МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

**кафедра “Інформаційні системи та мережі”**

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи на тему:

**Інформаційна система розподіленого зберігання файлів**

**Студента групи КН-411 Нанівського О.І.**

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Консультант** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Нормоконтроль**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Рецензент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Завідувач кафедри ІСМ Литвин В. В.**

«\_28\_\_» \_\_травня\_\_\_\_\_ **2021 р.**

**ЛЬВІВ – 2021**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра «Інформаційні системи та мережі»

Спеціальність 122 "Комп'ютерні науки"

Перший (бакалаврський) рівень вищої освіти

«ЗАТВЕРДЖУЮ**»**

Завідувач кафедри ІСМ Литвин В.В.

«\_28\_» \_\_травня\_\_\_ 2021 р.

**ЗАВДАННЯ**

**на бакалаврську кваліфікаційну роботу студента групи КН-411**

Нанівського Олега Ігоровича

1. Тема роботи “Інформаційна система розподіленого зберігання файлів”

затверджена наказом по НУ «ЛП» від «\_10\_»\_\_березня\_\_\_2021р. №\_652-4-08\_\_\_\_\_

2. Термін здачі студентом закінченої роботи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані для роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Перелік програмних продуктів, які належить використати в процесі розроблення роботи (проекту): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Консультування роботи, із зазначенням розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультанти | Підпис, дата | |
| завдання видав | завдання отримав |
| Економічний |  |  |  |
|  |  |  |  |

8. Дата, коли видано завдання \_\_\_\_\_\_\_1.02.2021 р.\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Завдання отримав до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Етапи дипломної роботи | Термін виконання етапів роботи | Примітки |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**АНОТАЦІЯ**

В даній роботі аналізуються проблеми зберігання файлів великих обсягів та розроблена розподілена файлова система за допомогою мови Python. Система складається з трьох складових: клієнтської частини, сервера і серверів даних. Обмін інформацією між ними здійснюється за допомогою RPC.

Архітектура системи розроблена за зразком GFS [[1]](#lsptd9pal5o5).

Ключові слова: розподілена файлова система, RPC, сервер, клієнт.

2000 символів

**ANNOTATION**

This paper analyzes problems of storing large files and describes developing of distributed file system using Python programming language. The system consists of three components: client part, server and data servers. The components communicate with each other using Remote Procedure Call (RPC) protocol.

The system architecture is based on GFS principles [[1]](#lsptd9pal5o5).

Keywords: distributed file system, RPC, server, client.

**ЗМІСТ**

ВСТУП 8

РОЗДІЛ 1 Аналіз джерел 9

1.1. Що таке розподілена файлова система 9

1.1.1. Короткий огляд 9

1.1.2. Переваги та властивості 9

1.2. Еволюція розподілених файлових систем 11

1.2.1. Network File System — NFS 11

1.2.2. Andrew file System — AFS 12

1.2.3. Google File system — GFS 15

1.2.4. Hadoop Distributed File System - HDFS 17

1.2.5. Обмеження і покращення HDFS і GFS 20

1.3. Використання розподілених файлових систем 23

1.4. Висновок 24

РОЗДІЛ 2 (20-25c.) Системний аналіз 26

2.1. Дерево цілей 26

2.2. Діаграми DFD для деталізації структури 27

2.3. Діаграми ієрархії процесів DFD 28

РОЗДІЛ 3 (20c.) Опис програмних засобів 29

3.1. Обґрунтування вибору засобів розв’язання задачі 29

3.2. Технічні характеристики обраних програмних засобів 29

РОЗДІЛ 4 (15c.) Практична реалізація 29

4.1. Опис програми 29

4.2. Інструкція користувача (IEEE STD 1063-2001) 29

4.3. Аналіз контрольного прикладу 29

РОЗДІЛ 5 Економічна 29

ВИСНОВКИ 29

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 30

ДОДАТКИ 33

# ВСТУП

Актуальність теми

Розподілені системи для зберігання файлів надають можливості зручного та безпечного зберігання використовуючи віддалені машини. Починаючи від їх першої імплементації NFS, яка комерційно використовувалася, розподілені файлові системи постійно вдосконалювалися і набували ще більшої популярності. Сьогодні вони є незамінними в сфері великих даних і хмарних обчислень. Проте від створення GFS, вони не зазнавали значних покращень в дизайні.

Мета і задачі дослідження

Дане робота має на меті продемонструвати принципи роботи розподілених файлових систем і їхні обмеження.

Об’єкт дослідження

Об’єктом дослідження є зберігання файлів.

Предмет дослідження

Предметом дослідження є збереження файлів на віддаленій машині з можливість доступу до

Наукова новизна одержаних результатів

Практичне значення одержаних результатів

практичне застосування

ступінь готовності

# РОЗДІЛ 1 Аналіз джерел

## 1.1. Що таке розподілена файлова система

### 1.1.1. Короткий огляд

Розподілена файлова система – це система, яка розподілена на багатьох серверах. Це дозволяє програмам отримувати або зберігати файли як локальні з будь-якого комп’ютера або мережі.

Головна мета розподіленої файлової системи – це дозволити фізично віддаленим користувачам ділитися їхніми даними і ресурсами використовуючи звичайну файлову систему.

Розподілена файлова система має два компоненти [[2]](#iegmtr4an47):

* Прозорість місце знаходження, яке здійснюється через простір імен.
* Надмірність використовуючи реплікацію файлів.

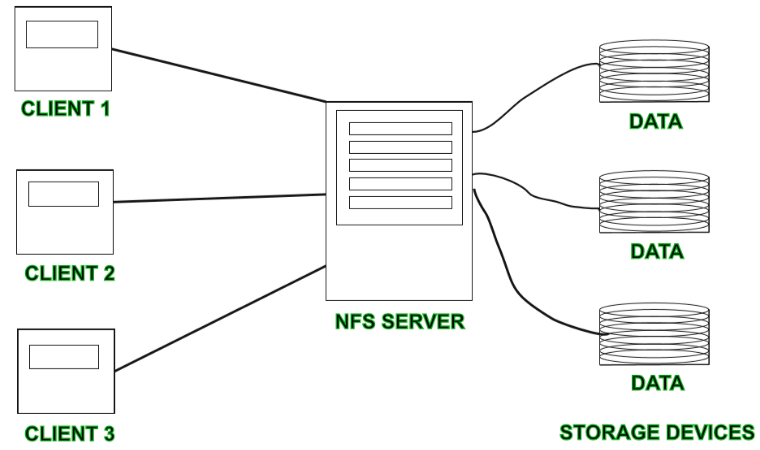
Прозорість приносить зручність для кінцевого користувача даючи йому змогу маніпулювати, читати і зберігати файли на локальній машині в той час, як сам процес відбувається на серверах.

В загальному розподілені файлові системи використовуються в LAN мережах, але вони також можуть бути використаними в WAN [[17]](#7rvmfvw9l7kq).

Окрім розподіленої файлової системи іншим варіантом віддаленого доступу до файлів є ділитися диском. Ця система дає клієнту контроль доступу, що призводить до недоступності файлів коли клієнт офлайн. Розподілена файлова система в свою чергу є толерантною до помилок і клієнт має доступ до даних навіть якщо, декілька вузлів системи є недоступними [[3]](#gbnf82fna7qk).

### 1.1.2. Переваги та властивості

Перевагами розподіленої файлової системи можна назвати обмеження в доступі базуючись на списку доступу або можливостей сервера і клієнта спираючись на то, як розроблено протокол. Оскільки сервер також надає одну центральну точку доступу припускається, що він буде справно працювати попри помилку окремих вузлів, як показано на *рис.1.1*. Проте єдина точка доступу також і є його слабкістю. Наприклад сервер може перестати працювати через DDoS атаку. Тому зазвичай використовують запасний сервер для резервного збереження даних [[3]](#gbnf82fna7qk).

*Рис.1.1 Абстрактна архітектура розподіленої файлової системи*

Розглянемо короткий огляд основних властивостей розподіленої файлової системи [[2]](#iegmtr4an47):

* Прозорість
  + Прозорість структури: клієнт не знає про кількість або розміщення файлових серверів. Велика кількість файлових серверів використовується для збільшення продуктивності, адаптивності та надійності.
  + Прозорість доступу: отримання файлу на локальній і віддалених машинах не має відрізнятися, система автоматично знаходить відповідний файл і надсилає його клієнту.
  + Прозорість імен: ім’я файлу не має змінюватися, або вказувати на його місце знаходження.
  + Прозорість реплікації: якщо файл скопійований на декількох вузлах, його копії і місце знаходження повинні бути сховані від інших вузлів.
* Мобільність користувача: система автоматично приносить домашню директорію користувача до вузла з якого користувач здійснює вхід.
* Продуктивність: продуктивність визначається середнім часом для обробки запиту клієнта. Рекомендовано, щоб цей час був подібним до централізованих систем.
* Простота: інтерфейс має бути простим з невеликою кількістю команд.
* Висока доступність: система повинна працювати в разі часткових помилок, як помилка з'єднання або помилка вузла. Високоадаптивні розподілені файлові системи повинні мати різні і незалежні файлові сервера для контролю різних і незалежних пристроїв зберігання.

## 1.2. Еволюція розподілених файлових систем

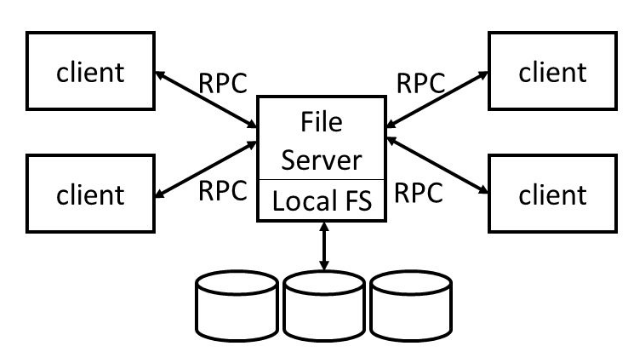
Причиною популярності розподілених файлових систем стала простота інформаційного обміну між багатьма користувачами. Також їхніми основними особливостями мають бути: узгодженість даних, рівномірність доступу, надійність, ефективність, продуктивність, керованість, доступність та безпека [[4]](#q41gwbaymqjv).

Існує багато розподілених файлових систем, деякі з них є CephFS, LustreFS, GlusterFS, XtreemFS, Plane 9, GPFS, GFS, та MooseFS.

### 1.2.1. Network File System — NFS

Розроблена в 1984р. компанією Sun Microsystems. Це одна з найперших розподілених файлових систем, яку широко застосовували. Сьогодні нею керує Internet Engineering Task Force. Остання версія NFSv4.2 була затверджена в листопаді 2016 і вимагає використання TCP протоколу, коли 2 і 3 версії дозволяли UDP протокол [[5]](#7gm69zuaucq7).

Архітектура NFS досить проста, проте це також накладає деякі обмеження. NFS була розроблена для прямого доступу до єдиного місця збереження даних на віддаленій машині. Також, для збільшення продуктивності, система зберігає кеш даних та метаданих на машині клієнта, це дозволяє зменшити час для наступних запитів [[6]](#bjtq1exfrkj0).

*Рис.1.2 Архітектура NFS*

Обмеженнями такої системи є брак надійності, оскільки усі файли знаходять на одній машині, помилка на цій машині приведе до недоступності файлів для користувачів. Іншим обмеженням є те, що NFS легко перевантажується, при доступі великої кількості користувачів [[4]](#q41gwbaymqjv).

Перевагою NFS є низька вартість встановлення, яку легко встановити, адже вона використовує наявну IP інфраструктуру. Також значною перевагою є можливість центрального керування, зменшуючи потребу в додатковому програмному забезпечені на системі користувача. Система є прозорою, що дає змогу клієнтам отримати доступ наче до локального жорсткого диску [[5]](#7gm69zuaucq7).

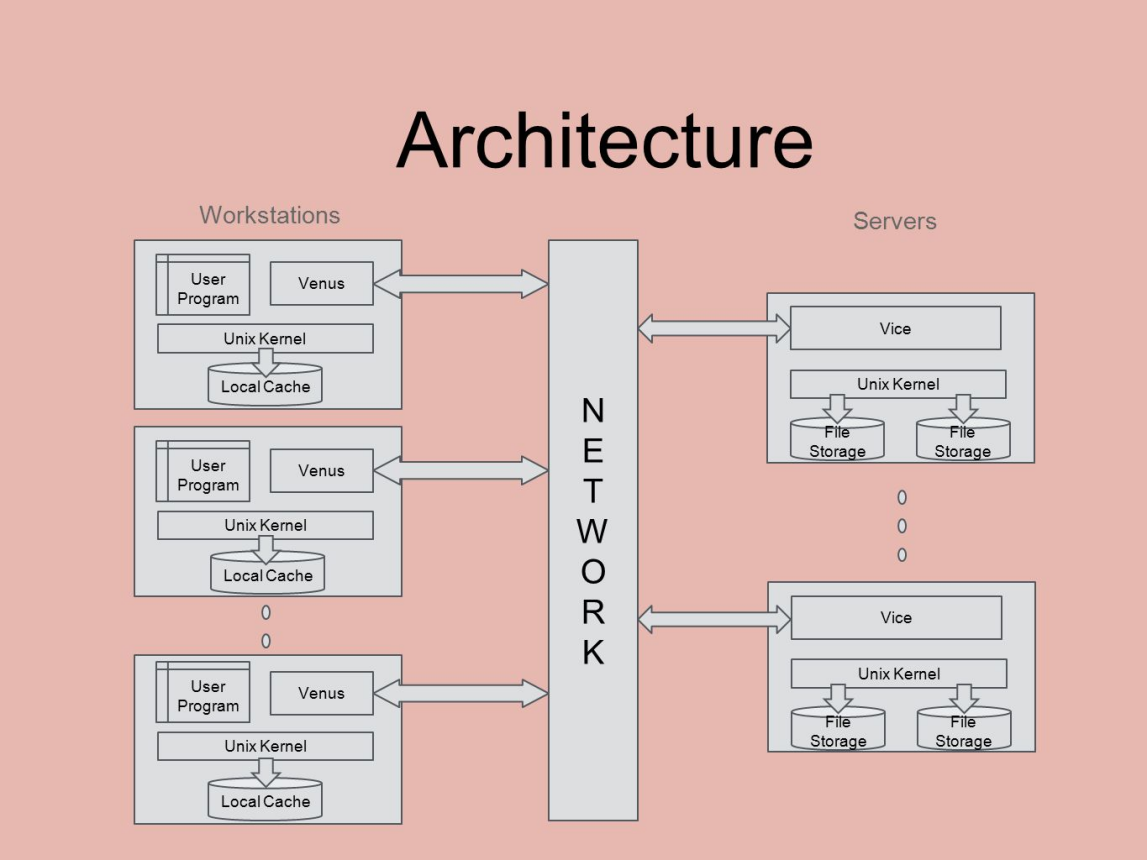
Отже, NFS призначена для невеликого масштабу, де є невелика кількість користувачів, яка одночасно звертається до серверу.

### 1.2.2. Andrew file System — AFS

Вперше представлена в 1980 роках університетом Карнегі Меллон для вирішення проблеми масштабованості серед розподілених файлових систем [[4]](#q41gwbaymqjv).

В свій час AFS була досить популярною системою. Вона використовувалася близько 20,000 клієнтами в 10 країнах світу. Оцінюється, що вона мала більше, ніж 100,000 користувачів [[7]](#3fi87hgjgozx).

В початковій версії AFS, цілий файл кешувався на пристрої користувача, щоб збільшити продуктивність для запитів для одного файлу на сервері. Що дозволило знизити завантаженість сервера, який містить цей файл. Вкінці, якщо файл був модифікований на локальній системі, остання версія надсилалася на сервер під час закриття сесії. Проте, перший доступ до не кешованого файлу був неефективним [[4]](#q41gwbaymqjv).

*Рис.1.3 Архітектура AFS*

Обмеженнями даної системи є [[4]](#q41gwbaymqjv):

1. Високий час проходження шляху: щоб, отримати доступ до файлу на стороні клієнта, сервер повинен пройти повний шлях з домашньої директорії до місце знаходження файлу. Цей процес займає значні ресурси сервера, які він міг витратити на обробку запитів від клієнтів.
2. Високий трафік: в AFS значна частина трафіку створюється у вигляді повідомлень перевірки чи файл був модифікований чи ні. Цей трафік зменшує ефективність в мережі.

Щоб покращити початкову версію AFS, було запропоновано модифікації для масштабованості та продуктивності [[4]](#q41gwbaymqjv):

1. Використання ідентифікатора файлів: цей ідентифікатор чітко вказує, який файловий сервер зацікавлений, що зменшує завантаженість головного сервера.
2. Використання функцій зворотних викликів: ця проста функція виконується сервером, яка повідомляє про модифікацію файлу до всіх клієнтів, які мають кешовану немодифіковану версію цього файлу. Тому, звичайні повідомлення перевірки більше непотрібні.

Andrew file System мала значний вплив на сучасні середовища хмарних обчислень своєю моделлю зберігання даних в хмарі і доставляння частин за допомогою кешування за вимогою. Це спрощує керування операціями зберігаючи продуктивність та масштабованість для кінцевих користувачів. Всі дані, які відносяться до певного користувача доставляються через мережу за вимогою, зберігаючи синхронізованими всі машини [[8]](#rwz4i6cv8j2d).

*Таблиця 1.1*

**Порівняльний аналіз NFS та AFS**

|  |  |  |
| --- | --- | --- |
| **Критерій** | **NFS** | **AFS** |
| **Простір імен** | Не має простору імен спільного користування,  індивідуальні простори імен для всіх клієнтів. | Глобальний простір імен спільного користування. |
| **Кешування файлів** | Не має локального кешування файлів. | Файли повністю кешуються на локальному диску. |
| **Масштабованість** | Для невеликої кількості користувачів (10-20). | Порівняно з NFS високо масштабована. |
| **Безпека** | ІД користувачів використовується для визначення доступу до файлу. | Протокол kerberos для верифікації. |
| **Резервне копіювання** | Основане на резервному копіюванні UNIX системи. | Має власну систему резервного копіювання. |
| **Впровадження** | Solaris, AIX, FreeBSD | Transarc (IBM), OpenAFS |

### 1.2.3. Google File system — GFS

Хоча наступна версія AFS ввела значні покращення, Google все ж стикалася з проблемою зберігання і обробки великих даних. В 2003 році, вони розробили Google File System, як їхнє власне вирішення проблеми керування великими даними. Це розподілена масштабована файлова система, розроблена для підтримки великих розподілених додатків з великою кількістю даних. Вона була розроблена з багатьма цілями спільними з іншими розподіленими файловими системами. Проте, були також інші фактори, які мотивували розробників, такі як, поламка апаратного забезпечення, обмеження пропускної здатності та тип затримки [[4]](#q41gwbaymqjv).

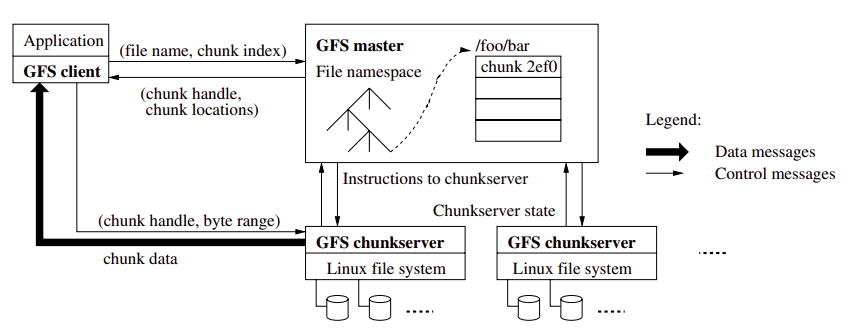
Властивостями GSF є: відмовостійкість, реплікація даних, автоматичне відновлення даних, висока пропускна здатність, зменшена взаємодія між клієнтом і основним сервером, керування простором імен, висока доступність.

Найбільший GFS кластер має більше ніж 1,000 вузлів з 300 терабайт дискового простору, що може бути доступний для сотень клієнтів [[9]](#3aoc84v594s0).

Розробка Google File System спрямовувалася певними припущеннями, які були виведені з попередніх спостережень [[1]](#lsptd9pal5o5):

* Система побудована з багатьох недорогих компонентів, які часто ламаються. Вона повинна постійно та періодично сканувати себе, і виявляти, відновлювати від помилок компонентів.
* Система зберігає скромну кількість великих файлів. Очікується зберігання мільйонів файлів від 100 МБ. Великі файли на кілька гігабайт є звичним випадком, тому повинні оброблятися ефективно. Також, повинна бути підтримка малих файлів, проте не потрібно проводити оптимізацію для них.
* Завантаженість в основному складається з двох видів читань:
  + великі потокові читання: одна операція часто читає сотні кілобайт, частіше мегабайт, або більше. Успішні операції від одного клієнта зазвичай читають близькі регіони файлу.
  + малі випадкові читання: читають кілька кілобайт з довільним зміщенням.
* Завантаженість також має багато послідовних записів, що додають дані до файлів. Типові розміри операцій схожі для читання. Після запису, файл рідко модифікується. Малі записи на довільні позиції мають підтримуватися, але не повинні бути ефективними.
* Система повинна ефективно впроваджувати добре визначену семантику для паралельного додавання до одного й того самого файлу багатьма користувачами.
* Висока пропускна здатність важливіша за низьку затримку.

Як показана на *Рис. 1.4* GFS кластер складається з одного сервера (master) та багатьох серверів даних (chunkservers), які доступні багатьом клієнтам одночасно. Вона працює на Linux машинах, як сервер процес. Досить легко запустити сервери даних і клієнта на одній машині допоки ресурси машини дозволяють. Файли поділені на чанки фіксованих розмірів, кожним з них керує основний сервер. Всі чанки зберігаються на локальних дисках, як файли для запису і читання даних.

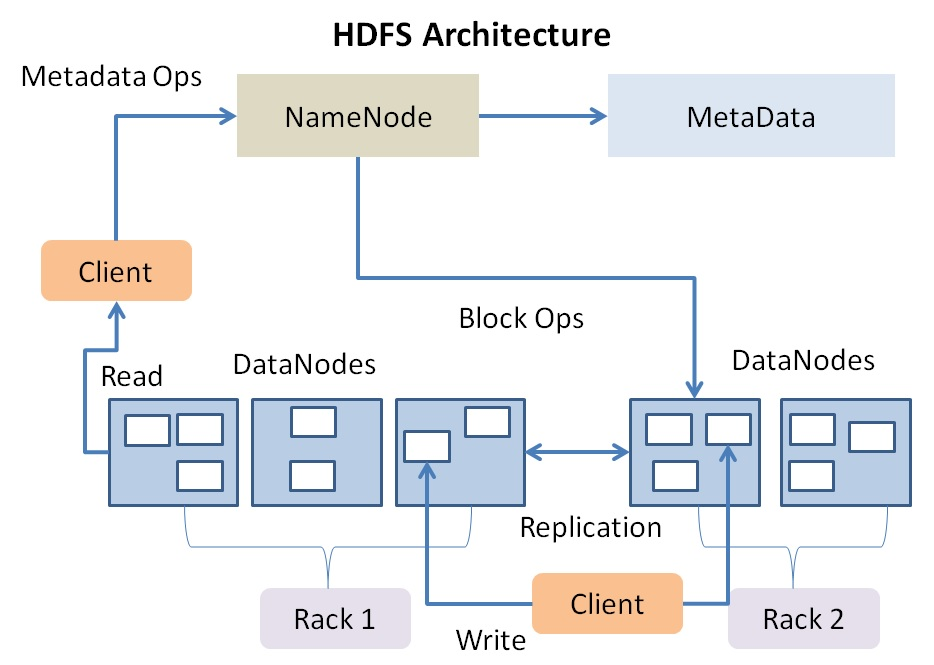
 Для надійності кожний чанк дублюється на кількох серверах даних. Основний сервер зберігає усі метадані файлової системи, що включає простір імен і інформацію про контроль доступу, як словник чанків та їхнього розташування. Основний сервер періодично комунікує з окремими чанками для перевірки їхнього стану, або надати їх інструкції. Для операцій пов’язаних з метаданими клієнт комунікує з основним сервером напряму. Для передачі даних файлів клієнт комунікує з серверами даних напряму, не включаючи основний сервер, що дозволяє зняти з нього завантаження [[4]](#q41gwbaymqjv).

*Рис. 1.4 Архітектура GFS*

### 1.2.4. Hadoop Distributed File System - HDFS

HDFS надихалася можливостями GFS працювати на дешевих компонентах. На відміну від інших розподілених файлових систем, HDFS вважається високонадійною розподіленою файловою системою. Вона була розроблена використовуючи дешеве обладнання, яке може зберігати велику кількість даних і забезпечити простіший доступ до них. Щоб зберігати такі великі чанки даних, файли зберігаються на багатьох машинах. Ці файли зберігаються в надлишковому форматі, щоб зберегти їх від можливої втрати в випадку поламки машини [[4]](#q41gwbaymqjv).

Hadoop є проектом верхнього рівня організації Apache Software Foundation, тому центральним репозиторієм рахується Apache Hadoop, проте її реалізації також є в Cloudera, Hortonworks та MapR [[10]](#thf1jq1k7g3o).

*Рис. 1.5 Архітектура HDFS*

Hadoop Distributed File System має master-slave архітектуру NameNode-DataNode, подібну до архітектури GFS. Кластер складається з одного NameNode, основного серверу, який керує простір імен файлової системи і регулює доступ до файлів, який відбуваються клієнтом. Також в кластері є кілька DataNode, зазвичай один на вузол.

Отже, HDFS показує високу масштабовану і толерантну до помилок файлову систему, яка дозволяє ефективно зберігати дані в файлах. Внутрішньо, файл розподілений на багато блоків, які зберігаються на кількох DataNodes, щоб забезпечити живучість даних в разі помилок. NameNode відповідальний за виконання операції простору імен файлової системи, таких як, відкриття, перейменування, закриття файлів чи директорій. Розподілення блоків до серверів даних також виконується головним сервером. Операції клієнта читання або запису виконуються серверами даних. Сервери даних також допомагають в створені, перейменуванні, або видалені блоків, як інструкції головного серверу [[4]](#q41gwbaymqjv).

*Таблиця 1.2*

**Порівняльний аналіз GFS та HDFS**

|  |  |  |
| --- | --- | --- |
| Критерій оцінки | GFS | HDFS |
| Апаратне забезпечення | Дешеві компоненти з Linux системою, які мають високий шанс поламки | Дешеві компоненти з великою місткістю |
| Комунікація | TCP комунікація  Періодичні командні повідомлення для синхронізації | RPC поверх TCP/IP  Періодичні командні повідомлення для синхронізації |
| Ієрархія | MasterNode і ChunkNodes | NameNode і DataNodes |
| Безпека | Google має багато серверів даних в невідомих місцях, щоб збільшити надмірність | Безпека основана на POSIX моделі користувачів і груп |
| Розмір блоку | 64 мегабайти | 128 мегабайт |
| Метадані для блоку | Кожен чанк має 64КБ даних і 32 біти чек суми | Для кожного чанку створено два файли:  1. Файл даних  2. Файл метаданих |
| Фактор реплікації | 3 рази, але може бути встановлена будь-яка кількість | 2 або 3 рази |
| Хто використовує | Google Inc. | Yahoo!, Facebook, IBM |
| Операції | Можливі випадкові записи, або читання  Модель багатьох читань і записів | Можливе тільки додавання до файлу  Багато читань, одинарний запис |

### 1.2.5. Обмеження і покращення HDFS і GFS

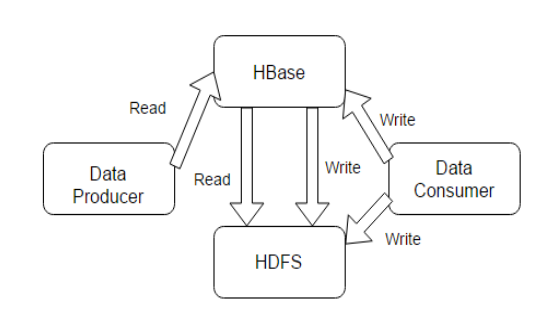
Google File System задумана зберігати великі чистина даних в надійний спосіб і хоча вона підтримує глобальний простір імен, ефективну обробку, знімки простору імен і одинарний запис, система має певні обмеження:

* Випадкові записи на малому рівні: основним завантаженням GFS є великі потокові записи. Забравши певний функціонал з GFS, ми можемо отримати більше переваг для випадкових записів в малі файли.
* Один розмір чанків: деколи єдиний розмір чанку в 64 мегабайти може створювати проблеми. Наприклад, коли останній чанк поділений на кілька частин для пошуку нового місця зберігання. Це також призводить до перевикористання комунікації.

Обидві GFS і HDFS не призначені для [[4]](#q41gwbaymqjv):

1. Доступу до даних з низькою затримкою: оскільки дані зберігаються великими частинами обидві розподілені файлові системи відстають в провадженні швидшого доступу до файлів. Обидві створені для передачі великих частин даних для більшої пропускної здатності.
2. Обробка файлів невеликого розміру: обидві розподілені файлові системи не призначені для обробки файлів менше 1 мегабайта. Хоча вони підтримують операції з малими файлами це призводить до втрати ефективності.
3. Випадки частої зміни даних: розподілені файлові системи добре справляються з великими одиничними записами, проте стикаються з проблемами при частій зміні даних.

Для вирішення проблеми низької затримки доступу до даних можна використати HBase. HBase — це розподілена, відсортована карта за зразком великої таблиці Google для швидших запитів в великих кількостях структурованих даних. Це база даних з відкритим кодом і масштабованістю на основі Hadoop. HBase працює на HDFS і написана на Java. HDFS файли індексуються в HBase для швидких запитів. Головною метою для HDFS є групова обробка файлів з великим розміром. Читання з низькою затримкою не є пріоритетом HDFS. Використовуючи HBase на HDFS ми можемо отримати низьку затримку доступу до малих файлів або таблиць з великих чанків даних [[4]](#q41gwbaymqjv).



*Рис. 1.6 HBase з Hadoop*

Серед переваг Apache HBase можна віднести наступні пункти [[11]](#bdkh9kloyrt6):

* Специфічна модель даних, яка добре підходить для розріджених наборів даних.
* Аналогія з реляційними СКБД в плані індексу початкового ключа.
* Влаштований механізм часових міток, які добавляються автоматично і не можуть бути змінені вручну.
* Наявність інструментів розширеності, які дозволяються працювати з даними в HBase, як з реляційними таблицями.
* Висока швидкість роботи за рахунок кешування в пам’яті і обробка даних на стороні сервера через фільтри і підпроцесори.
* Висока доступність і відмовостійкість завдяки файлової системи Hadoop.
* Масштабованість.

Покращення часу відповіді на запит використовуючи кешування. Централізоване керування кешом є механізмом, який значно покращує час відповіді на запит файлової системи [[12]](#8fn5ojoljlyk). Це дозволяє користувачам встановити шлях для кешування в головній пам’яті HDFS, як показана на *Рис. 1.7.* В цьому випадку, NameNode комунікує з DataNodes, що мають потрібні файли на їхніх дисках і надає їм інструкції для кешування відповідних блоків в кеш-пам’яті, щоб покращити ефективність напряму надаючи кешований шлях для такого самого запиту від інших користувачів.

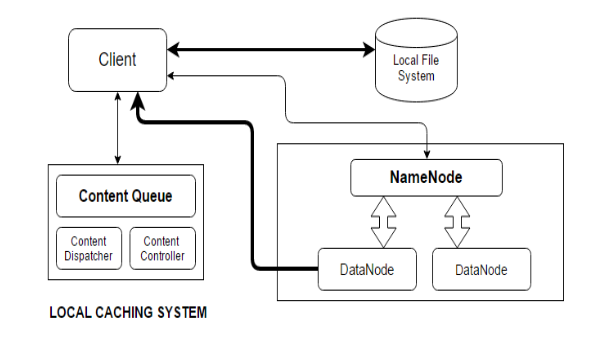
Централізоване керування кешом має багато переваг:

1. Утримання часто використовуваних даних: кешування запобігає вилученню часто використовуваних даних з пам’яті. Це корисно коли розмір робочого сету перевищує доступну основну пам’ять.

2. Ефективні операції читання: збільшення ефективності відбувається за рахунок спільного розташування завдання з кешованою реплікою блоку, керованою NameNode.

3. Використання нового API: використання нового ефективного API стає легшим, оскільки суми верифікації кешованих даних зроблені раніше на DataNodes. Отже, клієнти не отримують додаткового навантаження при використанні нового API.

4. Ефективне керування пам’яттю: кешування покращує використання в пам’яті в кластері в загальному. З централізованим керуванням користувач може вибрати певну кількість реплік, тим самим зберігаючи пам’ять.

*Рис. 1.7 HDFS з покращеним кешуванням*

## 1.3. Використання розподілених файлових систем

Інтерфейс файлової системи є досить загальним і підходить для широкого спектру програм, проте більшість реалізацій розподілених файлових систем оптимізовані для певного класу програм. Наприклад, Andrew File System оптимізована для користувацьких домашніх директорій, XrootD оптимізована для доступу з високою пропускною здатністю до сетів даних високо-енергетичної фізики [[18]](#v0otdwvxbm8m), Hadoop FIle System розроблена, як шар зберігання для фреймворку MapReduce [[19]](#v0t54aw9djm), CernVM File System оптимізована для розподілених бінарних файлів [[20]](#kz1ccgz3cfol), і Lustre є оптимізованою, як простір для взаємодії програм на суперкомп’ютерах [[21]](#vuuaw662xei2).

З погляду програм є різні рівні інтеграції, яку може запропонувати розподілена файлова система. Більшість файлових систем пропонують бібліотеку з інтерфейсом, що нагадує файлову систему POSIX. Перевагою, є те, що інтерфейс може бути адаптований для різних випадків. Наприклад, Google File System розширює семантику POSIX для атомного додавання, функція, яка особливо корисна для фази злиття робіт в MapReduce. Інтерфейс бібліотека реалізуються ціною прозорості. Програми повинні бути розроблені і компільовані для певної розподіленої файлової системи [[14]](#jsbp78kxli07).

Системи інтерпозиції представляють шар опосередкованості, що прозоро перенаправляє виклики файлової системи програмою до бібліотеки, або зовнішнього процесу. Система Parrot створює пісочницю навколо користувацького процесу і перехоплює його системні виклики [[16]](#i7unr6gcx8dq). Fuse — файлова система на рівні ядра перенаправляє виклики файлової системи до спеціальних процесів користувача [[22]](#yf440dlfi46o). Системи інтерпозиції реалізуються ціною виконання для шарів опосередкованості.

Деякі розподілені файлові системи є реалізованими, як розширення ядра операційної системи (NFS, AFS, Lustre). Це забезпечує краще виконання порівняно до систем інтерпозиції, але розгортання є складним і помилки реалізації зазвичай приводять до крашу ядра операційної системи.

Розподілені файлові системи не повністю дотримуються стандарту файлової системи POSIX. Кожна розподілена файлова система повинна бути протестована з ральними програмами. Функціональність, що часто недостатньо добре підтримується в розподілених файлових системах є блокування файлів, атомне перейменування файлів і директорій, кілька жорстких посилань, і видалення відкритих файлів [[14]](#jsbp78kxli07).

## 1.4. Висновок

Ефективне зберігання даних є важливою частиною дослідження в сфері хмарних обчислень і великих даних. Різке збільшення вимог хмарних послуг, збільшення користувачів та даних зробили неможливим пропонувати таке саму ефективність застосовуючи старі методи. Отже, нам потрібно запропонувати кращі та більш ефективні методи. Наприклад, можливі вирішення можуть привести нас до зберігання меншої кількості метаданих. В основі всі системи зберігання файлів ми можемо назвати файловими системами, вони визначають, як файли називаються, організовані, їхній запис і читання. Одною з найбільш популярних файлових систем є Розподілена Файлова Система через свою гнучкість та масштабованість [[4]](#q41gwbaymqjv).

# РОЗДІЛ 2 (20-25c.) Системний аналіз

Системний аналіз є процесом зібрання і інтерпретації фактів, визначення проблем і декомпозиція системи на її компоненти.

Системний аналіз використовується для вивчення системи, або її частин для визначення їх цілей. Це техніка вирішення проблем, що покращує систему і забезпечує, що всі компоненти системи працюють ефективно [[24]](#2mz5vaid8nik).

Система — це організований набір компонентів, або підсистем, що тісно пов’язані для досягнення спільної мети. Система має різні вхідні дані, які проходять через процеси, щоб отримати певні вихідні дані, разом це виконує бажану мету системи [[25]](#n09dcj7c5tg6).

Система має мати три обмеження [[24]](#2mz5vaid8nik):

* певна структура і поведінка розроблена для досягнення наперед визначеної цілі;
* взаємозв’язок та взаємозалежність серед компонентів систем;
* цілі організації мають вищий пріоритет, ніж цілі її підсистем.

## 2.1. Дерево цілей

Дерево цілей є інструментом для раціонального аналізу всіх необхідних умов і їхніх залежностей для досягнення мети. Це центральний інструмент для процесу логічного мислення.

На самому верху дерева знаходиться мета, причина, або бачення. Тобто чому ця система існує. На наступному рівні знаходиться від 3 до 5 критичних факторів успіху — головні цілі, що є необхідними для досягнення мети. Під кожним критичним фактором успіху знаходиться необхідні умови для їх виконання. Як і у випадку з критичними факторами успіху, які є необхідними для виконання Мети, необхідні умови повинні бути виконаними для виконання критичних факторів.

Дерево цілей побудоване не “логіці необхідності”, тобто такі взаємовідносини читаються, як “щоб отримати/досягти ...(вища ціль) ми мусимо отримати/досягти ... (нижча ціль) ” [[26]](#qbzhae6pbtxq).

Після побудови дерево цілей має три функції:

1. Логічна карта майбутнього стану: оскільки мета може бути досягнутою тільки після виконання всіх необхідних умов, які очевидно ще не є досягнуті на момент побудови дерева цілей, воно дає натяк на майбутній стан.
2. Зображення теперішньої ситуації: зображення різниці між теперішнім станом і майбутнім на дереві цілей.
3. Карта розробки: з позначенням різниці і цілей, і в якому порядку, можна використовувати дерево цілей, як карту розробки.

Дана система будується з метою показати принцип роботи розподілених файлових систем.

Конкретизуємо мету за допомогою дерева цілей.

Основна мета — критерії якості системи.

Проміжні рівні — підаспекти основної мети.

Альтернативні варіанти побудови системи (метод аналітичної ієрархії, морфологічного аналізу ...)

Обрати найкращий за допомогою дерева цілей і множини альтернативних варіантів.

## 2.2. Діаграми DFD для деталізації структури

## 2.3. Діаграми ієрархії процесів DFD

# РОЗДІЛ 3 (20c.) Опис програмних засобів

## 3.1. Обґрунтування вибору засобів розв’язання задачі

## 3.2. Технічні характеристики обраних програмних засобів

# РОЗДІЛ 4 (15c.) Практична реалізація

## 4.1. Опис програми

## 4.2. Інструкція користувача (IEEE STD 1063-2001)

## 4.3. Аналіз контрольного прикладу

# РОЗДІЛ 5 Економічна

# ВИСНОВКИ

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The Google File System [Електронний ресурс] / Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung Google – Режим доступу: <https://storage.googleapis.com/pub-tools-public-publication-data/pdf/035fc972c796d33122033a0614bc94cff1527999.pdf> .

2. What is DFS(Distributed File System)? [Електронний ресурс] / Snigdha Yambadwar - Режим доступу: <https://www.geeksforgeeks.org/what-is-dfsdistributed-file-system/> .

3. Distributed File System (DFS) [Електронний ресурс] / techopedia - Режим доступу: <https://www.techopedia.com/definition/1825/distributed-file-system-dfs> .

4. Evolution and Analysis of Distributed File Systems in Cloud Storage: Analytical Survey [Електронний ресурс] / Dharavath Ramesh, Neeraj Patidar, Gaurav Kumar, Teja Vunnam — Режим доступу: <https://www.researchgate.net/publication/312469756_Evolution_and_Analysis_of_Distributed_File_Systems_in_Cloud_Storage_Analytical_Survey> .

5. Network File System (NFS) [Електронний ресурс] / TechTarget — Режим доступу: <https://searchenterprisedesktop.techtarget.com/definition/Network-File-System> .

6. Network File System (NFS) [Електронний ресурс] / GeekForGeeks — Режим доступу: <https://www.geeksforgeeks.org/network-file-system-nfs/> .

7. Mirjana Spasojevic. An empirical study of a wide-area distributed file system / Mirjana Spasojevic, Mahadev Satyanarayanan - ACM Transactions on Computer SystemsMay 1996.

8. You really should know what the Andrew File System is [Електронний ресурс] / Bob Brown, Network World - <https://www.networkworld.com/article/3195838/you-really-should-know-what-the-andrew-file-system-is.html> .

9. Google File System (GFS) [Електронний ресурс] / techopedia - Режим доступу: <https://www.techopedia.com/definition/26906/google-file-system-gfs> .

10. Hadoop: что, где и зачем [Електронний ресурс] / Хабр, @ffriend – Режим доступу: <https://habr.com/ru/post/240405/> .

11. 7 основных преимуществ и пара недостатков Apache HBase для Big Data систем [Електронний ресурс] / Анна Вичугова, Big Data School - Режим доступу: <https://www.bigdataschool.ru/blog/hbase-use-cases.html> .

12. Centralized Cache Management in HDFS [Електронний ресурс] / The Apache Software Foundation — Режим доступу: <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/CentralizedCacheManagement.html> .

13.

14. J Blomer. A Survey on Distributed File System Technology / J Blomer — CERN 2015.

15.

16. Thain D. and Livny M. Scalable Computing: Practice and Experience, 2005.

17. A Full Introduction to DFS (Distributed File System) [Електронний ресурс] / MiniTool — Режим доступу: <https://www.minitool.com/lib/distributed-file-system.html> .

18. WSEAS Transactions on Computers 4 / [Dorigo A, Elmer P, Furano F, Hanushevsky A], 2005. - С. 348–353.

19. Proc. of the 26th IEEE Sympoisum on Mass Storage and Technologies (MSST’10) / [Shvachko K, Kuang H, Radia S, Chansler R], 2010. - С. 1-10.

20. Journal of Physics: Conference Series / [Blomer J, Aguado-Sanchez C, Buncic P, Harutyunyan A], 2011.

21. Schwan P. Proc. of the 2003 Linux Symposium / 2003. - С. 380–386.

22. Filesystem in Userspace (FUSE) [Електронний ресурс] / Henk C., Szeredi M. — Режим доступу: [http://fuse.sourceforge.net](http://fuse.sourceforge.net/) .

23.

24. System Analysis and Design — Overview [Електронний ресурс] / Режим доступу: <https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_overview.htm> .

25. Catter McNamara. Field Guide to Consulting and Organizational Development / Catter McNamara // MBA — 140с.

26. What is a Goal Tree? [Електронний ресурс] / Chris Hohmann — Режим доступу: <https://hohmannchris.wordpress.com/2014/03/07/what-is-a-goal-tree/> .

27.

28.

29.

30.

# ДОДАТКИ