Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/05 Современные интеллектуальные программно-аппаратные комплексы**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ МАГИСТРА НА ТЕМУ:

Интеллектуальная система распределения нагрузки в вычислительном кластере

Студент	ИУ6-43М		В.Д. Шульман
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Руководитель			О.Ю. Ерёмин
		(Подпись, дата)	(И.О. Фамилия)
Нормоконтролер			Ю.И. Бауман
		(Подпись, дата)	(И.О. Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ___ИУ6___

_____ А.В. Пролетарский

« __ » _____ 2022 г.

З А Д А Н И Е
на выполнение выпускной квалификационной работы магистра

Студент группы _____ИУ6-43М_____

_____Шульман Виталий Дмитриевич_____
(Фамилия, имя, отчество)

Тема квалификационной работы ___«Интеллектуальная система распределения нагрузки в вычислительном кластере»_____

_____

_____

Источник тематики (НИР кафедры, заказ организаций и т.п.)

_____НИР кафедры_____

_____

Тема квалификационной работы утверждена распоряжением по факультету ИУ № _03.02.01 - 04.03/17_от « _11_ » _ноября_ 2021 г.

Часть 1. Исследовательская

__Проанализировать актуальные методы и современные технологии распределения нагрузки в вычислительных кластерах. Оценить актуальность и критичность проблемы балансировки нагрузки в распределенных вычислительных системах. Провести сравнительный анализ существующих прикладных алгоритмов распределения нагрузки в вычислительных кластерах. Проанализировать возможность повышения эффективности работы прикладных алгоритмов распределения нагрузки путем их модификации. Исследовать возможность повышения эффективности работы прикладных алгоритмов за счет оптимизирующего мета-алгоритма._____

Часть 2. Конструкторская

Провести анализ требований к интеллектуальной информационной системе. Обеспечить модульность и потенциал масштабирования интеллектуальной системы. Осуществить выбор наиболее подходящих технологий для реализации программных компонентов системы. Спроектировать программную модель вычислительного кластера. Реализовать алгоритмы распределения нагрузки по узлам вычислительного кластера. Описать варианты использования. Изобразить концептуальную модель предметной области. Спроектировать базу данных. Спроектировать формы интерфейса системного администратора. Построить диаграммы вариантов использования и состояний интерфейса.

Часть 3. Технологическая

Проработать технологию и методологию разработки интеллектуальной информационной системы. Разработать технологию отладки и тестирования разрабатываемого программного обеспечения. Проработать компоненты аудита и логирования системы. Обеспечить надежное хранение данных в системе и их восстановление в случае сбоев. Реализовать возможность гибкой конфигурации системы. Реализовать в программном интерфейсе интеллектуальной информационной системы компоненты для интеграции с другими программами и системами.

Оформление квалификационной работы:

Расчетно-пояснительная записка на 95–105 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Схема структурная информационной системы. Схема функциональная программного обеспечения. Схемы (модели) процессов (методов формирования результатов, механизмы выводов и т.п.). Диаграмма вариантов использования. Концептуальная модель предметной области. Схемы структурные компонент, даталогическая и инфологическая схемы базы данных. Схема взаимодействия модулей. Граф (диаграмма) состояний интерфейса. Формы интерфейса. Схема процесса разработки программного продукта.

Дата выдачи задания « 1 » сентября 2021 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до « 1 » июня 2022 г.

Руководитель квалификационной работы	(Подпись, дата)	О.Ю. Ерёмин (И.О. Фамилия)
Студент	(Подпись, дата)	В.Д. Шульман (И.О. Фамилия)

Примечание:1. Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления УТВЕРЖДАЮ

КАФЕДРА Компьютерные системы и сети Заведующий кафедрой ИУ6

ГРУППА ИУ6-43М _____ А.В. Пролетарский

« _____ » _____ 2022 г.

КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы магистра

студента: Шульмана Виталия Дмитриевича
(фамилия, имя, отчество)

Тема квалификационной работы «Интеллектуальная система распределения нагрузки в вычислительном кластере»

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	09.2021 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
2.	1 часть Исследовательская	12.2021 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	02.2022 Планируемая дата		Заведующий кафедрой	А.В. Пролетарский
4.	2 часть Конструкторская	05.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
5.	3 часть Технологическая	05.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
6.	1-я редакция работы	05.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
7.	Подготовка доклада и презентации	06.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
8.	Заключение руководителя	06.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
9.	Нормоконтроль	06.2022 Планируемая дата		Нормоконтролер	
10.	Внешняя рецензия	06.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин
11.	Защита работы на ГЭК	06.2022 Планируемая дата		Руководитель ВКР	О.Ю. Ерёмин

Студент _____ (подпись, дата)

Руководитель работы _____ (подпись, дата)

АННОТАЦИЯ

В настоящей выпускной квалификационной работе магистра описан процесс разработки системы моделирования искусственной жизни с использованием цифровых автоматов.

Произведен анализ и классификация существующих подходов к использованию генетических алгоритмов. Были рассмотрены дискретные автоматы, как одно из возможных средств при реализации генетического алгоритма, что и было сделано в данной работе.

Разработаны технологии задания параметров, отправки данных на удаленный сервер, тестирование системы. Разработан интерфейс программного обеспечения.

ABSTRACT

В настоящей выпускной квалификационной работе магистра

РЕФЕРАТ

Расчётно-пояснительная записка с. 100, рис. 42, табл. 13, источников 25, приложений 4.

ГЕНЕТИЧЕСКИЙ АЛГОРИТМ, СЕЛЕКЦИОННЫЙ МЕТОД, ОТБОР, АГЕНТ, ХРАНИЛИЩЕ ГЕНОВ, КОНЕЧНЫЙ АВТОМАТ, КЛЕТОЧНЫЙ АВТОМАТ, ОКРЕСТНОСТЬ КЛЕТОЧНОГО АВТОМАТА, ЭВОЛЮЦИОНИРУЮЩИЙ КЛЕТОЧНЫЙ АВТОМАТ

Рассматриваемыми объектами в данной работе являются.

Целью работы является

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Вычислительный кластер — массив вычислительных узлов и хранилищ данных, связанных сетью, которые представляют комплексам программ вычислительные ресурсы для выполнения поставленных перед ними задач

Программный комплекс — совокупность программ, программных систем и подсистем, которые взаимодействуют для решения конечного набора вычислительных задач

Распределенная информационная система — совокупность взаимодействующих программных подсистем, каждая из которых может рассматриваться как программный модуль, исполняемый в рамках отдельного процесса

Интеллектуальная система — это система с целью, обладающая возможностью накапливать и продуцировать новые знания, способная производить логические заключения подобно человеку

Искусственный интеллект — это научное направление, связанное с автоматизированной обработкой информации, представленной в виде знаний

Знание — это совокупность фактов, закономерностей и эвристических правил, на базе которых возможно осуществить процесс логического вывода

Микросервис — небольшой программный компонент, решающий одну определенную задачу

Балансировка нагрузки — процесс распределения задач между вычислительными сервисами с целью оптимизации использования ресурсов, сокращения времени выполнения, горизонтального масштабирования кластера, а также обеспечения отказоустойчивости.

Предметная область базы данных — часть реального мира, о которой база данных хранит, собирает и анализирует информацию

Система управления базами данных — совокупность программных средств, обеспечивающих управление, создание и использование БД

Go — императивный язык программирования

SQL — декларативный язык программирования

PostgreSQL — свободная объектно-реляционная СУБД

Шардирование — метод горизонтального масштабирования записи

Репликация — метод горизонтального масштабирования чтения

Эволюционное моделирование — подход, в основе которого лежат принципы и понятийный аппарат, заимствованный из популяционной генетики, а также ряд компьютерных методов

Генетический алгоритм — метод эволюционного моделирования, построенный на принципах естественного отбора и генетической рекомбинации

Целевая функция — функция нескольких переменных, подлежащая оптимизации в целях решения некой задачи

Агент — эволюционная единица, представляющая одно конкретное решение задачи из всего множества возможных

Ген — элементарная структурная единица агента, подвергающаяся модификации в ходе процесса эволюционного моделирования

Поколение — совокупность агентов на конкретной итерации работы генетического алгоритма

Генотип — пространство поиска решений

Фенотип — совокупность найденных решений

ПЕРЕЧЕНЬ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

ГОСТ — государственный стандарт

ВКРМ — выпускная квалификационная работа магистра

НИР — научно-исследовательская работа

РПЗ — расчётно-пояснительная записка

ВК — вычислительный кластер

ЭВМ - — электронно-вычислительная машина

БД — база данных

СУБД — система управления базами данных

ПрК — программный комплекс

ПП — программный продукт

ИС — информационная система

ИТ — информационная технология

ИИС — интеллектуальная информационная система

ИИ — искусственный интеллект

АС — автоматизированная система

СА — система аудита

УЗ — учётная запись

ТУЗ — техническая учётная запись

ЦПУ — центральное процессорное устройство

ПЗУ — постоянно-запоминающее устройство

ОЗУ — оперативно-запоминающее устройство

ЭМ — эволюционное моделирование

ЭП — эволюционный процесс

ГА — генетический алгоритм

АБН — алгоритм балансировки нагрузки

ГК — генетический код

ЦФ — целевая функция

ПМ — программная модель

ММ —математическая модель

DNS — система доменных имён

IP — уникальный сетевой идентификатор

TCP — сетевой протокол транспортного уровня

ISO — международная организация по стандартизации

OSI — модель взаимодействия открытых систем

L4 — транспортный уровень модели OSI

L7 — прикладной уровень модели OSI

URL — унифицированный указатель ресурса

LAN — локальная вычислительная сеть

ВВЕДЕНИЕ

Вычислительный кластер – это множество вычислительных узлов, которые объединены сетью и функционируют как единое целое. Вычислительные кластеры могут быть как однородные (состоять из узлов идентичной конфигурации), так и не однородные (состоять из узлов различной конфигурации и, как следствия, узлов разных мощностей) [1].

Как правило, кластерные системы применяются при решении сложных вычислительных задач. Активное применение кластерные системы находят в решении задач моделирования и Data Mining. Анализ больших объемов данных требует высокой скорости их обработки. В связи с этим задачи анализа данных могут также решаться при помощи вычислительных кластеров.

Кластеры стали фактическим стандартом в области высокопроизводительных вычислений. Можно с большой долей уверенности сказать, что этот подход будет актуален всегда: сколь бы совершенен не был один компьютер, кластер из узлов такого типа справится с любой задачей гораздо быстрее [1].

Одна из основных задач, решаемая в рамках эксплуатации вычислительных кластеров, — это распределение (балансировка) нагрузки между их узлами. Нагрузка между ними распределяется при помощи комплекса специальных методов. Эффективность кластеризации напрямую зависит от того, как распределяется нагрузка между элементами кластерах [2].

Балансировка нагрузки может осуществляться при помощи как аппаратных, так и программных инструментов. Технологии балансировки нагрузки активно развиваются и представляют сейчас большой интерес с точки зрения IT-отрасли.

1 Исследование проблемы распределения нагрузки в вычислительном кластере и путей её решения

1.1 Понятие вычислительного кластера и пути его масштабирования

Функционирование вычислительных систем строится на основе двух компонентов — физического вычислительного кластера и программного комплекса.

Программный комплекс представляет собой совокупность программ, программных систем и подсистем, которые взаимодействуют для решения конечного набора вычислительных задач. Для функционирования программных комплексов необходимы вычислительные ресурсы в виде электронно-вычислительной техники.

Вычислительный кластер является массивов вычислительных узлов и хранилищ данных, связанных сетью, которые представляют комплексам программ вычислительные ресурсы для выполнения поставленных перед ними задач (рис. 1).

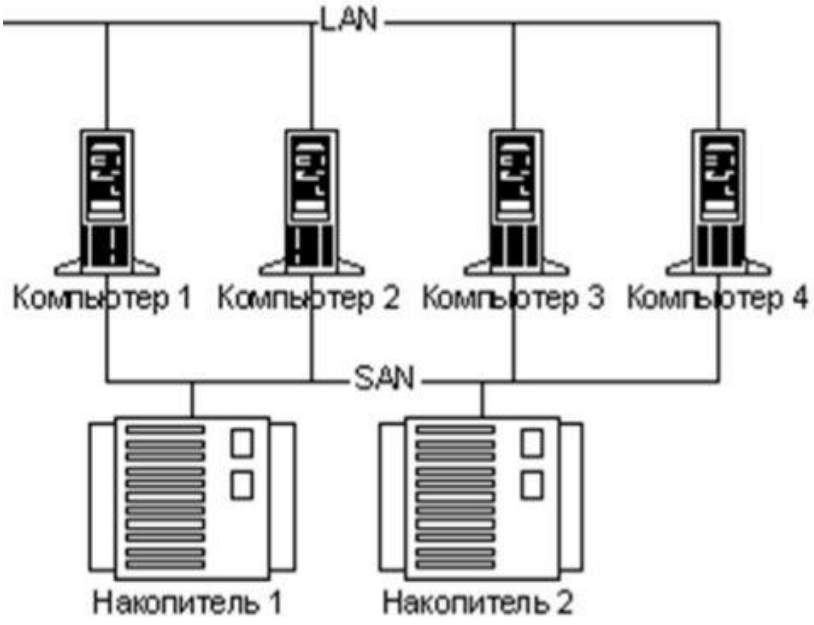

Рисунок 1 – Вычислительный кластер

Обычно к вычислительным кластерам предъявляются следующие требования:

– доступность;

– быстродействие;

– масштабируемость.

Кластеры могут иметь различную организационную структуру [1]. По физической реализации кластеры бывают:

– Специализированные кластеры (суперкомпьютеры). Используются для решение особо трудоемких в плане вычислительных ресурсов задач. Для организации таких кластеров используется специальные программно-аппаратные решения.

– Кластеры на основе локальных сетей из персональных компьютеров. Для организации работы таких кластеров достаточно обеспечить доступность между узлами и установить дополнительное программное обеспечение. Такие кластеры не годятся для использования в реальных проектах, но могут применяться для обучения.

– Кластеры на основе виртуальных машин или контейнеров. Все компоненты программной системы запускаются на виртуальных машинах или контейнерах, а сам физический кластер только предоставляет ресурсы для функционирования этих виртуальных сред. Кластер может быть не однородным и состоять из вычислительных узлов разных мощностей.

– Кластеры на основе физических серверов. Кластер представляет из себя набор серверов, которые могут взаимодействовать между друг-другом через сеть. Каждый узел такого кластера является самостоятельной вычислительной машиной. Также допускается гетерогенность такого кластера.

Кроме структуры вычислительные кластеры можно классифицировать по однородности [1]:

– Гомогенные. Все узлы кластера имеют одинаковую конфигурацию (ЦП, ОЗУ, ПЗУ). В случае гомогенного кластера задача распределения нагрузки между вычислительными узлами значительно упрощается.

– Гетерогенные. В качестве узлов кластера могут выступать любые вычислители, что может приводить к неравномерности загрузки узлов и проблеме распределения вычислительных задач.

Также среди вычислительных кластеров выделяют по архитектурному признаку вертикальные и горизонтальные архитектуры [3]. Сравнительный анализ этих архитектур представлен в таблице 1.

Таблица 1 – Сравнение горизонтальных и вертикальных архитектур

Параметры	**Вертикальные системы**	**Горизонтальные системы**
Память	Совместно используемая	Выделенная
Потоки исполнения	Множество взаимосвязанных потоков	Множество независимых потоков
Соединение	Высокопроизводительная шина	Стандартные сетевые технологии
Количество ОС	Одна копия ОС для всех центральных процессоров	Отдельная ОС для каждого вычислительного узла со своими ЦП
Компоновка	В одном шкафу	Могут находиться на разных континентах
Оборудование	Обычное или специальное высокопроизводительное	Обычное
Масштабирование	Установка более мощных компонентов	Добавление дополнительных узлов

Наращивать мощность вычислительного кластера можно экстенсивным или интенсивным путем.

Под экстенсивным ростом производительности вычислительного кластера будем понимать увеличение количества рабочих узлов.

В случае интенсивного способа наращивания мощности кластера будем понимать увеличение количества ядер, частоты процессора, объема постоянной и оперативной памяти и т.п. в рамках отдельных узлов вычислительной системы.

В области информационных технологий также распространены понятия вертикального и горизонтального масштабирования, которые применяются не к самому физическому вычислительному кластеру, а к программам и программным системам, которые на них эксплуатируются [4].

Несмотря на это принципиальное различие приведенных понятий, терминология горизонтального и вертикального масштабирования также довольно часто применяется в контексте производительности вычислительных кластеров (рис. 2).

Рисунок 2 – Вертикальное и горизонтальное масштабирование кластера

С интенсивным путем наращивания мощностей со временем все больше проблем. Такой вид масштабирования весьма удобен с точки зрения обслуживания и администрирования кластера. Он подразумевает замену компонентов уже сконфигурированной в кластере машины более мощными. В случае с памятью (как ПЗУ, так ОЗУ) проблем действительно не возникает. Узким местом при данном подходе является процессор. Здесь присутствуют как финансовые препятствия (рост стоимости не пропорционален

производительности), так и технологические (существует потолок частоты и количества ядер процессора).

Финансовое препятствие возникает вследствие усложнения техпроцесса при производстве чипов с транзисторами крайне малых размеров. На данный момент один транзистор с нормой 22 нм стоит дороже, чем транзистор, выполненный на норме 28 нм, что является прямым противоречием для всеми известного закона Мура [5–7].

Технологическое препятствие представляет собой совокупность факторов физического характера. Тут можно выделить проблемы литографии, туннельного эффекта, возможности пробоя. Даже если бы эти проблемы были решены, то в конце концов возникла бы проблема более фундаментального характера — не представляется возможным сделать транзистор размером с пару атомов.

Добавление нового узла не является простым процессом. Возможность такого хода должна быть предусмотрена как работающим на кластере программным обеспечением, так и архитектурой самого кластера. Для горизонтальной масштабируемости кластера должны решаться следующие задачи:

– распределение (балансировка) нагрузки;

– обеспечение отказоустойчивости и доступности;

– обеспечение возможности репликации и шардирования данных;

Для распределения вычислительной нагрузки используются балансировщики сетевого трафика на различных уровнях (прикладном, транспортном или сетевом), а также брокеры сообщений (например, Apache Kafka или RabbitMQ) [8]. В совокупности брокеры сообщений и балансировщики трафика позволяют равномерно распределять нагрузку на узлы вычислительной системы независимо от количества этих узлов (рис. 3).

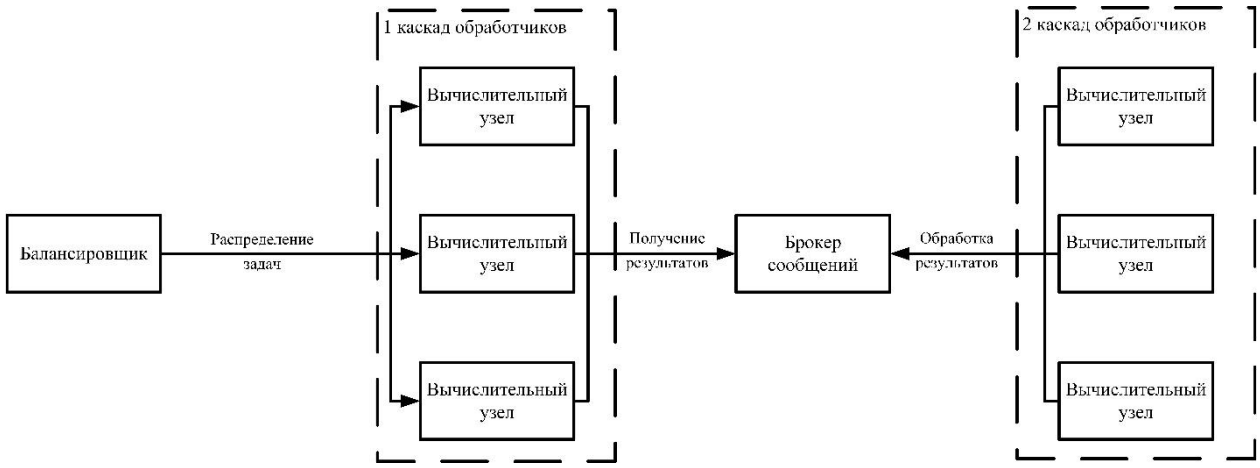

Рисунок 3 – Комбинирование балансировщика и брокера сообщений

Для обеспечения доступности и отказоустойчивости на программном уровне обычно применяются системы оркестрации (например, Kubernetes). При таком подходе отдельные компоненты информационной системы, работающие на кластере, можно рассматривать как программные master-slave и multimaster решения со встроенными механизмами отказоустойчивости.

Для распределенных вычислений и хранения данных на кластерах используются целые технологические стеки, например, Hadoop (рис. 4).

Рисунок 4 – Технологический стек Hadoop.

Совокупность программных компонентов Hadoop вместе с технологиями виртуализации и контейнеризации даёт возможность ещё сильнее абстрагироваться от физической конфигурации вычислительного кластера.

Экстенсивный путь развития вычислительных кластеров весьма перспективен.

Одним из факторов является то, что потенциал интенсивного наращивания мощностей ограничен, а рост стоимости непропорционален.

Вторым основным фактором можно выделить наличие и дальнейшее развитие информационных технологий, позволяющих эффективнее эксплуатировать гетерогенные вычислительные кластеры с большим количеством узлов.

1.2 Задача распределения нагрузки в кластере и методы её решения

Балансировка нагрузки осуществляется при помощи целого комплекса подходов и методов, соответствующим следующим уровням модели OSI:

– сетевой;

– транспортный;

– прикладной.

В случае балансировки на сетевом уровне предполагается, что задача распределения нагрузки осуществляются посредством IP-адресов. Подразумевается, что за один конкретный IP-адрес отвечает сразу несколько физических машин. Данный вид балансировки может осуществляться с помощью следующих методов:

– DNS-балансировка;

– NLB-балансировка;

– балансировка с помощью дополнительных маршрутизаторов;

– балансировка по территориальному признаку.

При балансировке на транспортном уровне распределение нагрузки осуществляется через так-называемые прокси-серверы. В отличии от сетевого уровня, где идет простое перенаправление запроса, в случае балансировки на транспортном уровне прокси-сервер выступает в качестве посредника и может добавлять в запрос дополнительные заголовки.

В случае, если балансировка нагрузки осуществляется на прикладном уровне, сервера-посредники, распределяющие нагрузку, работают как «умные прокси-серверы». Балансировщики нагрузки прикладного уровня анализируют содержимое клиентских запросов и перенаправляют их на различные серверы системы в зависимости от целевого контента и типа операций с ним. Наиболее популярные примеры решений для балансировки нагрузки на прикладном уровне – это Nginx и PGpool [2]. Первый используется в составе веб-систем, где необходимо распределять запросы между различными сервисами, а второй в системах баз данных.

Существует много различных алгоритмов и методов балансировки нагрузки. Выбирая конкретный алгоритм, нужно исходить, во-первых, из специфики конкретного проекта, а во-вторых — из целей, которые необходимо достичь.

Из числа целей, которые необходимо достичь в рамках решения задачи балансировка, можно выделить следующие:

– справедливость: гарантируется, что однотипные запросы равноправны и обрабатываются с одинаковой степенью приоритета;

– эффективность (равномерность): не допускается ситуации, когда один сервера загружен на 100%, а другой простаивает;

– минимальное время выполнения: обеспечивается минимально возможный интервал между начало и окончанием обработки запроса;

– минимальное время отклика: обеспечивается минимальный временной интервал между запросом и откликом пользователю.

– детерминируемость: в эквивалентных условиях алгоритм работает одинаково;

– масштабируемость: при росте нагрузки эффективность и стабильность и предсказуемость работы алгоритма не снижается.

Round Robin, или алгоритм кругового обслуживания, представляет собой перебор по кругу: первый запрос передаётся первому серверу, затем

следующий запрос передаётся второму и так до достижения последнего сервера, а затем всё начинается сначала.

Самой распространённой имплементацией этого алгоритма является, конечно же, метод балансировки Round Robin DNS. Как известно, любой DNS-сервер хранит пару «имя хоста — IP-адрес» для каждой машины в определённом домене [2].

В числе несомненных плюсов этого алгоритма следует назвать, во-первых, независимость от протокола высокого уровня. Для работы по алгоритму Round Robin используется любой протокол, в котором обращение к серверу идёт по имени.

Использование алгоритма Round Robin не требует связи между серверами, поэтому он может использоваться как для локальной, так и для глобальной балансировки. Решения на базе алгоритма Round Robin отличаются низкой стоимостью: чтобы они начали работать, достаточно просто добавить несколько записей в DNS.

Алгоритм Round Robin имеет и целый ряд существенных недостатков недостатков. Чтобы распределение нагрузки по этому алгоритму отвечало упомянутым выше критериями справедливости и эффективности, нужно, чтобы у каждого сервера был в наличии одинаковый набор ресурсов (рис. 5). При выполнении всех операций также должно быть задействовано одинаковое количество ресурсов. В современной практике эти условия в большинстве случаев оказываются невыполнимыми [2].

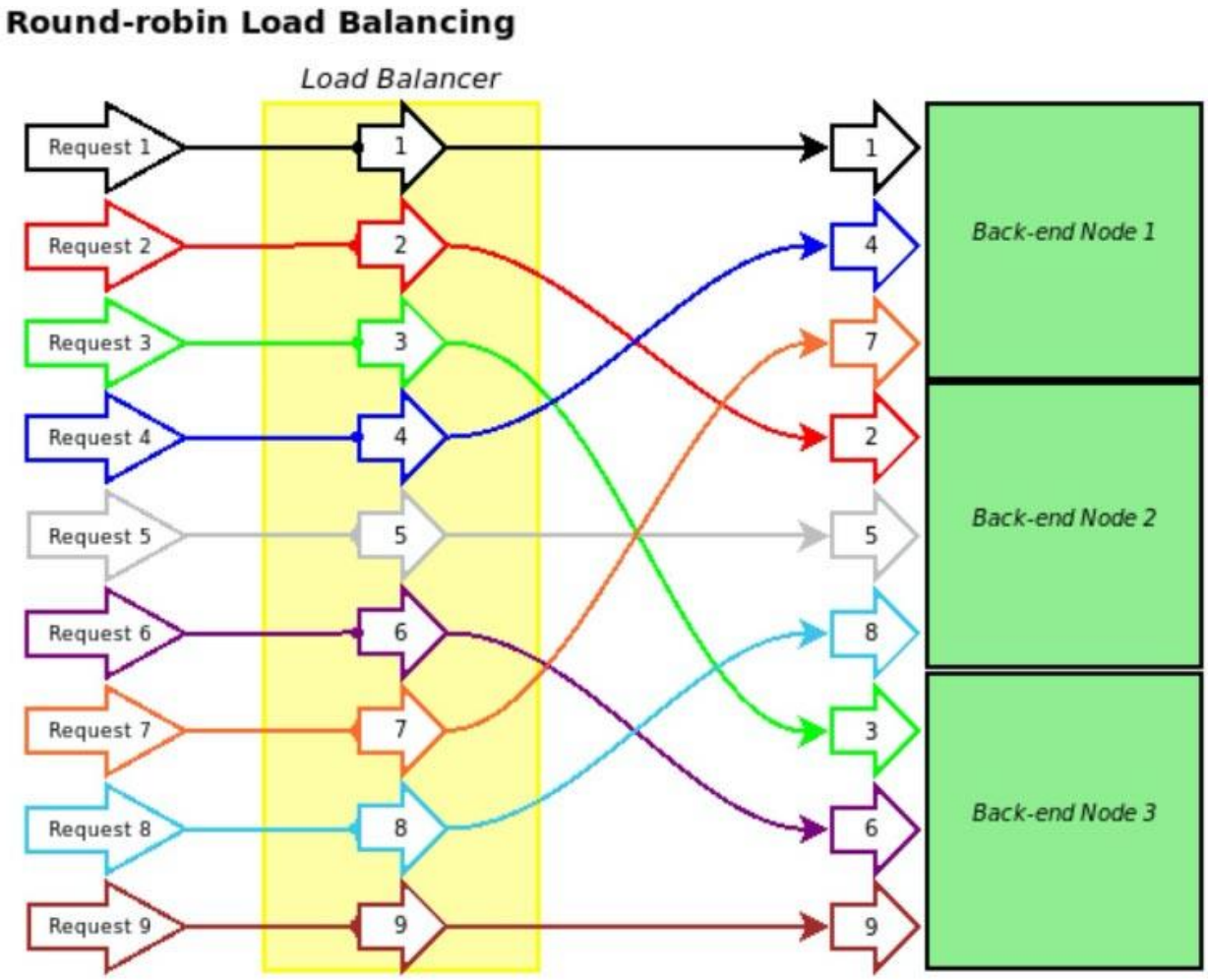

Рисунок 5 – Round Robin.

Weighted Round Robin. Это усовершенствованная версия алгоритма Round Robin. Суть усовершенствований заключается в следующем: каждому серверу присваивается весовой коэффициент в соответствии с его производительностью и мощностью. Это помогает распределять нагрузку более гибко: серверы с большим весом обрабатывают больше запросов. Однако всех проблем с отказоустойчивостью это отнюдь не решает. Более эффективную балансировку обеспечивают другие методы, в которых при планировании и распределении нагрузки учитывается большее количество параметров.

Least Connections. Его особенность в том, что он учитывает количество подключений, поддерживаемых серверами в текущий момент времени. Каждый следующий вопрос передаётся серверу с наименьшим количеством активных подключений (рис. 6).

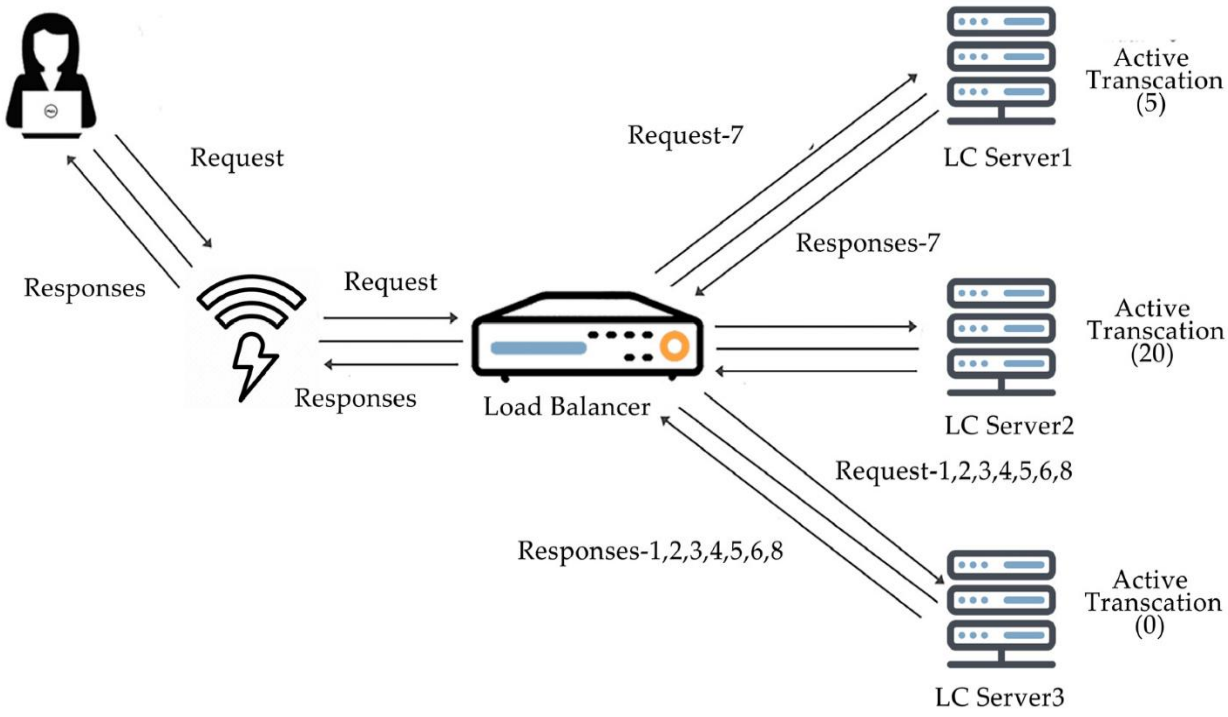

Рисунок 6 – Least Connections.

Существует усовершенствованный вариант этого алгоритма, предназначенный в первую очередь для использования в кластерах, состоящих из серверов с разными техническими характеристиками и разной производительностью. Он называется Weighted Least Connections и учитывает при распределении нагрузки не только количество активных подключений, но и весовой коэффициент серверов [2].

В числе других усовершенствованных вариантов алгоритма Least Connections следует прежде всего выделить Locality-Based Least Connection Scheduling и Locality-Based Least Connection Scheduling with Replication Scheduling.

Первый метод был создан специально для кэширующих прокси-серверов. Его суть заключается в следующем: наибольшее количество запросов передаётся серверам с наименьшим количеством активных подключений. За каждым из клиентских серверов закрепляется группа клиентских IP. Запросы с этих IP направляются на «родной» сервер, если он не загружен полностью. В противном случае запрос будет перенаправлен на другой сервер (он должен быть загружен менее чем наполовину).

В алгоритме Locality-Based Least Connection Scheduling with Replication Scheduling каждый IP-адрес или группа IP-адресов закрепляется не за отдельным

сервером, а за целой группой серверов. Запрос передаётся наименее загруженному серверу из группы. Если же все серверы из «родной» группы перегружены, то будет зарезервирован новый сервер. Этот новый сервер будет добавлен к группе, обслуживающей IP, с которого был отправлен запрос. В свою очередь наиболее загруженный сервер из этой группы будет удалён — это позволяет избежать избыточной репликации.

Можно также выделить 2 интересных алгоритма базирующиеся на идее кэширования. Destination Hash Scheduling и Source Hash Scheduling.

Алгоритм Destination Hash Scheduling был создан для работы с кластером кэширующих прокси-серверов, но он часто используется и в других случаях. В этом алгоритме сервер, обрабатывающий запрос, выбирается из статической таблицы по IP-адресу получателя.

Алгоритм Source Hash Scheduling основывается на тех же самых принципах, что и предыдущий, только сервер, который будет обрабатывать запрос, выбирается из таблицы по IP-адресу отправителя.

Sticky Sessions — алгоритм распределения входящих запросов, при котором соединения передаются на один и тот же сервер группы. Он используется, например, в веб-сервере Nginx. Сессии пользователя могут быть закреплены за конкретным сервером с помощью метода I. С помощью этого метода запросы распределяются по серверам на основе IP-адреса клиента. Метод гарантирует, что запросы одного и того же клиента будет передаваться на один и тот же сервер (рис. 7). Если закреплённый за конкретным адресом сервер недоступен, запрос будет перенаправлен на другой сервер.

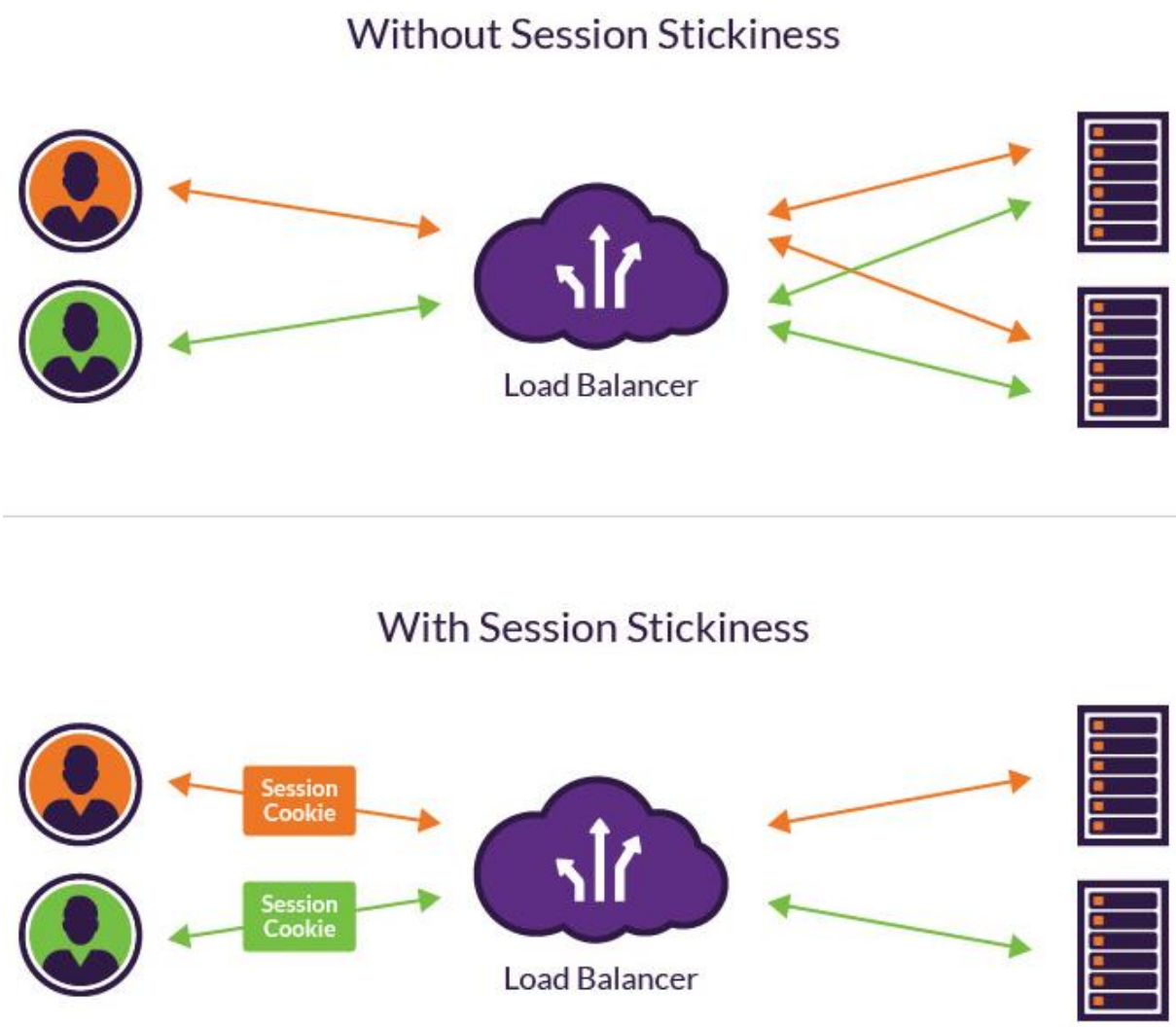

Рисунок 7 – Sticky Sessions.

Применение этого метода сопряжено с некоторыми проблемами. Проблемы с привязкой сессий могут возникнуть, если клиент использует динамический IP. В ситуации, когда большое количество запросов проходит через один прокси-сервер, балансировку вряд ли можно назвать эффективной и справедливой. Описанные проблемы, однако, можно решить, используя cookies. В коммерческой версии Nginx имеется специальный модуль sticky, который как раз использует cookies для балансировки. Есть у него и бесплатные аналоги — например, nginx-sticky-module.

1.3 Эволюционное моделирование как метод оптимизации

Эволюционное моделирование (ЭМ) – это группа эвристических методов, которые заимствуют принципы и понятийный аппарат у популяционной

генетики (рис. 8). Эти методы позволяют осуществлять поиск решения для задач оптимизации [11]. ЭМ эксплуатирует идею «мягких вычислений» [12].

Рисунок 8 – Эволюционное моделирование

Исходя из сказанного выше, можно утверждать, что ЭМ применяется для следующего перечня задач:

1) совершенствования существующих информационных систем за счёт наделения их свойствами адаптивного поведения и самоорганизации;

2) для автоматизации решения оптимизационных задач в различных областях науки и техники;

3) для изучения и моделирования отдельных процессов, по внешним признакам напоминающих процессы в естественной эволюции.

Методы эволюционного моделирования можно рассматривать также как базу для упражнений в совершенствовании техник программирования отдельных алгоритмов и программных систем [11].

Особое место в ЭМ занимает генетический алгоритм (ГА), содержащий все существующие лингвистические конструкции современных языков программирования (включая возможности параллельного программирования) и

одновременно являющийся отправной точкой для создания его модификаций – новых генетических алгоритмов для решения специальных прикладных задач. Сами ГА, наравне с другими методами, также достаточно разнообразны (рис. 9 и).

Рисунок 9 – Виды эволюционного моделирования

Рисунок 10 – Классификация методов эволюционного моделирования

Во время работы ГА выполняется параллельный анализ разных областей пространств решений. ГА способны накапливать и использовать знания об исследованном пространстве поиска, что позволяет им развиваться (обучаться или эволюционировать). Однако, в отличие от машинного обучения, в ГА используется абсолютное значение целевой функции (ЦФ), а не её приращение.

Процесс поиска может продолжаться до тер пор, пока не будут рассмотрены все точки исследуемого пространства. В качестве ограничения могут использоваться критерии оптимума, лимит количества поколений, порог приращения ЦФ.

Естественный отбор моделируется через выполнение процедур селекции, скрещивания и мутации. Качество агента из популяции пропорционально вероятности его перехода полностью (копирование) или частично (в виде потомков) в следующее поколение [11]. Потомки наследуют характеристики родителей с некоторыми изменениями (мутациями). Таким образом, эволюционный процесс подразумевает выполнение 3-х условий [11]:

– наличие наследственной изменчивости как предпосылки эволюции;

– наличие соревновательного аспекта, а также направляющего фактора;

– наличие естественного отбора как преобразующего фактора.

Далее будет использоваться соответствие терминологий из таблицы 1.

Таблица 2 – Соответствие терминов эволюционной и мат. модели

Эволюционная модель	**Математическая модель**
Агент	Решение, объект, строка, последовательность
Ген	Переменная, параметр, характеристика, признак
Генотип	Пространство поиска
Фенотип	Пространство решений
Fitness-функция	Целевая функция
Популяция	Множество решений
Поколение	Итерация работы эволюционного алгоритма

В общем виде последовательность работы генетического алгоритма может быть проиллюстрирована рисунком 5.

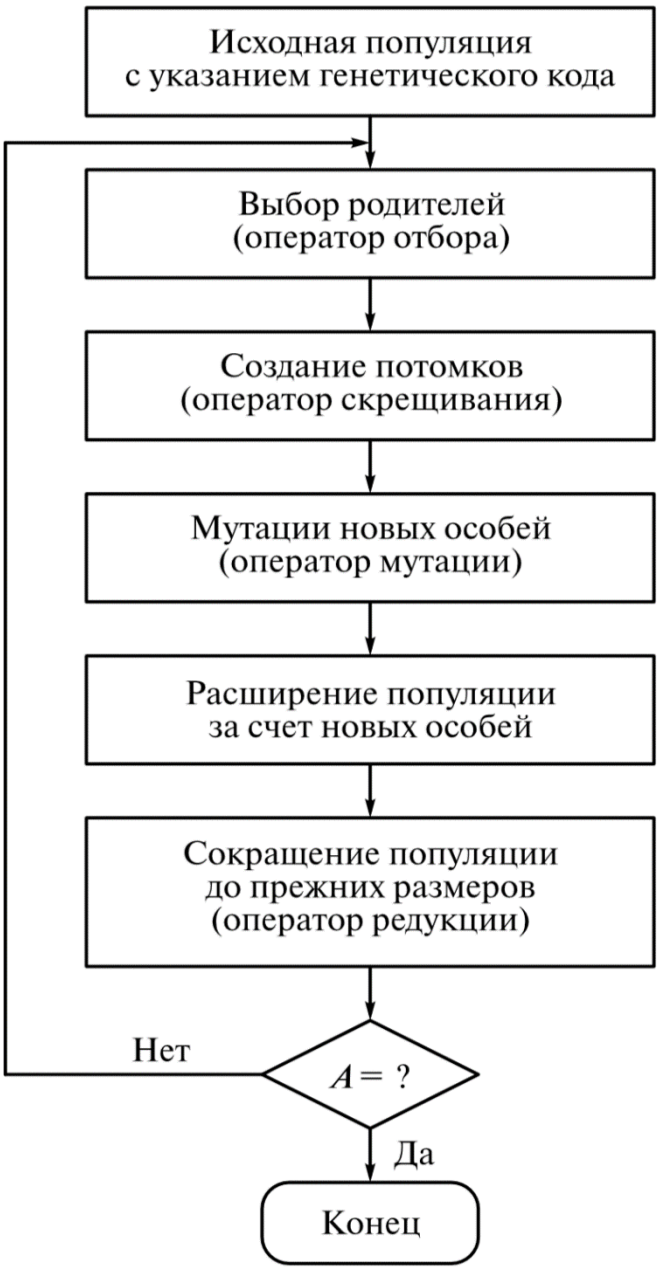

Рисунок 11 – Схема ГА

Приспособленность агента в популяции (качество решения) оценивается с помощью Fitness-функции (ЦФ). Чем выше её значение, тем больше потомков в следующем поколении будет у данного агента. Конечная цель работы всего ГА – это достижение как можно большего значения ЦФ.

Вместе с обычным значением параметров, также часто используются закодированные представления значений параметров задачи. Поиск решения оптимизационной задачи осуществляется не из единственной точки, а из «популяции» точек.

Специфика работы метода ЭМ позволяет накапливать и использовать знания об исследованном пространстве поиска (рис. 12) и, следовательно, позволяет проявлять способность к самообучению.

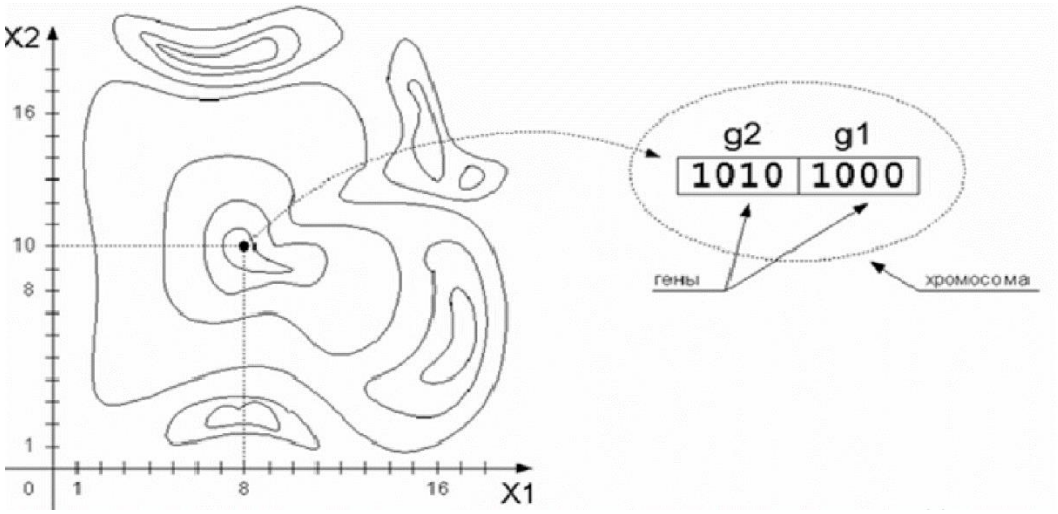

Рисунок 12 – Двумерное пространство поиска

Самообучение и накопление знаний свойственно для интеллектуальных программных систем. В самом же процессе поиска используется значение целевой функции, а не её приращение, как принято в методах машинного обучения.

В процессе ЭМ применяются вероятностные, а не детерминированные правила поиска генерации решений.

Во время работы процесса ЭМ выполняется параллельный анализ разных областей пространств решений, в связи с чем образуется возможность нахождения новых областей с более оптимальными значениями ЦФ за счёт объединения оптимальных решений из разных популяций.

В связи с обозначенными выше особенностями метод ЭМ выгодно выступает с целым рядом преимуществ:

– независимость от вида функций;

– независимость от области определения и типов переменных;

– применимость к широкому кругу задач без нужды в изменениях.

Для получения новой популяции решений из текущей в ГА применяются операторы [11], т.е. для формирования следующего фенотипа на основе

предыдущего. По-другому их ещё называют этапами, т.к. выполняются последовательно на каждой итерации. В стандартном ГА используются следующие операторы:

– оператор отбора (селекции);

– оператор кроссинговера (рекомбинации);

– оператор мутации или инверсии.

Стоит отметить, что в этапы работы ГА также обычно включают этап формирования начальной популяции, но, как можно догадаться, в отличии от обозначенных выше, этот этап выполняется всего один раз.

Оператор отбора используется для определения на основе fitness-функции кандидатов, гены которых будут использоваться для формирования следующего поколения. В ГА могут применяться разные схемы селекции. Некоторые из этих схем проиллюстрированы на рисунке .

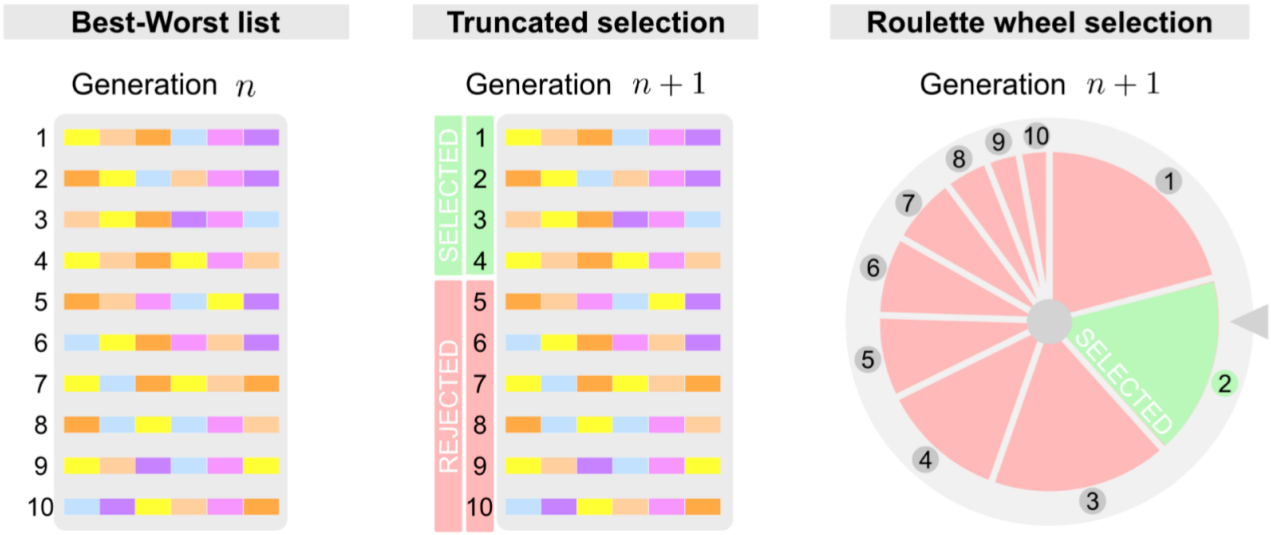

Рисунок 13 –

Наиболее распространены следующие схемы:

– Пропорциональный отбор. Число копий хромосомы пропорционально её оптимальности. В следующее поколение перейдут только хромосомы с оптимальностью выше средней.

– Отбор на основе <<колеса рулетки>> \cite{аверченков2012эволюционное}. Чем выше оптимальность хромосомы (решения) --- тем больше её сектор на колесе рулетки. Случайность дает

возможность всем решениям попасть в следующее поколение, даже не самым успешным.

– Турнирный отбор. Популяция разбивается на группы случайным образом. Из каждой группы выбирается лучшая хромосома.

– Ранжированный отбор. Решению присваивается ранг, на основе которого вычисляется вероятность хромосомы попасть в следующие поколение.

– Стратегия элитизма. Её суть сводится к сохранению на всех этапах ЭМ постоянной по объёму прослойки лучших особей популяции.

Общий вид цикла работы операторов представлен на рисунке .

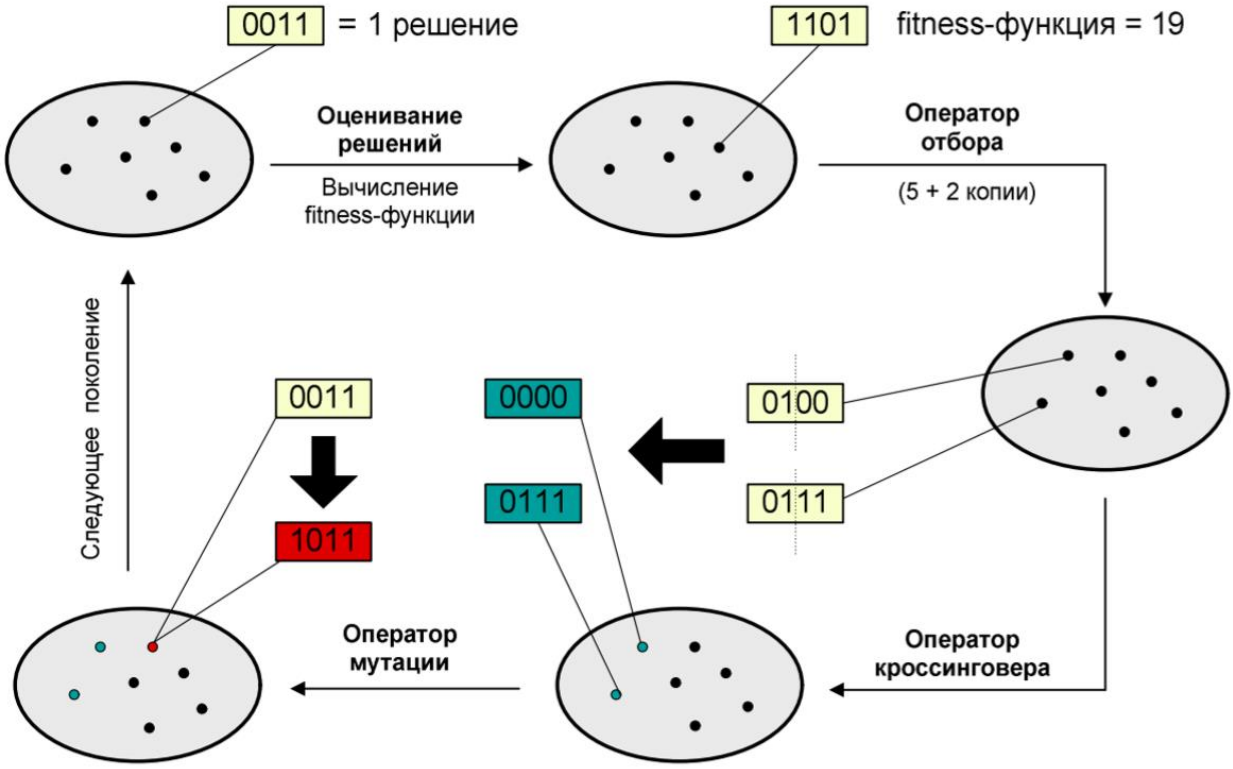

Рисунок 14 – Цикл работы ГА

Оператор кроссинговера предназначен для обмена части генетического материала между родительскими хромосомами, которые успешно прошли этап селекции. Целью такого обмена является генерация новых хромосом для следующего поколения.

В обмене могут участвовать от 2-х до n-го количества родителей. Также принято довольно часто вообще не производить рекомбинацию и использовать

для генерации новых хромосом только одного родителя. Говорят, что в таком случае оператор кроссинговера не выполняется, а происходит генерация копий наиболее успешных хромосом из предыдущего поколения.

Оператор мутации или инверсии со случайной вероятностью изменяет (инвертирует) случайный ген в хромосомах, полученных после работы оператора кроссинговера. После этого происходит оценка fitness-функции и весь цикл начинается сначала.

1.4 Феномен интеллектуальных систем

Интеллектуальные системы (ИС) – это системы с целью. Проектирование сложных систем с целью связано с задачами, отличительными признаками которых являются: принятие решений на основе неполной и «зашумленной» информации, необходимость учета опыта предыдущих решений, выработка стратегии для получения однозначно оптимального решения и др. Наиболее известными задачами являются задачи интерпретации, диагностики, контроля, прогнозирования, планирования, проектирования и т. д. [9].

Понятие ИС сформировалось в процессе развития кибернетики, теории управления, теории алгоритмов и современных информационных технологий. Обобщение научных знаний по перечисленным областям информатики привело к возникновению рассматриваемого понятия ИС. Так или иначе, сами же ИС относят к научной области искусственного интеллекта (ИИ) [9].

В отличии от обычных информационных систем, ИС позволяют получить решение трудно формализуемых и слабо структурированных задач. Это вытекает из следующих признаков и особенностей ИС:

– направленность на реализацию «мягких» моделей;

– работа с динамическими данными;

– развитие системы со временем и извлечение знаний;

– вывод новой информации из уже имеющейся;

– умение объяснять свои действия.

Возможность ИС работать со слабоструктурированными данными подразумевает наличие следующих важных качеств:

– способность решать задачи, описанные в терминах «мягких» моделей, когда зависимости между основными показателями являются не вполне определенными или даже не известными в пределах некоторого класса;

– способность к работе с динамически изменяющимися данными, что позволяет ИС корректировать свои действия, исходя из текущей ситуации;

– способность к развитию и накоплению знаний системой со временем.

Таким образом отличия ИС от информационной системы можно представить таблицей 3 .

Таблица 3 – Сравнение ИС и обычных программных систем

Характеристика	**Интеллектуальная система**	**Информационная система**
Тип обработки	Символьный	Выделенная
Метод	Эвристический поиск	Точный алгоритм
Задание шагов решения	Неявное	Явное
Искомое решение	Удовлетворительное	Оптимальное
Управление и данные	Смешанные	Разделенные
Знания (данные)	Неточные	Точные
Модификации	Частые	Редкие

Создание ИС – это творческий процесс. Существует множество различных классификаций ИС, одна из которых представлена на рисунке 15. Обычно выделяют: ЭС (экспертные системы), ИНС (искусственные нейронные сети), СГА (системы с генетическими алгоритмами), МАС (мультиагентные системы) и другие.

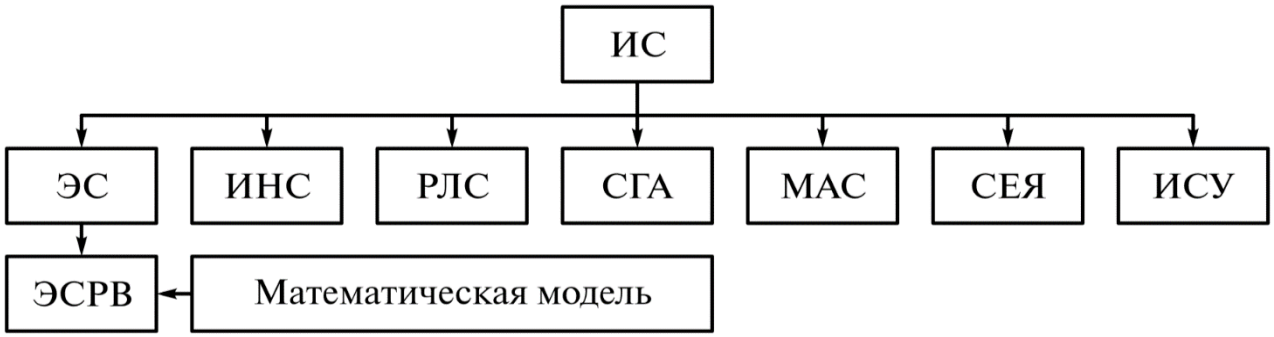

Рисунок 15 – Классификация ИС

Создание ИС связано с решением следующих взаимосвязанных проблем:

– Проблема формализации знаний. Данная задача решается посредством привлечения экспертов по исследуемой предметной области. Результатом решения является готовая концептуальная схема модели.

– Проблема представления знаний. Её решение состоит в разработке формальной модели, с помощью которой будут представлены знания в системе.

– Проблема использования знаний. Здесь возникает необходимость в разработке теории вычислений и преобразованиях, которые будет проводить ИС для достижения поставленной цели или целей.

– Проблема создания базы знаний (БЗ) и системы управления ею (СУБЗ). Эта задача ложится на плечи системных программистов.

Знания о предметной области и способах решения в ней задач весьма разнообразны. Возможны различные классификации этих знаний.

По мере развития научно-технического прогресса размеры потоков текстовой и числовой информации увеличиваются. Задачей ИС является поиск закономерностей и аналогий в совокупности данных, что представляет из себя, по сути, процесс интеллектуального анализа данных.

Интеллектуальный анализ данных --- это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей, то есть извлечения информации, которая может быть охарактеризована как знание.

В последнее время наряду с использованием классической статистики активно развиваются альтернативные методы анализа данных и извлечения

знаний, базирующиеся на иных, нежели традиционная интегро-дифференциальная парадигма, подходах и представленные на рис. 16 [10].

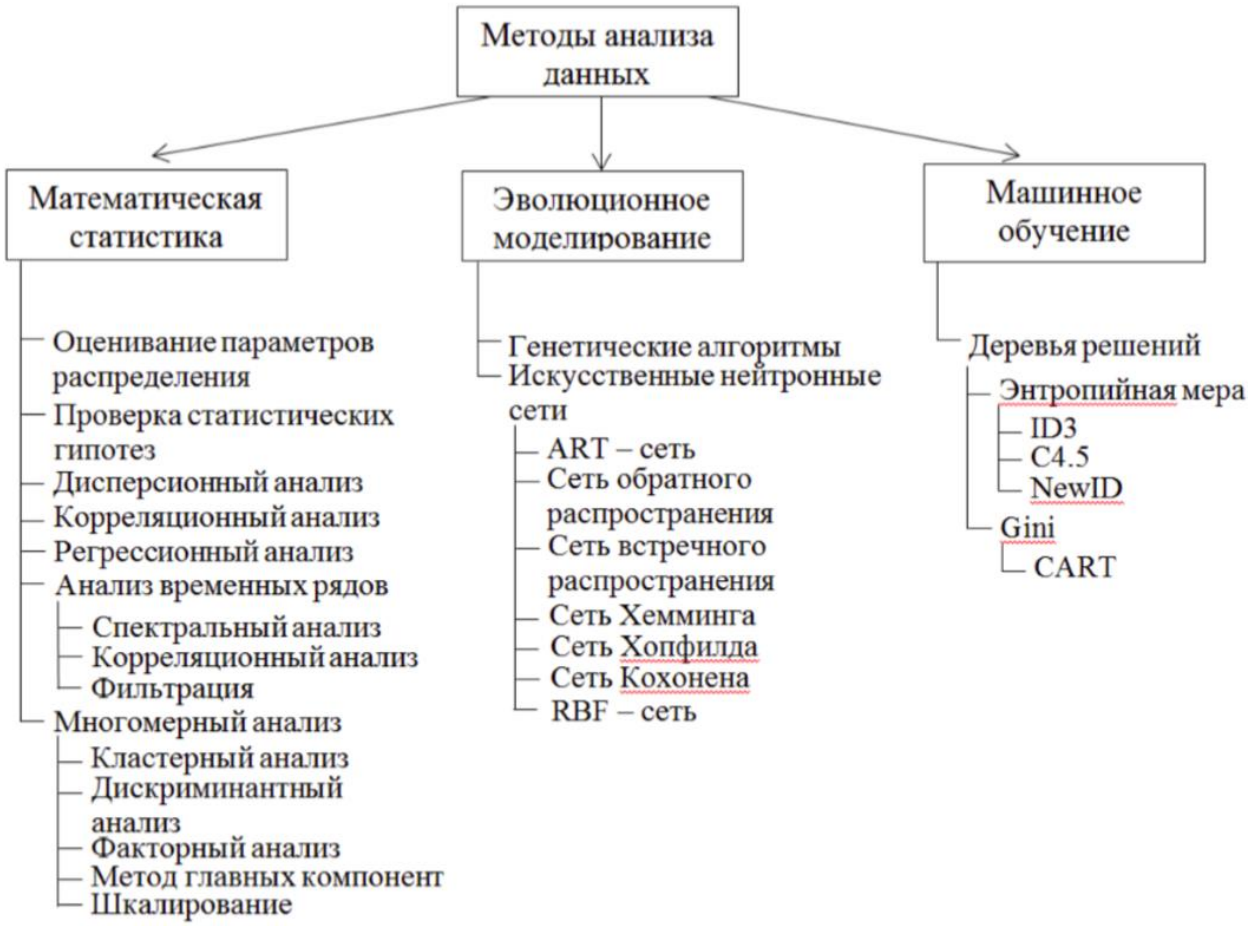

Рисунок 16 — Классификация методов анализа данных

Термин «Эволюционное моделирование» в настоящее время является достаточно устоявшимся, и общепринято под этим термином подразумевать генетические алгоритмы, однако, согласно источнику [10], к эволюционному моделированию также иногда относят искусственные нейронные сети.

Термин «машинное обучение» оставляет больше возможностей для дискуссий о том, какие методы имеются ввиду, в частности, сюда относятся деревья решений [10].

Приведенные способы анализа данных, поиска закономерностей и извлечения знаний могут быть применены практический к любой предметной области, поэтому они могут с успехом использоваться для решения самого широкого спектра задач.

1.5 Интеллектуальная система эволюционного моделирования

Одно из направлений использования ИС – это решение задач оптимизации. Например, СГА, которые используют подход эволюционного моделирования.

Специфика работы методов ЭМ позволяет накапливать и использовать знания об исследованном пространстве поиска и, следовательно, позволяет проявлять способность к самообучению, что свойственно для интеллектуальных систем [9]. В самом же процессе поиска используется значение целевой функции, а не её приращение, как принято в методах машинного обучения.

Таким образом, подход ЭМ обладает преимуществами:

– независимость от вида функций;

– независимость от области определения и типов переменных;

– применимость к широкому кругу задач без нужды в модификации.

Технологии объектно-ориентированного программирования позволяет реализовать ГА максимально гибко и удобно для программиста при дальнейшем внедрении в информационные системы и сопровождении [11].

В дальнейшем для краткости вместо словосочетаний «интеллектуальная система, использующая метод эволюционного моделирования» будем использовать аббревиатуру ИСЭМ (интеллектуальная система эволюционного моделирования).

С точки зрения ИС процесс ЭМ является генератором или источником мета-данных (мета-знаний) [13]. В методе ЭМ мета-знаниями выступает выбранные стратегии и параметры проведения эволюционного процесса. К этим стратегиям и параметрам могут относиться:

– выбранный оператор отбора (пропорциональный, «рулетка», «турнирный», на основе ранжирования и т.д.);

– выбранный оператор кроссинговера или его отсутствие;

– выбранный оператор мутации.

– набор параметров для выше обозначенных операторов;

– вид fitness-функции;

– настройки начального и последующих размеров популяции хромосом;

– размер хромосом;

– критерии остановки (достижение заданного числа поколений, снижение разнообразия популяции, снижение скорости сходимости алгоритма и т.д.);

– лимиты по времени и количеству поколений (без них, очевидно, в случае неудачного подбора параметров, процесс ЭМ может никогда не остановиться);

– и так далее и тому подобное.

Агрегатором процедурных знаний выступает используемый ГА с теми параметрами, которые были выбраны на основе метазнаний. Знания ГА отвечают на такие вопросы, как: сколько сгенерировать начальных особей, как вычислить fitness-функцию, как применить оператор кроссинговера и мутации, т.д.

Результатом работы ГА будут являться декларативные знания, то есть конкретные особи с конкретными параметрами и значениями fitness-функции (можно их обозначить как факты).

Не трудно догадаться, что если метод ЭМ является слоем метаданных, то сама часть ИС, которая использует метод ЭМ для достижения своих целей, является прослойкой мета-мета-данных .[13]

Обобщив сказанное выше, можно проиллюстрировать соответствия между элементами иерархии видов знаний и конкретными знаниями из ИСЭМ рисунком .

Рисунок 17 – Уровни знаний

На основании мета и мета-мета -знаний формируется общая стратегия решения задачи [13]. В рамках этих стратегий даже может быть принято решение об запуске сразу нескольких процессов ЭМ.

В самих же процессах ЭМ может быть несколько параллельно работающих ГА, обрабатывающих свои изолированные популяции хромосом. Такой подход позволяет сильно распарралелить работу ИСЭМ, что потребует от такой системы для выработки лучшей стратегии ещё и такого мета-мета знания, как осведомленность о количестве доступных физических процессоров и памяти и т.д.

Подобный параллелизм сильно увеличивает количество затрачиваемой памяти. Даже результаты неудачных моделей необходимо хранить, т.к. на их анализе ИС делается вывод о том, какие настройки ЭМ необходимо изменить, чтобы потенциально достичь более оптимального результата.

Одной из основных актуальных проблем, решаемых при создании интеллектуальных систем, является обеспечение в условиях их непрерывного функционирования адаптации программных средств и структур представления данных и знаний для оперативного решения различных сложных задач [14] Способность ИС к самообучению является её основным признаком.

Для того, чтобы система могла обучаться, она и/или её подсистемы хранения и обработки данных должны обладать свойством адаптивности, наращиваемости, изменяемости, т.е. в некотором смысле [14]. Это означает, ИСЭМ должна хранить все те результаты, которые были ею получены в ходе процессов ЭМ, причем даже тех, которые являлись провальными с точки зрения поставленных задач. В целом, хранение результатов и конфигураций неэффективных моделей является логически верным, т.к. на ошибках учатся.

С точки зрения человека подход проб и ошибок и идея «негативный опыт — тоже опыт» кажутся вполне себе естественными, однако использование той же самой идеи в ИСЭМ не является тривиальной задачей. Опустим момент формирования новых знаний на основе имеющихся и подробнее рассмотрим вторую наиболее острую проблему — необходимость хранения и обработки довольно большого количества данных, которые необходимо как-то оценивать и хранить.

2 Проектирование интеллектуальной системы распределения вычислительной нагрузки

2.1 Создание модели задачи балансировки нагрузки между узлами распределенной вычислительной системы

Узлы вычислительного кластера представляют собой множество $P = \{P_1, \dots, P_n\}$. Узлы взаимосвязаны с друг другом и располагают некоторыми конфигурационными характеристиками, такими как ЦПУ, ОЗУ, ПЗУ и т.п.

Пусть на узлах инсталлирован программный комплекс, образующий в совокупности с физическим кластером вычислительную систему. Если узлы вычислите кластера расположены отдаленно друг от друга, то имеет место быть распределенная вычислительная система (РВС). Если физические узлы вычислительного кластера обладают разной конфигурацией, то РВС можно считать гетерогенной.

Распределенные гетерогенные системы представляют наибольший практический интерес с точки зрения задачи распределения вычислительной нагрузки. т.к. необходимо учитывать большее количество факторов. Гомогенные и/или централизованные вычислительные системы являются частными случаями, из чего можно сделать логическое заключение, что, в случае успешной формализации и решения задачи для распределенной гетерогенной системы, найти решения для частных случаев не составит труда.

Таким образом, в рамках ВКРМ предлагается реализовывать программную модель вычислительного кластера для случая гетерогенной РВС. В таком случае необходимо детализировать конфигурацию каждого отдельного вычислительного узла, а также длины расстояний между ними.

Тогда пусть каждый P_i есть кортеж вида $\{p_{proc}, p_{ram}, p_{rom}\}$, которые олицетворяют собой производительность центрального процессора, емкости оперативной и постоянной памяти соответственно.

Пусть имеется множество связей $U = \{U_1, \dots, U_m\}$, где каждый U_j есть кортеж $\{P_s, P_t, L\}$, где P_s и P_t узлы, а L расстояние между ними.

2.2 Проектирование структуры интеллектуальной системы распределения нагрузки

Если не вдаваться в подробности взаимосвязей внутренних компонентов проектируемого программного продукта, то описать его структуру и способ взаимодействия с внешним миров можно с помощью рисунка 18.

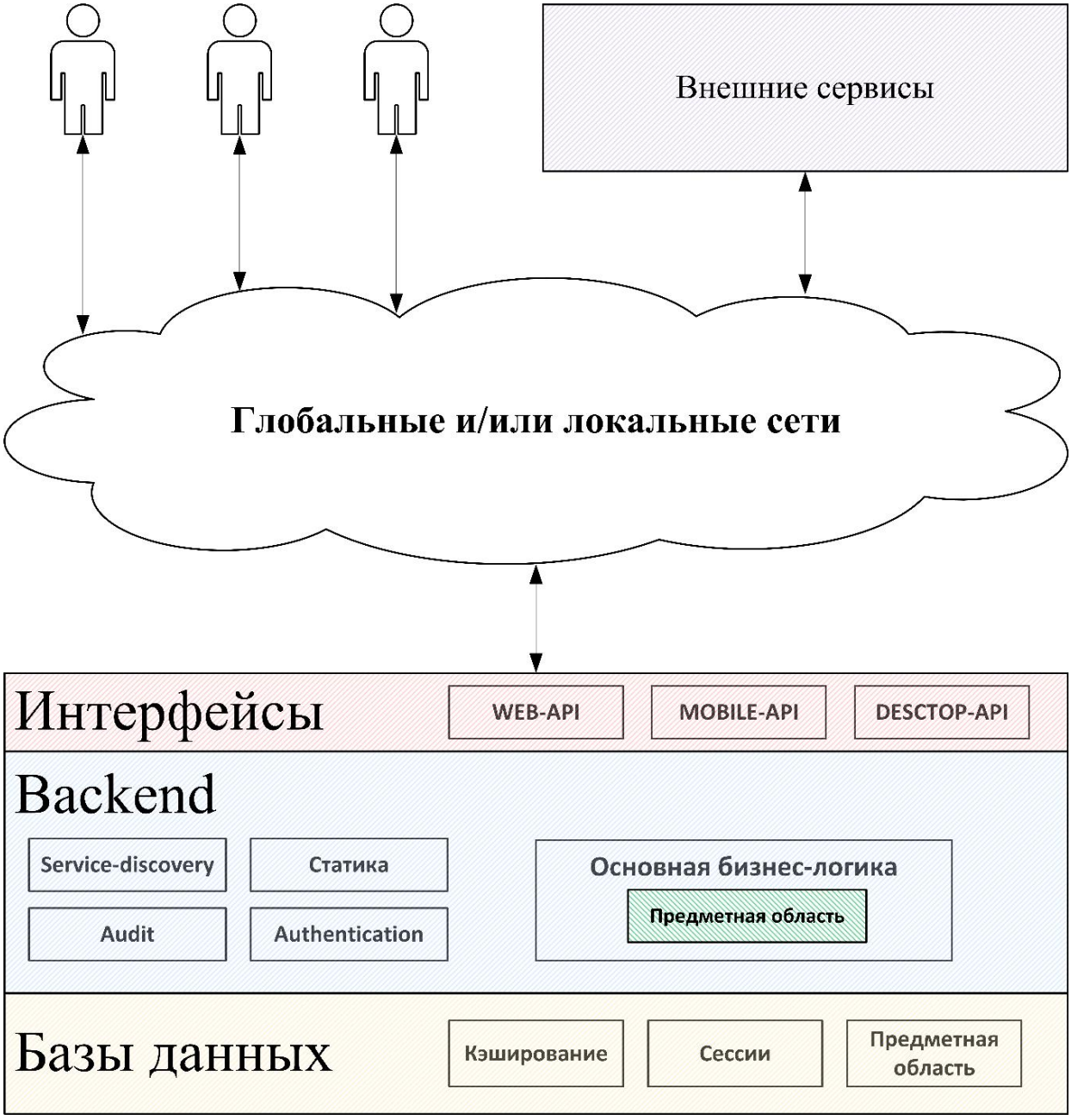

Рисунок 18 – Упрощенная структура системы

Как видно из рисунка выше, вся бизнес-логика, связанная с особенностями предметной области, инкапсулирована. Она не должна влиять на более общие процессы, как аудит, service-discovery или аутентификация.

В действительности, в рамках выпускной квалификационной работы магистра реализуются не все компоненты. Это происходит по 2 причинам: либо программный компонент уже имеется в виде свободно распространяемого готового решения (например, для service-discovery есть Consul), либо он является по своей сути дубликатом-аналогом уже имеющейся функциональности (для взаимодействия с системой достаточно одного desktop, web или mobile интерфейса).

В соответствии с требованиями ТЗ, сервис распределения нагрузки реализуется как веб-сервис, с которым взаимодействие пользователей и других АС осуществляется с помощью сетевого протокола прикладного уровня HTTP.

Веб-сервис распределения нагрузки проектируется в соответствии с многоуровневой архитектурой. Конкретной в рамках данной ВКРМ используется 3-х уровневая имплементация (рис. 19).

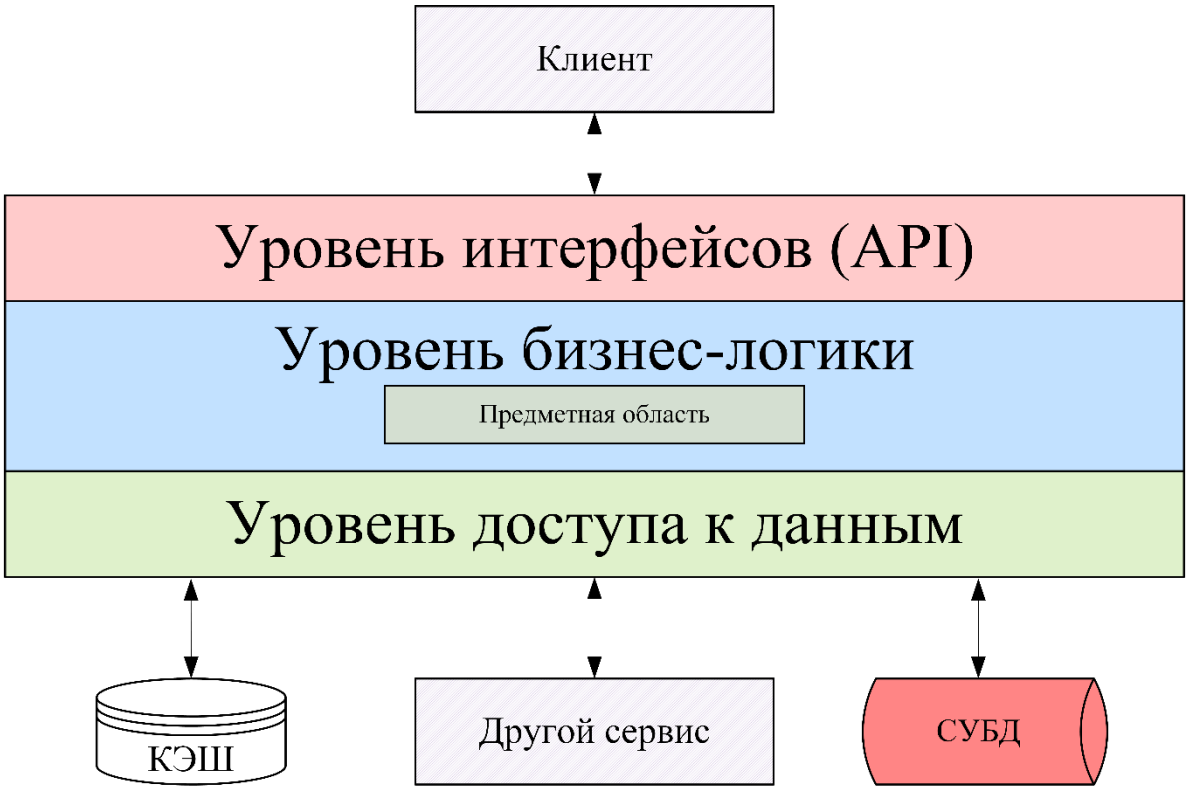

Рисунок 19 – Многоуровневая архитектура

В представленной архитектуре используется принцип вертикального управления. Каждый уровень использует нижестоящий уровень для

выполнения своих функций, при этом нижестоящие уровни не должны ничего знать про вышестоящие.

В качестве клиентов системы могут выступать как живые люди, так и другие программы. Если в случае других систем для взаимодействия достаточно воспользоваться API (application programming interface), проектируемой в рамках данной курсовой работы системы, то для обеспечения возможности взаимодействия пользователя-человека с системой необходима разработка отдельных графических интерфейсов.

Примеры сквозного взаимодействия пользователя-человека и пользователя-программы с распределенной системой показаны на рисунке .

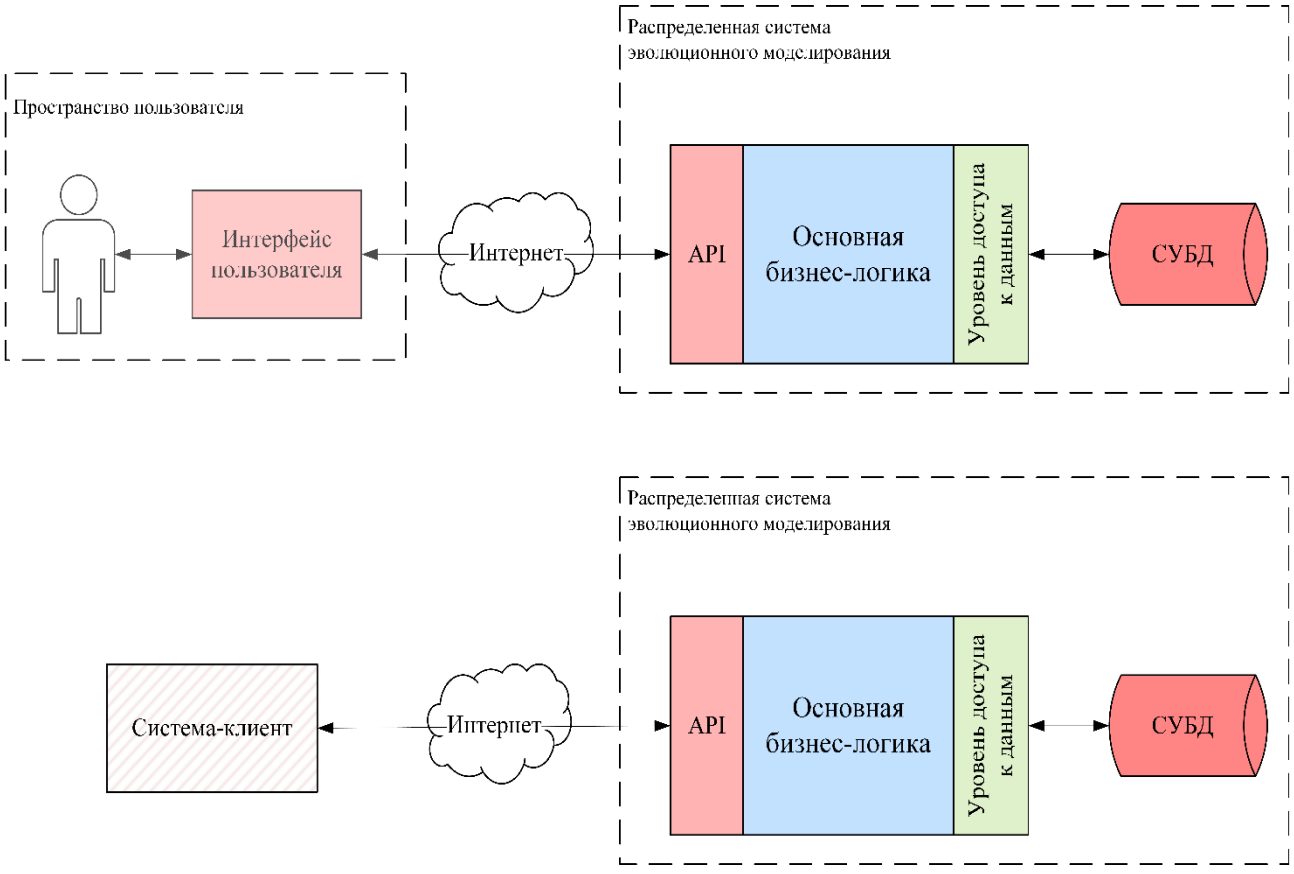

Рисунок 20 – Сквозное взаимодействие

Как видно из рисунка выше, для взаимодействия с системой пользователю необходимо получить соответствующее ПО с пользовательских интерфейсов. В случае веб-интерфейса пользователю достаточно перейти на соответствующий адрес.

Для мобильного или desktop интерфейса всё немного сложнее, однако соответствующее ПО может быть распространено с помощью сервисов цифровой дистрибьюции. В настоящей курсовой работе для демонстрации работоспособности системы предлагается использовать desktop интерфейс, о котором подробнее будет говориться в следующем разделе РПЗ.

2.3 Проектирование алгоритмов функционирования

Наиболее распространенное применение ГА — это решение оптимизационных задач. Чаще всего ГА не решает оптимизационную задачу напрямую, а используется в качестве алгоритма оптимизации (настройки) уже существующего специализированного эвристического алгоритма (рис. 21)

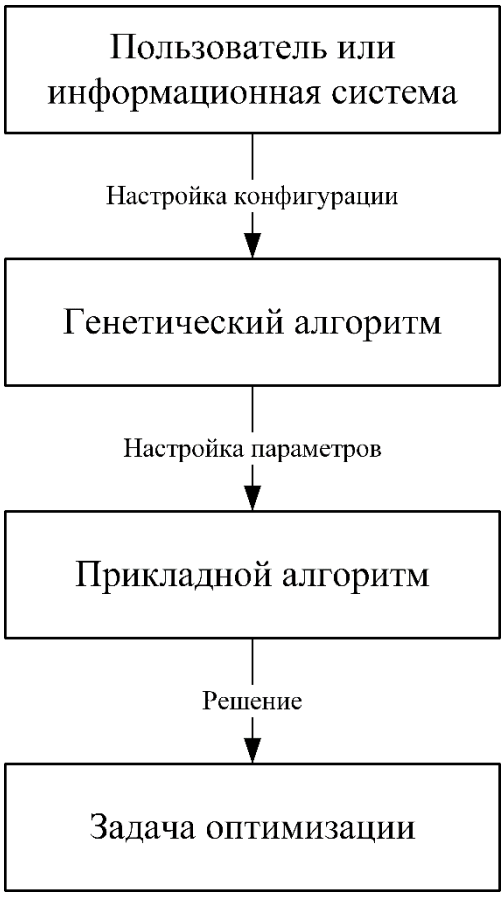

Рисунок 21 – Иерархическая структура работы алгоритмов

Чтобы использование такого подхода было оправдано, необходимо, чтобы оптимизируемый алгоритм обладал достаточным перечнем параметров (обычно хватает от 3-х и более) для того, чтобы было что варьировать средствами ГА с целью избегания полного перебора.

В данной курсовой работе для демонстрации состоятельности такого иерархического подхода предлагается в качестве решаемой оптимизационной задачи использовать графовый полносвязный вариант задачи Коммивояжера. В качестве прикладного алгоритма использовать муравьиный алгоритм. В качестве ГА использовать классический ГА с 3 операторами.

Структура классического 3-операторного ГА представлена на рисунке 22. Из него видно, что ГА состоит из нескольких основных компонентов: набора операторов, последовательно применяемых к каждому агенту в поколении, популяции агентов и блока с оценкой приспособленности агентов.

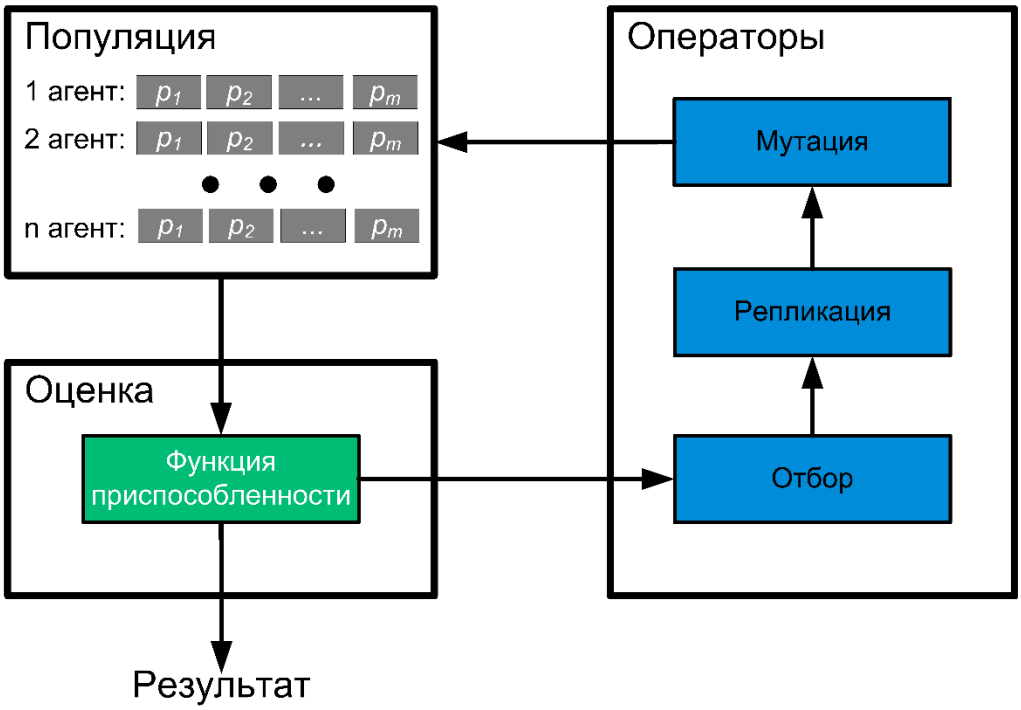

Рисунок 22 – Структура генетического алгоритма

Результатом работы ГА является некоторая лучшая конфигурация агента, позволившая достигнуть наилучшего результата во время решения задачи оптимизации. Как правило, ГА используется там, где невозможно использовать алгоритмы с полиномиальной сложностью. ГА позволяют достичь некоторого неплохого результата, однако не дают гарантии того, что найденное решение является наилучшим.

ГА весьма гибки, каждый из компонентов поддается настройке. Существует множество вариантов стратегий отбора, репликации и мутаций

агентов. В данном курсовом проекте предлагается использовать простые реализации этих операторов: табличный отбор, репликация на основе случайного выбора агента, мутация случайного гена с заранее заданной вероятностью.

При решении задачи оптимизации средствами проектируемой распределенной системы эволюционного моделирования чётко прослеживается иерархичность во взаимодействии имеющих место алгоритмов. Можно сказать, что имеет место наличие мета-алгоритмов.

Таким образом, по аналогии с многоуровневой архитектурой, вышестоящие алгоритмы используют нижестоящие для решения задачи, при этом нижестоящие алгоритмы ничего не знают вышестоящих.

Подобный процесс гибридизации теоретически может продолжаться бесконечно, однако в рамках данной курсовой работы было принято решение, что целесообразно остановиться на 2-компонентной гибридизации (использовать генетический и муравьиные алгоритмы) (рис. 23).

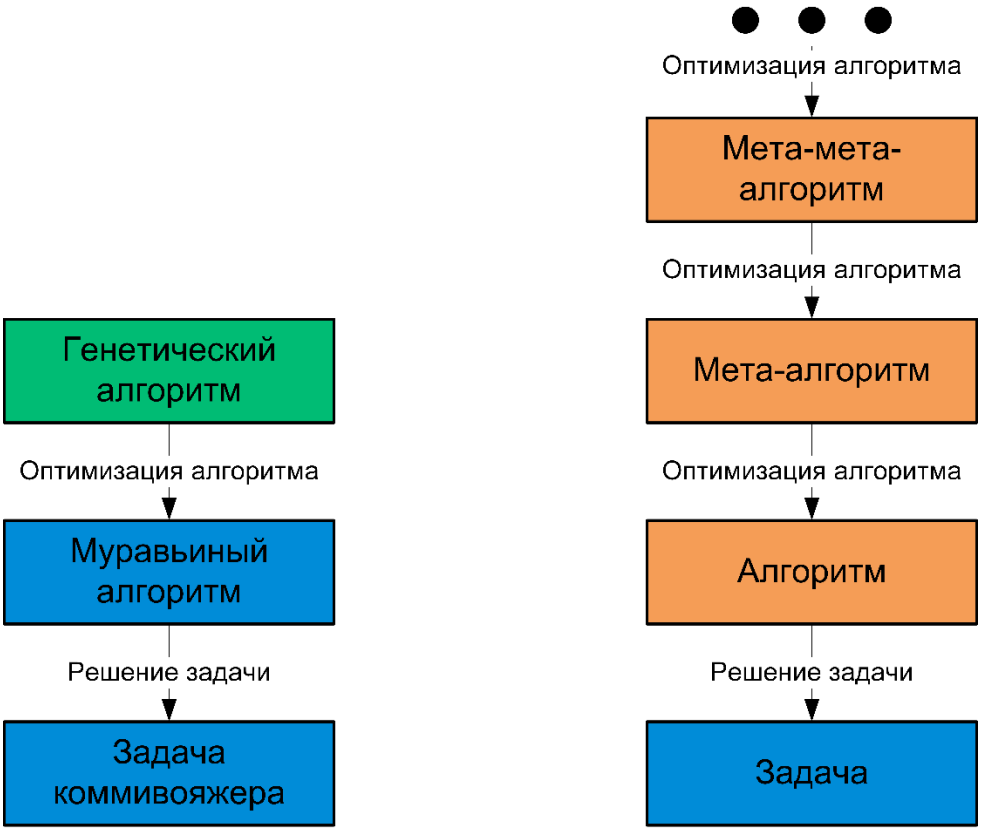

Рисунок 23 – Суть гибридизации

Таким образом, если ГА является мата-алгоритмом оптимизации алгоритма решения задачи оптимизации, то для настройки самого ГА должен использоваться какой-то мета-мета-алгоритм, однако в рамках курсового проекта в роли настройщика ГА выступает пользователь системы.

Подбор конфигурации ГА и параметров для него – это нетривиальный процесс. Безусловно, конфигурация и параметры для ГА могут задаваться конечным пользователем, однако, если пользователь не является экспертом в предметной области задачи и предметной области эволюционного моделирования, это будет малоэффективно.

С точки зрения феномена интеллектуальных систем было бы желательно, чтобы параметры и варианты конфигурации подбирались системой автоматически. Для этого могут применяться уже накопленные результаты предыдущих решений задач. В таком случае, следуя принципу правдоподобия, конфигурация для какой-либо задачи по умолчанию подбиралась бы на основе конфигурации для уже решенной максимально похожей задачи. Однако в случае отсутствия достаточного количества данных это может также не сработать.

Поскольку временные ресурсы ограничены, а ТЗ не подразумевает создание интеллектуальной системы, то вполне допустимым будет использование некоторых базовых зашитых в систему параметров ГА в комбинации с возможностью их варьирования со стороны пользователя. В качестве таких параметров могут выступать:

– размер популяции (количество агентов в поколении);

– процент отсеиваемых агентов;

– вероятность мутации агентов во время репликации;

– максимальное количество поколений (количество итераций ГА);

– критерий основа;

– формула расчёта функции приспособленности;

– интервалы возможных значений для генов;

– и т.д.

Для муравьиного алгоритма также существует множество гиперпараметров для варьирования, однако подбор этих параметров входит в задачи генетического алгоритма, а не конечного пользователя системы. Одна из целей разрабатываемой системы – избежать полного перебора параметров, возложив эту функцию на ГА.

Разрабатываемая система проектируется таким образом, чтобы её потенциально возможно было переиспользовать и для других алгоритмов оптимизации и решаемых ими задач. В конце концов, если отбросить подробности внутренней работы, ГА занимается перебором параметров настраиваемого алгоритма.

ГА нет необходимости знать, какую задачу решает оптимизируемый прикладной алгоритм. Это потенциально позволяет использовать один и тот же ГА для оптимизации (настройки) разных прикладных алгоритмов (рис. 24).

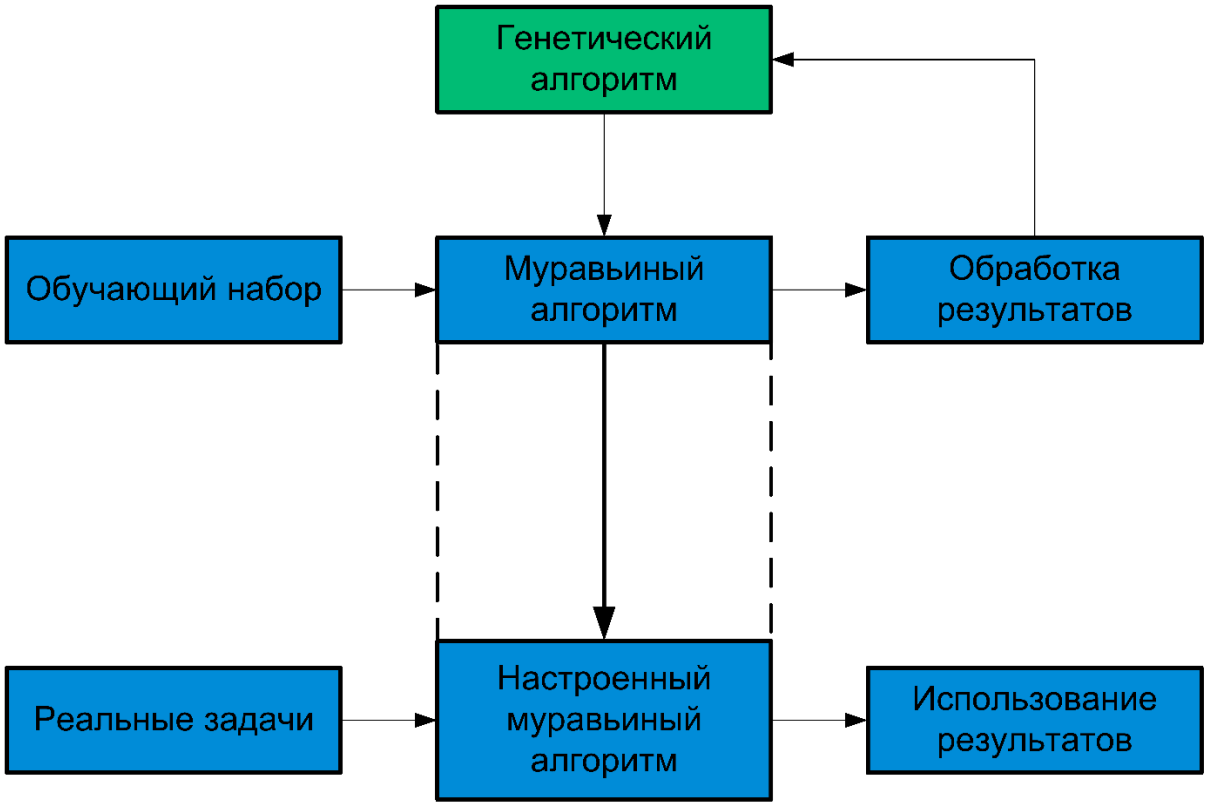

Рисунок 24 – Иллюстрация практической пользы системы

2.4 Проработка инфраструктурной составляющей

Планируется обеспечивать гибкость и масштабируемость системы с использованием концепции service-discovery (рис. 25).

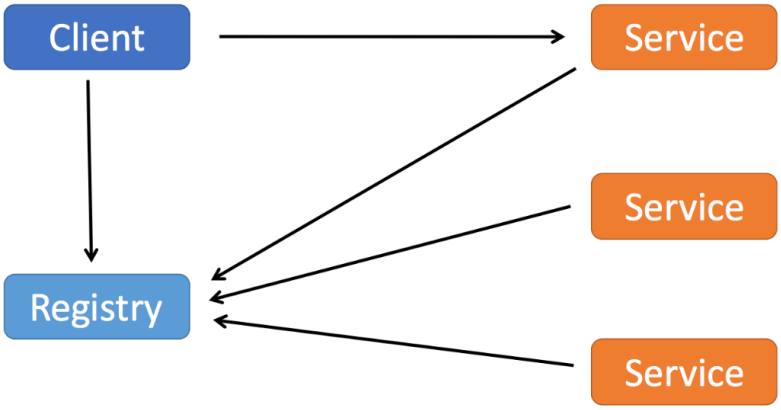

Рисунок 25 – Концепция service-discovery

Аудит системы планируется производить средствами отдельного сервиса. В компонентах системы должны быть предусмотрены соответствующие элементы API, позволяющие специализированному сервису аудита по необходимости считывать метрики с сервисом с необходимой ему периодичностью.

В комбинации с концепцией service-discovery осуществление аудита с точки зрения администрирования становится значительно более простым (рисунок 26).

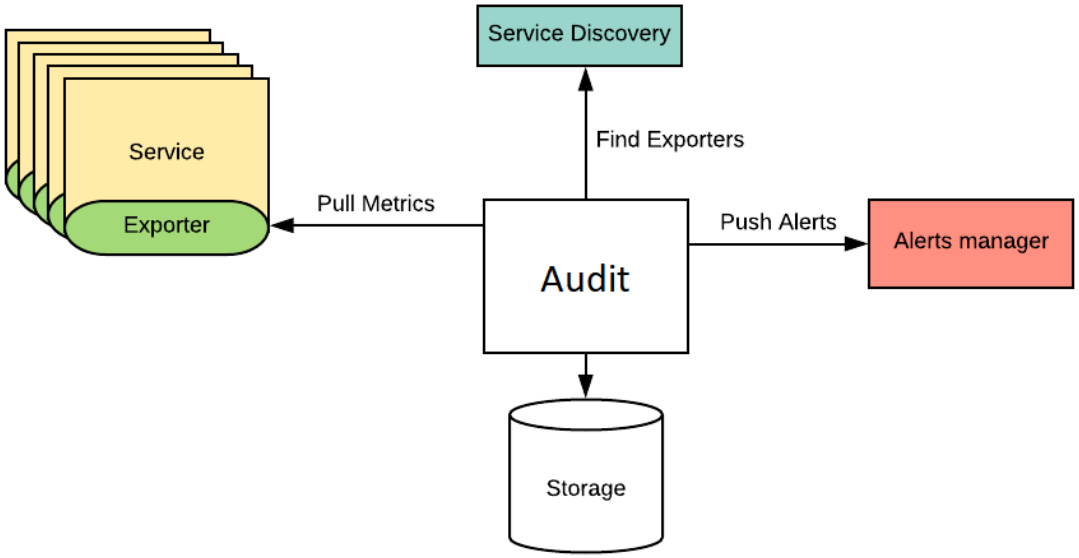

Рисунок 26 – Система аудита в комбинации с service-discovery

2.5 Проектирование базы данных интеллектуальной системы

БД описывается совокупностью сервисных, пользовательских и сущностей предметной области и их связей (таблицы 4 и 5).

Таблица 4 – Сущности в БД без ссылочных атрибутов

Сущность	Атрибуты	Описание
Пользователь	ID, логин, пароль	Конечный пользователь системы
Сессия пользователя	ID, метка «протухания»	Идентификатор сессии пользователя системы
Агент	ID, значение ЦФ	Вся работа ЭМ сводится к генерации агентов с как можно большим значением ЦФ
Генокод	ID, код	Параметр агента, исп. для вычисления значения ЦФ
Поколение	ID, порядковый номер	Набор агентов на одной итерации работы процесса ЭМ
Сеанс	ID, название, дата создания, описание	Связующая и разграничивающая структурная единица
Генетический алгоритм	ID, название, описание, параметры	Мета-алгоритм оптимизации
Прикладной алгоритм	ID, название, описание, параметры	Алгоритм оптимизации
Задача оптимизации	ID, название, описание	Оптимизационная задача, решаемая прикладным алгоритмом

Некоторые атрибуты сущностей, которые носят такие имена, как «свойства, параметры, показатели, настройки, схема и т.д.» не являются строго типизированными. Это означает, что в зависимости от используемых методов и алгоритмов, данные атрибуты могут иметь разное представление.

Использование нестрого типизированных атрибутов позволяет повысить гибкость и универсальность схемы БД. В противном случае приходилось бы использовать множество дополнительных сущностей и полей.

Таблица 5 – Связи между сущностями БД

Связь	Тип	Описание
Пользователь-сессия	Один-ко-многим	Действующая всегда только одна
Поколение – Агент	Один-ко-многим	В рамках одного поколения существует множество агентов
Агент – Генокод	Один-ко-многим	Агенты кодируются произвольным кол-вом генов
Агент - Агент	Один-ко-многим	У одного агента может быть множество предков и потомков
Пользователь-сеанс	Один-ко-многим	У одного пользователя может быть множество сеансов моделирования
Сеанс – Поколение	Один-ко-многим	За один сеанс генерируется произвольное кол-во поколений
ГА – Сеанс	Один-ко-многим	Один и тот же ГА может использоваться в разных сеансах
ПА – Сеанс	Один-ко-многим	Один и тот же ПА может использоваться в разных сеансах
Задача - ПА	Один-ко-многим	Для решения задачи могут использоваться разные алгоритмы

Совокупность связей и сущностей, описанных в таблицах выше, может быть представлена схемой БД на рисунке 27.

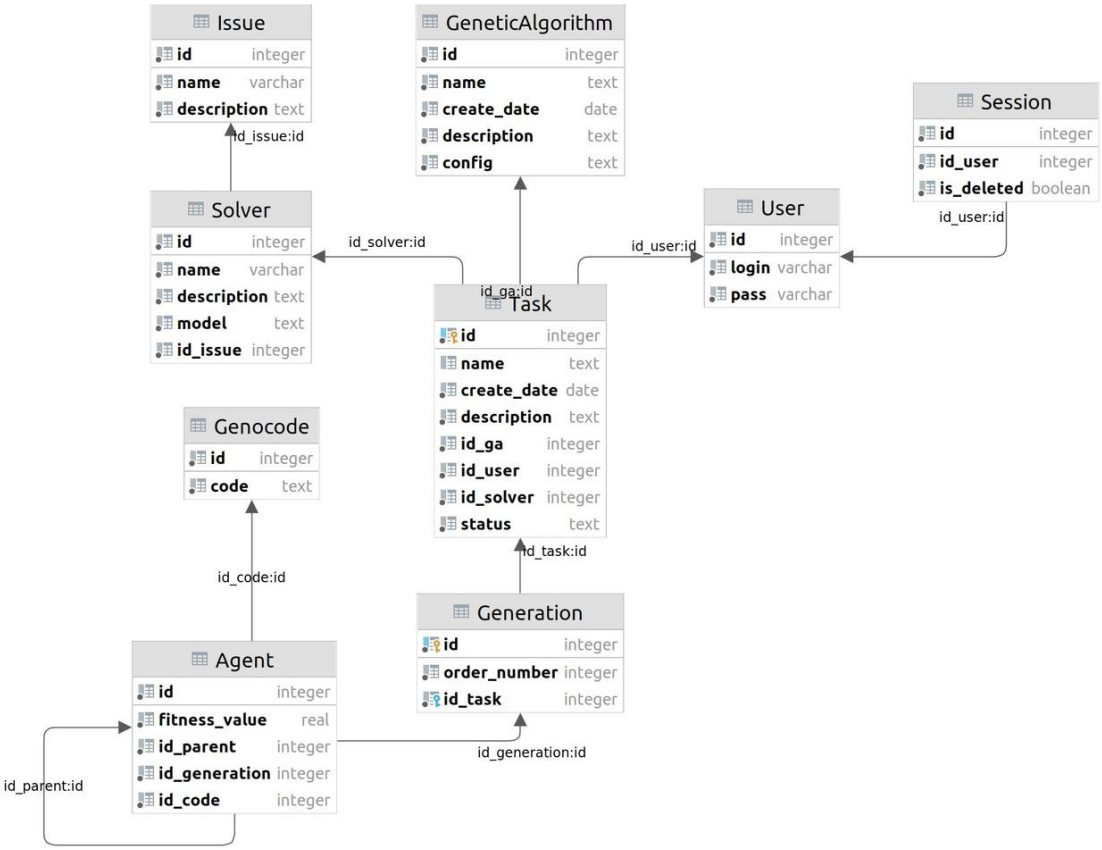

Рисунок 27 – Инфологическая модель БД

3 Выбор инструментов и технологий. Реализация программных компонентов

3.1 Выбор программных решений

Для реализации интеллектуальной системы распределения нагрузки в вычислительном кластере было принято решение разработать следующие программные компоненты:

– интерфейс системного администратора (конфигурирование, запуск, выключение и сопровождение системы);

– сервис балансировки нагрузки (непосредственное выполнение целевой задачи);

– реляционная СУБД (хранений, чтение и запись данных);

– сервис оркестровки (получение информации о конфигурации кластера);

– сервис мониторинга (получение информации о состоянии кластера).

Выбранные технологии реализации перечисленных технических компонентов представлены в таблице 6.

Таблица 6 – Выбранные программные решения

Программный компонент	**Технология реализации**	**Описание**
Интерфейс системного администратора	Python (PySimpleGUI)	Универсальный интерпретируемый язык программирования
Сервис балансировки нагрузки	Golang	Компилируемый многопоточный язык программирования, который широко используется при написании серверов с параллельной обработкой запросов.
Реляционная СУБД	PostgreSQL	Одна из наиболее популярных свободно распространяемых реляционных СУБД
Оркестровка	Consul	Предоставляет ряд функций, обеспечивающих доступность информации об инфраструктуре
Мониторинг	Prometheus	Система мониторинга с богатым

		графическим интерфейсом и полностью открытым исходным кодом

Consul и Prometheus являются готовыми решениями, которые не требуют написания программного кода, а нуждаются лишь в настройке в соответствии с решаемыми задачами. Микросервис Golang же является рукописным.

Таблицы, связи, ограничения, процедуры, триггеры и другое в PostgreSQL реализуются написанием SQL кода.

3.2 Выбор инструментария разработки

В качестве основных инструментов был выбран набор продуктов от компании JetBrains — международная компания, которая разрабатывает инструменты для разработки на всех более-менее распространенных языках программирования. Компания предоставляет бесплатные лицензии студентам.

Среди продуктов JetBrains использовался следующий набор для разработки и администрирования: Goland, Pycharm, Datagrip.

Помимо инструментов с графическим интерфейсом использовалась децентрализованная система контроля версий git.

3.3 Реализация интерфейсов. Разработка API

API реализуется в виде набора протоколов, допустимых форматов передачи данных, процедур и функций.

При разработке сервиса распределения нагрузки соблюдались соглашения об именовании функций и процедур API для каждого уровня в соответствии с таблицей 7.

Таблица 7 – Соответствие между видами операций и их именованием на каждом из уровней

Действие	**Уровень представления**	**Уровень бизнес-логики**	**Уровень доступа к данным**
Добавление	POST	Add	Insert
Выборка	GET	Read	Select
Модификация	PUT	Change	Update

Удаление	DELETE	Remove	Delete

Реализация API микросервисов Golang осуществляется в соответствии с набором интерфейсных ограничений, вытекающих из набора принципов REST:

– каждый ресурс обозначен уникальным идентификатором;

– все операции с ресурсами осуществляются через представление;

– самодостаточные сообщения в рамках одного запроса;

Основным форматом сериализации данных в реализуемой подсистеме является JSON. Передача структуры JSON осуществляется средствами протокола HTTP в составе тела запроса или ответа

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы магистра была создана ИС «Интеллектуальная система распределения нагрузки в вычислительном кластере».

В процессе выполнения выпускной квалификационной работы был проведен анализ предметной области, выделены основные сущности предметной области и бизнес-процессы. На основе выделенных сущностей и связей была спроектирована и реализована схема базы данных. На основе выделенных бизнес-процессов были обозначены решаемые кейсы и выбраны проектно-технические решения.

Для спроектированной системы были выбраны инструменты программной реализации. Все использованные инструменты являются свободно распространяемым программным обеспечением.

Итоговая ИС выполняет поставленные перед ней задачи в достаточно мере.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНКОВ

1. Малявко А. А., Менжулин С. А. Суперкомпьютеры и системы. Построение вычислительных кластеров: учебное пособие / А. А. Малявко, С. А. Менжулин, Новосибирск: Изд-во НГТУ, 2018. 96 с.

2. Емельянов А. Балансировка нагрузки : основные алгоритмы и методы // Habr [Электронный ресурс]. URL: https://habr.com/ru/company/selectel/blog/250201 (дата обращения: 18.02.2022).

3. Спиряев О. Вертикальное и горизонтальное масштабирование систем // BYTE [Электронный ресурс]. URL: https://www.bytemag.ru/articles/detail.php?ID=6670 (дата обращения: 26.02.2022).

4. Krzywda J. [и др.]. Power-performance tradeoffs in data center servers: DVFS, CPU pinning, horizontal, and vertical scaling // Future Generation Computer Systems. 2018. № November (81). С. 114–128.

5. Технологии микроэлектроники на пальцах : «закона Мура», маркетинговые ходы и почему нанометры нынче не те. Часть 1 // Habr [Электронный ресурс]. URL: https://habr.com/ru/post/453438 (дата обращения: 11.03.2022).

6. Технологии микроэлектроники на пальцах : «закона Мура», маркетинговые ходы и почему нанометры нынче не те. Часть 2 // Habr [Электронный ресурс]. URL: https://habr.com/ru/post/456298 (дата обращения: 13.03.2022).

7. Технологии микроэлектроники на пальцах : «закона Мура», маркетинговые ходы и почему нанометры нынче не те. Часть 3 // Habr [Электронный ресурс]. URL: https://habr.com/ru/post/456306 (дата обращения: 13.03.2022).

8. Тамбовцев, А Ю Смольянов А. Г. О практических аспектах создания приложений микросервисной архитектуры совместно с распределённым программным брокером сообщений Apache Kafka // XXI ВЕК: ИТОГИ ПРОШЛОГО И ПРОБЛЕМЫ НАСТОЯЩЕГО ПЛЮС. 2021. № 53 (1). С. 31–34.

9. Советов Б. Я., Цехановский В. В., Чертовской В. Д. Интеллектуальные системы и технологии / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской, Москва: Академия, 2013. 320 с.

10. Коровин А. М. Интеллектуальные системы / А. М. Коровин, Челябинск:

Издательский центр Южно-Уральского государственного университета, 2015. 61 с.
11. Аверченков В. И., Казаков П. В. Эволюционное моделирование и его применение / В. И. Аверченков, П. В. Казаков, Москва: ФЛИНТА, 2016. 200 с.
12. Zadeh L. A. Soft Computing and Fuzzy Logic // IEEE Software. 1994. № 6 (11). С. 48–56.
13. Пугачев Е. К. Интеллектуальные технологии и системы / Е. К. Пугачев, Москва:, 2019.
14. Варламов О. О. Эволюционные базы данных и знаний для адаптивного синтеза интеллектуальных систем. Миварное информационное пространство / О. О. Варламов, Москва: Радио и связь, 2002. 282 с.

ПРИЛОЖЕНИЕ А

Техническое задание

Листов 9

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ6

_____ А.В. Пролетарский

«__» _____ 2022 г.

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА РАСПРЕДЕЛЕНИЯ НАГРУЗКИ В ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ

Техническое задание

Листов 9

Студент	ИУ6-43М	_____	В.Д. Шульман
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Руководитель		_____	О.Ю. Ерёмин
		(Подпись, дата)	(И.О. Фамилия)

1 ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку программной системы «Интеллектуальная система распределения нагрузки в вычислительном кластере», используемой для балансировки нагрузки в высоконагруженных распределенных вычислительных системах и предназначенной для внедрения и эксплуатации в качестве подсистемы.

Область распределенных вычислительных систем в настоящее время характеризуется быстрыми темпами изменения идеологий и подходов. Если говорить о ситуации в вебе, то прослеживается устойчивая тенденция к «утончению» клиента и усложнению серверной части, в том числе повышается степень её распределённости. В значительной мере этому способствует развитие технологий контейнеризации и виртуализации. Возникает потребность в разработке и совершенствовании технологий распределения нагрузки в сложных вычислительных системах.

2 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Интеллектуальная система распределения нагрузки в вычислительном кластере разрабатывается в соответствии с тематикой кафедры ИУ6 «Компьютерные системы и сети» факультета ИУ «Информатика и системы управления» МГТУ им. Баумана.

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Основное назначение интеллектуальной системы распределения нагрузки в вычислительном кластере заключается в балансировке нагрузки в распределенных информационных системах между вычислительными узлами.

4 ИСХОДНЫЕ ДАННЫЕ, ЦЕЛИ И ЗАДАЧИ

4.1 Исходные данные

4.1.1 Исходными данными для разработки являются следующие материалы:

4.1.1.1 Перечень работ, содержащих исходные данные для разработки:

• Кондратьев Алексей Анатольевич Скрытые проблемы распределенных вычислений // Электротехнические и информационные комплексы и системы. 2015. №3. URL: https://cyberleninka.ru/article/n/skrytye-problemy-raspredelennyh-vychisleniy (дата обращения: 02.12.2021).

• Шамакина Анастасия Валерьевна Обзор технологий распределенных вычислений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. №3. URL: https://cyberleninka.ru/article/n/obzor-tehnologiy-raspredelennyh-vychisleniy (дата обращения: 05.12.2021).

• Алексеев И.А., Егунов В.А., Панюлайтис С.В., Чекушкин А.А. МЕТОДЫ И СРЕДСТВА БАЛАНСИРОВКИ НАГРУЗКИ В НЕОДНОРОДНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ // ИВД. 2020. №11 (71). URL: https://cyberleninka.ru/article/n/metody-i-sredstva-balansirovki-nagruzki-v-neodnorodnyh-vychislitelnyh-sistemah (дата обращения: 10.12.2021).

4.1.1.2 Перечень прототипов:

• Голубева Яна Вадимовна Разработка и исследование алгоритмов балансировки нагрузки в параллельной реализации метода ветвей и границ // Современные информационные технологии и ИТ-образование. 2015. №11. URL: https://cyberleninka.ru/article/n/razrabotka-i-issledovanie-algoritmov-balansirovki-nagruzki-v-parallelnoy-realizatsii-metoda-vetvey-i-granits (дата обращения: 07.12.2021).

• Перепелкин Владислав Александрович, Сумбатянц Илья Ильич Стенд для отладки и тестирования качества работы локальных системных распределенных алгоритмов динамической балансировки нагрузки // Вестник ЮУрГУ. Серия: Вычислительная мате-

матика и информатика. 2015. №3. URL: https://cyberleninka.ru/article/n/stend-dlya-otladki-i-testirovaniya-kachestva-raboty-lokalnyh-sistemnyh-raspredelennyh-algoritmov-dinamicheskoy-balansirovki-nagruzki (дата обращения: 12.12.2021).

4.2 Цель работы

Целью работы является прототип интеллектуальной системы распределения нагрузки в вычислительном кластере для балансировки нагрузки в распределенных вычислительных системах.

4.3 Решаемые задачи

4.3.1 Выбор методов проектирования и технологий разработки системы.

4.3.2 Анализ требований технического задания с точки зрения выбранных технологий и уточнение требований к информационной системе.

4.3.3 Исследование предметной области – разработка моделей, описывающих предметную область, постановка задачи и выбор методов её решения.

4.3.4 Определение архитектуры информационной системы: разработка ее структуры; определение набора программных компонентов.

4.3.5 Анализ требований технического задания и разработка спецификаций проектируемого программного обеспечения.

4.3.6 Разработка структуры программного обеспечения и определение спецификаций его компонентов.

4.3.7 Проектирование компонентов программного продукта и баз данных.

4.3.8 Реализация компонентов с использованием выбранных средств и их автономное тестирование.

4.3.9 Сборка программного обеспечения и его комплексное тестирование.

4.3.10 Оценочное, функциональное, интеграционное и нагрузочное тестирование программного обеспечения

5 ТРЕБОВАНИЯ К ПРОГРАММНОМУ ИЗДЕЛИЮ

5.1 Требования к функциональным характеристикам

5.1.1 Выполняемые функции

5.1.1.1 Для пользователя:

- идентификация и аутентификация;
- настройка системы;
- конфигурация и/или загрузка модели кластера;
- выбор алгоритма балансировки.
- конфигурация тестовой нагрузки;
- запуск задачи настройки алгоритма;
- просмотр и выгрузка результатов выполнения задач.

5.1.1.2 Для администратора системы (если он предусматривается):

- идентификация и аутентификация;
- настройка системы;
- блокировка пользователей;
- принудительное завершение задач.

5.1.2 Исходные данные:

- модель кластера.

5.1.3 Результаты:

- настройка алгоритма балансировки.

5.2 Требования к надежности

5.2.1 Предусмотреть контроль вводимой информации.

5.2.2 Предусмотреть защиту от некорректных действий пользователя.

5.2.3 Обеспечить целостность информации в базе данных.

5.3 Условия эксплуатации

5.3.1 Условия эксплуатации в соответствие с СанПиН 2.2.2/2.4.1340-03.

5.4 Требования к составу и параметрам технических средств

5.4.1 Минимальная конфигурация технических средств:

5.4.2.1 Тип процессора ..... Pentium.

5.4.2.2 Объем ОЗУ ..... 2048 Мб.

5.5 Требования к информационной и программной совместимости

5.5.1 Программное обеспечение должно работать под управлением операционных систем семейства Linux.

5.5.2 Входные данные должны быть представлены в формате JSON.

5.5.3 Результаты должны быть представлены в формате JSON.

5.5.4 Система должна взаимодействовать с другим программным обеспечением и системами через протокол HTTP.

5.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

5.7 Требования к транспортированию и хранению

Требования к транспортировке и хранению не предъявляются.

5.8 Специальные требования

Сгенерировать установочную версию программного обеспечения.

6 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

6.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

6.2 Разрабатываемое программное обеспечение должно включать справочную систему.

6.3 В состав сопровождающей документации должны входить:

6.3.1 Расчетно-пояснительная записка на 95-105 листах формата А4 (без приложений).

6.3.2 Техническое задание (Приложение А).

6.3.3 Руководство пользователя (Приложение Б).

6.3.4 Фрагмент исходного текста программного обеспечения (Приложение Г).

6.4 Графическая часть должна быть выполнена на 10 листах формата А1 (копии формата А3/А4 включить в качестве приложений к расчетно-пояснительной записке):

6.4.1 Схема структурная информационной системы.

6.4.2 Схема функциональная программного обеспечения.

6.4.3 Схемы (модели) процессов (методов формирования результатов, механизмы выводов и т.п.).

6.4.4 Диаграмма вариантов использования.

6.4.5 Концептуальная модель предметной области.

6.4.6 Схемы структурные компонент, даталогическая и инфологическая схемы базы данных.

6.4.7 Схема взаимодействия модулей.

6.4.8 Граф (диаграмма) состояний интерфейса.

6.4.9 Формы интерфейса.

6.4.10 Схема процесса разработки программного продукта (при различных технологиях, например, при структурном, объектном, нисходящем, восходящем подходах и т.п.).

7 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Выполнить технико-экономическое обоснование разработки.

8. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

№	Название этапа	Срок, даты, %	Отчетность
1.	Разработка технического задания	2.02.2022 - 28.02.2022 5 %	Утвержденное техническое задание и задание на выпускную квалификационную работу
2.	Анализ требований и уточнение спецификаций (эскизный проект)	1.03.2022 - 7.03.2022 5 %	Спецификации программного обеспечения.
3.	Проектирование структуры программного обеспечения, проектирование компонентов (технический проект)	8.03.2022 - 31.03.2022 35 %	Схема структурная системы и спецификации компонентов.
4.	Реализация компонентов и автономное тестирование компонентов. Сборка и тестирование.	1.04.2022 - 30.04.2022 40 %	Тексты программных компонентов. Тесты.
5.	Разработка документации.	1.05.2022 - 25.05.2022 8 %	Расчетно-пояснительная записка.
6.	Прохождение нормоконтроля, проверка на антиплагиат, получение рецензии, подготовка доклада и предзащита.	25.05.2022- 5.06.2022 5 %	Иллюстративный материал, доклад, рецензия, справки о нормоконтроле и проценте плагиата.
7.	Защита выпускной квалификационной работы.	6.06.2022- 04.07.2022 2 %	

9 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

9.1 Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

9.2 Порядок защиты

Защита осуществляется перед государственной экзаменационной комиссией (ГЭК).

9.3 Срок защиты

Срок защиты определяется в соответствии с планом заседаний ГЭК.

10 ПРИМЕЧАНИЕ

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.