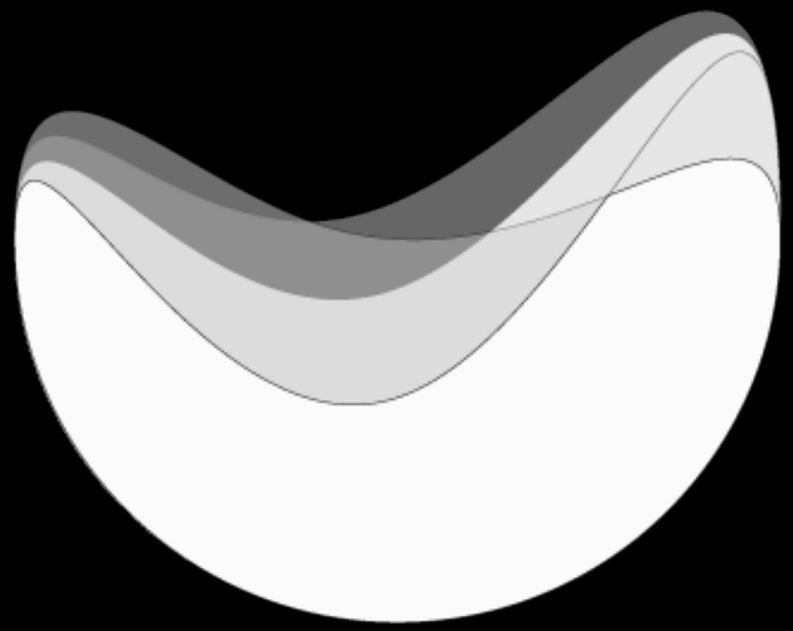
Unavoidable MapReduce

When The Aggregation Framework Isn't Enough

Who I Am

- João Ferreira (@jmnsferreira)
- Developer @ lqd.io



What We Do

- App Personalisation
- Analytics
- Store and Search lots of Data

Our Data Topology

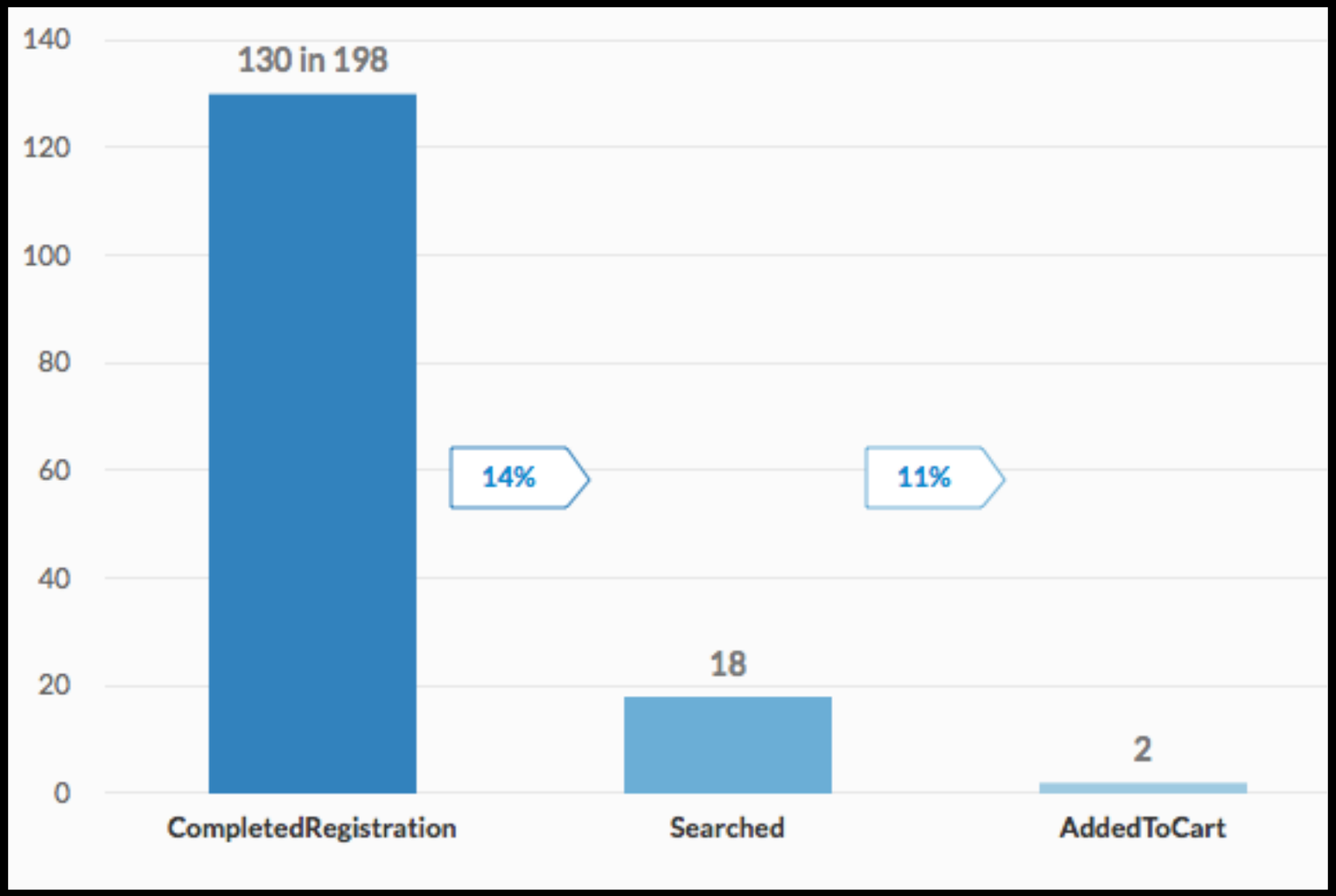
- Administrative logic is in PostgreSQL
- Analytics Data is in MongoDB
    - Two Databases per One App (Prod/Dev)

MongoDB Server Structure

- 4 Replica Sets
    - Replicated (Primary / Secondary)
- 4 Servers

Data Point Structure

```
{
    “user”: {},
    “device”: {},
    “session”: {},
    “event”: {},
    “values”: []
}
```



Funnels

The Use Case

Easy with SQL

- Get all the first step's events;
- **Join** them with all the second step's events;
- Group by session/user;
- Done.

Not So Easy in MongoDB

- Documents depend on each other;
- Aggregation can only operate on one at a time.

"Store your data to fit your needs."

–On StackOverflow, everywhere

For Each Funnel

```
{
   session_id: …,
   user_id: …,
   step_count: 0..n_steps
}
```

Let’s Aggregate

```
[
    { $sort: { timestamp: 1 } },
    { $group: {
        _id: “$session.unique_id”,
        user_id: { $first: “$user.unique_id” },
        events: { $push: “$event.name” }
    } }
]
```

```
{
   _id: <session_id>,
   user_id: <user_id>,
   events: [<event1>, <event2>, …]
}
```

What Now?

- $in is not enough;
- $setEquals ignores order, duplicates;
- $pop is an Update Operator;
- $where does not work in $match;
- We need to run custom code.

Map Reduce

- query
- map
- reduce
- finalize

Query

- { “event.name”: { $in: event_names } }

Map

- Key: { session.unique_id, user.unique_id };
- Value: { event.name, timestamp }

Reduce

1. Sort event names by timestamp;

2. Return array of sorted event names;

```
{
  _id: {
    session_id: …,
    user_id: …
  },
  events: [<event1>, <event2>, …]
}
```

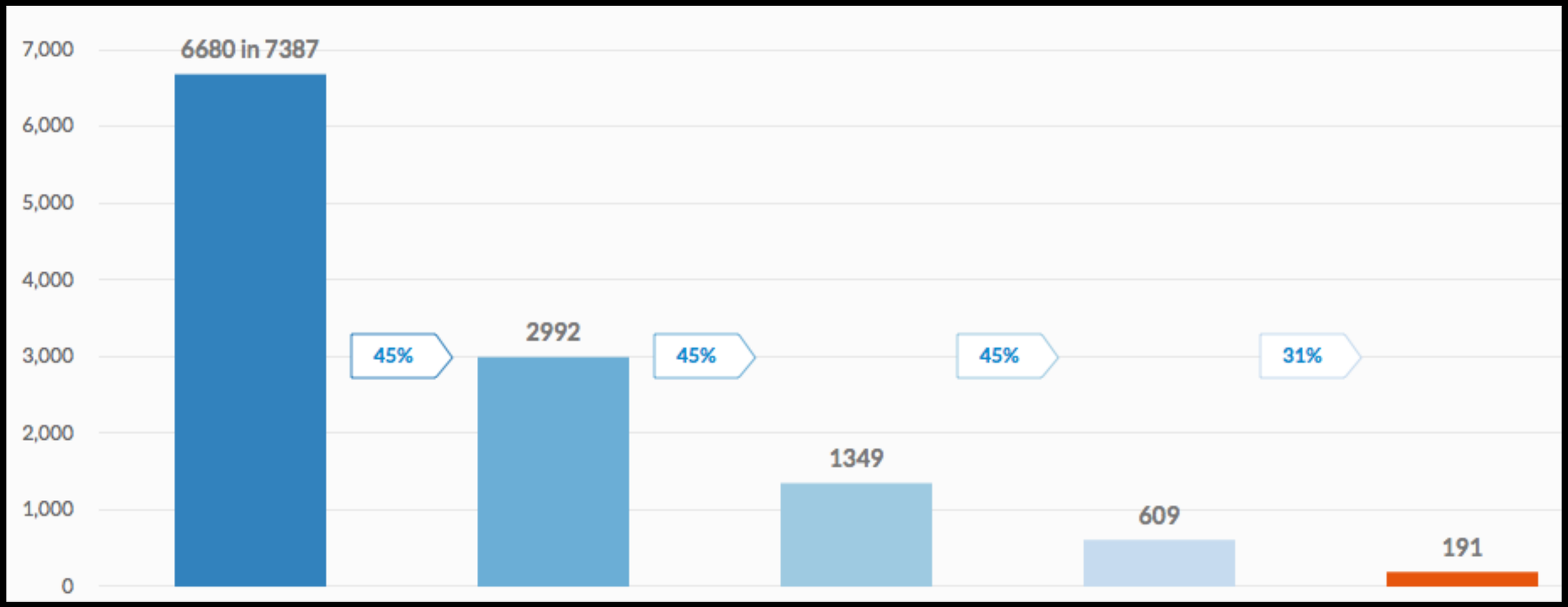
Finalize

1. Loop through event names array;
2. Find largest sub array of scope.events;
3. Get the sub array's size;
4. Return: How many steps this user followed.

```
{
  _id: {
    session_id: …,
    user_id: …
  },
  step_count: 0..n_steps
}
```

Sessions in First Step

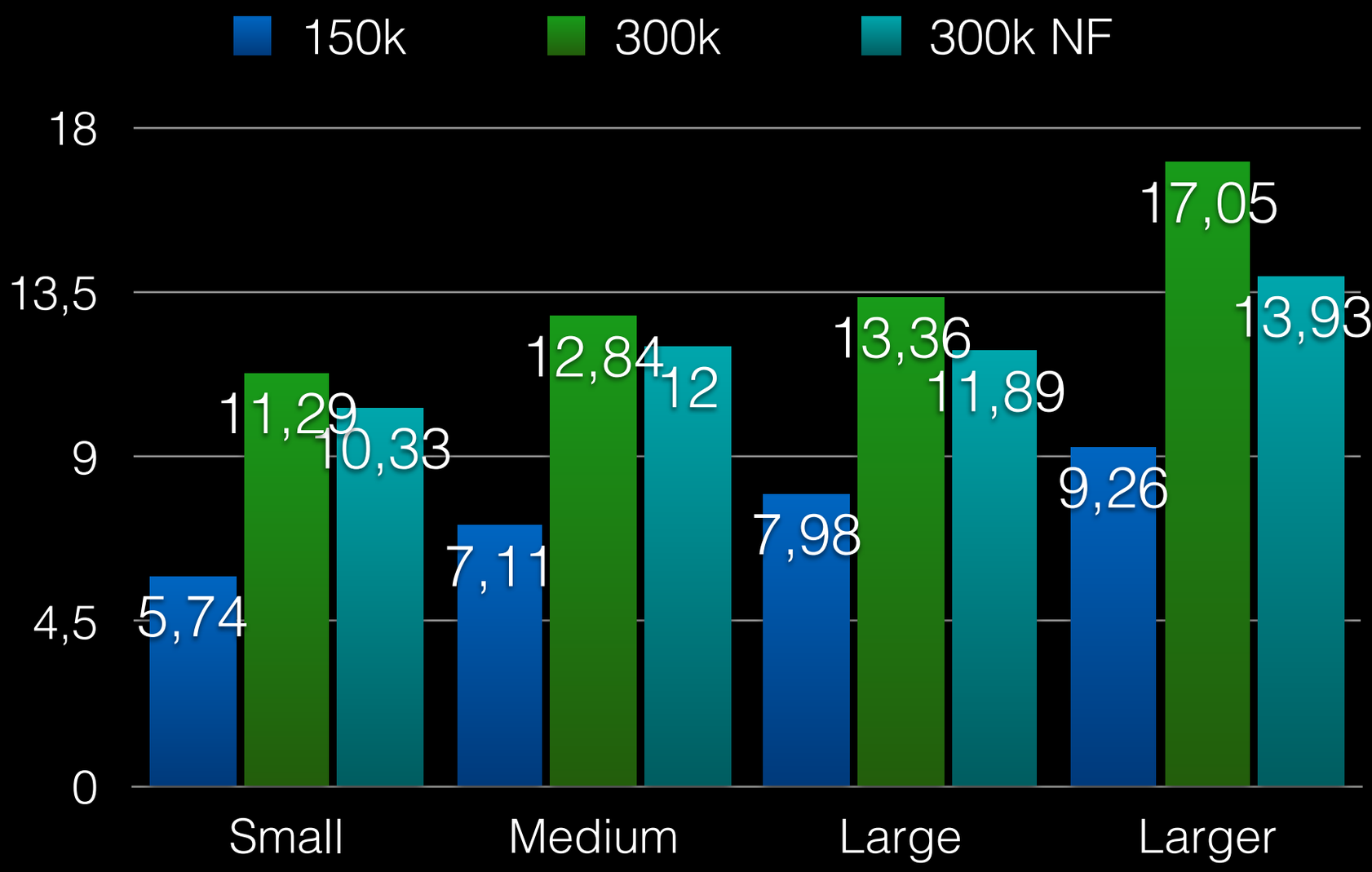
- `find({ step_counts: { $gte: 1 } })`



Benchmarks

150k/300k DataPoints

Benchmarks


150k
300k
300k NF
18
13,5
9
4,5
0
5,74
11,29
10,33
7,11
12,84
12
7,98
13,36
11,89
9,26
17,05
13,93
Small
Medium
Large
Larger

Not Very Fast

And doesn’t scale very well…

Optimize!

- Our Map Reduce stage is actually the old aggregation;
- Replace it;
- Query data with $where.

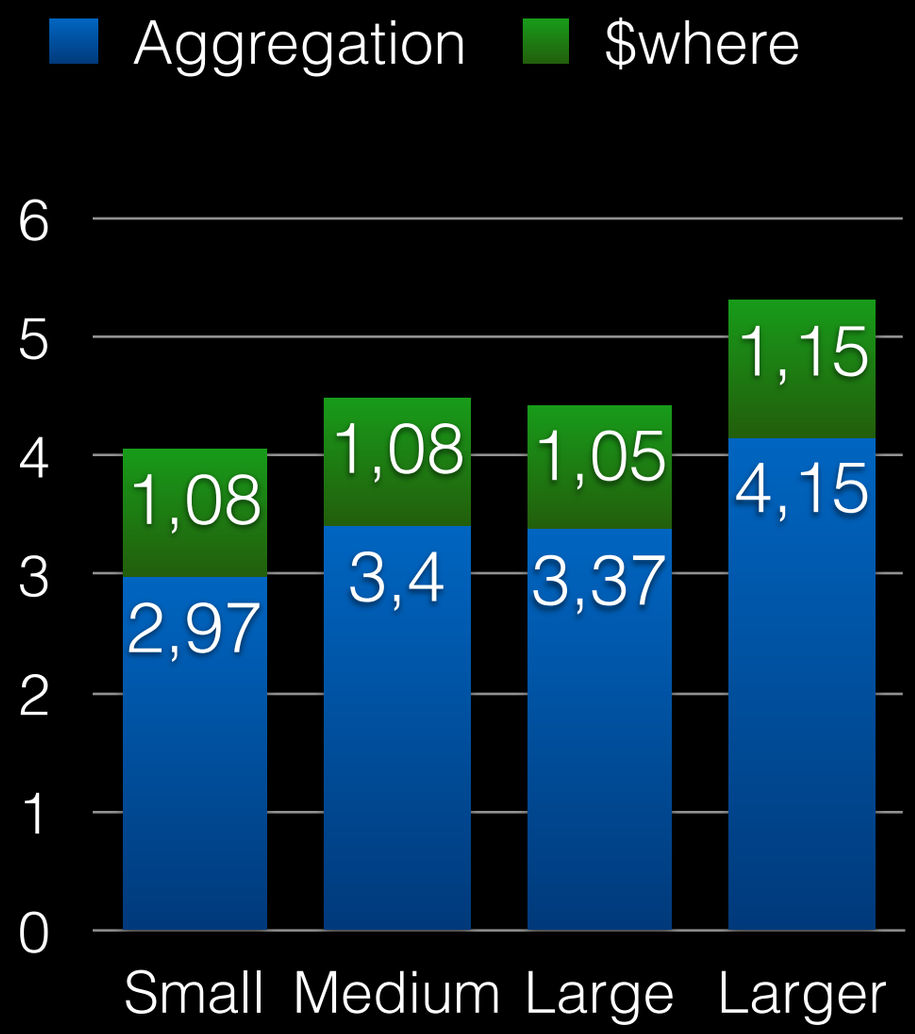
Refactor Aggregation

```
[
    { $sort: { timestamp: 1 } },
    { $group: {
        _id: “$session.unique_id”,
        user_id: { $first: “$user.unique_id” },
        events: { $push: “$event.name” }
    } },
    { $out: “funnel_agg” }
]
```

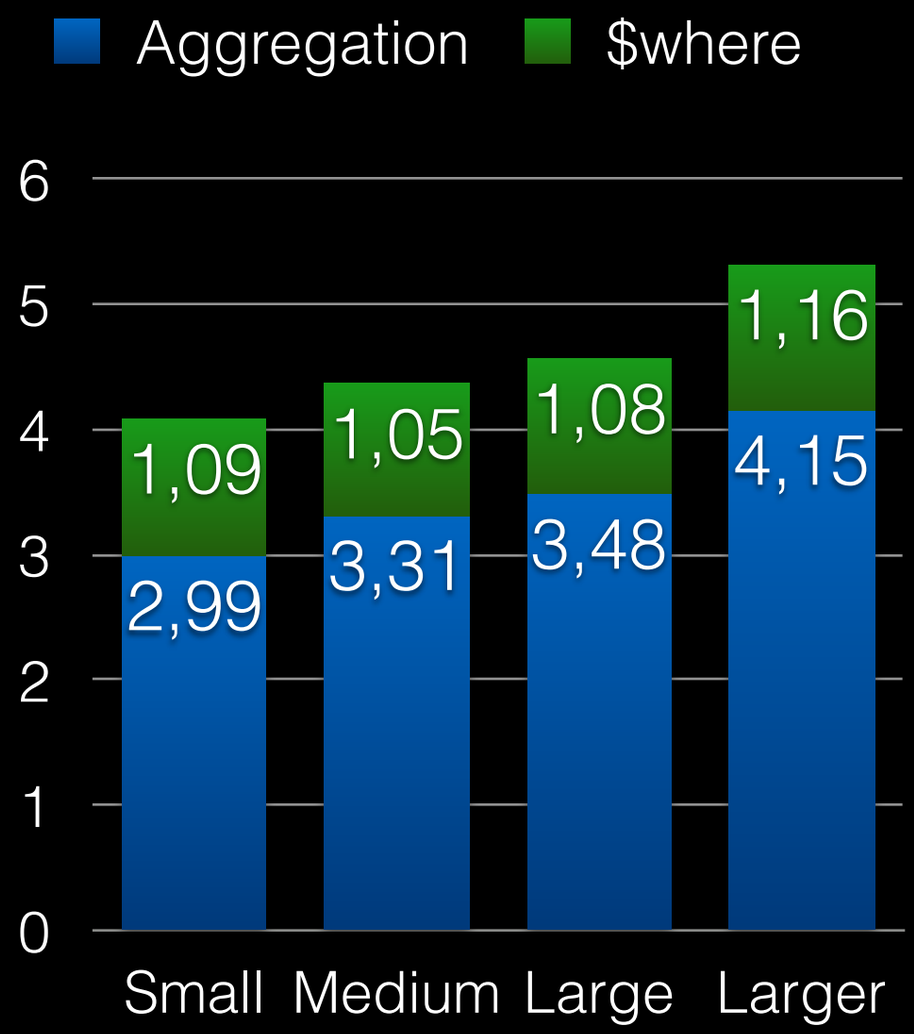
$where

```
db.funnel_agg.distinct("user_id",
   { $where: function () {
     // loop events array
     // search subarray of step events
     // return true if found
   }}, { events: 0 });
```

Benchmarks



150k



300k

Better!

And much less affected by Data Point counts

But…

- Funnels are about:
  - **Counts**;
  - **Conversions**;
- $where gives us **user IDs.**

Map Reduce!

- Map:
  - Key: user_id;
  - Value: calculateStep(events);
- Reduce:
  - Return biggest step for key.

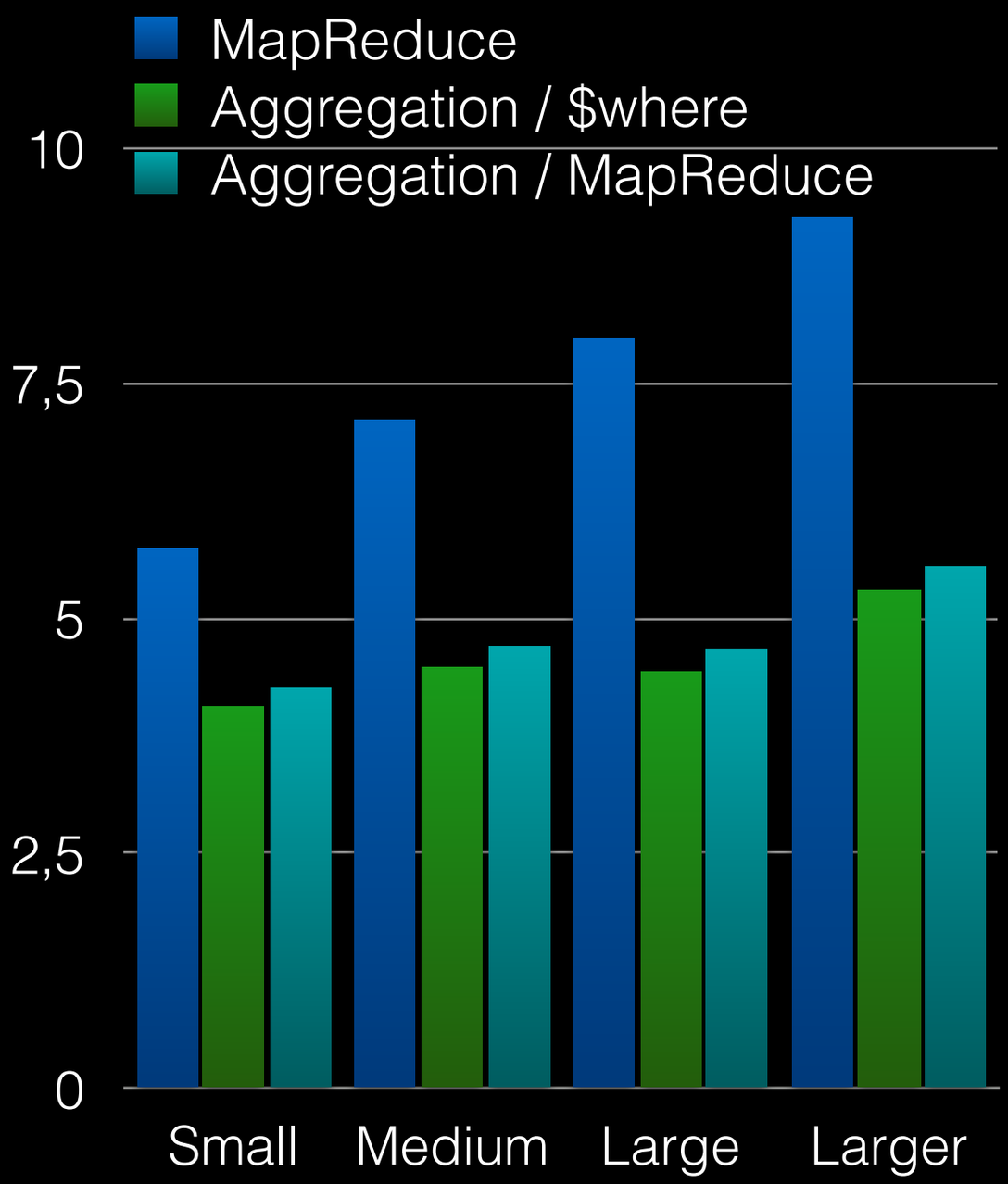
Our Final Document

```
{
   user_id: …,

   step: 0-n_step

}
```

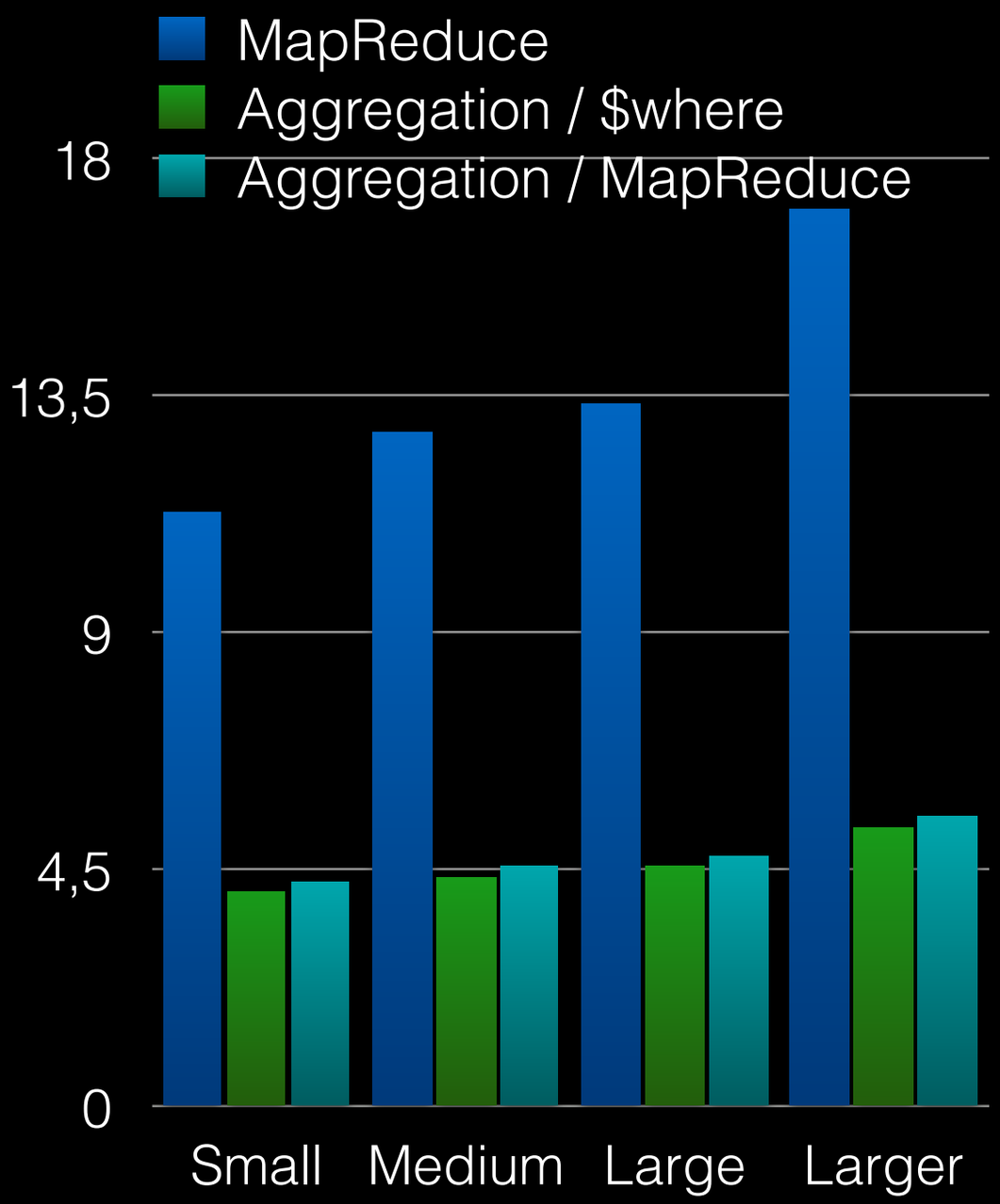
Advantages

- Counting users in each step is very fast;
- No unnecessary IDs reach the server;
- Events are aggregated only once.

Benchmarks



150k



300k

For The Future

- First aggregation global & permanent
  - Must use MapReduce or update in real-time;
- First aggregation sharded
  - Must use MapReduce or update in real-time;
- Index final aggregation on step counts
  - Probably minimal benefits.

Conclusion

Aggregation

- Very fast (C);
- Can use Indexes;
- Lacking in operators;
- No custom code;
- No sharded output;
- No reusable output;
- Use it!

Map Reduce

- Slow (JavaScript);
- Index only in pre-queries;
- Any operation that can be coded;
- Can shard output;
- Can merge output;
- Heavily affected by Document size;
- Use it, sparingly.