In-Memory Computing SUMMIT EUROPE 2019

Healthcare Outbounds: Unbounded Scalability

Robert Stephens, Director of Revenue Cycle Development

Sujoy Acharya, Director & Principal Engineer

Cerner

TalkConfiguration<Integer, String> agenda

agenda.put(1, "**The Problem Statement**");

agenda.put(2, "**v1. High level Design**");

agenda.put(3, "**SP Results**");

agenda.put(4, "**v2. Redesigning the system with Apache Ignite**");

agenda.put(5, "**v3. Refactoring**");

agenda.put(6, "**Q/A**");

Outbound

The Problem Statement

Performance Bottleneck

se
nto

ıble to

Background

Healthcare revenue cycle product used in United States and global market

Includes Scheduling, Registration, Patient Accounting

The registration ADT (Admission, Discharge, Transfer) transaction data needs to be sent out to other systems in HL7 format

Java platform with horizontal scaling

Challenged to scale the outbound interface for larger clients

Options

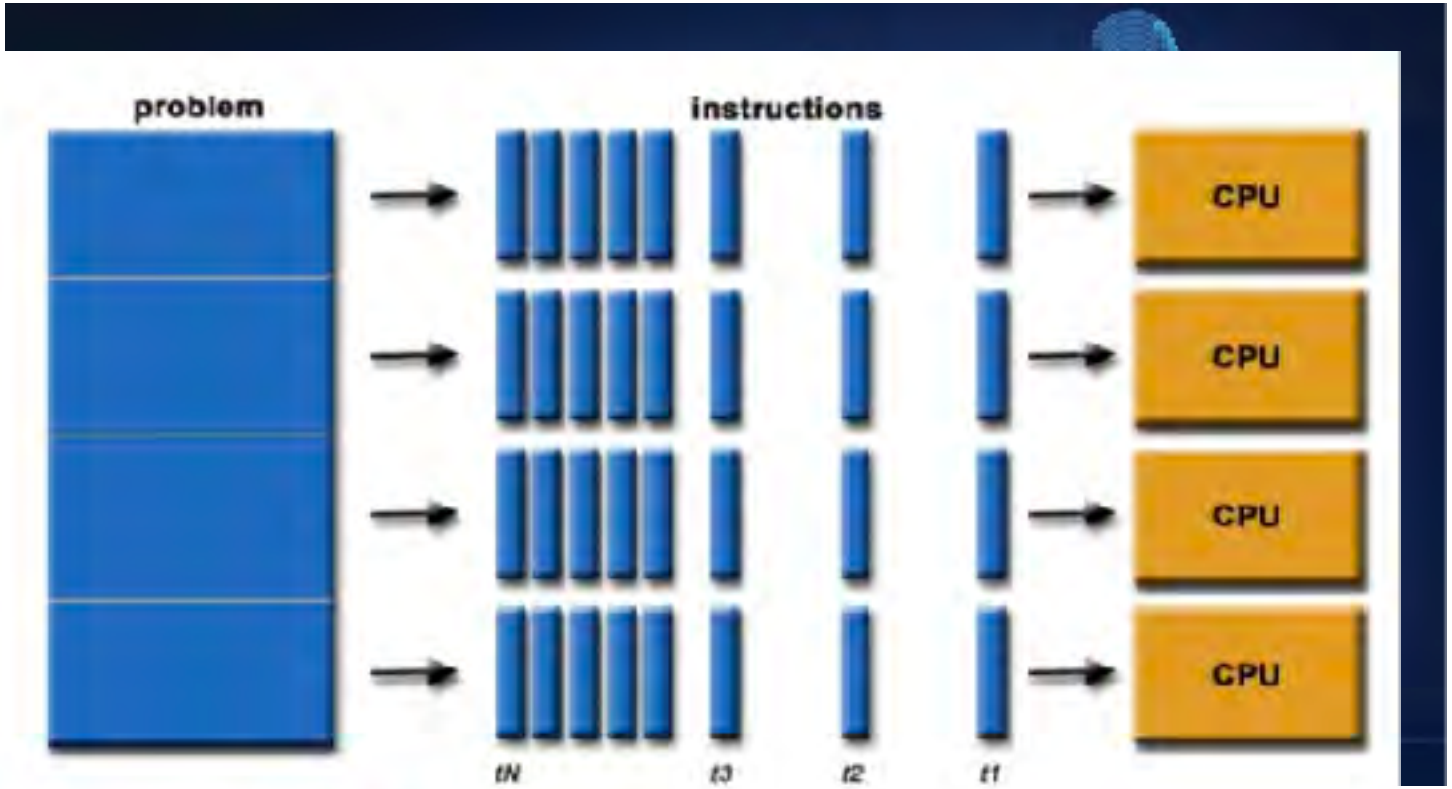
Tune code

Scale vertical (Bigger Server

Execute transformation soon

Process in parallel.

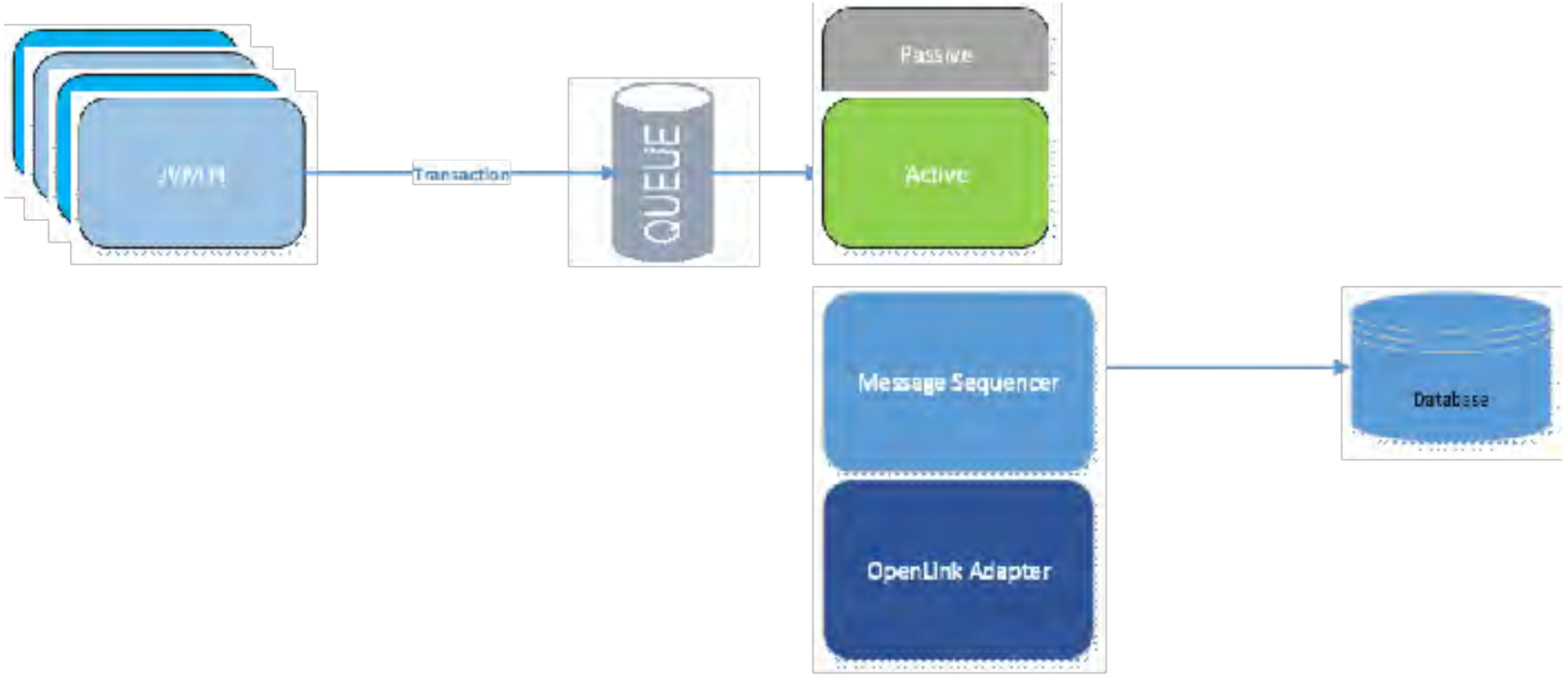
This approach seemed best



High Level Design

V1. Database Driven Design

Initial Design



Initial Design

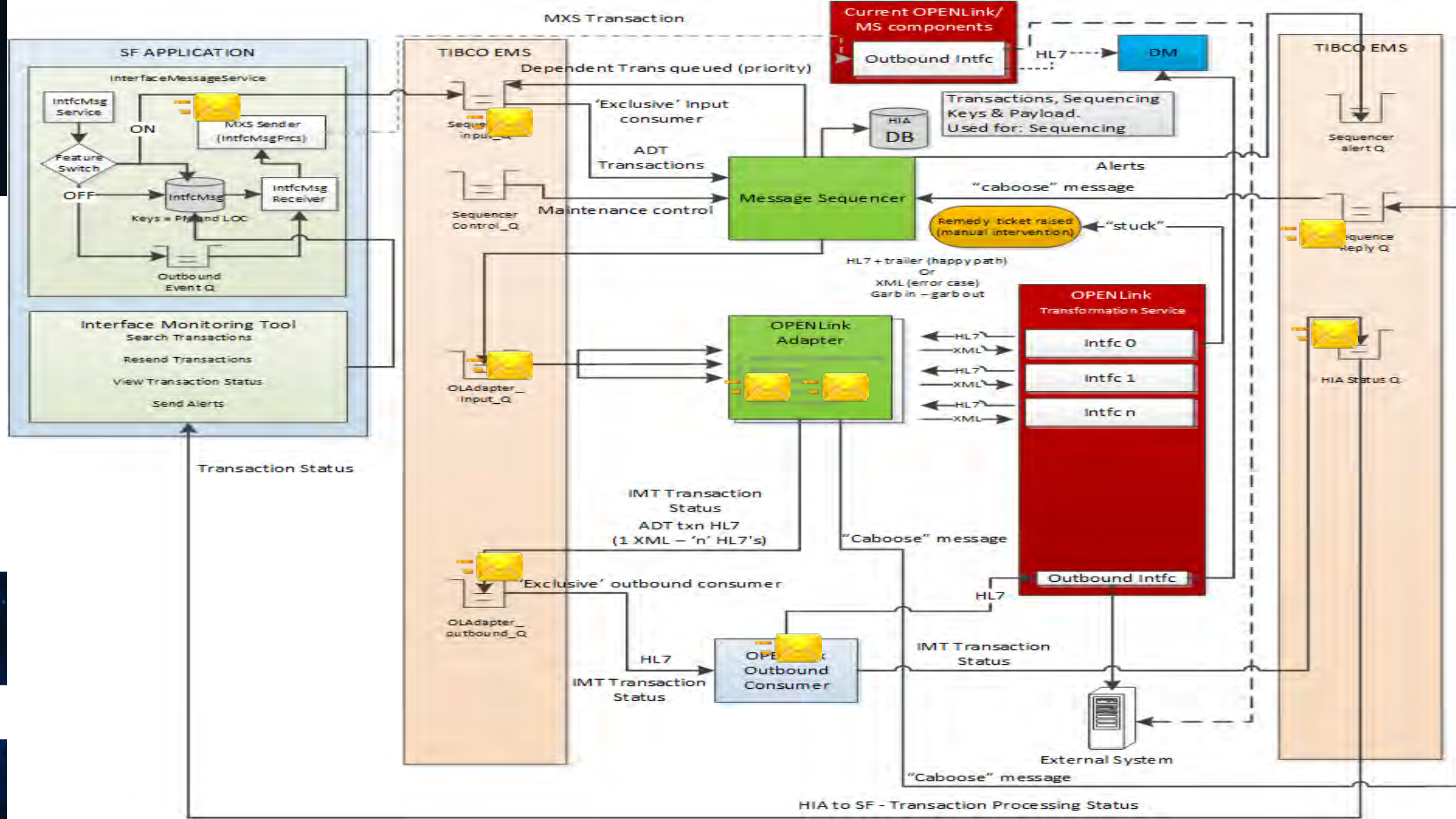
key1=bed01

key1=bed999

key1=bed01

TransactionID	Key1	...	Key12	Status
t001	bed01			Processing
t002	bed999			Processing
t003	bed01			Waiting

Seq No	TransactionID	Key
1	t001	bed01
2	t002	bed999
3	t003	bed01


MXS Transaction
SF APPLICATION
InterfaceMessageService
IntfcMsg Service
ON
MXS Sender (IntfcMsgPrcs)
Feature Switch
OFF
IntfcMsg
IntfcMsg Receiver
Keys = PN and LOC
Outbound Event Q
Interface Monitoring Tool
Search Transactions
Resend Transactions
View Transaction Status
Send Alerts
Transaction Status
TIBCO EMS
Dependent Trans queued (priority)
'Exclusive' Input consumer
ADT Transactions
Sequencer Control_Q
Maintenance control
OLAdapter_ Input_Q
IMT Transaction Status
ADT txn HL7
(1 XML – 'n' HL7's)
'Exclusive' outbound consumer
OLAdapter_ outbound_Q
HL7
IMT Transaction Status
Current OPENLink/ MS components
Outbound Intfc
HL7
DM
HIA DB
Transactions, Sequencing Keys & Payload. Used for: Sequencing
Alerts
"caboose" message
Message Sequencer
Remedy ticket raised (manual intervention)
"stuck"
HL7 + trailer (happy path)
Or
XML (error case)
Garb in – garb out
OPENLink Adapter
OPENLink Transformation Service
Intfc 0
Intfc 1
Intfc n
HL7
XML
Outbound Intfc
"Caboose" message
IMT Transaction Status
External System
"Caboose" message
HIA to SF - Transaction Processing Status
TIBCO EMS
Sequencer alert Q
HIA Status Q

Results

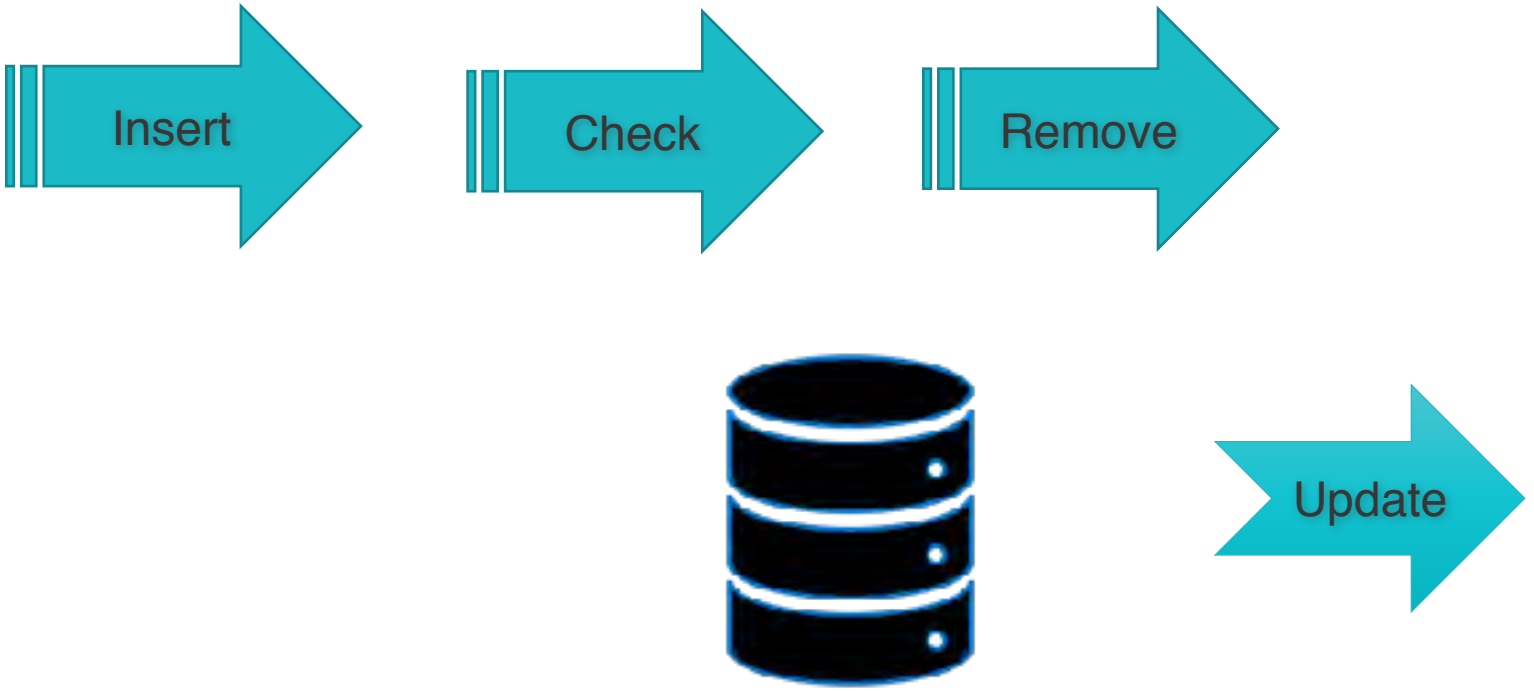
Stored Proc

Processing Rate

Re-Designing

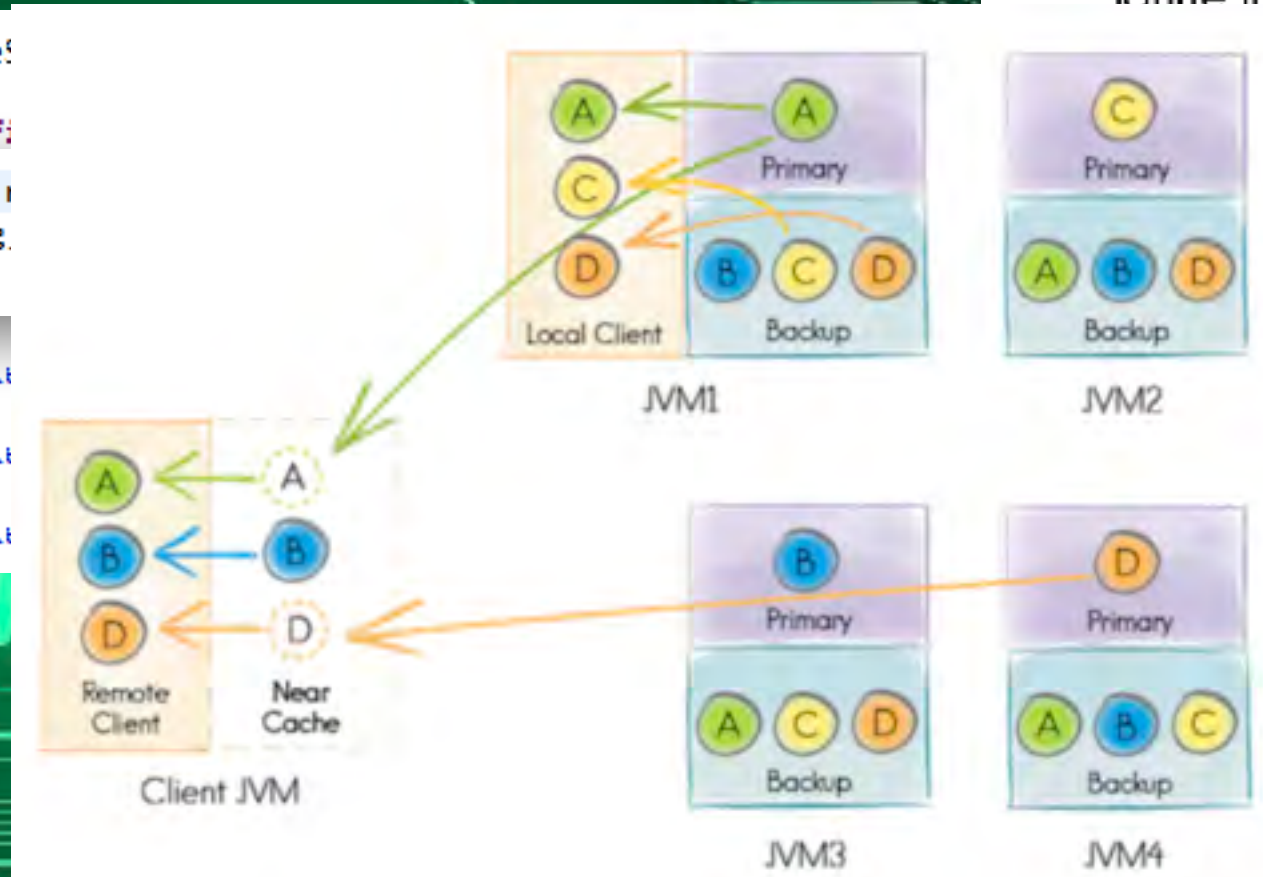
In-memory Architecture

@DataAccessPoints()



@Cacheable?

```
1  public class Message
2
3      private static fi
4
5      IgniteCache<Stri
6      IgniteCache<Long,
7
8      @QuerySqlField()
9      private String ka
0      @QuerySqlField()
1      private String ka
2      @QuerySqlField()
3      private String ka
```



new SqlQuery(Check, “Dependency")

```
String queryString = String.format("FROM MESSAGEKEY WHERE KEY in (%s)", commaSeparatedKeys);
SqlQuery<Long, MessageKey> igniteQuery = new SqlQuery<>(MessageKey.class, queryString);
igniteQuery.setLocal(true);

List<Entry<Long, MessageKey>> matchingKeys = messageKeyCache.query(igniteQuery).getAll();
boolean hasDependencyInd = matchingKeys != null && matchingKeys.size() > 0;
```

new InsertQuery(Dependency, “Status")

```
List<String> keys = newTransactionMessage.getKeysAsString();
String transactionStatus = hasDependency(keys) ? WAITING : PROCESSING;
logger.debug(String.format("Keys are %s and Status %s", keys, transactionStatus));

newTransactionMessage.setStatus(transactionStatus);

/**
 * Insert in message status and key
 */
messageStsCache.put(newTransactionMessage.getJmsId(), newTransactionMessage);

newTransactionMessage.getKeys().forEach(key -> {
    messageKeyCache.put(key.getId(), key);
});
```

new RemoveQuery(Release, “Dependant")

```
messageKeyCache.query(
        new SqlFieldsQuery("DELETE FROM MessageKey  WHERE jmsid = ?").setArgs(messageStatus.getJmsId()));
msgStsCache.query(
        new SqlFieldsQuery("DELETE FROM MessageStatus  WHERE jmsid = ?").setArgs(messageStatus.getJmsId()));

/**
 * Find oldest transaction for each key
 */
Set<String> oldestWaitingTransactionForEachKey = new HashSet<>();
messageStatus.getKeys().forEach(key -> {

    messageKeyCache.query(new SqlFieldsQuery("SELECT top 1 jmsId FROM MessageKey WHERE key = ? order by id")
            .setArgs(key.getKey())).forEach(oldTransaction -> {
                oldestWaitingTransactionForEachKey.add(oldTransaction.get(0).toString());

            });
});
```

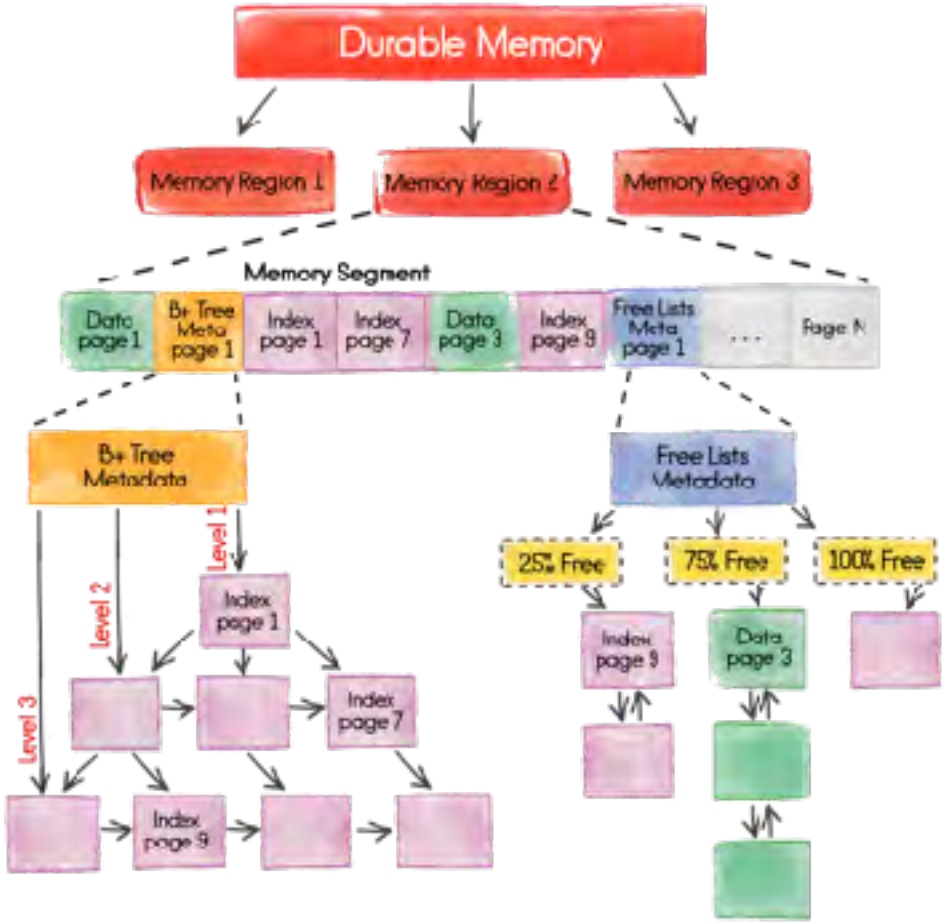
new Result(Metrics)

```
long startTime = System.nanoTime();
hasDependencies = messageStatusDao.insertMessage(msDto);
long endTime = System.nanoTime();

long timeSpentInDb = endTime - startTime;

startTime = System.nanoTime();
hasDependencies = messageStatusCacheableDao.insertMessage(msDto);
endTime = System.nanoTime();

long timeSpentInCache = endTime - startTime;

metricLogger.info(String.format(S_S_S_S_S, Operation.INSERT , msDto.getIntfcMsgObjId(),
        timeSpentInDb, timeSpentInCache, (timeSpentInDb - timeSpentInCache)));
```

Refactoring

Algorithm

Durable Memory




9000
THANK YOU!

QA

References

[1]. 50 lane traffic - https://www.citylab.com/transportation/2015/10/chinas-50-lane-traffic-jam-is-every-commuters-worst-nightmare/409639/

[2]. SQL grid - https://files.readme.io/ee4c650-SQL-Grid-Diagram_v4.png

[3]. Memory grid - https://files.readme.io/9d858ef-Durable_Memory_Diagram.png

[4]. SQL - https://apacheignite-sql.readme.io/docs/performance-and-debugging