**Cloud Architectures :**

This document will help to understand different architecture patterns for use cases in AWS.

Initially we will focus on couple of applications already running on-premise and see how it can be migrated to cloud.

**Project One :**

There is an IoT based platform that will capture the network (XDR’s) and billing (CDR’s) events, process it and store where different consumers will try to retrieve the data.

The platform is very complex and consists of multiple subsystems deployed as applications on standalone VM’s and as containers in an OpenShift cluster.

In this example we will concentrate only on the Real Time Analytics service that will fetch the signaling events from third party system where the 3rd part system will get the events from the network.

The Real Time Analytics platform is deployed as a container based solution in OpenShift and HDP cluster on standalone VM’s.

What is the IoT ?

IoT is called Internet of Things.

The actual definition is “e Internet of Things describes physical objects with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks”.

There are millions and billions of devices that try to exchange data using sensors and devices. The devices can be trucks, cars, utility meters, smart devices at home etc. which will generate some data.

The data in general is referred as events which has some set of fields that contain values assigned. In the platform what it captures the data from IoT devices are signaling events called XDR’s which has some standard 3GPP schema and is of JSON based i.e. of a key value pair.

Example of an event in the IoT platform is as below..

{ "\_index": "xx022-05-30", "\_type": "\_doc", "\_id": "110753CD48217CE3BE75", "\_version": 1, "\_score": null, "\_source": { "xxx": "1", "xxx": "2022-05-30T11:07:53.694Z", "xxx": "2022-05-30T11:07:53.708Z", "xxx”: "96000001", "xxx": "610364920000570", "xxx": "617", "xxx": "53a9", "xxx": "4e", "xxx": "52896", "xxx": "9189", "xxx": 15468263, "event": "N/A", "xxx": "internet.ts.m2m.mnc020.mcc238.gprs", "xxx": "88944467", "xxx": "7", "xxx": "1a7", "xxx": 100000050201640, "xxx": "03", "xxx": "171.9.149.70", "xxx": "96000012", "xxx": "86", "xxx": "operator\_Generi", "xxx": "8", "xxx": "000001756885774", "xxx": 0, "xxx": "enterprise\_96000012" }, "fields": { "timestamp\_event": [ "2022-05-30T11:07:53.694Z" ], "timestamp\_processing": [ "2022-05-30T11:07:53.708Z" ] }, "sort": [ 1653908873708 ] }

What is the throughput ?

Throughput is the amount of data the system can handle at any given point of time.

The current system is capable of handling 40K messages/sec of indexing the data on an average and can handle around peak of 200K messages per sec from some minutes.

The consumers can execute 20 queries per sec which is mixed of light and heavy queries.

There can be different consumers and in this case there are 3 main consumers.

1. A portal page that will retrieve the data from the database. The page will do some aggregations based on time and calculate the sum of data uploaded, downloaded etc. No.of events generated for a particular device in past 12hrs etc.
2. There is another script that will try to execute some queries towards the database endpoint every minute to verify the health of the system.
3. There are API’s that will fetch the raw data from the database mainly the events captured from the network where customers can see what events are generated from the device.

Some KPI’s at high level

|  |  |  |
| --- | --- | --- |
| S No | Indexing | Query |
| 1 | 40K msg/sec | 20 RPS |

Current Architecture :

**Infrastructure:**

The application uses a dedicated OpenShift and HDP cluster that consists of 8 physical servers.

The OCP cluster has 3 types of roles like Master, Infra and worker nodes.

The Master and Infra nodes are needed for the cluster itself and the worker nodes are needed for running the applications as containers.

|  |  |  |
| --- | --- | --- |
| CPU | 40 Cores / 80 Threads | 2 \* [6230N](https://ark.intel.com/content/www/us/en/ark/products/192450/intel-xeon-gold-6230n-processor-27-5m-cache-2-30-ghz.html) |
| RAM | 384 GB | 6 \* 64 GB modules |
| Disk | 20,8 TB | 13 \* [1,6 TB disk](https://www.hpe.com/sg/en/product-catalog/servers/server-solid-state-drives/pip.hpe-server-mixed-use-mu-solid-state-drives.1011175264.html) |

There are 13 local disks SSD’s connected to the 8 physical servers which are used by different applications for storing the data.

RAID0 is used by applications which have redundancy and RAID 5 will be used by other applications

The solution will be virtualized using VMware vSphere, meaning the physical servers will be running the ESXi hypervisor. On top of this hypervisor we will be running the VMs. Solution consists of two main parts; the OCP cluster (Openshift) and the HDP cluster (Hadoop).

Software Stack :

Once the OCP cluster is installed along with the HDP then the software stack installation can be done.

The OCP cluster consists of 3 master nodes, 3 infra nodes and 8 worker nodes where the different applications are deployed as containers on different nodes to ensure HA and the system is resilient for failures.

Different software components are as below.

1. Kafka cluster
2. Zookeeper
3. Elasticsearch
4. Stolon
5. Prometheus

3rd party

1. Grafana
2. Graphite Exporter
3. Kube-State metrics
4. Logstash
5. Flow-application (That connects to HDP cluster)
6. Kibana
7. Kafka exporter
8. Monitoring Elasticsearch

3rd party

Kafka

Grafana

Consumer1

Consumer2

User

Kibana

Exporters

2

1

3rd party

User

Consumer2

Consumer1

6

8

7

Exporters

Stolon

5

4

3

Consumer2

Consumer1

Prometheus

Logstash

Grafana

Kibana

ES

HDP

Kafka

Real Time Analytics

1. All the orange boxes are deployed as containers in OCP cluster
2. The HDP cluster is run on standlone VM’s
3. There is a 3rd party vendor software that will capture network calls and send to Kafka

**Here is the E2E flow for the data.**

1. Event happens in the core network by the devices
2. 3rd party component captures the events from the network and send it to the Analytics platform.
3. Kafka acts as a message bus and captures the events
4. Kafka sends it to flow-application running in HDP which is based on Apache spark
5. Flow-application will index it in Elasticsearch i.e. a NO/SQL database
6. A consumer i.e. portal web page will retrieve the needed data from ES.
7. Different consumer i.e. API’s which are basically ES API’s can be run by end users to get the raw events directly.
8. There is logging and monitoring solution that uses Grafana.

**Current Challenges with this solution :**

The solution is run as private cloud on an on-prem environment that uses the cloud native technologies.

All the applications except the HDP part are containerized and run as Docker containers in Kubernetes/OCP cluster.

Different needed services like Grafana, Elasticsearch, Kafka are exposed as services that uses NodePort for external consumers to retrieve the data.

There are 5 main challenges that I see with the existing solution and of course there are other challenges that can be identified but for now I wanted to consider the Top 5 challenges only.

1. Scalability
2. Cost
3. Time to Market
4. Elasticity
5. Innovation

**Scalability :**

As IoT is a data hungry technology where millions and billions of devices will generate huge amount of data which the current systems are unable to meet the demand.

Considering the trend I see when the system initial built as a PoC in 2017 it was handling around 3K messages per sec and as on today i.e. May 30th 2022 it is handling 40K messages per sec and the trend is expected to grow quarter on quarter where more subscriptions or devices will be added to the platform that will generate data or existing devices start generating more data.

I wanted to upgrade the platform by re-designing the components and bringing new hardware etc.

The challenge I faced is to scale the system based on the incoming data as I need to plan for the needed servers, raise the investment ticket, get the needed approvals, lock the vendor, make an order, get the procurement, Installation of the servers etc. which almost took 6 months to 1 year for getting 8 physical servers which is a very huge lead times.

In the meantime, the current platform was getting more traffic which started underperforming because of the more incoming data and consumer pattern has changed resulting more queries towards the system.

Considering all above the current platform was handling the extra load having some performance issues and quite a few incidents where we had to lose customer trust.

The system is so complicated that there are many interdependent components and interfaces thus other components must scale in parallel causing more time and issues.

Ex: There are couple of switches that were handling the current load which there are no extra ports to plugin the new servers for which new switches were ordered and same lead times caused the delay. The more interesting and challenging part of the procedure for configuring the switches and stack the same to the existing switches to serve the traffic is another time consuming process due to the change management procedures and the delays.

**Cost :**

The cost for building the platform that will capture the network data, process and store which are later consumed by different applications is quite huge considering multiple aspects as below

1. Hardware
2. Software development
3. Installation, Data Center maintenance (Power, Cooling etc.)
4. Hardware and software licenses
5. Co-ordination and onboarding multiple teams for building the platform
6. Maintenance cost i.e. HW maintenance, software maintenance tec.

Considering all above the cost of the project for CAPEX and OPEX from 2018 till 2021 (CAPEX) and OPEX from 2021 till 2026 is 4 million USD.

The traffic pattern looks as below..

Weekdays :

8 AM till 6 PM --- 35K messages per sec on average

6 PM till 7 AM – 15K messages per sec on average

7 AM till 8 AM --- The messages start increasing gradually from 15K and reach 35K by 8 AM

Weekends :

During weekends the traffic pattern is around 12K to 14K messages per sec.

Also there will some peaks up to 50K, 100K for couple of minutes during some times.

Based on the above pattern the current implementation will have overprovision of resources during night and weekend as there is no automatic scale-down of the workload happens based on the alarm.

When the peak load is occurred, there are performance issues of the system and some time the application is very slow causing timeouts for customers.

With this scenario the cost is not balanced, and the system is always underutilized and paying more for less usage.

**Traffic Pattern:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S NO** | **8 AM to 6 PM** | **6 PM to 7 AM** | **7 AM to 8 AM** | **weekends** |
| 1 | 35K (average) | 15K (average) | 15K to 35K(average) | 12K to 14K(average) |
| 2 | 40K, 70K, 100K (Peak for minutes) | 18K, 20K (Peak for minutes) | NA | 16K, 18K (Peak for minutes) |
|  |  |  |  |  |
|  |  |  |  |  |

**Time To Market :**

Considering the above scenarios where the total time it took for the project to go-Live due to multiple dependencies, HW procurement delays, change management approval process delays, people competency issues etc. the project took 2 years more than the initial estimated which the new platform with additional capabilities and new features caused a very long time to market where customers were not happy and we have lost a big amount of market share by not able to capture new customers.

**Elasticity :**

Elasticity is scaling the resources up and down based on the need.

With the on-prem environment it is a real challenge to predict the traffic patterns and then add the servers automatically based on the event is not possible with on-prem infrastructure.

To do this the company need to invest huge on hardware, procure well in advance and wait till more traffic occurs then start adding new machines to the system which if no extra traffic comes then the hardware resources are of waste that can’t be utilized in a proper way.

Most of the cases with on-prem environment the situation will be either the resources are overprovisioned or they don’t have extra servers to handle the peak or incoming traffic causing to have performance issues.

**Innovation :**

With IoT there is quite lot of data that it generates and as of today we are handling 20 TB of data in the database only for Real Time Analytics platform.

With the current solution the usage of the data is limited as there is no much room and scope for building data lakes and doing Bigdata analytics.

We are confined to some data analytics using technologies like flink with minimal resources to give some insights on the data.

Due to storage space and performance issues we couldn’t store more data thus unable to do innovation by using AI/ML techniques and other aspects.

As on today there is no mechanism to predict the business and traffic pattern based on the historical data that is causing to do the estimations and business evaluation for server procedure or workload provisioning.

**What is the proposed Solution :**

Considering the above challenges migrating to cloud (AWS) will help to solve the current challenges and brings value to the business with more agility.

**Reference Architecture in AWS for the current use case :**

As mentioned the complete BSS platform is a very complex solution that involves multiple services so for now the scope is only limited to Real time Analytics service and we will use a Hybrid architecture for getting the data from 3rd part vendor running on-prem and consumers connecting to RTA application also runs on-prem.

We will use AWS direct connect for high speed bandwidth and security considerations.