A REFERENCE ARCHITECTURE FOR INTERNET OF THINGS

SUJEE MANIYAM

FOUNDER / PRINCIPAL @ ELEPHANTSCALE

SUJEE@ELEPHANTSCALE.COM

HI, I'M SUJEE MANIYAM

- **Founder / Principal @ ElephantScale**
  - Consulting & Training in Big Data
  - Training in Spark / Hadoop / NoSQL / Data Science

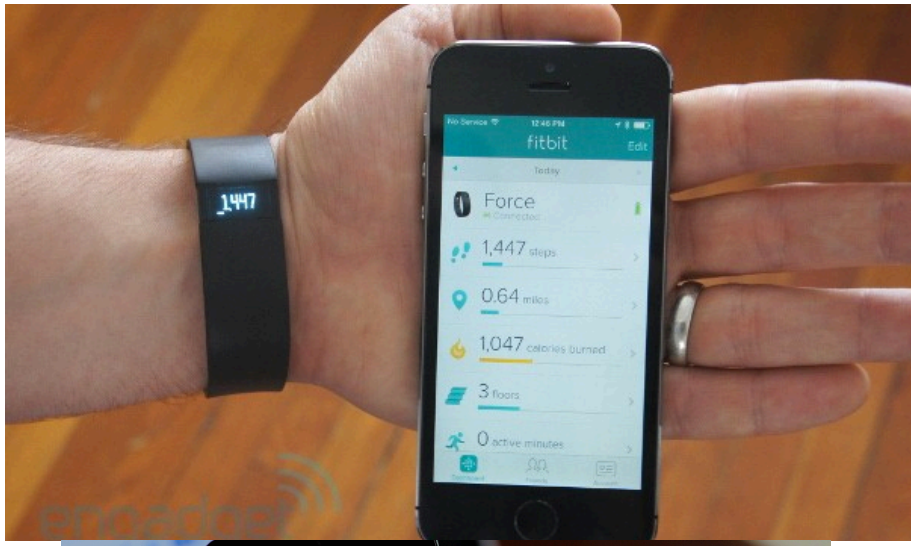
- **Author**
  - "Hadoop illuminated" open source book
  - "HBase Design Patterns

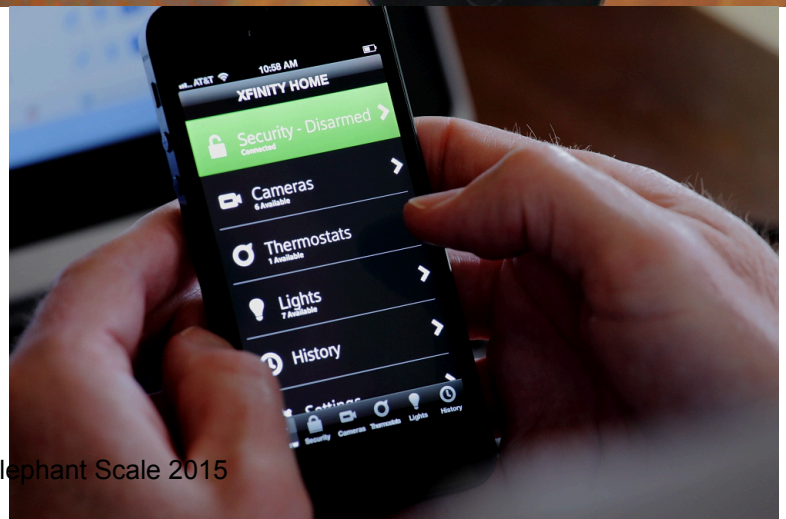
- **Open Source contributor: github.com/sujee**

- **sujee@elephantscale.com**
- **http://sujee.net**

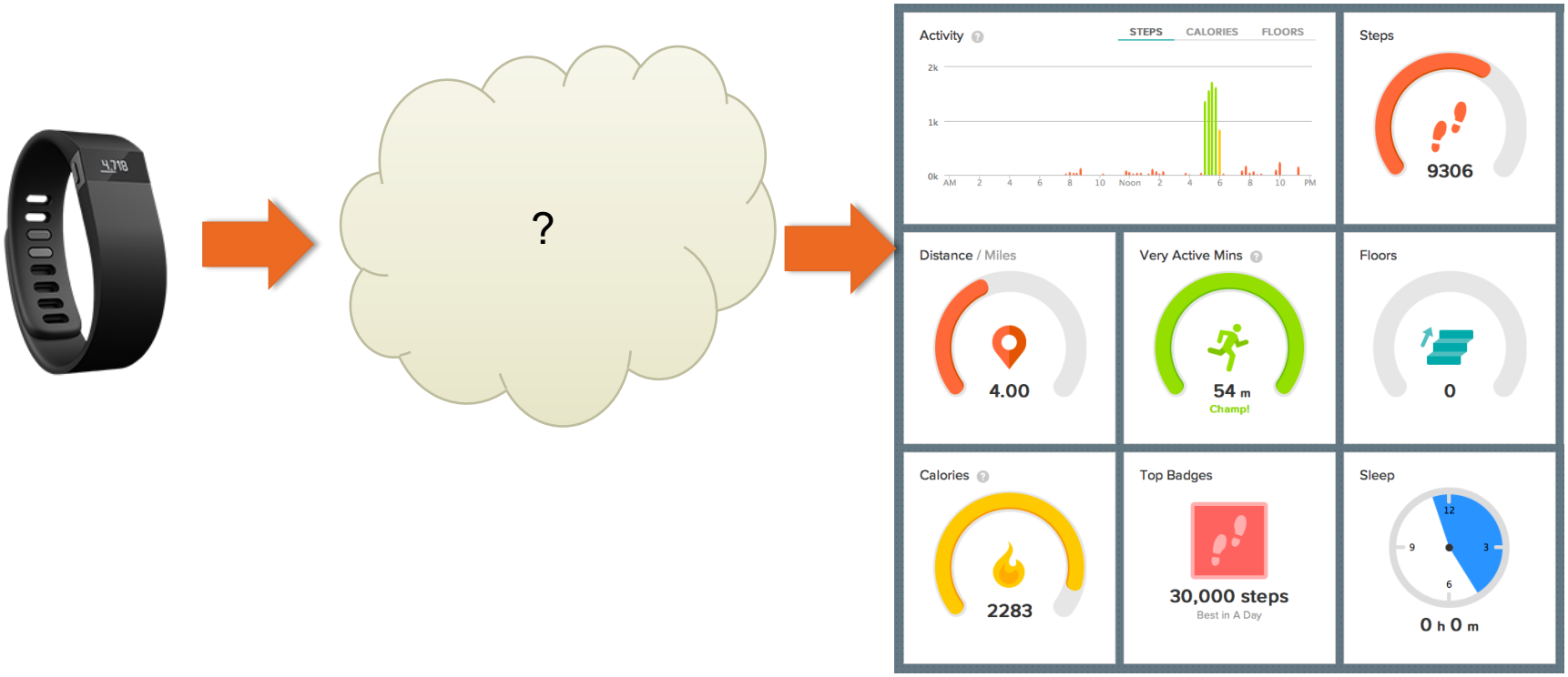
Spark Training available!

INTERNET OF THINGS – A REALITY





DATA INFRASTRUCTURE



DATA VOLUME ?
A NAPKIN CALCULATION

Say we have

- **Million sensors**
- **Each sensor reports every minute**
- **data size 1KB**

This will result in :

- **1.44 Billions events / day !**
- **1.44 TB / day !!**

SENSOR DATA WORKSHEET

IoT Temperature Sensor Projection

variables	description				
sensors	1M	1.00E+06	1	million	
signal frequency	every min / 60 sec	60	secs		
event size	1KB	1000	bytes		
events per day per sensor		1440			
total events per day		1.44E+09	1440	million	1.44 billion
total events / sec		1.67E+04			16,666.67
total data size per day		1.44E+12	1440	GB	1.44 TB

SENSOR DATA : TEXAS UTILITIES SMART METER DATA

Texas Smart Meter Projections

variables	description				
sensors	10 million customers	1.00E+07	10	million	
signal frequency	every 15 mins	900	secs		
event size	1.4 K	1400	bytes		
events per day per sensor		96			
total events per day		9.60E+08	960	millions	0.96 billion
total events / sec		1.11E+04			11,111.11
total data size per day		1.34E+12	1344	GB	1.344 TB

BIG ‘DATA’

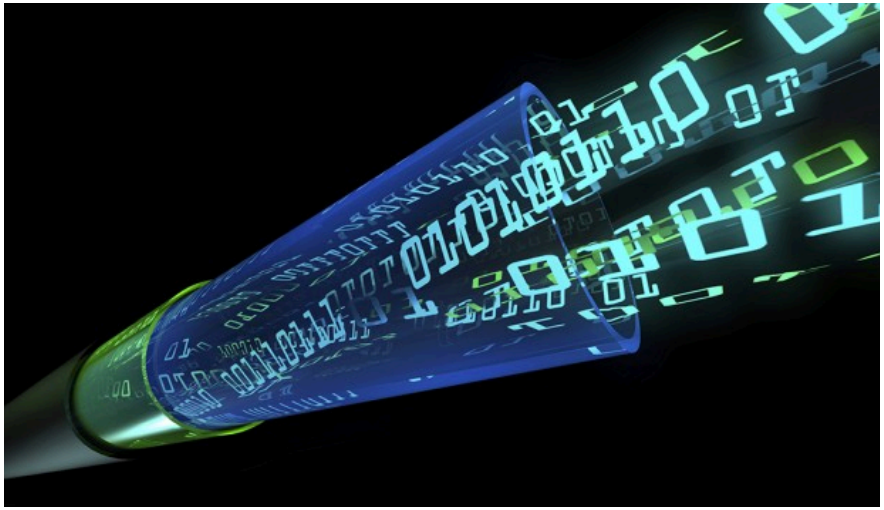
DATA VELOCITY

Say we have

- **Million sensors**
- **Each sensor reports every minute**
- **data size 1KB**

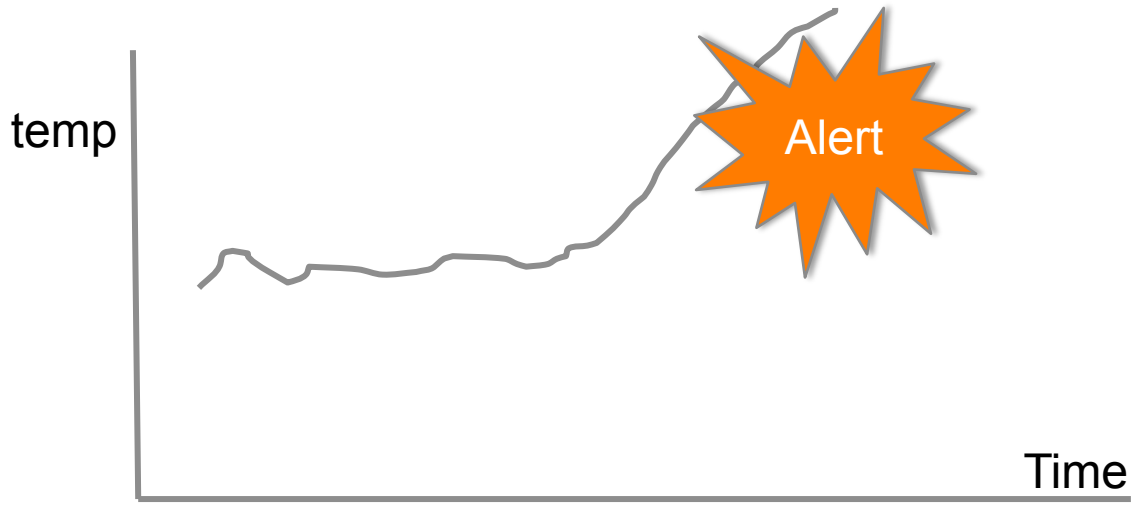
➔

- **Millions events / minute**
- **~ 17,000 events / sec**



DATA PROCESSING SPEED

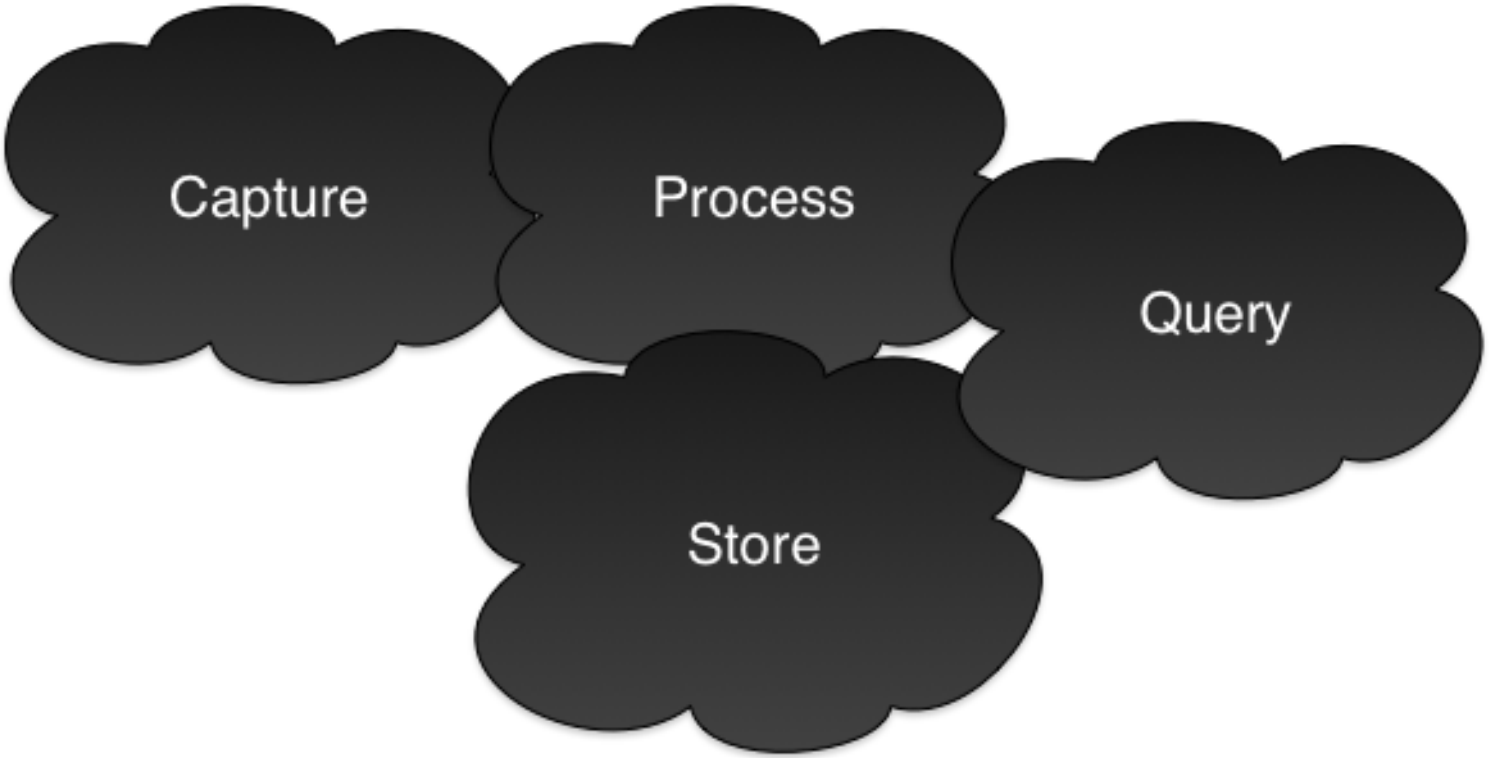
- **Need (near) real time processing most of the time**
  - E.g. Need to alert if temperature suddenly spikes



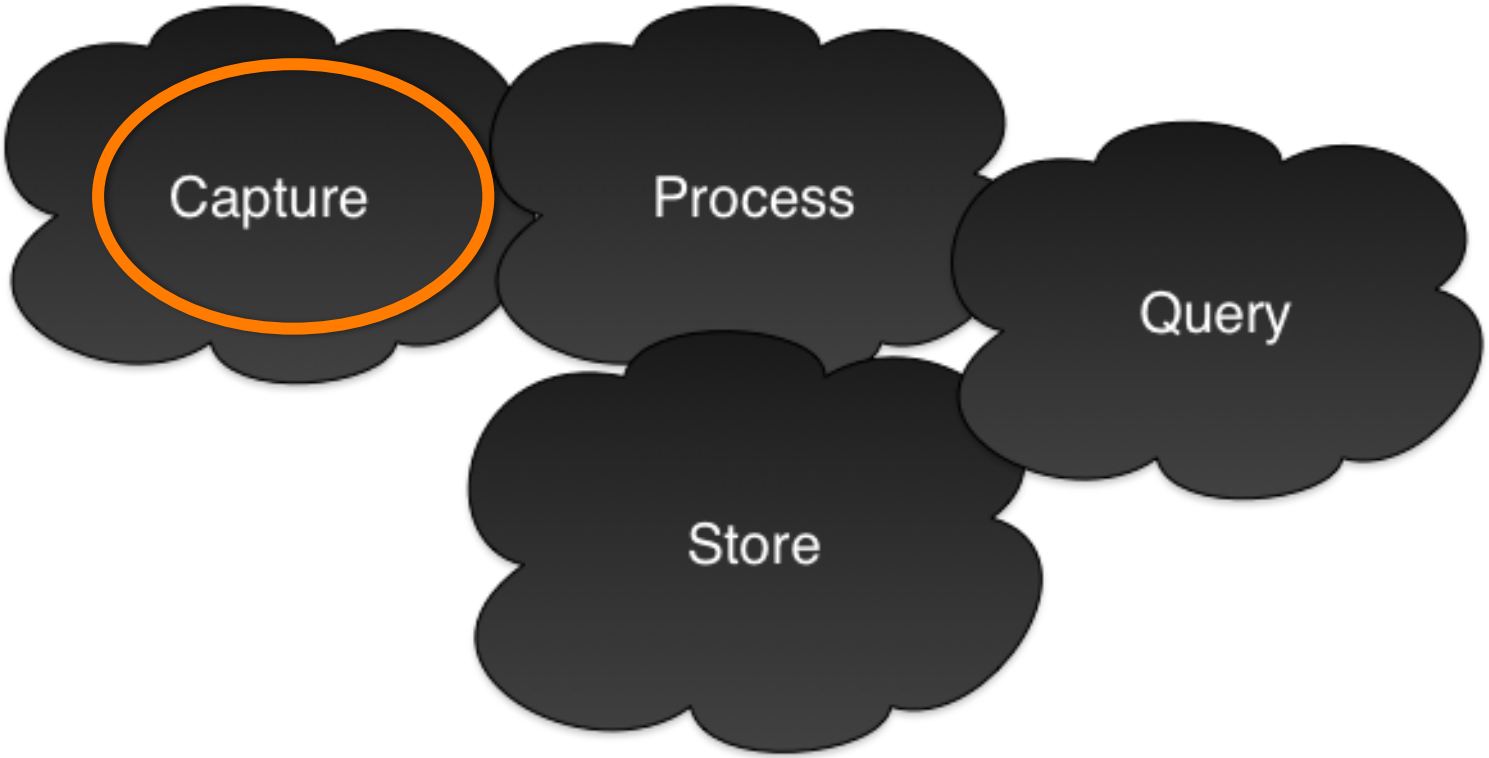
CHALLENGE = BIG DATA + REAL TIME

- **Don't loose events !**
  - Any event could be important
  - Most events are mundane (e.g. temperature stays between 68'F – 72' F)
- **Process them in near real time**
- **Store the events for a long time**
  - Audit
  - Diagnose
- **Support various queries**
  - Real time (what is the latest temperature for sensor id 123?)
  - Aggregate (what is the avg. temp in zipcode 12345)

HIGH LEVEL ARCHITECTURE



NEXT : (1) CAPTURE



(1) CAPTURE REQUIREMENTS

Requirements:

- **Capture events coming at high speed**
  - Tens of thousands events / sec (some times millions / sec)
- **Don’t loose events**
  - Tolerate hardware / software failure
  - Tolerate intermittent connectivity issues
- **Scale ‘easily’**

(1) CAPTURE CHOICES

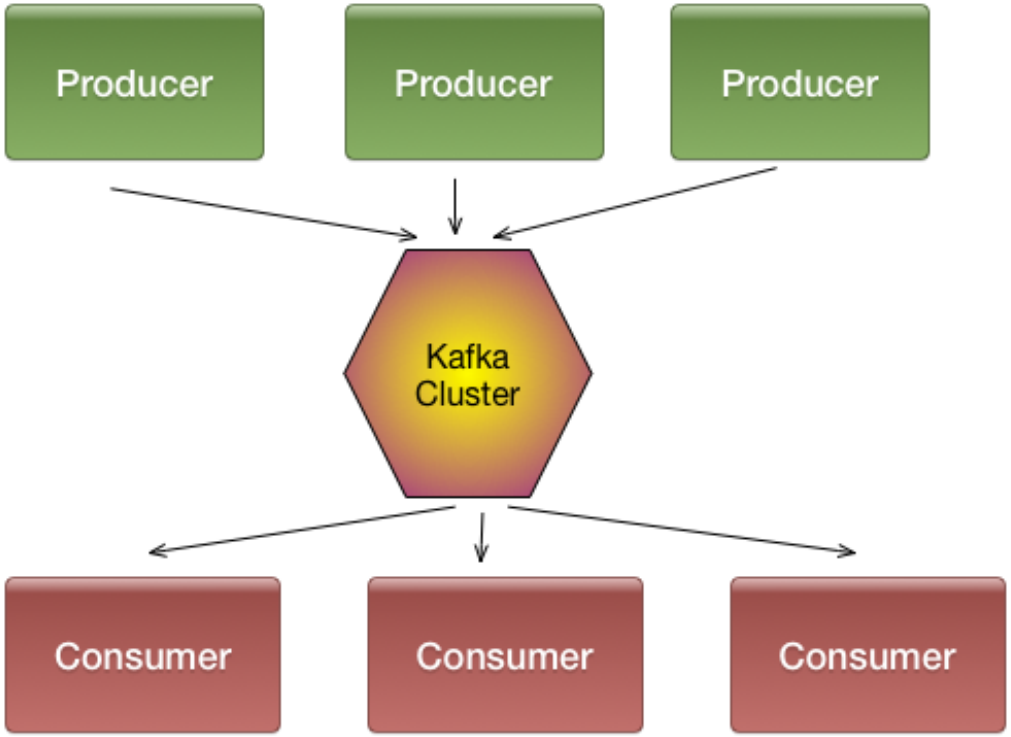
- **MQ (RabbitMQ ..etc)**
  - Good adoption in enterprises / durable
- **FluentD**
  - Data collector for various sources
- **Flume**
  - Part of Hadoop eco system
  - Good for collecting logs from many sources
- **AWS Kinesis**
  - Queue system in Amazon Cloud
- **Kafka**
  - Distributed queue

(1) CAPTURE
MEET KAFKA

- **Apache Kafka is a distributed messaging system**
- **Came out of LinkedIn… open sourced in 2011**
- **Built to tolerate hardware / software / network failures**
- **Built for high throughput and scale**
  - LinkedIn : 220 Billion messages / day
  - At peak : 3+ million messages / day

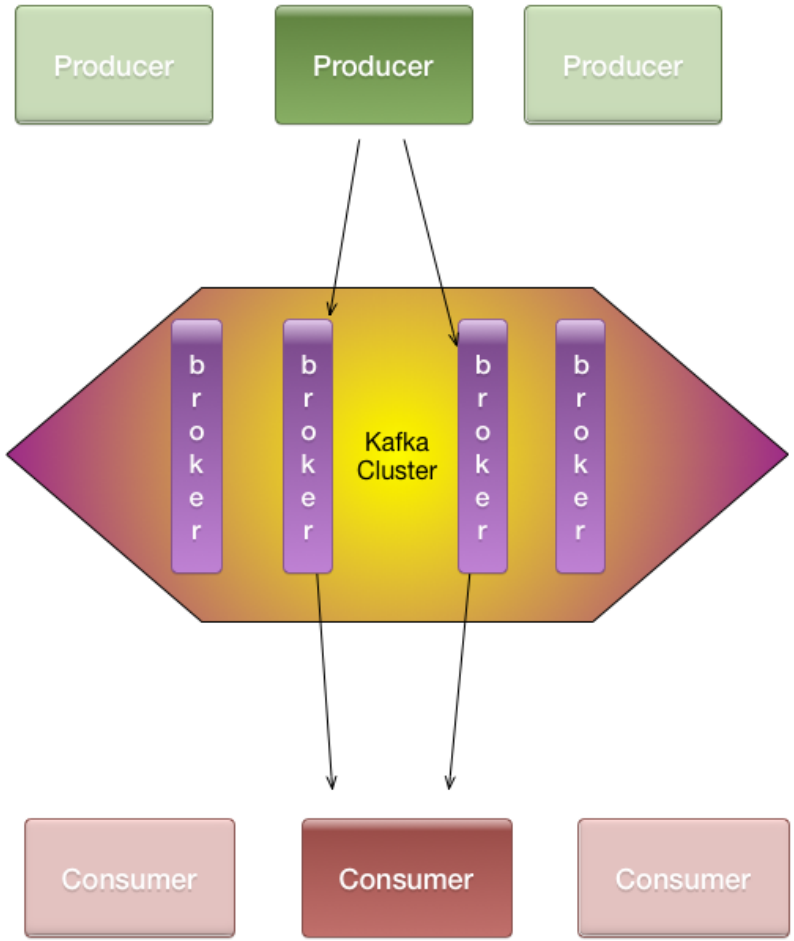
(1) CAPTURE
KAFKA ARCHITECTURE

- **Publisher - subscriber / producer – consumer model**



(1) CAPTURE
KAFKA ARCHITECTURE

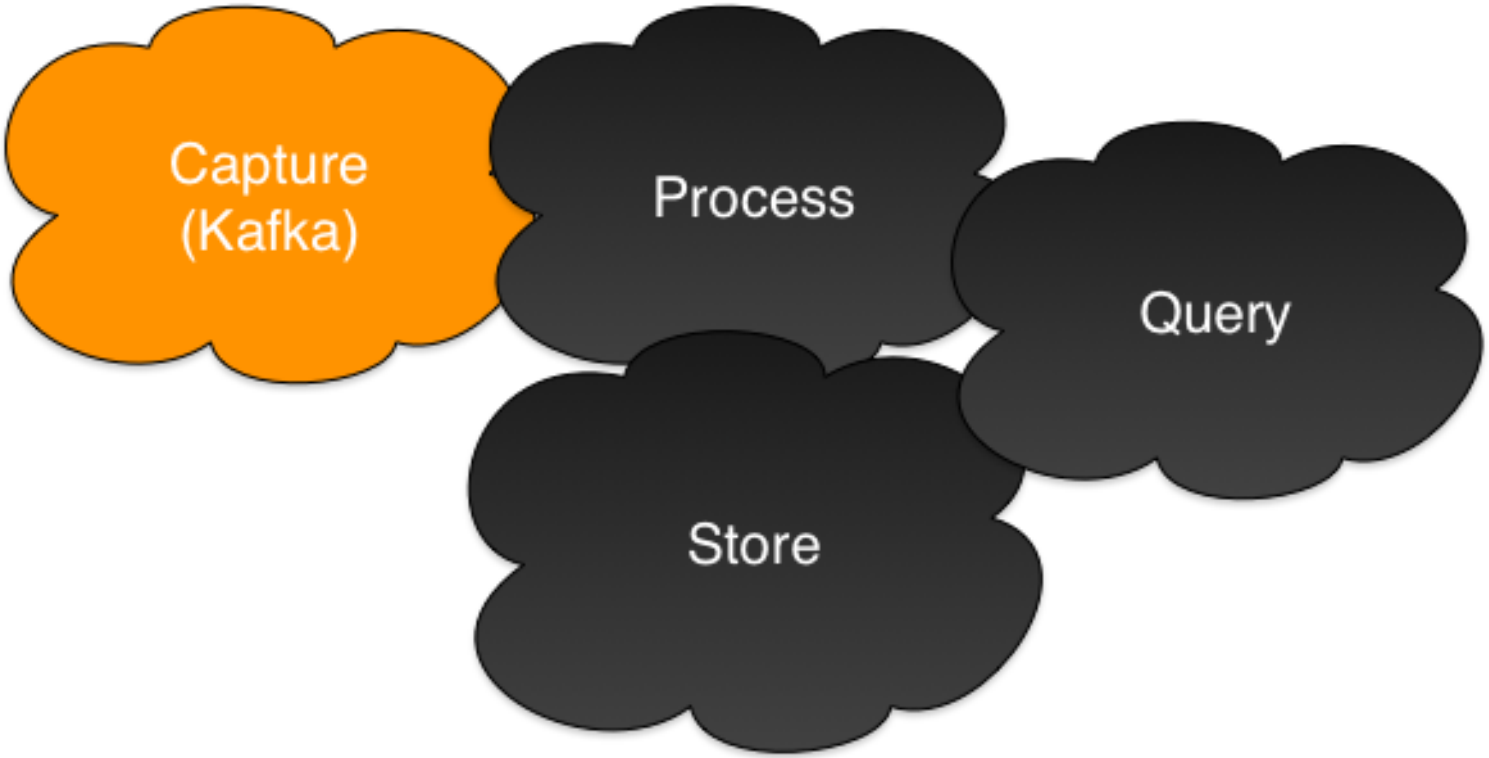
- **Producers write data to brokers**
- **Consumers read data from brokers**
- **All of this is distributed / parallel**
- **Failure tolerant**
- **Data is stored as topics**
  - “sensor_data”
  - “alerts”
  - “emails”



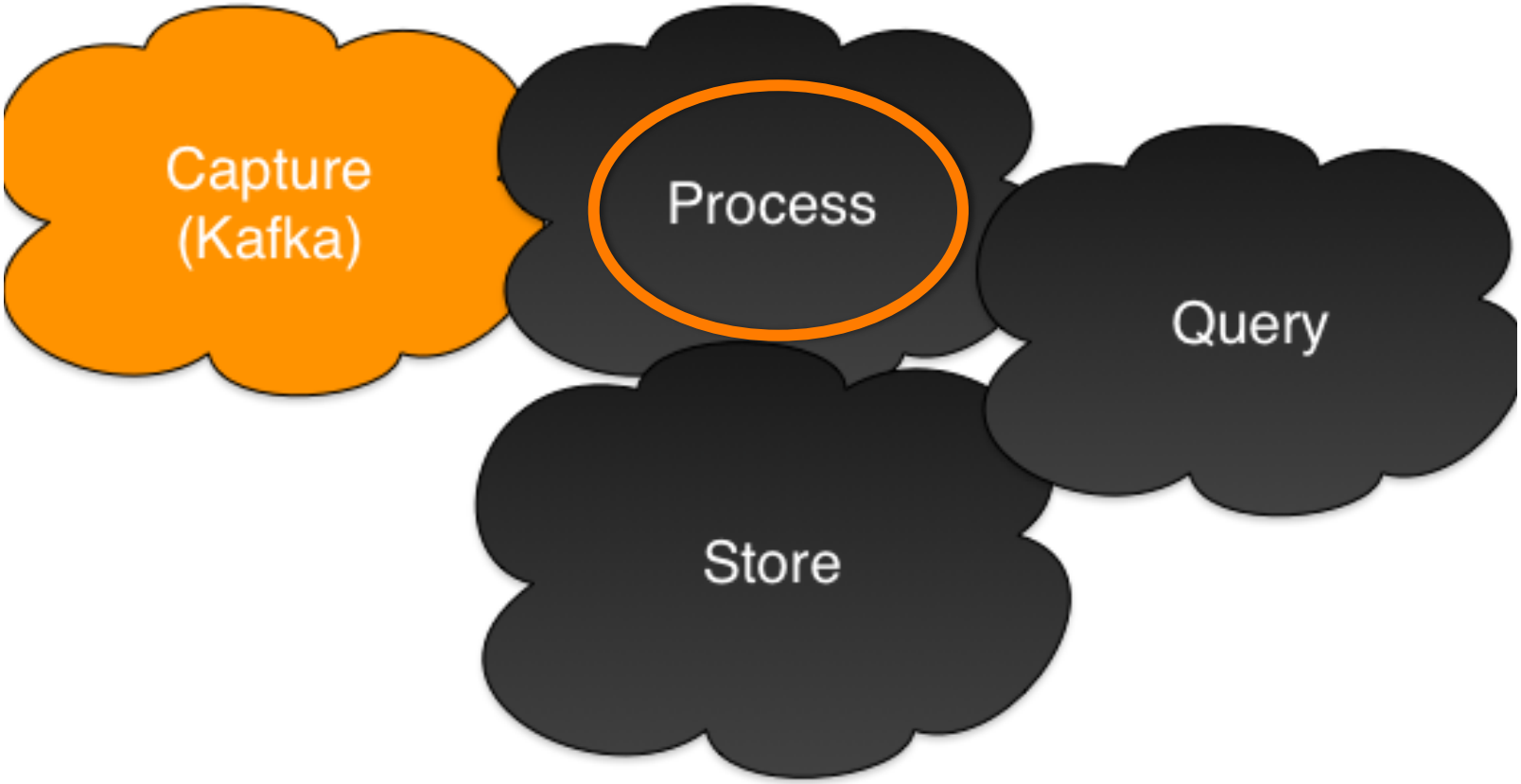
(1) CAPTURE KAFKA USERS

- **Linked In**
  - Track user activities
  - Sending emails
- **Netflix**
  - Real time monitoring
- **Spotify**
  - Ship logs to hadoop
- **Uber… AirBnB….**

(1) CAPTURE
ARCHITECTURE WITH KAFKA



NEXT : (2) PROCESSING



(2) PROCESSING REQUIREMENTS

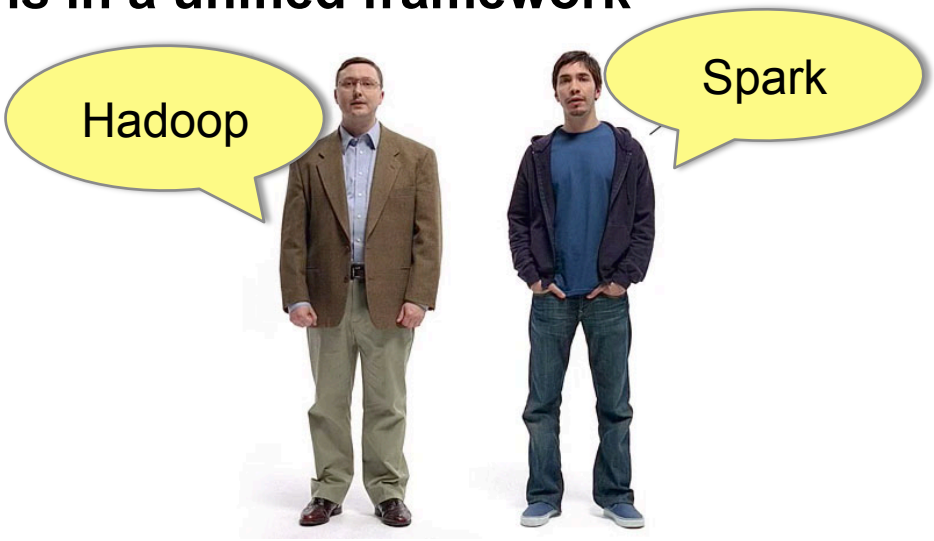
- **Process events in real time or near real time**
- **High velocity**
  - Tens of thousands → millions of events / sec
- **Guaranteed processing**
  - Process an event at-least-once
  - Exactly-once (harder to achieve)
- **Failure tolerant**
- **Scale 'easily'**

(2) PROCESSING CHOICES

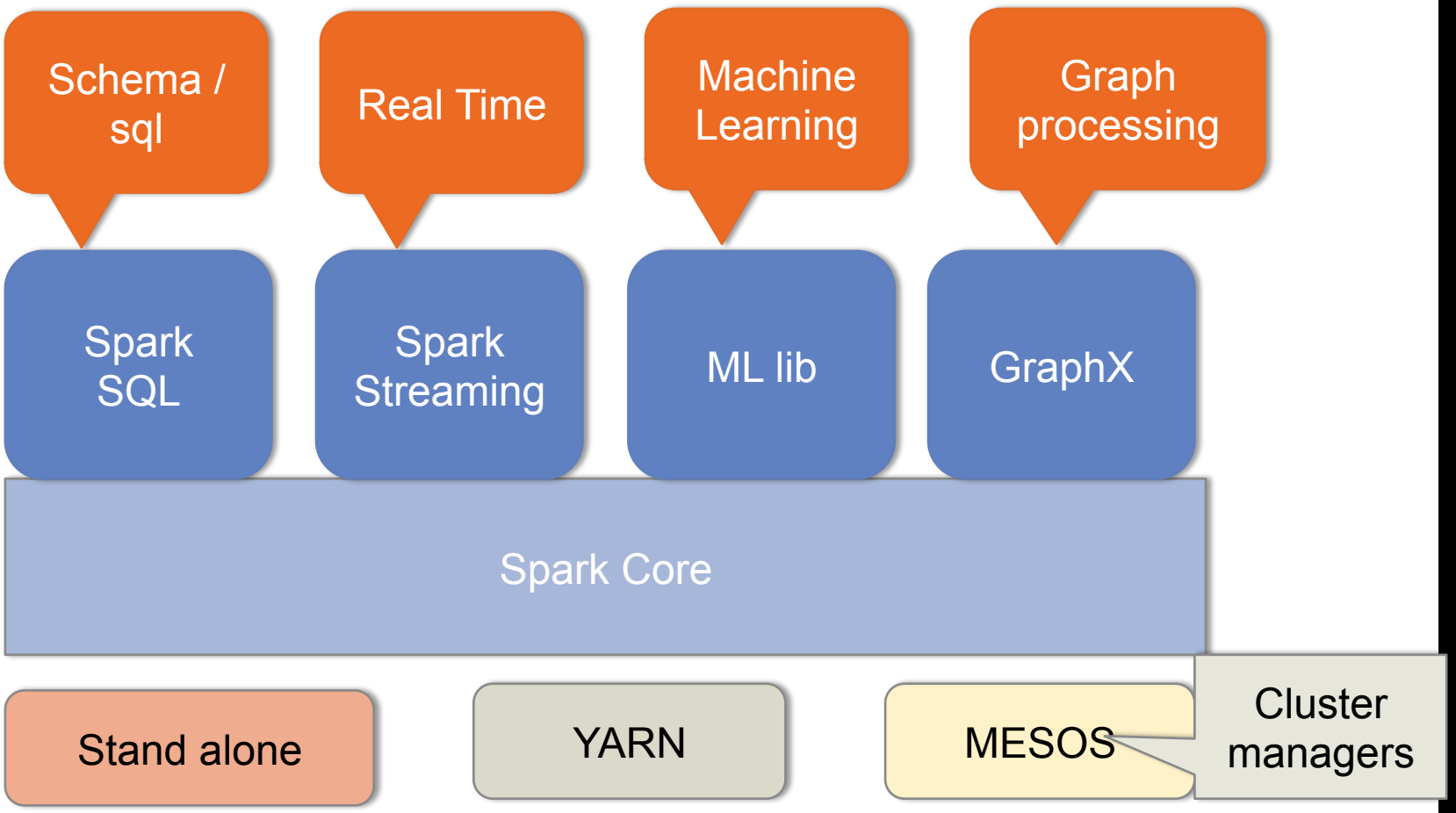
- **Storm**
  - Process streams
  - Events based
  - Came out of twitter
- **Apache Samza**
  - Stream processing framework based on Kafka + Hadoop YARN
- **Apache NiFi**
  - Data flow
  - New project / incubating
- **Spark streaming**

(2) PROCESSING
MEET SPARK

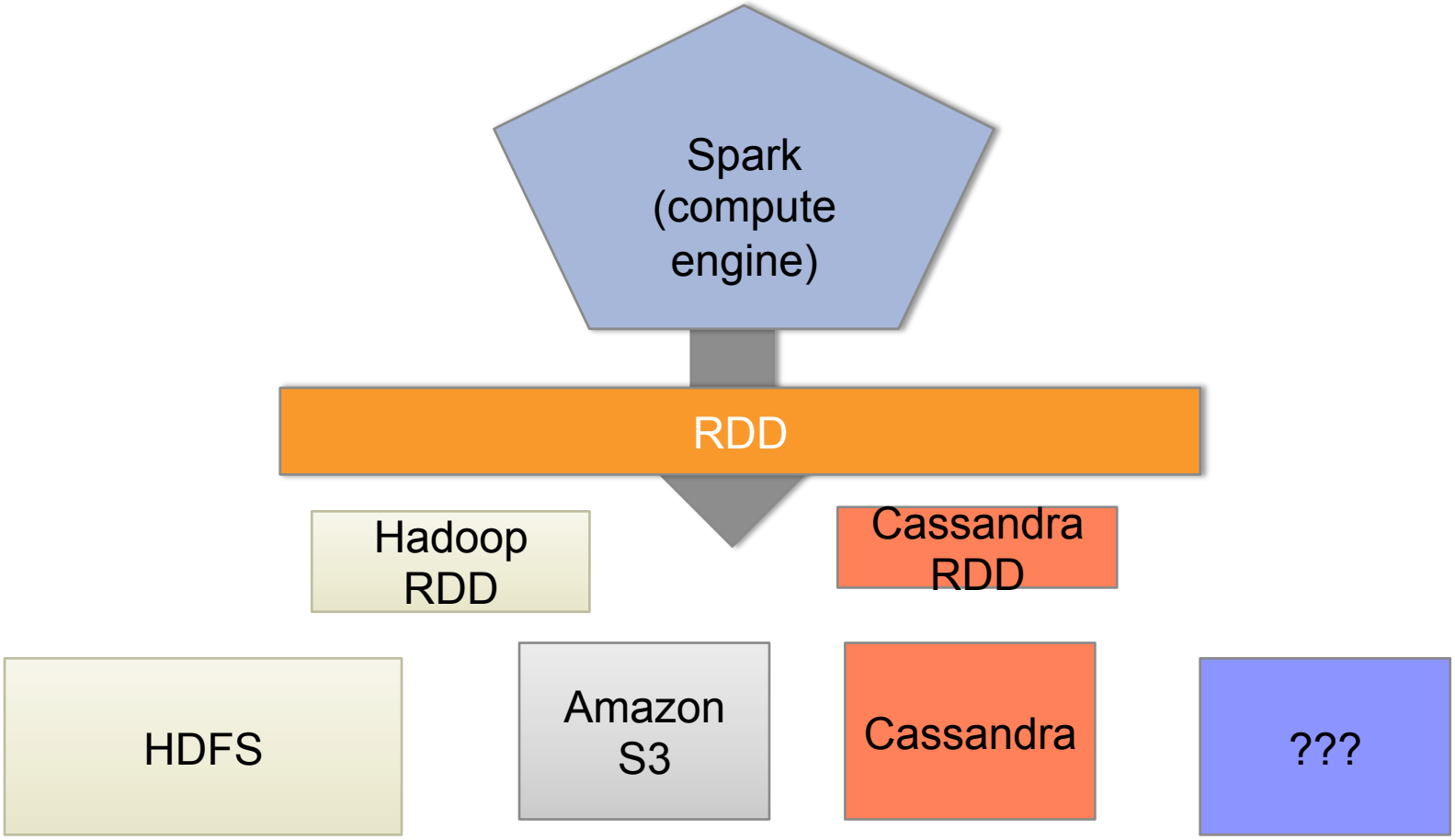
- **Spark is the new darling of 'Big Data' world**
  - Lot's of activity and interest
- **Fast and Expressive Cluster Compute Engine**
- **"First Big Data platform to integrate batch, streaming and interactive computations in a unified framework" – stratio.com**



(2) PROCESSING
SPARK ECO-SYSTEM



(2) PROCESSING
SPARK DATA SOURCE ABSTRACTION



(2) PROCESSING
SPARK : 'UNIFIED' STACK

Spark supports multiple programming models

- Map reduce style batch processing
- Streaming / real time processing
- Querying via SQL
- Machine learning

All modules are tightly integrated

- Facilitates rich applications

Spark can be only stack you need !

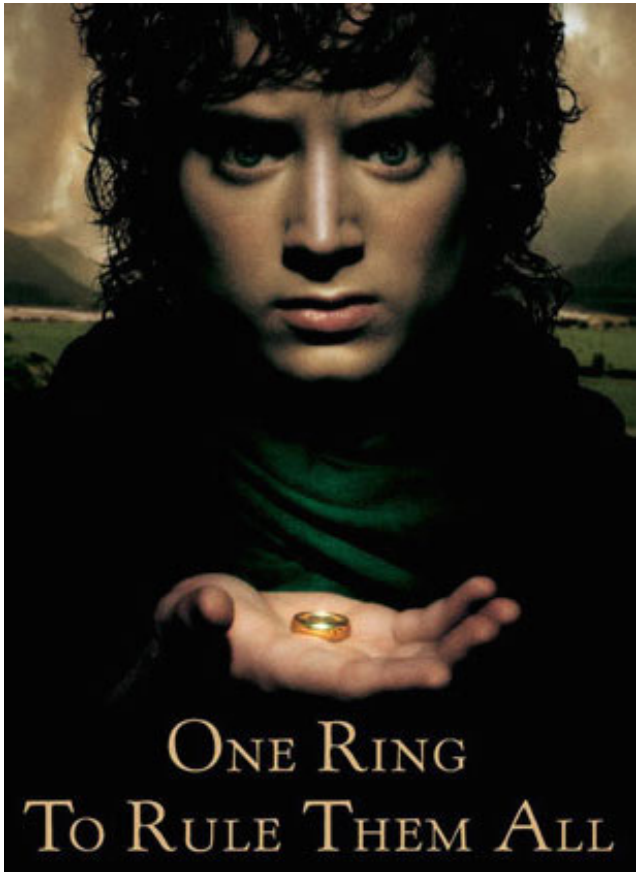
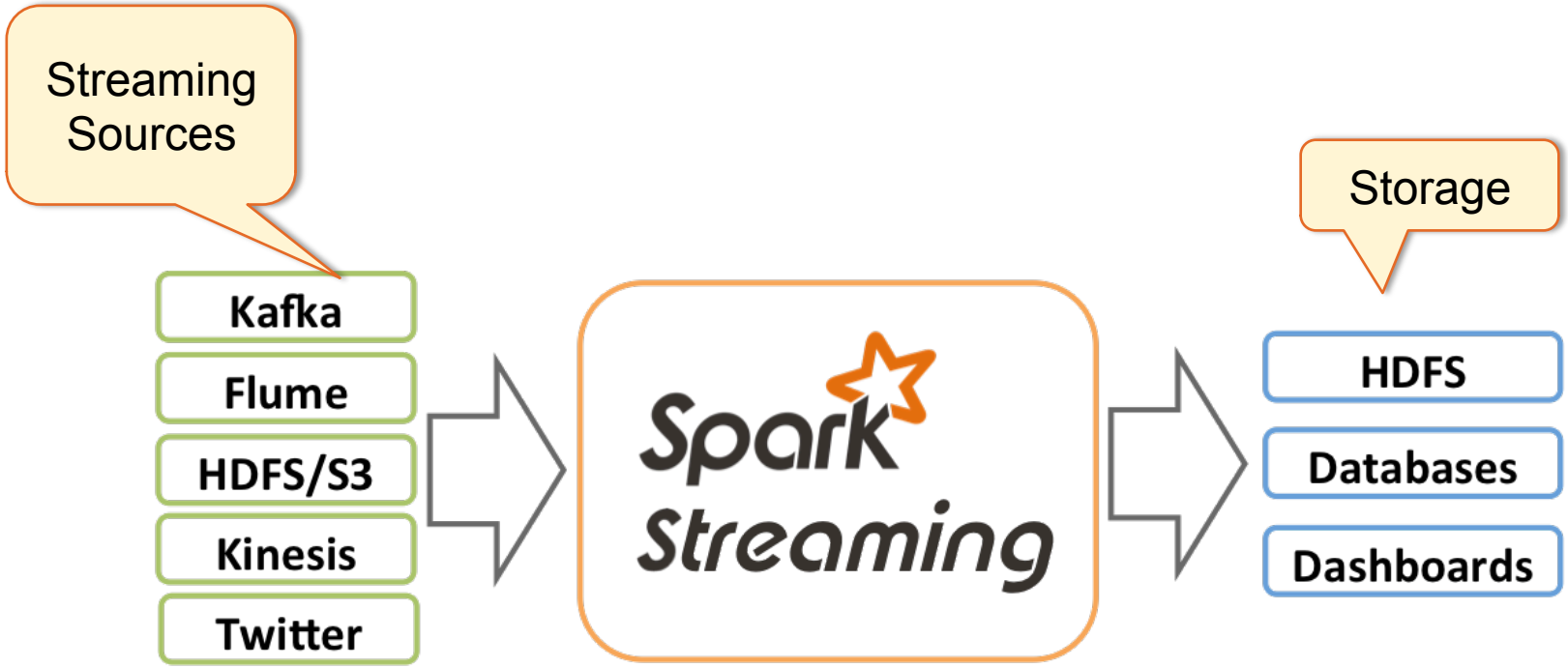


Image: buymeposters.com

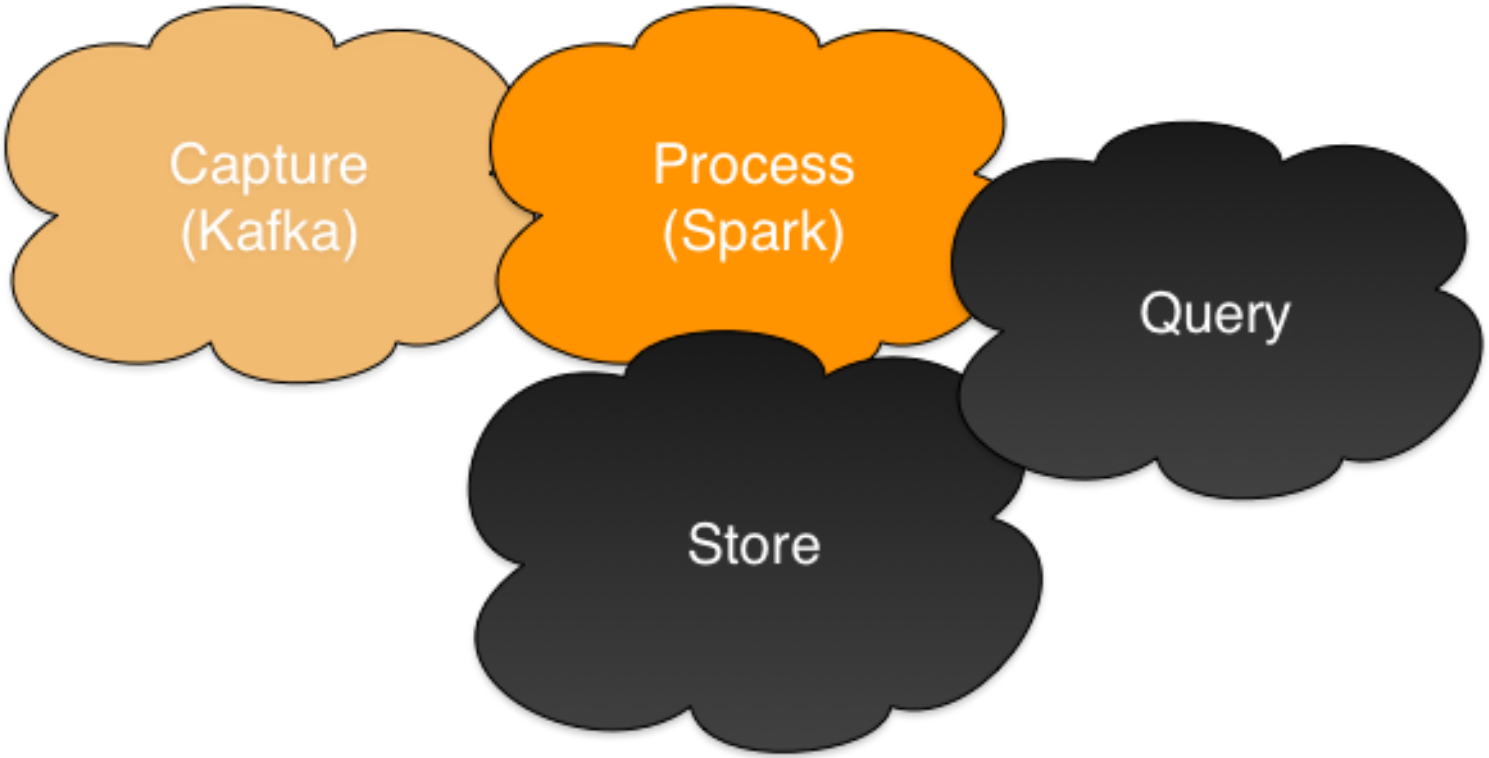
(2) PROCESSING
SPARK STREAMING



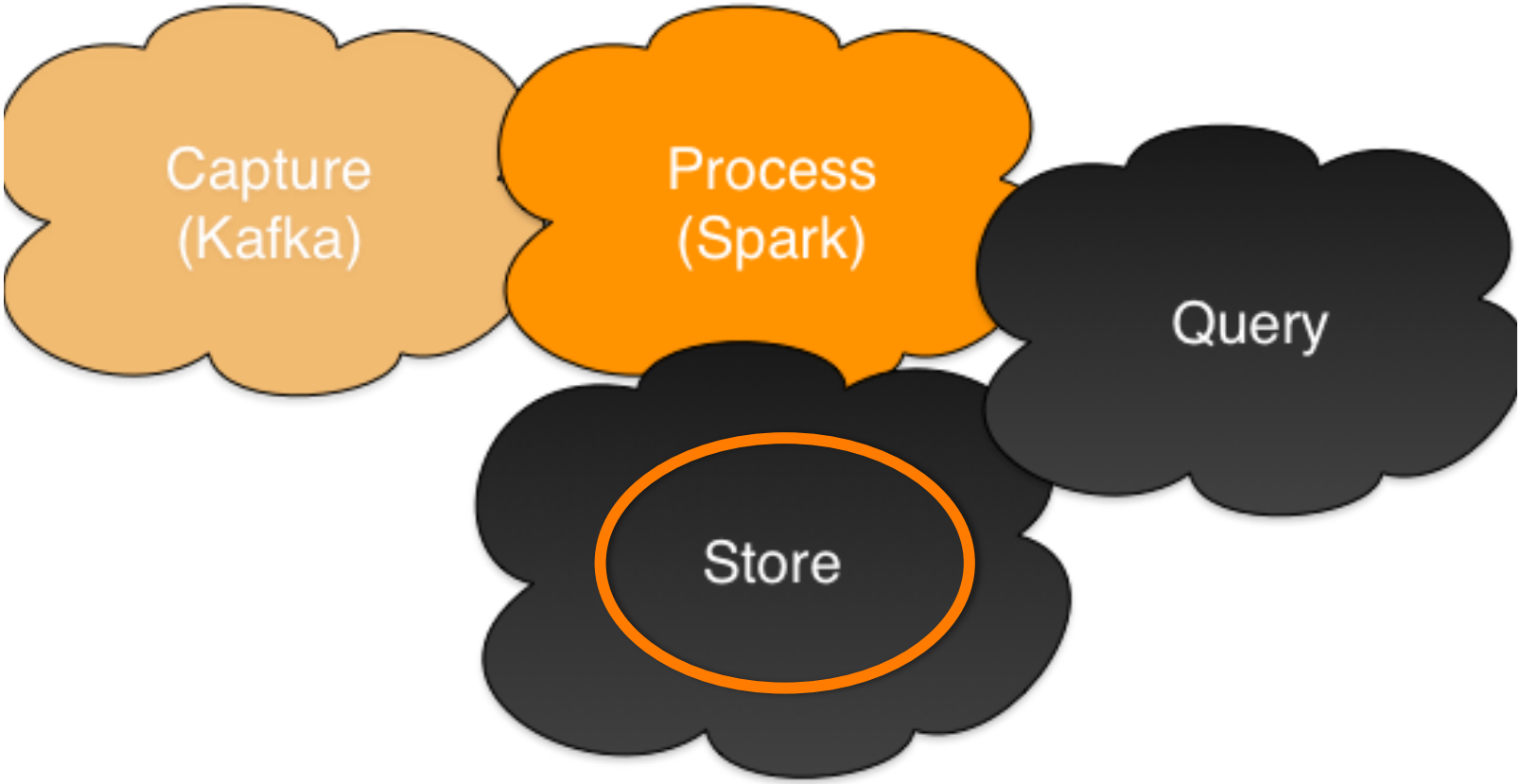
(2) PROCESSING
SPARK STREAMING

- **Provides 'high level' operations in time windows**
  - E.g. 'calculate X for the past 10 seconds'
- **Good adoption**

(2) PROCESSING
ARCHITECTURE WITH SPARK STREAMING



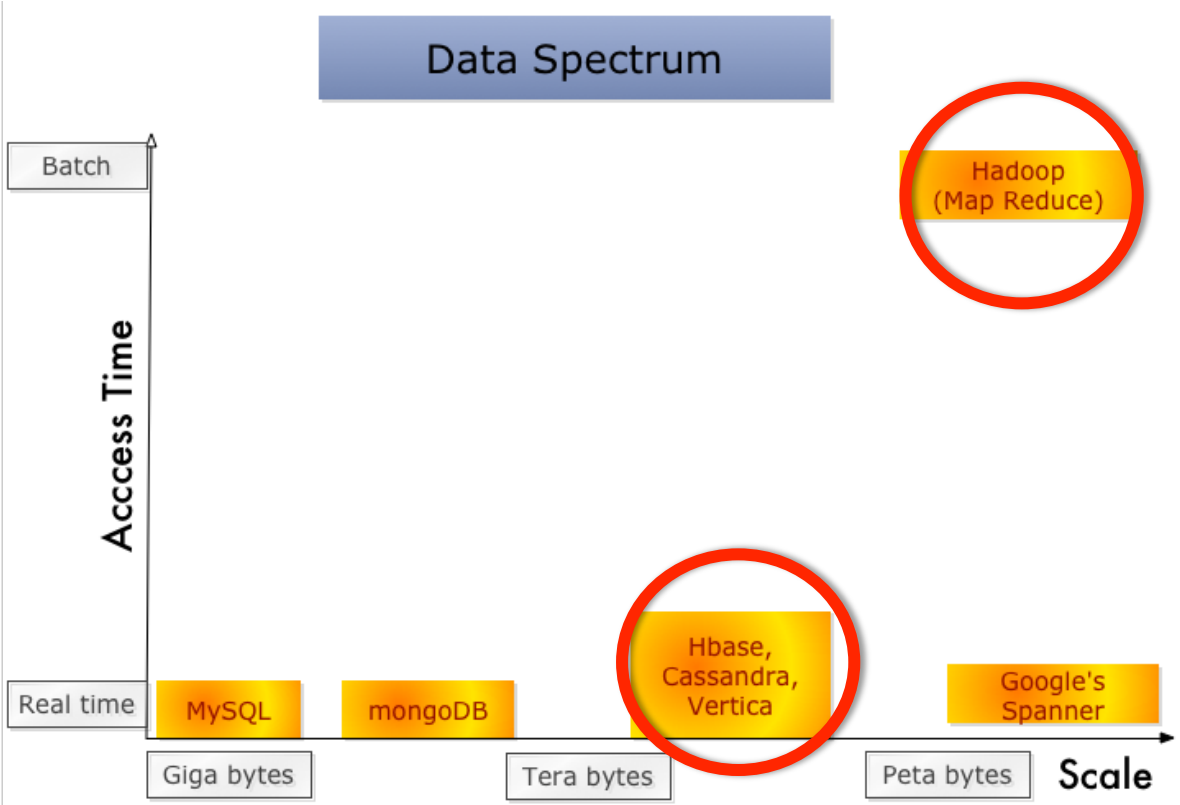
NEXT : STORAGE



(3) STORAGE REQUIREMENTS

- **Handle 'Big Data' (1 TB / day !)**
- **Traditional storages are not effective (or too expensive)**
- **Need two types of storage**
  1. 'forever' storage
     - Store multi terabytes of data for a long periods
     - Support Batch queries
  2. 'fast / real-time lookup' storage
     - Query in real time (milliseconds)
       "what is the latest reading for sensor-123 ?"
     - Store latest / new data (e.g. last 3 months)
     - Flexible schema for semi-structured data
- **Both need to scale**

(3) STORAGE REQUIREMENTS



adapted from : http://www.slideshare.net/medriscoll/driscoll-strata-buildingdatastartups25may2011clean

(3) STORAGE CHOICES

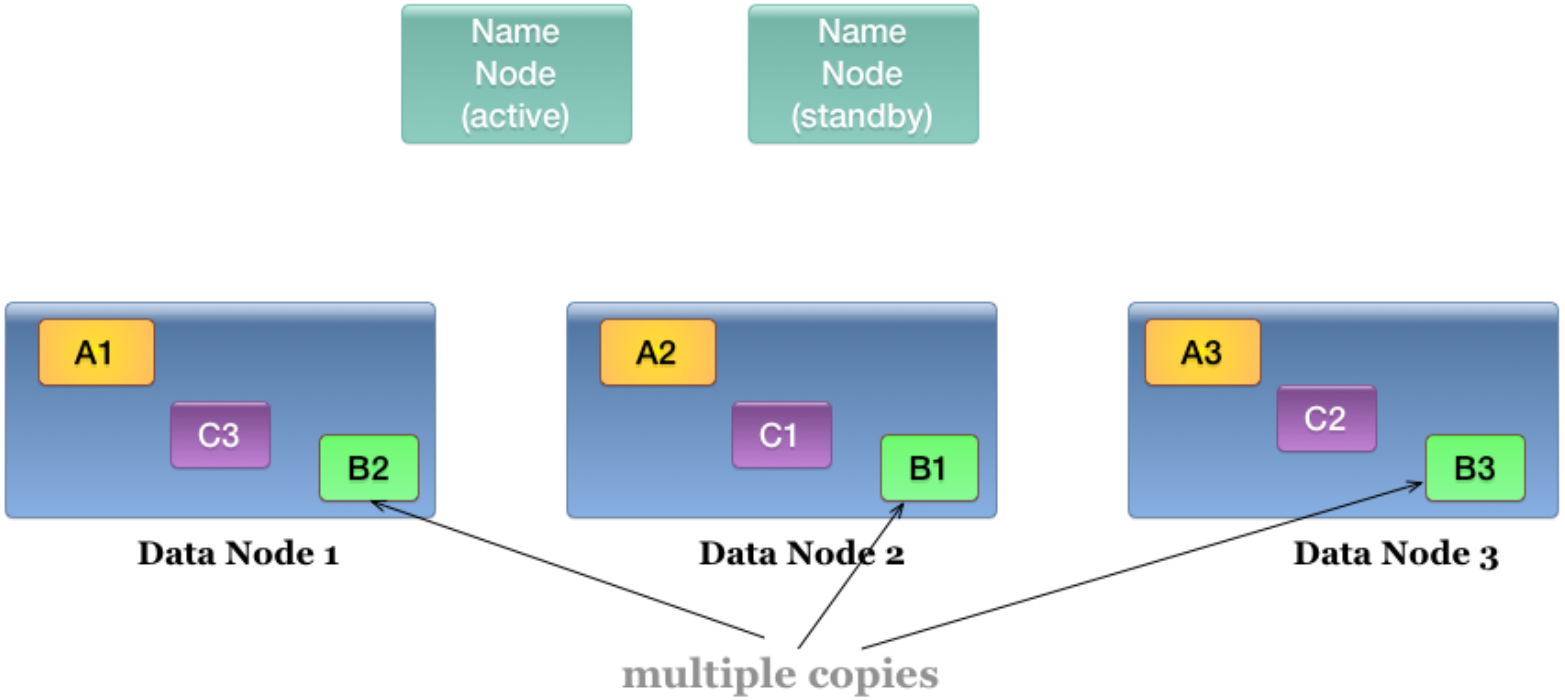
- **‘forever’ storage**
  - Scalable distributed file systems
  - Hadoop ! (HDFS actually)

- **‘real time store’**
  - Traditional RDBMS won’t work
    - Don’t scale well (or too expensive)
    - Rigid schema layout
  - NoSQL !

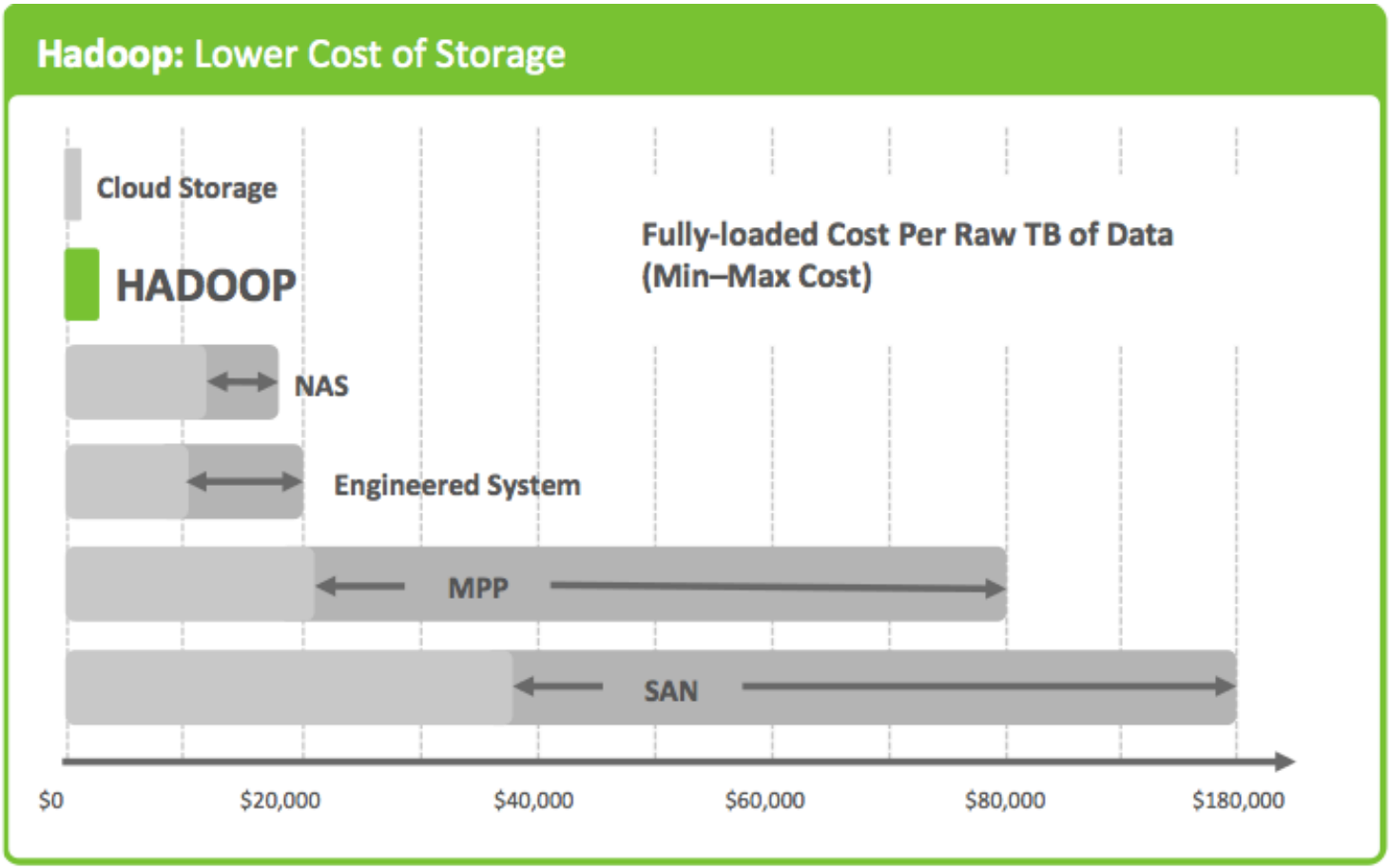
(3) STORAGE
HDFS (IN 20 SECS)

- **Distributed file system**
- **Runs on commodity servers**
  - → high ROI
- **Can keep ticking even when nodes go down**
  - → fault tolerant
- **Replicates data to prevent data loss in case of node failures**
  - → built in backup ☺
- **Scales to Peta bytes (horizontal scalability)**
- **Proven in the field**

(3) STORAGE
HDFS ARCHITECTURE



(3) STORAGE
COST OF BIG DATA



Source : hortonworks

(3) STORAGE
HDFS

- **Can handle big data**
- **Scales easily**
- **Cost effective**
- **“Source of Truth”**
  - Files are immutable within HDFS (new data is ‘appended’)
  - Audit friendly

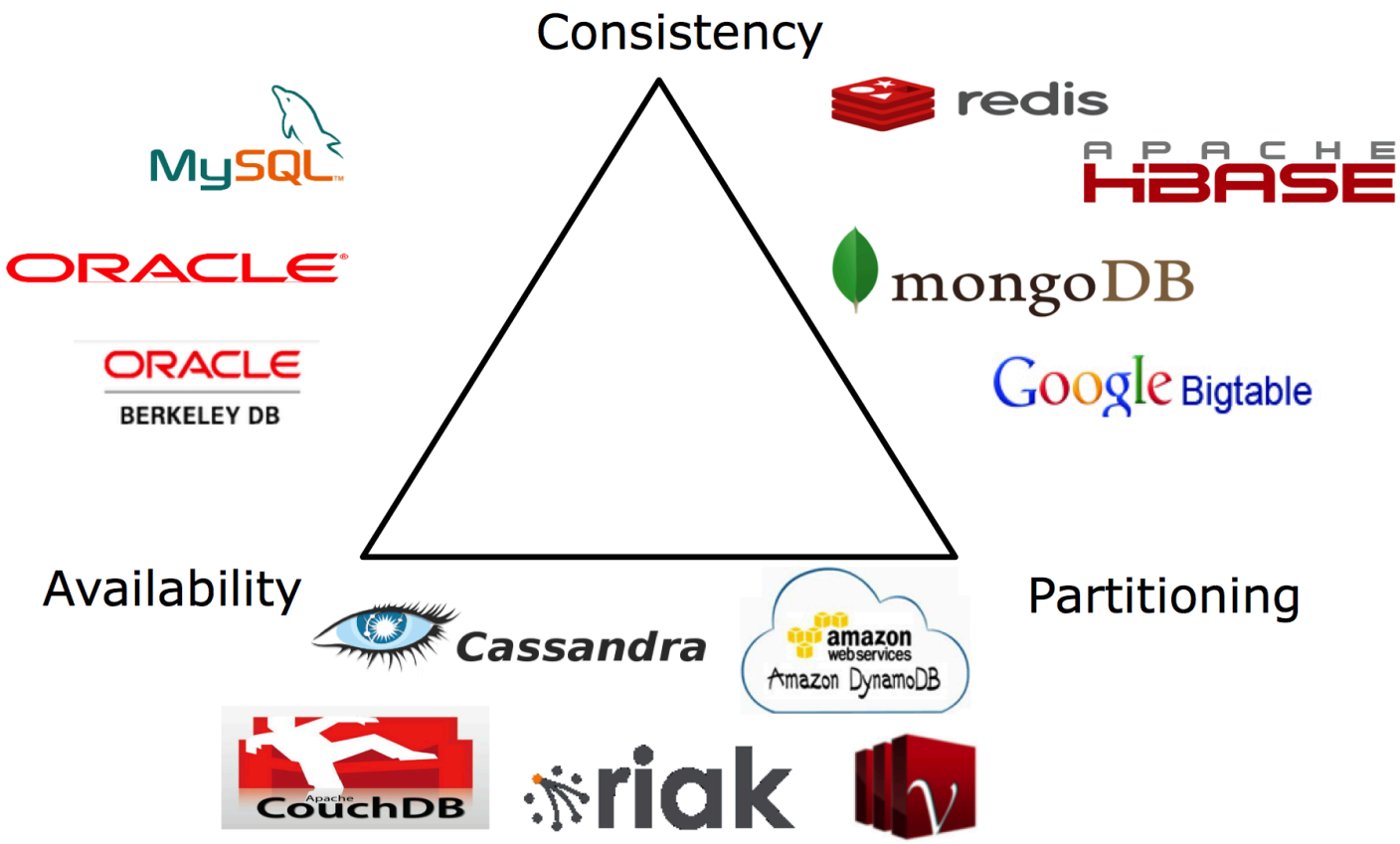
(3) STORAGE
CAPACITY PLANNING (HADOOP)

Variables (tweak these)	description	value	units		
Average daily ingest		1000	GB		
raw data node storage	eg. 12 disks x 3TB	36	TB		
replication	default 3	3			
space allocated for HDFS	HDFS 75% + Mapreduce 25%	75.00%			
growth per month	(not calculated)	0			
Calculation					
effective data storage per node		27	TB		
growth	**1 month**	**6 month**	**1 yr**	**2 yr**	
data size (TB)	90	540	1,080	2,160	
# data nodes	3.333333333	20	40	80	

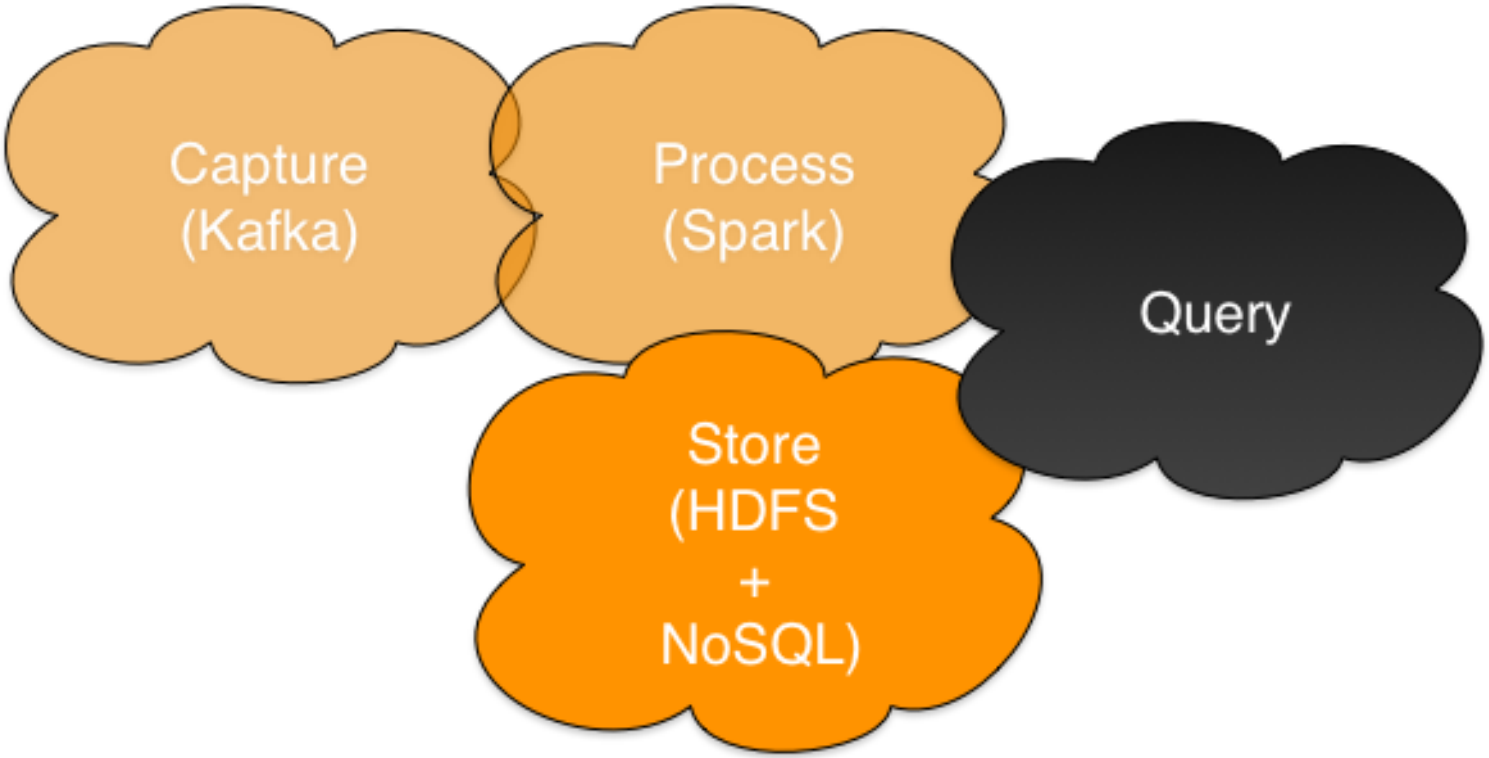
(3) STORAGE (REAL TIME) CHOICES FOR NOSQL

- **Too many ! ☺**
- **HBase**
  - Part of Hadoop eco system
  - Uses HDFS for storage
  - Provides consistent view of data
- **Cassandra**
  - Popular NoSQL store
  - No Single Point of Failure (SPOF) – ring architecture
  - No dependency on Hadoop
- **Accumulo**
  - Came out of NSA !
  - Uses HDFS for storage
  - Provides very good security (naturally !)

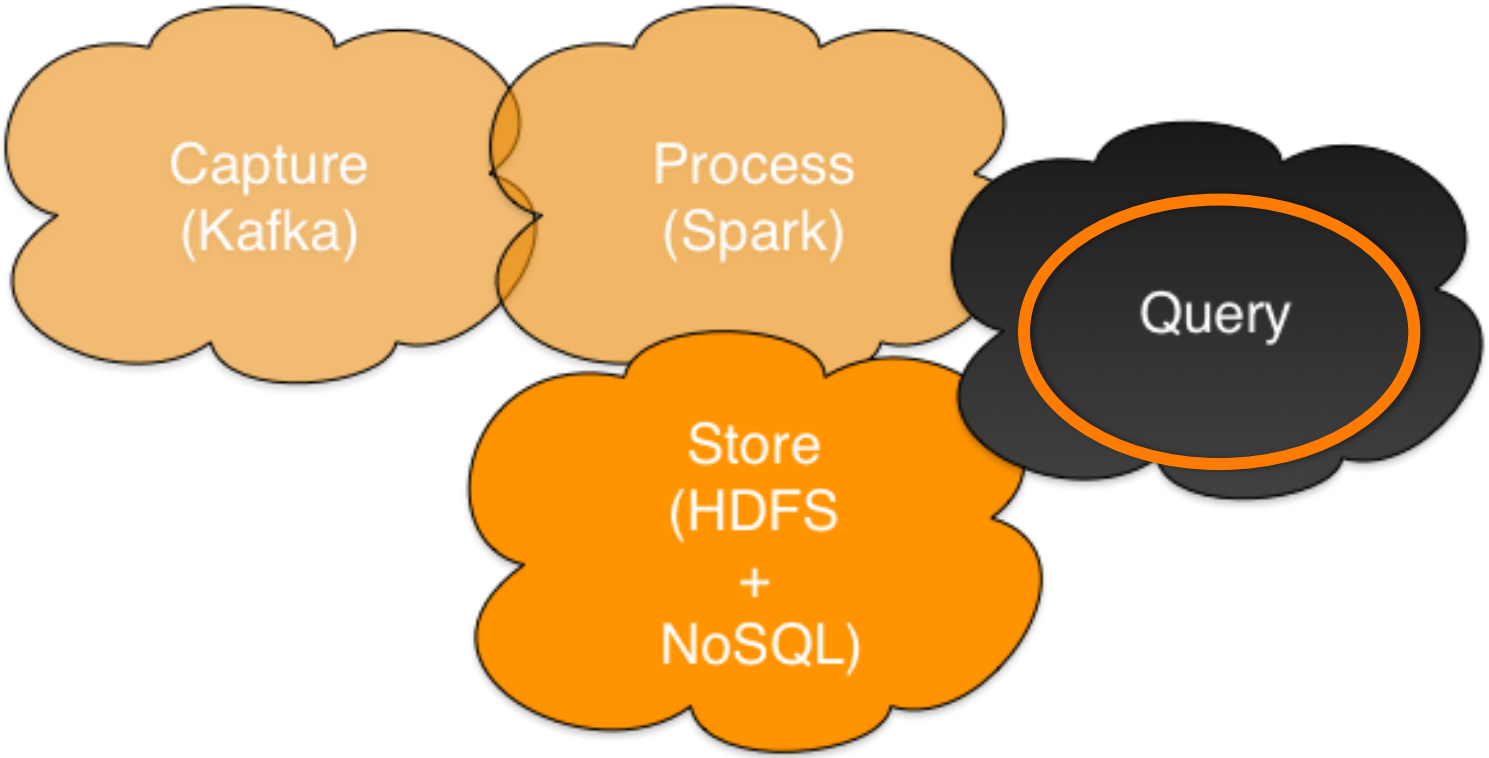
(3) STORAGE
CAP THEOREM



(3) STORAGE
ARCHITECTURE SO FAR



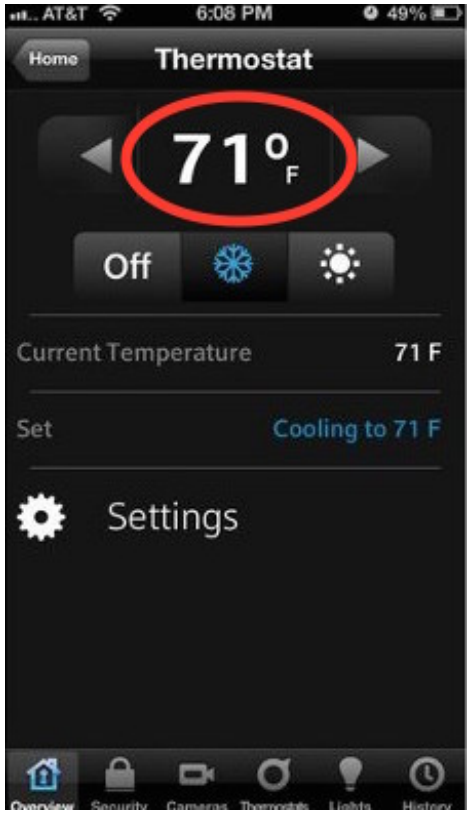
NEXT : QUERY



(3) QUERY REQUIREMENTS

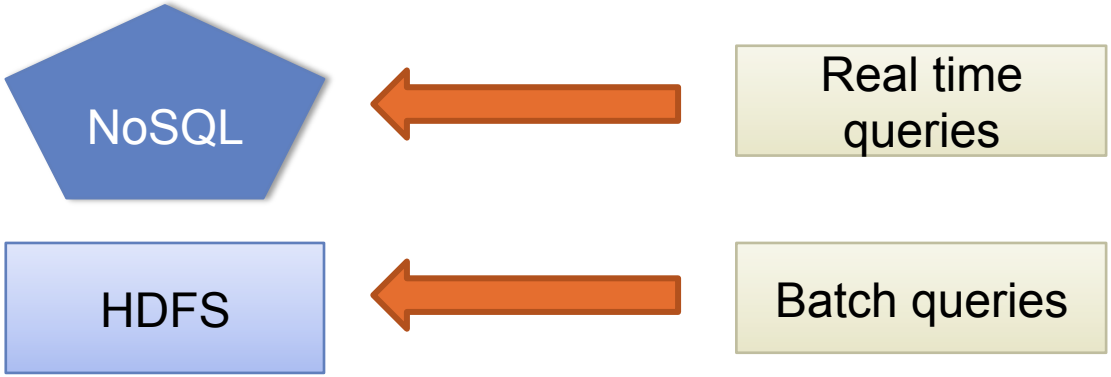
- **Real Time queries**
  - "what is the latest reading for the sensor id = 123"
  - Useful for building applications / dashboards
  - Latency : milli-seconds

- **Batch / Aggregate queries**
  - "What is the average temperature in zip code = 12345" ?
  - May need to go through large data points
  - Latency : 'batch' (minutes / hours)

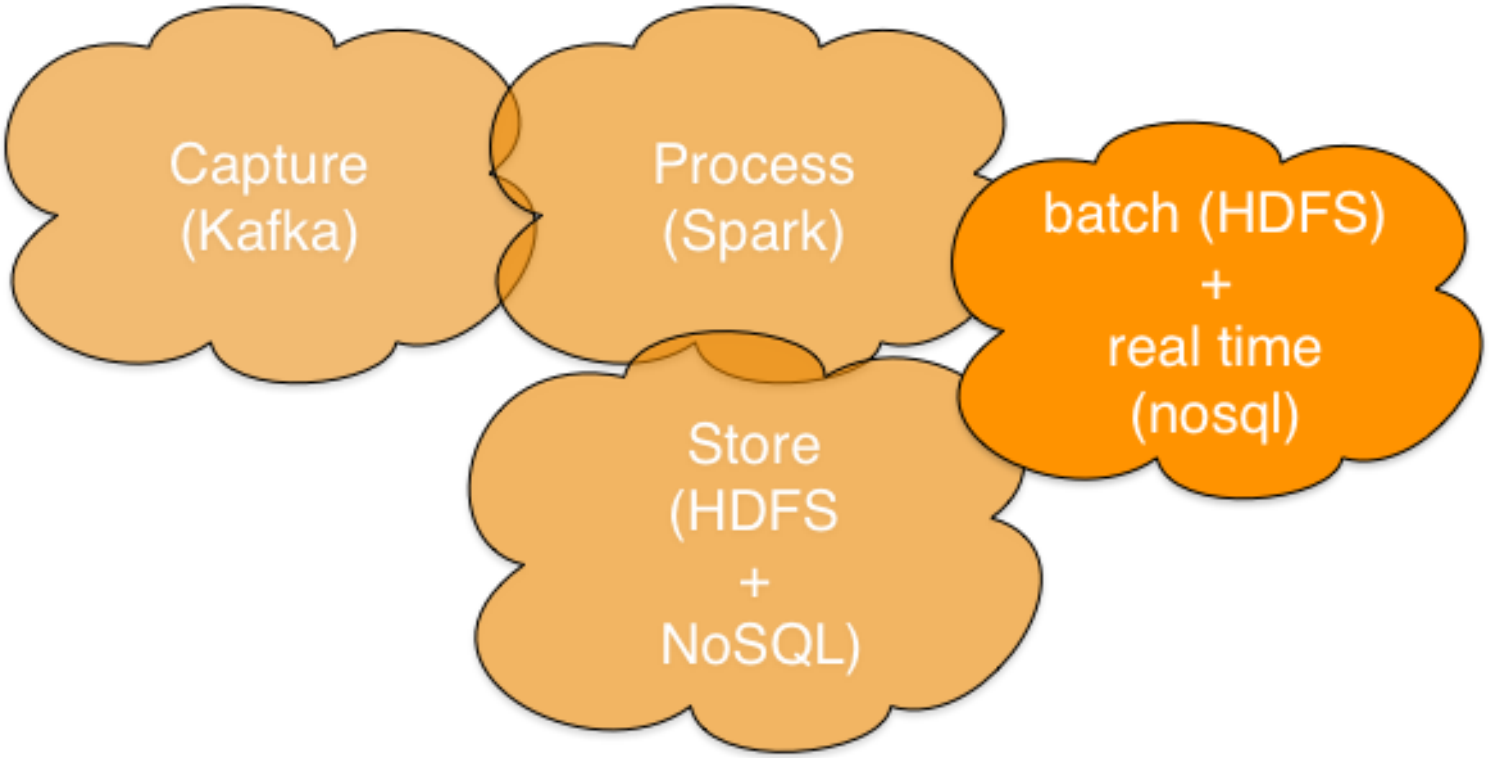


(3) QUERY SOLUTIONS

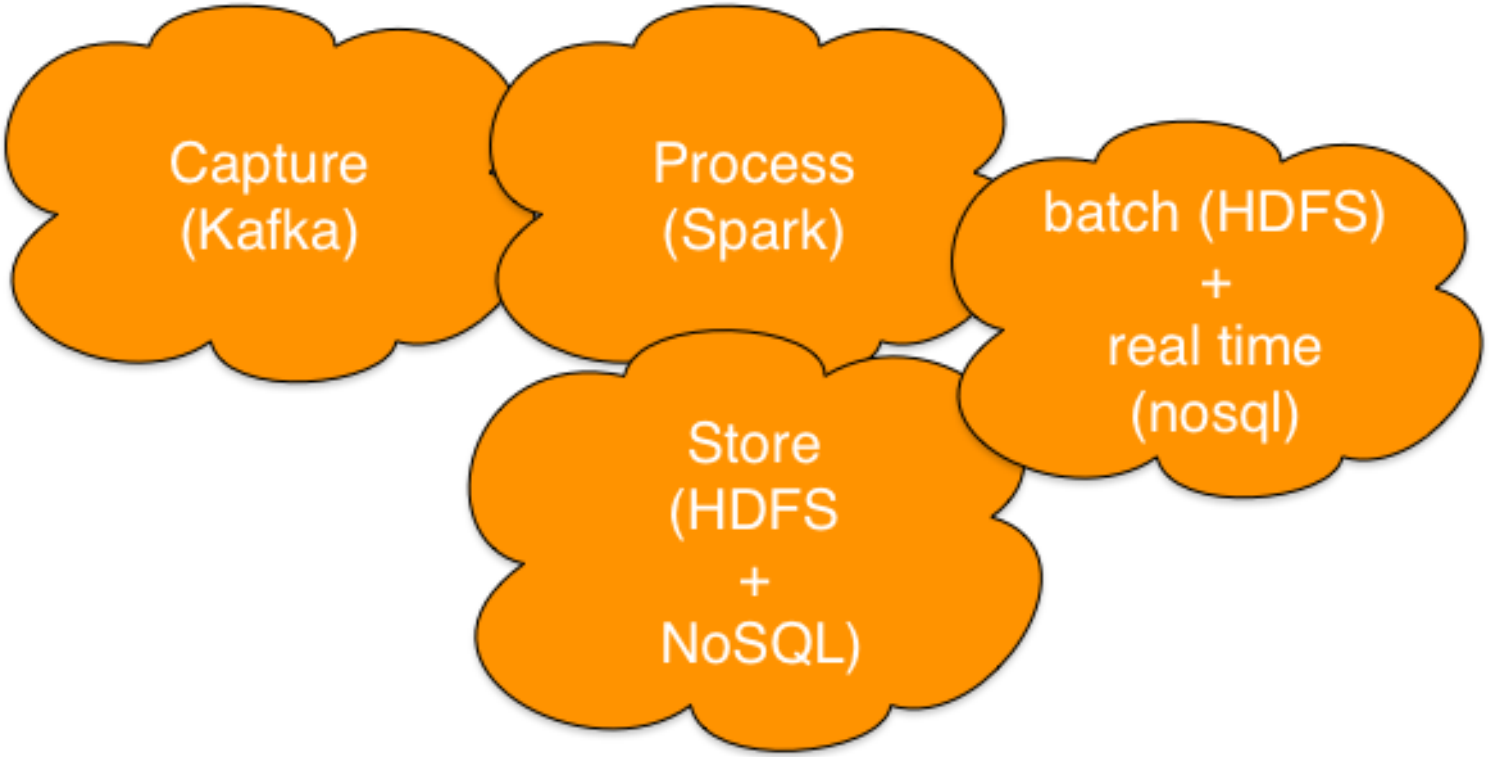
- **Batch queries**
  - Query data in HDFS (and or NoSQL)
  - Hadoop mapreduce (Pig / Hive)
  - Spark batch analytics
- **Real time queries**
  - Queries to go NoSQL store



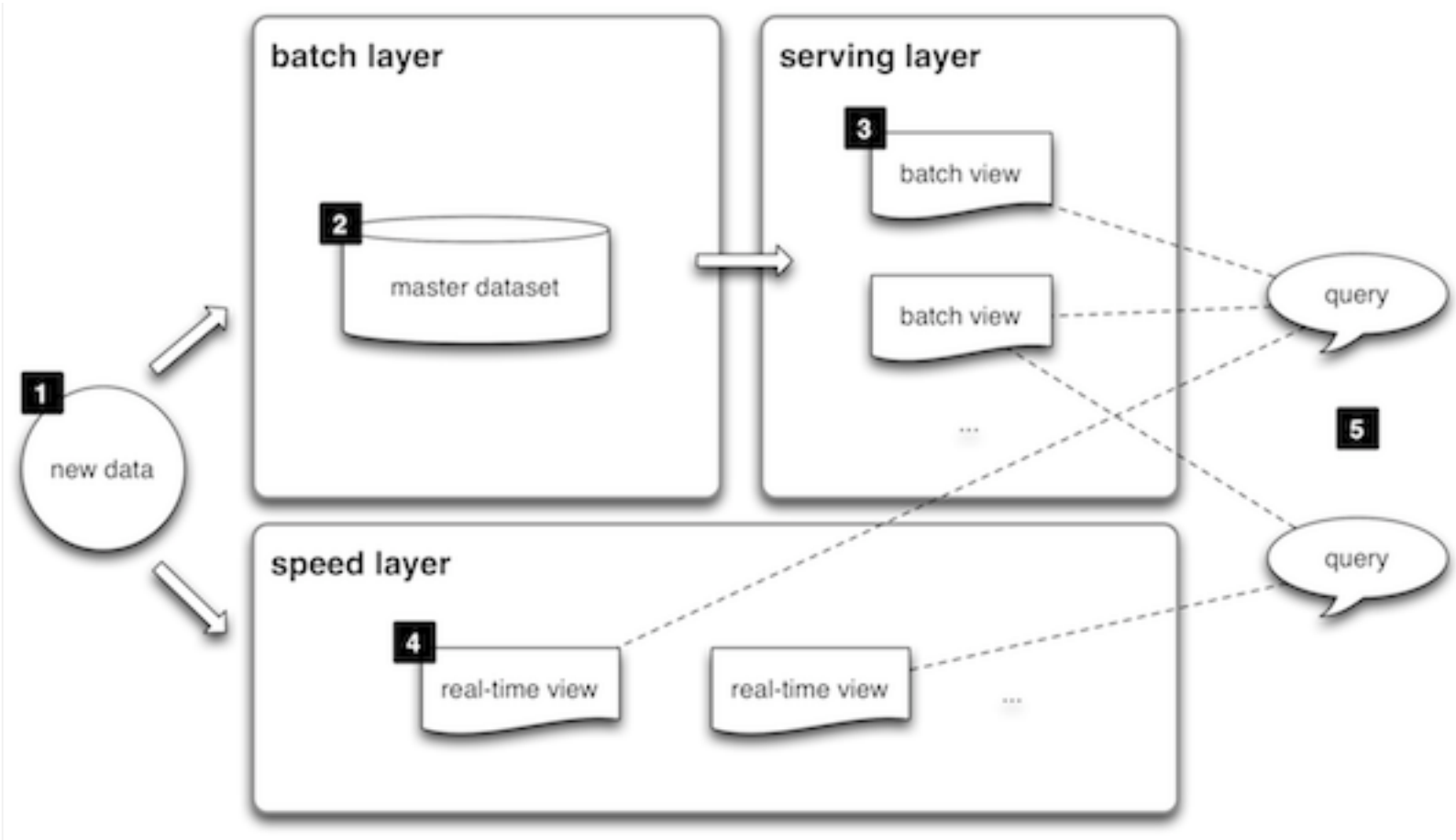
(3) QUERY ARCHITECTURE



FINAL ARCHITECTURE



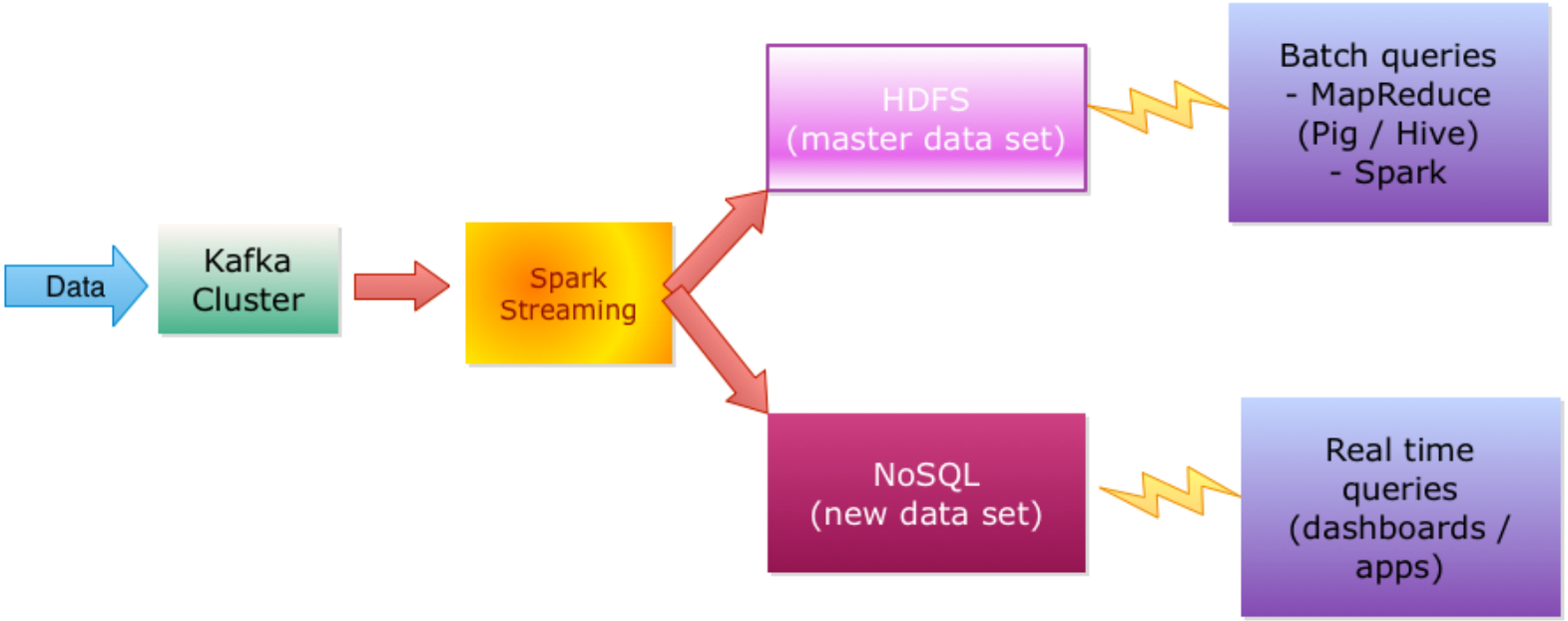
LAMBDA ARCHITECTURE



LAMBDA ARCHITECTURE EXPLAINED

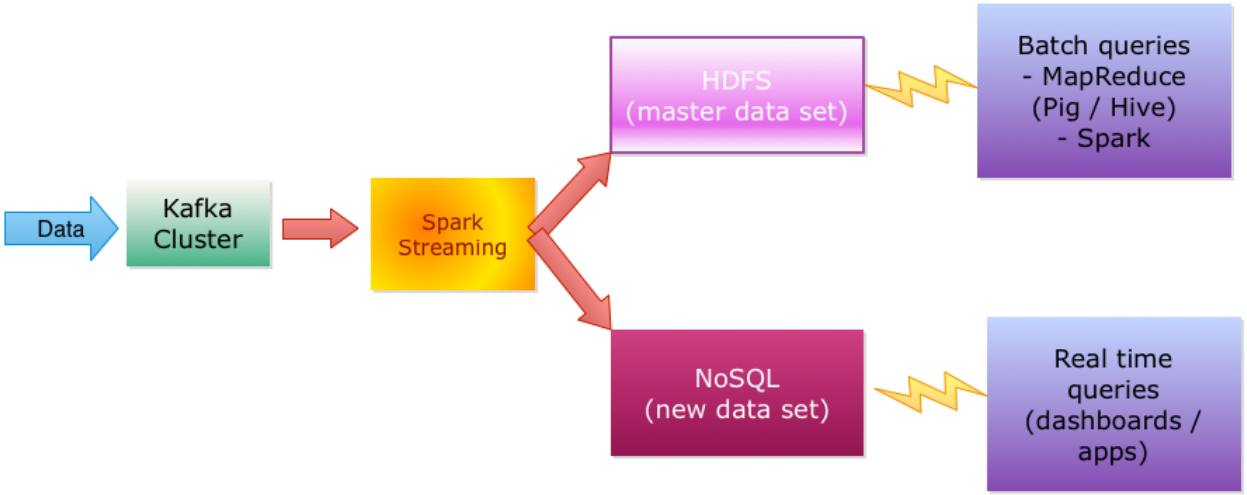
1. **All new data is sent to both batch layer and speed layer**
2. **Batch layer**
   - Holds master data set (immutable , append-only)
   - Answers batch queries
3. **Serving layer**
   - updates batch views so they can be queried adhoc
4. **Speed Layer**
   - Handles new data
   - Facilitates fast / real-time queries
5. **Query layer**
   - Answers queries using batch & real-time views

INCORPORATING LAMBDA ARCHITECTURE



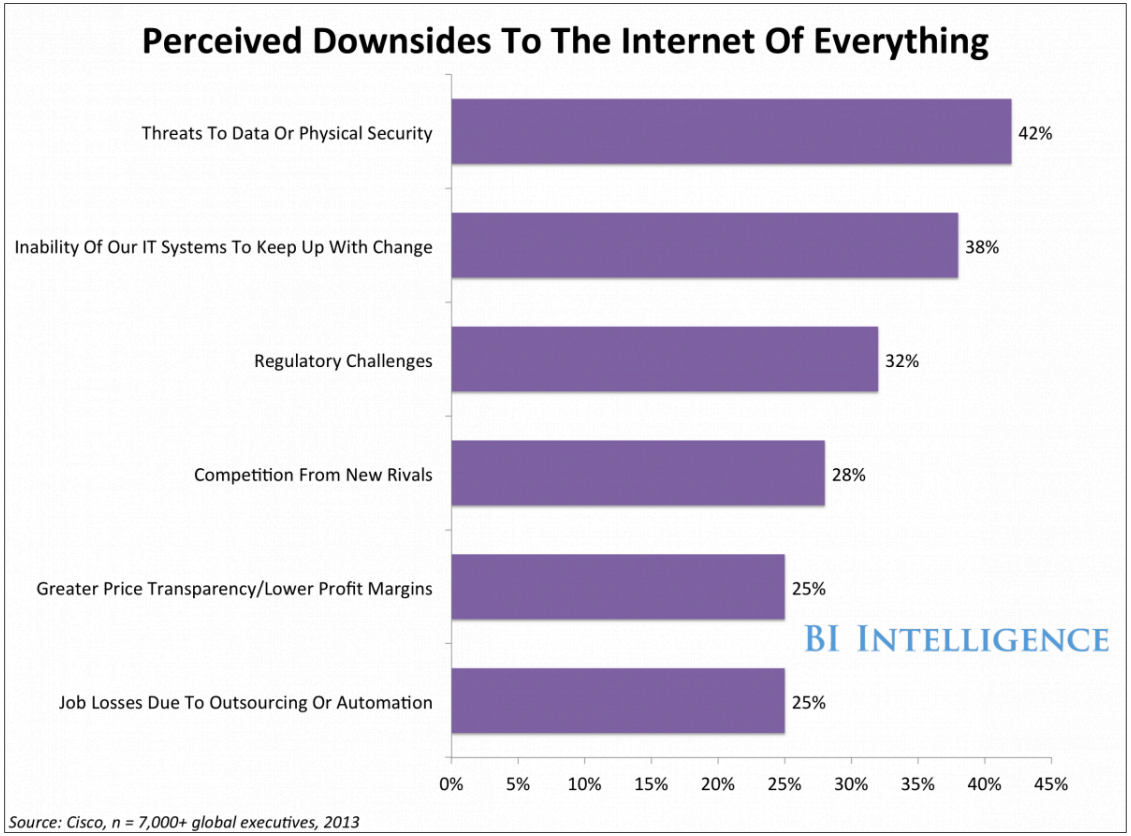
OUR ARCHITECTURE

- Each component is **scalable**
- Each component is **fault tolerant**
- Incorporates **best practices**
- All **open source** !



… AND ONE MORE THING…

- **Security !**



Source : businessinsider.com

HOW EVER…

≠

At scale nothing works as advertised !

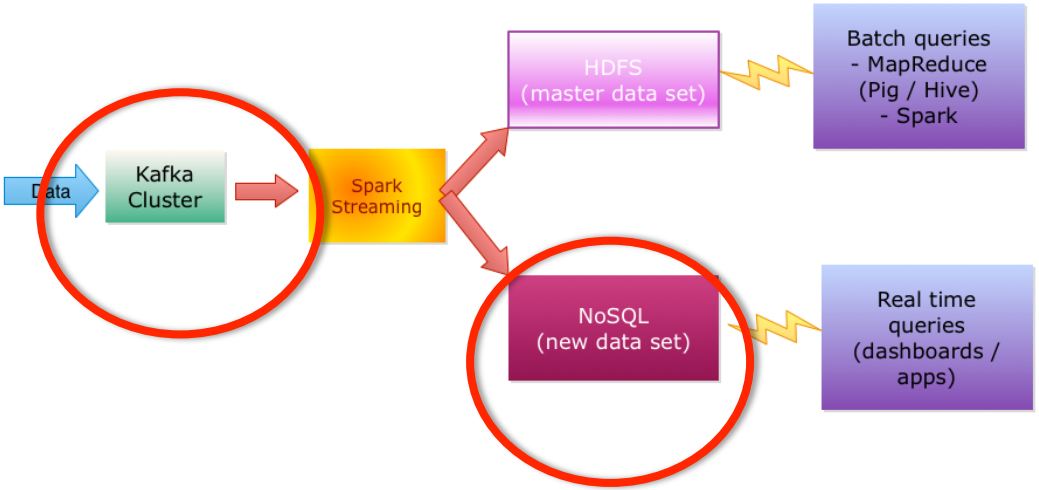
GOOD NEWS !

- **We'd like to build an open source, reference data platform for IoT / connected devices!**
- **Yes, open source ! ☺**
- **ElephantScale is a strong believer in open source**
  - "hadoop illuminated" – open source Hadoop book
  - Github.com/elephantscale
- **Best practices**
- **Bringing together lots of expertise in Big Data systems**
- **Register your interest**

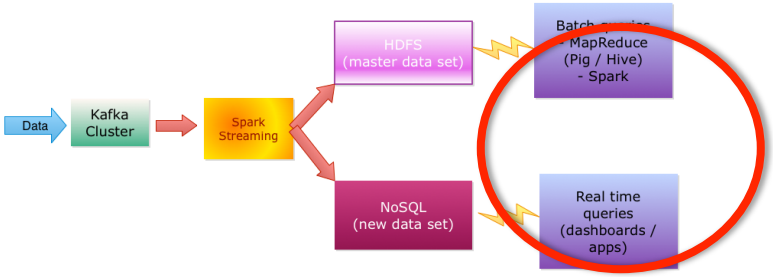
http://elephantscale.com/iotx/

GOALS FOR IOTX
http://elephantscale.com/iotx/

- **Use open-source proven components**
- **Capture :**
  - Kafka
  - Kinesis (AWS)
- **Processing : Spark Streaming**
- **Batch storage : Hadoop / HDFS**
- **Real Time Store : support multiple data stores**
  - Cassandra
  - Hbase
  - Accumulo
  - ???



GOALS FOR IOTX…
http://elephantscale.com/iotx/



- **Query templates using**
  - Spark
  - Hadoop Map Reduce (Pig / H
- **Incorporate third party libraries for**
  - Outlier detection (temperature is outside norms)
  - Trend detection (stock price is trending up)
  - Alerts (fire !)
- **Monitoring & Metrics (key !!)**
  - What's going in the system?
  - Host / system level (cpu / network ..etc) – easier
  - application level (e.g. find slow queries) – harder
  - Incorporate third party libraries

THANKS AND QUESTIONS?

"A Reference Architecture for Internet of Things (IoT)"

Sujee Maniyam

Founder / Principal @ ElephantScale

Expert Consulting + Training in Big Data technologies

sujee@elephantscale.com

Elephantscale.com

Project sign up page : http://elephantscale.com/iotx/

IMAGE CREDITS

- www.engadget.com
- Xfinity.com
- Tesla.com