УДК 004.78

ОСОБЕННОСТИ МОНИТОРИНГА МОБИЛЬНЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ

О. Панарин[1], И. Захаров[2]

[1,2] *Сколковский институт науки и технологий, г. Москва*

[1]O.panarin@skoltech.ru, [2]i.zacharov@skoltech.ru

Аннотация

Рассмотрена реализация подсистемы мониторинга систем обработки информации на мобильных платформах и ее применение на беспилотных автомобилях. В условиях беспилотной эксплуатации автомобиля предъявляются наиболее жёсткие требования к надежности систем обработки информации, принятию решения о готовности этих систем к эксплуатации и обеспечению анализа их возможных сбоев. Представленная система мониторинга pLOG сочетает в себе функционал записи событий операционной системы устройств и измерений параметров систем в реальном времени, при этом запись производится как файлы, так и в базу данных временных рядов (TSDB). При этом каждый сервер в составе системы обработки информации на мобильных платформах дублирует запись обо всех событиях в системе.

Ключевые слова: *сенсорные данные, распределенные системы, мониторинг*

MONITORING MOBILE INFORMATION PROCESSING SYSTEMS

O. Panarin[1], I. Zacharov[1]

[1]*Skolkovo Institute of Science and Technology, Moscow*

[1]O.panarin@skoltech.ru, [2]i.zacharov@skoltech.ru

Abstract

We describe the implementation of the monitoring for the IT systems at the core of the autonomous driving vehicle. The role of the monitoring is to assist in decision to

start the driving cycle and continuous assessment for the fitness to drive the vehicle. The requirements for the monitoring system with the increased resiliency and data replication make it sufficiently different from standard monitoring systems and warrant a unique implementation tuned for the autonomous driving requirements. The monitoring system combines the OS events and real-time measurements of sensor data. The information is stored in flat files for emergency access as well as in a Time Series Data Base (TSDB).

Keywords: *mobile systems, distributed systems, sensor data, monitoring*

INTRODUCTION

The autonomous driving is facilitated with information processing system analyzing data coming from many different sources (for example the LIDARs for the distance measurement, stereo-vision, global positioning, etc.) in combination with the embedded maps, real time object classification and decision taking (Neural networks, expert systems). The information processing system is built with modern high performance servers in ruggedized enclosure with water cooling. Combined performance of such system in autonomous vehicle may exceed 100 Tflop/s (e.g. for the single precision floating point operations) while the power consumption is in the order of 5 KW with contemporary technology. The performance is distributed between several systems usually serving one kind of data source for analysis each. For example, there is the LIDAR analysis system, system for stereo-vision of the right wing and so on, as well as a system that combines all the information and taking the steering decisions.

While specific implementation is vehicle dependent the common element in the contemporary systems is the high performance server running Linux OS with GPU accelerator to speed up the neural network operations. The information processing system in an autonomous vehicle is computational cluster consisting from such servers with a number of networks. In particular, there is a high speed network that shares all raw and/or analyzed sensor data and allows to store the data in a logging operation at a speed of 5 GB/s. The given performance is required and sufficient at current technological level (and with the current sensor resolution) for vehicle autonomous operation on one hand and to store the data for transfer to the laboratory and analysis off line. We continue to analyze the data storage requirements.

For completeness we mention another type of the event storage system in an autonomous vehicle to store data related to an emergency and/or crash analysis. This type of storage system should provide an absolute data integrity that may be required to analyze vehicle movement from the liability perspective. The speed or size of storage should be sufficient to store the last 30 seconds of vehicle movement before an aversive event. This subsystem is required for the SAE vehicle readiness level 3 and above [1] to cater for the insurance claims.

The main subject of this article is the subsystem dedicated to monitoring the servers and the networks they connect in an autonomous vehicle coupled with the storage and analysis of the monitoring data. We have reported previously an advanced implementation of the monitoring in autonomous vehicle [14] and extend the study in this article.

Before an autonomous vehicle may be allowed to move there should be a clear understanding that all subsystems work as expected, the sensors are collecting the data, servers are up and running and the vehicle is ready for the operation. During the operation any deviation from the expected behavior should be reported to the operator to reclaim the vehicle (level 2, 3 of SAE) or it should lead to an emergency/controlled stop (level 4, 5 SAE). During the regular operation monitoring systems is not critical and is necessary only if problems are detected. If there is a problem the collected monitoring information should be sufficient to analyze the situation.

Monitoring of servers in Datacenter setting is well understood with a number of successful implementations. For example, there are the most flexible and adoptable Nagios [2] or Zabbix [3] monitoring systems with highest deployment track record for Datacenters. The autonomous system monitoring has a similar task with a number of peculiarities.

Here is the list of features unique to the monitoring in autonomous vehicles:

- The reliability requirement must exclude any single point of failure and allow for multiple failures in the monitoring system. Of course, even a single failure in a non-critical component, such as the monitoring system may lead to the decision to stop the vehicle. However, the analysis of the failure has to be performed with the same monitoring system; therefore, the monitoring must be especially robust and allow for multiple points of failure. Using this monitoring system to analyze failures

may lead to the resolution of the problem “in the field” and allow to continue vehicle operation

- There are non-standard (i.e. not typically found in the Datacenter) sensors in the system. In the typical Datacenter monitoring systems the sensors are attached with IMPI and SNMP protocol; in a vehicle there are additional standards (such as the COM-Bus and networks like 100Base-T1) that must be integrated in the monitoring system
- The autonomous vehicle cluster has small number of servers (nodes). The Datacenter monitoring systems scale to 100s and 1000s of nodes while the autonomous vehicle cluster is limited to about 10 servers. This peculiarity allows to implement monitoring systems with smaller complexity and aids to increase the robustness
- There is typically no need to keep the monitoring data for duration beyond the immediate run or a few runs of the autonomous vehicle. This relaxes the storage requirement and allows to increase the robustness by duplication of the incoming monitoring data on all the nodes
- There is a need to provide the monitoring data to the operator through different interfaces, such as the GUI or WEB-interface for a comfortable analysis by operator without special training, as well as harder to interpret data with an interface not featuring any graphics. This is necessary for casual analysis of the monitoring data, as well as for the emergency analysis to restore system functionality after a crash
- The monitoring system should adjust itself to the running environment without the need for a special configuration or tuning in an autonomous vehicle. We expect that there is a great number of autonomous vehicles where the monitoring system is deployed and it is not possible to follow up and tune each separate installation manually. With other words the monitoring system should run “out-of-the-box” and only in special conditions require analysis and tuning.

In what follows we consider how a monitoring system that fulfills all these requirements may be implemented for monitoring servers in an autonomous vehicle. Further we provide examples running this system. Before that we analyze competing strategies for building such monitoring system.

STRATEGIES FOR THE MONITORING SYSTEM

System under study — high performance cluster with nodes using OS Linux. There are a number of tools exist to monitor each individual system. For example, top, htop, nmon, vmstat, iotop, iftop, netstat, bmon, glances and so on (see the list in ref. [4]). All these utility programs use Linux statistics that is provided through the (pseudo-) file system /proc. For example, the processor load statistics can be read from the file /proc/stat, the memory usage from /proc/meminfo, network /proc/net/dev, storage /proc/diskstats and so on. During each read from these files Linux kernel driver is providing internal counters data. The only difference between the utility programs is how each structures the output and how fast the programs are refreshing the view for the user.

There several solution to monitor the whole cluster consisting of several servers. For example, Nagios, ganglia, zabbix, cacti and so on (see the list in ref. [5]). These programs obtain the data from the same interface /proc, but structure it in a way comfortable to judge the state of the whole cluster. Also, some of these programs (eg. zabbix) store the data in a database.

Typically, the autonomous vehicle would integrate monitoring solution based on the listed programs. The contemporary computational base uses Linux servers (clusters) for the implementation of the autonomy function in this functional testing phase, which is considered as an intermediate phase on the way to broad deployment of autonomous vehicles that will use a different computing hardware. Current Linux servers running in the vehicles should be replaced by specialized hardware suitable for massive application. Therefore, investment into the monitoring software at this stage may seem premature.

Our strategy is different. We suggest that the evolution of Linux clusters in autonomous vehicles will follow the specialization and miniaturization trend but will keep the current software interfaces. Following this hardware evolution the monitoring software will demand health management function which is not part and cannot be expected from the standard Linux monitoring programs. This transformation will go in hand with the integration of multitude of sensors that will provide the data for the health management. With the use of standard Linux interfaces the general purpose monitoring systems like Zabbix can be extended with additional functionality but the

complexity will grow substantially which is conflicting with the stated requirement. Therefore, on the specialized future computational platforms used to run autonomous vehicles the monitoring and health management software will also be special, even if it will continue use todays software interfaces. The proposed monitoring software is expected to evolve in this direction.

On a timeline of 5–7 years the monitoring system transformed and integrated into the health management system of an autonomous vehicle will follow same milestones of security provisioning in the maned aviation [6]. In aviation this process is far from over [7], but this experience influences our planning in preparing software for autonomous vehicles.

IMPLEMENTATION OF THE MONITORING SYSTEM

The implementation is based on a symmetric deployment of the software on all nodes of the cluster (servers) independent of the intended usage or characteristics of the server or application running on it. There should be however a dedicated node that collects and stores all data from the cluster, including the sensor data. Naturally the regular access to the monitoring data will provided from this node, but because of the symmetry all other nodes collect and store same monitoring data. In case of network failure and node separation the monitoring data may be obtained from any surviving node. It will contain information about the working of the cluster before the separation and each server will have information about its own state after the separation and it can be used to analyze the crash.

The analysis of cluster events is aided by time marks recorded with the data. The cluster is fully time-synchronized with the PTP (IEEE 1588 [8]) protocol having resolution of about 1 micro-second. In case of network separation the clock drift is not considered critical, since in this emergency mode the cluster should work only few (tens) of minutes before the communication is restored.

The information is collected by daemons that are started when OS is booted on the cluster nodes. The logical interaction scheme is shown in Fig. 1 and each cluster node implementing the same scheme.

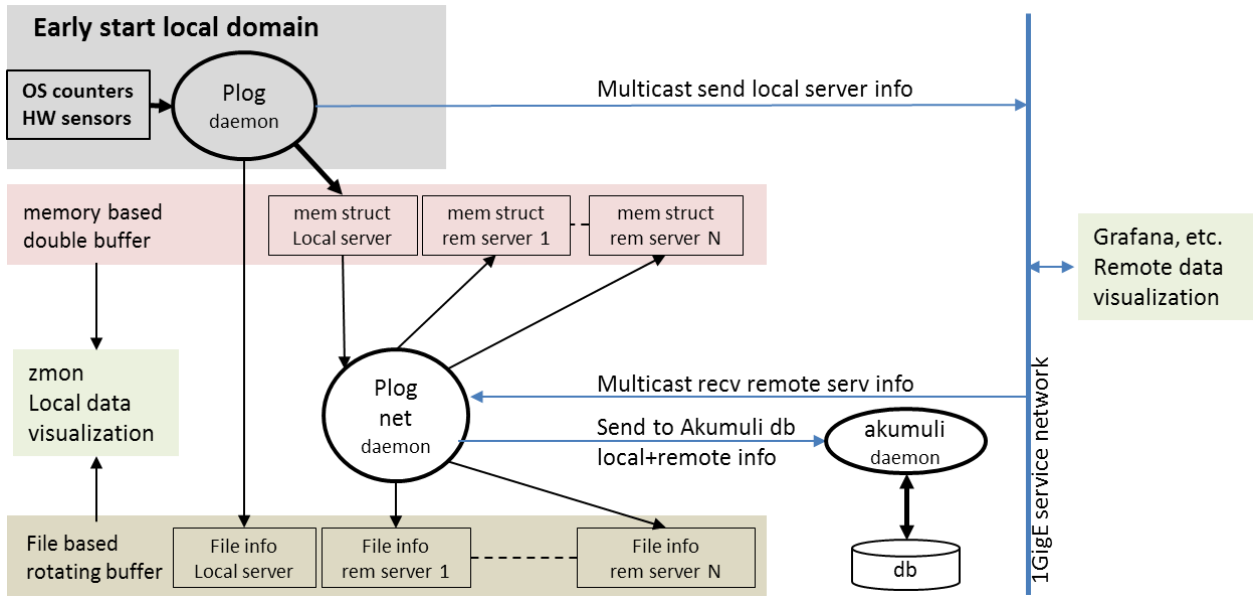

Fig. 1. The architecture of the monitoring system in autonomous vehicle for each node

On each node we start the OS Linux, Plog daemon, Plog-net daemon and Akumuli daemon. The basic Plog daemon starts first when Linux boots, initializes memory structures, collects the sensor and OS data, writes them to into the memory and an asci file. Plog is refreshing this data every 2 seconds (tunable) and sends this data in a multicast packet if network is available. After Plog – the Plog-net daemon starts, it collects the data packets sent by Plog daemon running on other servers on the network and adds this data to the structure in memory and writes an asci file for each remote server. The dump to file is important, as it increases the chances that post mortem analysis may be done after a crash; to limit the size of the asci file it contains data of last 20 minutes of running (size can be adjusted).

The functional splitting between Plog and Plog-net is determined by the requirement of monitoring without the network. Local monitoring collects data from the server and sensors that monitor external condition of each server. These sensors are attached to each server and network router to monitor the power and cooling conditions. Typically, each server collects data from its own sensor. The sensors from the network devices are attached to (one or several) server(s) that process that data in addition to their own monitoring. If network is separated it is possible to analyze the monitoring data from network devices on these servers.

Users watch each node (and all sensors attached to it) with command zmon that displays data on a textual console. A column for each server is displayed. The zmon

command is modeled after the well-known Linux utility nmon [4], build with ncursors library [9]. Command zmon with –R flag will show historical data stored in asci files and displays them circularly with tunable speed.

The Plog-net daemon implements full functionality by collecting packets sent by Plog from other nodes. In this way each node has the full picture of the working cluster.

In addition to the asci files where last 20 minutes of data is stored – all data is sent to the local time-series database Akumuli [10]. It maintains a larger data collection of about 1 month worth data (tunable) that results in about 4 GB of storage size. Akumuli can supply this data in format json to the Grafana [11] plotting package. Each cluster node has its own akumuli agent and grafana-server daemon. If graphics head is available the monitoring data can be analyzed with graphics tools where trends are clearly visible.

This implementation is fully symmetrical and does not need a specialization or per server tuning. When the test fleet of autonomous vehicles grows it will save time and resources in setting up the system and eliminates configuration mistakes.

The usage of the Time Series Data Base (TSDB) Akumuli for monitoring in Data-centers has been proposed earlier (see for example [12]). Akumuli – is developed by Evgeny Lazin [13] and its distinctive feature is speed and compactness.

The known Akumuli limitations, such as the strict locality of application (not supporting distributed data collection) and chronological data entry order do not play a role in our setup, since the Plog-net daemon will utilize local Akumuli agent with strict data entry with the local time mark. In addition, all servers support and run time synchronization protocol, therefore a discrepancy between the “local” and “remote” time mark is a symptom of dysfunction of the system and a reason to stop the vehicle. On the other hand the speed and compactness were decisive criteria when choosing this TSDB for autonomous vehicle monitoring data.

All accesses to the TSDB Akumuli write node id and other static information that allows performing a search and selection of data related to a specific node. In this way we can analyze the work of each and single node as well as cluster as a whole starting from the information stored on each node in an autonomous vehicle cluster.

THE METRICS SELECTION

Generally all sensor and OS data are stored in the TSDB and therefore available for the retrieval and graphical analysis with Grafana. For the zmon tool that has to fit the information on a single console display we are limited to a generic 224x256 (columns x rows) letters and numbers that has to be structured in a most intuitive way. For the first implementation in zmon we have chosen to present all of the environmental data for each server, i.e. the power, temperatures (in/out) and flow rate of the coolant, air temperatures measured inside the server enclosure and dew point temperatures. We also present integral characteristics of the processing, such as the total idle, user and system time, total network and total storage performance, GPU power draw and occupancy. The GPU data should give a quick assessment if the application (using GPU for neural network computation) is started and running on the system.

This selection is driven by the need to quickly assess the fitness of the system for the field run and allows analyzing failures. Sensor data is connected to the safety system that will protect servers against any adverse conditions. However, it may not be immediately obvious what has triggered the alarm and/or the safety shutdown, especially if one or more systems are down and only few consoles are operating. The present selection of signals for the display should help in this analysis.

A significant step from monitoring to a health management system has been done in collecting the per-process data. This is not displayed in the zmon console view, but stored in the TSDB and can be retrieved from there in a separate analysis program which is planned for later. The per-process monitoring will narrow down cases when the processing slows down or changes its characteristics from known values which may be a signature of an imminent failure. We are tuning this feature on the Zhores cluster at Skoltech [15] where we also do analysis for the failure signatures in user programs processing data.

IMPLEMENTATION RESULTS

The monitoring activity loads the network at about 4 kB/s (or 13 MB/hour). There is compression in the Akumuli TSDB therefore the database retains data of about one month worth. After that period the new data replaces the oldest data and the monitoring does not need manual maintenance related to the uncontrolled use of storage.

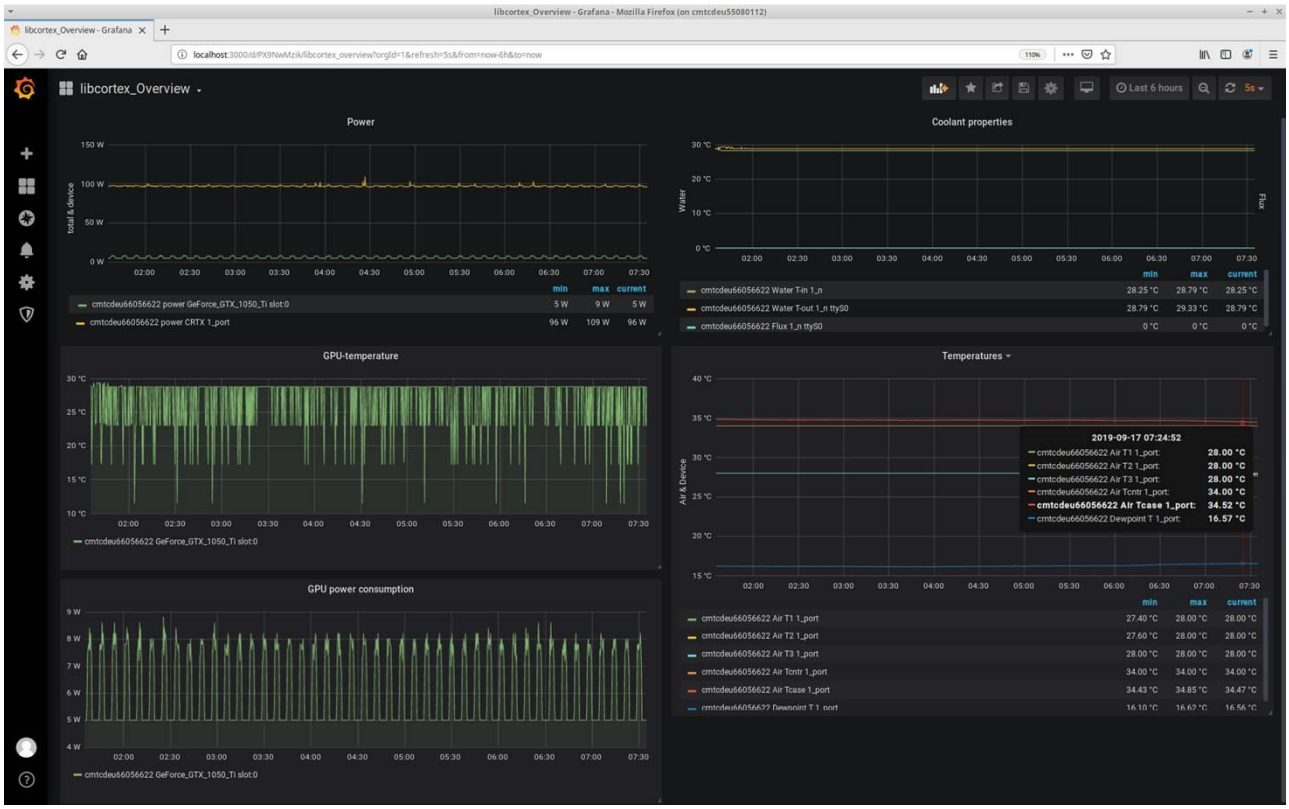

Fig. 2. Typical Grafana view of the monitoring results in a web brouser

The graphical analysis Grafana shows a typical representation, see Fig. 2. That figure illustrates the upper Dashboard for one server, where we show (left to right and up to down) the overall power consumption, temperature of the coolant, temperature of the GPU, a series of air temperatures and the GPU power draw. Note that the graphical representation is not finalized yet and will change as more experience is collected using the system. The Grafana plotting package is flexible to make many different representations of the same data, therefore it is more to the users of the system to define the proper views. The actual monitoring implementation does not depend on the Grafana graphics views. This is not the case for the zmon tool, which must be reprogrammed if a different view on the system is required.

Fig. 3 shows a typical console display from the zmon tool. When analyzing the system with the zmon program it is possible to switch on and off the information panels represented with the header in inverse video. It is also possible to select and unselect servers to view as part of the cluster configuration. In historic view this may serve as a way to focus on certain configurations.

Fig. 3. Typical zmon (-R) display on a console terminal. Each column shows one node in cluster

CONCLUSION

We have discussed the requirements and the implementation methodology of the autonomous vehicle cluster monitoring. These requirements and the implementation are distinctly different from the monitoring in the Datacenters, although similar devices (servers, routers) are used in these clusters. The project has been implemented for autonomous vehicle testing fleet.

Further development will encompass the monitoring of the monitoring system self, as well as the information from the applications that govern the data processing in the autonomous vehicle.

REFERENCES / СПИСОК ЛИТЕРАТУРЫ

1. SAE J3016 “Levels of Driving Automation”. URL: https://www.sae. org/news/2019/01/sae-updates-j3016-automated-driving-graphic (accessed: 12/05/2019)

2. Nagios. URL: https://www.nagios.org (accessed: 12/05/2019)

3. Zabbix. URL: https://www.zabbix.com (accessed: 12/05/2019)

4. Most Comprehensive List of Linux Monitoring Tools For SysAdmin. URL: https://www.ubuntupit.com/most-comprehensive-list-of-linux-monitoring-tools-for-sysadmin (accessed: 12/05/2019)

5. Top FREE Server Monitoring Tools. URL: https://www.dnsstuff.com/free-server-monitoring-tools (accessed: 12/05/2019)

6. In-Time Aviation Safety Management: Challenges and Research for an Evolving Aviation System (2018). ISBN 978-0-309-46880-0. DOI 10.17226/24962

7. Aircracft Health Monitoring – FAA Perspective, presented to IATA Conference by Tim Shaver, 13 November 2017. URL: https://www.iata.org/whatwedo/workgroups/Documents/Paperless_Conference_2017/Day1/1130-1200_AircraftHealthMonitoring_FAA.pdf (accessed: 12/05/2019)

8. NIST Intelligent Systems Division. URL: https://www.nist.gov /el/intelligent-systems-division-73500/ieee-1588 (accessed: 12/05/2019)

9. NCURSES Programming HOWTO. URL: https://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/intro.html (accessed: 12/05/2019).

10. Akumuli. URL: https://akumuli.org/ (дата обращения: 12/05/2019)

11. The open platform for beautiful analytics and monitoring. URL: https://grafana.com (accessed: 12/05/2019)

12. *Живчикова Н.С., Шевчук Ю.В.* Подсистема архивации данных системы мониторинга Botikmon3 // Научный сервис в сети Интернет: труды XX Всероссийской научной конференции (17–22 сентября 2018 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2018. С. 223–229.
URL: http://keldysh.ru/abrau/2018/theses/26.pdf doi:10.20948/abrau-2018-26

13. *Лазин* E. Numeric B+tree reference. URL: https://akumuli.org/akumuli/2017/ 04/29/nbplustree/ (accessed: 20/11/2019)

14. *Панарин О., Захаров И.* Особенности мониторинга мобильных систем обработки информации// Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23–28 сентября 2019 г., г. Новороссийск). М.: ИПМ им. М.В. Келдыша, 2019. С. 551–560. URL: http://keldysh.ru/abrau/2019/ theses/81.pdf (accessed 16/11/2019). doi: 10.20948/abrau-2019-81

15. *Zacharov I., Arslanov R., Gunin M., Stefonishin D., Bykov A., Pavlov S., Panarin O., Maliutin A., Rykovanov S., Fedorov M.,* “Zhores” – Petaflops supercomputer

for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology// Open Engineering. Published 2019-10-26, V. 9. Issue 1. doi: https://doi.org/10.1515/eng-2019-0059

СВЕДЕНИЯ ОБ АВТОРАХ

ПАНАРИН Олег Анатольевич – менеджер по информационным сервисам и обработке данных Сколковского института науки и технологий, специалист в области высокопроизводительных компьютерных систем и систем мониторинга.

Oleg PANARIN – Manager of Data and Information Services at Skolkovo institute for Science and Technology specializing in High Performance Computing and System Monitoring.

e-mail: o.panarin@skoltech.ru

ЗАХАРОВ Игорь Евгеньевич – старший научный сотрудник Сколковского института науки и технологий, специалист в области высокопроизводительных компьютерных систем и системного программирования.

Igor ZACHAROV – Senior researcher at Skolkovo institute for Science and Technology specializing in High Performance Computing and System Programming.

e-mail: i.zacharov@skoltech.ru

Материал поступил в редакцию 15 ноября 2019 года