Noel Welsh

_.underscore

myna

Real-time A/B testing
mynaweb.com

Real-time Analytics in Scala

Or, getting better numbers to make better decisions

Scala eXchange 2012

Plan

- Motivation
- Streaming algorithms
- Hash functions
- Frequent items
- Set cardinality
- Implementation

Motivation

I write code

I'm not paid to write code

I'm paid to create value

How does one create more value?

Hard work?

Talent?

Vision?

Roll the dice

Or, Empiricism

Build

Measure

Learn

The only way that consistently works

How do I build?

We're developers. This is what we do.

How do I learn?

Sometimes things are obvious

Sometimes things are not obvious

Use statistical techniques

Intricate, but not my focus today

How do I measure?

Harder than you might think

The focus today

Measurements are

complex

- Numbers (e.g. how many engaged users)
- Strings (e.g. survey answers)
- Events (e.g. a user engages with the product)
- And compound data

Frequency variations

- ~10 signups / day
- ~500K requests / day
- Frequency and value are inversely related

Recency matters

How to store measurements?

Buy don't build

Analytics SaaS too limited

Hadoop too complex

Build don't buy

Requirements

Mental energy must be conserved

I don't want to think

about what I can store

I don't want to think about scalability and sys admin

I don’t want to think about when I can get results

How?

Infrequent data is easy to store and analyse

For frequent data I will lose accuracy for scale and speed

and it doesn't matter

Core technical element: streaming algorithms

Definition: a streaming algorithm processes each data point once

I can apply to data on disk

I can apply to a live data stream

but I can only process each data point once

The cost: probably approximately correct answers

The benefits: use vastly less memory than exact algorithms (KB vs GB)

The benefits: yield real-time results

Summary

- Streaming algorithms process each data point once
- Trade accuracy for reduced memory usage and real-time update

Hash Functions

Streaming algorithms
love hash functions!
Let's do a quick review

Deterministic

Uniform distribution

Bit values are independent

In Practice?

Use Murmur Hash 3

In Scala

- scala.util.hashing Scala 2.10+
- Google Guava
- Be aware of version (2 vs 3), variant (ia32 vs x86_64), and size (32-bit vs128-bit)

Frequent Items

Frequent Items

- Find and count occurrence of most frequent items in set. “Who are our most active users?”

Space Saver

- Store *k* tuples of (item, count)
- Observe item
  - If it's in our list, increment the count
  - Otherwise remove the least frequent item and replace with this one, keeping the count

That's it!

Properties

- Deterministic
- Uses O(*k*) space
- Constant time updates
- Error depends on data distribution

In Scala

- Mutable Stream Summary data structure
  - Two-level tree
    - Double linked list at each level
- scala.collection.mutable.DoubleLinkedList
- ~100 lines of code
- Immutable would be nicer

Distinct Values

Count the size of a set

How many users arrived from the Scala eXchange website?

The Joy of Sets

- With set algebra we can answer many questions of interest
- How many users came from Scala eXchange OR Twitter?
- How many users came from Twitter AND purchased?

An Analytics Platform

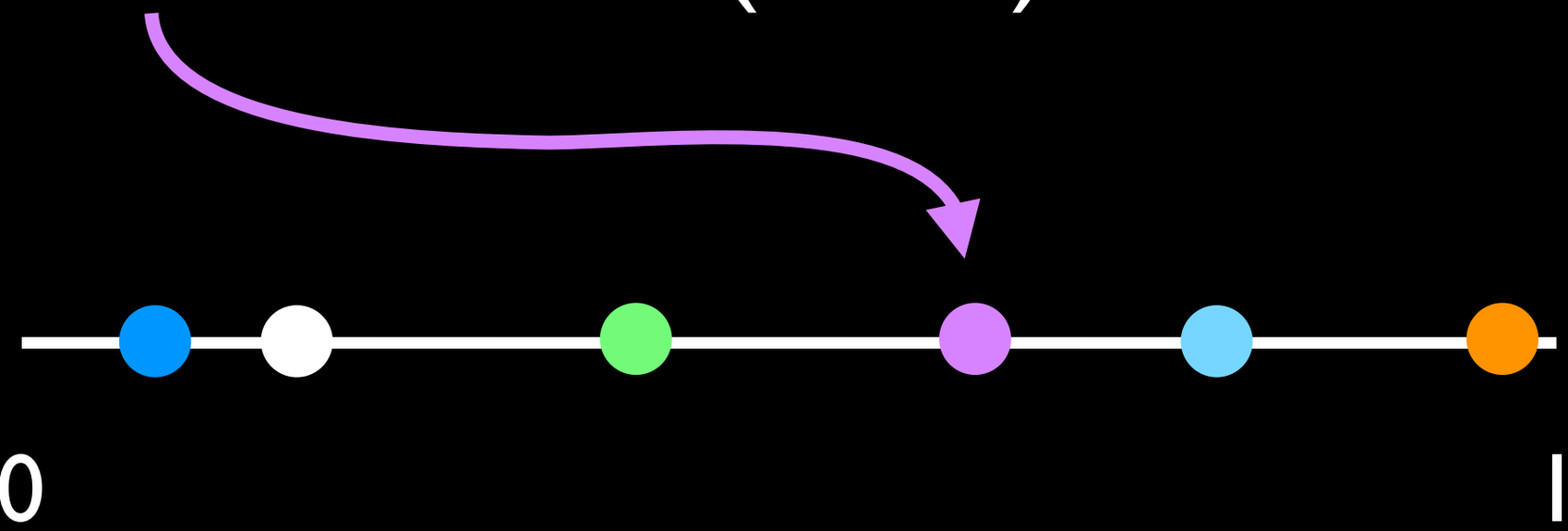
- We can build a fairly general analytics platform with just distinct values and set operations!

Many Roads

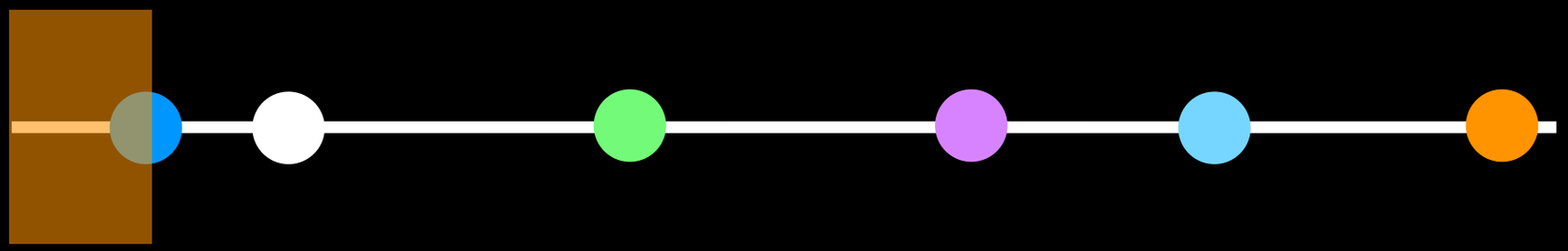
- A lot of research has been done
- Flajolet-Martin sketches (LogLog and HyperLogLog) are popular
- Optimal (but complex) algorithm published in 2010

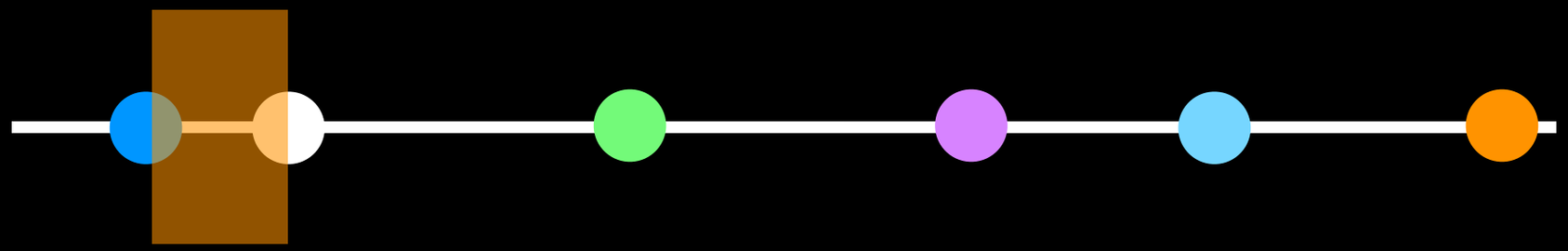
k-Minimum Values

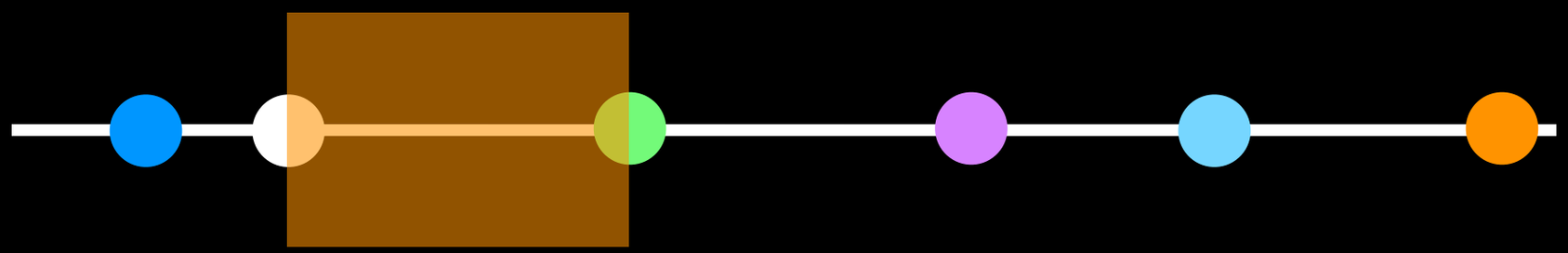
index = hash(data) / maxHash

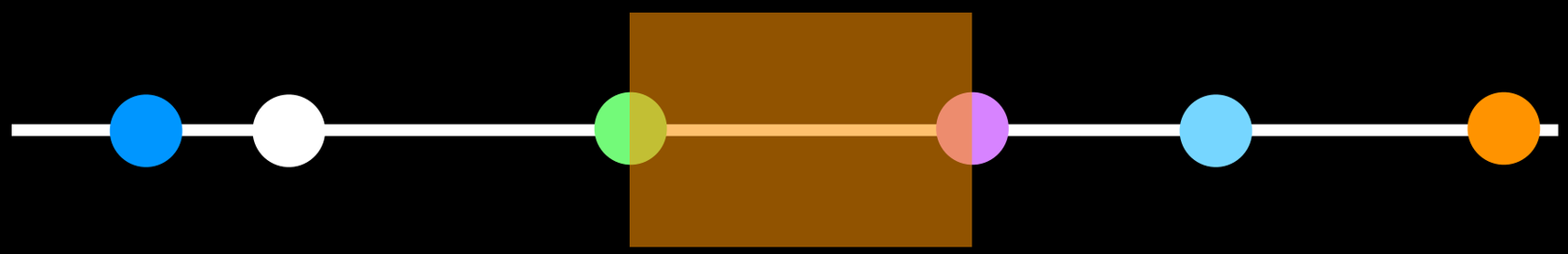


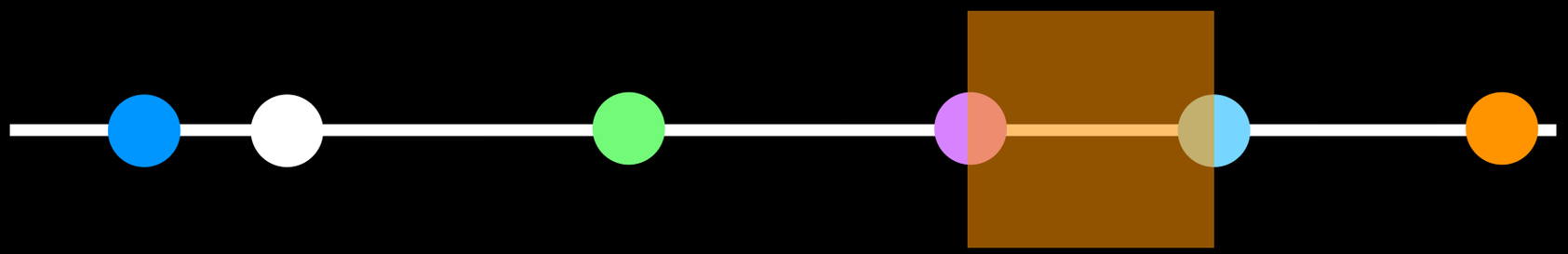
Average distance between elements inversely proportional to cardinality



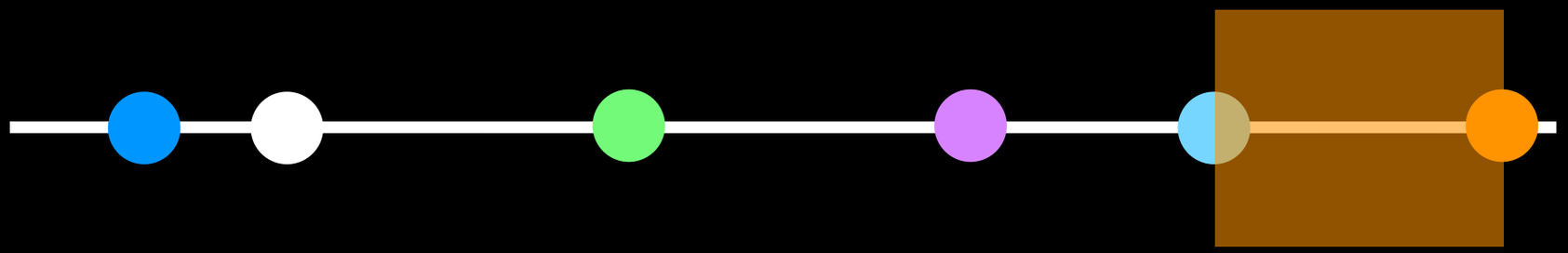


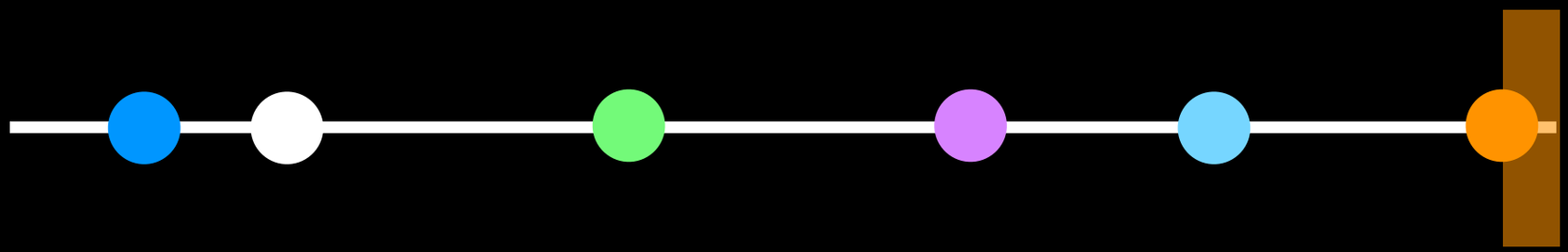






Thursday, 22 November 12





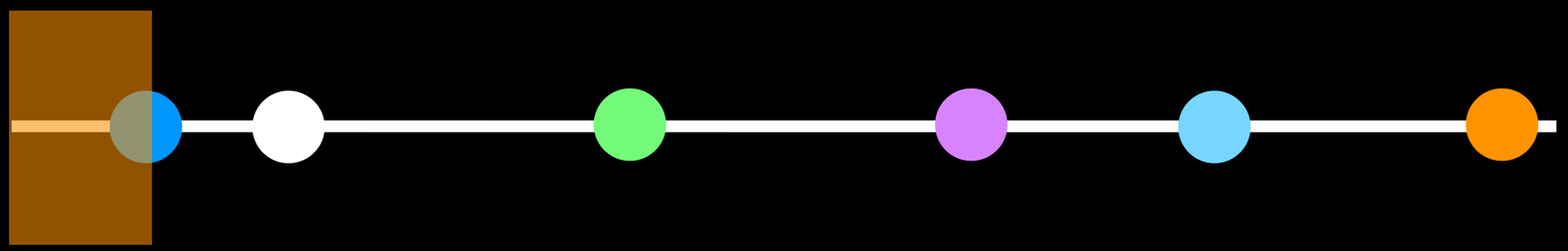
Thursday, 22 November 12

Can’t store all these distances

Big Idea

- Store minimum value. This gives us *one* distance
- Estimate size of set

$$|S| = \frac{1}{\text{minimum}}$$

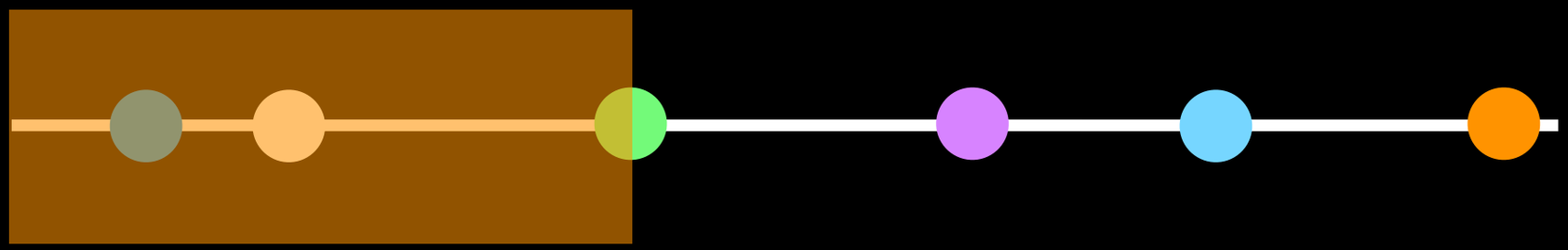


Very noisy!

Refinement

- Store *k* minimum values
- Estimate cardinality as

$$|S| = \frac{k-1}{\text{largest value stored}}$$



$k = 3$

Error Rate

$$\mathbb{E}\left[\frac{|S|_{est} - |S|}{|S|}\right] \approx \sqrt{\frac{2}{\pi(k-2)}}$$

- Independent of size of set
- See papers for other error bounds

Example

- Storing $k = 1024$ values (typically 4K) gives expected error of 2.5%

Set Algebra

- Set union is just the *k* minimum values from the union of the two sets
- Set intersection from Jaccard coefficient
- Set difference if we add counters to each element we store

Summary

- *k*-Minimum Values is simple
- Space usage and error are small
- Real-time processing extremely feasible
- Look at (Hyper)LogLog if applying for real

Implementation

Must be convenient to use

Must be convenient to implement

Must be fast enough

Use

Add data: /event

View data: `/view`

JSON in and out

Required `key` field

Implementation

Overview

- Tree of processing steps
- Take apart and validate JSON
- End by adding it to a sink (e.g. save to disk, increment counter)

Common Abstraction

- A sink is something that is initially empty
- and we can add data to it
- A monoid!

Monoids

- Composable
- Pimpable via type classes (so `Int` is a monoid, etc.)
- Standard library for free, via Scalaz

Speed

Batch I/O

Save to disk every 5 seconds

Allow clients to batch requests

Use Futures

Fast is fairly easy in Scala

More

Other Algorithms

- We've only touched the surface
- Quantiles, clustering, graph properties, etc.
- Online learning is an area I'm excited about. Goes beyond summarising data to taking actions.

Code

- Clearspring's stream-lib implements the most common streaming analytics algorithms in Java.
https://github.com/clearspring/stream-lib

Writing

- Lots of blog posts, tutorials, etc. Ask Google
- Alex Smola's course is a good overview http://alex.smola.org/teaching/berkeley2012/streams.html
- *k*-Minimum Values is in http://www.mpi-inf.mpg.de/~rgemulla/publications/beyer07distinct.pdf

Talking

- Interested? Let's chat!

Me

- Slides will be on noelwelsh.com
- noel@underscoreconsulting.com
- noel@mynaweb.com
- @noelwelsh