Владимир Колобаев

Lead system administrator

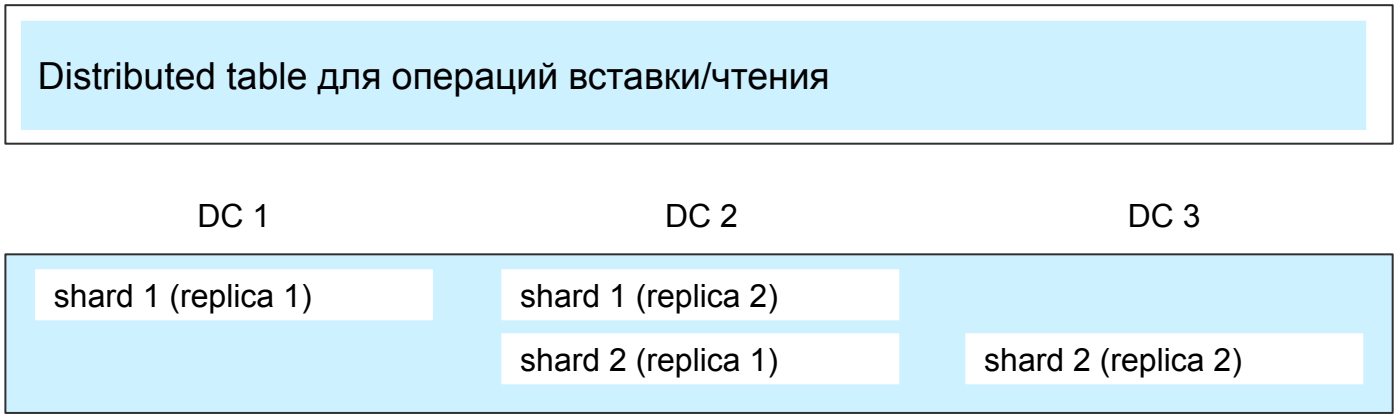
Принципиальная схема кластера. (Graphite)

Distributed таблица: jumpConsistentHash(cityHash64(Path), 3))

```
CREATE TABLE graphite.data ON CLUSTER ssd_cluster (
        `Path` String,
        `Value` Float64,
        `Time` UInt32 CODEC(Delta(4), ZSTD(1)),
        `Date` Date,
        `Timestamp` UInt32
) ENGINE = ReplicatedGraphiteMergeTree(
        '/clickhouse/tables/graphite/{shard}/data', '{replica}',
        Date,  (Path, Time), 8192, 'graphite_rollup'
)
```

Shard	table	size	rows	byte per row	parts
1	data	2.65 TiB	950965609962	3.066	45
2	data	2.53 TiB	887760745964	3.133	45
3	data	2.63 TiB	946029211229	3.051	45

Принципиальная схема кластера. (Logs)



Distributed таблица: random()

- DB для логов Nginx
- DB для логов Services
- DB для инфраструктурных логов управления Servers/LXCs

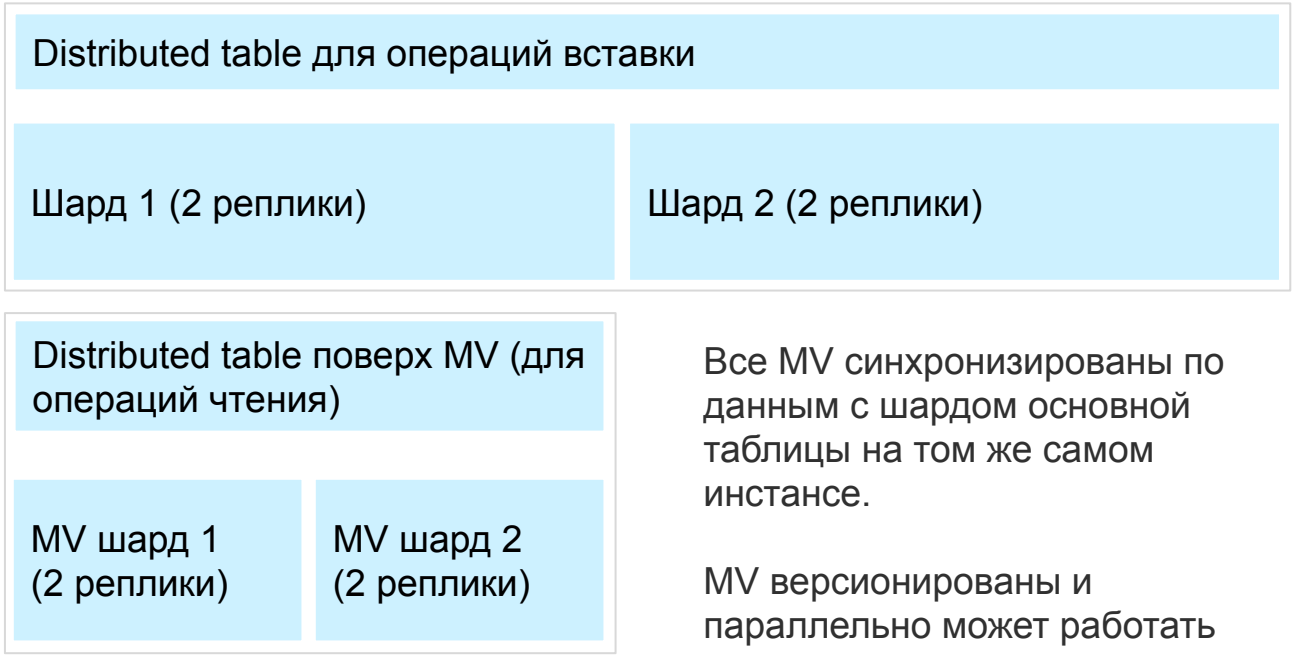
DB	table	size	rows	days	avgDaySize
nginx	access	57.24 TiB	2578258974631	287	204.22 GiB
nginx	error	436.15 GiB	20422933476	535	834.80 MiB
services	moira	380.27 GiB	10649132201	67	5.68 GiB
services	anomaly_detector	251.53 MiB	5971534	4	62.88 MiB
infra	lxc	145.99 KiB	5658	10	14.60 KiB

Максим Котяков

Senior backend engineer

Принципиальная схема кластера.

Планируется переезд на 3 реплики по 3 шарда в каждой с полной репликой всех данных в рамках одного ДЦ.



Все MV синхронизированы по данным с шардом основной таблицы на том же самом инстансе.

MV версионированы и параллельно может работать несколько поколений.

Основная таблица:

- “широкая” (> 50 столбцов);
- unique по генерируемому уникальному идентификатору (ReplacingMergeTree);
- подневные партиции.

Таблицы под MV:

- “узкие” (5 - 10 столбцов);
- unique по различным наборам параметров (ReplacingMergeTree);
- measures хранятся в виде агрегатных функций;
- партиционирование согласовано с потребностью быстро отдавать данные (в некоторых и с необходимостью обеспечивать replacing значений из глубокого прошлого): дневное, недельное или вообще отсутствует.

Нагрузка на чтение:

- пока около 5000 rpm при 8% cpu usage.

Нагрузка на запись:

- стабильная около 500K rows per minute, в пиках до 10M rows per minute.

Негарантированная доставка

Гарантированная доставка

Сос Саакян

Middle engineer в направлении big data

Кластер AB

Шард 1 (1 реплика)	Шард 2 (1 реплика)

Каскад MV

Парсинг JSON:
- смотрит в локальную, записывает в Distributed;
- ~ 50 полей, парсятся из сырого json-а.

Counters:
- смотрит в локальную, записывает в локальную;
- ~ 15 полей.

Бизнес-логика (несколько параллельных MV):
- смотрит в локальную, записывает в локальную;
- ~ 35 полей.

Python скрипт:
- запускается раз в 15 минут;
- считает статтесты.

Отчеты:
- настроены на одну ноду(хотим прикрутить прокси-балансер);
- используем внешние словари в отчетах.

Внешние словари:
- реквизиты прописываем в настройках odbc драйвера;
- в настройках кликхауса все остальное.

Мониторинг над Distributed таблицами в пайплайне MV:
- проверка на количество in-out строк с учетом логики MV;
- при последнем обновлении кликхауса на кластере функция notEmpty() изменила ожидаемое поведение. Заменил функцией length().

TTL:
- данные чистятся по cron-у, дропом партиций.