Osmium to the Rescue
Solving OSM Problems with Osmium

Jochen Topf

Osmium has Something for Everybody

libosmium (C++ Header Library)

osmium (Command Line Tool)

node-osmium (Javascript, NodeJS)

Background

Osmium in development
since October 2010.

„New Osmium“ for about a year now.

CC-BY http://www.flickr.com/photos/jronaldlee/4479381576/

Has to work with data of entire planet!

...or a small extract!

New Osmium

Has all functionality of the old Osmium,
and much more...

Needs tests, some polishing and documentation.

Start using it now!
(If you can use C++11)

Concatenate / Format conversion

```
osmium cat \
  berlin.osm.pbf \
  -o berlin.osm.bz2
```

Input/Output

Read from: file, stdin or URL.
Write to: file or stdout.

XML or PBF or OPL.
Compressed or uncompressed.
OSM data (.osm) or changes (.osc).
With or without history.

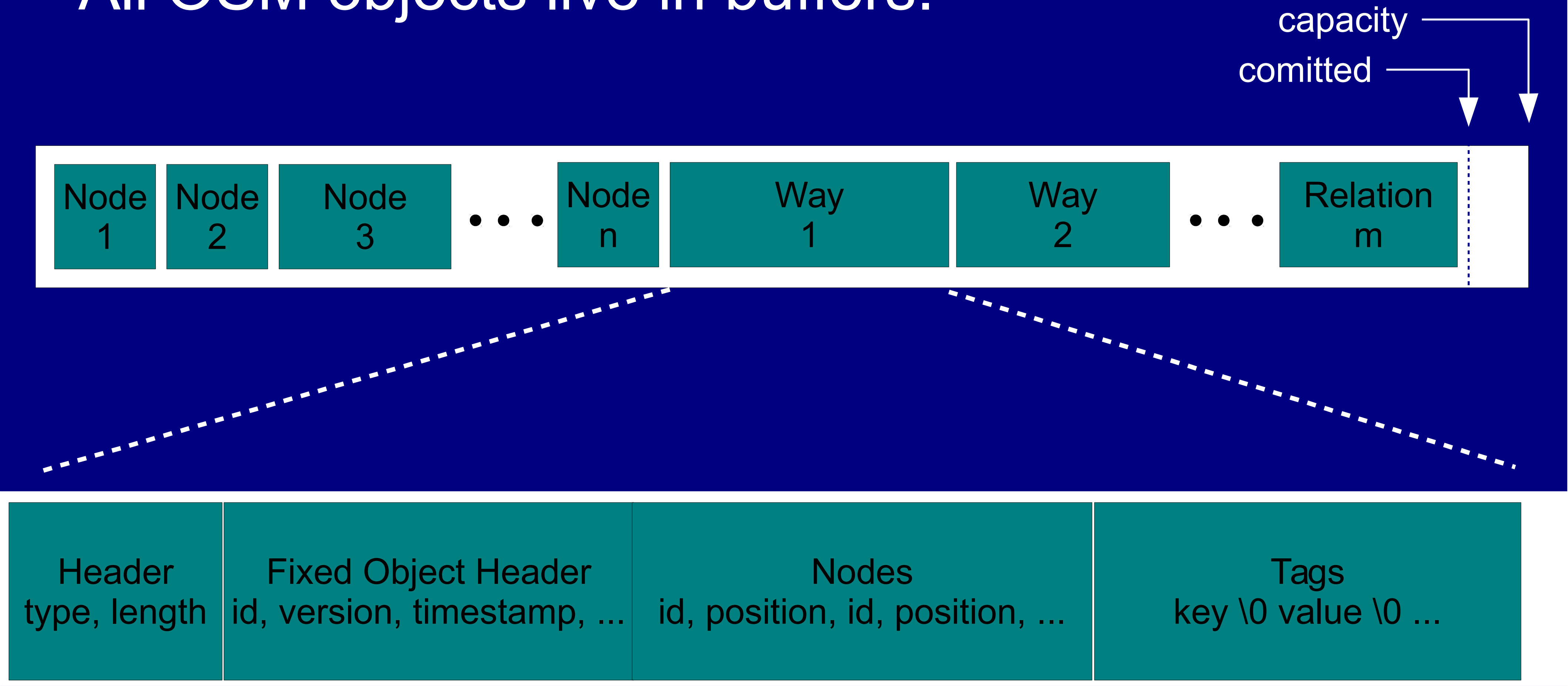
C++: File Converter

osmium::io::Reader

```
Reader reader("berlin.osm.pbf");
Header header = reader.header();
header.set("generator", "MyConverterProgram");

Writer writer("berlin.osm.bz2", header);

while (Buffer buffer = reader.read()) {
    writer(std::move(buffer));
}

writer.close();
```

Buffers

All OSM objects live in buffers.



capacity

comitted

Node 1

Node 2

Node 3

• • •

Node n

Way 1

Way 2

• • •

Relation m

Header
type, length

Fixed Object Header
id, version, timestamp, ...

Nodes
id, position, id, position, ...

Tags
key \0 value \0 ...

Buffers

Objects can't change size. More or less const.

Need special Builder objects
to build OSM objects in buffers.

Buffer content can only be freed all at once.

Fixed buffers and auto-resizing buffers.

Buffers

Less memory management overhead.

Easier multithreading.
I/O is multithreaded.

Easy to write buffers to disk/sockets.

Updating OSM (History) File

```
osmium apply-change \
  old.osm.pbf \
  change1.osc.gz \
  change2.osc.gz \
  -o new.osm.pbf
```

Data/Changes/Historic

OSM Data/OSM Changes/With History
All handled the same way

Different than Osmosis!

Many things just work
out-of-the-box with historic data.

Extract from OSM History File

```
osmium time-filter \
  history-planet.osh.pbf \
  2008-01-01T00:00:00Z \
  -o planet-20080101.osm.pbf
```

Concatenate / Format conversion

```
osmium cat \
  berlin.osm.pbf \
  -o berlin.osm.opl
```

OPL Format

```
n2909896397 v1 dV c22858411 t2014-06-10T20:24:55Z
  i34845 u4Ems
  T
  x13.2625290 y52.4441154

w4045150 v33 dV c20465397 t2014-02-09T14:02:12Z
  i7295 uelgolfo
  Thighway=residential,maxspeed=30,
   name=Waldstraße,surface=concrete
  Nn1234120411,n262876417,n262877047,n21432146,n262146912

r3798972 v1 dV c22753667 t2014-06-05T11:18:57Z
  i1879543 uglibbertorsten
  Tbuilding=yes,type=multipolygon
  Mw39293991@outer,
   w39293994@inner,w39293992@inner,w39293993@inner
```

OPL Format

```
n2909896397 v1 dV c22858411 t2014-06-10T20:24:55Z
  i34845 u4Ems
  T
  x13.2625290 y52.4441154

w4045150 v33 dV c20465397 t2014-02-09T14:02:12Z
  i7295 uelgolfo
  Thighway=residential,maxspeed=30,
   name=Waldstraße,surface=concrete
  Nn1234120411,n262876417,n262877047,n21432146,n262146912

r3798972 v1 dV c22753667 t2014-06-05T11:18:57Z
  i1879543 uglibbertorsten
  Tbuilding=yes,type=multipolygon
  Mw39293991@outer,
   w39293994@inner,w39293992@inner,w39293993@inner
```

OPL Format

```
grep amenity=post_box \
     berlin.osm.opl \
  | sort -k2.3,2n \
  | tail -1
```

OPL Format

```
n308817106
v21
dV
c12281993
t2012-07-18T12:32:28Z
i722137
uOSMF%0020Redaction%0020Account
Tamenity=post_box,
 collection_times=Mo-Fr...,
 operator=Deutsche%0020Post,
 postal_code=10439,
 ref=Schivelbeiner%0020Straße%002024-25
x13.4048585
y52.5511278
```

OPL Format (with History)

```
grep '^n308817106 ' \
     berlin.osh.opl \
  | cut -d' ' -f2,9-
```

Postal Box Versions

```
v1 x13.4045280 y52.5512085
v2 x13.4045280 y52.5511876
v3 x13.4043578 y52.5512087
v4 x13.4048613 y52.5511425
v5 x13.4048613 y52.5511425
v6 x13.4046862 y52.5511710
v7 x13.4046975 y52.5511400
v8 x13.4046975 y52.5511400
v9 x13.4046975 y52.5511400
v10 x13.4045205 y52.5511723
v11 x13.4046426 y52.5511656
v12 x13.4048509 y52.5511653
v13 x13.4048585 y52.5511474
v21 x13.4048585 y52.5511278
```

XAPI with OPL

```
osmium cat -F osm -f opl \
    'http://...xapi_meta?way[bridge=yes][bbox=...]' \
  | egrep '[T,]name=' \
  | cut -d' ' -f8 \
  | sed -e 's/.*[T,]name=\([^,]*\)\(,.*\|$\)/\1/' \
  | sort -u
```

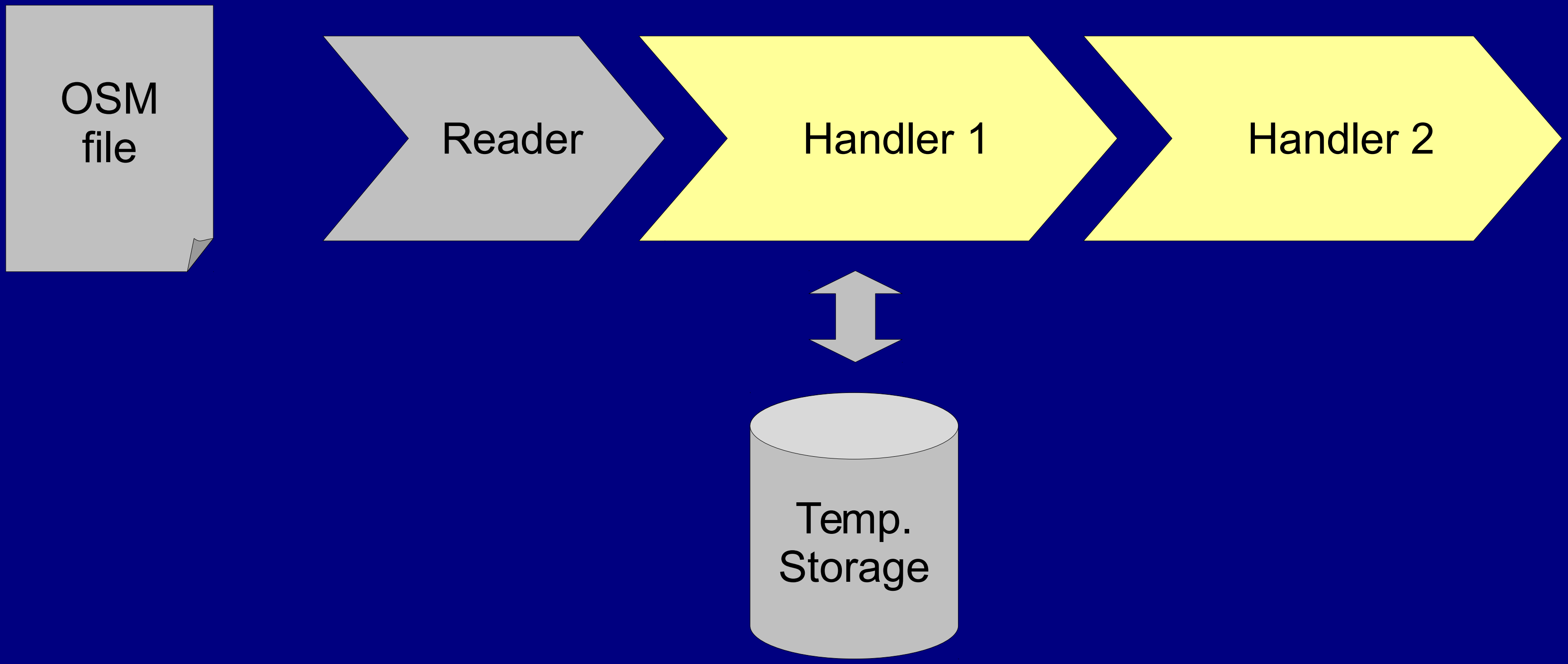
C++ Example: Iterator

```
Reader reader("berlin.osm.pbf");

auto begin = InputIterator<Reader, Object> {reader};
auto end   = InputIterator<Reader, Object> {};

for (auto it = begin; it != end; ++it) {
  ...
}
```

C++ Example: Iterator

```cpp
Reader reader("berlin.osm.pbf");

auto begin = InputIterator<Reader, Object> {reader};
auto end   = InputIterator<Reader, Object> {};

Writer writer("berlin.osm.bz2");
OutputIterator<Writer> out(writer);

std::copy(begin, end, out);
```

Handler



I/O Example: Handler

```
class MyHandlerClass : public osmium::handler::Handler
{

public:

  void node(Node& node) {
    ...
  }

  void way(Way& way) {
    ...
  }

  ...
};
```

I/O Example: Handler

```
Reader reader("berlin.osm.pbf");

MyHandlerClass handler;

osmium::apply(reader, handler);
```

Example:
Compute Road Length

https://github.com/osmcode/osmium-contrib/tree/master/road_length

Road Length: handler

```
#include <osmium/handler.hpp>
#include <osmium/geom/haversine.hpp>

struct RoadLengthHandler : public Handler {

 double length=0;

 void way(const osmium::Way& way) {
   if (way.tags().get_value_by_key("highway")) {
    length +=
      osmium::geom::haversine::distance(way.nodes());
   }
 }

};
```

Road Length: Node Locations

```
#include <osmium/index/map/sparse_table.hpp>

typedef
  SparseTable<unsigned_object_id_type, Location>
  index_type;


#include <osmium/handler/node_locations_for_ways.hpp>

typedef
  NodeLocationsForWays<index_type>
  location_handler_type;
```

Node Length: main

```
#include <osmium/io/any_input.hpp>
#include <osmium/visitor.hpp>
#include <iostream>

int main(int argc, char* argv[]) {
  Reader reader(argv[1]);

  index_type index;
  location_handler_type location_handler(index);

  RoadLengthHandler handler;

  apply(reader, location_handler, handler);

  std::cout << handler.length / 1000 << " km\n";
}
```

Geometry

Add node locations to ways

Assemble Multipolygons

Line length (haversine)

Convert geometries to WKT, WKB, OGR, GEOS

NodeJS Module

```
npm install osmium
```

NodeJS Module

```
var osmium = require('osmium');

var location_handler = new osmium.LocationHandler(),
    handler          = new osmium.Handler();

handler.on('way', function(way) {
    console.log(way.wkt());
});

var reader = new osmium.Reader(process.argv[2]);
reader.apply(location_handler, handler);
```

C++: Areas/OGR Example

```
osmium::area::Collector<Assembler> collector;


Reader reader1("berlin.osm.pbf");
collector.read_relations(reader1);
```

C++: Areas/OGR Example

```
index_type index;
location_handler_type location_handler(index);

MyHandler myhandler;

Reader reader2("berlin.osm.pbf");

apply(reader2,
      location_handler,
      myhandler,
      collector.handler(
      [&myhandler](const Buffer& area_buffer) {
          apply(area_buffer, myhandler);
      })
);
```

C++: Areas/OGR Example

```
struct MyHandler : public Handler {
  osmium::geom::OGRFactory m_factory;
  OGRLayer* m_layer;

  void area(const Area& area) {
    OGRFeature* feature =
      CreateFeature(m_layer->GetLayerDefn());

    feature->SetGeometry(
                 m_factory.create_multipolygon(area));

    m_layer->CreateFeature(feature);
  }

};
```

Changes from Old Osmium

Now uses C++11.

Many function signatures and other details have changed.

Namespaces are lower case.

Name changes (example: Position → Location).

Changes from Old Osmium

OSM Objects in Buffers.

Memory management through buffers.

Indexes and Maps.

Multithreaded I/O.

Multipolygon assembly without GEOS.

osmjs → NodeJS module.

Thanks!

GEOFABRIK

MapBox

THE END

osmcode.org
github.com/osmcode

libosmium, node-osmium, osmium-tool,
osmium-contrib, osm-testdata, libosmium-manual

Jochen Topf
jochen@topf.org
jochentopf.com