mojaloop

PI-9 Performance Workstream

April 2020

Team (Alphabetical Order):
Coil
Confluent
CrossLake
ModusBox
Sybrin

mojaloop

mojaloop

Performance

MODUSBOX

Team (Alphabetical Order):

Bryan Schneider

James Bush

Miguel de Barros (Presenter)

Rajiv Mothilal

Roman Pietrzak

Sam Kummary

Valentin Genev

Performance – PI-8 Performance (Characterization) Report

1. Financial Transaction End-to-End

2. Financial Transaction Prepare-only

3. Financial Transaction Fulfil-only

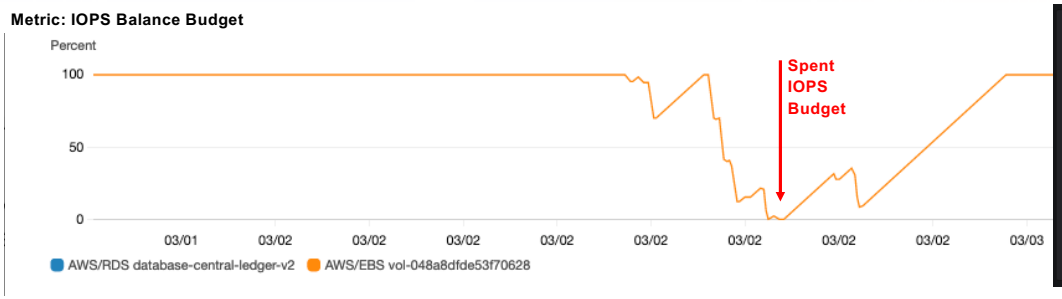
4. Individual Mojaloop Component Characterization
   a. Mojaloop Services & Handlers
   b. Mojaloop Streaming Architecture & Libraries
   c. Database

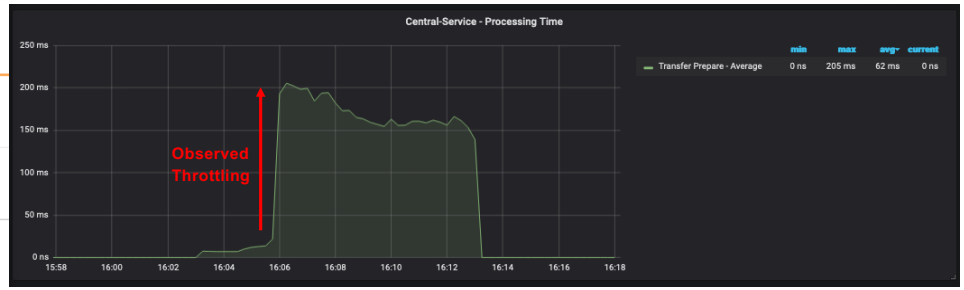
Download Report URI:

https://github.com/mojaloop/documentation-artifacts/blob/master/presentations/April%202020%20Community%20Event/Pre_Read/202003-PI8-Performance-Report-v1.0.pdf

Resolved Limiting Factors on AWS cloud-provider

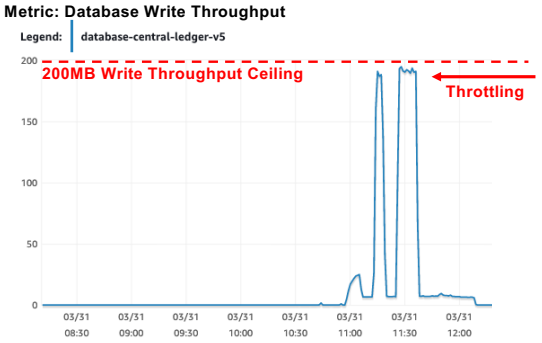
1. IOPS Budget Balance Throttling
   a) Default 300 IOPS limit for GP2 instances @ 100GB
   b) Avoid by Provisioning dedicated IOPS





2. IOPS Ceiling
   a) Ensure enough IOPS provisioned on storage
   b) Performance tests used 4k—20k IOPS (depends on expected Write Throughput)

3. Write Throughput Ceiling
   a) Ensure dedicated network throughput for attached storage matches write throughput requirements
      a) 100MB/s - db.r4.xlarge
      b) 200MB/s - db.r4.2xlarge
      c) 375MB/s - db.r4.2xlarge



Performance – Enhancement Changes

Key:
[●] Migrated to Master branch (v9.5.x)
[●] Performance branch only

1. Optimized Logging (central-services-logger)
   a) Added conditional logging to reduce unnecessary processing/parsing logic:
      - [●] ML-API-Adapter
      - [●] Central-Ledger
      - [●] Simulator
      - [●] Central-Services-Shared
      - [●] Central-Services-Stream

```
-     Logger.debug('create::payload(%s)', JSON.stringify(request.payload))
+     Logger.isDebugEnabled && Logger.debug('create::payload(%s)', JSON.stringify(request.payload))
```

2. Caching of Non-Stateful Data
   a) [●] Participants - central-ledger
   b) [●] Participant Accounts - central-ledger
   c) [●] Participant Limits - central-ledger

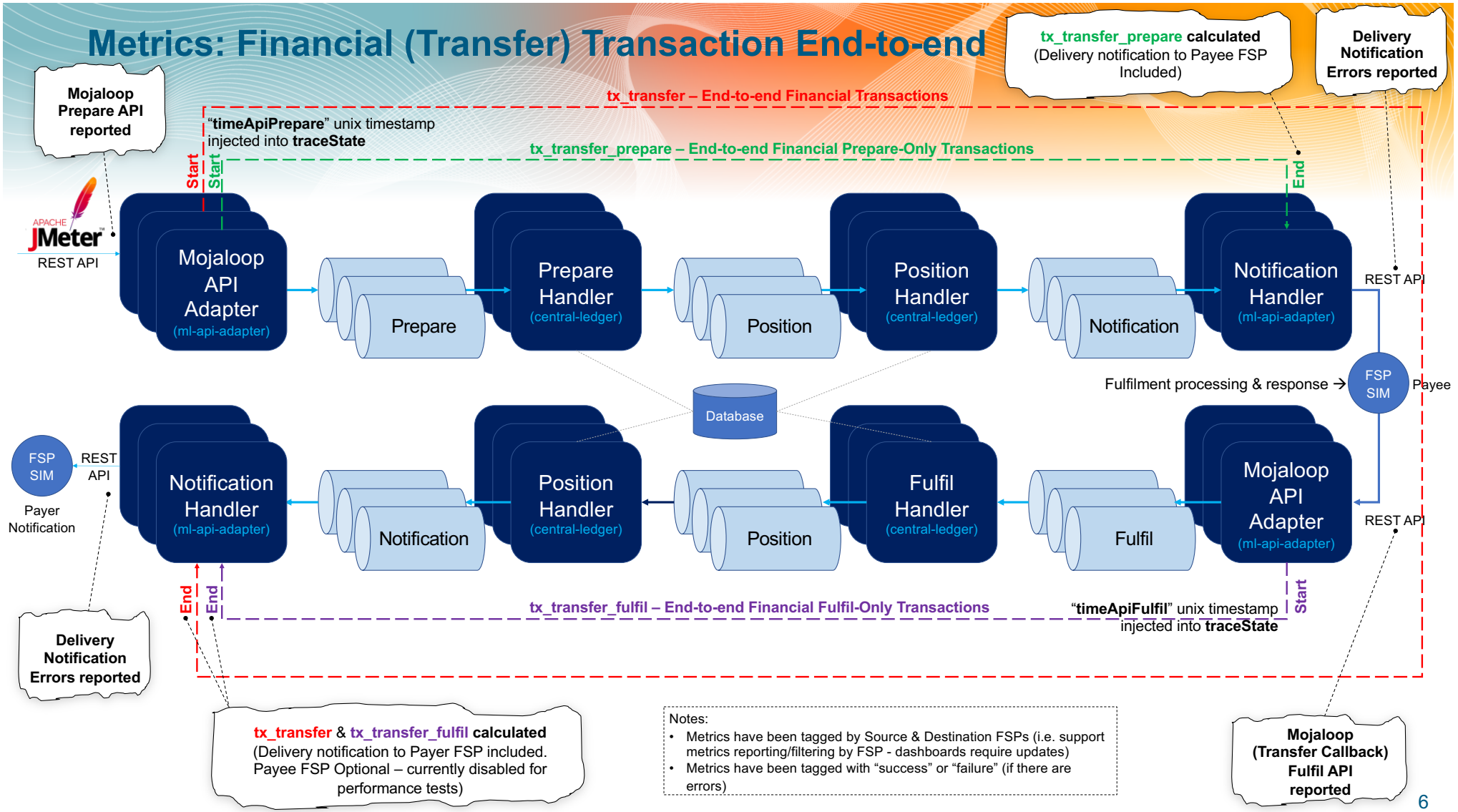
3. Optimized SQL Queries
   a) Duplicate Checks
      a) [●] Insert-only logic
   b) Position changes
      a) [●] Reduced locks by in-line calculation

4. Metrics (central-services-metrics)
   a) v9.5.x Release supports Histograms, and Summary (new)
   b) New Instrumentations
      - ML-API-Adapter - reported by Notification Handler
        - [●] Notification Delivery
        - [●] Errors for Notification Delivery
        - [●] End-to-end Financial Transactions - 'tx_transfer'
        - [●] Prepare-only Financial Transactions - 'tx_transfer_prepare'
        - [●] Fulfil-only Financial Transactions - 'tx_transfer_fulfil'
        - [●] Transaction processing time buckets
        - [●] % of Transaction longer than 1 sec
      - Central-Ledger – reported by Prepare, Position & Fulfil Handlers
        - [●] Domain (functional) logic
        - [●] SQL queries, inserts & updates
        - [●] Cache Hits / Misses

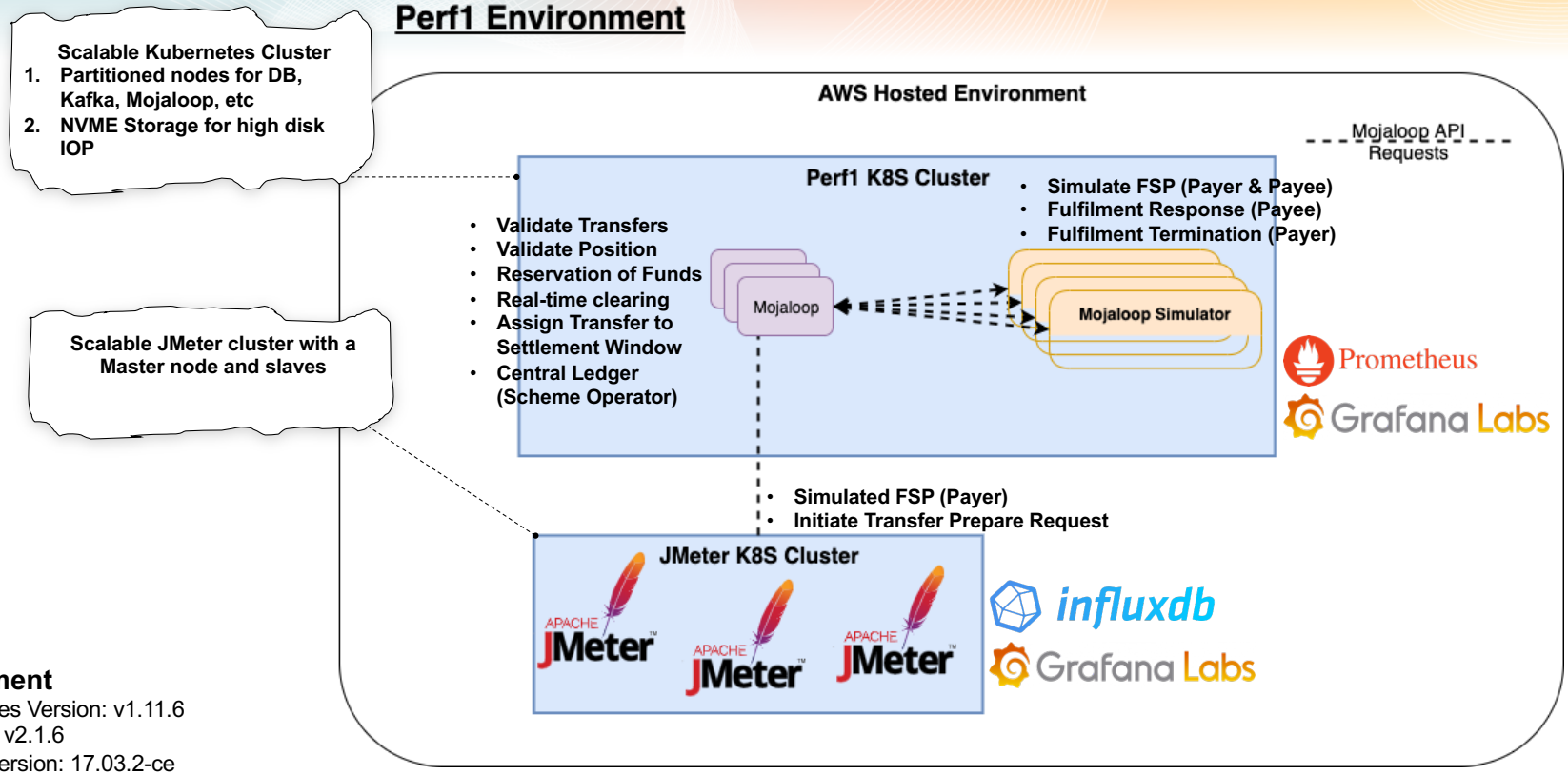
5. Tracing (event-sdk)
   a) v9.5.x Release supports base64 encoded traceState - containing spanId & custom key-value pairs
      - [●] timeApiPrepare – unix timestamp for start of Prepare Process
      - [●] timeApiFulfil – unix timestamp for start of Fulfil Process

6. Optimized Central-Ledger Handlers → Proof of Concept
   a) [●] Prepare + Position -> Prepare-Position Handler
   b) [●] Fulfil + Position -> Fulfil-Position Handler

Metrics: Financial (Transfer) Transaction End-to-end



Performance: Environment



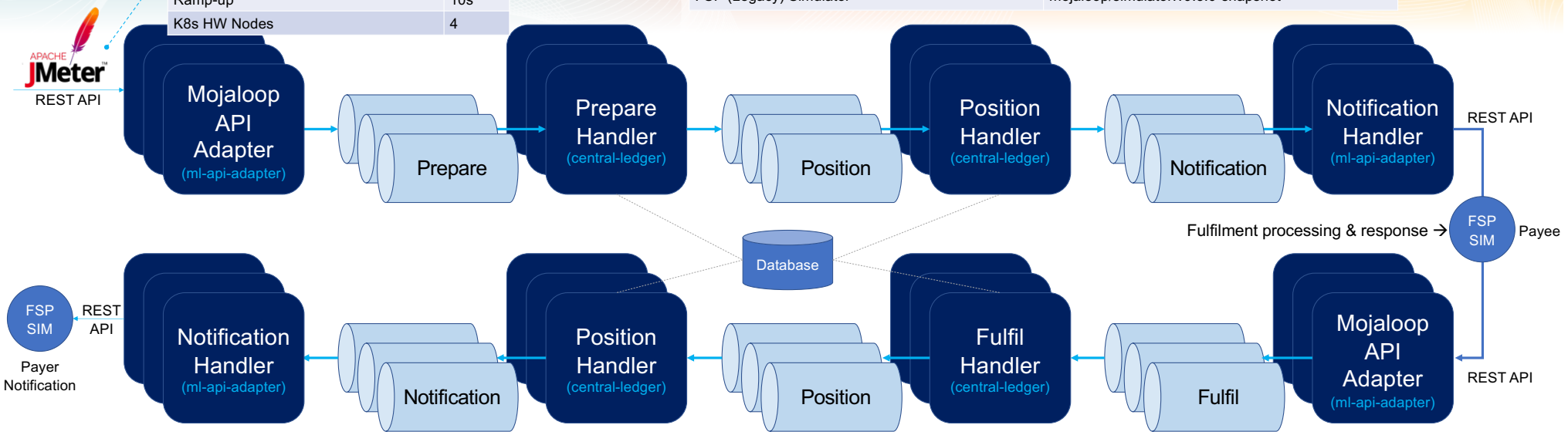
Environment

- Kubernetes Version: v1.11.6
- Rancher: v2.1.6
- Docker Version: 17.03.2-ce
- Kernel Version: 4.4.0-1052-aws
- Operating System: Ubuntu 16.04.4 LTS

Master Branch: Setup

Simulated FSP (Payer initiation)	Values
Number of FSPs	8
Slaves / Threads (Pods / Containers)	4
Target TP/S (100 TPS each Thread)	400
Ramp-up	10s
K8s HW Nodes	4

Component (Pod)	Docker Version
ML-API-Adapter – Mojaloop API Service	mojaloop/ml-api-adapter:v9.5.0-snapshot
ML-API-Adapter – Notification Service	mojaloop/ml-api-adapter:v9.5.0-snapshot
Central-Ledger – Prepare Handler	mojaloop/central-ledger:v9.5.1-snapshot
Central-Ledger – Position Handler	mojaloop/central-ledger:v9.5.1-snapshot
Central-Ledger – Fulfil Handler	mojaloop/central-ledger:v9.5.1-snapshot
FSP (Legacy) Simulator	mojaloop/simulator:v9.5.0-snapshot



AWS Machine
i3.xlarge
- 4 CPU
- 30GB Ram

AWS Machine
i3.xlarge
- 4 CPU
- 30GB Ram

AWS RDS db.r4.2xlarge
MySQL v5.7.22
- 8 CPU
- 30GB Ram
- 1TB HDD (Network Attached)
- 15k IOPS
- 200MB Write Throughput

Percona XtraDB i3.2xlarge
MySQL v5.7.19
- 8 CPU
- 61GB Ram
- 2TB HDD (NVMe)
- 180k IOPS
- N/A Write Throughput (local HDD)

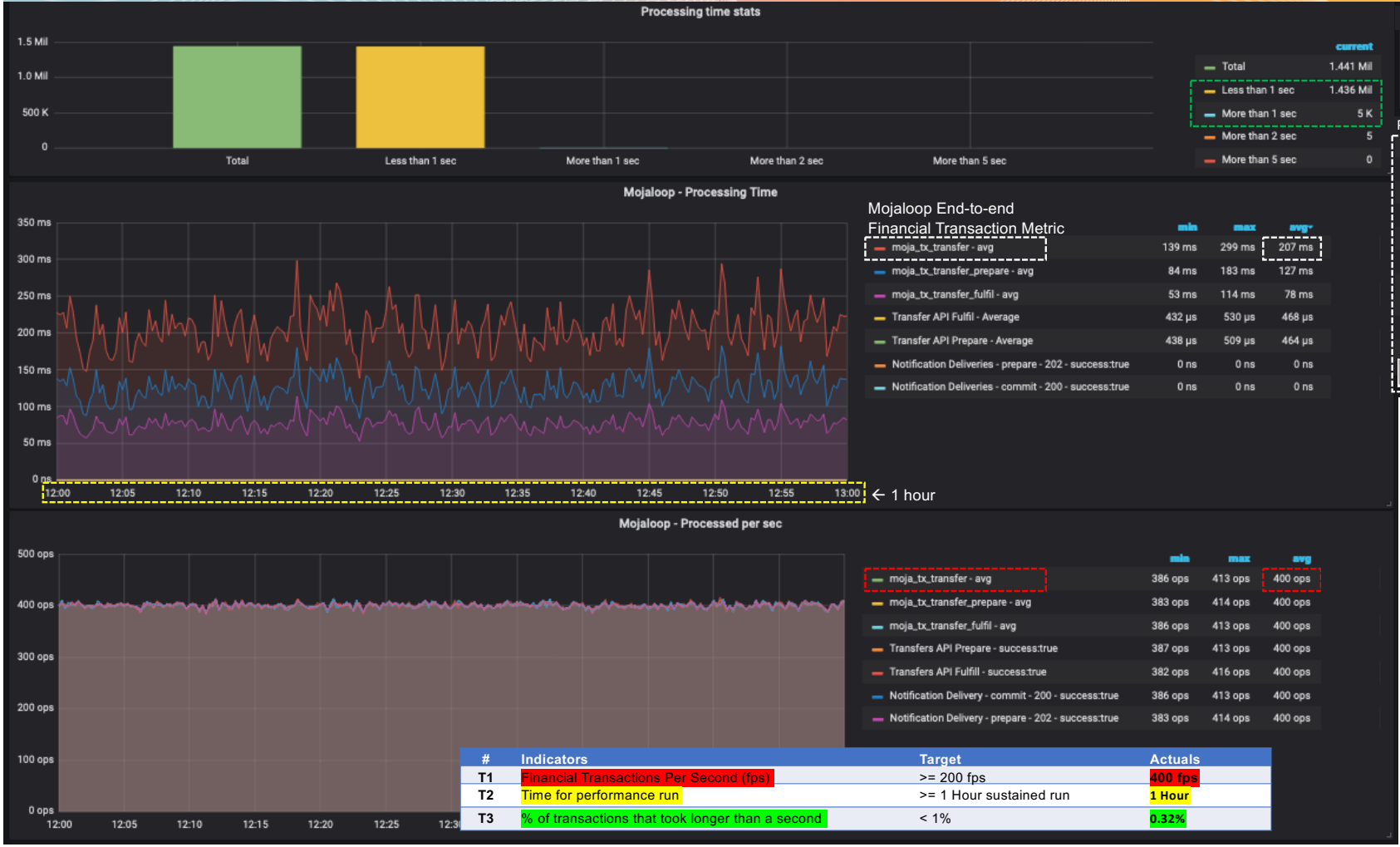
DB Options

APACHE JMeter
AWS Machine
m4.xlarge
- 4 CPU
- 16GB Ram

kafka
AWS Machine
i3.xlarge
- 4 CPU
- 16GB Ram

Notes:
- Account-lookup-Service disabled
- Quoting-Services disabled
- Central-Settlement disabled
- Partitions → 1:1 to Pod Scaling
- Logging
  - Async
  - Info level
  - Audits enabled
- Tracing
  - Event-Sidecars disabled

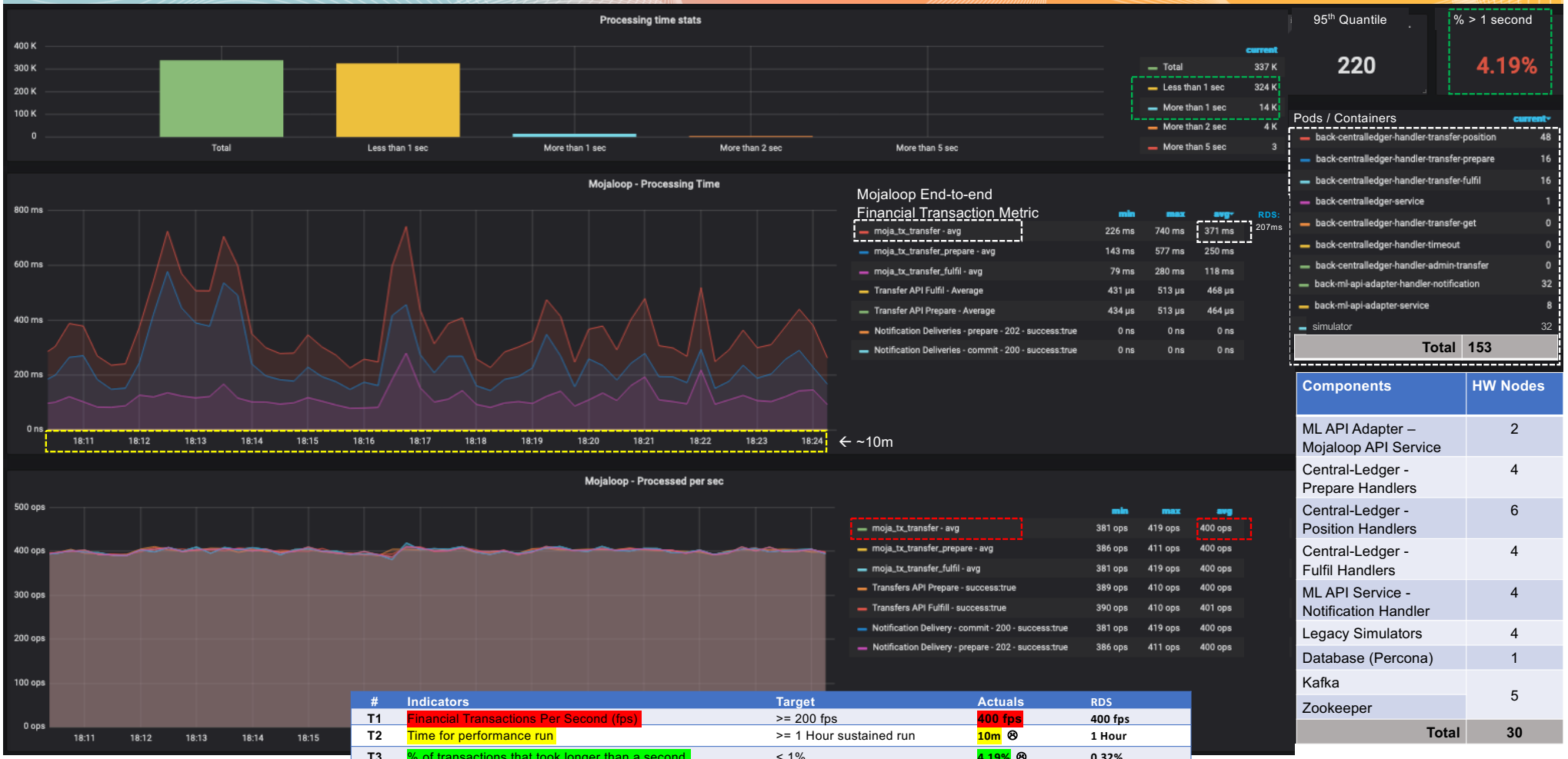
Master Branch: Results @ 400 financial transaction per/second (ftp/s) with RDS



#	Indicators	Target	Actuals
T1	Financial Transactions Per Second (fps)	>= 200 fps	400 fps
T2	Time for performance run	>= 1 Hour sustained run	1 Hour
T3	% of transactions that took longer than a second	< 1%	0.32%

Components	HW Nodes
ML API Adapter – Mojaloop API Service	2
Central-Ledger - Prepare Handlers	4
Central-Ledger - Position Handlers	6
Central-Ledger - Fulfil Handlers	4
ML API Service - Notification Handler	4
Legacy Simulators	4
Database (RDS)	1
Kafka	5
Zookeeper	
Total	**30**

Master Branch: Results @ 400 financial transaction per/second (ftp/s) self-hosted Percona XtraDB MySQL (unoptimized on-prem DB - early tentative result)

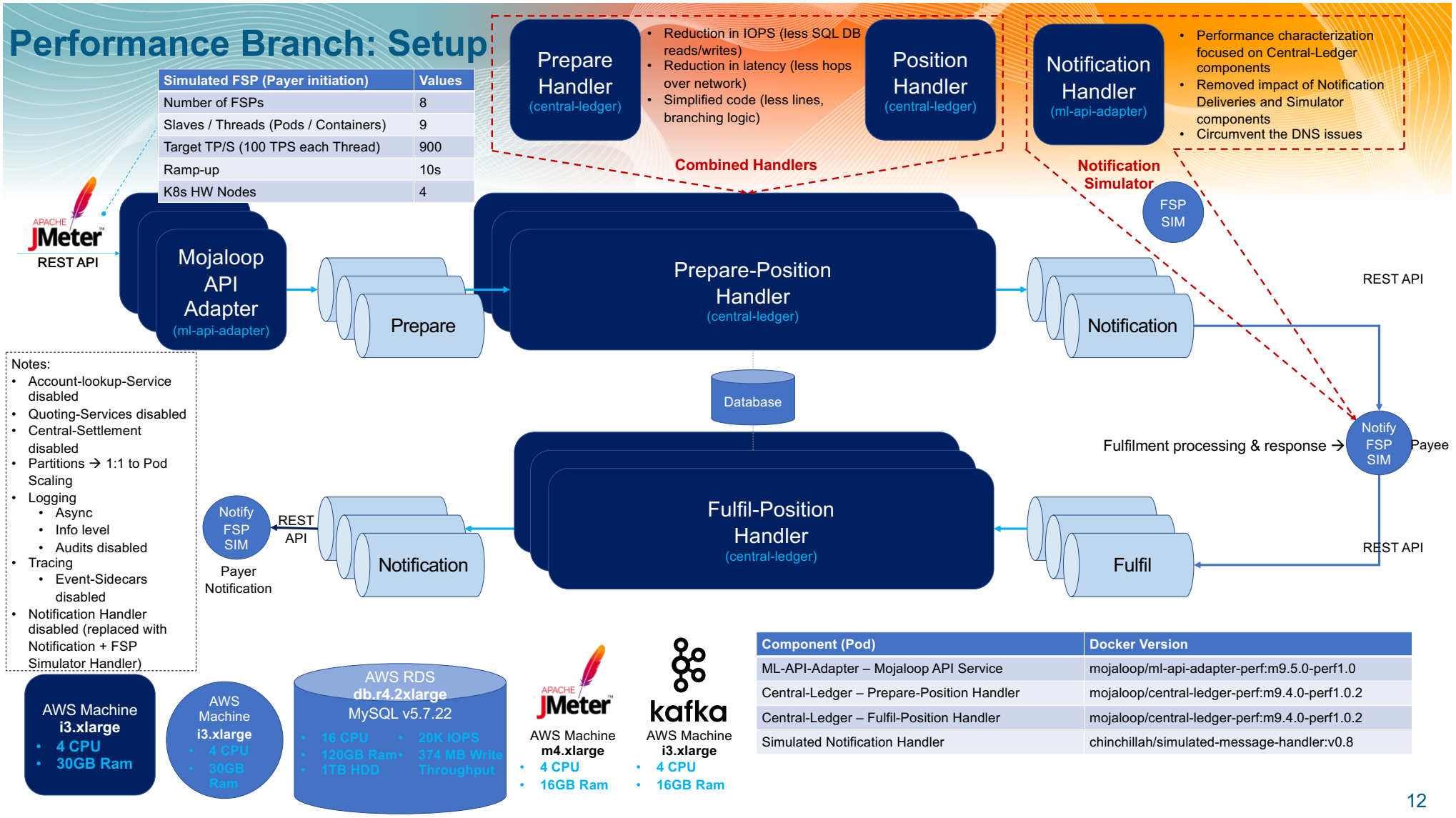


#	Indicators	Target	Actuals	RDS
T1	Financial Transactions Per Second (fps)	>= 200 fps	400 fps	400 fps
T2	Time for performance run	>= 1 Hour sustained run	10m ⊗	1 Hour
T3	% of transactions that took longer than a second	< 1%	4.19% ⊗	0.32%

Components	HW Nodes
ML API Adapter – Mojaloop API Service	2
Central-Ledger - Prepare Handlers	4
Central-Ledger - Position Handlers	6
Central-Ledger - Fulfil Handlers	4
ML API Service - Notification Handler	4
Legacy Simulators	4
Database (Percona)	1
Kafka	5
Zookeeper	
Total	**30**

Performance Branch: Setup

Simulated FSP (Payer initiation)	Values
Number of FSPs	8
Slaves / Threads (Pods / Containers)	9
Target TP/S (100 TPS each Thread)	900
Ramp-up	10s
K8s HW Nodes	4



Component (Pod)	Docker Version
ML-API-Adapter – Mojaloop API Service	mojaloop/ml-api-adapter-perf:m9.5.0-perf1.0
Central-Ledger – Prepare-Position Handler	mojaloop/central-ledger-perf:m9.4.0-perf1.0.2
Central-Ledger – Fulfil-Position Handler	mojaloop/central-ledger-perf:m9.4.0-perf1.0.2
Simulated Notification Handler	chinchillah/simulated-message-handler:v0.8

Performance Branch: Comparison

1. Differences to Master (Changes to be reviewed, verified and productionized)
   a) Combined Prepare + Position Handlers
   b) Combined Fulfil + Position Handlers
   c) Optimized SQL
      a) Duplicate Checks
      b) Position Changes
   d) Full Caching for non-stateful data (participants)
   e) Simulated Notification Handler (Direct-bypass to Fulfil topic from Notification Handler)

2. Improvements over Master with equivalent Database (db.r4.2xllarge @ 200MB/s Write Throughput)
   a) Less Hardware Nodes: 30 → 17
   b) Performance Improvement: ~20%+
   c) Less IOPS (reduced SQL Queries)
   d) Simpler code (improved maintainability)

3. Scaling Infrastructure on Performance Branch

Components	HW Nodes (i3.xlarge)	Pods / Containers
ML API Adapter – Mojaloop API Service	3	20
Central-Ledger – Prepare-Position Handlers	4	34
Central-Ledger - Fulfil-Position Handlers	4	34
Simulated Notification Handler	4	20
Database (RDS)	1	N/A
Kafka	5	5
Zookeeper		3
Total	**17**	**125**

+ AWS RDS db.r4.2xlarge
- 8 CPU
- 30GB Ram
- 15k IOPS / 500GB
- 200MB/s Write Throughput

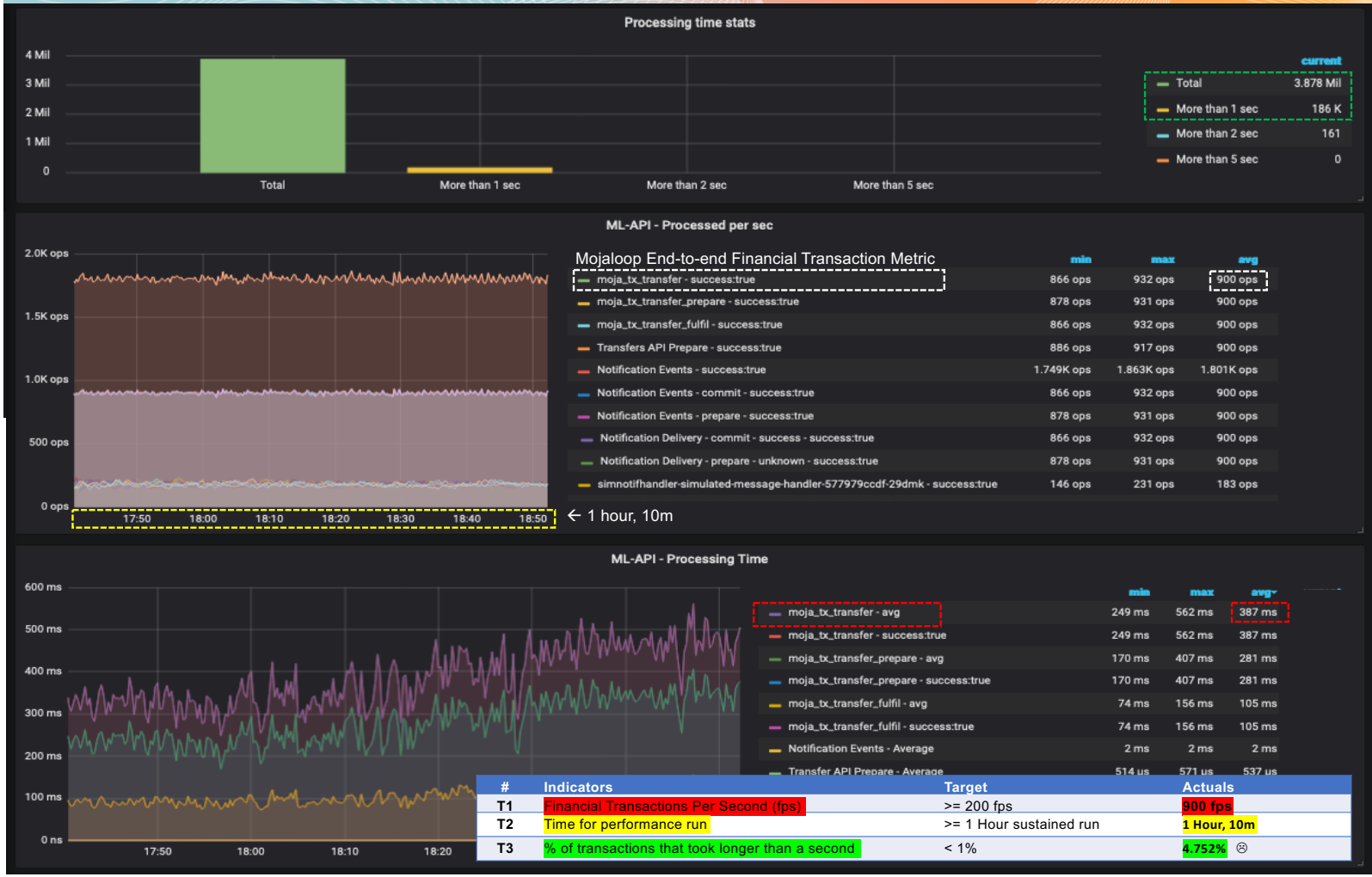
500-600 ftp/s → 900-1000 ftp/s

Notes:
- The hardware footprint has NOT been optimized for the Performance Branch

Components	HW Nodes (i3.xlarge)	Pods / Containers
ML API Adapter – Mojaloop API Service	7	7
Central-Ledger – Prepare-Position Handlers	7	50
Central-Ledger - Fulfil-Position Handlers	6	50
Simulated Notification Handler	3	10
Database (RDS)	1	N/A
Kafka	5	5
Zookeeper		3
Total	**29**	**125**

+ AWS RDS db.r4.4xlarge
- 16 CPU
- 120GB Ram
- 20k IOPS / 1TB
- 375MB/s Write Throughput

Performance Branch: Results @ financial transaction per/second (ftp/s)

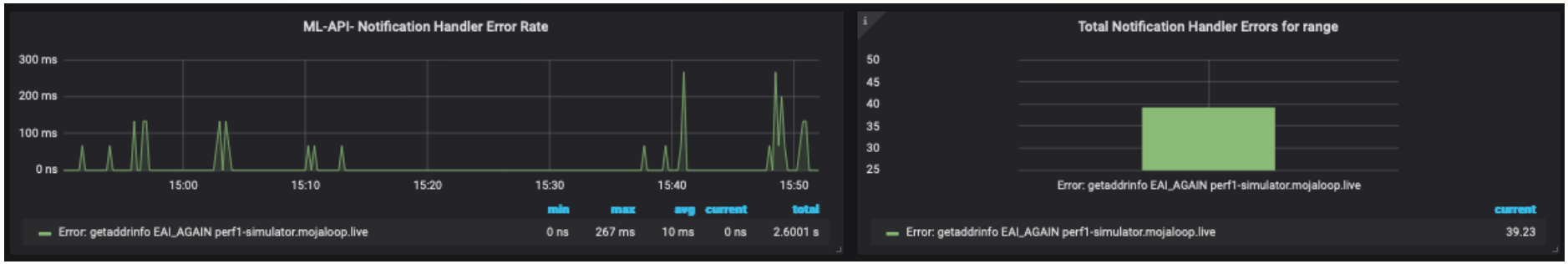


95th Quantile	% > 1 Sec
483	4.752%

Components	HW Nodes	Pods / Containers
ML API Adapter – Mojaloop API Service	7	7
Central-Ledger – Prepare-Position Handlers	7	50
Central-Ledger - Fulfil-Position Handlers	6	50
Simulated Notification Handlers	3	10
Database (RDS)	1	N/A
Kafka	5	5
Zookeeper		3
Total	**29**	**125**

Unresolved Issues

1. Notification Delivery
   a) DNS Issues due to issue on underlying Kubernetes OSS Environment (**i.e. not code related**) → temporarily resolved by going directly to Simulator's Service Cluster-IP address



2. Optimize On-prem MySQL Percona Xtra DB Cluster Deployment:
   a) Optimize DB configuration for underlying hardware:
      a) Queue Cache Size
      b) Utilize Available Memory
      c) etc.
   b) Optimize infrastructure (e.g. AWS)
      a) Ensure storage has guaranteed IOPS provisioned
   c) Resolve scaling issues
      a) Refactor Transaction Logic to cater for optimistic locking (**note: not an issue on Azure/AWS managed DBs**)
      b) Consider using alternative pessimistic distributed locking (e.g. Redlock) system in conjunction

3. Cloud Provider (AWS) Shared Infrastructure
   a) Provision dedicated instances for key systems (e.g. Kafka, Zookeeper, Database, etc)

15

Roadmap - Future Enhancements

1. Optimize Run-time Environment
   a) Running Kubernetes Cluster on dedicated hardware
   b) Identify & Resolve DNS Issue

2. Optimize On-prem MySQL Percona Xtra DB Cluster Deployment:
   a) Optimize DB configuration for underlying hardware:
      a) Queue Cache Size
      b) Available Memory
   b) Optimize infrastructure (e.g. AWS)
      a) Ensure storage has guaranteed IOPS provisioned
   c) Resolve scaling issues
      a) Refactor Transaction Logic to cater for optimistic locking
      b) Consider using alternative pessimistic distributed locking (e.g. Redlock) system in conjunction

3. Optimize On-prem Kafka Deployment:
   a) Separate Kafka / Zookeeper clusters
   b) Ensure dedicated hardware

4. Optimize Notification Handlers / Simulators
   a) Optimize HTTP Connections (keep-alive, connection-pool, etc)

5. Optimized SQL
   1. Duplicate Inserts
   2. Position Change

6. Caching
   a) Productionize Participant Limits (central-ledger)

7. Metrics – Instrument End-to-end Financial metrics
   a) Account-lookup
   b) Quoting-service
   c) Settlement-service
   d) Etc

8. Optimize Logging on remaining components
   a) Account-lookup
   b) Quoting-service
   c) Settlement-service
   d) Etc

9. Timeout Handler causes an intermittent dead-lock during high-loads
   a) Ensure no table scan locks (dirty reads)

10. Review & Productionize Combined Handler PoC

Roadmap - Future Characterizations

1. Scalability
   a) Horizontal Scaling of Mojaloop
   b) FSP ← impact of increasing # of FSPs

2. Other Mojaloop Services/Flows
   a) Account Lookup
   b) Quoting
   c) Settlements
   d) Transactions-requests
   e) Unhappy Paths

3. Event-Framework
   a) Event-Sidecar
   b) Event-Stream-Processor
   c) ElasticSearch (Event / Log ingestions)
   d) APM (Trace flow reporting)

4. API Gateway

5. Security
   a) JWS
   b) TLS

6. Simulators
   a) SDK-Schema-Adapter

7. Proof of Concepts
   a) Compare against baselines: master and performance branches

mojaloop

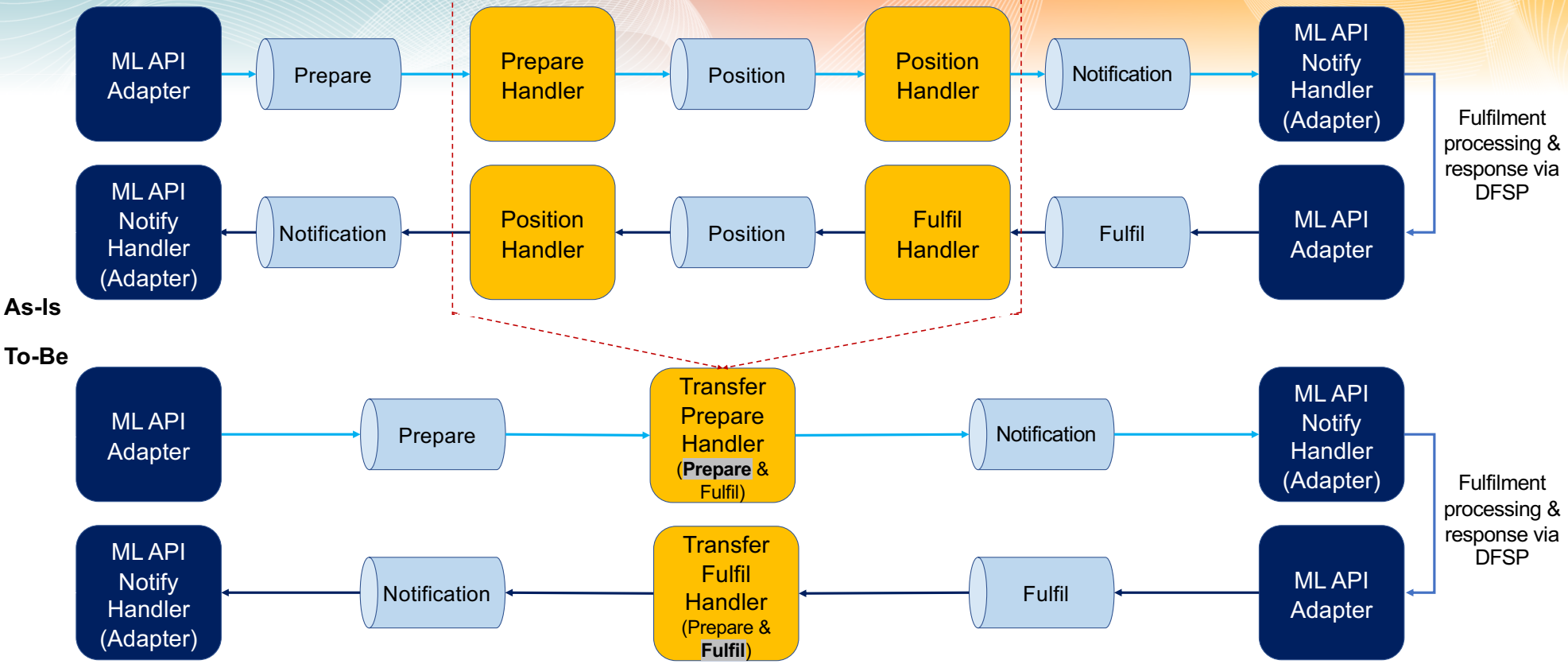
Questions?

mojaloop

Performance

APPENDIX

Performance Branch: Combined Central Handlers



- Reduction in IOPS (less SQL DB reads/writes)
- Reduction in latency (less hops over network)
- Simplified code (less lines, branching logic)

In-Memory Bulk Processing

Objective: Compete historic state record at time of rule execution, while never breeching liquidity checks

1. Position Handler Turnstile – single instance, bulk transfer set for processing

Single Position Handler

Position Lock per DFSP

1. DFSP's Position is held in READ/UPDATE lock while processing transfer set read from topic
2. Means there can only be one active handler per DFSP

2. Position Reserved Handler – Reserved Position, multi-instances

Multiple Position Handlers

Position Reserve and Position Finalise, separated with in memory processing

1. Position/Liquidity is reserved for total of transfer set
2. After in memory processing:
   a. Finalised transfer states are INSERTed, and
   b. DFSP Liquidity is finalised with UPDATE
3. Works in mixed or per DFSP processing mode
4. Multiple active handlers