PoC for a modern distributed pricing architecture based on open source components

Andreas Pfadler, d-fine GmbH
Motivation for this talk

Basic ingredients

Live Demo

Outlook
Motivation

» There exist a number of commercial closed source platforms and in-house systems which in some way combine
  › Analytics, i.e. pricing libraries
  › Grid computing frameworks
  › Grid wide caching / in memory computing
  › HTTP/Rest interfaces and HTML5 GUIs
  › Excel integration
  › Workflows for product/model development and deployment

» In this talk we show how we can build similar systems using open source software, only.

» Our main ingredients are
  › QuantLib with Swig Java bindings
  › Apache Ignite
  › Jetty
  › Scala
  › AngularJS
Basic Ingredients
Apache Ignite (1)

» Originally developed by Grid Gain Systems
» Recently promoted to a top level project of the Apache Software Foundation
» Can be thought of as a kind of **distributed in memory data fabric**.
» Based on Java. Configurable using Spring.
» Sometimes seen as a „competitor“ to Apache Spark.

**Main Features:**

- Distributed Computing
- Cluster Wide In-Memory Caches
- Service Deployment
- Distributed Datastructures (Queues, etc.)

Accessible through easy to use high-level APIs
Apache Ignite (2)

» **Example 1: distributed closure** (see Ignite documentation)

```java
IgniteCompute compute = ignite.compute();

// Execute closure on all cluster nodes.
Collection<Integer> res = compute.apply((String::length, Arrays.asList("How many characters".split(" ")));

// Add all the word lengths received from cluster nodes.
int total = res.stream().mapToInt(Integer::intValue).sum();
```

» **Example 2: Query a grid-wide cache using a predicate** (see Ignite Documentation)

```java
IgniteCache<Long, Person> cache = ignite.cache("mycache");

// Find only persons earning more than 1,000.
try (QueryCursor cursor = cache.query(new ScanQuery((k, p) -> p.getSalary() > 1000)))
{
  for (Person p : cursor)
    System.out.println(p.toString());
}
```

(SQL based Cache queries are also possible, but require some additional configuration effort)
HTTP/HTML5 Interface: Jetty and AngularJS

Jetty

» Traditional web app deployed as packages on some application server
» If your application is already implemented in some sort daemon process anyway and you don’t want to run a full application server, why not directly embed a http server?
» Standard option in the Java world: **Jetty**
» Features easy integration (Maven packages) and allows for running servlets or even fully fledged applications packaged in a .war file.

AngularJS

» Maintained mostly by Google
» Targeted at single page apps based on MVC pattern
» Features custom HTML5 directives and bidirectional data binding
» Makes JS development less painful…
Quantlib and Scala Integration

Quantlib / JNI / Thread-Safety

» Thread safety: Since we will embed Quantlib through JNI and run several Java threads in parallel across a number of Ignite nodes, this is of paramount importance.

» Need to make use of Klaus Spanderen's implementation of the thread safe observer pattern (https://hpcquantlib.wordpress.com/2013/07/26/multi-threading-and-quantlib/) – Thank you very much, Sir!

Scala Integration

» Nice to have: payoff-scripting facility

» Also nice to have: Offer users a way to quickly achieve results using an embedded scripting facility similar to IPython notebooks, etc.

» To solve both of these problems we integrate a Scala interpreter into our system (although something like Jython would be equally well suited…)

The Big Picture

Clients
- HTML 5 GUI
- Excel
- Batchprograms

Ignite Cluster
- Master Node
  - Embedded HTTP Server
    - Cache
  - Worker Node
    - Cache
  - Worker Node
    - Cache
  - Worker Node
    - Cache

Libs
- Quantlib / JNI
- Other
Live Demo
Outlook

A lot remains to be done

» The architecture provides a solid foundation for distributed pricing combined with an easy to use cluster wide caching mechanism.

» However, for concrete applications a lot remains to be inspected closer or worked on in the future:
  › Overall stability and performance / Integration into production environments
  › Caching of compiled Scala scripts / XML Contexts
  › High-level market data API on top of Ignite cache and integration of live market data feeds
  › Security considerations
  › Flexible and easy to use ways for defining new products and models
  › Clear workflows for quantitative development and production deployment

Key Message

» There are great open source projects that help you building sophisticated and easy to use pricing platforms

» Don’t start developing from scratch and keep an open mind with regards to what’s out there in terms of (not only finance related) open source software
Dr. Andreas Pfadler
Senior Consultant
andreas.pfadler@d-fine.de
Tel +49 (0)69 90737 0
Mobil +49 (0)162 2630029

d-fine GmbH
Frankfurt
München
London
Wien
Zürich

Zentrale
d-fine GmbH
Opernplatz 2
D-60313 Frankfurt/Main

Tel +49 69 90737-0
Fax +49 69 90737-200
www.d-fine.com
d-fine