УДК 681.3

В.В. Ткаченко

ВИКОРИСТАННЯ ГРІД-ТЕХНОЛОГІЙ ДЛЯ АНАЛІЗУ СОЦІАЛЬНИХ МЕРЕЖ

Розглянуто питання можливості та доцільності використання Grid-технологій для вирішення задачі аналізу соціальних мереж. Реалізовано один з методів соціального аналізу та проведено експериментальні обчислення на тестовій Grid системі з використанням соціальних даних з реальних соціальних мереж. Визначено недоліки обраного методу розв'язання задачі та окреслено шляхи їх подолання.

Вступ

Масове використання Інтернету та природне прагнення людей до спілкування та об'єднання у соціальні групи сприяло створенню оперативних (онлайн) соціальних сервісів. Соціальний сервіс – це Інтернет-сервіс, шо надає послуги з утворення та підтримки соціальних спільнот і мереж. Тобто, з позиції соціології, люди, здійснюючи різні комунікації у цьому сервісі утворюють соціальну мережу, що в кінці-кінців може охопити весь світ (якщо припустити, що абсолютно ізольованих груп людей не буває).

Соціальна мережа – це спільнота людей, об'єднаних однаковими інтересами, уподобаннями, або тих, що мають інші причини для безпосереднього спілкування між собою. Сучасні Інтернет-сервіси забезпечують користувачів усіма можливими інструментами для спілкування одне з одним – відео, чати, зображення, музика, блоги, форуми тощо.

Для бізнесу соціальні мережі виступають новим каналом комунікації із споживачем, та інструментом дослідження уподобань аудиторії. У щорічній доповіді Gartner Emerging Technologies Hype Cycle [1] представлені досягнення в області інтернету та програмного забезпечення, які вплинуть на розвиток бізнесу в найближчі 10 років. В області Web 2.0 найбільш важливою технологією названий аналіз соціальних мереж, що досягне своєї зрілості менш ніж за два роки. Аналіз соціальних мереж (у визначенні Gartner) – це використання інформації та знань, зібраних з персональних ресурсів людей, з метою вивчення цільових ринків, винесення бізнес-рішень і створення команд виконавців проектів. Тобто, задача аналізу мереж є важливою не тільки для соціологів, але й для вирішення задач більш ефективного ведення бізнесу. Методи аналізу соціальних мереж успішно застосовуються для аналізу політичних угруповань [2], терористичних організацій [3], поширення інфорації серед працівників компаній та визначення реального положення посадовців. Таких прикладів можно навести значно більше.

Зупинимось на задачі соціального аналізу. Існує чотири підходи щодо аналізу соціальних мереж [4]:

1. Структурний – акцентує увагу на геометричній формі та інтенсивності взаємодій (вазі ребер). Всі актори розглядаються як вершини графа, які впливають на конфігурацію ребер і інших акторів мережі. Особлива увага приділяється взаємному розташуванню вершин, центральності, транзитивності взаємодій. Для інтерпретації результатів у даному випадку використовуються структурні теорії і теорії мережевого обміну.

2. Динамічний – увага акцентована на змінах у мережевій структурі з часом. Вивчаються причини зникнення і появи ребер мережі; зміни структури мережі при зовнішніх діях; стаціонарні конфігурації соціальної мережі.

3. Нормативний – вивчає рівень довіри між акторами, а також норми, правила та санкції, які впливають на поведінку акторів у соціальній мережі та процеси їх взаємодій. У цьому випадку аналізуються соціальні ролі, які пов'язані з даним ребром мережі, наприклад, відношення керівника і

підлеглого, дружні або родинні зв'язки. Комбінація індивідуальних і мережевих ресурсів актора з нормами і правилами, що діють в даній соціальній мережі, утворює його «соціальний капітал».

4. Ресурсний – розглядає можливості акторів по залученню індивідуальних і мережевих ресурсів для досягнення певної мети і диференціює акторів, що знаходяться в ідентичних структурних позиціях соціальної мережі, за їх ресурсами. Як індивідуальні ресурси можуть виступати знання, престиж, багатство, (етнічность), стать (гендерна ідентичність). Під мережевими ресурсами розуміються вплив, статус, інформація, капітал.

властивостей соціальних мереж. Приблизні розміри соціальних мереж наведено у табл. 1.

У даній роботі розглядаються саме оперативні соціальні мережі. Як задача, яка буде реалізована з виконанням грід-технологій, обрано задачу пошуку соціальних ланцюгів між усіма соціальними акторами у мережі

Існуючі системи для аналізу соціальних мереж

На сьогоднішній день існує багато систем аналізу соціальних даних, що використовуються здебільшого соціологами

Таблиця 1. Приблизні розміри деяких соціальних мереж

Мережа	Кількість акторів, тис	Кількість зв'язків, тис	Джерело даних
MSN messenger	180 000	1 300 000	[5]
LinkedIn	22 000	Немає даних	[6]
FaceBook	100 000	Немає даних	[7]
Orkut	3 000	223 500	[8]
LiveJournal	5 000	77 500	[8]
Середня мережа блогів	4 400	5 000	[Liben-Nowell et al. 2005, Backstrom et at. 2006]
Мережа цитувань серед фізичних статей	31	350	[5]

Наступні аналітичні методи застосовуються у соціальному аналізі [4]:

1) методи теорії графів;
2) методи знаходження локальних властивостей суб'єктів, методи визначення еквівалентності акторів, включаючи їх структурну еквівалентність;
3) блокові моделі і ролева алгебра;
4) аналіз діад і тріад;
5) імовірнісні моделі;
6) кореспондентський аналіз і топологічні методи, що представляють мережу як деякий симпліціальний комплекс.

Особливістю оперативних соціальних мереж, як джерел соціальних даних є те, що користувачі розширюють свої мережі доволі швидко та без додаткових витрат зі сторони соціального аналітика. Це зумовлює великі розміри соціальних графів, на яких можуть здійснюватись масштабні дослідження різноманітних

для проведення досліджень: UCINET, NetDraw, Pajek, Netminer, Visone, SNA/R, StOCNET, Negopy, InFlow, GUESS, NetworkX, prefuse, JUNG, BGL/Python та інші. З детальним списком систем для аналізу соціальних мереж можна ознайомитись на сайті міжнародної спільноти з аналізу соціальних мереж ISNA [9] та в огляді [10], а нині зупинимось на деяких з них, які я намагалась застосовувати для аналізу даних з оперативних соціальних мереж. Однією з найвідоміших є UCINET – комерційний продукт, що розроблюється американською компанією Analytic Technologies. Вона дозволяє здійснювати аналіз соціальних мереж використовуючи широкий спектр методів аналізу, експортувати дані з найпопулярніших форматів, інтегрується з системою візуалізації мереж NetDraw. Одним з найголовніших обмежень є максимальний розмір акторів у ме-

режі 32767, але й при обробці даних для 5,000 10,000 акторів виникають значні затримки у роботі. Для обробки більших за розмірів даних створена словенськими розробниками Vladimir Batagelj та Andrej Mrvar система Pajek [11]. Обробка великих соціальних мереж досягається її кластеризацією на менші і застосовуванням адаптованих алгоритмів.

Жоден з розглянутих автором продуктів не дозволив у прийнятний час проаналізувати соціальну мережу з 2 000 000 акторів. Тому для зберігання такого (або більше) об'єму даних та його аналізу має бути розроблений інший продукт, що мав би більш зручний та доступний для більшої кількості користувачів інтерфейс. Звичайно, створюваний продукт не дозволить здійснити всі види соціального аналізу, але оскільки «генераторами» соціальної мережі є звичайні люди (не соціологи), то зацікавлені у характеристиках мережі саме вони. Тому акцент у системі буде зміщено з повноти охоплення методів соціального аналізу на можливість аналізу реальних оперативних соціальних мереж та зручний доступ до результатів.

Опис типової задачі аналізу

Основною задачею соціальних мереж є підтримання існуючих та пошук нових зв'язків між людьми. Тому велику практичну цінність несе інформація про довжину та склад соціального ланцюга між двома акторами у мережі. Введемо наступну термінологію для опису соціальних графів:

1. Соціальний об'єкт (актор) – структурний елемент соціальної мережі, наприклад, людина, група людей, організація, країна і т.д.

2. Соціальний зв'язок (взаємозв'язок) – будь-який акт взаємодії між соціальними об'єктами, наприклад, товариство, листування, цитування, коментування і т.п.

3. Соціальний граф – граф, вузлами якого є соціальні об'єкти, а ребрами – соціальні зв'язки.

Розглянемо зміст цих понять на наступному прикладі. Нехай тестова соціальна мережа складається з шістьох акторів: A, B, C, D, E та F. Відповідний соціальний граф показано на рис. 1.

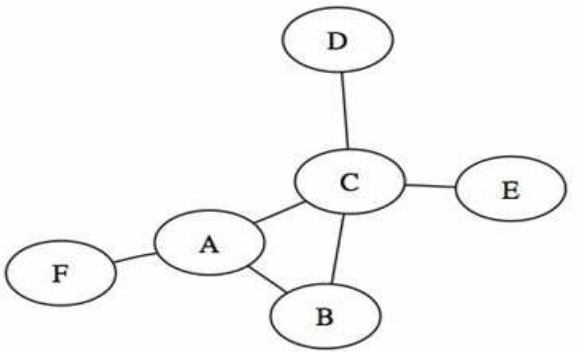

Рис. 1. Тестовий соціальний граф

Соціальний ланцюг між акторами F та E складається з двох інших акторів: A -> C. Зауважимо, що існує й інший, з трьох акторів: A -> B -> C. Практична цінність пошуку таких ланцюгів полягає у встановленні контакту з якимось актором через знайомства.

Найпростішим методом пошуку всіх таких ланцюгів є простий перебір усіх варіантів. Одним з способів здійснення такого перебору є піднесення до степеня матриці суміжності M. Так, щоб отримати кількість ланцюгів довжиною 2 необхідно піднести M до квадрата і отримане значення $M^2(i,j)$ буде вказувати на кількість ланцюгів між акторами *i* та *j* (*0* у випадку їх відсутності), для M^n відповідно це значення буде вказувати на кількість ланцюгів довжиною n-1.

Для нашого тестового графа матриця інцидентності матиме вигляд:

ij	A	B	C	D	E	F
A	3	1	1	1	1	0
B	1	2	1	1	1	1
C	1	1	4	0	0	1
D	1	1	0	1	1	0
E	1	1	0	1	1	0
F	0	1	1	0	0	1

Відповідно M^2 =

ij	A	B	C	D	E	F
A	0	1	1	0	0	1
B	1	0	1	0	0	0
C	1	1	0	1	1	0
D	0	0	1	0	0	0
E	0	0	1	0	0	0
F	1	0	0	0	0	0

Для прикладу, $M^2(A,E) = 1$, що підтверджується наявністю зв'язків вершини A з вершиною C і C з E.

Але, характер соціальних даних, отриманих з оперативних соціальних мереж такий, що соціальний граф розріджений, і формат зберігання даних у вигляді матриць суміжності не є оптимальним, а метод множення матриці суміжності також не є оптимальним. Тому застосовується модифікація описаного алгоритму.

Опис алгоритму

Для знаходження ланцюга соціальних знайомств, застосовується доволі простий та неефективний метод перебору всіх варіантів ланцюгів (складність $O(n^2)$), характеристики якого будуть покращено за рахунок використання розподілених обчислень.

Представимо соціальний граф у вигляді масиву множин об'єктів, що мають зв'язок з даних об'єктом, тобто

$$SocGraph = [S_1, S_2, \dots, S_n],$$

$$S_i = < O_1, O_2, \dots, O_m > .$$

Отримання ланцюга соціальних зв'язків 1-го порядку:

$$SocGraph_i^2 = \text{Union } Si < \text{intersect} > S_j,$$

$$j = 0..N, j! = i .$$

На мові псевдокоду алгоритм має наступний вигляд:

```
for i:=1 to < кількість акторів > – 1 do
   множ1 = отримати список друзів для
   i-го актора
   for j:= i+1 to < кількість акторів > do
        множ2 = отримати список друзів
        для j-го актора
        if (перетин множ1 та множ2 не пус-
        тий) then
            Зберегти (i та j друзі)
        else
            Зберегти (i та j не друзі)
        fi
   od
od
```

У поточній реалізації алгоритму розпаралелюванню підлягає зовнішній цикл по *i*.

Для зменшення кількості варіантів для перебору застосовуються наступні евристики:

- якщо актор не має друзів, то для нього не шукаються ланцюги, він просто ігнорується;
- якщо два актора вже є друзями, то це не підтверджується ще раз;
- як потенційні друзі розглядаються тільки актори з більшим ідентифікатором (це випливає з симетричності матриці суміжностей для неорієнтованих графів, якими здебільшого є соціальні).

Модель розпаралелювання та Грід-платформа

Як схема обробки соціальних даних застосовується модель MapReduce [12], що складається з двох фаз Map (розподіл задач) та Reduce (формування результату). Зупинимось на кожній з них окремо.

- Фаза Map: вхідні дані (у нашому випадку списки суміжності $<V_1,V_2, \dots, V_n>$) розподілюються поміж обчислювачів блоками за k списками value = (V_i,V_j) і кожному набору ставиться у відповідність ключ, що складається з номерів відповідних вершин key = (i,j).

Кожен обчислювач отримує вхідні дані – пару <key,value> і виконує необхідні операції над ними. Наступний його крок полягає у формуванні проміжного результату:

intermResult = <key, operation(value)>

і надсилання його на керуючий вузол, що виконує процедуру об'єднання.

- Фаза Reduce: дана фаза полягає у формуванні кінцевого результату з отриманих від обчислювачів проміжних, тобто $finalResult = ReduceOperation\ itermResult_i$.

У якості програмно-апаратної платформи для соціального аналізу доцільно використати Грід. Такі спроби робились у [13].

Грід або Грід-інфраструктура – це розподілена програмно-апаратна комп'ютерна мережа з принципово новою організацією обчислень і керування потоками

завдань і даних. Така комп'ютерна інфраструктура призначена для об'єднання обчислювальних потужностей різноманітних організацій. На основі технології Гріду (Грід-технології) здійснюється інтегрування регіональних і навіть національних обчислювальних комп'ютерних інфраструктур для створення об'єднаних інтернаціональних ресурсів, щоб розв'язувати маштабні науково-технічні задачі. В Гріді передбачено інтегрування великої кількості географічно віддаленних комп'ютерних ресурсів. У ідеальному випадку користувачеві не потрібно знати, де розміщено ресурси, які він використовує (у цьому і є аналогія з мережою електропостачання).

Серед основних на даний час завдань, розв'язуваних за допомогою Грід-технологій, можна виділити:

• організацію ефективного використання ресурсів для невеликих задач з залученням тимчасово простоюваних комп'ютерних ресурсів;

• розподілені суперобчислення, розв'язки дуже великих задач, які вимагають величезних процесорних ресурсів, пам'яті тощо;

• обчислення з залученням великих обсягів географічно розподілених даних, наприклад, в метерелогії, астрономії, фізиці високих енергій;

• колективні обчислення за участю користувачів з різних організацій.

Більш детально з Грід-технологіями можна ознайомитись у [14].

Основними вимогами до платформи для аналізу є можливість реалізації запропонованої схеми розпаралелювання, використання неспеціалізованних комп'ютерів як обчислювачів, легкість конфігурування, інтеграція з рішеннями для кешування даних. У якості програмної платформи обрано Gridgain [15], так як вона на відміну від інших (Globus toolkit, GridWay, UNICORE, Sun Grid Engine та інші) є найбільш простою у використанні, що дозволяє більш швидко реалізовувати необхідні обчислення.

Gridgain – це Open Source Грід-платформа, що заснована на Java та підтримує балансування навантаження й обробку розподілених даних. Архітектура системи на базі цієї платформи показана на рис. 2.

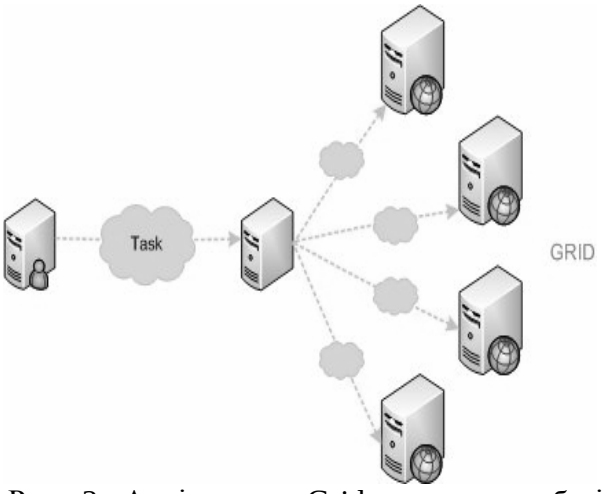

Рис. 2. Архітектура Grid-системи на базі GridGain

Заснований на SPI (Service Provider Interface – інтерфейс надання послуг) архітектурі Gridgain дозволяє розширювати функціональність без зміни ядра системи, що робить можливою реалізацію нових концепцій (у тому числі й призначених для користувача) дуже простою. Балансування навантаження має декілька стратегій у тому числі: Round Robin (за умовчанням), адаптивне і прив'язане до даних (афінне) балансування. Працюючи з великими обсягами даних необхідно уникати пересилки їх по мережі з вузла на вузол. Gridgain надає наступну функціональність для оптимізації роботи з великим обсягом даних – афінний балансувальник навантаження, що дозволяє прив'язувати завдання до оброблюваних даних і виконувати їх на тих вузлах, де знаходяться дані.

GridGain працює за принципом динамічного перерозподілу завдань ("Work stealing") між вузлами гріда (передаючи їх з переобтяжених вузлів у гріді на найменш завантажені). Цей SPI так само як й інші підтримує відмовостійке і кероване виконання завдань на Грід. Початкова ідея "Work stealing" ґрунтується на методі fork/join, що запропонований у [16] та запланований у версії SE 7 платформи Java. Ця функціональність може використовуватися для уникнення "зависання" завдань на слабких вузлах.

Сегментація Гріда на групи, які пра-

цюють над своїми завданнями, дозволяє при необхідності реалізувати концепцію керуючих та обчислювальних вузлів. Збереження проміжних результатів обчислень, що необхідне при виконанні тривалих завдань, дозволяє продовжити обчислення з останньої "збереженої" крапки.

Платформа включає також наступну функціональність:

• моніторинг: GridGain дозволяє отримувати різну інформацію про вузли, таку як використання пам'яті або процесора, кількість виконуваних завдань на вузлі та іншу;

• IOC(Inversion of control): GridGain підтримує можливість загального доступу до призначених для користувача ресурсів (таких як JDBC з'єднання або Spring біни в процесі виконання);

• контекст виконання завдання, що дозволяє підзадачам обмінюватися інформацією.

GridGain інтегрується з багатьма open source і комерційними продуктами, такими як JUnit, ASPECTJ, Spring, JBoss & JGroups, GlassFish, WebLogic, WebSphere, Coherence, Mule, JXInsight та GigaSpaces.

Розглянемо основні концепції створення Грід-систем на базі GridGain та спрощену послідовність виконання задачі на Гріді (рис. 3).

Рис. 3. Виконання підзадачі на вузлі при централізованому доступі

Одиницею виконуваної задачі є GridTask, який відповідальний за розподіл бізнес логіки на декілька підзадач (GridJob). Рішення як саме та в залежності від яких параметрів розподіляти задачу на підзадачі приймається кодується у методі GridTask::map, вхідними параметрами якого є параметри задачі та список доступних вузлів, на яких буде виконуватись задача. Даний метод викликається перед виконанням задачі і результатом роботи його є список підзадач, які необхідно виконати. Далі Грід-система забезпечує передачу підзадач на обчислювачі (крок 1 рис. 3), отримання необхідних для вирішення підзадач ресурсів, їх виконання (крок 2 і 3) та повернення результатів обчислення підзадачі (крок 4) до вузла, який ініціював задачу (керуючий вузол). Список результатів передається у метод GridTask::reduce відповідної задачі, в якому результати об’єднуються за заданим алгоритмом і формується фактичний результат виконання задачі.

Для більшої наочності розглянемо обчислення простої та добре знайомої задачі пошуку заданого тексту в деякій сукупності текстових файлів. Задачею у даному випадку буде пошук тесту, а вхідними параметрами, сам текст та список файлів. Ураховуючи кількість вузлів, список файлів буде розподілений серед вузлів і на вхід кожної поступить текст, який необхідно знайти, та список файлів, за пошук у яких цей вузол відповідальний. Доступ до файлів, які в нашому випадку знаходяться в якомусь хранилищі, може бути здійсненним, наприклад, за протоколами передачі даних HTTP, FTP, SFTP і т. п. Після отримання файлів за списком та пошуком у них, вузол формує результат виконання підзадачі у вигляді списку файлів та позицій тексту в них та надсилає на керуючий вузол. Як результат отримаємо місцезнаходження заданого тесту. Якщо потрібно буде знайти якийсь інший текст у тому ж списку файлів, то немає необхідності завантажувати ці файли знову. В цьому випадку доцільно використати ідеї кешування інформації, що у GridGain реалізується прив’язанним до даних розподілом навантаження, яке вищеописано. Дана ідея

показана на рис. 4. При надходженні (крок 1) підзадачі на обчислювальний вузол, у залежності від розміщення необхідних даних вона пересилається на потрібний вузол (2 чи 3 на рис. 4). Далі, підзадача обчислюється (крок 2) на визначеному вузлі та результат передається (кроки 3-4) на керуючий вузол.

Рис. 4. Виконання підзадачі з прив'язаним до даних розподілом навантаження

Обчислювальні ресурси

Як обчислювальні ресурси у Грід-системах можуть використовуватись як звичайні комп'ютери, так і більш потужні сервери, кластери та суперкомп'ютери. Вузли можуть класифікуватися за різними технічними параметрами і задачі розподілятися за вузлами у залежності від її класу (у GridGain існує вбудований спосіб оцінки можливостей вузлів – GridLocalNodeBenchmark).

Тестовий Грід для дослідження задачі складався з 5 вузлів, що мали приблизні конфігурації наведені у таблиці 2 (варто зазначити, що швидкість обміну даними з сервером БД є величиною доволі нестабільною і наведені виміри при передачі файлу розміром 100Мб лише характеризують клас мережі).

У Гріді все ж таки існує два централізовані елементи:

- керуючий вузол (macbook) – розподіляє задачі серед обчислювачів, сама є обчислювачем, вимірює час виконання задачі;

Таблиця 2. Склад тестового Гріда

Кодова назва	Технічні характеристики, CPU, RAM, OS, Net, JRE, GridGain	Швидкість обміну даними з сервером БД, 100Mb
selena	AMD Athlon 64 Processor 2800+ 1.83 GHz, 1 Gb, Debian (2.6.18), UA-IX JRE6, 2.1.0	Майже необмежена, бо і є сервером БД
netcat	AMD Athlon 64 Processor 2800+ 1.83 GHz, 1 Gb, Debian (2.6.18), UA-IX JRE6, 2.1.0	500KB/s
macbook	Intel Core 2 Duo 1.83 GHz, 1.5 GB Mac OS X 10.5.6, UA-IX JRE5, 2.1.0	10MB/s
frontera	Intel Core2 Duo CPU 2.33 GHz, 3 Gb Debian (2.6.18), UA-IX JRE6, 2.1.0	20KB/s
netsfera	Intel Core2 Duo CPU 2.33 GHz, 1 Gb Windows XP, UA-IX JRE6, 2.1.o	10MB/s

• сервер баз даних (selena) – зберігає соціальні дані та результати аналізу, найбільш «вузьке» місце системи.

Схема мережевого зв'язку між вузлами та інформаційні потоки показані на рис. 5. Темніші лінії показують двосторонні потоки обміну данними з сервером БД.

Як вищезазначено, платформа GridGain має компонентну архітектуру (компонентом є SPI) і за пошук та об'єднання окремих вузлів у топологію відповідає Discovery SPI. Існує багато цього SPI: Jgroups, Jms, Mail, Mule, Coherence, Multicast та інші. Для об'єднання тестового Гріда використовувався JGroups [17] тулкіт для здійснення надійних мультикаст та TCP (Transmission Control Protocol) коммунікацій між хостами у мережі. Хости, яким необхідно об'єднатись, обмінюватись даними, об'єднуються у групи. Політика додавання/видалення елементів у групі конфігурується. Одним з найпростіших варіантів є задання списку IP (Internet protocol) адрес інших учасників групи.

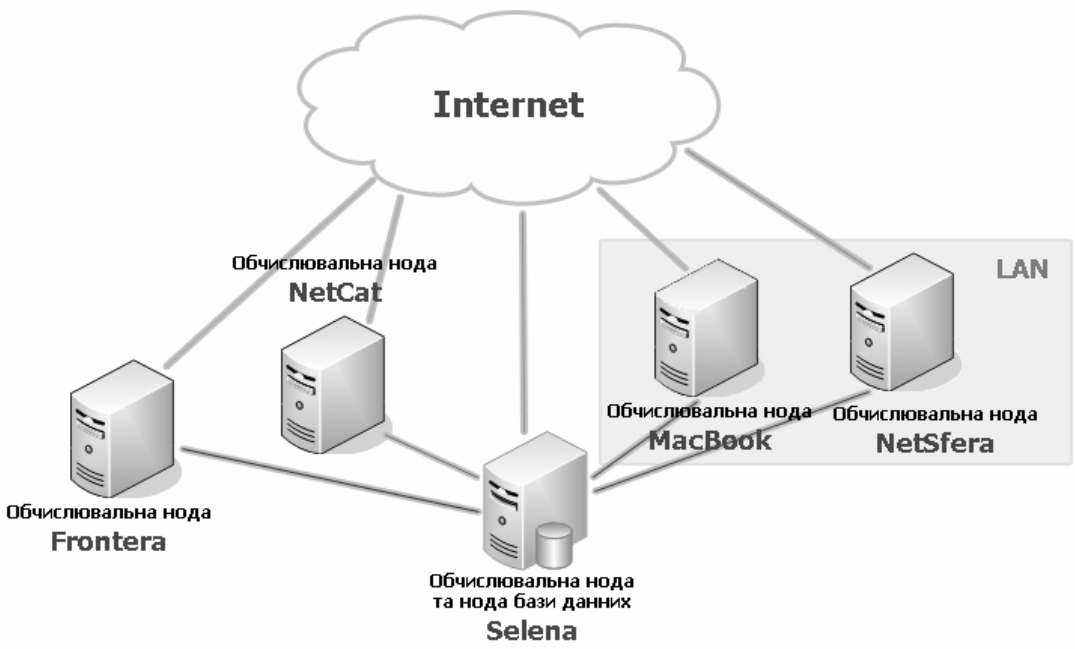

Рис. 5. Схема мережі та інформаційних потоків у тестовому Гріді

На рис. 6 показаний «відбиток» топології Гріда на момент «випадання» з нього 1-го вузла (netsfera) і на ньому видно, що Грід складається з 4-х вузлів і має сумарну кількість процесорів 7. Також наведені деякі характеристики вузлів та їх унікальні ідентифікаційні номери.

```
>>> ------------------
>>> Discovery Snapshot.
>>> ------------------
>>> Number of nodes: 4
>>> Topology hash: 0x45E1943E
>>> Local: DC6DE47A-69AA-4323-B387-E6BCF37ADC23, 77.120.101.19, Linux i386 2.6.18-4-686, aver, Java(TM) SE Runtime Environment 1.6.0_11-b03
>>> Remote: 52EB6526-0BB7-4F44-B754-1715802E1643, 193.178.34.50, Linux i386 2.6.18-4-686, vera, Java(TM) SE Runtime Environment 1.6.0_11-b03
>>> Remote: B58BC4FE-7046-476A-A2AD-1E7F9B00F227, 77.47.176.4, Mac OS X i386 10.5.6, ivira, Java(TM) 2 Runtime Environment, Standard Edition 1
.5.0_16-b06-284
>>> Remote: 3EF2A080-F4FD-43CF-B58F-AEC1A67FAF78, 91.211.117.19, Linux i386 2.6.18-6-486, vera, Java(TM) SE Runtime Environment 1.6.0_11-b03
>>> Total number of CPUs: 7
```

Рис. 6. Миттєвий «відбиток» топології Гріда на базі GridGain

Соціальні дані

Створювана система ставить на меті аналіз реальних, а не змодельованих, соціальних мереж. На жаль, дані, що збираються популярними оперативними мережами є закритими для вільного доступу, точніше важкодоступними. Щодо створення статичного образу мережі існує три підходи:

- прямий доступ до даних у внутрішньому форматі сервісу – цей метод може бути використаний лише за домовленністю з власником сервісу;
- використовувати API для доступу до інформації про користувача – ця можливість інколи пропонується у деяких оперативних сервісах і може мати деякі обмеження (необхідність авторизації, обмеження кількості запитів);
- "краулінг" – тобто простий обхід сторінок користувачів мережі та їх синтаксичний аналіз із метою отримання даних про користувача, а в нашому випадку списку його друзів.

Таблиця 3. Характер даних, що використовувались для тестування

Параметр	bigmir.net	LiveJournal	Orkut
Кількість акторів	2 148 838	5 284 457	3 072 441
Кількість активних акторів	136 643	Немає даних	Немає даних
Кількість зв'язків (ребер)	741 236	77 402 652	223 534 301
Дата отримання даних	Серпень 2007	9-11 грудня 2006	3 жовтня - 11 листопада 2006

Для тестування системи використовувались дані з трьох популярних в Україні та за кордоном сервісів, що пропонують користувачам можливість об'єднання у соціальні мережі, а саме такі:

- український портал bigmir.net, що надає серед інших сервісів можливість створення щоденників, фотоальбомів та ін.; дані були отримані першим методом;
- LiveJournal (www.livejournal.com) – віртуальна спільнота, в якій користувачі можуть вести свої блоги;
- Orkut (www.orkut.com) соціальна мережа створена компанією Google.

Дані для останніх двох отримані з проекту збору соціальних даних з оперативних мереж (табл. 3) [17].

Підсистема зберігання даних

Вибір підсистеми зберігання соціальних даних залежить від характеру самих даних, алгоритмів роботи з ними та способу використання отриманих результатів. На жаль, на сьогодні майже кожна з систем для аналізу соціальних мереж та оперативних соціальних сервісів має свій внутрішній формат зберігання даних. Враховуючи те, що дані соціальних мереж можуть бути представлені у вигляді соціального графа, і використовують модифікації форматів для зберігання графів. Прикладами таких форматів є:

- csv (comma separated values) – найпростіший та найпоширеніший з форматів для представлення табличних даних (будь-який граф може буде представлено у вигляді таблиці, де кожен рядок відображає відношення між зазначеними у ньому двома вершинами);
- dot – використовується у системі візуалізації графів GraphViz;
- pajek – формат даних відомої системи аналізу та візулізації соціальних мереж Pajek;
- dl – внутрішній формат системи UCINET;
- XML подібні формати – NetML, DyNetML.

У eSNA наразі реалізована підтримка двох форматів вхідних даних csv файли та дані, що були попередньо експортовані у внутрішню базу даних системи.

Дані про відношення між акторами зберігаються у вигляді реляційної табл. 4.

Таблиця 4. Формат зберігання даних

Поле	Формат даних	Опис
userID	BIGINT, 8 знаків	Ідентифікатор актора
friendID	BIGINT, 8 знаків	Ідентифікатор пов'язаного з ним актора
level	TINYINT, 2 знаки	Довжина соціального ланцюга, 1 – безпосередньо знайомі

У якості СКБД була використана MySQL, що була розміщена на централізованому сервері. Це зменшує продуктивність та надійсність системи і є найслабшим та набільш навантаженним елементом системи. У подальшому необхідно застосувати ідеї розподіленого зберігання даних для зменшення навантаження на цю підсистему.

На самому початку level=1 для всих зв'язків. По мірі опрацювання алгоритмом даних додаються нові знайдені пари пов'язаних людей і записується значення level+1 у поле level.

Аналіз результатів експериментів

Експерименти здійснювались на вищеописаному тестовому гріді та над частиною соціальних даних. Досліджувався вплив розміру блоку даних що передається на обробку обчислювачу, а також прискорення обчислень для різної кількості обчислювачів. У табл. 5 наведено звідну таблицю результатів експериментів та графічно показано на рис. 7. Абсолютні значення часу виконання є доволі неточними, оскільки на час здійснення тестових запусків швидкість передачі даних між сервером БД була порівняно малою.

Таблиця 5. Результати експериментів, час у мс

Дані	1 вузол	2 вузли	4 вузли
50 по 10	24342	39180	50334
100 по 10	45647	46401	514431
150 по 10	111270	103450	80453
300 по 10	410849	410849	350478
500 по 50	1008168	816540	715521
300 по 100	410849	309557	250521
500 по 100	1008168	746542	675125

При збільшенні розміру блоку даних, що формує одну підзадачу, зменшуються додаткові витрати на координування, але у разі помилки під час виконання задачі та ж сама задача підлягає виконанню з самого початку. Для уникнення подібних ситуацій можна зберігати проміжні результати підзадачі в разі неможливості її завершення на поточному вузлі, наступна буде продовжувати з останнього збереженного стану. В платформі GridGain це технологія має назву GridCheckPointSPI.

Прискорення обчислень при використанні 2 обчислювачів замість одного склало приблизно 1.2, а при використанні