PROCESSING AND VISUALIZING DATA

IN A CONTINUOUS COLLECTION ENVIRONMENT

BYRON ELLIS

SPONGECELL, INC

AN OBSERVATION

You know a lot about your compute environment!

WE DON'T...

- Only vague information about the physical layout
- Unreliable interconnect between nodes
- Huge node-to-node performance variability and availability

BUT WE TRY TO...

- Move about ~500M events/day
- ~5k to ~12k events/second
- From several different regions around the world
- All as quickly as we can manage it (usually a few seconds)

DATA APPLICATIONS

- Monitoring
- Reporting
- Modeling/Optimization

TWO PIECES OF THE PUZZLE

- Data Motion
- Lightweight Visualization

DATA MOTION

FAULT TOLERANCE IS NOT OPTIONAL

MASSIVE REDUNDANCY IS A GIVEN

DECOUPLE PROCESSES AS MUCH AS POSSIBLE

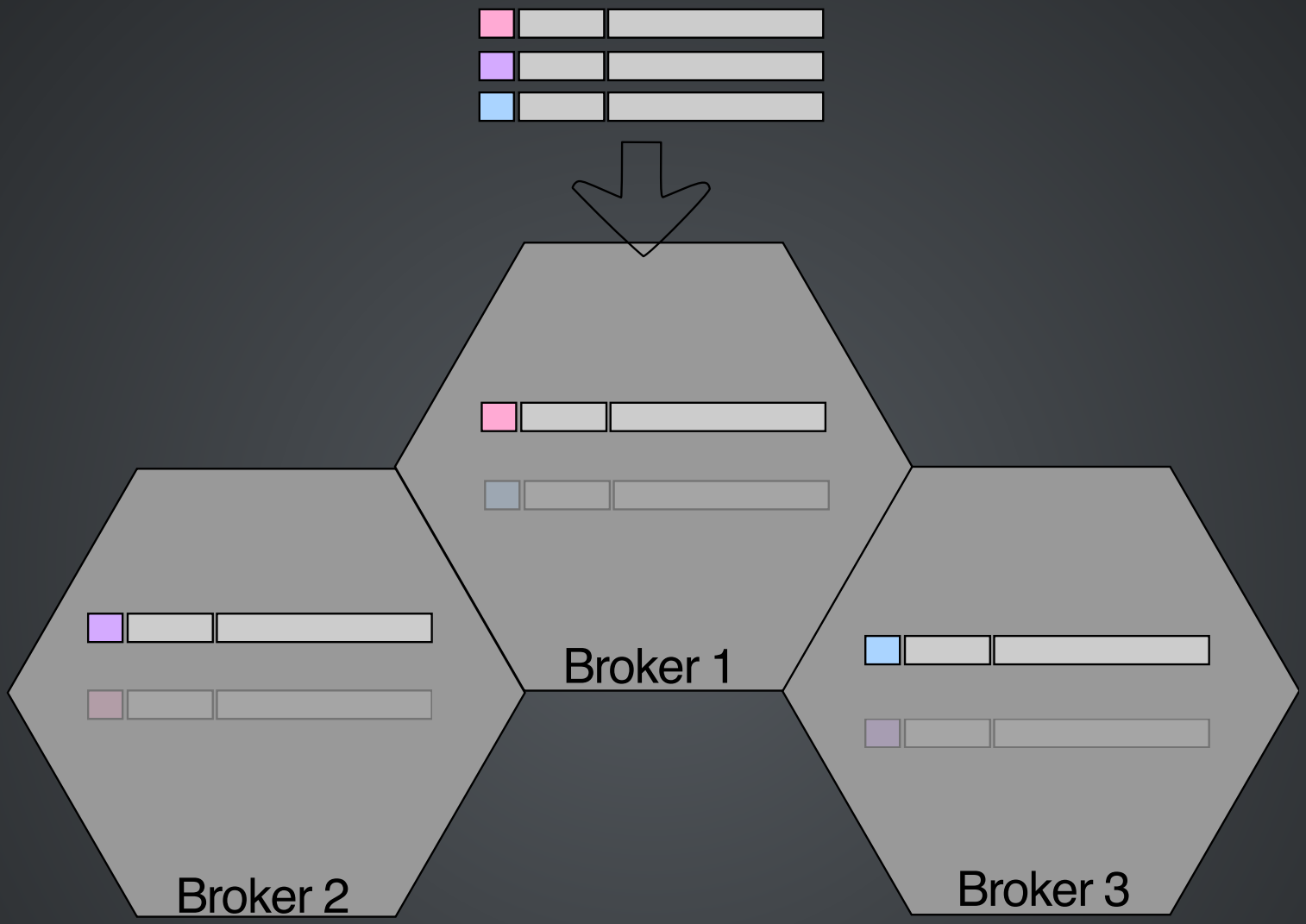
KAFKA: WRITE AHEAD LOG AS A SERVICE

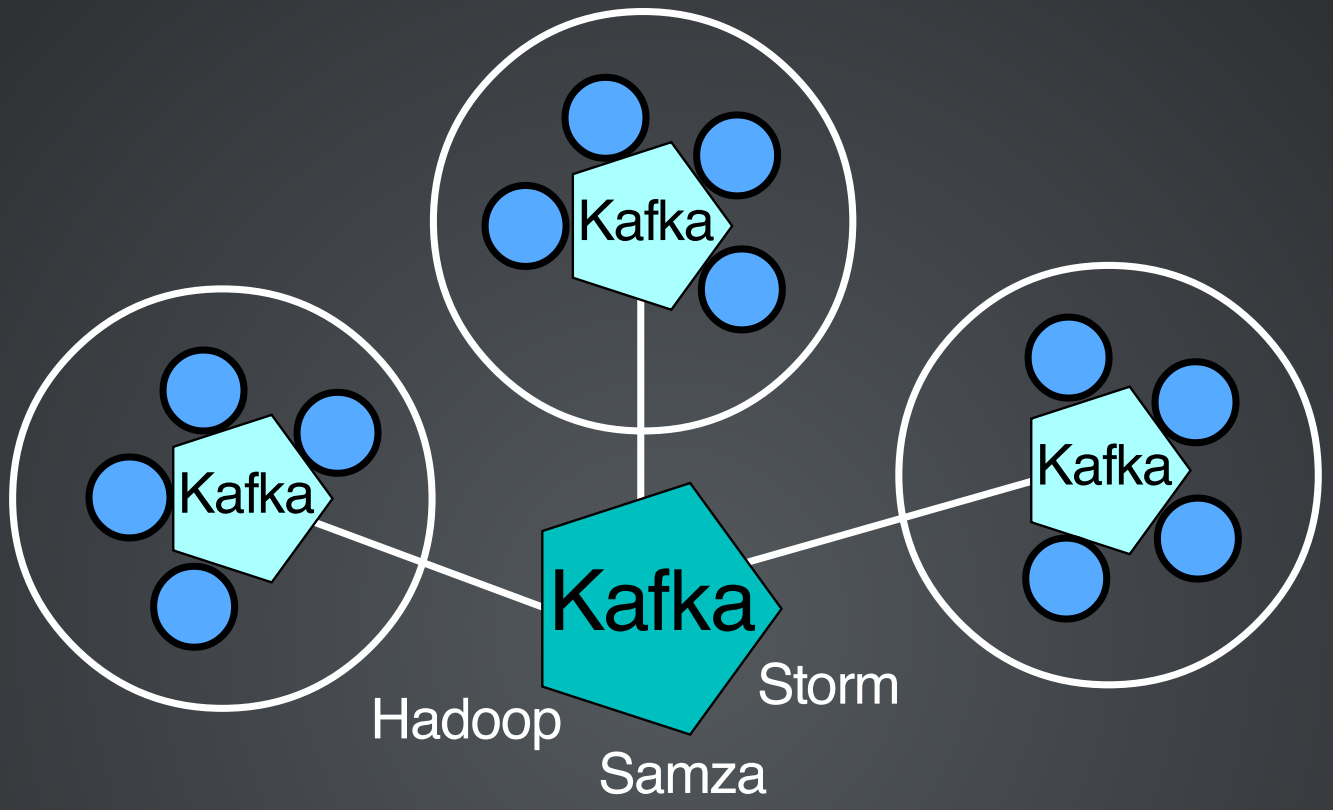
FEATURES

- Partitioned storage of key-value messages
  - Streams of similar data arranged into topics
  - Topics divided into partitions, spread across physical brokers
  - Random by default, but key-based and custom partitioning is easy
- User selectable internal replication of messages
- New partitions and brokers (servers) can be added on-the-fly
- Data is retained for a fixed amount of time. Storage management is your problem.

LOG STRUCTURE

1	Key	Value
2	Key	Value
3	Key	Value
...		
n	Key	Value


Broker 1
Broker 2
Broker 3

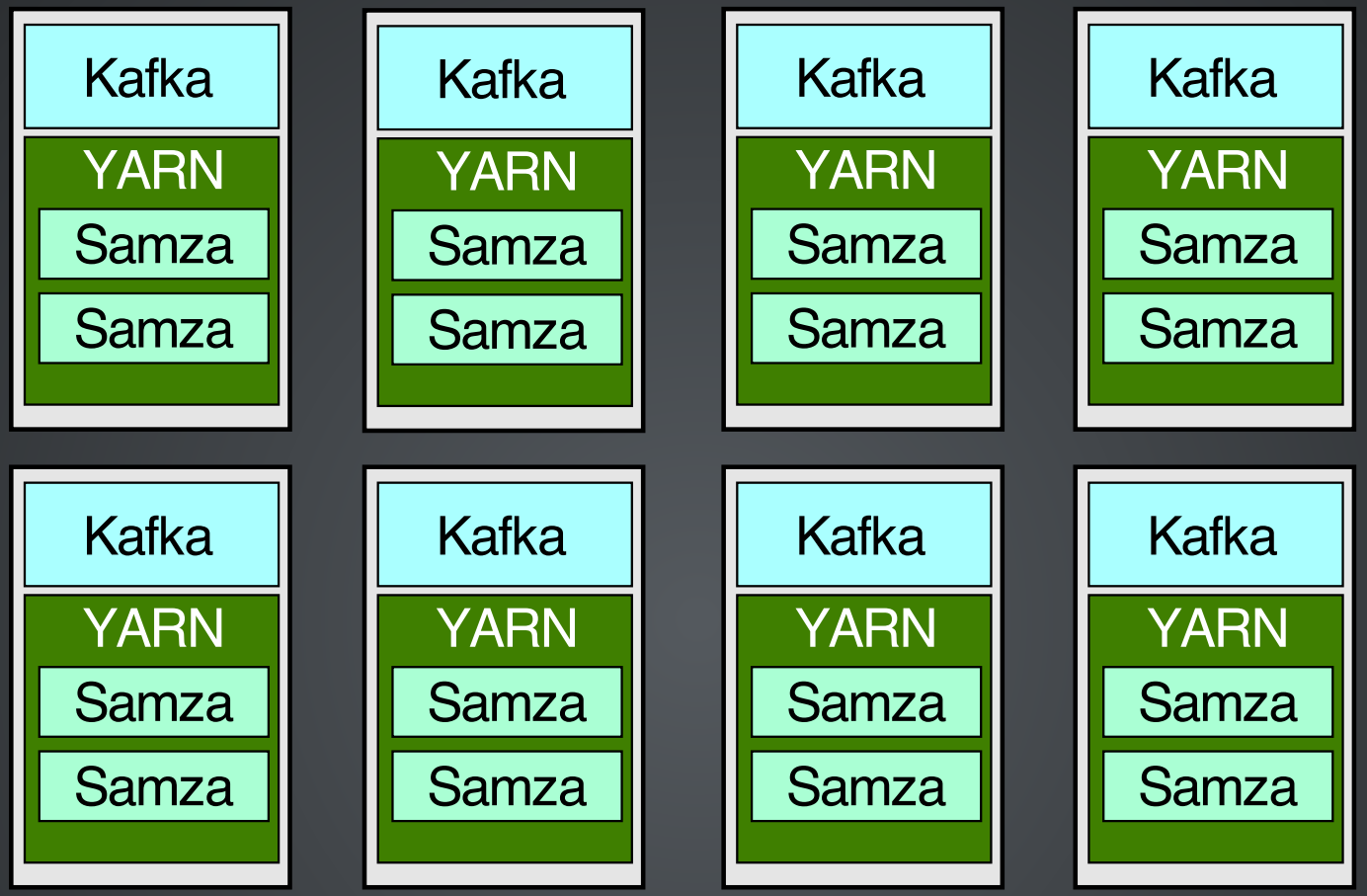

Kafka
Kafka
Kafka
Kafka
Hadoop
Storm
Samza

TRADITIONAL PROCESSING PARADIGM

- Regular jobs to ingest from Kafka to parallel filesystem (usually HDFS)
- Extract and Transform
- Model directly on output or load into persistence stores
- Robust, easy to use, slow

AN INGEST RUN IS COMPLETELY DEFINED BY THE ID RANGE FOR EACH PARTITION

STILL PLENTY OF COMPUTE ON THESE BOXES


Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza
Kafka
YARN
Samza
Samza

STREAMING COMPUTE GRID

- YARN is "Yet Another Resource Manager" (aka Hadoop 2)
- Samza is a tool for writing jobs that read from one Kafka topic and write to another.
- Batch jobs can consume output of streaming jobs for persistence and other tasks
- Processing is managed such that all messages have "at least once" processing semantics

LIGHTWEIGHT VISUALIZATION

MOTIVATION

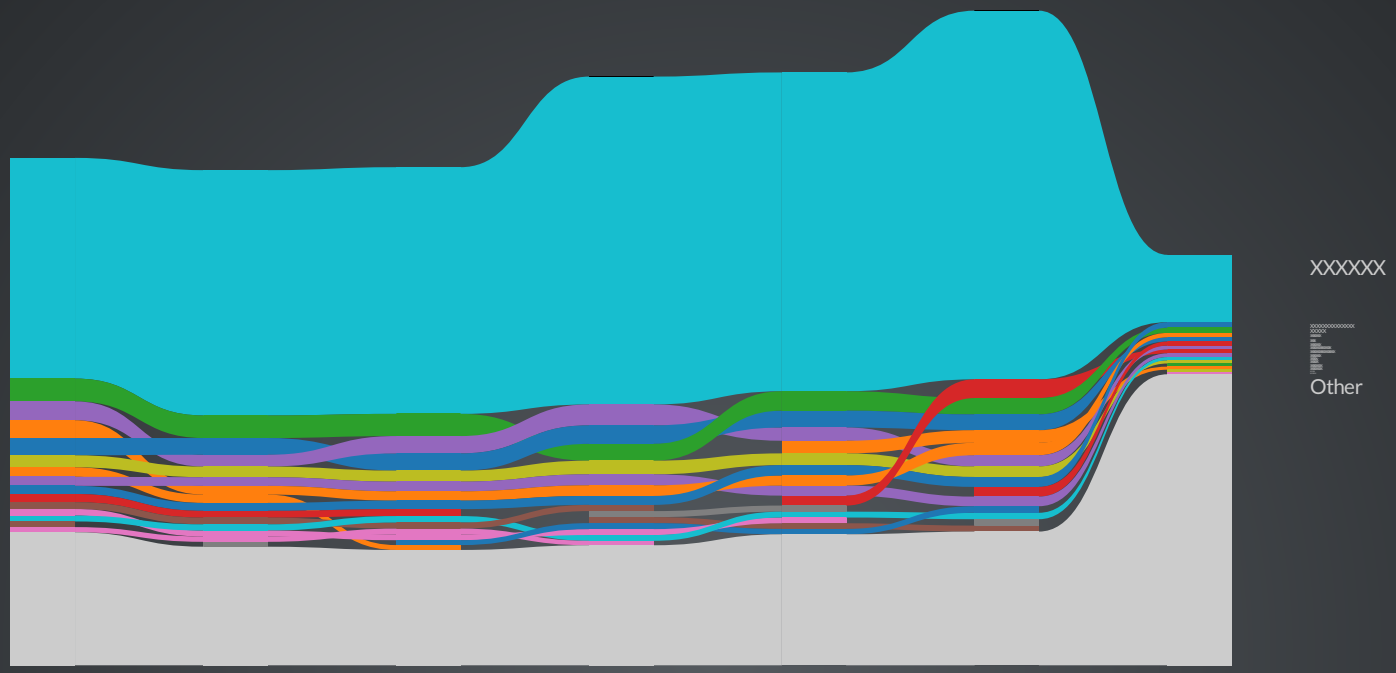
Not much point in doing streaming data processing if you can't use it for something.

THE BAD NEWS...

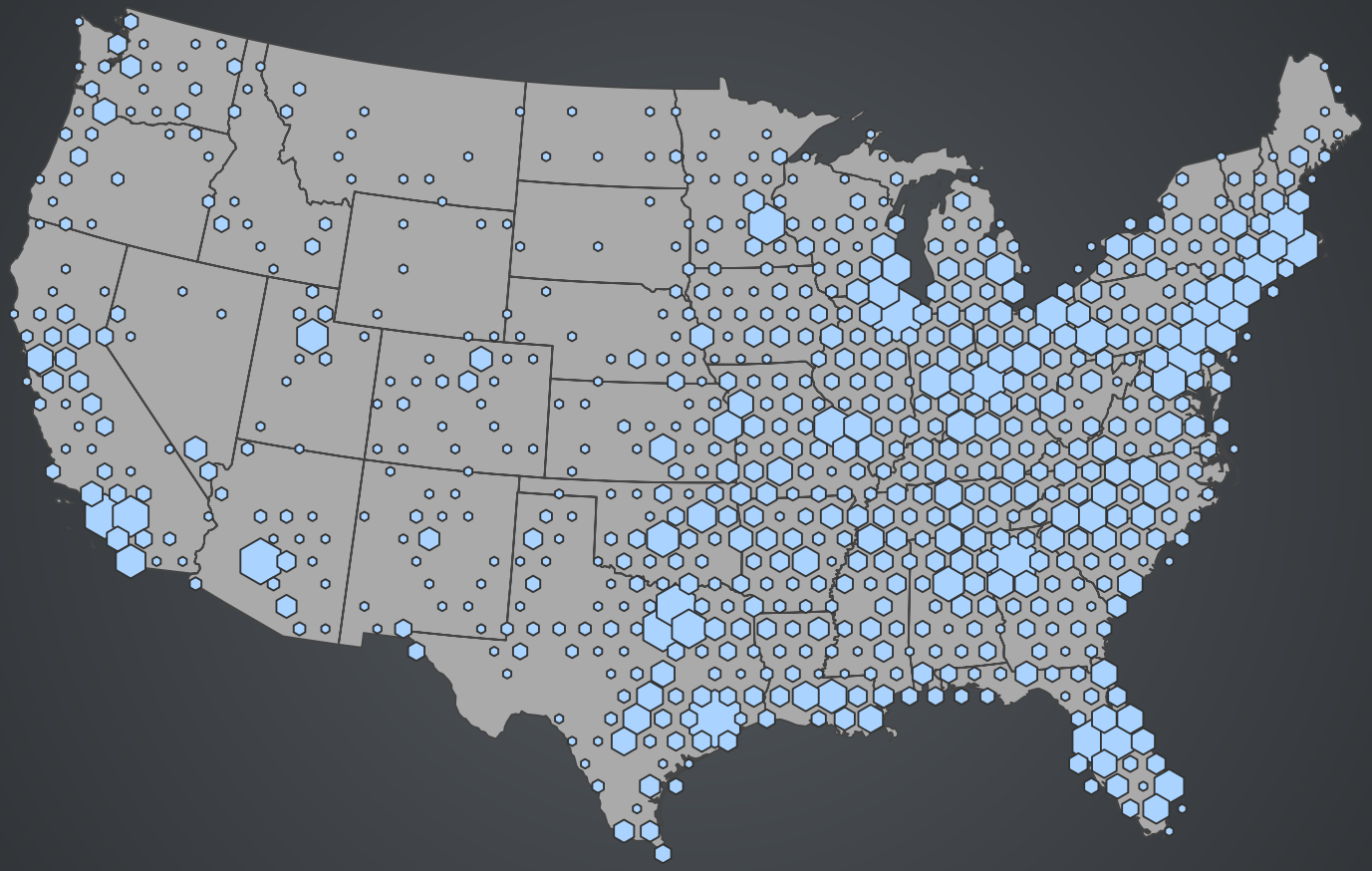
- Limited to the web browser
- Support for features is not uniform
- Mobile support is no longer optional (no plugins)

THE GOOD NEWS...

- Javascript is getting (relatively) fast.
  - 100k data points for desktop class
  - 10k for tablet/phone class
- Good support for retained and immediate-mode 2D rendering.
- Growing support for native 3D visualization (WebGL).
- Streaming links between server and client allow split computation
- Pretty good libraries for implementing interesting visualization


XXXXXX
Other

All All



POUROVER.JS

- Developed by the New York Times
- Provides sorting and filtering for categorical data
- Designed to operate on ~100k records

INITIALIZE POUROVER

```
d3.tsv("walmarts.tsv",function(data) {
      data = data.map(function(x) {
        var d = projection(x);
        var m = x.date.split("\/");
        var x = {x:1.0*d[0],y:1.0*d[1],month:1*m[0],day:1*m[1],year:1*m[2]};
        var d = new Date(x.year,x.month-1,x.day);
        x.dow = dows[d.getDay()];
        return x;
      });
      var po  = new PourOver.Collection(data);
      po.addFilters(PourOver.makeExactFilter("dow",dows));
      po.addFilters(PourOver.makeRangeFilter("year",[[1960,1969],[1970,1979],
```

UPDATE FILTER

```
var dow = d3.select('select#dow');
        dow = d3.select(dow.selectAll("option")[0][dow.node().selectedIndex])
        if(dow == "") {
          po.filters.dow.clearQuery();
        } else {
          po.filters.dow.query(dow);
        }
        //...
        map.update();
```

D3.JS

- "Data Driven Documents"
- Binds an array of data to an array of DOM elements
- Supplies a number of helper functions for different applications
- Has a number of plugins for more complicated applications

SCALE MANAGEMENT

```
var radius = d3.scale.sqrt()
        .domain([0, 12])
        .range([0, 8]);
```

GEOGRAPHICAL PROJECTIONS

```
var projection = d3.geo.albers()
      .scale(1000)
      .translate([width / 2, height / 2])
      .precision(.1);
```

DATA LAYOUT

```
var hexbin = d3.hexbin()
        .size([width, height])
        .x(function(d) { return d.x; })
        .y(function(d) { return d.y; })
        .radius(8);
```

PATH GENERATION

```
var path = d3.geo.path()
      .projection(projection);
```

DRAWING THE MAP

```
d3.json("us.json",function(us) {
      back.append("path")
        .datum(topojson.feature(us,us.objects.land))
        .attr("class","land")
        .attr("d",path);

      back.append("path")
        .datum(topojson.feature(us,us.objects.states,
          function(a,b) { return a != b; }))
        .attr("class","state")
        .attr("d",path);
    });
```

UPDATING THE HEXBINS

```
var data = hexbin(this.getCurrentItems());
      var h = hex.selectAll("path.hex")
      .data(
        data.sort(function(a,b) { return b.length - a.length; })
      );

      h.exit().remove();

      h.enter().append("path").attr("class","hex");

      h
      .attr("d",function(d) {
        return hexbin.hexagon(radius(d.length))
      })
      .attr("transform",function(d) { return "translate("+d.x+","+d.y+")"; })
```

OTHER D3 FEATURES

- Many layouts: hierarchical, force directed, circle packing, etc
- Path generators allow for complicated structures like flow charts
- Animation and transitions when new data arrives

INTEGRATING WITH DATA ENVIRONMENT

- Two primary methods: WebSockets and Server Sent Events
- Both allow data to be streamed asychronously to the client from the server
- WebRTC may also be an option. Designed to stream high-bandwidth payloads such as audio and video
- All allow for event-driven programming

FINAL THOUGHTS

HPC AND BIG DATA, MORE SIMILAR THAN DIFFERENT

- 'Cutting Edge' Big Data Compute looks a lot like MPI
- Failure Recovery is a big deal (message logging is the big data default)
- Compute loads are getting more varied
- Execution environments are getting more varied

A LOT OF EXCHANGE POINTS

- Resource Managers
- Deployment Techniques
- Exotic data structures
- Building tools that let non-experts use our systems
- Geographic distribution of compute?

THANK YOU

http://github.com/byronellis/salishan2014