LF NETWORKING

Developer & Testing Forum

# Scaling CPS

## Performance Improvement Learnings

Lee Anjella Macabuhay

Toine Siebelink

(Daniel Hanrahan)

Nov. 2023

https://lfnetworking.org

# Agenda

- High Level Overview CPS
- CPS Evolution
- Highlights
- Case Studies
  - ✓CM-Handle (de)-Registration
  - ✓CPS-Path Query Performance
  - ✓High Memory Usage
- Conclusions

LF NETWORKING
LFN Developer & Testing Forum

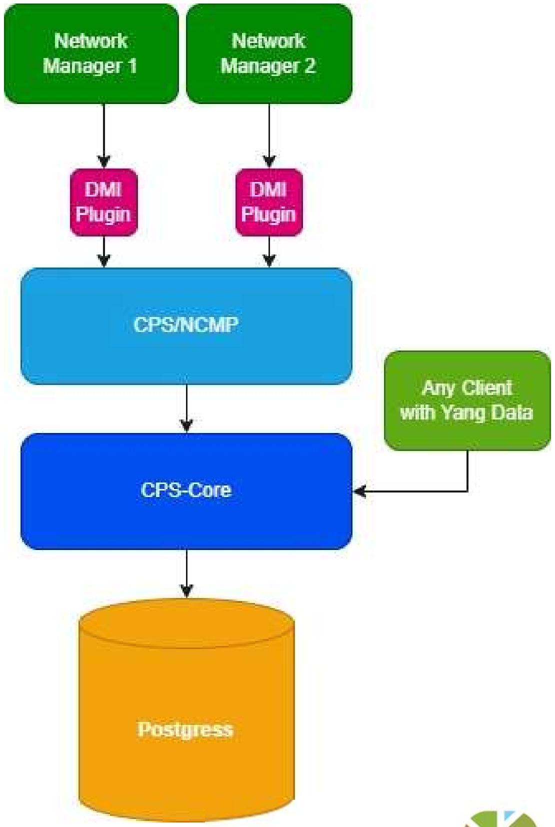
# CPS Overview

ERICSSON

CPS

# Configuration Persistence Service

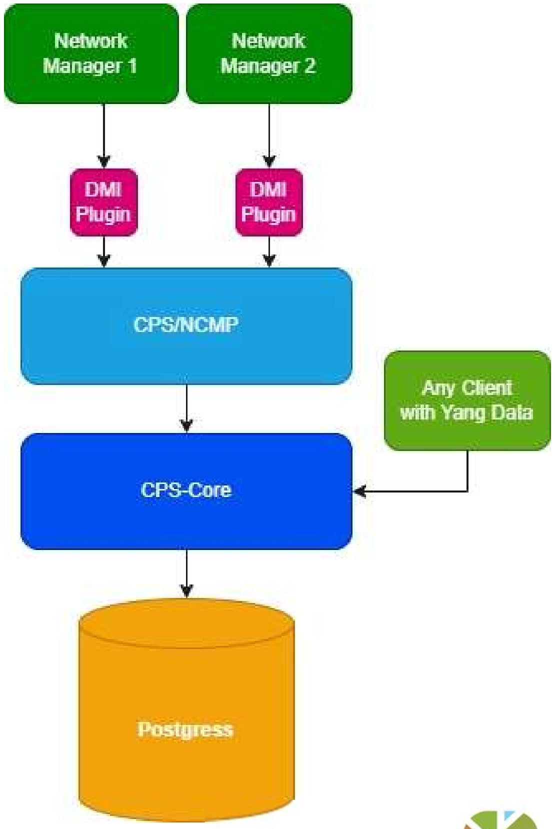
- CPS is a component designed to serve as a data repository for runtime data that needs persistence.
  - Example: Storing config parameters used by xNFs, like storing 5G network configuration parameter for a PNF that sets the mechanical tilt.
- CPS Implementation was started in 2020.
- Developed as part of the ONAP (Open Network Automation Platform)
  - Use in production by Deutsche Telekom and Ericsson!



# Configuration Persistence Service

LF NETWORKING
LFN Developer & Testing Forum

- Components of CPS:
  - CPS-Core provides the generic storage of Yang module data.
  - NCMP (Network Configuration Management Proxy) provides access to network configuration data at a higher level than CPS-core.
  - NCMP is designed to be vendor-neutral, using DMI (Data-Model-Inventory) plugins.
- CPS-core has CRUD operations + query language based on XPath.
  - Uses YANG for data modeling.
- CPS is cloud-native (REST), with SPI.
- Tech stack: Java, Spring, JPA+JDBC, Hazelcast, Kafka, Groovy/Spock



ERICSSON CPS

LF NETWORKING

LFN Developer & Testing Forum

# CPS Evolution

ERICSSON

CPS

# CPS Evolution

LFN Developer & Testing Forum

## Original Requirements

- ‘PoC’
- Focus on Functional
- Focus on Interfaces & Standardization
- Support ‘a few’ Nodes

→

## Evolved Requirements

- ‘PoP’
- Handle Large Data Sets
- Perform with Speed
- Scalability

# CPS Evolution

## Technology Choices

- SpringBoot
- Postgress DB
- JPA (Hibernate)

# CPS Evolution

## Community Feedback

- **Stakeholders**
  - ➢ Wipro (OpenRoadm model)
  - ➢ Deutsche Telekom (Queries)
  - ➢ T-Mobile
  - ➢ Capgemini
  - ➢ Ericsson (20,000 Nodes)

- **Challenges Highlighted**
  - ➢ Data Performance
    - o Writing large data slow
    - o Deleting slow
    - o Queries slow
  - ➢ Stability concerns
    - o Out of memory crashes

LF NETWORKING
LFN Developer & Testing Forum

# Highlights

ERICSSON CPS

# Highlights

The throughput of many CPS operations has been improved by orders of magnitude.

✓**CPS Path Query Optimization**

- ○ Worst-case (find all) time complexity reduced significantly:
    - From O(N^2) (quadratic) to O(N) (linear)
- ○ Best case (find one) improved from O(N) to O(1) constant

✓**Uniform Time Complexity**

- ○ All CPS operations now exhibit O(N) worst-case time complexity.

✓**New performance test suite (measuring time and memory)**

✓**Memory Efficiency**

- ○ Memory consumption during read operations reduced by more than 90%.

LF NETWORKING

LFN Developer & Testing Forum

# Case Study 1

CM-Handle (de)-Registration

ERICSSON

CPS

# CM-Handle (de-)Registration

LFN Developer & Testing Forum

- Ericsson had specific performance requirements
- Assessed current performance with Postman:
  - CPS was 100's of times slower than needed
  - CM-handle de-registration had $O(N^2)$ performance
- Many improvements made, driven by analysis & metrics
- Added Prometheus metrics
  - Discovered that some delete DB operations took a long time
  - Hundreds of thousands of DB calls for de-registering 20k CM-handles

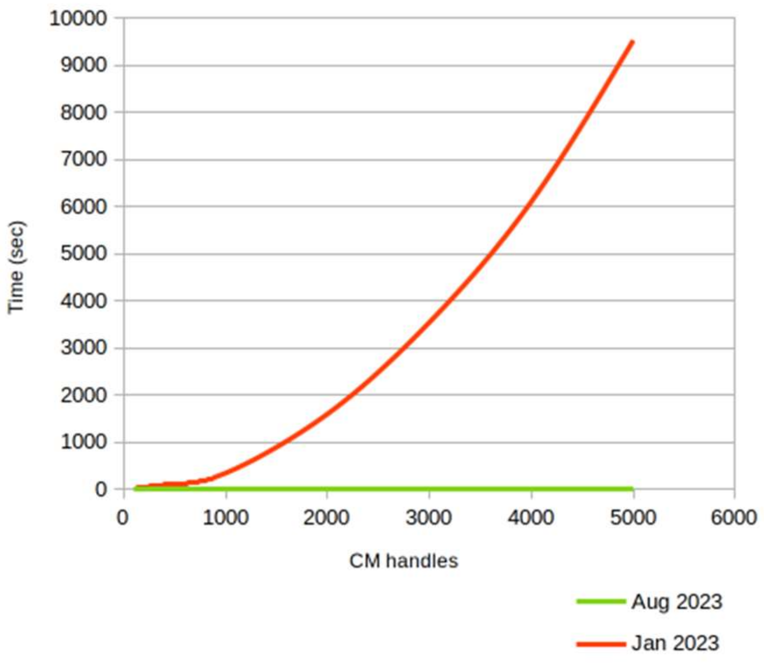
# CM-Handle (de)-registration

- Types of Improvements:
  - Batch implementation
  - SQL query optimization (online example: https://gerrit.onap.org/r/c/cps/+/133347)
  - Reduced total DB calls by 98% (see example #1)
  - Added DB indexes to speed some operations
  - Algorithmic changes for fetching descendants in data-trees (see example #2)

# CM-Handle (de)-registration

- Overall time complexity reduced from quadratic to linear
- In absolute terms, for Ericsson's use-case, performance is 1000's times faster (from 2 days to 1 minute)
- Performance exceeded requirements
- Addition of new performance tests prevent regressions (example #3)

| CM-handle deregistration (1st August 2023) | | |
|---|---|---|
| Total CM-handles | Time (sec) | CM-handles/sec |
| 500 | 1.53 | 327 |
| 1,000 | 2.65 | 378 |
| 5,000 | 13.26 | 377 |
| 10,000 | 25.93 | 386 |
| 20,000 | 56.15 | 356 |



LF NETWORKING
LFN Developer & Testing Forum

# Case Study 2

CPS Path Query Performance

ERICSSON

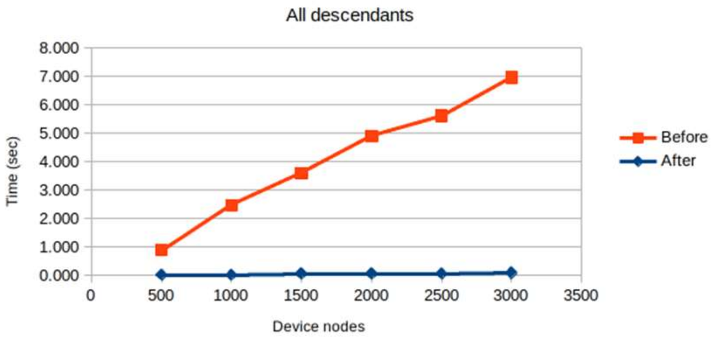
CPS

# CPS Path Query Performance

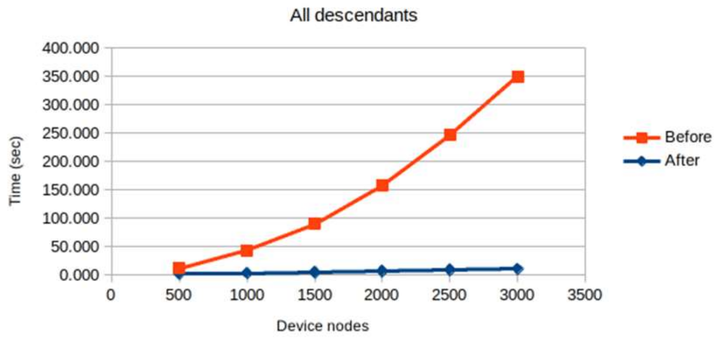
- Unacceptably slow queries
- Queries taking hours, preferably take < 1 minute
- Added new test cases, using OpenRoadM NM data, and compiled report showing quadratic time complexity.
- Proposal identifying causes and suggested improvements. See *Performance Analysis Study (wiki)*
- Delivered solution exhibiting :
  - Constant (irrespective of DB size) performance for single node data
  - Linear performance for query that return all data

# CPS Path Query Performance

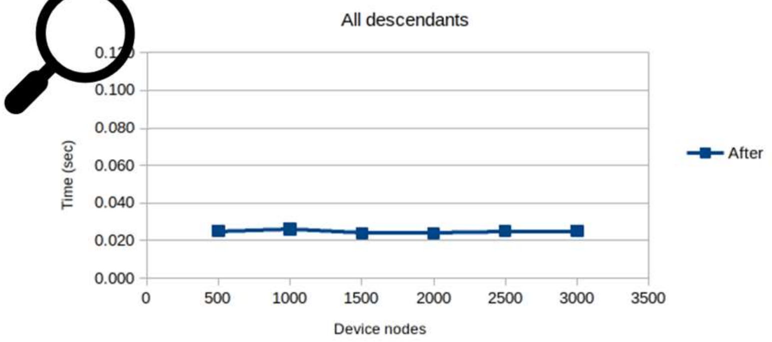
**Constant (irrespective of DB size) performance for single node data**



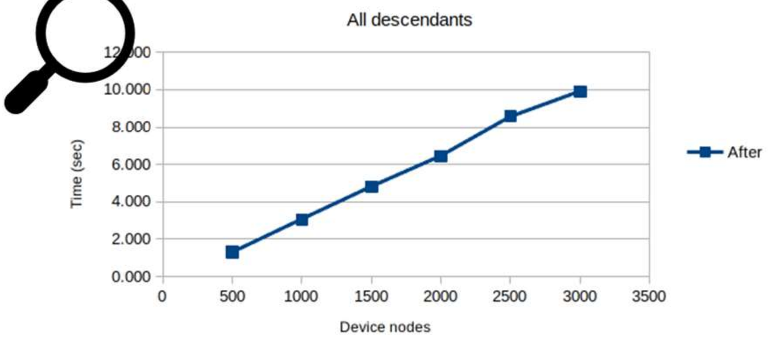
**Linear performance for query that return all data**



**Constant (irrespective of DB size) performance for single node data**



**Linear performance for query that return all data**



LF NETWORKING
LFN Developer & Testing Forum

# Case Study 3

High Memory Usage

ERICSSON

CPS

# High Memory Usage

- Problem: peak memory usage causing Out Of Memory Errors
- Used VisualVM for heap dump analysis
- Identified two possible causes and improvements
  - Single char bug: <= instead of < when fetch descendants in a tree structure ☹
  - Spring Data feature: “Interface Projection” convenient but costly! (see example #4)
- Exceeded the requirement of memory reduction by ~99%
- Corrective Actions: measure memory usage in tests (see example #5)

LF NETWORKING
LFN Developer & Testing Forum

# Conclusions

ERICSSON CPS

# Conclusions

1. Plans Change!

2. JPA / Hibernate generated code good to get started

3. Human designed code best for optimization
   Hibernate can Mix & Match

4. Value of Metrics

5. Importance of Early Performance Test, Daily Graphs

# Thank You For Your Attention!

## Any Questions?

LF NETWORKING

LFN Developer & Testing Forum

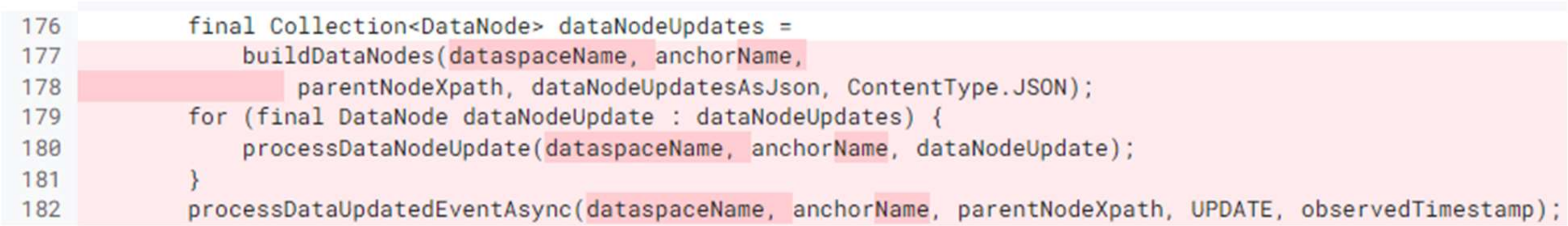
# Examples

ERICSSON

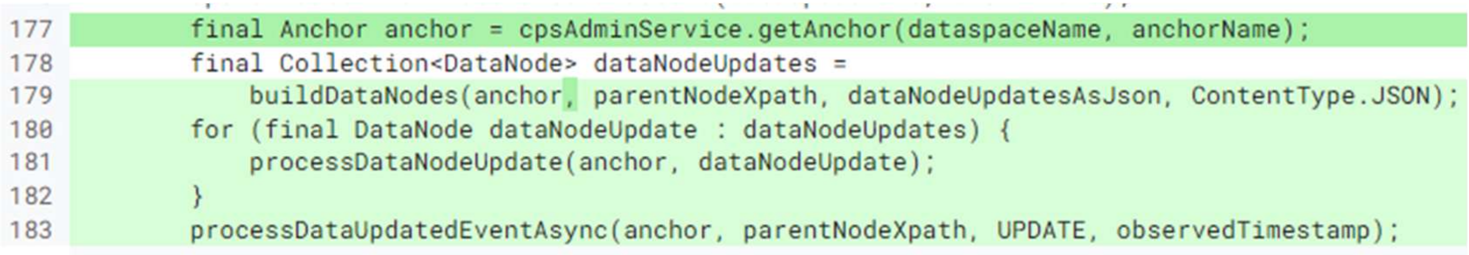
CPS

# Example 1: Reducing DB Calls

See https://gerrit.onap.org/r/c/cps/+/133627/6/cps-service/src/main/java/org/onap/cps/api/impl/CpsDataServiceImpl.java



```
        final Collection<DataNode> dataNodeUpdates =
            buildDataNodes(dataspaceName, anchorName,
                parentNodeXpath, dataNodeUpdatesAsJson, ContentType.JSON);
        for (final DataNode dataNodeUpdate : dataNodeUpdates) {
            processDataNodeUpdate(dataspaceName, anchorName, dataNodeUpdate);
        }
        processDataUpdatedEventAsync(dataspaceName, anchorName, parentNodeXpath, UPDATE, observedTimestamp);
```



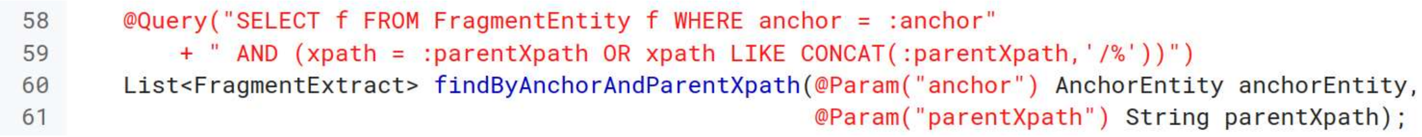
```
        final Anchor anchor = cpsAdminService.getAnchor(dataspaceName, anchorName);
        final Collection<DataNode> dataNodeUpdates =
            buildDataNodes(anchor, parentNodeXpath, dataNodeUpdatesAsJson, ContentType.JSON);
        for (final DataNode dataNodeUpdate : dataNodeUpdates) {
            processDataNodeUpdate(anchor, dataNodeUpdate);
        }
        processDataUpdatedEventAsync(anchor, parentNodeXpath, UPDATE, observedTimestamp);
```

| Method | Before | | After | |
|---|---|---|---|---|
| | # Calls | Sec. | # Calls | Sec. |
| `findByName` | 61,617 | 25.3 | 1,417 | 0.8 |
| `findByDataSpaceAndName` | 60,817 | 24.6 | 423 | 0.2 |

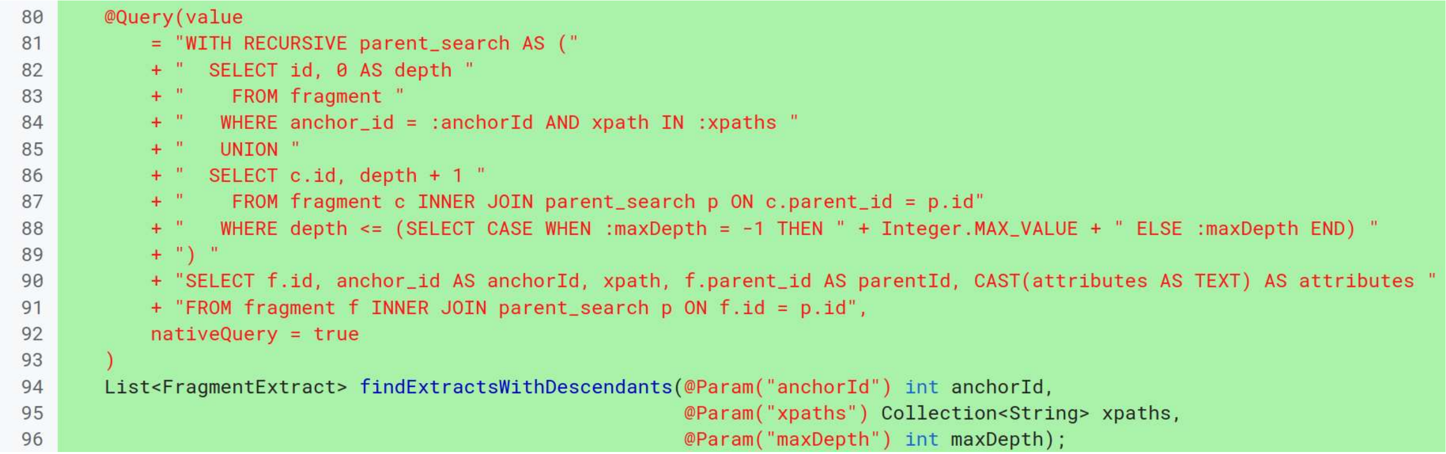
< Back to Case Study 1

# Example 2: Algorithm Changes

See https://gerrit.onap.org/r/c/cps/+/133511/12/cps-ri/src/main/java/org/onap/cps/spi/repository/FragmentRepository.java



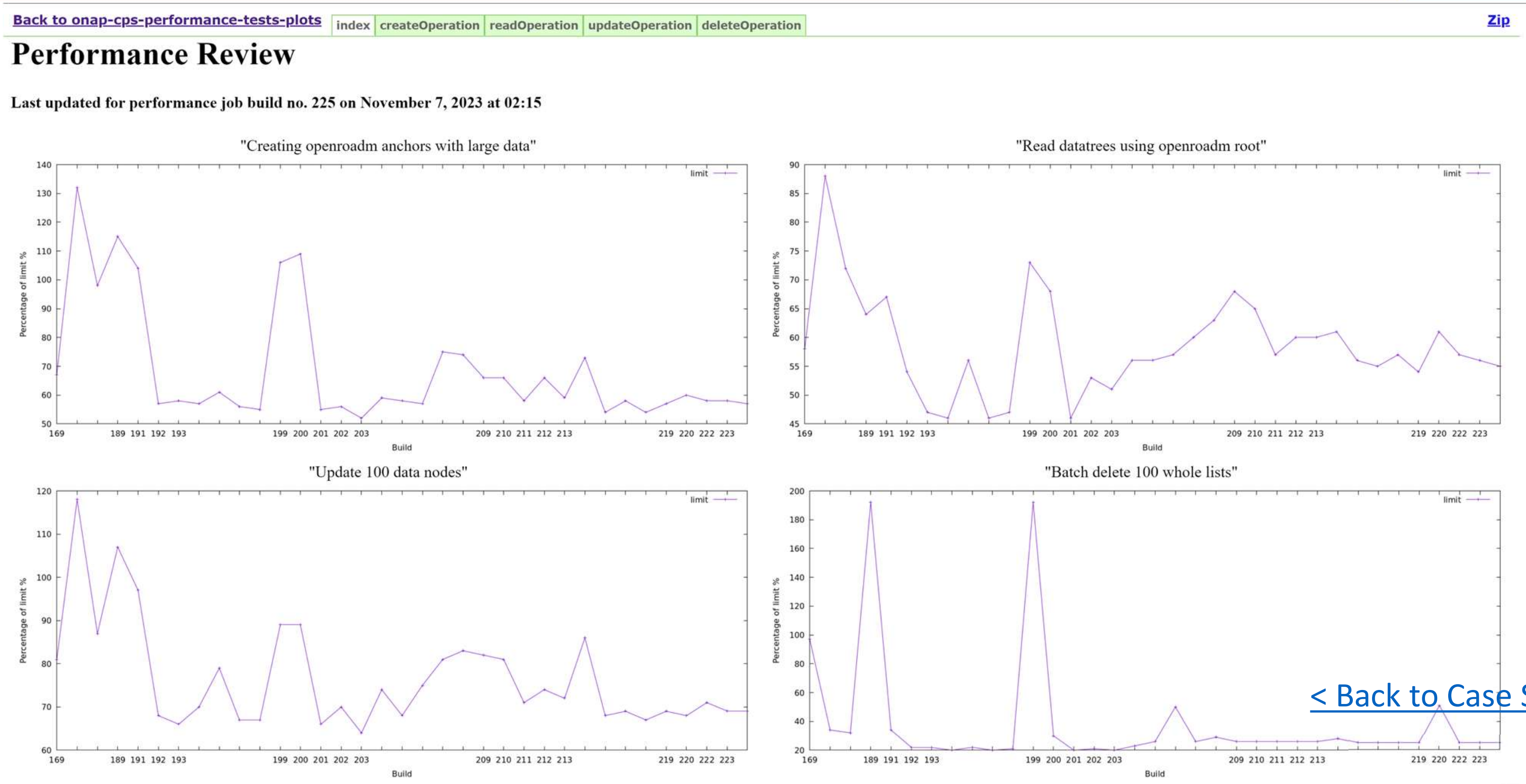
```
@Query("SELECT f FROM FragmentEntity f WHERE anchor = :anchor"
    + " AND (xpath = :parentXpath OR xpath LIKE CONCAT(:parentXpath,'/%'))")
List<FragmentExtract> findByAnchorAndParentXpath(@Param("anchor") AnchorEntity anchorEntity,
                                                 @Param("parentXpath") String parentXpath);
```



```
@Query(value
    = "WITH RECURSIVE parent_search AS ("
    + "  SELECT id, 0 AS depth "
    + "    FROM fragment "
    + "   WHERE anchor_id = :anchorId AND xpath IN :xpaths "
    + "   UNION "
    + "  SELECT c.id, depth + 1 "
    + "    FROM fragment c INNER JOIN parent_search p ON c.parent_id = p.id"
    + "   WHERE depth <= (SELECT CASE WHEN :maxDepth = -1 THEN " + Integer.MAX_VALUE + " ELSE :maxDepth END) "
    + ") "
    + "SELECT f.id, anchor_id AS anchorId, xpath, f.parent_id AS parentId, CAST(attributes AS TEXT) AS attributes "
    + "FROM fragment f INNER JOIN parent_search p ON f.id = p.id",
    nativeQuery = true
)
List<FragmentExtract> findExtractsWithDescendants(@Param("anchorId") int anchorId,
                                                  @Param("xpaths") Collection<String> xpaths,
                                                  @Param("maxDepth") int maxDepth);
```

< Back to Case Study 1

# Example 3: Performance tests plots



Back to onap-cps-performance-tests-plots | index | createOperation | readOperation | updateOperation | deleteOperation | Zip

## Performance Review

**Last updated for performance job build no. 225 on November 7, 2023 at 02:15**

"Creating openroadm anchors with large data"

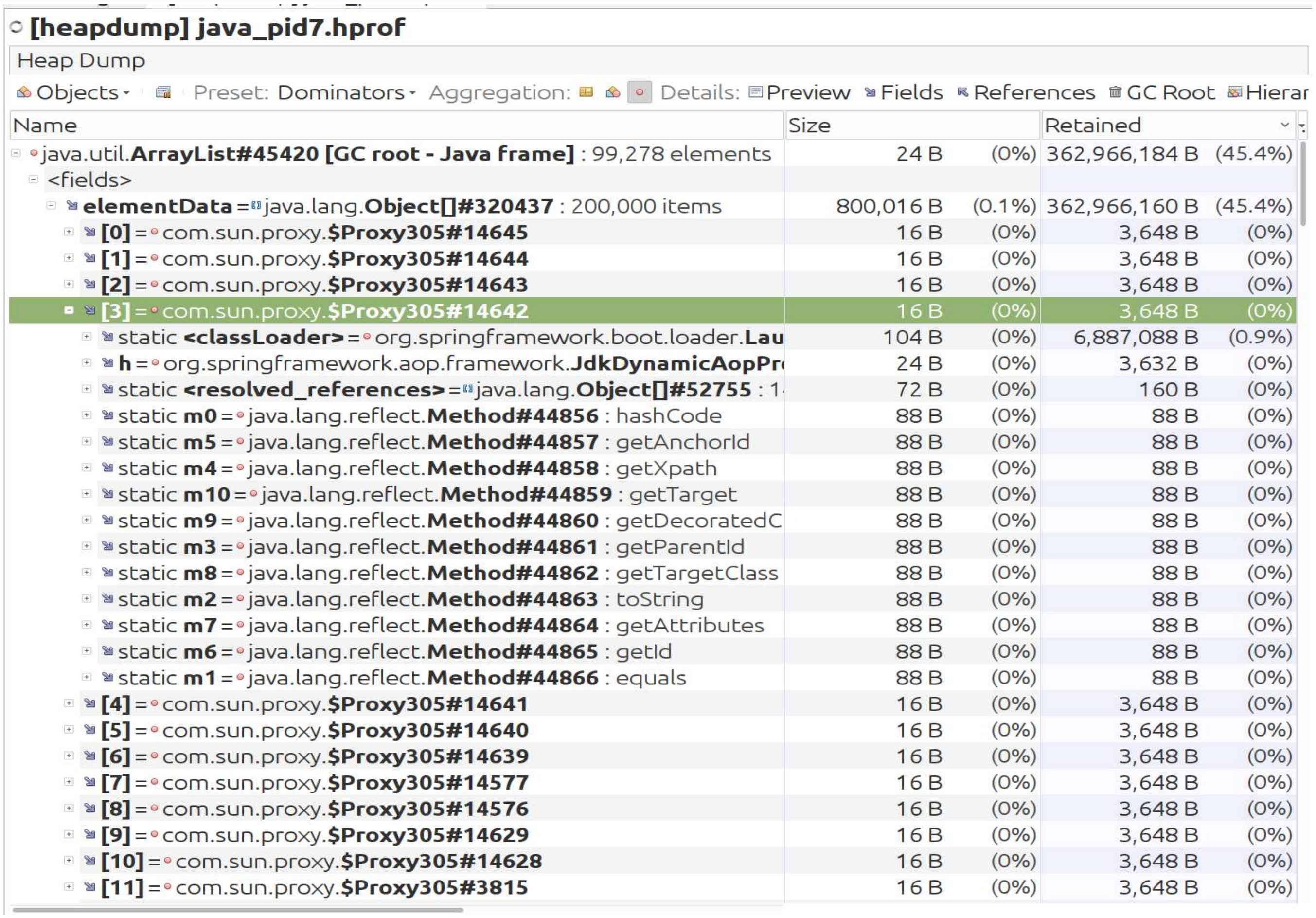
"Read datatrees using openroadm root"

"Update 100 data nodes"

"Batch delete 100 whole lists"

< Back to Case Study 1

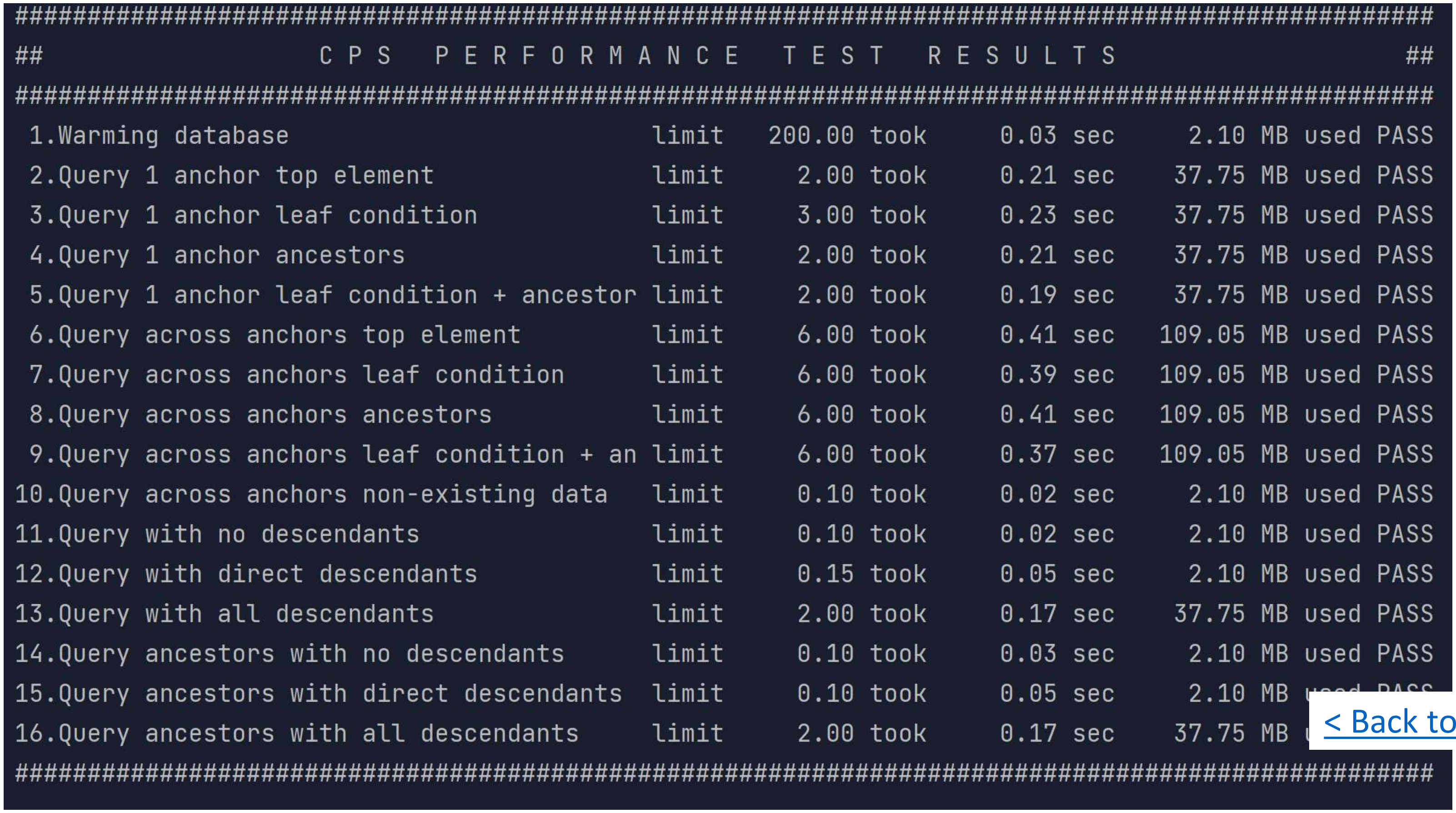
# Example 4: JVM Dump Analysis



| Name | Size | | Retained | |
|---|---|---|---|---|
| java.util.**ArrayList#45420 [GC root - Java frame]** : 99,278 elements | 24 B | (0%) | 362,966,184 B | (45.4%) |
| <fields> | | | | |
| **elementData** = java.lang.**Object[]#320437** : 200,000 items | 800,016 B | (0.1%) | 362,966,160 B | (45.4%) |
| **[0]** = com.sun.proxy.**$Proxy305#14645** | 16 B | (0%) | 3,648 B | (0%) |
| **[1]** = com.sun.proxy.**$Proxy305#14644** | 16 B | (0%) | 3,648 B | (0%) |
| **[2]** = com.sun.proxy.**$Proxy305#14643** | 16 B | (0%) | 3,648 B | (0%) |
| **[3]** = com.sun.proxy.**$Proxy305#14642** | 16 B | (0%) | 3,648 B | (0%) |
| static **<classLoader>** = org.springframework.boot.loader.**Lau** | 104 B | (0%) | 6,887,088 B | (0.9%) |
| **h** = org.springframework.aop.framework.**JdkDynamicAopPr** | 24 B | (0%) | 3,632 B | (0%) |
| static **<resolved_references>** = java.lang.**Object[]#52755** : 1 | 72 B | (0%) | 160 B | (0%) |
| static **m0** = java.lang.reflect.**Method#44856** : hashCode | 88 B | (0%) | 88 B | (0%) |
| static **m5** = java.lang.reflect.**Method#44857** : getAnchorId | 88 B | (0%) | 88 B | (0%) |
| static **m4** = java.lang.reflect.**Method#44858** : getXpath | 88 B | (0%) | 88 B | (0%) |
| static **m10** = java.lang.reflect.**Method#44859** : getTarget | 88 B | (0%) | 88 B | (0%) |
| static **m9** = java.lang.reflect.**Method#44860** : getDecoratedC | 88 B | (0%) | 88 B | (0%) |
| static **m3** = java.lang.reflect.**Method#44861** : getParentId | 88 B | (0%) | 88 B | (0%) |
| static **m8** = java.lang.reflect.**Method#44862** : getTargetClass | 88 B | (0%) | 88 B | (0%) |
| static **m2** = java.lang.reflect.**Method#44863** : toString | 88 B | (0%) | 88 B | (0%) |
| static **m7** = java.lang.reflect.**Method#44864** : getAttributes | 88 B | (0%) | 88 B | (0%) |
| static **m6** = java.lang.reflect.**Method#44865** : getId | 88 B | (0%) | 88 B | (0%) |
| static **m1** = java.lang.reflect.**Method#44866** : equals | 88 B | (0%) | 88 B | (0%) |
| **[4]** = com.sun.proxy.**$Proxy305#14641** | 16 B | (0%) | 3,648 B | (0%) |
| **[5]** = com.sun.proxy.**$Proxy305#14640** | 16 B | (0%) | 3,648 B | (0%) |
| **[6]** = com.sun.proxy.**$Proxy305#14639** | 16 B | (0%) | 3,648 B | (0%) |
| **[7]** = com.sun.proxy.**$Proxy305#14577** | 16 B | (0%) | 3,648 B | (0%) |
| **[8]** = com.sun.proxy.**$Proxy305#14576** | 16 B | (0%) | 3,648 B | (0%) |
| **[9]** = com.sun.proxy.**$Proxy305#14629** | 16 B | (0%) | 3,648 B | (0%) |
| **[10]** = com.sun.proxy.**$Proxy305#14628** | 16 B | (0%) | 3,648 B | (0%) |
| **[11]** = com.sun.proxy.**$Proxy305#3815** | 16 B | (0%) | 3,648 B | (0%) |

< Back to Case Study 3

# Example 5:
# Sample Performance Test Report



```
##############################################################################################
##                   C P S   P E R F O R M A N C E   T E S T   R E S U L T S               ##
##############################################################################################
 1.Warming database                           limit   200.00 took     0.03 sec     2.10 MB used PASS
 2.Query 1 anchor top element                 limit     2.00 took     0.21 sec    37.75 MB used PASS
 3.Query 1 anchor leaf condition              limit     3.00 took     0.23 sec    37.75 MB used PASS
 4.Query 1 anchor ancestors                   limit     2.00 took     0.21 sec    37.75 MB used PASS
 5.Query 1 anchor leaf condition + ancestor limit     2.00 took     0.19 sec    37.75 MB used PASS
 6.Query across anchors top element           limit     6.00 took     0.41 sec   109.05 MB used PASS
 7.Query across anchors leaf condition        limit     6.00 took     0.39 sec   109.05 MB used PASS
 8.Query across anchors ancestors             limit     6.00 took     0.41 sec   109.05 MB used PASS
 9.Query across anchors leaf condition + an limit     6.00 took     0.37 sec   109.05 MB used PASS
10.Query across anchors non-existing data     limit     0.10 took     0.02 sec     2.10 MB used PASS
11.Query with no descendants                  limit     0.10 took     0.02 sec     2.10 MB used PASS
12.Query with direct descendants              limit     0.15 took     0.05 sec     2.10 MB used PASS
13.Query with all descendants                 limit     2.00 took     0.17 sec    37.75 MB used PASS
14.Query ancestors with no descendants        limit     0.10 took     0.03 sec     2.10 MB used PASS
15.Query ancestors with direct descendants    limit     0.10 took     0.05 sec     2.10 MB used PASS
16.Query ancestors with all descendants       limit     2.00 took     0.17 sec    37.75 MB u
##############################################################################################
```

< Back to Case Study 3