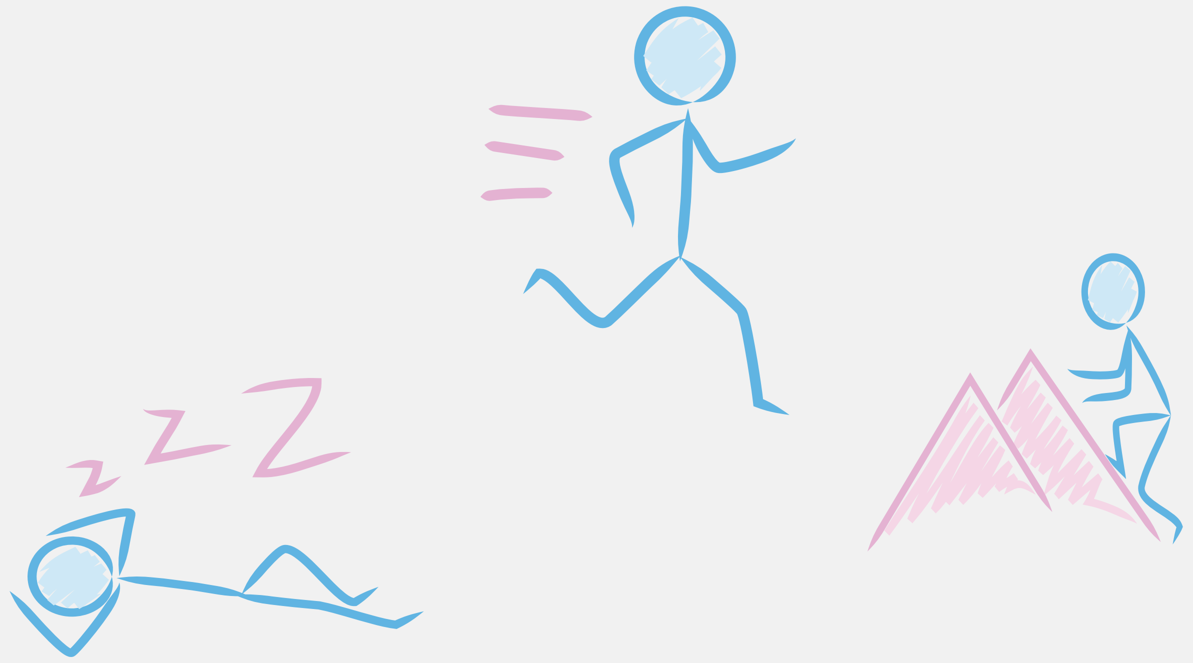
Low latency high throughput streaming using

Apache Apex and Apache Kudu

Dataworks Summit 2017

Ananth Gundabattula

Hypothetical business case



- IOT style enabled end user devices
  - Click Streams
  - Smart phones - Accelerometer, Gyroscope

- Better fraud rules
  - Behavioural analytics by ingesting Human activity recognition feeds
    - Did the user really travel to another Geo ?
    - Does the user generally drive around this area ?

Solution Goals

- Activity recognition stream is processed by a machine learning model to detect human activity
- Data needs to be processed well within a lower end of double digit millisecond time frames
- Data needs to be available for querying within a few milliseconds for operational analytics

Apache Apex introduction

1. Low latency

2. Distributed

3. Streaming

4. **Enterprise grade features**
    - Highly customisable DAG
    - Checkpointing
    - End to End Exactly once
    - Hadoop Security compatible
    - YARN enabled

Apache Kudu introduction

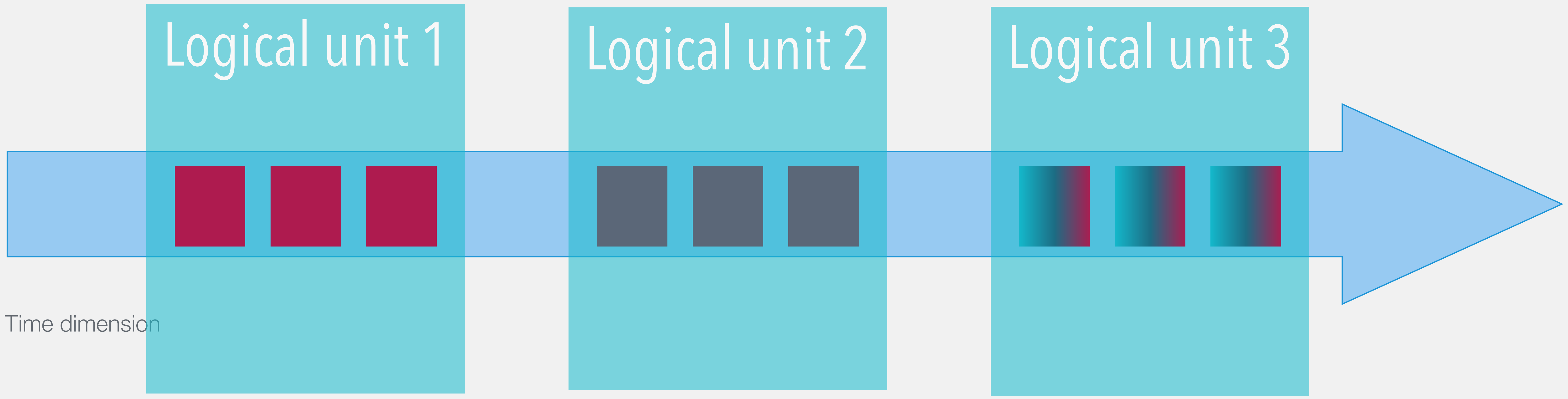
1. Tabular structure

2. Distributed

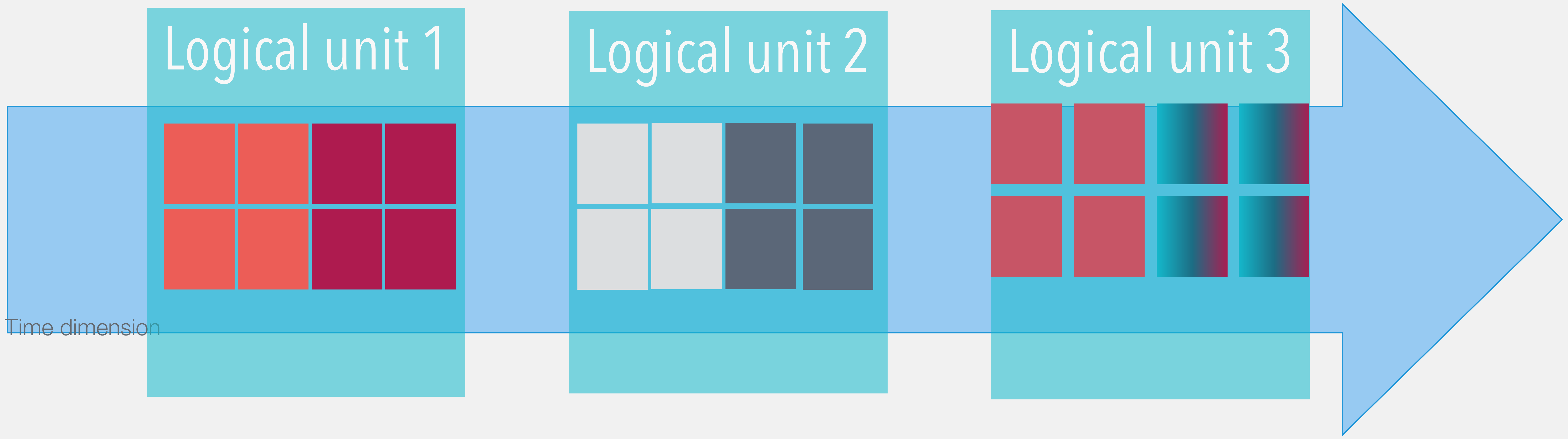
3. Low latency random access

4. **Interesting features**
   - Auto compaction
   - Mutable
   - Columnar optimised storage format
   - Fault tolerant
   - Hadoop ecosystem citizen

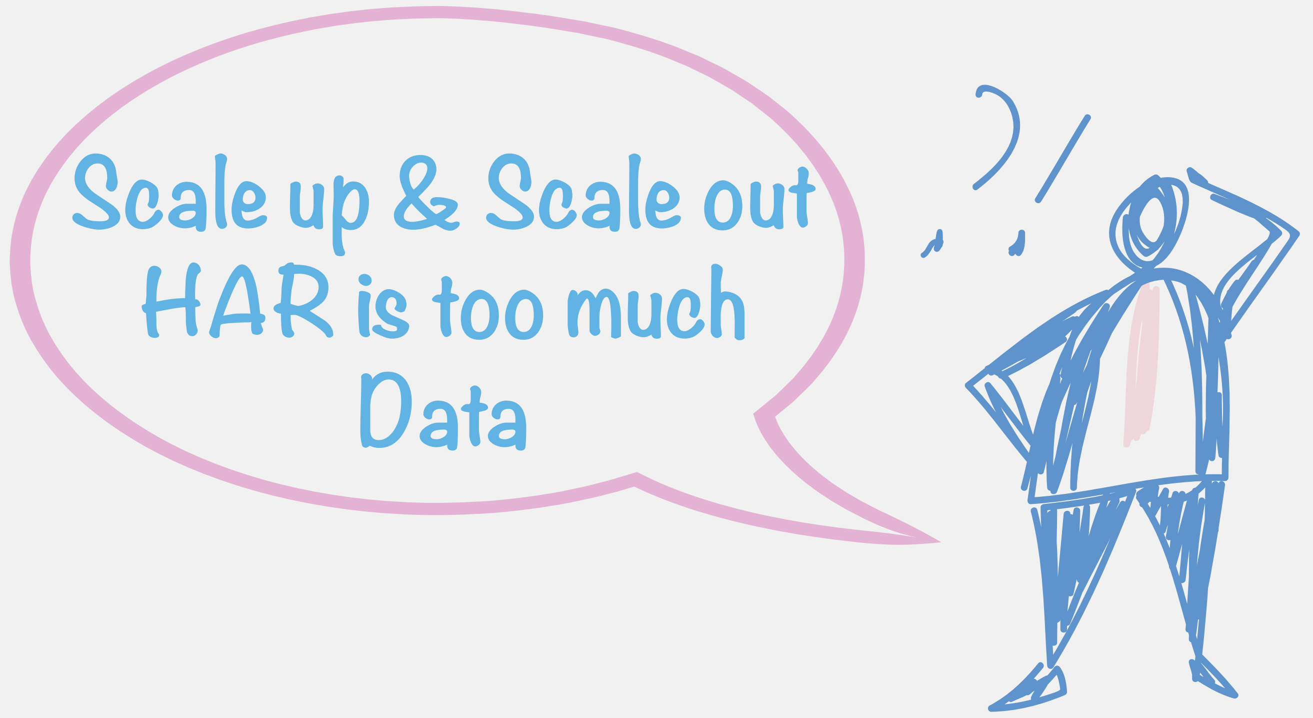
Apex as a **streaming engine**



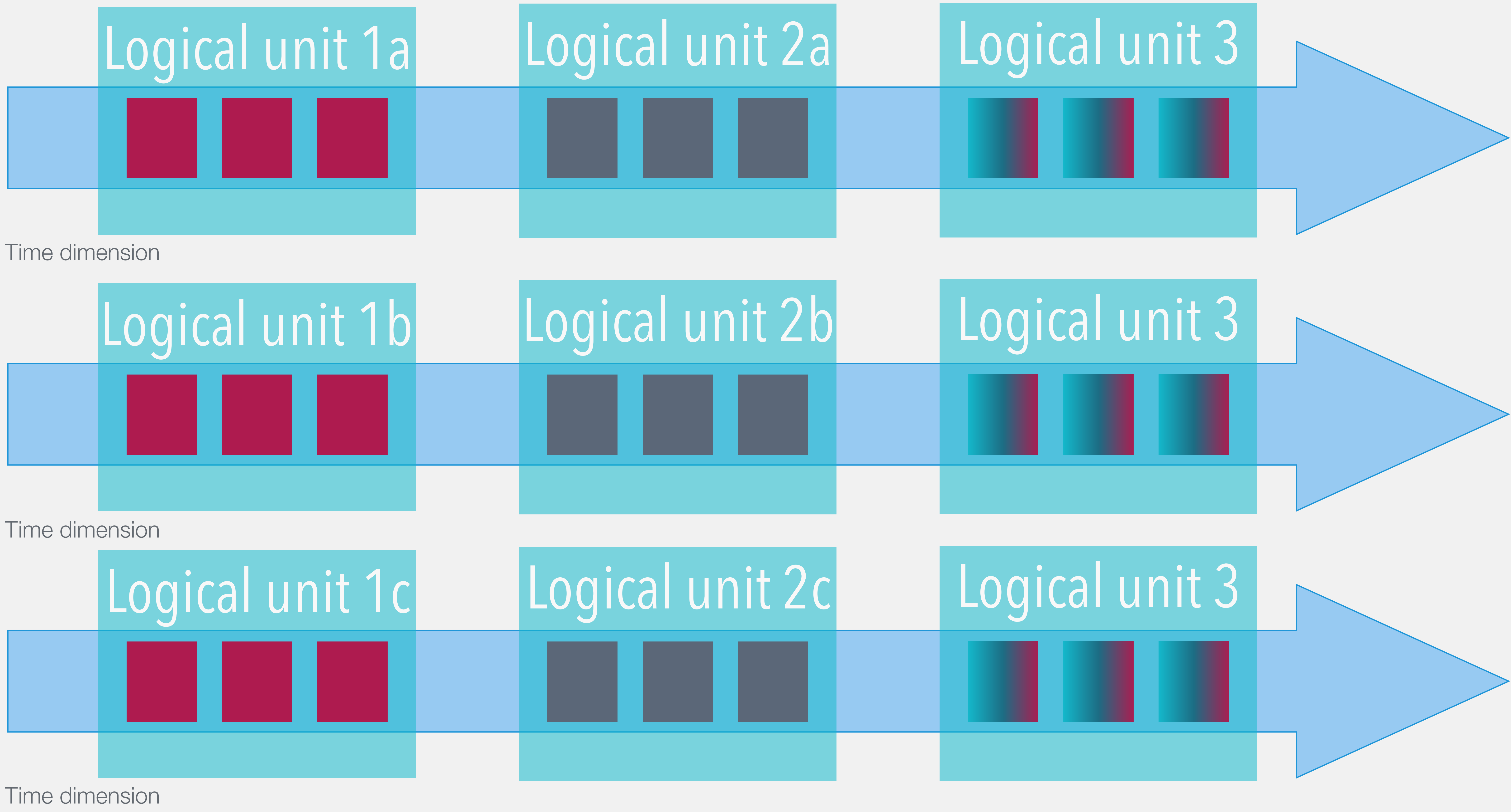
VS



Apex - distributed engine

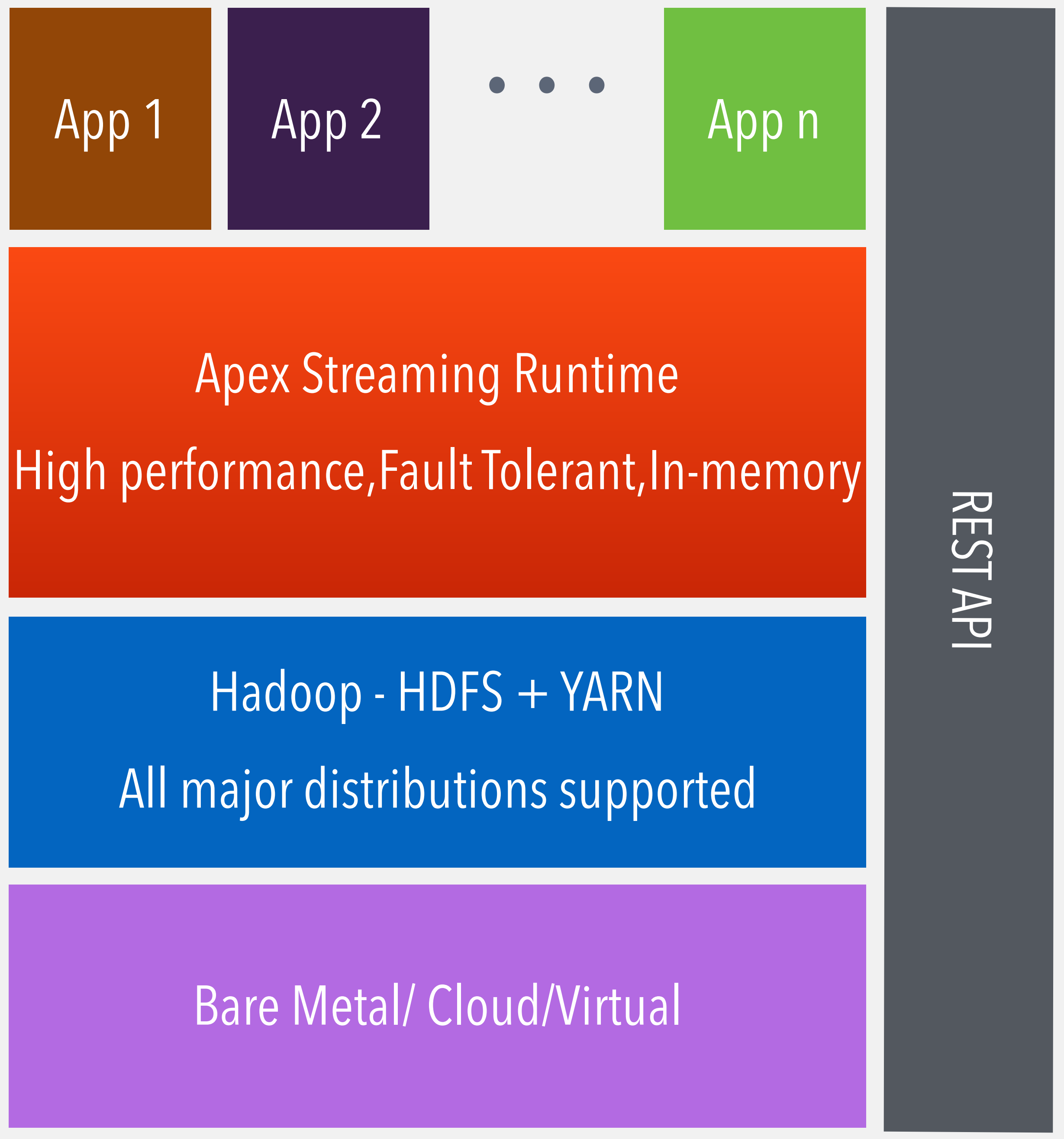


- YARN enabled
  - Resource Managed
- MESOS support on the roadmap



Apex - application layout

APEX STACK



App 1

App 2

. . .

App n

Apex Streaming Runtime

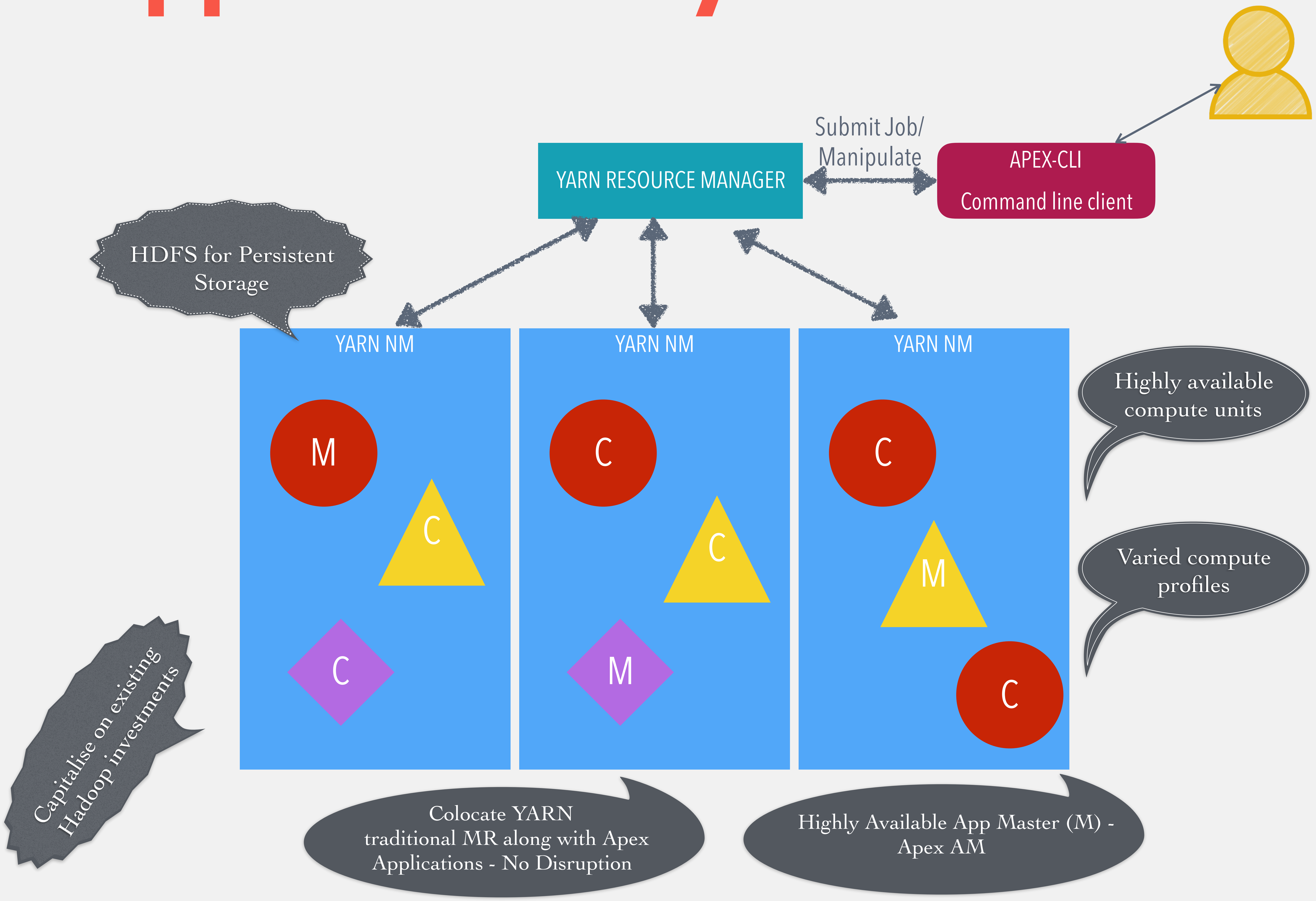
High performance,Fault Tolerant,In-memory

Hadoop - HDFS + YARN

All major distributions supported

Bare Metal/ Cloud/Virtual

REST API



YARN RESOURCE MANAGER

Submit Job/ Manipulate

APEX-CLI

Command line client

YARN NM

YARN NM

YARN NM

HDFS for Persistent Storage

Highly available compute units

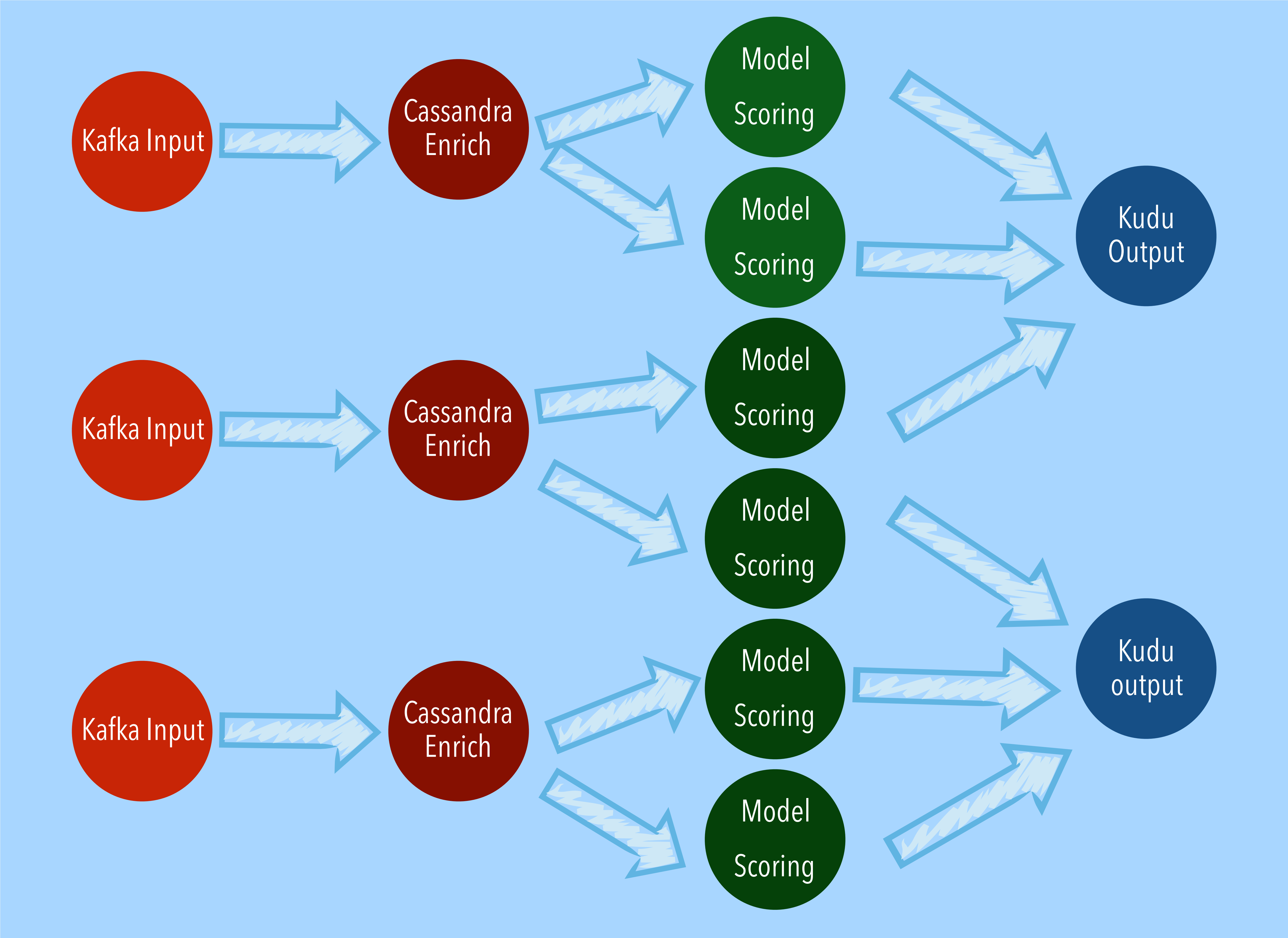
Varied compute profiles

Capitalise on existing Hadoop investments

Colocate YARN traditional MR along with Apex Applications - No Disruption

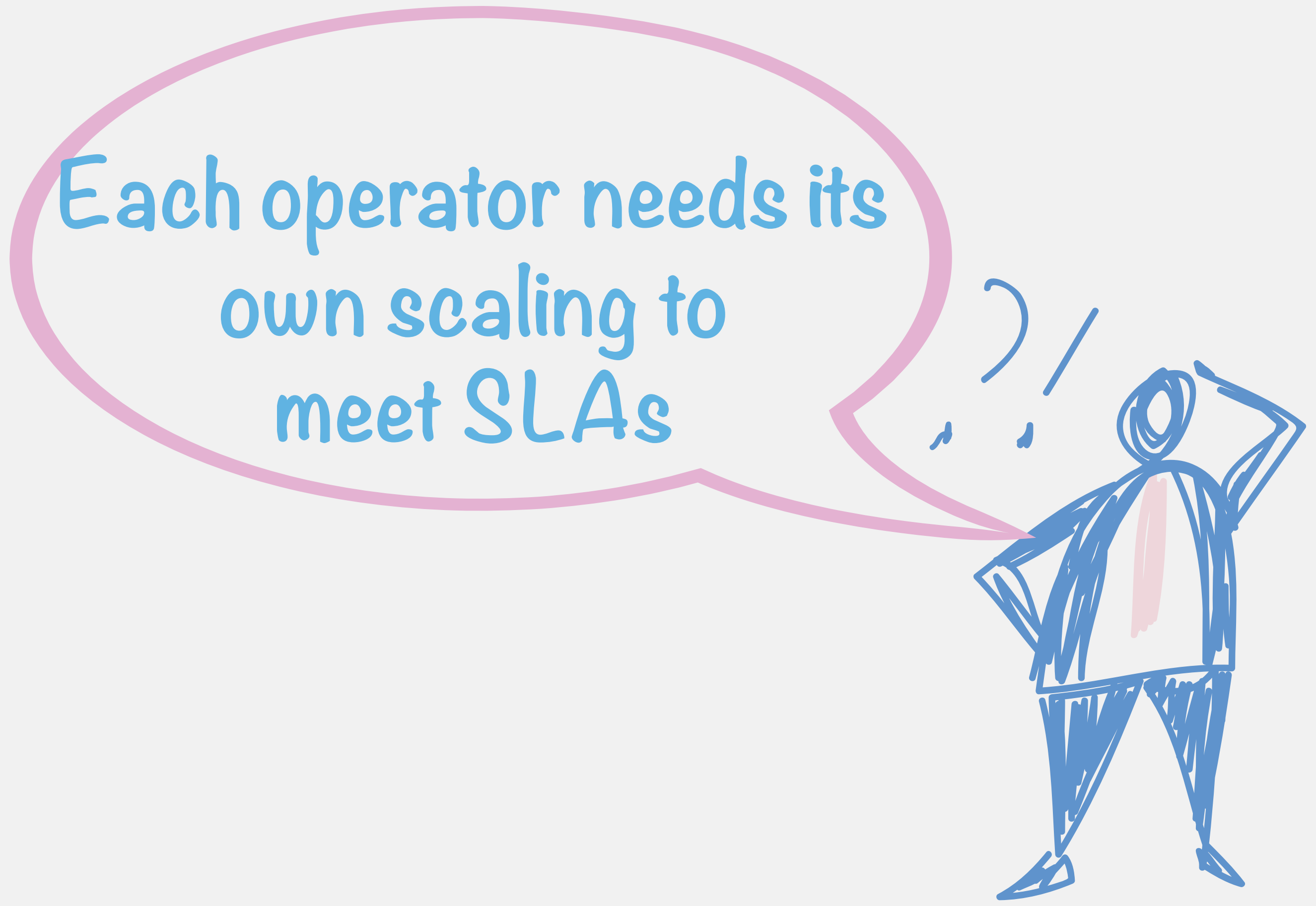
Highly Available App Master (M) - Apex AM

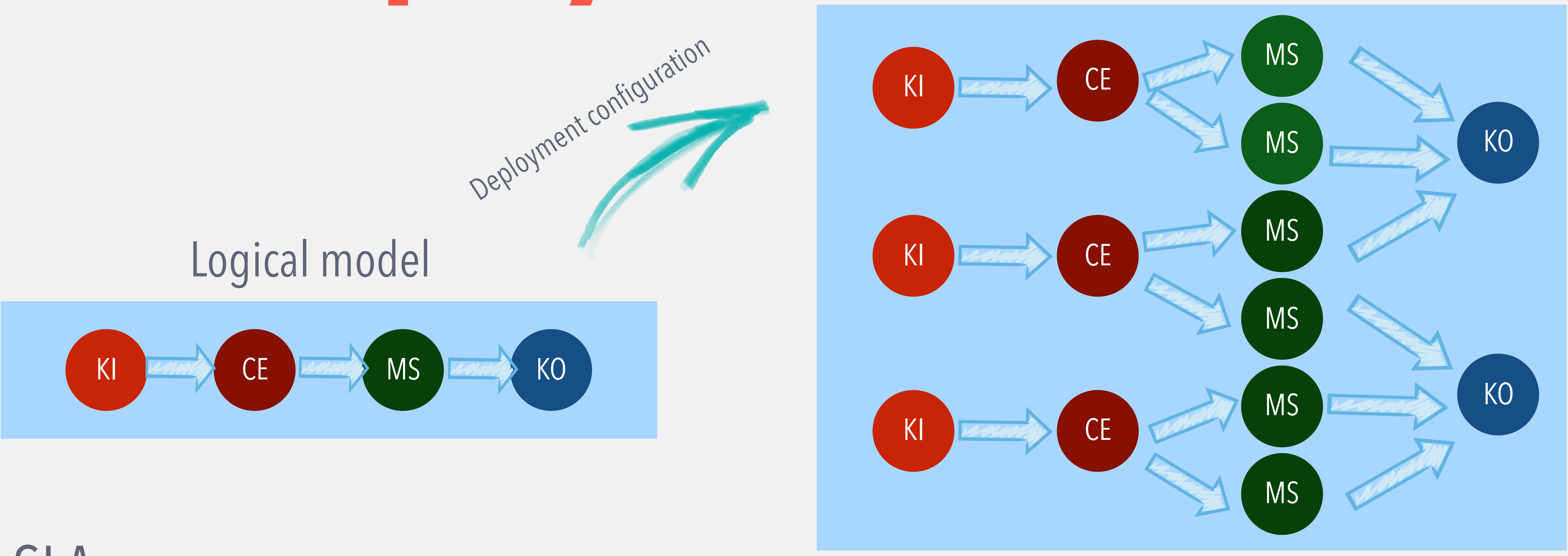
Apex application development model



- Stream is a sequence of data tuples
- An Operator consumes one or more input streams , processes tuples using custom business logic and emits to one or more output streams
- DAG is made up of operators and streams
- Rich collection of operators available from Apache Malhar
  - NOSQL - Cassandra, Geode ..
  - Kudu
  - Relational - JDBC
  - Messaging - Kafka, JMS , Solace
  - File Systems - HDFS , S3, NFS
  - Nifi
  - ….

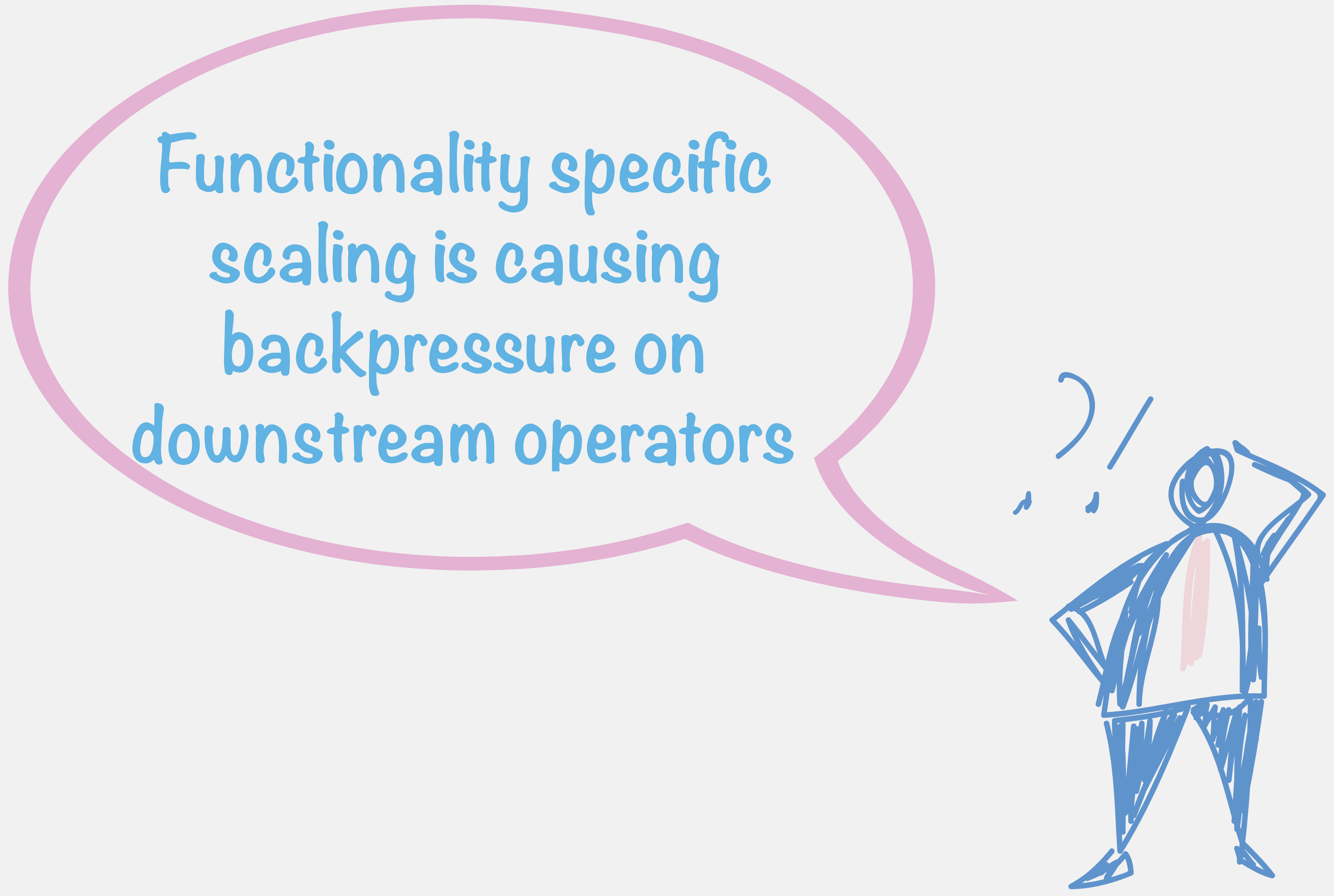
Apex application **deployment model**





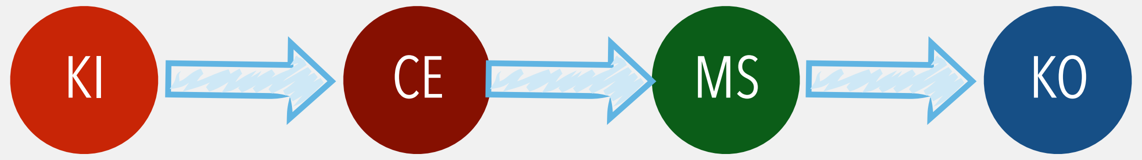
- Non-intrusive model to meet overall SLAs
  - Different operators can be configured independently to meet SLA needs.
    - Compute intensive vs I/O intensive
- Custom stream codecs enable configurable tuple routing patterns

Unifiers

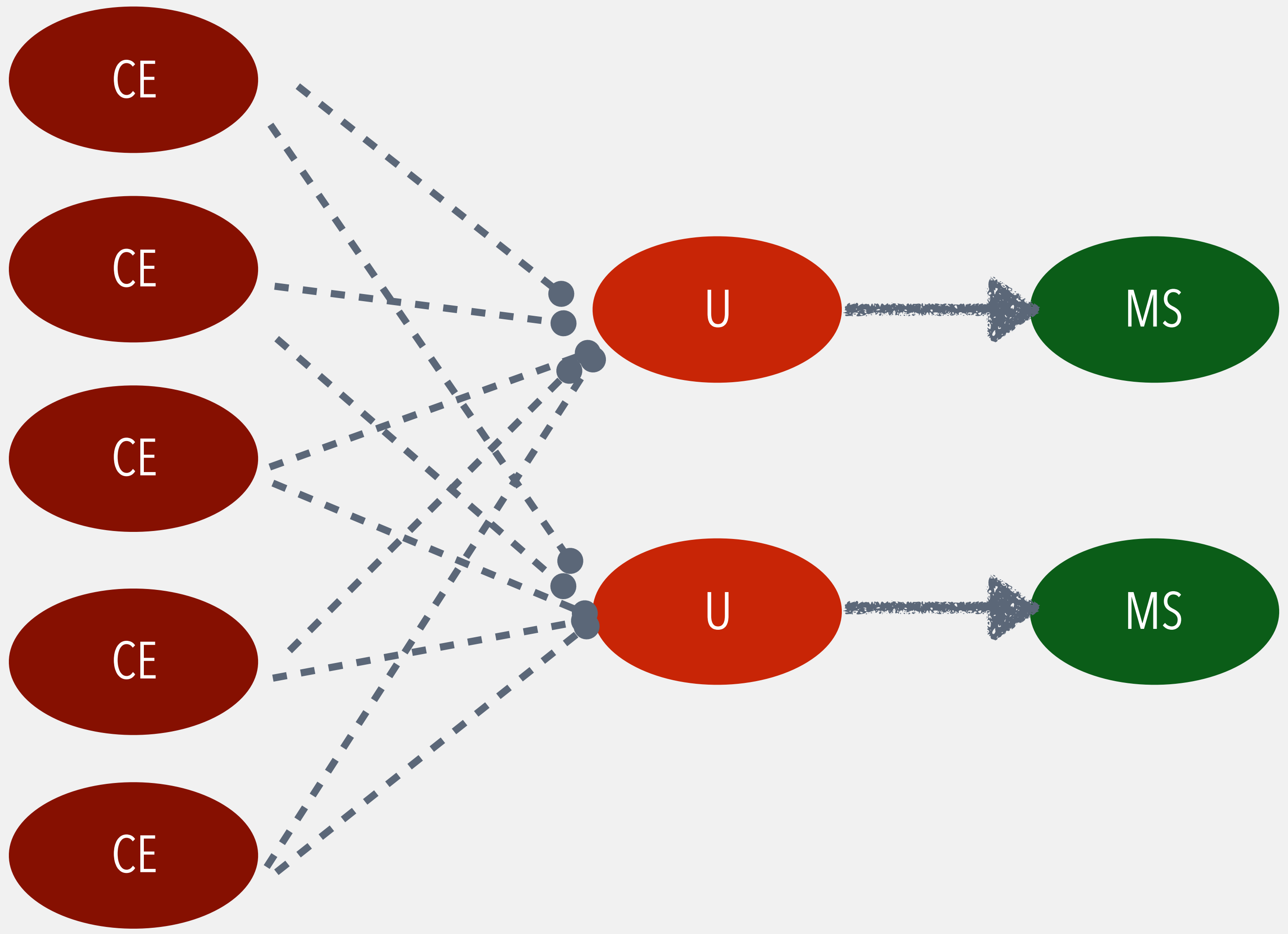


Functionality specific scaling is causing backpressure on downstream operators

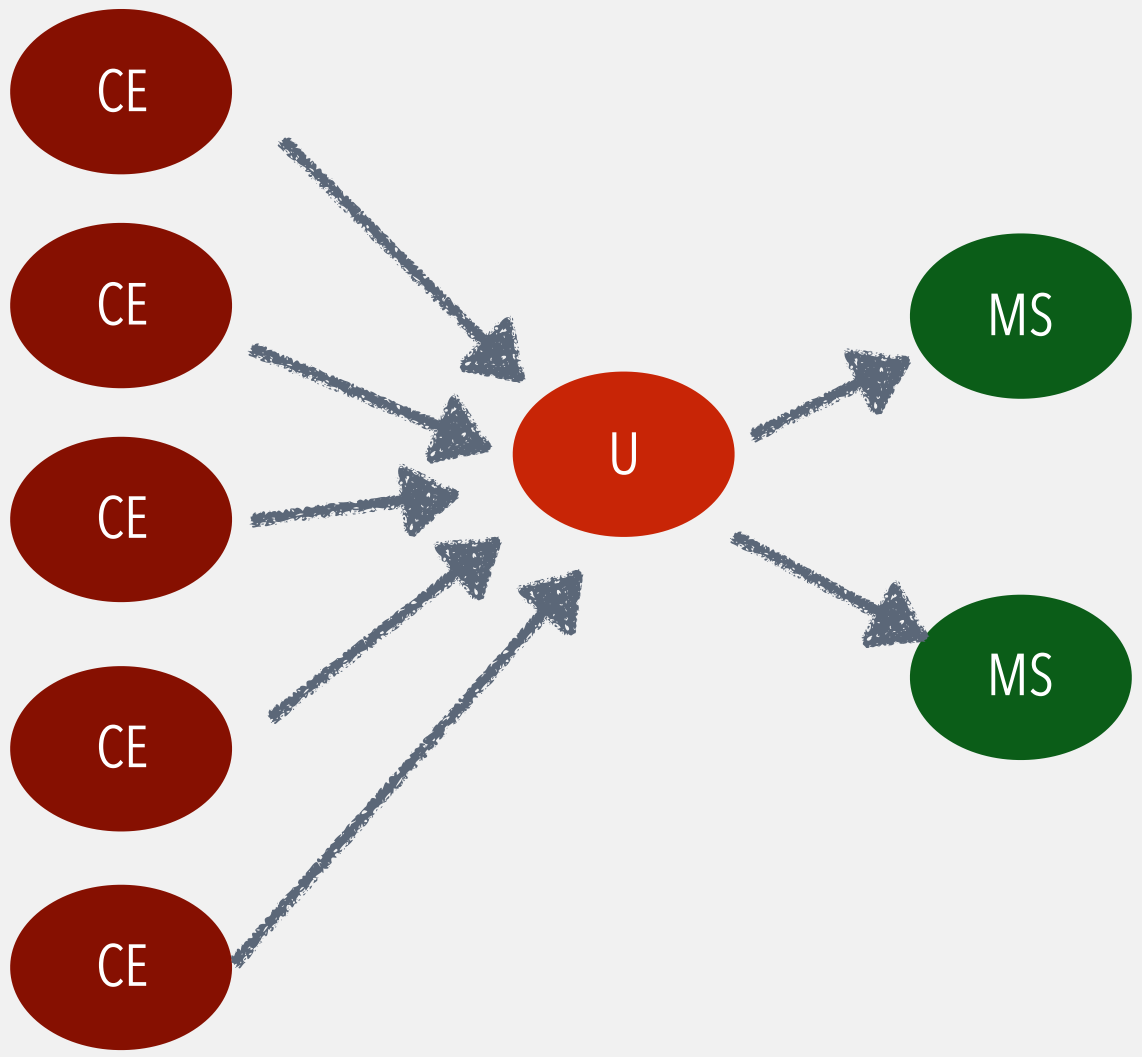
Logical Plan



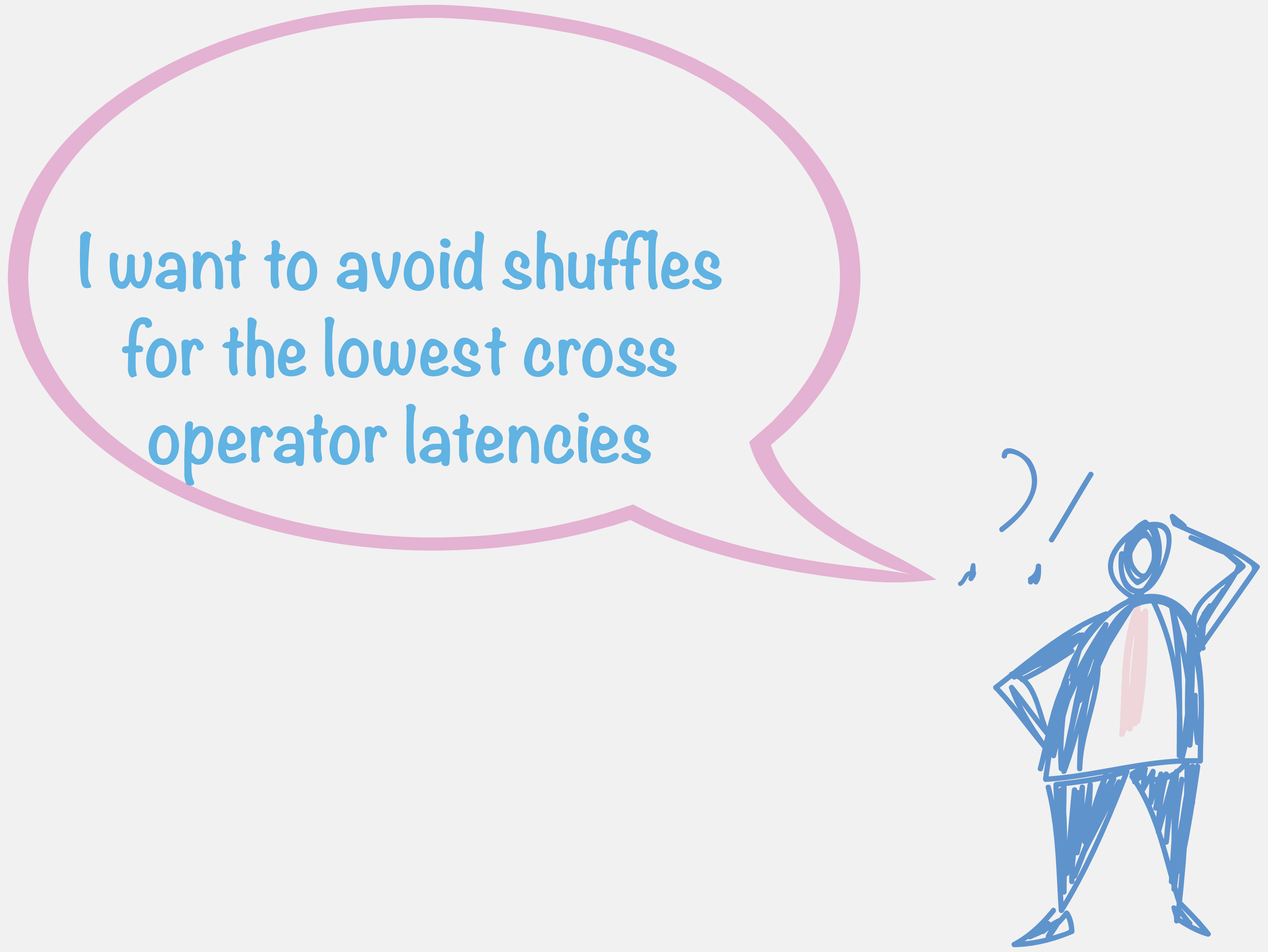
Scaled up unifiers



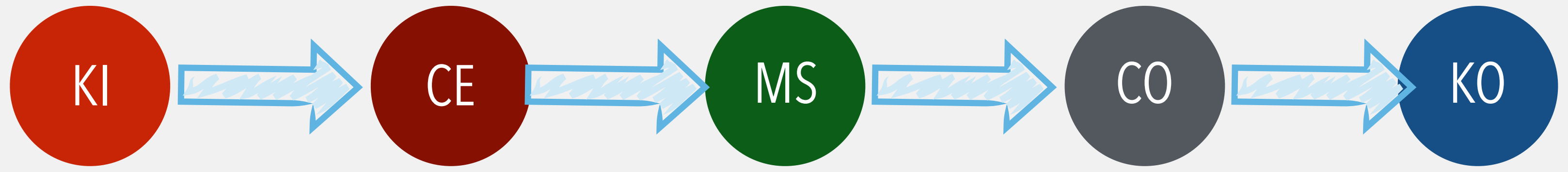
Bottleneck @Unifier



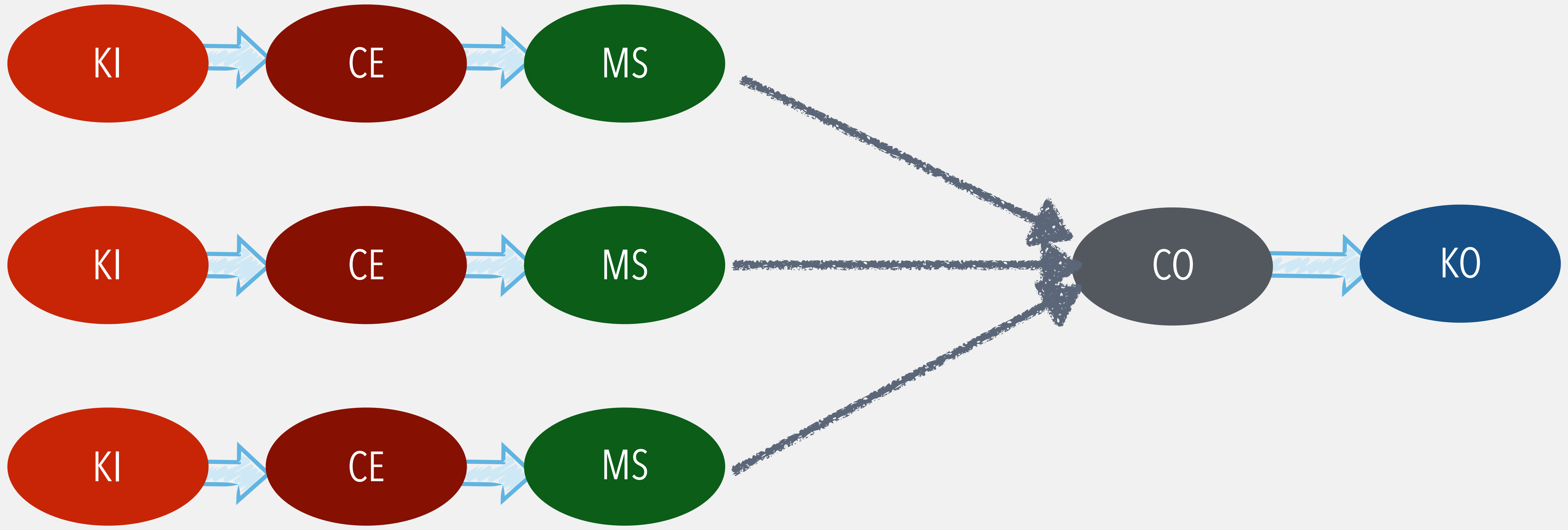
Parallel partitioning


I want to avoid shuffles for the lowest cross operator latencies

Logical Plan

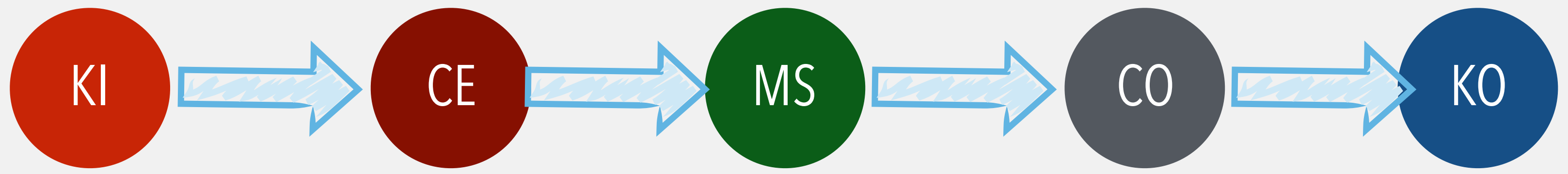


Parallel Partitioning



Dynamic partitioning

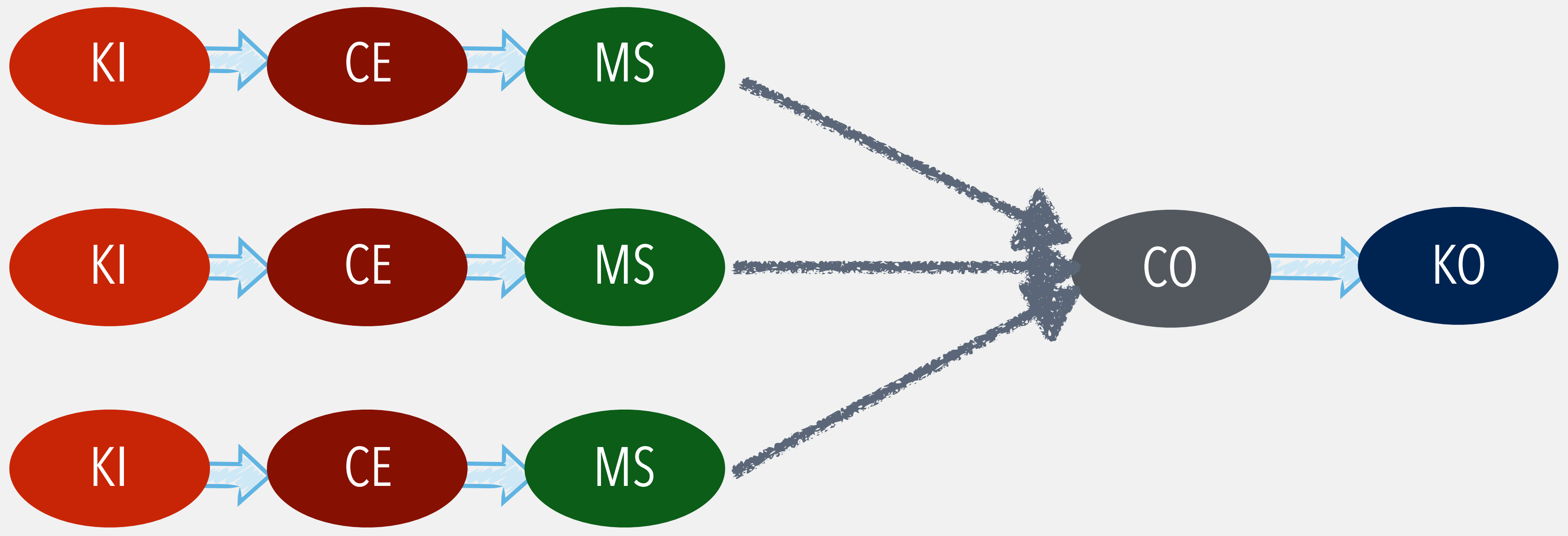
Logical Plan



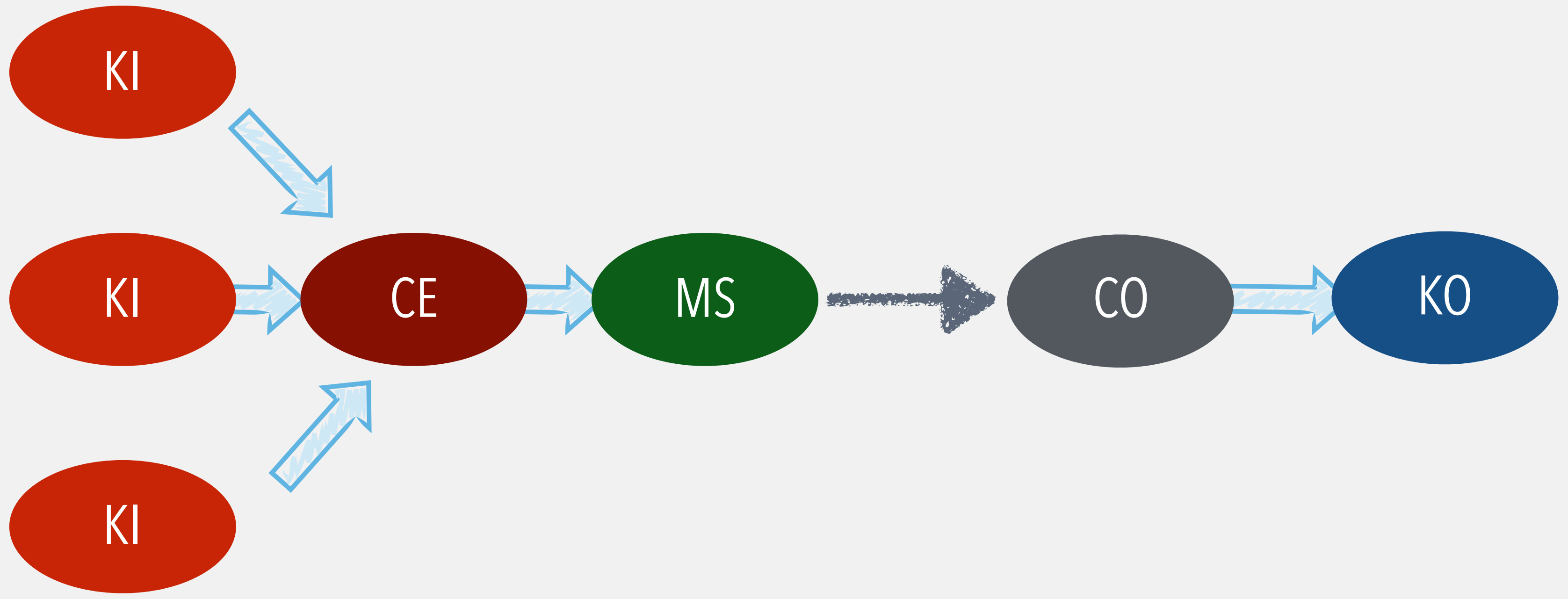
- Utilise hardware for nightly batch compute needs
  - Ex: Zip code based average driving speeds for HAR Features

Dynamic Partitioning

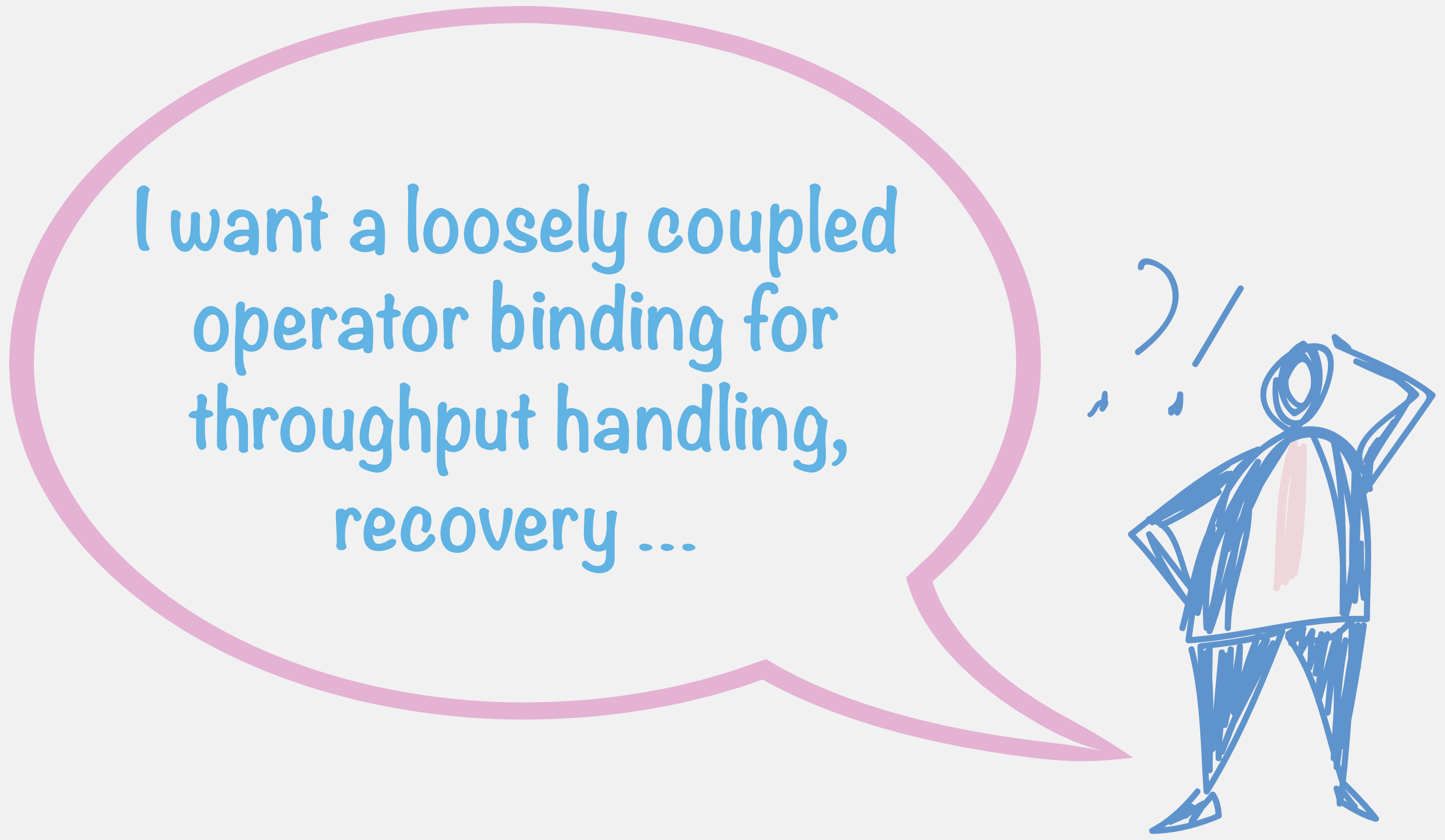
Daytime topology

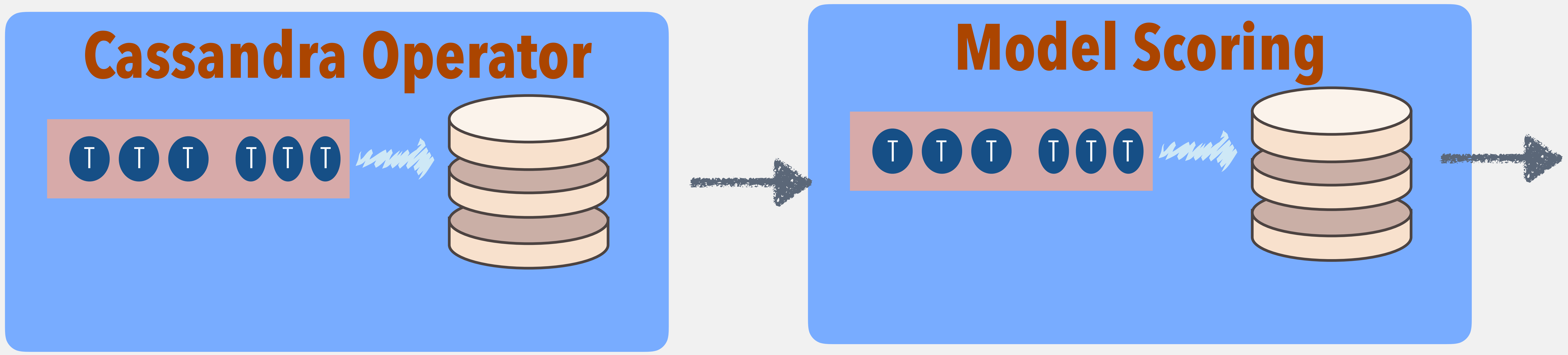


Nighttime topology



Pub sub

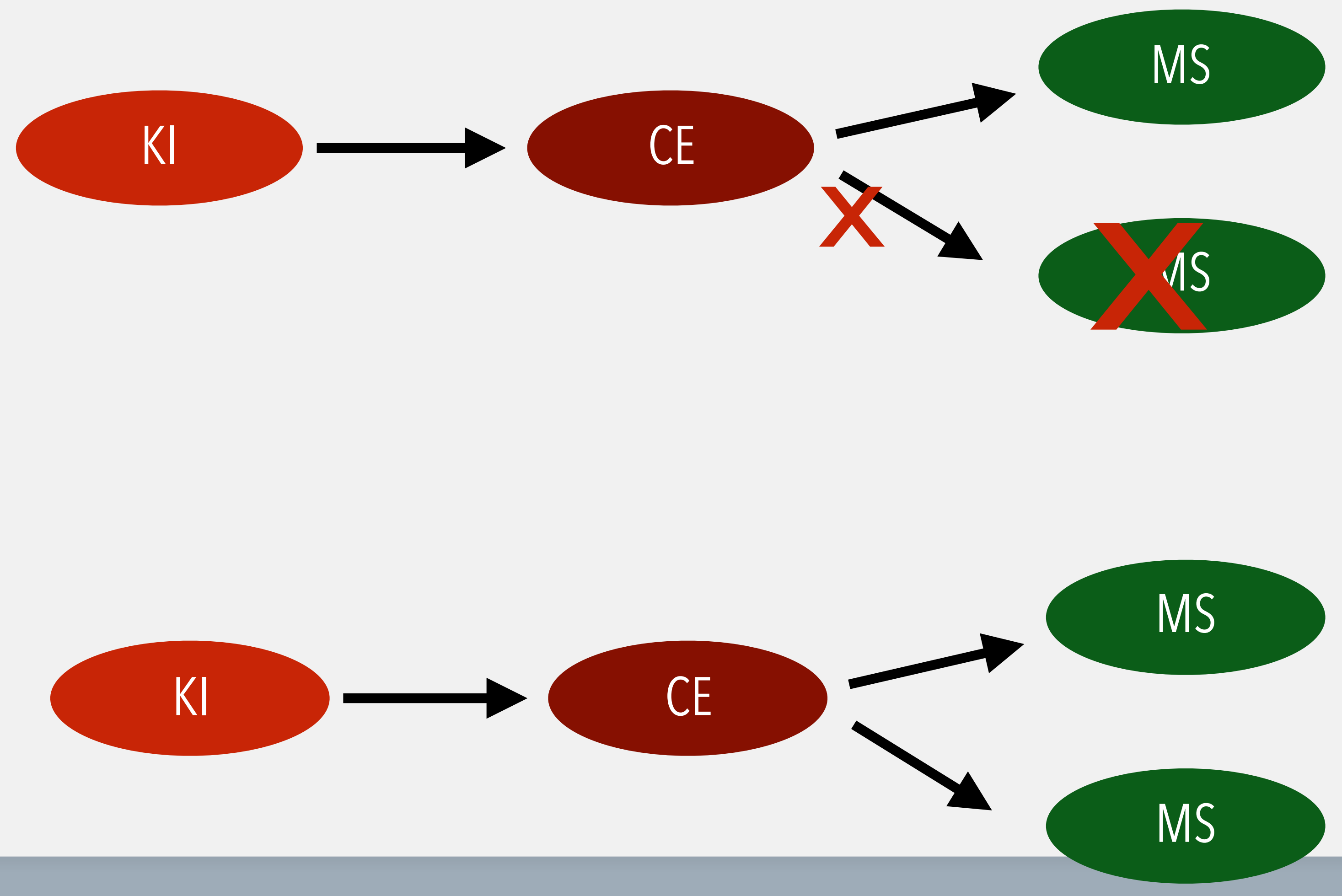

I want a loosely coupled operator binding for throughput handling, recovery ...


Cassandra Operator

Model Scoring

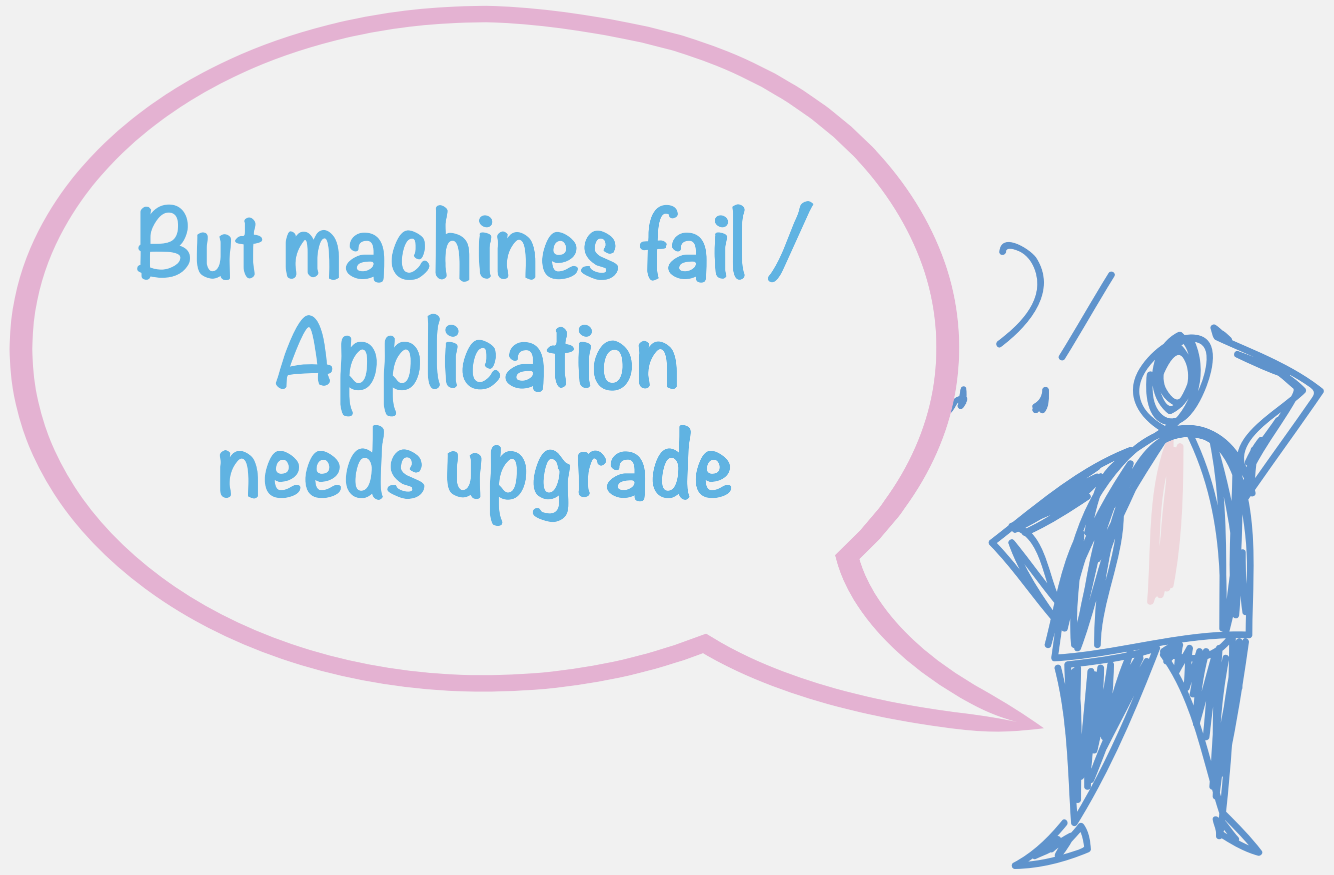
- High performant in-memory pub-sub messaging
- Provides ordering & idempotency for failure scenarios
- Buffers tuples in memory until the tuples are committed
- Spills to disk in back-pressure scenarios

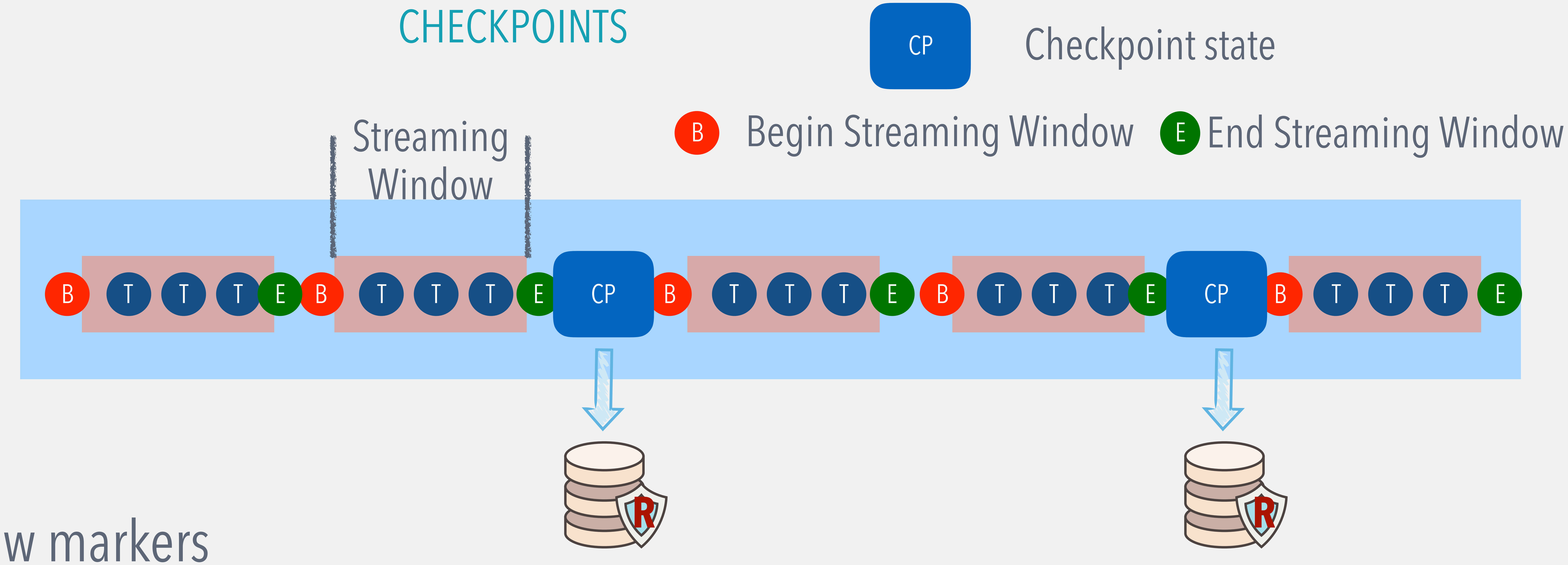
Recoverability in parts of DAG


KI → CE → MS, MS

KI → CE → MS, MS

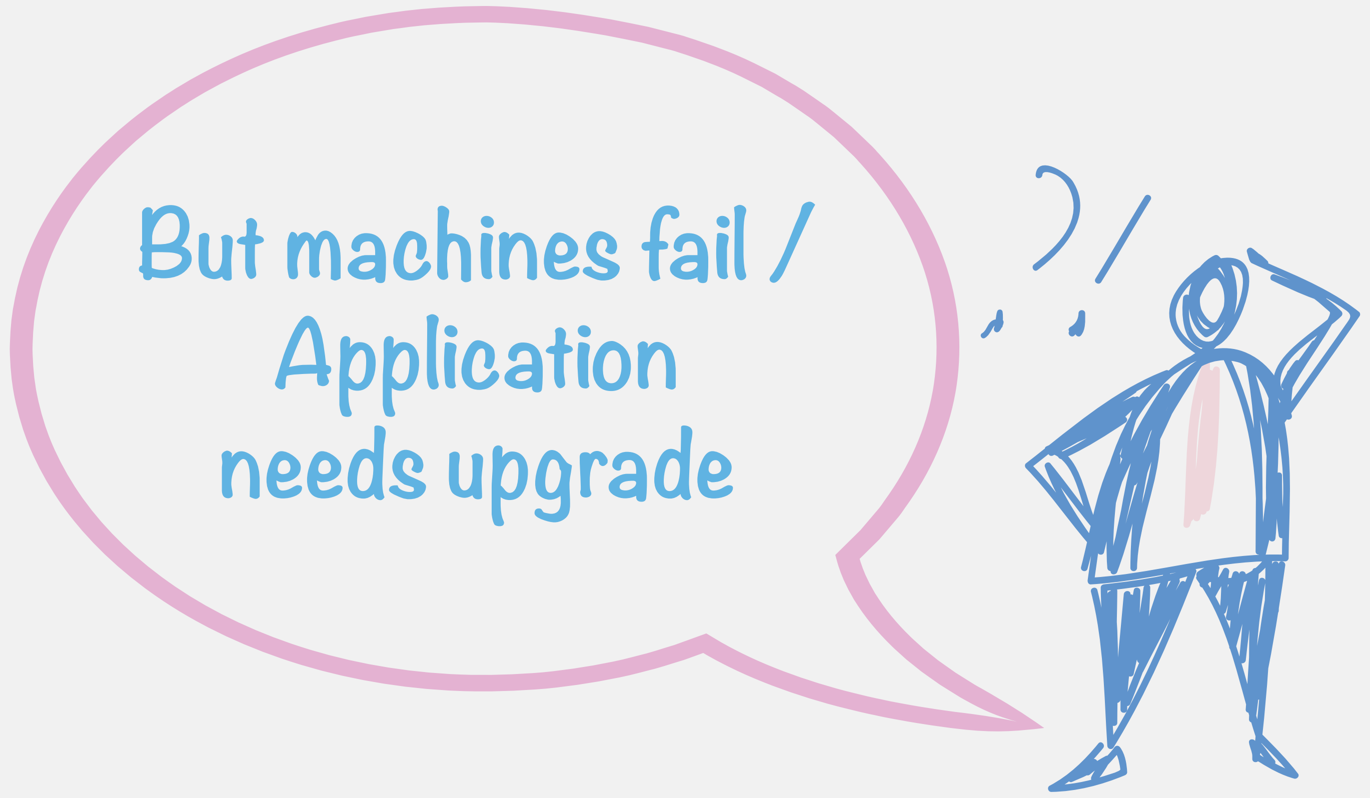
Checkpointing

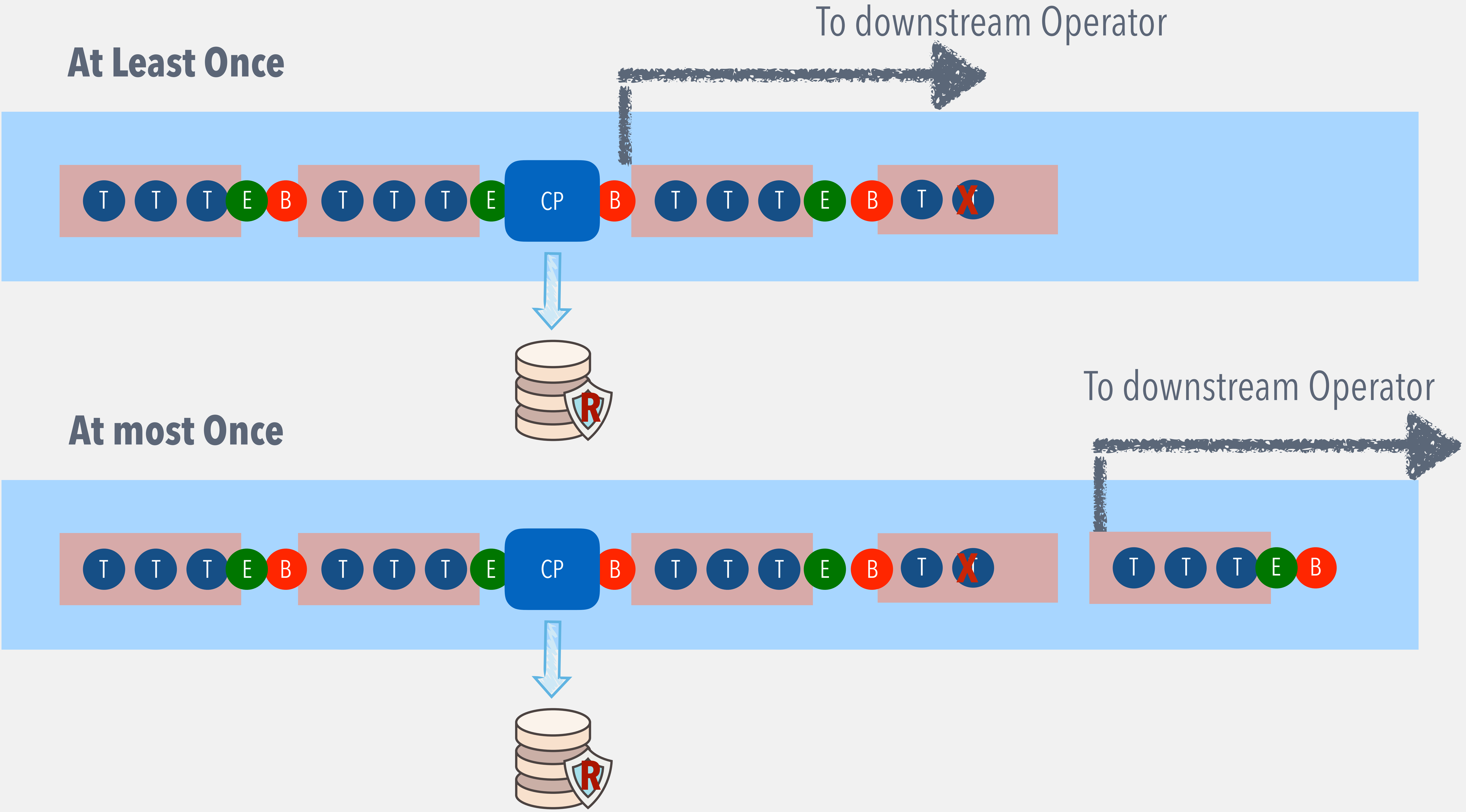




- Non-intrusive streaming window markers
- In-memory processing of data & checkpointing at streaming window boundaries
- Configurable checkpoint store - HDFS backed store replicated & highly available
- One or multiple windows (configurable) make a checkpoint boundary
- Persist non-transient & Operator specific checkpointing data structure - Ex: Kafka : (C,T,P,O)

Processing guarantees

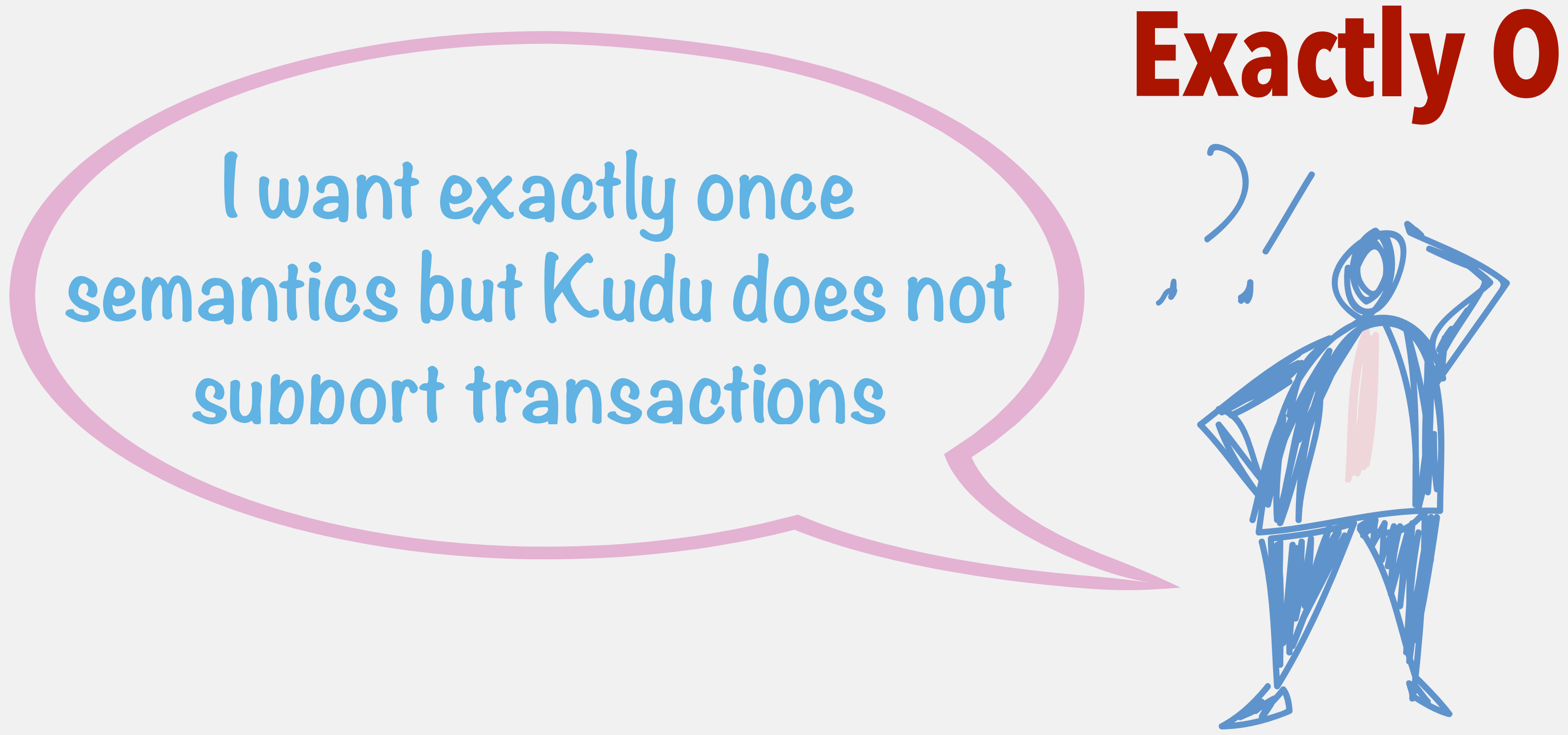



At Least Once
To downstream Operator
T T T E B T T T E CP B T T T E B T X
At most Once
To downstream Operator
T T T E B T T T E CP B T T T E B T X T T T E B
R

Exactly Once = At least once + Idempotency (Pub-Sub) + Operator logic

Kudu output operator - Exactly once

Exactly Once = At least once + Idempotency (Pub-Sub) + Operator logic


I want exactly once semantics but Kudu does not support transactions

Business Logic callback to
Exactly detect already written records

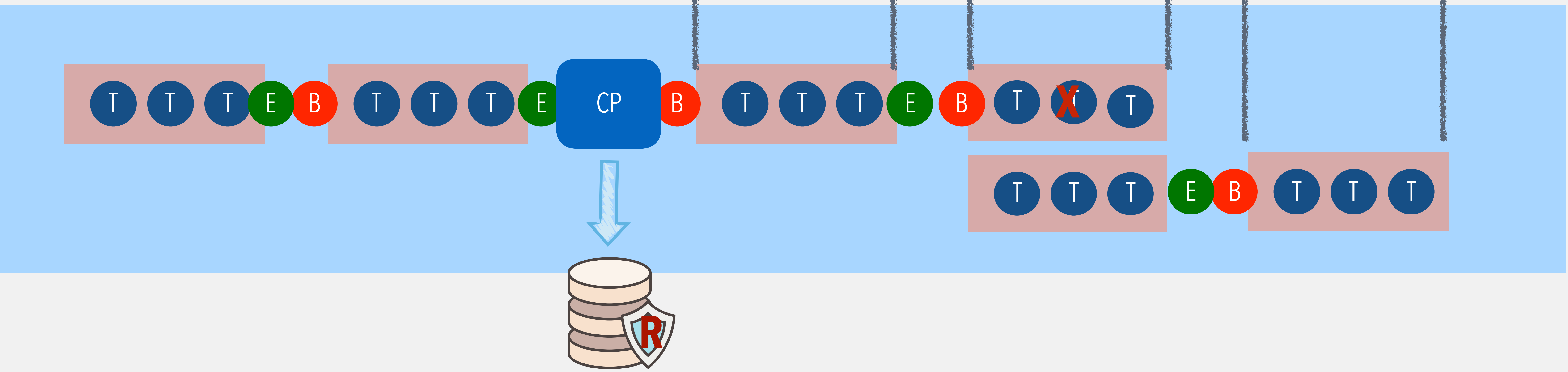
Automatically Skipped

Safe Mode Window/s

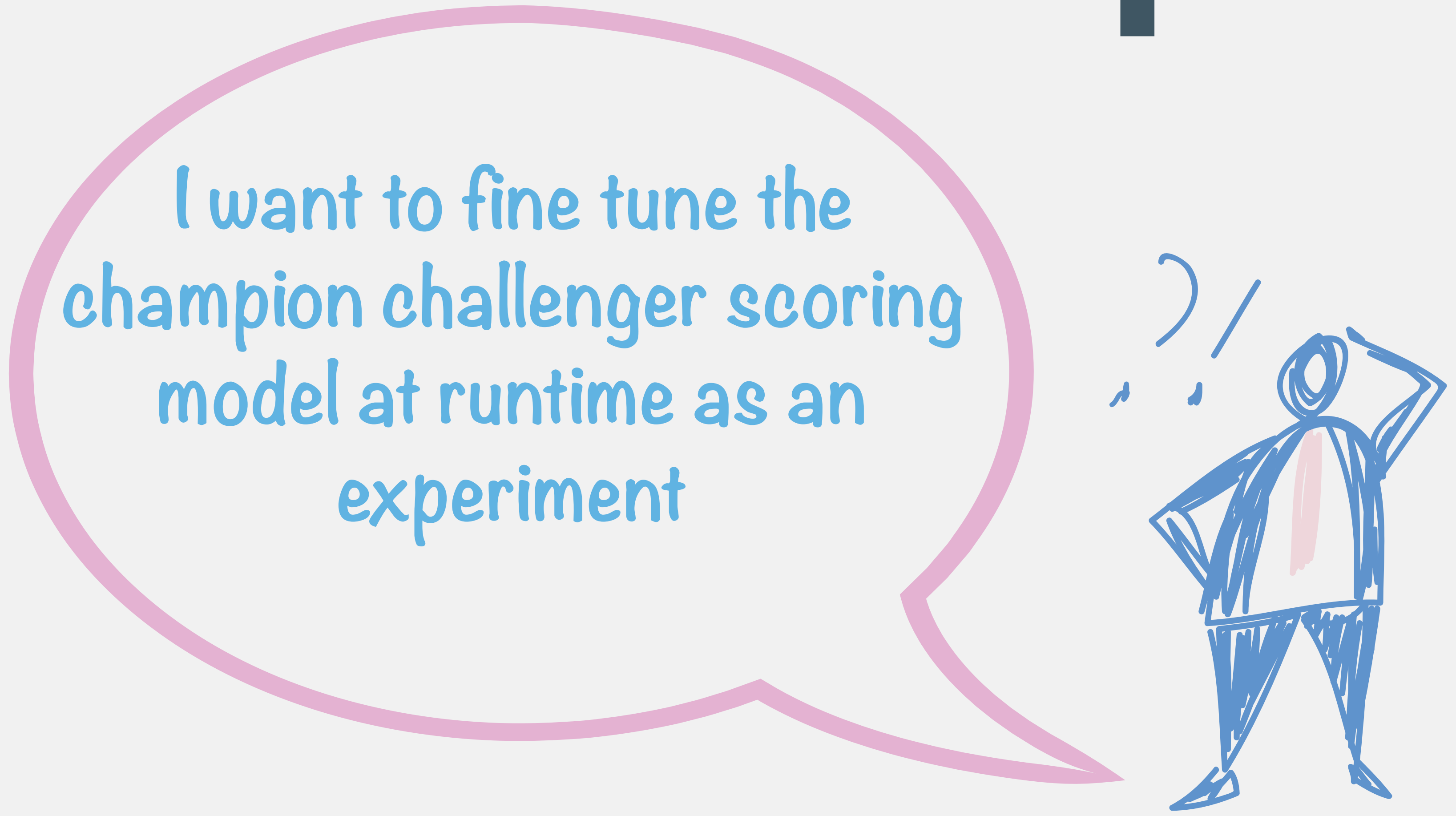
Reconciling Mode

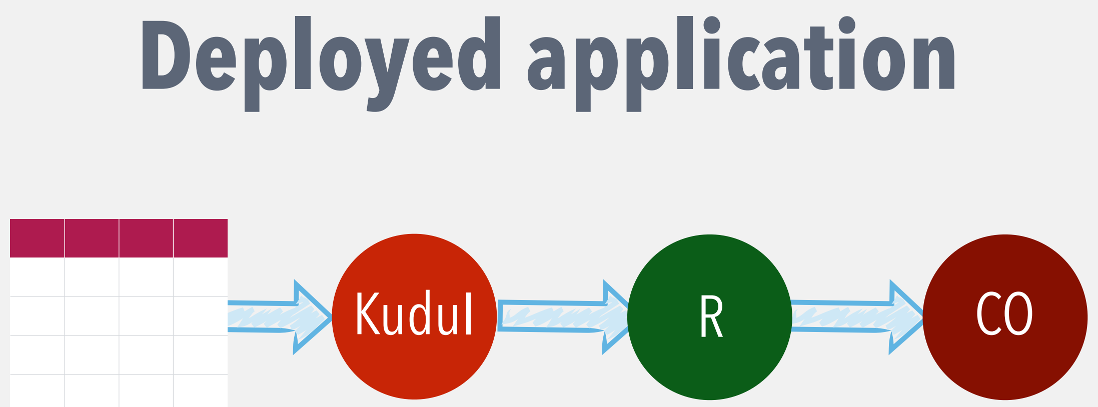
Normal Mode

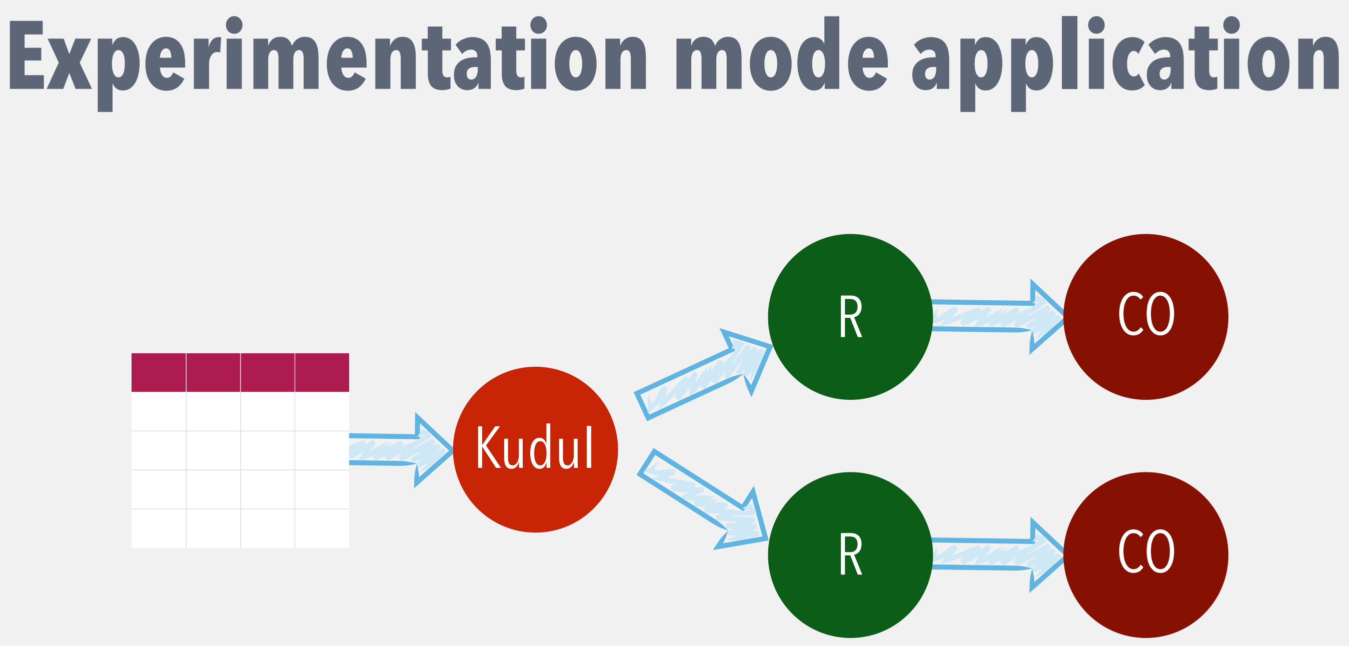
Exactly Once - Upstream window processing view



Apex Command line client


I want to fine tune the champion challenger scoring model at runtime as an experiment



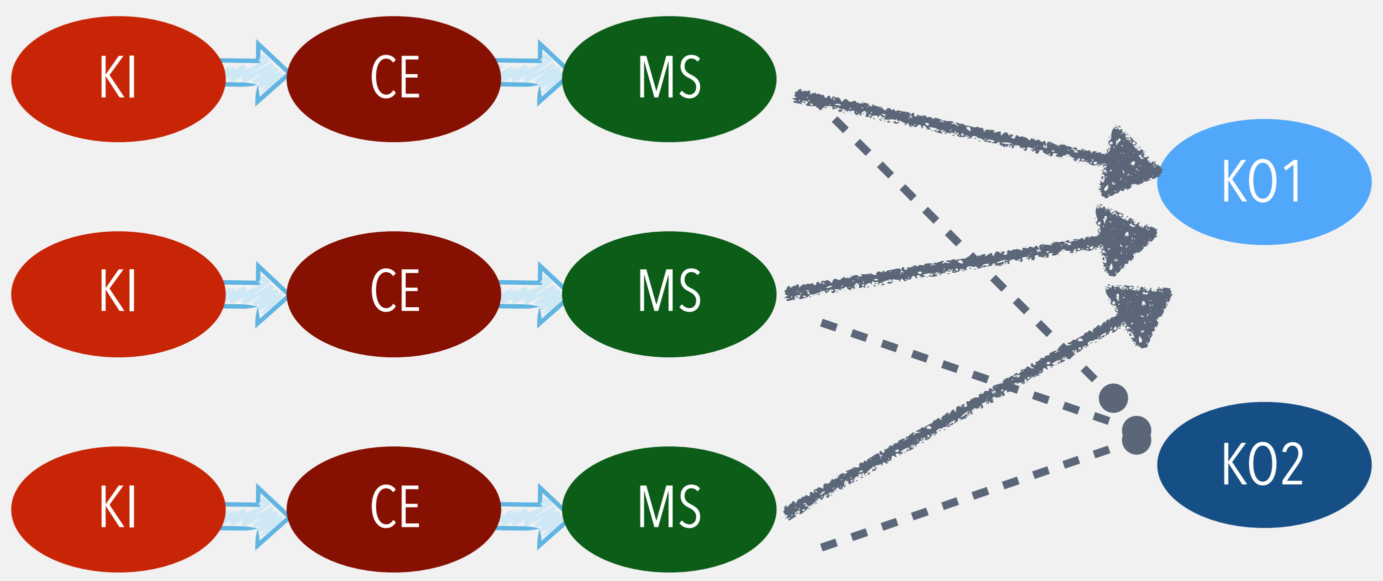


- Apex Command line client provides capabilities for
  - Launching an apex application on the cluster
  - Specifying configuration files and properties
  - Managing lifecycle of an application - Kill, shutdown
- Change the logical plan of the running application
  - We can add a new R operator with different configurations as a champion challenger
- Control operator properties at runtime
  - Ex: Change the throttle config in the Kudu Input operator
- No downtime !!

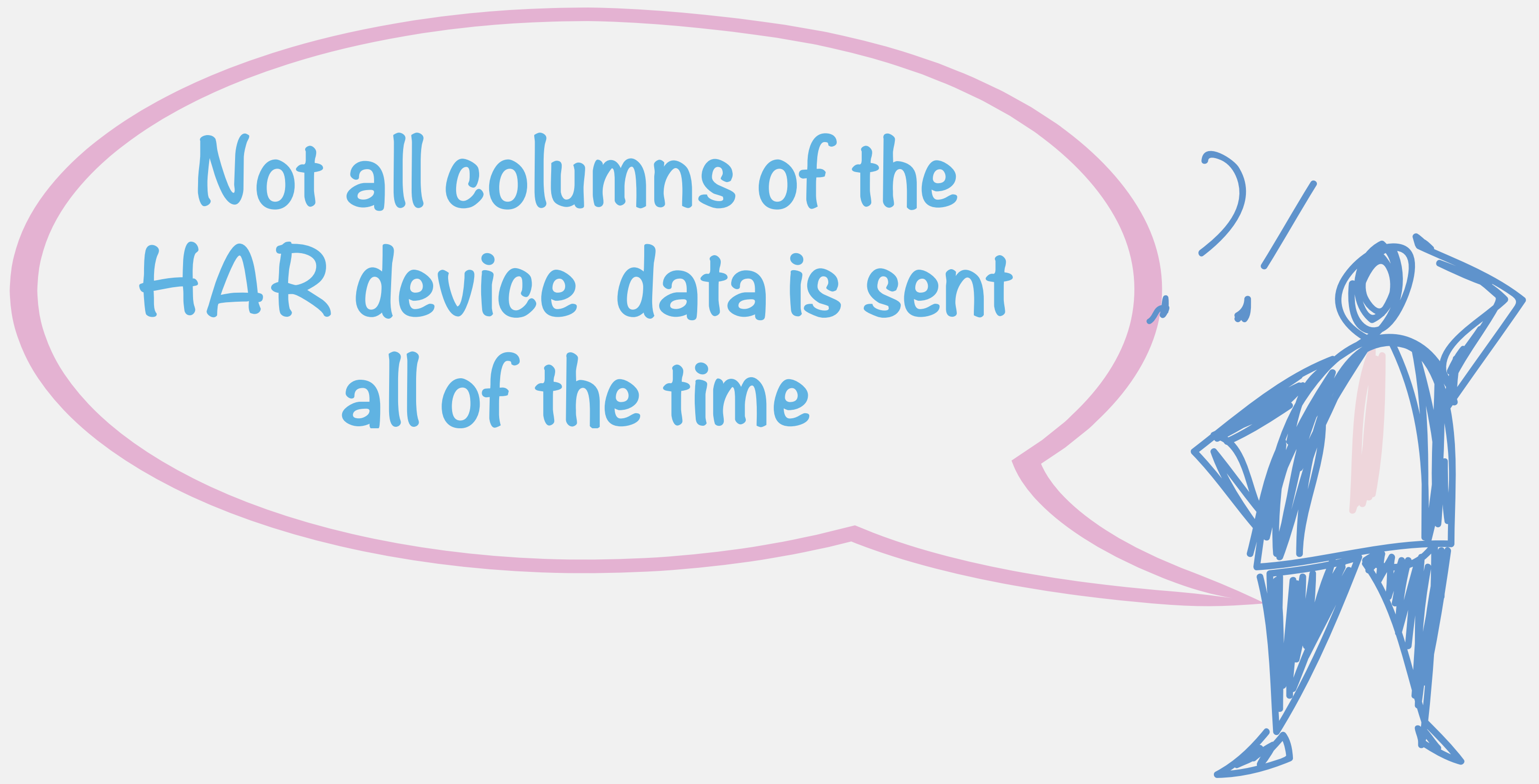
Apex Kudu Integration

- Kudu Input Operator
  - Scans a single table using a SQL expression using a distributed scan approach
  - ANTLR4 parser compensates for the missing JDBC driver for Kudu.
- Kudu Output Operator
  - Used to mutate a single table basing on the context. Supports
    - Insert
    - Update
    - Upsert
    - Delete
- Available post 3.8.0 release of Malhar

Kudu Output Operator



- Same POJO mapping to multiple tables
  - No extra transformation required
  - Automatic schema detection
  - Override Column name mapping if required



DeviceID	First Seen	LastSeen	LastKnownGeo

- Can choose to write only a subset of the column
  - Ex: LastSeen can be updated without reading FirstSeen

Kudu output operator Autometrics

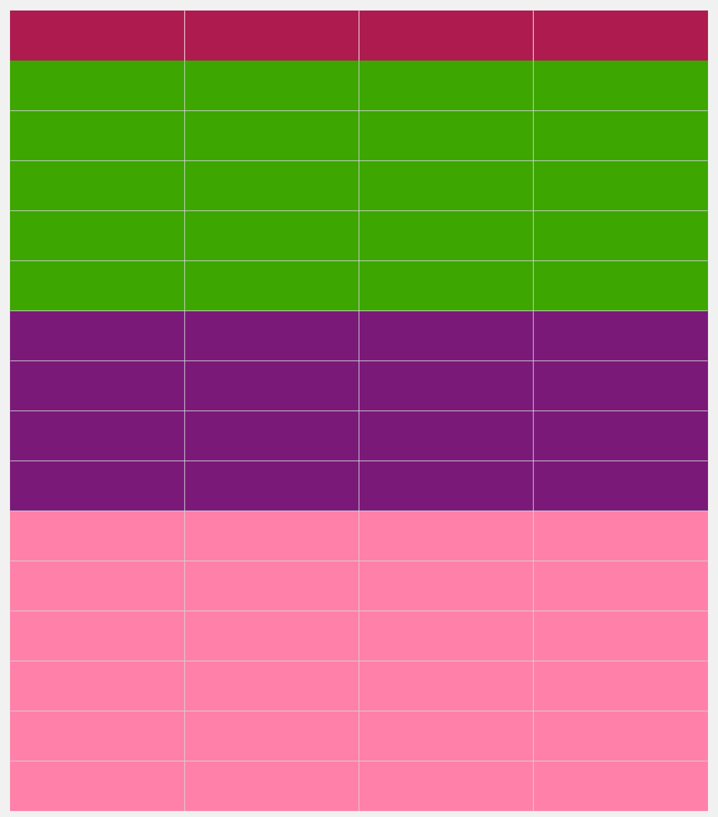
- Apex engine allows for metrics collection and monitoring
  - Termed as Autometrics
  - Metrics are automatically aggregated over the entire instances of the operator
  - Supports complex types as a metric construct
  - Metrics are also available as a REST API endpoint.

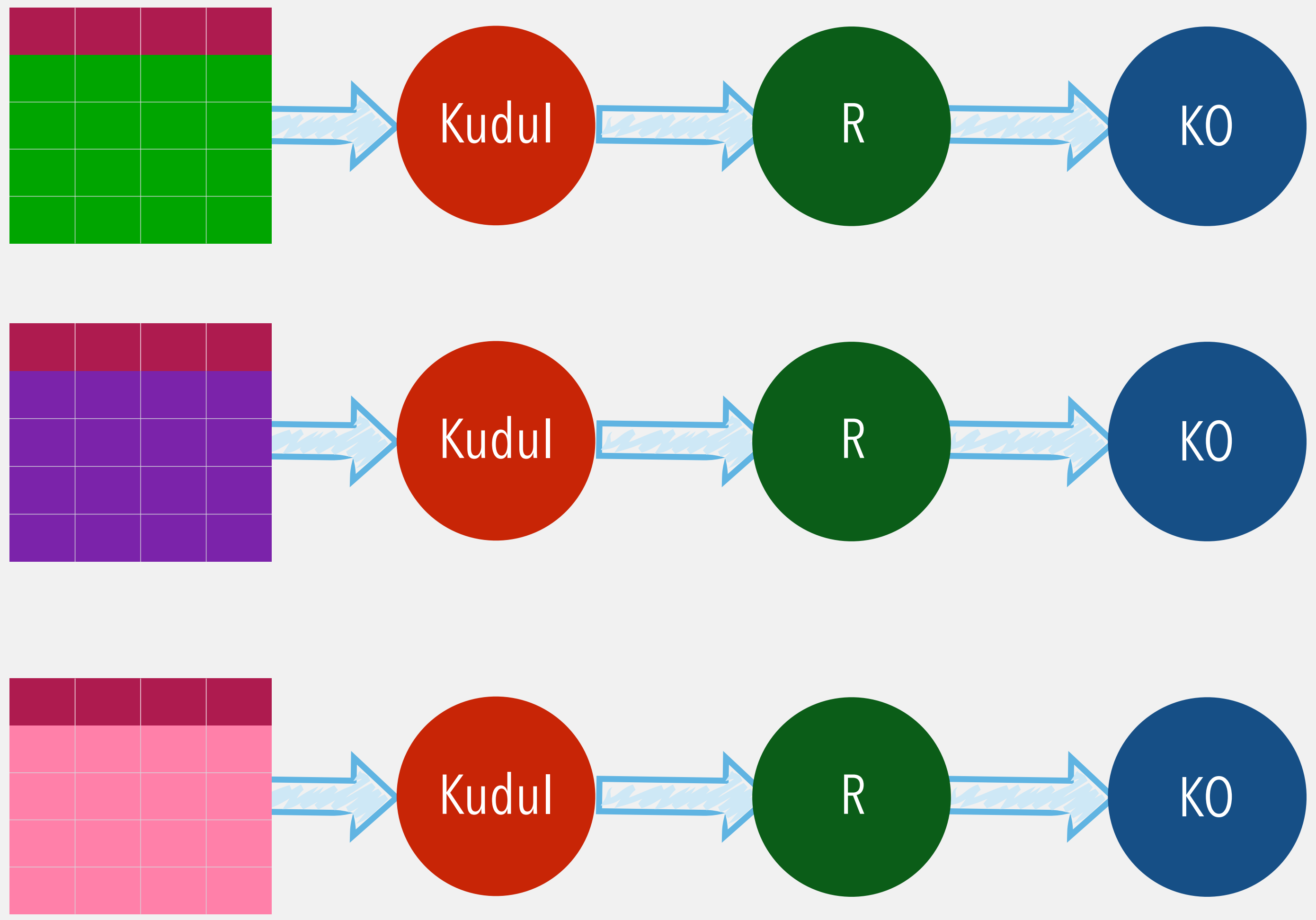
- Metrics supported by the Kudu output operator
  - On a per window basis
    - Inserts,updates,upserts,deletes, bytes written, write operations, write RPCs, RPC errors, Operational errors
  - On a global basis (i.e. from start of application)
    - Same as above

Kudu Input Operator

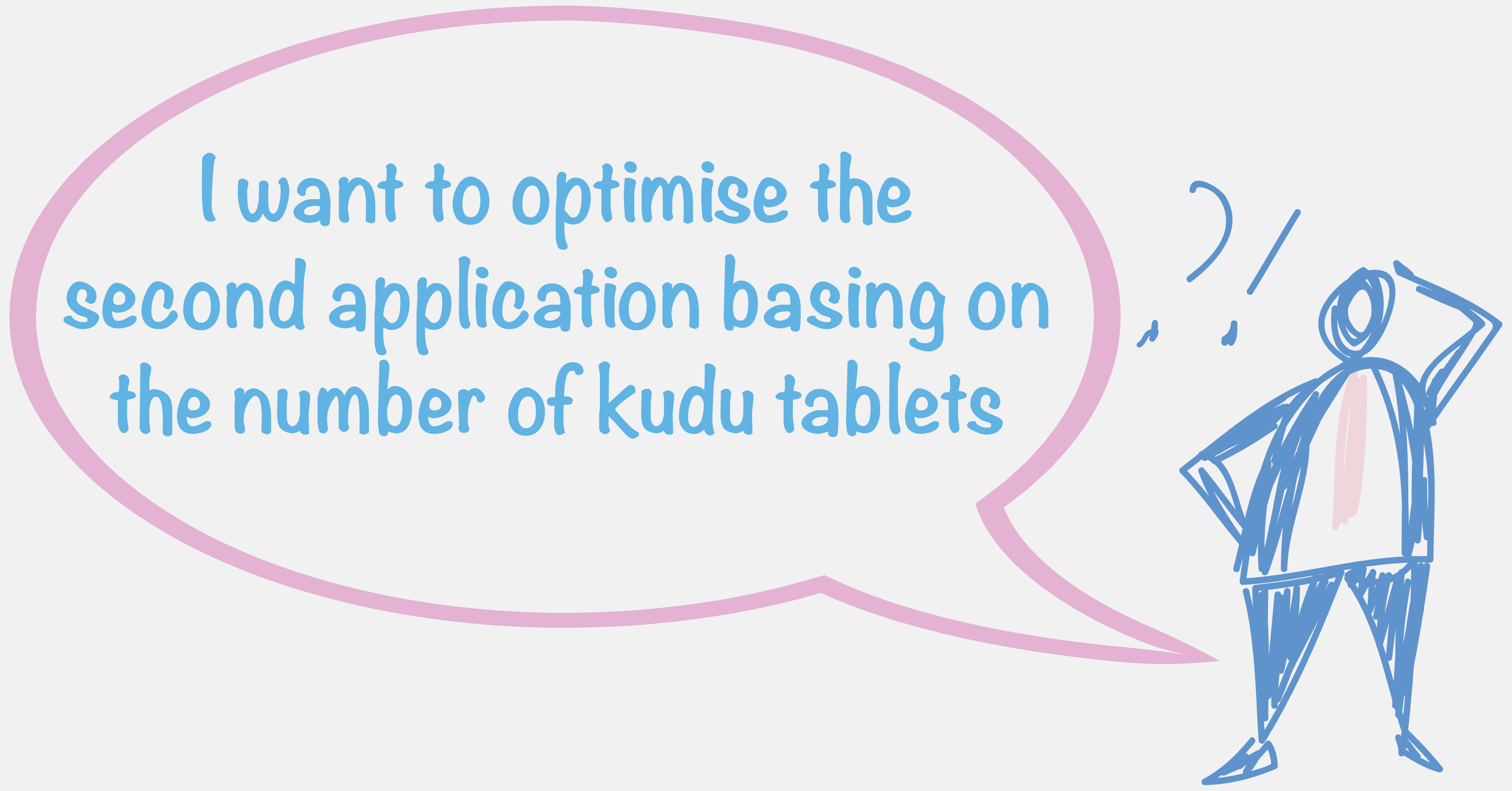
- Scans a single kudu table
- Streams one row as POJO tuple to downstream Operators
- Accepts a SQL expression to determine the rows that need to be read
- The query processing is distributed across
  - All Apex Operators that divide the stream work equally
- Disruptor Queue for maximum throughput

Query Plan



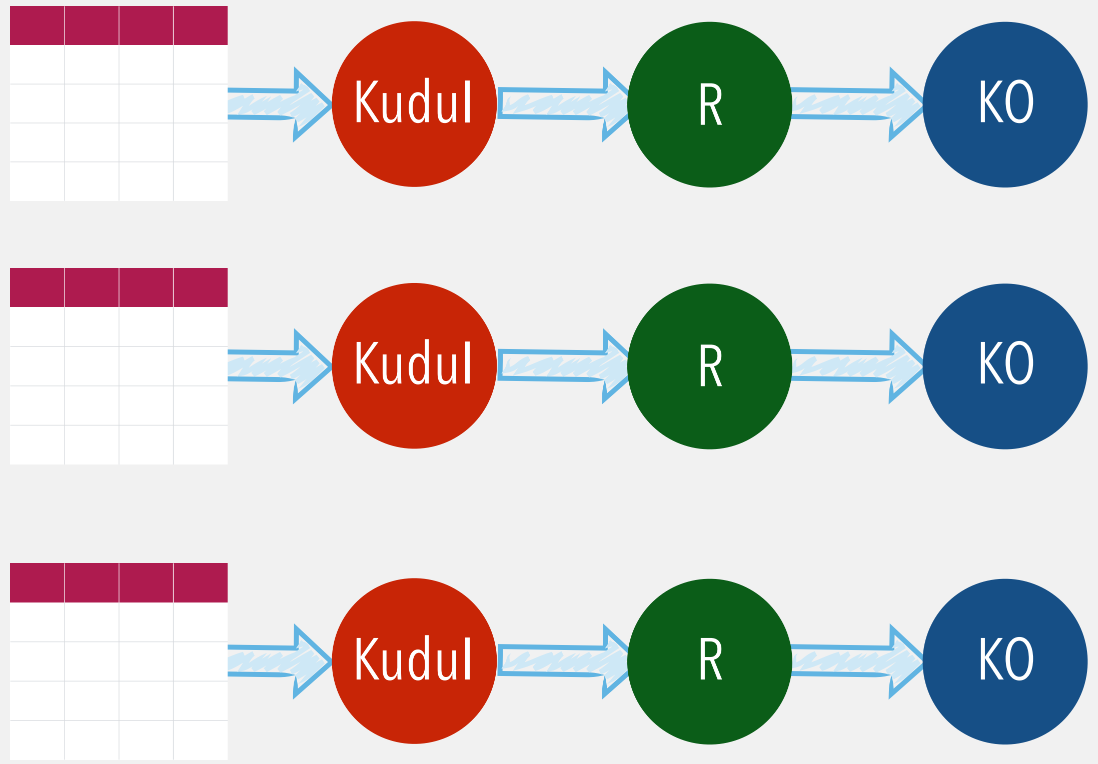


Kudu Input Operator Partitioning options

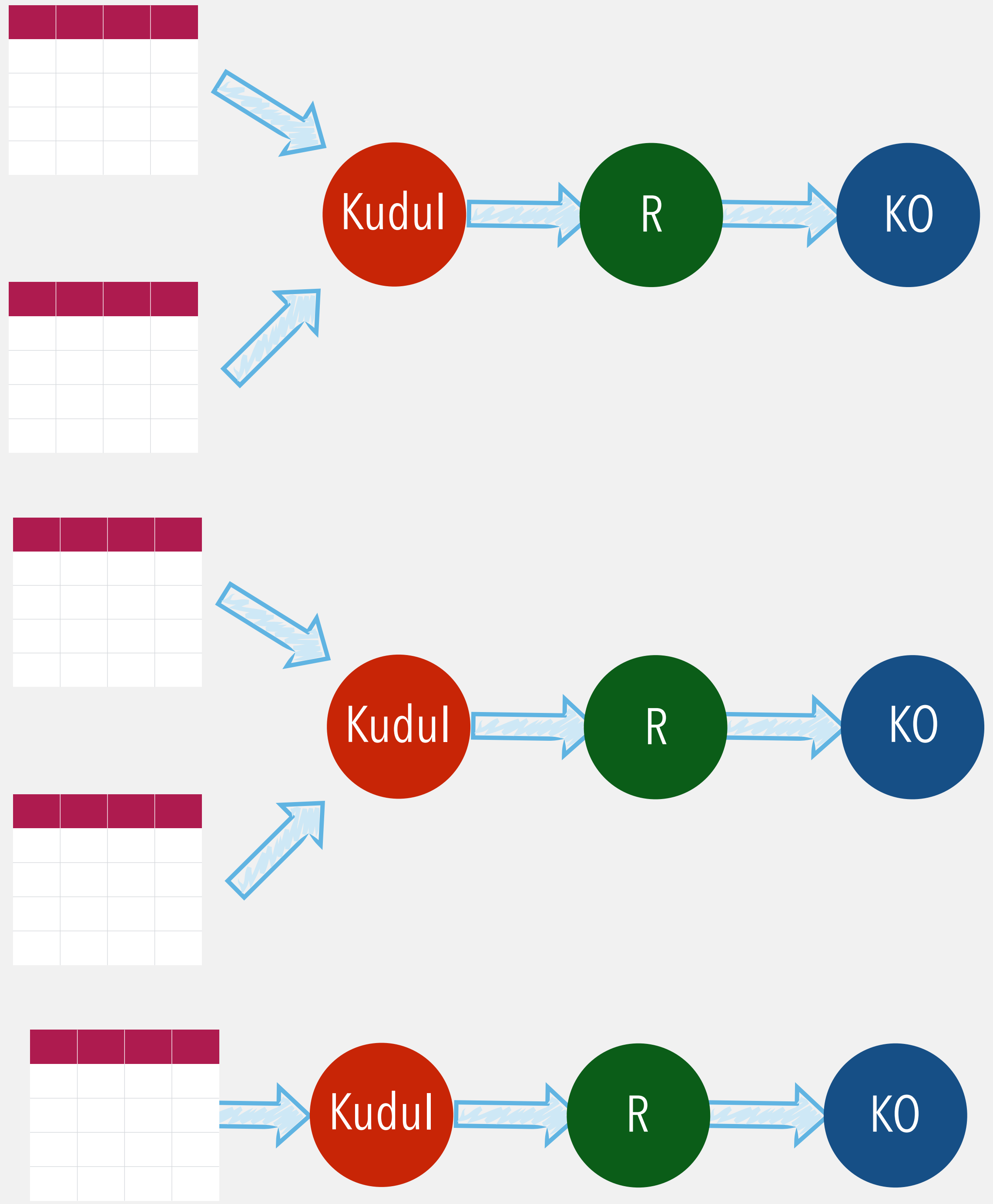

I want to optimise the second application basing on the number of kudu tablets

Operator config allows for flexible Kudu tablet to Apex operator mapping

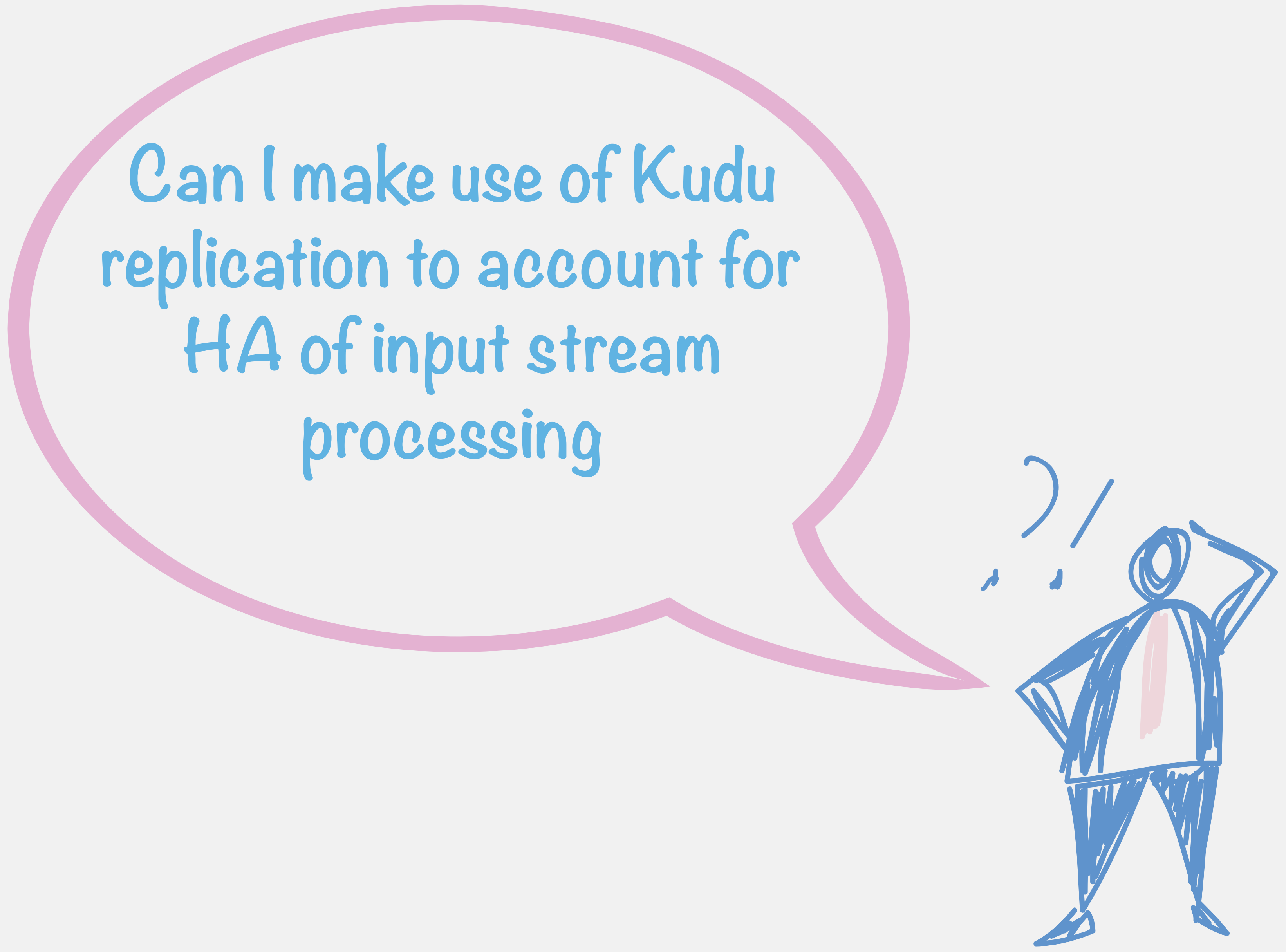
One to One mapping

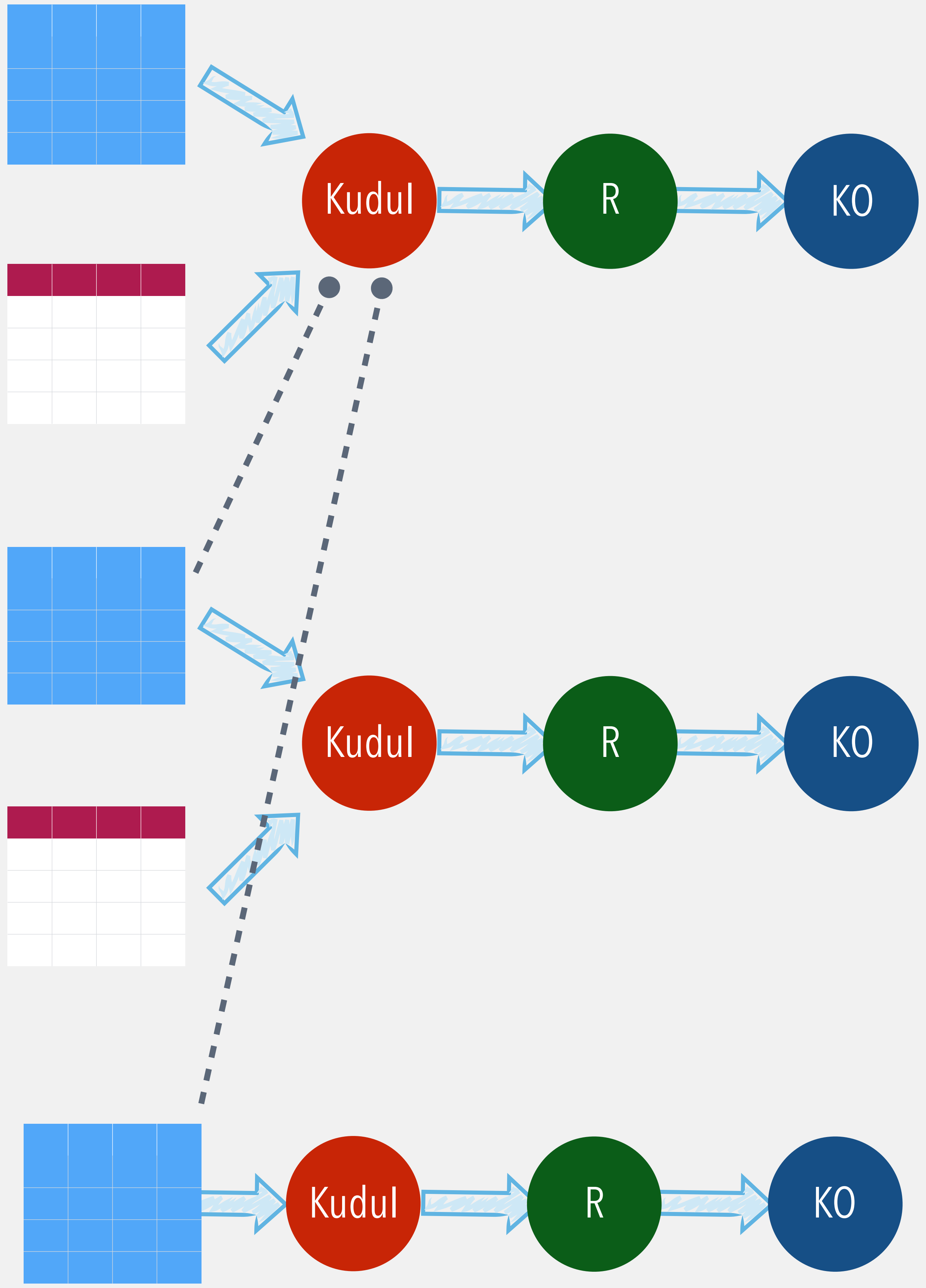


Many to One mapping



Kudu Input Operator Fault tolerance

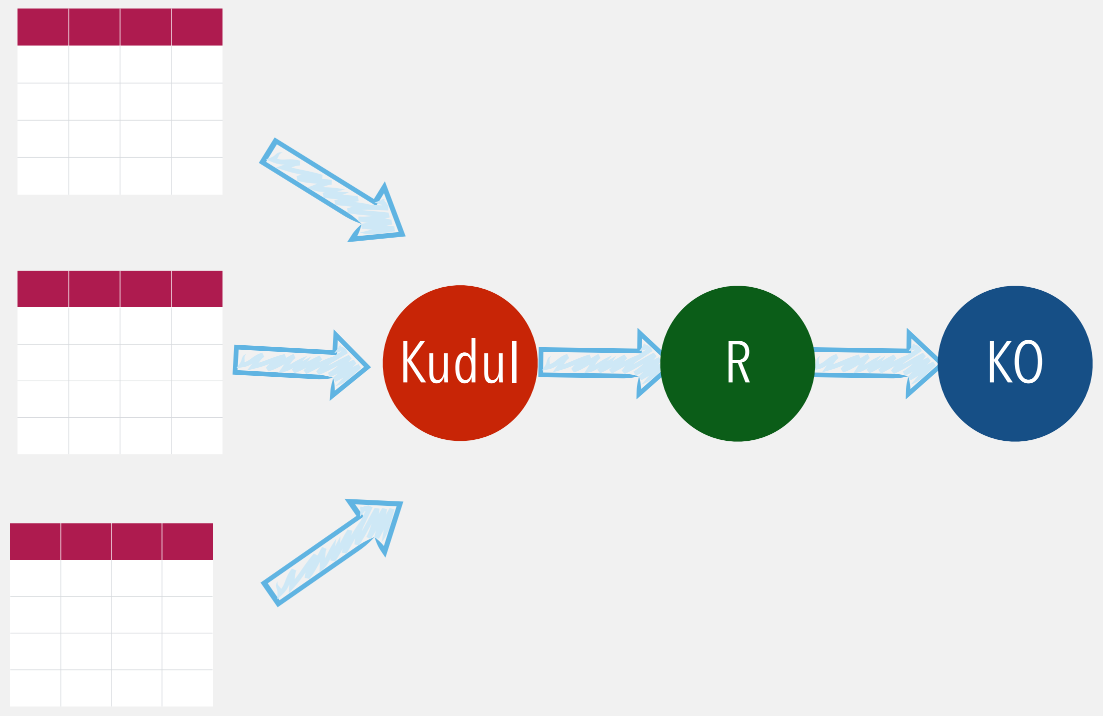

Can I make use of Kudu replication to account for HA of input stream processing



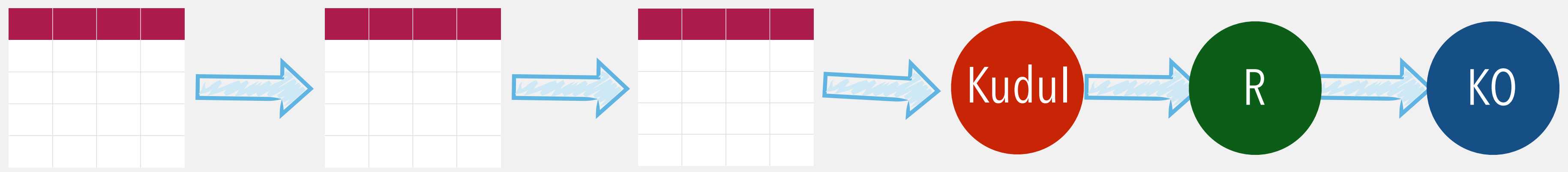
Kudu Input Operator Scan ordering

Can I tune for throughput or exactly once semantics basing on my requirements

Random order scanning



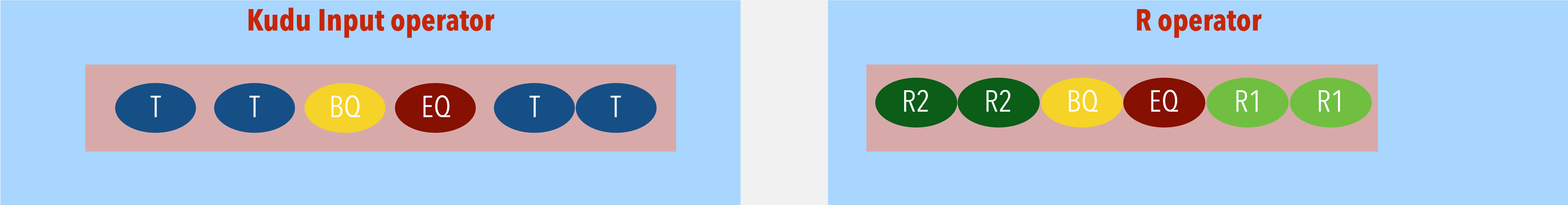
Consistent order scanning

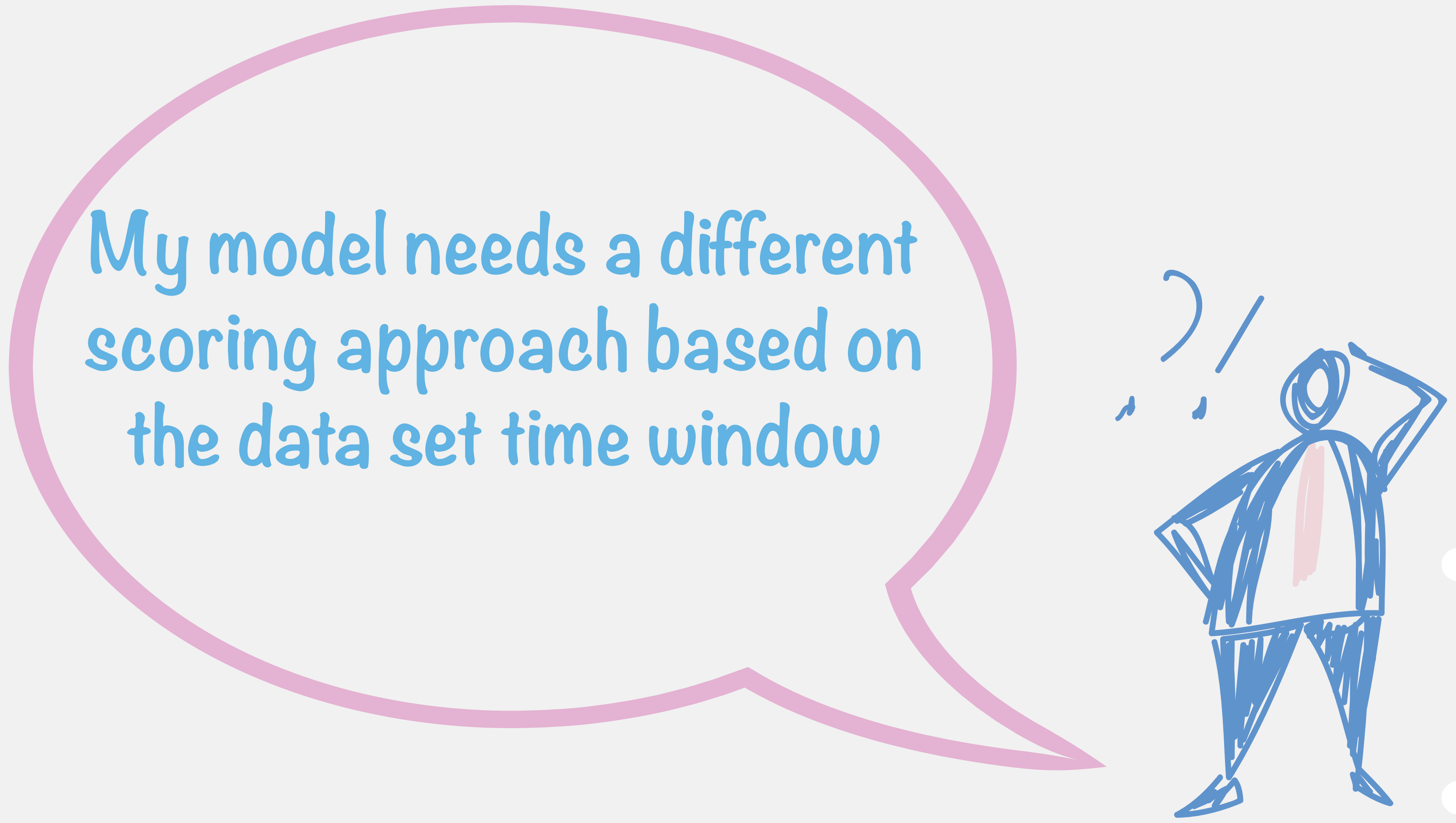


- Simple configuration switch to choose between random order & consistent order
- Consistent ordering
  - Automatically sets Fault tolerance to true
  - Exactly once processing only possible in Consistent ordering mode
  - Results in lower throughput

Kudu Input Operator Control tuples

Control tuple flow




My model needs a different scoring approach based on the data set time window

- Apex allows for control tuples (user defined watermarks) to be intermixed the data tuples flowing in the DAG
- Kudu Input operator currently allows for
  - Begin Query control tuple
  - End query control tuple
- Control tuples are custom definable
  - Ex: New query expression in a begin query control tuple
  - Ex: Window time value at the end of the query processing
- Control tuples can be sent either sent at window boundaries or inline
  - It is inline for Kudu Input operator

Kudu input operator extensibility Time travel operator

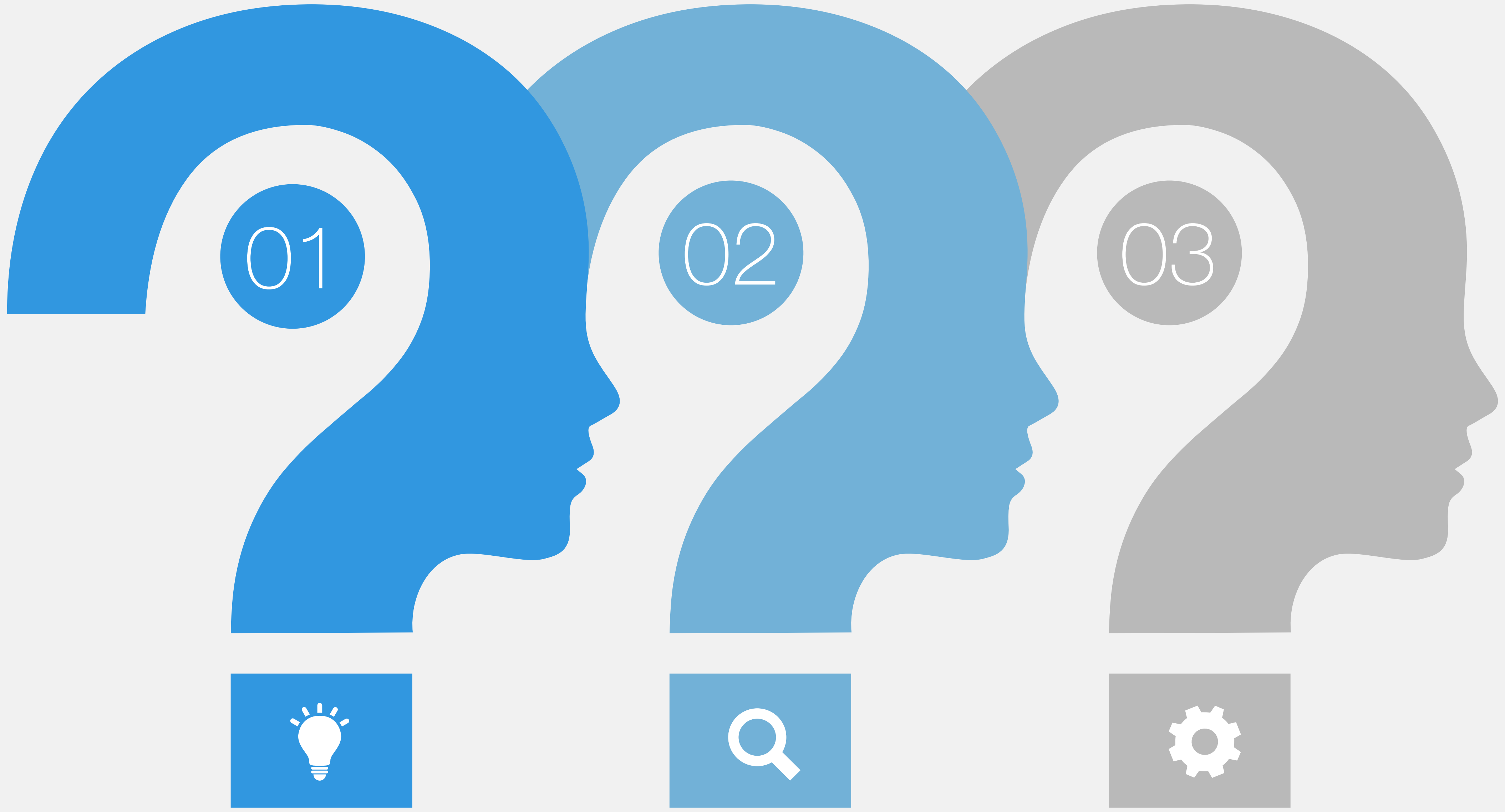
I want to run a nightly model basing on the state of data at hourly boundaries during the daytime

- As part of SQL expression allows for setting
  - Control Tuple END query message
  - Kudu READ_SNAPSHOT_TIME
- Time Travel operator
  - Each input query can scan the entire table (with appropriate filters) for data present at specified READ_SNAPSHOT_TIME time
  - "SELECT * FROM TABLE where col1 = 234 **using options READ_SNAPSHOT_TIME** = <3 A.M>"

Production References

- GE prefix platform processes IOT streaming data for analytics at sub-millisecond time frames
- Capitol One
  - 99.999 % uptime 24x7
  - Single digit millisecond end to end latencies
- Threatmetrix data pipelines for visualising fraud patterns were processed at single digit millisecond processing latencies
  - These times exclude the latencies to write to a Cassandra cluster
- A leading global financial institution (non-AUS)
  - Demonstrate AML compliance
  - Integrate with Teradata, Vertica and Hadoop

Q & A



- Apex Community http://apex.apache.org/community.html
- Docs http://apex.apache.org/docs.html
- Powered by Apache Apex http://apex.apache.org/powered-by-apex.html
- REST-API Server https://github.com/atrato/atrato-server
- Twitter handle https://twitter.com/apacheapex
- Examples https://github.com/apache/apex-malhar/tree/master/examples

https://www.linkedin.com/in/ananth-kalyan-chakravarthy-ph-d-7a46156/

@_ananth_g