Big Data Architectural Patterns and Best Practices on AWS

Keith Steward, Ph.D.
Specialist (EMR) Solution Architect
Amazon Web Services

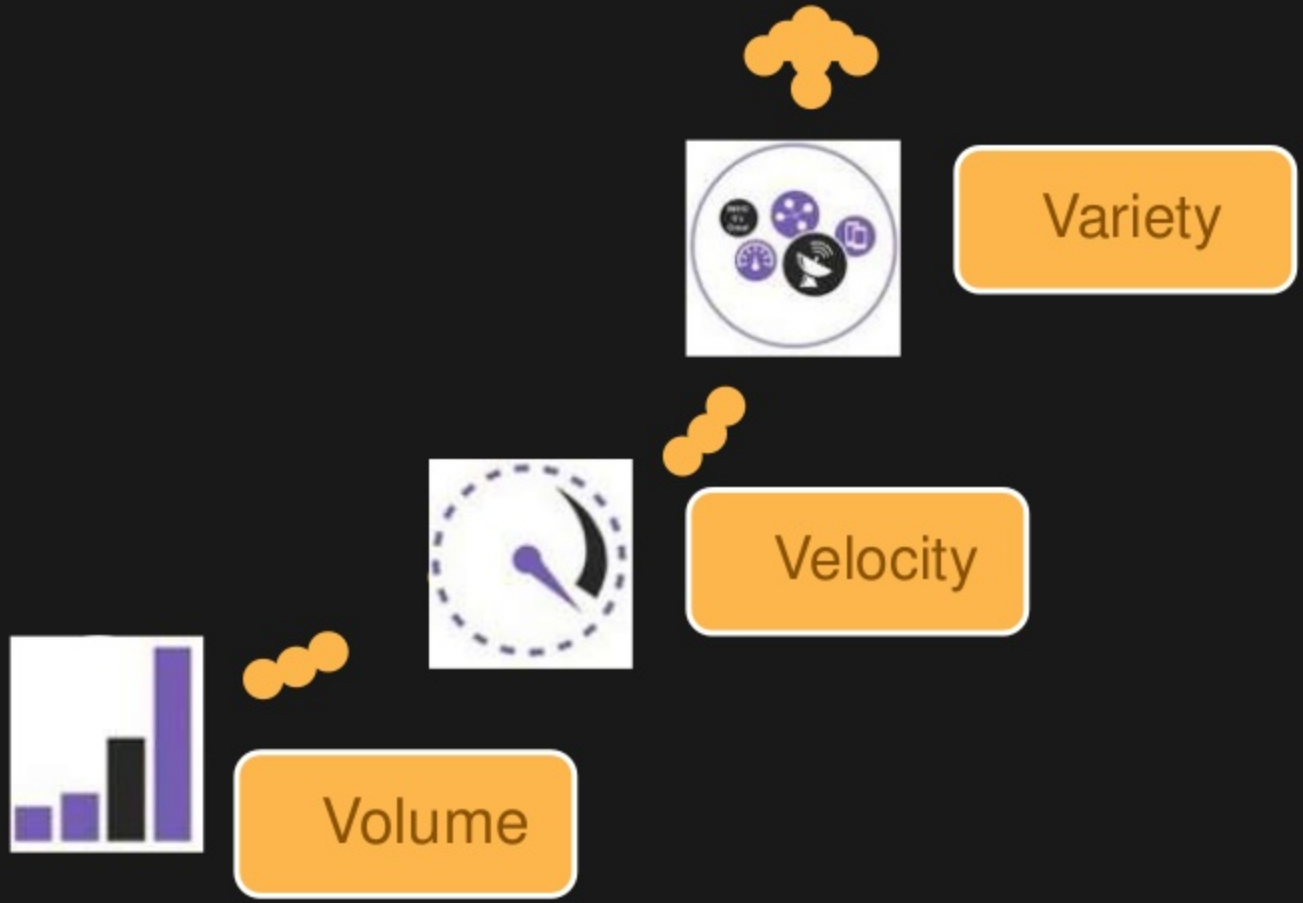
October 2016

amazon web services

What we'll cover:

1. Big data challenges
2. How to simplify big data processing
3. What technologies should you use?
   - Why?
   - How?
4. Reference architecture
5. Design patterns

Ever Increasing Big Data



Big Data Evolution

Batch → Real-time → Prediction

Reports

Alerts

Forecasts

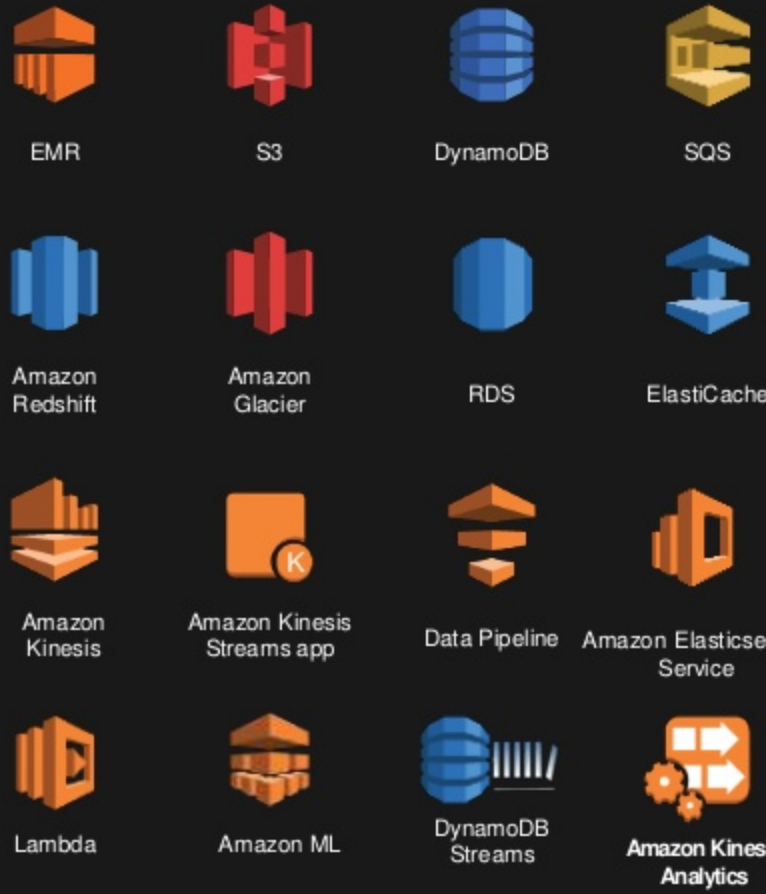
Cloud Services Evolution

Virtual machines

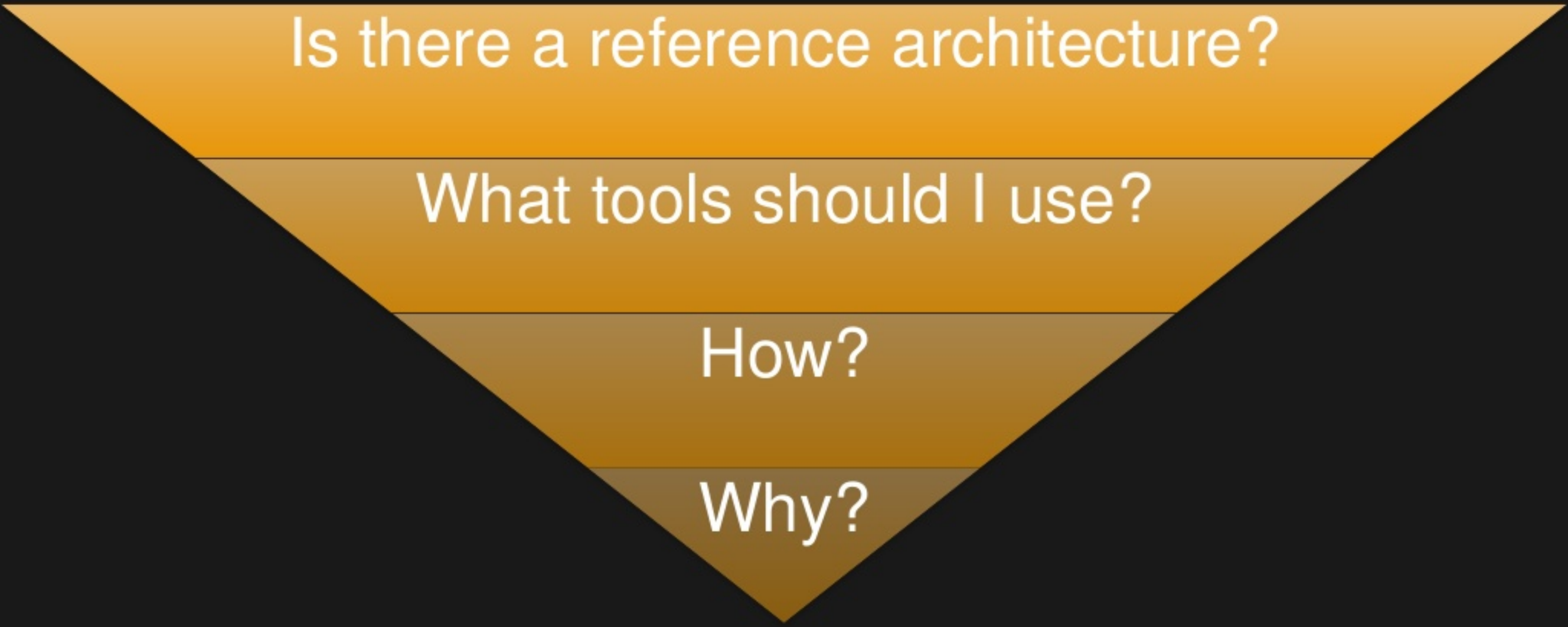
Managed services

Serverless

Plethora of Tools



Big Data Challenges



Architectural Principles

Decoupled “data bus”

- Data → Store → Process → Store → Analyze → Answers

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Leverage AWS managed services

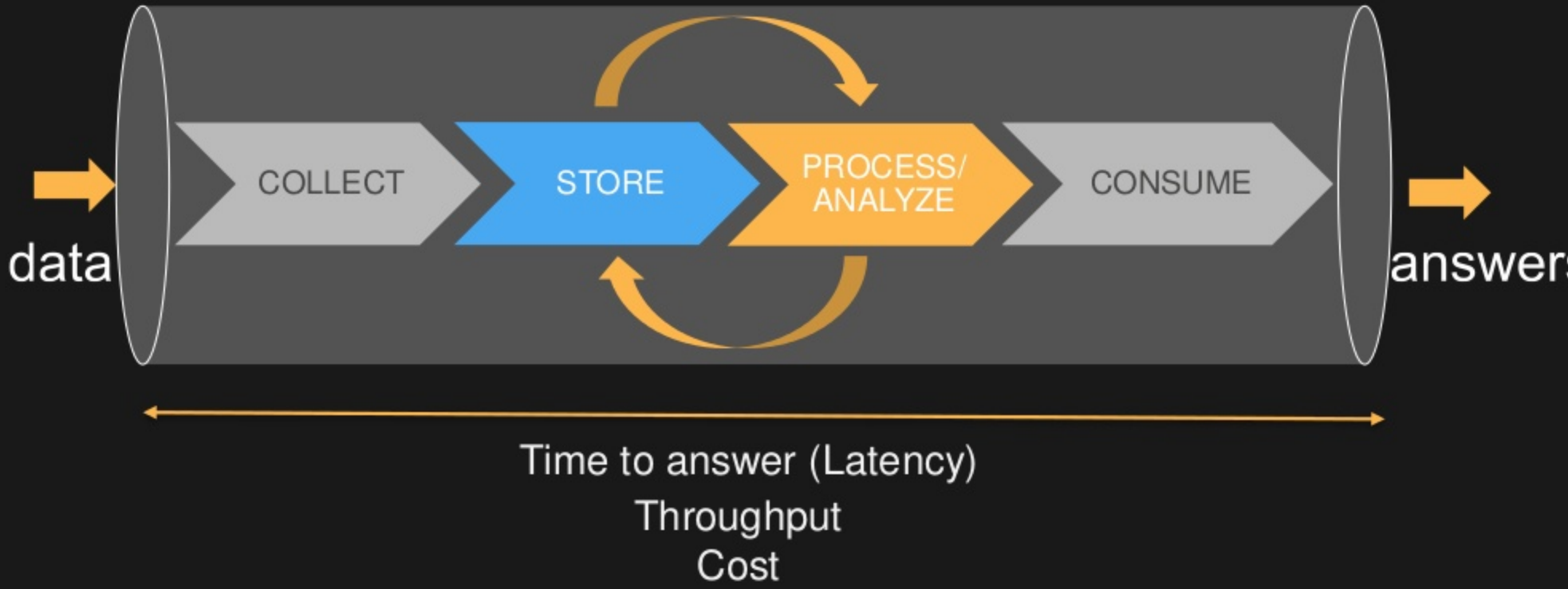
- Scalable / elastic, available, reliable, secure, no / low admin

Use Lambda architecture ideas

- Immutable (append-only) log, batch / real-time / serving layer

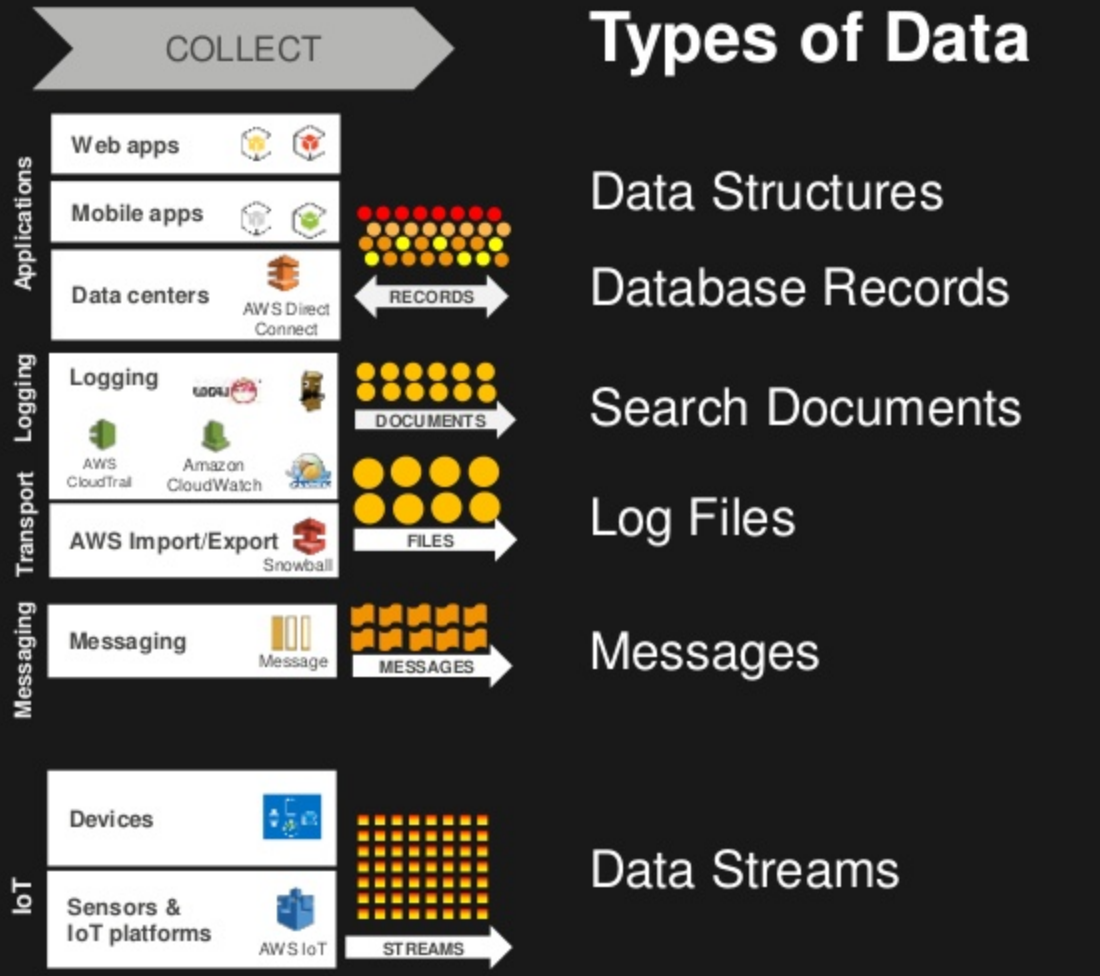
Big data ≠ big cost

Conceptual Framework to Simplify Big Data Processing




COLLECT

Types of Data



Data Structures

Database Records

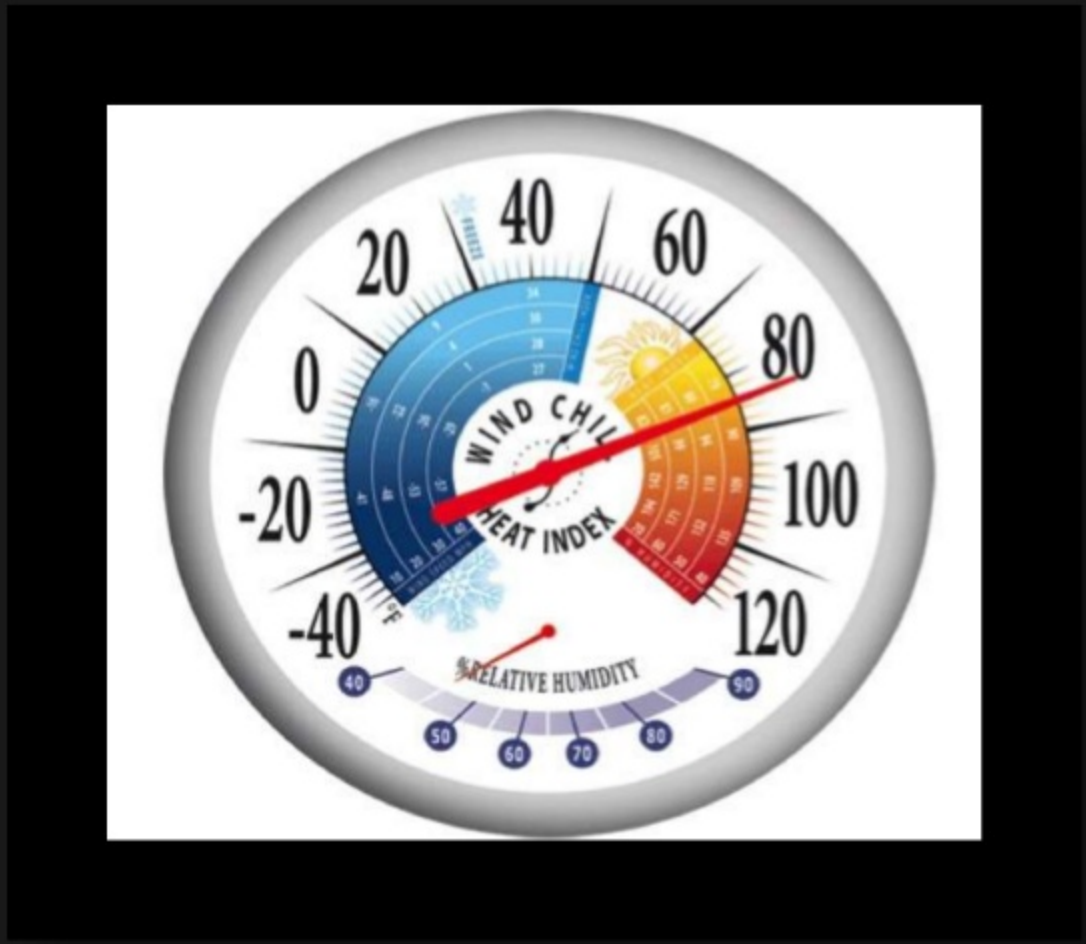
Search Documents

Log Files

Messages

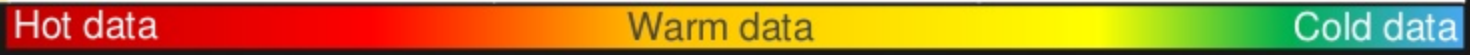
Data Streams

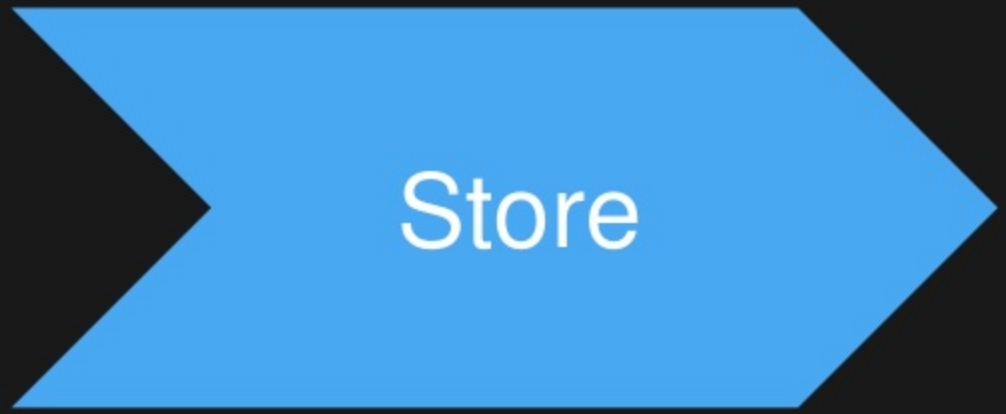
What Is the Temperature of Your Data ?

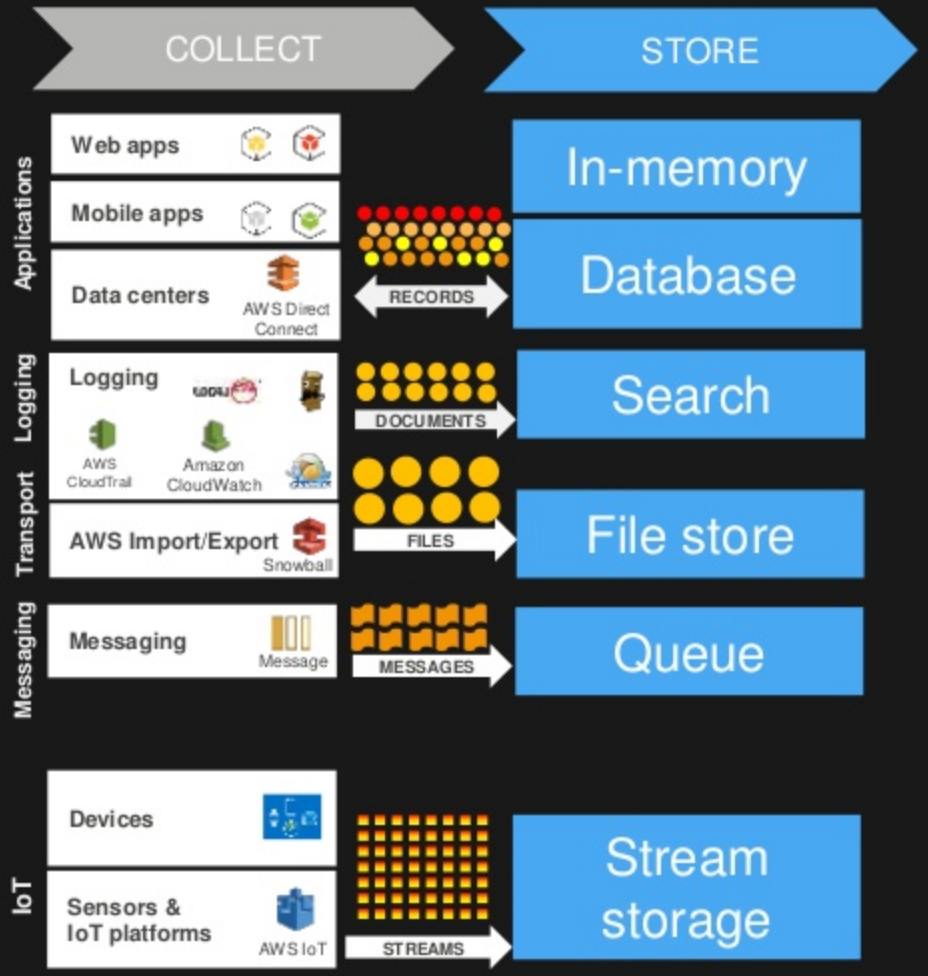


Data Characteristics: Hot, Warm, Cold

	Hot	Warm	Cold
Volume	MB–GB	GB–TB	PB–EB
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	$$-$	$-¢¢	¢




Store



Types of Data Stores

Caches, Data Structure Servers

SQL & NoSQL Databases

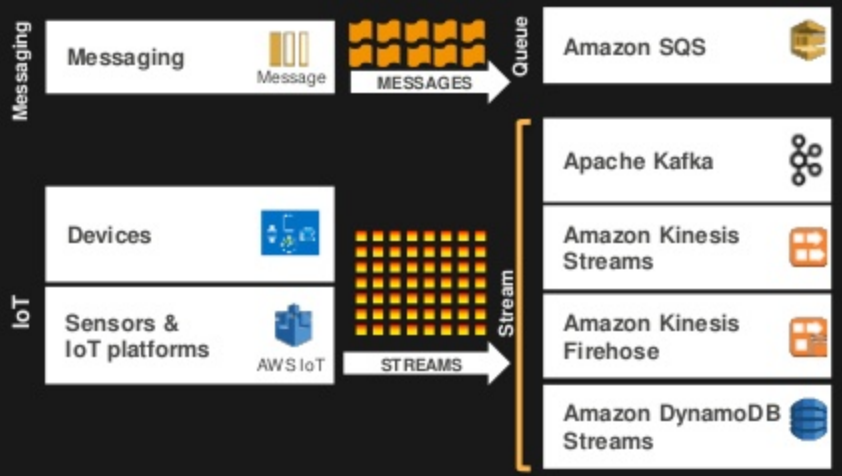
Search Engines

File Systems

Message Queues

Pub/Sub Message Queues

COLLECT STORE



Message & Stream Storage

Amazon SQS

- Managed message queue service

Apache Kafka

- High throughput distributed messaging system

Amazon Kinesis Streams

- Managed stream storage + processing

Amazon Kinesis Firehose

- Managed data delivery

Amazon DynamoDB

- Managed NoSQL database
- Tables can be stream-enabled

Why Stream Storage?

Decouple producers & consumers

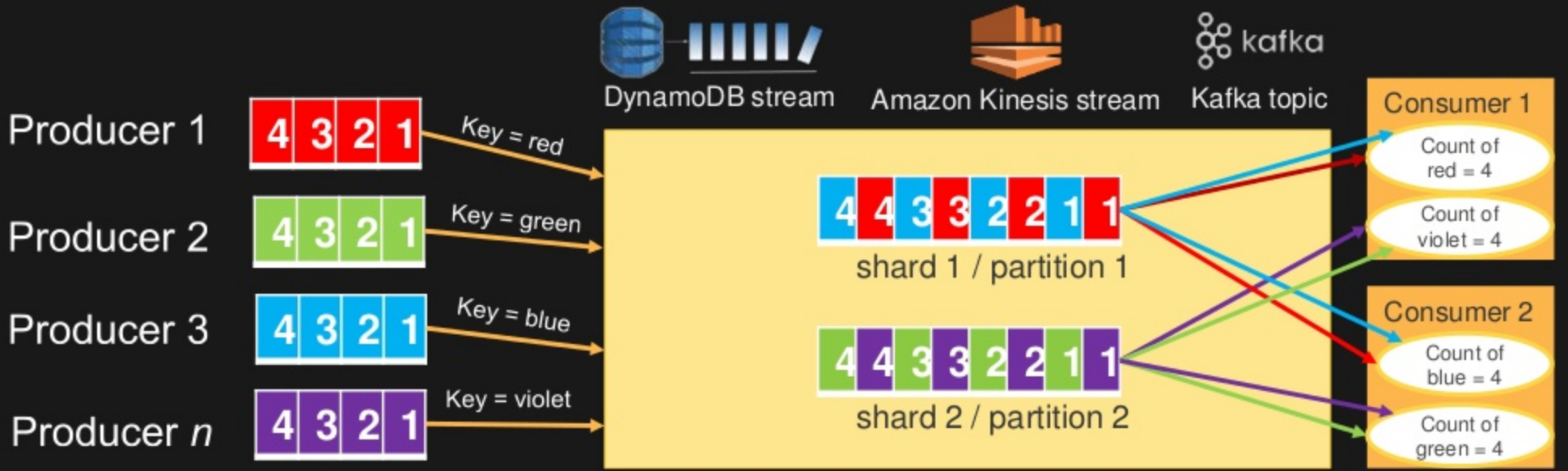
Persistent buffer

Collect multiple streams

Preserve client ordering

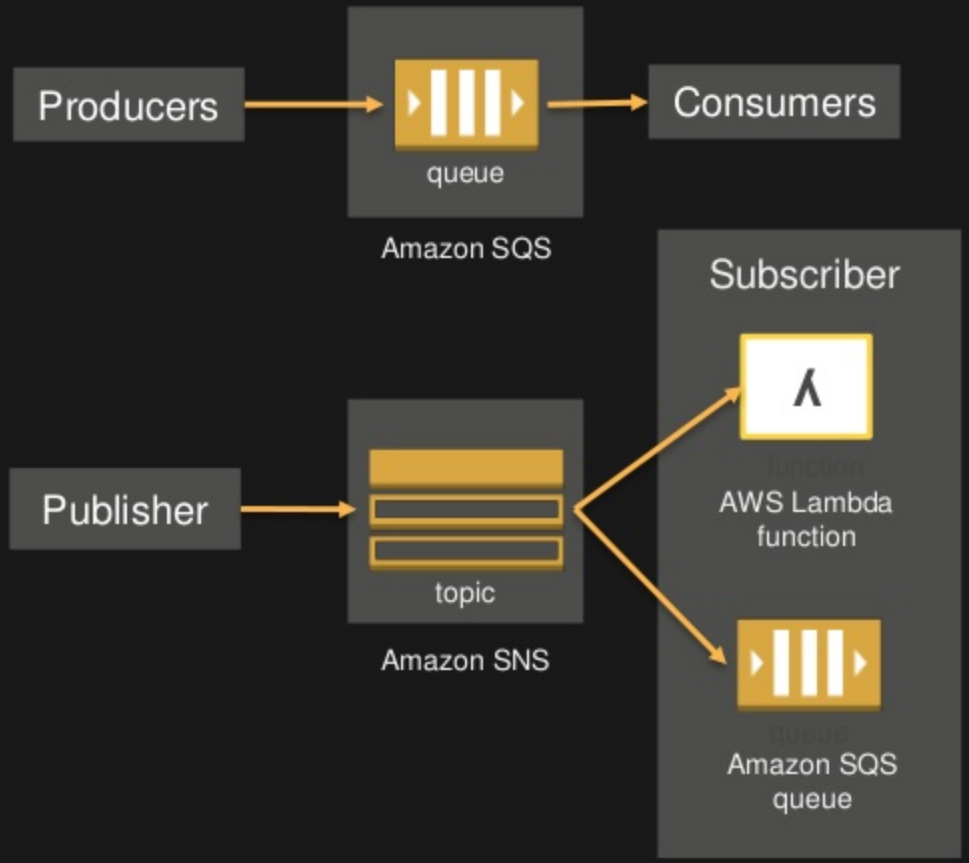
Parallel consumption

Streaming MapReduce



What About Messaging?

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



What Stream / Message Storage Should I Use?

	Amazon DynamoDB Streams	Amazon Kinesis Streams	Amazon Kinesis Firehose	Apache Kafka	Amazon SQS
AWS managed	Yes	Yes	Yes	No	Yes
Guaranteed ordering	Yes	Yes	No	Yes	No
Delivery (deduping)	Exactly-once	At-least-once	At-least-once	At-least-once	At-least-once
Data retention period	24 hours	7 days	N/A	Configurable	14 days
Availability	3 AZ	3 AZ	3 AZ	Configurable	3 AZ
Scale / throughput	No limit / ~ table IOPS	No limit / ~ shards	No limit / automatic	No limit / ~ nodes	No limits / automatic
Parallel clients	Yes	Yes	No	Yes	No
Stream MapReduce	Yes	Yes	N/A	Yes	N/A
Row/object size	400 KB	1 MB	Destination row/object size	Configurable	256 KB
Cost	Higher (table cost)	Low	Low	Low (+admin)	Low-medium

Hot — Warm

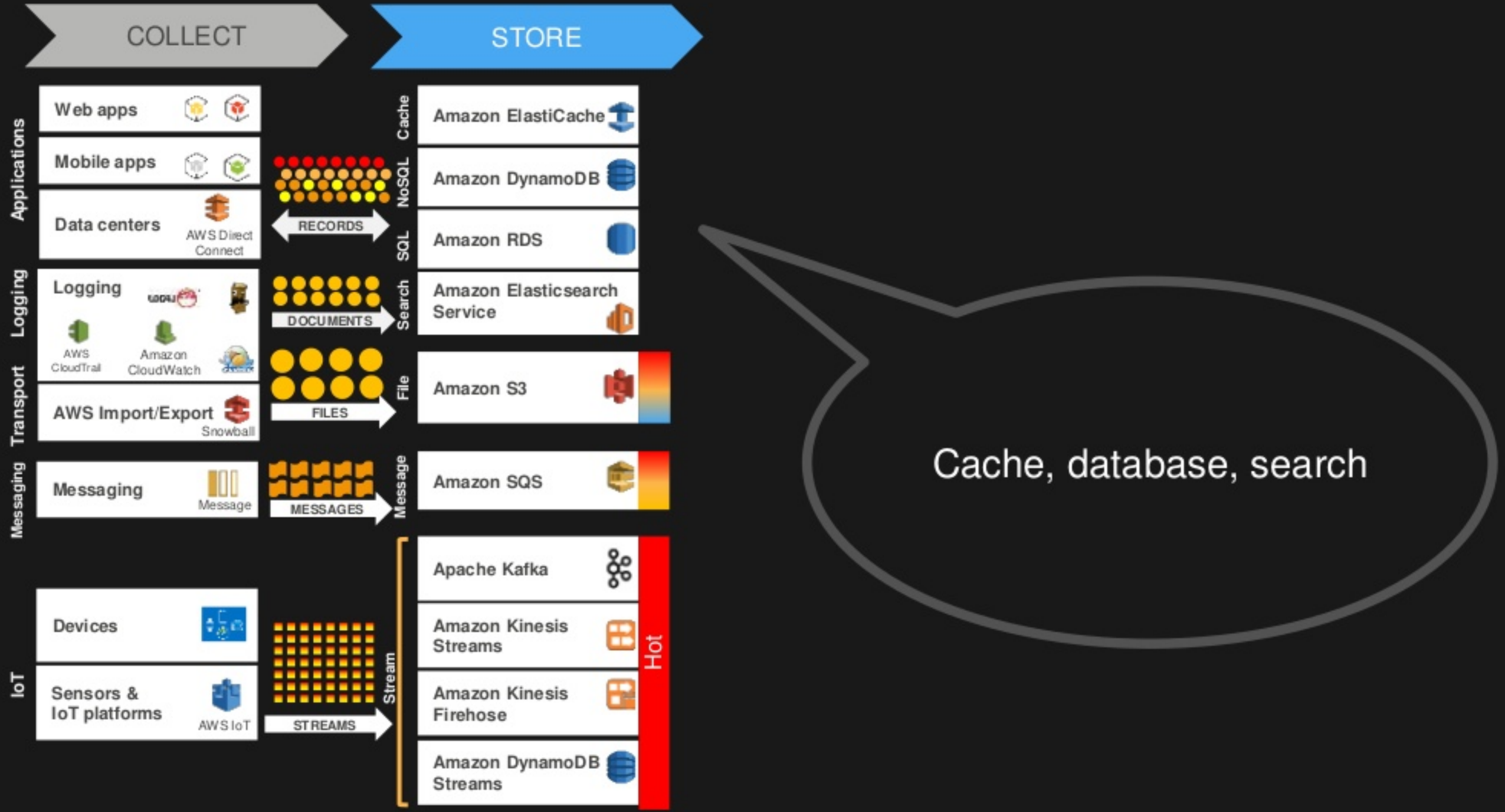
File Storage

Why is Amazon S3 Good for Big Data?

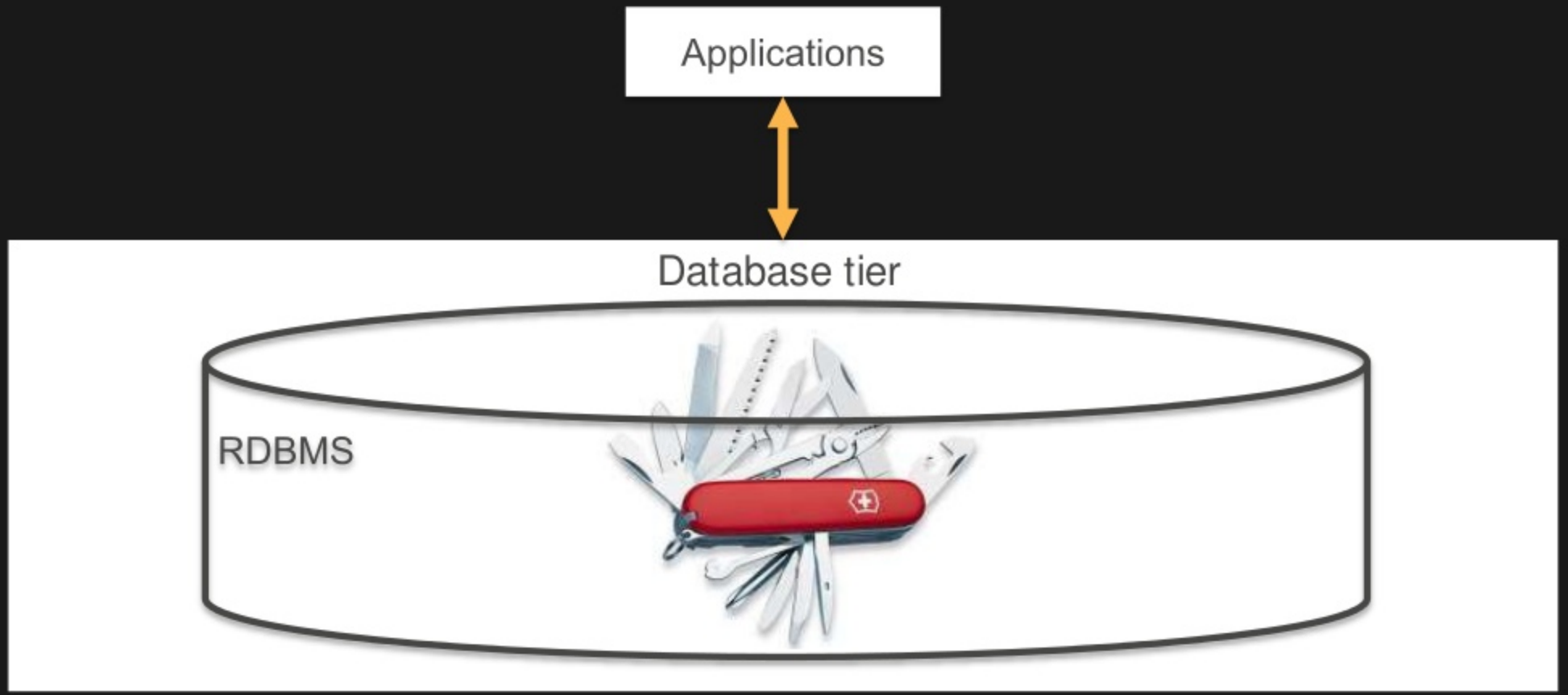
- Native support by big data frameworks (Spark, Hive, Presto, etc.)
- No need to run compute clusters for storage (unlike HDFS)
- Supports transient Hadoop clusters & Amazon EC2 Spot Instances
- Multiple distinct (Spark, Hive, Presto) clusters can use the same data
- Elastic: Unlimited number of objects
- 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?

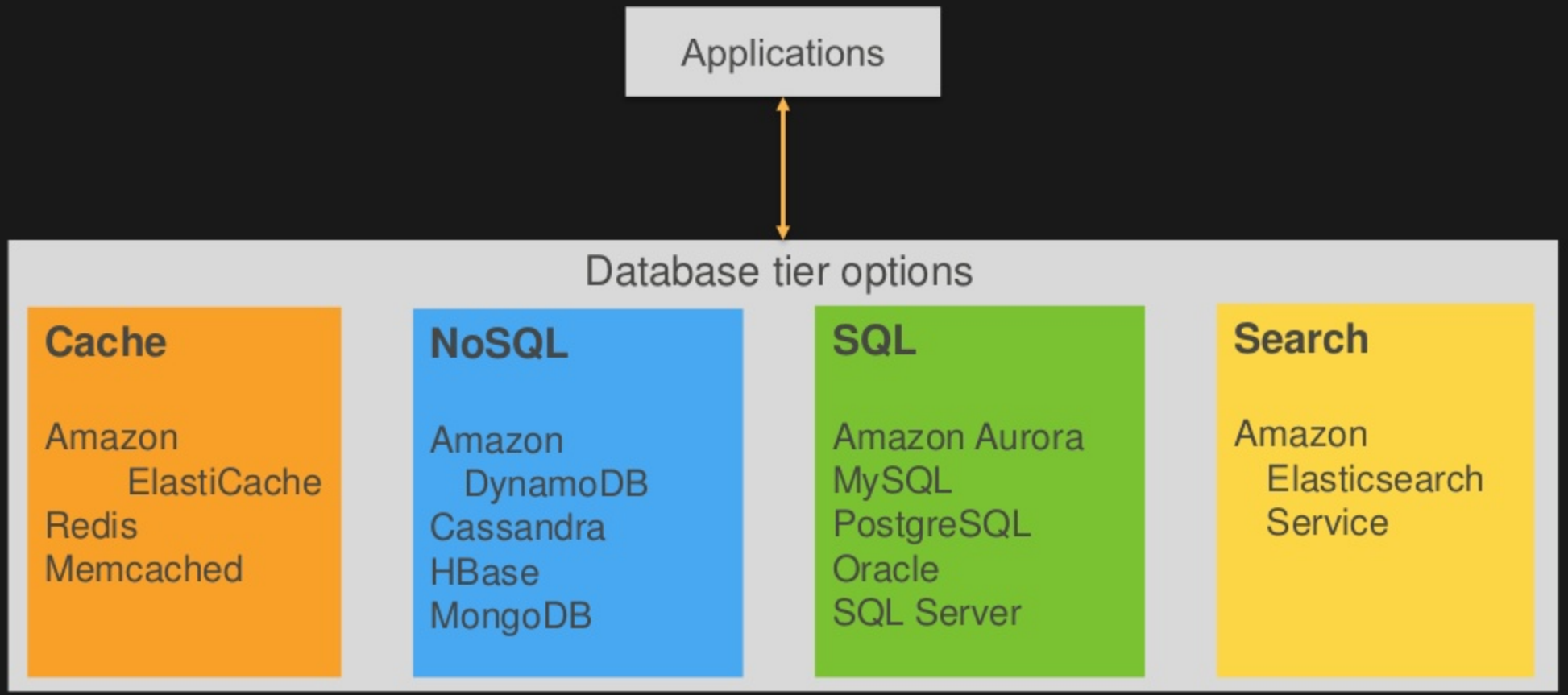
- Use HDFS for very frequently accessed (hot) data
- Use Amazon S3 Standard for frequently accessed data
- Use Amazon S3 Standard – IA for infrequently accessed data
- Use Amazon Glacier for archiving cold data


COLLECT
STORE
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
Transport
AWS Import/Export
Snowball
Messaging
Messaging
Message
IoT
Devices
Sensors & IoT platforms
AWS IoT
RECORDS
DOCUMENTS
FILES
MESSAGES
STREAMS
Cache
NoSQL
SQL
Search
File
Message
Stream
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon Elasticsearch Service
Amazon S3
Amazon SQS
Apache Kafka
Amazon Kinesis Streams
Amazon Kinesis Firehose
Amazon DynamoDB Streams
Hot
Cache, database, search

Database Anti-pattern



Best Practice - Use the Right Tool for the Job



What Data Store Should I Use?

Data structure? → Fixed schema, JSON, key-value

Access patterns? → Store data in 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


Low
Structure
High
S3
Amazon Glacier
NoSQL
Search
In-memory
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?

	Amazon ElastiCache	Amazon DynamoDB	Amazon RDS/Aurora	Amazon Elasticsearch	Amazon S3	Amazon Glacier
Average latency	ms	ms	ms, sec	ms,sec	ms,sec,min (~ size)	hrs
Typical data stored	GB	GB–TBs (no limit)	GB–TB (64 TB max)	GB–TB	MB–PB (no limit)	GB–PB (no limit)
Typical item size	B-KB	KB (400 KB max)	KB (64 KB max)	KB (2 GB max)	KB-TB (5 TB max)	GB (40 TB max)
Request Rate	High – very high	Very high (no limit)	High	High	Low – high (no limit)	Very low
Storage cost GB/month	$$	¢¢	¢¢	¢¢	¢	¢/10
Durability	Low - moderate	Very high	Very high	High	Very high	Very high
Availability	High 2 AZ	Very high 3 AZ	Very high 3 AZ	High 2 AZ	Very high 3 AZ	Very high 3 AZ

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. 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

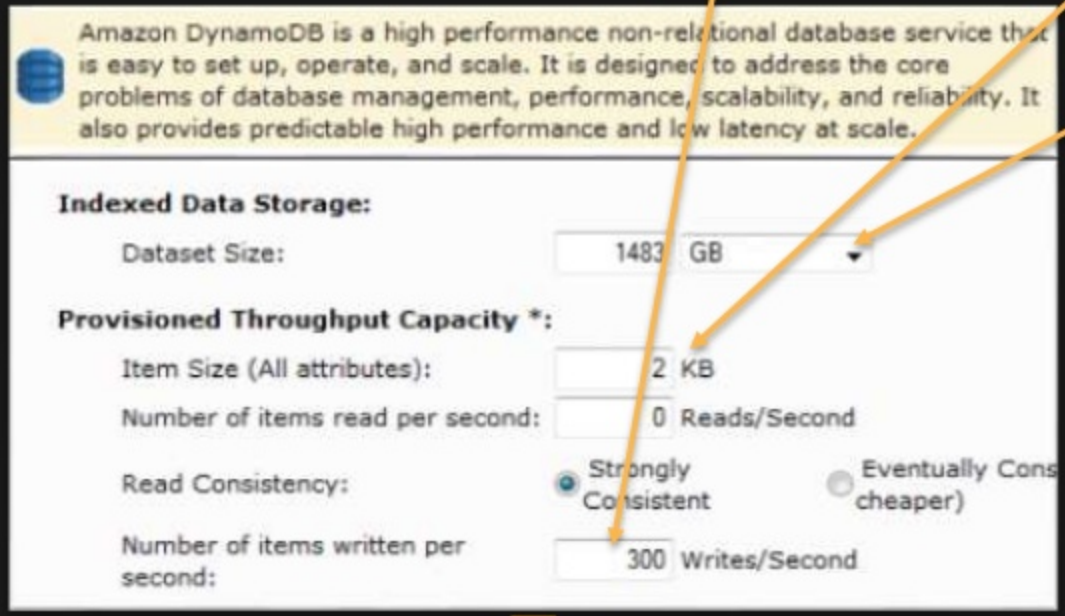
Example: Should I use Amazon S3 or Amazon DynamoDB?

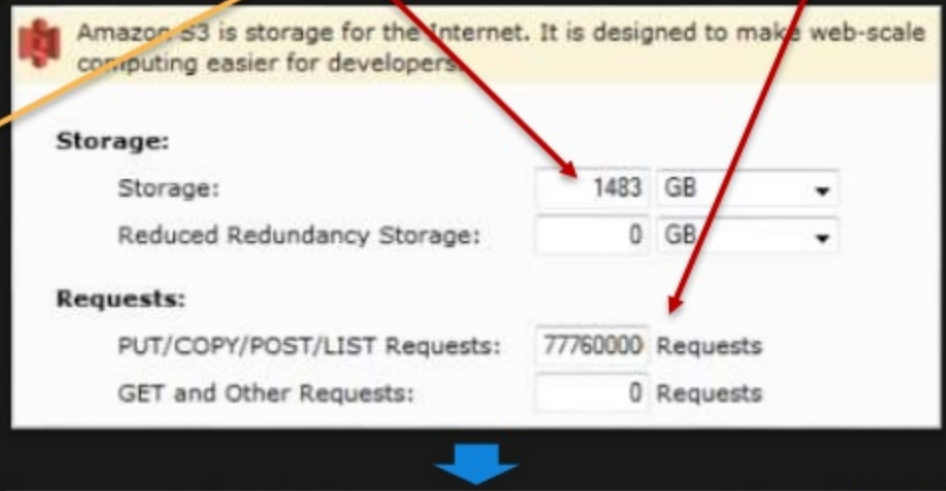
Simple Monthly Calculator

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

Amazon S3 or 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 Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	

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	

Cost Conscious Design

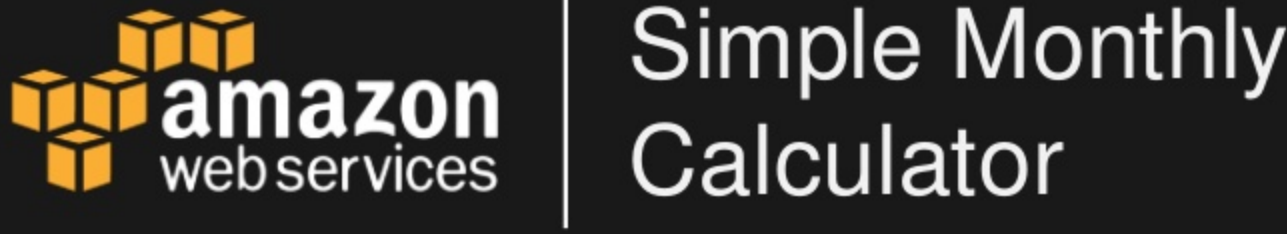
Example: Should I use Amazon S3 or Amazon DynamoDB?

"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 Consistent (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	


COLLECT
STORE
PROCESS / ANALYZE
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
Transport
AWS Import/Export
Snowball
Messaging
Messaging
Message
IoT
Devices
Sensors & IoT platforms
AWS IoT
RECORDS
DOCUMENTS
FILES
MESSAGES
STREAMS
Cache
NoSQL
SQL
Search
File
Message
Stream
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon Elasticsearch Service
Amazon S3
Amazon SQS
Apache Kafka
Amazon Kinesis Streams
Amazon Kinesis Firehose
Amazon DynamoDB Streams
Hot
Warm
Hot


PROCESS /
ANALYZE

Analytics Types & Frameworks

Batch

Takes minutes to hours

Example: Daily/weekly/monthly reports

Amazon EMR (MapReduce, Hive, Pig, Spark)

Interactive

Takes seconds

Example: Self-service dashboards

Amazon Redshift, Amazon EMR (Presto, Spark)

Message

Takes milliseconds to seconds

Example: Message processing

Amazon SQS applications on Amazon EC2

Stream

Takes milliseconds to seconds

Example: Fraud alerts, 1 minute metrics

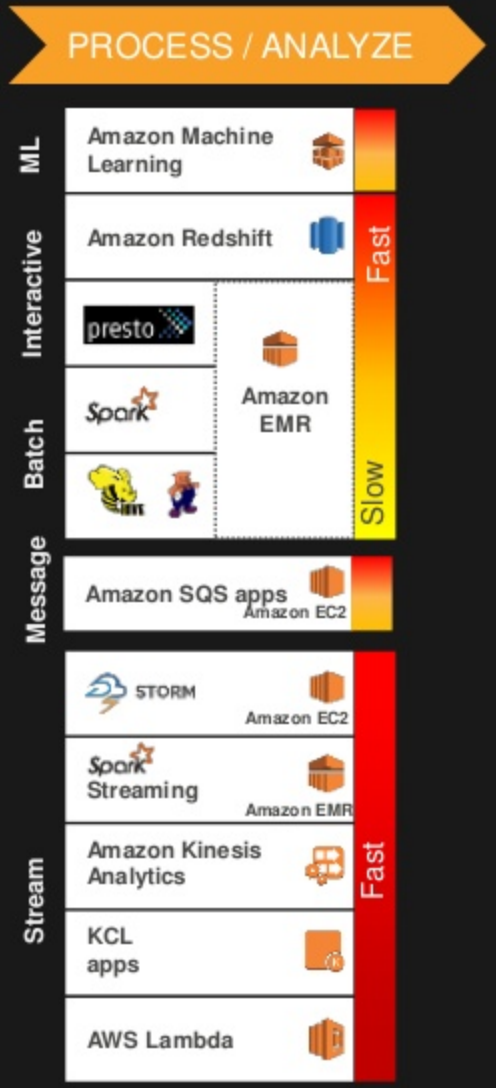
Amazon EMR (Spark Streaming), Amazon Kinesis Analytics, KCL, Storm, AWS Lambda

Machine Learning

Takes milliseconds to minutes

Example: Fraud detection, forecast demand

Amazon ML, Amazon EMR (Spark ML)



What Stream & Message Processing Technology Should I Use?

	Amazon EMR (Spark Streaming)	Apache Storm	KCL Application	Amazon Kinesis Analytics	AWS Lambda	Amazon SQS Application
AWS managed	Yes (Amazon EMR)	No (Do it yourself)	No (EC2 + Auto Scaling)	Yes	Yes	No (EC2 + Auto Scaling)
Serverless	No	No	No	Yes	Yes	No
Scale / throughput	No limits / ~ nodes	No limits / ~ nodes	No limits / ~ nodes	Up to 8 KPU / automatic	No limits / automatic	No limits / ~ nodes
Availability	Single AZ	Configurable	Multi-AZ	Multi-AZ	Multi-AZ	Multi-AZ
Programming languages	Java, Python, Scala	Almost any language via Thrift	Java, others via MultiLangDaemon	ANSI SQL with extensions	Node.js, Java, Python	AWS SDK languages (Java, .NET, Python, …)
Uses	Multistage processing	Multistage processing	Single stage processing	Multistage processing	Simple event-based triggers	Simple event based triggers
Reliability	KCL and Spark checkpoints	Framework managed	Managed by KCL	Managed by Amazon Kinesis Analytics	Managed by AWS Lambda	Managed by SQS Visibility Timeout

Fast

Which Analysis Tool Should I Use?

	Amazon Redshift	Amazon EMR		
		Presto	Spark	Hive
Use case	Optimized for Data Warehousing	Interactive query	General purpose (iterative ML, RT, ..)	Batch
Scale/throughput	~Nodes	~ Nodes		
Storage	Local Storage	Amazon S3, HDFS		
Optimization	Columnar storage, Data compression, and Zone maps	Framework dependent		
Metadata	Amazon Redshift Managed	Hive Meta-store		
BI tools supports	Yes (JDBC/ODBC)	Yes (JDBC/ODBC & Custom)		
Access controls	Users, Groups and Access Controls	Integration with LDAP		
UDF support	Yes (Scalar)	Yes		

Fast Slow

What About ETL?

Data Integration

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

ATTUNITY — Attunity CloudBeam

informatica — Informatica Cloud

Matillion — Matillion ETL for Redshift

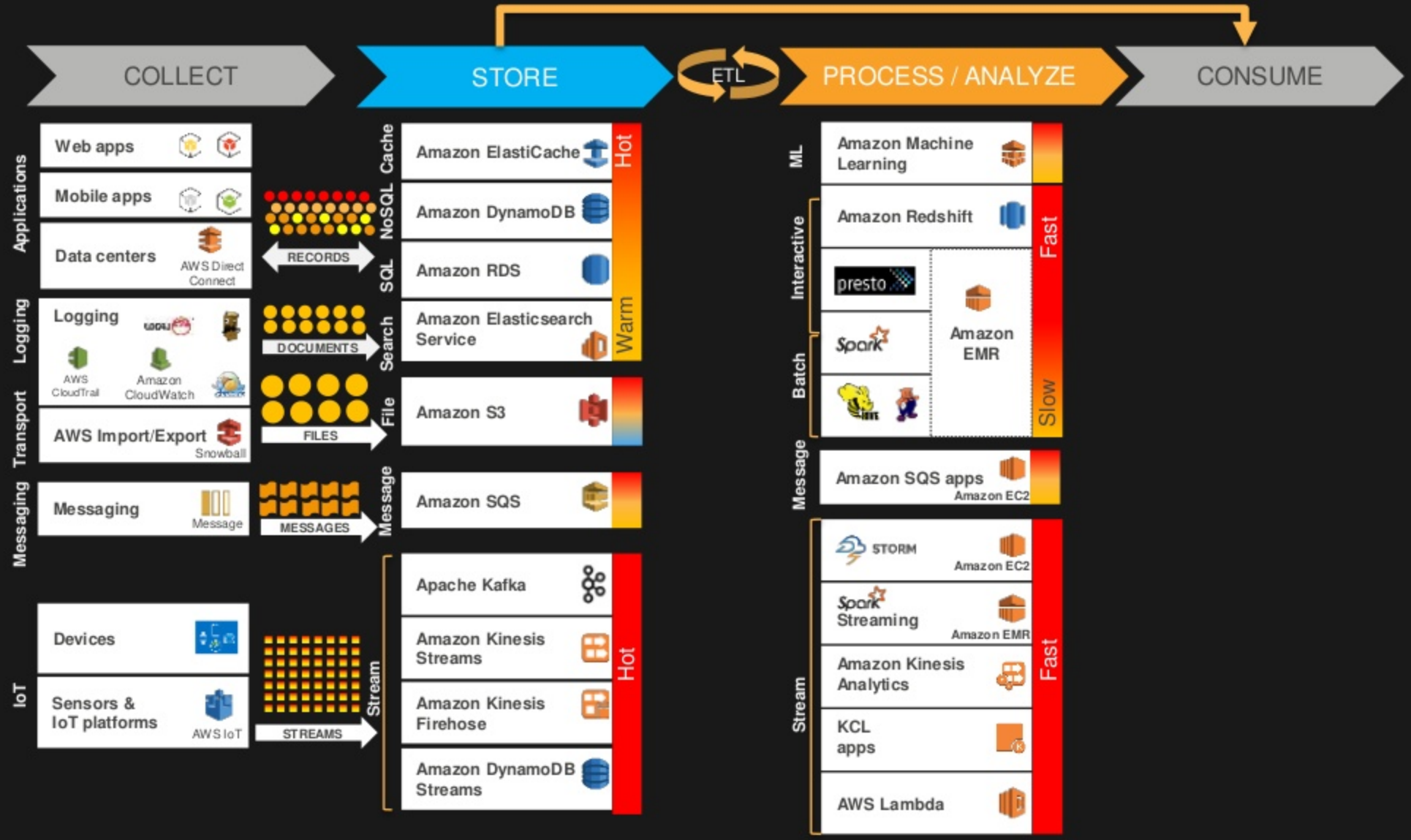
snapLogic — snapLogic

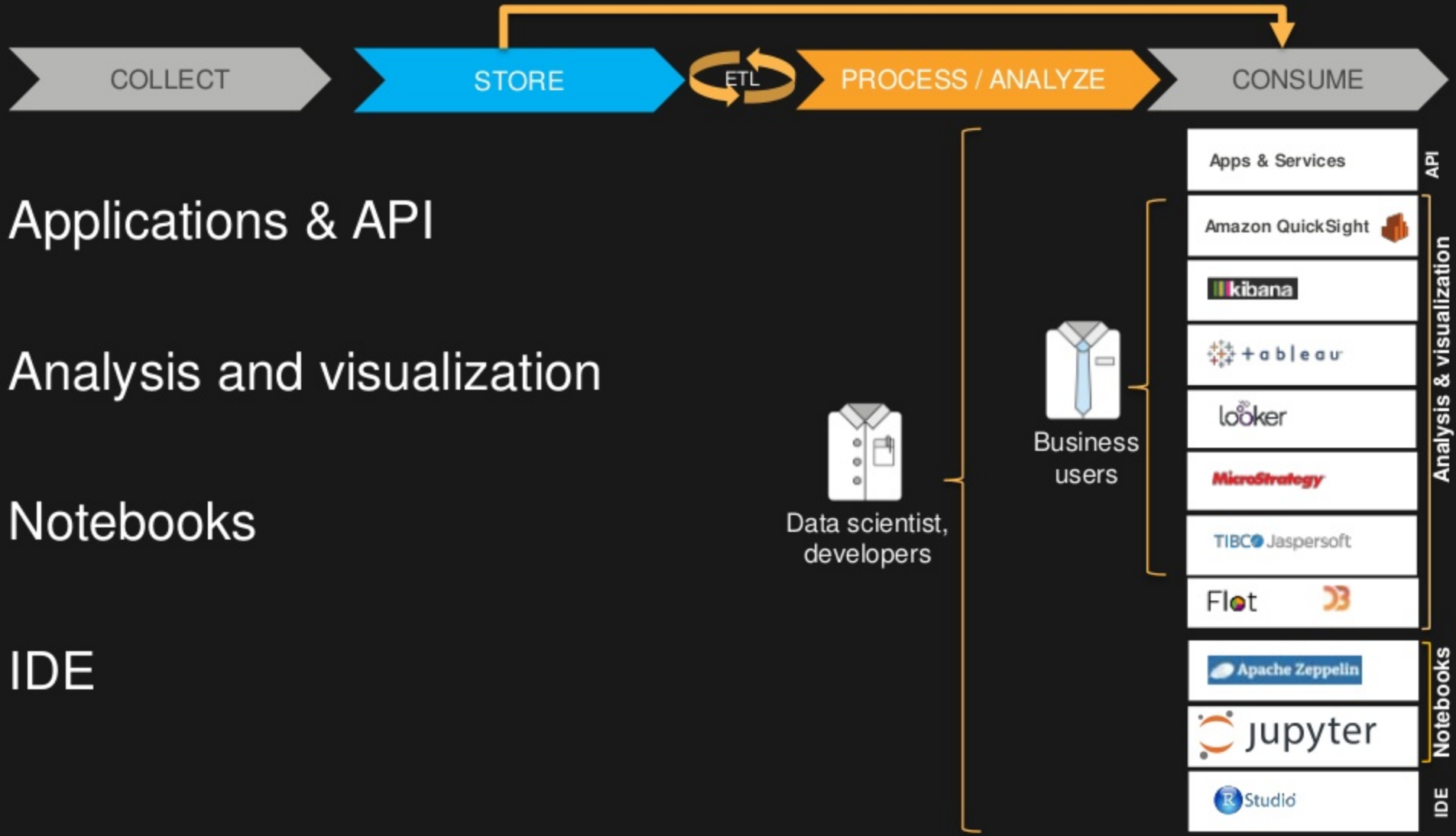
alteryx — alteryx

• • • • •

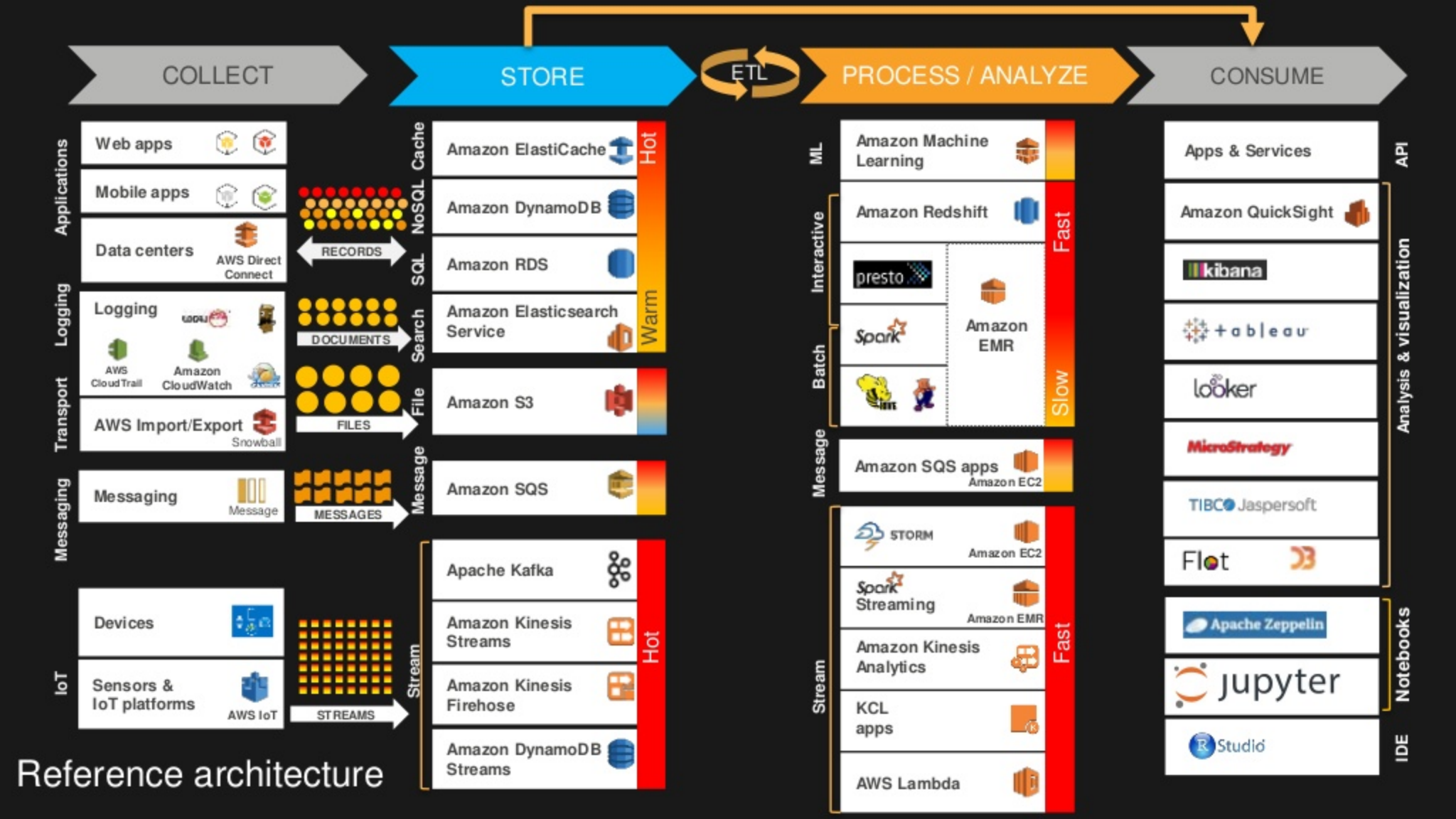
https://aws.amazon.com/big-data/partner-solutions/


CONSUME


COLLECT
STORE
ETL
PROCESS / ANALYZE
CONSUME
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
Transport
AWS Import/Export
Snowball
Messaging
Messaging
Message
IoT
Devices
Sensors & IoT platforms
AWS IoT
RECORDS
DOCUMENTS
FILES
MESSAGES
STREAMS
Cache
NoSQL
SQL
Search
File
Message
Stream
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon Elasticsearch Service
Amazon S3
Amazon SQS
Apache Kafka
Amazon Kinesis Streams
Amazon Kinesis Firehose
Amazon DynamoDB Streams
Hot
Warm
Hot
ML
Amazon Machine Learning
Interactive
Amazon Redshift
presto
Batch
Spark
Amazon EMR
Fast
Slow
Message
Amazon SQS apps
Amazon EC2
Stream
STORM
Amazon EC2
Spark Streaming
Amazon EMR
Amazon Kinesis Analytics
KCL apps
AWS Lambda
Fast


COLLECT
STORE
ETL
PROCESS / ANALYZE
CONSUME
Applications & API
Analysis and visualization
Notebooks
IDE
Data scientist, developers
Business users
Apps & Services
Amazon QuickSight
kibana
tableau
looker
MicroStrategy
TIBCO Jaspersoft
Flot
Apache Zeppelin
jupyter
Studio
API
Analysis & visualization
Notebooks
IDE

Putting It All Together


COLLECT
STORE
ETL
PROCESS / ANALYZE
CONSUME
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
Transport
AWS Import/Export
Snowball
Messaging
Messaging
Message
IoT
Devices
Sensors & IoT platforms
AWS IoT
RECORDS
DOCUMENTS
FILES
MESSAGES
STREAMS
Cache
NoSQL
SQL
Search
File
Message
Stream
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon Elasticsearch Service
Amazon S3
Amazon SQS
Apache Kafka
Amazon Kinesis Streams
Amazon Kinesis Firehose
Amazon DynamoDB Streams
Hot
Warm
Hot
ML
Amazon Machine Learning
Interactive
Amazon Redshift
presto
Batch
Spark
Amazon EMR
Fast
Slow
Message
Amazon SQS apps
Amazon EC2
Stream
STORM
Amazon EC2
Spark Streaming
Amazon EMR
Amazon Kinesis Analytics
KCL apps
AWS Lambda
Fast
API
Apps & Services
Analysis & visualization
Amazon QuickSight
kibana
+ableau
looker
MicroStrategy
TIBCO Jaspersoft
Flot
Notebooks
Apache Zeppelin
jupyter
IDE
RStudio
Reference architecture

Design Patterns

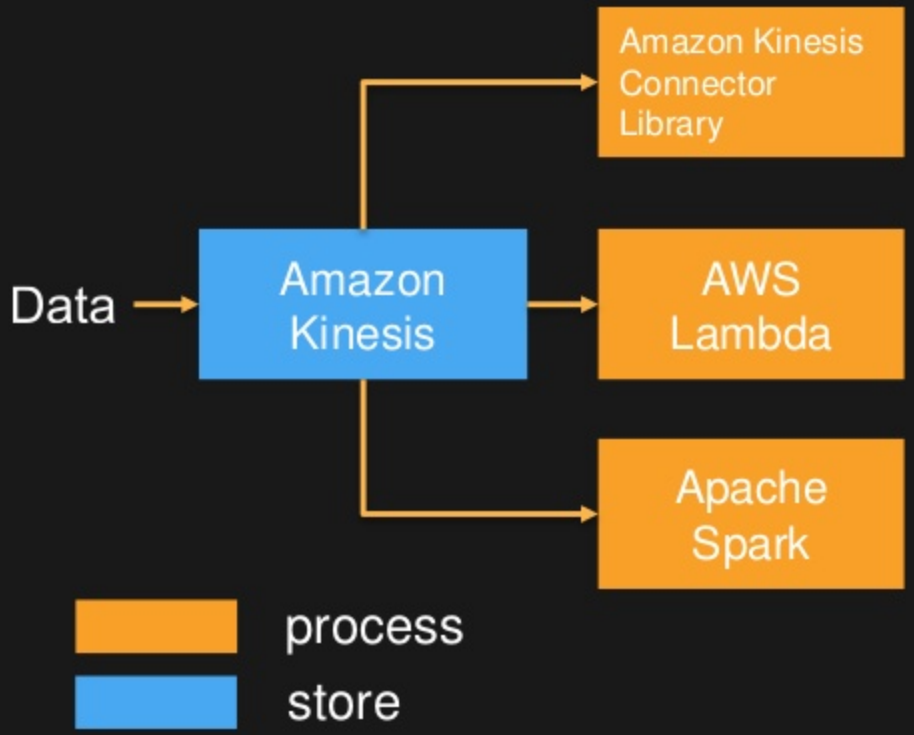
Primitive: Decoupled “Data Bus”

Storage decoupled from processing

Multiple stages

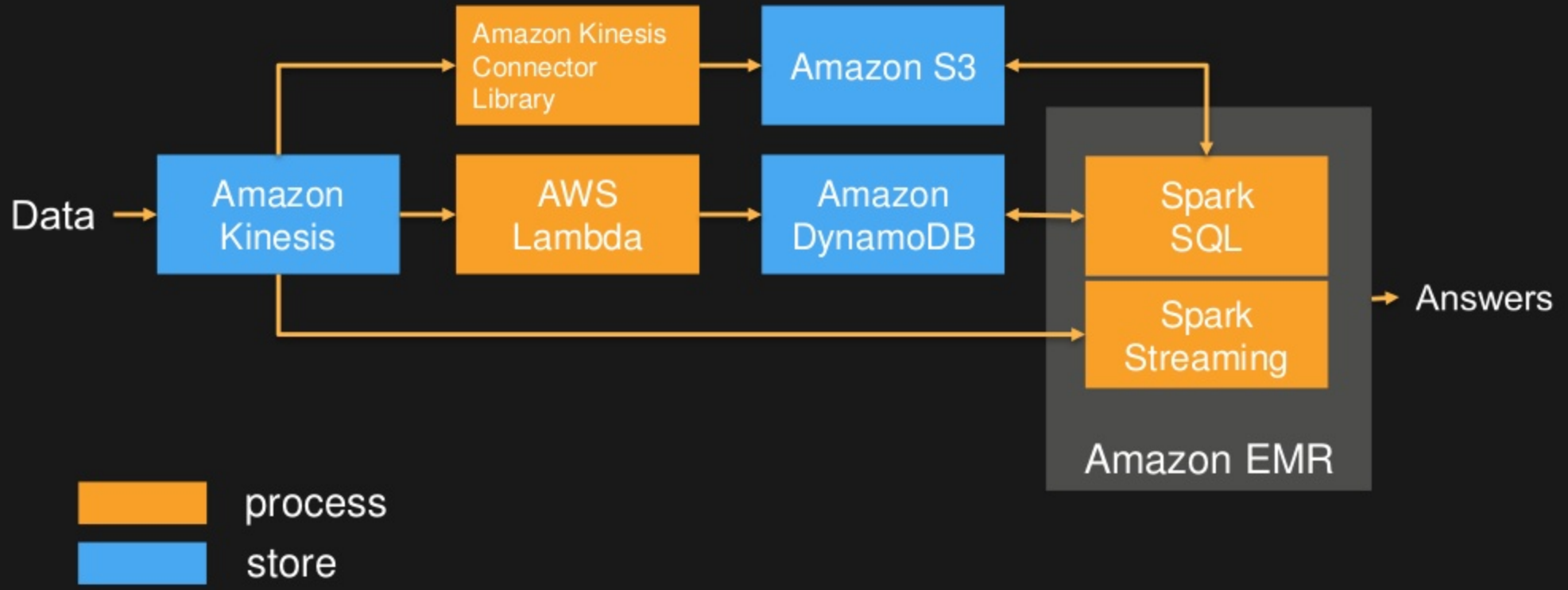
Primitive: Pub/Sub

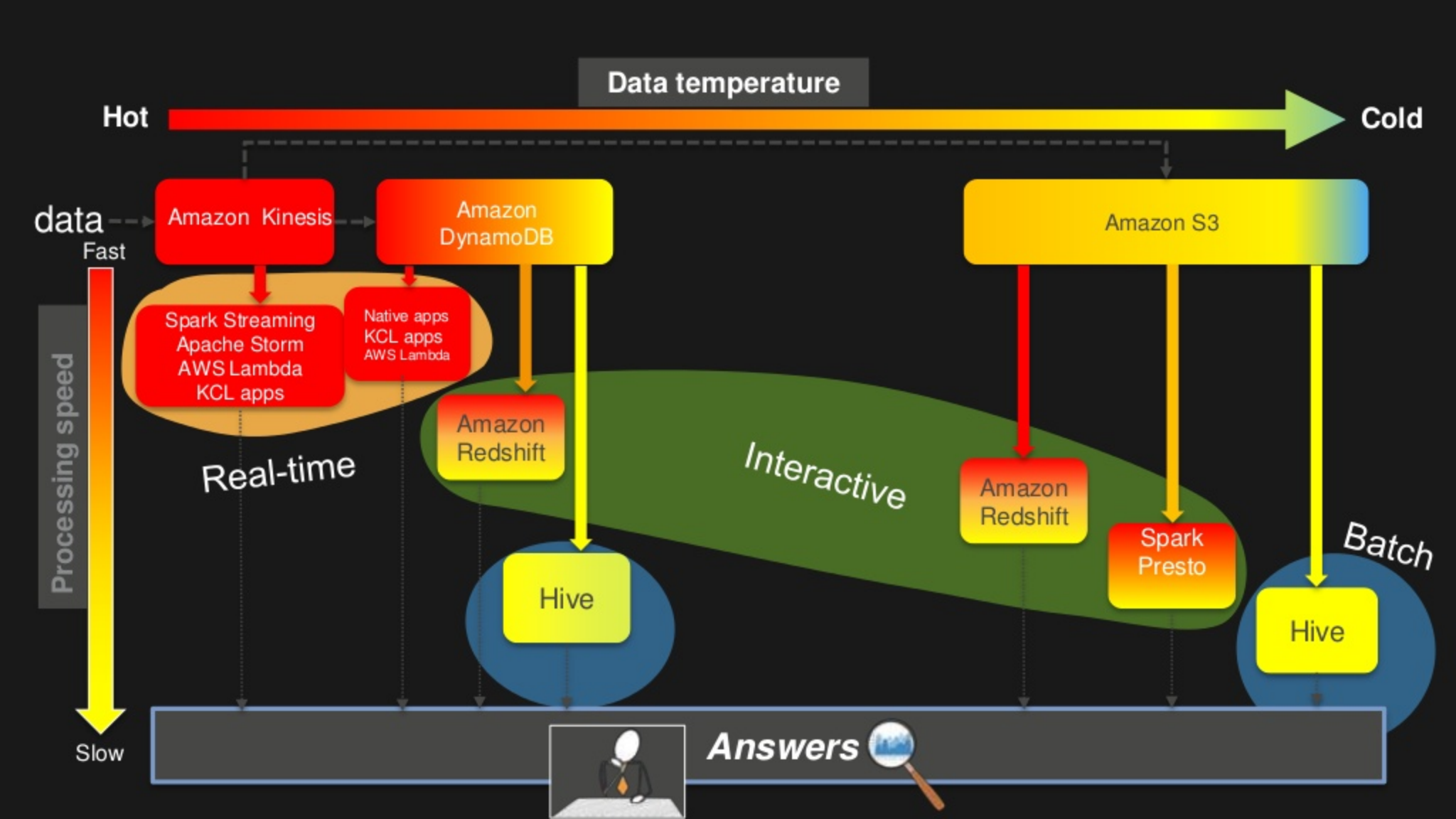
Parallel stream consumption/processing



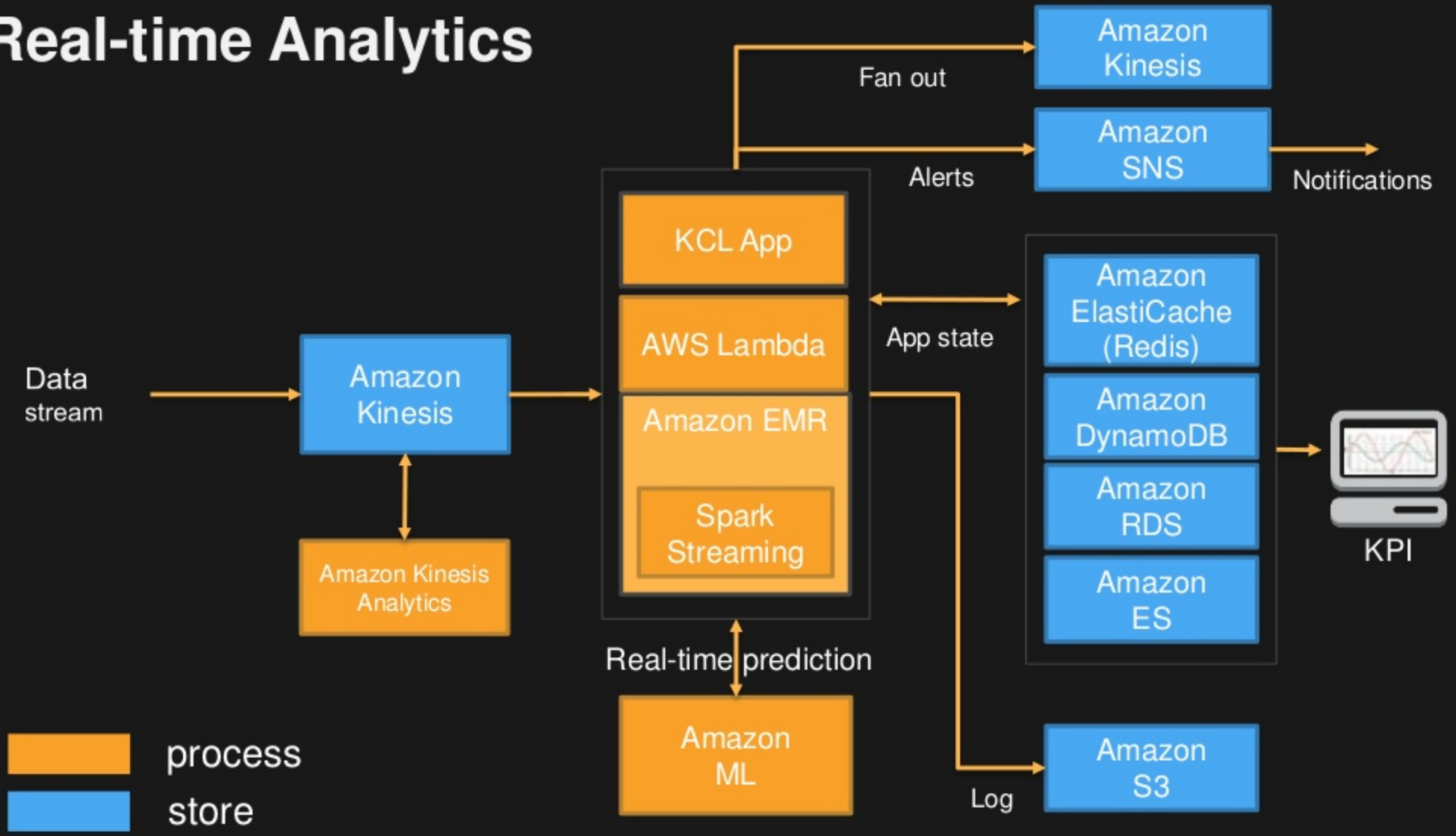
Primitive:

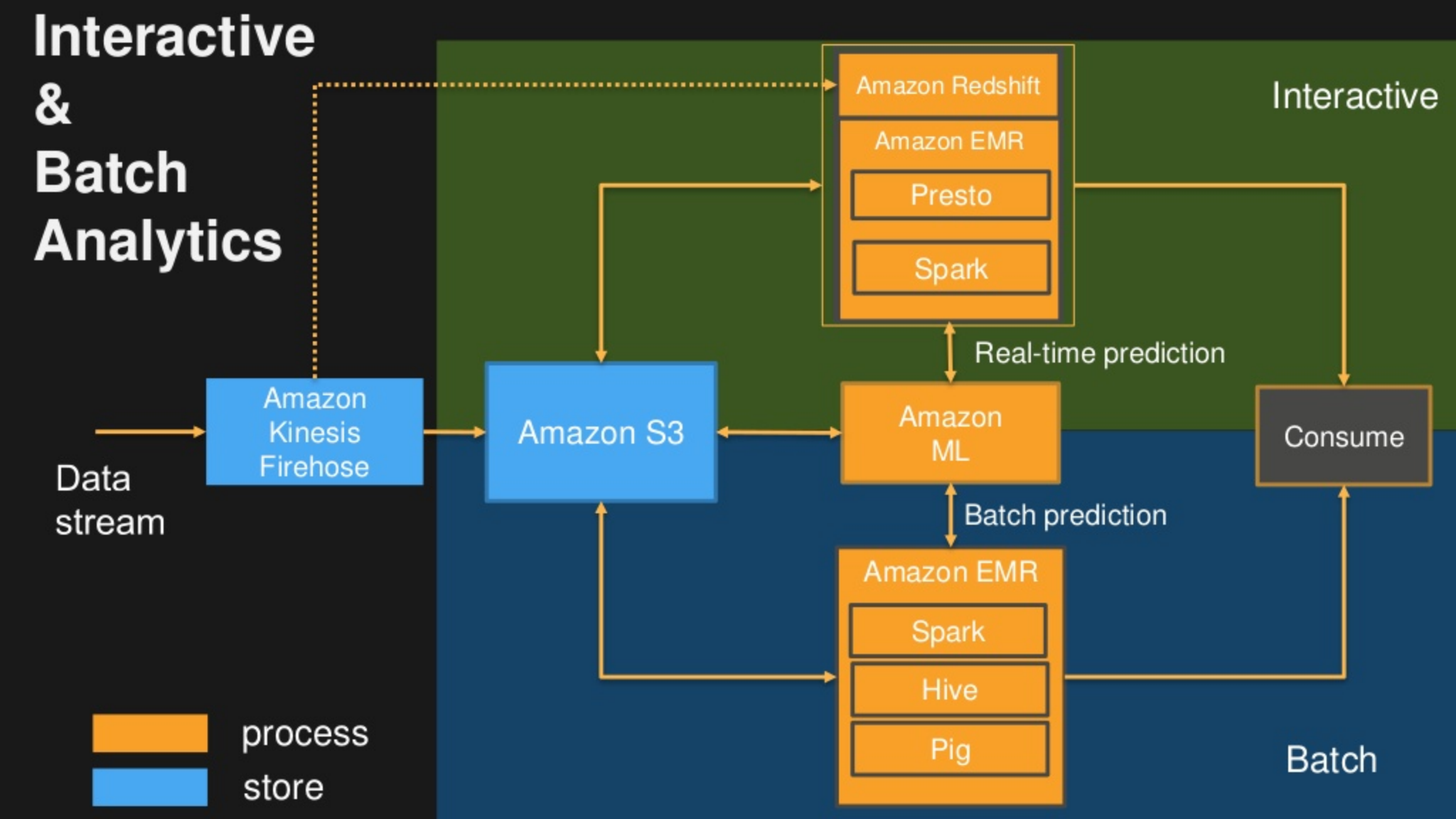
Analysis framework read from or write to multiple data stores



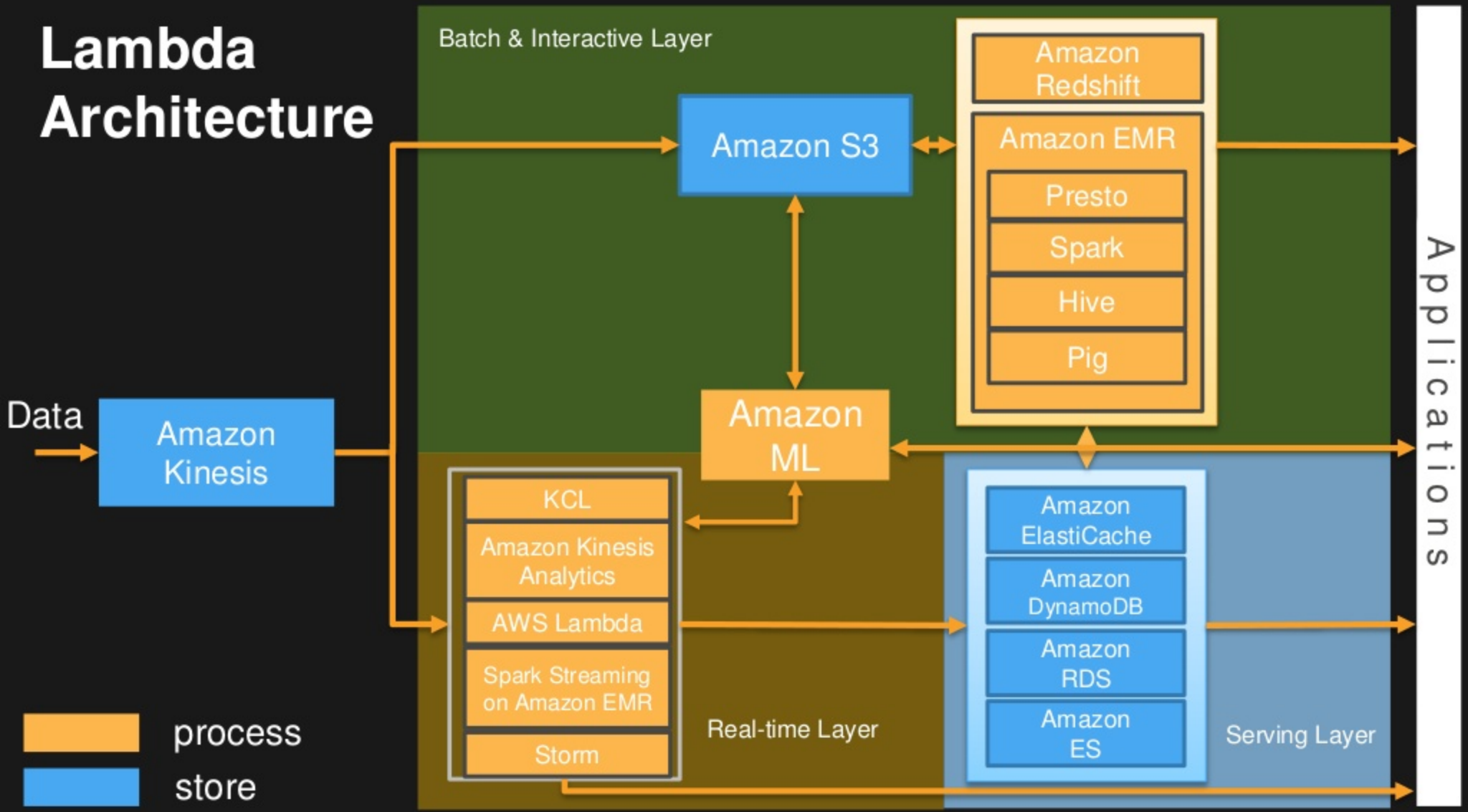

Data temperature
Hot
Cold
data
Fast
Amazon Kinesis
Amazon DynamoDB
Amazon S3
Spark Streaming
Apache Storm
AWS Lambda
KCL apps
Native apps
KCL apps
AWS Lambda
Amazon Redshift
Real-time
Interactive
Amazon Redshift
Spark Presto
Batch
Hive
Hive
Processing speed
Slow
Answers

Real-time Analytics




Interactive & Batch Analytics
Amazon Redshift
Amazon EMR
Presto
Spark
Interactive
Real-time prediction
Amazon Kinesis Firehose
Amazon S3
Amazon ML
Consume
Data stream
Batch prediction
Amazon EMR
Spark
Hive
Pig
Batch
process
store

Lambda Architecture



Summary

Decoupled “data bus”

- Data → Store → Process → Store → Analyze → Answers

Use the right tool for the job

- Data structure, latency, throughput, access patterns

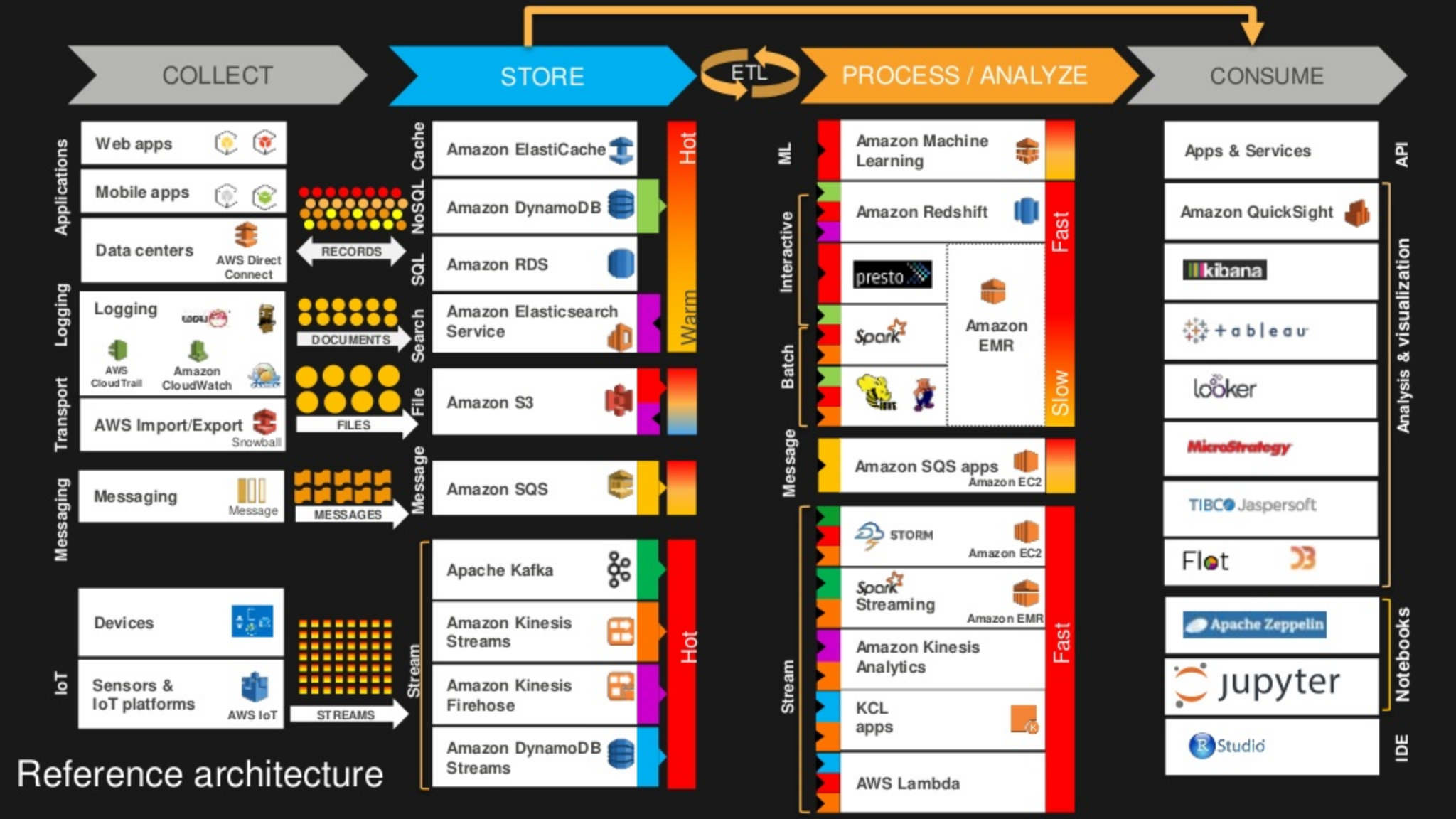
Leverage AWS managed services

- Scalable/elastic, available, reliable, secure, no/low admin

Use Lambda architecture ideas

- Immutable (append-only) log, batch/real-time/serving layer

Big data ≠ big cost



Reference architecture

Thank you!

aws.amazon.com/big-data