

Streaming to humans:

Can open source hack it?



Deephaven



UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN



WALLEYE CAPITAL



Deephaven



**Getting data to
people matters**

What do people want?

1

See real-time data

2

Interact with tables & widgets

3

Consume and produce

4

Be first class citizens in the data system

table-2022-05-09

Formulas Data Review

Lookup & Reference Math & Trig More Functions

Name Manager

Function Library Defined Names

fx 2022-05-09

	B	C	D	E	
	Sym	Dollars Tot	Last Price	Open Price	Int
2022-05-09	SPY	\$18,985,349,859.38	\$401.0200	\$406.6400	
2022-05-09	QQQ	\$11,959,475,327.88	\$300.0000	\$305.5800	
2022-05-09	AAPL	\$7,836,494,723.30	\$152.5700	\$156.4800	0.024
2022-05-09	AMZN	\$4,838,213,327.42	\$2,219.0200	\$2,272.2500	0.024
2022-05-09	NVDA	\$4,258,629,386.13	\$174.4700	\$185.2200	0.0616
2022-05-09	MSFT	\$4,245,830,181.59	\$265.7700	\$271.4200	0.0213
2022-05-09	IWM	\$2,993,519,694.66	\$176.5850	\$180.7100	0.0234
2022-05-09	FB	\$2,689,136,034.37	\$197.8400	\$202.9200	0.0257
2022-05-09	XOM	\$1,547,348,640.63	\$86.2100	\$92.0200	0.0674
2022-05-09	GOOGL	\$1,483,381,498.00	\$2,274.2950	\$2,287.1600	0.008
2022-05-09	GGOG	\$1,338,986,766.94	\$2,281.1300	\$2,250.0000	-0.008
2022-05-09	SHOP	\$1,081,029,323.25	\$342.0100	\$375.6000	
2022-05-09	NFLX	\$984,831,107.13	\$175.6700	\$180.3700	
2022-05-09	OXY	\$939,540,065.24	\$58.6200	\$64.0000	
2022-05-09	QQQ	\$909,098,076.46	\$84.2200	\$95.1500	
2022-05-09	BA	\$883,915,164.90	\$85.4250	\$88.0000	
2022-05-09		\$802,325,869.61	\$48.1609		
2022-05-09		\$769,653,948.46	\$325.9300		
2022-05-09		\$739,640,571.06	\$107.5300		
2022-05-09		\$729,976,534.06	\$161.9600		
2022-05-09		\$681,966,824.34	\$79.6500		
2022-05-09		\$671,398,877.35	\$194.0000		
2022-05-09		\$33,051,878.80	\$12.0000		
2022-05-09		\$85,544.31			

The background is a dark blue gradient with various white line-art icons representing technical and scientific concepts. These include a 3D cube with arrows, a person standing, a person at a computer, a rocket, a bar chart, a pie chart, a cylinder, a sphere, and a circuit board.

Those needs catalyze today's discussion

- Distill technical implications
- Review currently popular options
- Itemize unsolved problems
- Describe open-source work toward a solution

Technical needs for a data transport



1. **Tables** that don't suck
2. A ubiquitous **backplane**
3. Efficient data consumption
4. **Browser** compatibility

Tables that don't suck

- They need to update in real time
- Schema changes must be transparent to the client
- The same must be true for widgets

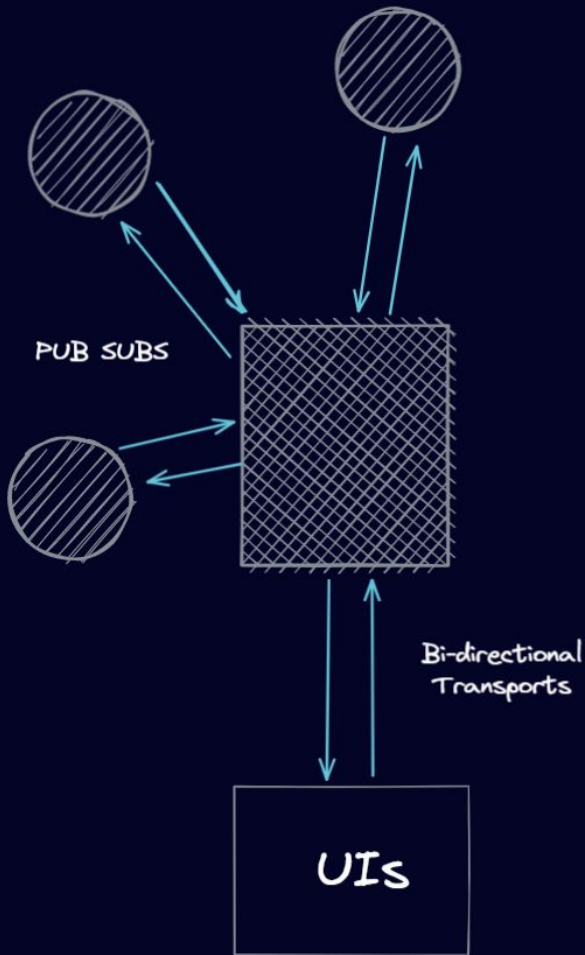
Tables that don't suck

- They need to update in real time
- Schema changes must be transparent to the client
- The same must be true for widgets

Static

Tables

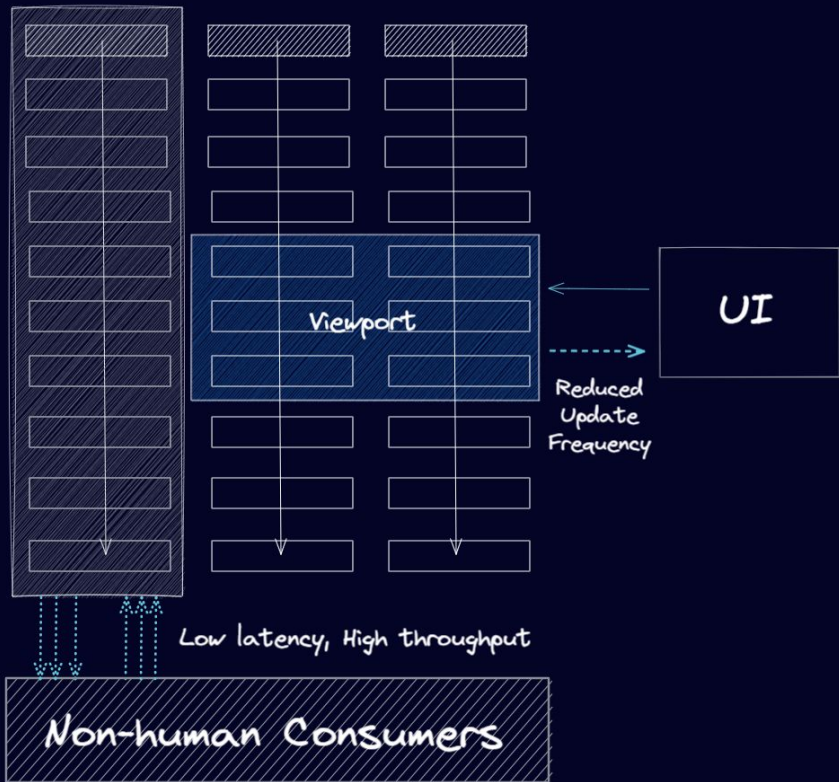




A ubiquitous backplane

- Reduce plumbing costs for new features
- Provide low latency, not just high throughput
- Support bi-directionality

Efficiently packed columnar data



Efficient data consumption

- Client must be able to control throughput and latency
- Design should aspire to be zero copy
- Data must be well packaged



Browser Compatibility

- Common denominator
- Lowest-entry point for eyes and fingers
- Platform independent solution
- Low resource technology
- Mobile-friendly bridge



Contenders for transport

Open, popular, modern



Contenders for transport

Open, popular, modern



Requirements	ZeroMQ	Kafka	Arrow Flight
Low latency	Can be	Can be	Yes
Variable schema	Yes	Yes	Yes
Capable of supporting tables	Yes	Yes	Yes
Efficiently packaged for tables	No	No	Yes
Zero Copy	No	No	Yes
Support for tables that update	Sort of	Sort of	No
Bi-directional streaming	Yes	Yes	Non-streaming
Client control of throughput and latency	No	No	No
Works in a browser	No	No	No

Kafka and ZeroMQ are non-starters

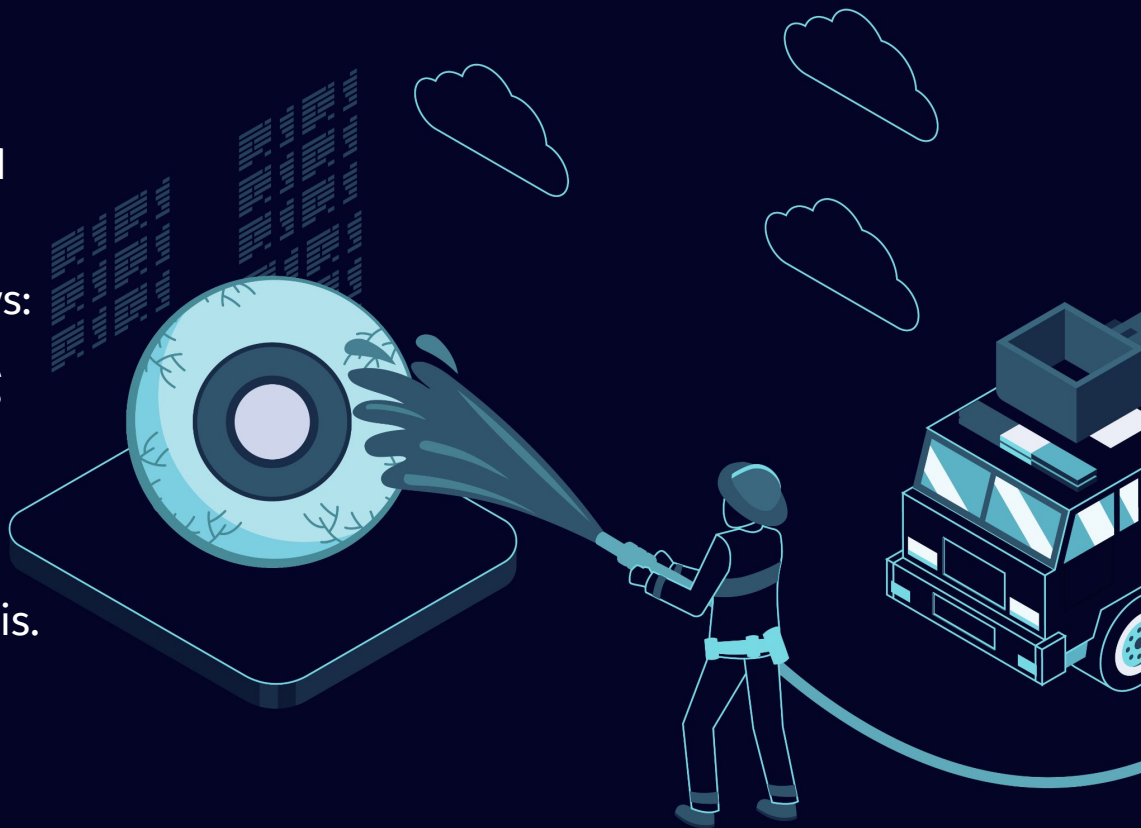
Good at real-time but...

For any pub or sub, the data is a blind appending stream.

This kills the requirements in two ways:

1. **No design for efficiently sending table data**
2. **Producer and consumer cannot collaborate directly**

There is no opportunity to improve this.





Arrow Flight

- Table super -powers
- Easily extendable
- Built on gRPC: so streaming seems “plausible”
- gRPC is based on http2, so browser support has potential

What we needed to do

1

Package “table changes” in Arrow Flight payloads

2

Employ Flight’s `DoExchange()` to implement custom streaming methods

3

Make a `JS client` that can connect to a Flight server with streaming support

Introducing

Barrage

Packaging “table changes”

- Data structure for describing table changes
- Table “**deltas**”: add, remove, modify, shift
- `streaming_tables`

Barrage: DoExchange() for streaming

snapshot(): “Tell me about a subset of this table”

A custom DoGet() call that can specify rows and columns.

subscribe(): “Give me the current contents and push me updates”

Snapshots plus inherited deltas.

publish(): “As a client, I’ll provide the server a snapshot and deltas”

The opposite of subscribe.

The Javascript client was a hard problem

Problems

Solutions

gRPC doesn't actually work in browsers

gRPC-Web is an almost-gRPC that IS accessible to browsers.

gRPC and gRPC-web are not actually compatible

Put a proxy between server and client (Envoy).

gRPC-web JS client does not support streaming binary data

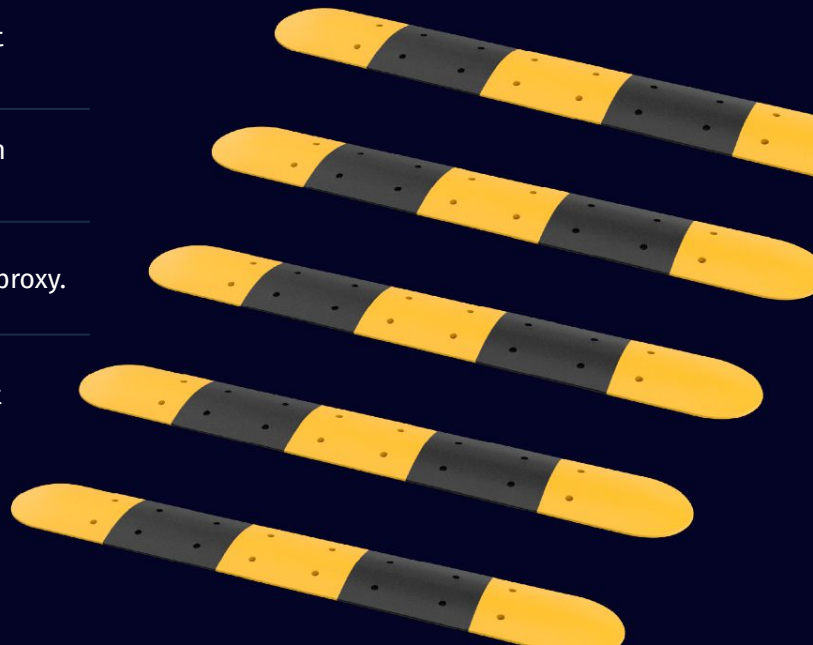
Use improbable-engineering's custom implementation.

Web browsers cannot stream data back to servers

Split methods and add a server-side proxy.

Browsers require SSL for http2: bad for localhost & secure environments

Improbable-engineering's web socket proxy for gRPC



Repos worth exploring



barrage

github.com/deephaven/barrage

deephaven-core

github.com/deephaven/deephaven-core

web-client-ui

github.com/deephaven/web-client-ui

```

> from deephaven import *
totals2 = tradesMkt.updateView("Dollars = Size * Price")\
    .by(caf.AggCombo(\
        caf.AggFirst("FirstPrice = Price"),\
        caf.AggLast("LastPrice = Price"),\
        caf.AggCount("Count"),\
        caf.AggSum("Shares = Size", "Dollars"),\
        caf.AggAvg("AvgPrice = Price"),\
        caf.AggWavg("Size", "WtdAvgPrice = Price")\
    ), "Sym")\
    .updateView("IntraChange = LastPrice - FirstPrice", "IntraPerc = LastPrice / FirstPrice -1")

```

Simple Query Basics.py

totalsWithJoin totals2 tradesYest tradesMkt oneStockTradesMkt qopl trades totals trad

```

4 #1 Get intraday trades
5 #tradesToday = db.i("FeedOS", "EquityTradeL1").where("Date = `2020-07-01`")
6 tradesToday = db.i("FeedOS", "EquityTradeL1").where("Date = `2020-07-01`")
8 #2 Do a couple basic things to that table
9 tradesToday = tradesToday.reverse().renameColumns("Sym" = "LocalCodeStr")
10
11 #3 Create a dependent table -- and demonstrate Aggregation
12 tradesLast = tradesToday.view("MarketTimestamp", "Sym", "Price", "Size").lastBy("Sym").sortBy("Sym")
13
14 #4 Get historical trades
15 # tradesYest = db.t("FeedOS", "EquityTradeL1").where("Date = lastBusinessDateNy()")
16 tradesYest = db.t("FeedOS", "EquityTradeL1").where("Date = `2020-07-01`")\
    .renameColumns("Sym" = "LocalCodeStr")
17
18
19 #5 You can of course merge them
20 trades = ttools.merge(tradesToday, tradesYest)
21
22 #6 Sort for trading time and calc totals
23 import deephaven.Calendars as Calendars
24 cal = Calendars.calendar("USNYSE")
25
26 tradesMkt = tradesToday.where("cal.isBusinessTime(MarketTimestamp)")
27 totals = tradesMkt.view("Date", "Sym", "Size", "Dollars = Size * Price").sumBy("Date", "Sym").sortBy("Date")
28
29 #7 Analyze performance
30 # One can use the following to figure out I/O versus compute time for various operations, etc.
31 qopl=db.i("DbInternal", "QueryOperationPerformanceLog")\
    .where("Date=currentDateNy()")
32

```

Try it at deephaven.io

Date	Timestamp	InternalCode	LocalCodeMarketId	Sym	ServerTimestamp	MarketTimestamp	Price
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T16:14:44.886	2020-10-07T16:00:00.000	340.92	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T16:14:16.949	2020-10-07T16:00:00.000	340.92	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.998	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	
2020-10-07	16:15:11.891	1,059,068,156	505 SPY	2020-10-07T15:59:59.997	2020-10-07T15:59:59.997	340.73	

