Virtual IMS User Group
August 22nd 2017

Real-Time Streaming: IMS to Apache Kafka and Hadoop - 2017

Scott Quillicy
SQData

Agenda

Outline methods of streaming mainframe data to big data platforms

Set throughput / latency expectations for popular big data targets

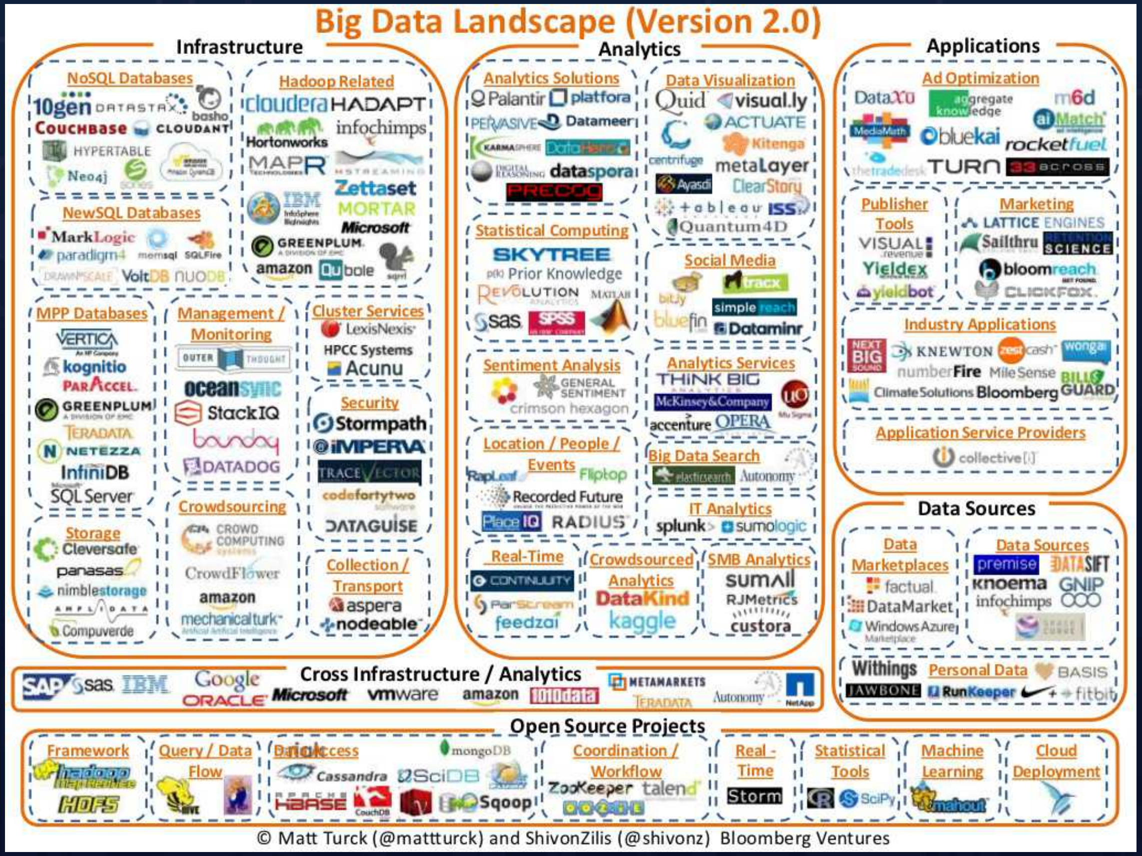
Highlight the top mistakes being made today and how to avoid them

Describe common mainframe streaming issues

Discuss general design / deployment considerations

Virtual IMS User Group
August 22nd 2017

You have a few choices (with more on the way…)



Big Data

The Reality: a large collection of data…in existence for 50+ years

Characteristics
- Significant amount of data
- Advanced analytics of disparate data
- Many different formats → structured, semi-structured, un-structured
- High rate of change

Exciting times ahead
- Large open source communities
- Rapid evolution of technology

Challenges
- Increasing data volumes → stress traditional RDBMS
- Computing and infrastructure costs to process / analyze
- Most companies in early stages of adoption

Why Real-Time Streaming of Mainframe Data to Big Data?

Analytics… Analytics… Analytics

- Decisions based on current information vs 24+ hour old data
- Quickly detect key events / trends
- Maintain a competitive advantage
- Provide better customer service
- Increase revenue / profitability

Real-Time vs. ETL

IDC study found that nearly 2/3rds of the data moved by ETL was at least 5 days old before reaching an analytics database.

Survey revealed that it takes at least 10 minutes to move 65% of CDC data into an analytics database.

75% of IT executives worry about data lag that might hurt their business.

27% said data disconnect is slowing productivity.

Over half of respondents said slow data is limiting operational efficiency.

The Great Divide

Today's Popular Big Data Stores

Hadoop HDFS
- Most commonly used Big Data store
- Foundation for other technologies (ie: Spark)
- Highly scalable

Hbase
- NO/SQL key-value store
- Tables split into column families
- Allows for Inserts, Updates
- Intended for real-time queries

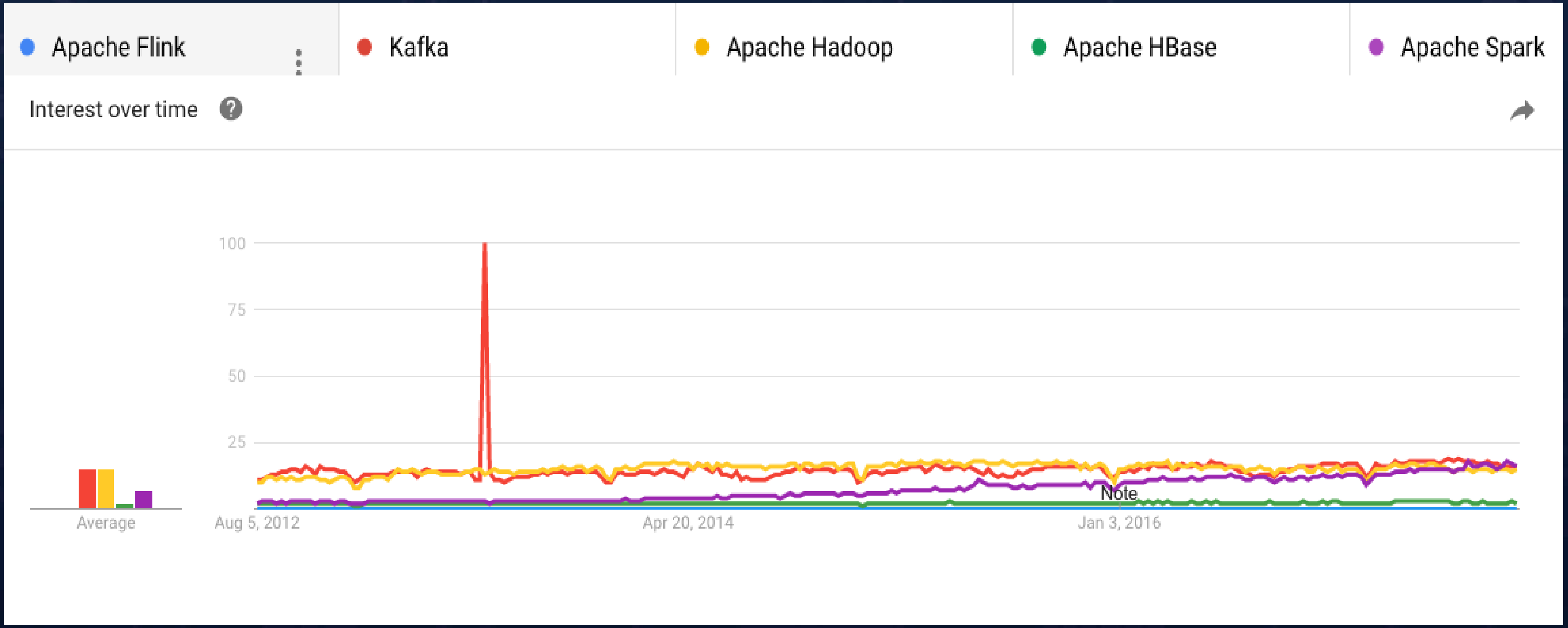
Hive
- Data warehouse infrastructure build on HDFS
- Allows for querying data stored on HDFS
- Runs only in batch → no interactive
- Intended for analyzing data collected over time

Kafka
- Ultra-fast message broker
- Streams data into most popular Big Data targets
- Multiple producers / consumers
- Ideal for real-time streaming

Other Popular Stores
- Cassandra
- MongoDB
- Spark*
- More appearing each day…

Virtual IMS User Group
August 22nd 2017

Interest over Time



Virtual IMS User Group
August 22nd 2017

Top Mistakes Being Made Today

Top Mistakes Being Made Today

No clear use-case(s)

"Build it and they will come" approach

- Great way to ensure failure
- Minimal focus on business needs
- Often caused by pressure to deploy
- Big Data solution

Data collection overkill

"Everything needs to be in data lakes" approach"

- Wastes time moving data with little business value
- Guarantees timeline and cost overruns
- Value does not exceed the expense (HW, SW, People)

Lack of an enterprise approach / strategy

"We can do it on our own" approach

- Independent deployments → departmental fiefdoms
- Minimal structure → easy way to run amok
- More costly to the business

Technology

- "Just copy the data as is into the data lake" approach
- Minimal understanding of mainframe in general
- Non-relational sources pose a significant challenge
  - IMS / VSAM
  - Re-defines repeating groups and weak Data Types
- Mainframe discipline is often lost on Big Data
- Improper tool selection
  - Not aligned with enterprise
  - Not strategic → could become obsolete
  - Increased support risks

No Clear Use-Cases

Key → Business users MUST be involved from the beginning
Pressure to deploy a Big Data solution plays a role

Use case must be clearly defined
- Identify source data elements
- Data delivery → real time vs. periodic ETL
- Success criteria fully understood

Use an agile methodology
- Iterative delivery
- Small, achievable milestones
- Start with most important data
- Success realized sooner

Leverage DevOps
- Data scientists
- Business analysts
- Technical operations
- Quality assurance

Data Collection Overkill

Key: focus on important business data first

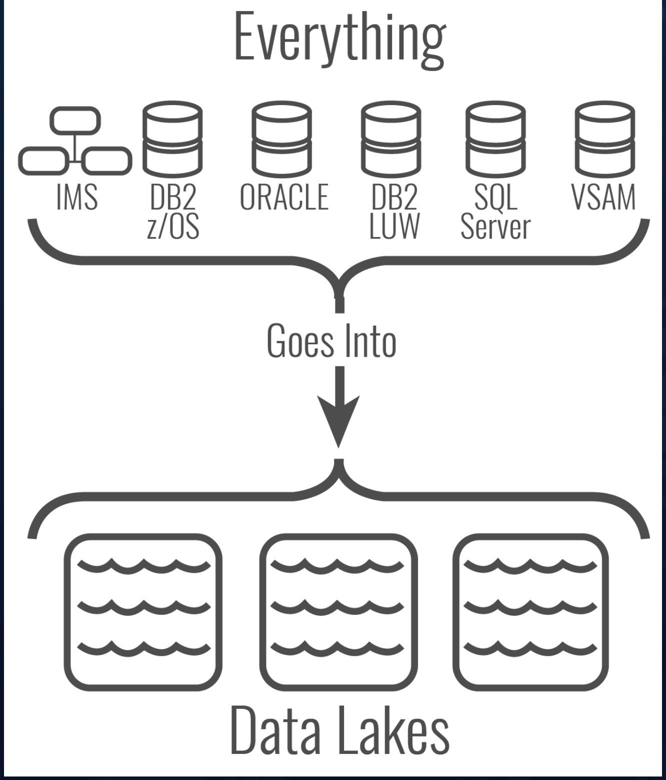
- The project that is rarely completed
- Similar to the old enterprise data warehouse
- Resource intensive
- Success criteria fully understood

Approach in small increments

- Realize success early
- Learn from mistakes
- Manageable costs and time

Involve the business

- They may “want everything”
- Identify key objectives
- Prioritize by importance
- Leverage DevOps / Agile

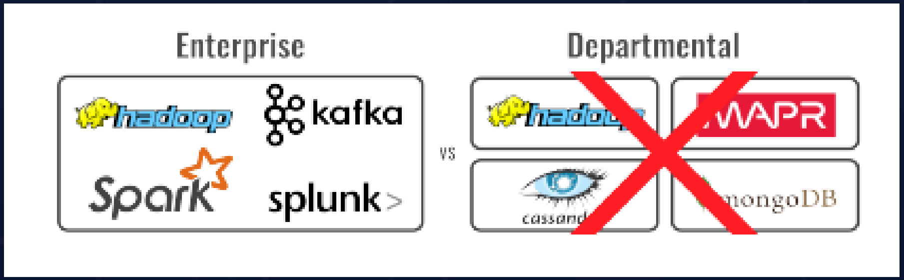


Lack of an Enterprise Approach / Strategy

Key: Deploy on an Enterprise Platform

Maintain a competitive advantage

- Provide better customer service
- Increase revenue / profitability
- Faster delivery → despite the "I/T Involvement is too much red tape"
- Reduced costs



Challenges

- Departmental fiefdoms → "it's our budget…we'll do it our way"
- Everyone has a different opinion on what is the best option
- Departments may be in I/T realm vs the business

Not Setting Proper Expectations

Reality → Projects are at least a 2 to 3 year effort

Relying on estimates from technical folks

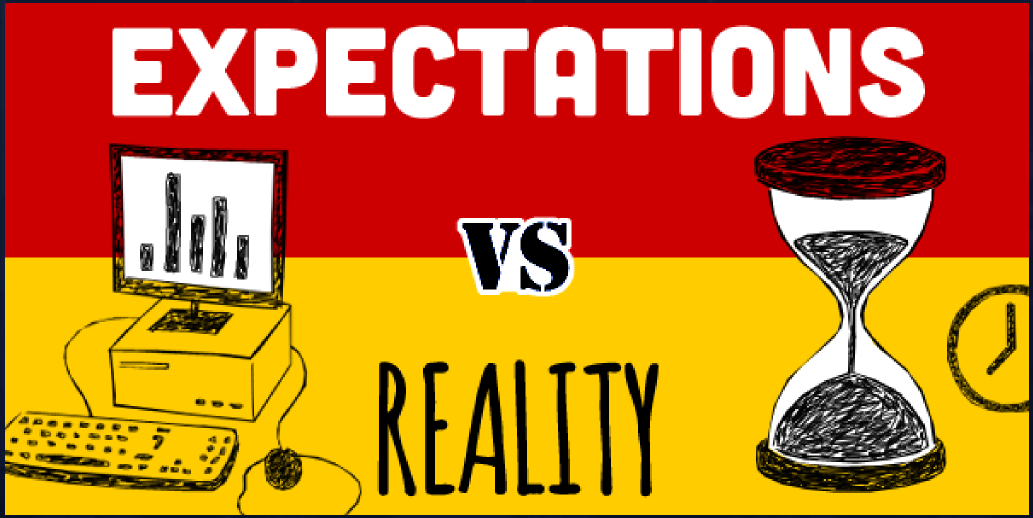
- Historically optimistic
- Do not anticipate obstacles
- Not understanding real-time vs. ETL
- Use the tech estimate x 2+

Success can be realized early

- Small subset of important data
- Assume DevOps / Agile
- Base infrastructure in place
- Technically competent team

Learn from others

- Big Data user groups
- Tech conferences
- Consultants



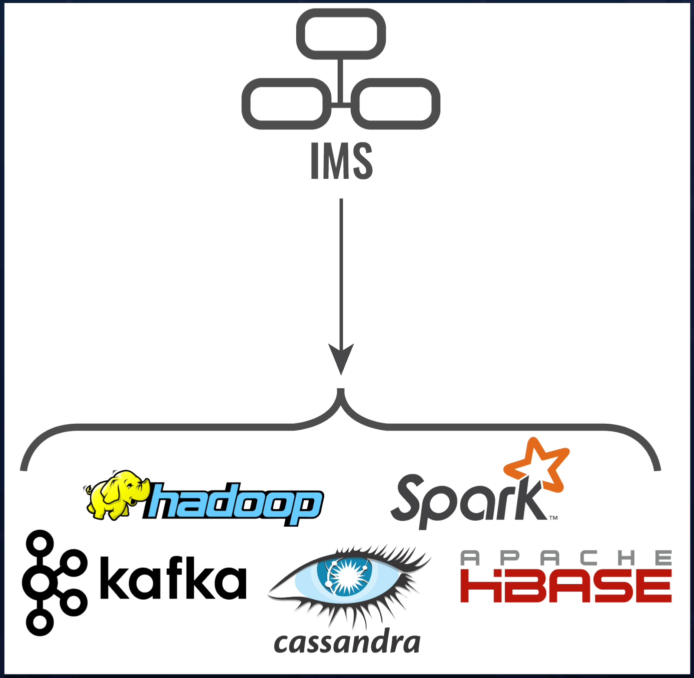
Technology

Minimal understanding of mainframe data
Particularly non-relational → IMS / VSAM

Common "I had no clue" items

- IMS structures in general
- Repeating groups (occurs)
- Redefines
- Dates
- Invalid data
- 'Special' fields (bits, Y2K, etc.)

Code page translation
Transaction consistency
Streaming vs. ETL
Target apply concepts / streaming
Normalization vs. denormalization
Not likely to get better...



A Note on Product Selection

Repositories / analytics
- Open source
- Large communities
- Proven results
- Beware of vendor lock

Supporting tools → ETL, replication
- Typically requires more than one
- Of little value if source data not understood
- Select the best tool for the use case → i.e. mainframe vs twitter

Licensing model considerations
- Typically subscription-based → traditional license + maintenance on the way out
- Optimal → licensing based on business use case
- Should be able to discontinue at any time → no long term commitment

Customer Examples

Use case → sales information into Big Data

- Tool selection → Cassandra
- Grew to 200 nodes
- Project cost → 2 years and $10M+
- Real-time updates were an afterthought
- Result → failed → nobody is using it
- Next steps → reworking by enterprise group into Hadoop / Spark

Use case → financial information into Big Data

- Tool selection → MongoDB
- Significant amount of data (multi-TB)
- Grew to 100 nodes
- Project cost → 1.5 Years and $6M+
- Did not realize Mongo does not scale well until it was too late
- Result → failed → not usable
- Next steps → trying to migrate to Hadoop

Customer Examples (cont)

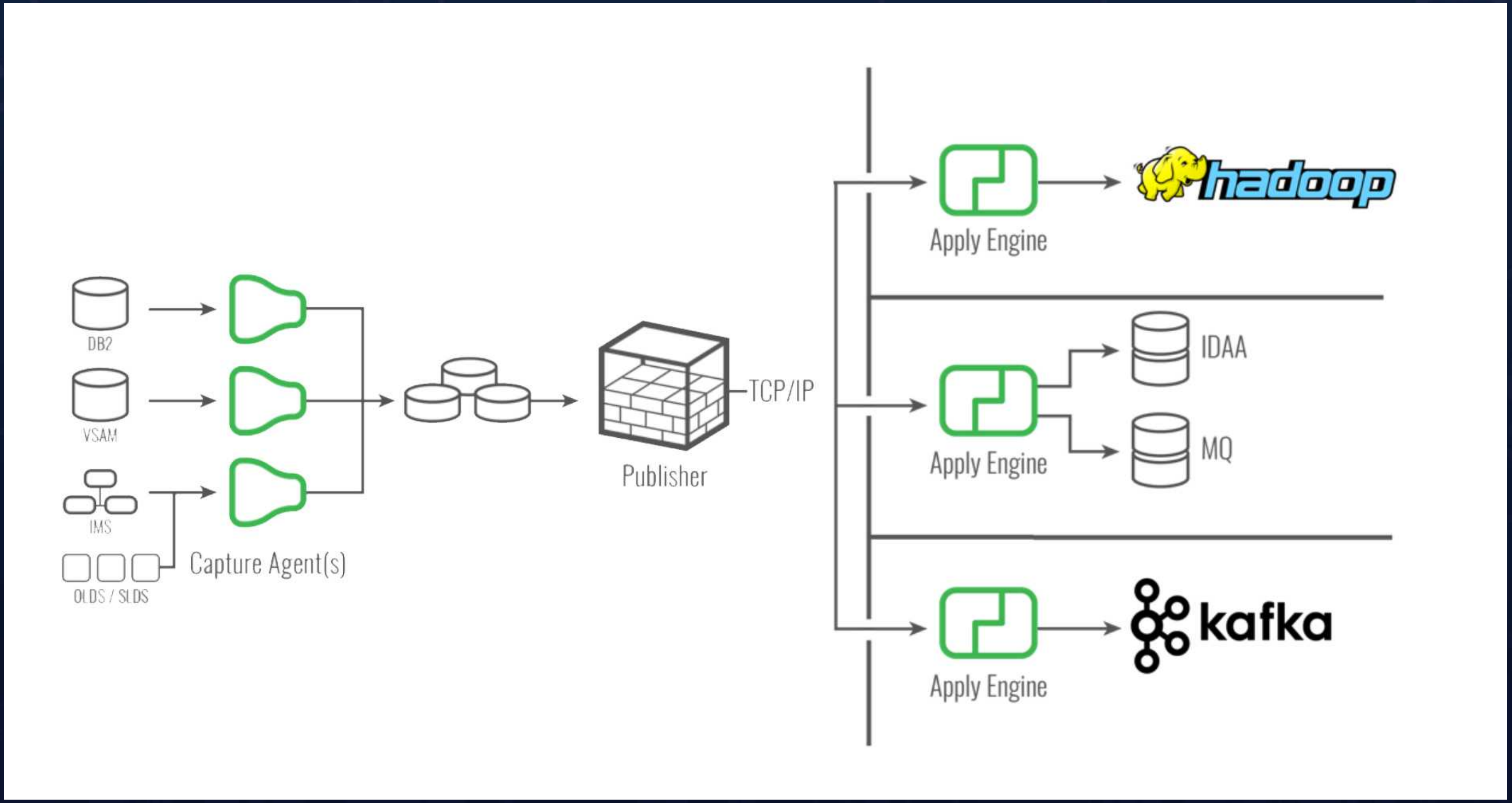
Use case → financial institution

- Tool selection → Hadoop, kafka, Spark
- Data dump without understanding relevance or relationships
- Project cost → 2+ Years until project cancelled
- Spent a LOT of time just trying to copy the data → with mixed results
- Result → failed → not usable
- Next steps → approach in smaller increments → leverage what has been done

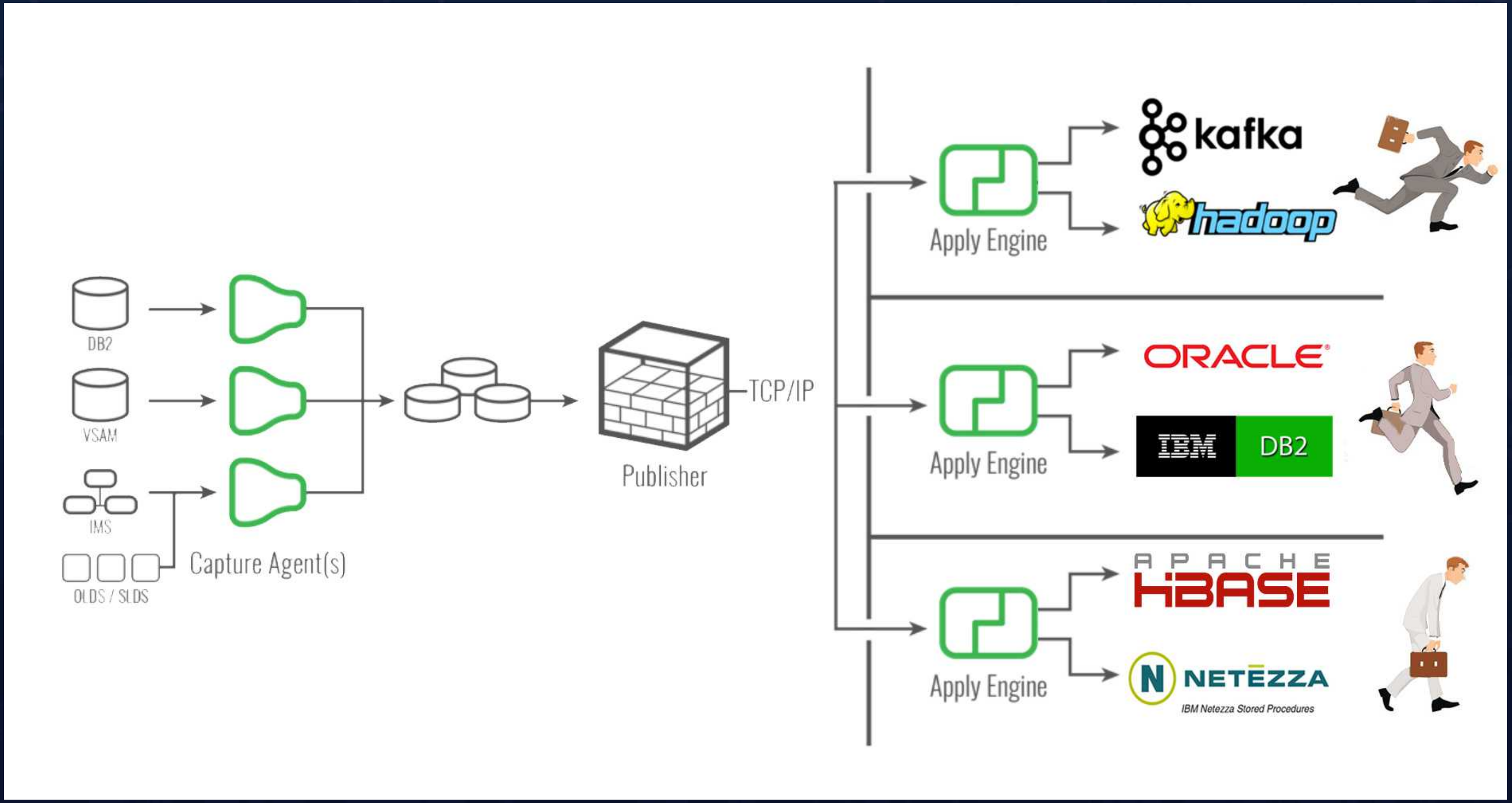
Mainframe Streaming

Virtual IMS User Group
August 22nd 2017

Mainframe Data Streaming Illustration



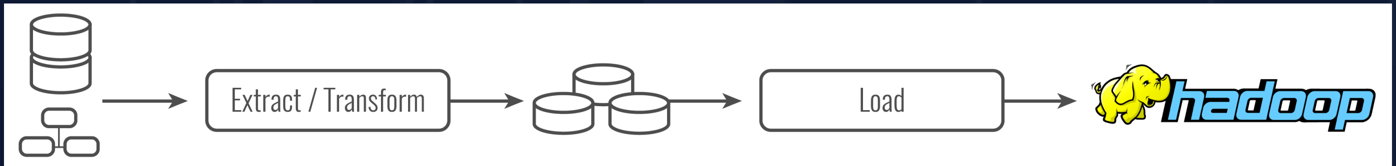
Target Speed and Effect on Latency



The Role of ETL and CDC

ETL (Extract, Transform, Load):

- Full data extract / load
- Data transformation logic defined in this step → reused by CDC
- Should be run against live data
- Should minimize data landing



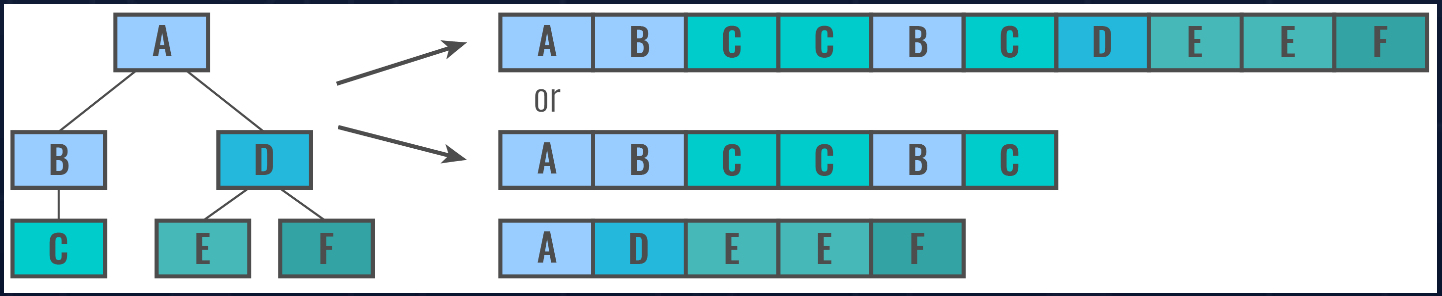
CDC (Changed Data Capture):

- Move only data that has changed
- Re-use data transformation logic from ETL
- Near-real-time / deferred latency
- Allows for time series analytics

ETL and Changed Data Capture (CDC)

ETL

- High level of control over level of de-normalization
- Can combine many source records/rows in target row/document
- Requires that ETL tool can handle consolidation during extract



Changed Data Capture

- *May dictate that target not de-normalized → depending on the target store*
- Target lookups may be required

Common Mainframe Data Challenges

Code page translation (CCSIDs)

Invalid data

- Non-numeric data in numeric fields
- Binary zeros in packed fields (or any field)
- Invalid data in character fields

Dates

- Must be decoded / validated if target column is DATE or TIMESTAMP
- May require knowledge of Y2K implementation
- Allow extra time for date intensive applications

Repeating groups

- Sparse arrays
- Number of elements
- Will probably be de-normalized

Redefines

Binary / 'Special' Fields

- Common in older applications
- Developed in 1970s / 80s
- Generally requires application
- Specific translation

CDC / ETL Data Format(s)

Recommended formats:

- JSON
- Avro
- Binary

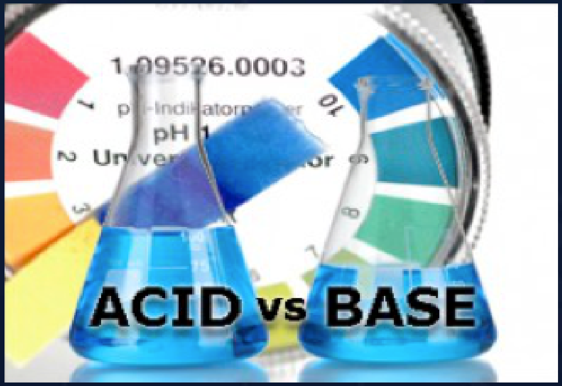
JSON recommended for data validation
Avro recommended for production deployment
Sample update CDC record in JSON format


```
{"DEPT": {
  "database": "EMPLOYEE",
  "change_op” : “U”,
  “change_time": "2015-10-15 16:45:32.72543",
  “after_image” : {
      "deptno": “A00”,
      "deptname": “SPIFFY COMPUTER SERVICE DIV.”,
      “mgrno” : “000010”,
      “admrdept” : “A00”,
      “location” : “Chicago”
   },
  “before_image” : {
      "deptno": “A00”,
      "deptname": “SPIFFY COMPUTER SERVICE DIV.”,
      “mgrno” : “000010”,
      “admrdept” : “A00”,
      “location” : “Dallas”
   }
}}
```

Acid vs. Base

ACID

- Guarantees DB transactions are processed reliably
- Atomicity → all or nothing
- Consistency → one valid state to another
- Isolation → concurrency
- Durability → once a transaction commits, it remains committed

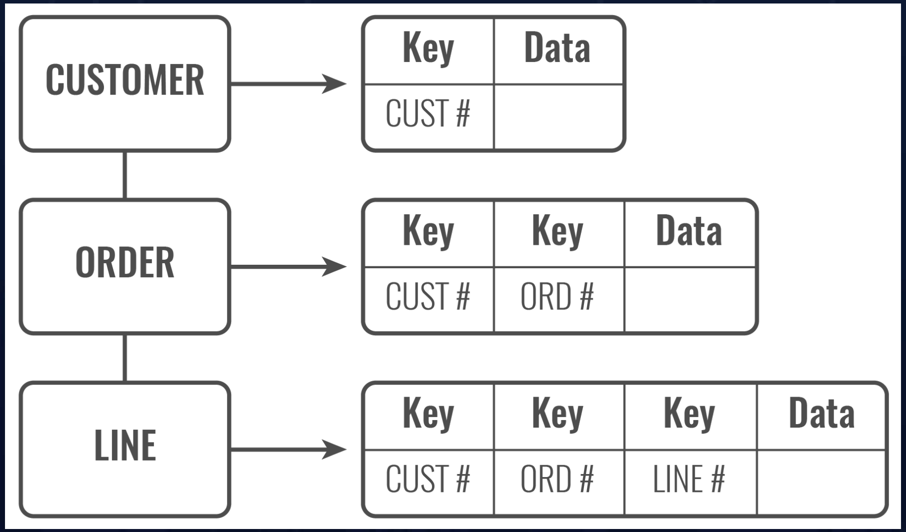


BASE

- “Eventually consistent”
- Basically available → data is there…no guarantees on consistency
- Soft state → data changing may not reflect commit scope
- Data will eventually be consistent

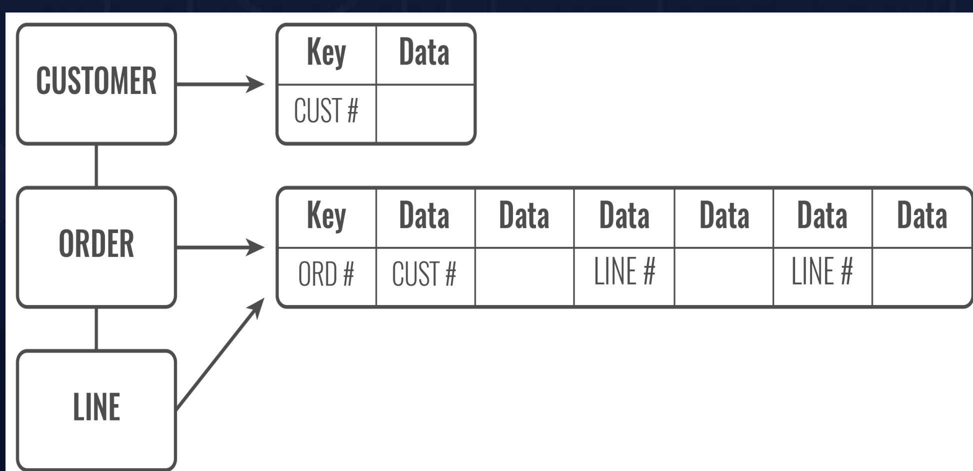
Design: Traditional IMS / VSAM to Relational

Each segment maps to one (1) or more tables
Strong target data types may require additional transformation
Tendency to over design / over normalize
Still required for relational type targets (PDA, Netezza, Teradata, etc.)



Design: IMS / VSAM to Big Data

De-normalized / minimal normalization
Still requires transformation (dates, binary values, etc.)
Good news → source structures already setup for Big Data



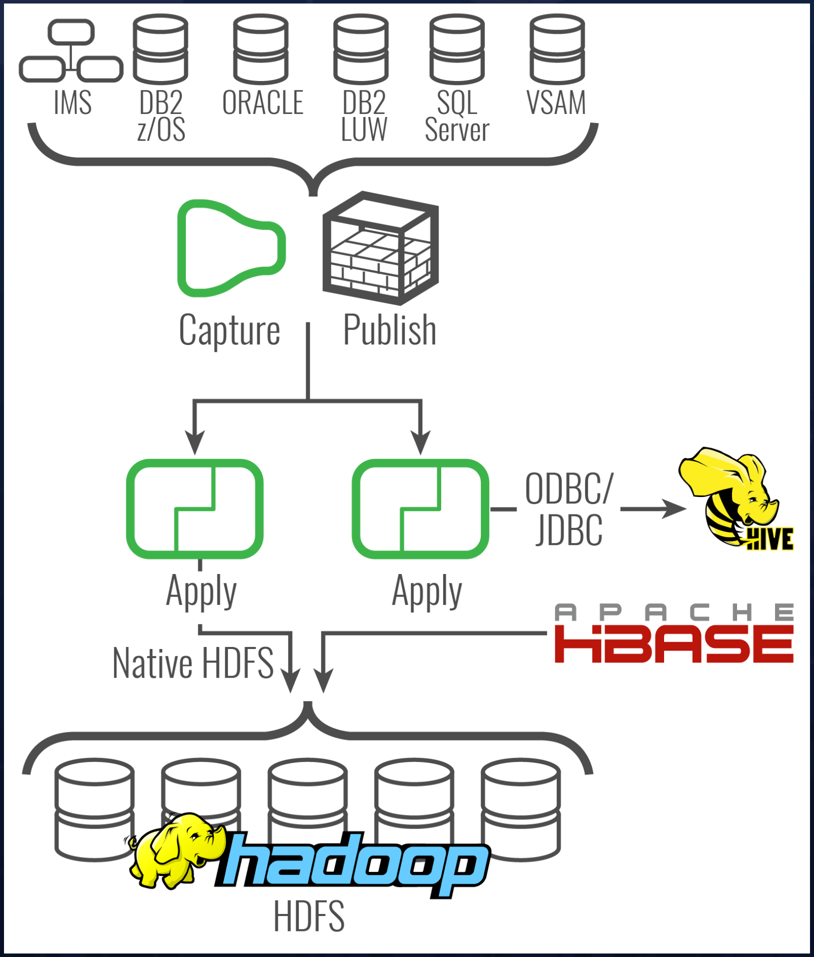

```
{ "company_name" : "Acme",
  "cust_no"      : "20223",
  "contact" :{ "name" : "Jane Smith",
              "address" : "123 Maple Street",
              "city" : "Pretendville",
              "state" : "NY",
              "zip"   : "12345" }
}
```

```
{ "order_no" : "12345",
  "cust_no"  : "20223",
  "price"  : 23.95,
  "Lines" : { "item" : "Widget1",
              "qty"  : "6",
              "cost" : "2.43"
              "item  : "Widge2y"
              "qty"  : "1",
              "cost" : "9.37"
            },
}
```

Streaming to Hadoop

HDFS format → CSV, JSON, Avro
Typical use → multiple files for same content
- File size based on # records / time interval
- Requires multi-file management

Partitioning → based on source value(s)
- Not native in HDFS
- Based on source data value(s)
- Requires cross-partition multi-file management

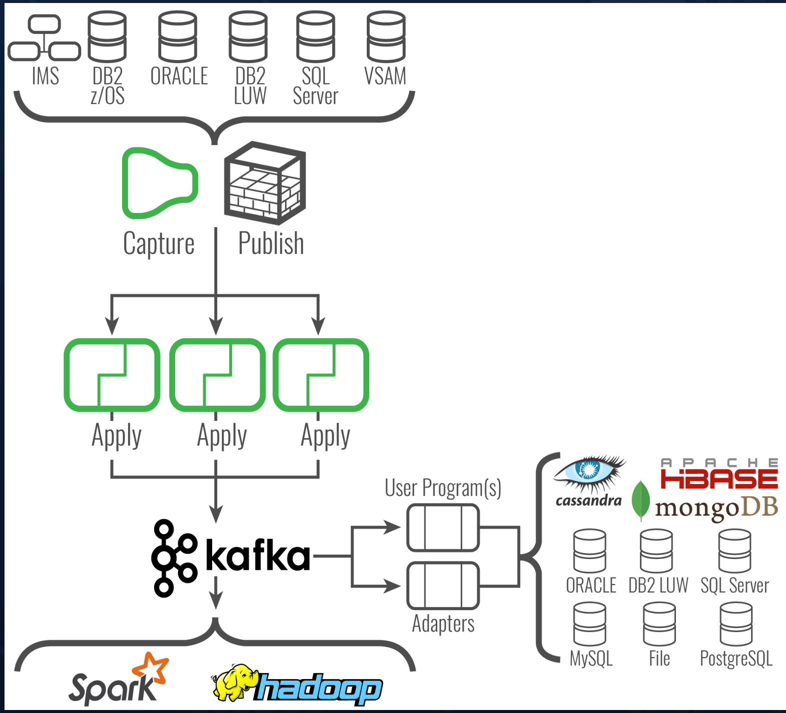


kafka

High-throughput, low-latency message broker
Open sourced by LinkedIn 2011 / Apache 2012
Supports a variety of targets → more on the way
Leverage JSON/Avro message format for CDC

Use cases:

- Basic messaging → similar to MQ
- Website activity tracking
- Metrics collection / monitoring
- Log aggregation
- Streaming



Best Practices Summary

Approach with a comprehensive strategy
- Common infrastructure / tools / support
- Established methods (DevOps / Agile)
- Beware the “fiefdoms”

Involve the business from the beginning
- They understand the source data
- They know the order of importance
- They can assist in design validation, QA, etc.

Avoid the data collection overkill
- Time and $$$ killer
- Focus on most important data first
- Iterate through remaining data → prioritize by importance

Set proper expectations
- 2 to 3 years minimum is expected…for an entire project
- Deliver in Increments → most important data first

Understand IMS data is ‘special’
- Patience is key
- Ask for help…

Q&A

Virtual IMS User Group
August 22nd 2017

Scott Quillicy
SQData
squillicy@sqdata.com

Real-Time Streaming:
IMS to Apache Kafka and Hadoop