ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

МЕТОД ПОБУДОВИ ВІРТУАЛЬНОГО КЛАСТЕРА НА ГРАНИЧНОМУ ШАРІ ІНТЕРНЕТУ РЕЧЕЙ

Доповідач:
студент групи СПм-22-4
Шеховцов О. В.
Керівник : зав. кафедри Коваленко А.А.

2024

2

АКТУАЛЬНІСТЬ

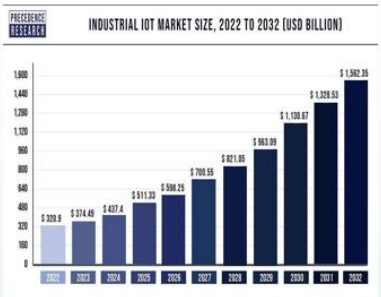


Рисунок 1 – Зростання ринку Інтернету речей

Рисунок 2 – Pico Cluster на основі п'яти Raspberry Pi 4

Рисунок 3 – SOPINE Clusterboard плата з 7 модулями SOPINE A64

,,

3

МЕТА ТА ЗАВДАННЯ РОБОТИ

Мета дослідження - підвищення ефективності використання одноплатних комп'ютерів з обмеженими можливостями, розташованих на граничному шарі Інтернету речей.

Основні задачі дослідження:

1) обґрунтувати вибір методів і алгоритмів побудови віртуального кластеру;
2) сформувати архітектуру віртуального кластера комп'ютерів з обмеженими ресурсами;
3) побудувати та дослідити ефективність віртуального розподіленого кластера граничного шару одноплатних комп'ютерів Інтернету речей.

4

РОЗРОБКА МЕТОДУ ПОБУДОВИ ВІРТУАЛЬНОГО КЛАСТЕРА

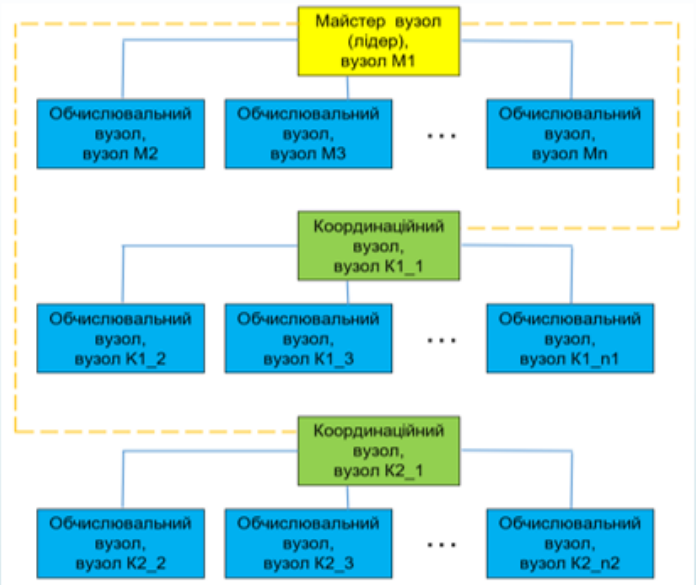


Рисунок 1 – Архітектура кластера з координаційними вузлами

,

5

ВИБІР АРХІТЕКТУРИ ДЛЯ МЕТОДУ ПОБУДОВИ ВІРТУАЛЬНОГО КЛАСТЕРА КОМП'ЮТЕРІВ З ОБМЕЖЕНИМИ РЕСУРСАМИ

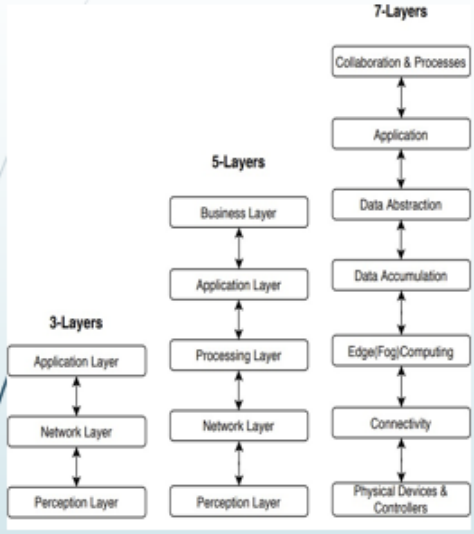


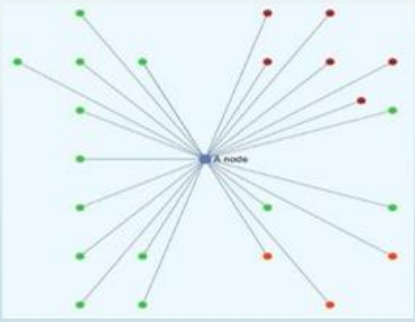
Рисунок 1 – Рівні архітектури Інтернету речей

Рисунок 2 – Архітектура віртуального гетерогенного кластера

6

ШАР ФОРМУВАННЯ ТА УПРАВЛІННЯ КЛАСТЕРОМ

Рисунок 1 – Основні компоненти шару



Вузол А – координаційний вузол;

помаранчевий колір – вузли зберігання даних;

зелений і темно червоний колір – віртуальні групи вузлів

Рисунок 2 – Візуалізація сформованого віртуального кластера

КООРДИНАЦІЙНИЙ ШАР КЛАСТЕРА

7

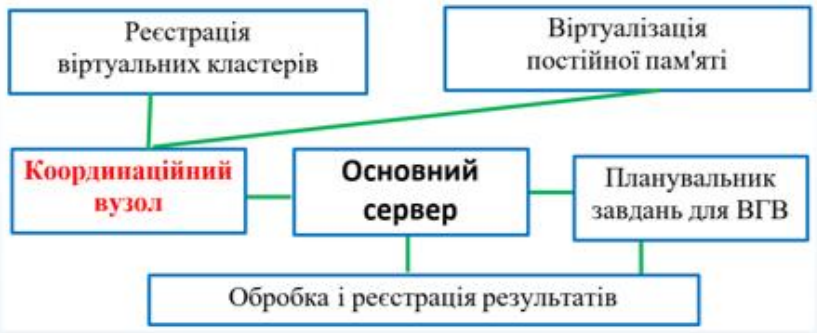


Рисунок 1 – Основні компоненти координаційного шару кластеру

Рисунок 2 – Приклад побудови віртуалізації пам'яті

МЕТОД ПОБУДОВИ ВІРТУАЛЬНОГО КЛАСТЕРА

8

9

РЕЗУЛЬТАТИ K-MEANS

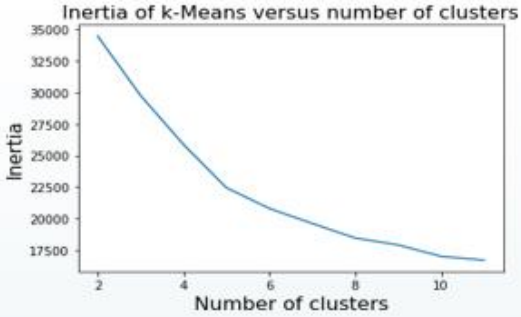


Рис. 1. Визначення кількості кластерів

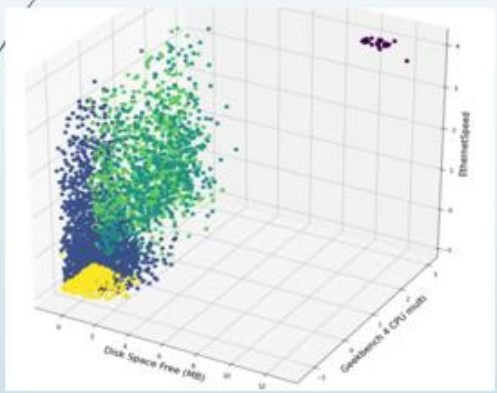


Рис. 2. Вхідний простір даних

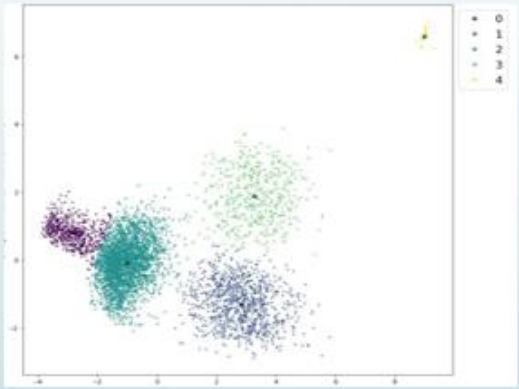
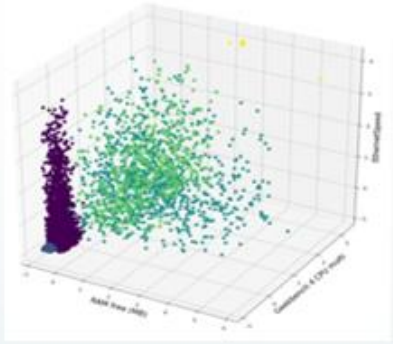
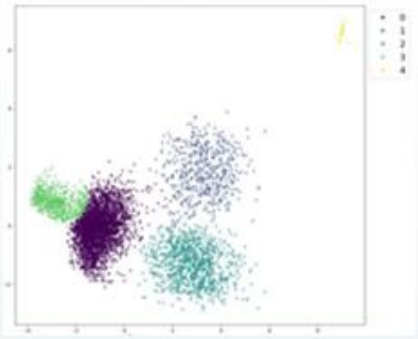


Рис. 3. Простір головних компонентів

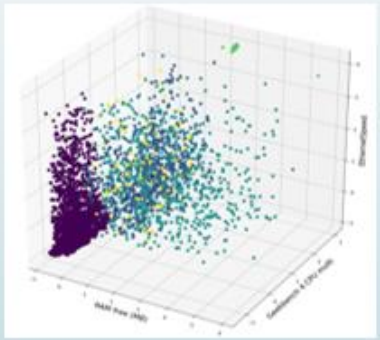
10

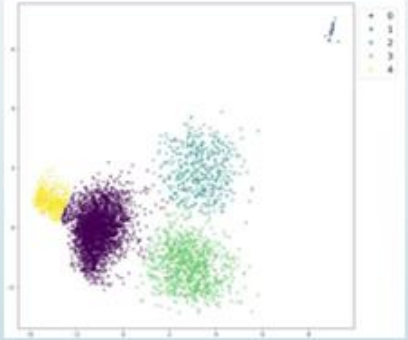
АГЛОМЕРАТИВНА КЛАСТЕРИЗАЦІЯ





СПЕКТРАЛЬНА КЛАСТЕРИЗАЦІЯ





11

ОЦІНКА РЕЗУЛЬТАТІВ МЕТОДУ

Таблиця 1 – Результати метрик у вхідному просторі

	Silhouette	Davies-Bouldin	Calinski&Harabasz
K- Means	0.404	1.093	1844.8
Agglomerative	0.413	1.085	1835.3
Spectral	0.406	1.920	1404.8

Таблиця 2 – Результати метрик в просторі головних компонент

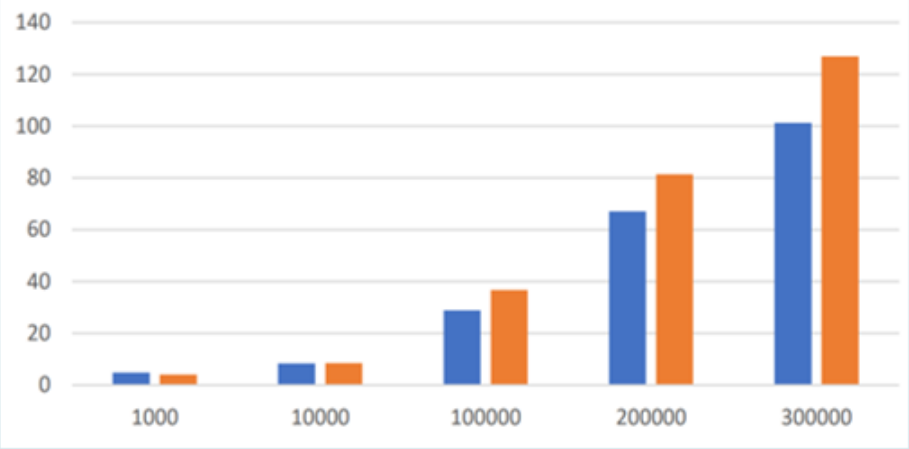
	Silhouette	Davies-Bouldin	Calinski&Harabasz
K- Means	0.618	0.556	8148.6
Agglomerative	0.604	0.574	7998.5
Spectral	0.605	0.602	7742.5

Таблиця 3 – Час виконання кластеризації

Назва методу	Час виконання (с)
K- Means	0,66
Agglomerative Clustering	1,39
Spectral Clustering	6,13

12

ПОРІВНЯННЯ МЕТОДУ ПОБУДОВИ ВІРТУАЛЬНОГО КЛАСТЕРА З АНАЛОГАМИ

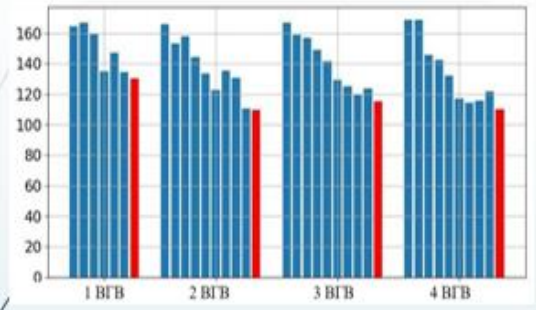


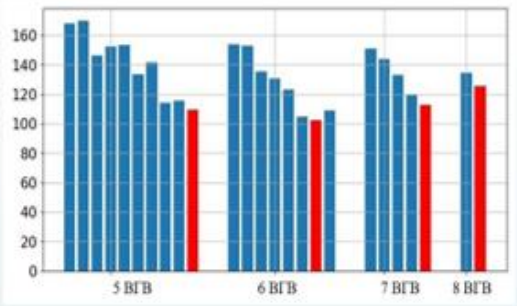
синій колір – час виконання завдань віртуальним кластером;

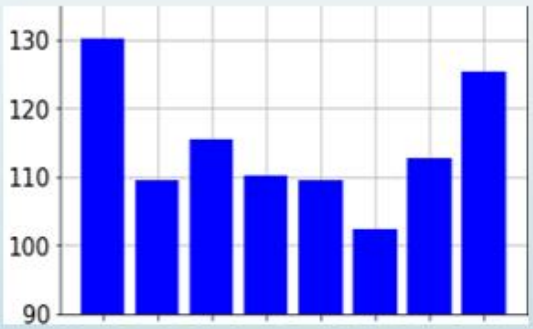
помаранчевий колір – час виконання завдань класичним кластером

13

ВИБІР КОНФІГУРАЦІЇ МЕТОДУ







14

ВИСНОВКИ

Сукупність отриманих у кваліфікаційній роботі результатів дозволило вирішити актуальне науково-технічне завдання удосконалення методу віртуалізації граничного шару одноплатних комп'ютерів Інтернету речей.

В результаті проведених досліджень отримані такі результати:

1. Проведений аналіз застосування комп'ютерів з обмеженими обчислювальними можливостями у мережах Інтернету речей.
2. Обґрунтований вибір методів і алгоритмів побудови віртуального кластеру.
3. Сформована архітектура віртуального кластера комп'ютерів з обмеженими ресурсами. У розробленій архітектурі використовуються чотири основних шари.
4. Удосконалений метод віртуалізації граничного шару одноплатних комп'ютерів Інтернету речей за рахунок побудови віртуального кластеру гетерогенних пристроїв з обмеженими обчислювальними ресурсами, що надало виграш за часом виконання завдань в середньому на 8%.

ISSN 2073-7394

Національний університет
"Полтавська політехніка імені Юрія Кондратюка"

National University
"Yuri Kondratyuk Poltava Polytechnic"

Системи управління, навігації та зв'язку

Control, navigation and communication systems

Випуск 2 (76)

Issue 2 (76)

Щоквартальне видання

Засноване у 2007 році

Quarterly

Founded in 2007

Phone: +38 (050) 302-20-71

E-mail of the editorial board: kuzhuk_nina@ukr.net

Information site: http://journals.nupp.edu.ua/sunz

Журнал індексується міжнародними наукометричними базами: Index Copernicus (ICV = 82.66), General Impact Factor, Google Scholar, Academic Resource Index, Scientific Indexing Service

Затверджений Вченою Радою Національного університету "Полтавська політехніка імені Юрія Кондратюка" (протокол від 30 квітня 2024 року № 8)

Свідоцтво про державну реєстрацію: КВ № 24464-14404 ПР від 27.03.2020 р.

Включений до "Переліку наукових фахових видань України, в яких можуть публікуватися результати дисертаційних робіт на здобуття наукових ступенів доктора наук, кандидата наук та ступенів доктора філософії" (із категорії Б – наказ МОН України від 17.03.2020 № 409 та від 09.02.2021 № 167)

Полтава • 2024

2522-2073-7394 Системи управління, навігації та зв'язку. 2024. № 2

UDC 004.9 doi: 10.26906/SUNZ.2024.2.141

I. Radchenko, O. Shekhovtsov, A. Kovalenko, O. Sytnyk

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

FORMATION OF CLUSTERS ON SINGLE-BOARD COMPUTERS IN IOT NETWORKS

Abstract. The article looks at the problem of using single-board computers for Internet technology . An analysis of current single-board computers in various countries was carried out. Single-board computers and clusters of single-board computers have found their place in the concept of edge computing, allowing optimization of hard computing by placing computing resources closer to the core. The idea of a "virtual cluster" lies in the unification and organization of disparate heterogeneous devices for the development of various complex computing tasks from the available resources of the existing infrastructure of edge computing, what is known in the area. First of all, we have secured the resources of single-board computers. Such a cluster will also allow the use of resources from the existing infrastructure in a more efficient way, for example, by activating additional services from processing and saving data.

Keywords: single-board computer, cluster, Internet of Things, edge computing.

Introduction

Currently, all over the world, Internet of Things technologies (IoT , Internet of Things) are increasingly becoming an integral part of our daily life, for example: smart home, smart city, agriculture, medicine, logistics, etc. The huge amount of data generated by Internet of Things devices requires further post-processing and analysis, including the use of machine learning. To process incoming information, "clouds" are used, where data center facilities directly analyze them and make decisions based on the results obtained. However, the number of devices connected to the Internet is constantly increasing, which leads to difficulties in processing data in the "clouds" due to an increase in incoming data, increased load on the network, increased delays in data transfer between nodes, etc.

One solution to this issue is the development of edge computing technologies, which allow us to take on part of the load and reduce the response time to an event. To organize edge computing, single-board computers are used, which have compact sizes and high energy efficiency, but low performance, which does not allow solving computationally complex problems using the resources of a single device. One solution to this problem is to create a local cluster consisting of computers with limited computing resources.

Analysis of current single board computers in various fields

Single board computer is a self-contained computer assembled on a single printed circuit board on which all the necessary components are installed to ensure its functioning: processor, RAM, input-output systems, etc.

The first truly single-board computer appeared back in 1976 and was called "MMD-1" (Mini-Micro Designer 1), but at the same time this format of computer execution became truly widespread and accessible only in 2012 with the advent of Raspberry Pi [1]. This computer is primarily aimed at the educational sphere and training in computer use, programming, etc. With a low price and sufficient performance to run a full-fledged Linux- based operating system, it has gained popularity among many enthusiasts and researchers from various fields. Thanks to the success of Raspberry Pi and other manufacturers began to offer their models of single-board computers and currently there are more than 200 models of single-board computers in the world. The most popular are various Raspberry models Pi 3 and 3b+ , RP Zero W and 2020 model Raspberry Pi 4, which has several variants with 2 GB, 4 GB and 8 GB of RAM. Among other manufacturers, it is worth noting Banana Pi M2, RockPro64, Odroid N2. Comparative characteristics of single-board computers are presented in Table 1.

Table 1 – **Comparison of characteristics of single-board computers**

	RAM (GB)	SoC	GPU	Ethernet	Storage
Raspberry Pi 3b+	1 LPDDR2	ARM Cortex A53	VideoCore IV	330 Mbit Ethernet, 2.4GHz and 5GHz IEEE 802.11 b/g/n ac wireless LAN, Bluetooth 4.2, BLE	microSD card
Raspberry Pi 4	2, 4, 8 LPDDR4	4 x Cortex-A72	VideoCore VI	1000 Mbit/s Ethernet 802.11b g/n ac WiFi 5 and Bluetooth 5.0	microSD cards
RP Zero W	0.5	1 x ARMv11 76ZJFS	VideoCore IV	802.11 b/g/n wireless LAN, BLE 4.1	microSD cards
RockPro64	4 LPDDR 4	4 x ARM Cortex-A53 2x ARM Cortex A72	Mali-T860-MP4	1000 Mbit/s Ethernet	slot for eMMC modules
Banana Pi M5	4 GB LPDDR4	Quad-Core Cortex-A55	Mali-G31 GPU	1000 Mbit/s Ethernet	microSD – card + eMMC
Odroid N2	2, 4	Quad-core Cortex-A 73 Dual -core Cortex-A53	Mali-G52 GPU	1000 Mbit/s Ethernet Optional WiFi USB adapters	microSD – card +eMMC

Дякую за увагу!

ДОДАТОК Б

Публікація

ISSN 2073-7394

Національний університет
"Полтавська політехніка імені Юрія Кондратюка"

National University
"Yuri Kondratyuk Poltava Polytechnic"

Системи управління, навігації та зв'язку

Control, navigation and communication systems

Випуск 2 (76)

Issue 2 (76)

Щоквартальне видання

Засноване у 2007 році

У журналі відображені результати наукових досліджень з розробки та удосконалення систем управління, навігації та зв'язку у різних проблемних галузях.

Засновник і видавець:
Національний університет
"Полтавська політехніка імені Юрія Кондратюка"

Телефон:
+38 (050) 302-20-71

E-mail редколегії:
kuchuk_nina@ukr.net

Інформаційний сайт:
http://journals.nupp.edu.ua/sunz

Quarterly

Founded in 2007

Journal represent the research results on the development and improvement of control, navigation and communication systems in various areas

Founder and publisher:
National University
"Yuri Kondratyuk Poltava Polytechnic"

Phone:
+38 (050) 302-20-71

E-mail of the editorial board:
kuchuk_nina@ukr.net

Information site:
http://journals.nupp.edu.ua/sunz

За достовірність викладених фактів, цитат та інших відомостей відповідальність несе автор

Журнал індексується міжнародними наукометричними базами: Index Copernicus (ICV = 82.05), General Impact Factor, Google Scholar, Academic Resource Index, Scientific Indexed Service

Затверджений до друку Вченою Радою Національного університету "Полтавська політехніка імені Юрія Кондратюка" (протокол від 30 квітня 2024 року № 5).

Свідоцтво про державну реєстрацію КВ № 24464-14404 ПР від 27.03.2020 р.

Включений до "Переліку наукових фахових видань України, в яких можуть публікуватися результати дисертаційних робіт на здобуття наукових ступенів доктора наук, кандидата наук та ступеня доктора філософії" до категорії Б – наказами МОН України від 17.03.2020 № 409 та від 09.02.2021 № 157

Полтава • 2024

ISSN 2073-7394 Системи управління, навігації та зв'язку. 2024. № 2

UDC 004.9 doi: 10.26906/SUNZ.2024.2.141

I. Radchenko, O. Shekhovtsov, A. Kovalenko, O. Sytnyk

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

FORMATION OF CLUSTERS ON SINGLE-BOARD COMPUTERS IN IOT NETWORKS

Abstract. The article looks at the problem of using single -board computers for Internet technology . An analysis of current single-board computers in various countries was carried out. Single-board computers and clusters of single-board computers have found their place in the concept of edge computing, allowing optimization of hard computing by placing computing resources closer to the core. The idea of a "virtual cluster" lies in the unification and organization of disparate heterogeneous devices for the development of various complex computing tasks from the available resources of the existing infrastructure of edge computing, what is known in the area. First of all, we have secured the resources of single-board computers. Such a cluster will also allow the use of resources from the existing infrastructure in a more efficient way, for example, by activating additional services from processing and saving data.

Keywords: single-board computer, cluster, Internet of Things, edge computing.

Introduction

Currently, all over the world, Internet of Things technologies (IoT , Internet of Things) are increasingly becoming an integral part of our daily life, for example: smart home, smart city, agriculture, medicine, logistics, etc. The huge amount of data generated by Internet of Things devices requires further post-processing and analysis, including the use of machine learning. To process incoming information, "clouds" are used, where data center facilities directly analyze them and make decisions based on the results obtained. However, the number of devices connected to the Internet is constantly increasing, which leads to difficulties in processing data in the "clouds" due to an increase in incoming data, increased load on the network, increased delays in data transfer between nodes, etc.

One solution to this issue is the development of edge computing technologies, which allow us to take on part of the load and reduce the response time to an event. To organize edge computing, single-board computers are used, which have compact sizes and high energy efficiency, but low performance, which does not allow solving computationally complex problems using the resources of a single device. One solution to this problem is to create a local cluster consisting of computers with limited computing resources.

Analysis of current single board computers in various fields

Single board computer is a self-contained computer assembled on a single printed circuit board on which all the necessary components are installed to ensure its functioning: processor, RAM, input-output systems, etc.

The first truly single-board computer appeared back in 1976 and was called "MMD-1" (Mini-Micro Designer 1), but at the same time this format of computer execution became truly widespread and accessible only in 2012 with the advent of Raspberry Pi [1]. This computer is primarily aimed at the educational sphere and training in computer use, programming, etc. With a low price and sufficient performance to run a full-fledged Linux- based operating system, it has gained popularity among many enthusiasts and researchers from various fields. Thanks to the success of Raspberry Pi and other manufacturers began to offer their models of single-board computers and currently there are more than 200 models of single-board computers in the world. The most popular are various Raspberry models Pi 3 and 3b+ , RP Zero W and 2020 model Raspberry Pi 4, which has several variants with 2 GB, 4 GB and 8 GB of RAM. Among other manufacturers, it is worth noting Banana Pi M5, RockPro64, Odroid N2. Comparative characteristics of single-board computers are presented in Table 1.

Table 1 – **Comparison of characteristics of single-board computers**

	RAM (GB)	SoC	GPU	Ethernet	Storage
Raspberry Pi 3b+	1 LPDDR2	ARM Cortex A53	VideoCore IV	330 Mbit Ethernet, 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE	microSD cards
Raspberry Pi 4	2, 4, 8 LPDDR4	4 x Cortex-A72	VideoCore VI	1000 Mbit/s Ethernet 802.11b/g/b/ac WiFi 5 and Bluetooth 5.0	microSD cards
RP Zero W	0.5	1 x ARM1176JZFS	VideoCore IV	802.11 b/g/n wireless LAN, BLE 4.1	microSD cards
RockPro64	4 LPDDR 4	4 x ARM CortexA53 2x ARM Cortex A72	MaliT860-MP4	1000 Mbit/s Ethernet	slot for eMMC module
Banana Pi M5	4 GB LPDDR4	Quad-Core Cortex-A55	Mali-G31 GPU	1000 Mbit/s Ethernet	microSD – card + eMMC
Odroid N2	2, 4	Quad-core Cortex-A 73 Dual -core Cortex-A53	Mali-G52 GPU	1000 Mbit/s Ethernet Optional WiFi USB adapters	microSD – card +eMMC

ДОДАТОК В

Текст програм та запитів

```
# НАЛАГОДЖЕННЯ ПРОТОКОЛУ IOT

import paho.mqtt.client as mqtt

def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("iot/test")

def on_message(client, userdata, msg):
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.connect("iot.eclipse.org", 1883, 60)
client.loop_forever()

# ПІДКЛЮЧЕННЯ ДАТЧИКА

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while True:
    GPIO.output(18, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(18, GPIO.LOW)
    time.sleep(1)

# ЗВ'ЯЗОК ІЗ ХМАРОЮ

from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
import time

myMQTTClient = AWSIoTMQTTClient("myClientID")
myMQTTClient.configureEndpoint("YOUR.ENDPOINT", 8883)
myMQTTClient.configureCredentials("ROOT_CA.pem", "PRIVATE_KE
Y.pem", "CERTIFICATE.pem")

myMQTTClient.connect()
print("Connected to AWS IoT")

while True:
    temperature = read_temperature_sensor()  # дані з датчику
    myMQTTClient.publish("iot/temperature", str(temperature),
```

```
1)
    time.sleep(60)

# КЕРУВАННЯ ПОРТАМИ

# importing the required modules
from gpiozero import Button
from time import sleep

# creating an object of Button
the_button = Button(2)

# using the if-else statement
while True:
    if the_button.is_pressed:
        print("Button Pressed")
    else:
        print("Button Released")
    sleep(1)

# КЕРУВАННЯ ДАТЧИКОМ

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD) # or GPIO.setmode(GPIO.BCM)
GPIO.setup(12, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(16, GPIO.OUT)
pwm_led = GPIO.PWM(23, 500)  //(1)
pwm_led.start(100)
GPIO.add_event_detect(18, GPIO.FALLING, callback=press, bounce
time=100)  //(2)
GPIO.add_event_detect(18, GPIO.RISING, callback=unpress, bounce
time=100)

def press():
     pwm_led.ChangeDutyCycle(50)  //(3)

def unpress():
     pwm_led.ChangeDutyCycle(100)

try:
while True:
     pass  //(4)
finally:
GPIO.cleanup()

# ВСТАНОВЛЕННЯ РЕАЛЬНОГО ЧАСУ

{
  "trip":{
     "trip_id":"120700_A..N",
     "start_time":"20:07:00",
     "start_date":"20220531",
```

```
      "route_id":"A"
   },
   "stop_time_update":[
      {
         "arrival":{
            "time":1654042672
         },
         "departure":{
            "time":1654042672
         },
         "stop_id":"H06N"
      },

      {
         "arrival":{
            "time":1654044957
         },
         "departure":{
            "time":1654044957
         },
         "stop_id":"A42N"
      }
   ]
}

# НАЛАГОДЖЕННЯ БУФЕРУ

def station_time_lookup(train_data, station):
    for trains in train_data:
        if trains.__contains__('trip_update'):
            unique_train_schedule = trains['trip_update']
            if unique_train_schedule.__contains__('stop_time
_update'):
                unique_arrival_times = unique_train_schedule['s
top_time_update']
                for scheduled_arrivals in unique_arrival_times:
                    stop_id = scheduled_arrivals.get('stop_id',
False)
                    if stop_id == f'{station}N':
                        time_data = scheduled_arrivals['arrival']
                        unique_time = time_data['time']
                        if unique_time != None:
                            northbound_times.append(unique_time)
                    elif stop_id == f'{station}S':
                        time_data = scheduled_arrivals['arrival']
                        unique_time = time_data['time']
                        if unique_time != None:
                            southbound_times.append(unique_time)

northbound_times = []
southbound_times = []

station_time_lookup(realtime_data, 'A42')
```

```
# ОБРОБКА ДАНИХ      I

# сортування отриманих показань в хронологічному порядку
northbound_times.sort()
southbound_times.sort()

nearest_northbound_arrival_time = northbound_times[0]
second_northbound_arrival_time = northbound_times[1]

nearest_southbound_arrival_time = southbound_times[0]
second_southbound_arrival_time = southbound_times[1]

def print_train_arrivals(
        direction,
        time_until_train,
        nearest_arrival_time,
        second_arrival_time):
    if time_until_train <= 0:
        next_arrival_time = second_arrival_time
    else nearest_arrival_time:
        next_arrival_time_s = time.strftime(
            "%I:%M %p",
            time.localtime(next_arrival_time))
    print(f"The next {direction} train will arrive at {ne
xt_arrival_time_s}")

current_time = int(time.time())
time_until_northbound_train = int(
    ((nearest_northbound_arrival_time - current_time) / 60))
time_until_southbound_train = int(
    ((nearest_southbound_arrival_time - current_time) / 60))
current_time_s = time.strftime("%I:%M %p")
print(f"It's currently {current_time_s}")

print_train_arrivals(
    "northbound",
    time_until_northbound_train,
    nearest_northbound_arrival_time,
    second_northbound_arrival_time)
print_train_arrivals(
    "southbound",
    time_until_southbound_train,
    nearest_southbound_arrival_time,
    time_until_southbound_train)

# СТВОРЕННЯ КЛІЄНТА

def create_client():
    # Instantiate the client
    client = IoTHubDeviceClient.create_from_connection_strin
g(CONNECTION_STRING)
```

```
    # Define the handler for method requests
    def method_request_handler(method_request):
        if method_request.name == "rebootDevice":
            # Act on the method by rebooting the device
            print("Rebooting device")
            time.sleep(20)
            print("Device rebooted")

            # ...and patching the reported properties
            current_time = str(datetime.datetime.now())
            reported_props = {"rebootTime": current_time}
            client.patch_twin_reported_properties(reporte
d_props)
            print( "Device twins updated with latest rebootTime
")

            # Create a method response indicating
            resp_status = 200
            resp_payload = {"Response": "This is the response
"}
            method_response = MethodResponse(method_request.re
quest_id, resp_status, resp_payload)

        else:
            # Create a method response indicating the method
            resp_status = 404
            resp_payload = {"Response": "Unknown method"}
            method_response = MethodResponse(method_request.re
quest_id, resp_status, resp_payload)

        # Send the method response
        client.send_method_response(method_response)

    try:
        # Attach the handler to the client
        client.on_method_request_received = method_request_han
dler
    except:
        # In the event of failure, clean up
        client.shutdown()

    return client

# ПРИКЛАД ПРЯМОГО ДОСТУПУ

def main():
    print ("Starting the IoT Hub Python sample...")
    client = create_client()

    print ("Waiting for commands, press Ctrl-C to exit")
    try:
        # Wait for program exit
        while True:
```

```
            time.sleep(1000)
    except KeyboardInterrupt:
        print("IoTHubDeviceClient sample stopped")
    finally:
        # Graceful exit
        print("Shutting down IoT Hub Client")
        client.shutdown()

if __name__ == '__main__':
    main()

# ВИКЛИК ДВІЙНИКА ПРИСТРОЮ

def iothub_devicemethod_sample_run():
    try:
        # Create IoTHubRegistryManager
        registry_manager = IoTHubRegistryManager(CONNECTION_S
TRING)

        print ( "" )
        print ( "Invoking device to reboot..." )

        # Call the direct method.
        deviceMethod = CloudToDeviceMethod(method_name=ME
THOD_NAME, payload=METHOD_PAYLOAD)
        response = registry_manager.invoke_device_method(DEVICE
_ID, deviceMethod)

        print ( "" )
        print ( "Successfully invoked the device to reboot." )

        print ( "" )
        print ( response.payload )

        while True:
            print ( "" )
            print ("IoTHubClient waiting for commands, press
Ctrl-C to exit")

            status_counter = 0
            while status_counter <= WAIT_COUNT:
                twin_info = registry_manager.get_twin(DEVICE_I
D)

                if twin_info.properties.reported.get("rebootTi
me") != None :
                    print ("Last reboot time: " + twin_in
fo.properties.reported.get("rebootTime"))
                else:
                    print ("Waiting for device to report last
reboot time...")

                time.sleep(5)
```

```
                status_counter += 1

    except Exception as ex:
        print ( "" )
        print ( "Unexpected error {0}".format(ex) )
        return
    except KeyboardInterrupt:
        print ( "" )
        print ( "IoTHubDeviceMethod sample stopped" )

if __name__ == '__main__':
    print ( "Starting the IoT Hub Service Client DeviceManageme
nt Python sample..." )
    print ( "    Connection string = {0}".format(CONNECTION_S
TRING) )
    print ( "    Device ID         = {0}".format(DEVICE_ID) )

    iothub_devicemethod_sample_run()

# ОНОВЛЕННЯ ДВІЙНИКА ПРИСТРОЮ

var twinPatch = {
   etag: '*',
   properties: {
       desired: {
           building: '43',
           floor: 3
       }
   }
};

var twinJobId = uuid.v4();

console.log('scheduling Twin Update job with id: ' + twinJobI
d);
jobClient.scheduleTwinUpdate(twinJobId,
                             queryCondition,
                             twinPatch,
                             startTime,
                             maxExecutionTimeInSeconds,
                             function(err) {
    if (err) {
        console.error('Could not schedule twin update job: ' +
err.message);
    } else {
        monitorJob(twinJobId, function(err, result) {
            if (err) {
                console.error('Could not monitor twin update
job: ' + err.message);
            } else {
                console.log(JSON.stringify(result, null, 2));
            }
        });
```

```
    }
});
```