Fineract with Advanced Batch and Event frameworks

Presented by Istvan Molnar
BaaSFlow Inc.

FinTech Track

COMMUNITY
THE ASF CONFERENCE
CODE

Halifax, NS
October 6-10, 2023

Istvan Molnar

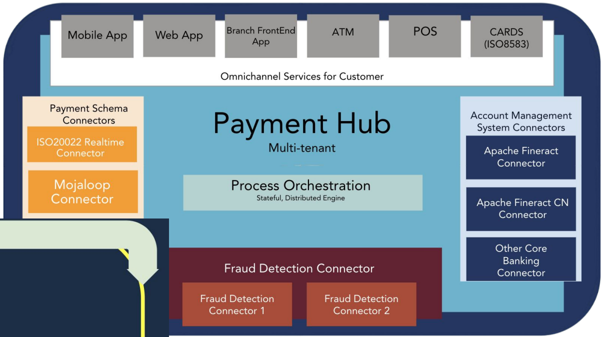
Banking, Payments, eCommerce

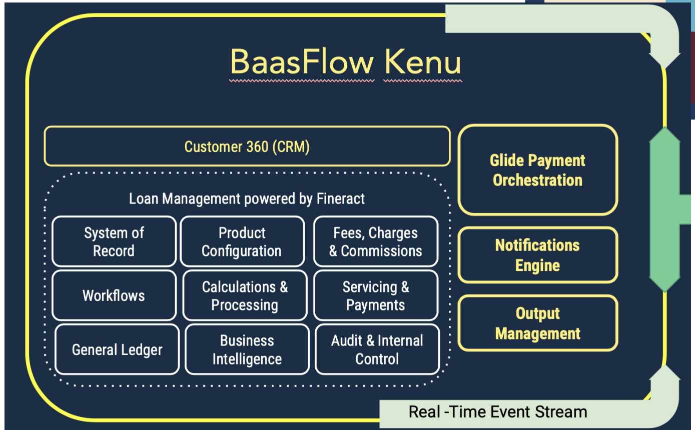
- https://baasflow.com/

BaaSFlow

consulting PIVOT DIGITAL Mifos®

Apache Fineract


Mobile App
Web App
Branch FrontEnd App
ATM
POS
CARDS (ISO8583)
Omnichannel Services for Customer
Payment Schema Connectors
ISO20022 Realtime Connector
Mojaloop Connector
Payment Hub
Multi-tenant
Process Orchestration
Stateful, Distributed Engine
Account Management System Connectors
Apache Fineract Connector
Apache Fineract CN Connector
Other Core Banking Connector
Fraud Detection Connector
Fraud Detection Connector 1
Fraud Detection Connector 2

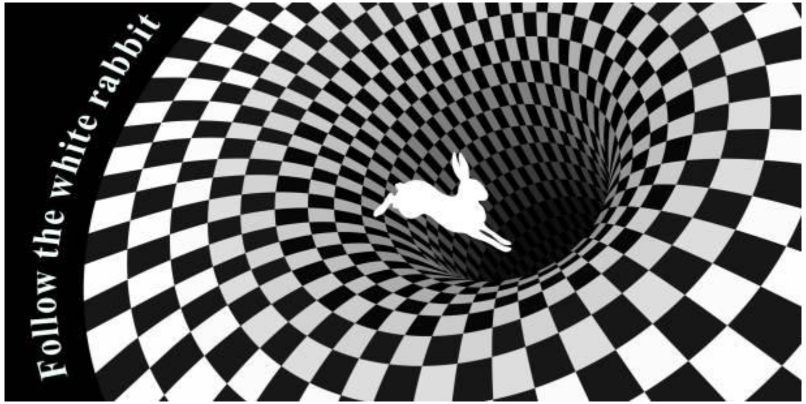


Advanced Capabilities

Context

Integration of Spring Batch

Event Data Streams



By John Tenniel - This file has been provided by the British Library from its digital collections.

Context

- The origins: Apache Fineract serving smaller communities with a single node, single deployments, even serving multiple tenants on that node
- The target: Serve only specific product (e.g. loan) for a single tenant (e.g. country) across a large set of kubernetes nodes, running the various specified containers across multiple nodes (write, read, batch, …) to serve huge write throughput (e.g. account creation), while serving even bigger read throughput from read replicas (e.g. retrieve transaction details), while executing **batch processes on many nodes parallel** on huge account base
  Meanwhile keep downstream systems (e.g data lake, reporting) near real-time updated with **financially reliable business events** on the ongoing changes.

Integration of Spring Batch

Batch Jobs

The jobs in Fineract can be divided into 2 categories:

- Normal batch jobs - running on a single node, a job executes its actions in a sequence
- Partitionable batch jobs - executes its actions in multiple partitions, can be distributed across multiple nodes to increase performance

The partitionable batch jobs in Fineract are implemented using Spring Batch.

The automatic scheduling can be done by the embedded Quartz Scheduler, but the platform supports triggering batch jobs via APIs by an external scheduler, such as Apache Airflow.

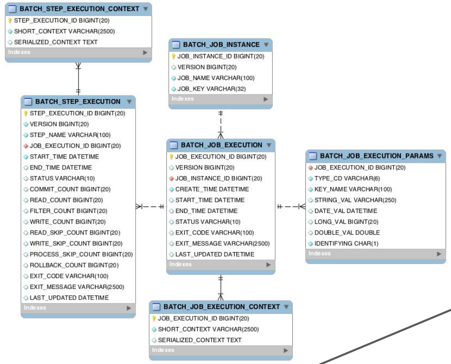
Batch job execution with partitioning

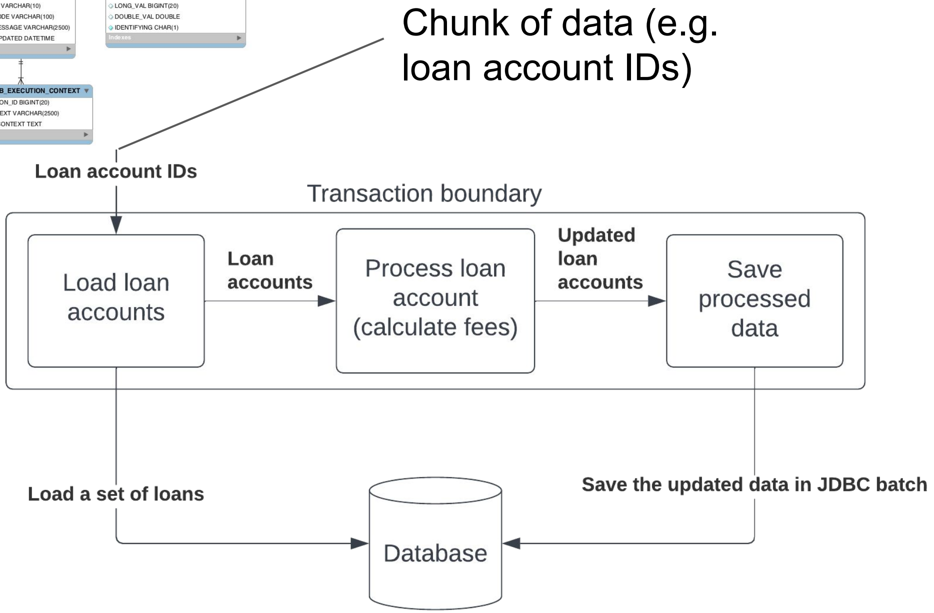
State management

- State management for the batch jobs is done by the Spring Batch provided state management.

Chunk oriented processing

- In order to save resources when starting/ committing/ rollbacking transactions for every single processed item, chunking is used.
- The transaction boundaries can be specified for a single processed chunk instead of a single item processing.
- The processing also benefit from JDBC batching.





Remote partitioning

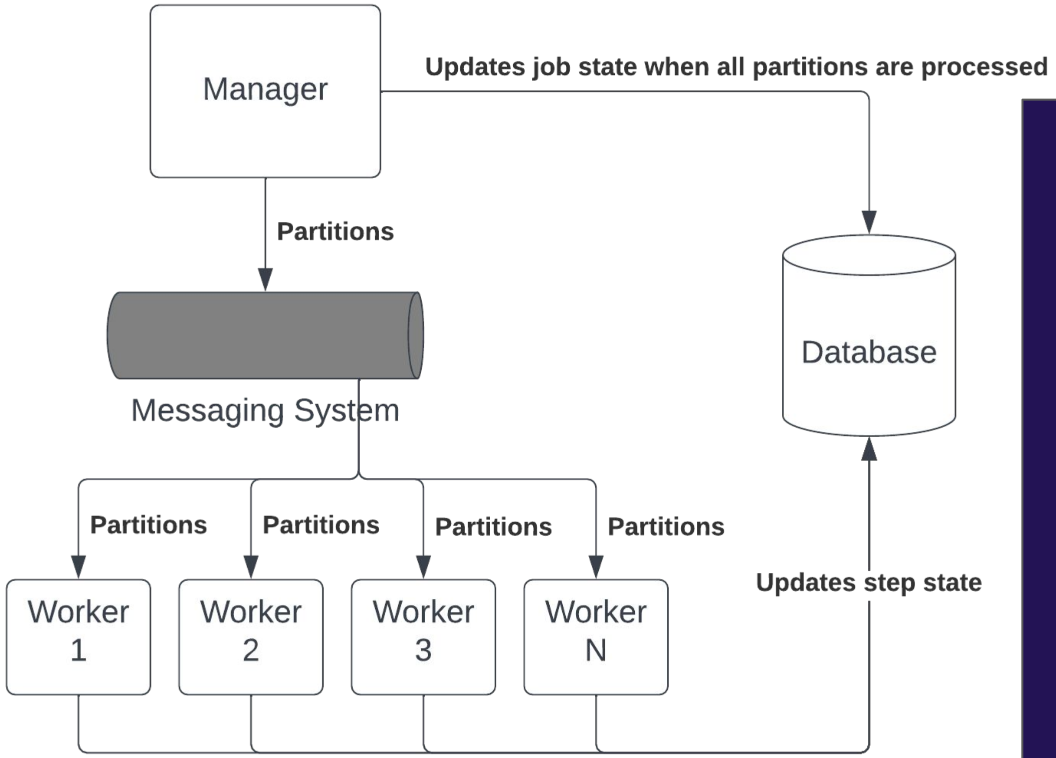
Manager node - who splits and distributes the work.

A number of worker nodes - who picks up the work from a messaging system and executes the predefined steps on the given chunk.

Chosen operation mode: When the worker does something to a partition - for example picks it up for processing - it updates the state of that partition in the database. In the meantime, the manager regularly polls the database until all partitions are processed.

Supported messaging systems: any JMS compatible message channels (e.g. ActiveMQ); Kafka support is under consideration

Fault tolerance provided by Batch framework



Business step configuration

The business steps can be configured for certain jobs. We want to allow the possibility for Fineract users to configure their very own business logic for generic jobs, like the Loan Close Of Business job where we want to do a formal "closing" of the loans at the end of the day.

All countries are different with a different set of regulations.

For example in the US, we might need the following logic for a day closing:

- Close fully repaid loan accounts
- Apply penalties
- Invoke IRS API for regulatory purposes

```
GET /fineract-provider/api/v1/jobs/{jobName}/steps?tenantIdentifier={tenantId}
HTTP 200

{ "jobName": "LOAN_CLOSE_OF_BUSINESS",
  "businessSteps": [ { "stepName": "APPLY_PENALTY_FOR_OVERDUE_LOANS", "order": 1 },          {
"stepName": "LOAN_TAGGING", "order": 2 } ] }
```

Loan account locking

To keep consistent state of loan accounts:

- Soft-locking loan accounts
  - when the Loan COB has been kicked off but workers not yet processing the chunk of loan accounts (i.e. the partition is waiting in the queue to be picked up) and during this time a real-time write request (e.g. a repayment/disbursement) comes in through the API, we simply do an "inlined" version of the Loan COB for that loan account
- Hard-locking loan accounts
  - when a worker picks up the loan account in the chunk, real-time write requests on those loan accounts will be simply rejected with an HTTP 409

Event Data Streams

Reliable event framework

Fineract is capable of generating and raising events for external consumers in a reliable way.

ACID (transactional - Atomicity, Consistency, Isolation, and Durability) guarantee
The event framework must support ACID guarantees on the business operation level.

A simple use-case:

1. A client applies to a loan on the UI
2. The loan is created on the server
3. A loan creation event is raised

What happens if step 3 fails? Shall it fail the original loan creation process?

What happens if step 2 fails but step 3 still gets executed? We’re raising an event for a loan that hasn’t been created in reality.

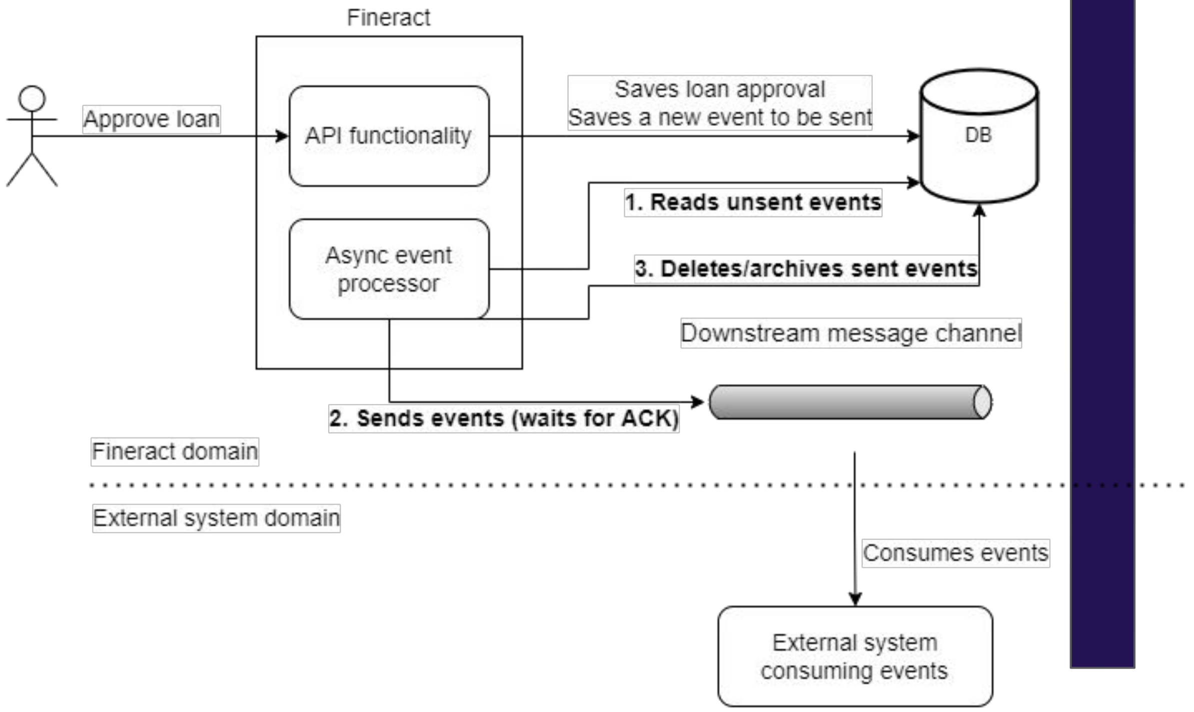
Therefore, raising an event is tied to the original business transaction to ensure the data that’s getting written into the database along with the respective events are saved in an all-or-nothing fashion.

Framework capabilities

- Messaging integration - ActiveMQ, Kafka
- Ordering guarantee - event sent in the order of generation
- Delivery guarantee - at-least-once delivery guarantee
- Reliability and fault-tolerance - handling the problems with the messaging platform
- Selective event producing - desired events configurable through the UI and API
- Standardized format - standardized format using Avro schemas
- Extendability and customizations - easily extended with new events for additional business operations or customizing existing events
- Ability to send events in bulk - queue events until they are ready to be sent and send them as a single message instead of sending each event as a separate one
    - COB process could raise multiple events, which will be combined to a single event

Architecture

1. An event gets raised in a business operation.
2. The event data gets saved to the database - to ensure ACID guarantees.
3. An asynchronous process takes the saved events from the database and puts them onto a message channel.



Foundational Business events

The framework is built upon an existing infrastructure in Fineract; the Business Events, which are Fineract events that can be raised at any place in a business operation using the `BusinessEventNotifierService`.

- Callbacks can be registered when a certain type of Business Event is raised and other business operations can be done.
- Business Events are tied to the original transaction which means if any of the processing on the subscriber's side fail, the entire original transaction will be rolled back.

Event database integration

- the framework is saving all the raised events into the same relational database that Fineract is using
    - to ensure transactionality
    - to eliminate the need of costly and error prone 2PC
    - to eliminate dependency on messaging platform availability

Event schemas

For serializing events, Fineract is using Apache Avro. Reasons:

- More compact storage since Avro is a binary format
- The Avro schemas are published with Fineract as a separate JAR so event consumers can directly map the events into POJOs

There are 3 different levels of Avro schemas used in Fineract for the Reliable event framework:

- Standard event schema - for the regular business event data
- Event message schema - wrapper around the standard event schema with extra metadata for the event consumers
- Bulk event schema - multiple events are intended to be sent together

Raising events

The key component to interact with is the `BusinessEventNotifierService#notifyPostBusinessEvent` method.

An instance of a BusinessEvent interface is needed, that’s going to be the event. There are many of them available already in the Fineract codebase and additional ones can be created.

```
@Override public CommandProcessingResult createClient(final JsonCommand command)
{  ...

businessEventNotifierService.notifyPostBusinessEvent(new          ClientCreateBusinessEvent
(newClient));
   ...
   return ...;
}
```

Example event message content

AVRO message represented as JSON:

```
{
  "id": 121,
  "source": "a65d759d-04f9-4ddf-ac52-34fa5d1f5a25",
  "type": "LoanApprovedBusinessEvent",
  "category": "Loan",
  "createdAt": "2022-09-05T10:15:30",
  "tenantId": "default",
  "idempotencyKey": "abda146d-68b5-48ca-b527-16d2b7c5daef",
  "dataschema": "org.apache.fineract.avro.loan.v1.LoanAccountDataV1",
  "data": "..."
}
```

Customizations

Creating new events (that’s already given by the Business Events)

- create an implementation of the `BusinessEvent` interface and that’s it

Creating new Avro schemas

- create a new Avro schema file in the `fineract-avro-schemas` project under the respective bounded context folder

Customizing what data gets serialized for existing events

- override the default behavior by registering a new custom serializer

Go to the next level

Utilising the powerful

- partitionable batch jobs
    - we can handle huge volume of active accounts without worrying about running out of available time to complete Close Of Business day jobs, or generating statements
- reliable event framework
    - feed datalakes, reporting systems, caches, event processors, notifications, financial crime monitoring systems, … near real-time with huge volume of events in a reliable manner, guaranteeing at least once delivery in a transactionally secured manner

Thank you

Istvan Molnar
istvan.molnar@baasflow.com

https://fineract.apache.org/docs/current/#_batch_execution_and_jobs
https://fineract.apache.org/docs/current/#_reliable_event_framework

To learn more about
BaaSFlow and our upstream
contributions.

www.baasflow.com