CPS Basic Performance (Load) Test

Table of Contents

- Test Flow Description
  - Test Data
- Environment
  - Hardware details
  - Software details
- Test Results
  - Operation execution time
  - Resource usage
- Test Application
  - Source
  - Setting up

Addresses **CPS-307** - Getting issue details... STATUS

Test Flow Description

According to requirements the test flow included following operation:

- Create new anchor with unique name in given dataspace
- Create data node - full data tree upload for given anchor
- Update data node - node fragment replacement
- Remove anchor (and associated data)

The dataspace and schema set were predefined.

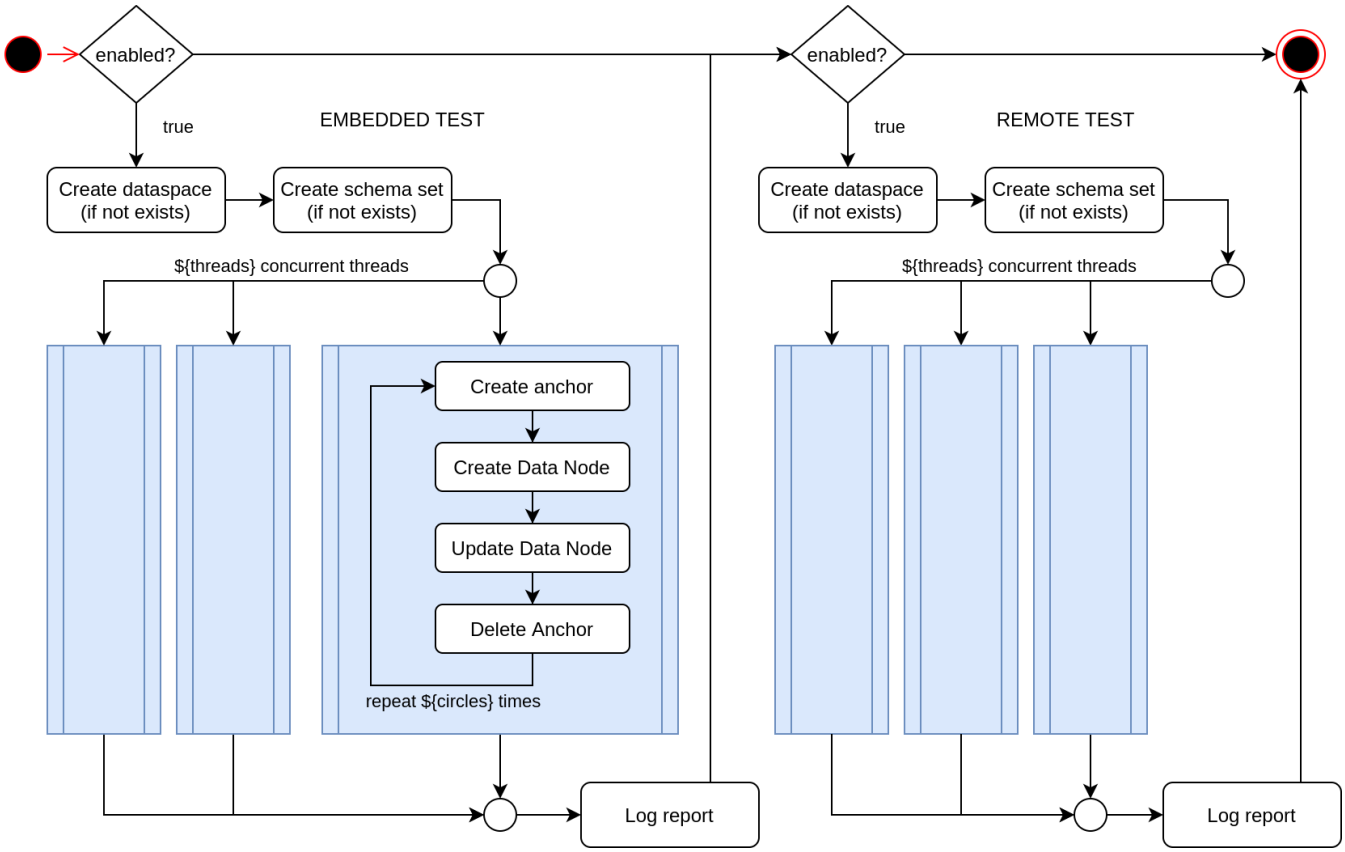
The service functionality was tested in two modes:

- Embedded – the CPS was accessed directly via Java API
- Remote – The CPS was accessed using REST API

Load:

- The flow was repeated in a loop multiple times (defined with `circles` parameter);
  no delay between loops
- The flow loops were executed in a multiple concurrent threads (`threads` parameter);
  all threads were started subsequently with no delay
- each thread used same CPS service (embedded case) and REST client (remote case) instances
- All the data used (except schema set preparation) was taken from memory (loaded on preparation stage)

Full test application flow is shown on diagram below



Test Data

Yang model (schema set) based on:

- **iana-if-type**@2017-01-19.yang
- **ieee802-dot1q-types**.yang
- **ietf-inet-types**@2013-07-15.yang
- **ietf-interfaces**@2018-02-20.yang
- **ietf-l2-topology**@2020-11-15.yang
- **ietf-l2-topology-state**@2020-11-15.yang
- **ietf-network**@2018-02-26.yang
- **ietf-network-state**@2018-02-26.yang
- **ietf-network-topology**@2018-02-26.yang
- **ietf-network-topology-state**@2018-02-26.yang
- **ietf-yang-types**@2013-07-15.yang

Full data JSON is (taken as is from from RFC-8944 Appendix-B):

- ietf-network-sample-rfc8944.json (5.5Kb, 52 data nodes total)

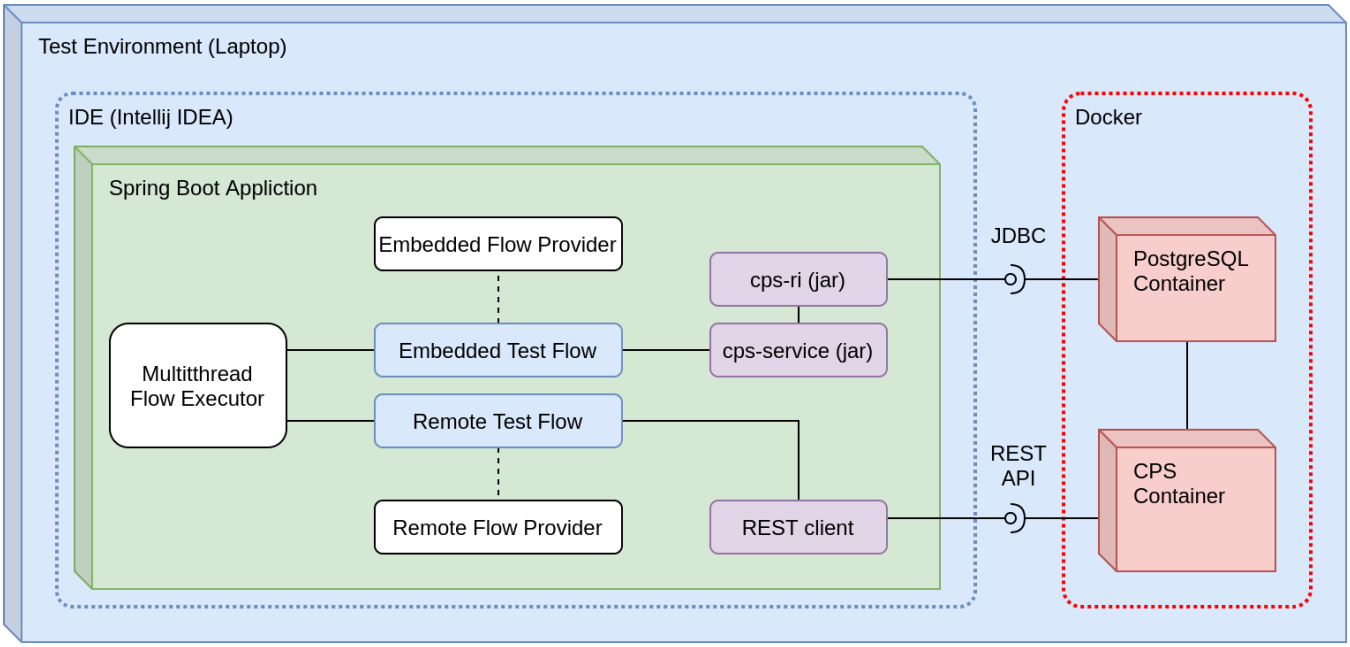
Update data:

```
// node parent xpath: /networks/network[@network-id='l2-topo-example']/node[@node-id='D3']
// replacement fragment:
{
    "ietf-l2-topology:l2-node-attributes":
        {
                    "management-address": [ "192.0.2.3", "2001:db8:0:3::" ]
                }
}
```

Environment

The load test was performed using dedicated Spring Boot Application. The application was executed directly from IDE.

Remote services were deployed as docker containers. Docker was running on same machine.



Hardware details

Laptop DELL Latitude 5500

- Processors: 8 × Intel® Core™ i7-8665U CPU @ 1.90GHz
- Memory: 15,5 GiB of RAM
- lshw > lshw.txt

Software details

OS

- Operating System: Kubuntu 20.04
- KDE Plasma Version: 5.18.5
- KDE Frameworks Version: 5.68.0
- Qt Version: 5.12.8
- Kernel Version: 5.4.0-70-generic
- OS Type: 64-bit

Docker

- Docker version 20.10.5, build 55c4c88

Test application runtime

- OpenJDK version 13.0.4, 2020-07-14
- OpenJDK Runtime Environment (build 13.0.4+8-Ubuntu-120.04)
- OpenJDK 64-Bit Server VM (build 13.0.4+8-Ubuntu-120.04, mixed mode)
- java -XX:+PrintFlagsFinal -version > java-print-flags-final.txt

Test Results

CPS revision

Tested CPS from a **master** branch at state on 12 Apr 2021
representing the **Honolulu 1.0.1** + DELETE ANCHOR functionality targeted for **Istanbul** release

Operation execution time

The operation per seconds was calculated as 1000/(exec time in millis).

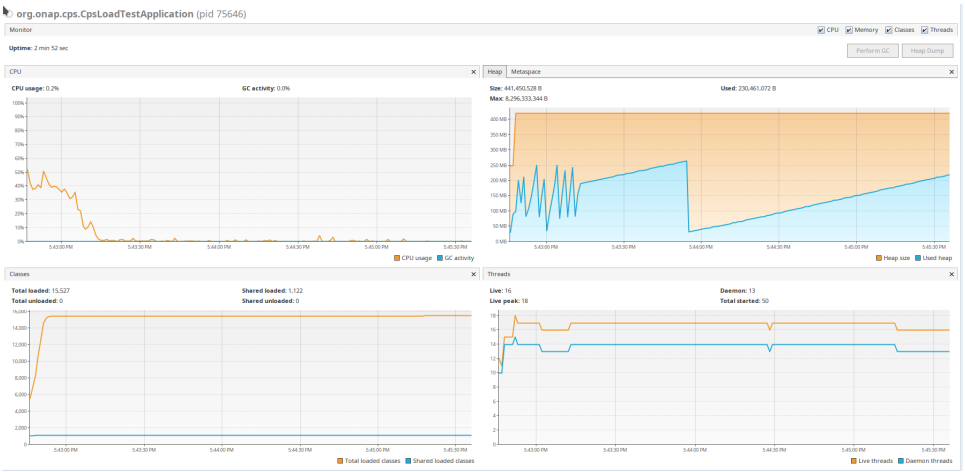
Operation	Embedded Flow						Remote Flow					
	Execution time, milliseconds			Operations per second			Execution time, milliseconds			Operations per second		
	Min	Max	**Average**	Min	Max	**Average**	Min	Max	**Average**	Min	Max	**Average**
Threads: 2 Circles: 500												
CREATE ANCHOR	0.992	29.111	**2.078**	34.351	1008.366	**481.185**	61.87	152.437	**69.08**	6.56	16.163	**14.476**
CREATE NODE	13.171	600.115	**23.898**	1.666	75.923	**41.844**	74.617	201.421	**85.499**	4.965	13.402	**11.696**
UPDATE NODE	1.92	27.538	**4.306**	36.314	520.911	**232.248**	62.098	249.319	**70.496**	4.011	16.104	**14.185**
DELETE ANCHOR	2.551	21.713	**4.353**	46.055	392.038	**229.704**	62.77	144.299	**71.053**	6.93	15.931	**14.074**
Threads: 10 Circles: 100												
CREATE_ANCHOR	1.646	78.207	**6.789**	12.787	607.464	**147.29**	66.84	326.61	**140.844**	3.062	14.961	**7.1**
CREATE_NODE	17.177	1115.065	**105.986**	0.897	58.219	**9.435**	82.275	325.542	**170.05**	3.072	12.154	**5.881**
UPDATE_NODE	3.218	115.991	**15.516**	8.621	310.788	**64.449**	63.226	256.484	**141.489**	3.899	15.816	**7.068**
DELETE_ANCHOR	3.56	53.776	**11.742**	18.596	280.904	**85.163**	64.741	411.13	**143.797**	2.432	15.446	**6.954**
Threads: 20 Circles: 50												
CREATE_ANCHOR	1.891	177.361	**28.412**	5.638	528.779	**35.196**	79.077	568.126	**297.916**	1.76	12.646	**3.357**
CREATE_NODE	20.426	1781.881	**186.942**	0.561	48.958	**5.349**	86.274	978.945	**376.145**	1.022	11.591	**2.659**
UPDATE_NODE	3.623	401.141	**86.079**	2.493	275.978	**11.617**	73.062	615.295	**297.539**	1.625	13.687	**3.361**
DELETE_ANCHOR	3.937	114.205	**18.554**	8.756	253.99	**53.896**	64.017	588.028	**299.171**	1.701	15.621	**3.343**
Threads: 50 Circles: 20												
CREATE_ANCHOR	2.498	805.037	**89.272**	1.242	400.328	**11.202**	129.312	2165.287	**788.826**	0.462	7.733	**1.268**
CREATE_NODE	41.46	3317.529	**435.917**	0.301	24.12	**2.294**	115.405	2403.888	**929.274**	0.416	8.665	**1.076**
UPDATE_NODE	6.039	1127.585	**323.545**	0.887	165.582	**3.091**	86.823	1729.146	**784.139**	0.578	11.518	**1.275**
DELETE_ANCHOR	6.412	364.367	**32.314**	2.744	155.956	**30.946**	80.536	2033.905	**785.358**	0.492	12.417	**1.273**

Resource usage

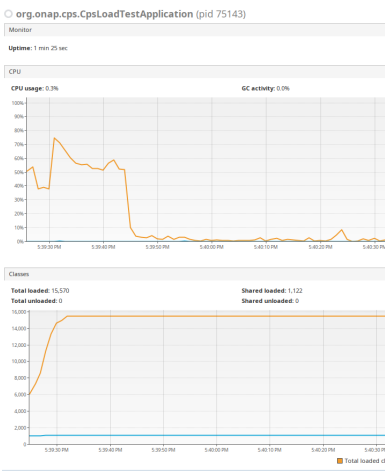
Below are resource usage diagrams (VisualVM monitor screenshots) for used Threads/Circles combinations.

Embedded then remote load tests were performed via single application execution (see test flow diagram above)

Threads: 2
Circles: 500

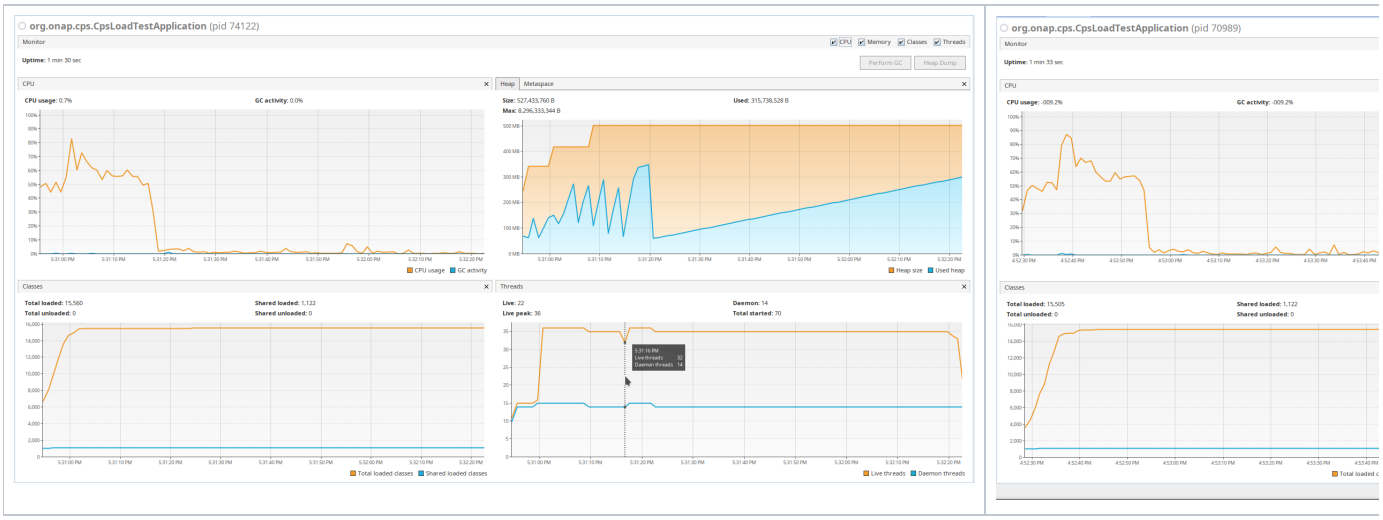


Threads: 10
Circles: 100



Threads: 20
Circles: 50

Threads: 50
Circles: 20



Following (calculated by default) heap settings were used:

```
ErgoHeapSizeLimit                = 0
HeapSizePerGCThread              = 43620760
InitialHeapSize                  = 260046848
LargePageHeapSizeThreshold       = 134217728
MaxHeapSize                      = 4148166656
NonNMethodCodeHeapSize           = 5836300
NonProfiledCodeHeapSize          = 122910970
ProfiledCodeHeapSize             = 122910970
```

Full dump of JVM flags is listed in attachment java-print-flags-final.txt

Test Application

Source

The source of test application is attached: cps-load-test-application.zip

Setting up

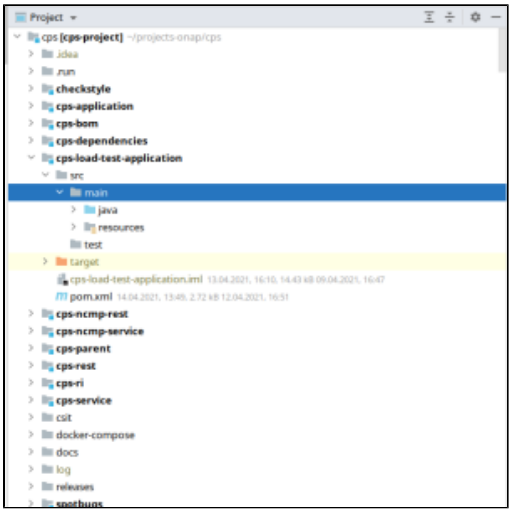
Unzip into **cps** folder.

Update root pom.xml file to include cps-load-test-application

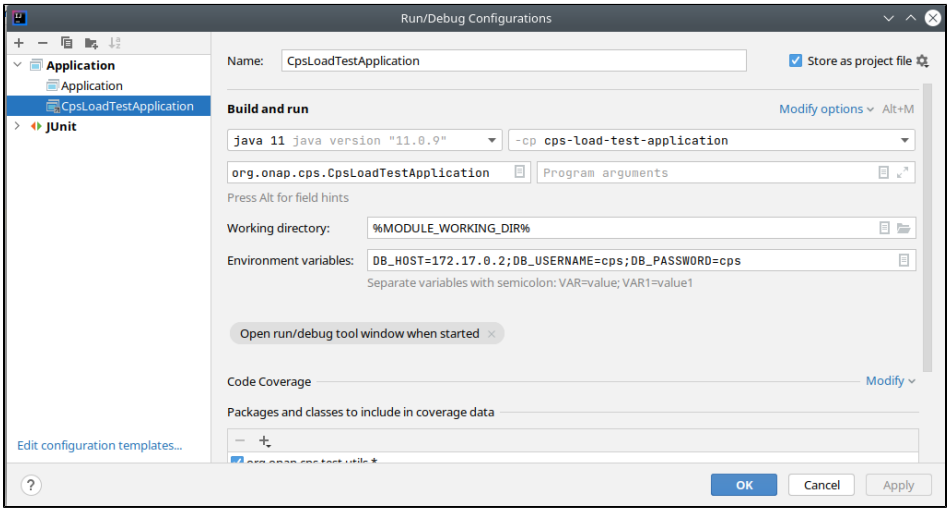
cps/pom.xml

```
...
    <modules>
        <module>cps-dependencies</module>
        <module>cps-bom</module>
        <module>cps-parent</module>
        <module>cps-service</module>
        <module>cps-rest</module>
        <module>cps-ncmp-service</module>
        <module>cps-ncmp-rest</module>
        <module>cps-ri</module>
        <module>checkstyle</module>
        <module>spotbugs</module>
        <module>cps-application</module>
        <!-- add following-line -->
        <module>cps-load-test-application</module>
    </modules>
...
```

Refresh the maven project. New module will appear as on screenshot below



Create run configuration using same environment variables as for core CPS Application



Tests related configuration is allocated within cps-load-test-application/src/main/resources/application.yml in load-test section like below

application.yml

```
load-test:
    preset:
        dataspace: test-dataspace
        schema-set: test-schema-set
        anchor-prefix: test-anchor-
    embedded:
        enabled: true
    remote:
        enabled: true
        # resources zip file is corrupted on build, so referencing original from resources
        resources-zip: src/main/resources/yang/network-topology.zip
        base-url: http://localhost:8883/cps/api/
        auth:
            username: ***
            password: ***
    # number of threads
    threads: 50
    # number of times the flow repeated within a thread
    circles: 20
    # thread termination timeout
    termination-timeout-seconds: 1200

spring:
...
```