Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки Информатика и вычислительная техника
Кафедра информатики и проектирования систем

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Проектирование и разработка горизонтально масштабируемой распределенной файловой системы

УДК 004.75

Студент

Группа	ФИО	Подпись	Дата
8ВМ4Б	Зензин Алексей Сергеевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ИПС	Ботыгин И.А.	к.т.н., доцент		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. менеджмента	Конотопский В.Ю.	к.э.н., доцент		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент каф. ЭБЖ	Акулов П.А.			

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
и.о. руководителя кафедры ИПС	Демин А. Ю.	к.т.н., доцент		

Томск – 2016 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ООП

Код результатов	Результат обучения (выпускник должен быть готов)	Требования ФГОС, критерии АИОР
Профессиональные компетенции		
Р1	Применять глубокие естественнонаучные и математические знания для решения научных и инженерных задач в области информатики и вычислительной техники.	Требования ФГОС (ОК-1, 2; ПК-1, 5, 6), критерий 5 АИОР (п. 1.1), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р2	Применять глубокие специальные знания в области информатики и вычислительной техники для решения междисциплинарных инженерных задач.	Требования ФГОС (ОК-6, ПК-1, 5, 6), критерий 5 АИОР (п. 1.1, 1.2) , соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р3	Ставить и решать инновационные задачи инженерного анализа, связанные с созданием аппаратных и программных средств информационных и автоматизированных систем, с использованием аналитических методов и сложных моделей.	Требования ФГОС (ОК-2; ПК-5), критерий 5 АИОР (п. 1.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.

Код результатов	Результат обучения (выпускник должен быть готов)	Требования ФГОС, критерии АИОР
Профессиональные компетенции		
Р4	Выполнять инновационные инженерные проекты по разработке аппаратных и программных средств автоматизированных систем различного назначения с использованием современных методов проектирования, систем автоматизированного проектирования, передового опыта разработки конкурентно способных изделий.	Требования ФГОС (ОК-4; ПК-3, 4, 5, 6), критерий 5 АИОР (п. 1.3), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р5	Планировать и проводить теоретические и экспериментальные исследования в области проектирования аппаратных и программных средств автоматизированных систем с использованием новейших достижений науки и техники, передового отечественного и зарубежного опыта. Критически оценивать полученные данные и делать выводы.	Требования ФГОС (ОК-4, 5; ПК-1), критерий 5 АИОР (п.1.4), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р6	Осуществлять авторское сопровождение процессов проектирования, внедрения и эксплуатации аппаратных и программных средств автоматизированных систем различного назначения.	Требования ФГОС (ОК-7; ПК-7), критерий 5 АИОР (п. 1.5), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Универсальные компетенции		
Р7	Использовать глубокие знания по проектному менеджменту для ведения инновационной инженерной деятельности	Требования ФГОС (ОК-6; ПК-4), критерий 5 АИОР

Код результатов	Результат обучения (выпускник должен быть готов)	Требования ФГОС, критерии АИОР
Профессиональные компетенции		
	с учетом юридических аспектов защиты интеллектуальной собственности.	(п. 2.1), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р8	Осуществлять коммуникации в профессиональной среде и в обществе в целом, активно владеть иностранным языком, разрабатывать документацию, презентовать и защищать результаты инновационной инженерной деятельности, в том числе на иностранном языке.	Требования ФГОС (ОК-3; ПК-4, 7), критерий 5 АИОР (п. 2.2), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р9	Эффективно работать индивидуально и в качестве члена и руководителя группы, в том числе междисциплинарной и международной, при решении инновационных инженерных задач.	Требования ФГОС (ОК-4; ПК-7), критерий 5 АИОР (п. 2.3), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.
Р10	Демонстрировать личную ответственность и ответственность за работу возглавляемого коллектива, приверженность и готовность следовать профессиональной этике и нормам ведения инновационной инженерной деятельности. Демонстрировать глубокие знания правовых, социальных,	Требования ФГОС (ОК-5; ПК-7), критерий 5 АИОР (п. 2.4, п. 2.5), соответствующий международным стандартам EUR-ACE и FEANI.

Код результатов	Результат обучения (выпускник должен быть готов)	Требования ФГОС, критерии АИОР
Профессиональные компетенции		
	экологических и культурных аспектов инновационной инженерной деятельности.	Запросы студентов, отечественных и зарубежных работодателей.
Р11	Демонстрировать способность к самостоятельному обучению, непрерывному самосовершенствованию в инженерной деятельности, способность к педагогической деятельности.	Требования ФГОС (ОК-2, ПК-2), критерий 5 АИОР (п. 2.6), соответствующий международным стандартам EUR-ACE и FEANI. Запросы студентов, отечественных и зарубежных работодателей.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки Информатика и вычислительная техника
Кафедра информатики и проектирования систем

УТВЕРЖДАЮ:
и. о. зав. Кафедрой ИПС

_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

магистерской диссертации

Студенту:

Группа	**ФИО**
8ВМ4Б	Зензин Алексей Сергеевич

Тема работы:

Проектирование и разработка горизонтально масштабируемой распределенной файловой системы	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	1. Наименование объекта — распределенное файловое хранилище. 2. Функциональные требования к разрабатываемому решению.
Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Обзор и сравнение распределенных файловых систем. 2. Оценка влияния на эффективность структурирования данных и распараллеливания вычислений. 3. Разработка обобщенной функциональной структуры распределенного хранилища данных. 4. Программная реализация и технологическая схема развертывания компонентов распределенного хранилища данных.
Перечень графического материала	1. Скринщоты приложений-компонентов хранилища

	2. Структурная схема системы

Консультанты по разделам выпускной квалификационной работы	
Раздел	**Консультант**
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Конотопский В. Ю.
Социальная ответственность	Акулов П.А.
Раздел на иностранном языке	Сидоренко Т. В.
Названия разделов, которые должны быть написаны на русском и иностранном языках:	
Обзор и сравнение распределенных файловых систем	

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
доцент каф. ИПС	Ботыгин И. А.	к.т.н., доцент		

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
8ВМ4Б	Зензин Алексей Сергеевич		

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт кибернетики
Направление подготовки Информатика и вычислительная техника
Уровень образования - магистр
Кафедра информатики и проектирования систем
Период выполнения

Форма представления работы:

Магистерская диссертация

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
25.01.2016	*Обзор и сравнение распределенных файловых систем.*	
25.02.2016	*Оценка влияния на эффективность структурирования данных и распараллеливания вычислений.*	
25.03.2016	*Разработка обобщенной функциональной структуры распределенного хранилища данных.*	
25.04.2016	*Программная реализация и технологическая схема развертывания компонентов распределенного хранилища данных.*	
10.05.2016	*Финансовый менеджмент, ресурсоэффективность и ресурсосбережение*	
15.05.2016	*Социальная ответственность*	
20.05.2016	*Раздел на иностранном языке*	

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент каф. ИПС	Ботыгин И. А.	к.т.н., доцент		

СОГЛАСОВАНО:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
и.о. руководителя кафедры ИПС	Демин А. Ю.	к.т.н., доцент		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ВМ4Б	Зензину Алексею Сергеевичу

Институт	ИК	Кафедра	ИПС
Уровень образования	Магистратура	**Направление/специальность**	09.04.01 Информатика и вычислительная техника

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	
Нормы и нормативы расходования ресурсов	
Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
Оценка коммерческого и инновационного потенциала НТИ	*1. Определение экономической эффективности* *2. Оценка научно — технического уровня НИР*
Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок	*1. Организация и планирование работ* *2. Смета затрат на выполнение проекта*
Определение ресурсной, финансовой, экономической эффективности	*1. Определение сроков окупаемости*
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. Линейный график работ	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. менеджмента	Конотопский В. Ю.	к.э.н., доцент		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ВМ4Б	Зензин Алексей Сергеевич		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8ВМ4Б	Зензину Алексею Сергеевичу

Институт	ИК	Кафедра	ИПС
Уровень образования	Магистратура	Направление/специальность	09.04.01 Информатика и вычислительная техника

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения.	*1. Разработка распределенного файлового хранилища. Программный продукт позволяет объединять персональные компьютеры для организации масштабируемого файлового хранилища. Может применяться различными НИИ и небольшими коммерческими организациями, работающими с данными больших объемов*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Профессиональная социальная безопасность. *1.1 Анализ вредных факторов, выявленных при разработке и эксплуатации проектируемого решения.* *1.2 Анализ опасных факторов, выявленных при разработке и эксплуатации проектируемого решения.* *1.3 Обоснование мероприятий по защите исследователя от действия опасных и вредных факторов.*	*1.1 Вредные производственное факторы,* *Электромагнитные излучения.* *Микроклимат;* *Освещенность;* *Монотонность работы.* *1.2 Опасные производственные факторы:* *Возникновение пожара.* *Поражение электрическим током.* *1.3 Мероприятия по защите от вредных факторов согласно нормативным документам:* *СанПиН 2.2.4.548-96;* *СанПиН 2.2.2/2.4.1340-03;* *СП 52.13330.2011.* *Мероприятия по защите от опасных факторов согласно нормативным документам:* *СанПиН 2.2.2/2.4.1340-03;* *ГОСТ Р 12.1.019-2009 ССБТ;* *СанПиН 2.2.1/2.1.1.1200-03.*
2. Экологическая безопасность. *2.1 Анализ влияния разрабатываемого проекта на окружающую среду.* *2.2 Обоснование мероприятий по защите окружающей среды.*	*1.1 Влияние разрабатываемого проекта на окружающую среду:* *Ресурсосбережение.* *1.2 Утилизация отходов (бытовой мусор)*
3. Безопасность в чрезвычайных ситуациях. *3.1 Анализ вероятных ЧС, которые может инициировать объект исследований.* *3.2 Анализ вероятных ЧС, которые могут возникнуть на рабочем месте при проведении исследований.* *3.3 Обоснование мероприятий по предотвращению ЧС и разработка порядка действия в случае возникновения ЧС.*	*3.1 Вероятные ЧС, инициируемые объектом исследования:* *Пожар.* *3.2 Вероятные ЧС, возникающие на рабочем месте:* *Пожары и взрывы;* *Обрушение зданий;* *Ураганы, ливни, заморозки;* *Наводнения, паводки;* *3.3 Эпидемии;Мероприятия по предотвращению наиболее типичной ЧС – пожара, согласно*

	нормативным документам: *НПБ 105-03;* *ППБ 01–03.*
4 Правовые и организационные вопросы обеспечения безопасности. *4.1 Специальные (характерные для проектируемой рабочей зоны) правовые нормы трудового законодательства.* *4.2 Организационные мероприятия при компоновке рабочей зоны.*	*4.1 Описание правовых норм для работ, связанных с работой за ПЭВМ согласно следующим документам:* *Трудовой кодекс Российской Федерации" от 30.12.2001 N 197-ФЗ (ред. от 30.12.2015).* *4.2 Влияние реализации проекта на организацию рабочего места системного администратора*

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
ассистент каф. ЭБЖ	Акулов П.А.			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ВМ4Б	Зензин Алексей Сергеевич		

РЕФЕРАТ

Выпускная квалификационная работа 98 с., 21 рис., 19 табл., 22 источника, 1 приложение.

Ключевые слова: распределенная файловая система, метаописание, многопоточность, Java, NetBeans.

Объектом исследования являются данные, получаемые от ультразвуковых анемометров, содержащие информацию о температуре, влажности, давлении, направлении и силе ветра и т. д. Данные собираются с периодичностью 12,5 мс.

Цель работы – разработка распределенного горизонтально - масштабируемого файлового хранилища

В процессе исследования проводились эксперименты, показывающие повышение производительности системы при применении многопоточности для распараллеливания вычислений и структурировании информации с помощью метаописаний.

В результате исследования был реализован проект распределенного файлового хранилища.

Степень внедрения: проект находится на стадии тестирования.

Область применения: различные НИИ и коммерческие организации, работающие с большими объемами данных.

Экономическая эффективность/значимость работы — проект позволяет минимизировать расходы на организацию хранения больших объемов данных

В будущем планируется внедрить систему в организацию, занимающуюся обработкой больших объемов данных. Также планируется добавить возможность обработки данных, хранимых в системе.

Определения

1. Распределенная файловая система — файловая система, не сосредоточенная на одной машине, а распределенная на многих узлах, имеющих постоянную связь друг с другом.
2. Метаописание — набор метатегов (ключевых слов), позволяющих определить вид информации, хранимый в описываемом файле.
3. Мультиагентная архитектура — способ организации распределенной системы, который характеризуется наличием одного или нескольких управляющих узлов (командных центров) и большого количества подключенных к ним рабочих узлов (агентов).

Оглавление

Введение

В настоящее время человеку в своей производственной и научной деятельности приходится иметь дело с огромным объемов различной информации. Например, система МФЦ (многофункциональные центры предоставления госуслуг) потенциально должна обрабатывать до 50000 документов в день или 12 миллионов в год, что составляет примерно 100 Гб в день и около 25 Тб в год. [1] Также данная проблема достаточно остро стоит для различных научно-исследовательских институтов, собирающих терабайты данных, связанных с проводимыми в них исследованиями. Естественным образом, вместе с получением и обработкой больших объемов данных встает проблема их хранения и структурирования.

На данный момент, рынок программного обеспечения предлагает множество решений для хранения больших данных. Однако, большинство из них для своего нормального функционирования требуют закупки дорогостоящей аппаратной части. Решения же, функционирующие в облаке, на базе уже существующих инфраструктур, тоже подходят не всем организациям, как ввиду затрат на аренду мощностей, так и из-за желания обеспечить дополнительную конфиденциальность данных и хранить их локально, в рамках данной организации.

Таким образом, разработка хранилища, работающего на доступных вычислительных мощностях академических институтов, с минимльными затратами на обслуживание, позволяющее создать единое корпоративное информационное пространство, и возможностью хранить большие и разнородные объемы данных является весьма актуальной задачей.

Разрабатываемая система хранения данных прежде всего ориентируется на применение в различных научно — исследовательских институтах, в которых множество персональных компьютеров (в том числе рабочих мест сотрудников) могут быть использованы для развертывания узлов хранения с последующим их объединением в распределенную систему хранения данных.

1 Обзор и сравнение распределенных файловых систем

1.1 Файловые системы

Файловая система — это некоторый способ организации хранения, именования и доступа к данным на различных носителях информации. Также часто под файловой системой понимают некоторый программный интерфейс, служащий прослойкой между API (интерфейс программирования приложений) доступа к файлам и непосредственно носителем информации.

Таким образом, в широком смысле понятие "файловая система" включает:

1. совокупность всех файлов на диске;
2. наборы структур данных, используемых для управления файлами, такие, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске;
3. комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами. [2]

Существует несколько видов файловых систем (для различных видов носителей, виртуальные файловые системы, специализированные и т.д.). Но, с точки зрения предмета рассмотрения данной работы, больший интерес представляют сетевые или распределенные файловые системы. Как следует из названия, данные системы не сосредоточены на одной машине, а могут охватывать большое количество компьютеров и серверов, не обязательно находящихся в непосредственной территориальной близости друг от друга. Также, одно из главных преимуществ таких файловых систем состоит в том, что они почти неограниченно масштабируются и могут хранить данные любых объемов.

Рассмотрим некоторые распределенные файловые системы, присутствующие на рынке.

1.2 Обзор решений, применяемых для хранения больших данных

1.2.1 Hadoop (HDFS)

Hadoop — проект фонда Apache Software Foundation, свободно распространяемый набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов. Используется для реализации поисковых и контекстных механизмов многих высоконагруженных веб-сайтов, в том числе, для Yahoo! и Facebook. Разработан на Java в рамках вычислительной парадигмы MapReduce, согласно которой приложение разделяется на большое количество одинаковых элементарных заданий, выполнимых на узлах кластера и естественным образом сводимых в конечный результат. [3]

Hadoop является одним из самых распространенных решений на рынке bigdata. Многие крупные IT – вендоры (Amazon, Cloudera, Dell) предлагают построение систем хранения и обработки данных на основе данного проекта.

Создатели Cloudera Hadoop, свободно распространяемого дестрибутива, приводят следующие системные требования:

- «легкая» конфигурация (1U) — 2 шестиядерных процесссора, 24-64 Гб памяти, 8 жестких дисков емкостью 1-2 Тб;
- рациональная конфигурация (1U) — 2 шестиядерных процессора, 48-128 Гб памяти, 12-16 жестких дисков (1 или 2 Тб), подключенных напрямую через контроллер материнской платы;
- «тяжелая» конфигурация для хранилищ (2U): 2 шестиядерных процессора, 48-96 Гб памяти, 16-24 жестких дисках.
- конфигурация для интенсивных вычислений: 2 шестиядерных процессора, 64-512 Гб памяти, 4-8 жестких дисков емкостью 1-2 Тб. [4]

Hadoop разбита на несколько модулей, в их число входит HDFS, представляющий собой распределенную файловую систему. Файлы в данной системе разбиты на блоки, что позволяет хранить файлы больших размеров. Все блоки в HDFS (кроме последнего блока файла) имеют одинаковый размер, и каждый блок может быть размещён на нескольких узлах, размер блока и

коэффициент репликации (количество узлов, на которых должен быть размещён каждый блок) определяются в настройках на уровне файла. Благодаря репликации обеспечивается устойчивость распределённой системы к отказам отдельных узлов. Файлы в HDFS могут быть записаны лишь однажды (модификация не поддерживается), а запись в файл в одно время может вести только один процесс. Организация файлов в пространстве имён — традиционная иерархическая: есть корневой каталог, поддерживается вложение каталогов, в одном каталоге могут располагаться и файлы, и другие каталоги.[3]

Стоит сказать несколько слов об архитектуре файловой системы. HDFS кластер состоит из единственного NameNode (узла имен), являющегося master – сервером. Он отвечает за пространство имен файловой системы и разграничение доступа для клиентов. Также в кластер входят несколько DataNodes (узлов данных), отвечающих непосредственно за хранение данных. HDFS позволяет хранить файлы пользователей в разделенном виде (разбитыми на блоки). В таком случае, блоки хранятся на нескольких узлах данных. Также узлы данных ответственны за проверку, репликацию и восстановление данных. Они же осуществляют непосредственно запись/чтение данных. Узел имен отвечает за операции над файлами (открытие, закрытие, переименование), а также хранит метаописание файлов.[5] Схема HDFS кластера представлена на рисунке 1.

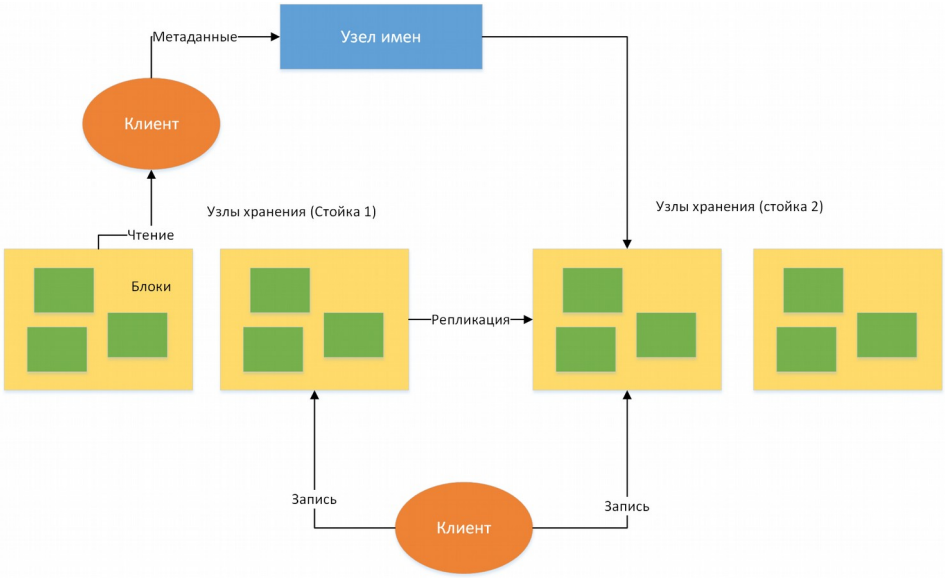

Рисунок 1 - Схема HDFS кластера

Развёртывание экземпляра HDFS предусматривает наличие центрального узла имён, хранящего метаданные файловой системы и метаинформацию о распределении блоков, и серии узлов данных, непосредственно хранящих блоки файлов. Узел имён отвечает за обработку операций уровня файлов и каталогов — открытие и закрытие файлов, манипуляция с каталогами, узлы данных непосредственно отрабатывают операции по записи и чтению данных. Узел имён и узлы данных снабжаются веб-серверами, отображающими текущий статус узлов и позволяющими просматривать содержимое файловой системы. Административные функции доступны из интерфейса командной строки. Как правило, HDFS предоставляется конечному пользователю в виде облачного сервиса, развернутого на мощностях поставщика, либо в виде настроенных кластеров для установки непосредственно у клиента.[3]

1.2.2 Amazon S3

Amazon Simple Storage Service (Amazon S3) — онлайновая веб-служба, предлагаемая Amazon Web Services, предоставляющая возможность для хранения и получения любого объёма данных, в любое время из любой точки сети, так называемый файловый хостинг. С помощью Amazon S3 достигается высокая масштабируемость, надёжность, высокая скорость и недорогая инфраструктура хранения данных. Впервые появилась в марте 2006 года в США и в ноябре 2007 года в Европе. [6]

При работе с Amazon S3 потребитель платит как за использованное пространство, так и за входящий/исходящий трафик.

К сожалению, информация об внутреннем устройстве и принципах построения данной системы отсутствуют в свободном доступе.

1.2.3 Google File System

Проприетарная распределенная файловая система, применяемая для внутренних нужд компании Google. Многие детали ее реализации являются коммерческой тайной, но некоторые факты все же известны — например, данная файловая система позволяет хранить файлы больших объемов разбивая

их на чанки по 64 Мб, а также обеспечивает надежность хранения за счет избыточности данных. Существует свободная реализация данной файловой системы, применяемая для нужд правительства Великобритании.

Google File System строилась исходя из следующих критериев:

1. В состав системы входит большое количество недорого оборудования, поэтому сбои в работе неизбежны. Для их нахождения в системе должен быть предусмотрен инструмент мониторинга.
2. Система должна хранить большое количество разнородных файлов (как маленьких, размером до 100 Мб, так и занимающих несколько гигабайт).
3. Из — за большого числа потенциальных запросов, система должна уметь синхронизировать работу с одним и тем же файлом.
4. Необходимо равномерно распределить нагрузку между отдельными узлами хранения данных.

GFS организована иерархически, в виде древа каталогов. В системе предусмотрен стандартный набор операций с файлами: создание, удаление, открытие, закрытие, чтение и запись. Также поддерживаются резервные копии файлов.

Компонентами системы являются мастер-серверы и чанк-серверы. Один кластер GFS состоит из одной главной машины мастера и множества машин, непосредственно хранящих фрагменты файлов чанк-серверы. Как было сказано выше, файлы в GFS разбиваются на куски фиксированного размера - чанки. Каждый чанк имеет уникальный и глобальный 64 — битный ключ, который выдается мастером при создании чанка. Чанк-серверы хранят чанки как обыкновенные файлы на жестком диске. Для обеспечения надежности хранения данных, каждый чанк реплицируется (обычно, до трех раз).

Еще одной особенностью GFS является то, что она не хранит состав файлов в директориях, вместо этого представляя совокупность файлов в пространстве имен как таблицу, которая отображает путь до файлов в метаданные.

Мастер отвечает за работу с метаданными всей файловой системы.

Метаданные содержат информацию о пространства имен, контроле доступа к данным, положении отдельных чанков. Помимо работы с метаданными, мастер осуществляет всю деятельность, связанную с контролем системы: управление свободными чанками, перемещение чанков между серверами, а также сборкой мусора (сбор более ненужных чанков). При необходимости получить файл, клиент связывается с мастер-сервером и узнает у него расположение нужного чанка. Далее, работу с файлом осуществляет чанк-сервер, хранящий данный фрагмент. При этом, при изменения данных должны отразится во всех репликах данного фрагмента. Процесс записи можно описать следующими шагами:

1. Клиент отправляет сообщение мастеру, содержащее запрос на получение чанка.
2. Мастер передает информацию о первичной реплике, а также обо всех вторичных.
3. Клиент отсылает новые данные во все реплики. Чанк-сервера хранят эти данные в специальном буфере.
4. Когда данные будут отосланы во все реплики, клиент посылает запрос на запись первичной реплике. Она, в свою очередь, устанавливает порядок, в котором будут производится все изменения над файлом (их может быть несколько, от различных клиентов).
5. Первичная реплика пересылает запросы всем остальным репликам, при этом они выполняют точно такую же последовательность действий, как и основная.
6. После получения ответа от всех реплик, первичная реплика отправляет ответ клиенту о завершении операции.

Также, при записи новых чанков, система пытается подобрать для хранения чанк-сервера, находящиеся на максимальном удалении друг от друга в рамках кластера (например, использует различные стойки).[7]

1.3 Заключение

Исходя из основных принципов работы и видов различных файловых

систем, наиболее логично организовать хранилище файлов в виде распределенной файловой системы.

По результату обзора представленных на рынке систем хранения больших данных, можно сделать вывод о том, что по тем или иным причинам (необходимость покупки и обслуживания дорогостоящего оборудования, затраты на аренду мощностей у различных вендоров и пр.) данные решения могут не подойти организациям, желающим с минимальными издержками построить свою систему хранения.

2 Оценка влияния на эффективность структурирования данных и распараллеливания вычислений

2.1 Структура метаописания

Поиск и структурирование информации — одна из важнейших задач, которую для своей корректной работы должна решить любая информационная система. В привычных большинству пользователей файловых системах (файловая система Windows, Linux системы, различные облачные хранилища) для решения данной проблемы используется распределение файлов в иерархической системе каталогов.

В разрабатываемой файловой системе было решено использовать другой подход к структурированному хранению данных — ключевые слова и метаописание. Метаописание — некоторый набор метатегов (ключевых слов), позволяющих определить вид информации, хранимый в описываемом файле. Это понятие тесно связано с метаданными - «данными о данных», описывающими внутреннее устройство, содержание, предназначение или иные характеристики различных файлов.

Можно по-разному классифицировать виды метаданных — например, данные, предназначенные для чтения человеком или машиной, или разделение на контентные и административные. Первые служат для описания всех аспектов конкретного объекта, вторые объединяют несколько различных групп, охватывающих вопросы, связанные с управлением информационным ресурсом (архивирование, контроль доступа, схемы хранения данных и т. д.). Существует несколько наборов метаданных, самый широко известный из которых — дублинское ядро.[8] Данный набор применяется для описания очень широкого спектра данных и часто используется различными информационными системами. В дублинское ядро входят несколько элементов метаданных, описывающими сущность (например, название, автор, тип, формат, предмет и т. д.). Каждый элемент имеет ряд параметров: имя, идентификатор, описание. Построение метаописания в данной работе опирается на те же принципы: используются элементы-параметры, имеющие имя и некоторое значение,

показывающее то или иное свойство заносимого файла.[9]

В настоящей работе именно концепция метаданных взята за основу организации и ведения внутренней структуры хранилища данных. Таким образом, подход с использованием метаописания, не предполагает наличия иерархий каталогов, но с каждым файлом ассоциируется некоторый набор метаданных, отражающий его внутреннее содержимое. Метаописания же, в свою очередь, группируются в файлы-каталоги, служащие для поиска нужной информации (рисунок 2). Одно из важных преимуществ такого подхода — возможность получения выборок файлов по ключевым словам.

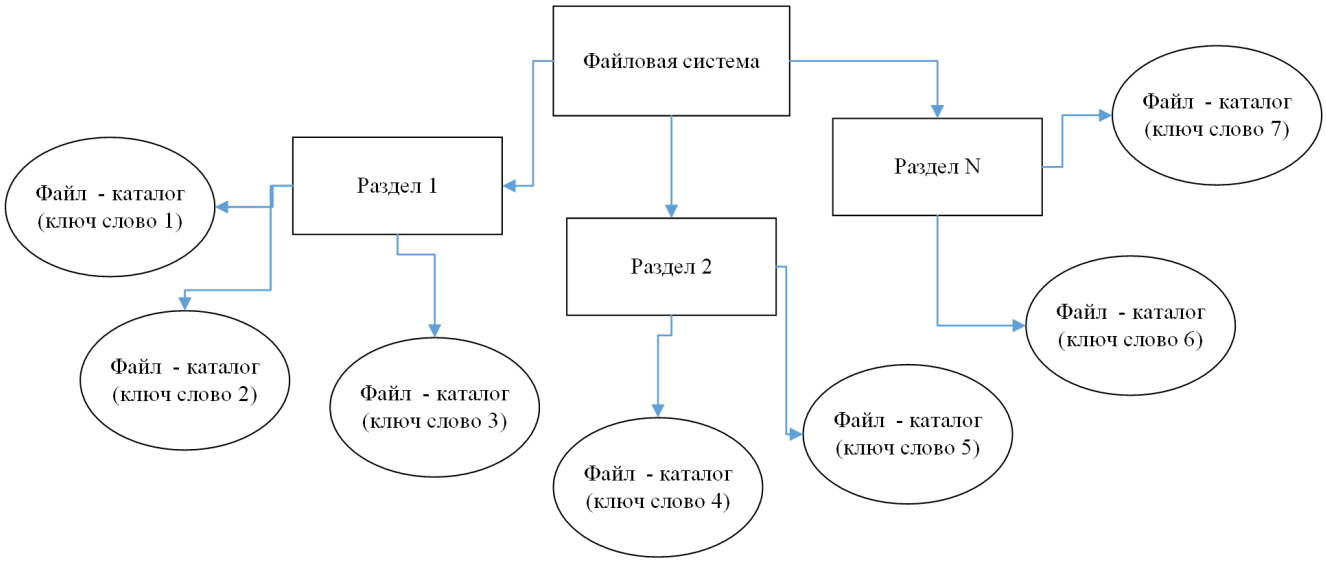

Рисунок 2 — Структура метаданных хранилища

Например, строка метаописания одного файла метеорологических наблюдений с ультразвукового термоанемометра выглядит следующим образом:

meta@fn:201531-1-Tue Apr 05 18-39-02 NOVT 2016@fs:89543654@kw:2015@dt:2016-04-05T12-39-02-160521Z@us:common@kw:Март@kw:1@sf:201531-1-Tue Apr 05 18-39-02 NOVT 20162016-04-05T12-39-02-160521Z@id:2@

Как видно, формат строки метаописания достаточно прост и может быть прочитан без предварительного парсинга. Отдельные части отделены друг от

друга символом «@». Начало строки (_meta_) является идентификатором, позволяющим понять хранилищу, что здесь начинается метаописание очередного файла. Последующие элементы выступают в качестве отдельных параметров и сопровождаются именами (например, «kw:») для определения роли того или иного параметра. Ниже приведен список параметров:

1. kw – ключевое слово;
2. fn – имя файла(пользовательское);
3. sf – имя файла(системное);
4. dt – дата создания файла;
5. fs – размер файла(в байтах);
6. us – раздел;
7. id – номер агента, содержащего данный файл.

Особо следует пояснить параметр “us”. На данном этапе реализации файловой системы в ней предусмотрены «разделы», позволяющие группировать файлы, относящиеся к одной предметной области (рисунок 3).

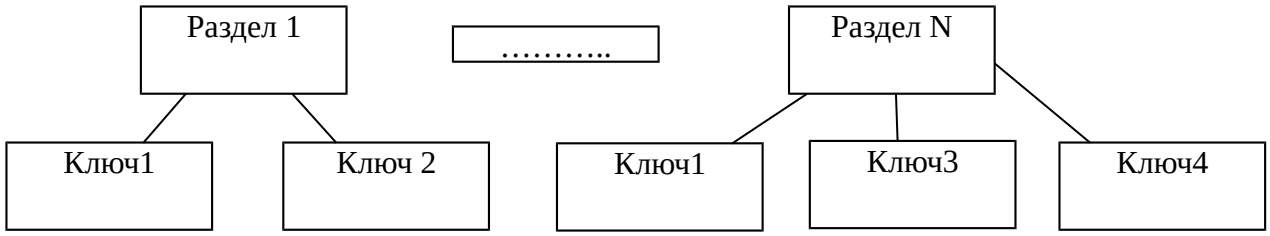

Рисунок 3 — Схема разделов файлового хранилища

Сделано это как из соображений производительности, так и с точки зрения тематического структурирования информации. Если в файловой системе присутствуют несколько непересекающихся предметных областей (например, данные о сотрудниках и показания замеров автоматических метеостанций) то логично будет разделить эти данные на разделы с возможностью дублирования ключевых слов в разных разделах. Таким образом, запрос в раздел показаний замеров, содержащий поиск по ключевому слову «2016» выдаст все файлы, содержащие собранные данные за 2016 год. В свою очередь, аналогичный запрос в раздел сотрудников выдаст, например, список принятых на работу в

2016 году. При таком подходе вполне понятен и выигрыш в производительности, поскольку сокращается количество файлов, среди которых идет поиск. Для наглядности приведем следующий эксперимент. Пусть в нашу файловую систему поступило 10 000 различных файлов. При этом, занесем их в систему без метаописания (не вводя ключевых слов). Естественно, при таком подходе все файлы будут храниться в одном списке, по которому и будет осуществляться поиск. Теперь занесем эти же файлы, но добавим к их описанию ключевые слова (в нашем примере их будет 2). При этом, наблюдается следующее распределение файлов:

1. Ключевое слово 1 — 6 000 файлов.
2. Ключевое слово 2 — 4 000 файлов.

Исходя из этого списка, поиск по запросу «ключевое слово 2» будет вестись среди 4000 файлов, но не среди 10000 (как в первом случае). В данном случае, время поиска сократилось более чем на 50 %.

2.2 Многопоточность в работе хранилища

Природа распределенных файловых систем такова, что им приходится работать с большим количеством клиентов и передавать огромные объемы данных. Таким образом, важным аспектом работы хранилища, непосредственно связанным его с производительностью, является возможность одновременно обрабатывать большое количество запросов. Распараллеливание вычислений даже на одном компьютере, имеющем несколько ядер, является нетривиальной задачей. В настоящей работе для распараллеливания вычислений используется концепция многопоточности, естественным образом реализованная в языке Java.[10]

Приведем небольшой пример, иллюстрирующий данную проблему. Имеется некоторая система хранения, обслуживающая клиентов по данной схеме (рисунок 4).

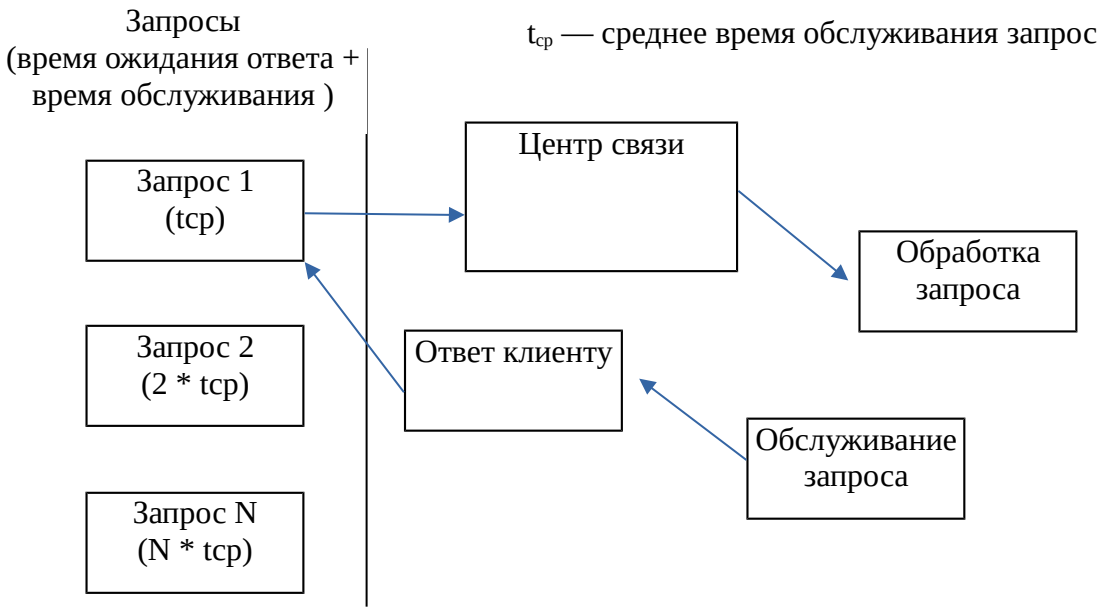

Рисунок 4 — Схема обслуживания клиентов абстрактной системы хранения

Теперь рассмотрим задачу занесения в данную систему некоторого количества файлов. При этом будем измерять время обработки каждого запроса вместе со временем непосредственной передачи файла (т.е. фактическое время обслуживания клиента). Также будем считать, что запросы на занесение файлов приходят одновременно и встают в очередь на выполнение. Ниже, в таблице 1 представлены результаты имитационного моделирования для 5-ти файлов.

Таблица 1 — Результаты имитационного моделирования

Размер (Мб)	Время передачи файла(с.)	Время ожидания(с.)
87,5	74	$0 + t_{об}$
87,5	66	$74 + t_{об}$
87,5	65	$140 + t_{об}$
87,5	61	$201 + t_{об}$
87,5	65	$266 + t_{об}$

Величина $t_{об}$ обозначает время, необходимое для обслуживания клиента (интерпретация сообщения и поиск файла) и в данном случае составляет менее 100 мс.

Таким образом, время ожидания для 5-го файла составит почти 3

минуты, что, естественно, неприемлемо. Если же обработка запросов идет параллельно, то время ожидания каждого клиента составит $t_{об}$. Как видно из результатов наблюдений, данная ситуация требует принятия мер для обеспечения параллельной обработки запросов.

Возможны несколько вариантов решения данной проблемы:

1. создание большого количества «центров связи» для приема и обработки запросов;
2. распараллеливание процесса обработки запросов и делегирование каждого входящего запроса отдельным потокам в рамках одного «центра связи».

Первый вариант обладает рядом очевидных недостатков — во-первых, он не решает проблему нерационального использования мощностей отдельно взятого центра связи, который будет продолжать выполнять всю работу в одном потоке (при условии, что даже бюджетные современные ПК обладают процессорами с несколькими ядрами, не говоря уже об серверах). Во-вторых, это приведет к ненужному разрастанию и усложнению инфраструктуры хранилища. В-третьих, количество центров связи все равно конечно и, таким образом, конечно и число запросов, одновременно обрабатываемых системой.

Второй подход лишен данных недостатков, но, все же наиболее эффективным подходом является комбинация двух предложенных вариантов. Распараллеливание процессов в одном центре связи позволит более полно использовать его ресурсы, а создание нескольких дополнительных центров не только ускорит отклик системы, но и обеспечит дополнительную отказоустойчивость в случае выхода из строя одного из центров связи. Для данной разработки был выбран комбинированный вариант, с распараллеливанием работы в рамках одного командного центра, а также с возможностью развертывания дополнительных дублирующих центров.

3 Разработка обобщенной функциональной структуры распределенного хранилища данных

3.1 Анализ требований. Выбор средств разработки

Перечислим ключевые требования, которым должна удовлетворять разрабатываемая система хранения данных:

1. Платформонезависимость.
2. Низкие системные требования.
3. Горизонтальная масштабируемость.
4. Быстрота поиска нужной информации.
5. Возможность обрабатывать несколько запросов одновременно.

Некоторые из этих требований были рассмотрены в главе 2 (пункты 4 и 5), где были описаны способы их удовлетворения.

Для обеспечения платформонезависимости в качестве языка программирования был выбран язык Java, поскольку виртуальная машина JVM реализована для огромного количества платформ. В качестве среды разработки было решено использовать IDE NetBeans.

3.2 Выбор архитектуры

В основу разрабатываемой системы хранения легла мультиагентная архитектура (рис. 5).

Ядром хранилища является приложение-командный центр. Оно осуществляет коммуникацию с внешней средой(веб-интерфейсы, клиентские приложения) и отслеживает состояние приложений-агентов. При необходимости обеспечения отказоустойчивости, может быть несколько дублирующих командных центров. Приложения-агенты устанавливаются на компьютеры, которые будут использоваться для хранения данных. При этом, каждое приложение-агент имеет постоянную связь с командным центром. Таким образом, облачная система легко масштабируется до любых размеров.

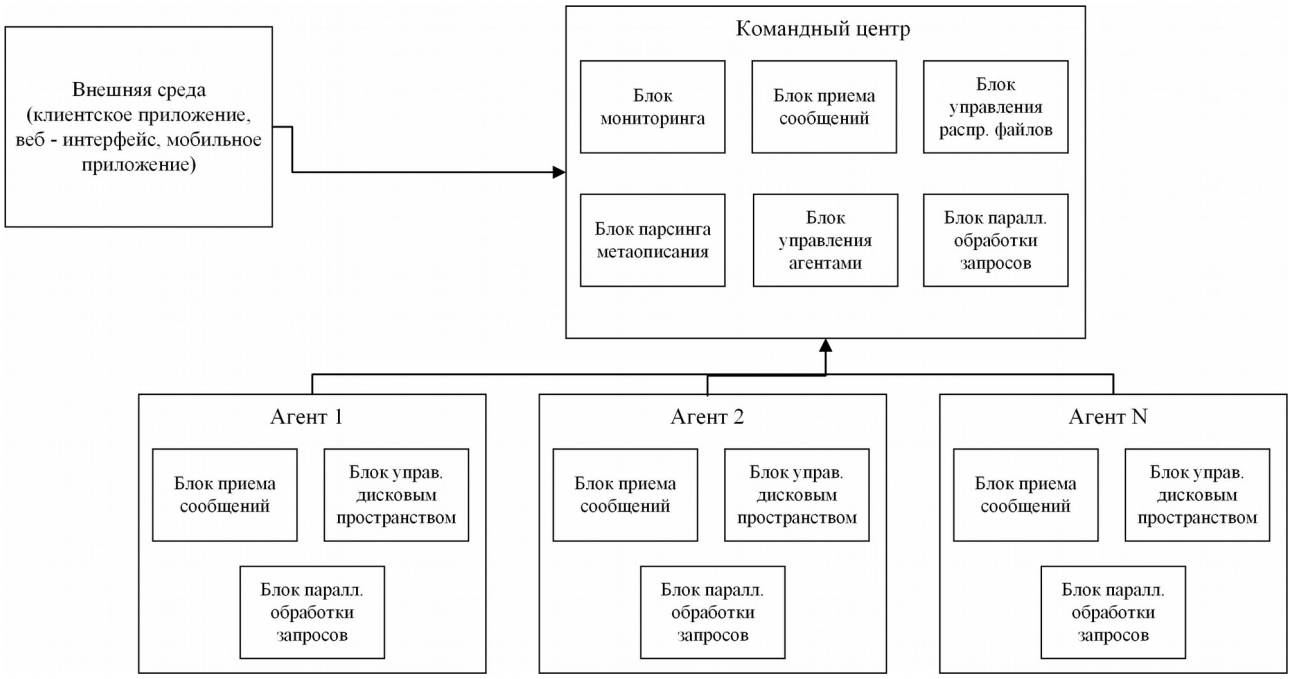

Рисунок 5 — Схема мультиагентной архитектуры

3.3 Приложение «Командный центр»

Рассмотрим каждую часть хранилища. Первой составляющей системы хранения является приложение - «командный центр». Как было сказано выше, данное приложение играет роль ядра системы. Можно выделить следующий список его функций:

1. Осуществление коммуникации с внешней средой(прием сообщений от пользователей).
2. Прием сообщений от приложений - «агентов», в том числе включение новых узлов в систему.
3. Мониторинг доступных ресурсов.
4. Хранение метаописаний всех файлов системы.
5. Хранение информации об узлах хранилища(приложениях - «агентах»).
6. Поиск агентов, содержащих определенный файл.

Важным требованием к работе командного центра является сокращение времени выдачи файла пользователю, а также возможность обрабатывать сразу несколько запросов (от одного или от нескольких клиентов). Поэтому на этапе проектирования было решено использовать возможности многопоточного программирования и выделить отдельные потоки для приема сообщений,

мониторинга агентов и графического интерфейса приложения. Также, по мере необходимости, создаются новые, «рабочие» потоки, служащие для выполнения задач поиска агента, файла, списка файлов и т. д. Схема реализации данного решения приведена на рис. 6

Рисунок 6 — Схема приложения «Командный центр»

Поток-приемник получает сообщение из внешней среды, интерпретирует его и создает необходимый «рабочий поток», непосредственно выполняющий всю необходимую работу. Благодаря такому подходу, приемник сообщений тратит минимум времени на взаимодействие с клиентом и способен обслужить большее количество запросов. Поток мониторинга запускается с некоторой заданной периодичностью и посылает запросы всем клиентам, находящимся онлайн. В качестве ответа, поток получает информацию о свободном и занятом дисковом пространстве агента. Если ответ не был получен, то агент переводится в список оффлайн и не опрашивается в дальнейшем.

Также командный центр хранит ряд важных сведений, необходимых для корректной работы системы:

2. Список агентов (с основной информацией о них: ip – адрес, порт, общий объем доступного дискового пространства и какая его часть свободна).

Данная информация нужна как для поиска агентов, так и для восстановления работоспособности системы после аварийных ситуаций.

3. Метаописания всех файлов системы. На каждое ключевое слово создается свой файл, содержащий данные о всех файлах, включающих данное ключевое слово в своем метаописании.
4. Информация о других командных центрах(если они присутствуют в системе).

3.4 Приложение «Агент»

Приложения-агенты осуществляют непосредственно хранение данных. Список функции данных приложений включает следующие пункты:

1. Прием сообщений от командного центра.
2. Хранение и выдача файлов по запросу.
3. Учет доступного дискового пространства.
4. Хранение метаописаний файлов, находящихся у данного агента.

Также, как и приложение-командный центр, агенты должны иметь возможность обрабатывать несколько запросов одновременно. Схема потоков, функционирующих в приложении, похожа на аналогичную схему командного центра (рис. 7).

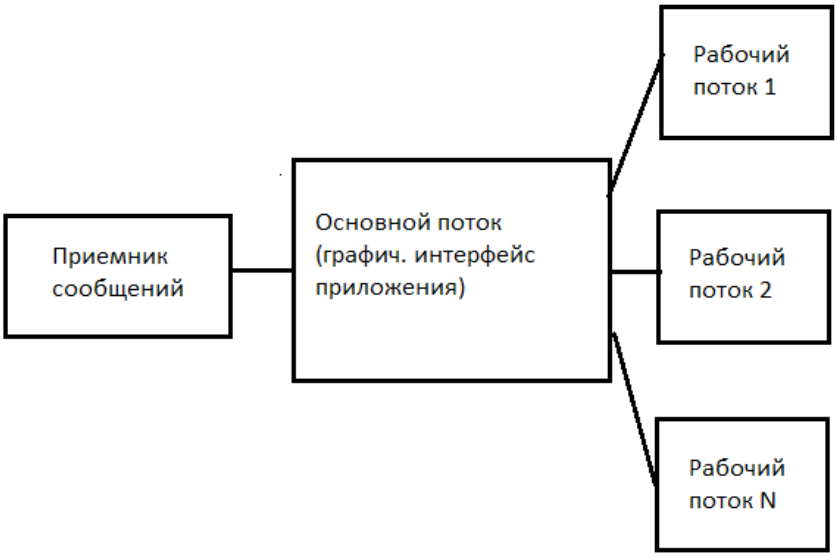

Рисунок 7 — Схема приложения «Агент»

Как видно из рисунка, в приложении постоянно работают основной поток (графический интерфейс) и поток-приемник сообщений. Рабочие потоки создаются по мере необходимости.

При первом запуске, приложение-«агент» требует некоторой первоначальной настройки. Во-первых, необходима информация о существующих в системе командных центрах (содержится в отдельном файле), во-вторых приложению нужна выделенная «рабочая область» (именно ее агент будет использовать для хранения файлов). Таким образом, пользователь, устанавливающий на своей машине приложение - «агент», сам решает, сколько и на каких логических томах агенту доступно дискового пространства.

3.5 Приложение «Файловый менеджер»

Важной частью инфраструктуры хранилища данных является инструмент, служащий для взаимодействия конечного пользователя с системой, т. е. инструмент, позволяющий манипулировать файлами в хранилище. Таким «инструментом по умолчанию» является приложение -файловый менеджер.

Приложение реализует все действия, составляющие необходимый минимум функционала любого файлового менеджера:

3. Поиск файла.
4. Копирование файла в локальную папку.
5. Переименование файла.
6. Удаление файла.
7. Изменение набора ключевых слов.

Последний пункт продиктован спецификой реализации логического структурирования информации (см пункт 2…).

Приложение взаимодействует с командным центром хранилища данных. Стоит отметить, что в командном центре реализован формализованный набор команд, позволяющий взаимодействовать с файловой системой. Для удобства была написана библиотека, выполняющая роль API и служащая высокоуровневой оберткой над данным набором команд, тем самым

значительно облегчая процесс программной интеграции взаимодействия с распределенной файловой системой в прикладное программное обеспечение. Однако, если необходимо, программист может написать собственные библиотеки для работы с хранилищем и использовать их в своих приложениях, опирающихся на данную систему хранения данных.

Схема приложения представлена на рисунке 8.

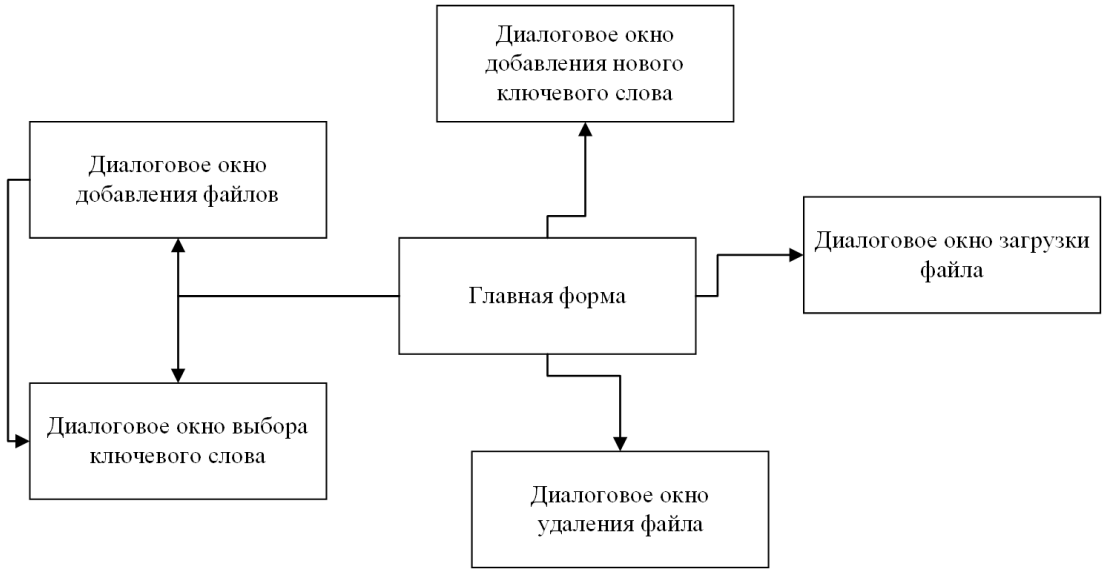

Рисунок 8 — Схема приложения «Файловый менеджер»

Большинство элементов файлового менеджера представляют собой элементы графического интерфейса пользователя. Также важной составляющей является API для взаимодействия с хранилищем.

3.6 Прием сообщений системой

Рассмотрим более подробно процесс приема сообщений системой из внешней среды. Стоит отметить, что детали внутреннего устройства хранилища скрыты от клиента, таким образом, все сообщения принимает только командный центр. Ниже, на рисунке 9, представлена схема последовательных действий системы при получении от внешней среды запроса на выдачу файла.

Клиент передает отправляет сообщение командному центру, содержащее запрос на выдачу определенного файла. В сообщение входит метаописание запрашиваемого файла, а также ip – адрес и порт сокета, на который нужно

отправить файл. Поток-приемник командного центра интерпретирует входящее сообщение и создает рабочий поток, задача которого — найти агента, содержащего нужный файл. Если такой агент не найден, то клиент получит сообщение об ошибке и невозможности получить файл. Может быть две причины возникновения подобной ситуации:

1. Неправильно составленное метаописание запрашиваемого файла. Например, указано лишнее ключевое слово (при поиске файла применяется конъюнкция всех ключевых слов, указанных в метаописании) или неправильное имя файла.
2. Агент, содержащий данный файл, в текущий момент находится в состоянии оффлайн. Данная ситуация требует вмешательства системного администратора для выяснения причины отключения агента от системы.

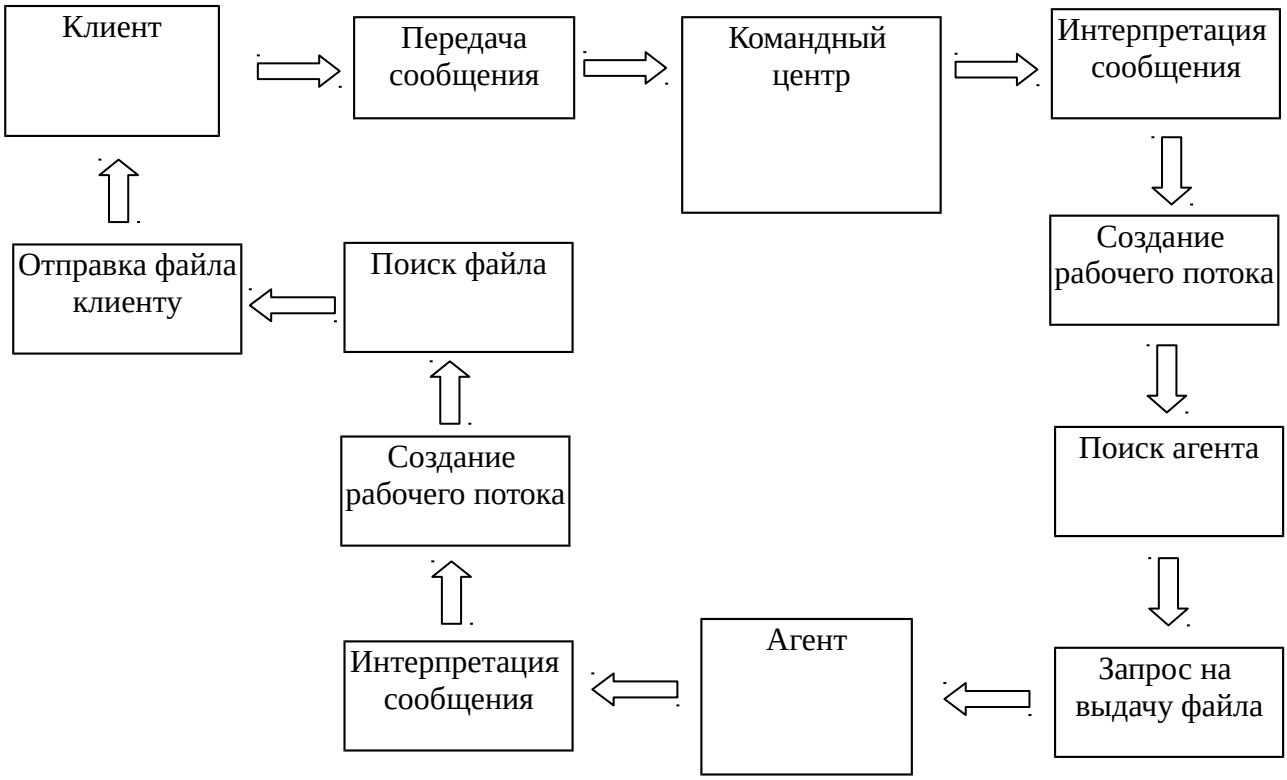

Рисунок 9 — Последовательность действий системы при приеме запроса на выдачу файла

После того, как нужный агент был найден, рабочий поток отправляет ему сообщение о необходимости передать файл на ip – адрес и порт, предоставленные клиентом. Поток-приемник сообщений агента принимает и

интерпретирует запрос, затем создает рабочий поток, осуществляющий поиск запрашиваемого файла. Далее, найденный файл передается клиенту. Если файл не был найден(данная ситуация может возникнуть при непосредственном удалении файла из рабочей области агента, минуя интерфейс файлового хранилища, без удаления соответствующих записей из каталога агента и командного центра), то клиент получает сообщение об ошибке. Эта ситуация также требует вмешательства системного администратора.

Некоторые запросы можно выполнить без непосредственного участия агентов. Ниже, на рисунке 10, представлена схема действий системы при поступлении запроса на получение списка файлов.

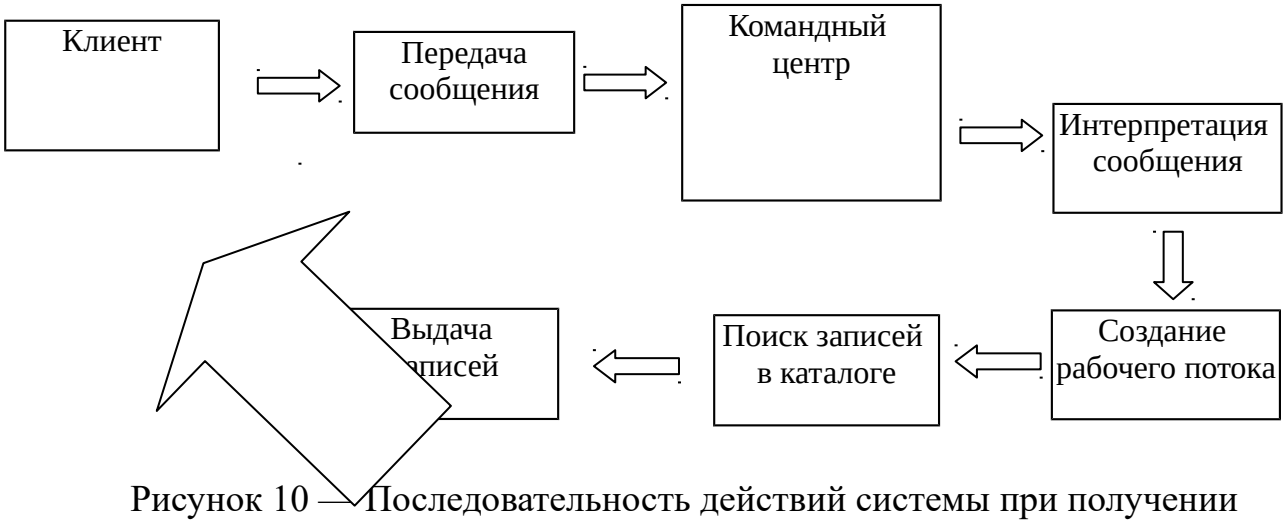

Рисунок 10 — Последовательность действий системы при получении запроса на выдачу списка файлов

Аналогично предыдущему случаю, клиент отправляет запрос командному центру. После интерпретации, командный центр создает рабочий поток, осуществляющий поиск по каталогу файлов и выбирающий те записи, метаописание которых удовлетворяет заявленным критериям поиска. Поскольку речь не идет о непосредственной передачи файлов клиенту, то нет необходимости устанавливать связь с агентом, содержащим очередной файл; рабочий поток передает только метаописания файлов.

4 Программная реализация и технологическая схема развертывания компонентов распределенного хранилища данных

4.1 Развертывание системы

Процесс развертывания распределенного файлового хранилища можно условно разделить на два этапа:

1. развертывание командного центра;
2. развертывание и подключение агентов.

Рассмотрим каждый этап подробнее. Дистрибутив приложения «Командный центр» состоит из одного -jar файла CommandCenter.jar. Данный файл необходимо скопировать на машину, которая будет исполнять роль командного центра. Запуск данного файла откроет главное окно командного центра (рисунок 11).

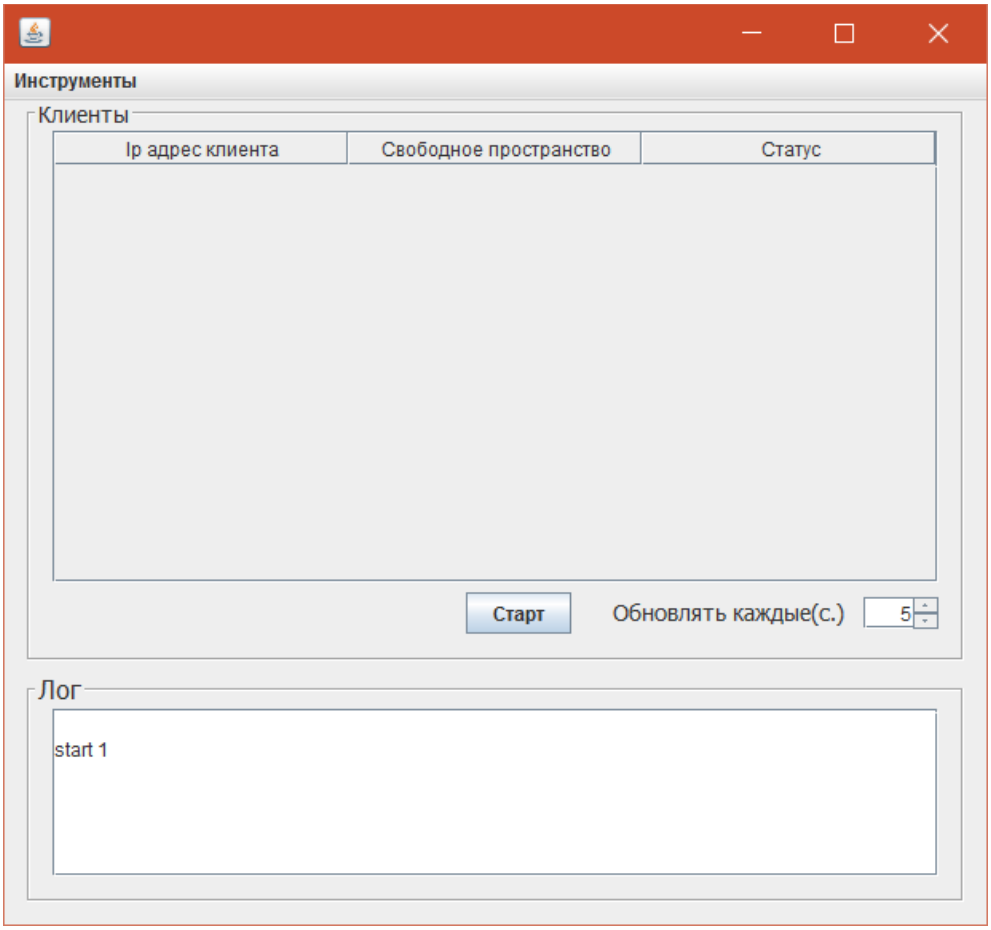

Рисунок 11 — Главное окно командного центра

После этого для запуска командного центра необходимо нажать кнопку «Старт». Подключение настроится автоматически, приложение готово к

подключению агентов. При желаний, можно указать частоту мониторинга агентов (по умолчанию каждые 5 секунд).

Следующий шаг — развертывание и подключение агентов. Первое, что необходимо сделать — это подготовить дистрибутив приложения. Он состоит из одного -jar файла Agent.jar и файла cclist, сгенерированного командным центром после запуска. Данный файл находится в том же каталоге, что и файл CommandCenter.jar и его необходимо скопировать в дистрибутив приложения-агента. После копирования и запуска агента на машине, выбранной в качестве узла хранения данных, появится главное окно приложения (рисунок 12).

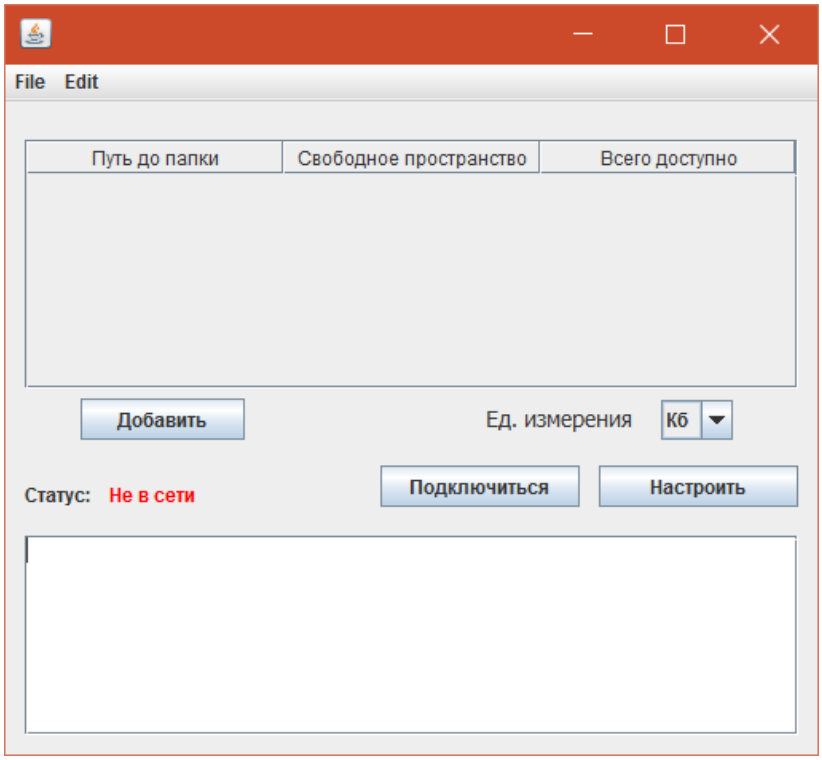

Рисунок 12 — Главное окно приложения-агента

Далее необходимо указать хотя бы одну «рабочую область», которую агент будет использовать для хранения данных. Для этого необходимо нажать кнопку «Добавить», в результате чего появится новое диалоговое окно. Окно, с примером настройки области приведено на рисунке 13.

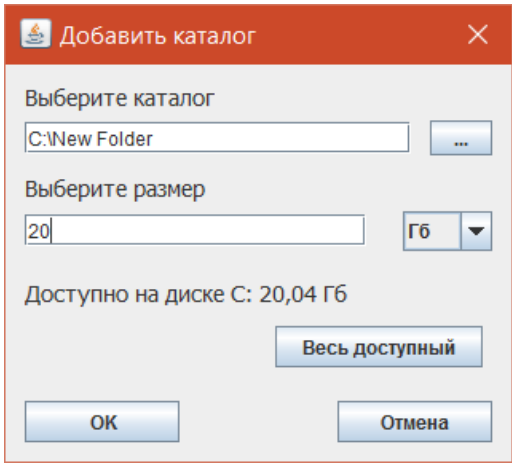

Рисунок 13 — Диалоговое окно «Добавить каталог»

Как видно из рисунка, для хранения файлов был выбран каталог C:\New Folder, доступный агенту объем был задан равным 20 Гб (из доступных 20,04 на диске C). На каждом логическом разделе может быть только одна рабочая область. Если логический раздел полностью отдается в распоряжение агенту, то можно нажать кнопку «Весь доступный», таким образом, весь свободный объем будет выделен агенту. Теперь, когда задана рабочая область, необходимо нажать кнопку «Подключиться» для создания связи с командным центром. В результате статус агента изменится, а командный центр получит информацию о новом агенте (рисунок 14).

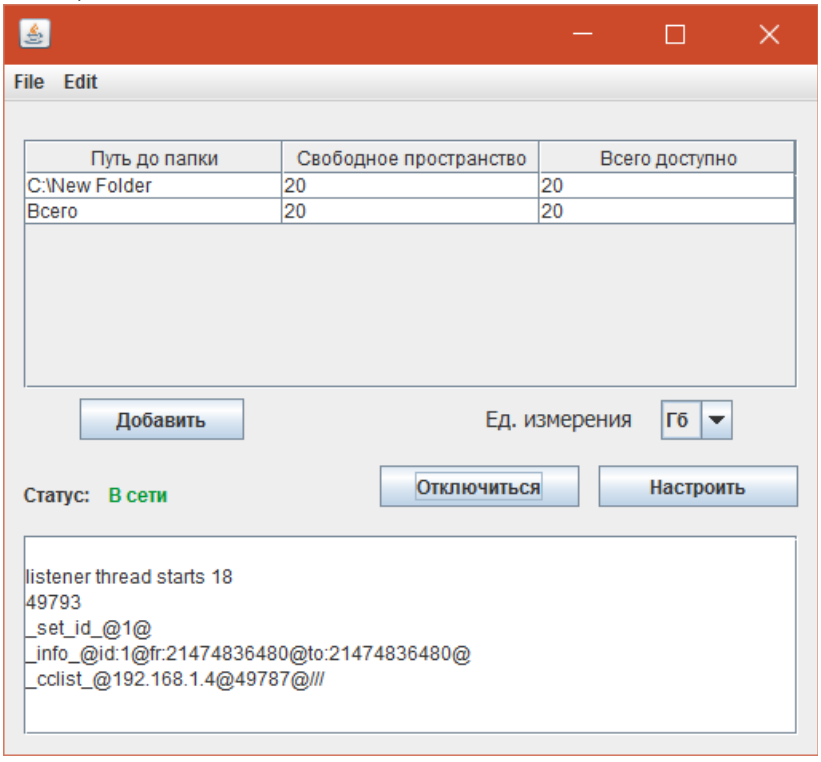

Рисунок 14 — Главное окно приложения агента с информацией о

каталогах

Информация об агенте отражается в главном окне командного центра (рисунок 15).

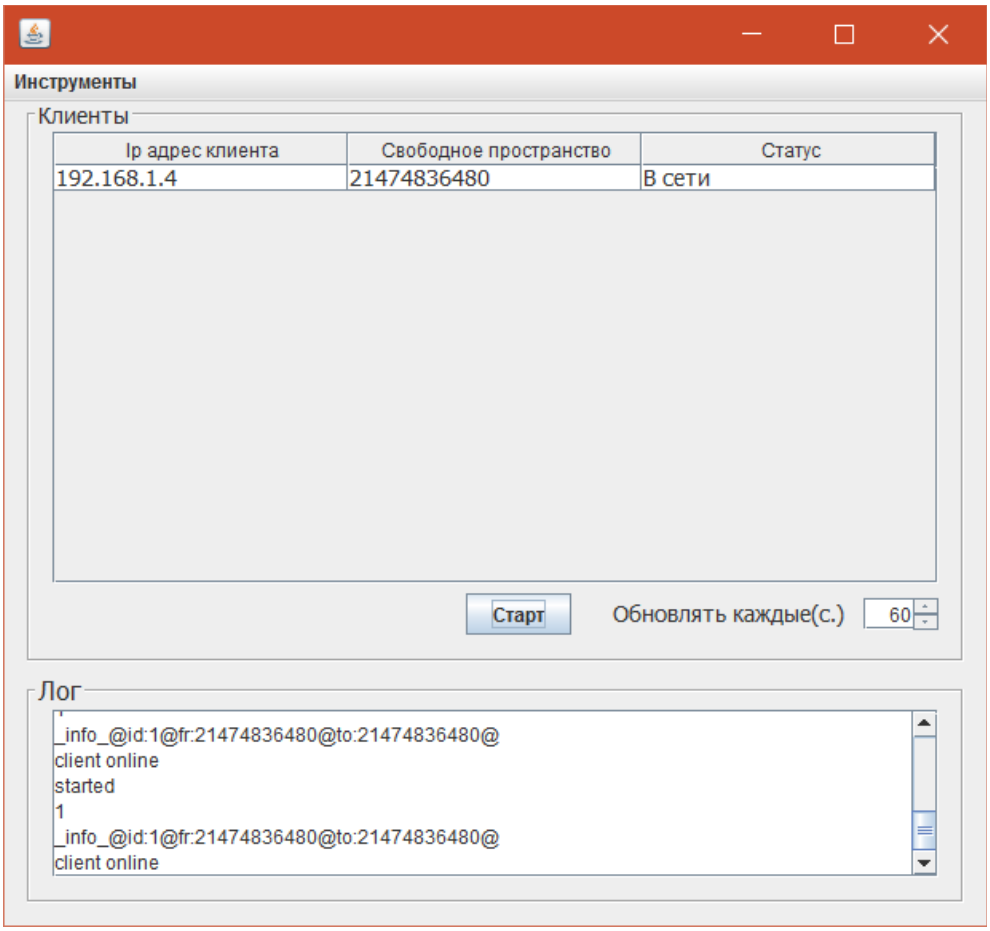

Рисунок 15 — Главное окно командного центра с информацией о новых агентах

Теперь агент входит в состав инфраструктуры хранилища и может принимать файлы на хранение. При необходимости увеличить число узлов хранения можно развернуть дополнительные приложения - «агенты» согласно описанной выше схеме.

4.2 Занесение файлов в систему с помощью приложения «Файловый менеджер»

Для работы с хранилищем предусмотрен инструмент по умолчанию — файловый менеджер. Главное окно приложения приведено на рисунке 16. Продемонстрируем, как с помощью данного приложения можно занести новый файл в систему хранения. При нажатии пункта меню «Добавить» появится

диалоговое окно добавления нового файла (рисунок 17).

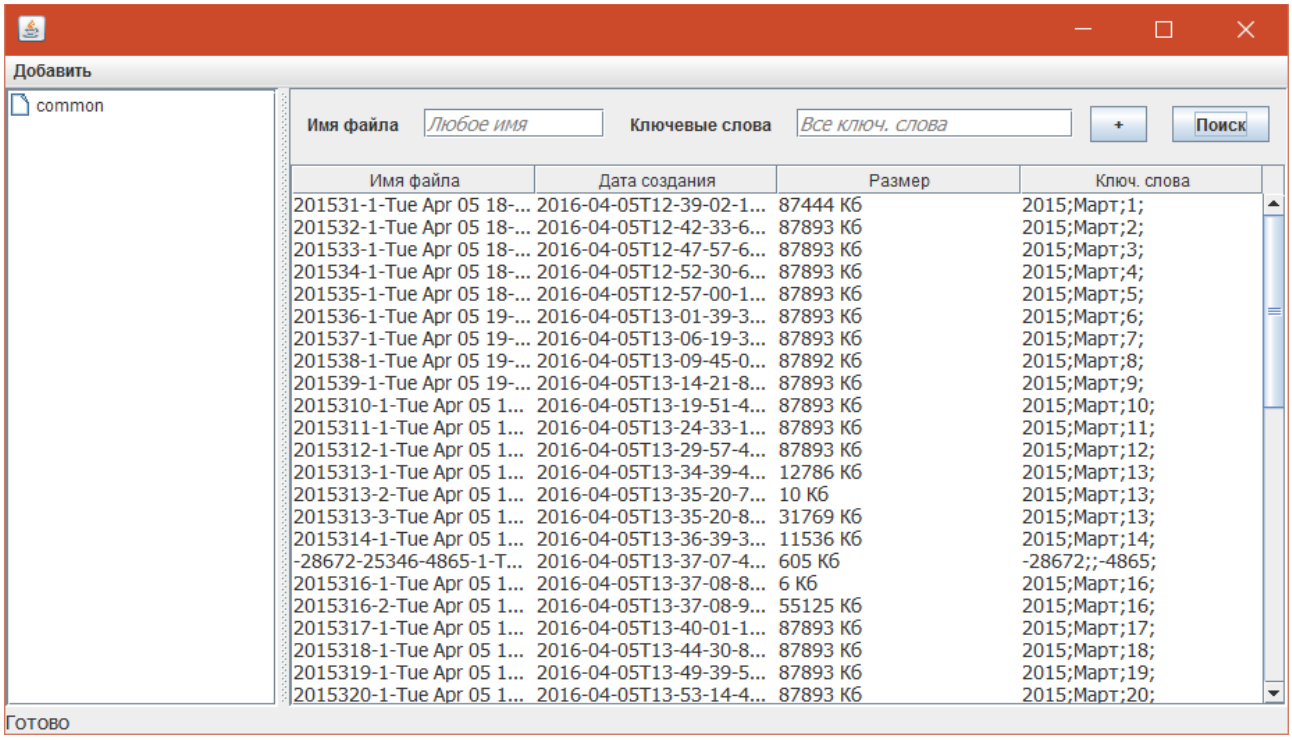

Рисунок 16 — Главное окно приложения «Файловый менеджер»

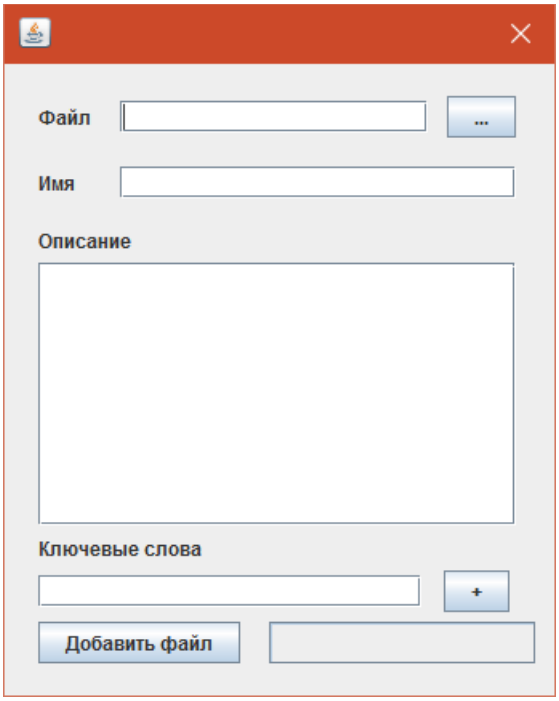

Рисунок 17 — Форма добавления нового файла в систему

Выберем файл и введем информацию о нем (имя, описание, набор ключевых слов). После заполнения соответствующих полей и нажатия кнопки «Добавить», новый файл будет загружен в систему хранения (рисунок 18)

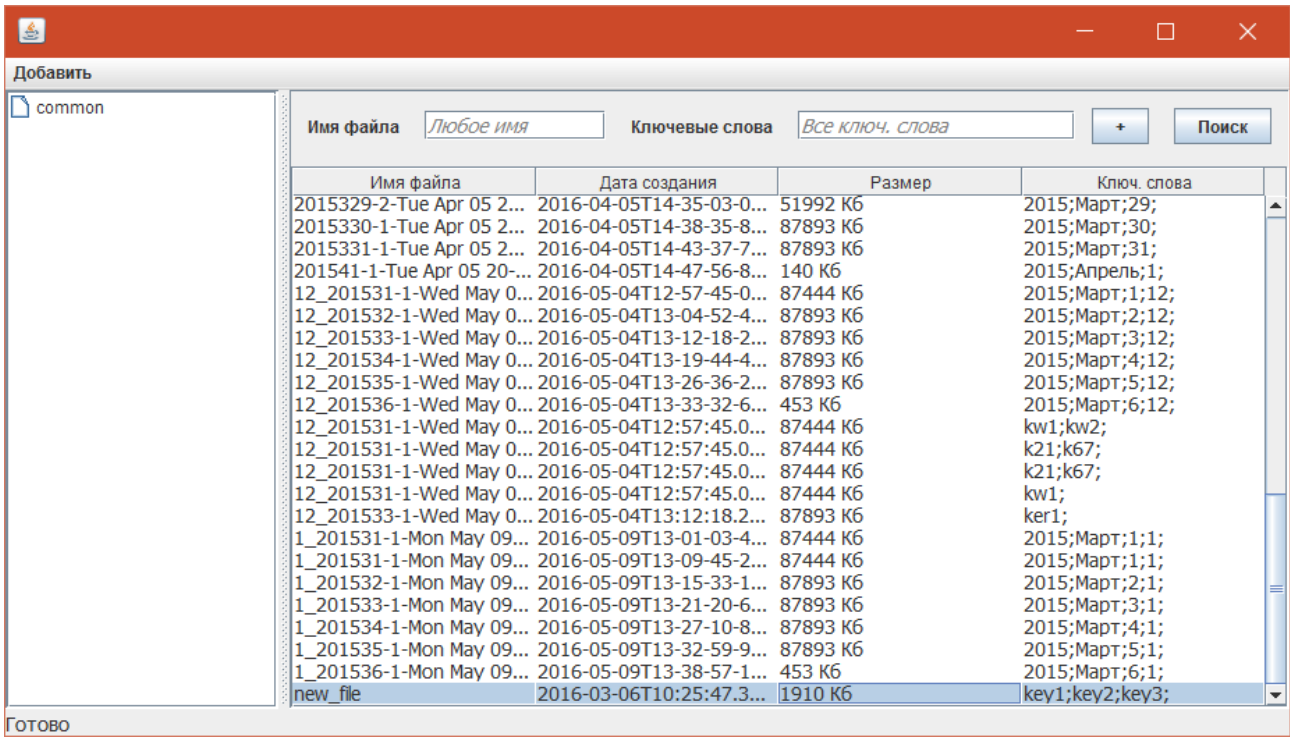

Рисунок 18 — Информация о новом файле в главном окне4.3 Применение системы

Одним из тестовых вариантов применения системы стала задача хранения данных замеров ультразвуковых анемометров. Один замер происходит раз в 12,5 мс и содержит информацию о температуре, влажности, давлении, силе ветра и т. д. Полученные данные группируются в 10-ти минутные интервалы, и в виде одного файла отправляются на сервер. Эти файлы было решено занести в разрабатываемое распределенное файловое хранилище.

Однако, прежде чем занести данные в хранилище, было решено подвергнуть их реструктуризации и объединить в файлы, содержащие суточный интервал. Такой способ компоновки более логичен и значительно упрощает как процесс занесения, так и последующий поиск по ключевым словам. Для упаковки и занесения файлов в систему было написано отдельное приложение с консольным интерфейсом (рисунок 19)

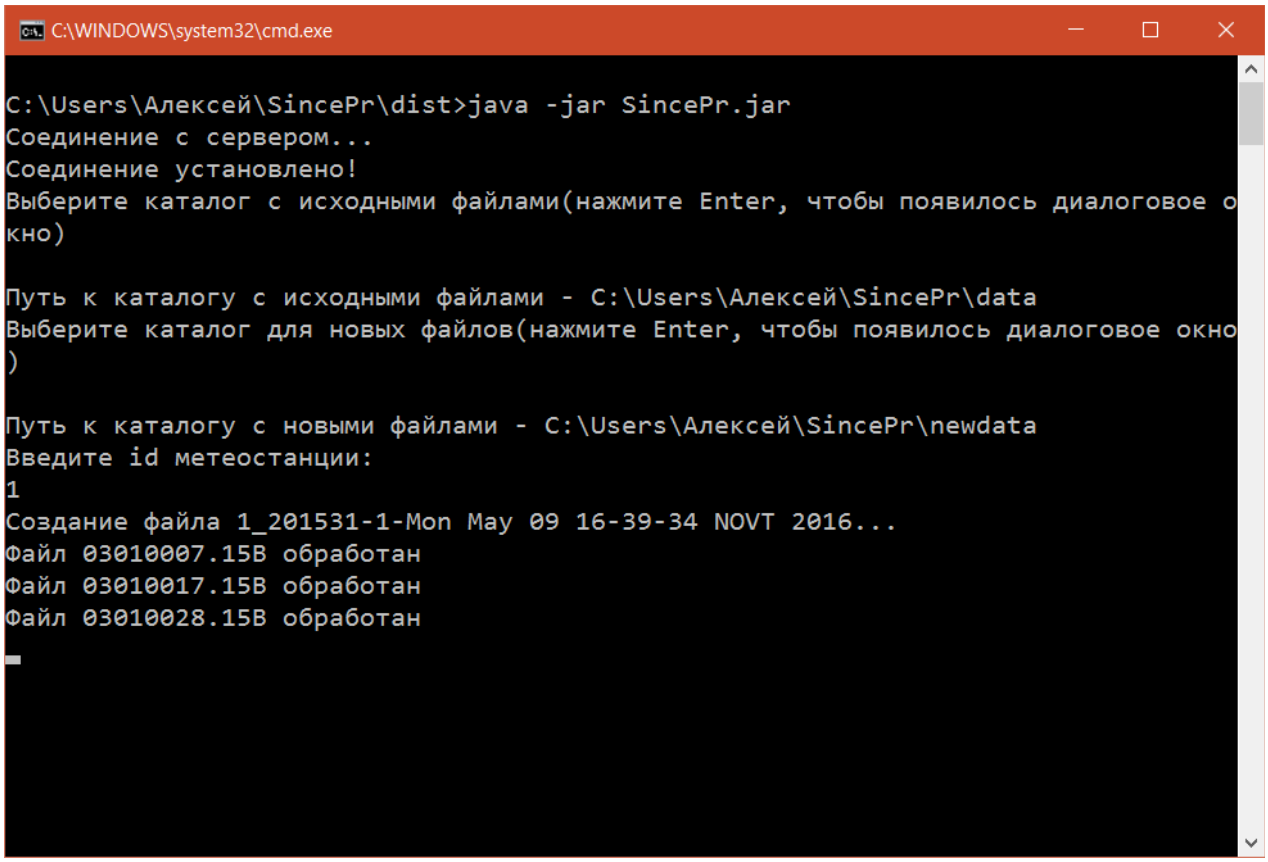

Рисунок 19 — Интерфейс приложения-упаковщика

Помимо создания и отправки суточных файлов, приложение так же формирует метаописание для получившихся файлов.

Для реализации данного приложения использовался язык Java и IDE NetBeans.

4.4 Описание классов основных компонентов

Рассмотрим классы, входящие в состав приложений «командный центр» и «агент». Ниже, в таблице 2 представлено краткое описание классов приложения «командный центр».

Таблица 2 - Описание классов приложения «Командный центр»

Имя класса	Наследует/реализует	Описание
CenterForm	JFrame, ChangeListener	Основной класс приложения, создает все дополнительные потоки, реализует пользовательский интерфейс.
Agent		Класс, представляющий собой

		отдельного агента. Поля данного класса соответствуют характеристикам агента.
CallCenterThread	Runnable	Класс, реализующий приемник сообщений. Работает в отдельном потоке, создает дополнительные рабочие потоки для выполнения заданий.
MonitoringThread	Runnable	Экземпляр данного класса осуществляет мониторинг агентов. Работает в отдельном потоке.
WorkingThread	Runnable	Абстрактный класс рабочего потока.
FindAgentThread	WorkingThread, Runnable	Класс, выполняющий работу по поиску агентов(содержащих определенный файл или готовых принять новый). Работает в отдельном потоке, создается по мере необходимости.
ContentThread	WorkingThread, Runnable	Поток для работы с содержимым агентов. Осуществляет поиск и выдачу списков файлов из метаописания.

Ниже, на рисунке 20 представлена схема взаимодействия классов.

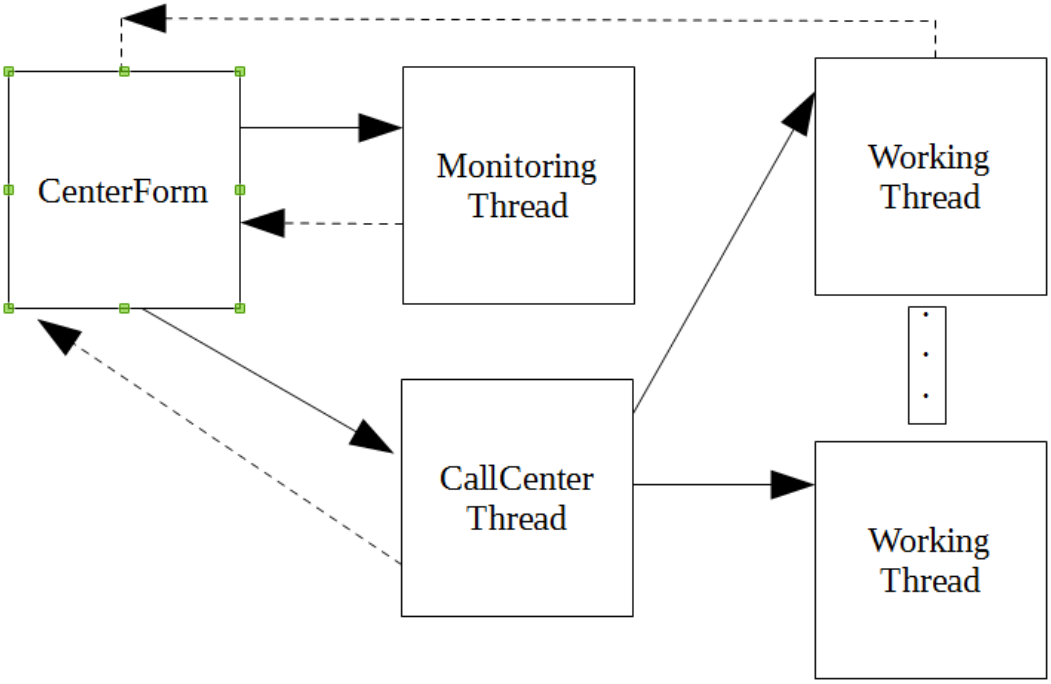

Рисунок 20 - Схема взаимодействия классов в приложении «Командный центр»

Некоторые пояснения к диаграмме. Сплошные стрелки означают, что класс создает экземпляр связанного класса в отдельном потоке. Пунктирная стрелка говорит о том, что класс вызывает в своем потоке методы другого класса. Линия, состоящая из точек означает, что класс создает экземпляр нового класса в своем потоке.

Как видно из диаграммы, основной класс, CenterForm, создает поток-приемник сообщений и поток мониторинга (при этом, эти потоки периодически вызываю методы CenterForm, например, обновление интерфейса пользователя и доступ к списку агентов). Поток CallCenterThread по мере необходимости способен создавать любое количество объектов, наследующих WorkingThread для выполнения запросов клиентов (поиск и выдача файлов, загрузка новых данных в систему и т. д.).

Теперь рассмотрим приложение -агент. В таблице 3 представлен список

его классов.

Таблица 3 - Описание классов приложения «Агент»

Имя класса	Наследует/реализует	Описание
AgentForm	JFrame	Основной класс приложения. Реализует интерфейс пользователя, а также создает поток, прослушивающий сообщения.
Directory		Класс, экземпляр которого представляет собой отдельную «рабочую» область, в которой агент может размещать данные. Имеет поля, содержащие информацию о пути(каталог) и выделенном пользователем размере.
AddNewFolder Dialog	JDialog	Форма добавления нового рабочего пространства агенту.
SetupConnection	JDialog	Форма настройки соединения с командным центром.
ListenerThread	Runnable	Класс реализует прием и обработку входящих сообщений. Для выполнения дальнейшей работы создает экземпляры класса
TransferFileThread	Runnable	Класс, реализующий рабочий поток агента. Передает, либо загружает файлы.

На рисунке 21 приведена схема взаимодействия классов в приложении

агенте.

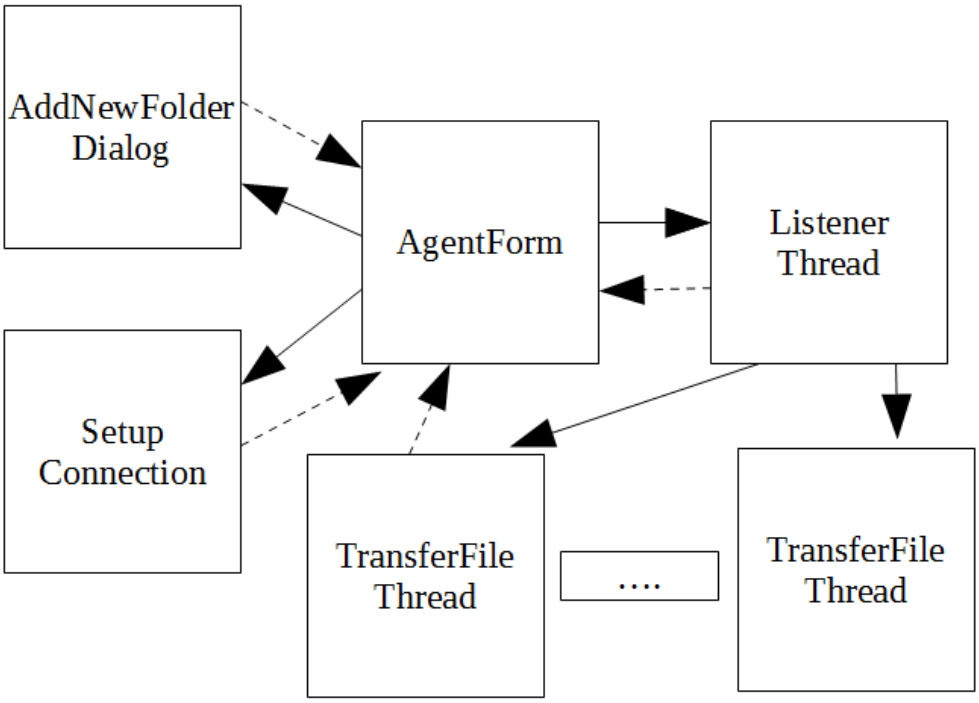

Рисунок 21 - Схема взаимодействия классов в приложении «Агент»

Основной формой (и основным классом) приложения-агента является AgentForm. Экземпляр данного класса содержит информацию о доступных рабочих областях агента и реализует пользовательский интерфейс. При установке соединения с командным центром создается экземпляр ListenerThread (в отдельном потоке), который выполняет роль приемника сообщения. Данный класс может вызывать методы AgentForm (например, для получения информации о рабочих областях и обновлении интерфейса). Экземпляры класса TransferFileThread создаются по мере необходимости для передачи/приема файлов. Дополнительные формы (AddNewFolderDialog и SetupConnection) могут создаваться для предоставления пользователю интерфейса настройки рабочих областей и подключения соответственно.

4.5 Описание классов приложения «Файловый менеджер»

Теперь стоит рассмотреть внутреннее устройство приложения «файловый менеджер». Его внутренняя структура состоит из двух частей — из классов, реализующих интерфейс пользователя, и классов, отвечающих за

взаимодействие с распределенной файловой системой. Последние вынесены в отдельную библиотеку и могут использоваться в других приложениях для организации взаимодействия с распределенным хранилищем. Ниже, в таблице 4 представлено описание классов файлового менеджера.

Таблица 4 - Описание классов приложения «Файловый менеджер»

Имя класса	Наследует/реализует	Описание
FileManagerForm	JFrame	Основная форма приложения. Реализует интерфейс пользователя.
AddFileDialog	JDialog	Диалог добавления нового файла в систему.
AddKeyWordDialog	JDialog	Диалог добавления ключевого слова в строку поиска. Служит для удобства работы, если пользователь не знает, какие ключевые слова есть в системе
NewCategDialog	JDialog	Форма добавления нового ключевого слова в систему.

Как видно из таблицы, все классы представляют собой формы пользовательского интерфейса, выполняющие те или иные функции. Теперь рассмотрим часть, отвечающую за взаимодействие с хранилищем (DataExchangeLib) (таблица 5)

Таблица 5 - Описание классов библиотеки DataExchangeLib

Имя класса	Наследует/реализует	Описание
DataExchange		Класс, реализующий основной функционал библиотеки. Содержит методы для работы с файлами в

		хранилище (получение, занесение, удаление, переименование и т.д.)
CommandParser		Класс, содержащий методы для извлечения параметров из команды. Также служит для формирования команд
FileInfo		Класс, представляющий собой метаописание файла.
CommandName		Класс — перечисление, содержащий константы — имена команд, необходимые для формирования команд
AttrName		Класс — перечисление, содержащий константы — имена параметров команд, необходимые для формирования команд
ExchangeException	Exception	Класс исключение, генерируемый в случае ошибок при взаимодействии с файловым хранилищем.

Рассмотрим подробнее данную библиотеку. Как видно из таблицы, основным классом, служащим для отправки запросов хранилищу, является класс DataExchange. В нем можно выделить следующий набор публичных методов:

- addFile – занесение нового файла в систему;
- getFile – получение файла из системы (согласно списку ключевых слов и/или имени файла);
- deleteFile – удалить файл;

- renameFile – переименовать файл;
- addKeyWord – добавить ключевое слово;
- getKeyWord – получить список ключевых слов.

С помощью этих методов приложение, в состав которого включена данная библиотека, может легко взаимодействовать с хранилищем. Для наглядности, продемонстрируем фрагмент кода, позволяющий получить файл из системы:

```
File fileToSave = new File(filePath + "\\" + fileToLoad.getFileName());
File comCenList = new File("cclist");
List<String> keyWords = new ArrayList<String>();
keyWords.add(“key1”);
keyWords.add(“key2”);
FileInfo fileToLoad = new FileInfo(“somefile.txt”, “common”, keyWords);
DataExchange exch = new DataExchange();
try{
     exch.getFile(fileToLoad, comCenList, fileToSave, null);
}
catch(ExchangeException ex){

}
```

Как видно из кода, в первых двух строчках создается два новых экземпляра класса File – fileToSave (будет использован для сохранения нового файла) и comCenList, файл, содержащий список доступных командных центров. Далее создается список ключевых слов, включающий два слова — key1 и key2. Для загрузки файла необходимо передать его метаописание, которое в набое классов библиотеки DataExchangeLib реализовано в виде класса FileInfo. В конструктор класса передается имя файла, имя категории (в данном случае common) и список ключевых слов. Далее создается экземпляр класса DataExchange, и вызывается метод getFile, параметрами которого являются:

1. Метаописание загружаемого файла
2. Файл, содержащий список командных центров
3. Экземпляр File, который будет использован для сохранения загруженного файла на локальной машине

4. Экземпляр функционального интерфейса CallBack, который может быть использован для действий, связанных с отображением процесса загрузки. В данном примере данный функционал ненужен, поэтому передано значение null.

В результате выполнения данного кода из файлового хранилища будет загружен файл с именем somefile.txt и ключевыми словами key1 и key2.

4.6 Мониторинг агентов

Важной функцией командного центра является мониторинг агентов, находящихся в сети. За эту работу отвечает класс MonitorThread, реализующий интерфейс Runnable. Ниже приведен фрагмент кода метода run, осуществляющий опрос агентов с заданной периодичностью.

```
while(isActive)
{
    List<Agent> onlineClients = sender.getOnlineClients();
    sender.log(String.valueOf(onlineClients.size()));
    for(Agent client : onlineClients){
        try{
            sock = new Socket();
            sock.connect(new InetSocketAddress(client.ipAddress,
client.port), 2000);
            input = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
            output = new PrintWriter(sock.getOutputStream(),true);
            output.println("_get_info_");
            String info = input.readLine();
            sender.log(info);
            client.updateAgent(ConnectionService.parseCommand(info));
            ...
        }
        catch(Exception ex){
            ….
    }
    sender.updateClientTable();
    try{
        Thread.sleep(timer * 1000);
    }
```

```
                catch(Exception ex){
                    isActive = false;
                }
            }
```

Как видно из кода метода, поток опрашивает всех агентов из списка onlineClients с периодичностью timer(в секундах). Список представляет собой коллекцию List объектов класса Agent. Каждый экземпляр данного класса содержит информацию о конкретном агенте, в том числе, данные о его сокетном соединении. В случае невозможности установить соединение, агент удаляется из списка.

4.7 Получение сообщений

Рассмотрим процесс получения сообщения с точки зрения программной реализации.

Как видно из таблицы 2, за прием сообщений отвечает класс CallCenterThread, реализующий интерфейс Runnable. В системе постоянно существует экземпляр данного класса, а в методе run запущен цикл, создающий заново сокетное соединение для приема сообщений.

```
                while(online){
                    socket = serv.accept();
                    getCall();
                }
```

Метод getCall реализует функционал «диалога» с системой. В данном методе в цикле происходит процесс приема сообщений и их интерпретация в методе interpCom. В качестве возвращаемого значения метода выступает ответ системы на сообщение от клиента. Если клиент разорвал соединение или прислал команду, означающую завершение диалога, то происходит выход из цикла и завершение коммуникации с данным клиентом.

Рассмотрим в качестве примера фрагмент метода interpCom, отвечающий за интерпретацию команды занесения файла в систему:

```
        if(message.contains("_add_file_") || message.contains("_get_file_") ||
message.contains("_delete_file_") || message.contains("_rename_file_")){
                FindAgentThread findAgentThread = new FindAgentThread(sender,
message);
```

```
            return "";
        }
```

В результате выполнения данного кода создается экземпляр класса FindAgentThread. Конструктор принимает два параметра — ссылку на главную форму приложения и строку — команду. Далее приведен код метода run:

```
    @Override
    public void run(){
        commandParams = ConnectionService.parseCommand(command);
        String commandWord = ConnectionService.getCommand(command);
        ...
        String fileCreationDate = "";
        List<String> keyWords = new ArrayList<String>();
        for(String param : commandParams){
          ...
        }
        if(commandWord .equals("_add_file_"))
            findFreeAgent(fileSize);
       ...
    }
```

Новый поток создается в конструкторе класса FindAgentThread, таким образом при создании нового экземпляра автоматически вызывается метод run. В нем происходит разбор строки — команды по параметрам, а также передача управления методу, отвечающему за обработку нужной команды. Поскольку рассматривается занесение файла в систему, то нас интересует метод findFreeAgent:

```
    private void findFreeAgent(long fileSize){
            String ans;
            agentList = centerForm.getOnlineClients();
            agentList.sort(new Agent.LoadSort());
            for(Agent agent : agentList){
                if(agent.getFreeSpace() > fileSize){
                    try(Socket sock = new Socket(agent.ipAddress, agent.port);
                            BufferedReader out = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
                            PrintWriter in = new PrintWriter(sock.getOutputStream(),
true);){
                        in.println(command);
                        ans = out.readLine();
```

```
                        if(ans.contains("_accept_")){
                            break;
                        }
                    }
                    catch(Exception ex){

                    }
                }
            }
        }
```

После получения списка агентов agentList, происходит сортировка агентов в данном списке по увеличению степени загруженности (процент свободного дискового пространства). Далее, агенты опрашиваются, и, если один из них готов принять файл, то цикл опроса прекращается. На этом работа данного потока завершена, и с точки зрения командного центра, клиент считается обслуженным — далее с ним начинает работу выбранный агент. Проследим дальнейший путь поток управления. Как было сказано выше, агент также имеет свой приемник сообщений(ListenerThread), в котором также присутствуют методы getCall и interpCom. Принцип их работы схож с аналогичными методами командного центра, поэтом приведем лишь фрагмент метода interpCom, отвечающий за команду принятия нового файла:

```
        if(ans.contains("_add_file_") || ans.contains("_get_file_") ||
ans.contains("_delete_file_") || ans.contains("_rename_file_")){
                    Directory dir = null;
                    int port = 0;
                    String ip = "";
                    String fileName = "";
                    params = ConnectionService.parseCommand(ans);
                    for(String param : params){
                        if(param.contains("fs:") && ans.contains("_add_file_"))
                            dir =
sender.getApropDirectory(Long.parseLong(param.substring(3)));
                        ...
                    }
                    if(dir != null){
                        TransferFileThread transferFileThread = new
TransferFileThread(ans, sender, dir, ip, port, fileName);
```

```
                return "_accept_";
            }
            return "_dec_";
        }
```

В данном коде, помимо непосредственно интерпретации, проходит проверка, в ходе которой выясняется, может ли агент принять данный файл:

```
dir = sender.getApropDirectory(Long.parseLong(param.substring(3)));
```

В данном примере, dir является экземпляром класса Directory, описывающем рабочий каталог агента, метод getApropDirectory пытается подобрать рабочий каталог, способный принять новый файл (стоит отметить, что рабочих каталогов может быть несколько, например, на разных логических дисках). Если подходящий каталог не был найден, то метод вернет null. Таким образом, если каталог был подобран, то создается экземпляр TransferFileThread, являющийся рабочим потоком приложения — агента. Код метода run приведены ниже:

```
@Override
public void run(){
    agent.log("add file thread starts " + Thread.currentThread().getId());
    if(command.contains("_add_file_"))
        addFile();
    if(command.contains("_get_file_"))
        getFile();
    if(command.contains("_delete_file_"))
        deleteFile();
    if(command.contains("_rename_file_"))
        renameFile();
}
```

В целом, логика работы повторяет подход, примененный в классе FindAgentThread. Далее рассмотрим метод addFile.

```
private void addFile(){
        ...
        try(Socket sock = new Socket(ip, port);
                DataInputStream in = new DataInputStream(sock.getInputStream());
                PrintWriter out = new PrintWriter(sock.getOutputStream(),
true);){
            out.println("_ready_");
```

```
                    ans = in.readUTF();
                    if(ans.contains("_trans_info_")){
                        List<String> fileParams = ConnectionService.parseCommand(ans);
                        for(String fileParam : fileParams){
                            if(fileParam.contains("bf:")){
                                buffSize = Integer.parseInt(fileParam.substring(3));
                                continue;
                            }
                        }
                    }
                    ans = in.readUTF();
                    if(ans.contains("_meta_")){
                        File metadiscrFile = new File("metadiscr");
                        List<String> fileParams = ConnectionService.parseCommand(ans);
                        for(String fileParam : fileParams){
                            ...
                        metaInfo = ans + "sf:" + fileName + "@" + "id:" + agent.getID()
+ "@";
                        writeToMetaFile("", metaInfo, true);
                    }
                    FileOutputStream fout = new FileOutputStream(destDir.getPath() +
"\\" + fileName);
                    byte[] buffer = new byte[buffSize];
                    int count;
                    while((count = in.read(buffer)) >= 0)
                        fout.write(buffer,0,count);
                    agent.log("end");
                    out.println("_end_trans_");
                    fout.close();
                    destDir.reCalcFreeSpace();
                    List<String> ccList = agent.getCCList();
                    for(int i = 0; i < ccList.size(); i += 2){
                        try(Socket ccSock = new
Socket(ccList.get(i),Integer.parseInt(ccList.get(i + 1)));
                                PrintWriter wr = new PrintWriter(new OutputStreamWriter(
            ccSock.getOutputStream(), StandardCharsets.UTF_8), true)){
                                wr.println(metaInfo.replace("_meta_", "_new_file_"));
                            }
                    }
                }
                catch(Exception ex){
```

```
            agent.log(ex.getMessage());
        }
    }
```

В данном методе агент последовательно выполняет ряд действий: подключается к клиенту и сообщает о своей готовности принять файл. Затем он получает служебную информацию от клиента ("_trans_info_"), в частности, размер буфера, который будет использоваться при передаче файла, и метаописание файла. Метаописание файла агент сохраняет в своем локальном файловом каталоге. Далее, происходит передача файла от клиента к агенту. После того, как файл был успешно принят, агент должен оповестить все командные центры о том, что в системе появился новый файл. В последнем цикле агент рассылает сообщения, содержащие метаописание нового файла, всем командным центрам.

Командный центр принимает сообщение о новом файле так же, как и прочие сообщения - через поток — приемник (методы getCall и interpCom). Ниже приведен фрагмент метода interpCom, отвечающий за интерпретацию команды занесения файла в систему:

```
        if(message.contains("_new_file_") || message.contains("_file_removed_") ||
message.contains("_file_renamed_")){
            ContentThread contThread = new ContentThread(message, sender);
            return "";
        }
```

Как видно из кода метода, после интерпретации вызывается конструктор класса ContentThread. Этот класс также, как и FindAgentThread наследует WorkingThread и является рабочим потоком для выполнения заданий, не связанных с поиском агентов. Ниже приведен код метода run:

```
    @Override
    public void run(){
        if(command.contains("_get_file_list_"))
            sendFileList();
        if(command.contains("_new_file_"))
            addNewFileToCat();
        if(command.contains("_file_removed_"))
            removeFile();
```

```
            if(command.contains("_file_renamed_"))
                renameFile();
        }
```

Далее управление передается методу addNewFileToCat:

```
    private void addNewFileToCat() {
            List<String> params = ConnectionService.parseCommand(command);
            List<String> keyWords = new ArrayList<String>();
            String meta = command.replace("_new_file_", "_meta_");
            ...
            String userCatFoldPath = center.catFoldStr + "/" + userName;
            PrintWriter fileWr;
            try{
                ...
                changeFileLock(userAllFiles.getPath(), false);
                for(String keyWord : keyWords){
                    File categoryFile = new File(userCatFoldPath + "/" + keyWord);
                    categoryFile.createNewFile();
                    getFileLock(categoryFile.getPath());
                    fileWr = new PrintWriter(new FileOutputStream(categoryFile,
true), true);
                    fileWr.println(meta);
                    fileWr.close();
                    changeFileLock(categoryFile.getPath(), false);
                }
            }
            catch(Exception ex){
                fileWr.close();
            }
        }
```

Данный метод добавляет запись о новом файле в файлы — каталоги, отвечающие за каждое ключевое слово, присутствующее в метаописании. При записи вызывается метод getFileLock, который в цикле пытается получить доступ к файлу:

```
    private void getFileLock(String filePath){
            while(!changeFileLock(filePath, true)){
                try{
                    Thread.sleep(1000);
                }
                catch(Exception ex){
```

```
            }
        }
    }
```

Код метода changeFileLock:

```
    protected static synchronized boolean changeFileLock(String file, boolean
getLock){
        if(getLock){
            if(fileLocks.contains(file))
                return false;
            else{
                fileLocks.add(file);
                return true;
            }
        }
        else{
            fileLocks.remove(file);
            return true;
        }
    }
```

Данный механизм сделан для предотвращения редактирования файла несколькими потоками. Ключевое слово synchronized в сигнатуре метода changeFileLock запрещает исполнение этого метода из параллельного потока(при этом вызывающий поток встает в очередь), если его уже выполняет другой поток. Метод getFileLock пытается получить доступ к файлу, и, если он занят другим потоком, повторяет попытку после 1000 мс.

4.8 Защита данных

Вместе с задачей хранения данных естественным образом встает задача защиты конфиденциальности хранимых данных. Существует множество механизмов, ограничивающих доступ к данным третьими лицами, но применительно к данной работе, наиболее логичными являются следующие меры защиты:

1. разграничение доступа для каждого пользователя;
2. криптографическое преобразование хранящихся данных.

Для реализации разграничения доступа необходимо дополнительно

реализовать сервер с многоуровневой авторизацией клиентов (может быть совмещен с командным центром), а также организовать безопасное хранение регистрационных данных пользователей (использовать любую СУБД, предоставляющую механизмы защиты).

Криптографическое преобразование может быть реализовано стандартными средствами языка Java, в которые входит возможность шифрования и расшифрования файлов. Реализацию механизма распределения ключей шифрования также можно возложить на командный центр (дополнительная функция при регистрации пользователя). При этом каждому пользователю будет сгенерирован индивидуальный ключ, в результате чего прочесть файлы сможет только тот пользователь, который занес их в систему.

Однако, стоит отметить, что на данном этапе разработки, реализация механизмов защиты не предусмотрена, так как в текущий момент времени система предназначена для использования в организациях, где каждый сотрудник имеет доступ к хранимым данным и добавление механизма защиты в данном случае только усложнит взаимодействие с хранилищем.

5 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

5.1 Организация и планирование работ

Рациональное планирование занятости всех участников разработки проекта является важной задачей, без решения которой в ходе работы над проектом могут возникнуть ряд проблем, в том числе неоправданное увеличение длительности разработки. Поэтому необходимо составить перечень работ с указанием исполнителей и рациональной продолжительности. Ниже, в таблице 6 перечислены основные этапы разработки и процент загруженности каждого исполнителя на данном этапе.

Таблица 6 - Перечень работ и продолжительность их выполнения

Этапы работы	**Исполнители**	**Загрузка исполнителей**
Постановка целей и задач, получение исходных данных	НР	НР – 100%
Составление и утверждение ТЗ	НР, И	НР – 100% И – 10%
Подбор и изучение материалов по тематике	НР, И	НР – 30% И – 100%
Разработка календарного плана	НР, И	НР – 100% И – 10%
Разработка архитектуры проекта	НР, И	НР – 20% И – 100%
Программная реализация	И	И – 100%
Тестирование и доработка	НР, И	НР – 30% И –100%
Оформление пояснительной записки	И	И – 100%
Оформление графического материала	И	И – 100%
Подведение итогов	НР, И	НР – 60% И – 100%

5.1.1 Продолжительность этапов работ

Существует несколько способов для расчета продолжительности каждого этапа. В данной работе выбран экспертный способ по формуле

$$t_{ожс} = \frac{t_{\min} + 4 \cdot t_{prob} + t_{max}}{6} \quad (1)$$

где t_{min} – минимальная продолжительность работы, дн.;

t_{max} – максимальная продолжительность работы, дн.;

t_{prob} – наиболее вероятная продолжительность работы, дн.

Для построения линейного графика необходимо рассчитать длительность этапов в рабочих днях, а затем перевести ее в календарные дни. Расчет продолжительности выполнения каждого этапа в рабочих днях ($\boldsymbol{T_{РД}}$) ведется по формуле:

$$T_{РД} = \frac{t_{ож}}{K_{ВН}} \cdot K_{Д} \qquad (2)$$

где $t_{ож}$ – продолжительность работы, дн.;

$K_{вн}$ – коэффициент выполнения работ, учитывающий влияние внешних факторов на соблюдение предварительно определенных длительностей, в данной работе $K_{вн} = 1$;

$K_{Д}$ – коэффициент, учитывающий дополнительное время на компенсацию непредвиденных задержек и согласование работ ($K_{Д} = 1{,}2$).

Расчет продолжительности этапа в календарных днях ведется по формуле:

$$T_{КД} = T_{РД} \cdot T_{К} \qquad (3)$$

где $T_{КД}$ – продолжительность выполнения этапа в календарных днях;

$T_{К}$ – коэффициент календарности, позволяющий перейти от длительности работ в рабочих днях к их аналогам в календарных днях(равен 1,205).

Ниже, в таблице 7 приведен результат расчетов продолжительности этапов разработки проекта. В таблице 8 приведен линейный график для данных этапов.

Таблица 7 - Трудозатраты на выполнение проекта

Этап	Исполнители	Продолжительность работ, дни				Трудоемкость работ по исполнителям чел.- дн.			
						Трд		*Ткд*	
		t_{min}	t_{prob}	t_{max}	$t_{ожид}$	НР	И	НР	И
Постановка целей и задач, получение исходных данных	НР	2	3	4	3	3,6	0	4,34	0
Составление и утверждение ТЗ	НР, И	2	2	3	2,2	2,6	0,26	3,13	0,31
Подбор и изучение материалов по тематике	НР, И	5	6	8	6,2	2,22	7,4	2,68	8,92
Разработка календарного плана	НР, И	1	2	3	2	2,4	0,24	2,89	0,29
Разработка архитектуры проекта	НР, И	12	15	20	15,3	3,68	16,56	4,43	19,95
Программная реализация	И	15	17	20	17,2	2,06	20,6	2,48	24,82
Тестирование и доработка	НР, И	10	12	15	12,2	7,3	13,14	8,79	15,83
Оформление пояснительной записки	И	6	7	9	7,2	0	8,6	0	10,36
Оформление графического материала	И	1	1	2	1,2	0	1,4	0	1,68
Подведение итогов	НР, И	5	6	8	6,2	4,44	7,4	5,35	8,92
Итог					72,5	28,3	75,6	34,1	91,1

Таблица 8 — Линейный график работ

Этап	НР	И	Март			Апрель			Май		
			10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**
	14,34	–									
2	3,13	0,31									
3	2,68	8,92									
4	2,89	0,29									
5	4,43	19,95									
6	2,48	24,82									
7	8,79	15,83									
	8–	10,36									
9	–	1,68									
10	5,35	8,92									

НР – ; И –

5.1.2 Расчет накопления готовности проекта

Цель данного пункта – оценка текущих состояний (результатов) работы над проектом. Величина накопления готовности работы показывает, на сколько процентов по окончании текущего (i-го) этапа выполнен общий объем работ по проекту в целом.

Введем обозначения:

Степень готовности определяется формулой

$$СГ_i = \frac{ТР_i^Н}{ТР_{общ.}} = \frac{\sum_{k=1}^{i} ТР_k}{ТР_{общ.}} = \frac{\sum_{k=1}^{i}\sum_{j=1}^{m} ТР_{km}}{\sum_{k=1}^{I}\sum_{j=1}^{m} ТР_{km}}. \quad (4)$$

где

ТРобщ. – общая трудоемкость проекта;

ТРi (ТРk) − трудоемкость i-го (k-го) этапа проекта, $i = \overline{1,I}$;

ТРiН − накопленная трудоемкость i-го этапа проекта по его завершении;

ТРij (ТРkj) − трудоемкость работ, выполняемых j-м участником на i-м этапе.

Ниже, в таблице 9 приведены величины $Сг_i$ и $Тр_i$ для конкретных этапов разработки.

Таблица 9 - Нарастание технической готовности работы и удельный вес каждого этапа

Этап	$ТР_i$, %	$СГ_i$, %
Постановка задачи	3,46	3,46
Разработка и утверждение технического задания (ТЗ)	2,75	6,21
Подбор и изучение материалов по тематике	9,25	15,47
Разработка календарного плана	2,54	18,01
Разработка архитектуры	19,48	37,49
Программная реализация	21,8	59,3
Тестирование и доработка	19,67	78,97
Оформление расчетно-пояснительной записки	8,27	87,26
Оформление графического материала	1,34	88,6
Подведение итогов	11,4	100.00

5.2 Расчет сметы затрат на выполнение проекта

5.2.1 Расчет затрат на материалы

В данную статью расходов включены различные материалы, покупные изделия и прочее, расходуемое в процессе разработки. Кроме того, сюда входят транспортно — заготовительные расходы, связанные с транспортировкой, хранением, доставкой до потребителя и т. д.

Таблица 10 - Расчет затрат на материалы

Наименование материалов	Цена за ед., руб.	Кол-во	Сумма, руб.
Бумага для принтера формата А4	205	1 уп.	205
Картридж для принтера	1500	1 шт.	1500
Итого:			**1705**

Допустим, что ТЗР составляют 5 % от отпускной цены материалов, тогда расходы на материалы с учетом ТЗР равны $С_{мат}$ = 1705 * 1,05 = 1790,25 руб.\

5.2.2 Расчет заработной платы

Данная статья расходов включает заработную плату научного руководителя и исполнителя проекта, а также премии, входящие в фонд заработной платы. Для расчета заработной платы необходимо получить среднедневную тарифную заработную плату:

$$ЗП_{дн-т} = МО/24,83 \qquad (5)$$

где МО — месячный оклад. Также в месяце в среднем 24,83 рабочих дня.

Также для расчета заработной платы необходимо учесть премии, дополнительные зарплаты и районную надбавку. Для этого вводится интегральный коэффициент $К_и$ = 1,1* 1,188*1,3 = 1,699. Ниже, в таблице 11 приведены расчеты заработной платы всех участников проекта.

Таблица 11 — Затраты на заработную плату

Исполнитель	Оклад, руб./мес.	Среднедневная ставка, руб./раб.день	Затраты времени, раб.дни	Коэффициент	Фонд з/платы, руб.
НР	23 264,86	936,97	29	1,699	46165,24
И	7864,11	316,71	76	1,699	40895,90
Итого:					87061,15

5.2.3 Расчет затрат на социальный налог

Затраты на единый социальный налог (ЕСН), включающий в себя отчисления в пенсионный фонд, на социальное и медицинское страхование, составляют 30 % от полной заработной платы по проекту, т.е. $С_{соц.} = С_{зп}*0,3$

Таким образом, $С_{соц} = 26118,34$

5.2.4 Расчет затрат на электроэнергию

Затраты на электроэнергию рассчитываются по следующей формуле:

$$С_{эл.об.} = Р_{об} \cdot t_{об} \cdot Ц_Э \quad (6)$$

где $P_{ОБ}$ – мощность, потребляемая оборудованием, кВт;

$Ц_Э$ – тариф на 1 кВт·час;

$t_{об}$ – время работы оборудования, час.

Для ТПУ $Ц_Э$ = 5,257 руб./квт·час (с НДС).

Время работы оборудования вычисляется по формуле:

$$t_{об} = T_{РД}* К_t, \qquad (7)$$

где $К_t \leq 1$– коэффициент использования оборудования по времени, равный отношению времени его работы в процессе выполнения проекта к $T_{РД}$.

Мощность, потребляемая оборудованием, определяется по формуле:

$$P_{ОБ} = P_{ном.} * К_С \qquad (8)$$

где $P_{ном.}$ – номинальная мощность оборудования, кВт;

$K_С \leq 1$ – коэффициент загрузки, зависящий от средней степени использования номинальной мощности. Для технологического оборудования малой мощности $K_С = 1$.

Расчет затрат на электроэнергию приведен в таблице 12

Таблица 12 - Затраты на электроэнергию технологическую

Наименование оборудования	Время работы оборудования $t_{ОБ}$, час	Потребляемая мощность $P_{ОБ}$, кВт	Затраты $Э_{ОБ}$, руб.
Ноутбук	604,8	0,101	321,12
Струйный принтер	2	0,1	1,05
Итого:			322,174

5.2.5 Расчет амортизационных расходов

Используется формула

$$С_{АМ} = \frac{Н_А * Ц_{ОБ} * t_{рф} * n}{F_Д}, \quad (9)$$

где $Н_А$ – годовая норма амортизации единицы оборудования;

$Ц_{ОБ}$ – балансовая стоимость единицы оборудования с учетом ТЗР.

$F_Д$ – действительный годовой фонд времени работы соответствующего оборудования, берется из специальных справочников или фактического режима его использования в текущем календарном году.

Расчет амортизации приведен в таблице 13.

Таблица 13 — Расчет амортизационных расходов

Наименование	Стоимость единицы, руб.	Норма амортизации	Годовой фонд времени работы, ч.	Расходы на амортизацию, руб.
Ноутбук	33000	0,4	2384	3348,72
Струйный принтер	9600	0,5	500	19,2
Итог				3367,92

5.2.6 Прочие расходы

В статье «Прочие расходы» отражены расходы на выполнение проекта, которые не учтены в предыдущих статьях, их следует принять равными 10% от суммы всех предыдущих расходов, т.е.

$С_{проч.} = (С_{мат} + С_{зп} + С_{соц} + С_{эл.об.} + С_{ам} + С_{нп}) \cdot 0,1 = (1790,25 + 87061,15 + 26118,34 + 322,174 + 3367,92) \cdot 0,1 = 11865,98$ руб.

5.2.7 Расчет общей себестоимости разработки

В таблице 14 представлена сумма всех статей затрат, рассмотренных в предыдущих пунктах

Таблица 14 - Смета затрат на разработку проекта

Статья затрат	Условное обозначение	Сумма, руб.
Материалы и покупные изделия	$С_{мат}$	1790,25
Основная заработная плата	$С_{зп}$	87061,15
Отчисления в социальные фонды	$С_{соц}$	26118,34
Расходы на электроэнергию	$С_{эл.}$	322,174
Амортизационные отчисления	$С_{ам}$	3367,92
Прочие расходы	$С_{проч}$	11865,98
Итого:		130525,82

Затраты на разработку составили $С$ = 130525,82 руб.

5.2.8 Расчет прибыли

Поскольку данные для проведения расчетов прибыли отсутствуют, то считаем, что прибыль составляет 10 % от полной себестоимости проекта, т. е. 130525,82 * 0,1 = 13052,58 руб.

5.2.9 Расчет НДС

НДС составляет 18% от суммы затрат на разработку и прибыли. В данном случае это (130525,82 + 13052,58) * 0,18 = 25844,11 руб.

5.2.10 Цена разработки НИР

Цена равна сумме полной себестоимости, прибыли и НДС:

$Ц_{НИР(КР)}$ = 130525,82 + 13052,58 + 25844,11 = 169422,51 руб.

5.3 Оценка экономической эффективности проекта

При оценке экономической эффективности данного проекта следует учесть тот факт, что данная разработка не предполагает получение прибыли и создавалась для свободного использования любыми заинтересованными в ней организациями. Основная цель данного проекта — минимизация ряда статей расходов, с которыми приходится сталкиваться при развертывании систем хранения больших данных, как например:

1. стоимость покупки серверного оборудования или его аренды;
2. затраты на развертывание более сложных систем (в том числе затраты, связанные с системным администрированием, обучением персонала, покупки лицензии ПО и т. д.);

3. затраты на привлечение услуг сторонних датацентров.

Таким образом, данная разработка может оказать положительный экономический эффект на предприятии, где она будет развернута.

1.4 Оценка научно — технического уровня НИР

Для определения влияния данной разработки на научно-технический прогресс необходимо оценить ее по трем критериям (уровень новизны, теоретический уровень, возможности реализации). Каждый пункт оценивается по 10-ти бальной шкале (таблица 15).

Таблица 15 - Оценки научно-технического уровня НИР

Значимость	Фактор НТУ	Уровень фактора	Выбранный балл	Обоснование выбранного балла
0,4	Уровень новизны	Не обладает новизной	0	Данная разработка опирается на ранее известные методы и принципы
0,1	Теоретический уровень	Разработка способа	2	Разработка архитектурного решения, опирающегося на существующие аналоги
0,5	Возможность реализации	В течение первых лет	10	За счет бесплатности и низких требовании может быть реализована достаточно быстро

После этого возможно подсчитать интегральный показатель научно-технического уровня НИР по следующей формуле:

$$K_{\text{НТУ}} = \sum_{i=1}^{3} R_i \cdot n_i, \quad (10)$$

где $K_{\text{НТУ}}$ – интегральный индекс научно-технического уровня;

R_i – весовой коэффициент i-го признака научно-технического эффекта;

n_i – количественная оценка i-го признака научно-технического эффекта, в баллах.

Таким образом, для данного проекта получаем $K_{нту}$ = 0,4*0 + 0,1*2 + 0,5*10 = 0,2 + 5 = 5,2

Исходя из данных таблицы 16, данная разработка обладает средним уровнем научно — технического эффекта.

Таблица 16 - Баллы для оценки уровня новизны

Уровень новизны	**Характеристика уровня новизны – n_1**	**Баллы**
Принципиально новая	Новое направление в науке и технике, новые факты и закономерности, новая теория, вещество, способ	8 – 10
Новая	По-новому объясняются те же факты, закономерности, новые понятия дополняют ранее полученные результаты	5 – 7
Относительно новая	Систематизируются, обобщаются имеющиеся сведения, новые связи между известными факторами	2 – 4
Не обладает новизной	Результат, который ранее был известен	0

6 Социальная ответственность

6.1 Введение

Дипломная работа посвящена разработке распределенного файлового хранилища (распределенной файловой системы). Данный программный комплекс позволяет объединить любое количество персональных компьютеров в распределенную грид — систему под управлением единого командного центра, таким образом аккумулируя их дисковое пространство для упорядоченного хранения данных больших объемов. Система в основном ориентирована на различные научно — исследовательские институты и небольшие коммерческие организации.

6.2 Производственная безопасность

Для обеспечения производственной безопасности необходимо проанализировать воздействия на человека вредных и опасных производственных факторов, которые могут возникать при разработке или эксплуатации проекта.

К опасным факторам относят негативное воздействие на работающего человека, которое может привести к травме или ухудшению здоровья. К вредным производственным факторам относят негативное воздействие, на человека, которое приводит к ухудшению здоровья или заболеванию.

На основании ГОСТ 12.0.003-74 представлена классификация опасных и вредных производственных факторов, имевшие место при выполнении работ на ПЭВМ.

Таблица 17 – Вредные и опасные производственные факторы при выполнении работ за ПЭВМ [11]

Источник фактора, наименование видов работ	Факторы (по ГОСТ 12.0.003-74)		Нормативные документы
	Вредные	Опасные	
1) Работа за ПЭВМ	1) Повышенная или пониженная температура воздуха рабочей зоны. 2) Повышенный уровень	1) Опасность поражения электрическим током.	1) СанПиН 2.2.4.548-96; 2) СанПиН 2.2.2/2.4.1340-03;

	электромагнитных излучений. 3) Недостаточная освещенность рабочей зоны. 4) Монотонный режим работы	2) Опасность возникновения пожара.	3) СП 52.13330.2011; 4) ГОСТ Р 12.1.019-2009 ССБТ; 5) СНиП 21-01-97; 6) СН 2.2.42.1.8.562-96 7) СанПиН 2.2.12.1.1.1278-03

6.2.1 Вредные производственные факторы

6.2.1.1 Отклонение показателей микроклимата

Микроклимат производственных помещений - метеорологические условия внутренней среды помещений, которые определяются действующими на организм человека сочетаниями температуры, влажности, скорости движения воздуха и теплового излучения; комплекс физических факторов, оказывающих влияние на теплообмен человека с окружающей средой, на тепловое состояние человека и определяющих самочувствие, работоспособность, здоровье и производительность труда. Показатели микроклимата: температура воздуха и его относительная влажность, скорость его движения, мощность теплового излучения.

Основные виды работ, выполняемые инженером-программистом, по степени физической тяжести, относятся к категории легких работ. Оптимальные величины показателей микроклимата на рабочих местах производственных помещений, в соответствии с периодом года и категорией работ, согласно СанПиН 2.2.4.548-96[12], предоставлены в таблице 18

Таблица 18 – Оптимальные параметры микроклимата производственных помещений оператора ПЭВМ

Период года	Температура воздуха, ºС	Температура поверхностей, ºС	Относительная влажность, %	Скорость движения воздуха, м/с
Холодный	**22–24**	**21–25**	**60–40**	**0,1**
Теплый	**23–25**	**22–26**	**60–40**	**0,1**

Параметры микроклимата в помещении не превышают допустимых значений. Для поддержания комфортной температуры воздуха помещение

оснащено кондиционером. Естественная вентиляция воздуха обеспечивается проветриванием помещения во время перерывов. В холодный период года оптимальные параметры микроклимата обеспечиваются системами отопления и кондиционирования воздуха.

6.2.1.2 Повышенный уровень электромагнитных излучений

Уровень электромагнитных излучений на рабочем месте оператора ПЭВМ является вредным фактором производственной среды, величины параметров которого определяются СанПиН 2.2.2/2.4.1340-03[13]. Основными источниками электромагнитных излучений в помещениях для работы операторов ПЭВМ являются дисплеи компьютеров, сеть электропроводки, системный блок, устройства бесперебойного питания, блоки питания.

Излучения, применительно к дисплеям современных ПЭВМ, можно разделить на следующие классы:

Переменные электрические поля (5 Гц – 400 кГц);

Переменные магнитные поля (5 Гц – 400 кГц).

Воздействие данных излучений на организм человека носит необратимый характер и зависит от напряженности полей, потока энергии, частоты колебаний, размера облучаемого тела. При воздействии полей, имеющих напряженность выше предельно допустимого уровня, развиваются нарушения нервной системы, кровеносной сердечно-сосудистой системы, органов пищеварения и половой системы [14].

В таблице 19 приведены допустимые уровни параметров электромагнитных полей

Таблица 19 – Временные допустимые уровни электромагнитных полей, создаваемых ПЭВМ на рабочих местах

Наименование параметров	Допустимые значения

Напряженность электрического поля	в диапазоне частот 5 Гц - 2 кГц	25 В/м
	в диапазоне частот 2 кГц - 400 кГц	2,5 В/м
Плотность магнитного потока	в диапазоне частот 5 Гц - 2 кГц	250 нТл
	в диапазоне частот 2 кГц - 400 кГц	25 нТл
Напряженность электростатического поля		15 кВ/м

6.2.1.3 Недостаточная освещенность рабочей зоны

Недостаточная освещенность рабочей зоны является вредным производственным фактором, возникающим при работе с ПЭВМ, уровни которого регламентируются СП 52.13330.2011[15], в соответствие с которым были выделены следующие требования к освещенности в помещениях и на рабочих местах с ПЭВМ:

1. В помещениях с ПЭВМ искусственное освещение должно быть равномерным. Для работы преимущественно с документами, допускается комбинированная система освещения.
2. Для поддержания оптимальных условий труда необходимо ограничивать сильную прямую и отраженную блёскость от осветительных приборов, при этом яркость светящихся поверхностей должна быть не выше 200кд/ кв. м..
3. Искусственное освещение рекомендуется создавать с помощью люминесцентных ламп типа ЛБ мощностью до 250 Вт. Для местного освещения разрешено использование ламп накаливания в светильниках.
4. Для поддержания оптимальных условий труда в помещениях с ПЭВМ необходимо проводить регулярную замену перегоревших ламп, а также мытье стекол и отчистка оконных проемов и осветительных приборов не менее двух раз в год.

В рассматриваемом помещении применяется совмещенное освещение.

Основным источником освещения является 6 люминисцентых светильников с зеркальными решетками, имеющие габаритные размеры длина – 543 мм, ширина – 543 мм. В каждом из светильников установлено 4 люминесцентные лампы типа ЛБ-40. Светильники расположены над рабочими местами в 2 ряда и создают равномерное освещение рабочих мест.

6.2.1.4 Монотонный режим работы

При работе с ПЭВМ основным фактором, влияющим на нервную систему программиста или пользователя является огромное количество информации, которое он должен воспринимать, что влияет на сознание и психофизическое состояние из-за монотонности работы. Поэтому меры, позволяющие снизить воздействие этого вредного производственного фактора, которые регулируются СанПиН 2.2.2/2.4.1340-03, являются важными в работе оператора ПЭВМ. Они позволяют увеличить производительность труда и предотвратить появление профессиональных болезней.

Вид трудовой деятельности относиться к группе В – творческая работа в режиме диалога с ПЭВМ. Категория тяжести для данной группы определяется исходя из общего времени работы с ПЭВМ за рабочий день. В зависимости от уровня нагрузки за рабочую смену устанавливается суммарное время регламентированных перерывов.

При 8-часовой рабочей смене и III категорией тяжести (до 6 часов непрерывной работы с компьютером) суммарное время перерывов должно составлять 90 минут. Рекомендуется организовывать перерывы на 10-15 минут через каждые 45-60 минут работы. Продолжительность постоянной работы с ПК не должна превышать 2 часов.

6.2.2 Опасные производственные факторы

6.2.2.1 Опасность поражения электрическим током

Поражение электрическим током является опасным производственным фактором и, поскольку оператор ПЭВМ имеет дело с электрооборудованием, то

вопросам электробезопасности на его рабочем месте должно уделяться много внимания. Нормы электробезопасности на рабочем месте регламентируются СанПиН 2.2.2/2.4.1340-03, вопросы требований к защите от поражения электрическим током освещены в ГОСТ Р 12.1.019-2009 ССБТ[16].

Электробезопасность – система организационных и технических мероприятий и средств, обеспечивающих защиту людей от вредного и опасного воздействия электрического тока, электрической дуги, электромагнитного поля и статического электричества.

Производственное помещение, в котором проводилось исследование расположено большое количество техники, но так как отсутствует влажность, высокая температура, токопроводящая пыль и возможность одновременного соприкосновения с имеющими соединение с землей металлическими предметами и металлическими корпусами оборудования по опасности электропоражения по классификации ПУЭ «Правила устройства электроустановок.» помещение считается без повышенной опасности. Основными причинами поражения человека электрическим током могут быть следующие:

– Непосредственное прикосновение к токоведущим частям, оказавшимся под напряжением;
– соприкосновение с конструктивными частями, оказавшимися под напряжением.

Электрический ток оказывает тепловое (ожоги, нагрев сосудов), механическое (разрыв тканей, сосудов при судорожных сокращениях мышц), химическое (электролиз крови), биологическое (раздражение и возбуждение живой ткани) или комбинированное воздействие.

Основными средствами и способами защиты от поражения электрическим током являются:

- Недоступность токоведущих частей для случайного прикосновения;

- защитное заземление, зануление или отключение;
- вывешивание предупреждающих надписей;
- контроль за состоянием изоляции электрических установок;
- использование дополнительных средств защиты.

6.2.2.2 Опасность возникновения пожара

Возникновение пожара является опасным производственным фактором, т.к. пожар на предприятии наносит большой материальный ущерб, а также часто сопровождается травмами и несчастными случаями. Регулирование пожаробезопасности производится СНиП 21-01-97[17].

В помещениях с ПЭВМ повышен риск возникновения пожара из-за присутствия множества факторов: наличие большого количества электронных схем, устройств электропитания, устройств кондиционирования воздуха; возможные неисправности электрооборудования, освещения, или неправильная их эксплуатация может послужить причиной пожара.

6.2.3 Мероприятия и рекомендации по устранению и минимизации

Для поддержания нормальных значений параметров микроклимата на рабочих метах рекомендуется оснащать их системами отопления, вентиляции и кондиционирования воздуха. Также, в некоторых случаях, целесообразно обеспечить питьевое водоснабжение. В помещениях для работы с ПЭВМ должна производиться ежедневная влажная уборка, а также систематическое проветривание после каждого часа работы [12].

Для защиты операторов ПЭВМ от негативного воздействия электромагнитных полей в первую очередь необходимо, чтобы используемая техника удовлетворяла нормам и правилам сертификации. При работе с ПЭВМ установлены регламентированные перерывы, а также иногда предусмотрено использование экранов и фильтров в целях защиты оператора [13].

Для создания и поддержания благоприятных условий освещения для операторов ПЭВМ, их рабочие места должны соответствовать санитарно-

эпидемиологическим правилам СанПиН 2.2.2/2.4.1340-03. Для рассеивания естественного освещения следует использовать жалюзи на окнах рабочих помещений. В качестве источников искусственного освещения должны быть использованы люминесцентные лампы, лампы накаливания – для местного освещения [13].

Для предупреждения преждевременной утомляемости пользователей ПЭВМ рекомендуется организовывать рабочую смену путем чередования работ с использованием ПЭВМ и без него. В случаях, когда характер работы требует постоянного взаимодействия с компьютером (работа программиста-разработчика) с напряжением внимания и сосредоточенности, при исключении возможности периодического переключения на другие виды трудовой деятельности, не связанные с ПЭВМ, рекомендуется организация перерывов на 10–15 мин. через каждые 45–60 мин. работы. При высоком уровне напряженности работы рекомендуется психологическая разгрузка в специально оборудованных помещениях.

К мероприятиям по предотвращению возможности поражения электрическим током относятся:

При производстве монтажных работ необходимо использовать только исправный инструмент, аттестованный службой КИПиА;

С целью защиты от поражения электрическим током, возникающим между корпусом приборов и инструментом при пробое сетевого напряжения на корпус, корпуса приборов и инструментов должны быть заземлены;

При включенном сетевом напряжении работы на задней панели должны быть запрещены;

Все работы по устранению неисправностей должен производить квалифицированный персонал;

Необходимо постоянно следить за исправностью электропроводки [13, 16].

Для профилактики организации действий при пожаре должен проводиться следующий комплекс организационных мер: должны обеспечиваться регулярные проверки пожарной сигнализации, первичных средств пожаротушения; должен проводиться инструктаж и тренировки по действиям в случае пожара; не должны загромождаться или блокироваться пожарные выходы; должны выполняться правила техники безопасности и технической эксплуатации электроустановок; во всех служебных помещениях должны быть установлены «Планы эвакуации людей при пожаре и других ЧС», регламентирующие действия персонала при возникновении пожара.

Для предотвращения пожара помещение с ПЭВМ должно быть оборудовано первичными средствами пожаротушения: углекислотными огнетушителями типа ОУ-2 или ОУ-5; пожарной сигнализацией, а также, в некоторых случаях, автоматической установкой объемного газового пожаротушения [17].

6.3 Экологическая безопасность

В данном разделе рассматривается воздействие на окружающую среду деятельности по разработке проекта, а также самого продукта в результате его реализации на производстве.

Разработка программного обеспечения и работа за ПЭВМ не являются экологически опасными работами, потому объект, на котором производилась разработка продукта, а также объекты, на которых будет производиться его использование операторами ПЭВМ относятся к предприятиям пятого класса, размер селитебной зоны для которых равен 50 м [18].

Деятельность человека, носящая производственный характер, оказывает негативно влияние на окружающую среду. Деятельность программиста не связана с производством и не является источником выброса опасных и вредных веществ в окружающую среду. Однако работа с ПЭВМ и другой оргтехникой требует большого объема энергопотребления, что в свою очередь приводит к большому потреблению электростанциями различных видов топлива,

выбрасывая при этом вредные вещества в окружающую среду. Больше всего от выбросов электростанций страдает атмосфера. Данный проект помогает снизить объем энергопотребления за счет уменьшения необходимых вычислительных мощностей.

На сегодняшний день в мире разрабатываются все новые и новые способы защиты атмосферы, например, альтернативные источники энергии или использование энергосберегающих систем.

Современная техника не содержит в себе большой концентрации вредных веществ, поэтому возможно ее использование без нанесения значительного вреда сотрудникам или окружающей среды.

Выбрасываемыми отходами в здание, в котором находится рассматриваемое помещение, являются бытовой мусор (например, макулатура, который выбрасывается в контейнеры, вывоз которых осуществляется регулярно) а также сточные воды, за очистку которых отвечает городской водоканал.

Также стоит отметить тот факт, что данная разработка снижает размер парка вычислительной техники, необходимой для работы организации, таким образом, представленная работа способствует ресурсосбережению.

6.4 Безопасность в чрезвычайных ситуациях

Чрезвычайная ситуация (ЧС) - это обстановка на определенной территории, сложившаяся в результате аварии, опасного природного явления, катастрофы, стихийного или иного бедствия, которые могут повлечь или повлекли за собой человеческие жертвы, ущерб здоровью людей или окружающей природной среде, значительные материальные потери и нарушение условий жизнедеятельности людей.

Чрезвычайные ситуации могут носить различный характер: технологический, природный, социальный, военный и т.д. Многие из чрезвычайных ситуаций являются форс-мажорными обстоятельствами, исключение которых невозможно. Однако необходимо выполнение всех мер по

предотвращению ЧС.

При работе на персональной электронно-вычислительной машине самым вероятной ЧС является возможность пожара.

Пожар – неконтролируемое возгорание и горение, наносящее вред жизни и здоровью людей, также материальный ущерб. Причинами возникновения пожаров чаще всего являются: короткие замыкания, несоблюдение правил эксплуатации производственного оборудования и электрических устройств, разряды статического электричества.

С целью уменьшения вреда жизни и здоровью населения и материального ущерба, наносимого пожаром необходимо реализация комплекса профилактических мероприятий, направленных на предупреждение и (или) устранение пожара.

Предупреждение пожаров является основной задачей руководителей и инженерно-технических работников предприятий. В работе по предупреждению пожаров большая роль принадлежит личному составу пожарной охраны, который проводит целый комплекс мероприятий по противопожарной защите объектов, осуществляет постовую и дозорную службу, выявляет имеющиеся недостатки и принимает меры к их своевременному устранению в соответствии с ФЗ от 22.07.2008 N 123-ФЗ (ред. от 13.07.2015) "Техническим регламентом о требованиях пожарной безопасности".

В соответствии с СП 12.13130.2009 «Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности» помещение, в котором выполнялась ВКР, относится к наименее опасной категории (Д) с пониженной пожароопасностью[19]. Само здание по взрывопожарной и пожарной опасности относится к категории (Д). Наружных установок здание не имеет.

К пожарно-профилактическим мероприятиям относятся:

- выбор качественного электрооборудования и правильых способов его

монтажа с учетом пожароопасности территории, а также регулярный контроль исправности защитных устройств и аппаратов на электрооборудовании, постоянный контроль за надлежащей эксплуатацией электроустановок и электросетей;

- систематический надзор за выполнением правил технической эксплуатации электрических устройств;
- регулярная проверка знаний противопожарной безопасности.
- пожарно-техническая проверка для выявления состояния объектов представителями пожарного надзора с последующим выполнением предписаний и приказов;
- систематическое выполнение противопожарных работ;
- проверка наличия и исправности первичных средств пожаротушения;
- проведение учебных тревог и эвакуаций персонала организации;
- прохождение противопожарного инструктажа.
- для предотвращения пожара помещение с ПЭВМ должно быть оборудовано первичными средствами пожаротушения: углекислотными огнетушителями типа ОУ-2 или ОУ-5; пожарной сигнализацией, а также, в некоторых случаях, автоматической установкой объемного газового пожаротушения [20].

6.5 Правовые и организационные вопросы обеспечения безопасности

6.5.1 Правовые нормы трудового законодательства для рабочей зоны оператора ПЭВМ

Государственный надзор и контроль в организациях независимо от организационно-правовых форм и форм собственности осуществляют специально уполномоченные на то государственные органы и инспекции в соответствии с федеральными законами.

Согласно трудовому кодексу РФ[21] и СанПиН 2.2.2/2.4.1340-03[13]:

- продолжительность рабочего дня не должна превышать 40 часов в неделю.

– не рекомендуется работать за компьютером более 6 часов за смену;
– рекомендуется делать перерывы в работе за ПК продолжительностью 10-15 минут через каждые 45-60 минут работы;
– продолжительность непрерывной работы за компьютером без регламентированного перерыва не должна превышать 1 час;
– во время регламентированных перерывов целесообразно выполнять комплексы упражнений и осуществлять проветривание помещения.
– при работе с ПЭВМ в ночную смену (с 22 до 6 ч), независимо от категории и вида трудовой деятельности, продолжительность регламентированных перерывов следует увеличивать на 30 %.

Государственный санитарно-эпидемиологический надзор за производством и эксплуатацией ПЭВМ осуществляется в соответствии с СанПиН 2.2.2/2.4.1340-03.

Производственный контроль за соблюдением санитарных правил осуществляется производителем и поставщиком ПЭВМ, а также предприятиями и организациями, эксплуатирующими ПЭВМ.

Существуют также специализированные органы, осуществляющие государственный контроль и надзор в организациях на предмет соблюдения существующих правил и норм. К таким органам относятся:

– Федеральная инспекция труда;
– Государственная экспертиза условий труда Федеральная служба по труду и занятости населения (Минтруда России Федеральная служба по экологическому, технологическому и атомному надзору (Госгортехнадзор, Госэнергонадзор, Госатомнадзор России)).
– Федеральная служба по надзору в сфере защиты прав потребителей и благополучия человека (Госсанэпиднадзор России) и др.

6.5.2 Организационные мероприятия при компоновке рабочей зоны

К мероприятиям, относящимся к компоновке рабочей зоны относятся

работы по организации рабочего места пользователя, позволяющие наилучшим образом организовать деятельность работника, делая его работу максимально удобной и безопасной.

На основании СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» и ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования»[22] для данной выпускной квалификационной работы были выявлены основные требования к организации рабочих мест пользователей ПЭВМ.

Помещения должны иметь естественное и искусственное освещение. Расположение рабочих мест за мониторами для взрослых пользователей в подвальных помещениях не допускается.

Площадь на одно рабочее место с компьютером для взрослых пользователей должна составлять не менее 6 $м^2$, а объем не менее - 20 $м^3$.

Помещения с компьютерами должны оборудоваться системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляцией.

Для внутренней отделки интерьера помещений с компьютерами должны использоваться диффузно-отражающие материалы с коэффициентом отражения для потолка — 0,7-0,8; для стен — 0,5-0,6; для пола — 0,3-0,5.

Поверхность пола в помещениях эксплуатации компьютеров должна быть ровной, без выбоин, нескользкой, удобной для очистки и влажной уборки, обладать антистатическими свойствами.

В помещении должны находиться аптечка первой медицинской помощи, углекислотный огнетушитель для тушения пожара.

Заключение

С каждым годом проблема хранения больших объемов данных становится все более актуальной. В связи с этим, на рынке растет и количество систем хранения, но по ряду различных причин (в особенности, из-за высокой стоимости), они могут не подойти организациям, желающим минимизировать свои расходы.

При проектировании распределенного файлового хранилища встает ряд важных проблем, таких как структурирование и быстрый поиск нужной информации, а также обеспечение отказоустойчивости и сокращения времени обслуживания запросов. Также, крайне важен выбор архитектуры разрабатываемого решения, поскольку от нее зависит, будет ли система удовлетворять заявленным требованиям.

В данной работе был проведен обзор и сравнение некоторых популярных решений на рынке хранения больших данных. Была рассмотрена их внутренняя структура и принципы работы.

Также была проведена оценка влияния многопоточности и структурирования информации на эффективность работы системы. В результате проведенных исследований было выяснено, что при распараллеливании работы командного центра уменьшается время обслуживания отдельного запроса. Для структурирования данных использовались метаописания файлов и разделение на тематические разделы, что позволило осуществлять эффективный поиск информации.

При создании обобщенной функциональной схемы хранилища была выбрана мультиагентная архитектура, которая дает возможность практически неограниченного горизонтального масштабирования системы и позволяет равномерно распределять нагрузку на отдельные узлы.

С точки зрения финансового менеджмента и ресурсоэффективности, данное решение позволит сократить расходы на покупку дорогостоящего

оборудования, а также позволит обойтись без аренды дискового пространства у различных вендоров.

При рассмотрении вопроса безопасности жизнедеятельности и экологической безопасности было выяснено, что разработка не наносит ущерб окружающей среде, а ее эксплуатация не требует соблюдения особых норм БЖД. Таким образом, при работе с файловым хранилищем, достаточно соблюдать основные санитарные нормы работы с ПК, а также требования к эргономике рабочего места.

Список публикаций.

1. Разработка корпоративных порталов на платформе SharePoint с дружественным пользовательским интерфейсом / Технологии Microsoft в теории и практике программирования: сборник трудов XI Всероссийской научно — практической конференции студентов, аспирантов и молодых учёных // Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2014. – с. 197 - 199.
2. Программная реализация кластеризатора для метода k-средних с уменьшенной размерностью пространства признаков / Технологии Microsoft в теории и практике программирования: сборник трудов X Всероссийской научно — практической конференции студентов , аспирантов и молодых учёных (19–20 марта 2013 г.) // Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2013. – с. 71 - 73
3. Математическое и программное обеспечение распределенной мультиагентной универсальной децентрализованной масштабируемой информационно-вычислительной системы / III Международная научная конференция «Информационные технологии в науке, управлении, социальной сфере и медицине» – Томск: Изд-во Томского политехнического университета, 2016. - В печати.

Список использованных источников

1. Big Data и ECM: рассмотрим практические примеры [Электронный ресурс] // URL: http://ecm-journal.ru/post/Big-Data-i-ECM-rassmotrim-prakticheskie-primery.aspx, (дата обращения: 20.04.2016).
2. Файловая система [Электронный ресурс] // URL: http://citforum.ru/operating_systems/sos/glava_10.shtml (дата обращения: 20.04.2016).
3. Структура проекта Hadoop организации Apache Software Foundation [Электронный ресурс] // URL: http://network-journal.mpei.ac.ru/cgi-bin/main.pl?l=ru&n=27&pa=12&ar=4 (дата обращения: 22.04.2016).
4. Hadoop, часть 1: развертывание кластера [Электронный ресурс] // URL: https://habrahabr.ru/company/selectel/blog/198534/ (дата обращения: 22.04.2016).
5. HDFS Architecture Guide [Электронный ресурс] // URL: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (дата обращения: 22.04.2016).
6. Технологии облачных вычислений [Электронный ресурс] // URL: http://www.intuit.ru/studies/courses/3508/750/lecture/27416?page=7 (дата обращения: 23.04.2016).
7. Распределенная файловая система GFS (Google File System) [Электронный ресурс] // URL: https://habrahabr.ru/post/73673/ (дата обращения: 23.04.2016).
8. Определение метаданных [Электронный ресурс] // URL: http://www.elbib.ru/index.phtml?page=elbib/rus/methodology/md_rev/md_def (дата обращения: 2.05.2016).
9. Набор элементов метаданных Dublin Core (Дублинского ядра) Версия 1.1: Справочное описание [Электронный ресурс] // URL: http://www.rusmarc.ru/soft/dc.html (дата обращения: 2.05.2016).
10. Java API documentation. Class Thread [Электронный ресурс] // URL: https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html (дата обращения: 3.05.2016).
11. ГОСТ 12.0.003-74. Система стандартов безопасности труда. Опасные и

вредные производственные факторы. Классификация // Библиотека ГОСТов. 2016. URL: http://vsegost.com/Catalog/41/41131.shtml (дата обращения: 9.05.2016).
12. СанПиН 2.2.4.548-96. Санитарные правила и нормы. Гигиенические требования к микроклимату производственных помещений // Библиотека гостов и нормативов. 2016. URL: http://ohranatruda.ru/ot_biblio/normativ/data_normativ/5/5225/ (дата обращения: 9.05.2016).
13. СанПиН 2.2.2/2.4.1340-03. Санитарно-эпидемиологические правила и нормы. Гигиенические требования к персональным электронно-вычислительным машинам и организации работы // Библиотека гостов и нормативов. 2016. URL: http://www.ohranatruda.ru/ot_biblio/normativ/data_normativ/39/39082/#i72870 (дата обращения: 9.05.2016).
14. Белов С. В. Безопасность жизнедеятельности и защита окружающей среды (техносферная безопасность): учебник / С. В. Белов. – 2-е изд., испр. и доп. – М.: Издательство Юрайт, 2011. – 680 с.
15. СП 52.13330.2011. Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95 // Докипедия. 2016. URL: http://dokipedia.ru/document/5147250 (дата обращения: 9.05.2016).
16. ГОСТ Р 12.1.019-2009 ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты // Электронный фонд правовой и нормативно-технической документации. 2010. URL: http://docs.cntd.ru/document/gost-r-12-1-019-2009-ssbt (дата обращения: 9.05.2016).
17. СНиП 21-01-97. Пожарная безопасность зданий и сооружений // Библиотека гостов и нормативов. 2016. URL: http://www.ohranatruda.ru/ot_biblio/normativ/data_normativ/2/2107/ (дата обращения: 10.05.2016).
18. СанПиН 2.2.1/2.1.1.1200-03. Санитарно-эпидемиологические правила и нормативы. Санитарно-защитные зоны и санитарная классификация

предприятий, сооружений и других объектов // Библиотека гостов и нормативов. 2016. URL: http://ohranatruda.ru/ot_biblio/normativ/data_normativ/11/11774/ (дата обращения: 10.05.2016).
19. НПБ 105-03 Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности // Электронный фонд правовой и нормативно-технической документации. 2016. URL: http://docs.cntd.ru/document/1200032102 (дата обращения: 10.05.2016).
20. ППБ 01–03. Правила пожарной безопасности в Российской Федерации. – М.: Министерство Российской Федерации по делам гражданской обороны, чрезвычайным ситуациям и ликвидации последствий стихийных бедствий, 2003
21. Трудовой кодекс Российской Федерации" от 30.12.2001 N 197-ФЗ (ред. от 30.12.2015) // Консультант Плюс. 2015. URL: http://www.consultant.ru/document/cons_doc_law_34683/?utm_campaign=law_doc&utm_source=google.adwords&utm_medium=cpc&=utm_content=Labor%20Code&gclid=CjwKEAjwgPe4BRCB66GG8PO69QkSJAC4EhHhU-5yAFZCJfmzkTLNGnrpgHHAYFPhhPzRo-sZGWmqnBoCPynw_wcB (дата обращения: 10.05.2016).
22. ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования» Классификация // Библиотека ГОСТов. 2016. URL: http://vsegost.com/Catalog/31/31970.shtml (дата обращения: 9.05.2016).

Приложение А. Раздел ВКР на иностранном языке

Раздел 1

Обзор и сравнение распределенных файловых систем

Студент:

Группа	ФИО	Подпись	Дата
8ВМ4Б	Зензин Алексей Сергеевич		

Консультант кафедры ИПС:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Ботыгин И. А.	к.т.н., доцент		

Консультант – лингвист кафедры ИЯ ИК:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
зав.каф.	Сидоренко Т. В.	к.п.н., доцент		

1. Distributed file systems overview and comparison

1.1 File systems

File system - is a method of storage, naming and access to data by using different data carriers. Also often file system is understood as some program interface serves as a layer file access API (application programming interface) and data carriers.

Thus, in a broaden sense the term "file system" includes:

1. collection of all the files on a disk;
2. sets of data structures, which are used for file control, for example, file folders, file descriptors, tables denoting the distribution of free and used space on a disk;
3. a set of software tools, which implements file controlling, in particular: creation, destruction, reading, writing, naming, searching and other operations connected with files.[2]

There are several types of file systems (for different types of media, virtual file system, specialized ones etc.). However, from the viewpoint of the subject of this study, it will be more interesting to examine the distributed file systems. According to the definition of this term, these systems are not focused on a single machine, and are able to cover a large number of computers and servers that are not necessarily in close geographical location to each other. One of the main advantages of such file systems is that they are infinitely scalable and can store any volume data.

Now it is necessary to describe some distributed file systems which are currently at the market.

1.2 An overview of solutions used for storing large data

1.2.1 Hadoop (HDFS)

Hadoop is the project of Apache Software Foundation, freely available set of tools, libraries and framework for the development and execution of distributed programs running on clusters of hundreds or thousands of nodes. It is used for creating search and context mechanisms for heavily loaded web-sites, for example

Yahoo! And Facebook. Developed in Java within the computing MapReduce paradigm, according to which the application is divided into a large number of identical elementary tasks parformed on the cluster nodes and naturally reducible to the final result.[3]

Hadoop is the most widespread solution at bigdata market. Many IT – vendors (Amazon, Cloudera, Dell) provide storage construction and processing data on the basis of this project.

Creators of Cloudera Hadoop, freely available distributive, cite the following system requirements:

- «light» configuration (1U) — 2 six-core processors, 24-64 GB memory, 8 HDD with 1-2 TB capacity;
- rational configuration (1U) — 2 six-core processors, 48-128 GB memory, 12-16 HDD (1 or 2 TB), directly connected through the motherboard controller;
- «heavy» configuration for storages (2U): 2 six-core processors, 48-96 GB memory, 16-24 HDD.
- configuration for intensive computing: 2 six-core processors, 64-512 GB memory, 4-8 HDD with 1 – 2 TB capacity. [4]

Hadoop is divided to several modules, including HDFS, which is a distributed file system. Files in this system are divided into blocks which allow storing large files. All blocks in HDFS (except the last block of the file) have the same size and every block can be arranged at several nodes, a size of block and replication coefficient (number of nodes, on which every block must be placed) is defined in the settings at the file level. Through replication the stability of the distributed system to failures of individual nodes is ensure. Files in HDFS can be recorded only once (modification is not supported), and the record to the file at one time can be responsible for only one process. Files organization in namespace - is traditional hierarchical: there is a root directory, the sub-directory is under the support, where the directory as well as files can be located.

It is necessary to say a few words about the file system architecture. HDFS

cluster consists of a single NameNode, which is the master – server as well. It is responsible for the file system namespace and access control for the clients. Also, the cluster includes several DataNodes, directly responsible for the data storage. HDFS allows users to store the files in the splitted form (divided into blocks). In this case, blocks are stored on the multiple data nodes. Also, data nodes are responsible for checking, replication and data recovery. They also carry out recording / reading the data. A node name is responsible for file operations (opening, closing, renaming), and stores the meta-description of files. [5] The scheme of HDFS cluster is shown in Figure 1.

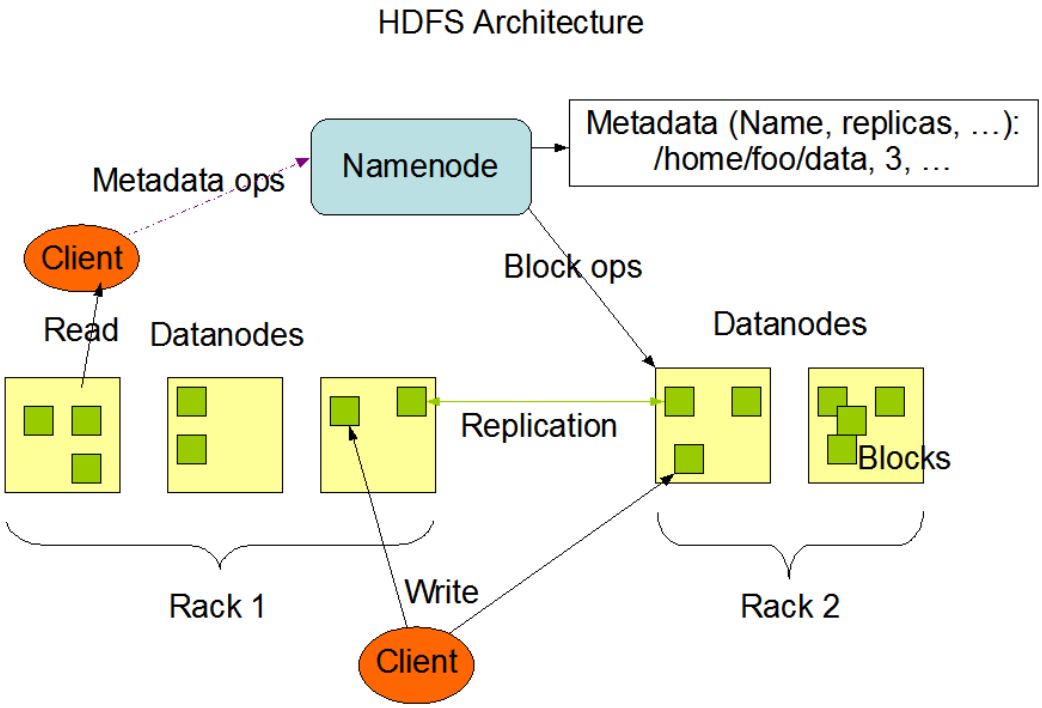

Figure 1 – HDFS cluster scheme

Revealing the pattern of HDFS provides for a central names node, storing metadata of a file system and meta-information about the distribution of the blocks, and a series of data nodes, directly storing the blocks of files. The name node is responsible for handling the level of file and directory operations - opening and closing files, manipulation of directories, data nodes directly fulfill the operations of recording and reading the data. The node names and data nodes are supplied with a Web server, which shows the current status of the components and allows to view the

contents of the file system. Administrative functions are available from the command line interface. Typically, HDFS is performed to an end user as a cloud service, being deployed on the supplier facilities, or in the form of some adjusted clusters to be installed directly for a client. [3]

1.2.2 Amazon S3

Amazon Simple Storage Service (Amazon S3) - an online web service offered by Amazon Web Services, which provides an opportunity to store and retrieve any amount of data, at any time, from anywhere in the network, the so-called file hosting. With the help of Amazon S3 we achieve a high scalability, reliability, high speed and inexpensive data storage infrastructure. First it was appeared in March 2006 in the US and in November 2007 in Europe. [6]

When working with Amazon S, a consumer pays for the used space, as well as for incoming / outgoing traffic.

Unfortunately, the information about the internal structure and principles of construction of the system there is no publicly available.

1.2.3 Google File System

Google File System is a proprietary distributed file system, used for the internal needs of the company Google. Many details of its implementation are a trade secret, but some facts are still known - for example, this file system allows to store the large volumes of files by dividing them into 64 MB chunks, and also provides a storage reliability through redundancy of data. There is a free implementation of this file system, used for the needs of the UK government. [5]

Google File System was built based on the following criteria:

1. The system includes a large number of inexpensive hardware, so malfunctions are inevitable. To find them in the system a monitoring tool is used.
2. The system must store a large number of heterogeneous file (as small, up to 100 MB, and of several gigabytes).
3. Due to the large number of potential query, the system must be able to

synchronize the work with the same file.

4. It is necessary to distribute the load evenly between the individual storage nodes.

GFS is organized hierarchically as a tree of directories. A standard set of file operations provided in the system: create, delete, open, close, read and write. Also, backup files are supported.

The components of the system are the master - servers and chunk - servers.A GFS cluster consists of one main master machine and multiple machines, directly storing the fragments of the files in the— chunk-servers. As mentioned above, files in the GFS are split into chunks of a fixed size. Each chunk has a unique and global 64 - bit key, which is issued by the master to create the chunk. The chunk servers store chunks like ordinary files on the hard disk. To ensure the reliability of data storage, each chunk is replicated (usually up to three times).

Another feature of GFS is the capacity of representing the collection of files in the name space as a table that displays the path to the file in metadata instead of storing files in the directory structure.

The master is responsible for the work with metadata of the entire file system. The metadata contains the information about namespaces, control access to data, and the position of individual chunks. In addition to the work with metadata, the master carries out all activities related to the control system: control of the free chunks, chunks moving between servers, as well as garbage collection (collection of unused chunks). If you want to receive the file, the client communicates with the master - server and learns from him the location of the desired chunk. Then, working with the file carries the chunk-server that holds the fragment. At the same time, under these changes should be reflected in all replicas of this fragment. The recording process can be described by the following steps:

- The client sends a message to the master that contains the request for the chunk.
- The master sends information about the primary replica, as well as all

secondary.

- The client sends the new data to all replicas. Chunk - servers store this data in a special buffer
- When the data will be sent to all replicas, the client sends a request to record the primary replica. It, in turn, establishes the order, in which will be made all the changes on the file (there may be more than one, from different clients)
- The primary replica sends requests to all other replicas, while they carry out exactly the same procedure as the main.
- After receiving the response from all the replicas, the primary replica sends a response to the client about completing of operation.

Also, when writing new chunks, the system tries to pick up a chunk- servers , which are located at a maximum distance from each other within the cluster (for example, uses a variety of rack). [7]

1.3 Conclusion

Based on the basic principles of different kinds of file systems and how they work, the most logical is to organize the storage of files in a distributed file system. According to the results of the overview on storage systems for large data offered at the market, we can conclude that, for various reasons (the need to purchase and maintain expensive equipment, the cost of rent facilities from different vendors, etc.) these solutions may not be appropriate for organizations who are willing to organize its own, storage system at the minimal cost.