Федеральное государственное бюджетное учреждение науки
Институт динамики систем и теории управления имени В.М. Матросова
Сибирского отделения Российской академии наук

На правах рукописи

Воскобойников Михаил Леонтьевич

Технология разработки и применения сервис-ориентированных приложений в контейнеризированной вычислительной среде

Специальность 2.3.5. – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук, доцент
А.Г. Феоктистов

Иркутск – 2025

Оглавление

Введение

Актуальность темы исследования. В настоящее время решение фундаментальных и прикладных задач зачастую осуществляется с помощью распределенных научных приложений, характеризующихся модульной структурой их прикладного программного обеспечения (ПО), развитым системным ПО (совокупностью программ, обеспечивающих создание приложения, организацию вычислений, обработку данных и взаимодействие различных категорий пользователей приложения) и ориентацией на решение определенного класса задач в гетерогенной распределенной вычислительной среде (ГРВС). В современных приложениях схема решения задачи реализуется научным рабочим процессом (НРП, англ., Scientific Workflow) – информационно-вычислительной структурой, отражающей бизнес-логику предметной области в интеграции предметных данных, ПО и вычислительных ресурсов в процессе проведения экспериментов. В процессе применения НРП при решении ресурсоемких задач, например, для исследования энергетических комплексов (ЭК), возникает необходимость решения *ряда важных проблем*: согласованного использования разнородных вычислительных ресурсов; учета специфики предметных областей решаемых задач; интеграции, тестирования и контейнеризации прикладного и системного ПО (ПСПО); стандартизации спецификаций объектов предметной области и вычислительной модели приложения, форматов представления и протоколов передачи данных; управления вычислениями в ГРВС; взаимодействия с системами управления прохождением заданий (СУПЗ), а также с метапланировщиками; использования технологий ускорения обработки данных в оперативной памяти (ОП), таких как In-Memory Data Grid (IMDG) [1]; поддержки стандарта Web Processing Service (WPS) [2]. IMDG имеет неоспоримое преимущество в скорости обработки данных по сравнению с традиционными базами данных (БД). Стандарт WPS незаменим при работе с большими массивами пространственно-распределенных данных.

Необходимость учета предметной специфики и обеспечения масштабирования вычислений обуславливает переход от использования ГРВС

общего назначения, которая может включать Grid-системы, облачные платформы, ресурсы центров коллективного пользования и другие вычислительные мощности, к применению предметно-ориентированной вычислительной среды (ПОВС). Такой переход обеспечивает рациональное сочетание возможностей ресурсов среды, потребностей и накладных затрат, обуславливаемых особенностями предметных областей для конкретных классов решаемых задач.

В силу актуальности вышеперечисленных проблем степень разработанности исследований относительно каждой из них достаточна высока. Результаты по концептуальному и сборочному программированию, а также построению и применению пакетов прикладных программ отражены в работах И.О. Бабаева, М.М. Горбунова-Посадова, А.Ю. Горнова, А.П. Ершова, Е.А. Жоголева, В.П. Ильина, Д.А. Корягина, Е.М. Лаврищевой, С.С. Лаврова, В.Э. Малышкина, В.М. Матросова, Г.А. Опарина, Э.Х. Тыугу, А.И. Тятюшкина, Г.С. Цейтина, D. Gries, R.T. Hartley, G. Mayers, S.P. Reiss, J.F. Sowa, W.M. Turski и др.

В рамках организации параллельных и распределенных вычислительных систем следует отметить работы А.И. Аветисяна, А.А. Белеванцева, В.Б. Бетелина, Вл.В. Воеводина, В.П. Гергеля, Б.М. Глинского, В.А. Ильина, В.В. Коренькова, В.Д. Корнеева, И.И. Левина, А.И. Легалова, Л.Б. Соколинского, В.В. Топоркова, В.Г. Хорошевского, А.Н. Черных, D. Abramson, H. Casanova, J. Dongarra, S.S. Gill, M. Livny, T. Tannenbaum, A. Zomaya и других российских и зарубежных ученых. Параллельно развиваются системы управления НРП (англ., Workflow Management System – WMS). Существенный вклад в это направление внесли T.L. Casavant, E. Deelman, Y. Gil, G. Juve, S. Pandey, D. Talia, I. Taylor, B. Wassermann и др.

Особое внимание научного сообщества уделяется развитию сервис-ориентированного подхода к организации распределенных вычислений (см., например, работы А.В. Бухановского, И.В. Бычкова, И.А. Каляева, Л.В. Массель, И.А. Пестунова, Г.И. Радченко, Г.М. Ружникова, С.И. Смагина, А.А. Сорокина, О.В. Сухорослова, Р.К. Федорова, А.М. Федотова, А.Е. Хмельнова, Ю.И. Шокина, М.В. Якобовского, О.Э. Якубайлика, R. Buyya, R. Fielding, I. Foster, M.B. Juric, C. Kesselman, S.J. Kim, X. Liu, B. Schuller, M.P. Singh, A. Rajasekar, S. Tuecke,

S. Weerawarana и др.). В настоящее время сервис-ориентированные приложения (СОП) становятся все более востребованными на практике.

Анализ современного состояния в области распределенных вычислений [3-5] показывает, что *известные методы и средства не обеспечивают в полной мере комплексного решения проблемы разработки и применения СОП на основе НРП и дальнейшего эффективного построения и применения ПОВС*. Требуется создание новых технологических решений, а также адаптация или модификация известных разработок с целью улучшения критериев подготовки и проведения экспериментов в ПОВС с поддержкой комплексного выполнения интеграции, тестирования и контейнеризации ПСПО. *Диссертационная работа направлена на решение данной актуальной задачи* [6]*, имеющей важное научно-техническое значение в области распределенных вычислений*. В работе выделяются базовые критерии пользователей (временные затраты на подготовку и проведение эксперимента и точность оценки требуемых ресурсов) и владельцев ресурсов (загрузка процессора, балансировка нагрузки и эффективность использования ресурсов).

Цель работы состоит в разработке моделей, алгоритмов и инструментальных средств создания и применения СОП в ПОВС, обеспечивающих улучшение базовых критериев пользователей и владельцев ресурсов в сравнении с известными разработками за счет использования знаний о специфике решаемых задач, комплексного выполнения интеграции, тестирования и контейнеризации ПСПО, а также применения IMDG.

Основные задачи диссертации, решаемые для достижения поставленной цели применительно к ПОВС:

- анализ современных подходов к организации СОП, известных систем управления НРП, методов и средств виртуализации и контейнеризации;
- разработка вычислительной модели СОП, включающей специализированные знания о сущностях и процессах интеграции, тестирования и контейнеризации ПСПО;
- разработка алгоритмов построения и выполнения НРП;

- разработка методики, алгоритмов и программных средств автоматизации применения технологии IMDG;
- поддержка интеграции, тестирования и контейнеризации ПСПО;
- реализация инструментального комплекса (ИК) разработки СОП для решения ресурсоемких научных и практических задач на основе НРП;
- апробация результатов диссертации на примерах решения ресурсоемких научных и практических задач.

Объектом исследования являются методы и средства разработки и применения распределенных научных приложений в ГРВС.

Предметом исследования выступают модели, алгоритмы и инструментальные средства разработки и применения СОП в ПОВС.

Методы исследования. При решении поставленных задач использовались методы и средства концептуального, модульного и сервис-ориентированного программирования, контейнеризации ПО, организации параллельных и распределенных вычислений.

Научная новизна диссертации заключается в развитии теории и практики распределенных вычислений относительно улучшения базовых критериев пользователей и владельцев ресурсов при создании СОП (поддерживающих стандарт WPS) в ПОВС в сравнении с известными разработками. При этом на защиту выносятся следующие основные положения:

1) предложена вычислительная модель СОП, расширяющая известные модели знаниями о сущностях и процессах интеграции, тестирования и контейнеризации ПСПО и тем самым позволяющая повысить качество управления вычислениями при ее использовании;
2) разработаны алгоритмы построения и выполнения НРП в ПОВС, обеспечивающие в сравнении с алгоритмами управления вычислениями в известных WMS согласование базовых критериев пользователей и владельцев ресурсов за счет использования расширенной модели и результатов тестирования рабочих процессов на испытательных стендах;

3) реализованы методика, алгоритмы и программные средства автоматизации динамического развертывания кластера IMDG, отличающиеся от известных разработок более высокой точностью оценки требуемых ресурсов посредством анализа влияния специфики данных на их размещение в ОП и сокращающие время развертывания кластера относительно ручного режима;
4) создан ИК, интегрирующий перечисленные модель, алгоритмы и программные средства в рамках единой технологии разработки и применения СОП для решения ресурсоемких научных и практических задач на основе НРП в ПОВС и обеспечивающий сокращение времени на подготовку и проведение экспериментов в сравнении с известными WMS.

Практическая значимость. Применение результатов диссертации обеспечивает эффективное управление НРП на основе согласования заданных критериев качества решения задач и предпочтений владельцев ресурсов. Основные результаты диссертации использованы при выполнении государственных заданий и научных исследований ИДСТУ СО РАН, в том числе в рамках следующих научно-технических работ: регионального проекта РФФИ и Правительства Иркутской области № 20-47-380002 р_а «Математическое и информационное моделирование инфраструктурных объектов Байкальской природной территории» (2020-2022 гг.); проекта Министерства науки и высшего образования РФ № FWEW-2021-0005 «Технологии разработки и анализа предметно-ориентированных интеллектуальных систем группового управления в недетерминированных распределенных средах» (2020-2025 гг.); гранта № 075-15-2024-533 Министерства науки и высшего образования РФ на выполнение крупного научного проекта по приоритетным направлениям научно-технологического развития (проект «Фундаментальные исследования Байкальской природной территории на основе системы взаимосвязанных базовых методов, моделей, нейронных сетей и цифровой платформы экологического мониторинга окружающей среды», 2024-2026 гг.).

Получены акты о внедрении результатов диссертации, в частности, ИК Framework for Development and Execution of Scientific WorkFlows (FDE-SWFs), в

Институте систем энергетики им. Л.А. Мелентьева СО РАН и Институте вычислительной математики и математической геофизики СО РАН.

Достоверность и обоснованность полученных в диссертации результатов подтверждается положительными результатами анализа адекватности предложенных модели и алгоритмов, корректным применением классических методов исследования и соответствием результатов экспериментов известным теоретическим оценкам.

Соответствие диссертации паспорту специальности. Тема и основные результаты диссертации соответствуют следующим областям исследований паспорта специальности 2.3.5. – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей:

- п. 1 – модели, методы и алгоритмы проектирования, анализа, трансформации, верификации и тестирования программ и программных систем;
- п. 3 – модели, методы, архитектуры, алгоритмы, языки и программные инструменты организации взаимодействия программ и программных систем;
- п. 8 – модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

Апробация. Основные результаты диссертации представлены на VIII Международной научно-практической конференции «Информационные технологии и высокопроизводительные вычисления» (Хабаровск, 2025 г.), Выездном совещании участников проекта «Фундаментальные исследования Байкальской природной территории на основе системы взаимосвязанных базовых методов, моделей, нейронных сетей и цифровой платформы экологического мониторинга окружающей среды» (г. Белокуриха, 2025 г.), X Международной конференции «Суперкомпьютерные дни в России» (Москва, 2024 г.), VI и VII International Workshops on Information, Computation, and Control Systems for Distributed Environments (Иркутск, 2024 и 2025 гг.), Международной конференции «Ляпуновские чтения» (Иркутск, 2023–2025 гг.), XXVII, XXIX и XXX Байкальской

Всероссийской конференции с международным участием «Информационные и математические технологии в науке и управлении» (Иркутск, 2022, 2024 и 2025 гг.), International Workshop on Critical Infrastructures in the Digital World (Большое Голоустное, 2024 г.), IV Scientific-practical Workshop on Information Technologies: Algorithms, Models, Systems (Иркутск, 2021 г.), а также семинарах ИДСТУ СО РАН.

Публикации. Результаты научных исследований автора отражены в 30 научных работах. Основные публикации представлены в российских журналах [7-13], рекомендованных ВАК для опубликования научных результатов диссертации, а также в работах, проиндексированных в международных базах цитирования Web of Science и Scopus [7, 8, 13-16]. Получено 3 свидетельства о государственной регистрации программ для ЭВМ [37-39].

Личный вклад автора. Все выносимые на защиту научные положения получены соискателем лично. Из совместных работ в диссертацию включены только те результаты, которые принадлежат непосредственно автору. В работах, выполненных в соавторстве, в частности, в основных публикациях [7-11, 14-16], лично соискателем проведен сравнительный анализ систем управления НРП, рассмотрены направления развития сервис-ориентированных технологий, включая стандарты описания рабочих процессов, разработаны компоненты ИК FDE-SWFs для создания и применения СОП, расширена вычислительная модель, разработаны алгоритмы управления вычислениями в ПОВС, созданы испытательные стенды для тестирования ПСПО, проведены вычислительные эксперименты. Методика контейнеризации ПО [13] разработана в неделимом соавторстве с Р.О. Костроминым. Вклад соискателя в разработанное ПО [37, 38] состоит в разработке и реализации архитектуры, а также алгоритмов работы программ.

Структура работы. Диссертация состоит из введения, четырех глав, заключения, библиографии из 174 наименования, списка принятых сокращений и 7 приложений. Общий объем основного текста работы – 143 страница, включая 14 таблиц и 46 рисунков.

Во введении обосновывается актуальность темы и направления исследования диссертации, формулируются ее цель и основные задачи,

подчеркивается научная и практическая значимость работы, приводится структура диссертации.

В первой главе рассматриваются основные понятия и определения, связанные с организацией распределенных вычислений в ПОВС, проводится обзор и выполняется сравнительный анализ известных систем управления НРП, обсуждаются текущее состояние и перспективные направления развития технологий разработки и применения СОП, а также обосновывается разработка нового инструментария для создания и применения СОП и формулируются функциональные и системные требования к его работе.

Вторая глава представляет предложенную вычислительную модель СОП, разработанную схему взаимодействия диспетчера FDE-SWFs с программами (модулями СОП) и программными системами – метапланировщиком и СУПЗ, а также алгоритмы построения и выполнения НРП.

Третья глава посвящена архитектуре, основным возможностям и аспектам реализации ИК FDE-SWFs для разработки и применения СОП. В ней предлагаются сценарии выполнения НРП, рассматриваются возможности ИК по работе с распределенными БД (РБД) в ОП ресурсов ПОВС, приводится методика интеграции и контейнеризации ПСПО, предлагается подход к разработке и применению испытательных стендов для тестирования СОП и их компонентов, а также обсуждаются средства визуализации НРП и расчетных данных.

В четвертой главе приводятся примеры применения результатов диссертации. Рассматриваются приложения, созданные с помощью ИК FDE-SWFs. Демонстрируются преимущества данного комплекса в сравнении с системами подобного назначения, а также сокращение трудозатрат на этапах подготовки и проведения экспериментов.

В заключении подытожены основные результаты диссертации.

Приложения включают копии актов о внедрении результатов диссертации, копии свидетельств о государственной регистрации программ, примеры спецификаций объектов СОП, экспериментальные данные, а также другие дополнительные материалы.

Глава 1. Разработка сервис-ориентированных приложений на основе научных рабочих процессов

В данной главе рассматриваются основные понятия и определения, связанные с организацией распределенных вычислений в ПОВС. Проводится сравнительный анализ известных систем управления НРП. Обсуждается развитие технологий разработки и применения СОП. Рассматриваются существующие стандарты описания сервис-ориентированных НРП. Формулируются функциональные и системные требования к разработке и применению СОП. В конце главы кратко подытоживаются достигнутые в ней научные результаты.

1.1. Основные понятия

Высокопроизводительные вычислительные системы сегодня широко востребованы. Развитие суперкомпьютерных технологий играет определяющую роль в решении больших научных и прикладных задач [40], в том числе задач математического моделирования, искусственного интеллекта, экологического мониторинга и социально-экономического развития природно-технических систем, планирования производства и др. В то же время активно продвигается направление по созданию и применению вычислительных кластеров благодаря их высокому уровню готовности при относительно низких затратах. Кластеры обеспечивают более гибкие и экономичные способы организации потенциально больших вычислительных мощностей, получения доступа к ним и их эффективного использования для решения определенных классов задач в сфере науки, техники и бизнеса, а также увеличения этих мощностей при необходимости.

Вычислительный кластер представляет собой совокупность параллельных или распределенных вычислительных узлов, соединенных между собой с помощью высокоскоростных сетей, таких как Gigabit Ethernet, SCI, Myrinet и Infiniband. Кластер обеспечивает совместную работу своих узлов при решении задач, требующих большого объема вычислений и данных, которые невозможно выполнить на одном компьютере пользователя. Кластеры используются в

основном для обеспечения высокой доступности, балансировки нагрузки и надежного решения вычислительных задач. Зачастую эти проблемы решаются путем резервирования узлов кластера, используемых для предоставления вычислительных услуг в процессе реконфигурирования вычислительной среды в случае сбоя ее компонентов. Производительность системы при этом повышается, поскольку даже в случае выхода из строя одного узла существует резервный узел, который выполнит задачу.

На кластере устанавливается СУПЗ, которая играет роль посредника между пользователями и узлами кластера. СУПЗ выполняет диспетчеризацию очередей заданий пользователей в соответствии с существующими административными политиками и квотами на ресурсы, установленными владельцами кластера, и распределяет вычислительную нагрузку между узлами кластера. Важными вопросами управления заданиями являются упорядочение поступающих заданий, назначение ресурсов для их выполнения, мониторинг очередей, ресурсов и вычислительных процессов, передача исходных данных и результатов расчетов, а также выполнение ряда вспомогательных операций.

Суперкомпьютеры и кластеры составляют основу центров коллективного пользования (ЦКП). Процесс решения задач в рамках ЦКП строго регламентирован и ограничен квотами на использование ресурсов.

Вычислительная среда типа Grid, как правило, на какой-то период времени объединяет вычислительные ресурсы из нескольких административных доменов для достижения общей цели в решения большой задачи. Эти ресурсы могут быть расформированы после завершения решения задачи. Одной из основных стратегий распределенных вычислений является использование связующего программного обеспечения (англ., Middleware) для разделения и распределения общей задачи на подзадачи, решаемые на разных ресурсах Grid. В рамках связующего программного обеспечения важную роль играют метапланировщики, выполняющие диспетчеризацию потоков заданий на уровне Grid.

Ресурсы Grid могут включать отдельные ПК и их совокупности, высокопроизводительные сервера, вычислительные кластеры. Размер Grid может

варьироваться от небольшой сети ПК внутри корпорации до крупных коллабораций многих компаний и сетей. Grid, состоящая из географически распределенных кластеров под разным административным управлением, представляет собой кластерную Grid.

В целом I. Foster определяет Grid как систему, которая координирует ресурсы, не подлежащие централизованному управлению, используя стандартные открытые протоколы и интерфейсы общего назначения для предоставления нетривиального качества обслуживания [41]. Дополняющее определение Grid дано в [42]. Grid определяется как тип параллельной и распределенной системы, которая позволяет динамически распределять, выбирать и агрегировать географически распределенные автономные ресурсы во время выполнения в зависимости от их доступности, возможностей, производительности, стоимости и требований пользователей к качеству обслуживания.

Облачные вычисления относятся как к приложениям, предоставляемым в виде услуг через Интернет, так и к аппаратному и системному программному обеспечению в центрах обработки данных, которые предоставляют эти услуги. Согласно [43] облачная платформа – это тип параллельной и распределенной системы, состоящей из набора взаимосвязанных и виртуализированных компьютеров, которые динамически выделяются и представляются как один или несколько унифицированных вычислительных ресурсов на основе соглашения об уровне обслуживания. В общем случае модель облачных вычислений предназначена для обеспечения сетевого доступа по требованию к вычислительным ресурсам – сетям, серверам, хранилищам, приложениям, услугам и т. п., которые предоставляются, настраиваются и освобождаются с минимальными накладными расходами на управление ими и взаимодействие с их провайдерами [44].

В процессе развития облачных вычислений был разработан широкий спектр моделей представления компонентов вычислительной среды в виде сервисов. В их числе Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), Workflow-as-a-Service (WaaS) и другие [45-47]. В рамках

перечисленных моделей основное внимание акцентируется на реализации сервисами удаленного программного доступа к соответствующему функциональному компоненту вычислительной среды.

В рамках диссертации рассматривается ГРВС, которая может интегрировать собственные кластерные ресурсы пользователей, а также кластерные ресурсы ЦКП, Grid и облачных платформ. В рамках ГРВС разрабатываются и применяются распределенные пакеты прикладных программ (РППП). Кластерные ресурсы для выполнения РППП могут объединяться и управляться на основе виртуализации или контейнеризации.

РППП [48] базируется на традиционном понятии пакета прикладных программ [49] и представляет собой комплекс взаимосвязанных прикладных программ и средств системного обеспечения (программных и языковых), предназначенный для автоматизации решения определенного класса задач в конкретной предметной области в ГРВС. В то же время РППП характеризуется следующими особенностями [11, 12]:

- ориентация на решение класса задач, требующих проведения расчетов с использованием больших объемов вычислительных ресурсов (процессорного времени, ОП, дискового пространства и других характеристик);
- допускается декомпозиция решаемых задач на более простые взаимосвязанные подзадачи;
- разработка прикладного ПО приложения производится на основе модульного подхода к созданию больших программных комплексов;
- предполагается разбиение исходных данных на блоки и независимая параллельная обработка этих блоков экземплярами модулей;
- не предполагается интенсивное взаимодействие между параллельными вычислительными процессами, а также обработка непрерывного потока больших данных терабайтного или петабайтного размера;
- расчеты осуществляются на основе выполнения НРП, представляющего собой схему решения задачи;

- вычисления выполняются, как правило, по одной слабо меняющейся схеме, требующей динамического управления процессом вычислений;
- управление вычислениями осуществляет диспетчер НРП;
- необходимо использование связующего ПО.

Архитектура РППП представлена на рисунке 1.1. В разработке, применении и поддержке функционирования РППП участвуют следующие категории специалистов:

- прикладной программист, строящий вычислительную модель предметной области, в частности, разрабатывающий алгоритмические знания в виде набора программных модулей (прикладного ПО) пакета, формирующий знания по их применению в процессе решения задач на основе понятий (концептов) предметной области (абстрактных параметров, выполняемых над ними операций, других объектов предметной области) и определяющий необходимые форматы и структуры расчетных данных;
- прикладной программист, реализующий дополнительное системное ПО для взаимодействия компонентов пакета со специализированным ПО, под управлением которого функционируют используемые вычислительные ресурсы;
- администратор пакета, выполняющий настройку и конфигурирование РППП, а также доставку, размещение и тестирование его компонентов на ресурсах ГРВС;
- администратор ГРВС, осуществляющий конфигурирование среды и ее ресурсов, производящий настройку взаимодействия компонентов пакета с СУПЗ, которые установлены на ресурсах среды, а также с ее метапланировщиками (связующим ПО), выполняющий виртуализацию и контейнеризацию ПСПО;
- конечные пользователи пакета, решающие свои задачи с его помощью.

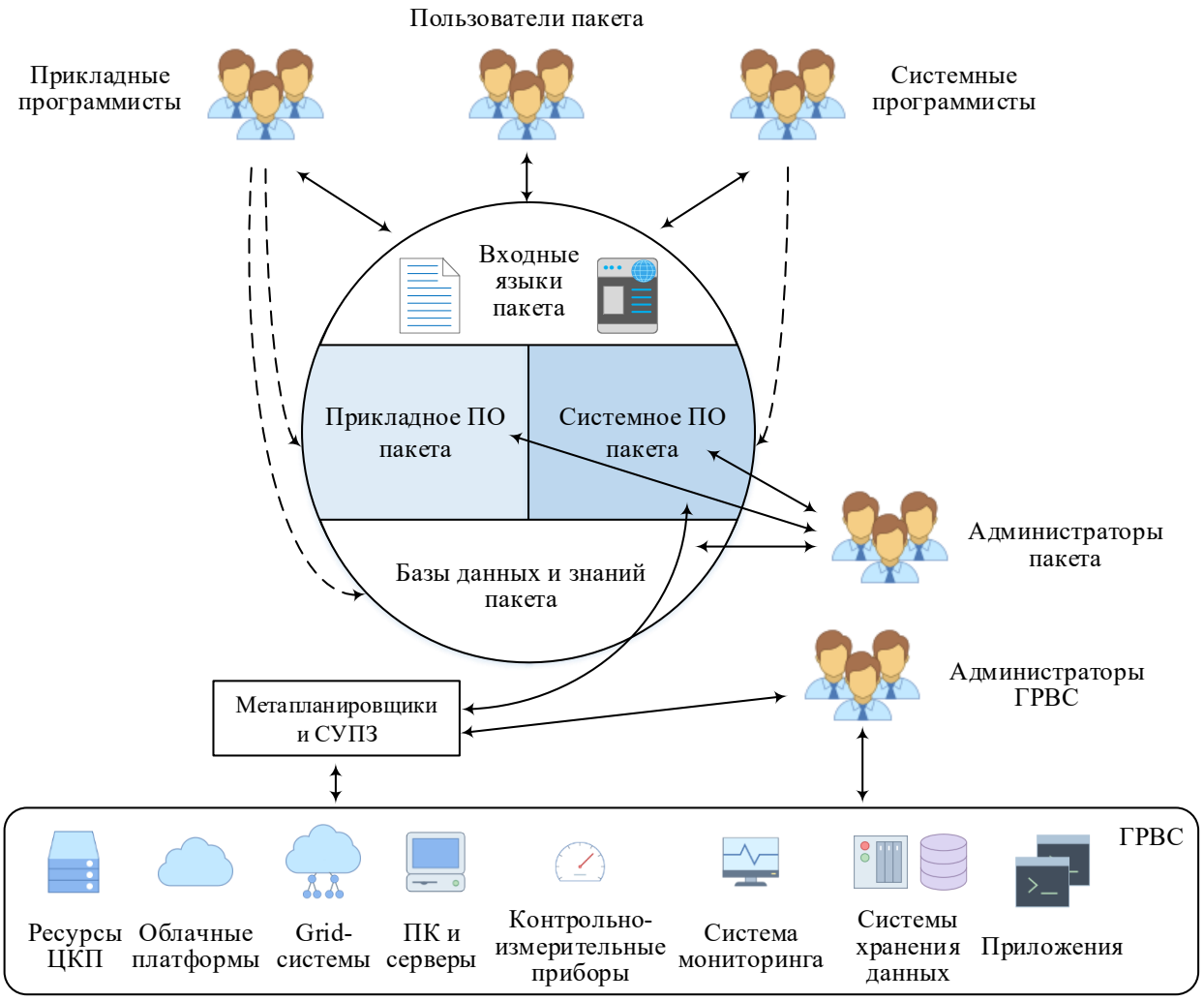


Рисунок 1.1 – Архитектура РППП

Взаимодействие перечисленных категорий специалистов с компонентами пакета осуществляется с помощью набора входных языков. Применяются входные языки двух типов: текстовые языки представления структурированных данных и языки для ввода информации с помощью веб-форм с последующим автоматическим выводом полученных спецификаций на соответствующие текстовые языки. Вычислительная модель предметной области сохраняется в базе знаний (БЗ) пакета. Для работы с исходными данными и результатами вычислений создаются расчетные базы данных.

Связующее ПО в контексте распределенных приложений представляет собой ПО, дополняющее возможности ОС с целью обеспечения обмена данными между различными системами, которые функционируют в рамках ГРВС, а также управления этими системами. Оно определяется как ПО, которое обеспечивает

связь между отдельными программными приложениями [50]. Связующее ПО поддерживает и упрощает функционирование сложных распределенных приложений. Оно предполагает использование веб-серверов, серверов приложений, средств обмена сообщениями и других инструментов, обеспечивающих разработку, развертывание и выполнение распределенных приложений. Системы разработки и применения научных приложений на основе рабочих процессов относятся к классу систем управления НРП (англ., Workflow Management Systems – WMSs). WMS предоставляет программную инфраструктуру для настройки, выполнения и мониторинга определенной последовательности задач, организованных в виде рабочего процесса со сложными зависимостями между решаемыми задачами [51].

РППП, прикладное ПО которого представлено в виде сервисов, представляет собой СОП. Общая схема вычислительной среды для выполнения СОП приведена на рисунке 1.2. Учет особенностей предметной области при разработке СОП позволяет перейти от ГРВС к ПОВС.

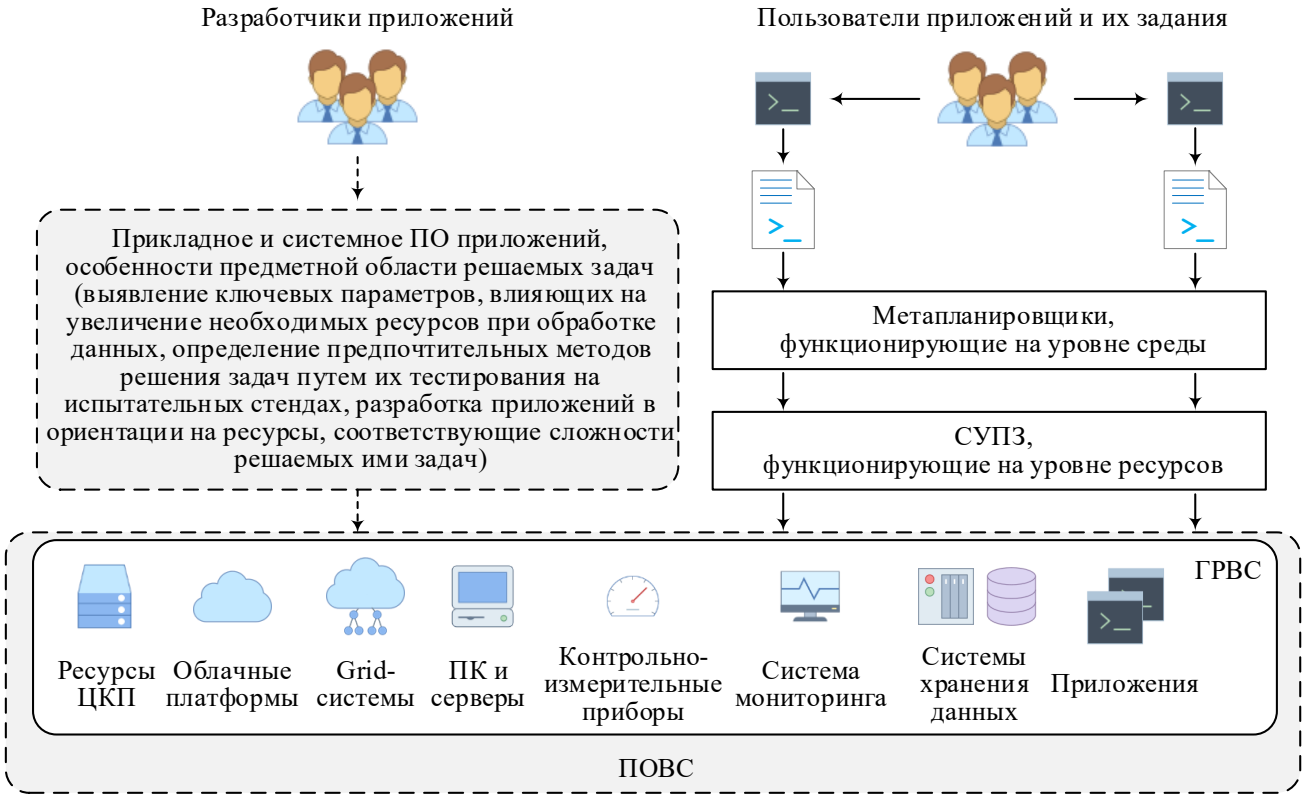


Рисунок 1.2 – Общая схема вычислительной среды

Пользователи приложений формируют задания в виде текстовых спецификаций вычислительных процессов, определяемых НРП. Такие спецификации включают сведения о требуемых ресурсах, исполняемом прикладном ПО, входных и выходных данных, а также иную необходимую информацию. Например, дополнительная информация может определять критерии качества решения своей задачи: время, стоимость, надежность, безопасность и другие характеристики.

Основными компонентами ПОВС являются вычислительные кластеры, кластеры виртуализированных и контейнеризированных ресурсов, которые используются в рамках таких вычислительных инфраструктур, как ЦКП, кластерные Grid и облачные платформы. ПОВС может интегрировать компоненты перечисленных выше инфраструктур.

В общем случае управление вычислениями в ПОВС включает в себя следующие уровни [11]:

- уровень приложений, где выполняется планирование вычислений, построение НРП и выбор вычислительных ресурсов для его выполнения;
- уровень среды, на котором потоки заданий попадают в очередь метапланировщиков ПОВС, поддерживающих взаимодействие с выбранными ресурсами, и затем, в соответствии с дисциплинами диспетчеризации, им назначаются конкретные ресурсы;
- уровень ресурсов, где задания распределяются СУПЗ между вычислительными узлами ресурсов с целью их дальнейшего выполнения на выделенных узлах.

1.2. Системы управления научными рабочими процессами

Проведен сравнительный анализ WMS [11, 30]. НРП представляется специализированной формой графа, которая описывает вычислительные процессы, их зависимости в контексте сбора, обработки и анализа данных. Он представляет собой бизнес-логику предметной области в применении предметно-ориентированных данных и ПО (набора прикладных модулей) для решения задач в

этой предметной области.

Различают два типа НРП: абстрактный и конкретный. Абстрактный НРП описывается в виде абстрактной схемы решения задачи без обращения к конкретным вычислительным ресурсам. Это позволяет определять НРП без детальной конкретизации его реализации на низком программно-аппаратном уровне. Операции абстрактного НРП могут реализовываться различным прикладным ПО и отображаться на любые подходящие вычислительные ресурсы с помощью механизмов планирования вычислений и управления ресурсами. В случае конкретного НРП его операции связаны с предопределенным ПО, выполняемом на заданных ресурсах.

Построение НРП осуществляется по процедурной или непроцедурной постановке задачи. В первом случае задается последовательность операций НРП и затем осуществляется информационное планирование его входных и выходных параметров.

Во втором случае задача формулируется следующим образом: «вычислить значения целевых (выходных) параметров по заданным значениям исходных (входных) параметров». Далее на вычислительной модели (описании предметной области – ее параметров, вычислительных операций и операций обработки данных, а также отношений между параметрами и операциями) осуществляется автоматическое планирование последовательности операций НРП, необходимых для решения поставленной задачи. Как правило, планирование вычислений осуществляется одним из системных компонентов WMS. Непроцедурная постановка задачи позволяет разработчикам приложения и его пользователям работать на более высоком уровне абстракции. Она скрывает детали реализации и позволяет сосредоточиться на описании задачи и отношений между объектами.

WMS – это программное средство, частично автоматизирующее определение и выполнение НРП, а также управление им в соответствии с его информационно-вычислительной структурой. Для задания НРП часто используется ориентированный ациклический граф (DAG). В общем случае вершины и ребра DAG соответственно представляют прикладные программные модули и потоки

данных между ними.

Следует отметить следующие WMS, которые хорошо известны, поддерживаются, развиваются и широко используются на практике: Uniform Interface to Computing Resources (UNICORE) [52], Directed Acyclic Graph Manager (DAGMan) [53], Pegasus [54], HyperFlow [55], Workflow-as-a-Service Cloud Platform (WaaS CP) [56], Galaxy [57], Apache Airflow (AA) [58] и Orlando Tools (OT) [59].

UNICORE, DAGMan, Pegasus, Apache Airflow и OT занимают лидирующие позиции в области традиционного управления НРП. Они основаны на модульном подходе к созданию и использованию таких процессов. В рамках НРП общая задача разбивается на набор задач, обычно представленных в виде DAG.

Комплексная поддержка веб-сервисов является актуальным направлением современных WMS. Использование веб-сервисов значительно расширяет вычислительные возможности приложений, основанных на НРП. WMS с поддержкой веб-сервисов позволяют гибко и динамично интегрировать различные НРП, созданные разными разработчиками, за счет управления данными и вычислениями при выполнении этих процессов. В этом направлении успешно развиваются HyperFlow, WaaS CP и Galaxy.

Для научных приложений, ориентированных на решение различных классов задач в области мониторинга окружающей среды, особое значение имеет интеграция с геоинформационными системами посредством специализированных сервисов веб-обработки (WPS). Примерами проектов, направленных на создание и использование таких приложений, являются GeoJModelBuilder (GJMB) [60] и Business Process Execution Language (BPEL) Designer Project (DP) [61]. GJMB и BPEL DP относятся к классу WMS. GJMB – это фреймворк для управления и координации геопространственных датчиков, данных и прикладного ПО в среде, основанной на НРП. BPEL DP реализует интеграцию веб-сервисов и WPS с помощью рабочих процессов на основе BPEL.

Ключевые возможности WMS, которые особенно важны с точки зрения исследования в рамках диссертации, приведены в таблице 1.1.

Таблица 1.1 – Возможности WMS для поддержки НРП

WMS	Язык описания НРП	Поддержка					
		Ветвления / циклы	Системные операции	Тип НРП	Постановка задачи	Веб-сервисы	WPS-сервисы
UNICORE	XML-like	+ / +	+	Конкретный	Процедурная	+	-
DAGMan	Script-like	- / -	+	Абстрактный	Процедурная	-	-
Pegasus	XML-like	- / -	+	Абстрактный	Процедурная	-	-
AA	Python	- / -	-	Конкретный	Процедурная	-	-
HyperFlow	JavaScript, JSON	- / -	+	Абстрактный, конкретный	Процедурная	+	-
WaaS CP	XML-like	- / -	-	Абстрактный	Процедурная	+	-
Galaxy	XML-like	- / +	+	Абстрактный, конкретный	Процедурная	+	-
GJMB	XML-like	- / -	+	Абстрактный, конкретный	Процедурная	+	+
BPEL DP	BPEL	+ / +	-	Абстрактный, конкретный	Процедурная	+	+
OT	XML-like	+ / +	+	Абстрактный, конкретный	Процедурная, непроцедурная	-	-

Рассматриваемые системы используют XML-подобные или скриптовые языки для описания предметных областей приложений и самих НРП. Использование BPEL характеризуется рядом преимуществ, перечисленных выше, для спецификации НРП. Однако его языковых средств зачастую бывает недостаточно, чтобы описать сложную вычислительную модель. Поэтому целесообразно использовать комбинацию XML-подобного языка и BPEL.

Поддержка ветвлений и циклов в рабочем процессе позволяет при необходимости более гибко организовать вычисления. UNICORE, OT и BPEL DP предоставляет такую поддержку в полном объеме для построения НРП, не связанного с DAG.

Дополнительным преимуществом UNICORE, DAGMan, Pegasus, HyperFlow,

Galaxy, GJMB и OT является возможность включения системных операций в структуру рабочего процесса. К таким операциям относятся пред- и постобработка данных, мониторинг вычислительной среды, взаимодействие с компонентами систем управления прохождением заданий и т. д.

HyperFlow, WaaSCP, Galaxy, OT, GJMB и BPEL DP позволяют строить как абстрактные, так и конкретные НРП. При этом OT обеспечивает оба вида постановок задач, на основе которых выполняется построение НРП: процедурную и непроцедурную.

HyperFlow, WaaSCP, Galaxy, GJMB и BPEL DP поддерживают веб-сервисы. Кроме того, GJMB и BPEL DP поддерживают стандарт WPS.

Большая часть рассматриваемых WMS являются системами общего назначения и могут успешно применяться в различных предметных областях. Некоторые системы являются более специализированными. Например, Galaxy ориентирована на поддержку проведения геномных исследований, а GJMB и BPEL DP позволяют решать задачи в области геоинформатики.

В таблице 1.2 показаны возможности WMS по поддержке параллельных и распределенных вычислений в рамках выполнения НРП.

Таблица 1.2 – Возможности WMS для поддержки параллельных и распределенных вычислений и обработки данных

WMS	Уровень поддержки вычислений	Вычислительная среда	Связующее ПО	Поддержка IMDG	Поддержка непрерывной интеграции ПО	Поддержка контейнеризации ПО
UNICORE	Задача, данные	Grid	-	-	-	-
DAGMan	Задача, данные	Кластер, Grid, облачная среда	HTCondor 23.0.1	-	-	-
Pegasus	Задача, данные	Кластер, Grid, облачная среда	DAGMan	-	-	-

Таблица 1.2 – Возможности WMS для поддержки параллельных и распределенных вычислений и обработки данных (продолжение)

WMS	Уровень поддержки вычислений	Вычислительная среда	Связующее ПО	Поддержка IMDG	Поддержка непрерывной интеграции ПО	Поддержка контейнеризации ПО
AA	Задача, данные, конвейер	Облачная среда	Helm Chart for Kubernetes	-	-	-
HyperFlow	Задача	Облачная среда	-	-	-	-
WaaSCP	Задача	Grid, облачная среда	JClouds 0.3.0 API	-	-	-
Galaxy	Задача	Облачная среда	-	-	-	-
GJMB	Задача	Кластер, Grid, облачная среда	-	-	-	-
BPEL DP	Задача	Кластер, Grid, облачная среда	-	-	-	-
OT	Задача, данные, конвейер	Кластер, Grid, облачная среда	-	+	+	-

В рамках организации вычислений уровень задач означает, что задачи, определенные структурой рабочего процесса, выполняются на независимых вычислительных узлах. На уровне данных набор данных делится на подмножества. Каждое подмножество обрабатывается на отдельном вычислительном узле всеми или частью прикладных программных модулей, используемых в рабочем процессе. На уровне конвейера последовательное выполнение прикладных модулей для обработки данных выполняется одновременно на разных подмножествах набора данных. AA и OT поддерживают все уровни распараллеливания вычислений.

DAGMan, Pegasus, OT, GJMB и BPEL DP допускают использование всех

типов вычислительной среды. Однако системе Pegasus требуется интеграция с DAGMan для управления заданиями приложений на уровне среды. При этом только ОТ предоставляет системные операции для поддержки технологии IMDG на основе связующего ПО Apache Ignite.

Инструменты автоматизации непрерывной интеграции ПСПО предоставляются в ОТ. В то же время обеспечение автоматизации процессов контейнеризации ПО является не решенной в полной мере проблемой для всех известных WMS. Более детальная информация о широком спектре WMS приведена в Приложении В.

1.3. Развитие сервис-ориентированных приложений

Парадигма разработки и применения СОП рассмотрена в [10, 15]. Она представляет собой логическую эволюцию от объектно-ориентированных систем к системам сервисов. Как и в объектно-ориентированных системах, некоторыми фундаментальными концепциями веб-сервисов являются инкапсуляция, передача сообщений и динамическое связывание. Однако парадигма, основанная на сервисах, выходит за рамки сигнатур методов. Информация о функциях сервиса, его местоположении, способах доступа и др. также могут быть представлены в интерфейсе сервиса. Разработку СОП также можно рассматривать как эволюцию модульного подхода к программированию, т. к. веб-сервисы представляют собой легкие, слабосвязанные, независимые от платформы и языка компоненты.

В настоящее время в основе рассматриваемой парадигмы доминирует сервис-ориентированная архитектура (СОА, англ., Service-Oriented Architecture – SOA). СОА основывается на использовании множества независимых веб-сервисов, выполняющих предопределенные операции, связанных с выполнением системных или прикладных приложений. Под веб-сервисом (англ., Web service) понимается программная система со стандартизированными интерфейсами, идентифицируемая уникальным веб-адресом (URL-адресом) [62]. При этом веб-сервисы не обладают знаниями о выполняемых приложениях, а приложения не нуждаются в информации о способах их выполнения веб-сервисами. Веб-

технологии на основе СОА активно поддерживаются крупными компаниями-разработчиками, что обеспечивает их широкое распространение и использование.

Применительно к вычислительной среде СОА обуславливает ряд следующих важных преимуществ в процессах ее организации и применения [63]:

- многократное использование компонентов среды для построения сложных распределенных программных комплексов;
- модульный подход к разработке ПО;
- поддержку сетевого доступа к компонентам среды для разработчиков и пользователей, а также их взаимодействие между собой;
- обеспечение открытости среды за счет использования стандартов протоколов передачи данных и представления сервисных операций над этими данными;
- кроссплатформенность, позволяющую смягчить зависимость вычислительного процесса от используемых программно-аппаратных платформ и языков программирования;
- возможность безболезненной интеграции ПО разных разработчиков.

ПО, разработанное на основе СОА, как правило, реализуется в виде набора веб-сервисов, взаимодействующих по протоколу Simple Object Access Protocol (SOAP). Веб-сервис является единицей модульности в рамках СОА ПО. В то же время СОА может быть реализована с использованием широкого спектра дополнительных технологий, таких как REpresentational State Transfer (REST), Remote Procedure Call (RPC), Distributed Component Object Model (DCOM) и Common Object Request Broker Architecture (CORBA).

Основными форматами представления структурированных данных являются eXtensible Markup Language (XML) и JavaScript Object Notation (JSON), для которых поддерживается проверка корректности данных с помощью XML Schema и JSON Schema соответственно. Неструктурированные данные, как правило, представляются в виде текстовых файлов или файлов других форматов. Передача данных между веб-сервисами осуществляется посредством их включения в тело сообщения (в случае небольшого размера передаваемой информации) или путем

передачи адреса источника данных (англ., Uniform Resource Locator – URL), откуда их можно скачать (в случае большого размера данных).

Существуют разные способы описания веб-сервисов [64]. В их числе можно выделить Web Service Description Language (WSDL) для описания веб-сервисов на основе SOAP и Web Application Description Language (WADL) для описания веб-приложений на основе HyperText Transfer Protocol (HTTP), в том числе веб-сервисов в стиле REST. В обоих случаях в качестве базового языка описания используется XML. WSDL предназначен для описания веб-сервисов, доступа к ним и передаваемых между ними сообщений. Описание веб-сервиса на WSDL включает следующие основные разделы:

- определение типов данных, указывающих вид отправляемых и получаемых сервисом XML-сообщений, проверка которых осуществляется с помощью XML Schema;
- описание элементов данных – списка сообщений, используемых сервисом;
- задание абстрактных операций (портов) – списка методов, которые могут быть выполнены применительно к сообщениям;
- связывание сервисов – определение способов доставки сообщений;
- адрес вызова сервиса.

Последняя официальная спецификация языка WSDL Version 2.0 позволяет описывать как вызовы различных специализированных веб-сервисов на основе SOAP, например, WPS-сервисов, так и сервисов на основе других протоколов, например, REST-сервисов.

Разработка и применение СОП характеризуется рядом преимуществ по сравнению с другими типами приложений. Наличие в СОП набора сервисов позволяет разработчикам приложения создавать, отлаживать, тестировать, разворачивать и модифицировать свои сервисы независимо от других разработчиков, что упрощает распределенную разработку приложения. Каждый сервис может быть разработан и развернут на разных ресурсах с различными характеристиками (производительность, объемы оперативной и дисковой памяти,

пропускная способностью интерконнекта и т. п.) в Grid-системах, на ресурсах суперкомпьютеров или облачных платформ. С использованием контейнеризации сервисы можно запустить на нескольких параллельно работающих узлах без необходимости развертывания на новом узле всего приложения в целом.

Важным преимуществом СОП является их отказоустойчивость. Отказ одного сервиса, как правило, не приводит к отказу всего приложения в целом. При этом отказавший сервис может быть легко перезапущен или его операции могут быть выполнены другим сервисом при наличии вычислительной избыточности в СОП.

В таблице 1.3 приведены сведения о разработках в области сервис-ориентированных вычислений, тесно связанных с различными аспектами исследования в рамках диссертации.

Таблица 1.3 – Разработки в области сервис-ориентированных вычислений

Источник	Аспекты поддержки сервис-ориентированных вычислений	Уровень поддержки вычислений
[65-69]	Grid и облачные вычисления, SOAP-сервисы, Grid-сервисы, SaaS, Globus Toolkit.	Среда
[56]	Облачные вычисления, SOAP-сервисы, WMS, WaaS Cloud Platform	Приложение
[55]	Облачные вычисления, SOAP-сервисы, WMS, HyperFlow	Приложение
[70, 71]	Методика спецификации SOAP-сервисов, методика спецификации НРП, WSDL, BPEL	Приложение
[72, 73]	Микросервисы, композиции сервисов, синтез программ	Приложение
[74]	REST	Среда
[75]	Grid и облачные вычисления, Grid-сервисы, SOAP, REST, WMS, UNICORE	Среда
[76]	Облачные вычисления, SOAP, REST, WMS, Galaxy	Среда
[77]	Кооперативные вычисления, управление данными, микросервисы, iRODS	Среда, приложение

Таблица 1.3 – Разработки в области сервис-ориентированных вычислений (продолжение)

Источник	Аспекты поддержки сервис-ориентированных вычислений	Уровень поддержки вычислений
[78, 79]	Облачные вычисления, SOAP, REST, MathCloud, Everest	Среда, приложение
[80, 81]	Grid и облачные вычисления, SOAP-сервисы, Grid-сервисы, CAEBeans, испытательные стенды	Среда
[82, 83]	Облачные вычисления, НРП, веб-сервисы, интеллектуализация управления вычислениями, IaaS, SaaS, PaaS, iPSE	Среда
[84]	Сервис-ориентированные научные приложения	Приложение
[85, 86]	HPC, Amazon Web Services, Google Compute Engine, OpenStack, Cloud Stack, IaaS, PaaS, SaaS	Среда
[87-90]	Сервис-ориентированные научные приложения, обработка данных	Среда, приложение
[91, 92]	Сервис-ориентированные научные приложения, обработка данных	Среда, приложение
[93]	Исследования в энергетике	Среда
[94, 95]	Геоинформатика, REST-сервисы, SOAP-сервисы, WPS-сервисы, композиции сервисов	Среда
[96, 97]	REST-сервисы, SOAP-сервисы, микросервисы, композиции сервисов, мультиагентное управление	Приложение
[98]	SOAP-сервисы, композиции сервисов, WSDL	Среда, приложение
[99]	Мультиагентное управление ресурсами, микросервисы, шаблоны сервисов	Среда

1.4. Стандарты представления сервис-ориентированных рабочих процессов

В работах [10, 15] рассмотрены стандарты представления сервис-

ориентированных рабочих процессов. Сервис-ориентированные НРП обеспечивают возможность проведения научных экспериментов с использованием больших наборов данных. При этом процессы обработки данных распределяются на различные вычислительные ресурсы. НРП могут включать операции обнаружения и связывания ресурсов, а также сбора, обработки, анализа и визуализации данных. НРП должны быть логичными, структурированными и надежными.

Операции НРП выполняются в соответствии со схемой решения задачи в логической последовательности, определяемой его структурой. Применение стандартов для описания и выполнения НРП позволяет распространять их среди научного сообщества и облегчает их многократное использование. НРП могут быть размещены в общедоступных репозиториях.

Стандарты представления НРП берут свое начало в области моделирования бизнес-процессов. Соответствующие решения были разработаны рядом коммерческих организаций, таких как IBM и Microsoft. Открытые стандарты разрабатываются независимыми консорциумами, в том числе The World Wide Web Consortium (W3C), Organization for the Advancement of Structured Information Standards (OASIS), Workflow Management Coalition (WFMC), Business Process Management Initiative (BPMI), United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) и Object Management Group (OMG) [100]. Некоторые консорциумы концентрируют усилия на разработке комплектов дополнительных стандартов, тогда как другие разрабатывают отдельные многоцелевые стандарты. Пока еще нет единого мнения относительно того, какие стандарты наиболее подходят для СОП, а также не существует установленной структуры стандартов для СОА.

WMS часто характеризуются описанием процессов с точки зрения потока данных, а не ориентацией на поток управления бизнес-процессом. В рамках ряда проектов проводятся исследования, сравнивающие применимость различных стандартов описания НРП в СОП. Важным направлением является разработка стандартов, позволяющих учитывать требования к вычислениям и передаче

данных для больших наборов данных, а также обеспечивающих разделение абстрактного уровня представления НРП и уровня его выполнения на конкретных программно-аппаратных ресурсах. В целом успешная реализация НРП зависит от использования системы стандартов, каждый из которых обеспечивает эффективное планирование и выполнение вычислительных операций и операций обработки данных.

Различными коммерческими организациями и консорциумами предложен ряд стандартов описания сервис-ориентированных НРП [101, 102]. В том числе разработаны языки XML Process Definition Language (XPDL), XLANG, Web Services Flow Language (WSFL), Business Process Modeling Language (BPML), Business Process Specification Schema (BPSS), Web Services Conversation Language (WSCL), Web Services Choreography Interface (WSCI), Yet Another Workflow Language (YAWL), Business Process Execution Language for Web Services (BPEL4WS или BPEL) 1.0, BPEL4WS 1.1, Web Services Choreography Description Language (WS-CDL) и Web Services Business Process Execution Language (WS-BPEL или BPEL) 2.0.

XPDL, разработанный Workflow Management Coalition (WFMC), предназначен для обмена определениями процессов между различными информационными системами, как в графическом так и в семантическом виде. XPDL неоднократно пересматривался. Последняя ревизия состоялась в 2012 г.

XLANG от Microsoft является расширением WSDL. Его основное назначение заключается в определении бизнес-процессов и организации обмена сообщениями между веб-сервисами.

WSFL, разработанный компанией IBM, представляет собой XML-язык, описывающий бизнес-процесс в виде композиции веб-сервисов, в которой описывается последовательность вызовов операций сервисов. Порядок выполнения операций определяется на основе потоков управления и данных между сервисами. Бизнес-процесс определяет операции по получению, обработке и обмену данными в заданной последовательности.

BPML, предложенный BPMI, представляет собой язык описания бизнес-

процессов на основе XML. Он предоставляет средства выполнения последовательных и параллельных операций, поддерживает ветвления и циклы, обеспечивает стандартные функции вызова сервисов, отправки и получения сообщений, позволяет разработчику НРП планировать выполнение задач в соответствии с заданным расписанием. В BPML предусмотрено управление НРП с длительным сроком их выполнения.

BPSS представляет собой стандартную структуру, описывающую процесс обмена информацией. BPSS основана на метамодели UN/CEFACT. Эта схема позволяет предприятиям определять бизнес-транзакции и организовывать их сотрудничество между партнерами для электронного обмена документами и сигналами в коммерческих целях. BPSS входит в состав инструментария Electronic Business using XML (ebXML) от OASIS и UN/CEFACT.

WSCL от Hewlett-Packard предназначен для определения диалогов бизнес-уровня в виде общедоступных процессов, поддерживаемых веб-сервисами. WSCL определяет последовательность обмена XML-документами. Определения диалогов WSCL также являются XML-документами и поэтому могут интерпретироваться веб-сервисами.

YAWL представляет собой расширение XML и предназначен для формализованного описания бизнес-процессов. YAWL разработан в Техническом университете г. Эйндховен. Также нидерландскими учеными была разработана специализированная программная платформа, поддерживающая текстовый и графический режим построения бизнес-процессов и средства их выполнения. Исходный код ПО распространяется под лицензией GNU Lesser General Public License (LGPL).

WS-CDL создан W3C на основе XML для спецификации однорангового взаимодействия веб-сервисов на основе хореографии – упорядоченного обмена сообщениями между внешними сущностями. Спецификации сервисов определяют связи между гетерогенными вычислительными средами, используемыми для разработки и размещения веб-приложений. В целом обеспечение хореографии веб-сервисов позволяет организовать функционально совместимое одноранговое

взаимодействие между любыми сервисами, независимо от поддерживающей платформы или модели программирования, используемой при их реализации.

К 2003 г. назрела необходимость перехода от разнородных стандартов отдельных консорциумов к некоторому единому стандарту. Усилиями IBM, Microsoft, BEA, OASIS и др. консорциумов началась разработка и развитие BPEL. С развитием веб-сервисов произошло слияние WSFL и XLANG, что привело к появлению нового поколения языка спецификаций BPEL4WS 1.1. Язык BPEL4WS 1.1 позволил расширить модель взаимодействия веб-сервисов и сделать ее применимой для отображения бизнес-транзакций.

С появлением BPEL список стандартов, широко востребованных на практике, сокращается. Формально считается, что BPEL и XPDL обеспечивают оркестрацию взаимодействий между внутренними и внешними сущностями процессов, а WS-CDL и ebXML – хореографию. Однако функциональные возможности BPEL и XPDL позволяют описывать и хореографию.

В целом BPEL 2.0 [103] определяет модель для описания поведения сервис-ориентированных НРП в терминах взаимодействий (совокупности сообщений) между процессами и их партнерами (внешними сервисами). Важными являются следующие дополнительными преимуществами BPEL:

- НРП могут не только вызывать веб-сервисы, но и сами быть представленными в виде сервисов;
- широкий спектр элементов управления и работы с данными, включающий элементы определения сложных структур данных и параллельных процессов их обработки, циклы, ветвления, подпроцессы, элементы реализации асинхронного взаимодействия веб-сервисов и др.;
- использование WSDL для описания интерфейсов веб-сервисов обеспечивает гибкую интеграцию с другими НРП и веб-приложениями;
- детальное описание НРП реализует оркестровку внутренних и внешних сущностей процесса, а спецификация процесса обмена сообщениями отражает хореографию внешних сущностей (вызываемых веб-сервисов).

В дополнение к перечисленным выше стандартам неформальная рабочая группа, объединяющая представителей различных организаций и частных лиц, которые заинтересованы в переносимости рабочих процессов, разрабатывает язык Common Workflow Language (CWL) [104]. Цель работы группы заключается в создании спецификаций, позволяющих представителям научного сообщества описывать мощные, простые в использовании, портативные и поддерживающие воспроизводимость рабочие процессы. При представлении ряда конструкций рабочих процессов (например, конвейеров) CWL использует возможности языков YAML Ain't Markup Language (YAML) и JSON. Для контейнеризации прикладного ПО в портативных средах выполнения применяется система Docker [105]. Она предназначена для описания НРП с интенсивным использованием данных в областях исследований, таких как биоинформатика, медицина, химия, физика и астрономия. Версия 1.0 языка CWL выпущена 8 июля 2016 г.

Разработка и развитие стандартов описания веб-сервисов представлены в ретроспективе на рисунке 1.3. Рассмотренное на данном рисунке развитие языков спецификации веб-сервисов обобщает, уточняет и дополняет диаграммы из [70, 71].

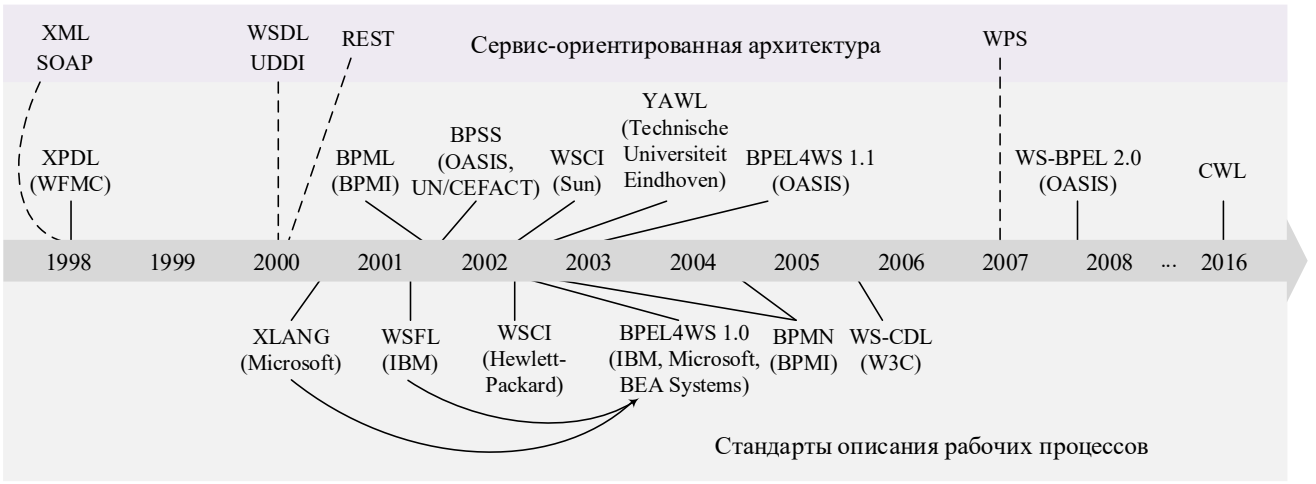


Рисунок 1.3 – Развитие языков спецификации веб-сервисов

1.5. Виртуализация и контейнеризация программного обеспечения

Для выполнения СОП требуется большое число разнообразного ПО, которое может быть несовместимо между собой, а также иметь несовместимые версии

используемых библиотек и другие зависимости. Как правило для создания изолированной среды выполнения СОП применяется виртуализация и контейнеризация.

Одним из первых подходов к распределению вычислительных ресурсов между несколькими пользователями была концепция разделения времени. В конце 1960-х годов из разделения времени (time-sharing) выросла концепция виртуальной машины (ВМ) как эффективной, изолированной копии реальной машины [106]. В ней ПО для управления ВМ получило название монитора виртуальных машин (ВМ), а в настоящее время используется термин «гипервизора». Выделяют 2 типа гипервизоров. Гипервизор первого типа устанавливается непосредственно на аппаратное обеспечение компьютера, что обеспечивает повышенную безопасность, а также производительность ВМ за счет прямого взаимодействия с аппаратным обеспечением компьютера для распределения выделенных ресурсов. Гипервизоры второго типа устанавливаются поверх существующей (основной, хостовой) ОС машины, с которой происходит согласование распределения выделенных ресурсов для ВМ. Поскольку основная ОС отдает приоритет собственным функциям и приложениям, а не функциям и приложениям ВМ, то их производительность значительно ниже, чем при использовании гипервизоров первого типа. К гипервизорам первого типа относятся такие продукты как VMware ESXi, Microsoft Hyper-V, KVM. Примерами гипервизоров второго типа служат виртуальная машина Oracle VirtualBox, рабочая станция VMware, виртуальный ПК Microsoft.

Первым этапом на пути к контейнеризации стало появление системного вызова chroot [107] в 7-ой версии ОС Unix, разработанной в 1979 году AT&T и Bell Laboratories. Он позволял изменять местоположение корневой папки процесса и его дочерних элементов. Это стало основой для изоляции процессов и разделения доступа к файлам для каждого из них. Каждому запущенному в системе процессу выделялся собственный объем памяти и определялись файловые дескрипторы. В 1982 году chroot был включен в состав BSD, что позволило использовать аналогичные механизмы изоляции в производных системах. Появление технологий

межпроцессного взаимодействия (IPC) стало еще одним важным этапом на пути к контейнеризации.

В 2000 году частной компаний R&D Associates была разработана технология Jail [108] для ОС FreeBSD, которая стала следующим шагом на пути к появлению контейнеризации. Эта технология позволяла запускать внутри FreeBSD еще несколько экземпляров FreeBSD. Они использовали то же ядро, но собственные независимые окружения, набор приложений. Создаваемая таким образом изолированная система использовала локальный набор файлов, процессов, учетных записей пользователей и суперпользователей. В версию FreeBSD 7.2 разработчики добавили поддержку нескольких адресов IPv4 и IPv6 для таких изолированных систем и возможность их привязки к определенным процессорам на многопроцессорных серверах. Аналогом технологии Jail для ОС Linux стала система виртуализации VServer, разработанная в 2001 году.

Дальнейшим развитием контейнеризации стала система виртуализации Solaris Containers в ОС Solaris 10 от Sun Microsystems. В терминологии Sun Microsystems такие изолированные системы для запуска ОС назывались «зонами». Для таких «зон» появились инструменты управления системными ресурсами, допускающие создание их «моментальных снимков» и клонирование. Зоны являлись полностью изолированными виртуальными серверами внутри основной ОС. Для каждой такой ОС задавалось сетевое имя, выделялись сетевые интерфейсы, определялась собственная файловая система, а также набор пользователей (включая root) и конфигурация.

В 2006 году корпорацией Google была разработана собственная система разделения ресурсов под названием Process Containers [109]. Затем она была переименована в Control Groups и включена в ядро Linux. Эта система представляла собой группу процессов, для которых на уровне ОС накладывались ограничения на использование различных ресурсов — памяти, ввода-вывода, сети. Такие группы процессов можно было объединять в иерархические системы и управлять ими. В 2008 году для ОС Linux была разработана контейнерная система LXC [110], которая обеспечивала полную виртуализацию на уровне ОС. Разработка системы

контейнеризации Docker началась в 2011 году. Изначально она подразумевалась как коммерческий сервис для облачных инфраструктур, который предоставлялся клиентам по модели SaaS. После 2013 года, когда Docker был представлен широкой публике, он стал основоположником целой экосистемы для управления контейнерами. Корпорация Google в 2014 году начала разработку Kubernetes [111], которая стала одной из наиболее популярных систем для автоматизации развертывания, масштабирования и управления контейнеризированными приложениями промышленного уровня.

В диссертации рассматривается научная задача подготовки и проведения вычислительных экспериментов, выполняемых СОП в ПОВС. Согласно базовым критериям пользователей и владельцев ресурсов накладные расходы на захват и освобождения ресурсов изолированной средой должны быть минимальны. Поэтому для выбора способа построения такой среды необходимо сравнить характеристики ВМ и контейнеров, их достоинства и недостатки применительно к задаче, поставленной в диссертации. При этом не рассматриваются ВМ второго типа, т.к. в этом случае хостовая ОС отдает приоритет не этим машинам, а своим функциям и приложениям.

На рисунке 1.4 изображена архитектура ВМ и контейнеров.

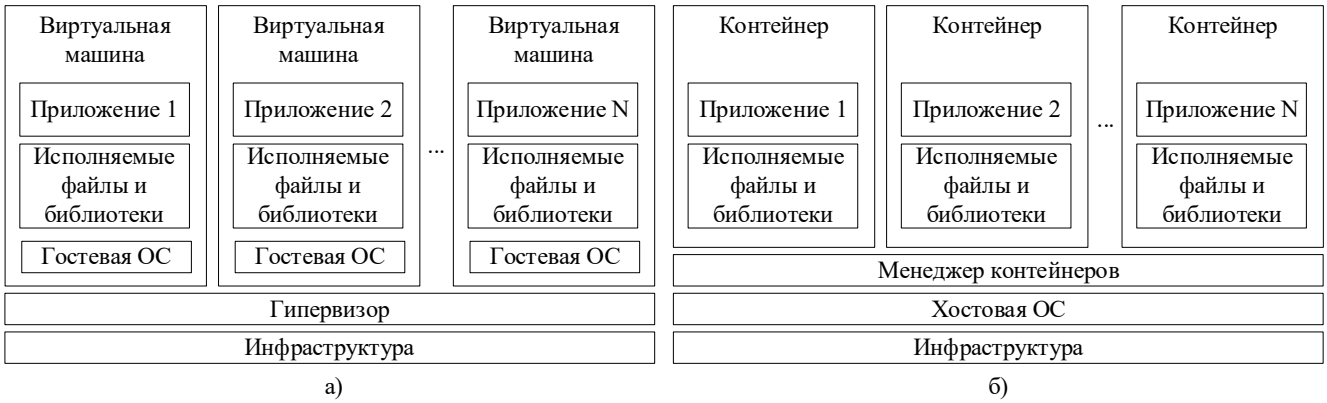


Рисунок 1.4 – Архитектура ВМ (а) и контейнеров (б)

В таблице 1.4 приведены характеристики виртуальных машин и контейнеров, рассмотренных в [112, 113].

Таблица 1.4 – Характеристики ВМ и контейнеров

Характеристика	Виртуальные машины	Контейнеры
Уровень изоляции	Полная изоляция на уровне аппаратного обеспечения (гипервизор).	Изоляция на уровне операционной системы (ядро ОС).
Операционная система	Требует гипервизор. Каждая ВМ имеет свою ОС.	Использует ядро хостовой ОС. Контейнеры делят ядро хоста.
Скорость запуска	Низкая скорость запуска из-за необходимости полной загрузки ОС	Высокая скорость запуска процессов в изолированном окружении
Размер образа	Образ имеет большой размер, который включает полную ОС	Образ имеет небольшой размер, который включает только приложение и зависимости
Портативность	Зависит от гипервизора и ОС	Контейнеры упакованы со всеми зависимостями и легко переносятся между разными ПК
Безопасность	Полная изоляции окружения	Изоляция на уровне ОС, поэтому уязвимости ядра могут затрагивать все контейнеры
Ресурсы	Каждая ВМ требует отдельную ОС, ресурсы памяти, CPU	Контейнеры разделяют между собой ресурсы хостовой ОС и не требуют установки отдельных ОС
Обмен файлами между хостовой и гостевой ОС	Требуется настройка гипервизора	При запуске контейнера указывает общий том с хостовой ОС
Масштабируемость	Присутствуют высокие накладные расходы на запуск ВМ	Можно быстро запускать и останавливать сотни контейнеров

Как видно из таблицы 1.4, ВМ по величине накладных расходов на захват и освобождение ресурсов превосходят контейнеры, что не соответствует

рассматриваемой в диссертации задаче. В [112] авторы выполняют сравнение производительности между ВМ и контейнерами в облачных средах AWS, Google Cloud и Azure на кредитных моделях глубокого обучения. Полученные результаты показали, что технология контейнеров Docker имеет существенное преимущество перед технологиями виртуализации KVM и Xen во всех трех облачных средах, использованных в исследовании. В работе [114] авторы сравнивают контейнеры Docker и ВМ KVM с точки зрения производительности процессора, памяти, дискового ввода/вывода, нагрузочного тестирования и измерения скорости работы. В каждом из тестов контейнеры Docker показывают значительно лучше результат в производительности, чем ВМ KVM. Как отмечают авторы наличие слоя QEMU в ВМ делает ее менее эффективной, чем контейнеры Docker.

Поэтому для создания вычислительной среды и интеграции ПСПО в диссертации будут использоваться контейнеры. Также необходимо выбрать технологию контейнеризации. Различают два вида контейнеров: контейнеры ОС и контейнеры приложений.

Контейнеры ОС являются виртуальной средой, которая совместно использует ядро ОС хоста, но обеспечивает изоляцию пространства пользователя. Это позволяют устанавливать, конфигурировать и запускать различные приложения, как и в любой ОС. Они полезны, когда необходимо запустить целый парк идентичных или разных дистрибутивов. В большинстве случаев контейнеры ОС создаются из шаблонов или образов, которые определяют структуру и содержимое контейнера. Таким образом, это позволяет создавать контейнеры с идентичными средами, одинаковыми версиями пакетов и конфигурациями во всех контейнерах. Создание контейнеров ОС обеспечивают такие технологии контейнеров, как, например, LXC, OpenVZ, Linux VServer, BSD Jails и зоны Solaris.

Контейнеры приложений, такие как Docker, существенно отличаются от контейнеров ОС. Когда создается контейнер, он запускает одно приложение. В Docker каждая команда RUN создает новый слой для контейнера. После запуска контейнера Docker объединяет все слои и запускает сам контейнер. Слои Docker помогают уменьшить дублирование и увеличить повторное использование

контейнеров. Это полезно, когда необходимо создать разные контейнеры для компонентов приложения. Начиная с базового образа, который является общим для всех компонентов, можно добавлять слои, специфичные для каждого из компонентов. Слои обеспечивают эффективный механизм отката изменений в контейнере, позволяя переключится на старые слои. При этом накладные расходы на переключения между слоями минимальны.

В рамках диссертации использование контейнеров приложений для разработки и применения СОП позволяет решить следующие задачи:

- изолировать ПСПО нескольких СОП в рамках одной ОС;
- конфигурировать и интегрировать ПСПО в виде образа, который можно перемещать между различными вычислительными ресурсами;
- исключить конфликты между версиями программных библиотек в хостовой ОС и контейнере.

1.6. Функциональные и системные требования к разработке и применению сервис-ориентированных научных рабочих процессов

В данном разделе с учетом современных тенденций создания распределенных приложений и результатов сравнительного анализа процессов построения НРП в известных WMS сформулированы функциональные и системные требования к ИК для разработки и применения сервис-ориентированных научных рабочих процессов.

Функциональные требования. Разрабатываемый ИК должен обеспечивать следующие функциональные возможности [10, 15, 28]:

- предоставление дружественного пользовательского интерфейса для взаимодействия с компонентами ИК и разрабатываемыми СОП разным категориям пользователей (разработчикам, администраторам и пользователям), поддерживающего как текстовые языки представления структурированных данных, так и языки для ввода информации с помощью

веб-форм с последующим автоматическим конвертированием полученных спецификаций на соответствующие текстовые языки;

- корректное описание вычислительной модели предметной области (параметров, операций, логических выражений, продукций и модулей, вычислительных ресурсов, элементов вычислительной истории, характеристик административных политик управления ресурсами и др., а также отношений между перечисленными объектами);
- поддержка сложных структур данных, таких как составные параметры и параллельные списки данных, а также неструктурированных файловых данных;
- статическое планирование вычислений по непроцедурной постановке задачи, построение НРП по процедурной постановке задачи с использованием различных управляющих конструкций ветвления и циклов, а также статико-динамическое планирование вычислений и распределение вычислительной нагрузки на ресурсы при работе с параллельными списками данных;
- обеспечение средств разработки и применения сервис-ориентированных НРП при сохранении поддержки традиционных рабочих процессов с исполняемыми модулями;
- возможность включения в состав НРП системных операторов обработки и анализа данных, статико-динамического планирования, конвейеризации вычислений, параллельного выполнения алгоритмов;
- генерация программ выполнения НРП на языке программирования общего назначения с целью их последующего автономного выполнения;
- использование стандарта языка представления сервис-ориентированных НРП;
- автоматизация поддержки интеграции, тестирования и контейнеризации ПСПО разрабатываемых СОП;

- автоматизация использования технологии IMDG на ресурсах ПОВС при выполнении НРП;
- интеграция ресурсов, различных вычислительных инфраструктур, таких как ЦКП, кластерные Grid и облачные платформы, при разработке и выполнении СОП;
- визуализация НРП и расчетных данных, в том числе предоставление набора типовых спецификаций для диаграмм и графиков;
- поддержка разработки испытательных стендов СОП, основой которых являются системные рабочие процессы (СРП), включающие широкий спектр системных библиотек для интеграционного тестирования НРП, анализа их работы и проверки взаимодействия с различными источниками данных.

Системные требования к разрабатываемому ИК:

- функционирование на ресурсах под управлением ОС Ubuntu 20.04 и выше;
- работа с БД MySQL, PostgreSQL;
- поддержка протоколов передачи данных HTTP, SOAP;
- использование Docker как средства контейнеризации ПСПО;
- взаимодействие ИК с контрольно-измерительными системами, функционирующими на ресурсах ПОВС.

В целом набор перечисленных выше функциональных и системных требований к ИК не поддерживается в полной мере в известных WMS. Поэтому реализация этих требований расширяет сферу практического применения данного ИК по сравнению с известными WMS.

1.7. Выводы

В первой главе получены следующие основные результаты:

- приведены основные понятия и определения, связанные с организацией распределенных вычислений в ПОВС;
- проведен сравнительный анализ известных систем управления НРП;

- обсуждены тенденции современного развития распределенных научных приложений на основе сервис-ориентированной парадигмы;
- рассмотрены известные стандарты представления сервис-ориентированных рабочих процессов;
- обсуждены особенности и преимущества виртуализации и контейнеризации ПО;
- с учетом современных тенденций создания распределенных приложений и результатов сравнительного анализа процессов построения НРП в известных WMS сформулированы функциональные и системные требования к ИК для разработки и применения сервис-ориентированных научных рабочих процессов.

Результаты исследований по данной главе опубликованы в [10, 11, 12, 15, 28, 30]. Терминология, связанная с понятиями вычислительной модели СОП, базируется на понятиях из работы [119]. Особенности РППП рассмотрены в работах [11, 12]. Результаты сравнительного анализа систем управления НРП представлены в работах [11, 30]. Вопросы стандартизации представления сервис-ориентированных НРП, парадигма разработки и применения СОП, а также функциональные требования к разрабатываемому ИК изложены в работах [10, 15, 28, 30].

Глава 2. Модели и алгоритмы управления вычислениями

В данной главе предлагается вычислительная модель СОП. Описываются алгоритмы, разработанные для планирования вычислений и условий решения задачи, информационного планирования, распределения вычислительной нагрузки и генерации вычислительных заданий на предложенной вычислительной модели. Рассматривается схема взаимодействия исполнительной подсистемы разрабатываемого ИК с метапланировщиком и СУПЗ в ПОВС.

2.1. Вычислительная модель

Предложенная в диссертации вычислительная модель СОП [10, 15, 28] является развитием моделей, представленных в [115-120]. Она описывается в виде структуры

$$\text{M} =< L_{ab}, L_{al}, L_{so}, L_{pa}, L_{cn}, R, Rc >,$$

$$L_{ab} =< Z, T, O, Op, Pr, St, W, Tb >,$$

$$L_{al} =< M, Pa >,$$

$$L_{so} =< Sr, Ms >,$$

$$L_{pa} =< J, N, Lrm, Sch, Us, Ap, Q, Cm, H >,$$

$$L_{cn} =< Cn, Sc >,$$

элементы которой интерпретируются следующим образом:

- L_{ab} – структура, представляющая элементы абстрактного уровня;
- L_{al} – структура, представляющая элементы алгоритмического уровня;
- L_{so} – структура, представляющая элементы сервис-ориентированного уровня;
- L_{pa} – структура, представляющая элементы программно-аппаратного уровня;
- L_{cn} – структура, представляющая элементы уровня контейнеризации;
- R – множество связей, в общем случае типа «многие-ко-многим», между объектами структур L_{ab}, L_{al}, L_{so}, L_{pa} и L_{cn};

- Rc – множество конфигурационных связей;
- Z – множество значимых параметров предметной области приложения;
- T – множество их допустимых типов данных;
- O – совокупность абстрактных вычислительных операций и операций обработки данных, отражающих семантику алгоритмических знаний в модели;
- Op – множество операторов, выполняемых над операциями;
- Pr – множество продукций, определяющих условия выполнения операций;
- St – множество постановок задач, формулируемых в процедурной или непроцедурной форме;
- W – множество НРП, построенных на основе процедурных или непроцедурных постановок задач;
- Tb – множество НРП, предназначенных для тестирования ПО и предметных данных;
- M – множество программных модулей, представляющих алгоритмические знания предметной области и реализующих операции;
- Pa – множество программ, сгенерированных для автономного выполнения НРП;
- Sr – множество сервисов, реализующих абстрактные операции и НРП;
- Ms – множество сообщений, передаваемых между сервисами;
- J – множество заданий по выполнению НРП;
- N – множество ресурсов ПОВС, на которых размещаются и выполняются модули и сервисы приложения;
- Lrm – множество систем управления прохождением заданий, установленных на ресурсах;
- Sch – множество метапланировщиков;
- Us – множество пользователей ресурсов ПОВС;
- Ap – множество административных политик, применяемых к ресурсам;
- Q – множество квот на использование ресурсов пользователями;

- Cm – множество телекоммуникационных каналов, соединяющих ресурсы;
- H – вычислительная история выполнения ПО;
- Cn – множество конфигураций контейнеризированной среды;
- Sc – множество сценариев контейнеризации.

НРП строятся на основе как процедурной, так и непроцедурной постановок задач, описанных в [119]. НРП могут представлять как традиционные схемы решения задач в виде последовательно-параллельных цепочек операций, так и композиции сервисов.

В общем случае множества Z, O, W, M, Pa и Sr включают подмножества прикладных и системных объектов. Прикладные объекты создаются разработчиком приложения и дополняются предопределенными системными объектами, предназначенными для поддержки взаимодействия модулей, сгенерированных программ, сервисов и НРП с компонентами инструментального комплекса для разработки приложений, внешними ресурсами и системами при подготовке и проведении экспериментов.

Вычислительная модель определяет понятия и связи между объектами ПОВС абстрактного уровня, алгоритмического уровня, сервис-ориентированного уровня, программно-аппаратного уровня и уровня контейнеризации (рисунок 2.1).

Она позволяет разработчикам и пользователям приложений взаимодействовать с ПОВС и управлять ее компонентами на абстрактном уровне, который во многом скрывает детали организации вычислительных процессов на других уровнях.

Элементы L_{ab}, L_{al} и L_{pa} вычислительной модели, рассматриваемые в диссертации, интерпретируются аналогично соответствующим элементам вычислительных моделей, представленными в [118, 119]. В диссертации множество Op расширено новыми операторами агрегирования и дезагрегирования параллельных списков данных, параллельного цикла, явного и неявного многометодного анализа. Соответственно, множество M расширено модулями, представляющими алгоритмы реализации перечисленных выше операторов.

Элементы L_{so} в целом отражают соответствующие элементы модели сервис-ориентированной информационно-аналитической среды для описания композиции сервисов [120]. Дополнительные конфигурационные связи задаются ссылками на конфигурационные файлы объектов ПОВС. Кроме того, предложенная модель расширена принципиально новой структурой L_{cn}, представляющей элементы уровня контейнеризации ПО, а также конфигурационными связями между ее объектами. Конфигурация контейнеризированной среды представляет собой метаописание сценариев интеграции и контейнеризации прикладного и системного ПО. Сценарий контейнеризации в общем случае включает набор инструкций по выполнению скриптов для развертывания системы Docker, сборки Docker-образа и запуска контейнера на основе собранного Docker-образа.

Рисунок 2.1 – Связи между уровнями вычислительной модели

Рассмотрим основные объекты вычислительной модели, использующей реляционную модель данных. В таблице 2.1 приведены типы и структуры данных параметров вычислительной модели.

Таблица 2.1 – Типы и структуры данных параметров и их дополнительные атрибуты

Тип	Структура данных	Размерность 1	Размерность 2	Базовый параметр	Список параметров
integer | float | double | char | Boolean | string | const	Скаляр	-	-	-	-
integer | float | double | char | Boolean | string | const	Вектор	Скалярный параметр типа integer	-	-	-
integer | float | double | char | Boolean | string | const	Матрица	Скалярный параметр типа integer	Скалярный параметр типа integer	-	-
-	Файл	-	-	-	-
-	Составной параметр	-	-	-	Список параметров, входящих в составной параметр
-	Параллельный список данных[1]	Скалярный параметр типа integer	-	Базовый параметр, на основе которого создан параллельный список данных любого структурного типа данных	Список параметров – экземпляров базового параметра

В таблице 2.1 размерности векторов и матриц, представленных соответственно одномерными и двумерными массивами, определяются

[1] Структура данных, допускающая независимую параллельную обработку ее элементов (см., например, понятия параллельных списков в работах [121, 122]).

скалярными параметрами типа integer.

Определенные типы и структуры данных обуславливают наличие дополнительных атрибутов параметров, таких как размерности массивов данных, списки взаимосвязанных параметров и базовые параметры, на основе которых создаются списки их экземпляров.

В множестве $Z = \{z_1, z_2, \dots, z_n\}$ параметров выделяются следующие подмножества:

- $Z^b = \{z_{i_1}, z_{i_2}, \dots, z_{i_b}\}$ – подмножество логических параметров типа Boolean, $i_1, i_2, \dots, i_b \in \overline{1,n}$, $Z^b \subset Z$,
- $Z^s = \{z_{j_1}, z_{j_2}, \dots, z_{j_s}\}$ – подмножество составных параметров, $j_1, j_2, \dots, j_s \in \overline{1,n}$, $Z^s \subset Z$,
- $Z^f = \{z_{k_1}, z_{k_2}, \dots, z_{k_f}\}$ – подмножество файлов, $k_1, k_2, \dots, k_f \in \overline{1,n}$, $Z^f \subset Z$,
- $Z^l = \{z_{q_1}, z_{q_2}, \dots, z_{q_l}\}$ – подмножество параллельных списков данных, $q_1, q_2, \dots, q_l \in \overline{1,n}$, $Z^l \subset Z$,
- $Z^d = \{z_{r_1}, z_{r_2}, \dots, z_{r_d}\}$ – подмножество параметров, передаваемых через базу данных, $r_1, r_2, \dots, r_d \in \overline{1,n}$, $Z^d \subset Z$.

Перечисленные подмножества Z^b, Z^s, Z^f и Z^l не пересекаются между собой $Z^b \cap Z^s = \emptyset$, $Z^b \cap Z^f = \emptyset$, $Z^b \cap Z^l = \emptyset$, $Z^s \cap Z^f = \emptyset$, $Z^s \cap Z^l = \emptyset$, $Z^f \cap Z^l = \emptyset$.

Параметры из Z^b, Z^s, Z^f и Z^l могут передаваться через базу данных.

Пусть X_i и Y_i – это соответственно множества входных и выходных параметров операции o_i. Введем следующие виды базовых операций: вычислительную и логическую операции, а также операцию обработки параллельного списка данных.

Определение 1. Операцию

$$o_i: X_i \to Y_i = Y_i \tag{1}$$

будем называть базовой вычислительной операцией, если для нее выполняются следующие условия:

$$X_i, Y_i \subset Z, \;\; X_i \cap Y_i = \emptyset, \;\; Y_i \cap Z^b = \emptyset, \;\; (X_i \cap Z^l = \emptyset) \vee (Y_i \cap Z^l = \emptyset). \tag{2}$$

Определение 2. Операцию (1) будем называть базовой логической операцией, если для нее выполняются следующие условия:

$$X_i, Y_i \subset Z, \;\; X_i \cap Y_i = \emptyset, \;\; Y_i \cap Z^b \neq \emptyset, \;\; (X_i \cap Z^l = \emptyset) \vee (Y_i \cap Z^l = \emptyset). \qquad (3)$$

Определение 3. Операцию (1) будем называть базовой операцией обработки параллельного списка данных, если для нее выполняются следующие условия:

$$X_i \subset Z, \;\; Y_i \subset Z^l \subset Z, \;\; X_i \cap Y_i = \emptyset, \;\; X_i \cap Z^l = \{x_j\}, \;\; Y_i \cap Z^l = \{y_k\}, |Y_i| = 1, \quad (4)$$

где $x_j =< x_l, m >$ и $y_k =< y_q, m >$ – это параллельные списки данных, m – целочисленный параметр, представляющий число экземпляров параметров x_l и y_q, на основе которых созданы параллельные списки данных x_j и y_k.

Обозначим множества вычислительных и логических операций, а также операций обработки параллельных списков данных соответственно, через O_c, O_l и O_p, $O = O_c \cup O_l \cup O_p$.

Определение 4. Две базовые операции o_i и o_j будем считать операциями, выполняющими избыточные вычисления, если для них выполняются условия определения базовых операций и следующие дополнительные условия:

$$X_i = X_j, \;\; Y_i = Y_j, \qquad (5)$$

где $o_i, o_j \in O$.

В случае выполнения избыточных вычислений операциями o_i и o_j предполагается, что они представляют разные алгоритмы вычисления своих выходных параметров из Y_j по заданным входным параметрам их X_j. Различия в формальных параметрах исходных программ, реализующих эти операции, могут экранироваться с помощью дополнительных программных оболочек, обеспечивающих необходимую параметрическую настройку этих программ.

Формулы (1)-(4) определяют соответственно модели вычислительных и логических операций, а также операций обработки параллельных списков данных. При этом условия (2)-(5) являются условиями корректности определения базовых операций в вычислительной модели.

В рамках диссертации вышеупомянутые модели базовых операций применяются при выполнении следующих работ:

- статическом планировании вычислений по непроцедурной постановке задачи (p_1);
- динамическом планировании вычислений по непроцедурной постановке задачи (p_2);
- планировании условий решения задачи по ее процедурной постановке (p_3);
- интерпретации НРП (p_4);
- формировании остаточного НРП при сбоях и отказах программно-аппаратных ресурсов в процессе выполнения рабочего процесса (p_5).

Составные операции вычислительной модели отражают схему вычислений, выполняемых с помощью НРП. Их множества входных и выходных параметров совпадают с множествами входных и выходных параметров НРП, которые формируются с помощью моделей [123] планирования условий решения задачи. В рамках диссертации эти модели дополнены и модифицированы.

Пусть X_w и Y_w – это множества входных и выходных параметров составной операции w. Введем следующие виды составных операций: агрегирования и дезагрегирования данных, параллельного выполнения экземпляров базовой операции, параллельного выполнения операций, последовательного выполнения операций, выполнения цикла типа for-повторения операций, выполнения цикла типа do-while-повторения операций, выполнения параллельного цикла повторения операций, выполнения ветвления, а также выполнения явного и неявного многометодного анализа.

Определение 5. Операцию

$$w: X_w \to Y_w = \left(x_j \Rightarrow x_k\right) = Y_w \tag{6}$$

будем называть составной операцией дезагрегирования параметра x_j, представляющего в вычислительной модели параллельный список данных, в список экземпляров $x_{k_1}, x_{k_2}, \dots, x_{k_m}$ базового параметра x_k, на основе которого создан x_j, при выполнении следующих условий:

$$x_j \in Z^l, \;\; x_k \in Z, \tag{7}$$

где $X_w = \{x_j\}$, $Y_w = \{x_k\}$, $x_j =< x_k, m >$, $x_k = \left(x_{k_1}, x_{k_2}, \dots, x_{k_m}\right)$, m –

целочисленный параметр, представляющий число элементов параллельного списка данных x_j (число вариантов значений x_k).

Определение 6. Операцию

$$w: X_w \to Y_w = \left(x_k \Rightarrow x_j\right) = Y_w \tag{8}$$

будем называть составной операцией агрегирования списка экземпляров $x_{k_1}, x_{k_2}, \dots, x_{k_m}$ базового параметра x_k в параметр x_j, представляющего в вычислительной модели параллельный список данных, при выполнении следующих условий:

$$x_j \in Z^l, \quad x_k \in Z, \tag{9}$$

где $X_w = \{x_k\}$, $Y_w = \{x_j\}$, $x_j =< x_k, m >$, $x_k = \left(x_{k_1}, x_{k_2}, \dots, x_{k_m}\right)$, m – целочисленный параметр, представляющий число элементов параллельного списка данных x_j (число вариантов значений x_k).

Определение 7. Операцию

$$w: X_w \to Y_w = \left(o_{i_r}\right) \overset{r=\overline{1,m}}{\longleftrightarrow} = Y_w \tag{10}$$

будем называть составной операцией параллельного выполнения экземпляров базовой операции o_i обработки параллельного списка данных при удовлетворении условиям определения операций обработки параллельного списка данных и следующим дополнительным условиям:

$$o_i \in O_p, \tag{11}$$

где $X_w = \{x_1, \dots, x_{j-1}, x_l, x_{j+1}, \dots, x_n\}$, $Y_w = \{y_q\}$, $X_i = \{x_1, \dots, x_{j-1}, x_{l_r}, x_{j+1}, \dots, x_n\}$ и $Y_i = \left\{y_{q_r}\right\}$ множества входных и выходных параметров *r*-го экземпляра операции o_i, $x_j =< x_l, m >$ и $y_k =< y_q, m >$ – параллельные списки данных, x_{l_r} – *r*-й вариант значения $x_l = \left(x_{l_1}, x_{l_2}, \dots, x_{l_m}\right)$, y_{q_r} – *r*-й вариант значения $y_q = \left(y_{q_1}, y_{q_2}, \dots, y_{q_m}\right)$, m – число элементов параллельных списков данных x_j и y_k (число вариантов значений x_l и y_q), n – число параметров в X_i.

Определение 8. Операцию

$$w: X_w \to Y_w = \left(o_i - o_j\right) = Y_w \tag{12}$$

будем называть составной операцией последовательного выполнения базовых операций o_i и o_j при удовлетворении условиям определения базовых операций и следующему дополнительному условию:

$$X_i \cap Y_j = \emptyset, \tag{13}$$

где $X_w = X_i \cup X_j \cap \bar{Y}_i$, $Y_w = Y_i \cup Y_j$.

Определение 9. Операцию

$$w: X_w \to Y_w = \left(o_i || o_j\right) = Y_w \tag{14}$$

будем называть составной операцией параллельного выполнения базовых операций o_i и o_j при удовлетворении условиям определения базовых операций и следующим дополнительным условиям:

$$\left(X_i \cup X_j\right) \cap \left(Y_i \cup Y_j\right) = \emptyset, \;\; Y_i \cap Y_j = \emptyset, \tag{15}$$

где $X_w = X_i \cup X_j$ и $Y_w = Y_i \cup Y_j$.

Определение 10. Операцию

$$w: X_w \to Y_w = \left(o_i - o_j\right)^{r}\!\leftrightarrow = Y_w \tag{16}$$

будем называть составной операцией выполнения цикла типа for-повторения базовых операций o_i и o_j при удовлетворении условиям определения базовых операций и следующим дополнительным условиям:

$$X_i \cap Y_j \neq \emptyset, \;\; r \geq 1, \tag{17}$$

где $X_w = X_i \cup X_j \cap (\bar{Y}_i) \cup \{r\}$, $Y_w = Y_i \cup Y_j$, r – параметр, определяющий число итераций цикла.

Определение 11. Операцию

$$w: X_w \to Y_w = \left(o_i - o_j\right)^{r,l}\!\leftrightarrow = Y_w \tag{18}$$

будем называть составной операцией выполнения цикла типа do-while-повторения базовых операций o_i и o_j при удовлетворении условиям определения базовых операций и следующим дополнительным условиям:

$$X_i \cap Y_j \neq \emptyset, \;\; Y_i \cap X_j \neq \emptyset, \;\; r \geq 1, \;\; o_i \in O_c, \;\; o_j \in O_l, \;\; \{l\} \subset Y_j, \tag{19}$$

где $X_w = X_i \cup X_j \cap (\bar{Y}_i) \cup \{r\}$, $Y_w = Y_i \cup Y_j$, r – параметр, определяющий ограничение на допустимое число итераций цикла, $l \in \{0,1\}$ – параметр

логического типа, определяющий условие продолжения ($l = 1$) или завершения ($l = 0$) выполнения итераций цикла.

Определение 12. Операцию

$$w: X_w \to Y_w = (o_i) \overset{r}{\leftrightarrow} = Y_w \tag{20}$$

будем называть составной операцией выполнения параллельного цикла повторения выполнения базовой операции o_i при удовлетворении условиям определения базовых операций и следующим дополнительным условиям:

$$\exists x_k = \left(x_{k_1}, x_{k_2}, \dots, x_{k_r}\right) \in X_i: x_{k_1} \neq x_{k_2} \neq \dots \neq x_{k_r}, \tag{21}$$

где $X_w = X_i \cup \{r\}$, $Y_w = Y_i = \{y_q\}$, $y_q = \left(y_{q_1}, y_{q_2}, \dots, y_{q_r}\right)$, $r \geq 1$ – параметр, определяющий число итераций цикла.

Определение 13. Операцию

$$w: X_w \to Y_w = \left(o_i | o_j\right) \overset{l}{\to} = Y_w \tag{22}$$

будем называть составной операцией ветвления выполнения базовых операций o_i и o_j при удовлетворении условиям определения базовых операций, где

$$X_w = X_i \cup X_j \cup \{l\},$$

$$Y_w = \begin{cases} Y_i \cap Y_j, & \text{при } \mathrm{p}_1 \text{ или } \mathrm{p}_3, \\ Y_i, & \text{при } \mathrm{p}_2, \mathrm{p}_4 \text{ или } \mathrm{p}_5 \ (l = 1), \\ Y_j, & \text{при } \mathrm{p}_2, \mathrm{p}_4 \text{ или } \mathrm{p}_5 \ (l = 0), \end{cases}$$

где $l \in \{0,1\}$ – параметр логического типа, определяющий условие выполнения операции o_i ($l = 1$) или выполнения операции o_j ($l = 0$).

Определение 14. Операцию

$$w: X_w \to Y_w = \left(o_{i_r} || \right) \xrightarrow{r=\overline{1,m}} = Y_w \tag{23}$$

будем называть составной операцией явного многометодного анализа с помощью базовых операций o_{i_r} при удовлетворении условиям определения базовых операций и следующим дополнительным условиям:

$$X_{i_1} = X_{i_2} = \dots = X_{i_m},$$

$$x_{j_1} \neq x_{j_2} \neq \dots \neq x_{j_m},$$

где $X_w = \bigcup_{l=1}^{m} X_{i_l}$, $Y_w = \bigcup_{l=1}^{m} Y_{i_l}$, $X_{i_r} = \left\{x_1, \dots, x_{j-1}, x_{j_r}, x_{j+1}, \dots, x_n\right\}$, x_{j_r} – r-й вариант значения параметра x_j, $Y_{i_r} = \{y_{k_r}\}$, y_{k_r} – r-й вариант значения y_k, $r \in \overline{1,m}$, m – число базовых операций, n – число параметров в X_{i_r}.

Определение 15. Операцию

$$w: X_w \to Y_w = \left(o_{i_r} || \right) \xrightarrow{r=\overline{1,m}} = Y_w \tag{24}$$

будем называть составной операцией неявного многометодного анализа с помощью базовой операции o_i при удовлетворении условиям определения базовых операций и следующему дополнительному условию:

$$\exists x_j(r) \in X_i, \tag{25}$$

где $X_w = X_i \cup \{r\}, Y_w = Y_i, X_i = \left\{x_1, \dots, x_{j-1}, x_{j_r}, x_{j+1}, \dots, x_n\right\}$, x_{j_r} – вариант настроек управляющего параметра x_j для выбора r-го метода, реализуемого r-м экземпляром базовой операцией o_{i_r}, $Y_{i_r} = \{y_{k_r}\}$, y_{k_r} – r-й вариант значения y_k, полученного с помощью r-го метода, $r \in \overline{1,m}$, m – число методов, n – число параметров в X_i.

Модуль представляет собой выделенную часть ПО. В рамках диссертации в качестве модулей рассматриваются исполняемые модули (скомпилированные программы на языках программирования общего назначения) и различного рода скрипты. Конструктор вычислительной модели поддерживает возможность создания новых операций в O на основе НРП, а также генерацию программ на языках программирования общего назначения (в текущей версии ИК на языке Python) для рабочих процессов с последующим включением сгенерированных программ в множество M. Модули и НРП могут быть представлены сервисами. В вычислительной модели установлено строгое соответствие сообщений сервисов параметрам модулей и НРП. Для выполнения модулей и НРП на ресурсах ПОВС из N, соединяемых телекоммуникационными каналами из Cm, под управлением метапланировщиков и СУПЗ, представленных множествами Sch и Lrm соответственно, формируются вычислительные задания из J. Пользователи из Us выполняют НРП приложений в соответствии с административными политиками из Ap, применяемых к ресурсам из N, и квотами на использование ресурсов из Q.

Вычислительная история выполнения НРП на узлах из N отражена элементами множества H (лог-файлами). Выбор и агрегирование информации из лог-файлов осуществляется с помощью специальных системных операций. НРП выполняются в контейнеризированной среде. В процессе подготовки и проведения вычислительных экспериментов создается множество Cn конфигураций контейнеризированной среды в соответствии с заданными сценариями из Sc. Вычислительная модель формируется на XML-подобном языке. Ее корректность проверяется XML-схемой. Формальная спецификация вычислительной модели с помощью расширенной формы Бэкуса-Наура представлена в Приложении Г.

2.2. Алгоритмы планирования вычислений и информационного планирования

В диссертации используется подход к управлению вычислениями на уровне приложений [124]. В отличие от подходов к управлению уровнях ГРВС и СУПЗ, суть данного подхода заключается в рациональном выборе ресурсов, наиболее подходящих для эффективного выполнения конкретного приложения. Выбор ресурсов выполняется на основе оценок времени, стоимости и надежности выполнения модулей НРП. Организация вычислительного процесса характеризуется следующими особенностями:

- схема решения задачи представляется в виде НРП, выполняемого в динамически развертываемой ПОВС на выделенных ресурсах;
- требования по выполнению операций НРП задаются в заданиях метапланировщику и СУПЗ, используемых на ресурсах среды;
- метапланировщик Condor DAGMan распределяет задания между ресурсами;
- СУПЗ HTCondor [125] управляет заданиями на ресурсах;
- ПО внешних библиотек задействуется при выполнении НРП;
- динамическое развертывание и выполнение прикладной и системной составляющих СОП на выделенных ресурсах требуют автоматизации методов и средств интеграции, тестирования и контейнеризации ПО;

- контейнеры используются для создания изолированной, воспроизводимой и портативной среды для выполнения приложений, включающей все необходимое системное и прикладное ПО;
- в качестве средства управления контейнерами используется Docker;
- автоматизация развертывания и конфигурирования ПОВС на выделенных ресурсах производится с помощью системы Ansible [126];
- процесс выполнения НРП включает фрагменты вычислений в асинхронном и синхронном режимах: базовые вычислительные операции НРП выполняются в асинхронном режиме, а операции обработки параллельных списков (вариантов) данных осуществляются в синхронном режиме;
- не предполагается взаимодействия по данным и управлению между экземплярами операций, обрабатывающих разные варианты данных;
- разрабатываемые приложения ориентированы на классы задач, в которых среднее время обработки любого варианта данных имеет незначительное отклонение от среднего времени обработки всех вариантов данных.

Схема работы диспетчера НРП с метапланировщиком Condor DAGMan и СУПЗ HTCondor представлена на рисунке 2.2.

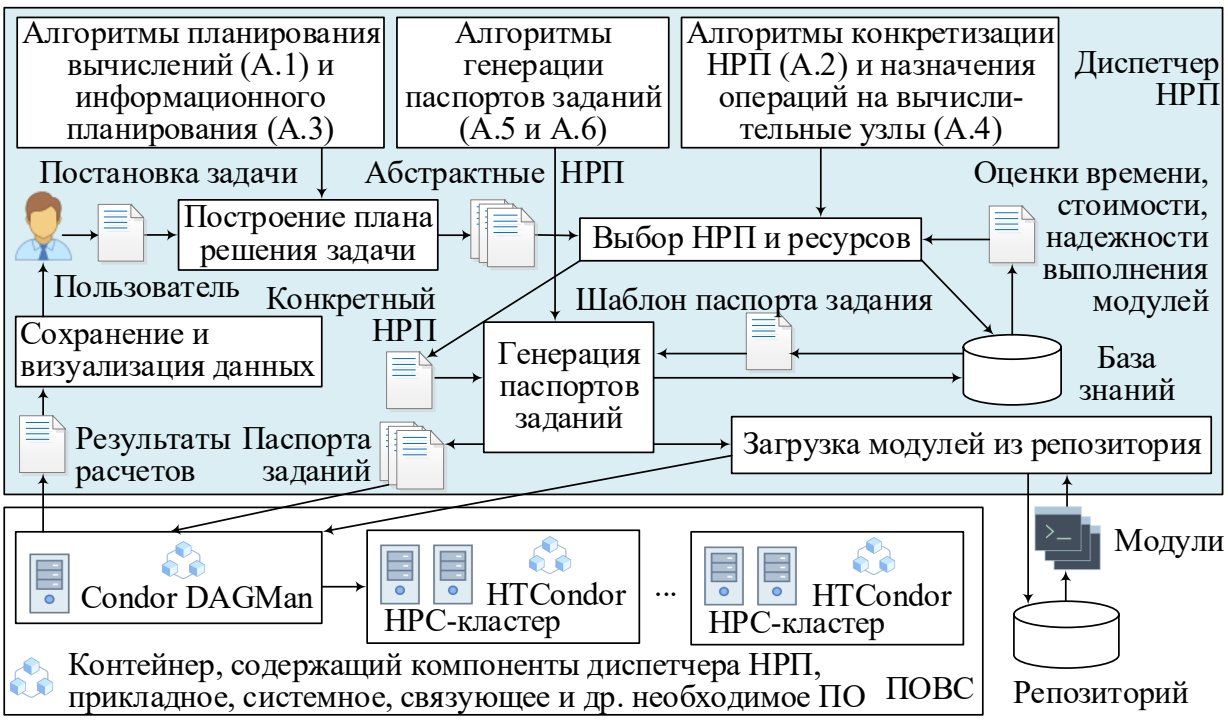


Рисунок 2.2 – Схема взаимодействия диспетчера НРП с Condor DAGMan и HTCondor

Для планирования вычислений по непроцедурной постановке задачи разработан ряд новых алгоритмов. Пусть W – план решения задачи, представленный в виде ярусно-параллельной формы размерности $k \times n$, где k – число ярусов плана, а n – число операций в O. Флажок $b1 = 1$ ($b1 = 0$) – означает, что план построен (план не построен). Флажок $b2 = 1$ ($b2 = 0$) – означает, что на очередной итерации просмотра множества операций O хотя бы одна операция была включена в план (ни одна операция не была включена в план). Множество A и B – вспомогательные множества, в которых хранятся входные и выходные параметры соответственно. Функция $parsingModel$ выполняет разбор текстового файла $model.txt$, в котором содержится модель предметной области в виде набора операций и их параметров, и его преобразование в списки входных и выходных параметров $inputs_1, outputs_1, inputs_2, outputs_2, \dots, inputs_n, outputs_n$ каждой операции из множества O. Структура файла $model.txt$ выглядит следующим образом:

$inputs(; z_{i1}, z_{i2}, \dots, z_{ik})$

$outputs(z_{j1}, z_{i2}, \dots, z_{jl};)$

$o_1(z_{i1}, z_{i2}, \dots, z_{ik}; z_{j1}, z_{i2}, \dots, z_{jl})$

$o_2(z_{i1}, z_{i2}, \dots, z_{ik}; z_{j1}, z_{i2}, \dots, z_{jl})$

...

$o_n(z_{i1}, z_{i2}, \dots, z_{ik}; z_{j1}, z_{i2}, \dots, z_{jl})$

Основные этапы алгоритма А.1, реализующего процесс планирования вычислений (построения частично-упорядоченной последовательности выполнения операций) по непроцедурной постановке задачи с помощью метода прямой волны [127], приведены ниже.

Алгоритм А.1. Планирование вычислений

Input: $model.txt$
Output: W

```
1   begin
2     parsingModel(model.txt →
      O, inputs_1, outputs_1, inputs_2, outputs_2 , … , inputs_n, outputs_n)
3     k = 1;
4     end = False;
```

```
5     for i = 1,n step 1 do
6       W[k,i] = 0;
7       use[i] = 1;
8     end do
9     b1 = False;
10    b2 = False;
11    A = ∅;
12    B = ∅;
13    while end = False do
14      for i = 1,n step 1 do
15        if use ≠ 0 then
16          if inputs_i = ∅ ∨ inputs_i ⊆ A then
17            W [k,i] = 1;
18            b2 = True;
19            if i = 1 then
20              b1 = True;
21              end = True;
22            end if
23          else
24            B = B∪outputs_i
25            use[i] = 0;
26          end if
27        end if
28      end do
29      if b2 = False then
30        end = True;
31      else if end = False then
32          A = A∪B;
33          B = ∅;
34          k = k + 1;
35          for i = 1,n step 1 do
36            W[k,i] = 0;
37          b2 = False;
38          end = False;
39      end if
40      if end = True then
41        end = False;
42        W[k,i] = 0;
43      end if
44    end do
45  end
```

В общем случае в результате выполнения алгоритма А.1 строится НРП, включающий множество сценариев решения задачи. В связи с этим разработан алгоритм А.2 конкретизации НРП путем удаления операций, выполняющих

избыточные вычисления. Суть алгоритма заключается в определении параметров НРП, значения которых вычисляют две или более операции. Для каждого такого параметра операция с наивысшим приоритетом относительно выбранной конфигурации ПОВС (см. раздел 2.1), вычисляющая значение этого параметра, остается в НРП. Остальные операции, вычисляющие то же самое значение, удаляются.

Пусть S – вектор размерности k, $saved$ – множество номеров операций, которые остаются на ярусе. Функция $parsingPriority$ выполняет разбор текстового файла $priority.txt$, содержащего список приоритетов операции относительно выбранной конфигурации ПОВС, и его преобразование в список приоритетов $p_1, p_2, \dots, p_n$ каждой операции из множества O. Структура файла $priority.txt$ выглядит следующим образом:

$o_1 = p_1$

$o_2 = p_2$

...

$o_n = p_n$

На каждой итерации прохождения яруса плана в позицию w вектора S добавляется множество индексов операций, которые остаются на этом ярусе плана. Алгоритм А.2 является модификацией метода обратной волны [127]. Основные этапы алгоритма приведены ниже.

Алгоритм А.2. Конкретизация НРП

Input:
$W, k, O, inputs_1, outputs_1, inputs_2, outputs_2, \dots, inputs_n, outputs_n, priority.txt$
Output: W, O

1 **begin**
2 $parsingPriority(priority.txt \to p_1, p_2, \dots, p_n)$
3 **for** $i \neq 2, i = \overline{1,n}$ **step** 1 **do**
4 $W[k,i] = 0$
5 **end do**
6 $A = \varnothing$;
7 $B = inputs_2$;
8 **for** $i = \overline{1,k}$ **step** 1 **do**
9 $S[k] = \varnothing$
10 **end do**
11 **for** $w = \overline{j \neq k, i}$ **step** -1 **do**

```
12        saved = ∅
13        for i = 1,n step 1 do
14          if W[w,i] == 1 then
15            if outputs_i ∩ B = ∅ then
16              W[w,i] = 0
17            else
18              B = B \ outputs_i
19              A = A ∪ inputs_i
20              max = i
21              for r = 1,n step 1 do
22                if max ≠ r then
23                  if outputs_r ∩ outputs_max ∧ p_r > p_max then
24                    max = r
25                  end if
26                end if
27              end do
28              saved = saved ∪ max
29            end if
30          end if
31        end do
32        B = B ∪ A
33        A = ∅
34        S[k] = saved
35      end do
36      for w = k − 1,1 step − 1 do
37          saved = S[k]
38          for i = 1,n step 1 do
39            if i ⊆ saved then
40              W[w,i] = 1
41            else
42              W[w,i] = 0
43            end if
44          end do
45      end do
46    end
```

Для планирования условий решения задачи (определения множеств исходных и целевых параметров) по процедурной постановке задачи разработан алгоритм А.3. Основные этапы алгоритма А.3, реализующего процесс информационного планирования, приведены ниже.

Алгоритм А.3. Информационное планирование

Input: W

Output: $inputs_1, outputs_1, inputs_2, outputs_2, \dots, inputs_n, outputs_n$

```
1    begin
```

```
2     A* = ∅
3     B* = ∅
4     A** = ∅
5     B** = ∅
6     for i = 1,n step 1 do
7       if W[2,i] == 1 then
8         A* = A* ∪ inputs_i
9         B* = B* ∪ outputs_i
10      end if
11    end do
12    for i = 3,k − 1 step − 1 do
13      for j = 1,n step 1 do
14        if W[i,j] = 1 then
15          A** = A** ∪ inputs_i
16          B** = B** ∪ outputs_i
17        end if
18        A* = A** ∩ ¬B*
19        B* = B** ∪ B*
20      end do
21    end do
22    outputs_1 = A*
23    inputs_2 = B*
24  end
```

2.3. Алгоритм назначения операций на вычислительные ресурсы

В работе [24] предложен алгоритм назначения операций на вычислительные узлы. Как правило, НРП имеют сложную структуру, состоящую из большого числа операций, экземпляры которых могут выполняться последовательно и параллельно. В СУПЗ HTCondor передача данных между вычислительными узлами производится через центральный узел. При этом число пересылок данных равняется $n * 2m$, где n – число операций НРП, m – число вычислительных узлов. При больших объемах пересылаемых данных это может приводить к существенному увеличению времени выполнения НРП и вычислительного эксперимента в целом. Для сокращения времени выполнения НРП предлагается пересылать данные между вычислительными узлами напрямую, минуя центральный узел.

При запуске НРП на вычислительных узлах создаются временные папки, в которые перемещаются данные и исполняемые модули. Задание выполняется от

имени пользователя nobody. Данный пользователь не имеет разрешений на выполнение модулей вне созданной временной папки и команд на перемещение данных между вычислительными узлами. Организация прямого перемещения данных между вычислительными узлами и выполнения на них модулей включает следующие этапы:

– задание пользователя для каждого слота HTCondor (слот представляет собой единицу представления ресурсов вычислительного узла) с разрешением на чтение/запись в папки вне временной папки, создаваемой в ОС при выполнении задания;
– планирование очередности выполнения модулей для обеспечения минимизации времени выполнения НРП с учетом перемещения данных между операциями;
– генерация скриптов-оболочек, в которых прописываются команды для выполнения модулей и передачи данных между вычислительными узлами в соответствии с НРП;
– формирование паспортов заданий, в которых в качестве исполняемых модулей указываются скрипты-оболочки и адреса вычислительных узлов.

На рисунке 2.3 показана схема работы механизма передачи данных через центральный узел. Можно заметить, что при запуске и завершении выполнения каждого прикладного модуля НРП происходит передача входных и выходных данных между вычислительными узлами и центральным узлом. На рисунке 2.3 изображены схемы работы механизма перемещения данных через центральный узел (рисунок 2.3а) и передачи данных между узлами (рисунок 2.3б).

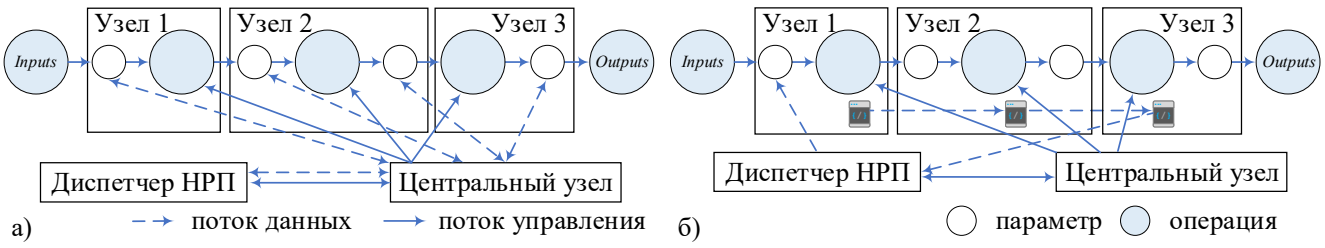


Рисунок 2.3 – Передача данных: через центральный узел (а) и прямая (б)

Для назначения операций на вычислительные узлы сформулируем

постановку задачи. Пусть имеются m узлов ПОВС (включая центральный узел), на которых должны быть выполнены n операций НРП, где b – индекс основного узла. Введем следующие обозначения:

- X – булева матрица размерности $m \times n$ (элемент $x_{li} = 1$ показывает, что i-я операция выполняется на l-м узле);
- τ_{ir} (τ_{jl}) – оценка времени, прошедшего с начала выполнения НРП до завершения операции o_i (o_j) на r-м (l-м) узле;
- q_{il} – оценка времени нахождения модуля, реализующего операцию o_i в очереди l-го узла;
- c_{il} – оценка временных затрат на контейнеризацию ПСПО на l-м узле для выполнения i-й операции;
- t_{il} – время выполнения i-й операции на l-м узле с учетом чтения и записи данных;
- P – булева матрица предшествования операций размерности $n \times n$ (элемент матрицы p_{ij} = 1 означает, что j-я операция предшествует i-й операции);
- D – матрица оценки размеров передаваемых данных размерности $n \times n$ (элемент матрицы $d_{ij} > 0$ показывает размер данных, передаваемых между i-й и j-й операциями);
- W – матрица пропускной способности интерконнекта размерности $m \times m$ (элемент матрицы $w_{lr} \geq 0$ демонстрирует пропускную способность интерконнекта между l-м и r-м узлами);
- $\omega_{lr}(t)$ – коэффициент снижения пропускной способности в момент времени t между l-м и r-м узлами;
- $m \leq \delta$ – число выделенных узлов;
- δ – квота на число узлов;
- n – число операций;
- b – индекс центрального узла;
- λ_l – квота на время использования l-го узла, $\tau_{il} \leq \lambda_l$.

Матрица X должна удовлетворять следующим условиям (26):

$$\bigvee_{i=1}^{n}\bigvee_{l=1}^{m-1}\bigvee_{k=l+1}^{m}(x_{li}\wedge x_{ki})=0,\ \ \bigvee_{i=1}^{n}\bigwedge_{l=1}^{m}\bar{x}_{li}=0. \tag{26}$$

В случае выполнения НРП средствами СУПЗ HTCondor, где выбор и назначение узлов для запуска операций осуществляется главным менеджером этой системы, а обмен данными между узлами осуществляется через центральный узел, оценка $T(X)$ времени выполнения НРП в асинхронном режиме по готовности данных определяется по формулам (27)-(28):

$$T(X)=\max_{i=\overline{1,n},\ l\in\overline{1,m}}\ \tau_{il}, \tag{27}$$

$$\tau_{il}=q_{il}+c_{il}+t_{il}+\max_{\forall j\in\overline{1,n}:p_{ij}=1,i\neq j,r\in\overline{1,m}}\left(\tau_{jr}+\frac{d_{ij}}{\omega_{rb}(t)w_{rb}}+\frac{d_{ij}}{\omega_{bl}(t)w_{bl}}\right). \tag{28}$$

Обмен данными между узлами через центральный узел существенно увеличивает общее время выполнения НРП. Поэтому в диспетчере НРП реализован дополнительный механизм передачи данных напрямую с узла выполнения предшествующей операции на узел выполнения последующей операции. В этом случае, если операции выполняются на одном узле, то размер передаваемых данных между ними считается равным нулю. Тогда оценка $Y(X)$ времени выполнения НРП в асинхронном режиме по готовности данных с помощью диспетчера НРП определяется следующим образом:

$$Y(X)=\max_{i=\overline{1,n},\ l\in\overline{1,m}}\ \tau_{il}, \tag{29}$$

$$\tau_{il}=q_{il}+c_{il}+t_{il}+\max_{\forall j\in\overline{1,n}:p_{ij}=1,i\neq j,r\in\overline{1,m}}\left(\tau_{jr}+\frac{s_{lr}}{\omega_{lr}(t)w_{lr}}\right), \tag{30}$$

$$s_{lr}=\begin{cases}d_{ij}, & \text{если } l\neq r,\\ 0 & \text{в противном случае.}\end{cases} \tag{31}$$

В (29)-(31) $Y(X)$ является целевой функцией, которую требуется минимизировать. В общем случае эта задача является NP-трудной задачей, частным случаем которой является построение m-процессорного расписания с условиями предшествования [128], оптимальное решение которой сложно получить за время, приемлемое для управления вычислениями в реальном времени. Поэтому в диссертации предлагается эвристический алгоритм решения этой задачи.

Суть алгоритма А.4 назначения операций на вычислительные узлы состоит в следующем: каждому узлу вычислительной модели назначаются такие операции из очереди операций, чтобы общее время выполнения НРП было минимальным. Для этого выполняется оценка времени выполнения операции на каждом узле с учетом оценок размера передаваемых данных между операциями и скоростью передачи данных между узлами ПОВС. При этом, если на очередном шаге работы алгоритма одному и тому же узлу назначаются две и более операции, то выбирается та операция, которая предшествует большему числу операций НРП. Учет структуры НРП в алгоритме позволяет уплотнить расписание выполнения заданий в сравнении с алгоритмом HTCondor и, тем самым, сократить время вычислений.

При реализации в НРП многовариантных расчетов с помощью экземпляров операции обработки параллельного списка данных выполнение НРП разделяется на фрагменты вычислений в асинхронном и синхронном режимах. К моменту синхронных вычислений все асинхронные расчеты должны быть завершены, и все узлы свободны. Реализация операции включает дезагрегирование данных на k элементов параллельного списка, синхронную обработку элементов списка экземплярами операции и последующее агрегирование результатов расчетов. При назначении узлов в синхронном режиме решается задача (32)-(33) распределения нагрузки в узлах и ее балансировки

$$\max_{l=\overline{1,m}} \frac{\nu_l}{\pi_l} \downarrow min, \tag{32}$$

$$\sum_{l=1}^{m} \nu_l = k, \quad 0 \le \frac{\nu_l}{\pi_l} \le \lambda_l, \tag{33}$$

где ν_l – переменная, представляющая искомое число экземпляров операции для l-го узла, k – общее число экземпляров, π_l – производительность l-го узла.

Функция $GetReadyOperations$ возвращает список операций R, готовых к выполнению из O. Функция $NextOperation$ возвращает следующую операцию для ее назначения на узел. Функция $BalanceOperationsOnNodes$ предназначена для распределения нагрузки в узлах и ее балансировки для многовариантных расчетов. Функция $shiftNodeTime$ сдвигает время на узлах после назначения на них операций. Функция $GetSubsequentOperations$ возвращает таблицу S, в которой

указано число последующих операций для каждой операции из O. Функция $CalculateOperationTimes$ производит расчет времени выполнения каждой операций из R для каждого свободного узла. Функция $GetOperationsOnNodes$ возвращает таблицу операции, назначенной на узел l. Функция $GetSuitableOperation$ возвращает номер операции, которая предшествует большему числу операций НРП. Функция $GetCountExecutedOperation$ возвращает число выполненных операций НРП. Функций $GetKnownParameteres$ возвращается множество известных параметров, которые были назначены на узлы. Функция *remove* удаляет операцию из списка операций.

Алгоритм А.4. Назначение операций на вычислительные узлы

Input: $inputs_1, outputs_1, inputs_2, outputs_2, \dots, inputs_n, outputs_n$, t, P, D, O, n, m, q, W, ω, c
Output: X

```
1   begin
2     time = 0
3     for l = 1,m step 1 do
4       for i = 1,n step 1 do
5         X[l,i] = 0
6       end do
7     end do
8     for l = 1,m step 1 do
9       F[l] = True
10      T[l] = 0
11    end do
12    A = A ∪ inputs_1
13    R = GetReadyOperations(O,A)
14    i = NextOperation(R)
15    if GetOperationType(i) == 'parallel' then
16      T = BalanceOperationsOnNodes(i,t,P,T,D,W,X,q,c,ω)
17      shiftNodeTime(time,T)
18      remove(R,i)
19      remove(O,i)
20    end if
21    time = GetMinTime(F,T)
22    shiftNodeTime(time,T)
23    S = GetSubsequentOperations(O)
24    while length(O) > 0 do
25      for l = 1,m step 1 do
26        for r = 1,m step 1 do
27          τ[l] = CalculateOperationsTime(t,l,r,R,P,D,W,q,c,ω))
28        end do
```

```
    end do
    C = GetOperationsOnNode(τ, F)
    for l = 1, m step 1 do
      if length(C[l]) == 1 then
        i = C[l]
      else if length(C[l]) > 1 then
        i = GetSuitableOperation(C[l], S)
      end if
      T[l] = GetNodeTime(i, t, P, D, W, q, c, ω)
      X[l, i] = 1
      remove(R, i)
      remove(O, i)
      F[l] = False
    end do
    do
      u = GetCountExecutedOperation(X, O)
      if u == n then
        exit
      end if
      time = GetMinTime(F, T)
      shiftNodeTime(time, T)
      for l = 1, m step 1 do
        if T[l] == time then
          F[l] = True
        end if
      end do
      for l = 1, m step 1 do
        for i = 1, n step 1 do
          if X[l, i] == 1 then
            A = A ∪ outputs_i
          end if
        end do
      end do
    while length(R) == 0
    R = GetReadyOperations(O, A)
    i = NextOperation(R)
    if GetOperationType(i) == 'parallel' then
      t = GetMaxTime(F, T)
      T = BalanceOperationsOnNodes (i, t, P, T, D, W, X, q, c, ω)
      shiftNodeTime(time, T)
      remove(R, i)
      remove(O, i)
    end if
  end do
end
```

2.4. Алгоритмы взаимодействия с внешними системами управления прохождением заданий и метапланировщиками

В диссертации предложена следующая модель взаимодействия с внешними СУПЗ и метапланировщиками. Пусть имеется конечное множество $Sch = \{sch_1, sch_2, \dots, sch_k\}$ метапланировщиков и $Lrm = \{lrm_1, lrm_2, \dots lrm_r\}$ СУПЗ. Метапланировщик использует план решения задачи, описанный в виде DAG, который определяет порядок вычислений. Программные модули являются узлами (вершинами) в графе, а ребра (дуги) определяют зависимости между их входами и выходами. Метапланировщик отправляет программные модули СУПЗ в порядке, представленном в DAG, и обрабатывает результаты их выполнения. Обычно DAG описывается в виде текстового файла, который имеет структуру, представленную в листинге 2.1.

JOB j_1 <файл j_1>
JOB j_2 <файл j_2>
…
JOB j_i <файл j_i>
JOB j_{i+1} <файл j_{i+1}>
…
JOB j_{n-1} <файл j_{n-1}>
JOB j_n <файл j_n>
PARENT j_1 CHILD j_2
…
PARENT j_i CHILD j_{i+1}
…
PARENT j_{n-1} CHILD J_n

Листинг 2.1 – Описание DAG

В данной структуре «JOB ... <…> …» – это команда, определяющая уникальное имя узла (задания) в файле описания DAG и связанный с ним файл паспорта задания, «PARENT … CHILD …» – команда, которая определяет зависимости между заданиями в DAG, $j_i -$ наименование i-го задания DAG,

$file_{j_i}$ – файл со паспортом i-го задания. Узлы являются родителями и/или дочерними узлами других узлов в DAG. Родительский узел должен быть успешно завершен, прежде чем любой из его дочерних узлов может быть запущен. Дочерний узел может быть запущен только после того, как все его родительские узлы успешно завершены. Успешное выполнение задания в общем случае зависит от наличия на вычислительном узле конечного множества программных модулей M, конечного множества библиотек $Lib = \{lib_1, lib_2, \dots, lib_l\}$, от которых зависит выполнение программных модулей и конечного множества входных файлов модулей $Inputs = \{input_1, input_2, \dots, input_n\}$.

Задания для СУПЗ описываются в виде текстовых файлов. Структура файла задания, которая используется в СУПЗ HTCondor, представлена в листинге 2.2.

```
executable = <имя исполняемого файла>
arguments = <список аргументов командной строки исполняемого файла>
requirements = TARGET.Machine == <имя узла, на который назначен модуль>
output = <имя файла, в котором сохраняется стандартный вывода задания>
error = <имя файла, в котором сохраняются стандартный вывод ошибок задания>
log = <имя файла, в котором сохраняются ход выполнения рабочего процесса>
queue
```

Листинг 2.2 – Структура файла задания HTCondor

Функция $parsingPlan$ выполняет разбор текстового файла $plan.txt$, в котором содержится список номеров операций $saved$, включенных в план решения задачи, а также список операций и параметров вычислительной модели и его преобразование в списки входных и выходных параметров $inputs_1, outputs_1, inputs_2, outputs_2, \dots, inputs_n, outputs_n$ каждой операции из множества O. Функция $savedag$ сохраняет DAG в текстовый файл в формате, поддерживаемого HTCondor. Основные этапы алгоритма А.5, реализующего генерацию файла с описанием DAG для метапланировщика, и алгоритма А.6, осуществляющего генерацию паспорта задания для СУПЗ, приведены ниже.

Алгоритм А.5. Генерация DAG

Input: $plan.txt$
Output: $dag.txt$

```
1   begin
2     parsingPlan(plan.txt →
      saved, O, inputs_1, outputs_1 , inputs_2, outputs_2 , … , inputs_n, outputs_n)
3     for i = 1,n step 1 do
4       if i ⊆ saved then
5         list = list∪("JOB " + O_i + str(O_i + ".job" ))
6       end if
7     end do
8     for i = 1,n step 1 do
9       parents = ∅
10      childs = ∅
11      for j = 1,n step 1 do
12        if i ≠ j then
13          if i ⊆ saved and j ⊆ saved then
14            if outputs_i ∩ inputs_j ≠ ∅ then
15              parents = parents∪O_i
16              childs = childs∪O_j
17            else if inputs_i ∩ outputs_j ≠ ∅ then
18              parents = parents∪O_j
19              childs = childs∪O_i
20            end if
21          end if
22        end if
23      end do
24      if parents ≠ ∅ and childs ≠ ∅ and parents ∩ childs ≠ ∅ then
25        list = list∪("PARENT" + str(parents) + "CHILD" + str(childs))
26      end if
27    end do
28    savedag(list, "dag.txt")
29  End
```

Алгоритм А.6. Генерация задания

Input: $execute\_file.txt, args.txt, libs.txt, inputs.txt, outputs.txt, nodes.txt$
Output: $file.job$

```
1   begin
2      exe_filename = readfile("execute_file.txt")
3      args = readfile("args.txt")
4      libs = readfile(«libs.txt»)
5      inputs = readfile("inputs.txt")
6      outputs = readfile("outputs.txt")
7      nodes = nodes("nodex.txt")
8      openfile("file.job")
9      writefile("file.job","execute = " + exe_filename)
10     writefile("file.job","arguments = " + args)
11     writefile("file.job","requirements = TARGET.Machine == " +
       nodes[exe_filename])
12     writefile("file.job","output = " + exe_filename + ".output")
13     writefile("file.job","error = " + exe_filename + ".error")
14     writefile("file.job","log = " + +exe_filename + ".log")
15     writefile("file.job","queue")
16     closefile("file.job")
17  end
```

2.5. Выводы

Во второй главе получены следующие основные результаты:

- предложена вычислительная модель СОП, являющаяся развитием известных моделей подобного назначения;
- разработаны оригинальные алгоритмы планирования вычислений, конкретизации НРП, информационного планирования, назначения операций на узлы, распределения нагрузки и генерации заданий;
- разработаны схема планирования вычислений на уровне приложения и оригинальный подход к перемещению данных между вычислительными узлами ПОВС, который позволяет существенно сократить время выполнения НРП по сравнению с перемещением данных через центральный узел.

Результаты исследований по данной главе опубликованы в [10, 15, 24, 28]. В частности, вычислительная модель рассмотрена в работах [10, 15, 28]. Механизмы передачи данных и алгоритм назначения операций на узлы ГРВС представлены в работе [24].

Глава 3. Инструментальный комплекс разработки и применения сервис-ориентированных приложений

В данной главе рассматривается архитектура, основные функциональные возможности и аспекты реализации ИК для разработки и применения СОП. Предлагаются сценарии выполнения НРП. Рассматриваются возможности ИК по работе с РБД в ОП ресурсов ПОВС. Приводится методика интеграции и контейнеризации ПСПО. Представляется подход к разработке и применению испытательных стендов для тестирования СОП и их компонентов. Обсуждаются средства визуализации НРП и расчетных данных.

3.1. Архитектура и функциональные возможности инструментального комплекса

В рамках диссертации разработан новый ИК Framework for Development and Execution of Scientific WorkFlows (FDE-SWFs) [8, 10, 15, 17, 19, 20, 27, 28, 34]. FDE-SWFs относится к классу WMS. Он базируется на подходе к разработке СОП в ПОВС.

FDE-SWFs включает следующие основные подсистемы: пользовательский интерфейс, инструменты разработчика для конструирования и спецификации вычислительных моделей, диспетчер НРП для управления вычислительными процессами и средой их выполнения. Для доступа пользователей к внешним информационно-вычислительным ресурсам и системам, с которыми им необходимо взаимодействовать при подготовке и проведении вычислительных экспериментов, он предоставляет набор специализированных API. База знаний ИК содержит спецификацию вычислительных моделей разрабатываемых приложений, НРП и сведения о вычислительных ресурсах. Исходная информация и результаты выполнения НРП хранятся в расчетных БД. Архитектура FDE-SWFs представлена на рисунке 3.1.

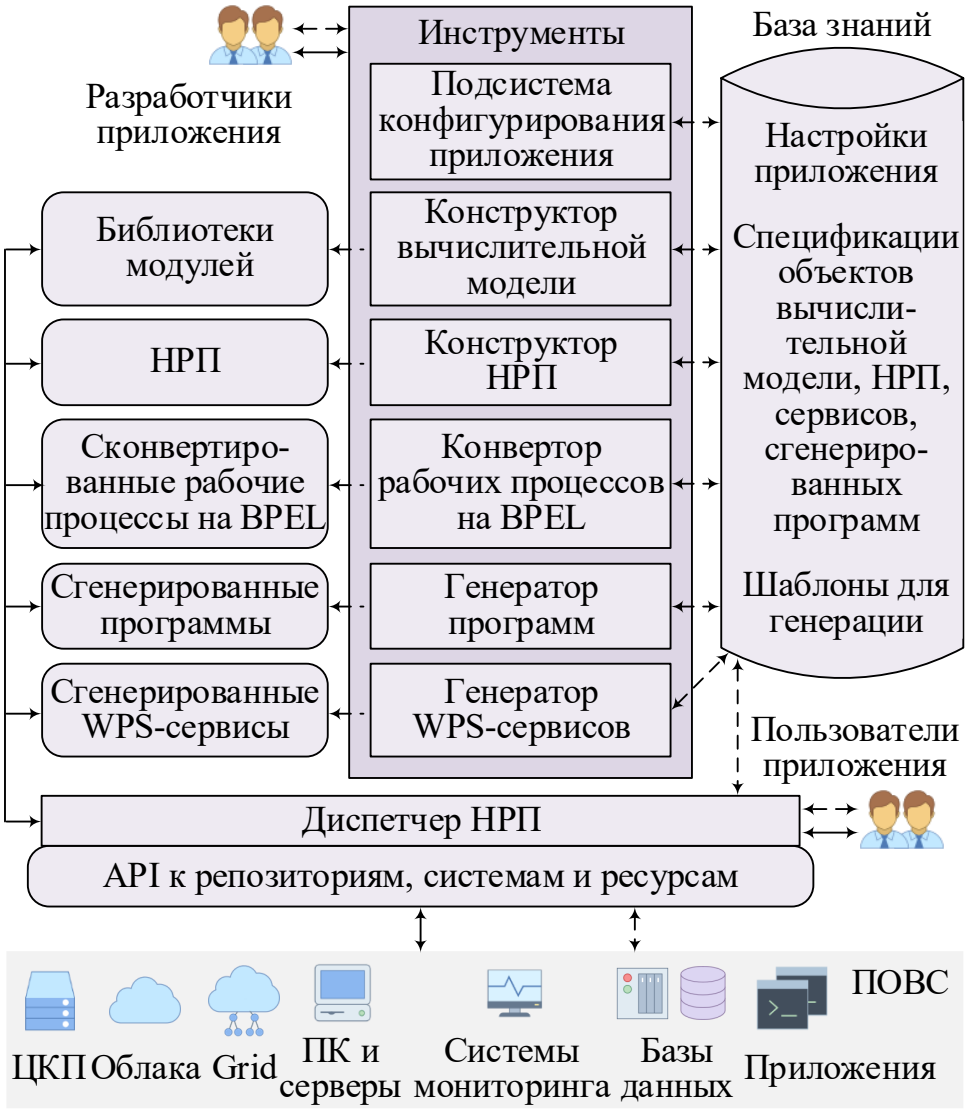


Рисунок 3.1 – Архитектура ИК

Подсистема конфигурирования приложения обеспечивает его разработчику следующие возможности: настройку системных компонентов разрабатываемого приложения, репозиториев ПО и БД; определение структур вычислительной модели, постановок задач и НРП; подключение прикладных и системных модулей приложения; указание требуемых информационных и вычислительных ресурсов и систем.

Конструктор вычислительной модели [8, 10, 15] поддерживает описание предметной области приложения в виде совокупности абстрактных параметров (переменных) и операций (частичных отношений), определяемых над множеством параметров и реализуемых программными модулями и сервисами, а также связей между всеми перечисленными объектами. Модель может быть дополнена

системными объектами, предназначенными для реализации взаимодействия рабочих процессов с информационно-вычислительными ресурсами, которые управляют системами и БД. Системные объекты могут использоваться при конструировании НРП. Как правило, в разработке вычислительной модели принимают участие прикладные программисты – специалисты в предметной области, реализующие модели и алгоритмы решения задач, и системные программисты – специалисты по работе с вычислительными ресурсами среды (рисунок 3.2).

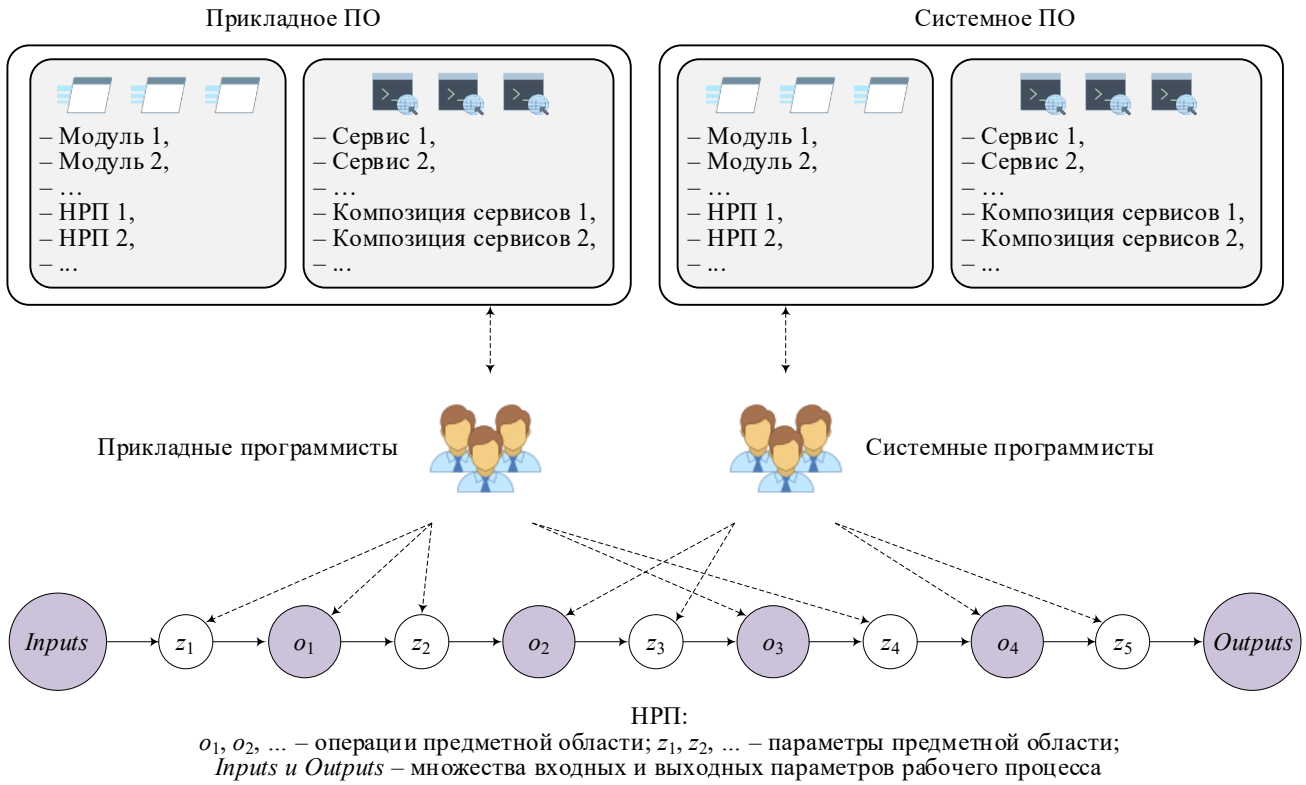


Рисунок 3.2 – Схема разработки вычислительной модели

На вычислительной модели по сформулированным постановкам задач строятся НРП [8, 10, 15]. Отличительной особенностью FDE-SWFs в сравнении с другими WMS является поддержка конструктором НРП формулировки постановок задач как в процедурной, так и непроцедурной формах. В первом случае разработчик конструирует НРП в виде абстрактной программы с вызовами операций, циклами и ветвлениями в графической форме методом drag-and-drop.

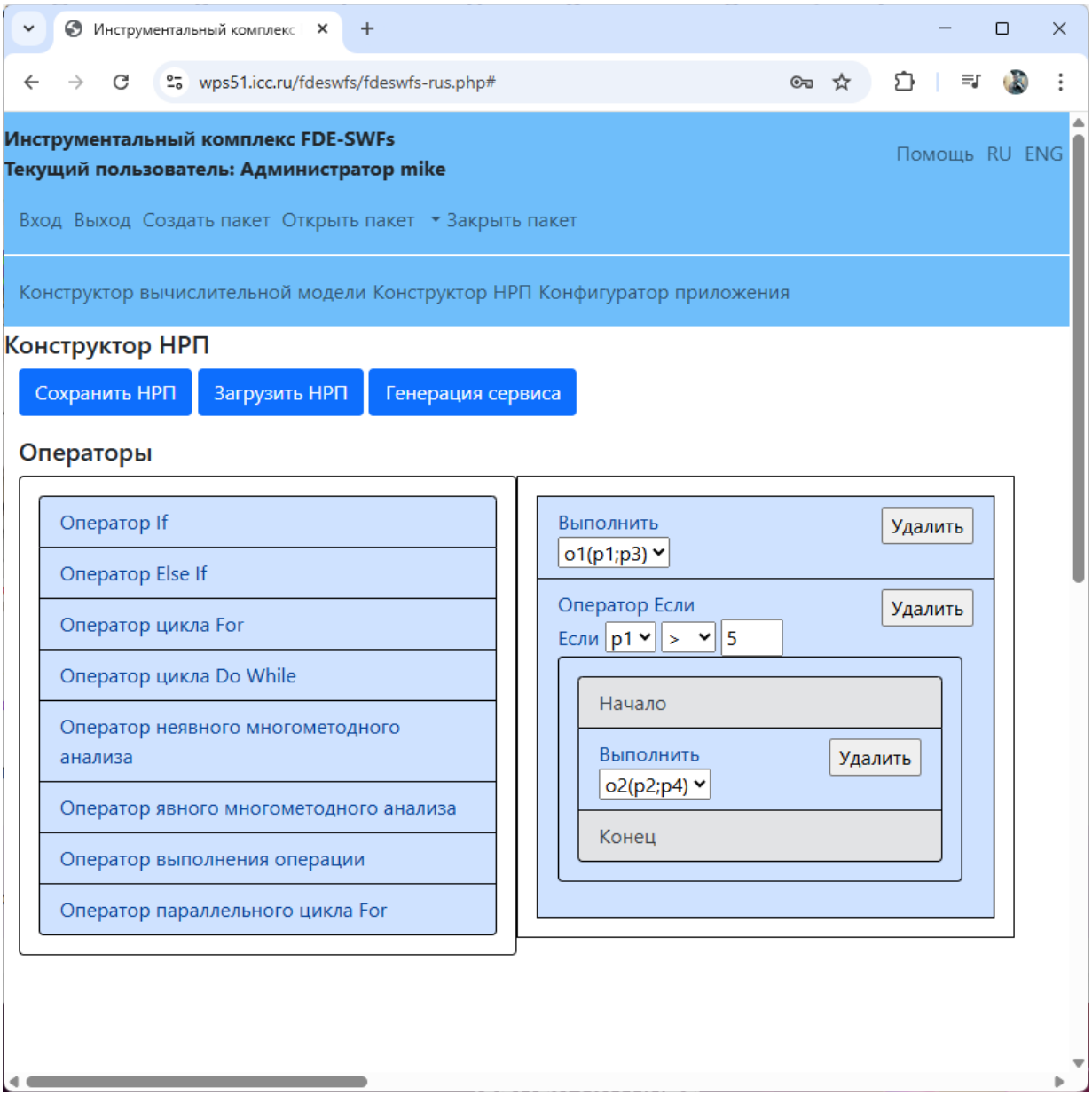


Рисунок 3.3 – Форма для конструирования НРП по процедурной постановке задачи

Во втором случае разработчик задает исходные (входные) и целевые (выходные) параметры НРП. Затем подсистема планирования вычислений конструктора НРП на основе связей между операциями, представленных в вычислительной модели, строит частично-упорядоченную последовательность выполнения операций в виде двудольного ориентированного ациклического графа. Скриншоты форм для конструирования НРП по процедурной и непроцедурной постановке задачи приведены на рисунки 3.3 и 3.4 соответственно.

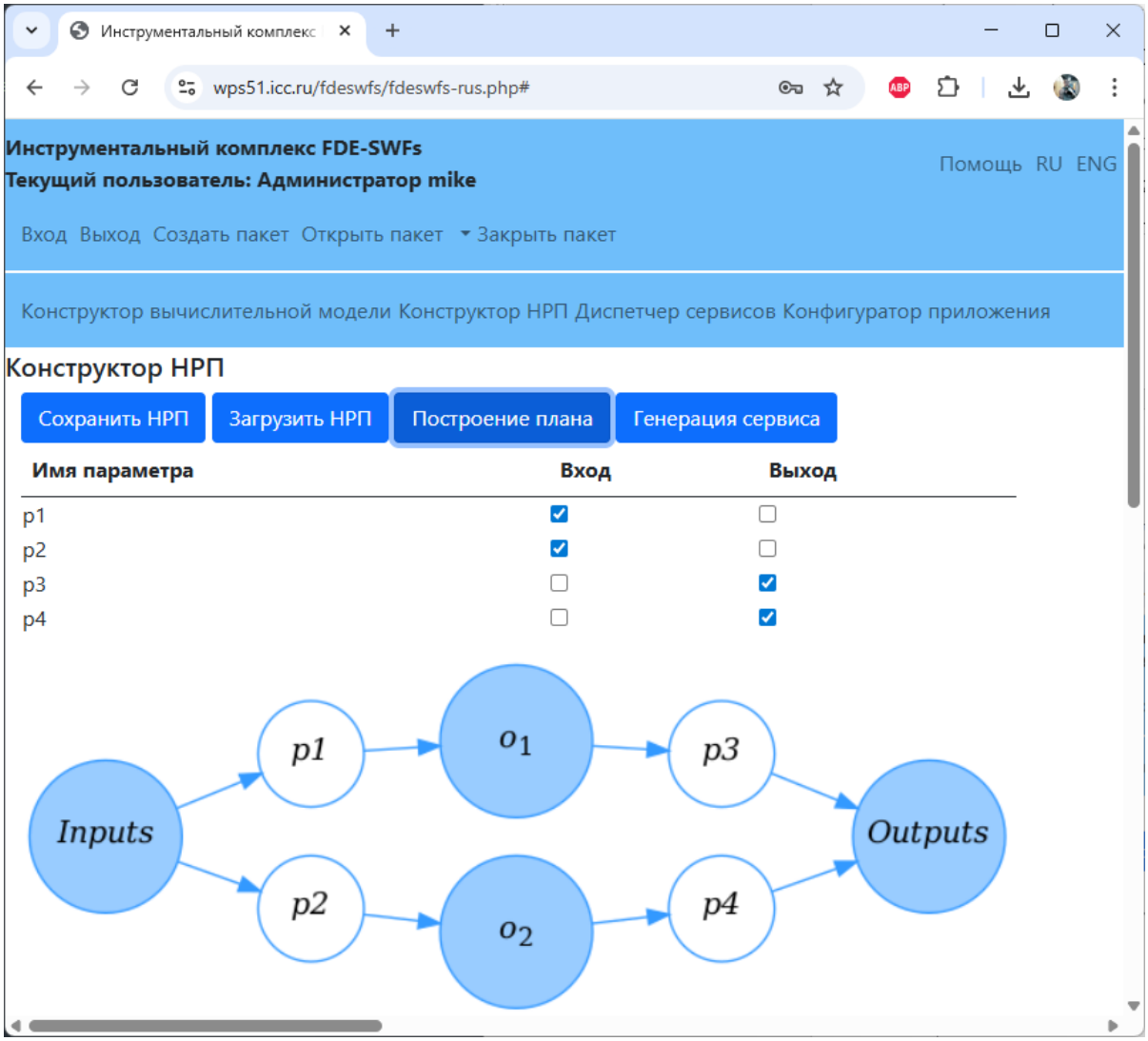


Рисунок 3.4 – Форма для конструирования НРП по непроцедурной постановке задачи

В FDE-SWFs поддерживается конвертирование НРП на язык BPEL для поддержки их выполнимости в других WMS, базирующихся на использовании данного языкового стандарта. Разработан специальный конвертор НРП на язык BPEL. В Приложении Д приведена таблица соответствия конструкций спецификации НРП конструкциям языка BPEL. Корректность кода на языке BPEL проверяется с помощью XML-схемы языка BPEL.

FDE-SWFs поддерживает возможность автоматической генерации программы на языке программирования общего назначения Python для НРП, построенного по процедурной или непроцедурной постановке задачи. Генерация программ для НРП производится с целью обеспечения возможности выполнения этих рабочих процессов автономно от среды FDE-SWFs. Разработан специальный

конвертор, который переводит спецификацию НРП в программу на языке программирования Python. Для синтаксического анализа спецификации НРП используется Python-библиотека parglare [129]. С помощью этой библиотеки конвертор разбирает конструкции спецификации НРП и преобразует их в соответствующие им конструкции языка Python. Конвертор позволяет создавать как программы на основе традиционного НРП с исполняемыми модулями, так и сервис-ориентированного НРП. В Приложении Е приведена таблица соответствия конструкций спецификации НРП инструкциям языка Python.

Генератор WPS-сервисов, предложенный в работе [8], поддерживает автоматическое развертывание модулей и НРП приложений в качестве новых сервисов, а также включение вновь созданных сервисов в вычислительную модель (рисунок 3.5).

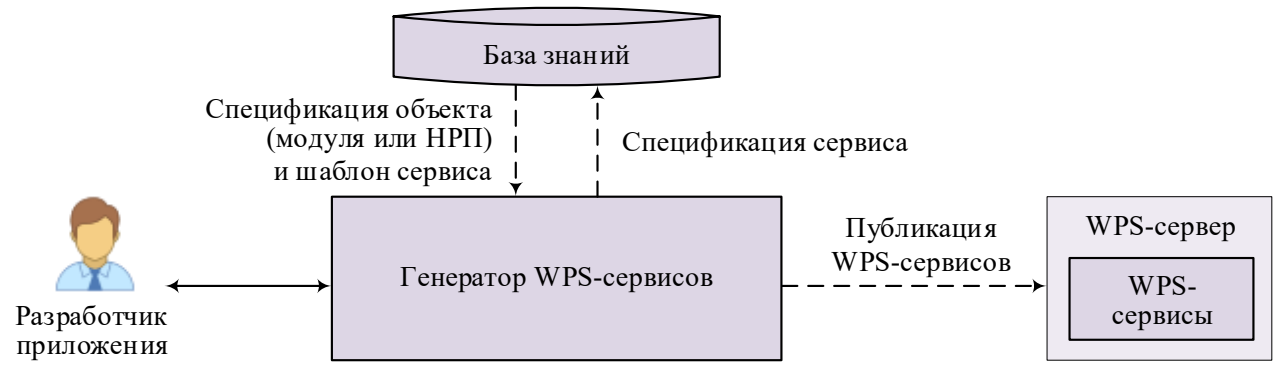


Рисунок 3.5 – Генерация WPS-сервисов и их включение в вычислительную модель

Разработчик пакета с помощью FDE-SWFs может генерировать WPS-сервисы для модулей и НРП и размещать их в каталоге сервисов. Генерация сервисов на основе рабочего процесса осуществляется в несколько этапов. Конструктор НРП извлекает из базы знаний спецификацию нужного рабочего процесса, формирует соответствующий JSON-объект и отправляет его генератору WPS-сервисов. Затем генератор на основе готового шаблона WPS-сервиса и полученной информации от конструктора НРП производит генерацию нового файла WPS-сервиса и сохраняет в каталоге сервисов WPS-сервера. Далее генератор добавляет информацию о новом WPS-сервисе в файл конфигурации WPS-сервера и перезагружает его. Схема генерации WPS-сервиса представлена на рисунке 3.6.

Диспетчер НРП представляет собой исполнительную подсистему FDE-SWFs и отвечает за выполнение рабочих процессов в ПОВС. С его помощью конечный пользователь приложения может запускать НРП и отслеживать процесс вычислений. Диспетчер взаимодействует с метапланировщиками и СУПЗ среды в процессе выделения вычислительных ресурсов для выполнения НРП. В частности, он генерирует паспорта заданий по выполнению модулей НРП для метапланировщиков и СУПЗ. Диспетчер также предоставляет доступ к расчетной БД и обеспечивает возможность загрузки и последующей визуализации результатов ранее проведенных экспериментов. На основе этой информации можно подготовить исходные данные для новых исследований. По каждому эксперименту в расчетной БД хранятся постановка задачи, НРП, исходные, выходные и необходимые промежуточные данные, а также системная информация о ходе выполнения вычислительного процесса в виде лог-файлов. Управление вычислениями диспетчером НРП рассмотрено в работах [10, 12-13, 15-17, 25, 27-28, 36].

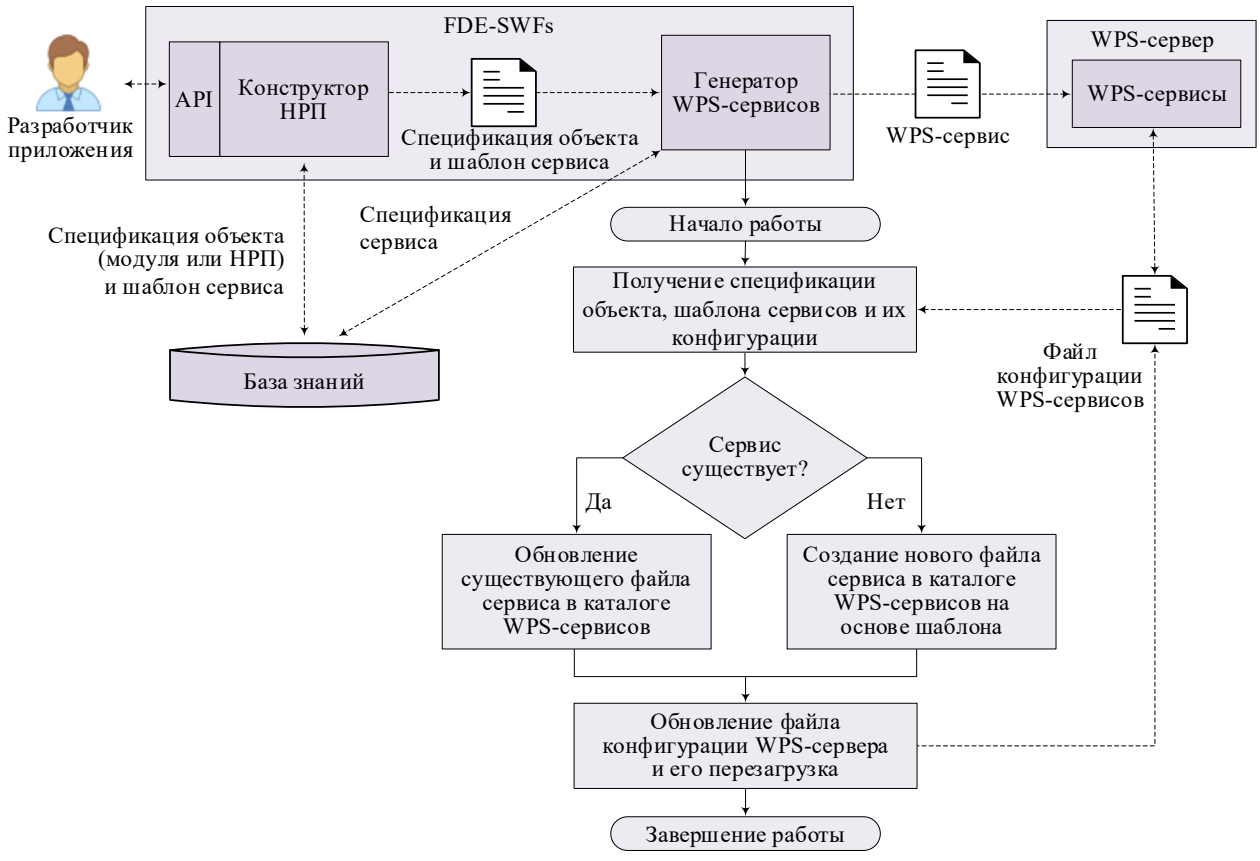


Рисунок 3.6 – Схема генерации WPS-сервиса

В интерпретатор НРП включены операторы выполнения операций, реализуемых сервисами. Эти операторы обеспечивают выполнение любого веб-сервиса на WSDL, реализующего процесс вычислений, получения или обработки данных. В частности, поддерживается работа с WPS-сервисами для сбора, обработки и анализа пространственно-распределенных геоданных [26, 29]. Конструктор вычислительной модели контролирует строгое соответствие параметров операций сообщениям сервисов при определении этих объектов.

Схема функционирования FDE-SWFs в ПОВС представлена на рисунке 3.7.

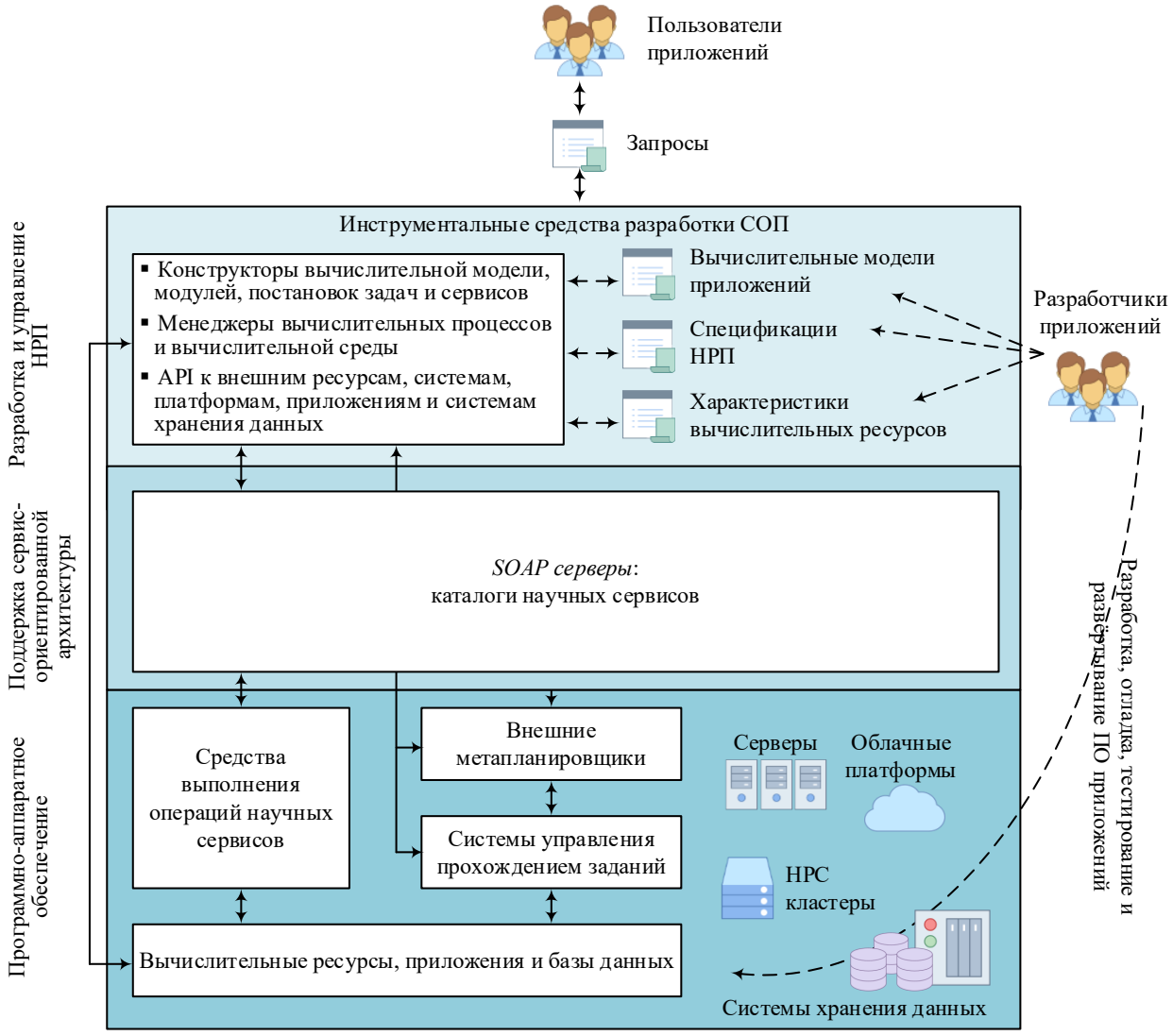


Рисунок 3.7 – Схема функционирования FDE-SWFs в ПОВС

На рисунке 3.8 представлена схема планирования и выполнения НРП на уровне приложений. На схеме представлены две категории пользователей:

– внешние пользователи НРП приложений;

– локальные пользователи приложений;

Внешние пользователи НРП приложений взаимодействуют с WPS-сервером для получения списка готовых НРП приложений, вызова НРП приложения и получения состояния его выполнения.

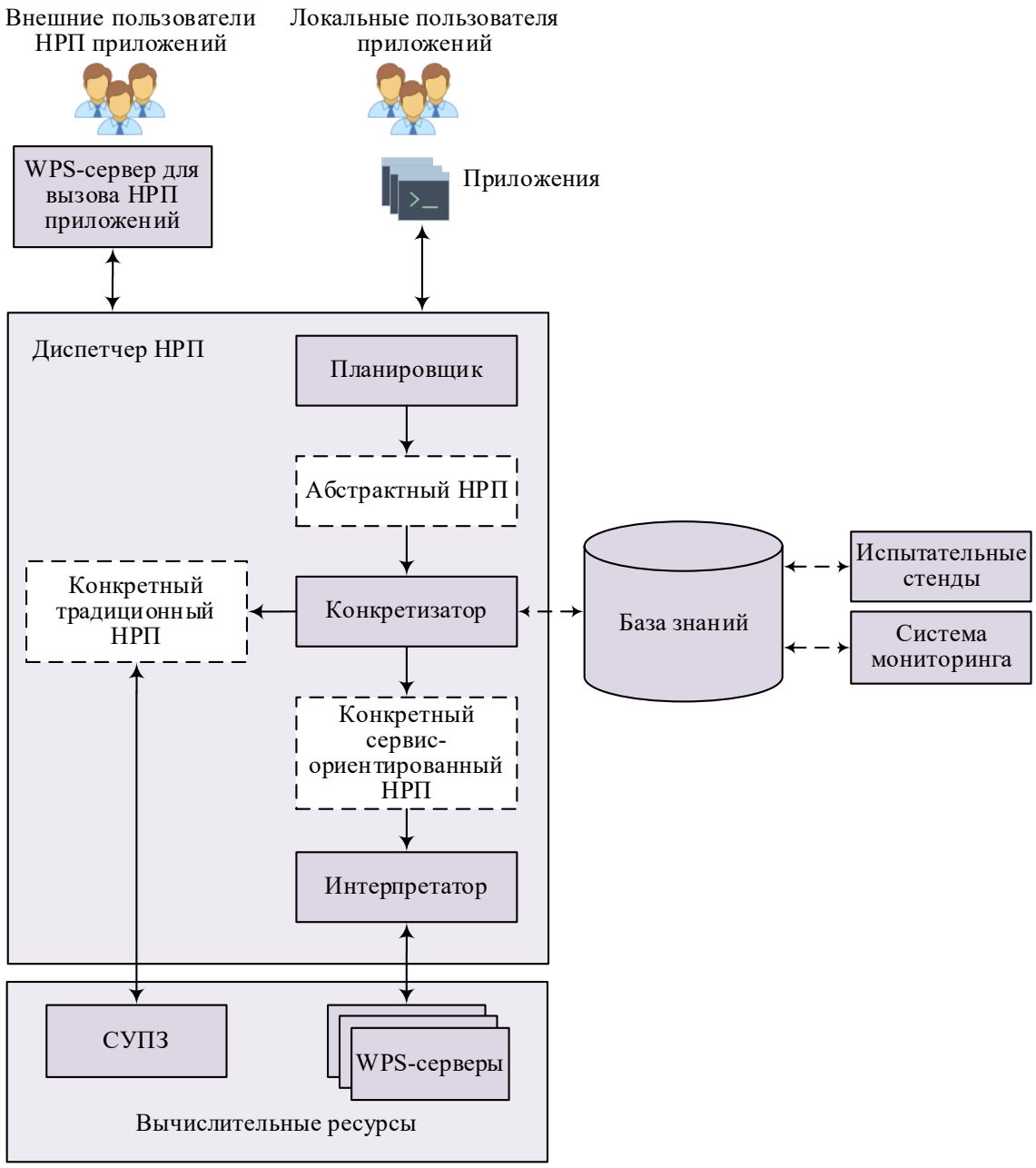


Рисунок 3.8 – Схема планирования и выполнения НРП на уровне приложений

При вызове НРП приложений локальными пользователями производится вызов планировщика, который строит несколько вариантов абстрактного НРП. Конкретизатор выполняет построение конкретного НРП. При его построении решаются следующие задачи:

- удаление операций, выполняющих избыточные вычисления;
- многокритериальный выбор вычислительных ресурсов для выполнения НРП на основе информации из базы знаний о надежности, стоимости и времени выполнения модулей НРП на вычислительных ресурсах;
- построение рационального плана выполнения НРП и назначения модулей на вычислительные узлы.

В зависимости от типа приложения результатом работы конкретизатора может быть сервис-ориентированный конкретный НРП или традиционный конкретный НРП. Сервис-ориентированный конкретный НРП передается интерпретатору, который выполняет построение композиции WPS-сервисов, выполняемых на назначенных WPS-серверах. Традиционный конкретный НРП передается СУПЗ для выполнения на назначенных вычислительных узлах.

3.2. Аспекты реализации инструментального комплекса

ИК имеет клиент-серверную архитектуру и представляет собой веб-приложение, разработанное с использованием современных веб-технологий. Детальная информация об используемом стеке программных средств и технологий приведена в таблице 3.1.

На уровне приложения для разработки прикладного ПО для достижения максимальной производительности и обеспечения параллельных вычислений используются языки программирования общего назначения C/C++. Для описания спецификаций вычислительных моделей, постановок задач и НРП используются XML-подобные языки, поскольку язык XML поддерживает проверку корректности кода с помощью XML-схемы. В расчетной БД хранятся исходные данные и результаты экспериментов в виде файлов. Поскольку расчетные данные могут быть представлены как текстовыми, так и двоичными файлами, хранение их в реляционной БД влечет достаточно большие накладные расходы, связанные с созданием для них метаданных и индексов, так и с преобразованием их в форматы, используемые в реляционной БД.

Таблица 3.1 – Информация об используемом стеке программных средств и технологий

	Компонент / процесс / структура данных	Средство / технология реализации
Уровень приложения	Прикладное ПО приложения	C/C++, Python, JavaScript, dll, exe-файлы, bat-файлы
	Системное ПО приложения	C/C++, Python, JavaScript, dll, exe-файлы, bat-файлы
	Спецификация вычислительной модели	XML+
	Спецификация постановок задач	XML+
	Спецификация НРП	XML+
	Расчетная БД	Файловая БД с метаданными на JSON
	Форматы данных	XML, JSON, txt-файлы, двоичные файлы
Уровень диспетчера НРП	Спецификация веб-сервисов	WSDL, WPS
	Фабрика веб-сервисов	Python
	Поддержка работы веб-сервисов	PHP, Python, BPEL
	Конструктор вычислительной модели	PHP
	Конструктор библиотек модулей	PHP
	Конструктор постановок задач	PHP
	Менеджер ПО	PHP
	Менеджер вычислительных процессов	NodeJS
	Менеджер вычислительной среды	NodeJS
Уровень вычислительной	Связующее ПО	Condor DAGMan
	Система управления рабочими процессами в узлах среды	HTCondor
	Платформа контейнеризации	Docker
	Средство поддержки технологии In-Memory Data Grid	Apache Ignite
	Доставка и развертывание ПО на узлах среды	Ansible

На уровне диспетчера НРП для описания спецификаций веб-сервисов используется два языка: WSDL и WPS. Язык WSDL используется для оркестрации сервисов с помощью НРП, описанных на языке BPEL, который является стандартом и поддерживается рядом WMS. Для построения композиций WPS-сервисов используется язык программирования общего назначения Python. Его выбор для решения этой задачи обусловлен тем, что для него настоящее время разработан достаточно большой набор библиотек для работы с сетью и сервисами, а также для обеспечения асинхронного и параллельного вызова WPS-сервисов. ИК реализован в виде веб-приложения, и поэтому для конструирования вычислительных моделей, постановок задач, НРП и управления ПО используется язык веб-программирования PHP. Его выбор обусловлен поддержкой большинством ОС и БД, высокой производительностью и простотой интеграции с HTML, что упрощает создание динамических веб-приложений. Для управления вычислениями используется платформа NodeJS, что объясняется наличием в ней асинхронной модели обработки запросов.

На уровне вычислительной среды для обеспечения высокопроизводительных вычислений в ПОВС используются метапланировщик Condor DAGMan и СУПЗ HTCondor. Они были выбраны, поскольку являются достаточно популярными программными средствами подобного назначения и выступают в качестве связующего ПО в лидирующих WMS таких как Pegasus и PanDA. Для доставки и развертывания контейнеризированной вычислительной среды на ресурсах используется программное средство Ansible, которое предназначено для поддержки автоматизации конфигурирования ПО.

3.3. Средства работы с распределенными базами данных

В работах [9, 20, 23] рассмотрены средства работы с распределенными базами данных в оперативной памяти вычислительных узлов. В настоящее время распределенные хранилища данных привлекают пристальное внимание научного сообщества. Такие хранилища зачастую обеспечивают повышение надежности и безопасности хранения данных [130], а также ускорение их обработки. В этом

контексте в FDE-SWFs разработаны средства поддержки In-Memory Data Grid (IMDG) – технология работы с БД, распределенными в ОП вычислительных ресурсов [131]. Системы хранения данных на основе IMDG имеют неоспоримое преимущество в скорости обработки данных по сравнению с традиционными БД [132]. Кроме того, имеется большой спектр инструментов для поддержки технологии IMDG на ресурсах высокопроизводительных вычислений [133, 134]. Каждый инструмент представляет собой связующее ПО для поддержки распределенной обработки больших наборов данных. Hazelcast 5.3.2 [135], Infinispan 14.0.7 [136] и Apache Ignite 2.15.0 [137] являются наиболее популярными программными средствами из свободно распространяемого связующего ПО, которое реализует возможности IMDG для научных приложений, основанных на рабочих процессах и потоках заданий, имеющих схожую функциональность и производительность. Они поддерживают клиентские API на Java SE 17, C++ (ISO/IEC 14882:2020) и Python 3.12.0. Конечные пользователи могут использовать их для создания кластеров IMDG с распределенными кэшами ключ/значение. Каждый из них может иметь преимущество в обработке для определенного сценария и набора данных. Платформа Hazelcast – это ПО для динамических операций с данными в режиме реального времени. Она объединяет высокопроизводительную потоковую обработку с быстрым хранением данных [138]. Некоторые компоненты Hazelcast распространяются в соответствии с Лицензионным соглашением сообщества Hazelcast версии 1.0. Infinispan – это инструмент IMDG, обеспечивающий гибкие возможности развертывания и надежные средства хранения, управления и обработки данных [139]. Он поддерживает и распределяет данные типа ключ-значение на масштабируемых кластерах IMDG с высокой доступностью и отказоустойчивостью. Infinispan доступен по лицензии Apache 2.0. Apache Ignite – это полнофункциональная, децентрализованная транзакционная БД типа ключ-значение с удобным и простым в использовании интерфейсом для работы с большими объемами данных в режиме реального времени, включая асинхронные вычисления в ОП [140]. Она поддерживает архитектуру долговременной памяти с ускорителем больших

данных в памяти и на диске для данных, вычислений, сервисов и потоковых сетей.

Apache Ignite – это продукт с открытым исходным кодом, распространяемый под лицензией Apache License 2.0. В отличие от Hazelcast, полная функциональность Apache Ignite и Infinispan бесплатна. В рамках SQL-сети Apache Ignite предоставляет в распоряжение операторов горизонтально масштабируемую и отказоустойчивую РБД SQL, которая полностью соответствует стандарту ANSI-99. Hazelcast и Infinispan поддерживают аналогичные SQL-запросы с исключениями.

В рамках диссертации развертывание связующего ПО IMDG на вычислительных узлах должно осуществляться динамически во время выполнения рабочего процесса. Для Hazelcast и Infinispan это ключевое требование является трудоемким процессом. Как правило, конечные пользователи Hazelcast и Infinispan сначала развертывают кластер и только потом запускают задачи обработки данных на узлах развернутого кластера. Это было отмечено в различных исследованиях энергосистем на базе IMDG [141-143]. Разработчиками Apache Ignite предложена методика определения числа узлов кластера IMDG на основе оценки размера ОП, необходимого для обработки данных [144]. Аналогичная методика была разработана и для Infinispan. Тем не менее, Apache Ignite дополнительно учитывает накладные расходы памяти при хранении данных и использовании диска. Hazelcast предоставляет только примеры, которые можно экстраполировать для конкретных рабочих нагрузок. В работах [141-143,145] рассмотрены преимущества IMDG в исследованиях энергетических систем. Сравнительный анализ средств поддержки IMDG, приведенный таблице 3.2, также показывает преимущества Apache Ignite в скорости обработки данных, поддержке стандарта SQL ANSI-99 для пользователей-экспертов и большого числа языков программирования для прикладных разработчиков. Как правило, преимущества Apache Ignite в скорости обработки данных достигаются для данных большого размера. В частности, в таблице 3.2 его превосходство показано относительно чтения полного набора данных размером более 1 ГБ. В отдельных экспериментах для маленьких и средних размеров данных менее 1 ГБ преимущество могут демонстрировать как Hazelcast,

так и Infinispan.

Таблица 3.2 – Сравнительный анализ средств поддержки IMDG

Критерий	Hazelcast	Infinispan	Apache Ignite
Лицензия	Open Source (Apache 2.0), коммерческая версия	Open Source (Apache 2.0), коммерческая поддержка	Open Source (Apache 2.0), коммерческая поддержка
Поддержка SQL	Да (Hazelcast SQL, Jet)	Ограниченная (через Hibernate, Lucene)	Полноценная (ANSI-99 SQL, JDBC/ODBC)
Скорость считывания полного набора данных*	~2763 МБ/с	~3792 МБ/с	~4863 МБ/с
Поддержка языков программирования	Java, Python, Go, Node.js, .NET, C++	Java, Python (Thrift), CLI (CLJS)	Java, .NET (C#), C++, Python, Node.js, PHP
API	Java (родной), REST, Memcached, SQL	Java (родной), REST, Memcached	Java (родной), SQL, JDBC/ODBC, REST

* В соответствии с оценками, полученными в работах [132, 146] для большого размера данных свыше 1 ГБ на узлах, сопоставимых по своим вычислительным характеристикам.

Разработаны методика, алгоритмы и программные средства автоматизации динамического развертывания кластера Apache Ignite, поддерживающего IMDG (рисунок 3.9). Учет детальной структуры данных, предназначенных для хранения в РБД, позволяет существенно сократить погрешность оценки требуемого размера ОП в сравнении с известными методиками и рационально выделять нужное число узлов вычислительных кластеров.

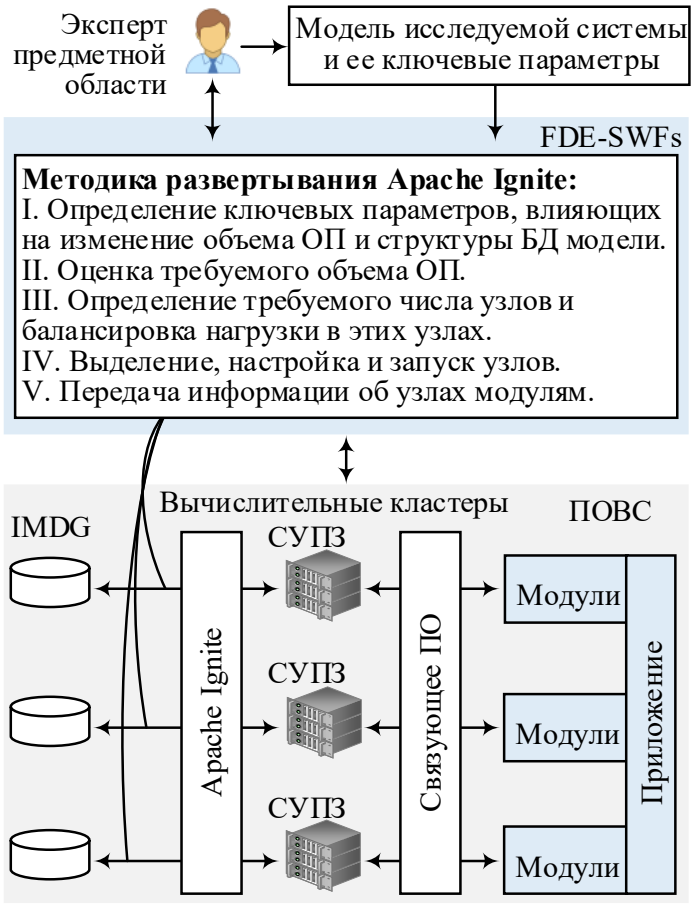


Рисунок 3.9 – Схема работы FDE-SWFs с IMDG

Для оценки размера ОП S при решении задач с использованием кластера Apache Ignite разработана методика, базирующаяся на расчетах по формулам (34)-(36):

$$S = D + (o_{data} + o_{index}) \sum_{i=1}^{t} r_i(x_i, x_2, \ldots, x_h), \quad (34)$$

$$D = \sum_{i=1}^{t} d_i, \quad (35)$$

$$d_i = r_i(x_1, x_2, \ldots, x_h) \sum_{j=1}^{c_i} s_{ij}, \quad (36)$$

где переменные интерпретируются следующим образом:

- D (D^*) – суммарный размер данных в байтах БД модели (тестовой модели) ЭК;
- $t > 0$ – число таблиц;
- $d_i > 0$ – размер данных в i-й таблице в байтах, рассчитываемый на основе информации о структуре данных;
- $o_{data} > 0$ – удельные накладные расходы на хранение данных, рассчитываемые для числа записей не меньше 16000000;

- $o_{index} > 0$ – удельные накладные расходы на хранение индексов, рассчитываемые для числа записей не меньше 16000000;
- $c_i > 0$ – число столбцов в i-й таблицы;
- $r_i(x_1, x_2, \dots, x_h) > 0$ – число строк в i-й таблице (функция r_i конкретизируется для каждой i-й таблицы с учетом ключевых параметров предметной области);
- $x_1, x_2, \dots, x_h$ – ключевые параметры предметной области, влияющие на изменение размера обрабатываемых данных;
- $h > 0$ – число ключевых параметров;
- $s_{ij} > 0$ – максимальный размер данных в j-м столбце i-й таблицы в байтах.

Значения o_{data} и o_{index} рассчитываются для каждой задачи предметной области при однократном занесении в РБД не менее 16000000 записей. При достижении такого числа записей эти значения стабилизируется.

Расчет накладных расходов для хранения данных выполняется по формуле:

$$o_{data} = \frac{M_{data} - D^*}{\sum_{i=1}^{t} r_i^*(x_1, x_2, \dots, x_h)}, \quad (37)$$

где M_{data} – это размер данных без учета индексов таблиц при записи $r_i^*(x_1, x_2, \dots, x_h)$ строк i-й таблицы, $r_i^* < r_i$, $i = \overline{1, t}$, который занимает БД, размещенная на вычислительных узлах.

Расчет накладных расходов для хранения индексов выполняется по формуле:

$$o_{index} = \frac{M_{index} - M_{data}}{\sum_{i=1}^{t} r_i^*(x_1, x_2, \dots, x_h)}, \quad (38)$$

где M_{index} – это размер данных с учетом индексов таблиц при записи $r_i^*(x_1, x_2, \dots, x_h)$ строк i-й таблицы, $r_i^* < r_i$, $i = \overline{1, t}$, который занимает БД, размещенная на вычислительных узлах.

Расчет накладных расходов на данные проводится с помощью алгоритма А.7:

I. Модификация исходной БД, в которой отключаются индексы во всех таблицах.

II. Занесение в БД 16000000 или более записей.

III. Выполнение следующих двух запросов к Apache Ignite с помощью SQL-консоли, которая входит в его состав:

а. Получение размера БД в ОП в байтах.

б. Получение числа записей в БД во всех таблицах.

IV. Вычисление удельных накладных расходов на хранение одной записи в БД.

Расчет накладных расходов на индексы проводится с помощью алгоритма А.8:

I. Модификация исходной БД, в которой включаются индексы во всех таблицах.

II. Занесение в БД 16000000 или более записей.

III. Выполнение следующих двух запросов к Apache Ignite с помощью SQL-консоли, которая входит в его состав:

а. Получение размера БД в ОП в байтах.

б. Получение числа записей в БД во всех таблицах.

IV. Вычисление удельных накладных расходов на хранение индексов.

Функция $r_i(x_1, x_2, \dots, x_h)$ конкретизируется для каждой i-ой таблицы с учетом ключевых параметров предметной области $x_1, x_2, \dots, x_h$. Предложенная методика является универсальной для прогнозирования требуемого размера ОП при решении задач с использованием кластеров Apache Ignite в разных предметных областях. Специфика предметной области определяется ее ключевыми параметрами $x_1, x_2, \dots, x_h$. В отличие от методик, представленных в [144, 145], данная методика не использует субъективные оценки размера ОП, требуемой для хранения данных и индексов. Она позволяет достаточно точно рассчитать требуемый объем ОП, опираясь на знания о структуре исходных данных и ключевых параметрах предметной области, а также на результаты одного конкретного эксперимента.

Число узлов, выделяемых в дальнейшем для решения задачи, определяется следующей формулой:

$$1 \le y = \left\lceil \frac{S}{p_l(1-k_l^{os})} \right\rceil \le q, \qquad (39)$$

где p_l – доступный объем памяти l-го узла в ГБ, $0 < k_l^{os} \leq 1$ – коэффициент использования ОП операционной системой l-го узла, q – квота на число узлов РВС для решения задачи пользователем, определяемая административной политикой РВС, $l \in \overline{1,e}$, e – число узлов РВС. Предполагаем, что все узлы РВС являются однородными. Определение требуемого числа узлов кластера Apache Ignite с учетом требуемого размера ОП и квот на их использование, а также балансировка нагрузки осуществляется с помощью оптимизационного алгоритма [147].

3.4. Сценарии выполнения научных рабочих процессов

В FDE-SWFs НРП может быть реализован следующими способами, рассмотренными в работах [10, 15, 25, 27, 28]: на основе композиции WPS-сервисов, представленной на языке программирования Python; в виде отдельного WPS-сервиса, который реализуется традиционным НРП на основе исполняемых модулей и заданий для Condor DAGMan и HTCondor; с помощью стандартизированного декларативного языка BPEL. В последнем случае НРП может использоваться любыми внешними WMS, поддерживающими BPEL. На рисунке 3.10 представлены сценарии выполнения НРП под управлением FDE-SWFs.

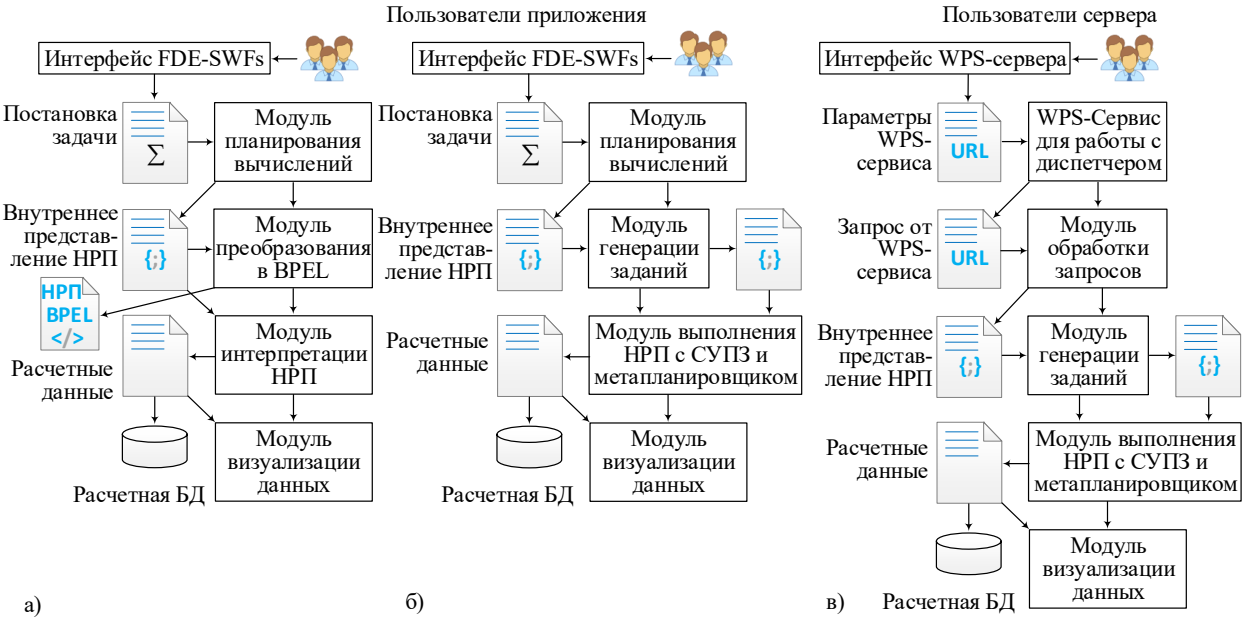


Рисунок 3.10 – Сценарии выполнения НРП:
Сценарий 1 (а), Сценарий 2 (б) и Сценарий 3 (в)

В рамках Сценария 1 пользователь формулирует постановку задачи. Используя вычислительную модель и алгоритмы планирования вычислений, модуль планирования вычислений выполняет проектирование и построение НРП во внутреннем представлении. Модуль преобразования производит трансформацию НРП из внутреннего представления на BPEL. Модуль интерпретации осуществляет асинхронное параллельное выполнение операций НРП, представленных в виде композиции WPS-сервисов. Расчетные данные сохраняются в расчетной БД. По завершению выполнения НРП пользователь может осуществить визуализацию данных с целью их дальнейшего анализа.

Сценарий 2, в отличии от Сценария 1, предполагает, что операции НРП представлены модулями. Пользователь формулирует постановку задачи. Используя вычислительную модель и алгоритмы планирования вычислений, модуль планирования вычислений выполняет проектирование и построение НРП во внутреннем представлении. Затем он генерирует паспорта заданий для Condor DAGMan и HTCondor по выполнению НРП и его модулей. Подсистема контейнеризации осуществляет выделение требуемых ресурсов, подготовку образов в соответствии с классами ресурсов и прикладными модулями НРП с последующим запуском контейнеров на данных ресурсах. НРП выполняется по готовности ресурсов. После завершения его выполнения диспетчер сохраняет результаты в расчетную БД и осуществляет визуализацию расчетных данных.

Выполнение НРП с помощью WPS-сервисов реализуется Сценарием 3. FDE-SWFs автоматизирует создание, регистрацию и применение WPS-сервисов, представляющих НРП. В частности, FDE-SWFs автоматически регистрирует программные модули и НРП в виде асинхронных WPS-сервисов в каталогах на WPS-сервера. НРП могут включать вызовы других WPS-сервисов, что позволяет работать с наборами сервисов. В FDE-SWFs поддерживается возможность обмена файлами между WPS-сервисами в качестве их параметров.

Обеспечение отказоустойчивости выполнения традиционных НРП осуществляется встроенными средствами HTCondor. Для обеспечения отказоустойчивости сервис-ориентированных НРП выполняется тестирование

сервисов для определения среднего времени их выполнения на узлах ПОВС. Допустимое время выполнения сервисов задается прикладным разработчиком на этапе конфигурирования СОП. Диспетчер НРП в процессе выполнения сервисов интерпретатором НРП получает информацию от системы мониторинга и при превышении допустимого времени выполнения сервиса выполняет его перезапуск. Схема обеспечения отказоустойчивости выполнения сервис-ориентированных НРП приведения на рисунке 3.11.

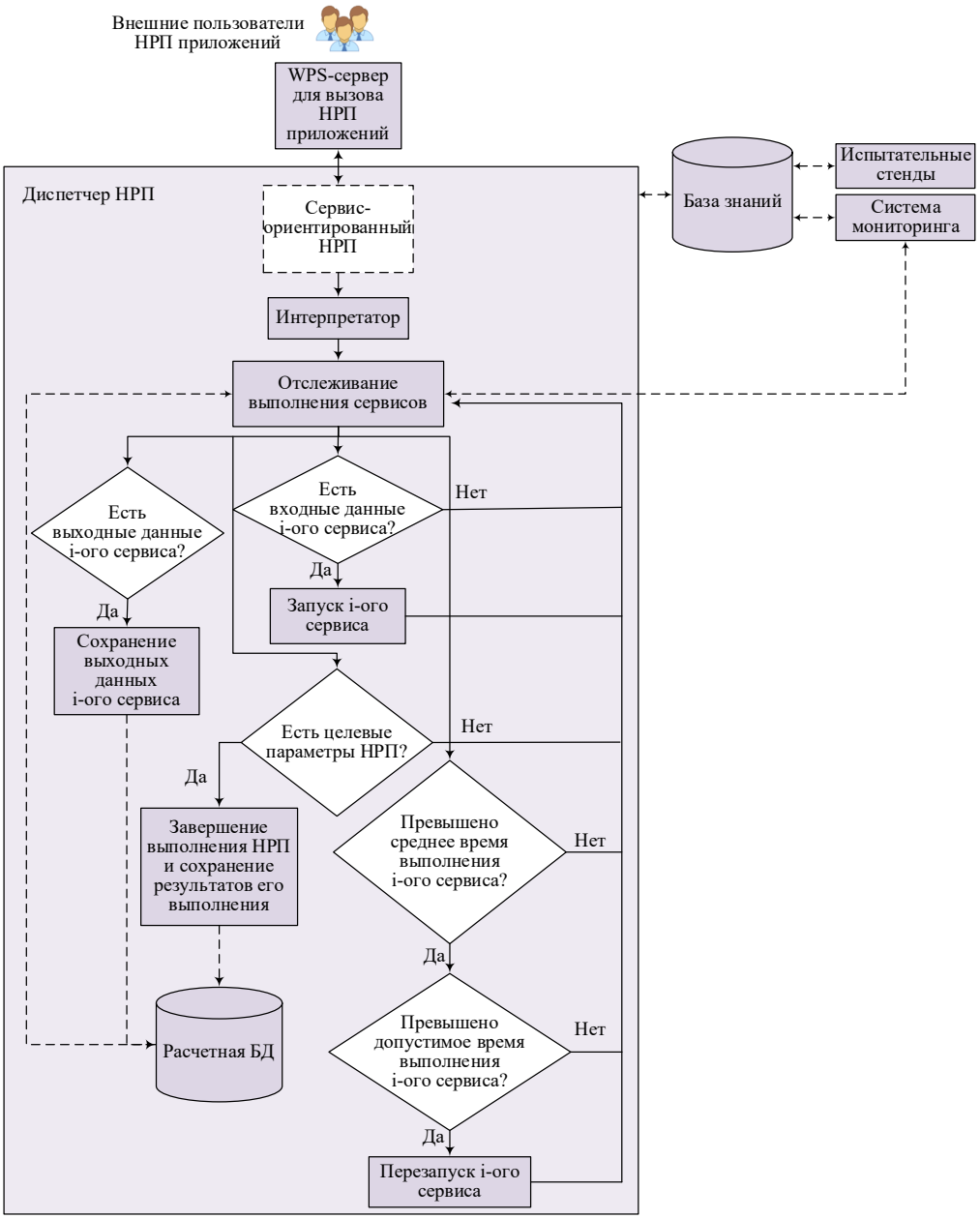


Рисунок 3.11 – Схема обеспечения отказоустойчивости выполнения сервис-ориентированных НРП

3.5. Средства создания испытательных стендов

В рамках FDE-SWFs разработаны средства для создания испытательных стендов, предназначенных для тестирования СОП и их компонентов [7, 31-33, 35]. Испытательные стенды широко используются в научной и промышленной сферах [148, 149]. В частности, они успешно применяются при создании, разработке и использовании информационных, вычислительных и телекоммуникационных систем и технологий.

Термин испытательный стенд (англ., testbed) [150] широко используется в научных исследованиях, в частности, в рамках создания, развития и использования информационно-вычислительных и телекоммуникационных систем и технологий, для определения программно-аппаратной среды, специализированной для проведения экспериментов, связанных с тестированием и испытанием новых алгоритмических и программных разработок, различных технологических решений и источников данных, а также систем их извлечения, обработки, передачи, хранения и анализа, перед внедрением создаваемых или модифицируемых систем и технологий. Испытательные стенды создаются в физической или виртуальной форме с целью обеспечения разработчиков контролируемой средой для проверки работоспособности, удовлетворения функциональным и системным требованиям, производительности, эффективности использования ресурсов и других свойств исследуемых систем и технологий.

Очевидно, что испытательные стенды очень полезны для разработчиков, так как позволяют протестировать прикладное ПО, интегрированное с системным, перед его развертыванием на ресурсах. Стенды успешно используются в различных областях, связанных с распределенными вычислениями. Например, известны следующие разработки для тестирования прикладного ПО в распределенной среде:

- экспериментальный высокопроизводительный испытательный стенд Grid [151];
- масштабируемый тестовый стенд IoT для гетерогенных устройств [152];

– методы и инструменты для тестирования аппаратных и программных компонентов в системах реального времени [153];

– тестовый стенд OpenStack для оценки его применимости [154].

В рамках FDE-SWFs испытательные стенды создаются в виде СРП, дополненных набором специальных системных операций. СРП выполняются диспетчером НРП. Общая структура испытательного стенда, разрабатываемого с помощью FDE-SWFs, представлена на рисунке 3.12.

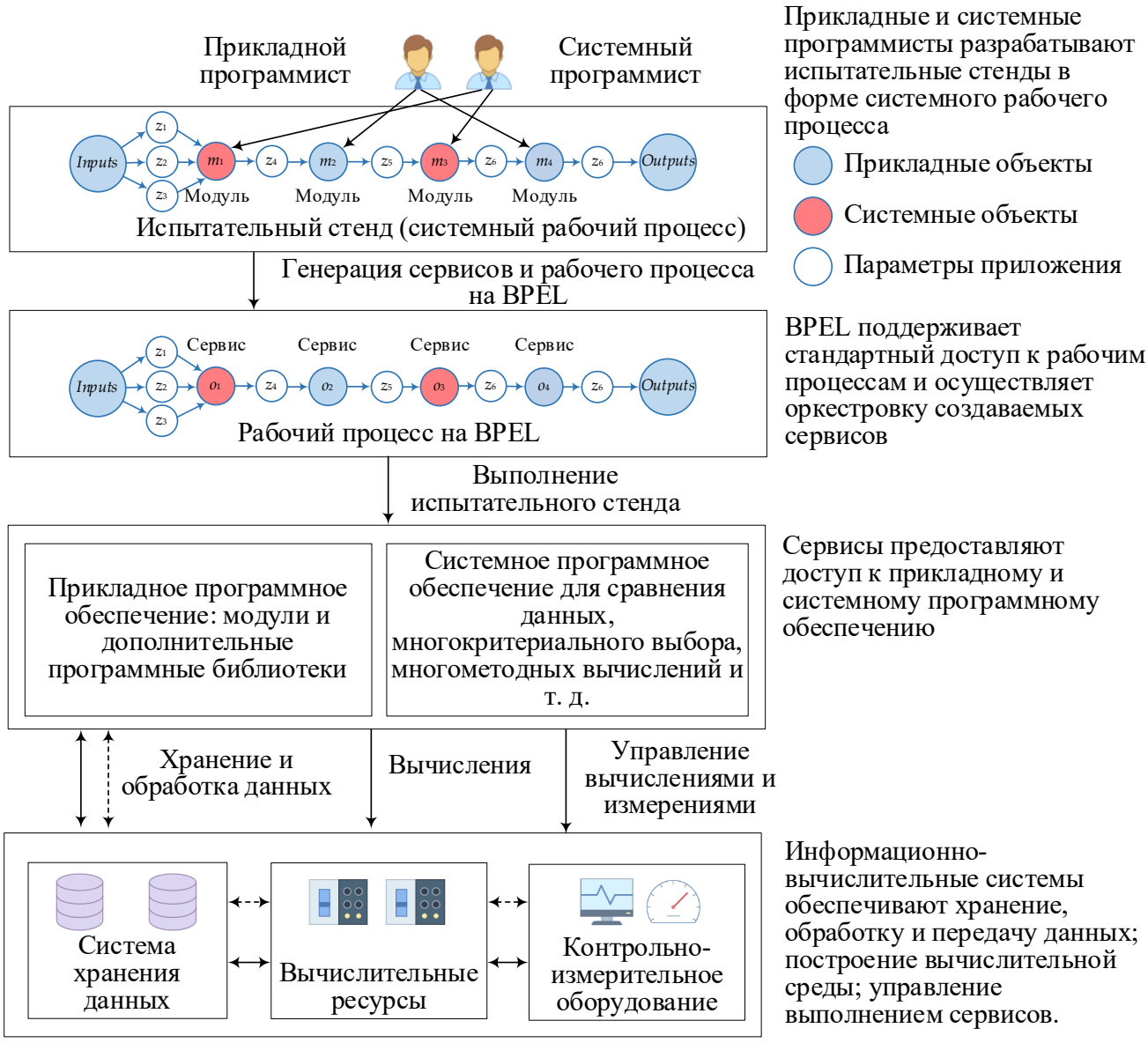


Рисунок 3.12 – Структура испытательного стенда

СРП представляется на стандартизированном декларативном языке BPEL и в дальнейшем может использоваться любыми внешними системами управления

рабочими процессами, поддерживающими BPEL.

В рамках СОА применяются сервисы, представляющие модули разрабатываемого приложения и внешние приложения. Внешние приложения реализуют дополнительные системные операции обработки и анализа данных, получения информации с контрольно-измерительного оборудования и многокритериального выбора оптимальных результатов вычислений.

На испытательном стенде в процессе обработки данных может осуществляться их валидация. В частности, для временных рядов пространственно-распределенных данных может выполняться проверка их типов, форматов, полноты, непротиворечивости и логической последовательности представления, допустимых диапазонов значений и заданных ограничений, а также выявление разного рода аномалий (индивидуальных и/или групповых). Операции валидации данных реализуются системными модулями с использованием специализированных библиотек сравнения, анализа и корректировки временных рядов.

Тестирование модулей, сервисов и рабочих процессов приложений производится на тестовых наборах данных, предоставляемых разработчиками приложений. В процессе тестирования оценивается работоспособность вычислительных сущностей, точность вычислений, а также эффективность использования вычислительных ресурсов, включая загрузку процессора, использование оперативной и дисковой памяти, балансировку нагрузки между разными ресурсами и другие параметры.

Вычислительная история валидации данных и тестирования модулей, сервисов и рабочих процессов отображается в специальных лог-файлах. Лог-файлы включаются в вычислительную модель приложения в качестве системных параметров.

3.6. Средства поддержки автоматизации интеграции и контейнеризации прикладного и системного программного обеспечения

Разработаны программные средства для автоматизации поддержки контейнеризации ПСПО, основанные на оригинальной методике [10, 13, 15, 25]. Подсистема контейнеризации диспетчера НРП осуществляет выделение требуемых ресурсов, подготовку образов в соответствии с классами ресурсов и прикладными модулями НРП с последующим запуском контейнеров (рисунок 3.13).

Рисунок 3.13 – Схема интеграции и контейнеризации ПСПО

Ниже приведены основные этапы подготовки и запуска вычислительной среды, рассмотренные в работе [13]:

– интеграция СУПЗ, метапланировщиков, WPS-сервера и компонентов диспетчера НРП из Git-репозитория в базовый образ БО:СПО модулем подготовки вычислительной среды;

– построение базовых образов БО:$К_1$, . . . , БО:$К_n$ путем адаптации образа

БО:СПО$_k$ различным классам ресурсов;

- интеграция в адаптированные образы О:ППО$_1$, . . . , О:ППО$_n$ и внешних библиотек из Git-репозитория, а также подключение временной БД и системы мониторинга;
- сборка, размещение, запуск и контроль состояния образов на ресурсах;
- выполнение НРП в среде;
- сохранение результатов, визуализация данных и освобождение ресурсов.

В целом процесс выполнения НРП в контейнеризированной вычислительной среде состоит в следующем. Специалист предметной области (конечный пользователь) формулирует постановку задачи, подготавливает исходные данные и выбирает необходимые ресурсы (ресурсы суперкомпьютерных ЦКП, облачных платформ, собственных высокопроизводительных серверов и др.) с учетом квот на их использование и способы доступа к ним. Диспетчер автоматически строит НРП и генерирует спецификацию вычислительного задания. Подсистема контейнеризации осуществляет выделение требуемых ресурсов, подготовку образов в соответствии с классами ресурсов и прикладными модулями НРП с последующим запуском контейнеров на данных ресурсах. По готовности ресурсов для запуска НРП передается сообщение подсистеме управления вычислениями на базе HTCondor, которая также интегрирована в образы. После завершения выполнения НРП компоненты диспетчера НРП осуществляют отправку полученных результатов в расчетную БД долгосрочного хранения и визуализацию данных. Подсистема контейнеризации диспетчера НРП стирает временную БД, освобождает ресурсы и расформировывает среду.

Построение контейнеров основано на применении сценариев формирования образов. Далее приведены примеры сценариев формирования образа для построения контейнеризированной вычислительной среды на базе ОС Ubuntu 22.04 и СУПЗ HTCondor и запуска приложения.

Пример сценария для построения образа system_modules с интегрированными системными модулями представлен ниже:

```
# задание базового образа для построения контейнера задается образ Ubuntu 22.04.
FROM ubuntu:22.04
# обновление локальной базы данных списков доступных пакетов из официальных репозиториев
RUN apt-get update
# установка системных утилит: curl, nano, nodejs
RUN apt-get install -y curl && apt-get install -y nano && apt-get install -y nodejs
# установки СУПЗ HTCondor
RUN curl -fsSL https://get.htcondor.org | /bin/bash -s -- --no-dry-run
# удаление стандартного файла конфигурации HTCondor.
RUN rm /etc/condor/config.d/00-minicondor
# копирование конфигурационных файлов HTCondor с хостовой машины в контейнер.
COPY condor /etc/condor
# копирование компонентов диспетчера НРП с хостовой машины в контейнер
COPY scheduler /home/scheduler
# указание TPC-порта 3000 для доступа извне к API компонентов
EXPOSE 3000
```

Листинг 3.1. Сценарий построения образа

Образ может адаптироваться под класс ресурсов. Например, при организации кластера IMDG в адаптированные образы добавляется ПО Apache Ignite.

Пример сценария для построения адаптированного образа resource_class_imdg для поддержки технологии In-Memory Data Grid:

```
# в качестве базового образа задается образ system_modules с системными компонентами
FROM system_modules
# копируется папка с модулями и библиотеками apache-ignite-2.6
COPY apache-ignite-2.6 /home/apache-ignite-2.6
```

Листинг 3.2. Сценарий построения адаптированного образа

В образ с интегрированными прикладными модулями включаются сами модули прикладного ПО, а также библиотеки и другие зависимости, необходимые для выполнения НРП.

Пример сценария для построения образа crit_elem с прикладными модулями приложения для поиска критических элементов:

```
# в качестве базового образа задается адаптированный образ
system_modules для поддержки технологии In-Memory Data Grid
FROM resource_class_imdg
# копирование РППП поиска критических элементов
COPY crit_elem /home/app/crit_elem
# запуск диспетчера НРП
ENTRYPOINT [“node”, “/home/scheduler/server.js”]
```

Листинг 3.3. Сценарий построения адаптированного образа с модулями

При разработке и размещении WPS-сервисов на WPS-сервере часто возникает проблема несовместимости версий библиотек используемого ими прикладного ПО. Для создания изолированной среды выполнения прикладного ПО, используемого WPS-сервисами, предлагается создавать отдельные контейнеры, в которые помещается прикладное ПО.

Пример сценария формирования образа для изоляции прикладного ПО WPS-сервиса:

```
# в качестве базового образа задается образ с поддержкой языка
программирования Python версии 3.10
FROM python:3.10
# обновление локальной базы данных списков доступных пакетов из
официальных репозиториев
RUN apt-get update
# установка необходимых версий библиотек numpy и pandas
RUN apt-get install -y python3-pip && pip install pandas==1.5.3
datetime urllib3 numpy==1.24.4
# установка библиотеки GSEE для расчета радиации на солнечной панели
RUN pip install -e git+https://github.com/renewables-
```

```
ninja/gsee.git#egg=gsee
# задание рабочий директории
WORKDIR /app
# запуск приложения
ENTRYPOINT ["python","solar_energy.py"]
```

Листинг 3.4. Сценарий для изоляции прикладного ПО WPS-сервиса

Доставка образов и развертывание контейнеризированной среды на ресурсах осуществляется с помощью системы. Для описания инструкций этой системе используются файлы Playbook в текстовом формате YAML.

Для описания ресурсов среды используются ini-файлы. Ниже приведен пример файла для описания среды, состоящей из 4 узлов СУПЗ HTCondor: узла central_manager для координации работы всего кластера и трех вычислительных узлов worker.

```
[central_manager]
manager ansible_host=192.168.56.110 ansible_user=manager
[execute_nodes]
worker ansible_host=192.168.56.111 ansible_user=execute
worker ansible_host=192.168.56.112 ansible_user=execute
worker ansible_host=192.168.56.113 ansible_user=execute
```

Листинг 3.5. Пример ini-файла системы Ansible для описания среды

Для описания инструкций автоматизации развертывания и конфигурирования ресурсов среды в Ansible используются файлы Playbook в текстовом формате YAML. Пример файла для конфигурирования узла central_manager приведен в Приложении Ж.

3.7. Средства и методы визуализации данных

Разработка методов и средств визуализации данных описана в работе [12]. Визуализация информации является актуальной задачей в широком спектре современных направлений исследований, таких как картография и климатология, геоинформатика, социально-экономическое развитие регионов и природных территорий, обработка данных дистанционного зондирования Земли, навигация в мобильной робототехнике, прогнозирование, интеллектуальный анализ

результатов моделирования природных и технических систем и др. [155]. При этом, как правило, основное внимание уделяется методам и средствам визуализации многомерных временных рядов данных, являющихся результатом наблюдений или выполнения натурных и/или вычислительных экспериментов, которые позволяют настраивать (конфигурировать) отображение данных.

Визуализация призвана облегчить восприятие информации с целью ее дальнейшего анализа, выявления закономерностей в зависимости и влиянии параметров исследуемых объектов и систем, а также поддержки принятия решений экспертами [156]. Поэтому важно иметь в наличии развитые методы и средства визуализации как первичных, так и вторичных данных, прошедших предварительную обработку и агрегирование. Это в равной степени относится как к визуализации текущих данных, собираемых различными системами измерения и контроля, так и к накопленной ретроспективной информации.

Зачастую визуализация данных реализует простейшую задачу отображения зависимости функции одной переменной (параметра). Сложность визуализации возрастает при отображении зависимости функции от нескольких переменных (параметров). Разнообразие данных и необходимость обеспечения их четкого восприятия обуславливает применение набора атрибутов процесса визуализации, к основным из которых относятся размер, форма, ориентация, цвет, текстура и значение цвета [157]. Эти атрибуты могут быть заданы с помощью различного рода шкал и уровней их значений. Поэтому любой метод визуализации должен сопровождаться инструментом, позволяющим настроить (конфигурировать) изображение данных.

Необходимость визуализации результатов расчетов, получаемых в результате выполнения ресурсоемких СОП, является актуальной проблемой в рамках решения задач экологического мониторинга [145].

В диссертации предложена спецификация параметров визуализации данных на языке JSON [158]. Спецификация определяет параметры визуализации данных, такие как: название, тип, цвет диаграммы, максимальные и минимальные значения данных по осям X и Y, название и местоположение легенды, размеры и начертания

шрифтов и т.д. Для разных видов диаграмм (диаграмма, график, круговая диаграмма, график с накоплением и др.) созданы типовые спецификации.

В листинге 3.6 приведены фрагменты типовой спецификации для построения графика. Параметр min (max) – это минимальное (максимальное) значение данных на координатной оси. Если параметры min и max явно не заданы, то подсистема визуализации автоматически определит и установит значения этих параметров. Параметр type – тип графика/диаграммы (line, spline, area, areaspline, column, bar, pie, scatter, gauge, arearange, areasplinerange и columnrange). Если в параметре series не заданы цвета рядов данных, то они определяются автоматически библиотекой Highcharts. Шаблон спецификации графика приведен в листинге 3.7.

```
chart: { type: ‘Тип графика/диаграммы’,
    title { text: ‘Название графика/диаграммы’,
          verticalAlign: `top | bottom | middle’,
          align: ‘left | right |center’},
    xAxis: {
        title: { text: 'Название оси X',
                 style: { fontSize: 'Размер шрифта подписи по оси X'
} },
        label: { style: { fontSize: ‘Размер шрифта’,  color:    ‘Цвет
шрифта’ } }
        min: 0, max: 100},
    yAxis: {
        title: { text: ‘Название оси Y’,
                 style: { fontSize: 'Размер шрифта подписи по оси Y'
} },
        min: 0, max: 100},
     legend: {
         itemStyle: { "color": "цвет", "fontSize": "размер",
                    "fontWeight": "начертание”},
        verticalAlign:`top | bottom | middle’, align: ‘left | right
     |center’ },
     marker: { symbol: ‘Тип маркера’,  radius: ‘Радиус’,
               lineColor: ‘Цвет контура маркера’,
               fillColor: ‘Цвет заливки маркера’},
     series: [{
          name: ‘Название легенды ряда данных’,
```

```
            marker: { symbol: 'Название вида маркера' },
            color: ‘Цвет линии данных’,
            lineWidth: ‘Толщина линии данных’}],
}
```

Листинг 3.6 – Фрагменты спецификации

```
Data Visualization <файл спецификации параметров визуализации расчетных данных на JSON> > <список сокращенных имен параметров предметной области приложения>
```

Листинг 3.7 – Шаблон спецификации параметров визуализации расчетных данных

В качестве библиотеки для визуализации данных выбрана библиотека Highcharts 11.2.0 [159] – одна из самых многофункциональных и популярных библиотек (см., например, AnyChart, Chart.js, Chartist.js и др. [160]), написанных на языке JavaScript, для построения графиков и диаграмм в формате HTML с рендерингом в формат SVG (VML), в том числе в интерактивном режиме. Данная библиотека является достаточно легковесной, поддерживает спектр разнообразных типов визуализаций (графиков, диаграмм и др.) и обеспечивает высокую производительность. Кроме того, функции библиотеки включают возможность автоматического выбора некоторых стилей графических объектов, если параметры этих стилей не заданы в спецификациях. Визуализация с использованием формата SVG осуществляется в стандартных браузерах, таких как Chrome, Firefox, Internet Explorer и др. Библиотека Highcharts имеет открытый исходный код и может бесплатно использоваться в некоммерческих целях. Разработано инструментальное средство на языке PHP 7.4.3, которое позволяет вывести результаты визуализации на экран и/или сохранить в виде файла в формате PNG. Примеры визуализации графика и графика с накоплением представлены на рисунках 3.14 и 3.15.

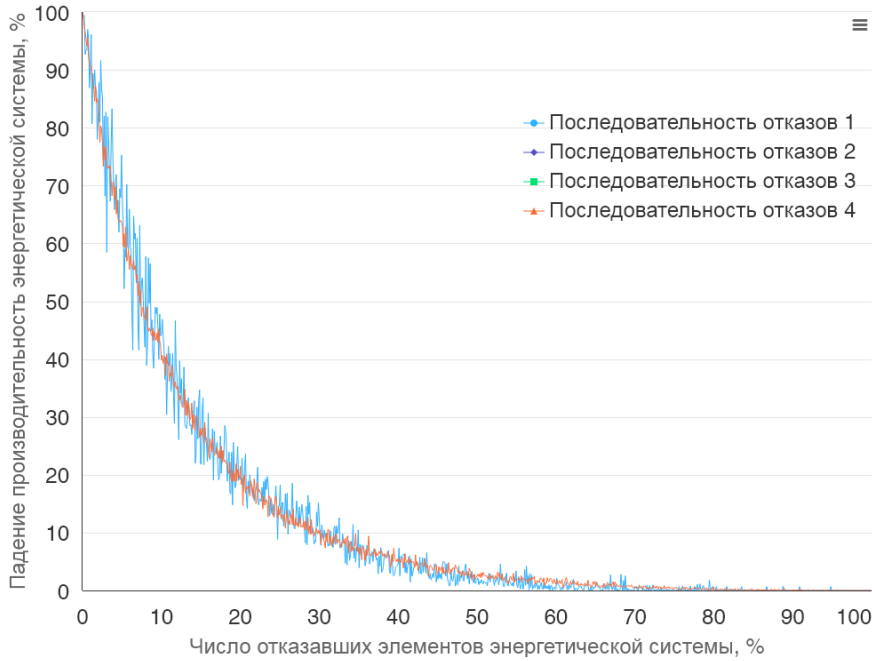


Рисунок 3.14 – Визуализация параметра на графике

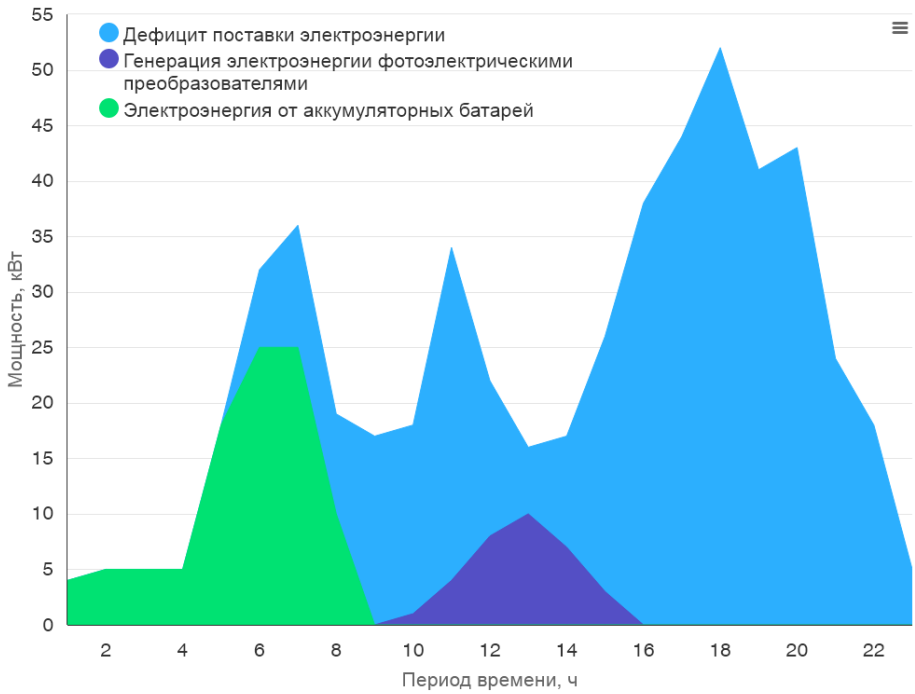


Рисунок 3.15 – Визуализация параметра на графике с накоплением

Разработано инструментальное средство на языке Python 3.8.10, которое выполняет визуализацию НРП в виде двудольного ориентированного графа. В качестве библиотеки для визуализации НРП используется библиотека Graphviz 0.20.1 [161] – одна из самых популярных библиотек (см., например, NetworkX, igraph, Graph-tool и др. [162]) для работы с графами на языке Python. Библиотека Graphviz представляет собой набор программ с открытым исходным

кодом для верстки и визуализации графов. Библиотека поддерживает веб-ориентированный интерактивный пользовательский интерфейс, включает набор вспомогательных инструментов и программных библиотек, а также обеспечивает привязку к различным форматам и языкам представления данных. В частности, Graphviz включает функции для изображения графов в форматах SVG для веб-страниц и Postscript для PDF-документов. Graphviz также поддерживает формат GXL – диалект языка XML. Граф описывается на простом и интуитивно понятном языке DOT (Graph Description Language). Кроме того, Graphviz поддерживает автоматическую раскладку графа с целью достижения оптимального расположения вершин и ребер графа на его изображении. Он также предоставляет подпрограммы для задания параметров конкретных графов, таких как цвета, шрифты, макеты узлов, стили линий, гиперссылки и пользовательские фигуры. На языке DOT разработан набор базовых спецификаций визуализации НРП. В листинге 3.8 представлен шаблон спецификации НРП. Пример визуализации НРП представлен на рисунке 3.16.

```
Workflow Visualization <файл спецификации параметров визуализации НРП на JSON> > <список сокращенных имен НРП>
```

Листинг 3.8 – Шаблон спецификации параметров визуализации НРП

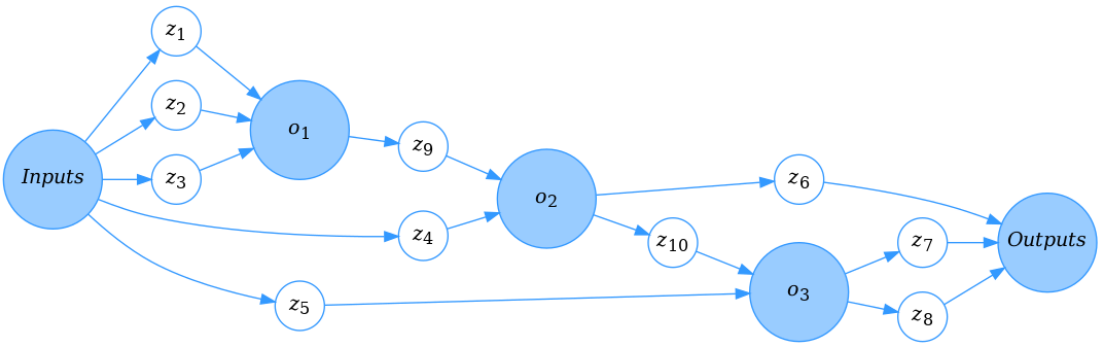


Рисунок 3.16 – Визуализация НРП

3.8. Выводы

В третьей главе получены следующие основные результаты:

- разработана архитектура, определены основные функциональные возможности и рассмотрены аспекты реализации ИК для создания и применения СОП;
- предложены сценарии выполнения НРП;
- в сравнении с системами подобного назначения реализованы уникальные возможности ИК по работе с БД, распределенными в ОП вычислительных ресурсов;
- разработана методика контейнеризации ПСПО; предложен новый подход к разработке и применению испытательных стендов для тестирования СОП и их компонентов;
- разработаны оригинальные средства визуализации НРП и расчетных данных.

Результаты исследований по данной главе опубликованы в [7, 8, 9, 10, 12-13, 15-17, 19-20, 23, 25-29, 31-33, 35]. В частности, архитектура ИК описана в работах [8, 10, 15, 17, 19-20, 27, 28, 34]. Конструктор вычислительной модели и НРП представлен в работах [8, 10, 15]. Генератор WPS-сервисов предложен в работе [8]. Управление вычислениями диспетчером НРП рассмотрено в работах [10, 12-13, 15-17, 25, 27-28, 36]. Поддержка стандарта WPS в ИК при разработке сервисов изложена в работах [26, 29]. Средства работы с распределенными БД рассмотрены в работах [9, 20, 23]. Сценарии выполнения НРП предложены в работах [10, 15, 25, 27, 28]. Средства создания испытательных стендов обсуждены в работах [7, 31-33, 35]. Средства поддержки интеграции и контейнеризации ПСПО рассмотрены в работах [10, 13, 15, 25]. Средства и методы визуализация данных описаны в работе [12].

Глава 4. Применение результатов диссертации при разработке сервис-ориентированных приложений

В четвертой главе приводятся примеры применения результатов диссертации. В частности, рассматриваются приложения, созданные с помощью ИК FDE-SWFs для решения задач структурно-параметрической оптимизации энергетических комплексов (ЭК) разных уровней их территориально-отраслевой иерархии, глобального анализа уязвимости ЭК и комплексной оценки критичности элементов ЭК. Демонстрируются преимущества данного комплекса в сравнении с системами подобного назначения, а также сокращение трудозатрат на этапах подготовки и проведения экспериментов.

4.1. Приложение для решения задач структурно-параметрической оптимизации энергетического комплекса

При моделировании энергетических комплексов необходимо выбрать метод оптимизации, который наилучшим образом соответствует характеристикам исследуемой модели, и настроить управляющие параметры этого метода. Как правило, выбор производится из большого спектра методов оптимизации. К сожалению, высокая вычислительная сложность такого выбора не позволяет исследователям проанализировать большое число методов оптимизации. Например, в работе [163] анализируются результаты моделирования четырех различных методов оптимизации. В работе [164] сравнивают шесть методов и несколько выбранных комбинаций технических параметров локального ЭК. Исследователи отмечают, что для детального анализа методов потребуется более 11 000 комбинаций технических параметров в трех различных пространственно-временных разрешениях. Однако используемые ими фреймворк и вычислительные ресурсы не могут справиться с такой вычислительной нагрузкой.

В рамках решения задачи структурно-параметрической оптимизации топливно-энергетического комплекса (ТЭК) России и локального ЭК населенного пункта, расположенного на Байкальской природной территории (БПТ),

разработано приложение, включающее следующие рабочие процессы: прикладной НРП для решения задач структурно-параметрической оптимизации ЭК и испытательный стенд (СРП) для тестирования прикладного НРП и его компонентов [7, 14, 17, 34]. На испытательном стенде параллельно выполнены и оценены 63 эволюционных метода оптимизации из библиотеки PaGMO [165]. Тестирование и многокритериальный выбор методов из Парето-оптимального множества выполнялся за заданное допустимое время (10 ч для ТЭК и 1 ч для локального ЭК) с учетом ограничений по двум группам критериев: суммарным метрикам живучести конфигураций ЭК и показателям эффективности использования ресурсов. Затем с помощью прикладного НРП, реализующего выбранный метод оптимизации, была успешно решена задача поиска оптимальных конфигураций локального ЭК населенного пункта, расположенного на БПТ [14, 34].

Прикладной НРП для решения задачи структурно-параметрической оптимизации ЭК представлен в виде композиции операций, реализуемых сервисами (рисунок 4.1).

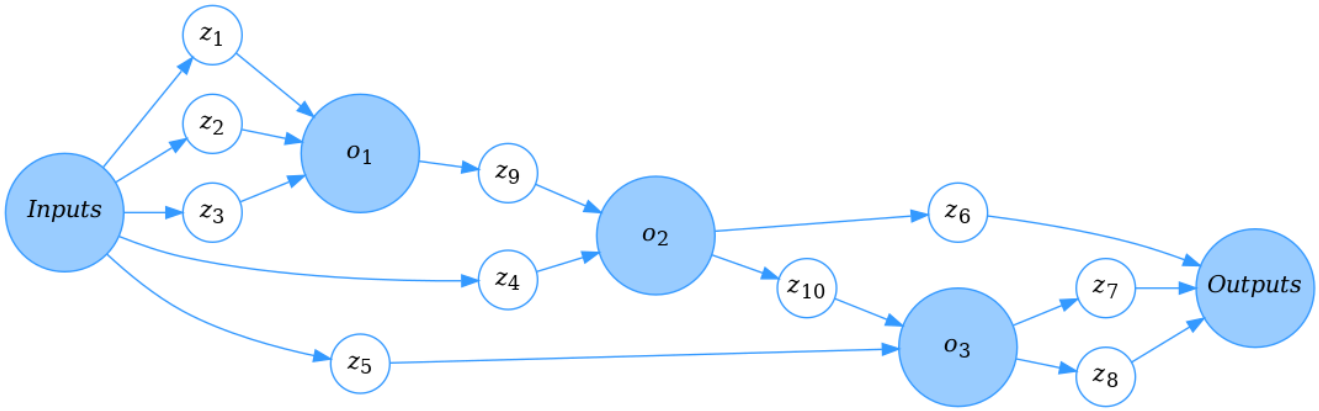


Рисунок 4.1 – Прикладной НРП

Операция o_1 реализует структурно-параметрическую оптимизацию ЭК с помощью наиболее подходящего метода оптимизации из библиотеки PaGMO, определяемого с помощью испытательного стенда. Далее операция o_2 обрабатывает результаты оптимизации. Наконец, операция o_3 создает отчет о результирующей конфигурации локального ЭК. Операции используют следующие параметры:

- z_1 – файл, содержащий настройки метода оптимизации;
- z_2 – входная БД, содержащая исходные данные для модели ЭК;
- z_3 – исходная БД для хранения результатов оптимизации локального ЭК;
- z_4 – SQL-скрипт для обработки результатов оптимизации;
- z_5 – SQL-скрипт для создания отчетов;
- z_6 – результат выполнения операции o_2;
- z_7 – результат выполнения операции o_3;
- z_8 – отчет о предпочтительных типах оборудования, их составе, характеристиках и местах расположения;
- z_9 – БД с результатами оптимизации;
- z_{10} – БД с обработанными и агрегированными результатами оптимизации для создания z_8.

Испытательный стенд разработан в виде системного НРП (рисунок 4.2). Он включает следующие операции:

- o_1 – операция для генерации списка файлов с именами методов;
- o_2 – системная операция для дезагрегирования списка файлов методов по отдельным директориям;
- o_3 – системная операция для дезагрегирования списка экземпляров модулей по отдельным директориям;
- o_4 – системная операция для изменения имени метода в файле конфигурации;
- o_5 – системная операция для изменения числа поколений в файле конфигурации;
- o_6 – системная операция для изменения размера популяции в файле конфигурации;
- o_7 – операция структурно-параметрической оптимизации;
- o_8 – системная операция для агрегирования результатов структурно-параметрической оптимизации;
- o_9 – системная операция многокритериального выбора метода.

Операции используют следующие параметры:

- z_1 – шаблон пути для генерации файлов с именами методов структурно-параметрической оптимизации;
- z_2 – список спецификаций методов;
- z_3 – список методов;
- z_4 – число поколений для методов;
- z_5 – размер популяции методов;
- z_6 – конфигурация структурно-параметрической оптимизации;
- z_7 и z_8 – расчетные БД;
- z_9 – список результатов структурно-параметрической оптимизации;
- z_{10} – имя наилучшего метода;
- z_{11} и z_{12} – журнал выполнения операций o_2 и o_3 соответственно;
- z_{13} содержит список экземпляров модулей и БД;
- z_{14} содержит агрегированные результаты структурно-параметрической оптимизации;
- z_{15}, z_{16} и z_{17} – конфигурация конфигурации структурно-параметрической оптимизации после изменения метода, его числа поколений и размера популяции.

Процесс работы испытательного стенда состоит из нескольких основных этапов: разработчик приложения настраивает вычислительную среду испытательного стенда из доступных вычислительных ресурсов и развертывает необходимое ПО на узлах среды, используя диспетчер НРП FDE-SWFs. Затем он размещает исходные данные в виде набора файлов на узлах, указанных в конфигурации среды. Далее разработчик определяет критерии оценки для исследуемых методов. Набор критериев включает в себя параметры предметной области и характеристики (время выполнения метода, загрузка процессора, использование ОП) выполнения модуля в узлах среды. Характеристики выполнения модуля (критерии эффективности использования вычислительных ресурсов) задаются для различных методов. Далее диспетчер НРП запускает вычислительный эксперимент. Он выполняет параллельный запуск экземпляров

модулей в вычислительной среде, мониторинг их выполнения, профилирование вычислительных процессов и сбор статистики по системным показателям выполнения экземпляров модуля. Каждый экземпляр модуля решает задачу одним из методов. Выбор метода определяется вариантом исходных данных, назначенных экземпляру модуля. Результаты вычислений каждого экземпляра модуля и системные метрики объединяются в параллельный список данных системными операциями НРП.

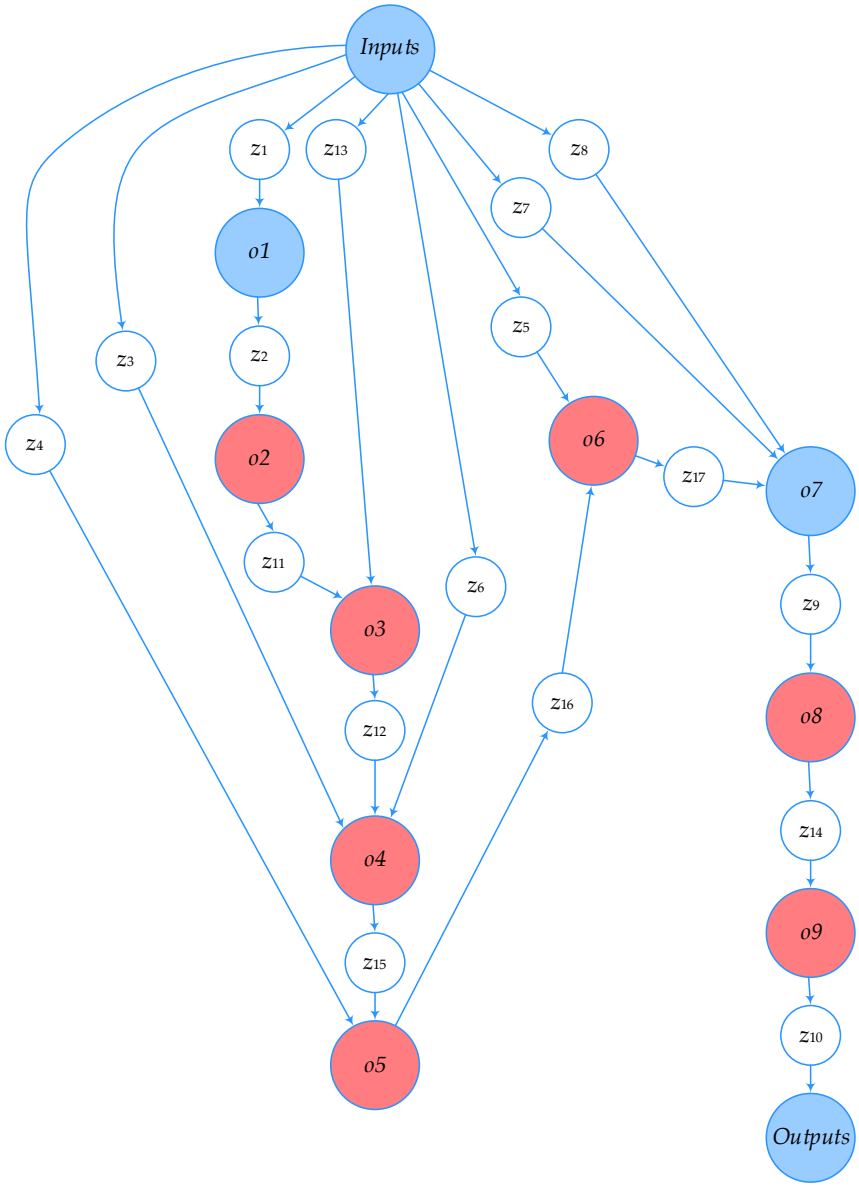


Рисунок 4.2 – Системный НРП (испытательный стенд)

Каждый элемент параллельного списка данных представляет собой вариант результатов списка данных соответственно. На основе этого списка значений критериев выделенная системная операция для многокритериального выбора

Парето-оптимального множества формирует набор методов и предлагает наиболее эффективный из них с учетом ранжирования этих критериев. Исходные данные и результаты вычислительного эксперимента сохраняются в базе расчетных данных.

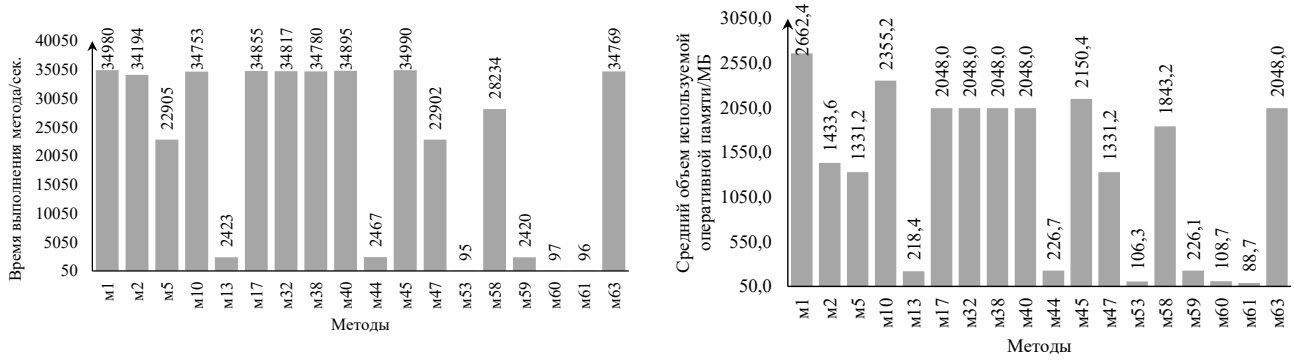
В таблице 4.1 приведены методы, выполнившие структурно-параметрическую оптимизацию ТЭК России в процессе тестирования за допустимое время (10 час).

Таблица 4.1 – Методы, успешно выполнившие структурно-параметрическую оптимизацию ТЭК России за 10 часов

Обозначение метода	Имя метода в библиотеки PaGMO
м1	Particle Swarm Optimization
м2	Grey Wolf Optimizer
м5	Single-objective Compass Search
м10	Simple Genetic Algorithm
м13	(N+1)-ES Simple Evolutionary Algorithm
м17	Particle Swarm Optimization Generational
м32	Self-adaptive DE (de_1220 aka pDE)
м38	Differential Evolution
м40	Self-adaptive DE (jDE and iDE)
м44	Multi-objective Improved Harmony Search
м45	Non-dominated Sorting PSO
м47	Multi-objective Compass Search
м53	Ipopt Unconstrained Problem
м58	PRAXIS
м59	Single-objective Improved Harmony Search
м60	Ipopt Constrained Problem
м61	Augmented Lagrangian Algorithm
м63	Non-dominated Sorting GA II

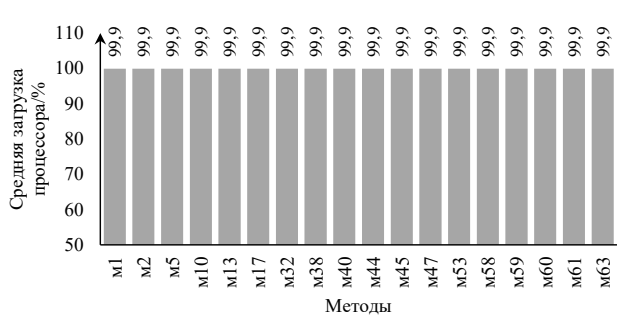
Методы м13, м44, м53, м59, м60 и м61 значительно превосходят другие методы по времени выполнения и среднему объему используемой ОП. При этом все методы демонстрируют схожую среднюю загрузку процессора. На рисунке 4.3

приведены критерии эффективности использования ресурсов – время выполнения метода (рисунок 4.3а), средний объем используемой ОП (рисунок 4.3б) и средняя загрузка процессора (рисунок 4.3в), а также показатели живучести ТЭК России – число эффективных мероприятий (рисунок 4.3г), длина дуг (рисунок 4.3д), суммарную метрику теплоснабжения (рисунок 4.3е), суммарную метрику электроснабжения (рисунок 4.3ж) и суммарную метрику снабжения природным газом (рисунок 4.3ж). С точки зрения критериев живучести наиболее приемлемым алгоритмом является метод м59, выбранный с помощью лексикографического правила многокритериального выбора. Другой вариант этого метода, м44, также близок к оптимальной производительности. Лексикографический алгоритм позволяет учитывать показатели предметной области, используемые для приоритизации. С точки зрения показателей эффективности использования ресурсов наилучшим является метод м53.

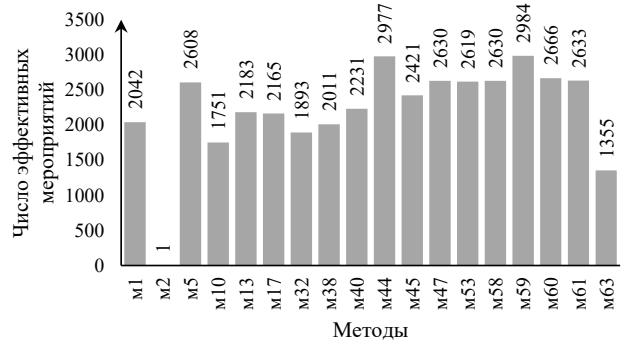


а)

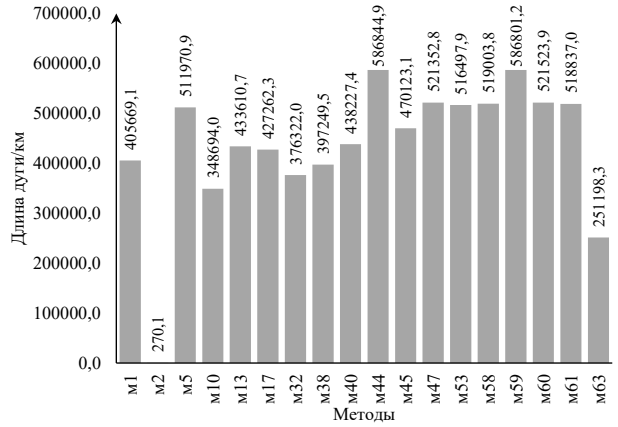
б)



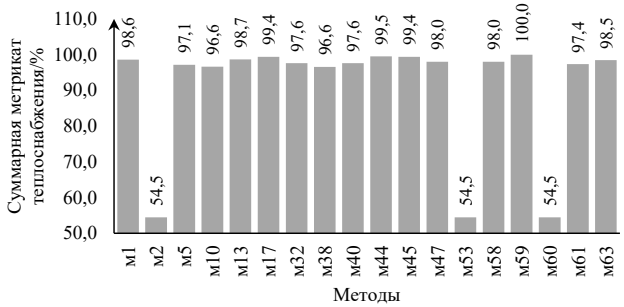
в)



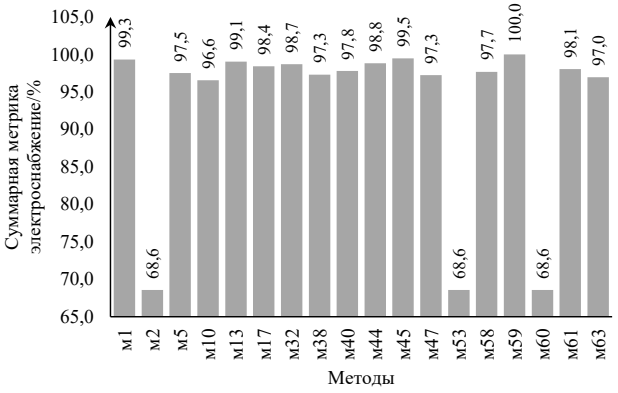
г)



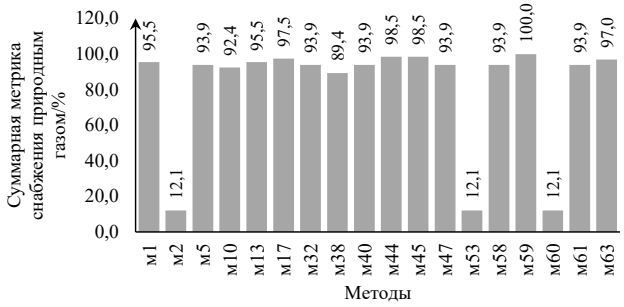
д)



е)



ж)



з)

Рисунок 4.3 – Показатели эффективности использования ресурсов – время выполнения алгоритма (а), средний объем используемой ОП (б) и средняя загрузка процессора (в); показатели живучести ТЭК России – число эффективных мероприятий (г), длина дуг (д), суммарная метрика теплоснабжения (е), суммарная метрика электроснабжения (ж) и суммарная метрика снабжения природным газом (з)

В таблице 4.2 приведены методы, выполнившие структурно-параметрическую оптимизацию локального ЭК в процессе тестирования за допустимое время (1 час).

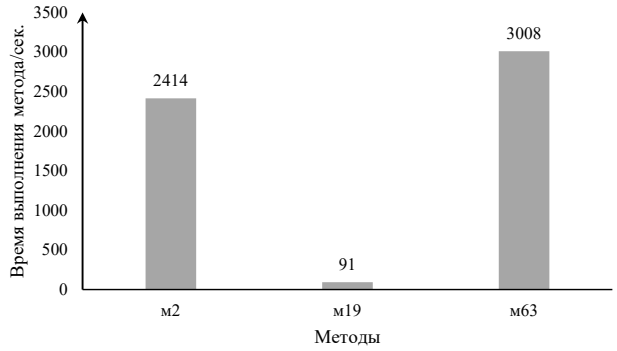
Таблица 4.2 – Методы, успешно выполнившие структурно-параметрическую оптимизацию локального ЭК за 1 час

Обозначение метода	Имя метода в библиотеки PaGMO
м1	Particle Swarm Optimization
м2	Grey Wolf Optimizer
м10	Simple Genetic Algorithm
м13	(N+1)-ES Simple Evolutionary Algorithm

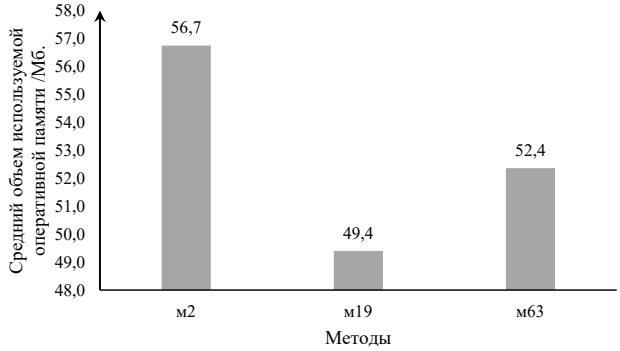
Таблица 4.2 – Методы, успешно выполнившие структурно-параметрическую оптимизацию локального ЭК за 1 час (продолжение)

м17	Particle Swarm Optimization Generational
м19	Covariance Matrix Adaptation Evo. Strategy
м32	Self-adaptive DE (de_1220 aka pDE)
м56	Exponential Natural Evolution Strategies
м63	Non-dominated Sorting GA II

В шести методах из девяти показатели живучести оказались ниже допустимого. Поэтому для дальнейшего рассмотрения отобраны только методы м2, м19 и м63. На рисунке 4.4 приведены показатели эффективности использования ресурсов – время выполнения метода (рисунок 4.4а), средний объем используемой ОП (рисунок 4.4б) и средняя загрузка процессора (рисунок 4.4в), а также показатели живучести локального ЭК – число эффективных мероприятий (рисунок 4.4г), объем инвестиций (рисунок 4.4д), суммарную метрику теплоснабжения (рисунок 4.4е) и суммарную метрику электроснабжения (рисунок 4.4ж). С точки зрения критериев живучести наиболее приемлемым методом является метод м63, выбранный с помощью лексикографического правила многокритериального выбора. С точки зрения эффективности использования ресурсов наилучшим является метод м19.



а)



б)

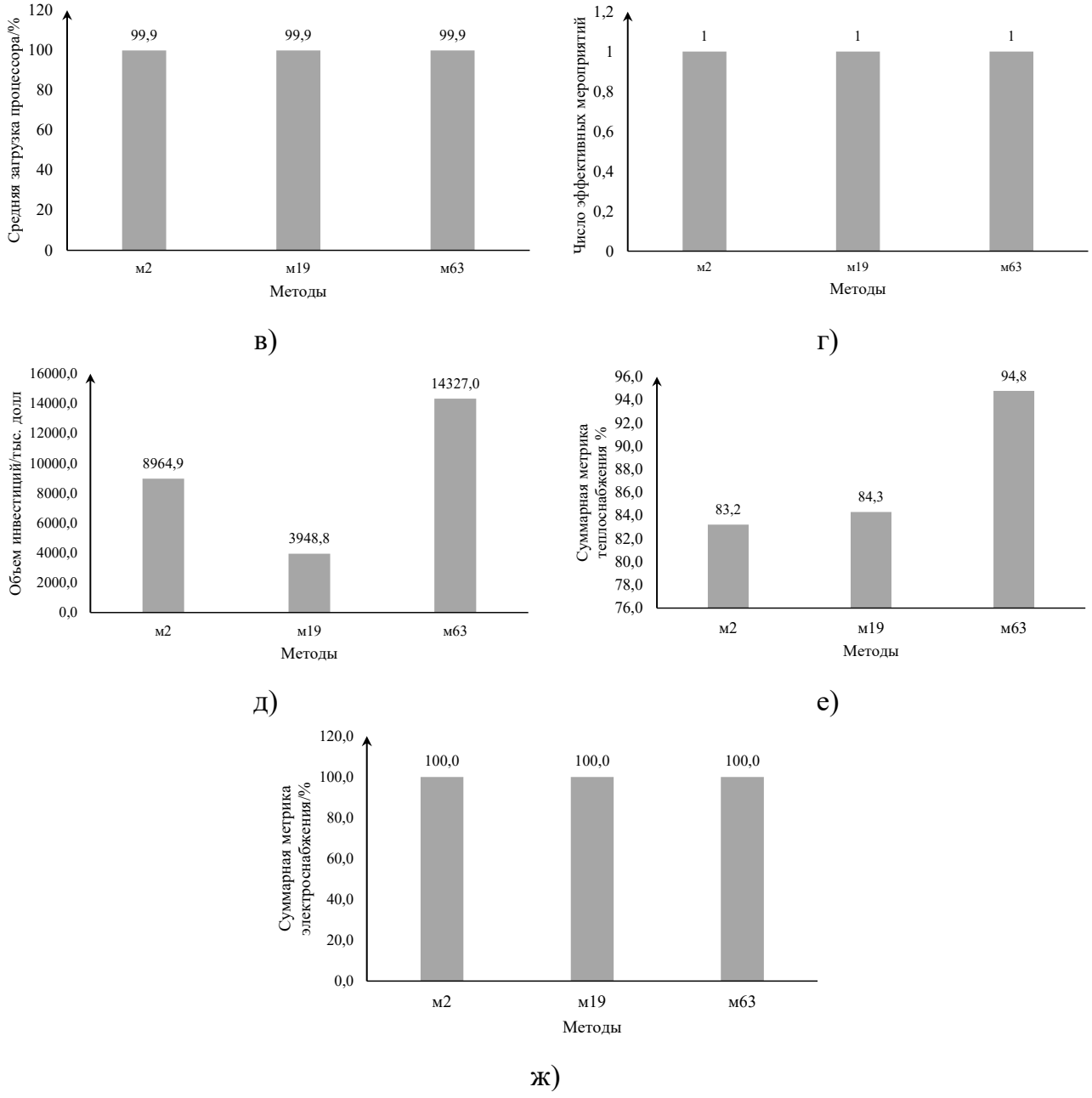
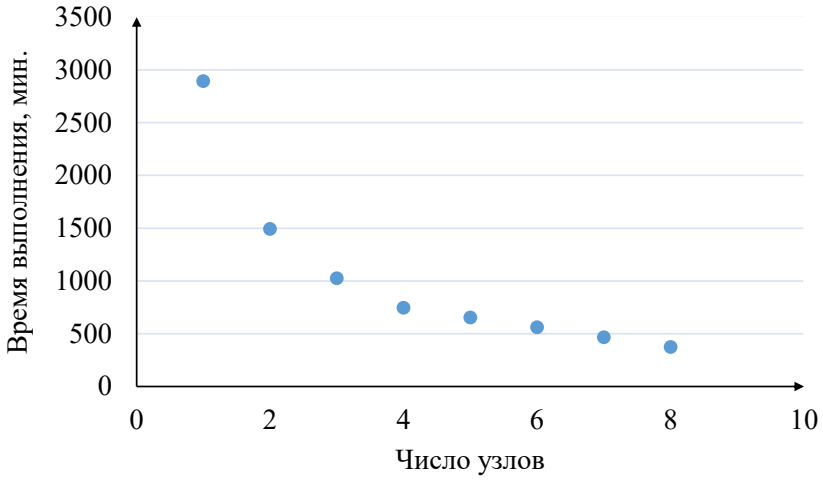


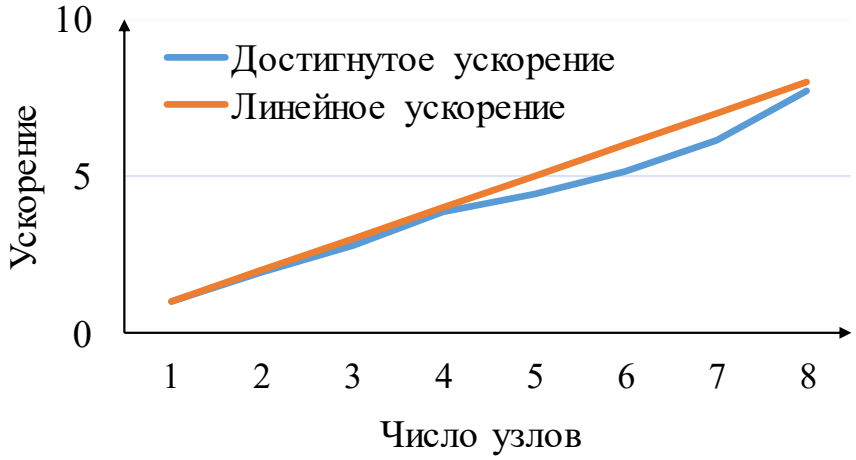
Рисунок 4.4 – Показатели эффективности использования ресурсов – время выполнения алгоритма (а), средний объем используемой ОП (б) и средняя загрузка процессора (в); показатели живучести локального ЭК – число эффективных мероприятий (г), объем инвестиций (д), суммарная метрика теплоснабжения (е) и суммарная метрика электроснабжения (ж)

Для оценки масштабируемости вычислений с помощью испытательного стенда методы были запущены в вычислительной среде, где число узлов увеличивалось от 1 до 8. Каждый из узлов имеет следующие характеристики: 2 × 16-ядерных процессора AMD Ryzen 9 5950X, 128 ГБ оперативной памяти и 2 ТБ

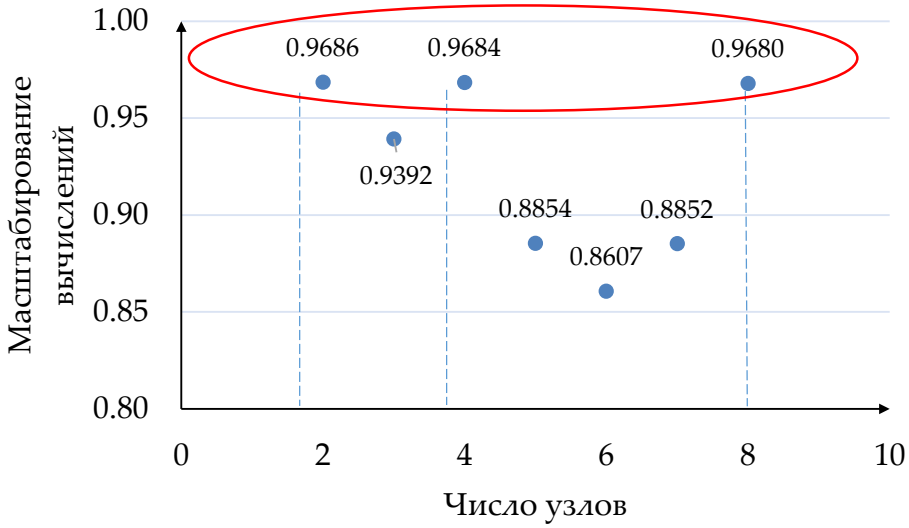
дискового пространства. Выполнено 3968 вычислительных заданий. На рисунке 4.5 представлены результаты экспериментального анализа.



(а) Время выполнения



(б) Ускорение вычислений



(в) Масштабирование вычислений

Рисунок 4.5 – Результаты экспериментального анализа: время выполнения (а), ускорение вычислений (б) и масштабирование вычислений (в).

Общее время решения задачи выбора метода показано на рисунке 4.5а. Очевидно, что увеличение числа узлов позволяет существенно сократить время выполнения. Можно видеть почти линейное ускорение вычислений (рисунок 4.5б). На рисунке 4.5в показано масштабирование вычислений, представленное отношением ускорения вычислений к числу узлов. Красным овалом выделено масштабирование вычислений на 2, 4 и 8 узлах, поскольку наиболее целесообразно использовать число узлов, кратное степени 2.

Проведена оценка сокращения трудозатрат на подготовку и проведение экспериментов с использованием испытательного стенда в сравнении с другими режимами выполнения работ: выполнение структурно-параметрической оптимизации модели локального ЭК в ручном режиме с помощью библиотеки PaGMO и использования известной системы Apache Airflow [58], предназначенной для управления рабочими процессами. В целом подготовка и проведение экспериментов по моделированию ЭК включала следующие основные этапы:

- разработка вычислительной модели приложения (э1);
- модификация кода модуля оптимизации для использования библиотеки алгоритмов оптимизации (э2);
- разработка приложения (э3);
- создание прикладного НРП (э4);
- организация и настройка ПОВС (э5);
- разработка тестового стенда (э6);
- модификация тестовых данных (э7);
- развертывание, настройка и контейнеризация ПО (э8);
- тестирование модулей и рабочих процессов (э9);
- выбор наилучшего метода (э10).

В FDE-SWFs все перечисленные этапы полностью или частично автоматизированы. На рисунке 4.6 показана оценка общего времени, затрачиваемого разработчиком приложения на подготовку экспериментов в ручном режиме, а также с использованием Apache Airflow и FDE-SWFs. Оценка

основана на кумулятивном итоговом показателе и приведена в минутах в соответствии с последовательностью основных этапов. Очевидно, что использование FDE-SWFs позволяет значительно сократить временные затраты на подготовку экспериментов по сравнению с Apache Airflow и ручным режимом. При использовании FDE-SWFs наибольшее сокращение временных затрат достигается при создании НРП, создании и настройке ПОВС, разработке испытательного стенда, развертывании, настройке и контейнеризации ПО, а также тестировании модулей и НРП. В случае Apache Airflow или ручного режима необходимо разработать дополнительные специализированные программы и скрипты, реализовать прикладные и системные сервисы, настроить и интегрировать программно-аппаратные ресурсы, подготовить спецификации для вычислительных заданий и переформатировать данные. Благодаря использованию Apache Airflow сократилось общее время подготовки эксперимента по сравнению с ручным режимом на 31.5 %. В случае использования FDE-SWFs общее время подготовки эксперимента сократилось более чем на 90% по сравнению со временем подготовки в ручном режиме.

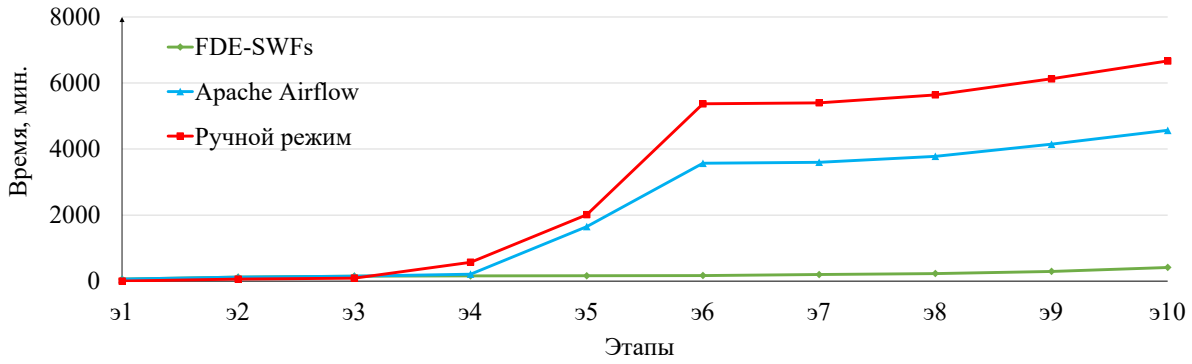


Рисунок 4.6 – Оценка совокупных временных затрат

4.2. Приложение для решения задачи глобального анализа уязвимости энергетического комплекса

В данном примере рассматривается одна из задач исследования уязвимости ЭК – глобальный анализ уязвимостей (англ., Global Vulnerability Analysis – GVA) [166]. Такой анализ позволяет оценить, насколько ухудшение производительности конфигурации ЭК зависит от параметров возмущения [167]. GVA основан на

моделировании серии сценариев возмущений с большим числом вышедших из строя элементов.

Исследователи, занимающиеся реализацией GVA, сталкиваются с ее высокой вычислительной сложностью. Для каждой конфигурации ЭК необходимо создавать наборы возможных сценариев возмущений, варьируя их параметры. Однако процесс решения задачи хорошо распараллеливается на множество независимых подзадач. Каждая подзадача исследует уязвимость конкретной конфигурации ЭК к набору сценариев возмущений.

Вычислительный эксперимент выполнен в работах [9, 17, 20, 23]. Пусть конфигурация ЭК описывается сетью $G = (V, U)$, где V – упорядоченное множество узлов, U – упорядоченное множество дуг. Сеть G является направленным графом, поэтому для каждой дуги $(i, j) \in U$ узел $i \in V$ является началом, а узел $j \in V$ – концом. Известные работы по исследованию уязвимости [168], как правило, основаны на оценке последствий множества сценариев возмущений, моделирующих отказ групп из k элементов ЭК. Множеством отказов F размером $k \geq 1$ называется группа из k номеров элементов сети G, отказ которых наступает одновременно: $F = \{c_l : 1 \leq c_l \leq |V| + |U|,\ l = (1, k)\}$, где c_l – это номер элемента сети G. Максимальное число возмущений рассчитывается по формуле (40):

$$f(n, m, k) = \sum_{i=1}^{k} \frac{(n+m)!}{((n+m-i)!\, i!)}, \tag{40}$$

где m и n это число дуг и узлов соответственно, которые задаются экспертом. Переменные m, n и k являются ключевыми параметрами предметной области, влияющими на изменение размера обрабатываемых данных.

Структура исходных данных включает 6 таблиц: DISTURBANCE, MES_ELEMENTS, SOLUTION_REPORTS, FAILED_ELEMENTS, TRANSPORT_DATA и REGIONAL_DATA. На рисунке 4.7 представлен НРП для полного перебора множеств отказов размера k. Он включает следующие операции: заполнение РБД данными о последствиях возмущений (o_1); обработка РБД

средствами Apache Ignite для оценки критичности элементов сети (o_2). В операциях используются следующие параметры:

- z_1 – БД с конфигурацией исследуемого ЭК;
- z_2 – результат запуска Apache Ignite;
- z_3 – размер группы отказов;
- z_4 – число сценариев возмущений;
- z_5 – адреса узлов кластера Apache Ignite;
- z_6 – результат заполнения распределенной БД;
- z_7 – результат оценки критичности элементов сети.

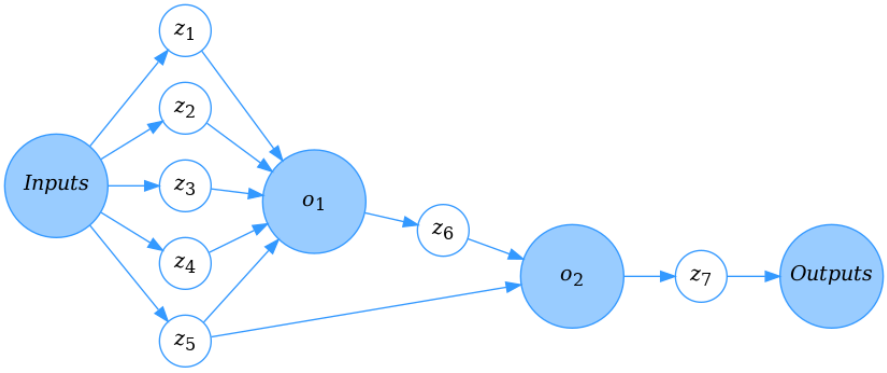


Рисунок 4.7 – НРП

Для каждой из таблиц определены соответственно функции $r_1 - r_6$. В рамках предложенной методики размеры этих таблиц рассчитывается соответственно по формулам (41)-(43), (44)-(46), (47)-(49), (50)-(52), (53)-(55) и (56)-(58), полученным путем конкретизации (34)-(36) с учетом ключевых параметров предметной области:

$$d_1 = r_1(n, m, k) \sum_{j=1}^{c_1} s_{1j}, \tag{41}$$

$$r_1(n, m, k) = \sum_{i=1}^{k} \frac{(n+m)!}{((n+m-i)!i!)}, \tag{42}$$

$$s_1 = \sum_{j=1}^{c_1} s_{1j}, \tag{43}$$

$$d_2 = r_2(n, m) \sum_{j=1}^{c_2} s_{2j}, \tag{44}$$

$$r_2(n, m) = n + m, \tag{45}$$

$$s_2 = \sum_{j=1}^{c_2} s_{2j}, \tag{46}$$

$$d_3 = r_3(n,m,k)\sum_{j=1}^{c_3} s_{3j}, \quad (47)$$

$$r_3(n,m,k) = r_1(n,m,k), \quad (48)$$

$$s_3 = \sum_{j=1}^{c_3} s_{3j}, \quad (49)$$

$$d_4 = r_4(n,m,k)\sum_{j=1}^{c_4} s_{4j}, \quad (50)$$

$$r_4(n,m,k) = \sum_{i=1}^{k}[ir_1(n,m,i)], \quad (51)$$

$$s_4 = \sum_{j=1}^{c_4} s_{4j}, \quad (52)$$

$$d_5 = r_5(n,m,k)\sum_{j=1}^{c_5} s_{5j}, \quad (53)$$

$$r_5(n,m,k) = r_1(n,m,k)|U|, \quad (54)$$

$$s_5 = \sum_{j=1}^{c_5} s_{5j}, \quad (55)$$

$$d_6 = r_6(n,m,k)\sum_{j=1}^{c_6} s_{6j}, \quad (56)$$

$$r_6(n,m,k) = r_1(n,m,k)|V|, \quad (57)$$

$$s_6 = \sum_{j=1}^{c_6} s_{6j}. \quad (58)$$

В качестве примера рассмотрены тестовые модели системы газоснабжения (модель 1), состоящей из 26 дуг и 20 узлов ($k = 8$, $m = 26$, $n = 0$) и ТЭК России (модель 2), включающего 2190 дуг и 1220 узлов ($k = 2, m = 2190, n = 1220$). По формулам (34)-(38) и (41)-(58) спрогнозирован требуемый размер ОП для исследования живучести указанных комплексов с помощью НРП на кластере Apache Ignite для двух пулов ресурсов: 8 узлов со следующими характеристиками – AMD Ryzen 9 5900X 12-Core Processor, 128 ГБ ОЗУ (пул 1); 2 узла со следующими характеристиками – AMD EPYC 9654 96-Core Processor, 768 ГБ ОЗУ (пул 2). Для модели системы газоснабжения экспериментальным путем получено, что величины накладных расходов на хранение данных (o_{data}) и индексов (o_{index}) стабилизируется на значениях 240 байт и 172 байта соответственно – рисунки 4.8 (а) и 4.8 (б). Для модели ТЭК величины накладных расходов на хранение данных (ХД) (o_{data}) и индексов (ХИ) (o_{index}) стабилизируются на значениях 20 байт и 206 байт соответственно – рисунки 4.8 (в) и 4.8 (г).

Методика (М), значения ключевых параметров (k, m и n), число записей (ЧЗ), пул используемых вычислительных ресурсов, прогнозируемый размер ОП

(ПР ОП), фактический размер ОП (ФР ОП), абсолютная погрешность прогноза (АПП) в сравнении с ФР ОП, относительная погрешность прогноза (ОПП), прогнозируемое число узлов (ПЧУ) и фактическое число узлов (ФЧУ) для двух моделей разной сложности приведены в таблице 4.3.

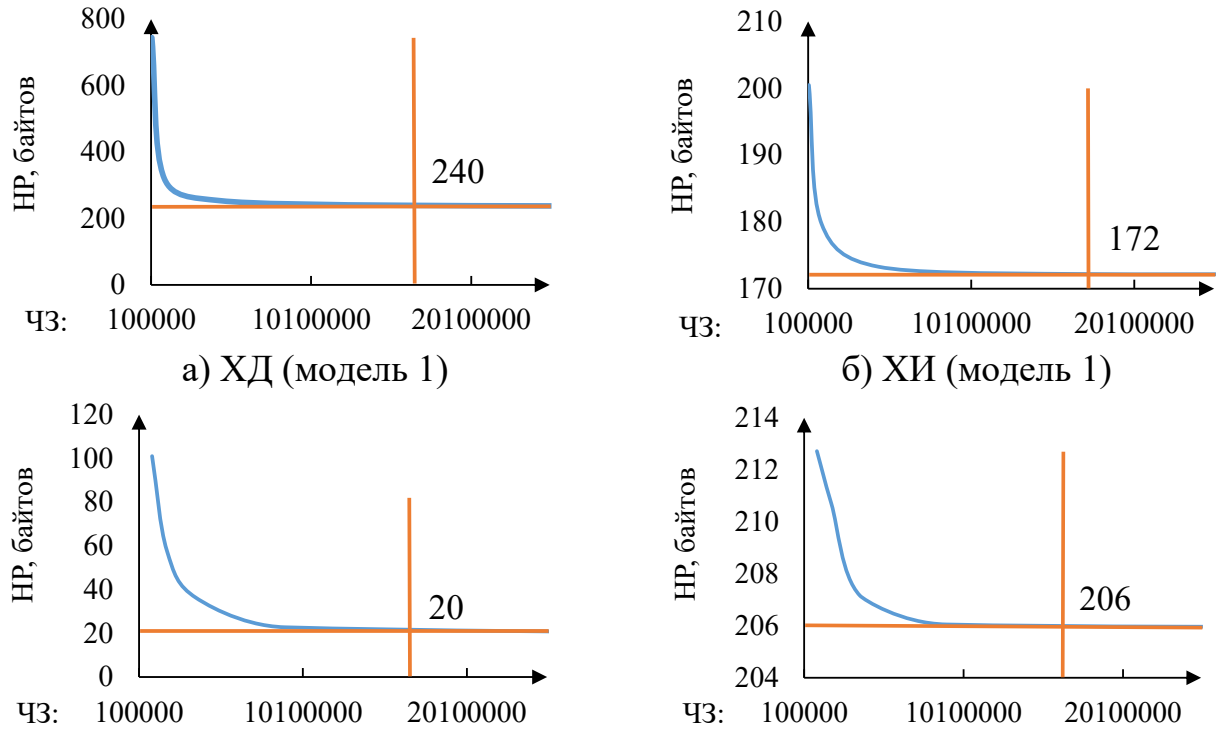


Рисунок 4.8 – Удельные расходы на хранение данных и индексов

Таблица 4.3 – Результаты экспериментов

M	k / m / n	Пул	ЧЗ	ПР ОП, ГБ	ФР ОП, ГБ	АПП, ГБ	ОПП, %	ПЧУ	ФЧУ
M1	8 / 26 / 0	1	140512730	62,837	61,887	0,950	1,54	1	1
M2				119,036		57,149	92,34	2	
M3				11,357		-50,530	81,65	1	
M1	8 / 26 / 0	2	140512730	62,837	61,887	0,950	1,54	1	1
M2				119,036		57,149	92,34	1	
M3				11,357		-50,530	81,65	1	
M1	2 / 2190 / 1220	1	902411450	479,012	473,214	5,798	1,23	6	6
M2				621,846		148,632	31,41	7	
M3				376,964		-96,25	20,34	5	
M1	2 / 2190 / 1220	2	902411450	479,012	473,214	5,798	1,23	1	1
M2				621,846		148,632	31,41	2	
M3				376,964		-96,25	20,34	1	

Проведено сравнение трех методик прогнозирования требуемого размера ОП: предложенная методика М1, рассмотренная в [9], методика М2, представленная в [131], и методика М3 разработчиков Apache Ignite. Исходя из результатов экспериментов, очевидно, что методика М1 обеспечивает наименьшую ОПП, не превышающую 1,54% на всех тестах для обеих моделей разной сложности, в сравнении с методиками М2 и М3, показывающими ОПП не менее 31,41% и 20,34% соответственно. При этом М1 обеспечивает отклонение ПР ОП только в большую сторону. Методика М2 дает более завышенный прогноз, что может снижать эффективность использования вычислительных ресурсов. В тоже время методика М3 прогнозирует недостаточный размер ОП, что зачастую приводит к выделению недостаточного числа узлов и отказу вычислительного процесса без использования дополнительной виртуальной памяти на диске или к существенному увеличению времени вычислений (более чем в 10 раз) при использовании дисковой памяти. В частности, для решения задачи на модели ТЭК на пуле 1 по методике М3 прогнозируется недостаточное число узлов.

Для расчета накладных расходов на хранение данных и индексов в рамках методики М1 разработаны испытательные стенды СРП1 и СРП2, представленных соответственно на рисунках 4.9–4.10. Системные операции СРП выделены красным цветом. СРП1 включает следующие операции: запуск кластера Apache Ignite для создания БД (o_3); выполнение НРП (o_4); расчет накладных расходов на хранение данных (o_5); остановка кластера Apache Ignite (o_6). В операциях используются следующие параметры: $z_1 - z_5$, z_7 – параметры НРП, представленного на рисунке 4.7; z_8 – конфигурационный файл Apache Ignite, описывающий структуру БД с отключенными индексами для всех таблиц; z_9 – файл с результатами расчета накладных расходов на хранение данных; z_{10} – результат остановки кластера Apache Ignite. СРП2 включает следующие операции: o_3, o_4 и o_6 – операции, описанные выше; расчет накладных расходов на хранение индексов (o_7). В операциях используются следующие параметры: $z_1 - z_5$ и $z_7 - z_{10}$ – параметры, описанные выше; z_{11} – конфигурационный файл Apache Ignite,

описывающий структуру БД с включенными индексами для всех таблиц БД; z_{12} – файл с результатами расчета накладных расходов на хранение индексов.

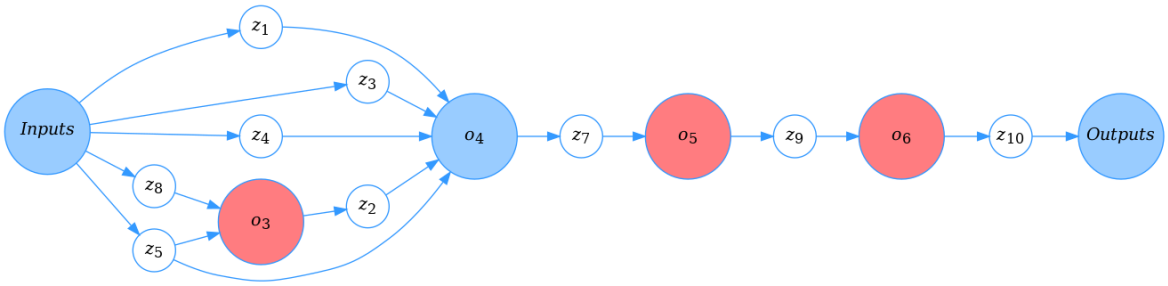


Рисунок 4.9 – СРП1

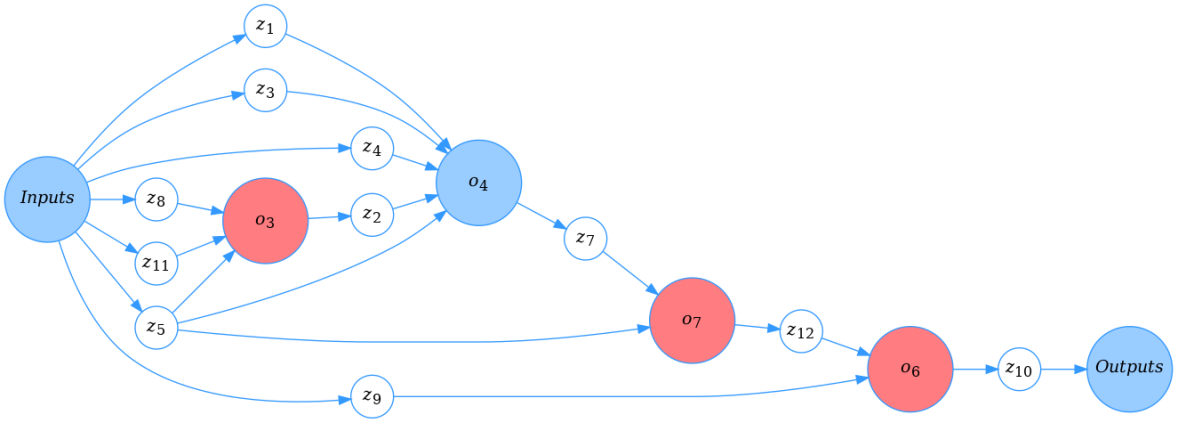


Рисунок 4.10 – СРП2

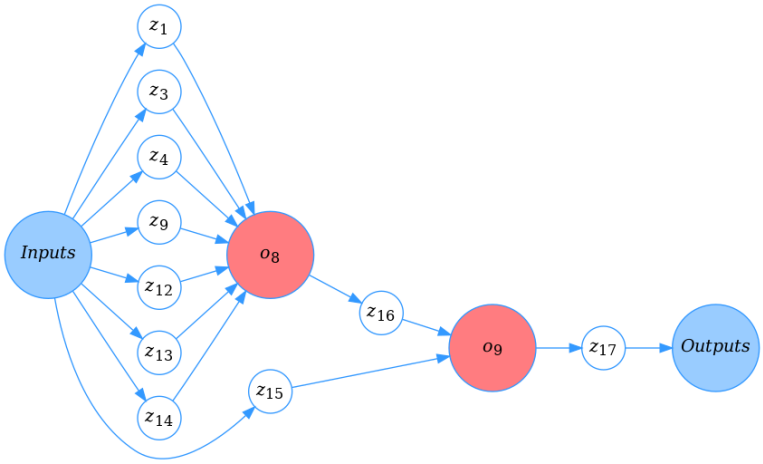


Рисунок 4.11 – СРП3

Для оценки требуемого размера ОП и формирования конфигурационного файла кластера Apache Ignite разработан СРП3 (рисунок 4.11). СРП3 включает следующие операции: расчет требуемого размера ОП (o_8); модификация шаблона конфигурационного файла (o_9). В операциях используются следующие параметры:

– z_1, z_2, z_9 и z_{11} – параметры, описанные выше;

– z_{13}– число узлов;

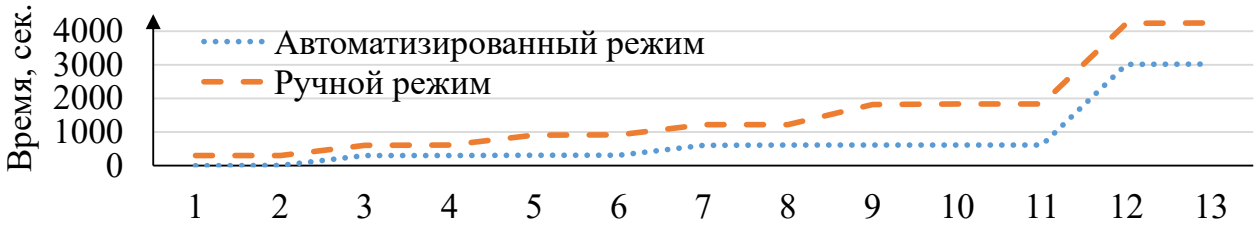
– z_{14}– число дуг;

– z_{15} – шаблон конфигурационного файла Apache Ignite;

– z_{16} – размер требуемого размера ОП;

– z_{17} – конфигурационный файл Apache Ignite.

На рисунке 4.12 приведено время развертывания кластера Apache Ignite. Очевидно, что временные затраты в автоматизированном режиме существенно ниже, чем в ручном.



Этапы: отключение индексов (1); запуск кластера (2, 6 и 11); тестирование (3 и 7); остановка кластера (4, 8 и 13); включение индексов (5); оценка требуемого размера ОП (9); выделение узлов кластера (10); выполнение НРП (12)

Рисунок 4.12 – Время развертывания кластера Apache Ignite

4.3. Предметно-ориентированная вычислительная среда для комплексной оценки критичности элементов энергетического комплекса

Разработано ПОВС для комплексной оценки критичности элементов ЭК [17-19, 21, 22]. В рамках ПОВС разработаны три приложения, реализующие следующие подходы: перебор множеств отказов; глобальный анализ чувствительности; анализ потоковых показателей важности дуг.

НРП приложений представлены сервисами, создаваемыми на основе стандарта WPS. Каждый из перечисленных подходов характеризуется разными требованиями к ресурсам и дополнительному ПО. Поэтому приложения, реализующие эти подходы, размещаются на разных ресурсах. Требования сервисов к ресурсам и среде представлены в таблице 4.4. Узлы ресурсов выделяются по запросу от сервисов, используемых при решении задач пользователями приложения или среды. Среда базируется на модели *Workflow-as-a-Service* [169]. В ней каждый ресурс представлен одним или несколькими сервисами, реализующими научные рабочие процессы приложений, размещенных на этих

ресурсах. Данная среда обеспечивает возможность комплексной оценки критичности элементов ЭК путем параллельного выполнения расчета разнородных показателей важности с помощью композиции wf_4 сервисов wf_1, wf_2 и wf_3.

Таблица 4.4 – Требования к ресурсам и среде*

Сервис	Требования к ресурсам		Требования к среде	
	Процессор	ОП	Дополнительное ПО	Испытательный стенд
wf_1	≥192 ядра	≥3 ТБ	IMDG-кластер	tb_1
wf_2	≥240 ядра	≥ 512 ГБ	Python-библиотека Salib	tb_2
wf_3	≥12 ядер	≥ 8 ГБ	–	–

* В соответствии с тестовыми оценками

СОП, реализующее полный перебор множества отказов, требует обработки большого массива данных за приемлемое время. Поэтому для решения этой задачи создано два WPS-сервиса. Первый сервис tb_1 представляет собой испытательный стенд для определения времени расчета одного сценария (рисунок 4.13). Второй сервис wf_1 реализует НРП, который генерирует сценарии возмущений, реализующие множества отказов заданного размера, в количестве V согласно формуле (6) и рассчитывает их последствия (рисунок 4.14). Для хранения и обработки информации о последствиях возмущений сервис wf_1 разворачивает РБД на основе технологий IMDG. Для ускорения обработки информации РБД создается только в ОП вычислительных узлов IMDG-кластера.

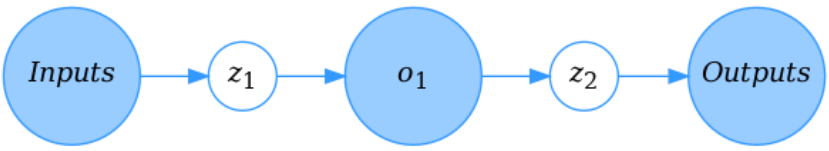


Рисунок 4.13 – Сервис tb_1

Сервис tb_1 включает одну операцию o_1, которая выполняет расчет времени выполнения одного сценария. В операции o_1 используются параметры z_1 и z_2, представляющие собой соответственно конфигурацию ЭК и время расчета одного сценария.

Сервис wf_1 реализует НРП для полного перебора множеств отказов размера k (рисунок 4.14). Он включает следующие операции:

- расчет числа сценариев (o_2);
- расчет размера требуемой ОП для определения числа узлов кластера Apache Ignite (o_3);
- определение необходимого числа узлов кластера Apache Ignite и создание его конфигурационного файла на основе шаблона (o_4);
- распределение множества обрабатываемых сценариев по узлам ресурса с целью балансировки вычислительной нагрузки и минимизации времени решения задачи с учетом времени выполнения одного сценария, определенного с помощью tb_1 (o_5);
- запуск кластера Apache Ignite для создания РБД (o_6);
- заполнение РБД данными о последствиях возмущений (o_7);
- обработка РБД средствами Apache Ignite для оценки критичности элементов сети (o_8);
- остановка кластера Apache Ignite (o_9).

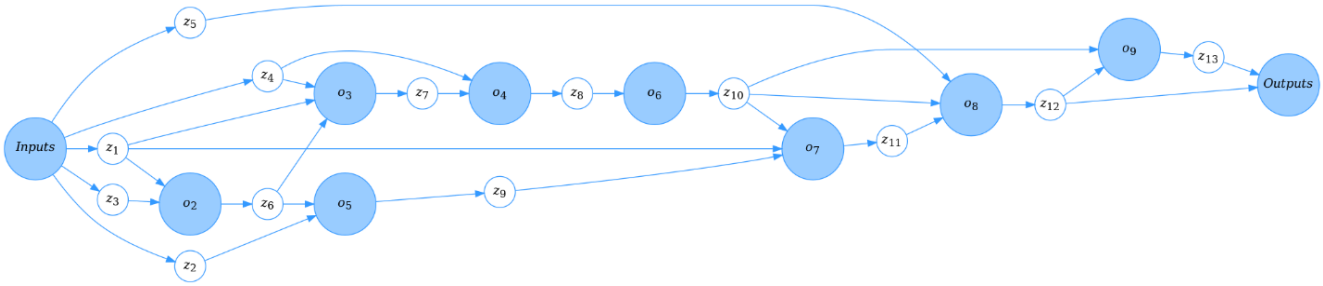


Рисунок 4.14 – Сервис wf_1

В операциях используются следующие параметры:

- z_3 – размер множества отказов;
- z_4 – шаблон конфигурационного файл Apache Ignite;
- z_5 – набор SQL-скриптов для оценки критичности элементов сети;
- z_6 – число сценариев;
- z_7 – общий размер ОП для развертывания РБД Apache Ignite;

- z_8 – конфигурационный файл Apache Ignite, описывающий структуру РБД;
- z_9 – число сценариев возмущений для каждого узла;
- z_{10} – адреса узлов кластера Apache Ignite;
- z_{11} – результат заполнения РБД;
- z_{12} – результат оценки критичности элементов сети;
- z_{13} – результат остановки кластера Apache Ignite.

Сервис-ориентированное приложение, реализующее GVA, построено на базе Python-библиотеки SALib [170]. SALib требует определить список входных факторов, затем генерирует их вариации. Рассчитанные на основе вариаций значения выходных факторов передаются одному из методов глобального анализа чувствительности: Соболя [171] или Delta [172]. В данном случае в качестве входных и выходных факторов выступают переменные модели ЭК. Результаты расчетов вариаций входных факторов на модели ЭК сохраняются в промежуточной БД.

Данное приложение включает два WPS-сервиса – wf_2 и tb_2. Сервис wf_2 реализует НРП для глобального анализа чувствительности (рисунок 4.15). Он включает следующие операции:

- генерация списка входных факторов (o_{10});
- генерация вариаций входных факторов (o_{11});
- создание промежуточной БД (o_{12});
- расчет вариаций входных факторов на модели ЭК (o_{13});
- расчет значений выходных факторов (o_{14});
- проведение глобального анализа чувствительности заданным методом (o_{15}).

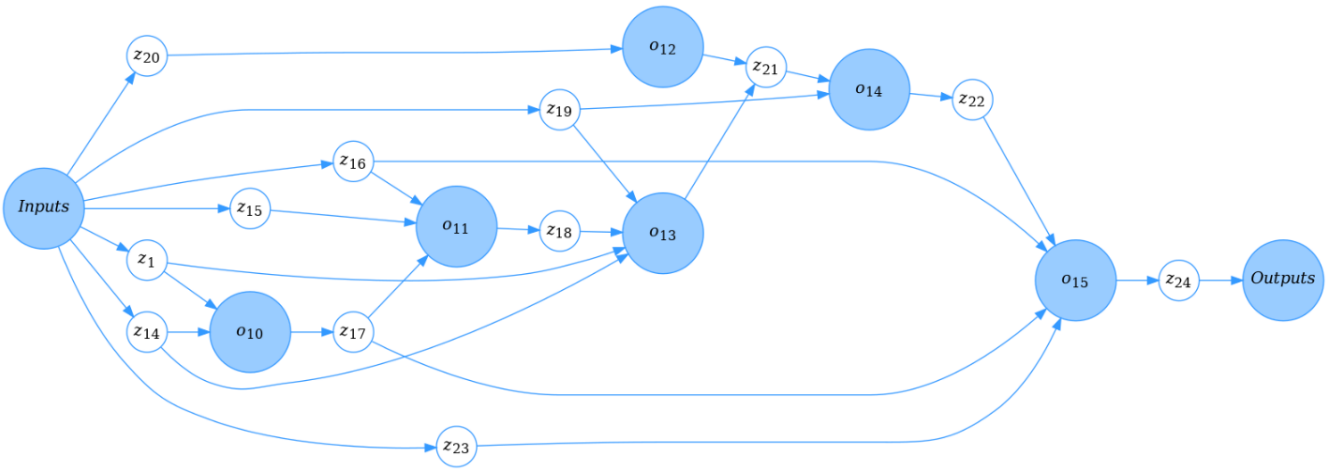


Рисунок 4.15 – Сервис wf_2

В операциях используются следующие параметры:

- z_{14} – файл для хранения настроек решателя задачи;
- z_{15} – файл с числом вариаций входных факторов;
- z_{16}– файл со значением глубины анализа чувствительности;
- z_{17} – файл со списком входных факторов;
- z_{18} – файл, содержащий вариации входных факторов;
- z_{19}– конфигурационный файл промежуточной БД;
- z_{20} – структура промежуточной БД;
- z_{21} – промежуточная БД;
- z_{22} – файл со значениями выходных факторов;
- z_{23} – название метода глобального анализа чувствительности;
- z_{24} – файл результатов глобального анализа чувствительности.

Сервис tb_2 представляет собой испытательный стенд для предварительного определения минимального числа вариаций входных факторов, необходимого для достижения устойчивости глобального анализа чувствительности, выполняемого с помощью сервиса wf_2 (рисунок 4.16). Он включает одну операцию o_{16}, экземпляры которой вызывают рабочий процесс wf_2 для каждого элемента параллельного списка z_{25} (списка файлов с разным числом вариаций входных факторов). Результатом работы является параллельный список z_{26}, элементы которого представляют собой результаты глобального анализа чувствительности z_{24}.

Эксперт анализирует полученные результаты и определяет достаточное число вариаций входных факторов, которые затем задаются в качестве параметра z_{15} для финального расчета.

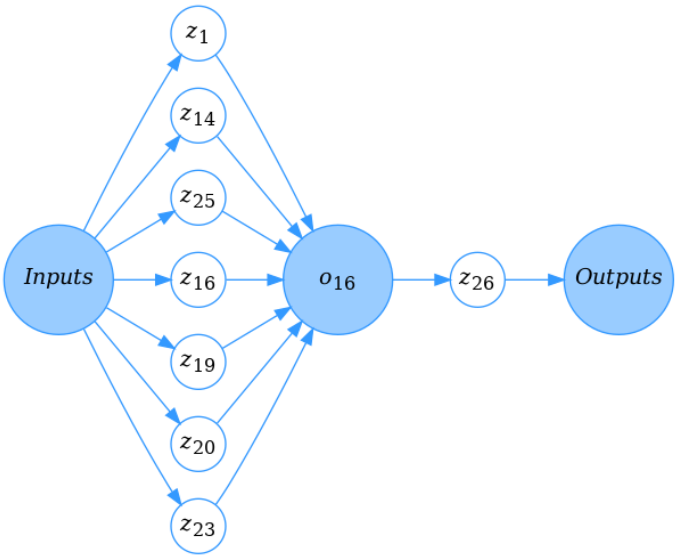


Рисунок 4.16 – Сервис tb_2

Сервис wf_3 реализует НРП для расчета потоковых показателей важности дуг (рисунок 4.17). Он включает одну операцию o_{17}. Данная операция рассчитывает значения показателей для заданной конфигурации ЭК z_1 и записывает их в файл z_{27}. Параметр z_{28} – результат выполнения сервиса wf_3.

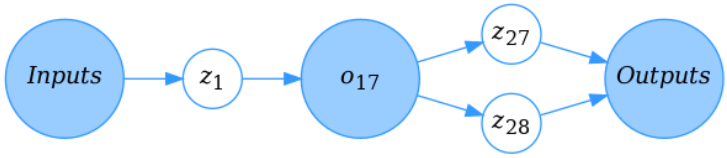


Рисунок 4.17 – Сервис wf_3

На основе рассмотренных подходов сформирована архитектура вычислительной среды (рисунок 4.18), которая позволяет получить комплексную оценку критичности элементов ЭК путем параллельного выполнения расчета разнородных показателей важности с помощью композиции wf_4 сервисов wf_1, wf_2 и wf_3.

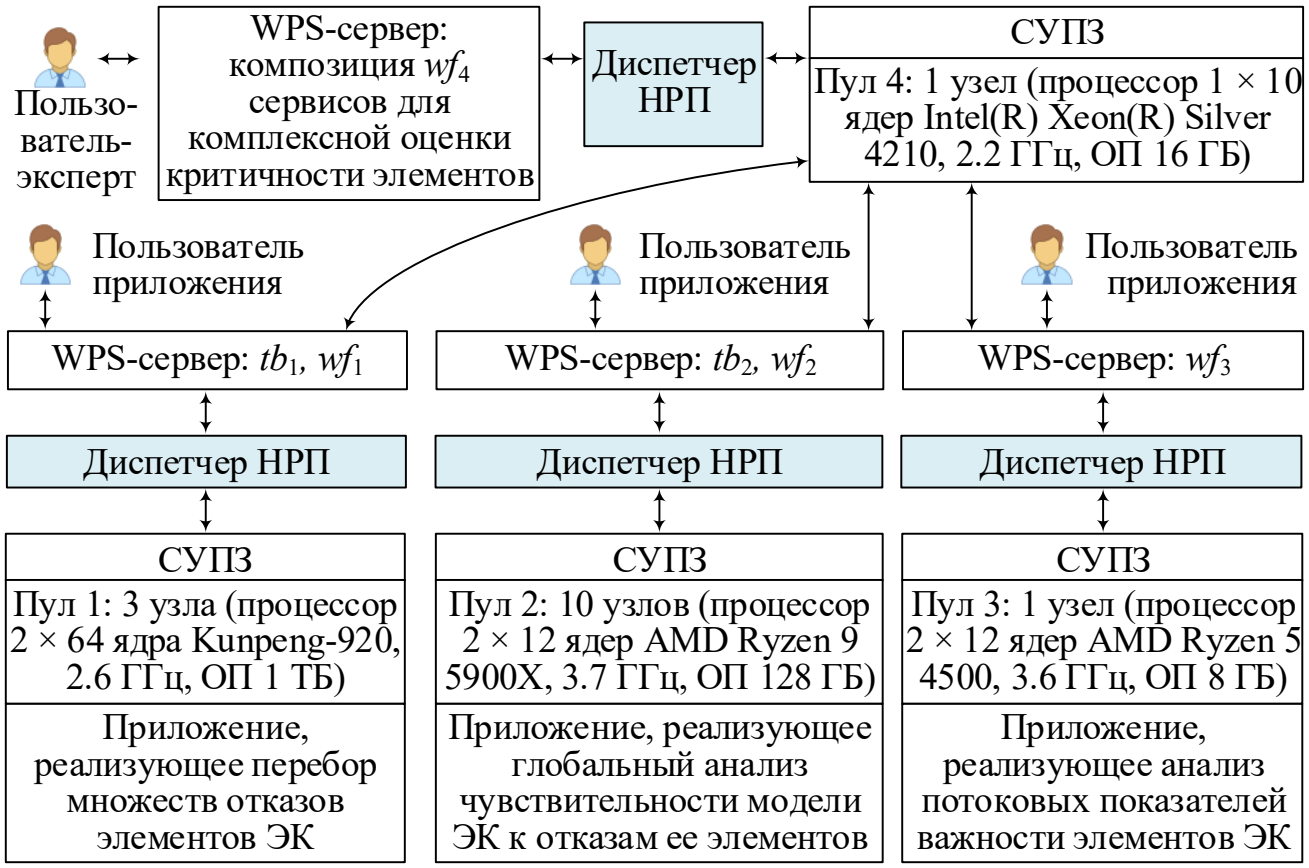


Рисунок 4.18 – Архитектура вычислительной среды

Приложение, в котором формируется композиция wf_4 (рисунок 4.19), размещается на ресурсе пользователя-эксперта. В композиции сервисов используются следующие параметры:

- z_{29} – агрегированный параметр, включающий входные параметры $z_2 - z_5$ сервиса wf_1;
- z_{30} – агрегированный параметр, включающий входные параметры $z_{14} - z_{16}$ сервиса wf_2;
- z_{12} – результат работы сервиса wf_1;
- z_{24} – результат работы сервиса wf_2;
- z_{27} – результат работы сервиса wf_3;
- z_{31} – результат комплексной оценки критичности элементов ЭК.

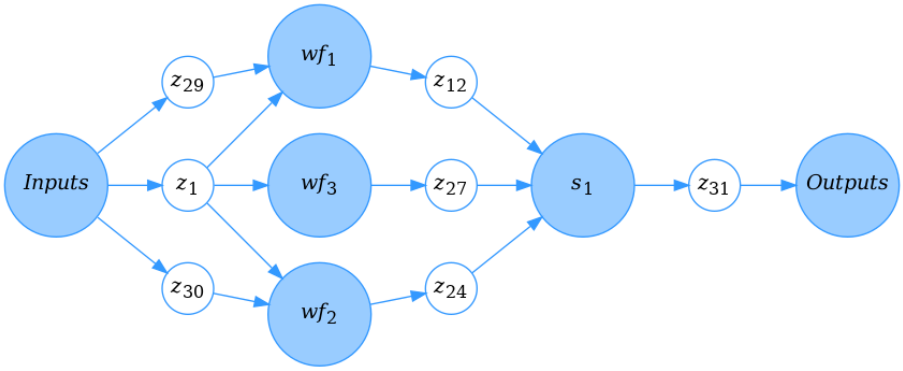


Рисунок 4.19 – Сервис wf_4

Среда использована для исследования газотранспортной системы, включающей 8 объектов добычи природного газа, 12 объектов потребления и 26 объектов транспорта. По результатам исследования было выделено две группы уязвимых элементов и одна группа узких мест с целью выработки рекомендаций экспертов по возможному развитию этих элементов.

Результаты экспериментов представлены в таблице 4.5. При выполнении сервиса wf_1 обработано 313911 сценариев возмущений. Средняя загрузка процессора ниже загрузки процессора при выполнении tb_1. Это объясняется большей долей последовательных вычислений в НРП, реализуемом сервисом wf_1. Предварительное определение среднего времени обработки одного сценария с помощью tb_1 обеспечило возможность балансировки нагрузки в узлах вычислительного ресурса и запроса на выделение рационального времени их использования. Экспериментальный анализ работы сервиса wf_1 в целом приведен в работе [173]. В частности, проанализирована возможность масштабируемости вычислений при росте числа узлов в кластере Apache Ignite и увеличении размера множеством отказов.

Предварительные расчеты минимально необходимого числа вариаций входных факторов на испытательном стенде tb_2 позволили существенно сократить затраты на вычисления с помощью сервиса wf_2 как в рамках данного исследования, так и при решении последующих новых задач. При выполнении tb_2 и wf_2 достигнута одинаковая средняя загрузка процессора, т. к. они реализуют один и тот же НРП.

С помощью сервиса wf_3 проведен расчет потоковых показателей важности дуг графа исследуемого ЭК. Обработано 2 854 879 сценариев возмущений. Данный расчет потребовал максимального времени вычислений в сравнении с сервисами wf_1 и wf_2. При этом достигнута максимальная средняя загрузка процессора, равная 96%.

При выполнении сервиса wf_4 учитывалось только время запуска сервисов wf_1, wf_2 и wf_3, периодической проверки статуса их выполнения и получения ссылок на результаты расчетов. Загрузка процессора при этом составила 63%.

Таблица 4.5 – Результаты экспериментального анализа

Вычислительные ресурсы	Сервис	Число вариантов	Время выполнения, ч	Загрузка процессора
3 узла (процессор 2 × 64 ядра Kunpeng-920, 2,6 ГГц, ОП 1 ТБ)	tb_1	1000	0,001	76%
	wf_1	313 911	0,327	74%
10 узлов (процессор 2 × 12 ядер AMD Ryzen 9 5900X, 3,7 ГГц, ОП 128 ГБ)	tb_2	55 296	23,875	95%
	wf_2	8 192	3,537	95%
1 узел (процессор 2 × 12 ядер AMD Ryzen 5 4500, 3,6 ГГц, ОП 8 ГБ)	wf_3	2 854 879	6,021	96%
1 узел (процессор 1 × 10 ядер Intel(R) Xeon(R) Silver 4210, 2,2 ГГц, ОП 16 ГБ)	wf_4	1	0,010	63%

4.4. Приложение для оценки времени выполнения известных научных рабочих процессов

Для демонстрации сокращения времени выполнения вычислений в ПОВС с помощью разработанных средств были выбраны два известных НРП – Epigenomics, CyberShake и LIGO [174], структура которых приведена на рисунке 4.20. В рамках эксперимента реализованы три схемы выполнения НРП [17, 24] под управлением HTCondor:

1) с централизованной передачей данных;

2) с помощью механизма прямой передачи данных;
3) с учетом структуры и использованием механизма прямой передачи данных.

Вычислительная среда состоит из четырех узлов со следующими характеристиками: 1 CPU, 2 ГБ памяти, жесткий диск размером 30 ГБ, пропускная способность сети – 5 Гбит/с. Расписания времени выполнения модулей Epigenomics, CyberShake и LIGO приведены соответственно на рисунке 4.21. Очевидно, что при переходе от схемы 1 к схемам 2 и 3 расписание выполнения НРП становится все более плотным, задержки на передачу данных становятся короче. Диаграмма времени выполнения Epigenomics, CyberShake и LIGO для трех схем дана на рисунке 4.22.

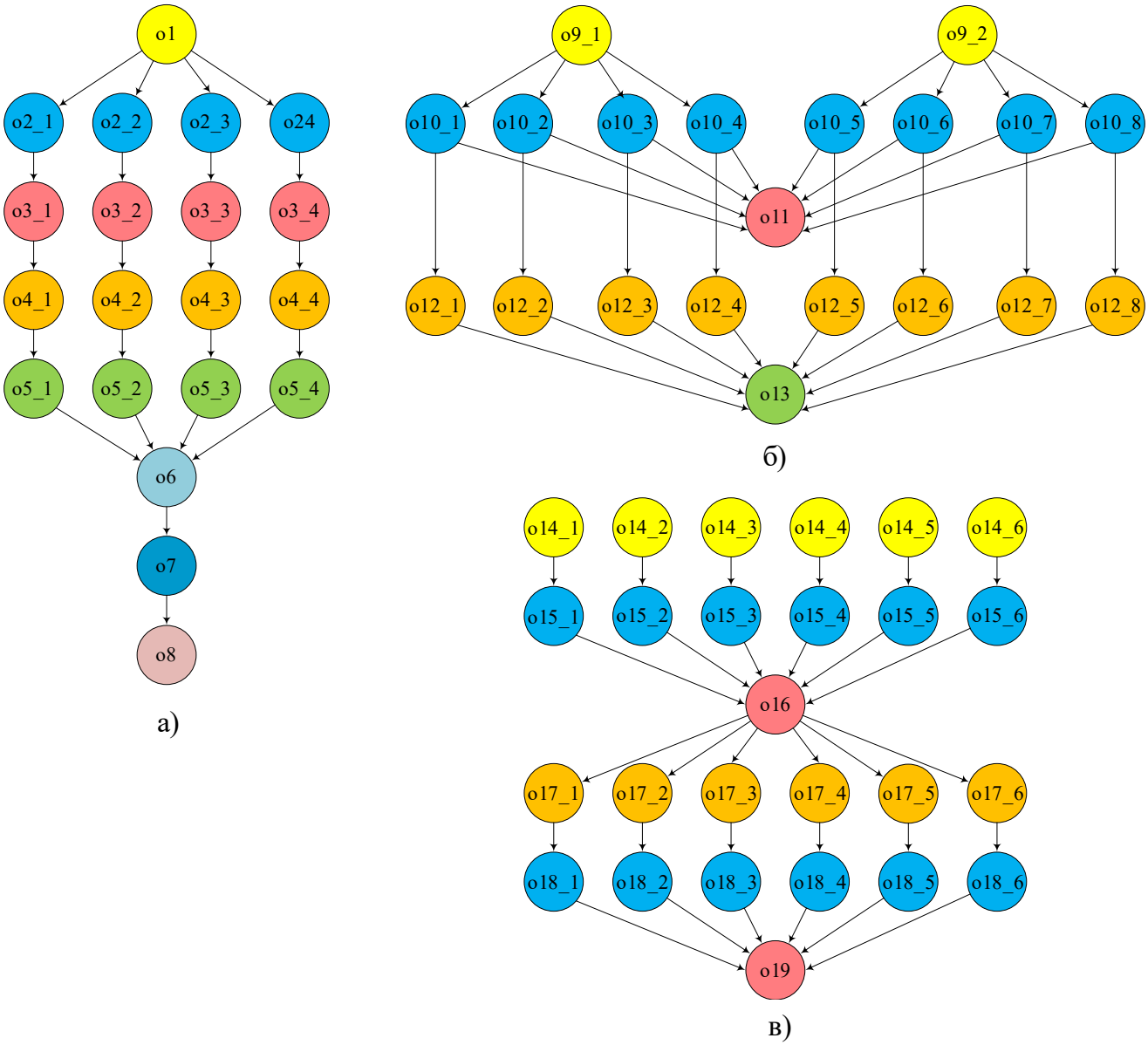


Рисунок 4.20 – НРП: Epigenomics (а), CyberShake (б) и Ligo (в)

В случае выполнения Epigenomics схемы 2 и 3 показали преимущество по сокращению времени вычислений в сравнении со схемой 1 более чем на 32% и 40% соответственно. Для CyberShake это преимущество оказалось еще существеннее. Схемы 2 и 3 позволили сократить время вычислений в сравнении со схемой 1 более чем на 47% и 53% соответственно. Для LIGO схемы 2 и 3 позволили сократить время вычислений в сравнении со схемой 1 на 14% и 21% соответственно.

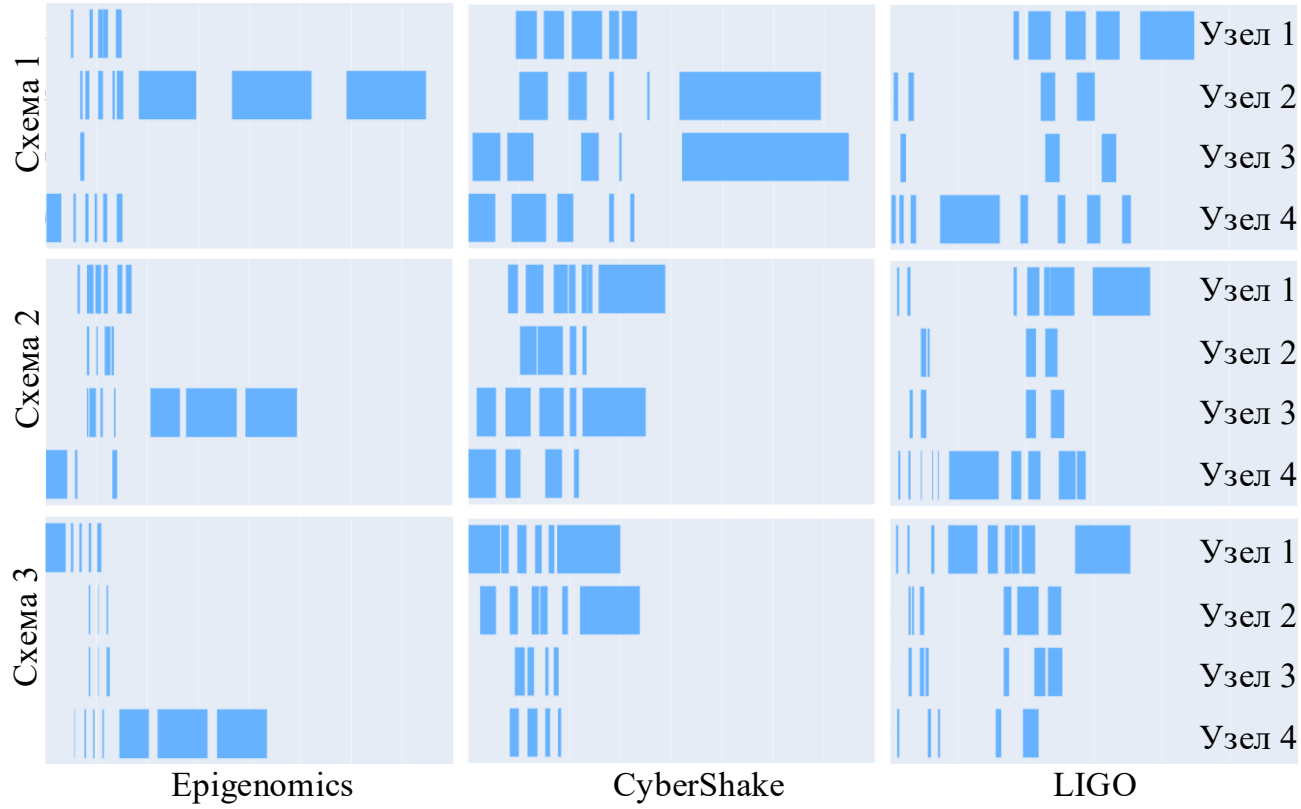


Рисунок 4.21 – Расписания выполнения заданий

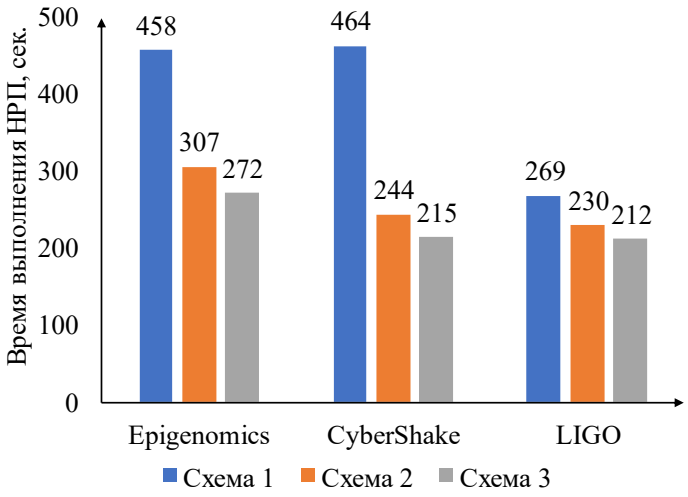


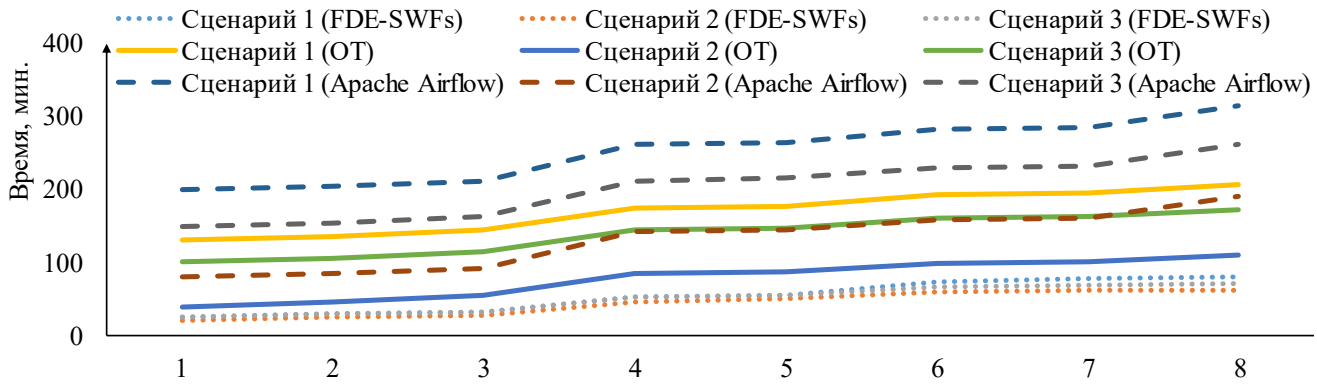
Рисунок 4.22 – Диаграмма времени выполнения НРП

4.5. Сравнение временных затрат на разработку и применение тестового приложения

Проведено сравнение временных затрат на разработку и применение тестового приложения с помощью ИК FDE-SWFs и ОТ – инструментария, являющегося предшественником данного ИК [10, 15]. Дополнительно оценены накладные расходы при подготовке и проведении экспериментов с использованием популярной в настоящее время системой Apache Airflow.

На рисунке 4.23 приведен график времени подготовки и проведения экспериментов для трех сценариев (см. раздел 3.4) выполнения НРП поэтапно с накоплением итога. Были учтены следующие этапы разработки и применения приложения для одной из задач исследования процессов функционирования и свойств ЭК:

1) описание вычислительной модели;
2) конструирование НРП по процедурной постановке задачи;
3) конструирование НРП по непроцедурной постановке задачи;
4) конфигурирование ресурсов ПОВС;
5) ввод исходных данных;
6) запуск и выполнение НРП;
7) получение результатов вычислений;
8) визуализация результатов вычислений.



Этапы: описание модели (1); конструирование НРП по процедурной (2) и непроцедурной (3) постановкам задач; конфигурирование ресурсов (4); ввод данных (5); выполнение НРП (6); получение (7) и визуализация (8) результатов

Рисунок 4.23 – Время подготовки и проведения экспериментов

При работе с сервисами в Apache Airflow дополнительно учитывались временные затраты на их создание в ручном режиме. Визуализация результатов вычислений в ОТ не поддерживается, поэтому выполнение данного этапа с его помощью осуществлялось в ручном режиме. В FDE-SWFs сервисы для выполнения модулей прикладного ПО создаются автоматически. Сравнение временных затрат показывает, что накладные расходы на выполнение каждого этапа с помощью FDE-SWFs существенно ниже, чем при использовании Apache Airflow. FDE-SWFs также превосходит ОТ по сокращению общего времени подготовки и проведения экспериментов.

4.6. Сравнительный анализ функциональных возможностей

В рамках диссертации ИК FDE-SWFs использован в качестве основного средства построения ПОВС и разработки приложений. Сравнительный анализ поддержки ключевых функциональных возможностей в рамках данного комплекса и систем подобного назначения [11, 22, 30] показал, что он наиболее полно предоставляет возможности по построению НРП, поддержке управляющих конструкций, созданию сервис-ориентированных НРП с поддержкой WPS и их представлению на BPEL (см. табл. 4.6). FDE-SWFs также максимально поддерживает требуемые уровни параллелизма, обеспечивает формирование среды с использованием ресурсов суперкомпьютерных центров, Grid-систем и облачных платформ, а также в отличие от большинства СУРП автоматизирует интеграцию и контейнеризацию ПО, предоставляет возможность применения технологии IMDG, существенно ускоряющей обработку данных (табл. 4.7). Дополнительно FDE-SWFs обеспечивает возможность организации испытательных стендов и тестирования на них СОП и их отдельных компонентов.

Таблица 4.6 – Поддержки возможностей построения НРП

СУРП	Поддержка						
	BPEL	ветвления / циклы	системные операции	НРП: абстрактный / конкретный	постановки задач: процедурная / непроцедурная	сервисы	WPS
UNICORE	–	+ / +	+	– / +	+ / –	+	–
DAGMan	–	– / –	+	+ / –	+ / –	–	–
Pegasus	–	– / –	+	+ / –	+ / –	–	–
Apache Airflow	–	– / –	–	– / +	+ / –	–	–
HyperFlow	–	– / –	+	+ / +	+ / –	+	–
WaaS Cloud Platform	–	– / –	–	+ / –	+ / –	+	–
Galaxy	–	– / +	+	+ / +	+ / –	+	–
BPEL Designer Project	+	+ / +	–	+ / +	+ / –	+	+
OT	–	+ / +	+	+ / +	+ / +	+	–
FDE-SWFs	+	+ / +	+	+ / +	+ / +	+	+

Таблица 4.7 – Поддержки возможностей выполнения НРП

СУРП	Поддержка					
	параллелизм: задача / данные / конвейер	среда: кластер / Grid / облако	IMDG	интеграция прикладного и системного ПО	контейнеризация прикладного и системного ПО	испытательные стенды
UNICORE	+ / + / –	– / + / –	–	–	–	–
DAGMan	+ / + / –	+ / + / +	–	–	–	–

Таблица 4.7 – Поддержки возможностей выполнения НРП (продолжение)

СУРП	Поддержка					
	параллелизм: задача / данные / конвейер	среда: кластер / Grid / облако	IMDG	интеграция прикладного и системного ПО	контейнеризация прикладного и системного ПО	испытательные стенды
Pegasus	+ / + / –	+ / + / +	–	–	–	–
Apache Airflow	+ / + / +	– / – / +	–	+	+	–
HyperFlow	+ / – / –	– / – / +	–	–	–	–
WaaS Cloud Platform	+ / – / –	– / + / +	–	–	–	–
Galaxy	+ / – / –	– / – / +	–	–	–	–
BPEL Designer Project	+ / – / –	+ / + / +	–	–	–	–
OT	+ / + / +	+ / + / +	+	+	–	–
FDE-SWFs	+ / + / +	+ / + / +	+	+	+	+

4.7. Выводы

В четвертой главе получены следующие основные результаты:

При активном участии соискателя в качестве системного разработчика разработаны приложения для решения задач структурно-параметрической оптимизации моделей ТЭК России и локального ЭК, глобального анализа уязвимости ЭК, комплексной оценки критичности элементов ЭК, а также подготовлены и проведены вычислительные эксперименты в ПОВС. Продемонстрированы преимущества разработанного ИК в сравнении с системами подобного назначения. Достигнуто существенное сокращение трудозатрат на этапах подготовки и проведения экспериментов.

Результаты исследований по данной главе опубликованы в [7, 9, 14, 17-24,

34]. В частности, разработка приложения для структурно-параметрической оптимизации ЭК рассмотрена в работах [7, 14, 17, 34]. Создание приложения для глобального анализа уязвимости ЭК представлено в работах [9, 17, 20, 23], Построение ПОВС для комплексной оценки критичности элементов описано в работах [17-19, 21, 22]. Оценка времени выполнения известных научных рабочих процессов на примере тестового приложения показана в работах [17, 24]. Сравнение временных затрат на разработку и применения тестового приложения рассмотрено в работах [10, 15]. Анализ функциональных возможностей разработанного ИК в сравнении с известными системами подобного назначения приведен в работах [11, 22, 30].

Заключение

В целом решение поставленных в диссертации задач, включая реализацию предложенной модели, а также разработанных алгоритмов и инструментальных средств, обеспечивает следующие преимущества при создании и применении СОП в ПОВС по сравнению с известными разработками:

- расширение совокупности знаний вычислительной модели, необходимых для создания и применения СОП в ПОВС, знаниями о сущностях и процессах интеграции, тестирования и контейнеризации ПСПО;
- построение, трансформацию и тестирование программ и программных систем, их взаимодействие между собой и другими программными комплексами, а также параллельную и распределенную обработку данных;
- учет особенностей предметной области решаемых задач (выявление ключевых параметров, влияющих на увеличение необходимых ресурсов при обработке данных, определение предпочтительных методов решения задач путем их тестирования на испытательных стендах, разработка приложений в ориентации на ресурсы, соответствующие сложности решаемых задач);
- улучшение базовых критериев пользователей и владельцев ресурсов при создании и применении приложений для решения ресурсоемких задач;
- автоматизацию развертывания кластера IMDG, сокращающую трудозатраты, и оценку требуемых для него ресурсов с высокой точностью;
- сокращение времени на подготовку и проведение экспериментов в целом.

Предложенные в диссертации комплексные технологические решения по организации ПОВС, основанные на разработанных моделях, алгоритмах и инструментальных средствах, допускают свое естественное развитие применительно к другим ГРВС различного назначения.

Литература

1. Zhang, H. In-memory big data management and processing: A survey / H. Zhang, G. Chen, B.C. Ooi // IEEE Transactions on Knowledge and Data Engineering. — 2015. — Vol. 27, № 7. — P. 1920–1948.

2. Schut, P. OpenGIS®Web Processing Service / P. Schut // Open Geospatial Consortium. — 2007. — 88 p.

3. Hossain, M.M. Extensibility Challenges of Scientific Workflow Management Systems / M.M. Hossain, B. Roy, C. Roy et al. // Proc. of the Intern. Conf. on Human-Computer Interaction. — Cham: Springer Nature Switzerland, 2023. — P. 51–70.

4. The evolution of distributed computing systems: from fundamental to new frontiers / D. Lindsay et al. // Computing. — 2021. — Vol. 103, № 8. — P. 1859–1878.

5. Padovano, R. Critical Analysis of Parallel and Distributed Computing and Future Research Direction of Cloud Computing / R. Padovano // Journal of Computing and Natural Science. — 2021. — Vol. 1, № 4. — P. 114–120.

6. Cheng, X. Meta computing / X. Cheng, M. Xu, R. Pan et al. // IEEE Network. — 2023. — Early Access. — Vol. 38, № 2. — P. 225–231.

7. Optimization of Integrated Energy System Resilience / I. Bychkov, A. Feoktistov, M. Voskoboinikov et al. // Информатика и автоматизация. 2025. — Vol. 24, № 3 — P. 951–981.

8. Бычков, И.В. Разработка сервис-ориентированного доступа к высокопроизводительной вычислительной среде на основе стандарта WPS / Бычков И.В., М.Л. Воскобойников, Я.А. Еделев, А.Г. Феоктистов // Вычислительные технологии — 2025. — Т. 30, № 2 — С. 87–99.

9. Воскобойников, М.Л. Использование технологии In-Memory Data Grid при анализе живучести энергетических инфраструктур / М.Л. Воскобойников, А.Г. Феоктистов // Информационные и математические технологии в науке и управлении — 2025. — № 3(39). — С. 154–163.

10. Воскобойников, М.Л. Разработка и применение сервис-ориентированных научных приложений в инструментальном комплексе FDE-SWFs /

М.Л. Воскобойников, А.Г. Феоктистов, А.Н. Черных // Труды ИСП РАН. — 2024. — Т. 35, № 6. — С. 195–214.

11. Воскобойников, М.Л. Сравнительный анализ систем управления научными рабочими процессами / М.Л. Воскобойников, А.Г. Феоктистов // Информационные и математические технологии в науке и управлении. — 2024. — № 3(35). — С. 102–111.

12. Воскобойников, М.Л. Диспетчер научных рабочих процессов: средства визуализации данных / М.Л. Воскобойников // Современные наукоемкие технологии. — 2024. — № 1. — С. 16–21.

13. Организация вычислительной среды разработки и применения научных рабочих процессов на основе контейнеризации / А.Г. Феоктистов, Р.О. Костромин, М.Л. Воскобойников и др. // Вычислительные технологии. — 2023. — Т. 28, № 6. — С. 151–164.

14. Scientific Workflow-Based Synthesis of Optimal Microgrid Configurations / O. Edeleva, A. Edelev, M. Voskoboinikov et al. // Energies. — 2024. — Vol. 17, № 23 — P. 6138.

15. Voskoboinikov, M. Framework for Development and Execution of Scientific WorkFlows: Designing Service-oriented Applications / M. Voskoboinikov, A. Feoktistov, A. Tchernykh // Programming and computer software. — 2024. — Vol. 49, № 8. — P. 897–910.

16. Kostromin, R. Service-Oriented Tools for Automating Digital Twin Development / R. Kostromin, A. Feoktistov, M. Voskoboinikov // Proc. of the 4th Scientific-practical Workshop on Information Technologies: Algorithms, Models, Systems (ITAMS 2021). — CEUR-WS Proceedings. — 2021. — Vol. 2984. — P. 95–100.

17. Воскобойников, М.Л. Использование инструментального комплекса FDE-SWFs для разработки и применения научных приложений / М.Л. Воскобойников, А.Г. Феоктистов, А.В. Еделев // Ляпуновские чтения: Материалы международной конференции (г. Иркутск, Россия, 27 октября - 1 ноября 2025 г.). — Иркутск: Изд-во ИДСТУ СО РАН, 2025. — С. 93–94.

18. Сервис-ориентированная среда для исследования живучести систем энергетики / А.В. Еделев, И.В. Бычков, А.Г. Феоктистов, М.Л. Воскобойников // Международная конференция «Системные исследования в энергетике – 2025» и 10-е Мелентьевские чтения, посвященные 65-летию ИСЭМ СО РАН и 40-летию ИНЭИ РАН: материалы конференции. г. Иркутск, Россия. — Иркутск: Изд-во ИСЭМ СО РАН, 2025. — С. 204-206.

19. Автоматизация создания и применения сервис-ориентированной среды исследования живучести систем энергетики / И.В. Бычков, А.В. Еделев, А.Г. Феоктистов, М.Л. Воскобойников // Материалы XVIII Всероссийской мультиконференции по проблемам управления (МКПУ-2025), 15-20 сентября 2025 г., г. Тула, Россия: в 4 т. — Тула: Изд-во ТулГУ, 2025. — Т. 2. Управление в распределенных и сетевых системах. — С. 318–321.

20. Применение технологии In-Memory Data Grid в исследовании живучести энергетических систем / А.Г. Феоктистов, М.Л. Воскобойников, И.В. Бычков и др. // Материалы VIII Международной научно-практической конференции «Информационные технологии и высокопроизводительные вычисления», 15-17 сентября 2025 г., Хабаровск, Россия. — Хабаровск: ХФИЦ ДВО РАН, 2025. — С. 289-292.

21. Моделирование природно-технических систем: модели, алгоритмы, сервисы и приложения / И.В. Бычков, А.Г. Феоктистов, М.Л. Воскобойников и др. // Материалы Выездного совещания участников проекта «Фундаментальные исследования Байкальской природной территории на основе системы взаимосвязанных базовых методов, моделей, нейронных сетей и цифровой платформы экологического мониторинга окружающей среды», 27 августа 2025 г., г. Белокуриха, Россия. — Иркутск: Изд-во ИДСТУ СО РАН, 2025. — С. 17–21.

22. Моделирование природно-технических систем: инструментальная поддержка / И.В. Бычков, А.Г. Феоктистов, М.Л. Воскобойников и др. // Материалы Выездного совещания участников проекта «Фундаментальные исследования Байкальской природной территории на основе системы взаимосвязанных базовых методов, моделей, нейронных сетей и цифровой платформы экологического

мониторинга окружающей среды», 27 августа 2025 г., г. Белокуриха, Россия. — Иркутск: Изд-во ИДСТУ СО РАН, 2025. — С. 22–26.

23. Воскобойников, М.Л. Применение In-Memory Data Grid в изучении энергосистем / М.Л. Воскобойников, А.Г. Феоктистов, А.В. Еделев // Тезисы XXX Байкальской Всероссийской конференции с международным участием «Информационные и математические технологии в науке и управлении», 1-9 июля 2025, г. Иркутск, Россия. — Иркутск: Издательский центр ИСЭМ СО РАН, 2025. — С. 19.

24. Воскобойников, М. Оценка эффективности управления вычислениями диспетчером научных рабочих процессов / М.Л. Воскобойников, А.Г. Феоктистов // Сборник научных трудов VII Международного семинара по информационным, вычислительным и управляющим системам для распределенных сред (ICCS-DE 2025), 7-11 июля 2025 г., г. Иркутск, Россия. — Иркутск: Изд-во ИДСТУ СО РАН, 2025. — С. 74–82.

25. Воскобойников, М.Л. Диспетчер научных рабочих процессов / М.Л. Воскобойников, А.Г. Феоктистов // Ляпуновские чтения: Материалы международной конференции (г. Иркутск, Россия, 2-6 декабря 2024 г.). — Иркутск: Изд-во ИДСТУ СО РАН, 2024. — С. 45–48.

26. Юмашев, Е.А. Сервис-ориентированное приложение получения, обработки и анализа метеорологических данных / Е.А. Юмашев, Д.Н. Карамов, М.Л. Воскобойников и др. // Ляпуновские чтения: Материалы международной конференции (г. Иркутск, Россия, 2-6 декабря 2024 г.). — Иркутск: Изд-во ИДСТУ СО РАН, 2024. — С. 248–251.

27. Воскобойников, М.Л. Разработка и применение сервис-ориентированных рабочих процессов в гетерогенной вычислительной среде / М.Л. Воскобойников, А.Г. Феоктистов // Суперкомпьютерные дни в России: Труды международной конференции, 23–24 сентября 2024 г., Москва, Россия. — М.: МАКС Пресс, 2024. — С. 243–245.

28. Воскобойников, М. Система управления сервис-ориентированными научными приложениями / М.Л. Воскобойников, А.Г. Феоктистов //

Информационные, вычислительные и управляющие системы для распределенные сред: практика и опыт. — Иркутск: Изд-во ИДСТУ СО РАН, 2024. — С. 4–8.

29. Инструментальные средства построения цифровых двойников информационно-вычислительных систем / А. Феоктистов, М. Воскобойников, Е. Юмашев и др. // Информационные, вычислительные и управляющие системы для распределенные сред: практика и опыт. — Иркутск: Изд-во ИДСТУ СО РАН, 2024. — С. 39–43.

30. Воскобойников, М.Л. Анализ систем управления рабочими процессами / М.Л. Воскобойников, А.Г. Феоктистов // Тезисы XXIX Байкальской Всероссийской конференции с международным участием «Информационные и математические технологии в науке и управлении», 28 июня – 7 июля, г. Иркутск, Россия. — Иркутск: Издательский центр ИСЭМ СО РАН, 2024. — С. 12.

31. Феоктистов, А. Методы и средства разработки и применения испытательного стенда сервис-ориентированных приложений / А. Феоктистов, М. Воскобойников, А. Еделев // Сборник научных трудов VI Международного семинара по информационным, вычислительным и управляющим системам для распределенных сред (ICCS-DE 2024), 1-5 июля 2024 г., г. Иркутск, Россия. — Иркутск: Изд-во ИДСТУ СО РАН, 2024. — С. 169–173.

32. Яковлев, Д. Средство мониторинга узлов системы управления сервис-ориентированными научными приложениями / Д. Яковлев, М. Воскобойников, Р. Костромин // Материалы VI Международного семинара по информационным, вычислительным и управляющим системам для распределенных сред (ICCS-DE 2024), 1-5 июля 2024 г., г. Иркутск, Россия. — Иркутск: Изд-во ИДСТУ СО РАН, 2024. — С. 220–222.

33. Danilov, G. Testbed-based approach to testing a library for evaluating network reliability algorithms / G. Danilov, M. Voskoboinikov // Proceedings of the International Workshop on Critical Infrastructures in the Digital World (IWCI-2024), March 14-20, 2024, Bolshoe Goloustno, Russia. — Irkutsk: ESI SB RAS, 2024. — P. 3–4.

34. Edeleva, O. Scientific Workflow-Based Synthesis of Microgrids / O. Edeleva, A. Edelev, M. Voskoboinikov et al. // Preprints. — 2024. — P. 2024110275.

35. Феоктистов, А.Г. Стенд тестирования сервис-ориентированных научных рабочих процессов / А.Г. Феоктистов, Р.О. Костромин, М.Л. Воскобойников и др. // Ляпуновские чтения: Материалы междунар. конф. (4-8 декабря 2023 г., г. Иркутск, Россия). — Иркутск: Изд-во ИДСТУ СО РАН, 2023. — С. 126.

36. Костромин, Р.О. Автоматизированное управление конфигурацией платформы имитационного моделирования инфраструктурных объектов / Р.О. Костромин, Г.М. Ружников, Е.С. Фереферов, И.А. Сидоров, М.Л. Воскобойников // Тезисы XXVII Байкальской Всероссийской конф. с междунар. участием «Информационные и математические технологии в науке и управлении» (29 июня – 8 июля, г. Иркутск, Россия). — Иркутск, Издательский центр ИСЭМ СО РАН, 2022. — С. 3.

37. Воскобойников, М.Л. Модуль прогнозирования размера базы данных в оперативной памяти узлов вычислительного кластера / М.Л. Воскобойников, А.Г. Феоктистов. Свидетельство о государственной регистрации программы для ЭВМ № 2025660056. — М.: Федеральная служба по интеллектуальной собственности (РОСПАТЕНТ), 2025.

38. Воскобойников, М.Л. Библиотека программных агентов организации распределенной базы данных в оперативной памяти / М.Л. Воскобойников, А.Г. Феоктистов. Свидетельство о государственной регистрации программы для ЭВМ № 2024668180. — М.: Федеральная служба по интеллектуальной собственности (РОСПАТЕНТ), 2024.

39. Воскобойников, М.Л. Планировщик схем решения задач для распределенных пакетов прикладных программ / М.Л. Воскобойников. Свидетельство о государственной регистрации программы для ЭВМ № 2023685789. — М.: Федеральная служба по интеллектуальной собственности (РОСПАТЕНТ), 2023.

40. Воеводин, В.В. Решение больших задач в распределенных вычислительных средах / В.В. Воеводин // Автоматика и телемеханика. — 2007. — № 5. — С. 32–45.

41. Foster, I. What is the Grid? A Three Point Checklist. [Электронный ресурс] / I. Foster. — 2002. — Режим доступа:

https://www.globus.org/sites/default/files/WhatIsTheGrid.pdf (дата обращения: 03.12.2025).

42. Chetty, M. Weaving Computational Grids: How Analogous Are They with Electrical Grids? / M. Chetty and R. Buyya // Computing in Science and Engineering. — 2002. — Vol. 4. — P. 61–71.

43. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener / R. Buyya, C. S. Yeo, S. Venugopal et al. // Computer Systems. — 2009. — Vol. 25, № 6. — P. 599–616.

44. The National Institute of Standards and Technology (NIST) Cloud Computing. [Электронный ресурс]. — Режим доступа: http://csrc.nist.gov/groups/SNS/cloud-computing/ (дата обращения: 03.12.2025).

45. Rani, D. A comparative study of SaaS, PaaS and IaaS in cloud computing / D. Rani, R.K. Ranjan // International Journal of Advanced Research in Computer Science and Software Engineering. — 2014. — Vol. 4, № 6. — P. 458–461.

46. Wang, J. Workflow as a service in the cloud: architecture and scheduling algorithms / J. Wang, P. Korambath, I. Altintas et al. // Procedia computer science. — 2014. — Vol. 29. — P. 546–556.

47. Vu, Q.H. Demods: A description model for data-as-a-service / Q.H. Vu, T.V. Pham, H.L. Truong et al. // Proc. of the 26th Intern. Conf. on Advanced Information Networking and Applications. — IEEE, 2012. — P. 605–612.

48. Feoktistov, A. Development of Distributed Subject–Oriented Applications for Cloud Computing through the Integration of Conceptual and Modular Programming / A. Feoktistov, R. Kostromin, I. Sidorov et al. // Proc. of the 41th Intern. Convention on information and communication technology, electronics and microelectronics (MIPRO–2018). — Riejka: IEEE, 2018. — P. 256–261.

49. Горбунов–Посадов, М.М. Системное обеспечение пакетов прикладных программ / М.М. Горбунов–Посадов, Д.А. Корягин, В.В. Мартынюк. — М.: Наука, 1990. — 208 с.

50. Krakowiak, S. Middleware Architecture with Patterns and Frameworks. [Электронный ресурс] / S. Krakowiak. — 2009. — Режим доступа: https://lig-membres.imag.fr/krakowia/Files/MW-Book/main.pdf (дата обращения: 03.12.2025).

51. Schäl, T. Workflow management systems for process organisations / T. Schäl // Lecture Notes in Computer Science. — 1996. — Vol. 1096. — 208 p.

52. UNICORE. [Электронный ресурс]. Режим доступа: https://www.unicore.eu/ (дата обращения: 29.11.2024).

53. DAGMan. [Электронный ресурс]. Режим доступа: https://htcondor.org/dagman/dagman.html (дата обращения: 03.12.2025).

54. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems / E. Deelman, G. Singh, M.H. Su et al. // Scientific Programming. — 2005. — Vol. 13. — P. 219–237.

55. Balis, B. HyperFlow: A model of computation, programming approach and enactment engine for complex distributed workflows / B. Balis // Future Generation Computer Systems. — 2016. — Vol. 55. — P. 147–162.

56. Hilman, M.H. Workflow-as-a-service cloud platform and deployment of bioinformatics workflow applications / M.H. Hilman, M.A. Rodriguez, R. Buyya // Knowledge Management in the Development of Data-Intensive Systems. I. Mistrik, M. Galster, B. Maxim, B. Tekiner-dogan, Eds. — Boca Raton, FL, USA: CRC Press, 2021. — P. 205–226.

57. Galaxy: A comprehensive approach for supporting accessible, reproduci-ble, and transparent computational research in the life sciences / J. Goecks, A. Nekrutenko, J. Taylor et al. // Genome biology. — 2010. — Vol. 11. — P. R86.

58. Singh, P. Airflow / P. Singh // Singh P. Learn PySpark: Build Python-based Machine Learning and Deep Learning Models. — Apress, 2019. — P. 67–84.

59. Collaborative Development and Use of Scientific Applications in Orlando Tools: Integration, Delivery, and Deployment / A. Feoktistov, S. Gorsky, I. Sidorov et al. // Communications in Computer and Information Science. — 2020. — Vol. 1087. — P. 18–32.

60. Yue, P. A geoprocessing workflow system for environmental monitoring and integrated modelling / P. Yue, M. Zhang, Z. Tan // Environmental Modelling and Software. — 2015. — Vol. 69. — P. 128–140.

61. Geoscience model service integrated workflow for rain-storm waterlogging analysis / X. Tan, J. Jiao, N. Chen et al. // International Journal of Digital Earth. — 2021. — Vol. 14. — P. 851–873.

62. Papazoglou M. Web Services: Principles and Technology / M. Papazoglou. — New York: Pearson Education, 2008. 752 p.

63. Welke, R. Service-oriented architecture maturity / R. Welke, R. Hirschheim, A. Schwarz // Computer. — 2011. — Vol. 44, № 2. — P. 61–67.

64. Tsalgatidou, A. An overview of standards and related technology in web services / A. Tsalgatidou, T. Pilioura // Distributed and parallel databases. — 2002. — Vol. 12. — P. 135–162.

65. Globus platform-as-a-service for collaborative science applications / R. Ananthakrishnan, K. Chard, I. Foster et al. // Concurrency and Computation: Practice and Experience. — 2015. — Vol. 27, № 2. — P. 290–305.

66. Foster, I. Globus Online: Accelerating and democratizing science through cloud-based services / I. Foster // IEEE Internet Computing. — 2011. — Vol. 15, № 3. — P. 70–73.

67. Foster, I. Globus toolkit version 4: Software for service-oriented systems / I. Foster // Journal of computer science and technology. — 2006. — Vol. 21. — P. 513–520.

68. Foster, I. The Grid: Blueprint for a new computing infrastructure / I. Foster, C. Kesselman. — Morgan-Kaufmann, 2002. — 667 p.

69. Foster, I. Globus: A metacomputing infrastructure toolkit / I. Foster, C. Kesselman // The International Journal of Supercomputer Applications and High Performance Computing. — 1997. — Vol. 11, № 2. — P. 115–128.

70. WS-BPEL 2.0 for SOA Composite applications with oracle SOA Suite 11g / M.B. Juric, S. Chandrasekaran, A. Frece et al. — Packt Publishing Ltd, 2010. — 644 p.

71. Juric, M.B. Business process execution language for web services: an architect and developer's guide to orchestrating web services using BPEL4WS / M.B. Juric, B. Mathew, P.G. Sarang. — Packt Publishing Ltd, 2006. — 353 p.

72. Kim, S.J. Foundation for composable microservices for rapid synthesis of highly reliable software systems / S.J. Kim. — The University of Texas at Dallas, 2004.

73. High-assurance synthesis of security services from basic microservices / S. Kim, F.B. Bastani, I.L. Yen et al. // Proc. of the 14th Intern. Symposium on Software Reliability Engineering (ISSRE 2003). — IEEE, 2003. — P. 154–165.

74. Fielding, R.T. Architectural styles and the design of network-based software architectures / R.T. Fielding. — University of California, Irvine, 2000.

75. UNICORE 7 – Middleware services for distributed and federated computing / K. Benedyczak, B. Schuller, M. Petrova-El Sayed et al. // Proc. of the Intern. Conf. on High Performance Computing & Simulation (HPCS). — IEEE, 2016. — P. 613–620.

76. Galaxy CloudMan: delivering cloud compute clusters / E. Afgan, D. Baker, N. Coraor et al. // BMC bioinformatics. — 2010. — Vol. 11, № 12. — P. 1–6.

77. Rajasekar, A. iRODS primer: integrated rule-oriented data system / A. Rajasekar. Morgan & Claypool Publishers, 2010.

78. Сухорослов, О.В. Реализация и композиция проблемно-ориентированных сервисов в среде MathCloud / О.В. Сухорослов // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. — 2011. — № 17 (234). — С. 101–112.

79. Sukhoroslov, O. Building web-based services for practical exercises in parallel and distributed computing / O. Sukhoroslov // Journal of Parallel and Distributed Computing. 2018. Vol. 118. P. 177–188.

80. Радченко, Г.И. Распределенные вычислительные системы / Г.И. Радченко. — Челябинск: Фотохудожник, 2012. — 184 с.

81. Savchenko, D.I. Microservices validation: Mjolnirr platform case study / D.I. Savchenko, G.I. Radchenko, O. Taipale // Proc. of the 38th Intern. convention on information and communication technology, electronics and microelectronics (MIPRO). — IEEE, 2015. — P. 235–240.

82. Бухановский, А.В. Интеллектуальные программные комплексы компьютерного моделирования сложных систем: концепция, архитектура и примеры реализации / А.В. Бухановский, С.В. Ковальчук, С.В. Марьин // Известия высших учебных заведений. Приборостроение. — 2009. — Т. 52, № 10. — С. 5–24.

83. CLAVIRE: облачная платформа для обработки данных больших объемов / В.Н. Васильев, К.В. Князьков, Т.Н. Чуров и др. // Информационно-измерительные и управляющие системы. — 2012. — Т. 10, № 11. — С. 7–16.

84. Облачный сервис для решения многомасштабных задач нанотехнологии на суперкомпьютерных системах / С.В. Поляков, А.В. Выродов, Д.В. Пузырьков и др. // Труды Института системного программирования РАН. — 2015. — Т. 27, № 6. — С. 409–420.

85. Разработка и реализация облачной системы для решения высокопроизводительных задач / А.О. Кудрявцев, В.К. Кошелев, А.О. Избышев и др. // Труды Института системного программирования РАН. — 2013. — Т. 24. — С. 13–34.

86. Высокопроизводительные вычисления как облачный сервис: ключевые проблемы / А.О. Кудрявцев, В.К. Кошелев, А.О. Избышев и др. // Параллельные вычислительные технологии (ПаВТ 2013): Труды междунар. науч. конф. (1-5 апреля 2013 г., г. Челябинск, Россия). — Челябинск: Издательский центр ЮУрГУ, 2013. — С. 432–438.

87. Развитие информационно-телекоммуникационных систем в ДВО РАН / А.И. Ханчук, А.А. Сорокин, С.И. Смагин и др. // Информационные технологии и вычислительные системы. — 2013. — № 4. — С. 45–57.

88. Разработка программных средств виртуальной интеграции распределенных источников данных для создания масштабных информационных инфраструктур профессионального назначения / А.Н. Поляков, А.А. Пойда, А.А. Сорокин и др. // Информатика и системы управления. — 2013. — № 3. — С. 152–160.

89. Denisov, S.A. Model of a management system for deterministic scientific services of digital platform / S.A. Denisov, A.A. Sorokin // Procedia Computer Science. — 2021. — Vol. 186. — P. 1–10.

90. Ayzel, G. Development and Evaluation of National-Scale Operational Hydrological Forecasting Services in Russia / G. Ayzel, A.A. Sorokin // Proceedings of the CEUR Workshop Proceedings, Khabarovsk, Russia. — 2021. — P.14-16.

91. Шокин, Ю.И. Технологии создания распределенных информационных систем для поддержки научных исследований / Ю.И. Шокин, А.М. Федотов, О.Л. Жижимов // Вычислительные технологии. — 2015. — Т. 20, № 5. — С. 251–274.

92. Интеграция разнородных данных в задачах исследования природных экосистем / О.Л. Жижимов, Ю.И. Молородов, И.А. Пестунов и др. // Вестник Новосибирского государственного университета. Серия: Информационные технологии. — 2011. — Т. 9, № 1. — С. 67–74.

93. ИТ-инфраструктура научных исследований: методический подход и реализация / Л.В. Массель, Е.А. Болдырев, Н.Н. Макагонова и др. // Вычислительные технологии. — 2006. — Т. 11, № S8. — С. 59–68.

94. Сервисы и инфраструктура пространственных данных междисциплинарных исследований геосистем Прибайкалья и Забайкалья / Бычков И.В., Ружников Г.М., Хмельнов А.Е. и др. // Геоинформационные технологии и математические модели для мониторинга и управления экологическими и социально-экономическими системами. Под ред. И.Н. Ротановой. — Барнаул: Ин-т водных и экологических проблем СО РАН, Ин-т вычислительных технологий СО РАН, 2011. — С. 145–156.

95. Система планирования и выполнения композиций веб-сервисов в гетерогенной динамической среде / И.В. Бычков, Г.М, Ружников, Р.К. Федоров и др. // Вычислительные технологии. — 2016. — Т. 21, № 6. — С. 18–35.

96. Сервис-ориентированное управление распределенными вычислениями на основе мультиагентных технологий / И.В. Бычков, Г.А. Опарин, А.Г. Феоктистов и др. // Известия Южного федерального университета. Технические науки. — 2014. — № 12. — С. 17–27.

97. Опарин, Г.А. Микросервисы как фундаментальная основа распределенного сборочного программирования / Г.А. Опарин, В.Г. Богданова, А.А. Пашинин //

ИТНОУ: Информационные технологии в науке, образовании и управлении. — 2018. — № 2 (6). — С. 21–26.

98. Сервис-ориентированный подход к организации распределенных вычислений с помощью инструментального комплекса DISCENT / Бычков И.В., Опарин Г.А., Феоктистов А.Г. и др. // Информационные технологии и вычислительные системы. — 2014, № 2. — С. 7–15.

99. Microservice-Based Approach to Simulating Environmentally-Friendly Equipment of Infrastructure Objects Taking into Account Meteorological Data / R. Kostromin, O. Basharina, A. Feoktistov et al. // Atmosphere. — 2021. — Vol. 12, № 9. — P. 1217.

100. Михеев, А. Война стандартов в мире workflow. [Электронный ресурс] / А. Михеев, М. Орлов. — 2007. — Режим доступа: https://ecm-journal.ru/material/Vojjna-standartov-v-mire-workflow (дата обращения: 03.12.2025).

101. Романов, М. Некоторые, наиболее известные стандарты описания бизнес-процессов. [Электронный ресурс] / М. Романов. — 2009. — Режим доступа: https://ecm-journal.ru/material/nekotorye-naibolee-izvestnye-standarty-opisanija-biznes-processov (дата обращения: 03.12.2025).

102. Артамонов, И.В. Описание бизнес-процессов: вопросы стандартизации / И.В. Артамонов // Прикладная информатика. — 2011. — № 3 (33). — С. 20–28.

103. Web Services Business Process Execution Language Version 2.0. [Электронный ресурс]. — 2007. — Режим доступа: https://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf (дата обращения: 03.12.2025).

104. Common Workflow Language. [Электронный ресурс]. — 2024. — Режим доступа: https://www.commonwl.org (дата обращения: 12.11.2025).

105. Docker. [Электронный ресурс]. — 2024. — Режим доступа: https://www.docker.com/ (дата обращения: 03.12.2025).

106. Popek, G.J. Formal Requirements for Virtualizable Third Generation Architectures / G.J. Popek, R.P. Goldberg // Communications of the ACM. — 1974. Vol. 17, № 7. — P. 412–421.

107. chroot (2). [Электронный ресурс]. — 2024. — Режим доступа: https://man7.org/linux/man-pages/man2/chroot.2.html (дата обращения: 03.12.2025).

108. Jail – manage system jails. [Электронный ресурс]. — 2023. — Режим доступа: https://man.freebsd.org/cgi/man.cgi?jail (дата обращения: 03.12.2025).

109. Google Code Archive. Linuxcontainers. [Электронный ресурс]. — 2016. — Режим доступа: https://code.google.com/archive/p/linuxcontainers/ (дата обращения: 03.12.2025).

110. Container and virtualization tools. [Электронный ресурс]. — 2024. — Режим доступа: https://linuxcontainers.org/ (дата обращения: 03.12.2025).

111. Kubernetes. [Электронный ресурс]. — 2024. — Режим доступа: https://kubernetes.io/ (дата обращения: 03.12.2025).

112. Chengeta, K. Comparing the performance between Virtual Machines and Containers using deep learning credit models / K. Chengeta // Proceedings of the International Conference on Artificial Intelligence and its Applications. ACM, 2021. — P. 8.

113. Performance Evaluation of Docker Container and Virtual Machine / A.M. Potdar, D.G. Narayan, S. Kengond et al. // Procedia Computer Science. 2020. — Vol. 171. — P. 1419–1428.

114. Ramírez-Peralta, D. Containers vs. virtual machines: performance comparison / D. Ramírez-Peralta, E. Alcudia-Fuentes // Journal of Scientific and Technical Applications. — Vol. 7, № 20. — P. 1–9.

115. Тыугу, Э.Х. Решение задач на вычислительных моделях / Э.Х. Тыугу // Журнал вычисл. математики и матем. физики. — 1970. — Т.10, № 3. — С. 716–733.

116. Опарин, Г.А. К теории планирования вычислительного процесса в пакетах прикладных программ / Г.А. Опарин // Пакеты прикладных программ. Методы и разработки. — Новосибирск: Наука, 1981. — С. 5–20.

117. Вальковский, В.А. Синтез параллельных программ и систем на вычислительных моделях / В.А. Вальковский, В.Э. Малышкин. — Новосибирск: Наука, 1988. — 129 с.

118. Опарин, Г.А. Автоматизация разработки и применения пакетов программ для исследования динамики сложных управляемых систем / Г.А. Опарин // Автореф. дис. докт. техн. наук: 05.13.11. — Иркутск: Изд–во ИДСТУ СО РАН, 1998. — 40 с.

119. Феоктистов, А.Г. Организация предметно-ориентированных распределенных вычислений в гетерогенной среде на основе мультиагентного управления заданиями / А.Г. Феоктистов // Автореф. дис. докт. техн. наук: 05.13.11. — Иркутск: Изд–во ИДСТУ СО РАН, 2021. — 39 с.

120. Федоров Р.К. Сервис-ориентированная информационно-аналитическая среда композиции сервисов обработки пространственных данных // Автореф. дис. докт. техн. наук: 2.3.5. — Иркутск: Изд–во ИДСТУ СО РАН, 2024. — 39 с.

121. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ / А.И. Легалов // Вычислительные технологии. — 2005. — № 1(10). — С. 71-89.

122. Опарин, Г.А. Инструментальный комплекс ORLANDO TOOLS / Г.А. Опарин, А.Г. Феоктистов, А.П. Новопашин, С.А. Горский // Программные продукты и системы. — 2007. — № 4. — С. 63-65.

123. Опарин, Г.А. Новые языковые средства пакетов прикладных программ в метасистеме САТУРН / Г.А. Опарин, Д.Г. Феоктистов // Пакеты прикладных программ. Функциональное наполнение. — Новосибирск: Наука, 1985. — С. 20–28.

124. Топорков, В.В. Модели распределенных вычислений / В.В. Топорков. — М.: Физматлит, 2004. — 320 с.

125. HTCondor. [Электронный ресурс]. Режим доступа: https://htcondor.org/ (дата обращения: 06.12.2025).

126. Elradi, M. D. Ansible: A reliable tool for automation / M. D. Elradi // Electrical and Computer Engineering Studies. – 2023. – Т. 2. – №. 1.

127. Горбунов–Посадов, М.М. Системное обеспечение пакетов прикладных программ / М.М. Горбунов–Посадов, Д.А. Корягин, В.В. Мартынюк. — М.: Наука, 1990. — 208 с.

128. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. — М., Мир. — 1982.

129. Dejanović I. Parglare: a LR/GLR parser for python // Science of Computer Programming. — 2022. — Т. 214. — С. 102734.

130. Tchernykh, A. En-AR-PRNS: Entropy-Based Reliability for Configurable and Scalable Distributed Storage Systems / A. Tchernykh, M. Babenko, A. Avetisyan, A.Yu. Drozdov // Mathematics. — 2022. — Vol. 10, № 1, — P. 84.

131. Arora, I. Improving performance of cloud based transactional applications using in-memory data grid / I. Arora, A. Gupta // International Journal of Computer Applications. — 2014. — Vol. 107, № 13. — P. 14–19.

132. Guroob, A.H. EA2-IMDG: Efficient Approach of Using an In-Memory Data Grid to Improve the Performance of Replication and Scheduling in Grid Environment Systems / A.H. Guroob // Computation. — 2023. — Vol. 11, № 3. — P. 65.

133. Grover, P. Big data analytics: A review on theoretical contributions and tools used in literature / P. Grover, A.K. Kar // Global Journal of Flexible Systems Management. — 2017. — Vol. 18. — P. 203–229.

134. Cimino de Souza, L. A middleware solution for integrating and exploring IoT and HPC capabilities / L. de Souza Cimino, J.E.E. de Resende, L.H.M. Silva et al. // Software Practice and Experience. — 2019. — Vol. 49. — P. 584–616.

135. Hazelcast. [Электронный ресурс]. — Режим доступа: https://hazelcast.com/ (дата обращения: 03.12.2025).

136. Infinispan. In-Memory Distributed Data Store. [Электронный ресурс]. — Режим доступа: https://infinispan.org/ (дата обращения: 03.12.2025).

137. Apache Ignite. Distributed Database for High Performance Applications with In-Memory Speed. [Электронный ресурс]. — Режим доступа: https://ignite.apache.org/ (дата обращения: 03.12.2025).

138. Johns, M. Getting Started with Hazelcast / M. Johns. — Birmingham, UK: Packt Publishing Ltd., — 2013. — 136 p.

139. Marchioni, F. Infinispan Data Grid Platform / F. Marchioni. — Birmingham, UK: Packt Publishing Ltd., — 2012. — 150 p.

140. Bhuiyan, S.A. High Performance In-Memory Computing with Apache Ignite / S.A. Bhuiyan, M. Zheludkov, T. Isachenko. Leanpub: Victoria, BC, Canada, 2018. [Электронный ресурс]. — Режим доступа: http://samples.leanpub.com/ignite-sample.pdf (дата обращения: 03.12.2025).

141. Kathiravelu, P. An adaptive distributed simulator for cloud and mapreduce algorithms and architectures / P. Kathiravelu, L. Veiga // Proceedings of the 7th International Conference on Utility and Cloud Computing, London, UK, 8–11 December 2014. — IEEE, 2014. — P. 79–88.

142. Zhou, M. Application of in-memory computing to online power grid analysis / M. Zhou, D. Feng // IFAC-PapersOnLine. — 2018. — Vol. 51. — P. 132–137.

143. Zhou, M. Graph Computing and Its Application in Power Grid Analysis / M. Zhou, J. Yan, Q. Wu // CSEE Journal of Power and Energy Systems. — 2022. — Vol. 8. — P. 1550–1557.

144. Capacity Planning. [Электронный ресурс]. — Режим доступа: https://apacheignite.readme.io/docs/capacity-planning (дата обращения: 29.04.2025).

145. Feoktistov, A. An Approach to Implementing High-Performance Computing for Problem Solving in Workflow-based Energy Infrastructure Resilience Studies / A. Feoktistov, A. Edelev, A. Tchernykh et al. // Computation. — 2023. — Vol. 11, № 12. — P. 243.

146. Kabakuş, A.T. A Performance Comparison of Java Cache Memory Implementations / A.T. Kabakuş // Gümüşhane Üniversitesi Fen Bilimleri Dergisi. — 2020. — Vol. 10, №. 3. — P. 844–852.

147. Kosyanov, N. On an Improvement to the Next-step Procedure / N. Kosyanov // Proceedings of 7th International Workshop on Information, Computation, and Control Systems for Distributed Environments. — 2025. — P. 149–151.

148. Sakellari, G. A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing / G. Sakellari, G. Loukas // Simulation Modelling Practice and Theory. — 2013. — Vol. 39. — P. 92–103.

149. Cintuglu, M.H. A survey on smart grid cyber-physical system testbeds / M.H. Cintuglu, O.A. Mohammed, K. Akkaya et al. // IEEE Communications Surveys & Tutorials. — 2016. — Vol. 19, № 1. — P. 446–464.

150. Radchenko, G. A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments / G. Radchenko, E. Hudyakova // Proceedings of the 8th UNICORE Summit, Dresden, 2012. — Vol. 15. — P. 57–66.

151. Fox, G.C. FutureGrid: a reconfigurable testbed for cloud, HPC, and Grid computing / G.C. Fox, G. von Laszewski, J. Diaz et al. // Contemporary High Performance Computing. Vetter J.S. (Ed.). — New York: Chapman and Hall/CRC, 2013. — P. 603–635.

152. Gao, Y. LinkLab: A scalable and heterogeneous testbed for remotely developing and experimenting IoT applications / Y. Gao, J. Zhang, G. Guan et al. // Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI). — IEEE, Sydney, 2020. — P. 176–188.

153. Smadi, A.A. A Comprehensive survey on cyber-physical smart grid testbed architectures: Requirements and challenges / A.A. Smadi, B.T. Ajao, B.K. Johnson et al. // Electronics. — 2021. — Vol. 10, № 9. — P. 1043.

154. Ramesh, J. Cloud infrastructure-as-a-service testbed implementation using OpenStack / J. Ramesh, D. Sankalpa, R. Aburukba et al. // Proceedings of the 2022 International Symposium on Networks, Computers and Communications (ISNCC). — IEEE, Shenzhen, 2022. — P. 1–8.

155. Паклин, Н.Б. Бизнес-аналитика. От данных к знаниям / Н.Б. Паклин, В.И. Орешков. — СПб.: Питер, 2013. — 622 с.

156. Романова, И.К. Современные методы визуализации многомерных данных: анализ, классификация, реализация, приложения в технических системах / И.К. Романова // Машиностроение и компьютерные технологии. — 2016. — № 3. — С. 133–167.

157. Bertin, J. Semiology of Graphics: Diagrams, Networks, Maps / J. Bertin // University of Wisconsin Press. — 1983. — 415 p.

158. JavaScript Object Notation. [Электронный ресурс]. — Режим доступа: https://www.json.org/ (дата обращения: 03.12.2025).

159. Highcharts. [Электронный ресурс]. — Режим доступа: https://cdnjs.com/libraries/highcharts (дата обращения: 03.12.2025).

160. Боровиков Р. 10 лучших JavaScript библиотек для визуализации данных на графиках и диаграммах. [Электронный ресурс]. — Режим доступа: https://habr.com/ru/articles/457946/ (дата обращения: 03.12.2025).

161. Graphviz. [Электронный ресурс]. — Режим доступа: https://graphviz.org/ (дата обращения: 03.12.2025).

162. graphviz 0.21. [Электронный ресурс]. — Режим доступа: https://pypi.org/project/graphviz/ (дата обращения: 03.12.2025).

163. Maulik, A. Optimal operation of microgrid using four different optimization techniques / A. Maulik, D. Das // Sustainable Energy Technologies and Assessments. — 2017. — Vol. 21. — P. 100–120.

164. Priesmann, J. Are complex energy system models more accurate? An intra-model comparison of power system optimization models / J. Priesmann, L. Nolting, A. Praktiknjo // Applied Energy. — 2019. — Vol. 255. — P. 113783.

165. Biscani, F. A parallel global multiobjective framework for optimization: pagmo / F. Biscani, D. Izzo // Journal of Open Source Software. — 2020. — Vol. 5 — № 53. — P. 2338.

166. Vulnerability analysis method based on risk assessment for gas transmission capabilities of natural gas pipeline networks / W. Wang, Y. Zhang, Y. Li et al. // Reliability Engineering and System Safety. — 2022. — Vol. 218. — P. 108150.

167. Mugume, S.N. A global analysis approach for investigating structural resilience in urban drainage systems / S.N. Mugume, D.E. Gomez, G. Fu et al. // Water Research. — 2015. — Vol. 81. — P. 15–26.

168. Jonsson, H. Identifying critical components in technical infrastructure networks / H. Jonsson, J. Johansson // Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability. — 2008. — Vol. 222, № 2. — P. 235–243.

169. Rodriguez, M.A. Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms / M.A. Rodriguez, R. Buyya // Future Generation Computer Systems. — 2017. — Vol. 79. — P. 739-750.

170. Iwanaga, T. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses / T. Iwanaga, W. Usher, J. Herman // Socio-Environmental Systems Modelling. — 2022. — Vol. 4. — P. 18155.

171. Соболь, И. М. Глобальные показатели чувствительности для изучения нелинейных математических моделей / И. М. Соболь // Математическое моделирование. — 2005. — Т. 17, № 9. — С. 43-52.

172. Plischke, E. Global sensitivity measures from given data / E. Plischke, E. Borgonovo, C. L. Smith // European Journal of Operational Research. — 2013. — Vol. 226, №3 —P. 536-550.

173. Edelev, A.V. Vulnerability analysis of energy infrastructure and its implementation / A.V. Edelev, N.M. Beresneva, S.A. Gorskiy // Modern high technologies. — 2022. — № 1. — P. 47–52.

174. Juve, A. Characterizing and profiling scientific workflows / G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi // Future Generation Computer Systems. — 2013. —Vol. 29, №. 3. — P. 682–692.

Список принятых сокращений

БЗ – база знаний

ВМ – виртуальная машина

ВО – виртуальный объект

ГРВС – гетерогенная распределенная вычислительная система

ИГУ – ФГБОУ ВО «Иркутский государственный университет»

ИДСТУ СО РАН – ФГБУН «Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук»

ИСЭМ СО РАН – ФГБУН «Институт систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук»

НРП – Научный рабочий процесс

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

ПОВС – предметно-ориентированная вычислительная среда

ПТС – природно-техническая система

БПТ – Байкальская природная территория

ППП – пакет прикладных программ

ПСПО – прикладное и системное программное обеспечение

РПППП – распределенный пакет прикладных программ

СОА – сервис-ориентированная архитектура

СОП – сервис-ориентированное приложение

СРП – системный рабочий процесс

СУПЗ – система управления прохождением заданий

СУРП – система управления рабочими процессами

СХД – система хранения данных

ЦКП – Центр коллективного пользования

ЭК – энергетические комплекс

API – Application Programming Interface

BPEL – Business Process Execution Language

BPEL4WS – Business Process Execution Language for Web Services

BPMI – Business Process Management Initiative

BPML – Business Process Modeling Language

BPSS – Business Process Specification Schema

CPU – Central Processing Unit

Condor DAGMan – Condor Directed Acyclic Graph Manager

CORBA – Common Object Request Broker Architecture

CWL – Common Workflow Language

DAG – Directed Acyclic Graph

DCOM – Distributed Component Object Model

DP – Designer Project

FDE-SWFs – Framework for Development and Execution of Scientific WorkFlows

FLOPS – Floating Operations Per Second

GJMB – GeoJModelBuilder

GT – Globus Toolkit

HPC – High Performance Computing

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IMDG – In-Memory Data Grid

JDL – Job Description Language

JSON – JavaScript Object Notation

MS – Microsoft

OASIS – Organization for the Advancement of Structured Information Standards

OGC – Open Geospatial Consortium

OMG – Object Management Group

Orlando Tools – Object Relations in LANguage of DescriptiOns Tools

OWL – Ontology Web Language

PanDA – Production and Distributed Analysis

PBS – Portable Batch System

PHP – Hypertext Preprocessor

RAM – Random Access Memory

REST – Representational State Transfer

SOAP – Simple Object Access Protocol

UNICORE – Uniform Interface to Computing Resources

UN/CEFACT – United Nations Centre for Trade Facilitation and Electronic Business

URL – Uniform Resource Locator

W3C – The World Wide Web Consortium

WADL – Web Application Description Language

WFMC – Workflow Management Coalition

WSCI – Web Services Choreography Interface

WSCL – Web Services Conversation Language

WS-CDL – Web Services Choreography Description Language

WS-BPEL – Web Services Business Process Execution Language

WSFL – Web Services Flow Language

WMS – Workflow Management System

WPS – Web Processing Service

WSDL – Web Services Description Language

XLANG – Executable Language Grounding

XML – eXtensible Markup Language

YAML – YAML Ain't Markup Language

YAWL – Yet Another Workflow Language

Приложение А. Акты о внедрении

Акт о внедрении результатов диссертации в ИСЭМ СО РАН.

Утверждаю:
Врио директора ИСЭМ СО РАН
академик РАН

В.А. Стенников

«24» 09 2025 г.

Акт
о внедрении результатов диссертационного исследования Воскобойникова Михаила Леонтьевича «Технология разработки и применения сервис-ориентированных приложений в контейнеризированной вычислительной среде», представленного на соискание ученой степени кандидата технических наук по научной специальности 2.3.5. – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Настоящим актом подтверждается внедрение результатов диссертационного исследования Михаила Леонтьевича «Технология разработки и применения сервис-ориентированных приложений в контейнеризированной вычислительной среде» в Институте систем энергетики им. Л.А. Мелентьева Сибирского отделения Российской академии наук (ИСЭМ СО РАН) с целью поддержки автоматизации создания и применения приложений, базирующихся на использовании сервис-ориентированных рабочих процессов.

Использование разработанных в рамках диссертационного исследования инструментальных средств создания и применения сервис-ориентированных приложений обеспечило организацию требуемой вычислительной среды для решения ряда актуальных задач, связанных с оценкой критичности элементов энергетических инфраструктур. Выполнение многовариантных расчетов в такой гетерогенной распределенной вычислительной среде под управлением диспетчера научных рабочих процессов, входящего в состав инструментальных средств, обеспечило существенное сокращение сроков подготовки и проведения вычислительных экспериментов при решения данных задач в сравнении с другими известными диспетчерами подобного назначения.

Заместитель директора по научной работе,
заведующий отделом энергетической безопасности № 30,
доктор технических наук С.М. Сендеров

Старший научный сотрудник
лаборатории живучести систем энергетики № 33,
кандидат технических наук А.В. Еделев

«24» 09 2025 г.

Акт о внедрении результатов диссертации в ИВМиМГ СО РАН

Утверждаю:
Директор ИВМиМГ СО РАН
д.ф.-м.н., проф. РАН
Марченко М.А.
«5» ноября 2025 г.

Акт
о внедрении результатов диссертации Воскобойникова Михаила Леонтьевича «Технология разработки и применения сервис-ориентированных приложений в контейнеризированной вычислительной среде», представленной на соискание ученой степени кандидата технических наук по научной специальности 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Настоящим актом подтверждается внедрение результатов диссертации Михаила Леонтьевича «Технология разработки и применения сервис-ориентированных приложений в контейнеризированной вычислительной среде» в Институте вычислительной математики и математической геофизики Сибирского отделения Российской академии наук (ИВМиМГ СО РАН) с целью поддержки автоматизации создания и применения сервисов для сервис-ориентированных приложений.

Использование разработанных в рамках диссертации средств инструментального комплекса Framework for Development and Execution of Scientific WorkFlows (FDE-SWFs) обеспечило разработку сервиса на основе стохастического «генератора погоды» для Байкальской природной территории (программного комплекса, реализующего численную стохастическую модель комплекса метеорологических параметров).

Разработанный сервис успешно использован в ИВМиМГ СО РАН в рамках выполнения гранта № 075-15-2024-533 Министерства науки и высшего образования РФ на выполнение крупного научного проекта по приоритетным направлениям научно-технологического развития (проект «Фундаментальные исследования Байкальской природной территории на основе системы взаимосвязанных базовых методов, моделей, нейронных сетей и цифровой платформы экологического мониторинга окружающей среды», рег. № 124052100088-3).

Заведующий лабораторией
стохастических задач,
доктор физико-математических наук Каргаполова Н.А.

«5» ноября 2025 г.

Приложение Б. Свидетельства о государственной регистрации программ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023685789

Планировщик схем решения задач для распределенных пакетов прикладных программ

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (RU)***

Автор(ы): ***Воскобойников Михаил Леонтьевич (RU)***

Заявка № **2023685658**
Дата поступления **30 ноября 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***30 ноября 2023 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024668180

Библиотека программных агентов организации распределенной базы данных в оперативной памяти

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (RU)***

Авторы: ***Воскобойников Михаил Леонтьевич (RU), Феоктистов Александр Геннадьевич (RU)***

Заявка № **2024667916**
Дата поступления **06 августа 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***06 августа 2024 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bca2026
Владелец **Зубов Юрий Сергеевич**
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025660056

Модуль прогнозирования размера базы данных в оперативной памяти узлов вычислительного кластера

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (RU)***

Авторы: ***Феоктистов Александр Геннадьевич (RU), Воскобойников Михаил Леонтьевич (RU)***

Заявка № **2025618995**
Дата поступления **21 апреля 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***21 апреля 2025 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

Ю.С. Зубов

Приложение В. Сведения о WMS

Таблица В.1 – Основные сведения о WMS: источник / сайт, область исследования

WMS	Источник / сайт	Область исследования
UNICORE	https://www.unicore.eu/	Общего назначения
DAGMan	https://htcondor.org/dagman/dagman.html	Общего назначения
Grid Application Development Software (GrADS)	http://cola.gmu.edu/grads/	Общего назначения
Pegasus	https://pegasus.isi.edu/	Общего назначения
Imperial College e-Science Network Infrastructure (ICENI)	–	Общего назначения
Triana	http://www.trianacode.org	Общего назначения
GridAnt	https://ieeexplore.ieee.org/document/1265491	Общего назначения
GridFlow	https://gridflow.ca/	Анализ данных мультимедиа
Taverna	http://www.taverna.org.uk/download/workbench/	Биоинформатика, Общего назначения
Kepler	https://kepler-project.org/index.html	Общего назначения
Askalon	https://askalon.org/	Общего назначения
Karajan	http://www.cogkit.org/wiki	Общего назначения
OT	http://orlando1.icc.ru/	Общего назначения
Weka4WS	https://scalab.dimes.unical.it/weka4ws/about/	Сбор данных
General Workflow Execution Service (GWES)	https://onlinelibrary.wiley.com/doi/10.1002/cpe.1002	Общего назначения
Galaxy	https://usegalaxy.org/	Геномные исследования

Таблица В.1 – Основные сведения о WMS: источник / сайт, область исследования (продолжение)

WMS	Источник / сайт	Область исследования
Konstanz Information Miner (KNIME)	https://www.knime.com/	Фармацевтика, анализ данных CRM, бизнес аналитика
GridBus	http://www.cloudbus.org/	Общего назначения
CloudBus	http://www.cloudbus.org/	Общего назначения
BPEL DP	https://projects.eclipse.org/projects/soa.bpel	Геоинформатика
GJMB	https://sourceforge.net/projects/geopw/files/	Геоинформатика
Apache Airflow	https://airflow.apache.org/	Общего назначения
HyperFlow	https://github.com/hyperflow-wms/hyperflow	Общего назначения
WaaS CP	http://www.cloudbus.org/	Общего назначения

Таблица В.2 – Основные сведения о WMS: начало проекта, поддерживается / не поддерживается, лицензия

WMS	Начало проекта	Поддержка проекта (+ / -)	Лицензия
UNICORE	1996 г.	+	Berkeley Software Distribution (BSD) License
DAGMan	Конец 1990-х	+	Apache license 2.0
GrADS	1999 г.	+	Коммерческое ПО
Pegasus	2001 г.	+	Apache license 2.0
ICENI	2002 г.	-	ICENI Open Source Code Licence
Triana	2002 г.	-	Apache license 2.0
GridAnt	2003 г.	-	Globus Toolkit Public License
GridFlow	2003 г.	+	GNU General Public License
Taverna	2003 г.	-	GNU General Public License
Kepler	2004 г.	+	Berkeley Software Distribution License

Таблица В.2 – Основные сведения о WMS: начало проекта, поддерживается / не поддерживается, лицензия (продолжение)

WMS	Начало проекта	Поддержка проекта (+ / -)	Лицензия
Askalon	2005 г.	+	Globus Toolkit Public License
Karajan	2005 г.	-	GNU General Public License
OT	2005 г.	-	GNU General Public License
Weka4WS	2005 г.	-	GNU General Public License
GWES	2006 г.	-	Globus Toolkit Public License
Galaxy	2007 г.	+	The Galaxy Service is a free, public, Internet accessible resource
KNIME	2007 г.	+	GNU General Public License
GridBus	2008 г.	-	GNU General Public License
CloudBus	2011 г.	-	GNU General Public License
BPEL DP	2012 г.	+	Eclipse Public License 2.0
GJMB	2013 г.	-	GNU General Public License
Apache Airflow	2014 г.	+	Apache License 2.0
HyperFlow	2014 г.	+	MIT License
WaaS CP	2021 г.	+	GNU General Public License

Таблица В.3 – Функциональные возможности WMS по формированию НРП

WMS	Текстовый язык описания НРП	Графический редактор НРП	Поддержка абстрактного / конкретного НРП	Поддержка DAG / Non-DAG	Поддержка циклов / ветвлений
UNICORE	XML+	+	- / +	+ / +	+ / +
DAGMan	Скриптовый язык	-	+ / +	+ / -	- / -
GrADS	Аннотированный язык программирования	+	+ / -	+ / -	- / -

Таблица В.3 – Функциональные возможности WMS по формированию НРП (продолжение)

WMS	Текстовый язык описания НРП	Графический редактор НРП	Поддержка абстрактного / конкретного НРП	Поддержка DAG / Non-DAG	Поддержка циклов / ветвлений
Pegasus	XML+	-	+ / -	+ / -	- / -
ICENI	XML+, BPEL	+	+ / -	+ / +	+ / +
Triana	XML+	+	+ / -	+ / +	+ / +
GridAnt	XML+	-	- / +	+ / +	+ / +
GridFlow	XML+, RSL	+	+ / -	+ / +	+ / +
Taverna	XML+	+	+ / +	+ / -	- / -
Kepler	XML+	+	+ / +	+ / +	+ / +
Askalon	XML+	-	+ / -	+ / +	+ / +
Karajan	XML+	+	+ / -	+ / +	+ / +
OT	XML+	+	+ / +	+ / +	+ / +
Weka4WS	–	+	+ / -	+ / -	- / -
GWES	Сеть Петри	+	+ / -	+ / -	- / -
Galaxy	–	+	+ / -	+ / -	+ / -
KNIME	XML+	+	+ / -	+ / -	- / -
GridBus	XML+	-	+ / +	+ / -	- / -
CloudBus	XML+	-	+ / +	+ / -	- / -
BPEL DP	BPEL	+	+ / -	+ / +	+ / +
GJMB	XML+	+	+ / -	+ / -	- / -
Apache Airflow	Python	-	- / +	+ / -	- / -
HyperFlow	JSON, JavaScript	-	+ / –	+ / -	- / -
WaaS CP	XML+	-	+ / +	+ / -	- / -

Таблица В.4 – Функциональные возможности WMS по управлению вычислениями

WMS	Поддержка комплексирования пакетов по данным / управлению	Поддержка процедурной / непроцедурной постановок задач	Поддержка дополнительных системных операций в НРП	Поддержка работы с веб-сервисами в НРП
UNICORE	- / -	+ / -	+	-
DAGMan	- / -	+ / -	+	-
GrADS	- / -	+ / -	+	-
Pegasus	- / -	+ / +	+	-
ICENI	+ / +	+ / -	+	+
Triana	+ / +	+ / -	-	+
GridAnt	- / -	+ / -	-	-
GridFlow	- / -	+ / -	-	-
Taverna	+ / +	+ / -	-	+
Kepler	+ / +	+ / -	+	+
Askalon	+ / +	+ / -	-	+
Karajan	+ / -	+ / -	+	-
OT	+ / +	+ / +	+	+
Weka4WS	+ / +	+ / -	-	+
GWES	+ / +	+ / -	-	+
Galaxy	+ / +	+ / -	+	+
KNIME	+ / +	+ / -	+	+
GridBus	- / -	+ / -	-	-
CloudBus	- / -	+ / -	-	-
BPEL DP	+ / +	+ / -	-	+
GJMB	+ / +	+ / -	+	+
Apache Airflow	+ / +	+ / -	-	-
HyperFlow	+ / +	+ / -	+	+
WaaS CP	+ / +	+ / -	-	+

Таблица В.5 – Возможности вычислительных моделей систем WMS по поддержке параллельных вычислений

WMS	Поддержка параллельных вычислений на уровне задачи / данных / конвейера	Вычислительная среда: кластер / Grid / облачная среда	Оценка времени выполнения на уровне модуля / НРП	Дополнительное связующее ПО
UNICORE	+ / + / -	- / + / -	- / -	-
DAGMan	+ / + / -	+ / + / +	- / -	HTCondor
GrADS	+ / + / -	- / + / -	+ / -	Globus Toolkit
Pegasus	+ / + / -	+ / + / +	+ / +	DAGMan
ICENI	+ / - / -	- / + / -	+ / -	Globus Toolkit
Triana	+ / + / +	- / + / -	- / -	Globus Toolkit
GridAnt	+ / - / -	- / + / -	- / -	Globus Toolkit
GridFlow	+ / - / -	- / + / -	+ / -	Titan, WebRun
Taverna	+ / + / +	+ / - / -	- / -	-
Kepler	+ / - / -	+ / + / -	- / -	Globus Toolkit
Askalon	+ / - / -	- / + / -	+ / -	Globus Toolkit
Karajan	+ / - / -	- / + / -	- / -	Globus Toolkit
OT	+ / + / +	+ / + / +	+ / +	-
Weka4WS	+ / + / -	- / + / -	- / -	Globus Toolkit
GWES	+ / - / -	+ / + / +	- / -	Globus Toolkit
Galaxy	- / + / -	- / - / +	- / -	-
KNIME	+ / + / -	+ / - / -	- / -	-
GridBus	+ / - / -	- / + / -	+ / +	Globus Toolkit UNICORE, DAGMan
CloudBus	+ / - / -	- / + / +	+ / +	Globus Toolkit UNICORE, DAGMan, Aneka

Таблица В.5 – Возможности вычислительных моделей систем WMS по поддержке параллельных вычислений (продолжение)

WMS	Поддержка параллельных вычислений на уровне задачи / данных / конвейера	Вычислительная среда: кластер / Grid / облачная среда	Оценка времени выполнения на уровне модуля / НРП	Дополнительное связующее ПО
BPEL DP	+ / - / -	+ / + / +	- / -	-
GJMB	+ / - / -	+ / + / +	- / -	-
Apache Airflow	+ / + / +	- / - / +	- / -	Helm Chart for Kubernetes
HyperFlow	+ / - / -	- / - / +	- / -	-
WaaS CP	+ / - / -	- / + / +	+ / +	JClouds API

Приложение Г. Спецификация вычислительной модели в расширенной форме Бэкуса-Наура

вычислительная модель = объект {объект};

объект = модуль | сервис | параметр | операция | продукция |

| постановка задачи | научный рабочий процесс | задание |

| вычислительный ресурс | административная политика |

| вычислительная история;

модуль = базовый модуль | системный модуль;

базовый модуль = имя модуля ’,’ список формальных параметров;

системный модуль = имя модуля ‘,’ список формальных параметров;

имя модуля = строка;

список формальных параметров = имя параметра ‘,’ назначение параметра {имя параметра ‘,’ назначение параметра};

назначение параметра = “входной” | “выходной”;

параметр = базовый параметр | системный параметр;

базовый параметр = имя базового параметра ‘,’ тип параметра ‘,’ структура данных ‘,’ [индекс 1] ‘,’ [индекс 2] ‘,’ [параметр, на основе которого создается параллельный список данных ‘,’ число элементов списка] ‘,’ [имя файла] ‘,’ [область допустимых значений];

системный параметр = имя системного параметра ‘,’ тип параметра ‘,’ структура данных ‘,’ [индекс 1] ‘,’ [индекс 2] ‘,’ [параметр, на основе которого создается параллельный список данных ‘,’ число элементов списка] ‘,’ [имя файла];

базовый параметр = вычислительный параметр | логический параметр;

имя базового параметра = имя параметра;

имя системного параметра = имя параметра;

имя параметра = строка;

тип параметра = “integer” | “long” | “Boolean” | “float” | “double” | “char” | “string”;

структура данных = “скаляр” | “вектор” | “матрица” | “файл” |

| “составной параметр” | “параллельный список данных”;

составной параметр = имя параметра ‘,’ список параметров;

список параметров = имя параметра ‘,’ имя параметра | имя параметра { имя параметра };

имя файла = строка;

индекс 1 = имя параметра;

индекс 2 = имя параметра;

параметр, на основе которого создается параллельный список данных =

имя параметра;

параллельный список данных = имя параметра ‘,’ параметр, на основе которого создается параллельный список данных ‘,’ число элементов списка;

число элементов списка = имя параметра;

область допустимых значений = ‘[‘ число ‘,’ число ‘]’ | число {число} |

строка {строка};

операция = базовая операция | системная операция | научный рабочий процесс;

базовая вычислительная операция = имя операции ‘,’ список параметров операции ‘,’ имя модуля;

базовая логическая операция = имя операции ‘,’ список параметров операции ‘,’ список логический параметров операции ‘,’ имя модуля;

базовая операция обработки параллельного списка данных = имя операции ‘,’ параллельный список данных ‘,’ параллельный список данных;

базовая операция = базовая вычислительная операция | базовая логическая операция | базовая операция обработки параллельного списка данных

операция дезагрегирования = имя операции ‘,’ параллельный список данных ‘,’ список экземпляров базового параметра;

операция агрегирования = имя операции ‘,’ параллельный список данных ‘,’ список экземпляров базового параметра;

операция параллельного выполнения экземпляров базовой операции = имя базовой операции ‘,’ параллелельный список данных;

операция последовательного выполнения = базовая операция ‘,’ базовая операция;

операция параллельного выполнения = базовая операция ‘,’ базовая операция;

операция for = базовая операция ‘,’ базовая операция ‘,’ количество итераций;

количество итераций = число;

операция dowhile = базовая операция ‘,’ базовая операция ‘,’ логический параметр ‘,’ допустимое число итерации;

операция параллельного цикла = базовая операция ‘,’ количество итераций;

операция ветвления = базовая операция ‘,’ базовая операция ‘,’ логическое выражение;

операция явного многометодного анализа = список базовых операций ‘,’ число базовых операций ‘,’ число параметров;

операция неявного многометодного анализа = список базовых операций ‘,’ число базовых операций ‘,’ число параметров;

число параметров = число;

допустимое число итерации = число;

имя операции = строка;

системная операция = имя операции ‘,’ список параметров операции ‘,’ имя модуля;

список параметров операции = имя параметра ‘,’ назначение параметра {имя параметра ‘,’ назначение параметра};

продукция = имя продукции ‘,’ логическое выражение ‘,’ имя операции;

имя продукции = строка;

логическое выражение = имя параметра {логический оператор имя параметра};

логический оператор = ‘∧’ | ‘∨’ | ‘ ¯ ’;

постановка задачи = [список исходных параметров] ‘,’ [список целевых параметров] ‘,’ [блок операторов] ‘,’ [список критериев эффективности решения задачи] ‘,’ [список критериев эффективности использования ресурсов];

список исходных параметров = имя параметра {имя параметра};

список целевых параметров = имя параметра {имя параметра};

список критериев эффективности решения задачи = имя системного параметра оператор число {имя системного параметра оператор число};

список критериев эффективности использования ресурсов = имя системного параметра оператор число {имя системного параметра оператор число};

научный рабочий процесс = имя научного рабочего процесса ‘,’

список исходных параметров ‘,’
список целевых параметров ‘,’
блок операторов;

блок операторов = оператор структурного программирования
{оператор структурного программирования};

оператор структурного программирования = оператор выполнения операции |
оператор ветвления |
оператор цикла |
оператор дезагрегирования |
оператор агрегирования |
оператор динамического планирования;

вычислительный ресурс = имя вычислительного ресурса ‘,’ список характеристик;

имя вычислительного ресурса = строка;

список характеристик = характеристика {характеристика};

характеристика = системный параметр;

административная политика = список ограничений;

список ограничений = ограничение {ограничение};

ограничение = имя системного параметра оператор сравнения число | строка;

вычислительная история = задание ‘,’ имя системного параметра;

оператор сравнения = ‘=’ | ‘≠’ | ‘>’ | ‘<’ | ‘≥’ | ‘≤’;

строка = буква {буква | цифра | специальный символ};

число = цифра {цифра};

буква = ‘A’ | ‘B’ | … | ‘Z’ | ‘a’ | ‘b’ | … | ‘z’;

цифра = ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’;

специальный символ = ‘_’;

Приложение Д. Таблица соответствия конструкций спецификации НРП конструкциям языка BPEL и WSDL

Таблица Д.5 – Таблица соответствия конструкций спецификации НРП конструкциями языка BPEL и WSDL

Конструкция спецификации НРП	Конструкции языков BPEL и WSDL
<paramList name=«имя списка параметров»> <param name=«имя параметра 1» type= «тип параметра 1»/> <param name= «имя параметра 2» type= «тип параметра 2»/> … <param name= «имя параметра n» type= «тип параметра n»/> </paramList>	Код BPEL: <variable name=«имя списка параметров» messageType=«message»+«имя списка параметров» /> Код WSDL: <message name=«message»+«имя списка параметров»> <part name=«имя параметра 1» type=«тип параметра 1»/> <part name=«имя параметра 2» type=«тип параметра 2»/> … <part name=«имя параметра n» type=«тип параметра n»/> </message>
<operation name= «имя операции» service= «имя сервиса» inputParamList= «имя списка параметров» outputParamList=«имя списка параметров»/>	Код WSDL: <portType name=«имя сервиса»+«PT»> <operation name=«имя операции»> <input message=«inputMessage»+«имя списка параметров»/> <output message=«outputMessage»+«имя списка параметров»/> </operation> </portType>
<service name=«имя сервиса» url=«URL сервиса»/>	Код BPEL: <partnerLink name= «имя сервиса» partnerLinkType=«имя сервиса»+«LT» myRole= «имя сервиса»+«Requester» partnerRole= «имя сервиса»+«Service» /> Код WSDL: <partnerLinkType name=«имя сервиса»+«LT»> <role name=«имя сервиса»+«Service» portType= «имя сервиса»+ «PT»/> <role name=”имя сервиса»+«Requester» portType=«имя сервиса»+ «CallbackPT» /> </partnerLinkType> <binding name=«имя сервиса»+«Binding» type=«имя сервиса»+ «PT»> <soap:binding style=”rpc” transport=”http://schemas.xmlsoap.org/soap/http”/> </binding>
<module name=«имя модуля» service= «имя сервиса»/>	Код WSDL: <service name=«Service»> <port name=«ServicePort» binding= «имя сервиса»+ «Binding»> <soap:address location=«URL-адрес service»+«имя модуля»> </port> </service>
<task name=«имя задачи» inputParamsList= «имя списка параметров» outputParamList= «имя списка параметров» />	Код BPEL: <receive partnerLink=«client»+«PL» portType=«имя сервиса»+«Port» operation=«имя задачи» variable=« имя списка параметров» createInstance= «yes»> <flow>…<flow> <reply partnerLink=«client»+«PL» portType=«имя задачи»+«PT» operation=«имя задачи» variable==« имя списка параметров»> </reply>

Таблица Д.5 – Таблица соответствия конструкций спецификации НРП конструкциями языка BPEL и WSDL (продолжение)

<if condition=“логическое выражение”> «список инструкций» <elseif condition=“логическое выражение”> «список инструкций» </elseif> <else/> «список инструкций» </if>	Код BPEL: <if standard-attributes> <condition> «логическое выражение» </condition> «список инструкций» <elseif> <condition> «логическое выражение» </condition> «список инструкций» </elseif> <else> «список инструкций» </else> </if>
<dowhile condition=”логическое выражение”> «список инструкций» </dowhile>	Код BPEL: <repeatUntil > «список инструкций» <condition>«логическое выражение»</condition> </repeatUntil>
<foreach counterName=”имя счетчика цикла” start=”начальное значение счетчика цикла” final “конченое значение счетчика цикла” parallel=”False”> «список инструкций» </foreach>	Код BPEL: <forEach counterName="имя счетчика цикла" parallel="no"> <startCounterValue>«начальное значение счетчика цикла» </startCounterValue> <finalCounterValue>«конечное значение счетчика цикла» </finalCounterValue> <scope> «список инструкций» </scope> </forEach>
<foreach counterName=”имя счетчика цикла” start=”начальное значение счетчика цикла” final “конченое значение счетчика цикла” parallel=”True”> «список инструкций» </foreach>	Код BPEL: <forEach counterName="имя счетчика цикла" parallel="yes"> <startCounterValue>«начальное значение счетчика цикла» </startCounterValue> <finalCounterValue>«конечное значение счетчика цикла» </finalCounterValue> <scope> «список инструкций» </scope> </forEach>
<execute name= “имя операции”/>	Код BPEL: <invoke name=«имя операции»+ «Invoke» partnerLink=«имя сервиса» portType=«имя сервиса»+PT operation=«имя операции» inputVariable= «имя списка параметров» outputVariable= «имя списка параметров»> </invoke> <receive name=«имя операции»+ «Receive» partnerLink= «имя сервиса» portType= «имя сервиса»+CallbackPT operation= «имя операции» " variable= «имя списка параметров»> </receive>

Приложение Е. Таблица соответствия конструкций спецификации НРП конструкциям языка Python

Конструкция спецификации НРП	Конструкции языка Python
<paramList name=«имя списка параметров»> <param name=«имя параметра 1» type=«тип параметра 1»/> <param name=«имя параметра 2» type=«тип параметра 2»/> … <param name=«имя параметра n» type=«тип параметра n»/> </paramList>	«имя параметра 1» = Param(type=«тип параметра»); «имя параметра 2» = Param(type=«тип параметра»); … «имя параметра n» = Param(type=«тип параметра»); «имя списка параметров» = ParamList([«имя параметра 1», «имя параметра 2»,…, «имя параметра n»]);
<module name=«имя модуля» paramList=«имя списка параметров»/>	«имя модуля» = Module(params=«имя списка параметров»);
<operation name=«имя операции» target=«имя модуля/имя сервиса» inputParamList=«имя списка параметров» outputParamList=«имя списка параметров»/>	«имя модуля/имя сервиса» = Operation(inputs=«имя списка параметров», outputs=«имя списка параметров», module=«имя модуля»);
<service name=«имя сервиса» paramList=«имя списка параметеров» url=«URL сервиса»/>	«имя сервиса» = Service(params=«имя список параметров», «URL сервиса»);
<task name=«имя задачи” inputParamsList=«имя списка параметров» outputParamList=«имя списка параметров»/>	«имя задачи» = Task(inputs=«имя списка параметров», outputs=«имя списка параметров»);

<if condition=«логическое выражение»>
«список инструкций»
<elseif condition=«логическое выражение»>
«список инструкций»
</elseif>
<else>
«список инструкций»
</else>
</if>
	if «логическое выражение»:
	«список инструкций»
	elseif «логическое выражение»:
	«список инструкций»
	else:
	«список инструкций»

<dowhile condition=«логическое выражение»>
«список инструкций»
</dowhile>
	while True:
	«список инструкций»
	if «логическое выражение» == True:
	break

<foreach counterName=”имя счетчика цикла” start=”начальное значение счетчика цикла” final “конченое значение счетчика цикла” parallel=”False”> «список инструкций» </foreach>	for «счетчик цикла» in range(«начальное значение счетчика цикла», «конечное значение счетчика цикла»): «список инструкций»
<foreach counterName=”имя счетчика цикла” start=”начальное значение счетчика цикла” final “конченое значение счетчика цикла” parallel=”True”> «список инструкций» </foreach>	from multiprocessing import Pool … def process_item_”имя счетчика цикла” (item): «список инструкций» index = range(«начальное значение счетчика цикла», «конечное значение счетчика цикла») with Pool() as pool: results = pool.map(process_item_”имя счетчика цикла”, index)
<execute name=«имя операции»/>	«имя операции»();

Приложение Ж. Пример Playbook для развертывания central_manager

```
- name: Настроить кластер HTCondor
  hosts: all
  become: yes
  gather_facts: yes
  vars:
    central_manager_ip: "192.168.56.110"
    condor_cluster_password: "qwerty"
    condor_config_dir: "/etc/condor"
  tasks:
    - name: Обновить кэш apt
      apt:
        update_cache: yes
    - name: Установить пакет HTCondor
      apt:
        name: htcondor
        state: present
    - name: Создать каталоги Condor с нужными правами
      file:
        path: "{{ item }}"
        state: directory
        owner: condor
        group: condor
        mode: "0755"
      loop:
        - "{{ condor_config_dir }}"
        - "{{ condor_config_dir }}/config.d"
        - "{{ condor_config_dir }}/passwords.d"
        - "/var/lib/condor/execute"
        - "/var/lib/condor/spool"
```

```
    - "/var/lib/condor/run"
    - "/var/lib/condor/lock"
    - "/var/log/condor"
- name: Установить пароль пула (владелец condor, 0600)
  copy:
    content: "{{ condor_cluster_password }}"
    dest: "{{ condor_config_dir }}/passwords.d/POOL"
    owner: condor
    group: condor
    mode: "0755"
  notify: Перезапустить condor
- name: Создать базовую конфигурацию путей
  copy:
    dest: "{{ condor_config_dir }}/config.d/00-base.conf"
    owner: condor
    group: condor
    mode: "0644"
    content: |
      LOCAL_DIR = /var/lib/condor
      LOG = /var/log/condor
      EXECUTE = /var/lib/condor/execute
      SPOOL = /var/lib/condor/spool
      RUN = /var/lib/condor/run
      LOCK = /var/lib/condor/lock
  notify: Перезапустить condor
- name: Настроить центральный менеджер
  template:
    src: central_manager.j2
    dest: "{{ condor_config_dir }}/condor_config"
    owner: condor
```

```
      group: condor
      mode: "0644"
    when: "'central_manager' in group_names"
    notify: Перезапустить condor
  - name: Настроить вычислительные узлы
    template:
      src: execute_node.j2
      dest: "{{ condor_config_dir }}/condor_config"
      owner: condor
      group: condor
  - name: Проверить конфигурацию HTCondor (dry run)
    command: condor_config_val -summary
    register: condor_cfg_chk
    changed_when: false
  - name: Включить и запустить сервис HTCondor
    systemd:
      name: condor
      enabled: yes
      state: started
handlers:
  - name: Перезапустить condor
    systemd:
      name: condor
      state: restarted
```