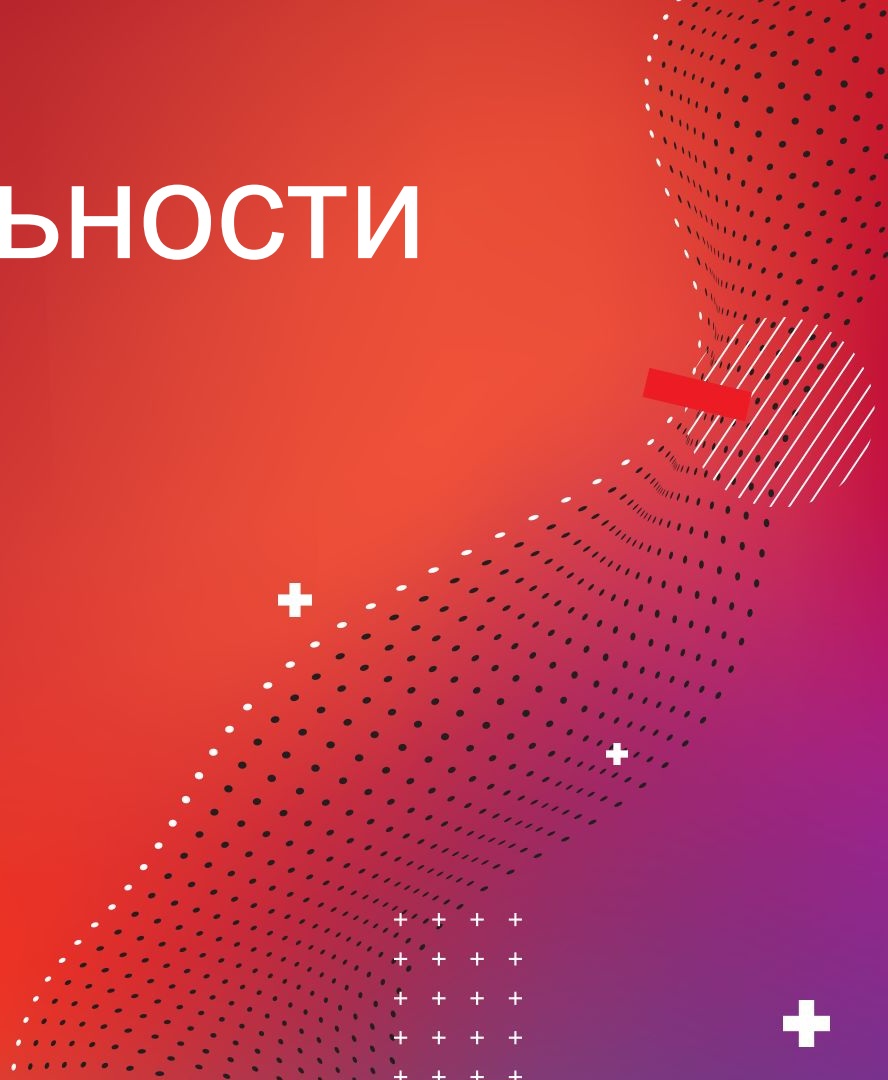


Тесты производительности ClickHouse

Александр Кузьменков
Яндекс



HighLoad++
Весна 2021



Что за ClickHouse?



- Колоночная СУБД для аналитики
- SQL
- Отчёты по сырым большим данным в реальном времени^2
- Базовая технология Яндекс.Метрики
- *Не тормозит*

Почему не тормозит?

- Эффективное хранение данных
 - Колонки
 - Сжатие
- Эффективная обработка
 - Многопоточная, распределённая, специализированные векторные алгоритмы
- Много бенчмарков

Бенчмарки

- Микробенчмарки для конкретного алгоритма
 - Оптимизация агрегатной функции в ClickHouse
- end-to-end
 - На синтетических данных
 - На реальных данных
- Бенчмарки оборудования
 - На моём телефоне

Непрерывное автоматическое тестирование

- Для каждого PR
- Для каждого коммита в мастер/релизную ветку
- Изменения за апрель ([интерактив здесь](#)):

authors	prs
44	401

tests	queries
215	2640



Performance — 3 faster, 72 unstable

[Details](#)

ev

ClickHouse performance comparison

Tested Commits[?]

Old	New
commit dd31634ba44168104d9e23b218148a2dad01b41e (origin/master) Merge: fa5cea7a0 7b085abdb Author: alexey-milovidov Date: Fri Apr 30 13:15:52 2021 +0300	commit 7467c5e3cf8ee702e75e80942ac9eba4402df69a Author: Maksim Kita Date: Fri Apr 30 13:56:56 2021 +0300
Merge pull request #23746 from oxidecomputer/master	Function default implementation for nulls small optimization
Adds support for building on Solaris-derived systems	Real tested commit is: commit 18d3679d808a68a8621d9c22bc7715b82d6ae1a5 (HEAD -> master, pr) Merge: dd31634ba 7467c5e3c Author: Maksim Kita Date: Fri Apr 30 14:01:14 2021 +0300
	Merge 7467c5e3cf8ee702e75e80942ac9eba4402df69a into dd31634ba44168104d9e23b218148a2dad01b41e

Changes in Performance[?]

Old, s	New, s	Ratio of speedup (-) or slowdown (+)	Relative difference (new - old) / old	p < 0.01 threshold	Test	# Query
0.317	0.140	-2.264x	-0.559	0.558	logical_functions_medium	18 SELECT count() FROM test_logical_functions_4_1_Nullable_UInt8 WHERE NOT ignore(xor(x1,x2,x3,x4))
0.163	0.075	-2.166x	-0.539	0.538	logical_functions_small	9 SELECT count() FROM (SELECT toNullable(materialize(1)) AS x1, toNullable(materialize(1)) AS x2 FROM
0.262	0.193	-1.359x	-0.265	0.264	logical_functions_medium	19 SELECT count() FROM test_logical_functions_4_1_Nullable_Mixed WHERE NOT ignore(xor(x1,x2,x3,x4))

Что мерить

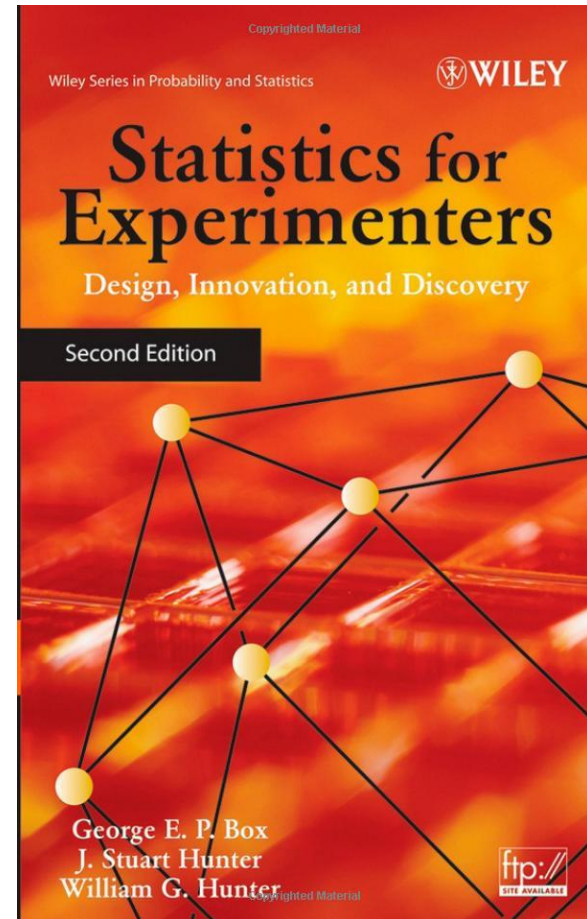
- Время выполнения запроса
- Несколько раз
- Среднее не видит странностей
- min/max не устойчивы к выбросам
- Мерим медиану

С чем сравнивать

- Модельное распределение
- Исторические данные
 - Разное железо
 - Разное окружение
- Предыдущая версия сервера, запущенная на том же сервере

Как сравнивать

Box, Hunter, Hunter, "Statistics for experimenters", p. 78: "A Randomized Design Used in the Comparison of Standard and Modified Fertilizer Mixtures for Tomato Plants."



Randomization distribution

- Что даёт?
 - Разницу, которую можно увидеть даже сравнивая сервер сам с собой (“S”).
 - $p < 0.01 \iff 99$ перцентиль.
 - $2000 * 1\% = 20$ false positive на прогон.
 - Для любой метрики (напр., потребление памяти)
- Как интерпретировать
 - $D < 5\%$ — неинтересно
 - $D < S$ — статистически незначимо
 - $D > S$ — производительность изменилась
 - $S > 5\%$ — плохой запрос (низкая точность)

Изменения производительности

- Поменяли код в этом месте
- Статистический ложноположительный результат
- Поменялась сборка
 - граничные эффекты в компиляторе
 - разное расположение функций в бинарнике
 - [BOLT](#), [Propeller](#)

Встроенный профайлер

```
set query_profiler_real_time_period_ns = 100000000;

clickhouse-client -q "SELECT
    arrayStringConcat(
        arrayMap(
            x -> concat(splitByChar('/', addressToLine(x))[-1],
                '#', demangle(addressToSymbol(x))),
            trace),
        ';' ) AS stack,
    count(*) AS samples
FROM system.trace_log
WHERE trace_type = 'Real'
    AND query_id = '4aac5305-b27f-4a5a-91c3-61c0cf52ec2a'
GROUP BY trace" \
| flamegraph.pl
```

Встроенный профайлер (2)

Reset Zoom

Flame Graph

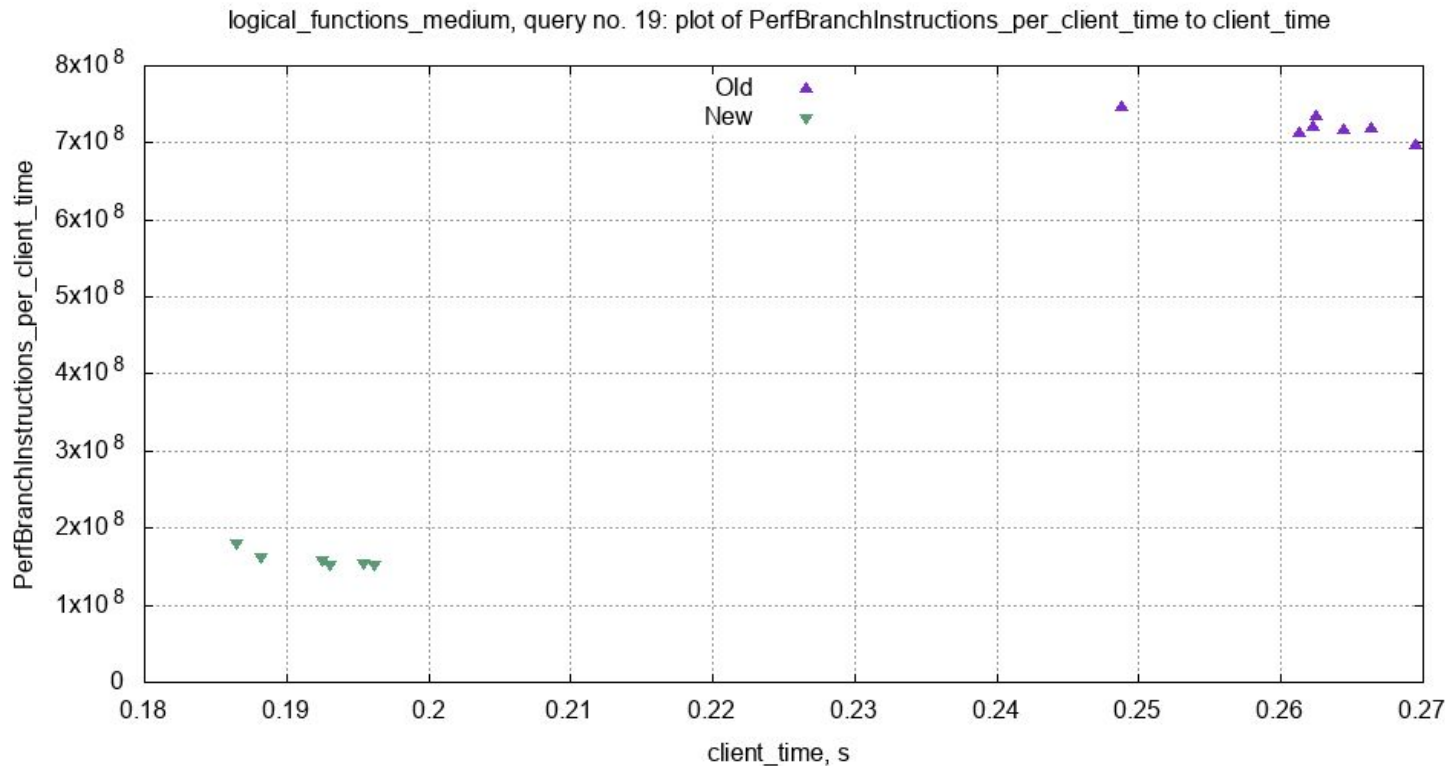
Search

ic

```
c.. clickhouse#DB::(anonymous namespace)::convertAnyColumnToBool(DB::IColumn const*, DB::PODArray<char8_t, 4096ul, Allocator<false, false>, 15ul,.. click..
clickhouse#DB::FunctionsLogicalDetail::FunctionAnyArityLogical<DB::FunctionsLogicalDetail::XorImpl, DB::NameXor>::executeImpl(std::__1::vector<DB::Colu..
clickhouse#DB::DefaultExecutable::execute(std::__1::vector<DB::ColumnWithTypeAndName, std::__1::allocator<DB::ColumnWithTypeAndName> > const&, std:...
clickhouse#DB::ExecutableFunctionAdaptor::executeWithoutLowCardinalityColumns(std::__1::vector<DB::ColumnWithTypeAndName, std::__1::allocator<DB::Co.. clickhouse#DB..
clickhouse#DB::ExecutableFunctionAdaptor::defaultImplementationForNulls(std::__1::vector<DB::ColumnWithTypeAndName, std::__1::allocator<DB::ColumnWithTypeAndName> >..
clickhouse#DB::ExecutableFunctionAdaptor::executeWithoutLowCardinalityColumns(std::__1::vector<DB::ColumnWithTypeAndName, std::__1::allocator<DB::ColumnWithTypeAnd..
clickhouse#DB::ExecutableFunctionAdaptor::execute(std::__1::vector<DB::ColumnWithTypeAndName, std::__1::allocator<DB::ColumnWithTypeAndName> > const&, std::__1::sha..
clickhouse#DB::ExpressionActions::execute(DB::Block&, unsigned long&, bool) const
clickhouse#DB::FilterTransform::transform(DB::Chunk&)
clickhouse#DB::ISimpleTransform::transform(DB::Chunk&, DB::Chunk&)
clickhouse#DB::ISimpleTransform::work()
clickhouse#void std::__1::__function::__policy_invoker<void ()>::__call_impl<std::__1::__function::__default_alloc_func<DB::PipelineExecutor::addJob(DB::ExecutingGraph::Node*)>::__$...
clickhouse#DB::PipelineExecutor::executeStepImpl(unsigned long, unsigned long, std::__1::atomic<bool>*)
clickhouse#DB::PipelineExecutor::executeImpl(unsigned long)
clickhouse#DB::PipelineExecutor::execute(unsigned long)
clickhouse#void std::__1::__function::__policy_invoker<void ()>::__call_impl<std::__1::__function::__default_alloc_func<ThreadFromGlobalPool::ThreadFromGlobalPool<DB::PullingAsyncP..
clickhouse#ThreadPoolImpl<std::__1::thread>::worker(std::__1::__list_iterator<std::__1::thread, void*>)
clickhouse#void* std::__1::__thread_proxy<std::__1::tuple<std::__1::unique_ptr<std::__1::__thread_struct>, std::__1::default_delete<std::__1::__thread_struct> >, void ThreadPoolImp..
libpthread-2.27.so#start_thread
clone.S (filtered by script)
all
```

Аппаратные метрики процессора

- `man perf-stat(1) / perf_event_open(2)`
- `set metrics_perf_events_enabled = 1`



Что ещё может пойти не так?

“Все зелёные тесты похожи друг на друга, каждый красный тест красен по-своему.”

— А. Эйнштейн

- Тест неправильно написан
 - Ошибка описании теста
 - Ошибка в запросе
 - Слишком медленный
 - Слишком быстрый
- Ошибка среды
 - Кончилась память/место
 - ClickHouse упал/не запустился
- Изменилась производительность
 - Из-за наших изменений в коде (единственный true positive)
 - Статистический false positive
 - Поменялась сборка
 - граничные эффекты в компиляторе
 - разное расположение функций в бинарнике ([BOLT](#), [Propeller](#))
- Нестабильное время выполнения

Нестабильные запросы

- Неправильно написан запрос
 - Слишком короткий, мерит шум
 - `system stop merges / optimize table final`
 - Таблица `Buffer` не того размера
- Неправильно написан код
- Внешние факторы

Нестабильные запросы (NUMA)

```
$ numactl --hardware
```

```
available: 2 nodes (0-1)
```

```
node 0 cpus: 0 ... 41
```

```
node 0 size: 257844 MB
```

```
node 0 free: 99115 MB
```

```
node 1 cpus: 14 ... 55
```

```
node 1 size: 258043 MB
```

```
node 1 free: 88101 MB
```

```
node distances:
```

```
node    0    1
```

```
  0:   10   21
```

```
  1:   21   10
```

Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz

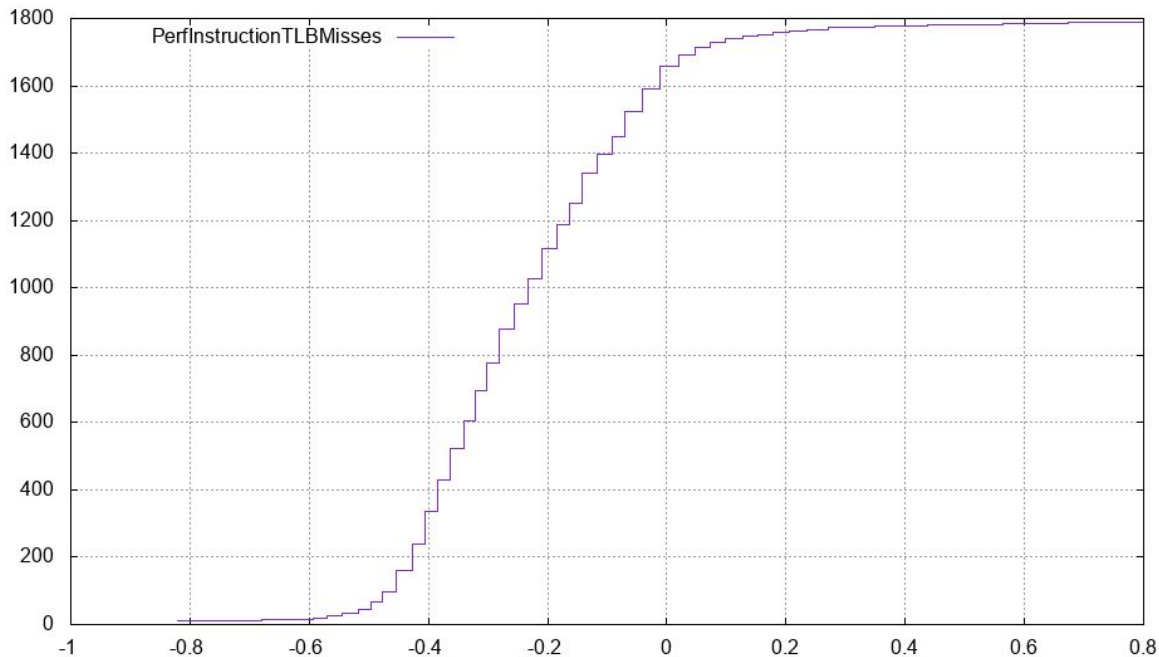
```
$ numactl --cpunodebind=0
```

```
--membind=0
```

400 -> 100 нестабильных запросов.

Нестабильные запросы (iTLB)

- Копирование бинарника в huge pages (https://youtu.be/icfec1r_2q8?t=1587)
The "worst" code I've ever written. — A. Milovidov



Потребительские свойства

Чего хочется от тестов:

- Заканчиваться в обозримое время
 - 3ч для ПР
- Минимум false positives
 - игнор <5%
 - статистическая значимость
 - ожидаем до трёх false positive => можно их игнорировать
 - нестабильные тесты размечены вручную
- Результат можно интерпретировать
 - Инструменты интроспекции
 - Подробная инструкция
- Избежать вырождения в систему демонстрации зелёной галки

Как...

Пользователь автотестов производительности

Я хочу...

Видеть зелёную галку

Чтобы...

Спокойно помержить свой пулреквест

— *тесты производительности по версии @ShitUserStory*

С чем ещё можно сравнивать

- Среднее геометрическое изменений по всем тестам
 - Больше 1% — что-то действительно поменялось
- Старые релизы
 - 20.8 -> 21.6 — в среднем 10% ускорения
- Исторические данные

queries	commits	months
1.09 billion	7.24 thousand	15.4

on disk	uncompressed data
6.27 GiB	31.32 GiB

WITH 0.05 AS s

```
SELECT old_sha, new_sha, event_time, message, old_value AS `old server`, new_value AS `new server`,  
before AS `prev 11 runs`, after AS `next 11 runs`, diff AS `diff, ratio`, stat_threshold_historical  
AS `stat threshold, ratio, historical`, stat_threshold AS `stat threshold, ratio, per-run`,  
cpu_model, query_display_name
```

FROM

```
(SELECT *, run_attributes_v1.value AS cpu_model,  
        median(old_value) OVER (PARTITION BY run_attributes_v1.value, test, query_index,  
query_display_name ORDER BY event_date ASC ROWS BETWEEN 11 PRECEDING AND CURRENT ROW) AS before,  
        median(new_value) OVER (PARTITION BY run_attributes_v1.value, test, query_index,  
query_display_name ORDER BY event_date ASC ROWS BETWEEN CURRENT ROW AND 11 FOLLOWING) AS after,  
        quantileExact(0.95)(abs(diff)) OVER (PARTITION BY run_attributes_v1.value, test, query_index,  
query_display_name ORDER BY event_date ASC ROWS BETWEEN 37 PRECEDING AND CURRENT ROW) AS  
stat_threshold_historical
```

```
FROM query_metrics_v2
```

```
LEFT JOIN run_attributes_v1 USING (old_sha, new_sha)
```

```
WHERE (attribute = 'lscpu-model-name') AND (metric = 'client_time') AND (pr_number = 0)
```

```
AND (test = 'logical_functions_medium') AND (query_index = 19)
```

) AS t

```
ANY LEFT JOIN `gh-data`.commits ON new_sha = sha
```

```
WHERE (((abs(after - before) / if(after > before, after, before)) AS step_height) >= greatest(s,  
stat_threshold_historical))
```

```
AND (abs(diff) >= greatest(stat_threshold, stat_threshold_historical, s))
```

```
AND (abs(diff) >= (0.7 * step_height))
```

Row 3:

old_sha: 1dd57645c28f03e66347abffd13266a9c6ad2750
new_sha: eeae539a9f02f00fa8f4d6ae2daf45ea8320818d
event_time: 2021-04-30 21:50:24
message: Merge pull request #23799 from
kitaisreal/function-default-implementation-for-nulls-small-optimization

Function default implementation for nulls small optimization

old server: 0.2608
new server: 0.1985
prev 11 runs: 0.2603999972343445
next 11 runs: 0.19845000654459
diff, ratio: -0.239
stat threshold, ratio, historical: 0.074
stat threshold, ratio, per-run: 0.212
cpu_model: Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00GHz
query_display_name: SELECT count() FROM
test_logical_functions._4_1_Nullable_Mixed WHERE NOT ignore(xor(x1,x2,x3,x4))

Интересные факты

497 # This is a lateral join in bash... please forgive me.

— A. Kuzmenkov, “compare.sh”

- Вся обработка на clickhouse-local
 - а обвязка — на bash
 - и на python (mymarilyn/clickhouse_driver)
 - ловили новые баги: в профайлере, в джоинах, в нативном протоколе и т.д.
- Почти самая тяжёлая проверка в нашем CI
 - (после сборок)

Всё.

Александр Кузьменков

github.com/akuzm

t.me/akuzm

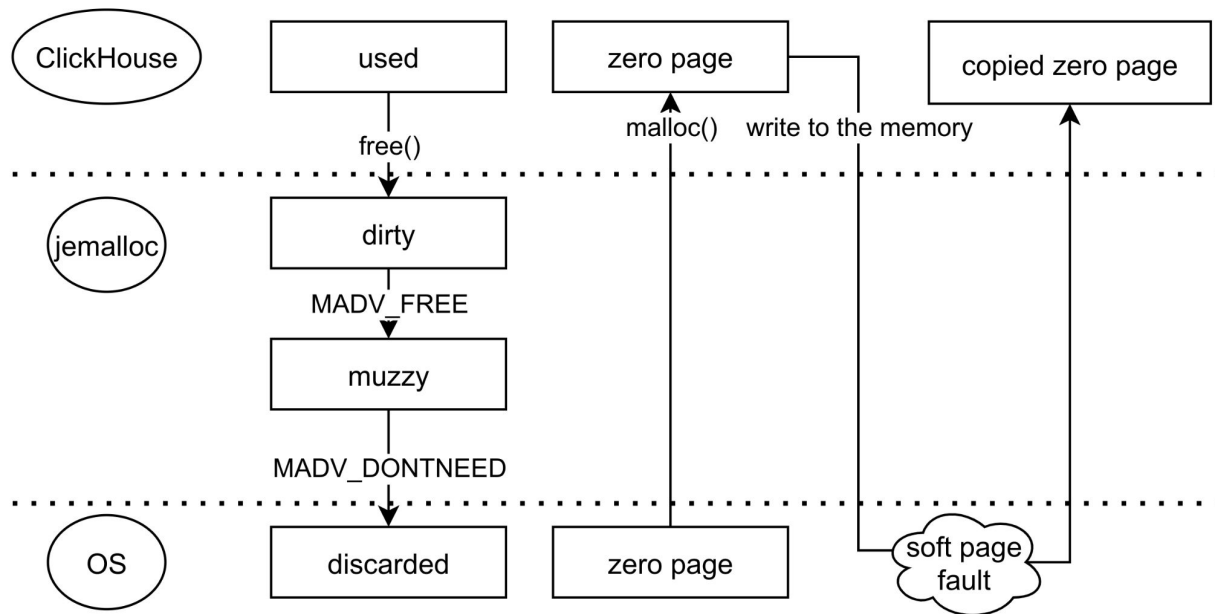
akuzm@yandex-team.ru

Нестабильные запросы (jemalloc)

- soft page faults
- jemalloc two-phase purging

Что поменяли:

- 1) включили MADV_FREE
- 2) muzzy_decay_ms = 10s



Нестабильные запросы (jemalloc) (2)

