3D and exact geometries for PostGIS

FOSDEM PGDay

02-01-2013 – Hugo Mercier / Oslandia

Oslandia

Oslandia

- PostGIS, QGIS, Mapserver suite
- Training
- Support
- Development

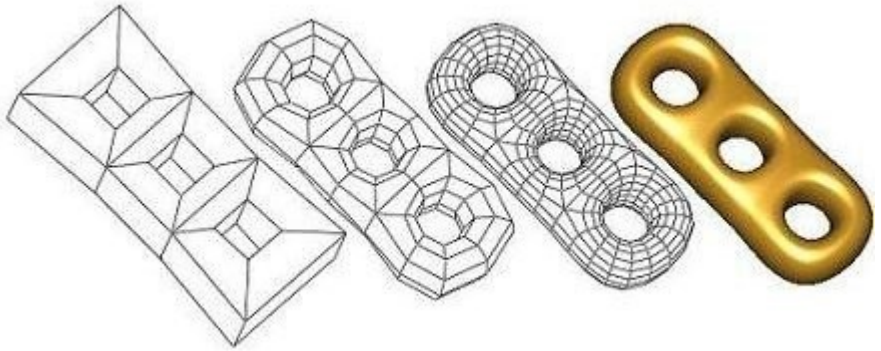
Context

- FEDER-funded program e-PLU
  - City modelling applications
- 3D spatial operations
  - PostGIS geometries can carry a z coordinate
  - What about spatial processing ?
  - IGN – Oslandia collaboration

CGAL

GEOS (PostGIS geometry backend) is 2D only

Appealing candidate : CGAL



Modern C++ framework

Lots of 2D/3D algorithms already implemented

Exact computational model !

Does it perform well ?

Exact computation

CGAL is templated by geometric 'Kernel's

Must use an 'exact' kernel for constructions

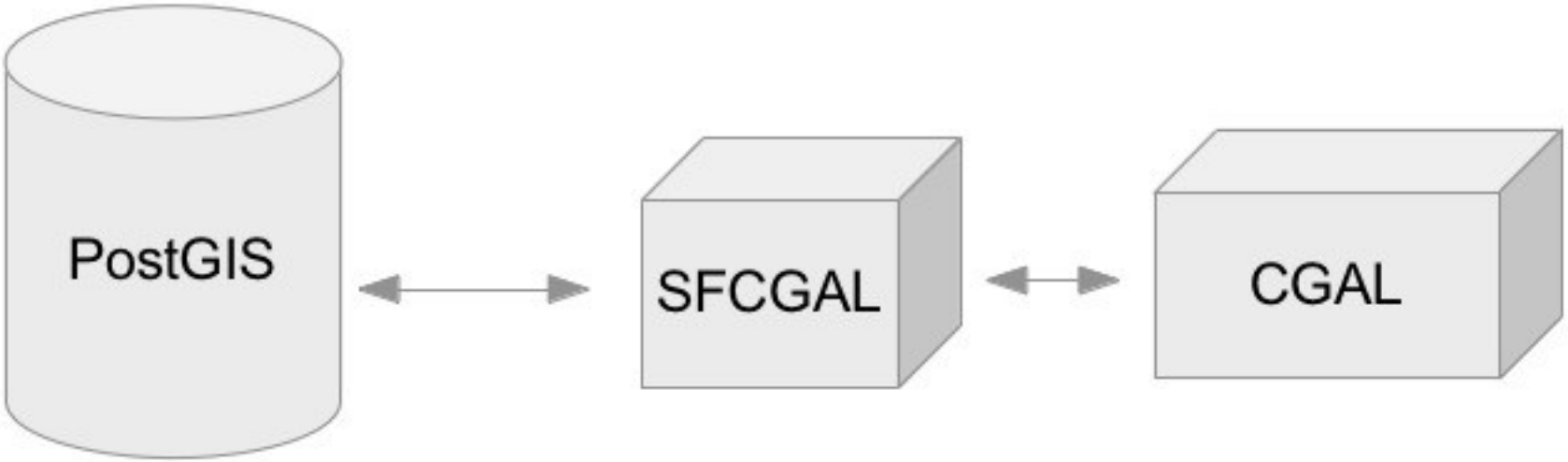
- Arbitrary precision numbers
- Lazy evaluation of an expression tree
- Interval arithmetics

<example?>

SFCGAL

Design of an OGC Simple Features compliant framework on top of CGAL

Our own PostGIS branch (postgis-sfcgal)



SFCGAL

Currently supported :

- 2D and 3D intersection (including solids)
- 2D and 3D intersection test (including solids)
- 2D and 3D convex hull
- 2D and 3D triangulations
- 3D extrusion
- 2D and 3D distances
- (in progress) buffers

PostGIS-SFCGAL

Postgis-sfcgal :

- Optional support for SFCGAL functions
- Using the 'sfcgal' schema
- e.g. :

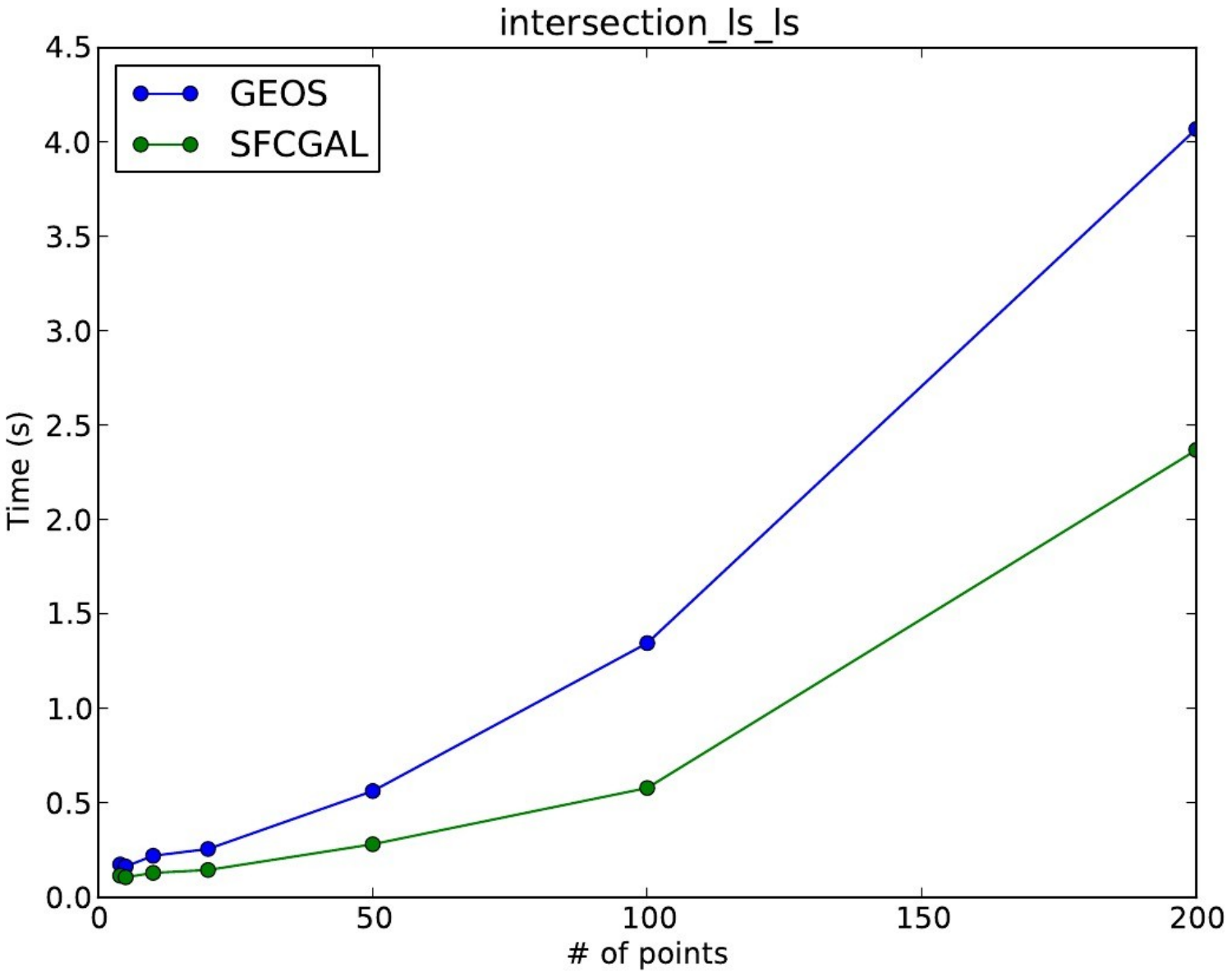
```
SELECT sfcgal.ST_Intersects( g1, g2 )
SELECT sfcgal.ST_3DIntersection(g1, g2)
...
```

SFCGAL vs GEOS

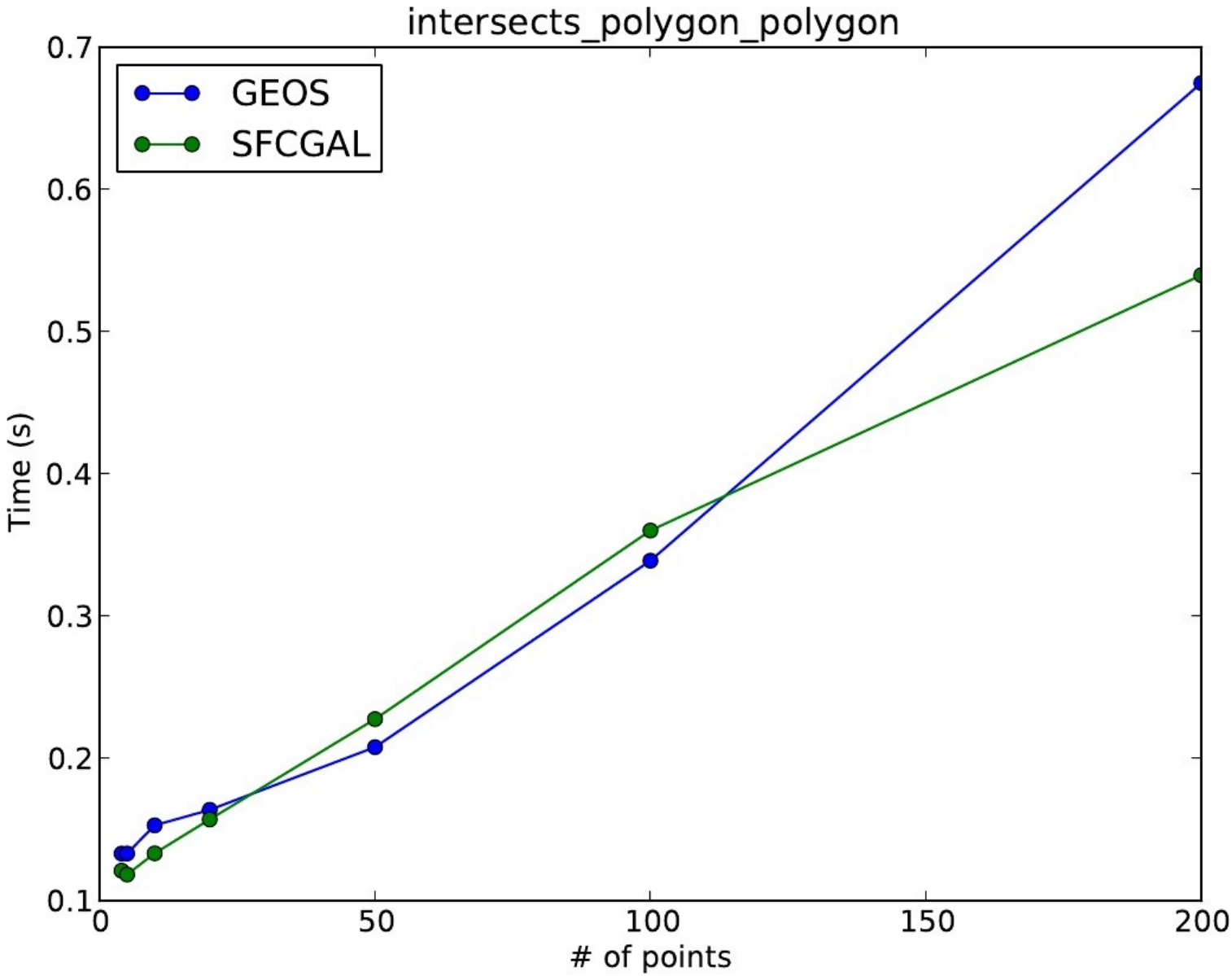
Performance comparison

- PostGIS based
- 2D Only
- Varying geometry's number of points

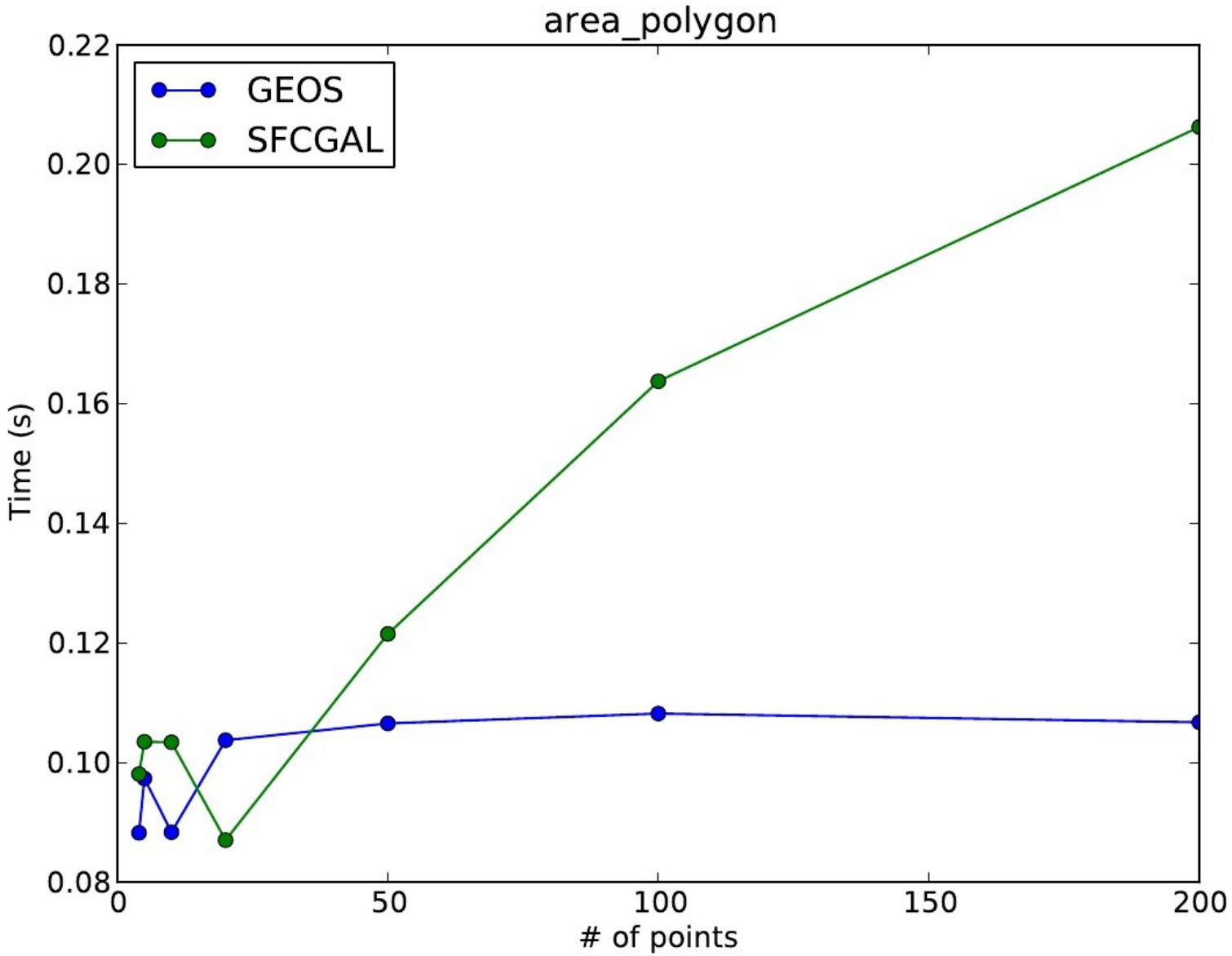
SFCGAL vs GEOS



SFCGAL vs GEOS



SFCGAL vs GEOS



OSLANDIA

SFCGAL vs GEOS

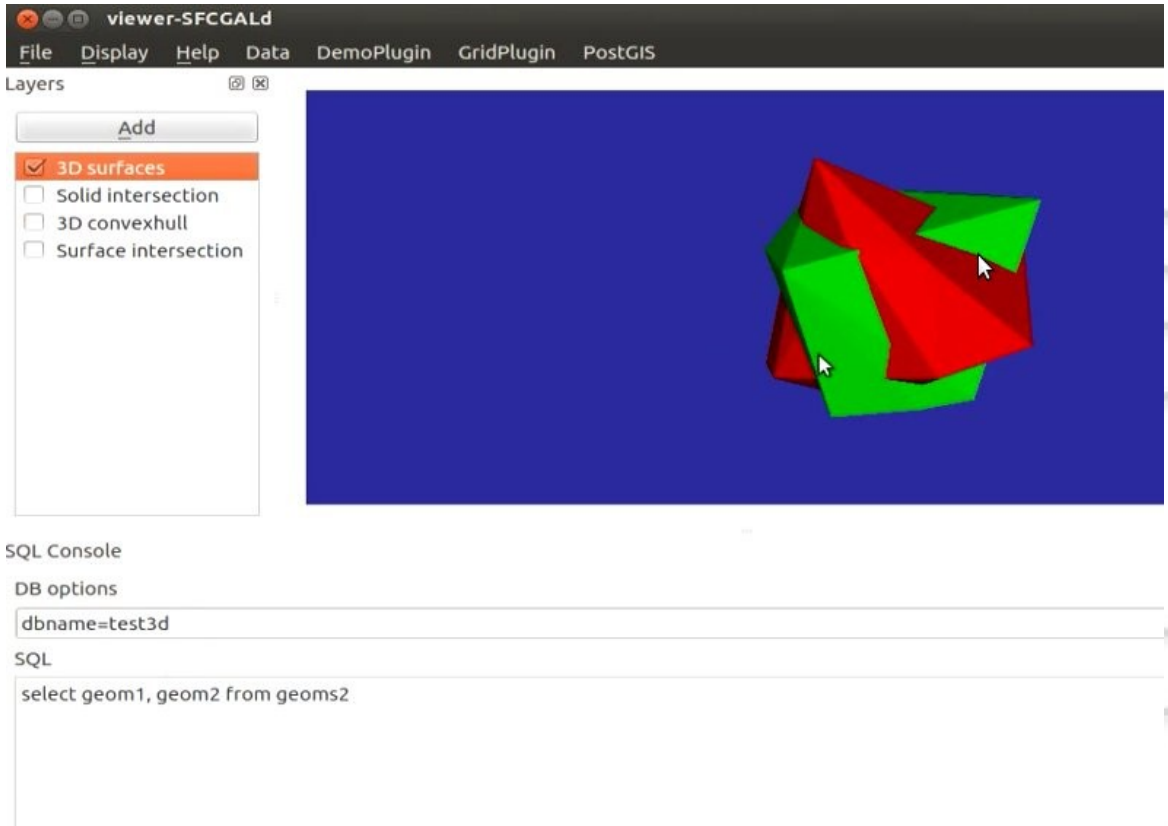
Results are very promising

- SFCGAL is sometimes better, sometimes worse
- Considering SFCGAL is way less mature than GEOS
- Comparable behaviour and space for improvements !

3D view

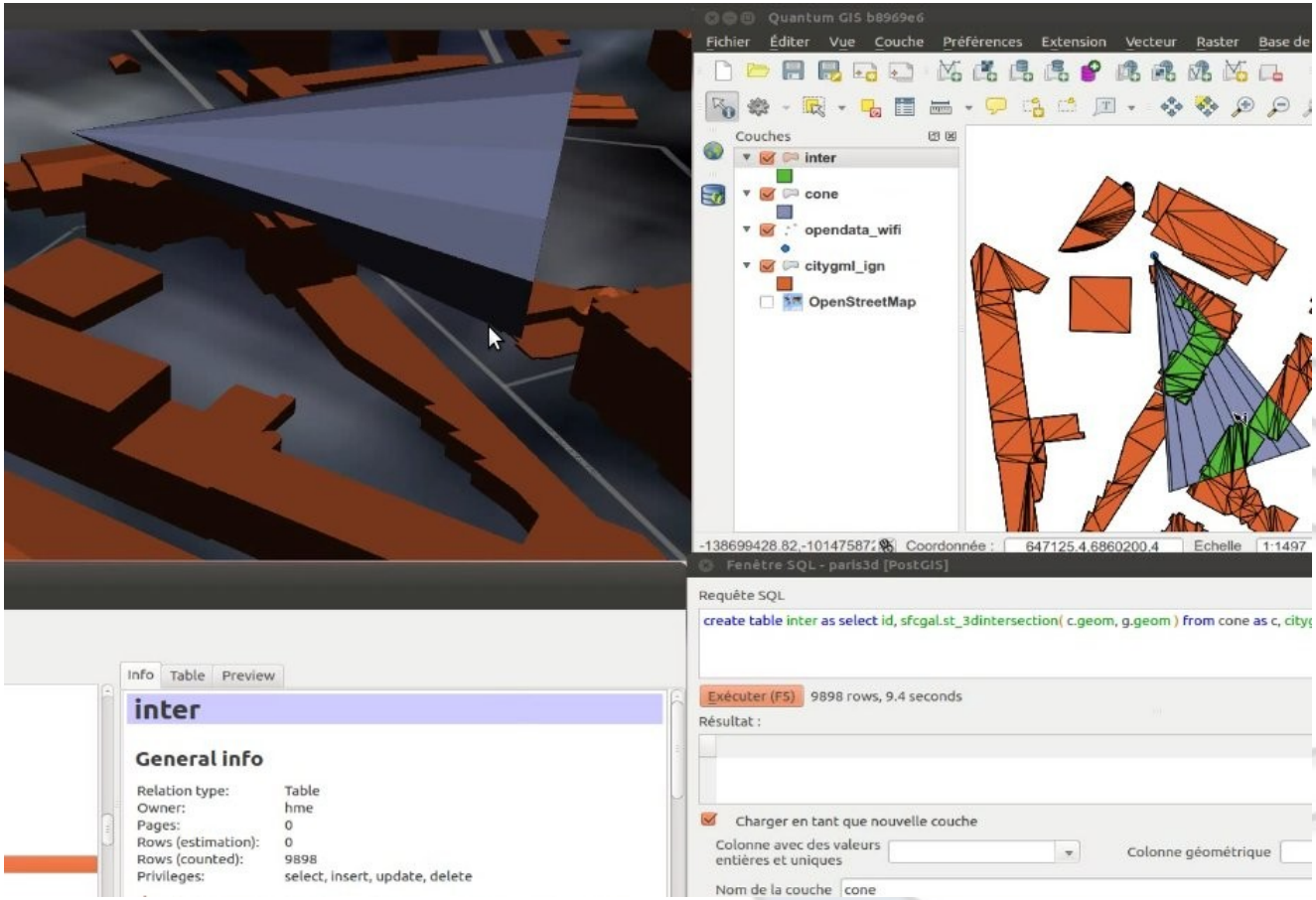
Couple PostGIS 3D to a 3D viewer

SFCGAL viewer



3D view

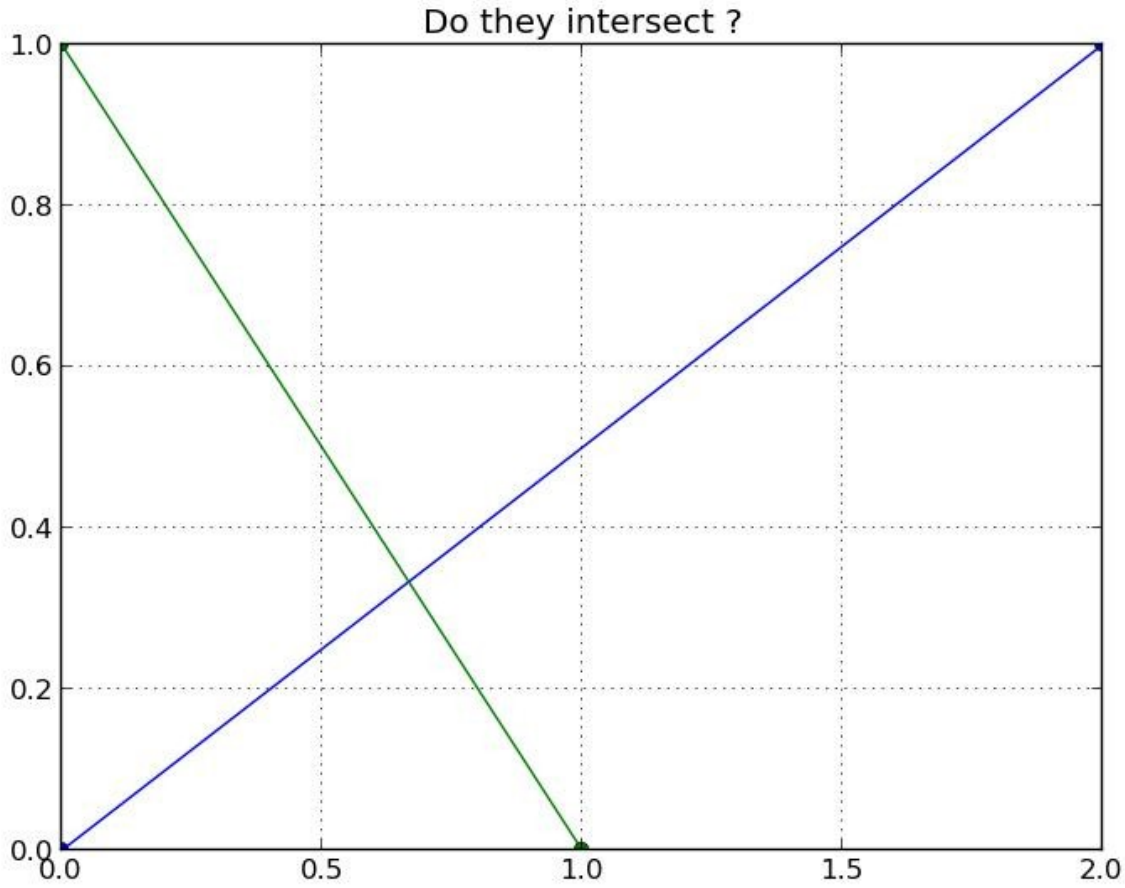
QGIS with the Globe plugin



OSLANDIA

Precision issues

Do they intersect ?



Precision issues

Do they intersect ?

```
SELECT
  ST_Intersects(
    ST_Intersection(
      'LINESTRING(0 0,2 1)'::geometry,
      'LINESTRING(1 0,0 1)'::geometry),
    'LINESTRING(0 0,2 1)'::geometry);
 st_intersects
---------------
 f
(1 row)
```

Should be true !

GEOS only supports 'double' numbers

Support for exact geometries

New 'exact_geometry' type

- Coordinates stored with arbitrary precision
- Serialization/deserialization process

```
SELECT Sfcgal.ST_Intersection(
      'LINESTRING(0 0,2 1)'::exact_geometry,
      'LINESTRING(1 0,0 1)'::exact_geometry) ;
 st_intersection
-----------------
 POINT(2/3 1/3)
SELECT Sfcgal.ST_Intersects(
    Sfcgal.ST_Intersection(
      'LINESTRING(0 0,2 1)'::exact_geometry,
      'LINESTRING(1 0,0 1)'::exact_geometry),
    'LINESTRING(0 0,2 1)'::exact_geometry);
 st_intersects
---------------
 t
(1 row)
```

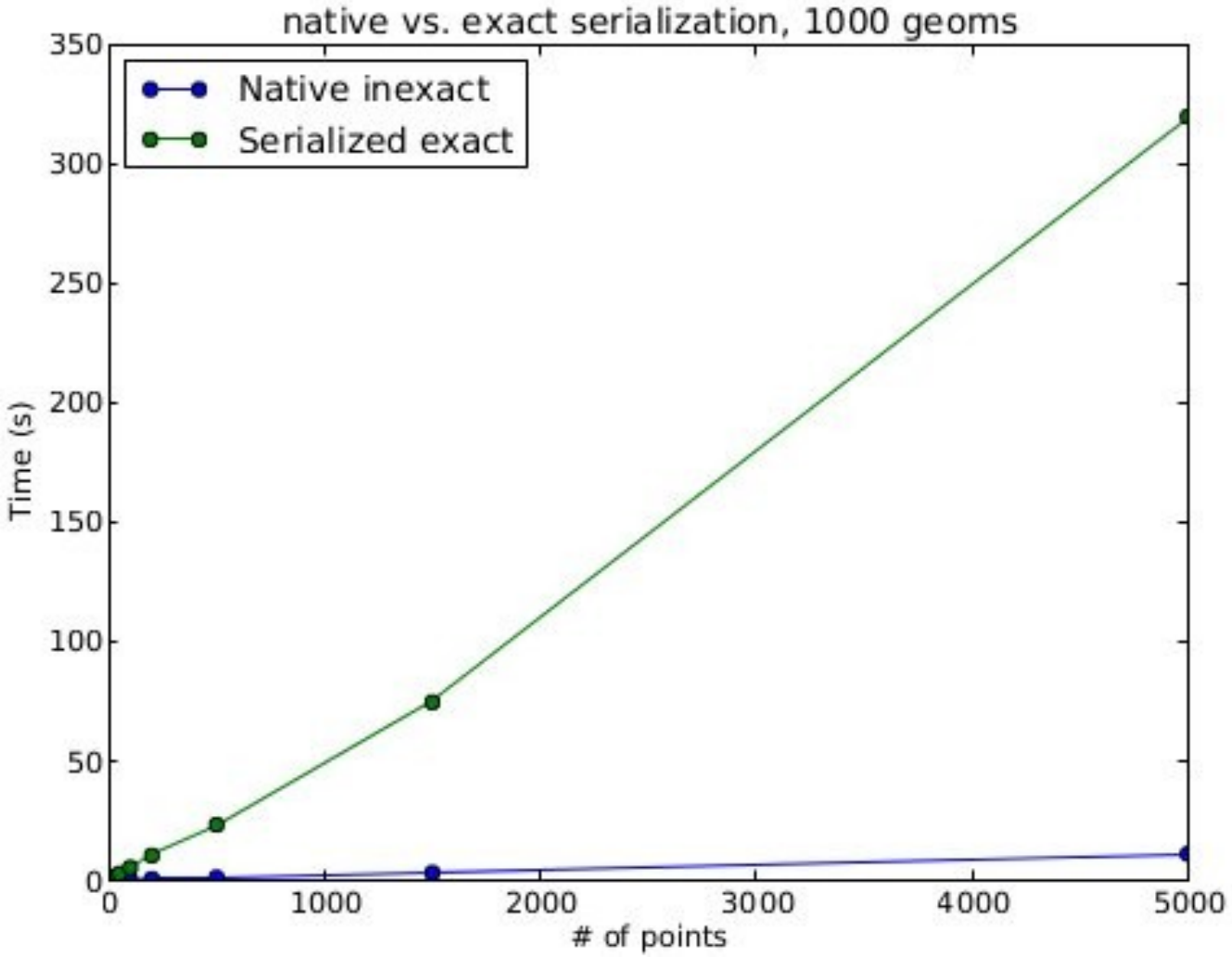
Serialization performances

'exact_geometry' serialization is slow !

Comparing 4 chained 'noop' functions

SELECT ST_Copy(ST_Copy(ST_Copy(ST_Copy(g))))

Serialization performances



Referenced geometries

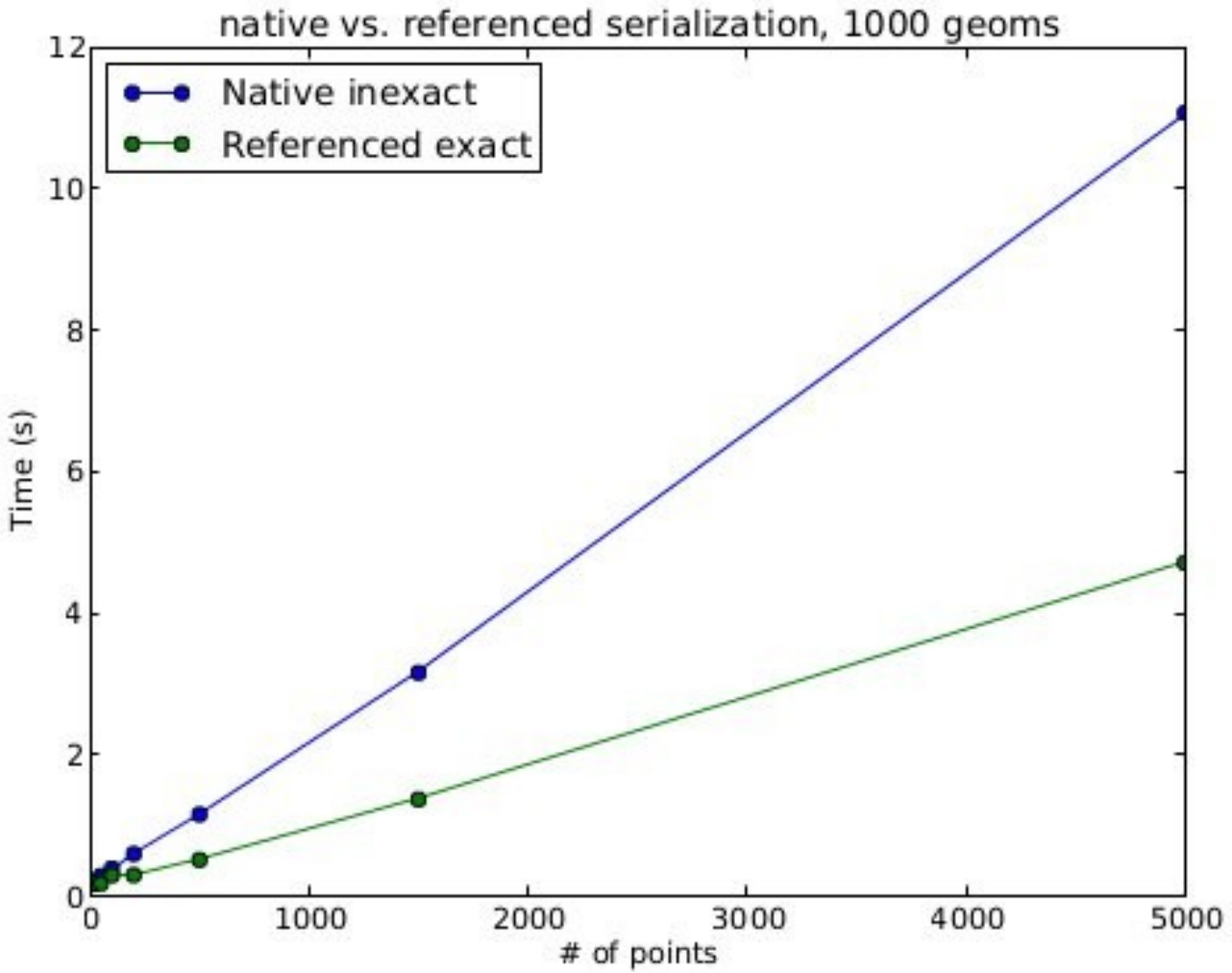
Btw, do we need to serialize ?

- Not if you only need temporary results
- ST_f1(ST_f2(ST_f3(g : geometry)))

New type 'ref_geometry'

- Complex C++ objects (SFCGAL::Geometry*) can be created and passed by reference

Referenced geometries



OSLANDIA

Referenced geometries

When to deallocate ?

Everything allocated by palloc() will be freed on Memory Context's reset/deletion

C++ objects need **destruction, not only deallocation !**

Solution

Use a child context with your own deletion method

Referenced geometries

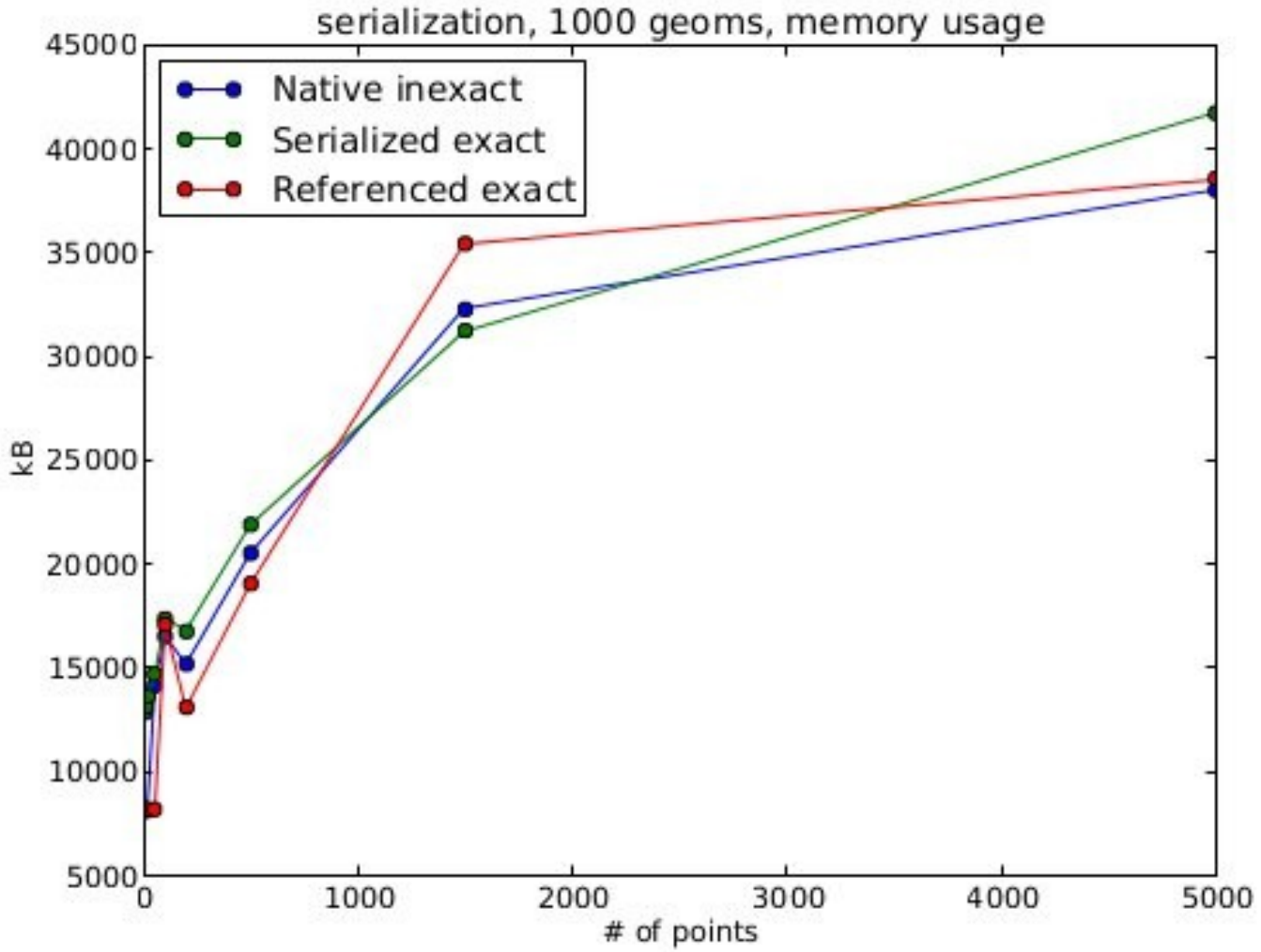
Where to allocate ?

- The parent context where to allocate is important
- If it vanishes too fast, we loose our objects
- If it lives too long, we explode memory

Current rule of thumb

- Attach to the ExprContext when we can
- Attach to a long-living context otherwise (MessageContext)

Referenced geometries



OSLANDIA

Referenced geometries

```
SELECT sfcgal.ST_Intersects(
    Sfcgal.ST_Intersection(
      'LINESTRING(0 0,2 1)'::ref_geometry,
      'LINESTRING(1 0,0 1)'::ref_geometry),
    'LINESTRING(0 0,2 1)'::ref_geometry);
 st_intersects
---------------
 t
(1 row)
```

Cannot be stored !

```
CREATE TEMPORARY TABLE t AS
  SELECT 'POINT(0 0)'::ref_geometry;

SELECT * FROM t;
NOTICE:  Referenced geometries must not be stored
 ref_geometry
--------------
 -deleted-
(1 row)
```

Referenced geometries

Control over serialization / deserialization

Through conversion functions

ST_Geometry(ref_geometry) : geometry

ST_RefGeometry(geometry) : ref_geometry

=>

SELECT **ST_Geometry**(
  ST_f1(
    ST_f2(
      ST_f3(**ST_RefGeometry(** g)
))))

Conclusion

High potential !

- 3D spatial processing
- Exact computation
- With good performances

Work in progress

PostGIS integration

Referenced geometry testing

Cache mechanism

Spatial operations (boolean set)

QGIS integration

Test and feedback

http://www.oslandia.com

On github

- Oslandia/SFCGAL
- Oslandia/postgis-sfcgal

hugo.mercier@oslandia.com