Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа киберфизических систем и управления

Работа допущена к защите
Руководитель ОП
 В.В. Потехин
09 июня 2020 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Разработка облачных распределенных систем управления и интеграция в IIOT платформу

по направлению подготовки
27.04.04 Управление в технических системах

Направленность (профиль)
27.04.04_07 Распределенные интеллектуальные системы управления

Выполнил
студент группы 3542704/80701  А.П. Алексеев

Руководитель
доцент ВШ КФСУ,  В.В. Потехин
к.т.н., доцент

Консультант
по нормоконтролю  Е.Н. Селиванова

Санкт-Петербург
2020

Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа киберфизических систем и управления

УТВЕРЖДАЮ
Руководитель ОП
 В.В. Потехин
28 января 2020 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

студенту группы 3542704/80701 Алексееву Антон Павловичу

1. Тема работы:

Разработка облачных распределенных систем управления и интеграция в IIoT платформу

2. Срок сдачи студентом законченной работы: 05.06.2020 г.
3. Исходные данные по работе:

Техническое задание на разработку

4. Содержание работы (перечень подлежащих разработке вопросов):

Определение проблем традиционной архитектуры АСУ ТП
Анализ сервисно-ориентированной архитектуры
Обзор существующих решений IIOT платформ и облачных РСУ
Разработка прототипа облачной РСУ и интеграция в IIOT платформу

5. Перечень графического материала (с указанием обязательных чертежей):

Схемы стенда сервера облачной платформы

6. Консультанты по работе:
7. Дата выдачи задания: 28.01.2020 г.

Руководитель ВКР  В.В. Потехин

Задание принял к исполнению 28.01.2020 г.
Студент  А.П. Алексеев

РЕФЕРАТ

На 65 с. , 25 рис. , 3 табл. , 2 прил. , 20 ист.

ЦИФРОВОЙ ДВОЙНИК, ИНДУСТРИЯ 4.0, СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА, ИНТЕРНЕТ ВЕЩЕЙ, АВТОМАТИЗАЦИЯ, РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ

Данная работа посвящена построению распределенных систем управления с применением облачных распределенных инфраструктур. В работе рассматривается традиционная архитектура САУ, ее проблемы, архитектура SOA в АСУ ТП, миграция традиционной архитектуры в SOA. Приведен обзор на современные IaaS платформы, решения облачных РСУ, требования к программно-аппаратной части разрабатываемой облачной платформы. В заключительной части работы описан процесс разработки прототипа облачной РСУ, сервисов автоматизированной сборки РСУ, аппаратной части проекта.

ABSTRACT

65 p. , 25 pic. , 3 tabl. , 2 app. , 20 ref.

DIGITAL TWIN, INDUSTRY 4.0, SOA, INTERNET OF THINGS, AUTOMATION, DCS

This work is devoted to the construction of distributed control systems using cloud distributed infrastructures. The paper considers the traditional architecture of self-propelled guns, its problems, the architecture of SOA in process control systems, the migration of traditional architecture in SOA. The review on modern IaaS platforms, cloud DCS solutions, requirements for the software and hardware of the developed cloud platform is given. The final part of the work describes the process of developing a prototype cloud DCS, services for automated assembly of DCS, and the hardware of the project.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Направление виртуальных АСУТП и SCADA – мировой инновационный тренд, разработки ведут ряд крупных вендоров систем АСУТП и несколько мировых промышленных концернов. Параллельно с этим под эгидой международных организаций разрабатываются необходимые инновационные технические решения/ спецификации/ стандарты. Сложность перехода от современной системы управления процессами к облачным РСУ и Web Scada состоит в том, чтобы сделать это структурированным образом, постепенно модернизировать высоко интегрированные и привязанные к поставщику стандарты в более открытую структуру, сохраняя при этом функциональность. Появилась необходимость изучить возможность эффективной реализации функций распределенных систем управления и функций операторов управления непрерывными процессами в облачной инфраструктуре и интеграции с архитектурой IIoT [1].

В ходе исследования были выявлены следующие проблемы современных распределенных систем управления:

- Отсутствие гибкости (часто необходимость замены аппаратной части) РСУ при модернизации и внесении изменений;
- При тиражировании ПО на контроллеры нового поколения необходимо полностью или частично переписывать и отлаживать код.
- Отсутствие универсальных решений по непрерывной разработки-тестированию-интеграции различных РСУ.

Также рассмотрены способы перехода из традиционной структуры АСУ ТП в архитектуру, удовлетворяющей следующим принципам Индустрии 4.0:

1. Совместимость – все устройства и машины должны уметь общаться друг с другом на одном языке посредством интернета вещей, т.е. они должны быть совместимы.
2. Прозрачность – создание цифровой копии продукта, сбор данных с микрочипов и датчиков посредством которых устройства общаются.
3. Техническая поддержка – программное обеспечение производит сбор, анализ, систематизацию, визуализацию данных, полученных с датчиков, и помогает человеку принимать решение или принимает их в автоматическом режиме, тем самым высвобождая человеческие ресурсы.
4. Децентрализация управленческих решений, автоматизация различных решений системами, максимально полное замещение человека.

В данной работе представлен подход применения облачных распределенных систем управления непрерывными процессами в облачной инфраструктуре и интеграции с архитектурой IIoT. Приведен оптимальный подход для построения облачной РСУ. Описан процесс разработки прототипа облачной РСУ, а также процесс тестирования прототипа по разработанной программе и методике испытаний.

В главе 1 описана традиционная архитектура САУ, ее проблемы, архитектура SOA в АСУ ТП, миграция традиционной архитектуры в SOA.

В главе 2 приведен обзор на современные IaaS платформы, решения облачных РСУ, требования к программно-аппаратной части разрабатываемой облачной платформы.

В главе 3 описан процесс разработки прототипа облачной РСУ, сервисов автоматизированной сборки РСУ, аппаратной части проекта.

1 Обзор современной архитектуры промышленных систем

1.1 Уровни автоматизированной системы управления технологическим процессом

На текущий момент современные системы автоматического управления можно представить в виде многоуровневой структуры. Опишем каждый уровень в направлении снизу-вверх:

1. Фундамент такой системы представляет собой периферийное оборудование, по своей сути исполнительные устройства, устройства управления, датчики (аналоговые и дискретные) и т.д. Без датчиков наша система не сможет нормально функционировать, так как основная информация поступает от них, можно сказать, что это зрение нашей системы.
2. Следующий уровень отвечает за исполнительные механизмы нашей системы, если проводить аналогию, то это опорно-двигательный механизм. Данный уровень управляет приводами технологического оборудования.
3. Завершает часть системы, которая отвечает за управление технологическим процессом уровень программируемых логических контроллеров. Все программы, в которых заложены инструкции для нижестоящего уровня управления приводами технологического оборудования на основе данных с уровня периферийных устройств, находятся и исполняются в контроллерах. Эти устройства работают в системах реального времени
4. За диспетчерское управление и организацию сбора данных отвечает SCADA уровень. Это программное обеспечение, которое предназначено для разработки систем сбора данных, их обработки и отображения на человеко-машинном интерфейсе.

Данный уровень является одной из интеграционных частей многоуровневой системы.

5. Завершает интеграционную часть многоуровневой системы MES уровень. Данное специализированное прикладное программное обеспечение предназначено для координирования, анализа и оптимизации выпускаемого продукта в рамках определенного производства.
6. Верхнюю часть такой системы занимает ERP уровень. Это корпоративная информационная система, которая позволяет оптимизировать процессы автоматизации учета, планирования, анализа и контроля всех бизнес-процессов в масштабе предприятия.

Рисунок 1 - Уровни САУ

Разбиение САУ на уровни таким образом облегчает задачу построения сетевой архитектуры, так как нет необходимости проектировать сетевую систему для всех составляющих САУ как одно целое. Проектирование происходит послойно, для каждого уровня отдельно, что позволяет упростить процесс отладки [2].

Фундаментом архитектуры является уровень, который соединяет системы управления с процессом с помощью датчиков и исполнительных механизмов. Через считывание сигналов датчиков система управления может получать информацию о состоянии объекта управления, а затем регулировать его состояние через актуаторы. Традиционные решения основываются на цикличном опрашивании устройств ввода/вывода, считывающий и записывающий данные с датчиков, передающий сигналы на актуаторы.

На втором уровне (диспетчерский) организовано взаимодействие оператора с контроллерами с помощью человеко-машинным интерфейсом, наиболее популярным вариантом является SCADA-пакеты. Оператор имеет возможность наблюдать за всеми процессами, которые заложены в станцию, на экране монитора компьютера или планшета, способен взаимодействовать с этими процессами с помощью кнопок или сенсорной панели. Синхронизация событий от пользователя и работой станции осуществляется с помощью SCADA-системы. Сигналы из компьютера/планшета передаются по протоколу TCP/IP, через Ethernet.

Третий уровень интегрирует систему АСУ ТП с автоматизированной системой управления предприятием. АСУП также может включать в себя четвертый уровень, который обеспечивает интеграцию с высшим руководством, расположенным на различных точках земного шара. На данном уровне обеспечивает работу TCP/IP и Ethernet.

На уровне ERP, как правило, происходит планирование ресурсов предприятия. Системы (ERP) устанавливаются для стратегического планирования всей работы завода в соответствии с бизнес-целями [3].

Традиционная структура АСУ ТП по сей день остается наиболее распространённой архитектурой, используемой на современных производствах. В связи с тенденцией к переходу промышленных систем к концепциям Индустрии 4.0 данная структура показала ряд недостатков описанных в следующем разделе.

1.2 Недостатки традиционной архитектуры САУ

Рассмотрим следующие проблемы традиционных САУ:

1. При расширении системы (добавления новых автоматизированных участков) необходимо провести полную установку шкафов управления, перенести программный код в контроллер вручную, подключить к подсети контроллер, модули ввода/вывода, преобразователи, частотники и тд. Это занимает достаточно большое количество времени, в связи с отсутствием гибкости и масштабируемости при дублировании систем управления. При развертывании станции, которая уже существует на производстве необходимо проводить те же операции, что и при развертывании такой станции в первый раз.
2. Для долговременной поддержки автоматизированной станции необходимо заранее определять, какой контроллер подойдет для данной задачи, выбирать с учетом возможности расширения системы, так как отсутствует возможность динамично добавлять вычислительные ресурсы в систему.
3. Поддержка разных протоколов передачи данных осуществляется с помощью дополнительных OPC серверов, установка которых нагромождает систему, достаточно сложна в развертывании.

Традиционная архитектура АСУ ТП не позволяет решить данные проблемы, не меняя подход к проектированию таких систем. При этом переход к архитектуре, удовлетворяющей концепциям Индустрии 4.0, должен выполнятся постепенно, безболезненно для непрерывных производств [4]. Прежде, чем обозначить преимущества SOA архитектуры и необходимые этапы миграции, необходимо рассмотреть функциональные и технические требования к одному из самых важных компонент традиционной структуры АСУ ТП – распределенных систему управления.

1.3 Функциональные и технические требования к РСУ

В данном разделе описаны стандартные особенности традиционных распределенных систем управления, их функциональные и технические требования, а также приведены примеры использования данного функционала в сервисно-ориентированной архитектуре.

Режим реального времени.

В традиционных системах управления используется цикличное опрашивание устройств ввода/вывода, которые регулирует определенную часть установки, с относительно небольшим количеством исполнительных механизмов и датчиков. Согласно функциональным требованием, время опроса датчика составляет примерно 80мс, при этом в сложных системах может доходить и до 10мс. Необходимо обеспечить низкую задержку сигнала и достаточную пропускную способность.

Основным преимуществом применения SOA является гибкий набор сервисов и приложений для диагностики и контроля, легко интегрируемые в системы SCADA. При использовании локального корпоративного облака задержка сигнала не будет превышать допустимые нормы.

Управление с помощью супервизоров.

Управление большим потоком информации на более высоком уровне намного медленнее, чем циклическое опрашивание устройств в реальном времени. Супервизорное управление использует технологии MES систем, чтобы посылать уставки для ПЛК, так как не имеет прямой доступ к датчикам и актуаторам.

В подходе SOA в локальной корпоративной сети все сервисы связаны друг с другом напрямую, в виде графа. В этом отличие от существующей иерархической структуры, в которой данные устройства ПЛК циклично опрашивает сенсоры, которые затем подают в. Видимость устройств затем улучшается без дополнительной нагрузки.

Распределенное управление.

Одной из особенности таких систем управления является распределенное управление, где основным признаком которых является удаленное друг от друга географически, что означает, что управление не может быть выполнено одним устройством (контроллером) с прямым доступом как к датчикам, так и к актуаторам.

Сервисно-ориентированная архитектура поддерживает распределение программ на несколько систем, виртуальных машин, сервисов или устройств. Конфигурации, содержащие логику управления, хранятся в центральном хранилище, поэтому обмен устройствами возможен без ручного перепрограммирования.

Агрегация системы.

Низкоуровневые устройства часто представляются в агрегированной форме системам более высокого уровня, чтобы предоставить информацию операторам, инженерам и другим лицам, работающим с системой.

Управление в сервисно-ориентированной архитектуре может быть реализовано инкапсулированным, но скоординированным способом. Это больше относится к пакетным приложениям, чем к непрерывным производственным процессам

Аварии и предупреждения.

Одной из важных задач РСУ является индикация аварий и предупреждений во время непрерывного процесса производства. Для РСУ заготавливается базовый набор функций, связанных с аварийными сигналами и предупреждениями, которые позволяют распространять информацию соответствующему персоналу на уровни выше, а также несколько способов подавления и подтверждения тревог и предупреждений.

Одно из преимуществ сервисно-ориентированного подхода состоит в том, что он будет получать только необходимые сигналы тревоги и предупреждения. Аварийные сигналы будут отфильтрованы в зависимости от пользователя, вошедшего в систему, предоставляя информацию,

необходимую для действий пользователя. Например, оператор будет проинформирован о том, что процесс остановлен без каких-либо дополнительных подробностей, в то время как группа технического обслуживания получит подробную информацию о неисправности машины.

Промышленные протоколы связи.

На данный момент разные вендоры предлагают не унифицированные способы взаимодействия между РСУ, используя промышленные протоколы связи. Необходимо подстраиваться под определенный протокол, имплементировать в РСУ поддержку многочисленных промышленных протоколов связи.

В сервисно-ориентированной архитектуре IoT-шлюзы используются для конвертации протоколов, передачи сырых данных по защищенному каналу связи в облако, сопоставления данных с внутренней моделью данных компонентов интеграции.

Отображение, сбор и хранение данных.

Для хранения данных используются сторонние приложение, трудно разворачиваемые в производственной среде, требующие драйверы и интерфейсы взаимодействия. Доступность данных необходима как для операторов, так и для руководства, чтобы оптимизировать производительность и анализировать исторические данные.

С точки зрения сервисно-ориентированной архитектуры необходимо изменить или перейти с традиционных систем такого типа на интеллектуальные граничные устройства, способные как получать необходимые данные, так и инкапсулировать их в веб-сервисы.

Аварийный останов.

Аварийный останов является важной частью большинства систем управления. Просто отключение питания всех компонентов не является решение данной задачи, поскольку это может вызвать ситуации, когда повышение температуры, давления или химическая реакция способны вызвать взрывоопасные и непредвиденные ситуации.

Сервисно-ориентированная инфраструктура позволяет гибко управлять несколькими стратегиями выключения в зависимости от различных аварийных условий, а также оптимизировать эти стратегии на протяжении всего жизненного цикла оборудования.

Аутентификация и безопасность.

Так как существует большое количество пользователей с разными ролями, которые работают с РСУ, важно, чтобы каждому человеку был предоставлен уровень информации, достаточный и соответствующий его уровню допуска. Для минимизации количества человеческих ошибок, а также злонамеренных действий, необходимо, чтобы весь персонал проходил аутентификацию на роль, в которой ему разрешен доступ к системе. Аутентификация не всегда может быть ограничена программным обеспечением, но может вместо этого включать ограничение физического доступа к определенным областям или станциям.

Структура безопасности SOA архитектуры должна распространяться на все предприятие, а прозрачность устройств в облачной инфраструктуре облегчает представление всей системы.

Ручное управление оператором.

Оператор может управлять системой вручную через HMI панель, для обработки нештатных ситуаций или управления в ручном режиме. Ручное управление необходимо для выполнения операций технического обслуживания, когда системы отключаются контролируемым образом.

SOA обеспечивает прямую связь между верхним уровнем и устройствами, так что такая важная информация легко доступна. Ручное управление может осуществляется через Web-Scada, напрямую через браузер мобильных устройств[5].

1.4 Сервисно-ориентированная архитектура управления производством.

Сервис-ориентированная инфраструктура АСУ включает в себя технологии, сочетающие облачные вычисления и веб-сервисы. Главная особенность данной архитектуры состоит в том, чтобы эффективно разрабатывать инструменты и методы, позволяющие достичь гибкой, реконфигурируемой, масштабируемой, интероперабельной сетевой структуры между децентрализованными и распределенными киберфизическими системами.

Первым шагом к созданию такой инфраструктуры является создание сервис-ориентированной экосистемы. То есть сетевые системы, состоящие из интеллектуальных встроенных устройств, взаимодействуя с как физической, так и облачной средой, способная динамично управлять сервисами, преследуя четко определенные системные цели. Данная система представлена на рисунке ниже.

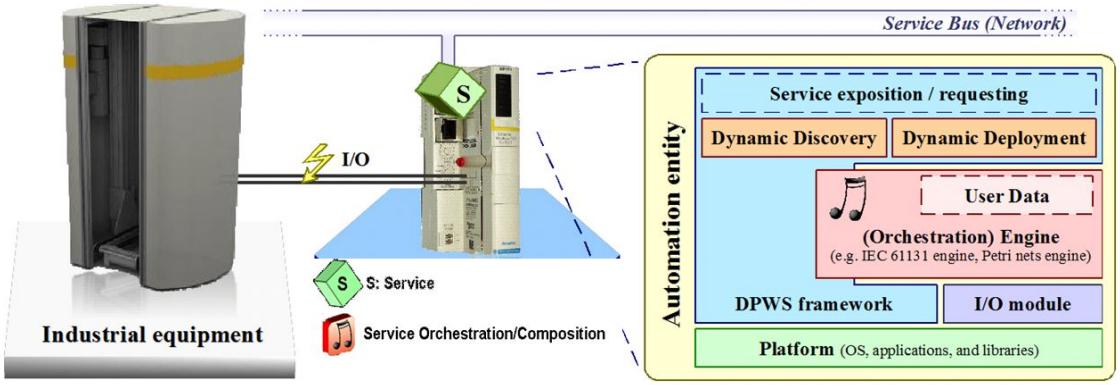

Рисунок 2 - Сетевые системы в SOA

Использование сервис-ориентированной архитектуры способствует созданию открытой, гибкой и масштабируемой среды в предприятии, начиная с самых низких уровней иерархии устройств и заканчивая системой управления бизнес-процессами более высокого уровня производственного предприятия.

Устройства и системы, расположенные на разных уровнях, должны иметь один и тот же интерфейс веб-службы и взаимодействовать между собой. Это функциональное взаимодействие полностью не зависит от физического положения в традиционной архитектуре АСУ ТП [6].

При переходе на облачную архитектуру АСУ ТП проблемы традиционной архитектуры решаются следующим образом:

1. Для расширения системы необходимо установить только модули ввода/вывода к каждой станции, подключить их к общей подсети. Среда исполнения контроллера находится в виртуальном контейнере, который можно продублировать с уже заложенной в нее программой.

2. При постепенном наращивании плотности программы в контроллере можно динамично добавлять больше оперативной памяти или ядер в виртуальный контейнер.

3. Для поддержки разнообразных сетевых протоколов используется IoT шлюз, способный конвертировать один протокол в другой.

Для решения данной проблемы необходимо переработать архитектуру системы автоматизированного управления. Часть системы, которая отвечает в большей степени за оптимизацию и анализ производственных процессов, хранение больших данных должна быть вынесена в облачное пространство, при этом все сегменты этого облака могут обмениваться друг с другом информацией, необходимо уйти от иерархической системы к графам, где каждый узел имеет связь с другим узлом. Для соединения с облачным пространством – IoT платформой - необходимо устройство, которое общается с уровнем управления и сбором данных. Для этого можно использовать специальные граничные контроллеры – Edge controllers – которые могут по открытым протоколом передавать данные от нижнего уровня в облако, а так же производить небольшие оптимизационные задачи, пересылать команды из облака в управляющий уровень. Схема такой архитектуры приведена ниже.

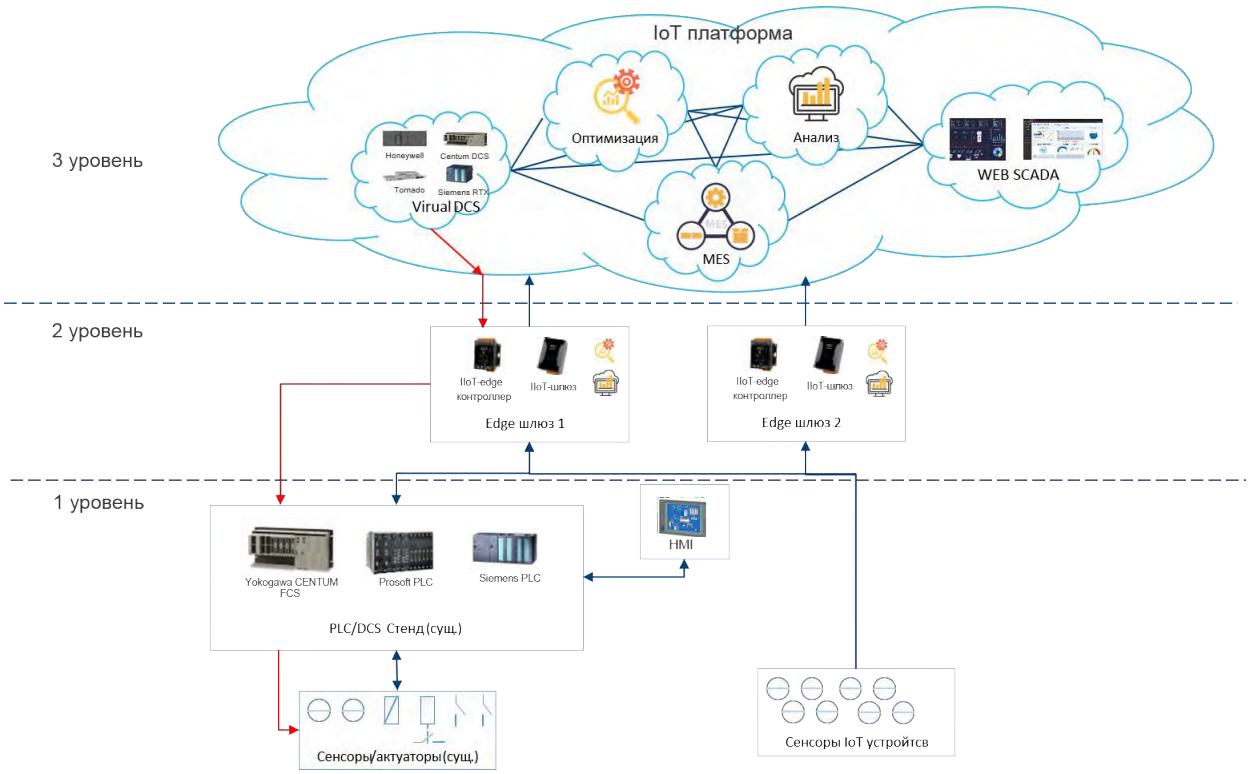

Рисунок 3 - SOA концепция в САУ

Данная схема не заменяет существующие АСУ ТП в производстве, а лишь дополняет ее. Упор сделан на масштабирование новых производственных участков и поддержка уже существующих. Необходимо двигаться в направлении сосуществования современной архитектуры АСУ ТП и облачной для постепенной миграции.

Используя данную концепцию можно разработать систему, которая позволяет организовать управление через облако совместно с существующей стандартной схемой САУ, добавляя возможность тиражировать готовые производственные решения, динамично распределять вычислительные ресурсы между контроллерами [7].

На данный момент автоматизация производства делится на иерархические уровни, с датчиков собирается информация на контроллеры, с контроллеров все процессы выводятся на Scada систему, далее верхние уровни отвечают за управление технологиями и планирование ресурсов. Необходимо понять, какие функции каждого уровня мы можем виртуализировать и вынести в облачное пространство, а какие жестко должны быть привязаны к реальному исполняемому железу. Так как

виртуальные контроллеры базируются на операционных системах, следовательно время исполняемого цикла меньше, чем у ПЛК, и при этом имеют удаленный доступ к комплекту терминального оборудования ввода/вывода, что значит большее время передачи сигнала, мы не можем вынести все управление в облако, полностью отказавшись от нижних уровней. Но, например, трудно вычисляемые задачи, такие как оптимизация и анализ данных, управление вспомогательными IoT устройствами, моделирование процессов и прогноз будущего состояния станций, формирование отчетов, распознавание и сигнализацию аварийных ситуаций мы можем исполнять в облаке.

Больше всего нас интересует гибкость и масштабируемость новой архитектуры. Насколько можно быстро внедрять новые системы, которые уже созданы, просто продублировав их и каким образом можно эффективно распределять вычислительные ресурсы между виртуальными контейнерами.

1.5 Миграция современных РСУ/SCADA систем в SOA

Современные системы автоматизации предприятия состоят из нескольких уровней, которые рассматриваются и взаимодействуют иерархически. С расширением возможностей, предоставляемых современными сервис-ориентированными архитектурами, функциональные возможности каждой системы или даже устройства могут быть использованы как один или несколько сервисов различной сложности, которые могут быть размещены в облаке и составлены другими сервисами. Промышленные приложения следующего поколения теперь можно быстро разворачивать, выбирая и комбинируя между собой разнообразные сервисы в для реализации своих целей. Предполагаемый переход к будущим облачным промышленным системам показан на рисунке ниже.

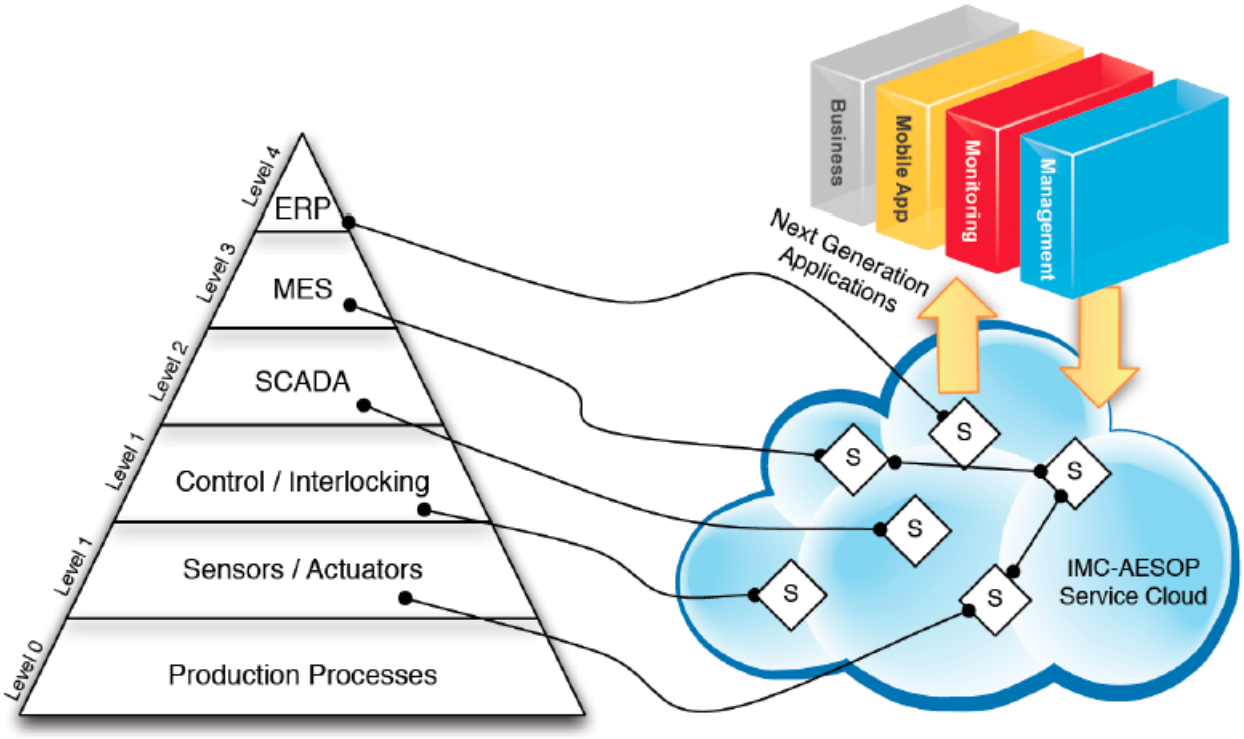

Рисунок 4 - Эволюция промышленной архитектуры в Индустрии 4.0

Как правило, современные производственные системы структурированы иерархически, как уже было сказано ранее. МЭК 62264 дополнительно определяет модель управления производственными процессами. Стандарт определяет функции, связанные с диспетчерским и MES уровнем, объекты, их характеристики и атрибуты, действия и функции, связанные с управлением предприятием, но ничего не говорит ни о решениях, в которых размещается конкретная операция, ни о точных присвоение одному из уровней сенсоров, диспетчерского и MES. Реализация зависит от индивидуальных потребностей заказчика и вариантов реализации поставщика решения. Композиция облака нацелена на соответствие требованиям МЭК 62264, сохраняя организационные аспекты, установленные в современных производственных системах.

При изучении данного вопроса необходимо описать требования к новой инфраструктуре. В статье «Migration of SCADA/DCS systems to the SOA cloud» описаны определенные требования, которых следует придерживаться при переходе от стандартных АСУ в облачные [8]:

- После миграции завод должен по-прежнему обеспечивать такие же или более эффективные показатели, продлить срок службы оборудования производства (например, технологическое оборудование, насосы, сосуды, клапаны).
- Динамические изменения и реорганизация должны поддерживаться на постоянной работающей системе.
- Новая архитектура и стратегия миграции должны обеспечивать такой же уровень надежности и доступности как стандартная система.
- Чтобы справиться с сосуществованием между стандартной системой и облачной на этапе миграции, решение облачной РСУ и Web-Scada должно поддерживать перенос устаревших подсистем.
- Процедура миграции не должна вызывать повышенный риск для персонала, оборудования и надежности процесса.
- При переносе физических компонентов в облачные сервисы необходимо учитывать требования по промышленной и информационной безопасности предприятия

Ниже можно увидеть схему традиционных распределенных систем управления.

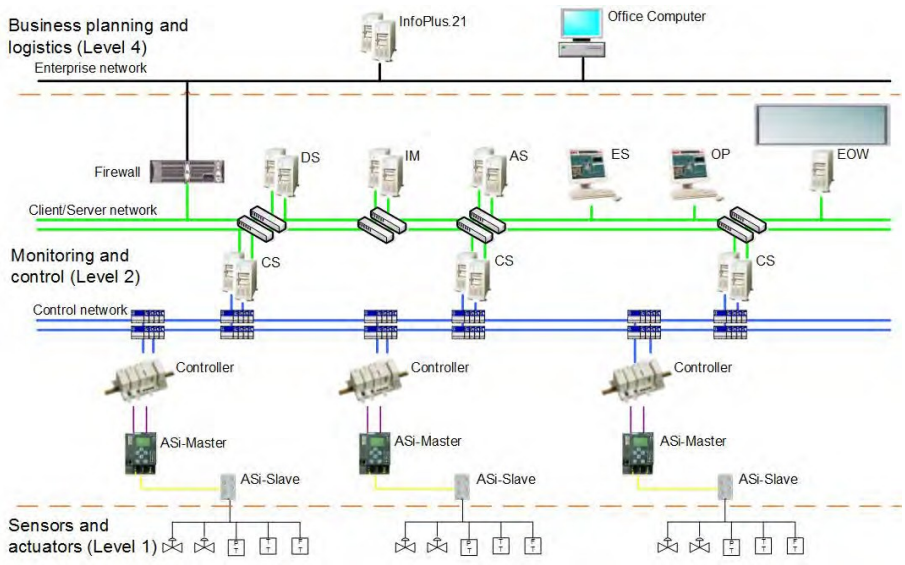

Рисунок 5 - Традиционная схема РСУ

Для миграции высокоинтегрированной РСУ необходимо решить следующие проблемы:

- **Необходимо сохранить функциональную интеграцию.** Высокоинтегрированная РСУ обеспечивает тесную связь между HMI и ПЛК. Таким образом, проектированиеи и ввод в эксплуатацию могут осуществляться универсально. Например, HMI и ПЛК могут быть настроены с помощью одних и тех же инструментов, что обеспечивает универсальность.
- **Распределение устройств**. В рамках данной системы необходимо определить, какие устройства следует перенести на SOA как физические устройства, а какие устройства следует интегрировать в облако Например, подсистема, использующая обратную связь и регулирование, требующая традиционные интерфейсы и работающая в режиме реального времени, должна взаимодействовать с облаком через посредника - IoT шлюза.
- **Исполнение программ в режиме реального времени.** Выполнение программ в реальном времени, которое в прежней системе используется в контроллерах, должно быть сохранено.

Для первого этапа миграции в SOA необходимо реализовать базовые сервисы для поддержки базовой связи и управления облаком. Как только базовая архитектура построена, первые периферийные подсистемы могут быть перенесены в облако и новые компоненты могут быть интегрированы в SOA. При миграции подсистем, а также при интеграции новых компонентов необходимо учитывать ограничения IoT шлюза. Схема РСУ после первого шага миграции представлена ниже.

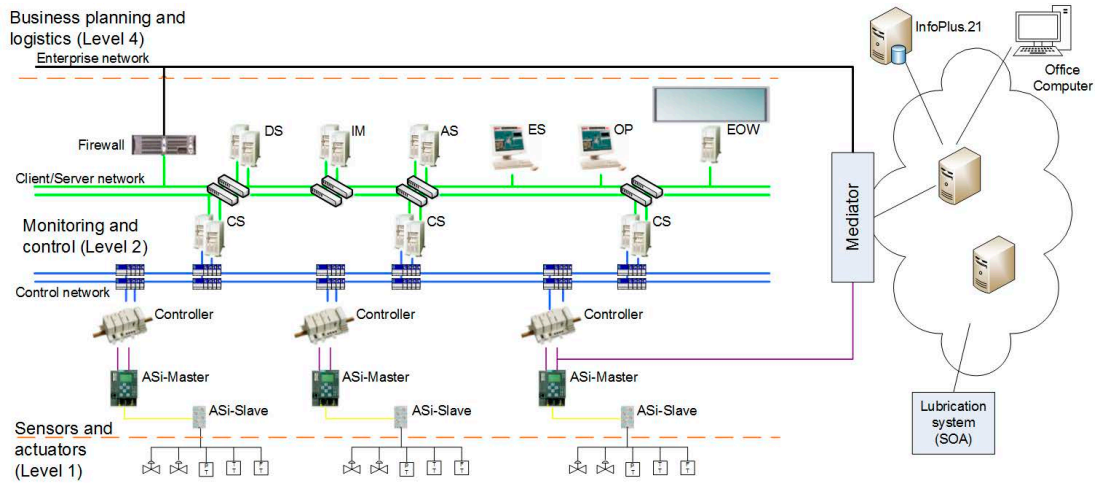

Рисунок 6 - РСУ после первого шага миграции

Цель следующего этапа - перенести распределенные системы управления, которые не требовательны к исполнению программы в режиме реального времени. Одними из первых компонент, которые можно интегрировать в облако являются инженерные станции (ES), которые используются для проектирования и конфигурации большинства частей DCS, как показано на рисунке ниже.

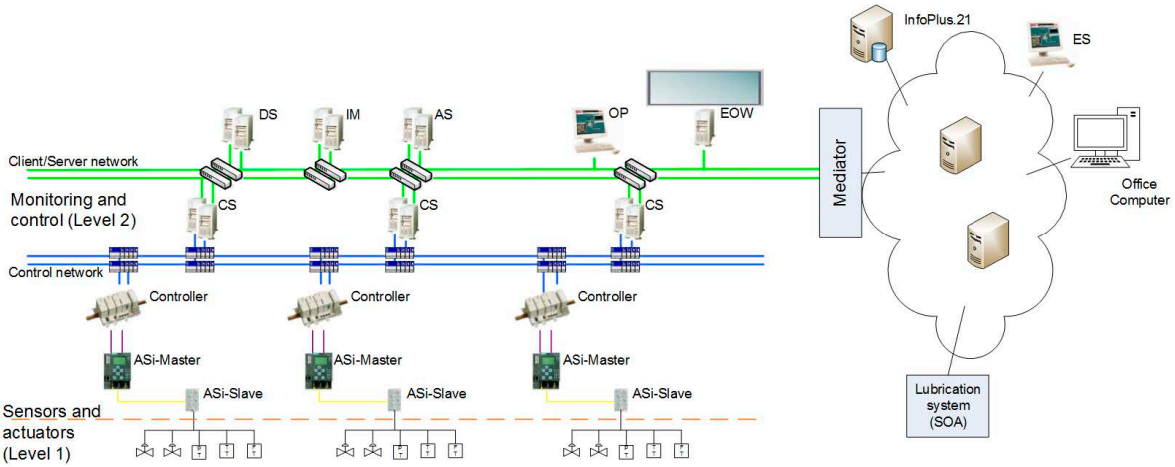

Рисунок 7 - РСУ после второго шага миграции

На третьем этапе миграция включает все компоненты, которым не требуется быстрое считывание сигналов (миллисекундный диапазон), которое в настоящее время не поддерживается технологией SOA. Данные компонентамы отображены на рисунке ниже:

- клиенты оператора OP
- инженерные станции EOW

- серверы управления AS
- серверы управления информацией IM.

Так как все компоненты взаимодействия пользователя с системой теперь перенесены на облако, то устаревшие сервера DS оказываются не используемыми.

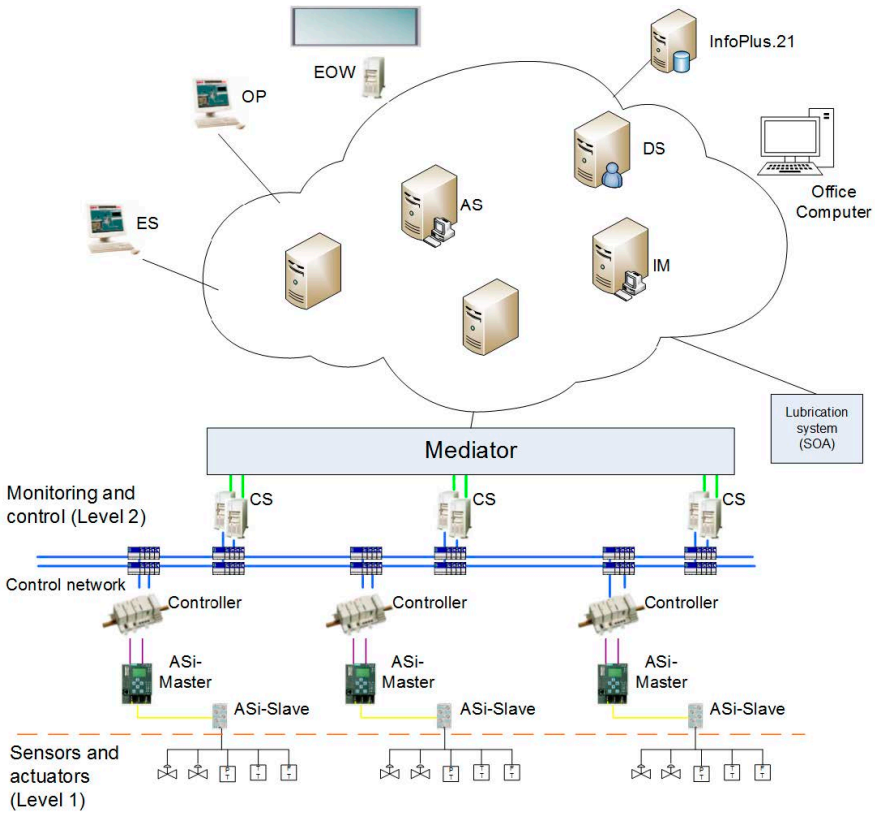

Рисунок 8 - РСУ после третьего этапа миграции

На последнем этапе миграции происходит перенос управляющих функций распределенных систем управления. Управление актуаторами, считывание сигналов с проводных/беспроводных датчиков происходит напрямую развёрнутыми сервисами – виртуальными контроллерами. Текущую задачу следует рассматривать как частичную миграцию, замену не требуемого к жесткой надежности и точности оборудования. Ниже приводится схема инфраструктуры после прохождения всех шагов миграции.

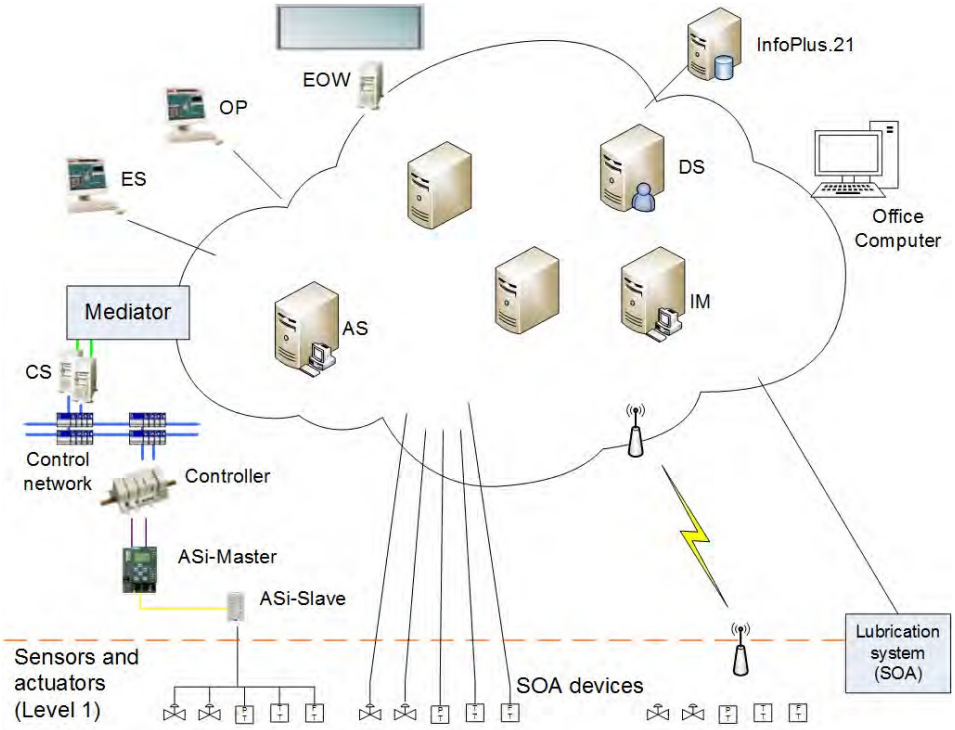

Рисунок 9 - Финальный этап миграции РСУ

Представленная концепция миграции полностью не меняет структурную иерархию системы управления, а лишь дополняет и интегрируется к ней, позволяет ей функционально вести себя как высокораспределенная плоская архитектура, основанная на службах.

2 Обзор существующих подходов к проектированию SOA систем

2.1 Обзор современных IAAS платформ

В данном разделе описаны современные IAAS платформы для построения облаков. Также выбрана платформа для дальнейшей разработки SOA.

Cloudstack

Cloudstack — это программное обеспечение с открытым исходным кодом, которое позволяет создавать, управлять, распределять сервисы облачной инфраструктуры. CloudStack обладает вычислительным функционалом, который выделяет виртуальные машины (ВМ) для отдельных серверов, а также сетевым функционалом, который управляет коммутаторами для создания и управления логическими сетями, системами хранения объектов и блоков, функцию управления изображениями и интерфейс управления облачными вычислениями, который поддерживает все компоненты программного стека. CloudStack позволяет развертывать сети виртуальных машин и управлять ими, используя следующие гипервизоры:

- Oracle VM server
- KVM
- Citrix XenServer
- VMware
- Xen Cloud Platform (XCP)
- Microsoft Hyper-V

Пользователи могут управлять своими облачными развертываниями через интерфейс командной строки (CLI), RESTful API или веб-интерфейс. CloudStack также предоставляет интерфейс прикладных программ (API),

который совместим с Amazon EC2 и S3 для упрощения развертывания гибридных облаков.

Eucalyptus Open Source

Eucalyptus — платная платформа для построения частных облаков фирмы Amazon Web Services.

Функционал Eucalyptus:

Управление гибридным облаком - запуск инстансов, моментальное создание снимков и управление автоматически масштабируемыми группами в частных или общедоступных облаках из единой среды. Мощный и простой в использовании интерфейс самообслуживания, который управляет ресурсами Eucalyptus Cloud, может управлять облачными ресурсами AWS.

Совместимость с AWS - Eucalyptus обеспечивает ведущую в отрасли совместимость с популярными API-интерфейсами Amazon Web Services (AWS), включая EC2, S3, Elastic Block Store (EBS), управление идентификацией и доступом (IAM), автоматическое масштабирование, эластичное распределение нагрузки (ELB) и CloudWatch. ,

Eucalyptus позволяет использовать стандартные серверные технологии, системы хранения, сети и технологии виртуализации для предоставления экономичных, совместимых с AWS облачных сервисов в центре обработки данных. Eucalyptus совместим с AWS EC2 и позволяет легко развертывать вычислительные ресурсы и эффективно увеличивать или уменьшать вычислительную мощность в зависимости от требований приложений.

Openstack

Openstack – это облачная open-source платформа, которая управляет большими пулами вычислительных ресурсов, хранилищ и сетевых ресурсов в центре обработки данных, и все они управляются с помощью панели мониторинга, которая дает администраторам возможность управления, предоставляя своим пользователям возможность предоставлять ресурсы через веб-интерфейс.

Преимуществами OpenStack являются:

- Наибольшее сообщество пользователей среди открытых проектов.
- API: соответствует стандартам индустрии, доступ на основе аутентификации с ограничениями максимальной интенсивности запросов.
- Широкие возможности по конфигурации для конкретных нужд и требований.
- Поддержка Amazon EC2 и S3 API упрощает миграцию и увеличивает совместимость с другими облачными решениями.
- Распределенная и асинхронная архитектура позволяет реализовать системы с неограниченным горизонтальным масштабированием и высокой надёжностью.
- Ролевая политика избирательного управления доступом.
- Возможность изоляции проектов различных групп пользователей с квотированием ресурсов.

Благодаря использованию облачной платформы OpenStack и технологии виртуализации сети (SDN, Software defined networking) сегмент вычислений может быть разделен на проекты, каждый из которых имеет собственную IP сеть, изолированную от других сетей. Благодаря этому возможно обеспечить изоляцию проектов друг от друга, что повышает защищенность стенда при одновременной работе нескольких групп пользователей.

Таким образом исходя из перечисленных преимуществ IaaS платформ выше, был сделан выбор в пользу Openstack, так как он является open-source продуктом, есть доступ к API, интегрируется с Linux системами.

2.2 Обзор существующих решений облачной РСУ

В данной работе были рассмотрены существующие решения виртуальной РСУ различных вендоров.

Siemens

SIMATIC WinAC (Windows Automation Center) – это программное обеспечение реализации функций S7-совместимых программируемых контроллеров в среде операционной системы Windows. Контроллеры SIMATIC WinAC существенно расширяют спектр возможных применений систем автоматизации производства SIEMENS.

Характеристики SIMATIC WinAC [3]:

- Функционирование на базе промышленных или офисных компьютеров, встраиваемых модульных контроллеров SIMATIC S7-mEC, а также многофункциональных панелей операторов SIMATIC MP x77.
- Применение дополнительного программного обеспечения WinAC ODK для включения кодов C/C++ в программы контроллеров WinAC.
- Полная программная совместимость с программируемыми контроллерами S7-300/ S7-400. Программирование, конфигурирование и диагностика из среды STEP 7.
- Параллельное функционирование с другими приложениями Windows. Использование единой аппаратной платформы для решения задач автоматического
- Поддержка систем распределенного ввода-вывода на основе сетей PROFIBUS DP и PROFINET IO.
- Наличие расширения RTX (Real Time Extension), обеспечивающего возможность функционирования

программируемых контроллеров SIMATIC WinAC в реальном масштабе времени.

- Поддержка функций противоаварийной защиты и обеспечения безопасности в контроллерах WinAC RTX F.

Интеграция WIN AC RTX с другими технологиями приведена ниже.

Рисунок 10 - Функционал Siemens RTX

Имеет существенный недостаток - рекомендуется использование преимущественно станций Siemens (SIMATIC S7-mEC, SIMATIC MP x77), связь с подсистемами в/в SIMATIC ET200 по PROFIBUS-DP, что подразумевает локально развёртываемую архитектуру системы, и ставит под сомнение возможность реальной централизации АСУТП и отнесения виртуальных контроллеров «далеко» в облако [10].

Omron

Компания Omron имеет пакет программ для разработки ПО для контроллеров CX-ONE, в который входит CX-ONE Simulator – виртуальный PLC, симулирующий работу реального PLC серии CS/CJ. Схема виртуального контроллера представлена ниже.

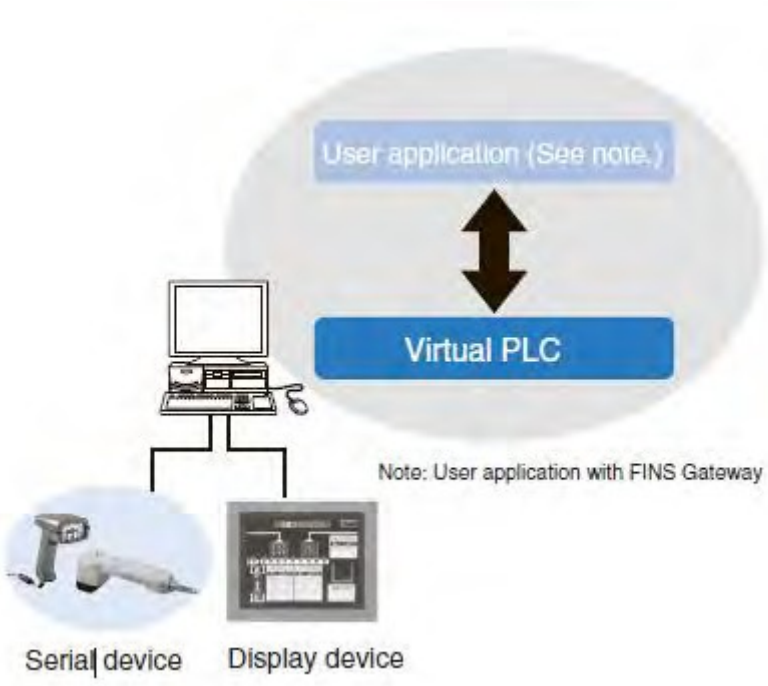

Рисунок 11 - Взаимодействие виртуального контроллера с устройствами ввода/вывода

Характеристики, взятые с официального сайта:

- Можно использовать все функции отладки, предусмотренные в CX- Programmer
- Длительность цикла может быть определена без включения реальной системы ПЛК
- Поддержка эффективных приемов отладки, таких как пошаговое выполнение, выполнение одиночных циклов и включение точек останова, которые были бы невозможны при работе с настоящим ПЛК
- Предусмотрено несколько способов имитации поступления входных сигналов
- Моделирование взаимной работы ПЛК и панели оператора без реального оборудования.

Моделирование системных ошибок, не прибегая для этого к использованию специальных команд в программе ПЛК [11].

ИнСАТ Master-PLC 4D

MasterPLC - это набор исполнительных модулей MasterSCADA для программирования контроллеров с открытой архитектурой (SoftLogic), базирующихся на операционных системах DOS, miniOS7, Linux, Ecos, Windows CE, Windows.

Коммуникационные возможности

- OPC DA, HDA (в зависимости от ОС), UA (клиент и сервер)
- Modbus RTU и TCP
- МЭК 61850
- SNMP
- MS SQL, PostgreSQL
- Открытый универсальный API
- Универсальный драйвер для разработки новых протоколов

Интегрированная среда разработки

- Разработка в Windows, исполнение: Windows, Linux, QNX, Android, Эльбрус
- Неограниченное масштабируемое поле редактора схем с миникартой для навигации
- Объектно-ориентированный подход к разработке - типизация и тиражирование элементов проекта, механизм клеммников (инкапсуляция)
- Возможность операций с любым свойством элемента проекта, включая визуальные
- Удаленная и локальная отладка с окном наблюдения в среде разработки
- Выбор тем для оформления графических окон во всем проекте
- Программирование пользовательских алгоритмов
- Полноценная поддержка языков стандарта МЭК 61131-3

- Возможность использования языков стандарта МЭК-61131-3 для разработки на всех уровнях проекта, включая графические клиенты
- Библиотека стандартов алгоритмов (математические и логические операции) [12]

CODESYS

CODESYS инструментальный программный комплекс промышленной автоматизации. Содержит систему исполнения (Control Runtime System). Она устанавливается в контроллер в процессе его изготовления. Существует специальный инструмент (Software development kit), позволяющий адаптировать её к различным аппаратным и программным платформам.

CODESYS Runtime основан на компонентно-ориентированной архитектуре, то есть представляет собой набор компонентов — модулей, на которые разделена каждая логическая или функциональная часть CODESYS Runtime.

Каждый компонент ответственен за свою задачу и область действия — например, отвечает за логирование, за сетевое взаимодействие, за взаимодействие по серийному кабелю, за распределенную нагрузку на ядрах, за отладку программы и т.д.

Среди всех компонентов CODESYS Runtime можно выделить основные:

- Component Manager, или CM — компонент для запуска и инициализации всех остальных компонентов в системе;
- System Components — группа компонентов для описания взаимодействий с операционной системой и с железной составляющей. Компоненты из этой группы отвечают за взаимодействие с физическими портами, с файловой системой, с работой по динамическому и статическому выделению памяти и т.д.

- Communication Components – группа компонентов для взаимодействия с внешним миром, например по сети или через последовательный кабель;
- Application components — компоненты управления программой ПЛК;
- Core components — компоненты для управления ПЛК и его состоянием [13].

Ниже компонентно-ориентированная архитектура CodeSys Runtime.

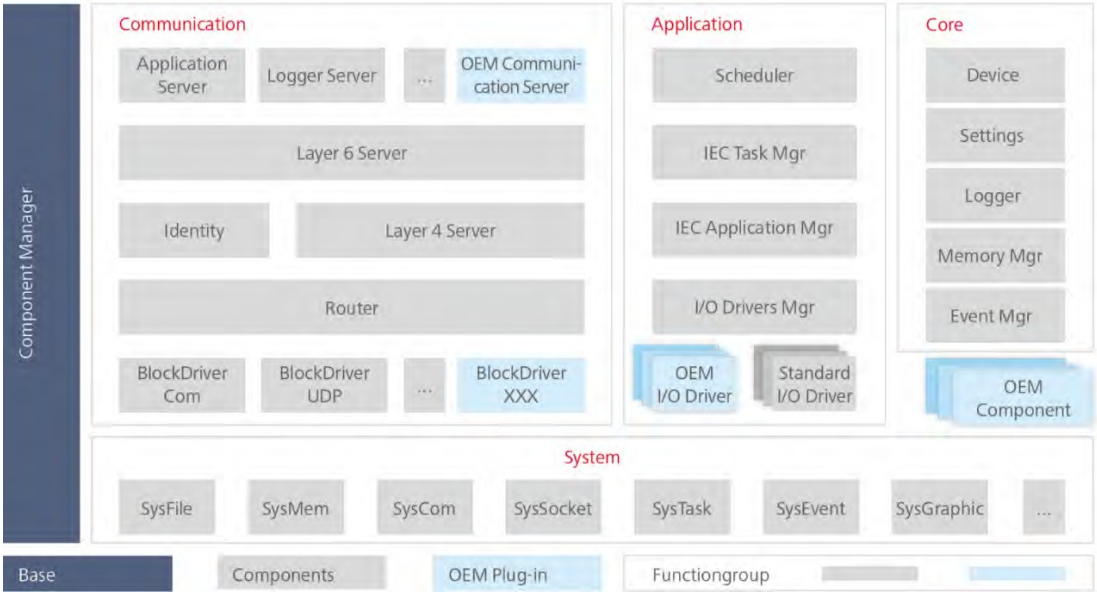

Рисунок 12 - Архитектура Codesys runtime

OpenPLC

OpenPLC - программируемый логический контроллер с открытым исходным кодом. Данный проект был создан в соответствии со стандартом IEC 61131-3, который определяет базовую архитектуру программного обеспечения и языки программирования для ПЛК. OpenPLC предназначен для промышленной и домашней автоматизации, интернета вещей.

Цель opensource проекта - обеспечить доступность PLC при низкой цене для автоматизации промышленности и исследованиях. Проект OpenPLC состоит из трех частей: runtime, среда разработки и HMI редактор. Среда выполнения может быть установлена на Windows, Linux или Raspberry Pi и отвечает за выполнение программы PLC. Среда разработки — это

программное обеспечение, которое используется для создания программы ПЛК. ScadaBR это HMI редактор, с помощью которого можно создавать Web-Scada. OpenPLC поддерживает протокол Modbus TCP, может быть как Master, так и Slave. [14]

Выводы

Таким образом, исходя из того, что предлагают крупные и отечественные вендоры, все кандидаты имеют поддержку необходимого стандарта IEC61499, необходимые сетевые протоколы, а также взаимодействия с Scada и IO. Одними из минусов программного обеспечения от крупных вендоров являются отсутствие поддержки кроссплатформенности, высокая цена, взаимодействие с человеко-машинным интерфейсом происходит через собственные разработанные Scada системы, что ограничивает гибкость системы.

- Omron представляет из себя в большей степени симмулятор ПЛК, чем виртуальный контроллер
- Siemens предлагают решения, которые можно интерпретировать, как полноценные ПЛК, несмотря на то, что решение предлагается не новое.
- Master PLC отвечает всем требованиям к виртуальному контроллеру, может базироваться на Linux ОС, образ включает в себя контроллер и Scada систему одновременно.
- Codesys предлагает использовать ядро для имитации контроллера, при этом распространяя среду программирования бесплатно. Он является PC- based linux системой, тем самым хорошо интегрируется в облачную инфраструктуру.
- Решения с открытым исходным кодом можно внедрять в IoT системы. Тем не менее не поддерживается большинство протоколов и сложных программ, но при этом такие системы, как OpenPLC, являются наипростейшими для разворачивания в облаке.

2.3 Обзор существующих решений Web-Scada

SIMATIC WinCC Open Architecture

Данный продукт является SCADA системой для визуализации и оперативного управления процессами. По информации, приведенной на официальном сайте, распределенная конфигурация позволяет подключать до 2048 автономных систем SIMATIC WinCC Open Architecture в одной сети. Каждая подсистема может быть представлена одно- или многопользовательсской конфигурацией с обычной или резервированной структурой.

SIMATIC WinCC Open Architecture опирается на объектно-ориентированный подход к формированию экранов отображения процесса и поддерживаемой структуры базы данных.

Модульная архитектура, масштабируемость, кросс-платформенность.

Система WinCC OA построена по модульному принципу и функционально разделена на несколько процессов, которые могут быть распределены по различным серверам в сети. Обмен данными между менеджерами осуществляется по событиям с использованием протокола TCP/IP. WinCC OA является кросс-платформенной системой – поддерживаются операционные системы Windows, Linux (Red Hat, OpenSUSE, CentOS), а также платформа виртуализации VMware ESXi. Возможно применение различных операционных систем на серверах и клиентах.

Сбор данных, коммуникации и интеграция со сторонними системами.

Платформа WinCC OA обеспечивает поддержку большинства распространенных протоколов обмена данными с устройствами и системами различных типов:

- протоколов семейства OPC – OPC UA (DA, AC – Client & Server, HA – Client), OPC DA/AE/HDA (Client & Server);
- протоколов на основе TCP/IP – SIMATIC S7, Modbus, Ethernet/IP, SNMP Manager & Agent, BACnet и др.;
- протоколов систем телемеханики и энергетики: IEC 60870-5-101, IEC 60870-5-104, IEC 61850/61400, DNP3, SINAUT и др.

Аналитический инструментарий.

В составе WinCC OA имеется инструментарий для статистической и аналитической обработки данных, позволяющий извлекать из общего потока данных значимую информацию, необходимую для поддержки оператора в процессе принятия решений, – подсистема SmartSCADA , которая показана на рисунке ниже.

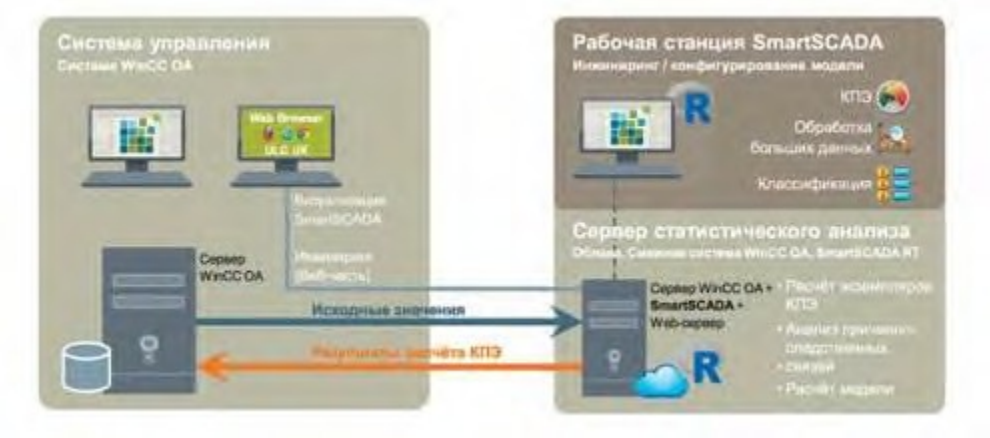

Рисунок 13 - Подсистема SmartScada

Wonderware InTouch

Wonderware System Platform 2017

Wonderware System Platform 2017 предназначена для взаимодействия с многочисленными управляющими устройствами и производственными системами. Эта системная платформа позволяет дополнительно интегрировать широкий спектр промышленного программного обеспечения — в том числе Wonderware Skelta BPM, Wonderware MES Operations, Wonderware Intelligence, Wonderware IntelaTrac, Wonderware Dream Report, Wonderware InBatch, Wonderware Recipe Manager Plus и Wonderware Online. Ниже приведен пример визуализации Scada с разнообразных устройств.

InTouch Access Anywhere

InTouch Access Anywhere позволяет дистанционным мобильным пользователям HMI и SCADA использовать веб-браузер для просмотра данных и управления производством в режиме реального времени.

Основные функции, описанные на сайте производителя

- Работает в обычном браузере
- Поддержка HTTP и HTTPS
- На стороне клиента не нужно ничего устанавливать
- Работает во всех поддерживающих HTML5 браузерах (IE, Safari, Chrome, Opera и т.д.), на различных программных платформах (Windows, iOS, Android, Linux,…)
- Безопасный доступ
- Полный доступ к приложениям InTouch в любое время в любом месте независимо от программной и аппаратной платформы и месторасположения
- Та же безопасность, что и у RDP

Scada Rapid

Rapid SCADA — это бесплатная, полнофункциональная SCADA-система с открытым исходным кодом.

С помощью Rapid SCADA можно создать автоматизированные системы следующих типов:

- Системы управления технологическими процессами (АСУ ТП).
- Системы «умный дом».
- Системы учёта энергоресурсов (АСКУЭ, АСТУЭ, АИИС КУЭ).
- Системы охранно-пожарной сигнализации (ОПС).
- Системы контроля доступа (СКУД).
- Любые системы, содержащие контроллеры, датчики и реле.

Rapid SCADA включает в себя драйвера, которые обеспечивают обмен данными с широким спектром устройств, используя распространённые стандарты связи, а также специализированные протоколы обмена данными от производителей оборудования. Новые драйвера могут быть реализованы как разработчиками Rapid SCADA, так и сторонними разработчиками. Архитектура программного комплекса позволяет создавать дополнительные модули, которые выполняют обмен данными с внешними системами и базами данных.

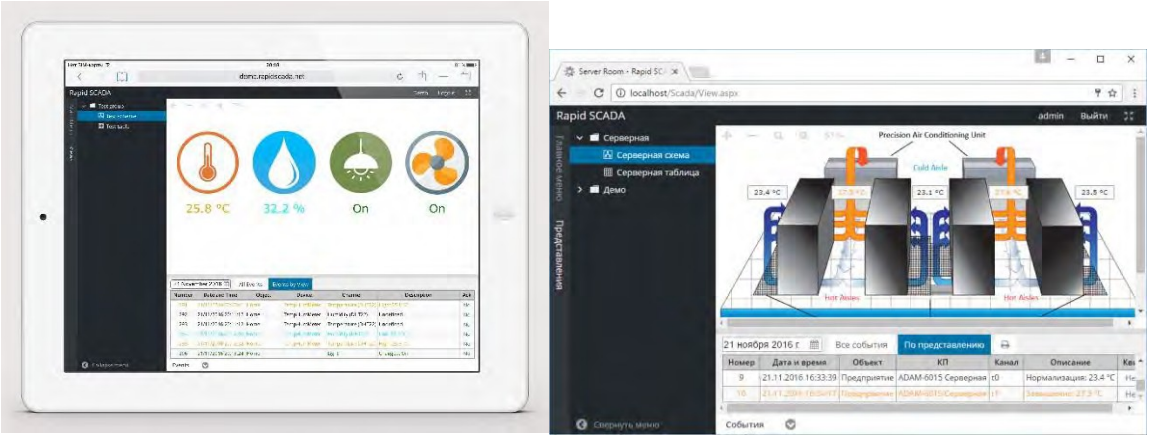

Рисунок 14 - Интерфейс Rapid Scada

Поддерживаемые СУБД

Rapid SCADA позволяет экспортировать данные в реальном времени в базы данных Oracle, Microsoft SQL Server, PostgreSQL и MySQL, используя родные драйвера, а также в любые другие базы данных, поддерживающие набор интерфейсов OLE DB.

Облачный мониторинг

Облачный мониторинг на базе Rapid SCADA позволяет создать надёжную автоматизированную систему, которая будет обслуживаться разработчиками Rapid SCADA. Сервера Rapid SCADA располагаются в дата-центрах лучших российских облачных провайдеров, а также используется Google Cloud Platform и Microsoft Azure. Контроллеры монтируются в любом месте, где есть подключение к сети Интернет, и обмениваются данными с облачным сервером по проводным и беспроводным каналам связи.

Выводы

Таким образом, рассмотрев решения от крупных вендоров, можно заметить, что Web Scada поставляется не самостоятельным приложением, а является частью созданной платформы, что может доставить трудности в проведении испытаний и интеграции с виртуальными контроллерами.

- В то же время можно рассмотреть созданные платформы полностью, например WinCC OA в рамках тестирования WinCC RXT. Siemens предлагает качественное решение, одним из главных плюсов которого является резервирование, которые необходимы по требованиям к Scada системам.
- Wonderware создала целую операционную среду, которую можно рассмотреть в качестве IoT-платформы, а также Web интерфейс с интеграцией InTouch.
- Из отечественных продуктов можно выделить Master Scada, которая отвечает всем требованиям к Web Scada, легко интегрируется вместе с Master PLC, может базироваться на Linux ОС, поставляется с демо-версией, в связи с тем рекомендуется к апробации.

Из решений на базе проектов с открытым кодом можно выделить российскую разработку Scada Rapid, которая имеет полноценную документацию, поддержку необходимых протоколов. Необходимо проверить, соответствует ли такая система требованиям Scada, иначе можно ли ее использовать для визуализации состояний IoT устройств.

2.4 Описание стенда для автоматизированного управления облачной платформой.

В данном разделе приведены требования к стенду для автоматизированного управления облачной платформой, а также технологии, которые наиболее эффективно используются для оркестрации виртуальных контейнеров.

Стенд для автоматизированного управления облачной платформой состоит из двух сегментов:

- Сегмент вычислений на базе облачной платформы OpenStack. Данных сегмент предназначен для размещения вычислительных ресурсов для обработки данных, запуска приложений, выполнения цифровых двойников, и так далее.
- Сегмент доступа к данным. Представляет из себя сервер, к которому подключены устройства, относящиеся к классу промышленного Интернета вещей. Кроме того, данный сервер может быть подключен к корпоративной сети передачи данных (КСПД) для обеспечения доступа к технологическим данным, поступающим из PI и других систем. На сервере доступа к данным будут развернуты системы управления базами данных, в которые могут быть загружены реальные технологические данные и предоставлены приложениям, размещенных в сегменте вычислений. Сегмент доступа к данным является демилитаризованной зоной между КСПД и сегментом вычислений.

Сервер IoT работает под управлением OS Windows (который может быть сертифицирован по каталогу ПО, что важно т.к. он граничит с сетью предприятия). В этом случае различные генераторы данных, требующие linux, могут быть перенесены в облачный сегмент. В то же время остается

возможность запуска необлачных статических виртуальных машин на windows-server.

Вычислительная инфраструктура, построенная на основе вычислительной платформы стенда, обладает всеми ключевыми возможностями облачной архитектуры [15]:

Самообслуживание по требованию. Потребитель самостоятельно определяет и изменяет вычислительные потребности, такие как серверное время, скорости обработки данных, объём хранимых данных без взаимодействия с представителем поставщика услуг.

Универсальный доступ по сети. Услуги доступны потребителям по сети передачи данных вне зависимости от используемого терминального устройства.

Консолидация ресурсов. Поставщик услуг объединяет ресурсы для обслуживания большого числа проектов (потребителей) в единый пул для динамического перераспределения мощностей между потребителями в условиях постоянного изменения спроса на мощности.

Эластичность. Услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме.

Учёт потребления. Поставщик услуг автоматически исчисляет потреблённые ресурсы (объём хранимых данных, пропускная способность, количество пользователей, количество транзакций)

ПО стенда основано на открытой облачной платформе OpenStack. В настоящее время это одна из наиболее известных открытых облачных платформ. В разработке OpenStack участвуют более 150 компаний, среди которых Cisco, HP, Dell, AMD, Intel и NEC.

ПО облачный платформы способно функционировать на широко распространённом оборудовании и может быть развёрнуто на основе существующей инфраструктуры. Сервера должны быть на основе процессоров с поддержкой технологии Intel VT или AMD SVM и работать

под управлением ОС CentOS. Особые требования к сетевой инфраструктуре и ресурсам хранения данных не предъявляются. Облачная система стенда предлагает различные модели организации сетевых ресурсов. Возможно использование VLAN, DHCP, IPv6 для гибкой настройки и обеспечения безопасности. Ресурсы хранения данных могут быть как сосредоточены на выделенных системах хранения данных (СХД), так и находиться на множестве серверов в распределенной файловой системы GlusterFS.

Архитектура облачной платформы предоставляет широкие возможности по конфигурации компонентов для решения различных классов задач. Для достижения этой цели каждый из составных сервисов спроектирован для предоставления «инфраструктуры как услуги» (laaS). Интеграция достигается при помощи программных интерфейсов приложений (API), предоставляемых каждым сервисом. По большей части это тот же набор API методов, который доступен пользователям облака [16].

Облачная платформа включает набор базовых и дополнительных сервисов. К базовым сервисам относятся:

- Инструментальная панель (horizon) предоставляет графический интерфейс для пользователей и администраторов, позволяя выполнять такие операции, как создание и запуск виртуальных машин, управление сетью и настройка контроля доступа. Служба Dashboard предоставляет панели управления Project, Admin и Settings по умолчанию. Модульная конструкция позволяет взаимодействовать с другими продуктами, такими как биллинг, мониторинг и др. инструменты управления.
- Служба безопасности (keystone) Обеспечивает аутентификацию и авторизацию пользователей для всех компонентов OpenStack. Поддерживает несколько механизмов аутентификации, включая имя пользователя и пароль, системы на основе токенов и учетные записи в стиле AWS.

- Сетевой интегратор (neutron). Отвечает за создание и управление инфраструктурой виртуальных сетей. Элементы инфраструктуры включают в себя сети, подсети и маршрутизаторы. Можно также развернуть расширенные службы, такие как брандмауэры или виртуальные частные сети (VPN).
- Оператор блочных устройств (cinder) Обеспечивает постоянное управление блочным хранилищем для виртуальных жестких дисков. Блочное хранилище позволяет пользователю создавать и удалять блочные устройства, а также управлять подключением блочных устройств к серверам.
- Менеджер виртуальных машин (nova) Служит ядром облака OpenStack, предоставляя виртуальные машины по требованию. Взаимодействуют с базовыми механизмами виртуализации, и, открывая другими компонентами облачной инфраструктуры.
- Реестр образов виртуальных машин (glance) Действует как реестр для образов виртуальных дисков. Пользователи могут добавлять новые образы или использовать снимок существующего сервера. Дает возможность использовать снимки для резервного копирования или в качестве шаблонов для новых серверов [17].

На рисунке ниже приведена функциональная схема взаимодействия сервисов стенда.

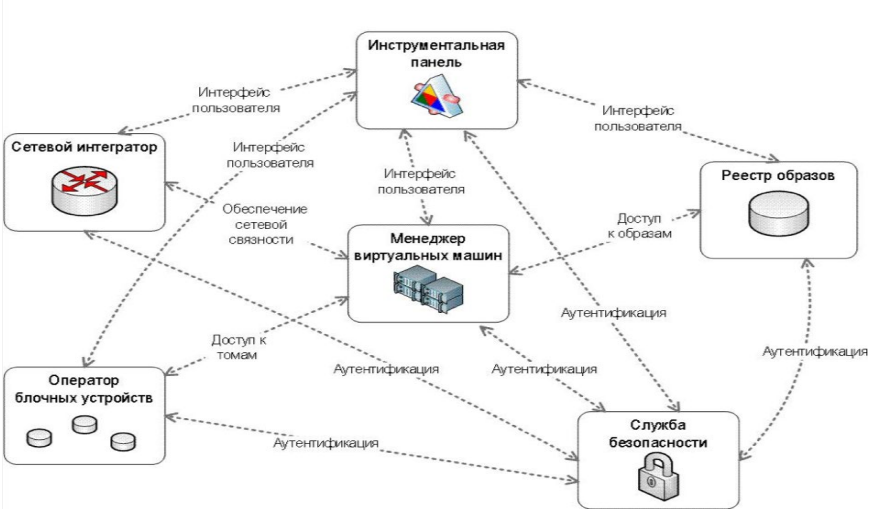

Рисунок 15 - Функциональная схема взаимодействия сервисов стенда

Внутренняя сеть взаимодействия сервисов OpenStack

Эта сеть используется для функций управления OpenStack и обмена, включающего в себя необходимые службы, а также обмен между различными типами узлов OpenStack с помощью OpenStack API и сообщений (например, nova-compute взаимодействует с keystone или cinder-volume обращается к nova-api).

Внешняя сеть

Данная сеть представляет собой сочетание:

- IP адресов для интерфейсов на общественной стороне в узлах контроллера (через которые конечные пользователи получат доступ к службам OpenStack)
- Диапазона подлежащих маршрутизации в общедоступных сетях сетевых адресов IPv4, которые будут использоваться сетевыми средствами OpenStack для плавающих IP.

Эта сеть подключена к узлам контроллера, следовательно, пользователи могут получить доступ к интерфейсам OpenStack, а также подключаться к сетевым узлам для обеспечения виртуальных машин функциональностью маршрутизацией обмена в общедоступных сетях. Эта сеть также подключает сервисные узлы, следовательно, может быть доступна любая сервисная служба, которая должна быть открыта.

Приватная сеть взаимодействия виртуальных машин

Это закрытая сеть, которая не использует общедоступную маршрутизацию и просто используется как частная, внутренняя сеть для обмена между виртуальными машинами в OpenStack, а также между виртуальными машинами и сетевыми узлами, которые обеспечивают маршрутизацию уровня L3 в публичную сеть (а также плавающие IP-адреса для обратной связи с виртуальными машинами). Поскольку эта сеть закрытая, используется другое адресное пространство по сравнению с остальными для четкого определения такого разделения. К этой сети должны подключаться только вычислительные узлы и сетевые узлы OpenStack.

3 Разработка прототипа облачной рсу

3.1 Используемые аппаратно-программные средства

Аппаратная часть стенда включает:

1) Управляющий сервер
2) Сервера виртуализации (количество определяется суммарной нагрузкой)
3) Сервер прототипирования (цифровых систем управления для IIoT)
4) Сетевую инфраструктуру (сети 10 Гб/с и 100 Мбит/с)
5) Станция администрирования
6) Подсистему бесперебойного питания
7) Подсистему IIoT-устройств
8) Стойка (шкаф 19") в сборе

Структурная схема стенда показана на рисунке ниже.

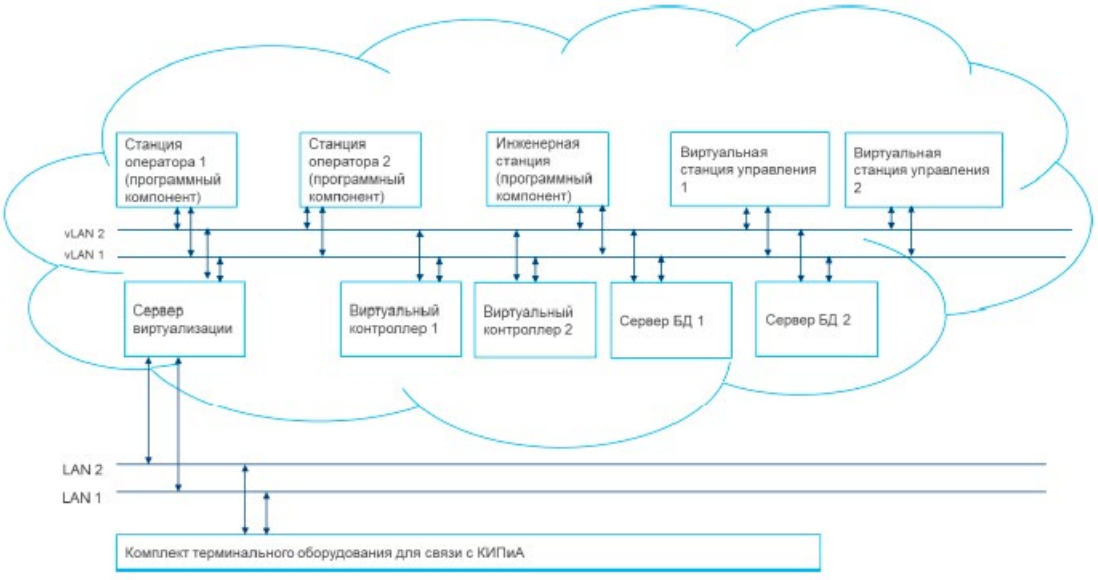

Рисунок 16 - Структурная схема стенда

В качестве первичных устройств в составе стенда предполагается использовать существующие стенды РСУ с программируемыми логическими контроллерами и сенсорами/актуаторами. Для имитации уровня IIoT-

устройств в рамках проекта создается IIoT-стенд с IIoT-контроллерами и шлюзами, которые могут связываться с ПЛК по промышленным протоколам с одной стороны, и передавать данные на серверный уровень, используя IIoT-протоколы (OPC UA, MQTT, IPv6) [18]. Для прототипирования систем управления выделен отдельный сервер (без виртуализации), на котором разворачиваются полноценные системные решения, требующие физических коммуникаций с полевыми устройствами и сегментом IIoT. Для развертывания всего остального информационного окружения в виде смежных систем, моделей объектов, потоков данных и пр. используется облачная среда виртуализации, развернутая на отдельных узлах. Оркестрация развертывания при этом осуществляется с сервера управления [19]. Серверы объединены в высокопроизводительную Ethernet-сеть со скоростью до 10Mbit/s. Для доступа пользователей по беспроводной сети, а также для поддержки доступа через сети мобильных операторов используется беспроводной маршрутизатор с поддержкой WiFi/4G. Питание оборудования стенда обеспечивается с помощью ИБП 220VAC и блоков питания 24VDC.

Внешний вид аппаратных компоненты приведен на рисунке ниже.

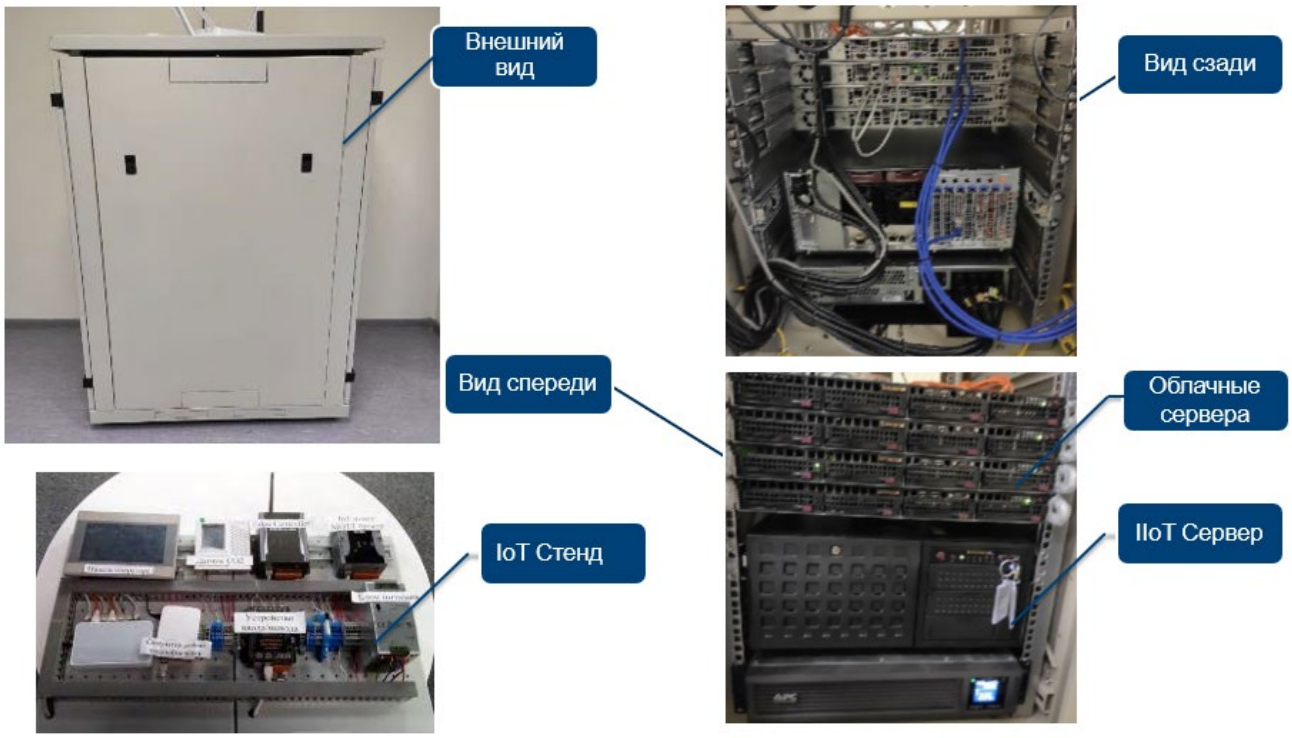

Рисунок 17 - Внешний вид стенда

Для симуляции объекта управления был разработан IoT стенд, описание компонентов которого приведено на рисунке ниже.

Рисунок 18 - IoT стенд

Для управление оркестрацией виртуальных контейнеров был использован OpenStack. В OpenStack организация отдельного проекта выглядит следующим образом:

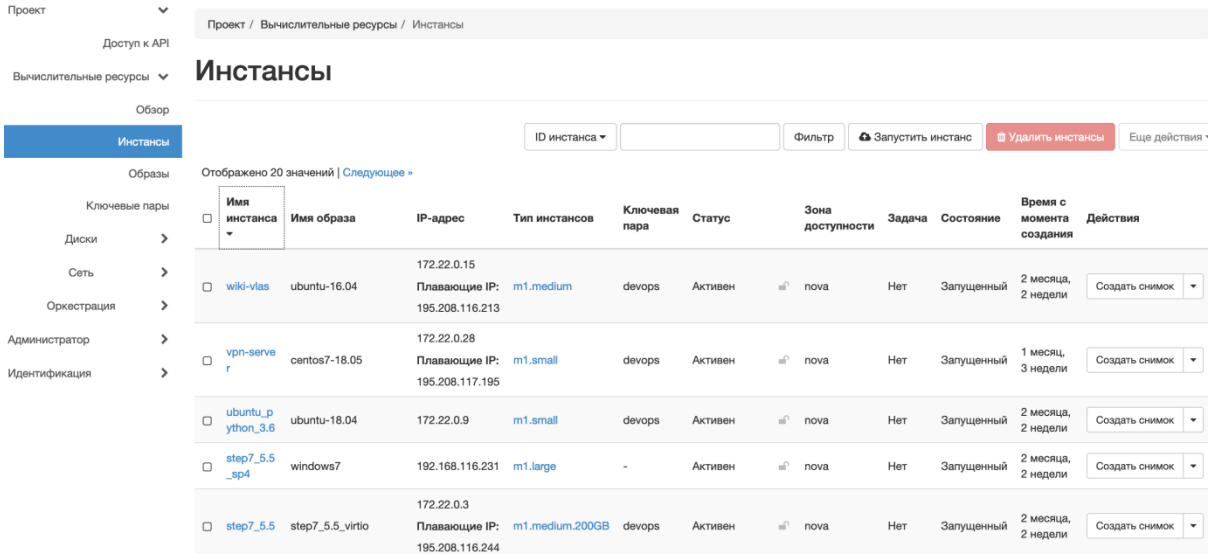

Рисунок 19 - Организация проектов в OpenStack

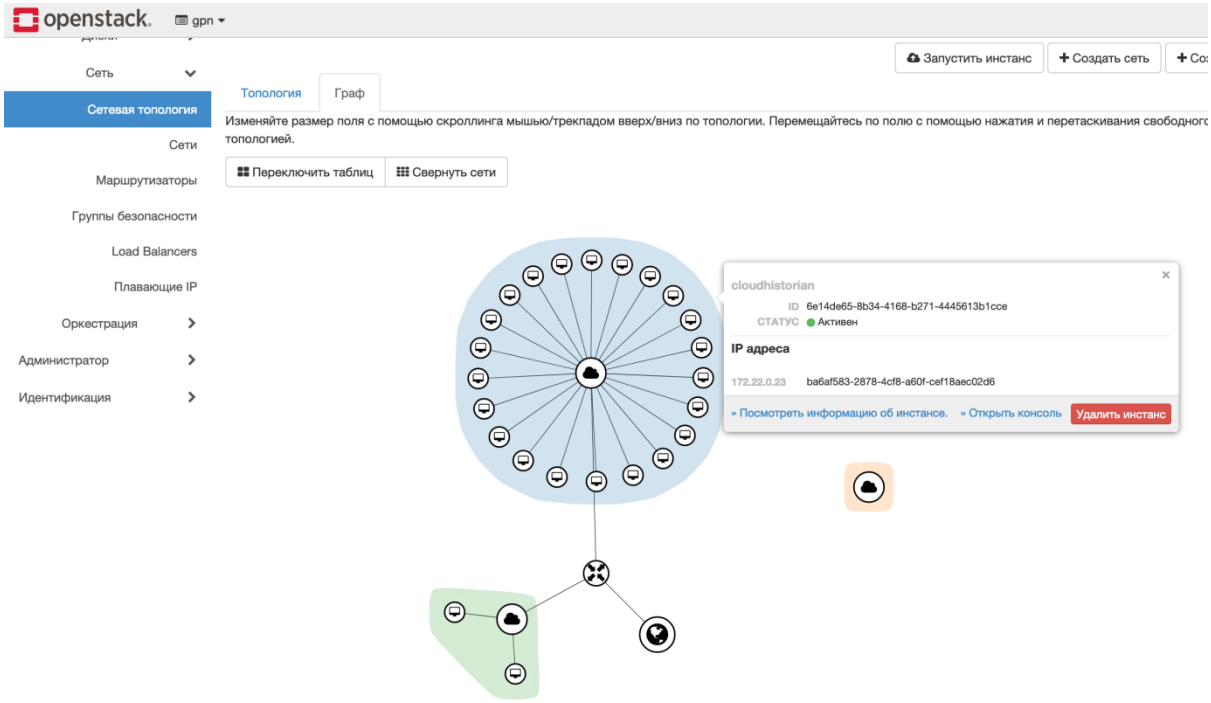

Рисунок 20 - Сетевая топология в OpenStack

1. Реализация стенда на облачной платформе OpenStack позволяет реконфигурировать доступные вычислительные ресурсы под решаемые задачи / апробируемые решения.
2. Одновременная работа различных групп пользователей над различными проектами может быть организована в изолированных проекта и в отдельных виртуальных сетях.
3. Работа с сегментом доступа к данным должна быть организована через правила МЭ, разрешающего доступ виртуальными машинами конкретного проекта к сервисам работы с данными [20].

Для разработки ПО для виртуальных РСУ был использован редактор MasterScada4D. Для разработки веб-сервиса конфигурирования автоматической сборки облачной РСУ был использован язык программирования Python, фреймворк Flask, JavaScript.

3.2 Разработка шаблона развертывания облачных РСУ

Ниже будет описано 2 типа развертывания ВМ: с использованием скриптов, а так же с помощью подготовленных образов.

Первый способ наиболее удобен в случае использования ВМ с Linux, так как большинство ПО, созданного для Linux имеет текстовые конфигурационные файлы, а также может иметь утилиты для конфигурирования из командной строки. Это позволяет писать сложные сценарии развертывания с помощью Bash или систем управления конфигурацией, таких как Ansible или Chef. Windows также позволяет конфигурировать систему с помощью Ansible или Powershell, но далеко не все ПО, которое требуется к установке имеет инструменты для управления своей конфигурацией из командной строки.

В случае если ВМ в качестве OS использует стандартную образы Linux, подготовленные для запуска в облачных сервисах, то дополнительных усилий не требуется.

Второй способ может использоваться в случаях, когда конфигурирование ПО может осуществляться только через графическую оболочку, или наоборот, когда конфигурирование не требуется вовсе и достаточно только иметь образ ВМ с предустановленным ПО.

В таком случае необходимо либо загрузить готовый образ, созданный в другом окружении, что может потребовать конвертации, описанной в 3.3, либо установить необходимое ПО на развёрнутую в Openstack чистую версию Windows или Linux, после чего создать снимок средствами Openstack. Данный снимок далее может использоваться так же как обычный образ.

Второй способ может использоваться в случаях, когда конфигурирование ПО может осуществляться только через графическую оболочку, или наоборот, когда конфигурирование не требуется вовсе и достаточно только иметь образ ВМ с предустановленным ПО.

В таком случае необходимо либо загрузить готовый образ, созданный в другом окружении, что может потребовать конвертации, описанной в 3.3, либо установить необходимое ПО на развёрнутую в Openstack чистую версию Windows или Linux, после чего создать снимок средствами Openstack. Данный снимок далее может использоваться так же как обычный образ.

Оба варианта могут использовать ПО для управления инфрастуктурой, такое как Terraform. Terraform - инструмент для управления инфраструктурой от компании Hashicorp. Он интегрирован с различными облачными сервисами, в том числе с Openstack и позволяет описывать в декларативном виде конфигурацию сети, ВМ и т.д. В частности можно определить скрипт инициализации новой машины. Например, первый блок на изображении вызывает скрипт из шаблона, второй - создает группу безопасности, третий - определяет правило для файрволла, разрешающее подключения на 22 порт

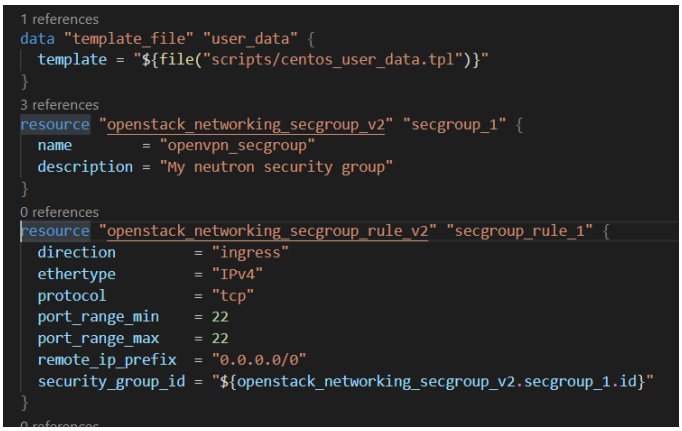

Рисунок 21 - Пример автоматического шаблона развёртывания

Для второго варианта это может использоваться в виде определения некоторого набора виртуальных машин, которые необходимо развернуть

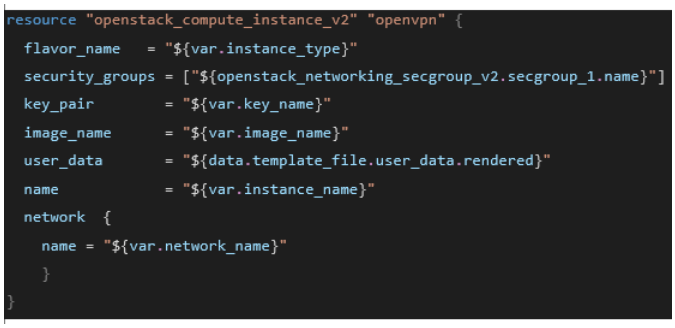

Рисунок 22 - Конфигурация шаблона развертывания

Таких блоков как блок выше может быть несколько, и каждый может описывать сою виртуальную машину. Достаточно менять переменную в var.image_name Таким образом можно определить и развернуть сразу некое окружение с заданным набором заранее сконфигурированных виртуальных машин для некоторого клиента облака.

3.3 Тестирование прототипа облачной РСУ

Облачные РСУ отобранных кандидатов на апробации были развернуты в созданных виртуальных машинах, были выделены следующие вычислительные мощности:

Таблица 1 - Описание виртуальных машин

Название	IP	Характеристики	Роль станции
OpenPLC	172.16.0.121	Ubuntu 18.04, 4гб RAM, 2 VCPUs, 40гб ROM	Виртуальная машина с OpenPLC runtime docker образом
MasterPLC	172.16.0.114	Ubuntu 16.04, 2гб RAM, 2 VCPUs, 40гб ROM	Виртуальная машина с MasterPLC runtime docker образом
Window 10	172.16.0.105	Windows 10, 15.6гб RAM, 3 VCPUs, 80гб ROM	Виртуальная машина с редактором OpenPLC editor и MasterScada

На рисунке ниже показана схема тестирования решений отобранных кандидатов. Подключение IoT устройств к стенду происходит через IoT сервер, который интегрирован в общую подсеть 172.16.0.0/24.

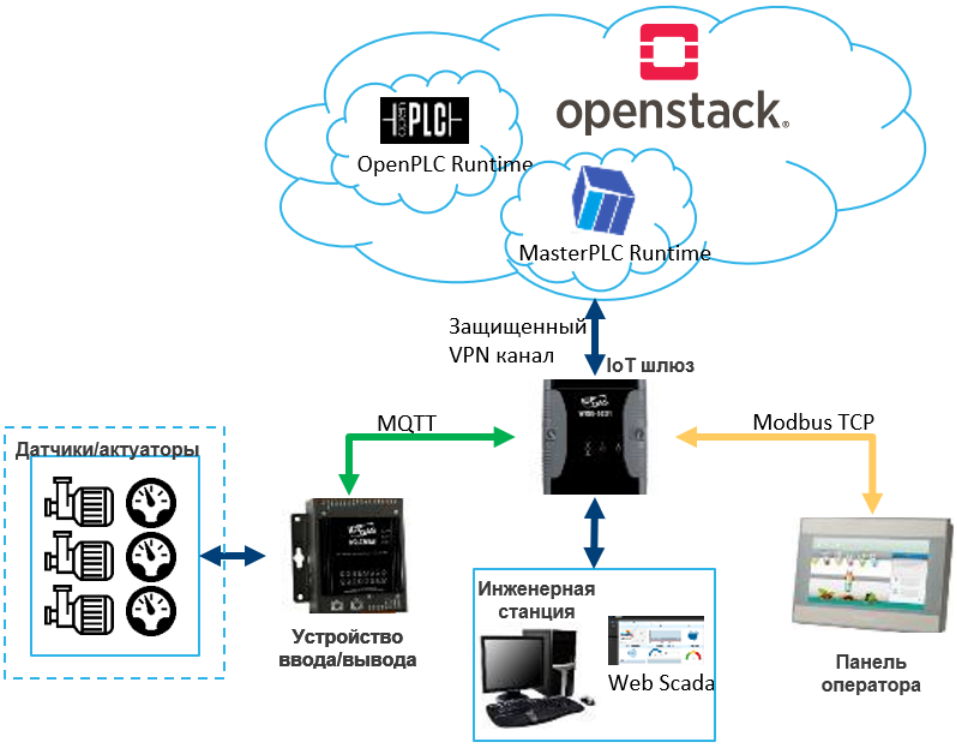

Рисунок 23 - Логическая схема программно-технических средств для испытаний виртуальных

Для демонстрации функционала облачной РСУ был выбран MasterPLC, так как данный кандидат наиболее полно может повторить функционал реальной РСУ.

Подключение к облаку происходит через интернет с помощью open vpn. В данной схеме IoT шлюз выступает в качестве MQTT брокера, через который masterPLC передает сообщение на MQTT IO устройство. Была написана программа на MasterScada для управления водонасосной станцией. Управлять можно как с web интерфейса, так и с HMI панели (панель обменивается данными с контроллером по протоколу Modbus, где контроллер выступает в качестве slave). IO устройство эмитирует работу насосов, которые управляются с помощью дискретных сигналов. На рисунке ниже можно увидеть интерфейс Web-Scada в web браузере, панель HMI и IO устройство. Лампочки на IO устройстве соответствуют номеру насоса. Соответственно на изображении можно увидеть, что номера активных насосов соответствуют номерам включенных насосов с Web-Scada и HMI панели.

Рисунок 24 - Интерфейс Web-Scada на базе Master Scada, взаимодействие IO устройства и панели оператора с облачным контроллером

Также с помощью языка Terraform были разработаны шаблоны автоматического развертывания образов облачной РСУ на стенде. При запуске разработанных скриптов автоматически на стенде создается виртуальная машина с указанными выше характеристиками, скачивается с интернета и устанавливаются образы виртуальных контроллеров и их зависимостей, подгружается выбранный проект в контроллер, подцепляются необходимые лицензии, запускается RunTime. Интерфейс для развертывания виртуального контроллера представлен ниже. Код программы приведен в приложениях А и Б.

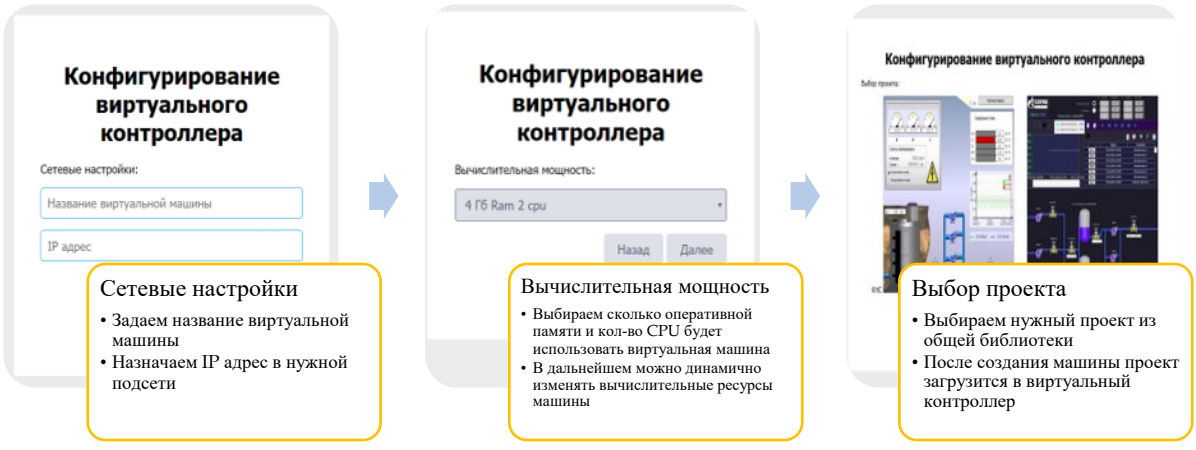

Рисунок 25 - Интерфейс развертывания виртуального контроллера

В результате тестирования облачной РСУ на стенде были сделаны следующие выводы:

OpenPLC:

- Легко виртуализуется на облачной платформе
- Может выступать облачным Modbus-коннектором с предобработкой данных
- Не поддерживает функции резервирования
- Может встраиваться в iot-устройства на микро-ПК в качестве среды исполнения программ управления на языках МЭК 61131-3
- Не устойчиво работает

Таблица 2 - Преимущества и недостатки OpenPLC

Преимущества	Недостатки
OpenSource проект. Программное обеспечение имеет открытый исходный код и лицензию GNU General Public License, которая позволяет использовать данный продукт в коммерческих проектах бесплатно.	Поддержка только одного протокола коммуникации. Платформа поддерживает только Modbus TCP, поддержку остальных протоколов необходимо дописывать вручную. Можно использовать только один протокол, но для взаимодействия с устройствами, которые не поддерживают Modbus необходимо настраивать IoT шлюз для конвертации протоколов
Кроссплатформенность. Opeplc можно развернуть как и на ОС Windows, так и на ОС Linux, что позволяет использовать докер-контейнеры и автоматические шаблоны развертывания на стенде.	Ограниченный функционал OpenPLC не способен полностью повторить функции промышленных контроллеров, в связи с множеством ручных настроек необходимого функционала, такого как резервирование, связь между несколькими контроллерами.
Форум с технической поддержкой. На официальном сайте разработчика предоставлен форум, где можно размещать сообщения с техническими проблемами и идеями для реализации, открытых для обсуждений.	

Выводы:

1. OpenPLC имеет многие признаки промышленного контроллера, способен управлять небольшой системой формата «умный дом» или «интернет вещей», несложной небольшой станцией
2. OpenPLC не подходит для реального производства, так как не способен поддерживать необходимые функции резервирования, устойчивости, протоколы.
3. Данный виртуальный контроллер легко развернуть в облаке, используя шаблоны развертывания на стенде, так как мы можем обернуть образ в докер контейнер, тем самым после развертывания ему автоматически присвоится ip адрес, к web интерфейсу контроллера можно подключиться через браузер.
4. Таким образом OpenPLC можно использовать для реализации программ, которые не учувствуют в непосредственном управлении производством, а анализирует, преобразует собранные по Modbus TCP регистры в верхний уровень.
5. Также OpenPLC можно установить в небольшие платы, например Raspberry Pi, для управления «умными» вещами и iot устройствами, так как написание программ управления возможно на языках стандарта МЭК 61131-3, нет необходимости изучать специальные языки таких плат.
6. Возможность эффективной реализации функций распределенных систем управления и функций операторов управления непрерывными процессами в облачной инфраструктуре и интеграции с архитектурой IIoT достигается только при управлении небольшими автоматическими системами, в основном включающие в себя IoT устройства. OpenSource проекты способны решить такие задачи, как управление умным домом, управление системами, где не требуется связь между контроллерами и наименьшее время цикла (например управление

Master PLC:

- Легко разворачивается в облаке с помощью шаблонов развертывания
- Web Scada интегрируется с исполняемой программой softPLC
- Поддерживает функции резервирования
- Поддерживает множество протоколов коммуникации и резервирования
- Цикл программы ~10 мс
- Задержка срабатывания 130-140 мс через интернет (ping ~50мс), ~10мс через прямое подключение устройств к IoT серверу.

Таблица 3 - Преимущества и недостатки MasterPLC

Преимущества	Недостатки
Импортозамещение. Разработчиком данного продукта является отечественная компания «ИнСАТ». Это облегчает процесс взаимодействия при тестировании и отладки программного обеспечения	Горячая загрузка. MasterPLC не поддерживает загрузку изменений конфигурации и программы без остановки контроллера..
Кроссплатформенность. MasterPLC и MasterScada можно развернуть на следующие виды операционных систем: Windows, Linux, QNX, Android, Эльбрус.	Ограничение на минимальное время цикла. В MasterPLC существует ограничение на минимальное время цикла – для 1мс ошибка составила 11%, для 10мс ошибка составила 1%. Таким образом нужно экспериментально подбирать необходимое время цикла.
Интерфейс. Среда разработки ПО имеет интуитивно-понятный интерфейс, обучающие видео и проекты, удобные окна разработки программ и редактор окон Web Scada	

Выводы:

1. MasterPLC (MasterScada) имеет множество признаков промышленного контроллера, способен управлять реальным производством, где не требуется большой точности во времени выполнения циклов.
2. Данный виртуальный контроллер легко развернуть в облаке, используя шаблоны развертывания на стенде, так как мы можем обернуть образ в докер контейнер, тем самым после развертывания ему автоматически присвоится ip адрес, к web интерфейсу контроллера можно подключиться через браузер.
3. Таким образом MasterPLC можно использовать для реализации проектов с автоматизацией производств, можно выделить полное управление производством виртуальному контроллеру, либо только часть, которая отвечает за анализ данных.
4. ПО поддерживает Web Scada, которая легко интегрируется с исполняемой программой, а также собственную базу данных, в которую можно архивировать временные ряды. Поддерживает множество протоколов коммуникации и резервирования.

Возможность эффективной реализации функций распределенных систем управления и функций операторов управления непрерывными процессами в облачной инфраструктуре и интеграции с архитектурой IIoT достигается за счет поддержки многочисленных

Исходя из результатов была подтверждена следующая гипотеза - возможность эффективной реализации функций распределенных систем управления и функций АРМ оператора для управления непрерывными технологическими процессами в облачной инфраструктуре и интеграции с архитектурой IIoT.

ЗАКЛЮЧЕНИЕ

Подведем итоги проведенного выпускного квалификационного исследования и охарактеризуем кратко его основные результаты.

Прежде всего, в главе 1 были обозначены проблемы традиционной архитектуры САУ, приведены способы решения данных проблем при переходе от традиционной архитектуры в SOA, описаны шаги миграции из традиционной архитектуры в SOA.

В главе 2 был приведен обзор на современные IaaS платформы, а также на существующие решений виртуальных контроллеров. Были приведены функциональные и технические требования к разработке программно-аппаратной части облака. Были выбран технологический стек, использованный в разработке прототипа облачной РСУ.

В главе 3 был описан процесс разработки прототипа облачной РСУ, приведено описание программно-аппаратной части, используемой в данном проекте.

Таким образом, ценностным предложением разработанной архитектуры и прототипа является:

1. Возможность гибкой модернизации (трансформации) существующих систем управления
2. Возможность построения полностью интегрированных цифровых систем управления
3. Сокращение и минимизация расходов на модернизацию промышленных систем

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1 Alekseev A.P., Efremov V.V., Potekhin V.V., Zhao Y., Du H. (2020) Digital Twin Analytic Predictive Applications in Cyber-Physical Systems. In: Arseniev D., Overmeyer L., Kälviäinen H., Katalinić B. (eds) Cyber-Physical Systems and Control. CPS&C 2019. Lecture Notes in Networks and Systems, vol 95. Springer, Cham. – P. 1-2

2 Шереметова Е.И., Потехин В.В. Распределенный анализ категориальных последовательностей для непрерывного производства // Современная техника и технологии, No 5 — 2017. – 10 с.

3 Ponomarev, K.; Kudryashov, N. & Popelnukha, N. (2017). Main Principals and Issues of Digital Twin Development for Complex Technological Processes, Proceedings of the 28th DAAAM International Symposium, pp.0523-0528, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978- 3-902734-11-2, ISSN 1726-9679, Vienna, Austria. – P. 22-24

4 Kerdprasop K., Kerdprasop N. Feature Selection Technique to Improve Performance Prediction in a Wafer Fabrication Process //Latest Trends in Engineering Mechanics, Structures, Engineering Geology. – 2014. – P. 128- 133.

5 Colombo, Armando & Bangemann, Thomas & Karnouskos, Stamatis & Delsing, Jerker & Stluka, Petr & Harrison, Robert & Jammes, Francois & Martinez Lastra, Jose Luis. (2014). Industrial cloud-based cyber-physical systems: The IMC-AESOP approach. 10.1007/978-3-319-05624-1. – P. 35

6 Colombo, Armando & Karnouskos, Stamatis & Bangemann, Thomas. (2014). Towards the Next Generation of Industrial Cyber-Physical Systems. 10.1007/978-3-319-05624-1_1. – P. 12

7 H. Ahmadi, A. Moosavian, and M. Khazaee, "An appropriate approach for misalignment fault diagnosis based on feature selection and least square support vector machine," International Journal of Mechanics, vol. 6, issue 2, 2012. - P. 97-104.

8 Delsing, Jerker & Carlsson, Oscar & Arrigucci, Fredrik & Bangemann, Thomas & Hübner, Christian & Colombo, Armando & Nappey, Philippe & Bony, Bernard & Karnouskos, Stamatis & Nessaether, Johan & Kyusakov, Rumen. (2014). Migration of SCADA/DCS systems to the SOA cloud. 10.1007/978-3-319-05624-1_5. – P. 10-12

9 Обзор платформ для построения облаков: https://habr.com/ru/post/140375/ (Дата обращения 23.02.2020)

10 ООО «Сименс» 2011 Программируемые контроллеры SIMATIC WinAC: https://docplayer.ru/51741792-Programmiruemye-kontrollery-simatic-winac-ooo- simens-2011.html (Дата обращения 04.10.2019)

11 Описание CX-One Simulator: https://industrial.omron.ru/ru/products/cx-simulator (Дата обращения 05.10.2019)

12 Описание MasterScada и MasterPLC: http://www.masterscada.ru/ (Дата обращения 24.11.2019)

13 Исследование безопасности Codesys runtime: https://ics-cert.kaspersky.ru/reports/2019/09/17/security-research-codesys-runtime-a-plc-control- framework-part-1/#_Toc16529220 (Дата обращения 18.12.2019)

14 Описание OpenPLC https://www.openplcproject.com (Дата обращения 14.09.2019)

15 Yerofeyev S.A., Ipatov O.A., Markov S.A., Potekhin V.V., Sulerova A.S. & Shkodyrev V.P. (2015) Adaptive Intelligent Manufacturing Control System, Proceedings; DAAAM International Symposium of Intelligent Manufacturing and Automation, DAAAM 2015; Vienna; Austria; Volume 26, No. 1, ISSN 2304-1382. – P. 11-13.

16 Kudriashov N., Protasov I., Markov S., Potekhin V., Yadgarova Y., Taratukhin V. (2016). Implementation of Cloud Services for Advance Management of Steel Transport for Continuous Casting Production, Proceedings of the 27th DAAAM International Symposium, pp.0457-0462, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-08-2, ISSN 1726-9679, Vienna, Austria. – P. 14-18.

17 Gilchrist, A. Industry 4.0, The Industrial Internet of Things [Text] / A. Gilchrist. – New York: Apress, 2016. – P. 532.

18 Marz, N. Big Data. Principles and best practices of scalable realtime data systems [Text] / N. Marz, J. Warren. – New York: Manning, 2015. – 689 p

19 Bagheri, B., Cyber-physical Systems Architecture for Self-Aware Machines in Industry 4.0 Environment [Text] / B. Bagheri, [et al]. — Boston : pbl. IFAC-PapersOnline, 2015. – P. 78 – 123

20 Yerofeyev S., Ipatov O., Markov S., Potekhin V., Sulerova A., Shkodyrev V. (2016). Adaptive Intelligent Manufacturing Control Systems, Proceedings of the 26th DAAAM International Symposium, pp.1016-1024, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, ISSN 1726-9679, Vienna, Austria. – P. 22.

ПРИЛОЖЕНИЕ А

(обязательное)

Код программы шаблона автоматического развертывания облачной РСУ на языке terraform

```
data "template_file" "user_data" {
  template = "${file(var.file_name)}"
}
resource "openstack_networking_port_v2" "port_1" {
  name           = "port_1"
  network_id     = "${var.network_id}"
  port_security_enabled = "false"
  admin_state_up = "true"
  fixed_ip {
    subnet_id = "${var.subnet_id}"
    ip_address = "${var.ip_address}"
  }
}

resource "openstack_compute_instance_v2" "mplc_test" {
  flavor_name   = "${var.instance_type}"
  #security_groups = ["${openstack_networking_secgroup_v2.secgroup_1.name}"]
  security_groups = ["${var.security_group}"]
  key_pair        = "${var.key_name}"
  image_name      = "${var.image_name}"
  user_data       = "${data.template_file.user_data.rendered}"
  name            = "${var.mplc_name}"
  network {
    name = "${var.network_name}"
    port = "${openstack_networking_port_v2.port_1.id}"
    }
}
variable "mplc_name" {
  description = "The name of the MPLC instance."
  default = "mplc_auto"
}

variable "instance_type" {
  default = "m1.medium"
}

variable "key_name" {
  default = "mplc"
  description = "SSH key name"
}

variable "image_name" {
  default = "ubuntu-18.04"
}
variable "network_name" {
  default = "flat-provider-network"
}
variable "network_id" {
  default = "45f38a6a-2732-43e0-9bec-bda8b9179270"
}

variable "subnet_id" {
  default = "dfe0e4c4-13aa-447c-a696-0489cebcae20"
}
```

```
variable "ip_address" {
  default = "172.16.0.168"
}

variable "floating_ip_networks" {
  default = ""
}

variable "security_group" {
  default = "jenkins_secgroup"#"27855609-e3fb-4d6b-a5b4-b1ec855ca80e"
}

variable "file_name" {
  default = "scripts/ubuntu_user_data_vns.tpl"
}
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Код программы веб-сервиса для конфигурирования облачной РСУ на языке Python

```
import time

from flask import Flask, render_template, request, jsonify, redirect
import random
from python_terraform import *

app = Flask(__name__)


@app.route('/')
def hello_world():
    return render_template("main.html")


@app.route("/create_controller", methods=['post'])
def create_controller():
    project = f"scripts/ubuntu_user_data_auto_{request.values['project']}.tpl"

    t = Terraform()
    t.init()
    return_code, stdout, stderr = t.plan(var={'mplc_name': request.values["name_virt"],
                                      'ip_address': request.values["ip_address"],
                                      'file_name': project,
                                      'instance_type': request.values["resource"]})
    return_code, stdout, stderr = t.apply(skip_plan=True, var={'mplc_name': request.values["name_virt"],
                                               'ip_address': request.values["ip_address"],
                                               'file_name': project,
                                               'instance_type': request.values["resource"]})
    time.sleep(30)
    return redirect('http://'+request.values["ip_address"])


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5003)
```