IMS to Big Data

Common Traits for Success

Prepared for the:

Virtual IMS User Group

4 October 2016

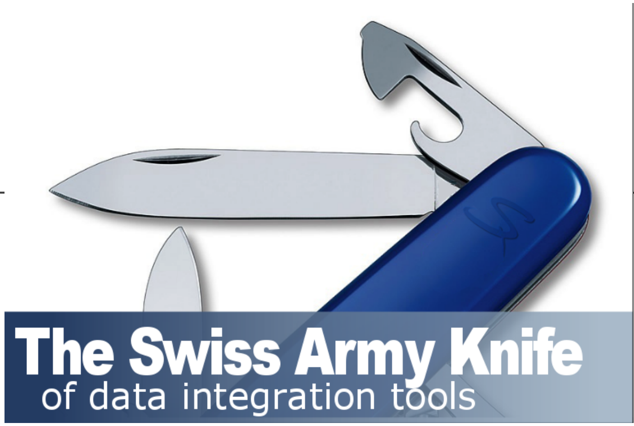
Objectives

- **Primary** → Outline the Challenges with Streaming Mainframe to Big Data
- Highlight the Top 5 Mistakes Customers Make
- Touch on the Current State of Big Data
- Drill Down → IMS to Big Data
  - ✔ CDC vs ETL
  - ✔ Streaming
  - ✔ Data / Design Considerations
- Recap Success Factors / Best Practices
- Address Any Questions that You May Have

About Me

- **Scott Quillicy**
  - 35 Years Database Experience
  - Database Software Development
  - Performance & Availability
- **Founded SQData to Provide Customers with:**
  - A Better Way of Replicating Mainframe Data → Particularly IMS
  - Solutions that Combine Consulting Expertise with Technology
  - Technology Built Around Best Practices
- **Specialization**
  - Database Trends and Direction
  - Data Replication
  - IMS to Relational
  - Big Data Streaming
  - Continuous Availability
  - Data Analytics

About SQData



- **Enterprise Class Data Replication**
- **Specialization**
  - High-Performance Changed Data Capture (CDC)
  - Non-Relational Data → IMS, VSAM, Flat Files
  - Relational Databases → DB2, Oracle, SQL Server, etc.
  - Big Data → Hadoop, kafka, etc.
  - Continuous Availability of Critical Applications
  - Data Conversions / Migrations
- **Customer Use Cases**
  - Real-Time Data Streaming to Big Data
  - Continuous Availability → Active-Active, Active-Passive
  - Non-Relational (IMS / VSAM) to Relational
  - ETL (Bulk Data Loads)
  - Event Publishing / Notification
  - Data Warehouse Feeds

Big Data

- **What You May Have Heard...**
  - The “Solution to Everything Analytics”
  - New Concept
  - Adopt or Get Left Behind
- **Reality** → Big Data has been Around for 50+ Years...
- **Characteristics**
  - Significant Amount of Data
  - Advanced Analytics of Disparate Data
  - Many Different Formats → Structured, Semi-Structured, Un-Structured
  - Able to Handle a High Rate of Change
- **Challenges**
  - Increasing Data Volumes → Stress Traditional RDBMS
  - Computing and Infrastructure Costs to Process / Analyze
  - Most Companies Still in Early Stages of Adoption
- **Exciting Times Ahead**
  - Large Open Source Communities
  - Rapid Evolution of Technology

You Have Several Options → More on the Way

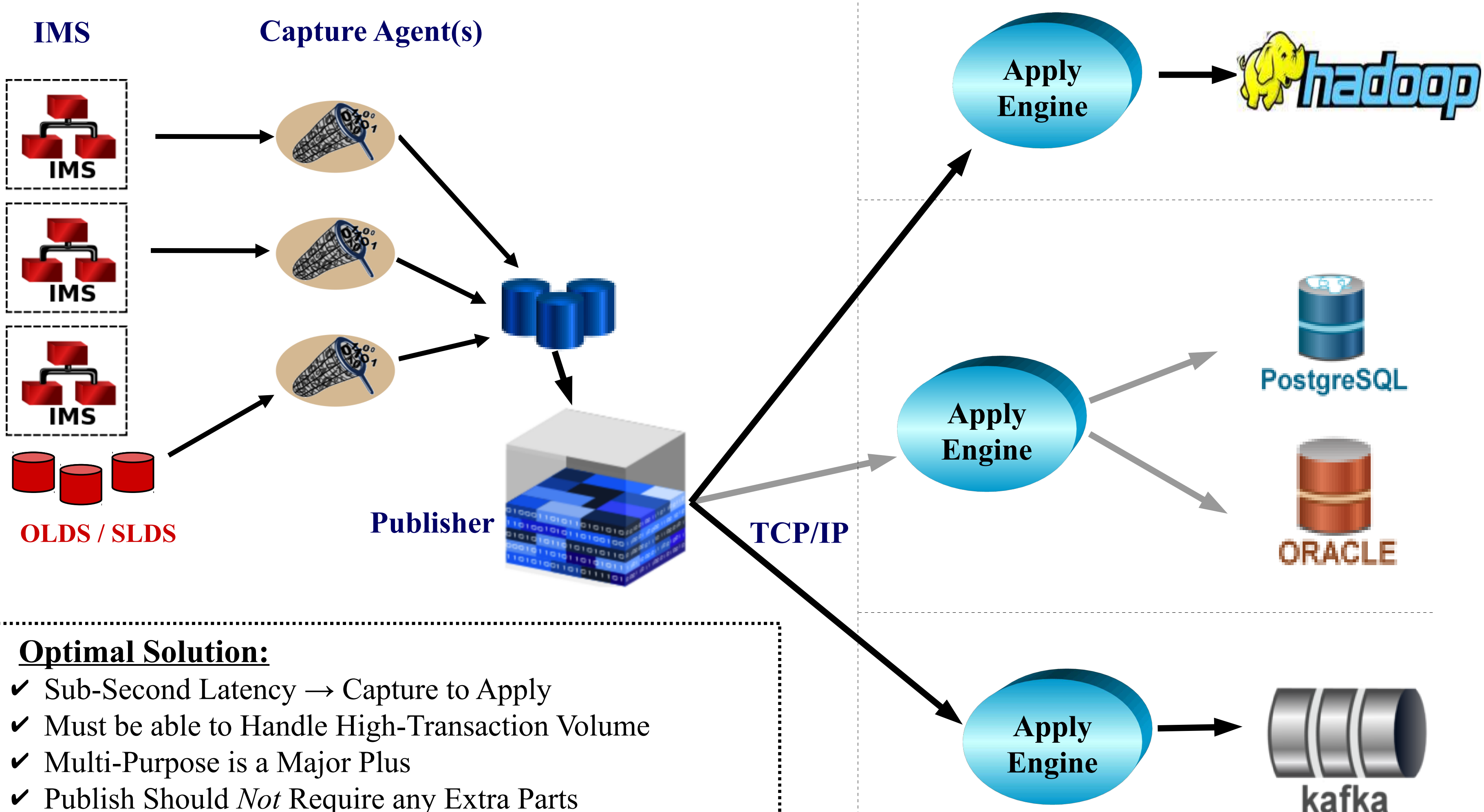
Why Stream IMS to Big Data?

- Real Time 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

Today's Popular Big Data Components

- **Hadoop HDFS**
  - ✔ Most Commonly Used Big Data Store
  - ✔ Foundation Layer for other Technologies such as Spark
  - ✔ Highly Scalable
- **Spark**
  - ✔ High-Performance Processing Engine
  - ✔ Extremely Fast and Versatile → 100x Faster than MapReduce
  - ✔ Runs on HDFS or Standalone
- **Kafka**
  - ✔ Ultra-Fast Message Broker
  - ✔ Streams Data into Most Common Big Data Repositories
  - ✔ Multiple Producers / Consumers
- **Other Popular Stores**
  - ✔ DB2AA / PureData Analytics (Netezza)
  - ✔ Cassandra
  - ✔ Greenplum, Teradata
  - ✔ MongoDB

Streaming IMS to Big Data



<u>Optimal Solution:</u>

- ✔ Sub-Second Latency → Capture to Apply
- ✔ Must be able to Handle High-Transaction Volume
- ✔ Multi-Purpose is a Major Plus
- ✔ Publish Should *Not* Require any Extra Parts
  - No Staging Tables
  - No Queues
- ✔ Must be Resilient / Fault Tolerant

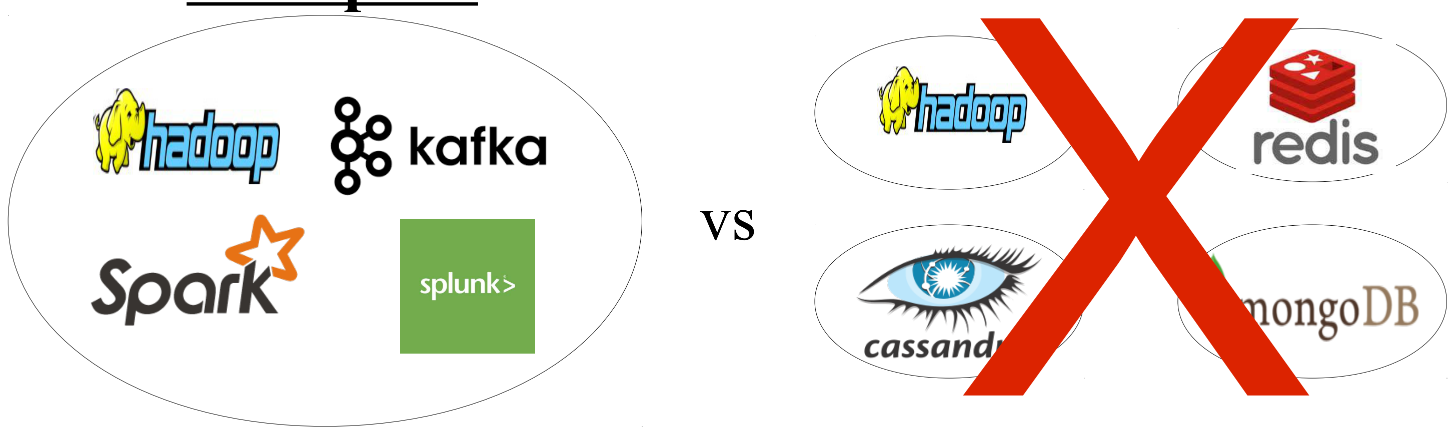
IMS to Big Data → Common Pitfalls

- **Lack of a Holistic Strategy**
  - "We Can Do it Ourselves" Approach
  - Multiple Departments Going into Big Data with Small Projects
  - Minimal Structure → Methods, Tools, Support
  - Significantly More Expensive → Time and $$$
- **Not Focusing on Business Needs**
  - "Build it and They will Come" Approach
  - No ***Clear*** Use Cases
  - Often Caused by Pressure to Deploy a Big Data Solution
- **Data Collection Overkill**
  - "Everything Needs to be in Data Lakes" Approach
  - Minimal Understanding of how to Relate the Data to Business Problems
  - Spend a LOT of Time Moving Data of Little Value to the Business
- **Not Setting Proper Expectations**
  - 'We Can Have Something for You in No Time' Approach
  - Guaranteed Project Timeline and Cost Overruns
- **Understanding Mainframe Data → Particularly IMS**
  - "Just Take the Data and Copy it into Hadoop" Approach
  - Non-Relational Nuances → There are Many...

#1 → Approaching with a Holistic Strategy

- **Key → Deploy on the Enterprise Platform**
  - ✔ Methodology More Mature
  - ✔ Common Technology
  - ✔ Centralized Support Model
  - ✔ Faster Delivery → Despite the “I/T Involvement is Too Much Red Tape”
  - ✔ Reduced Costs

Enterprise **Departmental**



- **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

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

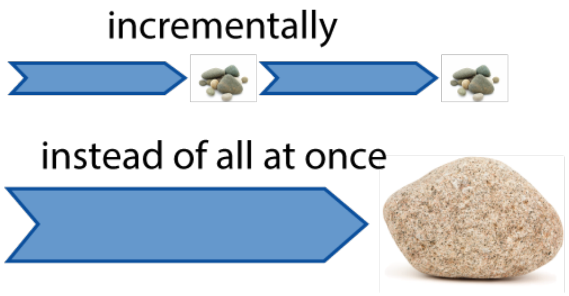
- **Use Case → Financial Information into Big Data**
  - ✔ Tool Selection → MongoDB
  - ✔ Significant Amount of Data (multi-TB)
  - ✔ Grew to 100 Nodes
  - ✔ Project Cost → 1.5 Years and $6M+
  - ✔ Did Not Realize Mongo Does Not Scale Well Until it was Too Late
  - ✔ Result → **Failed** → Not Usable
  - ✔ Next Steps → Trying to Migrate to Hadoop

#2 → Focus on the Business Need

- **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

Source: https://en.wikipedia.org/wiki/DevOps

- **Leverage DevOps**
  - ✔ Data Scientists
  - ✔ Business Analysts
  - ✔ Technical Operations
  - ✔ Quality Assurance
- **Use an Agile Methodology**
  - ✔ Iterative Delivery
  - ✔ Small, Achievable Milestones
  - ✔ Start with Most Important Data
  - ✔ Success Realized Sooner



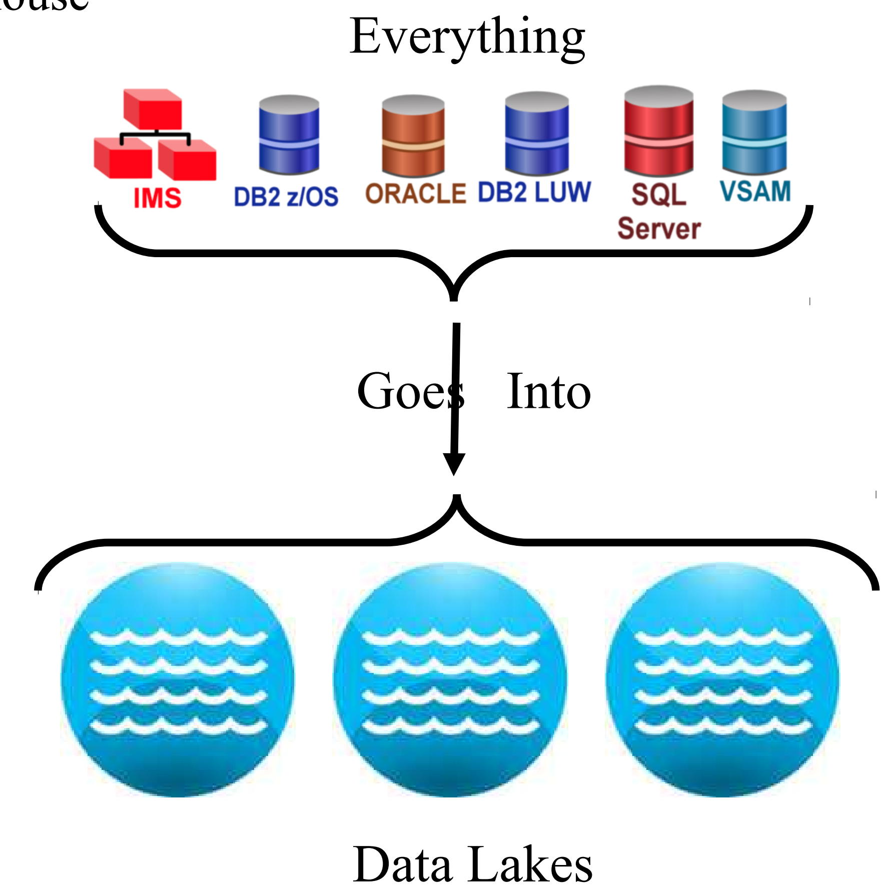
Customer Examples

- **Use Case → Manufacturing Information to Big Data**
  - Tool Selection → HBase
  - Project Cost → 1.5 Years and $7M+
  - Data Dump without Understanding Relationships
  - Result → **Failed** → Not Usable
  - Next Steps → Reworking by Enterprise Group and Business

- **Use Case → Claim Information to Big Data**
  - Tool Selection → MongoDB
  - Project Cost → 2+ Years and $10M+ (est)
  - Data Dump without Understanding Relationships
  - Result → **Failed** → Not Usable
  - Next Steps → Reworking by Enterprise Group and Business into Hadoop

#3 → 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



Customer Example

- **Use Case → Financial Institution**
  - ✔ Tool Selection → Hadoop, kafka, Spark
  - ✔ 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

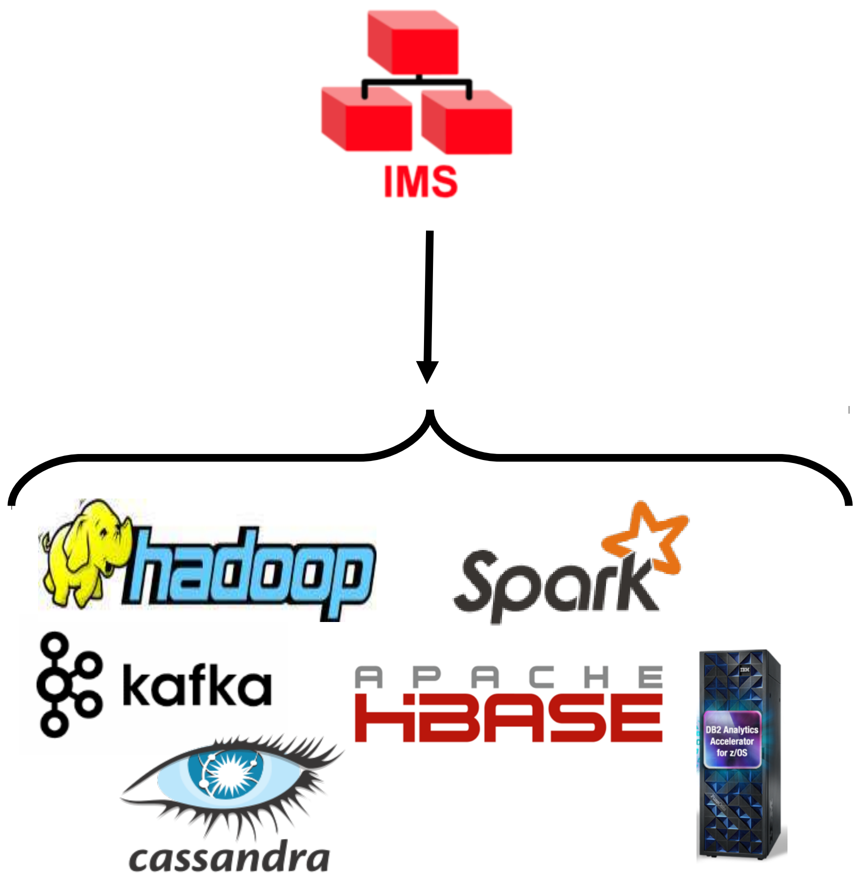
#4 → Not Setting Proper Expectations

- **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

Source: https://gothinkbig.co.uk/

#5 → Not Understanding 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**
- **Is Not Likely to Get Better...**



ACID vs BASE

- **ACID** → Properties Guarantee DB Transactions are Processed Reliably
  - **A**tomicity → All or Nothing...either the Transaction Commits or it Doesn't
  - **C**onsistency → Transaction brings DB from One Valid State to Another
  - **I**solation → Concurrency
  - **D**urability → Once a Transaction Commits, it Remains Committed

- **BASE** → Eventual Consistency
  - **B**asically **A**vailable → Data is There...No Guarantees on Consistency
  - **S**oft State → Data Changing Over Time...May Not Reflect Commit Scope
  - **E**ventual Consistency → Data will *Eventually* become Consistent

More Info: Charles Rowe – Shifting pH of Database Transaction Processing

Source: http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/

Common IMS Data Challenges

- **Code Page Translation**
- **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

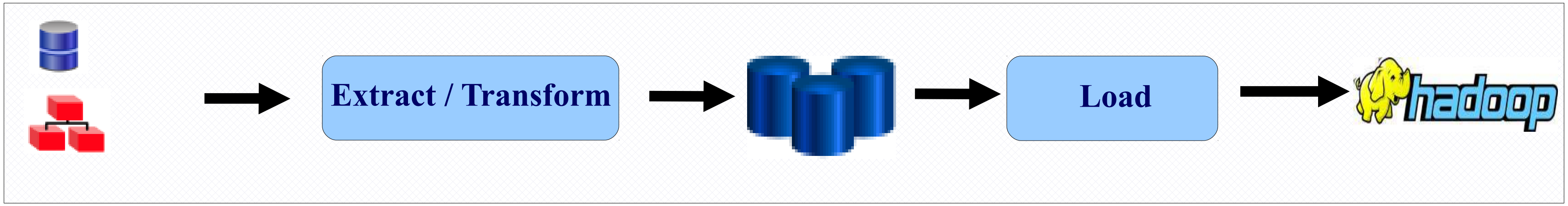
Additional Considerations

- **Data Delivery / Latency**
  - Business Driven
  - Full Extracts → Periodic
  - Near-Real-Time / Scheduled Updates
- **Workload Characteristics**
  - Read vs Update Ratio
  - Update Volume → Transaction Arrival Rate
  - Will Effect Big Data Repository Selection
- **Format**
  - Level of Normalization → Less is Usually Desirable
  - Common Across Multiple Applications / Languages
  - Level of Transformation Required

The Role of ETL and CDC

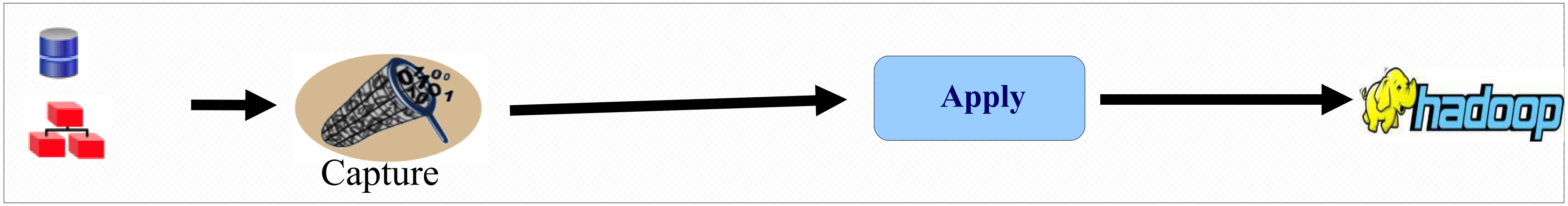
ETL (Extract, Transform, Load):

✓ 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):

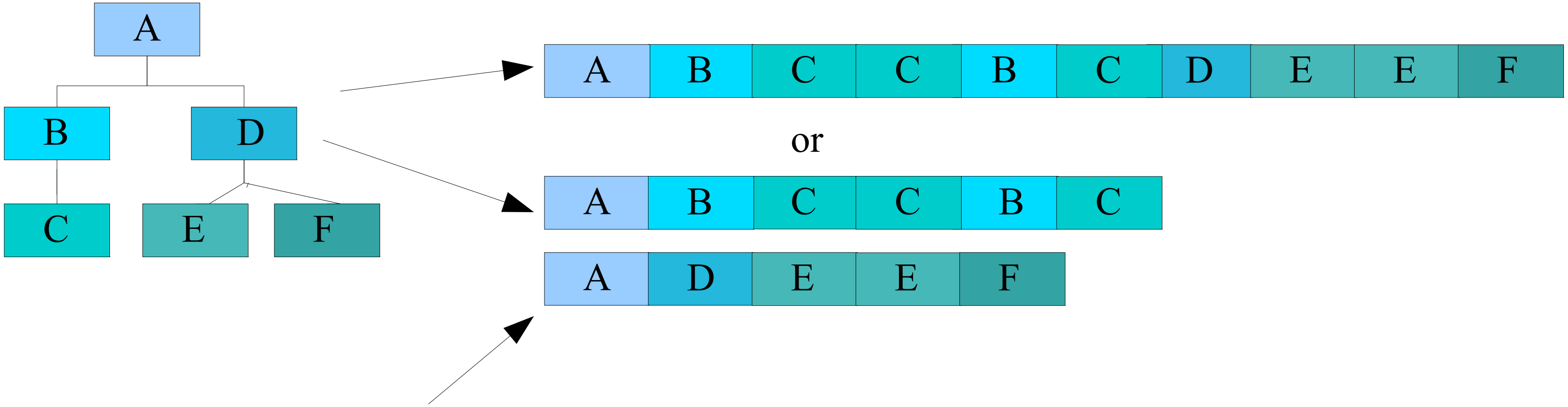
✓ Move Only Data that has Changed
✓ Re-Use Data Transformation Logic from ETL
✓ Near-Real-Time / Deferred Latency
✓ Allows for Time Series Deliver



ETL and Changed Data Capture (CDC)

ETL

- ✔ High Level of Control Over Level of De-Normalization
- ✔ Can Combine Many Segments in Target Row / Document
- ✔ Requires that ETL Tool can Handle Consolidation during Extract



Changed Data Capture

- ✔ May Dictate that Target not Fully Denormalized
- ✔ Capture Along One (1) Branch of IMS DB Record
- ✔ Path / Lookups *may* be Required

Target Apply Concepts

- **Frequency**
  - ✔ Near-Real-Time
    - Continuous Stream
    - Low Latency → Typically Sub-Second, but May be a Bit Higher for Larger Transactions
  - ✔ Batches
    - Triggered by # Records and/or Time Interval
    - Time Based
    - Latency Varies
- **Time Series**
  - ✔ Analyze Data Changes Over Time
  - ✔ All CDC Data is Inserted into Target
  - ✔ timeuuid type Key
- **Incremental Updates (Synchronized)**
  - ✔ Source Matches Target
  - ✔ Requires Query Adjustments for Insert-Only Targets (i.e. Hadoop HDFS)
    - Get Latest Image of Record by Key(s)
    - Filter Out Deletes
    - Merge into 'Master' File on Periodic Basis

CDC / ETL Data Format(s)

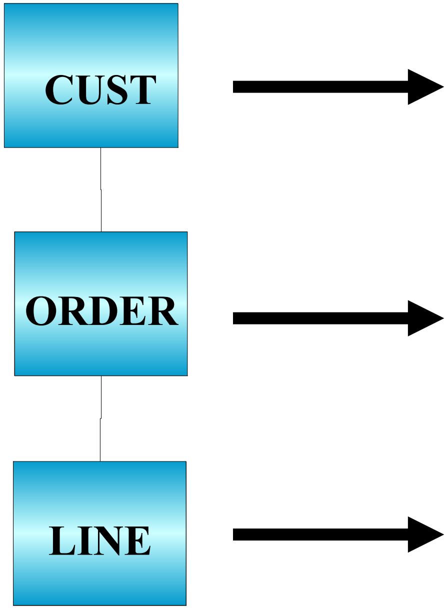
- **Common Formats → JSON, Avro, Delimited, XML, Relational**
- **JSON Recommended for CDC/ETL Data**
  - ✔ Especially for Data Lakes
  - ✔ Records are Self-Described → Encapsulated Metadata
  - ✔ Payload Lighter than XML

Sample Update CDC Record in JSON Format


```
{"DEPT": {
  "database": "IMSDB01",
  "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"
  }
}}
```

Design → Traditional IMS 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 (DB2AA, Netezza, Teradata, etc.)



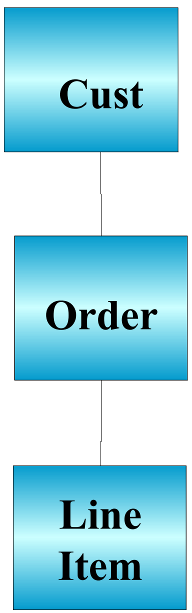
Key	Data
CUST #	

Key	Key	Data
CUST #	ORD #	

Key	Key	Key	Data
CUST #	ORD #	LINE #	

Design → IMS to Big Data

- De- Normalized / Minimal Normalization
- Still Requires Transformation (dates, binary values, etc.)
- Good News → IMS Structure Already Setup for Big Data



Key	Data
Cust#	

Key	Data	Data	Data	Data	Data	Data
Order#	Cust#		Line #		Line#	

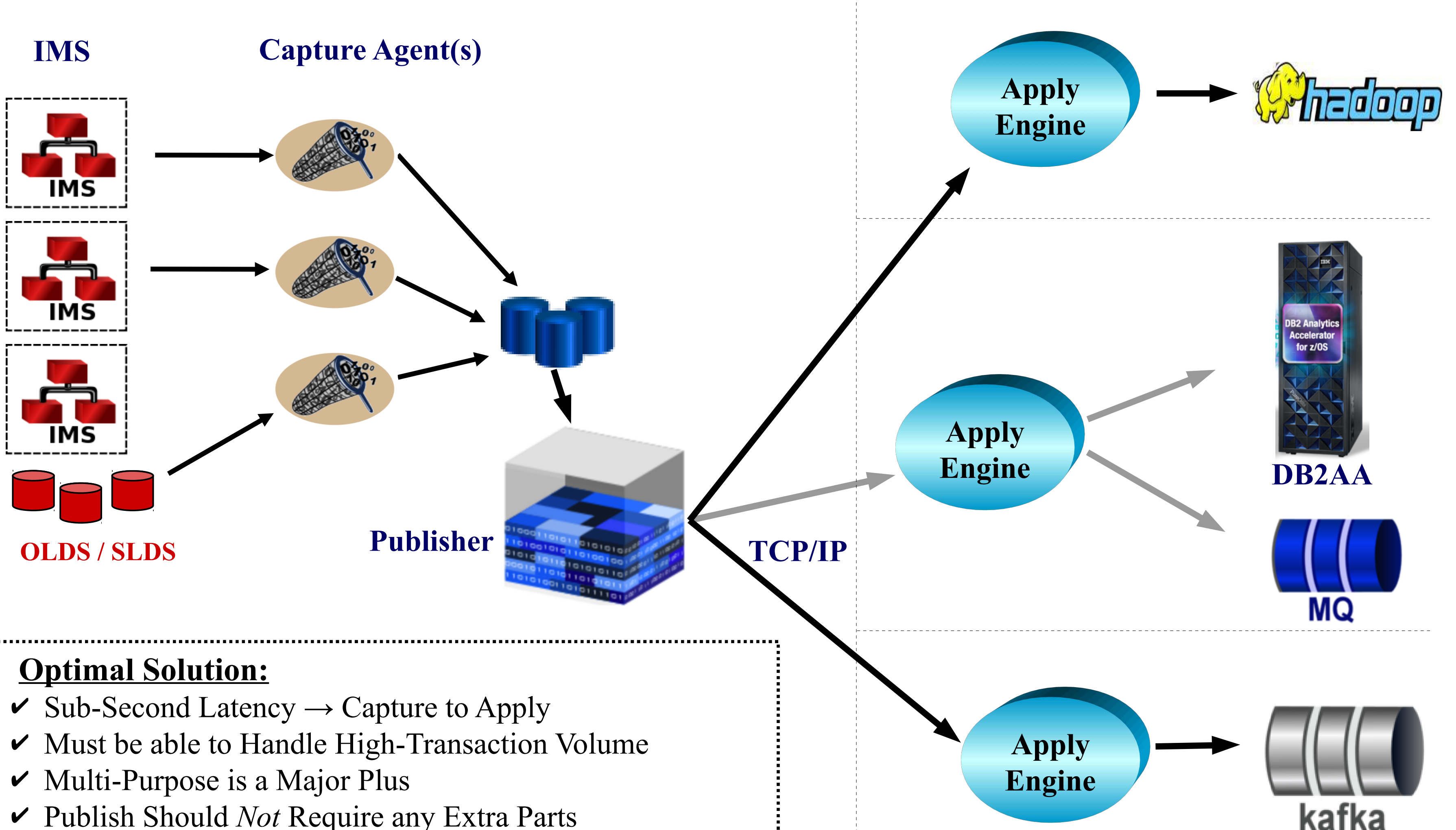

```
{ "company_name" : "Acme",
  "cust_no"      : "20223",
  "contact" :{ "name" : "Jane Smith",
              "address" : "123 Maple Street",
              "city" : "Pretendville",
              "state" : "NY",
              "zip"   : "12345" }
}
```

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

IMS Data Capture Methods

- **Primary Methods of Capture**
  - ✓ Data Capture Exit Routines
  - ✓ Log Based
- **Database Capture Exit Routines**
  - ✓ Near-Real-Time for IMS TM/DB
  - ✓ Extremely Fast and Efficient
  - ✓ Scalability → Capture / Apply by FP Area, HALDB Partition, PSB, Database
  - ✓ Do Not Require x'99' Log Records → No Impact to IMS Logging
- **Log Based**
  - ✓ Near-Real-Time or Asynchronous
  - ✓ CICS / DBCTL Environments
  - ✓ Requires x'99' Log Records
  - ✓ Scalability → Same as Database Exit Routines

IMS Streaming

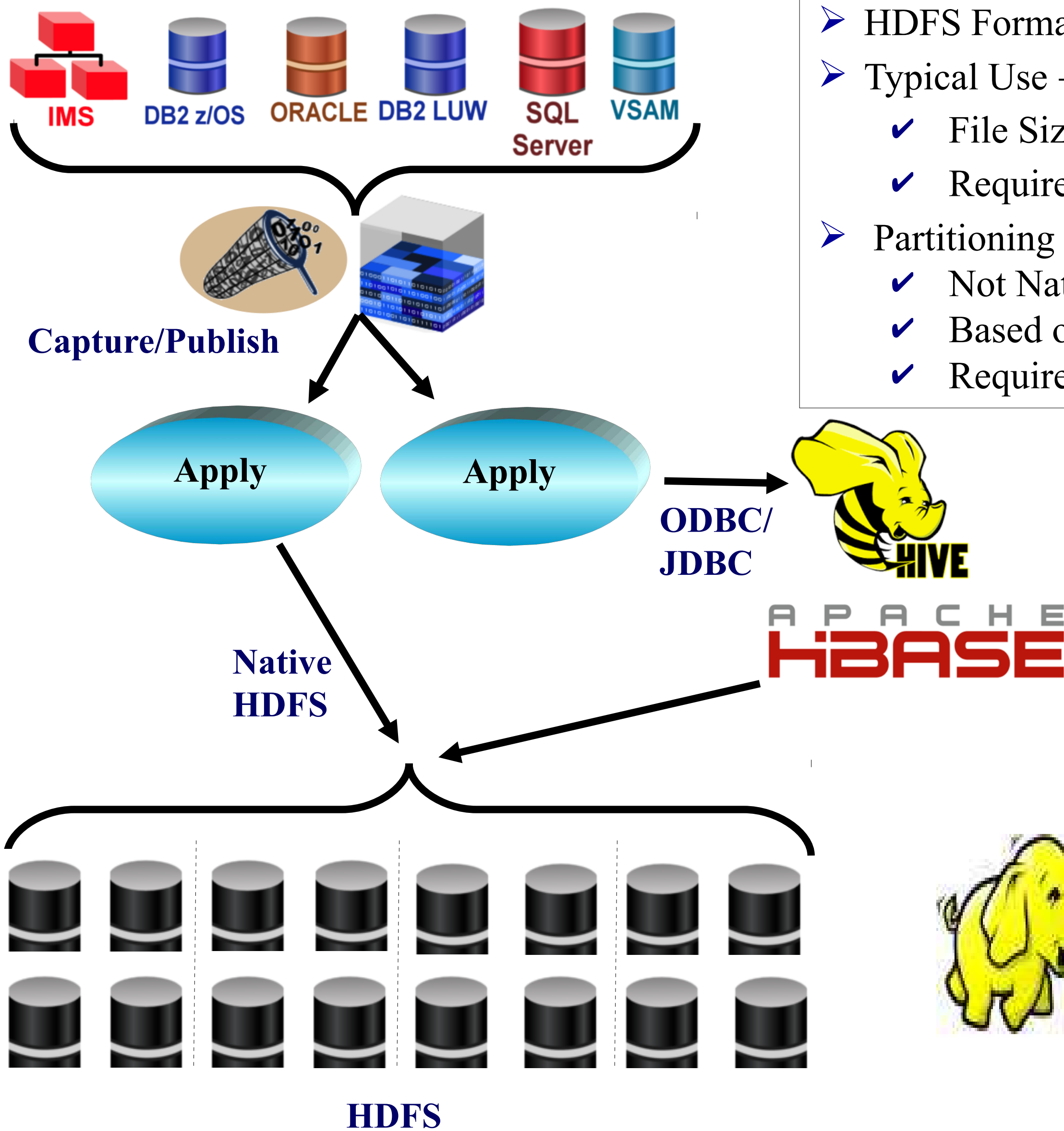


<u>Optimal Solution:</u>

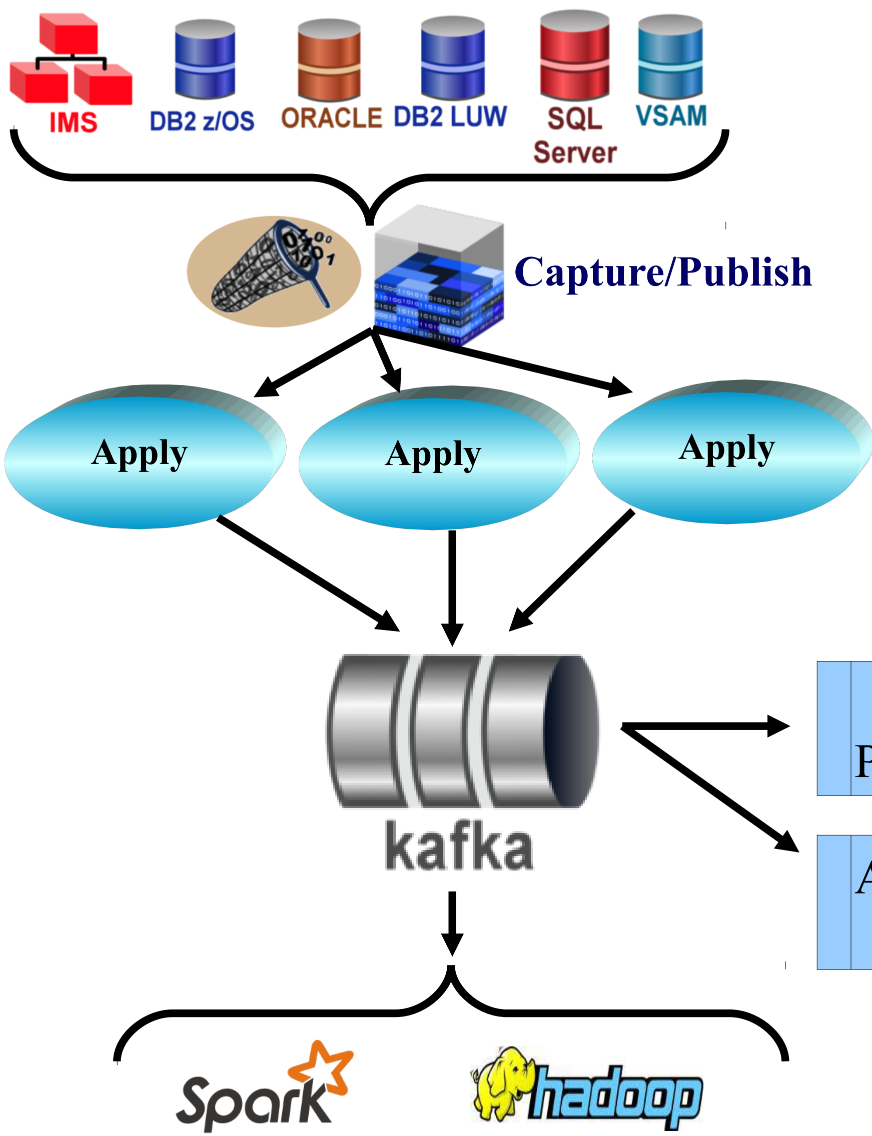
- ✔ Sub-Second Latency → Capture to Apply
- ✔ Must be able to Handle High-Transaction Volume
- ✔ Multi-Purpose is a Major Plus
- ✔ Publish Should *Not* Require any Extra Parts
  - No Staging Tables
  - No Queues
- ✔ Must be Resilient / Fault Tolerant

Streaming to Hadoop

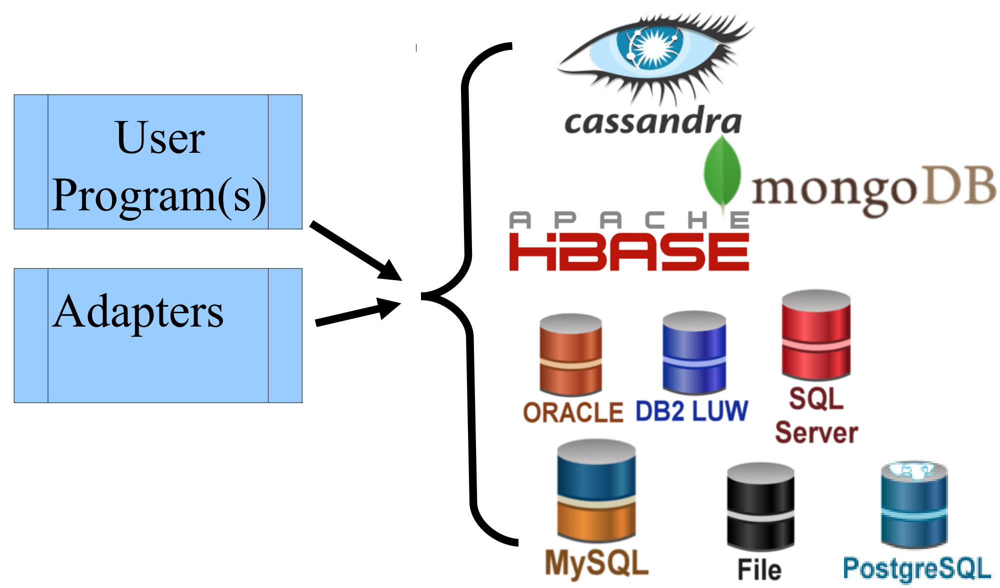
- HDFS Format → JSON, XML, Delimited, Custom
- Typical Use → Multiple Files for Same Content
  - ✔ File Size Based on # Records / Time Interval
  - ✔ Requires Multi-File Management
- Partitioning → Based on Source Value(s)
  - ✔ Not Native in HDFS
  - ✔ Based on Source Data Value(s)
  - ✔ Requires Cross-Partition Multi-File Management



Streaming to 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 Message Format for CDC
- Use Cases:
  - ✔ Basic Messaging → Similar to MQ
  - ✔ Website Activity Tracking
  - ✔ Metrics Collection / Monitoring
  - ✔ Log Aggregation
  - ✔ Streaming



Success Factor Summary → Best Practices

- **Approach with a Holistic Strategy**
  - ✔ Common Infrastructure / Tools / Support
  - ✔ Established Methods (DevOps / Agile)
  - ✔ Beware the "Fiefdoms"
- **Involve the Business from the Start**
  - ✔ 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 the IMS Data is 'Special'**
  - ✔ Patience is Key
  - ✔ Do Not Hesitate to Ask for Help...

Thank You!!

IMS to Big Data

Common Traits for Success

Prepared for the:

Virtual IMS User Group

4 October 2016