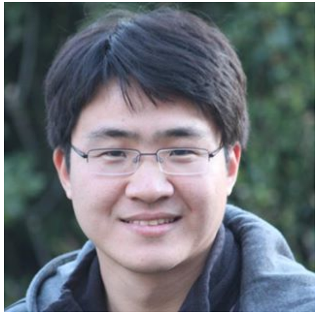
Geospatial support in Apache Pinot

yupeng@uber.com

Uber

About Me



Yupeng Fu (yupeng9@github)

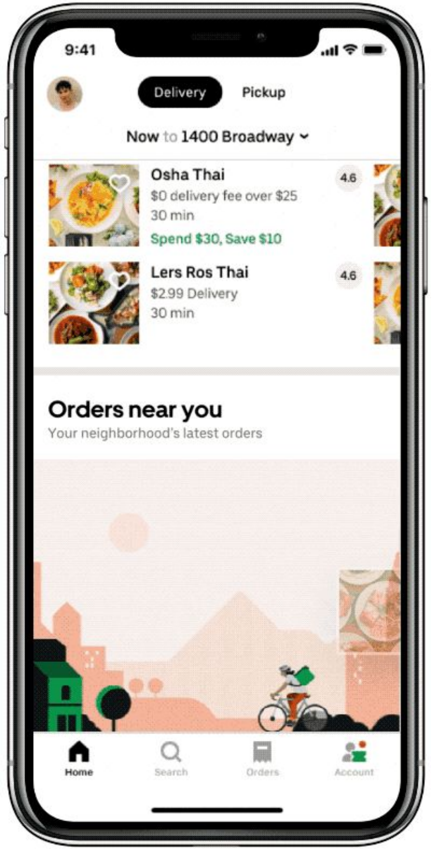
- Principal Engineer @ Uber.Inc
- Real-time Data Platform
- Search Platform
- Committer: Apache Pinot, Alluxio

Why geospatial real-time analytics?

Uber

- Uber's business is highly real-time and geospatial-data related in nature
  - Drivers, riders, restaurants, eaters
  - Trips, cities, routes, locations
- Powerful insights for Uber users
  - What your neighbors are ordering right now?
  - How many drivers/riders in a geolocation?

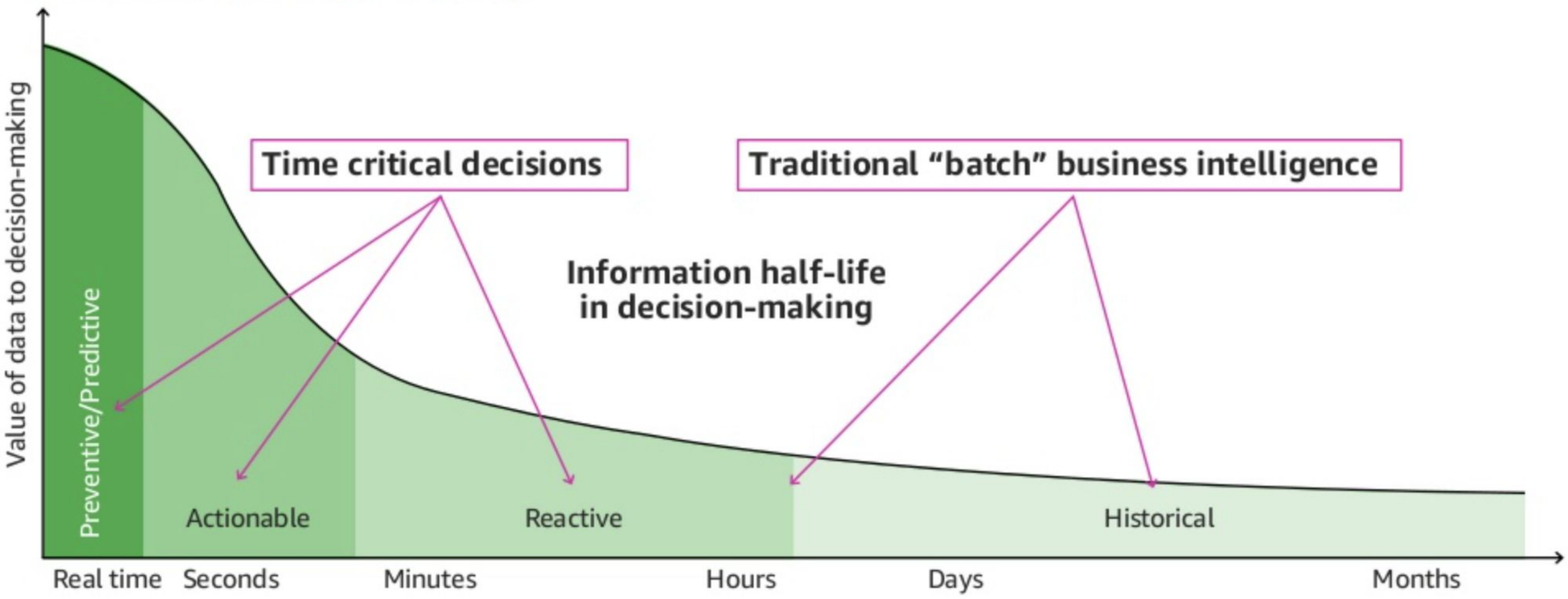
```
SELECT *
FROM   Orders
WHERE  ST_Distance(location_st_point_1,
ST_Point(-90.5, 14.596, 1)) < 16000
AND numberOfItems > 0
AND createdOrderTimestamp > 1612997591
```



Real-time Analytics @Uber

Value of Data over Time

Uber



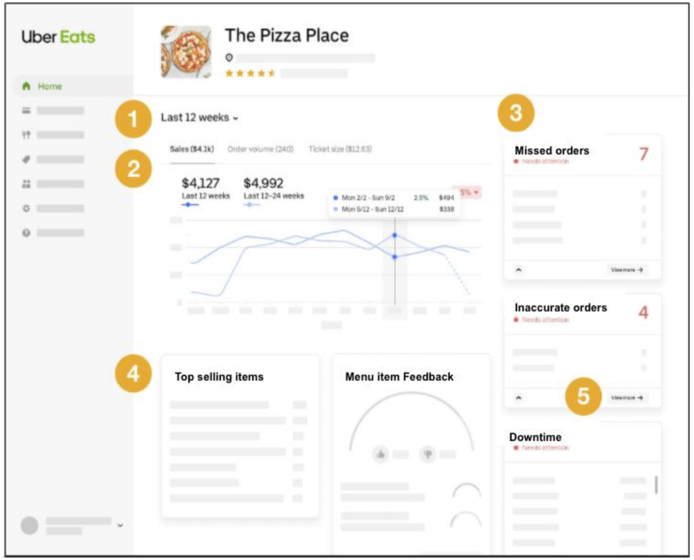
Source: Perishable insights, Mike Gualtieri, Forrester

Uber

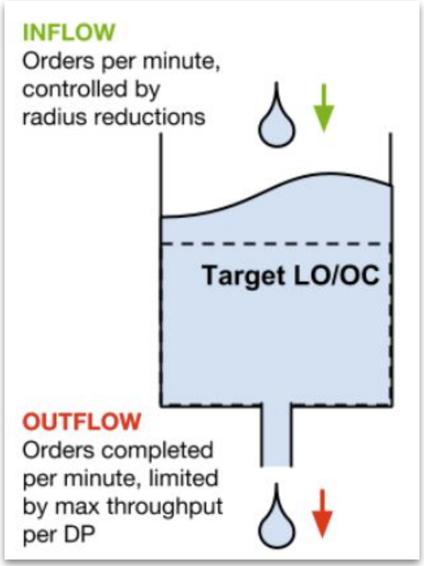
Real-time Analytics in business

1. **Real-time** and **actionable** insights
2. **Time-sensitive** decisions
3. User **engagement** growth

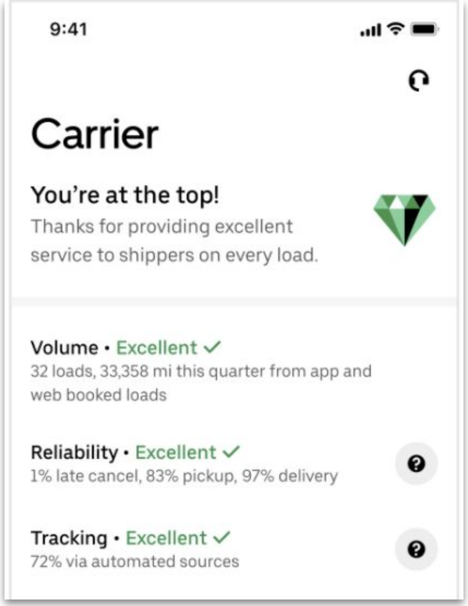
Fast *Access to* ***Fresh*** *Data* ***at Scale***



Restaurant Performance View



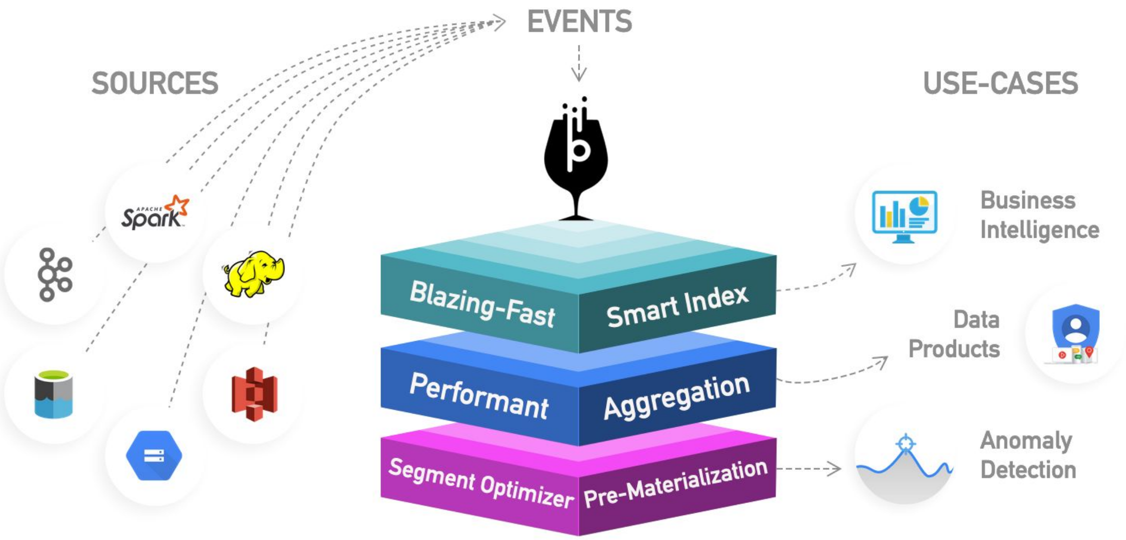
Demand/Supply Management



Freight Carrier Score Card

Apache Pinot: Realtime distributed OLAP datastore

Uber



1M+

Events/sec

170k+

Peak QPS

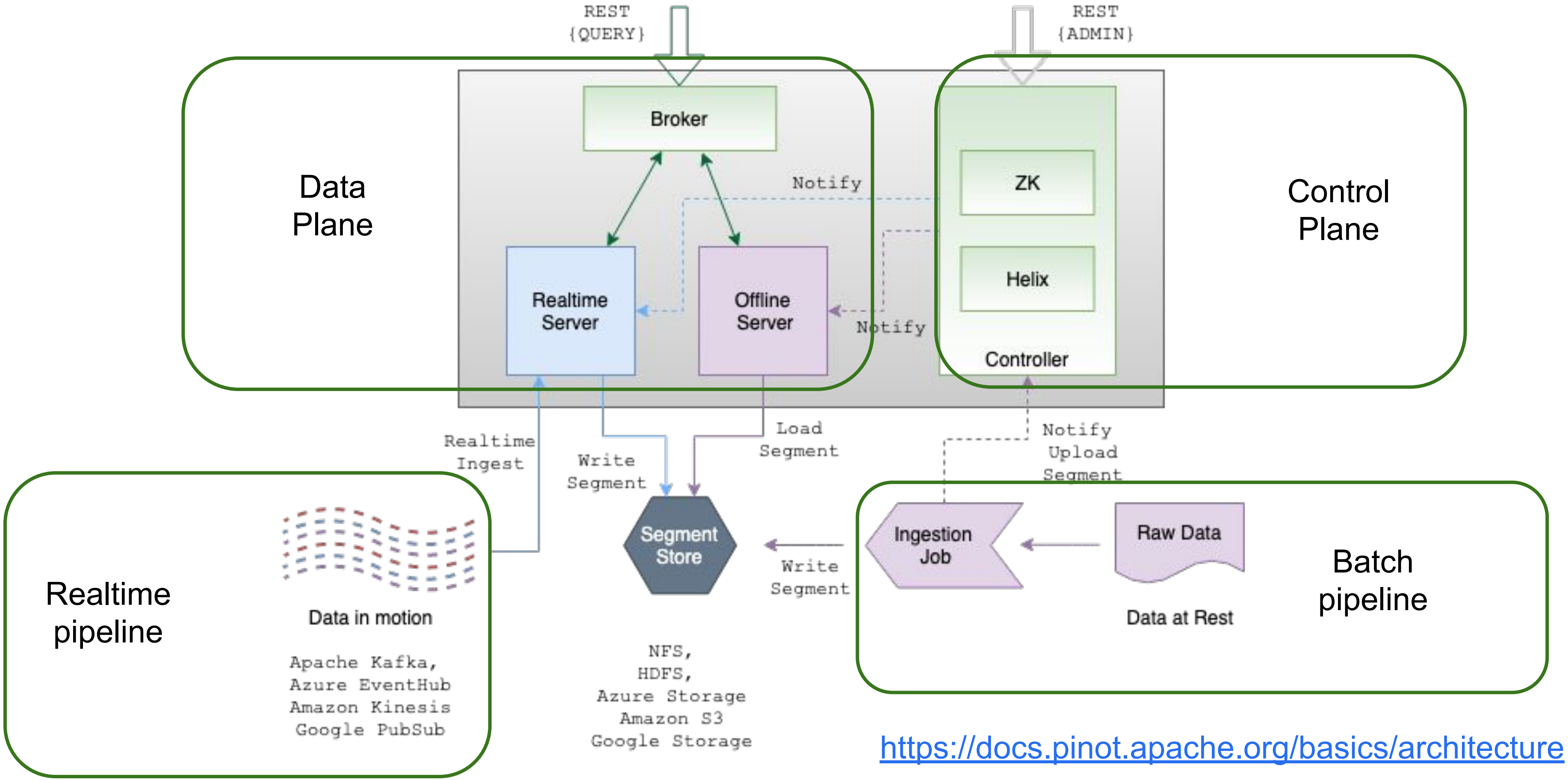
ms

Query Latency

LinkedIn
stripe
UBER
Walmart Save money. Live better.
Microsoft Teams
TARGET
微博 weibo.com
CLOUD KITCHENS
slack
wepay a CHASE company
factual.
confluera
TRACEABLE.

Apache Pinot's High Level Architecture

Uber



https://docs.pinot.apache.org/basics/architecture

Apache Pinot for real-time OLAP

- Chosen for its
  - High QPS, low latency query support
  - Cost effective as compared to others
  - Read more in *Real-time Data Infrastructure at Uber [SIGMOD21]*
- Use cases at Uber
  - User-Facing Analytics (Restaurant Manager, Orders near you)
  - Dashboards
  - Operational Intelligence
  - Financial Intelligence
- Self-onboarding
- Query via Presto connector

99.99% Uptime

Milliseconds latency

Hundreds TBs Data

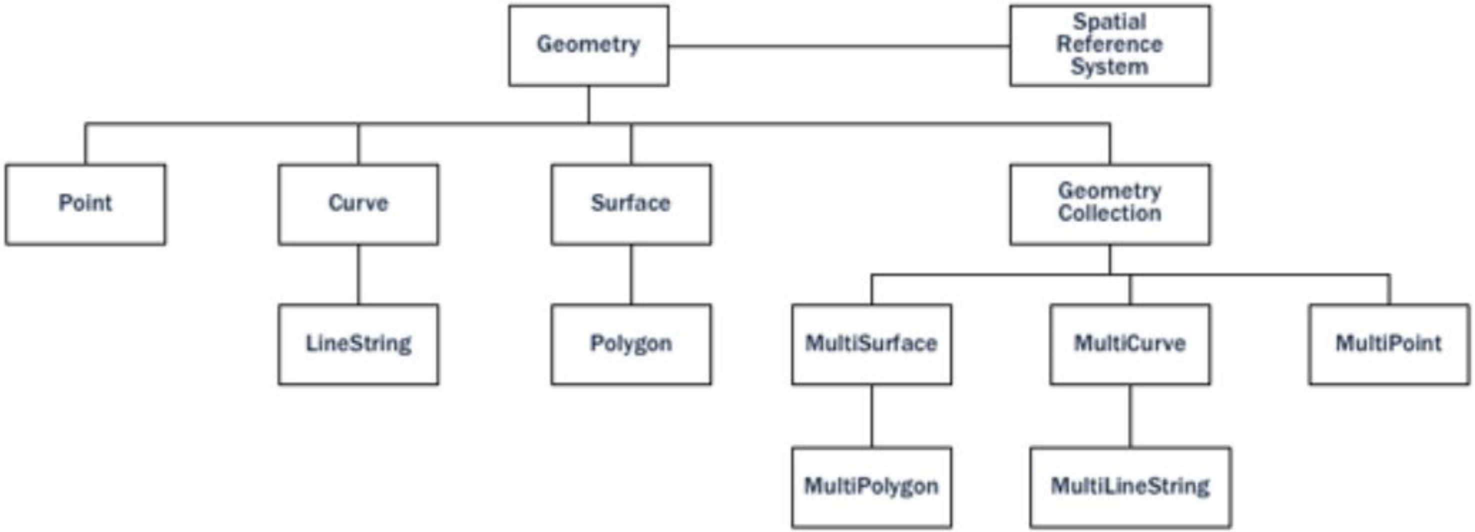
Tens of Thousands QPS

https://arxiv.org/abs/2104.00087

Geospatial Challenges

Challenges - complex data types

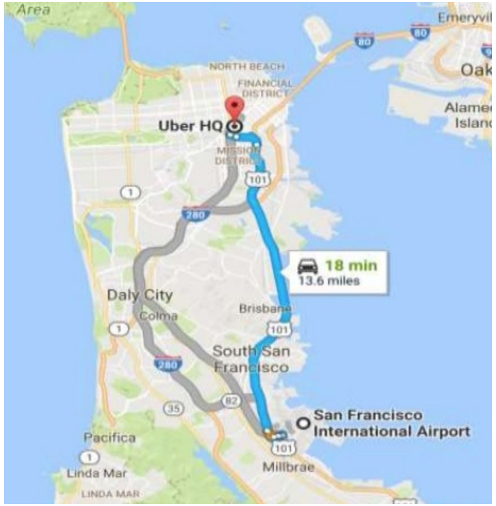
- Geometry hierarchy defined by OGC (Open Geospatial Consortium)

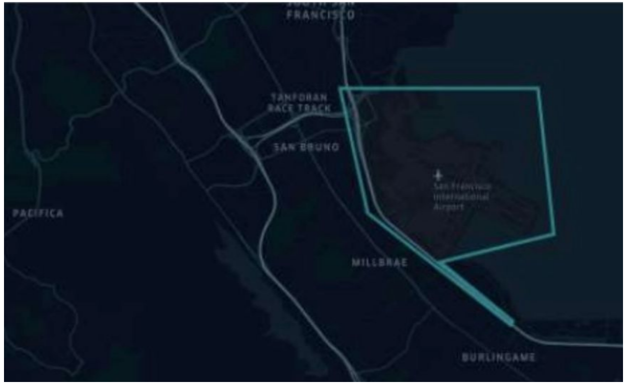


Geospatial data - Primitives

Geometry primitives (2D)

Type	Examples
Point	POINT (30 10)
LineString	LINESTRING (30 10, 10 30, 40 40)
Polygon	POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
	POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10),(20 30, 35 35, 30 20, 20 30))

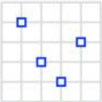




Geospatial data - Multiples

- Multi-* a collection of geometries of the same type

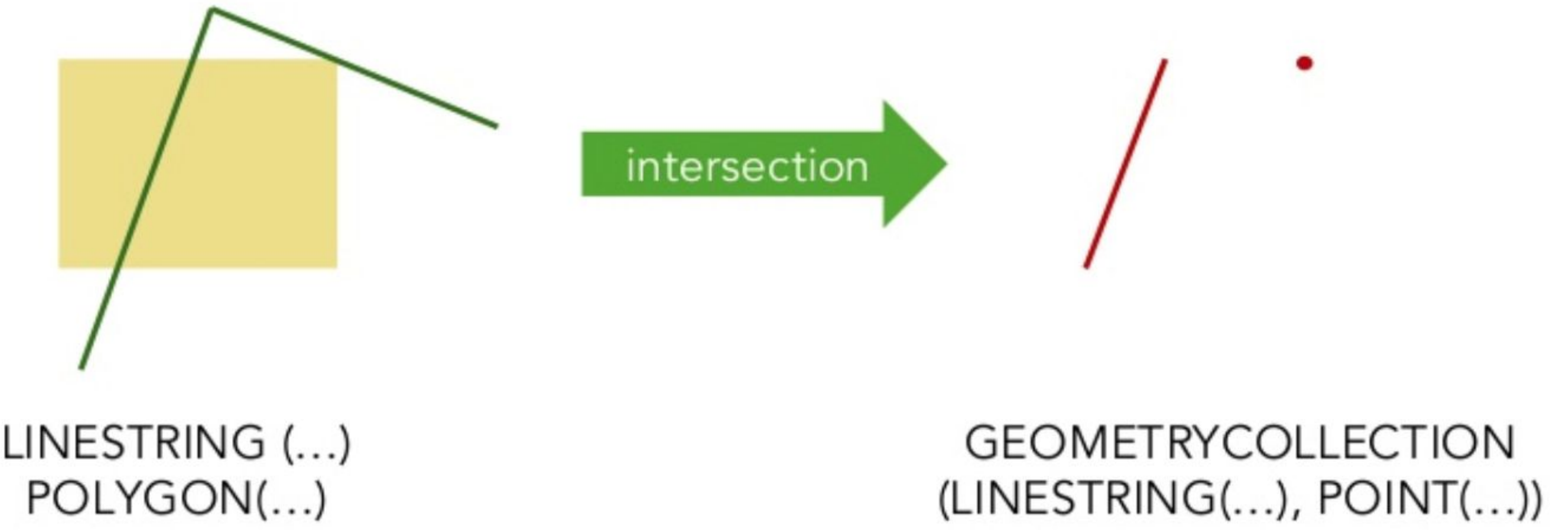
Multipart geometries (2D)

Type	Examples	
MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`
		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`

Geospatial data - Geometry collection

Uber

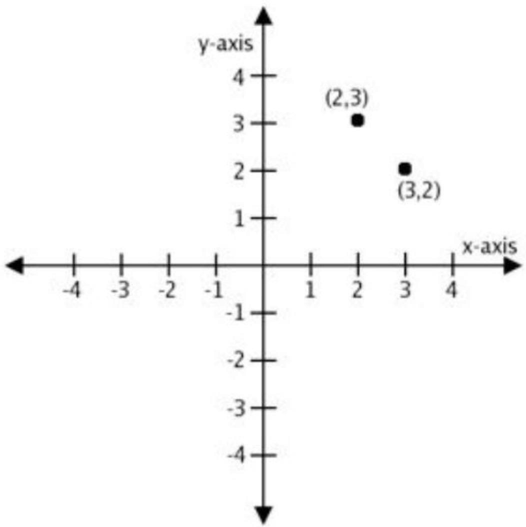
- A collection of geometries of **different** types
- Used to capture the result of an operation,
  - e.g. intersection, difference, etc

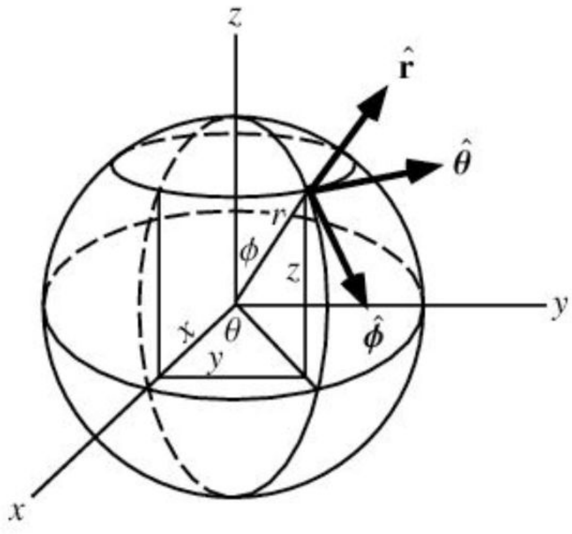


Geometry vs Geography

Uber

- Cartesian: planar coordinates (x,y)
- Spherical: angular coordinates (longitude, latitude)





Geometry vs Geography

Planar coordinate system

Spheric coordinate system

ST_Distance(Vancouver, Paris)

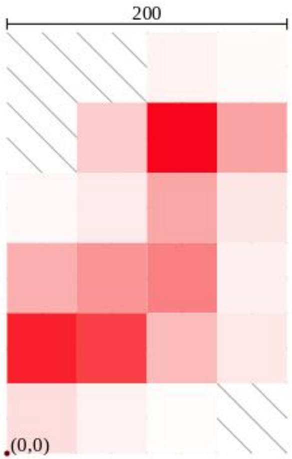
Challenges: many geospatial formats

Uber

- Vector formats such as WKT/WKB, GeoJSON, KML
- Raster formats such as Esri grid, GeoTIFF
- Navigational standards such as AIS and GPS
- OGC web standards such as WCS, WFS

```
GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))
POINT ZM (1 1 5 60)
POINT M (1 1 80)
POINT EMPTY
MULTIPOLYGON EMPTY
TRIANGLE((0 0 0,0 1 0,1 1 0,0 0 0))
TIN (((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)))
POLYHEDRALSURFACE Z ( PATCHES
    ((0 0 0, 0 1 0, 1 1 0, 1 0 0, 0 0 0)),
    ((0 0 0, 0 1 0, 0 1 1, 0 0 1, 0 0 0)),
    ((0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 0 0)),
    ((1 1 1, 1 0 1, 0 0 1, 0 1 1, 1 1 1)),
    ((1 1 1, 1 0 1, 1 0 0, 1 1 0, 1 1 1)),
    ((1 1 1, 1 1 0, 0 1 0, 0 1 1, 1 1 1))
  )
```

Grid image

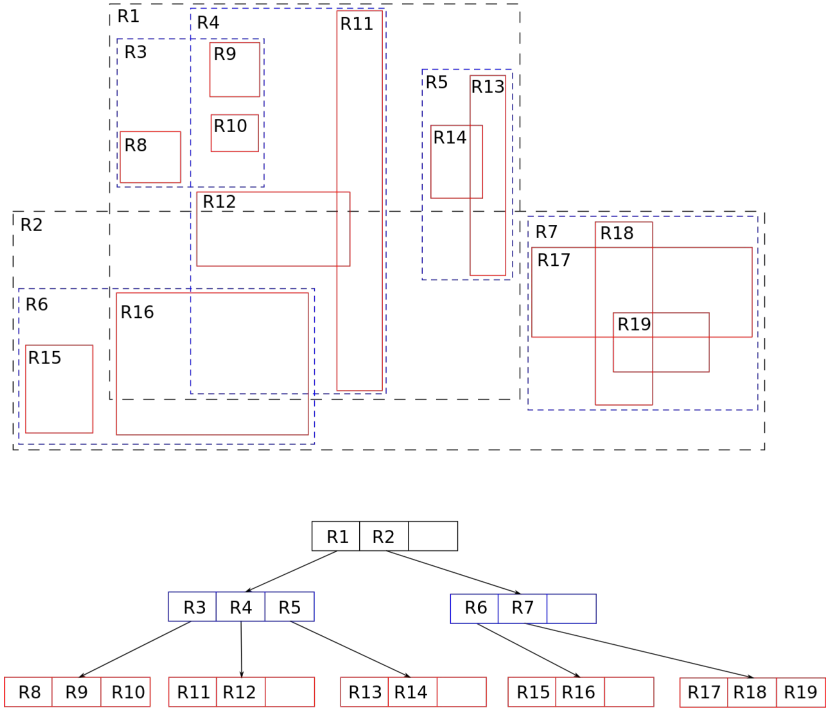


Values

	25	75	125	175
275	NA	NA	5	2
225	NA	20	100	36
175	3	8	35	10
125	32	42	50	6
75	88	75	27	9
25	13	5	1	NA

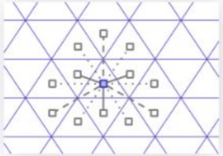
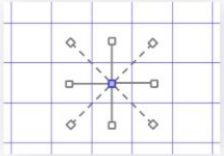
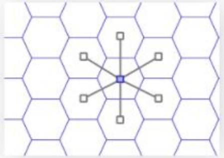
Challenges: spatial indexing

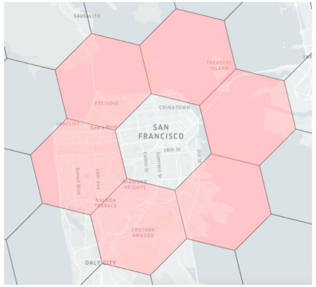
- Efficient records retrieval on large datasets
  - Spatial join
  - ST_Contains, ST_Distance
- Many indexing techniques
  - R-tree
  - Quadtree
  - Geohash
  - Grid (S2, H3)
- Tradeoff between latency and accuracy



Challenges: spatial indexing - H3

- Uber's open-source grid lib
- Hexagon Based
  - 6 neighbors
  - All neighbors are equidistant
- Hierarchical grid system
  - Approximating circles
  - ***NOT*** cleanly subdivide into seven finer hexagons
  - Compact containment

Triangle	Square	Hexagon
		
Triangles have 12 neighbors	Squares have 8 neighbors	Hexagons have 6 neighbors



Geospatial Data in Apache Pinot

Data Types

- Full geometry type hierarchy
- Reuse bytes type
- Geometry: JTS
  - Used by PostGIS/Presto/GeoSpark

- `POINT (0, 0)`
- `LINESTRING (0 0, 1 1, 2 1, 2 2)`
- `POLYGON (0 0, 10 0, 10 10, 0 10, 0 0),(1 1, 1 2, 2 2, 2 1, 1 1)`
- `MULTIPOINT (0 0, 1 2)`
- `MULTILINESTRING ((0 0, 1 1, 1 2), (2 3, 3 2, 5 4))`
- `MULTIPOLYGON (((0 0, 4 0, 4 4, 0 4, 0 0), (1 1, 2 1, 2 2, 1 2, 1 1)), ((-1 -1, -1 -2, -2 -2, -2 -1, -1 -1)))`
- `GEOMETRYCOLLECTION(POINT(2 0),POLYGON((0 0, 1 0, 1 1, 0 1, 0 0)))`

https://docs.pinot.apache.org/basics/indexing/geospatial-support#geospatial-data-types

Geo data serialization/conversion

- Needed for storage and function eval
- Options
  - Well-Known Text (WKT)
  - Well-known Binary (WKB)
- Support both Geometry vs Geography
  - Converted via functions
  - Some functions can only be applied on geometry or geography

Geo data ingestion

- Data transformation needed during the data ingestion
- Use *transformFunction* to store native geodata
- A set of built-in transform functions

```
{
  "dimensionFieldSpecs": [
    {
      "dataType": "STRING",
      "name": "event_name"
    },
...
    {
      "dataType": "DOUBLE",
      "name":"lat"
    },
    {
      "dataType": "DOUBLE",
      "name":"lon"
    },
    {
      "dataType": "BYTES",
      "name":"location",
      "transformFunction":
            "toSphericalGeography(stPoint(lon,lat))"
    }

  ]
  "schemaName": "meetupRsvp"
}
```

Geospatial Functions in Apache Pinot

Geospatial functions

- ISO Standard - SQL/MM Part 3
- ST_ prefix (S - spatial, T - temporal)
- Simple Feature Access - SQL

  https://www.ogc.org/standards/sfs/

Geospatial functions

- Constructors
  - e.g. ST_GeomFromText, ST_Point
- Measurements
  - e.g. ST_Area, ST_Distance
- Outputs
  - e.g. ST_AsBinary, ST_AsText
- Relationship
  - e.g. ST_Contains, ST_Equals
- Aggregations
  - e.g. ST_Union

Measurements

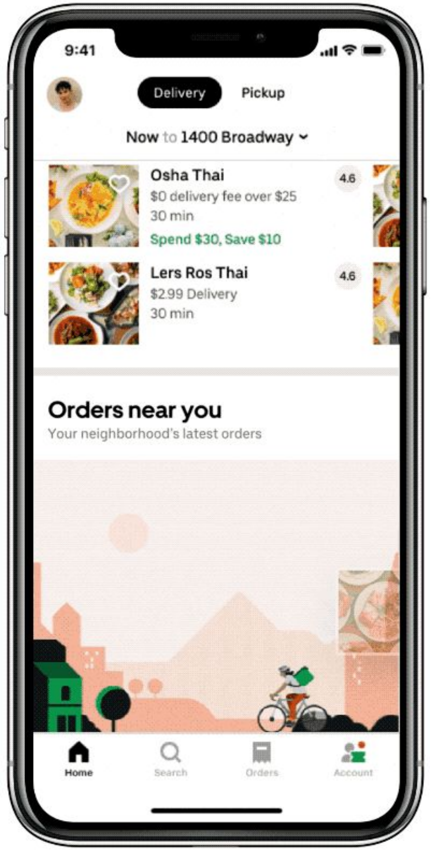
- **ST_Area(Geometry/Geography g) → double** For geometry type, it returns the 2D Euclidean area of a geometry. For geography, returns the area of a polygon or multi-polygon in square meters using a spherical model for Earth.
- **ST_Distance(Geometry/Geography g1, Geometry/Geography g2) → double** For geometry type, returns the 2-dimensional cartesian minimum distance (based on spatial ref) between two geometries in projected units. For geography, returns the great-circle distance in meters between two SphericalGeography points. Note that g1, g2 shall have the same type.
- **ST_GeometryType(Geometry g) → String** Returns the type of the geometry as a string. e.g.: `ST_Linestring`, `ST_Polygon`, `ST_MultiPolygon` etc.

https://docs.pinot.apache.org/basics/indexing/geospatial-support#geospatial-functions

Orders near you

Uber

```
SELECT *
FROM   Orders
WHERE  ST_Distance(location_st_point_1,
ST_Point(-90.5, 14.596, 1)) < 16000
AND numberOfItems > 0
AND createdOrderTimestamp > 1612997591
```

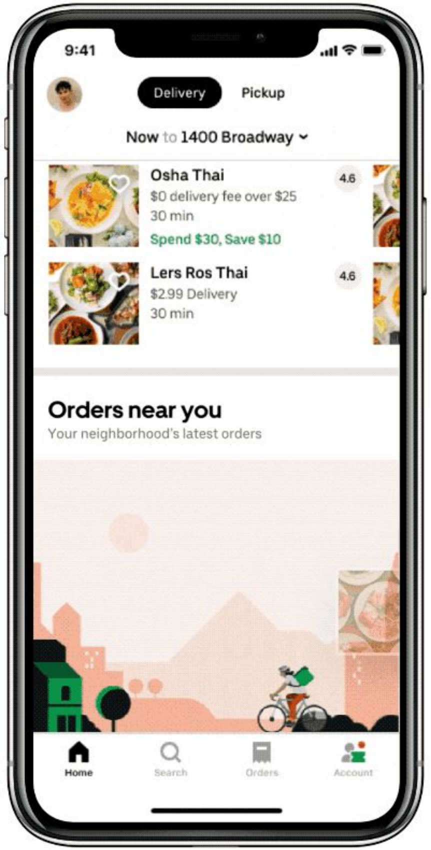


Geospatial indexing in Apache Pinot

Orders near you

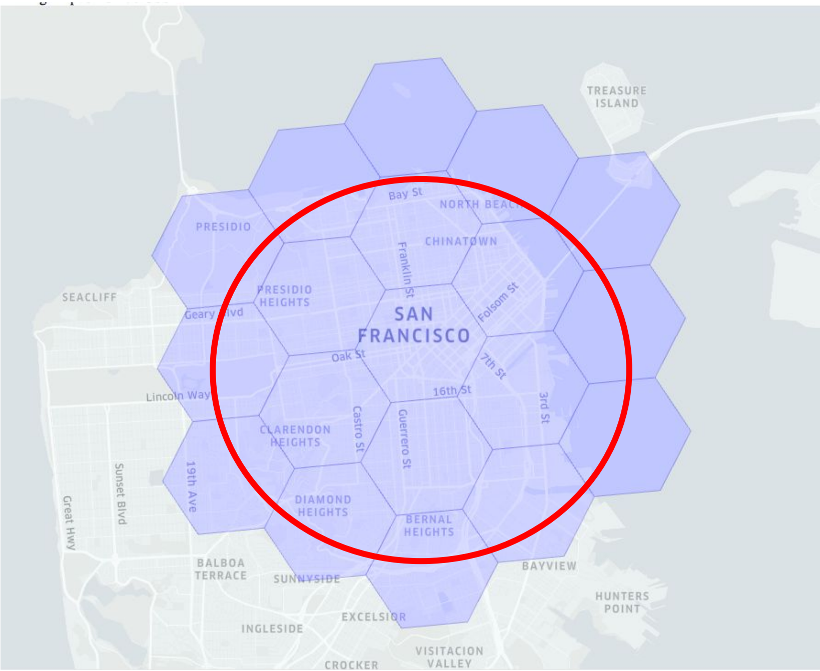
Uber

```sql
SELECT *
FROM   Orders
WHERE  ST_Distance(location_st_point_1,
ST_Point(-90.5, 14.596, 1)) < 16000
AND numberOfItems > 0
AND createdOrderTimestamp > 1612997591
```



Query acceleration with index

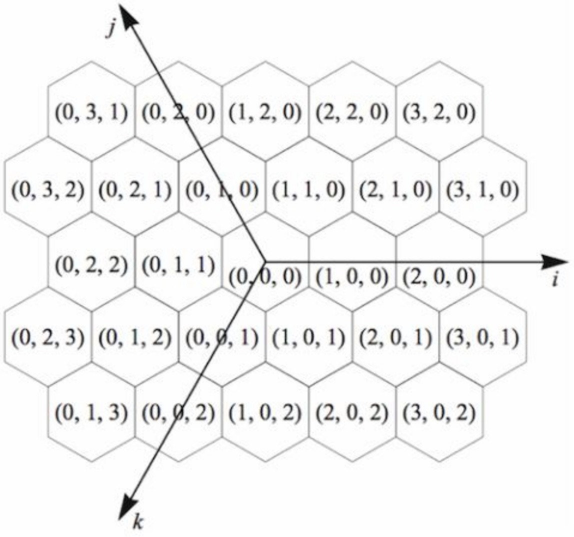
Uber




```
SELECT *
FROM   Orders
WHERE  ST_Distance(location_st_point_1,
ST_Point(-90.5, 14.596, 1)) < 16000
AND numberOfItems > 0
AND createdOrderTimestamp > 1612997591
```

Geo indexing with H3

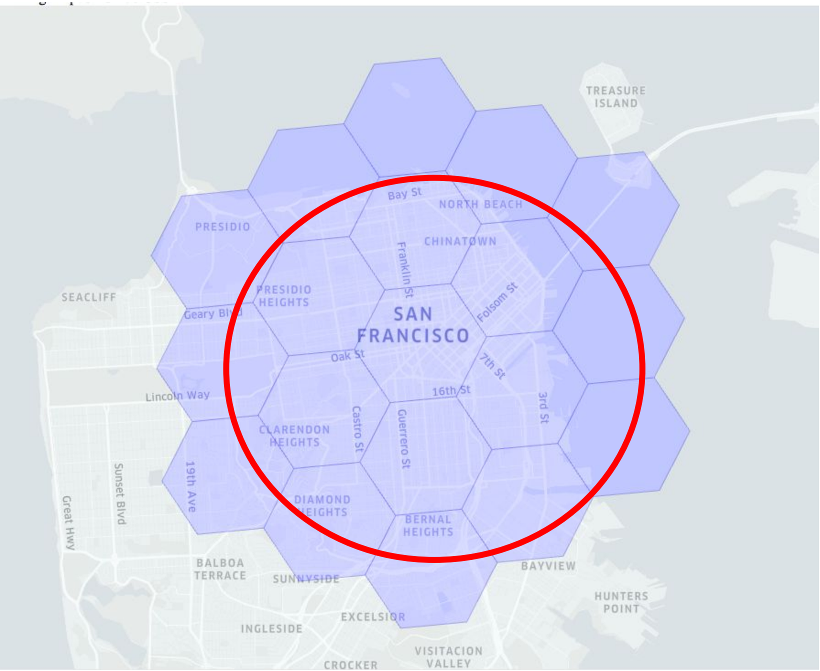
- Indexes the location at the specified resolution
  - `H3Index geoToH3(const GeoCoord *g, int res);`
- Finds the boundary of the index
  - `void h3ToGeoBoundary(H3Index h3, GeoBoundary *gp);`
- Find the indices within k distance of the origin index
  - `void kRing(H3Index origin, int k, H3Index* out);`



Geo indexing with H3 - algorithm

Uber

- Find the H3 distance x that contains the range
- Points within the H3 distance (i.e. covered by the hexagons within kRing(x)),) can be directly taken without filtering
- Points falling into the H3 distance, are filtered by evaluating the condition *ST_Distance(loc1, loc2) < x*



Geo indexing creation

Uber

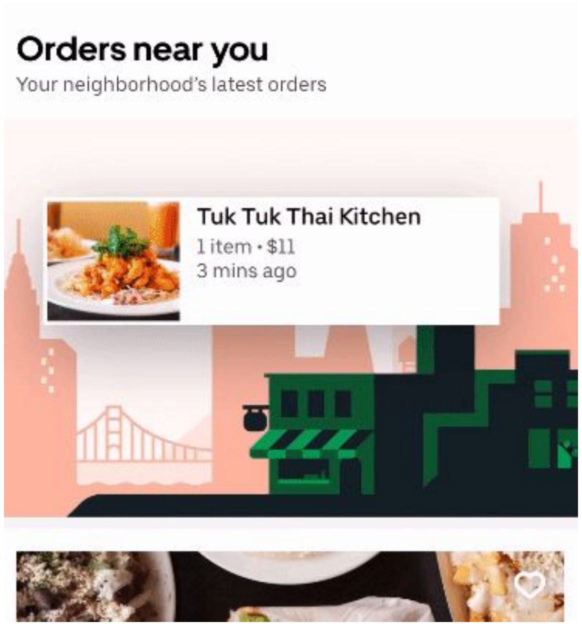
```
{
  "dimensionFieldSpecs": [
    {
      "dataType": "STRING",
      "name": "event_name"
    },
...
    {
      "dataType": "DOUBLE",
      "name":"group_lat"
    },
    {
      "dataType": "DOUBLE",
      "name":"group_lon"
    },
    {
      "dataType": "BYTES",
      "name":"group_location",
      "transformFunction":
            "toSphericalGeography(stPoint(lon,lat))"
    }

  ]
  "schemaName": "meetupRsvp"
}
```

```
geoindex tableConfig
{
  "fieldConfigList": [
  {
    "name": "location_st_point",
    "encodingType":"RAW",
    "indexType":"H3",
    "properties": {
    "resolutions": "5"
     }
    }
  ],
  "tableIndexConfig": {
    "loadMode": "MMAP",
    "noDictionaryColumns": [
      "location_st_point"
    ]
  },
}
```

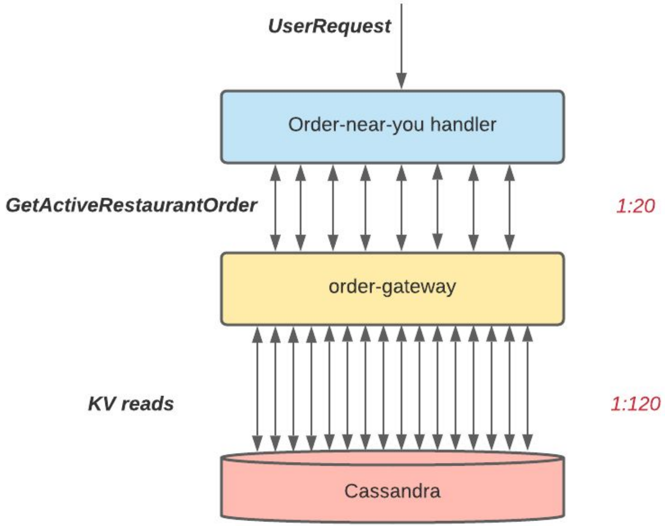
https://docs.pinot.apache.org/basics/indexing/geospatial-support#geospatial-index

Insights gained from storage choice for user-facing analytics

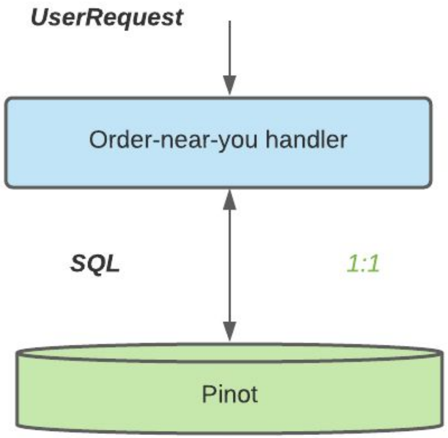
Orders near you - storage challenge



- First Launched in Oct Using Cassandra
- 3K QPS -> 360K Cassandra reads
- 6x increase in Cassandra capacity needed

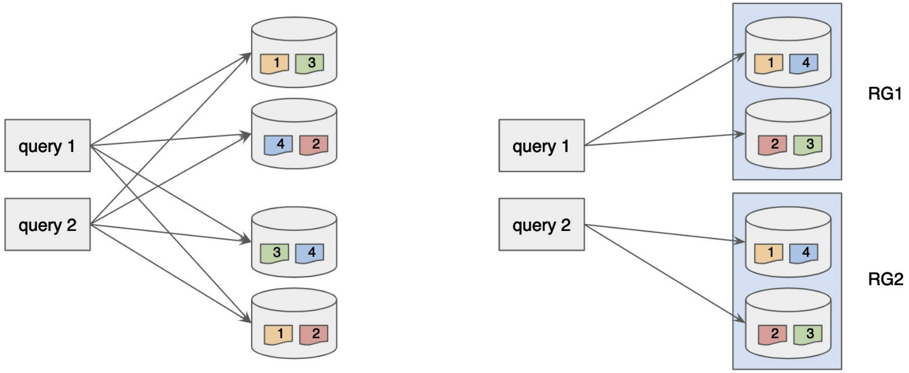


Orders near you - solution with Pinot



- 1:1 Query between Mobile and Pinot. 10 servers support all of Eats Load (3K QPS)
- P95 at 50ms vs 2 secs, Reduced Latency by *10X*

Orders near you - horizontal scaling with Pinot



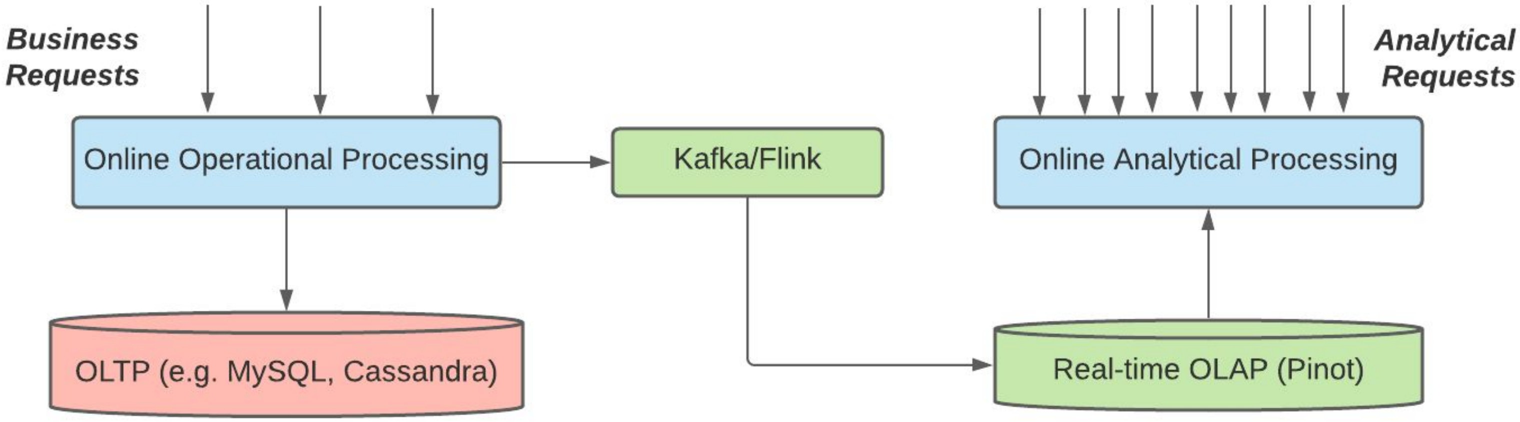

```
...
    "routing": {
      "instanceSelectorType": "replicaGroup"
    },
    "instanceAssignmentConfigMap": {
      "CONSUMING": {
        "tagPoolConfig": {
          "tag": "realtimeTenant_REALTIME",
          "poolBased": false,
          "numPools": 0
        },
        "replicaGroupPartitionConfig": {
          "replicaGroupBased": true,
          "numInstances": 0,
          "numReplicaGroups": 16,
          "numInstancesPerReplicaGroup": 2,
          "numPartitions": 0,
          "numInstancesPerPartition": 0
        }
      }
    },
...
```

https://docs.pinot.apache.org/operators/operating-pinot/tuning/routing

Insights gained

Separation of Operational Database (OLTP) vs Online Analytical Database (OLAP)

- Better Reliability
- Higher Developer Productivity (a few weeks to launch) & self-serve
- Better Query Latency
- Better Cost Efficiency



Useful links

- Release notes
  https://docs.pinot.apache.org/basics/releases/0.7.1
- User guide
  https://docs.pinot.apache.org/basics/indexing/geospatial-support
- Design doc
  https://docs.google.com/document/d/1Mkm5RHS_tof-vIUt5-UNeOgRYSBAN6M_pN-hedV6Q0g
- Introduction blog
  https://medium.com/apache-pinot-developer-blog/introduction-to-geospatial-queries-in-apache-pinot-b63e2362e2a9
- Uber Engineering blog
  https://eng.uber.com/orders-near-you/

Q&A