Towards a Well-Known Binary Format for Moving Features

Contact: Esteban Zimányi (ezimanyi@ulb.ac.be)
Mahmoud Sakr (mahmoud.sakr@ulb.ac.be)

MobilityDB: Architecture

MobilityDB

PostGIS

PostgreSQL

tgeompoint, tgeogpoint,
tint, tfloat, ttext, tbool, ...

geometry, geography

numeric, monetary, character,
data/time, boolean, enum,
arrays, range,
XML, JSON, ...

PostGIS Input/Output Formats

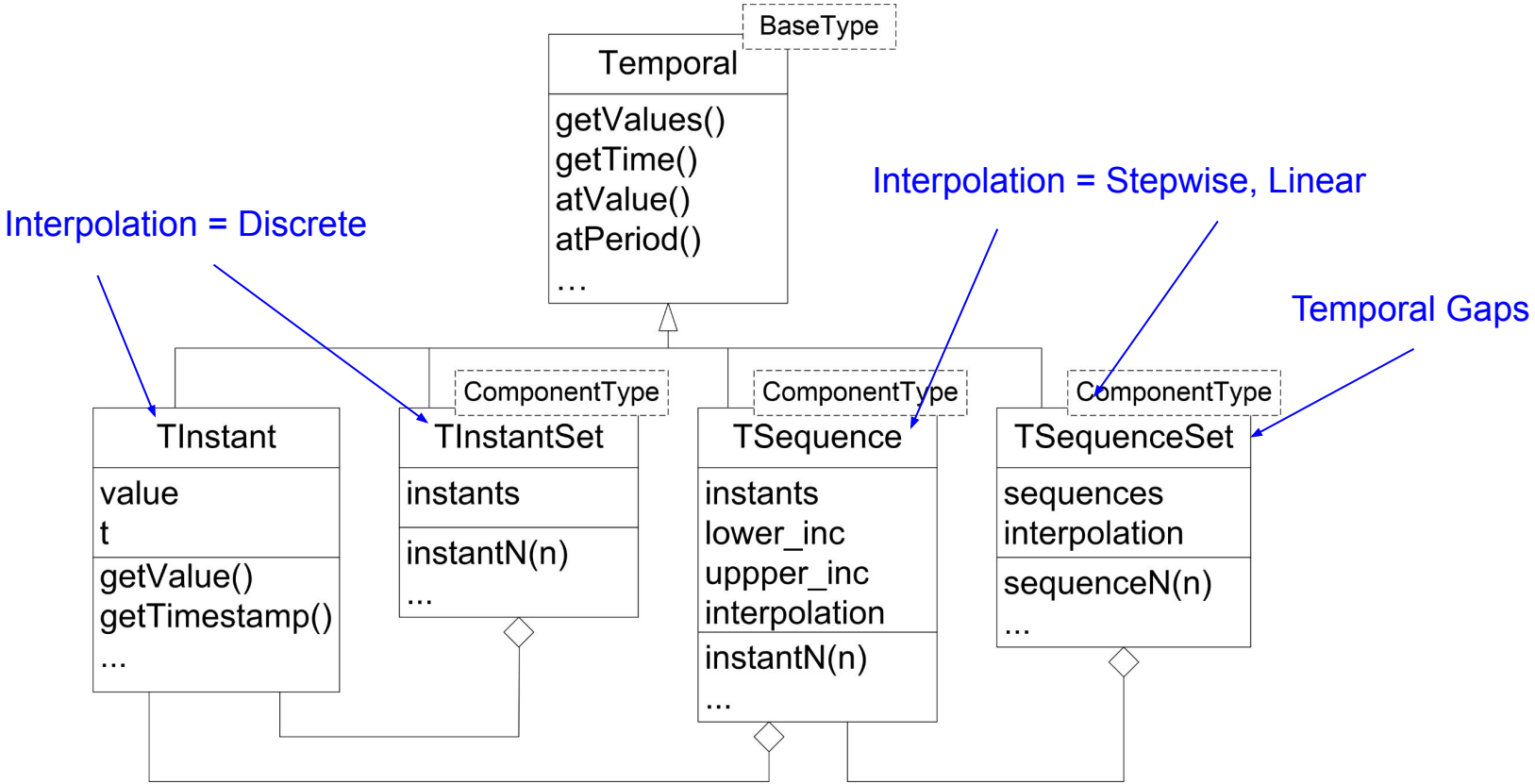
- (Extended) Well-Known Text, (Extended) Well-Known Binary, GeoJSON, ...

```
SELECT ST_AsEWKT(geometry 'SRID=5676;Linestring(1 1,2 2)');
    SRID=5676;LINESTRING(1 1,2 2)
SELECT ST_AsEWKB(geometry 'SRID=5676;Linestring(1 1,2 2)');
    \x01020000202c160000020000000000000000f03f000000000000f03f0...
SELECT ST_AsGeoJSON(geometry 'SRID=5676;Linestring(1 1,2 2)');
    {"type":"LineString","coordinates":[[1,1],[2,2]]}
SELECT ST_AsGeoJSON(geometry 'SRID=5676;Linestring(1 1,2 2)',options:=2);
    {"type":"LineString","crs":{"type":"name",
    "properties":{"name":"EPSG:5676"}},"coordinates":[[1,1],[2,2]]}
```

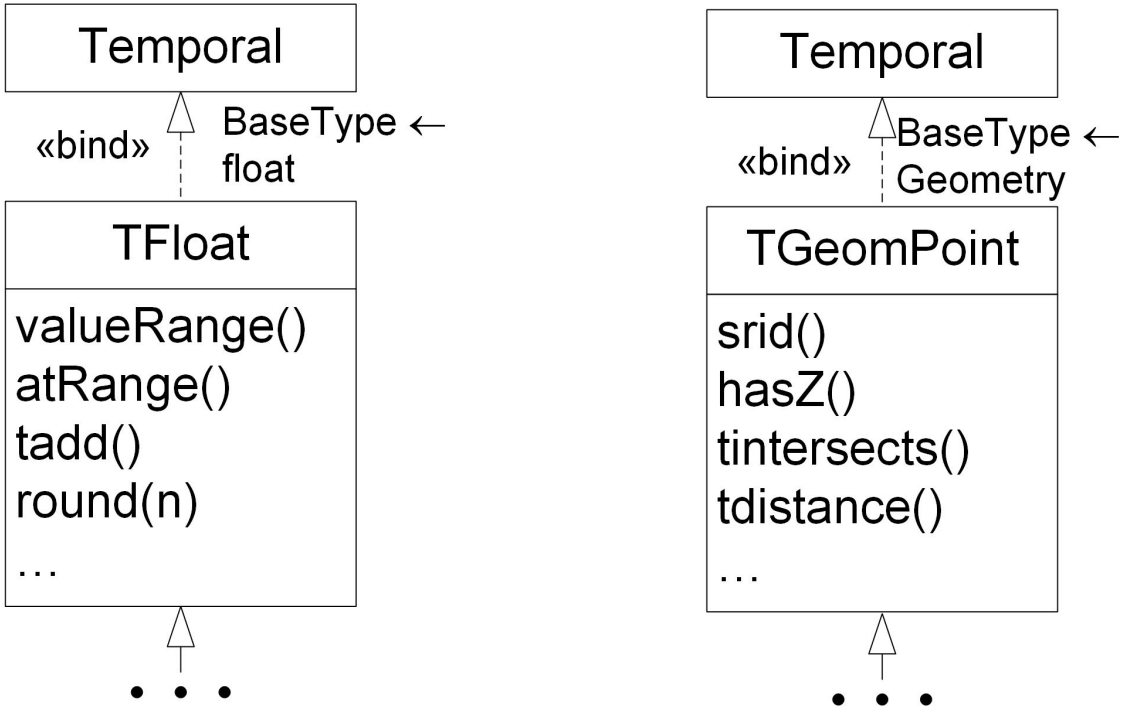
WKB: From Simple Features to Moving Features

- Geometries encoded using existing standards OGC Simple Feature Access and ISO/IEC 13249-3:2016
- We need to add timestamp, interpolation, and timeframe information
- Timestamp: Moving features are a series of observation couples
  $\{geo_1@t_1, \dots , geo_n@t_n\}$
- Interpolation: determines the value between two consecutive timestamps
  Discrete, stepwise, linear, …
- Timeframe: cope with signal loss (e.g., in tunnels) or explicit stop (e.g., lunch break of a driver)
  $\{geo_1@t_1, \dots , geo_n@t_n\}, \{geo_{n+1}@t_{n+1}, \dots , geo_{n+m}@t_{n+m}\}, ...$

MobilityDB Type System: Python Version (1)



MobilityDB Type System: Python Version (2)



MobilityDB Type System: SQL Version

```
SELECT tfloat '1@2000-01-01';

SELECT tgeompoint '{Point(1 1)@2000-01-01, Point(2 2)@2000-01-02,
    Point(1 1)@2000-01-03}';

SELECT tgeompoint 'SRID=5676;[Point(1 1)@2000-01-01,
    Point(2 2)@2000-01-02, Point(1 1)@2000-01-03)';

SELECT tgeogpoint 'Interp=Stepwise;{
    [Point(1 1)@2000-01-01, Point(2 2)@2000-01-02],
    [Point(2 2)@2000-01-03, Point(1 1)@2000-01-04) }';
```

MobilityDB Input/Output Formats

- Delegate spatial processing to PostGIS, takes care of the timestamp, interpolation and timeframe information
- Output: asText, asEWKT, asMFJSON, asBinary, asEWKB, asHexEWKB
- Input: tgeompointFromX, tgeogpointFromX for all the formats
- Idempotent operations up to floating-point precision for text encoding
- In the regression tests

```
SELECT DISTINCT asEWKT(tgeompointFromMFJSON(asMFJSON(temp)))
      = asEWKT(temp) FROM tbl_tgeompoint;
SELECT DISTINCT tgeompointFromEWKB(asEWKB(temp)) = temp
FROM tbl_tgeompoint;
```

EWKB in MobilityDB (1)

```
SELECT asEWKB(tgeompoint 'Point(1 1)@2000-01-01');

\x0181000000000000f03f000000000000f03f005c6c29ffffffff
Endian Flags 1 1 2000-01-01

SELECT asEWKB(tgeompoint '{Point(1 1)@2000-01-01,
      Point(2 2)@2000-01-02}');

\x0182020000000000000000f03f000000000000f03f005c6c29ffffffff
000000000000004000000000000000004000bc434713000000
Endian Flags #Instants 1 1 2000-01-01
2 2 2000-01-02
```

EWKB in MobilityDB (2)

```
SELECT asEWKB(tgeompoint '[Point(1 1)@2000-01-01,
      Point(2 2)@2000-01-02)');

\x0183020000001000000000000f03f000000000000f03f005c6c29ffffffff
000000000000004000000000000000004000bc434713000000
```

Endian Flags #Instants Bounds 1 1 2000-01-01 2 2 2000-01-02

```
SELECT asEWKB(tgeompoint 'SRID=5676;[Point(1 1)@2000-01-01,
Point(2 2)@2000-01-02]');

\x01c32c1600000200000003000000000000f03f000000000000f03f005c6c29ffffffff
000000000000004000000000000000004000bc434713000000
```

Endian Flags SRID #Instants Bounds 1 1 2000-01-01
2 2 2000-01-02

EWKB in MobilityDB (3)

```
SELECT asEWKB(tgeompoint '{
    [Point(1 1)@2000-01-01, Point(2 2)@2000-01-02),
    [Point(2 2)@2000-01-03, Point(1 1)@2000-01-04)}');
```

```
\x01840200000002000000010000000000000f03f000000000000f03f005c6c29ffffffff
0000000000000040000000000000004000bc434713000000
02000000010000000000000040000000000000004000 1c1b6527000000
000000000000f03f000000000000f03f007cf2823b000000
```

Endian Flags #Sequences #Instants Bounds 1 1 2000-01-01
2 2 2000-01-02
#Instants Bounds 2 2 2000-01-03
1 1 2000-01-04

WKB Implementation in MobilityDB (1)

```
/**
 * Writes into the buffer the temporal point represented in
 * Well-Known Binary (WKB) format
 */
static uint8_t *
tpoint_to_wkb_buf(const Temporal *temp, uint8_t *buf, uint8_t variant)
{
  ensure_valid_tempsubtype(temp->subtype);
  if (temp->subtype == INSTANT)
    return tpointinst_to_wkb_buf((TInstant *) temp, buf, variant);
  else if (temp->subtype == INSTANTSET)
    return tpointinstset_to_wkb_buf((TInstantSet *) temp, buf, variant);
  else if (temp->subtype == SEQUENCE)
    return tpointseq_to_wkb_buf((TSequence *) temp, buf, variant);
  else /* temp->subtype == SEQUENCESET */
    return tpointseqset_to_wkb_buf((TSequenceSet *) temp, buf, variant);
}
```

WKB Implementation in MobilityDB (2)

```c
/**
 * Writes into the buffer the temporal instant set point represented in
 * Well-Known Binary (WKB) format as follows
 * - Endian Flag
 * - Temporal Flags: Linear interpolation, hasSRID, isGeodetic, hasZ, subtype
 * - SRID (if requested)
 * - Count: Number of instants
 * - Output of the instants by function coords_ts_to_wkb_buf
 */
static uint8_t *
tpointinstset_to_wkb_buf(const TInstantSet *ti, uint8_t *buf, uint8_t variant)
{
  /* Set the endian flag */
  buf = endian_to_wkb_buf(buf, variant);
  /* Set the temporal flags */
  buf = tpoint_wkb_type((Temporal *) ti, buf, variant);
  /* Set the optional SRID for extended variant */
  if (tpoint_wkb_needs_srid((Temporal *) ti, variant))
    buf = integer_to_wkb_buf(tpointinstset_srid(ti), buf, variant);
  /* Set the count */
  buf = integer_to_wkb_buf(ti->count, buf, variant);
  /* Set the array of instants */
  for (int i = 0; i < ti->count; i++)
  {
    const TInstant *inst = tinstantset_inst_n(ti, i);
    buf = coords_ts_to_wkb_buf(inst, buf, variant);
  }
  return buf;
}
```

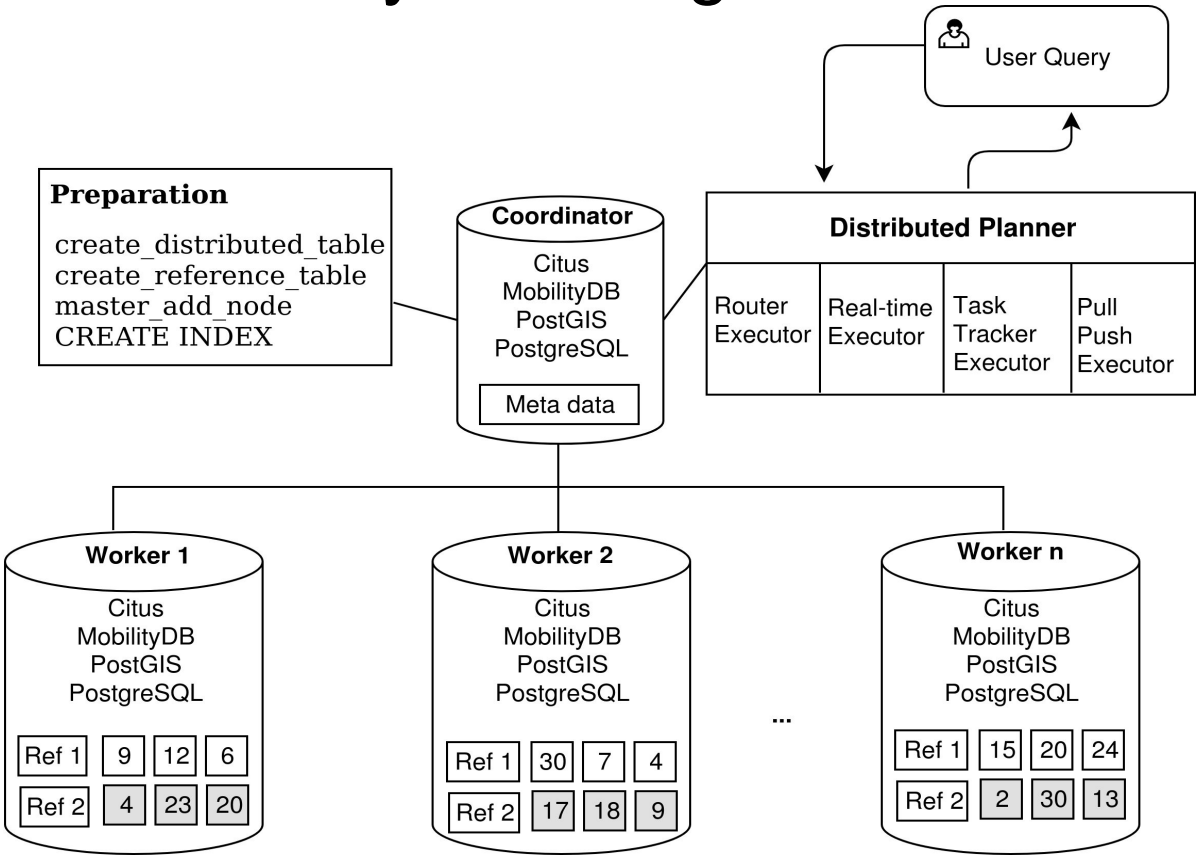
WKB Implementation in MobilityDB (3)

```
/**
 * Writes into the buffer the coordinates of the temporal instant point
 * represented in Well-Known Binary (WKB) format as follows
 * - 2 or 3 doubles for the coordinates depending on whether there is Z
 * - 1 timestamp
 */
static uint8_t *
coords_ts_to_wkb_buf(const TInstant *inst, uint8_t *buf, uint8_t variant)
{
  if (MOBDB_FLAGS_GET_Z(inst->flags))
  {
    const POINT3DZ *point = datum_get_point3dz_p(tinstant_value(inst));
    buf = double_to_wkb_buf(point->x, buf, variant);
    buf = double_to_wkb_buf(point->y, buf, variant);
    buf = double_to_wkb_buf(point->z, buf, variant);
  }
  else
  {
    const POINT2D *point = datum_get_point2d_p(tinstant_value(inst));
    buf = double_to_wkb_buf(point->x, buf, variant);
    buf = double_to_wkb_buf(point->y, buf, variant);
  }
  buf = timestamp_to_wkb_buf(inst->t, buf, variant);
  return buf;
}
```

Efficiency of Different Input/Output Formats

- Benchmark data transfer between tables of varying size
- Assess the data size and transfer time of various formats
- Necessary in various situations
- Partitioning: Split a large data set into the nodes of the cluster or in a cloud environment
- Autoscaling: dynamically adjust the number of nodes according the load, which implies repartitioning the data to the current number of nodes
- Distributed query processing: Coordinator node receives a query that is executed in parallel by the worker nodes and the partial results of the query should be transferred to the coordinator
- Visualization: Send data to QGIS, Kepler.gl, ….

Distributed MobilityDB: Integration with Citus



Visualization with Kepler.gl*

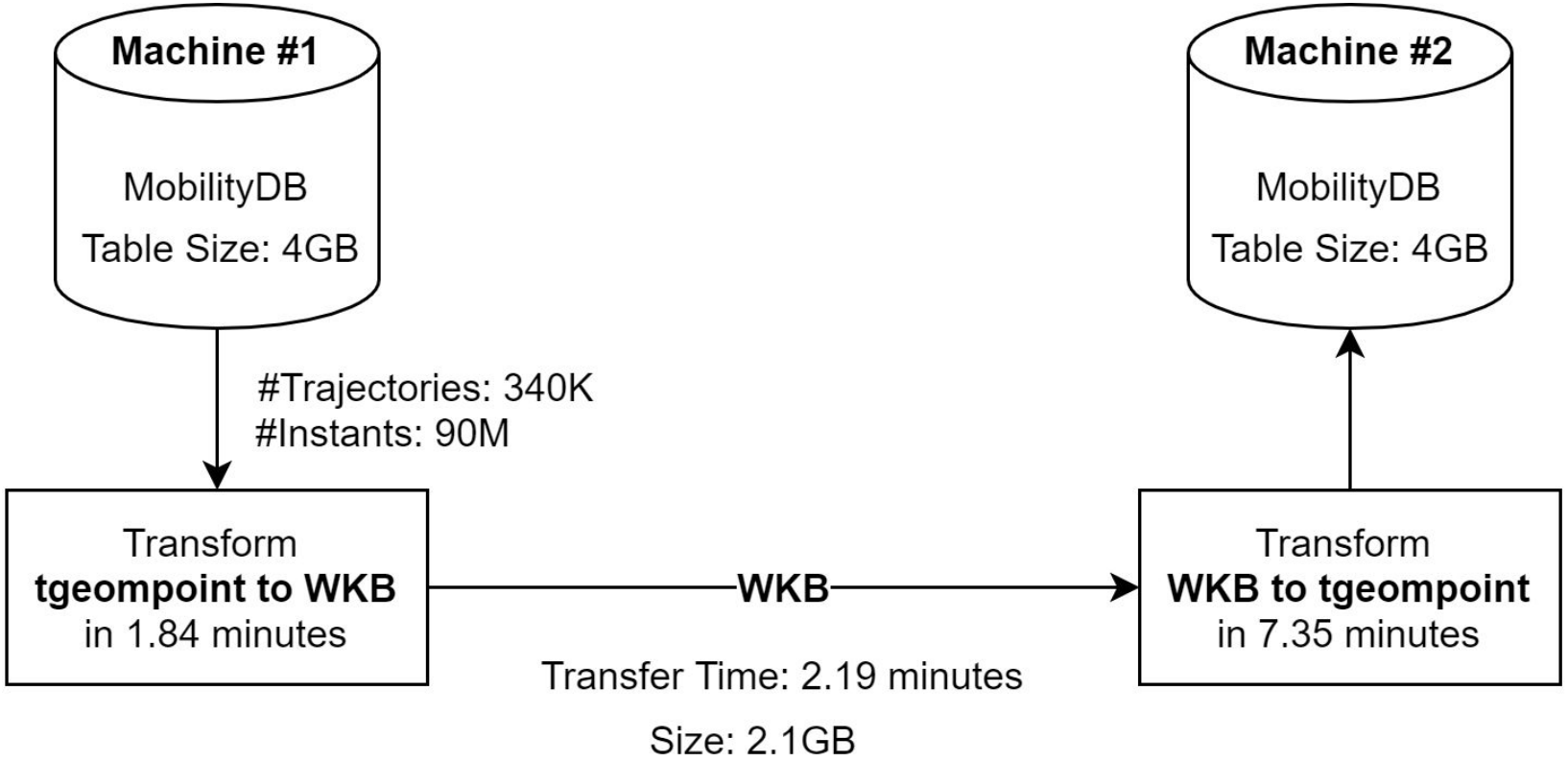
* Work done by Fabrício Ferreira as part of his Master's Thesis

Description of the Benchmark*

- Objective: Measuring the performance of transferring moving point data from one machine to another using different formats
- MobilityDB formats: WKT, EWKT, WKB, EWKB, HexEWKB, and MFJSON
- Dataset: MobilityDB Generator based on BerlinMOD
- MobilityDB data type: TGeomPoint (Sequence subtype)
- Table sizes: 1GB, 2GB, 4GB, 8GB, and 16GB
- Number of runs: Each experience is repeated five times and the resulting time is averaged

* Work done by Mohamed Bakli as part of his PhD Thesis

Description of the Benchmark: 4GB Table



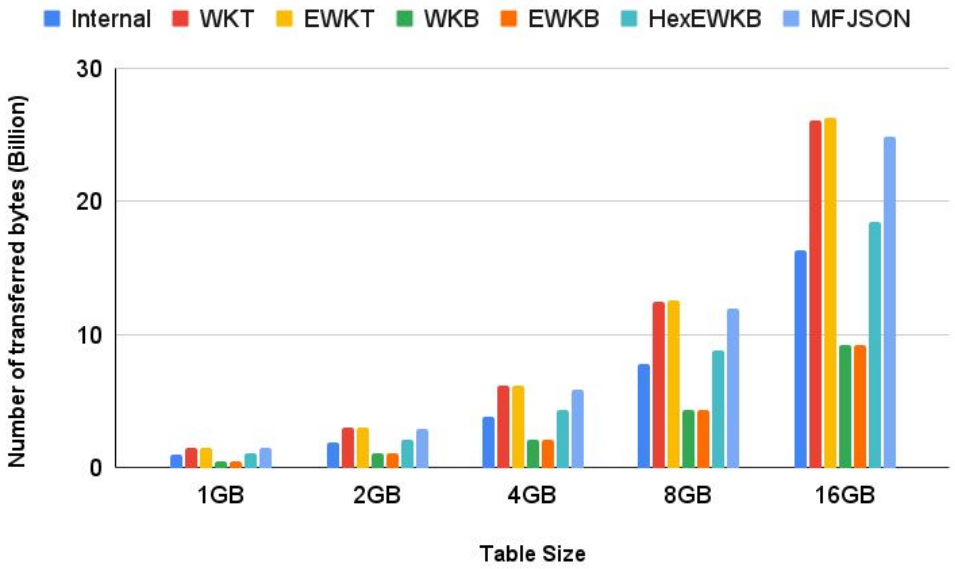
Benchmark Results

MobilityDB Table - Specs					MobilityDB Formats - Run-time (Minutes)						
Table Size	# trajectories	# Instants	Min #Instants/ Sequence	Max #Instants/ Sequence	Internal	WKT	EWKT	WKB	EWKB	HexEWKB	MFJSON
1GB	113K	22.5M	2	1900	4.31	6.64	6.68	2.19	2.25	2.26	6.11
2GB	201K	44.6M	2	1792	9.04	13.31	13.35	4.38	4.42	4.51	12.19
4GB	340K	90M	2	1900	17.35	26.52	26.7	8.79	8.92	9.05	24.58
8GB	888K	182.2M	2	1857	35.01	54.25	54.32	17.78	17.86	18.52	49.27
16GB	1.90M	369.5M	2	1900	70.08	110.2	110.98	35.32	35.82	37.68	102.48

- **(E)WKT:** (E)WKT text format, extended for moving features
- **(E)WKB:** (E)WKB binary format, extended for moving features
- **HexEWKB:** Hexadecimal EWKB binary format, extended for moving features
- **MFJSON**: OGC standard format for moving features

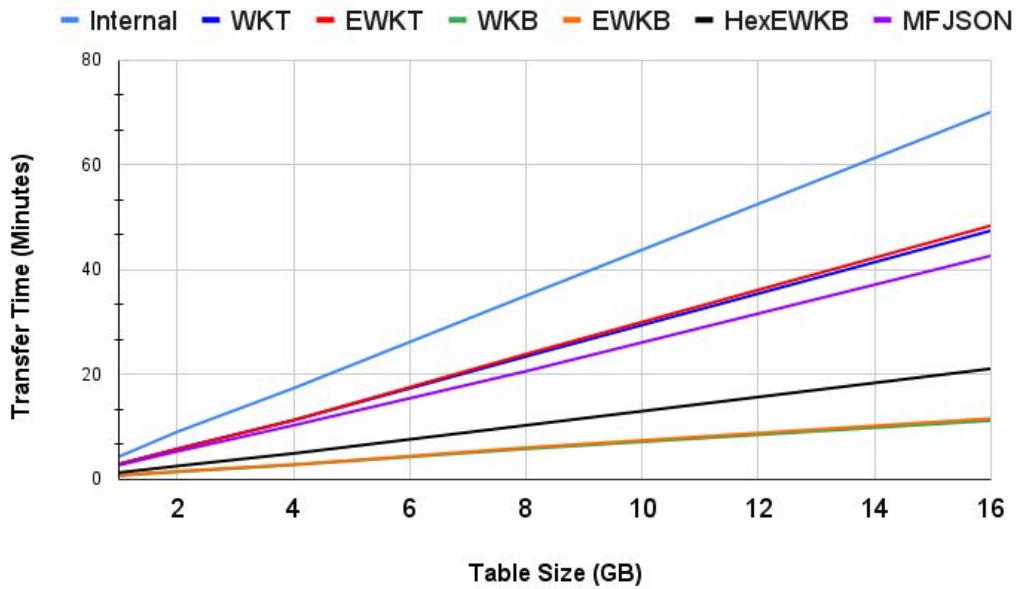
Benchmark Results

- Number of transferred characters from one machine to another using different formats:



Benchmark Results

- Transfer time from one machine to another using different formats:



Thanks for listening !