FOR SMALL BUT ROBUST TEAMS

SMALLLOAD!

FROM GUY WHO WANTED MONADS IN GO

"OVERSIMPLIFICATION IS PREMATURE OPTIMISATION"

NO, 1,000,000 REQ/DAY IS NOT THE HIGHLOAD

YOU ARE NOT A GOOGLE

WE HAVE A LOAD

- hadoop
- HDFS
- Kafka
- Cassandra
- All The AWS

I DON'T KNOW WHAT I'M DOING

- N+1
- overfetching

ONE ANALYTICAL CASE

- a lot of data: clicks, urls etc.
- a lot of dimensions
- a lot of clients

USE YOUR DATABASE, NOT ORM

- postgresql is a very feature rich
- divide tables per client for speed
- sqlalchemy
- aldjemy (don’t fear!)

FIRST PASS

```
from project.utils import to_json
from project.apps.analytics.models import (
    get_customer_data_table,
)
from project.apps.analytics.quieries import (
    create_query,
)



def view(request):
    table = get_customer_data_table(customer.id)
    return to_json(
        create_query(table, request)
    )
```

KNOW YOUR DATABASE

- json
- simplejson
- ujson
- postgresql

POSTGRESQL JSON SAMPLE

```
from project.utils import select_as_json
from project.apps.analytics.models import (
    get_customer_data_table,
)
from project.apps.analytics.quieries import (
    create_query,
)



def view(request):
    table = get_customer_data_table(customer.id)
    return select_as_json(
        create_query(table, request)
    )
```

POSTGRESQL JSON SAMPLE

```
def select_as_json(query):
    if not isinstance(query, Alias):
        query = query.subquery('tojson_aggregate')
    return JSONResult((
        db.session.query(
            cast(
                func.json_agg(text(query.name)),
                Text
            )
        )
        .select_from(query)
        .first()
    )[0])
```

STILL NOT GOOD ENOUGH

MAP'N'REDUCE SAMPLE

```
from itertools import groupby
from collections import namedtuple

MapResult = namedtuple('MapResult', ('key', 'item'))


GET_KEY = lambda x: x.key


def map_reduce(rows, mapper, reducer):
    return (
        reducer(key, (row.item for row in rows))
        for key, rows in groupby(sorted(filter(None,
(mapper(row) for row in rows)), key=GET_KEY), GET_KEY)
    )
```

HEAPQ

- push
- pop
- ...
- groupby(heapq.merge(*streams))

FASTEST DESERIALIZE — DON'T DESERIALIZE

- json
- bson – not real serialize
- messagepack
- cap'n'proto
- flatbuffers

FLATBUFERS SAMPLE

```
import flatbuffers
from my_models import Customer

builder = flatbuffers.Builder(0)
customer = Customer.CustomerStart(builder)
customer.CustomerAddName(
    builder.CreateString('The Big Galaxy Mammal')
)
end = Customer.CustomerEnd(builder)
builder.Finish(end)
```

MMAP

- can be good enough
- use your operating system cache

MMAP SAMPLE

```
import mmap
import flatbuffers
from my_models import Customer


def mmap_all_the_things(filename):
    f = open(filename, 'r')
    buf = mmap.mmap(
        f.fileno(), 0, mmap.MAP_SHARED, mmap.PROT_READ,
    )
    Customer.GetRootAsCustomer(buf, 0)
```

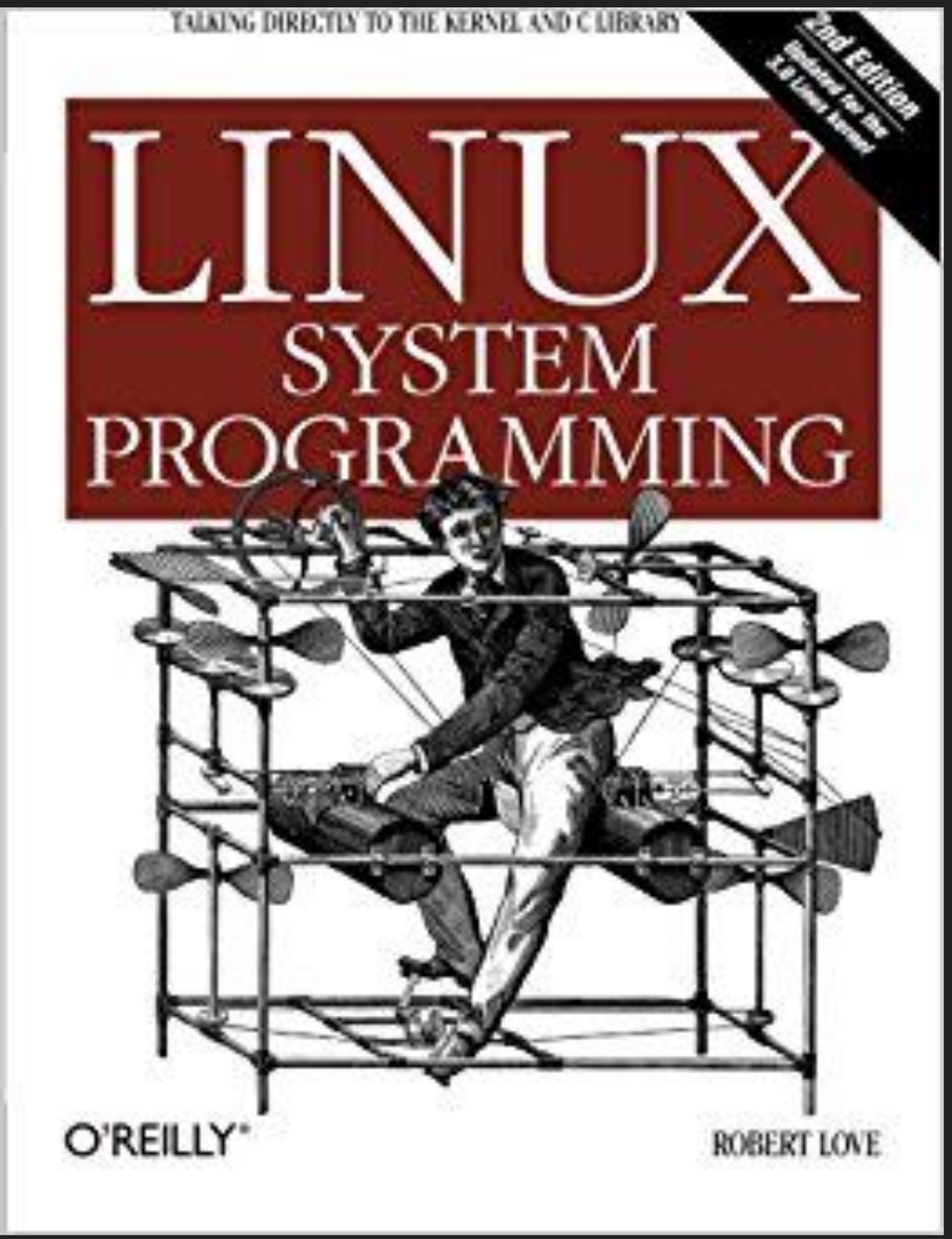
I KNOW ONE AMAZING STORAGE

- append only
- blazingly fast
- key value
- replication
- optional snapshots

FILESYSTEM

MAP AND REDUCE

- wat? you need not hadoop to make map'n'reduce?
- map
- sort
- heap merge
- reduce



THE MUST READ BOOK

LINUX SYSTEM PROGRAMMING

БЫСТРО СОБРАТЬ ВСЁ

- супер быстрая база данных?
- текстовые файлы
- bcpcopy
- COPY FROM

DO YOU KNOW HOW YOUR SYSTEM WORKS?

- what is a setbrk?
- how exec really work?
- truncate
- signals

DATA DEPENDENCIES

- make cascades
- poll dependencies
- hash config

KNOW YOUR NGINX

- sendfile
- auth
- inner redirect

IT AS BUSINESS COMPONENT

- Project Phoenix
- Goldrat

DEVOPS

- ansible
- kubernetes

MAKE IT FAST
PLEASE

FRONTEND

TEXT

WHAT YOU CAN'T SIMPLIFY

- dashboardsssssssss
- 1. smart SQL
- 2. so, we came to the point where map'n'reduce and streaming processing is useful :-P

TEXT

SINGLE TABLE DATABASE VERTICAL SHARDING

- For super fast use ultra thin records table
- Example: users, sessions, video metadata, photo metadata
- Tarantool

IT'S OK IF WE ARE NOT A BANK

- fraud
- banks loose money
- really
- banks loose money, but risks are insured

QUESTIONS?

Me