High Frequency Trading and NoSQL

Peter Lawrey
CEO, Principal Consultant
Higher Frequency Trading

Agenda

Who are we?

Brief introduction to OpenHFT.

What does a typical trading system look like

What requirements do these systems have

OpenHFT performance.

Who are we

Higher Frequency Trading is a small consulting and software development house specialising in

- Low latency, high throughput software
- 8 developers in Europe and USA.
- Sponsor HFT related open source projects
- Core Java engineering

Who am I?

Peter Lawrey

- CEO and Principal Consultant
- 3rd on Stackoverflow for Java,
  most Java Performance answers.
- Founder of the Performance Java User's Group
- An Australian, based in the U.K.

What is our OSS

Key OpenHFT projects

- Chronicle, low latency logging, event store and IPC. (record / log everything)
- HugeCollections, cross process embedded persisted data stores. (only need the latest)

Millions of operations per second.

Micro-second latency.

With other NoSQL databases

Uses with NoSQL

- Off heap cache of data from a DB.
- Low latency queue for persisting to a DB
- Map which replicates only the latest values. Supports very high update rates by only replicating the latest value.

What is HFT?

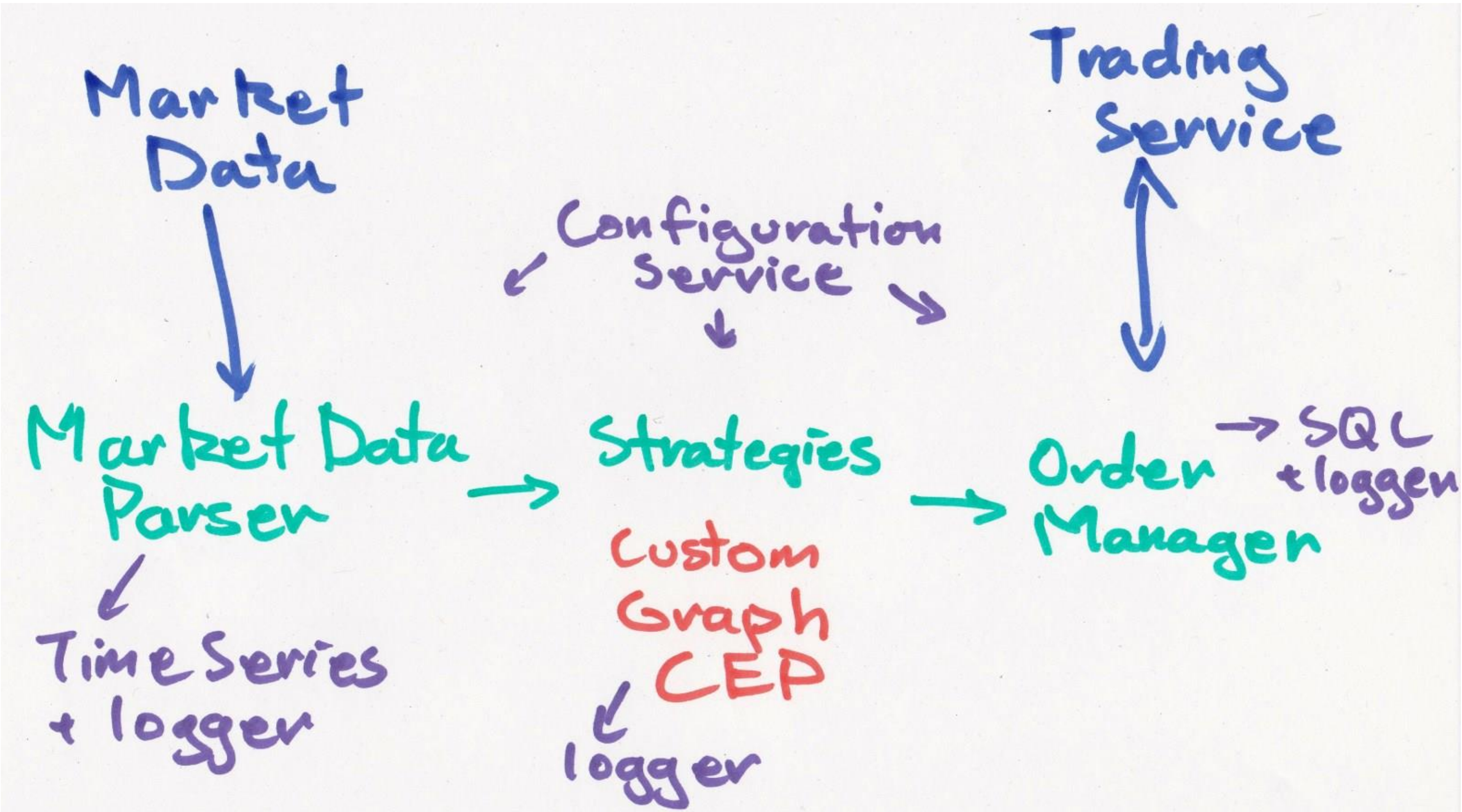
- No standard definition.
- Trading faster than a human can see.
- Being fast can make the difference between making and losing money.
- For different systems this means typical latencies of between
  - 10 micro-seconds and
  - 10 milli-second.

(Latencies external to the provider)

Time scales every developer should know.

Operation	Latency	In human terms
L1 Cache hit	1 ns	A blink of an eye (~20 ms)
L2 Cache hit	3 ns	Noticeable flicker
L3 Cache hit	10 – 20 ns	Time to say "A"
Main memory	70 – 100 ns	Time to say a ten word sentence
Signal down a 200m fibre cable	1 µsec	One slide (speaking quickly)
SSD access	5 – 25 µsec	Time to reheat a meal (3 mins)
HDD access	8 msec	Time to flight around the world. (1.8 days)
Network packet from Germany to the USA	45 msec	Waiting for a 7 working day delivery

Simple Trading System



Event driven processing

Trading system use event driven processing to minimise latency in a system.

- Any data needed should already be loaded in memory, not go off to a slow SQL database.
- Each input event triggers a response, unless there is a need to limit the output.

Critical Path

A trading system is designed around the critical path. This has to be as short in terms of latency as possible.

- Critical path has a tight latency budget which excludes many traditional databases.
- Even the number of network hops can be minimised.
- Non critical path can use tradition databases

Critical Path databases

- Time Series databases
  - Kdb, kona
  - InfluxDB
  - OpenTSDB

Designed for millions of writes per second.

Column based database => 100 Million operations per second e.g. sum a column.

Critical Path Databases

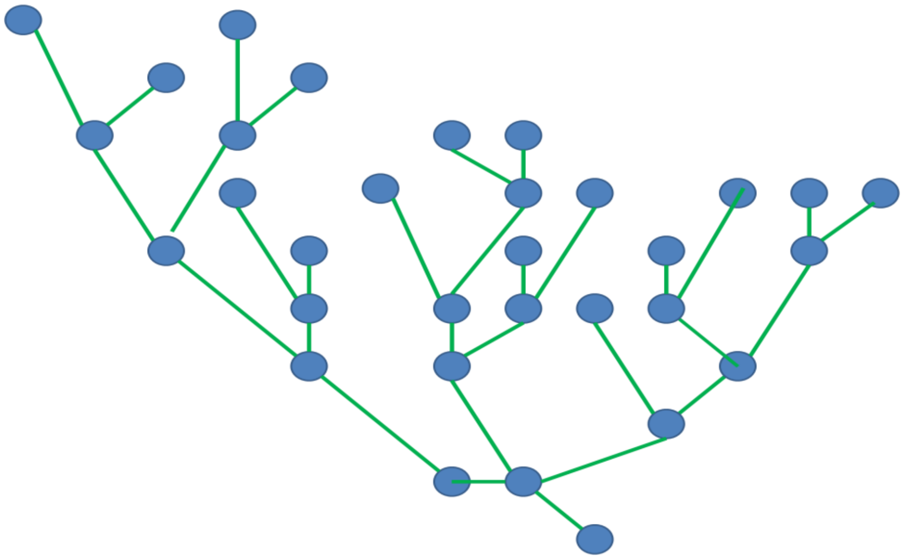
STAC-M3 Shasta suite (allows solution to exploit available memory)

	MEAN RESPONSE TIMES (ms)		
STAC-M3.β1...	**1 TB system (KDB140206)**	**6 TB system (KDB140116)**	**Speedup**
10T.MKTSNAP/s.TIME	2,569	57	44.8
10T.STATS-AGG/s.TIME	19,213	2,871	6.7
50T.STATS-UI/s.TIME	17,707	3,291	5.4
1T.YRHIBID/s.TIME	1,213	287	4.2
100T.VWAB-12D-NO/s.TIME	17,150	4,348	3.9
100T.STATS-UI/s.TIME	15,145	6,293	2.4
10T.THEOPL/s.TIME	81	46	1.7
10T.VOLCURV/s.TIME	14,910	10,253	1.5
1T.YRHIBID-2/s.TIME	339	280	1.2
1T.STATS-UI/s.TIME	222	227	1.0
1T.VWAB-D/s.TIME	21	22	0.9
1T.WKHIBID/s.TIME	24	27	0.9
1T.MOHIBID/s.TIME	43	50	0.9
10T.STATS-UI/s.TIME	716	940	0.8
1T.QTRHIBID/s.TIME	51	77	0.7

February 18, 2014 Source: www.STACresearch.com

HFT

Critical Path data store



HFT strategies are;

- described using graphs.
- handle events in real time ~10 – 100 µsec.
- cache state rather than query a database.
- all custom written libraries AFAIK.

Critical Path data store

Logging is performed by appending to memory mapped files.

OpenHFT's Java Chronicle makes this easier to do in Java in a GC-free, off heap, lock less way.

Such low level coding is relatively easy in C or C++.

Non-critical Datastore

Configuration management

- ZooKeeper, etcd
- Plain files with Version control
- LDAP
- Any distributed key-value store. e.g. MongoDB

Big Data

Back testing a HFT system is critical and a number of solutions are available

- Hadoop
- Matlab
- Time series
- R

Operational Infrastructure

Control and management infrastructure

- JMS, JMX
- Tibco RV, LBM
- Terracotta
- MongoDB

Reliable persistence

Trades and Orders are high value data and less voluminous than Market data or strategy results.

- Typically SQL Database.
- Sometimes multiple databases for different applications.

Why use more exotic database?

- Mostly for high throughput.
  - Million per second in one node.
- Often for low latency.
  - Latencies well below a milli-second.

Why wouldn't you use exotic DB

- Not easy to learn, high knowledge investment.

  `(!R)@&{&/x!/:2_!x}'!R`

- Often harder to use.
  - Less management tools.
  - Not designed to work with web applications.
- More sensitive to the details of the hardware and what else is running on the same machine.

Low latency at high throughput

Java Chronicle is designed as a low latency logger and IPC.

At one million small messages per second

- Almost zero garbage
- Latency between processes around 1 micro-second
- Concurrent readers and writers

Supports bursts of 10 million messages/sec.

Chronicle and replication

Replication is point to point (TCP)

Server A records an event

- replicates to Server B

Server B reads local copy

- B processes the event

Server B stores the result.

- replicates to Server A

Server A replies.

Round trip
25 micro-seconds
99% of the time

GC-free
Lock less
Off heap
Unbounded

HugeCollections

HugeCollections provides key-value storage.

- Persisted (by the OS)
- Embedded in multiple processes
- Concurrent reads and writes
- Off heap accessible without serialization.

HugeCollections and throughput

SharedHashMap tested on a machine with 128 GB, 16 cores, 32 threads.

String keys, 64-bit long values.

- 10 million key-values updated at 37 M/s
- 500 million key-values updated at 23 M/s
- On tmpfs, 2.5 billion key-values at 26 M/s

HugeCollections and latency

For a Map of small key-values (both 64-bit longs)

With an update rate of 1 M/s, one thread.

Percentile	100K entries	1 M entries	10 M entries
50% (typical)	0.1 μsec	0.2 μsec	0.2 μsec
90% (worst 1 in 10)	0.4 μsec	0.5 μsec	0.5 μsec
99% (worst 1 in 100)	4.4 μsec	5.5 μsec	7 μsec
99.9%	9 μsec	10 μsec	10 μsec
99.99%	10 μsec	12 μsec	13 μsec
worst	24 μsec	29 μsec	26 μsec

Bonus topic: Units

A peak times an application writes 49 “mb/s” to a disk which supports 50 “mb/s” and is replicated over a 100 “mb/s” network.

What units were probably intended and where would you expect buffering if any?

HFT

Bonus topic: Units

A peak times an application writes 49 MiB/s to a disk which supports 50 MB/s and is replicated over a 100 Mb/s network.

MiB = 1024^2 bytes

MB = 1000^2 bytes

Mb = 125,000 bytes

The 49 MiB/s is the highest rate and 100 Mb/s is the lowest.

Bonus topic: Units

Unit	bandwidth	Used for
mb - miili-bit	mb/s – milli-bits per second	?
mB - milli-byte	mB/s – milli-bytes per second	?
kb – kilo-bit (1000)	kb/s – kilo-bits (baud) per second	Dial up bandwidth
kB – kilo-byte (1000)	kB/s – kilo-bytes per second	?
Mb – mega-bit (1000^2)	Mb/s – mega-bits (baud) per second	Cat 5 ethernet
MB - mega-byte (1000^2)	MB/s – mega bytes per second	Disk bandwidth
Mib – mibi-bit (1024^2)	Mib – Mibi-bits per second	?
MiB – mibi-byte (1024^2)	MiB – Mibi-bytes per second	Memory bandwidth
Gb – giga-bit (1000^3)	Gb/s – giga-bit (baud) per second	High speed networks
GB – giga-byte (1000^3)	GB/s – giga-byte per second	-
Gib – gibi-bit (1024^3)	Gib/s – gibi-bit per second	-
GiB – gibi-byte (1024^3)	GiB/s – gibi-byte per second.	Memory Bandwidth

Q & A

https://github.com/OpenHFT/OpenHFT

@PeterLawrey

peter.lawrey@higherfrequencytrading.com

HFT