Real-time flow analysis using OSS based distributed infrastructure

OSSで実現する分散基盤を使ったリアルタイムflow分析

Shubham Saha Aakash Nand Abhik Datta Banik Oya Yu Asano Shuhei
NTT Communications

Agenda

- Self Introduction
- Background
- Open Source Stack
- Anomaly detection
- Infrastructure and Architecture
- Demo
- Future Works
- Discussion

Introductions

Self-introduction

Team Kaminari

- A team of network engineers, data engineers, and data scientists for analyzing backbone network traffic.
- We are developing the application of anomaly detection (DDoS etc) using ML and distributed infrastructure technology.
- About 1.5 years since team started

Background

Background

DDoS attacks

- Number, scale, and severity of the impact of DDoS attacks.
- IoT devices and 5G fuel Botnets.

NTT Communications (ISP and ICT solutions provider) : monitor security threats in the network.

NEED OF THE HOUR

- Intelligent, managed DDoS protection solutions
- **distributed infrastructure** X **deep learning** for enhanced real-time efficiency.

Open Source Stack

Open Source Tech Stack

pmacct is a small set of multi-purpose passive network monitoring tools

Apache Kafka is a stream-processing software platform capable of handling trillions of events a day.

Apache Hive is a data warehouse built on top of Apache Hadoop for providing data query and analysis.

Apache Spark is an distributed general-purpose cluster-computing framework.

Kubernetes is a system for managing containerized applications across a cluster of nodes.

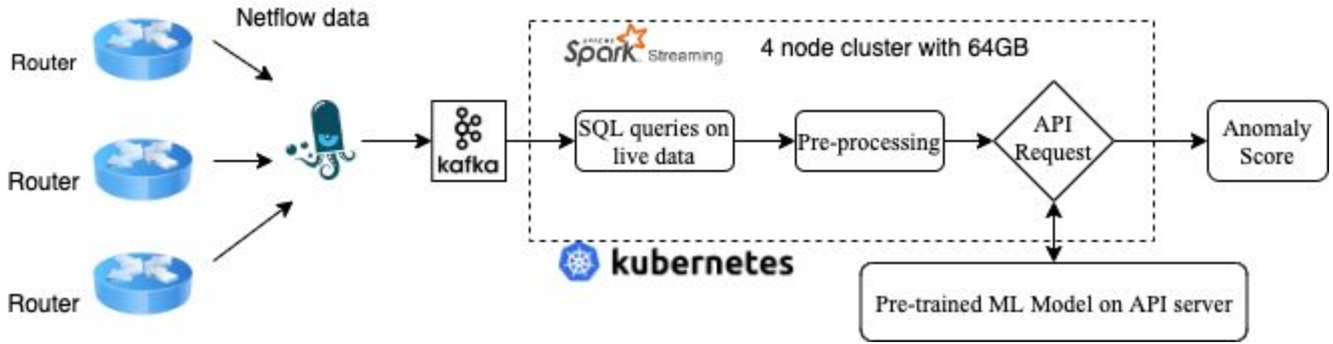
Infrastructure and Architecture

Architecture overview

Flow collection

Preprocessing

Machine Learing



Netflow data collection

pmacct

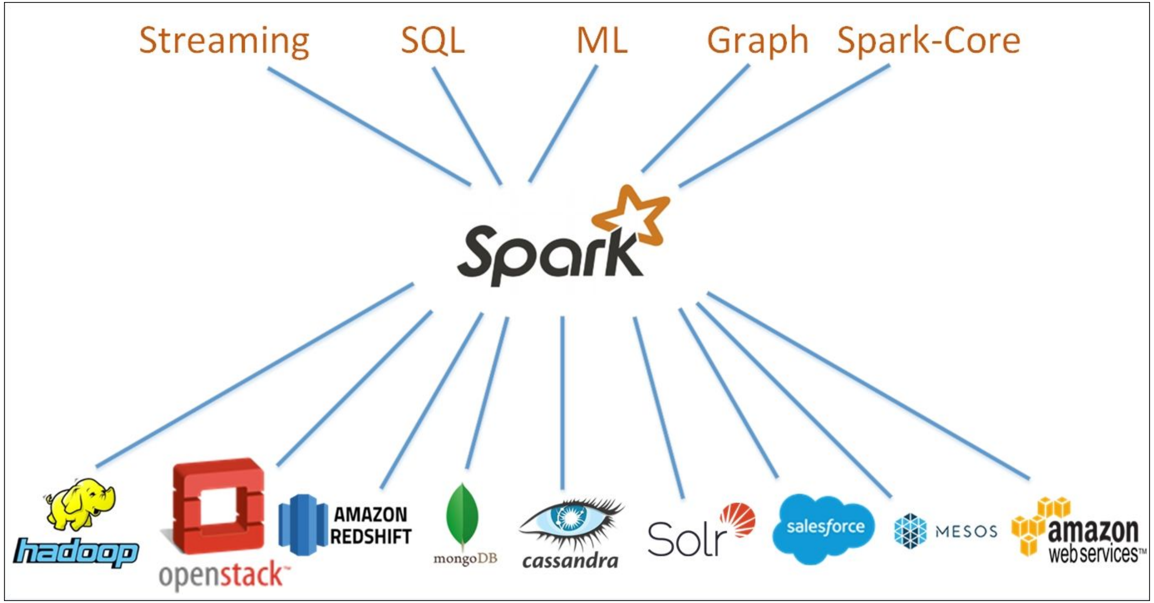
- Passive network monitoring tool for network data collection
- Streaming telemetry
- Collects data through libpcap, Netlink/NFLOG, NetFlow v1/v5/v7/v8/v9, sFlow v2/v4/v5 and IPFIX
- Saves data to a number of backends including:
    - Relational databases: MySQL, PostgreSQL and SQLite
    - noSQL databases: MongoDB and BerkeleyDB
    - AMQP message exchanges: RabbitMQ
    - Kafka message brokers
    - memory tables
    - flat files

http://www.pmacct.net/

What is Spark?

- Apache Spark is a open source lightning-fast unified analytics engine for big data and machine learning

- It is an optimized engine that supports general computation graphs for data analysis.



What is k8s?

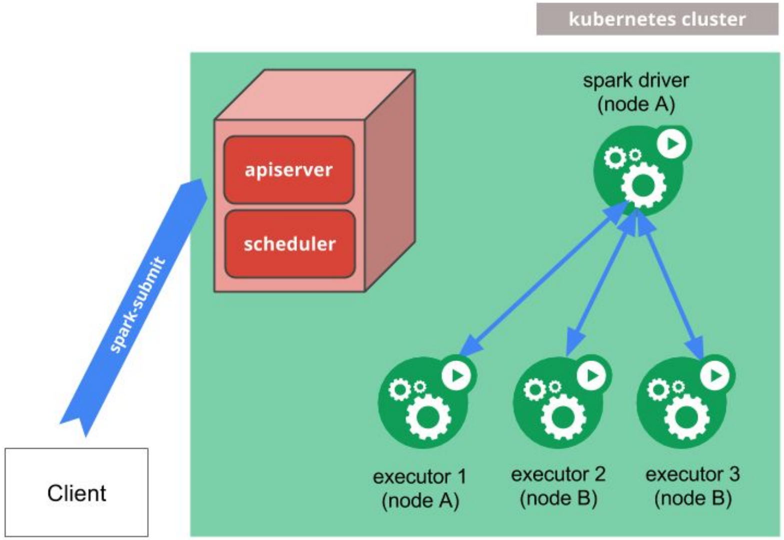
Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

kubernetes

Why Spark?

- Speed
- Real-Time
- Deployment
- Native support for Machine Learning:
- Works well with Big Data platforms

Spark on k8s



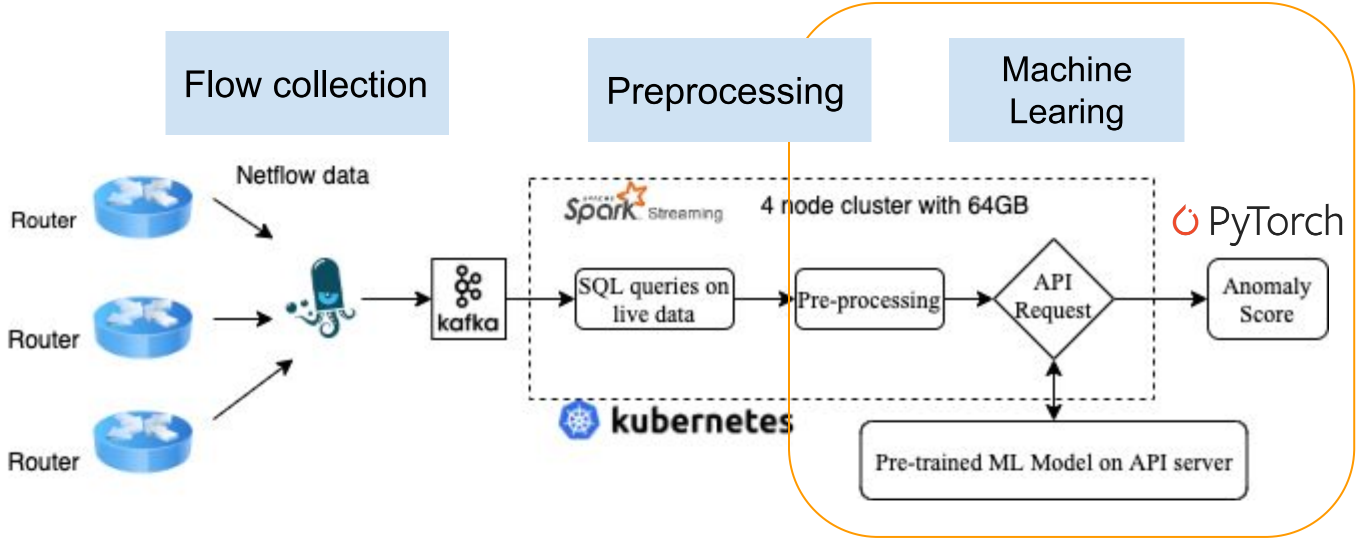

```
./bin/spark-submit \
--master k8s://https://[redacted]:6443 \
--deploy-mode cluster
--name predict-on-spark \
--jars
[redacted]/user/aakashnand/spark-streaming-kafka-
0-8-assembly_2.11-2.4.6.jar \
--conf spark.executor.instances=10 \
--conf
spark.kubernetes.container.image=[redacted]/test:latest \
--conf spark.kubernetes.authenticate.driver.serviceAccountName=spark
hdfs://[redacted]:8020/tmp/asano/predict-on-spark.py
```

Why Spark on Kubernetes (k8s)

	Spark on Yarn	Spark on k8s
Dependency management	Poor ❌	Good 🟢
Admin Overhead	High ❌	Low 🟢
Spark Version management	Rigid ❌	Flexible 🟢
Container Customization	Poor ❌	Good 🟢
Resource Allocator	Yarn Resource Manager	k8s Scheduler API
Spark application management	Convenient 🟢	Difficult ❌
Learning Curve	fast 🟢	slow ❌

Anomaly Detection

Architecture overview



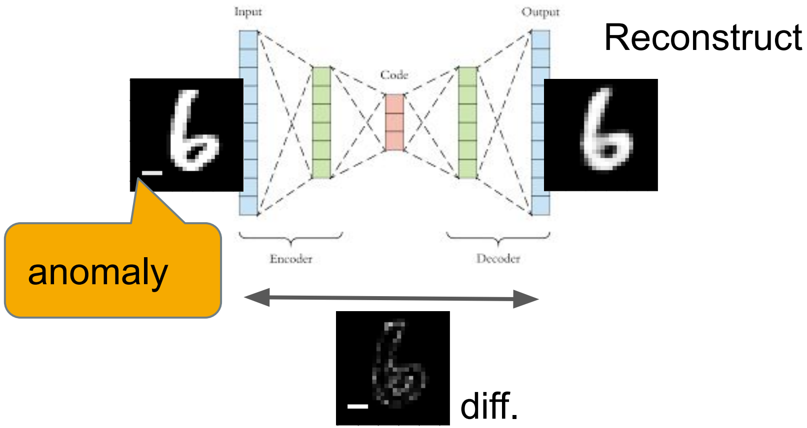
What is Torch?

- Torch is an open source machine learning library.

PyTorch

- A number of pieces of Deep Learning software are built on top of Torch.

Very powerful
(High accuracy)



Very useful
(Low cost of dev.)

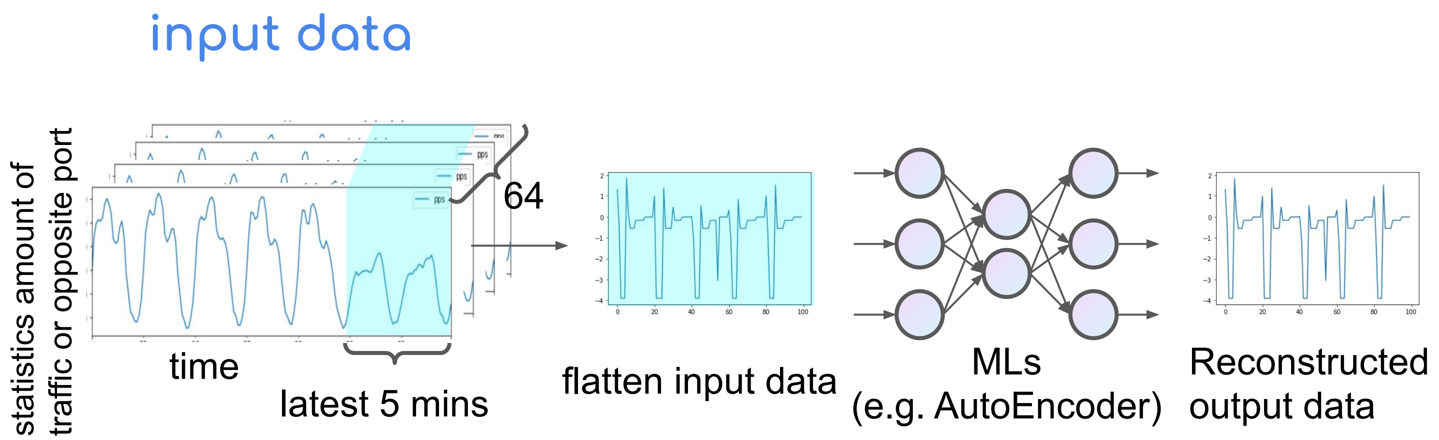
```
import torch
import torch.nn as nn
def preprocessing(x):
    return
class autoencoder(nn.Module):
    def __init__(self, inunit=320, unit=20, mean=0):
        super(autoencoder, self).__init__()
        self.preprocessing = lambda x: x.add(1).log().add(mean)
        self.encoder = nn.Sequential(
            nn.Linear(inunit, unit),
            nn.Sigmoid(),
            nn.Linear(unit, unit),
            nn.Sigmoid())
        self.decoder = nn.Sequential(
            nn.Linear(unit, unit),
            nn.Sigmoid(),
            nn.Linear(unit, inunit))

    def forward(self, x):
        x = self.preprocessing(x)
        z = self.encoder(x)
        z = self.decoder(z)
        z = nn.functional.mse_loss(x, z, reduction='none')
        if z.dim() == 1:
            z = torch.mean(z)
        else:
            z = torch.mean(z, dim=1, keepdim=True)
        return z
```

Detection model overview

- Preprocessing

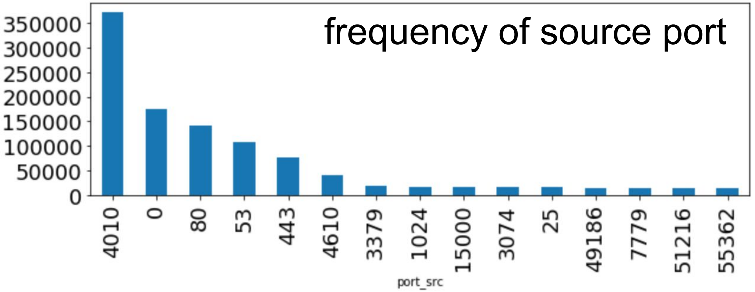
  5 tuples --> counting data --> input data (vector)

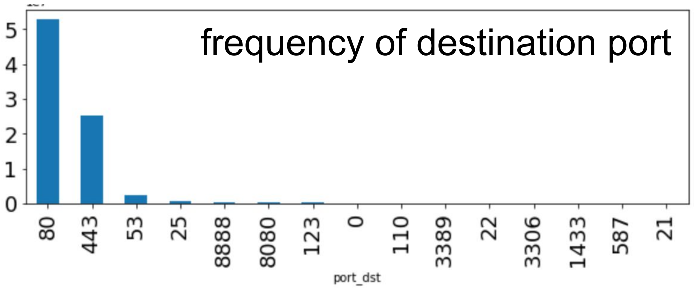
- Training and Prediction (per 5 mins.)



Netflow data preprocessing

- To select 15 + 15 ports to avoid sparse array

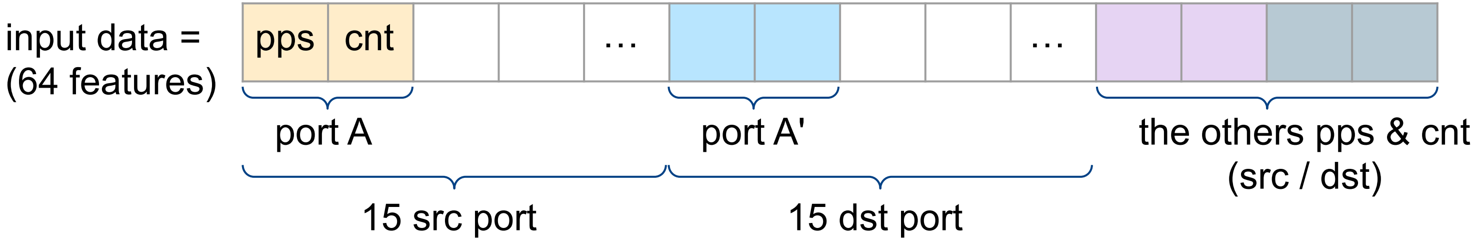




src_port	usage
4010	Samsung Unidex ?
0	Fragment ?
80	HTTP
53	DNS
443	HTTPS
4610	QualiSystems
3379	SOCORFS
1024	?
15000	psyBNC
3074	Xbox LIVE
25	SMTP
49186	cisco dpe port ?
7779	trojan ? botnet C&C
51216	?
55362	?

dst_port	usage
80	http
443	https
53	DNS
25	SMTP
8888	HTTP alternate
8080	HTTP alternate
123	NTP
0	Framgment ?
110	POP3
3389	RDP (microsoft)
22	ssh
3306	MySQL
1433	MSSQL
587	SMTP
21	IFTP

- Vectorization of traffic (pps) and number of opposite ports (count)



Demo

Pyspark sample code

```
from pyspark.streaming.kafka import KafkaUtils
from pyspark.streaming import StreamingContext
from pyspark.sql import Row
import json
from datetime import datetime

ssc = StreamingContext(sc, 60)  #60秒間隔
kafkaStream = KafkaUtils.createStream(ssc, KAFKA_ZOOKEEPER_IP ,
'test_id', {TOPIC_NAME:1})
lines = kafkaStream \
        .map(lambda x: json.loads(x[1])) \
        .map(lambda x: Row(
            timestamp=datetime.strptime(x["timestamp_arrival"],
'%Y-%m-%dT%H:%M:%S.%f%z').timestamp(),
            port_src=x["port_src"],
            packets=x["packets"]
        ))
lines.pprint(num=5)
ssc.start()
ssc.awaitTermination()
```

- Input flow from Kafka

```
-------------------------------------------
Time: 2020-08-26 19:29:00
-------------------------------------------
Row(packets=1, port_src=56108, timestamp=1598437642.0)
Row(packets=1, port_src=80, timestamp=1598437642.0)
Row(packets=1, port_src=11507, timestamp=1598437642.0)
Row(packets=1, port_src=28400, timestamp=1598437642.0)
Row(packets=1, port_src=45393, timestamp=1598437642.0)
```

Pyspark sample code

Preprocessing flow

```
lines.foreachRDD(process)

def process(lines):
    ...
    df_dst = df.groupby("timestamp") \
                .pivot("port_dst") \
                .agg(F.sum("packets").alias("pps_src"),
                     F.count("port_src").alias("num_src"))
    ...
```

You can use SQL/pandas like functions to filter, aggregate, groupby etc.

Pyspark sample code

```
def process(lines):
    ...
    AE_scores= AE_API(features)

def AE_API(x):
    ...
    result = requests.post("http://****", data=pickle.dumps(x))
    return result
```

```
def process(lines):
    ...
    outputs= np.concatenate([features,AE_scores,AR_scores], 1)
    VIS_API(outputs)

def VIS_API(lines)
    ...
    requests.post("http://****", data=pickle.dumps(x))
```

- Post preprocessed data to API running on flask
- AE_API returns anomaly score by Auto Encoder
- VIS_API stores and visualize result

Future Work

Future work

Our vision is to make anomaly detection not only easy to use but also universal.

Today we have demonstrated highly accurate DDoS detection using machine learning.

In future we would like to

- enhance the deployment and operation of the system using CI/CD
- extend this capability to other anomalies in networks and other infrastructures.
  - using various data sources
  - custom algorithms for different data source types

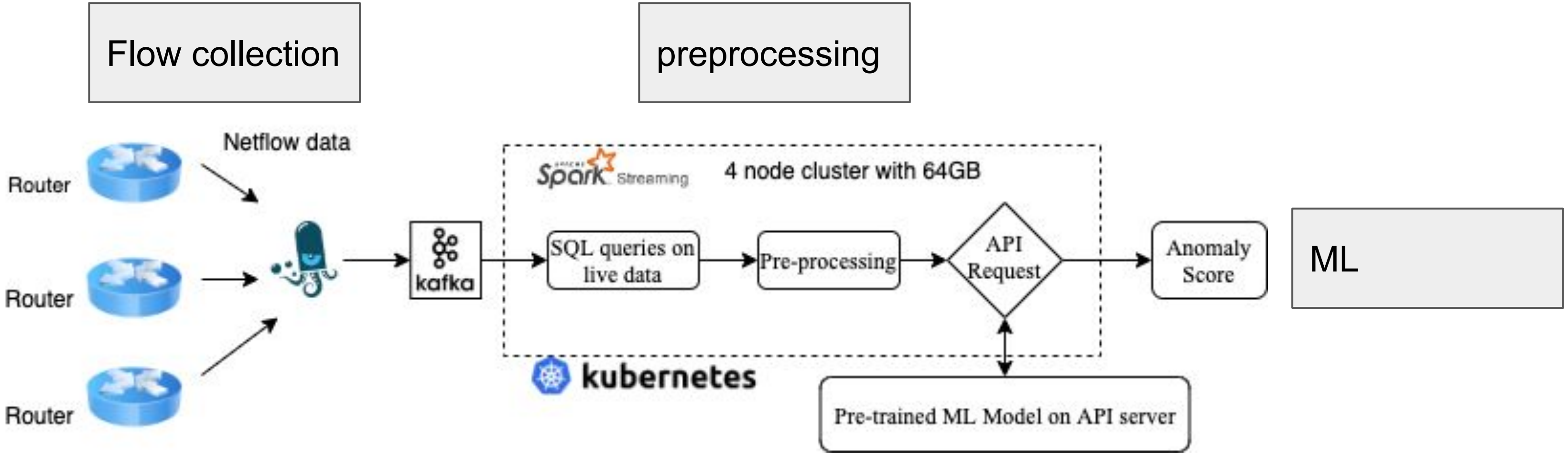
Disscussion

- How have you treated large volume traffic logs on your network?

- What else can be realized using
  Netflow × Distributed Computing × Machine Learning?

- How to input netflow into machine learning model?
- Which features do you think should be selected (ports, protocols…)?

Questions?



Flow collection
preprocessing
Netflow data
Router
Router
Router
kafka
Spark Streaming
4 node cluster with 64GB
SQL queries on live data
Pre-processing
API Request
Anomaly Score
ML
kubernetes
Pre-trained ML Model on API server