

ElasticSearch Inside Out

Что мы знаем?
Знаем ли мы хоть что-нибудь?
Давайте это выясним!



...или история о том, как
завафлить хороший доклад



Argumentum ad verecundiam

GRUBHUBTM

О чём это я?

Давайте приоткроем капот и посмотрим на
несколько фич, чтобы хотя бы издалека
представлять, как оно работает

Оглавление

- Постановка проблемы
- Инструментарий
- Wildcard-запросы
- Scoring
- ~~OR запросы~~
- ~~Полигонные запросы~~
- Scripting
- Collapsing
- ~~Max clause count~~
- DFS Query-Then-Fetch

Проблема

Проблема



Проблема

“

- Маслята, я тут выбираю из MySQL пару миллионов записей с семью джойнами, стал агрегировать по количеству комментариев, и что-то он тормозить стал.

Я думаю поставить Redis, что скажете?

”

Решение

- Используй эластик / solr / sphinx

Проблема

“

- Маслята, мы потратили девять месяцев, три миллиона на консультирование, вместе с обедами нам доставляют антидепрессанты, а инструктаж по увеличению хипа занял неделю. Вчера выкатили в прод, и оно еще сильней тормозит, пока не свалится в ООМ.

Что делать?

”

Решение

- Твоя проблема, ты и разбирайся)))

Проблема

проблема

importance

Проблема

Там внутри вроде есть инвертированные
индексы

Никто не знает, что там есть кроме
инвертированных индексов

Три человека отправились на разведку, но
так и не вернулись

Проблема

“

- Мы сделали для вас фичу X, но только вы ее не используйте, пожалуйста, это плохо для производительности.

”

Насколько плохо? Где теряется скорость? Я могу это использовать, если очень надо? А если у меня возвращается только пять результатов?

Проблема

ElasticSearch – классический черный ящик.

Все знают про анализ и инвертированные индексы для текста и простых атрибутов.

Никто не знает, что на самом деле происходит внутри.

Никто

И поэтому мы здесь

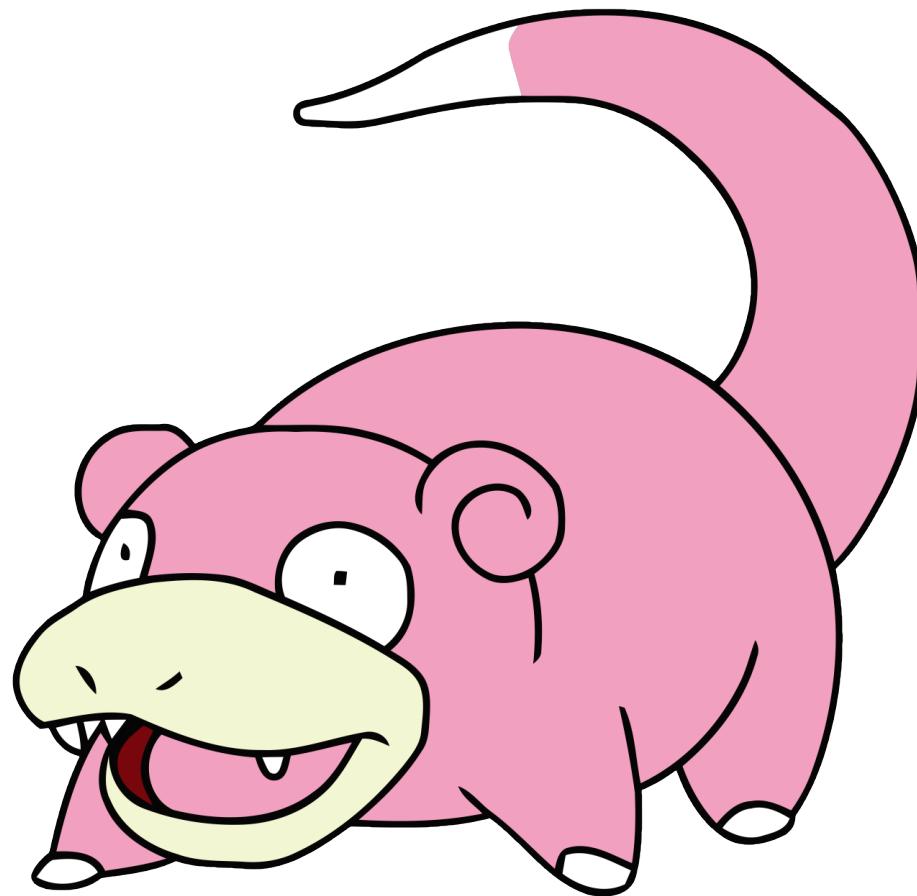
Поехали!

*шестьдесят лет считай прошло, могли бы и доехать уже куда-нибудь

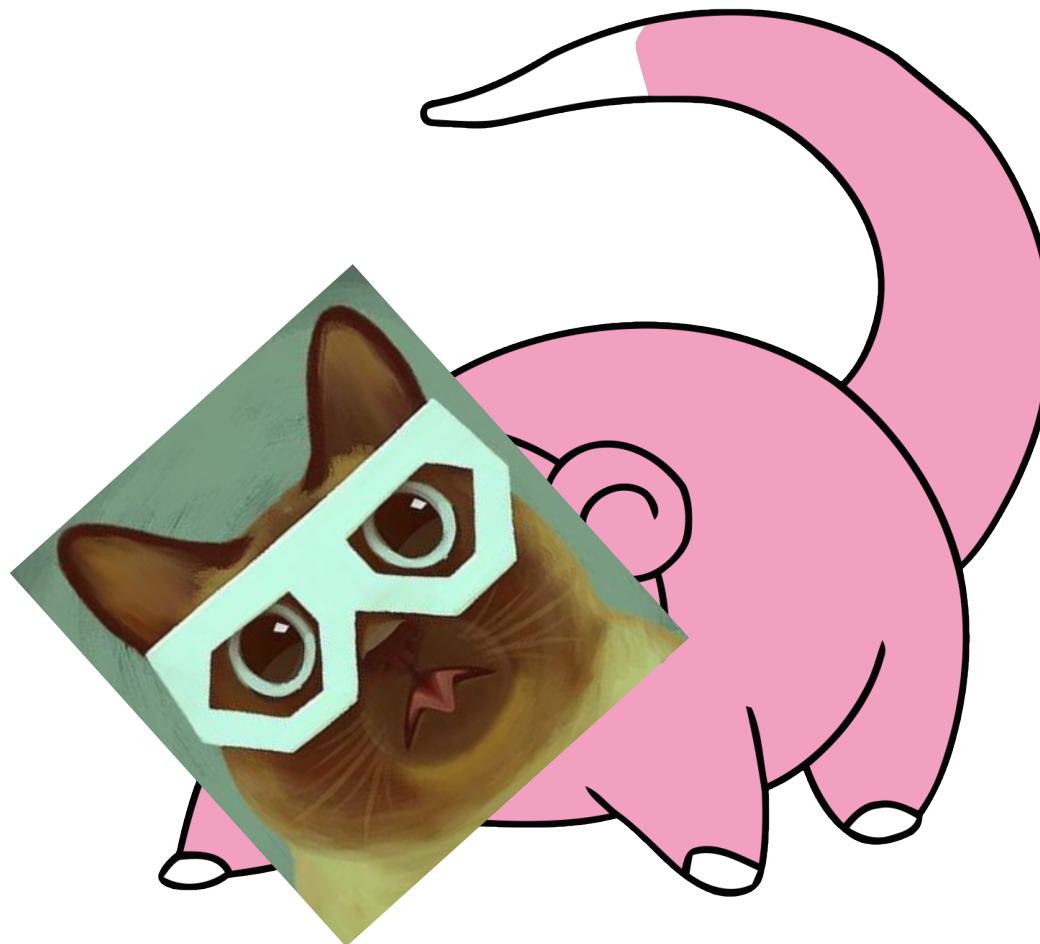
Шкала слоупока

Мерять проблему мы будем слоупоками, не
более десяти за раз

Шкала слоупока



Шкала слоупока



Тулчейн

- 3x Hetzner Cloud / CX51
- Кластер из трех нод, ~~одна мастером
пологена~~
- EsRally, нагрузка только на чтение, 3x CX31
- Датасет wikiquote
- Perf, Java Async Profiler, FlameGraphs
- Red bull

JVM Tuning

-XX:+UseTransparentHugePages

Скорее всего ничего не сделает, но
почему бы и нет

-XX:-TieredCompilation

У меня доклад уже завтра, в смысле я
должен ждать пока JVM прогреется

Порядок проведения тестов

- Каждый тест выполняется пять минут с трех нод
- Во время проведения теста эластик на одной из нод профилируется профайлером
- Смотрим циферки и делаем вид, что что-то понимаем

Кстати, нормальные люди так не делают

- Тестовые датасеты должны быть гораздо сложнее
- Переход на C2 без обычной профилировки вряд ли эффективен
- ТНР вообще вряд ли чего-то дадут, мне просто хотелось уж где-нибудь их включить
- В реальной жизни мы не только читаем, но и обновляем
- Гипотезы выдвигаются, но не проверяются
- Я где-то точно налажал, говорю вам

Правильный порядок проведения тестов

- Удаление всех индексов
- Загрузка датасета по новой
- sync; echo 1 > /proc/sys/vm/drop_caches
- * скорее всего никак не влияет, но парни из старших классов так делают
- _optimize?max_num_segments=1
- * потому что иначе могут быть различные исходные условия
- Прогрев
- Запуск запросов на чтение
- Параллельное обновление датасета

...но у меня не было времени :(

Кстати, нормальные люди
так не делают

Но ведь нам же просто
интересно посмотреть
примерные цифры, верно?

Wildcard-запросы

Wildcard-запросы

“

- ...Note that this query can be slow, as it needs to iterate over many terms. In order to prevent **extremely slow** wildcard queries, a wildcard term should not start with one of the wildcards * or ?

”

Wildcard-запросы

Насколько *extremely slow?*

Wildcard-запросы

- 1.query: term = prophecy
- 2.query: term = ?rophecy
- 3.query: term = ???hecy
- 4.query: term = prop????
- 5.query: term = prop*
- 6.query: term = *hecy

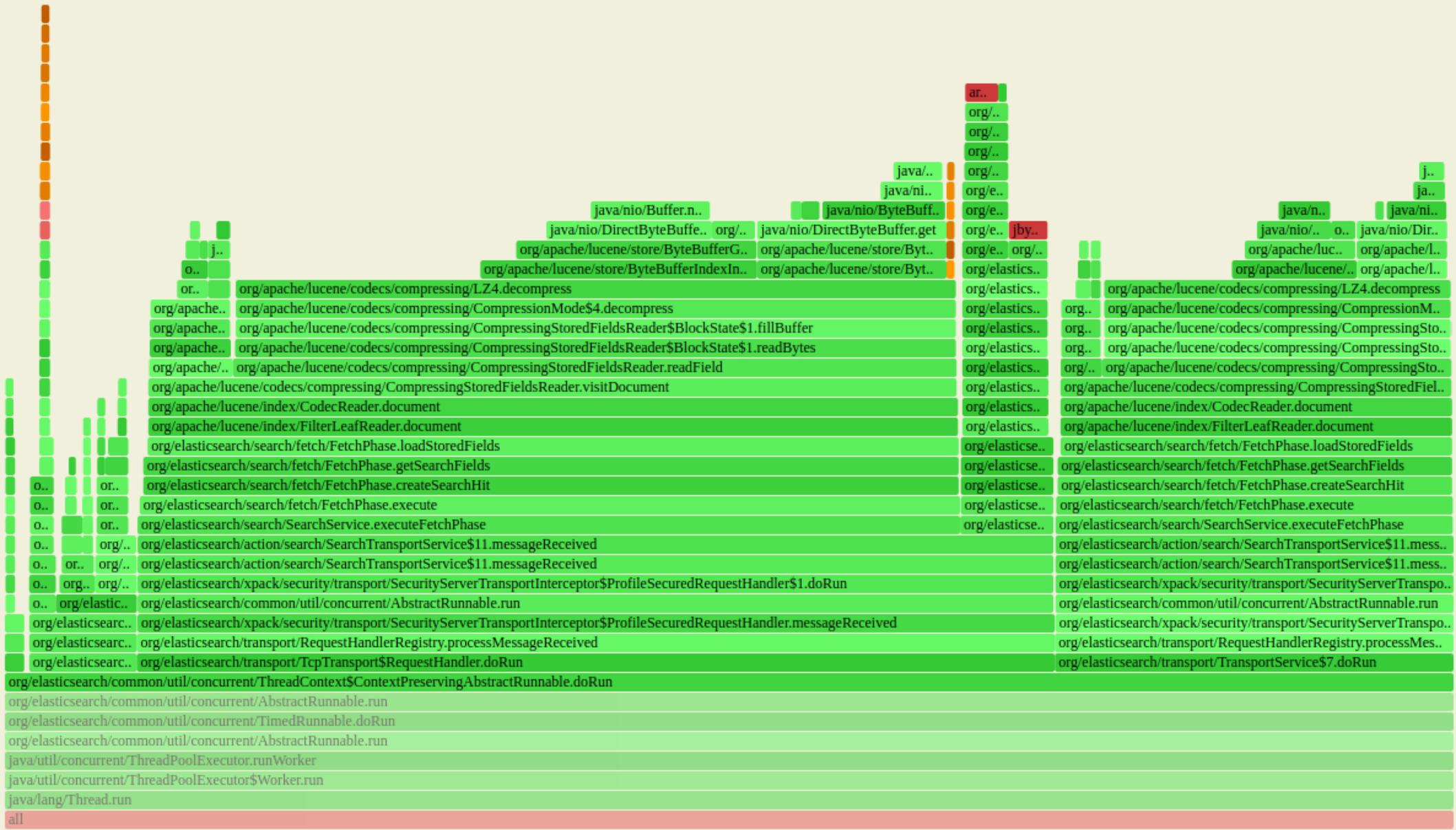
Wildcard-запросы

1. Prophecy: ~45rps
2. ?rophecy: ~45rps
- 3.????hecy: ~60rps
4. prop????: ~105rps
5. prop*: ~120rps
6. *hecy: ~45rps

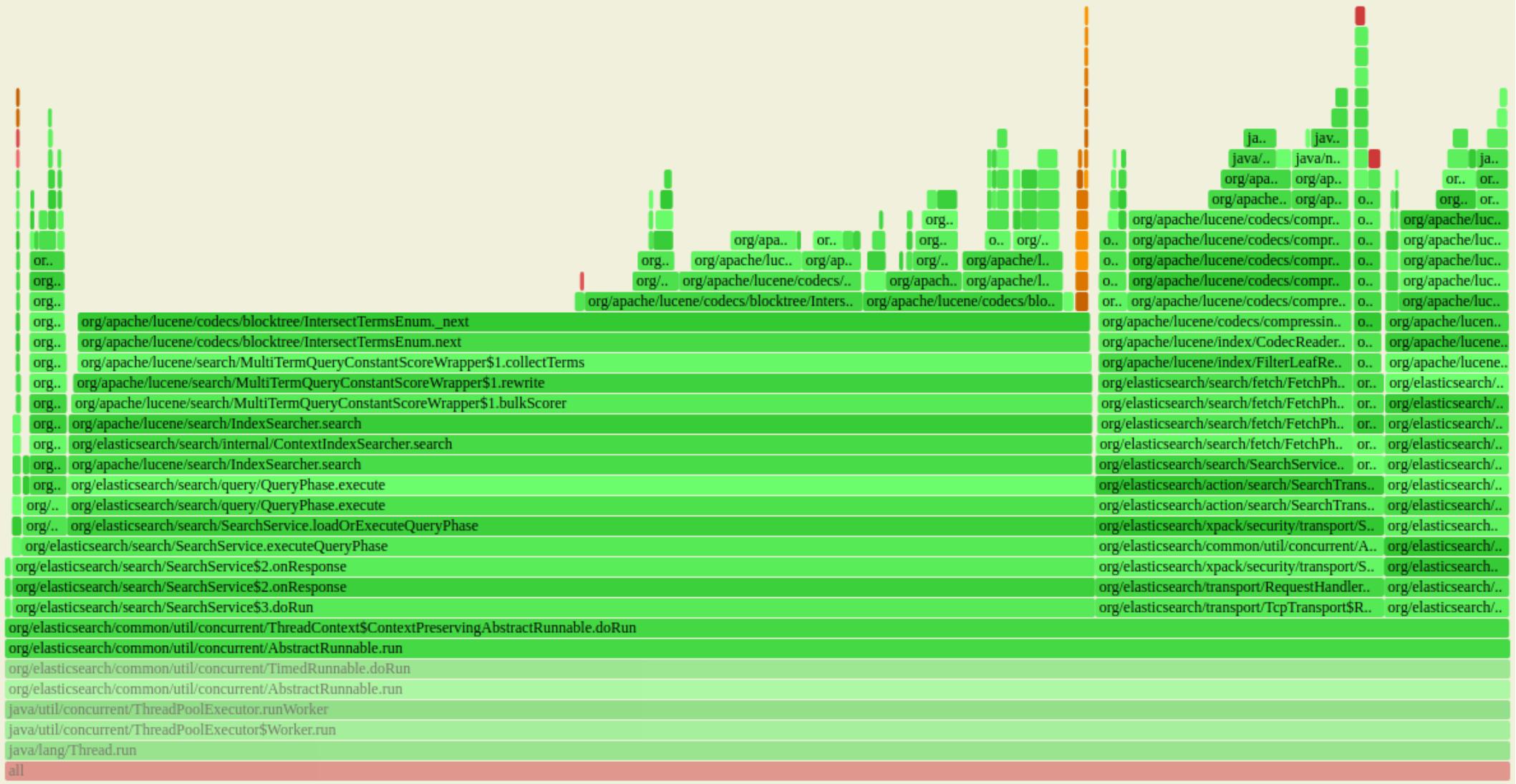
Wildcard-запросы

...кто-то налажал с тестом, так
что давайте лучше заглянем в
флеймграфы

Baseline



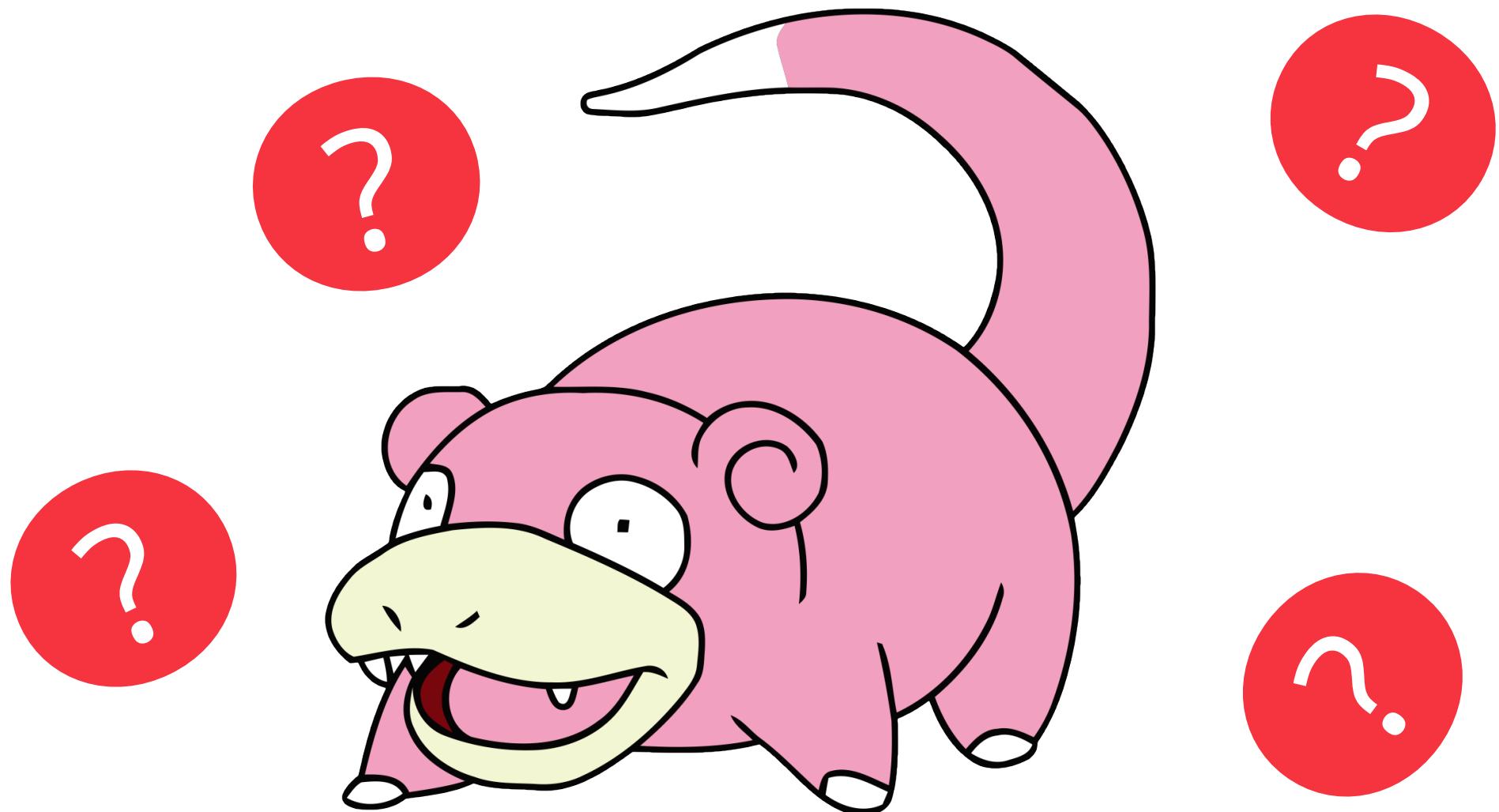
Wildcard prefix



Что происходит?

- Запрос превращается в конечный автомат
- По конечному автомату ищутся все подходящие трёмы
- Конечный автомат может выделить общий префикс для облегчения поиска, чтобы предоставить возможность внешнему коду фильтровать термы
- Но только если этот префикс есть

Оценка



Scoring

Scoring

- Разделение query context / filter context:
скор считается только в query
- Разделение query context / filter context:
скор считается только в query
- По умолчанию скор не возвращается,
если нет сортировки `_score` или не
включен `track_scores`

Если `score` отключают, значит, это кому-то нужно?

Scoring

1.filter: text = queen

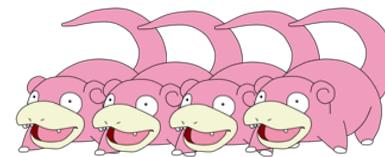
2.filter: text = queen, sort: _score

3.filter: text = queen, track_scores

Scoring

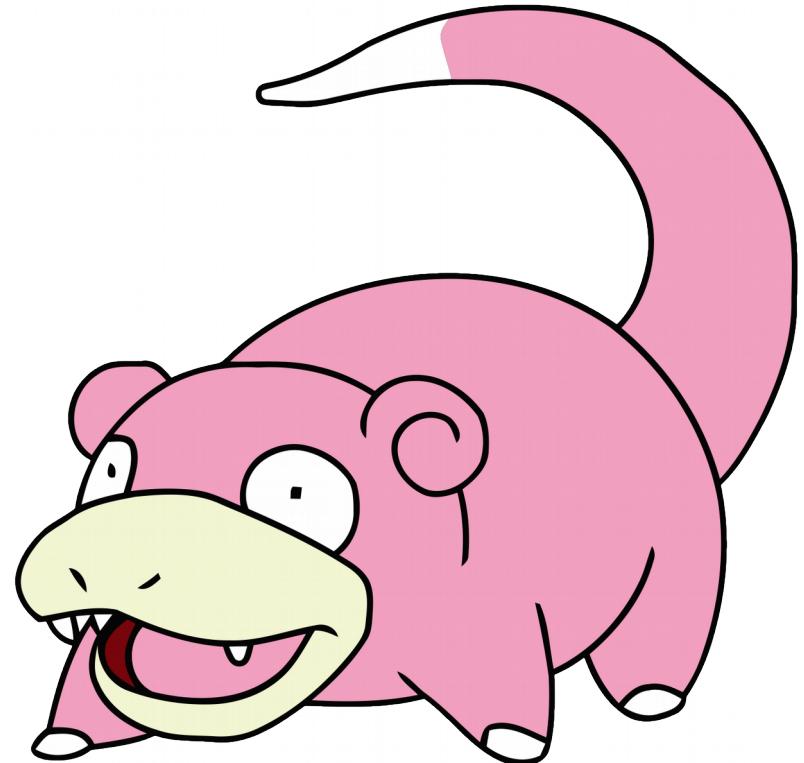
- 1.Filter/baseline: ~150rps
- 2.Filter/sort: ~105rps
- 3.Filter/track_scores: ~105rps

Оценка



4 / 10

*Call Nikola Tesla, I
have an idea of
electromobile*



Ог-запросы

OR-запросы

- Умный запрос: берем самый мелкий подзапрос и все остальные подзапросы считаем внутри него
- Or-запрос: либо A, либо B))))
- Просто A или просто B легко посчитать, потому что каждый из них - это отдельный индекс, $A \cup B$ - уже больно

OR vs UNION

- SQL: `SELECT ... WHERE x = A OR x = B`
=>
`... WHERE x = A UNION ... WHERE X = B`
- ElasticSearch: юниона как такового нет, есть
multi request `_(ツ)_/``
 - `term: { keyword: alpha }`
 - `term: { keyword: beta }`

OR vs UNION

Multi request к одному шарду выполняется последовательно или параллельно?

Знаем ли мы это?

Знаем ли мы хоть что-нибудь?

Давайте это выясним!

OR vs UNION

...EsRally не умеет в мультиреквест, а у меня
не было времени продумать, как это
протестировать в обход ((

Range-запросы

Полигонные запросы

TL;не успел

- ES хранит геоданные в виде k-d деревьев
- ...которые, кажется надо обойти целиком в случае полигонного поиска
 - в общем, оче плохо

Scripting

Scripting

Скрипты приносят дополнительную нагрузку, и это - норма.

Но сколько стоит эта нагрузка?

Scripting

1.baseline, script = _score

2.baseline, script = 1

3.script = Math.pow(1003.14, 1003.14)

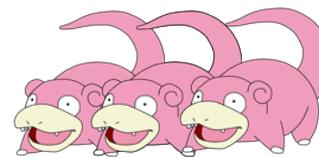
4.script = for (i = 1 ... 1000) j *= i

5.~~script = source.unindexedField~~

Scripting

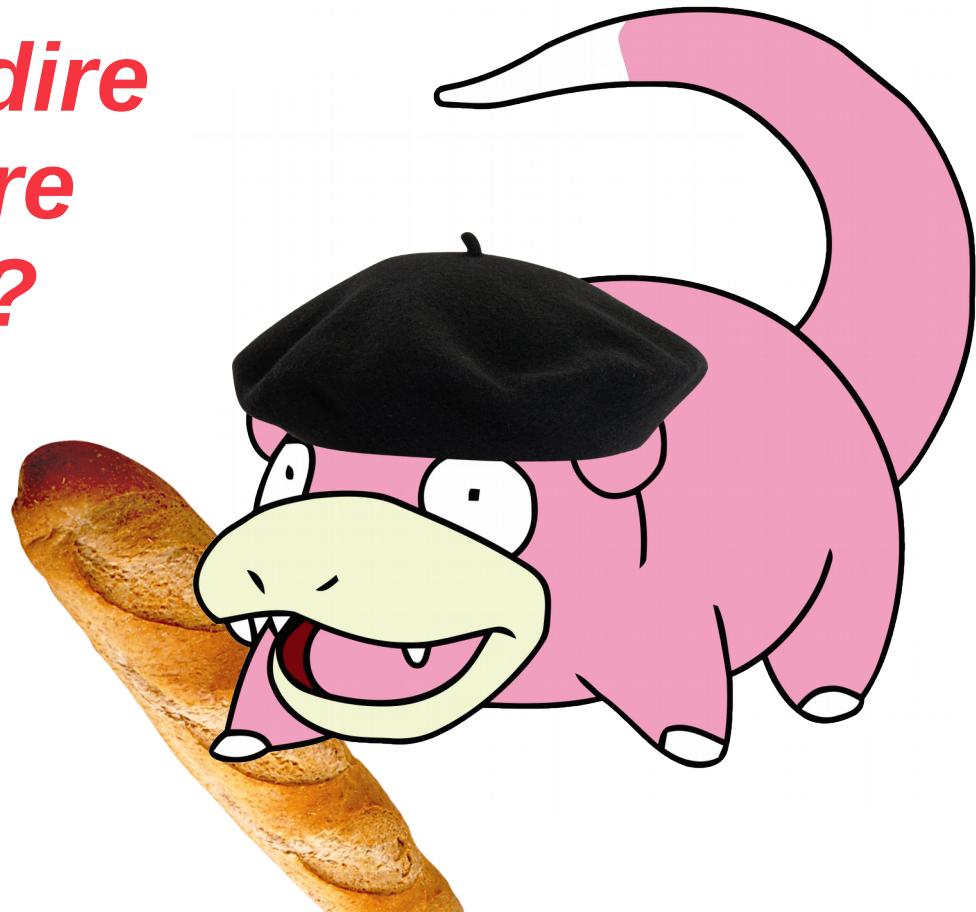
- 1._score: ~400rps
- 2.const: ~300rps
- 3.power: ~300rps
- 4.iterate: ~200rps
- 5.Unindexed field: well, don't do that, i've tried

Оценка



3 / 10

*Pouvez-vous me dire
comment se rendre
à la cryochambre?*



Collapsing

Collapsing

1. Baseline: match_all без коллапса
2. Тест: 5 индексов, match_all, коллапс по text_bytes

Collapsing

1.Baseline: ~350rps
2.5x + collapse: ~300rps

Оценка



dfs_query_then_fetch

dfs_query_then_fetch

По умолчанию, ElasticSearch использует TF/IDF каждого шарда в изоляции от других шардов, т.е. каждый шард определяет релевантность документа относительно своего содержимого.

dfs_query_then_fetch исправляет ситуацию за счет лишнего RTT.

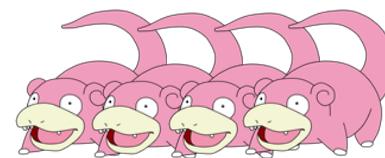
Сколько это стоит?

dfs_query_then_fetch

Обычный поиск: ~240rps

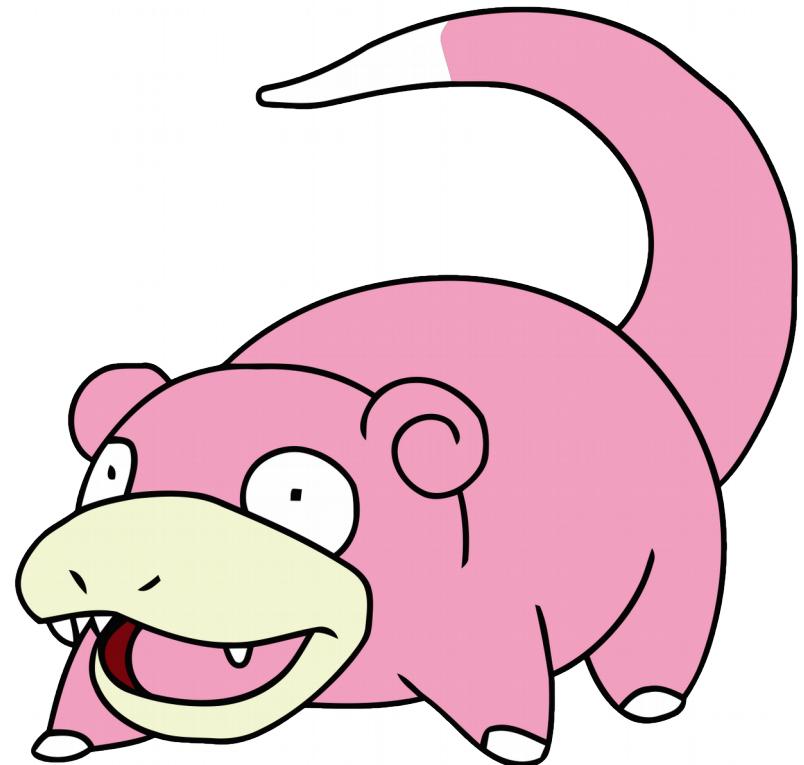
dfs_query_then_fetch: ~200rps

Оценка



4 / 10

*Скоро уже
начнется мезозой?*



Выводы

Выводы

- Все очень сложна
- Слепо доверять EsRally нельзя
- Близкие клиенты могут показывать совсем разные значения
- Доверяй, но профилируй

Благодарности

Благодарности

- Андрей Паньгин
- Алексей Шипилев
- Брендан Грегг
- Ярослав Щекин – за ссылку на OR-проблему
- Анно Хидеаки - за лучшую в мире драму
- Старый Хрыч - за то, что он есть
- Red bull

БНОПНЯШ & Answers

<https://github.com/ayte-io/slides>

Scripting

1.baseline, script = _score

2.baseline, script = 1

3.script = Math.pow(1003.14, 1003.14)

4.script = for (i = 1 ... 1000) j *= i

5.~~script = source.unindexedField~~