How to build a European scale instant payments platform

SIA and Red Hat

Giovanni Fulco, Giuseppe Bonocore, Ugo Landini

May 2018
SIA in a nutshell

THE GROUP

CARDS 6.1 billion operations
PAYMENTS 3.3 billion transactions
CAPITAL MARKETS 56.2 billion financial transactions
NETWORK 784 terabytes of data carried
SIA Vision

“Everyone will be able to use their own money anytime, anywhere, simply and securely.”

Red Hat Vision

“Open unlocks the world’s potential.”
SIA & Red Hat

2000 - First application delivered on Linux OS

2004 - New Trading platform on RHEL 2.1

2010 - Real Time Kernel Adoption

2013 - Partnership to develop JDG Library

2016 - Instant Payment Platform
Wire transfer without Instant Payments
Select amount:

100 €

SEND
... File transfer “magic” happens...
... up to 4 days...
Wire Account
Received:
100 €
OK
Wire transfer with Instant Payments
My Bank

New Wire Transfer
My Bank

Select Recipient:

1. Giovanni Fulco
2. Mattia Ronchi
3. Giuseppe Bonocore
   - Ugo Landini
My Bank

Select amount:

100 €

Instant Payment: ✔

SEND
< 50mS
My Homebanking

Wire Account
Received:
100 €

OK
What’s under the hood
What’s under the hood

- **JDG**, In Memory Data Grid
  - Scalable and elastic grid for exceptional performances
- **AMQ**
  - High performance messaging
- **FIS**, Camel on OpenShift
  - Superdynamic integrations
- **CASSANDRA**, NoSQL DB
  - Store transaction history for non repudiation / antifraud
Request sent to bank

Bank calls Instant Payments

Request is stored in JDG

JDG replicates cross site

Req is stored in Cassandra

Tx is propagated to bank

Receiver asked for confirm

Receiver

Sender
Bank Confirms Payment

Confirmation is stored in JDG
JDG replicates cross site
Req is stored in Cassandra
Tx is propagated to bank
Sender is notified

< 50mS !
JDG, In Memory Data Grid
Memory is the new disk

<table>
<thead>
<tr>
<th>LATENCY COMPARISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Pizza Delivery Time Conversion

- 30 minutes
- 3 weeks
- 5.5 years

<table>
<thead>
<tr>
<th>EXAMPLES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex pricing algorithm run from 12+hours to &lt;30 seconds</td>
</tr>
<tr>
<td>Real-time billing and payment</td>
</tr>
<tr>
<td>Major telco 2,000 SMS notifications &amp; 500 offers fulfills per second</td>
</tr>
<tr>
<td>Beauty products company manages surge in order volume to 20k per hour during peak promotion</td>
</tr>
</tbody>
</table>
JDG: JBoss Data Grid

- Polyglot
- Extreme Performance
- Linear Scalability
- Fault Tolerant
- Event driven

See "memory" across machines as a unified data store

Read-through, write-through, write-behind

In-memory data = all data needed is to be kept in memory
Grid = too big for one node, data is distributed in cluster
Memory cluster
Distributed mode (typically 1 replica)
Adding a value

PUT(K1, V)
Adding a value

PUT(K1,V)
Values are distributed

PUT(K2, V)
Values are distributed

PUT(K2, V)
Keys are distributed consistently through the cluster
Consistent Hashing

Given the same topology (i.e. number of nodes), same keys are always hashed on the same nodes. Consistent hashing can be calculated on clients too, so they can reach the right node for any given key.
So reads go directly on the “right” node...

GET(K2)
... thx to consistent hashing
If you lose or add a node (topology change)...

Diagram showing a network topology with nodes labeled K1, K2, K3, and K4, with a node K5 marked as removed.
...keys are automatically redistributed
The client gets the new topology too...
Data Affinity

Data affinity means **co-locating data** together to improve performance and scalability.

Data affinity means **co-locating computing code** with data too.
Data Affinity

“Grouping” together all the affine data, for example:

- All **Customer** data
- All **Credit card** data
- Whatever **partitioning** criteria is better

Gives:

- **Highest** possible performance
- **Lowest** possible round trips
Same color == same group
Project Requirements
RT1 is an instant payment system that will provide the European payments industry with a pan-European infrastructure platform for real-time payments in euro from day one of the SEPA Instant Credit Transfer Scheme.
Constraints

5,000 tx/sec

27 Mln Payments/day

< 900 ms Roundtrip

Active/Active geo sites

0 Message loss

24/7/365 Availability
Challenges and Solutions
Due to low latency and replica requirements, we could not adopt a traditional db-centric application.
We adopted a full in-memory solution relying on Jboss DataGrid
Challenge

Cross-site replica needs to be synchronous, and the number of remote operations must be low.
Solution
Minimize remote communication

DataGrid key affinity:
Each transaction is handled by the node which owns the relevant data.
Challenge

Incoming messages need to be managed in an highly available, cross site infrastructure
Solution
Highly available messaging

AMQ Network of brokers allow highly available, zero message loss topologies (site disaster resilient)
Multi site active-active increases the risks of split brains
Datagrid owner distribution between sites allow to operate in case of network split

Solution
Avoid split brain
processed since November 2017

1 million Instant Payments

Learn more
Next Steps
Next Steps
EnMasse

Adoption of **EnMasse** (Messaging As A Service in OpenShift), to streamline the management of queues
Next Steps

Release and scale, without service disruption

Containerization with OpenShift to allow safer releases of newer version and instances
Federate other payment circuits, in order to reach more customers
THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat