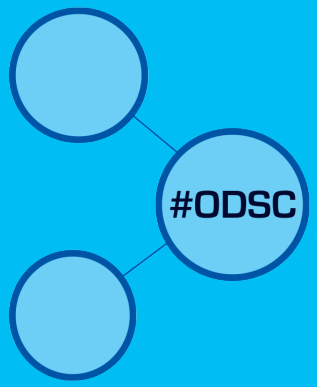
OPEN
DATA
SCIENCE
CONFERENCE_



BOSTON 2015

@opendatasci

Spark, Python, and Parquet

Learn How to Use Spark, Python, and Parquet for Loading and Transforming Data in 45 Minutes

May, 2015

Douglas Eisenstein - Advanti
Stanislav Seltser - Advanti

Agenda

Use Case Background

What's Spark and Parquet?

Demo: code + data = fun part

Questions

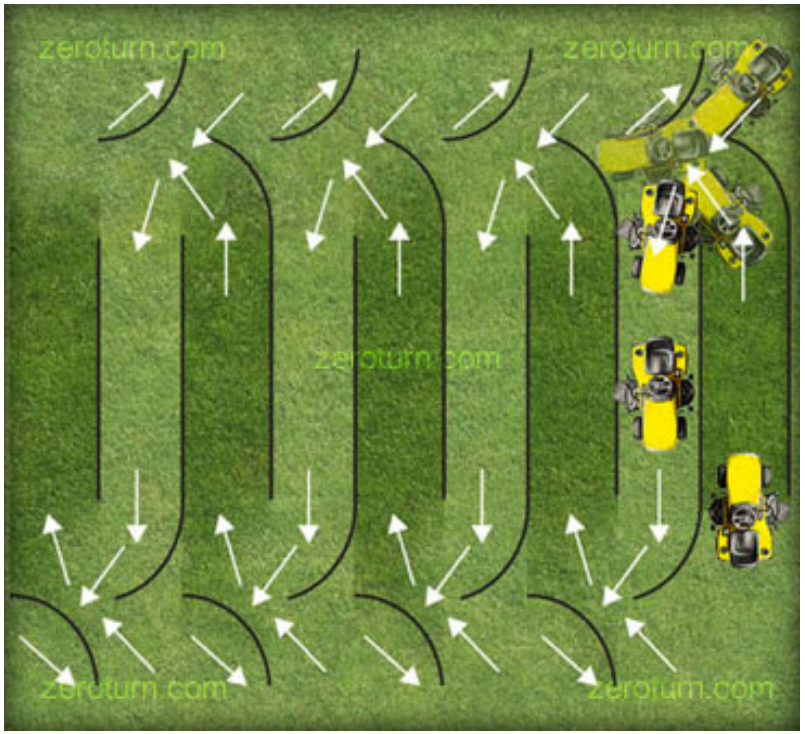
Learn new technologies and be motivated to start using them

TAKEAWAYS: WHAT WILL YOU LEARN TODAY?

In about the same time it will take you to mow your lawn…

Takeaways: you will learn …

1. What is Spark and Parquet and why to consider it?
2. A live demo showing you 5 useful transforms in Spark
3. Instructions for DIY cluster: Spark, Hadoop, Hive, Parquet, and C*
4. "Take home" fund holdings open dataset from Morningstar



What's our use case? Why choose Spark?

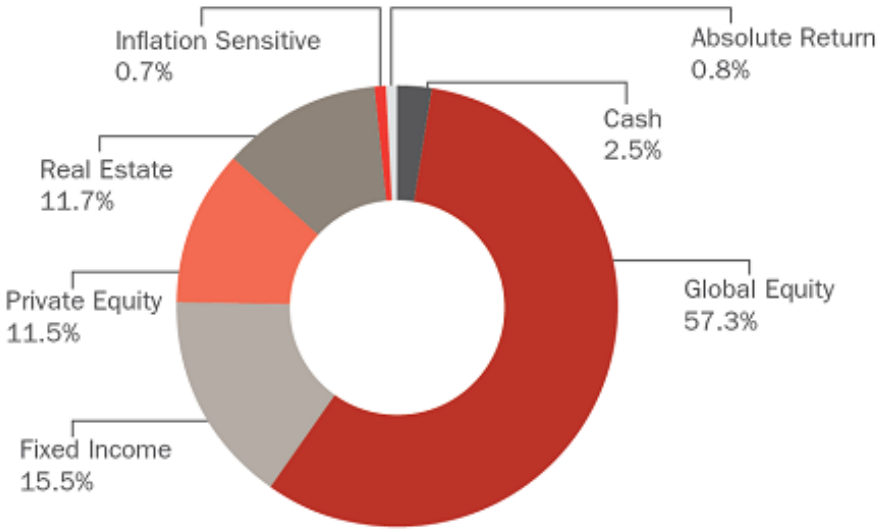
USE CASE: DATASET, TRADEOFFS, SOLUTION

Holdings Data

"The contents of an investment portfolio held by an individual or entity such as a mutual fund or pension fund." – Investopedia 2015

Sources:

- 3rd Parties: Morningstar, Lipper, FactSet, Bloomberg, Thomson Reuters
- Internal: Specialized portfolio accounting systems per asset type
- Public: SEC 13F's when AUM is >= $100M, includes hedge funds



Use Case: Challenges

1. Millions of files, Reshape data, pre-aggregate holdings, disambiguate securities, unstack/stack data

2. Create wide tables (think columnar) for storing time series data about 1M instruments over 12 months for 381k funds

3. Rules are abstracted to work on "holdings" making them vendor-agnostic and reusable (13F's, 3rd parties, proprietary, etc)

4. All sorts of "data usability" issues: missing positions, missing identifiers, sparse derivative data, irregular fund reporting, etc

5. Need to create your own "ownership views" by asset class and period (ex. Fixed Income Ownership View)

Open Dataset Description

Open Dataset: open-ended funds for report period 2014-06-10

Universes Available
- Open-end Mutual Funds
- Closed-end Funds
- Exchange-traded Funds
- 529 College Savings Plans
- Separate Accounts
- Variable Annuities

Key data points
- Holding Ticker/ID
- Holding Name
- Holding Type
- Market Value
- % Asset Weight
- Maturity Date
- Coupon
- Sector
- Region
- # of Shares

Applications
- Analyzing peers and competitors
- Monitoring investment style
- Populating Web sites, tools, and statements
- Developing accurate asset allocation models

MORNINGSTAR®

Morningstar® Full Holdings Data License

Key Points
- Holdings are electively contributed for fresher ranks
- Long and Short positions included, unique…
- Datasets: free open data, trial data, paid licensed data
- Our dataset starts in 2014-01 across all fund types

Descriptive Statistics
- History starts in 2000+
- Fund types: Open, Closed, ETFs, SMAs
- 381k funds since 2014
- 1M unique securities since 2014
- 82 legal types, ex Corporate Bond or Equity Swap

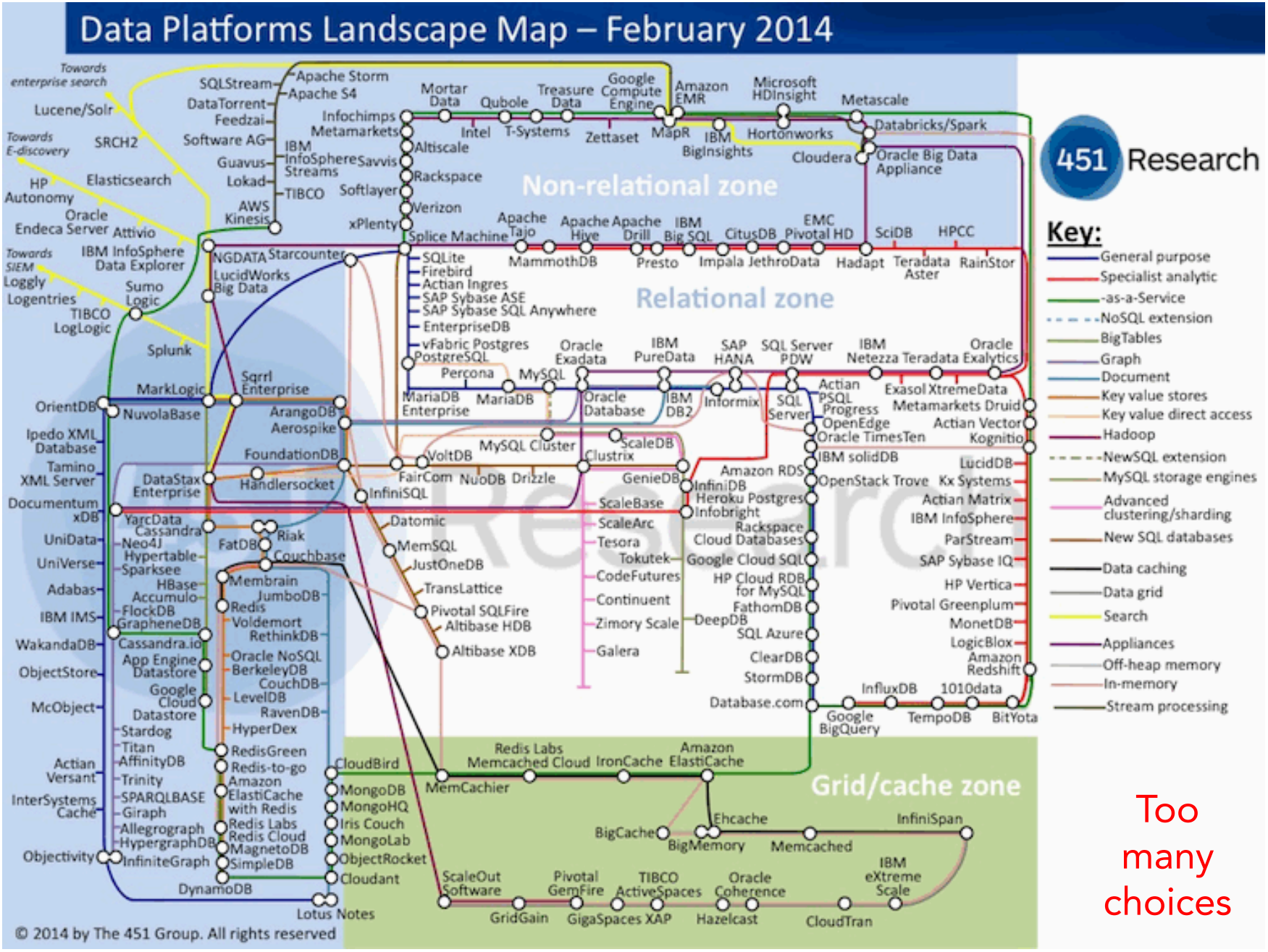
MORNINGSTAR®

Common Usages

1. Ownership: Top N largest long/short positions per asset class
2. Flow: Compute holdings flow across various financial assets
3. Comparison: Peer group analysis through holdings attribution

p_date	iv_id	cusip	isin	sedol	symbol	sector	country	legaltype	weighting	numberofshare	marketvalue
2013-11-27 00:00:00.0	F00000MJP8	J27743103		B05PDH7	JPSWF	10	Japan	E	3	14000	-7546000
2013-11-27 00:00:00.0	F00000MJP8	J74358144		B01F3C6	SSDOF	5	Japan	E	3	3000	-5775000
2013-11-27 00:00:00.0	F00000MJP8	654111103		B020SL4	NINOF	11	Japan	E	2	2000	-5573000
2013-11-27 00:00:00.0	F00000MJP8	J77884112		B02LM82	SMTUF	2	Japan	E	2	3000	-5289000
2013-11-27 00:00:00.0	F00000MJP8	J3590M101		B05PDW2	8218	2	Japan	E	2	2000	-5100000
2013-11-27 00:00:00.0	F00000MJP8	J1257M109		B3BH2R1	EJPRF	10	Japan	E	2	600	-5088000
2013-11-27 00:00:00.0	F00000MJP8	J76379106		B01DR28	SNEJF	11	Japan	E	2	2000	-5046000
2013-07-16 00:00:00.0	F000000N9W	J12852117		B01DGT8	ESALF	6	Japan	E	2	2000	-12441000
2013-07-16 00:00:00.0	F000000N9W	J77282119		B02LLM9	SSUMF	10	Japan	E	2	9000	-11957000
2013-07-16 00:00:00.0	F000000N9W	J39788138		B02HT01	MARUF	10	Japan	E	2	14000	-10066000
2013-07-16 00:00:00.0	F000000N9W	J74358144		B01F3C6	SSDOF	5	Japan	E	2	6000	-9207000
2013-07-16 00:00:00.0	F000000N9W	J30642169		B01DFC4	KAOCF	5	Japan	E	2	2000	-8625000
2013-07-16 00:00:00.0	F000000N9W	J72810120		B02LJ25	SHECF	1	Japan	E	2	1000	-8388000
2013-07-16 00:00:00.0	F000000N9W	J94104114		B02NJV0	UNCHF	5	Japan	E	2	1000	-8250000
2013-07-16 00:00:00.0	F000000N9W	J2501P104		B02H2R9	ITOCF	10	Japan	E	2	6000	-8177000
2013-07-16 00:00:00.0	F000000N9W	J69972107		B018RR8	SOMLF	10	Japan	E	1	1000	-7966000
2013-07-16 00:00:00.0	F000000N9W	J29051109		B02HL94	2811	5	Japan	E	1	4000	-7299000
2013-07-16 00:00:00.0	F000000N9W	J7165H108		B0L4N67	SVNDF	5	Japan	E	1	1000	-7137000
2013-07-16 00:00:00.0	F000000N9W	J05124144		B021CR1	CAJFF	10	Japan	E	1	2000	-6920000
2013-07-16 00:00:00.0	F000000N9W	J2224T102		6432715	1379	5	Japan	E	1	3000	-6798000
2013-07-16 00:00:00.0	F000000N9W	J32491102		B02HPZ8	KYCCF	11	Japan	E	1	200	-6730000
2013-07-16 00:00:00.0	F000000N9W	J8657U110		B3BHHW1	TGRLF	9	Japan	E	1	7000	-6727000
2013-07-16 00:00:00.0	F000000N9W	J89494116		B02MH57	TRYIF	1	Japan	E	1	10000	-6630000
2013-07-16 00:00:00.0	F000000N9W	J77411114		B02LLQ3	SMTOF	11	Japan	E	1	4000	-6316000
2013-07-16 00:00:00.0	F000000N9W	J22848121		B0L2KY8		11	Japan	E	1	2000	-6235000
2013-07-16 00:00:00.0	F000000N9W	J11151107		B050736	DITTF	2	Japan	E	1	700	-6167000
2013-07-16 00:00:00.0	F000000N9W	J52968104		B05PHB9	NNDNF	10	Japan	E	1	800	-5808000
2013-07-16 00:00:00.0	F000000N9W	J86957115		B02LVL8	TOELF	11	Japan	E	1	1000	-5588000
2013-07-16 00:00:00.0	F000000N9W	J43873116		B16TL60	MIELF	11	Japan	E	1	5000	-5460000
2013-07-16 00:00:00.0	F000000N9W	J12075107		B021NW3	DNZOF	2	Japan	E	1	1000	-5417000
2013-07-16 00:00:00.0	F000000N9W	J96612114		B0507F8	YATRF	10	Japan	E	1	2000	-5373000

What's the best part about starting a new project? You get to "play" with new tech toys right? But which ones….?

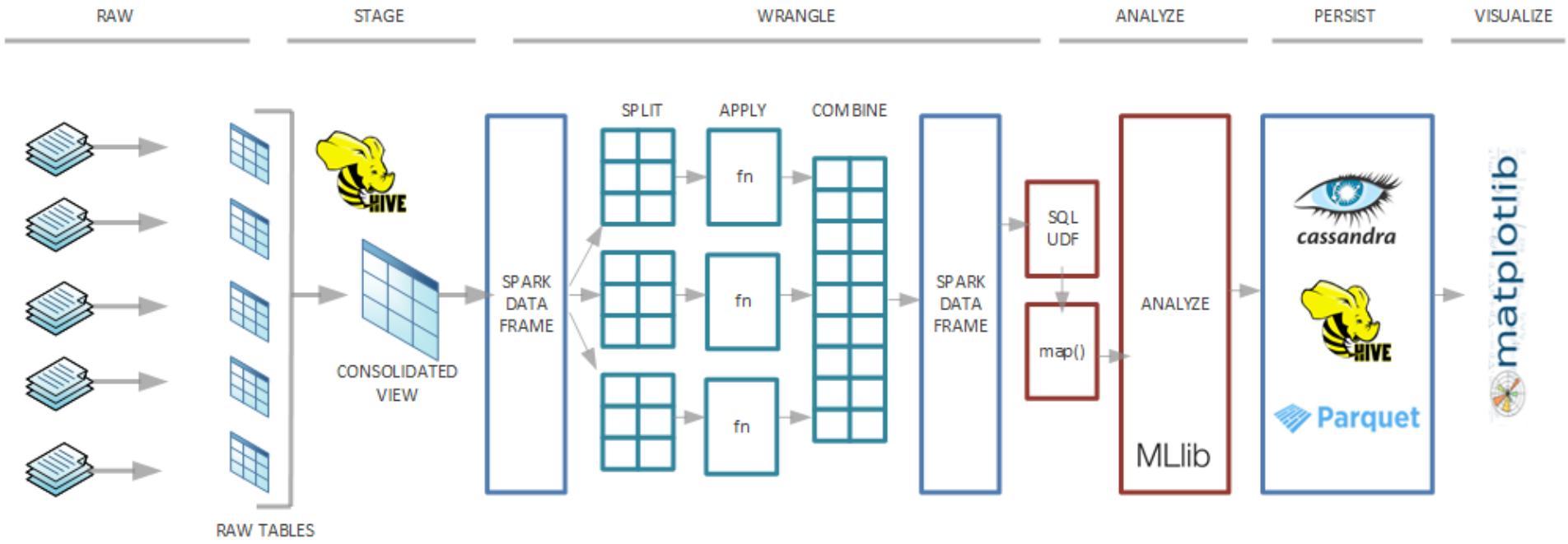

Data Platforms Landscape Map – February 2014
451 Research
Non-relational zone
Relational zone
Grid/cache zone
Key:
General purpose
Specialist analytic
-as-a-Service
NoSQL extension
BigTables
Graph
Document
Key value stores
Key value direct access
Hadoop
NewSQL extension
MySQL storage engines
Advanced clustering/sharding
New SQL databases
Data caching
Data grid
Search
Appliances
Off-heap memory
In-memory
Stream processing
Too many choices
© 2014 by The 451 Group. All rights reserved

Modern General Purpose Data Pipeline

Why:

1. Spark for in-memory transforms/analytics and fast exploration
2. Parquet for fast columnar data retrieval
3. Python as the glue layer and to re-use data transforms

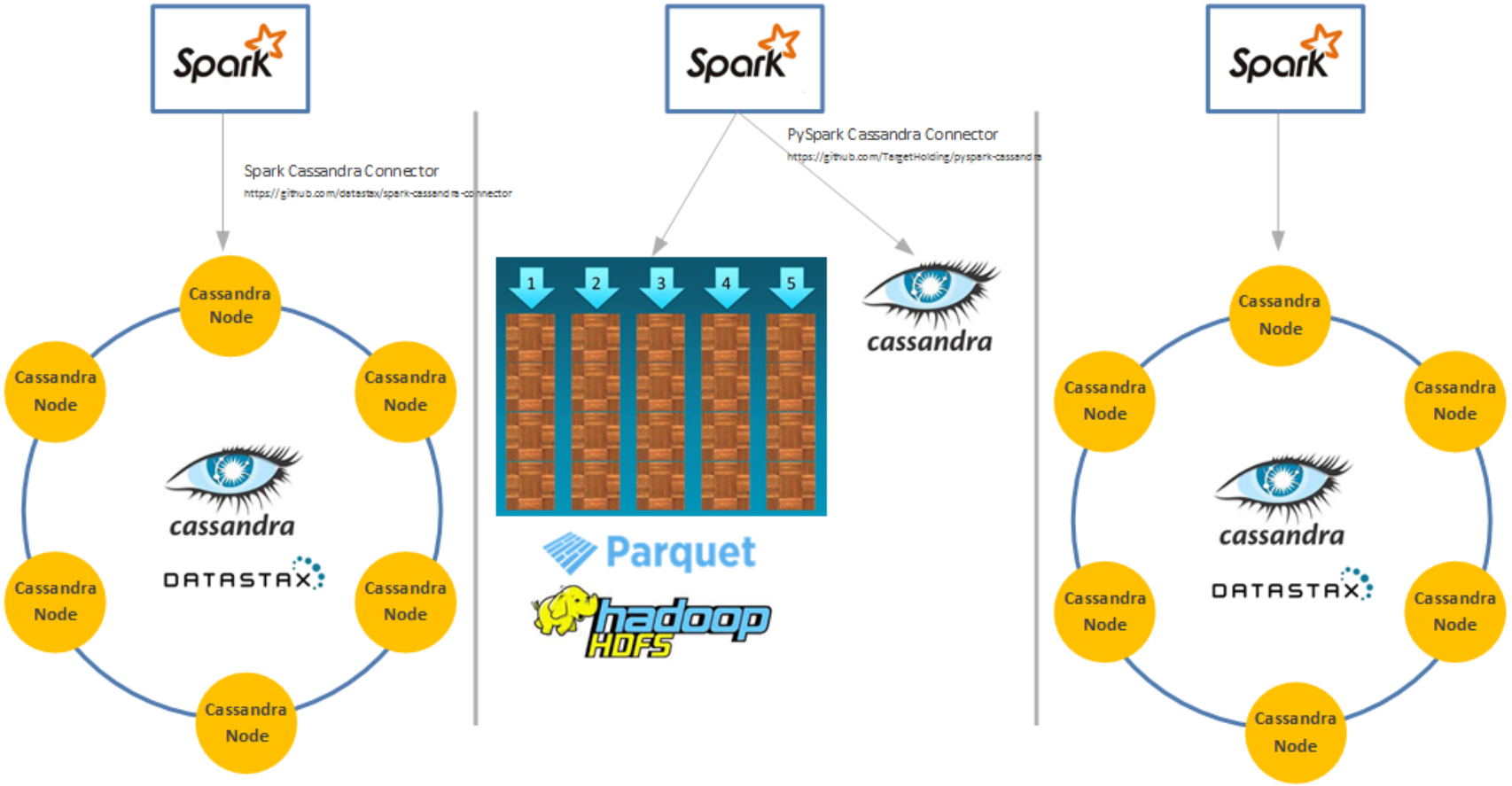
Data Pipeline:



Granular Options

Remember… Technology moves quickly!

Bleeding Edge ;)



Tradeoffs

Pro's	Con's
-DataStax Community -R/W to Cassandra	-No PySpark yet ☹ -No DataFrame yet ☹ -Connector is tricky -Burned 3 days

Tradeoffs

Pro's	Con's
-Simple, just works -Columnar storage -Universal format -Strong following	-Parquet limited dtypes -No Cassandra support -Early stage, but traction

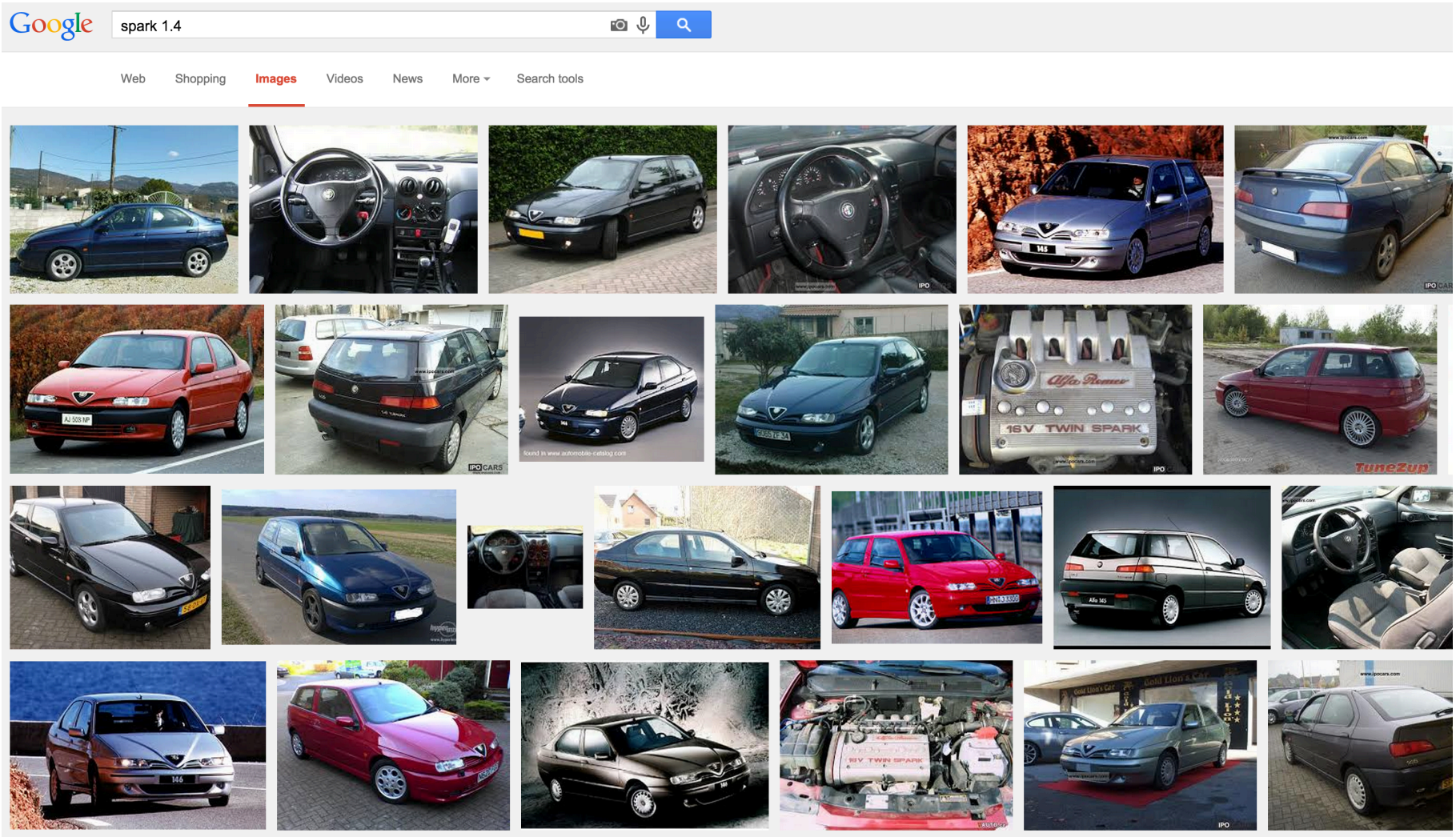
Tradeoffs

Pro's	Con's
-PySpark enabled -Easy to use -DSE "start-up" program -Great support	-Locked into DataStax -DSE costs $$ -No DataFrames

Everyone hears about it, but what is Spark?

SPARK: WHAT IS IT?

If you do any Googling, don't search for Spark 1.4

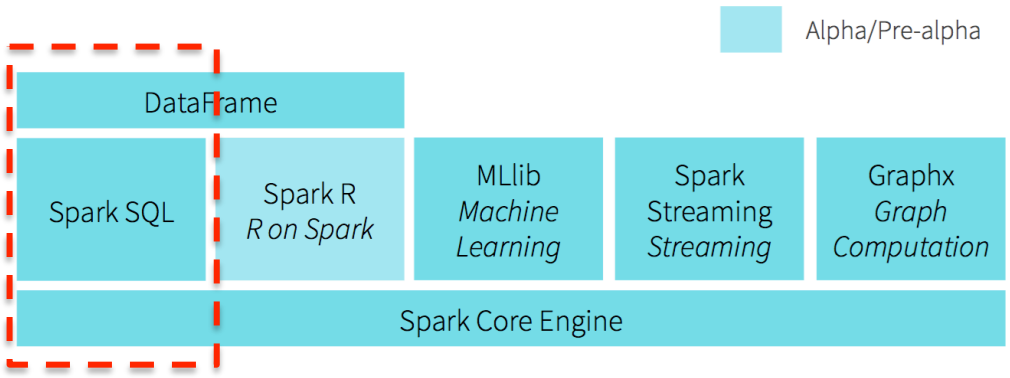


What is Spark?

Highlights
- General purpose data processing engine
- In-memory data persistence
- Interactive shells in Scala and Python

Key Concepts
- RDD: Collections of objects stored in RAM or on Disk
- Transforms: `map(), filter(), reduceByKey(), join()`
- Actions: `count(), collect(), save()`

Spark SQL
- Runs SQL or HQL queries
- In-line SQL UDF
- Distributed DataFrame
- Parquet, Hive, Cassandra



Graphic sourced: https://goo.gl/fpLrSs

What is a Spark DataFrame?

What's DataFrame?

- A collection of rows organized into named columns
- Expressive language for wrangling data into something usable
- Ability to select, filter, and aggregate structured data

What's a Spark DataFrame?

- Distributed collection of data organized into named columns
- Conceptually equivalent to a table in a relational database
- Relational data processing: project, filter, aggregate, join, etc
- Operations: `groupBy()`, `join()`, `sql()`, `unique()`
- Create UDF's and push them into SparkSQL

```
# How many holdings are there per fund and holding period
holdings_group = holdings.groupBy("iv_id", "p_date") # group by
holdings_group.count().orderBy(desc("count")).show(10)
```

```
iv_id      p_date              count
F000000MEI 2013-05-31 00:00:... 1673
F000000MOM 2013-07-18 00:00:... 1400
F000000N06 2013-05-27 00:00:... 1382
F000000M1V 2013-10-15 00:00:... 1084
F000000NBQ 2013-11-19 00:00:... 725
F000000MGJ 2014-03-17 00:00:... 698
F000000NPA 2013-09-24 00:00:... 674
F000000NHU 2014-03-31 00:00:... 576
F000000MVL 2013-11-20 00:00:... 565
F000000NJN 2013-05-27 00:00:... 524
```

Distributed Data Aggregation

- Distributed high-performance data, once only available by expensive appliances: Netezza, GreenPlum, Exadata, Teradata, etc
- Code 1-liner doing a SUM() between Pandas (standalone) and Spark (distributed)

Pandas DataFrame

```
# Spark DataFrame: total market value by fund, report date, and asset class of holding position
frame = holdings.toPandas()
frame.groupby(["iv_id", "p_date", "legaltype"])["marketvalue"].sum().head(15)
```

```
iv_id       p_date      legaltype
F000000LVG  2013-08-26  C              88418629
                        EX           4384662000
                        FO             71680000
F000000LVH  2013-06-12  C             350095155
                        FO           4331150000
F000000LVI  2013-12-10  C             287395141
                        EX          20510016000
                        FO            175789000
F000000LVJ  2013-11-25  C             335808813
                        EX          47814529000
                        FO            532038000
F000000LVK  2013-06-25  C              70710234
                        E             334228000
F000000LVN  2013-12-02  C                708018
                        FO           1069759000
Name: marketvalue, dtype: int64
```

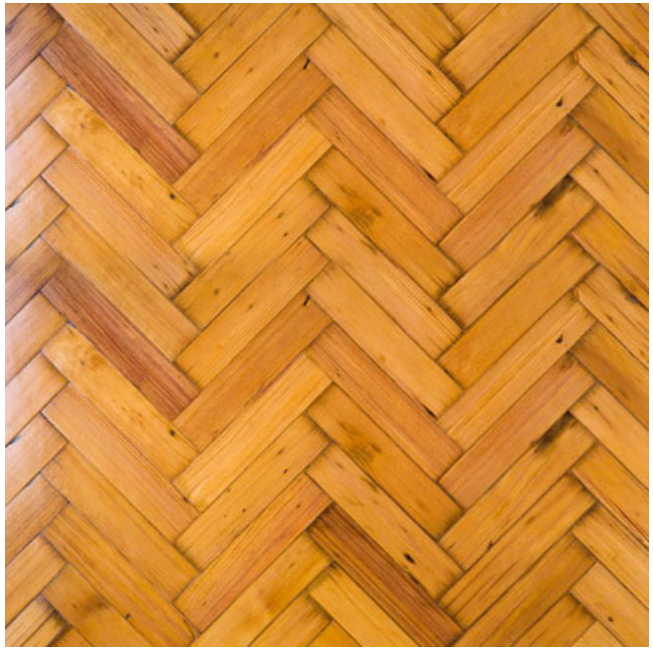
Spark DataFrame

```
holdings.groupBy("iv_id", "p_date", "legaltype").agg({"marketvalue": "sum"}).show()
```

```
iv_id      p_date               legaltype SUM(marketvalue#34L)
F00000P7R8 2013-09-19 00:00:... C         8298843
F00000LKJW 2013-09-06 00:00:... FO        296779000
F000000MQE 2013-07-01 00:00:... FO        11626784000
F0000022M0 2013-12-24 00:00:... C         3216779
F00000P05T 2013-09-12 00:00:... FO        1683805000
F00000LI63 2013-11-15 00:00:... FO        2078879000
F00000NG8O 2013-12-09 00:00:... EX        281268244
F00000PV9P 2013-10-15 00:00:... EX        1230496000
F00000N86T 2013-11-15 00:00:... C         22508
F00000J7GW 2013-09-20 00:00:... EX        36670869000
F00000O5F8 2014-02-24 00:00:... FO        142000
F000000N6G 2013-09-17 00:00:... FO        3235735000
F00000P05M 2014-02-17 00:00:... FO        33215000
F00000OS1D 2013-09-09 00:00:... Q         814095776
F00000Q7LV 2013-11-21 00:00:... EX        1577666575
F00000H68X 2014-03-17 00:00:... C         25011628
F000005HF2 2013-10-15 00:00:... FO        1504740833
F000000N63 2013-04-12 00:00:... FO        497748000
F000000MC7 2014-04-21 00:00:... C         95199322
F000000MQF 2013-11-20 00:00:... FO        13776424621
```

No, it's not your Grandma's flooring…

PARQUET: WHAT IS IT?



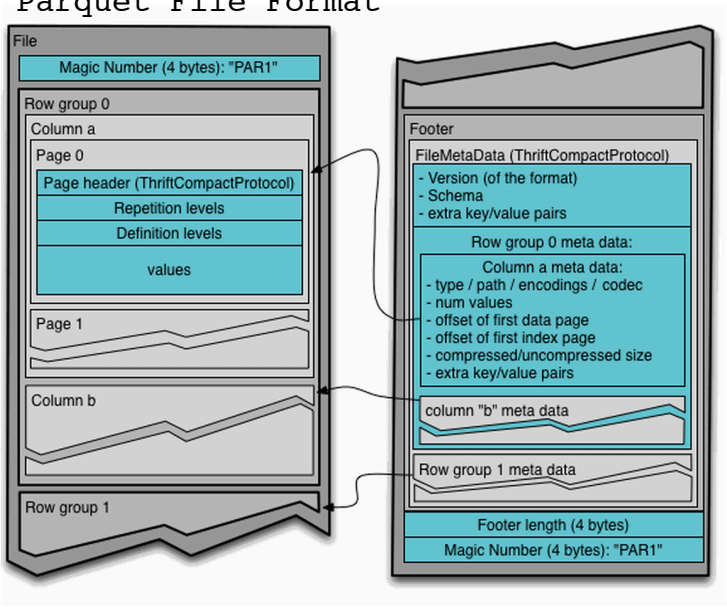
What is Parquet?

"Apache Parquet is a columnar storage format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language." – parquet.apache.org

Features

- Columnar File Format
- Supports Nested Data Structures
- Not tied to any commercial framework
- Accessible by HIVE, Spark, Pig, Drill, MR
- R/W in HDFS or local file system
- Gaining Strong Usage…

Parquet File Format



Parquet in HDFS

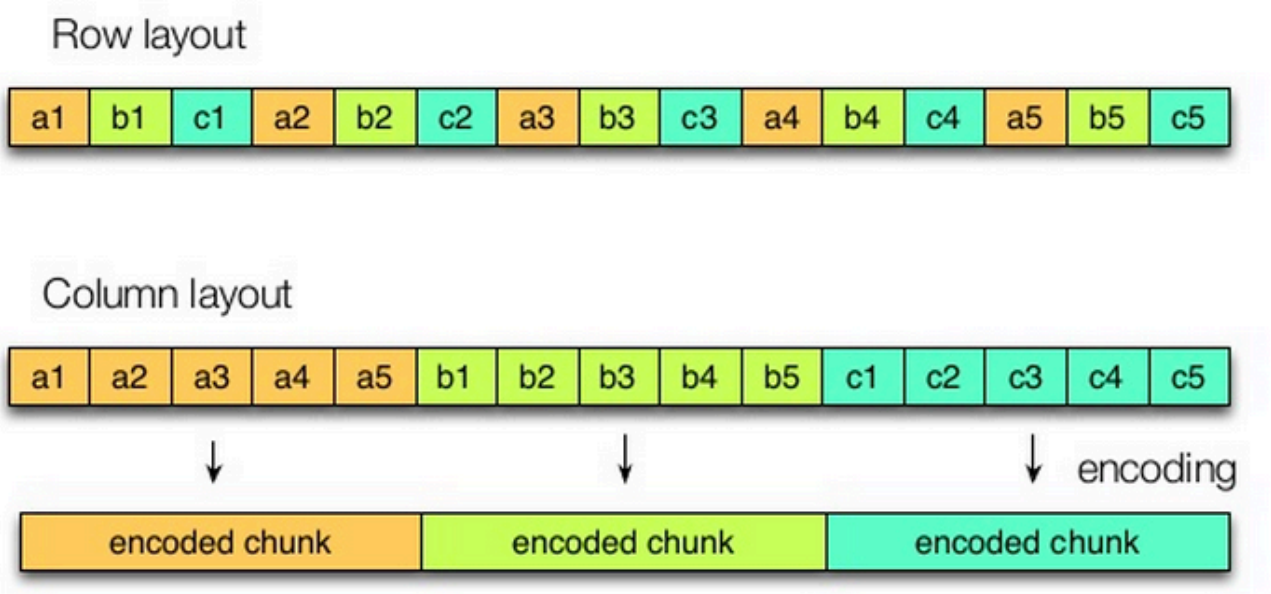
```
advanti@aslin108:~$ ls /user/hive/warehouse/holdings_total_marketvalue/
_common_metadata      part-r-00025.parquet  part-r-00051.parquet  part-r-00077.parquet
_metadata             part-r-00026.parquet  part-r-00052.parquet  part-r-00078.parquet
part-r-00001.parquet  part-r-00027.parquet  part-r-00053.parquet  part-r-00079.parquet
part-r-00002.parquet  part-r-00028.parquet  part-r-00054.parquet  part-r-00080.parquet
part-r-00003.parquet  part-r-00029.parquet  part-r-00055.parquet  part-r-00081.parquet
part-r-00004.parquet  part-r-00030.parquet  part-r-00056.parquet  part-r-00082.parquet
part-r-00005.parquet  part-r-00031.parquet  part-r-00057.parquet  part-r-00083.parquet
part-r-00006.parquet  part-r-00032.parquet  part-r-00058.parquet  part-r-00084.parquet
part-r-00007.parquet  part-r-00033.parquet  part-r-00059.parquet  part-r-00085.parquet
part-r-00008.parquet  part-r-00034.parquet  part-r-00060.parquet  part-r-00086.parquet
part-r-00009.parquet  part-r-00035.parquet  part-r-00061.parquet  part-r-00087.parquet
part-r-00010.parquet  part-r-00036.parquet  part-r-00062.parquet  part-r-00088.parquet
part-r-00011.parquet  part-r-00037.parquet  part-r-00063.parquet  part-r-00089.parquet
part-r-00012.parquet  part-r-00038.parquet  part-r-00064.parquet  part-r-00090.parquet
part-r-00013.parquet  part-r-00039.parquet  part-r-00065.parquet  part-r-00091.parquet
part-r-00014.parquet  part-r-00040.parquet  part-r-00066.parquet  part-r-00092.parquet
part-r-00015.parquet  part-r-00041.parquet  part-r-00067.parquet  part-r-00093.parquet
part-r-00016.parquet  part-r-00042.parquet  part-r-00068.parquet  part-r-00094.parquet
part-r-00017.parquet  part-r-00043.parquet  part-r-00069.parquet  part-r-00095.parquet
part-r-00018.parquet  part-r-00044.parquet  part-r-00070.parquet  part-r-00096.parquet
part-r-00019.parquet  part-r-00045.parquet  part-r-00071.parquet  part-r-00097.parquet
part-r-00020.parquet  part-r-00046.parquet  part-r-00072.parquet  part-r-00098.parquet
part-r-00021.parquet  part-r-00047.parquet  part-r-00073.parquet  part-r-00099.parquet
part-r-00022.parquet  part-r-00048.parquet  part-r-00074.parquet  part-r-00100.parquet
part-r-00023.parquet  part-r-00049.parquet  part-r-00075.parquet  part-r-00101.parquet
part-r-00024.parquet  part-r-00050.parquet  part-r-00076.parquet  part-r-00102.parquet
```

What is Columnar Data?

- Limit's IO to data needed
- Columnar compresses better
- Type specific encodings available
- Enables vectorized execution engines

Logical table representation

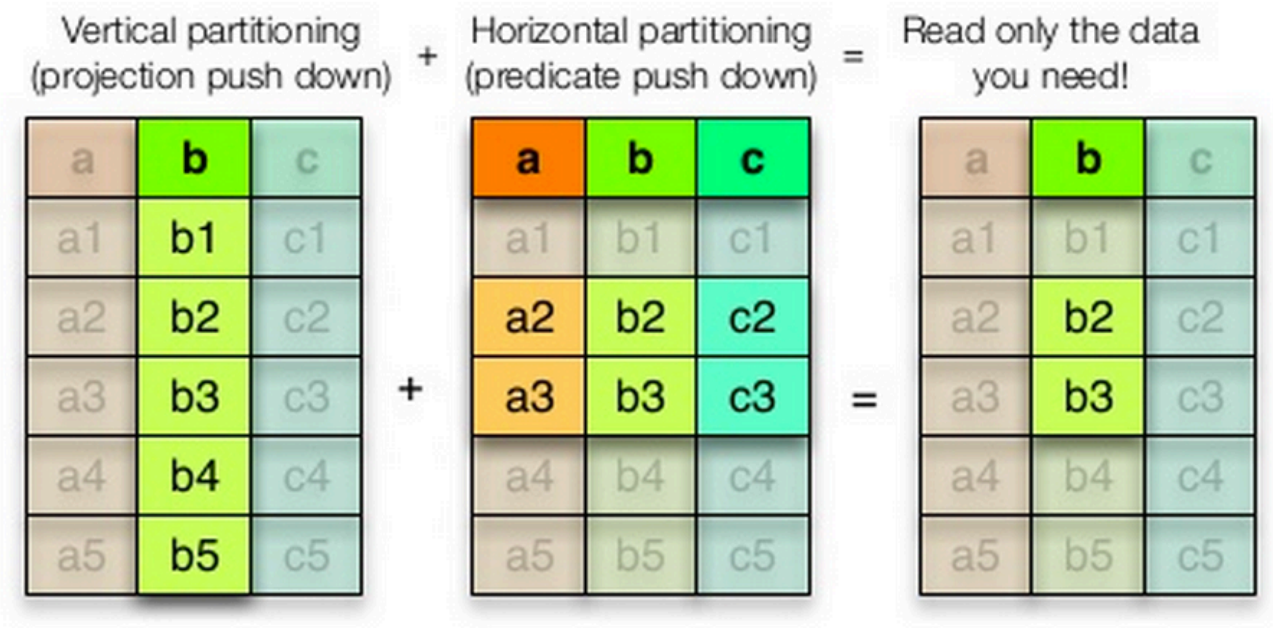
a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5



How is Columnar Data Read?

50 Columns Wide

```
CREATE TABLE IF NOT EXISTS HoldingLevelFlat_Parquet (
dataobjectinstanceversionkey bigint,
dataeffectivedt      timestamp,
investmentvehicle_id int,
iv_id                string,
shareclassbasics_id int,
scb_cusip            string,
scb_ticker           string,
scb_legaltype        string,
scb_name             string,
operation_id         int,
fundshareclass_id    int,
fsc_id               string,
portfoliolist_id     int,
portfolio_id         int,
hp_externalid        string,
currencyid           string,
portfoliosummary_id int,
p_date               timestamp,
holding_id           int,
holdingdetail_id     int,
hd_externalid        string,
hd_id                string,
cusip                string,
isin                 string,
sedol                string,
symbol               string,
region               int,
sector               string,
country              string,
currency             string,
currencycode         string,
securityname         string,
legaltype            string,
stylebox             int,
marketvalue          bigint,
costbasis            decimal,
maturitydate         timestamp,
accruedinterest      decimal,
coupon               decimal,
holdingytdreturn     decimal,
marketcapital        int,
localmarketvalue     string,
paymenttype          string,
rule144aeligible     string,
altmintaxeligible    string,
lessthanoneyearbond string,
firstboughtdate      timestamp,
weighting            decimal,
numberofshare        bigint,
sharechange          bigint
)
STORED AS PARQUET
;
```

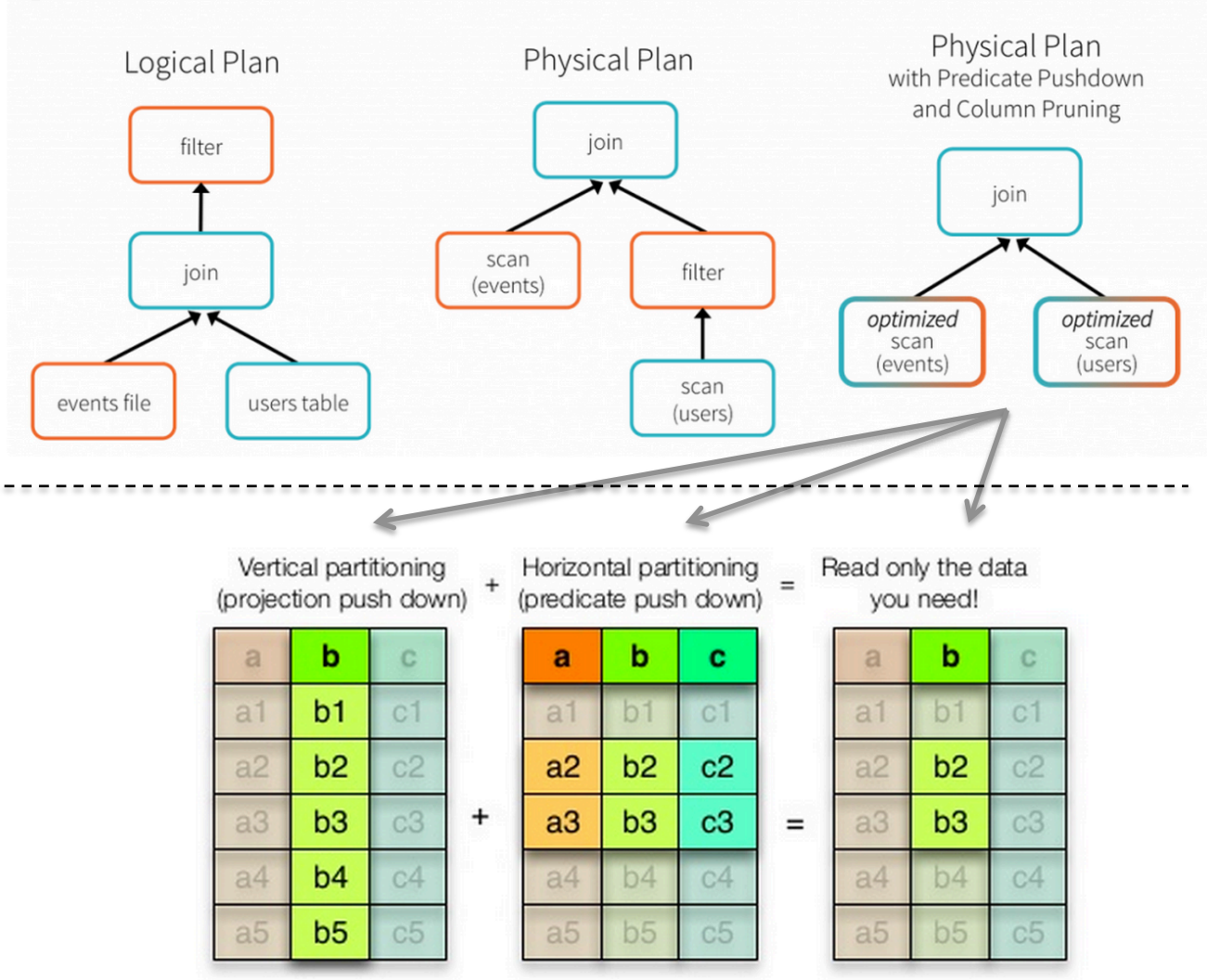


Projection → 9 columns

Predicate →

```
select
p_date,
iv_id, cusip, isin, sedol, legaltype,
weighting, numberofshare, marketvalue
from importdatafullholdingsholdinglevelflat_parquet
where
    marketvalue < 0
    and legaltype not in ('C', 'FO')
;
```

Spark DataFrame ➔ Parquet



Prepare CSV's in HIVE, persist in Parquet, show Spark SQL and DataFrame transforms using interactive shell in PySpark

DEMO: HIVE, PARQUET, SPARK SQL, DATAFRAME

Spark SQL is Experimental

Apache Spark

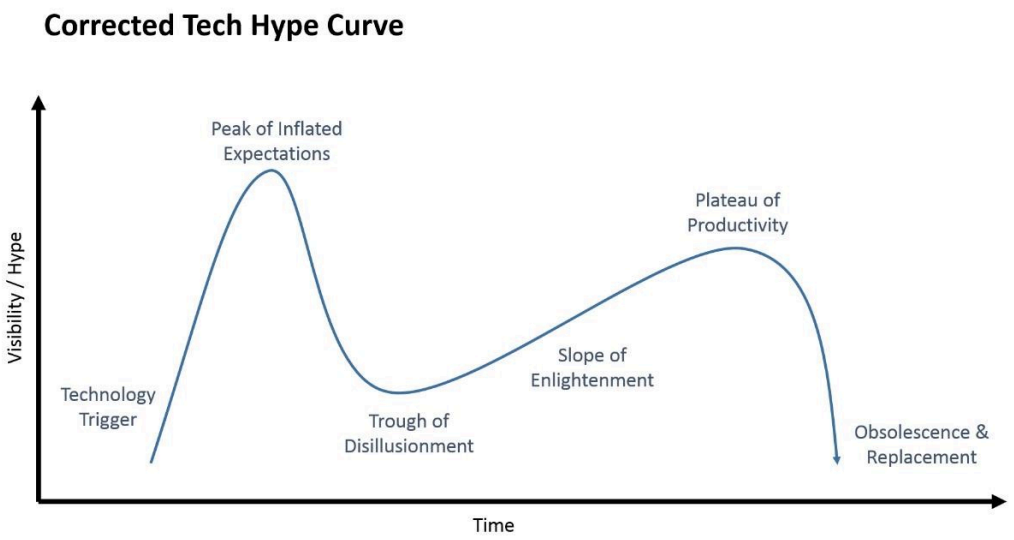
CDH 5.4.0 Spark is rebased on Apache Spark 1.3.0 and provides the following new capabilities:

- Spark Streaming WAL (write-ahead log) on HDFS, preventing any data loss on driver failure
- Spark external shuffle service
- Improvements in automatically setting CDH classpaths for Avro, Parquet, Flume, and Hive
- Improvements in the collection of task metrics
- Kafka connector for Spark Streaming to avoid the need for the HDFS WAL

The following is not yet supported in a production environment because of its immaturity:

- Spark SQL (which now includes dataframes)

See also **Apache Spark Known Issues** and **Apache Spark Incompatible Changes**.



Demo Outline

```
## DEMO OUTLINE: Hive, Parquet, Spark SQL, DataFrame
# 000 - Put Flat Files into HDFS and Tail
# 001 - Create HIVE transient tables: explain LOCATION in DDL
# 005 - Create HIVE view: explain purpose of VW
# 010 - Create HIVE parquet table: explain datatype differences
# 015 - Load into parquet table: use beeline to load data into parquet then select 4 columns
# 019 - PySpark data loading 3 ways: HIVE, CSV, Parquet
# 020 - PySpark data access examples: parquet from hive, and pull from cassandra
# 025 - PySpark data wrangling: split-apply-combine, scalar function
# 030 - PySpark save to parquet:
# 030 - PySpark save to Cassandra:
```

Expect workarounds and time spent hacking, but consider this, it's a learning opportunity…

OBSTACLES: NOTHING IS EASY

Nothing is easy – unless you're Ricky Bobby – "Shake and Bake"

Obstacles

Spark

- Don't expect Pandas-convenient Spark DataFrames (yet at least) (e.g. no upsampling, backfilling, etc)
- RDD's are powerful, but you'll need to get your hands dirty writing lower-level code (not a bad thing)
- Allocate enough memory on all of your nodes, it's a hog!

Parquet

- Date and binary support are pending although timestamp, decimal, char, and varchar are now supported in Hive 0.14.0
- You won't get Vertica or Cassandra level response times (maybe in the future – that's my speculation)

Getting Started

- Learn by doing: reading is useful to gain the basics but just "jump right in and do it"

Douglas Eisenstein
@dougeisenstein
doug.eisenstein@advantisolutions.com

Helping to create fast, reliable, and transparent modern data pipelines for financial analytics

Talk feedback: https://goo.gl/Y0UuWy

QUESTIONS

Using Spark, Python, and Cassandra for Loading and Transformations at Scale

APPENDIX

Resources

http://training.databricks.com/workshop/itas_workshop.pdf (Intro to Spark PDF)
https://databricks-training.s3.amazonaws.com/slides/SparkSQLTraining.Summit.July2014.pdf (Spark SQL Training Module)
http://training.databricks.com/workshop/itas_workshop.pdf (Spark 101)
http://spark-summit.org/2014/training (General Spark Videos)
https://spark.apache.org/docs/1.3.0/sql-programming-guide.html (Spark SQL and DataFrame's — awesome)
http://www.slideshare.net/EvanChan2/2014-07olapcassspark (OLAP with Cassandra and Spark)
https://databricks.com/blog/2015/03/24/spark-sql-graduates-from-alpha-in-spark-1-3.html (Latest Spark/DataFrame/Parquet)
https://spark.apache.org/docs/latest/programming-guide.html (Basics of Spark Development)
https://spark.apache.org/docs/latest/configuration.html (Spark Configuration)
https://academy.datastax.com/demos/datastax-enterprise-joining-tables-apache-spark (Joining Cassandra Tables in Spark)
http://www.infoobjects.com/author/rishi/ (Spark / Parquet Integration)
https://parquet.incubator.apache.org/presentations/ (Parquet Videos/Slides, awesome)
http://www.slideshare.net/databricks/spark-sqlsse2015public (All about DataFrame by the author)
http://www.infoobjects.com/category/spark_cookbook/ (Good walkthrough of Spark Demos)
http://tobert.github.io/post/2014-07-15-installing-cassandra-spark-stack.html (Spark / Cassandra Integration from scratch)
http://blog.cloudera.com/blog/2015/05/working-with-apache-spark-or-how-i-learned-to-stop-worrying-and-love-the-shuffle/ (Spark by Data Engineer)
http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-1/ (Tuning Spark)
https://spark-summit.org/2015-east/wp-content/uploads/2015/03/SSE15-21-Sandy-Ryza.pdf ()
https://www.youtube.com/watch?v=0OM68k3np0E&list=PL-x35fyliRwiiYSXHyI61RXdHlYR3QjZ1&index=5 ()
http://www.slideshare.net/dataera/parquet-format

About Me

- My vision is to make data preparation fast and reliable
- I help financial firms with data-intensive processes
- 

- In my spare time: CrossFit, Baseball, No Gardening!

 @dougeisenstein