Graphistry
Scaling Visual Exploration with GPUs and Design

Leo Meyerovich (@LMeyerov)
CEO of Graphistry.com | UC Berkeley
Today: Small or Static Visualizations
Ballot Boxes: 100K rows x 30 col CSV
Most towns had ~40% people vote

ballot box stuffing?
Demo: Voter Precinct by Party & Turnout

Ruling Party: 50%

Legend

- Small square: voting precinct
- Square size: votes cast
- Red: town voted for the incumbent
- Black borders: different districts
Tiny square shows town size (area) and vote (color)
Filter for towns w/ high turnout
Tag suspicious with black
Analyze suspicious activity in context
What parts of the supply chain were hit?
A slider is worth a hundred queries.
H/s  Hypotheses per second

100X more visibility over space and time?
GPUs are Chainsaws

Parallel: 2X slower / thread, but 100X+ more threads

Skip JavaScript: low-level shaders / OpenCL
Announcing New Amazon EC2 GPU Instance Type

Posted On: Nov 4, 2013
Uber: Trip Start to End
Direct Edge Placement: Overplotting
Speed ➔ Design ➔ Edge Bundling
Distributed Browser w/ GPU Cloud Streaming

**Cloud**
- WebCL Layout
- filter, physics, ...

**Browser**
- WebGL Renderer
- paint pixels

**GPU**
- 1,536 cores
- 4GB memory

**GPU**
- 384 cores
- 1GB memory
Distributed Browser w/ GPU Cloud Streaming

Cloud

15 fps

WebCL Layout
filter, physics, ...

1,536 cores
4GB memory

shapes

commands

The internet is now the interconnect!

Browser

IE, Chrome, iOS, ...

60 fps

WebGL Renderer
paint pixels

GPU
384 cores
1GB memory
Toolchain for Creating & Customizing Visualizations

Superconductor language, graph layout engine, OpenGL streaming runtime, …
The Future of Data Visualization

Increase visibility over space and time by 100X using GPU cloud streaming
We’re hiring: web infoviz, data/ops eng (& come chat about embedding!)

Leo Meyerovich (@LMeyerov)
CEO of Graphistry.com | UC Berkeley
BACKUP
Ex: Time Series in IBM's IT Monitor
GE Demo
Page Load CPU Activity

no network

real-time is 30ms

Data Parsing & Preparation

Render

Layout
event loop is now 80ms & non-blocking

Multicore + GPU Acceleration
SUPERCONDUCTOR
Client Only Version

webpage

HTML data
CSS styling
JS script

Renderer

Pixels

Parser.js
Parser

Selectors.CL
Selectors

Layout.CL
Layout

Renderer.GL
Renderer

 GPU

superconductor.js

Data stays on GPU!

data viz
data
styling
styling
widgets
widgets

Compiler
Grammar 1/2: *Schema of Attributed Tree*

Webpage = Tree

Node attributes
Grammar 2/2: Schema Constraints

1. Local

\[ HBox \rightarrow \text{left}=HBox, \text{right}=\text{Leaf} \]

\[ w = \text{left}.w + \text{right}.w \]

2. Single-assignment

[Kastens 1980, Saraiva 2003] [WWW 2010, PPOPP 2013]
Linguistic Extensions to Attribute Grammars

**Productivity**

Nominal typing (Objects):

```plaintext
interface Node { var x }
class HBox : Node { x := 1 }
```

Traits for code reuse:

```plaintext
trait Paint { x := 1 }
class HBox(Paint) : Node { }
```

Macro-expandable

**Expressivity**

Declarative loops:

```plaintext
w := id + 1
loop c in children:
    c[i].id := c[i-1].id + 1
    c[i].x := c[i].w
```

Reduce to AG scheduling:

1. unroll: iteration 0, 1, 2, 3, 4
2. schedule
3. contract: loop body == iteration 2

Partial behavioral specification:

Section 3 of talk 😊
Solution: Layout as Tree Traversals

\[
\text{sched} = \{w, h\}; \quad \{x, y\}; \quad \{\ldots\}; \quad \ldots
\]

---

Leaf

Solution: Layout as Tree Traversals

- Logical joins
- Logical spawns
- Constraints on node attributes

Document tree

Can rearrange into 9 parallel passes!

\[w_0 = f(w_1, w_2)\]

\[h_0 = \max(h_1, h_2)\]

\[\text{sched} \quad \text{valid for all webpages (in subset):}\]

Schedule is static -- used to build the browser!
GPU Traversals: Flattened & Level-Synchronous

Compiler automates code + data transformations.