*International Journal of Engineering & Technology*, 7 (4.36) (2018) 197-208

International Journal of Engineering & Technology

*Website: www.sciencepubco.com/index.php/IJET*

***Research paper***

# Technology and Architecture for a System of High-Speed Sensor Data Stream Collection and Processing

**Vladimir V. Kopytov, Pavel V. Kharechkin, Vladimir V. Naumenko, Aleksey V. Savartsov, Oleg V. Dorofeyev, Mikhail V. Batashan**

*LLC Infocom-S, Stavropol, Russia*
**Corresponding author E-mail: p.harechkin@infocom-s.ru*

## Abstract

**Objective:** The objective is to address the challenges of monitoring process facility and environmental parameters, which can be analyzed to anticipate dangerous and critical conditions.
**Methodology/approach:** This article proposes the technology and architecture for a system of high-speed stream data collection and processing, which combines the advantages of both the cloud and fog computing models for data collection, storage and processing.
**Conclusion:** The proposed technology and architecture for a system of high-speed stream data collection and processing make it possible to adapt to various monitoring and situation control challenges and can be used to set up centers for processing monitoring data of different levels.
**Originality/value:** The originality of the proposed technology and architecture consists in the application of a set of universal programming solutions aiming to set up a data processing center. Such a center would require a minimum amount of work related to designing an automated data collection system and to developing additional software. Furthermore, it will provide ample opportunities for further scaling and expanding its functionality.

***Key words:*** *facility monitoring, environmental monitoring, fog computing, cloud computing, information sensors.*

## 1. Introduction

One of major factors behind growing amounts of information in the world is an increased share of automatically generated data. Research conducted by the International Data Corporation (IDC) [1] projects an increase of automatically generated data by more than 40% of the overall data by 2020 as compared to 11% in 2005, and the Open Fog Consortium [2] estimates the amount of data to grow up to 2.3 exabytes. Over 21 billion devices are projected [3] to get Internet access by 2020 mostly due to the rapidly growing and expanding IoT or Internet of Things.
Of special importance among various monitoring devices are information and measurement sensors used to monitor process facilities and the environment, to project the emerging dangerous and critical conditions and to prevent emergencies [4]. The need to obtain full information on surveillance targets facilitating timely decision-making inevitably leads to the increased amount of data obtained from information sensors. As a result, this affects the requirements imposed on data collection systems and the possibility of assessing situations. The main sources of such data are Earth remote sensing satellites, GPS, mobile devices and measurement sensors, among others.
Consequently, a pressing need exists to update stream data collection and processing methods, which requires testing or new technologies and architectures to set up data processing centers (DPCs).
Currently, these tasks are accomplished by means of cloud technologies using software tools in a cluster-based architecture for data processing centers. The increasing amount of stored and transferred data, however, prevents cloud architectures from achieving the required capacity output. This challenge can be addressed by adopting a technology known as cloud computing, which ensures effective, safe and secure interaction of a multitude of devices between themselves and with local and cloud DPCs.
The notion of cloud computing appeared in 2012 when Cisco published an article entitled *Fog Computing and Its Role in the Internet of Things* [5]. In 2015, ARM, Cisco, Dell, Intel, Microsoft and Princeton University established the Open Fog Consortium [2] with a view to develop common approaches to the implementation of the fog computing technology. The first technical specifications describing fog computing architecture dates back to 2017.
Fog computing architecture allows users to transfer computations, storage, connection as well as management and decision-making systems to terminals that have limited resources and are directly linked to the physical world beyond the cloud. This is why the joint use of cloud and fog computing seems an optimal solution in terms of performance, security, scalability and cost minimization.
This article proposes a technology and architecture that combines the advantages of both the cloud and fog computing models for data collection, storage and processing and adapts monitoring and situation management challenges. Another major feature of the proposed technology is the possibility of setting up new DPCs and of expanding, both functionally and technologically, the existing and operating automated management and control systems.

## 2. Materials and Methods

### 2.1 A diagram of high-speed stream data collection and processing in situation control systems

Before describing the proposed technological solutions, the authors suggest to examine the formal description of high-speed stream data collection and processing in situation control systems presented as a conceptual diagram (see Fig. 1).

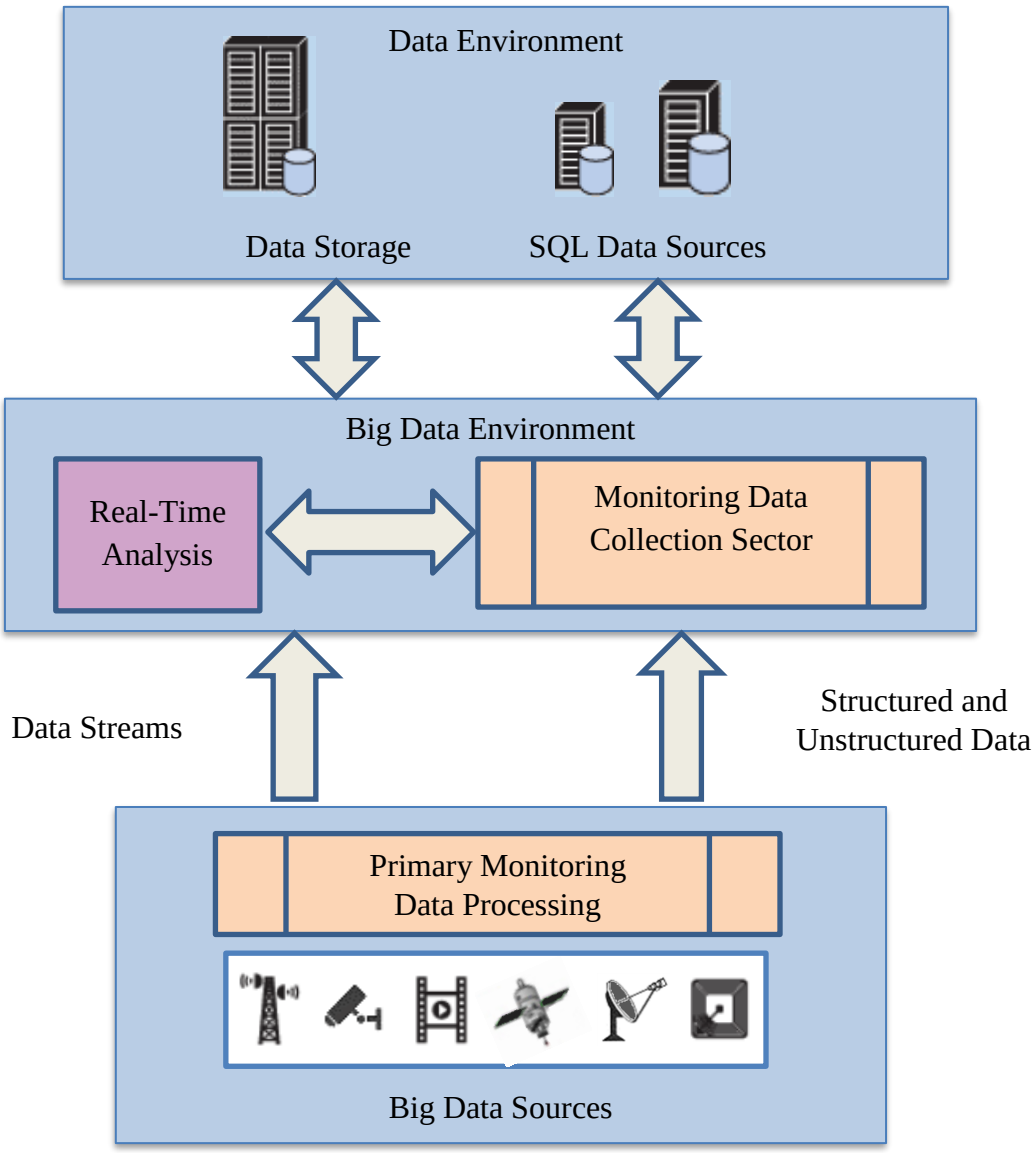


**Fig. 1.** Diagram of high-speed stream data collection and processing in situation control systems

The following three conceptual levels represent the diagram of high-speed stream data collection and processing from diverse information sensors:

- Big Data sources;
- Big Data environment;
- data environment.

The level of Big Data sources contains the following:

- information sensors generating large amounts of structured and unstructured stream data;
- communication channels and network equipment integrating stream data coming from information sensors for their further transmission to the next level;
- software and hardware tools (computing machines) performing the parallel primary digital processing of stream data.

The Big Data environment provides the opportunity to create a monitoring data collection sector for high-speed processing of structured and unstructured data coming from diverse information sensors in real-time mode and for their further research by using the methods and algorithms of predictive analysis, projection and analysis.

The Big Data environment in the monitoring data collection sector tackles the following tasks:

- preliminary data processing in real-time mode resulting in extraction and transformation of diverse structured and unstructured data for the purposes of their analysis as well as of comparison, detection and storage of key metadata;
- parallel data recording into a distributed system for data storage which ensures high-speed recording of Big Data streams coming from information sensors;
- research on Big Data coming from information sensors to high-performance distributed computing systems by using the methods and algorithms of predictive modeling, projection and analysis;
- presentation and visualization of the results of research on monitoring data for decision-making support related to monitoring objects and processes.

The level of data environment ensures the infrastructure of top-level applications and is aimed at aggregating the results obtained

by the Big Data environment in the processing and research of monitoring data and at supporting decision-making based on the predictive modeling of dangerous situations.
The data environment includes the following:
- data storage that aggregates the results of monitoring data processing and research;
- connectable SQL data sources necessary to support decision-making.
The following are SQL data sources:
- automated management and control system in place on the monitoring object;
- SCADA systems in place on the monitoring object;
- external automated systems and services that supply available data;
- external systems and services for projecting the evolution of emergencies.
In the proposed technology, the Big Data environment can be set up using either the cloud infrastructure of DPCs only or an integrated system comprising the cloud infrastructure and the fog computing environment. What follows is an examination of the architecture for DPCs used in this technology.

### 2.2 Architecture for Data Processing Centers

A data processing center is a high-speed data collection and processing cluster that is part of high-performance computing clusters [7] and is used to accomplish the following tasks:
- use of various techniques to process sensor data streams stored in a time series database;
- scheduled data processing;
- interactive management of sensor data processing.
The architecture of a typical computing cluster is a group of joint operations servers sharing the same data warehouse. Depending on the task, the high-speed data collection and processing cluster shall also include software for collecting data from diverse measuring devices.
Therefore, the architecture for a high-speed data collection and processing cluster can be described as four interacting base subsystems (see fig. 2):
- sensor data processing subsystem;
- data storage subsystem;
- cluster management subsystem.

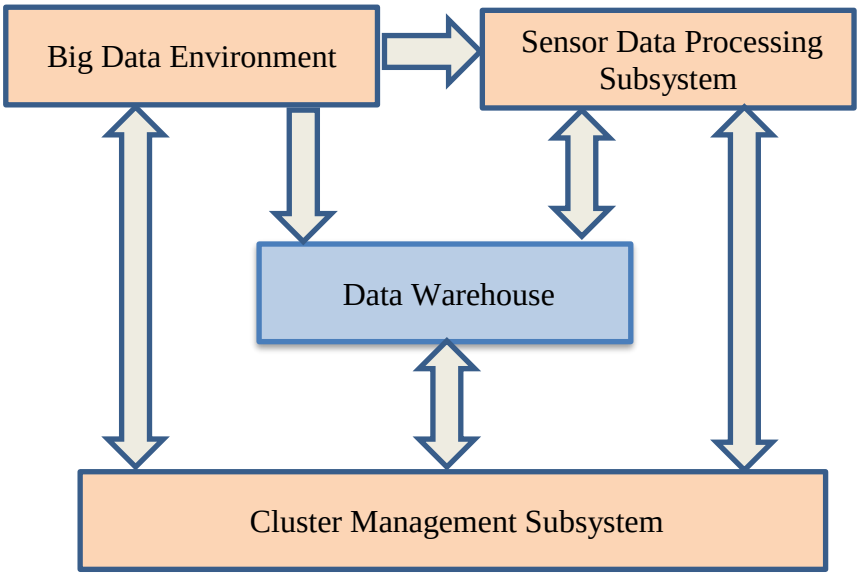


**Fig. 2.** Architecture for a high-speed data collection and processing cluster

The sensor data collection subsystem is a bridge between sensor device terminals and data processing and storage components and provides a common interface for their connection to a computing cluster, i.e. an enterprise service bus, by means of device drivers or external monitoring systems. This study understands device drivers as software serving as a link between the enterprise service bus and a device (or an external monitoring system) and ensuring the compliance of the device's interface with that of the enterprise service bus.
The sensor data processing subsystem comprises tools allowing users to adopt various algorithms for sensor data stream processing, which are stored in a time series database, and for scheduled data processing.
A data warehouse should also meet the requirements to ensure horizontal scalability of DPCs depending on the number of connected sensors and the amount of sensor data requiring the application of the distributed data storage technology.
The cluster management subsystem provides mechanisms for interactively managing sensor data processing and, specifically, computing task scheduling.
Given this architecture, research was conducted on the following software classes to achieve the objectives defined:
- software designed to aggregate data from diverse devices;
- systems for managing distributed databases;
- software for unfolding and managing distributed data processing.
The choice of software is based on the analysis of state-of-the-art solutions in terms of their significance to achieve the stated objectives [8].
Software designed to aggregate data from diverse devices belongs to the software class known as a message broker, whose primary aim is to ensure interaction between different program modules within the same protocol. This is why analysıs and research were conducted on the following popular platforms that process large amounts of data and allow users to work within cluster architectures:
- Apache Kafka;
- Apache ActiveMQ;
- Mosquitto MQTT;
- RabbitMQ.
Apache Kafka [9] is a high-speed distributed platform for exchanging large amounts of highly dense messages between the components of a program system, implementing in real-time mode a publish-subscribe-based messaging mechanism for data transmission.
The following are the main functional advantages of Apache Kafka:

- high-speed stream data processing: one compute node is capable of processing over 100,000 messages per second, with an overall volume being over 10 MB per second;
- linear scaling: a computing cluster is expandable up to thousands of servers with no downtimes in the data processing system;
- a user-friendly interface easily integrated into external and interactive applications;
- sufficient flexibility: easy setups, configuration and administration, full technical support by the producer;
- high security: messages in a computing cluster may be replicated and the replication level can be configured in settings; each node may store terabytes of data on the hard disk for further batch processing with no loss in performance;
- low delay time due to the division of subscribers into groups;
- high fault tolerance: stream data are automatically balanced in case one or several cluster nodes fail; additionally, messages are synchronized between interacting data centers.

The only detected deficiency of Apache Kafka is the low compatibility of interfaces when switching from one version of Apache Kafka to another.

Apache ActiveMQ [10] is open source software acting as a mediator to transmit and process messages in a distributed environment in asynchronous mode and based on Java Message Service (JMS), although it is compatible with other protocols and platforms. As a rule, Apache ActiveMQ is used in service-oriented projects as a secure message queue processing system.

The following are the main functional advantages of Apache ActiveMQ:

- relatively high-speed stream data processing;
- easy integration with other components and interacting systems due to interface support for different platforms and programming languages;
- good scalability: data processing tasks are automatically distributed across cluster nodes and streams in every node;
- high degree of data loss protection: temporary data storage on hard drives, data caching and logging;
- high fault tolerance (recovery from failures): if a handler is down, tasks are reassigned to other handlers;
- secure communication and messaging owing to SSL protocol support.

Among the deficiencies of Apache ActiveMQ are the following:

- sudden drops in Apache ActiveMQ's performance when working with databases;
- intricate asynchronous data stream processing functions that can be performed only by highly experienced developers;
- according to the expert community, Apache ActiveMQ contains a significant number of errors in its source code, resulting in functional risks.

Mosquitto MQTT [11] is a lightweight protocol based on top of TCP/IP and designed for messaging between devices using the publish-subscribe pattern. IBM, Microsoft and Amazon are among the major cloud service providers supporting this software protocol. MQTT requires a data broker, which is a program serving as a TCP server with a dynamic database. The central idea of this technology is that all devices send and receive data to/from the broker only.

The following are the main functional advantages of Mosquito MQTT:

- easy to use and manage: the protocol is a program module with no superfluous functionality that can be easily built into any complex system;
- compatible with mobile sensors, including navigation controllers;
- capable of maximally unburdening the network architecture, since transmitted messages contain only useful information;
- stable and secure: the protocol ensures secure messaging in case of continuous connection disruptions and other online issues;
- universal: the protocol imposes no restrictions on the format of message traffic.

Among the deficiencies of Mosquitto MQTT are the following:

- insufficient system scaling potential: the protocol limits the density of messaging depending on its tool;
- as a purely program technology, Mosquitto MQTT does not have long-term data storage, since it contains no intermediary disk warehouse, in technological terms;
- Mosquitto MQTT is bad at processing alarm events and, consequently, clients do not receive updated information about the communication line status.

RabbitMQ [12] is a distributed message queue processing server based on the extended network protocol AMQP and designed for messaging between publish-subscribe network applications. Rabbit MQ supports multiple platforms and interacts well with various programming languages. In addition to being a classical message broker, RabbitMQ supports the remote procedure call system (RPC), thus ensuring feedback between message senders and receivers.

The following are the main functional advantages of RabbitMQ:

- high fault tolerance: if the server did not shut down properly, queued data are not lost and, after restart, processing continues where it left off;
- fully open: RabbitMQ is licensed under the Mozilla Public License, as is the case with AMQP, whose libraries are available in all major programming languages and platforms;
- horizontally scalable: the system status is automatically replicated between cluster nodes; in most cases, two or three disk nodes are sufficient, the remaining ones being spared from having to work with the disk subsystem to improve performance;
- universal: RabbitMQ supports many protocols, client libraries and plugins;
- resource-intensive: there is no limit to the number of messages placed on queues in storage; this said, the messaging server can be located far from both the source of messages and the consumer.

The following are the main functional advantages of RabbitMQ:

- absence of timeout-based message processing: if the queue runner freezes due to a code error or for some other reason, the message will be not transmitted to another runner until the frozen one breaks connection;
- large amount of service data per message: the memory footprint for the same number of messages can differ markedly from launch to launch.

In choosing a messaging platform, special attention was given to data rates and security. This is why Apache Kafka with its ZooKeeper-based advanced fault-tolerant technology was selected to aggregate data from diverse devices [13].

Today, systems for managing distributed databases use the NoSQL technology [14], characterized by no superfluous sophistication, high capacity and accessibility, immunity to data partitioning as well as unlimited horizontal scaling. NoSQL does not ensure data consistency [15], which is not required for storing time-series sensor data; it has, however, the following important features to address issues related to sensor data collection:

- recording and storage of large amounts of data;
- real-time data processing;
- quick query evaluation;
- fault tolerance support.

Storage of large amounts of data means that the system should handle the increasing amounts of data, i.e. to be easily and linearly scalable.

Quick query evaluation means that queries are to be evaluated with maximum speed and this speed is to be minimally dependent on the growing amount of data collected.

Fault tolerance support means that service performance should not deteriorate in case of failures and disruptions, which is particularly difficult given the large amounts of data and their real-time processing.

Today, the most popular NoSQL product is Hbase, a database based on Apache Hadoop's distributed file system, HDFS, which shows high data processing rates due to data locality [16]. OpenTSDB [17], an application on top of Hbase, ensures time-series storage in the most user-friendly BigTable format.

Distributed data processing in a computing cluster comprises data processing management tools within cluster architecture as well as the component parts of programs or program units by using methods for joint data processing of spatially distributed information sensors and prediction.

Distributed data processing can be presented in two ways:

- independent development of an application by using methods for joint data processing of spatially distributed information sensors and prediction with functionalities that allow users to work in cluster architecture;
- use of a ready-to-use computing environment, which requires writing individual program modules using methods for joint data processing of spatially distributed information sensors and prediction.

To ensure the high security and accessibility of a computing cluster, users may employ resource management tools to plan sensor data processing and computing resources allocated to them.

Resource management tools of a computing cluster may be implemented independently as the individual application units, which either use methods for joint data processing of spatially distributed information sensors and prediction or are based on existing open program solutions.

Given that Apache Hadoop-based solutions have been selected as a data storage system, it may be appropriate, when to use Yarn [18], a resource manager that is part of Apache Hadoop and fully compatible with Apache Spark, as a system for managing the resources of a computing cluster.

Below is the content and functionality of cluster program components ensuring high-speed data collection and processing in compliance with the architecture and technological solutions chosen in this study.

Fig. 3 shows a diagram of cluster stream data ensuring high-speed data collection and processing.

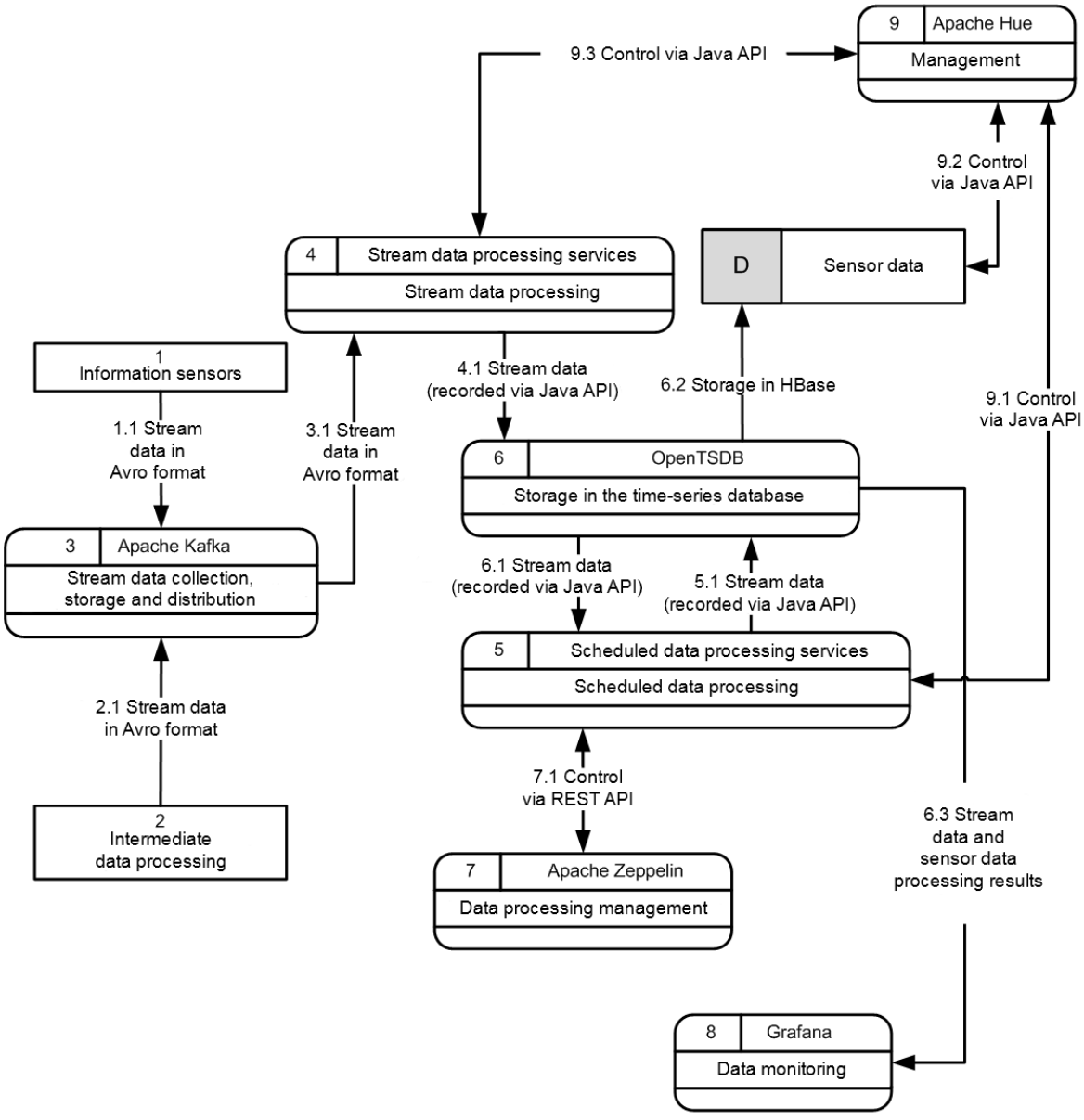


**Fig. 3** Diagram of stream data in a cluster to ensure high-speed data collection and processing

The following are the components of the diagram above:

- information sensors;
- intermediary data processing servers;
- Apache Kafka message broker;
- stream processing services;
- scheduled data processing services;
- distributed time-series database;
- Apache Zeppelin, a web-based notebook to work with Big Data;
- Apache Hue, a stream scheduling service for deferred computing;
- Grafana, a tool for metric analytics.

Data streams of the above diagram are the following:

- sensor data streams from end-user devices such as information sensors and intermediate data processing servers;

- sensor data streams between a message broker and data stream processing services;
- data streams received by data processing services via Java API;
- control commands via Java API and REST API.

Sensor device drivers are connected to a single enterprise service bus based on the Apache Kafka message broker, shaping stream data in Avro format. Among Apache Kafka's functions are transmission, storage and distribution of data received.

Sensor data stream processing services and scheduled data processing services are the component parts of applications used in task analysis, projection or predictive modeling, to be discussed later.

Sensor data stream processing services obtain data directly from Apache Kafka, operate inside the Apache Spark platform and use the Spark Streaming library.

Scheduled data processing services interact with OpenTSDB, a scalable time-series database, operate inside the Apache Spark platform and use the Spark SQL library.

The interface for computing cluster management and for displaying sensor data and their processing results is based on Grafana, a tool using Varnish to have access to OpenTSDB.

The web-based notebook Apache Zeppelin supports remote program control for scheduled processing services in Apache Spark via Livy, a web service for Spark.

Apache Hue, a tool that interacts with the time-series database and Apache Spark, is used to schedule workflows for deferred computing and data analysis in Apache Hadoop.

Fig. 4 shows the program components' location and their interconnections.

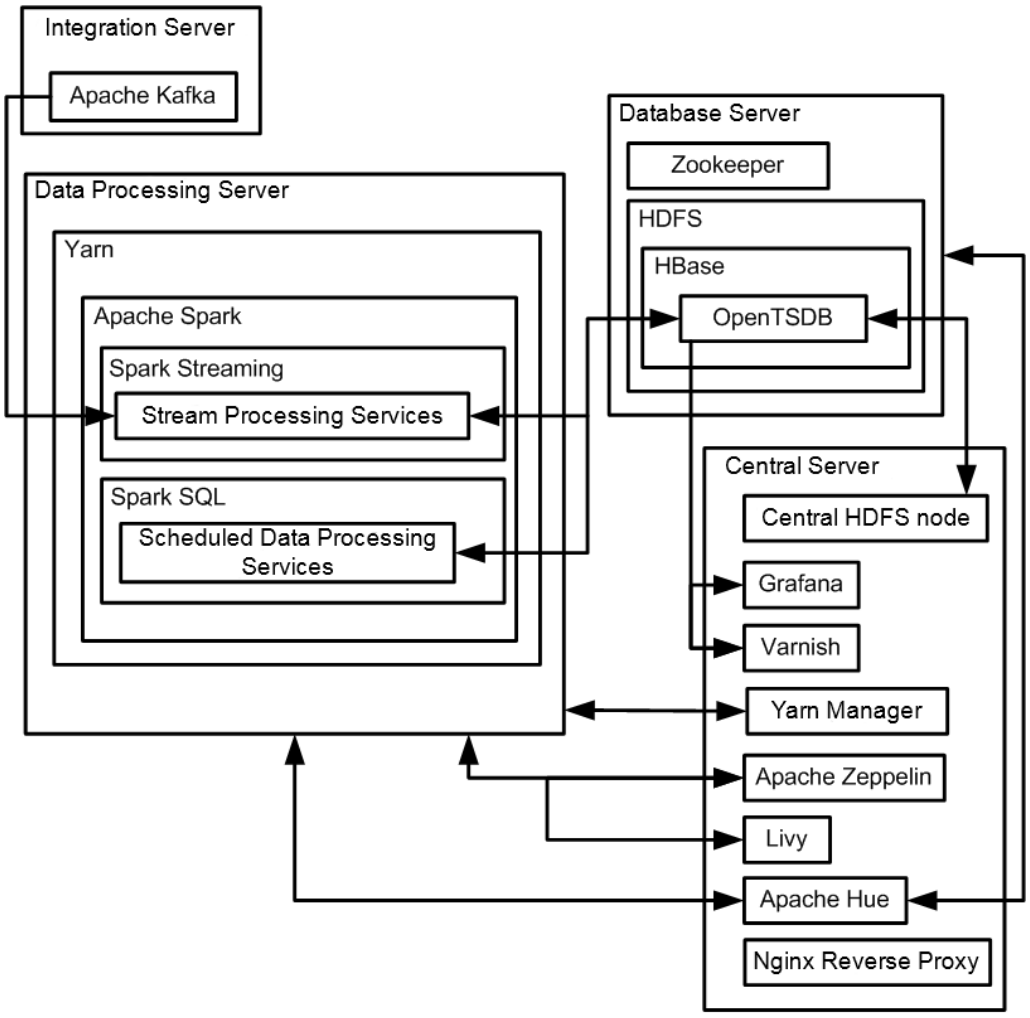


**Fig. 4** Diagram of the cluster program components' location to ensure high-speed data collection and processing

All program components are based on Linux Ubuntu and use three types of servers:

- database server;
- data processing server;
- integration server.

A database server is one of the nodes storing time series from sensor data and comprises the following program components:

- HDFS, a file system node;
- HBase, a region server;
- Zookeeper, a node;
- OpenTSDB.

A data processing server is one of the cluster nodes used to process data (real-time and/or scheduled stream data) and comprises the following program components:

- Yarn, a node;
- Apache Spark;
- Data processing services.

An integration server is designed for displaying the components of the Apache Kafka-based enterprise service bus.

The central server displaying the following components serves as a controller:

- Yarn, a manager;
- Central HDFS node;
- Grafana, a tool for metric analytics;
- Apache Zeppelin, a web-based notebook to work with Big Data;
- Varnish, a cache service;
- Livy, a web service;
- Apache Hue, a stream scheduling service for deferred computing;
- Nginx reverse proxy.

The proposed architectural solutions for organizing a high-speed data collection and processing cluster fully support the basic functionality and the localization of program modules using the data processing and predictive modeling methods and algorithms of cloud DPCs. What is special about the proposed architecture is the application of technological solutions ensuring high capacity and unlimited horizontal scaling.
Data processing on end-user devices will be discussed as part of the proposed technology using fog computing.

## 2.3 Aggregation and high-speed data processing from distributed information sensors

The basic elements of the proposed technology are device drivers or external monitoring system drivers as well as program applications used in task analysis, projection or predictive modeling.
The technology under investigation implies three data collection and processing options, depending on the following tasks:
- primary data collection without preliminary processing;
- preliminary data processing on interface devices;
- data collection and analytical processing using interface devices.

Primary data collection without preliminary processing implies that the data is processed only in a cloud DPC (cluster), while the drivers serve to transform the device's interface to operate with Apache Kafka (an enterprise service bus). The driver can be installed on both the servers of a DPC's enterprise service bus and separate devices outside the DPC.
Preliminary data processing on interface devices is that the device driver performs some data processing tasks (preliminary data processing), i.e. the data are recorded in the cluster storage after preliminary processing and the remaining data processing is performed by a cloud DPC. Intermediate data processing devices outside DPCs, in which a device driver is installed, reduce pressure on the DPC cluster.
Data collection and analytical processing using interface devices implies that data processing tasks are performed by external interface devices acting both as drivers and computing modules. At the same time, the device's software has to secure M2M connectivity with other devices, thus merging them into a cluster for joint task execution.
Each data collection and processing option requires a uniform technological solution adaptable for specific DPC tasks. Practice [19] shows that a possible solution may be the use of inexpensive single-board Linux computers [20] as devices for installing drivers such as Raspberry Pi (Raspberry Pi Trading, Ltd), Khadas VIM (Shenzhen Wesion Technology Co., Ltd) and Odroid XU-4 (Hardkernel Co., Ltd). As a result, device drivers partially carries out fog computing functions by taking some data processing functions outside DPCs.
In this case, device drivers shall be composed of the following functional units:
- interface unit with a monitoring device (information sensor);
- computing unit;
- interface unit with a DPC enterprise service bus (Apache Kafka).

Interface units with monitoring devices and interface units with DPC enterprise service buses perform data transformation tasks. At the same time, computing units have to allow execution of a set of standard functions for working with the time series of measurement data, such as calculating the average value of a function over an interval, exponential smoothing and Fourier transformation, or for making calculations according to a pre-determined mathematical function.
Software designed for task analysis, projection or predictive modeling shall be composed of a computing section including data stream processing services and scheduled data processing services as well as an interface allowing the user to configure a device driver in compliance with a specific computing task. Device driver settings include the following:
- configuring primary measurement data processing settings using mathematical time-series processing functions;
- configuring connections to a device (information sensor);
- configuring connections to Apache Kafka.

Fig. 5 shows the structure of applications used to perform analysis, projection and predictive modeling tasks and that of device drivers

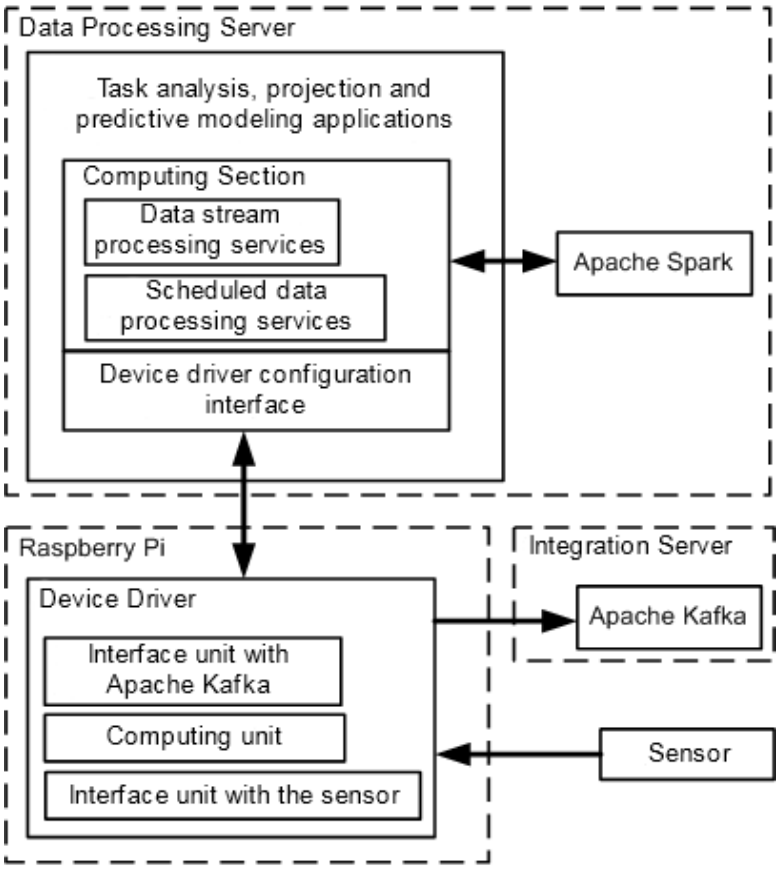


**Fig. 5**. Structure of applications for preliminary data processing on interface devices

This diagram makes it possible to perform data processing tasks on devices outside DPCs, thus undervolting DPC clusters.

Specialized software capable of creating a fog computing architecture in compliance with Open Fog specifications is required to perform distributed calculations accomplishing all analysis, projection and predictive analysis computing tasks outside cloud DPCs [6]. As an example, Cisco is now working on creating software for its network equipment, such as Ethernet routers, access point and IP video cameras [21], ensuring protected M2M communication to perform distributed calculations in the perimeter network. This will require, however, more expenditure on network infrastructure updating, which will create difficulties in setting up new DPCs.

This is why the optimal solution is to use single-board computers, as with preliminary data processing. As of now, the most attractive option for performing distributed calculations is Greengrass, a platform launched by Amazon [22]. Greengrass is a program module container launched on a Greengrass device and allowing devices to exchange information regardless of whether they are connected to an external network or not. Greengrass is fully compatible with most Linux platforms used in single-board computers [23]. The Greengrass platform uses AWS Greengrass, external software that manages distributed calculations on various devices.

In this case, applications used to perform analysis, projection or predictive modeling tasks comprise a computing sector, which includes integration services with AWS Greengrass and the data stream processing service, and an interface enabling developers to configure the parameters of computing tasks performed by Greengrass. The data stream processing service uploads the processed data from an AWS Greengrass cloud and stores them in the data warehouse of a cluster. In their turn, local Greengrass Core applications interacting with AWS Greengrass are deployed on interface devices, as well as the interface services of connected measurement devices (information sensors) that support Greengrass API.

Fig. 6 shows the structure of applications used to perform analysis, projection and predictive modeling tasks and that of device drivers.

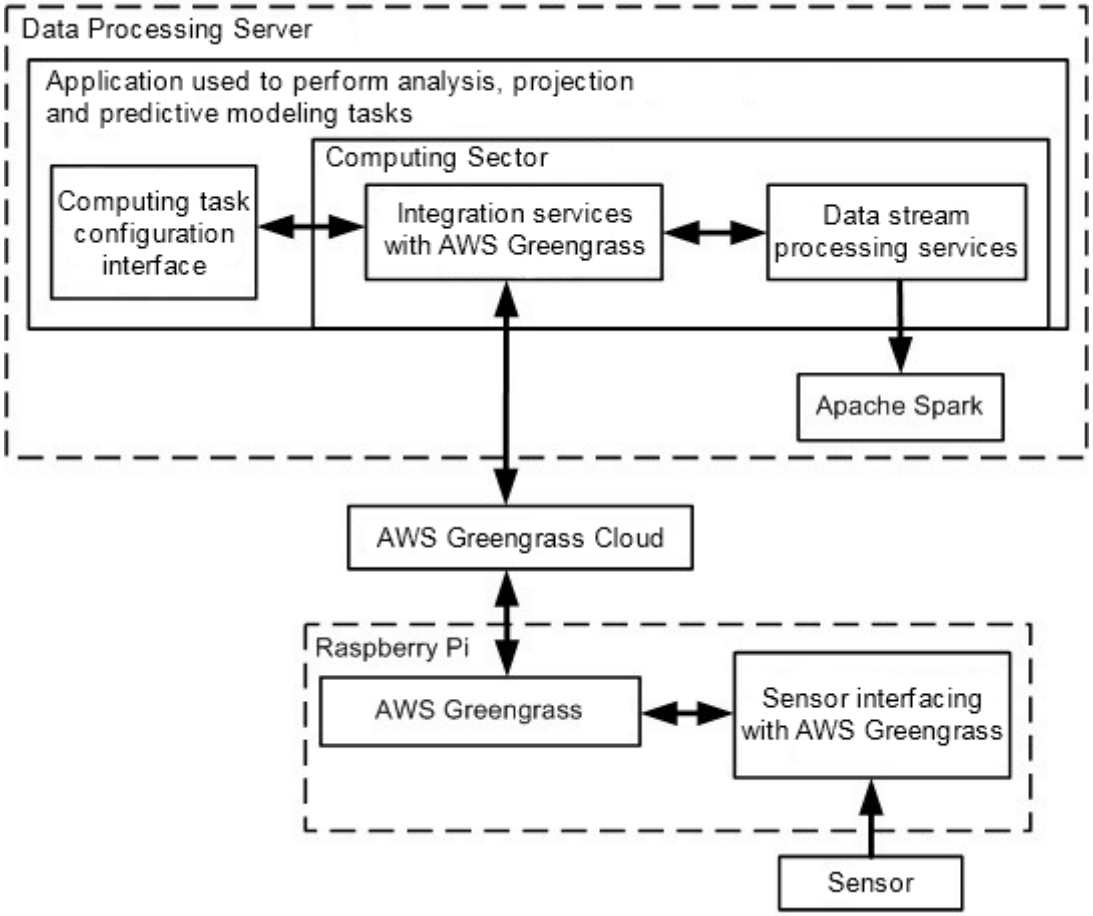


**Fig. 6.** Structure of applications for data collection and analytical processing using interface devices

The following are the major limitations to using distributed calculations with interface devices:

- Location of interface devices in the same subnetwork;
- Stable network connection and network capacity sufficient for performing computing tasks between devices, between each of the devices and AWS Greengrass cloud as well as between a DPC and AWS Greengrass cloud;
- Input data locality, i.e. only data obtained from information sensors connected to Greengrass devices can serve as input data for a computing task.

Consequently, using Greengrass as part of DPCs responds to the challenge of collecting and analytically processing data using interface devices without using DPC cloud resources.

## 3. Results

The developed technology and architecture was used to set up the software and hardware system for high-speed stream collection and processing of the Earth's ionospheric sounding data [24] as part of research on near-Earth space by assessing various parameters of Earth's ionosphere obtained from signals sent by the Global Navigation Satellite System (GNSS) [25]. These include full ionospheric electron content data (ECD), scintillation index, Rice distribution, mean square deviation of the small-scale variants of the ionospheric ECD, mean square deviation of phase fluctuations of a wave front and signal error rate, among others.

Data on Earth's ionospheric parameters are of interest to many technical and scientific applications. Global maps of ECD distribution and fluctuations are used to evaluate current radio weather, and many research studies have analyzed the impact of major seismic events on the ionosphere. The main objective of the system under investigation is to identify the geographic coordinates and to assess the linear dimensions of the ionosphere with intense small-scale heterogeneities based on the ongoing ECD received from GNSS receivers.

The main tasks of the system for high-speed stream collection and processing of the Earth's ionospheric sounding data are as follows:

- obtaining data from a GNSS receiver to monitor the ionosphere;
- dividing the data into parallel streams and their preliminary processing;
- recording data streams into a time-series database (OpenTSDB database);
- processing scheduled data to identify the geographic coordinates of an under-ionospheric point;
- visualizing the data processing results.

The hardware and software system for high-speed stream collection and processing of the Earth's ionospheric sounding data comprises the following:

- three 1480Q1 Depo Storm computing servers;
- NovAtelGPStation-6, a dual-frequency GNSS receiver;
- interface module based on Raspberry Pi 3 Model B.

The driver of a GNSS receiver is installed on the interface model with a view to read and pre-process monitoring logs and, then, to send the data to Apache Kafka, the enterprise service bus, at a speed of 0.8 Mbps. Fifty-six satellites collect data at a speed of 50 values per second for each data stream. Data stream processing covered an area of 4,000 geographic coordinates. Scheduled processing aimed at calculating the geographic coordinates of ECD focused on an area of 180,000 geographic coordinates. Grafana, a tool for metric analytics that is part of cluster program components (see Fig. 7 and 8), was used to visualize the data processing results and the satellites' current location. Research studies [25] and [26] provide a detailed description of the parameters, i.e. input data peculiar to ECD and output data obtained from the near-Earth space dimensioning calculations.


**Fig. 7.** Ionosphere parameter monitoring based on the evaluation of delays in trans-ionospheric radio wave propagation

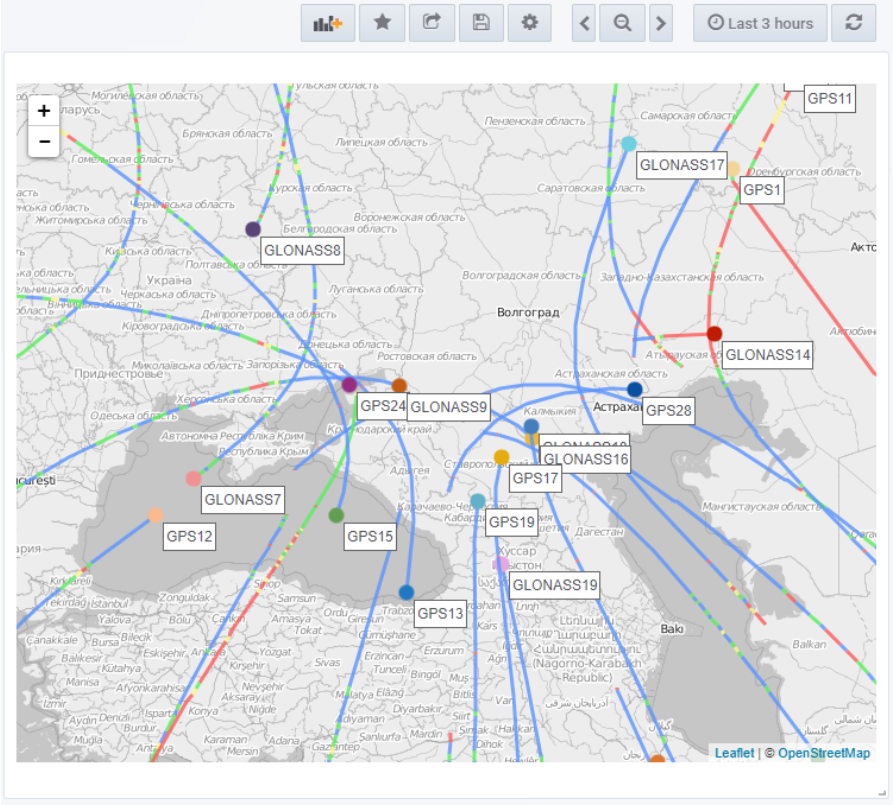


**Fig. 8**. Ionospheric condition monitoring system according to the satellites' current location and trajectory

Metric-based monitoring of the cluster's hardware component (CPU, random access memory, disk array etc.) and of data stream collection and processing (duration of tasks and duration of recording into the time-series database OpenTSDB) has been executed to control the computer system's performance (see Fig. 9 and 10) and to assess the effectiveness of the suggested architecture and technology.

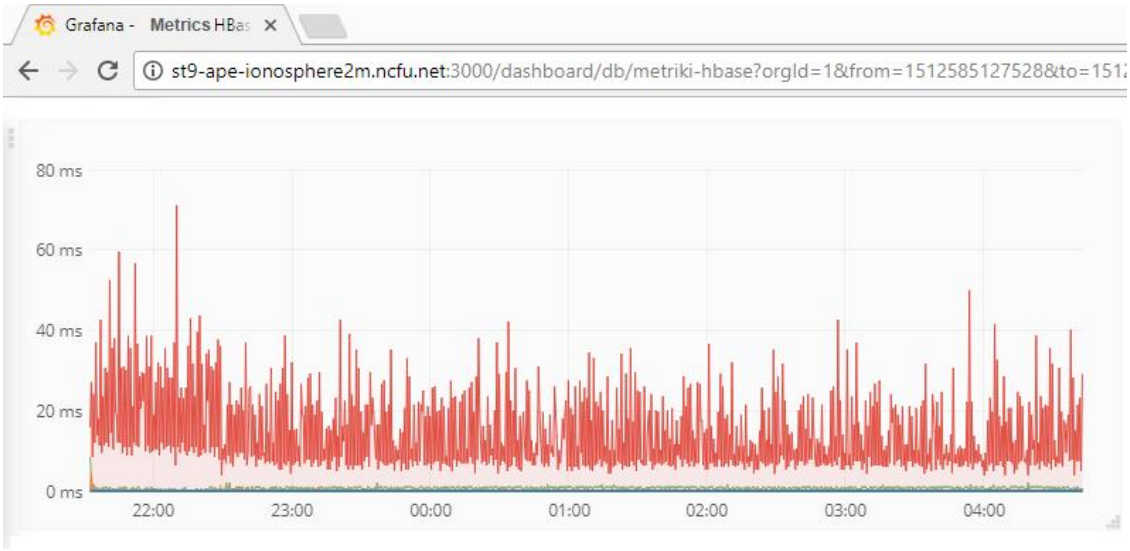


**Fig. 9.** Monitoring of the duration of tasks

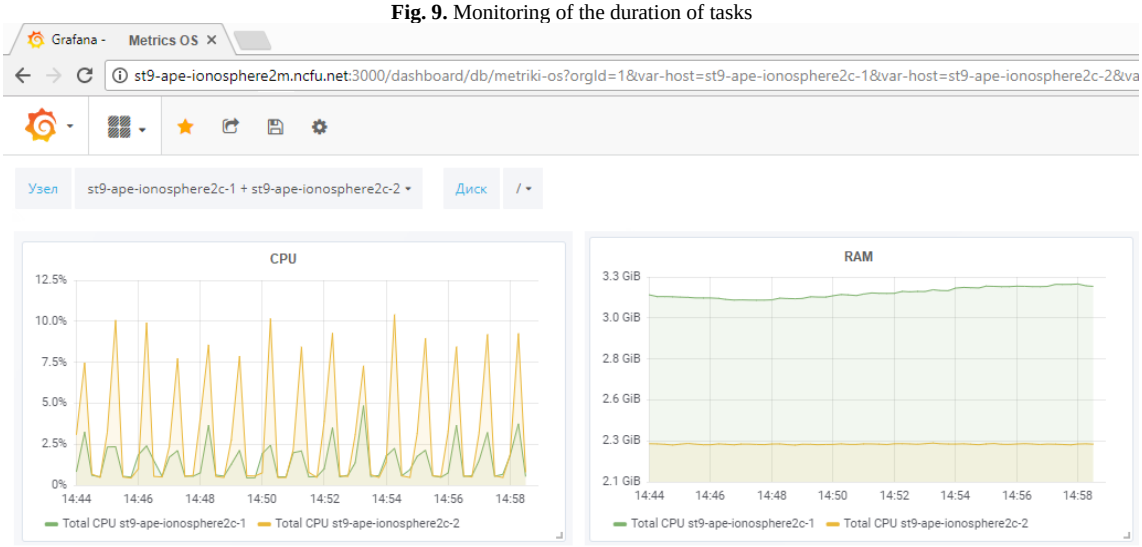


**Fig. 10.** Monitoring of the use of CPU and of random access memory during stream collection and processing of the Earth's ionospheric sounding data

In addition to completing the main tasks related to ionospheric electron content, the study assessed the effectiveness of the suggested architecture and technology when using preliminary data processing on interface devices based on the setup's following general performance indicators:

- average scheduled data processing time;
- average value of using the cluster servers' CPU;
- average value of using the cluster servers' random access memory.

The software and hardware system operated in the following two modes:

- using preliminary data processing, under which an interface device performed intermediary data stream processing by means of sliding window flow control; coordinate transformations; and calculation of two parameters, i.e. mean square deviation of the small-scale variants of the ionospheric ECD and mean square deviation of phase fluctuations of a wave front and signal error rate, which are required for further analysis of near-Earth space;
- without using preliminary data processing, under which the GNSS receiver's driver transforms only the primary data and records data streams into Apache Kafka (enterprise service bus).

Selection of computing tasks remoted to an interface device is made taking into consideration the hardware capability of the single-board computer Raspberry Pi3. The values of the setup's above-mentioned performance parameters are calculated using the data collected during 24 hours of the cluster's uninterrupted operation. Table 1 shows the results obtained.

**Table 1.** Results of the study on the software and hardware system for high-speed stream collection and processing of the Earth's ionospheric sounding in different data processing modes on interface devices

| No | Use of CPU cores, % | Use of random access memory, GB | Data processing time, ms |
|---|---|---|---|
| 1 | Without using preliminary data processing | | |
| | 6.4 | 2.3 | 95 |
| 2 | Using preliminary data processing | | |
| | 4.8 | 2.1 | 69 |

The results of the study on the performance parameters of the setup's software and hardware system show that using preliminary data processing on interface devices may considerably save the resources of a DPC's computing cluster. This is why this architecture for software and hardware systems will be adopted in further research studies using several GNSS receivers distributed by area, thereby expanding the area's extent, improving the accuracy of estimations and testing the variant using the external services of AWS Greengrass with a view to calculate all output parameters of near-Earth space.

## 4. Discussion

This article proposed the technology and architecture for a high-speed stream data collection and processing system, whose main advantages are as follows:
- ample opportunity for integration of diverse monitoring devices on the basis of the same data bus;
- use of cluster technologies for setting up DPCs ensuring high capacity and accessibility, immunity to data partitioning as well as unlimited horizontal scaling;
- use of the fog computing technology ensuring data collection and analytical processing using interface devices, which makes it possible to take data analytical processing functions outside DPCs;
- use of preliminary data processing on interface devices, when the specific nature of a task does not allow developers to perform distributed data processing on external devices.

Among the advantages of the proposed technology and architecture is the possibility of combining different technological solutions of centralized and distributed data processing aimed at adjusting to various issues in monitoring and situation control, thereby making the creation and exploitation process less time-consuming. The disadvantage of the proposed technology and architecture is its dependence on connection to external services to perform fog computing using interface devices. This is why further research on this topic is related to the development of open source software for Linux-based fog computing.

## 5. Conclusion

To sum up, the proposed technology and architecture for a system of high-speed stream data collection and processing is designed to deal with a wide range of issues relating to technological and environmental monitoring aimed at predicting emergencies. That said, the main advantage of the developed technology and architecture is their ability to adjust to different issues in monitoring and situation control and to be used in setting up centers for processing various monitoring data.

The obtained scientific and application results can be used in further research on the development of interface device software capable of performing distributed data processing without resortıng to external services and of reacting to changes in task flows occurring over time, while choosing the best method for processing data.

## 6. Acknowledgements

This study has been carried out as part of the Development of Means for High-Speed Processing of Information Sensor Data in Situation Control Systems, a project under the 2014-2020 Federal Targeted Program for Research and Development (identification number: RFMEFI57916X0135) funded by the Ministry of Education and Science of the Russian Federation.

## References

[1] J. Gantz, D. Reinsel. The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East–United States, IDC Country Brief. URL: http://www.emc.com/collateral/analyst-reports/idc-digital-universe-united-states.pdf

[2] OpenFog Reference Architecture for Fog Computing. URL: https://knect365.com/cloud-enterprise-tech/article/0fa40de2-6596-4060-901d-8bdddf167cfe/openfog-reference-architecture-for-fog-computing

[3] Rob van der Meulen. 6.4 Billion Connected "Things Will Be in Use in 2016", Gartner. URL: https://www.gartner.com/newsroom/id/3165317

[4] Tebueva F.B., Kopytov V.V., Petrenko V.I., Kharechkin P.V., Sidorchuk A.V. Method for Detecting and Eliminating Data Time Series Outlier in High-Speed Process Data Sensors. International Journal on Communications Antenna and Propagation (IRECAP), vol. 7, no. 7, 2017, pp. 603-612.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog Computing and Its Role in the Internet of Things" in Proc. 1st Edition MCC Workshop Mobile Cloud Comput., Helsinki, Finland, 2012, pp. 13-16.

[6] OpenFog Reference Architecture for Fog Computing. URL: https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_ Architecture_2_09_17-FINAL.pdf

[7] Furht, B. and Escalante, A. Cloud Computing Fundamentals. In Handbook of Cloud Computing, Springer, 2010.

[8] Manuel Díaz, Cristian Martín, Bartolomé Rubio. State-of-the-Art, Challenges, and Open Issues in the Integration of Internet of Things and Cloud Computing (Preprint), 2016. DOI: http://dx.doi.org/10.1016/j.jnca.2016.01.010

[9] Apache Kafka. URL: https://kafka.apache.org/

[10] Apache ActiveMQ. URL: http://activemq.apache.org/

[11] Eclipse Mosquitto (an open source MQTT broker). URL: https://mosquitto.org/

[12] RabbitMQ. URL: https://www.rabbitmq.com/

[13] Apache ZooKeeper. URL: https://zookeeper.apache.org/

[14] Dmitry Namiot. On Big Data Stream Processing. International Journal of Open Information Technologies, vol. 3, no. 8, 2015, pp. 48-51.

[15] Rick Cattell. Scalable SQL and NoSQL Data Stores, ACM SIGMOD. Record 39, no. 4, 2011, pp. 12-27.

[16] Kałużka, J., Napieralska, M., Romero, O., Jovanovic, P. Data Locality in Hadoop. International Journal of Microelectronics and Computer Science, vol. 8, no. 1, 2017, pp. 16-20.

[17] OpenTSDB (The Scalable Time Series Database). URL: http://opentsdb.net/

[18] Adam Kawa "Introduction to YARN". URL: https://www.ibm.com/developerworks/library/bd-yarn-intro/index.html

[19] Nandor Verba, Kuo-Ming Chao, Anne James, Daniel Goldsmith, Xiang Fei, Sergiu-Dan Stan Platform as a Service Gateway for the Fog of Things. Advanced Engineering Informatics, vol. 33, 2017, pp. 243-257.

[20] Catalog of 98 Open-Spec, Hacker Friendly SBCs. URL: http://linuxgizmos.com/catalog-of-98-open-spec-hacker-friendly-sbcs/

[21] Cisco IoT Networking. Deploy. Accelerate. Innovate. URL: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/brochure-c02-734481.pdf

[22] AWS Greengrass. URL: https://aws.amazon.com/greengrass/

[23] AWS Greengrass FAQs. URL: https://aws.amazon.com/greengrass/faqs/

[24] P. V. Kharechkin, A. V. Savartsov. System for High-Speed Collection and Processing of the Sounding of the Earth's Ionosphere // 3rd All-Russian Scientific Conference on the Fundamentals and Applications of Computer Technologies and Information Security, Taganrog, Russia, 2017, pp. 330-333.

[25] K.A. Katkov, V.P. Pashintsev, E.K. Katkov, N.N. Gakhova, R.P. Gakhov, A.I. Titov. Forecast Accuracy of Determining Pseudo Range in Satellite Navigation System Through Analysis of Data from Ionosphere Monitoring. Journal of Fundamental and Applied Sciences, vol. 9, no. 1S, 2017. DOI: http://dx.doi.org/10.4314/jfas.v9i1s.744

[26] V. P. Pashintsev, A. F. Chipiga, V. A Tsimbal, M. V. Peskov. A System for Determining Ionospheric Areas with Small-Scale Heterogeneities based on the GPS Monitoring Data // Izvestiya (News) of the Samara Scientific Center of the Russian Academy of Sciences vol. 18, no. 2(3), 2016, pp. 941-945.