
@iHijazi
HITACHI
Inspire the Next
λ & K
Lambda & Kappa Architecture for IoT with Pentaho
Issam Hijazi
Solution Engineering Team Lead
Monday, November 26, 2018
1
© Hitachi Vantara Corporation 2018. All Rights Reserved

Lambda Architecture is
“a distributed data processing **architecture** that is designed to handle **big data** by taking advantage of **batch** and **stream** processing”

History & Objectives

HITACHI
Inspire the Next

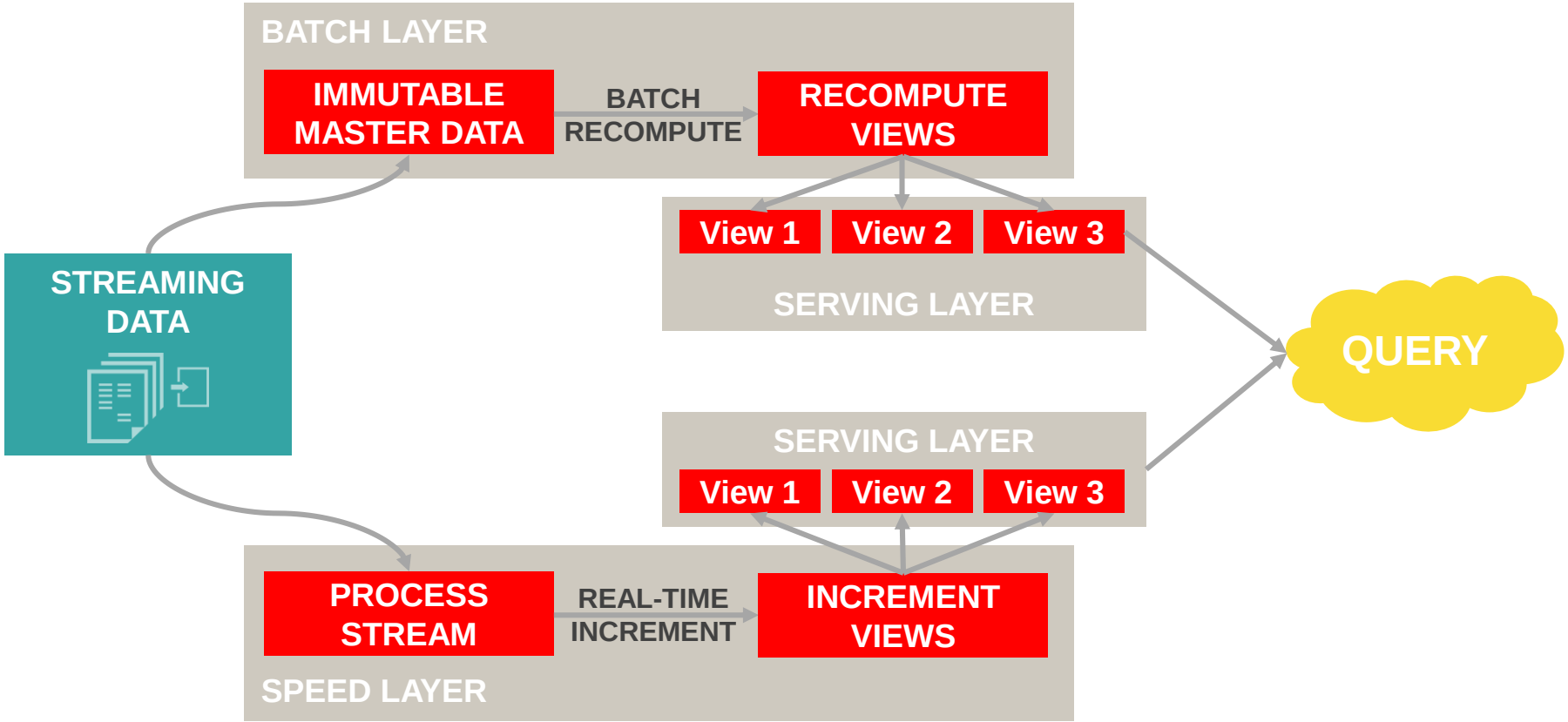
- Coined by *Nathan Marz* in 2012:
  1. Creator of Apache Storm & Elephant DB
  2. Worked for Twitter & Backtype on distributed data processing systems

- Objective to have a system that's:
  1. Linearly scalable
  2. Allowed R/W with low latency
  3. Robust
  4. Fault-tolerant (from human and hardware)
  5. Extensible

- Beating the CAP theorem

Lambda Architecture

HITACHI
Inspire the Next



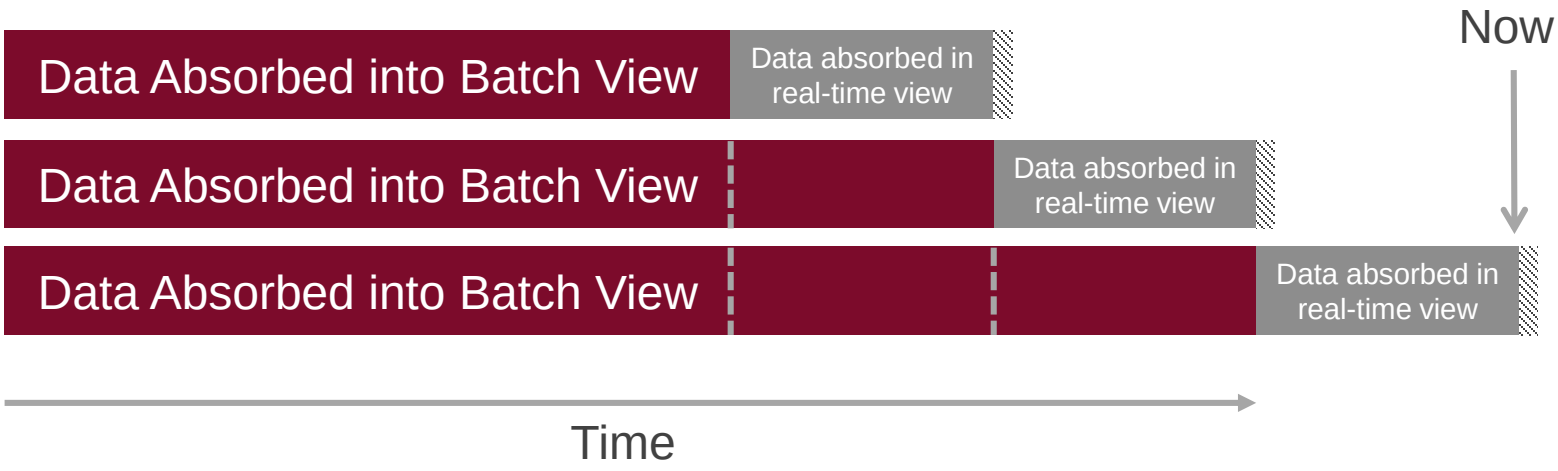
Lambda Architecture Layers

HITACHI
Inspire the Next

- **Batch Layer**
  1. Manage all available data
  2. Immutable
  3. Append only
- **Speed Layer**
  1. Manages recent data only
  2. Low latency
- **Serving Layers**
  1. Result of Batch/Speed Layers (indexed)
  2. Interface for queries
  3. Low latency & Ad-hoc

Relevance of Data for Batch & Real-Time Views

HITACHI
Inspire the Next



Example | Immutable Data + Views

HITACHI
Inspire the Next

Timestamp	Airport	Flight	Action
2018-22-11T09:05:00	DXB	EK093	Take-off
2018-22-11T09:10:00	BLQ	QA121	Take-off
2018-22-11T09:15:00	AMM	RJ978	Take-off
2018-22-11T09:15:00	AMS	KL025	Landing
2018-22-11T09:20:00	AUH	EA287	Landing
2018-22-11T09:25:00	LHR	BA999	Landing

Example | Immutable Data + Views

HITACHI
Inspire the Next

Timestamp	Airport	Flight	Action
2018-22-11T09:05:00	DXB	EK093	Take-off
2018-22-11T09:10:00	BLQ	QA121	Take-off
2018-22-11T09:15:00	AMM	RJ978	Take-off
2018-22-11T09:15:00	AMS	KL025	Landing
2018-22-11T09:20:00	AUH	EA287	Landing
2018-22-11T09:25:00	LHR	BA999	Landing

Immutable Master Dataset

How many air-borne? 4868

Air-borne per airline:

airline	planes
EK	146
RJ	35
BA	105

Airport load:

Airport	Load
DXB	56
LHR	45
BLQ	8

Views

Notes:

1. Last X-hours data is NOT available
2. Can't query up-to-date information
   i.e How many planes are in DXB now?
   i.e How many plans landed in DXB since 1 year until now

Example | Batch Views + Real-Time Views

HITACHI
Inspire the Next

Timestamp	Airport	Flight	Action
2018-22-11T09:05:00	DXB	EK093	Take-off
2018-22-11T09:10:00	BLQ	QA121	Take-off
2018-22-11T09:15:00	AMM	RJ978	Take-off
2018-22-11T09:15:00	AMS	KL025	Landing
2018-22-11T09:20:00	AUH	EA287	Landing
2018-22-11T09:25:00	LHR	BA999	Landing

Immutable Master Dataset

Timestamp	Airport	Flight	Action
2018-23-11T09:05:00	IST	TR124	Take-off
2018-23-11T09:10:00	PAR	LK142	Landing
2018-23-11T09:15:00	JFK	AA999	Take-off
2018-23-11T09:15:00	MUC	MM123	Take-off
2018-23-11T09:20:00	JNB	SA075	Landing
2018-23-11T09:25:00	CPT	MA999	Take-off

Real-Time Dataset

Airport	Load
LHR	56000
DXB	48000
BLQ	8000

Batch View

Airport	Load
LHR	50
DXB	45
BLQ	5

Real-Time View

QUERY

***Now we can query and combine results From both**

Cons of Lambda Architecture

HITACHI
Inspire the Next

- Coding overhead, maintaining two version of codes for:
  1. Batch Layer
  2. Speed Layer

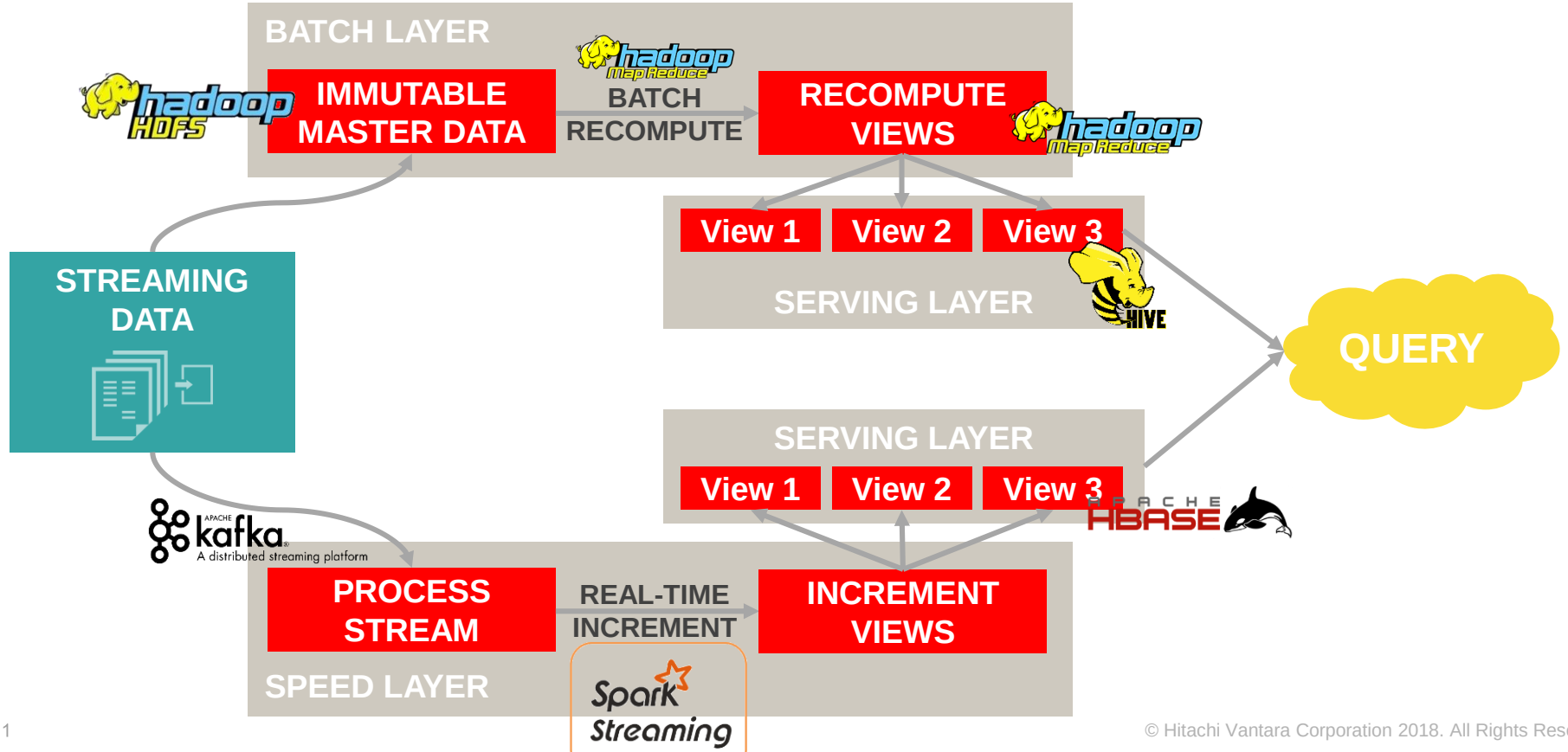
  Quiz: How can we overcome this?

- Re-processing every batch cycle

- Harder to migrate or reorganize

Lambda Architecture – Technology Overview

HITACHI
Inspire the Next



Kappa Architecture is “**simplification** of Lambda Architecture” where “data is simply fed through the **streaming** system quickly and **processed**”

History & Objectives

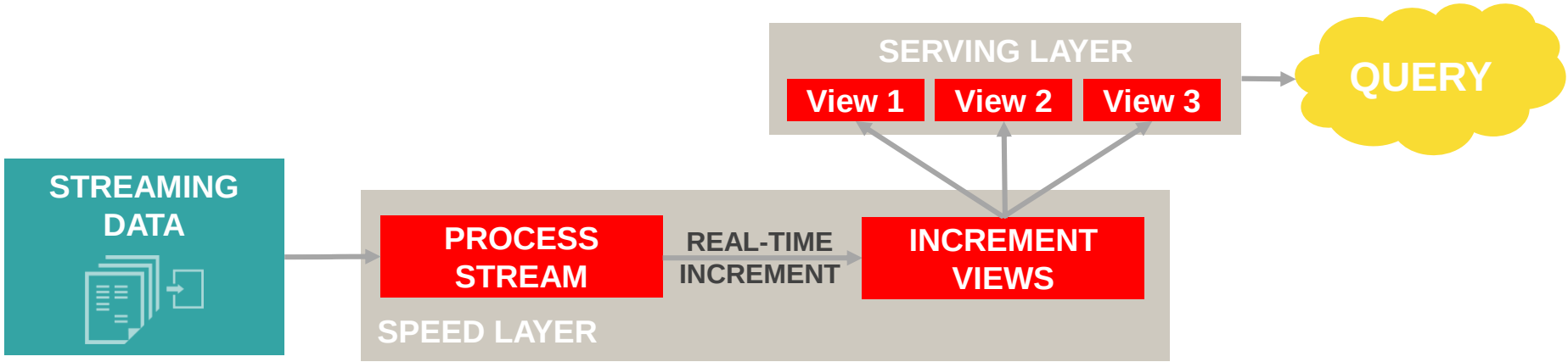
HITACHI
Inspire the Next

- Coined by *Jay Kreps* in 2014:
  1. Author of Apache Kafka & Samza. Co-founder & CEO of Confluent
  2. Worked as a lead architect for LinkedIn

- *Jay* questioned the “Lambda Architecture”, mainly the cons we previously discussed.
- Objectives:
  1. Do both real-time processing and also handle reprocessing (when needed)
  2. Beat the cons of Lambda architecture

Kappa Architecture

HITACHI
Inspire the Next



Kappa Architecture Layers

HITACHI
Inspire the Next

- **Speed Layer**
  1. Manages recent data only
  2. Low latency

- **Serving Layers**
  1. Result of Speed Layer (indexed)
  2. Interface for queries
  3. Low latency & Ad-hoc

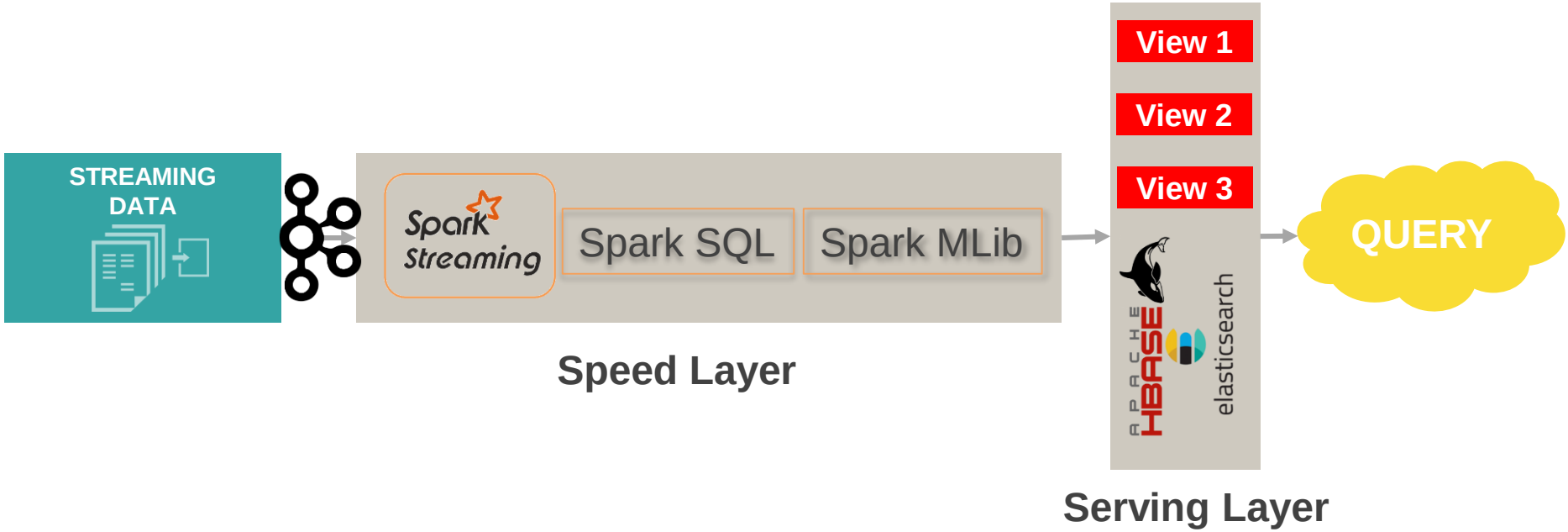
Pros of Kappa Architecture

HITACHI
Inspire the Next

- Everything is a stream
- The starting data is not modified
- Recalculation/reprocessing can always be replayed
- One processing flow, meaning one code to maintain

Kappa Architecture – Technology Overview

HITACHI
Inspire the Next



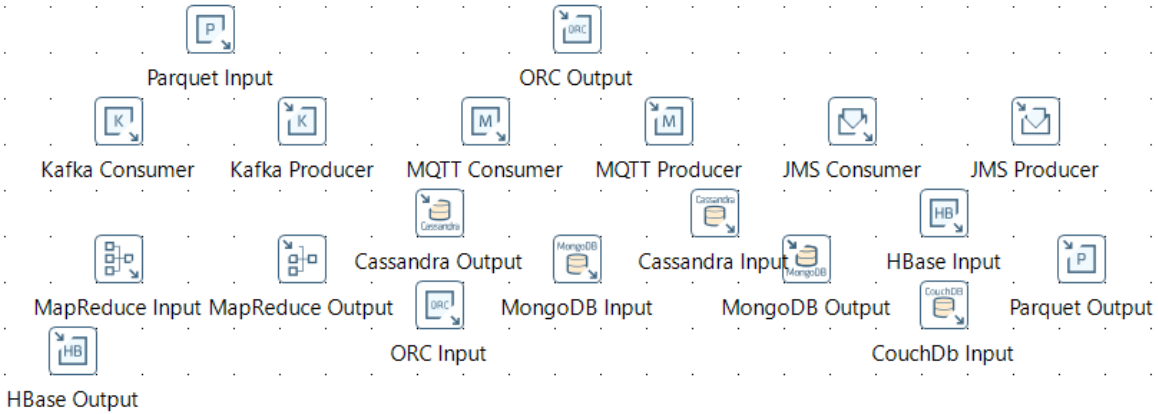
HITACHI
Inspire the Next

What's in it for Pentaho?

Pentaho Speaks λ and κ … Natively!

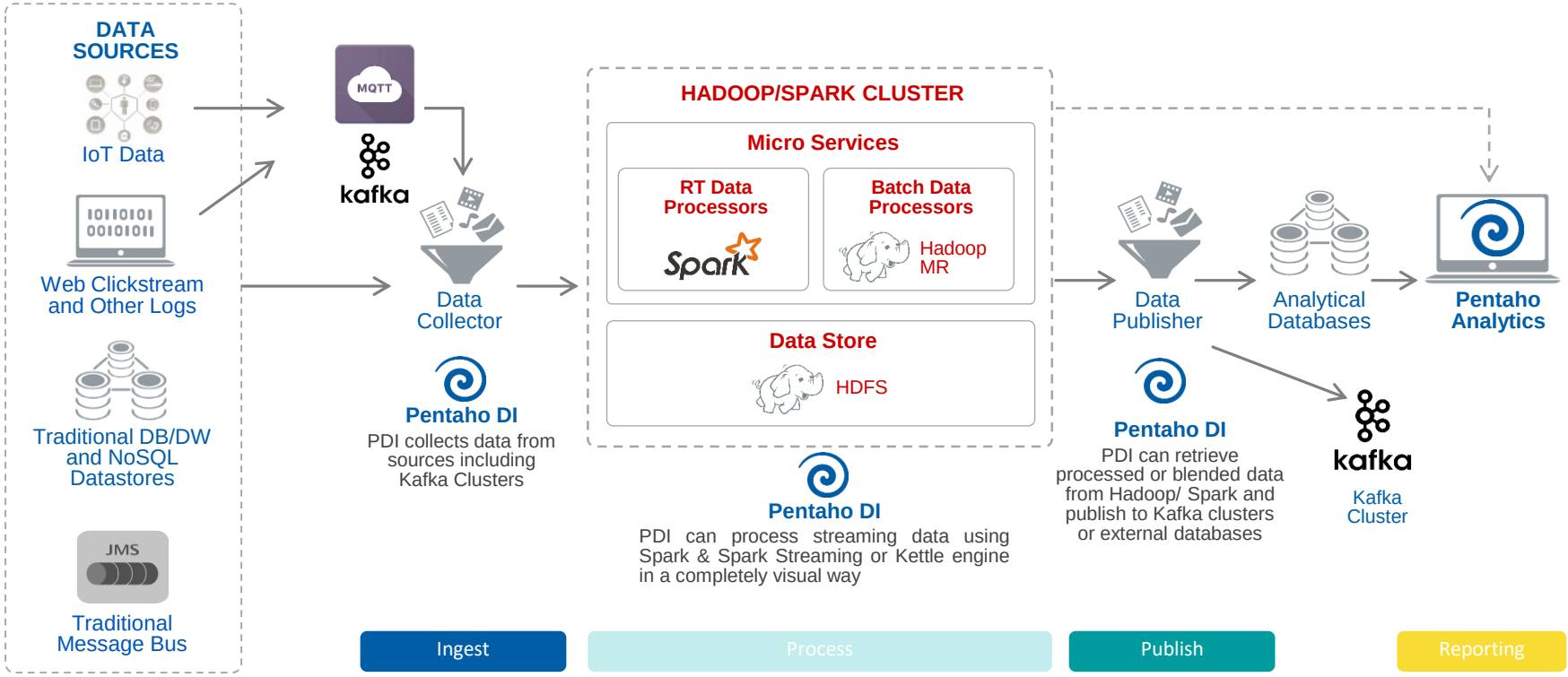
HITACHI
Inspire the Next

- Native steps to implement both architectures
- Native execution on both Spark, MapReduce, Kettle engine itself
- It even goes beyond by eliminating the some of the cons:
  i.e No coding is even required + Adaptive Execution Layer

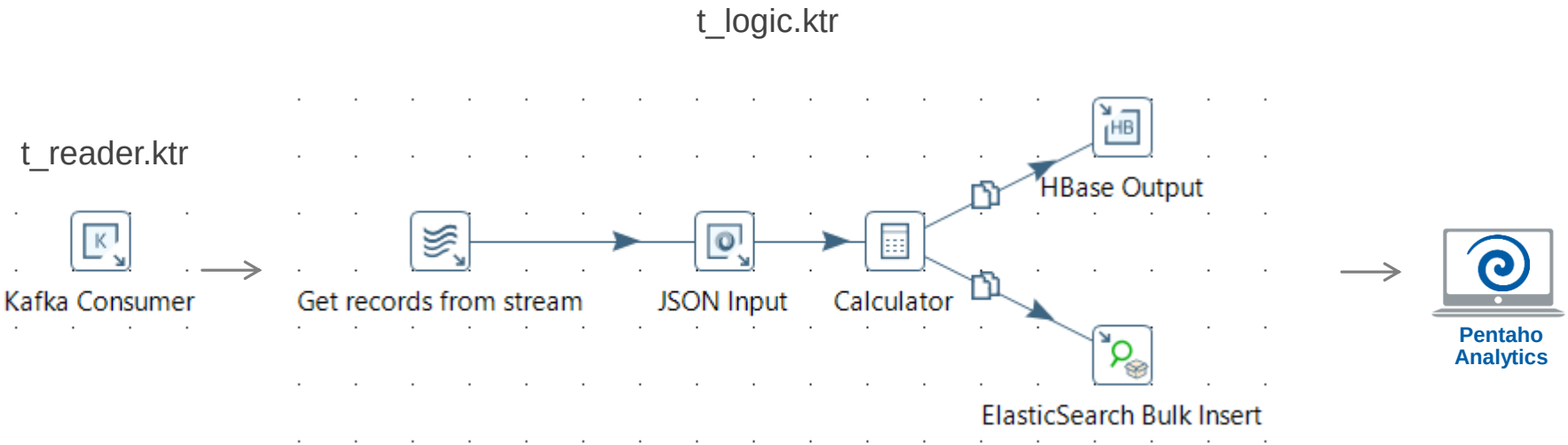


Combined Data Processing Using Pentaho

HITACHI
Inspire the Next



Sample KTR on Kappa

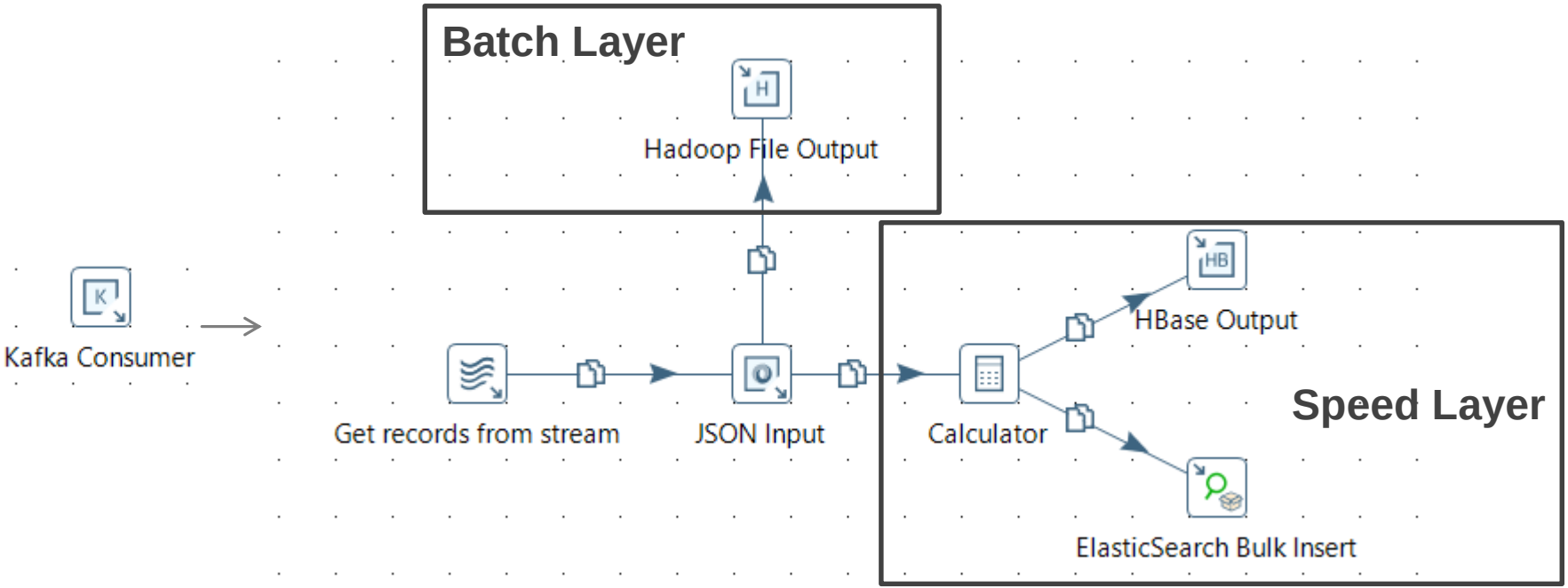


Note that this can be executed on:

1. Pentaho Engine (aka Kettle)
2. Spark Streaming via AEL

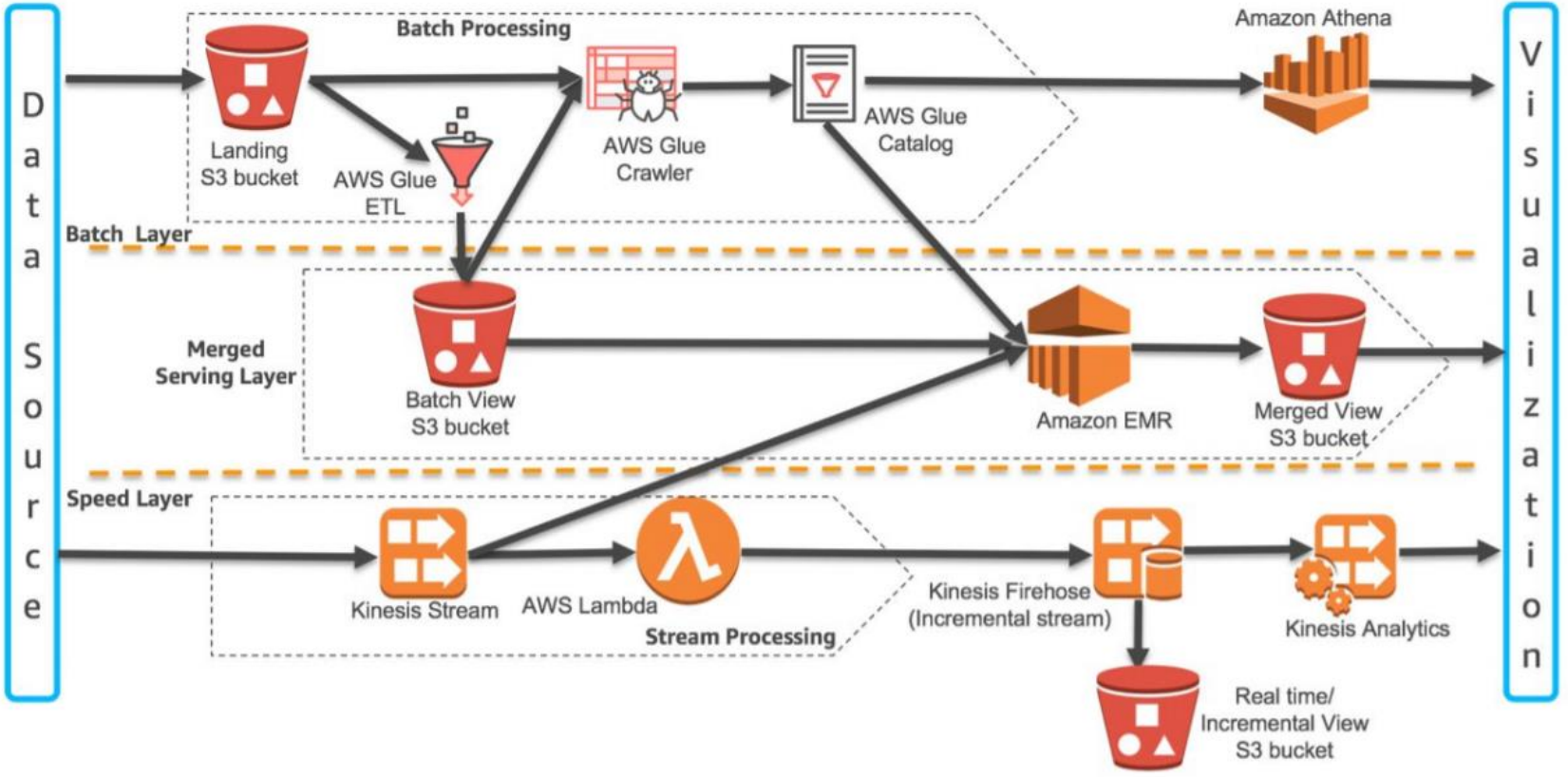
Sample KTR on Lambda

HITACHI
Inspire the Next



Bonus: Lambda Architecture on AWS

HITACHI
Inspire the Next



Last Notes

HITACHI
Inspire the Next

- Input stream doesn’t have to be Kafka, it can be anything
- You can design your flow logic via Jobs and Transformation to send/receive results from one KTR/KTJ to another
- This applies to most use-cases
- Deciding which Architecture to use depends on the use-case:
  1. Is there ML/AI involved?
  2. Is it IoT?
  3. Are resources and/or cost is an issue?

HITACHI

Inspire the Next