DATA+AI
SUMMIT 2022

MOSAIC

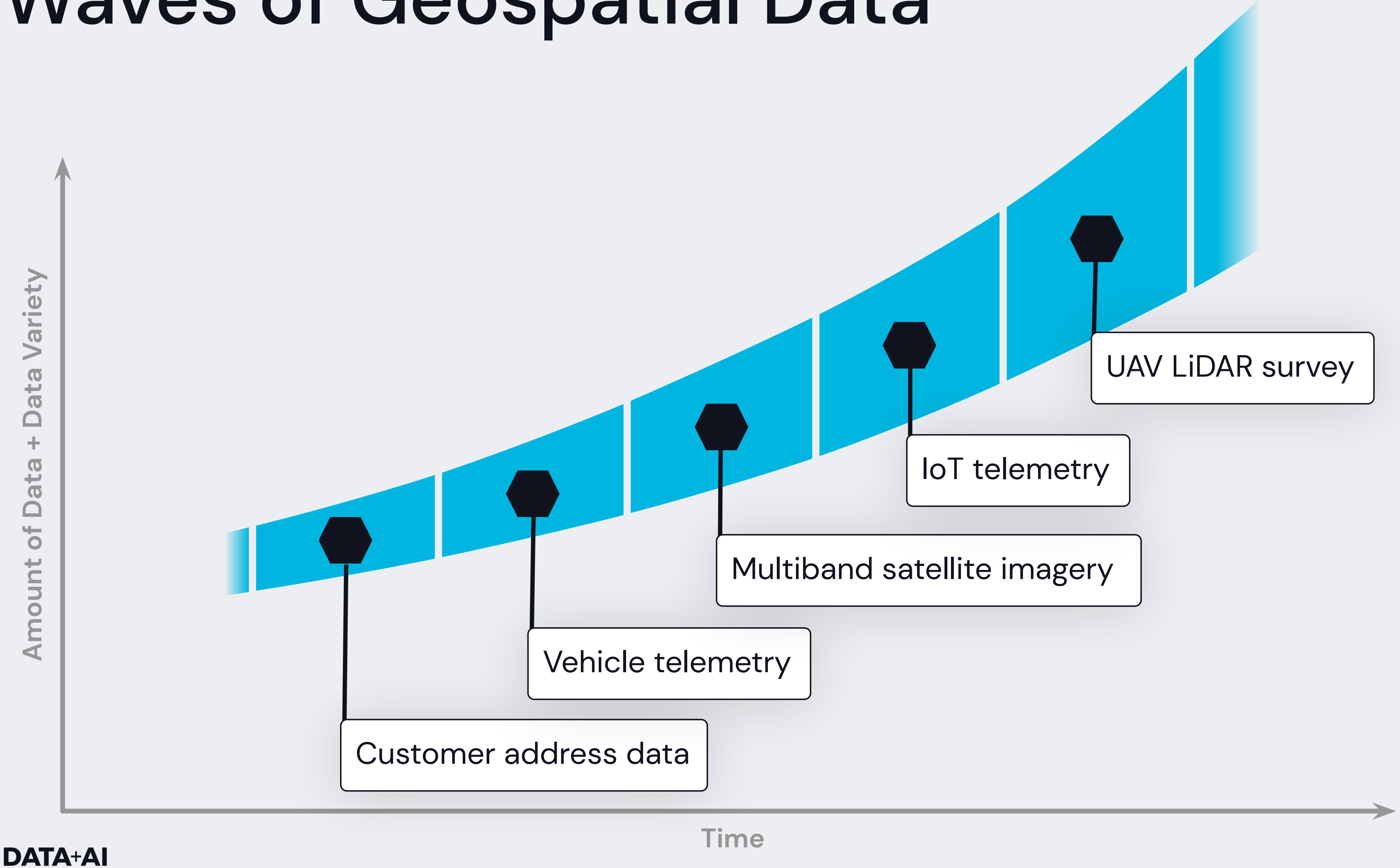
A Framework for Geospatial Analytics at Scale

Milos Colic
Sr. Solutions Architect,
Databricks

Stuart Lynn
Sr. Solutions Architect,
Databricks

ORGANIZED BY databricks

Waves of Geospatial Data



Efficient Point in Polygon Joins via PySpark and BNG Geospatial Indexing

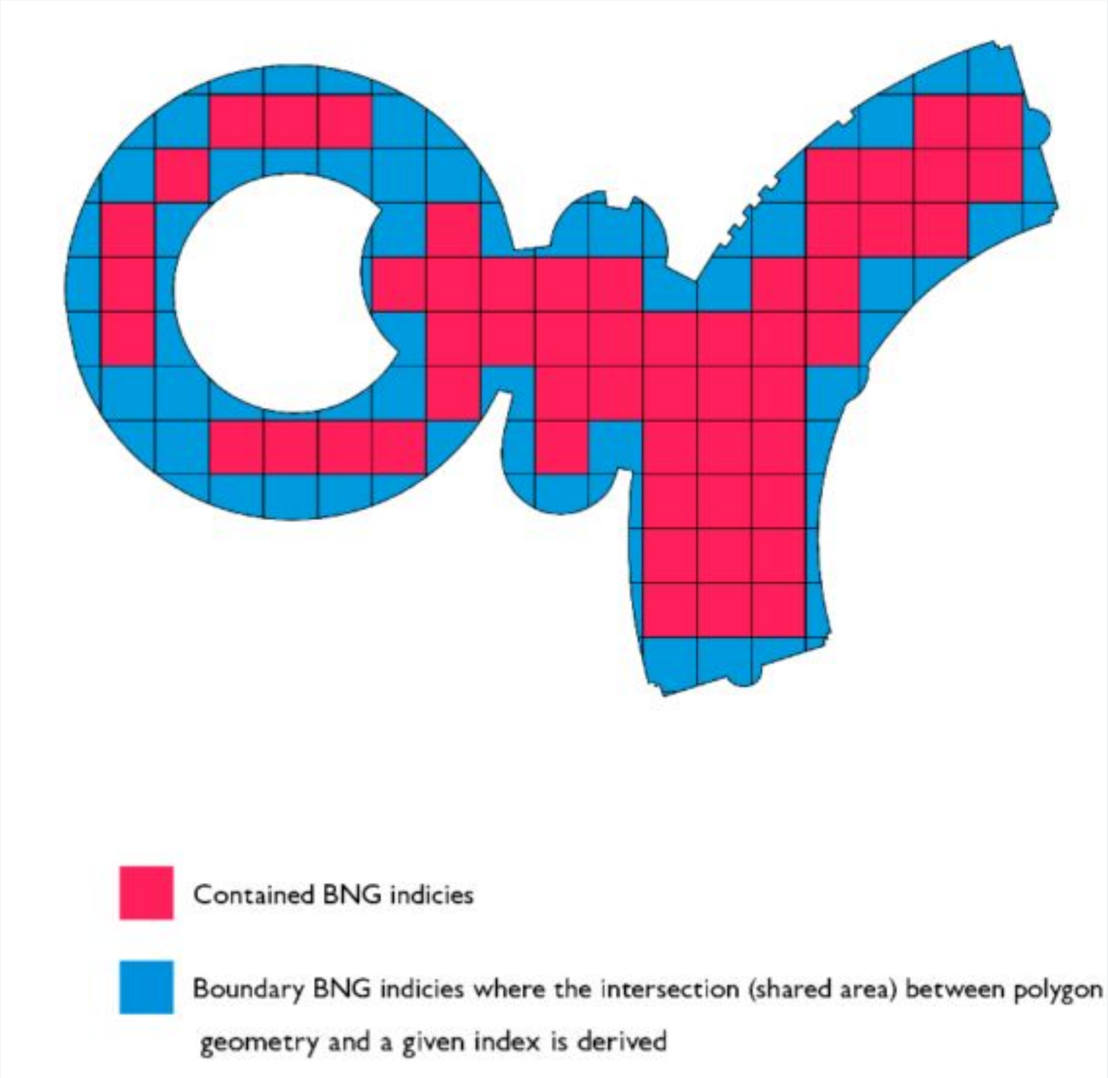
by Milos Colic , Robert Whiffin , Pritesh Patel , Charis Doidge , Steve Kingston and Linda Sheard

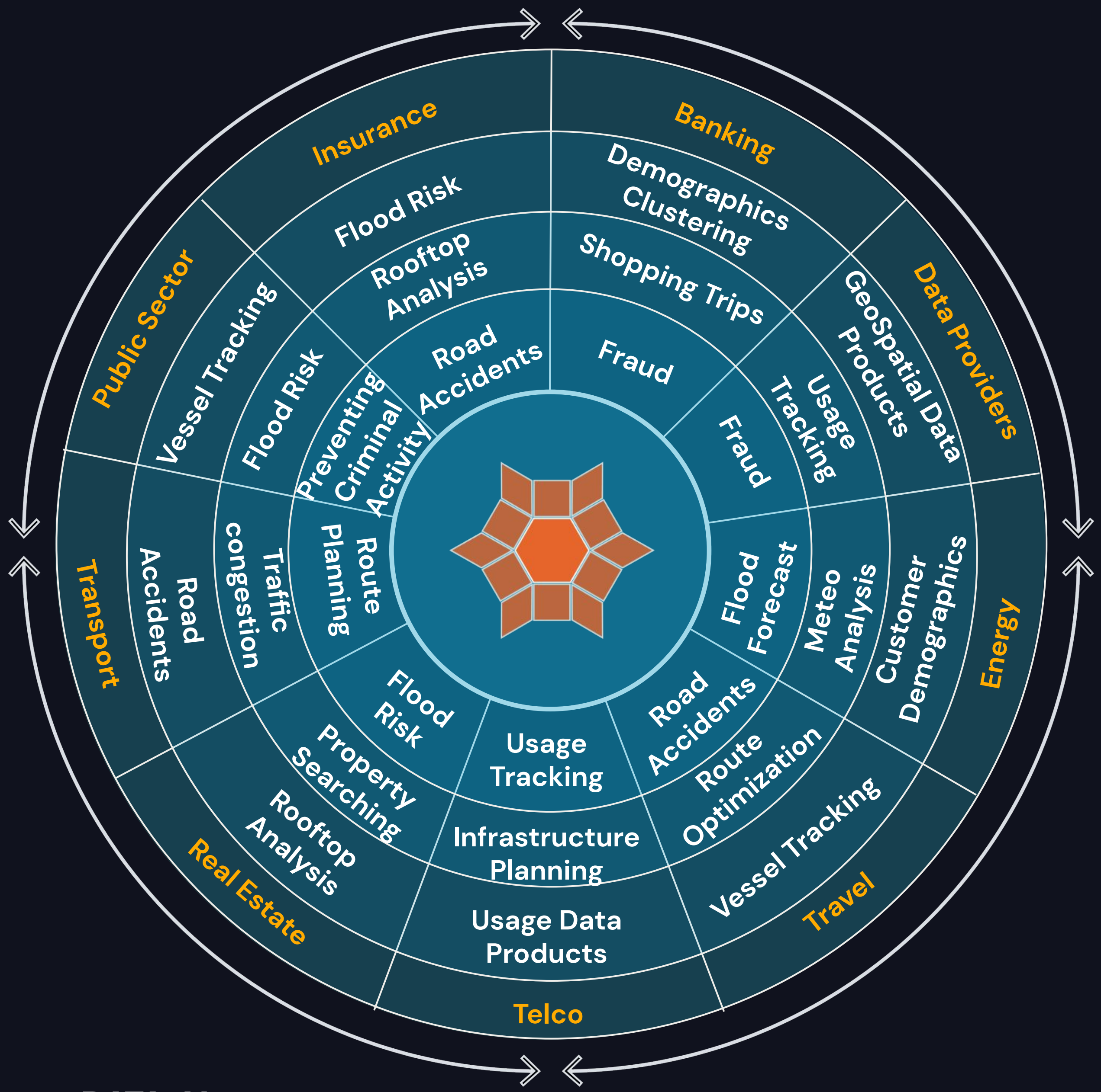
October 11, 2021 in Engineering Blog

Share this post

This is a collaborative post by Ordnance Survey, Microsoft and Databricks. We thank Charis Doidge, Senior Data Engineer, and Steve Kingston, Senior Data Scientist, Ordnance Survey, and Linda Sheard, Cloud Solution Architect for Advanced Analytics and AI at Microsoft, for their contributions.

This blog presents a collaboration between Ordnance Survey (OS), Databricks and Microsoft that explores spatial partitioning using the British National Grid (BNG).





Building a data + AI driven GeoSpatial Ecosystem

Lakehouse Architecture

Structured

Semi-structured

Unstructured

Streaming

DELTA LAKE

Open | Fast | Reliable

Azure Data Lake Storage

Google Cloud Storage

Amazon S3

Data Quality

- Optimized Performance
- Data Consistency and Reliability

Data Lineage

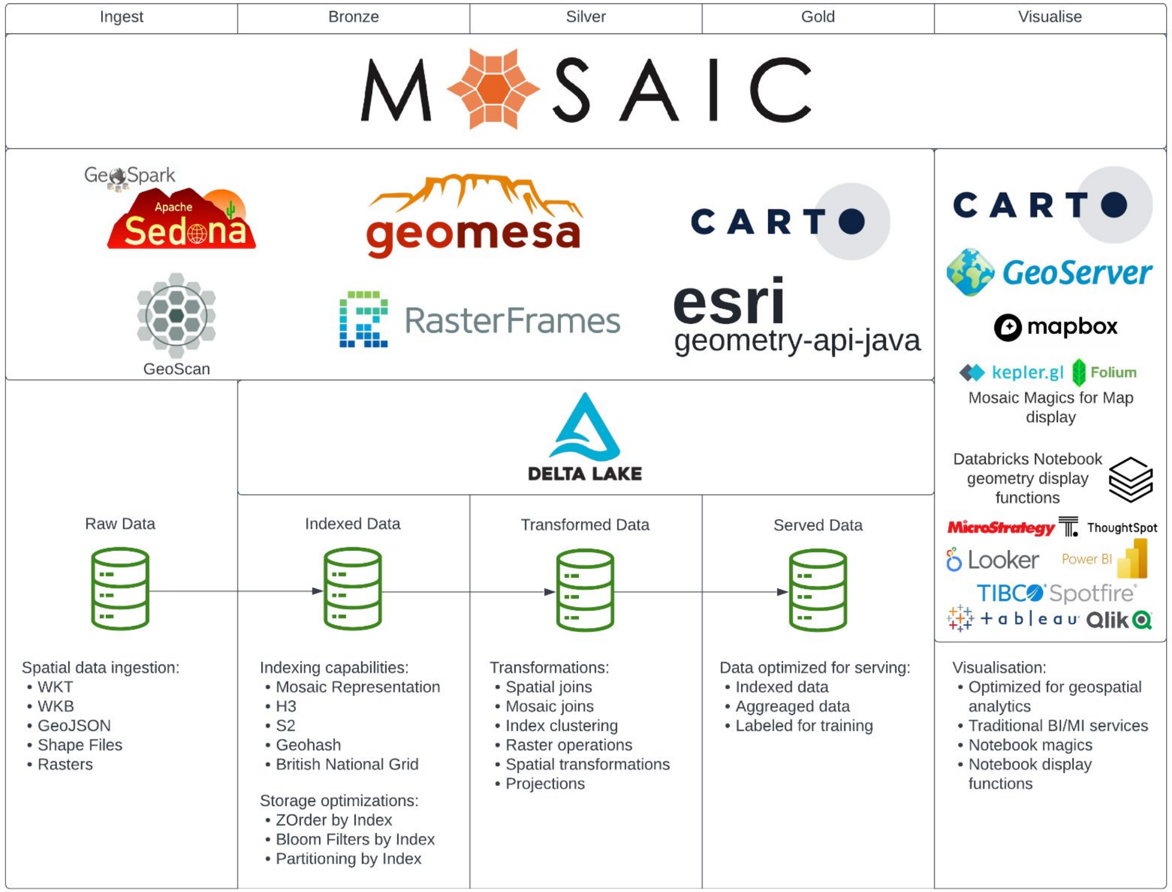
- Data Version Tracking
- Data Governance

Featurization

- Feature Definition, Discovery and Sharing
- Feature Publishing and Drift Monitoring

Mosaic Ecosystem

Easy + fast processing of very large geospatial datasets.



- Uniquely leverages the power of Delta Lake on Databricks
- High performance through implementation of Apache Spark Java code generation
- Flexible for use with other libraries & partners
- Unlock AI/ML + advanced analytics capabilities of geospatial data on top of Databricks Lakehouse

Uber

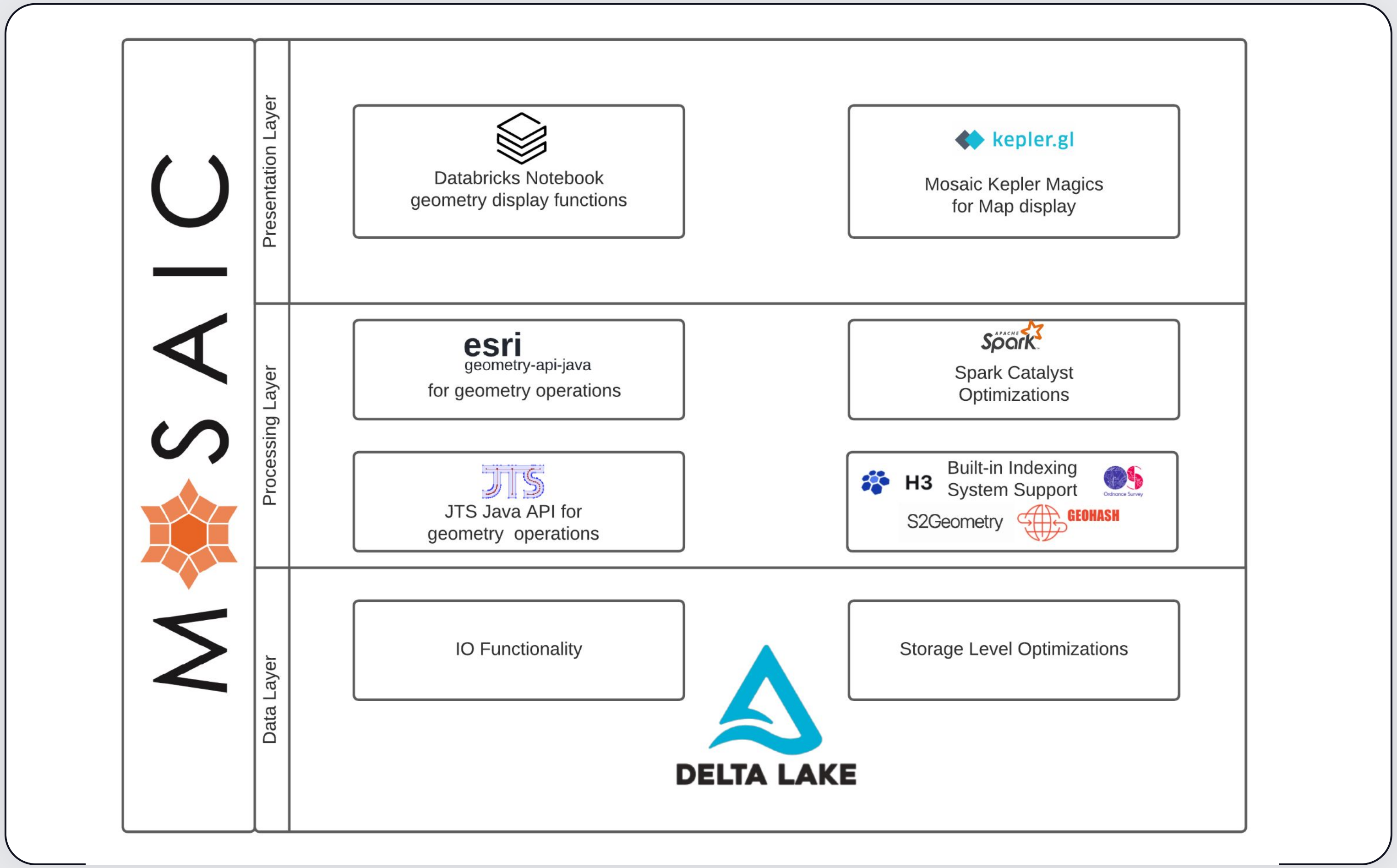
Google

esri
(geometry-api-java)

MOSAIC

LocationTech

Design & implementation



- Spark Expressions for transforming, aggregating, indexing and joining spatial datasets
- Optimizations for performing spatial joins at scale
- Easy conversion between common spatial data encodings such as WKT, WKB and GeoJSON
- A choice of Scala, SQL, Python and R APIs
- Notebook mapping

Colocation of vessels

Unauthorized ship-to-ship transfer of goods

- Large volumes of positional data of vessels gathered from AIS transponders
- Identify if the vessels are near each other at the same point in time
- Applications in preventing criminal activity e.g. illegal exchange of goods or illegal fishing

```
AIS_df = (
  spark
    .read
    .option("badRecordsPath", "/tmp/ais_invalid") #Quarantine
    .csv("/tmp/vessels/2018", header=True, schema=schema)
    .filter("VesselType = 70") # Only select cargos
    .filter("Status IS NOT NULL")
)
display(AIS_df)
```

Table | Data Profile

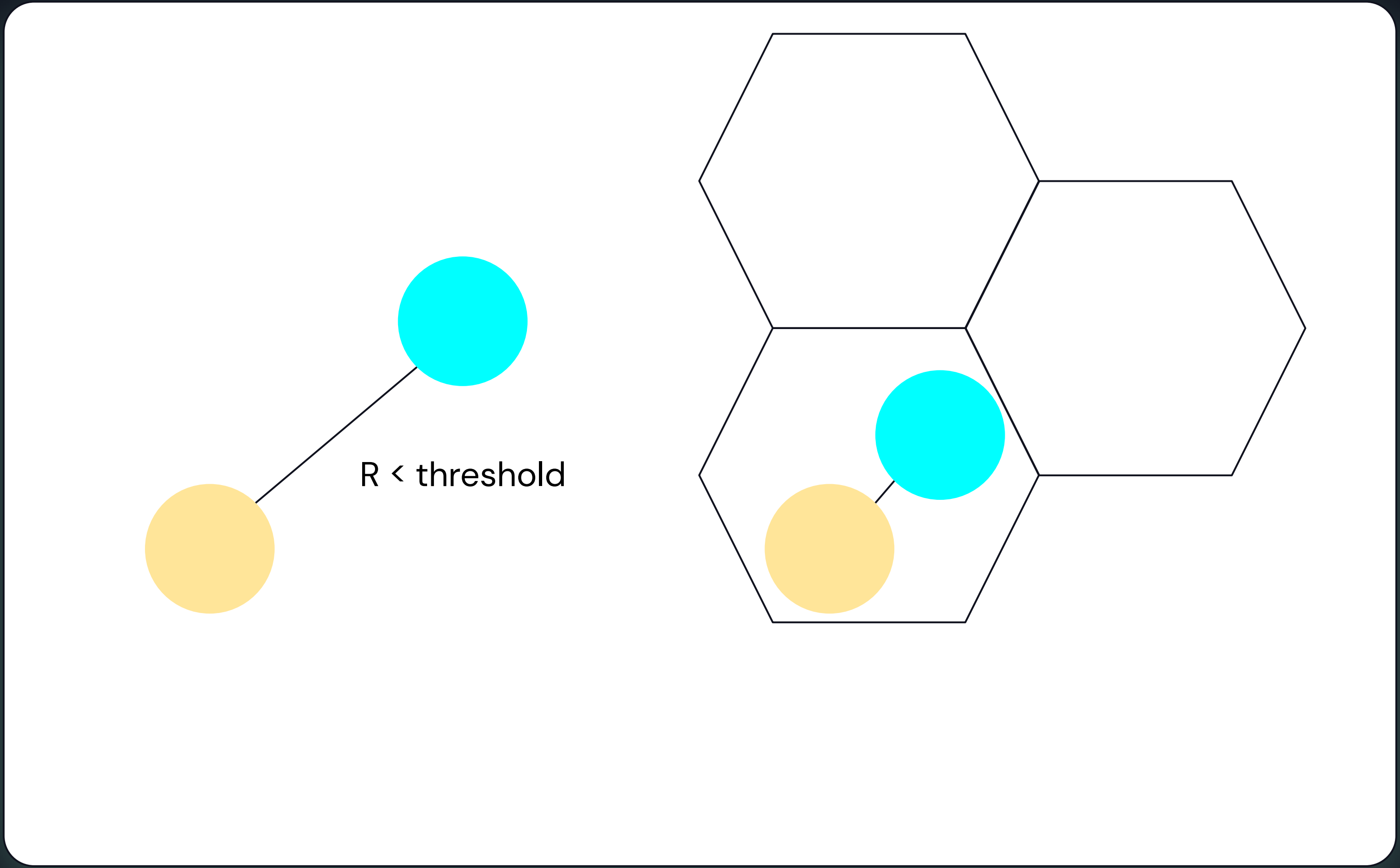
	MMSI	BaseDateTime	LAT	LON	SOG	COG	Heading	VesselName	IMO
1	265491000	2018-01-01T00:00:06.000+0000	39.02249	-76.37648	15.4	162.3	162	MIGNON	IMO9189251
2	316029000	2018-01-01T00:00:00.000+0000	44.71361	-82.79381	11.9	161.4	161	CSL NIAGARA	IMO7128423
3	316001797	2018-01-01T00:00:03.000+0000	43.65205	-82.29167	4.5	-50.6	345	ALGOWAY	IMO7221251
4	370024000	2018-01-01T00:00:01.000+0000	46.19988	-123.42848	13.7	9.4	9	SANTA VISTA	IMO9527946
5	636091883	2018-01-01T00:00:00.000+0000	33.80496	-78.04026	15.2	194.4	195	ZIM COLOMBO	IMO9456977
6	311000310	2018-01-01T00:00:03.000+0000	23.18602	-79.91904	13	110.5	110	BTG EVEREST	IMO9687837

Colocation of vessels

Unauthorized ship-to-ship transfer of goods

Naive approach

Distance Join



- Cross product of all position pairs that occur at the same time
- Very costly
- Skewed on time axis can cause query never to finish
- Use index system (e.g. H3) to avoid endless compute

Naive approach

Long running job that ultimately fails

Cmd 15

```
(
  cargos_indexed.alias("left")
  .crossJoin(cargos_indexed.alias("right"))
  .where(st_distance(col("left.point_geom"), col("right.point_geom")) < buffer)
  .where((col("left.timestamp").cast("long") - col("right.timestamp").cast("long")) > 600)
).count()
```

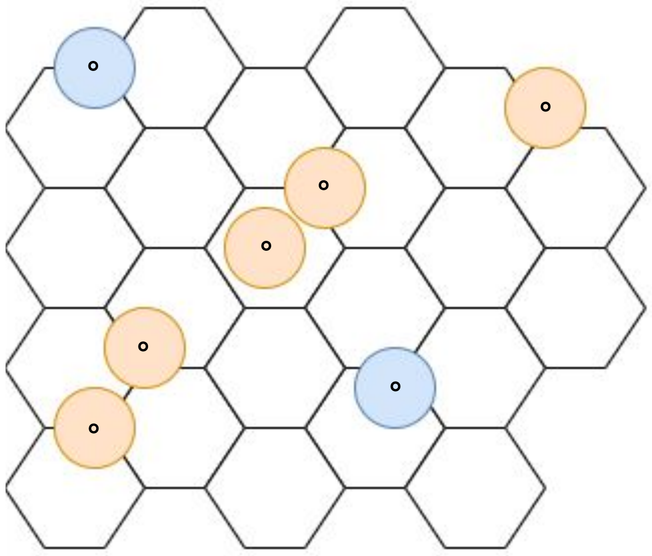
▸ (1) Spark Jobs

```
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 17.0 failed 4 tim
1): ExecutorLostFailure (executor 1 exited caused by one of the running tasks) Reason: worker lost
```

Command took 11.42 hours -- by milos.colic@databricks.com at 27/06/2022, 14:46:32 on MosaicDemo

Buffer approach

Mosaic Polygon Intersection Join



- Represent points as circles by buffering
- Index using Mosaic
- Run polygon intersection join using Mosaic

Buffer approach

Install & enable Mosaic

Cmd 2

```python
%pip install databricks-mosaic
```

Cmd 3

```python
from pyspark.sql.functions import *
from mosaic import *
enable_mosaic(spark, dbutils)
```

Command took 1.33 seconds -- by stuart.lynn@databricks.com at 27/06/2022, 14:40:31 on Shared Autoscaling EMEA

Buffer approach

Buffer positions

Cmd 6

```
one_metre = (0.00001 -      0.000001)
buffer = 100 * one_metre

(
  cargos_indexed
  .withColumn('buffer_geom', st_buffer("point_geom", lit(buffer)))
  .withColumn("ix", mosaic_explode("buffer_geom", lit(9)))
  .write
  .mode('overwrite')
  .saveAsTable('ship2ship.cargos_buffered')
)
```

▸ (4) Spark Jobs

Command took 14.98 minutes -- by milos.colic@databricks.com at 15/06/2022, 07:21:41 on MosaicDemo

Buffer approach

Index vessel data and z-order

Cmd 7

We can optimise our table to colocate data and make querying faster

SQL

```
%sql
OPTIMIZE ship2ship.cargos_buffered ZORDER BY (ix.index_id, timestamp);
SELECT * FROM ship2ship.cargos_buffered;
```

(13) Spark Jobs

Table | Data Profile

	latitude	longitude	sog	heading	status	point_geom	ix
1	40.68374	-74.07344	0	132	5	AQEAAAD4ja89s4RSwMh71cqEV0RA	{"is_core": false, "index_id": "617733150546591743", "wkb": "AQMAAAABAAAAYQAAAPiNrz2zhFLASDMWTWdXREA6FsxGsoF (truncated)"}
2	49.30762	-123.1868	0.1	244	1	AQEAAAAkufyH9MtewDrMlxdgp0hA	{"is_core": false, "index_id": "617712098091991039", "wkb": "AQMAAAABAAAAGwAAACS5/If0v17AuoPYmUKnSEBmQRmR88te

Truncated results, showing first 1000 rows.

Click to re-execute with maximum result limits.

Command took 23.04 seconds -- by milos.colic@databricks.com at 15/06/2022, 07:37:20 on MosaicDemo

Buffer approach

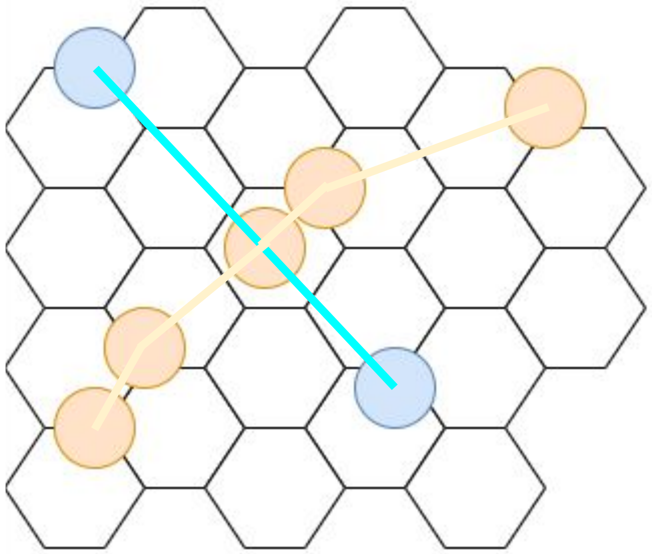
Join on index and only perform intersection if necessary

```
candidates = (
  buffered_events.alias("a")
  .join(
    buffered_events.alias("b"),
    [col("a.ix.index_id") == col("b.ix.index_id"),                           # to only compare across efficient indices
     col('a.mmsi') < col('b.mmsi'),                                          # to prevent comparing candidates bidirectionally
     ts_diff('a.timestamp', 'b.timestamp') <
       time_window("a.sog_kmph", "b.sog_kmph", "a.heading", "b.heading", buffer)
    ]
  )
  .where(
    (col('a.ix.is_core') |  col('b.ix.is_core'))                             # if either candidate fully covers an index, no further comparison is needed
    | st_intersects('a.ix.wkb', 'b.ix.wkb')                                  # limit geospatial querying to cases where indices are not enough
  )
```

```
Command took 31.35 minutes -- by milos.colic@databricks.com at 15/06/2022, 09:31:26 on MosaicDemo
```

Line String approach

Mosaic Polygon Intersection Join



- Connect the points into lines and buffer
- Index line polygons using Mosaic
- Run polygon intersection join using Mosaic

Line String approach

Mosaic Polygon Intersection Join

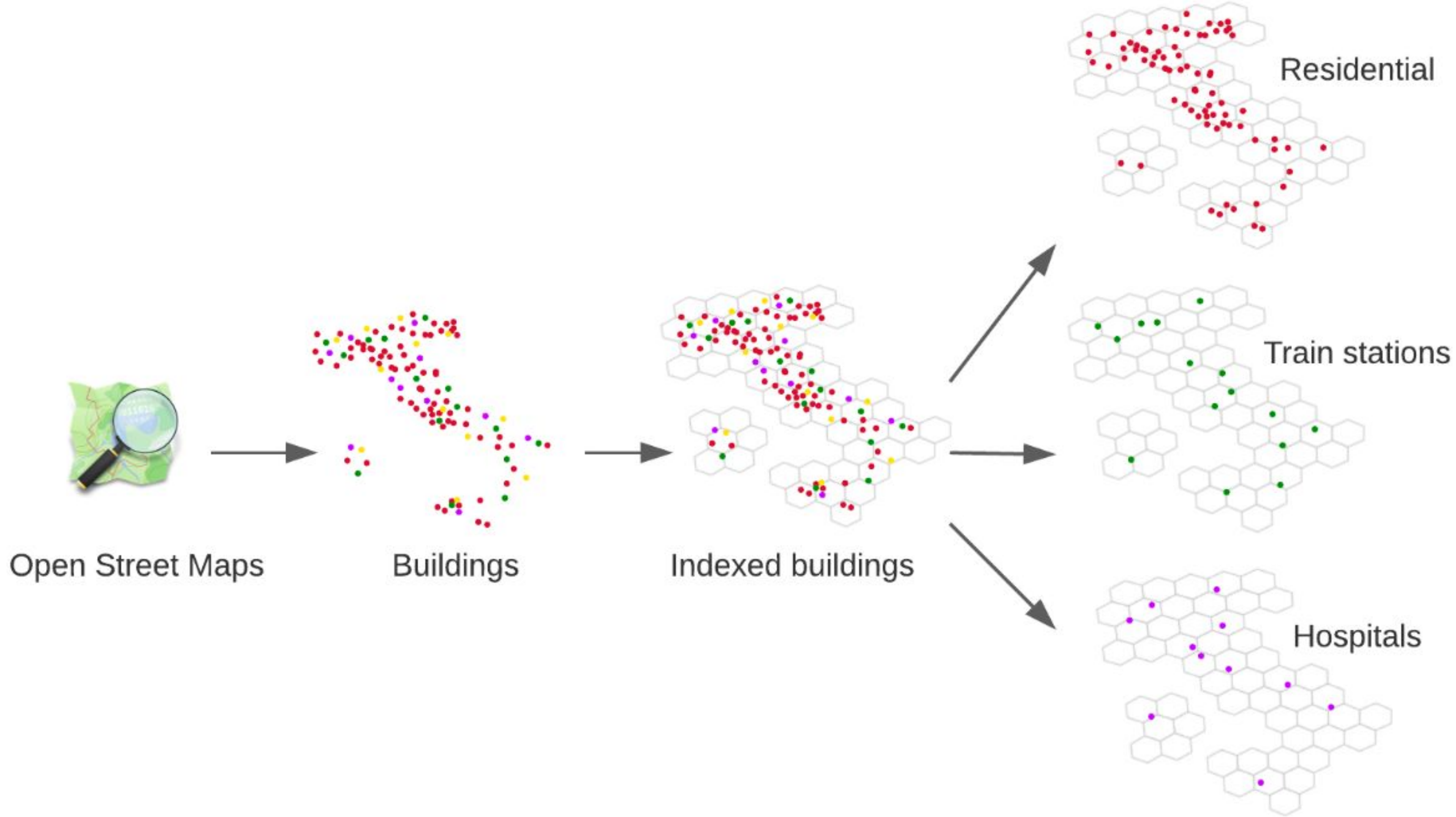
```python
lines = (cargos_indexed
  .repartition(sc.defaultParallelism * 20)
  .groupBy("mmsi", window("timestamp", "15 minutes"))
  .agg(
    collect_list(struct(col("point_geom"), col("timestamp")))
      .alias("coords")
  )
  .withColumn("coords", expr("""
      array_sort(coords, (left, right) ->
        case
          when left.timestamp < right.timestamp then -1
          when left.timestamp > right.timestamp then 1
          else 0
        end
      )"""))
  .withColumn("line", st_makeline(col("coords.point_geom")))
  .cache()
)
```

```python
one_metre = (0.00001 - 0.000001)
buffer = 200 * one_metre

def get_buffer(line):
  np = expr(f"st_numpoints({line})")
  max_np = lines.select(max(np)).collect()[0][0]
  return lit(max_np) * lit(buffer) / np # inverse proportional to number of points, larger buffer for slower ships

cargo_movement = (
  lines
    .withColumn("buffer_r", get_buffer("line"))
    .withColumn("buffer_geom", st_buffer('line', col("buffer_r")))
    .withColumn('buffer', st_astext('buffer_geom'))
    .withColumn('ix', mosaic_explode('buffer_geom', lit(9)))
)
```


```
Command took 6.88 minutes -- by milos.colic@databricks.com at 28/06/2022, 02:46:08 on MosaicDemo
```

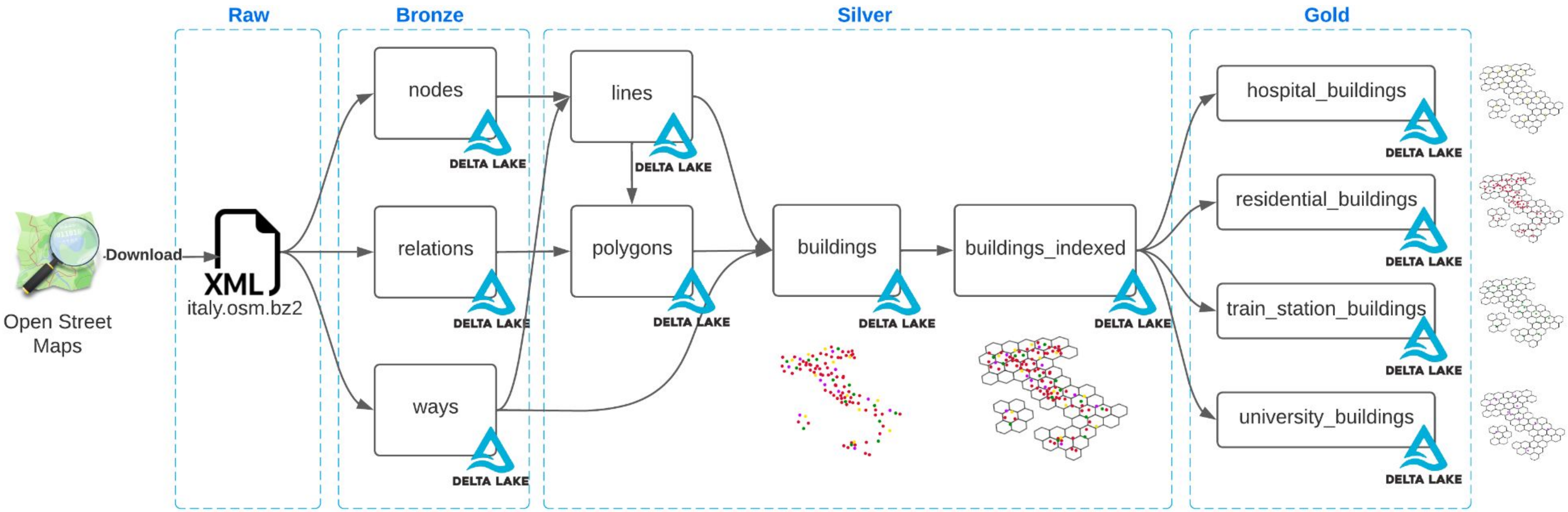
Processing OpenStreetMap data with Live Tables

Creating feature layers from OSM



Processing OpenStreetMap data with Live Tables

Creating feature layers from OSM



Purpose-built for Geospatial Analytics

Solutions
Purpose built solution accelerators that address the highest value use-cases in many industry verticals.

Connectors & Data Models
Connectors to common data sources and solutions for common data models.

Solution Partners
Experienced partners help accelerate solutions with Lakehouse in geospatial analytics domain.

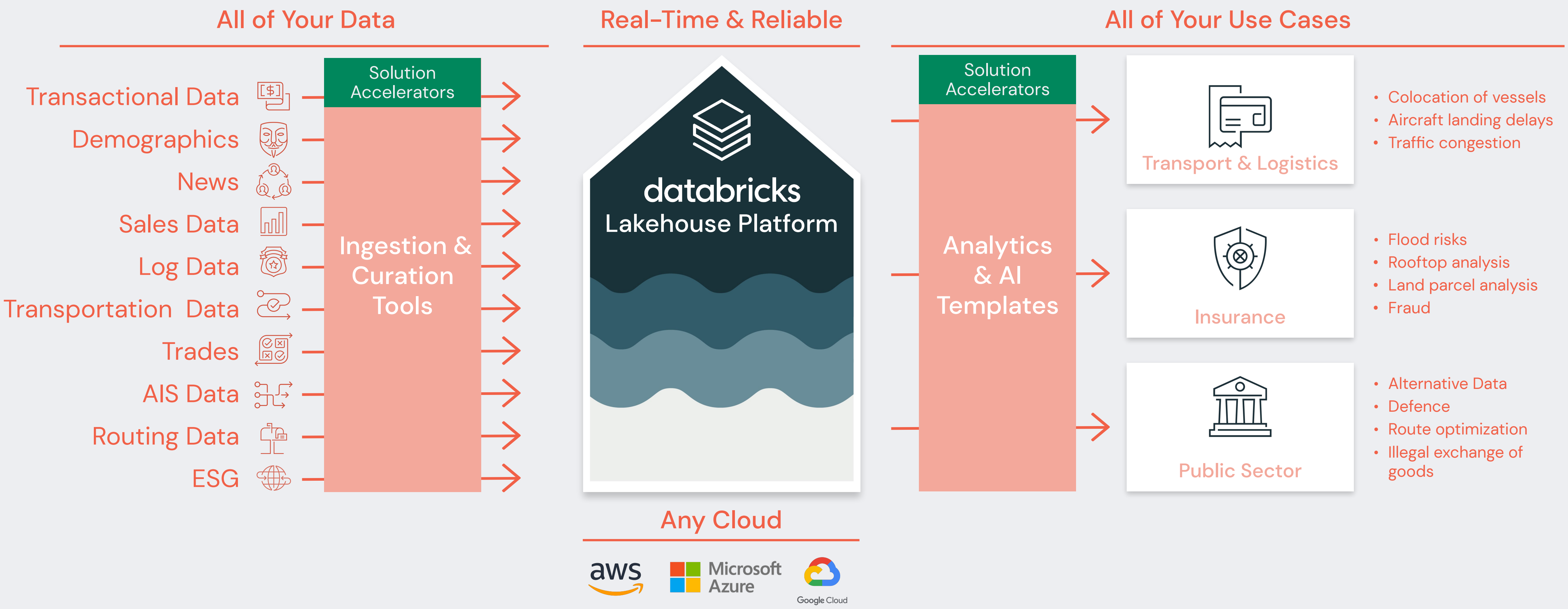
Databricks Product Aligned
Geospatial functionality coming to Databricks product natively and powered by Photon.

Data Sharing & Ecosystem
Consume geospatial data from any data vendors and monetize insights through Delta sharing.

Industry Collaboration
Integrated Lakehouse into multiple geospatial OSS projects in the market today.

Geospatial Analytics on Lakehouse

Helping organizations build a data asset strategy to enable multiple use cases



DATA+AI
SUMMIT 2022

Thank you

Milos Colic
Sr. Solutions Architect,
Databricks

Stuart Lynn
Sr. Solutions Architect,
Databricks

MOSAIC

- Documentation
- GitHub Repo