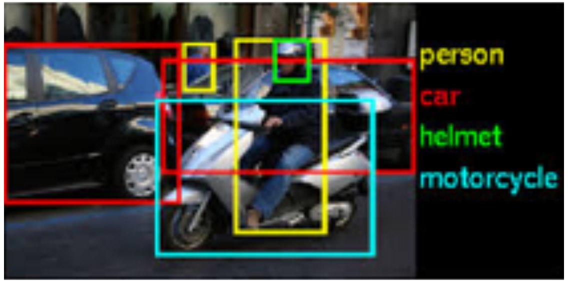
CASSANDRA SUMMIT 2016

C* for Deep Learning

Tractable

- Artificial Intelligence products for businesses
- Combining Deep Learning / Neural Networks and traditional Machine Learning
- AI using Deep Learning has surpassed human intelligence:
  - Go
  - Image Recognition



Deep Learning 101

Deep Learning

A branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations.

1.

Requires large amounts of data to train networks

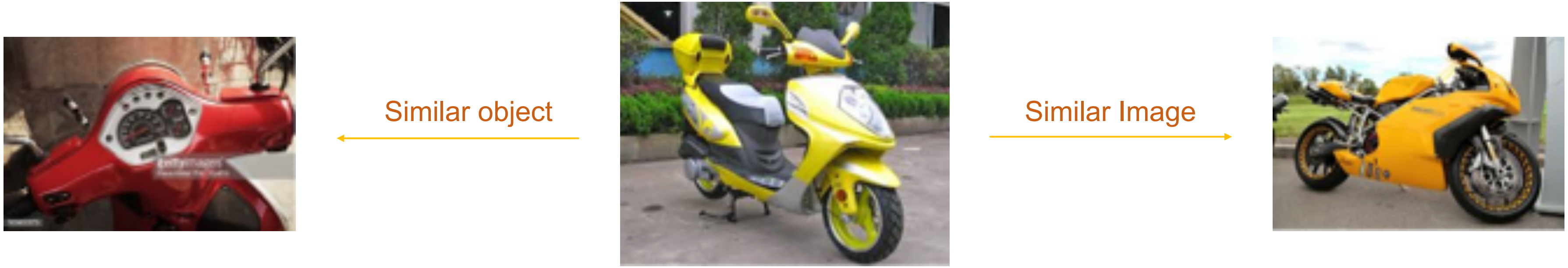
2.

Computations made feasible by use of GPUs for dramatic speedup

Semantic Image Search

Search in the meaning of the image

NOT search on the image itself



Similar object

Similar Image

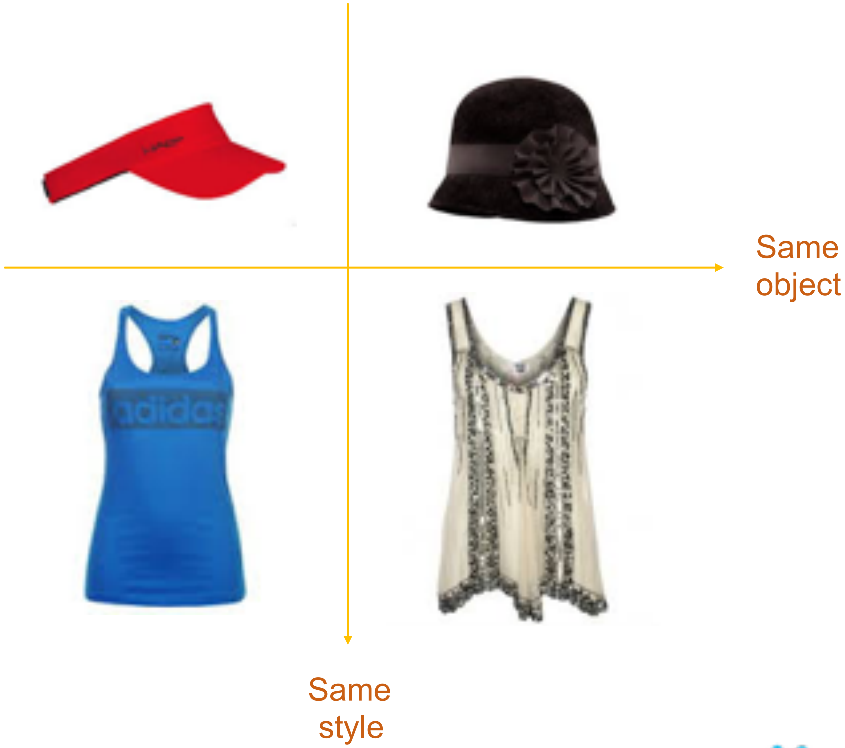
Semantic Image Search

Similar object

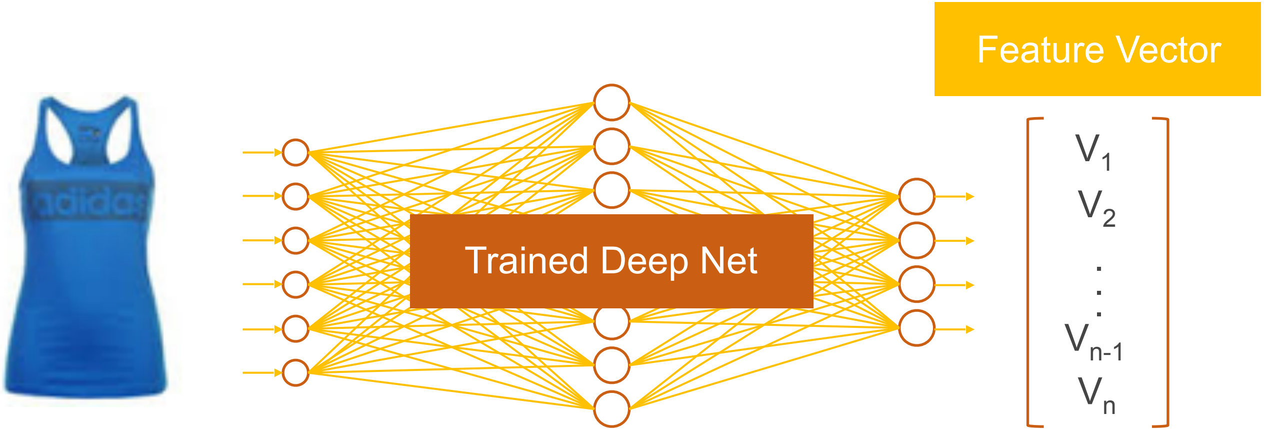
Similar Image

Semantic Search 2

- Intent can vary!
- With training AI can do both of these tasks
- Semantic search is more than just classification:
  - Ranking within a classification
  - Search for things that are not classification categories



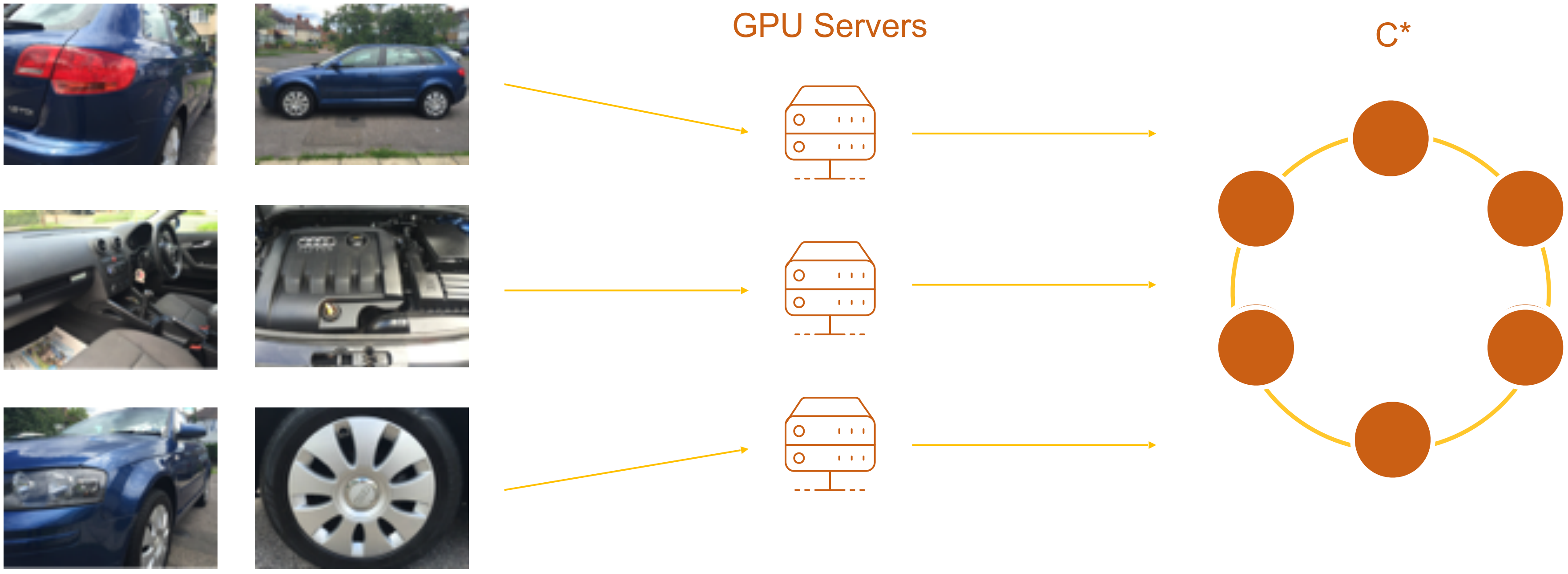
Feature Extraction



1 Image → 4000 dimension vector

1 TB Images → 300 GB of features

Feature Extraction



GPU Servers

C*

Feature Extraction

- Processing images at 5 GigaBytes per second
- Features generated at 1.6 GigaBytes per second

```
CREATE TABLE features
   listing_id uuid,
   image_id uuid,
   feature_vector blob,
PRIMARY KEY ((listing_id), image_id)
```

Search

- 300 GB of features
- Millions of rows in C*

"I want a truck like this"

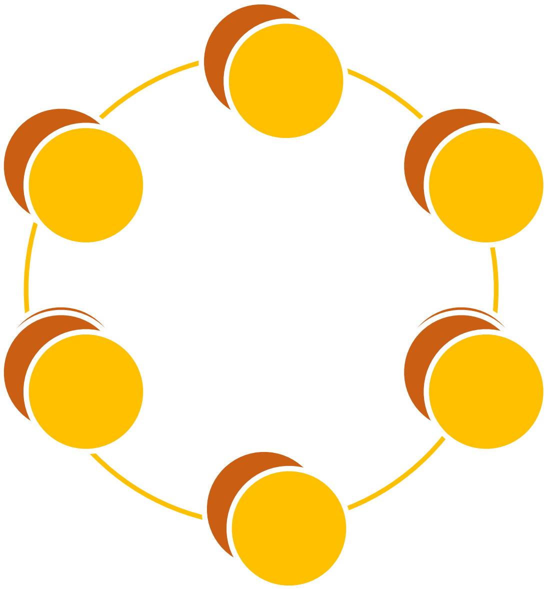
Spark

Distributed, in-memory computation

- Map-reduce
- Graph analysis
- SQL abstraction

DataStax Spark Connector

- Understands Cassandra partitioning
- Push-down queries to C* keys where possible
- Join on C* tables



Spark Similarity Search

1. Cache our 300GB features in Spark
2. Score every feature vector with our search image
3. Aggregate score for images in each listing
4. (optional) Join on ‘live’ C* data
5. Write result back to C*

```
CREATE TABLE features
  listing_id uuid,
  image_id uuid,
  feature_vector blob,
PRIMARY KEY ((listing_id), image_id)

CREATE TABLE stock_level
  listing_id uuid,
  inventory int,
  etag uuid,
PRIMARY KEY (listing_id)

CREATE TABLE query_results
  query_id uuid,
  listing_id uuid,
  score float,
PRIMARY KEY ((query_id), score)
```

Search

Search returns results within 5 seconds

100x Speedup from holding data in-memory in Spark

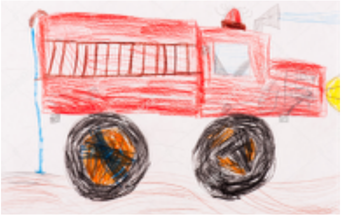
10x Speedup from co-locating Spark and C*

20x Speedup from partitioning on the group key (listing id)

5x Speedup from “Cassandra sort”

Search Results

Search Results



Sophisticated Analytics

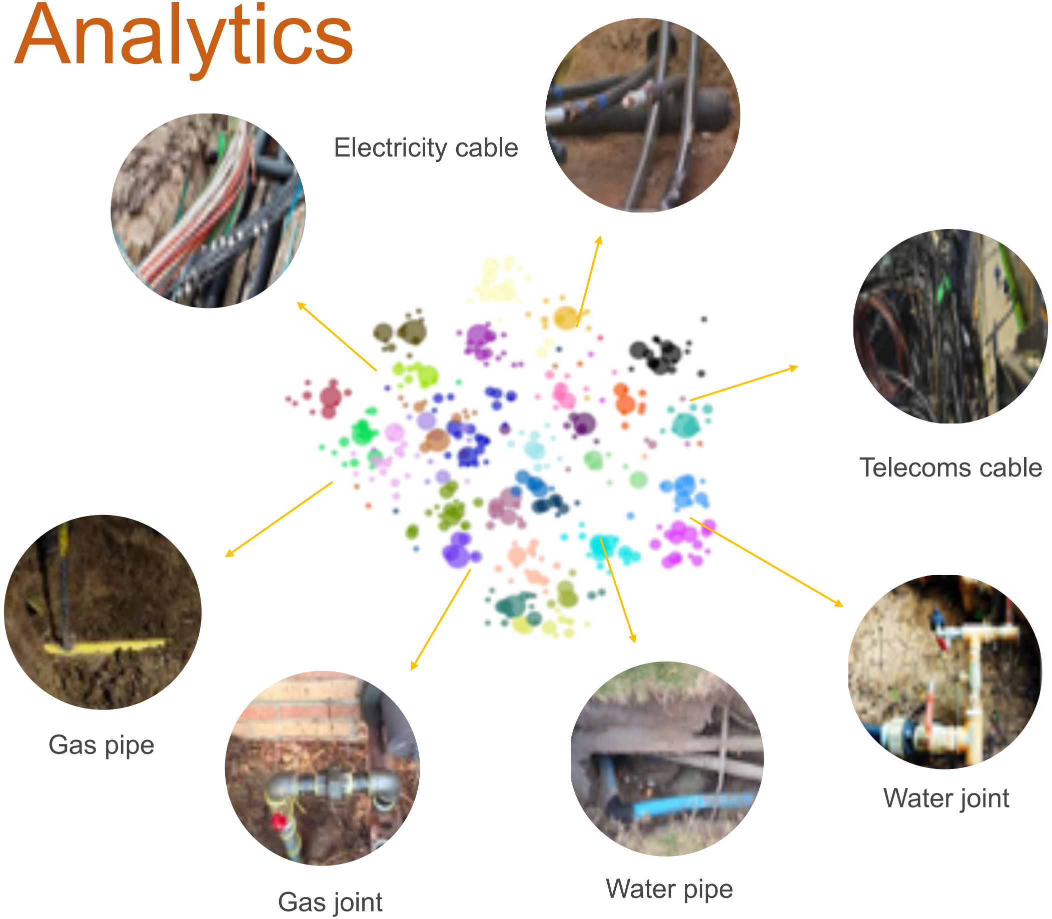
- Visualisation
- Clustering
- Outlier Detection

Real time, interactive exploration of large visual data sets

Sophisticated Analytics

- Visualisation
- Clustering
- Outlier Detection

Real time, interactive exploration of large visual data sets



Electricity cable

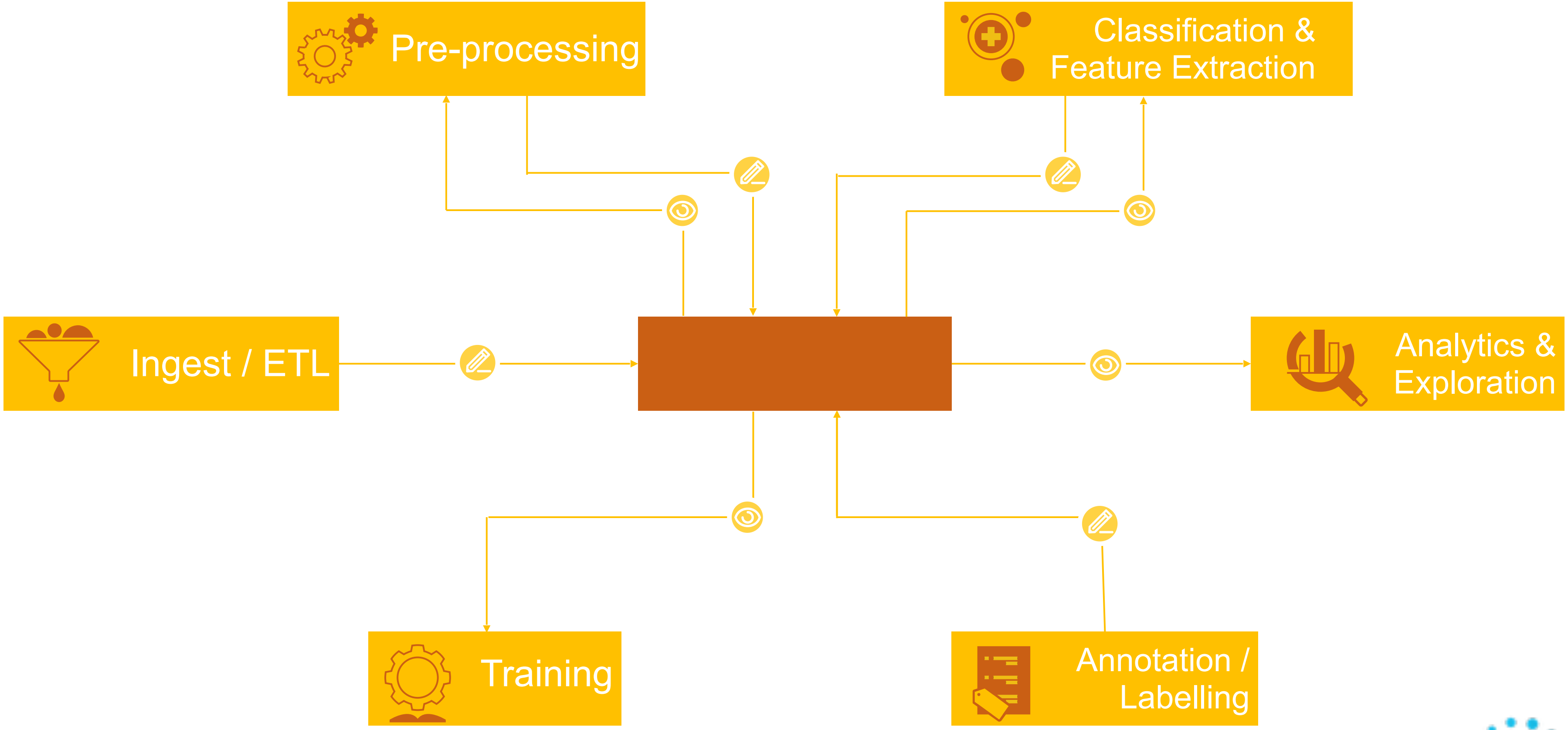
Telecoms cable

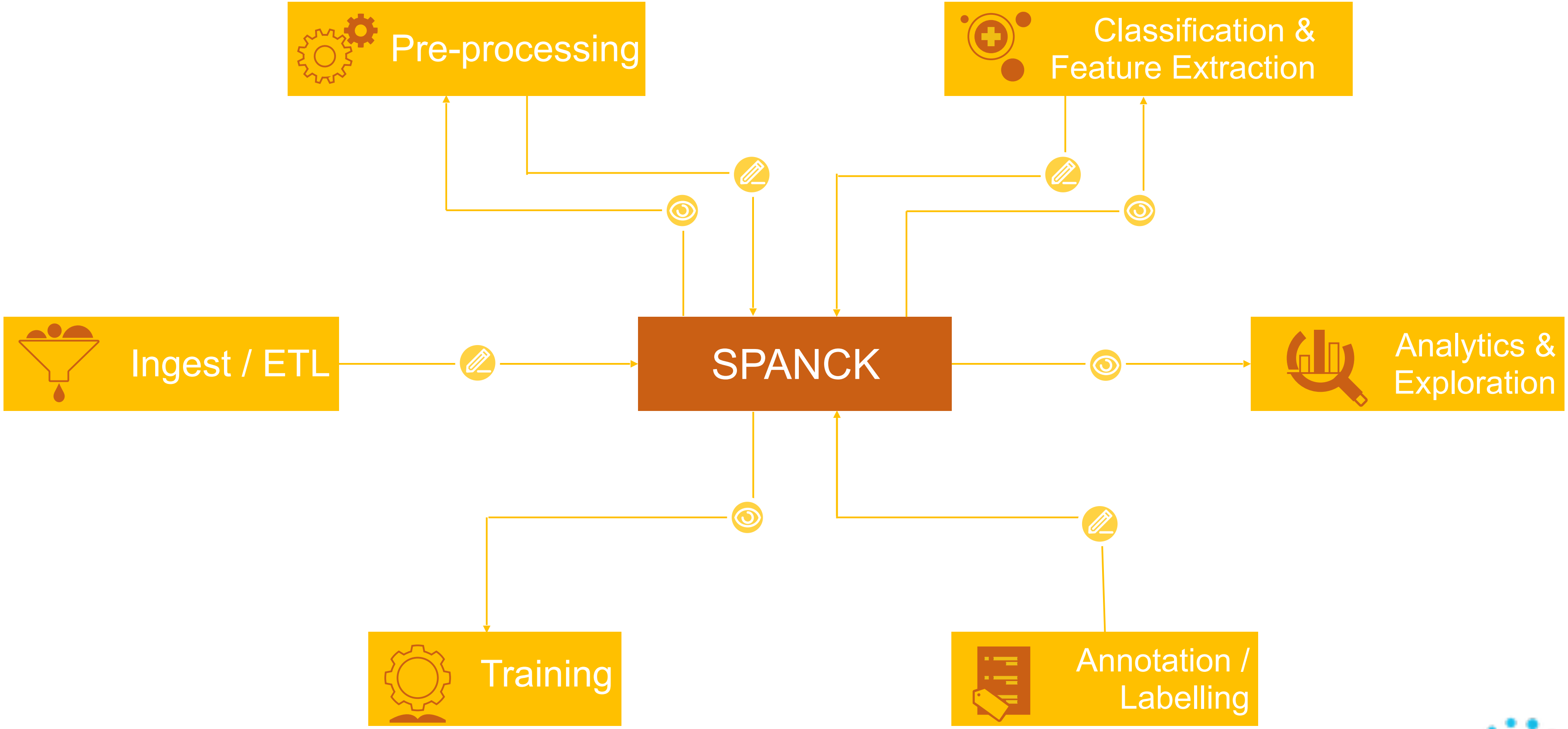
Water joint

Water pipe

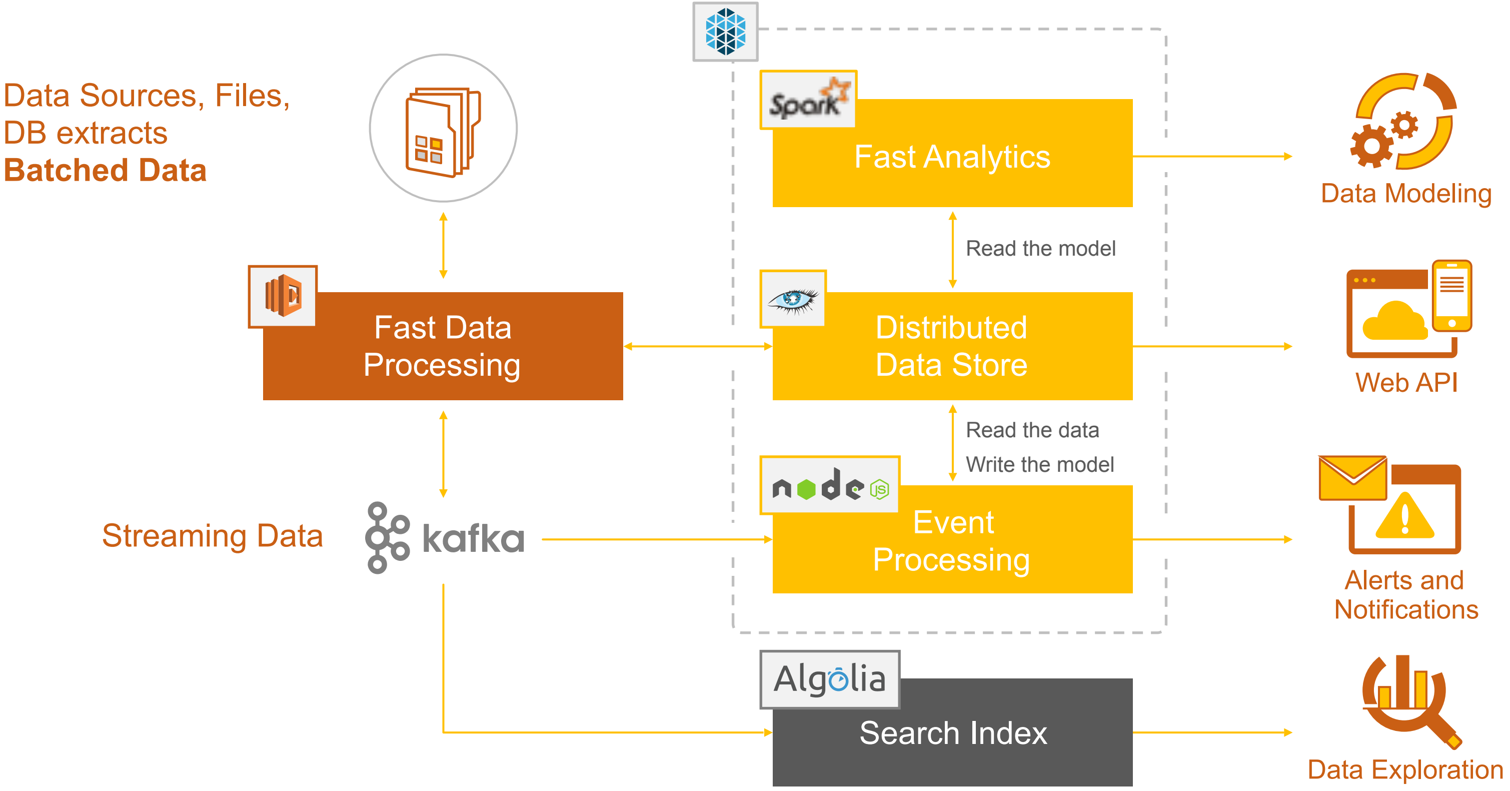
Gas joint

Gas pipe


Pre-processing
Classification & Feature Extraction
Ingest / ETL
Analytics & Exploration
Training
Annotation / Labelling


Pre-processing
Classification & Feature Extraction
Ingest / ETL
SPANCK
Analytics & Exploration
Training
Annotation / Labelling

SPANCK: Spark-Python Algolia Node Cassandra Kafka


Data Sources, Files, DB extracts
Batched Data

Fast Data Processing

Streaming Data

Fast Analytics

Read the model

Distributed Data Store

Read the data

Write the model

Event Processing

Search Index

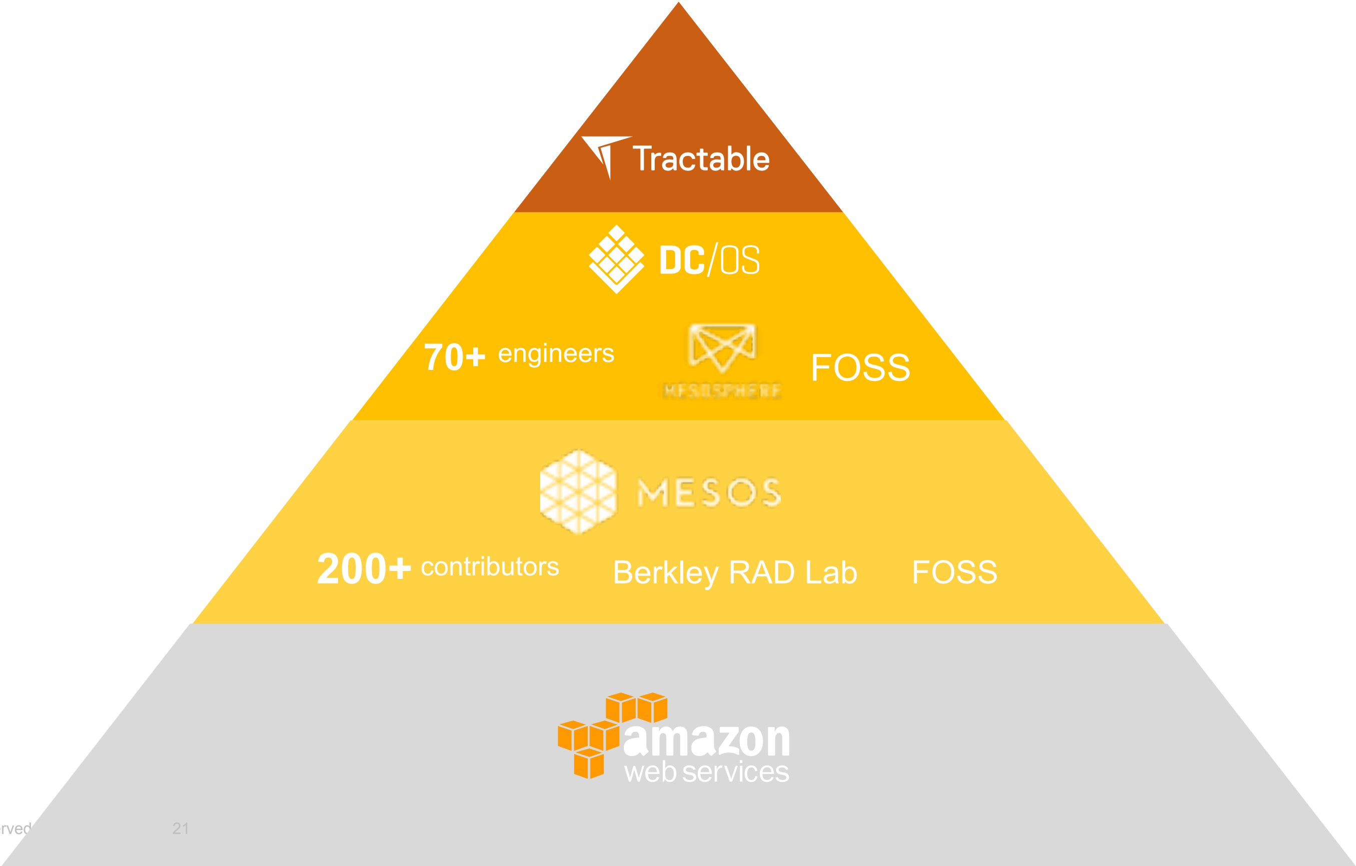
Data Modeling

Web API

Alerts and Notifications

Data Exploration

Many Services, 1 Engineer



Tractable

DC/OS

70+ engineers MESOSPHERE FOSS

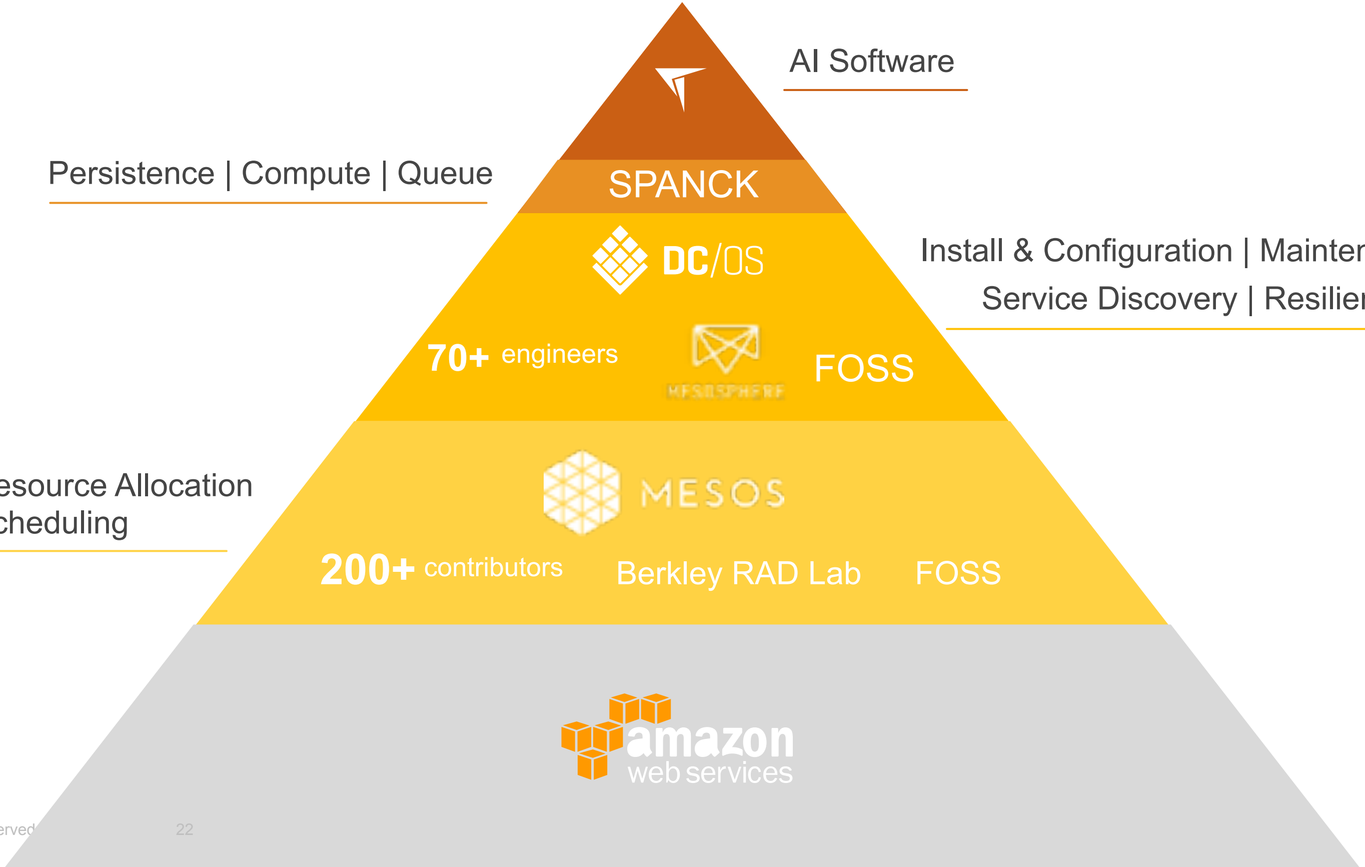
MESOS

200+ contributors Berkley RAD Lab FOSS

amazon web services

CASSANDRA SUMMIT 2016

Many Services, 1 Engineer



AI Software

SPANCK

Persistence | Compute | Queue

DC/OS

70+ engineers

MESOSPHERE

FOSS

Install & Configuration | Maintenance
Service Discovery | Resilience

MESOS

Resource Allocation
Scheduling

200+ contributors

Berkley RAD Lab

FOSS

amazon
web services

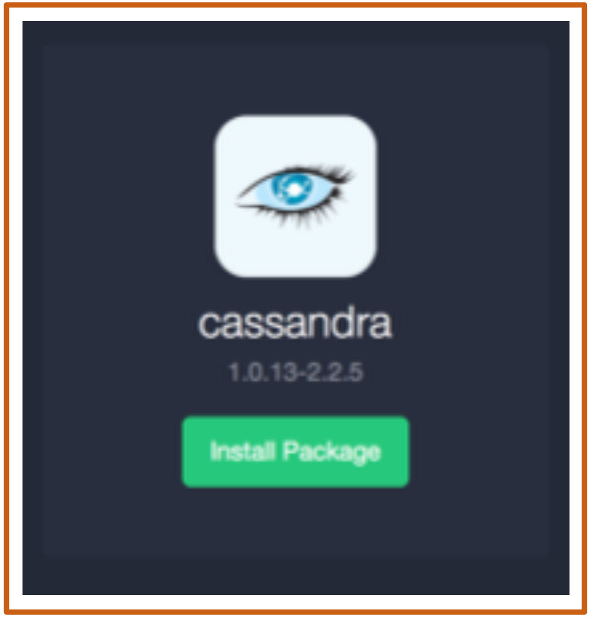
DCOS Frameworks

Easy Install

$ dcos package install kafka --options kafka-options.json

$ dcos package install cassandra --options cassandra-options.json

```
{
  "nodes": {
    "count": 12,
    "cpus": 9,
    "mem": 30000,
    "disk": 400000,
    "heap": {
      "size": 8000,
      "new": 1500
      }
  },
  "cassandra": {
    "row_cache_size_in_mb": 12000,
    "row_cache_save_period": 600,
    "commitlog_segment_size_in_mb": 64,
    "concurrent_reads": 24,
    "concurrent_writes": 64,
    "memtable_allocation_type": "offheap_objects",
    "compaction_throughput_mb_per_sec": 128
  }
}
```



DCOS Frameworks

Service Discovery

- 0 configuration files
- 0 configuration / orchestration systems
- Works ever time, in every environment

```
function fetchMarathonServiceBaseUrl(serviceName){
   var dns_url = ['_' + serviceName, "_tcp.marathon.mesos"].join('.');
   return resolveSrv(dns_url).then(function(addresses){
      return "http://" + addresses[0].name + ":" + addresses[0].port;
   })
}


fetchCassandraNodes = (mesosHost, authToken) ->
  mesosDns.fetchMarathonServiceBaseUrl('cassandra').then (baseUrl) ->
   request(
     url: baseUrl + "/v1/nodes/connect",
     headers: headers
   ).then (result) ->
     result = JSON.parse(result)
     return _.pluck(result.nodes, “ip")
```

DCOS Frameworks

Maintenance and Admin DCOS Command Line Tools

- Replace a C* node
- Backup / Restore C* to AWS S3
- Run C* repair / cleanup
- Restart C* nodes
- Replace a Kafka broker
- Rebalance Kafka brokers
- Restart Kafka brokers

```
dcos cassandra —name=cassandra \
     cleanup —key_spaces=dev,test
```

```
dcos cassandra —name=cassandra \
     replace node-4
```

```
dcos kafka broker replace 3
```

DCOS Frameworks

Shipping Docker Apps

1 JSON file

1 CLI command

Apps include

- Spark Drivers
- Web Servers
- Kafka Producers
- Kafka Consumers
- APIs

```
{
  "id": "/my-app",
  "cpus": 1,
  "mem": 2048,
  "instances": 1,
  "env": {
    "NODE_ENV": "dev"
  },
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "tractableio/my-app:0.1",
      "network": "BRIDGE"
    }
  },
  "healthChecks": [
    {
      "path": "/healthy",
      "maxConsecutiveFailures": 2
    }
  ]
}
```

*JSON Slightly simplified

DCOS Cluster in 15 minutes

1.

Deploy DCOS cluster into AWS using CloudFormation template

2.

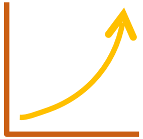
Install Cassandra & Kafka

3.

Deploy app docker containers into DCOS

4.

Profit!



Conclusion

- Cassandra’s high write speeds allow it to ingest features from Deep Networks
- Cassandra and Spark provides a powerful compute+storage system
- Spark, Cassandra and Kafka can provide a versatile data backbone that supports a range of use cases
- Mesosphere DCOS is a low-effort, high-reward way of running distributed systems