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)

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

1 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).

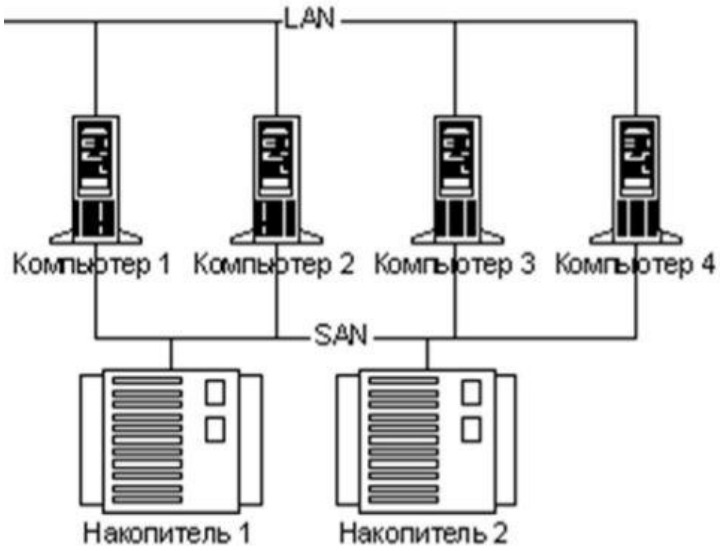

Рисунок 1 – Вычислительный кластер

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).

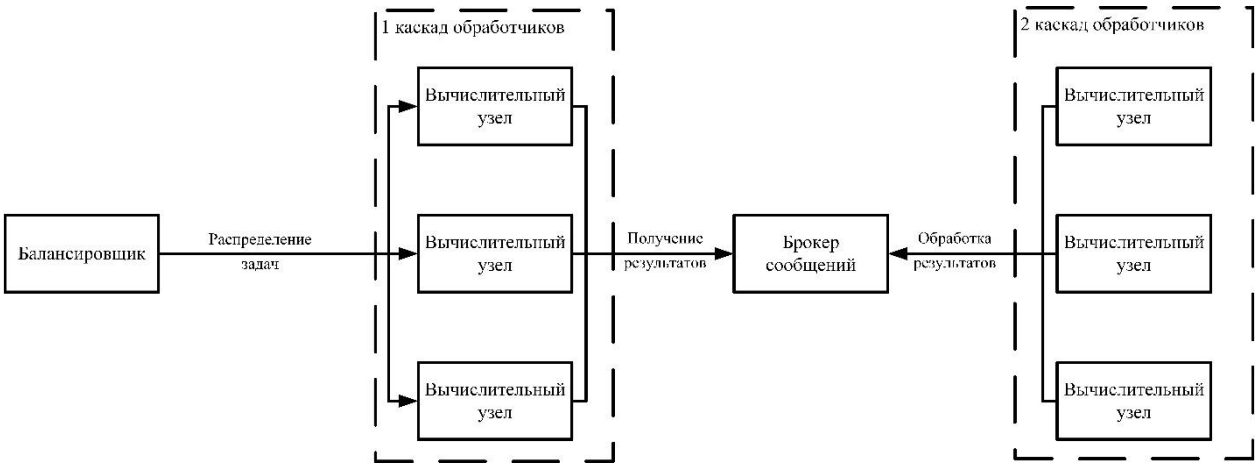

Рисунок 3 – Комбинирование балансировщика и брокера сообщений

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].

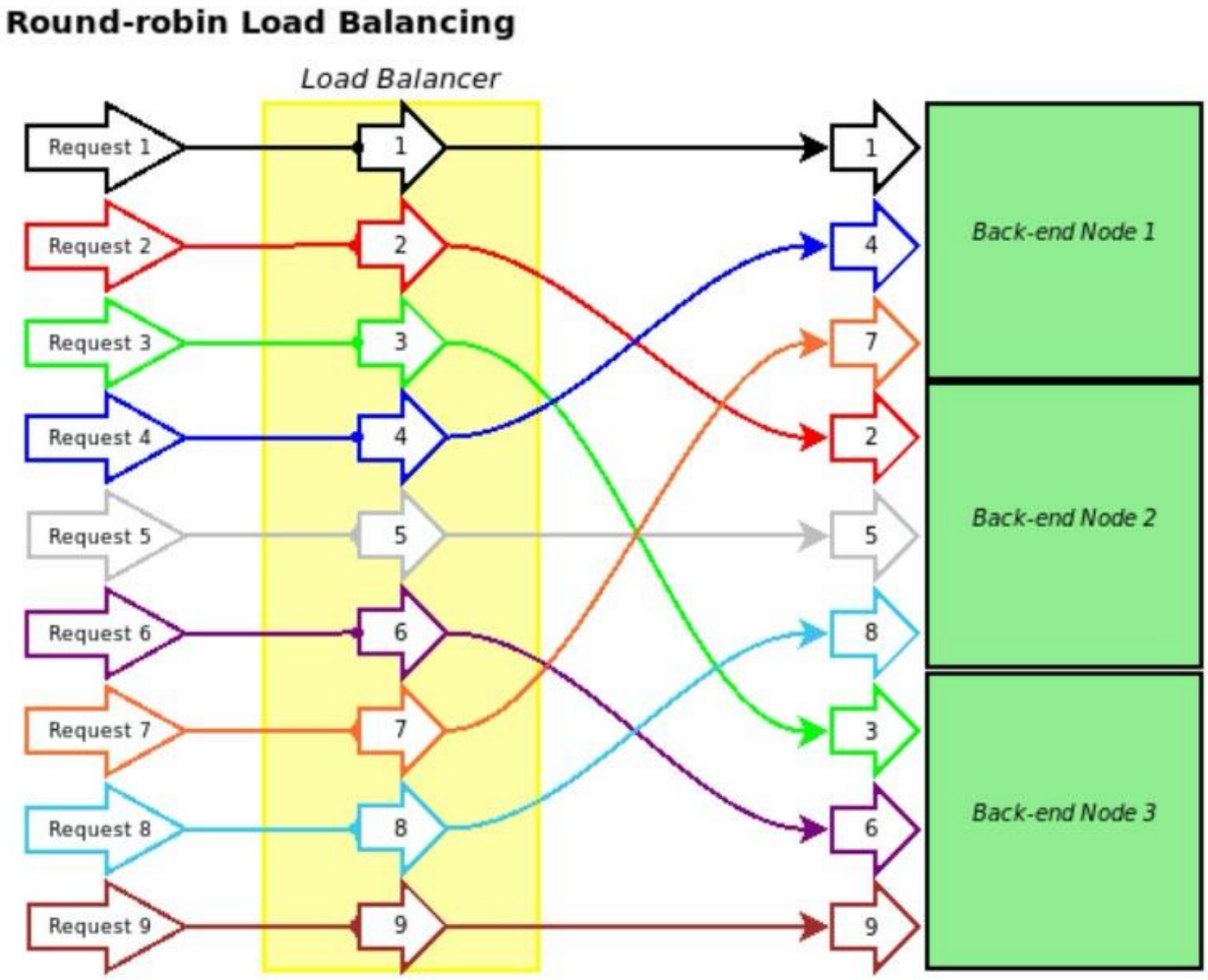

Рисунок 5 – Round Robin.

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).

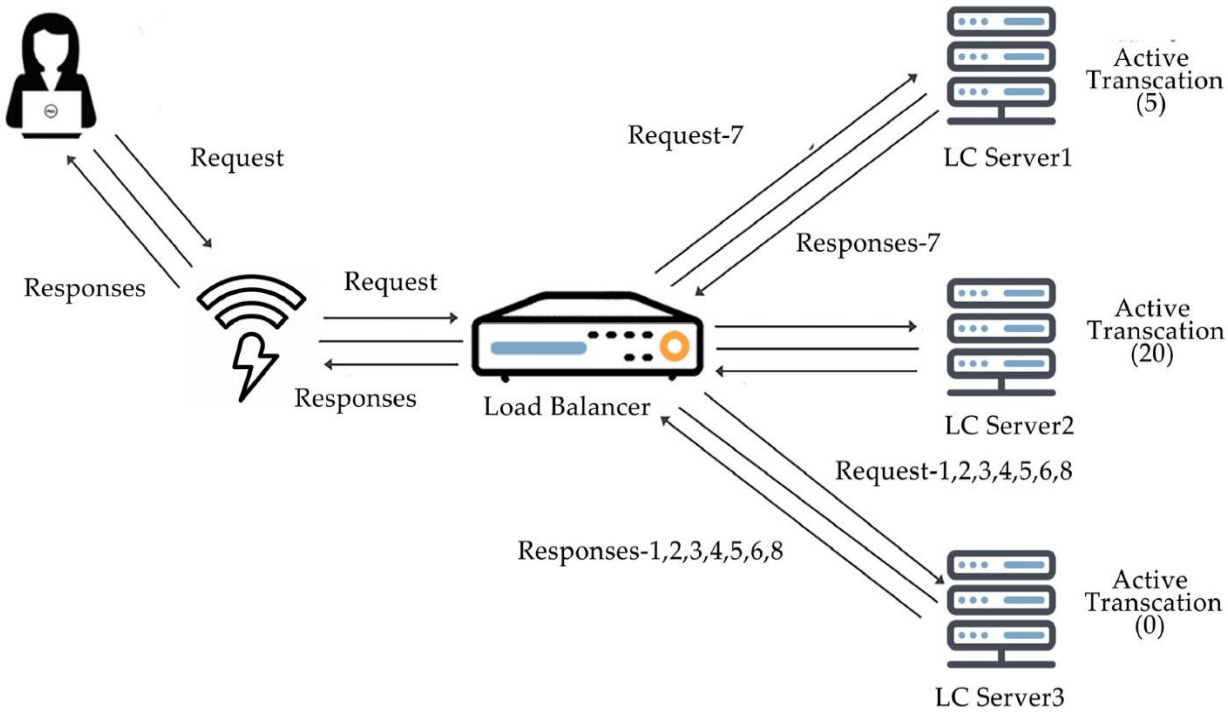

Рисунок 6 – Least Connections.

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.

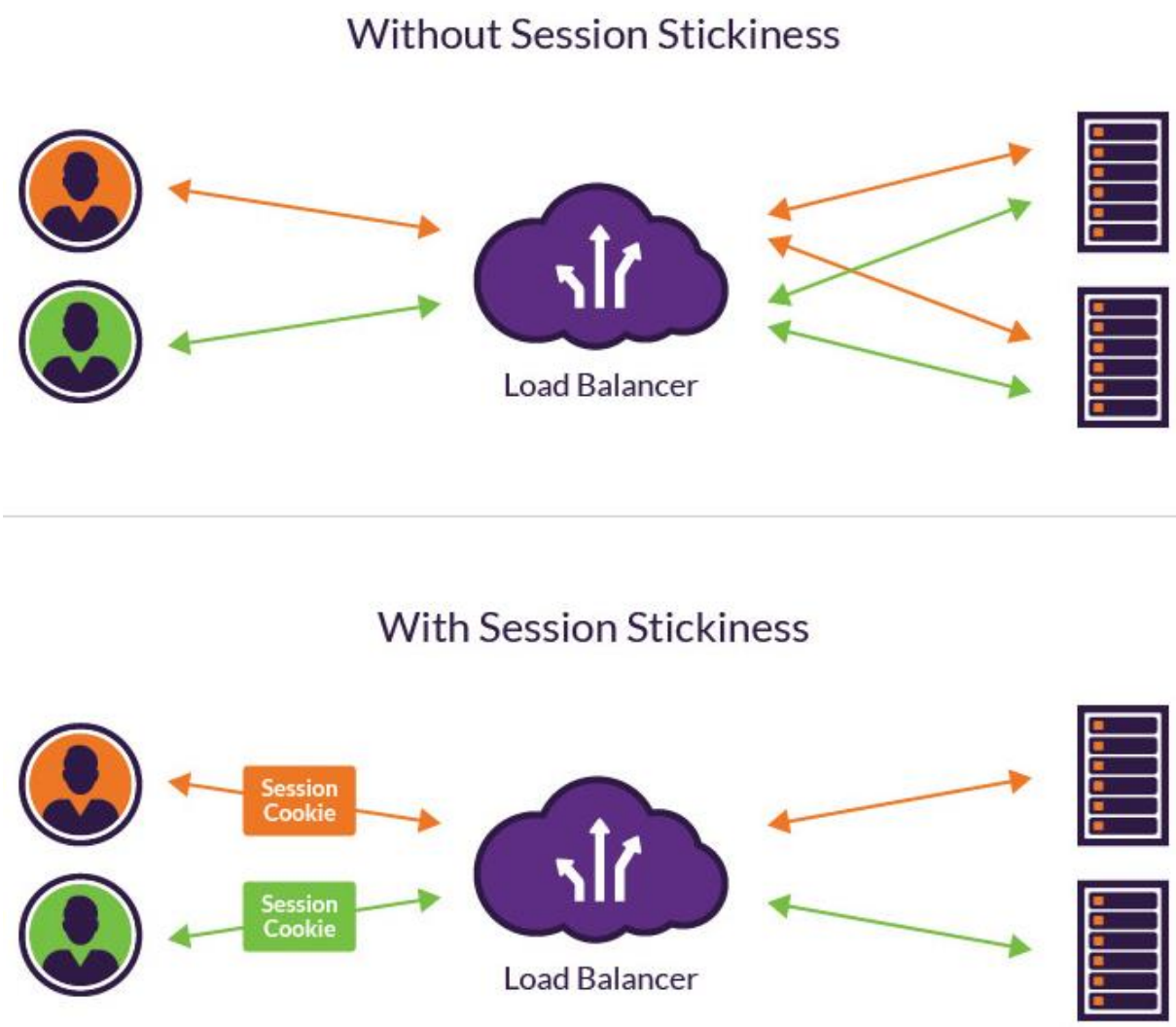

Рисунок 7 – Sticky Sessions.

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 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 взаимосвязи между различными показателями не детерминированы;

– 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.

Создание ИС – это творческий процесс. Существует множество различных классификаций ИС, одна из которых представлена на рисунке 8. Обычно выделяют: ЭС (экспертные системы), ИНС (искусственные нейронные сети), СГА (системы с генетическими алгоритмами), МАС (мультиагентные системы) и другие.

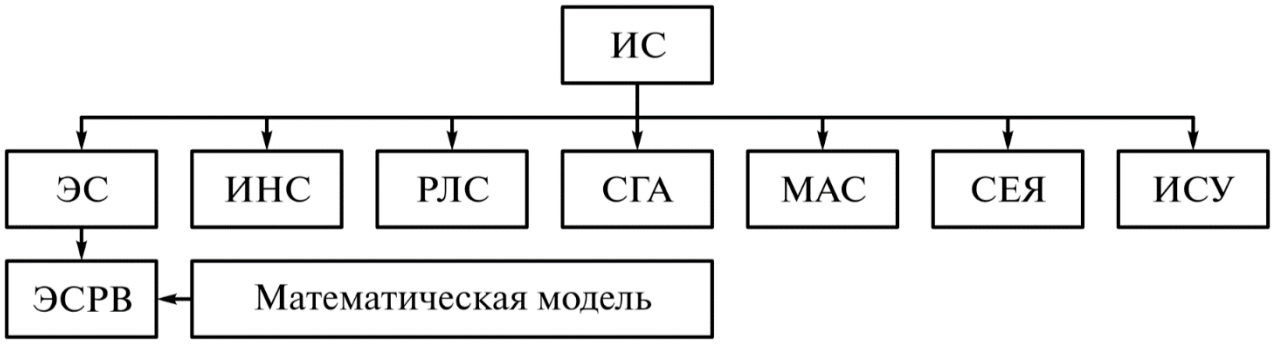

Рисунок 8 – Классификация ИС

1.4 Интеллектуальная система эволюционного моделирования

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

Эволюционное моделирование (ЭМ) – это группа эвристических методов, которые заимствуют принципы и понятийный аппарат у популяционной генетики (рис. 9). Эти методы позволяют осуществлять поиск решения для задач оптимизации [10]. ЭМ эксплуатирует идею «мягких вычислений» [11].

Рисунок 9 – 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.

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.

Методы эволюционного моделирования можно рассматривать также как базу для упражнений в совершенствовании техник программирования отдельных алгоритмов и программных систем [10].

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.

Технологии объектно-ориентированного программирования позволяет реализовать ГА максимально гибко и удобно для программиста при дальнейшем внедрении в информационные системы и сопровождении [10].

Сами ГА, наравне с другими методами, также достаточно разнообразны (рис. 10).

Рисунок 10 – Классификация методов ЭМ

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 [10]. Потомки наследуют характеристики родителей с некоторыми изменениями (мутациями). Таким образом, эволюционный процесс подразумевает выполнение 3-х условий [10]:

– 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.

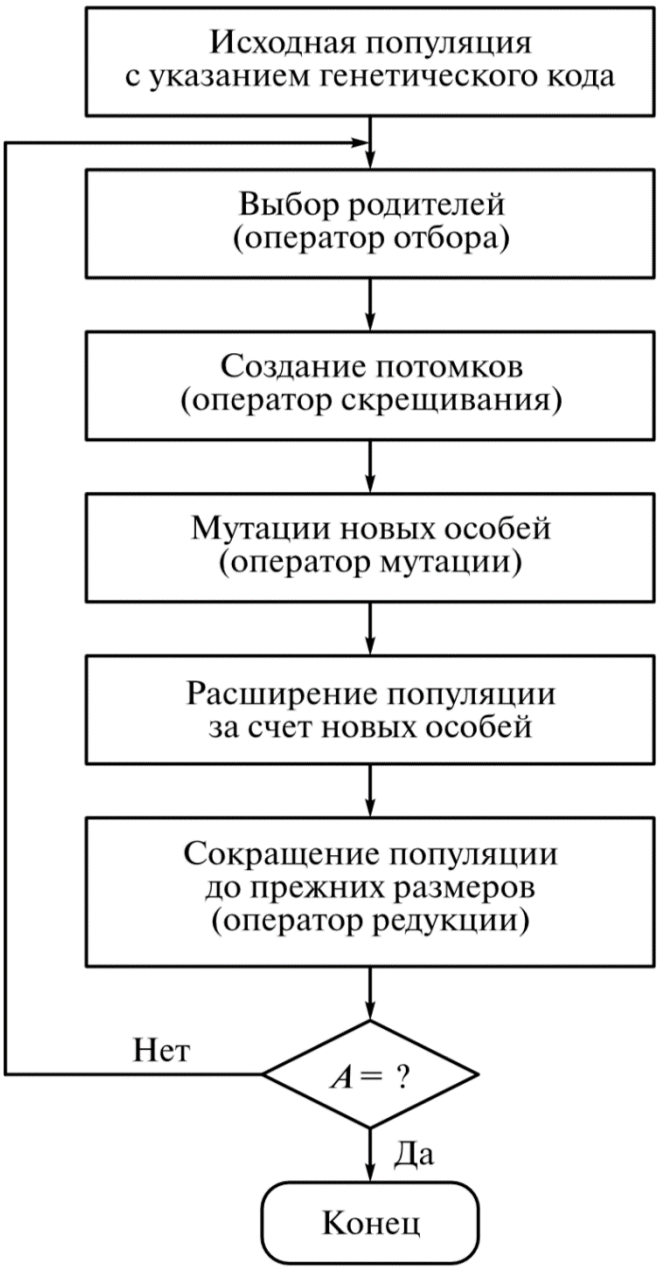

Рисунок 11 – Схема ГА

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.

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. Аверченков В. И., Казаков П. В. Эволюционное моделирование и его

применение / В. И. Аверченков, П. В. Казаков, Москва: ФЛИНТА, 2016. 200 с.
11. Zadeh L. A. Soft Computing and Fuzzy Logic // IEEE Software. 1994. № 6 (11). С. 48–56.