AQUAQ Analytics

Chronicle Queue and kdb+

Experts in fast data solutions

for demanding environments

AQUAQ Analytics

Chronicle Queue and kdb+

- Why are we investigating this adapter?
  - AquaQ do more than just kdb+, and work on several installations where Chronicle Queue is an intrinsic component
  - More than 80% of the Top 100 banks globally, use Chronicle's Enterprise or Open-source products.
  - Many of these banks are also kdb+ users.
- Chronicle and AquaQ have entered a partnership.
  - Aim is to develop an adapter to offer their respective Clients.
  - Providing a ready-made integration between the two products, something that firms have previously had to build and maintain themselves.

Who are Chronicle Software

Chronicle SOFTWARE

Founded in 2013 by Java Champion Peter Lawrey

Specialist provider of low-latency, high-performance technology solutions for the financial services industry

Solutions available in Java and C++

Trusted Partner to Multiple Banks and Trading Organisations

Methodology that allows easy development and deployment

Chronicle By Numbers

17,000,000 Downloads Per Year

400,000 Open Source Users Globally

5,000 Chronicle Queue downloads per month

7,000 Chronicle Map downloads per month

80%+ of the worlds Top 100 Banks use Chronicle Software

Why Chronicle?

Chronicle SOFTWARE

Buy and Build The Best of Both Worlds

The only Low Latency Trading Provider to give you full source code

Ease of development and implementation

Our microservices architecture ensures rapid deployment and enhanced ROI

Experience

Peter and the team have a wealth of experience working across multiple successful large scale trading solutions

Chronicle Solutions

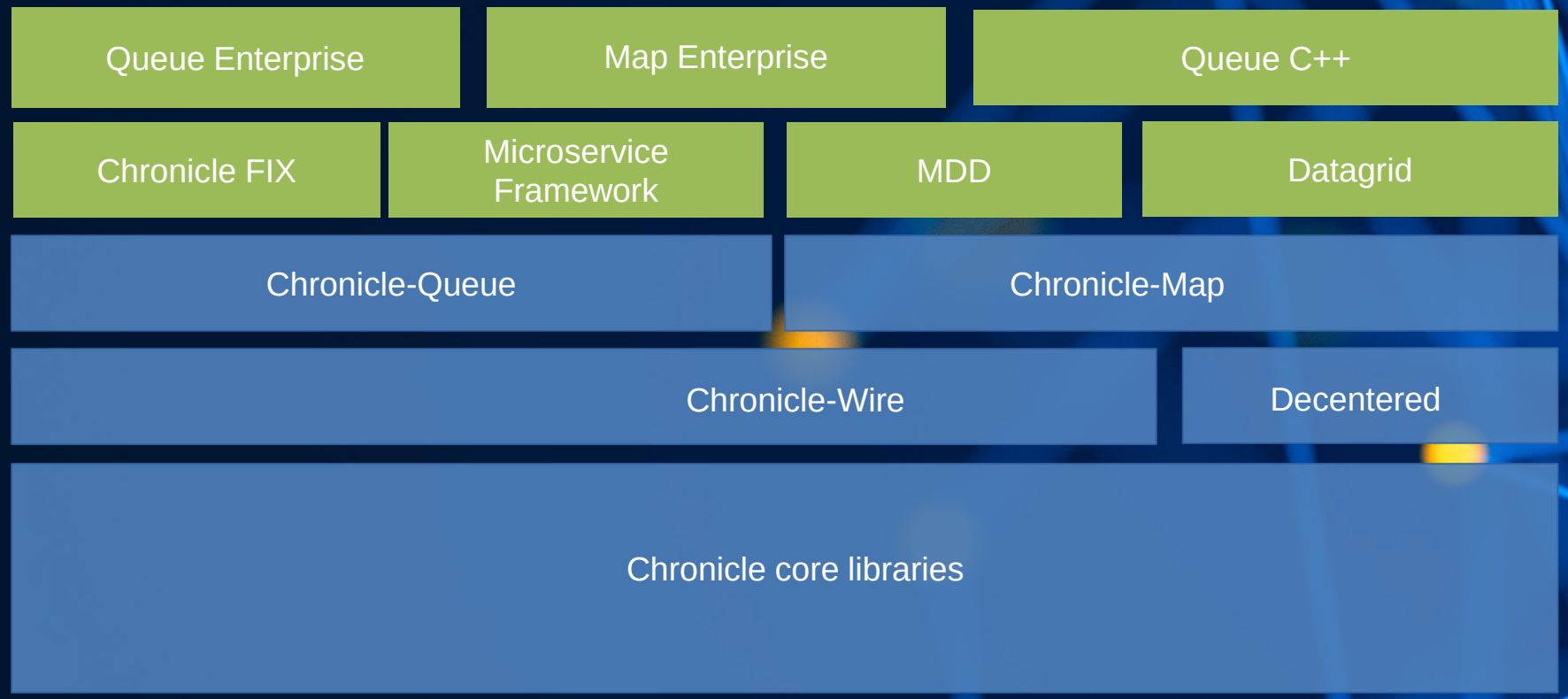
Chronicle solutions enable you to buy core infrastructure and immediately leverage a low latency framework

Chronicle solutions enables you to take a microservice based approach for fast incremental gains

Chronicle Solutions give you the flexibility to make swift trading adjustments and optimise market opportunities

Chronicles unique development methodology allows you to increase developer productivity by up to 30%

Chronicle Stack



Enterprise

OpenHFT

Chronicle SOFTWARE

Chronicle Queue persists all data to a memory mapped file

Main Features Include:-

- The ability to persist every single event and message
- Persistence through memory mapped files
- Low latency Interprocess communications (IPC)
- Network replication for High Availability
- Available in C++ and Java

Benchmarks

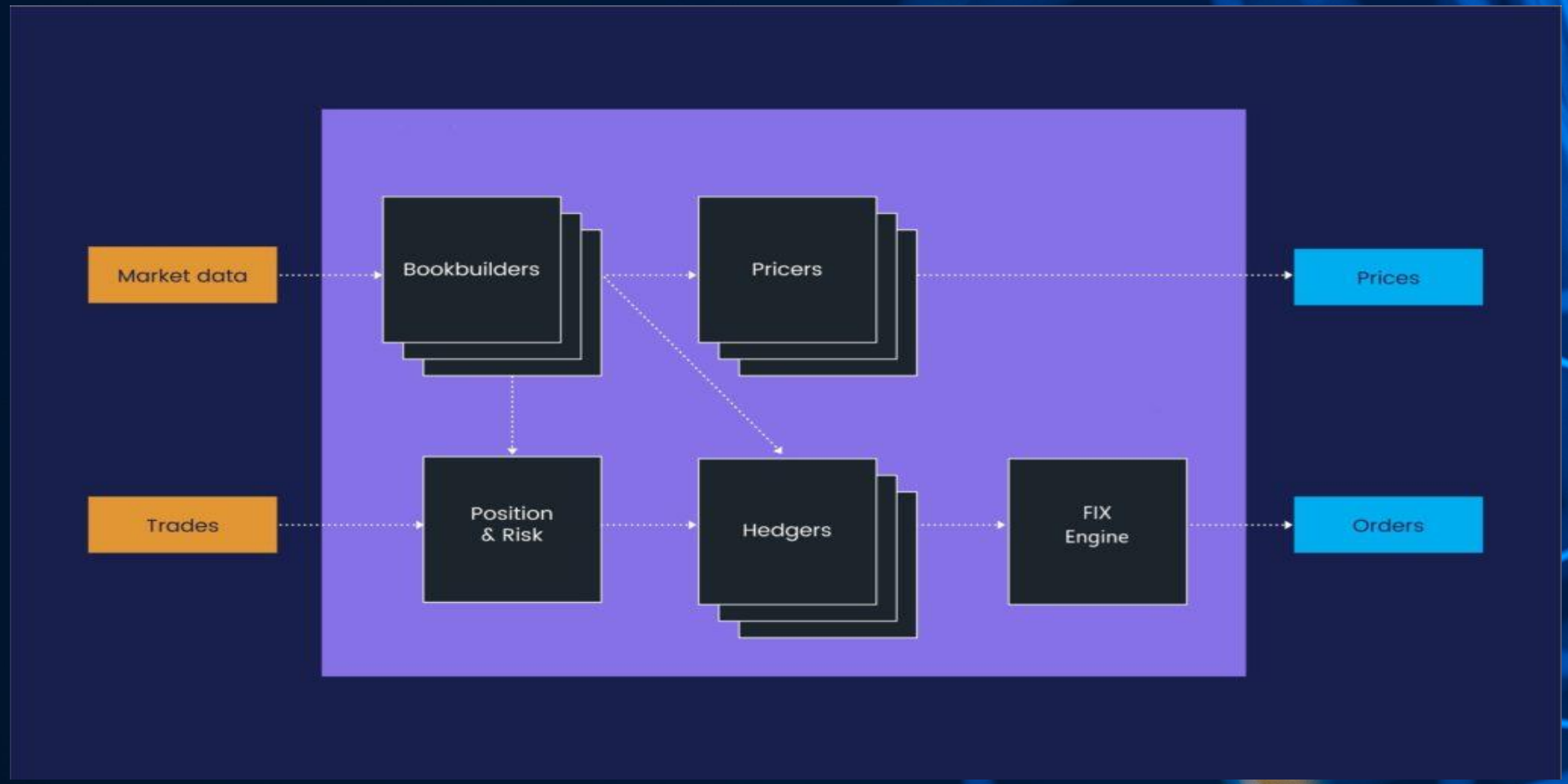
Chronicle SOFTWARE

We have recently benchmarked some throughput numbers for read/write on Chronicle Queue, these numbers illustrate how Queue handles bursts of data:-

Dsize=60 bytes
Writing 140,500,273 messages took 5.025 seconds, at a rate of **27,959,000 per second**
Reading 140,500,273 messages took 6.619 seconds, at a rate of **21,226,000 per second**
Dsize=200 bytes
Writing 44,039,331 messages took 5.017 seconds, at a rate of **8,778,000 per second**
Reading 44,039,331 messages took 2.846 seconds, at a rate of **15,472,000 per second**
Dsize=500 bytes
Writing 23,068,441 messages took 5.076 seconds, at a rate of **4,544,000 per second**
Reading 23,068,441 messages took 2.728 seconds, at a rate of **8,456,000 per second**

Chronicle SOFTWARE

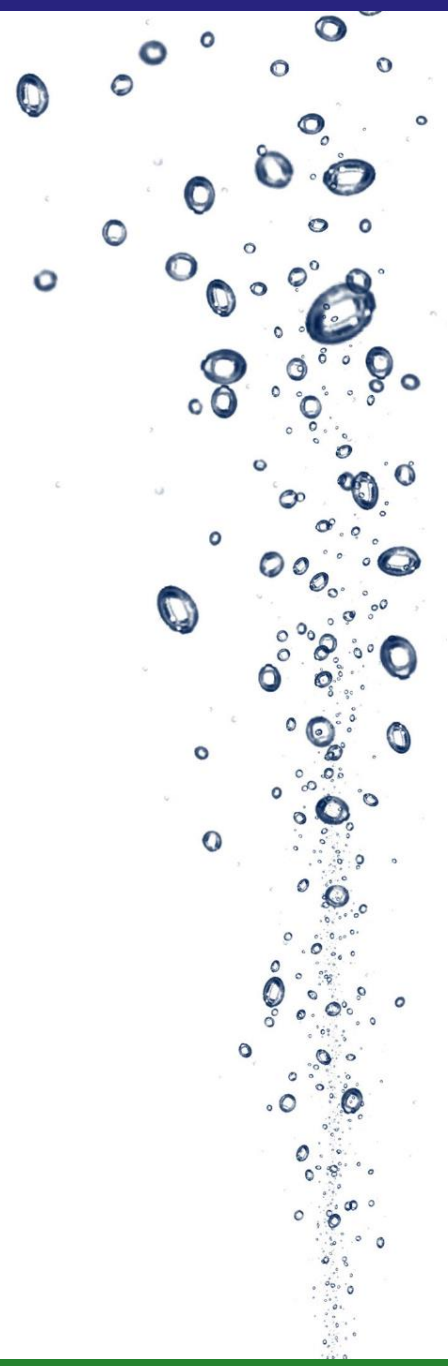
Chronicle EFX brings together all the components in a single solution



AQUAQ Analytics

Adapter Requirements

- Read data from a Chronicle Queue source
- Write data to a kdb+ destination
  - Configurable source and destination
  - Configurable adapter “type”
  - Configurable mapping from source to destination format
  - Configurable batching on kdb+ write side
  - Error handling, rollback on failure



AQUAQ Analytics

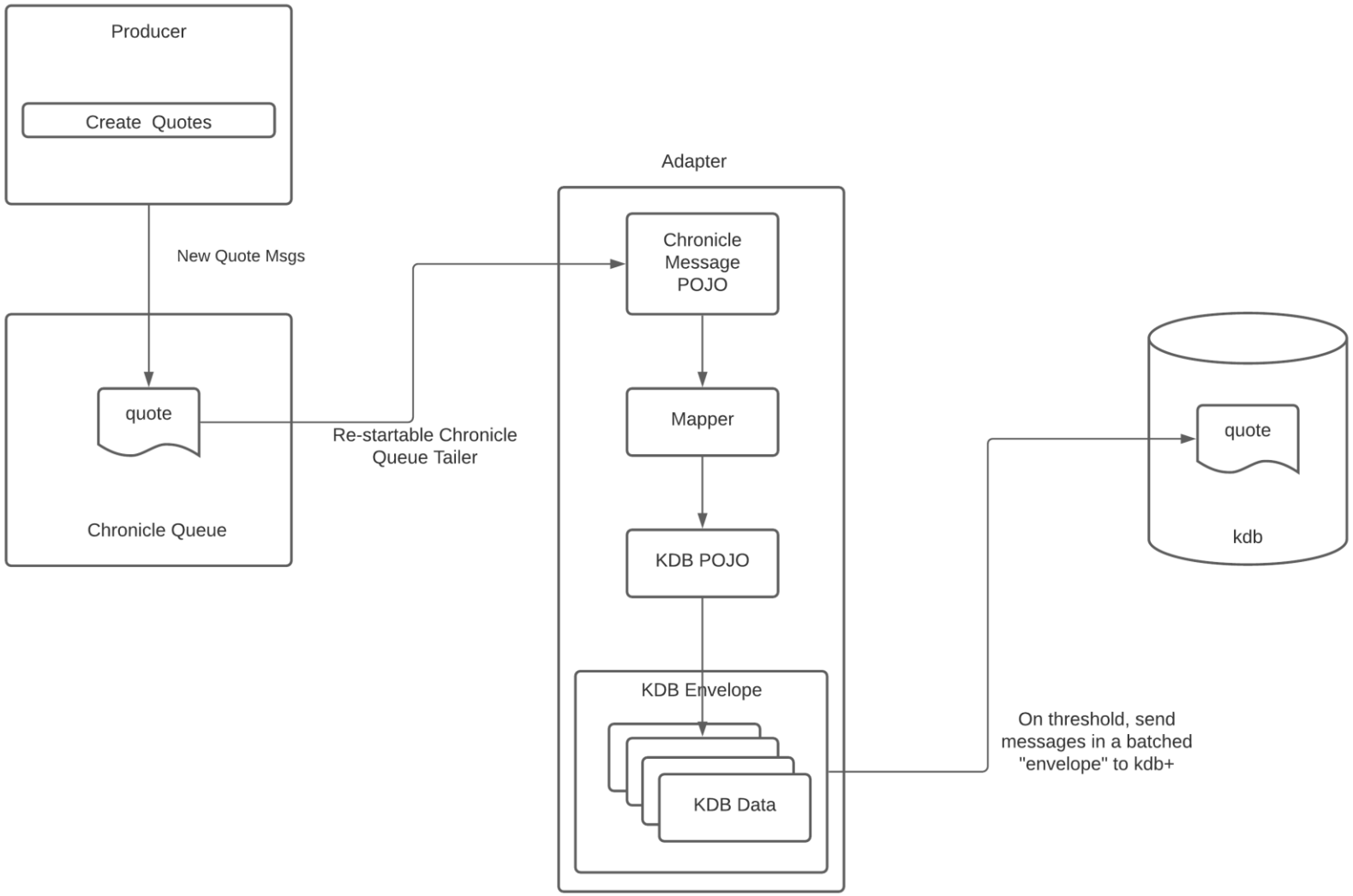
Adapter Development

- Build Adapter framework
  - Java, Maven based application
  - Use Abstract Factory Design Pattern
  - Use Chronicle's Java library classes to process the Queue
  - Use KX Java library classes to write to kdb+

- Implement scenario specifics following framework approach
  - Read "Quote" messages from Chronicle Queue
  - Write rows in "Quote" table in kdb+

AQUAQ Analytics

Adapter Process



AQUAQ Analytics

Chronicle Queue

- How do you write messages to a Chronicle Queue?
  - Data is written to a queue using an **Appender**.
  - Referred to as "**storing an excerpt**".
  - It can be any data type, including text, numbers, or serialised data.
  - [Generated 4-byte header + Excerpt] is called a **Document.**
  - Sequential writes, append to the end only.

AQUAQ Analytics

Chronicle Queue

- How do you read messages from a Chronicle Queue?
  - Data is read from a queue using a **Tailer.**
  - Every tailer reading from a queue sees every message.
  - Sequential and random reads, forwards and backwards.
  - “Re-start-able”, i.e. start reading from where you left off previously.

AQUAQ Analytics

Kdb+

- Typically write / publish data to kdb+ specifying:
  - Update function
  - Destination Table name
  - Data (Object[])
- *To maximize throughput and efficiency, it is recommended to publish multiple rows in one go.*

```
// "Single row insert"
String[] sym = new String[] {"IBM"};
double[] bid = new double[] {100.25};
double[] ask = new double[] {100.26};
int[] bSize = new int[]{1000};
int[] aSize = new int[]{1000};
Object[] data = new Object[] {sym, bid, ask, bSize, aSize};
```

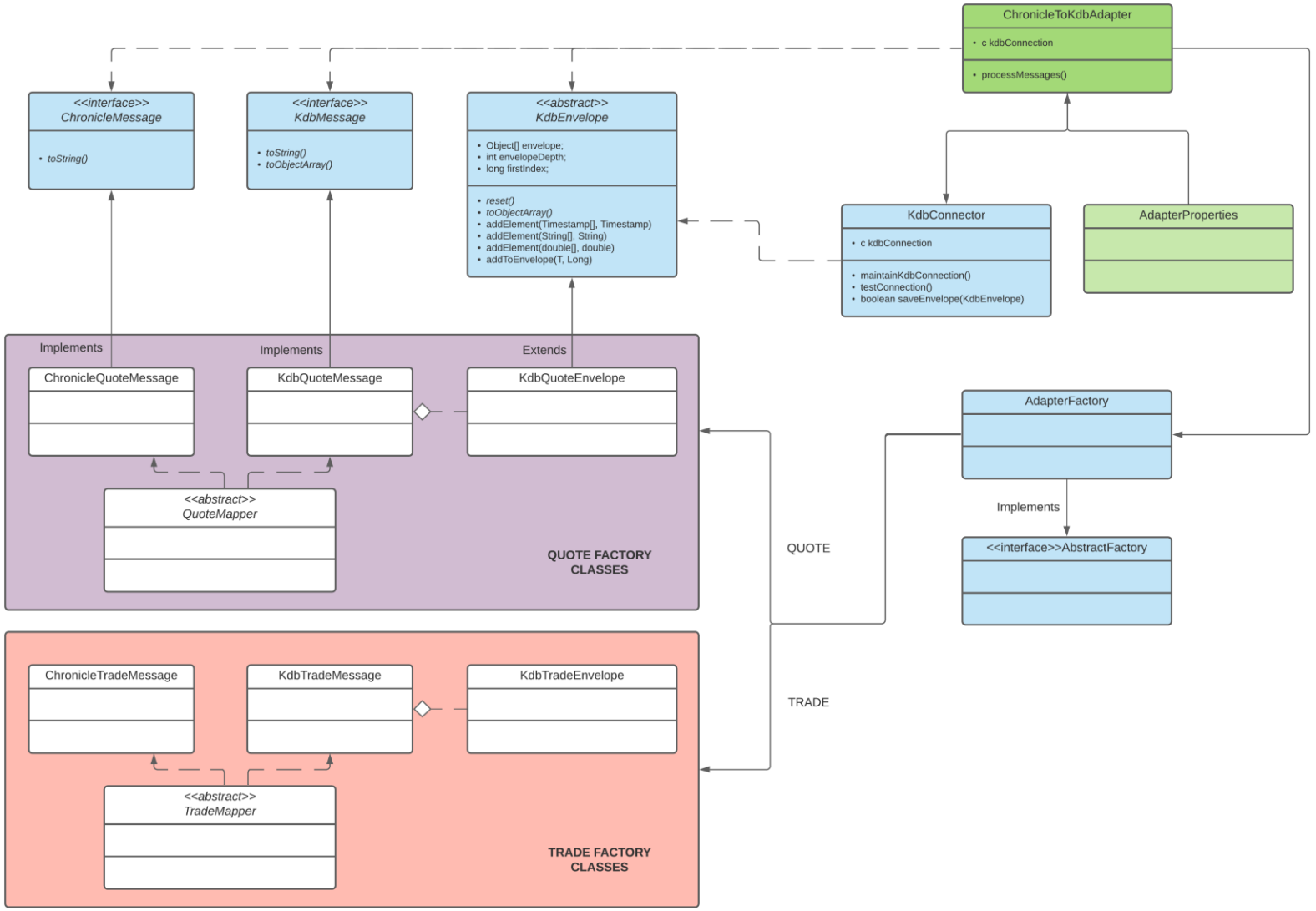
OK

```
// "Multi-row insert"
String[] sym = new String[] {"IBM", "VOD.L", "AAPL"};
double[] bid = new double[] {100.25, 99.2, 43.25};
double[] ask = new double[] {100.26, 98.4, 45.3};
int[] bSize = new int[]{1000, 1000, 1000};
int[] aSize = new int[]{1000, 1000, 1000};
Object[] data = new Object[] {sym, bid, ask, bSize, aSize};
```

BETTER

AQUAQ Analytics

Adapter Objects



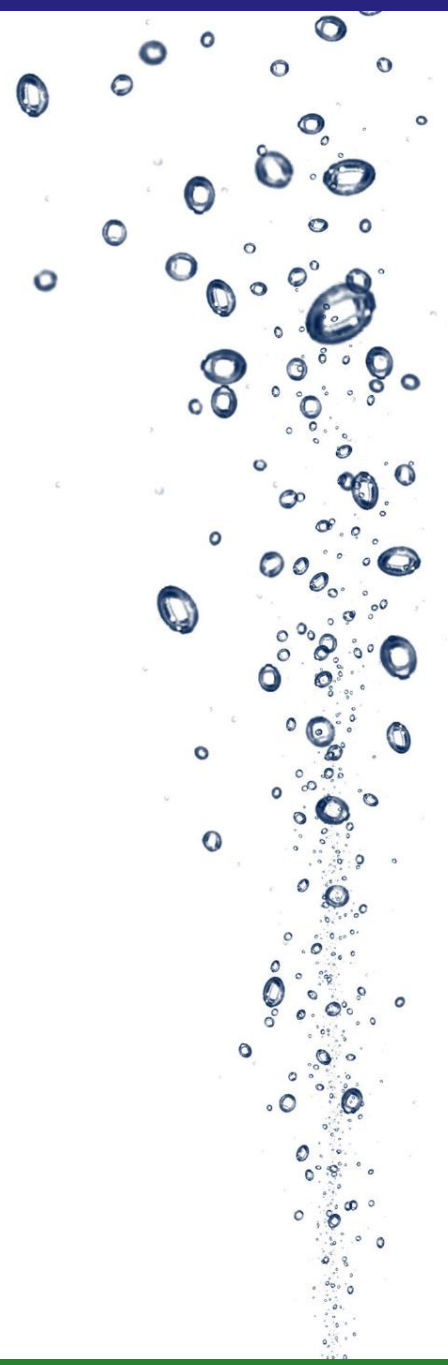
AQUAQ Analytics

Adapter Config

- # What mode the Adapter will run in - NORMAL, BENCH or KDB_BENCH
- **adapter.runMode**=NORMAL

- # Location and name of stop file. If this exists the Adapter will exit / not start
- **adapter.stopFile**=C:\\ChronicleQueue\\Producer\\quote\\STOP.txt
- # How frequently (in seconds) the adapter will check for a stop file when running
- **adapter.stopFile.checkInterval**=5

- # Adapter can be tied to a core where required. Integer >= 0. -1 means ignore
- **adapter.coreAffinity**=-1



AQUAQ Analytics

Adapter Config

- # Type of message Adapter will process. Currently running for QUOTE, TRADE
- **adapter.messageType**=QUOTE
- # Adapter can filter messages on SYM. Allows different Adapter threads to run independently
- **adapter.messageFilter**=JUVE.MI,VOD.L,HEIN.AS
- # Redundant. May come back in.
- adapter.waitTime.whenNoMsgs=50

- # Filesystem location of Chronicle queue to process
- **chronicle.source**=C:\\ChronicleQueue\\Producer\\quote
- # Named Tailed name. Allows re-start.
- **chronicle.tailerName**=quoteTailer

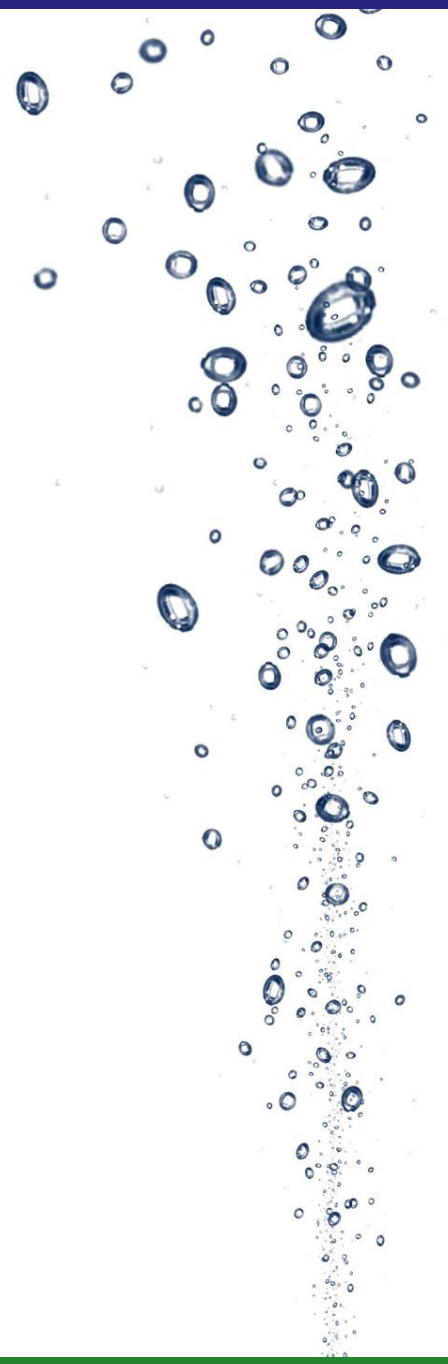
AQUAQ Analytics

Adapter Config

- # Kdb connection details
- **kdb.host**=localhost
- **kdb.port**=5000
- **kdb.login**=username:password
- **kdb.connection.enabled**=true

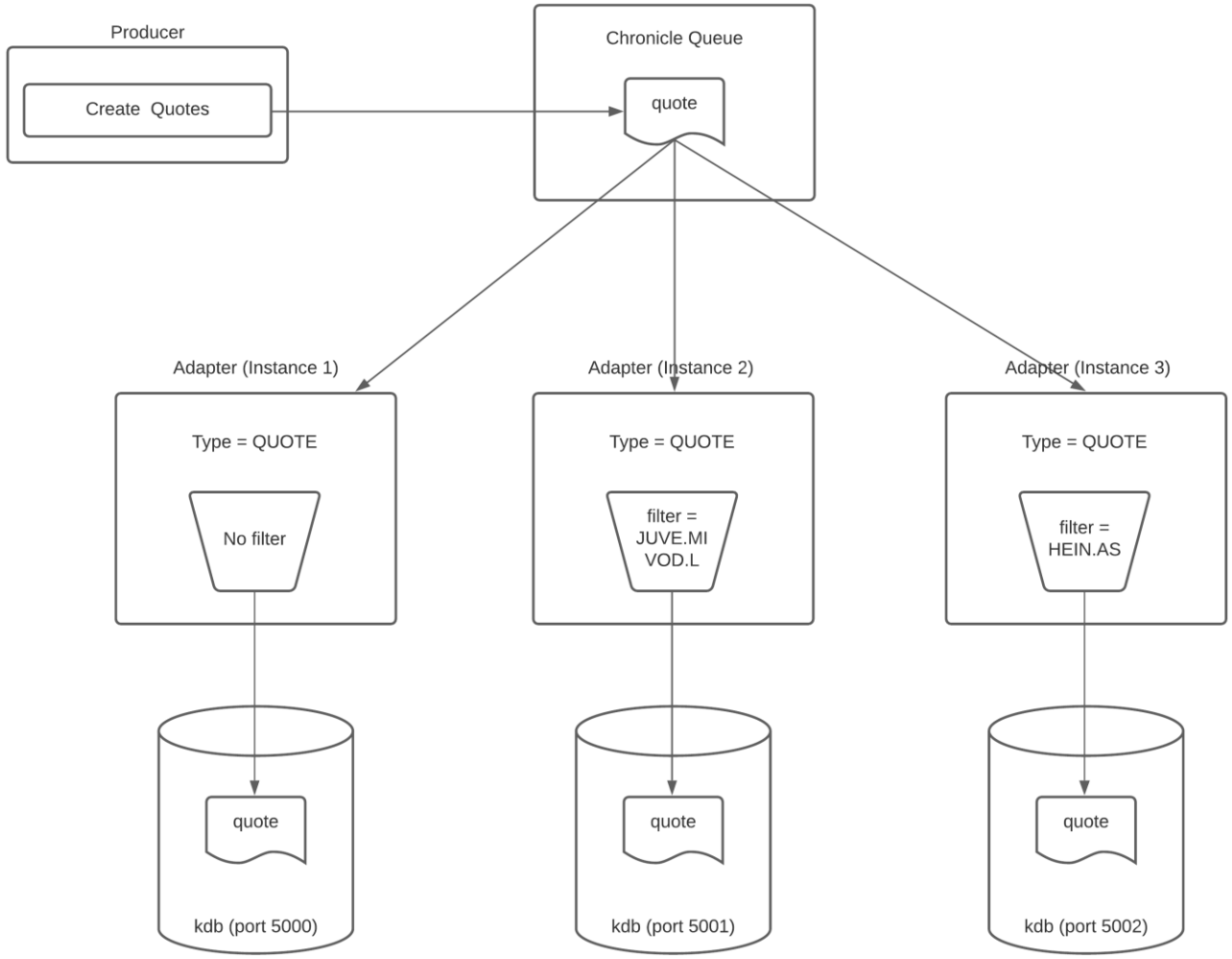
- # Kdb destination table quote, trade
- **kdb.destination**=quote
- # Kdb destination helper function
- **kdb.destination.function**=.u.upd

- # Max batch size for send to Kdb
- **kdb.envelope.size**=100



AQUAQ Analytics

Adapter Demo



AQUAQ Analytics

Adapter Requirements

- Read data from a Chronicle Queue source ✓
- Write data to a kdb+ destination ✓

  - Configurable source and destination ✓
  - Configurable adapter “type” ✓
  - Configurable mapping from source to destination format ✓
  - Configurable batching on kdb+ write side ✓
  - Error handling, rollback on failure ✓

AQUAQ Analytics

Benchmarking

- Done
  - Latency of end to end process per message
    - Worked closely with Chronicle (thanks Roger Simmons)
    - Java Latency Benchmark Harness (JLBH)
- *TODO*
  - *Throughput*
  - *Burst testing*

AQUAQ Analytics

Benchmarking (Latency)

- JLBH Settings:
  - Warmup: 50k; Runs: 5, Iterations: 300,000; Throughput: 30,000 = 1 message every 33us(microseconds)
- Adpater Settings:
  - Type = QUOTE; kdb.envelope.size=100; adapter.coreAffinity=2

```
-------------------------------- SUMMARY (end to end) ---------------------------------
Percentile   run1         run2         run3         run4         run5      % Variation
50:          12.64        12.77        12.77        12.70        12.83        0.67
90:          14.24        14.05        13.79        13.73        13.60        2.15
99:          21.44        19.90        19.90        19.90        19.90        0.00
99.7:        33.15        23.23        23.10        22.85        22.59        1.85
99.9:        73.47        73.47        73.47        71.94        72.45        1.40
worst:       2940.93      3432.45      4964.35      3858.43      3301.38      25.14
---------------------------------------------------------------------------------------
```

- Enterprise grade infrastructure.
  - Server 1: Chronicle Queue + Adapter (single thread, tied to one core)
  - Server 2: kdb+
  - High speed network connection between servers.

AQUAQ Analytics

Benchmarking (Throughput)

- Single instance of Adapter
  - Processing all SYM's together e.g. [VOD.L, HEIN.AS, JUVE.MI….]
  - *Local PC setup > 110k per second*
  - *Enterprise infrastructure ?? x2 ??*

- Multiple instances
  - Multiple threads
  - Use filtering config in Adapter
    - VOD.L instance
    - HEIN.AS instance
    - Etc..
  - ??

AQUAQ Analytics

More information

Project Repo: https://github.com/AquaQAnalytics/kdb-chronicle-queue

Chronicle Queue: https://chronicle.software/libraries/queue/

Kdb+: https://code.kx.com/q/

“Gang of Four” Design Patterns: https://springframework.guru/gang-of-four-design-patterns/

Java Latency Benchmark Harness (JLBH): https://github.com/OpenHFT/JLBH

AQUAQ Analytics

Chronicle Queue adapter to kdb+

Any questions or feedback?

Upcoming:

- Thursday 25th March: Level 2 and Level 3 Data Storage Optimisation
- Thursday 1st April: Interfacing kdb+ with Refinitiv's TickHistory in Google BigQuery
- Thursday 15th April: Introduction to Shakti by Fintan Quill, Director of Sales Engineering