Best practices for building web apps that visualize large datasets

Jeremy Bartley, Kristian Ekenes

2021 ESRI
DEVELOPER SUMMIT

Overview

- Initial considerations
- What the JS API and ArcGIS platform do for you
- Optimizing your apps and data for better performance
  - Choosing the right layer type
  - Visualization Techniques
    - Scale-dependent filtering
    - Clustering
  - Data prep/processing
    - Reducing/generalizing features and attributes
    - Aggregation

Considerations before you start development

Jeremy Bartley

Considerations before you start development

- Know your audience
  - Just a few specialized users?
  - Public or private?
  - Potential to go viral?
- Devices
  - Desktop only?
  - Mobile and tablets
- Generalization
  - Choose an appropriate geometry type for scale
  - Reduce number of features
  - Reduce vertices

Behind the scenes

What the JS API and ArcGIS do for you

Feature tiles

Efficient, flexible, and scalable way to bring data to the client

Not secure | jbartley2/webdev/4.15/visualization-vv-color-animate.html

New York Construction

Query String Parameters view source view URL encoded

f: pbf

geometry: {"spatialReference":{"latestWkid":3857,"wkid":102100},"x
3.06989474,"ymin"
02691362}

maxRecordCountFac

outFields: CNSTRCT_

outSR: 102100

quantizationParame
102100},"xmin":-8
0463991,"ymax":49
t","tolerance":38

resultType: tile

returnExceededLimi

spatialRel: esriSpat

where: CNSTRCT_YR

geometryType: esriGeometryEnvelope

inSR: 102100

Name	Met...	St...	Pr...	Sche...	Type	Size	Time	C...	x-cache	x-esri-ftiles-ca	x
query?f=...	GET	200	h2	https	fetch	65.2 KB	90 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	89.6 KB	173 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	709 B	89 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	711 B	42 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	710 B	100 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	35.1 KB	57 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	29.5 KB	79 ms	br	Hit from cl...	true	
query?f=...	GET	200	h2	https	fetch	30.1 KB	245 ms	br	RefreshHit...	true	
query?f=...	GET	200	h2	https	fetch	23.7 KB	230 ms	br	Miss from ...	true	
query?f=...	GET	200	h2	https	fetch	50.2 KB	382 ms	br	Miss from ...	true	

Hit from cloudfront

NYC OpenData, State of New Jersey, Esri, HERE, Garmin, SafeGraph, METI/NASA, USGS, EPA, NPS, USDA

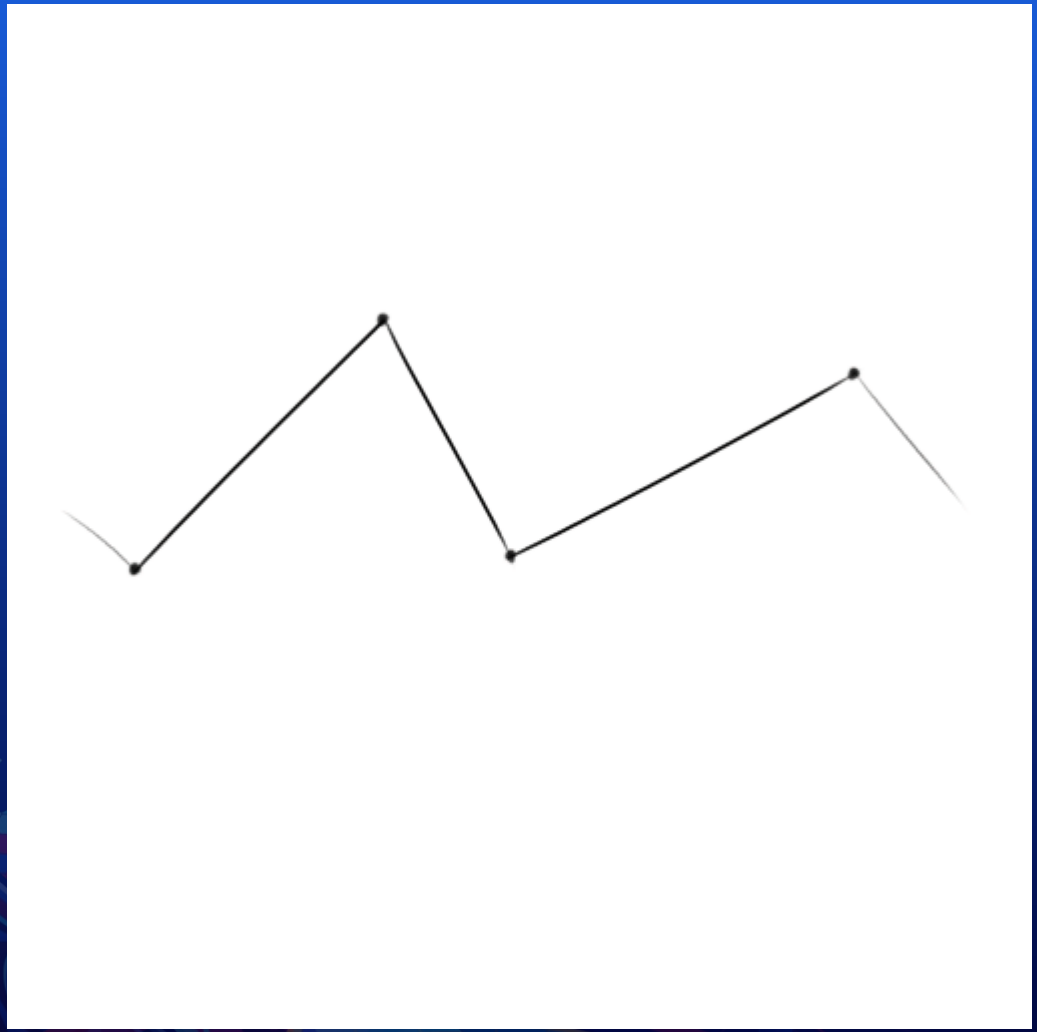
Powered by Esri

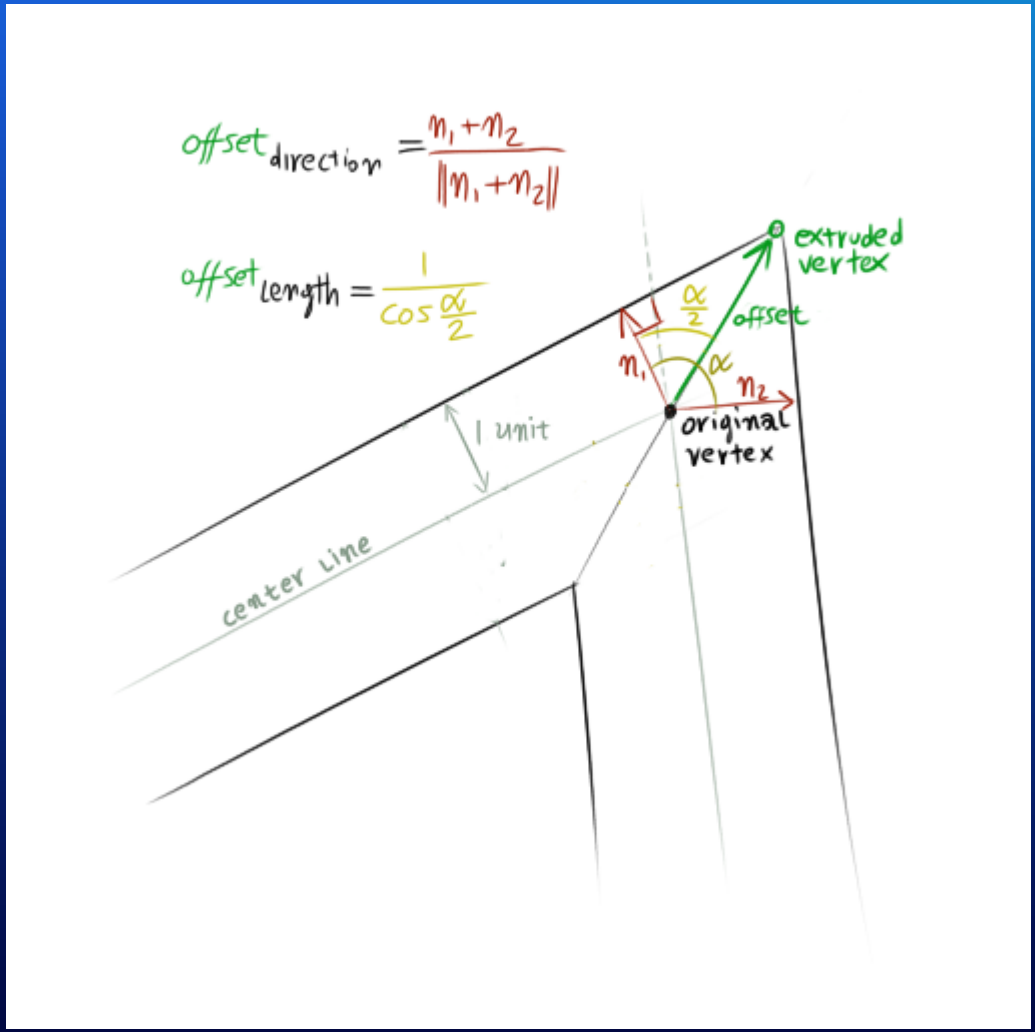
1984 1880 2017 Play

Data structure returned from feature tile query is highly efficient

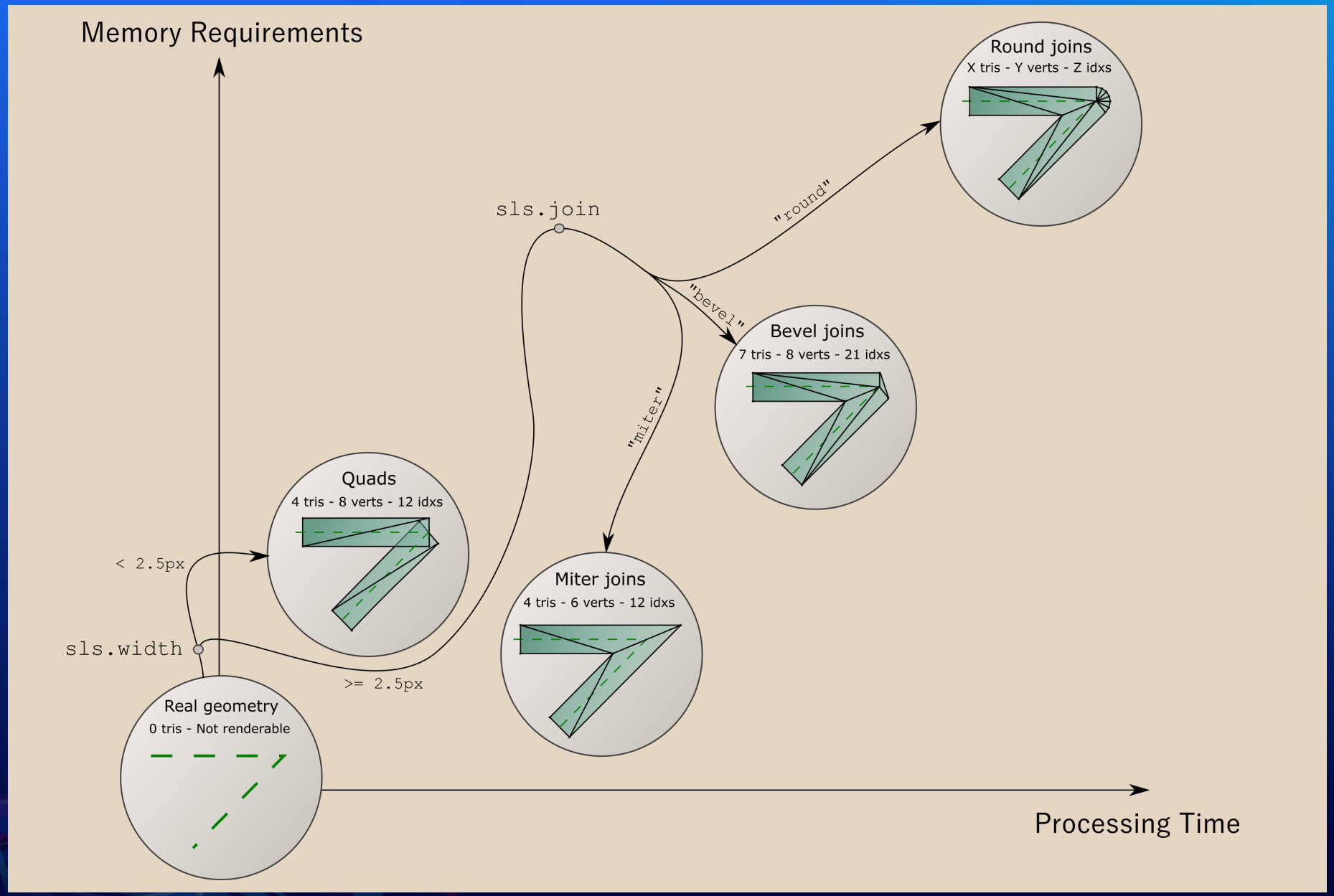

```
▼transform: {originPosition: "upperLeft", scale: [38.2185141425781
  originPosition: "upperLeft"
  ▶scale: [38.21851414257816, 38.21851414257816]
  ▶translate: [-8257645.0397049915, 4970241.327216988]
▶fields: [{name: "CNSTRCT_YR", type: "esriFieldTypeInteger", alias
▼features: [{attributes: {CNSTRCT_YR: 1925, OBJECTID_1: 42064},…},
  ▼[0 … 99]
    ▼0: {attributes: {CNSTRCT_YR: 1925, OBJECTID_1: 42064},…}
      ▶attributes: {CNSTRCT_YR: 1925, OBJECTID_1: 42064}
      ▼geometry: {rings: [[[166, 392], [0, 1], [-1, -1], [1, 0]]]}
        ▼rings: [[[166, 392], [0, 1], [-1, -1], [1, 0]]]
          ▼0: [[166, 392], [0, 1], [-1, -1], [1, 0]]
            ▶0: [166, 392]
            ▶1: [0, 1]
            ▶2: [-1, -1]
            ▶3: [1, 0]
    ▶1: {attributes: {CNSTRCT_YR: 1950, OBJECTID_1: 42080}, geometr
```

Efficiently process and prepare features to be drawn on the graphics card



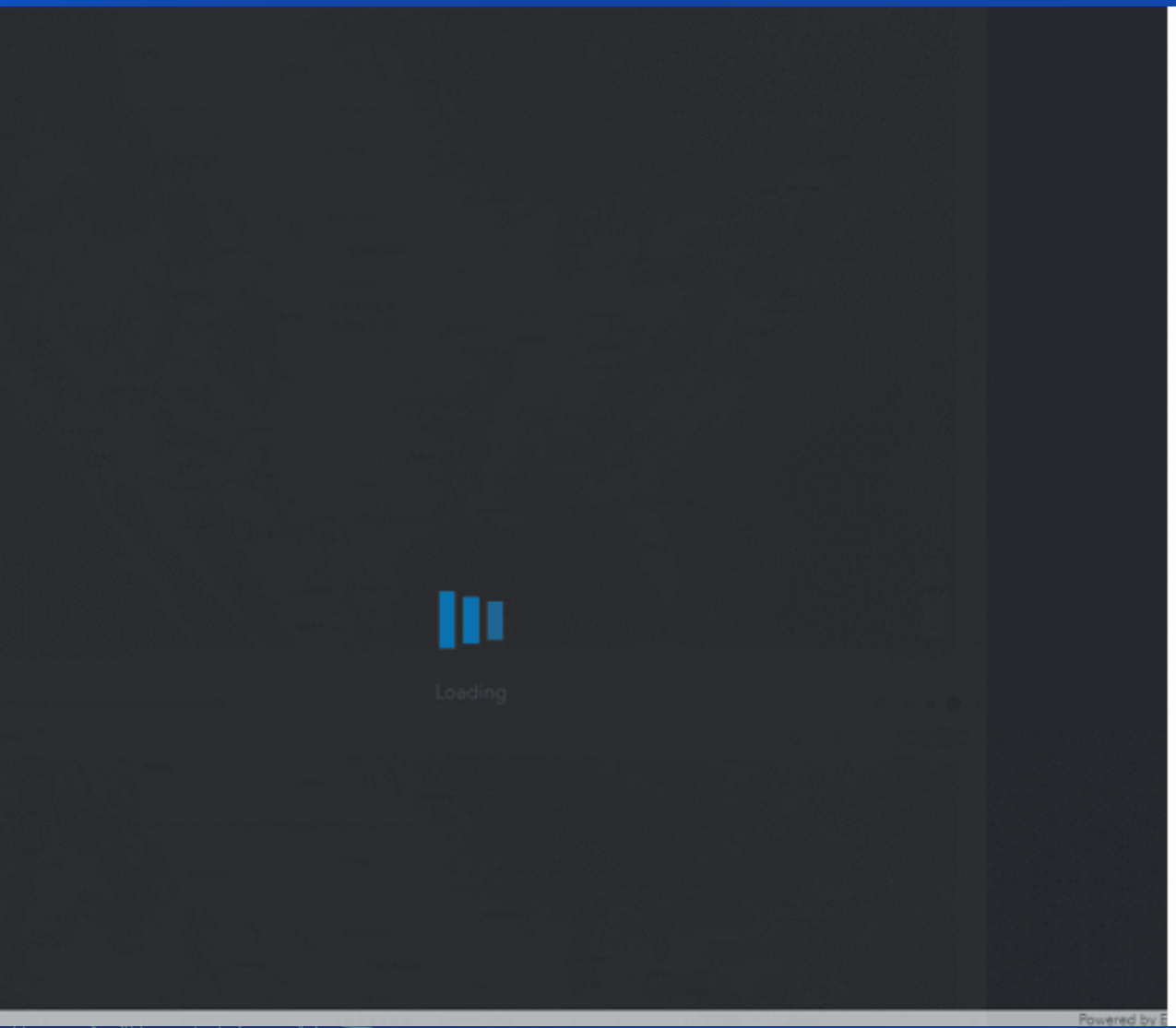


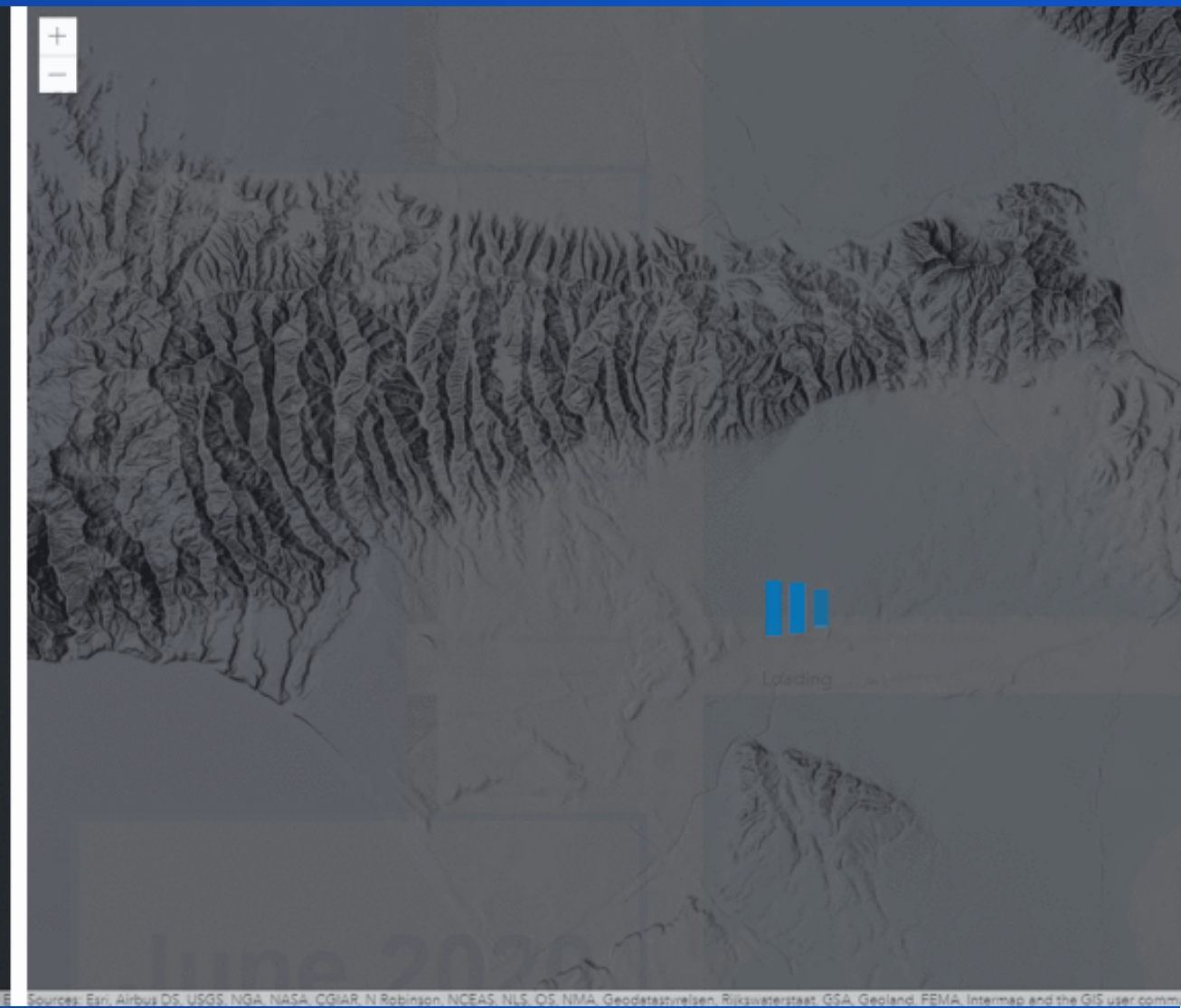
Line optimizations: balancing display quality with overall performance



PERFORMANCE

Initial load, visualizing large datasets
Continuous performance enhancements





Different types of layers

Use the right layer for the job

Different layer types

- Feature Layers
  - In almost all cases best layer for you to use
  - Built on the notion of shareable queries across users, across apps
  - Performance gets better as layer is used
  - Little to no effort to manage
  - Works great for interactive apps
- Raster Tiled Layers
  - Pregenerate raster tiles of data
  - Performance will be best of any layer type
  - Can still do popups and legends
  - Visualization cannot be controlled after built
- Vector Tiled Layer
  - Pregenerate vector tiles
  - Tiles include graphics and whatever attributes you included
  - Labels are dynamic
  - No legend, no popups

…Feature Layer is the best all around layer type

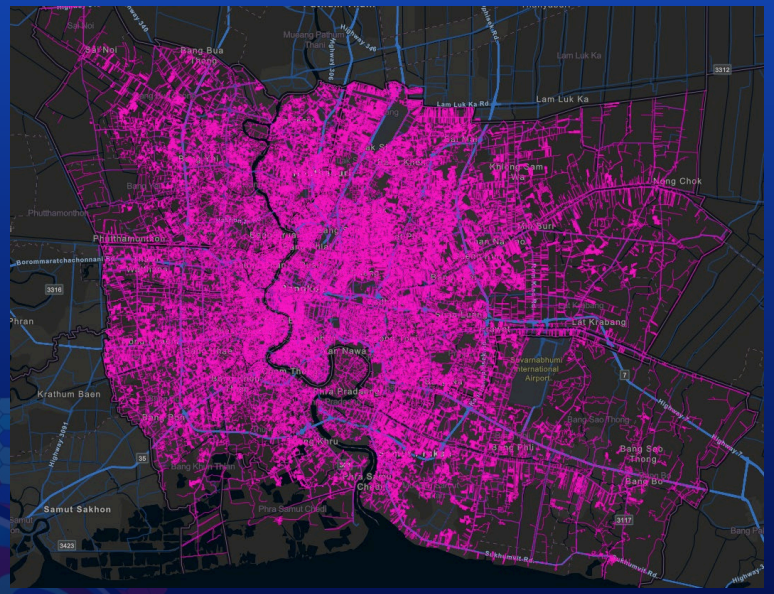
Visualization techniques

Scale-dependent filtering

Water distribution pipes

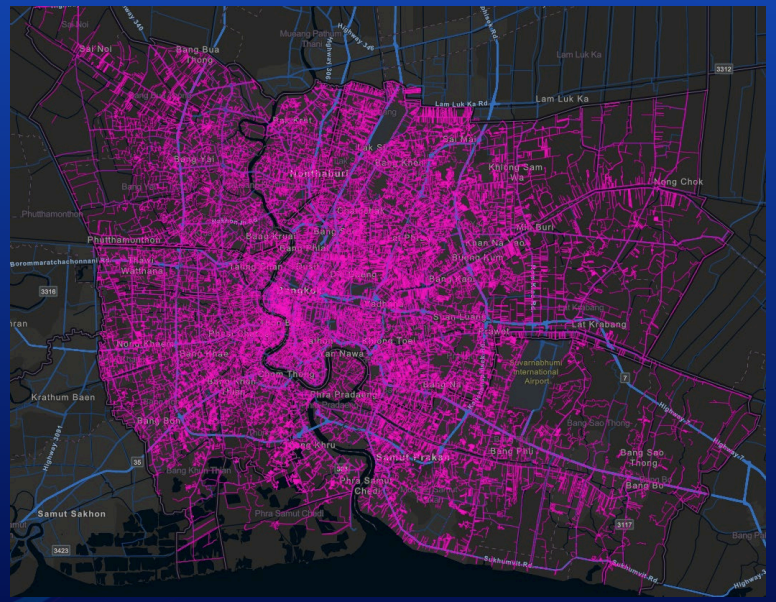
Filter by scale using attributes

- No filter
- 17.7 MB (uncompressed)
- ~7s (desktop on fast network)



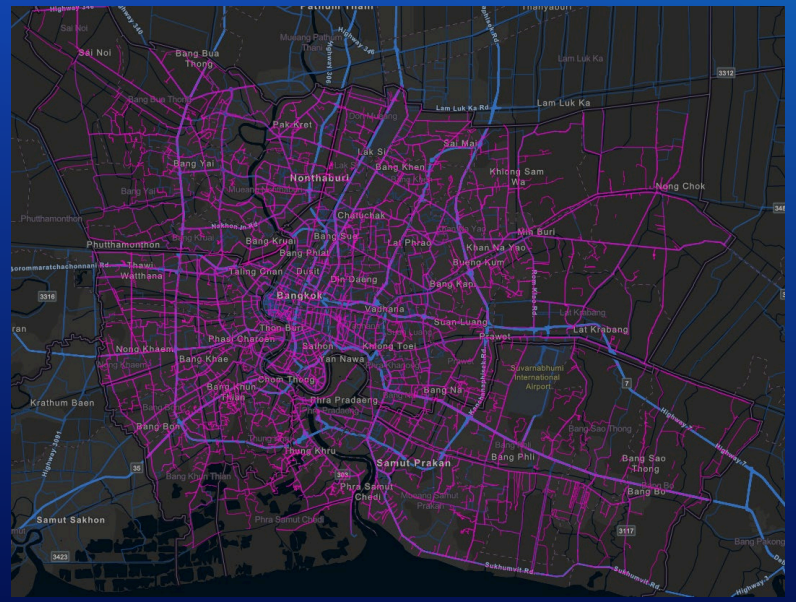
550,578 features

- Filter: pipe_size >= 150
- 14.7 MB (uncompressed)
- ~5s (desktop on fast network)



358,921 features

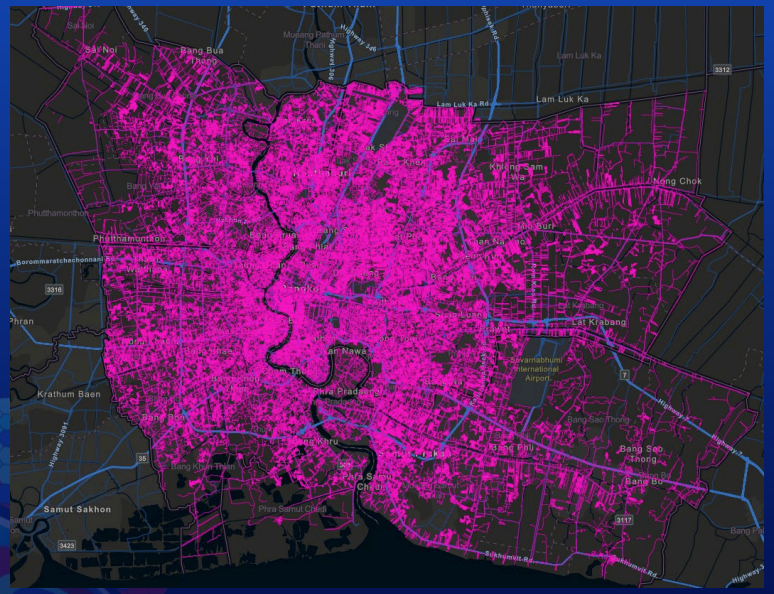
- Filter: pipe_size >= 300
- 11.2 MB (uncompressed)
- ~3s (desktop on fast network)



Still 107,256 features!

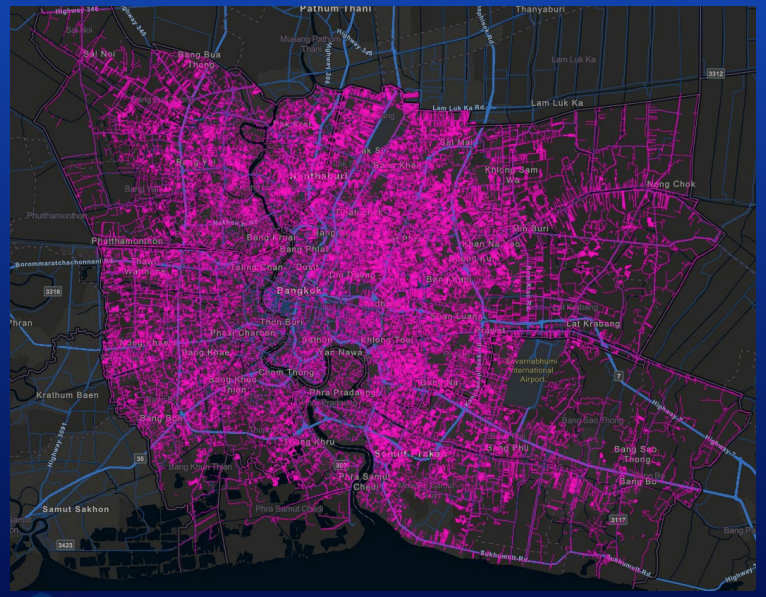
Filter by scale using view resolution

- No filter
- 17.7 MB (uncompressed)
- ~7s (desktop on fast network)



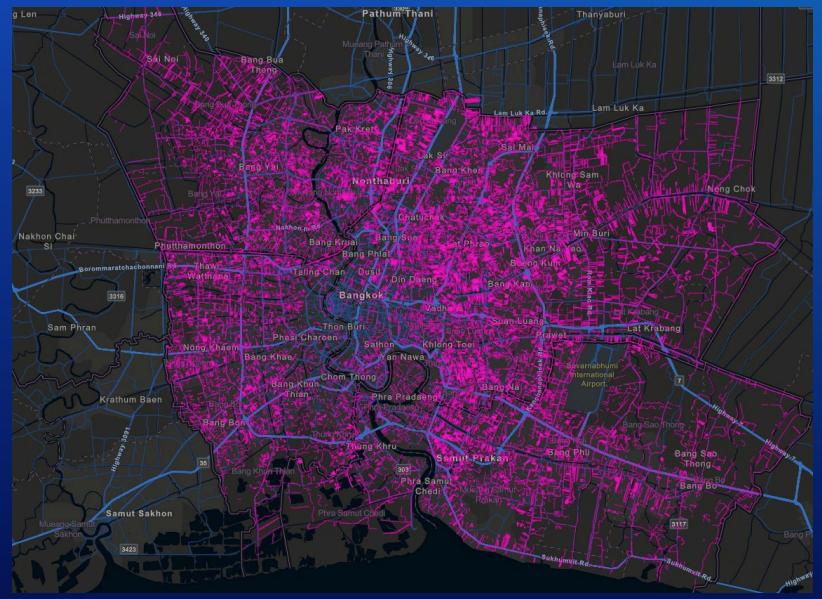
550,578 features

- view.resolution * 2
- 11.2 MB (uncompressed)
- ~3s (desktop on fast network)



Only 62,313 features!

- view.resolution * 3
- 10.5 MB (uncompressed)
- ~2.8s (desktop on fast network)



30,316 features!

Visualization techniques

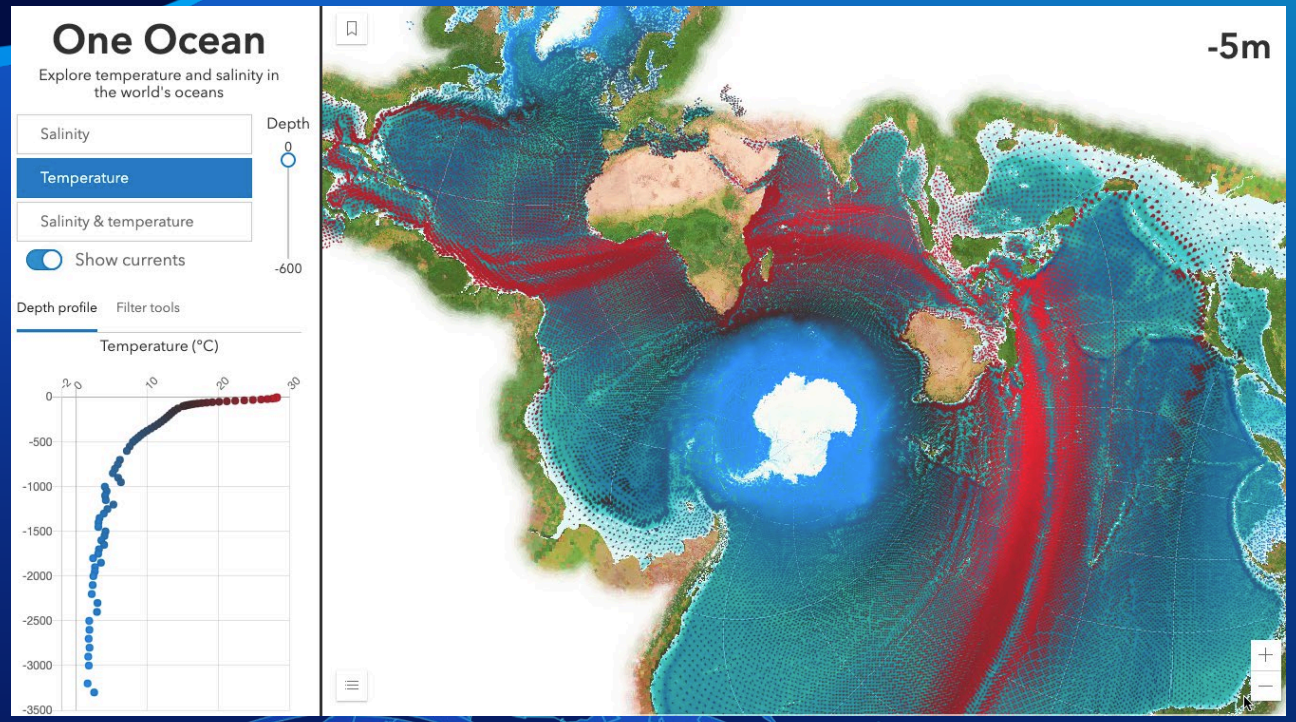
Clustering

Data prep and processing

Working with different geometry types

Data prep and processing

Thinning geometries and attributes

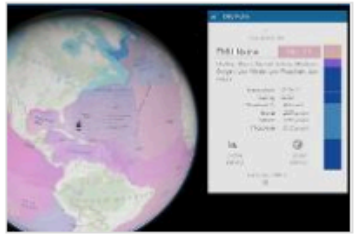


One Ocean

Kristian Ekenes

EMU Global Ocean (Pro)

Overview



Ecological Marine Units V1, Global Oceans

Project P

Created: Oc 480

Living At

Open in ArcGIS Pro

Download

Details

Size: 4,027 MB

Description

Ecological Marine Units (EMUs) for the e roject Package. Users are encouraged to download and explore th ArcGIS Map Packages (available within this Group) are available for down

A group on GeoNet is available for user

Details

Size: 4,027 MB

Owner

esri_oceans

Tags

Global Oceans, EMU, USGS, NOAA, Esri, Project Package, ppkx, 2D, ArcGIS Pro, 3D, Oceans, Ocean, Marine, esri_oceans, esri_marine, Oceans3

Terms of Use

Freely available for use!

Comments (1)

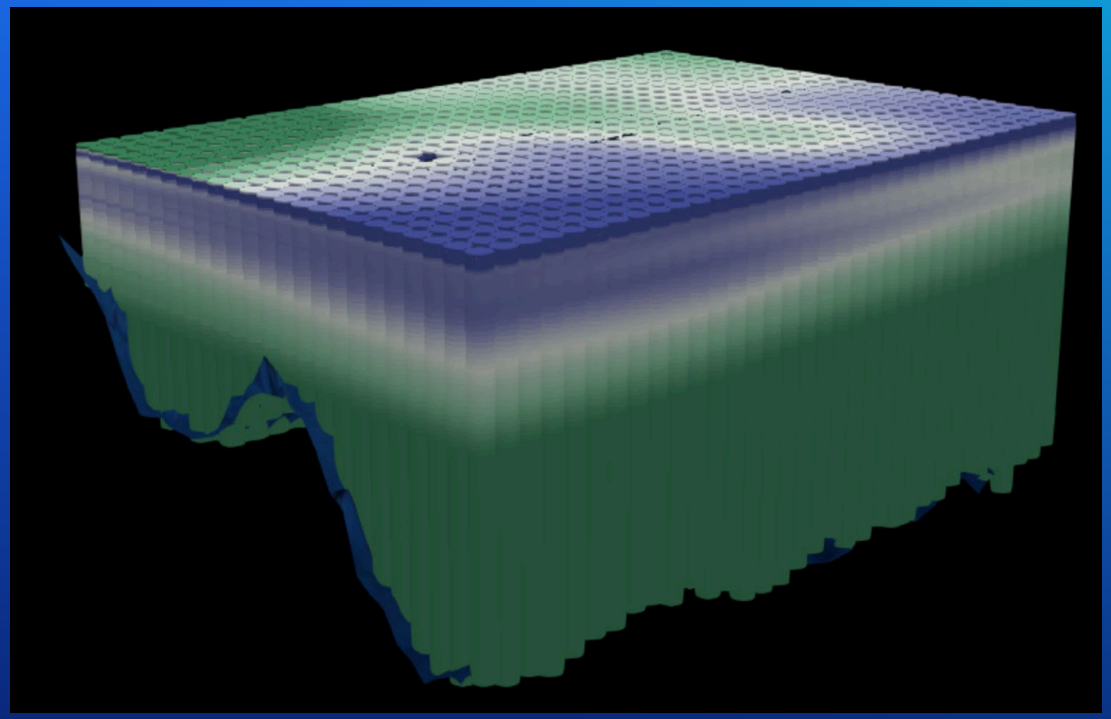
Sort by New

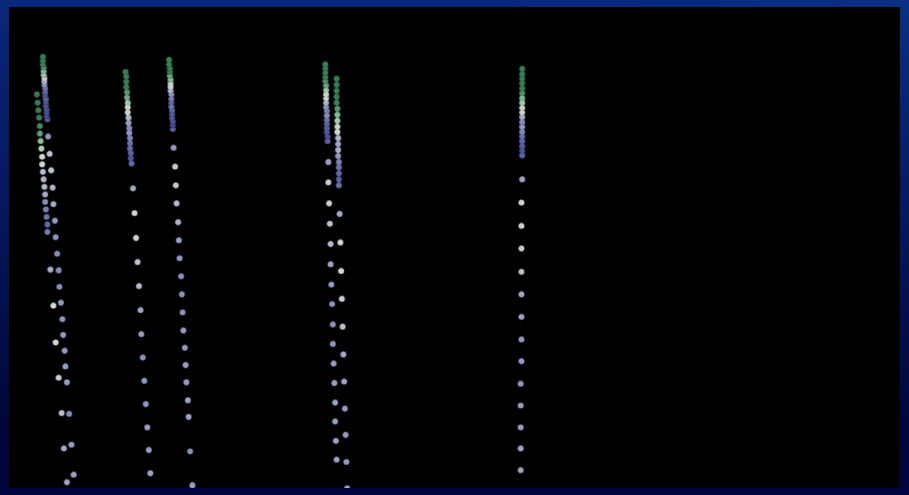
Credits (Attribution)

Goal: get the total download as small as possible

Know your data!

- 52 million points!
- > 4 GB of data
- Points at every ¼ degree of lat/lon
- Up to 102 points stacked at each location (from surface to ~5,000m depth)
- 12 attributes per point (temperature, salinity, phosphate, silicates, EMU cluster, etc.)





Goal: get the total download as small as possible

Step 1: Generalization

One point per x,y location

- Flatten the table
  - spatial join or
  - table pivot
- Reduced number of points to 677,109 features. Data is now one row per x,y location with fields for each attribute at each depth level.
- New problem – Now we have 1,224 fields.

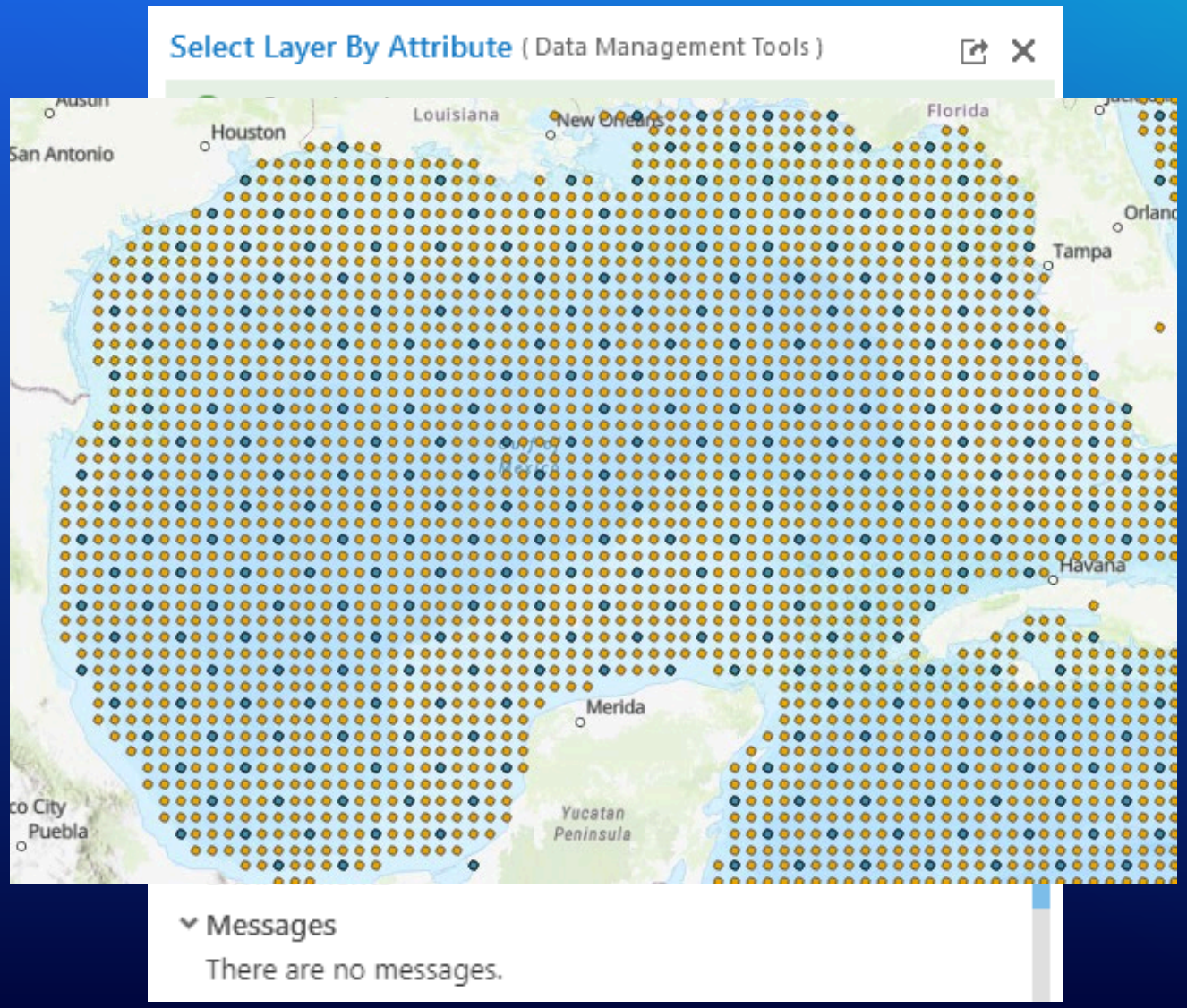
silicate	silicate_1	Single
depth_lvl	depth_lvl_1	Double
Cluster37	Cluster37_1	Integer
DistFromCluster37	DistFromCluster37_1	Double
Direction	Direction_1	Single
Velocity	Velocity_1	Single
pointid	pointid_2	Integer
temp	temp_2	Single
salinity	salinity_2	Single
dissO2	dissO2_2	Single
nitrate	nitrate_2	Single
phosphate	phosphate_2	Single
silicate	silicate_2	Single
depth_lvl	depth_lvl_2	Double
Cluster37	Cluster37_2	Integer
DistFromCluster37	DistFromCluster37_2	Double
Direction	Direction_2	Single

Goal: get the total download as small as possible

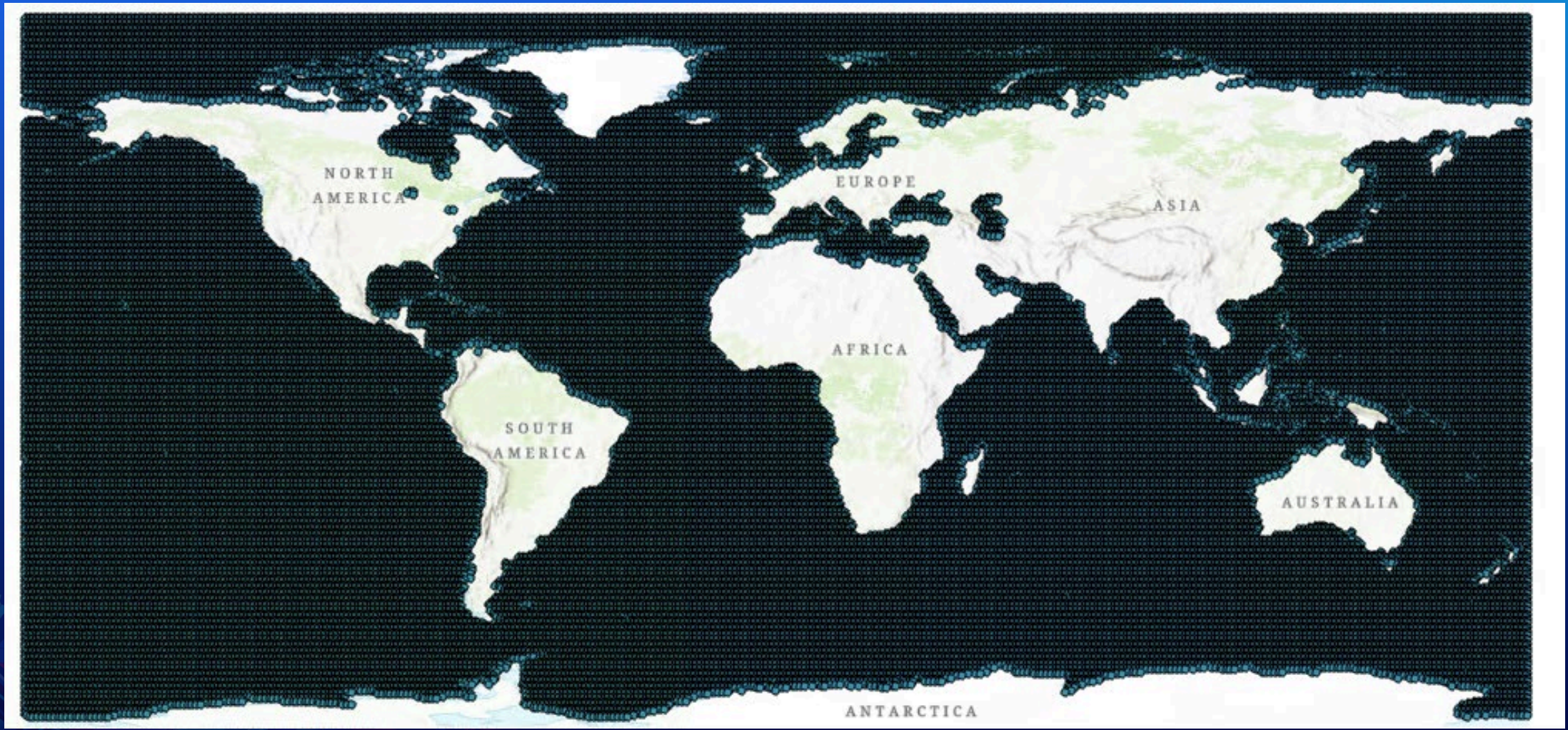
Step 2: Generalization

Geometry thinning

- Thin points to every ½ degree (¼ degree resolution is too high for worldwide views)
- Reduced number of points to 84,711 features! A reasonable number to work with.



Goal: get the total download as small as possible



Goal: get the total download as small as possible

Step 3: Reduce columns

Attribute thinning

- Feature Services limit you to 1,024 fields.
- I removed ~300 fields and was able to publish the service.
- Remaining problems:
  - 960 fields is still too much data to work with on the client.
  - Setting outFields to one attribute for every depth level results in huge query string requiring the POST method (not cacheable)

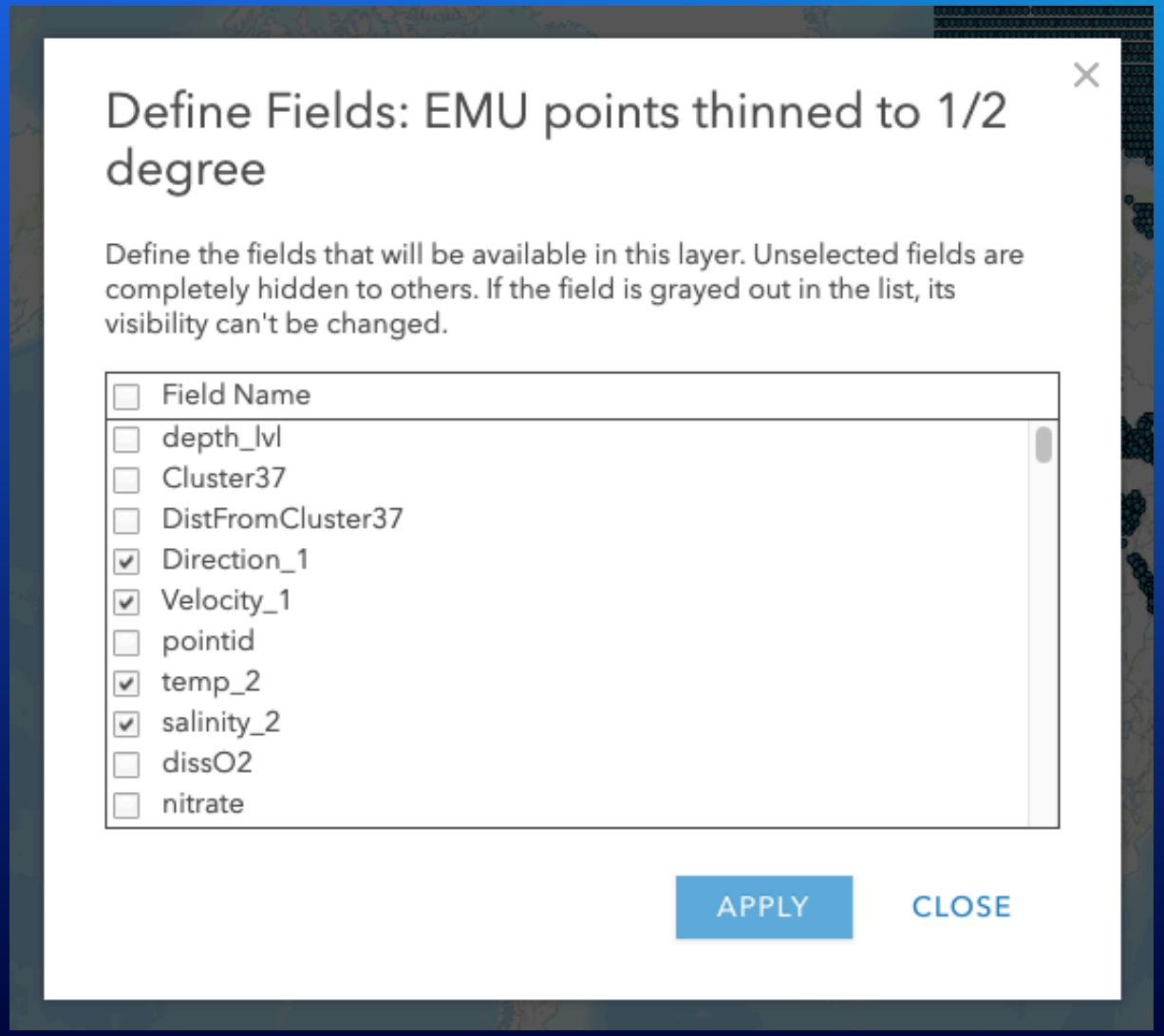
☐	Velocity	Velocity_79	Single
☐	pointid	pointid_80	Integer
☐	temp	temp_80	Single
☐	salinity	salinity_80	Single
☐	dissO2	dissO2_80	Single
☐	nitrate	nitrate_80	Single
☐	phosphate	phosphate_80	Single
☐	silicate	silicate_80	Single
☐	depth_lvl	depth_lvl_80	Double
☐	QtrDegreeID	QtrDegreeID_80	Integer
☐	Cluster37	Cluster37_80	Integer
☐	DistFromCluster37	DistFromCluster37_80	Double
☐	Direction	Direction_80	Single
☐	Velocity	Velocity_80	Single
☐	longitude	x_coord	Double
☐	latitude	y_coord	Double

Goal: get the total download as small as possible

Step 3: Reduce columns

Attribute thinning

- Create hosted Feature View and restrict fields to only the ones you need.
- Now we have ~300 fields (useable)
- Queries can request outFields using [“*”], which can be cached.
- Set maxAge on tiles to 1 hour.
- Now we have a manageable dataset we can work with in the browser!



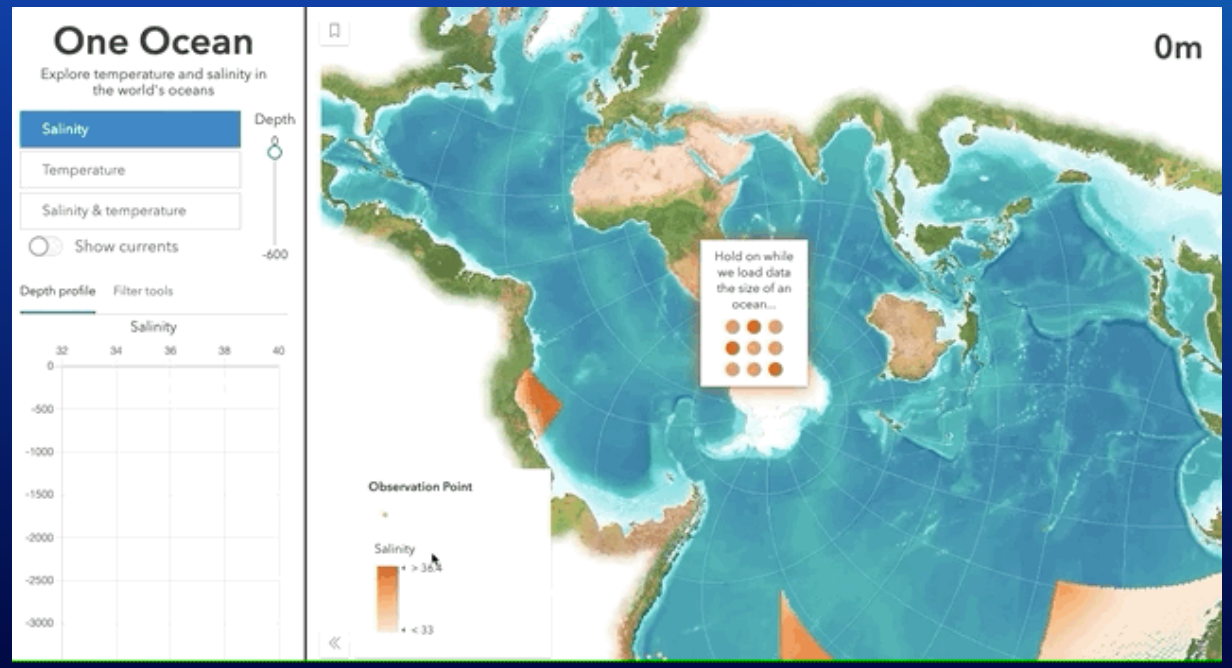
Goal: get the total download as small as possible

Step 4: Request data up front

For client-side goodness

```
const emuLayer = new FeatureLayer({
  title: "Observation Point",
  portalItem: {
    id: "06de2b96a7fa42d5891c4e79e17dd347",
    portal: {…
  },
  outFields: ["*"],
  popupEnabled: false
});
```

- Now we have a manageable dataset we can work with in the browser!
- Costs:
  - Still loads 280 MB of data!
  - Not useable for mobile devices
  - Initial load time is longer, but…
  - UX is a lot nicer once the data loads.



Conclusion

- The ArcGIS platform does a lot behind the scenes to optimize your data and apps when loading large abounts of data, but
- YOU can do more to optimize your apps and data for better performance
  - Choosing the right layer type
  - Visualization Techniques
    - Scale-dependent filtering
    - Clustering
  - Data prep/processing
    - Reducing/generalizing features and attributes
    - Aggregation

Please provide your feedback for this session by clicking on the session survey link directly below the video.

esri®

THE SCIENCE OF WHERE®

Copyright © 2021 Esri. All rights reserved.