SaaSyML

On-Board Software as a Service for Machine Learning

FINAL PRESENTATION, 2022.11.29

Cesar Guzman[1], **Georges Labrèche**[2], Miguel Lordelo[1], and **Liliana Medina**[1]

[1] VisionSpace Technologies GmBH, Darmstadt, Germany.
[2] Tanagra Space OÜ, Tallinn, Estonia.

AGENDA

1. Background and Objectives
2. Approach and Demo
3. Results
4. Summary and Outlook
5. Discussion

1. BACKGROUND AND OBJECTIVES

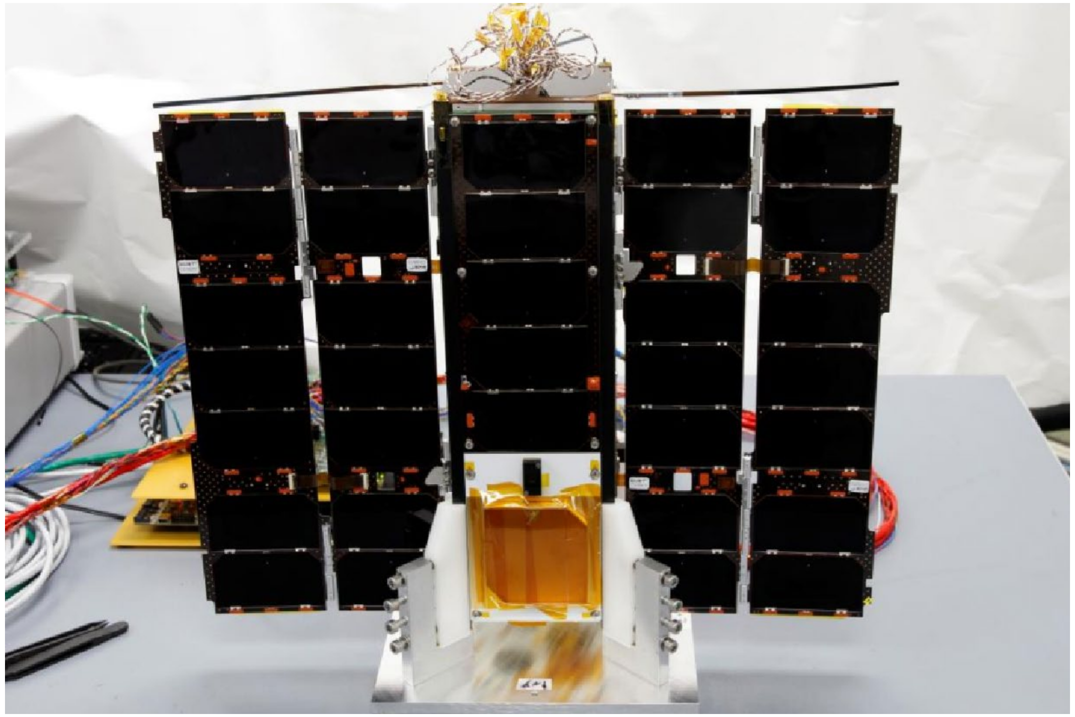
OPS-SAT SPACE LAB

- Satellite Experimental Processing Platform (SEPP)
  - Able to reuse and run open-source software

- Support for on-board apps *"easily developed, debugged, tested, deployed, and updated at any time without causing any major problem to the spacecraft"*

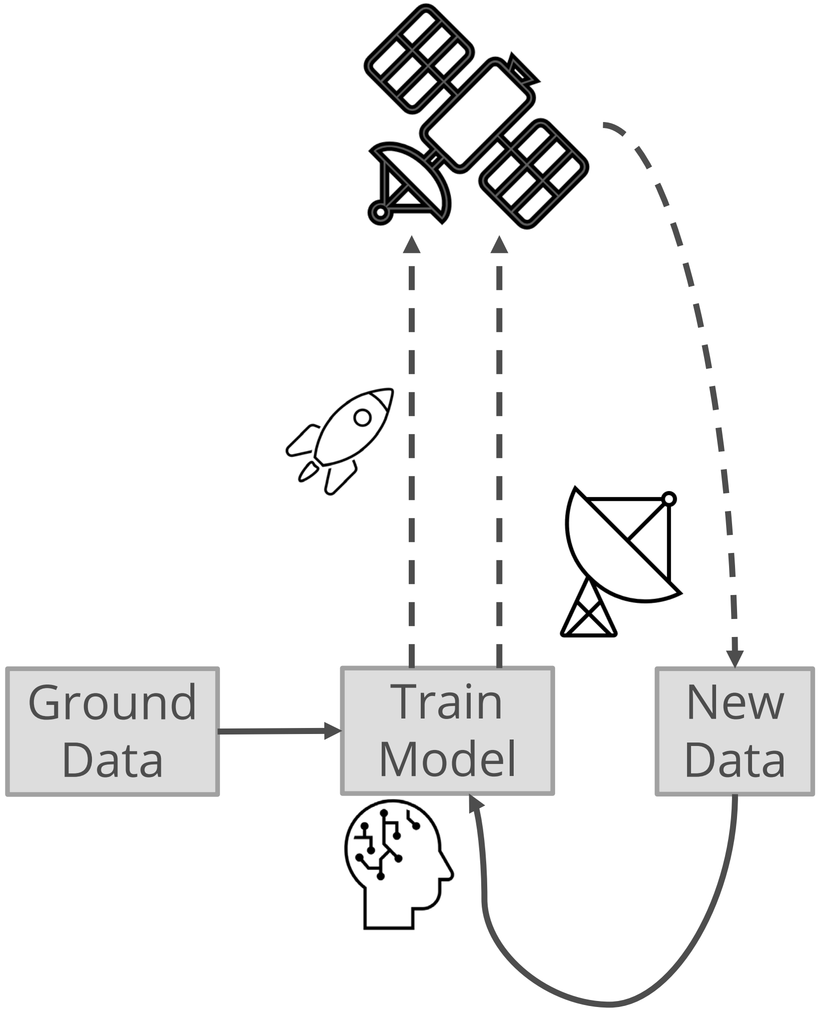
- OPS-SAT "apps in space" concept supported by the Java NanoSat MO Framework (NMF)

- OPS-SAT community platform for experimenters to develop and test their apps



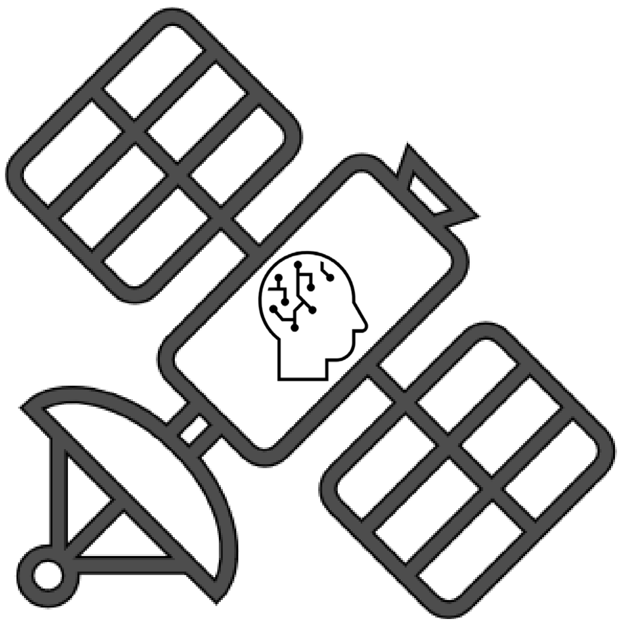
MACHINE LEARNING ON-BOARD

- ML models **typically** trained on ground
  - Models integrated with spacecraft during launch or uplinked later
  - Both approaches impose limitations on access to on-board data



MACHINE LEARNING ON-BOARD

- ML models **typically** trained on ground
  - Models integrated with spacecraft during launch or uplinked later
  - Both approaches impose limitations on access to on-board data
- **OrbitAI** experiment took a different approach
  - Training and inferencing on-board OPS-SAT
  - But ML capabilities somewhat restricted
    - Did not take full advantage of the NMF Java ecosystem
    - Hardcoded data feed configurations



A USE CASE FOR ON-BOARD ML

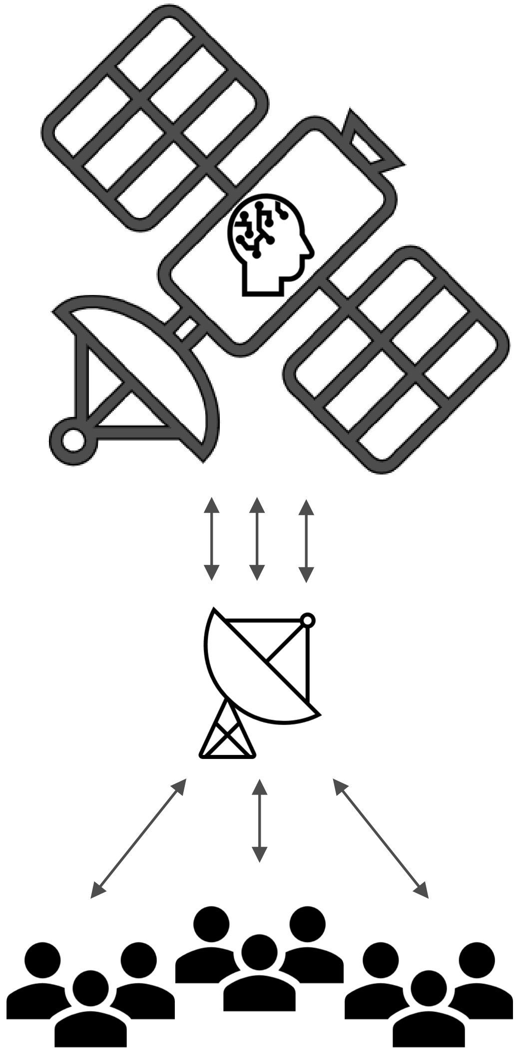
- **OrbitAI** experimented with fault detection, isolation, and recovery (FDIR) models on-board OPS-SAT

- **Goal -** poll relevant sensor data and train FDIR models to detect events which require protecting the on-board camera’s lens against exposure to sunlight.

MOTIVATION

Satellite Platform as a Service (SPaaS) app for Machine Learning

- Abstract complex data provisioning and ML operations
- Support a data subscription service to feed selected training data from any of OPS-SAT instruments
- Take advantage of SEPP’s JVM thread pool implementations and dual core processor

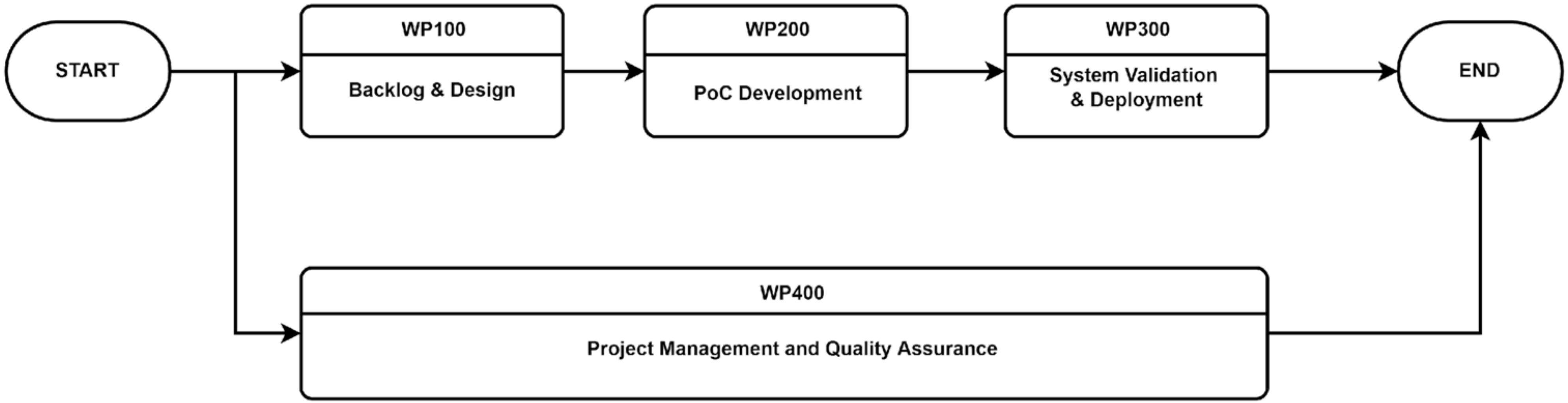


OBJECTIVES

- Provide to experimenters a **SPaaS** app for ML operations, **SaaSyML**.
- Make ML algorithms **accessible to all OPS-SAT experimenters** through SaaSyML.
- Demonstrate how to interact with SaaSyML via an **API**.
- Demonstrate a **use case** for autonomously training and deploying on-board ML models.
- Disseminate knowledge through **publication in journal or conference proceedings.**

SCHEDULE

11th of April 2022 → **28th of November 2022**

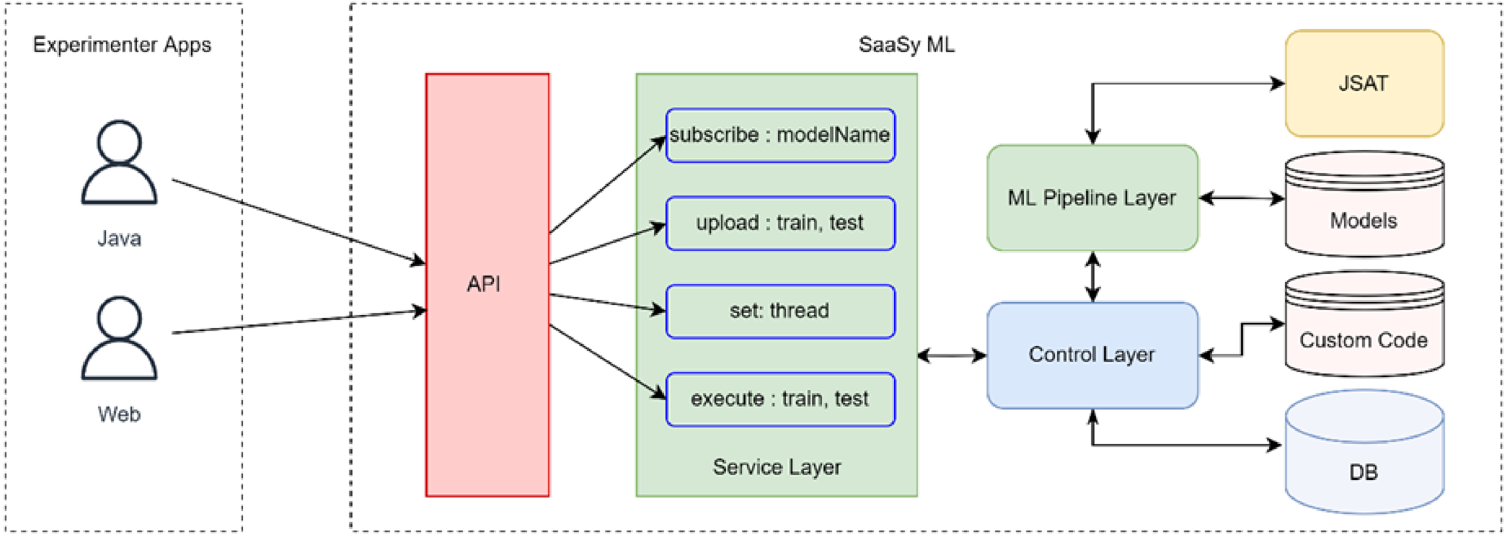


2. APPROACH AND DEMO

SOFTWARE STACK

- SaaSyML developed with open-source libraries as a **NMF** app
- **Eclipse Vert.x** toolkit: enables non-blocking ML operations in parallel for multiple app users
- **JSAT**: Java library supporting ML algorithms for different tasks
- **SQLite**: database engine to store both datasets and metadata of trained models
- Plugin Framework for Java (**PF4J**): allows experimenters to inject custom code via plugins to compute labels for supervised model training

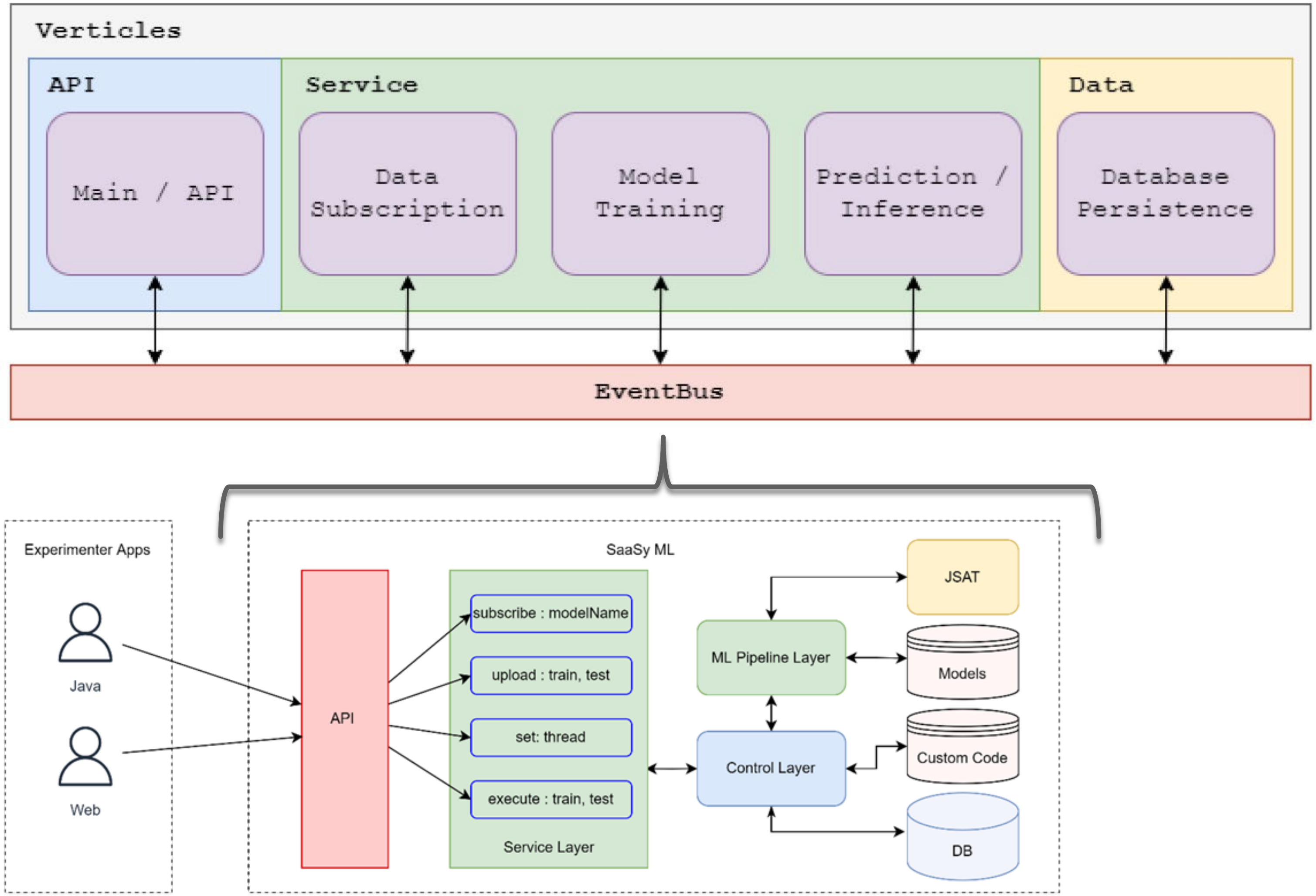
ARCHITECTURE



- User app uses API endpoints to send requests to SaaSyML **(Service Layer)**
- User subscribes real operational data from OPS-SAT **(Control Layer)**
- User may use custom plugin to compute data labels **(Control Layer)**
- User feeds data sets into predefined ML algorithms for training/inferencing **(ML Pipeline Layer)**

ARCHITECTURE

Verticle - scalable chunk of code that gets deployed and run



API ENDPOINTS AND REQUESTS

- API allows users to send requests and receive responses to/from SaaSyML
- Endpoints support multiple operations
  - Subscribe to a data feed
  - Use a custom plugin to compute label values
  - Train models on subscribed data
  - Get model metadata
  - Make inferences using a saved model on new data points

API USE: SIMPLE ML FLOW

1. **Subscribe to data**
2. Train models
3. Fetch model metadata
4. Inference using new data

POST http://<HOST>:<PORT>/api/v1/training/data/subscribe

Custom values


```
{
    "expId": 123,
    "datasetId": 1,
    "iterations": 10,
    "interval": 2,
    "labelsPlugin": "esa.mo.nmf.apps.saasyml.plugins.CameraStateLabels",
    "params": ["CADC0884","CADC0886","CADC0888","CADC0890","CADC0892","CADC0894"]
}
```


RESPONSE


```
{
    "response": "Successfully subscribed to training data feed."
}
```

API USE: SIMPLE ML FLOW

1. Subscribe to data
2. **Train models**
3. Fetch model metadata
4. Inference using new data

POST http://<HOST>:<PORT>/api/v1/training/regressor


```
{
    "expId": 123,
    "datasetId": 1,
    "algorithm": "MultipleLinearRegression"
}
```


RESPONSE


```
{
    "response": "Training the model(s) has been triggered. Query the /api/v1/download/models endpoint for training status."
}
```

API USE: SIMPLE ML FLOW

1. Subscribe to data
2. Train models
3. **Fetch model metadata**
4. Inference using new data

POST http://<HOST>:<PORT>/api/v1/download/models


```
{
    "expId": 123,
    "datasetId": 1,
    "formatToInference": true
}
```


RESPONSE


```
{
    "response": {
        "expId": 123,
        "models": [
            {
                "type": "Regressor",
                "filepath": "./models/E123-D1-MultipleLinearRegression-2022-11-18_09-39-41.model"
            }
        ]
    }
}
```

API USE: SIMPLE ML FLOW

1. Subscribe to data

2. Train models

3. Fetch model metadata

4. **Inference using new data**

POST `http://<HOST>:<PORT>/api/v1/inference`

```
{
    "expId": 123,
    "datasetId": 1,
    "data": [···
    ],
    "models": [
        {
            "filepath": "./models/E123-D1-MultipleLinearRegression-2022-11-18_09-39-41.model",
            "type": "Regressor"
        }
    ]
}
```

RESPONSE

```
{
    "expId": 123,
    "models": [
        {
            "filepath": "./models/E123-D1-MultipleLinearRegression-2022-11-18_09-39-41.model",
            "type": "Regressor",
            "inference": [
                1.2653721682847898,
                -0.2949029126213589
            ]
        }
    ]
}
```

API USE: SIMPLE ML FLOW

1. Subscribe to data
2. Train models
3. Fetch model metadata
4. **Inference using new data**

POST http://<HOST>:<PORT>/api/v1/inference


```json
{
    "expId": 123,
    "datasetId": 1,
    "data": [...
    ],
    "models":
    [
        {
            "type": "Regressor",
            "filepath": "./models/E123-D1-StochasticRidgeRegression-2022-11-24_05-18-52.model"
        },
        {
            "type": "Regressor",
            "filepath": "./models/E123-D1-StochasticGradientBoosting-2022-11-24_05-18-49.model"
        },
        {
            "type": "Regressor",
            "filepath": "./models/E123-D1-MultipleLinearRegression-2022-11-24_05-18-47.model"
        }
    ]
}
```


RESPONSE

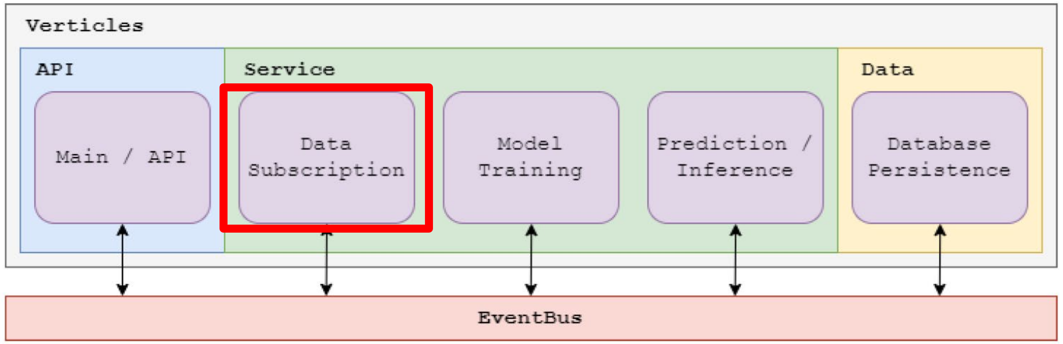

```json
{
    "expId": 123,
    "models": [
        {
            "type": "Regressor",
            "filepath": "./models/E123-D1-StochasticRidgeRegression-2022-11-24_05-18-52.model",
            "inference": [
                0.20227459102966386,
                0.2979326875832567
            ]
        },
        {
            "type": "Regressor",
            "filepath": "./models/E123-D1-StochasticGradientBoosting-2022-11-24_05-18-49.model",
            "inference": [
                0.3761502127457619,
                0.11542686656067
            ]
        },
        {
            "type": "Regressor",
            "filepath": "./models/E123-D1-MultipleLinearRegression-2022-11-24_05-18-47.model",
            "inference": [
                1.2653721682847898,
                -0.2949029126213589
            ]
        }
    ]
}
```

DEMO

- Run app locally
- Demonstrate simple use case
  - Save data with custom labels from plugin
  - Train classifier models
  - Get model metadata
  - Use trained models for inference

4. RESULTS

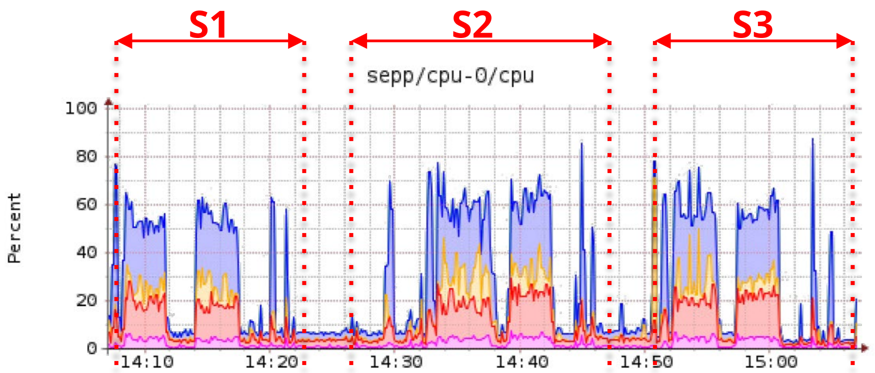
EM - TEST CASE 1



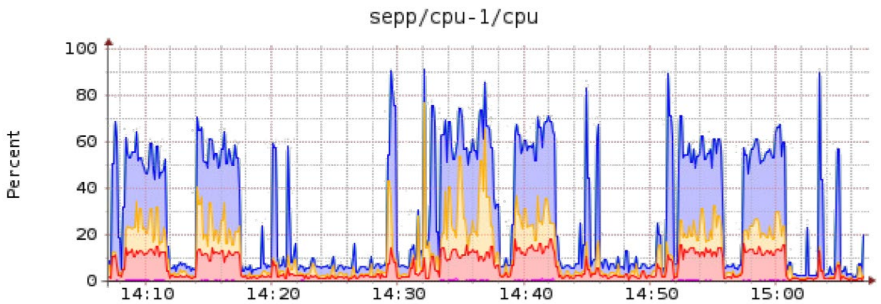
- EM session 12/10/22
- 3 scenarios: increasing thread counts for the “Data Subscription” verticle (1, 5, and 10)
  - 10 parallel requests are executed representing 10 separate users
- Each training data feed fetches data for 5 data pool parameters
- Testing the whole ML loop (fetch data, train, inference) for 6 classification algorithms

EM RESULTS

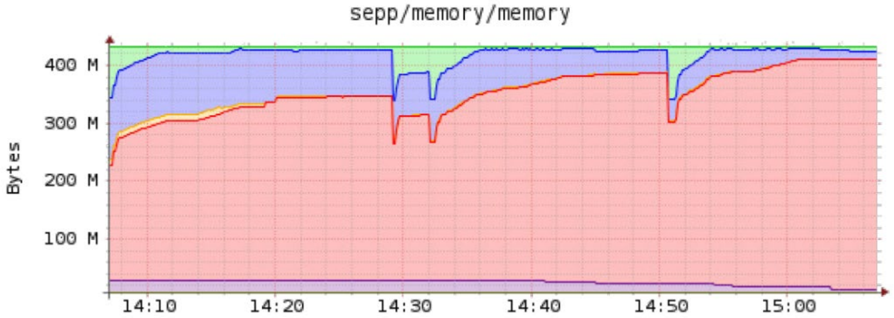
Data Subscription Verticle Counts
S1: 1 instance
S2: 5 instances
S3: 10 instances



idle	12.5 Min,	70.7 Avg,	97.4 Max,	79.1 Last
nice	0.0 Min,	0.0 Avg,	0.0 Max,	0.0 Last
user	0.8 Min,	15.2 Avg,	65.9 Max,	10.0 Last
wait	0.0 Min,	3.8 Avg,	59.5 Max,	5.6 Last
system	0.8 Min,	8.1 Avg,	22.7 Max,	4.1 Last
softirq	0.2 Min,	2.3 Avg,	7.7 Max,	1.2 Last
interrupt	0.0 Min,	0.0 Avg,	0.0 Max,	0.0 Last
steal	0.0 Min,	0.0 Avg,	0.0 Max,	0.0 Last

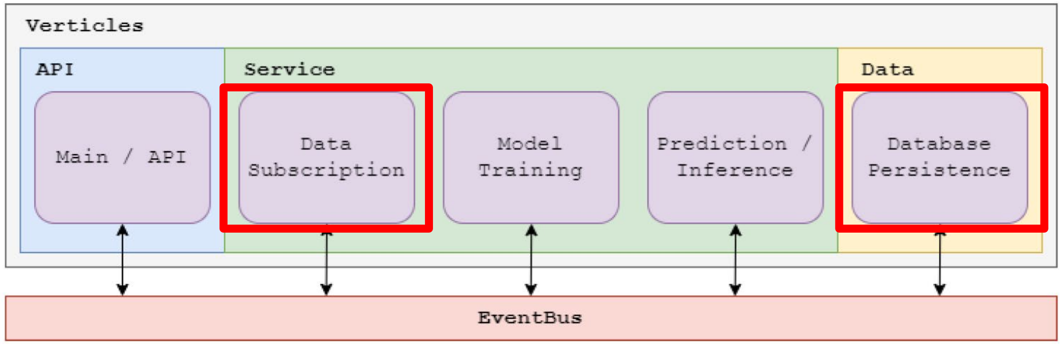


idle	8.6 Min,	70.3 Avg,	98.3 Max,	80.2 Last
nice	0.0 Min,	0.0 Avg,	0.0 Max,	0.0 Last
user	1.0 Min,	17.5 Avg,	78.0 Max,	17.4 Last
wait	0.0 Min,	6.0 Avg,	67.1 Max,	0.6 Last
system	0.4 Min,	6.1 Avg,	17.5 Max,	1.8 Last
softirq	0.0 Min,	0.2 Avg,	1.2 Max,	0.0 Last
interrupt	0.0 Min,	0.0 Avg,	0.0 Max,	0.0 Last
steal	0.0 Min,	0.0 Avg,	0.0 Max,	0.0 Last



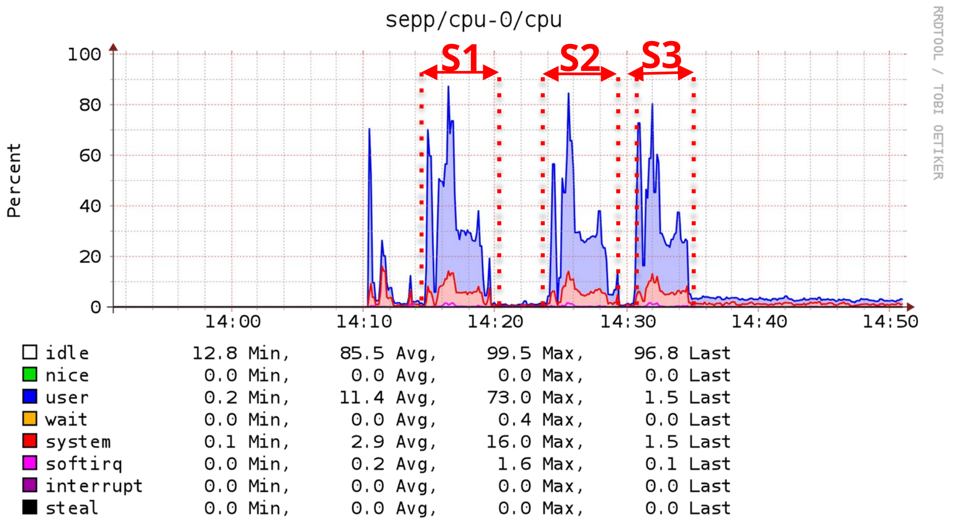
free	3.3M Min,	13.7M Avg,	94.0M Max,	7.3M Last
cached	12.9M Min,	62.1M Avg,	109.1M Max,	13.2M Last
buffered	176.0k Min,	3.0M Avg,	11.2M Max,	1.4M Last
used	198.5M Min,	327.4M Avg,	396.7M Max,	396.7M Last
slab_recl	5.3M Min,	17.8M Avg,	22.3M Max,	5.3M Last
slab_unrecl	7.6M Min,	7.9M Avg,	8.3M Max,	7.9M Last

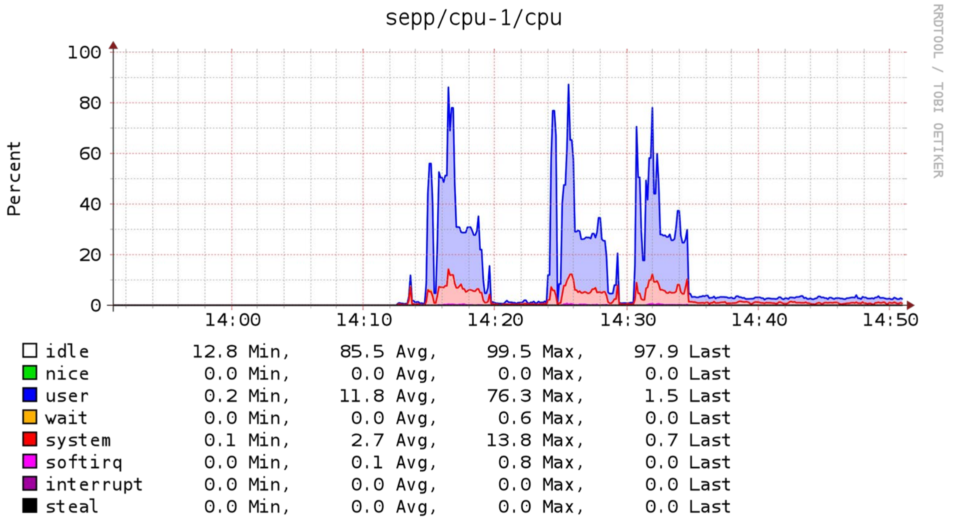
EM - TEST CASE 2



- EM session 21/11/22
- Similar 3 scenarios to test case 1 but “Database Persistence” verticle count also increases

EM RESULTS





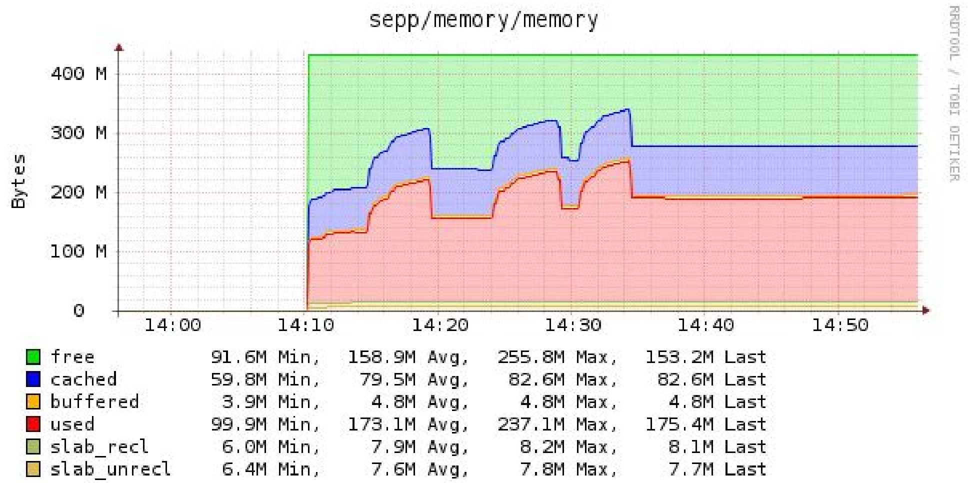
Data Subscription Verticle Counts

S1: 1 instance
S2: 5 instances
S3: 10 instances

Database Persistence Verticle Counts

S1: 1 instance
S2: 5 instances
S3: 10 instances

sepp



FDIR USE CASE RESULTS

- **OrbitAI experiment successfully replicated locally**
- 8 binary classifier models trained on 1190 dataset records
  - Expected label values calculated with a custom plugin
    - If value from sensor < 1.0472, then label is set to 1
    - Otherwise, label is 0
- Inference on new data points using trained models
  - Predictions match expected label values

5. SUMMARY AND OUTLOOK

[redacted]

SUMMARY

- SaaSyML technology demonstrator for **'Satellite Platform as a Service' (SPaaS).**
  - data provisioning service, plugin module for user-defined logic, integration with a ML library, and a service interface
- EM sessions demonstrated how to interact with SaaSyML via an API
- Exhaustive tests of entire ML loop (fetch data, train, inference) in multi-thread scenario suggest app scales without increase in resource utilization
- Code repository: https://github.com/visionspacetec/opssat-saasy-ml
- Paper accepted for peer-reviewed publication at IEEE Aerospace Conference 2023

OUTLOOK

- SaaSyML deployment on-board OPS-SAT for FDIR use case
  - Possibility of using SaaSyML for outlier detection use case
- Invite OPS-SAT experimenters to use SaaSyML
- Enhancements
  - Inference feed subscription (**ongoing**)
  - Plugin-based algorithm integration
  - Extend training requests to customize algorithm parameters
  - Evaluation of supervised models

5. DISCUSSION

THANK YOU!

VisionSpace Technologies GmbH
Robert-Bosch-Strasse 7
64293 Darmstadt
Germany

https://www.visionspace.com/

https://tanagraspace.com/