Distributed Big Data Driven Framework for Cellular Network Monitoring Data

Alexander Suleykin
V. A. Trapeznikov Institute of Control Sciences,
Russian Academy of Sciences,
Moscow, Russia
aless.sull@mail.ru

Peter Panfilov
School of Business Informatics
National Research University – Higher School of Economics
Moscow, Russia
ppanfilov@hse.ru

***Abstract*—The smart monitoring system (SMS) vision relies on the use of ICT to efficiently manage and maximize the utility of network infrastructures and services in order to improve the quality of service and network performance. Many aspects of SMS projects are dynamic data driven application system where data from sensors monitoring the system state are used to drive computations that in turn can dynamically adapt and improve the monitoring process as the complex system evolves. In this context, a research and development of new paradigm of Distributed Big Data Driven Framework (DBDF) for monitoring data in mobile network infrastructures entails the ability to dynamically incorporate more accurate information for network monitoring and controlling purposes through obtaining real-time measurements from the base stations, user demands and claims, and other sensors (for weather conditions, etc.). The proposed framework consists of network probes, data parsing application, Message-Oriented Middleware, real-time and offline data models, Big Data storage and Decision layers, and Other data sources. Each Big Data layer might be implemented using comparative analysis of the most effective Big Data solutions. In addition, as a proof of concept, the roaming users detection model was created based on Apache Spark application. The model filters streaming protocols data, deserializes it into Json format and finally sends it to Kafka application. The experiments with the model demonstrated and acknowledged the capacities of the Apache Spark in building foundation for Big Data hub as a basic application for online mobile network data processing.**

I. INTRODUCTION

The number of smartphone users has already reached 4.61 billion users in 2016, and upward trend is forecast for the market with 5.07 billion users by the end of 2019 [1]. The process of rapid mobile users growth is immutably leading to the proportional increase of data being generated by mobile subscribers, user equipment, cellular nodes and whole mobile network. This is becoming more challenging for mobile operators to overcome constantly increasing data volumes for many different protocols and mobile network interfaces using traditional approaches with standalone systems, relational databases, many different formats of data storage and transmission. Thus, the appearance of new challenges generates new approaches such as Big Data, Internet of Things, Machine-to-Machine Communications etc. in application to cellular network data storage, aggregation, transformation, enhancement and transfer.

In fact, cellular network has many different protocols for data transmission and corresponding interfaces. Each node of the network is communicating with other node(s) and external environment according to worldwide standards 3GPP [2] and ITU [3]. Every protocol has its own parameters, which are different from one network element to another. The complexity of protocols, its differentiation, volumes of data being transferred and importance of data have led to the need of searching new ways of cellular data analysis, using last technology achievements such as Big Data methods and technologies.

In many respects, the monitoring services in mobile networks are dynamic data driven application systems (DDDAS) where data from sensors monitoring the system drives computations that in turn can dynamically adapt and improve the monitoring process as the network evolves. In this work, we propose a Distributed Big Data Driven Framework (DBDF) for Cellular Network Monitoring Data on the basis of the Dynamic Data-Driven Application System (DDDAS) paradigm [4] and a core concept of Lambda architecture [5-7], specifically targeted at scalable and secure real-time Big Data application systems, comparative analysis of different Big Data methods, techniques and tools.

The new concept of DBDF is targeted at management representatives and decision makers in Mobile Engineering, Operating and Planning departments and is intended to help them in managing cellular network in real-time on the basis of network measurements, simulation and optimization models. The suggested Framework consists of different components that realize distributed smart monitoring paradigm for the cellular network data as it is presented in Fig. 1.

II. RELATED WORK

Big Data models and techniques, such as machine learning and data mining demonstrate a huge potential for the ever growing and demanding application area of cellular network data processing. In our previous work [8], a Data-Driven Smart Management (DDSM) approach for cellular network operation and planning was introduced, which proposes the development of a system comprising of subsystems for real base stations and network measurements, network simulation, optimization model and real-time decision making. The simulation of network problem zones detection and prioritization based on user complaints data, area population density data and base station characteristics was enhanced with heuristic-based optimization model using analytically predefined threshold. The recent survey of Big Data analytics in wireless and wired network design [9] showed that integration of many different protocols/traffic layers all together is the best way to building robust data communication networks with refined performance and intelligent features. There are examples of research works where an architectural framework for applying the big data analytics in the mobile cellular networks [10] is considered. However, actual output of this work was limited to the analysis of hardware aspects of data exchange (i.e. GPS and base stations) and the discussion of a role of MapReduce in handling this data. In [11], many challenges in creating big data frameworks were introduced based on real-time big data processing, as well as, the role of RAM, CPU and GPU and different computational models discussed. In addition, the role of data

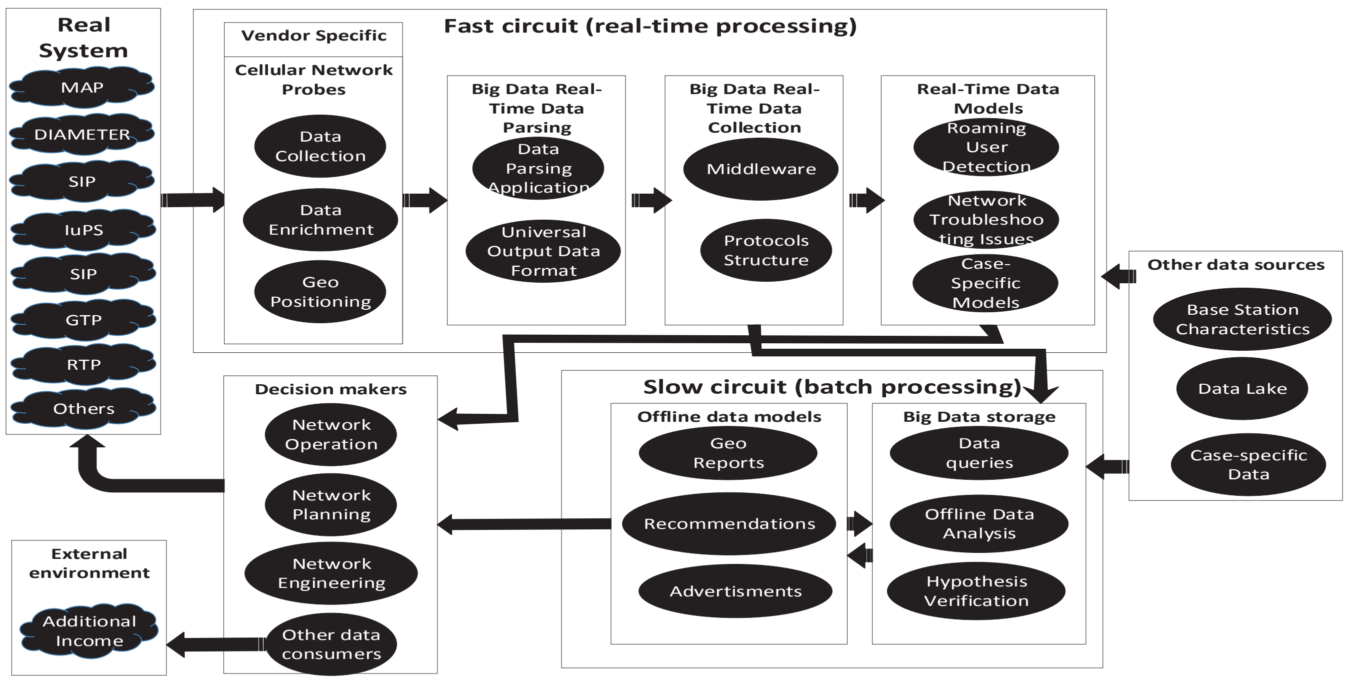


Fig. 1. The architectural overview of the DBDF – a concept

management, data security, collection and analytics was illustrated, and the real-time processing frameworks based on Apache Spark and Storm were introduced.

In our attempt of building the Big Data framework for the cellular network data, we are aiming at full cellular network distributed data architecture from the collection of data from network elements thought parsing, aggregation and real-time data analysis to storage, offline data analysis and the data exchange with external consumers for additional profit for telecom companies (Fig. 1).

III. Distributed Computing and Big Data

A. Distributed systems issues

The area of distributed systems in the general area of computer science studies theoretical issues of organization of distributed computing and data processing. Also, distributed systems are sometimes defined in more narrow sense, as the application of distributed computer systems to solve time-consuming computational tasks. In this context, distributed computations are a special case of parallel computations, i.e. simultaneous or concurrent solution of different parts of a single computational problem using several (interconnected) computing devices. Note that when studying parallel concepts, the main emphasis is usually on methods of division of a solved task into subtasks, which can be solved concurrently for the maximum acceleration of computations. The main issue of the organization of parallel computing using distributed systems is in the accounting for differences in the characteristics of available computing resources and the availability of significant time delay in the exchange of data between them.

Over the past few years, distributed systems have become increasingly popular and important in many application areas and for many application problems. The main reasons for the growth of their significance can be identified as follows:

- Geographically distributed computing environment. Today in most cases the computing environment itself is by nature a geographically distributed system. As an example is the banking network. Each bank serves accounts of their customers and processes transactions with them. In case of transferring money from one bank to another it requires the implementation of interbank transactions and the bank systems interaction. Another example of a geographically distributed computing environment is the Internet itself.
- The requirement to increase computing performance. The performance of traditional uniprocessor systems is fast and steadily approaching to its limit. Different architectures (such as superscalar architecture, matrix and vector processors, single-chip multiprocessor systems) are designed to increase the performance of computing systems due to various mechanisms for parallel execution of commands. However, all these techniques can increase productivity by a factor of tens of comparisons with classical sequential solutions. Besides, the scalability of such approaches leaves much to be desired. To increase the productivity in hundreds or thousands of times and at the same time to ensure good scalability of the solution it is necessary to put together multiple processors and ensure their effective interaction. This principle is realized in the form of large multiprocessor or multi-computer complex.
- Shared resources. An important goal of creating and the use of distributed systems is the provision of users (and applications) to access remote resources and ensuring their joint use. In this formulation the term resource refers both to hardware components computer system, and to software abstractions with which the distributed system works. For example, a computer user 1 can use the disk space of computer 2 for storing data files. Or, a single application can use free computing power of several computers to speed up own computations. Distributed databases and distributed system of objects can be an excellent example of a joint use of software components, where appropriate software abstractions are distributed over several computers and co-served by several processes.
- Fault tolerance. In the traditional "unallocated" computer systems based on a single computer (possibly high-performance), failure of one of its components usually leads to the failure of the entire system. Such a malfunction in one or more components of the system is called a partial failure if it does not affect other components. A characteristic feature of distributed systems, which distinguishes them from single

computers, is resistance to partial failures, i.e. the system continues to function after partial failures, however, slightly reducing the overall performance. A similar possibility is achieved by redundancy when the system additional hardware (hardware redundancy) or processes (software redundancy) that make possible correct functioning of the system if it is not working or incorrect operation of some of its components. In this case distributed system tries to hide the facts of failures or errors in some processes from other processes. For example, in systems with triple modular redundancy (English Triple Modular Redundancy, TMR) three identical computational modules are used for identical calculations, and the correct result is determined by a simple vote [12].

B. Big Data issues

Distributed computing techniques have been widely used by data scientists before the advent of Big Data concept. Thus, standard and time-consuming algorithms were successfully replaced by their distributed versions with the aim of agilizing the learning process. For many of current issues with Big Data processing, especially in cellular network monitoring with such high volumes of data, a distributed approach is becoming immutable nowadays. The first framework that enabled the processing of large-scale data was MapReduce concept (2003). This tool was aimed to easily handle huge datasets in an automatic and distributed way using MapReduce paradigm, and usual user is able to use a distributed and scalable tool without worrying about technical nuances: data partitioning, failure recovery or job communication. However, MapReduce concept is not designed to scale well when dealing with iterative and online processes, and usually deal with batch data tasks with relatively huge latency comparing with online data processing. This paradigm would be implemented in Batch layer for Big Data storage.

Another group of methods is based on distributed in-memory computing, micro-batch and real-time techniques. These methods are used for online Big Data processing with millions of tuples per second per node. In our proposed solution these methods are used in real-time data transformation component, real-time data parsing and also might be used in real-time data models.

Message-driven applications are applications, which allow processing of future messages that arrive after subscription. The main advantage of such systems is that many consumers can access the same data in independent way. Thus, a message-driven approach is proposed to be used in Big Data real-time data collection to enable many data consumers easily access cellular network monitoring data.

C. Dynamic Data Driven Application Systems Paradigm

The DDDAS paradigm implies that real-time application data are dynamically incorporated into computations in order to steer the measurement process of an application system [4]. A generic DDDAS platform performs timely monitoring, planning, and control of distributed operations at complex systems and allows to reproducing the system operations in the most accurate manner. The components of system architecture of DDDAS platform and methodologies include 1) smart monitor (implements smart algorithms for state estimation, fault detection and localization, model fidelity selection); 2) real time data driven adaptive distributed simulation, using 3) distributed data base of the system, and 4) decision support system (user or customer side management decisions, service pricing and operation planning).

It has been demonstrated that this paradigm can be directly used to realize smart management systems and services in mobile network infrastructures. For example, DDDAS-supported DDSM architecture provides an approach to address issues such as creating more efficient and reliable cellular networks and to mitigate mobile traffic and QoS problems [12].

The concept of DDSM in investigates new approaches and tools for the real-time data acquisition and timely control of complex CN environments via incorporation of dynamic data into adaptive simulations. Such an application of the DDDAS paradigm in cellular network infrastructures promises more accurate information for monitoring and controlling purposes through obtaining real-time measurements from the base stations, computing and networking resources, and other sensors (for weather conditions, etc.).

D. Lambda Architecture for Big Data Applications

Building a reliable and efficient distributed big data application that satisfies a variety of end-user latency requirements is a challenging task. Lambda Architecture (LA) represents a useful framework to designing such applications. The appearance of the LA concept was inspired by the following motivations:

- the need for a robust and fault-tolerant system, both against human mistakes and hardware failures;
- the system should be linearly scalable scaling out rather than up;
- to serve a wide range of workloads and use cases, where low-latency reads and updates are required, with support of ad-hoc queries;
- the system should be extensible and features should be added easily.

The Lambda Architecture (LA) comprises of 3 main parts:

- Batch layer. This layer has two functions: manage the main append-only raw data streams and pre-compute arbitrary query functions calling batch views. In our DBDF architecture this is a so called “slow circuit” layer, where cellular network data is coming from real-time data collection component using batches.
- Speed layer. There are different fast and incremental algorithms are used with low latency, the speed layer deals only with recent data. In our proposed DBDF framework these are key components that are represented as “fast circuit”, and all components inside this Speed layer deal with online (real-time) data. Computation latency is key issue there.
- Serving layer. This layer indexes the batch views in Batch layer and data can be queried using ad hoc with low latency. This layer is not included in our current DBDF implementation and will be considered as a future work aimed at organizing and building Cellular network monitoring Serving data layer [6-8].

E. Distributed Computing for CN monitoring data

In fact, cellular network data is a streaming data from interfaces, base stations, billing system etc. represent high-loaded systems with Gigabytes of data per second, and it is even compressed data. Thus, CN streaming monitoring data need to be parsed and processed on the cluster of computers with application of Big Data methods for processing of very large volumes of data. Distributed methods and techniques of data processing for cellular monitoring data play important role for whole CN data processing pipeline. All application systems and methods should be adopted for distributed computing, high performance data analytics, and support scalable, reliable, secure and distributed data processing. This is a key requirement to all selected or developed tools for processing CN data.

IV. Key Architectural Components of DBDF

We can group major components of the DBDF in 8 blocks:

1) Real CN that consists of network elements like RNCs, BSs, NodeBs, eNodeBs, MMEs, GGSNs and other nodes collecting and transferring data to/from different cellular interfaces. The real cellular network consists of many different elements, and every network generation has their own protocols for data exchange and interfaces. The complexity of the network is amplified by the plane of data transfer – user plane and control plane, and the type of transferred data – voice, SMS, data itself.

In addition, the CN measurements might be represented as base stations characteristics, weather conditions, or subscriber's complaints, etc. This data comprises of different attributes and properties of the real CN system.

2) Events-Driven data records are records that appeared after some event happened on CN. It means that different data records should be created and transferred after some CN event. Thus, these events might be presented as data intercommunication between nodes (transactions), data about base station parameters, user complaints, etc.

Base station records are the characteristics of base station with their geo-positioning, supportive technology, LAC code, Cells, address, vendor etc. User complaints are complaints records that being collected in connection to the particular geographical location. Different transactions are records that represent a part of one subscriber session. The data in transactions are being transferred according to 3GPP and ITU standards in predefined format – CN data exchange protocols.

3) Big Data Driven programming languages is a special application capable of processing huge volumes of data (gigabit per second, Gbps) that parses all highly compressed protocol data (depends on vendor format) to the one unified data format, or capable to process data records received from real system (BS parameters, complaints). The data parsing is implemented via programming language, the choice of which is highly important because different frameworks offer diverse functionality and more adopted for some particular cases.

4) Message-Oriented Middleware is the Continuous Streaming data storage with predefined data structure according to different protocols. This is middleware that provides opportunity for many data consumers to get data in real-time as well as send data for storage using batches. The thing is that there might be many data consumers, which might be internal mobile company departments as well as external partners. Actually, it depends on the use cases – more data use cases, more data consumers are. So the most important requirements for this layer is to be capable with extremely fast data streams, be reliable, scalable and support many different data consumers.

5) Real-time and offline data models. It might be different data driven models and it might be considered as a simulation engine of the DBDFCN architecture. The models like those used in Problem Zones detection and prioritization [8] allow for optimization of operational costs for Mobile equipment maintenance and finding new ways of profit generation for Mobile providers.

6) Other data sources are sources within a mobile network company, which are used for model generation. It might be different databases – SQL, NoSQL, in-memory and not, but the main concept is that data should be used for Real-time data models. Other data sources as a part of Architectural framework are considered as additional on-demand data for models. These sources are usually data bases that might be maintained and exploited by different departments. For our previous work [8] we have used Base stations data base, complaints data base and population density per region data. Generally, the particular use cases and different requirements dictate the selection of the tool.

7) Big Data storage and queries layer is a component responsible for protocols data storage, report generation, validation of hypotheses and protocols data analysis. It can be done by using SQL queries to some database.

Offline storage is an Architectural component that realizes the cellular network protocols data reliable and scalable storage for offline analysis, hypothesis validation and different kinds of on-demand reports.

8) Decision layer, which is the output of Big Data models and reports. It might be streaming data in special predefined format, regularly-based reports or triggered data events after filtering. The decision support layer is the environment specially designed for Network Engineers, Planners, Operators, Managers and other data consumers. This includes all applications that are needed in order to make decision based on models output performed in previous steps of data pipeline. Decision makers are making their decisions based on this data and manage the real system – detect and weight problem zones on the cellular network [8], plan the network capacity and performance, base stations construction, manage Radio network elements and other.

V. Experiments and Results

A. Use case and data description

In order to prove the concept of applicability of proposed architectural solution, the Roaming users detection model has been created and tested on the basis of real cellular network of one of the largest mobile providers in Russia.

Mobile providers are curious about their subscribers who go abroad as well as migration in different regions internally. In fact, the significant percent of all telecom operator's income belongs to roaming users. Usually data usage, voice calls and SMS delivery are highly expensive in other countries, and telecom companies are extremely interested to not only detect abroad users, but also do it in almost real-time regime in order to offer to such users special set of services and options, as well as for internally relocated users.

Thus, telecom companies need:

- the ability to obtain data about the geolocation of the subscriber in real time for communication with the subscriber in Real-Time Marketing system;
- the ability to proactively detect the presence of blocking inconsistencies on the side of billing systems and HLRs (Home Location Registered).

To analyze the subscribers movement to other countries and to other regions inside country, according to specifications 3GPP and ITU we have looked at MAP (Mobile Application Part) protocol for 2G and 3G technologies (if user is currently using 2G or 3G network), or Diameter for 4G generation. The following attributes of the data would have been organized in messaging system entities (Table I).

We have searched for Cancel location and Update location events only. From the business logic perspective, we would be searching for the following events:

- start of another country visit;
- change of operator;
- change of country;
- return to country of origin;
- change of VLR (Visitor Location Registered).

TABLE I. FILTERED MAP AND DIAMETER ATTRIBUTES

Protocol	Event type	Filtered Attributes
MAP	CancelLocation/ UpdateLocation	MSISDN
		IMSI
		VLR
		Timestamp
		SccpCallingDigits
		SccpCalledDigits
MAP	UpdateGPRSLocation	MSISDN
		IMSI
		Back Calling Address
		Timestamp
Diameter	UpdateLocation/ Cancel Location	IMSI
		MSISDN
		OriginRealm

B. Experimental setup parameters

For messaging system, an Apache Kafka application was selected as a system with strong performance according to different benchmarks. It can handle more than 100 000 events per second [13].

The model itself was executed on the basis of Apache Spark application, which is the largest open source project in data processing [14]. Since its release, Apache Spark, the unified analytics engine, has seen rapid adoption by enterprises across a wide range of industries. Internet powerhouses such as Netflix, Yahoo, and eBay have deployed Spark at massive scale, collectively processing multiple petabytes of data on clusters of over 8,000 nodes. It has quickly become the largest open source community in big data, with over 1000 contributors from 250+ organizations [14-15].

For experiment, the YARN was chosen as a resource manager for Apache Spark. This manager is actually managing all cluster resources available for Spark jobs, which means that the capacity and performance of application are limited by resource manager YARN.

The data streaming of MAP and Diameter protocols was organized using one of the largest telecom company in Russia, and data were received in Kafka application in thrift data format, during the job implementation converted in Json data format and finally sent to another Kafka messaging system to topics according to event types described above (Table I). The average streaming size is really challenging with average 37 300 records per second for MAP protocol and average 24 200 records per second for Diameter protocol, with 61 500 records per second in total.

All installations of Spark, YARN and other support applications were done by Hortonworks (HDP version is 2.6.3.0-235).

Common configuration parameters are:

- Java version is 1.8.0_77 (Oracle Corporation);
- Scala version is 2.11.8;
- Operational system is Linux;
- Operational system version is 3.10.0-514.21.1.el7.x86_64.

The experiment was performed on powerful cluster. The characteristics of cluster and YARN resources available for all Spark jobs performed on the cluster are the following:

- 3 nodes for YARN allocated;
- Memory allocated for all YARN containers on a node is 306Gb. Total memory is 918 GB;
- Minimum Container Size (Memory) is 2048Mb;
- Maximum Container Size (Memory) is 100Gb;
- Number of virtual cores is 32;
- Percentage of physical CPU allocated for all containers on a node is 80%;
- Minimum Container Size (VCores) is 1;
- Maximum Container Size (VCores) is 32. VCores total is 96;
- 1 second interval between jobs;
- 2 Spark Executors selected on default.

The configuration file of Spark has the following parameters (Table II):

TABLE II. SPARK CONFIGURATION PARAMETERS

Parameter	Description	Value
app.test	Mark of a test run. With it, the launch time is limited, as are the data about the number of rows in stdout.	true/false
app.test.milisecond	The test run time in milliseconds, because Stream is triggered every 1 second, it is necessary that the value is > 1000 (preferably> 10000 because the time is spent also for the start of the job waiting, etc., it may not to have time to start)	120000
app.log	The level of logging in the job. Toggles the standard for log4j logging levels.	OFF/ FATAL/ ERROR/ WARN/ INFO/ DEBUG/ TRACE/ ALL
app.metrics.enable	Enabling metric records	true/false
app.metrics.index	Index name in Elastic for index storage	Index_name (i. e. kafka2rtm_metrics)
app.metrics.dateformat	Date format for the index. Corresponds to the formats for Elastic	Date_format (i. e. yyyy-MM)
app.metrics.hosts	Hosts Elastic	Host 1:Port1;Host2:Port 2;... Example (http://big-data:9092;....)
app.buffer.min	The size of the delay in minutes. The message is not older than the current time - the delay will be filtered and sent to the output queue.	minutes(example 20)
in.kafka.servers	Inbound topics Kafka servers	Host 1:Port1;Host2:Port 2;... Example (http://big-data:9092;....)
in.kafka.group.id	The name of the application in Kafka for reading from the topics, should be unique for the kafka cluster, under this name kafka saves the offset.	name(example spark-streaming-kafka-rtm)
in.kafka.offset.reset	The policy of picking up messages if there is no offset, two options are earliest or latest (read from the first message in the topic or from the last, respectively)	earliest/latest
in.kafka.gsmmap.topik	The topic name for incoming messages of type GsmMap	Test_gsm
in.kafka.diameter.topik	The topic name for incoming messages of type Diameter	Test_ diameter
out.kafka.servers	Outbound topics Kafka servers	Server1:port, server2:port
out.kafka.updateLocationTopik	Topic name of GsmMap UpdateLocation/CancelLocation	test_rtm_gsmmap_ulcl
out.kafka.updateGPRSTopik	Topic name of GsmMap UpdateGRPSLocation	test_rtm_gsmmap_ur
out.kafka.diameter.updateLocationTopik	Topic name Diameter UpdateLocation/CancelLocation	test_rtm_diameter_ulcl
out.kafka.group.id	Application name in Kafka for record in topic	spark-streaming-kafka-out

To run the application it is necessary to have 3 files in the folder:

- the startup script (run.sh is in the project in the ./src/main folder);
- stream-1.0-SNAPSHOT-jar-with-dependencies.jar – main code implementation file;
- the folder with configuration file.

C. Apache Spark job result analysis

As a result of Spark job implementation, the following results of YARN resources are allocated for this particular Spark job:

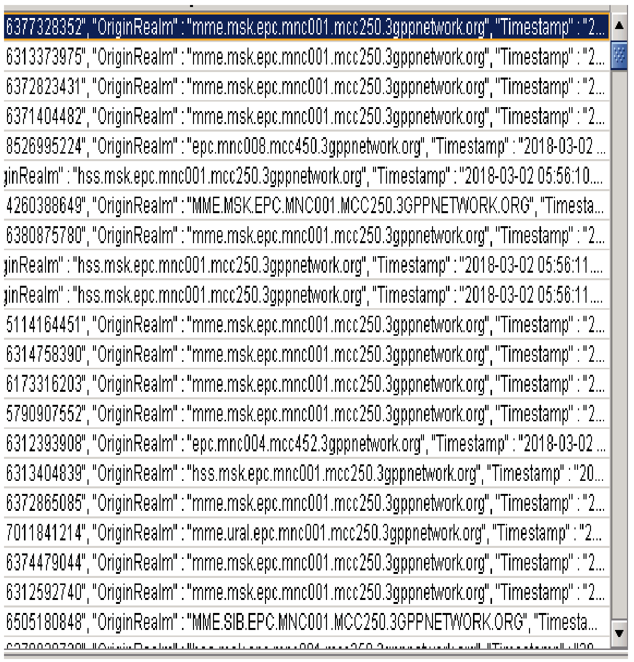
- Allocated memory is 6144Mb;
- Allocated CPU VCores is 3;
- 3 containers running;
- average Scheduling Delay is 14 ms;
- average Processing Time is 464 ms;
- total Delay is 478 ms.

The data can be viewed using Kafka UI tool. It shows the messages in topic (Fig. 2).

The result of implementation of an algorithm is data streaming with described fields above. IMSI and MSISDN fields represent "private" information and are cut from the image, but the other informative fields are visible.

The implementation of the model can be repeated for any telecom operator using the same protocols, taking into account that the cluster should be with the same performance in order to achieve stable work of application. Or, the data stream can be also proportionally decreased along with the amount of nodes in a cluster and their capacity.

The adequacy of the model was successfully checked by the comparison of amount of filtered messages for particular period. Thus, using python programming language we connected to Kafka application and took two regarded data streams – Map and Diameter protocols. We analyzed the same period of time of messages in Kafka and filtered them using the same rules that were used for Spark application. The amount of filtered messages using python and Spark was equal.



```
6377328352", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6313373975", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6372823431", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6371404482", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
8526995224", "OriginRealm" : "epc.mnc008.mcc450.3gppnetwork.org", "Timestamp" : "2018-03-02 ...
jinRealm" : "hss.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2018-03-02 05:56:10....
4260388649", "OriginRealm" : "MME.MSK.EPC.MNC001.MCC250.3GPPNETWORK.ORG", "Timesta...
6380875780", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
jinRealm" : "hss.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2018-03-02 05:56:11....
jinRealm" : "hss.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2018-03-02 05:56:11....
5114164451", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6314758390", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6173316203", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
5790907552", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6312393908", "OriginRealm" : "epc.mnc004.mcc452.3gppnetwork.org", "Timestamp" : "2018-03-02 ...
6313404839", "OriginRealm" : "hss.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "20...
6372865085", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
7011841214", "OriginRealm" : "mme.ural.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6374479044", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6312592740", "OriginRealm" : "mme.msk.epc.mnc001.mcc250.3gppnetwork.org", "Timestamp" : "2...
6505180848", "OriginRealm" : "MME.SIB.EPC.MNC001.MCC250.3GPPNETWORK.ORG", "Timesta...
```

Fig. 2. Result messages in Kafka application after Spark job implementation

The model has proved the possibility of real-time usage of new cellular network monitoring service for Big Data processing. Apache Spark application might be used for many other real-time models and can be considered a real-time Big Data hub engine, which will process data according to any of the needed business logic and data protocols.

D. Model advantages

The comparison with traditional cellular network monitoring system and batch processing of data shows the advantages of the model based on Apache Spark. In total, the benefits of the model can be described as follow (Fig. 3).

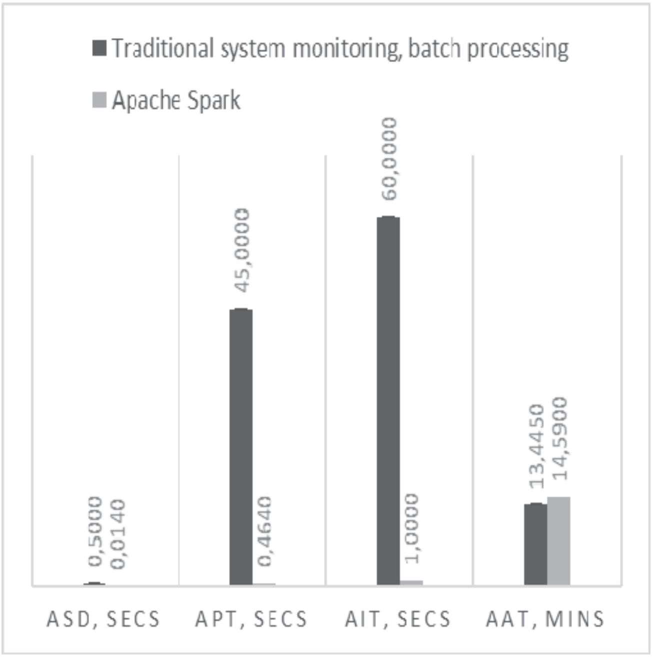


Fig. 3. The performance of the roaming users model implemented on the basis of Apache Spark and traditional batch processing

The comparisons show that Spark streaming has many advantages comparing with usual batch streaming:

- average scheduling delay (ASD) for batch processing is much longer than Spark delay. Apache Spark streaming runs its jobs with only 0.015 seconds delay, while traditional batch processing has 0.5 seconds delay in average;

- average processing time (APT) shows that the same amount of data might be processed in 45 seconds intervals, while Spark streaming process data in 0.464 time intervals. It is achieved because Spark jobs runs each second, and the data processing is really fast, in-memory and efficient;

- average interval time (AIT) between Spark jobs is 1 second, while interval between batch jobs is usually 1 minute. Batch processing cannot run faster because of overheads before job start. Each start of job takes some additional resources and needs some time to start job itself. For batch processing it is larger than for streaming;

- average available time (AAT) for decision makers to trigger roaming users shows that time to make a decision for decision makers about some action against "caught" users is larger with Spark due to its faster computation comparing to the traditional batch processing. Usually telecom providers are interested in users with no more than 15 minutes delay when the event occurred that is the user crossed the country border and this event has been caught by system. After we have this event in messaging system, all the rest is depending on us – how fast we process data and filter it for triggering and sending notifications. Thus, if we consider average time between appearance of event in messaging system and this event filtered - Spark shows only 1 second delay on average, while traditional batch requires more than 1 minute and 15 seconds. It means that decision makers can have more time to understand this user, his behavior and decide on sending any notifications.

E. DBDF benefits and comparisons

In total, new DBDF framework has the following advantages over traditional monitoring systems (Table III):

TABLE III. DBDF VS TRADITIONAL MONITORING SYSTEM

metric/system	**Traditional system monitoring**	**DBDF**
Scalability	low	high, not limited
Reliability	low	high
Speed and performance	low	high
Amount of possible use cases	usually alone	not limited
Data access	strict, within department	not strict, within company
New hypotheses checks	not possible, not enough data	easily, all monitoring data

Thus, all DBDF components are scalable and with addition of new node in a cluster more performance is available.

- The storage and processing memory are scalable for all DBDF components that represents a significant advantage over traditional standalone monitoring systems. The metric is important because of constant traffic growth worldwide.
- The reliability of the solution is explained by the fact that all data are replicated in a cluster that makes framework reliable. In case of failures of some nodes data are not lost.
- Speed and performance shows the huge difference. Because of the cluster mode and in-memory computations, DBDF is processing data very fast, while traditional standalone systems are usually performed worse.
- Amount of possible use cases is not limited with DBDF – all monitoring data are collected and stored, and many new use cases can be created and discovered. In traditional system usually one system is covering one use case, or one department. With DBDF, new use cases can be easily implemented with all company departments based on processing rules (online streaming) or new hypotheses validation (offline streaming).
- Data access is usually strict in traditional monitoring systems, while with DBDF all departments can have access to all monitoring data and achieve synergy effect all together. It means that departments can work together for new use cases adaptation and verification.
- New hypotheses checks are almost not possible with traditional systems because of not all monitoring protocols are presented in place. In contrast, DBDF open up new horizons with petabytes of data exploration.

CONCLUSION

The adaptation of new DBDFCN in telecom provider environment and deployment of proposed architectural components will help to achieve effective, reliable, scalable, speed and secure CN monitoring data processing. In CN high volume data streaming it is extremely important to build a powerful framework for data processing, aggregation, enhancement, enrichment and storage. The proposed distributed framework is fully capable with high-loaded CN data streams and can be a foundation for future models creation, making sure that all data are reliably saved and not lost.

The proof of concept was achieved by creation of near real-time Big Data model for roaming users detection with processing performance above 60 000 events per second. The model has been created using Apache Spark application and the adequacy of the model was checked by the python programming language. The created model has revealed that Apache Spark is capable of handling thousands and even more events per second and may be considered as a foundation for real-time Big Data hub creation.

The comparisons of the model and DBDF with traditional standalone monitoring systems demonstrates many benefits of DBDF framework such as its scalability, reliability, speed and performance, possibility to check new hypotheses. Apache Spark streaming with example of roaming user detection showed that it has less delay, less processing time and more time for decision makers comparing with traditional batch processing.

REFERENCES

[1] https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide (2017). Number of mobile phone users worldwide from 2013 to 2019 (in billions), Accessed on: 2017-10-11.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] http://www.3gpp.org/about-3gpp. Accessed on: 2018-01-31. K. Elissa, “Title of paper if known,” unpublished.

[3] https://www.itu.int/en/itutelecom/Pages/default.aspx. Accessed on: 2018-01-31.

[4] Darema, F. (2004). Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements. International Conference on Computational Science. M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 662–669, 2004. © Springer-Verlag Berlin Heidelberg 2004.

[5] Marz, Nathan and Warren, James (2015). Big Data: Principles and best practices of scalable realtime data systems, 1st ed.. Manning Publication Co., 2015.

[6] The Lambda architecture: principles for architecting realtime Big Data systems, blog post by James Kinle. Available at: http://jameskinley.tumblr.com/post/37398560534/the-lambda-architecture-principles-for Accessed on: 2018-02-05.

[7] Lambda Architecture: A state-of-the-art, post by Pere Ferrera. Available at: http://www.datasalt.com/2014/01/lambda-architecture-a-state-of-the-art/ Accessed on: 2018-02-05.

[8] A. Suleykin, P. Panfilov (2017). The Simulation-Based Smart Management Approach for Cellular Network Operation and Planning, in: Annals for DAAAM for 2017 & Proceedings, DAAAM International, Viena, 2017, pp.0423-0432.

[9] Big Data Analytics for Wireless and Wired Network Design: A Survey. Computer Networks 132:180-199 · January 2018. Accessed on: 02.02.2019.

[10] Big Data Analytics in Mobile Cellular Networks. IEEE Access 4:1985-1996 · May 2016. Accessed on: 02.02.2019.

[11] Real-Time Big Data Processing Framework: Challenges and Solutions. Appl. Math. Inf. Sci. 9, No. 6, 3169-3190 (2015). Accessed on: 02.02.2019.

[12] http://window.edu.ru/catalog/pdf2txt/503/80503/60870, p. 1-20. Accessed on: 2018-03-21.

[13] https://kafka.apache.org/documentation.html#introduction. Kafka 1.0 Documentation. Accessed on: 2018-02-04.

[14] http://spark.apache.org. Apache Spark. Accessed on: 2018-02-04.

[15] https://databricks.com/spark/about/. Accessed on: 2018-03-08.