AWS re:Invent

BDT310

Big Data Architectural Patterns and Best Practices on AWS

Siva Raghupathy, Principal Solutions Architect, Amazon Web Services

October 2015

amazon web services

What to Expect from the Session

Big data challenges

How to simplify big data processing

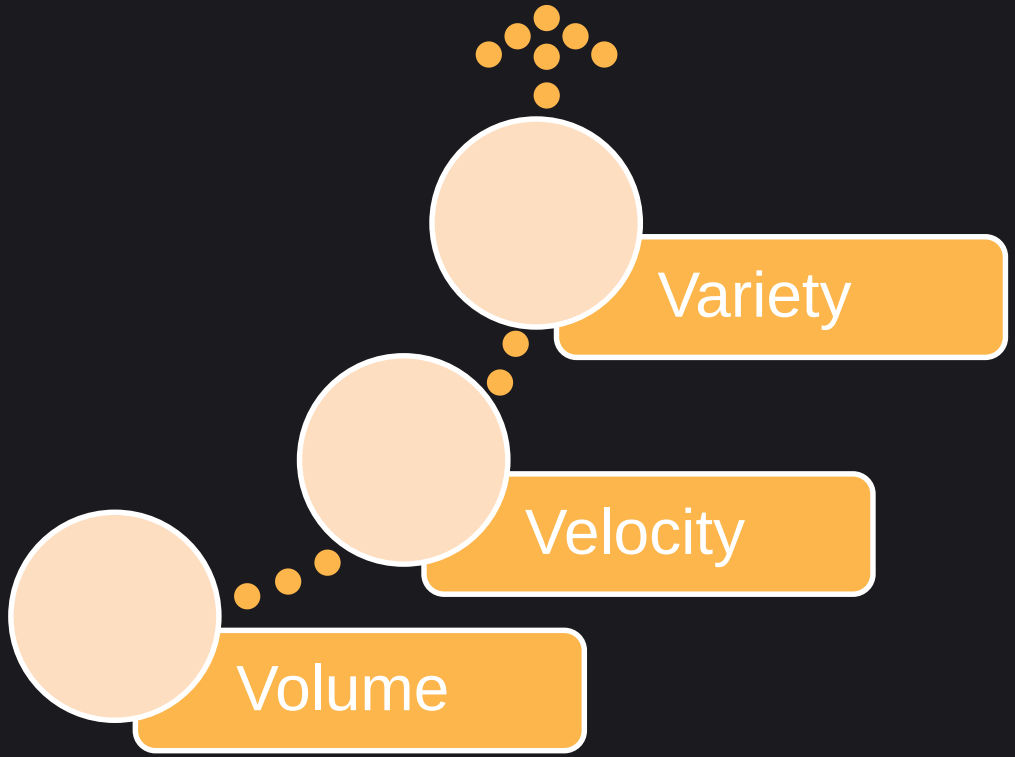
What technologies should you use?

- Why?
- How?

Reference architecture

Design patterns

Ever Increasing Big Data



Big Data Evolution

Batch → Real-time → Prediction

Report

Alerts

Forecast

Plethora of Tools

Cassandra

EMR

S3

DynamoDB

SQS

Amazon Redshift

Amazon Glacier

RDS

ElastiCache

Amazon Kinesis

Kinesis-enabled app

Data Pipeline

CloudSearch

Lambda

ML

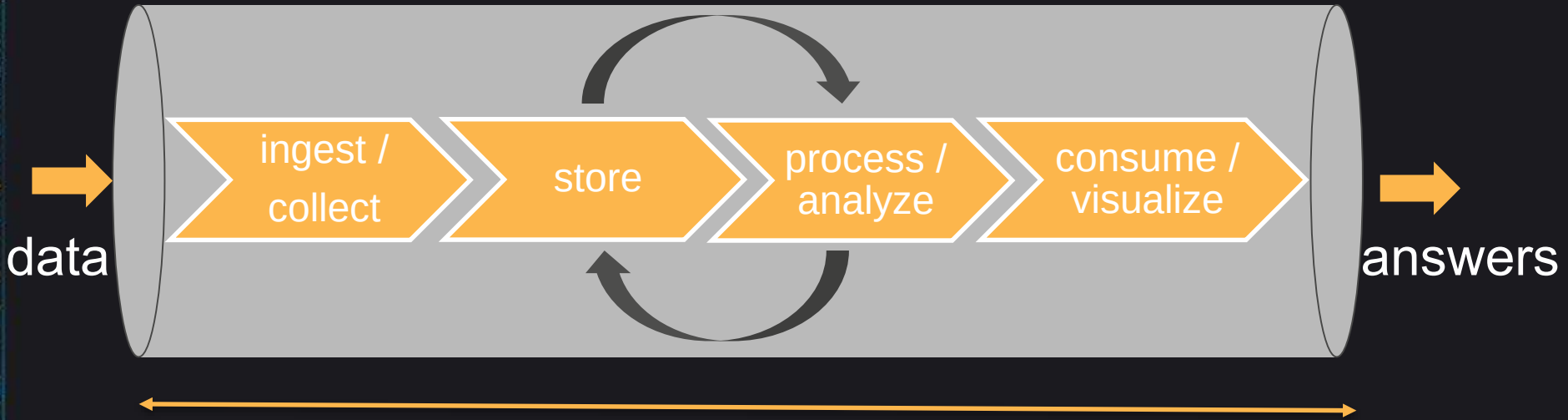
DynamoDB Streams

Is there a reference architecture ?
What tools should I use ?
How ?
Why ?

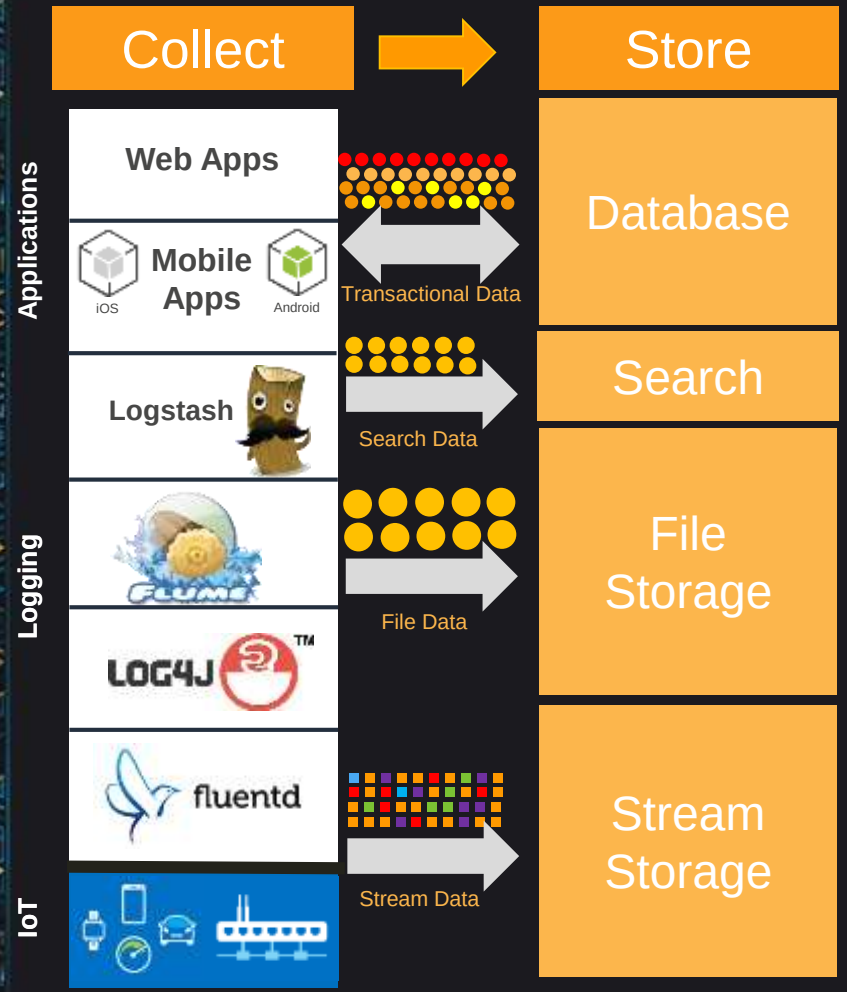
Architectural Principles

- Decoupled “data bus”
  - Data → Store → Process → Answers
- Use the right tool for the job
  - Data structure, latency, throughput, access patterns
- Use Lambda architecture ideas
  - Immutable (append-only) log, batch/speed/serving layer
- Leverage AWS managed services
  - No/low admin
- Big data ≠ big cost

Simplify Big Data Processing



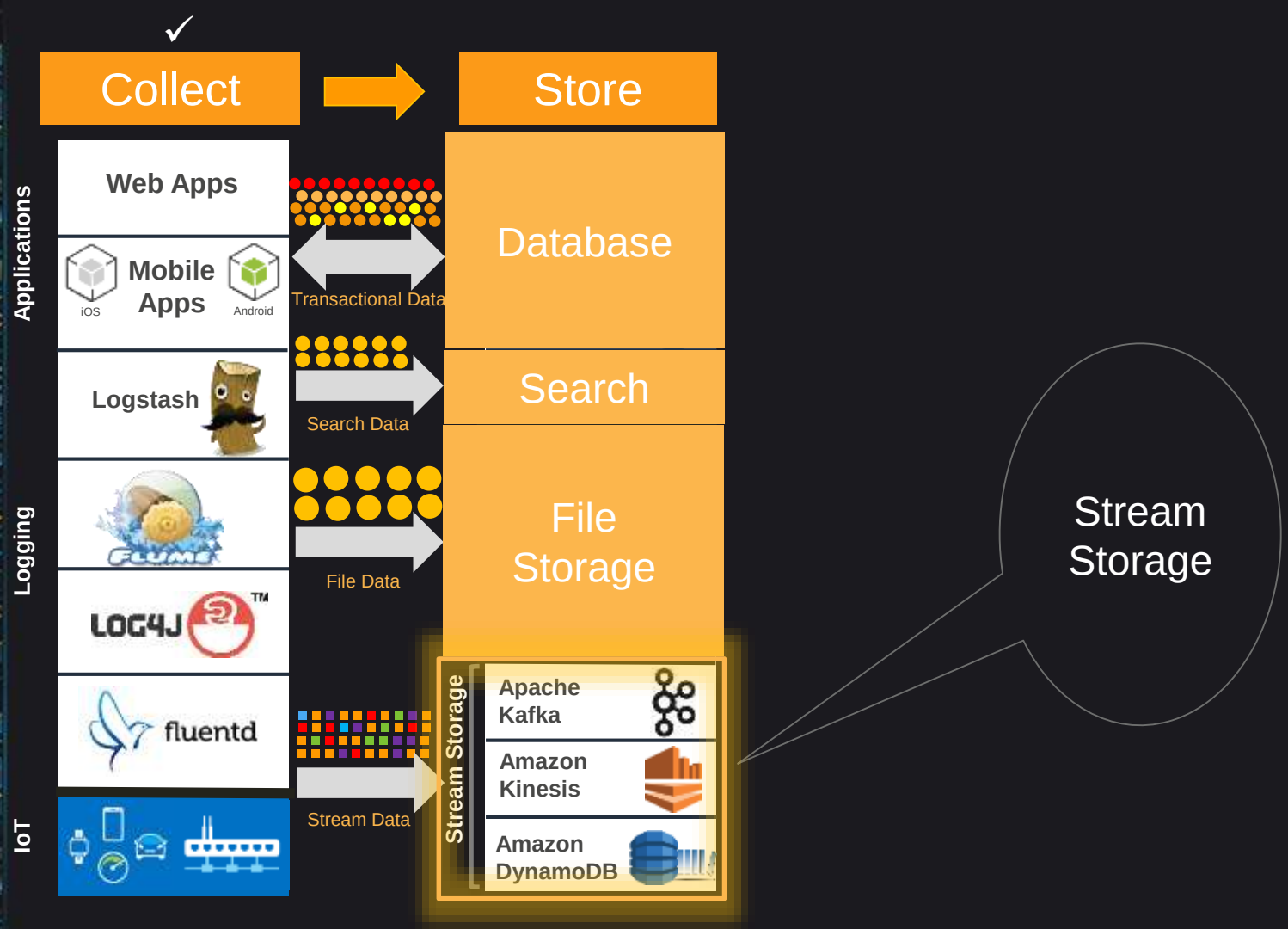

Collect /
Ingest



Types of Data

- Transactional
  - Database reads & writes (OLTP)
  - Cache
- Search
  - Logs
  - Streams
- File
  - Log files (/var/log)
  - Log collectors & frameworks
- Stream
  - Log records
  - Sensors & IoT data


Store

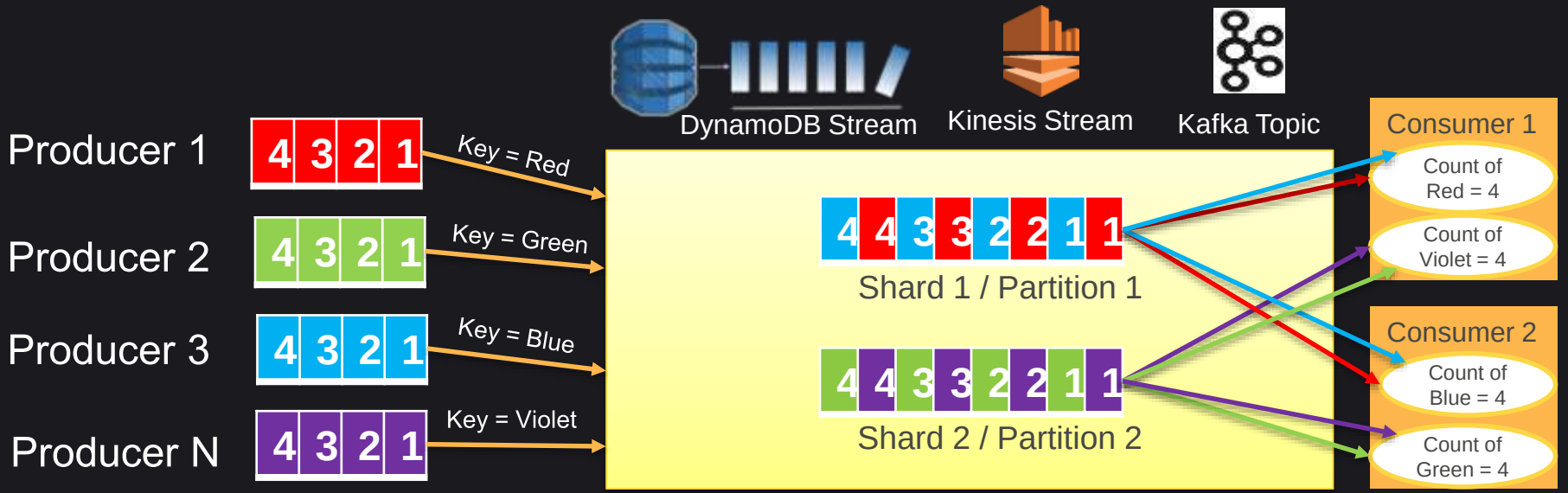

Collect
Store
Applications
Logging
IoT
Web Apps
Mobile Apps
iOS
Android
Logstash
FLUME
LOG4J
fluentd
Transactional Data
Search Data
File Data
Stream Data
Database
Search
File Storage
Stream Storage
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
Stream Storage

Stream Storage Options

- AWS managed services
  - Amazon Kinesis → streams
  - DynamoDB Streams → table + streams
  - Amazon SQS → queue
  - Amazon SNS → pub/sub
- Unmanaged
  - Apache Kafka → stream

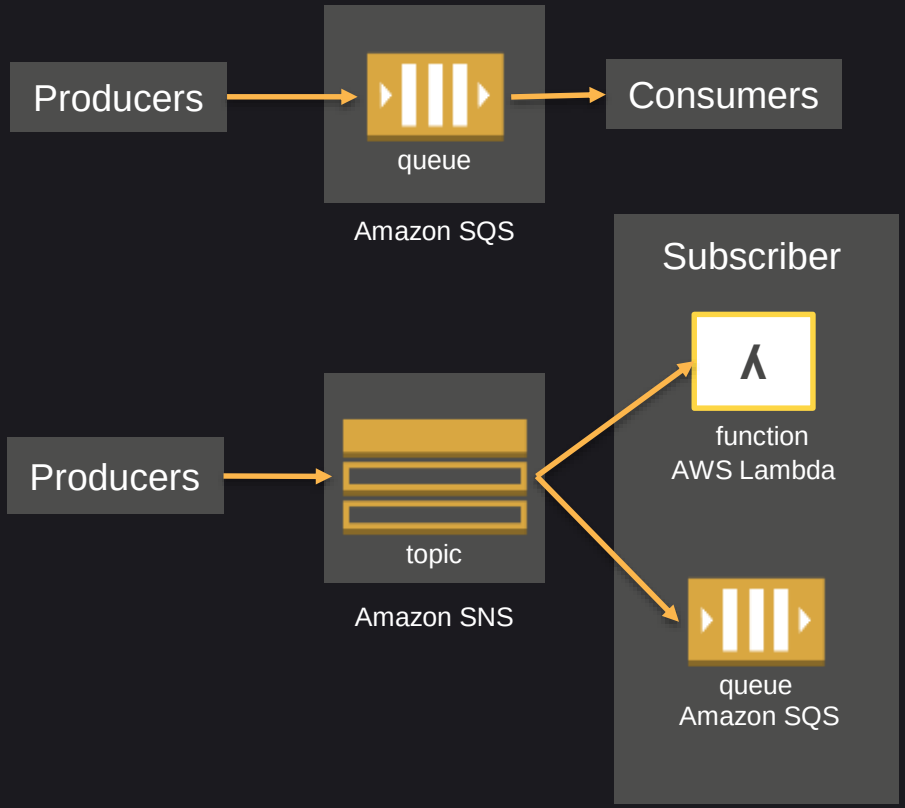
Why Stream Storage?

- Decouple producers & consumers
- Persistent buffer
- Collect multiple streams
- Preserve client ordering
- Streaming MapReduce
- Parallel consumption



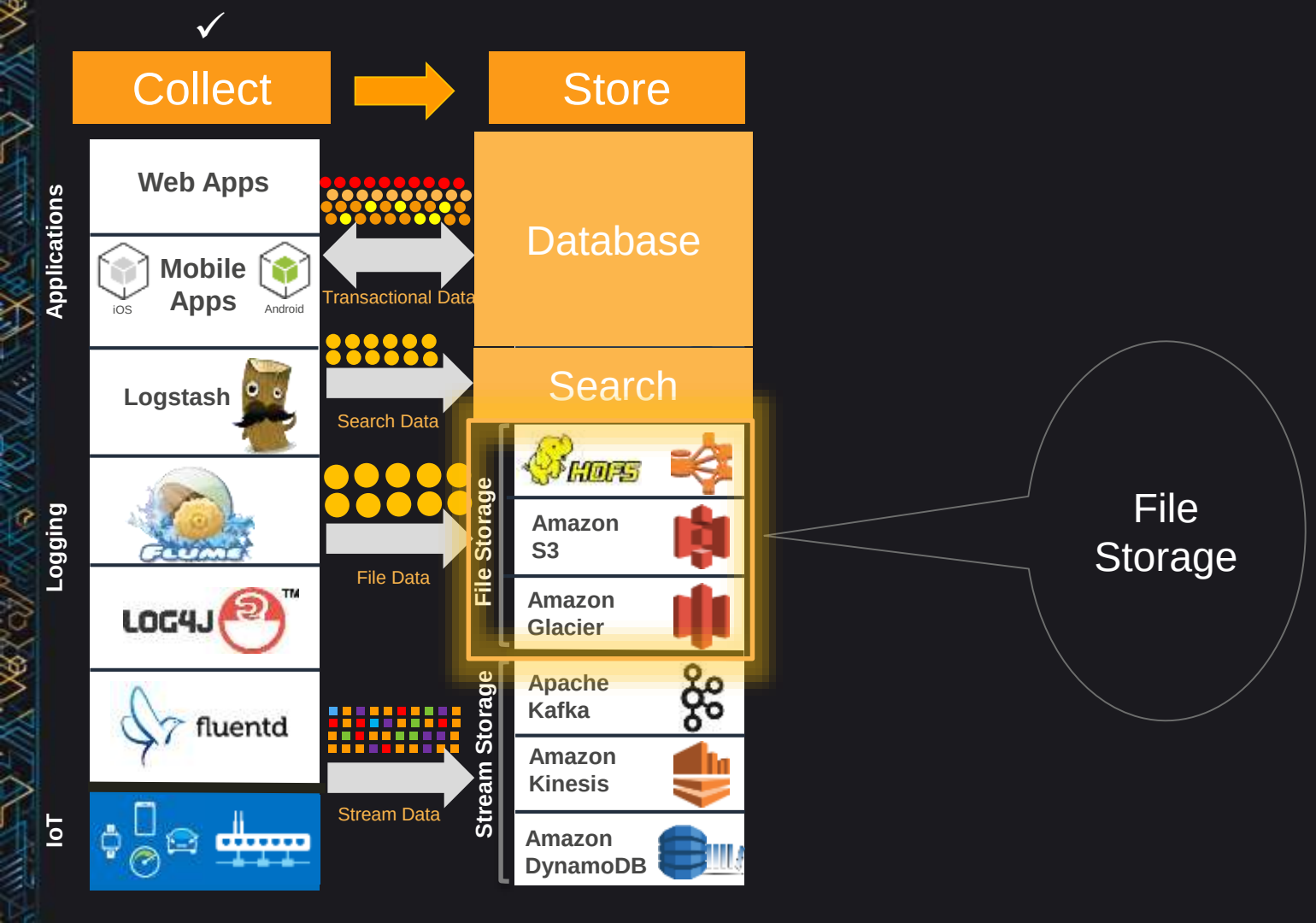
What About Queues & Pub/Sub ?

- Decouple producers & consumers/subscribers
- Persistent buffer
- Collect multiple streams
- No client ordering
- No parallel consumption for Amazon SQS
  - Amazon SNS can route to multiple queues or λ functions
- No streaming MapReduce



Which stream storage should I use?

	Amazon Kinesis	DynamoDB Streams	Amazon SQS Amazon SNS	Kafka
Managed	Yes	Yes	Yes	No
Ordering	Yes	Yes	No	Yes
Delivery	at-least-once	exactly-once	at-least-once	at-least-once
Lifetime	7 days	24 hours	14 days	Configurable
Replication	3 AZ	3 AZ	3 AZ	Configurable
Throughput	No Limit	No Limit	No Limit	~ Nodes
Parallel Clients	Yes	Yes	No (SQS)	Yes
MapReduce	Yes	Yes	No	Yes
Record size	1MB	400KB	256KB	Configurable
Cost	Low	Higher(table cost)	Low-Medium	Low (+admin)

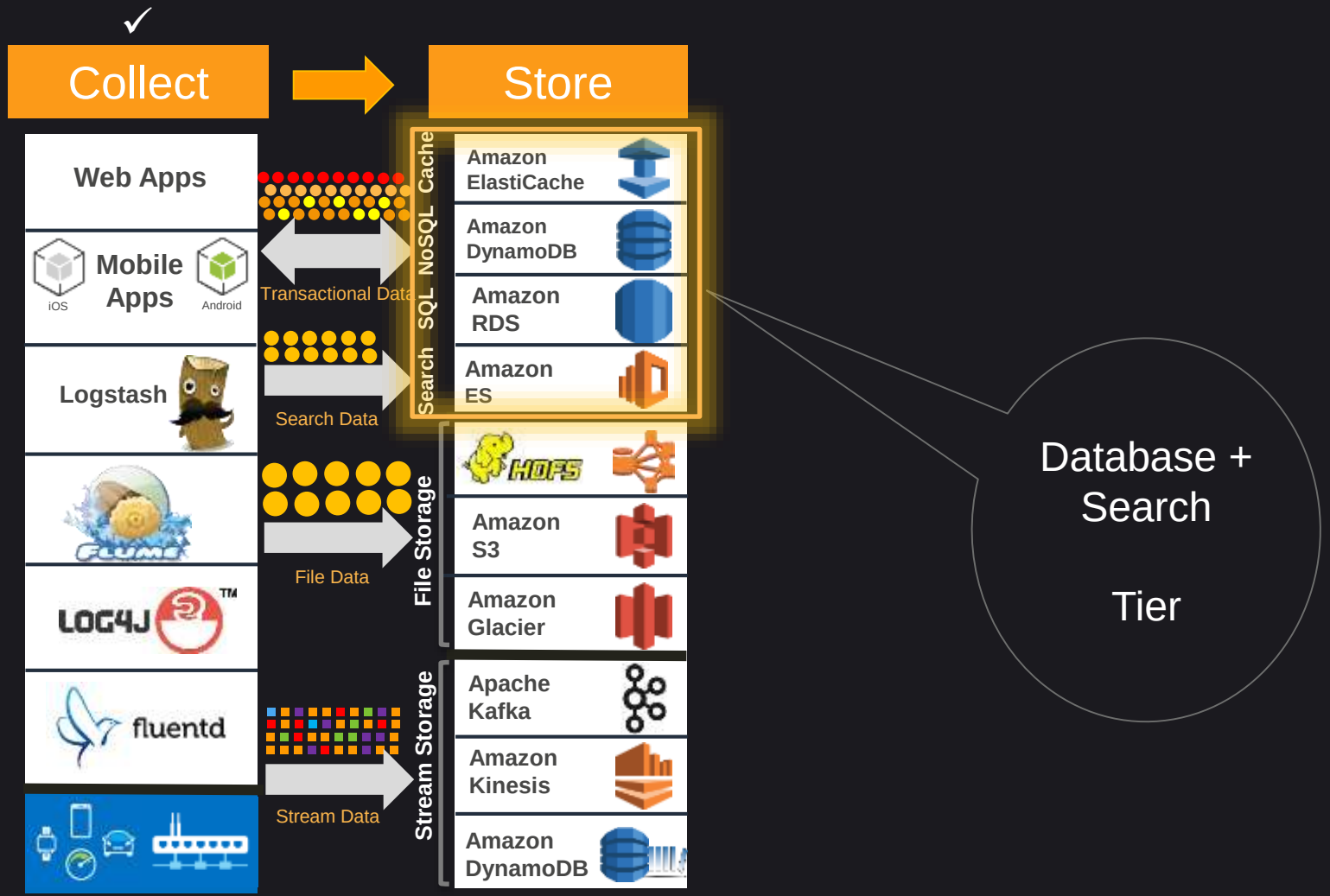

Collect
Store
Applications
Web Apps
Mobile Apps
iOS
Android
Logstash
Logging
Flume
LOG4J
fluentd
IoT
Transactional Data
Search Data
File Data
Stream Data
Database
Search
File Storage
HDFS
Amazon S3
Amazon Glacier
Stream Storage
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
File Storage

Why Is Amazon S3 Good for Big Data?

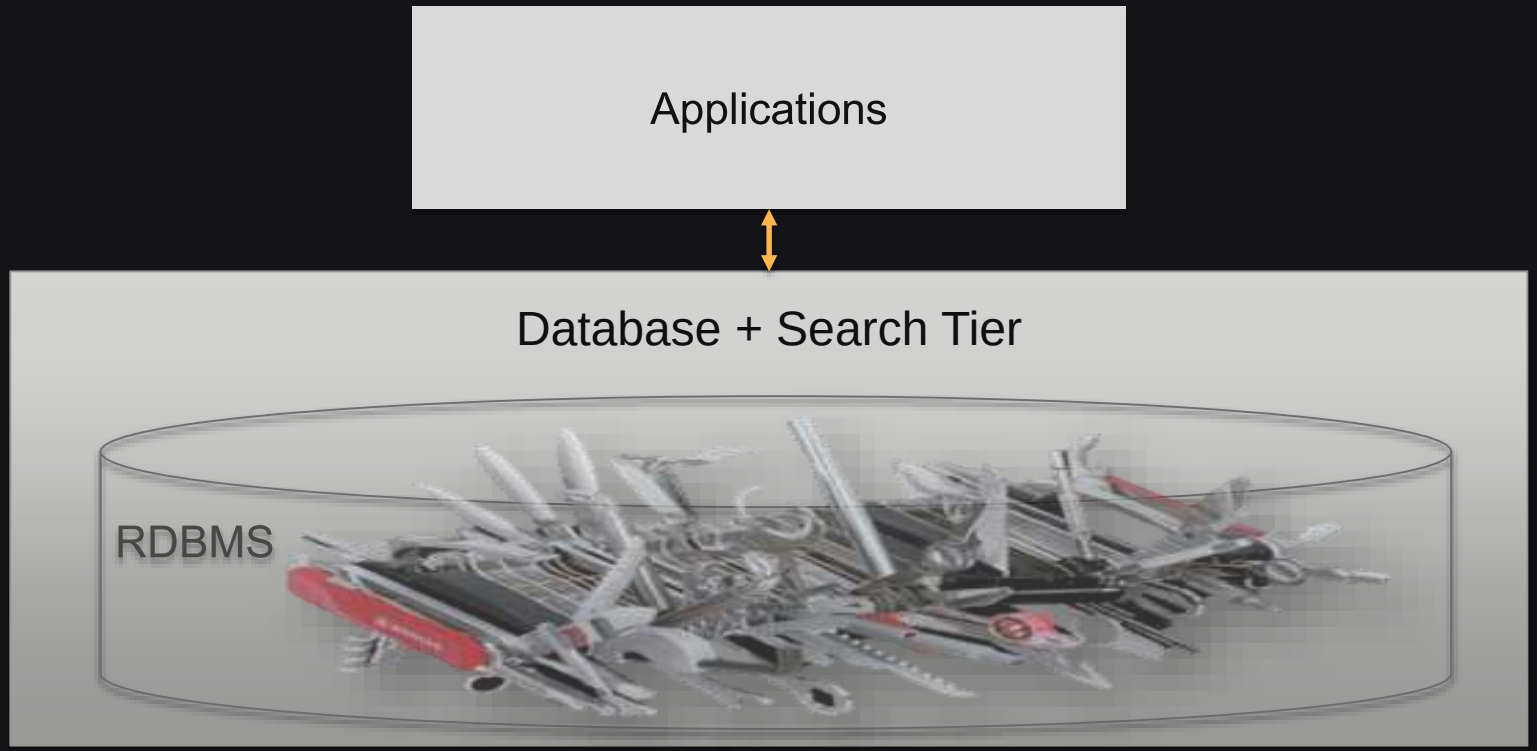
- Natively supported by big data frameworks (Spark, Hive, Presto, etc.)
- No need to run compute clusters for storage (unlike HDFS)
- Can run transient Hadoop clusters & Amazon EC2 Spot instances
- Multiple distinct (Spark, Hive, Presto) clusters can use the same data
- Unlimited number of objects
- Very high bandwidth – no aggregate throughput limit
- Highly available – can tolerate AZ failure
- Designed for 99.999999999% durability
- Tired-storage (Standard, IA, Amazon Glacier) via life-cycle policy
- Secure – SSL, client/server-side encryption at rest
- Low cost

What about HDFS & Amazon Glacier?

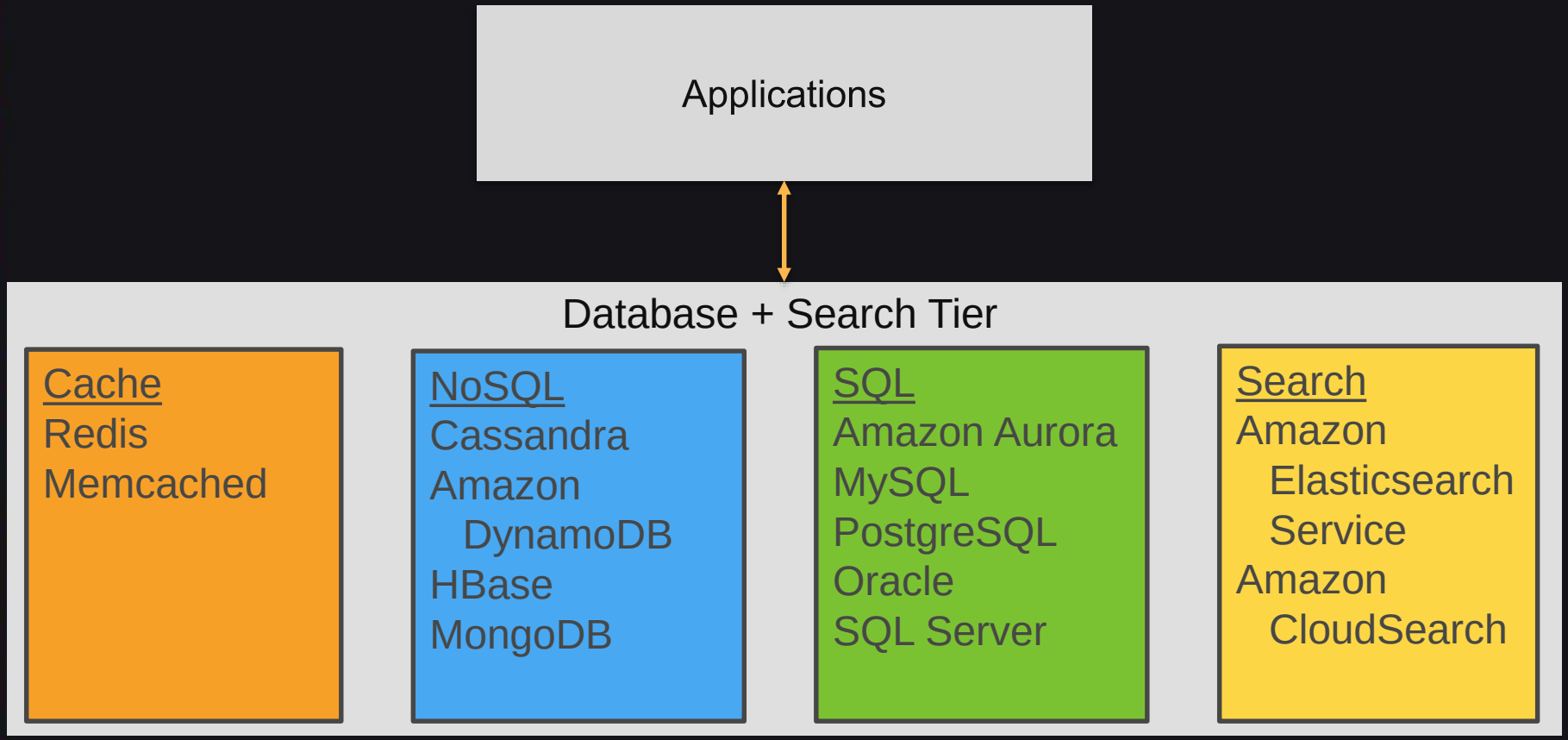
- Use HDFS for very frequently accessed (hot) data
- Use Amazon S3 Standard for frequently accessed data
- Use Amazon S3 Standard – IA for infrequently accessed data
- Use Amazon Glacier for archiving cold data


Collect
Store
Web Apps
Mobile Apps
iOS
Android
Logstash
FLUME
LOG4J
fluentd
Transactional Data
Search Data
File Data
Stream Data
Cache
NoSQL
SQL
Search
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
HDFS
Amazon S3
Amazon Glacier
File Storage
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
Stream Storage
Database + Search
Tier

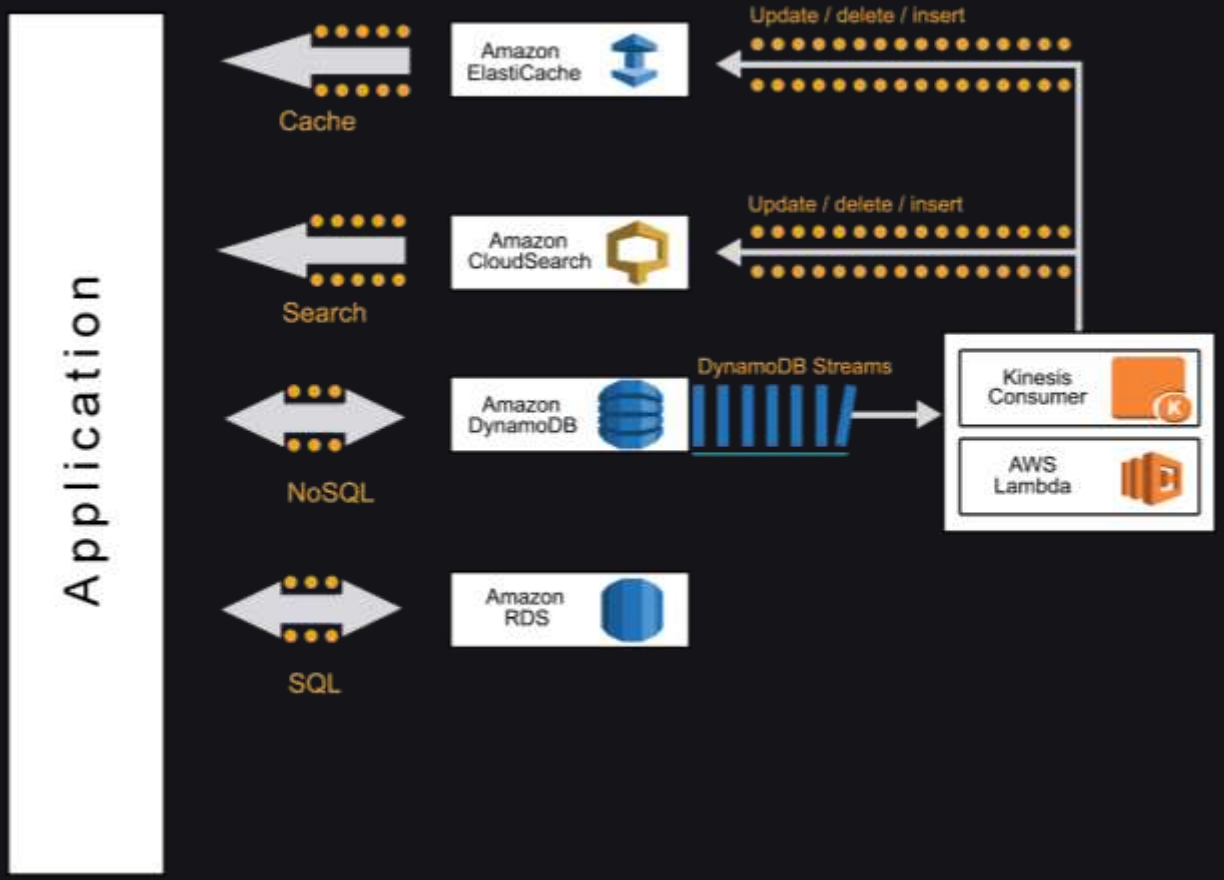
Database + Search Tier Anti-pattern



Best Practice — Use the Right Tool for the Job



Materialized Views



What Data Store Should I Use?

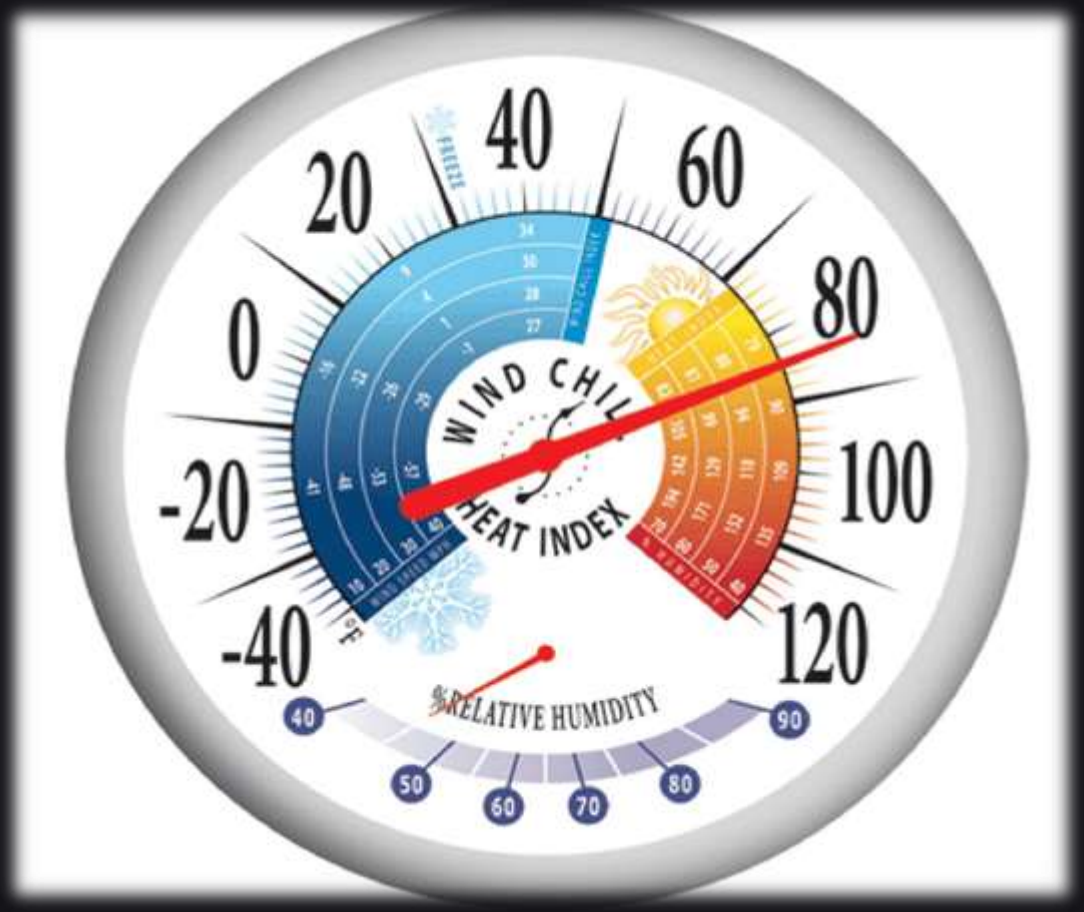
- Data structure → Fixed schema, JSON, key-value
- Access patterns → Store data in the format you will access it
- Data / access characteristics → Hot, warm, cold
- Cost → Right cost

Data Structure and Access Patterns

Access Patterns	What to use?
Put/Get (Key, Value)	Cache, NoSQL
Simple relationships → 1:N, M:N	NoSQL
Cross table joins, transaction, SQL	SQL
Faceting, Search	Search

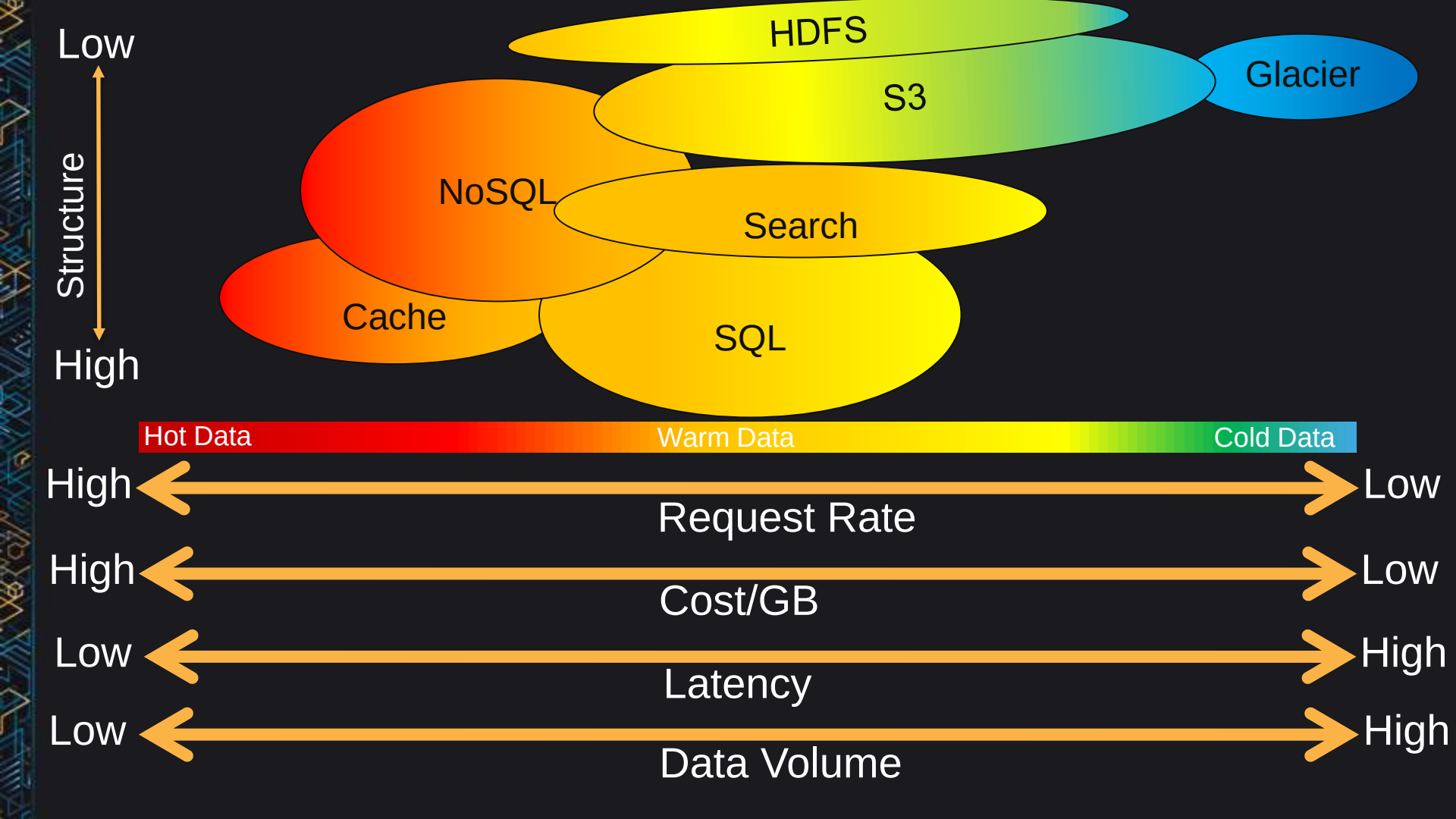
Data Structure	What to use?
Fixed schema	SQL, NoSQL
Schema-free (JSON)	NoSQL, Search
(Key, Value)	Cache, NoSQL

What Is the Temperature of Your Data / Access ?



Data / Access Characteristics: Hot, Warm, Cold

	Hot	Warm	Cold
Volume	MB–GB	GB–TB	PB
Item size	B–KB	KB–MB	KB–TB
Latency	ms	ms, sec	min, hrs
Durability	Low–High	High	Very High
Request rate	Very High	High	Low
Cost/GB	$$-$	$-¢¢	¢


Low
Structure
High
HDFS
S3
Glacier
NoSQL
Search
Cache
SQL
Hot Data
Warm Data
Cold Data
High
Request Rate
Low
High
Cost/GB
Low
Low
Latency
High
Low
Data Volume
High

What Data Store Should I Use?

Hot Data — Warm Data — Cold Data

	Amazon ElastiCache	Amazon DynamoDB	Amazon Aurora	Amazon Elasticsearch	Amazon EMR (HDFS)	Amazon S3	Amazon Glacier
Average latency	ms	ms	ms, sec	ms,sec	sec,min,hrs	ms,sec,min (~ size)	hrs
Data volume	GB	GB–TBs (no limit)	GB–TB (64 TB Max)	GB–TB	GB–PB (~nodes)	MB–PB (no limit)	GB–PB (no limit)
Item size	B-KB	KB (400 KB max)	KB (64 KB)	KB (1 MB max)	MB-GB	KB-GB (5 TB max)	GB (40 TB max)
Request rate	High - Very High	Very High (no limit)	High	High	Low – Very High	Low – Very High (no limit)	Very Low
Storage cost GB/month	$$	¢¢	¢¢	¢¢	¢	¢	¢/10
Durability	Low - Moderate	Very High	Very High	High	High	Very High	Very High

Hot Data — Warm Data — Cold Data

Cost Conscious Design

Example: Should I use Amazon S3 or Amazon DynamoDB?

"I'm currently scoping out a project that will greatly increase my team's use of Amazon S3. Hoping you could answer some questions. The current iteration of the design calls for **many small files**, perhaps up to a **billion during peak**. The **total size** would be on the order of **1.5 TB per month**…"

Request rate (Writes/sec)	Object size (Bytes)	Total size (GB/month)	Objects per month
300	2048	1483	777,600,000

Cost Conscious Design

Example: Should I use Amazon S3 or Amazon DynamoDB?

https://calculator.s3.amazonaws.com/index.html

Amazon S3 or Amazon DynamoDB?

Request rate (Writes/sec)	Object size (Bytes)	Total size (GB/month)	Objects per month
300	2,048	1,483	777,600,000

Amazon DynamoDB is a high performance non-relational database service that is easy to set up, operate, and scale. It is designed to address the core problems of database management, performance, scalability, and reliability. It also provides predictable high performance and low latency at scale.

Indexed Data Storage:

Dataset Size: 1483 GB

Provisioned Throughput Capacity *:

Item Size (All attributes): 2 KB

Number of items read per second: 0 Reads/Second

Read Consistency: Strongly Consistent / Eventually Cons cheaper)

Number of items written per second: 300 Writes/Second

Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Storage:

Storage: 1483 GB

Reduced Redundancy Storage: 0 GB

Requests:

PUT/COPY/POST/LIST Requests: 77760000 Requests

GET and Other Requests: 0 Requests

Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	

amazon web services SIMPLE MONTHLY CALCULATOR

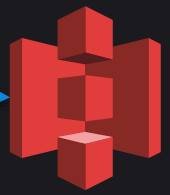
Amazon DynamoDB

use

Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	
DynamoDB Streams:	$ 0.00	

	Request rate (Writes/sec)	Object size (Bytes)	Total size (GB/month)	Objects per month
Scenario 1	300	2,048	1,483	777,600,000
Scenario 2	300	32,768	23,730	777,600,000

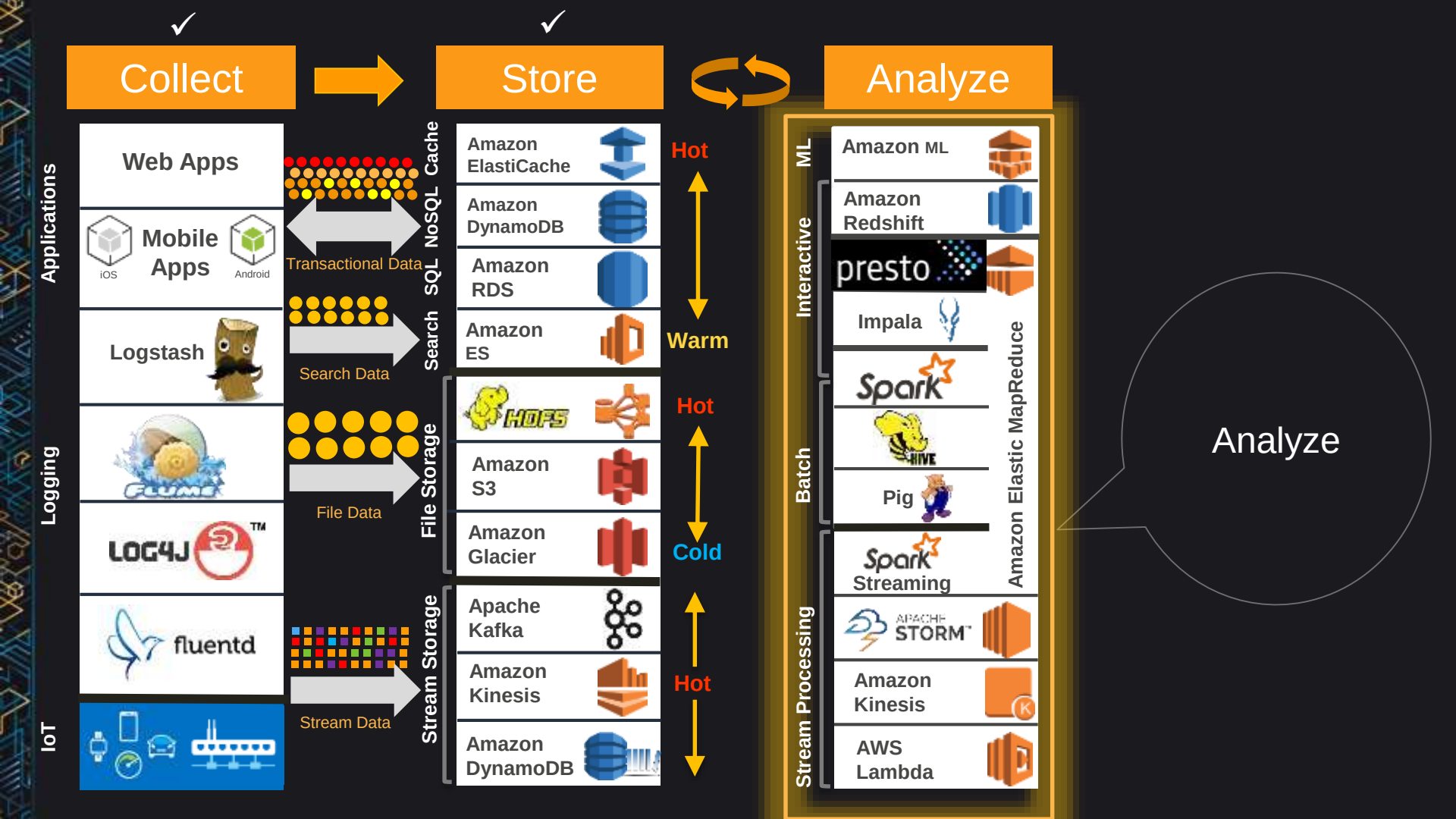
use



Amazon S3

Amazon S3 Service (US-East)		$ 4588.55
Storage:	$ 700.55	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 10131.40
Provisioned Throughput Capacity:	$ 4187.04	
Indexed Data Storage:	$ 5944.36	
DynamoDB Streams:	$ 0.00	


Process /
Analyze


Collect
Store
Analyze
Applications
Web Apps
Mobile Apps
iOS
Android
Logging
Logstash
Flume
Log4J
fluentd
IoT
Transactional Data
Search Data
File Data
Stream Data
Cache
NoSQL
SQL
Search
File Storage
Stream Storage
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
HDFS
Amazon S3
Amazon Glacier
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
Hot
Warm
Hot
Cold
Hot
ML
Amazon ML
Interactive
Amazon Redshift
presto
Impala
Batch
Spark
Hive
Pig
Stream Processing
Spark Streaming
Apache Storm
Amazon Kinesis
AWS Lambda
Amazon Elastic MapReduce
Analyze

Process / Analyze

Analysis of data is a process of inspecting, cleaning, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.

Examples

- Interactive dashboards → Interactive analytics
- Daily/weekly/monthly reports → Batch analytics
- Billing/fraud alerts, 1 minute metrics → Real-time analytics
- Sentiment analysis, prediction models → Machine learning

Interactive Analytics

Takes large amount of (warm/cold) data

Takes seconds to get answers back

Example: Self-service dashboards

Batch Analytics

Takes large amount of (warm/cold) data

Takes minutes or hours to get answers back

Example: Generating daily, weekly, or monthly reports

Real-Time Analytics

Take small amount of hot data and ask questions

Takes short amount of time (milliseconds or seconds) to get your answer back

- Real-time (event)
  - Real-time response to events in data streams
  - Example: Billing/Fraud Alerts
- Near real-time (micro-batch)
  - Near real-time operations on small batches of events in data streams
  - Example: 1 Minute Metrics

Predictions via Machine Learning

ML gives computers the ability to learn without being explicitly programmed

Machine Learning Algorithms:

- Supervised Learning ← “teach” program
    - Classification ← Is this transaction fraud? (Yes/No)
    - Regression ← Customer Life-time value?
- Unsupervised Learning ← let it learn by itself
    - Clustering ← Market Segmentation

Analysis Tools and Frameworks

Machine Learning

- Mahout, Spark ML, Amazon ML

Interactive Analytics

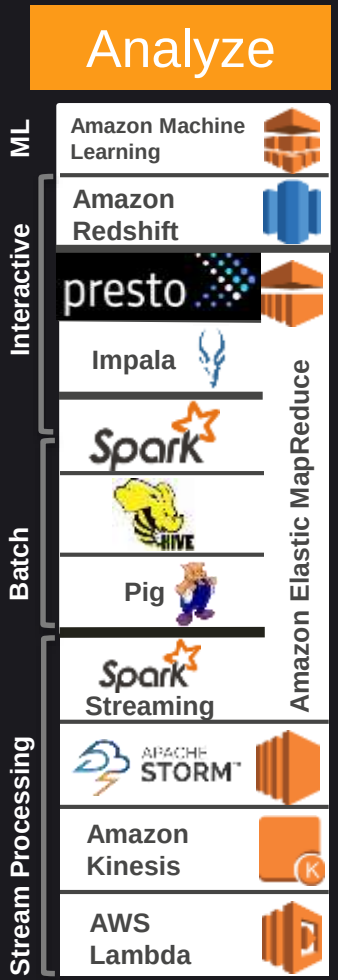
- Amazon Redshift, Presto, Impala, Spark

Batch Processing

- MapReduce, Hive, Pig, Spark

Stream Processing

- Micro-batch: Spark Streaming, KCL, Hive, Pig
- Real-time: Storm, AWS Lambda, KCL



What Stream Processing Technology Should I Use?

	Spark Streaming	Apache Storm	Amazon Kinesis Client Library	AWS Lambda	Amazon EMR (Hive, Pig)
Scale / Throughput	~ Nodes	~ Nodes	~ Nodes	Automatic	~ Nodes
Batch or Real-time	Real-time	Real-time	Real-time	Real-time	Batch
Manageability	Yes (Amazon EMR)	Do it yourself	Amazon EC2 + Auto Scaling	AWS managed	Yes (Amazon EMR)
Fault Tolerance	Single AZ	Configurable	Multi-AZ	Multi-AZ	Single AZ
Programming languages	Java, Python, Scala	Any language via Thrift	Java, via MultiLangDaemon (.Net, Python, Ruby, Node.js)	Node.js, Java	Hive, Pig, Streaming languages

What Data Processing Technology Should I Use?

	Amazon Redshift	Impala	Presto	Spark	Hive
Query Latency	Low	Low	Low	Low	Medium (Tez) – High (MapReduce)
Durability	High	High	High	High	High
Data Volume	1.6 PB Max	~Nodes	~Nodes	~Nodes	~Nodes
Managed	Yes	Yes (EMR)	Yes (EMR)	Yes (EMR)	Yes (EMR)
Storage	Native	HDFS / S3A*	HDFS / S3	HDFS / S3	HDFS / S3
SQL Compatibility	High	Medium	High	Low (SparkSQL)	Medium (HQL)

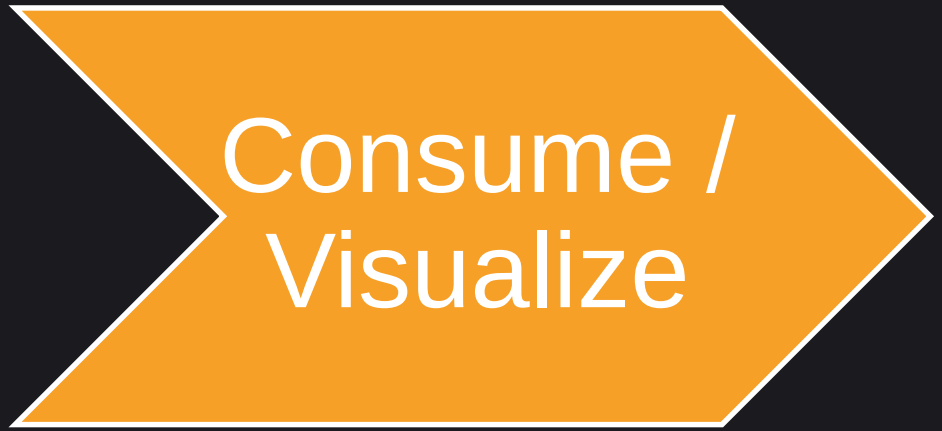
What about ETL?

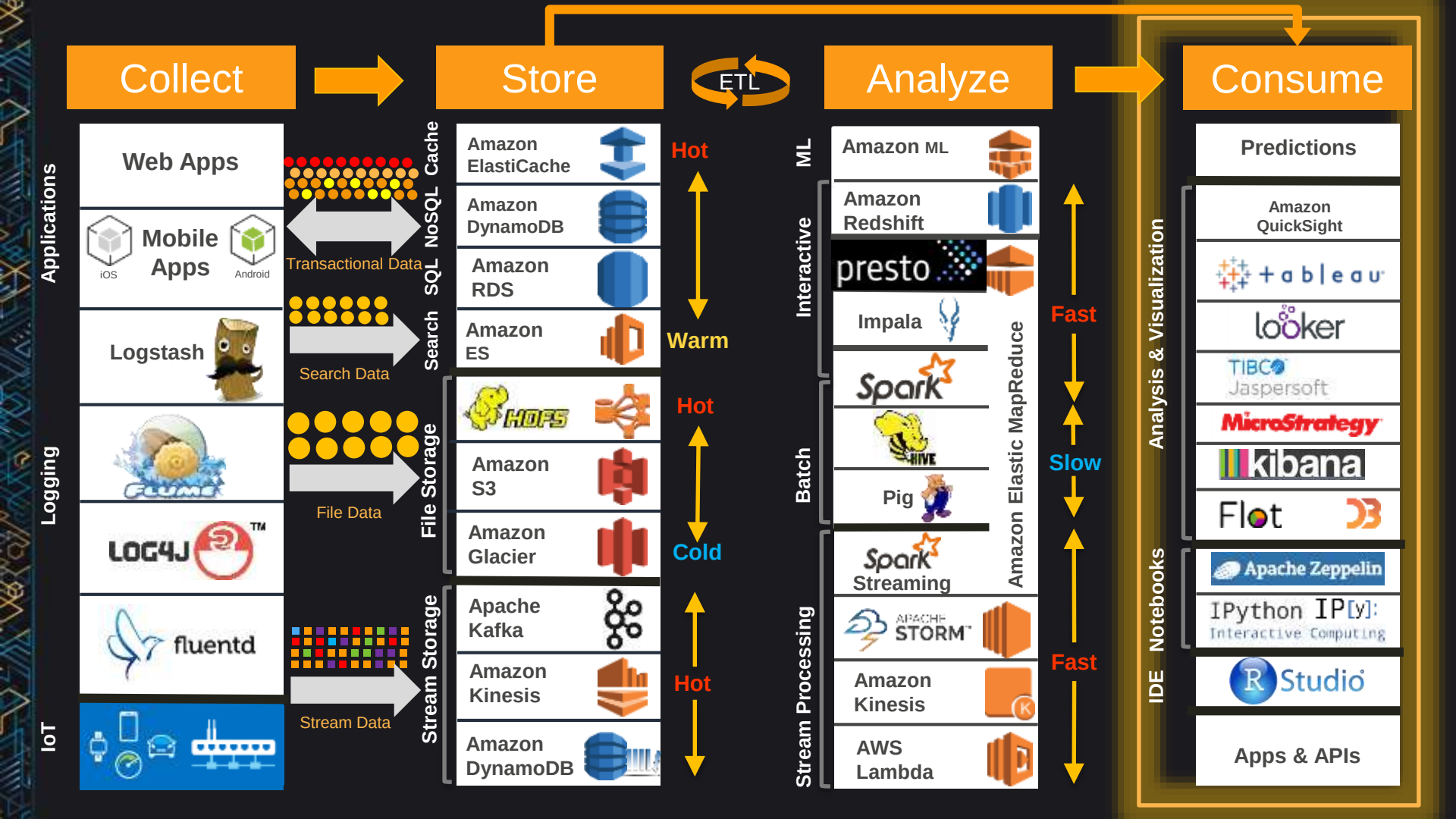
Data Integration

Reduce the effort to move, cleanse, synchronize, manage, and automatize data related processes.

Attunity CloudBeam	Informatica Cloud	Matillion ETL for Redshift	snapLogic
alteryx			

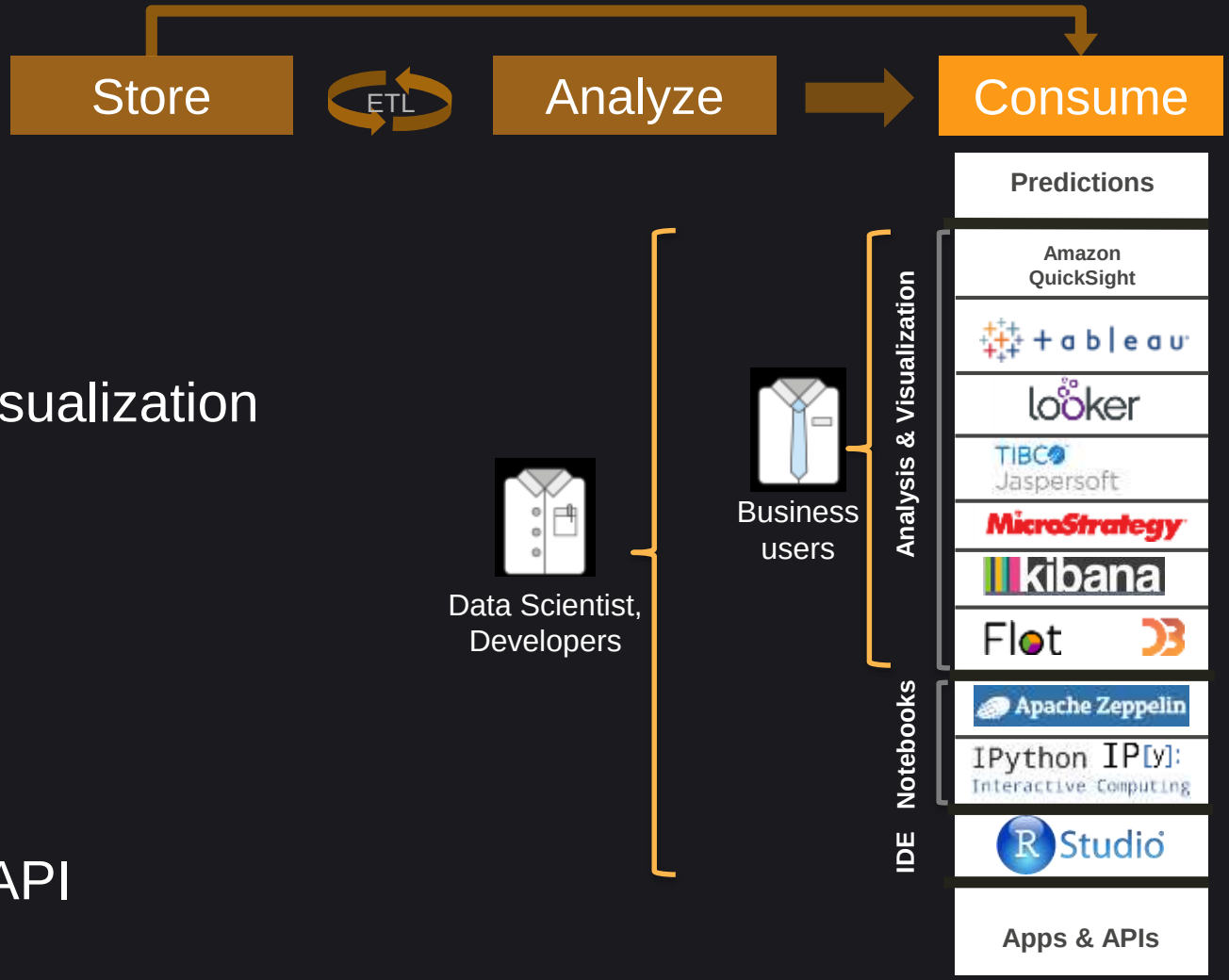
https://aws.amazon.com/big-data/partner-solutions/


Consume /
Visualize


Collect
Store
ETL
Analyze
Consume
Applications
Web Apps
Mobile Apps
iOS
Android
Logging
Logstash
Flume
Log4J
fluentd
IoT
Transactional Data
Search Data
File Data
Stream Data
Cache
NoSQL
SQL
Search
File Storage
Stream Storage
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
HDFS
Amazon S3
Amazon Glacier
Apache Kafka
Amazon Kinesis
Amazon DynamoDB
Hot
Warm
Hot
Cold
Hot
ML
Interactive
Batch
Stream Processing
Amazon ML
Amazon Redshift
presto
Impala
Spark
Hive
Pig
Spark Streaming
Apache Storm
Amazon Kinesis
AWS Lambda
Amazon Elastic MapReduce
Fast
Slow
Fast
Predictions
Analysis & Visualization
Amazon QuickSight
tableau
looker
TIBCO Jaspersoft
MicroStrategy
kibana
Flot
Notebooks
IDE
Apache Zeppelin
IPython IP[y]: Interactive Computing
RStudio
Apps & APIs

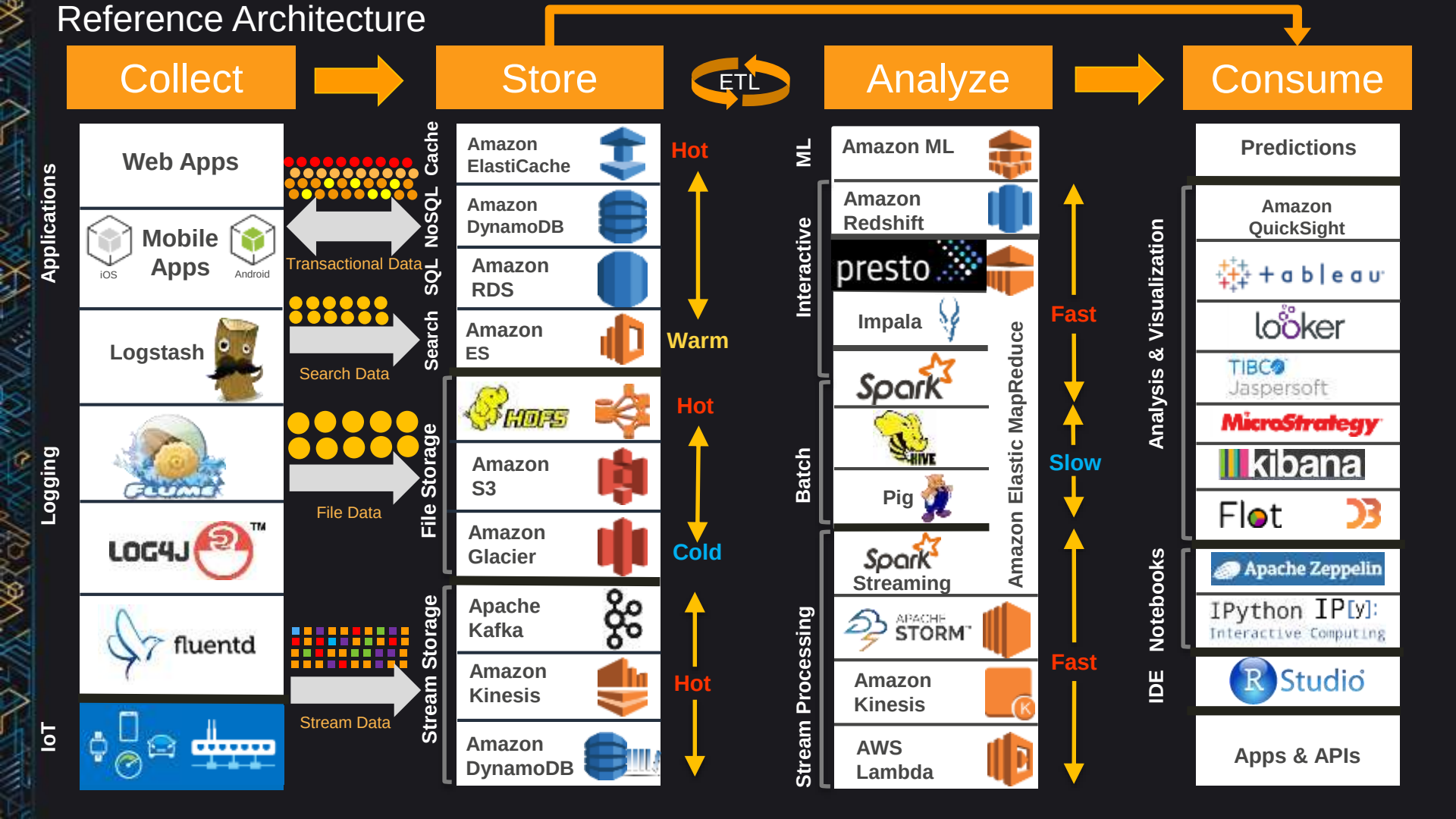
Consume

- Predictions
- Analysis and Visualization
- Notebooks
- IDE
- Applications & API



Putting It All Together

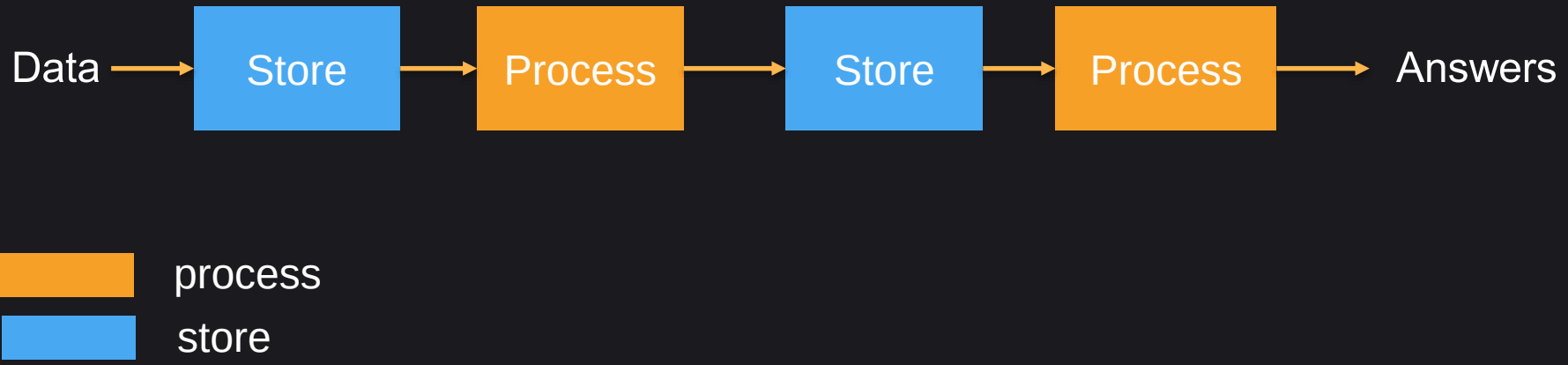
Reference Architecture



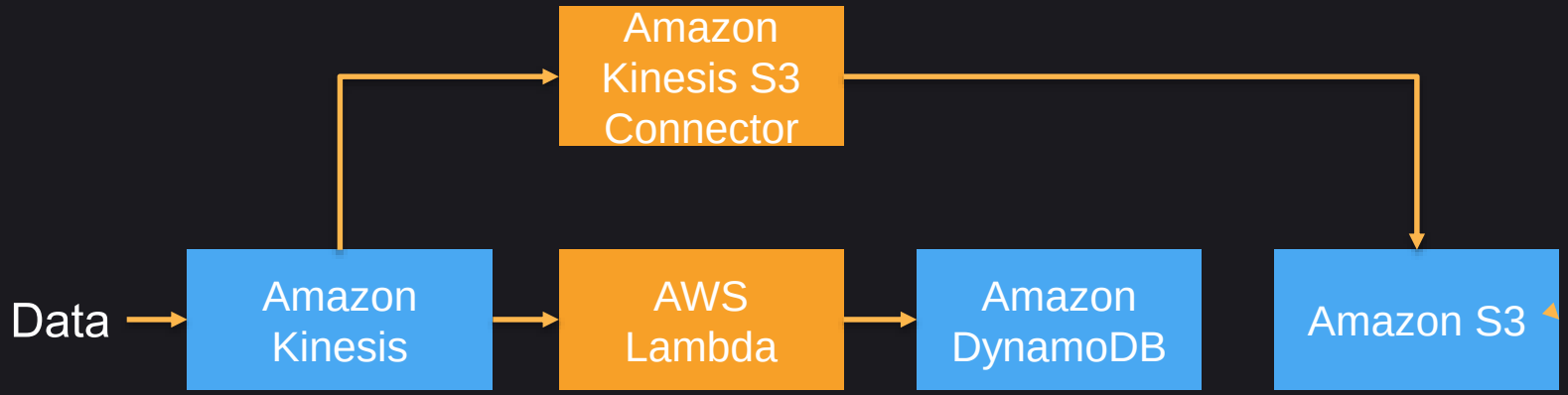
Design Patterns

Multi-Stage Decoupled “Data Bus”

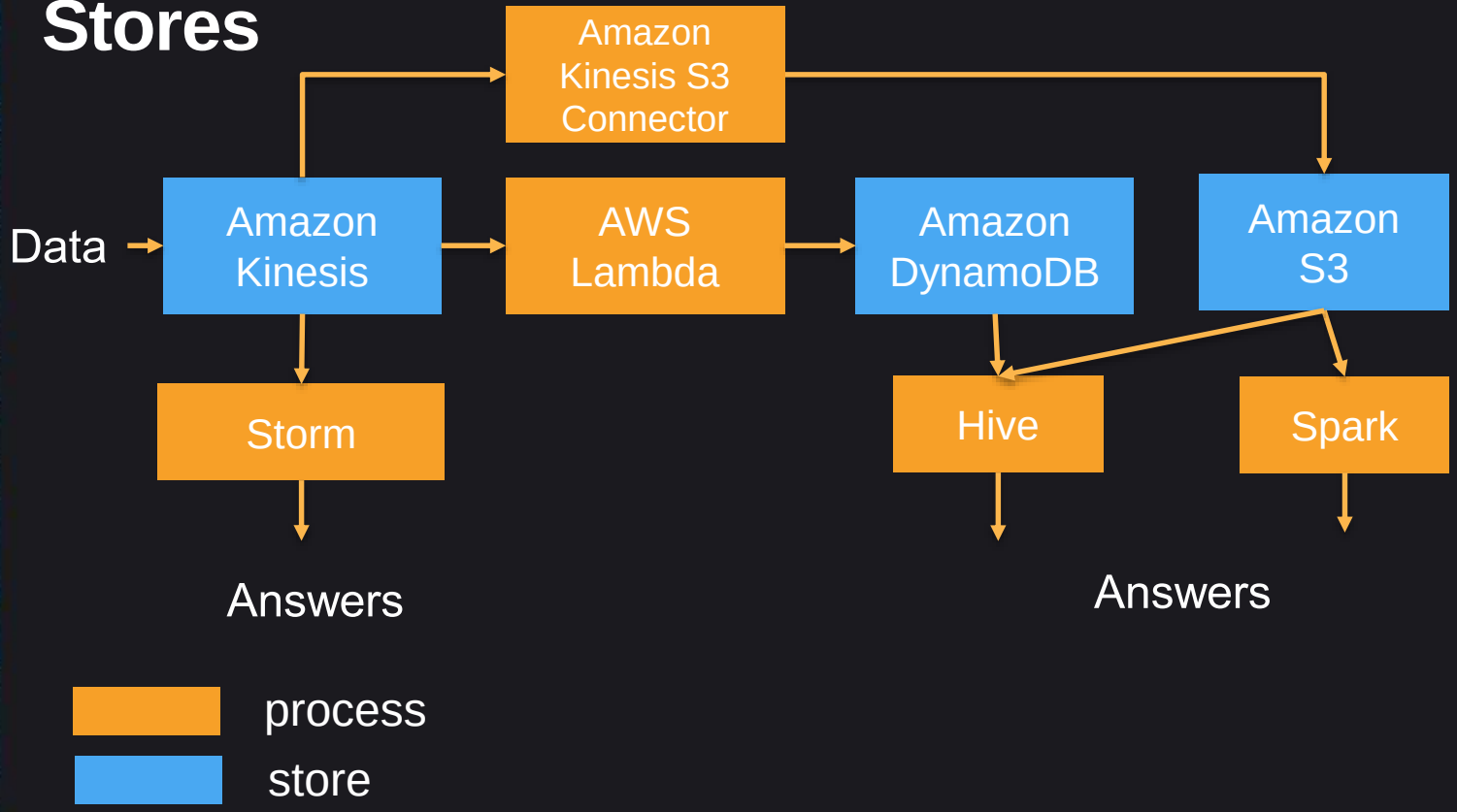
- Multiple stages
- Storage decoupled from processing



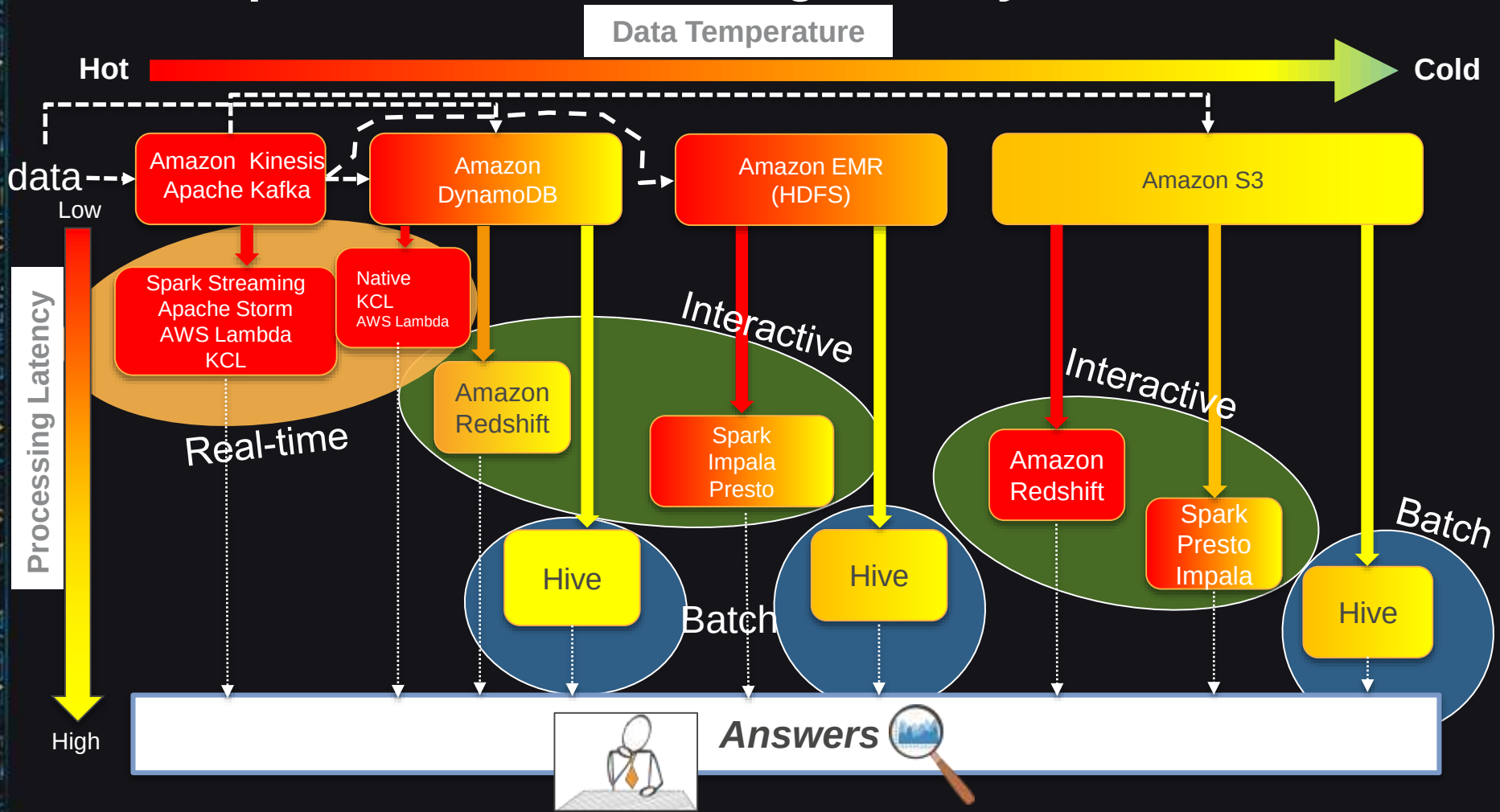
Multiple Processing Applications (or Connectors) Can Read from or Write to Multiple Data Stores



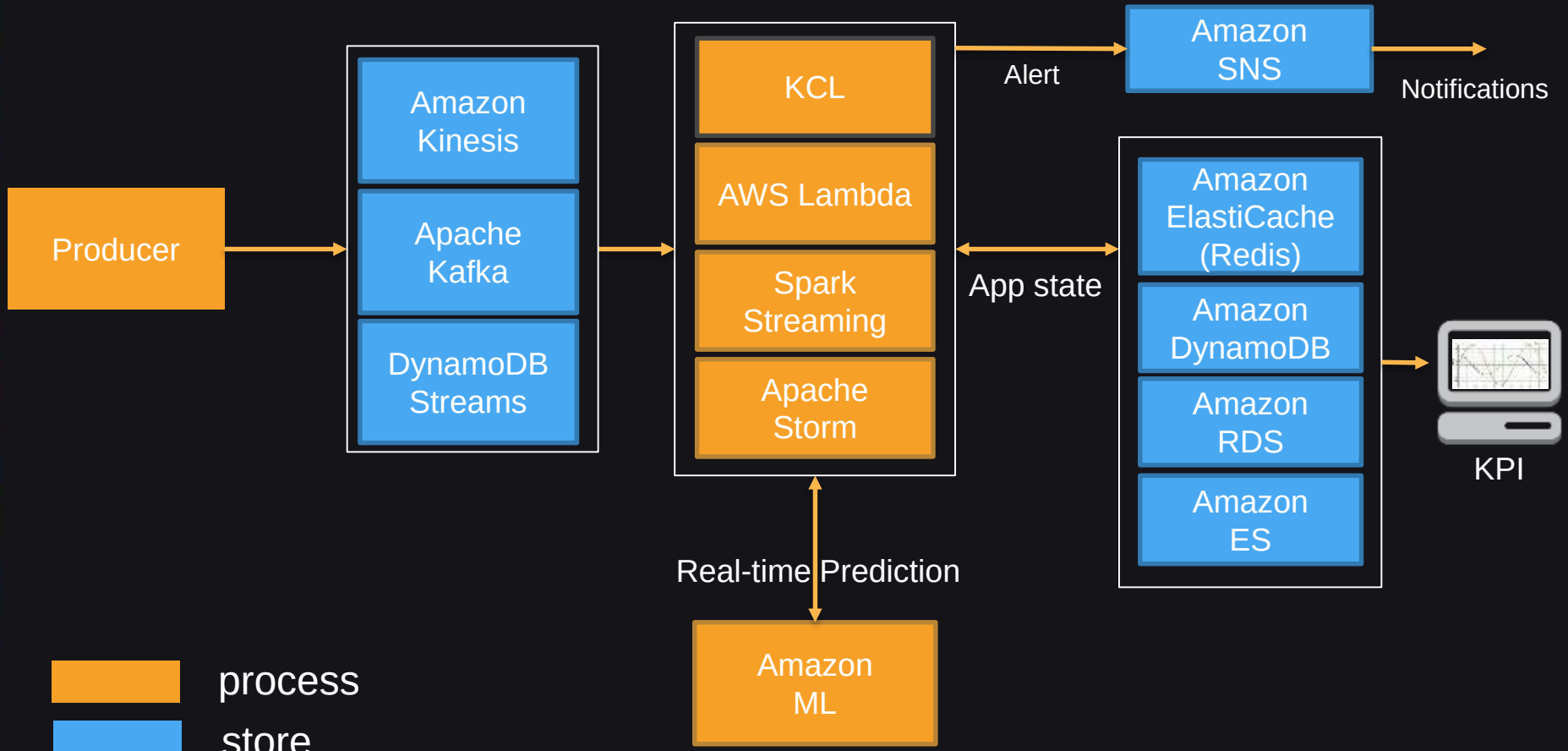
Processing Frameworks (KCL, Storm, Hive, Spark, etc.) Could Read from Multiple Data Stores

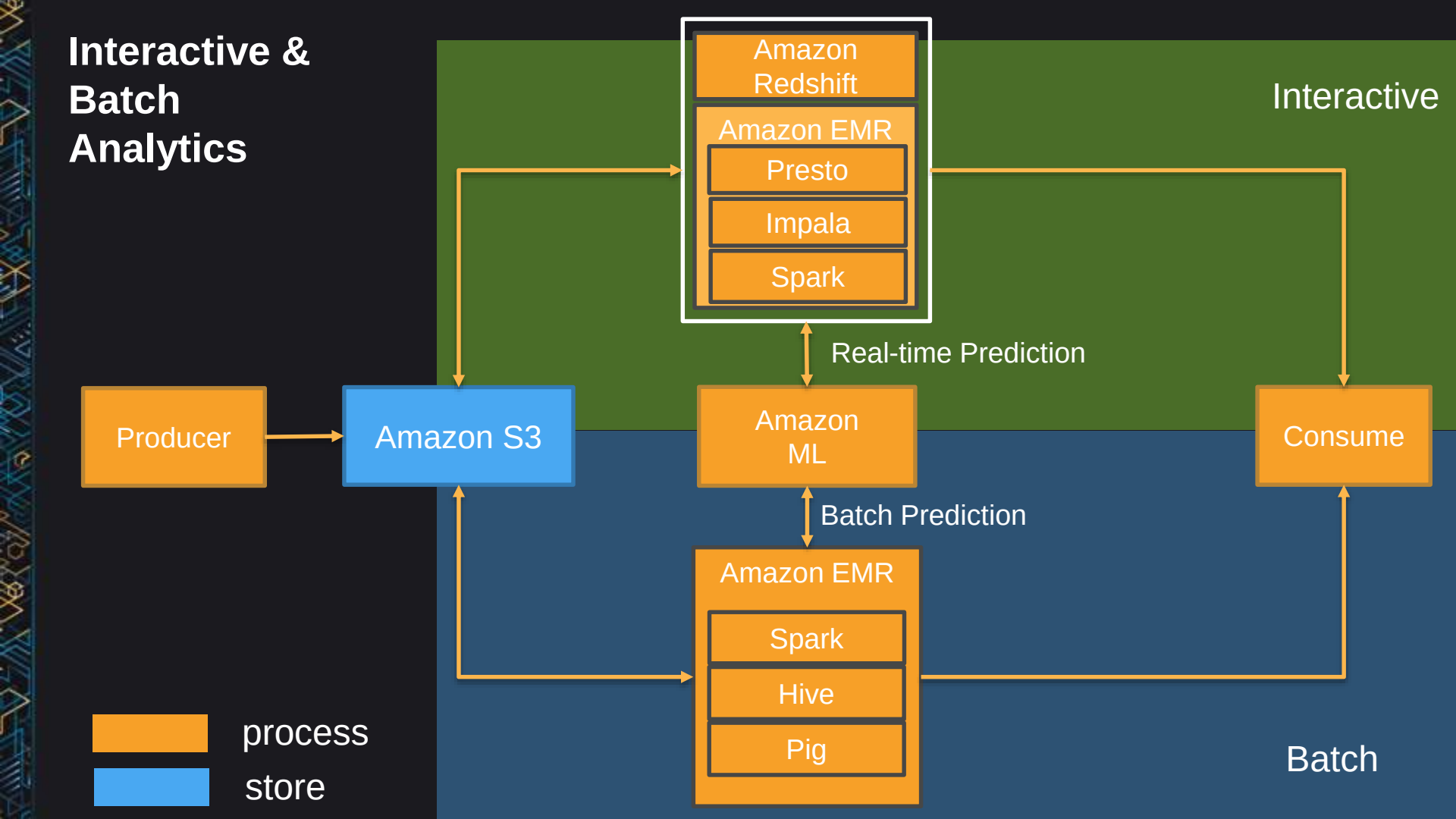


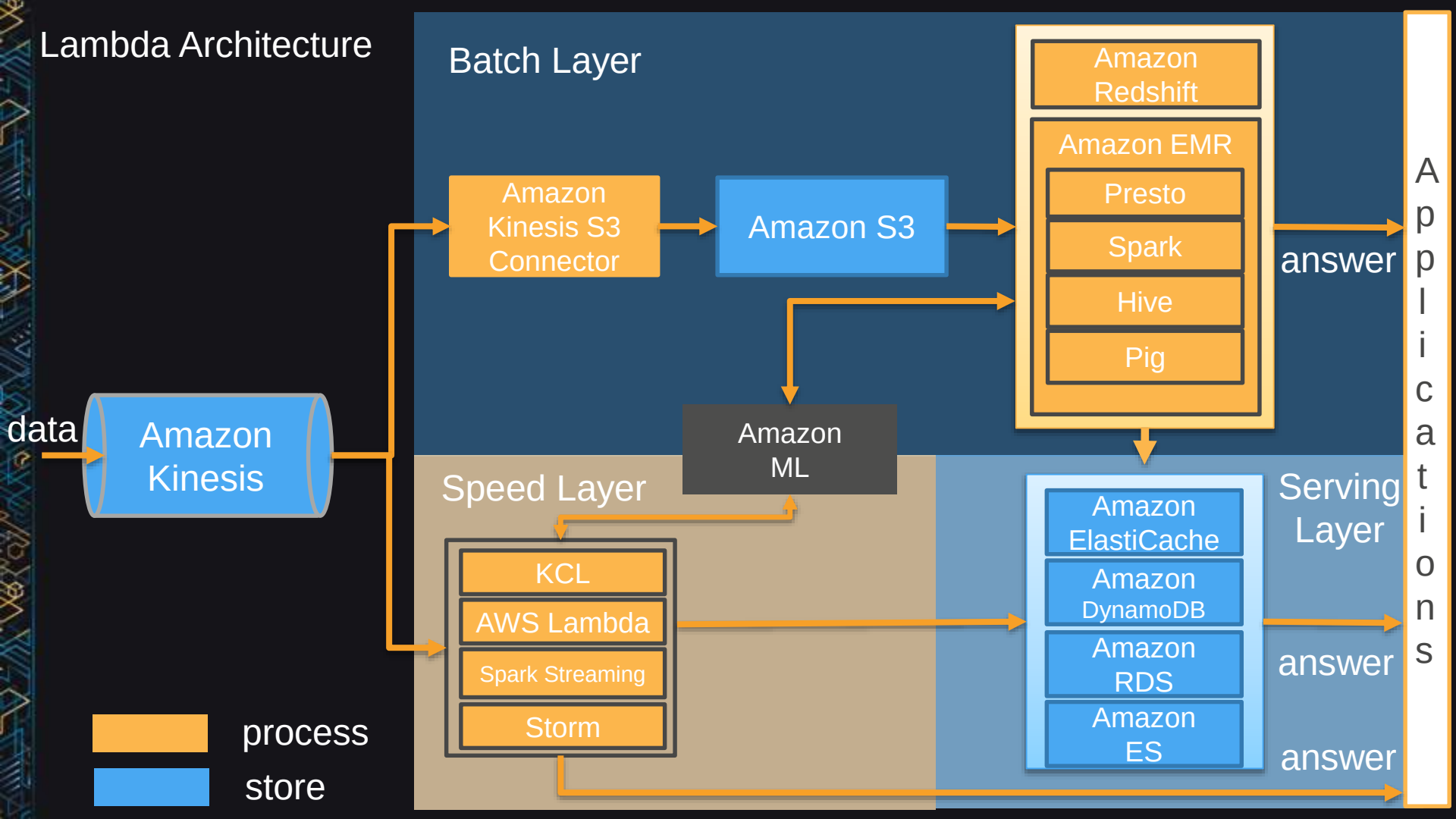
Data Temperature vs Processing Latency



Real-time Analytics




Interactive & Batch Analytics
Amazon Redshift
Amazon EMR
Presto
Impala
Spark
Interactive
Real-time Prediction
Producer
Amazon S3
Amazon ML
Consume
Batch Prediction
Amazon EMR
Spark
Hive
Pig
process
store
Batch


Lambda Architecture
Batch Layer
Amazon Redshift
Amazon EMR
Presto
Spark
Hive
Pig
Amazon Kinesis S3 Connector
Amazon S3
answer
Applications
data
Amazon Kinesis
Amazon ML
Speed Layer
KCL
AWS Lambda
Spark Streaming
Storm
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
Serving Layer
answer
answer
process
store

Summary

- Build decoupled “data bus”
  - Data → Store ↔ Process → Answers
- Use the right tool for the job
  - Latency, throughput, access patterns
- Use Lambda architecture ideas
  - Immutable (append-only) log, batch/speed/serving layer
- Leverage AWS managed services
  - No/low admin
- Be cost conscious
  - Big data ≠ big cost

Remember to complete your evaluations!

Thank you!