Road public transport in Moscow analysis:
from PostGIS to MobilityDB

Nina Belyavskaya

Feb 5th 2020

PgConf.ru

Московский Транспорт

Мосгортранс

Mosgortrans Mobility Department

Our mission to optimize road public transport routes and timetables in Moscow.

Such metropolis as Moscow needs a comfortable public transport system.

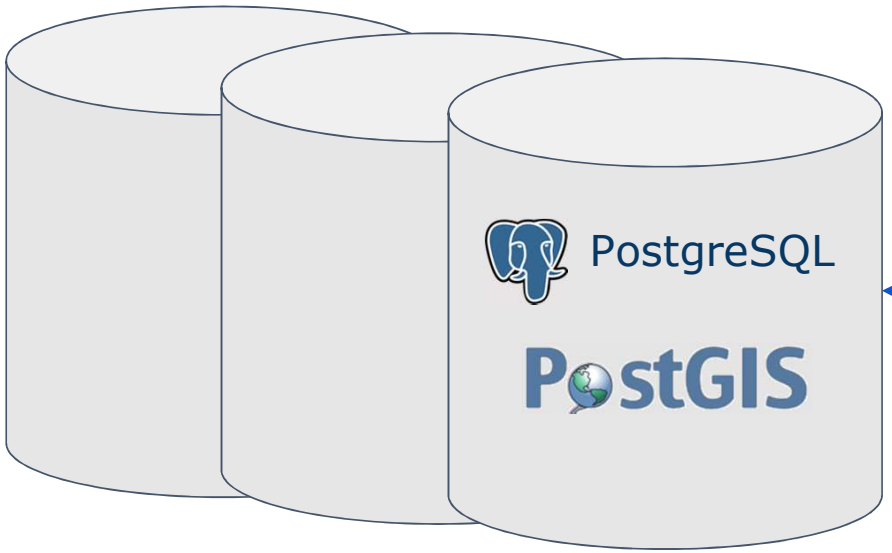
- Not only subway, but road transport - bus, trolleybus, tram

Transport should

- go frequently
- follow the timetable
- go quickly.

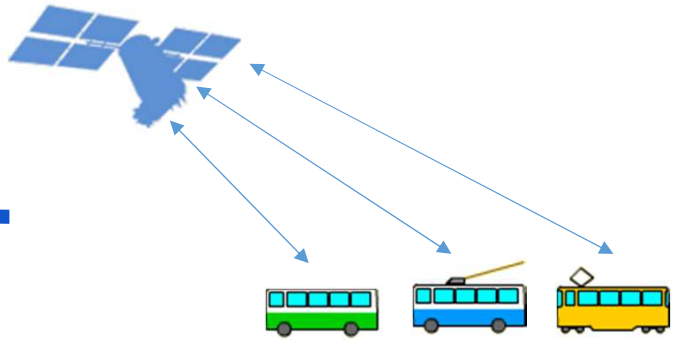
Data collection from navigation system

All Moscow road public transport uses GLONASS equipment.



~ 300 GB per year

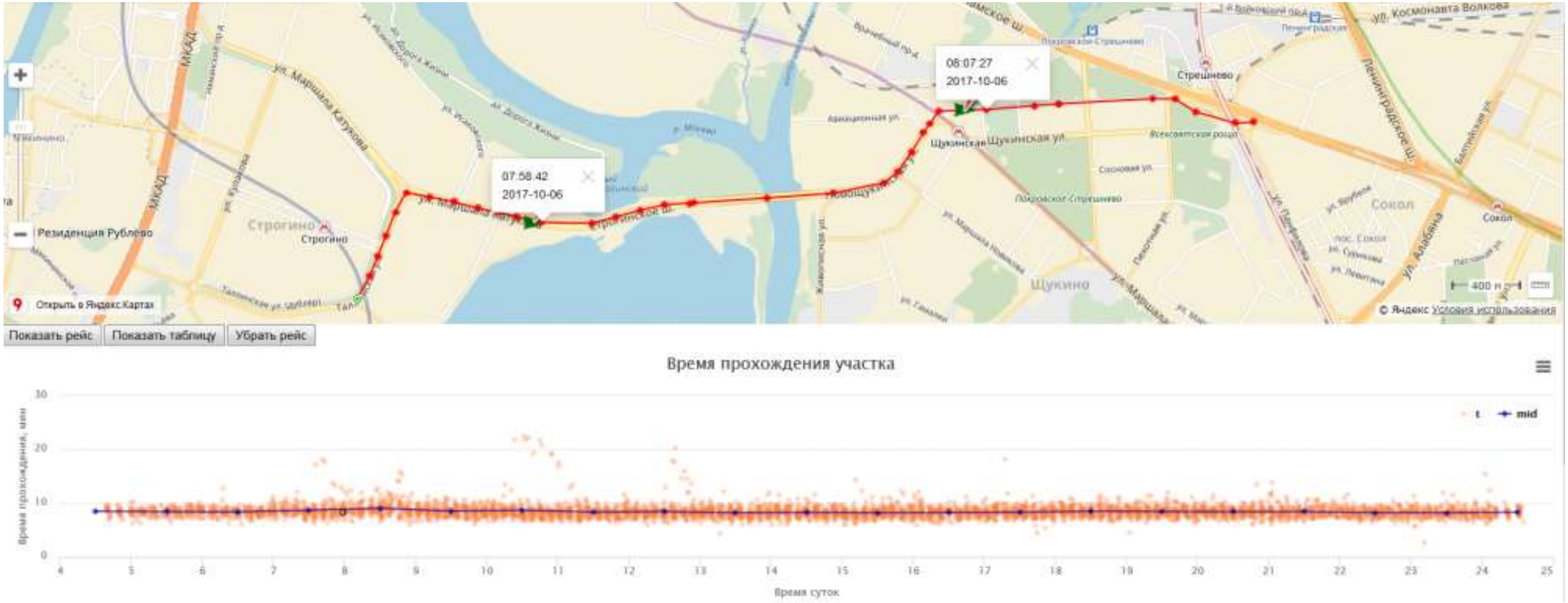
> 500 MB per day



The coordinates of each vehicle are determined every 30 sec.

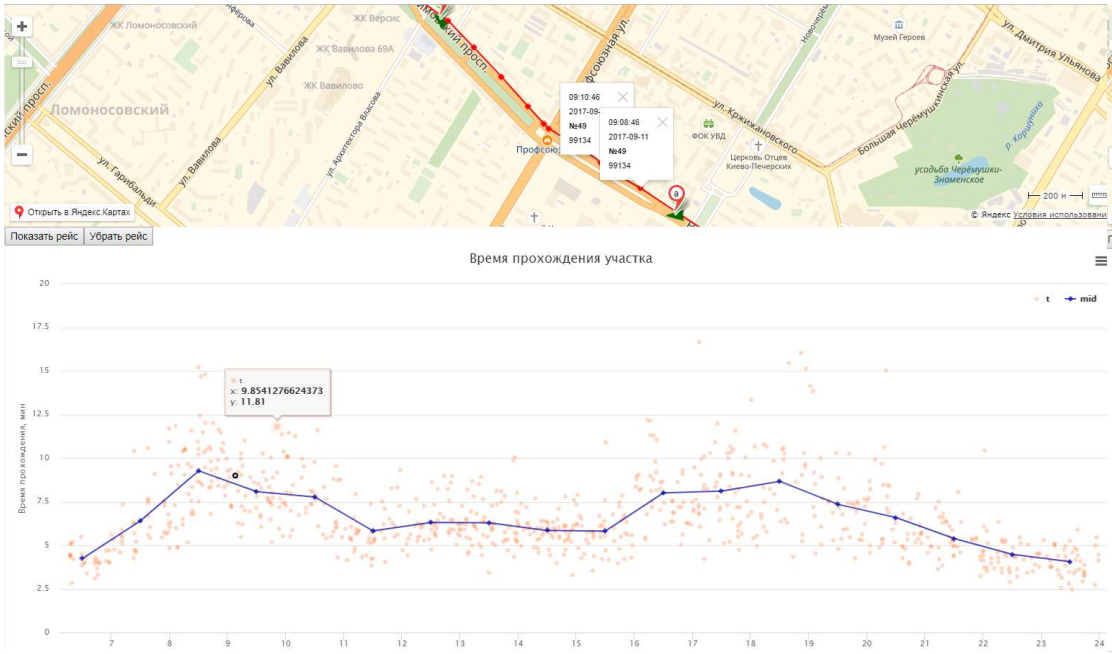
Data analysis – travel time

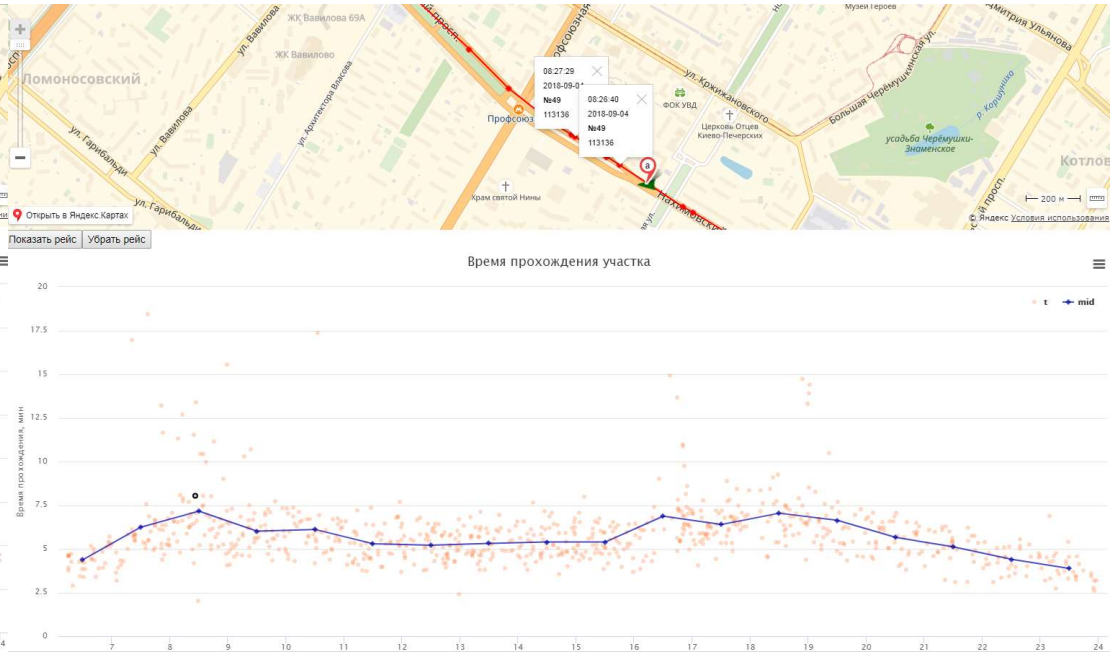
Tram №21 October 2017



Data analysis – travel time

Trolleybus №49



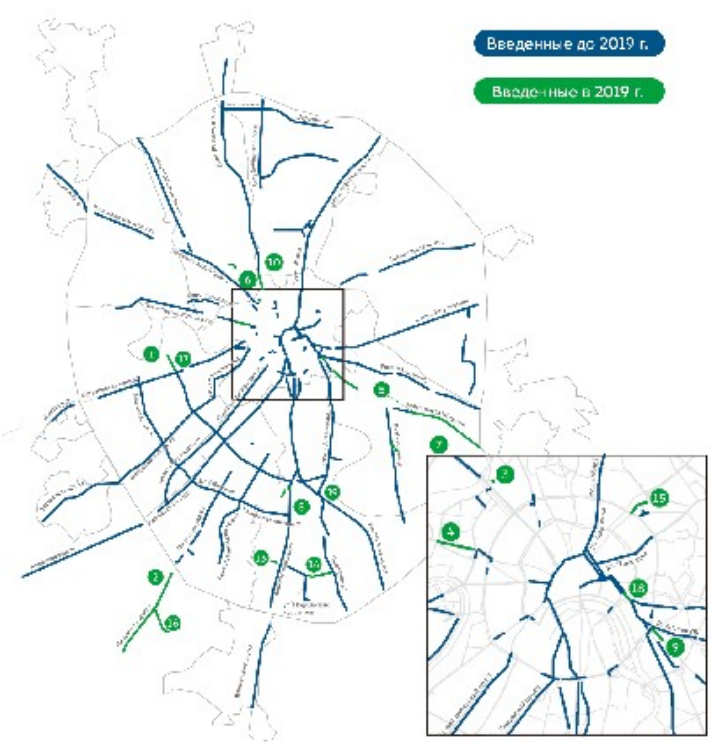


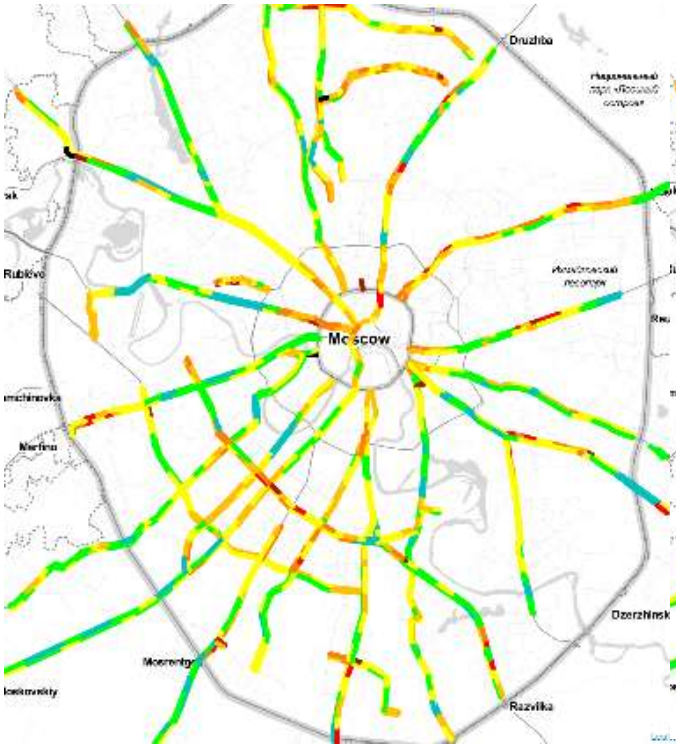
September 2017

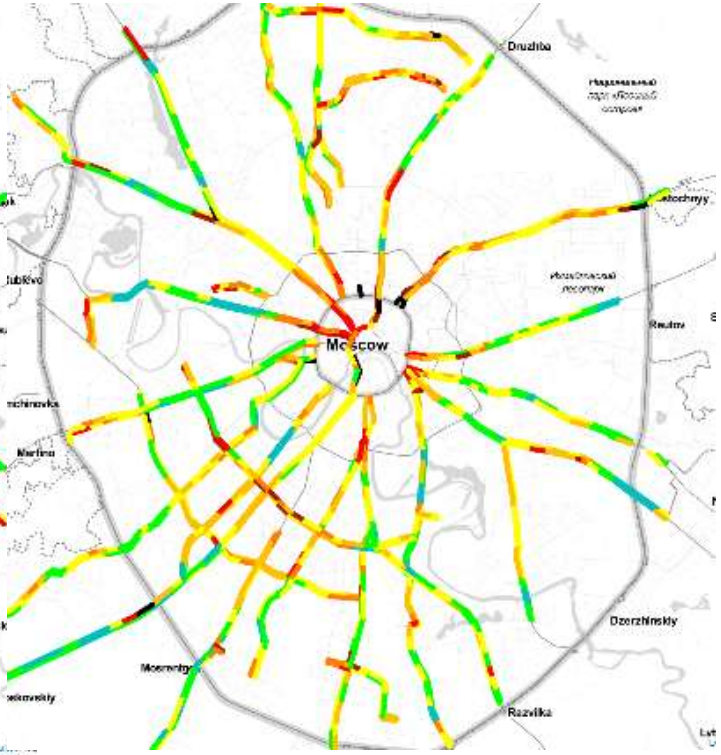
September 2018

Data analysis – velocity maps

Moscow bus lanes







MobilityDB

https://github.com/ULB-CoDE-WIT/MobilityDB

MobilityDB is an open source software that adds support for temporal and spatio-temporal objects to the PostgreSQL database and its spatial extension PostGIS.

MobilityDB is developed by the Computer & Decision Engineering Department of the Université Libre de Bruxelles (ULB) under the direction of Prof. Esteban Zimányi.

PostGIS -> MobilityDB migration

```
SELECT gl_uuid, dtime,
ST_X(gl_point) as x,
ST_Y(gl_point) as y
…
ORDER BY gl_uuid, dtime
```

```
gl_uuid,
tgeompoint '[
        Point(x1 y1)@t1,
        …
        Point(xN yN)@tN
]'
```

Tracks breaks when $t_N - t_{N-1} > 5$ min

At PostGIS point and time are stored in different columns, every point is in it's own row.

At MobilityDB point and time are united so that one row is a whole track.

PostGIS -> MobilityDB migration

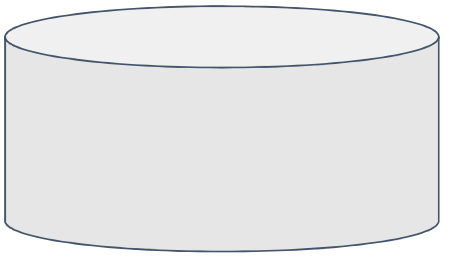
44384	2015-04-06 06:38:00	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:38:30	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:39:00	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:39:30	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:40:00	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:40:30	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:41:00	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:41:30	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:42:00	POINT(37.3826816 55.7937783)
44384	2015-04-06 06:42:30	POINT(37.3826816 55.7937783)

10 billion rows a day

44384	[POINT(37.3826816 55.7937783)@2015-04-06 06:38:00+03, POINT(37.3826816 55.7937783)@2015-04-06 0...
44399	[POINT(37.6126166 55.7274032)@2015-04-06 07:14:29+03, POINT(37.6118683 55.7274732)@2015-04-06 0...
44399	[POINT(37.6127783 55.7265099)@2015-04-06 05:32:14+03, POINT(37.6127783 55.7265099)@2015-04-06 0...
62736	[POINT(37.6078283 55.7158566)@2015-04-06 05:35:17+03, POINT(37.607475 55.71504)@2015-04-06 05:35:...
62771	[POINT(37.6124233 55.7264416)@2015-04-06 05:07:57+03, POINT(37.6124233 55.7264416)@2015-04-06 0...
67756	[POINT(37.608135 55.7163933)@2015-04-06 04:47:23+03, POINT(37.60777 55.7153983)@2015-04-06 04:47:...
67762	[POINT(37.6093483 55.7190449)@2015-04-06 16:58:07+03, POINT(37.6094966 55.7188982)@2015-04-06 1...
67762	[POINT(37.6099266 55.7209099)@2015-04-06 04:41:30+03, POINT(37.60921 55.7190516)@2015-04-06 04:4...

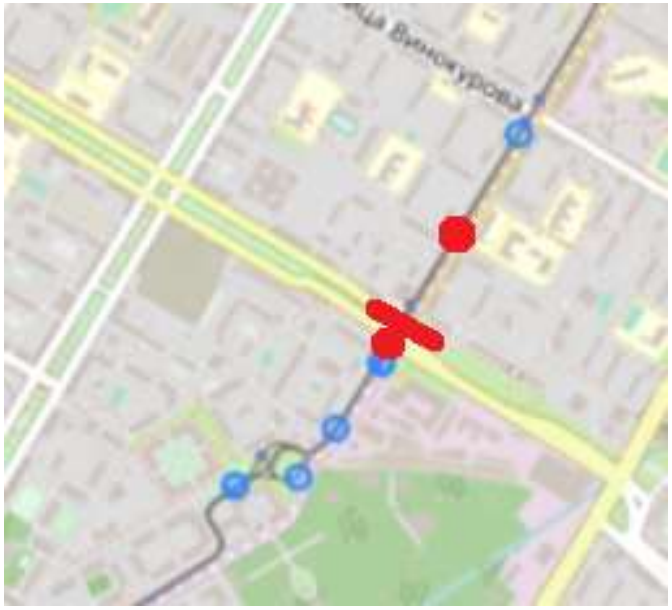
15 thousand rows

~ 5 MB per day



2GB per year

Travel time - PostGIS

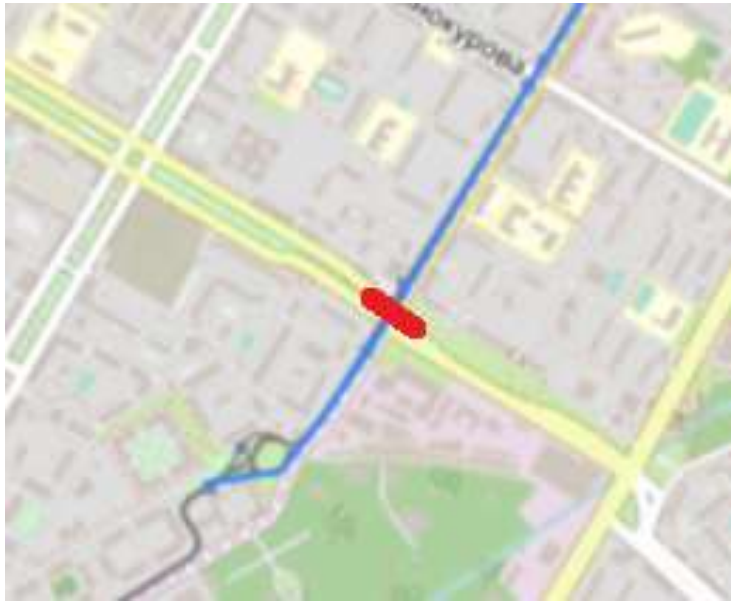


The query returns two points. We have to approximate exact intersection time.

```
WITH endpoint AS (SELECT ST_LineFromText(
'LINESTRING(37.5393505 55.6936058, 37.5365505
55.6908058)'
) AS l)
SELECT gl_uuid, t, t1
    ST_X(p0) as x0, ST_Y(p0) as y0,
    ST_X(p1) as x1, ST_Y(p1) as y1
FROM (
    SELECT gl_uuid, t, gl_point AS p0,
                lead(t) over w as t1, lead(gl_point)
over w as p1
     FROM tracks
     WINDOW w AS (PARTITION BY gl_uuid ORDER BY t)
) as points, endpoint
WHERE ST_LineCrossingDirection(l,ST_MakeLine(p0,p1))=1
AND abs(ST_X(p1) - ST_X(p0)) < 0.05
AND abs(ST_Y(p1) - ST_Y(p0)) < 0.05
ORDER BY gl_uuid, t1
```

The query uses window function to make segments from points.

Travel time - MobilityDB

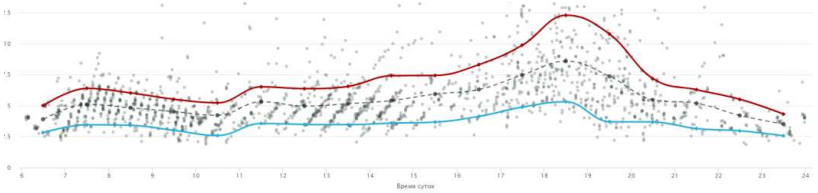


The query returns an exact intersection time.

```
WITH
  endpoint as (SELECT ST_LineFromText(
'LINESTRING(37.5393505 55.6936058, 37.5365505 55.6908058)'
  ) AS l),
  timesets as (
    SELECT gl_uuid, timestamps(atGeometry(tline,l)) as ts
      FROM mtracks, endpoint
      WHERE route IN ('49') AND tline && l),
  times as (SELECT gl_uuid, unnest(ts) as t FROM timesets),
  lines as (
     SELECT a.gl_uuid, route, t,
          getValues(atPeriod(tline,period(t-interval '15
sec',t+interval '15 sec'))) as points
      FROM times a, mtracks b
      WHERE a.gl_uuid=b.gl_uuid AND t <@ tline)
SELECT gl_uuid, route, t::time, 'B' as point
FROM lines, endpoint
WHERE ST_LineCrossingDirection(l,points)=1
ORDER BY gl_uuid, t
```

Travel time calculation - comparison

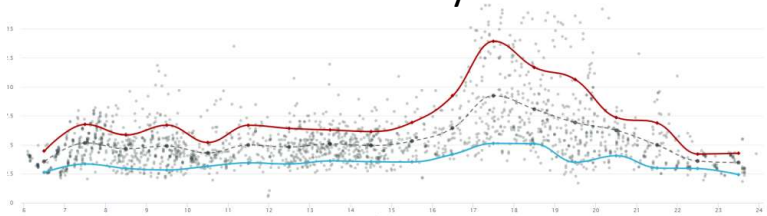
PostGIS



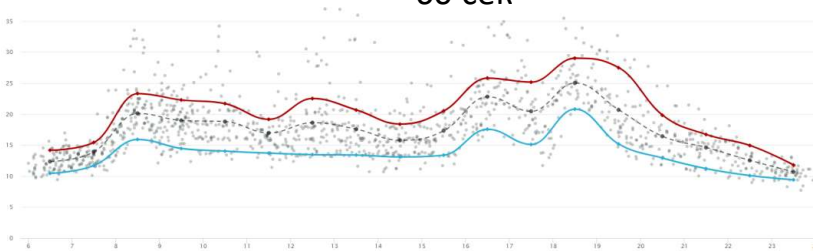
~60 сек

Bus 224
Vernadsky
- Leninsky.
April 2015

MobilityDB

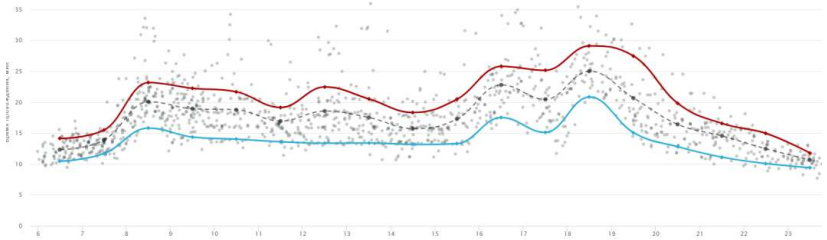


~20 сек

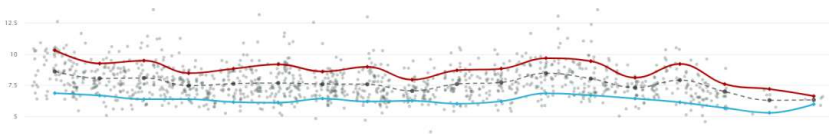


~20 сек

Trolleybus 49
Tsurupy str.
- Leninsky.
February 2015

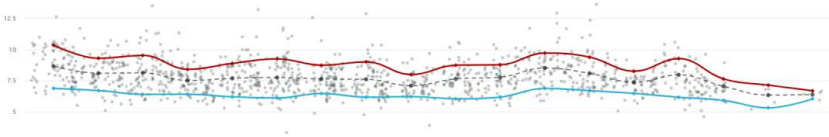


~10 сек



~9 сек

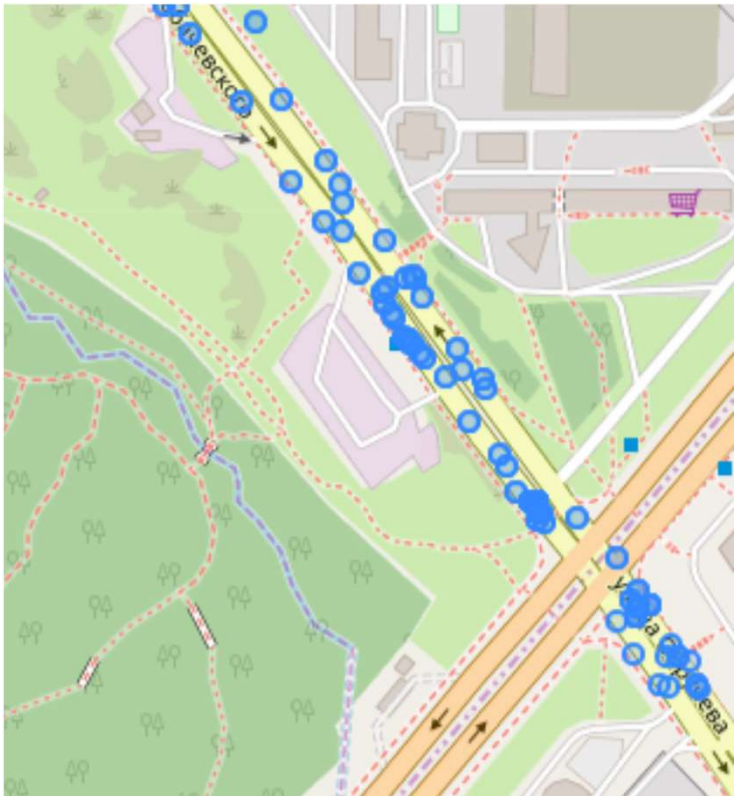
Tram 21
Strogino -
Shukino
February 2015



~5.5 сек

Московский Транспорт | Мосгортранс

Velocity map – PostGIS



```
WITH segments AS (
     SELECT
'LINESTRING(37.44654993 55.72910825,…,
            37.44820638 55.72529983)'::geometry
        AS line
)
SELECT
    gl_uuid, t, gl_point, ST_X(gl_point) as lon,
ST_Y(gl_point) as lat, route,
    ST_DistanceSphere(gl_point,line) AS r FROM
    tracks tr, segments s
    WHERE t>'7:00' AND t< '9:00'
    AND ST_DistanceSphere(gl_point,line) < 20
    ORDER BY gl_uuid
```

We have got a set of points, we make tracks from them and choose those that go in right direction.

Velocity map – MobilityDB

We just have got a velocity.

```
WITH segments as (SELECT
 ST_Buffer(
  'LINESTRING(37.44654993 55.72910825,…,37.44820638
55.72529983)'::geography,
  20)::geometry as shape,
 degrees(ST_Azimuth(
      'POINT(37.44654993 55.72910825)'::geometry,
      'POINT(37.44820638 37.44820638)'::geometry)) as azimuth),
pbox as (
SELECT atPeriod(tline, period '[2015-04-06 07:00,2015-04-06 09:00]') AS
tline
     FROM mtracks),
lines as (SELECT unnest(sequences(atGeometry(tline,shape))) as track
                    FROM pbox, segments
                    WHERE intersects(shape,tline))
SELECT
     ST_Length(getValues(track)::geography)/
     extract('epoch' from (endTimestamp(track)-startTimestamp(track)))
     as v
FROM lines, segments WHERE
ABS(degrees(ST_Azimuth(startValue(track),endValue(track)))-azimuth)<60
```

Velocity map calculation – comparison

PostGIS

~ 4.3 min.

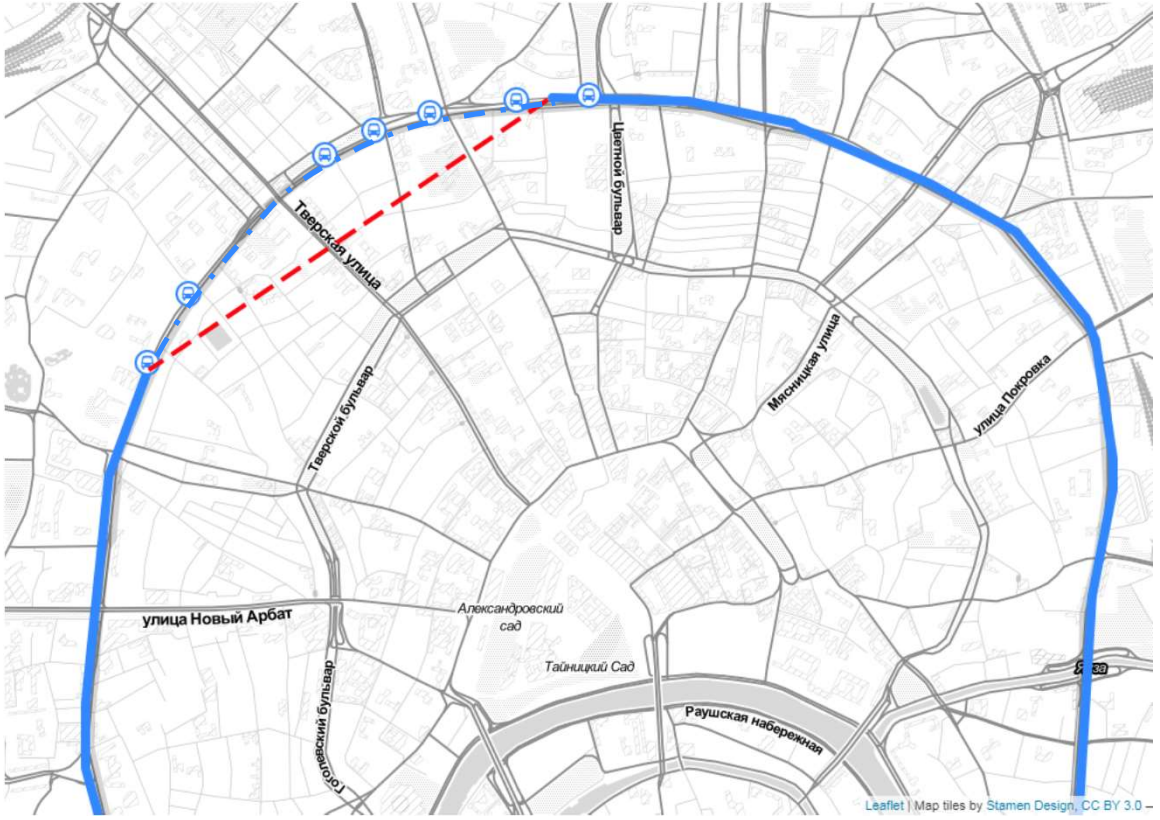
MobilityDB

~ 2.5 min.

Future: track restoration?

There are situations of navigation failure when no data is received within several minutes.

- Not suitable for velocity measurement because velocity isn't constant.
- Useful for estimation the frequency of arrival of vehicles to a stop.



Conclusions

MobilityDB extension allows:

- significantly save disk space when storing transport tracks;
- accelerate transport analytics calculations.

MobilityDB development team is ready to cooperation, they answer the questions quickly and help with problem solution.

MobilityDB is new usable tool. It worth to learn it and to work with it.

https://github.com/ULB-CoDE-WIT/MobilityDB

Thanks for attention!

belyavskaya_no@mosgortrans.ru
nina@fable.ru

Московский Транспорт

Мосгортранс