Elastic MDS™

Bringing Modern Computing Paradigms to Market Data

Andrew MacGaffey, President & CTO, MetaFluent LLC

October 20, 2020

Presentation for Global STAC Live

Me

Andrew MacGaffey

- 30+ years in real-time market data distribution
  - 19 years at Reuters – development and product management
  - Founded MetaFluent in 2006

MetaFluent

MetaFluent makes enterprise-class data distribution software

In wide-scale production deployment since 2007

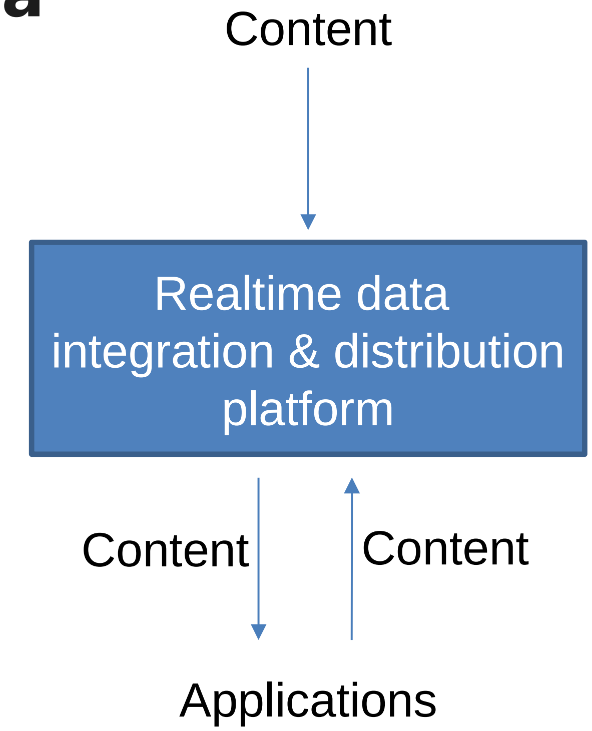
Multiple patents granted or pending

Target

Realtime apps, down to sub-millisecond latency

Mission

Improve business agility and reduce costs for app dev and operations



The problem with traditional architectures

They push complexity onto the developer

- Connection management, threading, dictionaries, etc.

They push complexity onto operations staff

- Complicated, fragile config
- Lack of visibility
- High cost to manage deployment environments

They aren’t keeping up with enterprise architecture trends:

- Rapid application development
- Public and private cloud migration
- Elastic data platforms

The roots of the problem run deep

Mainstream market data distribution thinking hasn’t changed much since the 1990s

So MetaFluent has questioned a lot of assumptions about what can and can’t be done

> “Your assumptions are your windows on the world. Scrub them off every once in a while, or the light won't come in.”
> — Isaac Asimov

Outdated assumption #1:

`Infrastructure scale is static`

Orthodoxy:

Configure to support specific feeds and specific number of app connections

Must build capacity for peak rates and peak usage

Adding feeds or users requires a change management process

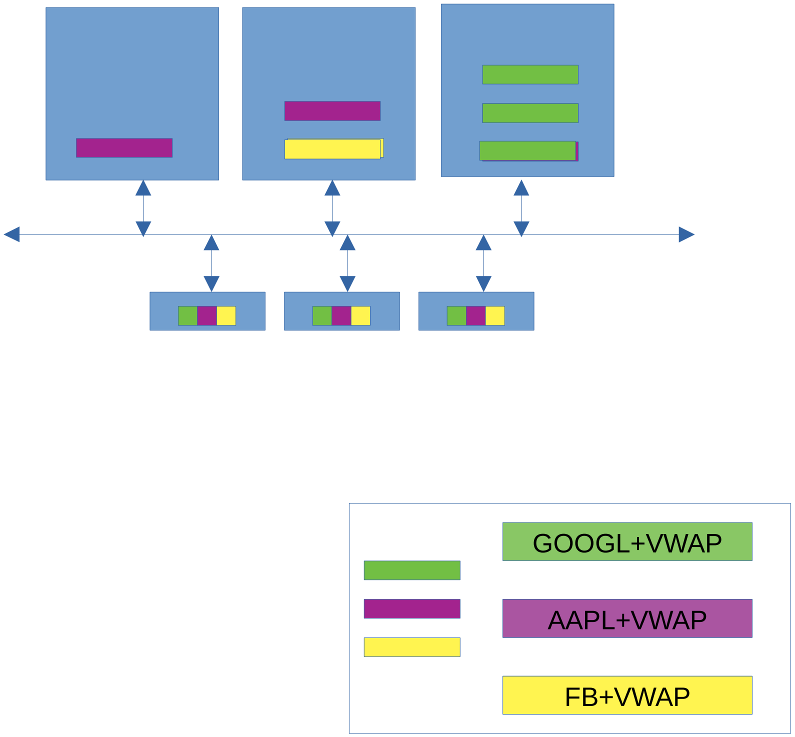
Consequences:

Slow business responsiveness

Inefficient use of hardware

Elastic MDS is...well...*elastic*

Animation sequence for auto-scaling



Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

Comprehensive line-of-sight for operations

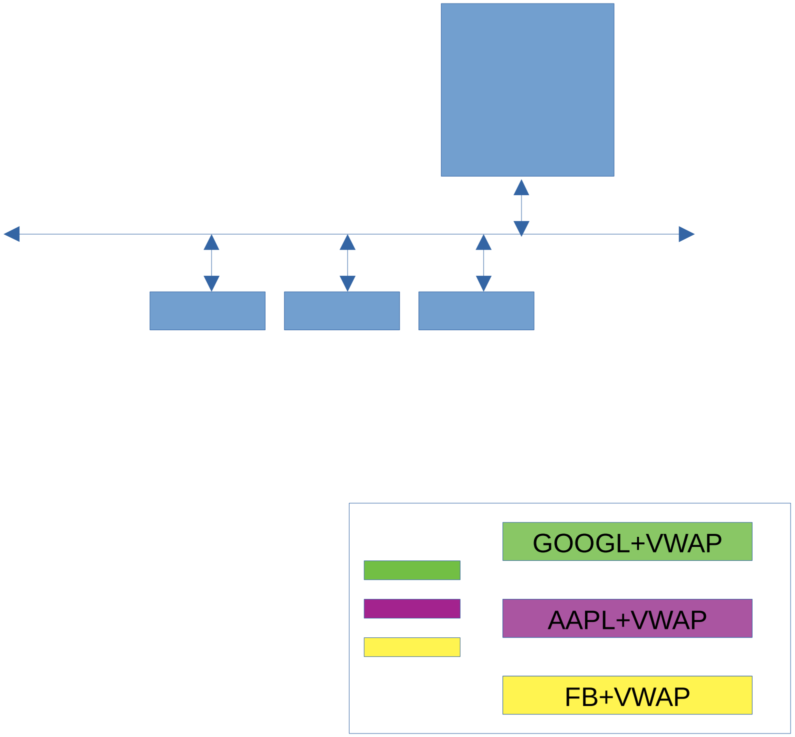
Tracing, metrics etc

REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is…well…*elastic*

Animation sequence for auto-scaling



Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

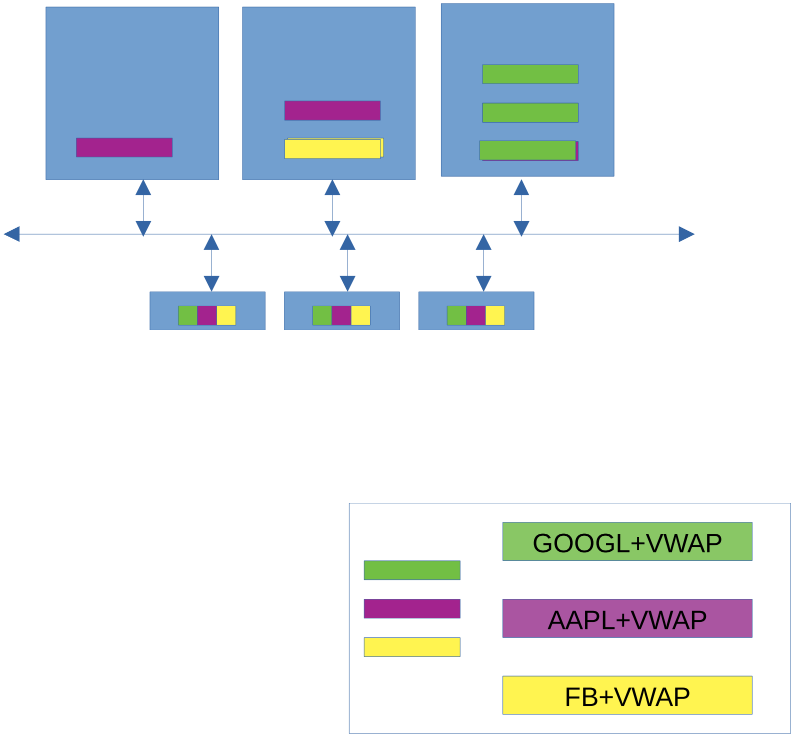
Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is...well...*elastic*

Animation sequence for auto-scaling



Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

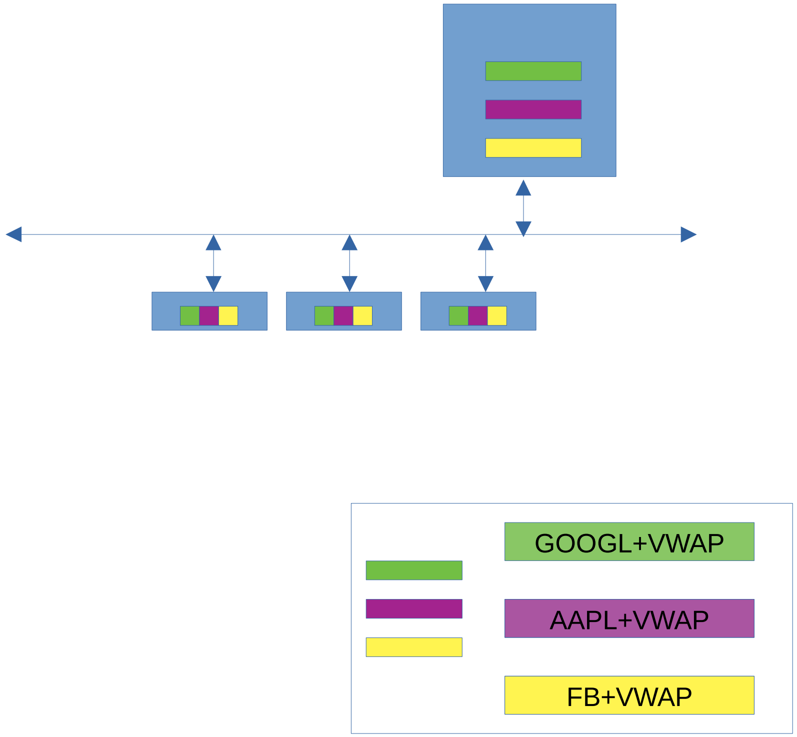
Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is...well...*elastic*

Animation sequence for auto-scaling



Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

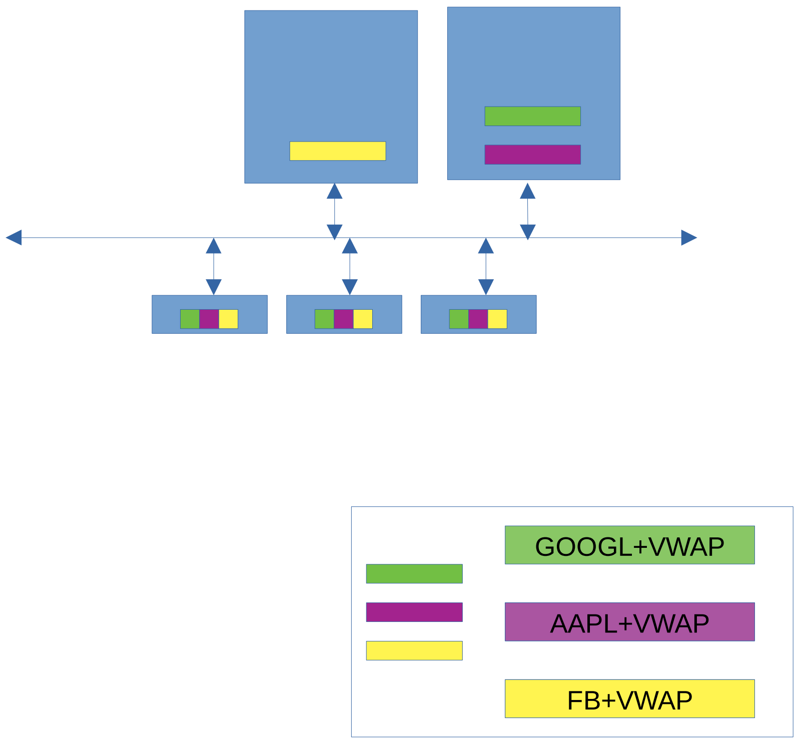
Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is…well…*elastic*

Animation sequence for auto-scaling



Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

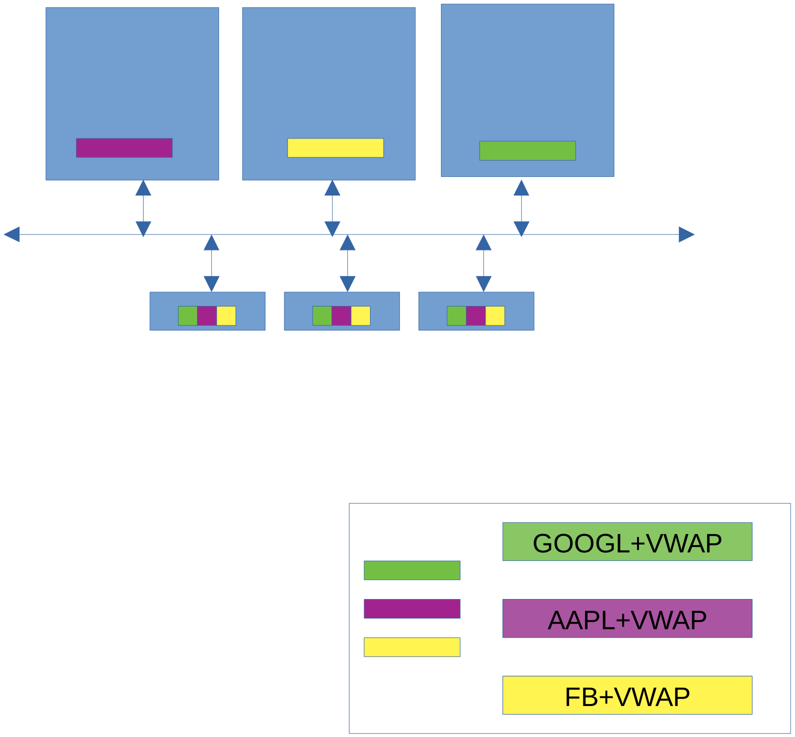
Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is...well...*elastic*

Animation sequence for auto-scaling



Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

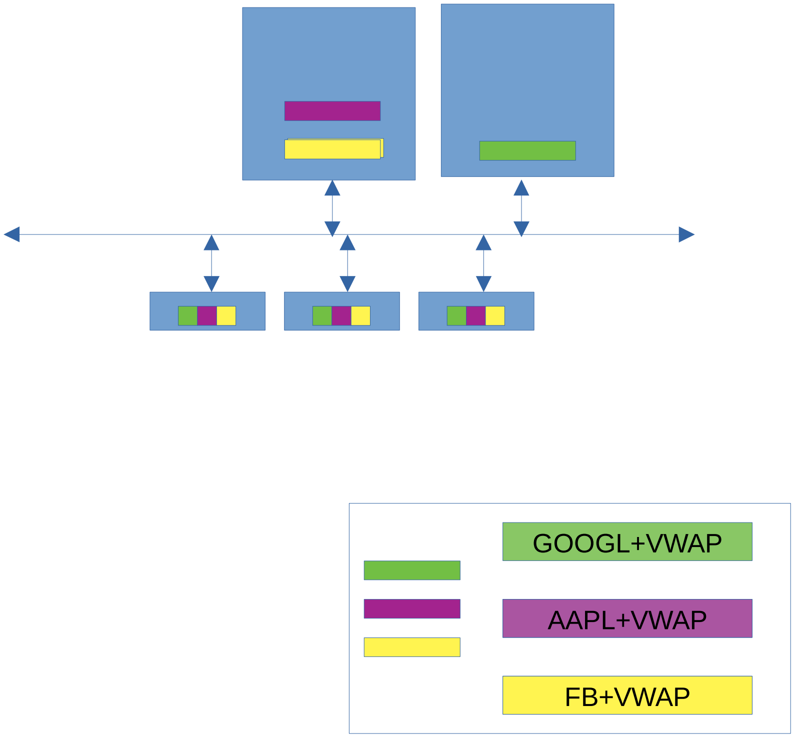
Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is…well…*elastic*

Animation sequence for auto-scaling



GOOGL+VWAP

AAPL+VWAP

FB+VWAP

Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

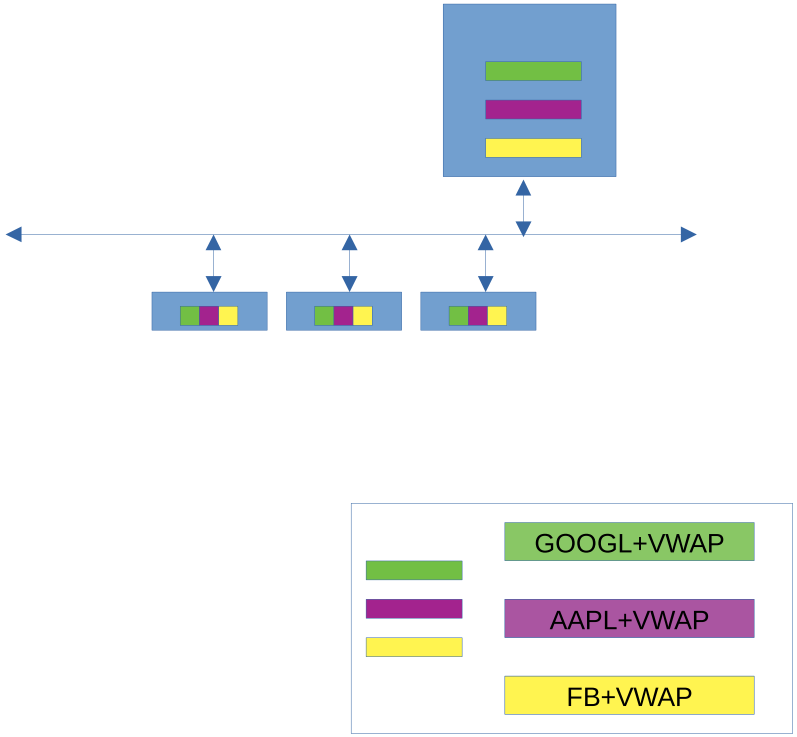
Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Elastic MDS is…well…*elastic*

Animation sequence for auto-scaling



GOOGL+VWAP

AAPL+VWAP

FB+VWAP

Resource auto-discovery

Cloud-ready with Kubernetes integration

Content-based load balancing

Fine-grained resilience

Policy-based autoscaling*

Comprehensive line-of-sight for operations

- Tracing, metrics etc
- REST-based management APIs

* Static scaling is still possible, of course

Out-of-date assumption #2:
`All realtime apps require data to be pushed`

The reality:

Some do but most don't

Business logic only needs data to be current when read

Consequence of push: apps spend a lot of time handling events and processing unneeded data

Complexity, resource intensity, scaling difficulty

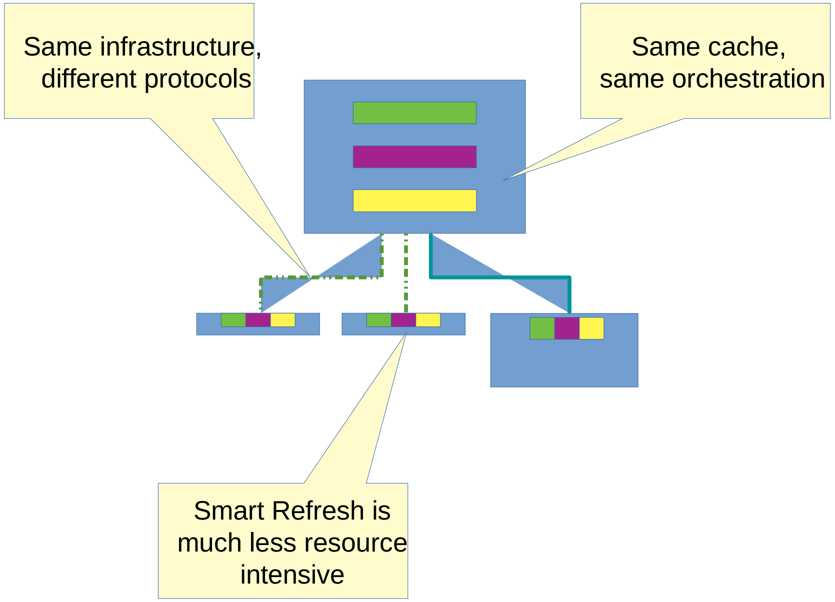
Slow-consumer issues

Difficult to virtualize apps

Data silos (what about reference data and time-series?)

MetaFluent Smart Refresh™

An alternative, non-push way for apps to get sub-millisecond data from the same platform that provides push


Same infrastructure, different protocols
Same cache, same orchestration
Smart Refresh is much less resource intensive

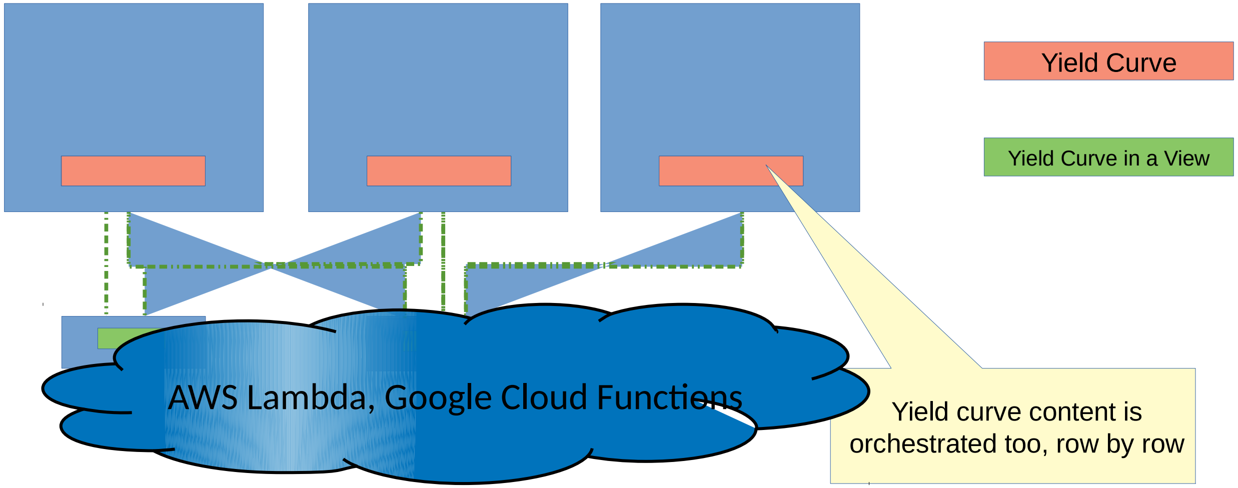
- App issues a query
- App reads data from results in local memory
- App refreshes results only where necessary
- Implementation uses highly optimized payloads, state awareness, and parallel I/O to achieve latency targets

Benefits

- Apps use less CPU
- No more streaming environment to manage south of the platform
- Lower bandwidth (think cloud exfiltration costs)

MetaFluent Smart Refresh™

- Smart Refresh even enables business logic to move to FaaS
- No more app servers ("serverless")


Yield Curve
Yield Curve in a View
AWS Lambda, Google Cloud Functions
Yield curve content is orchestrated too, row by row

Outdated assumption #3:

Development complexity is inevitable

Sub-assumptions:

Sophisticated capabilities necessarily expose complexity to the developer.

A new system requires a new API.

Vendors operating on these assumptions cost customers time, money, and flexibility

Elastic MDS shields developers from complexity

Standard enterprise APIs, such as:

- JDBC or REST for Smart Refresh
- JMS for push

Only other knowledge developers need:

- What content they want

It’s so easy, *you already know how to use it*.™

A Quick Example

```
String url = String.format("jdbc-mfrtc-session-json://%s:%d", getHost(), getPort());
fConnection = DriverManager.getConnection(url, properties);

…
String expression = "SELECT BID,ASK,TRDPRC_1 FROM QUOTES...";
      boolean hasResult = fStatement.execute(expression);
fStatement = fConnection.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
boolean hasResult = fStatement.execute(expression);

fResultSet.absolute(startIndex+1);
      while ( !fResultSet.isAfterLast() ) {
      for ( int c = 1; c <= fResultSet.getMetaData().getColumnCount(); c++ ) {
      fResultSet.getString(c)
      }
      fResultSet.next();
      }
```

How about a Reuters chain with VWAP injected? Same code exactly, different SQL

```
SELECT
DSPLY_NAME, BID, ASK, TRDPRC_1,
MFMath.VWA(TRDPRC_1, TRDVOL_1, 5) AS VWAP5,
ACVOL_1
FROM Chains.RDF WHERE _ChainSymbol = '0#.FTSE'
```

Summary

Please tick MetaFluent now on your response form for more info or a demo

More details

What was that

These should be talking points for the code example

- Simplicity
  - Initialization
  - Data handling
- Completely standard
- Refresh 100x a second, or every 2 minutes … whatever you need
- For VWAP example:
  - Could be any computation

Smart Refresh allow apps devs to stop worrying about

Complex initialization

Event queues

Events

Threads for market data

Decoding

Data dictionaries

Training?

Features of Elastic MDS push capability

Publish/subscribe

Tick-by-tick

Access to full range of content, including:

- User-defined server-side computations
- Collections (chains, portfolios, and other collections accessed as single topics; the platform returns a multi-stream response)

Orchestration over multiple connections improves resilience and flow-control

Use a JDBC query to create a view with of chain having selected fields along with computed fields

- only push what you need

Not used

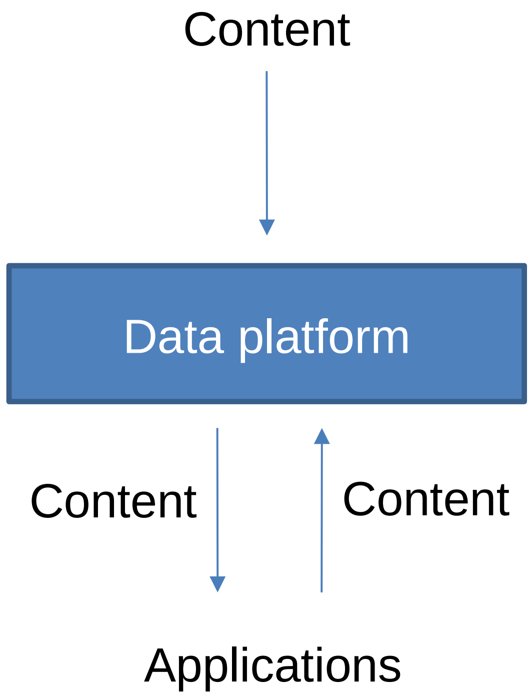
Table stakes for enterprise market data platforms

Plumbing:

- High throughput/low latency
- No data loss
- Scalability
- Resilience

Data management:

- Content integration
- User-defined calcs/derived data
- Content-based entitlements (cascaded)
- Data quality indication (cascaded)


Content
Data platform
Content
Content
Applications