CityJSON: does (file) size matter?

The compression of 3D city models in the CityJSON encoding

Jordi van Liempt
MSc Geomatics P5 presentation

Hugo Ledoux
Balázs Dukai
Ken Arroyo Ohori
Willem Korthals Altes

Contents

- Introduction (motivation and CityJSON)
- Theory/related work on compression
- Methodology
- Benchmark results
- Conclusion/discussion

Introduction: motivation

- Possibilities 3D city models increasingly explored
- Geoinformation on the web is popular
- Files can become massive, network speed can be a bottleneck
- → Therefore: compression of CityJSON

Building

Edit

b11267a1d-00ba-11e6-b420-2bdcc4ab5d7f

20 Attributes | 1 Geometries

bgt_status	bestaand
bronhouder	G0503
creationdate	2014-07-09
eindregistratie	
hoek	(1:32.7)
identificatiebagpnd	503100000004048
identificatiebagvbohoogstehuisnummer	
identificatiebagvbolaagstehuisnummer	(1:503010000027095)
inonderzoek	0
lokaalid	G0503.032e68f0085849cce0532ee22091b28c
lv_publicatiedatum	2016-04-12T11:54:23.000
measuredHeight	7.6399998664856
min-height-surface	-0.0199999995529652
namespace	NL.IMGeo
plaatsingspunt	(1:84841.951447542.723)
plus_status	geenWaarde
relatievehoogteligging	0
tekst	(1:168)
terminationdate	
tijdstipregistratie	2016-03-15T12:02:49.000

Introduction: research

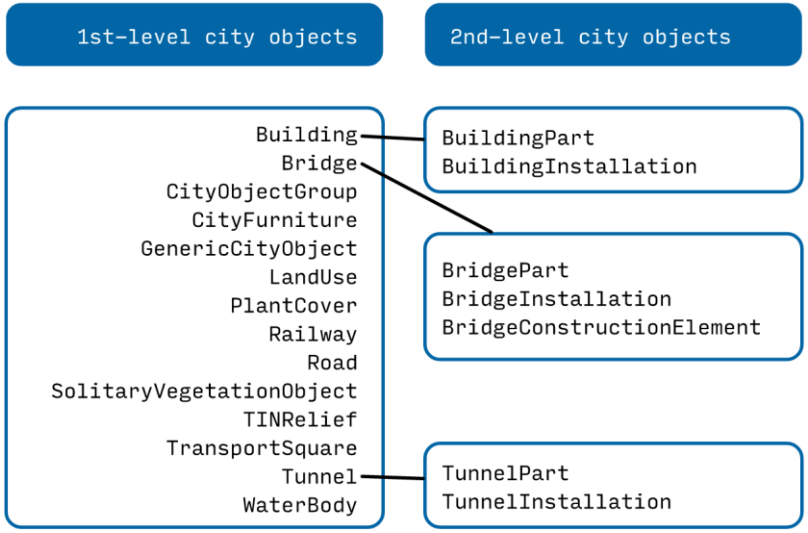
- Different use cases for 3D city models on the web

? *To what extent can the implementation of different compression techniques improve CityJSON's performance on the web, considering **file size**, **visualisation**, **querying**, **spatial analysis**, and **editing** performance?*

Compression type performance indicators
Visualisation time
Querying time
Spatial analysis time
Editing time
File size compression
Lossiness

Introduction: CityJSON

- Follows the CityGML data model
- But: on average 6x smaller
- Wavefront OBJ-style geometries



Introduction: CityJSON

Highlighted: attributes

geometry


```
{
    "type": "CityJSON",
    "version": "1.0",
     "CityObjects": {
        "b0a8da4cc-2d2a-11e6-9a38": {
            "type": "BuildingPart",
            "attributes": {
            "creationdate": "2014-07-09",
                "terminationdate": "",
                "function": "garage"
                },
                "geometry": [
                {
                    "type": "Solid",
                    "boundaries": [
                        [
                            [
                                [
                                    [0, 1, 2],
                                    [1, 2, 3],
                                    [1, 3, 4],
                                    [0, 1, 4]
                                ]
                            ]
                        ]]
                    },
                    "lod": 2,
                },
            "parents": ["b0a8da4cc-2d2a-11e6-9a39
                 "]
            },
        "b1105d28c-00ba-11e6-b420": {..},
        "b1126a169-00ba-11e6-b420" {..}
    },
    "vertices": [..]
}
```



```
{
    "type": "CityJSON",
    "version": "1.0",
    "metadata": {..},
    "CityObjects": {..},
     "vertices": [
          85012.343,
          447455.577,
          -0.27
          ],
          [
          85010.804,
          447448.808,
          -0.27
          ],
          [
          85013.832,
          447447.447,
          -0.27
          ],
          ...
}
```

Introduction: CityJSON

Highlighted: attributes

geometry


```
{
  "type": "CityJSON",
  "version": "1.0",
   "CityObjects": {
       "b0a8da4cc-2d2a-11e6-9a38": {
           "type": "BuildingPart",
           "attributes": {
           "creationdate": "2014-07-09",
               "terminationdate": "",
               "function": "garage"
               },
               "geometry": [
               {
                   "type": "Solid",
                   "boundaries": [
                       [
                           [
                               [
                                   [0, 1, 2],
                                   [1, 2, 3],
                                   [1, 3, 4],
                                   [0, 1, 4]
                               ]
                           ]
                       ]
                   },
                   "lod": 2,
               },
           "parents": ["b0a8da4cc-2d2a-11e6-9a39
                "]
           },
       "b1105d28c-00ba-11e6-b420": {..},
       "b1126a169-00ba-11e6-b420" {..}
   },
   "vertices": [..]
}
```



```
{
    "type": "CityJSON",
    "version": "1.0",
    "metadata": {..},
    "CityObjects": {..},
     "vertices": [
          85012.343,
          447455.577,
          -0.27
          ],
          [
          85010.804,
          447448.808,
          -0.27
          ],
          [
          85013.832,
          447447.447,
          -0.27
          ],
          ...
}
```

Introduction: CityJSON

Highlighted: attributes

geometry


```
{
    "type": "CityJSON",
    "version": "1.0",
    "CityObjects": {
        "b0a8da4cc-2d2a-11e6-9a38": {
            "type": "BuildingPart",
            "attributes": {
            "creationdate": "2014-07-09",
                "terminationdate": "",
                "function": "garage"
                },
                "geometry": [
                {
                    "type": "Solid",
                    "boundaries": [
                        [
                            [
                                [
                                    [0, 1, 2],
                                    [1, 2, 3],
                                    [1, 3, 4],
                                    [0, 1, 4]
                                ]
                            ]
                        ]]
                    },
                    "lod": 2,
                },
            "parents": ["b0a8da4cc-2d2a-11e6-9a39
                "]
            },
        "b1105d28c-00ba-11e6-b420": {..},
        "b1126a169-00ba-11e6-b420" {..}
    },
    "vertices": [..]
}
```



```
{
    "type": "CityJSON",
    "version": "1.0",
    "metadata": {..},
    "CityObjects": {..},
     "vertices": [
          85012.343,
          447455.577,
          -0.27
          ],
          [
          85010.804,
          447448.808,
          -0.27
          ],
          [
          85013.832,
          447447.447,
          -0.27
          ],
          ...
}
```

Introduction: CityJSON

Highlighted: mandatory members

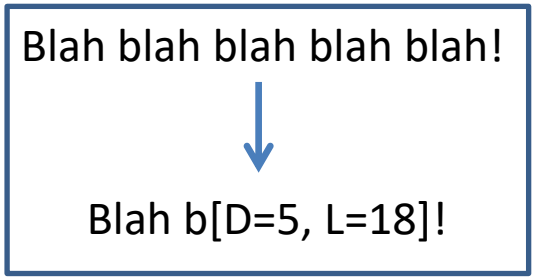

```
{
    "type": "CityJSON",
    "version": "1.0",
    "CityObjects": {..},
    "vertices": {..},
    "metadata": {
        "geographicalExtent": [
            84616.468,
            447422.999,
            -0.47,
            85140.839,
            447750.636,
            13.8
            ],
        "referenceSystem": "urn:ogc:def:crs:EPSG::7415"
        },
    "transform": {
        "scale": [
            0.001,
            0.001,
            0.001
            ].
        "translate": [
            84616.468,
            447422.999,
            -0.47
    }
}
```

Theory: compression

- Reduction of redundancy vs. (de)compression speed
- Lossy vs. lossless
- General-purpose vs. specific purpose
- ! Transmission time gain > (de)compression time

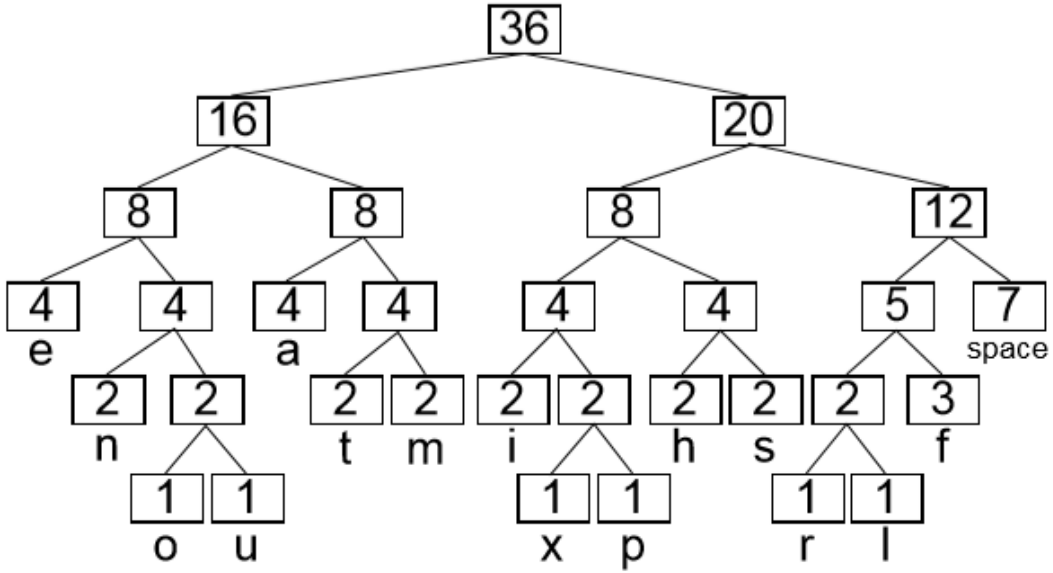
Theory: zlib

Lempel-Ziv 1977 and Huffman coding



Theory: zlib

Lempel-Ziv 1977 and **Huffman coding**



Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000

Source: Dcoetzee [2007]

Theory: attribute replace

```
{
    "cm": {
        "type": "Compressed CityJSON",
        "CityObjects": {
            0: "1",
            2: {
                3: "4",
                5: "6",
                7: "8"
            }
        },
    "str": ["type", "building", "attributes", "bgt_status", "bestaand",
             "creationdate", "2014-07-09", "class", "dek"],
    "vertices": [..]
}
```

Theory: attribute replace

```
{
    "cm": {
        "type": "Compressed CityJSON",
        "CityObjects": {
            0: "1",
            2: {
                3: "4",
                5: "6",
                7: "8"
            }
        },
    "str": ["type", "building", "attributes", "bgt_status", "bestaand",
             "creationdate", "2014-07-09", "class", "dek"],
    "vertices": [..]
}
```

Theory: attribute replace

```
{
    "cm": {
        "type": "Compressed CityJSON",
        "CityObjects": {
            0: "1",
            2: {
                3: "4",
                5: "6",
                7: "8"
            }
        },
    "str": ["type", "building", "attributes", "bgt_status", "bestaand",
            "creationdate", "2014-07-09", "class", "dek"],
    "vertices": [..]
}
```

Theory: binary JSON

- JSON has key-value structure, human-readable
- But: binary files are concise and processed faster
- CBOR is one binary encoding for JSON
- Binary code for data type and length of data

Related work: Draco

- Draco library for improved storage and transmission of 3D graphics
- Different compression levels, metadata
- Quantisation, Edgebreaker, parallelogram prediction, delta encoding

Methodology: tested CityJSON variants

	Original geometry	**Draco geometry**
1	original	draco
2	original-zlib	draco-zlib
3	original-cbor	draco-cbor
4	original-cbor-zlib	draco-cbor-zlib
5	original-replace	draco-replace
6	original-cbor-replace	draco-cbor-replace
7	original-cbor-replace-zlib	draco-cbor-replace-zlib

Methodology: benchmarking

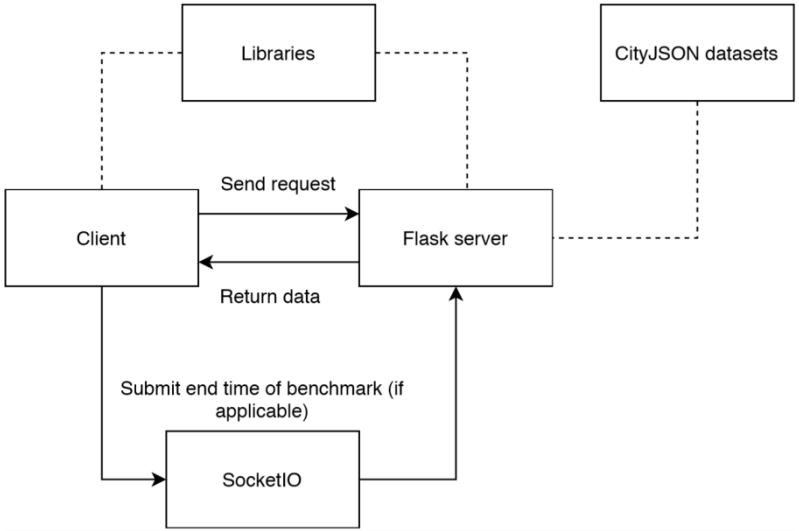
8 (16) datasets

large vs. small
attributes vs. no attributes

2 server implementations

Compression in advance vs. on the fly

9 data operations

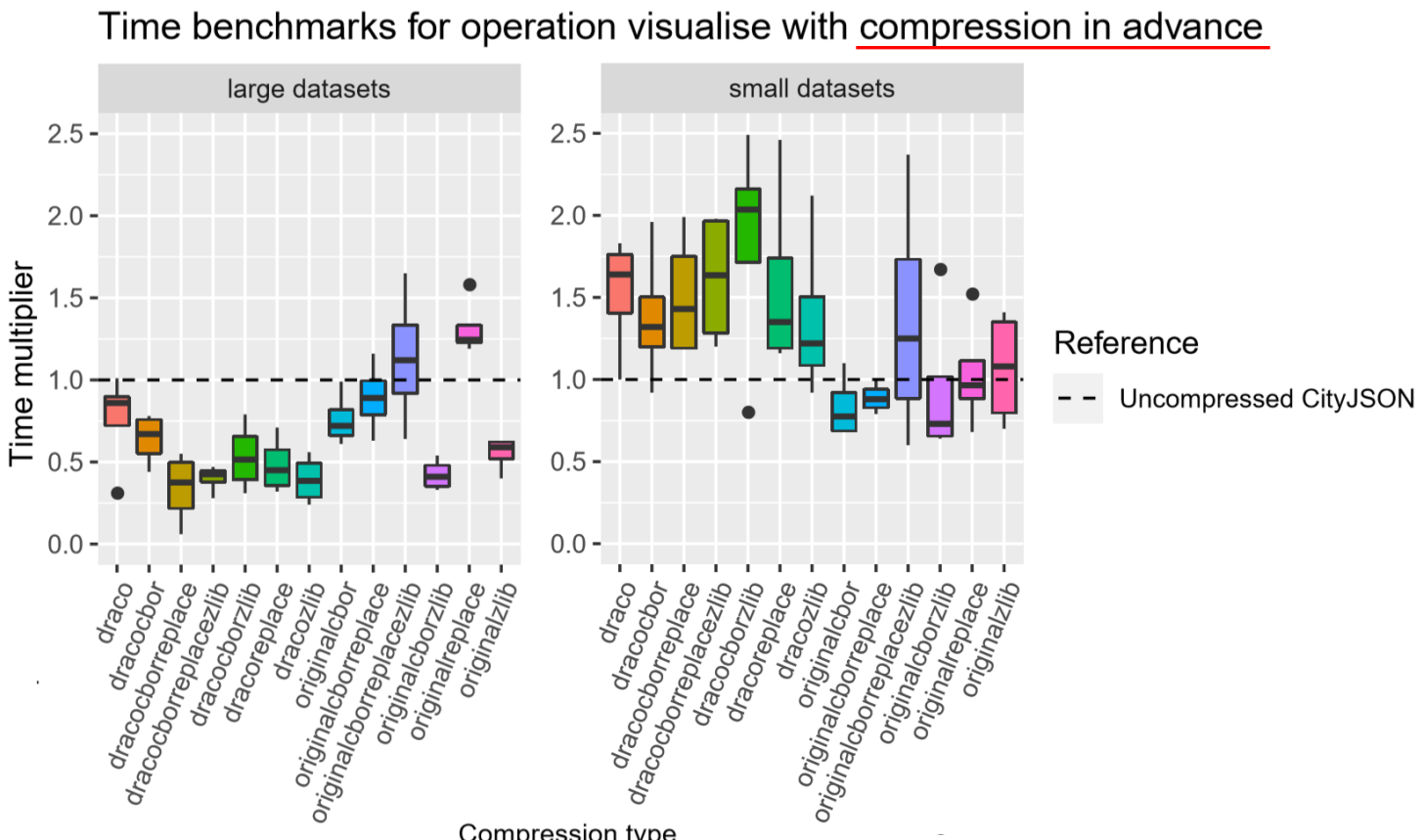


Use case	Operations
Visualisation	Visualise all features
Querying	Query one feature Query all features
Spatial analysis	Buffer one feature Buffer all features
Editing (attributes)	Edit one feature Edit all features
Editing (geometries)	Edit one feature Edit all features

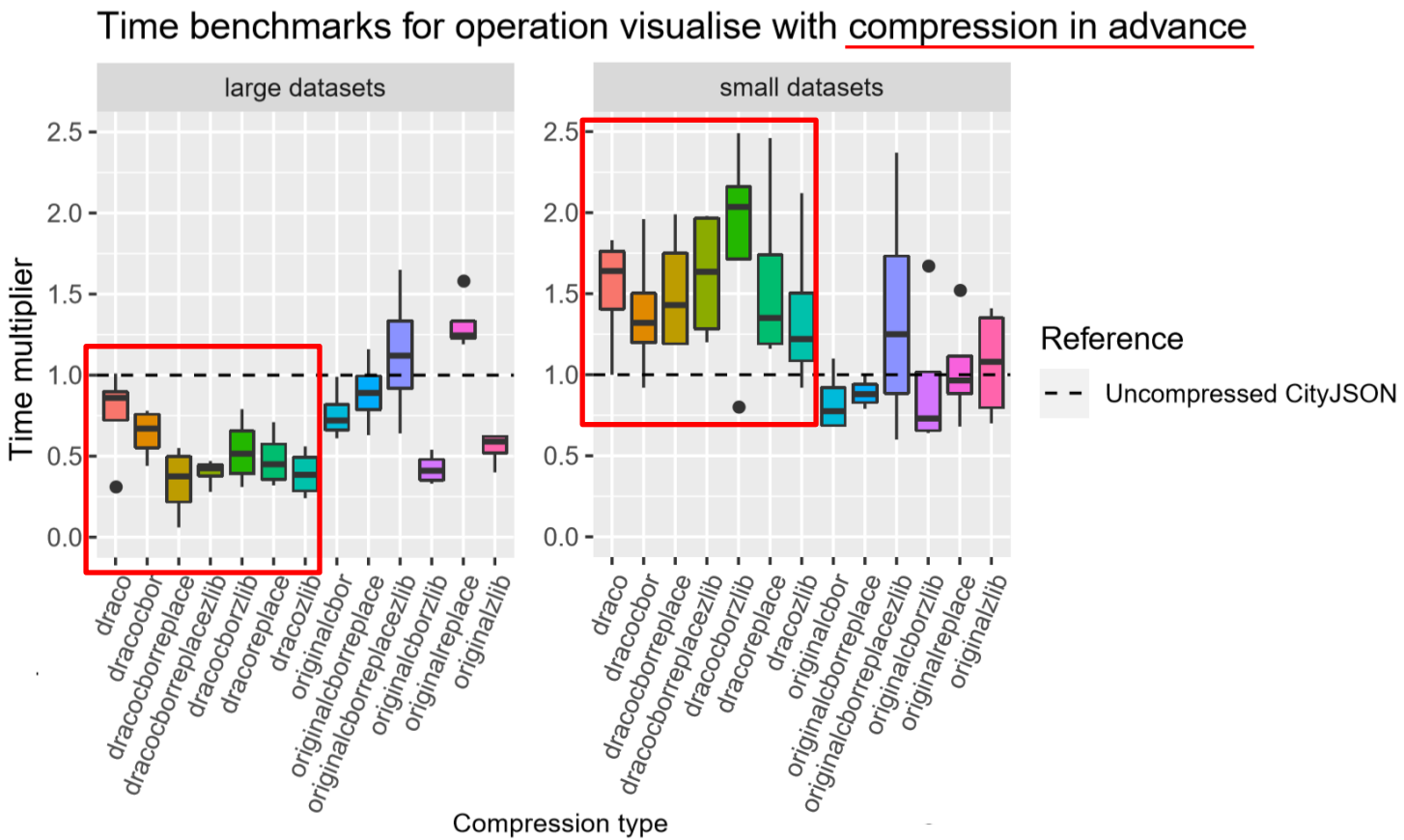
Results: notes

- Performance improvement: function of file size and (de)compression time
- Lossiness: all techniques used are lossless
- Box plots: variability in datasets

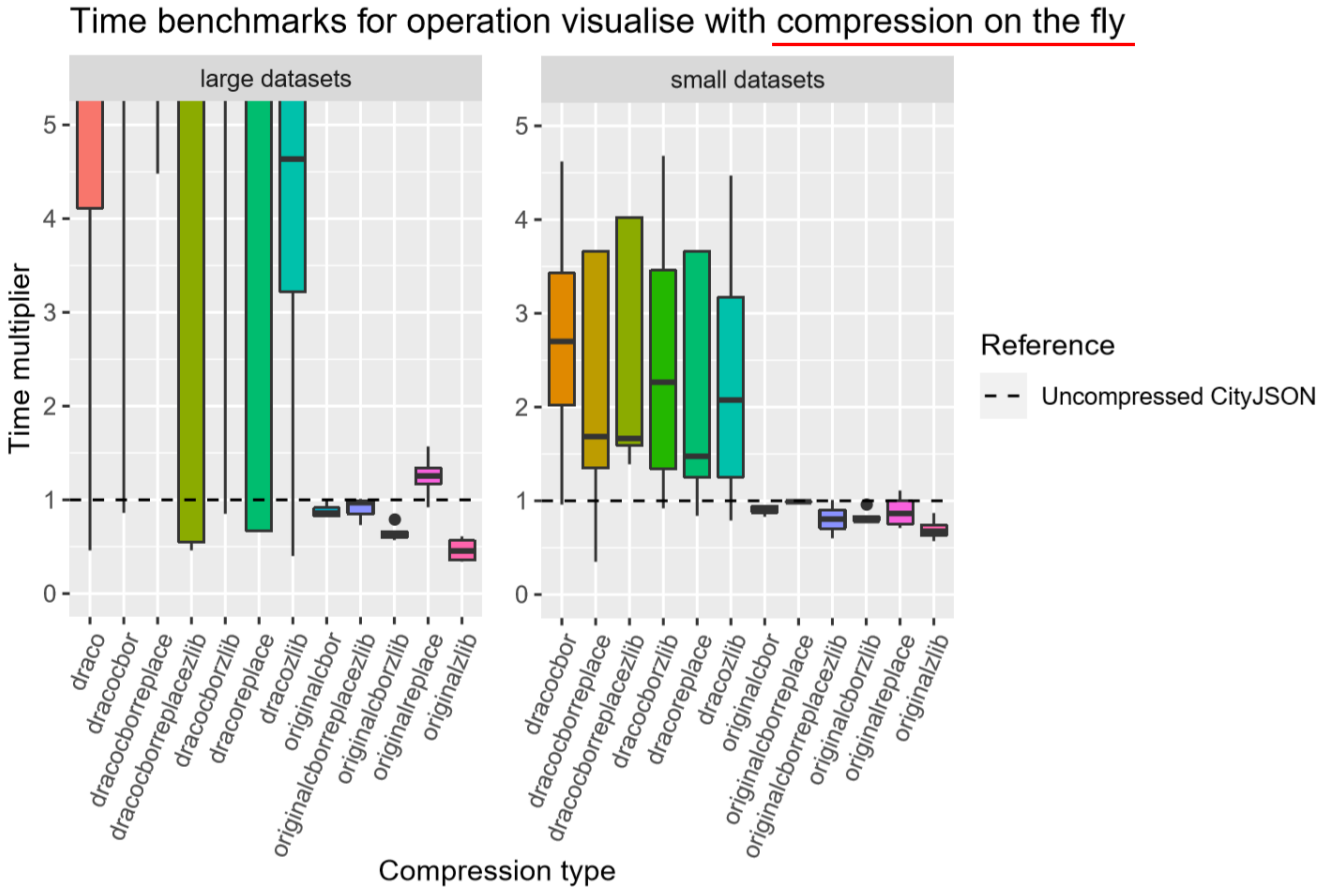
Results: visualisation



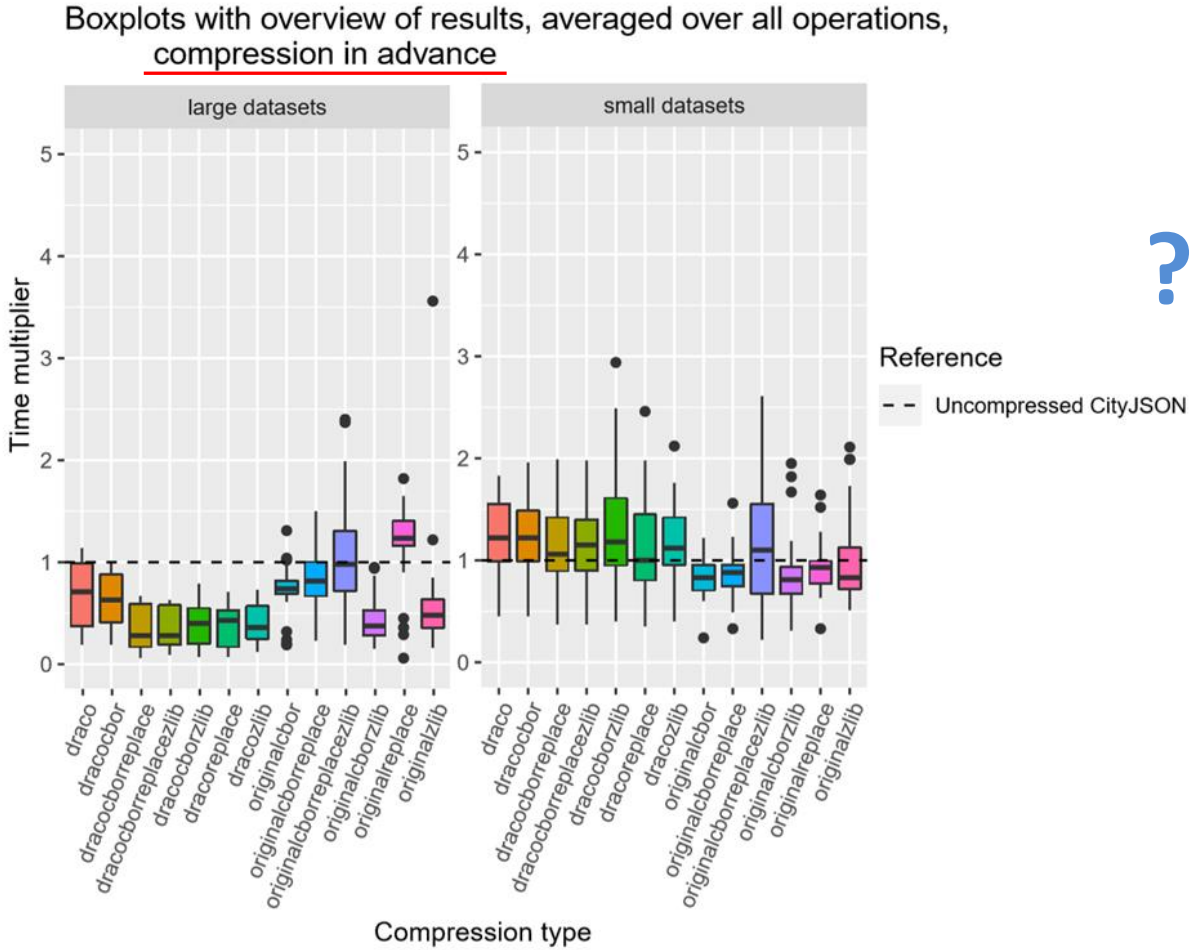
Results: visualisation



Results: visualisation

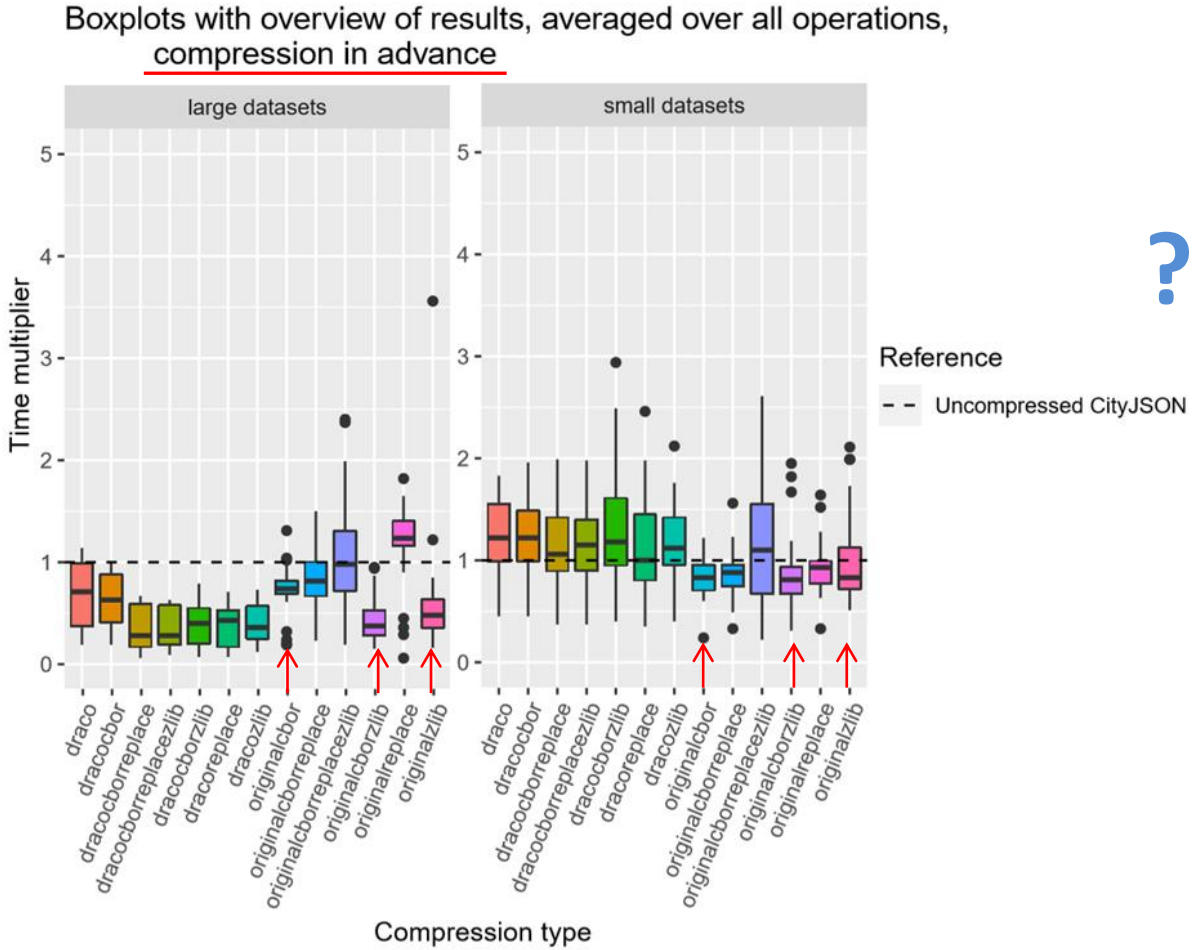


Results: conclusion



? *"I have this dataset and want to do anything with it, which compression types are the most suitable?"*

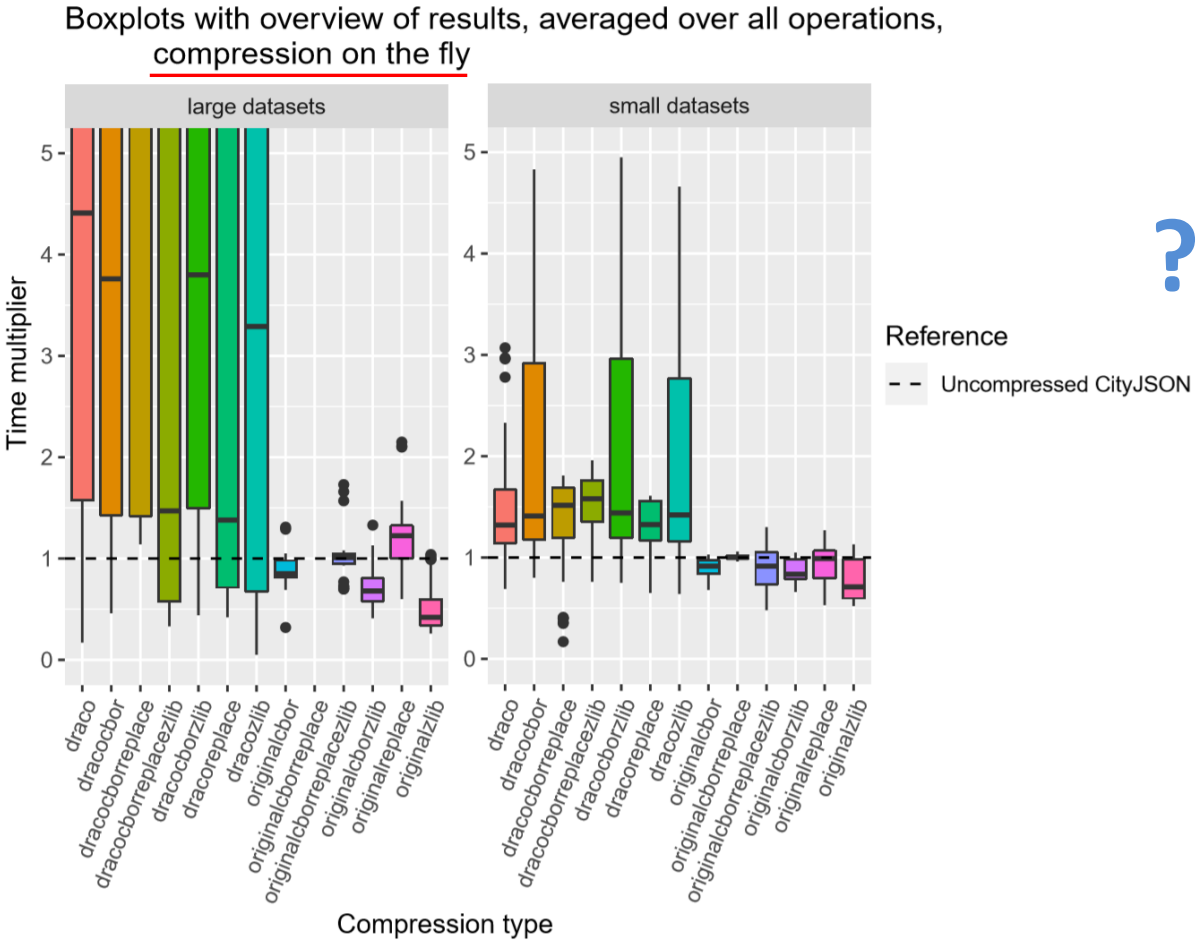
Results: conclusion



?

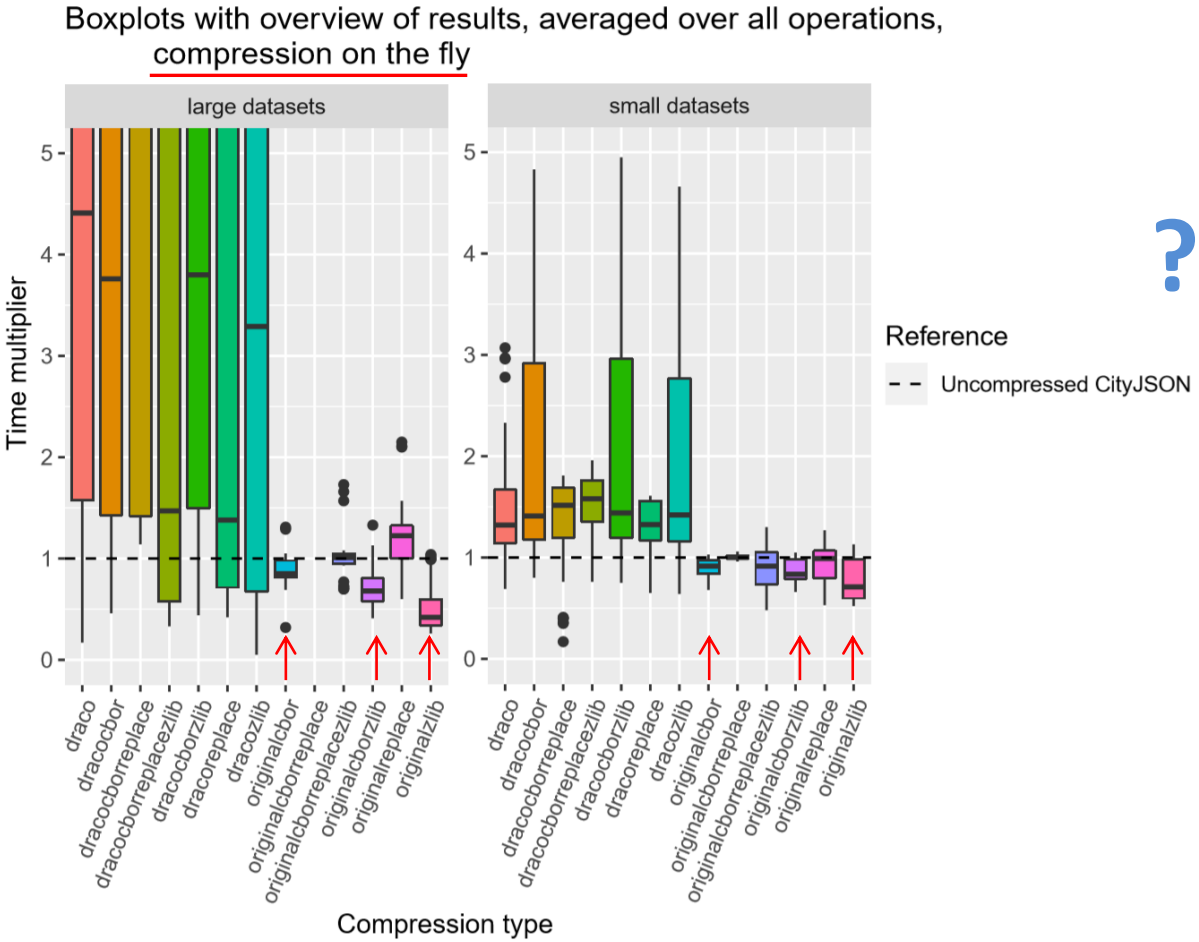
"I have this dataset and want to do anything with it, which compression types are the most suitable?"

Results: conclusion



? *"I have this dataset and want to do anything with it, which compression types are the most suitable?"*

Results: conclusion



?

"I have this dataset and want to do anything with it, which compression types are the most suitable?"

Discussion

- Alternatives: **b3dm** and **I3S**
- Created for high (visualisation) performance
- ! CityJSON much easier to use, and should be kept like that!
- Compression can be beneficial, but needs some further investigation
- 3DCityDB, streaming?

References (important ones)

- CBOR. Concise Binary Object Representation (CBOR), 2013. URL https://tools.ietf.org/html/ rfc7049.
- Dcoetzee. File:Huffman tree.svg, 2007. URL https://commons.wikimedia.org/wiki/File: Huffman_tree.svg.
- Google. Draco 3D data compression, 2019b. URL https://google.github.io/draco/.
- D. A. Huffman. A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9):1098–1101, 1952.
- H. Ledoux, K. Ohori, K. Kumar, B. Dukai, A. Labetski, and S. Vitalis. CityJSON: A compact and easyto-use encoding of the CityGML data model. Open Geospatial Data, Software and Standards, 4 (1):4, 2019.
- J. Rossignac, A. Safonova, and A. Szymczak. Edgebreaker on a corner table: A simple technique for representing and compressing triangulated surfaces. In Hierarchical and geometrical methods in scientific visualization, pages 41–50. Springer, 2003.
- G. Taubin, W. P. Horn, F. Lazarus, and J. Rossignac. Geometry coding and vrml. Proceedings of the IEEE, 86(6):1228–1243, 1998.
- zlib. zlib - A Massively Spiffy Yet Delicately Unobtrusive Compression Library, 2020. URL https: //zlib.net/.

Extra slides

Relevant work: b3dm

- Batched 3D Model
- Cesium 3D Tiles
- Feature table, batch table, glTF
- Mix of JSON/binary

- Attributes: "height" : [10.0, 20.0, 15.0]
- But: feature template

Relevant work: I3S

- Indexed 3D Scene Layer
- Heterogeneous features (3D meshes and point clouds)
- Also tiles, but other specification
- Mix of JSON/binary
- CRS per node

Relevant work: streaming

File size in MB of different streaming implementations

```
{
  "type": "CityJSONFeature",
  "id": "myid", //-- to clearly identify which of the CityObjects is the "main" one
  "CityObjects": {},
  "vertices": [],
  "appearance": {},
}
```

Dataset	collection	collection-cbor	collection-cbor-draco	collection-cbor-draco2
Delft	1.2	0.7	0.6	0.6
Den Haag	3.3	2.4	2.4	2.3
Rotterdam	3.9	2.5	2.6	2.6
Montréal	5.4	3.5	2.9	2.9
Singapore	54.5	34.4	32.1	32.1
TU Delft campus	70.6	41.8	25.5	25.5
New York	109.8	69.7	70.2	70.2
Zürich	284.2	182.5	163.2	165.6

Theory: quantisation

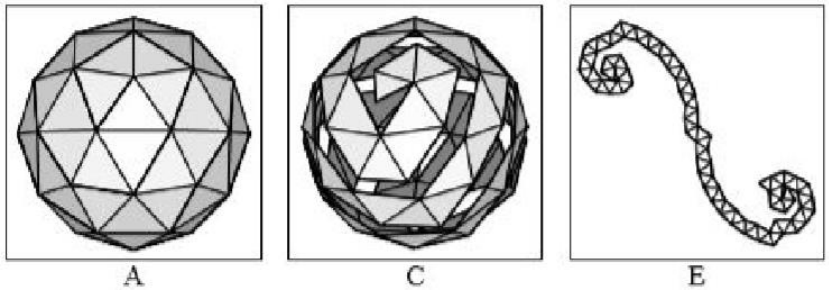
- Lossy—complete vertices and precision
- Coordinates close to each other are mapped to 1 of these coordinates

85016.719, 447470.356, 0.357
85016.643, 447470.446, 0.358
85016.780, 447470.352, 0.402

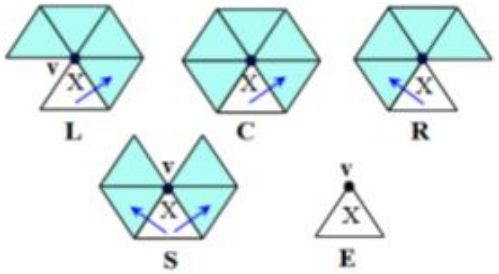
85016.643, 447470.446, 0.358

Theory: edgebreaker

- Compresses mesh connectivity
- Mesh -> triangle spanning tree -> CLERS string
- Decompression: triangle edges zipped based on CLERS character



Source: Taubin et al. (1998)

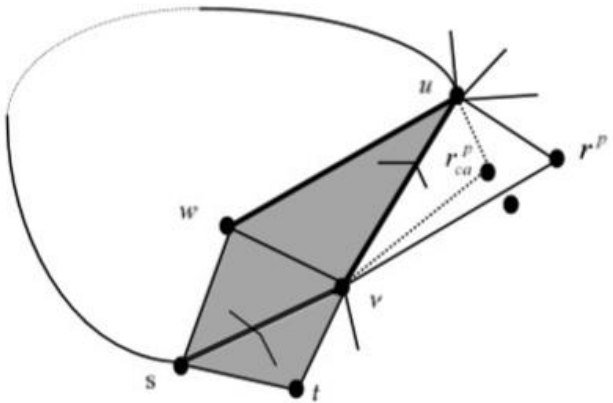


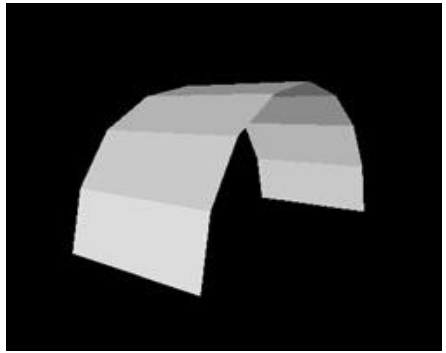
Source: Rossignac et al. (2003)

Theory: parallelogram prediction

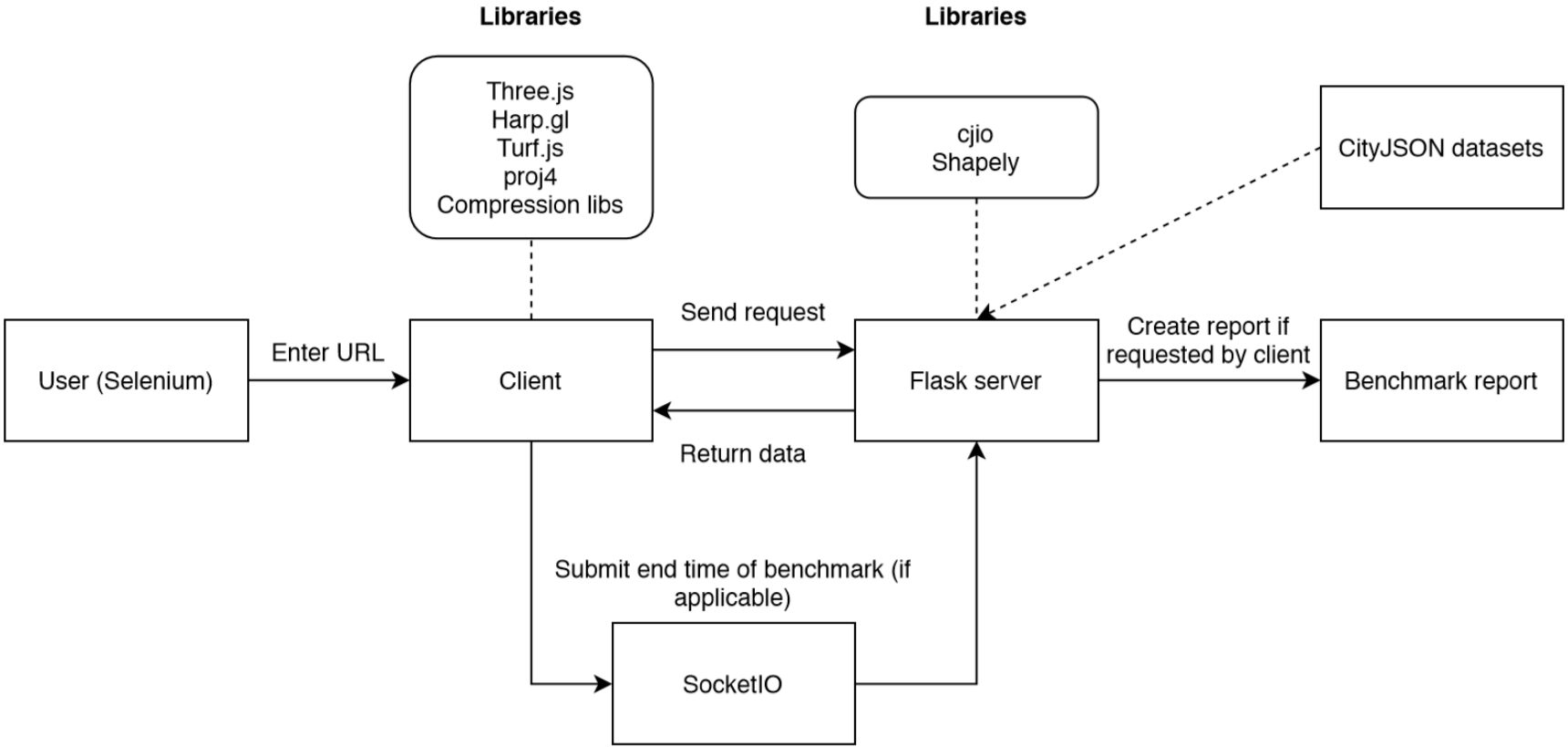
- Storing error of prediction instead of full coordinate
- Parallelogram rule

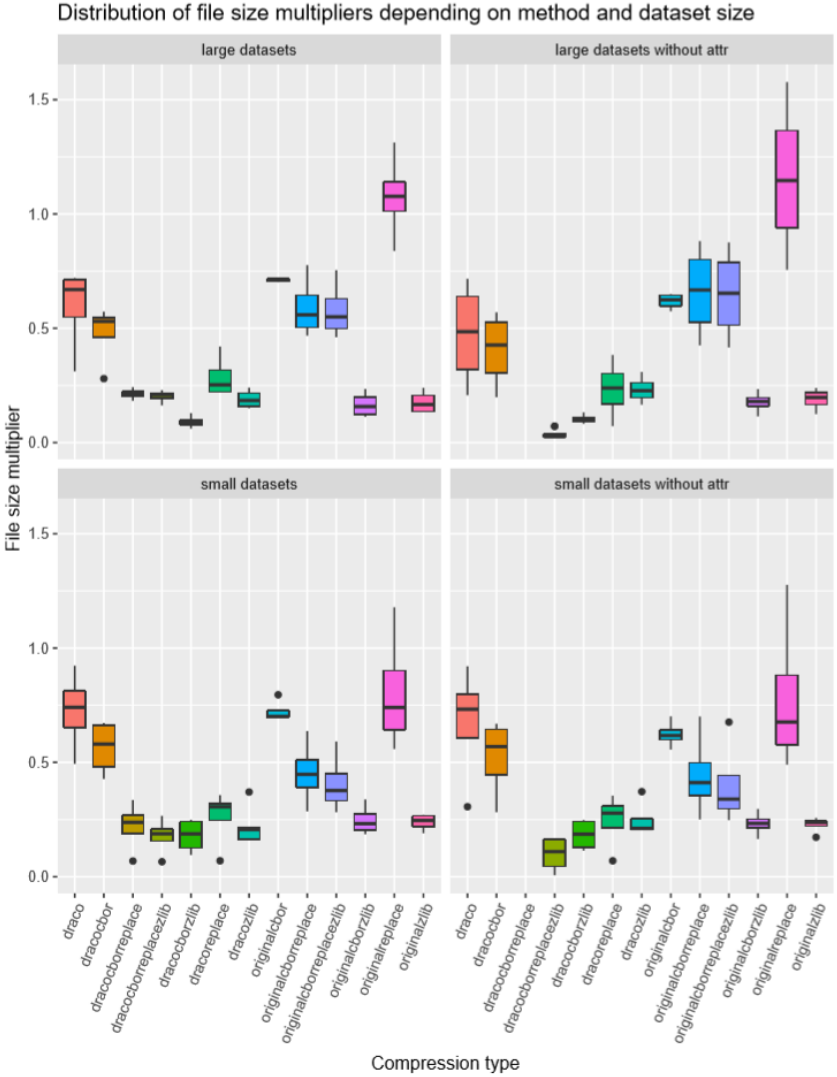
$$r^p = v + w - u$$

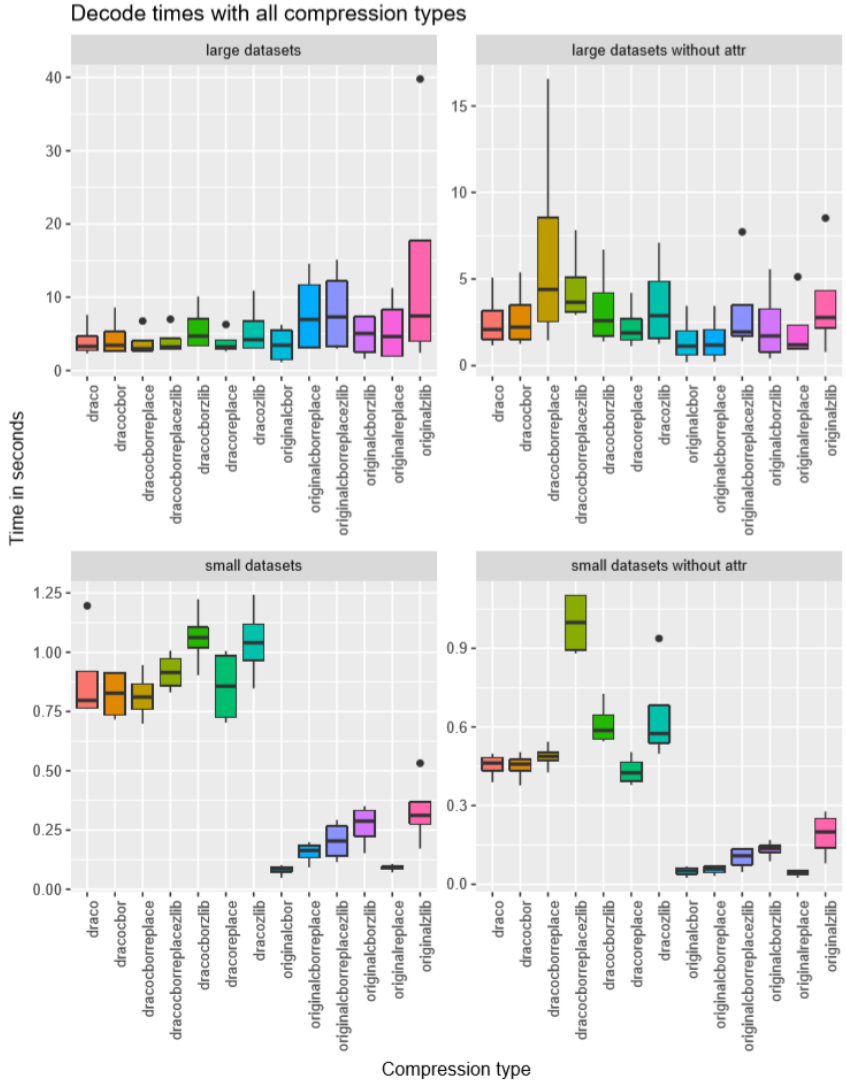


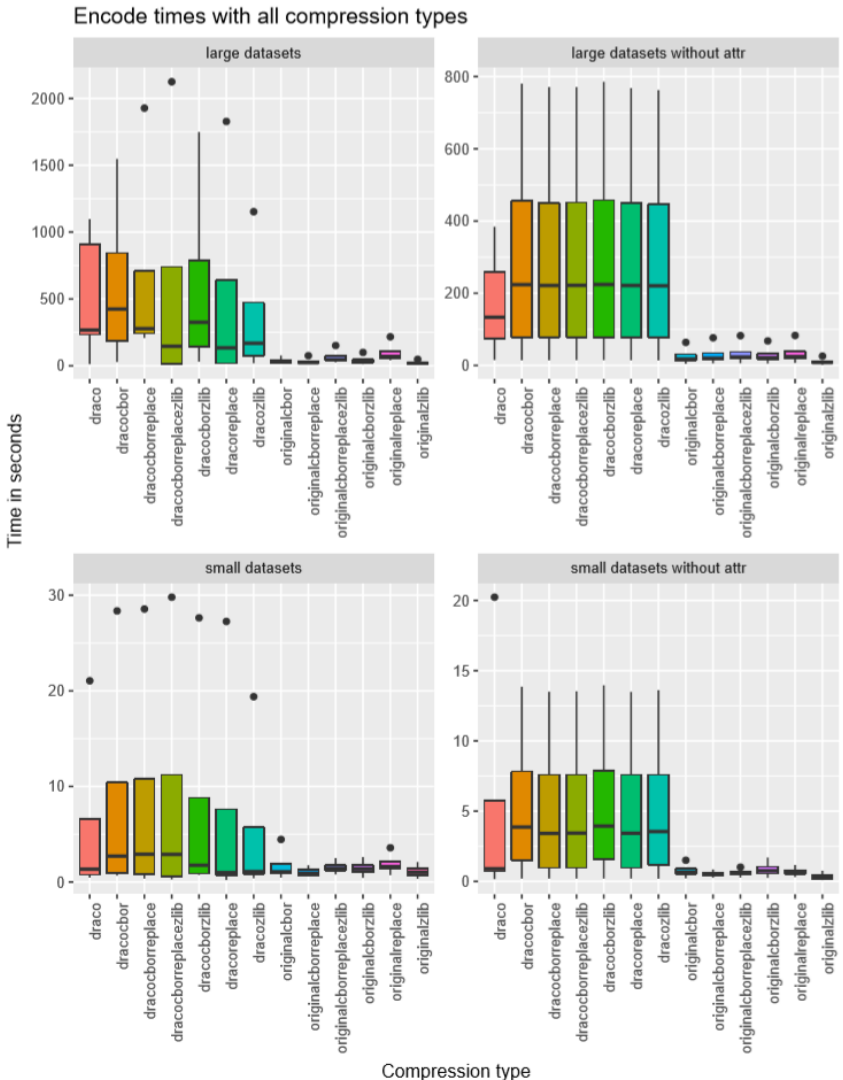


Methodology: testing platform




Distribution of file size multipliers depending on method and dataset size
large datasets
large datasets without attr
small datasets
small datasets without attr
File size multiplier
1.5
1.0
0.5
0.0
draco
dracocbor
dracocborreplace
dracocborreplacezlib
dracocborzlib
dracoreplace
dracozlib
originalcbor
originalcborreplace
originalcborreplacezlib
originalcborzlib
originalreplace
originalzlib
Compression type


Decode times with all compression types
large datasets
large datasets without attr
small datasets
small datasets without attr
Time in seconds
Compression type
draco
dracocbor
dracocborreplace
dracocborreplacezlib
dracocborzlib
dracoreplace
dracozlib
originalcbor
originalcborreplace
originalcborreplacezlib
originalcborzlib
originalreplace
originalzlib


Encode times with all compression types
large datasets
large datasets without attr
small datasets
small datasets without attr
Time in seconds
Compression type
draco
dracocbor
dracocborreplace
dracocborreplacezlib
dracocborzlib
dracoreplace
dracozlib
originalcbor
originalcborreplace
originalcborreplacezlib
originalcborzlib
originalreplace
originalzlib