Обработка данных в RTB: быстро, дешево и на 98% точно

Павел Калайдин

Конференция разработчиков
высоконагруженных систем

Павел Калайдин

@facultyofwonder

данных очень много
памяти разумно мало
хотим все знать за один проход
время обработки - константно

max, min, mean

Как посчитать медиану?

Случайная выборка?

Reservoir sampling?
Возможно

Вероятностные алгоритмы?

Сойдет приблизительный ответ,
хотим знать распределение ошибки

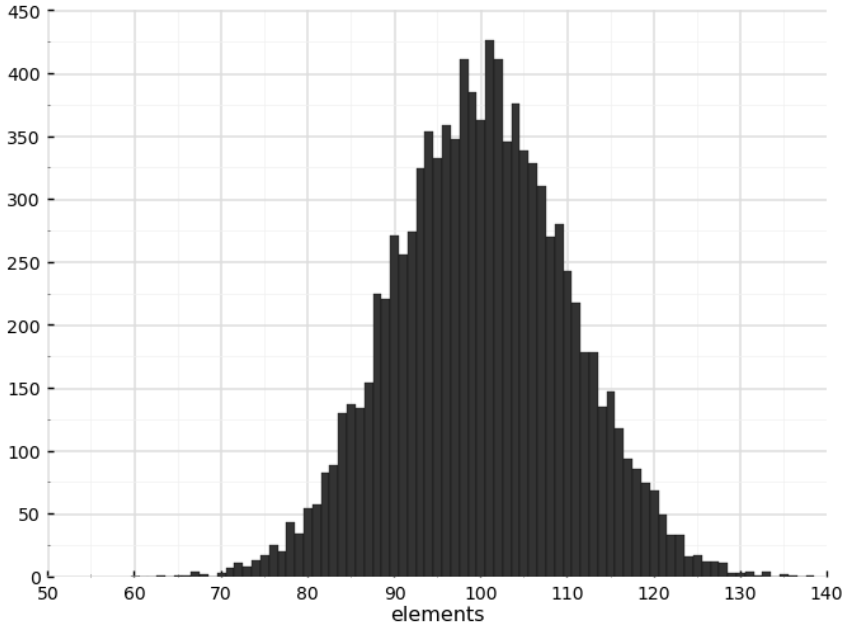
```
frugal <- function(stream) {
 m <- 0
 for (val in stream) {
   if (val > m)
     m = m + 1
   else if (val < m)
     m = m - 1
 }
 return(m)
}
```

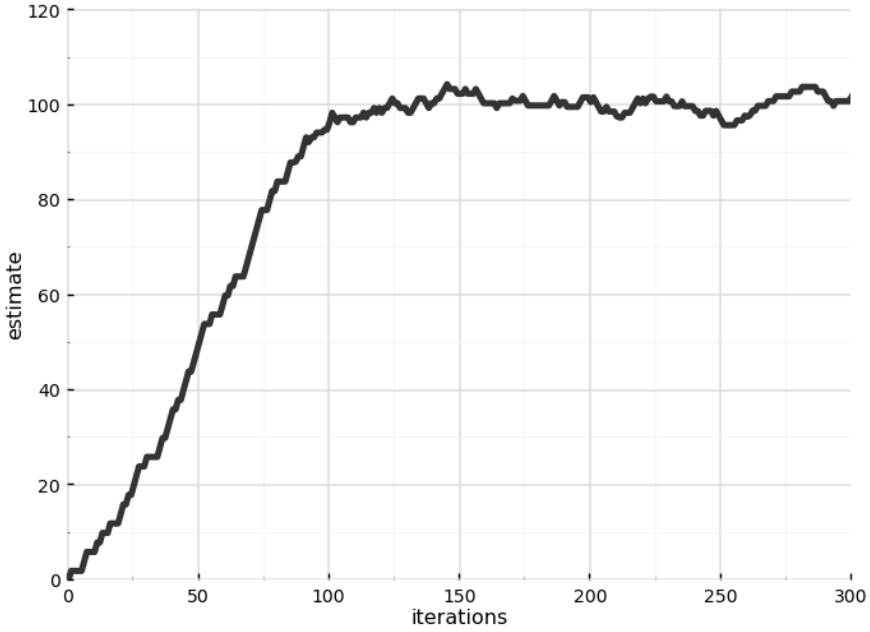
```
def frugal(stream):
  m = 0
  for val in stream:
    if val > m:
      m += 1
    elif val < m:
      m -= 1
  return m
```

```
def frugal(stream):
 m = 0
 for val in stream:
   if val > m:
     m += 1
   elif val < m:
     m -= 1
 return m
```

Реально работает?

Ограничение по памяти - 1 int!





Персентили?

```
def frugal_1u(stream, m=0, q=0.5):
 for val in stream:
   r = np.random.random()
     if val > m and r > 1 - q:
       m += 1
     elif val < m and r > q:
       m -= 1
 return m
```

Демо: bit.ly/frugalsketch

Потоковый + вероятностный = скетч

Что мы хотим?

Знать число уникальных пользователей

aka мощность множества или кардинальное число

Что мы хотим?

Знать число уникальных пользователей
по сайтам, интересам, временным интервалам

Когда мы это хотим?

Прямо сейчас

Данные:
10^{10} элементов,
10^{9} уникальных int32
40Гб

Решение в лоб: хеш-таблица

Хеш-таблица:
4Гб

HyperLogLog:
1.5кб, 2% ошибка

Все начинается с алгоритма LogLog

Представьте, что сегодня утром я бросал монетку и записал, какое максимальное число раз подряд выпала решка

2 раза
100 раз

В каком случае я бросал дольше?

Нас интересуют паттерны
в хешах входных значений
(число 0 = решек в начале)

Хешируем, не семплируем!*

* нужна хорошая хеш-функция

Ждем:

0xxxxxx хешей - ~50%

1xxxxxx хешей - ~50%

00xxxxx хешей - ~25%

и т.д.

оценка - 2^R,
где R - максимальное число лидирующих нулей

Я могу провести несколько экспериментов с подбрасыванием монетки и записать результаты на листок бумаги

и взять среднее число

Этот способ называется - стохастическое усреднение

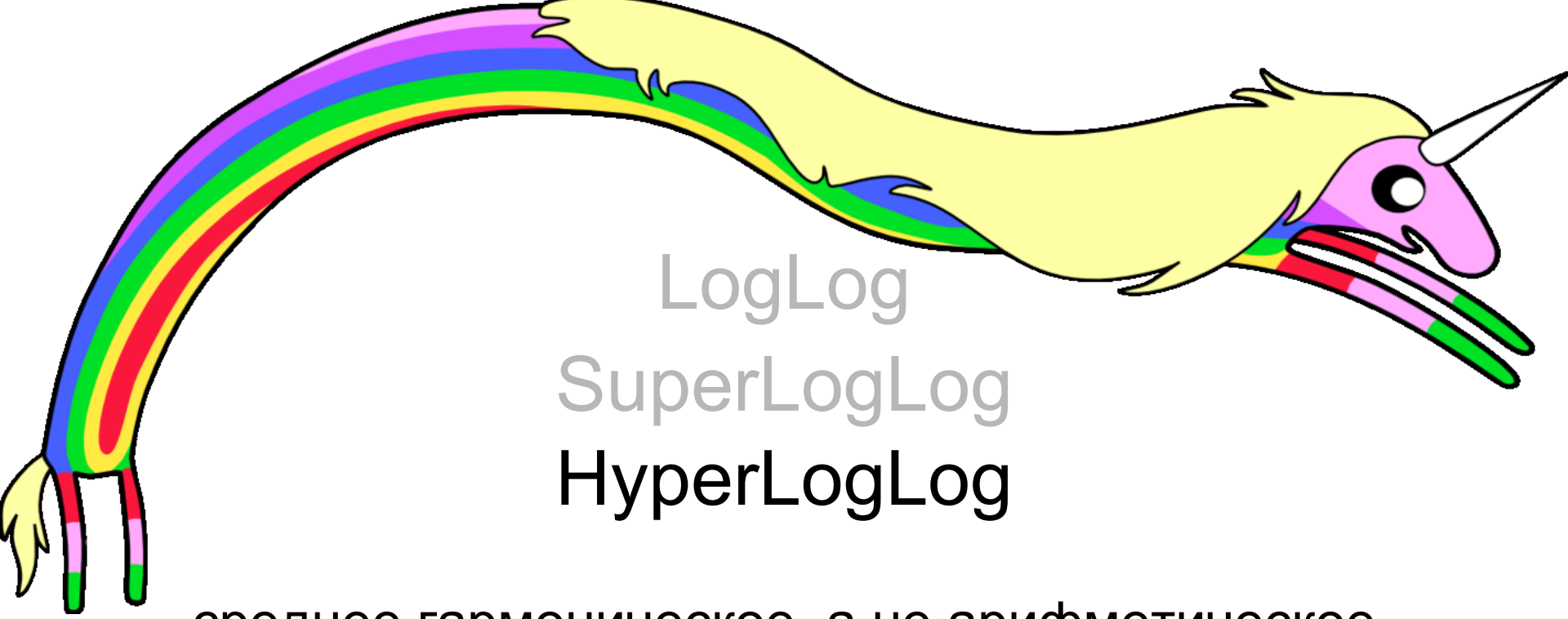
Напомню, текущая оценка - 2^R, где R - максимальное число лидирующих нулей

Будем использовать M корзин, в каждой из которой будем запоминать свой R

$$\alpha \cdot 2^{\frac{1}{M}\sum_{1}^{M} R_m}$$

где α - нормирующая константа

Это и есть LogLog алгоритм



LogLog

SuperLogLog

HyperLogLog

среднее гармоническое, а не арифметическое

\+ некоторые поправки

Число корзин и длина хеша определяют ошибку

LogLog

SuperLogLog

HyperLogLog

HyperLogLog++

Google, 2013

32 -> 64bit + поправки для маленьких мощностей

bit.ly/HLLGoogle

LogLog

SuperLogLog

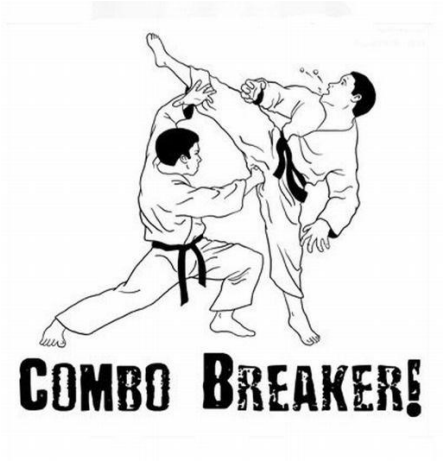
HyperLogLog

HyperLogLog++

Discrete Max-Count

Facebook, 2014

bit.ly/DiscreteMaxCount



Large scale?

Объединение без потери точности!

У нас есть два HLL-скетча, возьмем максимальное значение из каждой корзины

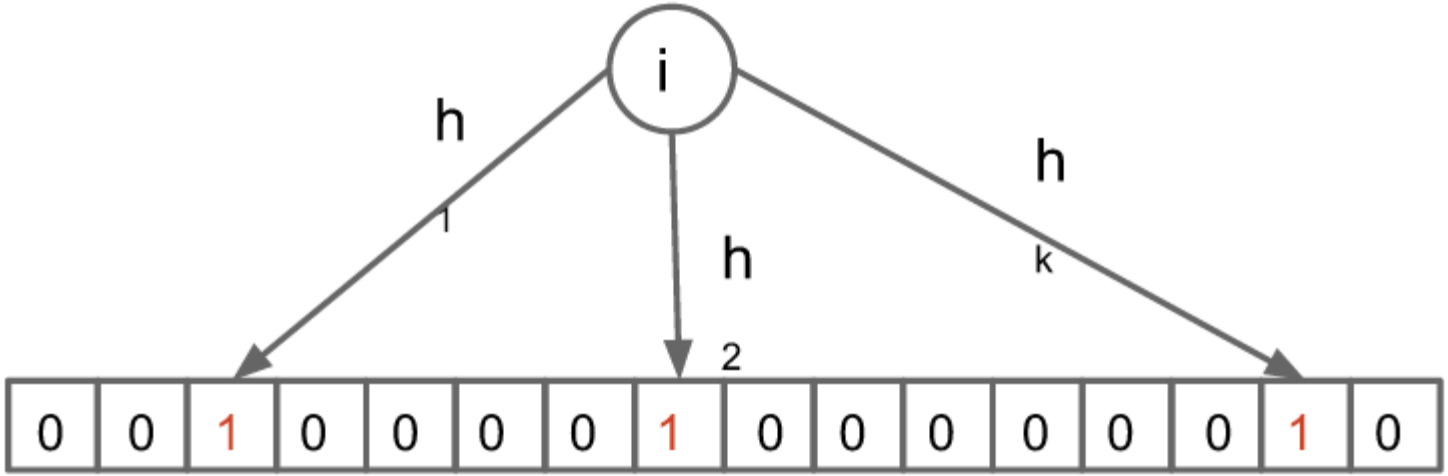
Voila! Результирующий скетч не потерял в точности

Код:
bit.ly/hyperloglog

Временное окно

bit.ly/SlidingHLL

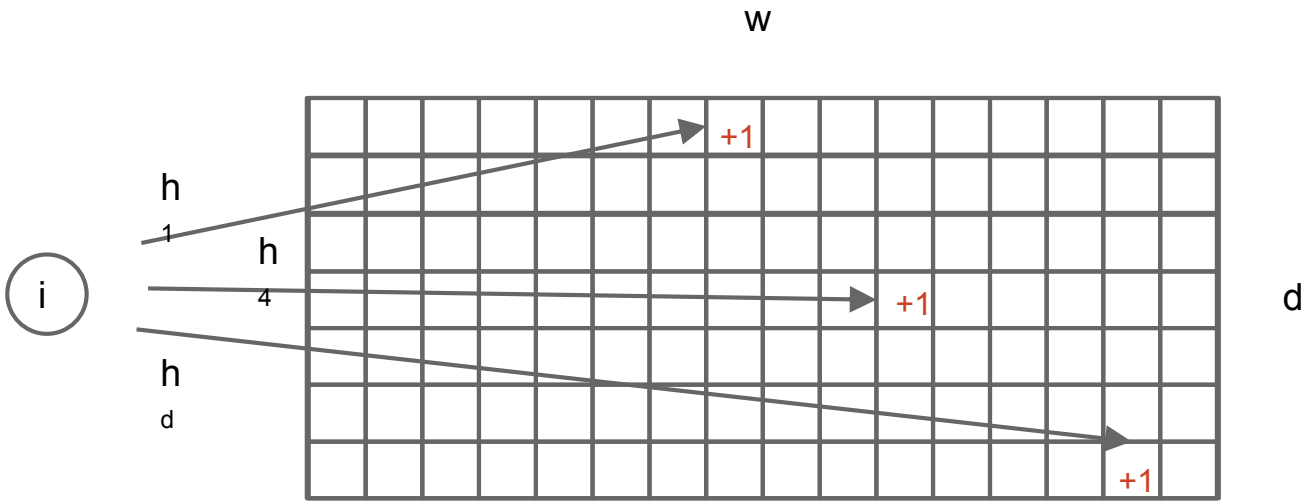
Проверка на вхождение – старый добрый фильтр Блума


i
h
1
h
2
h
k
0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0

Сколько раз мы встречали
человека в интернете
за эту неделю?

Ответ - Count-Min скетч

bit.ly/CountMinSketch



Оценка - возьмем минимум из d значений.

хешируем, не семплируем
даем приблизительный ответ
экономим память
обрабатываем по мере поступления

этот слайд - “скетч” всего доклада

Много разных скетчей для разных задач:

персентили
частоты
коэффициент Жаккара
(похожесть множеств)

Это развивающаяся область:
stay tuned!

Еще по теме

Блог Neustar Research:
bit.ly/NRsketches
Обзор скетчей:
bit.ly/SketchesOverview
Лекции по потоковым алгоритмам:
bit.ly/streaming-lectures

Спасибо за внимание и happy sketching!

Знаете, как посчитать
экономнее и точнее?
pavel@rutarget.ru

Бонус

HyperLogLog на SQL:
bit.ly/HLLinSQL