Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Прикладной Математики и Информатики
Кафедра технологий цифровой трансформации

Направление подготовки / специальность: 03.04.01 Прикладные математика и физика
Направленность (профиль) подготовки: Цифровая инженерия, информационные технологии и дискретная математика

РАЗРАБОТКА ЦИФРОВОЙ ПЛАТФОРМЫ АСИНХРОННОЙ СЕНСОРНОЙ СИСТЕМЫ ДИСТАНЦИОННОГО МОНИТОРИНГА РЕСУРСОПОТРЕБЛЕНИЯ ДОМОХОЗЯЙСТВ

(магистерская диссертация)

Студент:
Филиппов Иван Михайлович

_____
(подпись студента)

Научный руководитель:
Логинов Валерий Николаевич,
канд. техн. наук, ст. науч. сотр.

_____
(подпись научного руководителя)

Консультант *(при наличии)*:
_____
_____
(подпись консультанта)

Москва 2024

АННОТАЦИЯ

Работа посвящена разработке цифровой платформы асинхронной сенсорной системы дистанционного мониторинга ресурсопотребления домохозяйств, предназначенной для получения, хранения, обработки и анализа данных о ресурсопотреблении домохозяйств, собираемых системой мониторинга.

Такая платформа может стать связующим звеном между производителями энергоэффективного оборудования и потребителями, нуждающимися в оптимизации потребления энергоресурсов, а также может явиться одним из инструментов формирования нового рынка – рынка данных о ресурсопотреблении городских агломераций.

Целью работы является разработка инфраструктурной части цифровой платформы дистанционного мониторинга ресурсопотребления домохозяйств, главная задача которой состоит в обеспечении получения, хранения и первичной обработки данных.

Для достижения этой цели в работе решаются задачи выбора формата входных данных цифровой платформы и разработка архитектуры и программного кода инфраструктурной части цифровой платформы.

Ключевые слова: цифровая трансформация, цифровая платформа, ресурсопотребление домохозяйств, дезагрегация, NILM.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящем отчете о НИР применяют следующие сокращения и обозначения:

_by_point_count и _by_peak_consumption – обозначения для используемых ограничений при проведении экспериментов.

accum – обозначение, используемое в названии файлов или переменных и методов в коде. Служит для обозначения данных, где значением выступает текущее (накопленное) значение счётчика, от англ. accumulated.

accumulated_distance и power_distance – метрики, используемые для сравнения эффективности подходов формирования данных о потреблении ресурсов (входных данных для цифровой платформы).

IDEAL – открытый датасет (набор данных), предназначенный для тренировки моделей машинного обучения, нацеленных на решение задач NILM.

NILM (Non-Intrusive Load Monitoring) – неинтрузивный мониторинг нагрузки. Процесс и метод определения состава потребителей электроэнергии на основе показаний единственного датчика, установленного на входе электросети домохозяйства.

power – обозначение, используемое в названии файлов или переменных и методов в коде. Служит для обозначения данных, где значением выступает текущая мощность потребления электроэнергии.

REDD (Reference Energy Disaggregation Data Set) – открытый датасет для тренировки моделей машинного обучения, нацеленных на решение задач NILM.

Лямбда-архитектура – паттерн обработки данных, главной концепцией которого является разделение входящих данных на два потока – для пакетной обработки и обработки в реальном времени.

Ресурс – какой-либо вид энергоресурса, потребляемого домохозяйством, – холодная и горячая вода, газ, электроэнергия.

Смарт-мониторинг – (англ. smart monitoring) – подход к сбору данных о потреблении ресурсов, обеспечивающий высокую степень детализации информации.

Счётчик, смарт-счётчик, датчик, сенсор – синонимичные обозначения (в рамках данной работы) для устройств, измеряющих потребление ресурсов домохозяйством.

Сырые данные – первичные, необработанные данные в исходном виде, собранные из источника.

ВВЕДЕНИЕ

1.1 Актуальность работы

В современном мире остро стоит проблема эффективного использования природных ресурсов. Ее решение важно, как для сохранения окружающей среды в целом, так и для смягчения последствий глобального изменения климата, вызванного выбросами парниковых газов предприятиями промышленности и энергетики.

Одним из основных типов потребителей энергетических и других видов ресурсов являются домохозяйства. Уменьшение потребления в этом секторе за счет оптимизации моделей потребления является одной из актуальных научно-технических задач, на решение которой нацелено научное сообщество.

Для анализа и оптимизации моделей потребления необходима организация мониторинга самого процесса потребления ресурсов. При этом для сбора данных применяется технология смарт-мониторинга, обеспечивающая отслеживание потребления ресурсов с высокой степенью детализации.

Существуют два основных подхода к мониторингу – синхронный и асинхронный, которые отличаются форматом собираемой информации. И тот, и другой вырабатывают много данных, которые нужно фиксировать, обрабатывать, хранить, анализировать и принимать решения по результатам обработки и анализа. В современных условиях цифровой экономики наиболее перспективным подходом для решения перечисленных задач является создание цифровой платформы.

Цифровая платформа мониторинга данных о ресурсопотреблении обеспечивает все операции по приему, обработке и хранению данных (инфраструктурные операции с данными), а также операции по аналитической обработке данных и принятию решений.

Представляется, что в перспективе разработка и внедрение таких платформ может обеспечить повышение эффективности планирования ресурсопотребления и принятия соответствующих решений как в масштабе больших городских агломераций, так и в масштабе небольших посёлков, расположенных в географических зонах с недостатком энергоресурсов (вахтовые поселки, военные городки и т.п.).

Также цифровая платформа может создать новое ценностное предложение для потребителей энергоресурсов и стать площадкой поиска клиентов для компаний-производителей энергоэффективного оборудования.

1.2 Цели и задачи исследований

Таким образом, целью проводимых исследований является разработка цифровой платформы дистанционного мониторинга ресурсопотребления домохозяйств.

При этом, очевидно, что в рамках магистерской диссертации не могут быть решены абсолютно все вопросы, связанные с разработкой такой платформы и, в частности, такие вопросы, как разработка, тестирование и отладка программного обеспечения на реальных данных, организация интерфейсов с устройствами передачи данных, разработка методов и программного обеспечения аналитической обработки данных и многие другие.

Поэтому в данной работе основной акцент был сделан на разработку прототипа инфраструктуры цифровой платформы, обеспечивающей получение, ввод, хранение и предварительную обработку данных о ресурсопотреблении.

Для достижения поставленной цели работы в процессе исследований были решены две задачи:

- определение требований к оптимальным характеристикам входного потока данных от сенсоров (датчиков, счётчиков), размещенных в

домохозяйствах, которые должны поступать для обработки цифровой платформой;

- разработка проекта инфраструктурной части цифровой платформы, обеспечивающей сбор и обработку поступающих данных, включая проектирование архитектуры системы, решение вопросов интеграции инструментов для работы с большими данными (Hadoop, Hive, Spark и Kafka), реализацию отдельных элементов платформы и обеспечение их скоординированной работы, а также разработку и реализацию проектных решений, обеспечивающих возможность в дальнейшем существенного расширения функционала платформы при возникновении необходимости в этом.

1.3 Основные результаты

В ходе проведенных исследований в рамках решения первой задачи были проведены численные эксперименты для сравнения эффективности двух существующих на данный момент подходов к формированию входных данных (синхронного и асинхронного), а также разработан метод анализа результатов экспериментов на базе общепринятых в задачах NILM датасетов.

По результатам анализа обоснован вывод о том, что при разработке цифровой платформы мониторинга ресурсопотребления целесообразно ориентироваться на входной поток данных, образующийся при асинхронном способе формирования данных.

В результате решения второй задачи создан прототип системы для обработки и хранения данных, генерируемых системой сбора информации о ресурсопотреблении домохозяйств, который может послужить основой для разработки соответствующего коммерческого продукта.

1.4 Научная и практическая значимость полученных результатов

Научная значимость работы заключается в том, что в результате проведенных исследований был сделан научно обоснованный вывод о том, что

при разработке цифровой платформы мониторинга данных о ресурсопотреблении целесообразно ориентироваться на входной поток данных, которые формируются при асинхронном подходе поскольку при этом обеспечивается возможность повышения качества анализа данных и улучшения точности предсказательных моделей потребителей.

Практическая значимость работы состоит в том, что создан прототип коммерческого продукта, который может внести существенный вклад в процесс возникновения и формирования рынка данных о потреблении ресурсов домохозяйствами.

В ходе выполнения настоящей работы была подготовлена научно-техническая статья и результаты исследований были доложены на 65-й Всероссийской научной конференции МФТИ в честь 115-летия Л.Д. Ландау с публикацией тезисов в трудах конференции [1].

2 Смарт-мониторинг и подходы к данным о потреблении ресурсов

Рассмотрим более подробно что же называют смарт-мониторингом. Это – технология, основанная на применении системы дистанционного сбора и мониторинга данных со счётчиков, фиксирующих потребление ресурсов (вода, газ, тепло, электроэнергия) в домохозяйствах [1]. Такая система требует использования специальных счётчиков, которые имеют возможность в автоматическом режиме отправлять данные для последующей обработки.

Важной деталью, которую необходимо отметить, является уровень детализации собираемой информации. Для получения практически значимых результатов анализа данных важно знание распределения потребления ресурсов во времени, поэтому данные могут поступать со счётчиков с высокой частотой. В частности, при решении модельной задачи дезагрегации обычно используют данные с дискретом по времени в 1 секунду. А это создаёт значительные объёмы данных.

В этой связи возникает необходимость в специализированном программном обеспечении, которое будет обрабатывать полученные данные о потреблении. Уже сейчас существуют системы, которые позволяют частным лицам установить дополнительное оборудование и отслеживать потребление в реальном времени, а также автоматически оплачивать счета. Однако в силу различных причин стоимостного и организационного характера такие инструменты не получили широкого распространения.

Возникает логичный вопрос: а зачем рядовому потребителю нужен такой мониторинг, какие преимущества он даёт? Использование умных мониторинговых систем позволяет выявлять экстренные ситуации, такие как прорыв труб водоснабжения или утечка газа, и автоматически на них реагировать, например, перекрывая подачу или вызывая специальную службу.

Помимо этого, выясняется, что при достаточной детализированности данных можно определять, какие приборы использует потребитель, и вырабатывать рекомендации по оптимизации его затрат. Для этого, в

частности, необходимо решить задачу дезагрегации данных (разделения каналов), поскольку счётчик снимает информации с единого потока, являющегося суммой каналов каждого прибора.

Как показали результаты исследований в области NILM-технологии (Nonintrusive appliance load monitoring) [2, 3, 4, 5, 6], из полученных данных можно восстановить информацию об используемых приборах и моделях потребления ресурсов различными домохозяйствами, узнать типы устройств, и их характеристики [1]. Рисунки 1, 2 и 3 демонстрируют визуальное отличие графиков потребления электроэнергии разными приборами. Когда приборы совместно подключены в сеть, то их графики потребления накладываются друг на друга. Схематически это показано на рисунке 4, а пример реальных данных приведён на рисунке 5.

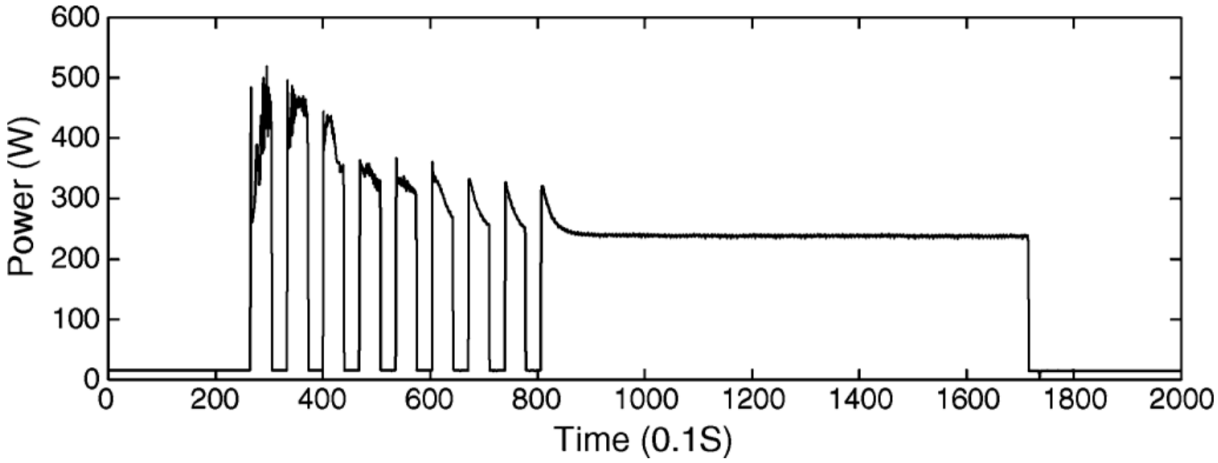

Рисунок 1 – Пример графика энергопотребления стиральной машины [4]

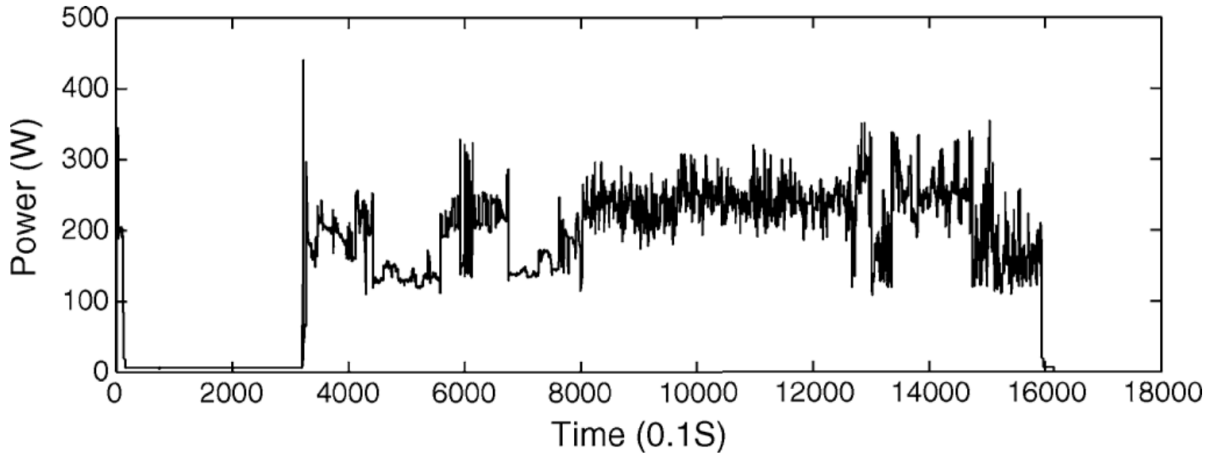

Рисунок 2 – Пример графика энергопотребления плазменного телевизора [4]

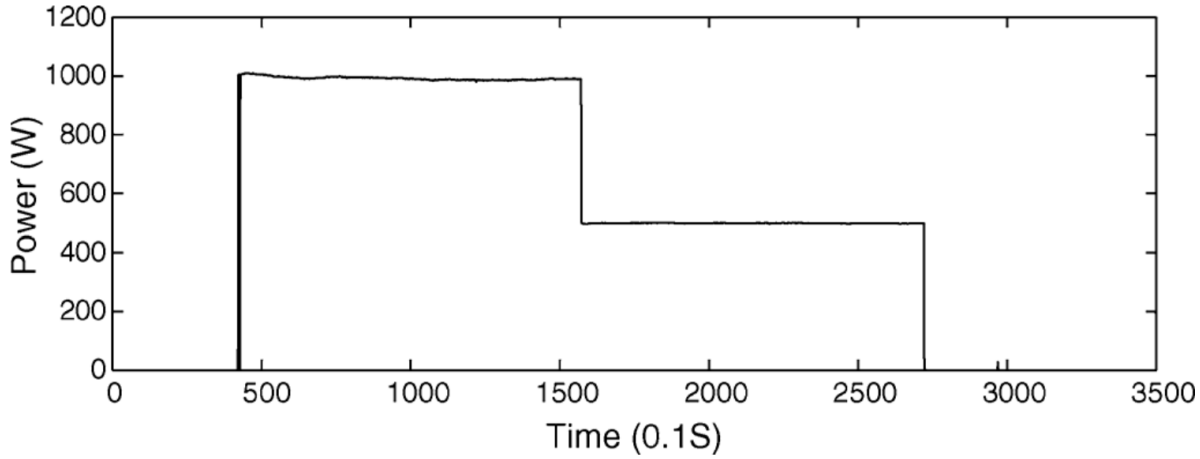

Рисунок 3 – Пример графика энергопотребления обогревателя [4]

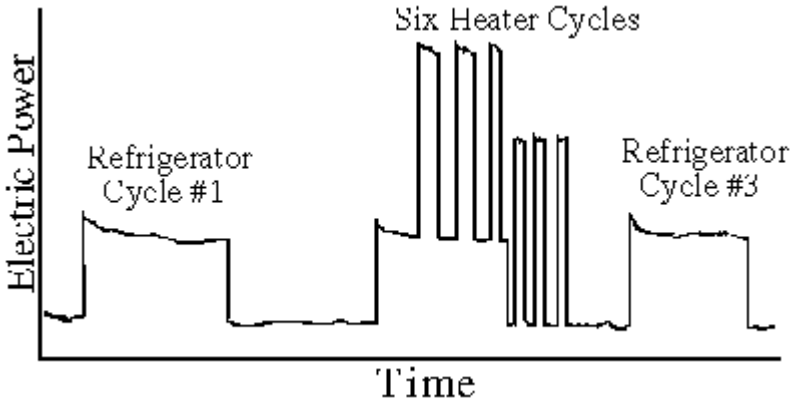

Рисунок 4 – Демонстрация наложения потребления приборов [3]

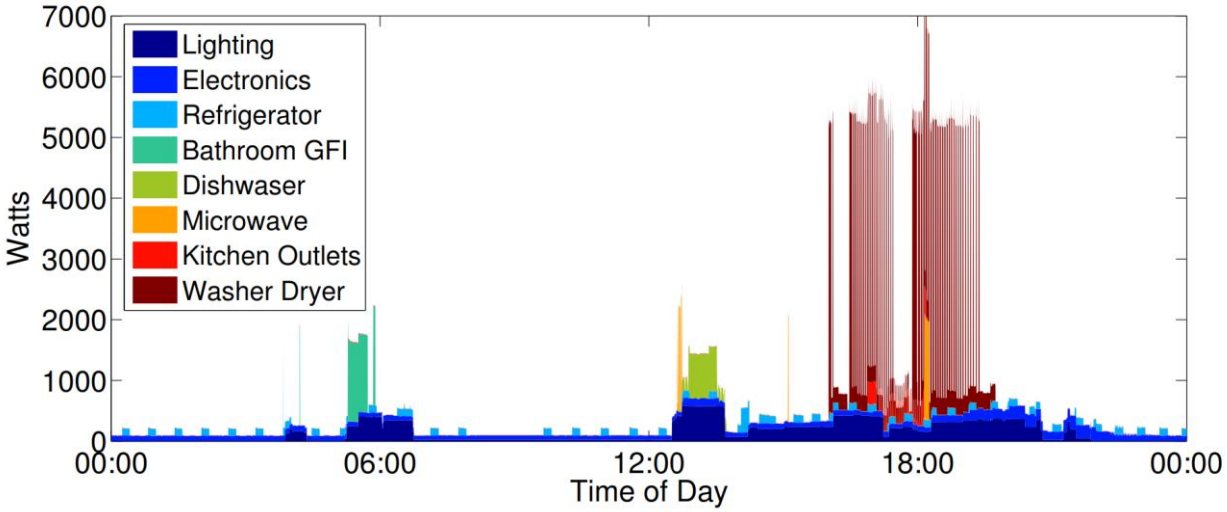

Рисунок 5 – Реальные данные о потреблении электроэнергии с разделением на отдельные приборы [7]

Анализ показателей потребления сразу нескольких видов ресурсов одним потребителем может дать ещё больше информации, чем анализ потребления одного из ресурсов. На основе этой информации потребителю

могут быть сформированы рекомендации по снижению потребления ресурсов как за счет изменения самой модели потребления (например, при использовании день-ночь тарифа), так и за счет замены устройства на более энергоэффективное [1].

В работах о смарт-мониторинге, как правило, рассматривается один из двух подходов к сбору данных – синхронный или асинхронный. В системах, построенных на базе синхронного подхода, счётчик посылает данные через равные промежутки времени Δt. Очевидно, что чем чаще опрашиваются счётчики, тем более подробную информацию можно получить. Такие системы довольно просты в реализации, однако они обладают некоторыми существенными недостатками:

- для получения уровня детализации, который необходим для построения моделей ресурсопотребления и формирования соответствующих рекомендаций по повышению его эффективности, Δt должно быть весьма мало;
- из-за частой отправки данных элементы питания счётчиков быстро разряжаются;
- счётчики избыточно нагружают каналы передачи данных, так как отправка информации производится и в те периоды времени, в которые ресурсы не потребляются.

При асинхронном сборе данных счётчик активируется не по прошествии определённого времени, а при расходе очередной «порции» ресурса Δr. Концепция построения такой системы описана в работе [8].

В асинхронной системе в том случае, если ресурс не потребляется, данные не отправляются. Это позволяет избежать избыточной загрузки каналов передачи данных и разряда элемента питания в паузах, например, если жильцы уехали в отпуск. Как показано в упомянутой работе, асинхронная система мониторинга имеет определенные преимущества перед синхронной

системой за счет, прежде всего, возможной более высокой степени детализации данных, которая необходима для построения адекватных моделей ресурсопотребления и формирования рекомендаций по повышению его эффективности, а также за счет меньшей загрузки каналов передачи данных, более длительного периода межсервисного обслуживания, связанного с заменой элементов питания, в целом более высокой надежности и, в частности, за счет меньшей вероятности ущерба при попытке хакерского взлома.

Примеры графиков данных о потребленном ресурсе для синхронной и асинхронной систем мониторинга представлены на рисунке 6.

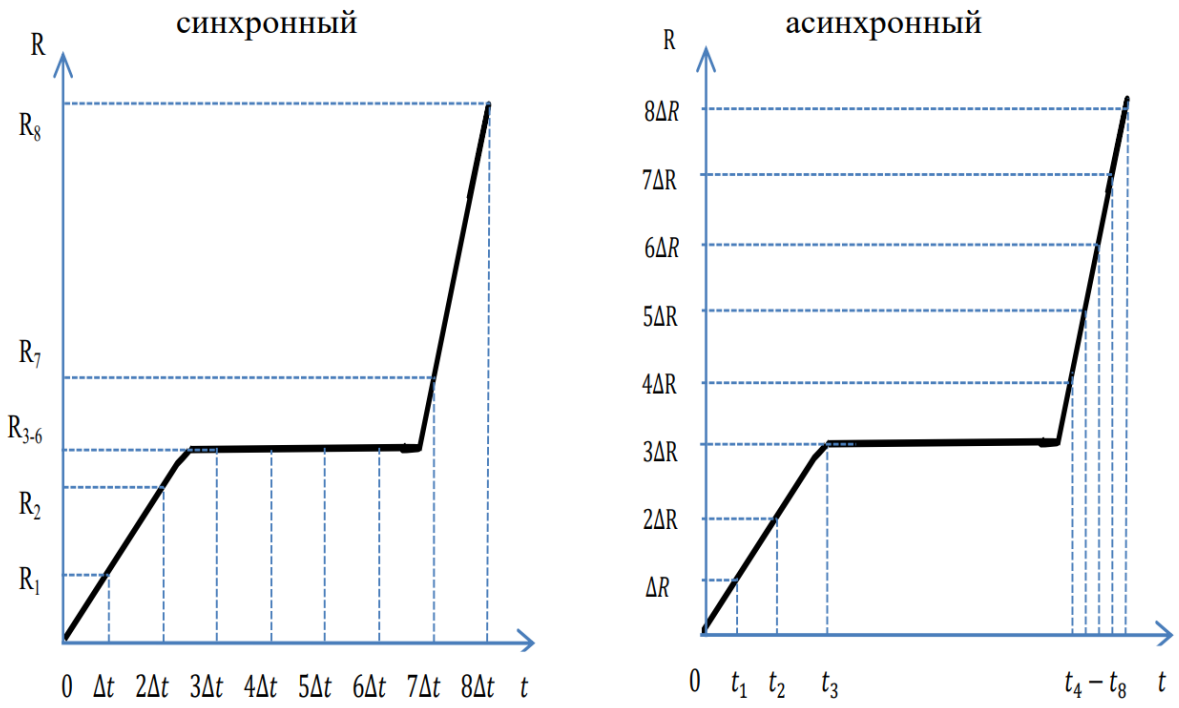

Рисунок 6 – Сравнение данных, собираемых синхронной и асинхронной системами [8]

3 Обоснование требований к потоку входных данных цифровой платформы

Одной из главных задач, решаемых при проектировании и разработке цифровой платформы мониторинга данных о ресурсопотреблении, является формирование требований к входному потоку данных, для обработки которого она будет создана. Из вышеприведенного материала очевидно, что входные потоки данных, соответствующие синхронному и асинхронному подходам, существенно отличаются друг от друга как по формату данных, по их объемам, так и по временному распределению.

Поэтому в ходе работы были проведены исследования, которые позволили дать ответ на вопрос, на какой именно подход – синхронный или асинхронный – следует ориентироваться при разработке цифровой платформы.

С этой целью были поставлены численные эксперименты и разработан метод анализа их результатов. Соответствующее программное обеспечение было разработано на языке Python, исходный код которого, а также таблицы с промежуточными и конечными результатами выложены в открытый доступ [9] с лицензией Apache-2.0 [10]. Для проведения вычислений были использованы личный компьютер и ресурсы, бесплатно предоставляемые платформой Google Colab. Были использованы реальные данные об уровне энергопотребления домохозяйствами во времени (см. п. 3.1).

3.1 Использованные датасеты

Были использованы два датасета (набора данных), содержащие информацию о потреблении электричества в домохозяйствах. Обычно подобные датасеты рассматриваются в рамках решения задач, связанных с NILM, однако нам здесь не нужна разметка данных по приборам, требуется только общий уровень потребления во времени.

Для упрощения работы была использована библиотека NILMTK [11, 12], приводящая данные к единому формату и позволяющая работать, используя

Python и Pandas. Подробнее о последующей предобработке данных написано в разделе 3.3.

3.1.1 REDD

REDD [7, 13] - Reference Energy Disaggregation Data Set. Данные собраны в 2011м году в США. В оригинальной статье говорится о 10-ти домах, в которых проводились измерения, NILMTK конвертирует 6 из них. Данные по техническим причинам получены в уже сконвертированном виде [14] из GitHub.

В этом датасете данные о потребляемой мощности сохранялись в ваттах с частотой в 1 герц. Оригинальная статья указывает, что измерения длились суммарно 119 дней, использовалось 268 датчиков и было получено более терабайта сырых данных.

Примеры данных датасета приведены на рисунках 7 и 8. На первом можно видеть повторяющиеся структуры, наложение нескольких сигналов, а также изменение минимального уровня потребления. На втором заметна область высокого потребления и вновь периодические структуры. В целом, картина напоминает примеры, показанные на рисунке 4. Графики получены при помощи файла REDD_IDEAL_data_samples_graph.ipynb, вертикальная ось обрезана по верхней границе для улучшения информативности визуализации.

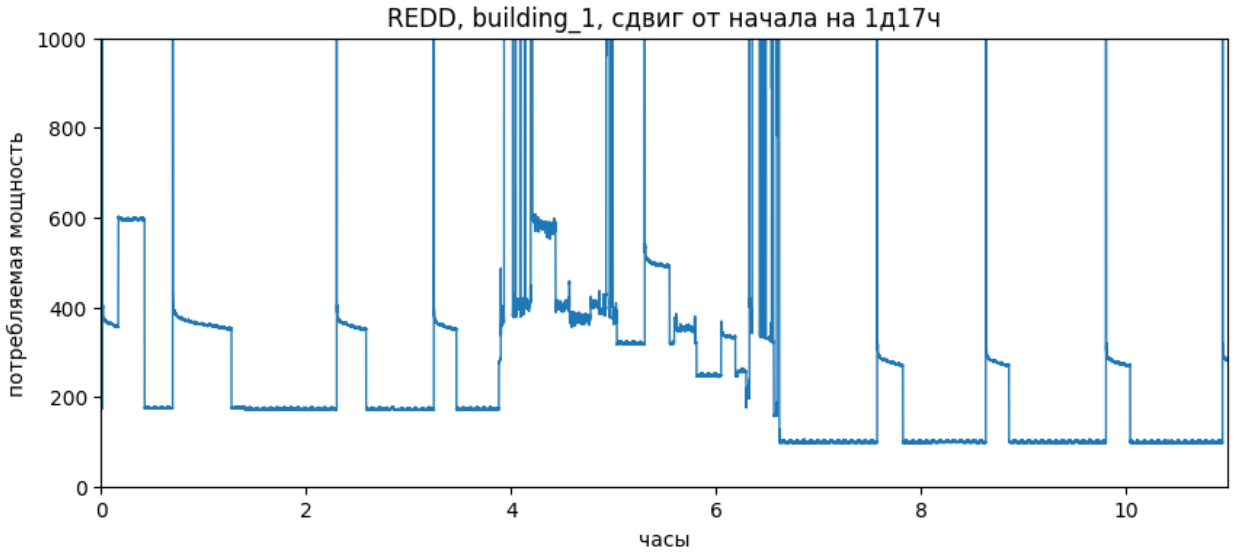

Рисунок 7 – Пример данных из датасета REDD

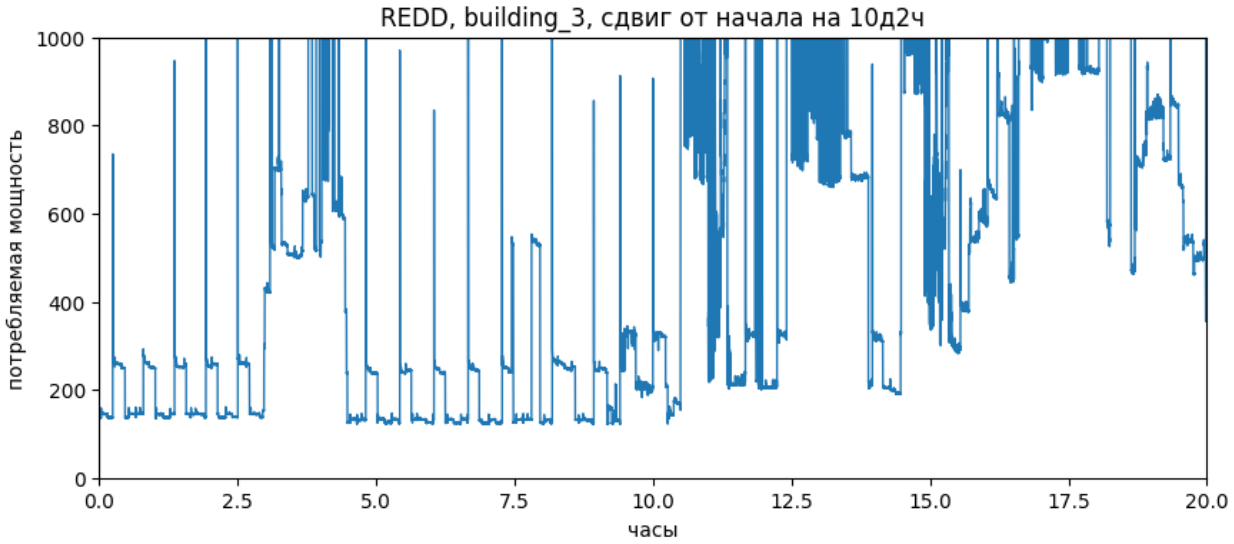

Рисунок 8 – Пример данных из датасета REDD

3.1.2 IDEAL

IDEAL [15] – второй использованный датасет, в котором представлены данные по 255 домам и квартирам Великобритании, согласно статье, сопровождавшей это исследование, среднее время измерений составило 286 дней, а всего сбор данных длился 23 месяца и закончился в 2018м году. В этом датасете доступна информация по потреблению не только электроэнергии, но и газа, а также данные по температуре в комнатах и бойлерных трубах и влажности. Аналогично REDD, в нём собраны данные о потреблении электроэнергии (мощности) с секундным интервалом, однако в IDEAL есть пропуски в данных различной длины, зачастую в несколько секунд. Также эти данные подходят для решения NILM-задач, потому что за отдельными приборами были закреплены датчики. Дадасет доступен для скачивания по ссылке [16].

Для перевода IDEAL в формат данных NILMTK потребовались дополнительные манипуляции. Рекомендуемый способ установки NILMTK (через conda) на момент выполнения работы позволяет поставить лишь версию 0.4.3, в которой отсутствует необходимый для конвертации данного датасета модуль – nilmtk/dataset_converters/ideal/convert_ideal.py. Поэтому был использован напрямую исходный код с GitHub из ветки master. NILMTK конвертирует данные 39 домов.

Примеры исходных данных датасета продемонстрированы на рисунках 9, 10 и 11. Можно видеть, что они схожи с данными датасета REDD и схематичным изображением на рисунке 4. Кроме того, на рисунке 10 можно наблюдать большой разрыв в записи данных, а на рисунке 11 – множество мелких и случай почти полного отсутствия энергопотребления.

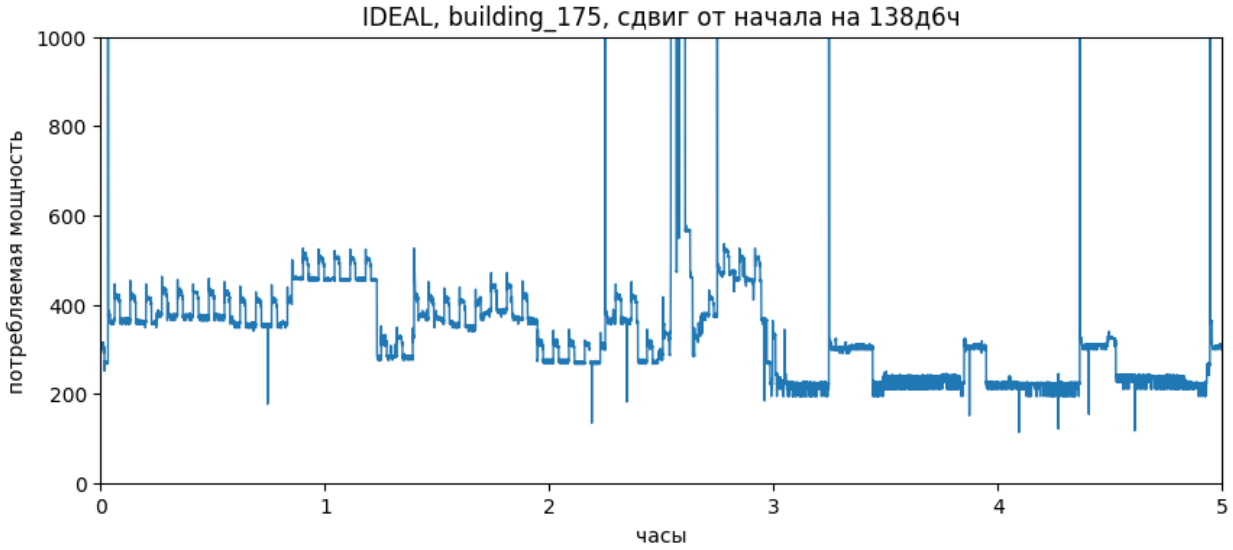

Рисунок 9 – Пример данных из датасета IDEAL

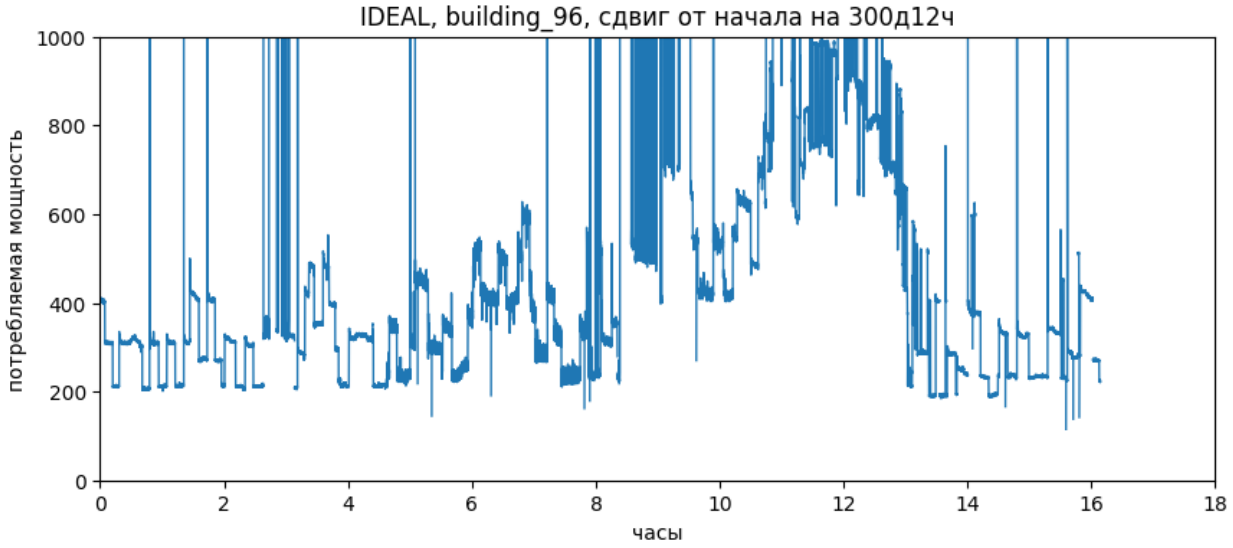

Рисунок 10 – Пример данных из датасета IDEAL

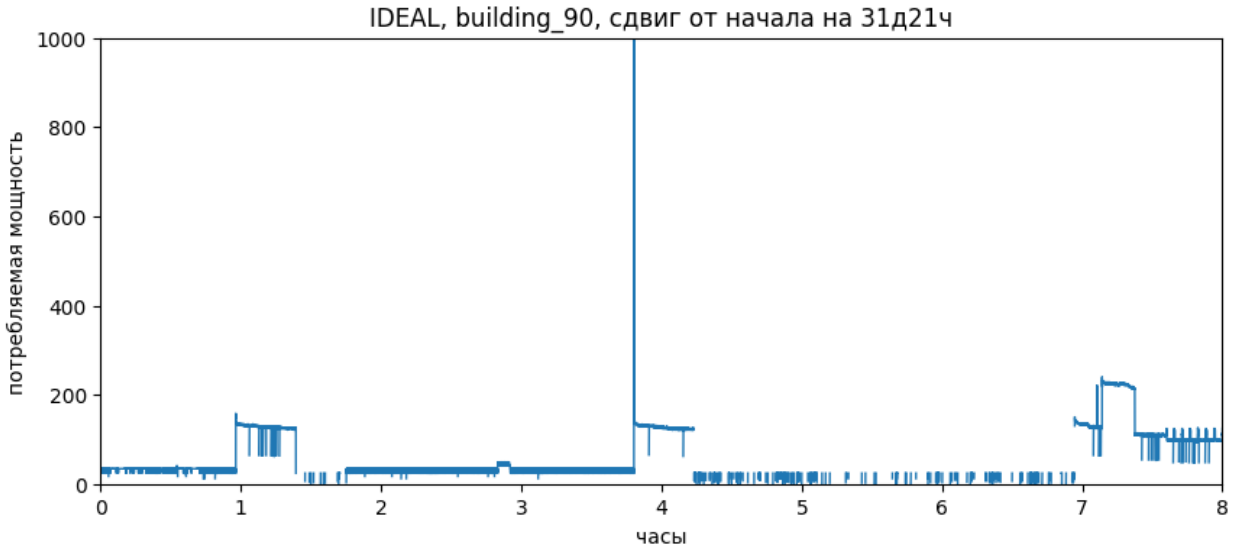

Рисунок 11 – Пример данных из датасета IDEAL

3.2 Методология

3.2.1 Метод сравнения подходов

Ниже кратко описан общий подход к производимому сравнению возможных форматов входных данных, а в последующих параграфах подробнее рассмотрены этапы проведения экспериментов и важные детали.

Информация о потреблении ресурса, по сути, представляет собой временной ряд, где значением выступает текущее значение счётчика (накопленное, далее будет обозначаться accum, accumulated) или текущий уровень потребления (мощность, power, так как для примера будут рассматриваться данные о потреблении только электроэнергии). В датасетах исходно хранятся данные о мощности, однако они легко могут быть переведены в формат, который создавали бы счётчики, накапливая показатель потреблённого ресурса.

Считаем, что исходные данные (с дискретизацией в 1 секунду) обладают высокой степенью детализации и точно описывают реальное потребление домохозяйства во времени. Поэтому принимаем их за эталон, в котором содержится наибольшее количество информации об описываемой модели потребления и с которым далее проводим сравнение временных рядов с меньшей детализацией.

Далее имитируется работа синхронного счётчика – сохраняются данные, которые бы он передал при данном потреблении. Аналогично генерируются показания, которые бы передал асинхронный датчик в тех же условиях. В обоих полученных рядах детализация ниже, чем в эталонном. Процесс генерации менее детализированных рядов путём имитации работы сенсоров далее будем называть огрублением.

Затем проводится сравнение сгенерированных рядов с эталонным. Ряд, оказавшийся «ближе» к эталонному, считается более качественно отображающим реальную картину, а значит он несёт больше полезной информации и может быть более качественно проанализирован. Для сравнения близости эталонного и сгенерированного рядов используются разработанные метрики, описанные в следующем параграфе.

При имитации работы датчиков используются параметры их настройки – Δt и Δr для синхронного и асинхронного подхода соответственно. Между ними задаётся связь, математически выражающаяся в коэффициенте пропорциональности. Далее она называется ограничением на параметры огрубления рядов. Она необходима для корректного анализа форматов входных данных и выявления из них более подходящего для работы платформы – подробнее в 3.2.3.

3.2.2 Использованные метрики

Для сравнения синхронного и асинхронного подходов к формированию входных данных платформы были реализованы две метрики accumulated_distance и power_distance. Обе вычисляются по формуле

$$D = \int_0^T \left(f(t) - \hat{f}(t)\right)^2 dt$$

$\hat{f}(t)$ – эталонные данные с шагом в 1 секунду;

$f(t)$ – сгенерированные (синхронные или асинхронные) данные, более грубые;

T – полная продолжительность эталонного временного ряда.

Чем меньше D, тем ближе f и $\hat{f}$.

Далее рассмотрим подробнее, как вычислялись эти метрики и в чём их отличие.

3.2.2.1 accumulated_distance

Для обоих наборов данных (эталонных и огрублённых) строится непрерывная кусочно-линейная функция объема потребленного ресурса от времени (accum). Разность этих функций (на рисунке заштриховано серым и показано отдельно зелёным) так же является непрерывной кусочно-линейной функцией, и её квадрат можно проинтегрировать точно (математически, а не численно). Использование тут квадрата разности довольно стандартный подход, например функция потерь MSE в машинном обучении вычисляется аналогичным образом, так существенное отклонение вносит больший эффект в общую ошибку, нежели при выборе абсолютного значения разности. На рисунке 12 отображены эталонные orig и сгенерированные огрублённые gen данные (асинхронные, так как этот вариант чуть более демонстративен – в синхронном случае шаг по времени был бы постоянен и отсечки на временной шкале совпадали с эталонными). Значения синтетические, подобранные для наглядного изображения так, чтобы разница эталонных и сгенерированных данных была лучше заметна.

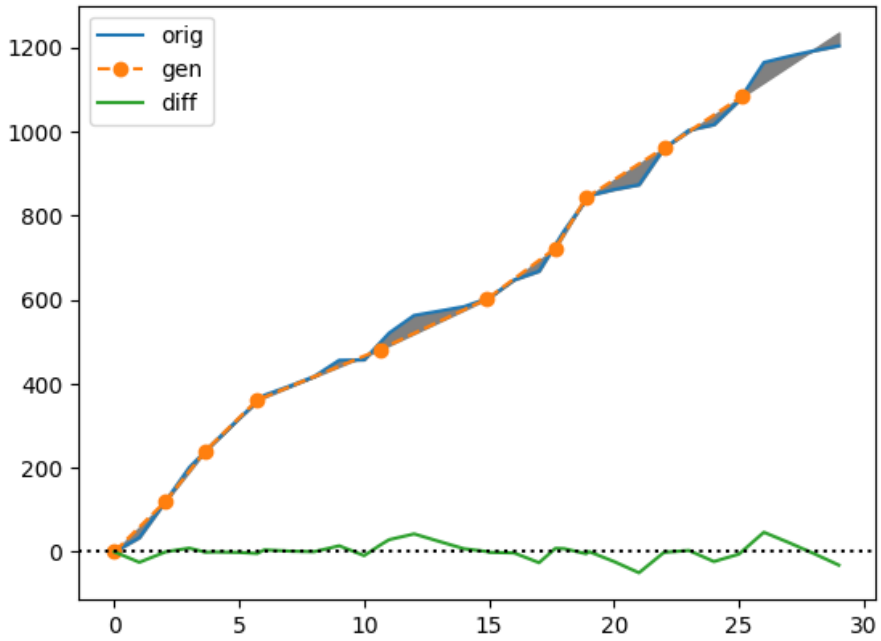

Рисунок 12 – Демонстрация метода вычисления метрики accumulated_distance

3.2.2.2 power_distance

В этом случае сравнивается информация о потребляемой мощности (power). Эталонные данные о мощности берутся из датасета.

Огрублённые данные в power формате вычисляются следующим образом. Данные о потреблённой энергии (accum) огрубляются обычным способом, описанным в 3.2.1, а затем от получаемой непрерывной кусочно-линейной функции берётся производная по времени. Таким образом получаются данные о мощности, восстановленные из данных, собранных счётчиком, тем же способом, каким это бы происходило внутри платформы.

Полученные функции являются кусочно-постоянными, так же, как и их разница. Это позволяет точно проинтегрировать расхождение огрублённых данных с эталонными. На рисунке 13 отображены эталонные orig и сгенерированные огрублённые gen данные. Значения синтетические, подобранные для наглядного изображения так, чтобы разница эталонных и сгенерированных данных была лучше заметна.

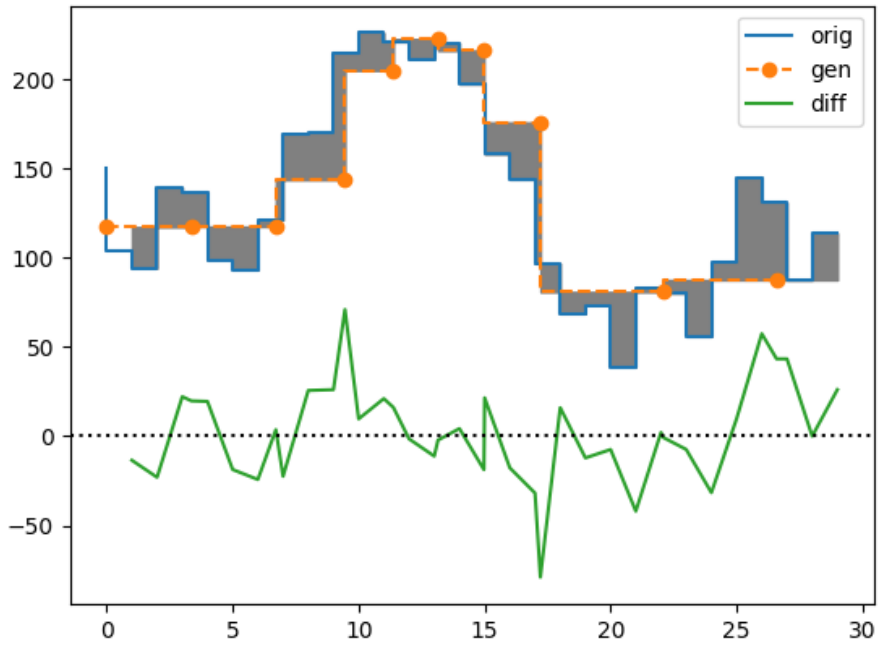

Рисунок 13 – Демонстрация метода вычисления метрики power_distance

3.2.3 Ограничения на параметры огрубления рядов данных

Как упоминалось ранее, для корректного сравнения синхронного и асинхронного подходов к сбору данных для платформы необходимо соблюдать некоторые ограничения. Например, очевидно, что будет неразумным сравнивать данные с интервалом в 1 минуту и данные с интервалом в 1мл потраченной воды. В первом случае записанных данных будет гораздо меньше, соответственно они будут менее детализировано отображать действительность и результаты их анализа будут не такими качественными. Так как стоит задача определить более подходящий метод сбора данных, необходимо сформулировать какие-то принципы, в рамках которых сравнение разных подходов будет целесообразно.

Далее рассмотрим, какие два предложенных ограничения были использованы в рамках настоящего исследования. При использовании обоих ограничений ряд синхронных данных генерируется одинаково – при помощи метода utils.preprocessing.generate_sync_signals с параметром time_delta – интервалом Δt в секундах для огрублённых данных. Алгоритм генерации асинхронных данных зависит от выбранного ограничения, которое определяет порядок расчёта параметра Δr для асинхронного сенсора.

3.2.3.1 Сравнение при одинаковом среднем количестве данных

Рассмотрим модельную ситуацию, когда синхронный и асинхронный подходы отправили одинаковое количество данных в один и тот же временной интервал. В этом случае, если мы получим, что один из форматов данных обеспечивает б*о*льшую информативность (т.е. меньшее значение D при вычислении метрики), это будет означать, что, используя этот подход, можно добиться снижения количества передач данных и уменьшения разряда батареи, не теряя в качестве извлекаемой информации.

Для генерации асинхронного ряда в описанном выше ограничении используется метод utils.preprocessing.generate_async_signals_by_point_count с параметром point_count. Этим параметром определяется количество точек в ряде, его значение берётся таким же, как у сгенерированного синхронного ряда. Выполняется соотношение point_count $= \frac{T}{\Delta t} = \frac{R}{\Delta r}$, где T и R это продолжительность эталонного временного ряда и полное количество потреблённого ресурса. Таким образом вычисляется значение параметра Δr, обеспечивающее равное количество данных в синхронном и асинхронном рядах.

3.2.3.2 Сравнение в ограничении максимальной загрузки канала

Вторая модельная ситуация, использованная для обоснования выбора формата входного потока, состоит в том, что мы предполагаем, что оба подхода при работе создают одинаковую максимальную нагрузку на каналы передачи данных. Возможность определения нагрузки, генерируемой датчиками, может быть необходима при проектировании канала связи для обеспечения эффективной передачи данных без потерь. При работе синхронного счётчика нагрузка на канал одинаковая в разное время суток, так как сигналы посылаются с постоянной периодичностью Δt. Для асинхронного же счётчика фиксированным параметром будет Δr, поэтому нагрузка на канал передачи данных будет пропорциональна потребляемой мощности, которая непостоянна в течении дня.

На рисунках 14 и 15 приведено распределение средней потребляемой мощности по времени для использованных в исследовании датасетов. Данные графики получены в ходе работы с файлом power_usage_cycle.ipynb. Хорошо видны пики активности, которые, как и ожидалось, приходятся на утро, когда большинство жителей собираются на работу и учёбу, и вечер, когда они возвращаются и занимаются домашними делами.

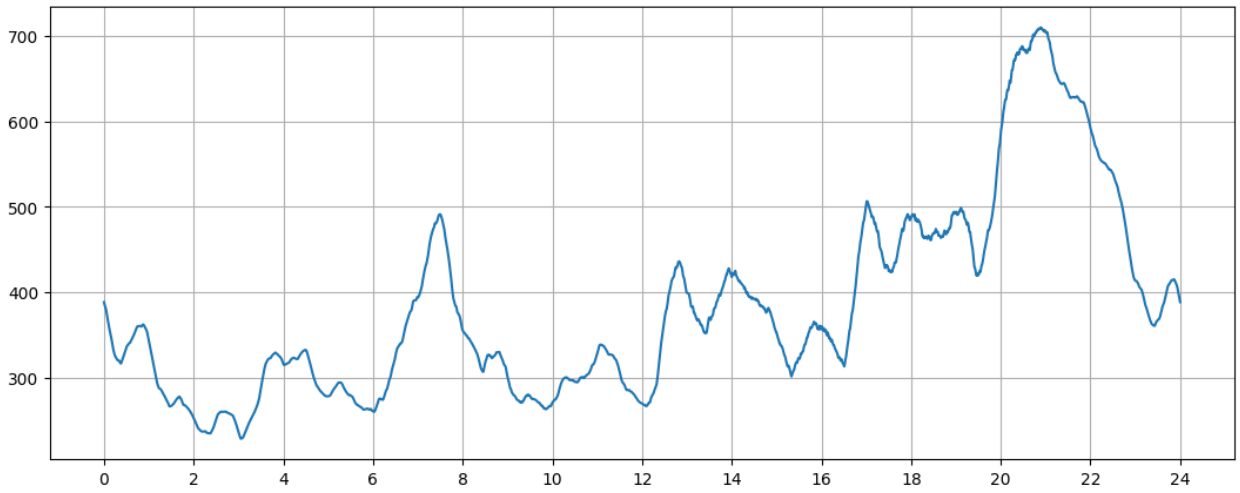

Рисунок 14 – Распределение уровня потребления электроэнергии по часам, датасет REDD

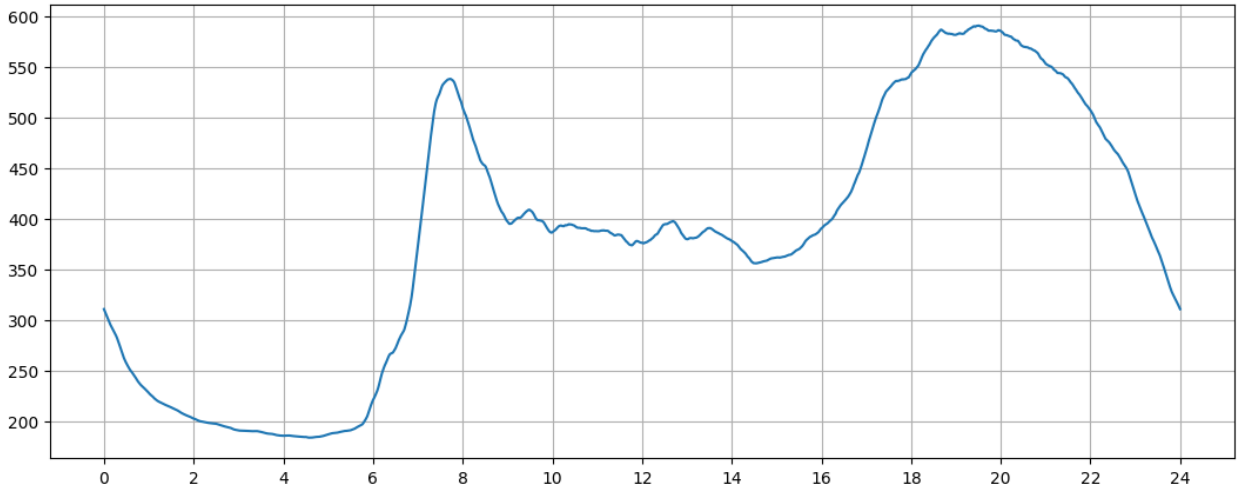

Рисунок 15 – Распределение уровня потребления электроэнергии по часам, датасет IDEAL

Как упоминалось выше, количество данных, генерируемое асинхронным счётчиком, пропорционально потребляемой мощности. В таком случае, для установления одинаковой максимальной нагрузки на канал за

коэффициент пропорциональности между параметрами генерации огрубленных рядов должно быть взято значение мощности, достигающееся в момент пикового потребления. Это значит, что б*о*льшую часть времени асинхронный счётчик будет передавать меньше данных, чем синхронный. Тогда достижение двумя рядам одинаковой близости к эталону будет означать, что асинхронный способ сбора информации является более предпочтительным, так как снижает количество данных, и следовательно, количество передаваемых сигналов, не теряя в качестве извлекаемой информации.

В этом случае для генерации асинхронного ряда в описанном выше ограничении используется метод utils.preprocessing.generate_async_signals с параметром resource_delta. Он вычисляется следующим образом: $\Delta r = \Delta t \cdot P$, где P – пиковое значение мощности потребления, на основе приведённых выше графиков было принято за 700 ватт.

3.3 Обработка данных

Для дальнейшего анализа в рамках настоящей работы полученные при помощи NILMTK данные из датасетов REDD и IDEAL были упрощены – отброшена ненужная информация: разметка по времени работы устройств, метаданные о количестве комнат, жильцов и так далее. Так же были удалены «мусорные» измерения – null, ±∞, отрицательные значения. Для этого использовалась функция utils.data_reading.clean_and_store_data(), сохраняющая предобработанные данные в Hierarchical Data Format файл (.h5).

Далее для работы с данными в экспериментах были произведены следующие преобразования. Были выделены куски данных одинаковой длины (далее – периоды). Для этого была реализована функция utils.preprocessing.get_stable_periods(). Эта функция принимает в качестве аргументов duration и max_gap – требуемая длительность периода и наибольший дозволяемый промежуток времени (подряд), в который нет измерений. Функция «нарезает» все имеющиеся ряды в датасете, согласно

указанным параметрам на непересекающиеся периоды. Чем меньше max_gap, тем достовернее данные, полученные в периоде, однако количество полученных периодов становится меньше.

Были использованы пары параметров, указанные в таблице 1 (duration указан в сутках, max_gap – в секундах). Также в таблице отображено количество полученных периодов для каждой пары параметров. При этом стоит отметить, что данные могут дублироваться при попадании в разные периоды. Например, одни и те же сутки данных с одного домохозяйства могут оказаться и в трёхдневном периоде с max_gap=15, и в пятидневном с max_gap=10.

Таблица 1 – Параметры периодов и их количество

duration	max_gap	количество периодов	
		redd	ideal
30	300	0	20
20	300	0	70
15	300	0	112
10	300	1	293
20	180	0	31
15	180	0	81
10	180	0	197
15	90	0	32
10	90	0	126
5	90	2	589
14	60	0	20
10	60	0	75
7	60	0	218
5	60	2	467
10	30	0	47
7	30	0	121
5	30	1	345
3	30	6	1015
7	15	0	61
5	15	1	200
3	15	6	720
5	10	1	47
3	10	6	302
1	10	54	3377
5	5	1	7
3	5	5	40
1	5	49	936
5	1	1	0
3	1	5	0
1	1	49	0

После «нарезания» данных пропущенные значения интерполируются кубическим сплайном при помощи scipy.interpolate.CubicSpline. Отрицательные интерполированные значения заменяются на нули. После этого данные сохраняются в директорию power_periods_storage – это данные о потребляемой мощности. Также проводится аккумулирование этих данных во времени для получения возрастающей последовательности (именно в таком

виде хранят данные о потреблении традиционные счётчики), которая показывает потреблённое количество энергии в джоулях (что легко может быть переведено в привычные киловатт-часы делением на 3600*1000). Информация в этом виде сохраняется в директорию periods_storage. Для краткой сводки о проведённых для предобработки действиях можно посмотреть файл periods_preprocessing_flow.py. Схематично весь процесс подготовки данных и их использования при проведении экспериментов показан на рисунке 16.

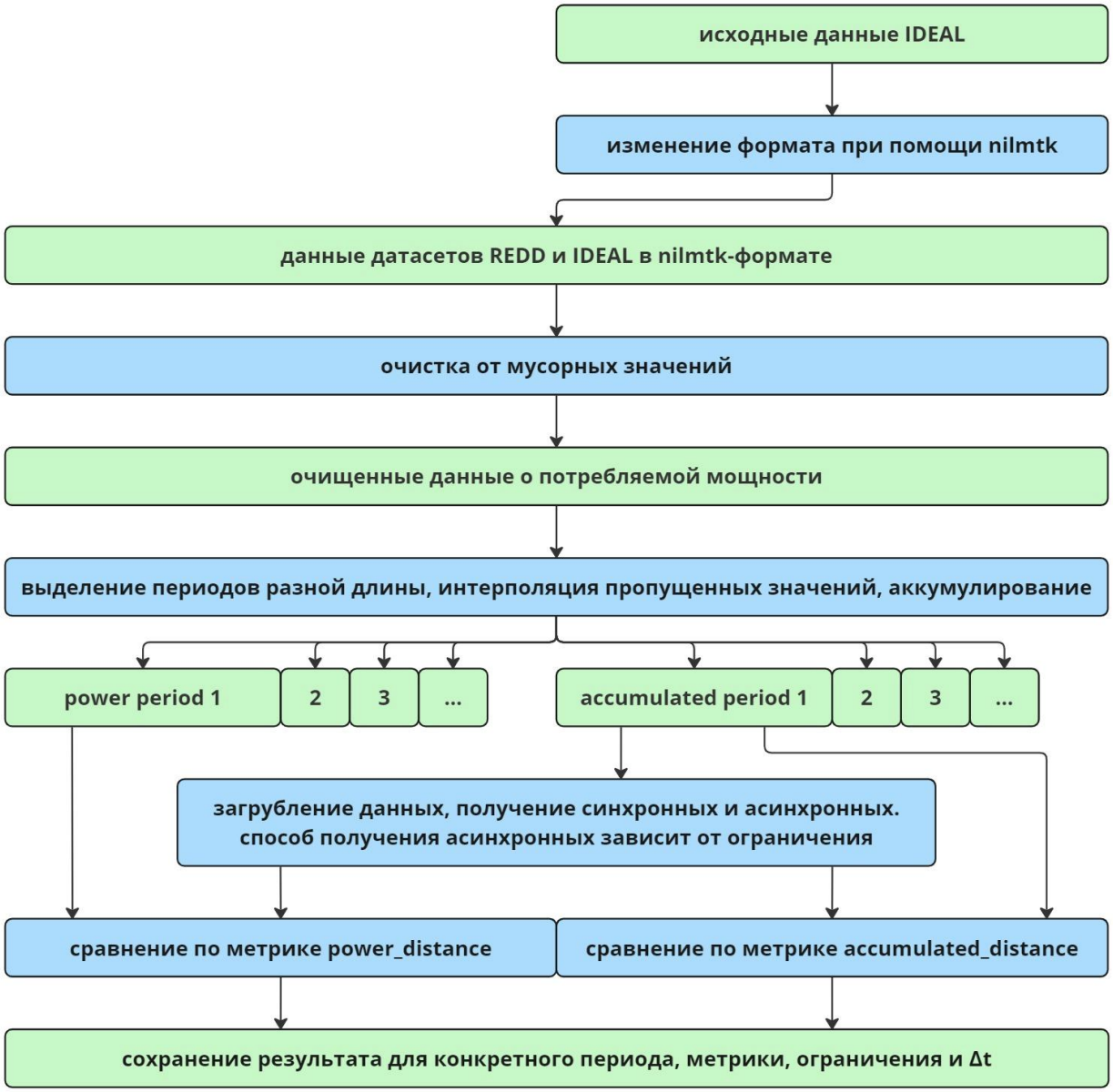

Рисунок 16 – Схема предобработки данных и проведения экспериментов

3.4 Проведённые эксперименты

Для периодов, описанных в предыдущем параграфе, были проведены эксперименты двух типов (отличающиеся использованными метриками – accumulated_distance и power_distance, описанными в разделе 3.2.2) в условиях двух ограничений (параграф 3.2.3).

В модуле main.py была настроена возможность параметрического запуска отдельных или нескольких экспериментов из командной строки. Их вызов осуществлялся при помощи методов utils.experiment.run_experiment_accum и utils.experiment.run_experiment_power. Параметром запуска было значение time_delta – шаг огрубления Δt синхронных данных.

Для всех пар из таблицы 1 (конечно, если в датасете нашлись такие периоды) были проведены по 4 эксперимента (2 метрики в 2х ограничениях) с time_delta равными 3 и 60 секунд. Помимо этого, для пар, указанных в таблице 2, было проведено расширенное количество экспериментов с дополнительными значениями Δt.

Таблица 2 – Параметры расширенных экспериментов

duration	max_gap	дополнительные time_delta
30	300	5, 10, 15, 20, 30, 120, 240, 600
10	60	
5	5	
1	1	

3.5 Получение результатов

Результаты запуска экспериментов сводились в таблицы, а затем объединялись для последующего анализа.

Каждый запуск отдельного эксперимента (пример параметров приведён в таблице 3) создавал excel-файл с полученными результатами. Каждая строка таблицы – это период, для которого была вычислена заданная метрика для

двух огрублений – синхронного и асинхронного. Упрощенно шапка таблицы с результатами имела вид, отображённый в таблице 4.

Таблица 3 – Пример параметров эксперимента

параметр	значение
dataset	redd
time_delta	3
duration	30
max_gap	300
метрика	accumulated_distance
ограничение	by_point_count

Таблица 4 – Упрощенная шапка таблицы результатов

period_id	duration	max_gap	time_delta	sync_dist	async_dist

После окончания всех экспериментов полученные таблицы были объединены при помощи модуля concat_excels.py. В результате были получены следующие файлы с двумя страницами (accumulated_distance и power_distance), содержащие в себе все результаты проведённых вычислений метрик:

exp_results_by_point_count/united_redd.xlsx

exp_results_by_point_count/united_ideal.xlsx

exp_results_by_peak_consumption/united_redd.xlsx

exp_results_by_peak_consumption/united_ideal.xlsx

В этих файлах был добавлен столбец sync_async_ratio, вычисляемый как результат деления sync_dist на async_dist. Далее при помощи модуля results_analyzing.py были получены два файла processed_results_point.xlsx и processed_results_consum.xlsx для каждого ограничения. Структура этих файлов такова: 16 страниц с названиями вида «<dataset>_<метрика>__by_<grouping_param>» – два датасета, две метрики и

четыре параметра группировки. Группировка проводилась по номеру домохозяйства в датасете, а также по параметрам duration, max_gap, time_delta.

Рассмотрим наполнение этих страниц на примере ideal_power__by_max_gap для ограничения на равную максимальную загрузку канала – таблица 5.

Таблица 5 – Файл processed_results_consum.xlsx страница ideal_power__by_max_gap

max_gap	count	geometric_mean	win_percentage
5	2022	2,14571	91,79031
10	7452	2,107434	92,95491
15	1962	2,030443	94,90316
30	3056	2,071491	95,87696
60	2160	1,957463	96,15741
90	1494	2,031202	96,98795
180	618	2,003327	97,73463
300	1150	1,997048	98,86957

Первый столбец показывает значение параметра, по которому производилась группировка. count – количество замеров на отдельных периодах, которые были агрегированы в эту строку. geometric_mean – среднее геометрическое sync_async_ratio для агрегированных строк. win_percentage – в каком проценте случаев sync_async_ratio оказался больше единицы, то есть асинхронный ряд данных оказался по метрике ближе к эталонному.

Анализируя группировки по параметрам duration, max_gap и time_delta, можно сделать качественные выводы – это и будут результаты экспериментальной части настоящей работы. Они представлены в следующем параграфе.

3.6 Анализ результатов

На основе таблиц, описанных в предыдущем параграфе, были построены столбчатые диаграммы для демонстрации результатов – рисунки 17, 18, 19 и 20. В названии каждой диаграммы указано, результаты экспериментов с какими параметрами она отражает – название датасета, вид использованной

метрики, параметр для группировки и ограничение. Каждый столбец отображает одно из значений параметра группировки, подписанное под столбцом. Синим цветом отображена доля периодов, для которых асинхронный способ оказался более информативным, то есть ближе к эталонному по вычисляемой метрике. Над столбцом указано количество периодов, попавших в эту группу, этот параметр стоит принимать во внимание, так как полученный результат может быть непоказателен при маленьком размере выборки.

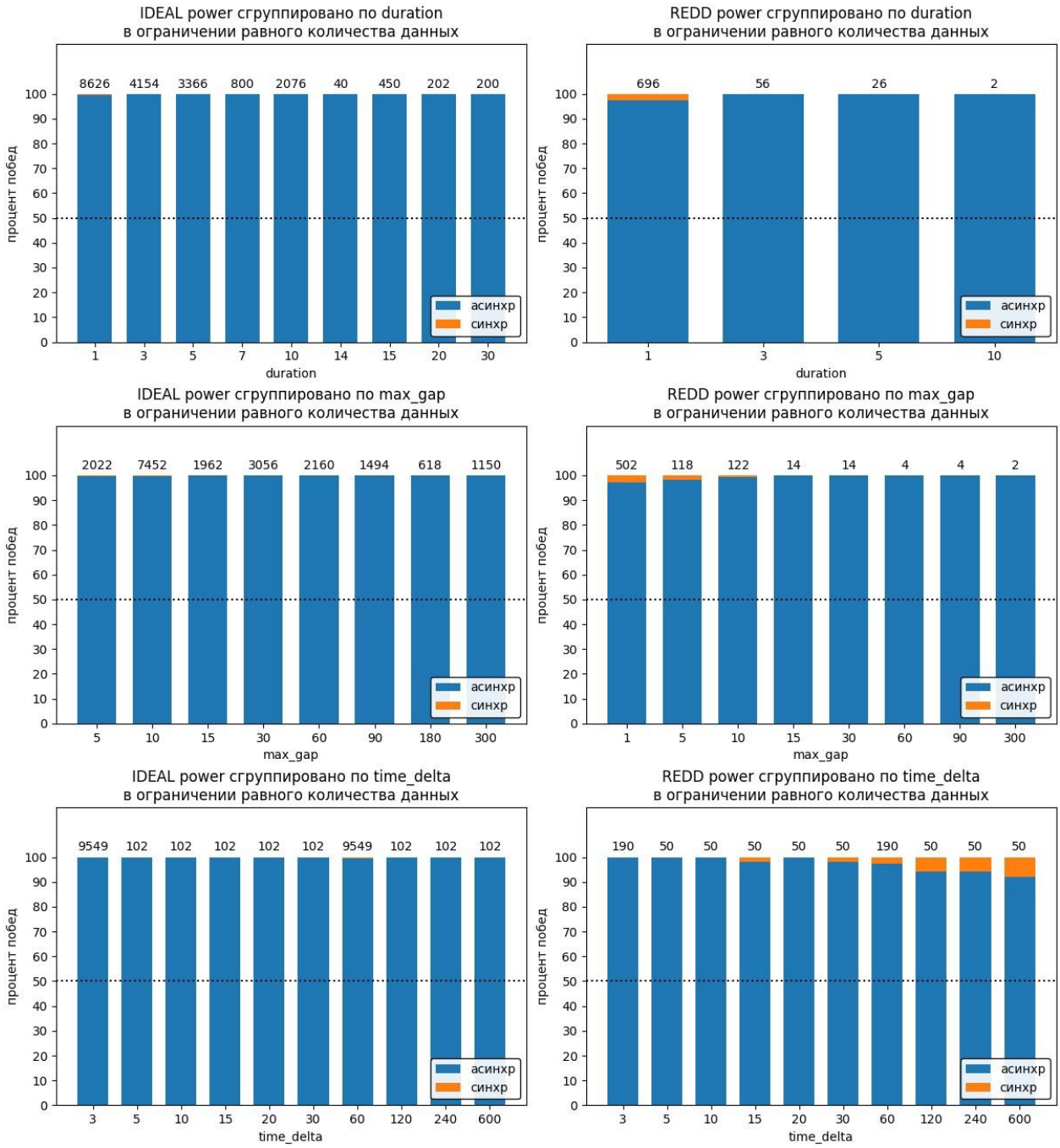

Рисунок 17 – Результаты экспериментов по метрике power_distance в ограничении равного количества данных

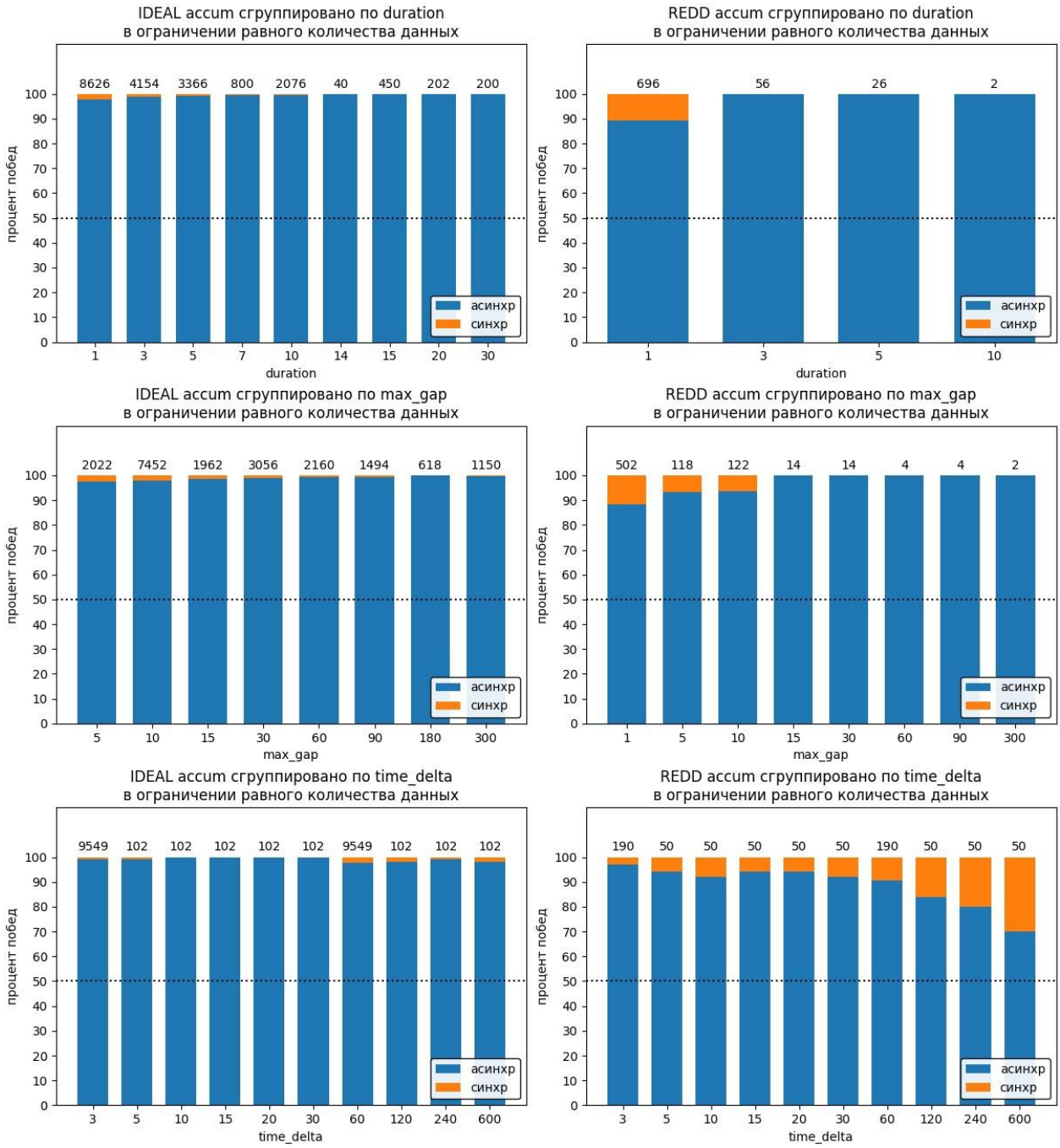

Рисунок 18 – Результаты экспериментов по метрике accumulated_distance в ограничении равного количества данных

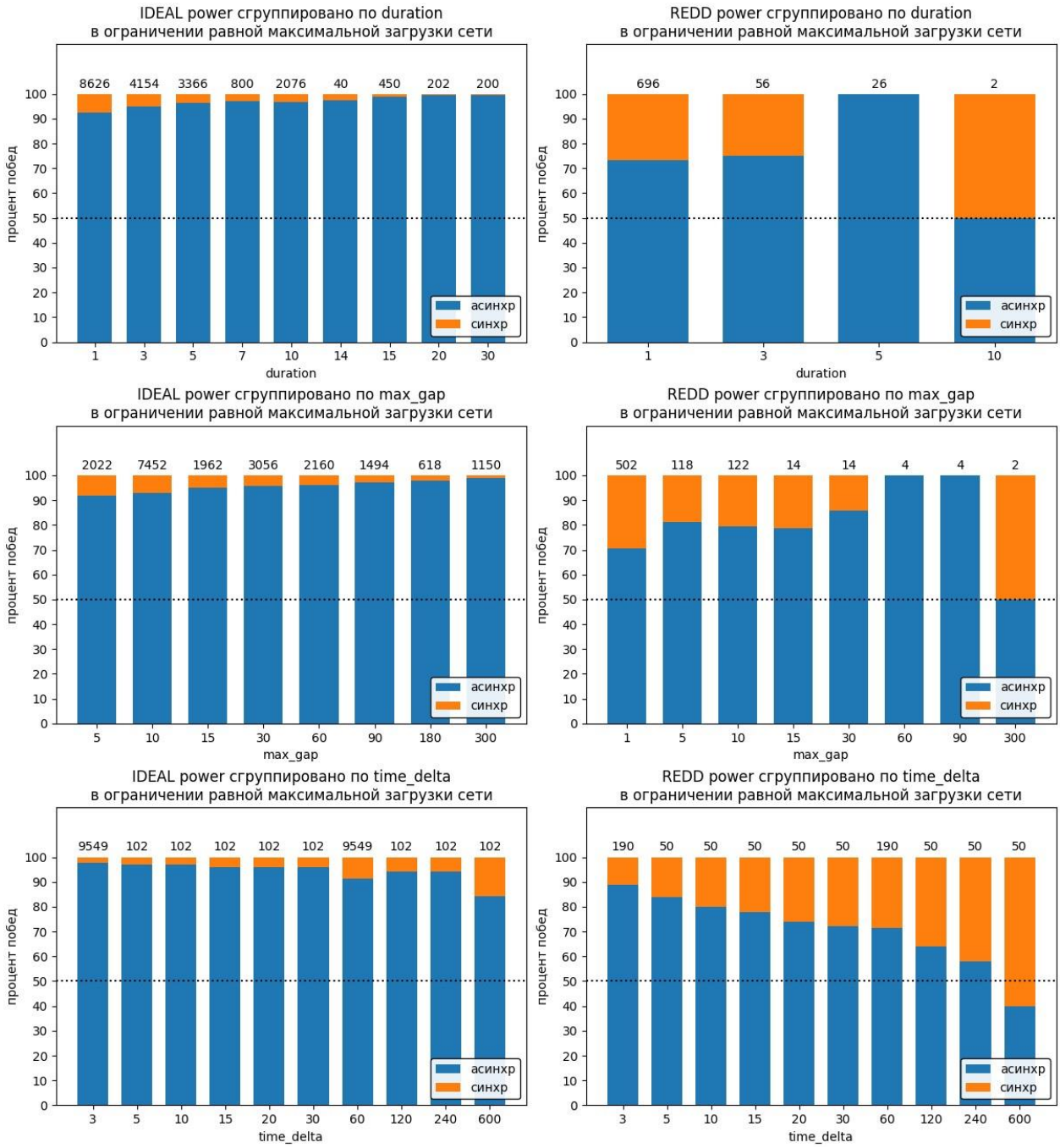

Рисунок 19 – Результаты экспериментов по метрике power_distance в ограничении равной максимальной загрузки сети

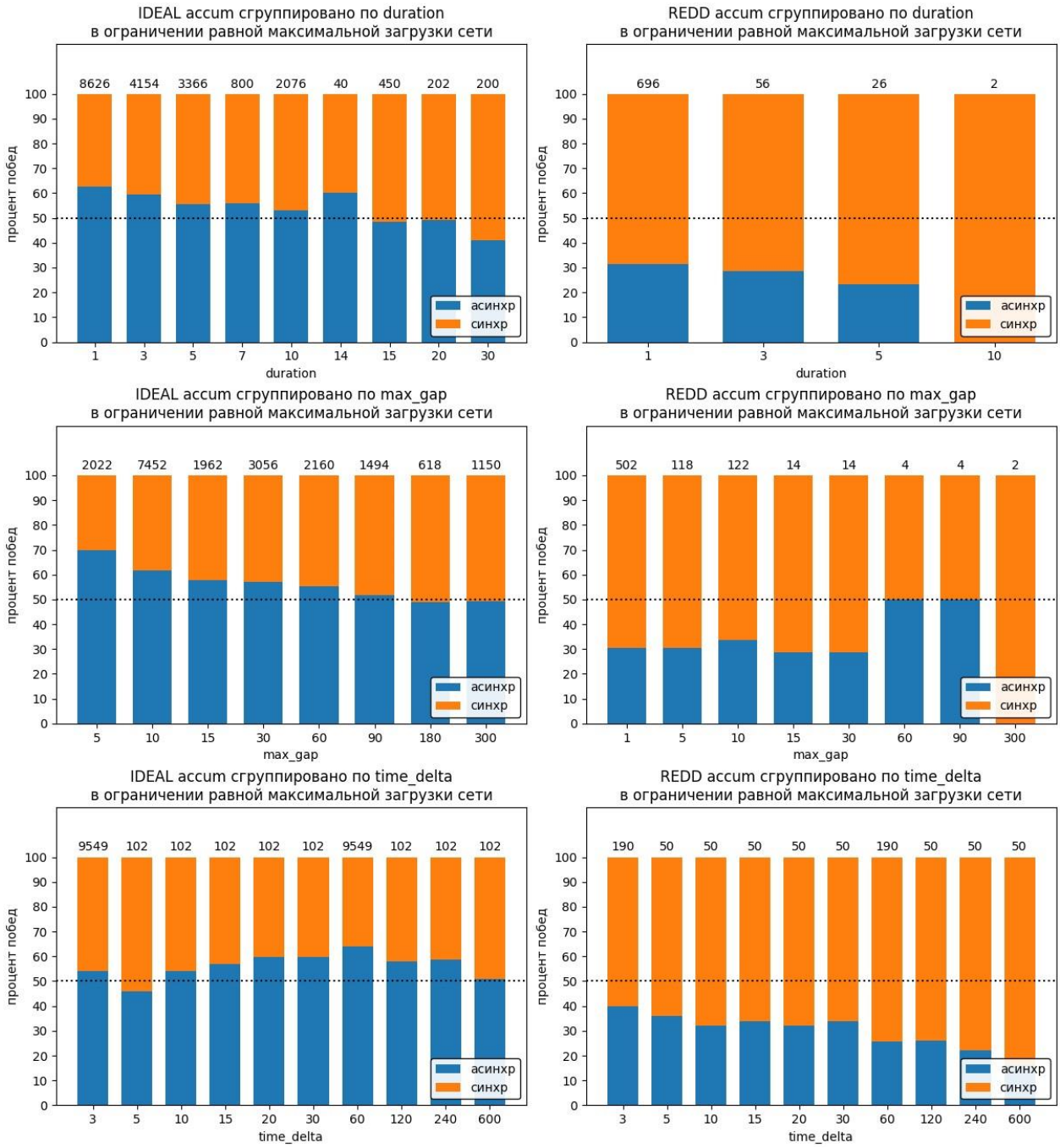

Рисунок 20 – Результаты экспериментов по метрике accumulated_distance в ограничении равной максимальной загрузки сети

Можно наглядно видеть, что в случае первого ограничения (на количество точек в ряде), асинхронный подход к формату входных данных демонстрирует полное превосходство над классическим синхронным. При любых исследованных значениях параметров, для обоих датасетов и обеих метрик асинхронный подход «выигрывает» существенно чаще, в большинстве случаев доля его побед не опускается ниже 95%.

При рассмотрении ограничения на максимальную нагрузку сети передачи данных преимущество асинхронного способа генерации данных уже не такое явное. Выборка же с метрикой accumulated_distance для датасета REDD и вовсе показывает, что примерно в 70% случаев более подходящим был синхронный способ. Однако, напомним, что данное ограничение существенно более жёсткое по отношению к асинхронному формату данных, подробнее это было разобрано в параграфе 3.2.3.2.

Можно провести и сравнение результатов, характерных для рассматриваемых метрик. Легко видеть, что асинхронный метод лучше себя проявил при использовании метрики power_distance. Высокие результаты по данной метрике – это отличный знак, что асинхронный формат данных лучше подходит для дезагрегации каналов, ведь при решении задач NILM для определения приборов используют как раз данные о мощности потребления электроэнергии, потому что в них удобнее различать паттерны, характерные для разного вида оборудования.

Кроме этого, можно проследить общие закономерности, характерные для полученных результатов. Так, отчётливо заметно, что на датасете IDEAL асинхронный подход показывает лучшие результаты, чем на REDD. Это может быть вызвано меньшим размером датасета, а также существенно меньшим количеством выделенных длинных периодов. Однако, возможно, такой эффект объясняется спецификой датасетов, ведь данные разные и было бы странно ожидать полного совпадения результатов.

Также можно видеть зависимость от параметров группировки данных. Например, при росте time_delta доля побед асинхронного подхода снижается, это прослеживается для всех четырёх групп графиков. Однако использование больших периодов дискретизации при сохранении данных о потреблении на практике невозможно, так как они становятся неинформативны при решении задачи дезагрегации.

Стоит заметить, что параметр max_gap в общем зависит от duration, ведь логично, что будет найдено мало длинных периодов с жёсткими требования к качеству исходных данных. Именно руководствуясь таким предположением, и подбирались пары параметров в таблице 1. Поэтому зависимости результатов от них схожи. Кроме особого случая, представленного на рисунке 20, с ростом duration и max_gap доля побед асинхронного подхода возрастает. Значит при долгосрочном использовании система, использующая этот формат данных, будет более эффективна. Хотя не стоит забывать, что увеличение max_gap ведёт к большему количеству интерполированных данных в периоде, а значит надёжность таких рядов немного ниже, чем у тех, где искусственно добавленных значений меньше.

Однако встаёт вопрос, почему же тогда при рассмотрении экспериментов с метрикой accumulated_distance в ограничении равной максимальной загрузки сети наблюдает противоположная зависимость – с ростом duration/max_gap асинхронный подход показывает результаты всё хуже. Предположения следующие.

Во-первых, при таком сочетании метрики и ограничения асинхронная система в целом показывает себя хуже всего. А во-вторых, в этом случае могли оказать влияние выходные дни. В выходные профиль потребления энергии может сильно отличаться от среднего. Выходных дней меньше, поэтому при рассмотрении суточных промежутков побед асинхронного метода будет больше. Однако, когда рассматриваются недельные промежутки, если в выходные дни эффективность асинхронной системы существенно ниже и это оказывает ощутимое влияние, то уже весь период будет «проигран». Подтверждение этого предположения или его опровержение может быть одним из объектов дальнейших исследований этой области.

Хотелось бы также добавить, что наиболее ценным, с точки зрения принятия решения об используемом подходе к формированию входных данных платформы, является набор графиков на рисунке 19. Тут рассмотрено

более жёсткое ограничение и более релевантная метрика. На представленных графиках можно видеть значительное превосходство асинхронного подхода.

Таким образом, по результатам проведенных исследований необходимо сделать вывод о том, что при разработке цифровой платформы мониторинга данных о ресурсопотреблении домохозяйств целесообразно исходить из того, что входной поток данных будет поступать на вход платформы в асинхронном виде. То есть, данные будут приходить не через равные промежутки времени, а в моменты, соответствующие прохождению через размещенные в домохозяйствах сенсоры (датчики) контрольного объема энергоресурсов.

4 Разработка архитектуры и программного обеспечения цифровой платформы

Разработанная платформа основана на лямбда-архитектуре, была реализована на языке Java в экосистеме инструментов для работы с большими данными от Apache. Выложена в открытый доступ [17] с лицензией Apache-2.0 [10]. Обоснование выбора такой архитектуры представлено ниже в пп. 4.1 и 4.2.

4.1 О лямбда-архитектуре

При построении архитектуры платформы был использован паттерн обработки данных, называемый лямбда-архитектурой [18, 19, 20]. Главной концепцией этого паттерна является обработка входящих данных в двух разделяемых потоках. Принято говорить о трёх составных частях такой системы: speed (realtime) layer, batch layer, service layer – слой обработки данных реального времени, слой пакетной (или батч) обработки, сервисный слой соответственно. Пример схемы реализации лямбда-архитектуры приведён на рисунке 21.

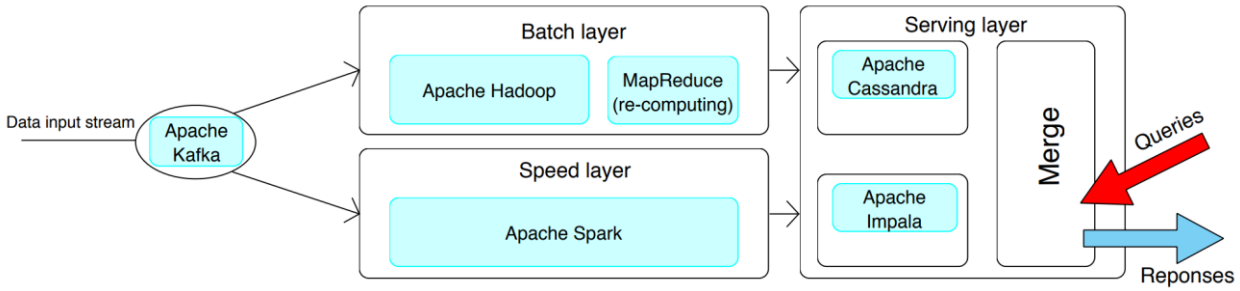

Рисунок 21 – Лямбда-архитектура [20]

Как говорилось выше, поступающие данные одновременно идут в два потока – в batch-слой и в speed-слой. В последнем они обрабатываются «на лету», при этом обработка может отличаться от той, что производится в batch layer. В пакетном же слое данные сохраняются в хранилище в сыром виде для последующей обработки большими пакетами (кусками, партиями, батчами). Результаты обработки батчей обычно называют batch-view.

Сервисный слой служит для предоставления интерфейса другим системам к результатам обработки данных. Например, (если этого требует специфика решаемой задачи) сервисный слой может объединять batch-view с результатами, полученными в realtime-слое от тех данных, что ещё не были упакованы и обработаны в пакеты – таким образом может поддерживаться максимальная актуальность данных.

Основными преимуществами лямбда-архитектуры являются гибкость манипулирования данными, надёжность такой системы, а также лёгкая масштабируемость.

Альтернативой лямбде является каппа-архитектура. В ней отсутствует batch-слой и поток обработки получается единый. Это упрощает процесс обработки, но создаёт и свои недостатки. Эффективность системы снижается, так как батч-обработка может выполняться более оптимизированно. Реакция на новые события также снижена, потому что требуется сначала обработать поступившую информацию. Кроме того, страдает надёжность системы из-за отсутствия хранилища сырых данных. На рисунке 22 продемонстрирована возможная схема реализации каппа-архитектуры.

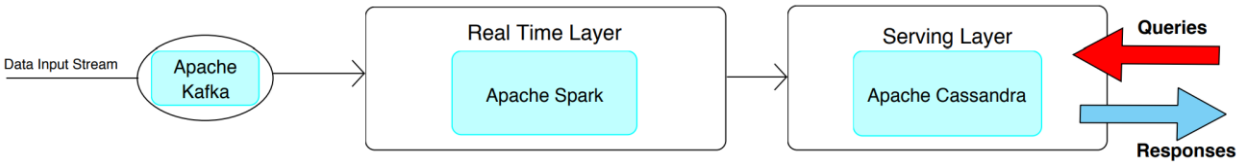

Рисунок 22 – Каппа-архитектура [20]

4.2 Архитектура цифровой платформы и используемые технологии

4.2.1 Общая схема и вводные пояснения

Реализованная платформа является имплементацией идеи, предложенной в [1], с некоторыми изменениями. За основу проекта был взят фреймворк Spring 5.3 (и Spring Boot 2.7), так как он позволяет наиболее удобно использовать различные инструменты совместно и в целом упрощает разработку. Сознательно были выбраны не самые последние версии фреймворка, так как в них отказались от поддержки Java 8, а большинство

инструментов из инфраструктуры больших данных наиболее стабильно работают именно на этой версии.

В текущей реализации платформы был сделан упор на создание «скелета» сложной многокомпонентной системы, задействующей в своей работе различные технологии работы с большими данными, которая может потенциально расширяться и дорабатываться.

Так как система запускалась на локально работающей инфраструктуре, стоит сделать несколько пояснений о том, как она была развёрнута. Проект имеет следующую структуру (здесь также можно видеть конкретные использованные версии):

```
.
├── hadoop
│   └── hadoop-3.3.6
├── hive
│   ├── apache-hive-3.1.3-bin
│   └── db-derby-10.14.1.0-bin
├── kafka
│   └── kafka_2.13-3.6.0
├── MonitorDataProcessingApp
│   ├── mdp-api
│   ├── mdp-app
│   ├── mdp-impl
│   └── mdp-web
├── NilmAnalyticsMlApp
└── spark
    └── spark-3.5.0-bin-hadoop3
```

MonitorDataProcessingApp – директория с основной частью проекта. Директории, соответствующие названиям технологий, использовались для установки необходимого программного обеспечения и его конфигурации. Необходимые файлы конфигураций выложены в [17]. Также стоит упомянуть, что для корректной работы всей инфраструктуры было необходимо установить некоторые переменные окружения. Это было сделано при помощи системных файлов /etc/environment и /etc/profile.d/update_path.sh.

Архитектура цифровой платформы, изображённая на рисунках 23 и 24 демонстрирует основные элементы системы, потоки перемещения данных, а также использованные технологии. Схема разбита на две части для улучшения читаемости, полная версия представлена в приложении А. Песочным цветом отмечены элементы, выходящие за рамки настоящей работы и требующие отдельного глубокого рассмотрения.

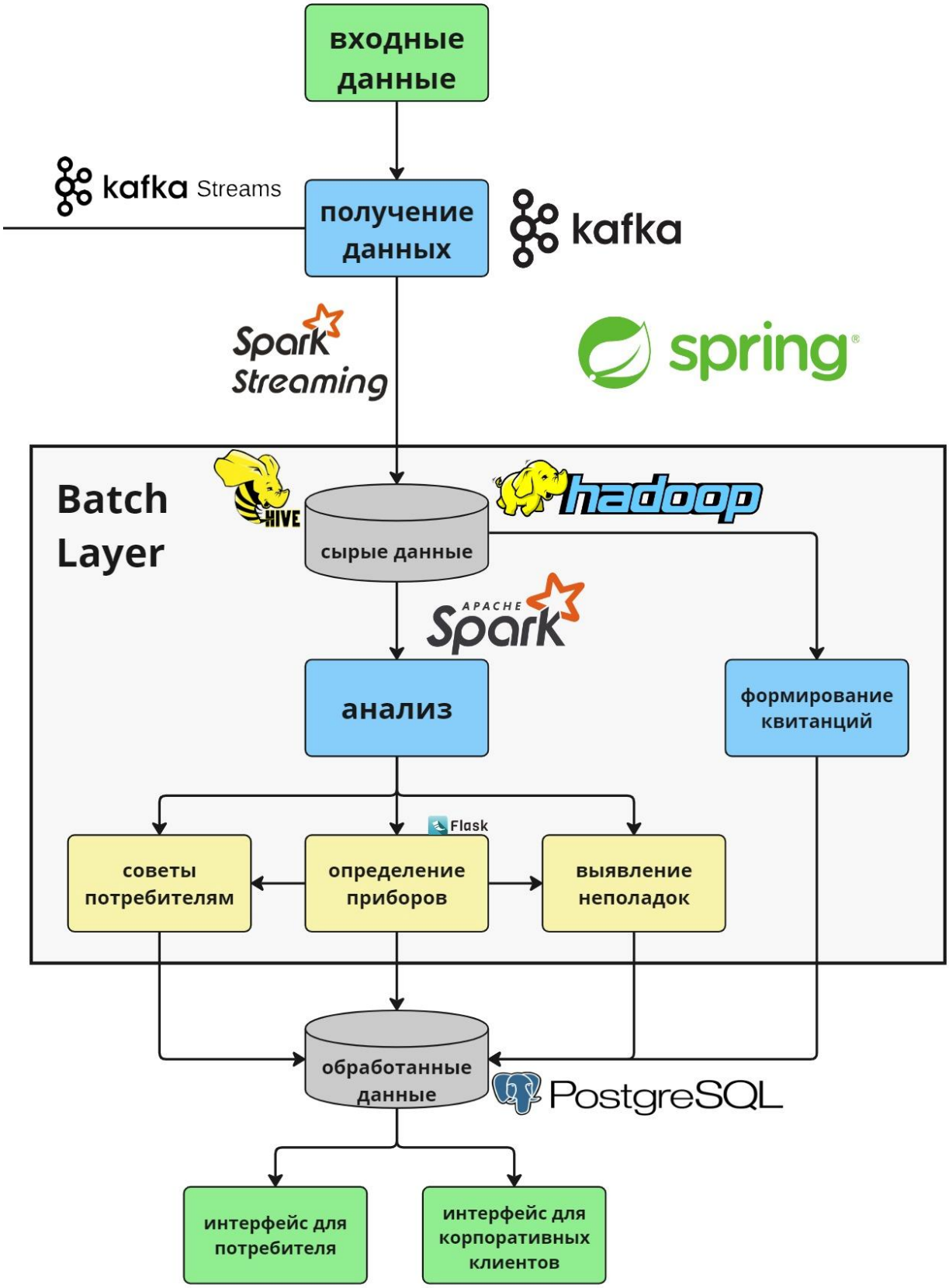

Рисунок 23 – Архитектура цифровой платформы

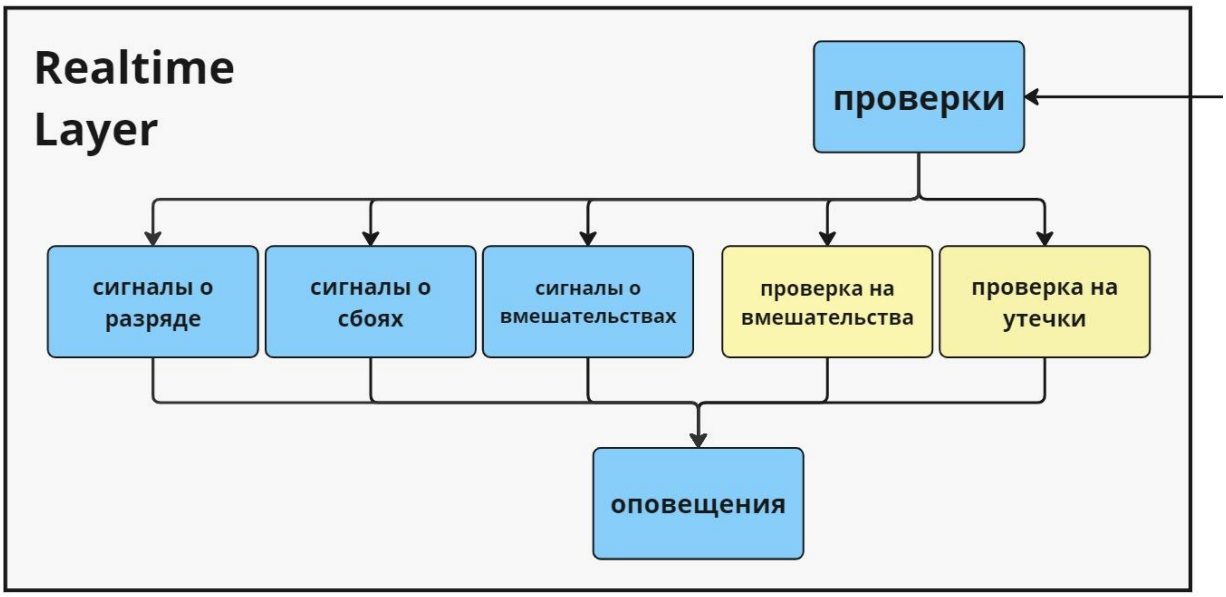

Рисунок 24 – Архитектура цифровой платформы (продолжение)

При попадании в систему данные оказываются в брокере сообщений – масштабируемом хранилище с высокой отказоустойчивостью. Был использован брокер Kafka 3.6. Во входной топик «monitor-data-input» в JSON формате принимаются полученные со счётчиков данные, имеющие следующий вид: {"monitorId": "345", "timestamp": "2023-10-09 10:00:02.002", "amount": 11.2, "flags": "101"}. В проекте для отправки данных в топик и инициализации процесса обработки имплементирован класс CsvTopicFiller, который позволяет отправить данные из файла в брокер сообщений.

Далее, в соответствии со схемой работы лямбда-архитектуры, данные в двух потоках данных направляются из брокера в пакетный слой и в слой реального времени, где они обрабатываются и анализируются независимо. Разделение платформы на слои также отражено рисунках 23 и 24. Их подробное рассмотрение – в последующих параграфах.

4.2.2 Realtime layer

Для оперативного анализа поступающих данных использовался инструмент Kafka Streams, позволяющий в потоковом режиме обрабатывать информацию из брокера. Здесь анализируются сигналы, полученные от счётчиков. Под сигналами тут имеются в виду значения бинарных флагов из

поля «flags». Можно реагировать на них сразу же или использовать более глубокую аналитику, например, проверять пришёл ли такой сигнал единоразово или это случается периодически.

Кроме получения явных сигналов, также возможно проверка на основе анализа получаемых данных. Последние полученные данные для каждого счётчика некоторое время хранятся в realtime-слое для осуществления такого анализа. Таким образом можно автоматически детектировать нештатную ситуацию или попытку вмешательства в работу устройства, выявив нестандартное поведение показаний.

Результаты всех проверок далее поступают в обработчик – в проекте это класс HotDataAnalyticsHandler. Тут происходит реагирование – это может быть вызов внешней интеграции с другой системой, например, уведомление в экстренную службу, или простая отправка пользователю уведомления на электронную почту о необходимости смены батареи в скором времени.

4.2.3 Batch layer

Для хранилища сырых данных был использован Hadoop 3.3 в связке с Hive 3.1. Это довольно стандартное решение для хранения больших данных, Hadoop проверенный временем низкоуровневый инструмент, а Hive позволяет упростить работу с данными, посредством оперирования с ними через SQL-подобный язык запросов. Для транслирования данных из Kafka в Hive был использован Spark 3.5 и его движок для обработки потоков данных Spark Structured Streaming.

Данные о потреблении ресурсов хранятся в виде Hive-таблицы MonitoringData, создаваемой скриптом hive_init.sql. Данная таблица содержит те же поля, что и JSON-объекты, поступающие в Kafka, а именно monitorId, timestamp, amount, flags. Таблица разбита на партиции по monitorId для оптимизации её работы.

В этом слое происходит основной анализ накопленных данных. В первую очередь речь, конечно, об определении подключенных в сеть приборов и их параметров. Как уже упоминалось, это NILM-задача. А на основе полученной информации можно предлагать потребителям основанные на их потребностях товары коммерческих партнёров. В то же время можно более точно искать неполадки в работе сети домохозяйства, чем в realtime-слое, или фиксировать подозрительную активность.

Здесь возможна реализация нескольких подходов взаимодействия с данными. Основная сложность в том, что модели машинного обучения практически всегда написаны на языке Python и просто вызвать их Java-приложения не выйдет. Возможен вынос ответственности за своевременную обработку данных в отдельное Python-приложение, которое будет использовать PySpark для самостоятельного извлечения данных из хранилища для последующего анализа. Другой вариант – сделать Python-сервер, который будет принимать данные для обработки в приходящих к нему запросах; временные ряды относительно легковесная сущность и такой подход может иметь место, хотя может вызвать трудности в будущем при росте количества данных, которые необходимо обрабатывать. В проекте на данный момент был реализован второй вариант с использованием фреймворка Flask (в директории NilmAnalyticsMlApp), так как этого достаточно для демонстрации схемы функционирования системы.

Также к этому слою можно отнести формирование обычных ежемесячных квитанций. Оно контролируется системой – каждый месяц автоматически запускается процесс оформления для каждого существующего счётчика.

Здесь же хочется отметить, что разработанная архитектура цифровой платформы предполагает применение множества разнородных технологий, каждая из которых требует особого порядка инициализации, запуска и выключения. Поэтому для упрощения работы были написаны скрипты

clean_all.sh и start_all.sh. Первый выключает все активные инструменты и очищает созданные ими файлы, хранящие состояние. Второй – запускает всю необходимую инфраструктуру: Kafka, Hadoop, Hive (в том числе, запускает hive_init.sql), PostgreSQL.

4.2.4 Service layer

Обозначение этого слоя опущено на схеме архитектуры системы во избежание излишнего загромождения. К нему относятся «обработанные данные», в которых хранится информация об определённых приборах пользователя, выставленные счета, а также другая информация, относящаяся к пользователям. К этим данным могут получать доступ «внешние» приложения, предназначенные для обслуживания пользователей и коммерческих партнёров. Потенциально это могут быть любые стандартные способы взаимодействия – через веб-сайт, мобильное приложение, чат-бота или с помощью уведомлений по электронной почте.

Целесообразно на текущем этапе использовать обычную реляционную базу данных, так как объём информации после обработки будет существенно меньше исходного и не будет требовать распределённого хранения. В проекте использована PostgreSQL. В перспективе, при большом количестве клиентов, конечно, может потребоваться переход на более продвинутый инструмент.

4.3 Возможности расширения и предоставленные интерфейсы

Построенная платформа на данный момент представляет собой прототип для создания реально применимой системы в перспективе. Для реализации полноценного продукта необходимо проведение существенного расширения всего функционала, однако основы для этого расширения уже заложены созданными и предполагаемыми интерфейсами. Также важной будет настройка и организация кластерной инфраструктуры, соразмерной количеству обрабатываемых данных.

Возможно расширение realtime-слоя. Во-первых, необходима реализация описанных ранее интеграций с внешними системами для

реагирования на те или иные исходы обработки. Во-вторых, разработка нейронных сетей и метода работы с ними для анализа получаемых данных. Для этих целей в проекте созданы интерфейсы DataAnalyser и AnalyticsHandler. Также возможно увеличение числа используемых флагов для индикации новых типов событий или состояний, в проекте для этого достаточно добавить новые записи в MonitorData.Flag и можно будет сразу же использовать новые бинарные флаги в отправляемых данных.

В batch-слое так же актуальны вопросы организации взаимодействия с нейронными сетями, подробнее рассмотренной в 4.2.3, и разработки самих моделей этих сетей. Необходима проработка порядка расчёта и формата счетов на оплату услуг, их донесения до пользователей и обработка платежей. Для этого могут быть расширены BillingEndPoint и BillingService. Помимо этого, необходимо создание политики автоматической обработки сырых данных – по прошествии определённого времени, при накоплении заданного количества данных или при выполнении какого-то иного условия. Также, могут быть задействованы многочисленные возможности оптимизации, предоставляемые используемыми инструментами. Например, можно настроить Hive-таблицу MonitoringData для хранения данных в сжатом виде, а также может быть добавлено новое поле для партицирования month, хранящее год и месяц, это может дополнительно оптимизировать работу с данными таблицы в долгосрочной перспективе.

Перечисленные выше вопросы находятся за пределами поставленной в данной работе задачи. Также, нужно иметь в виду, что для запуска реального продукта необходимым является создание средств взаимодействия с клиентами.

ЗАКЛЮЧЕНИЕ

В настоящей работе представлены результаты разработки прототипа цифровой платформы для мониторинга ресурсопотребления домохозяйств в части ее инфраструктурной основы, обеспечивающей получение, первичную обработку и хранение данных, поступающих от сенсоров.

Для обеспечения эффективности этой платформы было проведено исследование с целью выявления оптимального подхода к формированию входных данных. При сравнении синхронного и асинхронного подходов сбора данных о ресурсопотреблении были сформированы и проведены численные эксперименты, выполнена обработка и анализ полученных результатов.

На основе результатов анализа был сделан вывод о том, что ориентация при разработке платформы на структуру входного потока данных, формируемого при асинхронном подходе, позволит строить эффективные аналитические модели потребления ресурсов, благодаря более высокой точности представления реального потребления теми наборами данных, которые при такой структуре входного потока будут сохраняться в системе.

Вторым положительным эффектом ориентации на указанную структуру входного потока является сокращение объема необходимой для хранения данных памяти, совместно с уменьшением нагрузки на систему передачи данных.

Также была разработана архитектура цифровой платформы, реализующей обработку асинхронных данных, поступающих со смарт-счётчиков. В рамках данной работы был реализован прототип этой платформы с использованием программной инфраструктуры для обработки больших данных на основе лямбда-архитектуры.

Кроме того, были рассмотрены направления дальнейших исследований, по результатам которых может быть доработан и расширен прототип цифровой платформы.

На основе разработанного программного кода, использованного при выполнении экспериментов, могут быть продолжены исследования других возможных форматов входных данных.

На защиту выносятся следующие результаты:

1. Архитектура цифровой платформы асинхронной сенсорной системы дистанционного мониторинга ресурсопотребления домохозяйств. Подготовлена соответствующая статья и представлен доклад на 65-й Всероссийской научной конференции МФТИ в честь 115-летия Л.Д. [1].

2. Разработанный прототип цифровой платформы, реализующей обработку асинхронных данных, поступающих со смарт-счётчиков с использованием программной инфраструктуры для обработки больших данных на основе лямбда-архитектуры.

3. Разработанный способ проведения численных экспериментов с целью выявления предпочтительного формата входных данных платформы.

4. Полученные результаты проведённых экспериментов на реальных данных об энергопотреблении домохозяйств.

5. Результаты выполненного анализа данных, полученных в ходе численных экспериментов.

6. Обоснование выбора формата входных данных цифровой платформы на основе результатов проведённого анализа итогов экспериментов.

К возможным направлениям дальнейшего развития работы следует отнести:

1. Дальнейший анализ возможных форматов входных данных цифровой платформы.

2. Расширение разработанного прототипа цифровой платформы с целью создания коммерческого продукта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] И. М. Филиппов, В. Н. Логинов и С. И. Сурнов, «Об архитектуре цифровой платформы асинхронной сенсорной системы дистанционного», в *ТРУДЫ 65-й Всероссийской научной конференции МФТИ в честь 115-летия Л. Д. Ландау*, Москва, 2023.

[2] И. А. Бычковский, Р. Э. Мукумов, Г. С. Сурнов и С. И. Сурнов, «SMART MONITORING: больше, чем «умный учет» в ЖКХ», *Энергосбережение,* № 6, сс. 38-41, 2017.

[3] G. Hart, "Nonintrusive appliance load monitoring", *Proceedings of the IEEE,* vol. 80, no. 12, pp. 1870-1891, 1992.

[4] Z. Wang and G. Zheng, "Residential Appliances Identification and Monitoring by a Nonintrusive Method", *IEEE Transactions on Smart Grid,* vol. 3, no. 1, pp. 80-92, 2012.

[5] M. Aiad and P. H. Lee, 2017 IEEE Power and Energy Conference at Illinois (PECI), 2017.

[6] J. Kelly and W. Knottenbelt, "Neural NILM: Deep Neural Networks Applied to Energy Disaggregation", 2015.

[7] J. Z. Kolter and M. J. Johnson, "REDD: A Public Data Set for Energy Disaggregation Research", 2011.

[8] В. Н. Логинов, И. А. Бычковский, Г. С. Сурнов и С. И. Сурнов, «Smart Monitoring – технология дистанционного мониторинга потребления электроэнергии, воды, тепловой энергии и газа в Smart City», *Труды МФТИ,* т. 12, № 1, сс. 90-99, 2020.

[9] И. Филиппов, «GitHub экспериментальной части» URL: https://github.com/FilippovIvan19/master_paper_statistical_experiments.

[10] T. A. S. Foundation, "Apache License Version 2.0", 2004 URL: https://www.apache.org/licenses/LICENSE-2.0. [Accessed 2024 April 30].

[11] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh and M. Srivastava, "NILMTK: an open source toolkit for non-intrusive load monitoring", in *Proceedings of the 5th international conference on Future energy systems*, Cambridge, UK, 2014.

[12] "GitHub NILMTK: Non-Intrusive Load Monitoring Toolkit" URL: https://github.com/nilmtk/nilmtk. [Accessed 30 April 2024].

[13] "REDD data" URL: http://redd.csail.mit.edu. [Accessed 29 April 2024].

[14] "nilmtk issue. REDD dataset link not working" URL: https://github.com/nilmtk/nilmtk/issues/962#issuecomment-1464555930. [Accessed 29 April 2024].

[15] M. Pullinger, J. Kilgour, N. Goddard, N. Berliner, L. Webb, M. Dzikovska, H. Lovell, J. Mann, C. Sutton, J. Webb and M. Zhong, "The IDEAL household energy dataset, electricity, gas, contextual sensor data and survey data for 255 UK homes", *Scientific Data*, vol. 8, no. 1, 2021.

[16] "IDEAL Household Energy Dataset" URL: https://datashare.ed.ac.uk/handle/10283/3647. [Accessed 29 April 2024].

[17] И. Филиппов, «GitHub практической части» URL: https://github.com/FilippovIvan19/master_paper_digital_platform.

[18] «Архитектурный паттерн для обработки больших данных: Lambda», 16 Октября 2023 URL: https://habr.com/ru/companies/otus/articles/766672/. [Дата обращения: 21 Апреля 2024].

[19] P. K. Gudipati, "Implementing a Lambda Architecture to perform real-time updates", Kansas State University, Manhattan, Kansas, 2016.

[20] M. Feick, N. Kleer and M. Kohn, "Fundamentals of Real-Time Data Processing Architectures Lambda and Kappa", in *SKILL 2018 - Studierendenkonferenz Informatik*, Bonn, 2018.

ПРИЛОЖЕНИЕ А

Полная архитектура цифровой платформы