



Deutschsprachige
PostgreSQL-Konferenz
Oberhausen, Rheinisches Industriemuseum, 11.11.2011

Tuning von PostGIS mit Read-Only-Daten von OpenStreetMap

Prof. Stefan Keller
(Fach-)Hochschule für Technik
Rapperswil (bei Zürich)

Was ist OpenStreetMap?



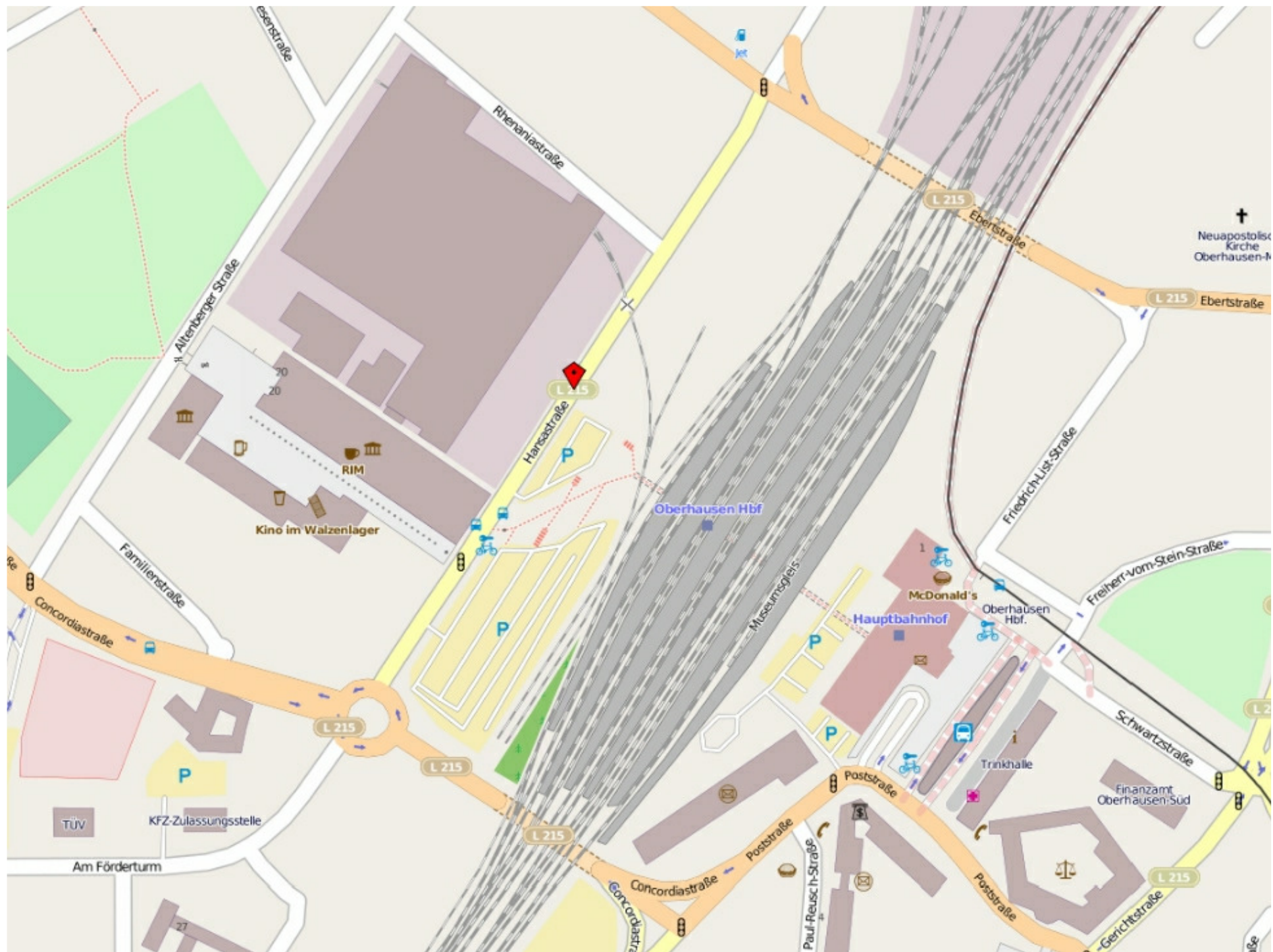
- „Wikipedia der Weltkarte“ / freie Wiki-Geodatenbank
 - CC-BY-SA-Lizenz
 - 2005 in England gestartet
 - 350.000 angemeldete Nutzer (10% aktiv)
 - 2 Mrd. GPS-Punkte
- (Quelle: F. Ramm, DGfK Stuttgart, 1.2011)



Wozu OSM? Wer nutzt OSM?



- Wozu OpenStreetMap?
 - Karten zeichnen, nutzen, drucken
 - Daten für Navigation und Analysen
 - verfügbare Kartendaten setzen Entwicklungspotential frei
- Wer nutzt OSM?
 - Hobbyprojekte, Vereine usw.
 - kleine gewerbliche Produktionen ("Apps")
 - große, internationale Anbieter (MapQuest, Bing Maps)
 - Forschung



Einige „Tags“



Schema / Format



```
<?xml version='1.0' encoding='UTF-8'?>
<osm version="0.6" generator="OpenStreetMap server">
```

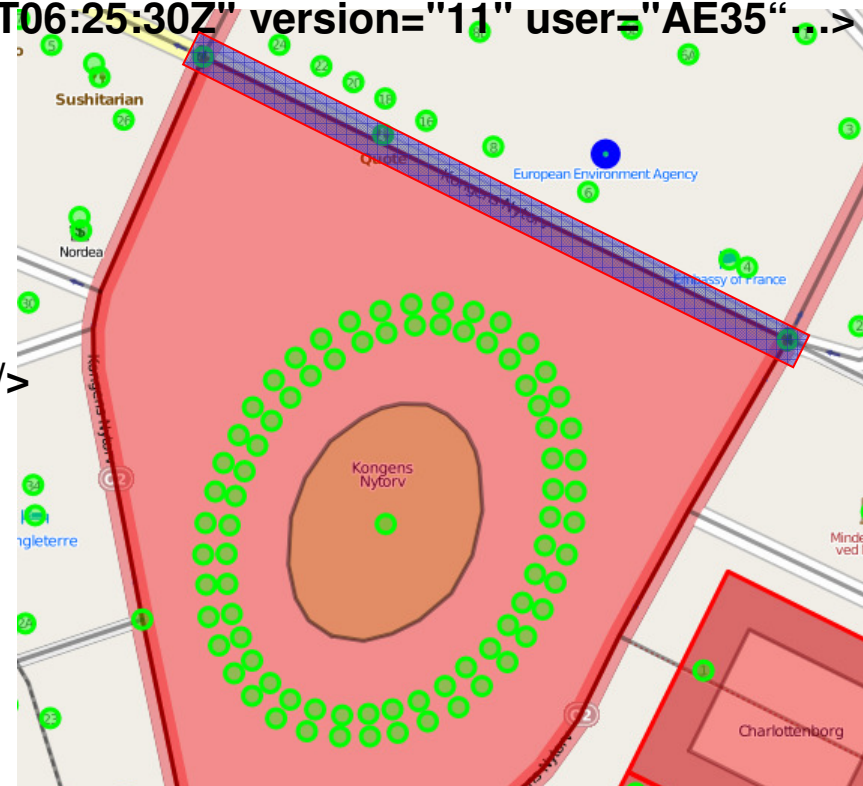
...

```
<way id="4369466" timestamp="2011-04-12T06:25:30Z" version="11" user="AE35"...>
  <nd ref="268690828"/>
  <nd ref="268690821"/>
  <tag k="cycleway" v="track"/>
  <tag k="cycleway:left" v="none"/>
  <tag k="highway" v="tertiary"/>
  <tag k="lit" v="yes"/>
  <tag k="name" v="Kongens Nytorv"/>
  <tag k="oneway" v="yes"/>
```

```
</way>
```

...

```
</osm>
```



Aus denmark.osm
11.11.2011

Datenmengen

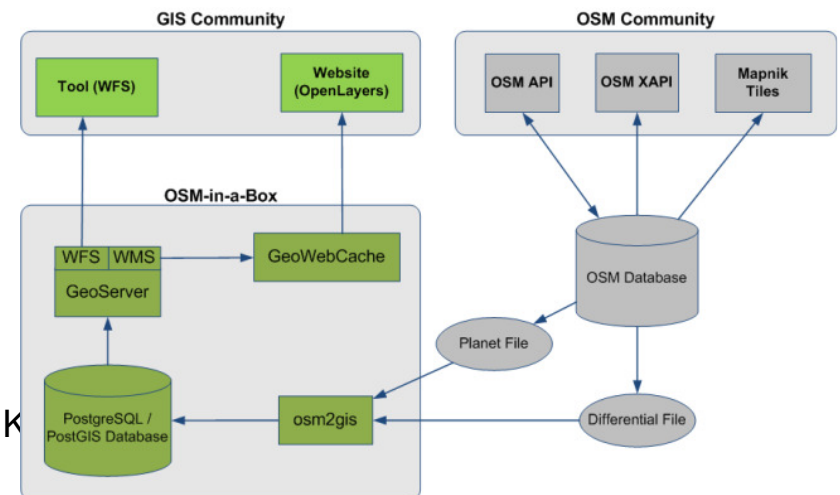


- Schweiz
 - 150 MB OSM XML bzip komprimiert
 - Point Tuples: ~ 10'000'000 (= Nodes mit Tags)
- Deutschland: 10x mehr
- Europa: 100x mehr
- Welt: 25 TB

Tool Chain



- In/Out:
 - In: Planet (=Welt) – das Original (OSM/XML)
 - Out: PostgreSQL, bzw. PostGIS
- „osm2gis“: Daten Extrahieren/Sync.
 1. „1:1“ osmosis
 2. Optimiert: osm2pgsql



Anforderungen



- Tägliche Aktualisierung, ev. stündlich
- Typische Abfrage: „Alle Zoos der Schweiz“



ca. 20

(Quelle: PostGIS Terminal)

Schema 1: Node



```
CREATE TABLE current_nodes (  
    id bigint NOT NULL PRIMARY KEY,  
    latitude integer NOT NULL,  
    longitude integer NOT NULL,  
    "timestamp" timestamp without time zone NOT NULL,  
    version bigint NOT NULL  
);
```

```
CREATE TABLE current_node_tags (  
    id bigint NOT NULL,  
    k character varying(255) DEFAULT ""::character varying NOT NULL,  
    v character varying(255) DEFAULT ""::character varying NOT NULL  
);
```

Query 1 (Schema 1)



```
SELECT id, latitude/1000, longitude/1000
FROM current_node node
JOIN current_node_tags tags
  ON node.id=tags-id
WHERE k = 'tourism' AND v = 'zoo'
AND latitude/1000 > 46.0
AND longitude/1000 > 6.1
AND latitude/1000 < 47.6
AND longitude/1000 < 9.3
```

Performance: geometry und hstore



- PostGIS
 - geometry type (abstrakt ohne weitere Ang.)
 - GiST-Index
- hstore
 - Datentyp für KV-Paare
 - GiST-Index
 - hstore vs. KVP-Tabelle
 - „Key/Value Pair versus hstore...”, Michel Ott, Juni 2011, Seminararbeit HSR.

Schema 2: Point



```
CREATE TABLE osm_point (  
  id integer, -- nullable mit btree index  
  name text, -- nullable  
  -- ...  
  -- ca. 30 Felder(!): ele, amenity, tourism, etc.  
  -- ...  
  version integer,  
  tags hstore, -- Tags als KV-Paare  
  way geometry('POINT','2','4326') -- PostGIS 2.0  
)
```

Weitere Tabellen:

- osm_line
- osm_polygon

Query 1 (Schema 2)



- ```
SELECT id, ST_AsText(way)
FROM osm_point
WHERE tags @> hstore('tourism','zoo')
```

Contains-Operator ,@>' „a contains b“:

```
field @> hstore('tag', 'value')
```



# Performance: Indexes

- `CREATE INDEX osm_point_name_idx  
ON osm_point USING btree (name)  
WITH (FILLFACTOR=100);`
- `ALTER TABLE osm_point  
CLUSTER ON osm_point_name_idx;`
- `CREATE INDEX osm_point_tags_idx  
ON osm_point USING gist (tags)  
WITH (FILLFACTOR=100);`
- `CREATE INDEX osm_point_index  
ON osm_point USING gist (way);`

# Statistik und HW



- Statistik (ca. Rohdaten 100 MB XML)
  - Table Size 1'029 MB
  - Toast Table Size 32 kB
  - Indexes Size 1'381 MB
    - osm\_point\_pkey 1029 MB
    - osm\_point\_index 1029 MB
    - osm\_point\_name\_idx 1029 MB
    - osm\_point\_tags\_idx 1029 MB
- HW
  - PostgreSQL 9.0.4
  - Ubuntu Linux 10.04 LTS (64-Bit)
  - 1 CPU Intel(R) Xeon(R) E5520 @ 2.27GHz
  - 2 GB RAM
  - 1.5 GB Swap space

# Das Problem



- Laden:
  - ca. 3h mit osm2pgsql
  - Inkl. VACUUM FREEZE ANALYZE
- Query 1:
  - 1 Anfrage > 1000 ms, alle weiteren Anfragen: < 10 ms  
vgl. PostGIS Terminal mit Query 1  
[http://labs.geometa.info/postgisterminal/?xapi=node\[tourism=zoo\]](http://labs.geometa.info/postgisterminal/?xapi=node[tourism=zoo])
- Query 2:
  - Mit UNION
- Query 3:
  - Anfragen mit PostGIS-Funktionen wie ST\_Intersection echt langsam
- I/O- oder CPU-Bottleneck?

# Ursachen schlechter Performance



- Falsche (oder fehlende) Konfiguration
- Anwendung stellt Anfragen falsch
- Logik in der Applikation statt in der Datenbank
- Datenbank-Layout ist falsch oder ungenügend (fehlende Indexes, fehlende Normalisierung)
- Anwendungen fragen zu viele (unnötige) Daten ab (z.B.: `SELECT *`, fehlendes `WHERE`, fehlendes `LIMIT`)
- Latenzzeiten (z.B. Netzwerk, Zugriffszeiten Festplatte)
- Ungenügende Hardware

Aus: "PostgreSQL optimieren", Chemnitzer Linuxtage 2009, Andreas Kretschmer und Andreas Scherbaum

# Konfiguration postgresql.conf



pgtune wizard:

shared\_buffers = 128MB  
#shared\_buffers = 24MB # min 128MB

effective\_cache\_size = 352MB  
#effective\_cache\_size = 128MB

work\_mem = 3MB # pgtune wizard 2011-05-06  
#work\_mem = 1MB # min 64kB

wal\_buffers = 8MB # pgtune wizard 2011-05-06  
#wal\_buffers = 64kB # min 32kB

checkpoint\_segments = 16 # pgtune wizard 2011-05-06  
#checkpoint\_segments = 3 # in logfile segments, min 1, 16MB each

# EXPLAIN ANALYZE



EXPLAIN ANALYZE  
<Query 1>:

Bitmap Heap Scan on osm\_point  
  (cost=322.61..30756.62 rows=9800 width=104)  
  (actual time=24.107..26.664 rows=20 loops=1)  
    Recheck Cond: (tags @> 'tourism=>zoo'::hstore)  
      -> Bitmap Index Scan on osm\_point\_tags\_idx  
          (cost=0.00..320.16 rows=9800 width=0)  
          (actual time=15.988..15.988 rows=593 loops=1)  
          Index Cond: (tags @> 'tourism=>zoo'::hstore)  
Total runtime: 26.845 ms

- Recheck Cond verschätzt sich ziemlich: 9800 statt effektiv 20

# Massnahmen?



- Planner-Statistik
  - ALTER TABLE <table> ALTER COLUMN <column>  
SET STATISTICS <number>;
- HW
  - Mehr Memory?
    - ?
  - Schnellere Sekundärspeicher?
    - SSD
    - NVRAM PCards

# Kontakt



Stefan Keller  
sfkeller(at)hsr.ch  
Twitter sfkeller  
[www.gis.hsr.ch](http://www.gis.hsr.ch)  
[www.postgres-support.de](http://www.postgres-support.de)  
[www.postgis.ch](http://www.postgis.ch)

