Lessons Learned 2022

IMS Application Modernization to the Cloud

Virtual IMS User Group

07 February 2023

Topics

- Introduction
- Recap of Last Year’s Lessons
- Application Modernization from a Data Perspective
- Current Trends and Directions
- Best Practices Tips and Hints

So…Who’s This Guy?

- Scott Quillicy
  - Officially a 40 Year Mainframer
  - Database Tools Software Development
  - 30+ Years in the Data Replication (CDC) Arena
- Thrown to the Wolves Right Out-of-the-Gate
  - Thank the Assembler Classes for That…
  - IMS 1.2, planning to migrate to 1.3 (DBRC intro)
  - Db2z 1.2 in 1984
- Founded SQData in 2000
  - Right about the Time that the Internet Bubble Burst
  - Address Shortcomings with Mainframe Data Replication
  - IMS Specialization
  - Acquired by Syncsort (now Precisely) in 2019....Been a Good Run Since… ☺
- Spend Most of My Time Working with Large Mainframe to Cloud Data Migration Projects
  - Keep people from making common mistakes
  - Help implement best practices

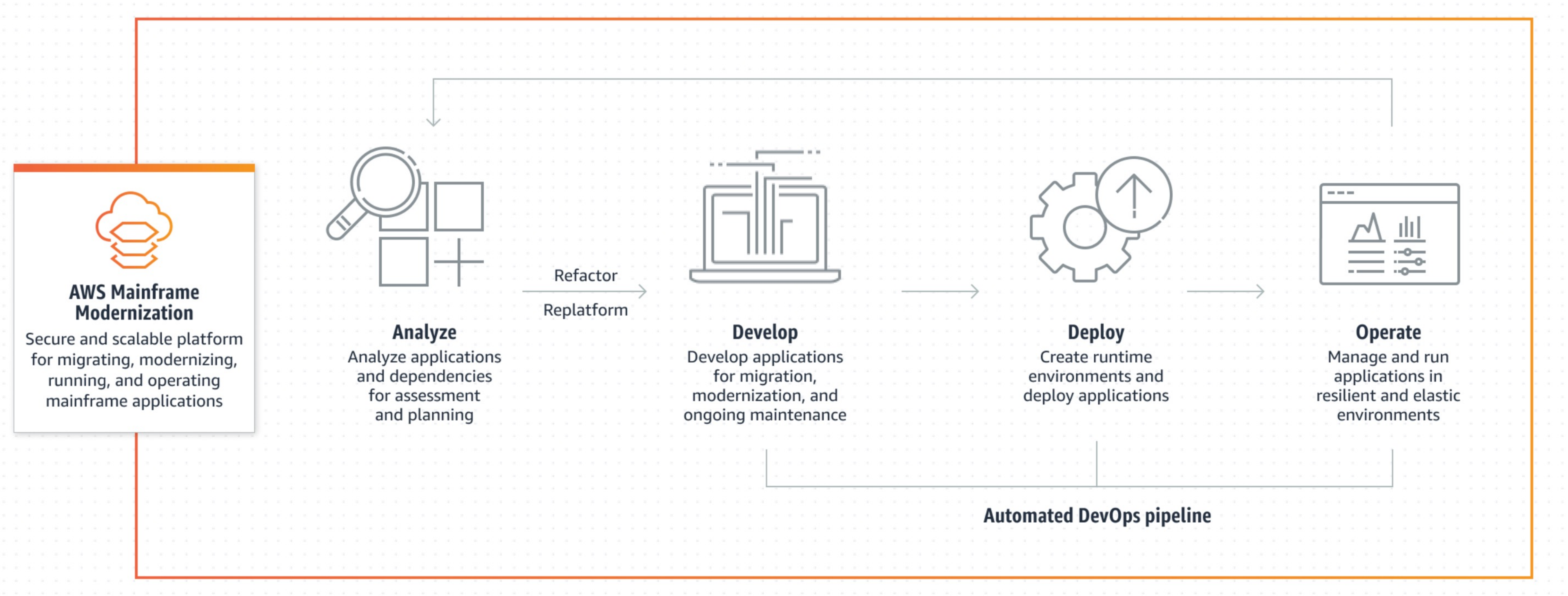
Terminology

Source	Data stream start point
Target	Data stream end point
Streaming	Replication of IMS updates off of the mainframe
Tokenization	Masking sensitive information (PII)
Forward Sync	Streaming data from mainframe sources to distributed/cloud targets
Reverse Sync	Streaming Data from Distributed/Cloud Sources to Mainframe Targets
Telemetry	Operational telemetry and monitoring

What Does Mainframe Modernization Really Mean?

- Migrate Workload from the Mainframe to the Cloud
  - Major Core Applications: IMS, Db2z and VSAM
  - Very High Transaction Volume: 1B+ updates/day
- Application Strategies
  - Replatform: “Lift and Shift” – run IMS apps…’unchanged’… on linux using a framework like Microfocus for IMS
  - Refactor: converts legacy COBOL/PL1 to a cloud friendly language such as Java
  - Redevelop: complete modernize applications
- Data
  - Bulk data transfer and data replication are *critical* components
  - Super Low latency requirements
  - Two (2) way synchronization: mainframe to cloud and back
- ***Our Focus for this Discussion will be on the IMS Data Replication Aspect of Modernization***

Mainframe Modernization



AWS Mainframe Modernization

Secure and scalable platform for migrating, modernizing, running, and operating mainframe applications

Analyze

Analyze applications and dependencies for assessment and planning

Refactor

Replatform

Develop

Develop applications for migration, modernization, and ongoing maintenance

Deploy

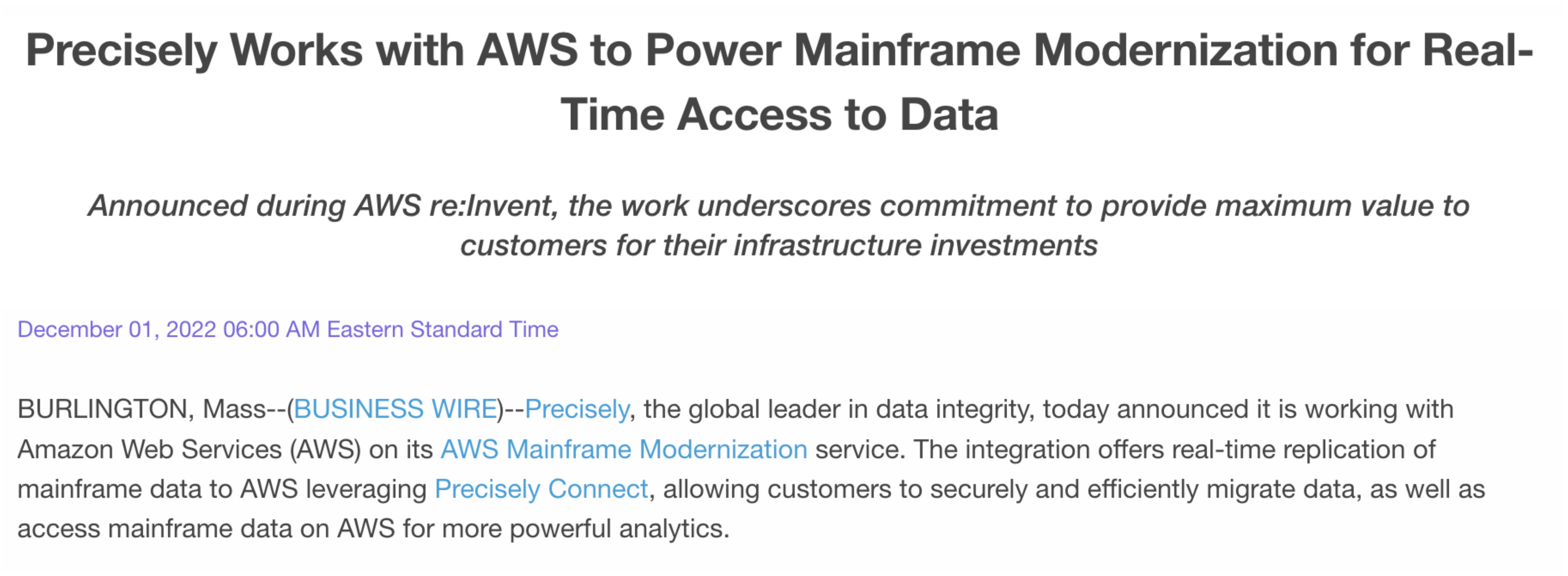
Create runtime environments and deploy applications

Operate

Manage and run applications in resilient and elastic environments

Automated DevOps pipeline

Source: https://aws.amazon.com/mainframe-modernization/



Precisely Works with AWS to Power Mainframe Modernization for Real-Time Access to Data

Announced during AWS re:Invent, the work underscores commitment to provide maximum value to customers for their infrastructure investments

December 01, 2022 06:00 AM Eastern Standard Time

BURLINGTON, Mass--(BUSINESS WIRE)--Precisely, the global leader in data integrity, today announced it is working with Amazon Web Services (AWS) on its AWS Mainframe Modernization service. The integration offers real-time replication of mainframe data to AWS leveraging Precisely Connect, allowing customers to securely and efficiently migrate data, as well as access mainframe data on AWS for more powerful analytics.

Source: https://www.businesswire.com/news/home/20221201005545/en/Precisely-Works-with-AWS-to-Power-Mainframe-Modernization-for-Real-Time-Access-to-Data

Cloud Advantages & Disadvantages

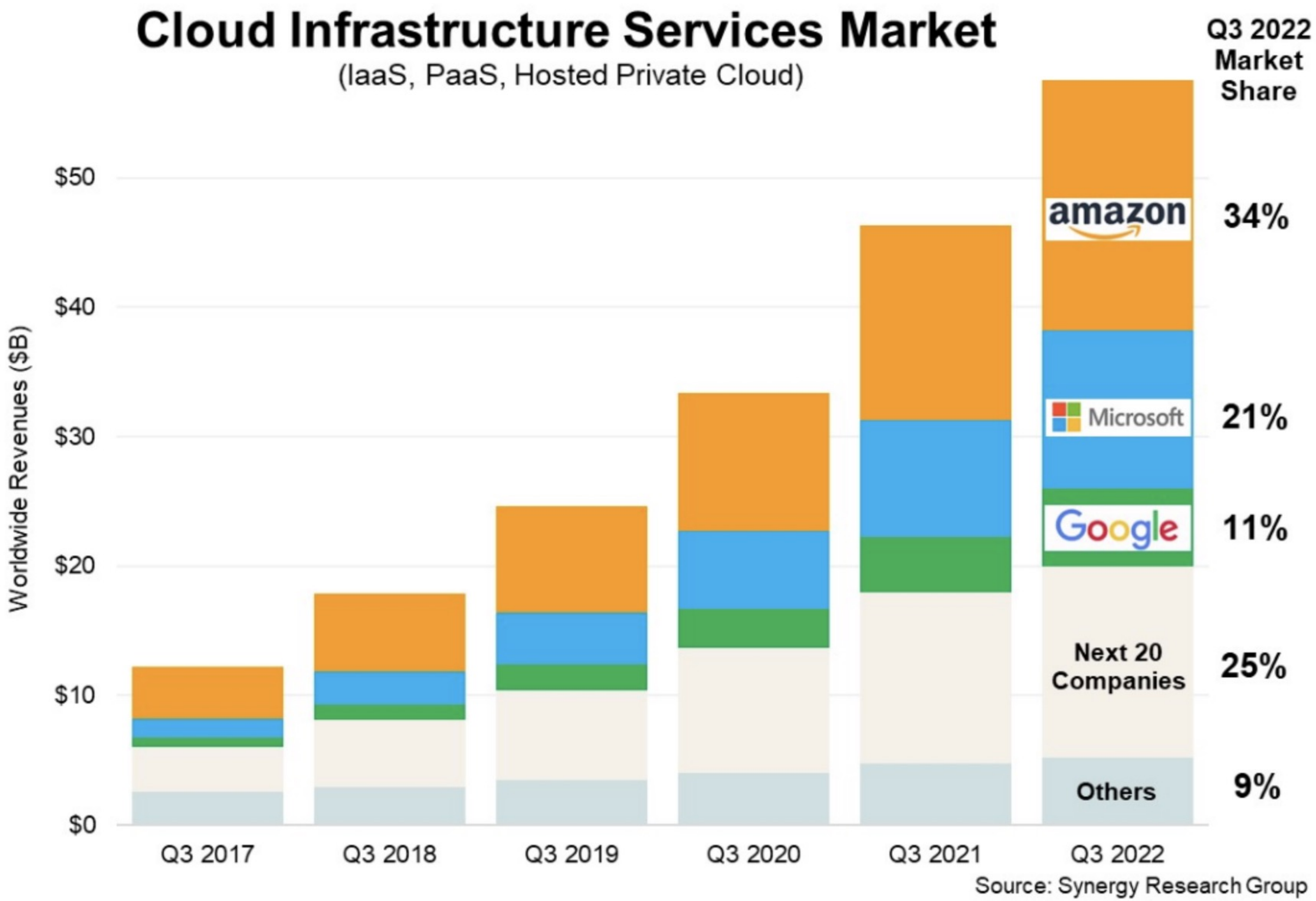
Advantages

- Cost Reduction
  - Physical Space
  - Hardware
  - Costly High-End Software
  - Support Personnel
- Tools
  - Wide Variety of Access to Latest Technology
  - Common framework across app infrastructure
  - ‘Shelfware’ Avoidance – use only what you need
- Security
  - Infrastructure
  - Must Comply with Industry Standards
  - Less Prone to Employee Theft
- Reliability
  - Built-In Redundancy
  - Most Providers Guarantee 99.99% Uptime
- Technical Skills Plentiful

Disadvantages

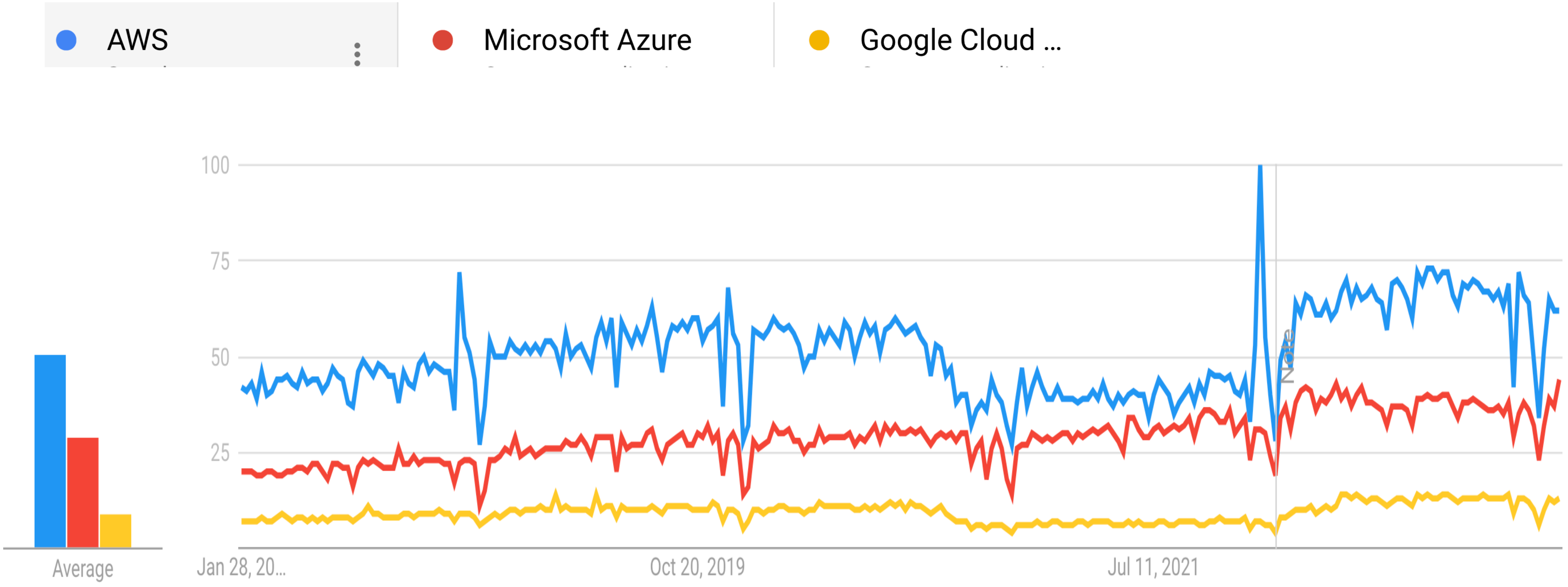
- Cost Creep
  - Compute Resources
  - Departmental Use/Abuse
- Downtime
  - Communication Failure
  - Internet Drops
- Security
  - Does not Guard Against Weak Digital Security Methods
  - Can be Like Leaving Your Laptop Open at Starbucks
- Performance
  - Traditional Batch Workloads
  - Complex Transactions
- Cloud Vendor Lock
  - Vendors Highly Encourage ‘Their Stuff’
  - Difficult to Move Off, Once You are There

Market Share by Cloud Provider



Source: Synergy Research Group

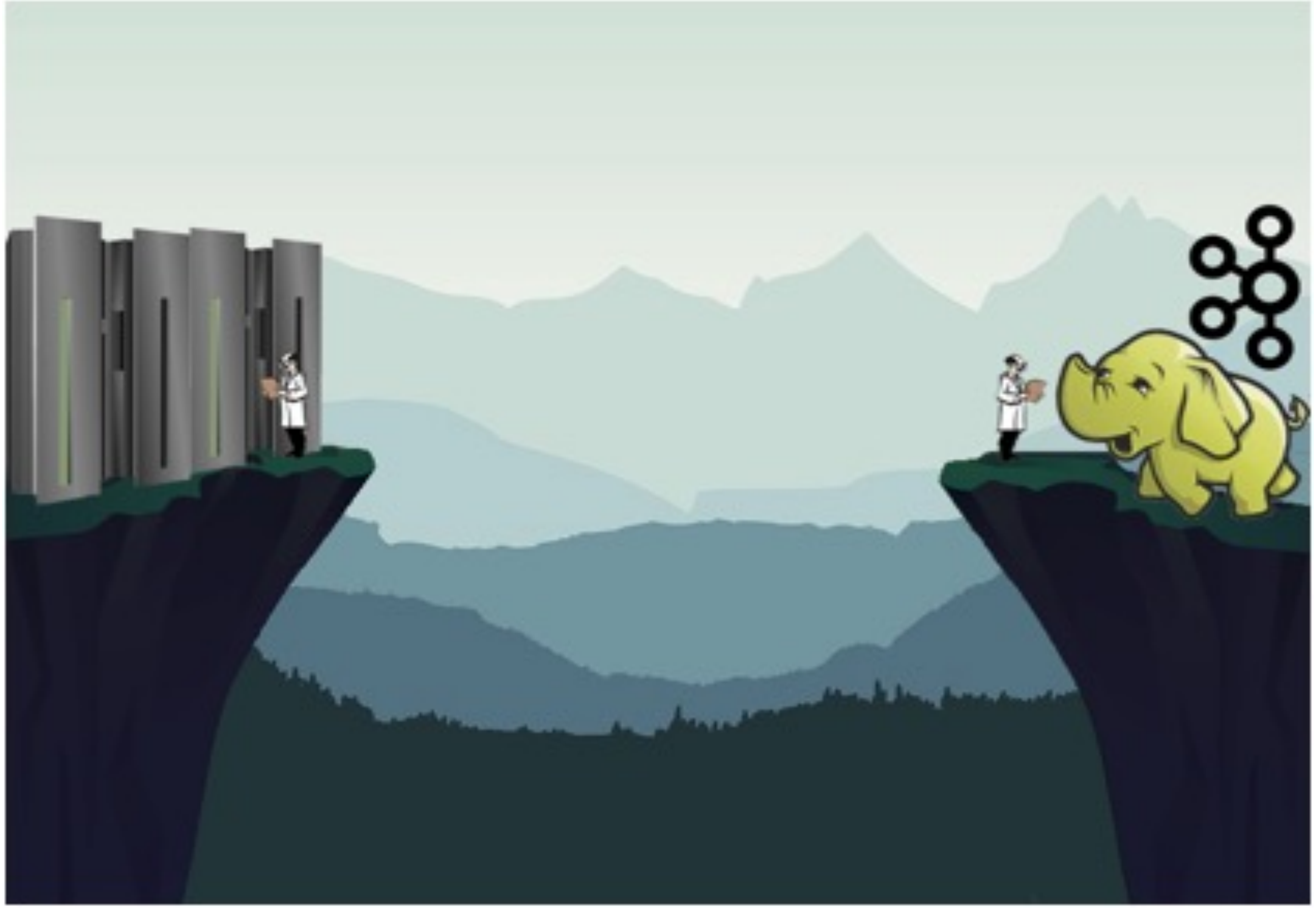
AWS vs Azure vs GCP: Interest Over Last 5 Years



Scenes from Last Year’s Episode…

The ‘Great Divide’ is Real

- Mainframe vs Distributed
- Significant Language Barrier
- Different Levels of Discipline
- Pride of Ownership (both sides)
- Shrinking Talent Pool
  - IMS
  - Mainframe in General
- Requires Patience
  - Communication
  - Collaboration



Lessons Learned

- Reality Checks
  - The ‘Great Divide’ is a real thing
  - Mainframe expertise is in short supply and IMS skills are rapidly becoming scarce
- Trends
  - Cloud is King - on-premise targets represent < 10% of deployments in 2020-21
  - Application Modernization is ***Super Hot***
  - ‘Moving off the mainframe’ has gained ***Serious*** momentum
- Technical
  - Cloud databases can be painfully slow for batch type workloads
  - Scaling is often an afterthought
  - Bi-directional replication (cloud to IMS) is a becoming a more common use case

Databases in the Cloud

- Highly Scalable for ‘Normal’ Transactions: Quick In / Quick Out
- ***Much*** Slower than Traditional On-Premise
  - Ping time is the major factor
  - Can be 10x slower than on-prem
  - Very important to consider for app modernization – batch window will increase
- Security Concerns are Much Higher
  - Increased data tokenization
  - Serious resistance to connecting mainframe to cloud
- High Volume Customers Need to Include a Streaming Component
  - Allows for a high degree of scaling to relational (and other) targets
  - Must be implemented carefully – Don’t just ‘slap it In’

Key Considerations

- Performance / Throughput Depends on the Speed of the Target
- Application Modernization Requires Lower Latencies
- Higher Volume Customers will Need to Scale
  - Peak IMS Transaction Arrival Rate (CDC records/second) – Usually Batch
  - Compare Against Speed of Target for a Single Process/Thread
    - Relational DB on-prem: 2K rows/second
    - Relational DB in cloud: 250 - 500 rows/second, depending on ping time
    - Streaming platform: Over 100K+ messages/second
  - Goal: target keeps up with source CDC record arrival rate with acceptable latency
- Slow Apply Rate Means
  - Increased Mainframe Back-Pressure
  - Higher CPU Usage
  - Unhappy Business Users

One Year Later…

The ‘Great Divide’ is Getting Out-of-Hand

- Mainframe vs **Cloud**/Distributed
- **Increased** Language Barrier
- Different Levels of Discipline
- Pride of Ownership (both sides)
- **Talent Pool Disappearing**
  - IMS
  - COBOL/PL1
  - Mainframe in General
- Requires ***Extreme*** Patience
  - Communication
  - Collaboration

Lessons Learned

- Reality Checks
  - The 'Great Divide' is a real **threat**
  - Mainframe expertise is in short supply and IMS skills are rapidly becoming scarce
- Trends
  - Cloud is King - on-premise targets represent < 10% of deployments in 2020-21
  - Application Modernization is ***Super Hot***
  - 'Moving off the mainframe' is the goal of many of the larger companies
- Technical
  - Cloud databases can be painfully slow for batch type workloads
  - Scaling is often an afterthought
  - Bi-directional replication (cloud to IMS) is a becoming a **requirement**

Databases in the Cloud

- Highly Scalable for ‘Normal’ Transactions: Quick In / Quick Out
- ***Much*** Slower than Traditional On-Premise
  - Ping time is the major factor
  - Can be 10x slower than on-prem
  - Very important to consider for app modernization
  - **Batch workload migration is a major source of concern**
- Security Concerns are Much Higher
  - Increased data tokenization
  - Serious resistance to connecting mainframe to cloud
- Higher Volume Customers **Require** a Streaming Component (i.e. Kafka, Event Hub)
  - Allows for a high degree of scaling to relational (and other) targets
  - Must be implemented carefully – Don’t just ‘slap it In’

Key Considerations

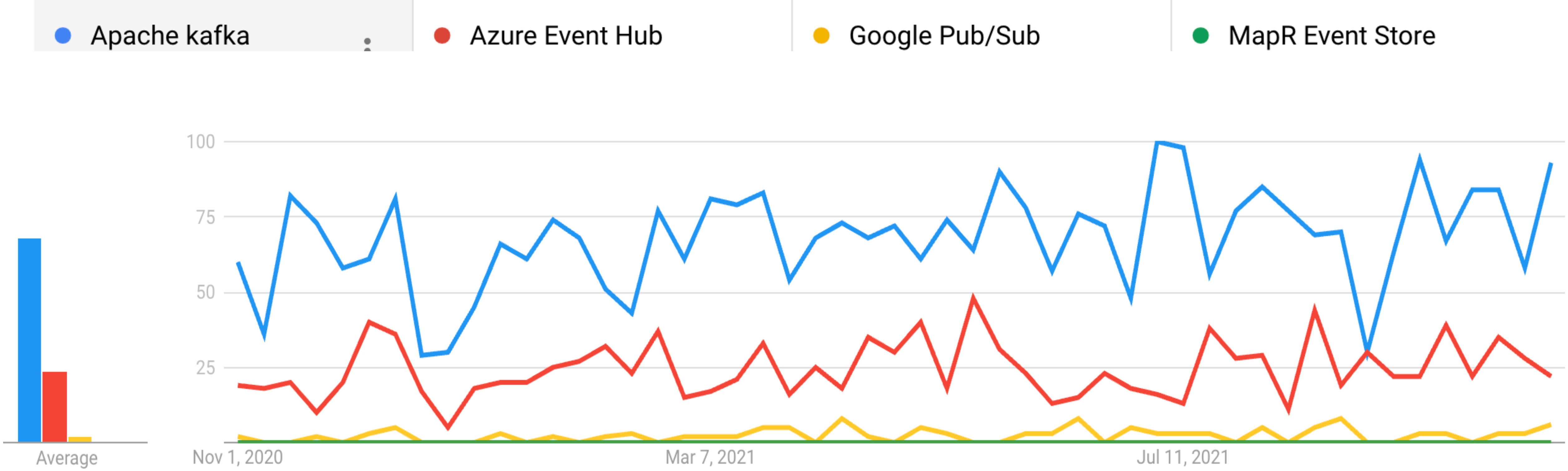
- Performance / Throughput Depends on the Speed of the Target
- Application Modernization **Demands** Low Latencies
- Moderate and Higher Volume Customers will Need to Scale
  - Peak IMS Transaction Arrival Rate (updates/second)
  - Batch Workload Presents a Challenge
  - Compare Against Speed of Target for a Single Process/Thread
    - Relational DB on-prem: 2K rows/second
    - Relational DB in cloud: 250 - 500 rows/second, depending on ping time
    - Streaming platform: Over 100K+ messages/second
  - Goal: target keeps up with source CDC record arrival rate with acceptable latency
- Slow Apply Rate Means
  - Increased Mainframe Back-Pressure
  - Higher CPU Usage
  - **Service Level Violations**

Trends and Directions

Helpful Tips

- Set Realistic Expectations: Latency and Throughput
  - Depends on transaction arrival rate, scaling factor and infrastructure capacity
  - Best to lower expectations (i.e. sub-second) vs unreasonable commitments
  - Batch transactions/uows *will not* likely be sub-second
  - Results may vary based on your environment
- The Speed of the Target Generally Dictates Latency & Throughput
  - Cloud can be significantly slower than on-premise
  - Network bandwidth becomes a factor
  - Kafka or other fast streaming component *highly* recommended
    - Allows for target side scaling
    - Multiple consumer groups for a single source
- The Final Solution will Consist of:
  - Software and APIs from multiple vendors, even competing offerings
  - A streaming component…if moderate/high volume with low latency

Streaming Platform Popularity



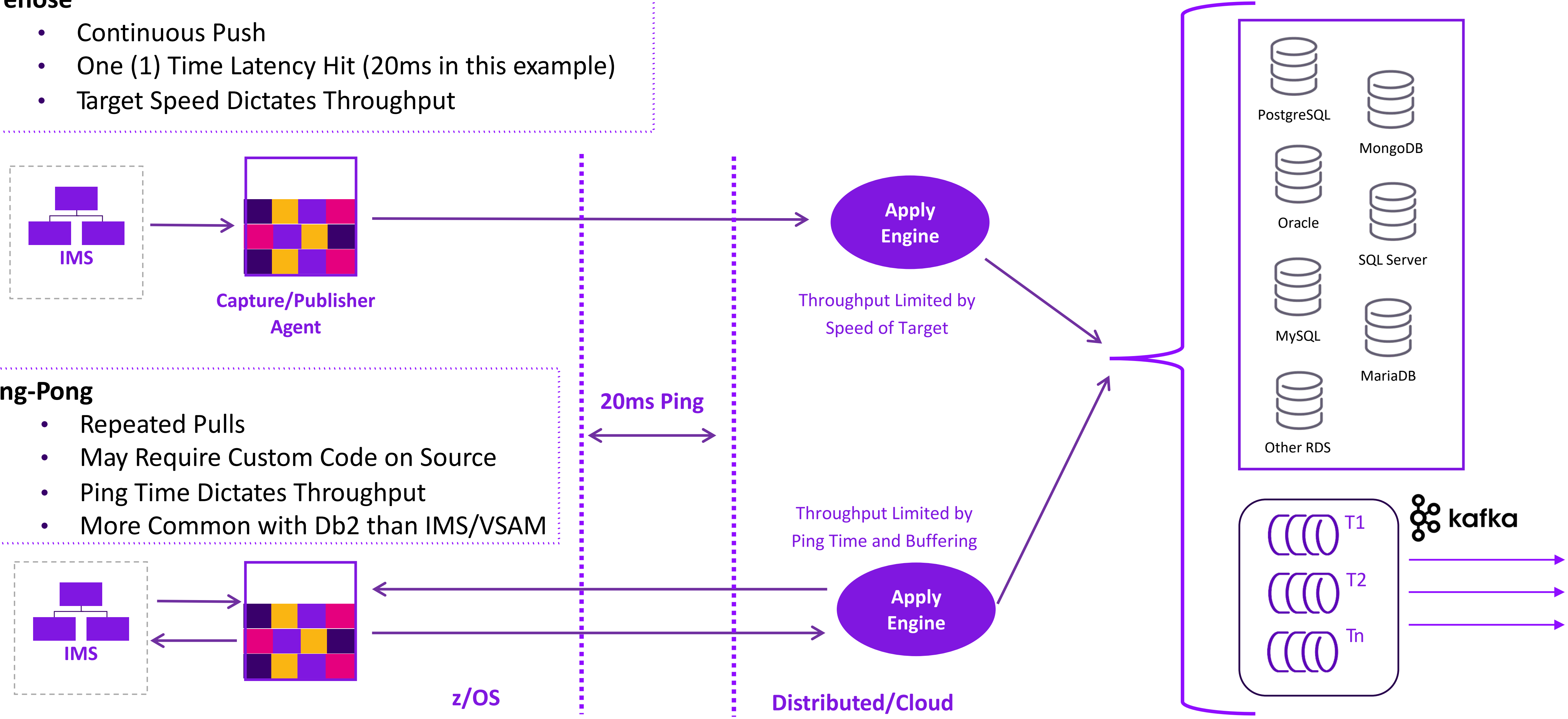
A Tale of Two Architectures: Forward Sync

Firehose

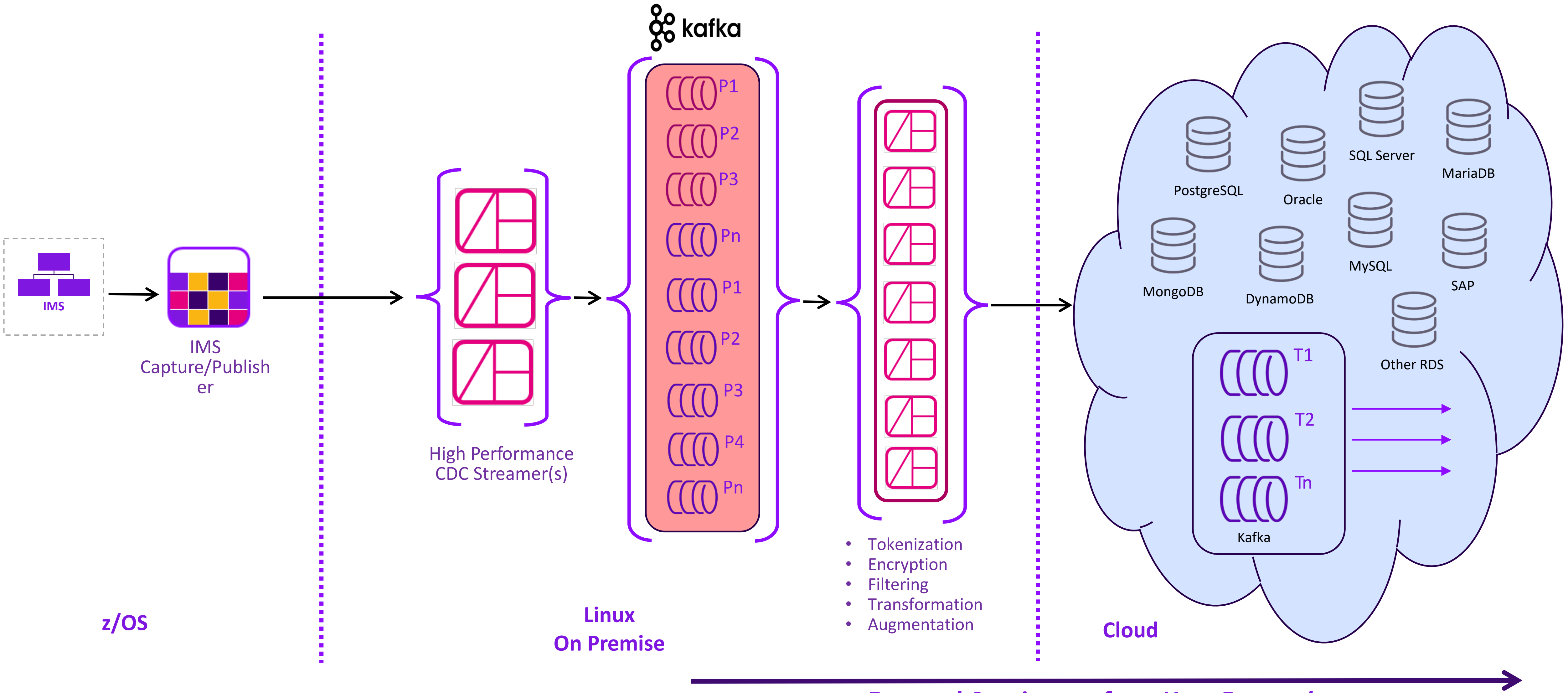
- Continuous Push
- One (1) Time Latency Hit (20ms in this example)
- Target Speed Dictates Throughput

Ping-Pong

- Repeated Pulls
- May Require Custom Code on Source
- Ping Time Dictates Throughput
- More Common with Db2 than IMS/VSAM



Popular Forward Synchronization Architecture



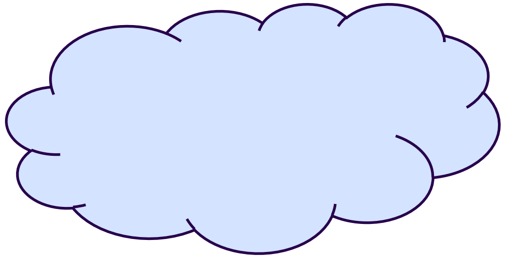
- Tokenization
- Encryption
- Filtering
- Transformation
- Augmentation

Eventual Consistency from Here Forward

More on Eventual Consistency

- Data will ***Eventually*** be in Synch
- Data Arrival
  - Can (and will) arrive out-of-order across topic partitions
  - Order maintained within a single partition
  - Be consistent with the partitioning key
- Some Folks Try Using ‘Commit Events’ to Consolidate Transactions
  - Complicates things
    - Doesn’t always work if transaction spread across multiple partitions
    - Commit record can arrive before all transaction data has been written
  - Slows the data flow
- Best Practice
  - Process data by physical key, making sure key used for topic partitioning
  - Combine transaction data *before* writing to streaming target
  - Carry source update timestamps to the target
    - Keeps things in order
    - Avoids overlaying newer data with older data

Reverse Synchronization



- Bi-Directional → Cloud to Mainframe
- Keep Legacy DBs in Synch with Updates from Cloud
- Interesting Trend
  - Initial planning: absolute requirement
  - Mid implementation: *maybe* for some data
  - Late implementation: only if required…minimal data
- Rules
  1. One ***and only one*** system of record (SOR)
  2. SOR **cannot** be shared across platforms (see #1)
  3. Collision detection required, in case #1 and #2 are violated
  4. Data context must be complete
     - Legacy applications may not function without enough data
     - Ask…"what happens on an insert"?
- Partial Updates will Slow Things Down by at Least ½
- Stay Tuned….We'll Keep You Posted on this One… ☺

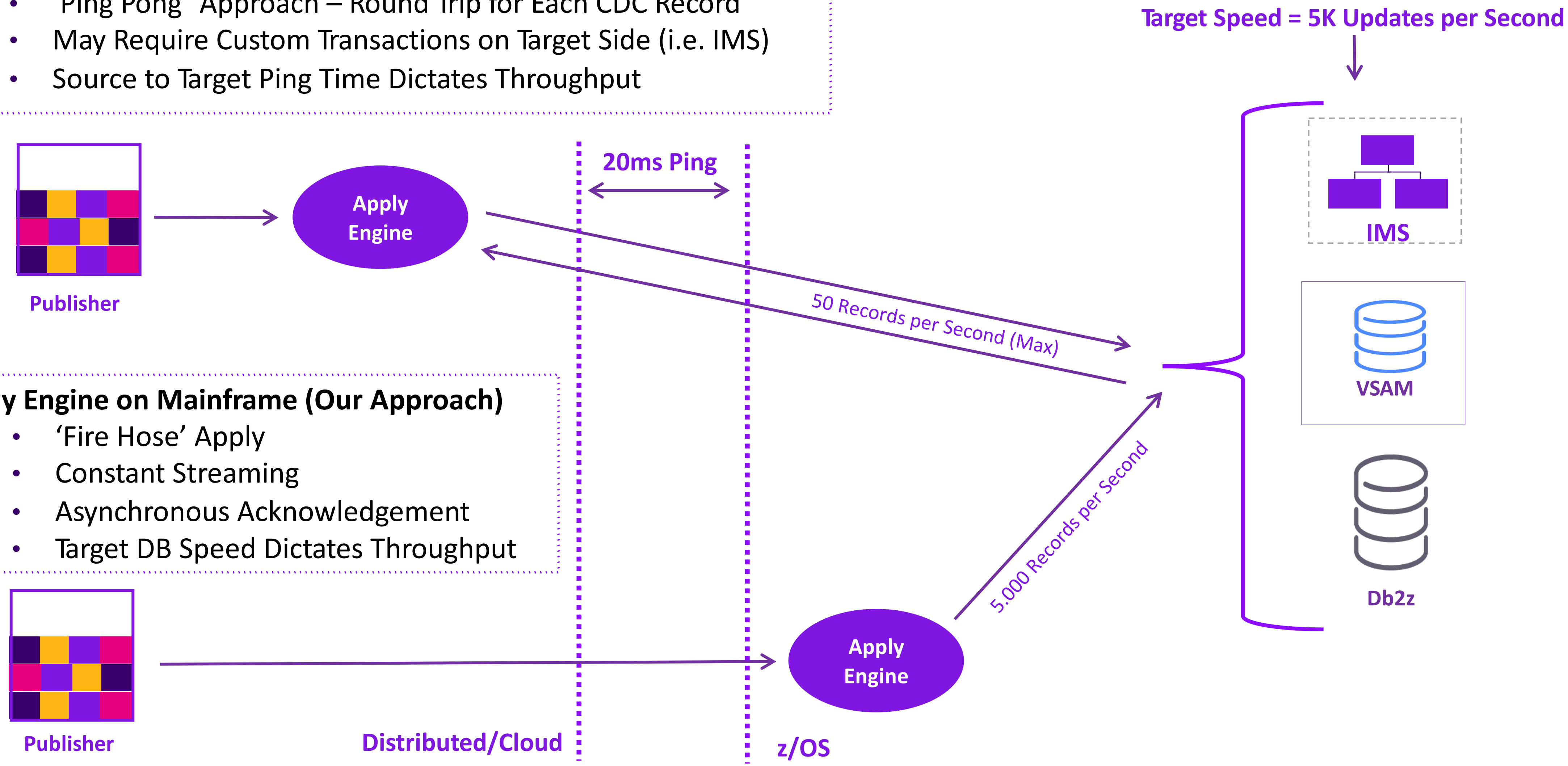
Another Tale of Two Architectures: Reverse Sync

Apply Engine on Distributed / Cloud

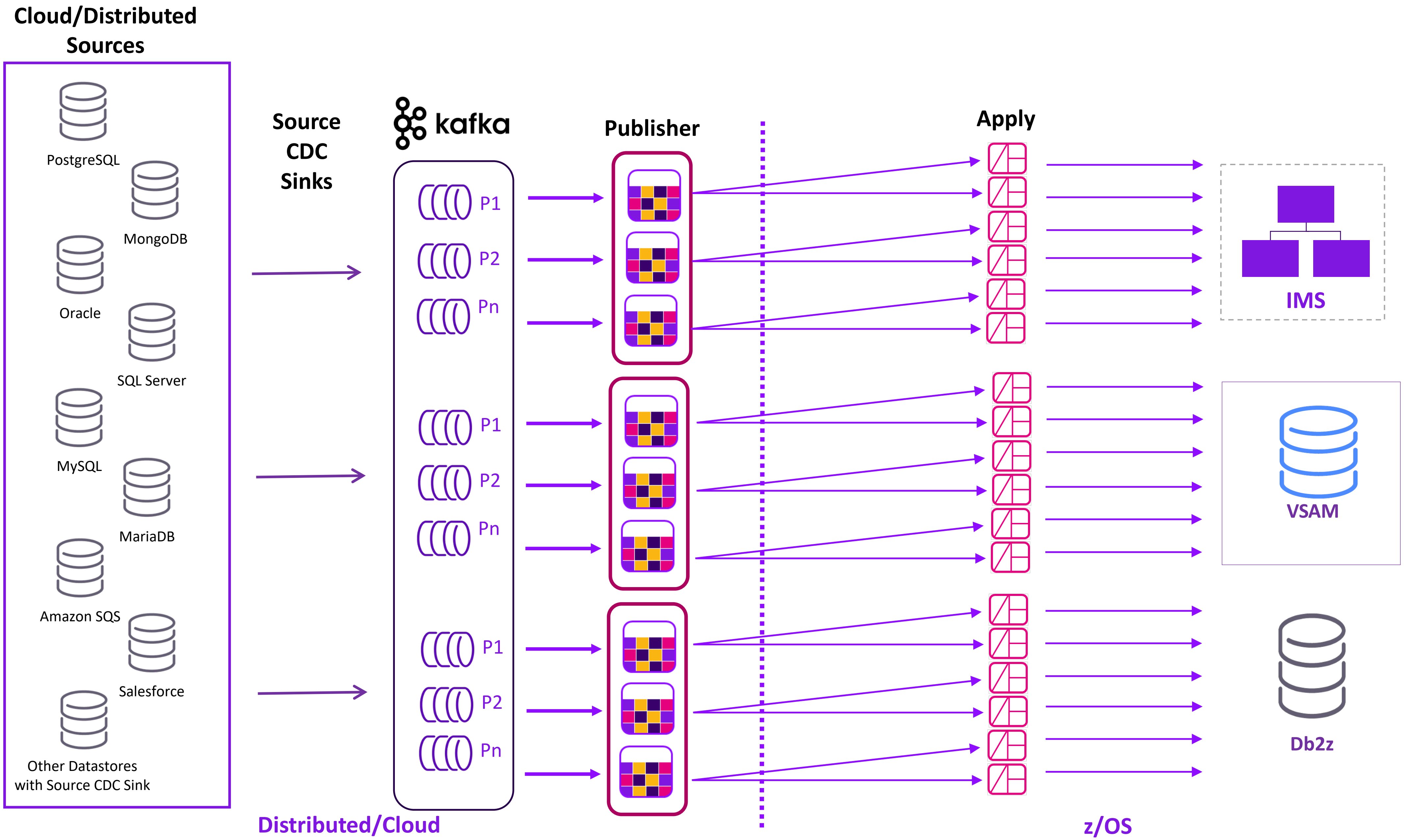
- 'Ping Pong' Approach – Round Trip for Each CDC Record
- May Require Custom Transactions on Target Side (i.e. IMS)
- Source to Target Ping Time Dictates Throughput

Apply Engine on Mainframe (Our Approach)

- 'Fire Hose' Apply
- Constant Streaming
- Asynchronous Acknowledgement
- Target DB Speed Dictates Throughput

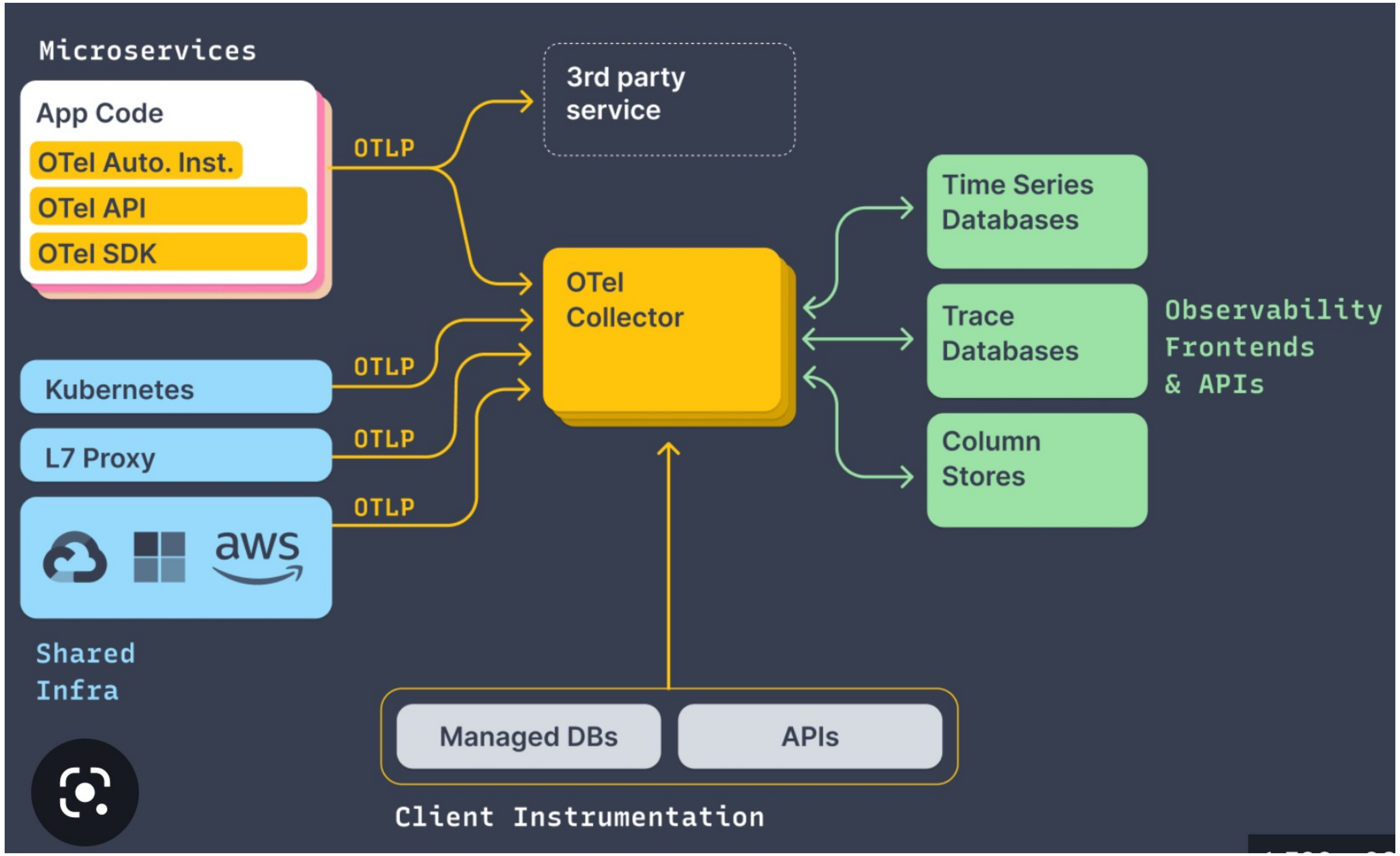


Sample Reverse Synchronization Setup



Observability

- Telemetry
  - Gathers metrics, logs, traces, etc.
  - Everyone plays…all critical agents
- Observability
  - Real time analysis/debugging
  - Explore events not defined in advance
  - Avoids / minimizes outages
- Monitoring
  - Analyzes pre-defined metrics
  - Prometheus backend – time series db
  - Grafana, CloudWatch front ends
- Ushers in the Modern…
  - Cool dashboards
  - Real time debugging
  - Common framework



That’s It for This Year…

- It’s Been All About Latency…
  - Set realistic expectations with the business
  - Low to mid sub-second if you have the right tech
- The Speed of the Target *Generally* Dictates Latency & Throughput
  - Scale on the target side first
  - Minimize load / back-pressure / MIPS on the mainframe
  - Mainframe *absolutely cannot* be the slow link…too much to do on the target side
- The Cloud is Really Not Terrible
  - Modern features with a common framework
  - A bit scary that it won’t work without internet connectivity
  - Need to take precautions…be wary of what’s coming back in the reverse synch pipelines
- Mainframe Plays a Critical Role in a Successful Application Modernization Journey
  - Mainframe skills, particularly IMS, are a premium commodity
  - Resistance is futile…those who ‘roll with the flow’ make the difference

Q&A