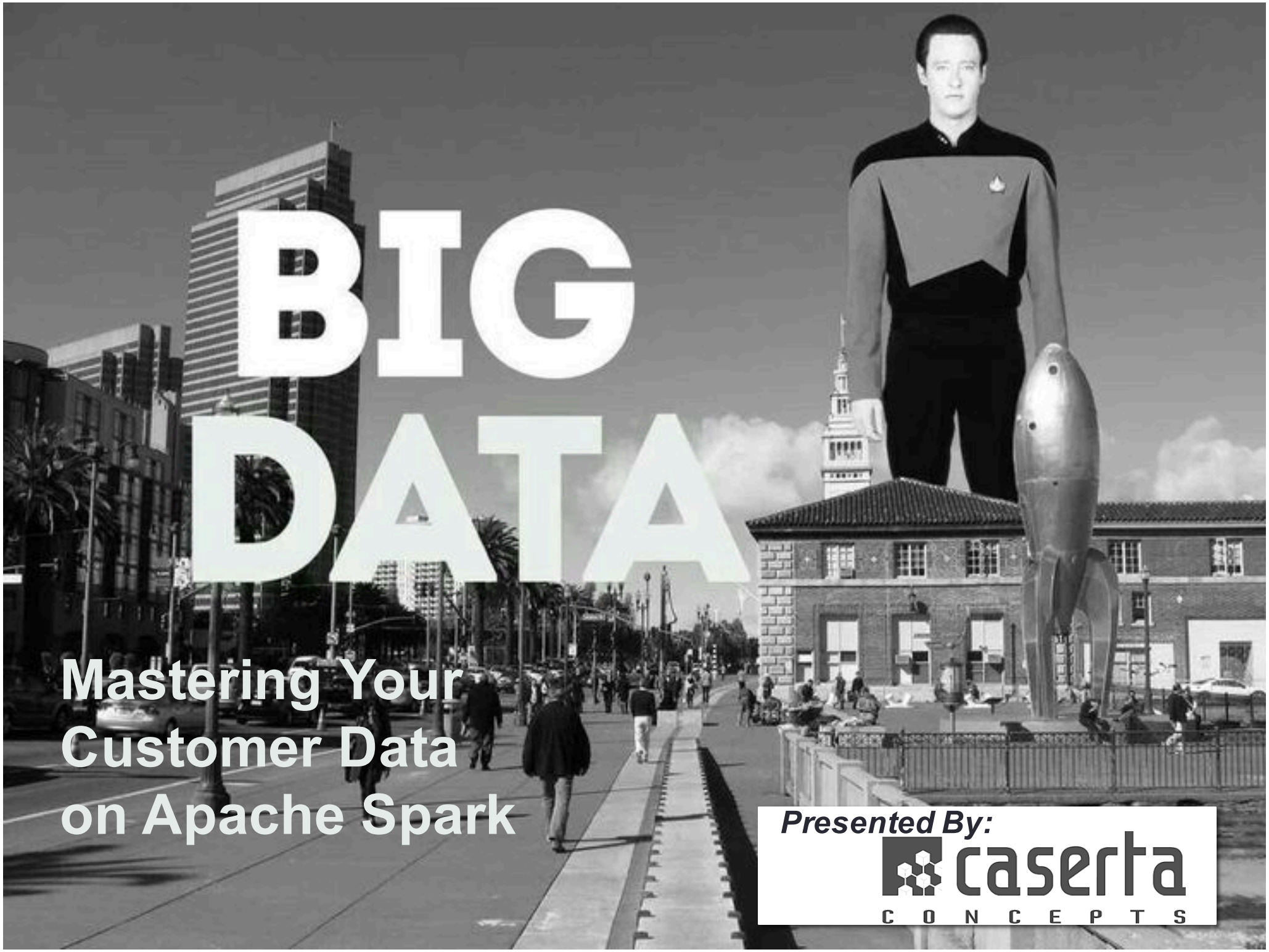
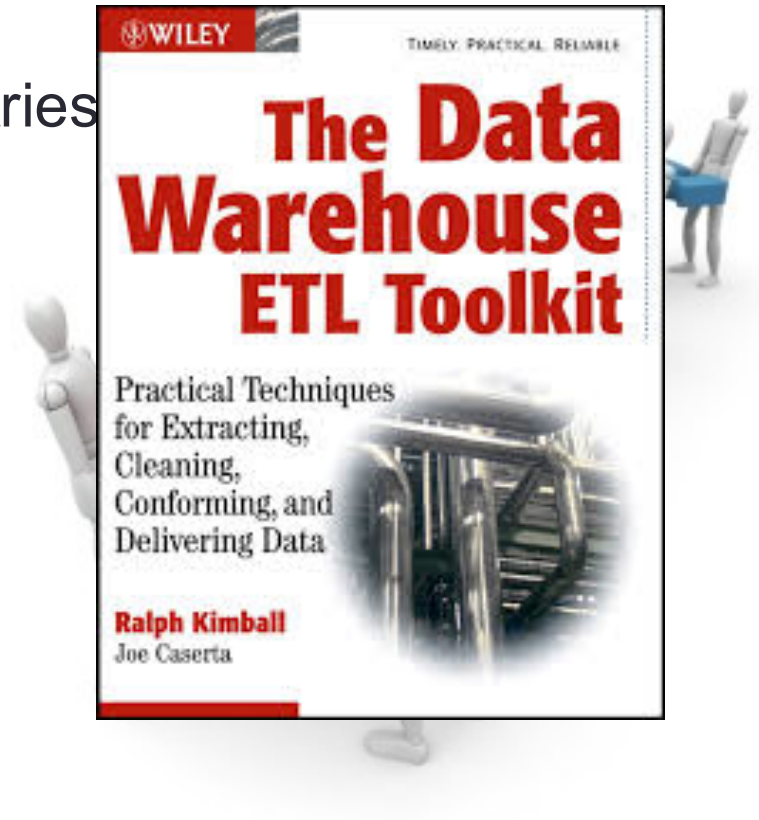

BIG
DATA
Mastering Your
Customer Data
on Apache Spark
Presented By:
caserta
CONCEPTS

About Caserta Concepts

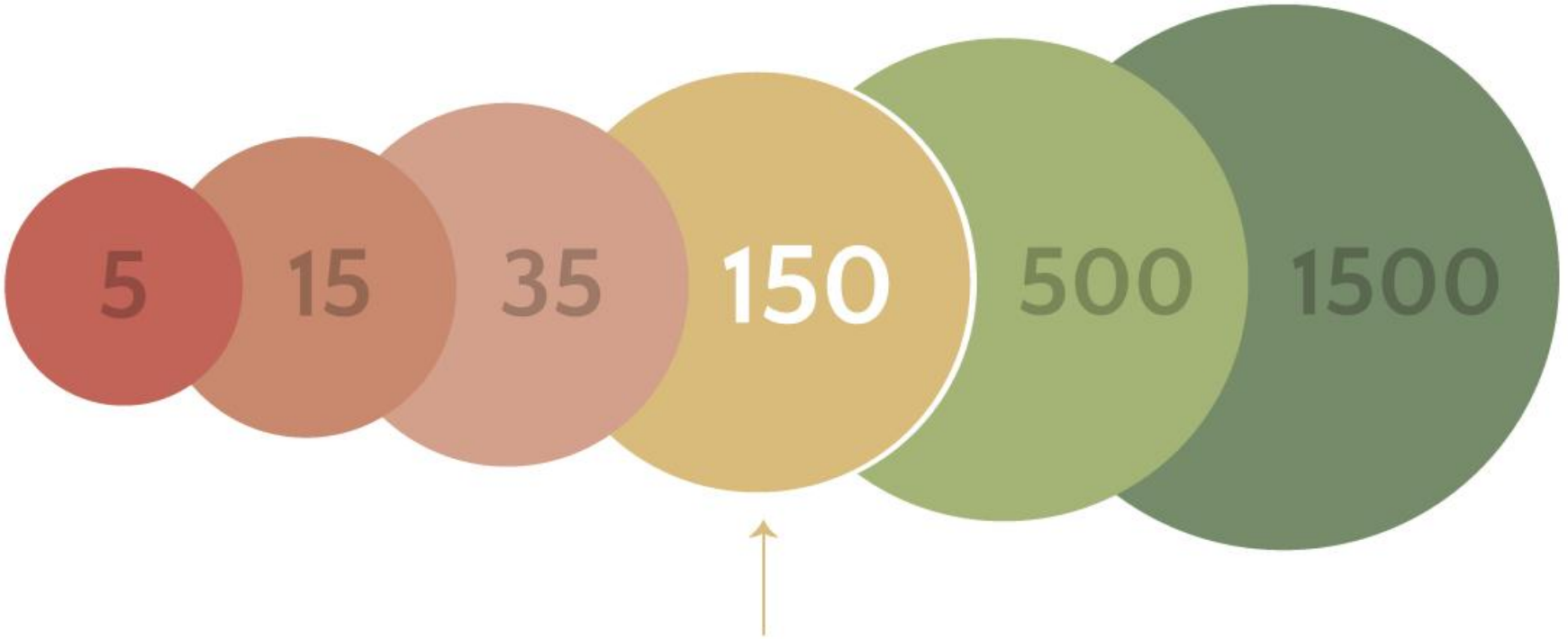
- Award-winning technology innovation consulting with expertise in:
  - Big Data Solutions
  - Data Warehousing
  - Business Intelligence
  - Data Science & Analytics
  - Data on the Cloud
  - Data Interaction & Visualization
- Core focus in the following industries
  - eCommerce / Retail / Marketing
  - Financial Services / Insurance
  - Healthcare / Ad Tech / Higher Ed
- Established in 2001:
  - Increased growth year-over-year
  - Industry recognized work force
  - Strategy, Implementation
  - Writing, Education, Mentoring



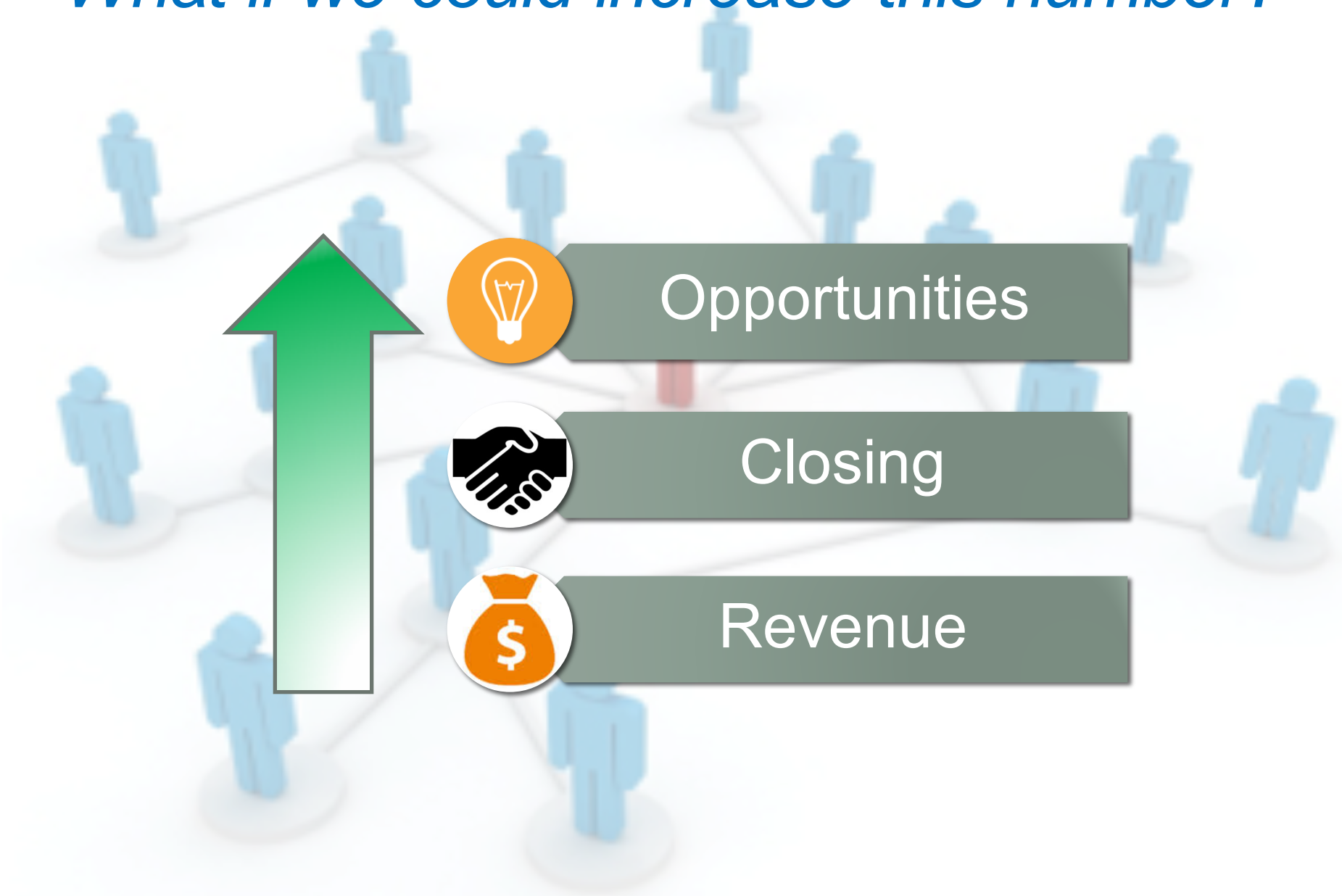
What motivated our solution

How many people do you know??

Anthropologists say it's 150…
MAX

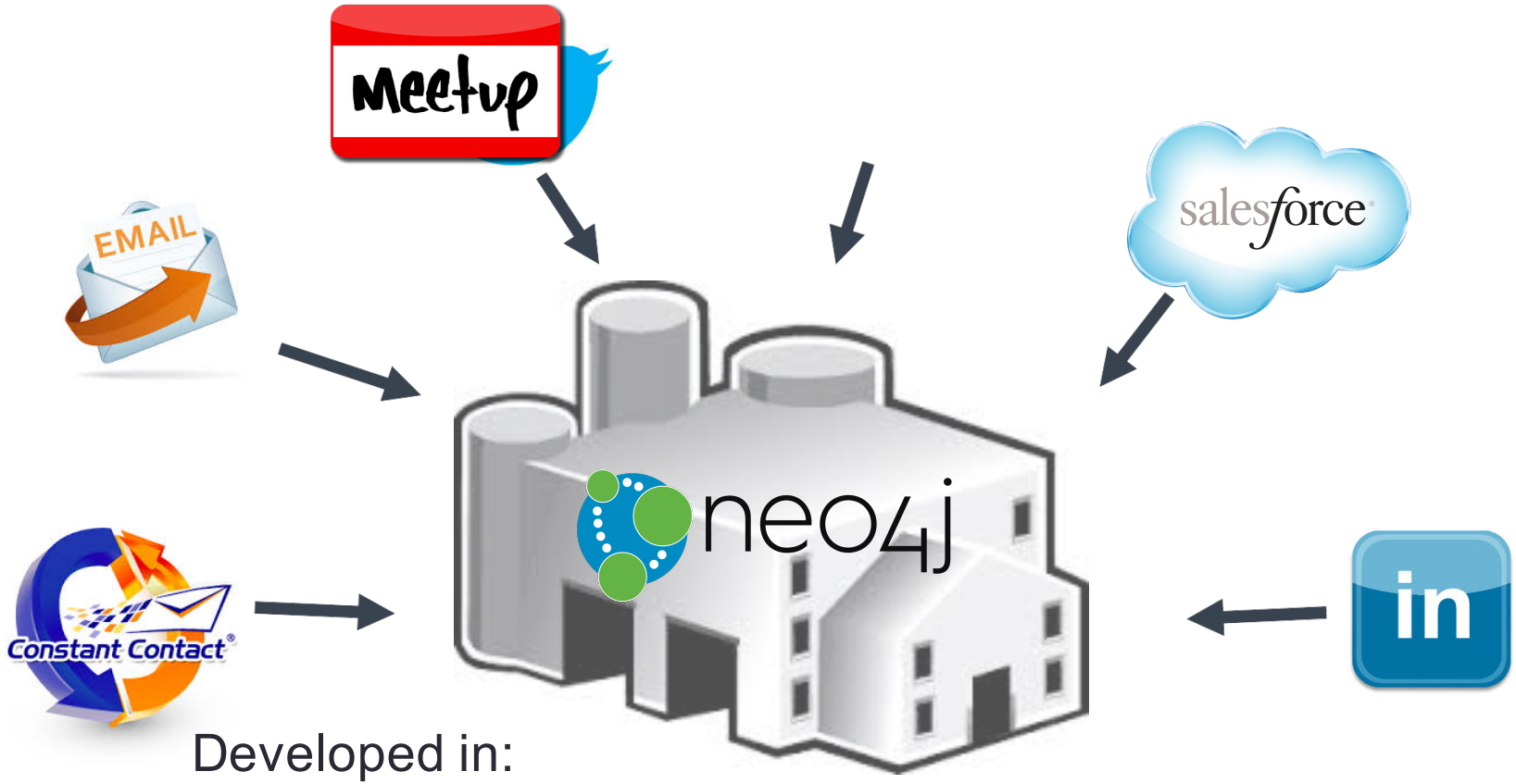


What if we could increase this number?



Project Dunbar - Internal

- Build a social graph based on internal and external data
- Run pathing algorithms to understand strategic opportunity advantages



Developed in:

- Python
- Neo4j Database

..and then

And then one of our customers wanted us to build it for them:

- Much larger dataset
- 6 million customers > 30% duplication rate
- 100's of millions of customer interactions
- Few direct links across channels

How to?

Throwing a bunch of unrelated points in a graph will not give us a useable solution.

- We need to clean and normalize our incoming interaction and relationship data (edges)
- Clean normalize and match our entities (vertexes)

We need to

MASTER

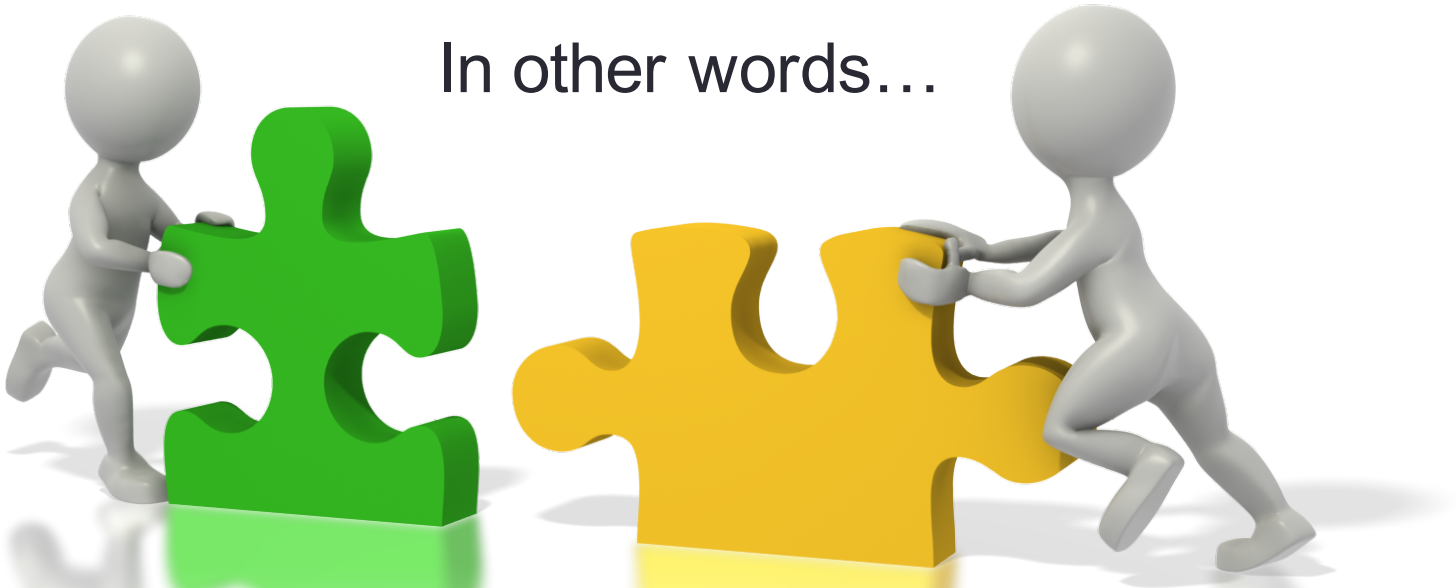
our contact data…

Mastering Customer Data

Customer Data Integration (CDI):

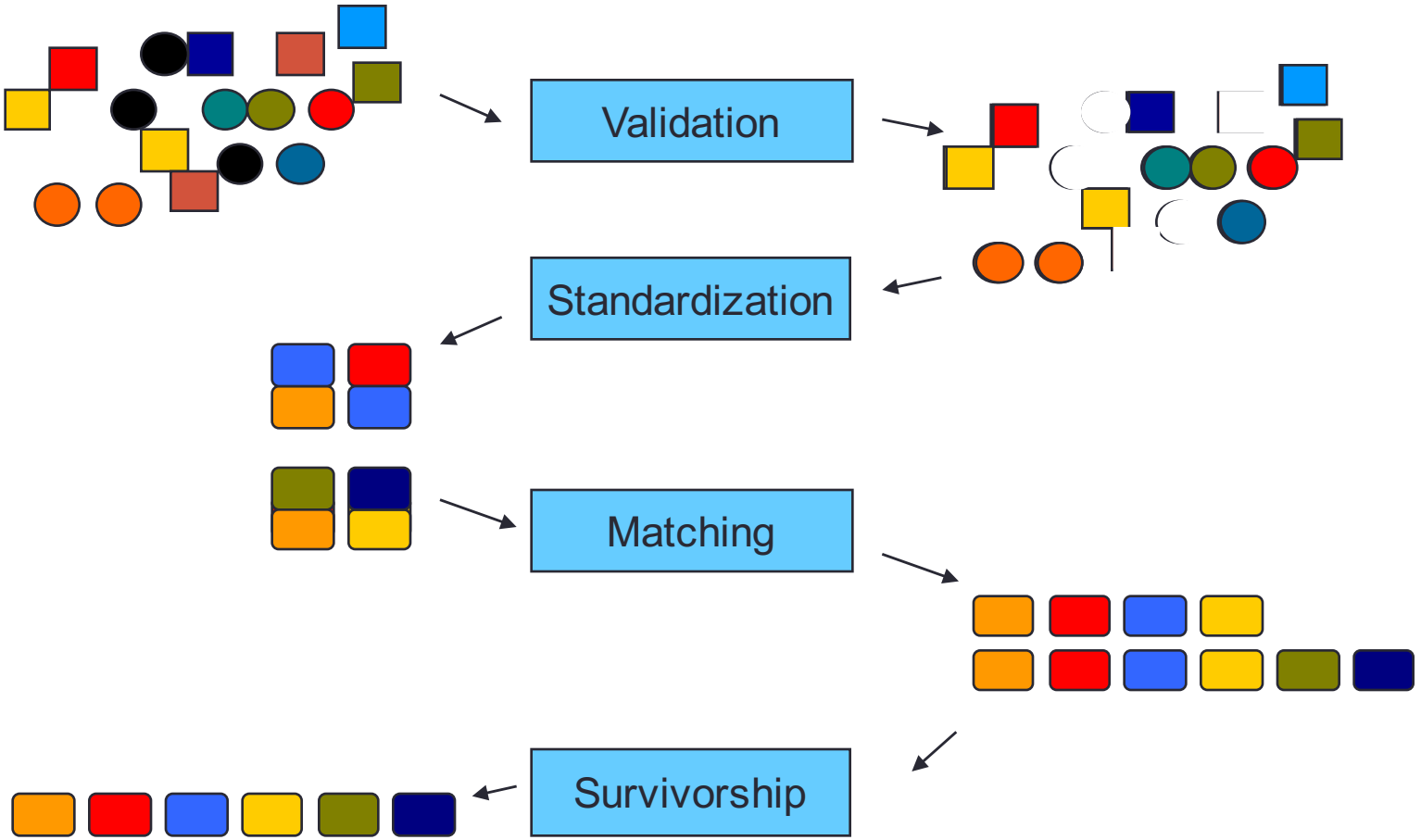
is the process of consolidating and managing customer information from all available sources.

In other words…



We need to figure out how to ***LINK*** people across systems!

Steps required



Traditional Standardization and Matching

Cleanse and Parse:

- Names
  - Resolve nicknames
  - Create deterministic hash, phonetic representation
- Addresses
- Emails
- Phone Numbers

Matching:

join based on combinations of cleansed and standardized data to create match results

Great – But the NEW data is different

Reveal

- Wait for the customer to "reveal" themselves
- Create a link between anonymous self and known profile

Vector

- May need behavioral statistical profiling
- Compare use vectors

Rebuild

- Recluster all prior activities
- Rebuild the Graph

Why Spark

"Big Box" MDM tools vs ROI?

- Prohibitively expensive → limited by licensing $$$
- Typically limited to the scalability of a single server

We ♥ Spark!

- Development local or distributed is identical
- Beautiful high level API's
- Databricks cloud is *soo* Easy
- Full universe of Python modules
- Open source and Free**
- Blazing fast!

Spark has become our *default* processing engine for a myriad of engineering problems

Spark map operations

Cleansing, transformation, and standardization of both interaction and customer data

Amazing universe of Python modules:

- Address Parsing: usaddress, postal-address, etc
- Name Hashing: fuzzy, etc
- Genderization: sexmachine, etc

And all the goodies of the standard library!

We can now parallelize our workload against a large number of machines:

```
import usaddress

parsed = df.map( lambda p: usaddress.tag(p))
```

Matching process

- We now have clean standardized, linkable data
- We need to resolve our links between our customer
- Large table self joins
- We can even use SQL:

```
create table matches
as
select c1.id as xid, c2.id as yid, "phone" as match_type
from customer_cleansed c1
   join customer_cleaned c2
     on c1.name_hash = c2.name_hash
       and c1.phone_std = c2.phone_std
 where c1.id <> c2.id
```

Matching process

The matching process output gives us the relationships between customers:

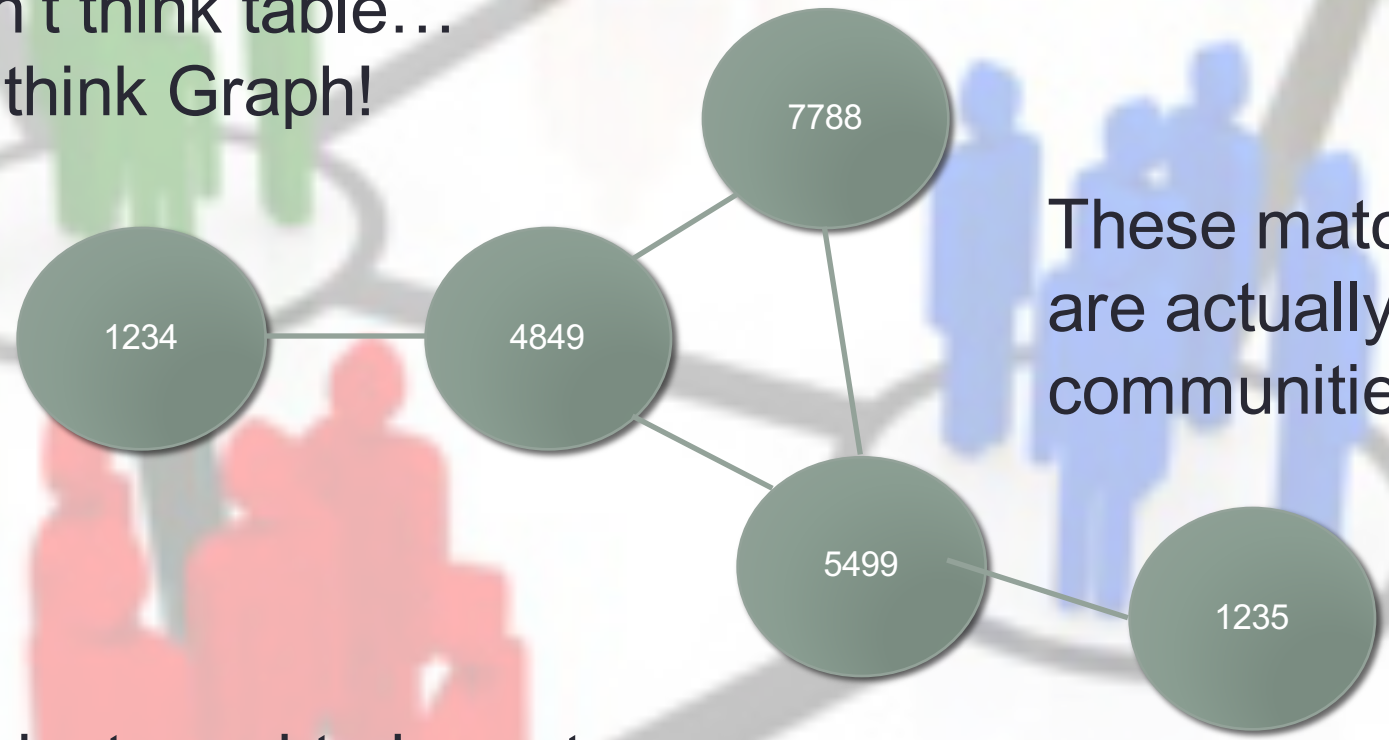
xid	yid	match_type
1234	4849	phone
4849	5499	email
5499	1235	address
4849	7788	cookie
5499	7788	cookie
4849	1234	phone

Great, but it's not very useable, you need to traverse the dataset to find out 1234 and 1235 are the same person (and this is a trivial case)

And we need to cluster and identify our survivors (vertex)

Graphx to the rescue

Don't think table…
think Graph!



These matches are actually communities

We just need to import our edges into a graph and "dump" out communities

Connected components

Connected Components algorithm labels each connected component of the graph with the ID of its lowest-numbered vertex

This lowest number vertex can serve as our “survivor” (not field survivorship)

Is it possible to write less code?

```
val graph = Graph(nodes_rdd, edges_rdd)

val cc = graph.connectedComponents().vertices
```

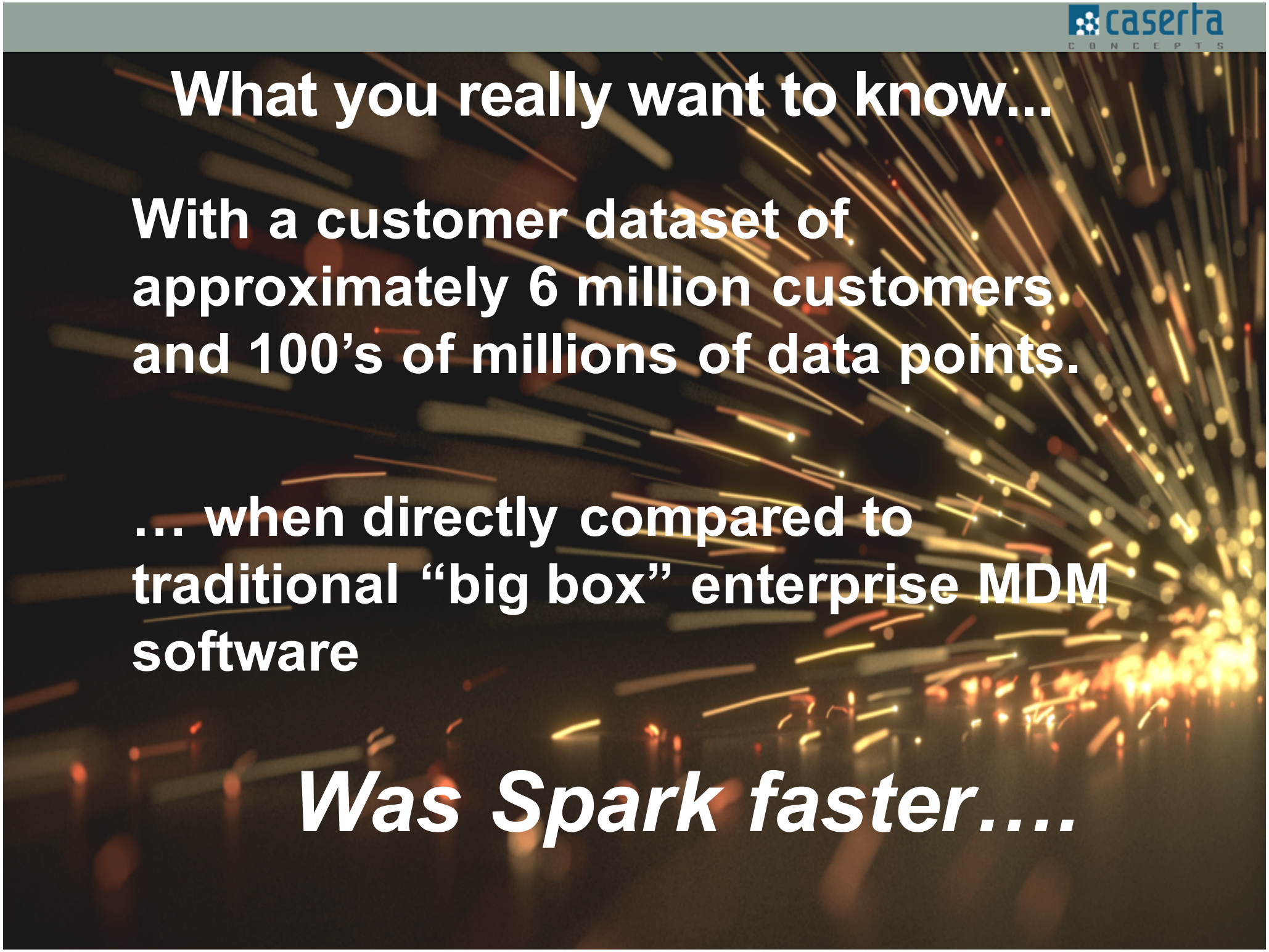
Field level survivorship rules

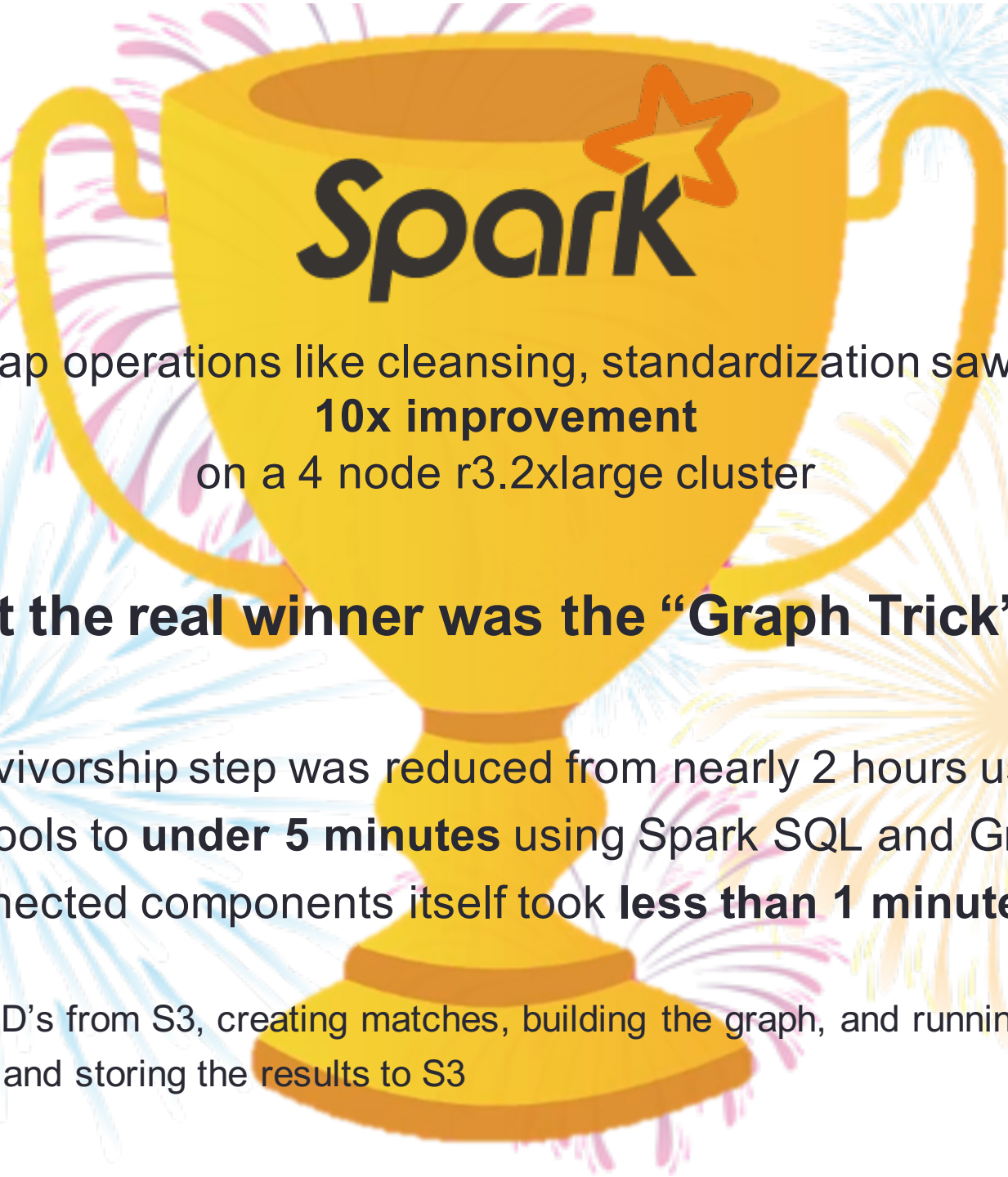
We now need to survive fields to our survivor to make it a "best record".

We then do some simple ranking (thanks windowed functions)

Depending on the attribution we choose:

- Latest reported
- Most frequently used


caserta
CONCEPTS
What you really want to know...
With a customer dataset of approximately 6 million customers and 100's of millions of data points.
… when directly compared to traditional "big box" enterprise MDM software
Was Spark faster….



Map operations like cleansing, standardization saw a
10x improvement
on a 4 node r3.2xlarge cluster

But the real winner was the "Graph Trick"…

The survivorship step was reduced from nearly 2 hours using traditional tools to **under 5 minutes** using Spark SQL and GraphX**
Connected components itself took **less than 1 minute**

**loading RDD's from S3, creating matches, building the graph, and running connected components and storing the results to S3

Community


BDW
Big Data Warehousing
Meetup


Sponsored By:
caserta CONCEPTS
Hortonworks
Big Data Warehousing
June 17, 2014
Today's Topic: YARN Tech Talk: The Data Operating System for Hadoop 2.0
BDW
Meetup


Big Data Warehousing Meetup
December 10, 2013
BDW
Meetup
caserta CONCEPTS
Real-time Trade Data Monitoring with Storm & Cassandra


Big Data Warehousing Meetup
Today's Topic: Introduction to NoSQL with 10Gen
BDW
Meetup
Sponsored By:
caserta CONCEPTS
10gen


Big Data Warehousing
February 10, 2014
BDW
Big Data Warehousing
Meetup
Today's Topic: Data Governance, Compliance and Security in Hadoop - with Cloudera
Sponsored By:
caserta CONCEPTS
cloudera


Big Data Warehousing Meetup
June 10, 2013
BDW
Meetup
Sponsored By:
caserta CONCEPTS
hortonworks


Big Data Warehousing Meetup
Today's Topic: Building a Relevance Engine using Hadoop, Mahout & Pig
JOE'S PUB
BDW
Meetup
Sponsored By:
caserta
LucidWorks


Big Data Warehousing Meetup
September 23, 2013
BDW
Meetup
Today's Topic: Processing 1.4 Trillion Events in Hadoop
Sponsored By:
caserta CONCEPTS
syncsort


Big Data Warehousing Meetup - April 8, 2014
Building a Big Data Warehouse on the Cloud in 30 Minutes
caserta CONCEPTS


Big Data Warehousing Meetup
Today's Topic: Exploring Big Data Analytics Techniques with Datameer
BDW
Meetup
Sponsored By:
caserta CONCEPTS
Datameer


Big Data Warehousing
January 20, 2014
Today's Topic: Big Data 2.0: YARN Distributed ETL & SQL with Hadoop
BDW
Meetup
Sponsored By:
caserta CONCEPTS
Actian


Big Data Warehousing
May 27, 2014
Today's Topic: DATA SCIENCE with Claudia Perlich and Revolution Analytics
BDW
Meetup
Sponsored By:
caserta CONCEPTS
REVOLUTION ANALYTICS

Thank You

Elliott Cordo
Chief Architect, Caserta Concepts
elliott@casertaconcepts.com

Kevin
Data Engineer
kevin@casertaconcepts.com

caserta CONCEPTS
info@casertaconcepts.com
1 (855) 755-2246
www.casertaconcepts.com