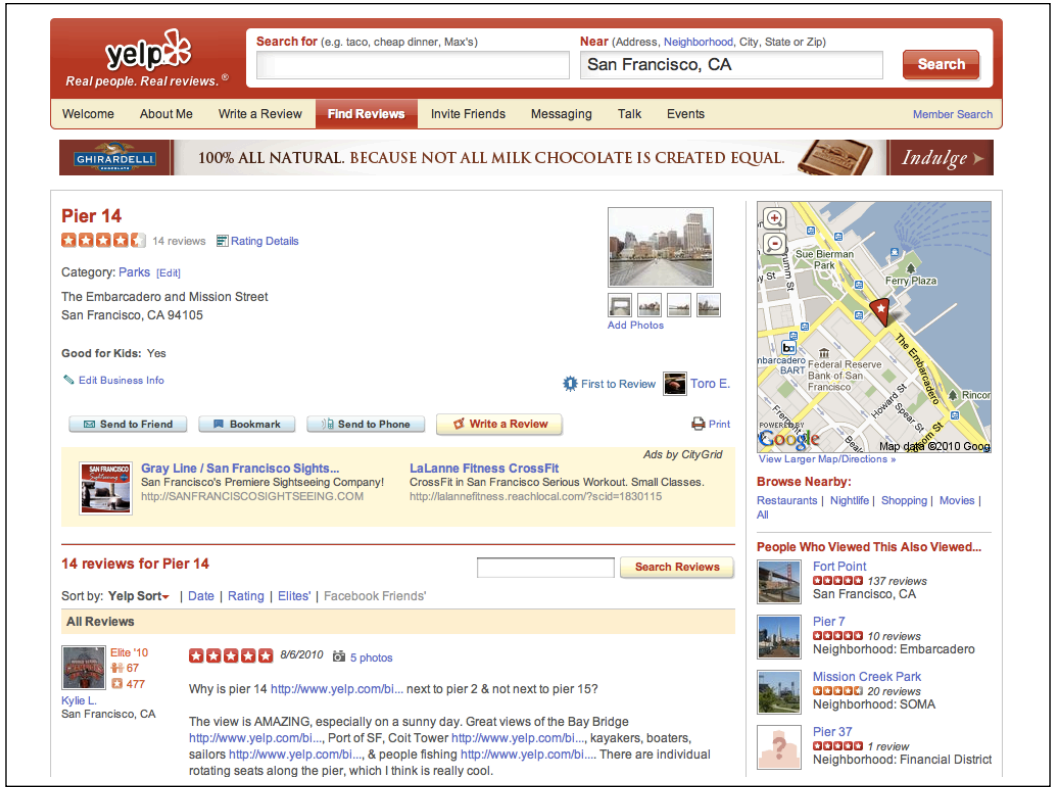
mrjob

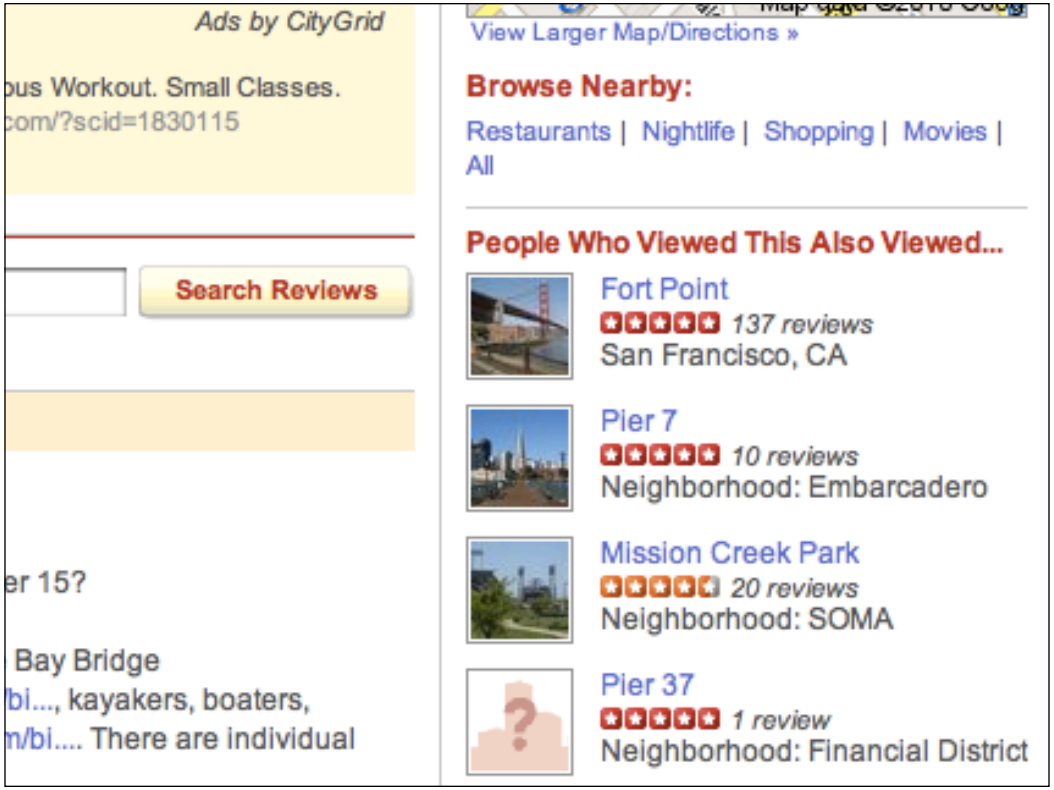
Jimmy Retzlaff
Yelp

What's the Problem?

- Yelp produces ~100GB of logs per day
- Many features rely on log analysis
- Computers are cheap, but leveraging many of them for a single task is hard



pier 14 is cool!



how do we do this? walk through our logs and collect stats about pairs of businesses viewed in the same browsing session – but it's 100GB!


Go, Dog. Go!

go
dog
go

go → counter() → (g, 1), (o, 1)
dog → counter() → (d, 1), (o, 1), (g, 1)
go → counter() → (g, 1), (o, 1)

(g, 1), (o, 1)
(d, 1), (o, 1), (g, 1)
(g, 1), (o, 1)

→ sort →

(d, 1)
(g, 1)
(g, 1)
(g, 1)
(o, 1)
(o, 1)
(o, 1)

(d, 1)

(g, 1)
(g, 1)
(g, 1)

(o, 1)
(o, 1)
(o, 1)

(d, 1) ⟶ summarizer(d, [1]) ⟶ (d, 1)

(g, 1)
(g, 1) ⟶ summarizer(g, [1, 1, 1]) ⟶ (g, 3)
(g, 1)

(o, 1)
(o, 1) ⟶ summarizer(o, [1, 1, 1]) ⟶ (o, 3)
(o, 1)

hey, this could be run on multiple computers!

go → counter() → (g, 1), (o, 1)
dog → counter() → (d, 1), (o, 1), (g, 1)
go → counter() → (g, 1), (o, 1)

go → counter() → (g, 1), (o, 1)

dog → counter() → (d, 1), (o, 1), (g, 1)

go → counter() → (g, 1), (o, 1)

how about a different name? mapper

go → mapper() → (g, 1), (o, 1)

dog → mapper() → (d, 1), (o, 1), (g, 1)

go → mapper() → (g, 1), (o, 1)

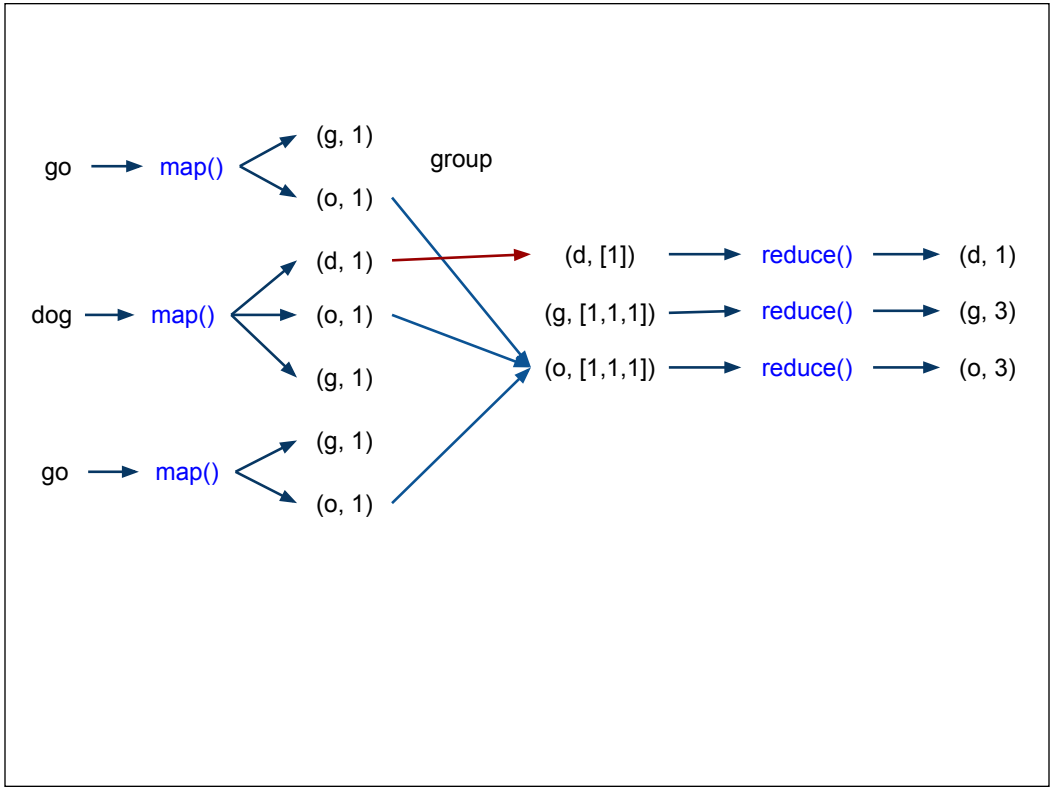
(d, 1) ⟶ summarizer(d, [1]) ⟶ (d, 1)

(g, 1)
(g, 1) ⟶ summarizer(g, [1, 1, 1]) ⟶ (g, 3)
(g, 1)

(o, 1)
(o, 1) ⟶ summarizer(o, [1, 1, 1]) ⟶ (o, 3)
(o, 1)

how about a different name? reducer

(d, 1)	$\longrightarrow$	reducer(d, [1])	$\longrightarrow$	(d, 1)
(g, 1) (g, 1) (g, 1)	$\longrightarrow$	reducer(g, [1, 1, 1])	$\longrightarrow$	(g, 3)
(o, 1) (o, 1) (o, 1)	$\longrightarrow$	reducer(o, [1, 1, 1])	$\longrightarrow$	(o, 3)


go
map()
(g, 1)
(o, 1)
dog
map()
(d, 1)
(o, 1)
(g, 1)
go
map()
(g, 1)
(o, 1)
group
(d, [1])
(g, [1,1,1])
(o, [1,1,1])
reduce()
reduce()
reduce()
(d, 1)
(g, 3)
(o, 3)

Yelp's mrjob

- sign up for Amazon Elastic MapReduce
- write a dozen lines of Python
- run on many Amazon computers

mrjob takes care of the plumbing – it launches the Amazon servers for you, let's you install your code on them, uploads your data, kicks off your job, monitors it, brings the results back, and shuts the servers down

```python
from mrjob.job import MRJob

class MRCharacterCount(MRJob):
    def mapper(self, _, text):
        for c in text:
            yield c, 1

    def reducer(self, c, counts):
        yield c, sum(counts)

if __name__ == '__main__':
    MRCharacterCount.run()
```

```python
from mrjob.job import MRJob

class MRWordCount(MRJob):
    def mapper(self, _, text):
        for word in text.split():
            yield word, 1

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCount.run()
```

click through rate = ads clicked / ads shown

show ad_event_log in BBEdit

Exercise 1: Run an mrjob!

http://snipurl.com/mr_word_count

$ python mr_word_count.py mr_word_count.py

```
from mrjob.job import MRJob

class MRWordCount(MRJob):
    def mapper(self, _, text):
        for word in text.split():
            yield word, 1

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCount.run()
```

Exercise 2: Write an mrjob!

Determine the number of times every two characters appear in order in the same word in some text.

Example: go dog go

```
g,o   2
d,o   1
o,g   1
```

Extra credit: After doing the first part, modify it so that order does not matter.... `g,o  3;  d,o  1`

Where can I get cool data to play with?

Where can I get cool data to play with?

Yelp is hiring!

yelp.com/jobs