CPS Updates & Demo

Toine Siebelink

May 2024
Who am I?

- Master Engineer with Ericsson 20+ Years
- Working in Open Source (ONAP) since 2020 as Project Technical Lead for CPS
- My Focus
  - Quality Testware
  - Quality Software
  - Performance
- CPS has achieved OpenSSF Gold Badge Best Practices (quality & security)
Agenda

1. Quick Recap of CPS Concepts
2. Demo of Delta Feature
3. Performance Improvements & Micro Benchmark Testing
4. Warm Up Effect
5. Performance Monitoring
CPS Concepts
Dataspace, Model Sets, Anchors
A ‘dataspace’ is like the fenced in patch of land containing only certain ‘types’ of trees.

A dataspace contains a list of the types of trees it can contain i.e. the Yang modules that describe the data trees.

An application defines a dataspace (name) and is responsible for maintaining the models in it.

Each dataspace has an owner who controls the gate to determine who can access the trees.
Live Demo 1
Delta Feature
Delta Feature

- Community initiated contribution
  - Compare data for different anchors

- Context:
  - Storing CM snapshots over time
  - Planned configuration changes

- Use Cases
  - Compare planned changes with recent snapshots
  - See impact of proposed configuration plans
  - Compare live configurations
  - Delta notifications (future)

- Can compare ‘Book Stores’ too 😊
Delta Feature

- Output
  - Based on IETF standards
    - RFC6902: JSON Patch
    - RFC9144 Comparison of Network Management Datastore Architecture (NMDA) Datastores
  - ADD
  - REMOVE
  - UPDATE

- Deutsche Telecom are planning to use CPS Delta Feature in production for next year
Delta Feature Demo

- **Swagger-UI**
- Create
  - Dataspace
  - Model Sets
  - Anchors
  - Data
- Query
- Delta Report
  - Update
  - Add
  - Remove
Performance

Improvements & Micro Benchmark Testing
<table>
<thead>
<tr>
<th>Slogan</th>
<th>Effect</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Hibernate Batching</td>
<td>&gt; 2x (store operations)</td>
<td>org.onap.cps.integration.performance.cps.WritePerfTest#Writing openroadm data has linear time.</td>
</tr>
<tr>
<td>Normalize JSON for store &amp; update</td>
<td>~ 2x (update operations)</td>
<td>org.onap.cps.integration.performance.cps.UpdatePerfTest#Replace single data node and descendants: #scenario.</td>
</tr>
<tr>
<td>Saving CM Handles (NCMP Network Configuration Mgmt. &amp; Persistence)</td>
<td>4x</td>
<td></td>
</tr>
<tr>
<td>Batch yang data parsing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faster absolute cps-path queries</td>
<td>5-10x (CPS Queries)</td>
<td>org.onap.cps.integration.performance.ncmp.CmHandleQueryPerfTest#CM-handle is looked up by alternate-id.</td>
</tr>
<tr>
<td>Removed Redundant Spring Security (using service mesh instead)</td>
<td>Overhead 100ms -&gt; 10ms</td>
<td></td>
</tr>
</tbody>
</table>
Micro-Benchmark Testing

- Spock & Groovy ‘unit’ test Framework
  - Given
  - When
  - Then
  - Where
- Spring-boot Test & Test Containers
  - PostgreSQLContainer
  - KafkaTestContainer
- Single Method, large data size and/or repeats
- Semi Integration
- Run fast & often in IDE
Base Test Class

```java
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK, classes = [CpsDataspaceService])
@Testcontainers
@EnableAutoConfiguration
@AutoConfigureMockMvc
@EnableJpaRepositories(basePackageClasses = [DataspaceRepository])
@ComponentScan(basePackages = ['org.onap.cps'])
@EntityScan('org.onap.cps.spi.entities')
abstract class CpsIntegrationSpecBase extends Specification {

    @Shared
    DatabaseTestContainer databaseTestContainer = DatabaseTestContainer.getInstance()

    @Shared
    KafkaTestContainer kafkaTestContainer = KafkaTestContainer.getInstance()

    @Autowired
    MockMvc mvc

    @Autowired
    CpsDataspaceService cpsDataspaceService
```
def 'Writing openroadm data has linear time.'():
    given: 'an empty anchor exists for openroadm'
        cpsAnchorService.createAnchor(CPS_PERFORMANCE_TEST_DATASPACE, LARGE_SCHEMA_SET, WRITE_TEST_ANCHOR)
    and: 'a list of device nodes to add'
        def jsonData = generateOpenRoadData(totalNodes)
    when: 'device nodes are added'
        resourceMeter.start()
        cpsDataService.saveData(CPS_PERFORMANCE_TEST_DATASPACE, WRITE_TEST_ANCHOR, jsonData, OffsetDateTime.now())
        resourceMeter.stop()
    then: 'the operation takes less than #expectedDuration and memory used is within limit'
        recordAndAssertResourceUsage("Writing ${totalNodes} devices", expectedDuration, resourceMeter.getTotalTimeInSeconds())
    cleanup:
        cpsAnchorService.deleteAnchor(CPS_PERFORMANCE_TEST_DATASPACE, WRITE_TEST_ANCHOR)
    where:
        totalNodes || expectedDuration
        50      || 1
        100     || 2
        200     || 3
        400     || 5
Live Demo 2
Micro Benchmark Testing
Performance Improvements

1. Writing openroadm data has linear time
2. Replace single data node and descendants
3. CM-handle is looked up by alternate-id
Warm Up Effect
Warm Up Effect

- JIT: Just-In-Time Compilation
  - Lazy Class Loading, first call slower
  - Native Code -Cache
- Tiered Compilation in JVM (default since Java 8)
  - Interpreted with Profiling
  - C1 (client) Compiler with Profiling
  - C2 (server) Compiler, non-profiled

References
- [https://www.baeldung.com/java-jvm-warmup](https://www.baeldung.com/java-jvm-warmup)
- [https://www.baeldung.com/jvm-tiered-compilation](https://www.baeldung.com/jvm-tiered-compilation)
Warm Up Effect

Effect of Warm Up on CM-Handle Look-Up Query (100 samples)
Performance Monitoring
Prevent Performance Regression
Performance Monitoring

- Run micro benchmark tests regularly (2h)
- Jenkins plugin to store and graph results
- Assess graphs every day during Standup
- Look for degradation
- Confirm improvements when tuning/fixing performance issues
Performance Monitoring

Lessons Learned

• Reasonable size test for less variation
• Run in well controlled environment (physical)
• Run multiple times day, not just nightly
• Linux / Windows, Laptop/Server Virtual/Physical affects different test differently
• Warm Up Effect
Future Test Improvements

- K6 (performance) Test
  - Full End 2 End
  - Using REST Interface
  - Designed for Performance Test
    - Concurrent users
  - More advance criteria
  - More control
    - Repeat requests in loop
    - Duration based instead of loop

- PoC in Progress
Stay in Touch

- Toine.Siebelink@est.tech
- https://wiki.onap.org/display/DW/Configuration+Persistence+Service+Developer%27s+Landing+Page
References

- https://wiki.onap.org/display/DW/Configuration+Persistence+Service+Developer%27s+Landing+Page
- https://spockframework.org/
- https://www.baeldung.com/java-jvm-warmup
- https://www.baeldung.com/jvm-tiered-compilation
- https://plugins.jenkins.io/htmlpublisher/
- https://k6.io/
- ONE Summit 2022: Spock & Groovy