UDC 004.9

A. Mokryi, I. Baklan

METHODS AND SOFTWARE FOR SOLAR PLANT CLUSTER MANAGEMENT

Abstract: Nowadays, solar panel production technologies are developing rapidly, investments in solar energy are growing, so users are interested in increasing energy production for faster return on investment. To increase the efficiency of solar panels, special rotating mechanisms are used to rotate the panel perpendicularly to the sun rays. This allows users to significantly increase the amount of energy produced by the panel over time. This article considers ways to increase the efficiency of solar panels’ energy generation by using a minimum number of additional sensors.

The article describes the solar power plant monitoring system which uses machine learning to resolve sun tracking problem. The advantages and disadvantages of a prototype are analyzed. The concept of microservices is described and the benefits of using it in the developed system are given. Different approaches to increase speed, quality and reliability of such systems are investigated on the example of a prototype. Efficiency of using proposed approaches has been tested on a solar simulator.

Keywords: solar power plant, solar power plant monitoring, reinforcement machine learning, q-learning, parallel programming, threads, microservices.

Problem description

The modern world is impossible to imagine without the use of electricity. Its importance is difficult to overestimate, as it is necessary for the operation of all available infrastructure, from farms to factories. City dwellers have become particularly dependent on electricity: without it, there will be no centralized water supply, heating and lighting.

There are significant benefits of using solar energy to generate electricity. Firstly, solar power plants do not pollute the environment, and the sun is a source of energy for billions of years. Secondly, solar panels can be placed anywhere in the open space, and this process does not require large-scale work, such as creation of reservoirs during the construction of hydropower plants. Thirdly, solar power plants are the easiest to operate among all types of power plants. The above arguments determine the growing popularity of this type of power plants, so users are interested in increasing energy production. To increase the efficiency of solar panels, special rotating mechanisms are used to rotate the panel perpendicularly to the sun rays. This allows to significantly increase the amount of energy produced by the panel over time.

Existing solutions require the use of additional sensors to implement sun tracking. They also impose severe restrictions on the equipment on which they are installed. There are no cross-platform solutions among the existing ones, they work only on equipment from a specific manufacturer. Designed solution considers the use of machine learning to resolve sun tracking problem with minimum number of sensors. This article describes various methods to increase speed, quality and reliability of solar plant monitoring systems.

Prototype overview

The prototype is a solar power plant monitoring system, which was developed by me during the bachelor's diploma project. This system consists of the following components (Fig. 1):

1. Central server.
2. Control system.
3. Web application.

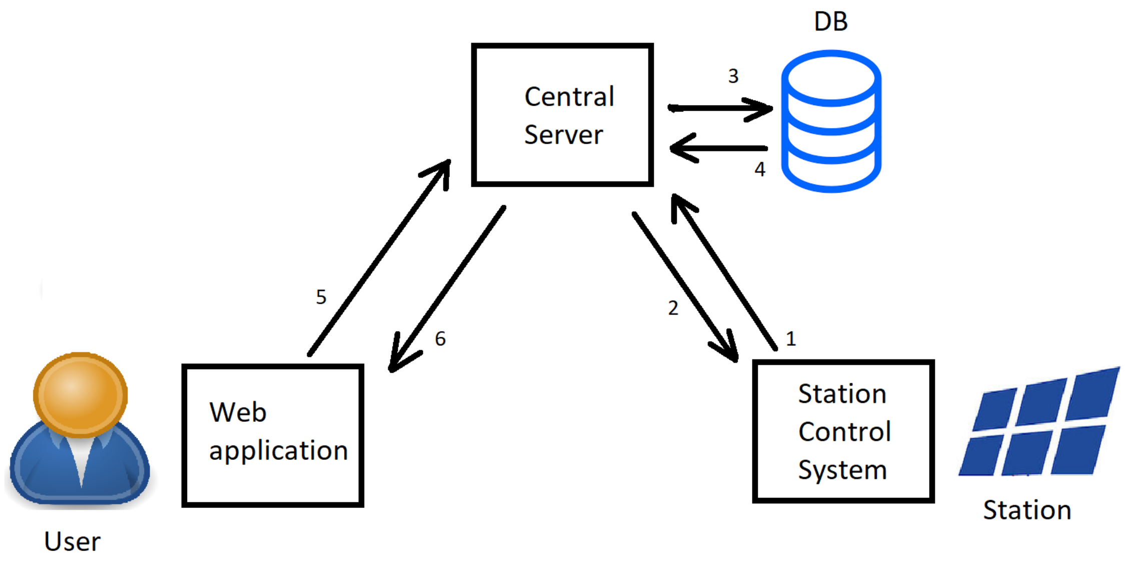

Figure 1. Interaction of the prototype’s modules

The central server has access to a database that stores information about system users, their panels, generated and consumed electricity. The figure shows that all requests from users and stations pass through a central server, which is the entry point for the system.

The control system level is a program that is installed directly on the power plant controller. This program reads the data for each panel at regular intervals, makes requests to the central server, transmits information about the status of the station and receives control commands such as "turn panel X 1 degree up". In this case, the request is transmitted by 1-3-4-2.

The web application allows the user to view current information about the operation of their station on the company's website, as well as manage the connection of individual units. To access their power plant, the user must log in, then the central server finds the station ID in the database for this user, then transmits control commands to the appropriate station.

Algorithm for finding the optimal position of the solar panel

To find the optimal position of the panel, a reinforcement learning algorithm is used [1]. Each panel at any given time has certain state, which is determined by the coordinates of azimuth (clockwise rotation from the North) and altitude. For each state there are 4 options (up, down, left, right). The action rotates panel 1 degree in the appropriate direction.

Each state has a utility value for each action. Assume the state X has values for actions up, down, left, right, respectively, a, b, c, d. The prototype uses the following algorithm:

1. Set the initial values of the utility of actions for current state. Since the Sun moves clockwise during the day, we set the utility of the action "turn right" to 0.1, and set the utility of other actions to 0. For the state "1 degree right" we set the utility of actions "left" and "right" to 0.
2. Calculate the current power P_X.
3. Choose the recommended action:

$$i := indexOf(max(a, b, c, d)). \tag{1}$$

4. Perform an action, this puts the panel in the state "$X + 1$".
5. Calculate the power in a result state P_{X+1}.
6. Calculate of the benefit of the action i:

$$diff := P_{X+1} - P_X. \tag{2}$$

7. Update utility values for action i.

Assume that the utility of action i in state X has increased by the value of $diff$:

$$X[\,i\,] := X[\,i\,] \; + \; diff \tag{3}$$

Since the Sun moves straight, we assume that, in case of "success" of action i, the panel should continue moving in the same direction. For this, we set

$$(X + 1)[\,i\,] := (X + 1)[\,i\,] \; + \; diff \,/\, 10. \tag{4}$$

So, if the move has increased the power of the panel, it will receive an incentive to move forward. If the move reduces the power of the panel, it will receive an incentive to go back and choose another path.

8. Check the established position on the criterion of optimality. If the utility for all actions is negative, then finish the algorithm, otherwise go to paragraph 2.
9. The end.

The system repeats this algorithm every 10 minutes.

Prototype features

1. User-independent search for the optimal position for each panel. The search works faster over time because of saving the most useful actions. Thus the system “learns”.
2. Provides the ability to remotely observe the operation of user’s station, as well as control the connection of panels and access to the electricity grid.

3. It is possible to accumulate a certain amount of energy and sell it at the time of greatest demand and, accordingly, its prices.

4. Allows user to save on equipment, as only the position and power sensors are used to find the optimal position of a panel. The system does not need to know the position of the Sun at any time for current latitude, so you do not need to install a GPS module. It is also not necessary to install the panels on a perfectly leveled surface and direct them strictly to the south. This saves on surface preparation for station installation without loss of productivity.

Prototype limitations

Machine learning involves working with the utility of certain actions in certain states. In case of a solar power plant monitoring system, the current state of a solar panel is determined by the coordinates of its position. According to the algorithm, at each step you need to find out the best action for the current state, and then update the utility of actions for this state. Thus, the station software makes a large number of queries to the database. Because all such requests pass through a central server, connecting many solar panels will cause it to overload.

Another disadvantage of the prototype is that the search for the optimal position is performed sequentially for each panel. In reality, updating the position of a panel can take tens of seconds, so the algorithm may not have time to update their positions in a given time for increased number of panels. This article considers the use of parallel programming methods to speed up the process of finding the optimal position of solar panels by several times.

Parallel calculation of the optimal position of solar panels

Key factors in efficiency of the solar power plant management system include the speed of solar panels’ response to changes of the Sun’s position. Since the maximum generation of electricity is reached when the sun rays fall perpendicularly to the plane of the panels, they must be adjusted periodically during the day. To do this, the station software searches for the optimal position for each panel every 10 minutes. During prototype testing, it was determined that it takes 12.66 seconds on the first day and 8.99 seconds on the second day to adjust the position of one panel. For real solar panels, this time will be longer, as it is needed to turn the panel and detect a power change. Based on the situation, it was decided to apply the algorithm for finding the optimal position in parallel for each panel.

There are 2 main approaches to implement parallel computing on a local computer [2]:

1. Multiprocessing – used for computations with allocated memory. Each process is independent, duplicates much of the status information, and has a separate address space. Each process performs its task and interact with other processes through systemic inter-process communication mechanisms.

2. Multithreading – used for shared memory computing. Intra-process threads share process status information, and have direct access to shared memory and other resources.

Switching context between process threads is faster than switching context between processes.

The system under construction solves the same problem in parallel, so it was decided to use a thread model. The algorithm of parallel calculation based on threads has the following form:

```
List<Thread> threads = new ArrayList<>();
for (PanelVO panel : panels) {
   Thread thread = new Thread(() -> doTaskForPanel(panel));
   thread.start();
   threads.add(thread);
   if (threads.size() >= maxSize) {
      waitForThreads(threads);
   }
}

…

private void waitForThreads(List<Thread> threads) {
   for (Thread thread : threads) {
      try {
         thread.join();
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
   }
   threads.clear();
}
```

Computer kernel resources are allocated to run the thread. Accordingly, the maximum number of threads that can run in parallel is equal to the number of cores in the system. Thus, the use of this approach guarantees the acceleration of the complex task by N times, where N is the number of processor cores.

Database access optimization

Critical system vulnerabilities were identified during prototype testing. When adjusting the position of the solar panels, a large number of queries to the database are performed, and all of them pass through the central server. Disabling it will not only make it impossible for the user to view the information, but will also block the process of tracking the panels by the Sun. Thus, even performing scheduled technical work on the server will prevent users from making profit from their power plants.

To solve this issue, it is proposed to implement direct access of station software to the database. Using this approach allows stations to work in case of problems with access to the server. It also significantly reduces the load on the server side of the system and speeds up the processing of requests. The scheme of the prototype database is shown in Fig. 2.

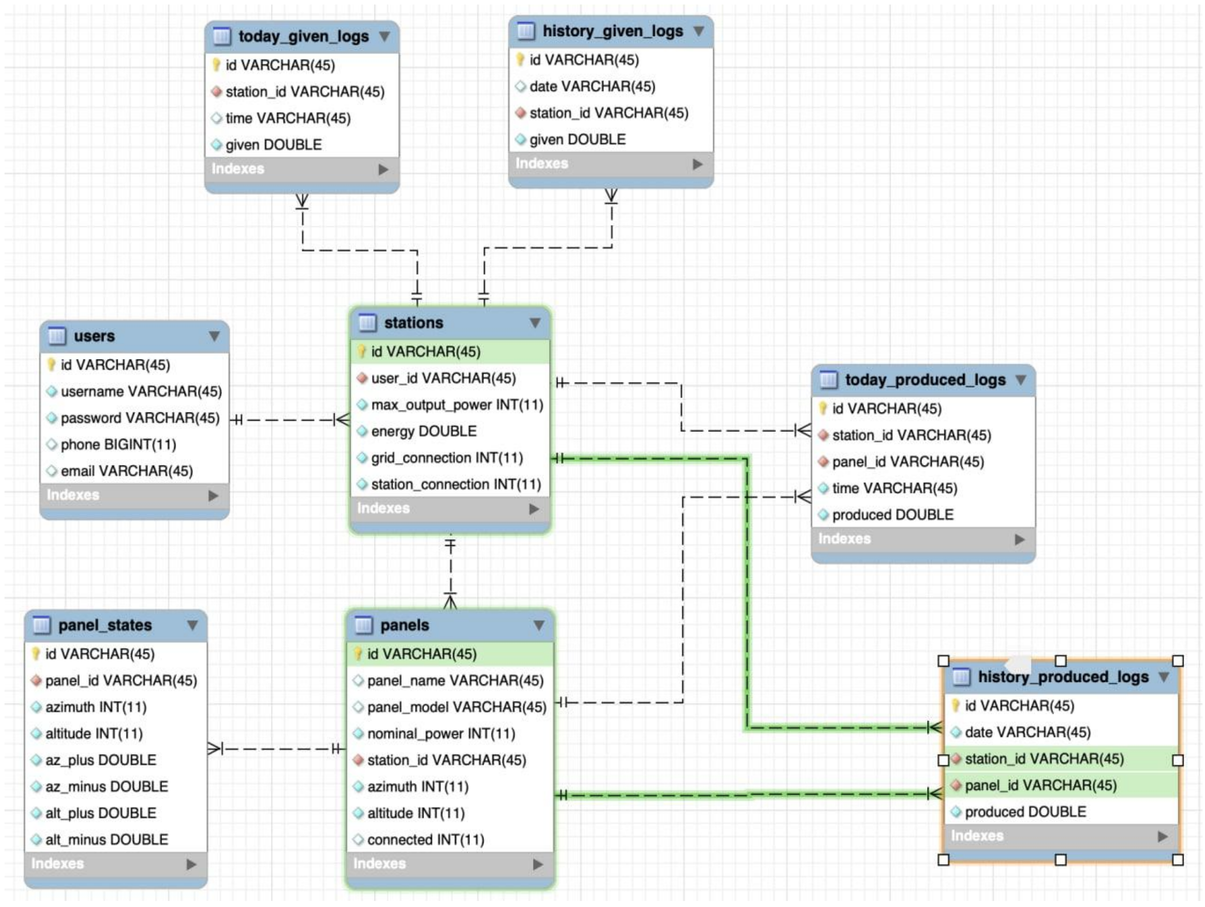

Figure 2. Prototype database scheme

In the diagram, we see the table panel_states, which saves the states of solar panels with values of the utility of possible actions az_plus, az_minus, alt_plus, alt_minus. According to the station software algorithm, changes in status information occur ten times more often than updating statistics and much more often than receiving user commands. Also, panel status information is no longer used anywhere. Therefore, it is proposed to extract the panel_states table in a separate database, which can be accessed by station services. Thus, the process of solar panels adjustment won’t depend on the system’s server part. On the one hand, implementation of the proposed approach will reduce the load on the main base, and on the other – power plant controllers will not have access to redundant information. It will also allow us to choose the optimal database management system for each database depending on the load.

Scaling the system’s server part

An important feature of software systems is scalability – the ability to easily adapt to increased load [3]. A system is called scalable if it is able to increase productivity in proportion to additional resources. Scalability is also meant as the possibility of attracting additional resources without structural changes in the system architecture. Scalability is a desirable feature of the system being created, as it allows you to react quickly to increased number of customers.

Scalability can be assessed by the ratio of the increase in system’s productivity to the increase in borrowed resources. The closer this value is to one, the better. In a system with poor scalability, adding resources leads to only a slight increase in productivity, and from a certain “threshold” point, adding resources does not give any useful effect.

There are two types of scalability: vertical and horizontal. Vertical scaling – increasing the performance of each component of the system in order to increase overall performance. Scalability in this context means the ability to replace components in an existing computing system with more powerful and faster ones as requirements and technology evolve. This is the easiest way to scale, because it does not require any changes in applications running on such systems.

Horizontal scaling – splitting the system into smaller structural components and spreading them over individual physical machines (or their groups), and (or) increasing the number of servers that perform the same function in parallel. Scalability in this context means the ability to add new nodes to the system, servers to increase overall performance. This method of scaling may require changes to the program code so that programs can take full advantage of the increased resources.

To ensure the horizontal scalability of the solar power plant cluster management system, this paper uses a microservice architecture, which involves splitting the application into many relatively simple functional components (microservices) that interact with each other. This approach has the following advantages [4]:

1. Services start faster, which speeds up system deployment.
2. Each service is deployed independently of the others. Therefore at change of one of them it is not necessary to restart all system, other microservices may continue to operate.
3. Each service can be scaled separately. It is possible to scale exactly the functionality which is a bottleneck at this stage.
4. A bug in one microservice does not affect the work of others. Only the functionality performed by this service will break.
5. Each service performs a specific task and is not engaged in the work of other services. They are easier to understand, support and test. Any service can be changed independently of others.

6. Independence from technology – when developing a new service, you can choose a new stack of technologies. Also, any service can be rewritten from scratch within a reasonable time and budget without the need to rebuild the entire system.
7. Separate services are easier to reconfigure to perform various tasks.

After analyzing the prototype’s functionality, it was decided to divide the central server into the following microservices:

- authentication service – to perform registration and authentication of users;
- statistics service – to collect data on electricity generation and sales for each station;
- management service – to transmit user commands to the desired station;
- request processing service – the system’s entry point, meant to redirect user’s requests to the desired service.

Thus, the architecture of the created system took the following form (Fig. 3):

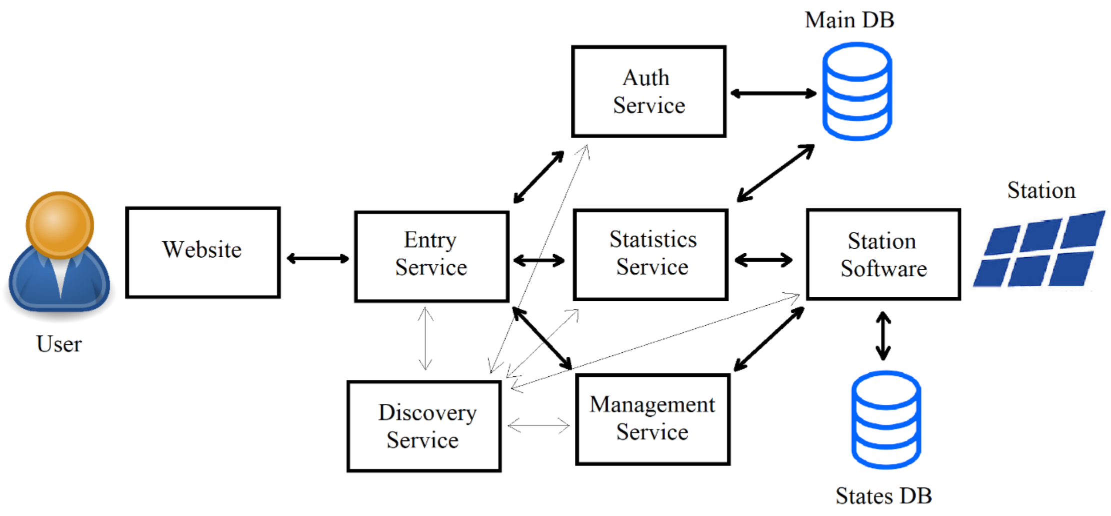

Figure 3. Architecture of the developed system

Station software is installed on the power plant controller and monitors its operation. This program periodically adjusts the position of the solar panels using the panel states database, reads the data about the station and panels and transmits this information to Statistics Service. It also executes control commands received from Management Service.

Entry Service is the entry point of the system. It redirects user’s requests to desired service.

Authentication Service performs registration and authorization of users in the system.

Statistics Service is responsible for data on electricity generation and sales for each station. Authentication Service and Statistics Service have access to the main database.

Management Service is designed to transmit user’s control commands to appropriate station.

Discovery Service registers all backend and power plant services. This allows registered services to address each other by service name without using IP addresses. In the developed system, this allows the management service to transmit user’s requests to the desired station without knowing their real IP addresses.

Website interacts with the user: it allows him to view current information about the work of his stations on the company’s website, as well as manage the connection of individual units. The site sends requests to Entry Service, from which they are redirected to required service.

Experiments

To test the system, there was created a service that stores data about actual position of the Sun on June 1-4, 2022 in Kyiv. The information is taken from the site Stellarium-web [5]. The created service contains an array of data in the format “key-value”, where the key is the date and time in the format “yyyy-mm-ddThh:mm:ss”, i.e. “2022-06-01T10:30:00”. Thus we obtain the coordinates of the Sun’s position at the given time. Next, knowing the position of Sun and the position of the panel, we determine the coefficient:

$$k = \cos\varphi, \tag{5}$$

where φ is the angle between the sun rays and perpendicular to the solar panel (Fig. 4), which is determined by mathematical formulas. Then the panel’s current power equals:

$$P_i = P_{nom} * k_i, \tag{6}$$

where P_{nom} is the panel’s nominal power, k_i is current coefficient.

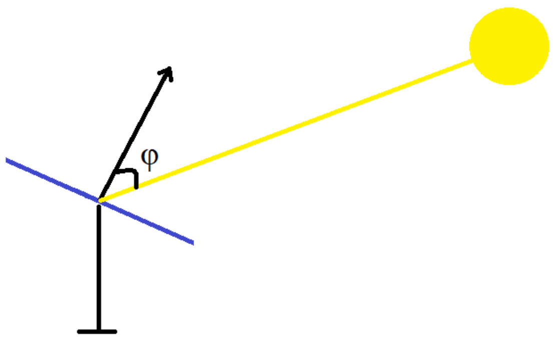

Figure 4. Angle between the sun ray and the perpendicular

To evaluate the efficiency of proposed solutions, we use the solar simulator. The daily average time required to move 10 solar panels in the optimal position is investigated. A study has been conducted for the following situations (Fig. 5):

1. Prototype – before any improvements.
2. Prototype after optimizing stations’ access to the database (intermediate result).
3. Prototype after optimizing stations’ access to the database and implementation of the algorithm for parallel calculation of the panels’ optimal position (end result).

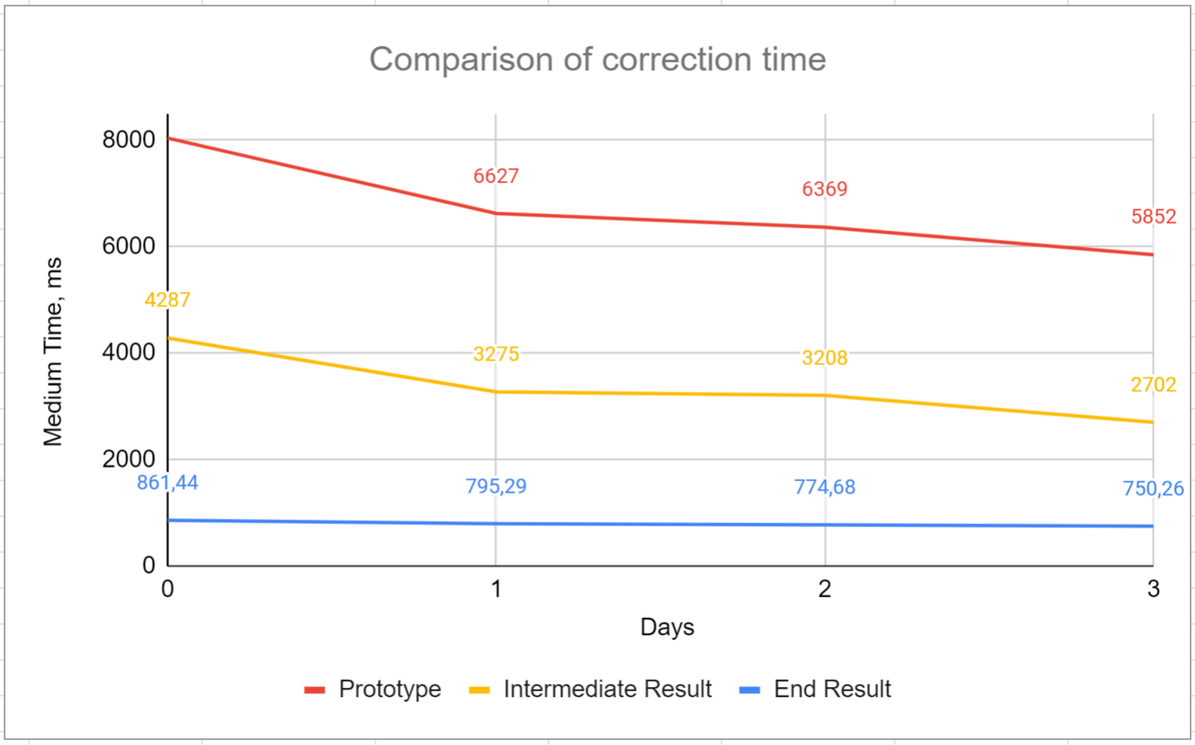

Figure 5. Comparison of correction time of 10 panels, ms

The figure shows that the implementation of direct access of stations to the database reduces the required time by an average of 49.9%. This is due to the fact that when correcting the position of the solar panel, the request travels twice shorter path between nodes because of optimizing the system architecture. The use of parallel computations reduces current time by 4.24 times. This result meets the expectations, as test computer has 4 cores.

Summary

The solar power plants cluster management system has been improved. Experimental results demonstrate an increase in speed and reliability compared to the prototype. The developed solution has the following advantages:

1. Autonomy of power plant software when adjusting the position of solar panels. The search for the optimal position of the panel is independent of the server side, which increases the stability of the system. It also reduces the search time by 2 times.

2. Parallel search for the optimal position through the use of multithreading. As a result, the search is performed simultaneously for N panels, where N is the number of processor cores of the station controller. Accordingly, the overall process of adjusting all the panels of the station is performed N times faster.

3. Due to the use of microservice architecture, the developed system can be scaled horizontally. You can add the required number of instances for each service.

The practical value of remote control systems is hard to overestimate in combat. They allow you to continue managing objects located in hard-to-reach areas. The development of this system took place in peacetime, but the concept of parallel calculation of the solar panels’ optimal position can be used to implement simultaneous aiming at coordinates for artillery batteries or air defense systems.

REFERENCES

1. Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf [last accessed: 15.05.2022]
2. Processes end threads. URL: https://tehnar.net.ua/protsesi-i-potoki/ [last accessed: 15.05.2022]
3. Scalability as software requirement. URL: https://uk.itpedia.nl/2021/07/20/schaalbaarheid-als-software-requirement-betekenis-en-definitie/ [last accessed: 15.05.2022].
4. Microservice architecture. URL: https://medium.com/@IvanZmerzlyi/microservices-architecture-461687045b3d [last accessed: 15.05.2022].
5. Sky map. URL: https://stellarium-web.org/ [last accessed: 15.05.2022].