High Performance Trading System

Jingqi Xu@Nextop

What makes a high performance trading system?

- Cost
- Speed
- Manageable
- Scalable & HA
- …

Cost

Reusable, business independent infrastructure code.

- @See common maven submodule.

Cost

Scalable deployment.

- Deployment topology has noting to do with the source code’s structure, maven modules, etc.
- Able to deploy apps not only across multiple JVMs, but also in fewer JVMs, even one!
- @See remoting APIs which try to eliminate the difference between inner-JVM and inter-JVM communication.

Cost

Testable & Monitorable

- Programming skills & tricks to facilitate functional test.
- @See monitor APIs which collect & analyse about 1M records/day in real time. (Log can also help, but it is useless under certain circumstances, such as high frequency operations).

Speed

Focus on the latency distribution of the whole trading loop.

- Profiler helps.
- Micro benchmarks also help.

Speed

Low latency networking.

- Brokerless messaging (@See messaging APIs). JMS ? No!
- Serialization is crucial. Java serialization? No! Kryo or protobuf? Yes, it is fast, but can be better(@See marshalling APIs).
- To improve throughput, bundle messages when possible.

Speed

Trade in memory.

- Software transaction framework(@See stx APIs). You don’t want dirty or corrupted data when encountering an exception, but how to do it without using database transaction?
- Avoid accessing remote data(database, redis, etc)

Speed

Garbage collection optimization.

- Off-heap storage, @See trading storage APIs.

Speed

Avoid shared data concurrency & context switch .

- Concurrency is hard, bug is not reproducible.
- Lock free & CPU cache-friendly algorithms
- Immutable objects help.
- @See pipeline APIs.

Speed

Fast persistence.

- Eventually Consistency.
- Merge IO operations in batch.
- Be aware of OS page cache, fsync or fdatasync?

Manageable

Don’t be slow, fail-fast instead.

Manageable

Able to downgrade service level under load instead of crash.

- Able to control service level by dynamic switches, @Ses config APIs.

Manageable

Why your app server is not responsible but system's load is not that high?

- Asynchronous servlet helps.

Manageable

Want to manage your app on the run?

- @See jmx APIs
- @See shell APIs

Scalable & HA

Proper sharding.

- Join & distributed transaction are expensive, should be avoided.

Manageable

Log should be avoided, but sometimes I really need it!

- Able to dynamic switch on the log on requirement.
  @See debug APIs which dynamically control the log's verbosity at account, symbol & application level.

Manageable

We use distributed cache(@see cache APIs), but how to make sure the cached data is consistent?

- @See digest APIs

Scalable & HA

How does Trader take over?

- Able to take over in seconds.
- Even though the database is down.