# **Визначення відстані Хеммінга в розподілених інфраструктурах для пошуку подібності.**

**Аннотація**. Сьогодні обсяги даних доступних в Інтернеті перевищують зетабайти (ZB). Ця умова визначає явище яке відоме в літературі як Big Data. Хоча традиційні бази даних дуже ефективні для пошуку та отримання потрібного контенту, вони не є ефективними в умовах роботи з великими обсягами даних, оскільки велика частина цих даних є неструктурованою та розподіленою в Інтернет. Таким чином для забезпечення пошуку подібності необхідні нові бази даних. Для того аби справлятися з даною ситуацією в даному розділі пропонується досліджувати схожість за Хеммінгом між ідентифікаторами контенту, що згенеровані з використанням функції випадкової гіперплощини хешування. Такі ідентифікатори забезпечують основу для побудови розподілених інфраструктур, що спрощує пошук подібності. В цьому розділі м представимо два різні підходи: P2P (Peer-to-Peer) (Hamming DHT (Distributed Hash Table)) та Data Center (HCube). Представлені оцінки що вказують на те, що обидва підходи здатні покращити результати пошуку подібності.

## **1. Вступ**

Згідно сучасного розуміння явища Big Data, користувачі стали джерелами даних, компанії зберігають незліченні обсяги інформації про клієнтів, мільйони сенсорів отримують інформацію з середовища, створюючи та обмінюючись даними в Інтернеті речей. Згідно дослідження International Data Corporation (IDC), опублікованого в травні 2010 [1], кількість даних доступних в Інтернеті перевищує 2 ZB в 2010, подвоююись кожні 2 роки і може перевищити 8 ZB в 2015. Дослідження також показало що близько 90% цих даних складаються з неструктурованих, гетерогенних та різних за природою даних як тексти, зображення та відео.

Нові технології такі як Hadoop [2] та MapReduce [3] є прикладами рішень створених щоб вирішувати проблеми які накладає Big Data – обсяг, різноманітність та швидкість. Використовуючи пераленьні обчислення в поєднанні з грід технологіями, чи, як нещодавно, використовуючи інфраструктуру хмарних обчислень, IT організації пропонують засоби для роботи з широкомасштабними, розподіленими та інформаційномісткими (такими що використовують великі об’єми даних) задачами. Зазвичай такі технології пропонують розподілену файлову систему та автоматизовані інструменти для налаштування в процесі роботи (“на льоту”) кількості серверів залучених в обробку задач. В цьому випадку великі об’єми даних пропускаються через мережу і підключаючись до серверів передають пари <ключ, значення> з маперів до ред'юсорів щоб отримати необхідні результати. В цьому випадку бажано мінімізувати потреби в передачі даних по мережі щоб пришвидшити роботу всієї обчислювальної задачі.

В той час як дані рішення є, беззаперечно, ефективними для роботи традиційних застосувань, як, наприклад, пакетної обробки великих об’ємів даних, вони не забезпечують достатню підтримку пошуку подібності [4], ціллю якої є вилучення наборів подібних даних за даним рівнем подібності. Наприклад, схожість даних може бути використана в системі рекомендацій заснованій на інформації соціальних профілів користувачів. В даному прикладі профіль користувача може бути визначених як набір характеристик що унікально впливають на те як користувачі приймають рішення. Користувачі зі схожими характеристиками, скоріш за все, матимуть схоже інтереси та вподобання.

Таким чином, щоб зробити систему пошуку подібності засновану на даних профілів користувачів, характеристики користувачів в соціальній мережі можуть бути поміщені в вектор і, використовуючи Vector Space Model (VSM), подібність між користувачами може бути визначена використовуючи метрики відстаней між векторами як, наприклад, Евклідова відстань, косинус між векторами та Хеммінгова відстань. Але, за виключенням Хеммінгової відстані, інші метрики піддаються прокляттю розмірності (значний ріст необхідних обчислень при рості розмірності задачі) [4]. Великі обчислювальні витрати внаслідок розмірності задачі – це виклик що постає перед системами поуку подібності в випадку Big Data.

В цьому розділі описано як підтримувати пошук подібності використовуючи Хеммінгову відстань як метрику подібності. Щоб це досягти, дані індексуються в базі даних використовуючи LSH (Locality Sensitive Hashing) функцію що називається функцією RHH (Random Hyperplane Hashing) [5]. RHH – це сімейство LSH функцій, що використовують косинус між векторами та Хеммінгову відстань як метрику відстані між згенерованими двійковими рядками, тобто чим більший косинус між парою векторів контенту, тим менша Хеммінгова відстань між двійковими рядками. Ці двійкові рядки представляють ідентифікатори даних чия подібність можу бути виміряна Хеммінговою відстанню. Кожен запит в цій базі даних вираховується використовуючи Хеммінгову відстань між ідентифікатором запиту та кожним ідентифікатором даних.

При пошуку подібності запит що обробляється складається з того ж набору характеристик контенту індексованого в базі даних. Кожен користувач системи пошуку подібності вводить бажані характеристики та рівень подібності згідно очікуваного об’єму відповіді на запит. Чим більший рівень подібності тим менша кількість даних отримується внаслідок більшої специфічності (націленості) запиту. Запит індексується в базі даних з використанням RHH функції та подібності за Хеммінгом між запитом та всіма обчисленими ідентифікаторами даних в базі даних. Всі профілі чия подібність за Хеммінгом задовольняє бажаному рівню подібності повертаються у відповідь на запит.

Щоб оцінити систему пошуку подібності були проведені наступні тести: взаємозв’язок подібності з використанням косинусу між векторами контенту та подібності за Хеммінгом їх ідентифікаторів представлені використовуючи чотири рівні подібності (0.7, 0.8, 0.9 та 0.95); частотних розподіл відстані за Хеммінгом між ідентифікаторами контенту відповідно до їх рівня подібності; також надані результати деяких запитів та відповідей на них. В наших експериментах вектори контенту представляють профілі користувачів в Adult Data Set репозиторію UCI [6].

В нашій попередній роботі [7] було представлено рішення з оверлеями для системи пошуку подібності що було створено на вершині DHT (Distributed Hash Table) структури. Головним чином була показана можливість зберігання подібних даних на серверах близько до логічного простору оверлейної мережі використовуючи примітив put(k,v) і що також можливо ефективно відновлювати набір подібних даних використовуючи один get(k,sim) примітив. В іншій попередній роботі [8] був показаний HCube – рішення для дата центру (центру обробки даних) для підтримки пошуку подібності в випадку Big Data, спрямоване на зменшення відстані щоб відновити схожий контент при пошуку подібності. В HCube подібні дані зберігаються на тому ж сервері чи на серверах розташованих близько в дата центрі.

Даний розділ організований наступним чином: частина 2 показує деякі основи технологій що використовуються в Hamming DHT та HCube. Частина 3 містить огляд літератури на пов’язані роботи з пошуку подібності в P2P (Peer-to-Peer) мережі та дата центрі. Частина 4 коротко показує рішення Hamming DHT та HCube. Частина 5 оцінює запропоновану систему пошуку подібності в розподіленому випадку. В частині 6 надані деякі ремарки та огляд майбутньої роботи.

## **2. Короткі теоретичні відомості**

Ця частина показує концепцію VSM (Vector Space Model) – модель представлення даних в вигляді векторів в багатовимірному просторі, функцію RHH – LSH функцію для генерування ідентифікаторів даних що зберігатимуть подібність між векторами контенту, та функцію подібності за Хеммінгом – функцію подібності що використовується для порівняння відстані за Хеммінгом між двійковими ідентифікаторами.

### 2.1 Модель векторів у просторі (VSM (Vector Space Model))

VSM – алгебраїчна модель представлення об’єктів як векторів. В загальному випадку, кожен вимір (кожна координата) цих векторів пов’язаний з характеристикою самого контенту як, наприклад, ключові слова в тексті, гістограма кольорів на зображенні чи атрибути профілю в соціальній мережі.

Набір векторів отриманих з Adult Data Set репозиторію UCI [6], використовується щоб описати процедуру перетворення таких атрибутів в вектор що може бути виміряний та порівняний з іншими векторами використовуючи алгебраїчні операції. В своїй суті цей набір даних містить інформацію про дорослих громадян що живуть в США включаючи наступні атрибути: вік, робочий клас, рівень освіти, кількість років проведених в школі, сімейний стан, вид діяльності, родинні зв’язки, расу, стать, фінансові здобутки та втрати за останній рік, кількість робочих годин на тиждень, країна походження, дохід за рік. Зразки цих профілів:

– ADULT1 - 43; Self-emp-not-inc; 5th-6th; 3;Married-civ-spouse;Craft-repair;Husband; White; Male; 0; 4700; 20; United-States; ≤50K

– ADULT2 - 56; Private; 10th; 6; Married-civ-spouse; Craft-repair; Husband;White; Male; 0; 0; 0.45; France; ≤50K

– ADULT3 - 50; Self-emp-inc; Prof-school; 15; Married-civ-spouse;Prof-specialty; Husband;White; Male; 0; 0; 36; United-States; ≥50K

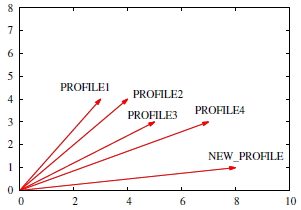
– ADULT4 - 30; Private; Prof-school; 15; Married-civ-spouse; Prof-specialty; Husband; White; Male; 0; 0; 30; United-States; ≥50K

Для експериментів повинні бути зроблені деякі перетворення над цими векторами. Числові атрибути представлені в векторах, такі як “вік”, повинні бути нормалізовані до границь [0..1]. Така нормалізація була зроблена шляхом ділення значення на найбільше значення в наборі. Координати що представляють дискретні атрибути (наприклад, стать, що може бути “чоловіча” чи “жіноча”) були розділені в 2 різні координати, кожна з яких в окремому вимірі відповідно до можливих значень. Для атрибуту “стать” було створено два виміри “чоловіча” та “жіноча”. Якщо людини – чоловік, відповідний вектор має значення “1” для виміру “чоловіча” та “0” для виміру “жіноча” і навпаки, якщо людина – жінка.

Як показано в [9] ці процедури були необхідні тому що поняття подібності чи відстані для дискретних даних не таке очевидні як для числових і тут це було основною задачею.Це виникає внаслідок того що різні значення дискретного атрибуту не упорядковані по свій суті і, таким чином, поняття впорядкованості для них не визначене. Також поняття подібності може відрізнятися в залежності від конкретного домену. Внаслідок цього кожен атрибут вектору має бути розширений на кількість вимірів відповідно до всіх значень які він [атрибут] містить. Використовуючи цю процедуру, вектор повинен бути розширений з 14 до 103 вимірів.

Можна виміряти подібність між векторами що представляють дані шляхом вирахування косинусу кута між ними (simcos). Як показано в [10], подібність за косинусом дає високоякісні результати в декількох доменах. Щоб показати це, на мал. 1 відображено застосування в якому профілі користувачів представлені двовимірними векторами, кожен вимір описує інтерес користувача до спорту та літератури, використовуючи масштаб в якому “0” означає відсутність інтересу, а “10” – повний інтерес до області. Розглянемо чотири профілі користувачів представлені кортежами PROFILE1(3,4) – ранг 3 для спорту та 4 для літератури, PROFILE2(4,4) – ранг 4 для спорту і літератури, PROFILE3(5,3) – ранг 5 для спорту та 3 для літератури та PROFILE4(7, 3) – ранг 7 для спорту та 3 для літератури.

Щоб дати розуміння використання подібності за косинусом (simcos), розглянемо розробку системи рекомендації друзів засновану на профілях користувачів відповідальну за вказування на друзів з подібними інтересами до профілю нового користувача представленого в вигляді кортежу NEW PROFILE(8,1) – ранг 8 для спорту та 1 для літератури. В цьому випадку система рекомендації запропонує користувачу NEW PROFILE наступний порядок для встановлення нових знайомств: 1) PROFILE4 чий simcos ≈ 0.96, 2) PROFILE2 чий simcos ≈ 0.9, 3) PROFILE3 чий simcos ≈ 0.8, and 4) PROFILE1 чий simcos ≈ 0.7.



Спорт

Мал. 1 Графічне представлення векторів профілей

### 2.2 RHH (Random Hyperplane Hashing) та подібність за Хеммінгом

Функції LSH зменшують розмірність векторів що представляють дані забезпечуючи те, що чим більш подібними два об’єкти є, тим більш подібними будуть хеш-значення їх векторів [4]. Кожне сімейство LSH функцій відноситься до певної функції подібності. RHH – приклад сімейства LSH функцій пов’язаних з подібністю за косинусом. В даному контексті, Чарікар [5] представляє техніку хешування резюмовану в даній частині.

Дано набір  з m векторів  кожна з координат яких випадковим чином отримана з стандартного нормального розподілу, та вектор . Хеш-функція  визначається наступним чином:



Для кожного  генерується один біт, а результат m об’єднується щоб створити m-бітний хеш-ключ для даного вектора . Для двох векторів даних  ймовірність згенерувати подібні ключі – значення косинусу кута між  та . Відповідно, чим більша подібність за косинусом, тим більш ймовірно що згенеровані ключі міститимуть спільні біти, приводячи схожість двох ідентифікаторів в відстані за Хеммінгом (тобто в кількості різних біт в двох двійкових рядках). І навпаки – подібність за Хеммінгом можна вирахувати як , де simh – подібність за Хеммінгом, Dh – Хеммінгова відстань, m – число біт в рядку що представляє ідентифікатор профілю користувача.

Наприклад, нехай застосування використовує 8-бітні ідентифікатори, повинна бути згенерована послідовність m=8 випадкових векторів  і повернені m  об'єднаних бітів щоб згенерувати 8-бітний ідентифікатор. В таблиці 1 показано приклад з порівнянням подібності за косинусом (simcos), відстані за Хеммінгом (Dh) та подібністю за Хеммінгом (simh) між 4 типовими 8-бітними ідентифікаторами профілів користувачів. В цьому прикладі “PROFILE B”, що найбільш подібний до “PROFILE A” на основі подібності за косинусом також має найменшу відстань за Хеммінгом (Dh) відповідну подібності за Хеммінгом (simh) в 0.875.

Таблиця 1. 8-бітні ідентифікатори профілів, відстань за Хеммінгом (Dh), подібність за Хеммінгом (simh) та подібність за косинусом (simcos) профілів користувачів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Профілі користувачів | 8-бітні ідентифікатори | Dh | simh | simcos |
| PROFILE B,  PROFILE A | 01001010,  01101010 | 1 | 0.875 | 0.99 |
| PROFILE C,  PROFILE A | 01001000,  01101010 | 2 | 0.75 | 0.95 |
| PROFILE D,  PROFILE A | 01011000,  01101010 | 3 | 0.625 | 0.85 |

## **3. Огляд літератури**

Серед пов’язаних праць на тему пошуку подібності доступних в літературі, більшість засновані на певному виді схем індексації, таких як, наприклад, функції хешування [4] чи SFC (Space Filling Curve) [11]. Обидві мотивовані проблемою найближчих сусідів [4], тобто як вилучити найбільш подібні дані в індексованому просторі. Використання властивості подібності за Хеммінгом RHH функції надає перевагу: немає необхідності використовувати криву Гільберта для об’єднання подібних ідентифікаторів.

Як показано в [4], SFC піддаються прокляттю розмірності [4]. Для вирішення проблеми розмірності, Індик [4] запропонував використовувати LSH функції. Ці функції можуть зменшити кількість вимірів вектору, створюючи ідентифікатор для нього, представлений двійковим рядком розміру m (m>=0). Відстань між двома двійковими рядками згенерованими застосуванням LSH функції пари векторів контенту обернено пропорційна їх схожості. Наша пропозиція використовувати відстань за Хемміном для виміру подібності контенту, що відображається на відповідну схожість профілів користувачів, на скільки нам відомо, -- новий підхід в створенні систем пошуку подібності.

В першій реалізації прототипу пошуку подібності було необхідно проводити пошук по всій базі даних. Ми знаємо, що цього недостатньо в випадку Big Data, що мотивувало дану роботу. Для того аби розв’язати це завдання в випадку Big Data, ми пропонуємо дві розподілені архітектури: Hamming DHT [7] та HCube [8], що можуть виступати як інфраструктури для пошуку подібності в контексті Big Data. Hamming DHT – це P2P мережа що використовує подібність за Хеммінгом щоб полегшити пошук та вилучення подібних профілів. HCube – інфраструктура дата центру що спеціалізується на пошуку подібності використовуючи подібність за Хеммінгом.

Оверлейні підходи з’явилися як рішення для допомоги керування великими об’ємами даних. В основному, ці рішення засновані на DHT (Distributed Hash Tables), поширенні даних між вузлами оверлейної мережі. Згідно [12], з’явилися багатовимірні методи індексації P2P як нова парадигма за останні декілька десятиліть. В цьому випадку DHT повинні бути обладнані багатовимірними запитами та можливостями для обробки пошуку подібності.

Hycube [13] – приклад DHT що використовує відстань за Хеммінгом як метрику відстані та організовує вузли в гіперкуб одиничного розміру. Однак витрати пов’язані з підтримкою гіперкуба та неповних кубів більші ніж витрати на послідовне хешування DHT, як в запропонованій Hamming DHT.

pSearch, запропонований Тангом та ін. [14] – P2P мережа що також використовує вектори контенту та подібність за косинусом, але відрізняється від Hamming DHT, оскільки спеціалізується на пошуку подібності текстових документів в P2P мережі. Також в pSearch обговорюється шлях побудови розподіленого словника термінів для індексації документів.

Бхаттачарія [15] використовує подібність за косинусом та LSH функції щоб запропонувати фреймворк для пошуку подібності в розподілених базах даних. Це розширює get(k) примітив, наданий будь-якою реалізацією DHT для підтримки рівня подібності get(k,Dh) на відстані за Хеммінгом Dh від k. Ця пропозиція не націлена на пошук подібності, оскільки це надбудова над будь-якою існуючою DHT для підтримки пошуку подібності.

З цього короткого огляду ми можемо виділити що Hamming DHT зосереджена на зменшенні відстані (в стрибках) необхідної для вилучення подібного контенту і на підвищенні ефективності пошуку в системах пошуку подібності . Використання LSH функцій для індексації подібного контенту – не нове, але дослідження подібності за Хеммінгом в організації ідентифікаторів контенту – оригінальна ідея. Ця модифікація робить Hamming DHT простішим за інші рішення.

Порівнюючи ці дві роботи, HCube – розподілене рішення що використовує такі методи як LSH функції та SFC для підтримки пошуку подібності, але HCube не засноване на оверлейному рішенні. HCube представляє серверно-орієнтовану структуру дата центрів що спеціалізується на пошуку подібності, де подібні дані зберігаються на фізично близьких серверах, з вилученням цих даних за зменшену кількість стрибків і з меншими вимогами до обробки.

З іншого боку, такі гравці як Google [16] та Amazon [17] створили дата центри що спеціалізуються на зберіганні та обробці великих об’ємів даних використовуючи рішення MapReduce, але вони не розглядаються для пошуку подібності. Таким чино HCube відкриває нове поде для досліджень де застосування можуть скористатися своєю структурою для пошуку подібності, уникаючи задач інтенсивної обробки як з’являються в MapReduce, оскільки подібні дані організовані на серверах що близько розташовані протягом фази сортування.

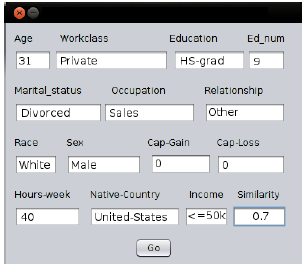
В наступній частині показано як здійснювати пошук в розподілених інфраструктурах заснованих на відстані за Хеммінгом, розглядаючи схожість як результат.

## **4. Пошук подібності заснований на відстані за Хеммінгом**

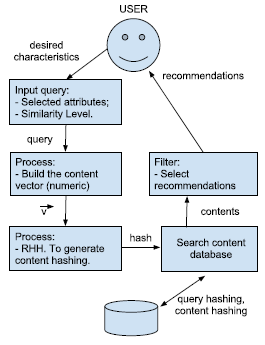
В запропонованій системі пошуку подібності, користувач що створює запит повинен обрати бажані характеристики та рівень подібності в інтервалі [0..1] для запиту. Потім відбувається пошук в контенті бази даних на предмет записів що задовольняють обраним характеристикам. В першій реалізації дозволені тільки повні запити з заповненими всіма атрибутами.

На мал. 2 показано приклад головного інтерфейсу що використовується для вибору бажаних характеристик для пошуку. В цьому ж інтерфейсі користувач вводить рівень подібності. Як було вказано раніше, чим більший рівень подібності, тим менша кількість отриманих результатів, тому що менше профілів задовольняють запиту. Результати організовані в список як результат пошуку подібності.

Мал. 3 резюмує операції прототипу. Користувач запропонованої системи пошуку подібності обирає всі бажані характеристики для запиту і бажаний рівень подібності. Після вибору бажаних характеристик, створюється вектор контенту за процедурою описаною в 2.1. Цей вектор  використовується як вхідні дані для RHH функції що генерує ідентифікатор хешу запиту. Хеммінгова відстань між  та всіма записами бази даних вираховується і всі записи які задовольняють обраному рівню подібності виводяться користувачеві. Може бути застосований фільтр другого рівня для, наприклад, покращення результатів чи вибору найновіших з них, але цей фільтр не представлений в даній реалізації.



Мал. 2. Прототип інтерфейсу для запиту



Мал. 3. Операції прототипу

Деякі приклади результатів, отримані використовуючи набір даних описаний в частині 2.1, представлені та проаналізовані. Прототип інтерфейсу використовувався для отримання результатів всіх подібних профілів в базі даних згідно обраного рівню подібності. Для представлення результатів було використано чотири рівня подібності 0.7, 0.8, 0.9 та 0.95.

В таблиці 2 представлені деякі результати. В усіх випадках вхідними даними для пошуку був профіль з наступними характеристиками: “31” рік; працює в “Private” секторі; закінчив школу, “HS-grad”; “9” років провів у школі; “Divorced”; працює в відділі “Sales”; родинні зв’язки – “Other-relative”; “White”; “Male”; народився в “United-States”; дохід на рік “<=50K” американських доларів. Був використаний рівень подібності 0.7, що означає що були отримані результати з подібність 0.7 і більше.

Таблиця 2. Деякі результати прототипу та їх подібність за Хеммінгом (simh) з використанням профілю користувача q як вхідні дані. q: <31; Private; HS-grad; 9; Divorced; Sales; Other-relative; White; Male;

United-States; <=50K>.

|  |  |
| --- | --- |
| R1, simh = 0.75 | <41; Private; HS-grad; 9; Married-civ-spouse; Adm-clerical; Wife; White; Female; United-States; >50K> |
| R2, simh = 0.7266 | <46; Private; Some-college; 10; Married-civ-spouse; Sales; Husband; White; Male; United-States; >50K> |
| R3, simh = 0.7734 | <36; Private; Some-college; 10; Married-civ-spouse; Craft-repair; Husband; White; Male; United-States; <= 50K> |
| R4, simh = 0.8359 | <22; Private; HS-grad; 9; Never-married; Other-service; Other-relative; White; Male; United-States; <=50K> |
| R5, simh = 0.8516 | <53; Private; HS-grad; 9; Married-civ-spouse; Craft-repair; Husband; White; Male; United-States; <=50K> |
| R6, simh = 0.8438 | <37; Private; HS-grad; 9; Separated; Handlers-cleaners; Not-in-family; White; Male; United-States; <=50K> |
| R7, simh = 0.9062 | ¡48; Private; Assoc-adm; 12; Never-married; Craft-repair; Not-in-family; White; Male; United-States; <=50K> |
| R8, simh = 0.9219 | ¡62; Private; HS-grad; 9; Never-married; Craft-repair; Not-in-family; White; Male; United-States; <=50K> |
| R9, simh = 0.9141 | <28; Private; HS-grad; 9; Never-married; Craft-repair; Other-relative; White; Male; United-States; <=50K> |
| R10, simh = 0.9766 | <38; Private; HS-grad; 9; Divorced; Sales; Not-in-family; White; Male; United-States; <=50K> |
| R11, simh = 0.9688 | <70; Private; HS-grad; 9; Never-married; Craft-repair; Other-relative; White; Male; United-States; <=50K> |
| R12, simh = 0.9844 | <33; Private; HS-grad; 9; Divorced; Sales; Not-in-family; White; Male; United-States; <=50K> |

Не всі результати представлені в таблиці 2, але було 48842 профілів в базі даних. В наступних тестах було зроблено 48842 запити над тими ж 48842 профілями що привели до 2385540964 результатів для оцінки при вибраному рівню подібності. Загальна кількість результатів змінна і залежить від розміру бази даних та рівня подібності. Чим більший рівень подібності тим менша загальна кількість вилучених профілів.

В наступних підрозділах ми представимо Hamming DHT та the HCube – два різні шляхи розподілу запропонованої системи пошуку подібності.

### 4.1 Hamming DHT

Hamming DHT переймає від Chord [18] підхід послідовного хешування та процедури приєднання та залишення, але пропонує дві нові властивості націлені на отримання максимальної переваги від запропонованого механізму генерації подібних ідентифікаторів для подібного контенту. Коротко, цими двома новими властивостями є: 1) використання кодів Грея для організації ідентифікаторів в кільце; 2) встановлення вказівників на основі відстані за Хеммінгом ідентифікаторів вузлів.

Ця частина представляє аспекти зберігання та вилучення в запропонованій системі, де механізм класифікації контенту створений використовуючи RHH функцію, що відповідає наступним властивостям:

– ∀ c1,c2 ∈ C : simcos(c1,c2) → [0..1], де c1 та c2 – вектори контенту в просторі векторів контенту C.

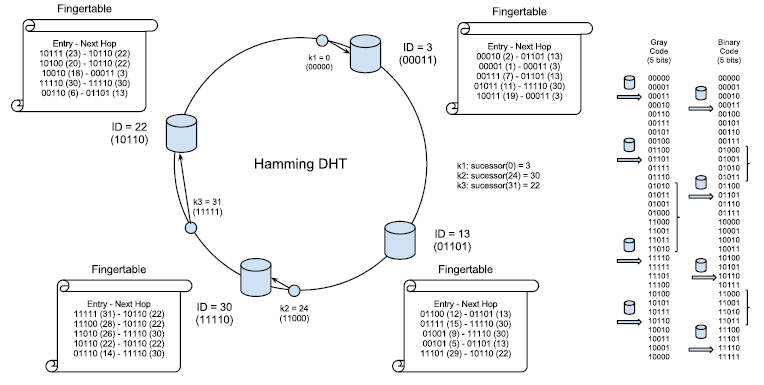
– ∀ c1,c2 ∈ C : Dh(RHH(c1),RHH(c2)) ∝ 1/simcos(c1,c2).

По суті, я показано в [19], властивості характерні для RHH функцій можуть представляти, з високою точністю, подібність між елементами контенту виміряну в подібності за Хеммінгом. Такі характеристики в поєднанні з організацією ідентифікаторів за кодами Грея в Hamming DHT та встановлення вказівників заснованому на відстані за Хеммінгом між вузлами, забезпечують ефективну систему пошуку подібності зменшуючи відстань (в стрибках) між вузлами що зберігають подібний контент.

#### 4.1.1. Побудова кільця

Всі вузли створюють віртуальне кільце HammingDHT та мають m-бітний ідентифікатор вузла. Щоб отримати такий m-бітний ідентифікатор вузли можуть використовувати будь-яку базову хеш-функцію як MD5 чи SHA-1, вирахувану над IP-адресою вузла та/або його приватним ключем, забезпечуючи унікальність ідентифікаторів.

Далі, після отримання своїх ідентифікаторів, вузли об’єднуються в кільце що організоване згідно послідовності коду Грея, на відміну від інших DHT як, наприклад, Chord, де кільце організоване півмісяцем в натуральному порядку ідентифікаторів. На мал. 4 показаний приклад запропонованої Hamming DHT з m=5. Як можна бачити на малюнку є чотири вузли (3 – 000112, 13 – 011012, 30 – 111102 та 22 – 101102) та три елементи (0 – 000002, 24 – 110002 та 31 – 111112). З правого боку мал. 4 можна бачити різницю між послідовністю коду Грея та серповидним натуральним порядком ідентифікаторів. З послідовності коду Грея маємо: 3 < 13 < 30 < 22.



Мал. 4. Приклад кільця Hamming DHT з m=5

#### 4.1.2. Послідовне хешування

Щоб зберігати контент в Hamming DHT, ключ k контенту призначається першому вузлу чий ідентифікатор більший чи рівний k згідно послідовності коду Грея. Цей вузол називається наступником ключа k і позначається successor(k). Коротко, successor(k) – це перший вузол за годинниковою стрілкою від k в m-бітному кільці коду Грея. Наприклад, вузол 3 відповідальний за зберігання контенту k=0, вузол 30 – за зберігання контенту k=24 та вузол 22 за – k=31.

Порівнюючи послідовність коду Грея та натуральний двійковий код (також показаний з правого боку на мал. 4) можна побачити деякі переваги послідовності коду Грея для об’єднання схожого контенту. Наприклад, всі елементи контенту з ідентифікаторами \*10\*\* послідовно розташовуються та зберігаються в вузлі 30 (11110), в той час як при натуральному двійковому порядку елементи зберігатимуться в двох різних місцях – вузлах 13 (01101) та 30 (11110).

#### 4.1.3. Встановлення вказівників

Коли кільце організоване, можна зберігати та вилучати інформацію на/з DHT якщо кожен вузол з’єднаний зі своїм наступником в кільці. На основі цих кругових взаємозв’язків дія зберігання чи вилучення даного ключа k вимагає маршрутизацію повідомлень по кільцю Грея через групу вузлів-наступників поки не буде знайдений вузол відповідальний за цей ключ контенту (successor(k)).

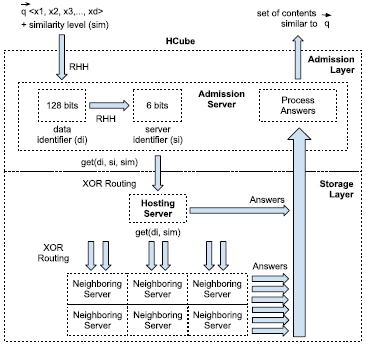
Однак для отримання кращої ефективності маршрутизації кожен вузол може містити таблицю маршрутизації, котру також називають таблицею вказівників, з (не більше ніж) m записами. Коли необхідний запис вказівника не вказує на вузол в DHT, він [запис] пов’язується з наступником.

i-ий запис (fi) в таблиці вказівників вузла p відповідає ідентифікатору отриманому заміною i-го біту ідентифікатору вузла. Запис (fi) пов’язується з вузлом successor(fi) в кільці коду Грея, тобто fi=(p⊕2i−1)→sucessor( fi), 1≤ i≤ m. Таблиця вказівників також містить локатор (наприклад, IP-адресу виділену вузлу) sucessor( fi). Наприклад, на мал. 4 показані таблиці вказівників для вузлів 3, 13, 30 та 22. В цьому прикладі таблиця вказівників вузла 13 вказує на наступника кожного з його m записів: 12, що вказує на 13; 15, що вказує на 30; 9 – на 30; 5 -- на 13 та 29 – на 22. Коли вузол p не має вказівника прямо установленого на successor(k) ключа k, він пересилає повідомлення вузлу  наявному в його таблиці вказівників, чий ідентифікатор передує k в кільці Грея. Цей процес повторюється поки не дійде до successor(k), відповідно до ситуації де число стрибків між вузлами більше ніж один.

Загальний опис архітектури HCube наведений в наступній частині, де показано організацію серверів в структурі дата центру і описано як ідентифікатори серверів назначаються згідно послідовності коду Грея та показано маршрутизацію eXclusive OR (XOR) відповідальну за пересилання трафіку.

### 4.2. HCube

Як можна бачити з мал. 5, логічна організація HCube складається з двох шарів – шару прийому та зберігання.



Мал. 5. Робота H-cube при запиті

Шар прийому забезпечує інтерфейс між зовнішнім світом та структурою HCube. Цей шар складається з набору серверів прийому котрі працюють зверху HCube, отримуючи запити від користувачів/застосувань та готуючи ці запити для введення в HCube для здійснення пошуку подібності. Для простоти, на мал. 5 показаний тільки один сервер прийому на шарі прийому.

В верхньому лівому куті мал. 5 показаний вектор запиту  що складається з d вимірів щ обули надані HCube разом з бажаним рівнем подібності sim. Призначення шару прийому – бути відповідальним за обробку таких запитів згідно, наприклад, географічного регіону звідки даний запит приходить.

Коли вектор запиту  отриманий, шар прийому отримує ідентифікатор даних di згідно процесу описаного в частині 2.2., що на мал. 5 довжиною 128 біт. di – це ідентифікатор даних елементу для процесу пошуку подібності. Далі шар прийому зменшує di використовуючи RHH функцію, що націлено на отримання ідентифікатору si, кортий вказує на хостинг-сервер відповідальний за посилання на дані di. В даному прикладі si – 6-бітний, врезультаті виходить HCube з 64 серверів (4x4x4 сервери).

Після прийому запиту , get створений з посиланням на ідентифікатор диних di, si хостинг-серверу та бажаного рівня подібності sim, видається на шар зберігання. Таке повідомлення направляється до si, використовуючи XOR Routing, описану далі в частині 4.2.3 і з хостинг-серверу si видається група повідомлень get(di,sim) що передається сусіднім серверам також з використанням механізму XOR.

Генерація таких get(di,sim) повідомлень керується деякими факторами, включаючи довідкові дані di і бажаний рівень подібності sim. Спеціально для мети оцінювання, повідомлення get(di,sim) поступово відправлялися на всі сервери відповідно до чергового порядку відстані за Хеммінгом між ідентифікаторами серверів, оскільки ціллю було провести повний аналіз відстані між серверами що зберігають подібні дані та ефективності пошуку де були вилучені подібні дані.

Оскільки get повідомлення передаються всередині HCube, сервери що мають дані з бажаним рівнем подібності sim передають їх до шару прийому, що відповідальний за обробку таких запитів на шарі прийому. Шар прийому підсумовує відповіді та доставляє набір подібних даних до запитуючого користувача/застосування, закінчуючи процес пошуку подібності. Як альтернативна реалізація, шар доступу може список посилань на подібні дані замість повернення всього набору даних. Такий вибір дозволяє уникнути непотрібного переміщення великих об’ємів даних та дозволяє користувачам обирати яку частину даних вони дійсно хочуть отримати, наприклад, лікар може лише декілька останніх діагнозів що відносяться до поточного лікування.

#### 4.2.1. Структура HCube

HCube складється з групи серверів організованої в тривимірну топологію кубу в якій сервери є основними елементами дата центру (сервероцентрична парадигма). За такої сервероцентричної схеми прості COTS (commodity off-the-shelf (готові, збрані елементи)) комутатори можуть бути використані для спрощення процесу проводки в дата центрі чи можуть бути встановлені прямі з’єднання між серверами. Найбільш важлива характеристика сервероцентричної схеми – те, що елементи мережі, такі як комутатори, не беруть участі в прийнятті рішення про передачу трафіку, вони просто забезпечують з’єднання між серверами, котрі відповідальні за прийняття всіх рішень про передачу трафіку.

Кожен сервер HCube містить процесор загального призначення з пам’яттю, постійний запам’ятовуючий пристрій та шість мережевих карт, названих від eth0 до eth5 і три розподілені вісі (x, y та z). Для того щоб бути встановленим в топологію HCube, кожен сервер використовує мережеві карти eth0 та eth1 в осі x, eth2 та eth3 в осі y та eth4 та eth5 в осі z.

Щоб забезпечити більшу кількість альтернативних шляхів в HCube, збільшити стійкість до збоїв та зменшити максимальну відстань між будь-якою парою серверів, прийнята технологія також охоплює всі з’єднання між пограничними серверами в усіх трьох осях. На мал. 6 показано топологію HCube з 27 серверів. На цьому малюнку варто звернути увагу на наявність шести мережевих карт на всіх серверах, зв’язки в трьох осях та огортаючи з’єднання між пограничними серверами.

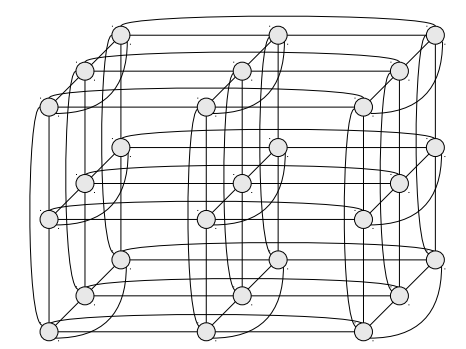
#### 4.2.2 Розподіл ідентифікаторів сервера

Щоб полегшити пошук подібності, куб Хеммінга приймає сірі коди для присвоєння ідентифікаторів всім серверам, що складають його структуру. При використанні сірих кодів [20] два послідовні ідентифікатори серверів відрізняються лише на один біт, тобто сусідні сервери розміщують відстань Хеммінга 1 між їх ідентифікаторами, і вони схильні зберігати подібні дані.

Пропонована організація слідує кривій заповнення сірого простору (SFC), яка забезпечує гарне співвідношення між кластеризацією даних та обчислювальною складністю кривої [11] і спеціалізується на кластеризації даних відповідно до їх відстані Хеммінга. Наприклад, на малюнку 7 показано куб Хеммінга з 64 серверами зберігання (ідентифікатори з довжиною 6 біт), що поширюються в куб Хеммінга з розмірами *4x4x4*. На малюнку представлені чотири шари (*L1, L2, L3 і L4*) куба Хеммінга, які з'єднані відповідно до деталей, наведених у розділі 4.2.1.

В якості прикладу розглянемо сервер 29 (), розташований у шарі L2 на малюнку 7. Шість сусідів цього сервера - 28 (, L2) і 31 (, L2) на осі x; 25 (, L2) і 21 (, L2) на осі y; 13 (, L1) і 61 (, L3) на осі z. Зверніть увагу, що всі ці сусіди представляють ідентифікатори, відстань Хеммінга яких до сервера 29 дорівнює 1.

У ще одному прикладі, де встановлюються зв'язки між крайовими серверами, розглянемо сервер 60 (), розташований у шарі L3 на малюнку 7. Шість сусідів цього сервера - 62 (, L3) і 61 (, L3) на осі x; 52 (, L3) і 56 (, L3) на осі y; 44 (, L4) і 28 (, L2) на осі z. Всі вони також відображають відстань Хеммінга 1 до ідентифікатора сервера 60.



**Рис. 6.** Куб Хеммінга з 27 серверами (*x = 3, y = 3, z = 3*)

**Рис. 7.** Зображення куба Хеммінга з 64 серверами (L1, L2, L3 та L4)

Особливий випадок представлений на рисунку 7: кількість NIC серверів відповідає довжині ідентифікаторів (6 біт). У даному випадку сусіди всіх серверів представляють відстань Хеммінга, що дорівнює 1. Однак для кубів Хеммінга з розміром більше 64 серверів тотожний простір для визначення ідентифікаторів серверів має бути більшим, ніж 6-бітний, тобто більше 6 серверів, відстань Хеммінга яких дорівнює 1. Це означає, що 6 серверів, що мають відстань Хеммінга 1, завжди будуть безпосередньо підключені до даного сервера, а інші сервери, відстань Хеммінга яких 1, будуть розміщені на більшій кількості стрибків.

На рис. 8 зображено куб Хеммінга більше 6 біт. На ньому показано шари *8x4x4* (7-бітний) куб зі 128 серверами. Звернемо увагу на сірий SFC, представлений стрілками, і розглянемо сервер 9 (), який знаходиться, наприклад, на шарі L1. Є 7 серверів з відстанню Хеммінга 1, організовані наступним чином: сервери 8 (, L1) і 11 (, L1) на осі x; сервери 25 (, L1) і 1 (, L1) на осі y; сервери 41 (, L2) і 73 (, L4) на осі z і, нарешті, сервер 13 (, L1), який знаходиться на більшій кількості стрибків від сервера 9. Відмітимо, що відстань у стрибках до сервера 13 зменшується з урахуванням згорнутих зв'язків, встановлених між крайовими серверами куба Хеммінга.

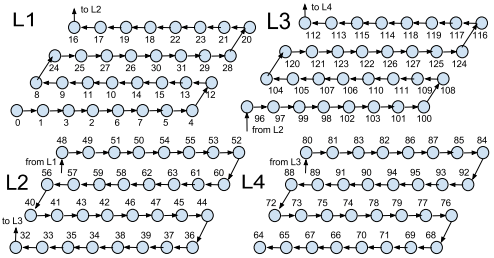


Рис. 8. Сірий код у кубі Хеммінга 8х4х4 (128 серверів)

Повний куб Хеммінга повинен мати n-розмірність, де n - кількість бітів ідентифікаційного простору, що використовується для серверів. Хоч така структура матиме кращі результати в пошуку подібності за допомогою подібності Хеммінга, її важко розгорнути та вона дуже далека від реальних електромонтажних структур центрів обробки даних. Таким чином, тривимірна структура куба Хеммінга пропонує гарний компроміс між складністю куба та відкликом у пошуку подібності.

#### 4.2.3 Маршрутизація XOR

Методика XOR використовує n-бітні плоскі ідентифікатори для організації таблиць маршрутизації в n стовпцях та маршрутизації пакетів через мережу. Його принцип маршрутизації використовує бітову ексклюзивну або (*XOR*) операцію між двома ідентифікаторами сервера a та b, оскільки їх відстань, яка представлена у вигляді d(a,b)=a⊕b, де d(a,a)=0 і d(a,b)>0, ∀a,b. З урахуванням пакета, ініційованого сервером *x* і призначеним для сервера *z*, і позначаючи *Y* як набір ідентифікаторів, що містяться в таблиці маршрутизації *x*, механізм маршрутизації на основі *XOR*, застосований на сервері *x*, обирає сервер y ∈ Y, який мінімізує відстань до *z*, що виражається в наступній формулі.

(1)

Для підтримки переадресації трафіку таблиці маршрутизації на базі XOR, що обслуговуються на одному сервері, формуються за допомогою O (n) записів, де знання про сусідні сервери розповсюджуються на n стовпчиків, що називаються сегментами і представлені як β*i*, 0≤ i≤n-1. Кожного разу, коли сервер дізнається про нового сусіда b, він зберігає інформацію щодо сервера b у сегменті β*n-1-i* з урахуванням найвищого i, що задовольняє наступну умову:

(2)

Для того, щоб показати створення таблиць маршрутизації, розглянемо перший приклад, наведений у розділі 4.2.2, припускаючи, що сервер 29 як a=011101 та його сусід 13 як b=001101. Відстань d(a,b)=010000 та найвищий *i*, який задовольняє умовам (2), *i* = 4, в результаті чого ідентифікатор b=001101 повинен зберігатися в сегменті β*n-1-i* = β*1*. В принципі, умова (2) означає, що сервер зберігає *b* у сегменті β*n-1-i*, де *n-1-i* - це довжина найдовшого звичайного префіксу (lcp), що існує між обома ідентифікаторами *a* та *b*. Це можна спостерігати в таблиці 3, де сегмент β*0*, β*1*, β*2*, β*3*, β*4*, β*5* зберігає ідентифікатори LCP довжиною *0, 1, 2, 3, 4, 5* з сервером 29 (011101).

**Таблиця 3.** Приблизна таблиця маршрутизації для сервера 29 (011101) в кубі Хеммінга з простором ідентифікації сервера, у якому n = 6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 111101 | **0**01101 | **01**0101 | **011**001 | **0111**11 | **01110**0 |
| 100000 | **0**00000 | **01**0000 | **011**000 | **0111**10 |  |
| 100010 | **0**00100 | **01**0100 | **011**010 |  |  |
| 111111 | **0**01111 | **01**0111 | **011**011 |  |  |

Такий підхід до таблиць маршрутизації є однією з основних переваг механізму, що базується на XOR, оскільки серверу потрібно лише знати одного сусіда на одному відсіку можливих 2n серверів, доступних у мережі для успішного маршруту пакетів. Якщо на сервері є більше одного запису на сегмент, такі додаткові записи можуть оптимізувати процес маршрутизації, зменшуючи кількість завантажень на шляху від джерела до місця призначення. Інша важлива характеристика цієї таблиці маршрутизації пов'язана з кількістю серверів, які входять у кожен з сегментів. Є лише один сервер, який вписується в останній сегмент (β*5*), два в β*4*, чотири в β*3*, подвоєння до досягнення першого сегмента (β*0*), де 50% всіх серверів вміщуються в такі частини (32 сервери для n = 6 ). Для простоти засвоєння в таблиці 3 наведено приклади сусідніх серверів, що обмежуються чотирма рядками.

Згодом, беручи до уваги розповсюдження серверних ідентифікаторів сірого коду, прийнятих в кубі Хеммінга, заповнення комірок простіше, оскільки кожен з сусідів з відстані Хеммінга вміщується рівно на однин відрізок таблиці маршрутизації, забезпечуючи всі переадресації трафіку всередині куба. Примітка: у першому рядку таблиці 3 навмисне присутні всі сервери, ідентифікатори яких відрізняються від відстані Хеммінга 1 до сервера 29 (011101): сервер 111101 в β*0*, 001101 в β*1*, 010101 в β*2*, 011001 в β*3*, 011111 в β*4* та 011100 в β*5*. Як згадувалося раніше, для кубів Хеммінга більше 64 серверів (6-біт), лише 6 серверів з відстані Хеммінга 1 будуть фізично підключені до даного сервера. У цьому випадку процес сигналізації [21] використовується для виявлення інших серверів, розташованих на відстанях більше одного стрибка.

У наступному розділі представлені деякі оцінки запропонованої системи пошуку подібності в розподілених середовищах.

## **5 Оцінки**

По-перше, була оцінена кореляція між Хеммінга та співвідношенням косинуса, показуючи, що можна використовувати функцію RHH для створення ідентифікаторів вмісту, що зберігають його подібність у відстані Хеммінга між ними. Потім були представлені деякі оцінки стосовно розподіленої хеш-таблиці (DHT) і куба Хеммінга. Більше результатів можна знайти в розділах [7] та [8]. Ці оцінки показують, що можна зменшити відстань між подібними даними в розподілених середовищах, використовуючи обидва варіанти.

### 5.1 Подібність Хеммінга

У цьому розділі оцінюється кореляція між косинусною подібністю профілів для дорослих, що використовуються в тестах, та їх відповідну подібність Хеммінга. Для виконання тестів дорослі профілі індексувались за допомогою RHH функції, в результаті чого виконався двійковий ідентифікатор на 128 біт. Такі ідентифікатори були використані для індексування дорослих профілів у базі даних. Кожен профіль у наборі для дорослих використовувався як вектор запиту в тій самій базі даних. Потім було розраховано співвідношення косинуса та подібність Хеммінга між та всіма іншими профілями і в результаті обирали ті, що представляють співвідношення косинусів >= 0,7.

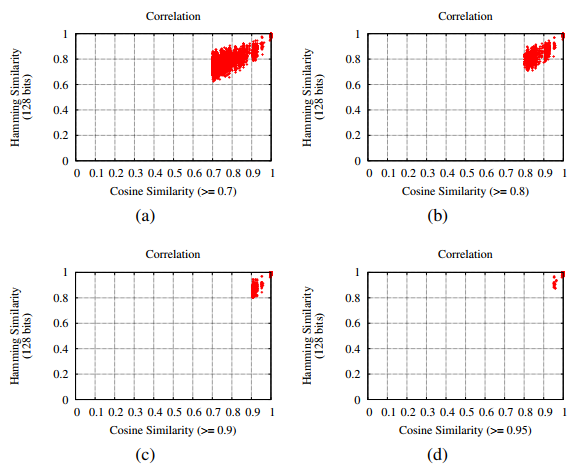
На рисунку 9 представлено кореляцію між косинусом та подібністю Хеммінга в тестах з використанням рівнів подібності 0,7, 0,8, 0,9 та 0,95. З рисунка 9 (а) можна помітити, що для співвідношення косинусів, більшого чи рівного 0,7, відповідна подібність Хеммінга змінюється в діапазоні [0,62 ... 0,85]. На рис. 9 можна помітити, що кореляція поліпшується на найвищому рівні подібності. Наприклад, для косинусної подібності 0,95 (рис. 9 (d)), подібність Хеммінга змінюється в приблизному діапазоні [0,9 ... 1].

Чим більша косинусна подібність, тим більша кореляція. У таблиці 4 показана середня кореляція між співвідношенням косинусів (sim*cos*) пар запитів і профілями для дорослих, а також подібністю Хеммінга (sim*ham*) їх 128-бітовими ідентифікаторами. Ці зразки сортували за такими рівнями подібності: 0,7, 0,8, 0,9 та 0,95.

**Таблиця 4.** Середня кореляція між співвідношеннями косинуса та Хеммінга

|  |  |
| --- | --- |
| **sim*cos*** | **Середня кореляція** |
| 0.7 | 0.84 |
| 0.8 | 0.9 |
| 0.9 | 0.95 |
| 0.95 | 0.98 |

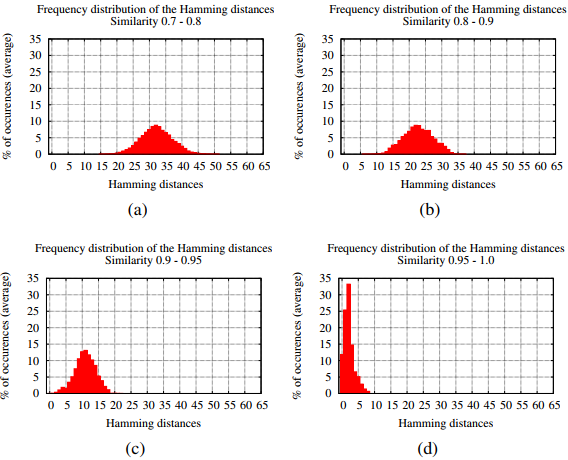
Результати показують сильну кореляцію між косинусом і подібністю Хеммінга для 128-бітних пар профілів дорослих, особливо для найвищих рівнів подібності. Досі вивчаючи цю таблицю, чим більше рівень подібності, тим більша кореляція між подібністю косинусів та ідентифікаторами профілю. Стандартні відхилення середніх значень для найнижчих рівнів подібності високі за рахунок імовірнісних характеристик функції RHH, але допустимі інтервали зразків були розраховані і є низькими та прийнятними. Для найвищих рівнів подібності стандартні відхилення є незначними, як допустимий інтервал зразків.



**Рис. 9.** Кореляція між співвідношеннями косинуса та Хеммінга за їх 128-бітовими ідентифікаторами з подібністю косинусів більше або рівно 0.7 (a), 0.8 (b), 0.9 (c) та 0.95 (d)

Іншим тестом для вираження доцільності пошуку подібності за цим сценарієм є оцінка частоти розподілу відстані Хеммінга для пари ідентифікаторів вмісту відповідно до їх косинусної подібності. Як зазначено у розділі 5.3.1, кожен профіль у наборі даних для дорослих був використаний як вектор запиту в усій базі даних. Для кожної пари дорослих було виміряно співвідношення косинуса та відповідну відстань Хеммінга їх ідентифікаторів. Оцінка проводилася з використанням чотирьох різних інтервалів подібності: [0.7..0.8], [0.8..0.9], [0.9..0.95], [0.95..1]).

На рис. 10 (а) показано розподіл частот відстані Хеммінга 128-бітних ідентифікаторів користувача, які мають співвідношення косинусів більше або рівне 0,7 і менше 0,8 [0,7 ... 0,8]. Як показано, результати мають нормальний розподіл, при якому більшість відстаней становить від 20% (25 біт) до 30% (38 біт) від загальної довжини ідентифікатора. Така ж поведінка спостерігається і в інших діапазонах: [0.8..0.9] (рисунок 10 (b)), [0.9..0.95] (рисунок 10 (c)) та [0.95..1.0] (малюнок 10 (г)).



**Рис. 10.** Розподіл частоти. Рівні схожості 0,7 (a), 0,8 (b), 0,9 (c) та 0,95 (d)

### 5.2 Розподілені хеш-таблиці (DHT) за Хеммінгом

Цей розділ описує деякі експерименти, спрямовані на оцінку DHTG Хеммінга. Основна ідея полягає в тому, щоб підтвердити таку DHT як цінний підхід для підтримки пошуку аналогічного вмісту в розподіленій середовищі. Для оцінки пропозиції для хеш-таблиць Хеммінга були виконані наступні експерименти:

* Ідентифікатори профілів для дорослих зі 128-бітною довжиною були згенеровані за допомогою RHH;
* Ідентифікатори цих профілів індексувалися та розподілялися в DHM за Хеммінгом та Хордом з 1000 та 10 000 однолітків;
* Кожен профіль цього набору був використаний як вектор запиту для подібних профілів у всіх аккаунтах одного і того ж набору даних. У наборі даних в загальній складності представлено 48842 профілів для дорослих, що означає, що запити 48842 x 48842 були реалізовані над набором даних для дорослих;
* Для кожного запиту був створений ідентифікатор, і пошуковий запит з таким ідентифікатором, як аргумент, був виконаний в обох DHT. Перевірка повідомлення пересилається пейеру, який відповідає за ідентифікатор запиту (хост-пейер), тобто наступнику ідентифікатора на кільці;
* Від хост-пейера завантажується кожен ідентифікатор профілю в межах рівня схожості запиту, а також вимірюється відстань у кількості стрибків від хост-пейерів до інших з подібним вмістом.

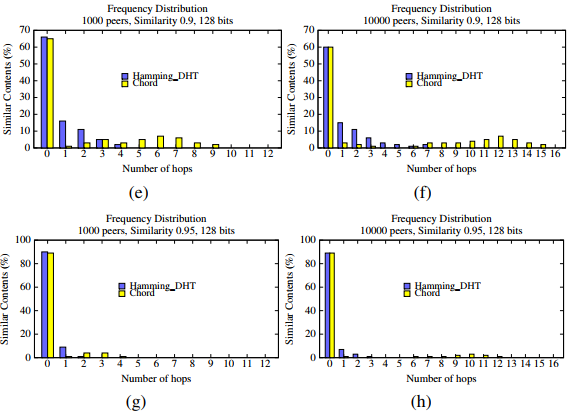
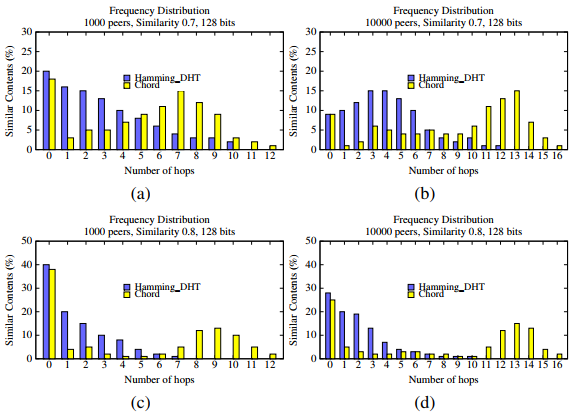
Ці випробування дозволили нам виділити наступні аспекти:

* Частотний розподіл кількості стрибків для отримання всіх схожих профілів у наборі відповідно до їх рівня схожості. Ця оцінка показує, що таблиця Хеммінга агрегує більше, ніж нормальна DHT у кільцевому стилі, наприклад, Хорд, зменшуючи відстань між подібними профілями в кількості стрибків;
* Відклик запиту, що відповідає частині профілів, що стосуються запиту та успішно завантажені. Ця оцінка показує, що можна побудувати більш ефективну пошукову систему на рівні таблиць Хеммінга за нижчою вартістю, виміряною у кількості стрибків для завершення запиту.

Для виконання тестів ми впровадили Хорд і DHM-симулятор Хеммінга, включаючи функцію генерації випадкових однолітків і приєднання їх до DHT кільцевого стилю. Також, розроблений симулятор показує і зберігає кожен ідентифікатор профілю k за допомогою операції put (k, v). Операція пошуку (k) повертає наступнику ключа k на кільце, яке представляє співрозмовника, відповідального за збереження профілю, пов'язаного з цією клавішею, хост-пейера. Операція get (k) була розширена для обробки запропонованого рівня взаємозв'язку з урахуванням формату get (k, sim): якщо вказано ключ (k) та рівень подібності (sim), всі аналогічні профілі, що зберігаються у хост-пейера, повертаються. Цей пошук поширюється на сусідів хост-пейерів з відстанню від 1 хеммінг (або довшими відстанями) з метою покращення результатів пошуку.

#### 5.2.1 Розподіл кількості стрибків

На графіках рис. 11 показано частотний розподіл кількості стрибків для отримання всього аналогічного вмісту. Графіки показують відсоток відновленого вмісту відносно загальної кількості аналогічного вмісту в наборі та його відповідну відстань до хост-пейера. Випробування проводились зі зміною кількості однолітків у кожному DHT для аналізу цього впливу в результатах. Було зімітовано 1000 гравців у Хорді та в DHTG Хеммінга з рівнями подібності 0,7 (рис.11 (а)), 0,8 (рис.11 (с)) та 0,9 (рис.11 (е)). Крім того, на рис. 11 (б), 11 (d) та 11 (f) показані результати, одержані за тими самими тестами, що імітували 10000 пейерів. Розмір ключів, використаних у цих тестах, становить 128 біт.



**Рис. 11.** Частотний розподіл кількості стрибків та 128-бітних ключів. 1000 однолітків з рівнем подібності 0,7, 0,8, 0,9 та 0,95. 10000 однолітків із рівнем подібності 0,7, 0,8, 0,9 та 0,95.

З цих результатів можна помітити, що DHT Хеммінга здатна скомпонувати більш подібні профілі в меншій кількості стрибків, тобто коротші відстані між ними. Довірчий інтервал для 95% зразків незначний. Процес набуття пальців для DHT Хеммінга, які показують відстань Хеммінга між пейерами, та організацією ідентифікаторів відповідно до послідовності сірого коду в кільці, сприяє результатам, представленим у цих тестах.

#### 5.2.2 Відклик

На рисунках 12 (а), 12 (с) та 12 (е) показаний відкликаний пошук подібності для Хорда та DHT Хеммінга з 1000 однолітків, з рівнями подібності 0,7, 0,8 та 0,9 на 128 біт. На малюнках 12 (b), 12 (d) та 12 (f) показані результати, отримані за тими самими тестами, що імітували 10000 однолітків.

З результатів на рис. 12 можна побачити, що використання DHT Хеммінга, як інфраструктури для підтримки пошуку подібності, є цінним підходом. Як приклад, з малюнка 12 (d) можна побачити, що в пошуковій системі, розробленій за DHT Хеммінга, яка має функцію get (k, 0.8) з глибиною 4 хемінга, може бути витягнуто приблизно 91% всіх подібних профілів, а за Хордом, лише близько 50%. Кращий розподіл частот, показаний на рисунку 11, підтверджує кращий відклик, отриманий за Хеммінгом в порівнянні з Хордом.

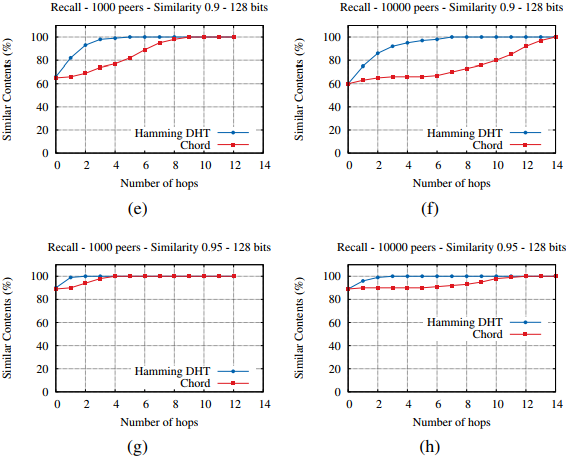
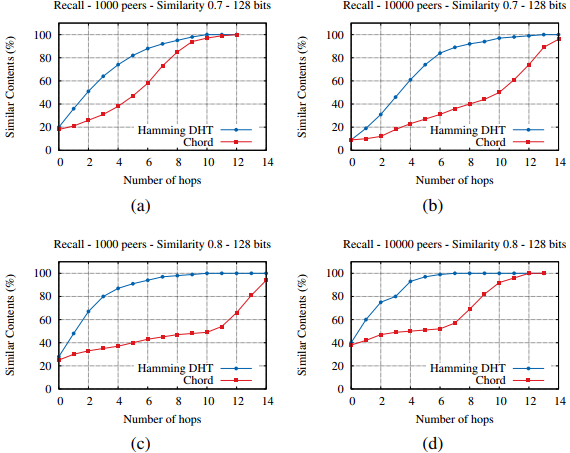
### 5.3 Куб Хеммінга (HCube)

Цей розділ описує експерименти, проведені при оцінці пропозиції HCube. Головна ідея полягає в тому, щоб підтвердити нашу пропозицію як цінний підхід для підтримки пошуку аналогічних даних. Для оцінки HCube виконані наступні експерименти:

* Ідентифікатори профілів для дорослих зі 128-бітовими файлами були створені за допомогою RHH, індексовані та розподілені в HCubes з 1024, 2048, 4096 та 8192 серверами;
* Профілі в наборі запитів були використані як векторні запити у наборі для дорослих. Пари запитів і дорослі були випадково вибрані для оцінки з урахуванням рівнів подібності більше або рівно 0,7, 0,8, 0,9 та 0,95;
* Для кожного запиту було створено ідентифікатор на 128 біт. Було здійснено пошук з такими ідентифікаторами та рівнями подібності Хеммінга. Для карти хост-сервера для кожного запиту кожен 128-бітний ідентифікатор був зведений до 10-, 11-, 12- і 13-бітного ідентифікатора з використанням RHH. Ці знижені ідентифікатори відповідають хост-серверам запитів для кожного оціненого розміру кубів Хеммінга;
* З хост-сервера завантажуються всі профілі, які відповідають рівню подібності Хеммінга.

Ці випробування дозволяють виділити наступні аспекти:

* Кореляція між косинусом та аналогією Хеммінга для ідентифікаторів дорослих (128 біт), а також між косинусом та аналогією Хеммінга для ідентифікаторів хостинг-серверів;
* Частотний розподіл кількості стрибків для отримання всіх аналогічних профілів у наборі відповідно до їх рівня схожості;
* Відклик запиту, що відповідає частці усіх відповідних профілів, які були успішно завантажені.



**Рис. 12.** Відклик. 1000 однолітків, рівні подібності 0,7, 0,8, 0,9 та 0,95. 10000 однолітків, рівні подібності 0,7, 0,8, 0,9 та 0,95.

Для виконання тестів дорослі профілі були індексовані та розповсюджені в HCube, використовуючи примітиву put (k, v). Примітива get (k, sim) була використана для вилучення всіх подібних профілів у хостинг-сервері та сусідів за ідентифікатором (k) та рівнем подібності (sim). Тільки для того, щоб дозволити нам оцінити відклики та кількість стрибків, цей пошук поширювався на всіх сусідів хостинг-сервера на налагодженій глибині.

#### 5.3.1 Кореляція

У таблиці 5 показано середнє значення кореляції між косинусною подібністю (simcos) пар запитів і профілями для дорослих та подібністю Хеммінга (simham) з їх 128-бітовими ідентифікаторами. Також представлена кореляція між подібністю косинусів та подібністю Хеммінга 10-, 11-, 12- та 13-бітових ідентифікаторів хост-сервера. Для розрахунку цих кореляцій ми випадковим чином обрали 10 зразків запитів і профілів для дорослих, кожен зразок з яких містить приблизно 1% від загальної кількості пар в наборі для дорослих. Ці зразки сортували за рівнем подібності: 0,7, 0,8, 0,9 та 0,95.

**Таблиця 5.** Середня кореляція між подібністю косинусів та Хеммінга

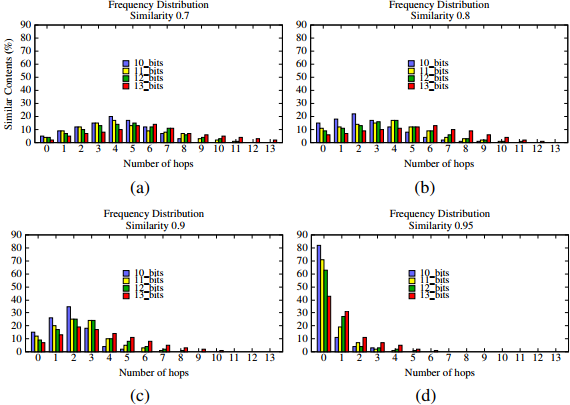
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 128-bits | 10-bits | 11-bits | 12-bits | 13-bits |
| Level |  |  |  |  |  |
| 0.7 | 0.84 | 0.44 | 0.48 | 0.50 | 0.52 |
| 0.8 | 0.90 | 0.48 | 0.52 | 0.63 | 0.68 |
| 0.9 | 0.98 | 0.79 | 0.84 | 0.86 | 0.90 |
| 0.95 | 1 | 0.98 | 0.99 | 0.98 | 1 |

Результати показують сильну кореляцію між косинусом і подібністю Хеммінга для 128-бітних пар профілів дорослих, особливо для найвищих рівнів подібності. Коли рівень подібності падає нижче 0,9, можна помітити помірну кореляцію (<0,8). Досі вивчаючи цю таблицю, можемо зробити висновок, що чим більший рівень подібності, тим більшою є кореляція між косинусною подібністю пар дорослих та схожістю Хеммінга з ідентифікаторами хостинг-серверів.

Стандартне відхилення цих середніх значень для найнижчих рівнів подібності є високим через ймовірнісну характеристику функції RHH, але інтервали допустимості для зразків були розраховані і низькі та прийнятні. Для найвищих рівнів подібності стандартні відхилення є незначними, як інтервали допустимості зразків.

#### 5.3.2 Розподіл кількості стрибків

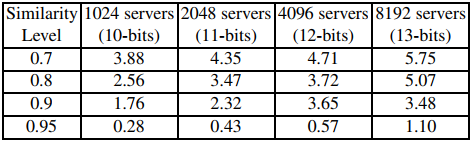
Результати на рис. 13 показують, що чим вищий рівень подібності запиту, тим нижчі зусилля, необхідні для відновлення аналогічних даних. Як приклад, на рис.13 (а), з використанням 10-бітного ідентифікатора сервера та рівня подібності Хеммінга 0,7, більшість подібних профілів заданого запиту розташовуються на відстані 2-6 стрибків від хост-сервера. Важливо тут помітити, що при використанні 10-бітного ідентифікатора сервера, подібність Хеммінга 0,7 еквівалентна відстані 3 хемінга у середньому. У 6-бітного HCube (повний HCube), що має 3 стрибки, так як відстань Хеммінга передбачає 3 стрибки від джерела до цілі. Як ще один приклад, на рис.13 (с), також використовуючи 10 біт для ідентифікатора сервера та рівень подібності 0,9, більшість аналогічних даних, пов'язаних із заданим запитом, становить від 0 до 3 стрибків. Подібність Хеммінга 0,9 у цьому випадку означає, що відстань Хеммінга між ідентифікаторами сервера, що зберігає подібні дані, в середньому дорівнює 1.



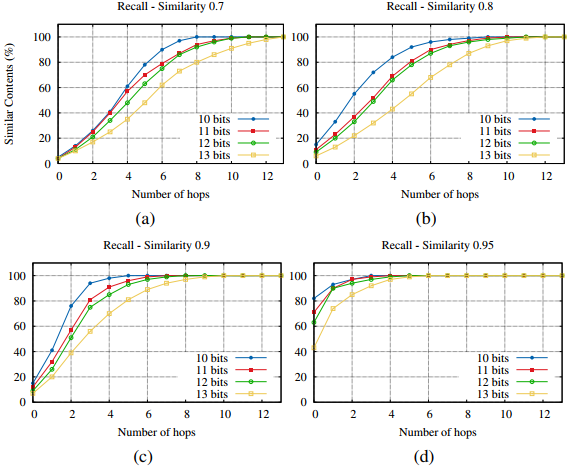
**Рис. 13.** Розподіл частоти. Рівні подібності 0.7 (a), 0.8 (b), 0.9 (c) та 0.95 (d).

У повному кубі Хеммінга з вимірністю n, де n - кількість біт ідентифікатора сервера, та рівнем подібності sim максимальна кількість хмелю, необхідна для відновлення подібних даних, дорівнює n-(sim\*n). Як приклад, з sim=0.7 та n=10, максимальна кількість стрибків, необхідних для відновлення всіх 0.7 аналогічних даних, становить 10-(0,7\*10)=3. Найвищі значення, представлені в результатах HCube, пов'язані зі зменшенням кількості розмірів, необхідних для побудови неповного куба Хеммінга. Середній показник кількості стрибків для отримання всіх подібних профілів у заданому запиті представлений у таблиці 6.

**Таблиця 6.** Середня кількість стрибків



#### 5.3.3 Відклик



**Рис. 14.** Відклик. Рівні подібності 0.7 (a), 0.8 (b), 0.9 (c) та 0.95 (d).

Важливою метрикою, яка вказує на ефективність будь-якої інформаційно-пошукової системи, є відклик, який представляє частку профілів, які є релевантними для запиту і успішно завантажені в пошук подібності. Ця оцінка показує, що можна побудувати ефективну пошукову машину на вершині HCube. На рис. 14 представлені результати оцінки відклику. Вони являють собою середнє значення всіх запитів і результатів у наборі для дорослих. З цих результатів можна побачити, що найкращі результати зустрічаються для найвищих рівнів подібності. Нарпиклад, приймаючи HCube з 4096 серверами (12-біт) і рівнем подібності 0,8, 20% аналогічних дорослих профілів витягується за допомогою 2 стрибків, і 50% з них отримують за 4 стрибки. У цьому ж сценарії, з використанням рівня подібності 0,95, 60% всіх подібних профілів зберігаються на хостинг-сервері, і розширюючи пошук своїх найближчих сусідів (1 стрибок), 90% всіх аналогічних профілів завантажуються.

У наступному розділі глава завершується, а майбутні роботи обговорюються.

## 6 Висновки та проблеми подальшого дослідження

В декількох словах, пошук подібності на основі метрики Хеммінга є інноваційною пропозицією. Основне припущення полягає в тому, що користувачі зі схожими профілями мають схожі інтереси. Профіль користувача може бути отриманий кількома способами, наприклад, шляхом сканування соціальних мереж, вилучення демографічної інформації, біографії навчальних програм або облікових записів користувачів. У порівнянні з іншими подібними роботами в літературі основними перевагами пропонованого пошуку подібності є: він не залежить від розмірності; подібність оцінюється незалежно від типу та семантики кожного аспекту вектора змісту; він передбачає низьку вартість обчислювальних операцій, таких як функція XOR, для обчислення подібності між двома ідентифікаторами. Також пошук подібності добре пристосований для індексації практично [7] або фізично розподілених систем [8] і може бути продовжений як гібридний підхід.

Використання ваг у пропозиції пошуку подібності - це майбутні дослідження. У цих початкових експериментах використання різних масштабів у деяких профільних атрибутах зумовлює їхню участь на різних рівнях для оцінки подібності векторних профілів. Наприклад, у наборі для дорослих, якщо користувач встановлює "вік" з більшою вагою, ніж інші характеристики, в таблиці 2, результат R9 може бути більш схожий на запит q, ніж R10. Те, як користувачі висловлюють свої уподобання, може бути реалізовано спільним способом.

Інший аспект пошуку подібності полягає в тому, що він легко адаптується до різнорідної пошукової системи. Система різнорідності може бути використана в кількох програмах для протиставлення різних рекомендацій у профілях. Потрібно лише шукати ідентифікатор зворотнього зв'язку профілю. Як приклад, пошук несхожості для наступного профілю <27; Приватний; Деякі коледжі; 11; Ніколи не одружений; Адм-канцелярський; Власна дитина; Білий; Жіночий; Сполучені Штати; > = 50K> використовуючи 0,9 як невідповідність рівня, повертає <38; Приватний; HS-grad; 9; Одружена-дружина-дружина; Ремісничий ремонт; Чоловік; Білий; Чоловік; Куба; <= 50K> і <37; Приватний; HS-grad; 9; Одружена-дружина-дружина; Продажі; Чоловік; Білий; Чоловік; Гаїті; <= 50K> серед інших.

Результати показують, що хеш-таблиця Хеммінга може бути корисним інструментом, який слугуватиме інфраструктурою накладання для пошуку подібності. Оцінка порівнює таблиці Хеммінга та Хорда, оскільки він може використовуватися як еталонний елемент у літературі розподілених хеш-таблиць, навіть якщо це не було запропоновано для пошуку подібності. Також, наскільки нам відомо, жодна інша глава в літературі DHT не вивчає подібність Хеммінга з ідентифікаторами вмісту, щоб запропонувати DHT, що спеціалізується на пошуку подібності, що ускладнює порівняння з іншими підходами.

HCube - це рішення Центру обробки даних, призначене для підтримки пошуку подібності у сценаріях великих даних, спрямованих на зменшення відстані та покращення відкликання подібного вмісту. У цьому розділі показано, що об'єднання представлення VSM, функції RHH, Gray SFC, тривимірної структури, що використовується в серверному центрі даних та XOR-рішення для маршрутизації, забезпечує необхідну основу для ефективного досягнення цілей HCube . Як майбутні роботи, будуть досліджені альтернативи для HCubes більше, ніж 8192 серверів. Практичним підходом може бути збільшення кількості мережевих ланок сервера. Однак, оскільки це фізично обмежений підхід, ми вважаємо, що цікавим варіантом може бути введення ієрархії HCube.

## Посилання

1. Gantz, J., Reinsel, D.: The Digital Universe Decade - Are You Ready? http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf (2010) (Online; Acesso em 2 de Marc¸o de 2013)
2. The Apache Software Foundation: ApacheR Hadoop, http://hadoop.apache.org/ (2013) (Online; Acesso em 5 de Marc¸o de 2013)
3. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
4. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: STOC 1998: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 604–613. ACM, New York (1998)
5. Indyk, P., Motwani, R.: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In: STOC 1998: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 604–613. ACM, New York (1998)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), <http://archive.ics.uci.edu/ml>
7. Villac¸a, R., de Paula, L.B., Pasquini, R., Magalh˜aes, M.F.: Hamming DHT: Taming the Similarity Search. In: Proceedings of the 10th Annual IEEE Consumer Communications and Networking Conference, CCNC 2013. IEEE Communications Society, Las Vegas (2013)
8. Villac¸a, R., Pasquini, R., de Paula, L.B., Magalh˜aes, M.F.: HCube: A Server-centric Data Center Structure for Similarity Search. In: Proceedings of the 27th International Conference on Advanced Information Networking and Applications, AINA 2013. IEEE Computer Society, Barcelona (2013)
9. Desai, A., Singh, H., Pudi, V.: DISC: Data-Intensive Similarity Measure for Categorical Data. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS (LNAI), vol. 6635, pp. 469–481. Springer, Heidelberg (2011)
10. Lee, D., Park, J., Shim, J., Lee, S.: Efficient Filtering Techniques for Cosine Similarity Joins. INFORMATION-An International Interdisciplinary Journal 14, 1265 (2011)
11. Lawder, J.: The application of Space-filling Curves to the Storage and Retrieval of Multidimensional Data. PhD thesis, University of London, London (December 1999)
12. Zhang, C., Xiao, W., Tang, D., Tang, J.: P2P-based multidimensional indexing methods: A survey. J. Syst. Softw. 84(12), 2348–2362 (2011)
13. Olszak, A.: Hycube: a dht routing system based on a hierarchical hypercube geometry. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2009, Part II. LNCS, vol. 6068, pp. 260–269. Springer, Heidelberg (2010)
14. Tang, C., Xu, Z., Mahalingam, M.: psearch: information retrieval in structured overlays. SIGCOMM Comput. Commun. Rev. 33, 89–94 (2003)
15. Bhattacharya, I., Kashyap, S., Parthasarathy, S.: Similarity Searching in Peer-to-Peer Databases. In: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, ICDCS 2005, pp. 329–338 (June 2005)
16. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. In: Proc. of the 7th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2006, vol. 7. USENIX, Berkeley (2006)
17. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon’s highly available key-value store. SIGOPS Oper. Syst. Rev. 41(6), 205–220 (2007)
18. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. IEEE/ACM Trans. Netw. 11(1), 17–32 (2003)
19. de Paula, L.B., Villac¸a, R.S., Magalh˜aes, M.F.: Analysis of Concept Similarity Methods Applied to an LSH Function. In: COMPSAC 2011: Computer Software and Applications Conference. IEEE, Munich (2011)
20. Faloutsos, C.: Gray Codes for Partial Match and Range Queries. IEEE Trans. Software Eng. 14(10), 1381–1393 (1988)
21. Pasquini, R.: Proposta de Roteamento Plano Baseado em uma M´etrica de OU-Exclusivo e Visibilidade Local. Phd. thesis, Faculdade de Engenharia Eletrica e Computac¸˜ao. Universidade Estadual de Campinas, Campinas, SP (June 2011)