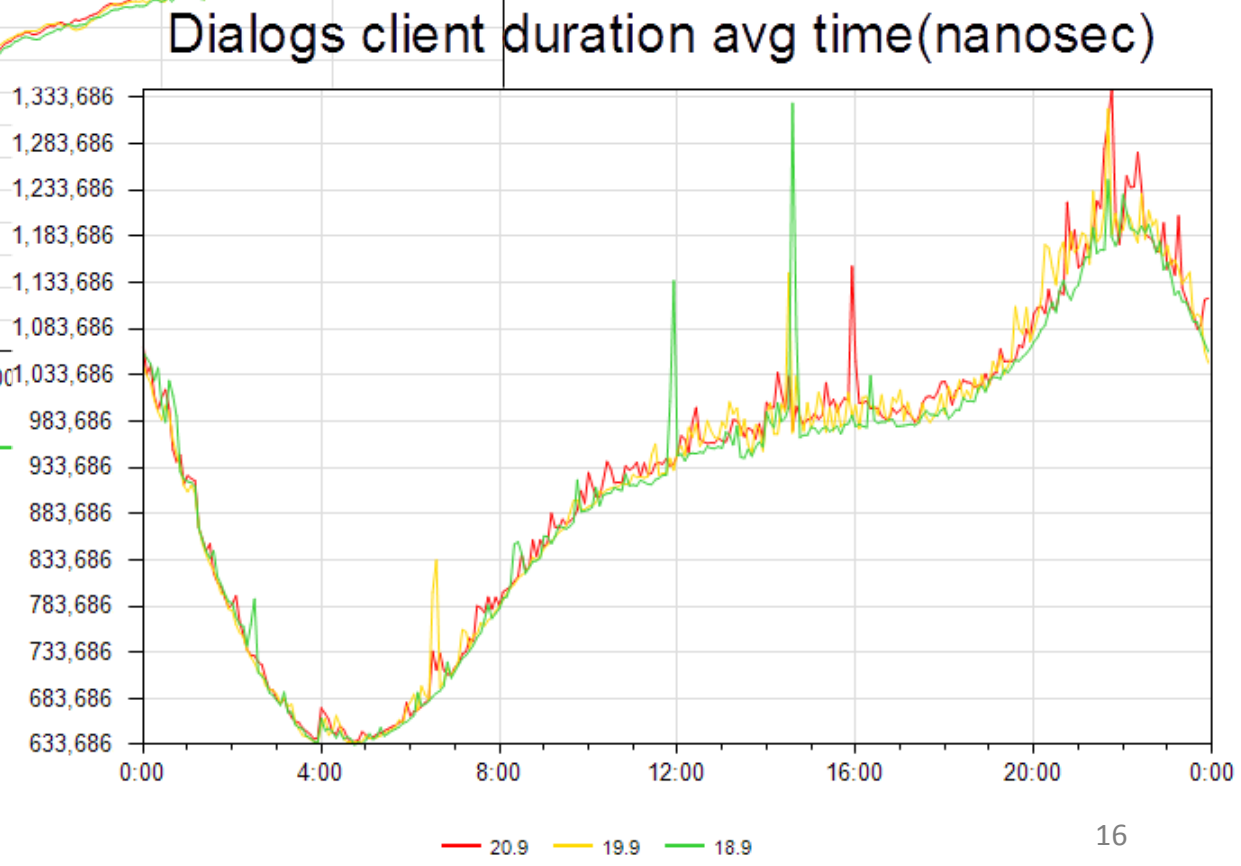
Всего на данный момент запущено 25 кластеров

Service	Servers	Calls / calls per node, ops	Average request time	Average data size	Cluster size
Instant-messenger	24	600k / 26k	1.3ms	1.5kb	224G
Suggested	12	140k / 11k	1.5ms	22kb	584G
Black list	12	400k / 33k	1.1ms	0.6kb	20G
Music playlist	9	37k / 4k	0.9ms	1.4kb	90G

P.S: ping local DC ~150µs , cross DC ~500µs

Графики на примере сервиса сообщений





одноклассники

Требования к key-value хранилищу. Итерация 2

- Избавиться от In-Memory ограничения
- Увеличение производительности
- Простая миграция на новое хранилище
- Embedded storage engine
- Желательно pure Java

Анализ решений на замену Tarantool

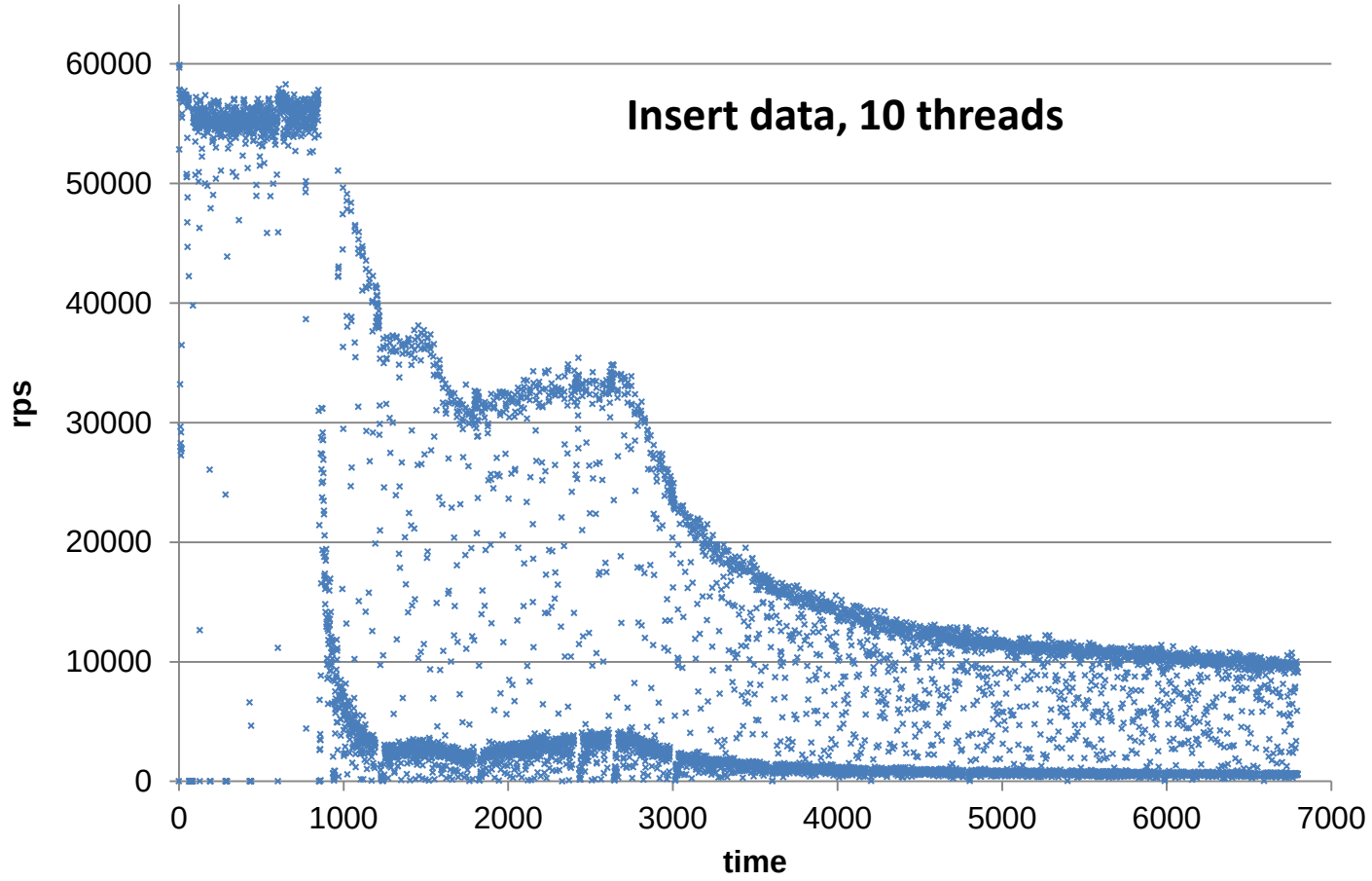
- Kyoto Cabinet 1.2.76
  - Key-Value
  - Различные типы storage (HashDB, TreeDB, …)
  - C++
- LevelDB 1.9.0
  - Key-Value
  - Использует LSM Tree и SSTable
  - C++
- Cassandra 0.6.4
  - Column Family
  - Использует LSM Tree и SSTable
  - Java

Тестовая конфигурация

- Тестовый сервер
  - Intel Xeon CPU E5506 2.13GHz
  - RAM: 48Gb
  - Linux kernel: 2.6.34
  - HDD ST3500320NS (7200rpm 32Mb)
  - SSD INTEL SSDSA2CW300G3
  - File system: xfs rw, noatime, nodiratime, delaylog, noquota
- Тестовые данные
  - 90M записей key – long , value – byte[1000]

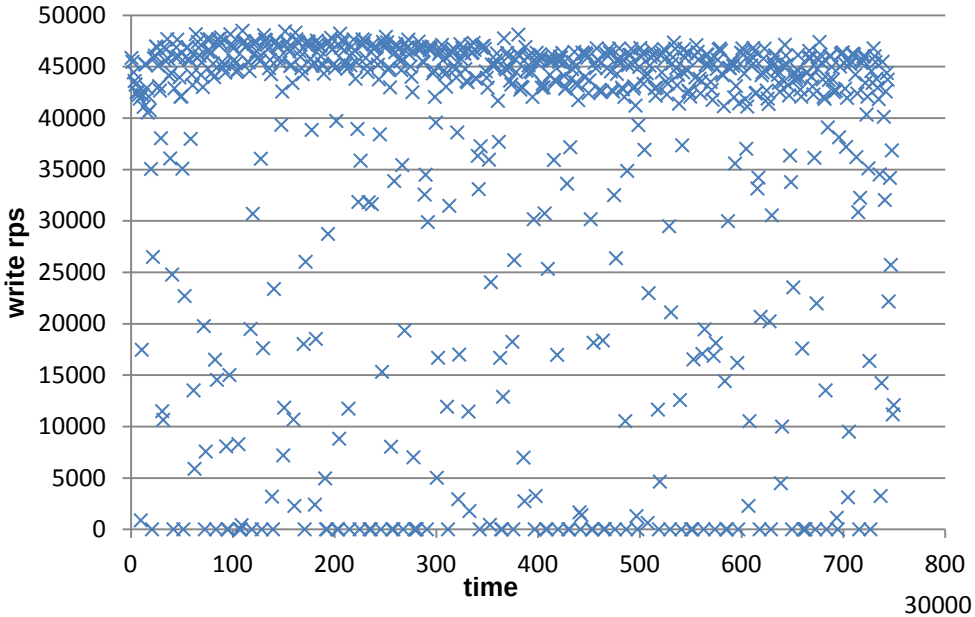
Kyoto Cabinet 1.2.76

Параметры: HashDB, opts=TLINEAR, bnum=200M, msiz=30G, dfunit=8



одноклассники

LevelDB 1.9.0

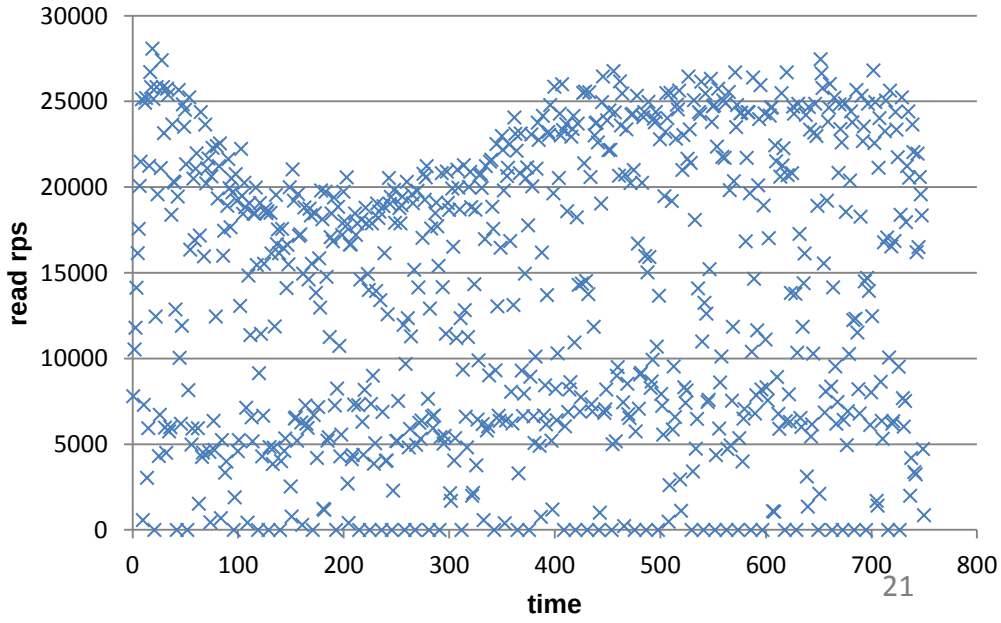


Параметры конфигурации:

- max_open_files=100K
- block_size=4000 byte
- write_buffer_size=512M
- block_cache=LRUCache(30G)
- filter_policy=NewBloomFilterPolicy(10)
- logs - hdd, sstables – ssd,
- увеличен размер L0

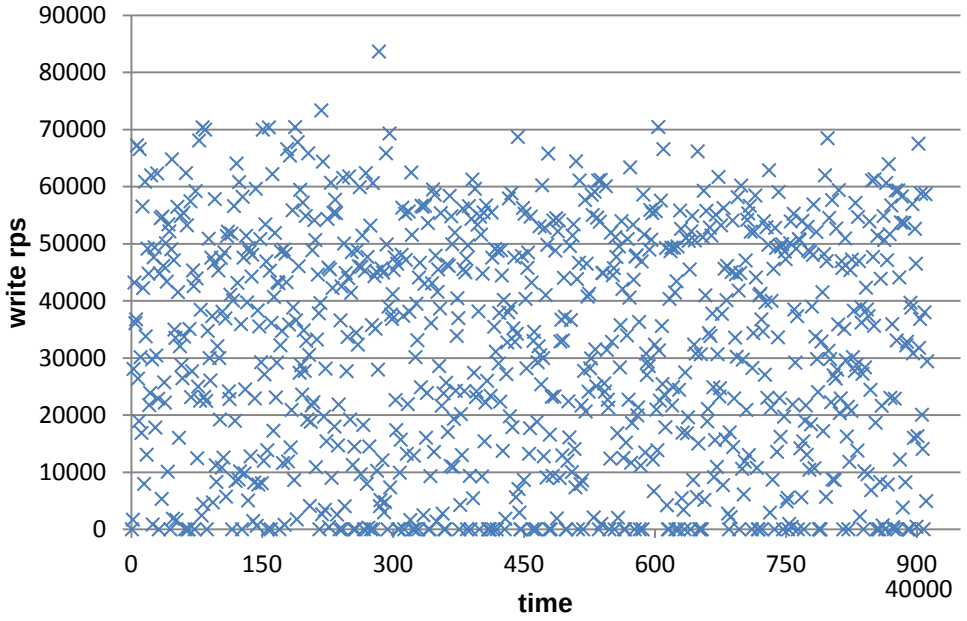
Параметры теста:

- cache hit rate 90%
- read threads 30
- write threads 10



одноклассники

Cassandra 0.6.4

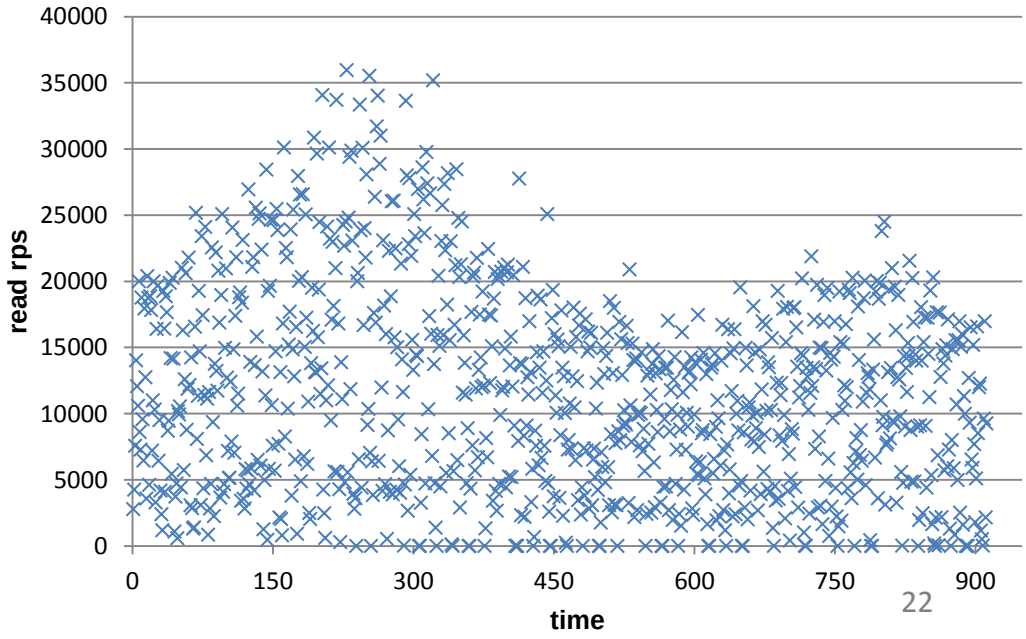


Параметры конфигурации:

- RowsCached=15M
- MemtableThroughputInMB=512
- IndexInterval=128
- CommitLogRotationThresholdMB=128
- DiskAccessMode=mmap
- logs - hdd, sstables – ssd

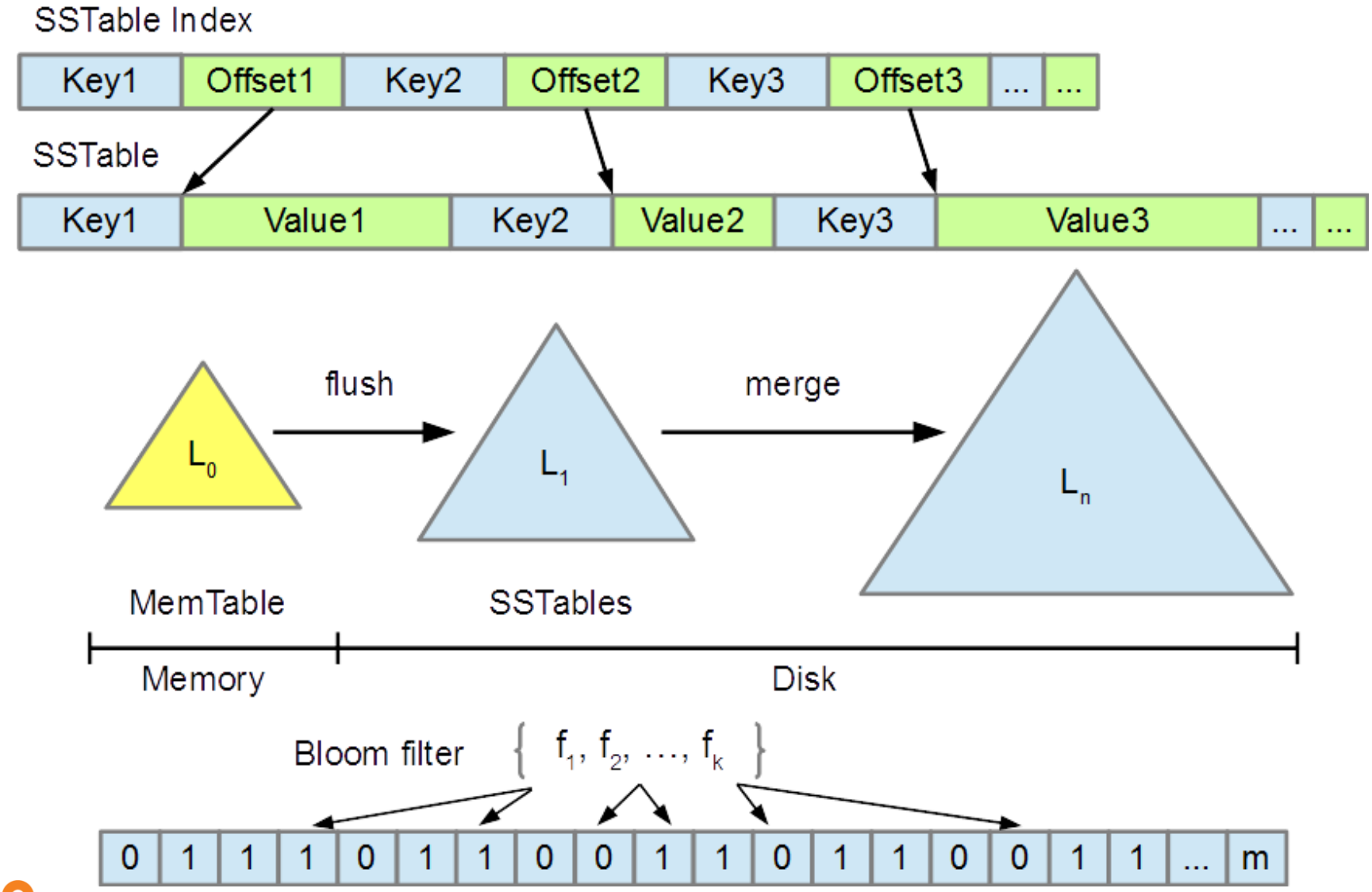
Параметры теста:

- cache hit rate 90%
- read threads 30
- write threads 10



одноклассники

LSM Tree & SSTable



Интересные факты

- Не используем Cassandra cache
  - Используем ConcurrentLinkedHashMap SECOND_CHANCE
  - Производительность ConcurrentLinkedHashMap 1.4 LRU < ~50%
  - В качестве Value MemoryWrapper через sun.misc.Unsafe

```
public class MemoryWrapper implements Memory {

    private final AtomicInteger references = new AtomicInteger(1);
    private final long position;
    private final int size;
```

- Off heap allocation malloc (glibc 2.5) -> tcmalloc
  - Отсутствие деградации
  - Average allocation time ~ 800 ns против десятков μs
- Тюнинг linux по примеру LVS и HAProxy, снижение average client time на ~30%

Результаты

- Снято ограничение In-Memory
- Embedded java storage
- Возможность максимально использовать железо
- Увеличение производительности
  - Instant-messenger (~1.5 kb данные) с одной ноды ~30K -> ~90K
  - Cache hit rate ~95%
  - Reads ~75K Writes ~15K
  - Сокращение количества серверов с 24 -> 12
- Cleanup без полного обхода, сохраняется информация о tombstone на каждой ноде

Спасибо!

Контакты

http://habrahabr.ru/company/odnoklassniki/blog/

https://github.com/odnoklassniki/

roman.antipin@odnoklassniki.ru

http://v.ok.ru