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 ~

字幕スクリプト

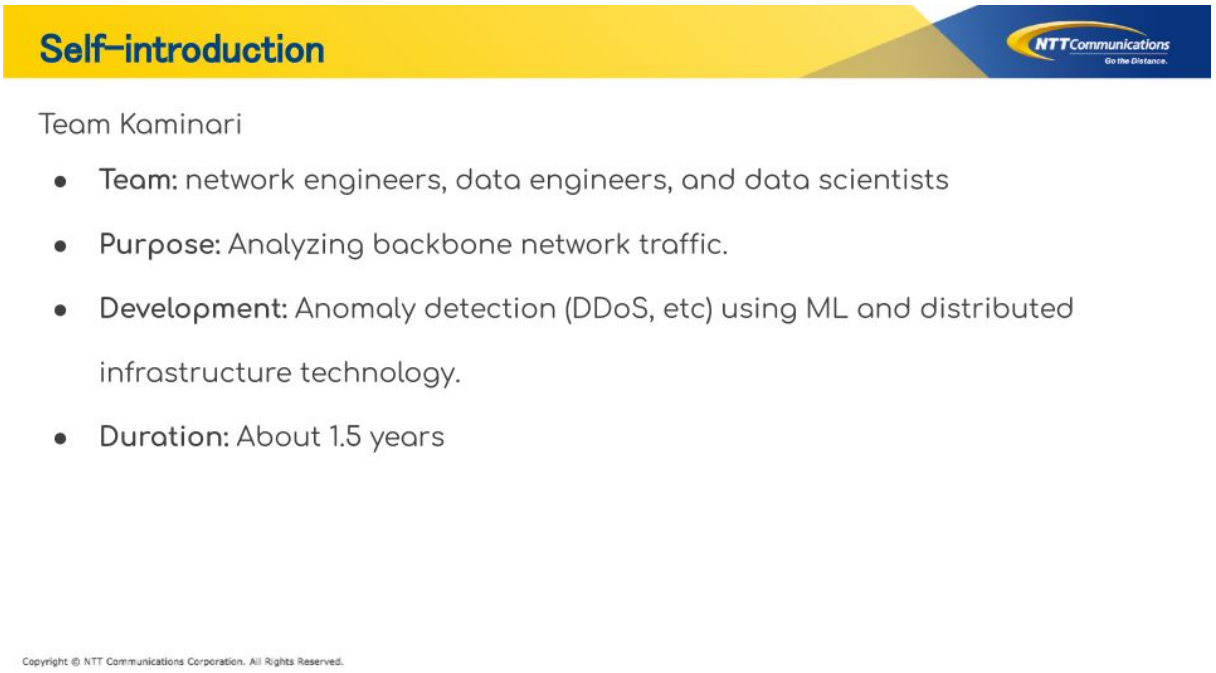
Page 2

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

The flow of the talk will be as follows.
We will cover the origin of the team and the project first.
Next we will dive right into the technologies we have used and how we have used it.
We will touch upon machine learning, distributed computing and much more.
Finally we will close with a demo, and discuss, with all of you, where do you think we should go from here.

話の流れは以下のようになります。
まず、私たちのチームの成り立ちとプロジェクトについて説明します
次に、これまでに使ってきたOSS技術とその使い方について
機械学習や分散コンピューティングなどについても触れていきます
最後にデモで締めくくり、皆さんと一緒に今後の方向性を考えてみましょう

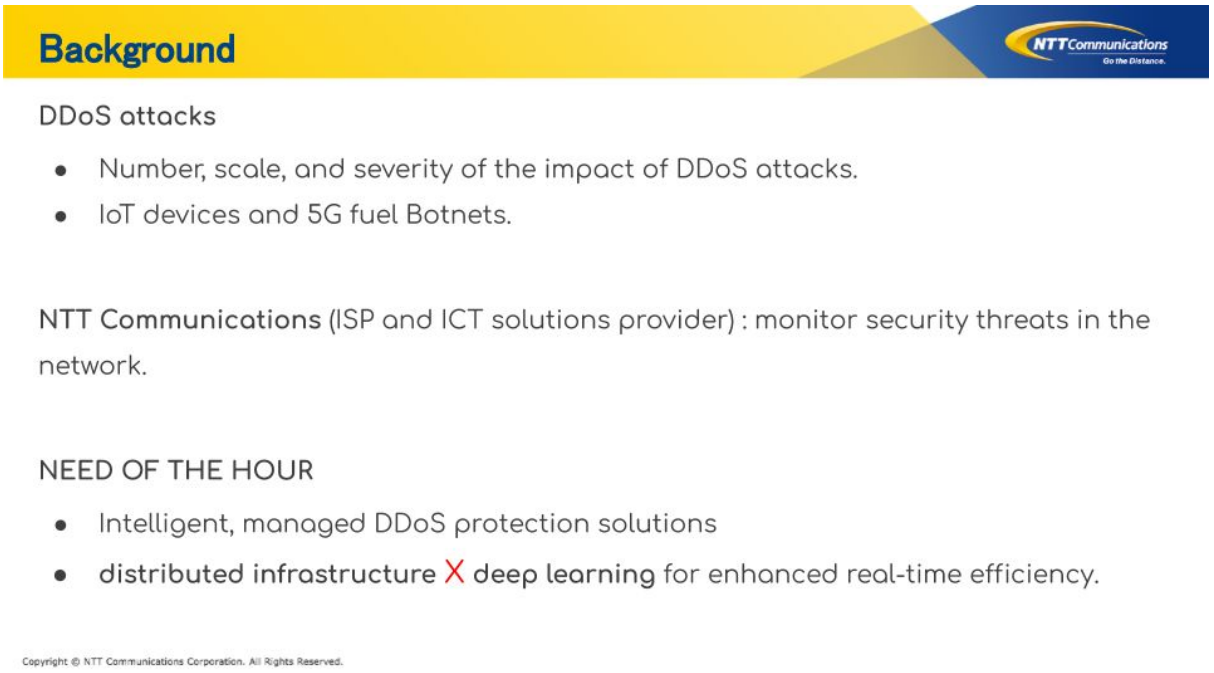
Page4



The name of our team is Kaminari
The team comprises network engineers, data engineers, and data scientists who are responsible for analyzing backbone network traffic.
We are developing an application of anomaly detection (particularly DDoS) using ML and distributed infrastructure technology.
About 1.5 years since team started

私たちは社内でKaminariという名前のチームで活動しています
ネットワークエンジニア、データエンジニア、データサイエンティストが集まったチームです
バックボーンネットワークのトラフィックを分析するのが目的で、
MLや分散型インフラ技術を用いた異常検知（特にDDoS）の応用開発を行っています
チーム発足から約1年半ほどです

Page6



There has been a sharp rise in number, scale, and severity of the impact of DDoS attacks. In the past 2 years itself, the scale and severity of their impact have risen by nearly 200%. Increasing number of IoT devices and 5G fuel Botnets will drive DDoS attacks even further in 2020.

NTT Communications is uniquely positioned as an ISP and also as a ICT solutions provider to monitor security threats in the network.

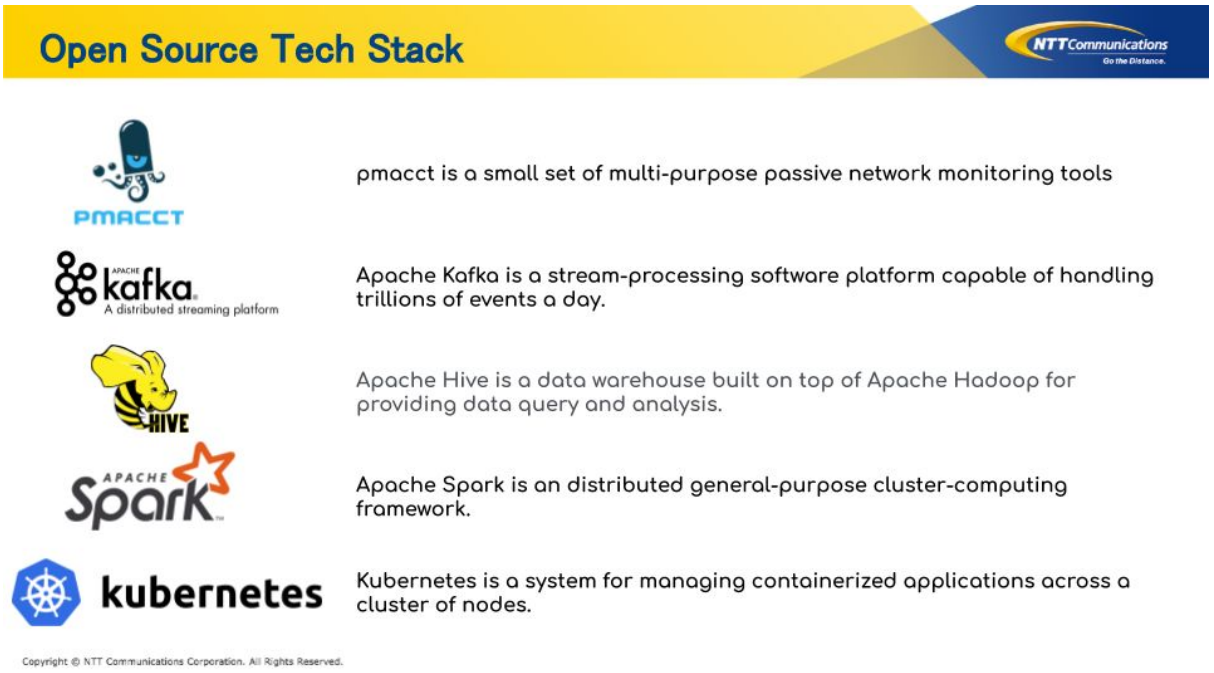
What is the need of the hour?
Intelligent, managed DDoS protection solutions will enable us to proactively mitigate these increasingly sophisticated attacks.
A high-accuracy real-time anomaly detection system can solve this problem by combining the strengths of distributed infrastructure and deep learning for enhanced real-time efficiency.

今必要なことは?
近年、DDoS攻撃の数、規模、社会に与える影響度が急増しています。
過去2年間で、その影響の規模と深刻度は200%近く上昇しています。
IoTデバイスの増加と5Gを背景に、2020年にはボットネットによるDDoS攻撃はさらに増加すると考えられています。

NTTコミュニケーションズは、ISPとして、またICTソリューションプロバイダとして、ネットワーク内のセキュリティの脅威を監視する立場にあります。

インテリジェントに管理されたDDoS対策により、ますます高度化するこれらの攻撃を積極的に緩和することが必要になります。
そこで、高精度なリアルタイム異常検知システムを、分散型インフラの強みと、高性能なディープラーニングを組み合わせることで、この問題を解決することを企図しています。

Page8



We have used the following opensource technologies in our architecture.
pmacct, Kafka, HIVE, Spark and Kubernetes.

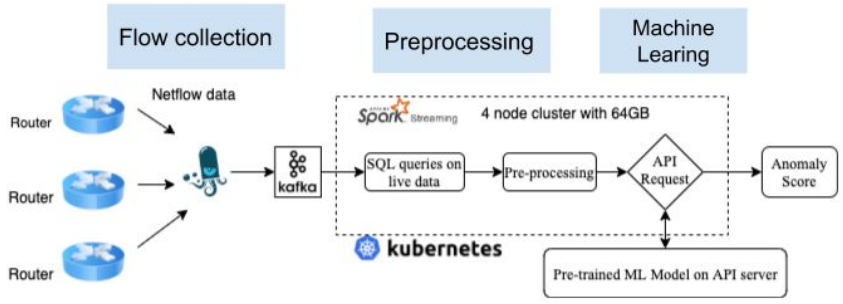
I will hand over to Aakash to explain each of these technologies and its use in our architecture in detail.

私たちのアーキテクチャでは、以下のオープンソース技術を組み合わせて構築しています。
pmacct、Kafka、HIVE、Spark、Kubernetesです。

これらの技術のそれぞれと、私たちのアーキテクチャでの使用について詳しく説明するために、Aakashに渡します。

Page10

Architecture overview



Our infrastructure mainly has three parts:

- Flow Collection= Collects Netflow data from Router and then connects data telemetry to kafka
- Preprocessing part = Processes streaming flow using hiveQL and extracts important information
- Machine Learning part = Runs as pyspark job which predicts the anomaly of live data using pre trained autoencoder model.

弊社のインフラは主に3つの部分で構成されています。

- Flow Collection: ルータからNetflowデータを収集し、データテレメトリをkafkaに接続します
- 次に前処理部分: hiveQLを使ってストリーミングフローを処理し、重要な情報を抽出します

- 最後に機械学習部分: 事前に学習したオートエンコーダーモデル(機械学習モデル)などを用いて、ライブデータの異常値を予測するpysparkジョブを実行します。

Page11

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/

pmacct is a small set of multi-purpose passive network monitoring tools. It can account, classify, aggregate, replicate and export forwarding-plane data, ie. IPv4 and IPv6 traffic; collect and correlate control-plane data via BGP and BMP; collect and correlate RPKI data; collect infrastructure data via Streaming Telemetry.
It can connect with multiple data sources such as Postgresql, mongodb,mysql, kafka, log files etc.
In our case we are using pmacct to collect Netflow data

pmacct は軽量で多目的なパッシブネットワーク監視ツールを複数集めたツールです。
IPv4 と IPv6 のトラフィックなどのフォワーディングプレーンデータのアカウント、分類、集計、レプリケート、エクスポート、
BGP と BMP を介したコントロールプレーンデータの収集と相関、
RPKI データの収集と相関、ストリーミングテレメトリを介したインフラストラクチャデータの収集が可能です。

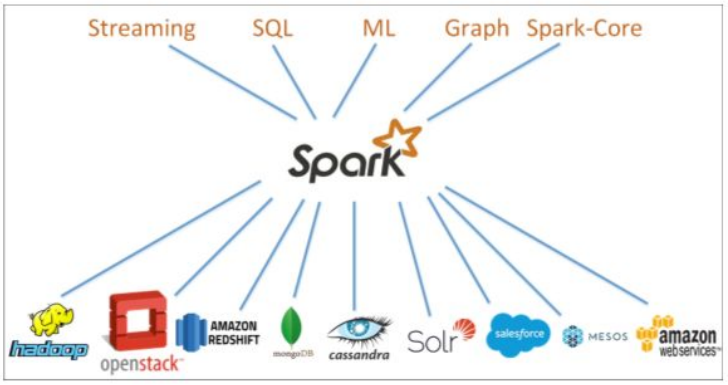
Postgresql, mongodb,mysql,kafka,ログファイルなどの複数のデータソースに接続することができます。

我々のケースでは、Netflow データを収集するために pmacct を使用しています。

Page12

What is Spark?

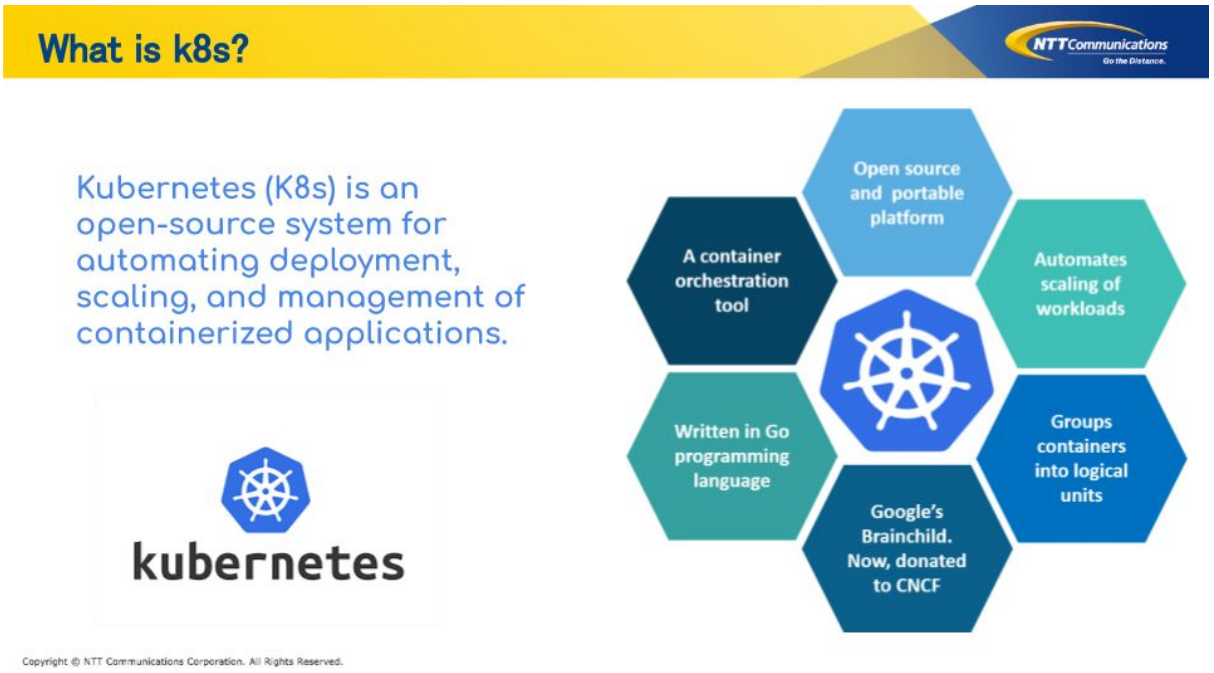
- Apache Spark is a open source lightning-fast unified analytics engine for big data and machine learning
- It is an optimized engine that supports general computation graphs for data analysis.



Apache Spark is a open source lightning-fast unified analytics engine for big data and machine learning
It has built in capability of processing streaming data, data analysis using pyspark, sparkR, spark SQL and built in support for distributed Machine Learning algorithms for big data.
It can consume data from almost all of the data sources such as hadoop, kafka, kinesis, various databases and from public cloud vendors.

また、Apache Sparkを使用しています。これはビッグデータと機械学習のためのオープンソースの高速で汎用な分析エンジンです。
ストリーミングデータの処理、pyspark, sparkR, sparkSQLを使ったデータ分析、ビッグデータ用の分散型機械学習アルゴリズムのサポートが組み込まれています。
Hadoop、Kafka、Kinesis、様々なデータベース、パブリッククラウドのベンダーなど、ほぼすべてのデータソースからデータを処理することができます。

Page13

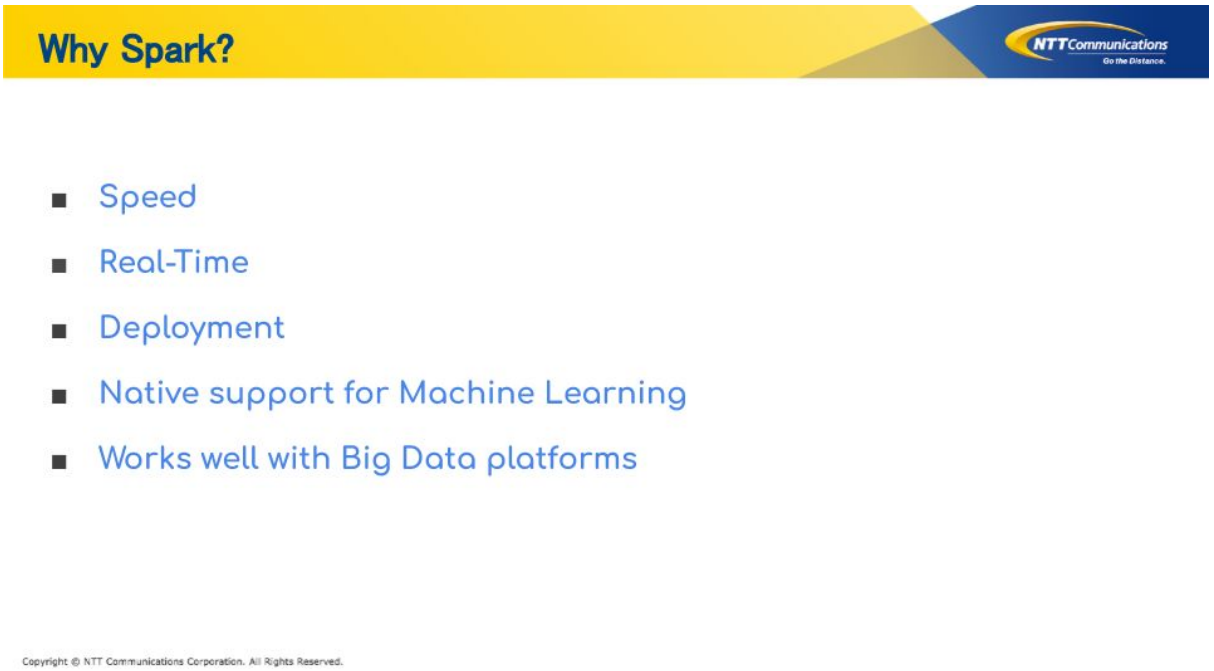


Kubernetes is container orchestration platform used for automating the deployment, scaling and management of containerized application
Kubernetes also scheduler API which is used for resource allocation and management.
In our case we have docker as container platform and we are using k8s to automate spark application deployment and using it as resource manager for spark.

Kubernetesは、コンテナ化されたアプリケーションのデプロイ、スケーリング、管理を自動化するために使用されるコンテナオーケストレーションプラットフォームです。

私たちはンテナプラットフォームとしてdockerを使用しており、また、aprakアプリケーションのデプロイを自動化するためにk8sを使用しており、saprkのリソースマネージャとして使用しています
KubernetesにはスケジューラAPIもあり、リソースの割り当てや管理に利用されています。

Page14



Why spark

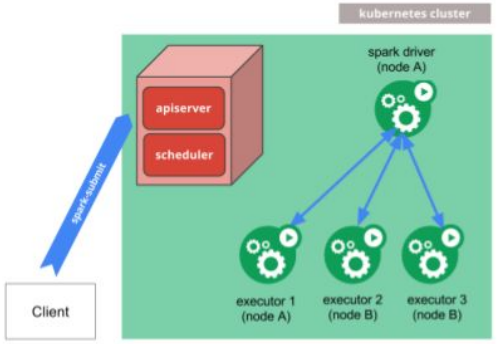
- Speed : Spark runs up to 100 times faster than Hadoop MapReduce for large-scale data processing
- Real-Time : Spark offers Real-time computation capability & low latency because of in-memory computation.
- Deployment: can be easily deployed through Spark's own cluster manager or Hadoop via YARN, or Kubernetes
- Native support for Machine Learning: MLib for ML Algorithms, Featurization, Pipelines, Persistence & Utilities
- Works well with Big Data platforms like Hadoop stack and enables seamless data streaming queries and management

なぜsparkを利用するのかについて説明します。

- 速度：Sparkは大規模データ処理のためにHadoop MapReduceの最大100倍の速度で動作します
- リアルタイム性 : インメモリで処理するため、Sparkはリアルタイム計算機能と低レイテンシを提供します。
- デプロイが簡単: Spark独自のクラスタマネージャやYARN、Kubernetes経由でHadoopに簡単にデプロイできます。
- 機械学習のネイティブサポート: MLアルゴリズム、前処理、パイプラインが実装できる機能があります (MLib)
- Hadoopスタックなどのビッグデータプラットフォームと連携し、シームレスなデータストリーミングクエリとその管理を可能にします。

page15

Spark on k8s

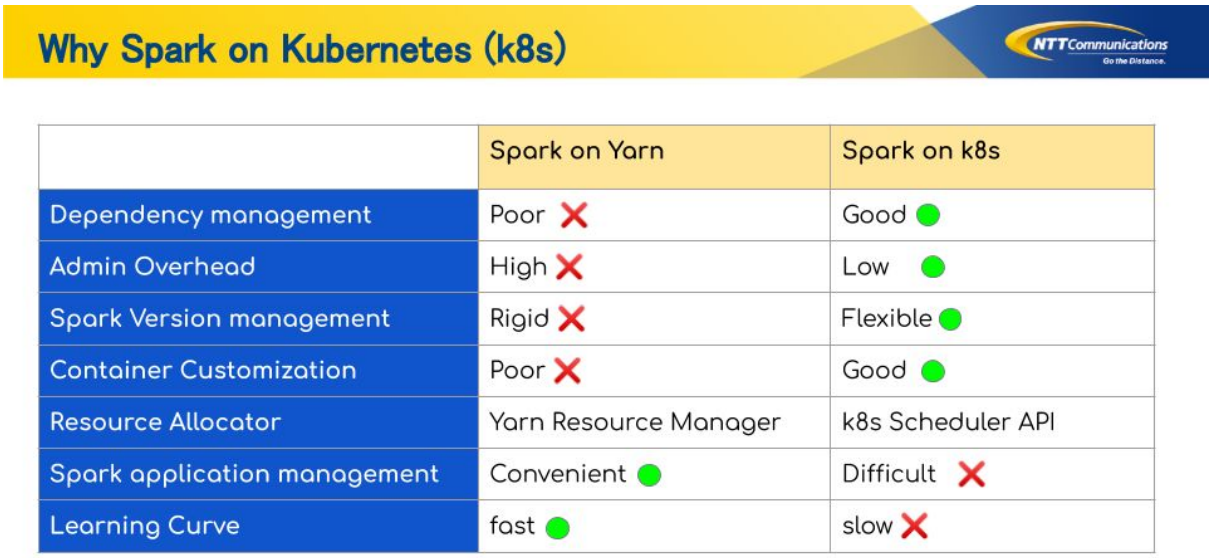



```
./bin/spark-submit \
--master k8s://https://[redacted]:6443 \
--deploy-mode cluster
--name predict-on-spark \
--jars
[redacted]/user/aakashnand/spark-streaming-kafka-
0-8-assembly_2.11-2.4.6.jar \
--conf spark.executor.instances=10 \
--conf
spark.kubernetes.container.image=[redacted]/test:latest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


How it works
spark-submit can be directly used to submit a Spark application to a k8s cluster.
Kubernetes master-that is apiserver creates a Spark driver as a k8s pod
We can specify the container image for driver and executor, in our case we are using custom-pyspark-image called “test”
The driver then creates executors which are also running as pod within Kubernetes and executes application code.
When the application completes, the executor pods terminate and are cleaned up, but the driver pod persists logs

私たちはsparkアプリケーションを作るとき、spark-submitという仕組みを使用します。
Spark-submit は、直接 k8s クラスタに Spark アプリケーションを投入するために使用できます。
Kubernetes master、つまりapi serverはますSparkドライバをk8sポッドとして作成します。
次に、Sparkドライバが、appの実行を担うエグゼキュータをk8sポッドとして作成します。
ドライバとエグゼキュータは両方k8sポッドなので、そのコンテナイメージを指定することができますが、今回の例は "test "という名前のカスタムpyspark-imageを使用しています。
右の実際の実行コマンドの例では、マスターにk8sのホストを指定したり、エグゼキュータの数を指定したり、実行時のコンテナimageを指定しています。

Page17



Why Spark on k8s

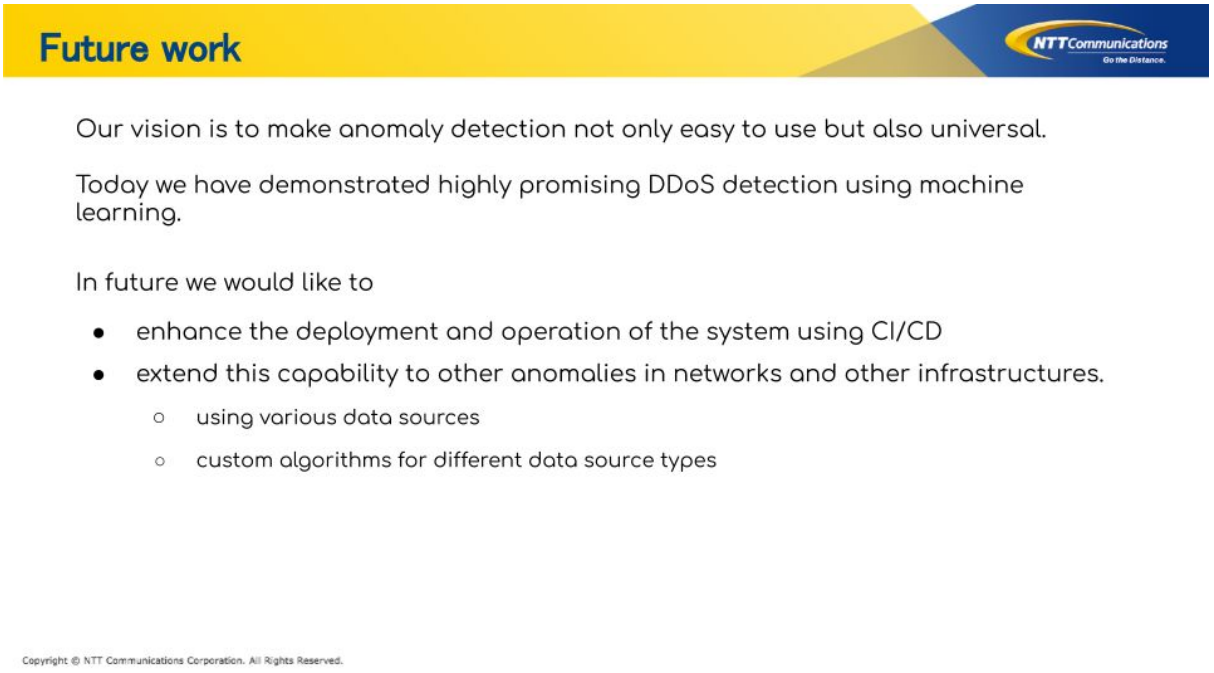
1. Dependency Management:
    a. Poor because lots of python environment, rigid structure
    b. Good because executors and drivers are isolated using container images
2. Admin Overhead
    a. High because need to manage spark, java, python version properly on entire cluster
    b. Low because container environment is customizable with different versions
3. Spark Version Management
    a. Rigid because all nodes needs to be upgraded for appropropriate version of spark
    b. Flexible because we can run many applications as kubernetes pods with different versions of spark-image or customised spark images
4. Container Customization
    a. Poor because in case of Yarn there is hardly any scope of customization of yarn containers except for memory configurations
    b. Good because we can create custom-spark images, pyspark images along with existing memory configurations
5. Resource Allocator
    a. Yarn uses YARN Resource Manager
    b. Kubernetes uses Kubernetes Scheduler API as resource manager
6. Learning Curve
    a. Spark on Yarn is easy to learn because there is no concept of container technology and orchestrator.
    b. Learning Curve is slow because we need to understand various concepts of container technology and kubernetes

私たちは以前はyarbでsparkを動かしていましたが、今はk8sで動かしています。
sparkをyarnとk8sで使用する際の違いについて説明します。

1.
    a. (依存関係の管理しやすさ) yarnの場合、実行するアプリケーションに合わせ

てpythonだと複数の環境を管理する必要があり、あまり柔軟でない
    b. (依存関係の管理しやすさ) k8sの場合、コンテナイメージを使って実行時の環境とドライバの環境が分離されているので良い
2. 管理者のオーバーヘッド
    a. (管理者のオーバーヘッド) yarnの場合、クラスタ全体で Spark, java, python のバージョンを適切に管理する必要があるため高レベルです
    b. (管理者のオーバーヘッド) k8sの場合、全てがk8sポッドで動くので、コンテナ環境のバージョンによってカスタマイズ可能なため低い
3. Sparkのバージョン管理
    a. (Sparkのバージョン管理) yarnの場合、すべてのノードが適切なバージョンのスパークにアップグレードする必要があるため、柔軟でない
    b. (Sparkのバージョン管理) k8sの場合、異なるバージョンのスパークイメージやカスタマイズされたスパークイメージで多くのアプリケーションをk8sポッドとして実行できるため、柔軟性が高い
4. コンテナのカスタマイズ
    a. (コンテナのカスタマイズ) yarnの場合、メモリ構成以外にyarnコンテナのカスタマイズの余地がほとんどないので悪い
    b. (コンテナのカスタマイズ) k8sの場合、既存のメモリ構成に合わせて、独自のsparkイメージやpysparkイメージを作成できる
5. リソースアロケータ
    a. (リソースアロケータ) yarnの場合、ヤーンはYARNリソースマネージャーを使用しています。
    b. (リソースアロケータ) k8sの場合、Kubernetesはリソースマネージャーとしてkubernetes Scheduler APIを使用しています。
6. 学習しやすさ
    a. (学習しやすさ) yarnの場合、Spark on Yarnはコンテナ技術やオーケストレータの概念がないので学習しやすい。
    b. (学習しやすさ) k8sの場合、コンテナ技術やkubernetesの様々な概念を理解する必要があるため、学習曲線は遅い。

Page27



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

Today we have demonstrated highly promising 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

私たちのビジョンは、異常検出を使いやすく、そして普遍的なものにすることです。
今日、高度な機械学習を使用したDDoS検知のデモを行いましたが、
将来的には、CI/CDと組み合わせてより開発をしやすくしたり
networkインフラだけでなく、他のインフラに適応可能なようデータソースタイプやアルゴリズムを拡張することを考えています。