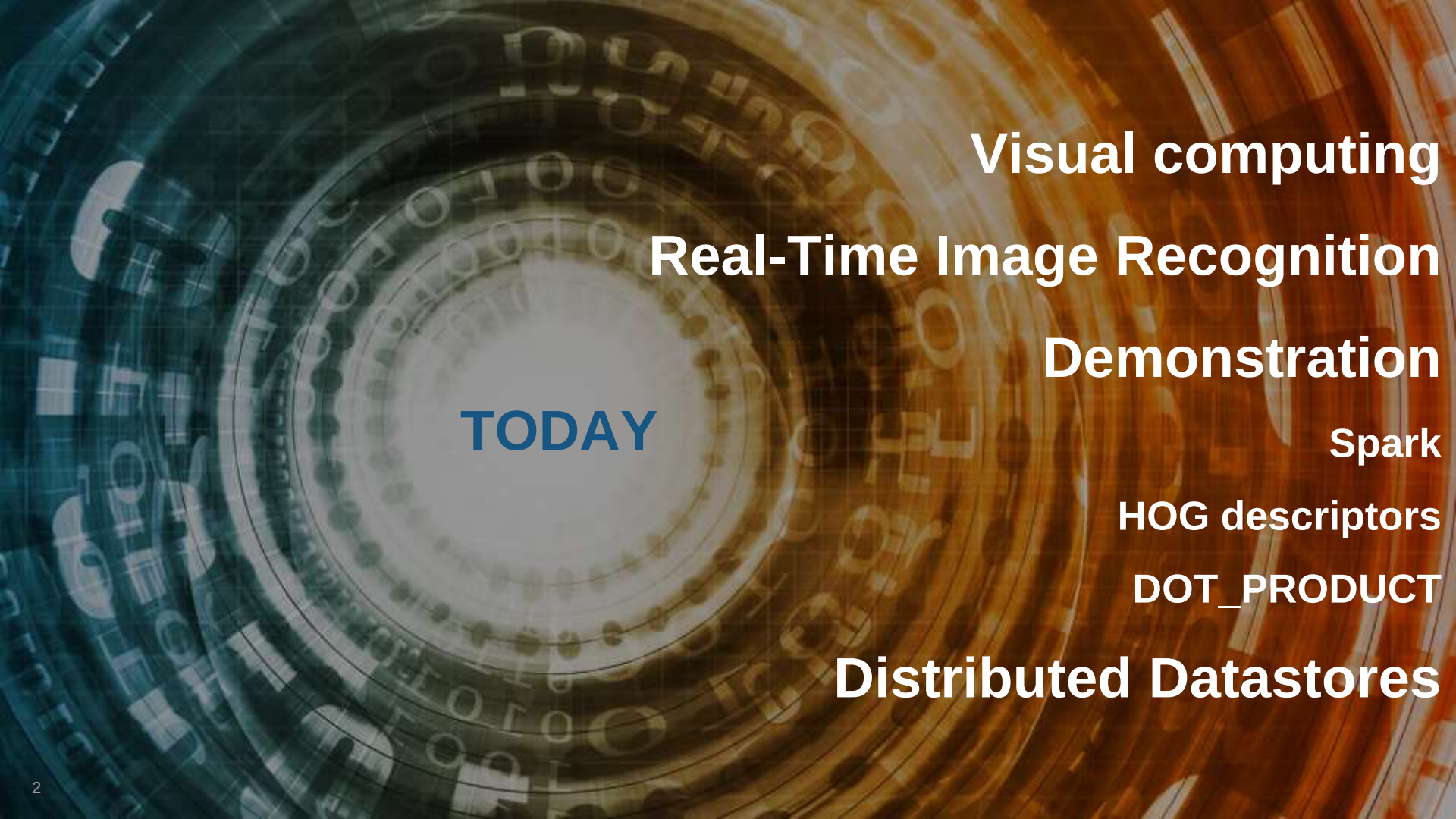
SPARK SUMMIT

Real-Time Image Recognition

with Apache Spark

@garyorenstein

Session hashtag #EUai0


TODAY
Visual computing
Real-Time Image Recognition
Demonstration
Spark
HOG descriptors
DOT_PRODUCT
Distributed Datastores

Visual Computing

The future of computing is visual

and also
numerical :)

The World’s Four Most Valuable Companies

	Market Cap $B
AAPL	807
GOOGL	680
MSFT	606
FB	495

ARFaceAnchor


Google

Microsoft®


box
TED
S


facebook


Oculus Connect 4 | Day 1 Keynote
Press esc to exit full screen
6:36 / 2:05:16

Image Recognition Today


thorn
DIGITAL DEFENDERS
OF CHILDREN

949:0.026740,961:0.011758,962:0.01 ...

12:0.005868,16:0.004575,49:0.002193,52:0.009880,67:0.034832,72:0.030992,77:0.012170,108:0.012382,120:0.012916,125:0.005741,137:0.015322,143:0.020548,157:0.030407,220:0.061202,228:0.026140,232:0.040047,236:0.023434,242:0.026605,252:0.007459,264:0.022012,269:0.016690,270:0.057932,282:0.011975,292:0.028855,298:0.006937,317:0.005120,333:0.028555,338:0.039100,348:0.017727,358:0.055682,376:0.006209,386:0.028764,413:0.017220,417:0.018298,422:0.004943,433:0.031690,443:0.011401,451:0.016825,452:0.000745,458:0.010769,460:0.044923,471:0.039836,479:0.008343,482:0.009446,484:0.019443,497:0.061289,502:0.015072,508:0.029485,530:0.013753,532:0.007153,543:0.044873,551:0.010136,555:0.012994,560:0.008001,563:0.038678,579:0.015128,610:0.007795,627:0.019286,634:0.021111,641:0.007065,642:0.007089,659:0.058285,672:0.018122,674:0.024745,703:0.012181,704:0.010520,705:0.019805,726:0.004800,734:0.020477,751:0.005154,753:0.023470,763:0.002651,783:0.033653,786:0.010800,824:0.017787,846:0.017696,850:0.040618,853:0.006627,880:0.020177,887:0.040712,901:0.004130,902:0.012970,926:0.011321,949:0.026740,961:0.

033653,786:0.010800,824:0.017787,846:0.017696,850:0.040618,853:0.006627,880:0.020177,887:0.040712,901:0.004130,902:0.012970,926:0.011321,949:0.026740,961:0.011758,962:0.01,.003080,966:0.025391,969:0.008317,980:0.024180,999:0.025001,1003:0.009995,1018:0.026575,1024:0.014152,1030:0.014807,1032:0.001685,1037:0.059401,1041:0.008451,1083:0.004498,1086:0.042539,1100:0.019762,1107:0.003233,1111:0.010055,1118:0.004970,1120:0.013391,1137:0.033611,1143:0.004184,1151:0.011988,1156:0.018991,1164:0.005059,1165:0.009926,1171:0.041736,1181:0.009872,1187:0.001813,1188:0.010391,1193:0.020764,1194:0.002471,1222:0.006705,1238:0.009757,1246:0.067453,1259:0.042624,1264:0.017558,1265:0.019401,1269:0.015384,1299:0.013593,1310:0.002139,1359:0.006642,1371:0.034178,1374:0.016396,1384:0.022928,1404:0.017169,1408:0.009406,1418:0.073914,1420:0.011940,1421:0.005672,1430:0.003974,1433:0.002776,1463:0.031537,1481:0.000885,1485:0.039955,1492:0.023929,1494:0.048229,1497:0.053608,1508:0.003894,1518:0.011840,1524:0.011318,1528:0.

067235,1551:0.002643,1569:0.030303,1592:0.000982,1595:0.021256,1606:0.029090,1619:0.030494,1628:0.007809,1630:0.012805,1632:0.074610,1658:0.046989,1663:0.011392,1683:0.025755,1689:0.000551,1690:0.019549,1707:0.002039,1718:0.000027,1753:0.003988,1761:0.016639,1787:0.004682,1788:0.036989,1793:0.010178,1799:0.032016,1820:0.001699,1862:0.026061,1865:0.033358,1888:0.015540,1893:0.015230,1913:0.029057,1917:0.017459,1930:0.012725,1932:0.020591,1939:0.036401,1940:0.001455,1941:0.029777,1948:0.028731,1950:0.015147,1966:0.008172,1976:0.004087,2009:0.005937,2011:0.026532,2016:0.018998,2023:0.003567,2024:0.033425,2043:0.024501,2060:0.035672,2077:0.026460,2092:0.006496,2099:0.042786,2110:0.031982,2117:0.026819,2118:0.002956,2127:0.002132,2171:0.006693,2174:0.006085,2193:0.038693,2207:0.080437,2210:0.036449,2215:0.027432,2216:0.000524,2228:0.022542,2232:0.023016,2245:0.035095,2258:0.008138,2291:0.014170,2297:0.024569,2301:0.019651,2310:0.037032,2333:0.010741,2337:0.010183,2353:0.056520,2382:0.005700,2406:0.012346,2409:0.045950,2411:0.005816,2415:0.001264,2424:0.046932,2439:0.010018,.

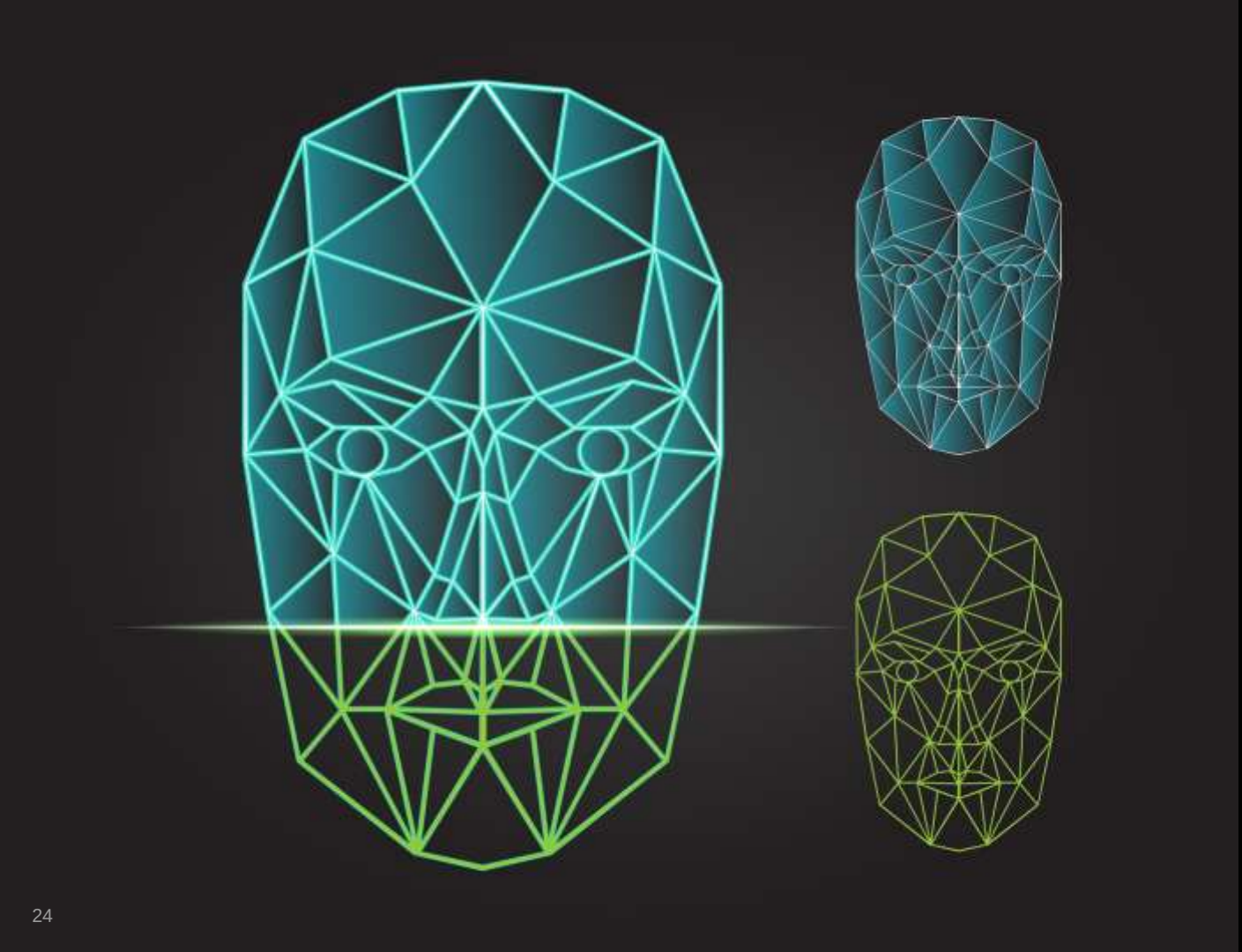
100s of millions of images to match

1,000x match improvement time

MEMSQL


26% 10:11
Capture your product

Real-Time Image Recognition



How It Works

Real-Time Image Recognition

Build and Train Models	Extract Feature Vectors	Build Real-Time Applications
Spark TensorFlow Gluon Caffe	Use the model to extract feature vectors and “classify”	Store every vector in a MemSQL table
OpenCV HOG Descriptor	Model+ Image => FV	DOT_PRODUCT for feature comparison

```
CREATE TABLE features (
    id int,
    a binary(4096)
KEY(id) USING CLUSTERED COLUMNSTORE
);
```

Working with Feature Vectors

For every image we store an `ID` and a normalized feature vector in a MemSQL table called `features`.

```
ID      |       Feature Vector
x       |       4KB
```

To find similar images we use this SQL query

```
SELECT
        id
FROM
        features
WHERE
        DOT_PRODUCT(feature * <input>) > 0.9
```

Understanding DOT_PRODUCT

DOT_PRODUCT is an algebraic operation

X = (x1, ... , XN), Y = (y1, …, yN)

(X*Y) = SUM(Xi*Yi)

With the specific model and normalized feature vectors DOT_PRODUCT results in a similarity score.

Scores closer to 1 indicate most similarity

Performance Enhancing Techniques

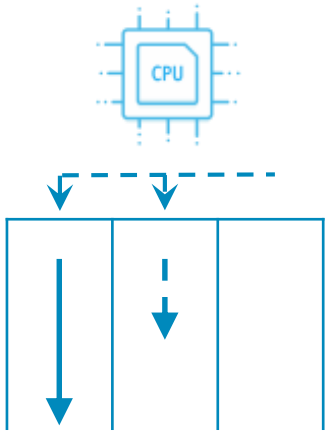
Achieving best-in-class DOT_PRODUCT implementation

- SIMD-powered
- Data compression
- Query parallelism
- Scale out

Result: Processing at **Memory Bandwidth Speed**

Query Vectorization Overview

Not Vectorized

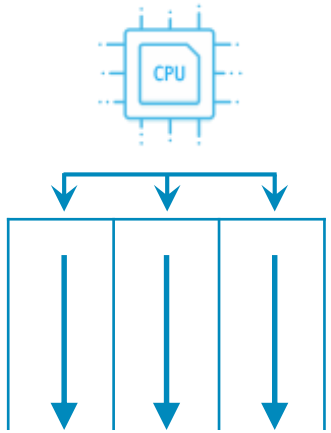


Single row, Single instruction

CPU constrained

10,000 rows / sec / core

Vectorized



Multiple rows, Single instruction

CPU optimized

1,000,000,000 rows / sec / core

Performance Numbers

Memory Speed ~**50** GB/sec

Each vector 4KB

Theoretical max

50 GB / 4KB = **12,500,000** images / second / node

1 Billion images a second on 100 node cluster

Pinned Memory Bandwidth (in MB/sec) – Anandtech.com

Mem Hierarchy	AMD "Naples" EPYC 7601 DDR4-2400	Intel "Skylake-SP" Xeon 8176 DDR4-2666	Intel "**Broadwell**-EP" Xeon E5-2699**v4** DDR4-2400
1 Thread	27490	12224	18555
2 Threads, same core same socket	27663	14313	19043
2 Threads, different cores same socket	29836	24462	37279
2 Threads, different socket	54997	24387	37333
4 threads on the first 4 cores same socket	29201	47986	53983
8 threads on the first 8 cores same socket	32703	77884	61450
8 threads on different dies (core 0,4,8,12...) same socket	98747	77880	61504

37-61 GB/s

One Machine – m4.xlarge 4 CPU, 16GB RAM

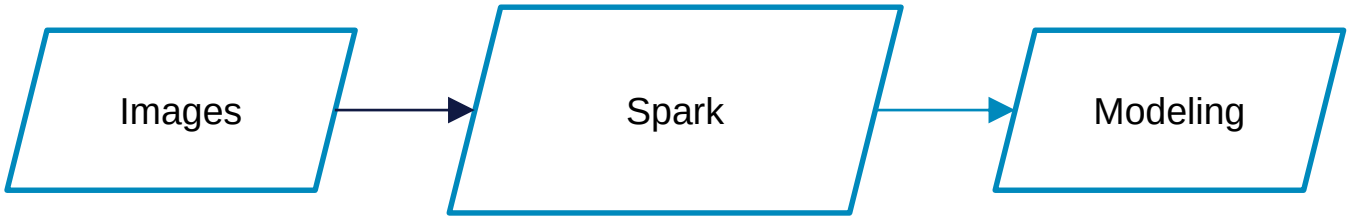
Latest generation of General Purpose Instances

Features:

2.3 GHz Intel Xeon® E5-2686 **v4** (**Broadwell**) processors or 2.4 GHz Intel Xeon® E5-2676 v3 (Haswell) processors

Demo

Demo Architecture – Part 1



Open Source Computer Vision Library

BSD license

C++, C, Python and Java interfaces

Windows, Linux, Mac OS, iOS and Android

Designed for computational efficiency

Strong focus on real-time applications

Written in optimized C/C++

Multi-core processing

Histogram of **O**riented **G**radients

HOG

feature descriptor

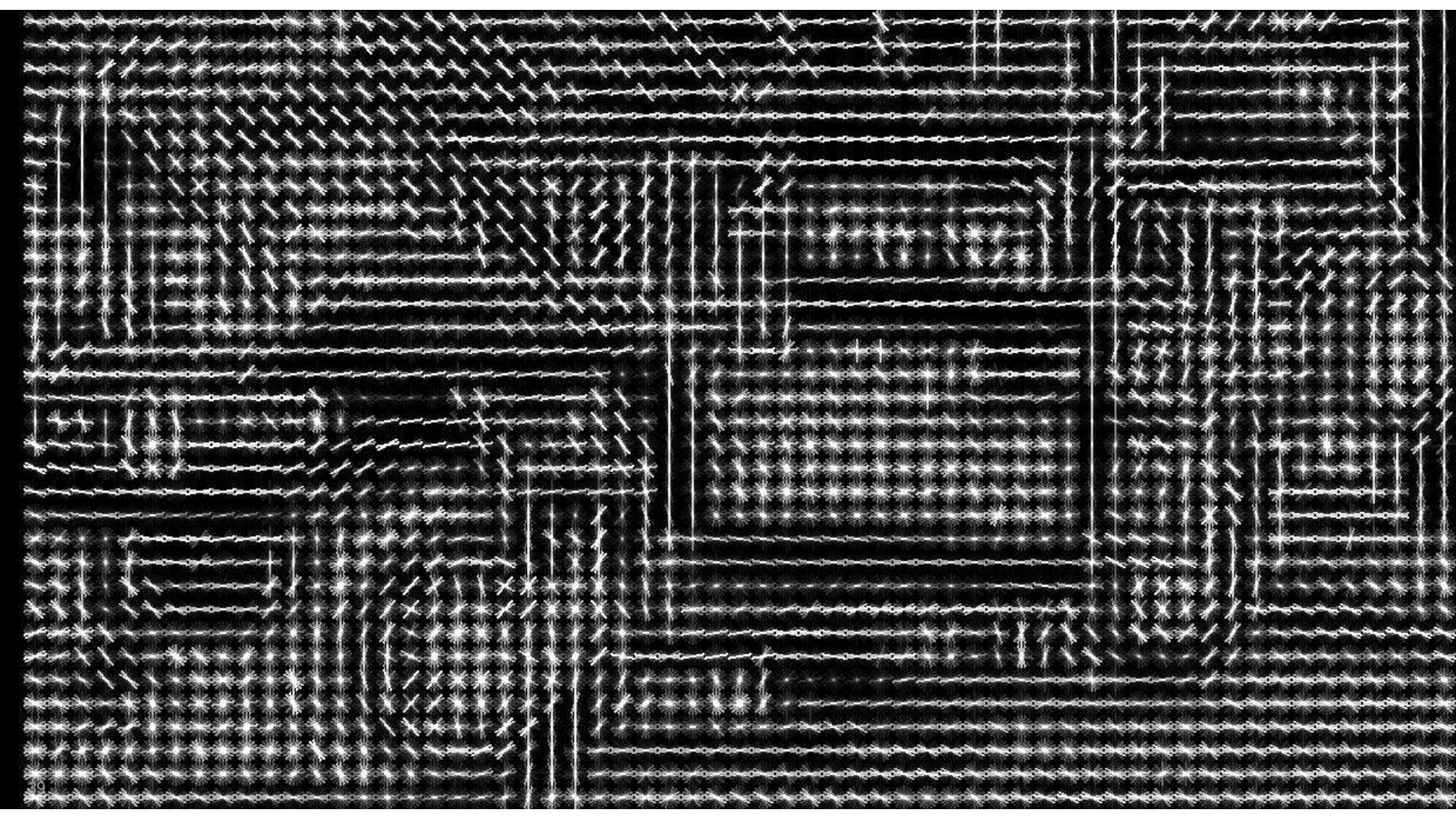
HOGgles: Visualizing Object Detection Features

Carl Vondrick Aditya Khosla Hamed Pirsiavash Tomasz Malisiewicz Antonio Torralba

Massachusetts Institute of Technology

Oral presentation at ICCV 2013

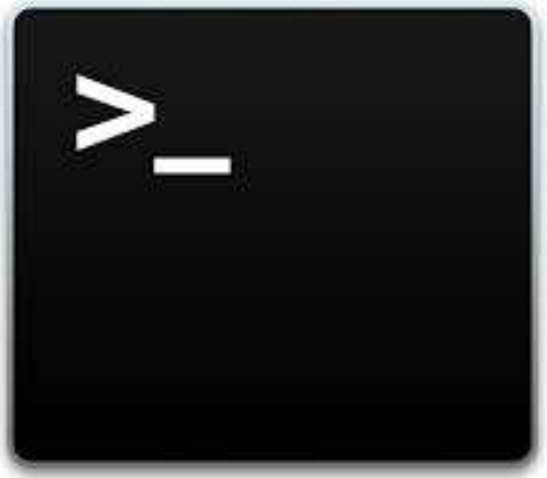
carlvondrick.com/ihog



openCV.py – Generate HOG Descriptor

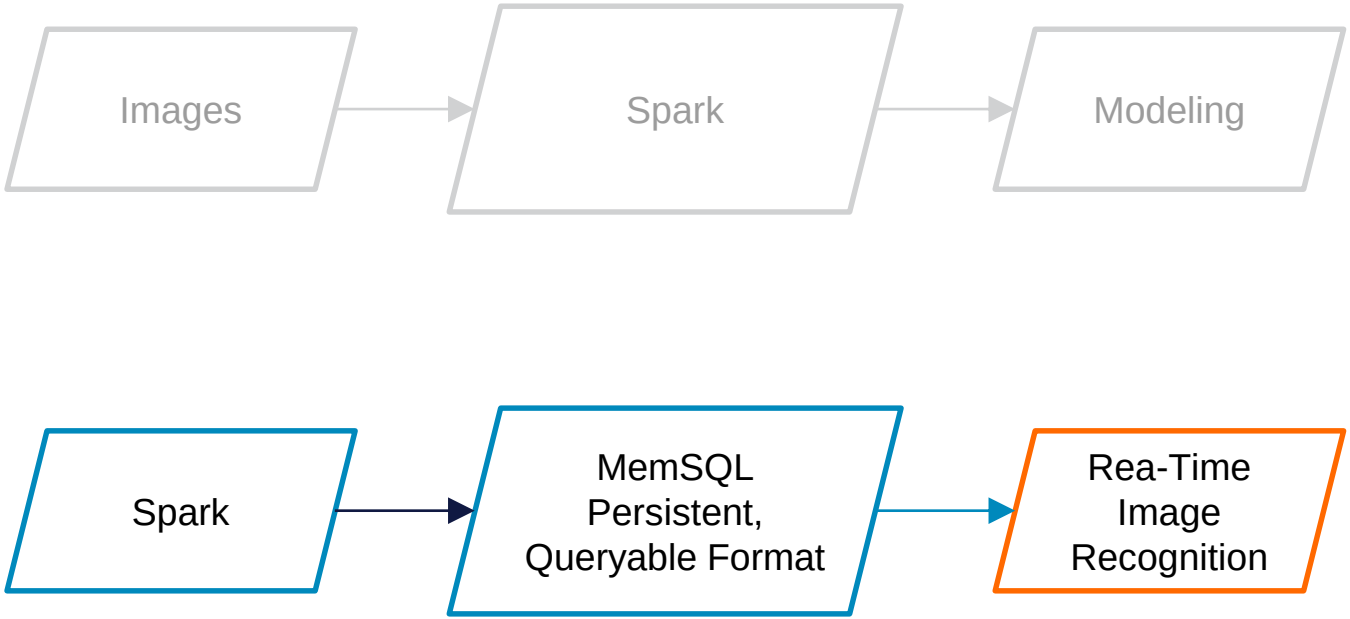
```
import cv2
#Calculate Hog Descriptor
hog = cv2.HOGDescriptor()
im = cv2.imread('gary.jpg')
h = hog.compute(im)

#turn list of lists into a single vector
finalList = list()
for i in h:
  finalList.append(i[0])
print finalList
```

gary.jpg



[0.016956484, 0.085721202, 0.26830149, 0.059500914, 0.3097299, 0.011304559, 0.0292686, 0.007835675, 0.01574548, 0.063435704, 0.16189502, 0.2643134, 0.048782568, 0.327362, 0.014534432, 0.045295727, 0.01273266, 0.024890875, 0.020722726, 0.10444726, 0.31398779, 0.050675713, 0.39446563, 0.0024092258, 0.023932906, 0.015324682, 0.020758351, 0.062628902, 0.15484644, 0.24577968, 0.067200541, 0.48367494, 0.0046112579, 0.041370217, 0.017862586, 0.02563199, 0.068649702, 0.11981961, 0.15649728, 0.062572055, 0.22964591, 0.0015388859, 0.0060626287, 0.0017874539, 0.025700238, 0.025409972, 0.064399928, 0.17423819, 0.27280667, 0.46269512, 0.0, 0.0, 0.0, 0.010291508, 0.071518339, 0.11347346, 0.11948239, 0.024386467, 0.29117766, 0.00073840714, 0.0061320281, 0.0043423013, 0.025102539, 0.02979758, 0.066815235, 0.13293755, 0.13425487, 0.63947016, 0.0, 0.0, 0.0039961291, 0.014642118, 0.00020346629, 0.009798184, 0.05444653, 0.20338155, 0.35373437, 0.0, 0.0, 0.0, 0.00020346629, 0.0, 0.0024293216, 0.020908076, 0.25147814, 0.48959622, 0.0020539484, 0.0010105423, 0.0, 0.0, 0.0012496823, 0.014430397, 0.044209853, 0.11989171, 0.48959622, 0.0, 0.0, 0.00095677841, 0.0032640444, 0.0010254302, 0.0089193229, 0.028186308, 0.15913466, 0.48959622, 0.0020619868, 0.0010144971, 0.00017375893, 0.00068302121, 0.0, 0.0010355262, 0.025299216, 0.23020783, 0.62172222, 0.01451685, 0.0071422886, 0.0, 0.0, 0.0, 0.012573234, 0.054655403, 0.059691429, 0.20888245, 0.011290883, 0.0055551128, 0.0, 0.0, 0.0012445236, 0.0074641695, 0.029826088, 0.15533075, 0.62172222, 0.014573662, 0.0071702395, 0.0, 0.00036294275, 0.0028592346, 0.02620801, 0.073913597, 0.039936692, 0.28820238, 0.011335071, 0.0058611422, 9.680009e-05, 0.0018202713, 0.0091516012, 0.060819447, 0.1838807, 0.0038819138, 0.20110132, 0.0, 0.0, 0.0, 0.0091516012, 0.016823623, 0.071278378, 0.21329209, 0.0, 0.56339824, 0.00042593898, 0.00020428553, 0.0, 0.016823623, 0.01358252, 0.092970245, 0.27070105, 0.0, 0.28949663, 0.0, 0.0034472728, 0.0016952065, 0.013177372, 0.02054593, 0.13707943, 0.39466769, 0.0060899607, 0.44661811, 0.0063890843, 0.0057454957, 0.012835419, 0.021771187, 0.026936026, 0.039015748, 0.12060358, 0.011743868, 0.5222494, 0.0028561817, 0.008516633, 0.009253067, 0.02824329, 0.14792503, 0.042693377, 0.047852691, 0.015099258, 0.347835, 0.006142302, 0.025793232, 0.041446675, 0.14895588, 0.047773097, 0.086114474, 0.25319242, 0.037829686, 0.42512667, 0.058012363, 0.033889204, 0.016778972, 0.049306624, 0.27457103, 0.14510864, 0.1595912, 0.037092552, 0.25107884, 0.066688649, 0.052581675, 0.043319989, 0.26188877, 0.19920787, 0.066857897, 0.034212913, 0.0026680217, 0.095135577, 0.019466352, 0.087019734, 0.089628078, 0.20096146, 0.21869336, 0.07828752, 0.10058635, 0.0, 0.032161083, 0.022195378, 0.17397901, 0.1314576, 0.22527759, 0.34009805, 0.19812617, 0.10773266, 0.0034303134, 0.083222859, 0.032420114, 0.081477858, 0.17678982, 0.33856934, 0.33169824, 0.16326106, 0.11812425, 0.0, 0.081663303, 0.048922073, 0.15971793, 0.27516699, 0.35285512, 0.16723804, 0.045273483, 0.083364449, 0.0, 0.15622558, 0.048965197, 0.17034502, 0.07251998, 0.16749914, 0.048772424, 0.017524442, 0.052417304, 0.0, 0.27051157, 0.045652088, 0.29263556, 0.095339336, 0.048772424, 0.28674465, 0.079618029, 0.096269205, 0.0, 0.26283014, 0.24753372, 0.15255482, 0.10626037, 0.28918052, 0.10103676, 0.020614117, 0.061667707, 0.0, 0.41343474, 0.2475501, 0.26837826, 0.086498402, 0.10103676, 0.083137847, 0.010956642, 0.032786358, 0.0, 0.28870174, 0.0046706107, 0.37027311, 0.1237675, 0.083137847, 0.15904032, 0.064760096, 0.19378686, 0.0, 0.22081088, 0.0, 0.10364717, 0.034637153, 0.15904032, 0.13996357, 0.017891405, 0.053537831, 0.0, 0.29055294, 0.033957928, 0.38227352, 0.12775132, 0.13996357, 0.27015582, 0.10796007, 0.32305765, 0.0, 0.10741439, 0.0, 0.093926013, 0.031388499, 0.27015582, 0.047117099, 0.072411381, 0.21668248, 0.0, 0.15464506, 0.0, 0.0067523676, 0.0022216472, 0.047117099, 0.010963425, 0.074762799, 0.24686243, 0.047237419, 0.48422337, 0.0, 0.10382935, 0.034410644, 0.010963425, 0.094231755, 0.070089191, 0.20978142, 0.0, 0.13647549, 0.0, 0.0047317459, 0.0015583917, 0.094231755, 0.040479153, 0.059797421, 0.1814011, 0.018099368, 0.67738122, 0.0, 0.079226345, 0.033079416, 0.035598744, 0.0, 0.012998134, 0.066920422, 0.08456859, 0.28807223, 0.0, 0.028599452, 0.009527443, 0.0, 0.0, 0.0097244699, 0.049000099, 0.16630217, 0.55629677, 0.0, 0.0083118174, 0.0027834533, 0.0, 0.014885568, 0.018756978, 0.053434443, 0.056956775, 0.39798912, 0.0, 0.02281102, 0.01010431, 0.0051969145, 0.012848671, 0.019682899, 0.052198373, 0.17997465, 0.59763575, 0.0, 0.0082910238, 0.0029412413, 0.0035903123, 0.0, 0.022289289,

Demo Architecture – Part 2



MEMSQL

Real-Time Query Across All Vectors

Assuming Memory Speed ~**50** GB/sec

Each vector 4KB

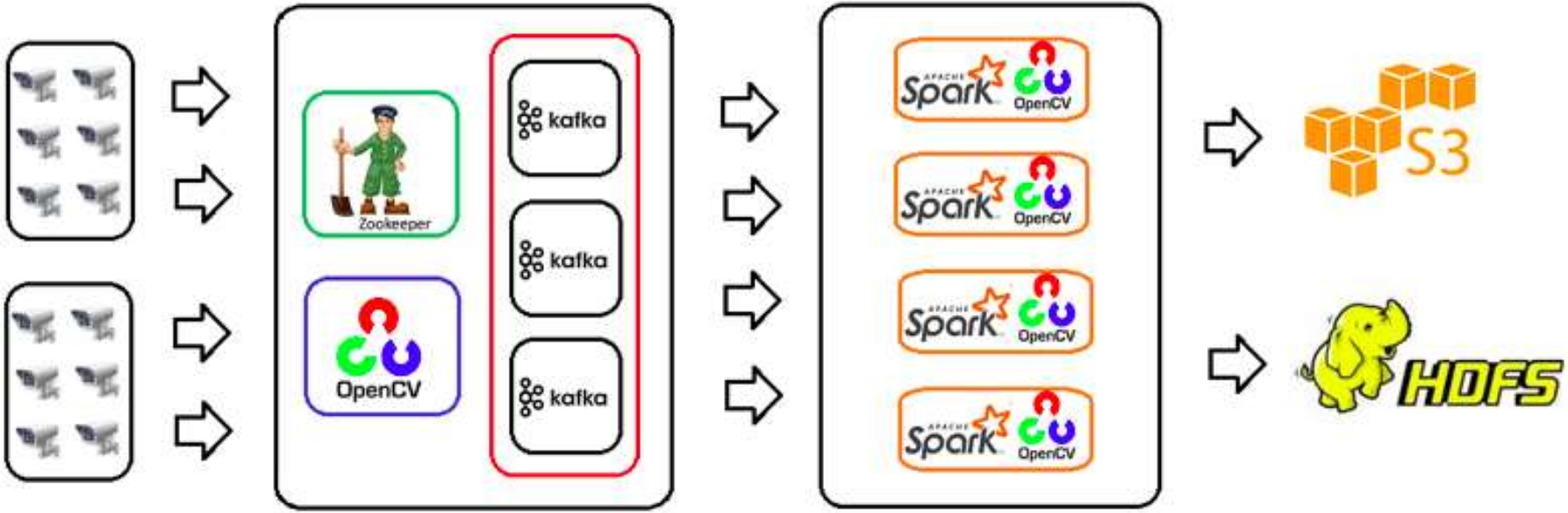
Theoretical max

50 GB / 4KB = **12,500,000** images / second / node

Full table scan should take about 1 second

```
SELECT
    count (*)
FROM
    features
WHERE
    DOT_PRODUCT(a, 0xa334efa…
) > .99;
```

Distributed Datastores


Zookeeper
kafka
kafka
kafka
OpenCV
Spark
OpenCV
Spark
OpenCV
Spark
OpenCV
Spark
OpenCV
S3
HDFS

About MemSQL

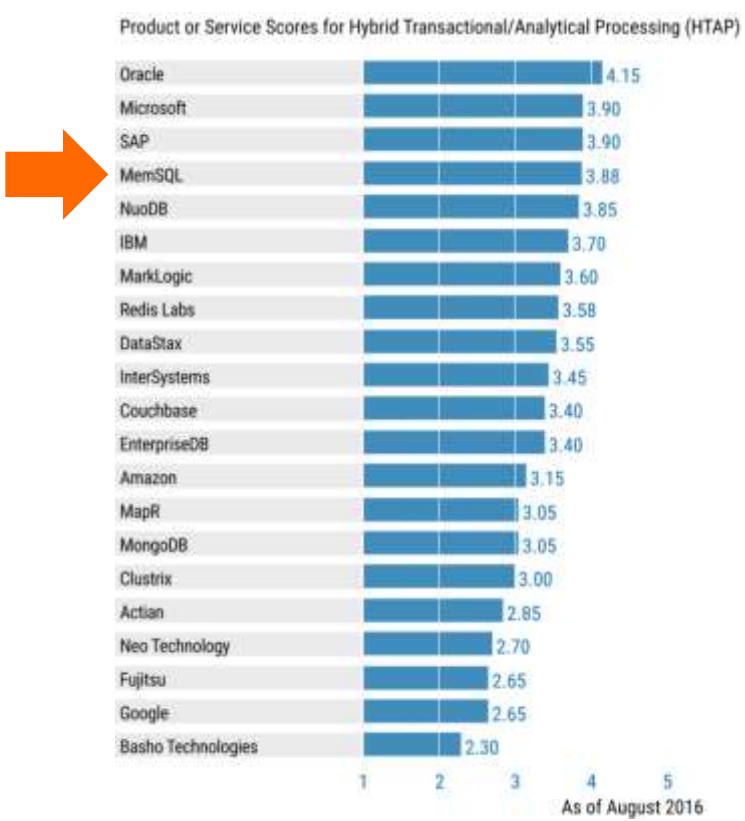
MemSQL: The Real-Time Data Warehouse

- Scalable
  - Petabyte scale
  - High Concurrency
  - System of record
- Real-time
  - Operational
- Compatible
  - ETL
  - Business Intelligence
  - Kafka (exactly-once)
  - Spark

- Deployment
  - On-premises
  - Any public cloud IaaS
  - MemSQL Cloud Service
- Developer Edition
  - Unlimited scale
  - Limited high availability and security features
- Enterprise Edition
  - Free 30 day trial

2017 Magic Quadrant for Data Management Solutions for Analytics

MemSQL also in Database MQ, HTAP focus



MEMSQL

Six Companies in BOTH the Database AND Data Warehouse Magic Quadrants

About Spark

Apache Spark™ is a fast and general engine for large-scale data processing

Source: spark.apache.org June 2017

MEMSQL

Understanding Spark and MemSQL

Spark

Fast, large scale

General processing engine

Great for computation, model training, classification

MemSQL

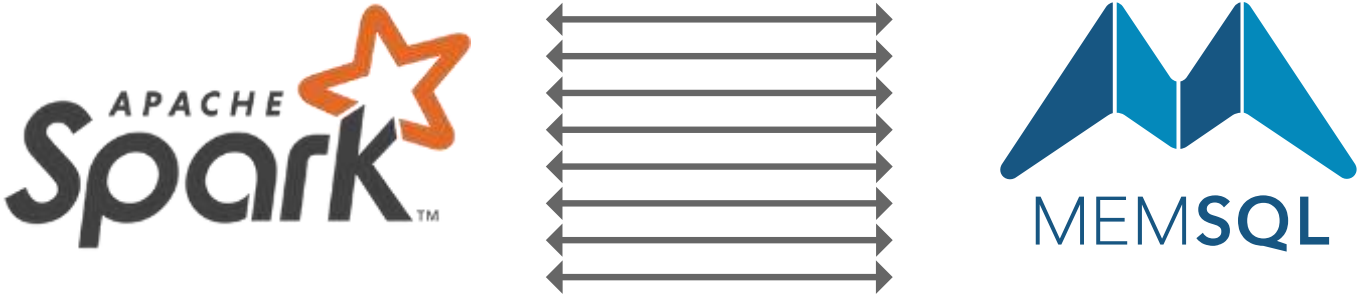
Fast, large scale

Real-time data warehouse

Great for SQL computation, persistence, transactions, applications, app analytics

MemSQL Spark Connector

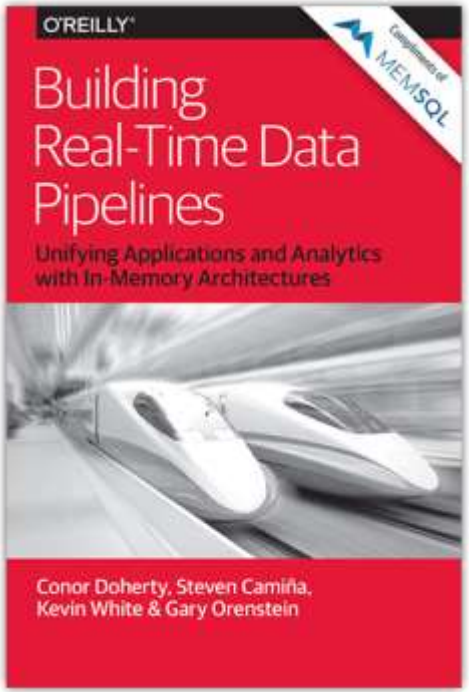
Highly parallel, high throughput, bi-directional



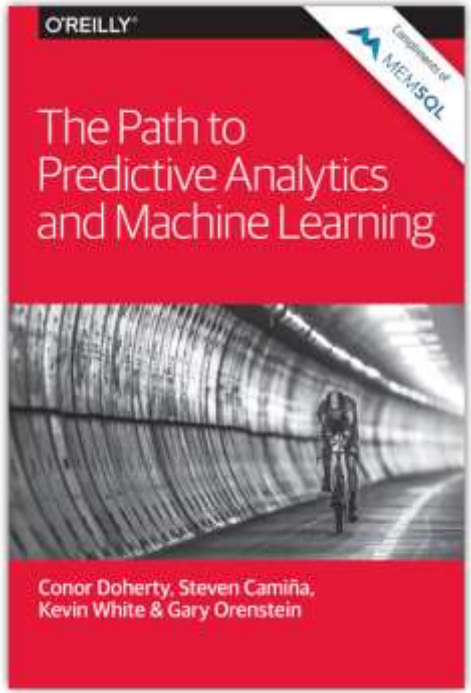
https://github.com/memsql/memsql-spark-connector

Complimentary ebooks

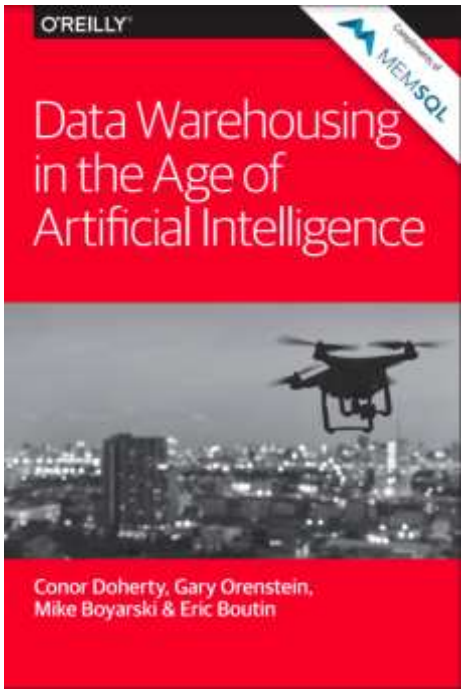
memsql.com/**oreilly**



memsql.com/**oreillyml**



memsql.com/**oreillyai**



MEMSQL

Thank you!

@garyorenstein

www.memsql.com

memsql.com/slideshare