Parsing OSM XML
Kushan Joshi
Frontend Developer

mapbox
kepta
kushan2020
Breaking down iD
Breaking down iD

A sample of what iD is doing behind the scenes
Breaking down JS

- Draw Vector: 42%
- GC: 13%
- DOM Selector: 27%
- Parse XML: 11%
- Others: 8%
Parsing XML eats up 25% of CPU Time
Solutions?

- Stop parsing XML?
Solutions?

• **Stop** parsing XML?
Solutions?

- Stop parsing XML?
- Use clever techniques to **defer** the parsing of XML
Solutions?

• Stop parsing XML?

• Use clever techniques to defer the parsing of XML
Solutions?

- Stop parsing XML?
- Use clever techniques to defer the parsing of XML
- Exploit the **multi-core** architecture of CPUs
Solutions?

- Stop parsing XML?
- Use clever techniques to defer the parsing of XML
- Exploit the multi-core architecture of CPUs
The Multi-threading Solution

- Even the cheapest smartphone has at least **two core** for processing.

- Unlike UI, parsing can be easily offloaded to a **separate thread**.

- This will improve the **response time** and alleviate some of the pressure on main thread.
Browsers and Threads

- Browser threads a.k.a web-workers do not support DOM/XML
Browsers and Threads

- Browser threads a.k.a web-workers do not support DOM/XML

- iD wasn’t written to be executed in a multi-threaded environment
Osm-Bizli

- To circumvent the problem of not having DOM, I created a new library called **Osm-Bizli**.

- It relies on string parsing of XML line by line.

- It only understands the **particular XML** returned by the bbox API.

- With these focused features, it is able to deliver impressive performance.
<table>
<thead>
<tr>
<th>Parsers</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>osm-bizli</td>
<td>0s, 41.30629ms</td>
</tr>
<tr>
<td>osmium</td>
<td>0s, 50.834623ms</td>
</tr>
<tr>
<td>osm-bizli (node)</td>
<td>0s, 56.342906ms</td>
</tr>
<tr>
<td>iD xml-parser</td>
<td>0s, 126.757749ms</td>
</tr>
<tr>
<td>osmtogeojson</td>
<td>0s, 156.496379ms</td>
</tr>
</tbody>
</table>
How does it work?

We parse the XML line by line.
How does it work?

For each **Entity** we create a corresponding blank object.
How does it work?

The attributes are populated.
How does it work?

```xml
<way id="226519199" uid="1233206">
  <nd ref="2353129101"/>
  <nd ref="2353129111"/>
  <nd ref="2353129105"/>
  <nd ref="2353129096"/>
  <nd ref="2353129101"/>
  <tag k="building" v="yes"/>
</way>
```
How does it work?

```xml
<way id="226519199" uid="1233206">
    <nd ref="2353129101"/>
    <nd ref="2353129111"/>
    <nd ref="2353129105"/>
    <nd ref="2353129096"/>
    <nd ref="2353129101"/>
    <tag k="building" v="yes"/>
</way>
```

We then move to next line
How does it work?

If the line starts with `<nd` or `<tag`
we fill our `way` object with it
How does it work?

Whenever we encounter a closing tag, we save the object and start fresh with new line.
Takeaways
Takeaways

• Multi-threading would improve the performance of iD

• Osm-bizli uses string parsing of OSM-xml to overcome the limitations of web-workers.

• Opens future possibility of offloading more tasks to web-worker.
Thanks