Computer vision at scale with Hadoop and Storm

Presented by: Huy Nguyen @huyng

Outline

- Computer vision at 10,000 feet
- Hadoop training system
- Storm continuous stream processing
- Hadoop batch processing
- Q&A

YAHOO!

500+ million images are uploaded to internet photo services every day

This upload rate is expected to double by the end of 2014

- Mary Meeker (KPCB)
2013 Internet Trends

hadoop YAHOO!

How can we help Flickr users better **organize**, **search**, and **browse** their photo collection?

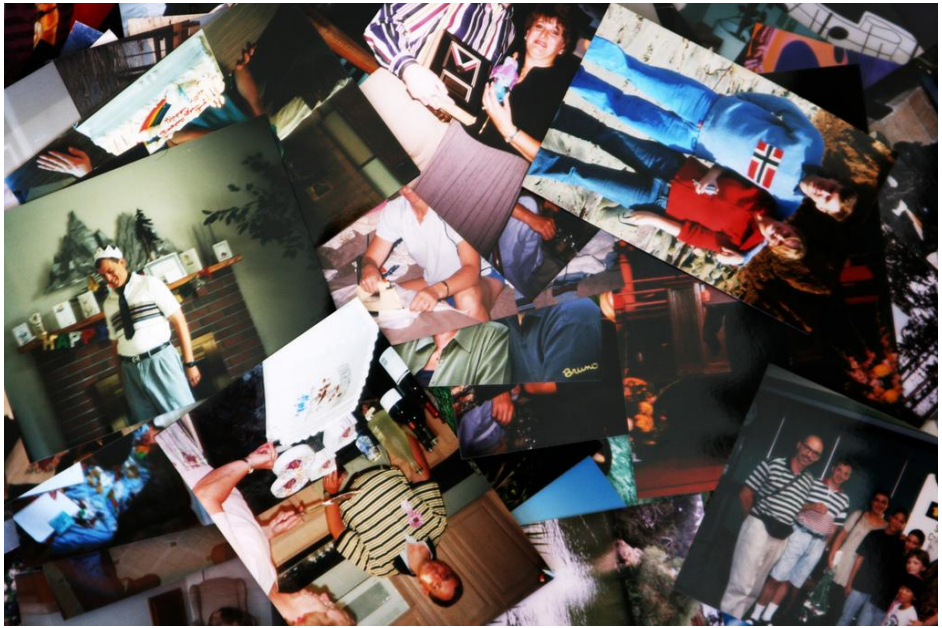
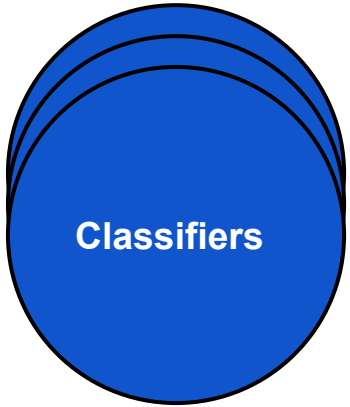


photo credit: Quinn Dombrowski

hadoop YAHOO!

What is AutoTagging?



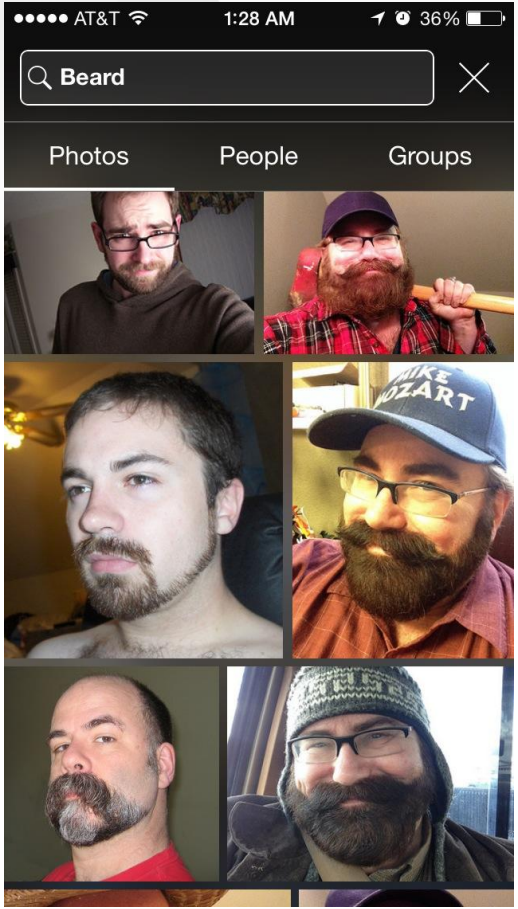
Class	Probability
Flowers	0.98
Outdoors	0.95
Cat	0.001
Grass	0.6

Any photo you upload to Flickr is now automatically classified using computer vision software

hadoop YAHOO!

Demo

YAHOO!



Auto Tags Feeding Search

- Flowers
- Buildings
- Outdoors
- Beach
- Beards
- and more!

hadoop YAHOO!

Our Goals

- Train thousands of classifiers
- Classify millions of photos per day
- Compute auto tags on backlog of billions of photos
- Enable reprocessing bi-weekly and every quarter
- Do it all within 90 days of being acquired!

YAHOO!

Our system for training classifiers

- High level overview of image classification
- Hadoop workflow

YAHOO!

Classification in a nutshell

How do we define a classifier?

A classifier is simply the **line** "z = wx - b" which divides positive and negative examples with the largest margin. Entirely defined by **w** and **b**

Training is simply adjusting **w** and **b** to find the line with the largest margin dividing negative and positive example data

Classification is done by computing **z**, given some data point **x**

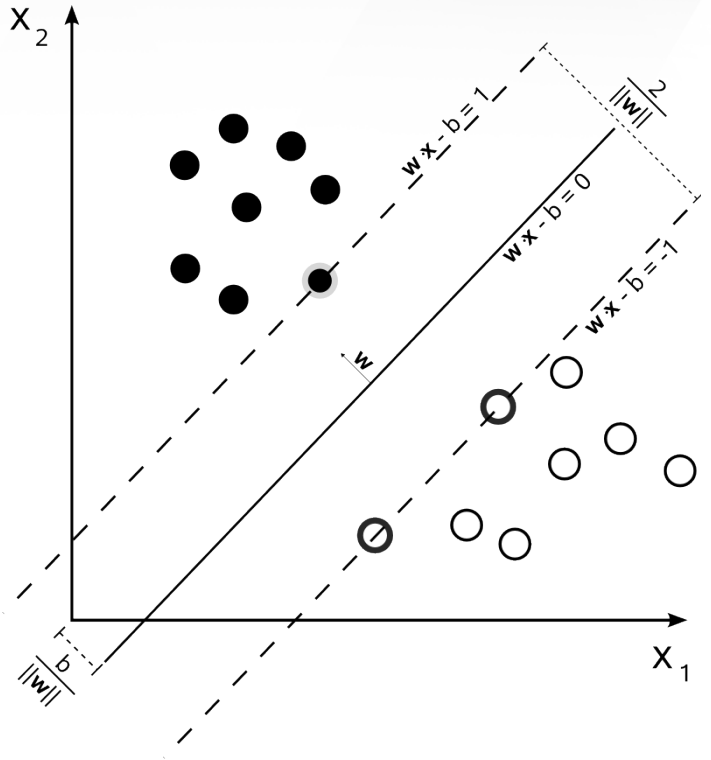
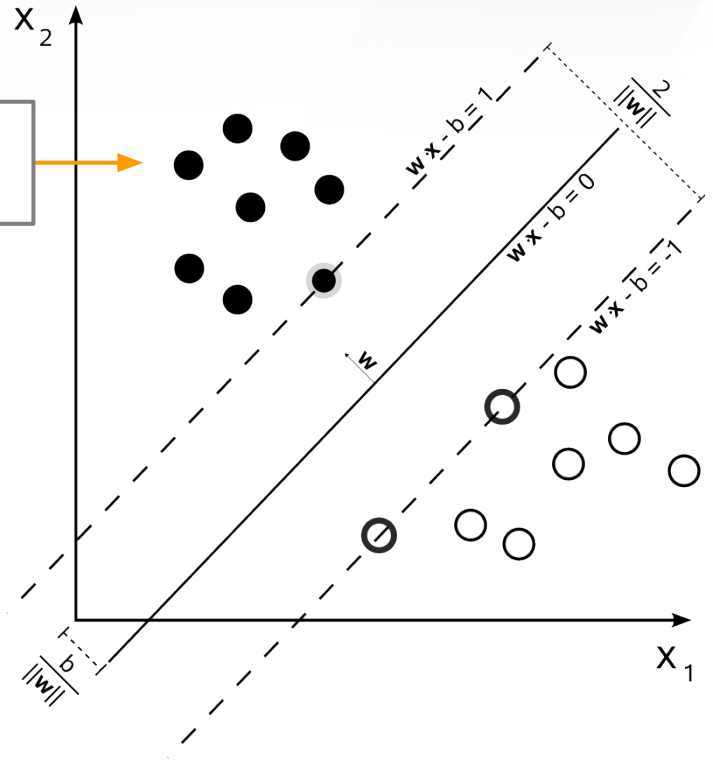


Image classification in a nutshell

compute feature

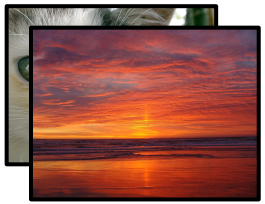


To classify images we need to convert raw pixels into a “feature” to plug into **x**

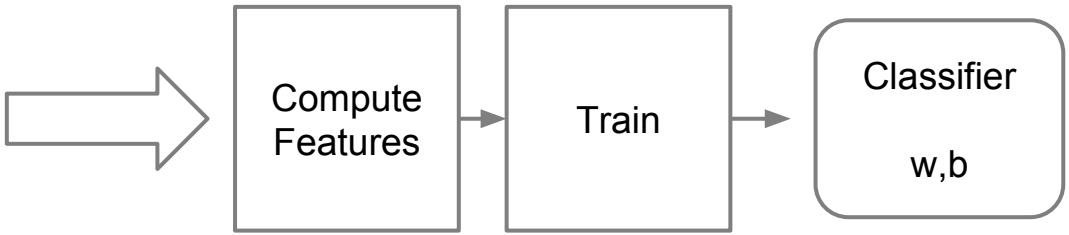
hadoop YAHOO!

How are image classifiers made?

Class Name



Negative Examples
(approx. 100K)

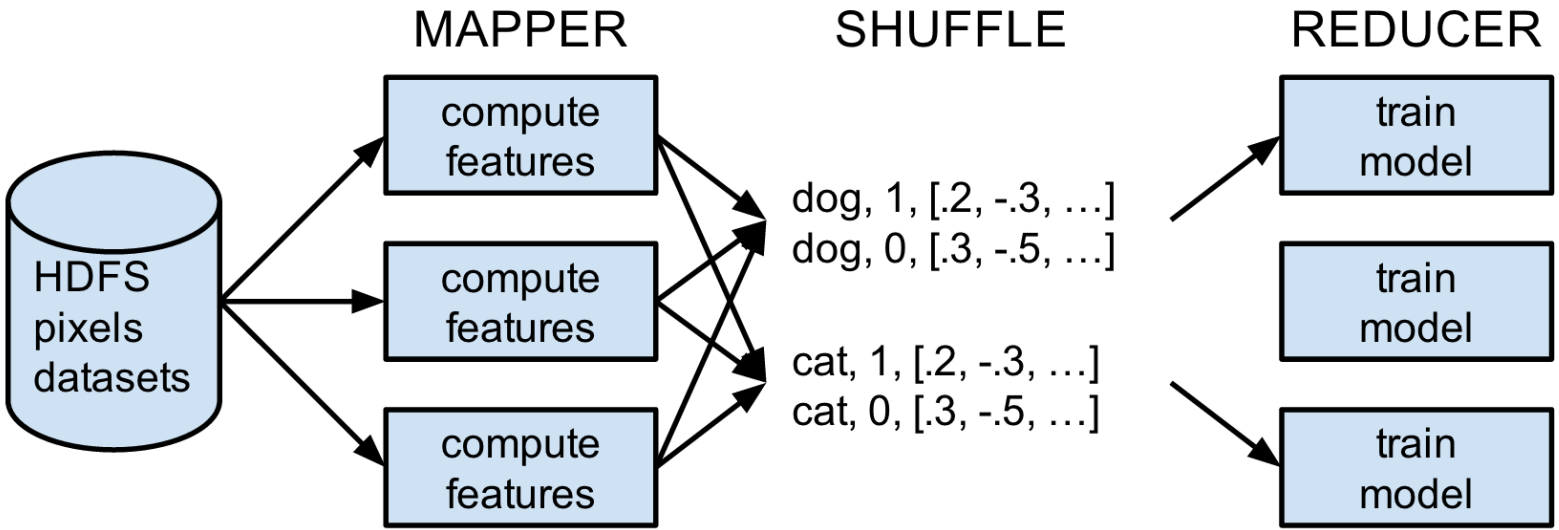


Positive Examples
(approx. 10K)

Repeat a few thousand times for each class

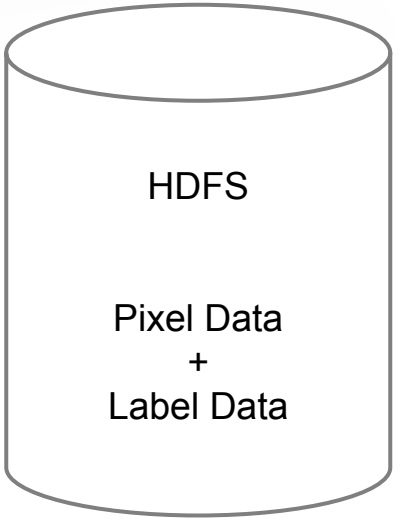
hadoop YAHOO!

Hadoop Classifier Training



hadoop YAHOO!

Hadoop Classifier Training



Pixel Data

1	<base64>
2	<base64>
3	<base64>
4	<base64>
.	
.	
n	<base64>

photo_id, pixel_data

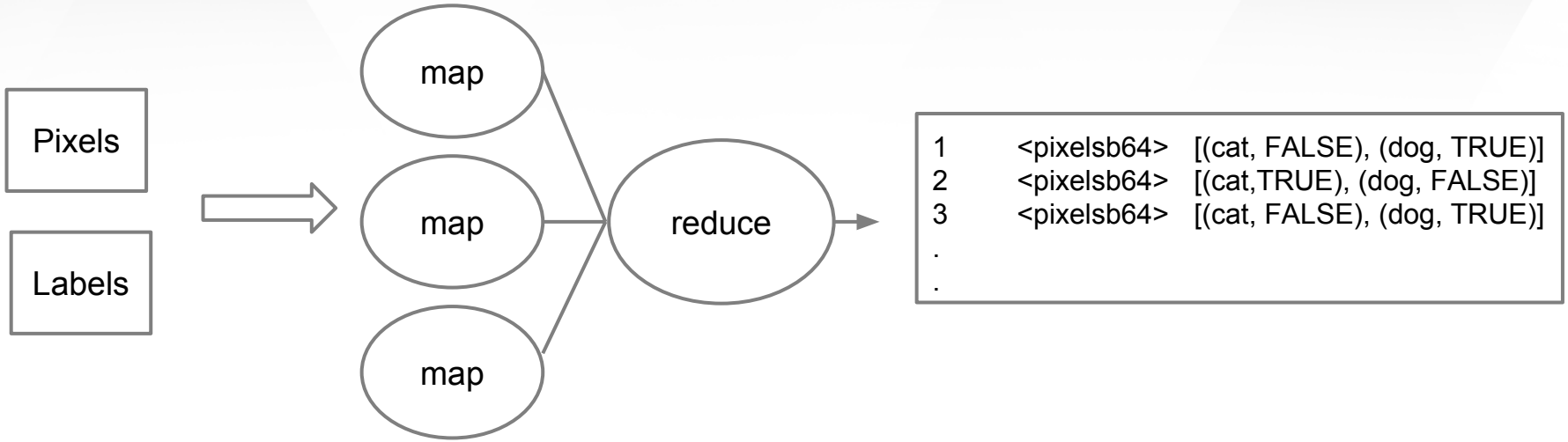
Label Data

dog	1	TRUE
dog	2	FALSE
dog	3	TRUE
cat	1	FALSE
cat	2	TRUE
cat	3	FALSE
.		
.		

class, photo_id, label

hadoop YAHOO!

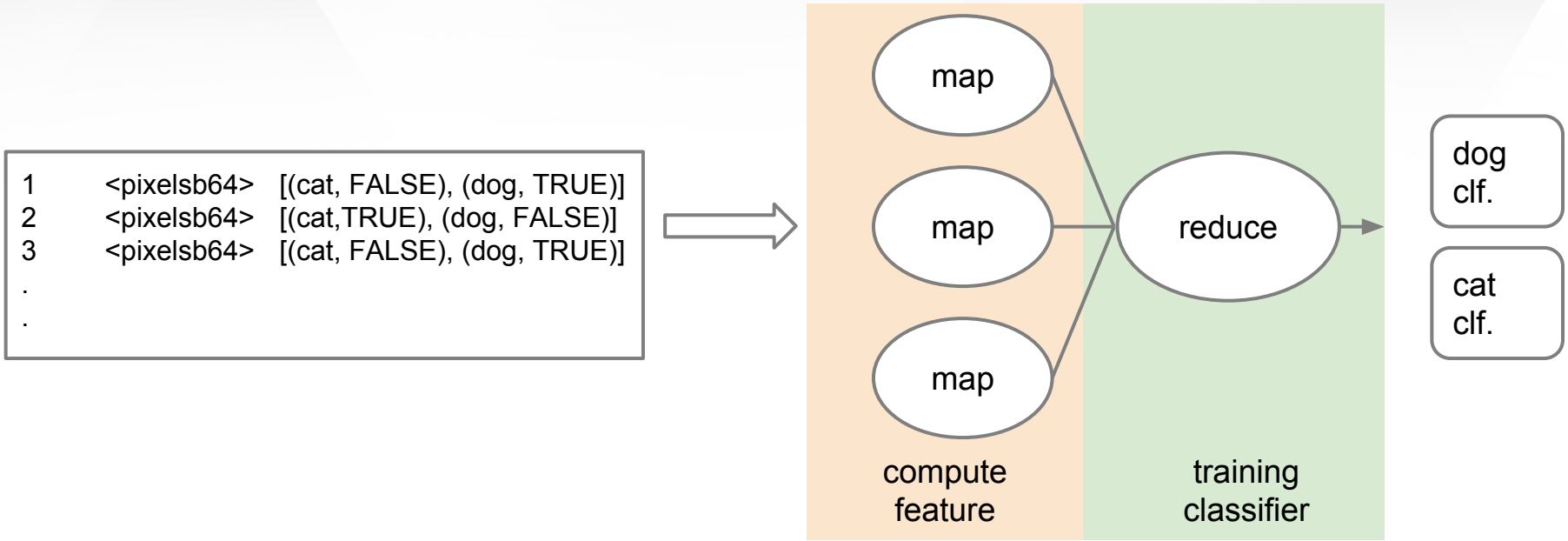
Hadoop Classifier Training - JOIN



- mappers emit (photo_id, pixels) **OR** (photo_id, class, label)
- reducers, takes advantage of sorting to emit (photo_id, pixels, json(label data))

hadoop YAHOO!

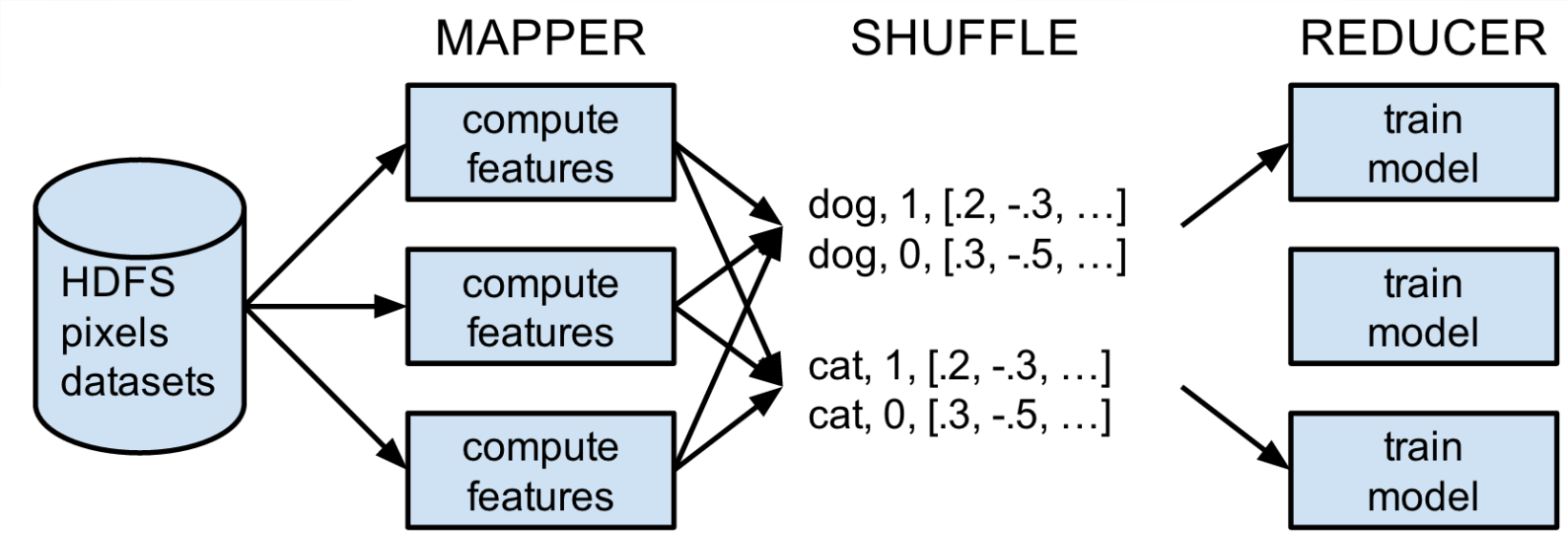
Hadoop Classifier Training - TRAIN



- mappers compute features, emits a (class, feat64, label) **for each class image associated with image**
- reducers, takes advantage of sorting to train. Emits (class, w, b)

hadoop YAHOO!

Hadoop Classifier Training



- Workflow coordinated with OOZIE

hadoop YAHOO!

Hadoop Classifier Training

- 10,000 mappers
- 1,000 reducers
- A few thousand classifiers in less than 6 hours
- Replaced OpenMPI based system
  - Easier deployment
  - Access to more hardware
  - Much simpler code

YAHOO!

Area of Improvements

- Reducers have all training data in memory
  - A potential bottleneck if we want to expand training set for any given set of classifiers
  - Known path to avoiding this bottleneck
- Incrementally update models as new data arrives

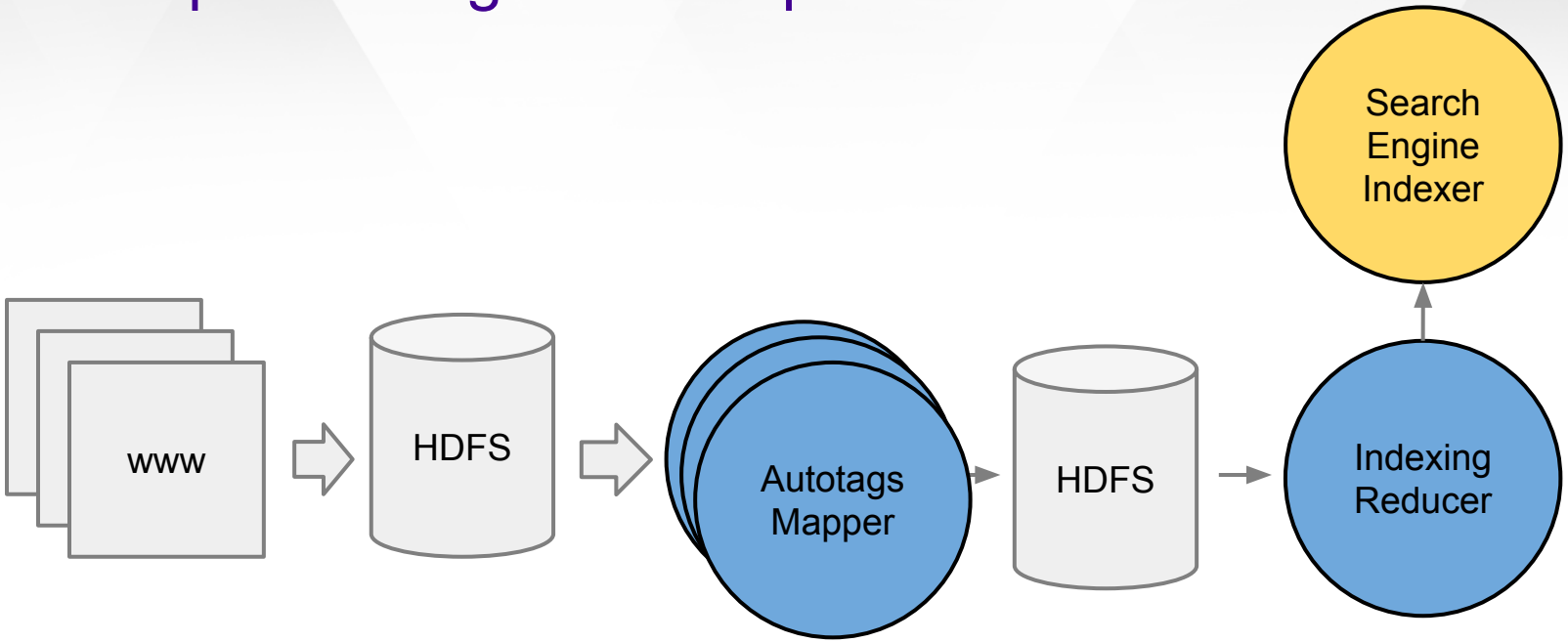
YAHOO!

Our system for tagging the flickr corpus

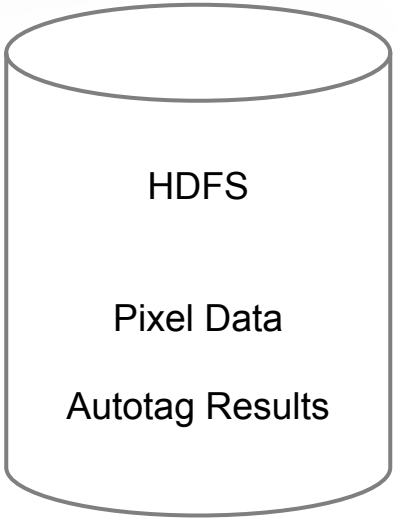
- Batch processing with Hadoop
- Stream processing with Storm

YAHOO!

Batch processing in Hadoop



Hadoop Auto tagging



Pixel Data

1	<base64>
2	<base64>
3	<base64>
4	<base64>
.	
.	
n	<base64>

photo_id, pixel_data

Autotag Results

1	{ “cat”: .98, “dog”: 0.3 }
2	{ “cat”: .97, “dog”: 0.2 }
3	{ “cat”: .2, “dog”: 0.3 }
4	{ “cat”: .5, “dog”: 0.9 }
.	
.	
n	{ “cat”: .90, “dog”: 0.23 }

photo_id, results

hadoop YAHOO!

Hadoop classification performance

- 10,000 Mappers
- Processed in about 1 week entire corpus

YAHOO!

Main challenge: packaging our code

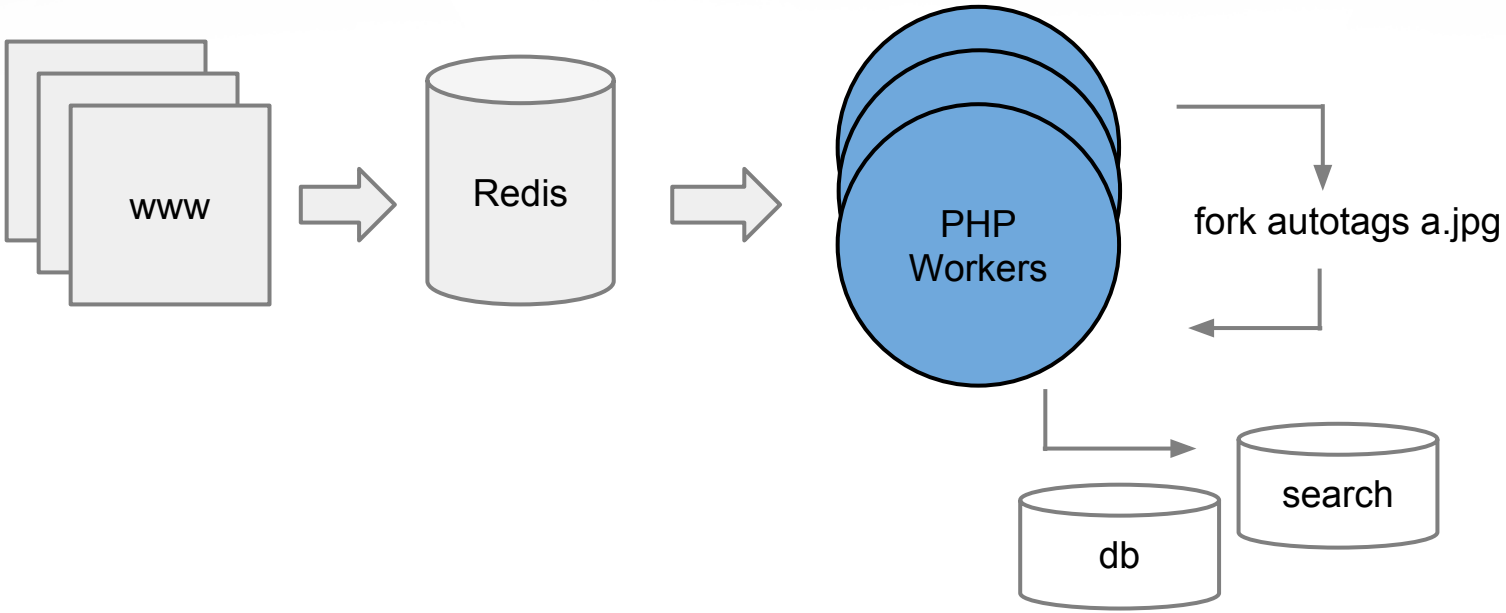
- Heterogenous Python & C++ codebase
- How do we get all of the shared library dependencies into a heterogenous system distributed across many machines?

Main challenge: packaging our code

- tried: pyinstaller, didn't work

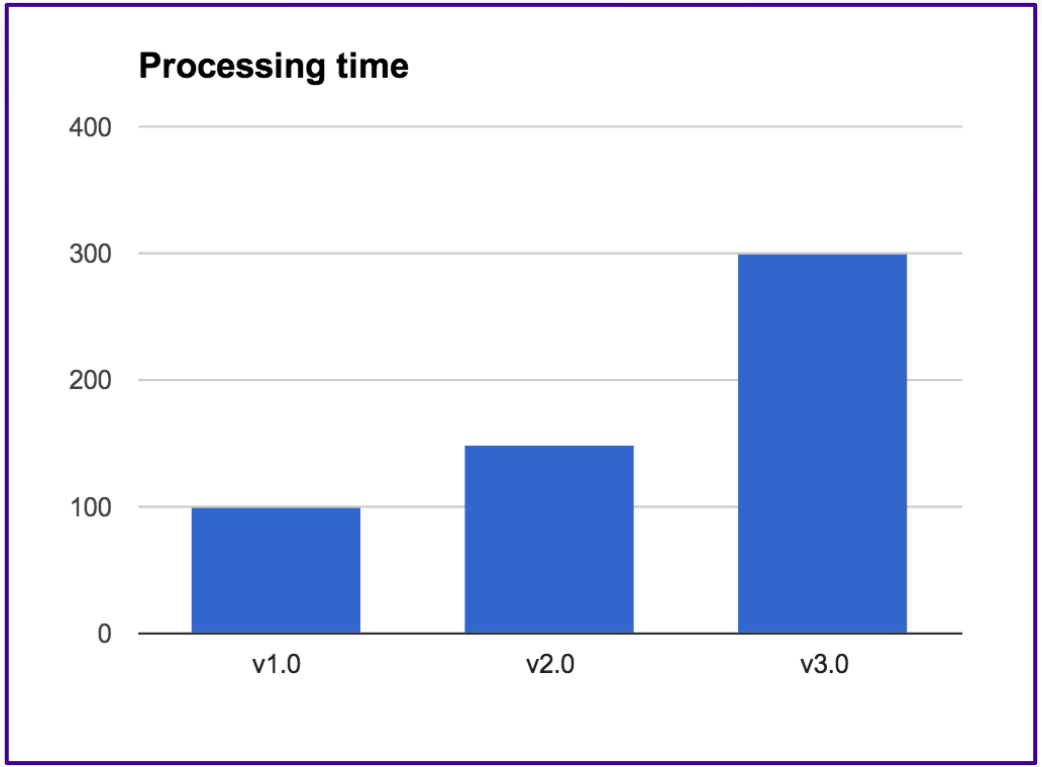
- rolled our own solution using strace
  - /autotags_package
    - /lib
    - /lib/python2.7/site-packages
    - /bin
    - /app

Stream processing (our first attempt)



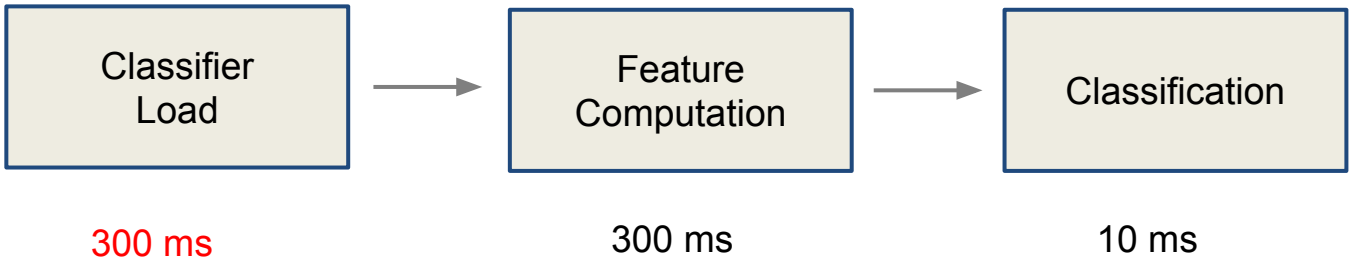
hadoop YAHOO!

Why we started looking into Storm



hadoop YAHOO!

Why we started looking into Storm



Classifier loading accounted for
49% of total processing time

Solutions we evaluated at the time

- Mem-mapping model
- Batching up operations
- **Daemonize our autotagging code**

Enter
Storm

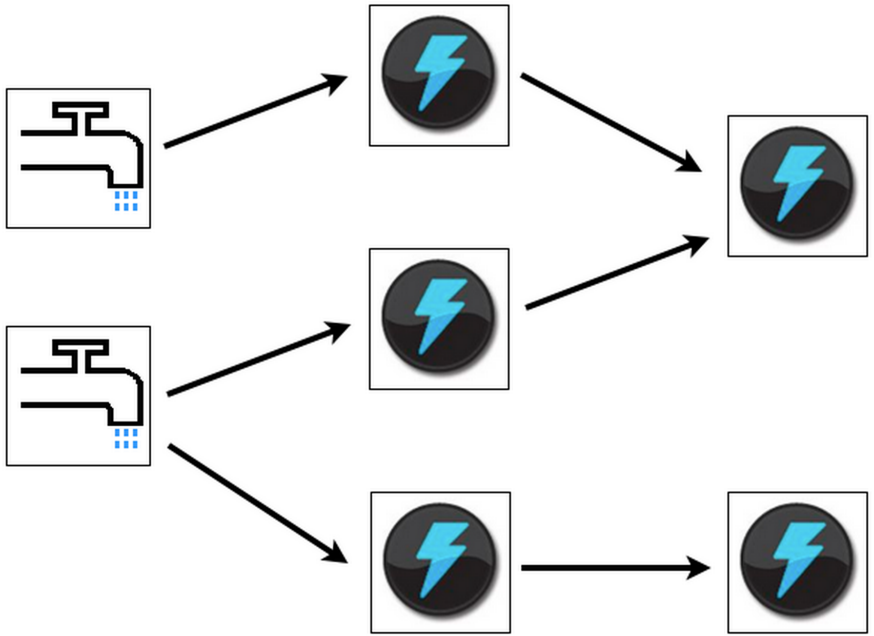
YAHOO!

	Computational Primitives	Use case
Hadoop	Map & Reduce	Batch Processing
Storm	Spout & Bolts	Stream Processing

YAHOO!

Storm computational framework

- Spouts
  - The source of your data stream. Spouts “emit” tuples
- Tuples
  - An atomic unit of work
- Bolts
  - Small programs that processes your tuple stream
- Topology
  - A graph of bolts and spouts



hadoop YAHOO!

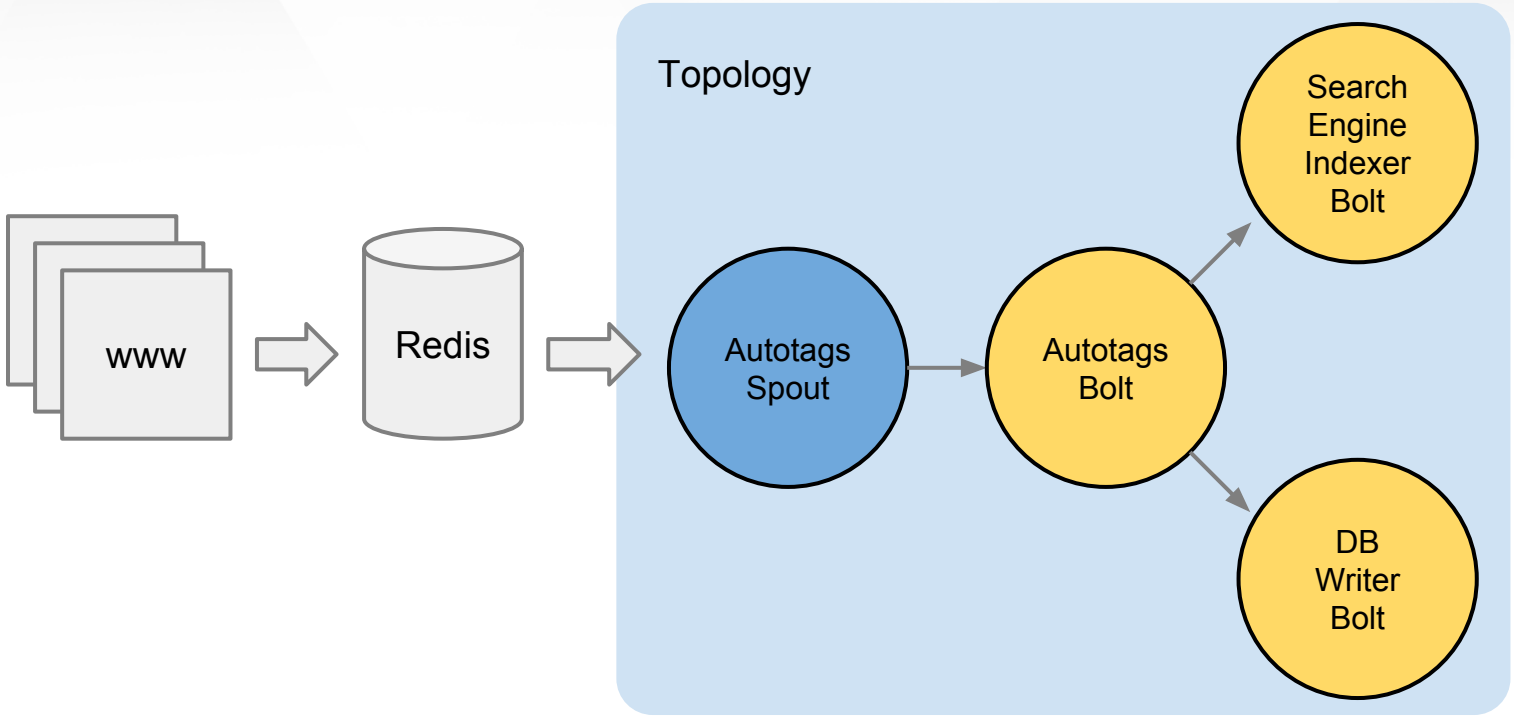
Defining a topology

```
TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("images", new ImageSpout(), 10);

builder.setBolt("autotag", new AutoTagBolt(), 3)
       .shuffleGrouping("images")

builder.setBolt("dbwriter", new DBWriterBolt(), 2)
       .shuffleGrouping("autotag")
```

Defining a bolt

```
class Autotag(storm.BasicBolt):

    def __init__(self):

        self.classifier.load()


    def process(self, tuple):

        <process tuple here>
```

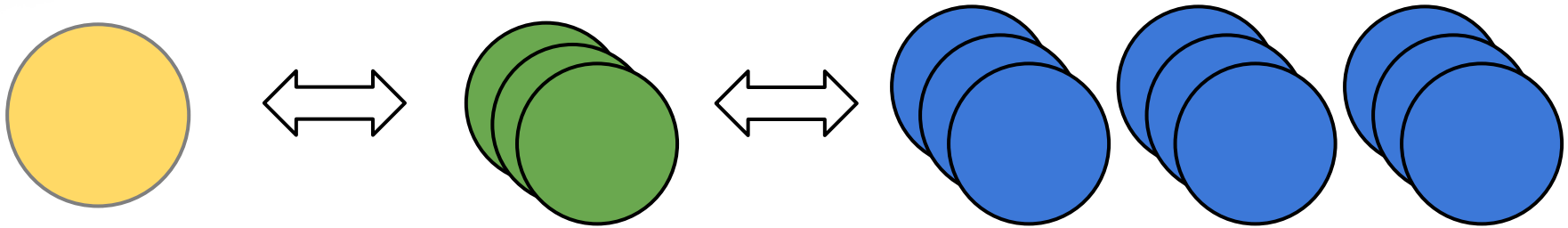
Our final solution



hadoop YAHOO!

Storm architecture

Spouts and bolts live here



Nimbus

(1 node)

Zookeepers

(3 nodes)

Supervisors

(10 nodes)

hadoop YAHOO!

Fault tolerant against node failures

Nimbus server dies

- no new jobs can be submitted
- current jobs stay running
- simply restart

Supervisor node dies

- All unprocessed jobs for that node get reassigned
- Supervisor restarted

Zookeeper node dies

- Restart node, no data loss as long as not too many die at once

YAHOO!

storm commands

```
storm jar topology-jar-path class
```

YAHOO!

storm commands

```
storm list
```

YAHOO!

storm commands

```
storm kill topology-name
```

YAHOO!

storm commands

```
storm deactivate topology-jar-path class
```

YAHOO!

storm commands

```
storm activate topology-name
```

YAHOO!

Is Storm just another queuing system?

YAHOO!

Yes, and more

- A framework for separating application design from scaling concerns
- Has a great deployment story
  - atomic start, kill, deactivate for free
  - no more rsync
- No single point of failure
- Wide industry adoption
- Is programming language agnostic

YAHOO!

Try storm out!

https://github.com/apache/incubator-storm/tree/master/examples/storm-starter

https://github.com/nathanmarz/storm-deploy

YAHOO!

Thank you

YAHOO!