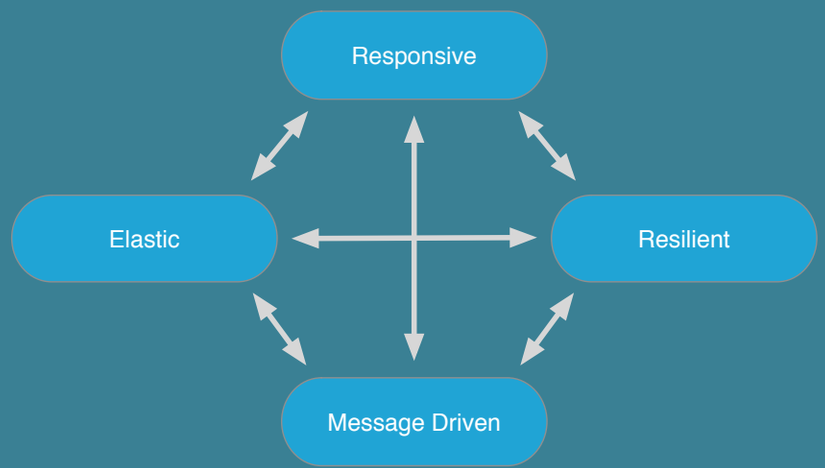

CASSANDRA
SUMMIT2015
Cassandra, Spark
and Kafka:
The Streaming Data Troika
Dean Wampler (Typesafe), Patrick Di Loreto (William Hill)

About Typesafe

Typesafe Reactive Platform

- Akka, Play, and Spark, for Scala and Java.
- typesafe.com/reactive-big-data

What's Reactive?



About

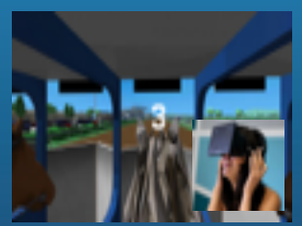
Online Sportsbook and Gaming provider

- Every day we push more than 5 millions price changes
- 160TB of data flowing through our platform each day

WHLABS

WH Apple Watch App

Interactive Scoreboard



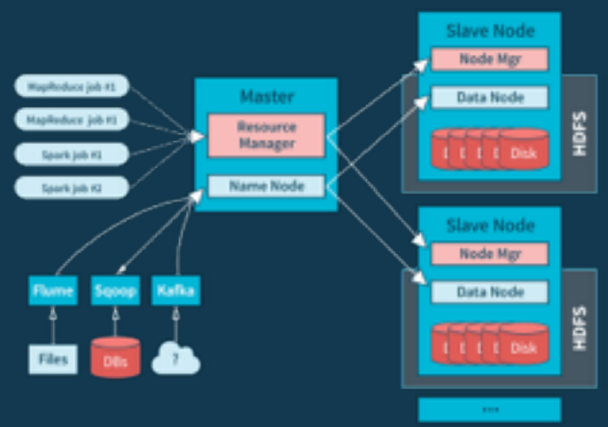
Virtual Reality Horse Race
Oculus Rift

We're Hiring
https://careers.williamhill.com

William HILL ONLINE

Big Data Circa 2010

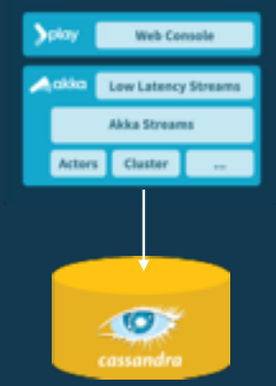
Big Data Circa 2010



Typesafe

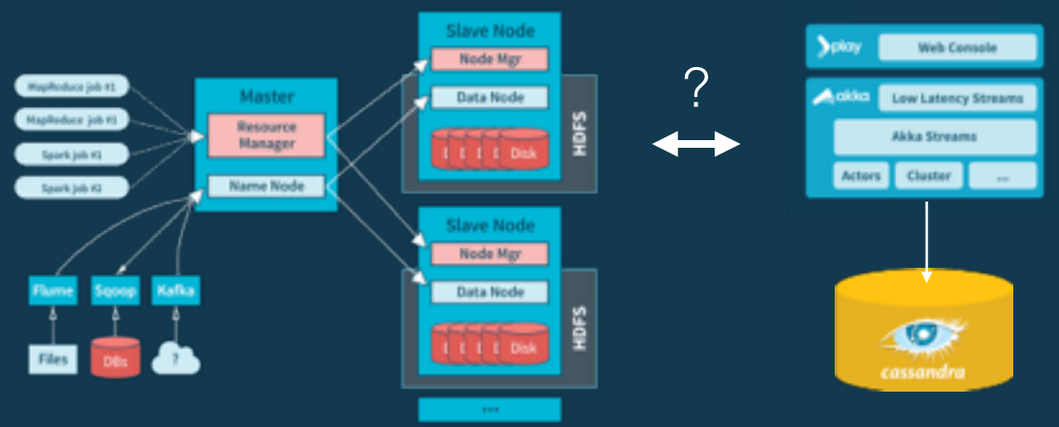
Generally two camps. One was the offline, batch-mode processing of massive data sets done with Hadoop.

Big Data Circa 2010



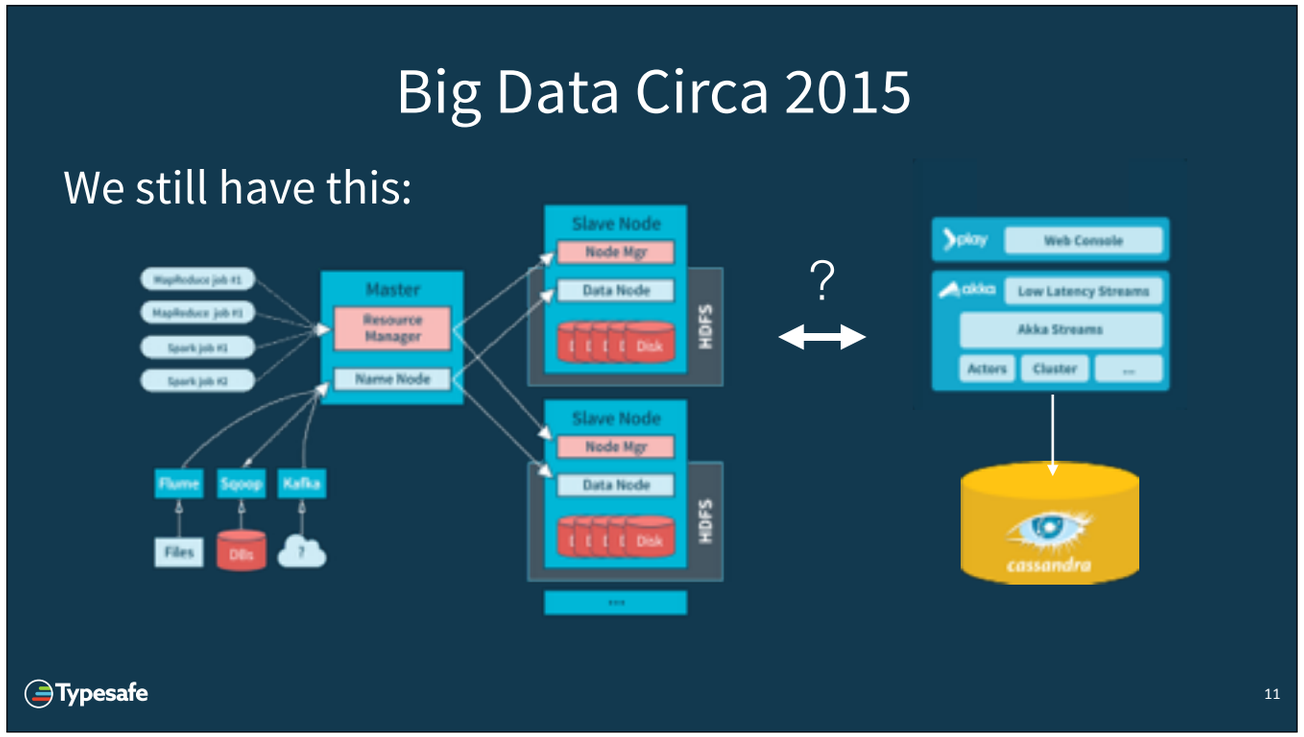
The other was the online, real-time processing and storage of data of “transactional” data at scale, as exemplified by Cassandra for the data store and middleware tools and libraries like Akka, Spring, etc.

Big Data Circa 2010



Two camps together with some overlap and connectivity, but not a lot.

Big Data Circa 2015

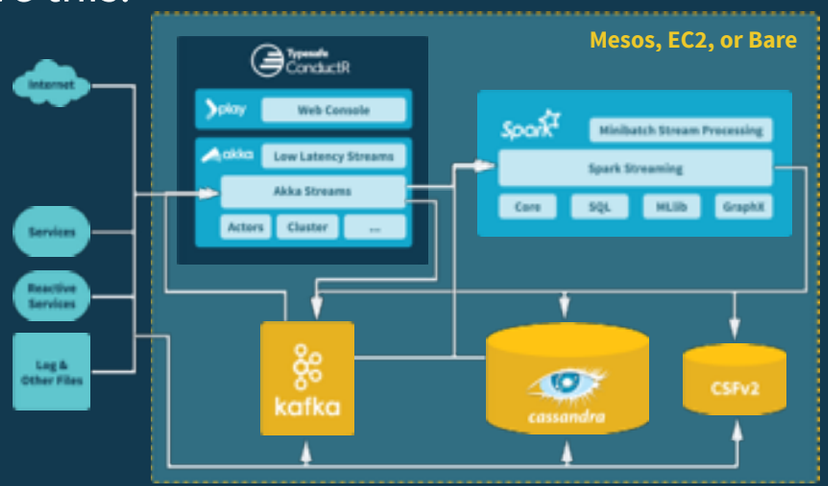


Five years later (this year), we still have these architectures in wide use, but…

Big Data Circa 2015

But now we have this:

Big Data
Streaming



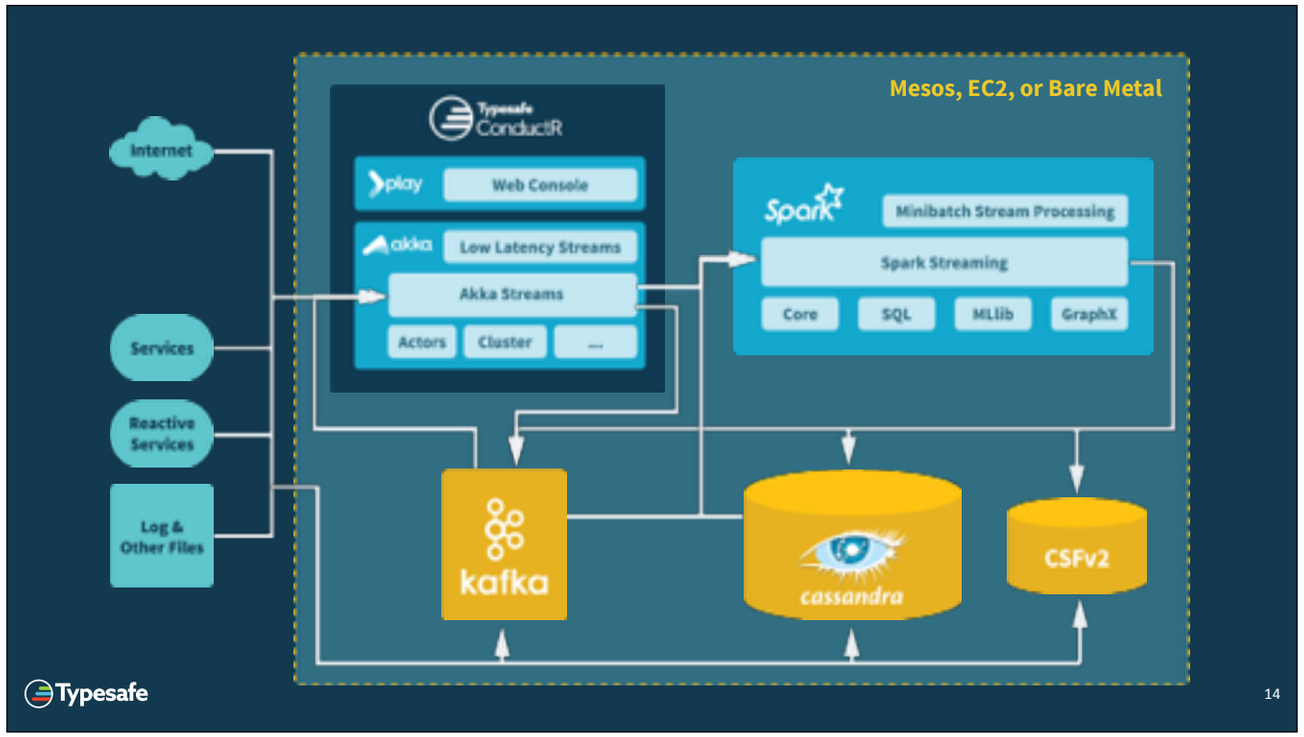
A new, streaming-oriented architecture is emerging, which can also be used for batch mode analysis, if we process resident data sets as finite streams.

General Principles

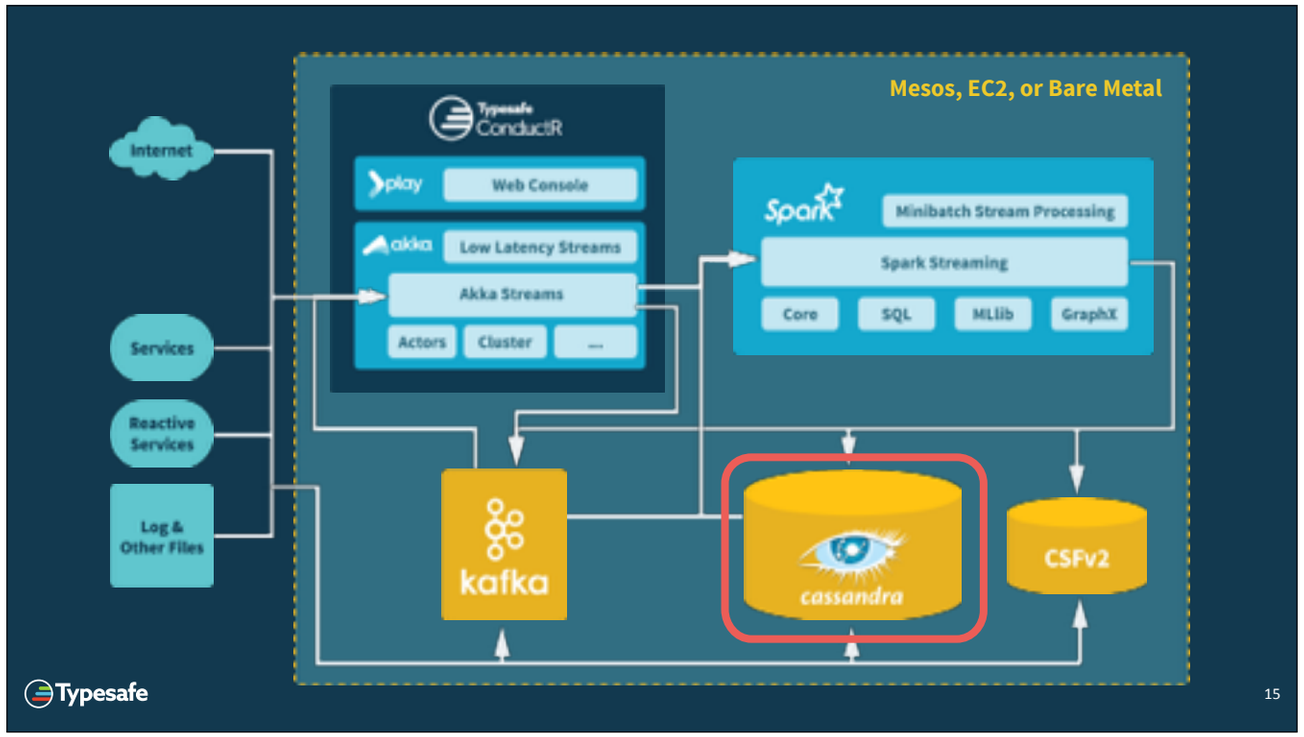
- Spark Streaming: Analytics/aggregations
- C*: Storage, queries
- Kafka: durable message store; allows replay of messages lost downstream.

Topic A

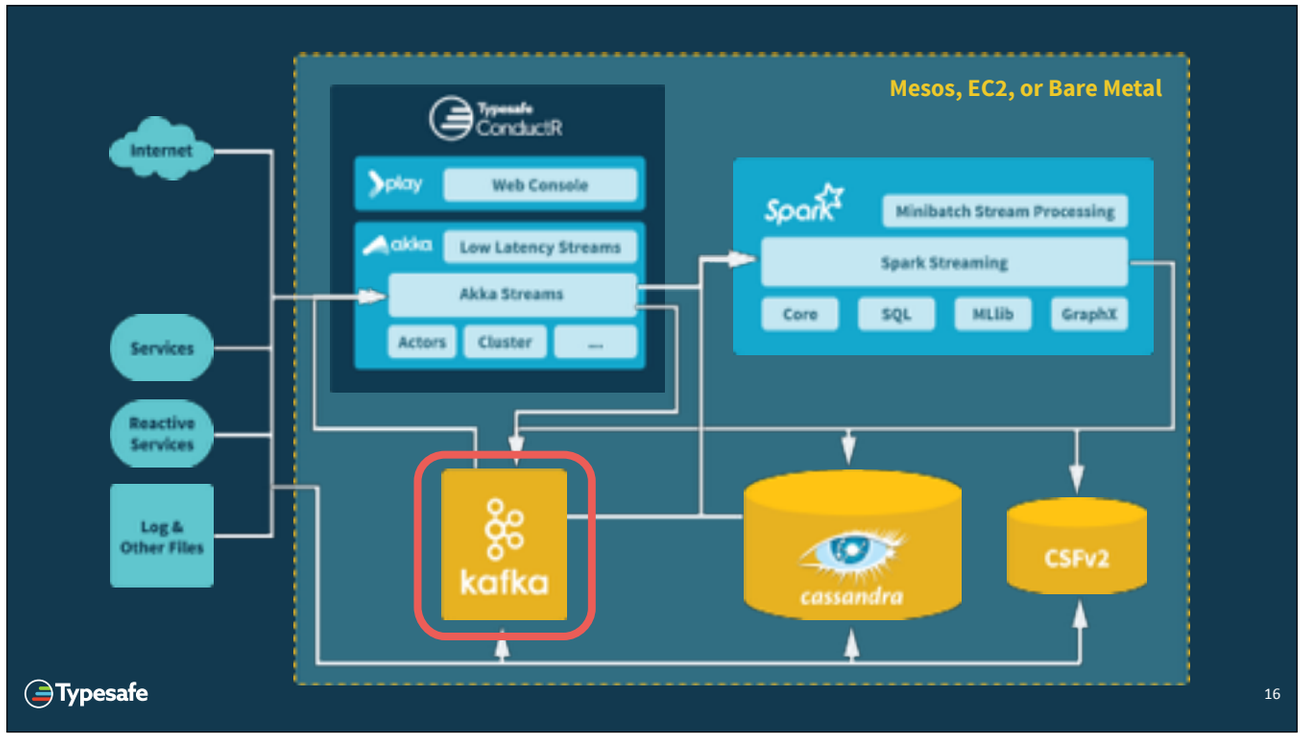
Spark Streaming provides rich analytics.
Need a durable system of record, like Kafka, which allows repeat reads in case of loss. See https://medium.com/@foundev/real-time-analytics-with-spark-streaming-and-cassandra-2f90d03342f7 for a nice summary of design patterns and tips.



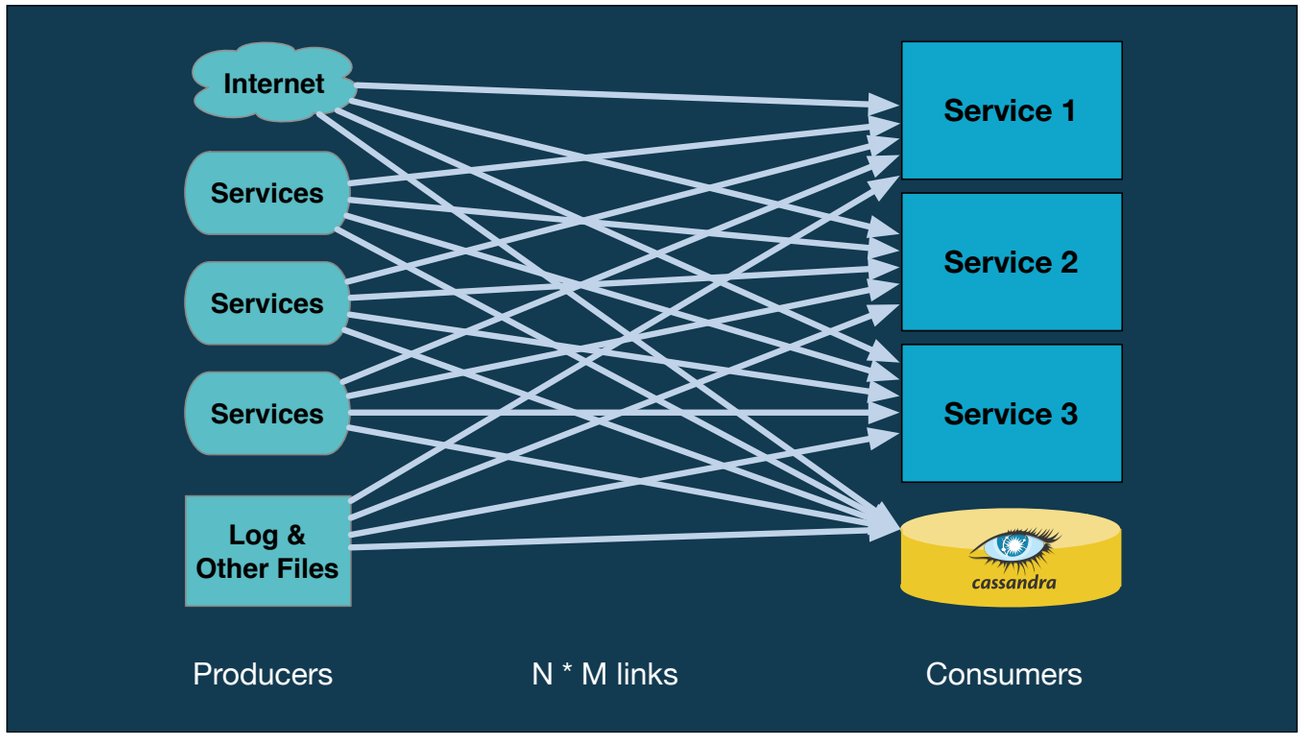
Let’s explore this.



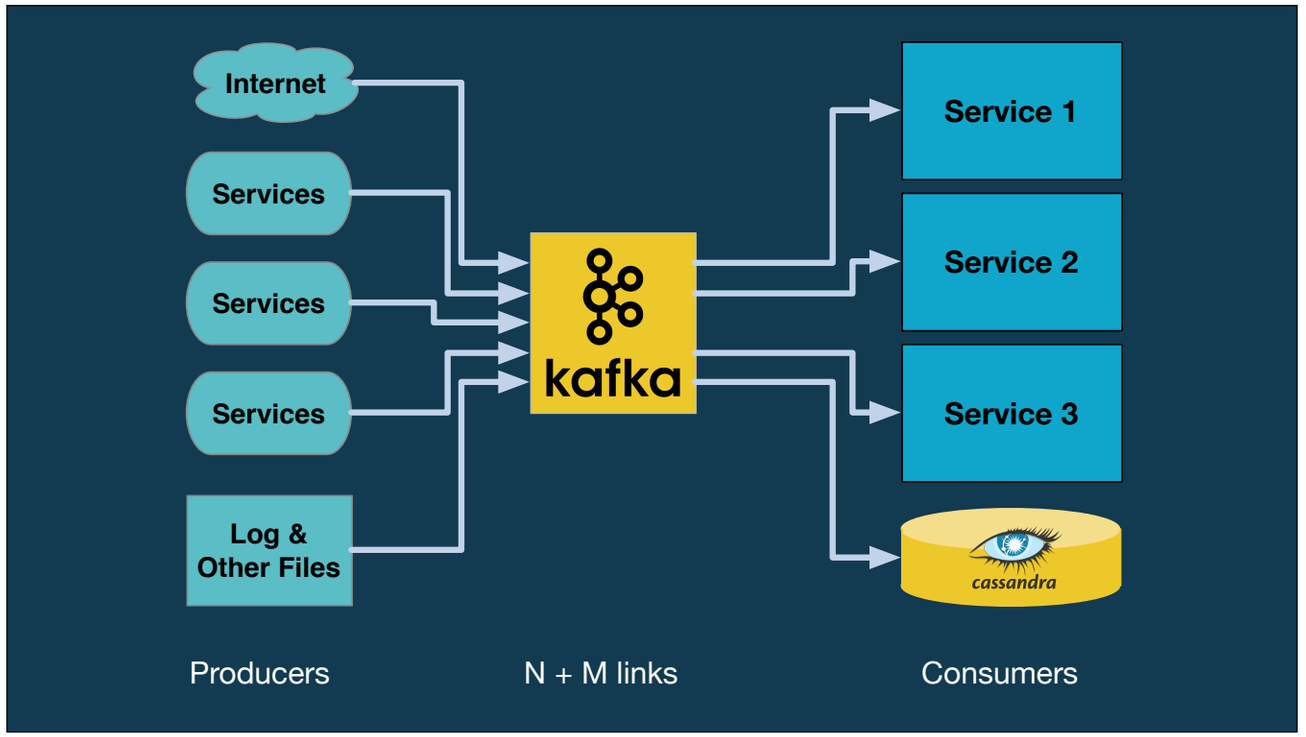
Cassandra remains the flexible, scalable datastore suitable for scalable ingesting of streaming data, such as event streams (e.g., click streams from web apps) and logs.



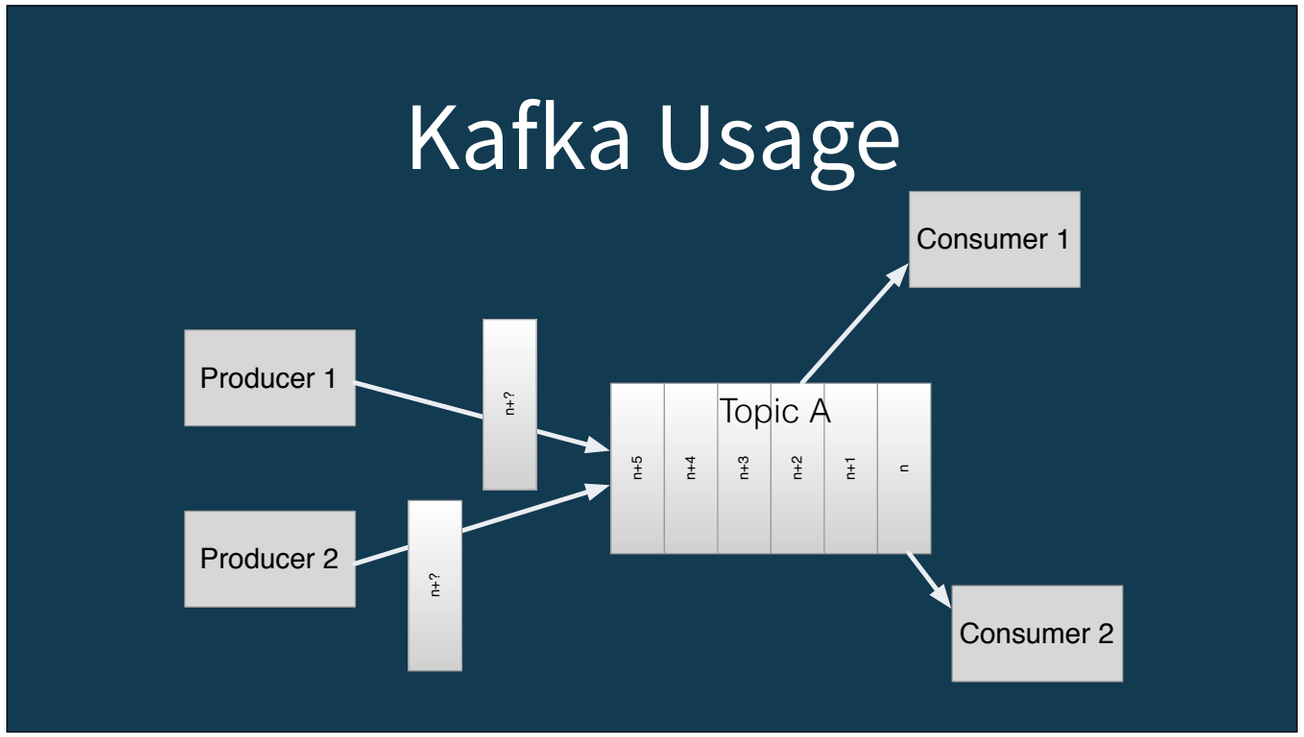
Kafka is growing popular as a tool for durable ingestion of diverse event streams with partitioning for scale and organization into topics (like a typical message queue) for downstream consumers.



One use of Kafka is to solve the problem of N*M direct links between producers and consumers. This is hard to manage and it couples services to directly, which is fragile when a given service needs to be scaled up through replication or replacement and sometimes in the protocol that both ends need to speak.



So Kafka can function as a central hub, yet it’s distributed and scalable so it isn’t a bottleneck or single point of failure.

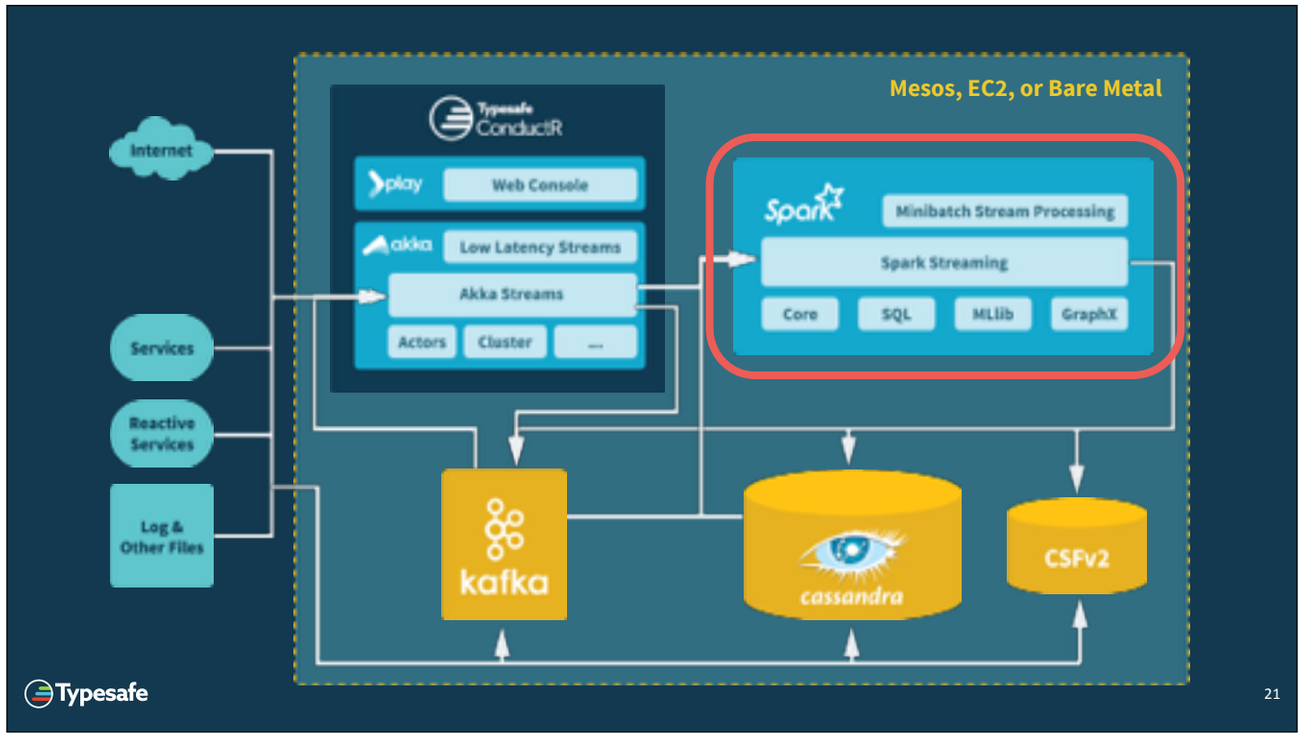


The message queue structure looks basically like this. Where different producers can write to append messages to a topic and different consumers can read the messages in the queue at their own pace, in order.

Kafka Resiliency

Data loss downstream? Can replay lost messages.

Could use C* for this, but then you've changed the read/write load (and hence tuning, scaling, etc. of your C* ring).



The third element of the “troika” is Spark, the next generation, scalable compute engine that is replacing MapReduce in Hadoop. However, Spark is flexible enough to run in many cluster configurations, including a local mode for development, a simple standalone cluster mode for simple scenarios, Mesos for general scalability and flexibility, and integrated with Cassandra itself.

Spark Streaming Dos/Don’ts

Do

Topic A

- Use for rich analytics and aggregations.
- Use with Kafka/C* source if data loss not tolerable. Or, use the write ahead log (WAL) - less optimal.

Spark Streaming offers rich analytics, even SQL, machine learning, and graph representations. It’s a more complex engine, so there is more “room” for data loss. Hence, use Kafka or C* for durability and replay capabilities, but if you do ingest data directly from other sources without replay capability, at least use the WAL.

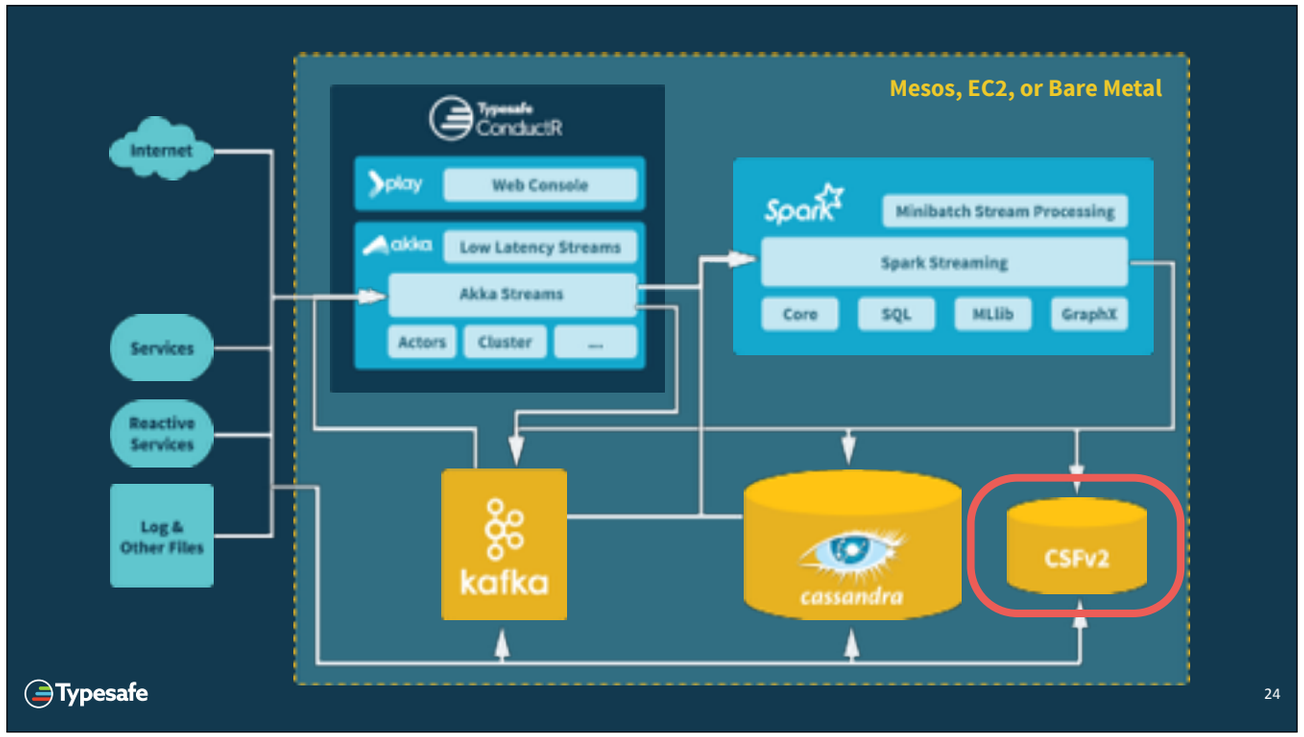
Spark Streaming Don’ts

Don’t

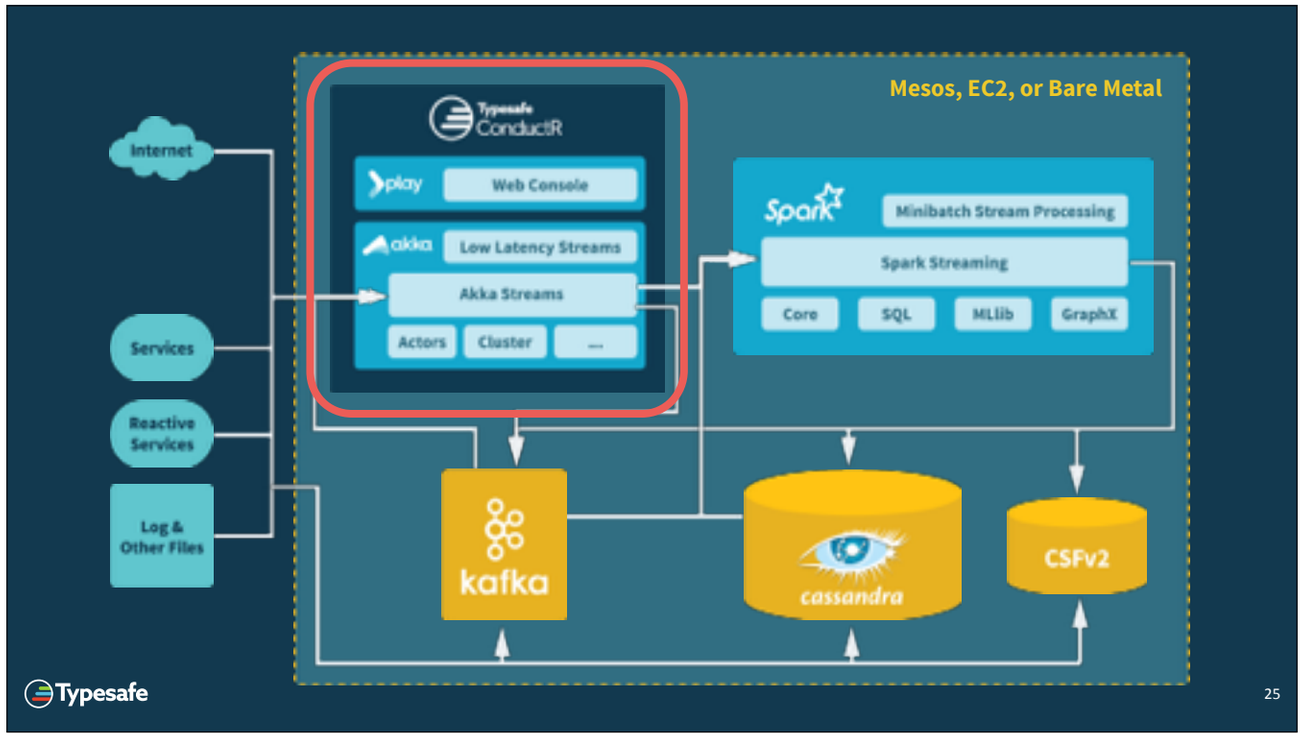
- Use for counting (use C*).
- Low-latency, per-event processing.

Topic A

C* is faster and more accurate for counting, because repeat execution of Spark tasks (for error recovery, speculative execution, etc.) will cause over-counting (e.g., using the “aggregator” feature). Also, Spark is a mini-batch system, for processing time slices of events (down to ~1 sec.). If you need low-latency and/or per-event processing, use Akka…



Other parts of complete infrastructure include a distributed file system like CSFv2, when you don’t need a full database, e.g., for logs that you’ll dump into the file system and then process in batches later on with Spark.



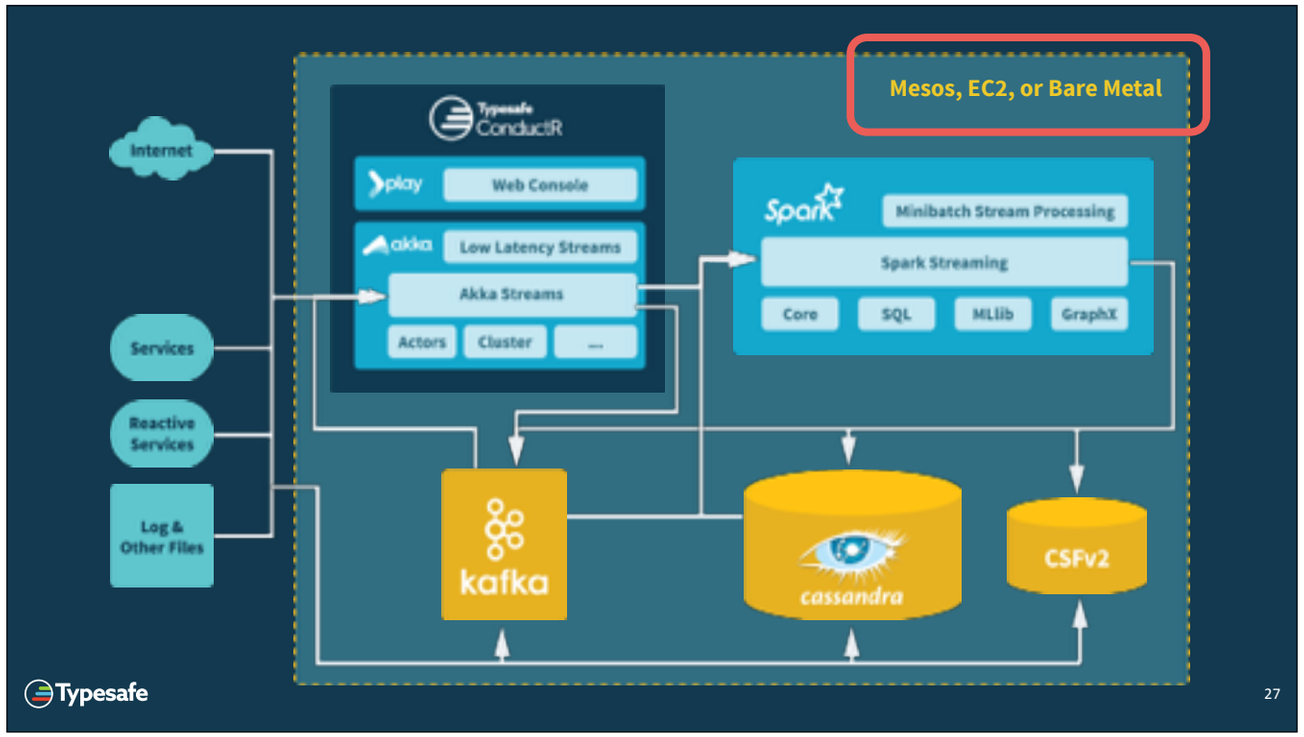
Typesafe Reactive Platform provides infrastructure tools for integrating these and other components, including Akka Streams for resilient, low-latency event processing (based on the Reactive Streams standard for streams with dynamic back pressure), ConductR for orchestrating services, and Play for web services and consoles.

Typesafe Reactive Platform

- Akka Streams: low-latency, per-event processing.
- ConductR for orchestrating services.
- Play for web services, consoles.
- … and commercial Spark support.

Topic A

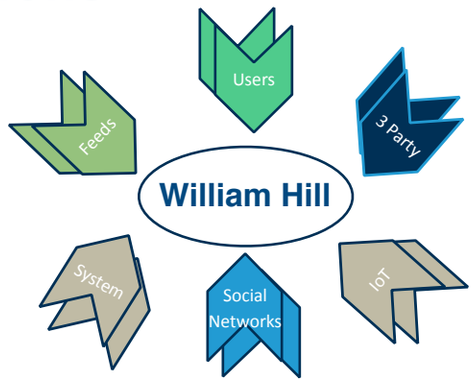
Akka Streams implements the Reactive Streams standard for streams with dynamic back pressure. It sits on top of the more general Akka Actor framework for highly distributed concurrent applications.
Typesafe offers commercial support for development teams developing advanced Spark applications. We offer production runtime support for Spark running on Mesos clusters.



Finally, there’s a wealth of cluster systems possible. You could deploy these tools on your servers for you Cassandra Ring, which has an excellent integration with Spark. You can run in EC2 or bare metal. You can use a general-purpose cluster management system like Mesos.


We Are Reactive
OMNIA
Distributed & Reactive
platform for data management
Presented by Patrick Di Loreto
R&D Engineering Lead
Site: https://developer.williamhill.com
Twitter: https://twitter.com/patricknoir
William HILL
ONLINE

Motivations

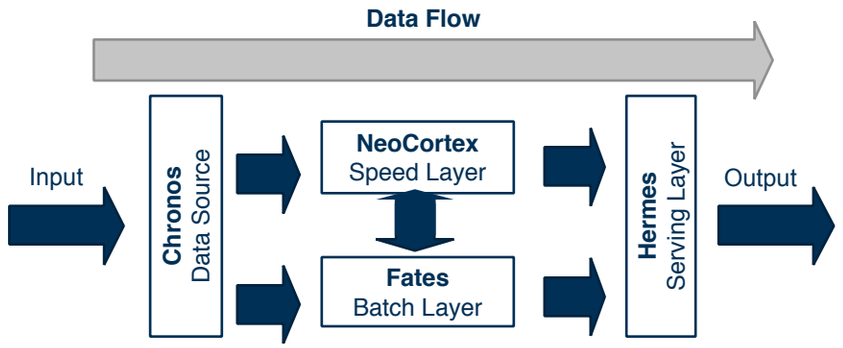


In order to be in a position to innovate we need to control and understand our data

Need for control over the data

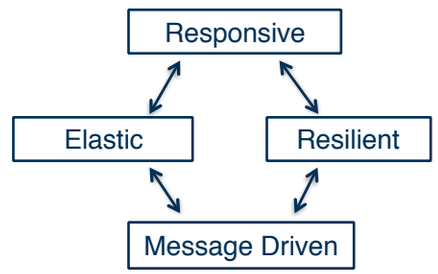
What is Omnia?

DMP based on the *Lambda architecture* and the *Reactive principles*



Lambda architecture

Reactive principles

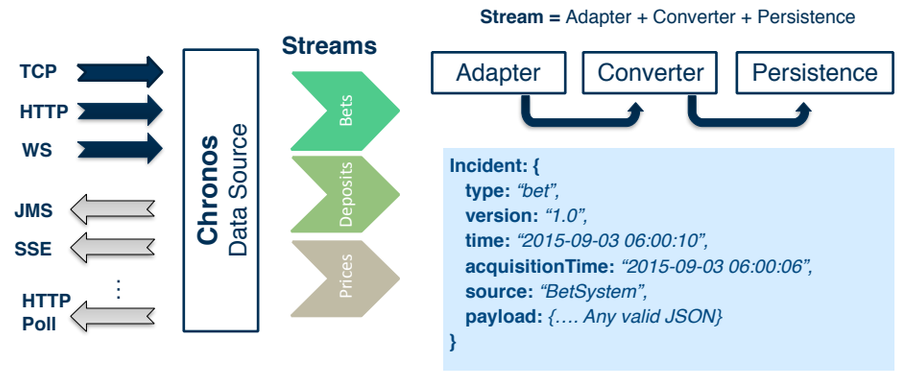


The Reactive Manifesto
http://www.reactivemanifesto.org/

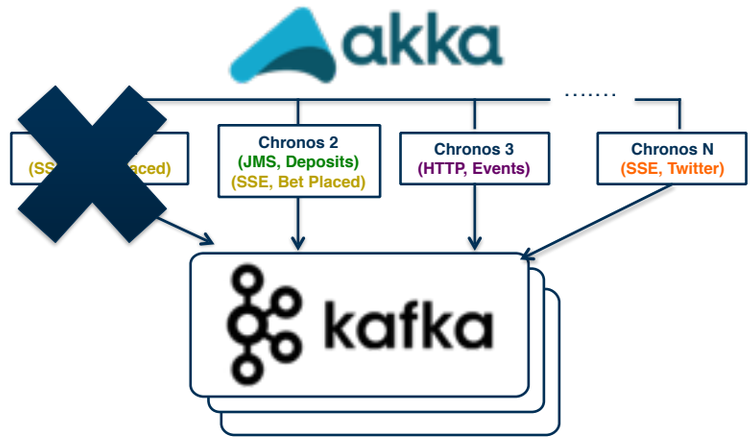
Reactive Manifesto

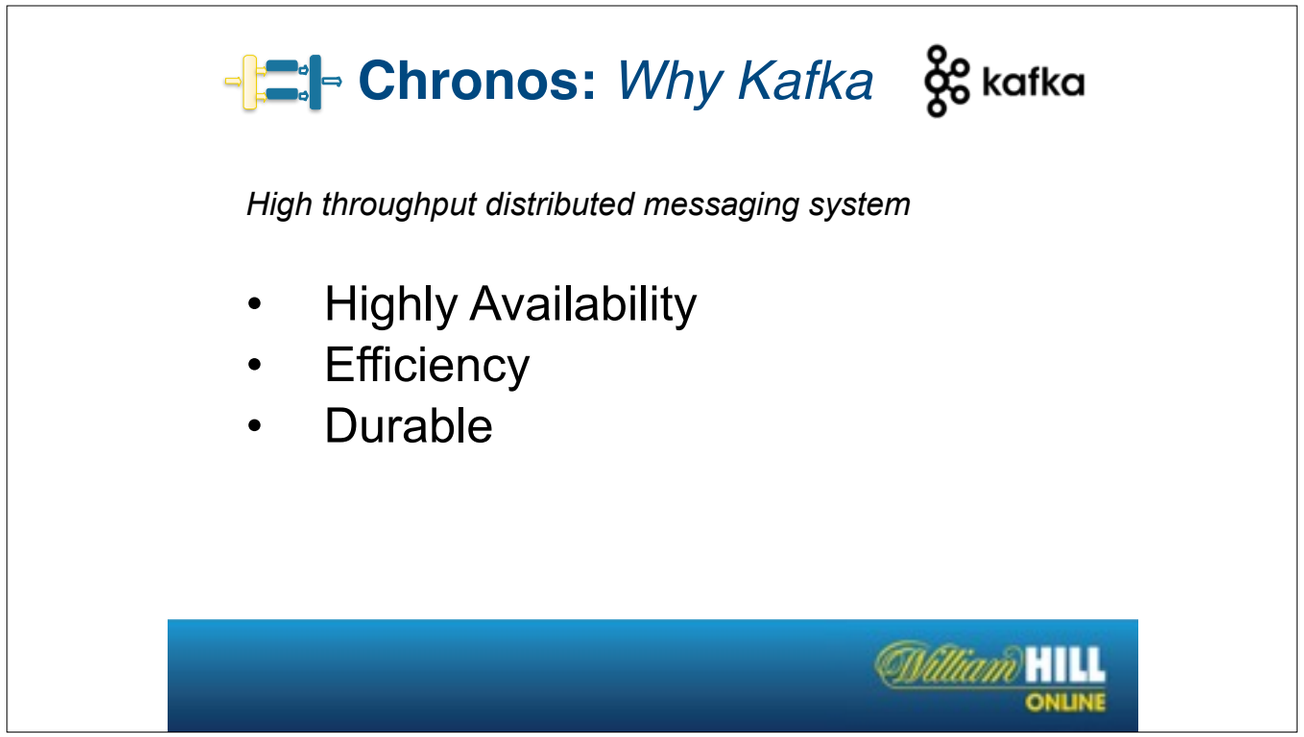
Chronos: *Data acquisition*

Chronos is a reliable and scalable component which collect data from different sources and organize them into **Streams** of observable events.



Chronos: *Data acquisition*





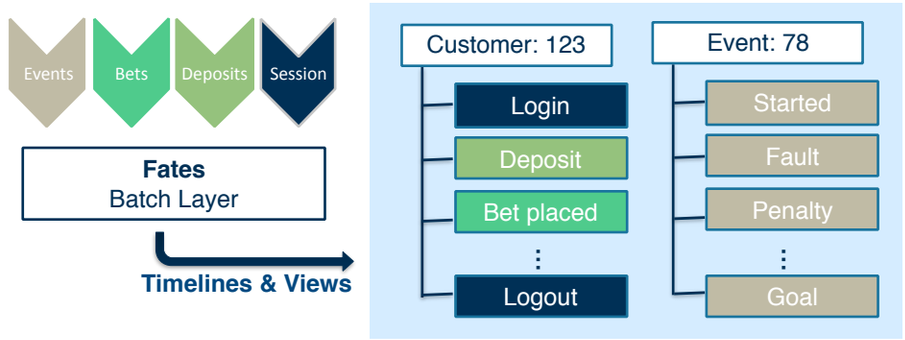
Kafka is a high-throughput distributed messaging system

Design Principles:

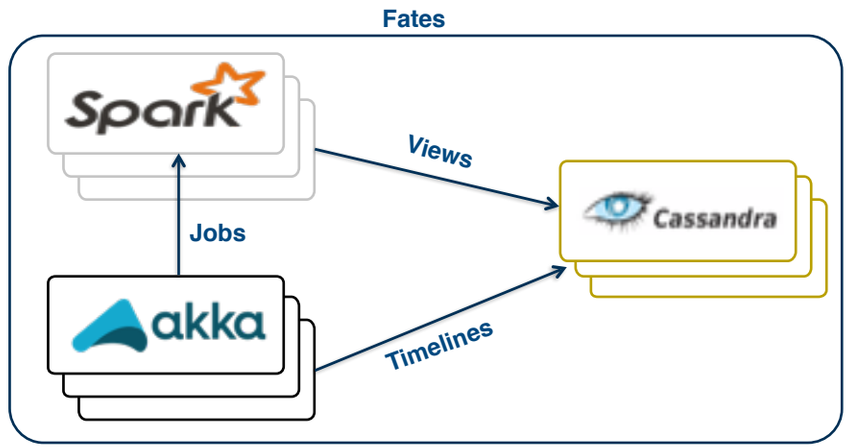
- Highly Available: Replicated Distributed
- High throughput: Stateless Broker
- Efficiency:
  - Disk Efficiency : “Don’t fear the file system” – modern OSs optimize sequential disk operations/disk caching strategy
    - Usage of OS filesystem cache rather than application level cache:
      - More efficient (no usage of GC)
      - Survive on application restart
  - I/O Efficiency : Batching – Reduces small I/O operations, this mortize network roundtrip overhead, enhance larger sequential disk operations
- Durable

Fates: *Batch layer*

Fates represents the long term memory of *Omnia*. It organizes the incidents that *Chronos* collected into **timelines** and also elaborates new information as **views** by using machine learning, logical reasoning and time series analysis.



Fates: *Batch layer*



Fates: *Cassandra*

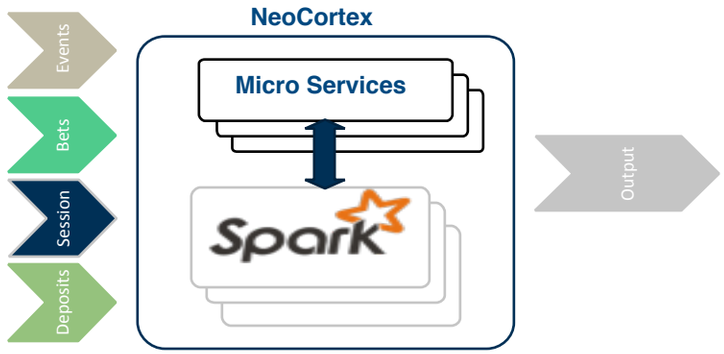
Cassandra is the long term storage for our data.

- Highly Available (CAP)
- Linear Scalability
- Multi DC – Separation of Concerns (Production and Analytic DCs)
- High performance and optimal for WRITE operations

William HILL ONLINE

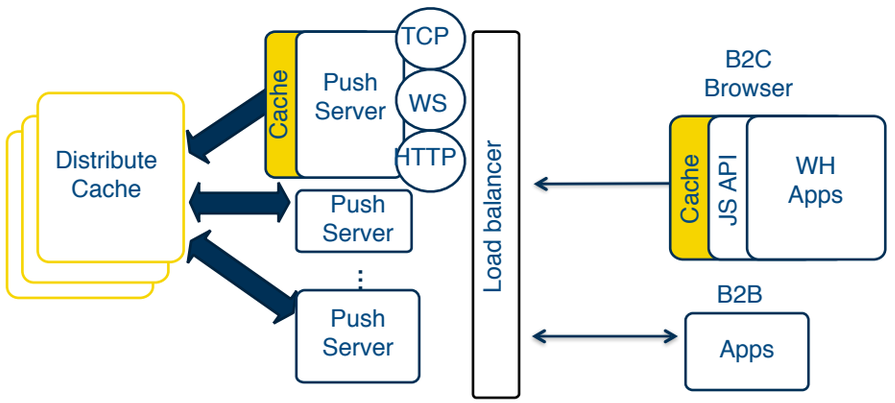
NeoCortex: *Speed layer*

NeoCortex represents the short term memory of *Omnia*. It offers a framework to develop **micro services** on top of *Apache Spark*. It performs fast and real time data processing with the data acquired from *Chronos* and *Fates*.

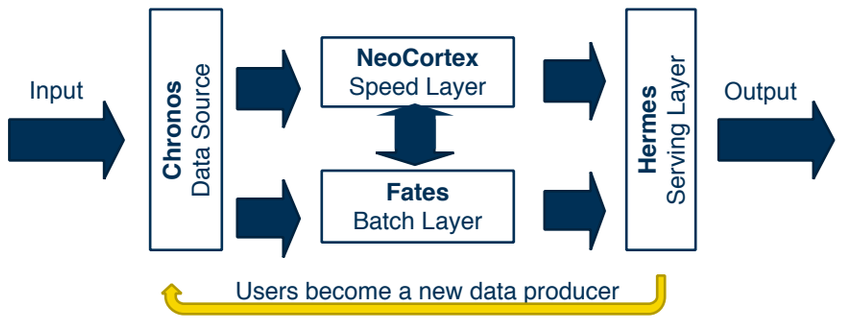


Hermes: *Serving Layer*

Hermes is a **scalable** and **full duplex** communication for *B2C* and *B2B*.

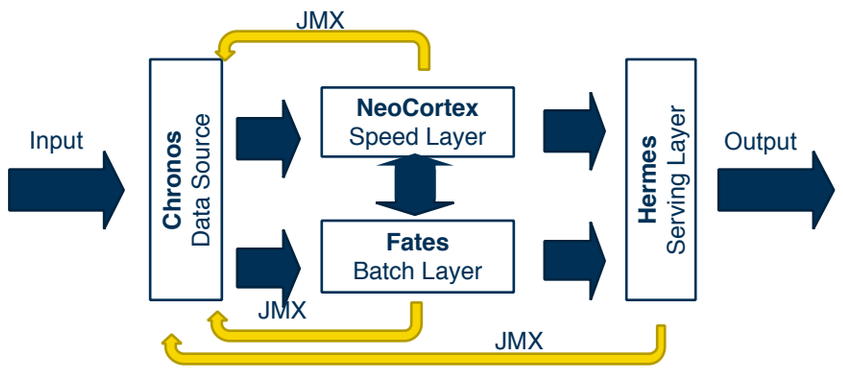


Omnia Data Flow



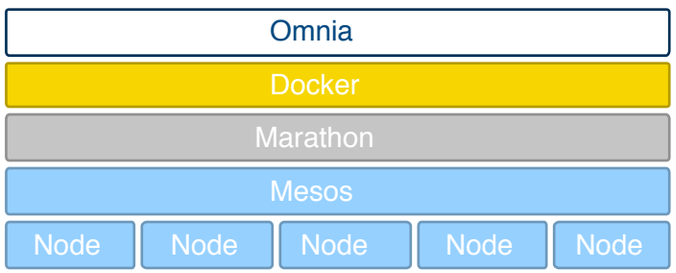
Custom advert, bonus, data load prediction, bot detection...

Omnia on Omnia



Real time monitoring and elasticity
Docker and Mesos: Scale In&Out based on demand,

Omnia infrastructure




Typesafe
CASSANDRA
SUMMIT2015
William HILL
We're Hiring!
Thank you
typesafe.com/reactive-big-data
careers.williamhillplc.com
omnia.williamhill.com/