Haskell for data science

About me

Haskell

Not mainstream


gifbin.com

Data Science Mainstream

Deep Learning

Pattern recognition

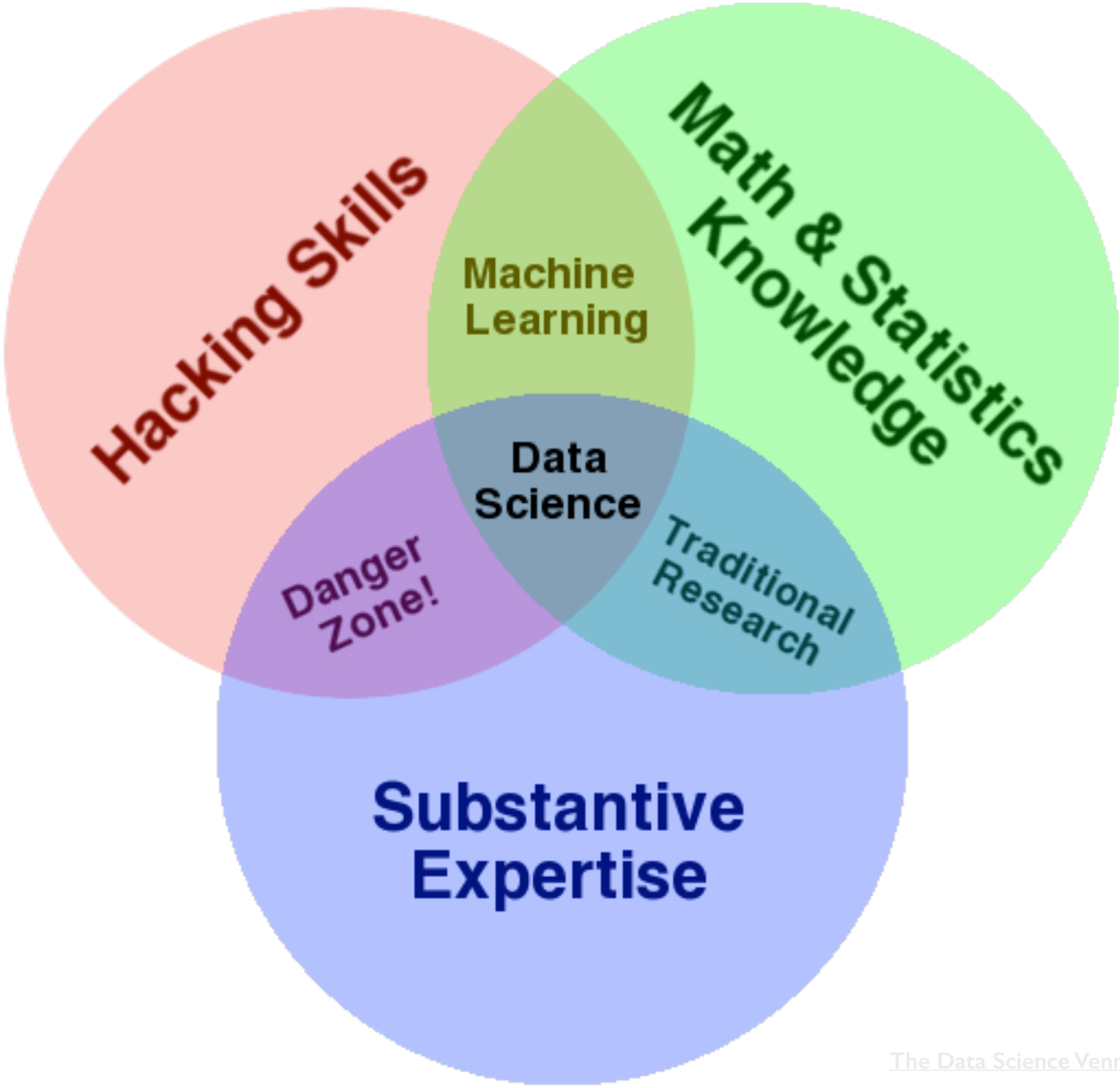
Machine Learning

Data Science

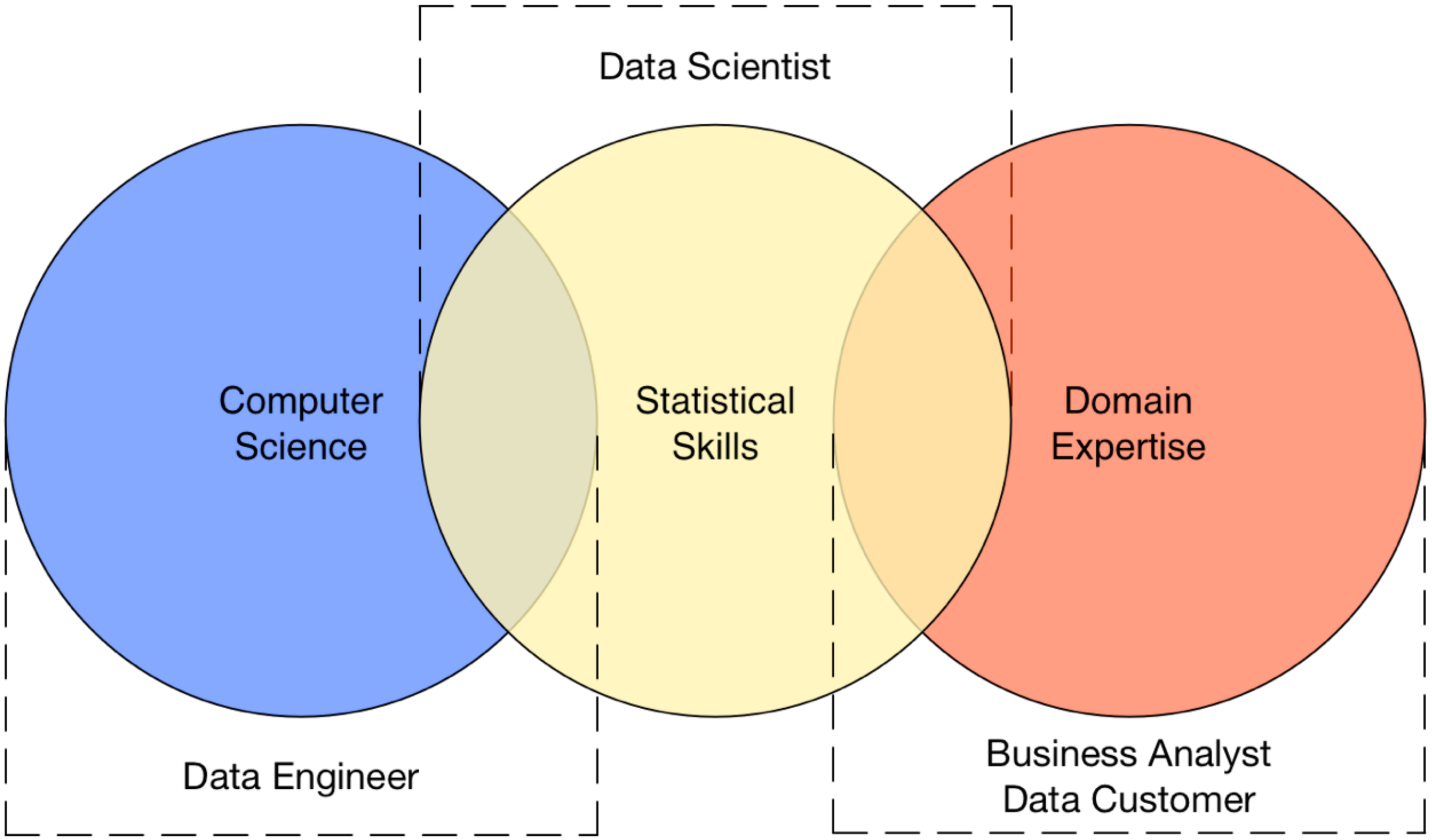
Big Data

Data Mining

Statistics

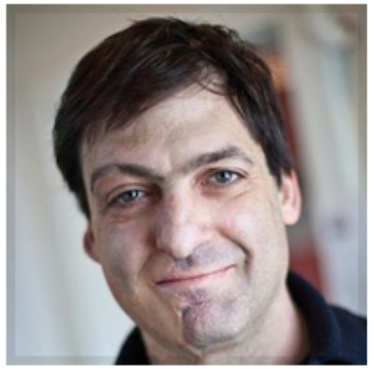


The Data Science Venn Diagram by Drew Conway



https://medium.com/@KevinSchmidtBiz/data-engineer-vs-data-scientist-vs-business-analyst-b68d201364bc

Big Data



Dan Ariely

January 6, 2013 ·

Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...

https://www.facebook.com/dan.ariely/posts/904383595868

IHaskell

A Haskell kernel for IPython

github.com/gibiansky/IHaskell

cabal install

ihaskell

```
ipython console
--kernel haskell
```

ipython notebook

IHaskell

In [1]:
```
data Thing = Thing Int Int
```

In [2]:
```
f :: Int → Thing → Int
f i (Thing a b) = i *( a + b)
```

In [3]:
```
import Data.List.Utils (replace)
```

In [4]:
```
x = Thing 10 9
```

In [5]:
```
f 3 x
```

```
57
```

Demo

html s3

Loading Data

txt

json

csv

github.com/slon1024/

haskell_read_data

txt

```
import Data.List.Split (splitOn)

main :: IO ()
main = do
  input <- readFile "data/input.txt"
  print $ filter allowQQQ $ lines input

allowQQQ :: String -> Bool
allowQQQ line = (getDomain line) == "qqq.com"
    where getDomain = last . splitOn "@" . last . splitOn " "
```

CSV

```
import Text.CSV

main = do
  let fileName = "data/input.csv"
  input <- readFile fileName
  let csv = parseCSV fileName input
  either handleError doWork csv

handleError csv = print "not a CSV"

doWork csv = print $ filter (\row -> last row == "1") $ rows csv
rows csv = filter (\x -> length x == length (head csv)) $ tail csv
```

json

```haskell
data Person = Person {   name :: String
                       , email :: String
                       , created_at :: String
                       , is_deleted :: Maybe Int
                       } deriving Generic

instance FromJSON Person

main :: IO ()
main = do
  input <- B.readFile "data/input.json"
  let mm = decode input :: Maybe Person
  case mm of
    Nothing -> print "error parsing JSON"
    Just m -> (putStrLn . printPerson) m

printPerson m = (show.name) m ++" has a email: "++ (show.email) m
```

s3

```haskell
main :: IO ()
main = do
  cfg <- Aws.baseConfiguration
  let s3cfg = Aws.defServiceConfig :: S3.S3Configuration Aws.NormalQuery

withManager $ \mgr -> do
  S3.GetObjectResponse { S3.gorResponse = rsp } <-
    Aws.pureAws cfg s3cfg mgr $
      S3.getObject "haskell-aws" "cloud-remote.pdf"

  responseBody rsp $$+- sinkFile "cloud-remote.pdf"
```

Demo

cabal install **hxl**

cabal install

HandsomeSoup

github.com/UweSchmidt/hxt

github.com/egonSchiele/HandsomeSoup

Extract elements

```
doc <<< css "a"
doc <<< css "*"
doc <<< css "a#link1"
doc <<< css "a.foo"
doc <<< css "p > a"
doc <<< css "p strong"
doc <<< css "#container h1"
doc <<< css "img[width]"
doc <<< css "img[width=400]"
doc <<< css "a[class~=bar]"
doc <<< css "a:first-child"
```

http://egonschiele.github.io/HandsomeSoup/

html

```
import Text.HandsomeSoup
import Text.XML.HXT.Core

main = runX $ fromUrl "http://www.datatau.com" >>> css "td.title a" !"href"
```

Demo

parallel-io

Haskell combinators for executing IO actions in parallel on a thread pool.

http://batterseapower.github.io/parallel-io/

```
cabal install
parallel-io
```

github.com/batterseapower/parallel-io

github.com/slon1024/

haskell_parallel_io

parallel-io

```
textFromUrl url = do
  doc <- parsePage url
  print $ (url, length $ clean doc)

main = do
  links <- runX $ fromUrl "http://www.datatau.com" >>> css "td.title a" !"href"

  parallel_ $ map textFromUrl links
  stopGlobalPool
```

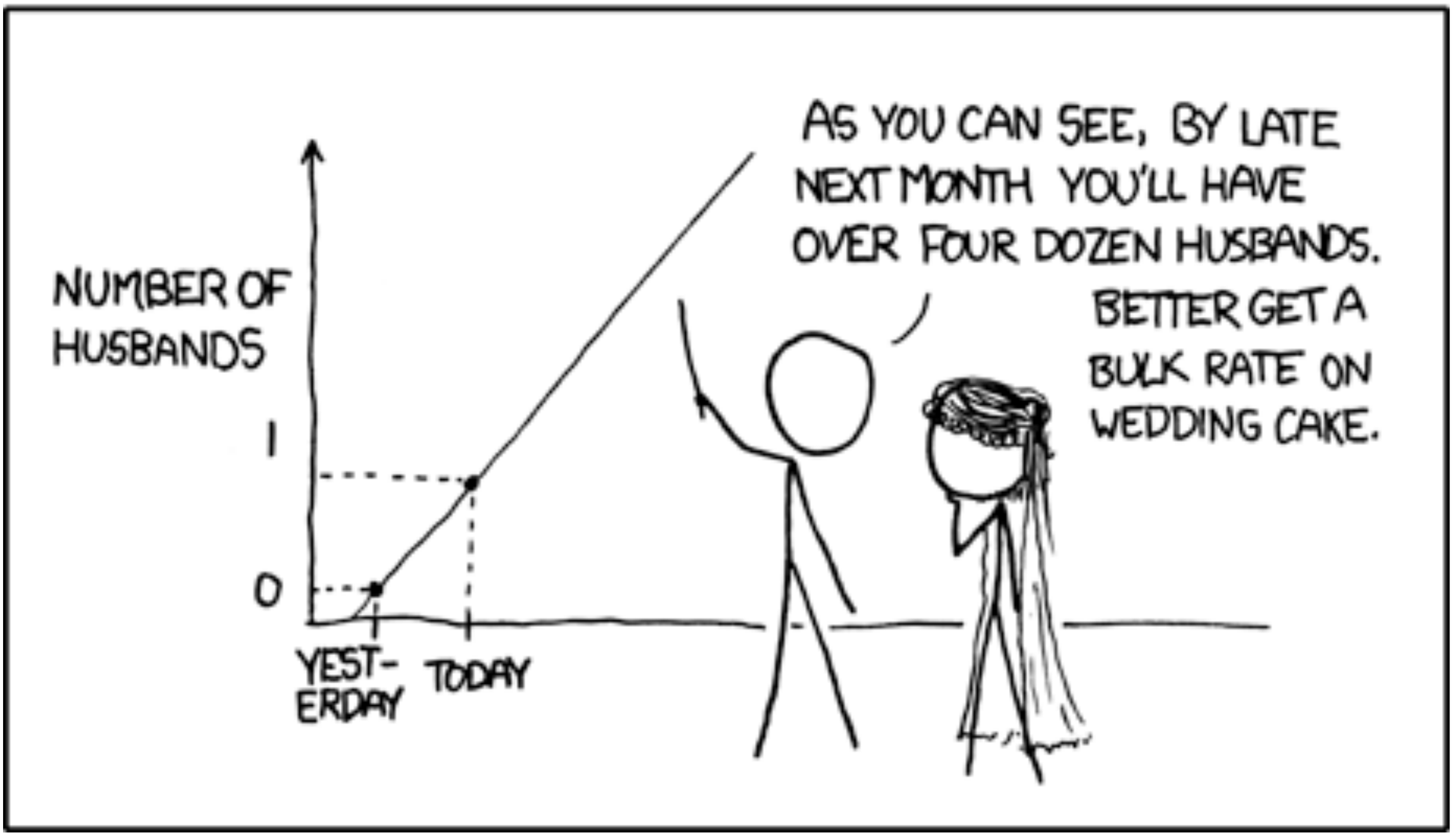
parallel-io

```
textFromUrl url = do
  doc <- parsePage url
  print $ (url, length $ clean doc)

main = do
  links <- runX $ fromUrl "http://www.datatau.com" >>> css "td.title a" !"href"

  parallel_ $ map textFromUrl links
  stopGlobalPool
```

Demo

Statistics

MY HOBBY: EXTRAPOLATING



http://xkcd.com/605/

cabal install

statistics

gist.github.com/slon1024/d0aa326c0ed38c6a1f69

Metrics

```
metrics = [
    ("mean",             S.mean),
    ("harmonicMean",     S.harmonicMean),
    ("geometricMean",    S.geometricMean),

    ("variance",         S.variance),
    ("varianceUnbiased",S.varianceUnbiased),

    ("stdDev",    S.stdDev),
    ("kurtosis", S.kurtosis)
    ]
```

Distributons

```
-- Probability density function
D.density standard 0

-- Cumulative distribution function
D.cumulative standard  3
D.cumulative (normalDistr 0 5) 15
D.cumulative (poisson 1) 0

-- Other
D.quantile standard 0.9
D.stdDev standard
```

cleaning data

Managing data

treating missing values

data transformations

random forest k-nn

...

Evaluating model(s)

linear regression svm

naïve bayes

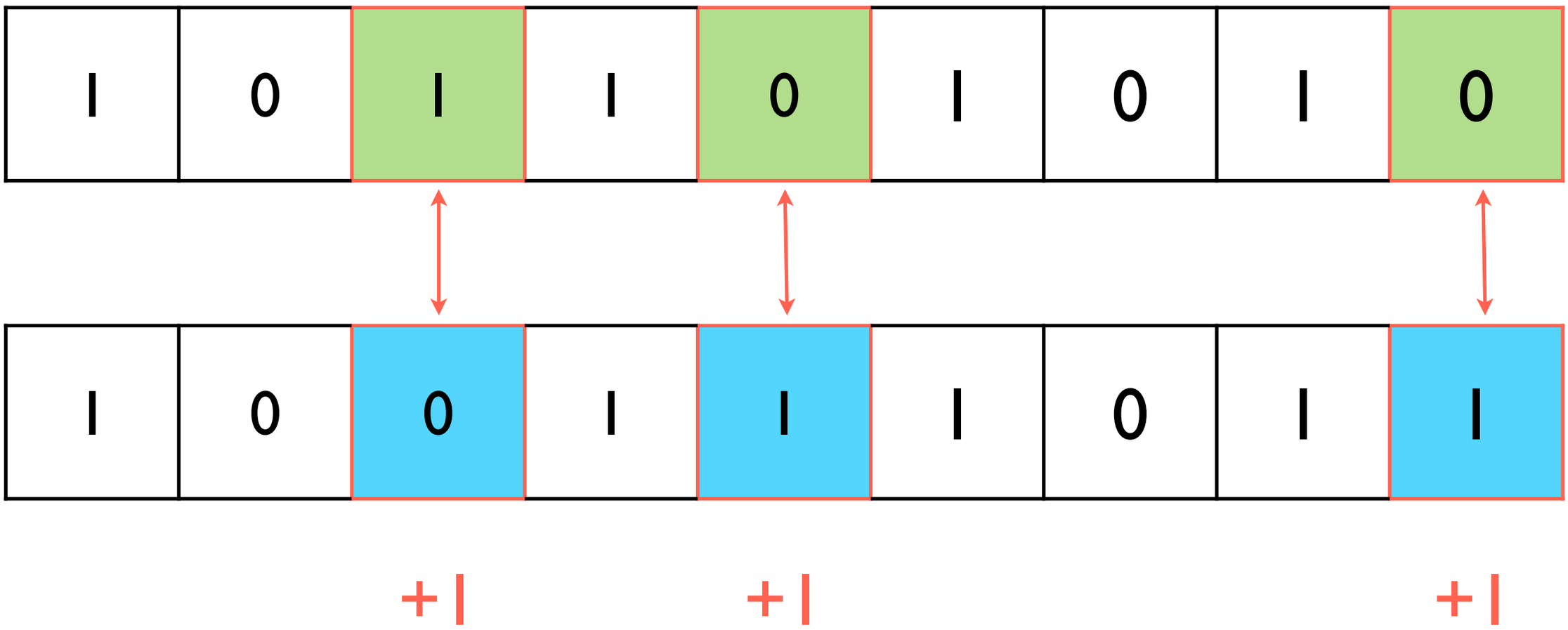
Perceptual hash

phash.org

github.com/slon1024/

haskell_phash

Hamming distance



hd is 3

phash

=1143699332312217752

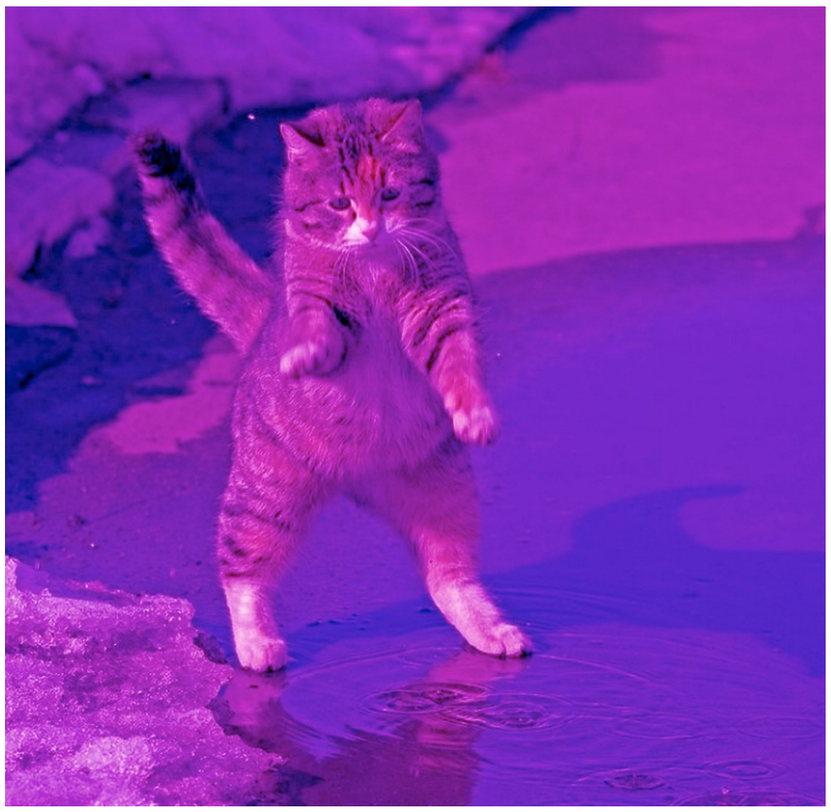
phash

= 5553391311484719000

phash

= 17009666824369565794

Change color

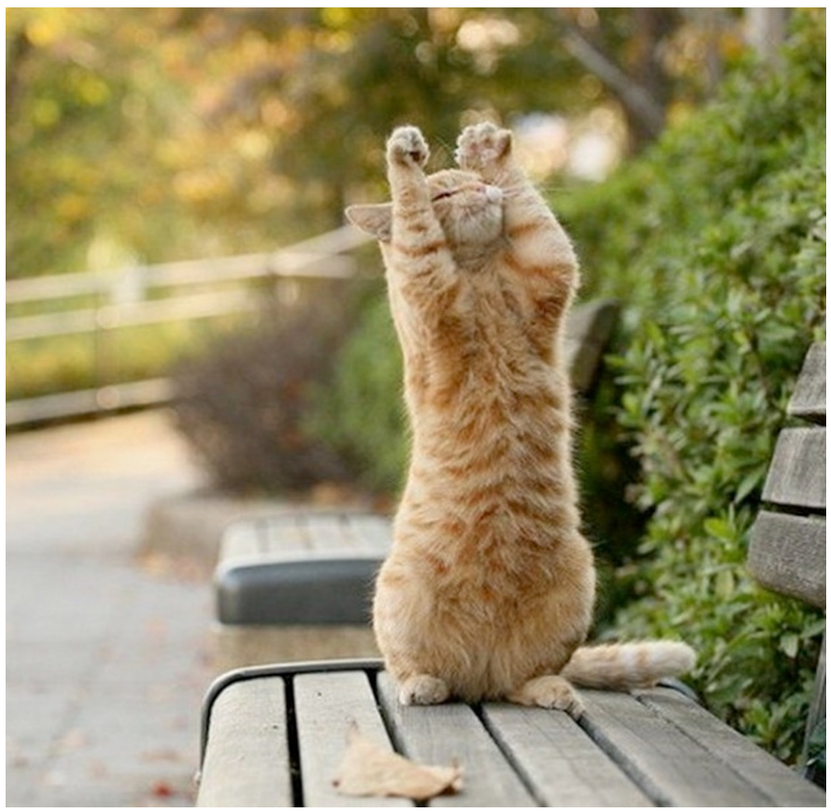


df = 0

df = 4

df = 4

Change color

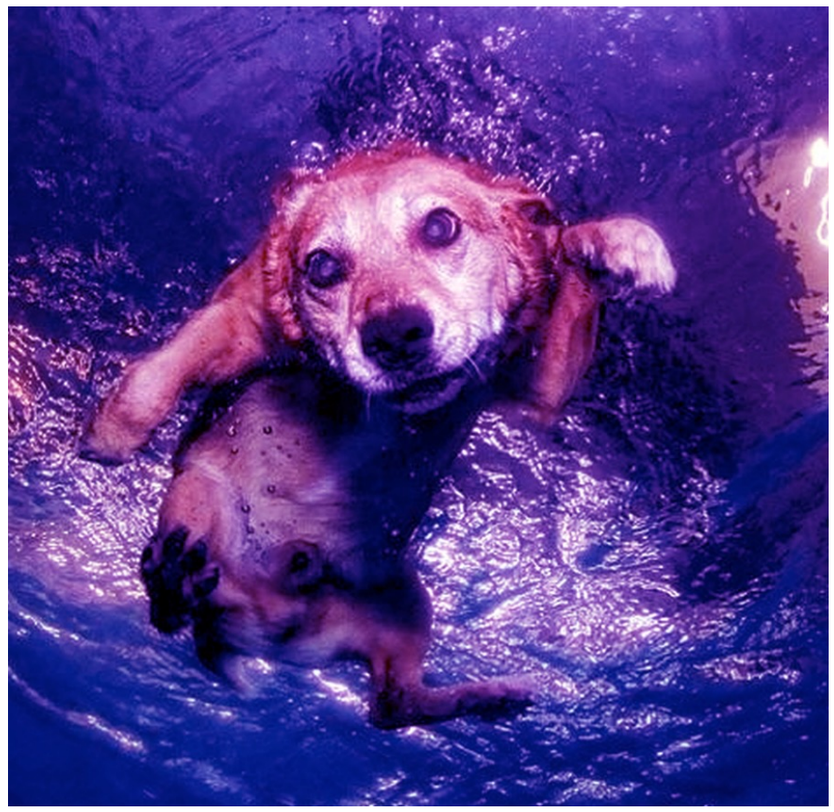


df = 4

df = 4

df = 4

Change color



df = 2

df = 2

df = 2

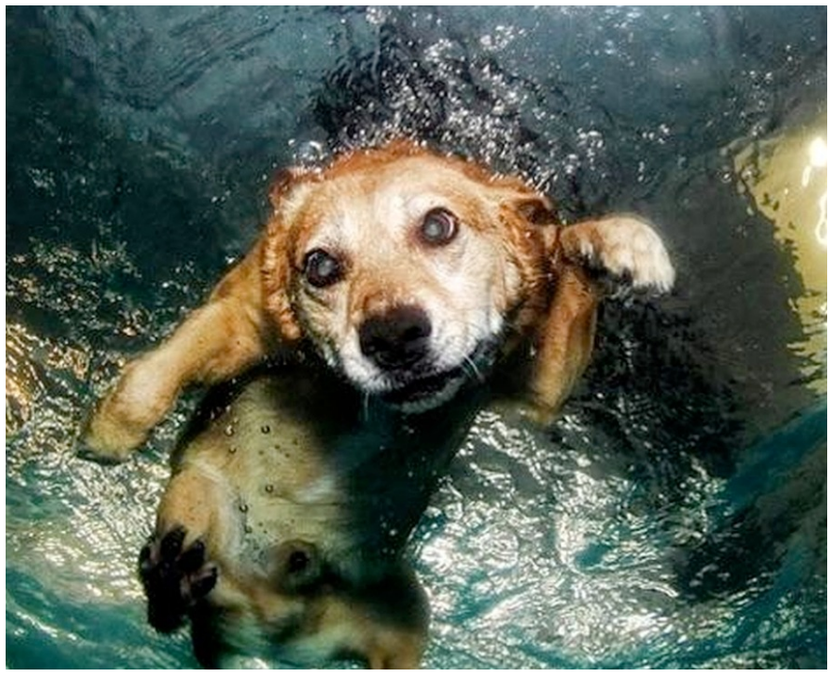
Add a new element

df = 12

df = 12

df = 16

Crop

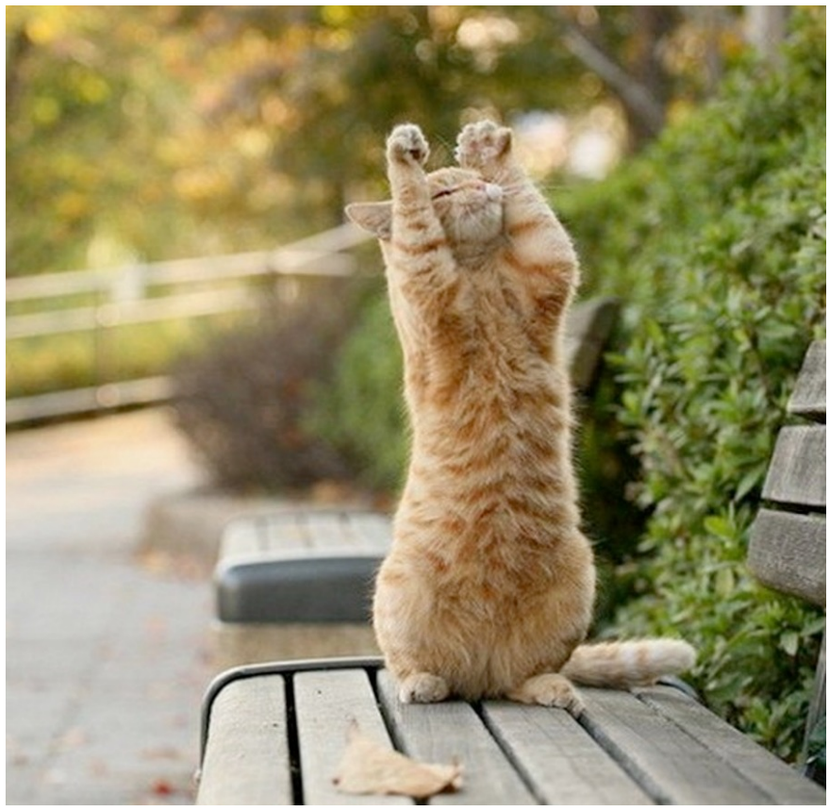


df = 26

df = 26

df = 34

Different images



df = 34

df = 34

df = 32

Parallel arrays

- Repa (REgular PArallel arrays)
- Yarr — yet another array library
- ...

NLP

natural language processing

Big Data

ZooKeeper

Kafka

Storm

github.com/slon1024/

vagrant-cluster-storm

./setup.sh

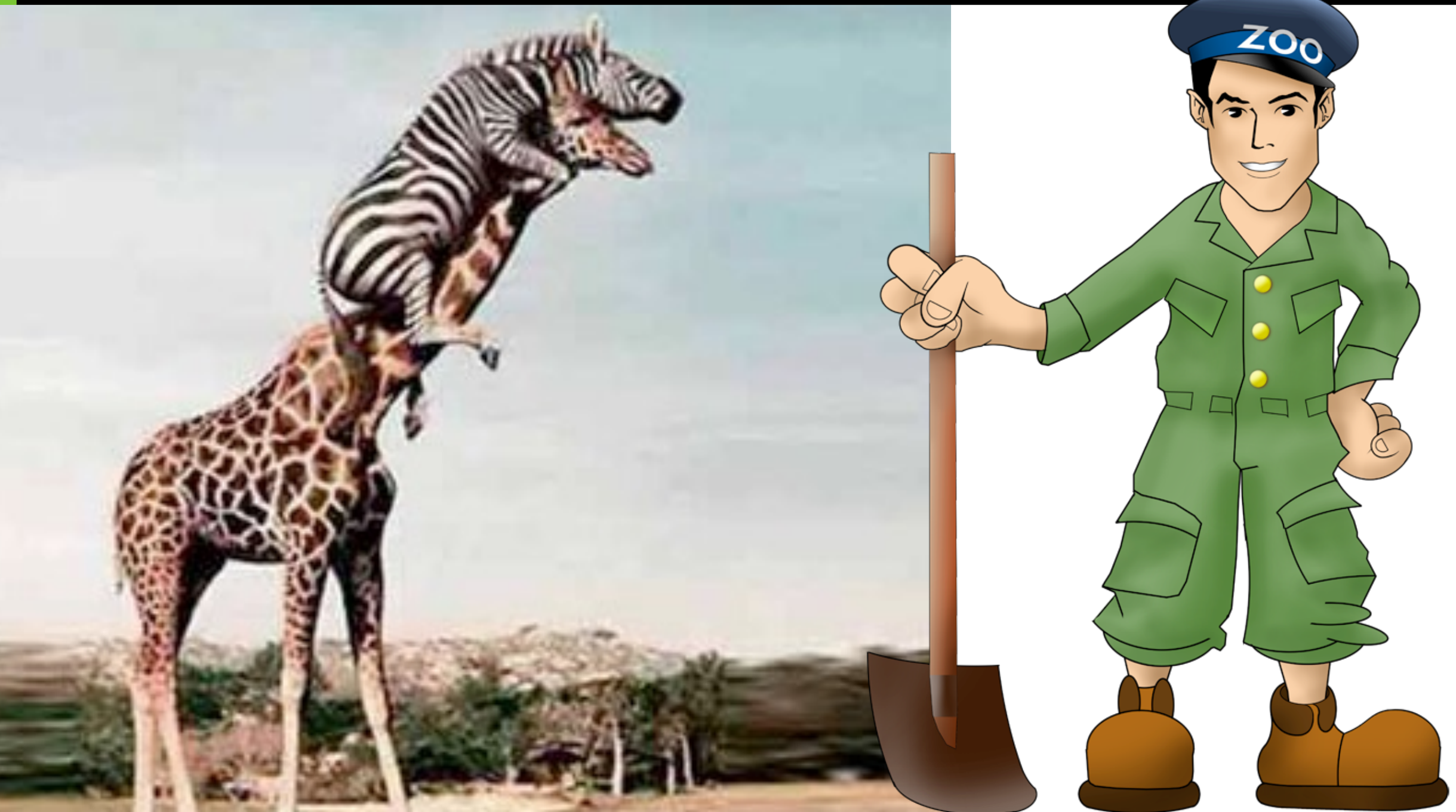
github.com/slon1024/vagrant-cluster-storm

mini cluster

- alpha
- beta
- gamma

Apache ZooKeeper

An open source server that reliably coordinates distributed processes



cabal install **hzk**

github.com/dgvncsz0f/hzk

github.com/slon1024/

haskell_zookeeper

create

```
create :: ZK.Zookeeper -> String -> IO String
create zh path = do
    node <- ZK.create zh path (Just $ C8.pack "value") ZK.OpenAclUnsafe []
    case node of
        Left ZK.NodeExistsError -> return "a node already exists"
        Left e  -> error $ "error: " ++ show e
        Right _ -> return "a node has been created"
```

get

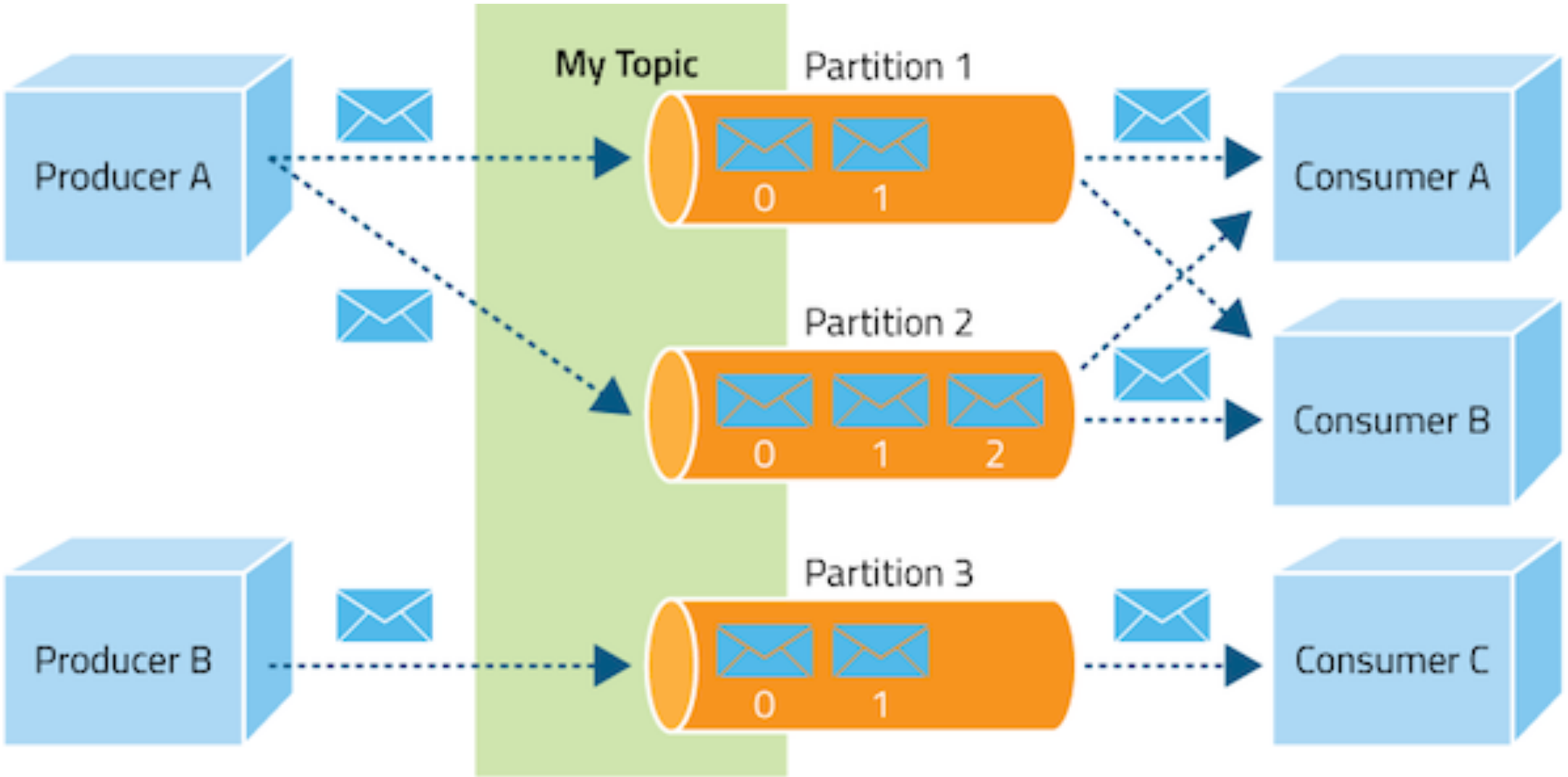
```
get :: ZK.Zookeeper -> String -> Maybe ZK.Watcher -> IO String
get zh path watcher = do
    node <- ZK.get zh path watcher
    case node of
        Left e  -> error $ "error: " ++ show e
        Right (Just value, st) -> return $  "value: " ++ C8.unpack value ++ show st
        Right (Nothing, st) -> return $ show st
```

set

```
set :: ZK.Zookeeper -> String -> String -> IO String
set zh path value = do
    node <- ZK.set zh path (Just $ C8.pack value) Nothing
    case node of
        Left e    -> error $ "error: " ++ show e
        Right res -> return $ show res
```

Demo

Kafka



http://www.cloudera.com/content/cloudera/en/documentation/cloudera-kafka/latest/topics/kafka.html

```
cabal install
```

haskakafka

github.com/cosbynator/haskakafka

github.com/slon1024/

haskell_kafka

Producer

```
producer message = do
  let
      kafkaConfig = [("socket.timeout.ms", "50000")]
      topicConfig = [("request.timeout.ms", "50000")]
      partition   = 0
      broker      = "10.77.1.2:9092"
      topic       = "test"
```

```
withKafkaProducer kafkaConfig topicConfig broker topic
    $ \kafka topic ->
        -- produceMessage topic (KafkaSpecifiedPartition partition) kafkaMessage
        -- produceKeyedMessage topic keyMessage
        produceMessageBatch topic KafkaUnassignedPartition messages
        where
            kafkaMessage = KafkaProduceMessage $ C8.pack message
            keyMessage = KafkaProduceKeyedMessage (C8.pack "Key") (C8.pack message)
            messages  = replicate 1 kafkaMessage
```

Consumer

```
consumer = do
  let
      kafkaConfig = [("socket.timeout.ms", "50000")]
      topicConfig = [("request.timeout.ms", "50000")]
      partition   = 0
      broker      = "10.77.1.2:9092"
      topic       = "test"
      offset      = 0
      timeoutMs   = 1000
      maxMessages = 100
```

```
withKafkaConsumer kafkaConfig topicConfig
    broker topic partition (KafkaOffset offset) -- KafkaOffsetBeginning, Kafka
    $ \kafka topic -> batchGetter kafka topic partition timeoutMs maxMessages
```

Demo

Books