1. БАЗОВОЕ СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Системные программные средства, для которых обеспечивается эффективная работа Программы:

- операционная система для серверов: Ubuntu 20.04;
- операционные системы для рабочих станций: Windows 7, Windows 8, Windows 10.

На сервере необходимо следующее программное обеспечение:

- Python version 3.8;
- Docker version 20.10.5;
- Docker-compose version 1.27.4;
- система управления базами данных PostgresSQL version 13.1;
- распределенный программный брокер сообщений Apache Kafka version 2.13-2.6.0;
- распределенный сервис конфигурирования и синхронизации Apache ZooKeeper version 3.5.8.

На клиентских рабочих станциях необходимо следующее программное обеспечение:

- браузер Google Chrome 51.0 и выше или Mozilla FireFox версии 14 и выше.

2. ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Установленные для эксплуатации Программы технические средства (персональные компьютеры, принтеры, устройства резервного хранения данных, сетевые компоненты) должны быть совместимы между собой и поддерживать сетевой протокол TCP/IP.

Для работы Программы используется «IBM-совместимые» компьютеры с операционной системой Windows, Linux.

Серверные компоненты Программы должны быть установлены на выделенном сервере, предназначенном исключительно для эксплуатации серверных компонент Программы.

Технические характеристики оборудования, на котором ведется эксплуатация ПО, определяются исходя из требуемого количества процессов видеоаналитики.

Ниже представлены минимальные технические характеристики серверного оборудования для запуска анализа, необходимые для одной камеры:

- двухпроцессорная система 2.2GHz;
- оперативная память 16GB;

- свободное дисковое пространство 40GB (+ дополнительное пространство для размещения прикладных систем и баз данных, размер которого зависит от количества производимых анализов).

Точных требований к техническим характеристикам клиентских компьютеров не предъявляется. Для работы с Программой необходимо наличие браузеров, указанных в п. 1.

Установка должна производиться на операционную систему **Ubuntu 20.04.**

Перед установкой необходимо убедиться, что в системе установлены следующие сервисы:

- Python version 3.8;
- Docker version 20.10.5;
- Docker-compose version 1.27.4;
- система управления базами данных PostgresSQL version 13.1;
- распределённый программный брокер сообщений Apache Kafka version 2.13-2.6.0;
- распределенный сервис конфигурирования и синхронизации Apache ZooKeeper version 3.5.8;
- система управления конфигурациями Ansible-playbook version 2.5.1.

Все пакеты устанавливаются из стандартных репозиториев.

Процедура установки состоит из четырех этапов:

1. Установка и настройка окружения программного брокера сообщения Kafka;
2. Установка и конфигурация основной базы данных;
3. Конфигурация и запуск консюмера;
4. Установка программного обеспечения.

Процесс установки программного брокера сообщения Kafka:

1. Скопировать на сервер архив srv.zip.
2. Распаковать его в директорию /srv.
3. С помощью команды *sudo nano /srv/ansible/init_playbook/inventory-new.yml* отредактировать файл inventory-new.yml:

```
---
# Project inventory
all:
  children:
    eputs:
      vars:
        project:
          name: eputs
      children:
        eputs_prod:
          children:
            eputs_prod_kafka:
              vars:
                project:
                  env: prod
                kafka:
                  cluster:
                    - название_сервера
              hosts:
                название_сервера:
                  ansible_host: 127.0.0.1
```

4. Выполнить команду *source venv/bin/activate*.
5. Выполнить команду *sudo apt-get install openssh-server*.
6. Перейти в директорию командой *cd /etc/sudoers*.
7. Создать файл командой *sudo nano kafka* и добавить в него:

```
kafka ALL = (ALL:ALL) NOPASSWD:ALL
```

8. Создать файл командой *sudo nano zookeeper* и добавить в него:

```
zookeeper ALL = (ALL:ALL) NOPASSWD:ALL
```

9. Создать файл командой *sudo nano имя_пользователя_сервера* и добавить в него:

```
Имя_пользователя_сервера ALL = (ALL:ALL) NOPASSWD:ALL
```

10. Выполнить команду *cat/home/имя_пользователя/.ssh/id_rsa.pub* и скопировать содержимое файла.
11. Вставить содержимое в файл /home/имя_пользователя/.ssh/authorized_keys.
12. Создать файл командой *sudo nano /etc/systemd/system/kafka.service* и добавить в него:

```
[Unit]
equires=zookeeper.service
After=zookeeper.service
[Service]
Type=simple
User=kafka
ExecStart=/srv/kafka-release/kafka_2.13-2.6.0/bin/kafka-server-start.sh /srv/kafka-release/kafka_2.13-2.6.0/config/server.properties
ExecStop=/srv/kafka-release/kafka_2.13-2.6.0/bin/kafka-server-stop.sh
Restart=on-abnormal
[Install]
WantedWantedBy=multi-user.target
```

13. Перейти в директорию cd /srv/smartroadsconsumer/smartroadsconsumer.
14. Выполнить команду *ansible-playbook playbook.yml -i inventory-new.yml -Dvv -t kafka*.

Процесс установки основной базы данных:

1. Скачать пакет командой *wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add –*.
2. Добавить репозиторий echo "deb http://apt.postgresql.org/pub/repos/apt/ lsb_release -cs-pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list.
3. Выполнить команду *sudo apt update*.
4. Установить пакет *sudo apt install postgresql-13 postgresql-client-13*.
5. Открыть файл командой *sudo nano /etc/postgresql/13/main/pg_hba.conf*.

Внести в файл следующие корректировки:

```
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     peer
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5
host    all             all             0.0.0.0/0               md5
```

6. Перезагрузить службу *systemctl restart postgresql.service*.
7. Сменить пользователя командой *sudo - su postgres*.
8. Перейти в CLI командой *psql*.
9. Выполнить команды создания роли и базы данных:

CREATE ROLE eputs_prod LOGIN PASSWORD 'rshughghavgrsvjkhvajeyvyawiuyr';
CREATE DATABASE eputs_prod.

10. Активировать окружение *source venv2/bin/activate*.
11. Экспортировать переменные, последовательно выполнив команды:

```
export POSTGRES_HOST=127.0.0.1
export POSTGRES_PORT=5432
export POSTGRES_DB=eputs_prod
export POSTGRES_USER=eputs_prod
export POSTGRES_PASSWORD=rshughghavgrsvjkhvajeyvyawiuyr
export    KAFKA_BOOTSTRAP_SERVERS=127.0.0.1:9092
```

12. Выполнить миграции *alembic upgrade head*.

Конфигурация и запуск консюмера:

1. Перейти в директорию командой *cd /etc/systemd/system*.
2. Создать файл командой *sudo nano smartroadsconsumer.service* и добавить в него:

```
[Unit]
Description=smartroadsconsumer
After=network

[Service]
Type=simple
User=имя_пользователя_сервера
WorkingDirectory=/srv/smartroadsconsumer/smartroadsconsumer/
ExecStart=/usr/bin/python3.6 -m eputs_consumer consumer
Restart=on-abort
EnvironmentFile=/srv/smartroadsconsumer/.env

[Install]
WantedBy=multi-user.target
```

3. Выполнить перезагрузку демона командой *systemctl daemon-reload*.
4. Активировать консюмер *systemctl start smartroadsconsumer.service*.

Процесс установки программного обеспечения:

1. Предварительно на сервер должны быть загружены следующие docker-контейнеры:

- docker_gui_2;
- docker_monitoring_2;
- nginx_proxy;
- postgres;
- docker_app_2;
- docker_redis_1.

Также на сервер должны быть загружены файлы:

- docker-compose-prod.yaml (конфигурационный файл).

И файлы весов нейронных сетей:

- yolov4.weights;
- grn_best.pt (веса yolov5 для ГРЗ);
- dtp_best.pt (веса сетки для детекции ДТП);
- new_weights.h5 (веса сетки для распознавания текста).

2. После загрузки контейнеров и yaml-файла необходимо поменять значения следующих переменных окружения:

VIDEO_FRAMES_PATH – путь к папке на сервере, в которую будут сохраняться кадры;

YOLOV4_WEIGHTS – абсолютный путь к весам yolov4. Файл yolov4.weights;

YOLOV5GRN_WEIGHTS – абсолютный путь к весам yolov5 для ГРЗ (необязательный параметр, если OCR – False). Файл grn_best.pt;

TEXT_DETECTOR_WEIGHTS – абсолютный путь к весам сетки для распознавания текста (необязательный параметр, если OCR – False). Файл new_weights.h5;

YOLOV5DTP_WEIGHTS – абсолютный путь к весам сетки для детекции ДТП. Файл dtp_best.pt.

В Приложении 1 к настоящей инструкции перечислены все переменные окружения контейнеров, которые позволяют точно настроить работу.

Запуск системы:

Запуск приложений происходит с помощью файла docker-compose-prod.yaml командой:

docker-compose -f <имя yaml файла> up -d.

Приложение №1

Описание переменных окружения контейнеров:

1. docker_gui:

Общие настройки:
LOG_TYPE – watched || stream - писать логи в файл или в стрим;
LOG_LEVEL – уровень логирования;
LOG_DIR – путь к папке с логами;
APP_LOG_NAME – имя файла с логами приложения;
WWW_LOG_NAME – имя файла с логами http-запросов;
LOG_MAX_BYTES – максимальный размер файла с логами (по умолчанию 100 мб);
LOG_COPIES – число копий (по умолчанию 5);
APP_HOST – str, <ip/имя контейнера>:port с аналитикой;
POSTGRES_LOG_DIR – путь к директории с логами postgres;
POSTGRES_LOG_NAME – имя файла с логами postgres;
KAFKA_LOG_DIR – путь к директории с логами kafka;
POSTGRES_LOG_DIR – имя файла с логами kafka.

Настройки postgres GUI:
GUI_POSTGRES_PASSWORD;
GUI_POSTGRES_USER;
GUI_POSTGRES_DB;
GUI_POSTGRES_HOST;
GUI_POSTGRES_PORT.

Настройки postgres с отчетами:
CON_POSTGRES_PASSWORD;
CON_POSTGRES_USER;
CON_POSTGRES_DB;
CON_POSTGRES_HOST;
CON_POSTGRES_PORT.

2. docker_monitoring:

APP_HOST – str, <ip/имя контейнера>:port с аналитикой;

MONITORING_INTERVAL – int, частота мониторинга скриптов, читающих видео (в секундах);

REFRESH_PROCESS_DATA – int, интервал обновления списка запущенных процессов для чтения кадров (в секундах);

LOGS_PATH – str, абсолютный путь к папке с логами.

Настройки redis:

REDIS_HOST;

REDIS_PORT;

REDIS_PASSWORD.

3. nginx

В данной конфигурации монтируется файл с настройками nginx.conf.

4. docker_app:

Общие настройки:
BATCH_SIZE – int, размер батча для подачи сеткам;
EXPECTED_BATCH_TIME – float, ожидаемое время обработки батча для регуляции скорости чтения кадров из видеопотока (в секундах, по умолчанию 1.6);
DEBUG – 'true' или 'false', включить вывод дополнительных логов;
VIDEO_CAPTURE – str, какой библиотекой читать кадры: FFMPEG || CV2;
VIDEO_PROVIDER – str, откуда получать кадры: FFMPEG || CV2 || REDIS;
VIDEO_FILES_PATH – str, абсолютный путь для сохранения файлов;
VIDEO_FRAMES_PATH – str, абсолютный путь для сохранения кадров;
SAVE_ORIG_INTERVAL – float, интервал сохранения кадров без Bounding Box (в секундах);
ORIG_VIDEO_FRAMES_PATH – str, абсолютный путь для сохранения кадров без Bounding Box;
LOGS_PATH – str, абсолютный путь к папке с логами;
REPORTS_PATH – str, абсолютный путь к папке с отчетами;
GUI_HOST – str, <ip/имя контейнера>:port с GUI;
APP_HOST – str, <ip/имя контейнера>:port с аналитикой.

Параметры отчетов:
AGGREGATE – 'true' или 'false', включить составление агрегированных отчетов;
AGGREGATION_INTERVAL – int, интервал накопления агрегированных данных для отчета (в секундах);
URL_PREFIX – str, префикс url для доступа к кадру;
MONITORING_INTERVAL – int, интервал отправки статуса камеры в кафку (в секундах).

Параметры весов сеток - файлы с весами должны быть примонтированы к контейнеру (указаны в разделе volumes yaml файла в формате <путь к файлу на хосте>:<путь к файлу в контейнере>).

Под абсолютным путем имеется в виду путь в контейнере:
YOLOV4_WEIGHTS – абсолютный путь к весам yolov4;
YOLOV5GRN_WEIGHTS – абсолютный путь к весам yolov5 для ГРЗ (необязательный параметр, если OCR – False);

TEXT_DETECTOR_WEIGHTS – абсолютный путь к весам сетки для распознавания текста (необязательный параметр, если OCR – False);

YOLOV5DTP_WEIGHTS – абсолютный путь к весам сетки для детекции ДТП.

Настройки выделения памяти на GPU под сетки tensorflow (заранее заданный лимит или динамическое выделение):

USE_MEMORY_LIMITS – 'true' или 'false', включить ограничение памяти на GPU;

YOLOV5GRN_MEMORY_LIMIT – int, лимит памяти для yolov5 для ГРЗ (по умолчанию 0);

YOLOV5DTP_MEMORY_LIMIT – int, лимит памяти для yolov5 для ДТП (по умолчанию 0);

TEXT_DETECTOR_MEMORY_LIMIT – int, лимит памяти сетки для распознавания текста (по умолчанию 256);

YOLOV4_MEMORY_LIMIT – int, лимит памяти сетки для распознавания текста (по умолчанию 1024);

MEM_UTIL_MAX – float, максимальный процент занятой видеопамяти GPU, для запуска на нем видеоанализа (по умолчанию 1);

GPU_UTIL_MAX – float, максимальный процент загруженности GPU, для запуска на нем видеоанализа (по умолчанию 1).

Следующие параметры учитываются только если в базе данных GUI в таблице configs нет конфигураций с такими именами или эта база данных недоступна:

OCR – 'true' или 'false', включить распознавание ГРЗ;

DTP – 'true' или 'false', включить детекцию ДТП;

PERES_POLOS – 'true' или 'false', включить детекцию пересечения полос;

ZAPRET_ZONE – 'true' или 'false', включить детекцию нахождения в запретной зоне;

DIRECTION – 'true' или 'false', включить детекцию направления;

SPEED – 'true' или 'false', включить детекцию скорости;

CAMERA_SHIFT – 'true' или 'false', включить детекцию сдвига камеры;

SAVE_FRAMES – 'true' или 'false', включить сохранение кадров;

SAVE_INTERVAL – float, интервал сохранения кадров (в секундах);

SAVE_ORIG_FRAMES – 'true' или 'false', включить сохранение кадров без Bounding Box;

DTP_THRESHOLD – float, трешхолд ДТП (по умолчанию 0.9);

COLOR_THRESHOLD – float, трешхолд цвета (по умолчанию 0);

MAKER_THRESHOLD – float, трешхолд марки/модели (по умолчанию 0).

Настройки для эвристик:

MAX_SPEED – float, в отчете отправлять скорость не больше, чем указанная (в м/с);

DEFAULT_FPS – float, если не получается прочитать fps видео или fps высокий (больше 100), считать, что fps равен указанному;

MIN_AREA – float, минимальная площадь Bounding Box для добавления в отчет;

DIRECTION_MIN_SPEED – int, минимальная скорость для определения направления ТС;

PERES_POLOS_THRESHOLD – float, трешхолд пересечения полос (по умолчанию 0);

ZAPRET_ZONE_THRESHOLD – float, трешхолд нахождения в запретной зоне (по умолчанию 0.5);

STOPPING_THRESHOLD – float, трешхолд остановки ТС, (по умолчанию 0.5).

Настройки kafka:
KAFKA_SEND_DATA_ENABLE – 'true' или 'false', слать отчеты в кафку;
KAFKA_SERVER – str, ip:port кафки;
KAFKA_BATCH_SIZE – int, размер буфера, в котором копятся данные для отправки в кафку;
KAFKA_LINGER_MS – int, время ожидания заполнения буфера;
KAFKA_TOPIC – str, имя топика обычных отчетов;
KAFKA_TOPIC_MONITORING – str, имя топика отчетов о статусе камеры;
KAFKA_TOPIC_AGGREGATION – str, имя топика агрегированных отчетов.

Настройки redis:
REDIS_HOST;
REDIS_PORT;
REDIS_PASSWORD.

Настройки postgres GUI:
GUI_POSTGRES_PASSWORD;
GUI_POSTGRES_USER;
GUI_POSTGRES_DB;
GUI_POSTGRES_HOST;
GUI_POSTGRES_PORT.

Настройки postgres с отчетами:
CON_POSTGRES_PASSWORD;
CON_POSTGRES_USER;
CON_POSTGRES_DB;
CON_POSTGRES_HOST;
CON_POSTGRES_PORT.

Настройки визуализации Bounding Box (BBox):
FONT_FAMILY – файл со шрифтом (по умолчанию 'consola.ttf');
FONT_SIZE – int, размер шрифта (по умолчанию 20);
BBOX_WIDTH – int, толщина BBox в пикселях (по умолчанию 3);
SHORT – 'true' или 'false', краткий вывод информации о BBox (по умолчанию true).