COGNITIVE SOLUTIONS

IVA CV

Руководство администратора

Развертывание, конфигурирование и администрирование платформы видеоаналитики IVA CV

ВВЕДЕНИЕ

В этом документе приведена техническая документация для системного администратора, касающаяся развёртывания, конфигурирования и администрирования платформы видеоаналитики IVA CV.

IVA CV – это универсальная высокопроизводительная платформа видеоаналитики и интеллектуального видеонаблюдения.

Архитектура платформы позволяет компонентам платформы эффективно работать как в условиях больших ЦОД, приватных и публичных облаков, с поддержкой масштабируемости и отказоустойчивости, так и на небольших (edge) устройствах – одноплатных компьютерах, неттопах, умных камерах, смартфонах и т.д.

Платформа поддерживает широкий спектр специализированных чипов для аппаратной акселерации обработки видео и инференса нейронных сетей, включая чипы, разработанные компаниями Nvidia, Intel, AMD, Rockchip, Apple, IVA (TPU).

Архитектура платформы разработана с учетом большого разнообразия задач, связанных с аналитической обработкой видео/аудио данных и поддерживает очень высокий уровень расширяемости и адаптивности к конкретным задачам.

В этом документе нет деталей, касающихся стандартных компонентов и технологий типа Linux, Docker, Mongodb и т.д. Документация по таким технологиям доступна в сети Интернет.

ОГЛАВЛЕНИЕ

1. КОМПОНЕНТЫ ПЛАТФОРМЫ IVA CV

На диаграмме ниже приведён верхнеуровневый взгляд на компоненты платформы. Разбивка компонентов на сетевые сегменты указана в качестве примера и может изменяться при развертывании платформы.

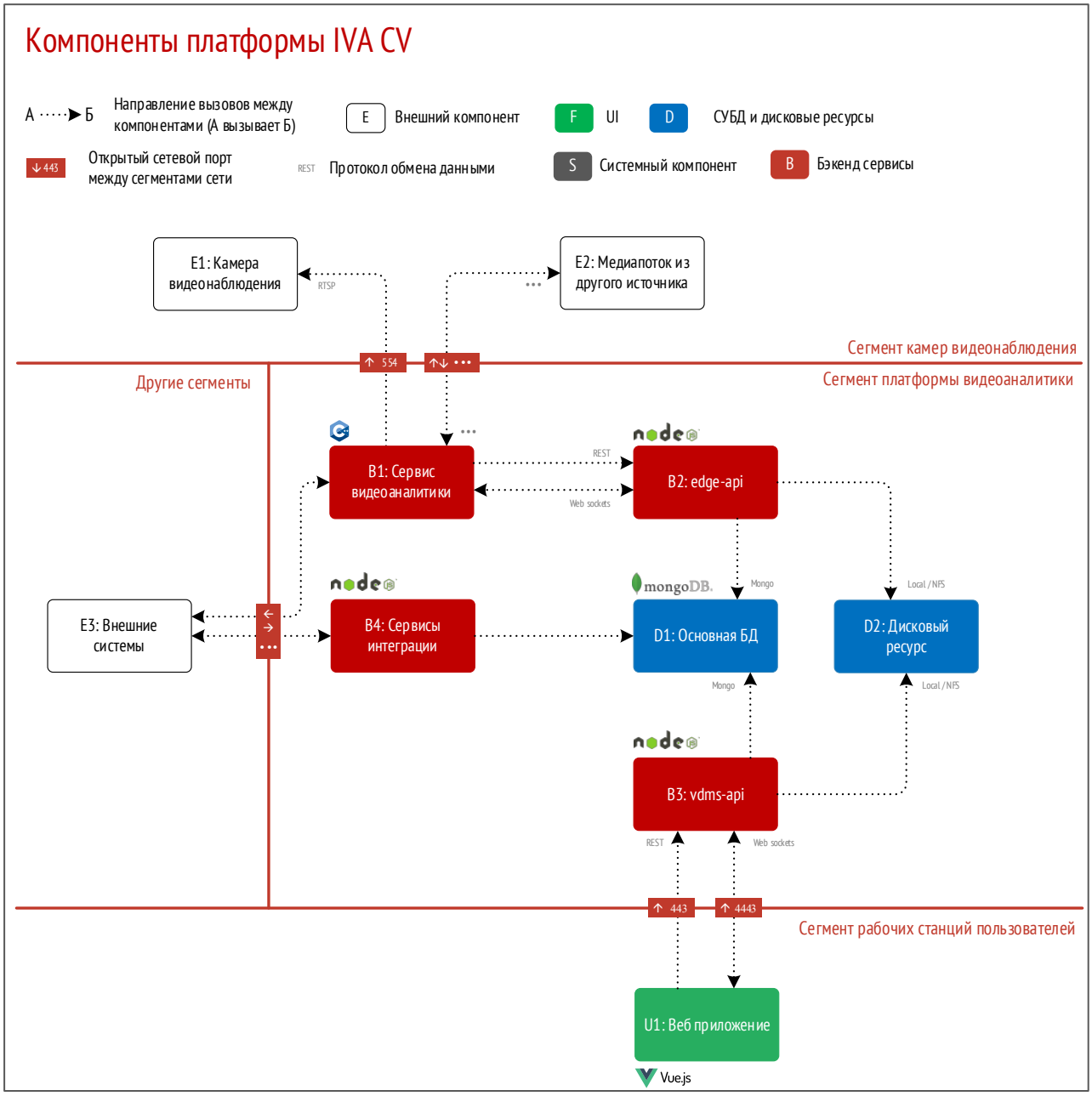

Рисунок 1. Компоненты платформы IVA CV

2. ТРЕБОВАНИЯ К ОБОРУДОВАНИЮ

Платформа видеоаналитики поддерживает работу всех компонентов платформы на следующих микропроцессорах:

- Intel Core i7, i9
- Intel Xeon

- AMD
- ARM

Платформа видеоаналитики поддерживает работу компонентов видеоаналитики с аппаратной поддержкой декодирования и кодирования видеопотоков, а также аппаратной акселерации инференса нейронных сетей на следующих чипах (GPU/NPU/TPU):

- Nvidia (с архитектурой не ниже Turing)
- Intel Iris
- Rockchip RK3588
- Linq (российский TPU)

Минимальный объем оперативной памяти: 8 Гб.

Минимальный объем оперативной жесткого диска: 256 Гб.

3. ТРЕБОВАНИЯ К ОПЕРАЦИОННЫМ СИСТЕМАМ

Платформа видеоаналитики поддерживает работу всех компонентов на любой версии Linux, с версией ядра Linux не ниже 6.0.

В операционной системе должны быть установлены необходимые драйверы для чипов (Nvidia, Intel, Rockchip и т.д.), в соответствии с документацией производителей соответствующих чипов.

В операционной системе должна быть установлена последняя версия Docker. Для использования с чипами Nvidia, необходимо установить специальную версию контейнерного райнтайма Nvidia Container Toolkit:

https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html

4. ДОМАШНИЙ КАТАЛОГ

Все файлы всех компонентов платформы располагаются в домашнем каталоге на сервере. Домашний каталог определяется при инсталляции компонентов системы. Домашних каталогов может быть несколько на одном сервере, с отдельными экземплярами компонентов системы.

Домашний каталог содержит в себе каталоги компонентов, а также общий каталог для данных.

Пример структуры домашнего каталога:

/msdata

/edge-api-1 - каталог компонента edge-api

/mongodb-1 - каталог БД Mongodb

/vdms-api-1 - каталог компонента vdms-api

/vdms-ui - каталог компонента vdms-ui

/va-node-1 - каталог компонента видеоаналитики

/shared - общий каталог для данных

Содержимое каталогов компонентов приведено в описании каждого компонента, в этом документе.

Пример содержимого общего каталога для данных:

/msdata/shared

/archive – архив видеозаписей

/chunks – вырезанные фрагменты видеозаписей для событий

/clusters – каталог с данными актуальной конфигурации кластеров

/crops - вырезанные фрагменты изображений для событий

/facesdb – БД лиц в файловой системе

/files – Различные файлы, используемые в UI (логотипы, карты, фото сотрудников и т.д.)

/frames – Изображения с кадрами событий

/input – Видеофайлы, используемые для эмуляции камер

/metric – Метрики производительности вычислительных узлов видеоаналитики

/output – Сгенерированные системой видеофайлы

/services – Инсталляционные пакеты сервисов (для облачной конфигурации системы)

/streams – Буферы видеопотоков

/vehiclesdb – БД автотранспорта в файловой системе

5. ОСНОВНАЯ БД

В качестве основной БД системы используется Mongodb версии 7 и выше, работающая в контейнере Docker.

БД должна быть настроена в режиме кластера (даже если экземпляр БД только один). Каждый экземпляр кластера БД должен быть именованным, первый

называется по умолчанию mongodb-1, второй mongodb-2 и так далее. Сетевое имя при подключении к этому экземпляру БД должно соответствовать имени этого экземпляра.

Пример настройки БД Mongodb в режиме кластера:

```
rs.initiate( {
   _id : "rs0",
   members: [
      { _id: 0, host: "mongodb-1:27017" },
   ]
})
```

Соответственно, для подключения к этому экземпляру БД должно использоваться имя mongodb-1.

Пример содержимого каталога БД mongodb:

/msdata/mongodb-1

/data/

db – Каталог с данными БД

dump – Каталог с дампами БД

logs – Каталог для файловых логов

/scripts/

logs.sh – Скрипт для просмотра консольных логов Mongodb

run.sh – Скрипт для запуска Mongodb

shell.sh – Скрипт для входа в консоль контейнера

stop.sh – Скрипт для остановки Mongodb

Пример содержимого скрипта запуска контейнера с Mongodb:

```
#!/bin/bash

docker run --restart=always --name mongodb-1 -e TZ=Europe/Moscow -e PGTZ=Europe/Moscow -p 27017:27017 \
-v /msdata/mongodb-1/data/db:/data/db \
-v /msdata/mongodb-1/data/logs:/var/log/mongodb \
-v /msdata/mongodb-1/data/dump:/dump \
-e HOST_HOSTNAME="mongodb-1" \
-h=mongodb-1 \
-v /etc/localtime:/etc/localtime \
-d mongo:7 --replSet rs0
```

В этом примере:

- Каталог с данными БД монтируется в контейнер из каталога хоста: /msdata/mongodb-1/data/db

- Каталог с логами БД монтируется в контейнер из каталога /msdata/mongodb-1/data/logs
- Каталог с дампами БД монтируется в контейнер из каталога /msdata/mongodb-1/data/dumps
- Имя экземпляра кластера указывается как mongodb-1
- Название кластера кластера указывается как rs0

Дополнительными параметрами конфигурации Mongodb можно управлять как с помощью параметров запуска контейнера (по аналогии с --replSet), так и примонтировав внутрь контейнера стандартный файл конфигурации Mongodb.

6. КОМПОНЕНТ ВИДЕОАНАЛИТИКИ

Этот компонент является основными вычислительным узлом видеоаналитики, который получает данные с камер (или файлов, или других видов видеопотоков), запускает пайплайны обработки этих видеопотоков и отправляет полученные результаты (видеопотоки, события, вырезанные изображения) в компонент Edge-api.

Компонент видеоаналитики сам по себе не является единым и стабильным приложением, это конструктор, на базе которого, для каждого проекта собирается сборка, содержащая нейронные сети, код и параметры конфигурации для конкретно этого проекта.

Системный администратор при поддержке решения не управляет параметрами пайплайнов видеоаналитики, это происходит силами вендора. Тем не менее, некоторыми параметрами работы видеоаналитики можно управлять из пользовательского интерфейса системы:

- Камеры и параметры подключения к ним
- Рабочие зоны камер различного вида
- Значения отсечки уверенности (confidence) для нейронных сетей

В зависимости от сборки, компонент видеоаналитики может выполнять множество дополнительных функций, помимо непосредственно обработки видеопотоков, например:

- Предоставление API (REST, gRPC, Websockets) для различных приложений
- Управление различными устройствами (электронные замки, шлагбаумы и т.д.)
- Трансляция видеопотоков в реальном времени (RTSP, RTP, Websockets)
- Экспорт данных во внешние системы в реальном времени
- Получение данных из внешних систем, датчиков и т.д.

Этот компонент взаимодействует с компонентом edge-api и должен видеть его по сети. Адрес подключения к edge-api настраивается с помощью переменных окружения IVA_EDGE_SERVER, IVA_EDGE_SERVER_GRPC, IVA_EDGE_SERVER_WS в команде docker run.

Пример содержимого каталога компонента видеоаналитики:

/msdata/walkers-node-1

/logs/ - Каталог для файловых логов компонента

/temp/ - Каталог для временных файлов компонента

logs.sh – Скрипт для просмотра консольных логов компонента

_start.sh – Скрипт для запуска компонента (без пересоздания контейнера)

_stop.sh – Скрипт для остановки контейнера (без уничтожения контейнера)

config-patch.json – Файл с патчем значений дефолтной внутренней конфигурации компонента (управляется вендором)

_create-run.sh – Скрипт для запуска компонента (первый запуск, либо после пересоздания контейнера)

pull.sh – Скрипт для обновления образа контейнера с компонентом с репозитория вендора

remove.sh – Скрипт для остановки и уничтожения контейнера

shell.sh – Скрипт для входа в консоль контейнера

update-restart.sh – Скрипт для обновления и перезапуска контейнера

Пример содержимого скрипта запуска контейнера, с использованием Linq TPU:

```
#!/bin/bash
export IVA_CONFIG=$(cat config-patch.json)
docker run --restart=always -d --name walkers-node-1 \
--link edge-api-1:edge-api-1 \
-e TZ=Europe/Moscow \
--hostname walkers-node-1 \
-e IVA_CONFIG="${IVA_CONFIG}" \
-e IVA_ENABLE_INSTALLER=0 \
-e IVA_TENANT="cbr" \
-e IVA_EDGE_SERVER="edge-api-1:16000" \
-e IVA_EDGE_SERVER_WS="ws://edge-api-1:9001/datahub" \
-v /etc/localtime:/etc/localtime \
-v /msdata/shared:/app/msdata/shared \
-v /msdata/shared:/app/msdata/data \
--device /dev/tpu0:/dev/tpu0 \
-v $PWD/temp:/app/msdata/temp \
-v $PWD/logs:/app/msdata/logs \
ivacv.tech:15444/common/walkers-node:0.1.3
```

Пример содержимого скрипта запуска контейнера, с использованием GPU Nvidia:

```
#!/bin/bash
export IVA_CONFIG=$(cat config-patch.json)
docker run --restart=always -d --name ivacv-service-node-1 \
--gpus all -e NVIDIA_VISIBLE_DEVICES=all -e NVIDIA_DRIVER_CAPABILITIES=compute,utility,video \
-e TZ=Europe/Moscow \
--hostname $(hostname)-docker \
-e IVA_CONFIG="${IVA_CONFIG}" \
-v /etc/localtime:/etc/localtime \
-v $PWD/data:/app/msdata/data \
-v $PWD/shared:/app/msdata/shared \
-v $PWD/distr:/app/msdata/distr \
-v $PWD/temp:/app/msdata/temp \
-v $PWD/config:/app/msdata/config \
-v $PWD/funcs:/app/msdata/funcs \
-v $PWD/models:/app/msdata/models \
-v $PWD/logs:/app/msdata/logs \
ivacv.tech:15444/radar-mms/ivacv-service-node-cuda:0.7.8
```

7. EDGE-API

Этот компонент предназначен для получения данных различного вида от компонентов видеоаналитики и сохранения этих данных в основную БД системы.

Компонент доступен по API различного вида (REST, WebSockets, gRPC). Адрес подключения к edge-api со стороны компонентов видеоаналитики настраивается с помощью переменных окружения IVA_EDGE_SERVER, IVA_EDGE_SERVER_GRPC, IVA_EDGE_SERVER_WS.

Этот компонент взаимодействует с основной БД Mongodb и должен видеть её по сети. Адрес подключения к Mongodb настраивается в параметрах файлов конфигурации компонента.

Пример содержимого каталога компонента Edge-api:

/msdata/edge-api-1

/data/

/config/ – Каталог с файлами конфигурации

/temp/ – Каталог с временными файлами

/logs/ – Каталог с файловыми логами

/scripts/

logs.sh – Скрипт для просмотра консольных логов компонента

run.sh – Скрипт для запуска компонента

shell.sh – Скрипт для входа в консоль контейнера

stop.sh – Скрипт для остановки компонента

pull.sh – Скрипт для обновления образа контейнера компонента с репозитория вендора

Пример содержимого скрипта запуска контейнера:

```
#!/bin/bash

docker run --restart=always --name edge-api-1 \
--link mongodb-1:mongodb-1 \
-p 16000:16000 \
-p 9001:9001 \
-e TZ=Europe/Moscow \
-v /msdata/edge-api-1/data:/usr/src/app/msdata \
-v /msdata/shared:/usr/src/app/msdata/shared \
-v /etc/localtime:/etc/localtime \
-d ivacv.tech:15444/common/edge-api:0.3.10
```

В этом примере:

- Каталог с данными БД монтируется в контейнер из каталога хоста: /msdata/edge-1/data/db
- Общий каталог с данными монтируется в контейнер из каталога /msdata/shared

Параметры в файлах конфигурации edge-api

Файлы конфигурации компонента расположены в подкаталоге /data/config. Представляют собой файлы в формате .json5, с различными параметрами конфигурации компонента.

log.json5

logMode – Режим логирования, может быть равен "console" или "file"

logLevel - Уровень логирования: "trace", "debug", "info", "warn", "error"

printCounters – Распечатывать в лог значения счетчиков производительности

printCountersSec - Отображение значения счетчиков производительности в секундах (вместо миллисекунд)

main.json5

serviceName – Наименование сервиса для логирования, например "edge-api"

instanceName - Наименование сервиса для логирования, например "edge-api-1"

grpcServerBind – Адрес биндинга и порт для gRPC API, например "0.0.0.0:50052"

healthCheckTcpPort – Порт для docker health check, по умолчанию 8901

wsPort – UDP порт для WebSockets API, например 9001

dbUrl – Url в стандартном формате Mongodb для подключения к основной БД

clearTempOnStart – Очищать временный каталог при старте сервиса

defaultTenantCode – Имя тенанта по умолчанию для API, для сохранения сущностей в БД без указания тенанта

restApi.json5

restApiEnabled - Включение или отключение REST API

allowOriginHeader – Ответ для CORS запросов, например: "*",

apiTcpPort – Порт для REST API, например 16000,

apiBindIp – Адрес биндинга для REST API, "0.0.0.0",

apiMethods – Массив с описанием методов REST API и их параметров

apps - Массив со списком uuid разрешенных приложений (api keys)

appsAccess – Перечень методов API, доступных для каждого из приложений

badUrls – Перечень url, которые сразу блокируются в API (для типовых url, которые перебирают роботы)

8. VDMS-API

Этот компонент предназначен для выполнения следующих функций:

- Предоставление API для пользовательского интерфейса
- Построение индексов в БД для ежедневных коллекций
- Архивированиие и удаление старых данных
- Трансляция видеопотоков для пользовательского интерфейса

Компонент доступен для клиентских и сторонних приложений по API различного вида (REST, WebSockets, gRPC).

Этот компонент взаимодействует с основной БД Mongodb и должен видеть её по сети. Адрес подключения к Mongodb настраивается в параметрах файлов конфигурации компонента. Также, этот компонент использует экземпляр компонента видеоаналитики, вызывая его по gRPC. Адрес подключения также настраивается в параметрах файлов конфигурации компонента.

Пример содержимого каталога компонента Vdms-api:

/msdata/vdms-api-1

/addons/ - Каталог с бинарными подключаемыми модулями

/data/

/config/ – Каталог с файлами конфигурации

/temp/ – Каталог с временными файлами

/logs/ – Каталог с файловыми логами

/scripts/

logs.sh – Скрипт для просмотра консольных логов компонента

run.sh – Скрипт для запуска компонента

shell.sh – Скрипт для входа в консоль контейнера

stop.sh – Скрипт для остановки компонента

pull.sh – Скрипт для обновления образа контейнера компонента с репозитория вендора

Пример содержимого скрипта запуска контейнера:

```
#!/bin/bash

docker run --restart=always --name vdms-api-1 \
--link mongodb-1:mongodb-1 \
--link walkers-node-1:walkers-node-1 \
-p 7777:7777 \
-p 17777:17777 \
-e TZ=Europe/Moscow \
-v /msdata/vdms-api-1/data:/usr/src/app/msdata \
-v /msdata/shared:/usr/src/app/msdata/data \
-v /msdata/vdms-api-1/addons:/usr/src/app/addons \
-v /etc/localtime:/etc/localtime \
-d ivacv.tech:15444/common/vdms-api:0.5.22
```

В этом примере:

- Каталог с данными БД монтируется в контейнер из каталога хоста: /msdata/vdms-1/data
- Общий каталог с данными монтируется в контейнер из каталога /msdata/shared
- Каталог с бинарными модулями монтируется в контейнер из каталога хоста: /msdata/vdms-1/addons

Параметры в файлах конфигурации vdms-api

Файлы конфигурации компонента расположены в подкаталоге /data/config. Представляют собой файлы в формате .json5, с различными параметрами конфигурации компонента.

log.json5

logMode – Режим логирования, может быть равен "console" или "file"

logLevel - Уровень логирования: "trace", "debug", "info", "warn", "error"

printCounters – Распечатывать в лог значения счетчиков производительности

printCountersSec - Отображение значения счетчиков производительности в секундах (вместо миллисекунд)

---

main.json5

---

serviceName – Наименование сервиса для логирования, например "vdms-api"

instanceName - Наименование сервиса для логирования, например "vdms-api-1"

defaultUserLocale – Язык сервера для возврата текстов ошибок в API, например "ru"

dbUrl – Url в стандартном формате Mongodb для подключения к основной БД

streamMonitorJobIntervalMs: Интервал запуска джоба мониторинга реалтайм медиапотоков, который включает или отключает муксеры на сервере для реалтайм-трансляции видео.

streamMonitorCamerasQuery: Выборка (в формате mongodb) для джоба мониторинга реалтайм медиапотоков, для выборки камер для мониторинга.

grpcApiUrl: Адрес gRPC API компонента видеоаналитики, например "walkers-node-1:16100".

grpcApiServiceFace: Название сервиса для запросов в gRPC API компонента видеоаналитики, который используется при добавлении лиц в БД лиц, например "service-common-face-api".

grpcApiCamCode: Код камеры для запросов в gRPC API компонента видеоаналитики, который используется при добавлении лиц в БД лиц, например "vdms-api".

grpcApiServiceFaceTimeoutSec: Таймаут в секундах для запросов в gRPC API компонента видеоаналитики, который используется при добавлении лиц в БД лиц, например 5.

defaultTenantCode: Код тенанта для запросов в gRPC API компонента видеоаналитики, который используется при добавлении лиц в БД лиц, например "cbr".

defaultVisibleEventTypes: Список кодов событий для отображения событий в пользовательском интерфейсе. Может использоваться для того, чтобы спрятать от всех пользователей определенные виды событий видеоаналитики. Например: ['detect-tailgate', 'car-gate-open', 'number-plate-not-found', 'detect-person', 'identify-person', 'identify-unknown', 'lock-open-in', 'lock-open-out'],

refFacesJobIntervalMs: Интервал запуска джоба обновления БД лиц в памяти из БД, например 5000.

unidentifiedFacesDbCacheDays: Размер кэша неидентифицированных лиц в памяти (для поиска по неидентифицированным лицам), например 1.

loadUnidentifiedFaces: Загрузка неидентифицированных лиц в БД лиц в памяти, например false.

eventsProcessorJobEnabled: Включение или отключение джоба обработки событий видеоаналитики, например true.

eventsProcessorJobIntervalMs: Интервал запуска джоба обработки событий видеоаналитики в миллисекундах, например 500.

eventsProcessorStuckRestartSec: Интервал в секундах, для перезапуска обработки записи в очереди событий видеоаналитики (для корректной обработки событий в очереди, которые обрабатывались в момент аварийной остановки компонента), например 10.

dataCleanupJobEnabled: Включение или отключение джоба удаления старых данных, например true.

dataCleanupJobIntervalHours: Интервал в часах запуска джоба очистки старых данных, например 0.5.

dataCleanupKeepDays: Количество дней хранения старых данных, например 3.

dataCleanupStreamsKeepHours: Длительность хранения текущих данных видеопотоков, в часах 0.5.

dataArchiverJobEnabled: Включение или отключение джоба архивации медиаданных.

dataArchiverJobIntervalMs: Интервал в миллисекундах запуска джоба архивации медиаданных, например 1000,

indexMonitorJobEnabled: Включение или отключение джоба создания индексов в БД Mongodb.

indexMonitorJobIntervalMs: Интервал в миллисекундах запуска джоба создания индексов, например 3600000

***Indexes**: Объекты с описанием индексов в БД Mongo

---

restApi.json5

---

restApiEnabled - Включение или отключение REST API

allowOriginHeader – Ответ для CORS запросов, например: "*",

apiTcpPort – Порт для REST API, например 7777,

apiBindIp – Адрес биндинга для REST API, "0.0.0.0",

apiMethods – Массив с описанием методов REST API и их параметров

apps - Массив со списком uuid разрешенных приложений (api keys)

appsAccess – Перечень методов API, доступных для каждого из приложений

badUrls – Перечень url, которые сразу блокируются в API (для типовых url, которые перебирают роботы)

wsApi.json5

wsApiEnabled - Включение или отключение WebServices API

apiTcpPort – Порт для WS API, например 17777,

apiBindIp – Адрес биндинга для WS API, "0.0.0.0",

topics, messages – Объекты с описанием топиков и сообщений WS API

apps - Массив со списком uuid разрешенных приложений (api keys)

appsAccess – Перечень методов API, доступных для каждого из приложений

9. VDMS-UI

Этот компонент представляет собой веб-приложение с пользовательским интерфейсом системы. Это набор статичных .html, .css и .js файлов, которые публикуются с помощью встроенного в контейнер nginx сервера.

Этот компонент взаимодействует с компонентом vdms-api и должен видеть его по сети. Адрес подключения к vdms-api настраивается в конфигурации nginx.

Пример содержимого каталога компонента Vdms-api:

/msdata/vdms-ui

/data/

/config/ – Каталог с файлами конфигурации nginx

/scripts/

logs.sh – Скрипт для просмотра консольных логов компонента

run.sh – Скрипт для запуска компонента

shell.sh – Скрипт для входа в консоль контейнера

stop.sh – Скрипт для остановки компонента

pull.sh – Скрипт для обновления образа контейнера компонента с репозитория вендора

Пример содержимого скрипта запуска контейнера:

```
#!/bin/bash

docker run --restart=always --name vdms-ui \
--link vdms-api-1:vdms-api-1 \
-p 8080:80 \
-e TZ=Europe/Moscow \
-v /etc/localtime:/etc/localtime \
-v /msdata/vdms-ui/data/config/nginx:/etc/nginx \
-d ivacv.tech:15444/common/vdms-ui:0.3.2
```

10. МОНИТОРИНГ И ДИАГНОСТИКА

Для мониторинга состояния системы и диагностики проблем можно использовать следующие инструменты и возможности:

Просмотр логов

У каждого компонента системы есть скрипт logs.sh, при запуске которого пользователь может посмотреть последние логи работы контейнера из консоли Docker.

Логи с ошибками отображаются красным цветом, отладочные зеленым, информационные синим.

Логи с ошибками не должны игнорироваться пользователями, то есть, не может быть «нормальных» ошибок, любое сообщение красным цветом в логах – требует незамедлительного анализа и исправления.

Уровень логирования управляется из файлов конфигурации (для api компонентов) или из БД системы (для компонентов видеоаналитики).

Консоль контейнера

У каждого компонента системы есть скрипт shell.sh, при запуске которого пользователь может войти внутрь работающего контейнера Docker. В каждом контейнере установлен набор базовых инструментов диагностики (типа ping), а также Midnight Commander.

Файлы конфигурации компонента видеоаналитики доступны внутри контейнера, в каталоге /app/msdata/config.

Встроенные средства восстановления Docker

Для всех компонентов системы используется встроенная в Docker возможность Health check, которая перезапускает компоненты в случае сбоев.

Средства мониторинга аппаратного обеспечения

Для мониторинга состояния аппаратного обеспечения (GPU и пр.) можно использовать множество различных инструментов, специфичных для использования оборудования, таких как nvidia-smi, nvitop, intel_gpu_top

11. ПЕРЕЗАГРУЗКА КОМПОНЕНТОВ

У каждого компонента есть набор скриптов для остановки и запуска контейнера (start/run и stop/remove).

Каждый компонент может быть остановлен и перезапущен отдельно от остальных, в перезагрузке всех компонентов сразу обычно нет никакой необходимости.

По умолчанию, порядок запуска компонентов следующий:

1. Mongodb
2. Edge-api
3. Сервис видеоаналитики
4. Vdms-api
5. Vdms-ui

12. ОБНОВЛЕНИЕ КОМПОНЕНТОВ

У каждого компонента есть скрипт для обновления образа с репозитория вендора (pull.sh). Для обновления версии контейнера, необходимо указать в этом скрипте, а также в скрипте run.sh новую версию компонента.

После обновления компонента, необходимо перезапустить компонент с пересозданием контейнера.

13. АРХИВАЦИЯ И УДАЛЕНИЕ ДАННЫХ

За архивацию и удаление данных (видеопотоков, событий, кадров и т.д.) отвечает компонент vdms-api. Параметры архивации управляются из файлов конфигурации этого компонента.

Архив видео расположен в общем каталоге /msdata/shared/archive

14. ИНСТАЛЛЯЦИЯ КОМПОНЕНТОВ

Из-за сложности и разнообразия задач и используемой инфраструктуры, платформа видеоаналитики не является программным обеспечением, которое может работать из коробки.

Это скорее платформа конструирования и разработки решений, работающих с медиапотоками и выполняющих самые разнообразные задачи, которые могут варьироваться от видеоаналитики на борту дрона или внутри шлагбаума, до использования в серверных кластерах в больших облачных платформах.

Также, из-за работы платформы с нейронными сетями, если попытаться собрать в один дистрибутив все используемые системой файлы с весами моделей, со всеми их вариантами, то общий объем дистрибутива получится не менее 500 гигабайт.

В связи с этим, общего типового инсталлятора компонентов системы нет. Для каждого проекта (который всегда уникален), вендором делается специальная сборка решения (из специфичного для задачи набора приложений, файлов конфигурации, файлов моделей и т.д.) и выкладывается в репозиторий, специальный для указанного заказчика и проекта. В этом смысле, подход к распространению и развертыванию системы похож на, например, биллинговые системы, АБС, системы платежных процессингов и т.д.

Тем не менее, есть набор определенных шагов, которые выполняются при инсталляции компонентов системы (в формате чеклиста).

1. Установка операционной системы и проверка ее работоспособности, версии ядра и т.д.
2. Установка Docker
3. Создание пользователя, от имени которого компоненты будут работать внутри контейнеров и в операционной системе хоста
4. Создание общих каталогов и установка прав доступа для них
5. Установка Mongodb (если используется на этом узле)
6. Импорт дефолтной БД Mongodb (если используется на этом узле)
7. Настройка данных и справочников БД Mongodb (если используется на этом узле)
8. Установка и конфигурирование компонента edge-api (если используется на этом узле), путем запуска скрипта инсталляции, копирования на хост файлов конфигурации и настройки этих файлов конфигурации, а также скриптов компонента.
9. Установка и конфигурирование компонента видеоаналитики (если используется на этом узле), путем запуска скрипта инсталляции и настройки скриптов компонента.
10. Установка и конфигурирование компонента vdms-api (если используется на этом узле), путем запуска скрипта инсталляции, копирования на хост файлов конфигурации и настройки этих файлов конфигурации, а также скриптов компонента.
11. Установка и конфигурирование компонента vdms-ui (если используется на этом узле), путем запуска скрипта инсталляции и настройки скриптов компонента.
12. Запуск компонентов и проверка работоспособности пользовательского интерфейса

13. Тестовый запуск и проверка работоспособности видеоаналитики (с тестовыми камерами, видеофайлами, вызовами API и т.д.)