

Coherence & Big Data

Ben Stopford



Can you do 'Big Data' in
Coherence?

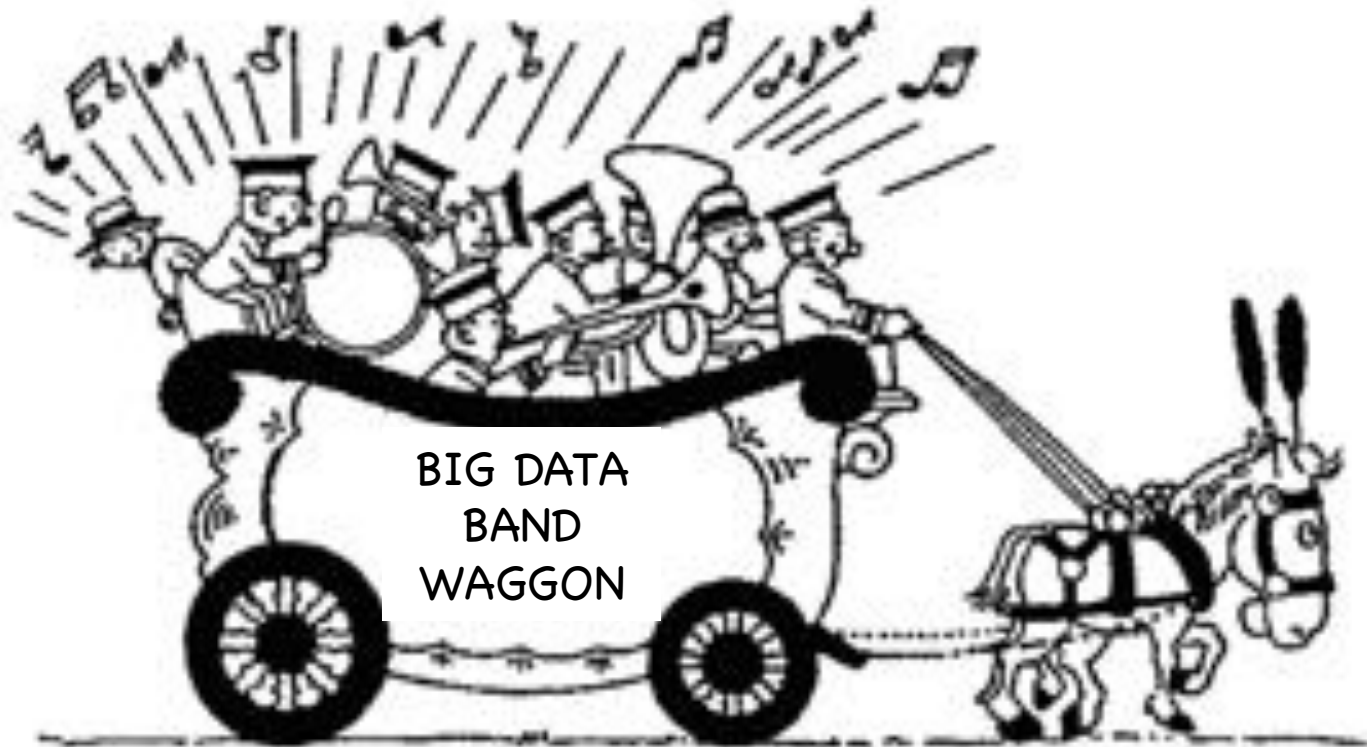
Maybe?!?!?!?

- Problem: Cost of memory / 6x storage ratio
 - > Elastic data (Disk or RAM)
 - > Keep number indexes small
 - > off heap indexes (coming)
- Problem: Getting your (big) data loaded
 - > Recoverable caching
 - > Use other distributed backing store

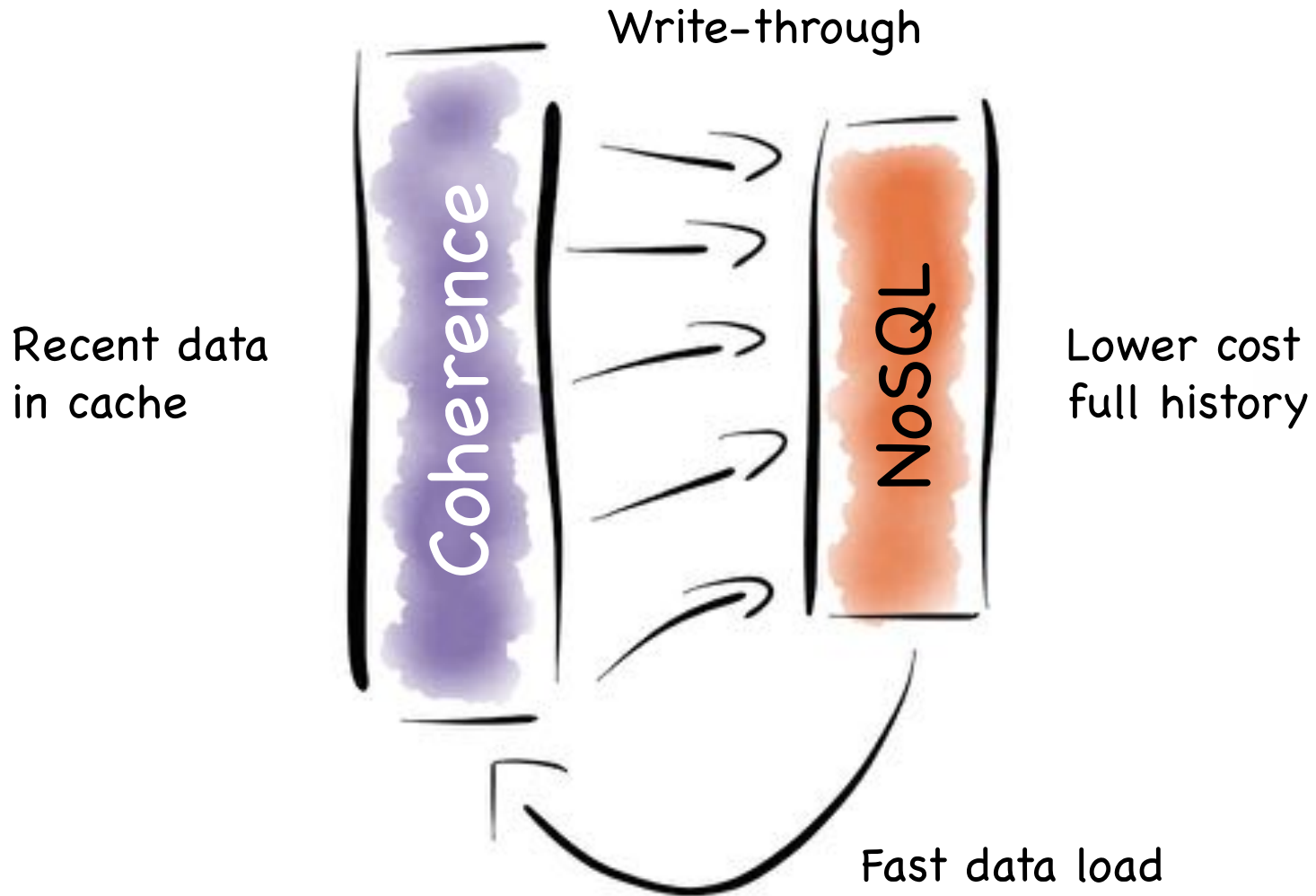
But

- Elastic data & recoverable caching are separate (plan to unify)
 - RC => ED is IO intensive (two distinct copies).
 - 2x disk footprint
 - No compression
 - Rebalance time
 - Memory Ratio (the 6x)
 - >>> Low TB Zone

BIG DATA BANDWAGON



Backing Layer



Hadoop

- Backing

- HDFS

- Big files (~GBs)
 - No random write (ok if you journal writes)
 - Use sequence files
 - Hard to manage active set








- Hbase (Better option)

- Fast writes (LSM)
 - Supports predicate pushdown
 - More complex setup (ZK, NN etc)



NoSQL Backing

- Cassandra  ← Low memory footprint, write optimised
- MongoDB  ← Read/Memory optimised (3.0 big improvement). Rich queries.
- Oracle NoSQL  ← KV with secondary indexes & range predicates
- Riak  ← KV but can scan with MR API. Eventually consistency may not suit
- Couchbase  ← Heavily memory optimised. Fast but too similar to Coherence to be a good fit

Streams



Message Stream Products

RabbitMQ



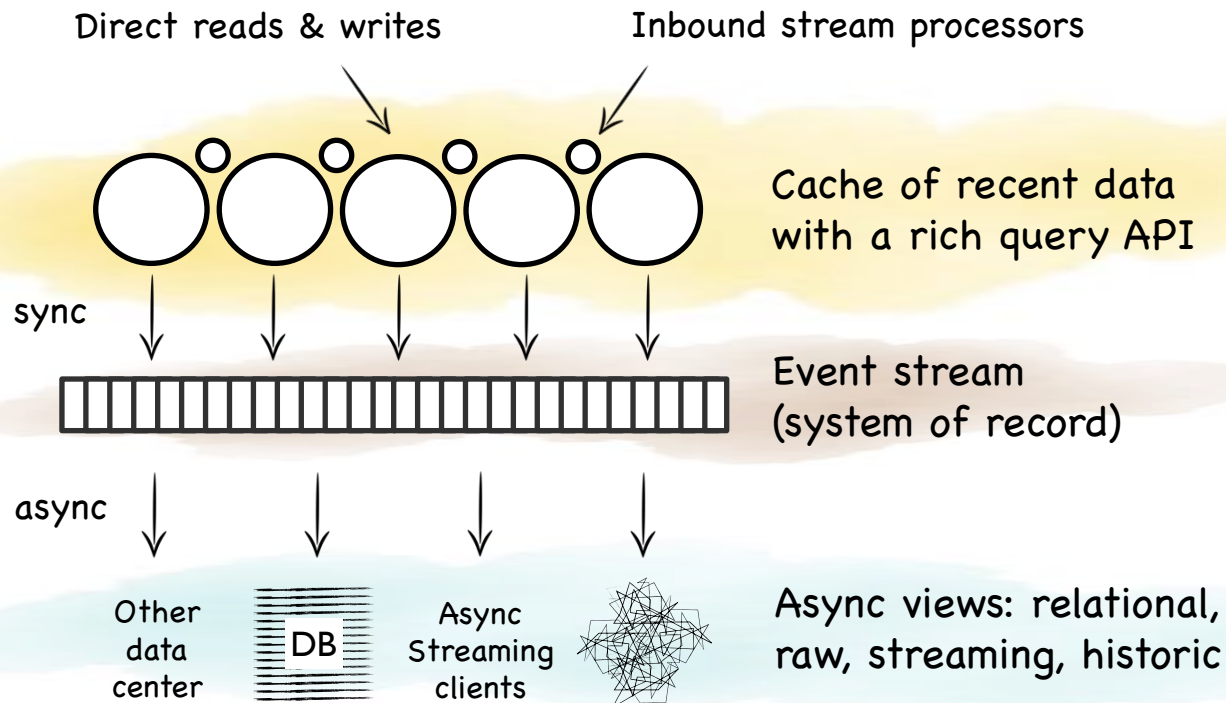
Kafka



Aeron

Messaging as a Backing Store

- Great complement for Coherence
- Write through to a topic. Immutable state.



Hang Tertiary 'VIEWS'

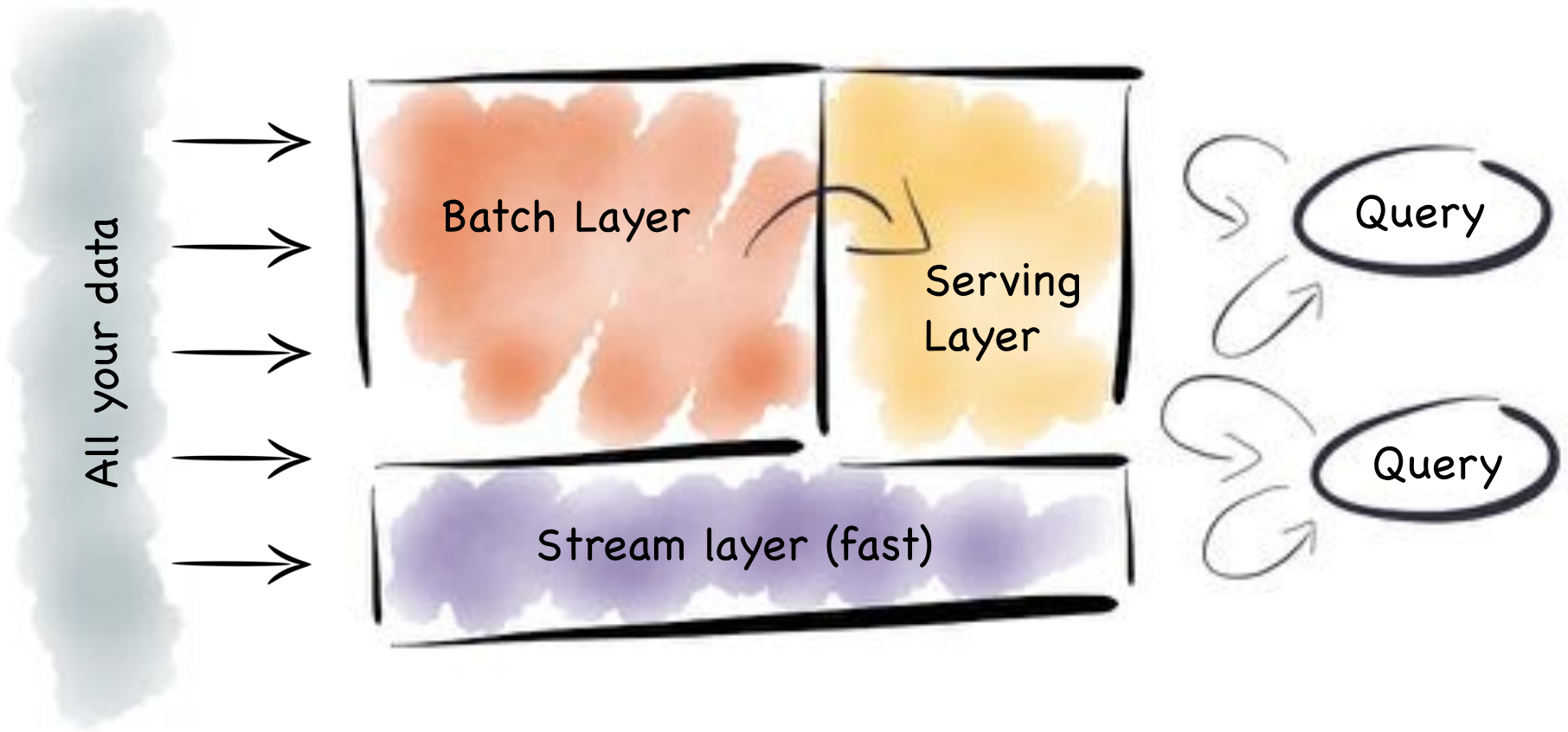
- Search: Elastic Search, Solr
- Graph: Neo4J, OrientDB
- Relational: Oracle, Postgres, Teradata
- Analytic: Exadata, Teradata, Greenplumb
- Document archive: Mongo
- Hadoop: HBase, HDFS, Parquet, avro, PB etc

- Complexity increases with Polyglot Persistence Pattern.
- Replica instantiation is good

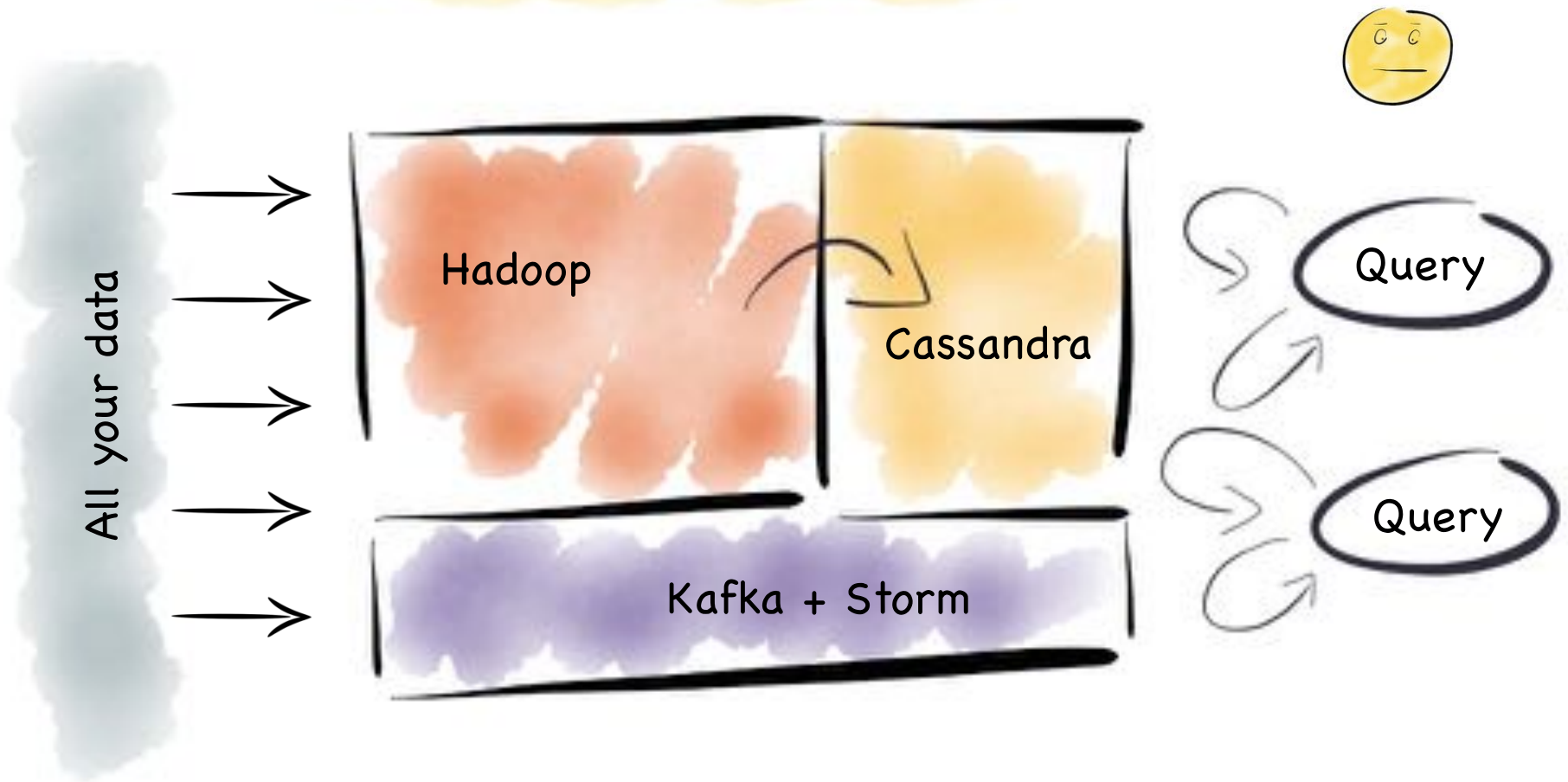
Streams Processors

- Storm
- Samza
- Spark Streaming (microbatch)
- Libraries such as Esper

Lambda Architecture

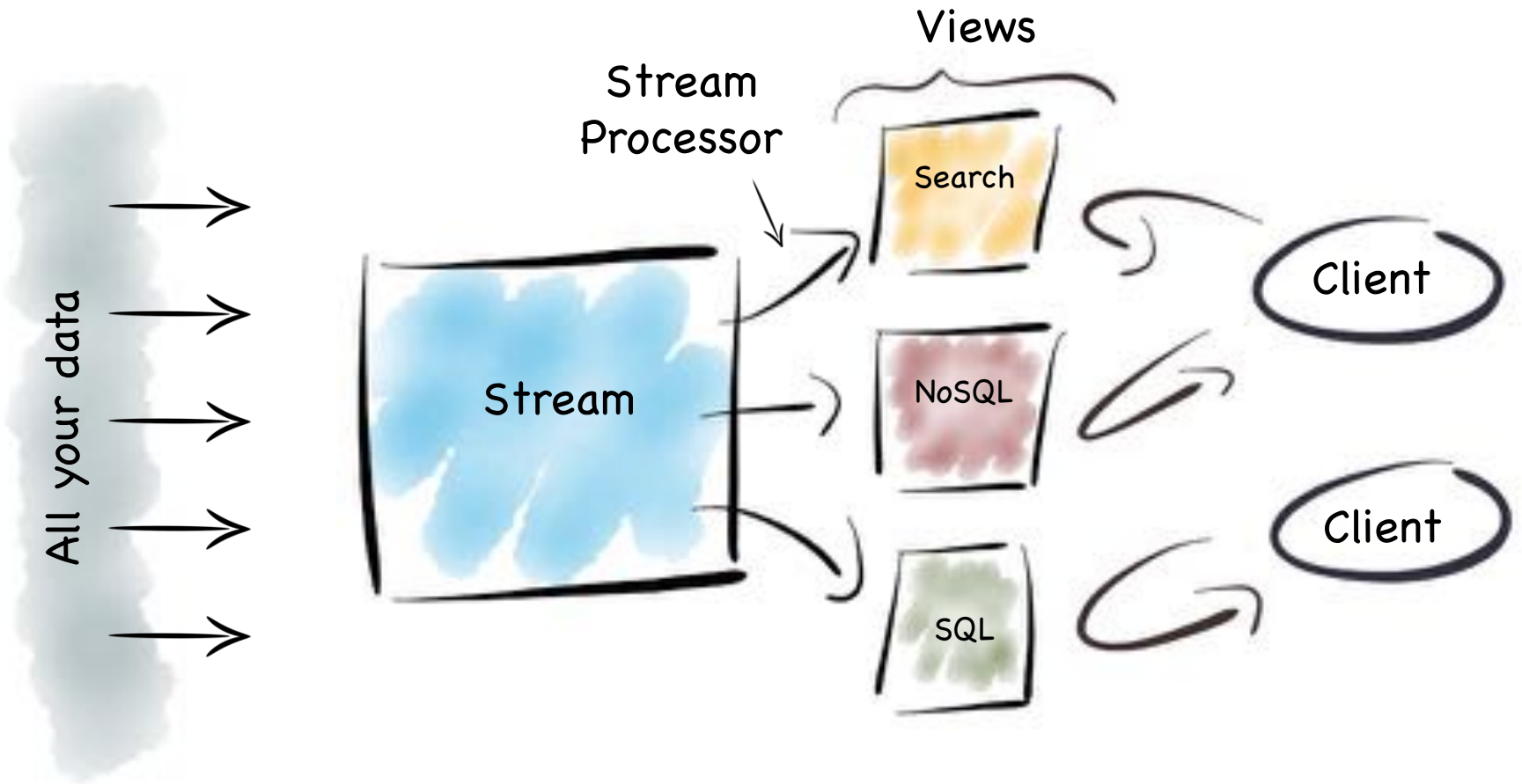


Lambda Architecture

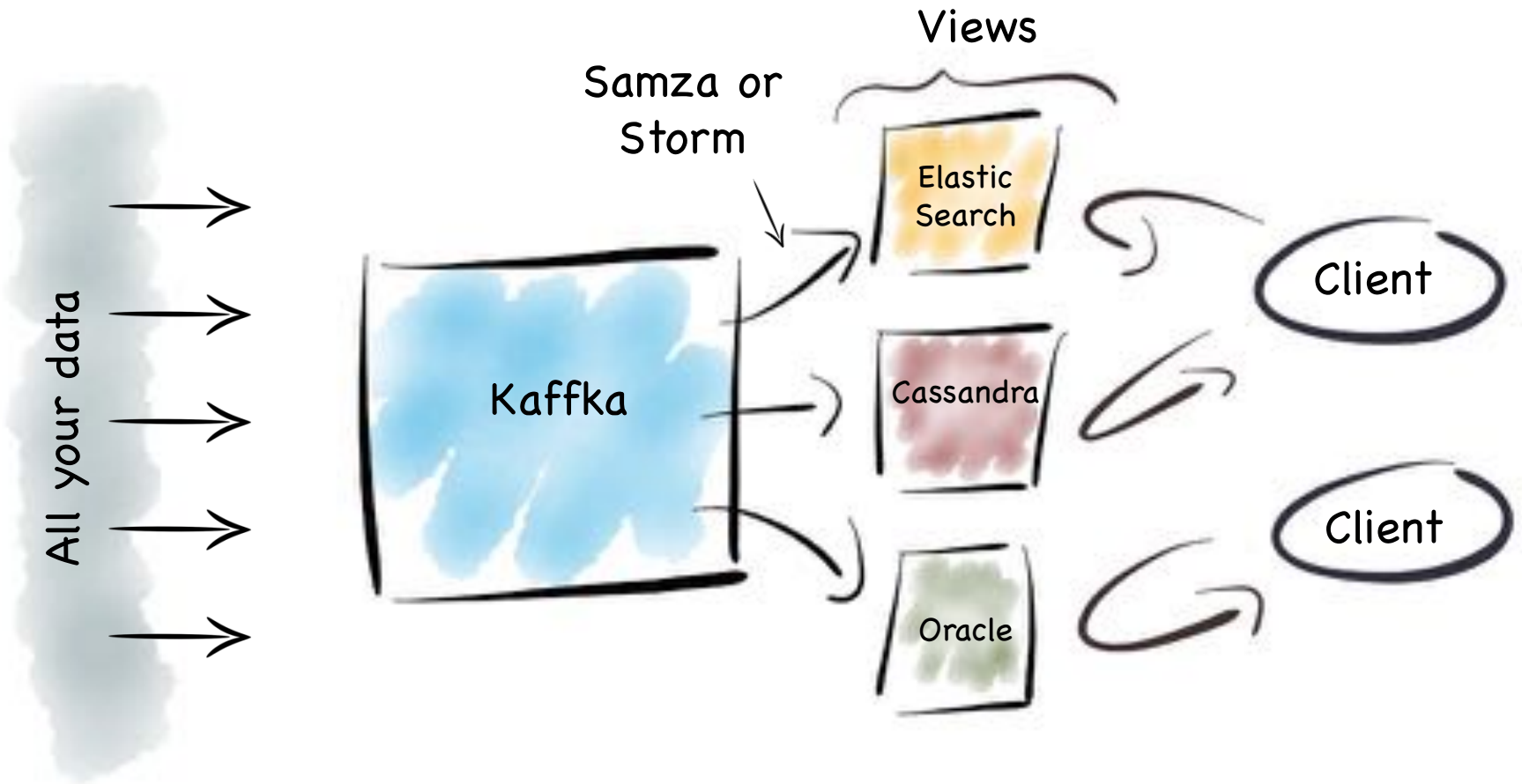


- Cool architecture for use cases that cannot work in a single pass.
- General applicability limited by double-query & double-coding.

Kappa Architecture

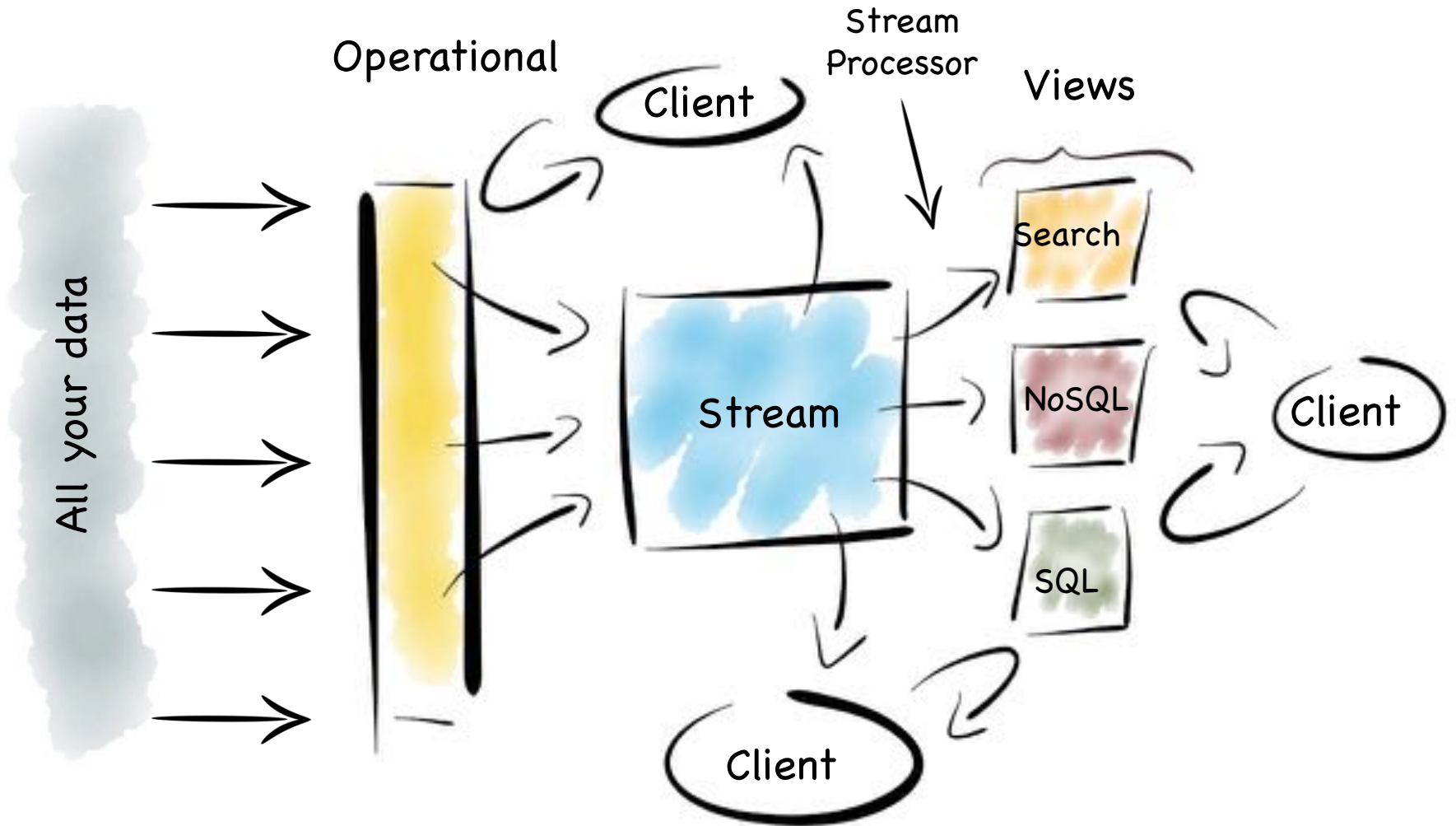


Kappa Architecture

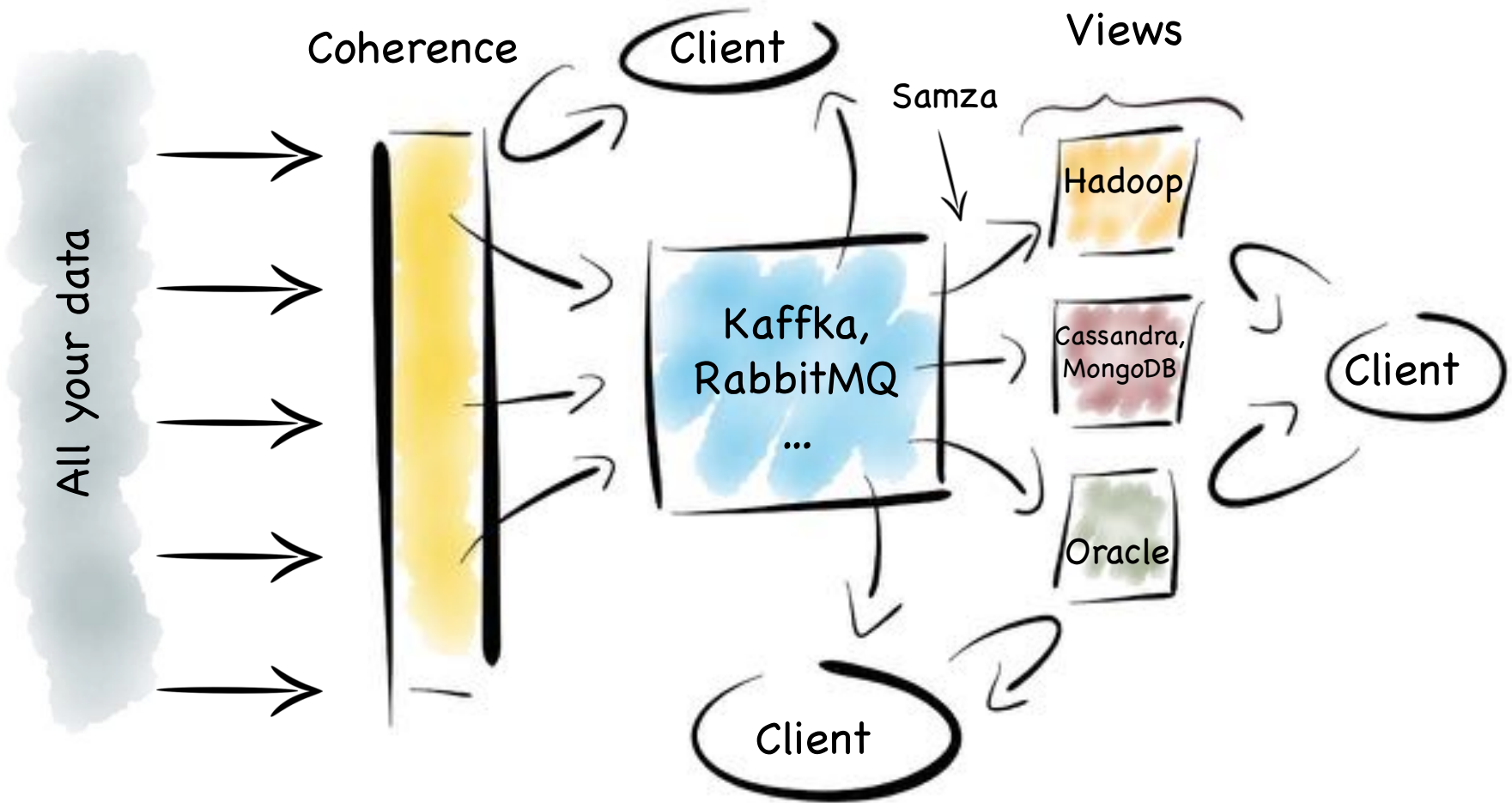


- Simpler choice where stream processors can handle full problem set

Operational /Analytic Bridge



Operational /Analytic Bridge



- Adds coordination layer needed for collaborative updates

Nice Stuff

- Scale-by-Sharding at the front, Scale-by-Replication at the back
- Some “normalisation” at front. Fully denormalised at the back.
- Rewind used to recreate ‘views’



- New Coherence features should make TB+ generally viable
- Sensible caching/processing layer over a simpler store
- NoSQL can provide a sensible interim backing store for larger datasets
- Forms a great write-through layer atop a streaming architecture (Op/Analytic Bridge)

Thanks!