Dynamically Re-Configurable Event-Driven Systems

Danny Goovaerts

CTO – The Glue

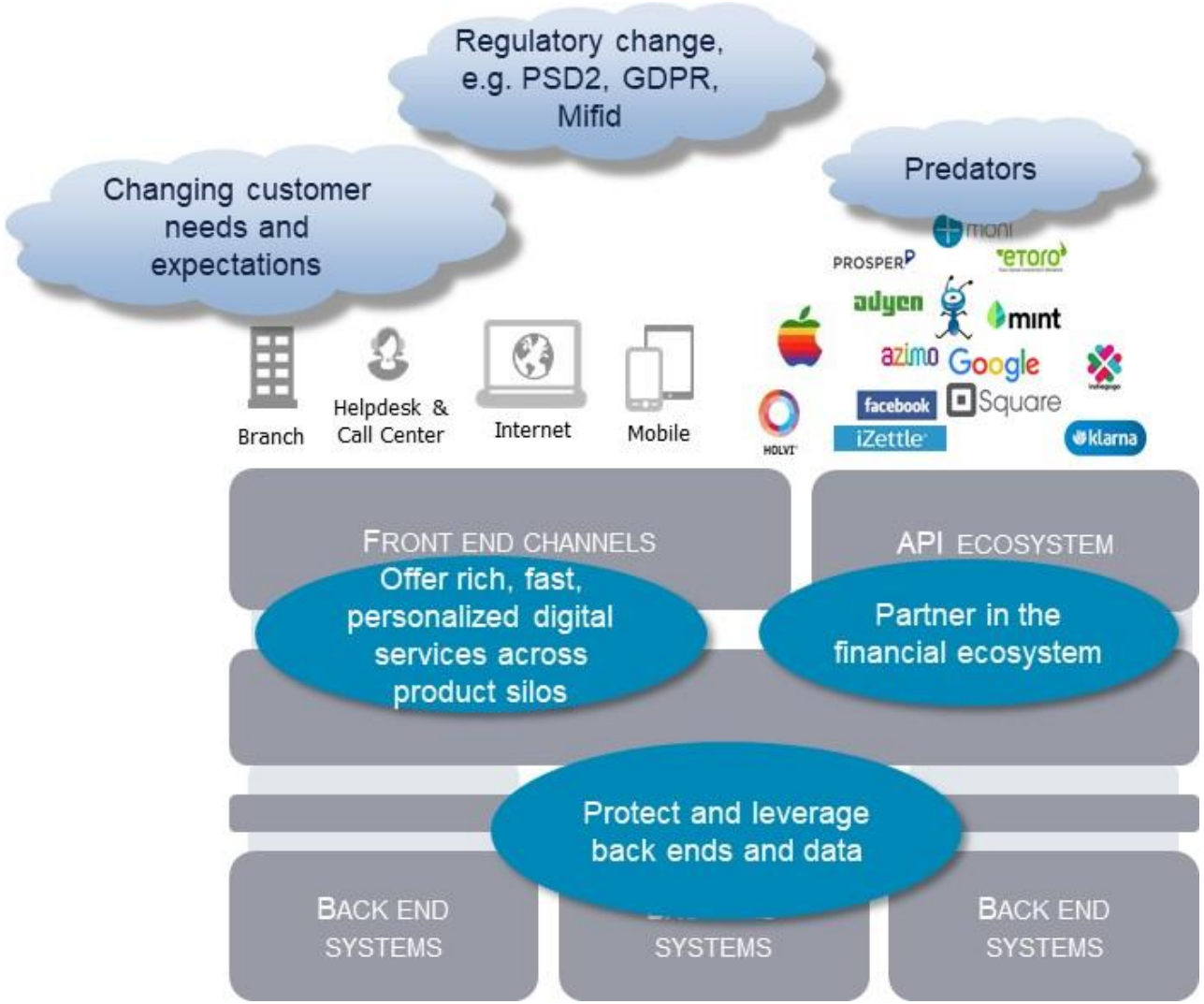
THE GLUE

FINTECH SOLUTIONS

Coming soon to a theatre near you

The ultimate banker's horror movie

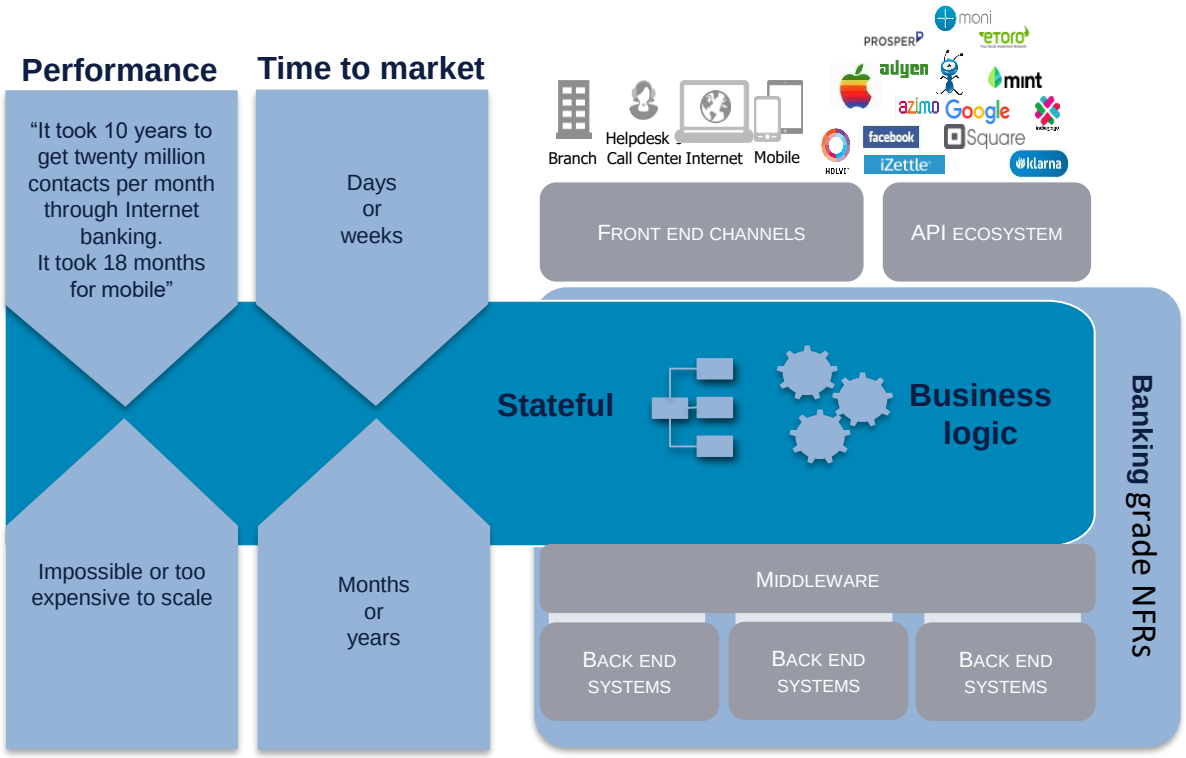
Transform incumbent banks from rigid organisations to embracing change



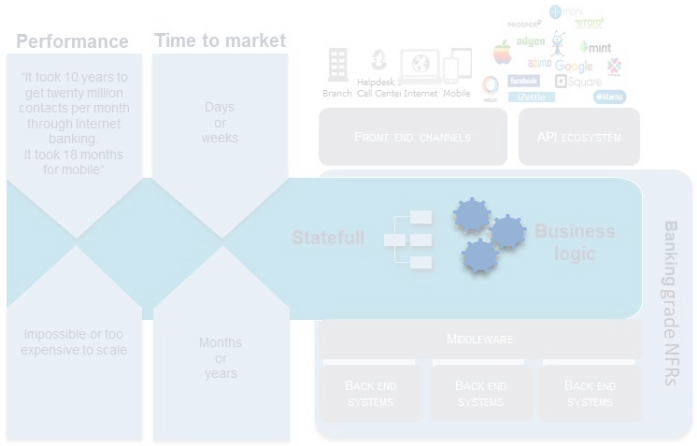
Journey of this presentation

- Which capabilities do we need for this?
- How can we combine new technologies and approaches to deliver these capabilities?

WHICH CAPABILITIES DO WE NEED FOR THIS?

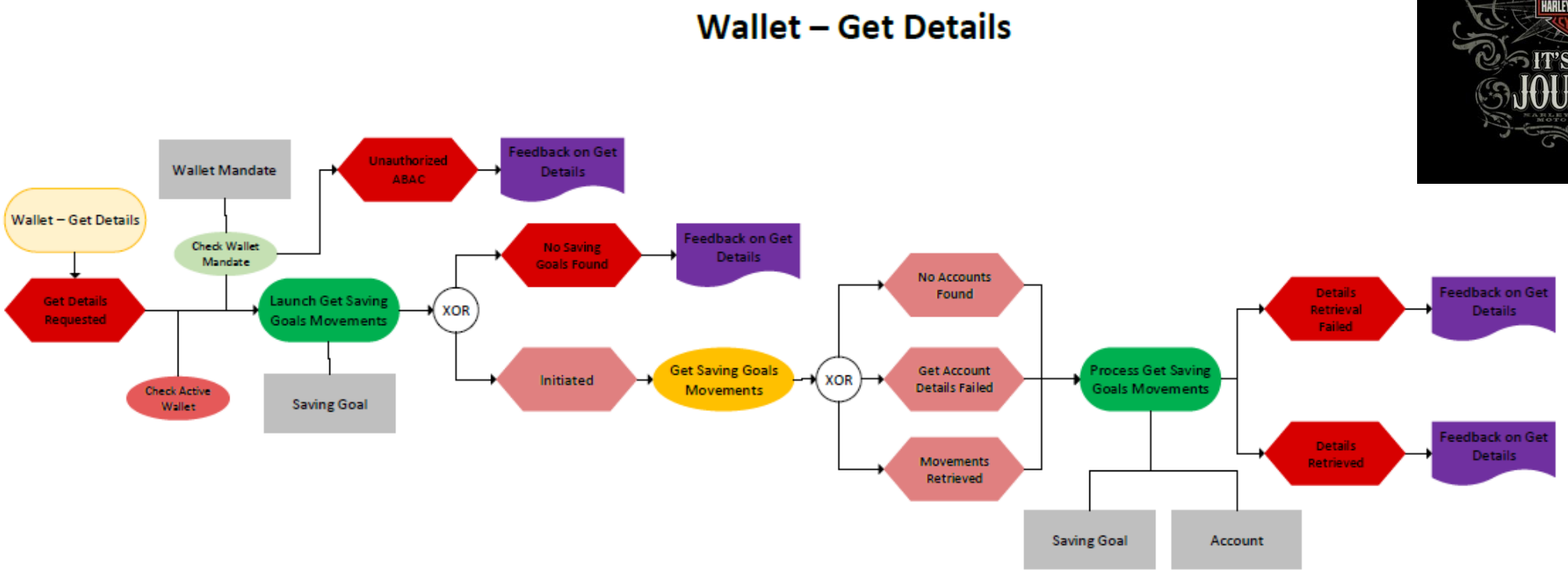


- Exactly Once Processing (EOP)
- High availability
- Scaleability

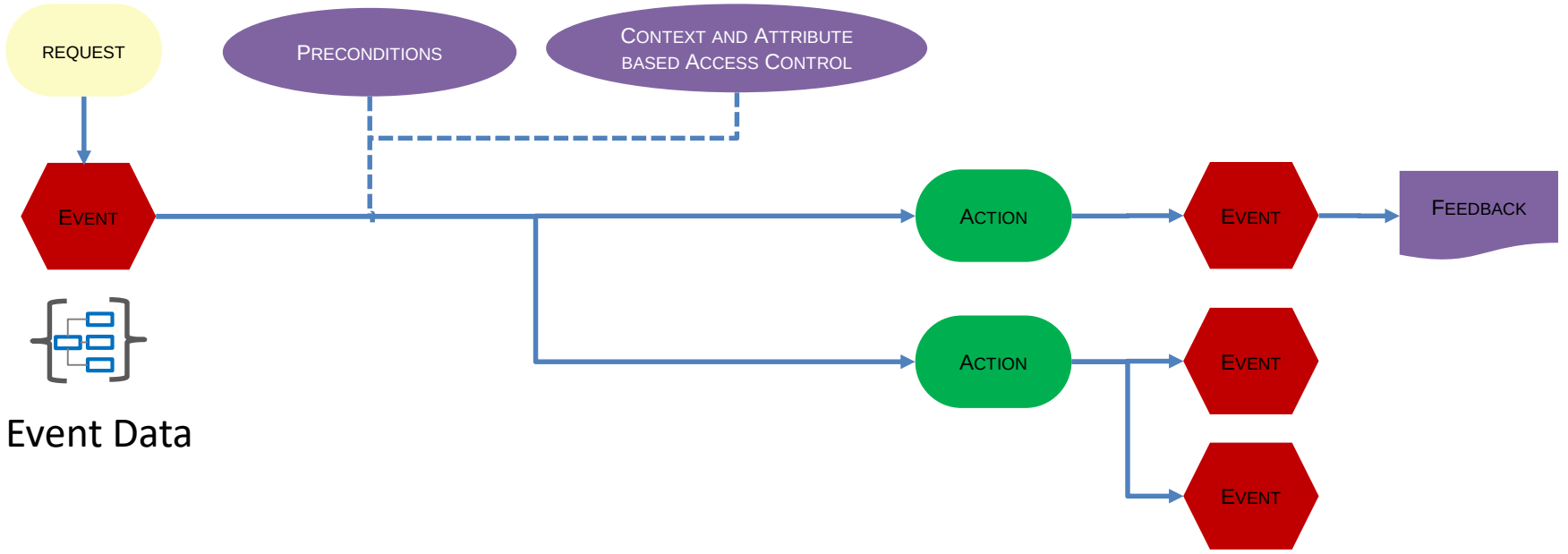


DEFINITION OF BUSINESS SERVICES

EPC (Event Processing Chain) based journey modelling



And the bead goes on ….



Journey definition is a list of beads specified using a DSL

```
package com.theglue.core.journey.pfm.wallet;

import ...

public class WalletJourney implements BusinessJourney {

    @Override
    public void configure(BusinessJourneyBuilder journey) {
        journey
                .withDataOfType(WalletJourneyData.class)
                .event(Initiated.class)
                    .isInitiating()                                          // Initiating event
                    .action(CreateJourney.class).end().end()                 // Initated event launched CreateJourney business processor
                .event(ModificationRequested.class)
                    .require(new CheckActiveWallet())                        // Pre-condition on event
                    .action(ProcessModification.class)                       // Definition of business processor to trigger
                        .require(new CheckWalletMandate()).end().end()       // ABAC check
                .event(GetDetailsRequested.class)
                    .require(new CheckActiveWallet())
                    .action(LaunchGetSavingGoalsMovements.class)
                        .require(new CheckWalletMandate()).end().end()
                .event(NoAccountsFound.class)
                    .fromJourney(GET_SAVING_GOALS_MOVEMENTS_JOURNEY)         // Listen to event of other journey (i.e. Saving Goals Mvts journey)
                    .action(ProcessGetSavingGoalsMovementsForNoAccountsFound.class).end().end()
                .event(GetAccountDetailsFailed.class)
                    .fromJourney(GET_SAVING_GOALS_MOVEMENTS_JOURNEY)
                    .action(ProcessGetSavingGoalsMovementsForGetAccountDetailsFailed.class).end().end()
                .event(PSD2GetTransHistFailed.class)                         // Listen to event of other journey (i.e. Get Account Details journey)
                    .fromJourney(GET_ACCOUNT_DETAILS_JOURNEY)
                    .action(ProcessGetAccountDetailsForPSD2GetTransHistFailed.class).end().end()
                .event(JourneyCreated.class)
                    .feedback(WalletManagementFeedbackForJourneyCreated.class)   // Feedback processor
                    .nextPossibleEvents(nextPossibleEvents)                      // Definition of next possible events
                .event(ModificationProcessed.class)
                    .feedback(WalletManagementFeedbackForModificationProcessed.class)
                    .nextPossibleEvents(nextPossibleEvents)
                .event(UnauthorizedAbac.class)
                    .feedback(WalletManagementFeedbackForUnauthorizedABAC.class).end()
```

Chacteristics of a business journey

Intrinsically stateful 

Traditional date modeled as a journey, i.e. state + processing

Short living (e.g. payment) or long living (e.g. mortgage)

```
package com.theglue.core.journey.pfm.wallet;

import com.theglue.api.ModuleReference;

public class WalletModuleReference {

    public static final ModuleReference WALLET_JOURNEY = ModuleReference.create("core", "pfm", "wallet", "1.0");

}
```

This is not stream processing

Event is defined in the context of a specific journey type (name space), e.g. “saving goal” or “wallet”

Event is either

- initiating : starts a new journey instance of the specific type : a “journey id” is created
- targeted to a specific instance of a journey type, e.g. “cancel payment event”

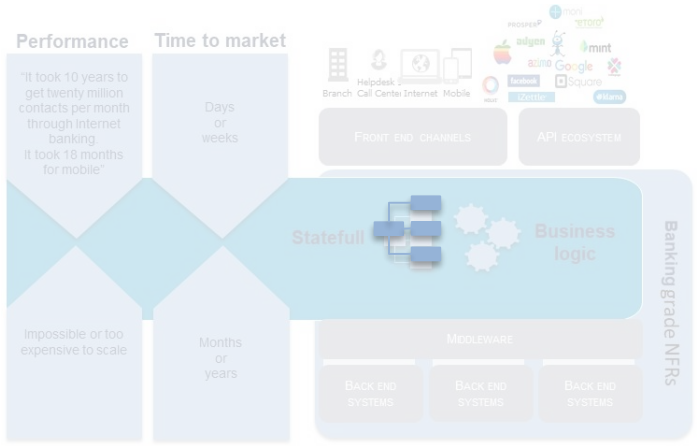
This is an actor based approach cfr. Akka

Event can originate from

- request ("API")
- backend interaction
- execution of actions of other events

The origin is captured in the context

- authenticated sender
- trust level
- geo location


Performance
Time to market
"It took 10 years to get twenty million contacts per month through Internet banking.
It took 18 months for mobile"
Days
or
weeks
Impossible or too expensive to scale
Months
or
years
Helpdesk
Branch
Call Center
Internet
Mobile
Front end channels
API ecosystem
Statefull
Business logic
Middleware
Back end systems
Back end systems
Back end systems
Banking grade NFRs

STATEFULL

Memory as primary storage : distributed in memory data grid

Operational data store for new data

High speed in memory cache for backend data (system of record is the back end)

Drivers

- performance
- high availability (sharding with primary and backup copies)
- scaleability

Apache Ignite as distributed in memory data grid

Information model defined in JSON schemas

Automatic generation of pojos which are stored in the grid

Persistent cache store

No data loss, even when the complete system is shut down

“Overflow area”

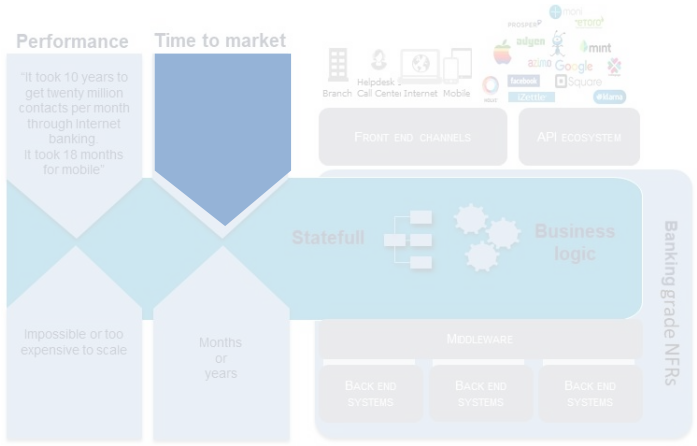
- Not all data needs to be available in memory at all times (e.g. transaction history)

Dedicated cache store

- MariaDB 10.2
- No object to relational mapping, objects are stored in JSON serialization

Queries are executed on the cache store as not all data is necessarily in memory

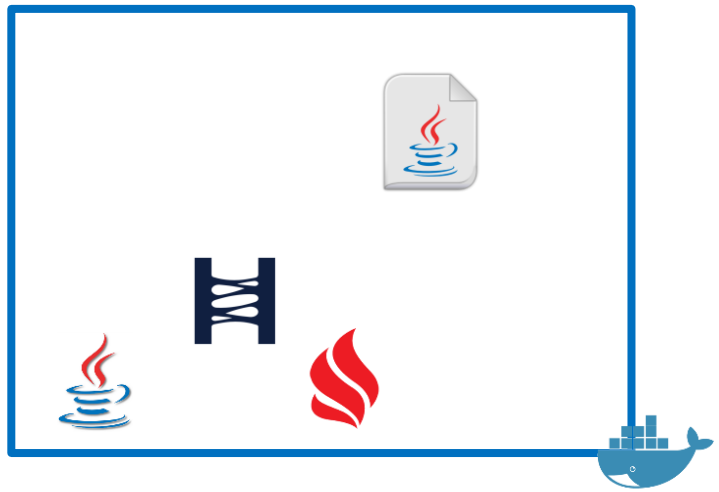
- Indexed virtual columns using JSON functions for search fields



TIME TO MARKET

Version of a journey as basic unit of deployment

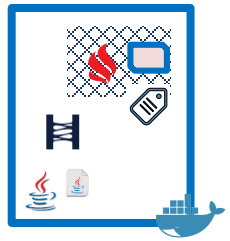
Core docker images
- JVM
- Apache Ignite :
  - Data storage
  - Event routing
- The Glue Framework
  - Event handling



Journey version specific artifacts are injected
- Journey dsl
- Journey data
- Events
- Action processors
- Queries

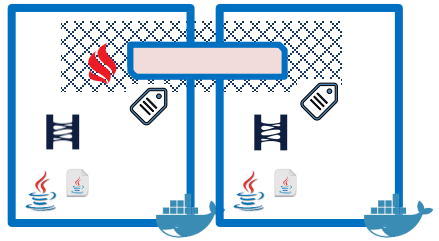
Fully self describing

Dynamically reconfigurable micro services architecture



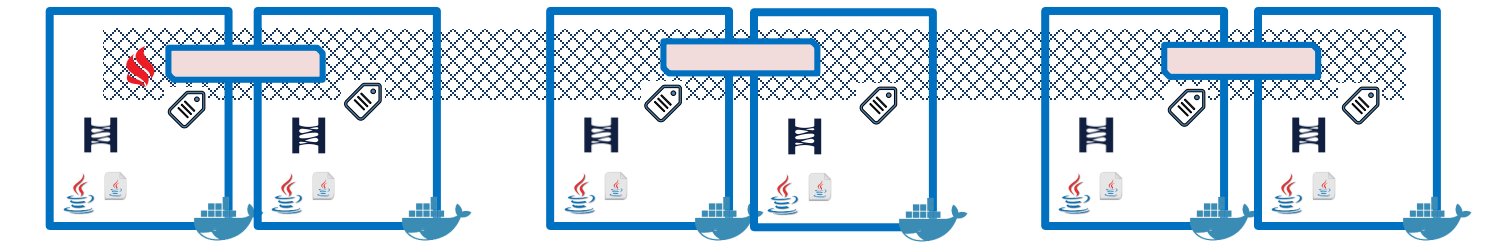
TheGlue/psd2/x2a/1.0

Dynamically reconfigurable micro services architecture



TheGlue/psd2/x2a/1.0

Dynamically reconfigurable micro services architecture



Enabler for change

A version of a journey as a microservice

A journey version is immutable. Coexistence instead of upgrading

Uplifting of journeys to a new version: forward and backward compatibility of the journey data using schema evolution

- Data is not versioned
- The way you work with and look at the data is versioned

Disruptive for IT organisations of incumbent financial institutions

EVENT ROUTING

In memory data and processing grid

IMDG offers collocation using a affinity key

- Journey data
- Events
- Jouney id is the partitioning and affinity key

Apache Ignite also supports collocation of processing and data using an affinity key

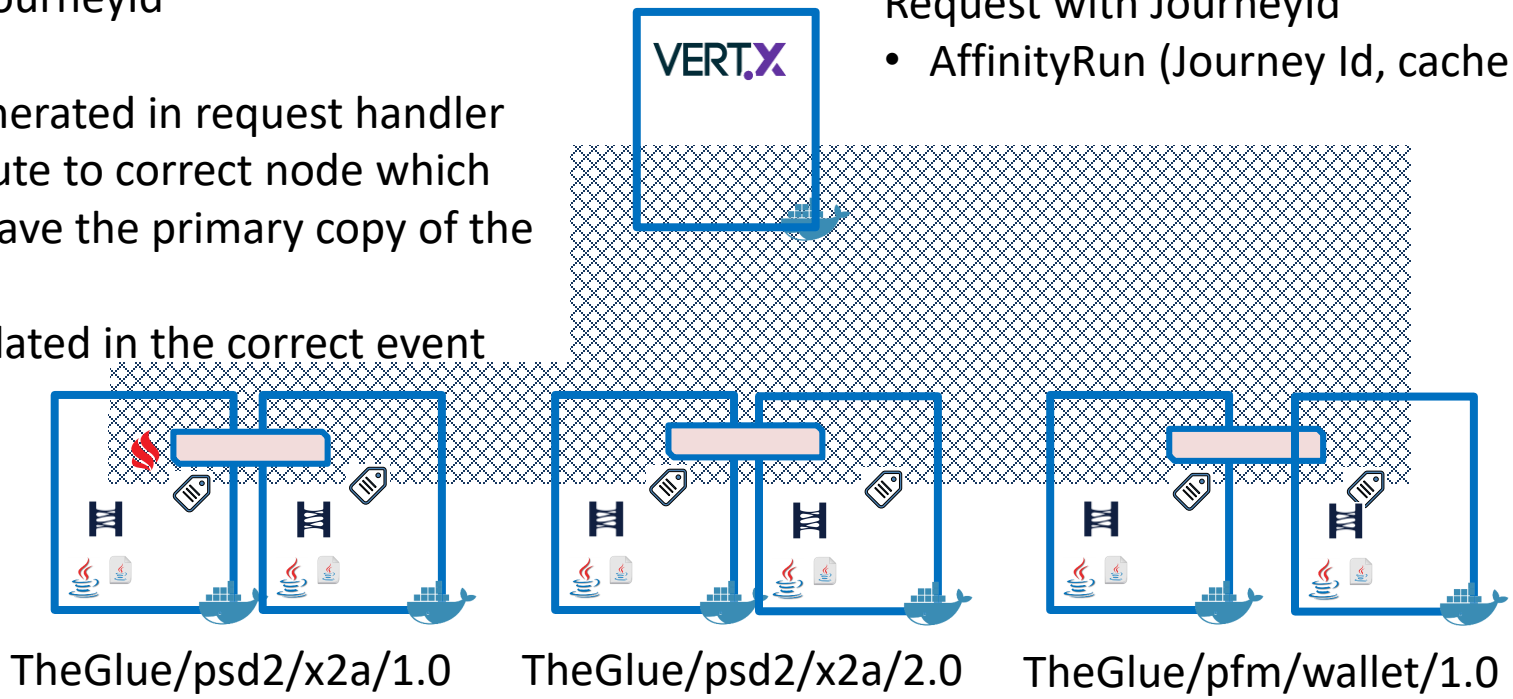
Routing of events to the correct node

Request without JourneyId

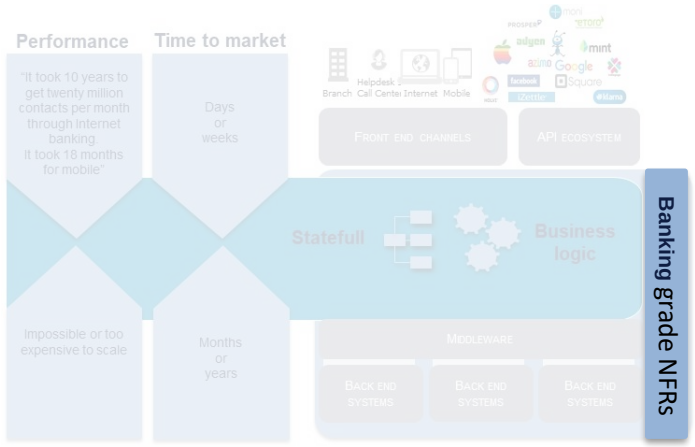
- Initiating event
- Journey id is generated in request handler
- Nodefilter to route to correct node which ultimately will have the primary copy of the data
- Request is translated in the correct event which is stored

Request with JourneyId

- AffinityRun (Journey Id, cache name)

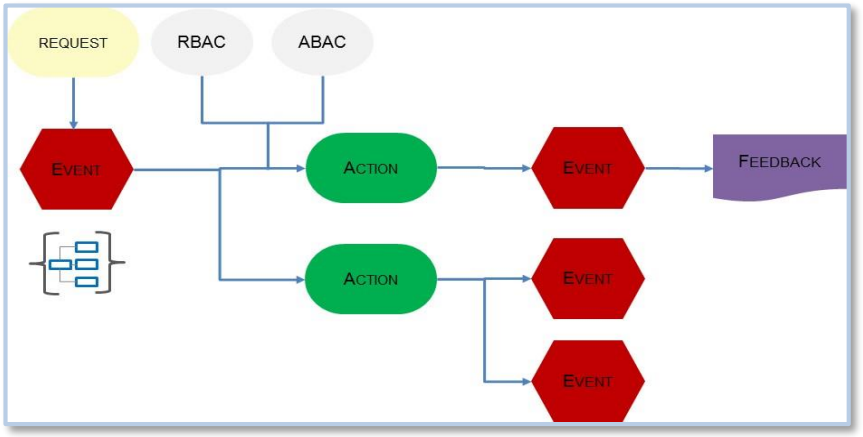


POST https://showcase-prd.theglue.com/theglue/core/psd2/credit-transfer-initiation/events/initiated/1.0



NFR : BUSINESS ROBUSTNESS (EOP)

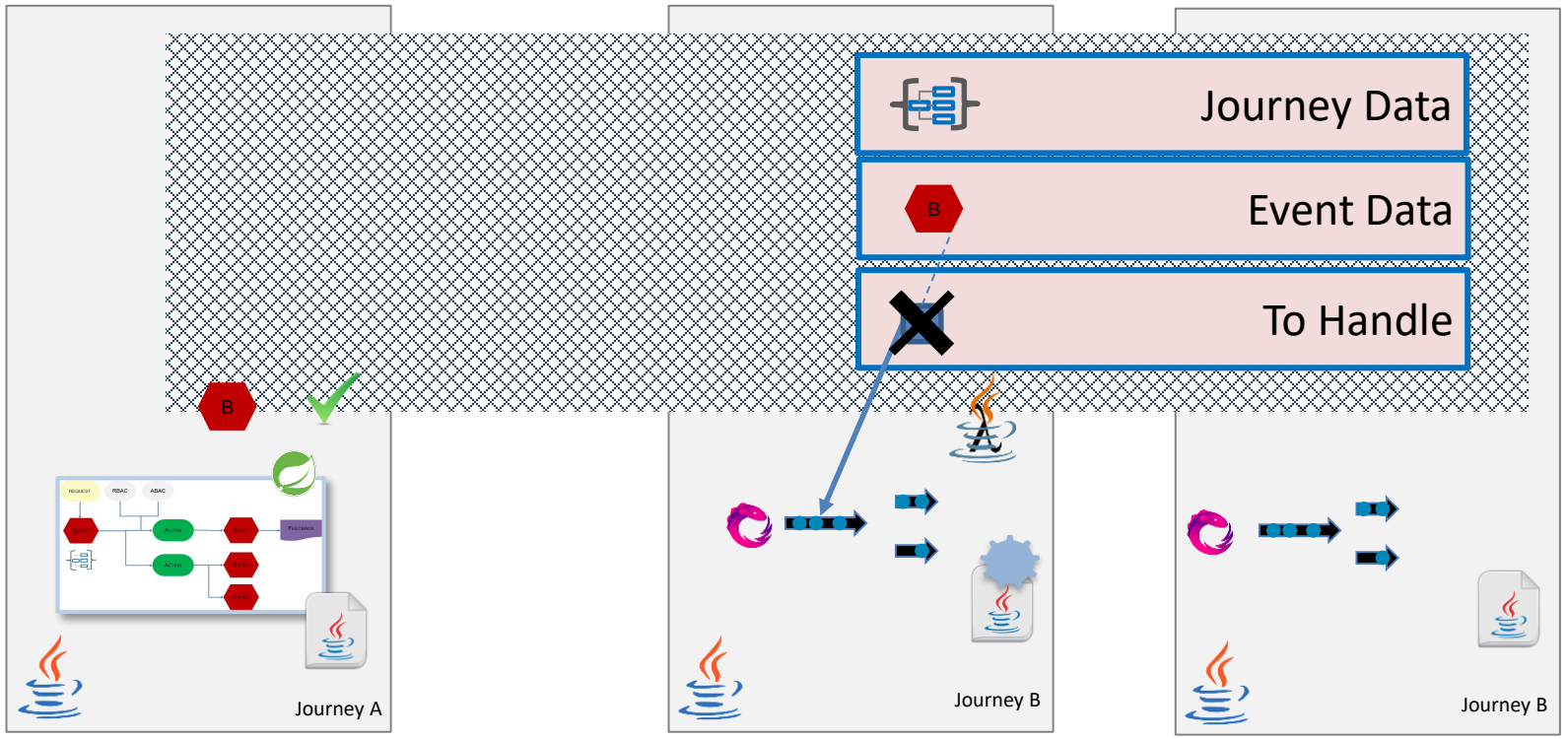
Business Robustness



Exactly once processing (EOP) of events

- Cannot be lost (RPO=0)
- MUST be handled exactly once without human intervention in good and bad weather conditions (RTO=0)
  - Nodes going down
  - Nodes being added
  - Cold restart
  - Technical errors in the environment
- Events for different journey instances can/should be handled in parallel
- Concurrent events for the same journey instance must be handled sequentially (avoid optimistic locking)

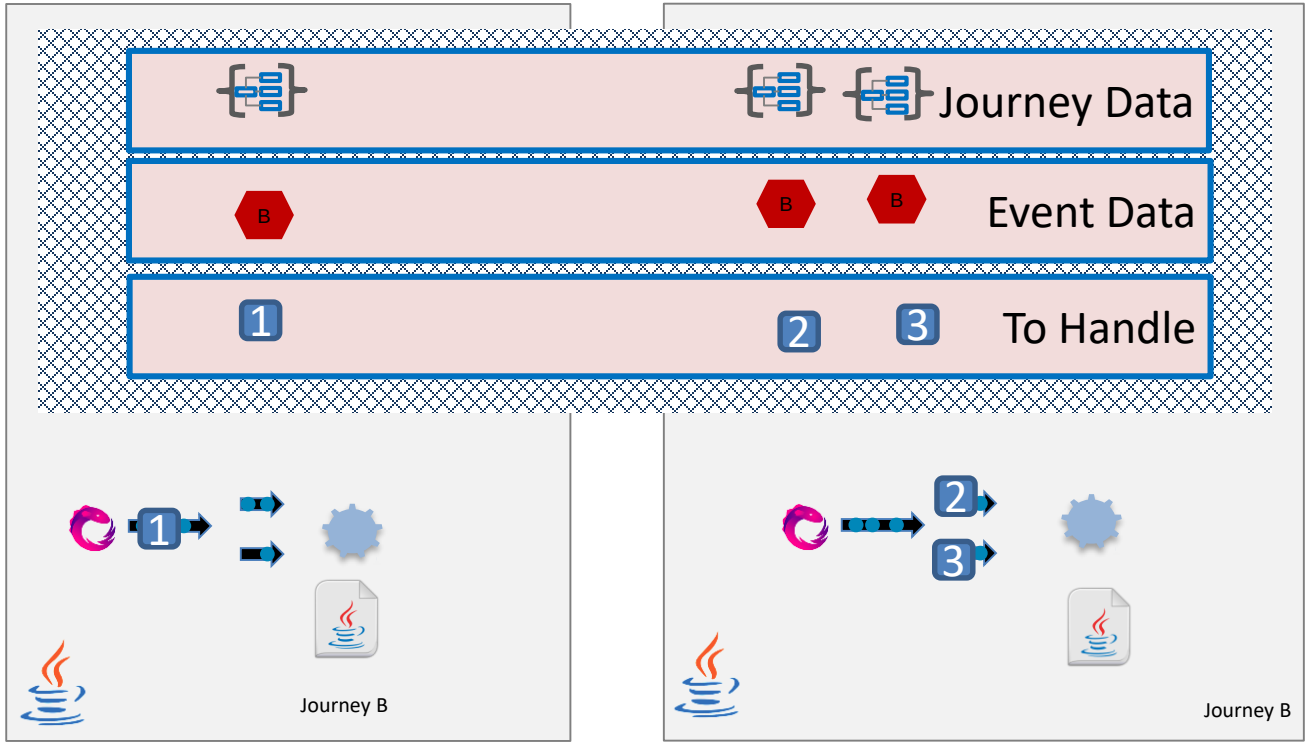
EOP in a distributed grid



theglue/signing/basketsign/1.3

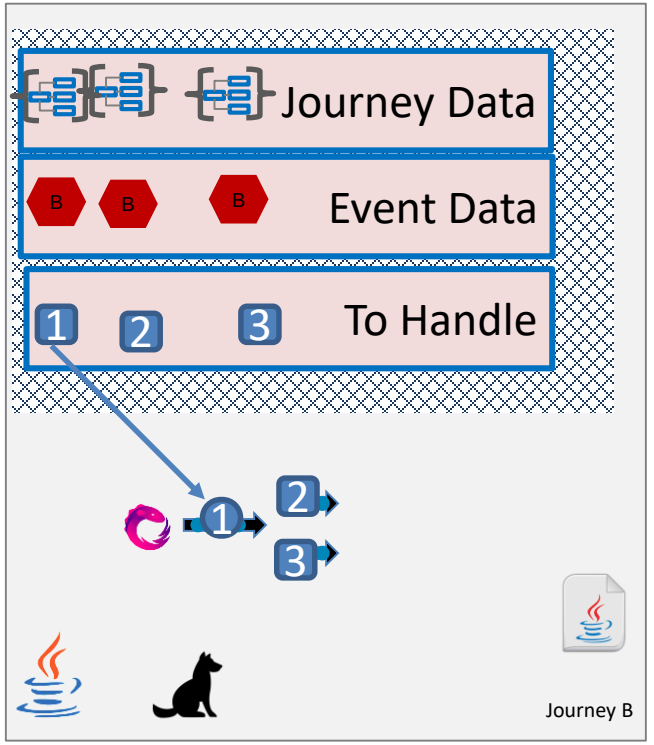
mybank/payment/sct/2.0

EOP in a distributed grid : node stops

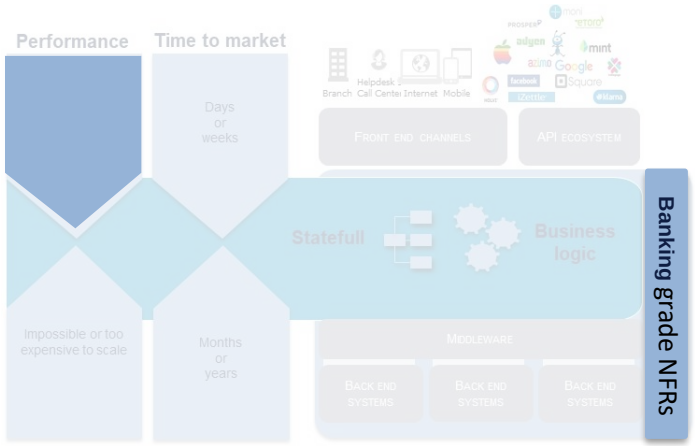


mybank/payment/sct/2.0

EOP in a distributed grid : node stops



mybank/payment/sct/2.0



NFR : SCALEABILITY AND PERFORMANCE TUNING

Scaling and performance tuning

It is extremely difficult to translate business throughput requirements into sizing requirements of the underlying technical components

Scalability and performance tuning

Journey node contains all the resources needed to drive an instance through its life

- Code + data
- Memory
- Processing threads

Maximum throughput of a journey node can be tested and tuned on the target hardware

Nodes can be added as the required throughput increases

Simple scaling model based business parameters, not technical parameters

Scaleability caveats

Data is heavy, so leave it in place as much as possible

- Correct routing and collocation
- Journey data has no backup partitions

Grid configuration changes (adding, removing nodes) leads to rebalancing of partitions

- In case of rebalancing, journey data partitions are redistributed but are left empty. Entries are fetched from the cache store when used.
- Event caches are kept small (limited to not yet handled events)

Summary

Using the combination of

- Event based service modelling
- In memory data and processing grid
- Docker based deployment

You can build a dynamically reconfigurable statefull microservice architecture which allows to

- Move very quickly from business idea to roll out in a production environment, and
- Roll out new versions in a controlled way

While adhering to the highest non functional requirementsto

- Exactly once processing
- Scalability

A great team!

Benjamin, Sven, Bart, Lenne, Jonathan, Felix, Mike, Bart, Lieven, Dieter, Ward, Ilse, Emilian, George, Irina, Cristi, Fox, Dan, Iustina, Pieter, Stefan, Robin, Ellen, Joris, Sam, Jo, Jens, Valentina, Paul, Thomas

Thank you for a hell of a journey over the last 18 months!

And thank you all for staying throughout this talk!