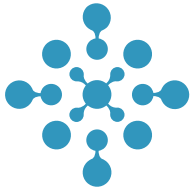


VIS 2019

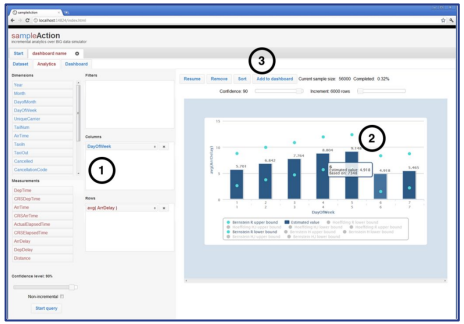
P5: Progressive Portable Parallel Processing Pipelines for Interactive Data Analysis & Visualization

Kelvin Li and Kwan-Liu Ma
University of California, Davis

Progressive Visual Analytics

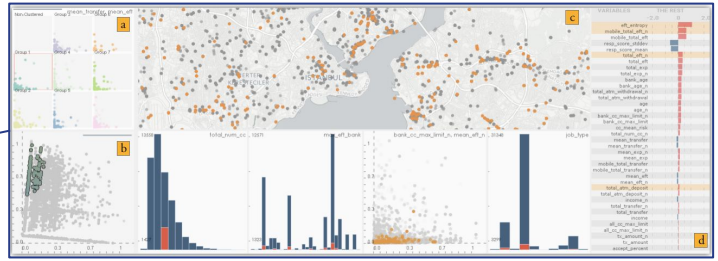
- Incrementally and interactively explore large datasets
  - Avoid long wait time for processing the entire dataset
  - Update the analysis results progressively
  - Allow the users to interact early and steer the analysis process

Research in Progressive Visual Analytics



Model & Frameworks

- Schulz et al. 2016
- Turkay et al. 2017

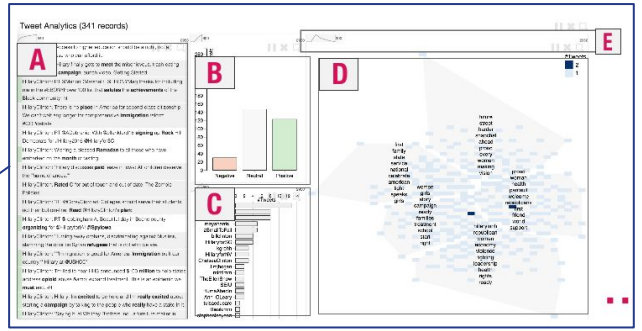


User Studies

- Fisher et al. 2012
- Zgraggen et al. 2017

Design Guidelines

- Stolper et al. 2014
- Muhlbacker et al. 2014
- Badma et al. 2017



Goal

A web-based visualization toolkit

- Declarative visualization grammar
- GPU computing
- Progressive data processing and visualization

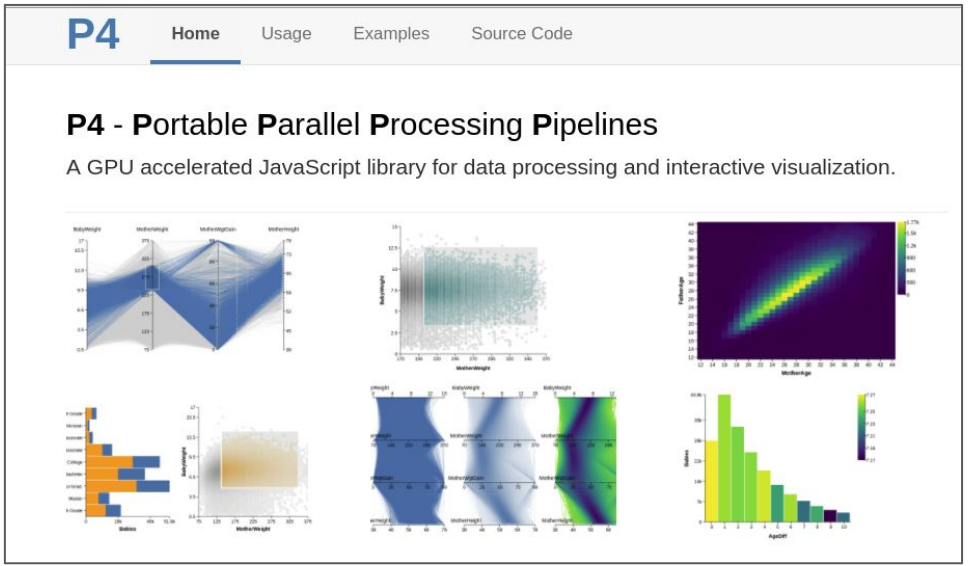
Motivation

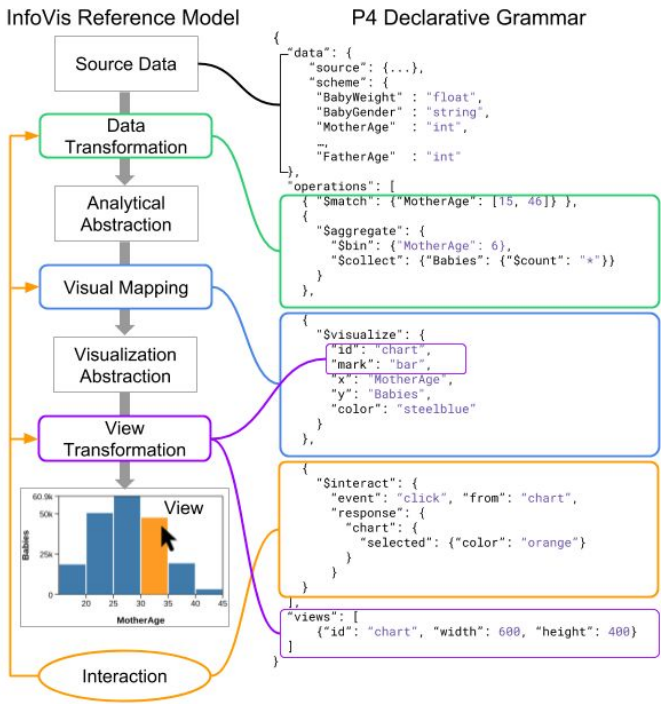
- Declarative grammar -> easier to create progressive visualization applications.
- GPU Computing + Progressive Processing ->
  - Process data that are large than GPU memory capacity
  - Provide progressive results at a faster rate

Declarative Grammar and GPU Computing for the Web

Provide ~20X speedup

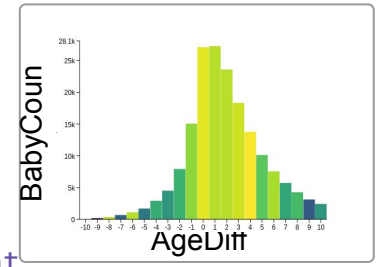
https://jpkli.github.io/p4/

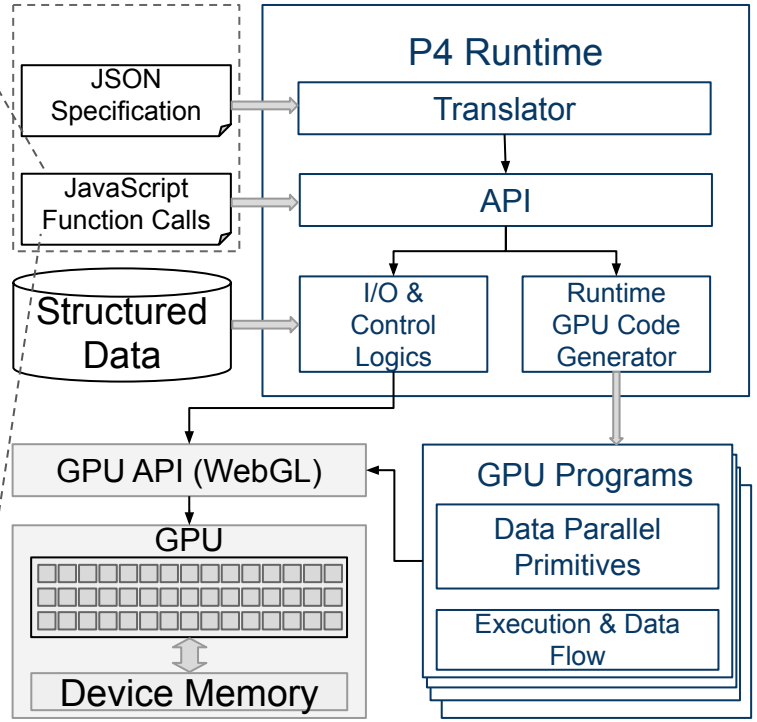




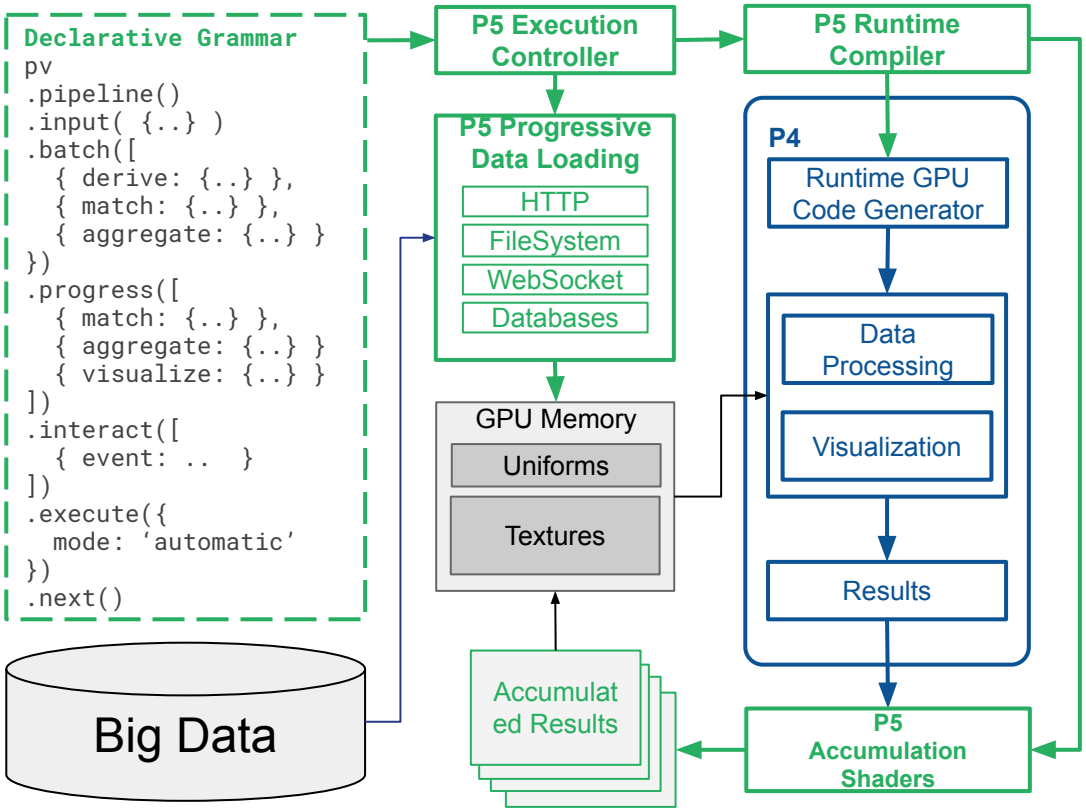
P4 Framework (Li & Ma TVCG 2018)

```
p4.data(...)
.derive({ AgeDiff: “FatherAge - MotherAge”})
.match({ AgeDiff: [-10, 10]})
.aggregate({
  $group: "AgeDiff",
  $collect: {
    BabyCount: { $count: "*" },
    AvgBabyWeight: { $avg: "BabyWeight" }
  }
})
.visualize({
    mark: "bar",
    x: "AgeDiff",
    y: "BabyCount",
    color: {
      field: “AvgBabyWeight”,
      scheme: “viridis”
    }
})
```



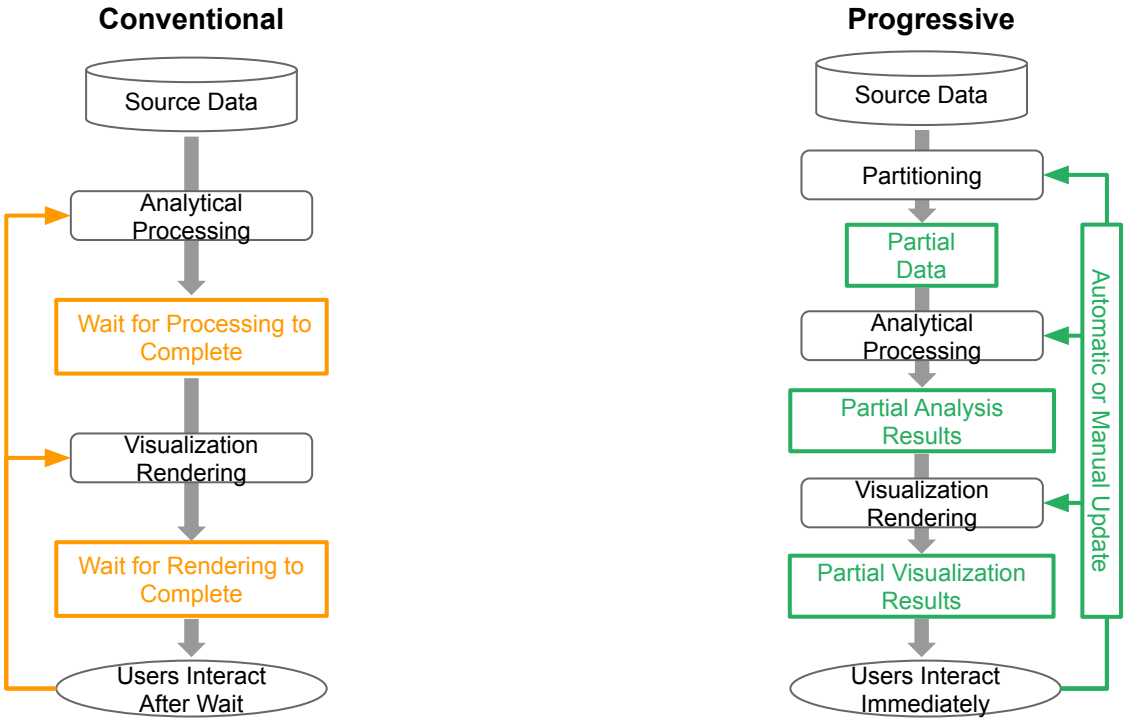


P5 System Architecture



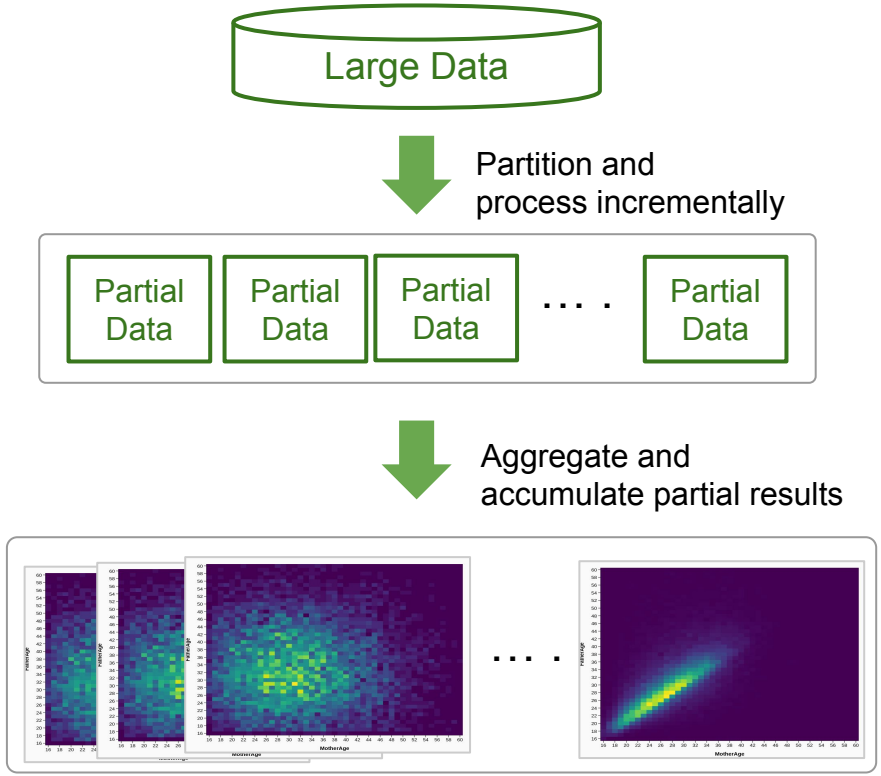
- Leverage P4 for parallel processing
- Accumulate progressive processing results using GPU
- Support progressive data loading and partitioning
- Provide intuitive API with declarative grammar

Conventional to Progressive Visualization Workflow



High-Level API for Progressive Visualization

```
p5.pipeline()
.input({
  source: 'http://.../data.csv',
  batchSize: 500000,
  type: 'text/csv',
  delimiter: ','
})
.batch([
  {
    match: {
      MotherAge: [18, 50],
      FatherAge: [18, 70]
    },
    aggregate: {
      $group: ['FatherAge', 'MotherAge'],
      $collect: {
        Babies: {$count: '*'}
      }
    }
  }
])
.progress([
  {
    visualize: {
      mark: 'rect',
      x: 'MotherAge',
      y: 'FatherAge',
      color: 'Babies'
    }
  }
])
.execute({mode: 'automatic'})
```



Supporting Interactions for Progressive Visualization

Interaction Specification

```
.interact({
 from: "view3", event: "brush",
 condition: {x: true, y: false},
 response: {
   view1: { unselected: {"color": "gray"} },
   view2: { unselected: {"color": "gray"} }
 }
})
```

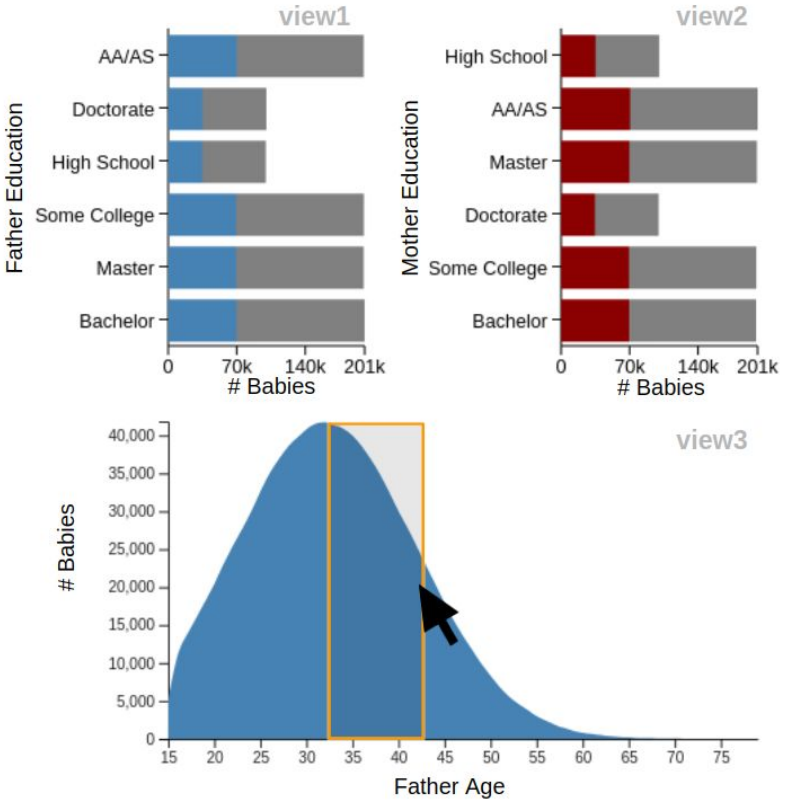
Data Cubes

Mother Education

Father Age	0	1	2
15	28	31	63
16	32	73	92
17	131	80	75

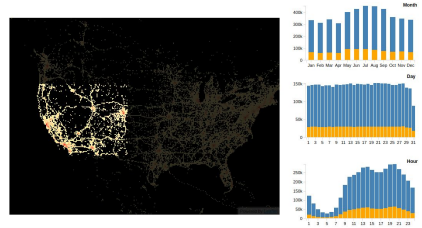
Father Education

Father Age	0	1	2
15	32	33	51
16	29	67	93
17	69	81	95



GPU-based Brushing-and-Linking in imMens

Liu, Zhicheng, Biye Jiang, and Jeffrey Heer. "imMens: Real-time Visual Querying of Big Data." *Computer Graphics Forum*. Vol. 32. No. 3. Oxford, UK: Blackwell Publishing Ltd, 2013.



a

```
let p = p5.pipeline(configs)
.input({
 method: 'file',
 source: '/home/p5/brightkitefile',
 schema : {
   uid: "int",
   lat: "float",
   lng: "float",
   time: "time"
 }
})
.views([
 {
   id: 'v1', offset: [0, 0],
   width: 760, height: 720,
 }, {
   id: 'v2', offset: [780, 0],
   width: 380, height: 240
 }, {
   id: 'v3', offset: [780, 240],
   width: 380, height: 240
 }, {
   id: 'v4', offset: [780, 480],
   width: 380, height: 240
 }
])
.preprocess({
  match: {
   lng: [-130, -66],
   lat: [22, 55]
})
```

b

```
p.batch([
 {
   aggregate: {
     $bin: [{lat: 256}, {lng: 256}],
     $collect: {values: {$count: '*'}},
   },
   out: 'map'
 },
 {
   derive: {hour: '$hour(time)'},
   aggregate: {
     $group: 'hour',
     $collect: {count: {$count: '*'}}
   },
   out: 'byHour'
 },
 {
   derive: {month: '$month(time)'},
   aggregate: {
     $group: 'month',
     $collect: {count: {$count: '*'}}
   },
   out: 'byMonth'
 },
 {
   derive: {DayOfWeek: '$dayOfWeek(time)'},
   aggregate: {
     $group: 'DayOfWeek',
     $collect: {count: {$count: '*'}}
   },
   out: 'byDayOfWeek'
 }
])
```

c

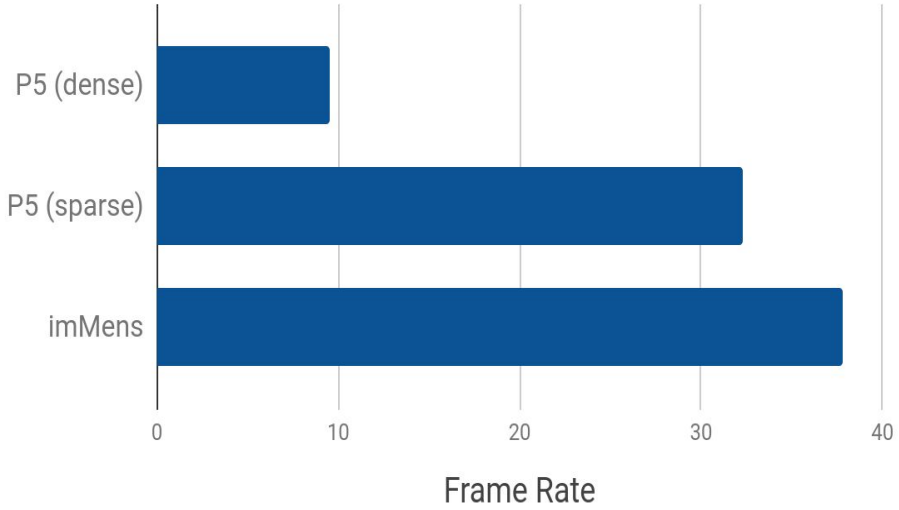
```
p.progress([
 {
   visualize: [
   {
     id: 'v1',
     in: 'map',
     project: 'geo',
     mark: 'circle', color: 'teal',
      x: 'lat', y: 'lng',
     color: {
       field: 'values',
       exponent: '0.33'
     }
   },
   {
     id: 'v2',
     in: 'byHour',
     mark: 'bar',color: 'teal',
     y: 'count', x: 'hour'
   },
   {
     id: 'v3',
     in: 'byMonth',
     mark: 'bar', color: 'teal',
     y: 'count', x: 'month',
   },
   {
     id: 'v4',
     in: 'byDayOfWeek',
     mark: 'bar', color: 'teal',
     y: 'count', x: 'DayOfWeek'
   }]
 }
])
```

e

```
p.interact([{
 event: "brush", from: "v1", response: {
   v2: { selected: { color: 'orange' } }, v3: { selected: { color: 'orange' } },
   v4: { selected: { color: 'orange' } }
  }
}])
```

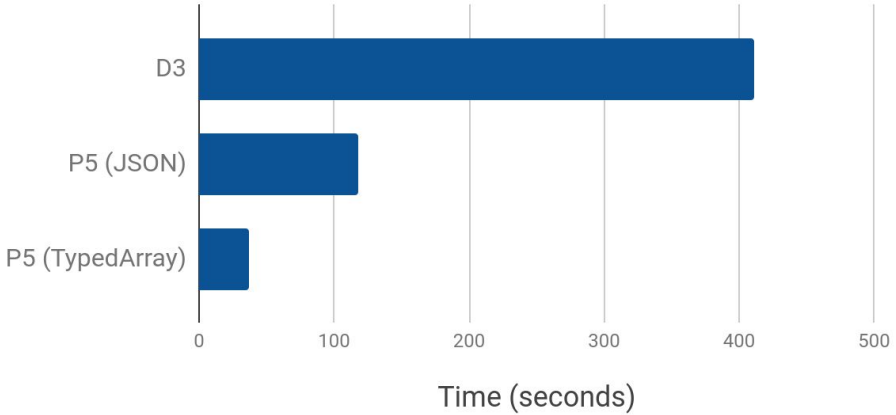
Performance Comparison with imMens

- Two different formats for storing and processing data cubes (**dense vs. sparse)** in P5
- Codes: **~100 lines** in P5 vs. more than a **thousand lines** in imMens



Performance Benchmark

- Progressive visualization of 100 million data records.
- Used two different input format: JSON vs. TypedArray.
- ~3 to 10 X better performance than D3.



Summary

A first step to provide a progressive visualization toolkit with declarative grammar and GPU computing.

Future work:

- Extend and improve our API.
- Support more progressive analytics operations, such as clustering and dimensionality reduction.
- Provide easy integration with other data analytics tools.

Source Codes and Demos:

PV: https://github.com/jpkli/pv

P4: https://github.com/jpkli/p4

Acknowledgement

This research is supported in part by the National Science Foundation via grant IIS-1528203 and the Department of Energy via grant DE-SC0014917.

Thank You!