
ORACLE®

JavaOne™

ORACLE®

Big Data on Big Maps

Displaying Vast Amounts of Geospatial Data

Roberto Mercado
Héctor Alejandro Saucedo Briseño
LJ Qian
Oracle Spatial and Graph
October 4, 2017

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Session Agenda

1. Big Data on Big Maps
2. Planet Dataset
3. Preparing the Data
4. Rendering
5. Summary

JavaOne ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Session Agenda

1. Big Data on Big Maps
2. Planet Dataset
3. Preparing the Data
4. Rendering
5. Summary

JavaOne ORACLE

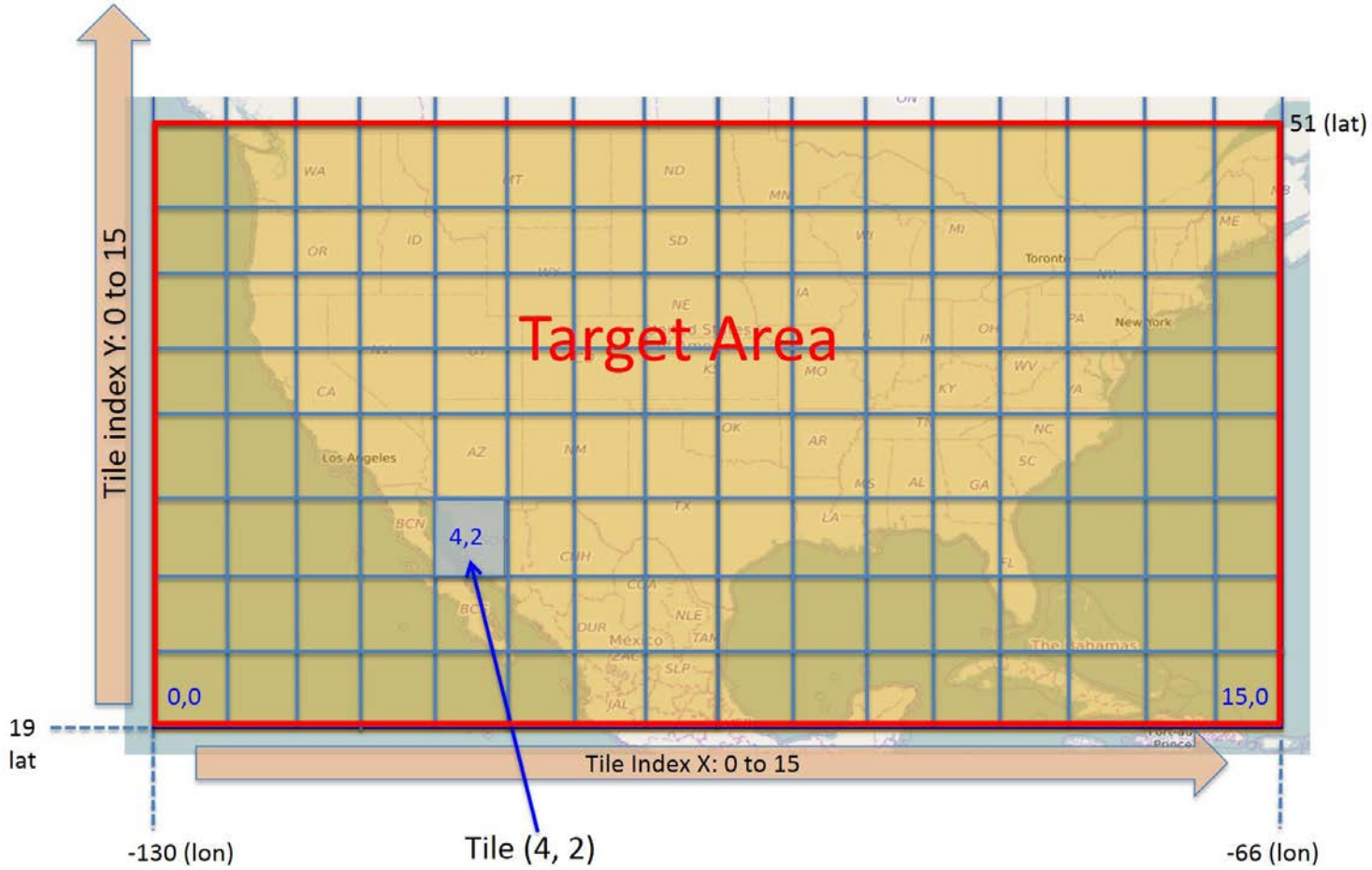
Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Big Data on Big Maps

- Goal: Render every street, road, route, and interstate highway
- Input: OpenStreetMap planet file
- Output: PNG image (57600x28800 pixels, or 4x8 ft)
  - Printable resolution for a wall-sized banner
- Target area: Defined bounding box, e.g. contiguous USA region
- Divide target area into smaller “tiles” (square areas)
- The application should create different tile sizes and resolutions based on the desired output image size and bounding region to render

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Target area



- Area is divided into smaller tiles
- Each tile is 4 by 4 degrees
- Target area is 64 longitude degrees by 32 latitude degrees

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | Confidential – Oracle Internal/Restricted/Highly Restricted

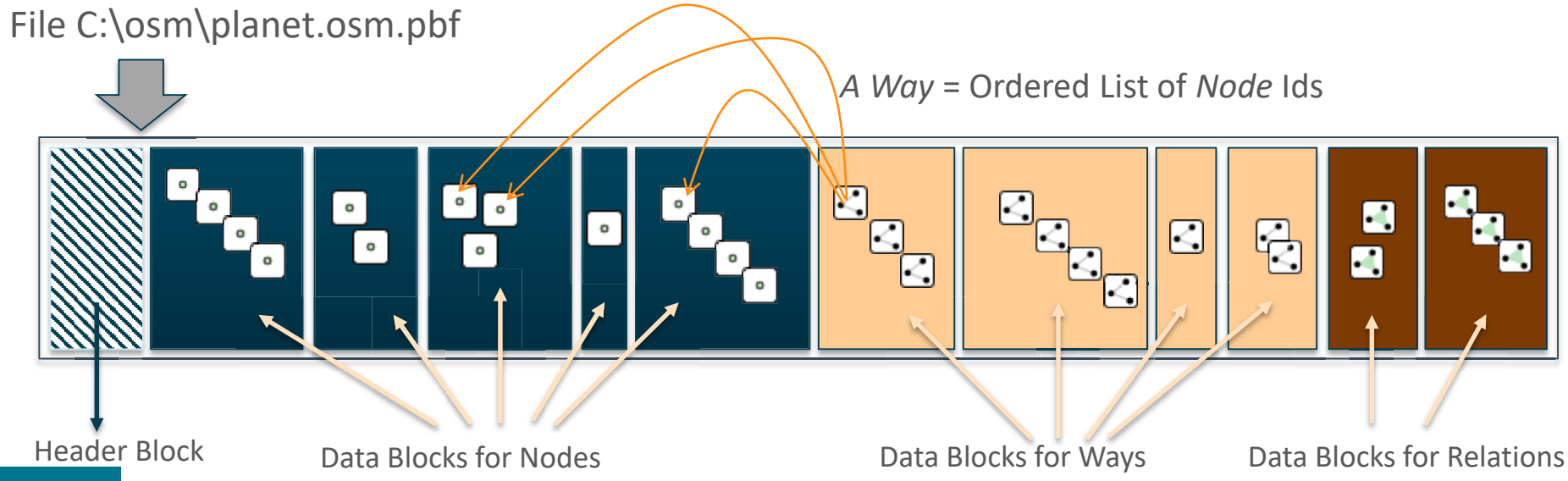
Session Agenda

1. Big Data on Big Maps
2. Planet Dataset
3. Preparing the Data
4. Rendering
5. Summary

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Planet Dataset

- OpenStreetMap planet dataset (.pbf) is 36GB
  - Protocolbuffer Binary Format is a compressed file format, about half of the size of a gzipped planet. It's about 5x faster to write and read.
  - It supports random access at the file-block granularity. Each file-block is independently decodable and contains ~8k OSM entities.



Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Example OSM XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
    <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="12.2524800"/>
    <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636“
        timestamp="2008-09-21T21:37:45Z"/>
    <node id="261728686" lat="54.0906309" lon="12.2441924" user="PikoWinter" uid="36744" visible="true" version="1" changeset="323878"
        timestamp="2008-05-03T13:39:23Z"/>
    <node id="1831881213" version="1" changeset="12370172" lat="54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible="true"
        timestamp="2012-07-20T09:43:19Z">
        <tag k="name" v="Neu Broderstorf"/>
        <tag k="traffic_sign" v="city_limit"/>
    </node>
    ...
    <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO" uid="46882" visible="true" version="1" changeset="676636"
        timestamp="2008-09-21T21:37:45Z"/>
    <way id="26659127" user="Masch" uid="55988" visible="true" version="5" changeset="4142606" timestamp="2010-03-16T11:47:08Z">
        <nd ref="292403538"/>
        <nd ref="298884289"/>
        ...
        <nd ref="261728686"/>
        <tag k="highway" v="unclassified"/>
        <tag k="name" v="Pastower Straße"/>
    </way>
    <relation id="56688" user="kmvar" uid="56190" visible="true" version="28" changeset="6947637" timestamp="2011-01-12T14:23:49Z">
        <member type="node" ref="294942404" role=""/>
        ...
        <member type="node" ref="364933006" role=""/>
        <member type="way" ref="4579143" role=""/>
        ...
        <member type="node" ref="249673494" role=""/>
        <tag k="name" v="Küstenbus Linie 123"/>
        <tag k="network" v="VVW"/>
        <tag k="operator" v="Regionalverkehr Küste"/>
        <tag k="ref" v="123"/>
        <tag k="route" v="bus"/>
        <tag k="type" v="route"/>
    </relation>
    ...
</osm>
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Session Agenda

1. Big Data on Big Maps
2. Planet Dataset
3. **Preparing the Data**
4. Rendering
5. Renderer

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Preparing the Data – Osmosis and Hadoop

- Data is read in file-blocks from .pbf file using Osmosis
- File-blocks are transferred to HDFS
- HDFS file-blocks are joined and preprocessed to get Way instances
- Ways are used to obtain geometries and their tags
- Relations and Nodes are filtered out
- JGeometry and Map<String,Object> are created using the Way's coordinates and the Way's tags respectively.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

.pbf file-blocks to HDFS

```
SequenceFile.Writer out = SequenceFile.createWriter(conf,
        SequenceFile.Writer.file(fOut),
        SequenceFile.Writer.keyClass(Text.class),
        SequenceFile.Writer.valueClass(ArrayPrimitiveWritable.class));

Text k = new Text();
ArrayPrimitiveWritable v = new ArrayPrimitiveWritable();

while (fileIn.available() > 0) {
        int headerLength = fileIn.readInt();
        BlobHeader blobHeader = readHeader(headerLength, fileIn);
        byte[] blobData = readRawBlob(blobHeader, fileIn);

        k.set(blobHeader.getType());
        v.set(blobData);
        out.append(k, v);
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Join file-blocks to corresponding Way

```
Job job = Job.getInstance(conf);
job.setMapperClass(JoinMapper.class);
job.setReducerClass(JoinReducer.class);
job.setInputFormatClass(
        org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat.class);
job.setMapOutputKeyClass(LongWritable.class);
job.setMapOutputValueClass(OSMGenericWritable.class);
job.setOutputKeyClass(LongWritable.class);
job.setOutputValueClass(OSMGenericWritable.class);

FileInputFormat.addInputPath(job, new Path(fInput));
job.setOutputFormatClass(
        org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat.class);
FileOutputFormat.setOutputPath(job, fOut);

job.waitForCompletion(true);
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Build WayWritable

```
Job job = Job.getInstance(conf);
job.setMapperClass(BuildMapper.class);
job.setReducerClass(BuildReducer.class);
job.setInputFormatClass(
       org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat.class);
job.setMapOutputKeyClass(LongWritable.class);
job.setMapOutputValueClass(OSMGenericWritable.class);
job.setOutputKeyClass(LongWritable.class);
job.setOutputValueClass(WayWritable.class);

FileInputFormat.addInputPath(job, new Path(fInput));
job.setOutputFormatClass(
       org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat.class);
FileOutputFormat.setOutputPath(job, fOut);

job.waitForCompletion(true);
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Preparing the Data - Spark

- Data is a spatially-partitioned RDD
- Each output cell (tile) is a partition
- Spatial data (JGeometry) is accessible

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Filtering and partitioning

```
JavaPairRDD<LongWritable, WayWritable> pairWayRDD = sc.newAPIHadoopFile(
        srcFile,
        SequenceFileInputFormat.class,
        LongWritable.class,
        WayWritable.class,
        sc.hadoopConfiguration());

SpatialJavaRDD<WayWritable> wayRDD = SpatialJavaRDD.fromJavaRDD(
        pairWayRDD.map(t->{return new WayWritable(t._2());}),
        new WayWritableRecordInfoProvider(8307),
        WayWritable.class);

JGeometry gridGeom = JGeometry.createLinearPolygon(
        new double[]{gridMBR[0], gridMBR[1], gridMBR[0], gridMBR[3], gridMBR[2],
                gridMBR[3],gridMBR[2], gridMBR[1], gridMBR[0], gridMBR[1] },
        2, 8307);
SpatialOperationConfig spatialOpConf = new SpatialOperationConfig(SpatialOperation.AnyInteract, gridGeom,
        spatialConf.getTolerance());
SpatialJavaRDD<WayWritable> filteredSpatialRDD = wayRDD.filter(null, spatialOpConf);

List<SpatialPartition> partitions = GridPartitioning.generateGridPartitions(gridMBR, cellWidth,
        cellHeight, spatialConf);

JavaPairRDD<PartitionKey, WayWritable> partRDD = GridPartitioning.partition(filteredSpatialRDD,
        partitions);

GridPartitioning.savePartitionedRDD(partRDD, partitions, WayWritable.class, destFile,
        sc.hadoopConfiguration());
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Session Agenda

1. Big Data on Big Maps
2. Planet Dataset
3. Preparing the Data
4. Rendering
5. Summary

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Renderer

- Spark transforms the data into a spatial-aware RDD
- Each partition (cell) gets rendered to a PNG file
- RDD aggregation action to generate the full image

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Load grid partitions to render

```
GridPartitionedRDDReader<WayWritable> reader = GridPartitioning.createReader(srcFile,
        WayWritable.class, sc);
JavaPairRDD<PartitionKey, WayWritable> partRDD = reader.getPartitionedRDD();
Map<Integer, SpatialPartition> partMap = reader.getSpatialPartitionsMap();

SimpleRenderer renderer = new SimpleRenderer();

JavaRDD<ImageInfo> imagesRDD = partRDD.mapPartitionsWithIndex(
        (pIndex, iterator)->{
                SpatialPartition part = partMap.get(pIndex);
                ImageInfo image = renderer.render(part, new ValueIterator<>(iterator));
                return Collections.singletonList(image).iterator();
        },
        true);

ImageInfo zeroImage = new ImageInfo(0, mbr); //Empty image

ImageInfo fullImage = imagesRDD.aggregate(
        zeroImage,
        (aggrImage, image)->{
                return renderer.aggregateImages(aggrImage, image);
        },
        (aggrImage1, aggrImage2)->{
                return renderer.combineImages(aggrImage1, aggrImage2);
        });
ImageFileUtil.saveBytesToFile(destFile, fullImage.getImageBytes());
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Render images

```
public ImageInfo render(SpatialPartition partition, Iterator<WayWritable> records) {
        OOWTile tile = new OOWTile(partition.getMbr());
        TileRenderingContext tc = new TileRenderingContext(tile);

        ImageInfo image = new ImageInfo(partition.index(), partition.getMbr());
        while (records.hasNext()) {
                WayWritable info = records.next();
                Map<String, String> tags  = info.getTagMap();

                double[] xys = info.getGeom().getOrdinatesArray();
                WorldMercatorUtils.lonLatToMeters(xys);
                tc.renderLineString(xys, tile.getMbrMercator());

        }
        byte[] imageBytes = tc.saveToBuffer();
        image.setImageBytes(imageBytes);
        return image;
}
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

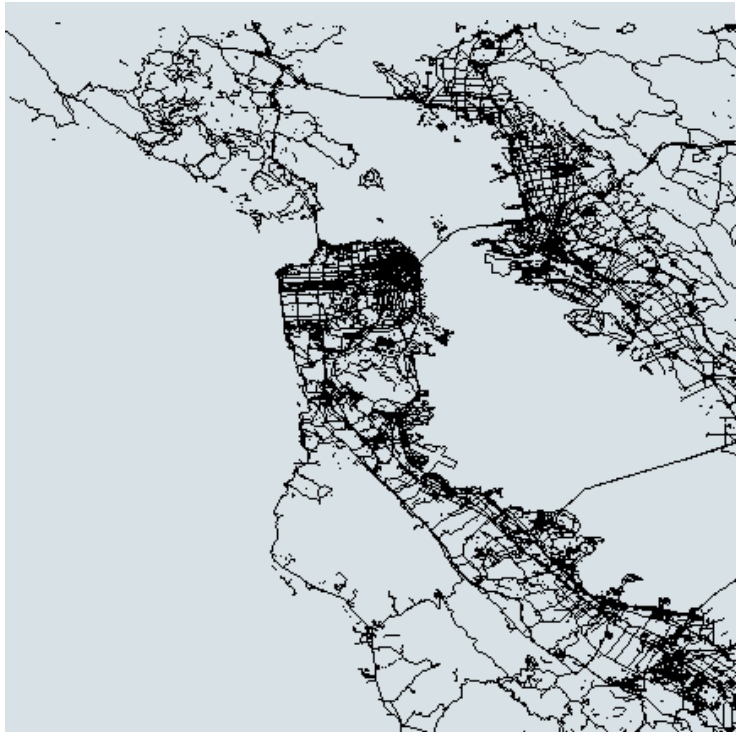
Render of all the roads in United States

Every street, road, route, and highway was scaled and rendered to fit in the image

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Whole image vs Tile

Whole image combined



San Francisco tile

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Session Agenda

1. Big Data on Big Maps
2. Planet Dataset
3. Preparing the Data
4. Rendering
5. Summary

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Summary

10 minutes to render

4.5

Billion records in OSM planet file

United States region filtered out and ready for rendering in less than 30 minutes

3-machine Cluster

1 driver, 2 executors

1 core, 12 CPUs x86_64

62 GB RAM

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

Discover more at:

- Planet dataset:
  - http://wiki.openstreetmap.org/wiki/Planet.osm
- OSM-PBF Format
  - http://wiki.openstreetmap.org/wiki/PBF_Format
- Oracle Big Data Spatial and Graph OTN
  - http://www.oracle.com/technetwork/database/database-technologies/bigdata-spatialandgraph/overview/index.html
- JGeometry
  - https://docs.oracle.com/database/121/SPAJV/oracle/spatial/geometry/JGeometry.html
- Oracle Big Data Spatial and Graph documentation
  - http://docs.oracle.com/bigdata/bda49/index.htm
- Osmosis
  - http://wiki.openstreetmap.org/wiki/Osmosis

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. | Confidential – Oracle Internal/Restricted/Highly Restricted

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. |

JavaOne™

ORACLE®


ORACLE®