Дослідження соціальних мереж на основі теорії графів

Студ. В. Боярський

Lviv Polytechnic National University, Automated Control Systems Department
28a/804 Bandera St., Lviv-13, 79013, Ukraine
E-mail: boyarsky.vitaliy@live.com

Abstract. Social network analysis is widely used in a wide range of applications and disciplines. Large amounts of data leads to the use of large-scale data processing. Apache Hadoop, which works on the programming model MapReduce, is one of such systems. A modified Dijkstra algorithm has been applied to investigate the theory of six degrees of separation. The results of the studies revealed an average length between any users of social networks, and confirmed the theory.

Keywords: Social network analysis (SNA), Graph, MapReduce, Six degrees of separation.

Вступ

Останнім часом дослідження соціальних мереж набуло великої популярності. В публічному доступі з'явилися персональні дані людей (факти біографії, відео-, фото-, аудіо-матеріали, маршрути подорожей, коментарі та інші дані.) та зв'язки між ними. Таким чином, соціальні мережі є унікальним джерелом особистих даних та інтересів реальних людей. Одним із способів вивчення соціальних мереж є соціальний граф, де основними елементами є взаємодія соціальних суб'єктів.

Граф є структурою даних в математиці та комп'ютерних науках, яка описується множинами вершин та зв'язків між ними. На сьогодні графи широко використовуються при моделюванні соціальних мереж. Постійне збільшення об'ємів графів призводить до використання систем масштабної обробки даних (Big Data).

Кожні два роки розмір "цифрового світу" подвоюється. Так за оцінками IDC розмір на 2006 р. становив $0{,}18\times10^{21}$ байт (0,18 зетабайт), а на 2013 рік він збільшився в десятки раз і становив 4,4 зетабайт. За прогнозами у2020 році "цифровий світ" буде містити стільки даних, скільки зірок в Всесвіті і досягне 44 зетабайт [1]. Розмір сховища даних Facebook щоденно збільшується на 500 терабайт. Станом на січень 2014 аудиторія соціальної мережі Facebook налічує 1,4 млрд. зареєстрованих користувачів. Мережа LinkedIn складається майже з 8 мільйонів вершин і 60 мільйонів ребер. Вконтакті, одна з найпопулярніших на території СНД і друга популярністю в Європі налічує більше 239 млн. користувачів.

Дослідження

Об'єми даних не дозволяють розмістити в пам'яті однієї машини, а доступ до жорсткого диску стає вузьким місцем. Для реалізації обробки графів необхідно опрацьовувати їх на декількох кластерах.

Модель розподілених обчислень MapReduce запропонована в 2003 році компанією Google, дозволяє паралельно опрацьовувати великі набори даних "Big Data", в комп'ютерних кластерах. Ця модель дозволила програмістам фірми Apache розробити Open Source проект Apache Hadoop, який на сьогоднішній день став ключовою технологією "Big Data". Щоправда, модель MapReduce не призначена для обробки запитів в реальному часі [2].

Робота MapReduce базується на двох функціях Map і Reduce. Вхідні дані розбиваються порціями і подаються на вхід функції Map у форматі (Key, Value). Результатом виконання є проміжні пари ключ/значення, які групуються за ключем, сортуються та подаються на вхід функції Reduce, яка формує кінцевий результат або пари для наступних ітерацій. Опрацювання даних лінійними алгоритмами дозволяє легко їх масштабувати на кластерах з будь якою кількістю машин [3].

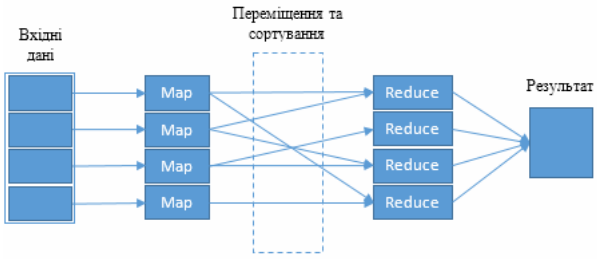

Рис.1. Принцип роботи MapReduce обчислень.

Для пошуку найкоротшого шляху від однієї вершини графу до інших, можна використати модифікації алгоритму Дейкстри та пошуку в ширину.

Алгоритм Дейкстри полягає у покроковому переборі всіх вершин графа G та присвоєнні мітки, яка є мінімальною відстанню від початкової вершини. Для початку присвоїмо початковій вершині значення 0, та всім іншим відстань рівну ∞. Перебираємо всі вершини графа, вибираючи вершину *u*, з найменшою відстанню. Для кожної сусідньої вершини *u*, крім вже відвіданих, вибираємо найменше між значенням мітки даної вершини та значенням нової довжини шляху, що становить суму відстані до вершини і довжини ребра [4].

Основою алгоритму Дейкстри є пріоритетна глобальна черга з відсортованим списком вузлів. Це неможливо в MapReduce, оскільки модель програмування не передбачає механізму обміну глобальними даними. Замість цього використано пошук в ширину, який досить легко масштабується.

Алгоритм працює шляхом зіставлення всіх вершин графа з їх сусідніми вершинами, створюючи пару ключ/значення, де ключем є сусідня вершина, а значенням – відстань та поточна вершина графа. Відстань до сусіднього вузла визначається сумою відстані від початкового вузла до поточного та довжини ребра до сусідньої вершини. На вході Reduce отримає значення всіх ключів, що є вершинами графа та список шляхів, які ведуть до відповідного вузла. Для кожної вершини вибираємо найменше значення шляху та обновляємо структуру графа.

```
class Mapper
        method Map(nid n; node N)
                d = N.Distance
                Emit(nid n; N)
                for all nodeid m = N.AdjacencyList do
                        Emit(nid m; d + 1)

class Reducer
        method Reduce(nid m; [d1; d2; ...])
        dmin = ∞
        M = Not(0) ;
        for all d = counts [d1; d2; ...] do
                if IsNode(d) then
                        M = d
                else if d < dmin then
                        dmin = d
        M.Distance = dmin
        Emit(nid m; node M)
```

Даний алгоритм є циклічним, де кожна ітерація відповідає новому MapReduce завданню. Від ітерації до ітерації потрібно зберігати структуру графа. Це досягається передачею вхідних даних Map на вихід разом з результатами обчислення відстаней до сусідніх вершин. В Reduce слід розрізняти структуру графа від списку значень відстаней. Знайдена мінімальна відстань оновлюється в структурі графа та подається на вихід Reduce. Результат виконання поточної ітерації слугуватиме вхідними даними для наступних ітерації. Кількість ітерацій для опрацювання всіх вершин дорівнює діаметру графа. У нашому випадку кількість ітерацій менша рівна шести [5].

Для проведення експериментів була вибрана соціальна мережа Вконтакті та мова програмування Java. Протягом двох тижнів було проаналізовано близько 90 млн. користувачів та 6,7 млрд. зв'язків між ними, тобто, близько 30% всієї мережі або 1,28% населення планети. Знайомими вважалися люди, які знаходилися в списку друзів, які були підтвердженні з обох сторін.

Для збору персональної інформації використано доступ до API-методів сервісу. Доступ до даних є тільки в авторизованих користувачів, що вимагало проходження авторизації за протоколом OAuth 2.0. Розміри даних зумовили використання паралельних методів збору (на стороні клієнта Вконтакті дозволяє викликати до 3 методів за секунду). Для підвищення швидкості завантаження, написано збережувальні процедури, які виконуються на стороні сервера (до 25 методів в одній процедурі).

Аналіз даних проводився у псевдо паралельному режимі Hadoop 2.2.0 з конфігурацією системи: Процесор Intel Core i5-4200M, 2.5 ГГц, 3Мб кеш-пам'ять, 2 ядра; 4Гб 1600МГц DDR3L SDRAM; HDD 600 Гб SATA (5400 об/хв); Windows 7 Ultimate.

Робота всіх ітерацій зайняла 4,5 год (рис. 2). Об'єм даних, що опрацьовувався становив 20,5 Гб.

Ефективність обчислень сильно залежать від пропускної можливості мережі між кластерами, оскільки структура графа передається по мережі після кожної ітерації. Оптимізувати цей алгоритм вдалося за рахунок використання оперативної пам'яті, в яку завантажуються відстані до відомих нам вершин. В такому випадку структура графа буде незмінна, а мережею будуть передаватися тільки відстані між вершинами.

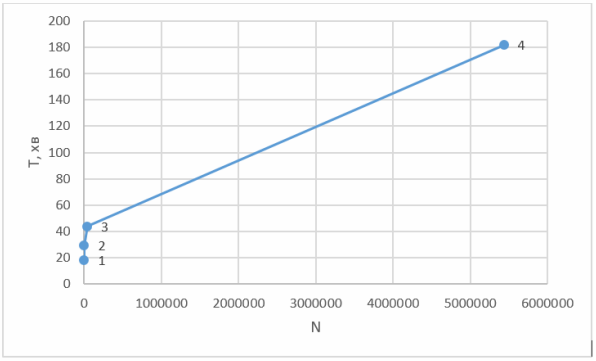

Рис.2. Залежність часу обчислень від розміру вхідних даних.

Використовуючи отриману таким чином велику базу даних користувачів, було вирішено перевірити число Данбара, суть дослідження якого полягала у визначенні максимальної кількості людей, з якими людина може підтримувати стабільні соціальні відносини. На основі отриманих даних Данбар вивів математичну залежність де кількість осіб в групі лежить від 100 до 230, найчастіше вважається рівною 150.

З отриманих результатів побудовано гістограму (рис. 3). Для наочності обрізано "хвіст" графіку, виключивши користувачів, кількість друзів яких є більше за тисячу. Це переважно відомі люди (музиканти, ведучі, шоумени, фотографи та інші) і, припустимо, вони навряд знають всіх своїх друзів.

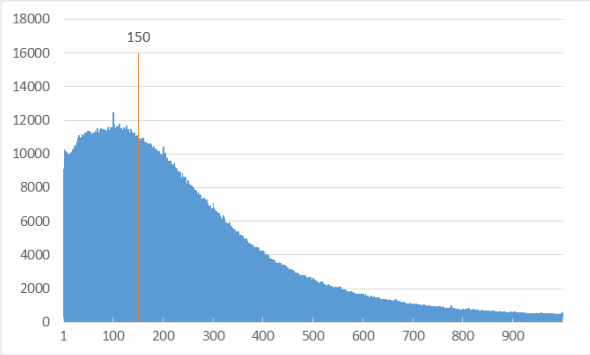

Рис.3. Кількість друзів користувачів.

У 1969 році американськими психологами С. Мілграмом та Дж. Треверсом була запропонована гіпотеза "шести рукостискань", яка полягала в тому, що кожна людина опосередковано знайома з будь-яким іншим жителем планети через недовгий ланцюжок спільних знайомих. У середньому цей ланцюжок складається з шести осіб.

Враховуючи кількість зав'язків, які індивід може підтримувати в соціальній мережі (число Данбара), у кожного з нас є приблизно 150 знайомих. У кожного з них теж приблизно по 150, що дає коло спілкування у 22,5 тис. осіб. На третьому кроці це коло буде складатись з 3,4 млн. осіб, а на четвертому – з 5 млрд. Це досить оптимістичні припущення, але насправді коло спілкування індивіда А може досить суттєво перетинатися з колом спілкування індивіда В. Тобто частина знайомих кожного з ваших друзів входить до вашого кола знайомств.

Також важливу роль відіграє кастовість населен-

ня, тобто люди схильні спілкуватися з собі подібними. І наявність в одному колі друзів представників двох різних соціальних груп є припустимо досить малою.

В результатів аналізу даних, отриманих з соціальної мережі, визначено кількість вершин та ребер на кожному кроці ітерації від початкової вершини (табл. 1). Кількість ребер відповідає загальній кількості знайомих, а вершин - реальній кількості знайомих після об'єднання спільних.

Таблиця 1.

Коло спілкування

Розмір кола спілкування	Кількість ребер	Кількість вершин
1	147	147
2	62895	38180
3	25046986	4953068
4	2318488285	85509543

Для перевірки гіпотези шести рукостискань було отримано дані про 100 тис. випадкових аккаунтів, 3% з яких виявилися фейковими (створеними програмами-роботами і т.ін.) і не були залучені до аналізу. Результати дослідження показали, що переважаюча більшість людей знайомі через чотири рукостискання (рис. 4).

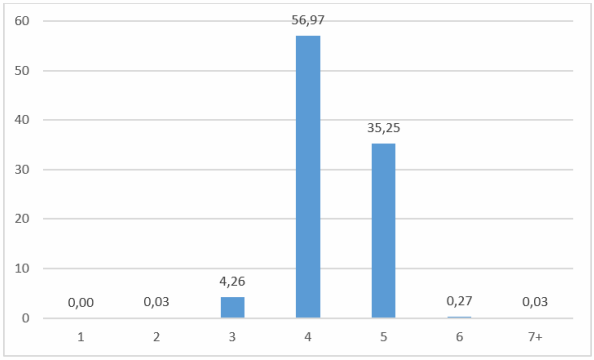

Рис.4. Розподіл знайомих за кількістю рукостискань

Як приклад, побудовано граф з найкоротших шляхів від автора статті до засновника Вконтакті П. Дурова. Граф містив 691 вершину та 517 різних руко стискань з довжиною ланцюжка у 4 рукостискання. Для наочного відображення кількість вершин відфільтрувалися з умовою: кількість друзів менша за тисячу. В результаті фільтрації граф зменшився до 36 вершин та 13 різних рукостискань (рис. 5). Щоб відобразити соціальні зв'язки використано Java бібліотеку Prefuse Beta.

Візуалізація соціальних мереж, як один зі способів їх аналізу, має важливе значення. Вона дає можливість виявити неформальні товариства, знайти слабкі місця, визначити лідерів і т.д. (рис.6).

Висновок

Системи для аналізу даних на основі моделі MapReduce не є оптимальними для обробки графових даних, оскільки більшість задач є ітераційними з використанням переходів по графу, а модель не передбачає обміну глобальними даними. При цьому ефективність обчислень сильно залежить від пропускної здатності мережі. Більш ефективними є системи обробки графових даних, що відрізняються від моделей на основі MapReduce. До таких систем, зокрема, можна віднести Giraph і GraphLab [2].

За результатами дослідження встановлено, що середня довжина шляху між будь-якими користувачами на графі соціальної мережі Вконтакті є 4,326. Це значення узгоджується з гіпотезою шести рукостискань. Тож це дослідження можна вважати ще одним її підтвердженням.

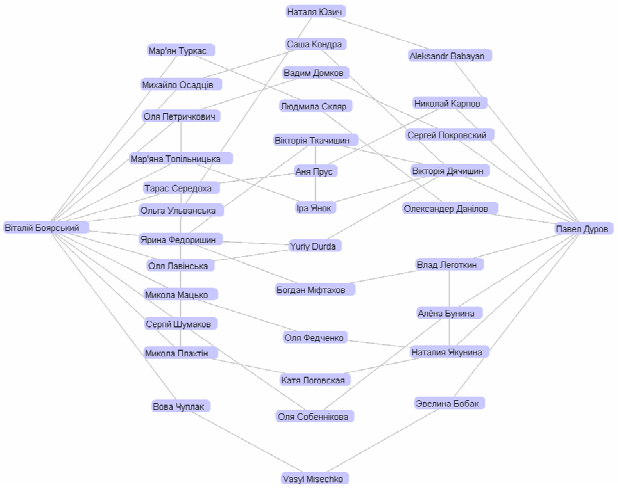

Рис.5. Візуалізація ланцюжка користувачів.

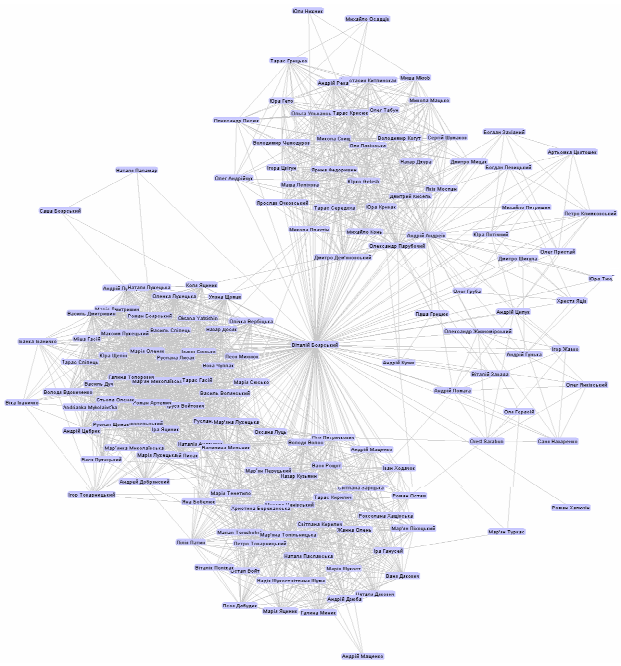

Рис.6. Соціальний граф.

[1] The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things http://ukraine.emc.com/leadership/digital-universe/2014iview/executive-summary.htm

[2] Шериф Сакр. Обработка больших объемов графовых данных: путеводитель по современным технологиям: http://www.ibm.com/developerworks/ru/library/os-giraph/

[3] T. White. Hadoop: The Definitive Guide, Third Edition. - O'Reilly Media, 2012. – 686 p.

[4] Уилсон Р. Введение в теорию графов. Пер. с. англ. - М.: Мир, 1977. – 208 с.

[5] J. Lin and C. Dyer. Data-Intensive Text Processing with MapReduce. - University of Maryland, 2010. – 175 p.