



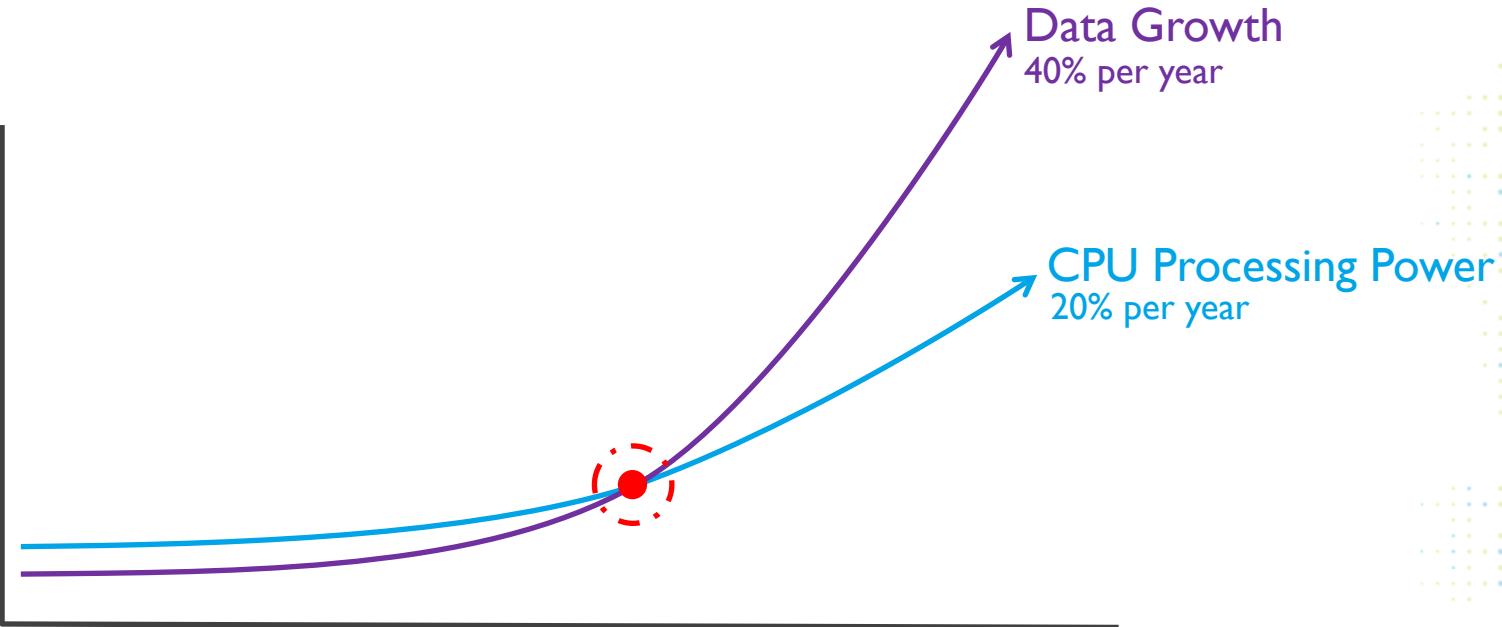
LEVERAGING GPU-ACCELERATED ANALYTICS ON TOP OF APACHE SPARK

June 6, 2017

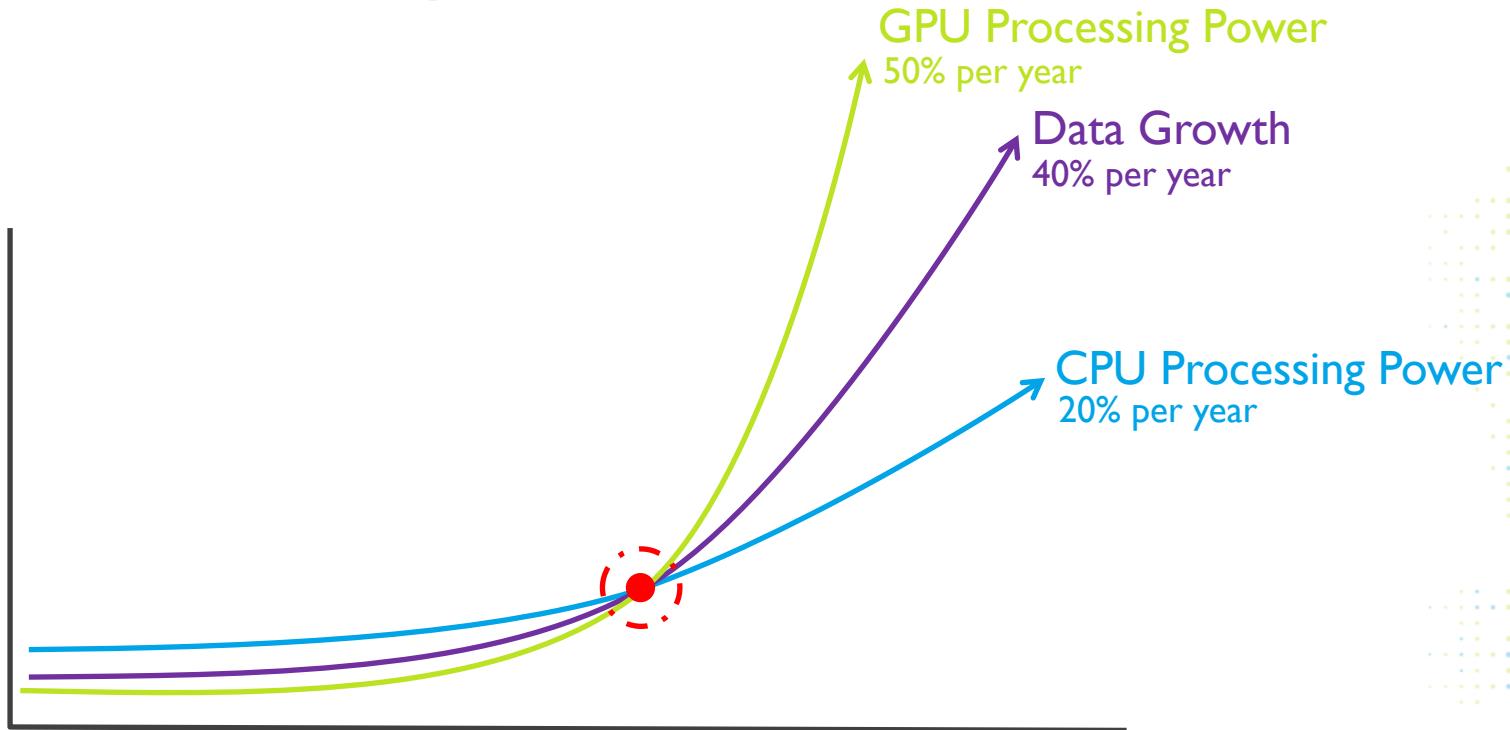
Todd Mostak | Co-founder + CEO, MapD

@toddmostak | @mapd

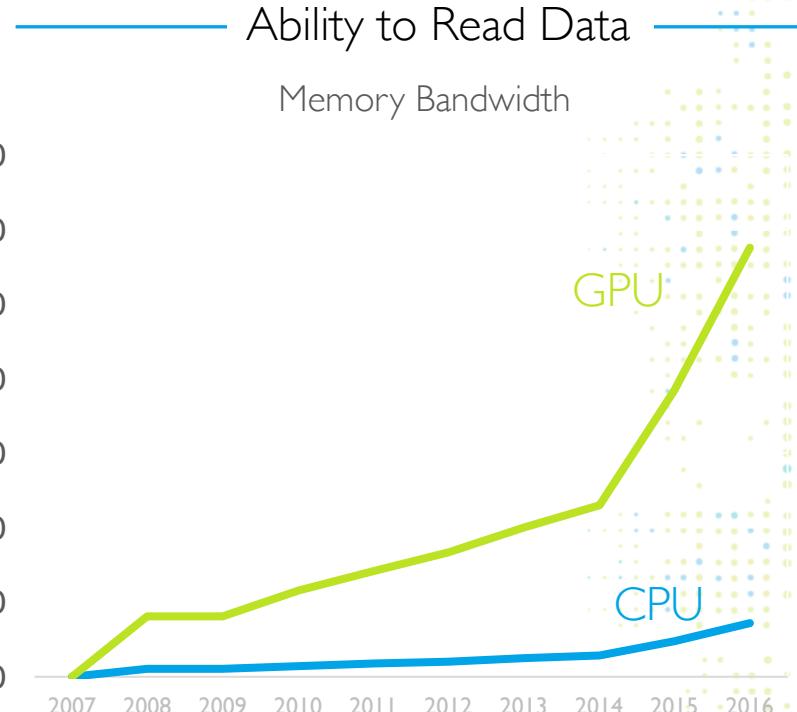
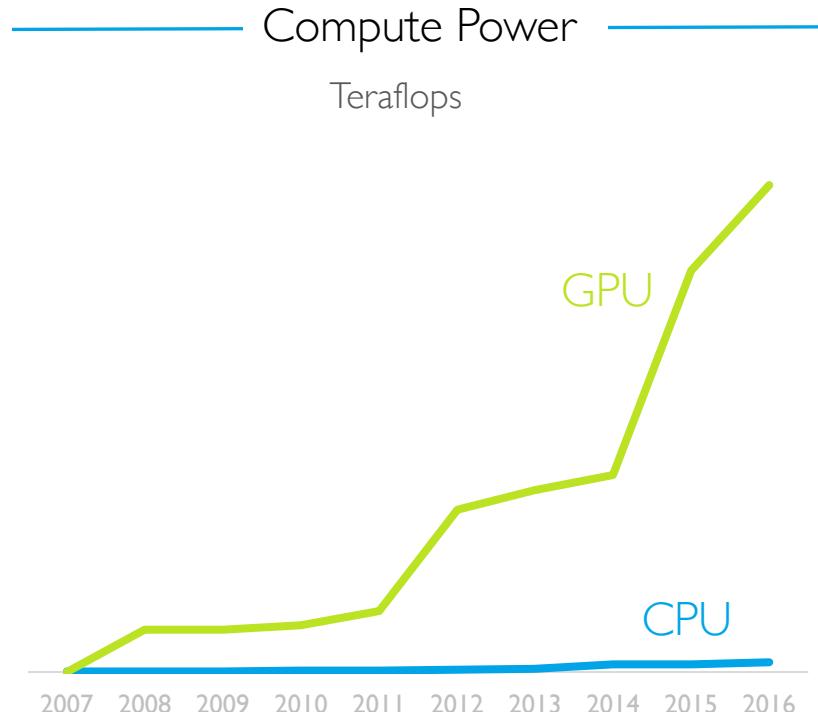
A compute inflection point



GPUs offer a way forward

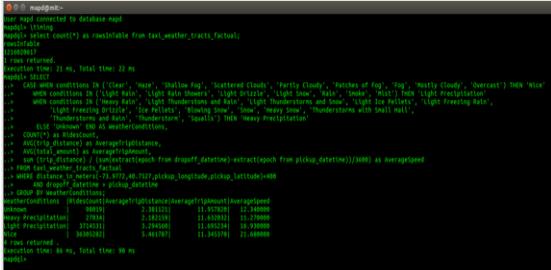


GPUs outperform CPUs in data critical areas



MapD: software optimized for the fastest hardware

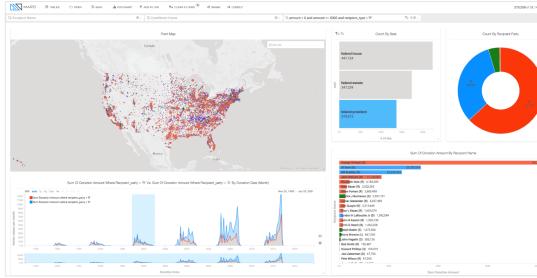
100x Faster Queries



```
mapd@mapd:~$ explain
+-----+
| User mapd connected to database mapd |
| mapdsql [1]: select count(*) as rowsavailable from text_weather_tracts_factual; |
| rowsavailable |
|-----|
| 1 rows returned, |
| execution time: 21 ms, Total time: 22 ms |
|-----|
| 1. CASE WHEN conditions in ('Clear', 'Haze', 'Shallow Fog', 'Partly Cloudy', 'Patches of Fog', 'Fog', 'Mostly Cloudy', 'Overcast') THEN 'Wise' |
| 2. CASE WHEN conditions in ('Light Rain', 'Rain', 'Heavy Rain', 'Thunderstorms', 'Light Thunderstorms and Rain', 'Heavy Thunderstorms and Rain', 'Light Precipitation', 'Snow', 'Heavy Snow', 'Thunderstorms with Small Hail', 'Light Hail', 'Rain and Hail', 'Snow and Hail', 'Thunderstorms with Large Hail', 'Heavy Hail', 'Heavy Precipitation') |
| 3. ELSE 'Unknown' END AS weatherconditions, |
| 4. COUNT(*) AS count |
| 5. AVG(lat) AS averageLat |
| 6. AVG(lon) AS averageLon |
| 7. AVG(lat) * AVG(lon) AS avgLatLon |
| 8. SUM(lat*distance) / COUNT(extractepoch from dropoff_datetime)-extract(epoch from pickup_datetime))/1000) AS averagedep |
|-----|
| FROM text_weather_tracts_factual |
| WHERE 1=1 |
| AND dropoff_datetime > pickup_datetime |
| AND dropoff_datetime <= pickup_datetime |
|-----|
| weatherconditions | count |
|-----|
| Unknown | 99931 |
| Light Rain | 2,38123 |
| Light Thunderstorms and Rain | 31,95703 |
| Light Precipitation | 3,94523 |
| Heavy Rain | 2,38123 |
| Thunderstorms with Small Hail | 1,36423 |
| Light Hail | 0,36423 |
| Rain and Hail | 0,36423 |
| Snow and Hail | 0,36423 |
| Thunderstorms with Large Hail | 0,36423 |
| Heavy Hail | 0,36423 |
| Heavy Precipitation | 0,36423 |
|-----|
| 1 rows returned |
| execution time: 20 ms, Total time: 20 ms |
|-----|
```



Speed of Thought Visualization



MapD Core

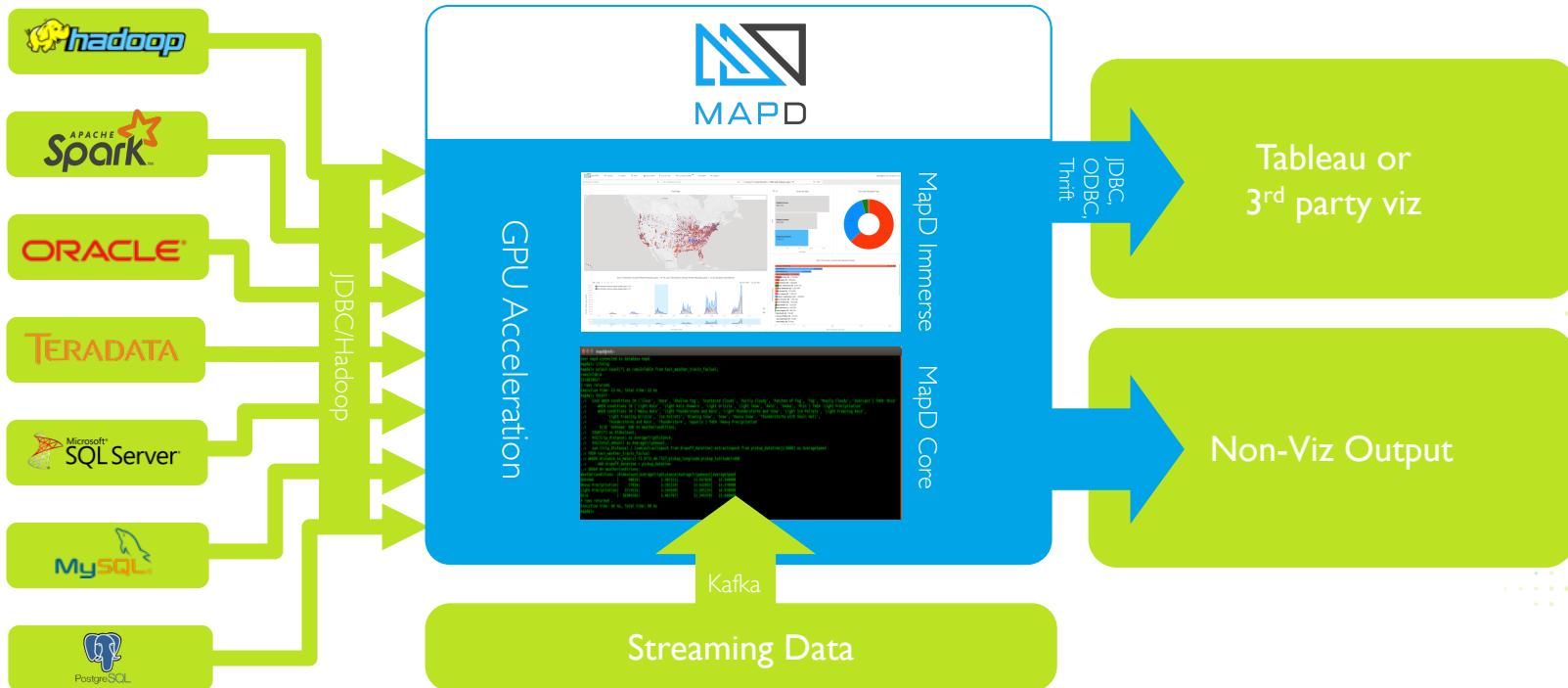
An in-memory, relational, column store database powered by GPUs

MapD Immerse

A visual analytics engine that leverages the speed + rendering capabilities of MapD Core



Where MapD sits



MapD Core

The world's fastest in-memory GPU database
powers the world's most immersive data
exploration experience

```
> and dropoff_datetime > pickup_datetime
> GROUP BY WeatherConditions,
WeatherConditions|RidesCount|AverageT
Unknown|1111.895455|7.855455|20.85000
Heavy Precipitation|170|1.875647|7.78
Light Precipitation|3993|2.223962|9.63
Nice|32218|2.213569|9.656598|12.560000
mapdql> \timing
mapdql> SELECT
> CASE WHEN conditions IN ('Clear'
THEN 'Nice'
> WHEN conditions IN ('Light
> WHEN conditions IN ('Heavy
light Freezing Drizzle', 'Ice Pellets',
galls') THEN 'Heavy Precipitation'
> ELSE 'Unknown'
> END AS WeatherConditions,
> COUNT(*) as RidesCount, AVG(trip_
dropoff_datetime)-extract(epoch from p
> FROM taxi_weather_tracts_factual
> WHERE distance_in_meters(-73.9772, 40.
> AND dropoff_datetime > pickup_
> ORDER BY WeatherConditions
```

Performance starts with memory management

SPEED INCREASES ↑

Hot Data

Speedup = 1500x to 5000x
Over Cold Data

Warm Data

Speedup = 35x to 120x
Over Cold Data

Cold Data

Data Lake/Data Warehouse/SOR

COMPUTE
LAYER

GPU RAM (L1)

24GB to 384GB
3000-5000 GB/sec

CPU RAM (L2)

32GB to 3TB
70-120 GB/sec

STORAGE
LAYER

SSD or NVRAM STORAGE (L3)

250GB to 20TB
1-2 GB/sec



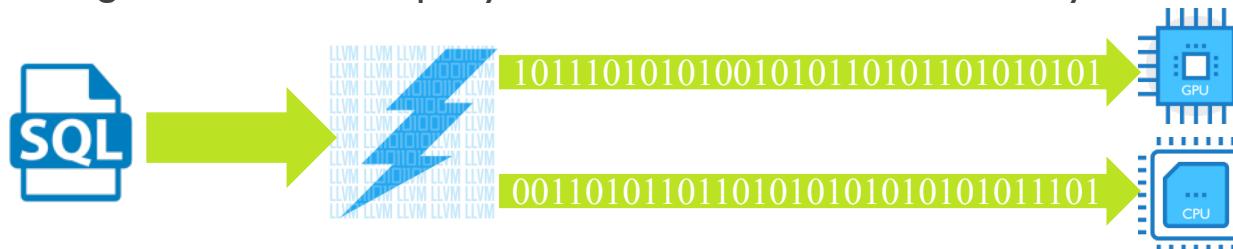
Query Compilation with LLVM

Traditional DBs can be highly inefficient

- each operator in SQL treated as a separate function
- incurs tremendous overhead and prevents vectorization

MapD compiles queries w/LLVM to create one custom function

- Queries run at speeds approaching hand-written functions
- LLVM enables generic targeting of different architectures (GPUs, X86, ARM, etc).
- Code can be generated to run query on CPU and GPU simultaneously



These innovations drive exceptional speed + scale

The table is sorted by the fastest time query 1 finished in (measured in seconds).

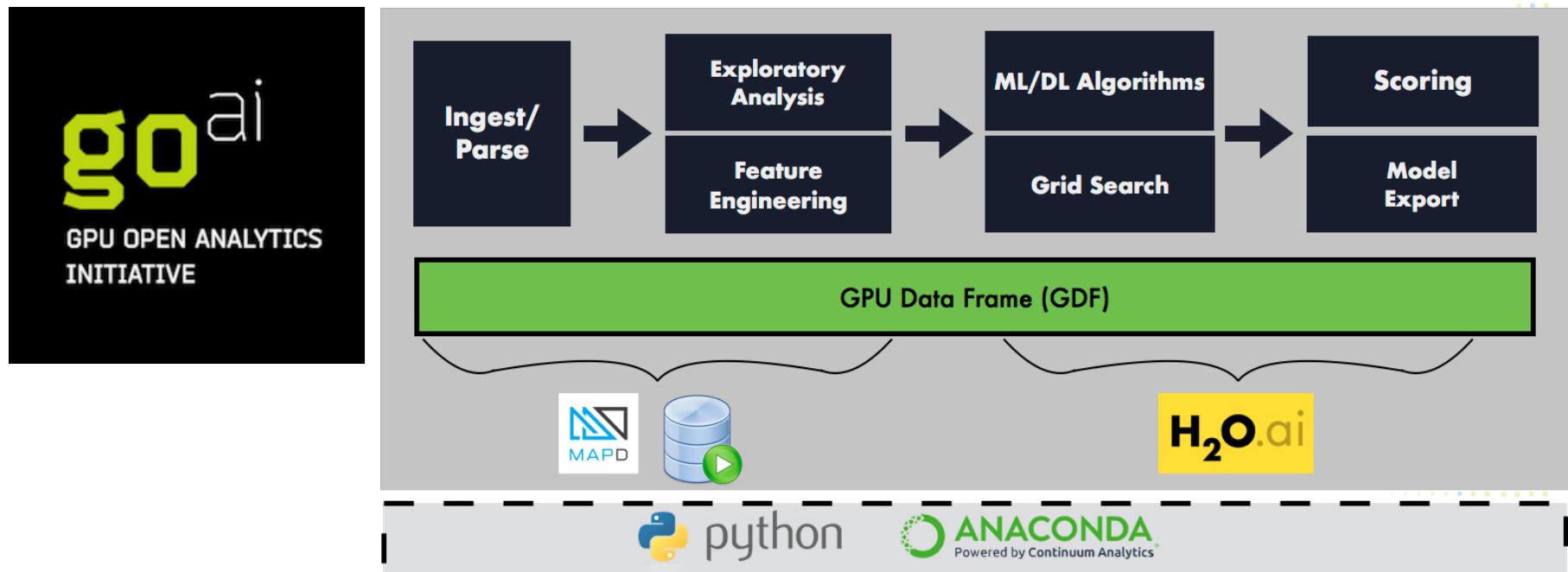
Query 1 Query 2 Query 3 Query 4 Setup

0.021	0.053	0.165	0.51	<u>MapD & 8 Nvidia Pascal Titan Xs</u>
0.027	0.083	0.163	0.891	<u>MapD & 8 Nvidia Tesla K80s</u>
0.028	0.2	0.237	0.578	<u>MapD & 4-node g2.8xlarge cluster</u>
0.036	0.131	0.439	0.964	<u>MapD & 4 Nvidia Titan Xs</u>
0.051	0.146	0.047	0.794	<u>kdb+/q & 4 Intel Xeon Phi 7210 CPUs</u>
1.034	3.058	5.354	12.748	<u>ClickHouse, Intel Core i5 4670K</u>
1.56	1.25	2.25	2.97	<u>Redshift, 6-node ds2.8xlarge cluster</u>
2	2	1	3	<u>BigQuery</u>
4	4	10	21	<u>Presto, 50-node n1-standard-4 cluster</u>
6.41	6.19	6.09	6.63	<u>Amazon Athena</u>
8.1	18.18	n/a	n/a	<u>Elasticsearch (heavily tuned)</u>
10.19	8.134	19.624	85.942	<u>Spark 2.1, 11 x m3.xlarge cluster w/ HDFS</u>
11	10	21	31	<u>Presto, 10-node n1-standard-4 cluster</u>
14.389	32.148	33.448	67.312	<u>Vertica, Intel Core i5 4670K</u>
34.48	63.3	n/a	b/a	<u>Elasticsearch (lightly tuned)</u>
35	39	64	81	<u>Presto, 5-node m3.xlarge cluster w/ HDFS</u>
43	45	27	44	<u>Presto, 50-node m3.xlarge cluster w/ S3</u>
152	175	235	368	<u>PostgreSQL 9.5 & cstore_fdw</u>
264	313	620	961	<u>Spark 1.6, 5-node m3.xlarge cluster w/ S3</u>

Noted DB blogger, Mark Litwintschik has benchmarked MapD vs. major CPU systems and found it to be between [74x](#) to [3,500x](#) faster than CPU DBs.

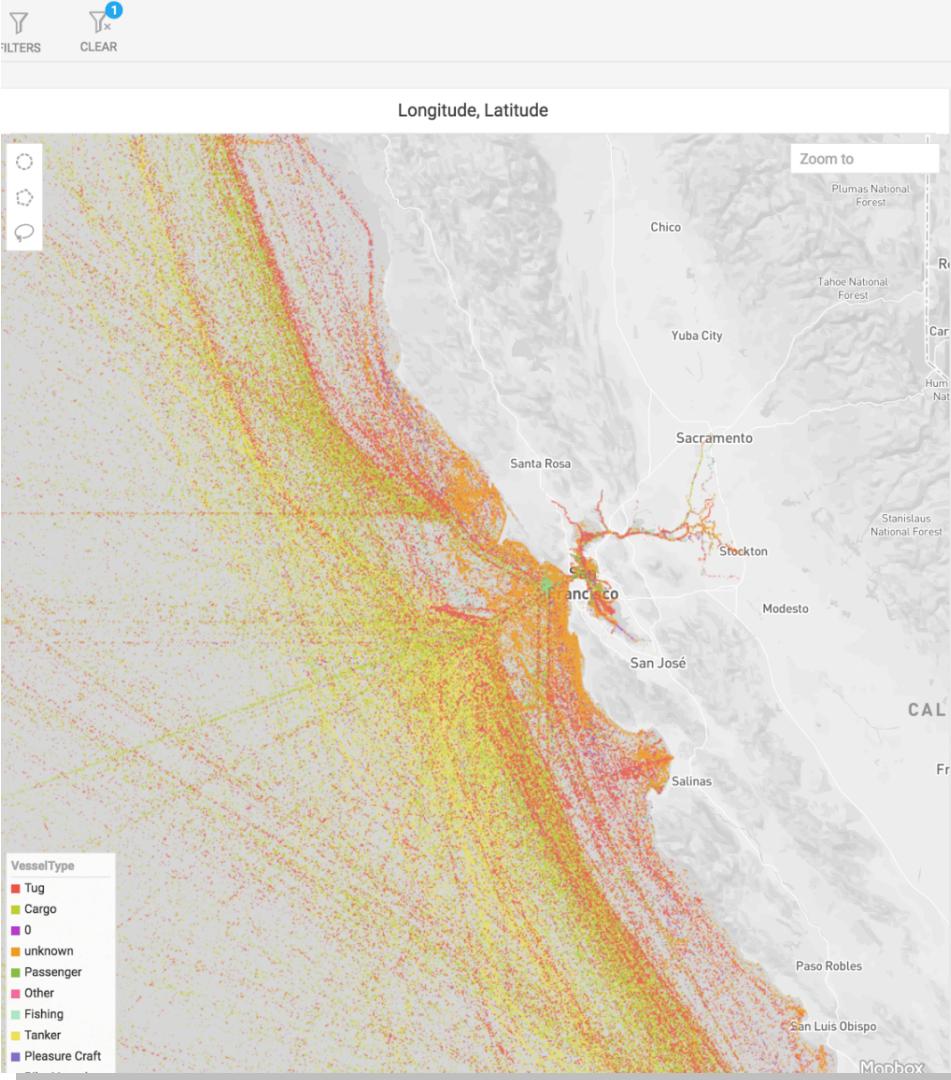


The GPU Open Analytics Initiative (GOAI) and the GPU Data Frame (GDF)

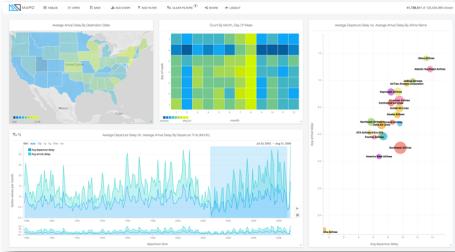


MapD Immerse

Lightning fast visual analytics for the
MapD Core database



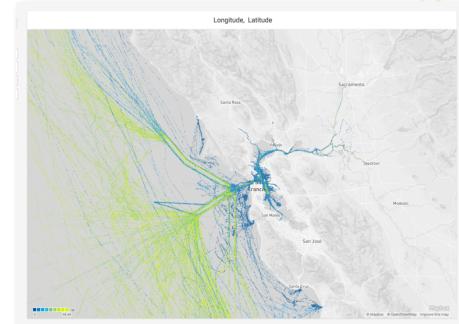
MapD Immerse: our hybrid approach



MapBox



React

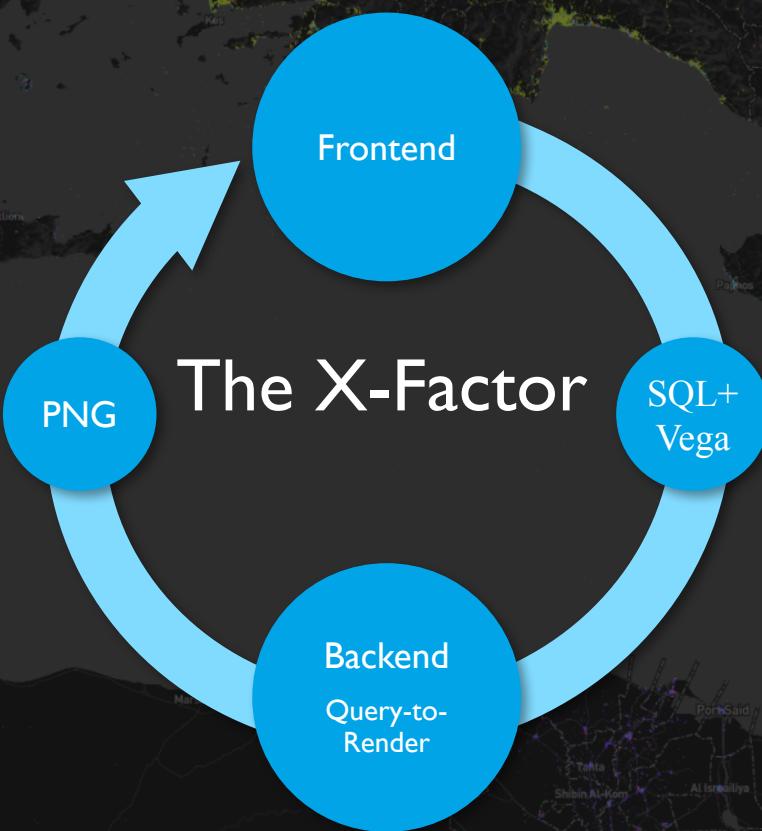


Basic charts are frontend rendered using D3 and other related toolkits

Scatterplots, pointmaps + polygons are backend rendered using the Iris Rendering Engine on GPUs

Geo-Viz is composited over a frontend rendered basemap

Server side rendering



Data goes from compute (CUDA) to graphics (OpenGL) pipeline without copy and comes back as compressed PNG (~100 KB) rather than raw data (> 1 GB)

Vega Spec (a visualization grammar)

A declarative JSON format for creating visualization designs

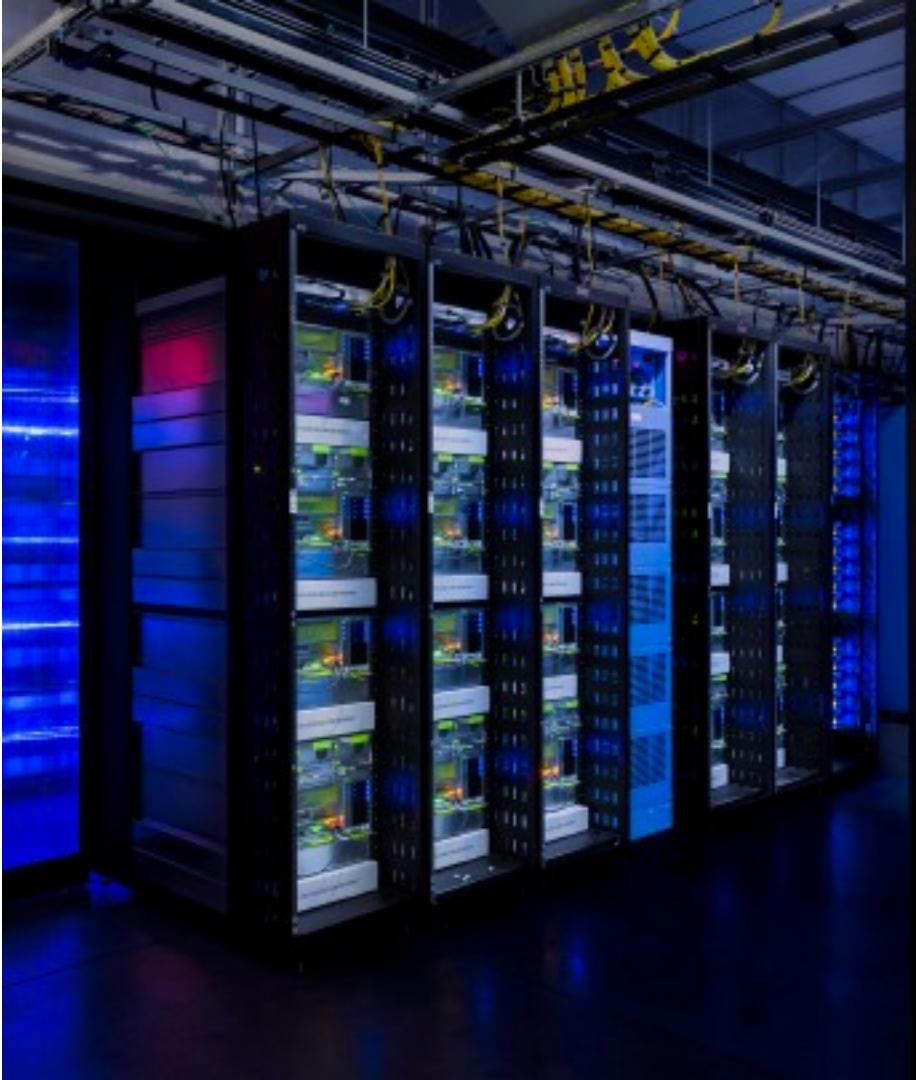
- Used to describe backend visualizations
- Defines attributes of render primitives which can be driven by data columns and mapped by scales

Shader Compilation Framework

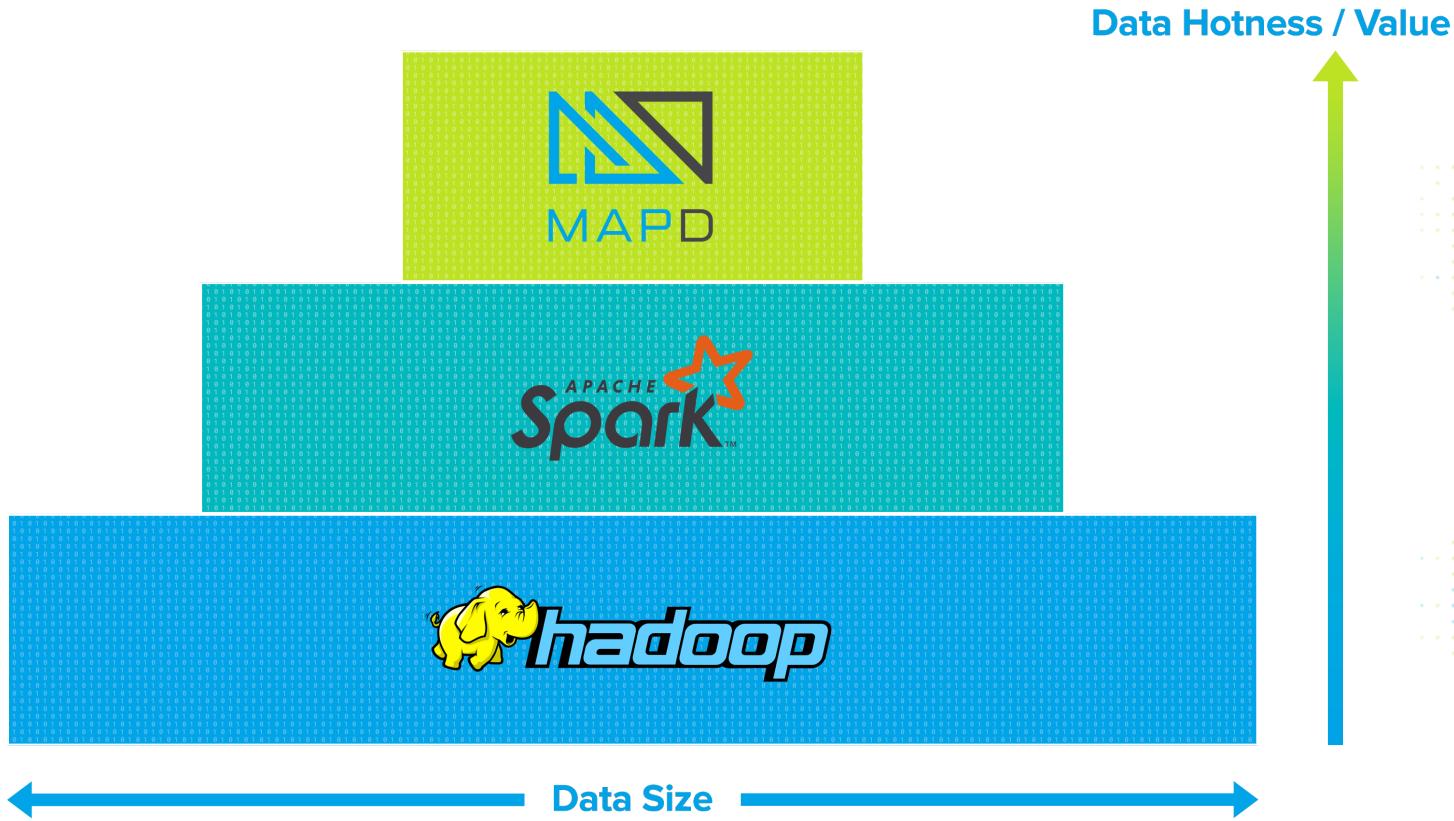
- Templatized: supports multiple types (ints, floats, colors, etc), and multiple continuities (discrete, continuous)

Supercharging Spark

Using MapD as a small-footprint, high-power
cache on top of Apache Spark



A Modern Analytics Stack





Demo

MapD, Now Open Source

Features	MapD Core Open Source	MapD Community	MapD Enterprise
Core Database	✓	✓	✓
Thrift API	✓	✓	✓
JDBC	✓	✓	✓
Charting API	✓	✓	✓
Immerse Visual Analytics Client	Non-Commercial Binary	Non-Commercial	✓
Rendering Engine	Non-Commercial Binary	Non-Commercial	✓
High Availability	✗	✗	✓
Distributed	✗	✗	✓
LDAP	✗	✗	✓
ODBC	✗	✗	✓
Warranty	✗	✗	✓
Support	Community Support	Community Support	Phone & Email



Delivering significant customer value across industries



Polling smartphones on demand to assess network health.

Took hours on Oracle previously.

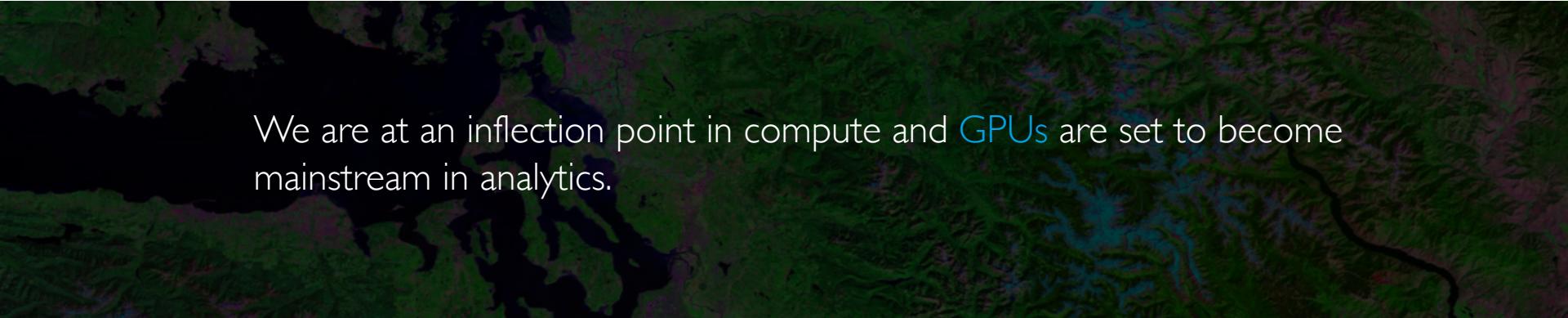
Running complex queries to analyze wireless QOS data.

Previous queries were non-interactive

EOG uses MapD to query and visualize well data

Other BI and GIS systems could not scale to their data sizes

Closing thoughts



We are at an inflection point in compute and GPUs are set to become mainstream in analytics.



Closing thoughts



GPUs allow users to **scale up** before needing to **scale out**, making GPU-accelerated analytics an excellent high-performance cache on a larger data lake or Spark deployment.



Closing thoughts



Integrated Analytics on GPUs comprising querying, viz and ML provide critical efficiencies and capabilities not found in siloed systems.





MapD Technologies, Inc.
One Front Street
Suite 2650
San Francisco, CA 94111
mapd.com

Cores and memory define the difference



24 cores

Traditional CPU Processing **vs.** GPU Processing



40k cores

(actually it would take a bit more than 24 of these slides to show 40k cores)

