Yandex Cloud

# Движение к универсальности: гибридная OLTP-база с поддержкой OLAP-запросов

Алексей Дмитриев,
Технический менеджер YDB
slonnn@yandex-team.ru

- Что такое HTAP и почему HTAP все хотят
- При чем тут YDB
- Виды HTAP
- Почему так мало баз данных с HTAP

# Что такое YDB

- NewSQL база данных
- Shared Nothing + разделение compute + storage
- OLTP-нагрузка

- Используется в Яндекс Маркете, Яндекс.Метрике и Yandex Cloud
- 500+ ТБ данных, 1.5+ млн RPS в одной БД

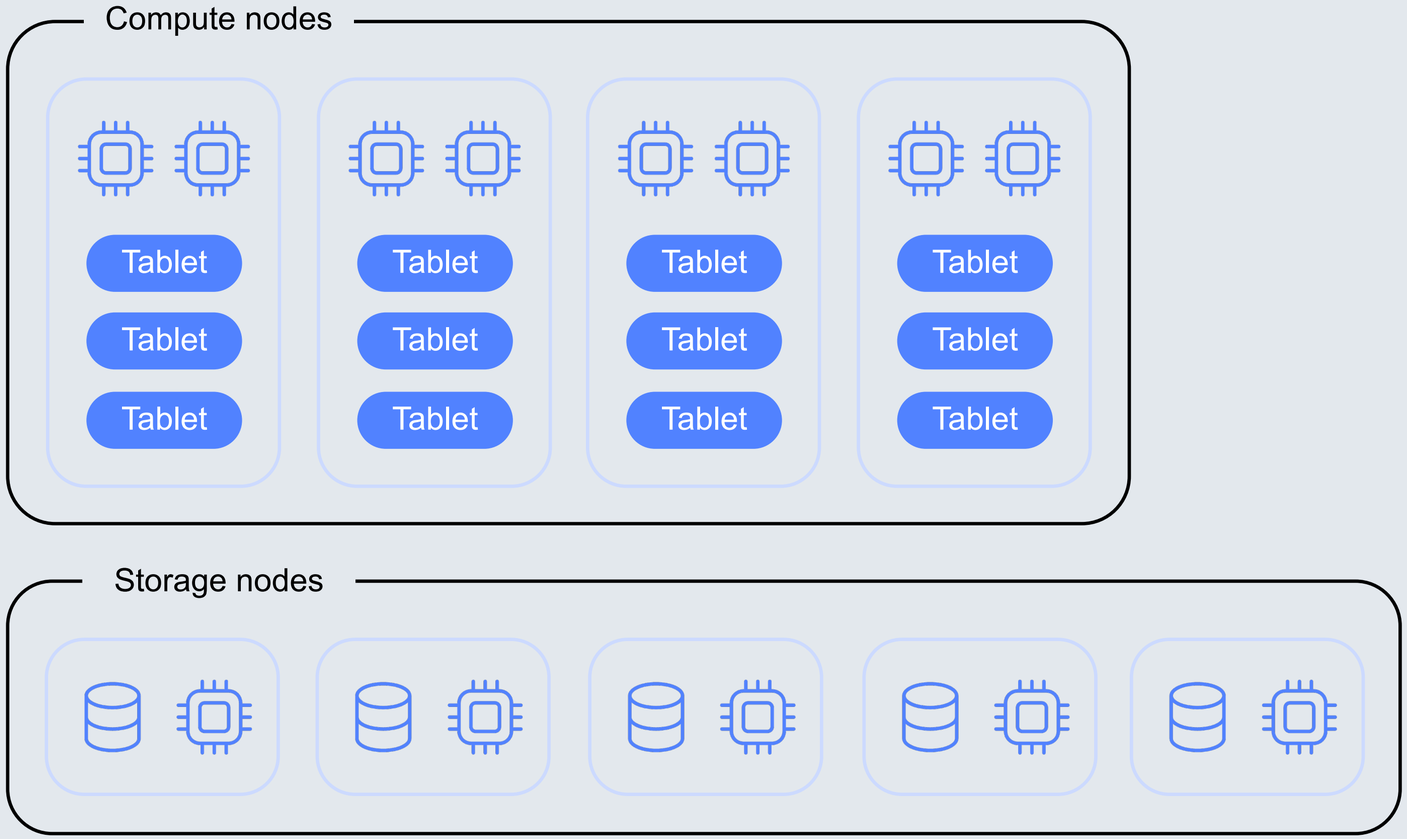


# Разделение слоёв Compute и Storage

Таблетки - RSM (Replicated State Machine)

Среды выполнения
для таблеток
и запросов запущены
на вычислительных узлах

Данные размещены
на узлах хранения



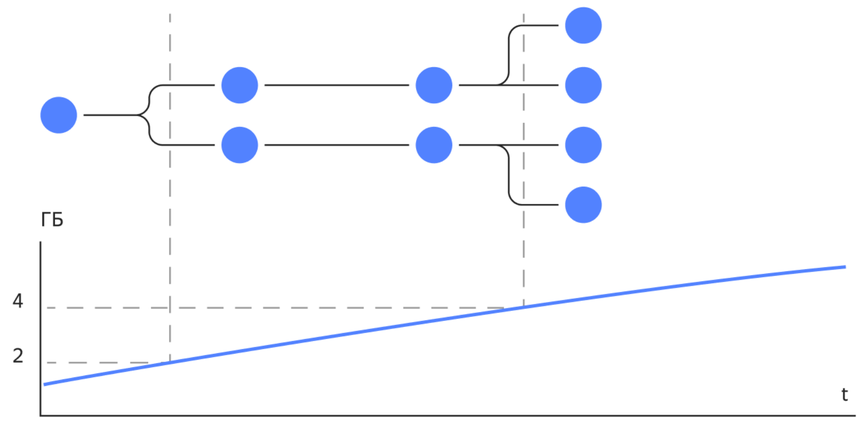
# Автошардирование

Автошардирование при росте объёмов данных и изменении нагрузки

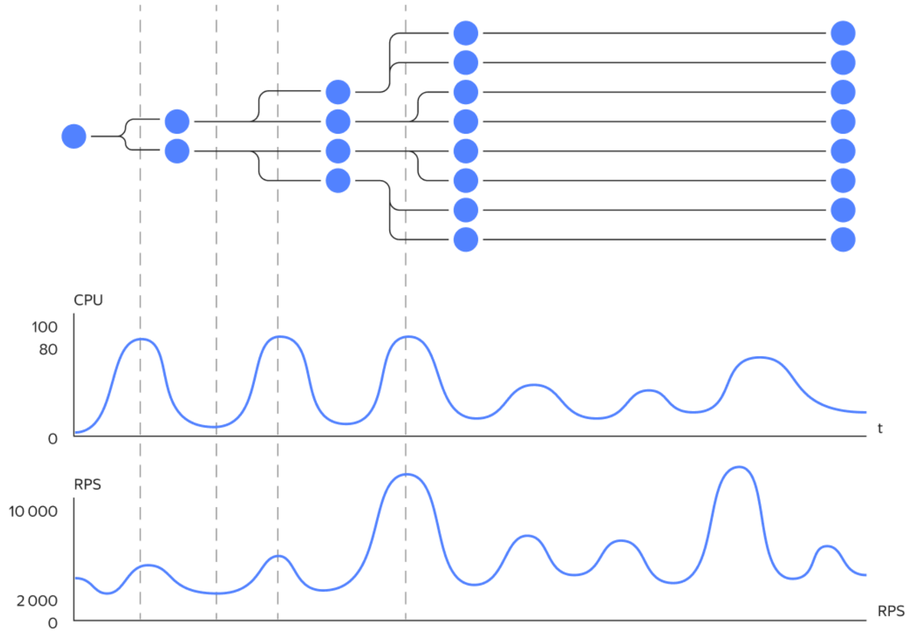
По размеру

По нагрузке

## Шардирование по объёму данных




## Шардирование по нагрузке



# Зачем OLAP

- По мере роста OLTP данных рано или поздно возникает задача анализа данных
- Не так много баз данных, способных обработать петабайты данных

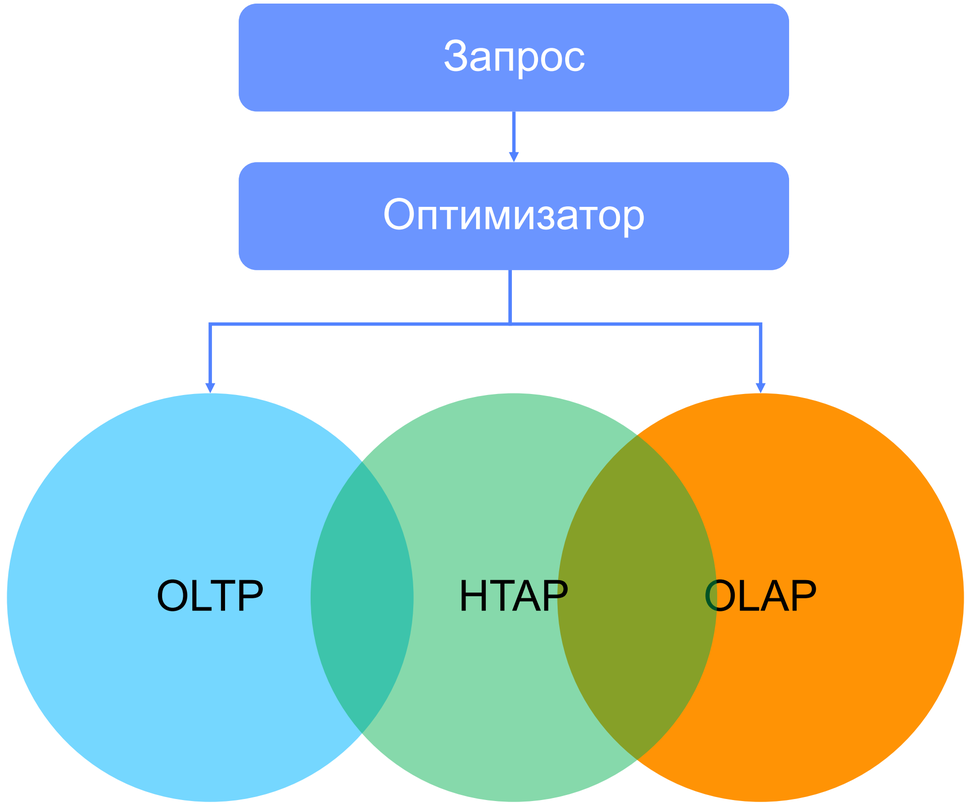
## Требования

- Хранение больших (терабайты-петабайты) объёмов данных
- Вставка редко изменяемых данных в больших объёмах с высокой скоростью
- Эффективный поиск по большим объёмам данных

# HTAP – гибридная транзакционно-аналитическая обработка

Одна база данных на оба вида нагрузки: OLTP, OLAP

- Нет необходимости в ETL
- Выполнение аналитических запросов над свежими транзакционными данными (оптимизатор запросов выбирает куда обращаться за данными в зависимости от запроса)

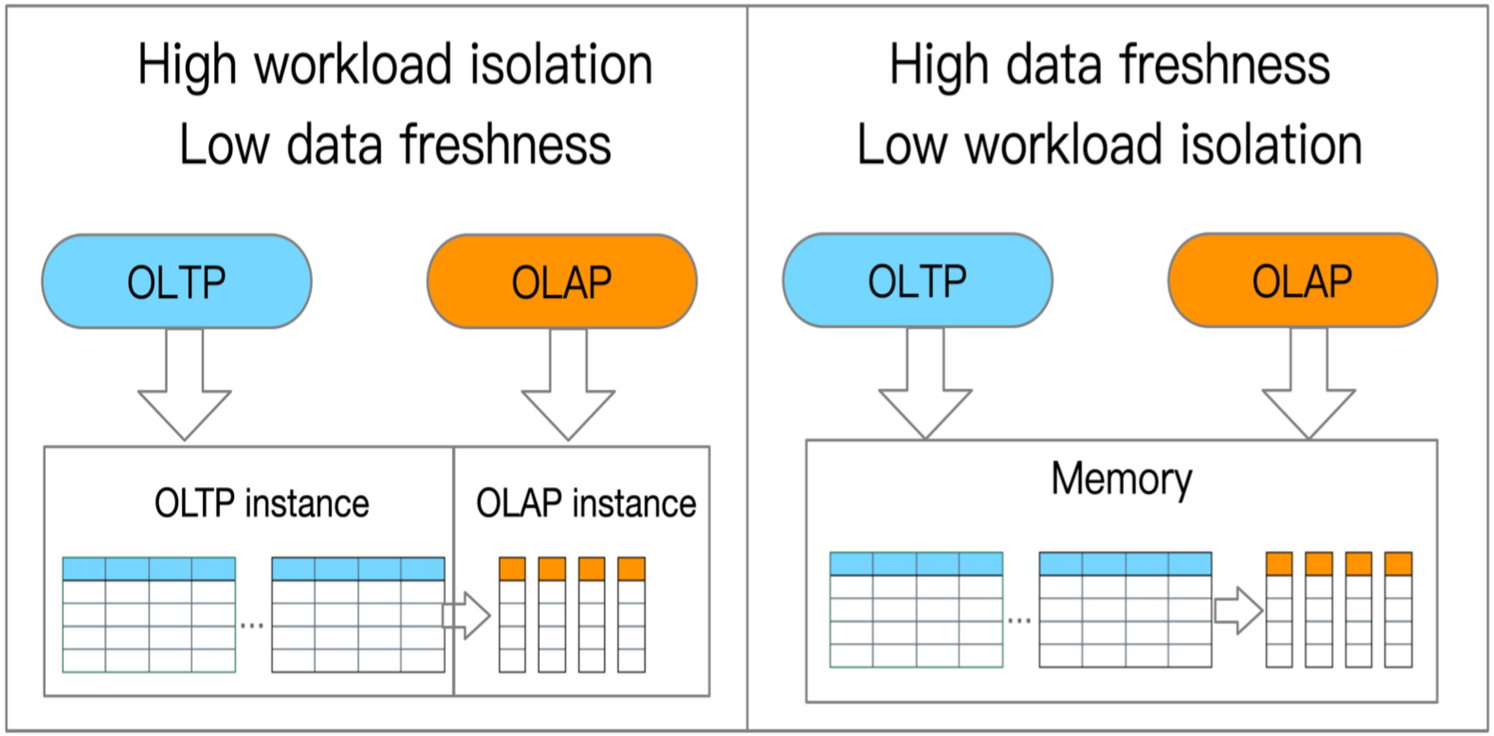


# Дилемма HTAP

- Свежесть данных

Или

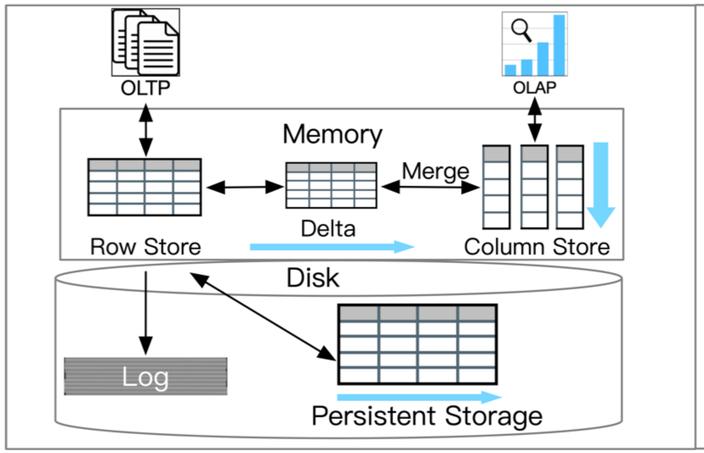
- Изоляция TP/AP



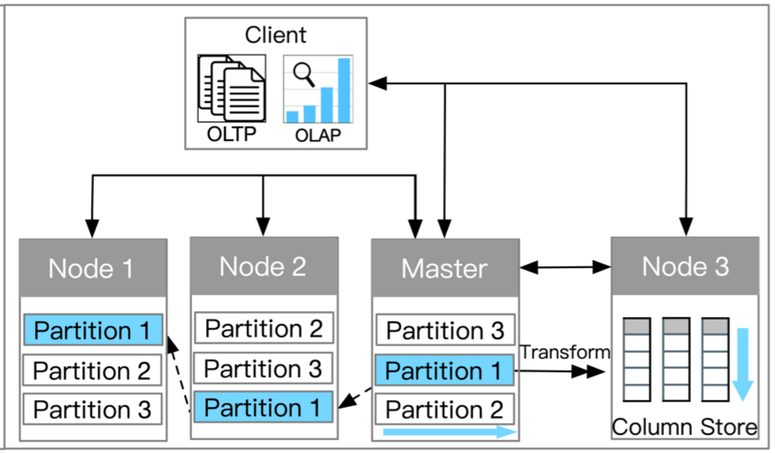

SIGMOD'22 Tutorial


* Из презентации TiDB

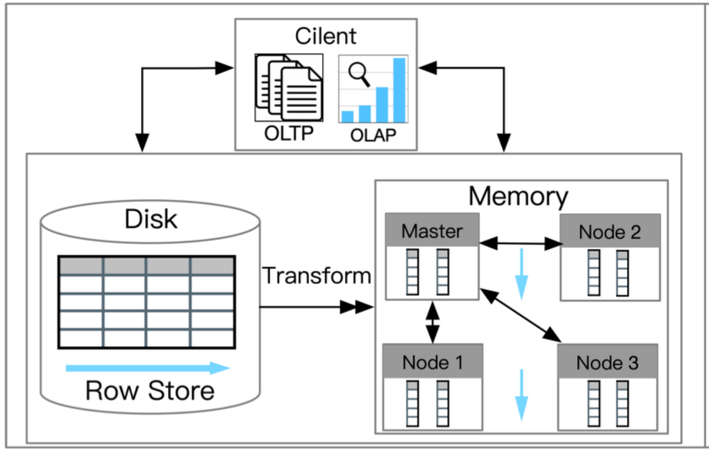
# Основные виды HTAP



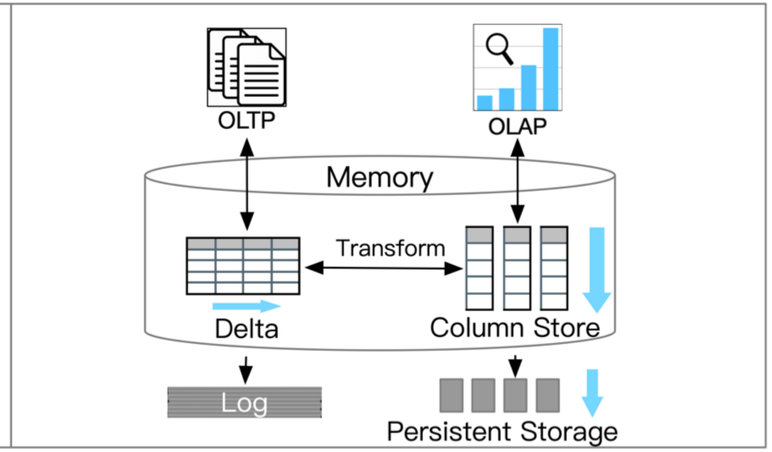

(a) Primary Row Store+In–Memory Column Store




(b) Distributed Row Store+Column Store Replica




(c) Disk Row Store+Distributed Column Store




(d) Primary Column Store+Delta Row Store

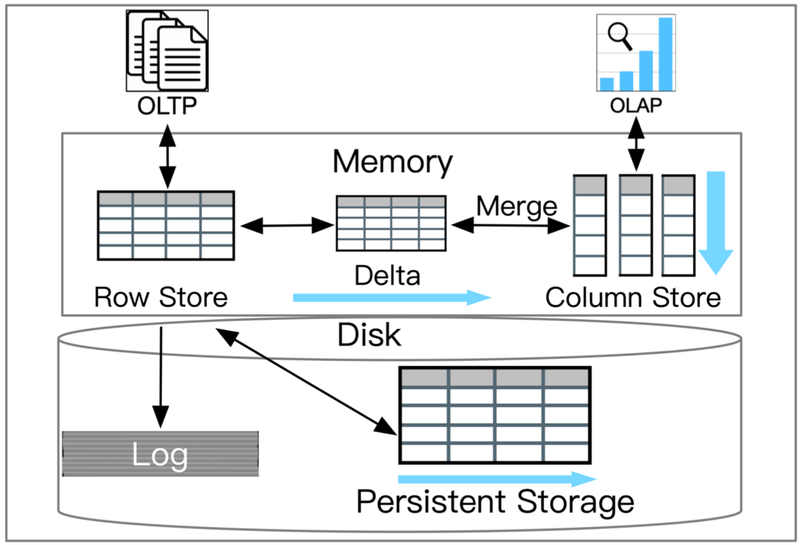
# Primary Row Store + In memory Column Store

**Плюсы**

- Высокая пропускная способность TP
- Высокая пропускная способность AP
- Высокая свежесть данных

**Минусы**

- Низкая масштабируемость
- Низкая изоляция TP/AP




Oracle, SQL Server

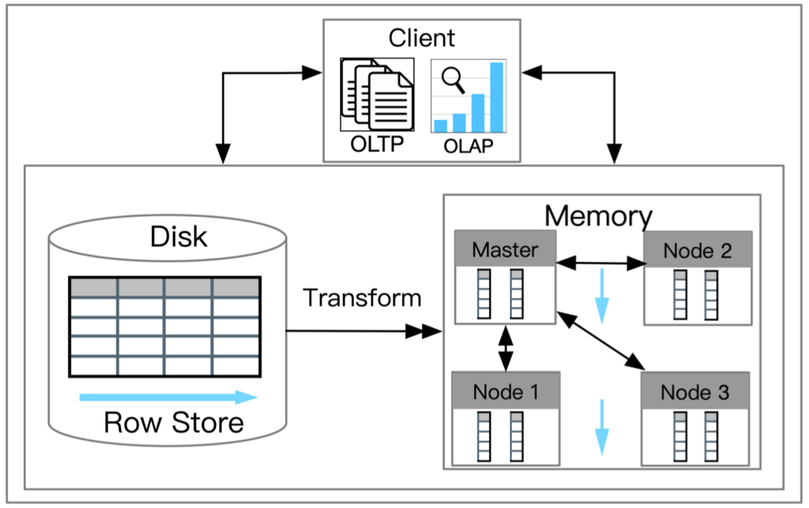
# Disk Row Store + Distributed Column Store

**Плюсы**

- Высокая пропускная способность AP
- Высокая изоляция данных TP/AP

**Минусы**

- Низкая свежесть данных
- Низкая масштабируемость




MySQL Heatwave, Oracle RAC

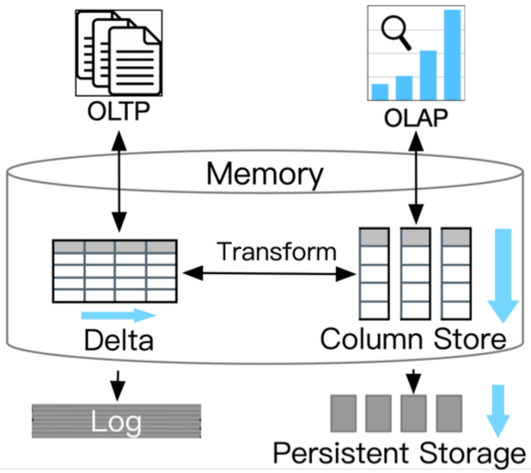
# Primary Column Store + Delta Row Store

**Плюсы**

- Высокая свежесть данных
- Высокая пропускная способность AP

**Минусы**

- Низкая пропускная способность TP
- Низкая изоляция TP/AP




SAP HANA, Snowflake Unistore

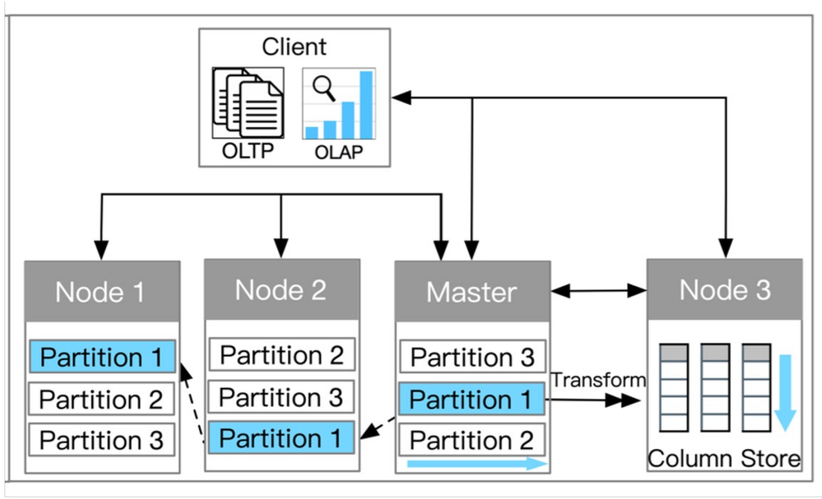
# Primary Column Store + Delta Row Store

**Плюсы**

- Физическое разделение ресурсов
- Возможность хранить произвольный объем данных
- Возможность использования как standalone аналитической базы данных

**Минусы**

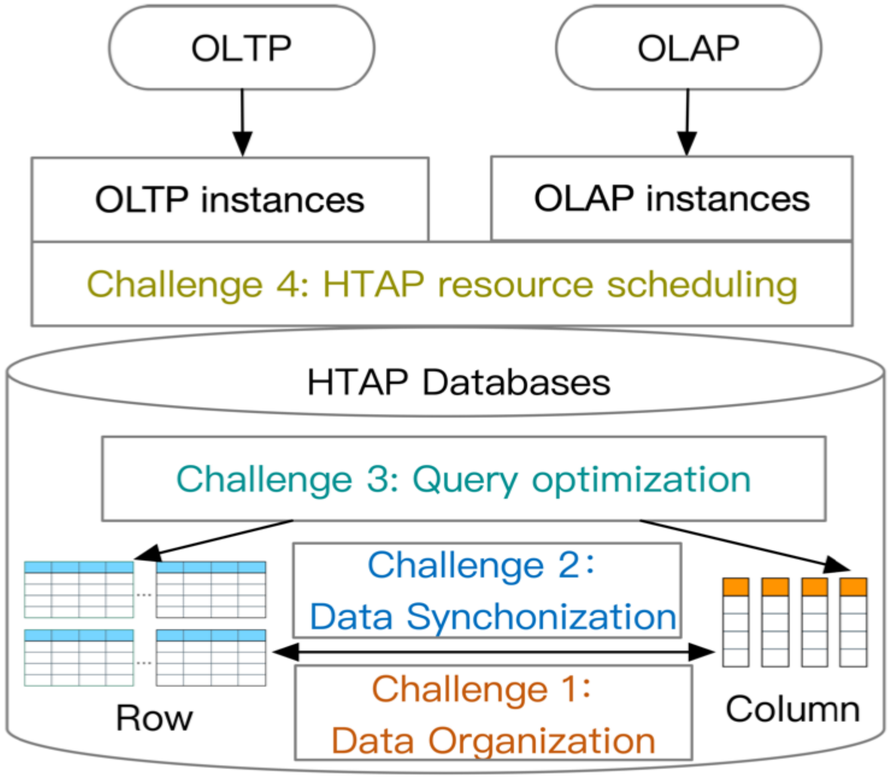
- бОльшая задержка обновления данных (секунды)




Google F1 Lightning, TiDB, YDB

# Challenges for HTAP databases

- **Challenge 1 (Data Organization): how to organize the data adaptively for HTAP workloads with high performance and low storage cost.**
- **Challenge 2 (Data Synchronization): how to synchronize the data from the row store to the column store for high throughput and data freshness**
- **Challenge 3 (Query Optimization): how to optimize the query with both row store and column store by exploring the huge plan space.**
- **Challenge 4 (Resource Scheduling): how to schedule the resources for OLTP and OLTP instances effectively for high throughput and data freshness.**



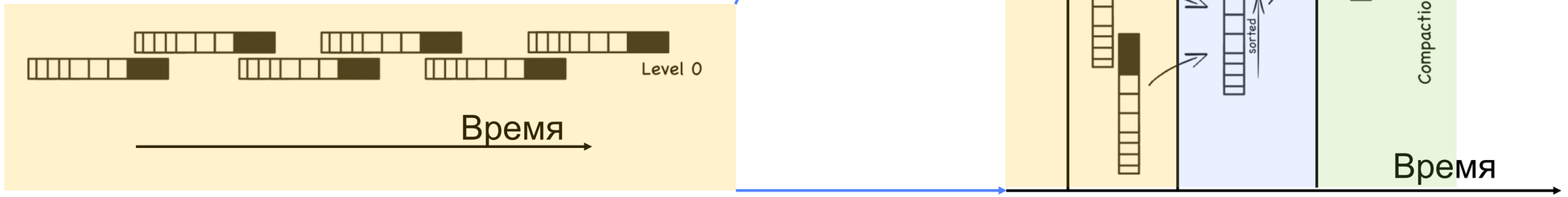
1. Организация хранения данных
2. Оптимизация запросов
3. Планирование и управление ресурсами

# Организация хранения данных

- Random IO это дорого
- Неизменяемые структуры данных
- Хранение данных в LSM или B+деревьях

## Особенности

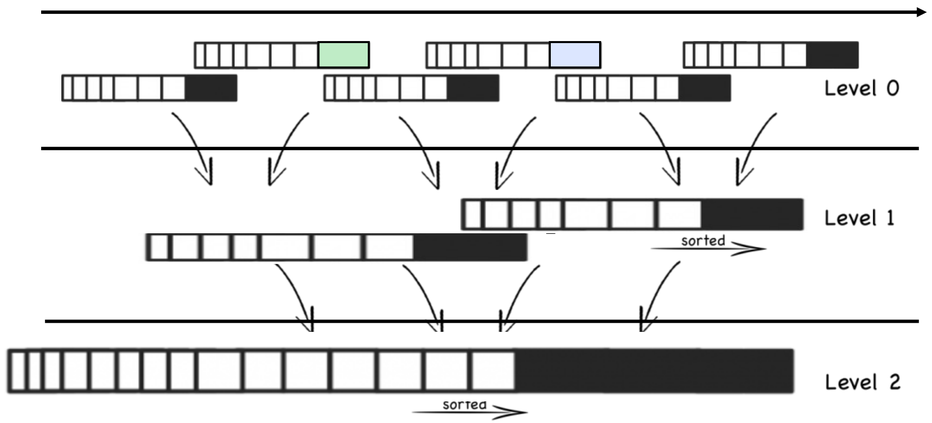
- Read amplification
- Write amplification
- Space amplification



# Tiered vs Leveled LSM

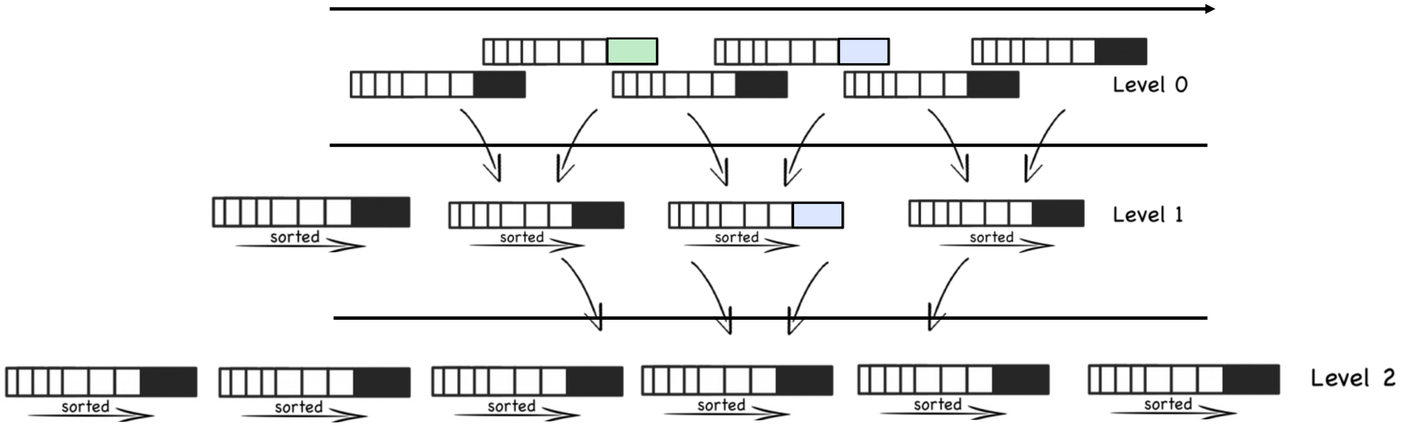
## Size tiered compaction

- Read amplification
- Write amplification
- Space amplification



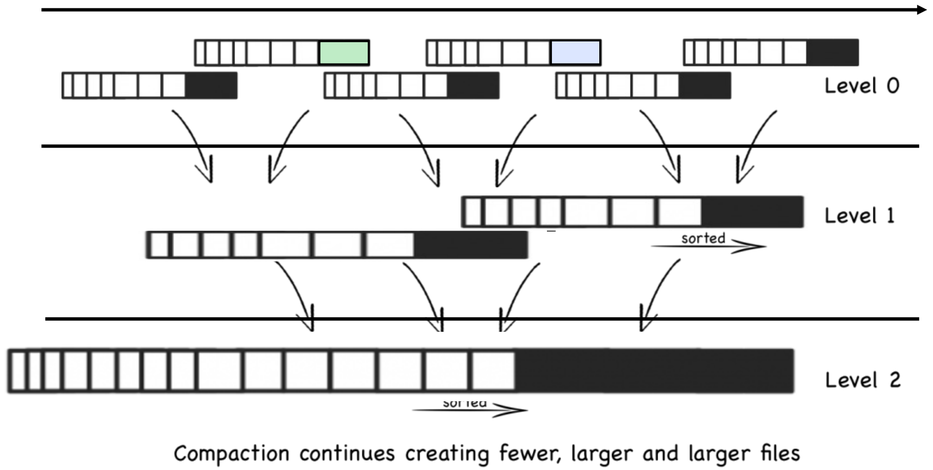

## Leveled compaction

- Read amplification
- Write amplification
- Space amplification



## OLTP

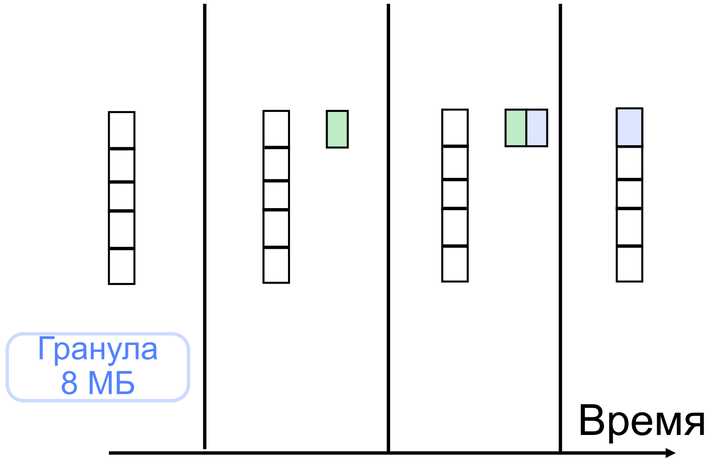
- Большое число случайных чтений-записи
- LSM с size tiered compaction.
  - Read amplification
  - Write amplification
  - Space amplification

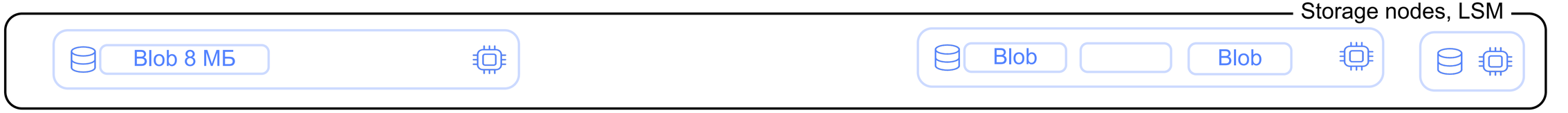


Compaction continues creating fewer, larger and larger files

## OLAP

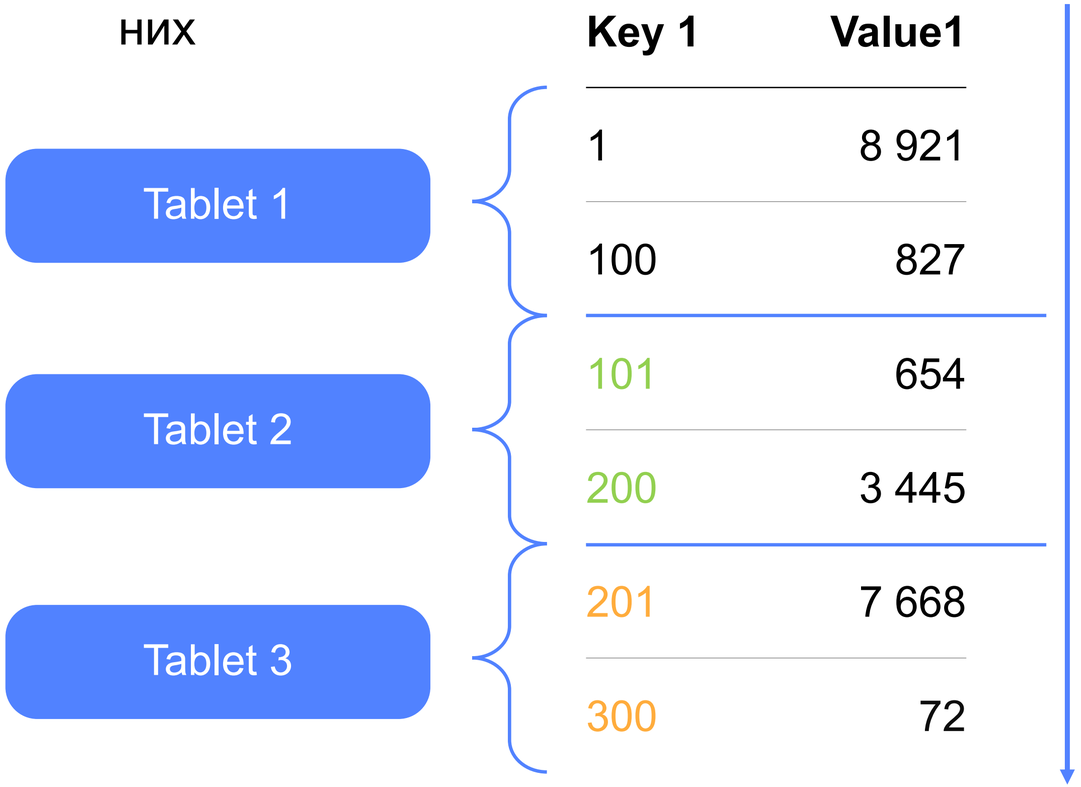
- Частые добавления и редкие изменения
- Неизменяемые данные с наложением изменений поверх
  - Read amplification
  - Write amplification
  - Space amplification



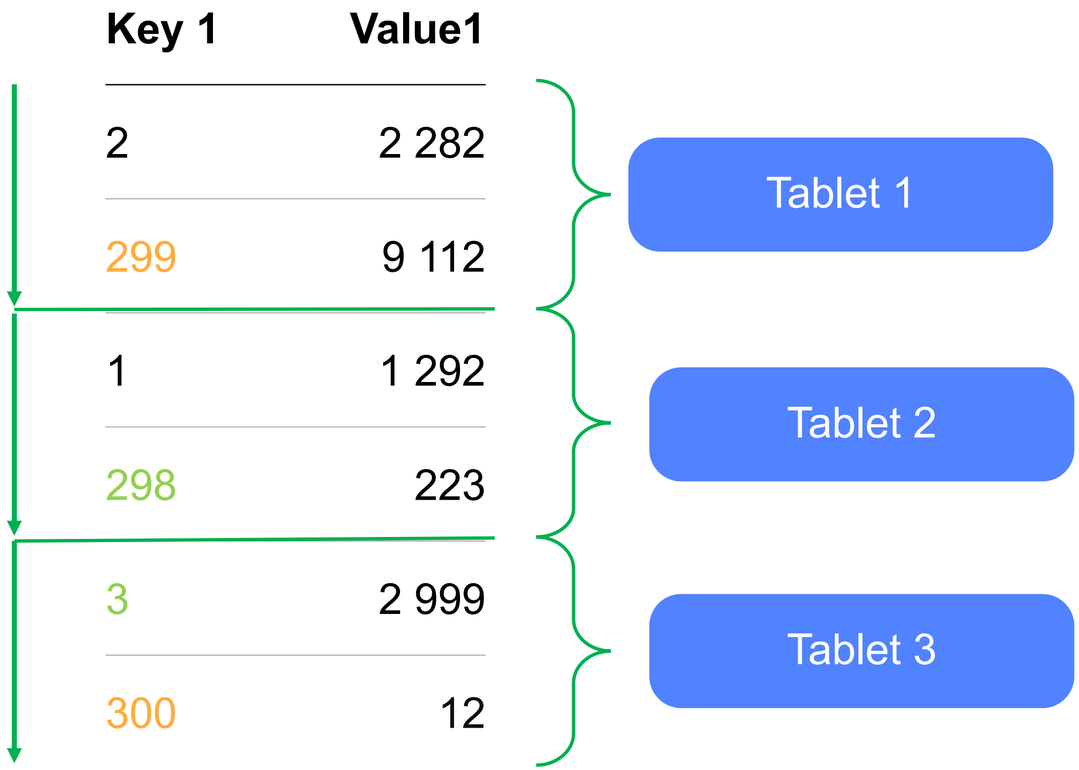
# OLTP

- Range-шардирование по ключу (partition by range)
- Данные группируются по шардам и по ключам в них




# OLAP

- Hash-шардирование по ключу (partition by hash)
- Данные «размазываются» на все хосты




Основная цель – параллельность обработки

1. Организация хранения данных
2. Оптимизация запросов
3. Планирование и управление ресурсами

OLTP структуры хранения оптимизированы под большое число операций изменения данных

OLAP оптимизируется под сканирование больших объёмов операций с отдельной оптимизацией под редкое обновление данных

# Оптимизация запросов

## OLTP

- Миллионы RPS.
- Типичное время выполнения – микросекунды (20000+ RPS в один ключ)
- Минимальное использование LLVM
- Отключение pushdown предикатов на простых запросах

## OLAP

- Типичное время выполнения сотни миллисекунд-десятки секунд
- Спиллинг на диск
- Распределенный Join (аналитический)
- Оптимизация транспорта под большие объёмы данных, zero-copy
- Активное использование LLVM
- Pushdown предикатов
- Векторные вычисления (SIMD)

# Векторные вычисления

- SIMD-инструкции, до 8 операций за такт
- Требуют заранее подготовленных блоков в памяти (Apache Arrow)

# Pushdown предикатов

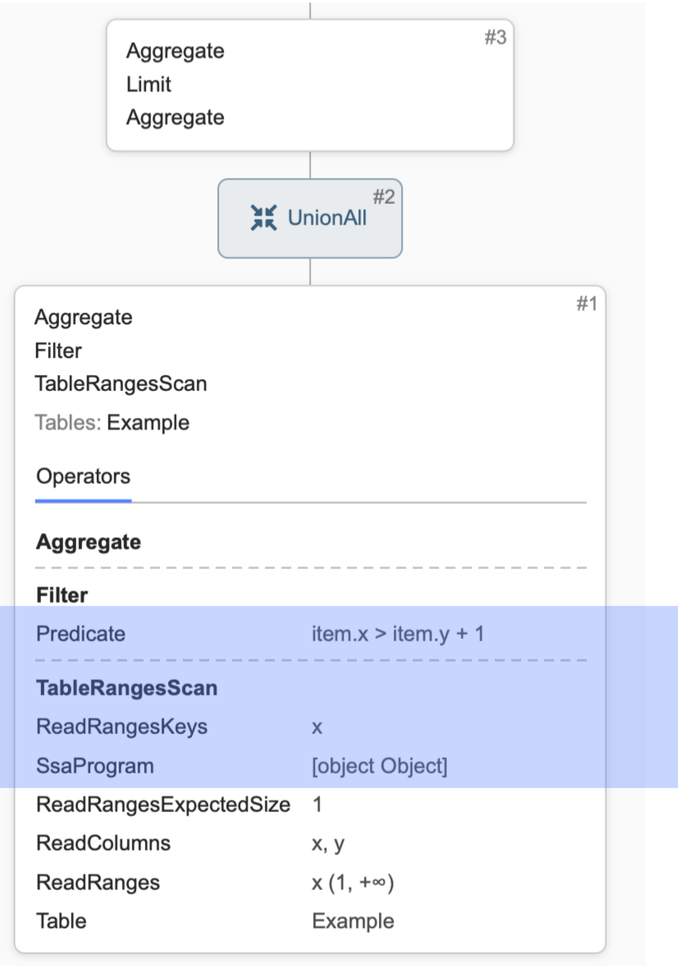
```
SELECT min(x) FROM Example

WHERE x > 1 AND x>y+1
```

## Физический оптимизатор

- Разбивает сложные операции на простые.
  Например, avg разбивается на sum()/count()
- Формирует SSA-программу

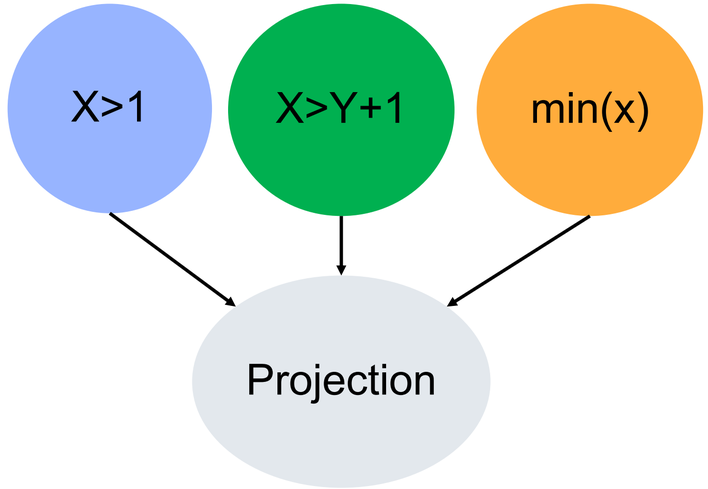
## SSA-программа (single statement assignment)

```
PROJECTION x, y
ASSIGN v1 = x>1
ASSIGN v2 = x>y+1
FILTER BY v1
FILTER BY v2
ASSIGN agg1 = min(x)
PROJECTION agg1
```



# Pushdown фильтров, SSA программа

```
PROJECTION x, y
ASSIGN v1 = x>1
FILTER BY v1
ASSIGN v2 = x>y+1
FILTER BY v2
ASSIGN agg1 = min(x)
PROJECTION agg1
```




| X | Y | v1 |
|---|---|---|
| | | False |
| | | True |
| | | False |

SSA-программа

- Легко параллелизовать (видна зависимость по данным)
- Использование Apache Arrow ядер
- Реестр ядер содержит 100+ оптимизированных ядер для различных типов и задач

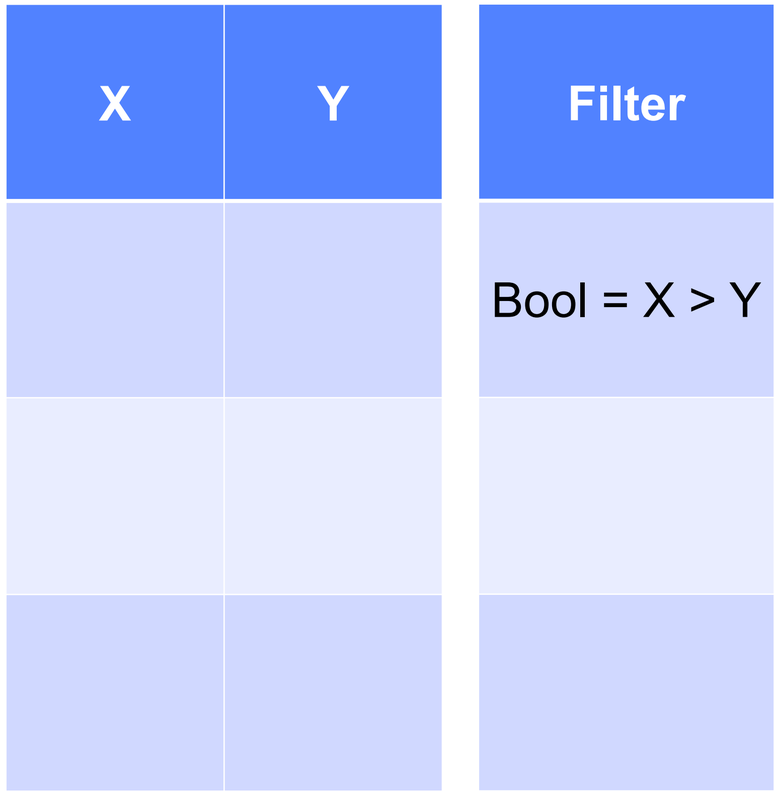
# Виды SSA-выражений

- Вычислить (x*y, x*y+z, x/y, regexp и так далее)
- Filter (bool, как результат вычисления простых выражений)
- Projection (взять часть колонок)
- Limit (взять N строк результата)

| X | Y | Вычислить |
|---|---|---|
| | | x*y+z |
| | | |
| | | |

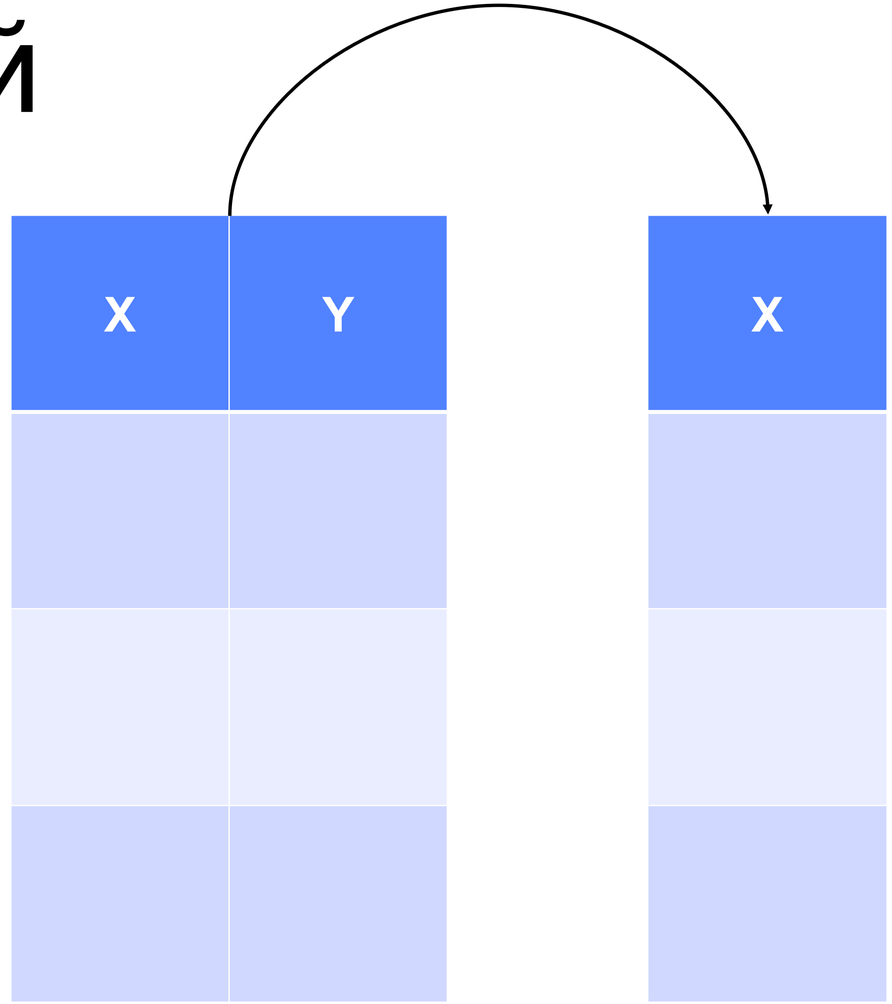
# Виды SSA-выражений

- Вычислить (x*y, x*y+z, x/y, regexp и так далее)
- Filter (bool, как результат вычисления простых выражений)
- Projection (взять часть колонок)
- Limit (взять N строк результата)



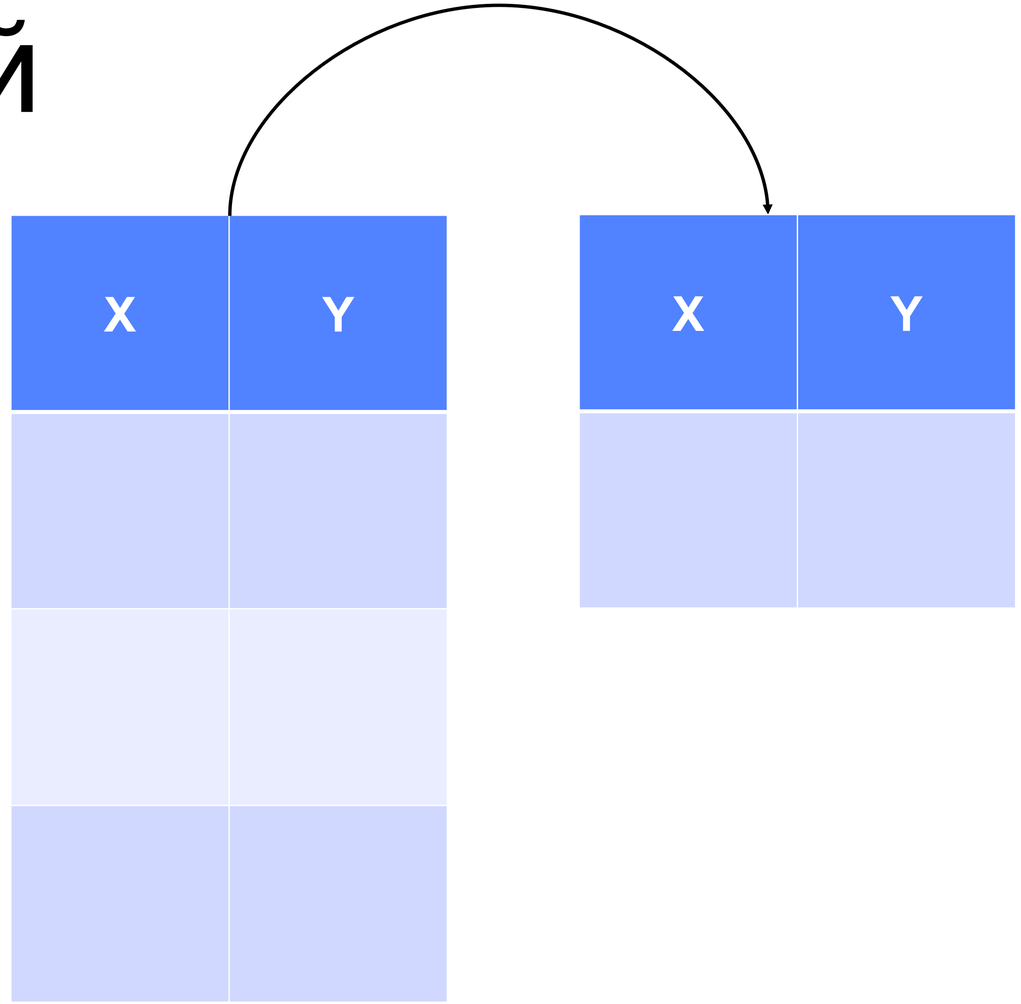
# Виды SSA-выражений

- Вычислить (x*y, x*y+z, x/y, regexp и так далее)
- Filter (bool, как результат вычисления простых выражений)
- Projection (взять часть колонок)
- Limit (взять N строк результата)



# Виды SSA-выражений

- Вычислить (x*y, x*y+z, x/y, regexp и так далее)
- Filter (bool, как результат вычисления простых выражений)
- Projection (взять часть колонок)
- Limit (взять N строк результата)

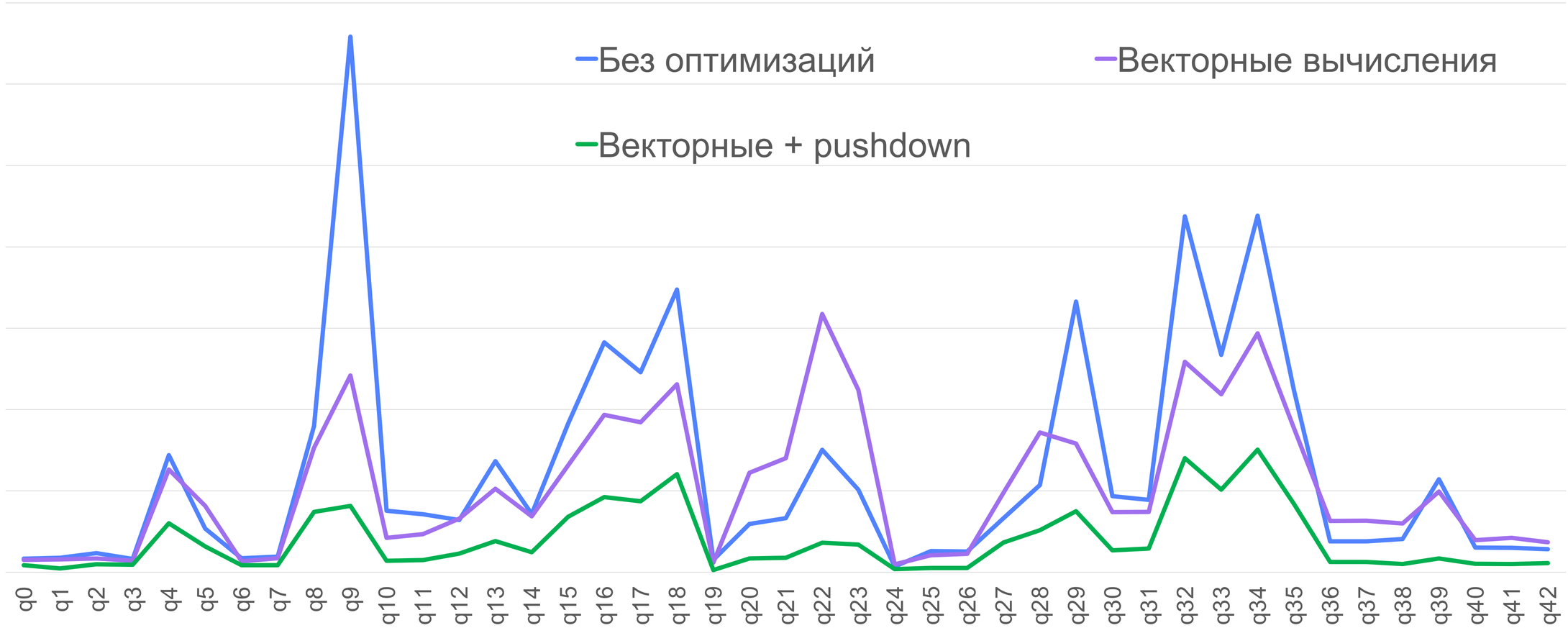


# Частичный pushdown фильтров

- Оценка идемпотентности предикатов (pure functions)
- Оценка логических предикатов
- Оценка возможности выполнить частичный pushdown

```
SELECT * FROM Example
WHERE
    timestamp>CURRENT_TIMESTAMP
OR
    year<2030
```

# Итоги OLAP оптимизаций



1. Организация хранения данных
2. Оптимизация запросов
3. Планирование и управление ресурсами

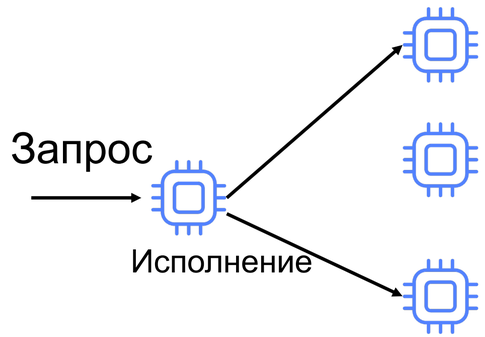
OLTP оптимизируется для скорости, отказ от «лишних» действий

OLAP оптимизируется под обработку больших данных, LLVM, SIMD, распределённые вычисления

# Планирование и управление ресурсами

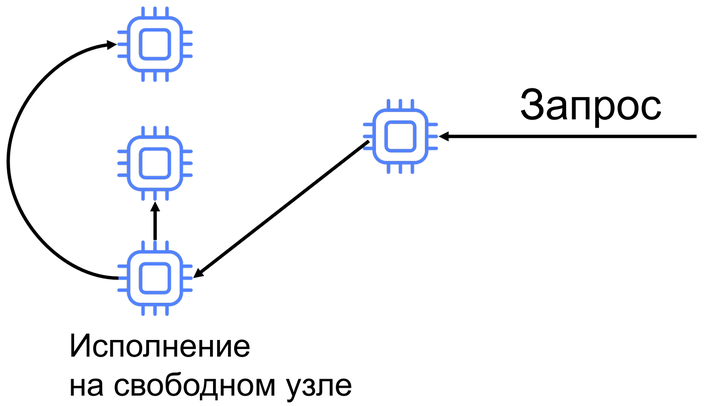
## OLTP

- Запрос исполняется на изначальном узле




## OLAP

- Оценивается количество свободных мощностей в кластере
- Выбираются узлы для исполнения запроса



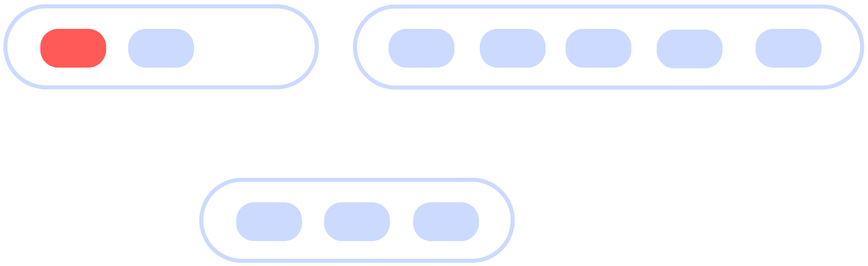
# Планирование и управление ресурсами

Пул ресурсов (таблетки)



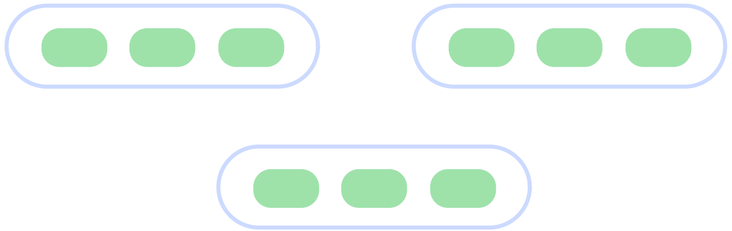
## OLTP

- Балансировка по загрузке CPU/памяти



## OLAP

- Равномерная балансировка



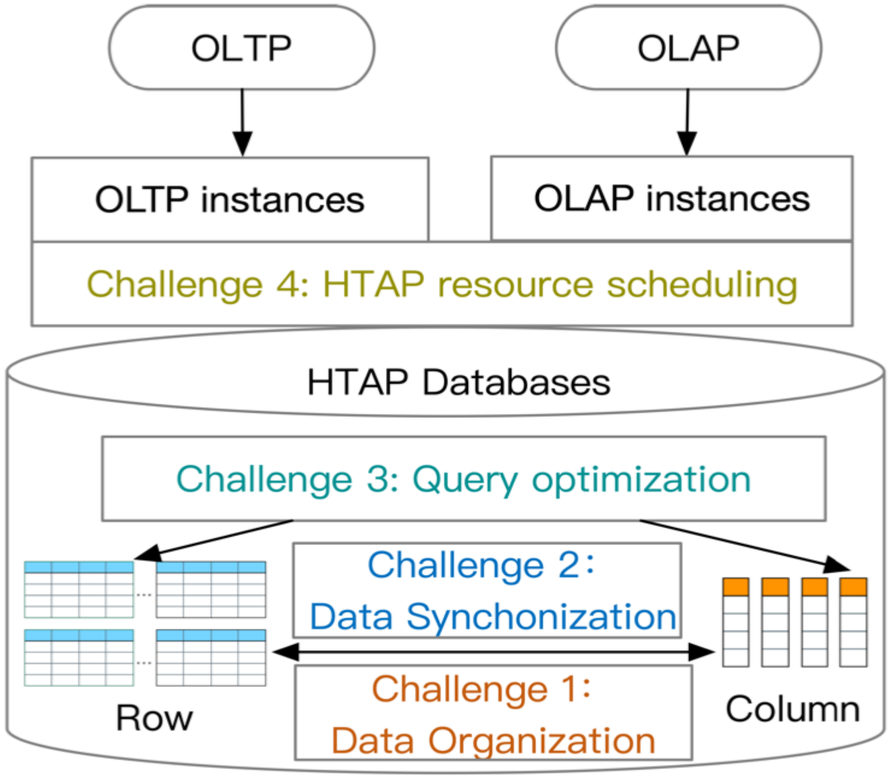
1. Организация хранения данных
2. Оптимизация запросов
3. Планирование и управление ресурсами

OLTP исполняется на первых доступных мощностях

OLAP планирует мощности для выполнения, ограничивает число одновременно выполняющихся запросов. Пытается использовать все доступные ресурсы

# Challenges for HTAP databases

- **Challenge 1 (Data Organization): how to organize the data adaptively for HTAP workloads with high performance and low storage cost.**
- **Challenge 2 (Data Synchronization): how to synchronize the data from the row store to the column store for high throughput and data freshness**
- **Challenge 3 (Query Optimization): how to optimize the query with both row store and column store by exploring the huge plan space.**
- **Challenge 4 (Resource Scheduling): how to schedule the resources for OLTP and OLTP instances effectively for high throughput and data freshness.**



# С чем мы столкнулись

1

Хранить данные по-другому (колоночно)

2

Другое представление в памяти (Apache Arrow)

3

Другая обработка данных (векторный колоночный движок)

4

Практически новая кодовая база

# Что дальше

1

Cost-Based Optimizer

2

Менеджмент ресурсов (взаимовлияние OLTP и OLAP на общих хостах)

3

И, наконец, HTAP

# Спасибо!

**https://github.com/ydb-platform**

**https://t.me/cloud_track**

Алексей Дмитриев,
slonnn@yandex-team.ru