The Swiss Interbank Clearing System

Fast and Always On

Stefan Ferstl

2024-12-03

What do we do?

SIX = Swiss Infrastructure and Exchange

I work here...

SCHWEIZERISCHE NATIONALBANK
BANQUE NATIONALE SUISSE
BANCA NAZIONALE SVIZZERA
BANCA NAZIUNALA SVIZRA
SWISS NATIONAL BANK

- We are developing and operating the “Swiss Interbank Clearing” (SIC) System on behalf of the Swiss National Bank
- The SIC system is Switzerland’s central payment system for Swiss banks and other financial market participants
- Processing of large-value transactions as well as retail payments (bank transfers, direct debit, etc.)
- Some numbers:
  - ~ 970 Million transactions in 2023, ~90% retail transactions
  - ~ 57 Trillion (10^{12}) CHF turnover in 2023, ~90% large-value transactions

https://www.snb.ch/en/the-snb/mandates-goals/payment-transactions/swiss-interbank-clearing

SIX

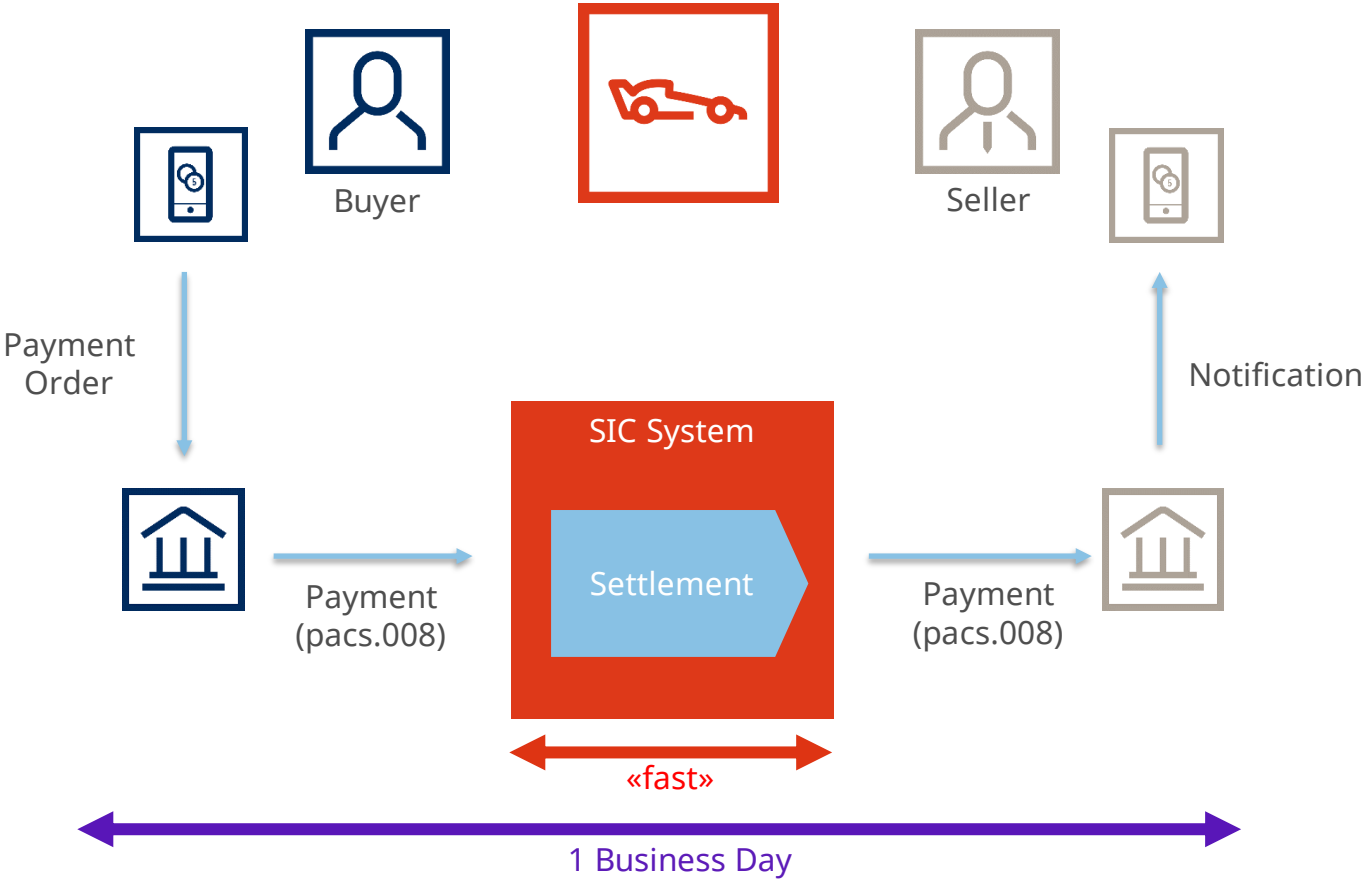
History of the SIC System

- The SIC system was initially launched in June 1987

- The SIC System is currently running in its 4th generation (SIC4)
  - Launched in 2017 as replacement for the former Mainframe-based System
  - Database-centric Java Application(s)

- For the introduction of “Instant Payments” we developed the 5th generation of the SIC System (SIC5)
  - Based on Aeron Cluster
  - Launched in November 2023 for “friends and family”. Market launch took place in August 2024.
  - Full replacement of the SIC4 System by the end of 2026

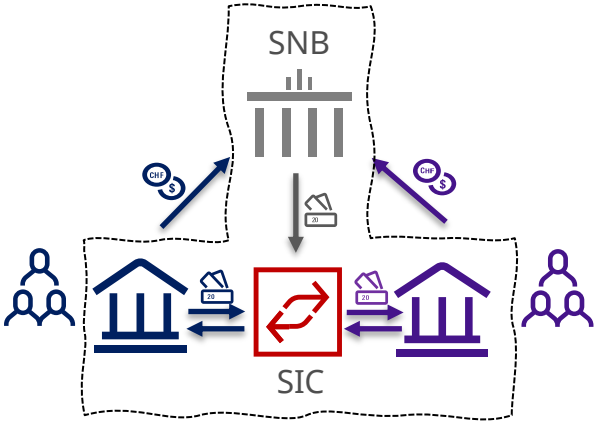
Interbank Clearing – How does it Work? (Retail)



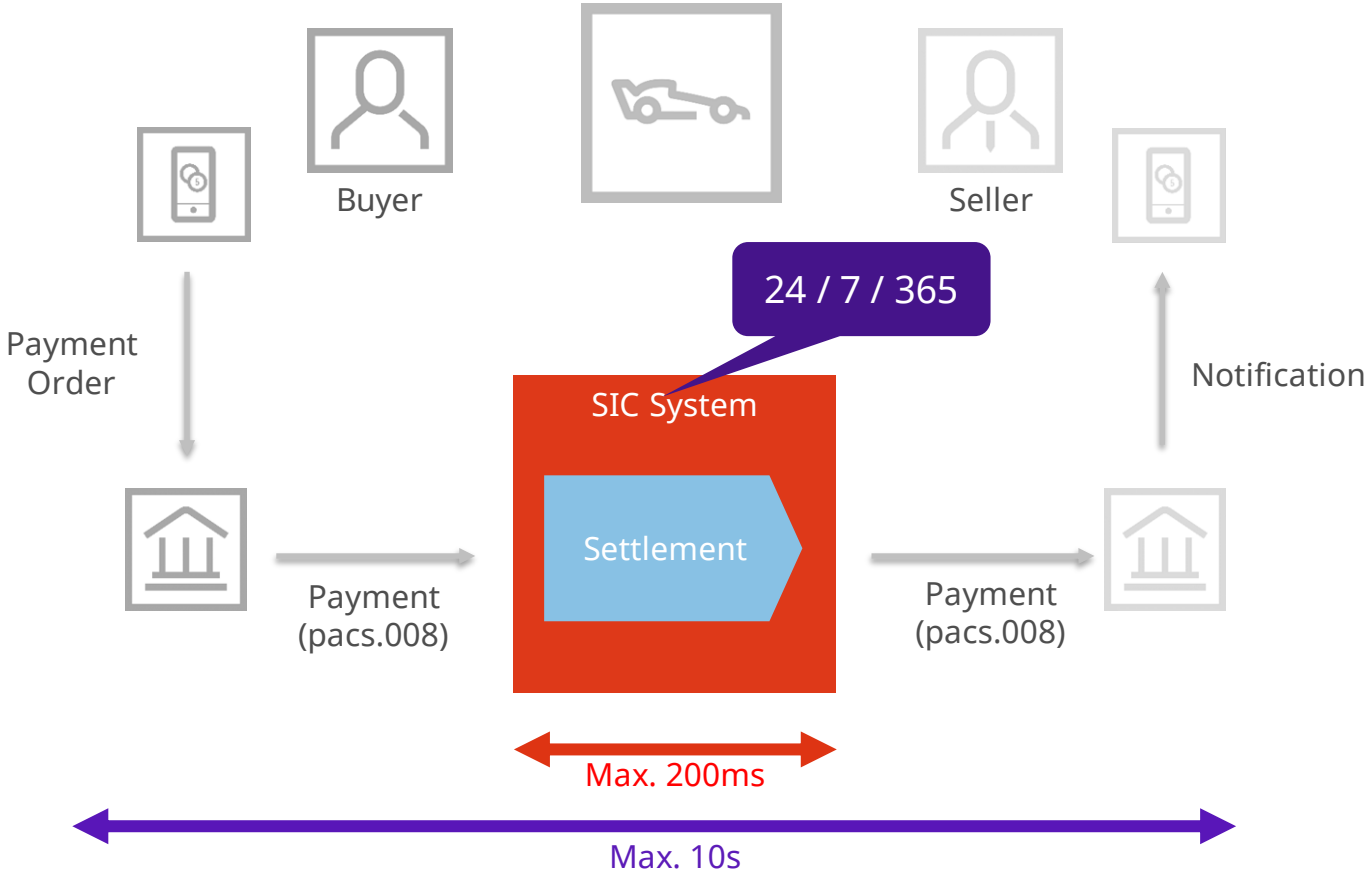
Interbank Settlement – What is special about it?

Central Bank Money

- Transaction Settlement is done in Central Bank Money
  - The SIC System keeps one settlement account for each bank
  - The settlement accounts are provisioned with Central Bank money by the Swiss National Bank
- Central Bank Money can neither be created, nor destroyed
  - Settlement can only occur when the amount is fully covered on the debited account (liquidity control)
  - Settlement is done irrevocably and final for each individual payment
  - Transaction data must not be lost under any circumstances



Instant Payments – How does it Work? (Retail only)



Main Requirements of the 5th Generation SIC System

Operate 24 / 7 / 365

never turn it off again

Guaranteed processing time

max. 10s end-to-end, 200ms within SIC

Capable of processing 35 million payments per day

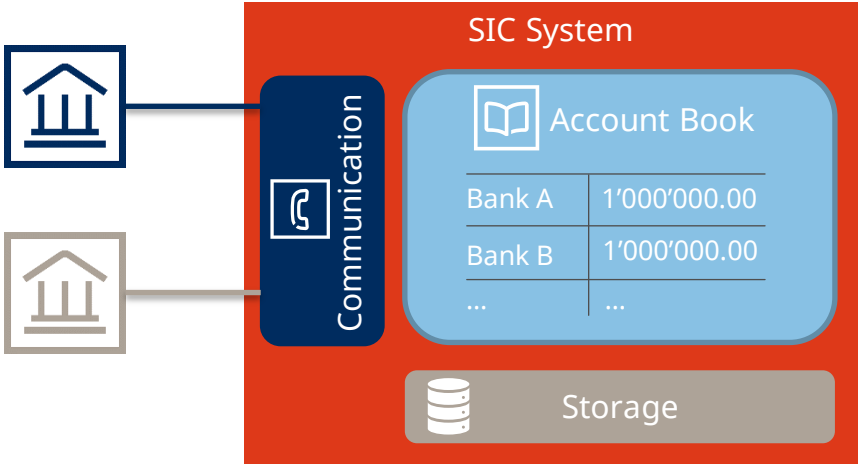
Peak at ~2000 payments/second over 1h

Never ever lose any data

What do we need

Building Blocks of the SIC System

What components do we need for payment processing?



Account Book
- Bookkeeping for all participants
- Liquidity control
- Settlement logic

Communication & Security
- Access to the system for all participants
- Participants send and receive payment messages
- Validation / Authorization
- Security (Transport Encryption, Signatures)

Storage
- Evidence for all processed transactions
- Further backoffice processing

Running the System

Operate 24 / 7 / 365
never turn it off again

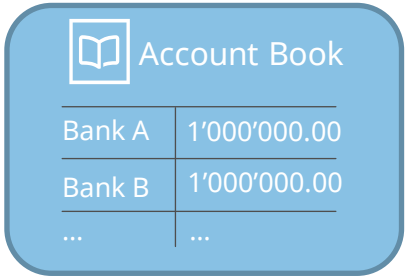
Faults
System components may fail for different reasons
→ Hardware, Network, etc.

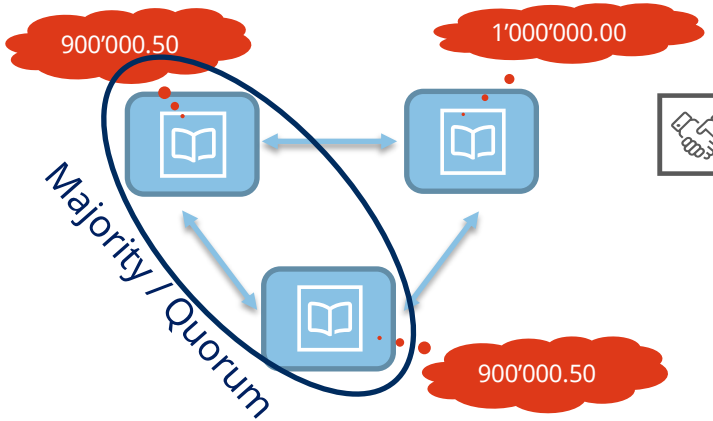
Maintenance
Components need to be taken out of service for maintenance (reboot, replacement, upgrade, etc.)

Solution: Redundancy
- **All** system components need to be redundant (hardware and software components)
- During maintenance or failures, the remaining components keep the system running

Global State

Redundancy is not that simple





Global State

- There is <u>exactly one</u> Account Book
- Each account balance must be well-known at any time
- Still, the account book cannot be a single point of failure
  We still need redundancy!

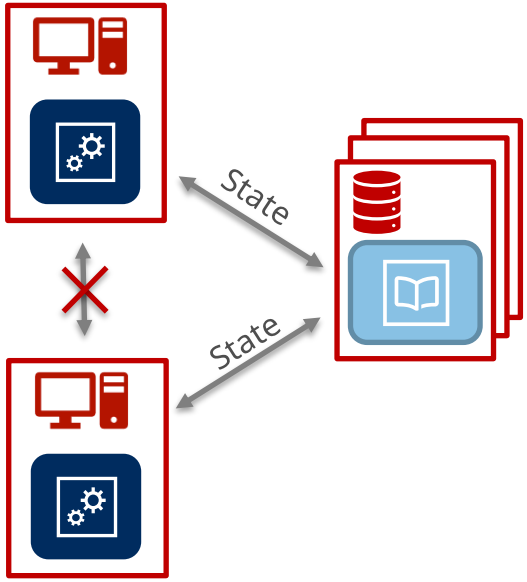
Coordination and Consensus

- When keeping global state redundant, there needs to be some sort of coordination
- Double redundancy does not work and may lead to “Split Brain” situation
- A majority of redundant components are able to agree on a global state. This is called “Clustering”

Global State

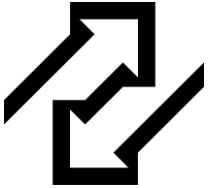
35 Mio Trx/Day
Peak at 2000 Trx/s over 1h

Let someone else deal with the Problem



- Handling of global state is most commonly delegated to some sort of storage system (Database, SAN, etc.)
- Some of these systems allow clustering and may be operated with zero downtime
- Most of these systems are not designed for running business logic
- Most commonly, applications modify the state and synchronize it back and forth
- The central storage system may become a bottleneck (roundtrips, locking, etc.)

Is there a more efficient way to deal with global state?



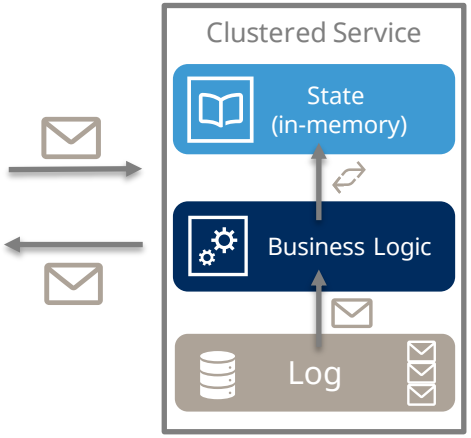
Aeron®

What is Aeron

- A high-performance UDP-based messaging framework
  - Implemented in Java (C-implementations of some components available)
  - Available on GitHub under the Apache 2.0 license: https://github.com/real-logic/aeron (Commercial features and support available)
  - Developed by people who really know their stuff
- Uses Simple Binary Encoding (SBE) for message serialization
- Three variations building on top of each other
  - **Aeron Transport**: High-performance messaging
  - **Aeron Archive**: Recording of message streams for later use or real-time replay
  - **Aeron Cluster**: State replication for fault-tolerant services ← this

Aeron Cluster

The Clustered Service



- The main component (from a developer's perspective) is the "Clustered Service"
- Aeron Cluster maintains a sequential log of all received messages
- Application-specific business logic acts on these messages and manipulates its internal in-memory state, e.g. the Account Book

→State can be re-created by reading the log

→State can be also be stored regularly in the log (Snapshotting)

Faster re-creation of state by reading the latest Snapshot and all messages since then

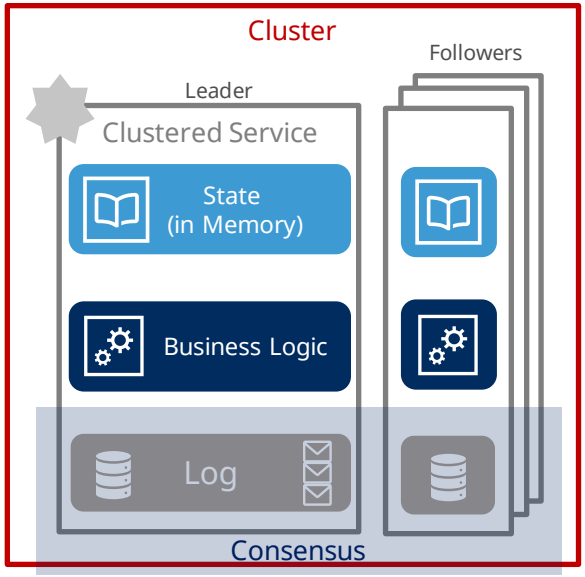
Aeron Cluster

How does it look in Java

```
public class AccountClusteredService implements ClusteredService {

   private final AccountBook accountBook = new AccountBook();

   void onStart(snapshotImage) {
     // Deserialize the snapshot image into the global state
     this.accountBook.load(snapshotImage);
   }

   void onSessionMessage(clientSession, timestamp, buffer, offset, length, header) {
      // Execute business logic
      var payment = Payment.read(buffer, offset, length);
      var result = this.accountBook.settle(payment, timestamp);
      result.sendAcknowledgment(clientSession);
   }

   void onTakeSnapshot(snapshotPublication) {
     // Serialize the global state
     this.accountBook.store(snapshotPublication);
   }

   …

}
```

Aeron Cluster

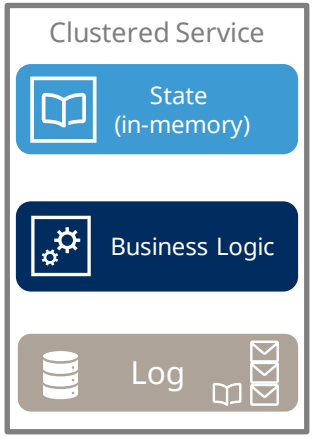
Raft Consensus



- Multiple instances of a Clustered Service form a “Custer”
- The cluster uses the “Raft” consensus algorithm to
  - Agree on the messages in the logs
  - Elect a “Leader” which coordinates the other instances (“Followers”)
- Consensus is reached when a majority of nodes contain the same messages in the same order
- As long as a majority of cluster nodes is available, the cluster keeps working
  → Fault tolerance
  → Rolling upgrades

Aeron Cluster

Business Logic and State



Business Logic within the Clustered Service must be 100% deterministic

- Across all instances of the clustered service
- During and after rolling upgrades
- Changes in behavior must be toggled and recorded in the log
- No undeterministic operations (`System.currentTimeMillis()`, `UUID.randomUUID()`, etc.)
- Use deterministic data structures for internal state, e.g. Agrona collections
- No multi-threading

Why is this better?

Performance

- Redundant global state is achieved directly within the application
- No unnecessary network roundtrips to synchronize state to a storage system
- Performance is only limited by:
  - Write speed of the log
  - Network latency / bandwidth
- State is kept and manipulated in memory only
- The log is only read during startup

Back to the SIC System

The 5th Generation SIC System

General Design Principles

- Separate stateless from stateful business logic
- Stateful business logic is running as Aeron Clustered Service
- Stateless business logic can be deployed redundantly as often as needed

The 5th Generation SIC System

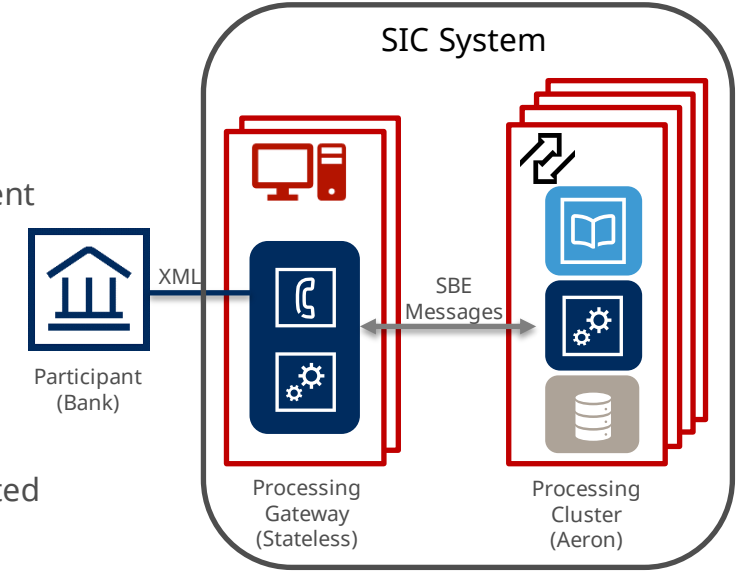
The new Architecture with Aeron Cluster

Processing Gateway (Stateless)

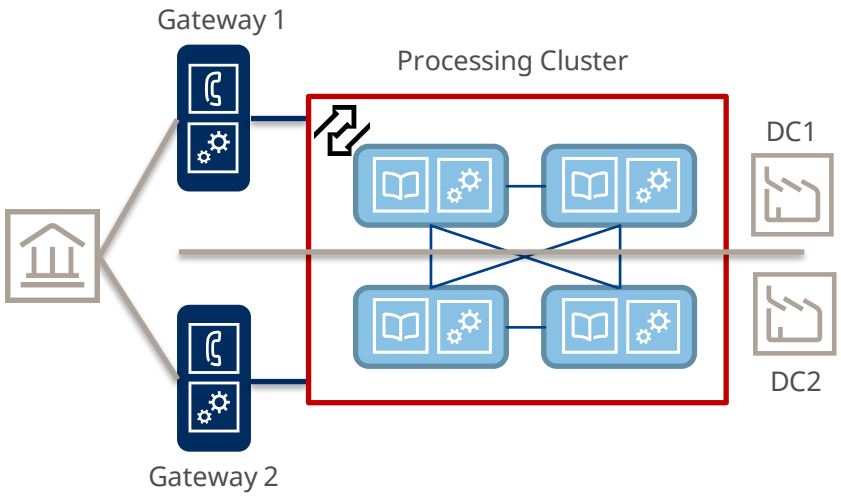
- Secure communication with the participants
- Stateless business logic for payment validation and authorization
- Valid payments are forwarded to the Processing Cluster for settlement

Processing Cluster (Stateful)

- Settlement logic is running in an Aeron Clustered Service
- Payments are processed as input events of the Clustered Service
- State (account balances) is held transiently in memory and updated by the Clustered Service
- Regular (daily) Snapshots



Deploying to Production



The SIC System in Production

- The SIC5 system is distributed over our two data centers
- Each DC provides one communication endpoint for the system participants
  - Banks are encouraged to maintain two connections to the system
- All state (account books, transaction information) is maintained in a **4-node Aeron Cluster**
  - No majority in one DC
  - Data is geographically distributed

Roundup

Experiences with Aeron Cluster

Experiences

- 🙂 Our performance requirements were reached without any tuning or optimizations
- 🙂 Aeron Cluster works very well with quite large messages and distribution across data centers
- 😐 Steep learning curve(s)
  - 24 / 7 / 365 operating model is <u>very very very</u> hard
  - Aeron APIs and programming model require familiarization among developers
- 😐 Fewer insights into the system (no “SELECT * FROM payment WHERE …”)

Lessons Learned

- 🤔 Upgrades and system changes have to be planned and tested thoroughly
  → <u>All</u> behavioral changes in the cluster need to be feature-toggled
- 🤔 Automated testing for rolling software updates is crucial

Some Words of Advice

Using a database or other storage solution is perfectly fine most of the times

Don’t do 24 / 7 / 365 unless it is absolutely necessary

Thank you!

Stefan.Ferstl@six-group.com

Disclaimer