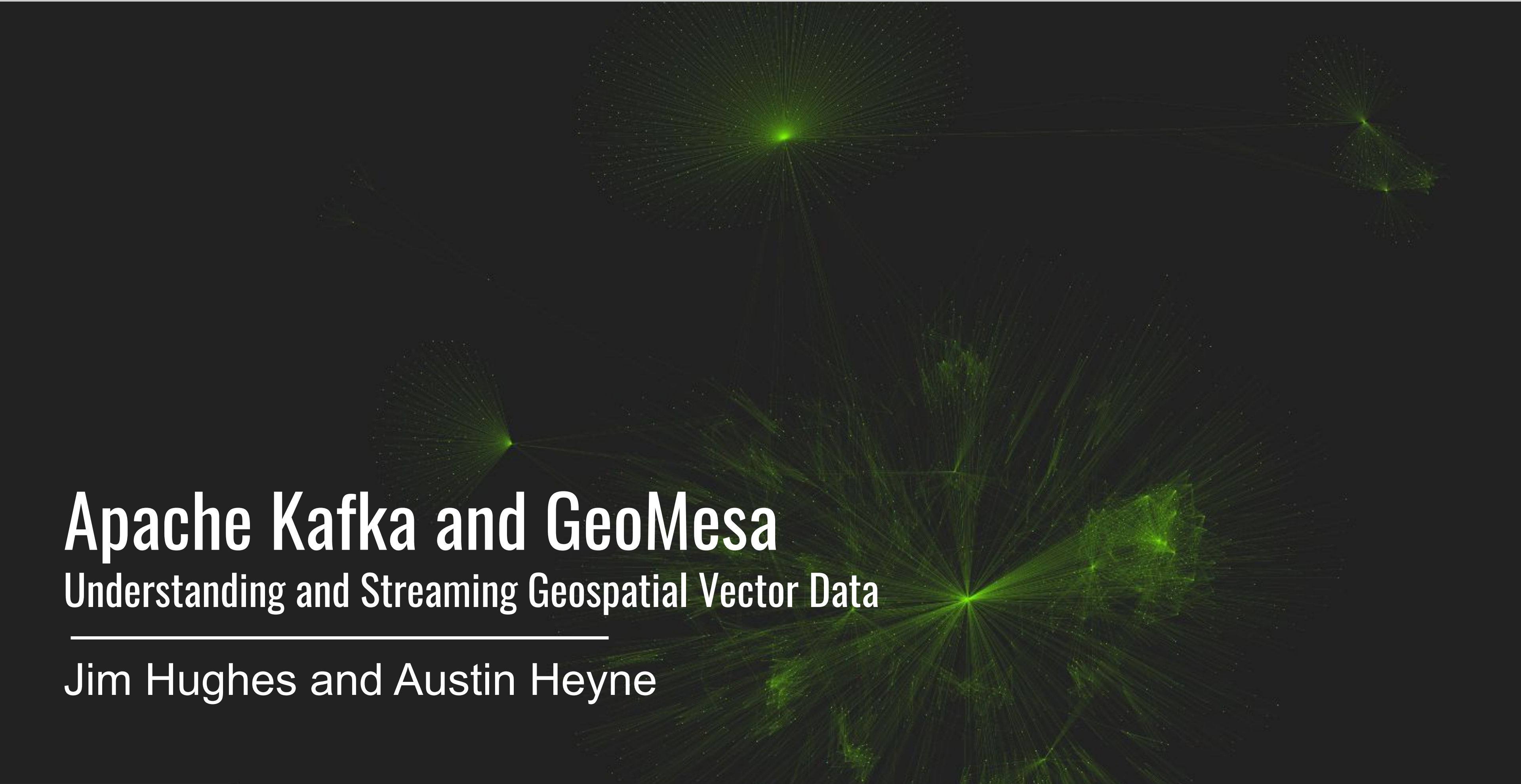


Apache Kafka and GeoMesa

Understanding and Streaming Geospatial Vector Data

Jim Hughes and Austin Heyne

Who we are:

Jim Hughes

- Software Engineer at Confluent
- GeoMesa committer
- SFCurve project lead
- JTS committer
- Contributor to GeoTools and GeoServer

Austin Heyne

- Software Engineer at GA-CCRi
- GeoMesa Lackey
- GeoSpatial Data Management SaaS Lead

Streaming Analytics in Kafka

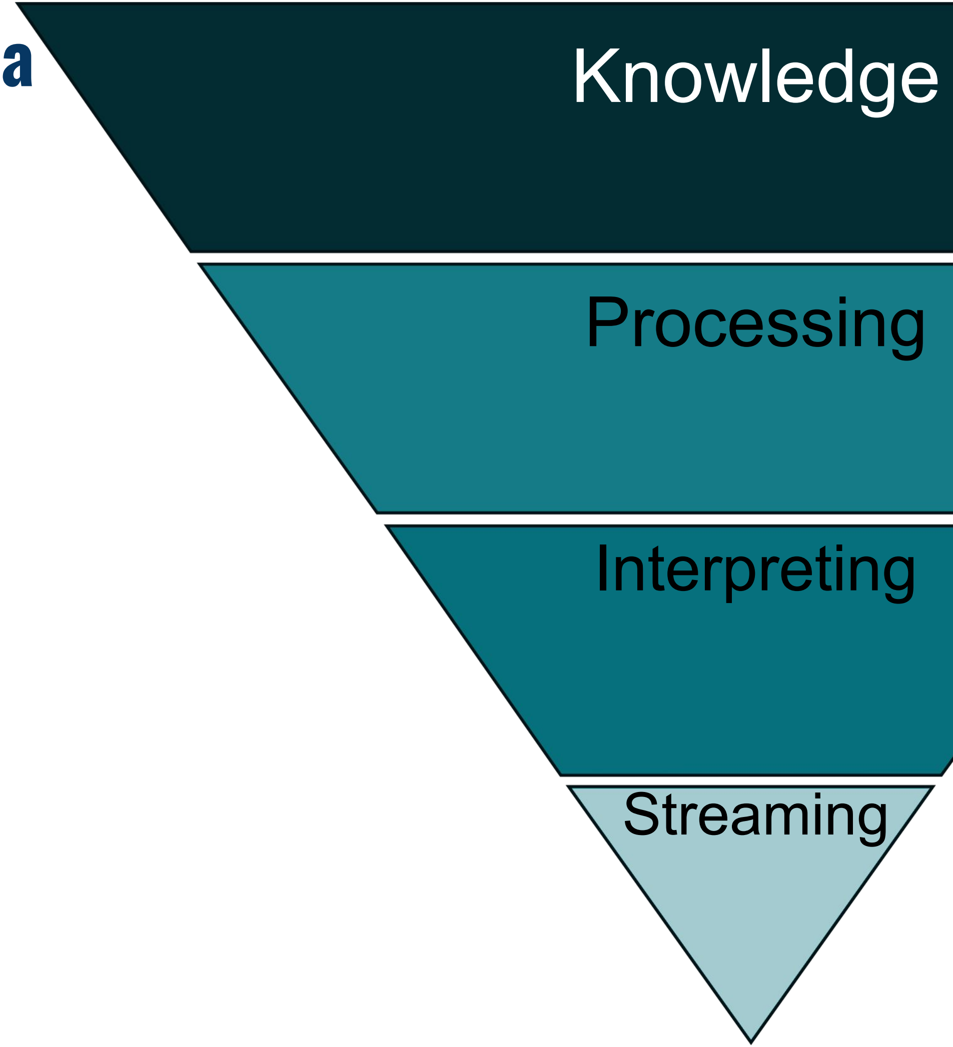
Streaming Data

Understanding Data through Types

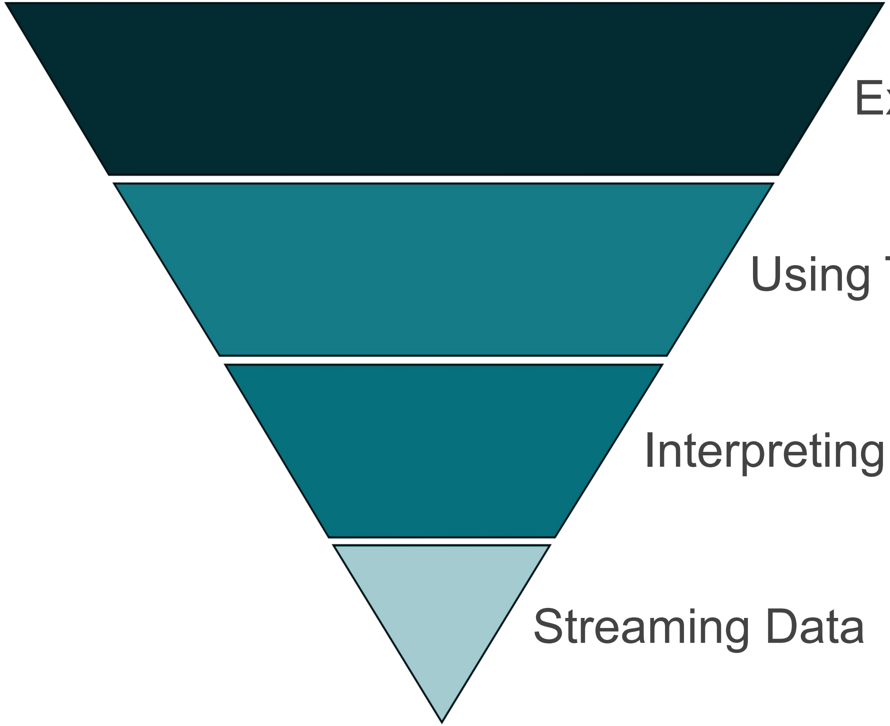
Asking Questions About Data

Producing Derivative Data

Using this scheme to understand and process GeoSpatial data.



Streaming Analytics in Kafka



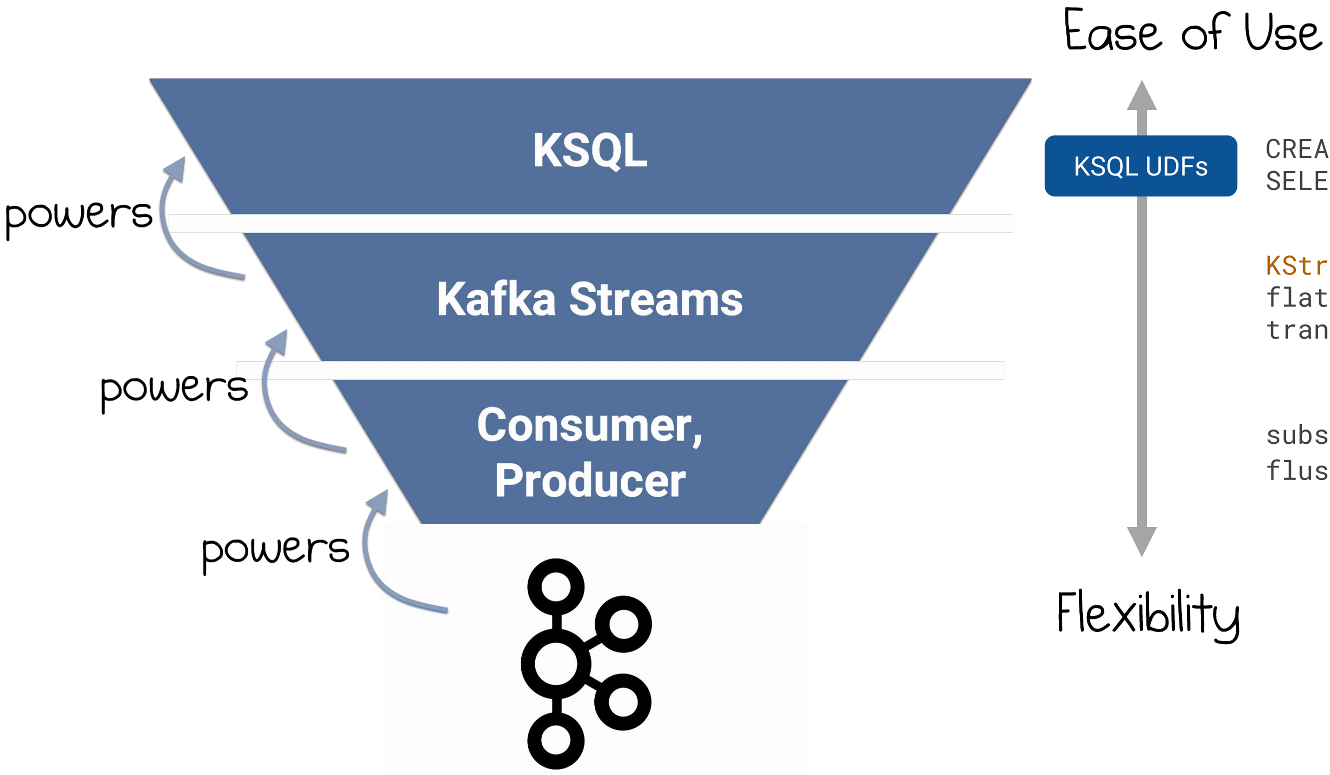
Extracting Knowledge from Processing

Using Types to Process Data

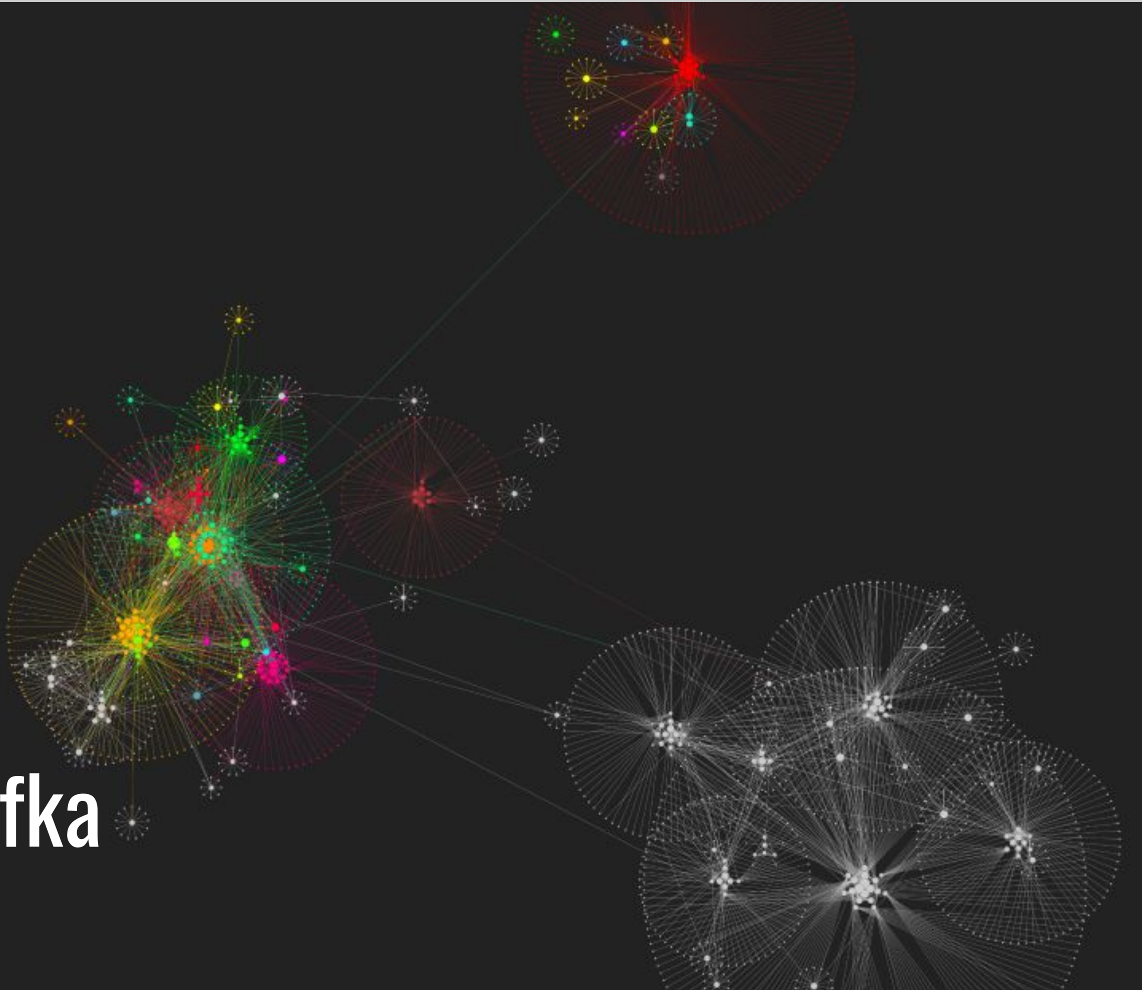
Interpreting Data with Types

Streaming Data

Using this scheme to understand and process GeoSpatial data.



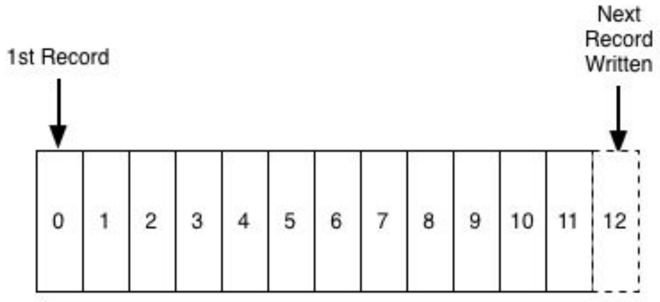
```
CREATE STREAM, CREATE TABLE,
SELECT, JOIN, GROUP BY, SUM, …

KStream, KTable, filter(), map(),
flatMap(), join(), aggregate(),
transform(), …

subscribe(), poll(), send(),
flush(), beginTransaction(), …
```



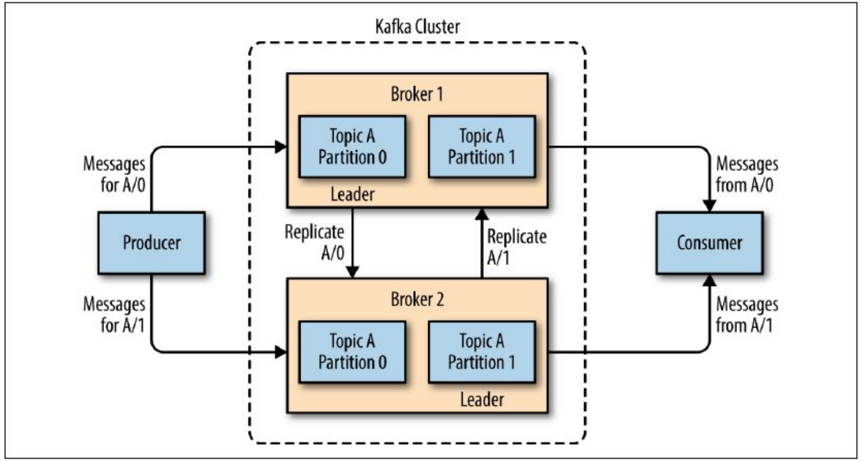
Streaming Data in Kafka

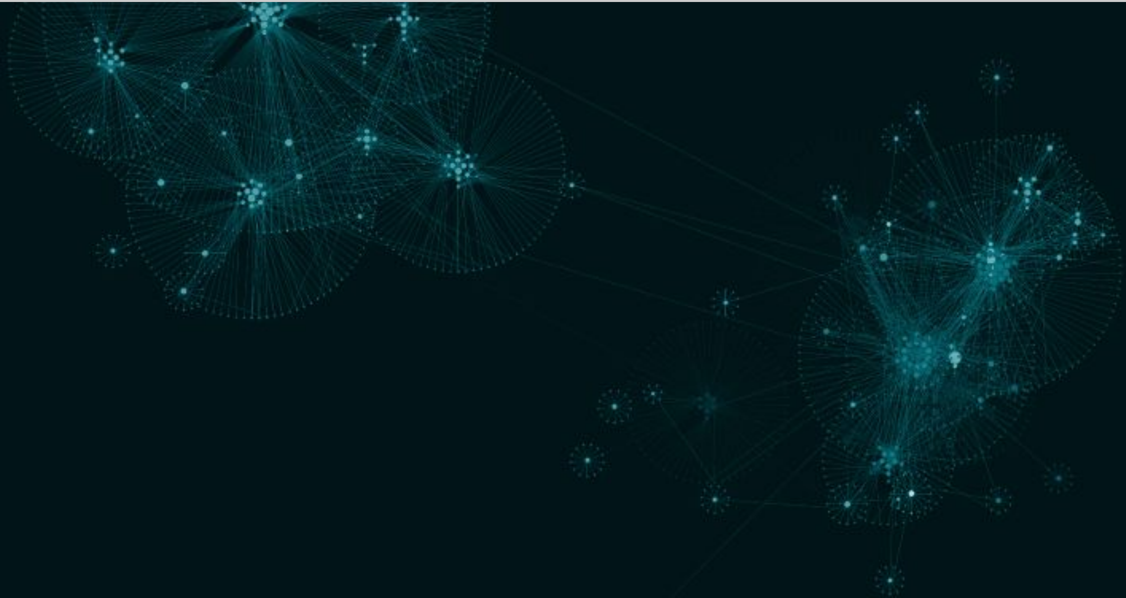
Apache Kafka ~~Overview~~ *Speedrun*

- Event streaming platform:
  - Publish (write)
  - Subscribe (read)
  - Store
  - Process



- Events are posted to Topics
- Topics are composed of partitions
- Consumers read partitions
- Consumers can work in groups
- Brokers serve topics to consumers

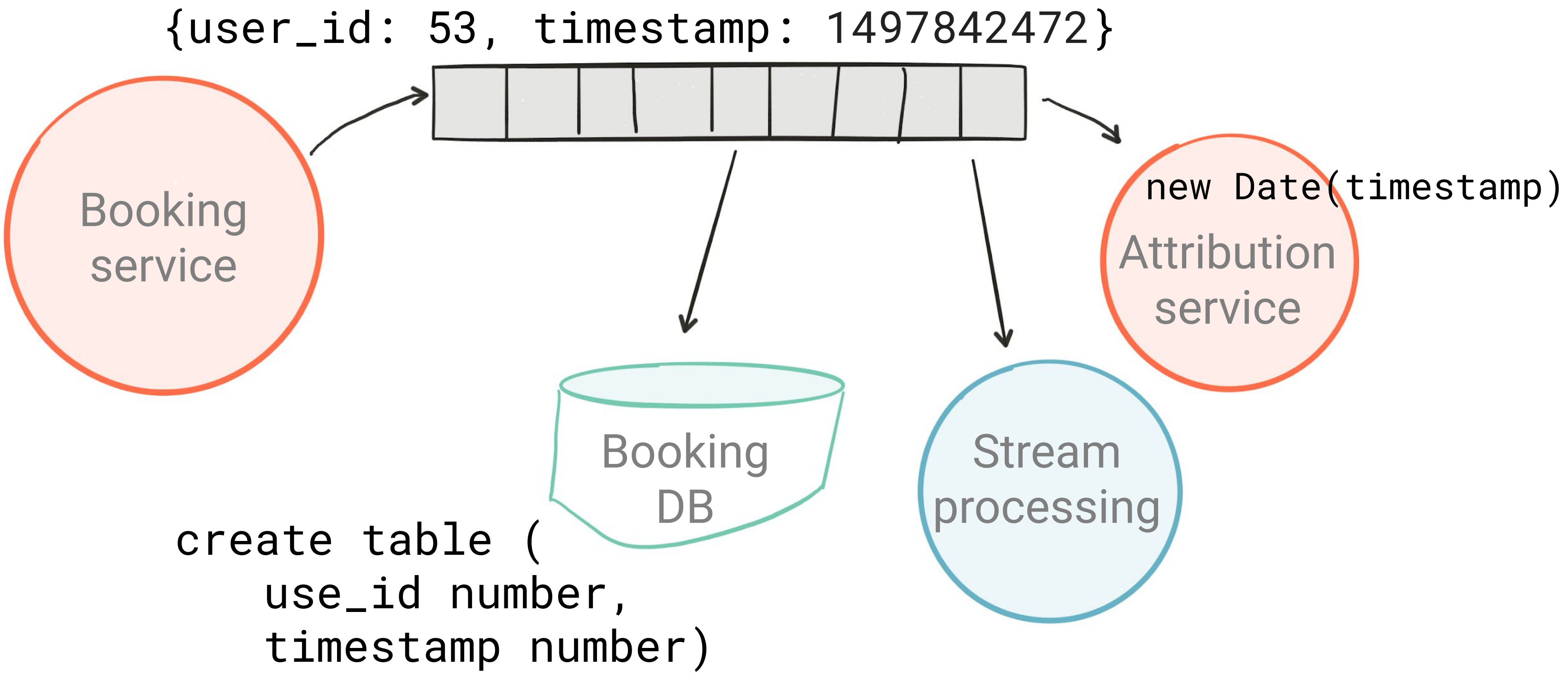




Interpreting Data in Kafka

Schemas are APIs

Schemas are about how teams work together



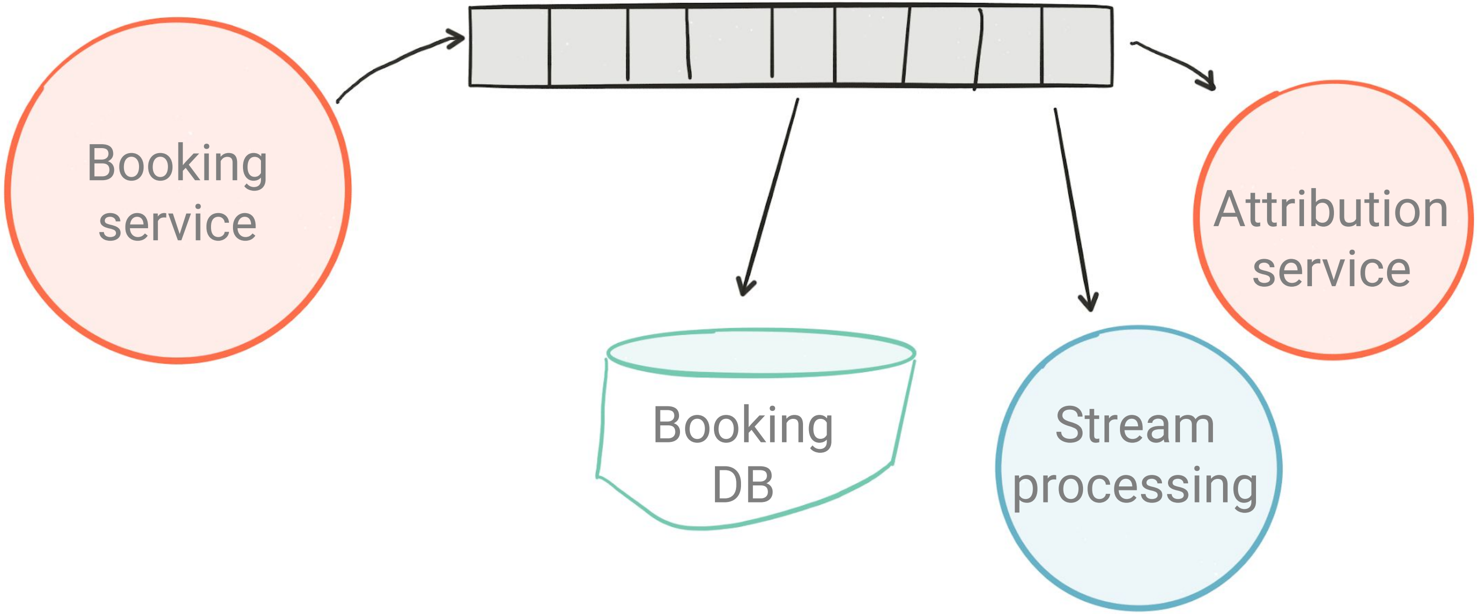
```
{user_id: 53, timestamp: 1497842472}
```

```
new Date(timestamp)
```

```
create table (
    use_id number,
    timestamp number)
```

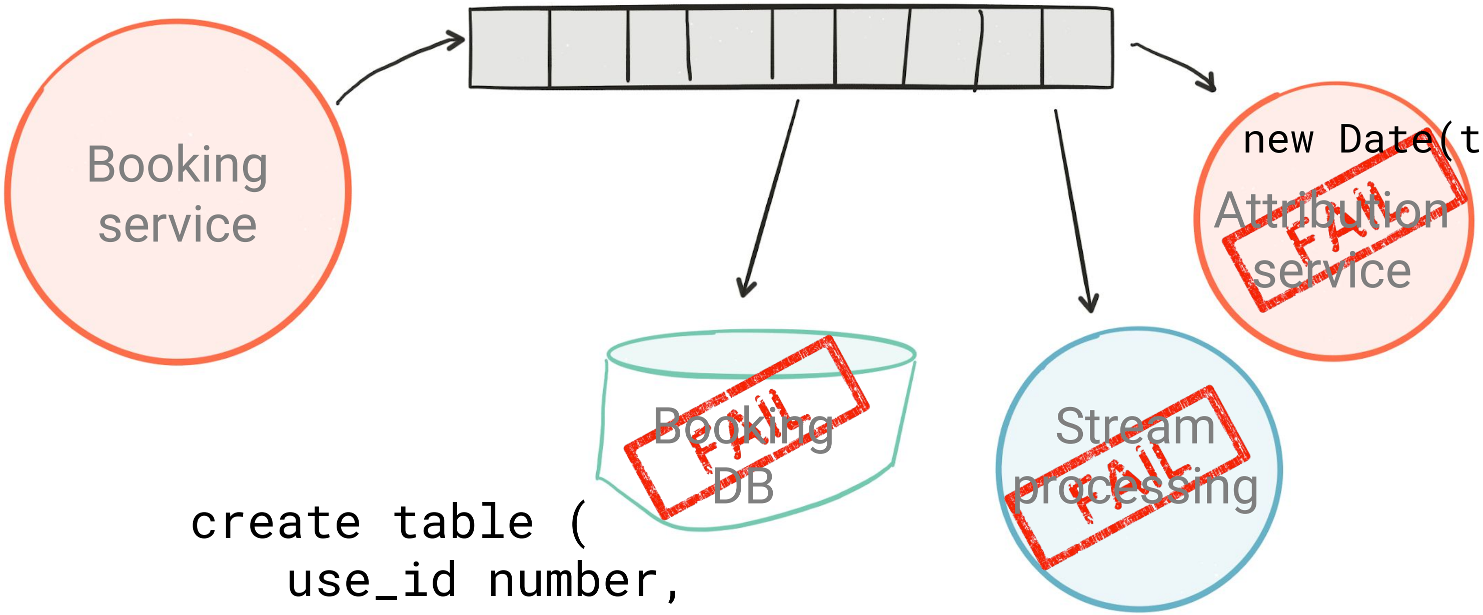
Schemas are about how teams work together

```
{user_id: 53, timestamp: “June 28, 2017 4:00pm”}
```



Schemas are about how teams work together

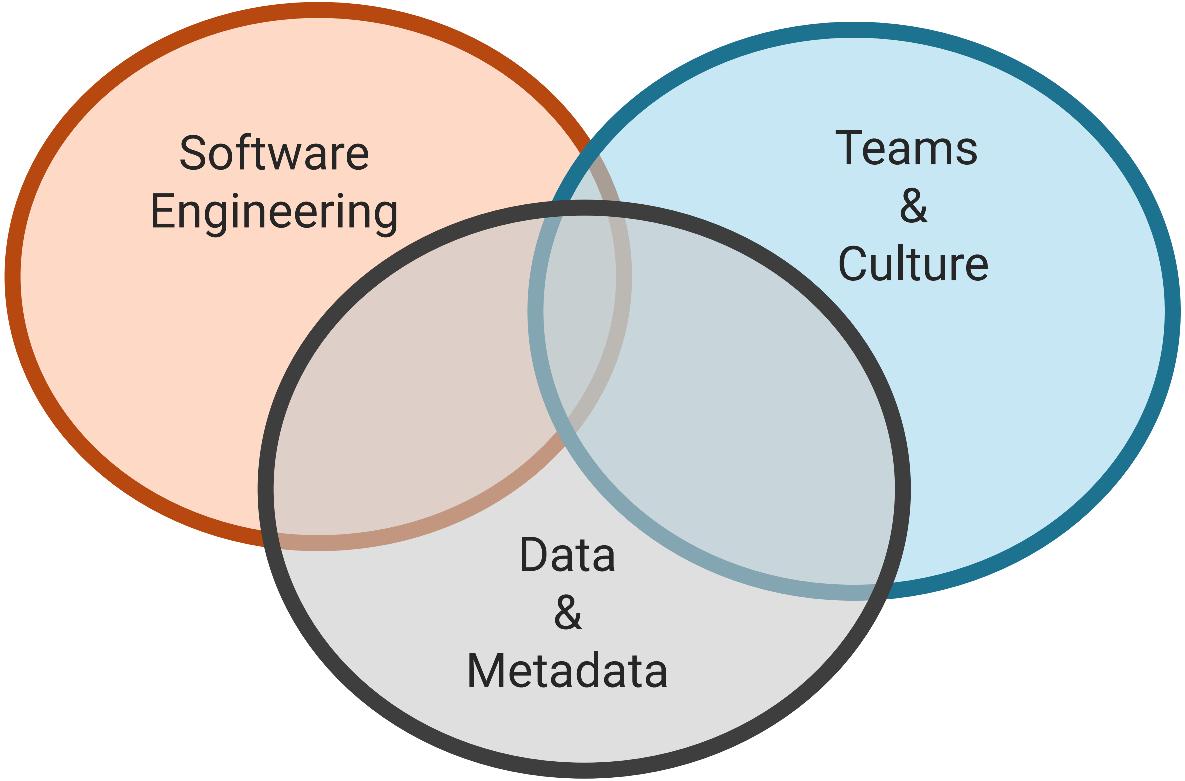
```
{user_id: 53, timestamp: "June 28, 2017 4:00pm"}
```



```
new Date(timestamp)
```

```
create table (
    use_id number,
    timestamp number)
```

It isn’t just about the services



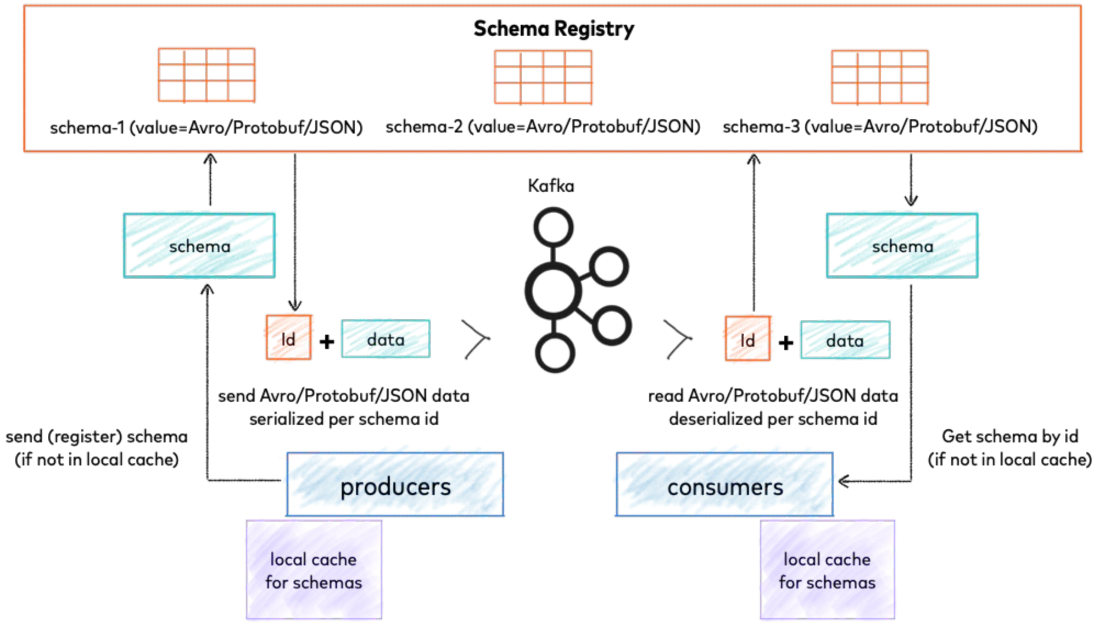
Schema are APIs

- They require specifications
- We need to make changes to them
- We need to detect breaking changes
- Versioning
- Tools

What do schema registries do?

1. Store schemas – put/get
2. Link one or more schema to each event
3. Java client that fetches and caches schemas
4. Enforcement of compatibility rules
5. Graphical browser

Confluent Schema Registry



Schema Registry
schema-1 (value=Avro/Protobuf/JSON)
schema-2 (value=Avro/Protobuf/JSON)
schema-3 (value=Avro/Protobuf/JSON)
Kafka
schema
schema
Id
+
data
Id
+
data
send Avro/Protobuf/JSON data serialized per schema id
read Avro/Protobuf/JSON data deserialized per schema id
send (register) schema (if not in local cache)
Get schema by id (if not in local cache)
producers
consumers
local cache for schemas
local cache for schemas

- Distributed REST interface for serving schemas
  - Provides schema versioning and evolution
  - Supports Avro, Protobuf, and JSON
- Provides more compaction than plain Avro
  - Schema ID is sent with each message

Processing Data with Kafka Streams

Kafka Streams Overview

- Java library for processing and analyzing data in Kafka
- Distinguishes between event time and processing time
- Windowing Support
- Real-time querying of application state
- Low barrier to entry
- Easily scalable

Kafka Streams Highlights

- Fault tolerant local state
- Exactly-once processing
- One-record-at-a-time processing
- Event time windowing (with out-of-order arrival!)
- Multiple levels of API/DSL

Kafka Streams is Not:

- A server side modification or standalone application
  - Client side library
  - Provides abstractions over base Kafka library and features

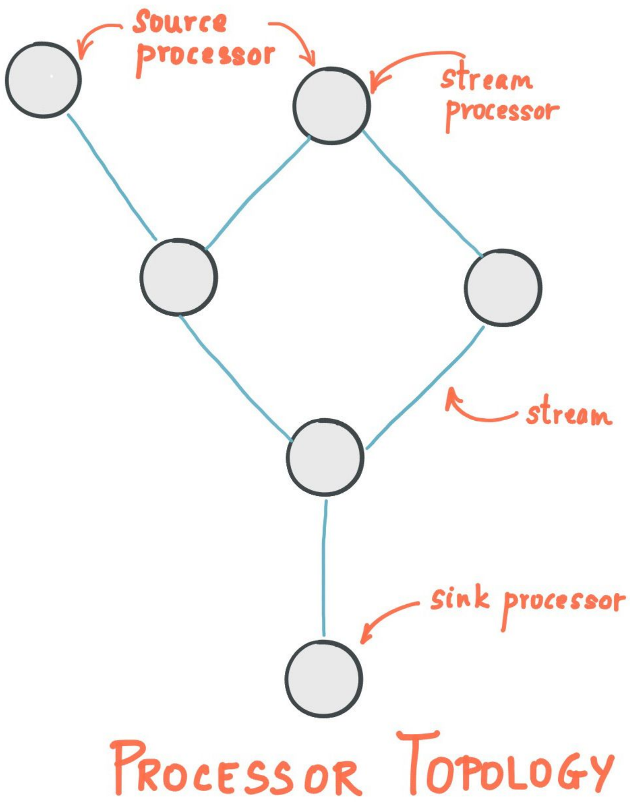
- A resource manager
  - Parallelism is achieved through normal Kafka features

Kafka Streams Topology

- Stream
  - Sequence of Immutable records
  - Unbounded
  - Continuous
  - Ordered (time)
  - Replayable
  - Fault tolerant

- Record
  - Key-value pair of byte arrays

- Event == Data Record



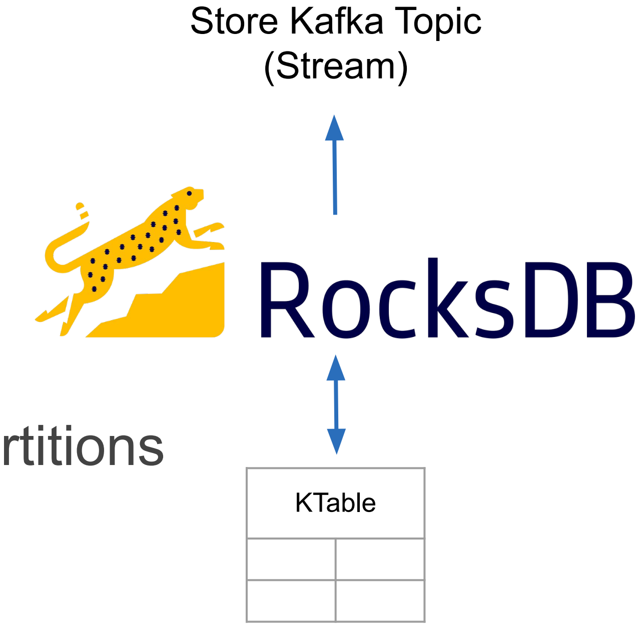
State Stores

State Stores maintain state for a specific processor locally

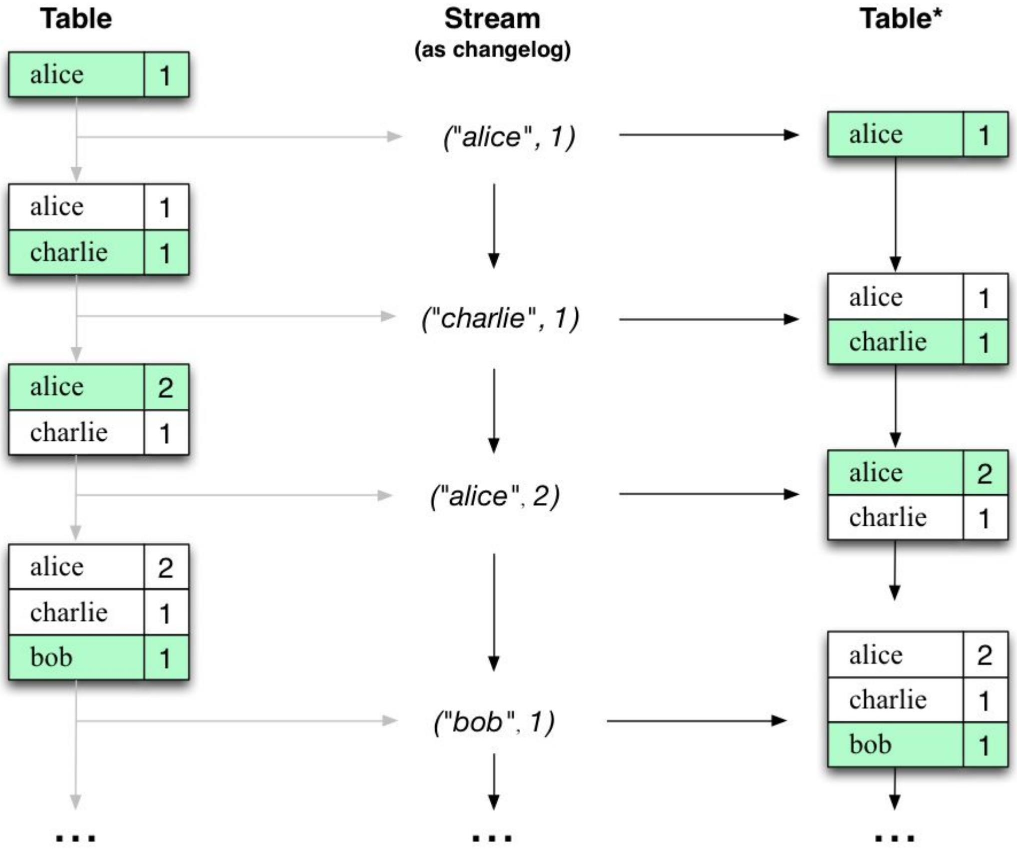
- KeyValueStore
- WindowStore
- SessionStore
- …

Changelogs are partitioned

- Local store only has own partitions
- # changelog partitions == # input topic partitions



Stream-Table Duality



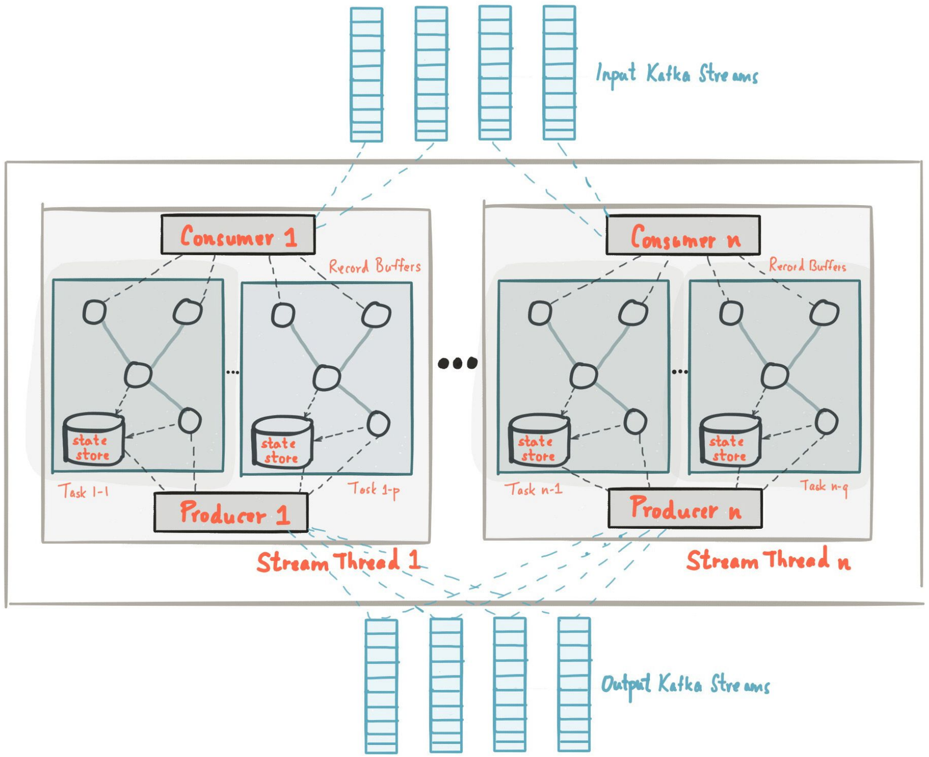
Stream as Table

- Changelog of table
- Changelog replayed to construct
- Aggregations produce tables
  - Pageviews by user

Table as Stream

- Table is a snapshot
- Latest value for each key

Kafka Streams Architecture



Parallelism

Kafka

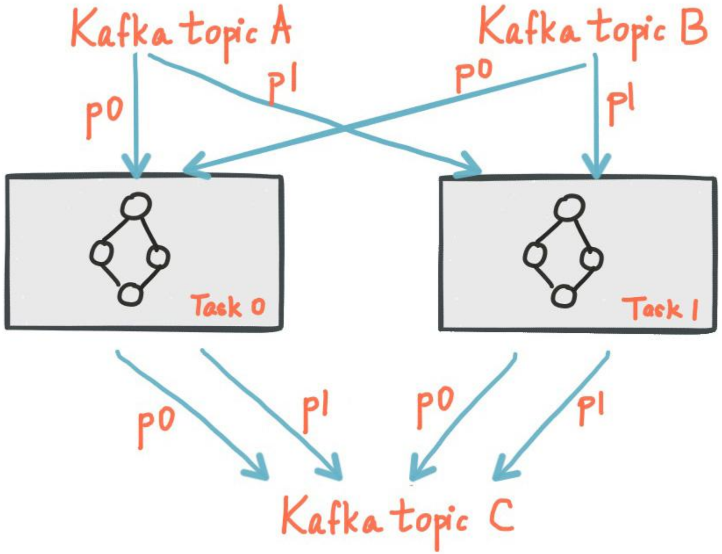
- partitions for storage and transport parallelism

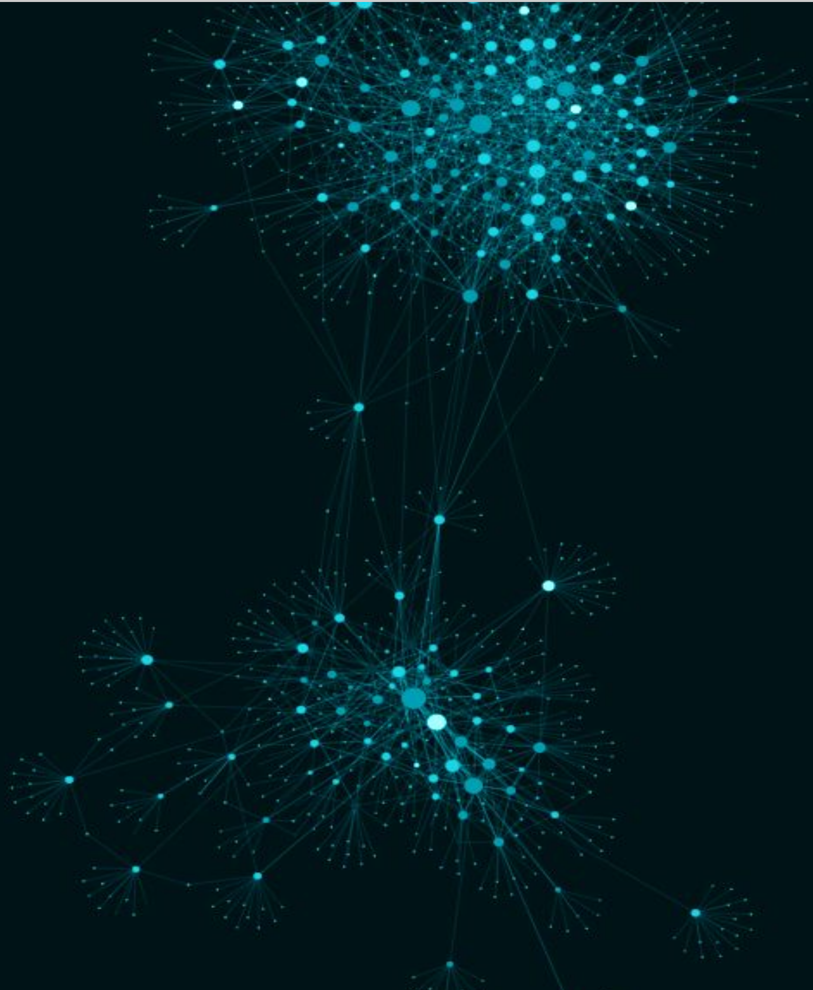
Kafka Streams

- partitions for compute and processing parallelism

Maximum parallelism == # partitions

Keys determine partition and thus processing order





What is GeoMesa?

What is GeoMesa?

A suite of tools for streaming, persisting, managing, and analyzing spatio-temporal data at scale

LocationTech

geomesa

What is GeoMesa?

A suite of tools for **streaming**, persisting, managing, and analyzing spatio-temporal data at scale

kafka

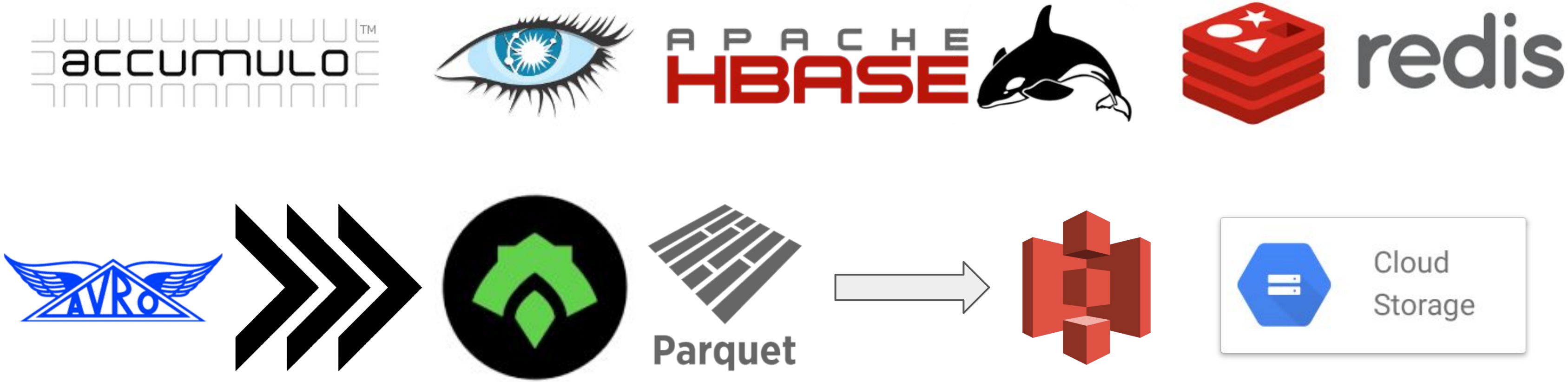
STORM

KSQL

Siddhi

What is GeoMesa?

A suite of tools for streaming, **persisting**, managing, and analyzing spatio-temporal data at scale

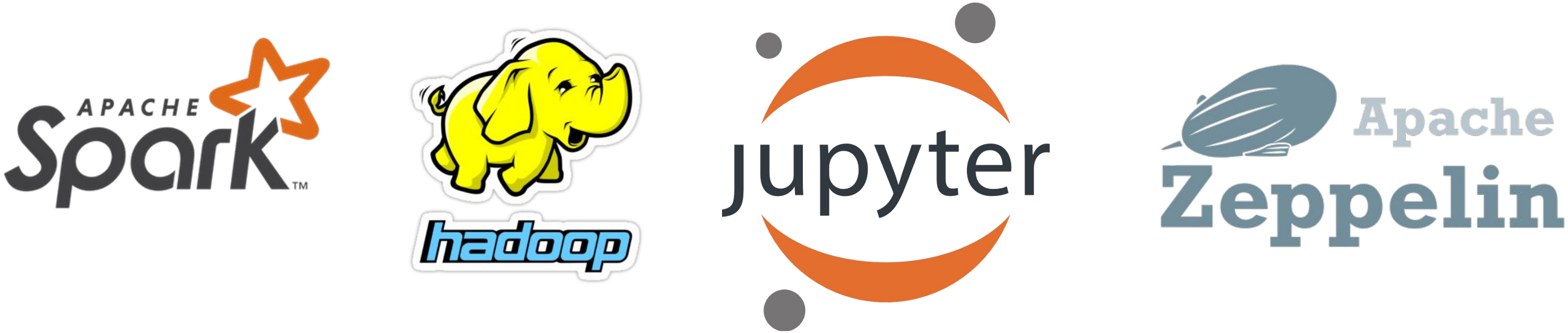


What is GeoMesa?

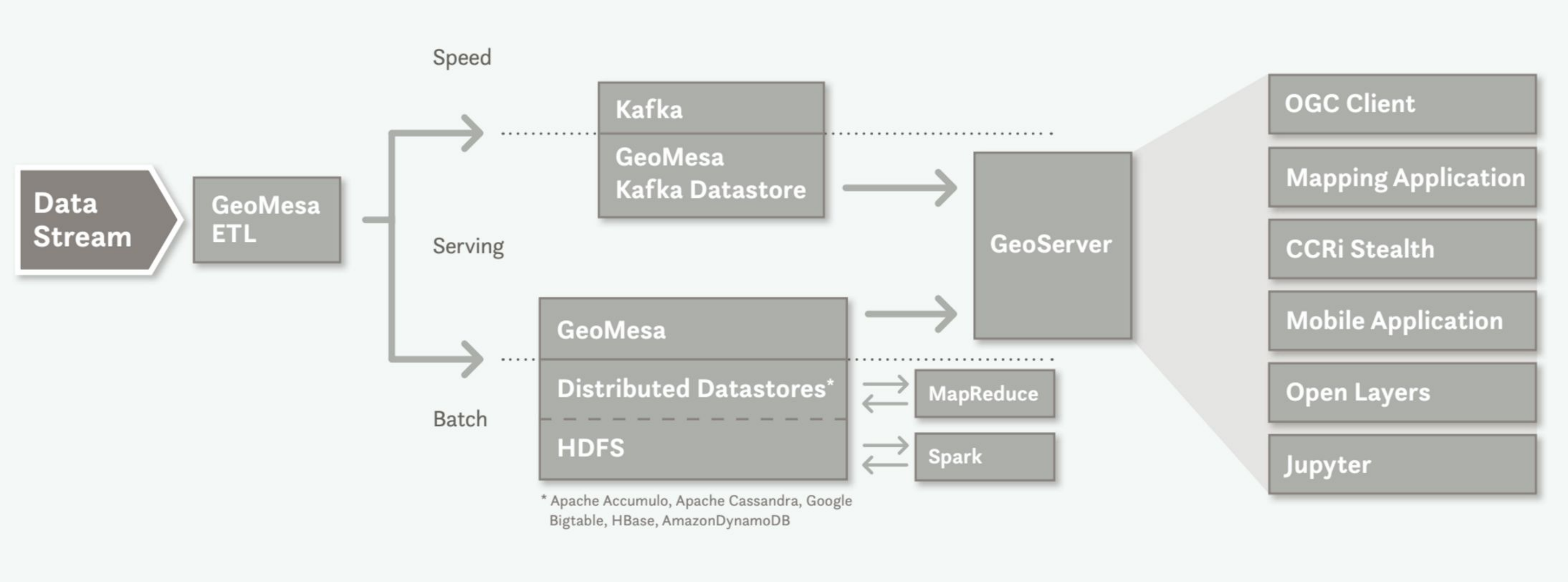
A suite of tools for streaming, persisting, **managing**, and analyzing spatio-temporal data at scale

What is GeoMesa?

A suite of tools for streaming, persisting, managing, and **analyzing** spatio-temporal data at scale



Proposed Reference Architecture



* Apache Accumulo, Apache Cassandra, Google Bigtable, HBase, AmazonDynamoDB

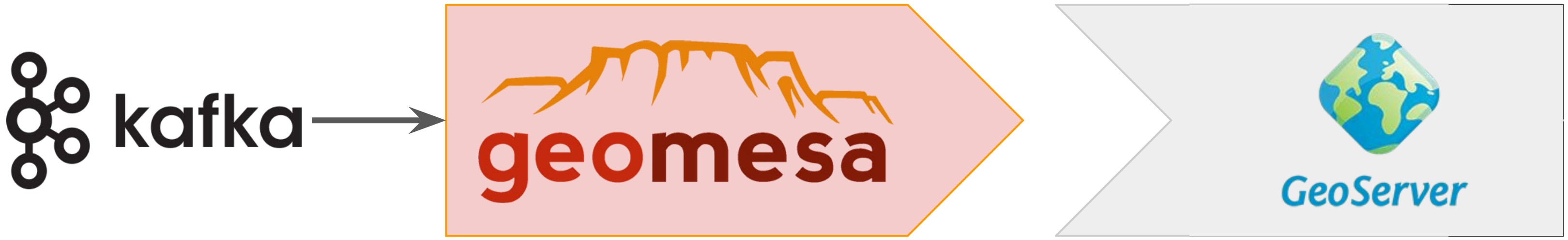
Streaming GeoSpatial Data

GeoMesa Kafka DataStore - In-Memory Database

GeoMesa KDS clients (like GeoServer)

1. Listen for updates from Kafka
2. Receive and answer spatial queries

These clients need an **in-memory database** structure that can be **updated quickly** as new updates come in.



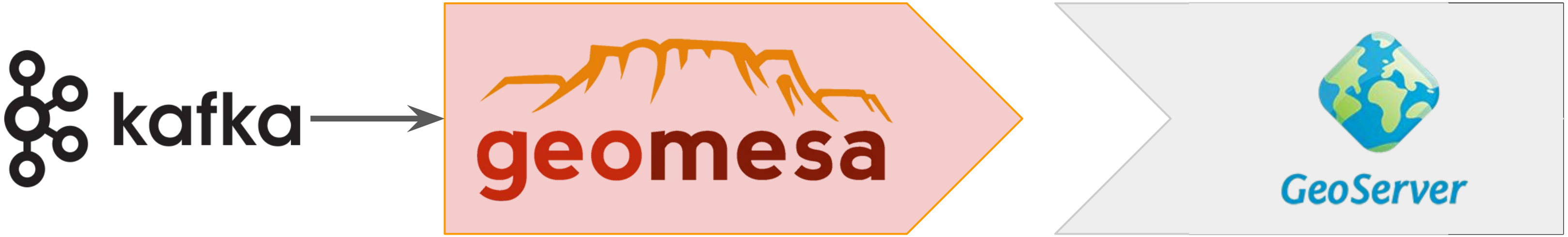
GeoMesa Kafka DataStore - In-Memory Database

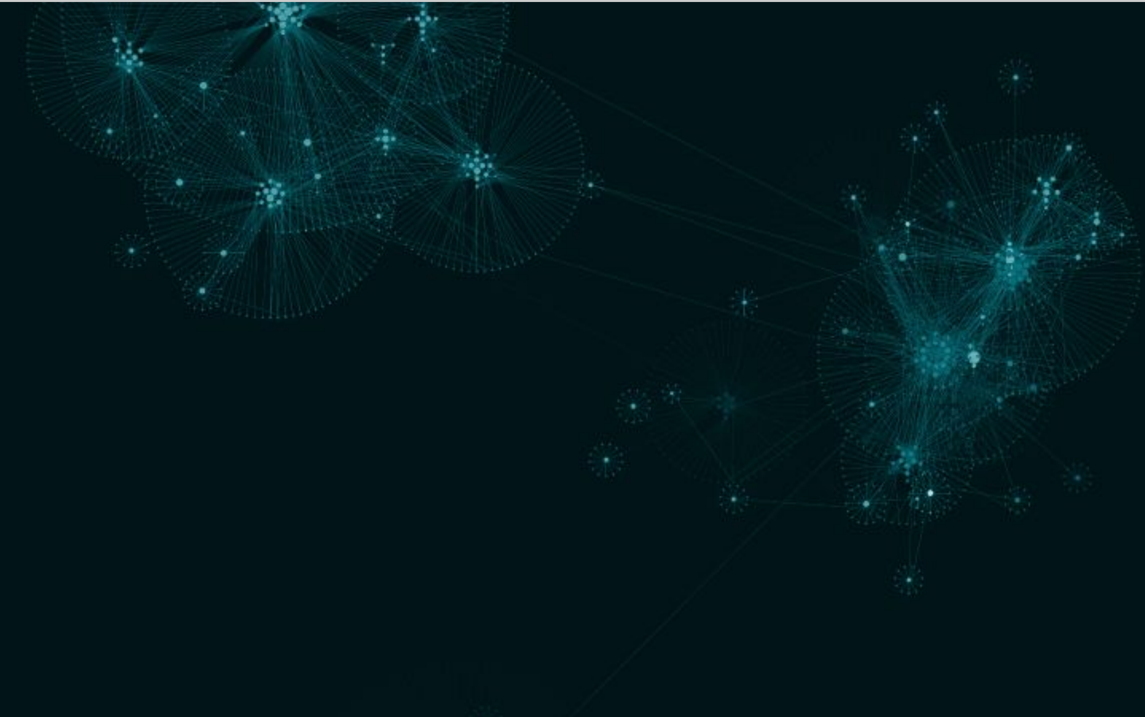
GeoMesa KDS clients (like GeoServer)

1. Listen for updates from Kafka
2. Receive and answer spatial queries

These clients need an **in-memory database** structure that can be **updated quickly** as new updates come in.

Effectively, the GeoMesa KDS is a **“spatially-enabled” KTable**.

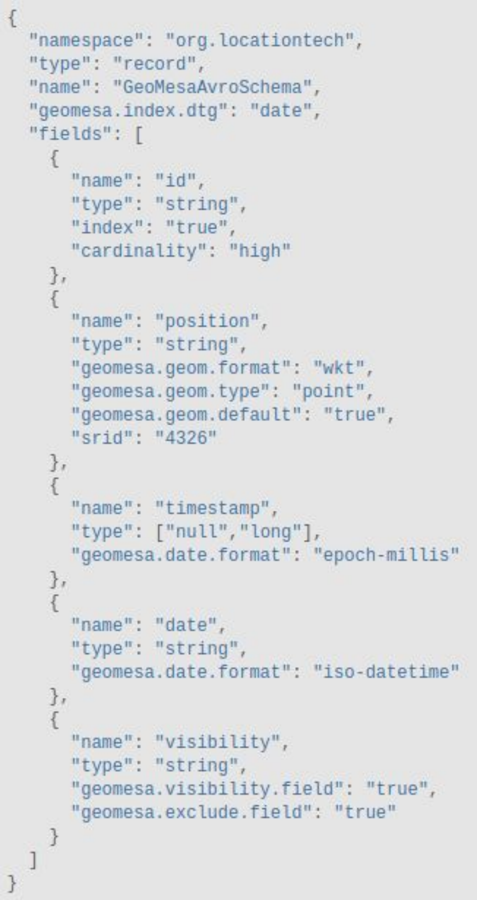




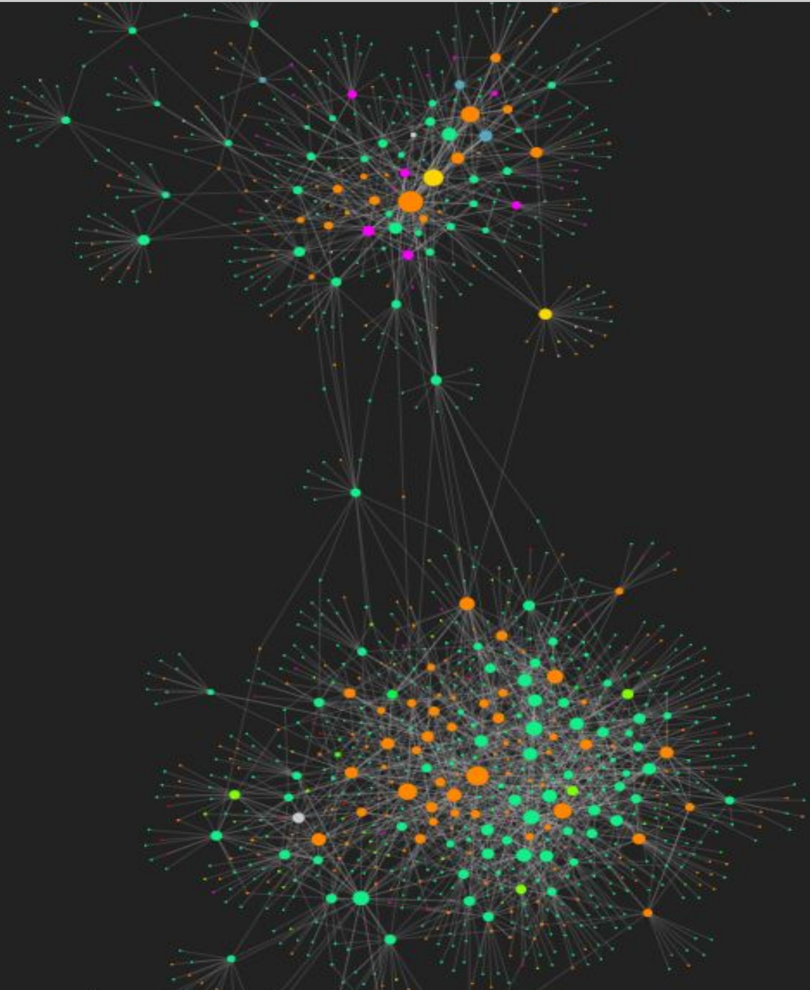
GeoMesa Kafka Confluent Schema Registry Integration
]

GeoMesa Confluent Kafka Data Store

- Extension of GeoMesa Kafka Data Store
- Convert Schema Registry Avro to SimpleFeatures
  - Converts a schema into a SimpleFeatureType
  - Without a converter
- Uses schema metadata to interpret fields that are not standard Avro
  - Geometry, Date, security, etc.
- Working on support for converting SimpleFeatures to Avro

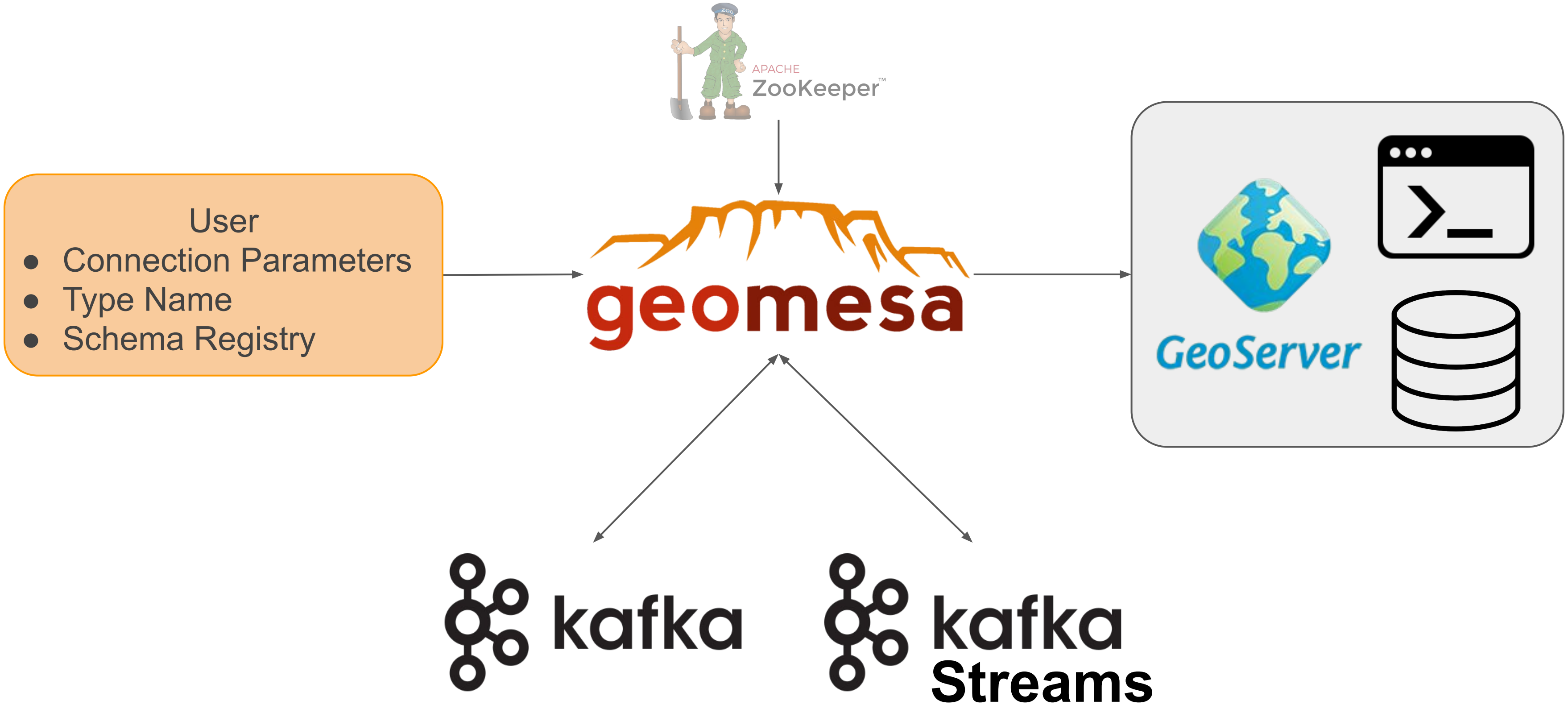


```
{
  "namespace": "org.locationtech",
  "type": "record",
  "name": "GeoMesaAvroSchema",
  "geomesa.index.dtg": "date",
  "fields": [
    {
      "name": "id",
      "type": "string",
      "index": "true",
      "cardinality": "high"
    },
    {
      "name": "position",
      "type": "string",
      "geomesa.geom.format": "wkt",
      "geomesa.geom.type": "point",
      "geomesa.geom.default": "true",
      "srid": "4326"
    },
    {
      "name": "timestamp",
      "type": ["null","long"],
      "geomesa.date.format": "epoch-millis"
    },
    {
      "name": "date",
      "type": "string",
      "geomesa.date.format": "iso-datetime"
    },
    {
      "name": "visibility",
      "type": "string",
      "geomesa.visibility.field": "true",
      "geomesa.exclude.field": "true"
    }
  ]
}
```



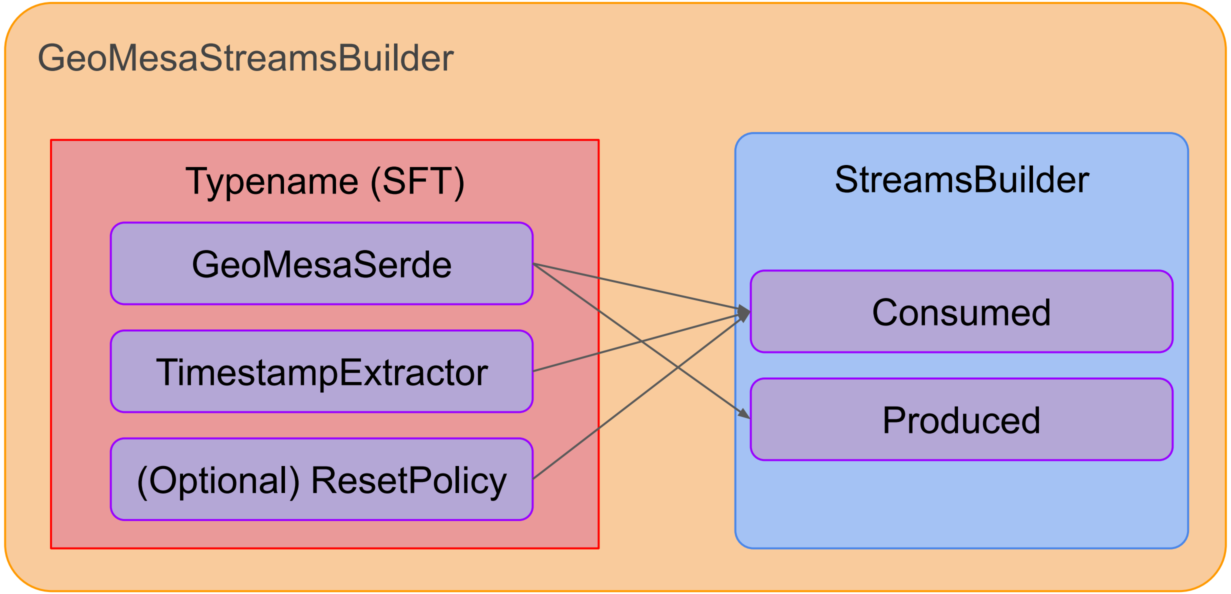
Stream Processing GeoSpatial Data

GeoMesa Kafka Streams Integration



User
- Connection Parameters
- Type Name
- Schema Registry

GeoMesa Kafka Streams Integration



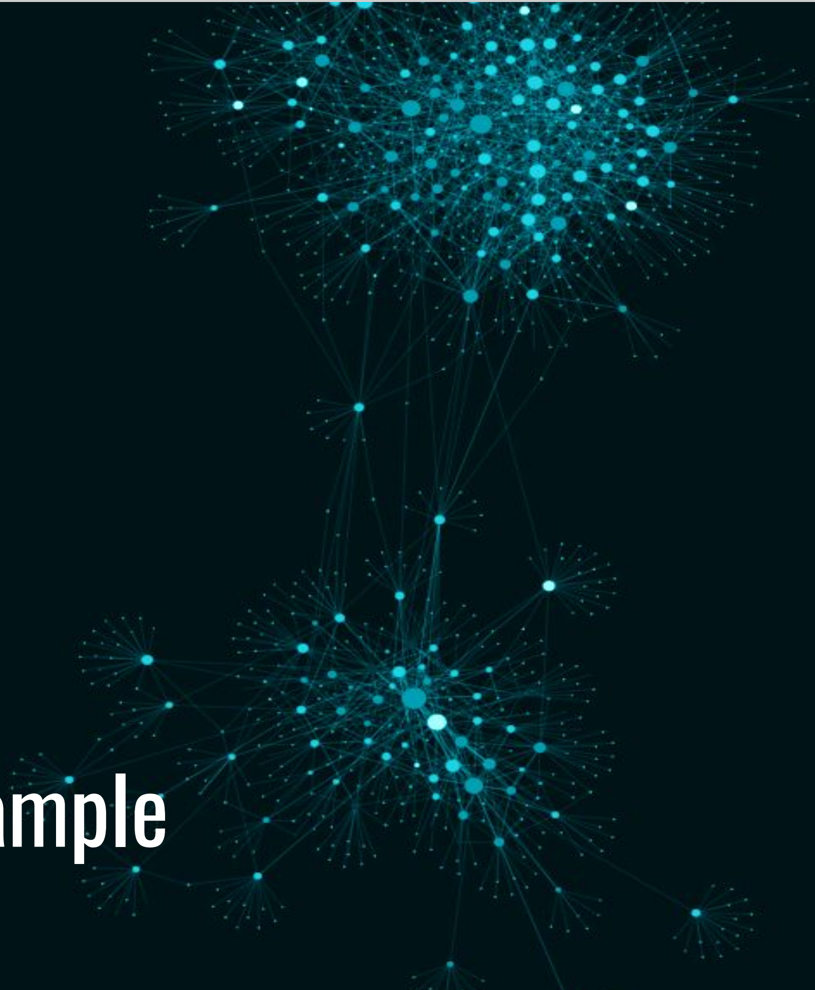
GeoMesa Kafka Topic

GeoMessage

- SimpleFeatures
  - Upsert / Change
- Control messages
  - Clear
  - Delete

GeoMesaMessage

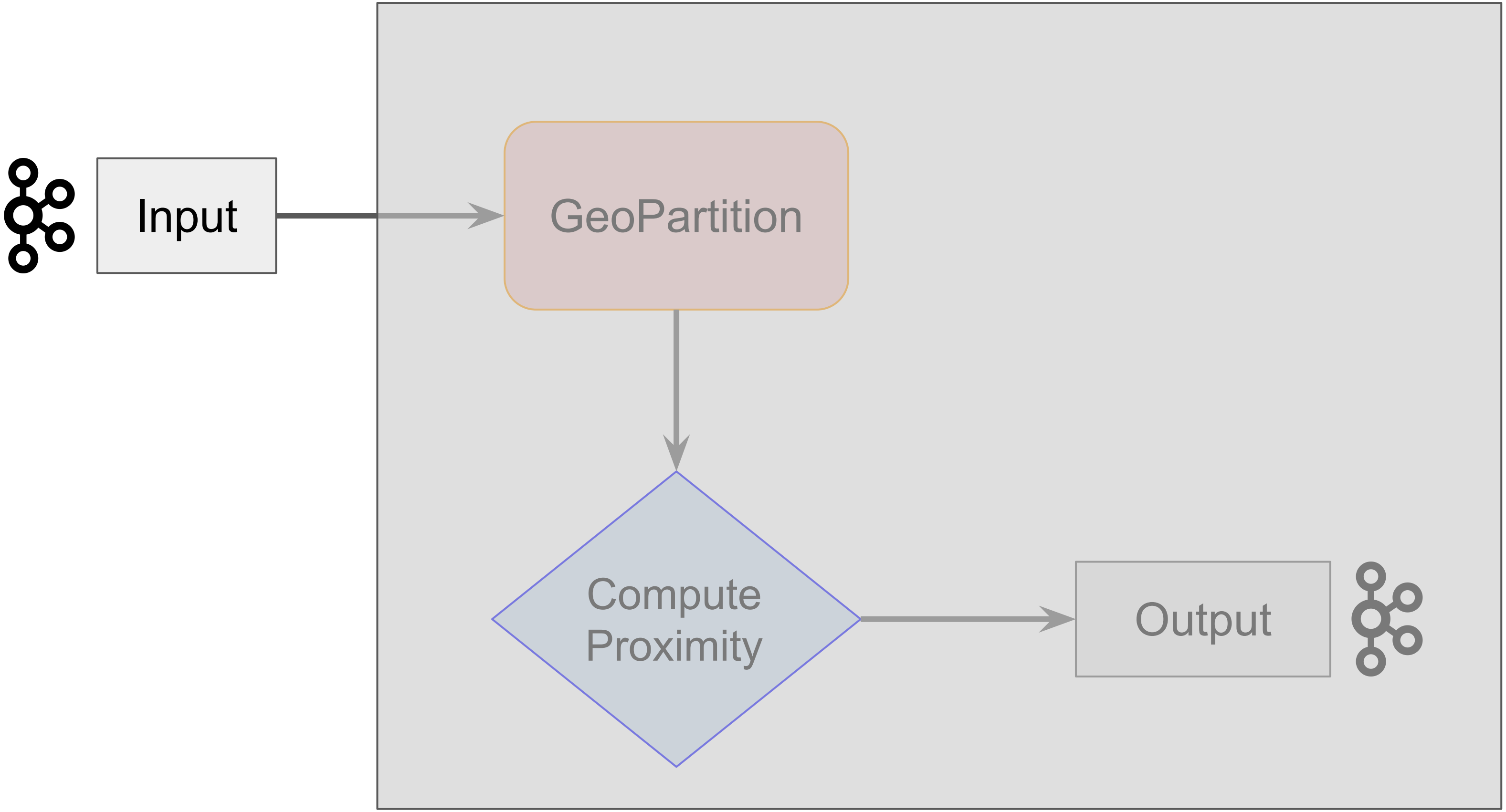
- SimpleFeatures
  - Data provided in String array
- Kafka Streams Serdes



Streaming Proximity Example

or close enough

Processing Architecture



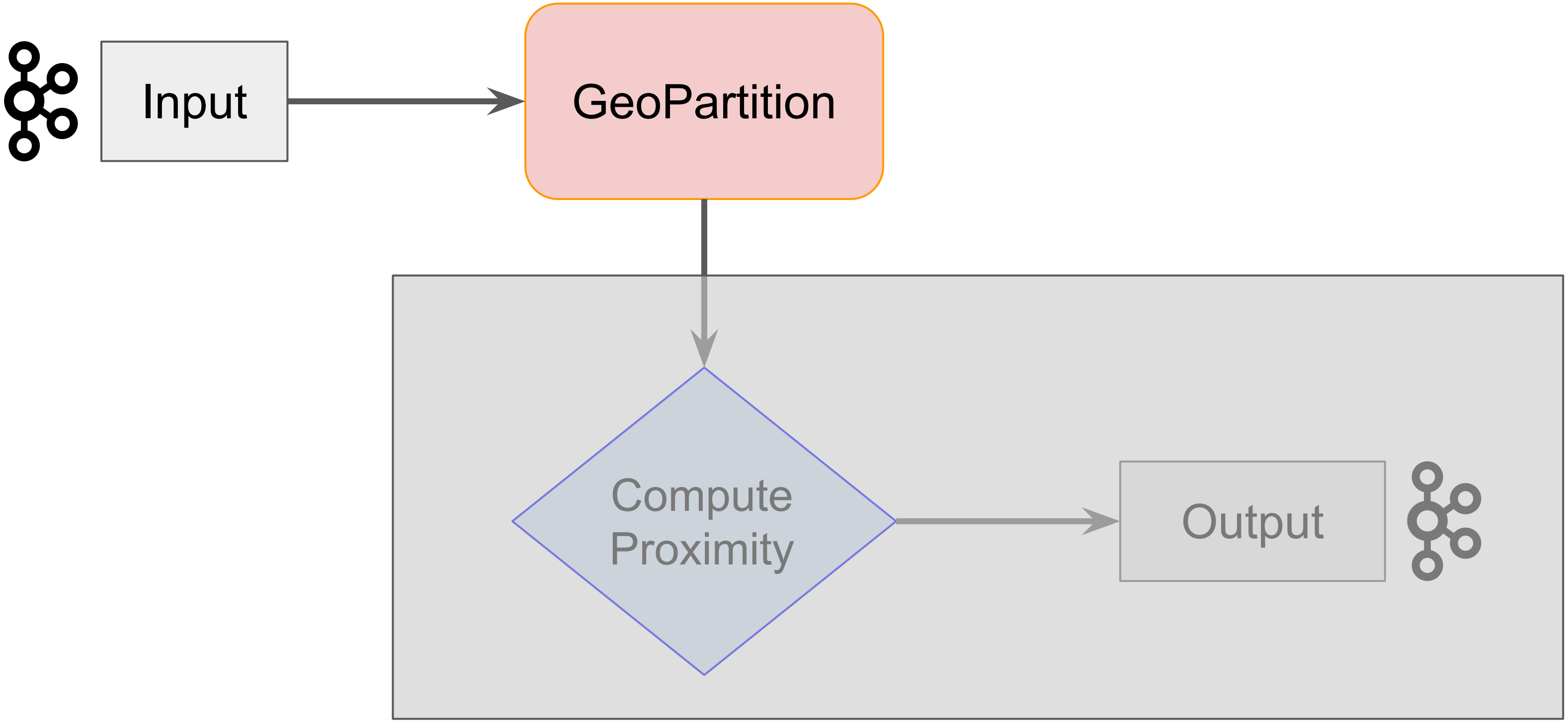
GeoMesa Kafka Streams Integration

Kafka Address
Consumer Count
etc

```
builder = GeoMesaStreamsBuilder.create(params,
                                       Topology.AutoOffsetReset.LATEST);

sft = DataStoreFinder.getDataStore(params).getSchema(typeName);


Serde<GeoMesaMessage> serde = builder.serde(typeName);

KStream<String, GeoMesaMessage> input = builder.stream(typeName);
```

‘mydataname’

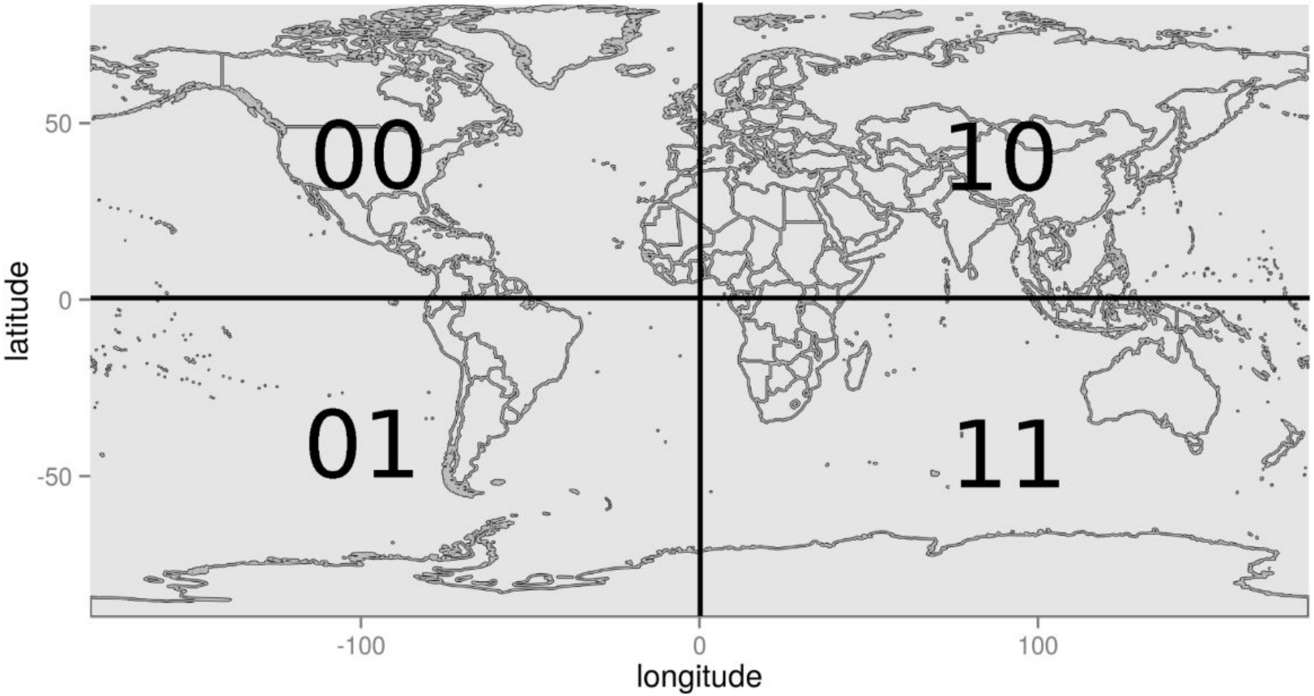
Processing Architecture



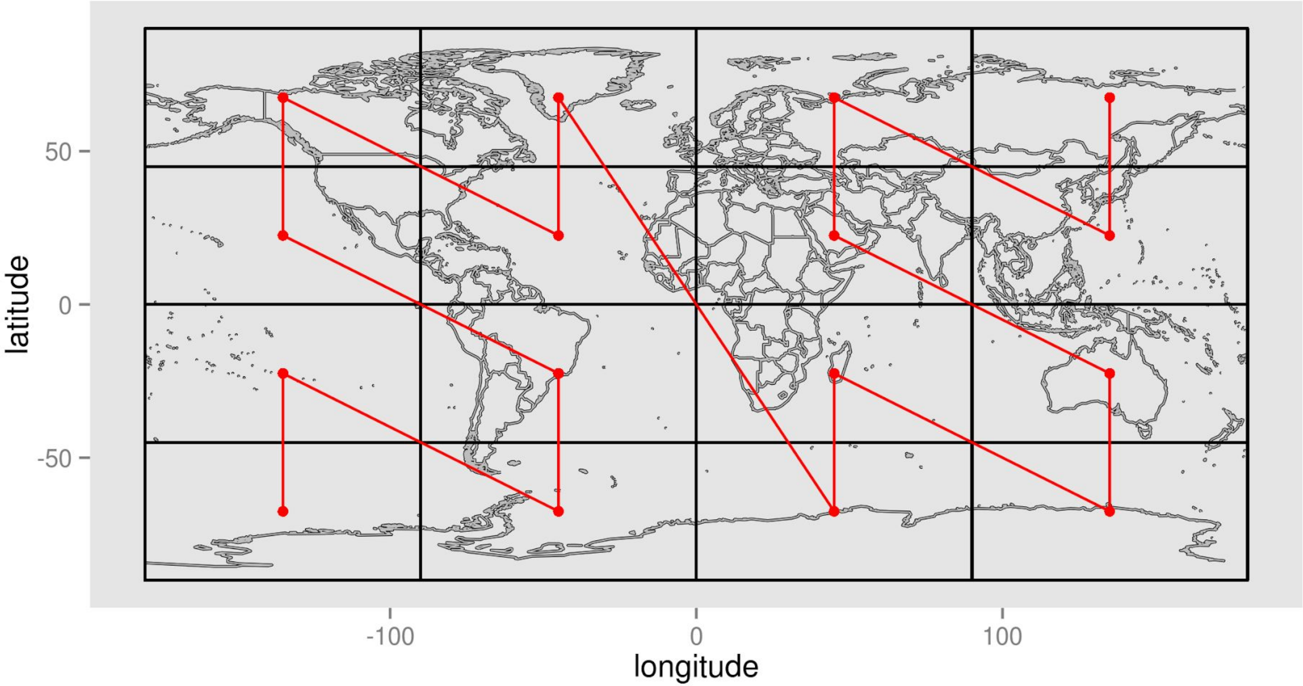
GeoMesa Kafka Streams Integration

```
Integer defaultGeomIndex = sft.indexOf(sft.getGeometryDescriptor()
                                          .getLocalName());

KStream<String, GeoMesaMessage> geoPartioned = input
   .selectKey(new GeoPartitioner(numbits, defaultGeomIndex));
```

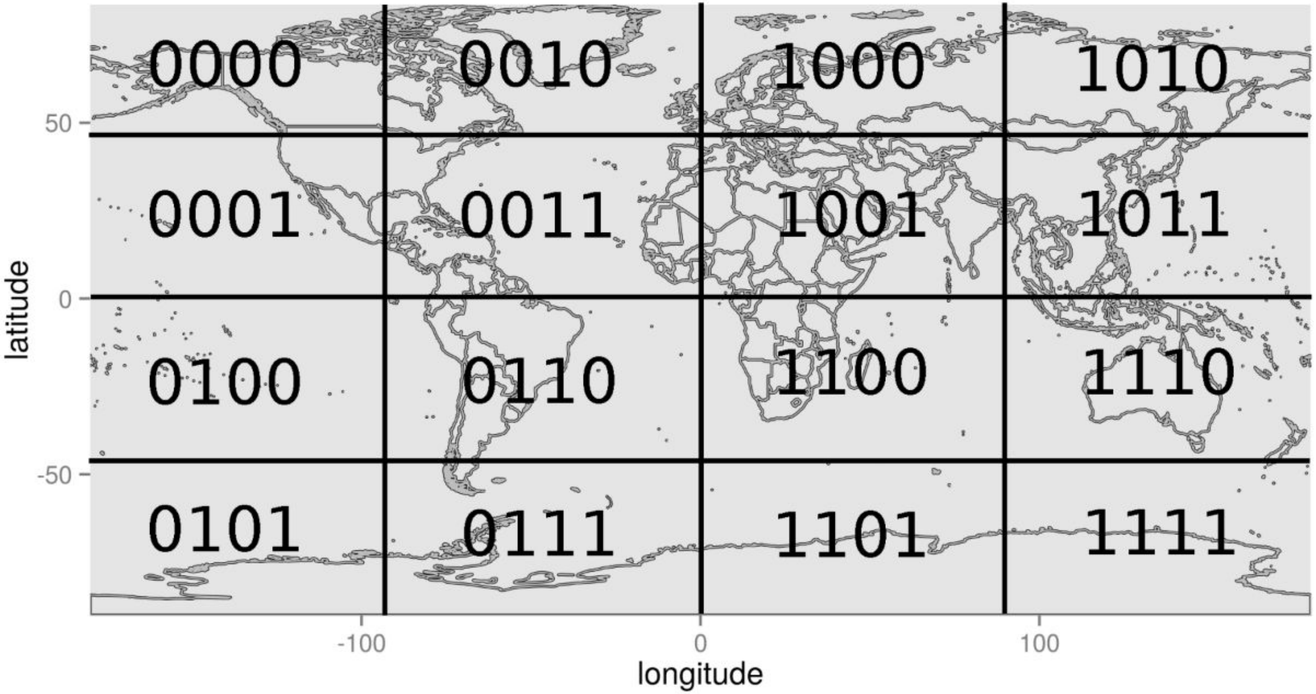
GeoHash



GeoHash

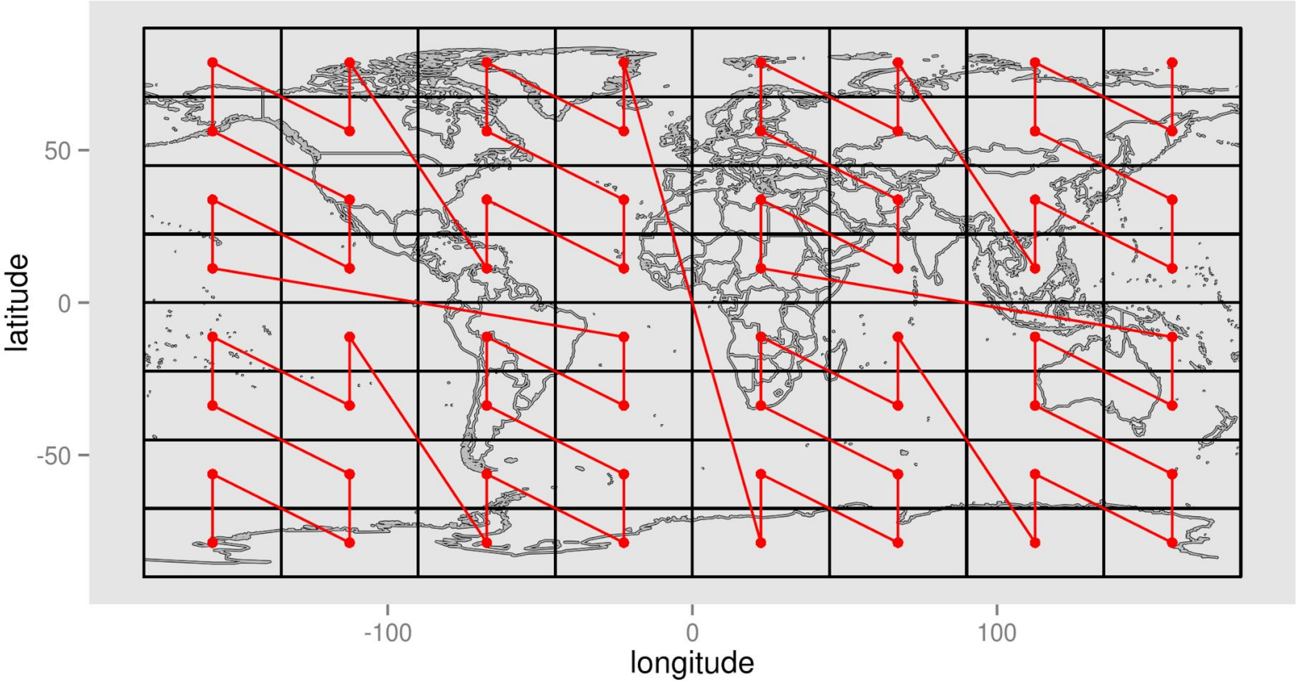


GeoHash



0000
0010
1000
1010
0001
0011
1001
1011
0100
0110
1100
1110
0101
0111
1101
1111
50
0
-50
latitude
-100
0
100
longitude

GeoHash



GeoPartioner

```
class GeoPartitioner implements KeyValueMapper<String, GeoMesaMessage,
                                               String> {
  …






}
```

GeoPartioner

```
class GeoPartitioner implements KeyValueMapper<String, GeoMesaMessage,
                                               String> {
  …

  @Override
  public String apply(String key, GeoMesaMessage value) {
    Geometry geom = (Geometry) value.attributes().apply(defaultGeomIndex);
    return getZBin(geom);
  }







}
```

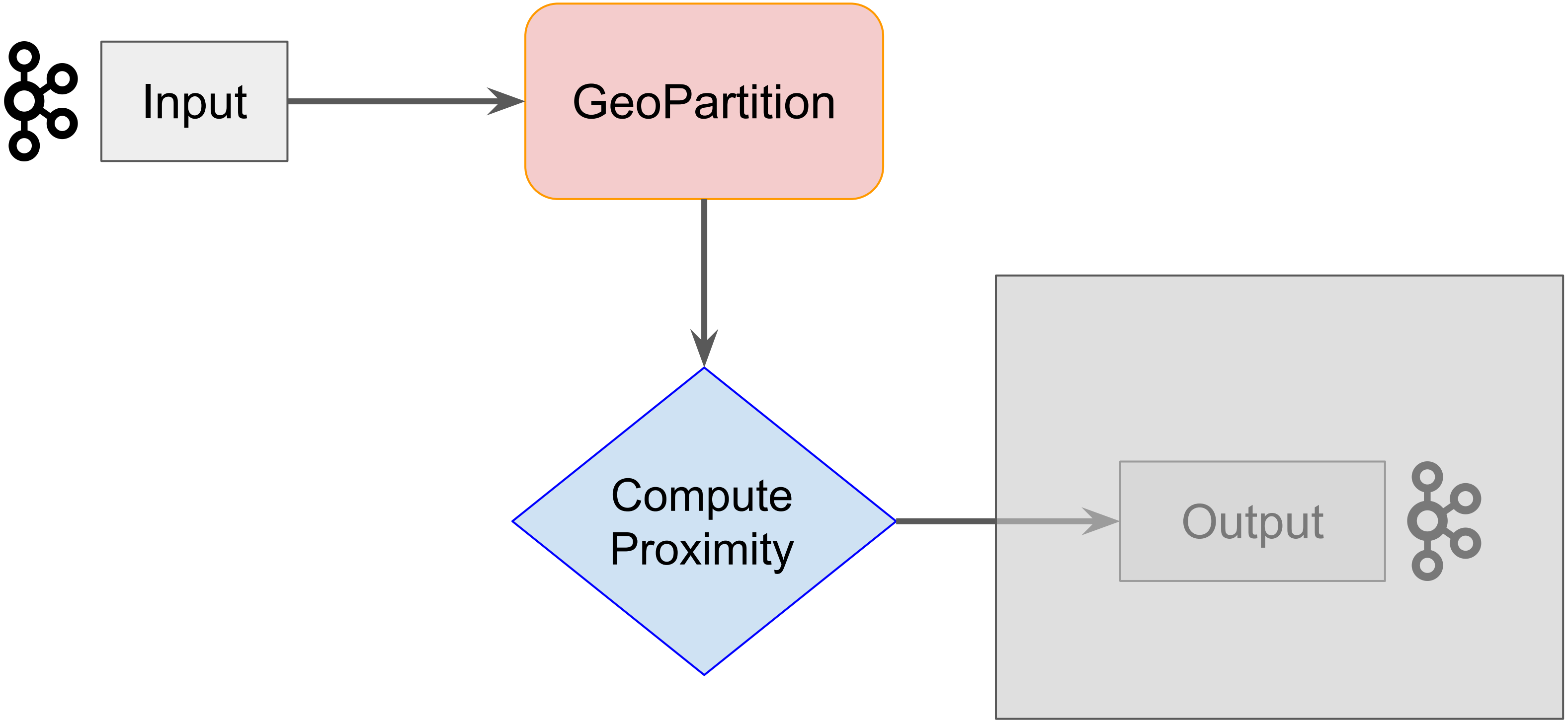
GeoPartioner

```
class GeoPartitioner implements KeyValueMapper<String, GeoMesaMessage,
                                               String> {
  …

  @Override
  public String apply(String key, GeoMesaMessage value) {
    Geometry geom = (Geometry) value.attributes().apply(defaultGeomIndex);
    return getZBin(geom);
  }

  private String getZBin(Geometry geom) {
    Point safeGeom =
GeohashUtils.getInternationalDateLineSafeGeometry(geom).get().getCentroid();
    Long index = z2.index(safeGeom.getX(), safeGeom.getY(), false);
    return String.format("%0" + partitionNumBits + "d", index);
  }
}
```

GeoMesa Kafka Streams Integration

```
Integer defaultGeomIndex = sft.indexOf(sft.getGeometryDescriptor()
                                          .getLocalName());

KStream<String, GeoMesaMessage> geoPartioned = input
   .selectKey(new GeoPartitioner(numbits, defaultGeomIndex));
```

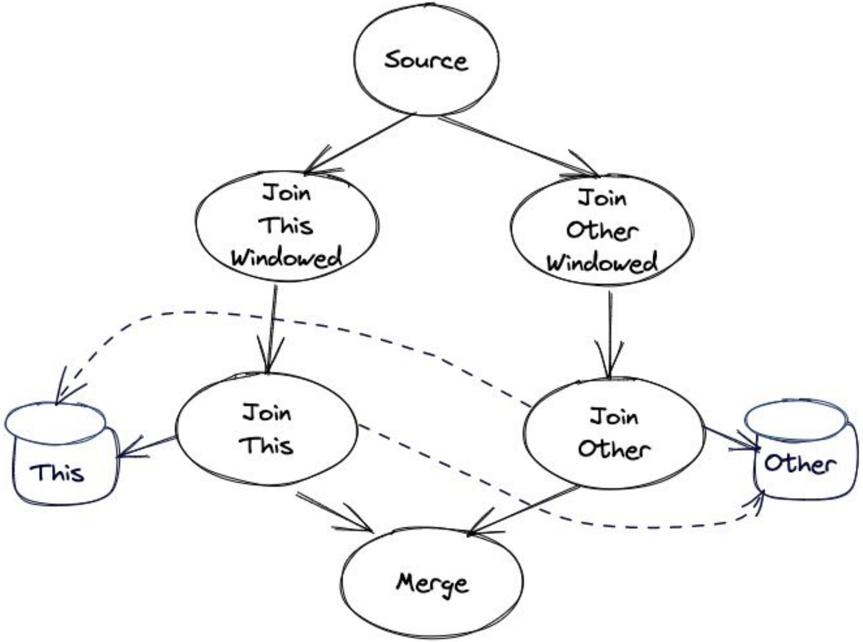
Processing Architecture



GeoSpatial Self Join

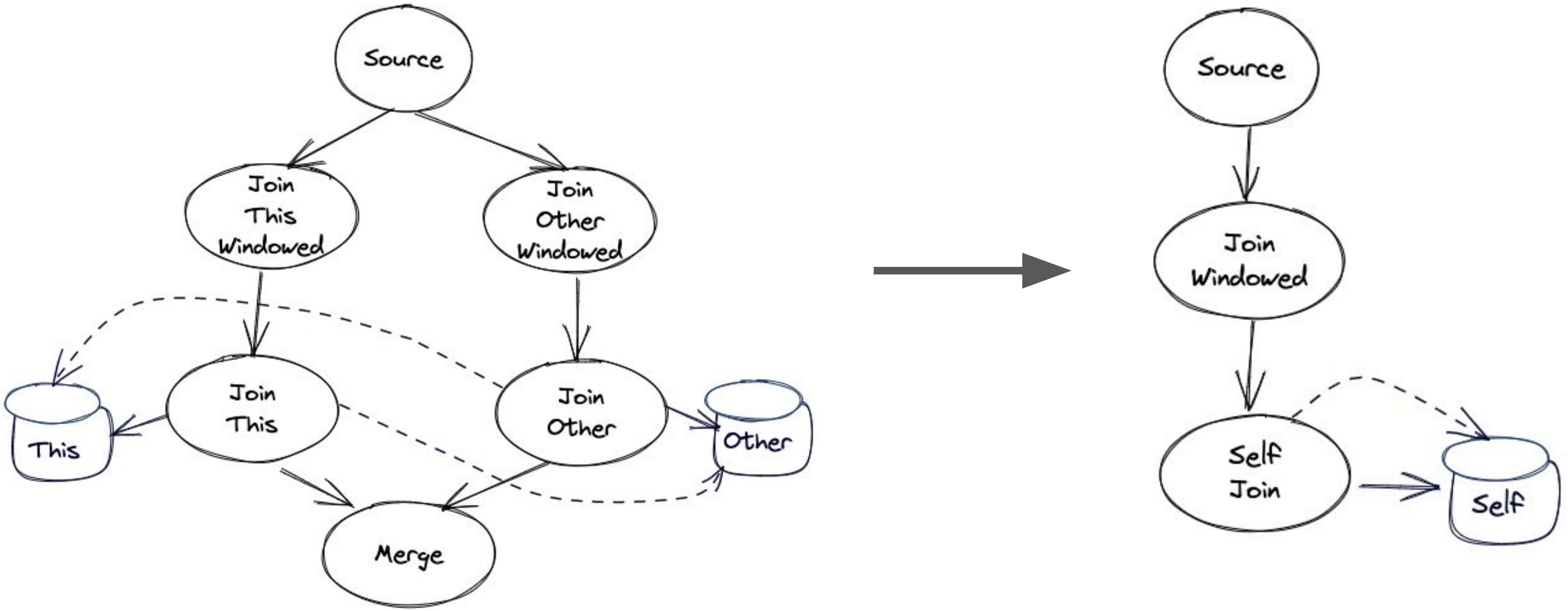
```
KStream<String, GeoMesaMessage> proximities = geoPartioned
    .join(geoPartioned,
          (left, right) -> new Proximity(left, right, defaultGeomIndex),
          JoinWindows.of(Duration.ofMinutes(2)),
          Joined.with(Serdes.String(), serde, serde))
```

Self Join Optimizations



Self Join Optimizations

KIP-862: Self-join optimization for stream-stream joins



GeoSpatial Self Join

```
KStream<String, GeoMesaMessage> proximities = geoPartioned
    .join(geoPartioned,
          (left, right) -> new Proximity(left, right, defaultGeomIndex),
          JoinWindows.of(Duration.ofMinutes(2)),
          Joined.with(Serdes.String(), serde, serde))
    .filter((k, v) -> v.areDifferent() && v.areNotProximities() &&
                      v.getDistance() < proximityDistanceMeters)
```

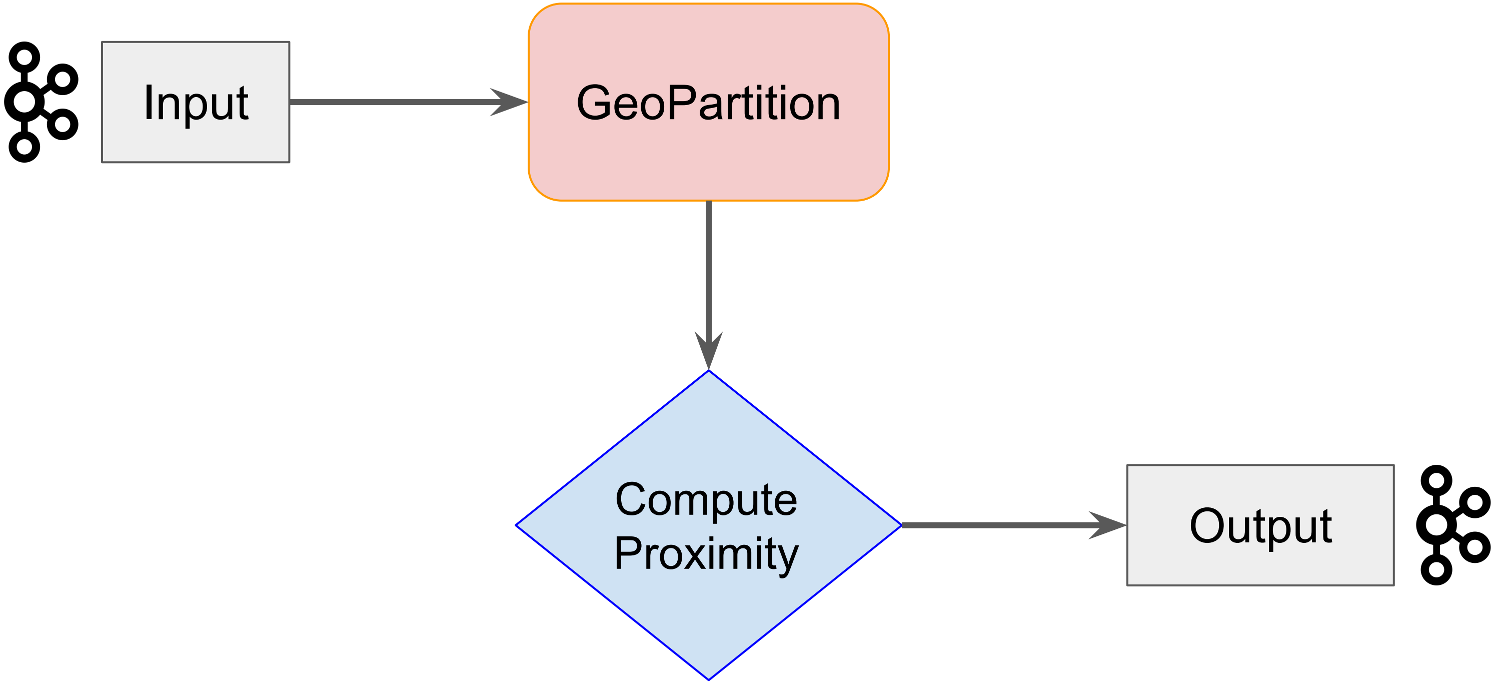
GeoSpatial Self Join

```
KStream<String, GeoMesaMessage> proximities = geoPartioned
    .join(geoPartioned,
          (left, right) -> new Proximity(left, right, defaultGeomIndex),
          JoinWindows.of(Duration.ofMinutes(2)),
          Joined.with(Serdes.String(), serde, serde))
    .filter((k, v) -> v.areDifferent() && v.areNotProximities() &&
                      v.getDistance() < proximityDistanceMeters)
    .mapValues(Proximity::toGeoMesaMessage)
```

GeoSpatial Self Join

```
KStream<String, GeoMesaMessage> proximities = geoPartioned
    .join(geoPartioned,
          (left, right) -> new Proximity(left, right, defaultGeomIndex),
          JoinWindows.of(Duration.ofMinutes(2)),
          Joined.with(Serdes.String(), serde, serde))
    .filter((k, v) -> v.areDifferent() && v.areNotProximities() &&
                      v.getDistance() < proximityDistanceMeters)
    .mapValues(Proximity::toGeoMesaMessage)
    .selectKey((k, v) -> proximityId + UUID.randomUUID());
```

Processing Architecture



Output

NB: Kafka Streams uses stream.to(‘topic’)

```
builder.to(typeName, proximities);
```

CHARLOTTESVILLE
Louisa
Lake Anna
Bowling Green
Scottsville
Columbia
Ashland
Glen Allen
Short Pump
Mechanicsville
Lakeside
Tuckahoe
Powhatan
RICHMOND
Highland Springs
Bon Air
Midlothian
Cumberland

CHARLOTTESVILLE
Louisa
Lake Anna
Bowling Green
Scottsville
Columbia
Ashland
Glen Allen
Short Pump
Mechanicsville
Lakeside
Tuckahoe
RICHMOND
Highland Springs
Bon Air
Midlothian
Powhatan
Cumberland

Future Optimizations

- GeoPartioning Boundary Problem
- Z2 aware ConsumerPartitionAssignor

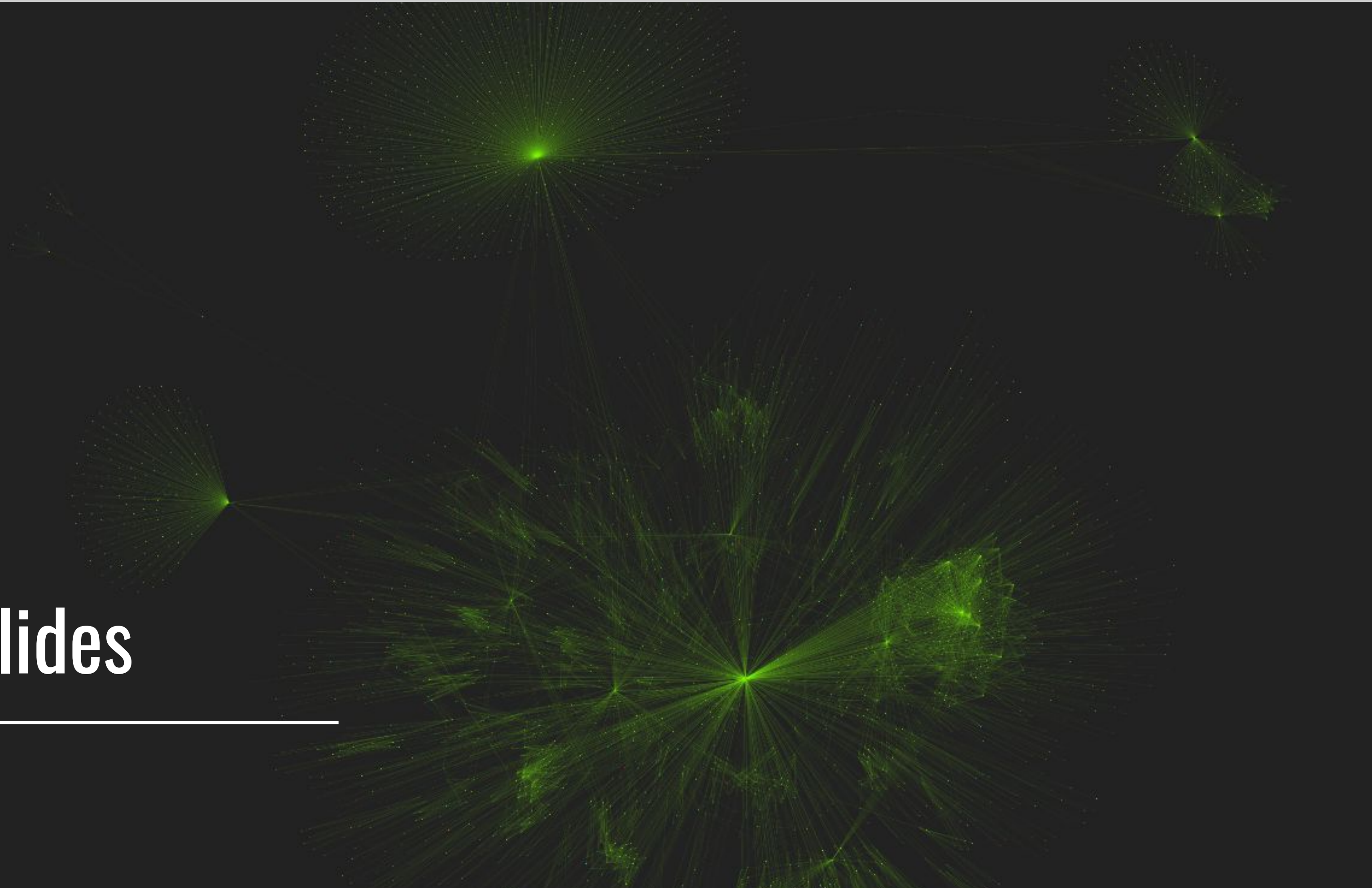
Resources, Questions and Eye Candy

GeoMesa: https://www.geomesa.org

GeoMesa Tutorials and Quickstarts: https://github.com/geomesa/geomesa-tutorials

GeoMesa Kafka Streams Quickstart: https://github.com/geomesa/geomesa-tutorials/pull/88

Gitter: https://gitter.im/locationtech/geomesa



Backup Slides

Kafka DataStore

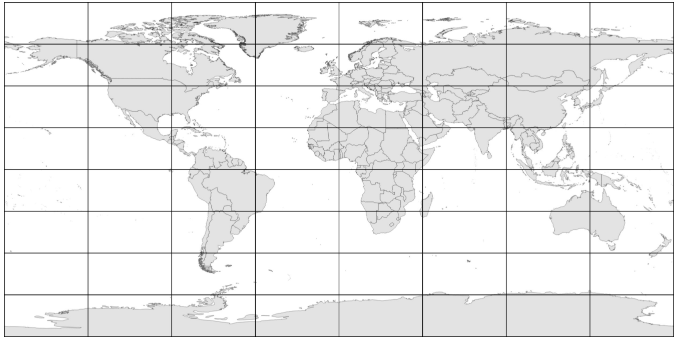
GeoMesa Kafka DataStore In-Memory Database

For most use cases, GeoMesa uses a class which maintains two things:

1. A HashMap of Feature IDs to records
2. A bucket index of spatial grid cells containing records

Updates:

- Find the old record in the HashMap
- Remove it from the bucket index
- Add the new element



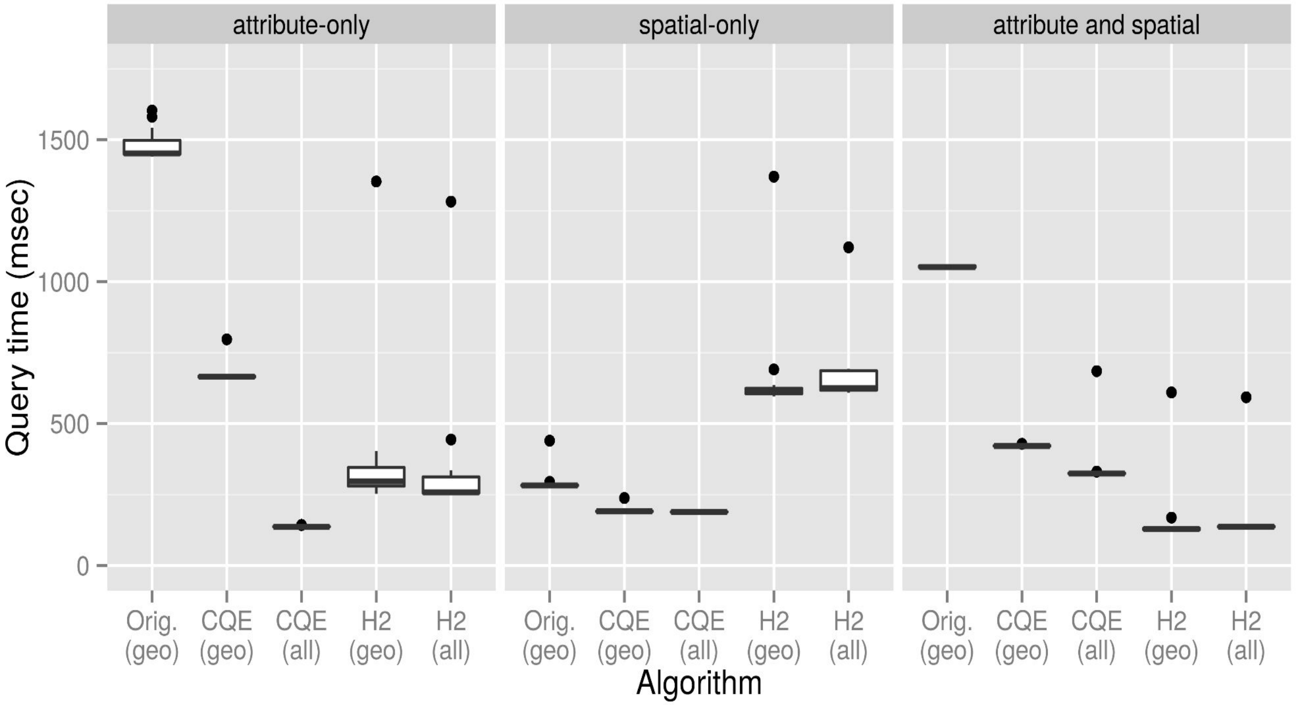
GeoMesa Kafka DataStore In-Memory Database

For situations when queries on attribute columns may be important, GeoMesa can be configured to use CQEngine!

For GeoServer use cases, it is faster than the standard KDS and H2.

- Hughes, Zimmerman, Eichelberger, and Fox. "A survey of techniques and open-source tools for processing streams of spatio-temporal events". Conference: the 7th ACM SIGSPATIAL International Workshop on GeoStreaming. October 2016. DOI: 10.1145/3003421.3003432

GeoMesa Kafka DataStore In-Memory Database



What tools are there?

Kafka has command line tools

- Manage topics
- Send messages
- Listen to topics

GeoMesa Kafka has command line tools

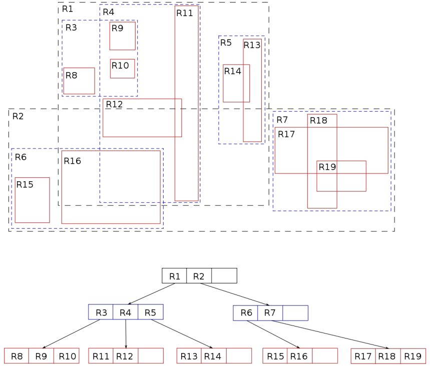
- Manage SimpleFeatureTypes
- Send SimpleFeatures as messages
- Listen to topics

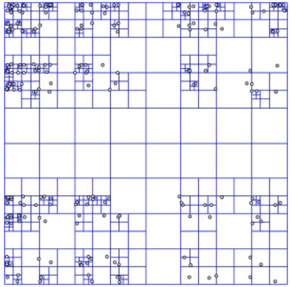
GeoMesa Kafka DataStore In-Memory Database

GeoMesa KDS clients (like GeoServer)

1. Listen for updates from Kafka
2. Receive and answer spatial queries

These clients need an **in-memory database** structure that can be **updated quickly** as new updates come in.





R-Trees and Quad-trees are **slow** to update with scale.

Other possibilities include trying H2's spatial support. Indexing in H2 was slow when we tried it. (Admittedly, back in 2016.)

To address this, GeoMesa has rolled its own lightweight, in-memory database.

Kafka Streams

Time in Kafka Streams

- Processing Time
  - now() when data is being processed
- Event Time
  - Point in time when event or data occurred
- Ingestion Time
  - Time when event is stored in a topic partition

Timestamp

Assigned to every Event

Defaults to Ingestion Time

TimestampExtractor

- Used to pull event time out of record

Stream Time

- Data driven time of stream
- Only progressed when data timestamps do

Parallelism

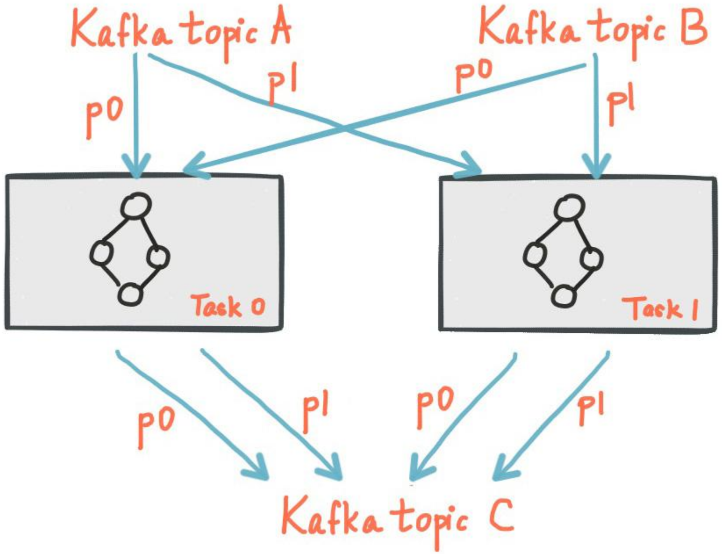
Kafka

- partitions for storage and transport parallelism

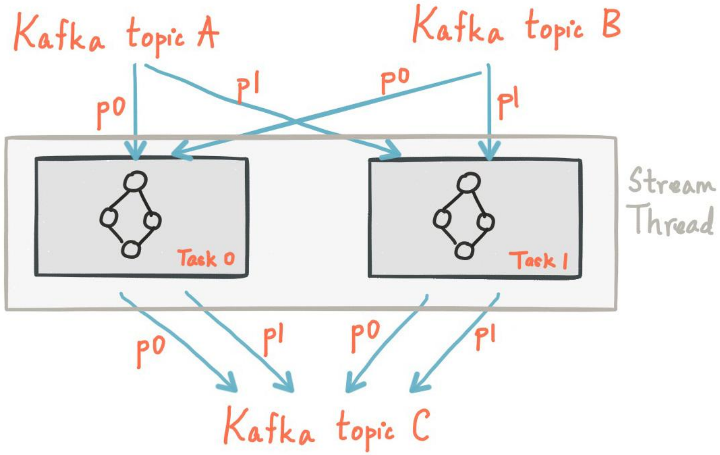
Kafka Streams

- partitions for compute and processing parallelism

Maximum parallelism == # partitions
Keys determine partition and thus processing order



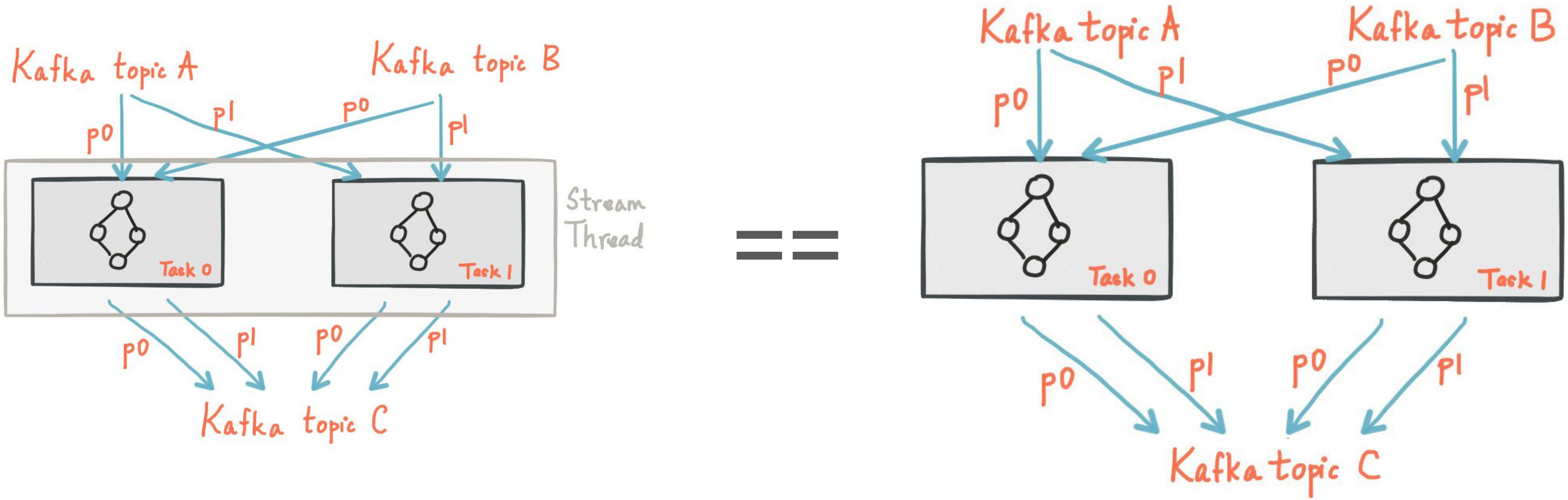
Parallelism



One Streams App can run multiple topology threads

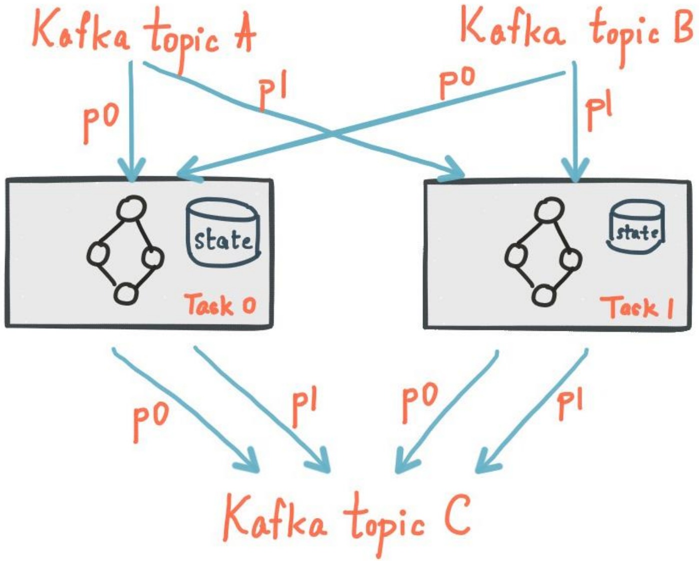
- Are all independent and do not share resources
- No inter-thread coordination needed

Parallelism



State Stores

Each Instance keeps copy of its state store partitions



Fault Tolerance

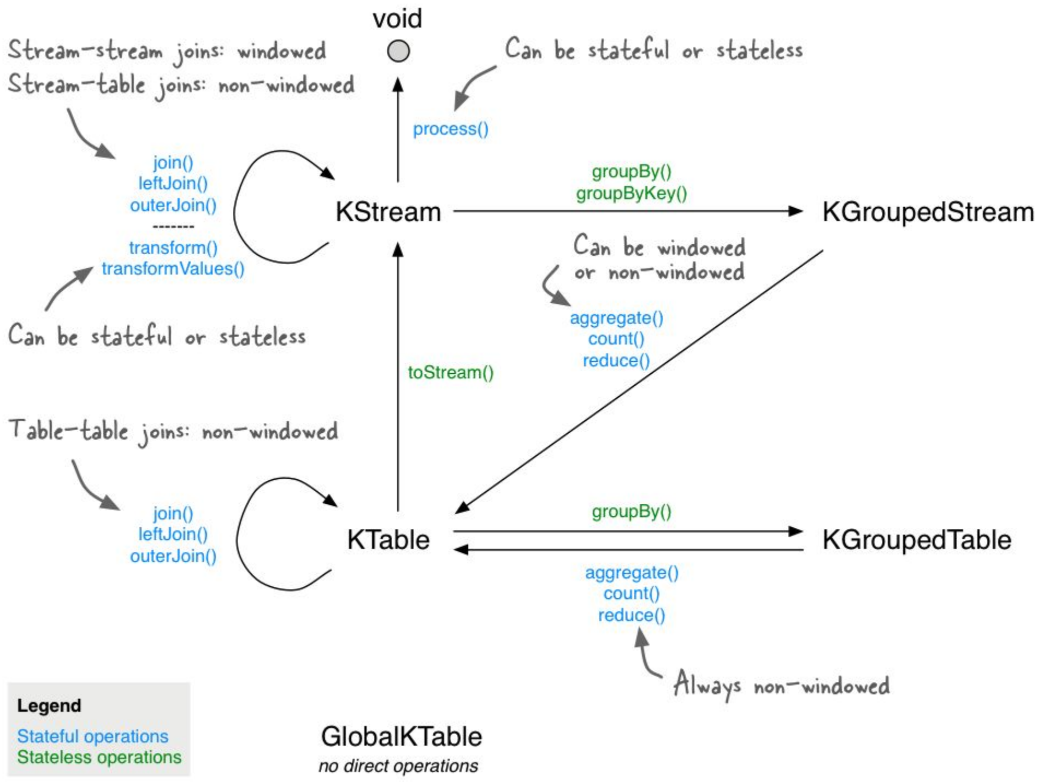
State

- Uses Kafka topics
  - Has Producer/Consumer tolerance capabilities
- Event processing transactions include State commits
- Can be restored from changelog topics
  - Log compaction reduces overhead

Failures

- Partitions are re-distributed to consumer group members
- Standby replicas will keep all state store partitions warm

Summary of Kafka Streams operations



Motivation / Use Case

- Original use case was to remove GeoMesa from some analytics
  - Analytics can output Avro with the Schema Registry
  - While minimizing additional processing to get features in GeoServer
    - Avoids the standard solution of using NiFi with a converter
    - Less processing overhead, less development
- Encourages more third-party integration with GeoMesa
  - Avro is a common data format
    - Fewer barriers to entry for new GeoMesa users
  - Opens up the door for other Kafka stream processing tools
    - KStreams, ksqlDB