Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

ДИПЛОМНА РОБОТА МАГІСТРА

Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних

Рівень вищої освіти Другий (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітня програма Освітньо-професійна програма інженерія програмного забезпечення

ДРІПЗ 2001112.20.02.05 ПЗ

Виконав: студент 2 курсу, група ІПЗм-20-2 _____ (підпис) А.І. Дрозд (Ініціали прізвище)
Керівник к.т.н, доцент . _____ (підпис) Д.М. Медзатий (Ініціали прізвище)
Нормоконтроль к.т.н, доцент _____ (підпис) Ю. В. Форкун (Ініціали прізвище)

До захисту допускаю:
Зав. кафедри ІПЗ д-р фіз.-мат. наук, проф. _____ (підпис) Л.П. Бедратюк (Ініціали прізвище)
7 грудня 2021 р.

Хмельницький 2021р.

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Другий (магістерський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
02.09 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Дрозду Андрію Ігоровичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних

Керівник проекту (роботи) Медзатий Дмитро Мколайович
к.т.н., Доцент
Затверджена наказом ректора університету від 25.08.2021 р. № 102

2. Строк подання студентом проекту (роботи) на кафедру 01.12.2021 р.

3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1 Дослідження предметної області та постановка задачі
2 Концепції, моделі та методи вирішення задачі
3 Алгоритми та технології вирішення задачі
4 Реалізація та тестування програмної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Презентаційні матеріали (слайди)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Антиплагіат	Гурман І.В., доцент	4.12.21	4.12.21
Нормоконтроль	Форкун Ю.В. доцент	30.11.21	3.12.21

7. Дата видачі завдання 01 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження; визначення структури дипломної роботи	01.09 – 09.09.2021	
2 Робота над розділом 1 дипломної роботи – вивчення літературних джерел; аналіз відомих моделей, методів та засобів за темою роботи; висновки до розділу та постановка задачі	10.09 – 25.09.2021	
3 Робота над розділом 2 дипломної роботи – розробка моделей та методів вирішення поставленої задачі; висновки до розділу	26.09 – 10.10.2021	
4 Робота над науковими публікаціями	11.10 – 20.10.2021	
5 Робота над розділом 3 дипломної роботи – розробка алгоритмів та технологій, проектування для вирішення поставленої задачі; висновки до розділу	11.10 – 26.10.2021	
6 Робота над розділом 4 дипломної роботи – програмна реалізація спроектованих рішень, результати експериментів, їх аналіз; висновки до розділу	27.10 – 15.11.2021	
7 Узгодження постановки задачі, отриманих результатів та висновків; написання вступу, загальних висновків, оформлення джерел посилання та додатків; оформлення пояснювальної записки та графічних матеріалів згідно вимог стандартів	16.11 – 30.11.2021	
8 Попередній захист дипломної роботи	Листопад (згідно графіка)	
9. Перевірка роботи на наявність плагіату; нормоконтроль; брошурування пояснювальної записки; підготовка супровідних документів	01.12 – 04.12.2021	
10 Підготовка до захисту дипломної роботи	05.12 – 08.12.2021	

Студент _____ А.І. Дрозд
Підпис Ініціали, прізвище

Керівник проєкту (роботи) _____ Д.М. Медзатий
Підпис Ініціали, прізвищ

РЕФЕРАТ

Тема кваліфікаційної роботи: «Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних». Автор проекту: Дрозд Андрій Ігорович.

Керівник роботи: Медзатий Дмитро Миколайович.

Пояснювальна записка: 72 с., 8 рис., 2 дод., 95 джерел.

КЛАСИФІКАТОР, ОБРОБКА ДАНИХ, РОЗПОДІЕНІ ОБЧИСЛЕННЯ, ОБЧИСЛЮВАЛЬНІ РЕСУРСИ, ПОТОКИ ДАНИХ, БАЛАНСУВАННЯ НАВАНТАЖЕННЯ.

Об'єктом дослідження є обробка розподілених потоків даних.

Предметом дослідження є методи розподілу обчислювальних ресурсів для обробки розподілених потоків даних.

Мета роботи – розробити метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних з метою досягнення ефективності обробки.

У кваліфікаційній роботі проаналізовані методи розподілу обчислювальних ресурсів для обробки розподілених потоків даних.

Розроблений метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних, що дає змогу досягти ефективності обробки, на відміну від відомих методів, за рахунок інтегрованого поєднання схем планування та балансування навантаження між обчислювальними ресурсами.

ADN
Підпис

03.12.2021
Дата

ABSTRACT

Theme of qualification work: "Method of distribution of computing resources for processing distributed data streams ".

Author of the project: Drozd Andriy Ihorovych.

Project manager: Medzatiy Dmitry Nikolaevich.

Explanatory note: 72 pp., 8 figs., 2 appendice, 95 sources.

CLASSIFIER, DATA PROCESSING, DISTRIBUTED CALCULATIONS, COMPUTER RESOURCES, DATA FLOWS, LOAD BALANCE.

The object of research is processing distributed data streams.

The subject of research is the methods of allocation of computing resources for processing distributed data streams.

The purpose of the work is to develop a method of allocating computing resources for processing distributed data streams in order to achieve processing efficiency.

The qualification work analyzes the methods of allocation of computing resources for processing distributed data streams.

A method of computing resources allocation for processing distributed data streams has been developed, which makes it possible to achieve processing efficiency, in contrast to known methods, due to an integrated combination of scheduling schemes and load balancing between computing resources.

ADN
Signature

03.12.2021
Date

ЗМІСТ

ВСТУП

Зі зростанням дослідницького інтересу до розподіленої обробки даних в останнє десятиліття дослідники та практики галузі розробили кілька потокових систем, наприклад, Aurora [8], Borealis [2], StreamBase [9], IBM InfoSphere Streams [10], S4 [11], Apache Storm [12], STREAM [9] та System S [13]. Ці системи DSP споживають і обробляють безперервні потоки даних відповідно до заздалегідь визначених запитів [14]. Запити DSP аналогічні тим, які визначені в системах управління базами даних (СКБД), в той час як більшість DSP (наприклад, Borealis [2]) підтримують модифікацію та налаштування динамічних запитів. Загальна практика полягає в тому, щоб перевести такі запити в ряд операторів, і кожен оператор представляє собою основну операцію, яка може бути повторно використаною.

Але потребують удосконалення методи обробки потокових даних для покращення ефективності таких систем обробки.

Мета роботи: розробити метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних з метою досягнення ефективності обробки

Для досягнення мети дослідження поставлено наступні завдання:

– розробити модель затримки оператора;

– розробити модель пропускної здатності системи з врахуванням особливостей проаналізованих недоліків в цій предметній області;

– здійснити моделювання системи та результати експериментів з моделлю;

– розробити метод та алгоритми розподілу обчислювальних ресурсів для обробки розподілених потоків даних;

– здійснити реалізацію запропонованого рішення.

Об’єктом дослідження є обчислювальних ресурсів для обробки розподілених потоків даних.

Предметом дослідження є методи розподілу обчислювальних ресурсів для обробки розподілених потоків даних.

Практична цінність роботи полягає в можливості реалізації розробленого методу розподілу обчислювальних ресурсів для обробки розподілених потоків даних та практичному застосуванні розроблених систем, що в результаті дало можливість покращити ефективність обробки на 3-5%.

Під час виконання завдань дослідження були застосовані:

– методи обробки даних;

– методи класифікації;

– методи розподілених обчислень.

Наукова новизна роботи [95]: розроблений метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних дає змогу досягти ефективності обробки, на відміну від відомих методів, за рахунок інтегрованого поєднання схем планування та балансування навантаження між обчислювальними ресурсами.

У кваліфікаційній роботі проаналізовані методи розподілу обчислювальних ресурсів для обробки розподілених потоків даних.

За результатами кваліфікаційної роботи опубліковані тези для наукової конференції «Актуальні проблеми комп’ютерних наук»:

Дрозд А. І., Форкун Ю.В. Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних // Зб. наук. пр. наукової конференції «АПКН-2021». Хмельницький ХНУ. – 2021. – С. 319 - 320.

1 АНАЛІЗ РОЗПОДІЛЕНОЇ ПАРАЛЕЛЬНОЇ ОБРОБКИ ПОТОКІВ

1.1 Поняття потокової обробки

Широко застосовувані традиційні рішення обробки даних (наприклад, реляційні системи управління базами даних) не здатні відповідати зростаючій потребі в миттєвій обробці щодо збільшення базових обчислень, а також можливостей зберігання та непередбачуваних запитів користувачів [1]. Тому обробнику обробки потоку даних пропонується розібратися з потоками даних миттєво за допомогою безперервних запитів [2] і декількох фреймворків для обробки потоків даних (DSP), які були запропоновані, щоб впоратися з необмеженими потоками даних. Вони в першу чергу спрямовані на досягнення результатів обробки з низькою затримкою і високою пропускною здатність [3, 4, 5]. Хоча менша затримка гарантує, що кортежі даних можуть оброблятися миттєво і направлятися для подальшої обробки, більш високий рівень пропускної здатності вказує на те, що система здатна обробляти велику кількість даних. Методи, включаючи перевпорядкування/розділення операторів, паралелізацію, застосовувалися для прискорення обробки потоку або підвищення рівня пропускної здатності [6]. DSP-застосунки дозволяють безперервно обробляти інформаційні потоки масштабованим і відмовостійким способом. Потоки формалізовані як сукупність кортежів даних для споживання та обробки таких даних, тоді як кортеж даних розглядається як процесор. Крім того, DSP-застосунки розроблені як системи пропуску, які складаються з серії операцій обробки, а типові функції включають агрегацію, розщеплення, злиття, ранжування тощо. Загальна мета проектування застосунків DSP полягає в тому, щоб зберегти операцію якомога простішою, а операції повинні бути реалізовані як окремі оператори [6]. Інтерпретуючи застосунок DSP як окремі оператори, потоки даних, які безперервно потрапляють в систему, можуть оброблятися за допомогою розподілених і паралельних методів програмування [7]. Зі зростанням дослідницького інтересу цієї нової парадигми обробки в останнє десятиліття дослідники та практики галузі розробили кілька потокових систем, наприклад, Aurora [8], Borealis [2], StreamBase

[9], IBM InfoSphere Streams [10], S4 [11], Apache Storm [12], STREAM [9] та System S [13]. Ці системи DSP споживають і обробляють безперервні потоки даних відповідно до заздалегідь визначених запитів [14]. Запити DSP аналогічні тим, які визначені в системах управління базами даних (СКБД), в той час як більшість DSP (наприклад, Borealis [2]) підтримують модифікацію та налаштування динамічних запитів. Загальна практика полягає в тому, щоб перевести такі запити в ряд операторів, і кожен оператор представляє собою основну операцію, яка може бути повторно використаною.

Оскільки основною проблемою все-таки залишається пропускна здатність, то було запропоновано кілька методів оптимізації для прискорення обробки потоку або максимізації пропускної здатності обробки. M.Hirzel et al. [15] Поділяють методи оптимізації на дві групи залежно від того, чи потрібно змінювати графік DSP. Більш конкретно, деякі методи вимагають зміни графіка DSP (наприклад, оператор злиття [16] для злиття операторів або поділу [15] для розділених операторів), в той час як інші намагаються змінити семантику DSP без зміни структури графіка (наприклад, пролиття навантаження [14]). Наприклад, переупорядкування оператора [17] переміщує більше вибіркових операторів до верхнього потоку, щоб зменшити загальний обсяг даних, які будуть оброблятися. Вибірковість позначає кількість кортежів даних, що генеруються одним вхідним кортежем даних [18]. Крім того, система розміщення операторів [16, 19, 20] приймає рішення про розподіл операторів у фізичних або віртуальних машинах відповідно до певних обмежень. Стратегія розміщення має тенденцію торгувати вартістю зв'язку проти використання ресурсів при розгортанні DSP в розподілених хостах з наявністю неоднорідності мережевих і обчислювальних ресурсів [20]. Щоб уникнути зміни макета та семантики застосунків DSP, розглядатимемо розміщення операторів DSP для оптимізації обробки потоків даних. Іншими словами, розглядатимемо ефективні стратегії розміщення для DSP щодо різних цілей. Зокрема, розміщення операторів в ПП також стосується розподілу ресурсу як неоднорідності базової розподіленої інфраструктури. Хоча управління ресурсами в загальних хмарних застосунках широко вивчалося в останні кілька десятиліть за

підтримки віртуалізації мережевих і обчислювальних ресурсів [15, 21, 22], існуючі методи не застосовуються в контексті обробки потоків даних. В основному це викликано динамічним характером обробки потоків даних і невизначеністю, а також складнощами, викликаними DSP, що ускладнює розробку загальної схеми управління ресурсами.

1.2 Кластер та його складові компоненти

Застосунки, які обробляють потоки інформації, як правило, можна класифікувати як два типи, обробку потоку даних (DSP) [6, 13, 23] і складну обробку подій (СОП) [24]. Хоча DSP нагадують СКБД, оскільки обробляє дані за допомогою послідовності операцій реляційної алгебри, СОП прагне мати справу з кількома пов'язаними подіями за допомогою виразів високого рівня [24]. Зокрема, дані зберігаються та індексуються в реляційних базах даних для обслуговування для цілей обробки, і оператор, щоб запустити процедуру обробки. Однак дані у форматі безперервних потоків не обов'язково зберігаються, а зовнішня подія в DSP може викликати взаємодію між користувачем і даними, тобто аномалія, виявлена в системі виявлення вторгнень або оповіщення про файл в системах моніторингу оточуючого середовища. Отже, очікується, що він доставить приблизний вихід, який постійно оновлюється DSP або СОП застосунків. DSP-застосунки дозволяють обробляти безперервні потоки даних масштабованим і відмовостійким способом за підтримки розподілених і паралельних методів програмування [25]. Зокрема, потокові дані формалізуються як сукупність кортежів для обробки, а кортеж даних розглядається як елемент даних в DSP. Ці кортежі схожі на рядки в реляційних базах даних, які складаються з імені та пов'язаних значень властивостей [25, 26]. На практиці велика кількість географічно розподілених застосунків може бути залучена для збору даних і розглядатися як джерело даних DSP. Зазвичай існує позначка часу, пов'язана з кортежем даних, щоб представити час, який він генерується або передається. Як споживач потокових даних, DSP обробляє потоки даних відповідно до попередньо визначених операторів системи обробки. Функції,

оголошені в операторах, подібні до операцій, визначених у запитах бази даних, і типових функцій, включаючи, але не обмежуючись: операція з групування підмножини кортежів даних щодо певних умов; операція з поділу всього потоку на менші блоки, які можуть бути пов'язані з різними рівнями паралелізму; операція об'єднання даних з різних каналів вводу стосується конкретних вимог; переосмислити кортежі на основі певного принципу; індивідуальна реалізація дозволяє застосовувати інтелектуальний аналіз даних, машинне навчання або інші методи для виконання складної аналітики [25]. Будь-яка складна система може бути розкладена на ці незалежні операції, які багаторазово використовуються в різних застосунках для полегшення управління базовим ресурсом. Саме такі робочі блоки нанесені на карту і розташовані в конкретних робочих вузлах до виконання РП. Крім того, на відміну від наукових робочих потоків або застосунків, які зазвичай виконуються у відносно короткостроковій перспективі, очікується, що DSP працюватимуть протягом тривалого періоду часу.

Як одна з найнадійніших відкритих джерел, розподілених в режимі реального часу обчислювальної системи, Apache Storm призначений для боротьби з необмеженим потоком даних швидким, масштабованим і відмовостійким способом [27]. Він був широко реалізований в промислових цілях, і відіграє значну роль у великих застосунках обробки даних (наприклад, Twitter, Yahoo!, Baidu) [28].

1.3 Стратегії розподілу та групування даних

Підвищення рівня паралелізму операторів є ефективним підходом до підвищення продуктивності обробки потоків [29, 30]. Паралелізм топології може бути реалізований шляхом налаштування кількості завдань, кількості виконавців і кількості робочих процесів [30]. Зокрема, оператори реалізуються як одне або кілька завдань (або екземплярів). Також, виконавець виконує завдання як потік, в той час як група потоків визначається як робочий процес. Віртуальна машина Java (JVM) зберігається для кожного робочого процесу, і завдання з різних топологій ізолюються шляхом зіставлення їх у різних робочих процесах. Загалом, кілька СОП

готуються як робочі процеси для топології. Робочі процеси можуть виконуватися в однакових або різних хостах, і обидва вони пов'язані з двома виконавцями. На практиці можливості обробки операторів можуть бути певним чином покращені за рахунок збільшення кількості завдань, особливо коли даний оператор стає вузьким місцем всього потоку.

У той час як безперервне потокове передавання даних обробляється операторами системи обробки, екземпляр оператор може бути призначений або всьому набору, або частині даних. Стратегія групування визначає правило поділу даних для екземплярів операторів [31]. Загальна мета розбиття даних полягає в тому, щоб збалансувати навантаження оператора в пов'язаних з ним екземплярах. Іншими словами, доцільно рівномірно розподіляти дані, щоб уникнути неефективного використання ресурсів, тоді як дані обробляються паралельно на географічно розподілених випадках [32]. У Apache Storm було запропоновано кілька механізмів групування потоків. Розглянемо три з них, які зазвичай застосовуються в реальних сценаріях. Групування тасування забезпечує рівномірний розподіл кортежів потоку для завдань потоку. Кортежі призначаються круговим способом, щоб збалансувати навантаження, і найкраще це працює для атомних операцій, де не задіяні тимчасові стани. Хоча стратегія групування дозволяє рівномірно розподіляти дані без зміни семантики програми, вона не може визначити стани як ключі, які використовуються для збереження станів, ігноруються під час розділу даних. Групування полів виділяє дані на основі значення певного поля, і обіцяє, що кортежі, які мають однакове значення даного поля, завжди можуть бути розміщені в одному місці призначення. Зокрема, він застосовує хеш-функцію до ключів, а тому кортежі з одним і тим же ключем завжди направляються в одне і те ж місце призначення. Така схема зазвичай приймається для розрахунку по конкретних полях, таких як підрахунок слів або сортування. Хоча він зберігає стани і семантику застосунків, вона не зможе збалансувати навантаження в існуванні перекосу [33]. Наприклад, програма DSP, призначена для пошуку популярної теми твітів кожну годину, буде відчувати спотворені дані, оскільки вона повинна підрахувати частоти слів, в той час як розподіл відповідає

певному закону [34]. Часткове групування ключів працює так само, як і групування полів. Він присвоює кортежі на основі значення конкретного поля, при цьому рівень використання ресурсів буде враховуватися при прийнятті рішень про виділення [34]. PKG застосовує техніку "сили двох варіантів" і вводить розщеплення ключа. Зокрема, він висуває два вузли кандидатів, які отримані двома різними хеш-функціями для кожного ключа. Потім динамічно призначає дані одному з двох варіантів, який має найменше навантаження. Він фіксує доступність ресурсу і рівномірно розподіляє дані при існуванні перекосу. Однак деякі функції не можуть підтримуватися, оскільки він пересилає дані з однаковим ключем до двох вузлів.

1.4 Керування станами та обробка даних

Стани операторів, як правило, реалізуються у форматі пар «ключ-значення», а групи даних повинні передаватися ключем як ідентифікатор [35]. Більше стани управляються на ключовій основі, і це обов'язково потрібно для спрямування даних з тим же ключем до того ж місця призначення (або екземпляр операторів нижнього потоку). Apache Storm управляє, впроваджуючи стани в пам'яті і гарантує збереження. Кожного разу, коли задіюється система обробки автоматично створюється застосунковий потік для контрольно-пропускних пунктів повідомлень. З великою важливістю надійної обробки в DSP, очікується, що він досягне різного рівня надійності, торгуючи між витратами на обчислення та зниженням продуктивності. Apache Storm забезпечує три різних гарантійних рівні доставки даних, а саме не більше одного разу, принаймні один раз і рівно один раз [37]. Він відразу гарантує обробку кортежів в порядку надходження. Таким чином, кортежі падають тільки в тому випадку, якщо мережа або система не працюють. Хоча б один раз дозволяє переводити невдалі кортежі. В результаті кортежі можуть бути оброблені з початку або більше одного разу. Крім того, саме один раз працює тільки, як сторонній інструмент, який знаходиться на верхній частині служби Storm. Унікальний ідентифікатор потрібен, коли призначені кортежі для

відстеження та побудови кортежів. Структура дерева дає можливість розділити повідомлення на окремі компоненти. Отже, кортежі можна позначити, а пов'язаний з ними тайм-аут налаштовується. Невдала частка потоків може бути швидко отримана і відтворена менеджером топології, якщо це необхідно. Таким чином, цілісність повідомлень завжди може бути гарантована, якщо не очікується, що вона буде торгувати пропускною здатністю або пропускною здатністю проти точності обробки даних.

1.5 Керування ресурсами при обробці даних

Управління ресурсами в хмарному середовищі широко вивчалося протягом останніх десятиліть, в той час як великі масштаби і неоднорідність ресурсів центрів обробки даних, постійно змінювалось через навантаження і різні цілі, пов'язані з хмарними застосунками, і були вирішені як ключові проблеми [37]. Віртуалізація мереж і обчислювальних потужностей, як одного з основних чинників ефективного управління ресурсами, дозволяє швидко реагувати і гнучко реагувати [38]. Однак більшість існуючих методів управління ресурсами не застосовуються в DSP щодо динамічного характеру та довгострокового виконання вимог необмежених потоків даних. Розглянемо загальну практику управління ресурсами в хмарі та проблеми, які необхідно вирішити при розробці стратегій управління ресурсами для DSP. Також розглянемо роботу з планування та управління ресурсами для хмарних застосунків, а також застосунків обробки потоків. Загальна практика управління ресурсами в хмарі для проведення зростаючих великомасштабних обчислювальних завдань в областях, починаючи від науково-дослідних, медичних, фінансових та інженерних, обчислювальна інфраструктура Grid [38] і peer-to-peer (P2P) [39] систем в основному необхідні для обміну обчислювальними, мережевими та складськими ресурсами, географічно розподіленими між установами або організаціями [23]. Крім того, хмарні обчислення дозволяють забезпечити надійні обчислення для групи застосунків за допомогою віртуалізації ресурсів [24, 40]. Управління ресурсами в хмарі - це процедура розподілу ресурсів (наприклад,

обчислювальної техніки, зберігання, мереж) для застосунків, гарантуючи при цьому цілі постачальників послуг і користувачів [41]. Мета управління ресурсами, в цілому, складається з двох аспектів: 1) забезпечення високої продуктивності застосунків, що відповідають угодам про рівень обслуговування з точки зору користувачів; 2) забезпечити ефективне використання ресурсів для мінімізації витрат на технічне обслуговування від потенційних постачальників послуг [41]. Крім того, існують особливі вимоги щодо вартості енергії хмарних дата-центрів. Стверджується, що середня вартість енергії дата-центрів становить до 25 000 від персональних ПК [42]. Таким чином, фокус високопродуктивних обчислень вимагає компромісу між оптимізацією продуктивності системи та використанням базових ресурсів [17]. Типовим рішенням підвищення рівня використання ресурсів є консолідація хостів, яка об'єднує робочі місця в меншу кількість фізичних машин, тоді як інші машини можуть бути вимкнені для ресурсозбереження [43].

Цикл MAPE [44], який широко розглядався в автономній системі, дозволяє розділити всю процедуру управління ресурсами на чотири окремі етапи, а саме моніторинг (M), аналіз (A), планування (P) і виконання (E) [44, 45]. Для ефективного розподілу ресурсів на набір застосунків, система потребує ретельного моніторингу процесу для захоплення виконуваного середовища під час виконання. Що ще більш важливо, спостерігати різкі зміни навантаження спонтанно, а функції спостереження повинні бути ретельно розроблені, щоб точно і ефективно описати систему. Зібрані дані будуть використовуватися для аналізу з основною метою виявлення несподіваної поведінки або конкретних робочих моделей. Здійснюючи результати аналізу і заздалегідь визначену SLA, дії, які повинні бути прийняті, виводяться в результаті етапу планування. Зрештою, система буде проінструктована результатом етапу планування і спрямована на досягнення короткострокових або довгострокових цілей після обробки часових інтервалів. Стратегії управління ресурсами можуть бути розроблені для вирішення одного або декількох етапів циклу MAPE з конкретними інтересами користувачів або постачальників послуг. Кілька проблем були вирішені в управлінні хмарними ресурсами, включаючи неоднорідність типів ресурсів, різноманітне навантаження

застосунків і конкретні цілі, що стосуються різних аспектів [46]. Крім того, недостатнє забезпечення, надмірне забезпечення та коливання є загальними проблемами, які сьогодні є проблемами, що сьогодні мають хмарні менеджери [47]. З огляду на коливання навантаження хмарних застосунків, постачальники послуг, як правило, направили набагато більше ресурсів для застосунків, ніж фактичні потреби. Хоча він обіцяє, що послуга може бути надана в рамках обмежень SLA, забезпечення призводить до непотрібної втрати дорогоцінного обчислювального ресурсу. Обмеження ресурсів, які будуть підготовлені для застосунків (недопідготовка), може призвести до іншого рівня порушення SLA і навіть неприйнятної продуктивності, особливо коли стикаються з різким збільшенням запиту користувача. Іншим питанням є коливання як поєднання недоліків проблем з недостатнього резервування та надмірного резервування і викликане частим коригуванням ресурсів. Оскільки час, підсумований на кожному кроці, що бере участь у циклі MAPE, є непідготовленим, недостатньо часу, що залишився для певного етапу, може призвести до неефективного розподілу рішень. Очікується, що він зменшить частоту регулювання або введе період зарядки, що дозволяє системі адаптуватися до модифікацій, щоб полегшити небажані ефекти, введені коливаннями даних.

Загалом, для управління різними видами ресурсів у хмарному середовищі широко застосовуються два типи підходів, а саме реактивний та проактивний [97]. Хоча реактивний спосіб включає в себе відносно прості обчислення на етапі планування, проактивні рішення, як правило, зосереджуються на аналізі, а також на етапах планування. Зокрема, метод на основі порогу є одним з реактивних, які інструктовані конкретними правилами [48, 49, 50]. Умови і дії визначаються заздалегідь, і відповідні реакції будуть спровоковані, якщо під час виконання виконуються певні умови. Крім того, для відображення стану системи вибирається набір показників ефективності, а пороги визначаються для забезпечення автоматичного керування джерелами повторного живлення. Незважаючи на те, що порогові правила прості в реалізації і прості у виконанні з легкими обчисленнями, важко визначити відповідні метрики і визначити практичні пороги, оскільки

продуктивність сильно залежить від визначення порогів [51]. Зовсім недавно адаптивні правила пропонуються для заміни статичних порогових правил, що стосуються динамічного характеру хмарних аплікацій [52, 53].

Було запропоновано кілька рішень проактивно з урахуванням відсіків і проблем застосування реактивних підходів. Аналіз SIS полегшує прогнозування майбутньої тенденції або визначити моделі навантаження відповідно до зібраних даних [54, 55, 56, 57, 58]. За допомогою точного прогнозування продуктивності системи та вхідного навантаження, реакції можуть бути вирішені, щоб реагувати на зміну навантаження в рамках обмежень SLA проактивно. Крім того, теорія черг [59, 60] широко вивчається і розгортається на етапі аналізу та планування циклу MAPE для управління ресурсами. Зокрема, теорія черг дозволяє створювати математичні моделі, що стосуються архітектури системи, а також часовий розподіл даних, що надходять і обробляються. Кілька формул, також, доступні для моделей черг (наприклад, Закон Літл, Ален-Кунін), щоб допомогти з оцінкою часу очікування та обслуговування. Крім того, теорія управління допомагає проактивно автоматично керувати ресурсами в хмарі, визначаючи шахраїв з явними керованими входами і виходами [59, 60, 61, 62]. Це дозволяє самостійно регулювати, вводячи цикл зворотного зв'язку, і виправлення будуть зроблені після, порівнюючи вихід системи з цільовими значеннями. Подібно до теорії черг, вона вимагає ретельно моделювати систему і вибирати ключову особливість, щоб точно і ефективно відображати стан системи. Крім того, навчання з підкріпленням стає все більш обґрунтованим [63, 64, 65, 66, 67] у розробці рішень з управління ресурсами. Вона не вимагає попереднього знання системи і спрямована на автоматичне визначення оптимального конфігурування ресурсу. Тим не менш, він вимагає великих обчислень, особливо коли набір дій величезний. Найголовніше, оскільки етап навчання може тривати довго, продуктивність на початковому етапі відносно погана, а адаптивність сумнівна [68]. Як загальна практика розгортання DSP в хмарному середовищі, пропонується розділити DSP-застосунки на розподілені машини для масштабованості [69]. У контексті розподіленої потокової обробки даних стратегія управління ресурсами визначає протокол для розміщення

окремих операторів на даній обчислювальній інфраструктурі [12]. Тому важливо розробити ефективні стратегії управління ресурсами для оптимізації ефективності DSP. Показники продуктивності застосунків DSP зазвичай містять пропускну здатність і затримку, які вказують на обсяг даних, які система може споживати протягом певних проміжків часу, і час, витрачений на виконання даних [6, 36, 70]. Є деякі інші показники, які широко використовуються в інтернет-застосунках, включаючи доступність, що позначає співвідношення послуг в Інтернеті / оффлайн, і чуйність, яка показує, наскільки швидко система може реагувати на запити користувачів [71]. Фіксуючи такі метрики в режимі реального часу, рішення про розподіл ресурсів будуть отримані для різних цілей. Кінцевою метою управління ресурсами та планування є забезпечення доступності послуг та задоволення обмежень SLA, але необхідно вирішити ряд проблем планування в DSP, включаючи використання ресурсів, управління даними (локальність даних, стратегія реплікації та розміщення), справедливість, відмовостійкість та енергоефективність [72]. Крім того, може виникнути конфлікт з деякими цілями на практиці [6]. Таким чином, це складно розробити загальні рамки для застосунків в різних областях. Наприклад, балансування навантаження між активними запущеними хостами дозволяє уникнути проблеми недостатньої або надмірної підготовки ресурсів, однак стратегія справедливості може призвести до серйозної деградації системи або призвести до втрати ресурсів, якщо різна модель трафіку або попит на ресурси в окремих застосунках DSP.

Планувальник DSP визначає принцип розміщення завдань на робочих вузлах і розглядається як основний компонент дизайну DSP. Індивідуальний планувальник може бути реалізований і розгорнутий в Nimbus, і він регулярно запускається в налаштованому інтервалі або кожного разу, коли користувачі надсилають нову топологію. При еквівалентній кількості підготовлених ресурсів продуктивність програми може бути змінена, і вона може бути сильно відрізнятися після застосування різних стратегій планування. Стратегія планування за замовчуванням Apache Storm призначена для розподілу виконавців між робочими процесами та рівномірного зіставлення робочих процесів з вузлами. Це статичний

підхід, що дозволяє легко бути реалізованим. Він застосував кругову стратегію (RR), яка в першу чергу спрямована (1) на справедливий розподіл робочих процесів між доступними фізичними машинами, і (2) збалансування використання ресурсів між активними хостами. Хоча очікується рівномірний розподіл робочого навантаження на вузли та процеси, така політика не враховує динамічну зміну обчислювальних можливостей та потреб у ресурсах. Як наслідок, це може призвести до недостатнього використання ресурсів і навіть викликати серйозну деградацію продуктивності. Наприклад, планувальник за промовчанням має на меті розподіл працівників на доступних машинах, тоді як деякі з них можуть бути об'єднані до меншої кількості вузлів для максимізації рівня використання ресурсів. Тому, стратегії планування в таких системах є важливими і потребують більш детального дослідження.

1.6 Схеми планування

Було запропоновано два широких підходи в автономному режимі або в Інтернеті для розробки схем планування в системах обробки розподілених потоків. Хоча автономне планування в основному відноситься до статичної структури застосунків DSP, він досліджує паралельний розділ і залежності даних для даної топології. Наприклад, ключовою метою офлайн-планувальника, запропонованого Аніело та ін. [72] є зниження витрат на зв'язок і, відповідно, скорочення затримки обробки. Однак, такий метод автономного планування завжди приймає планування рішення перед виконанням DSP застосунків. Хоча він уникає введення накладних витрат на виконання DSP, його обмеження швидко виявляється, оскільки він не адаптується до різних умов руху даних під час виконання. Крім того, неоднорідність хмарних ресурсів не вирішується при розміщенні застосунків DSP, що може призвести до різноманітного рівня порушень QoS. Тому, більшість останніх робіт [73-92] запропоновані для обробки орієнтовані на онлайн-підходи для вирішення проблем планування. Онлайн-планувальники враховують значення постійно мінливого попиту на ресурси та обчислювальної потужності. Крім того,

вони розробляють картографічний зв'язок між операторами DSP та основними обчислювальними ресурсами щодо динамічного характеру DSP. Оскільки методи, згадані в загальній практиці управління ресурсами в хмарних застосунках, аналіз часових рядів, теорія черг, теорія управління та навчання підкріпленням також застосовуються в онлайн-планувальниках для DSP. Стратегії планування, обізнані з використанням ресурсів, пропонуються для фіксації динамічної зміни робочого навантаження та відповідного попиту на ресурси з головною метою як оптимізація рівня використання ресурсів. R-storm [93], наприклад, реалізує схему планування з інформуванням про ресурси в Storm щодо обмежень процесора та пам'яті. Це також відноситься до відстані мережі між підключеними компонентами та різноманітністю типів ресурсів, які були враховані при прийнятті рішень щодо планування. Два кроки, пов'язані з процесом планування, вибирають завдання і вузол для розміщення конкретного завдання. R-storm досягає збільшення пропускної здатності та використання процесора в порівнянні з Storm [94]. Аналогічним чином, T-Storm застосовує онлайн-підхід, щоб дозволити динамічне коригування параметрів розкладу, щоб забезпечити виконання з меншою кількістю робочих вузлів при прискоренні обробки даних. Він також розглядав моделі руху під час виконання. Хоча він не гарантує відмовостійкість обробки потоку, але він досягає кращої продуктивності в оптимізації рівня використання ресурсів [93], ніж планувальник Storm.

Для подальшої оптимізації ефективності системи було запропоновано набір проактивних стратегій для підвищення пропускної здатності та затримки. Зокрема, було запропоновано онлайн-підхід, ввівши контролер зворотного зв'язку для контролю часу очікування даних у черзі на основі відсотка даних, які оброблялися протягом терміну. Він був спрямований на оптимізацію кількості віртуальних машин, які виділені для конкретного потоку даних, а також стосується обчислень шляхом надання конкретного протоколу міграції. Модель предикативного контролю дозволяє вивчити конфігурацію системи щодо кількості ядер процесора і частоти динамічно. Крім того, рішення на основі контролю є основним вирішенням недостатньої та надмірної підготовки ресурсів у кластері Storm та

постійного руху статусу системи для своєчасного коригування. Що стосується розділу даних, то пропонується онлайн-стратегія для автоматичного вивчення паралельного рівня даної топології на основі виміряного статусу заторів та пропускної здатності, які відображають використання ресурсів у режимі реального часу та навантаження на програми.

Таким чином, врахування схем планування при обробці потоків даних є досить актуальним.

1.7 Постановка задачі дослідження

В результаті аналізу проблематики предметної області дослідження необхідним і актуальним є продовження досліджень щодо розподілу обчислювальних ресурсів для обробки розподілених потоків даних.

Результатом таких досліджень має бути метод для усунення недоліків попередніх використовуваних методів і стратегій.

Для досягнення поставленої мети необхідно:

– розробити модель затримки оператора;

– розробити модель пропускної здатності системи з врахуванням особливостей проаналізованих недоліків в цій предметній області;

– здійснити моделювання системи та результати експериментів з моделлю;

– розробити метод та алгоритми розподілу обчислювальних ресурсів для обробки розподілених потоків даних;

– здійснити реалізацію запропонованого рішення.

1.8 Висновки до першого розділу

За результатами проведеного аналізу щодо розподілу обчислювальних ресурсів для обробки розподілених потоків даних було встановлено межі предметної області дослідження, визначено поняття потокової обробки, а також

виділено недоліки у відомих рішеннях щодо організації кластеру та його складових компонентів, стратегіях розподілу та групування даних, керуванні станами та обробка даних, керуванні ресурсами при обробці даних та особливостях схем планування, що впливають на досягнення ефективності.

2 МОДЕЛЬ СИСТЕМИ

2.1 Модель затримки оператора

Невизначеність і складність потокових даних ускладнює стійкість поведінки системи, і, отже, створює проблеми для ефективного управління основними ресурсами. Щоб забезпечити еластичність ресурсів застосунків обробки потоку, пропонується модельний метод автоматичного масштабування, який фіксує поведінку системи виконання та профілі окремих операторів для цілей розміщення. Необхідно також впровадити загальні математичні моделі для оцінки продуктивності систем обробки потоків (тобто затримки та пропускної здатності) при різних умовах. Розроблений метод дозволяє проводити динамічну оцінку системи через затримки і приймає до уваги зовнішні змінні (наприклад, швидкість передачі даних і доступність ресурсів). В основі розробленого методу є модель обробки, яку розглянемо детальніше.

Еластичність ресурсів, як критична особливість хмарного середовища, інтерпретує здатність системи регулювати розподіл ресурсів відповідно до постійно мінливого попиту на аплікацію [40, 81]. Загалом, очікується надання застосункових ресурсів в момент значного навантаження, а також вивільнення певної кількості ресурсів з запитів користувачів на мінімізацію економічних витрат [94]. Автоматичне масштабування - це метод, запропонований для забезпечення еластичності ресурсів шляхом динамічної модифікації розподілу ресурсів на хмарні програми [71]. Він спрямований на те, щоб гарантувати рівень QoS шляхом автоматичного масштабування кількості ресурсів залежно від навантаження застосунків за допомогою втручань користувача. Однак неефективна реалізація цього процесу в результуючому підсумку призведе до недостатньої підготовки або надмірного забезпечення ситуацій [93]. Система не зможе гарантувати вимоги QoS, якщо ресурсів, підготовлених для цільових застосунків, недостатньо, і вона може навіть зіткнутися з серйозною деградацією продуктивності та призупинити виконання через відсутність обчислювальної потужності. На відміну від цього, підтримка набагато більше ресурсів, ніж фактична вимогливість, призводить до

величезних економічних витрат, хоча певний рівень надмірної підготовки бажаний для подолання випадкових коливань навантаження. Система іноді відчуває коливання, які з'єднують недо- і надмірні підсилення, коли частота регулювання досить мала, або є затримка між зміною робочого навантаження та застосованими повторними конфігураціями [25, 92].

Очікується, що ефективний авто масштаб буде незалежним від характеристик робочого навантаження, адаптивним до динамічного середовища потокового передавання даних і надаватиме перевагу обміну ресурсами в хмарних екосистемах [71]. Тому, пропонується залучити MAPE [91,92] до проектування методів автоматичного масштабування, які вимагають чотирьох процесів, а саме моніторингу, аналізу, планування та виконання. Примітно, що застосування циклу MAPE в обробці потоку зазвичай починається з системного моніторингу, який має основні функції системи через певні проміжки часу. Особливості повинні бути ретельно підібрані, щоб вони могли точно описати поведінку системи і допомогти з рішеннями масштабування. Потім авто масштабувальник виконує аналіз і прогнозування на основі дані, зібрані з монітора під час другого етапу. Оскільки навантаження є основним показником попиту на ресурси [72], прогноз коливання навантаження необхідний для досягнення високої продуктивності в хмарних застосунках [83]. На жаль, етап планування і виконання вирішує розподіл ресурсів і проведення дій за вказівкою отриманих рішень. Оскільки еластичність в обробці потоку зазвичай відноситься до гнучкості розміщення операторів [40], авто масштабувальник в DSP в основному стосується рівня паралелізму операторів і стратегій розміщення для екземплярів. Також передбачається, що етап планування має істотне значення при проектуванні авто масштабувальник [91], який визначає протокол розміщення операторів в авто масштабувальник DSP. Основна мета масштабування повинна бути вирішена на цьому етапі, щоб гарантувати, що отримані рішення можуть досягти поставлених цілей.

Однією з ключових проблем застосування циклу MAPE при автоматичному масштабуванні для DSP є розробка точних моделей системи. Моделі повинні бути досить загальними, щоб описати поведінку системи, що стосується динамічних

особливостей (наприклад, навантаження, ресурс). Більше того, очікується, що ефект коригування певних умов можна кількісно оцінити просто. Теорія черг була широко вивчена в машинобудуванні і в основному введена для скорочення часу очікування клієнтів або збільшення виробничих показників [45, 88]. Тому, було запропоновано кілька моделей черг [41, 43, 45, 92] для формулювання обробки потокових даних. Зокрема, моделі M/M/ припускають, що час надходження та обробки даних слідує за експоненціальним розподілом. Вказується, що кілька каналів подаються для передачі даних, що дозволяє паралельно виконувати завдання. Оскільки швидкість передачі даних і час обробки можуть бути змінені під час виконання, і він не строго дотримується експоненціального розподілу, моделі M /M/ застосовуються лише в обмежених випадках. Крім того, були запропоновані моделі G/G/1 [41,43,45,92] для більш загальний підхід, який послаблює обмеження розподілу часу надходження та обробки даних. Однак моделі G/G/1 не змогли врахувати рівень паралелізму обробки потоків, оскільки моделі, імовірно, подаються з одним сервісним каналом. Що стосується паралелізму DSP на різних рівнях (наприклад, операторів, потоків, застосунків), то бажано моделювати DSP як СМО з декількома каналами. Тому, на відміну від більшості існуючих рішень, які або вимагають знань про процес прибуття та запитуваного часу обслуговування або обробки потоку моделі як СМО з одним каналом, визначаємо DSP як моделі G / G / . Якщо б не було б жодного припущення щодо моделі розподілу часу між прибуття або обслуговування потокових даних, то це дозволяє будь-яку кількість каналів, збережених для цілей обробки. Крім того, розробляємо модель для оцінки пропускної здатності системи, враховуючи навантаження і ресурсну ємність. Завдяки цим двом математичним моделям, пропонується авто масштабувальник з циклом MAPE і прийняття рішення масштабування, посилаючись на оцінку продуктивності та вимоги QoS. Точність моделей затримки і пропускної здатності виправдані перед оцінкою ефективності авто масштабувальника, а авто масштабувальник оцінюється за програмами DSP з різним розташуванням операторів.

Таким чином, для подальшого розвитку моделей в системі вирішення цієї задачі аналізуватимемо відповідні методології, які були застосовані для забезпечення автоматичного масштабування, і в першу чергу зосереджуємося на контексті обробки потоків. В подальшому розроблятимемо запропоновані загальні математичні моделі для оцінки ефективності DSPs з точки зору пропускної здатності та затримки. Модельний метод автоматичного масштабування вводиться за підтримки циклу MAPE, і як пропускна здатність, так і затримка розглядаються як керовані фактори для розробки рішень масштабування. Підсумком роботи стане оцінка точності прийнятих та запропонованих рішень в сукупності.

2.2 Модель пропускної здатності системи

Щоб забезпечити еластичність в системах DSP, використаємо проактивний підхід до автоматичного масштабування на основі теорії черг. Оскільки автоматичне масштабування в DSP в основному відноситься до розміщення оператора [40], авто скалятор призначений для розробки ефективного розміщення щодо навантаження застосунків і динамічного реагування на зміну попиту користувачів. На відміну від попередніх досліджень [41, 43, 45, 92], змоделюємо оператори DSP як черги G/G/, в яких час між прибуттям і час обробки мають загальний розподіл і доступні кілька каналів обслуговування. Зокрема, кілька серверів можуть бути збережені для окремих операторів, щоб дозволити різноманітний рівень паралелізму. Крім того, модель затримки приймає робоче навантаження і обчислювальну потужність в якості вхідних даних та оцінює час очікування за формулою Аллена-Кунніна (A-C) [19]. Тим часом, модель пропускної здатності розробляється шляхом розгляду вхідного трафіку і ресурсного потенціалу, підготовленого для даної програми. Ці дві моделі фіксують поведінку системи під час виконання та виконують оцінки, щоб забезпечити активне коригування розміщення для операторів.

Затримка системи описує час, витрачений на обробку кортежів даних. Визначаємо затримку явно як весь час перебування кортежу даних у системі,

включаючи час виконання та очікування. Зокрема, час виконання в першу чергу залежить від ємності хоста, і він залишається незмінним, якщо базова обчислювальна потужність залишається незмінною. Тим не менш, час очікування може бути надзвичайно різноманітним, коли потік відчуває навантаження. Щоб забезпечити динамічне налаштування ресурсу при різному навантаженні, зосереджуємося на оцінці часу очікування та вирішуємо затримку кортежу, яка страждає в черзі окремих операторів. Теорія черг широко застосовується для дослідження часу очікування і часу поступки в комп'ютерних системах [41, 88]. Загалом, модель визначається трьома факторами A/B/C, в яких A і B представляють часовий розподіл даних і час обслуговування, а C позначає кількість каналів обслуговування [67]. У той час як M представляє собою експоненційний розподіл A або B, то G показує, що час прибуття (A) або час обслуговування (B) слідує загальному розподілу. Оскільки очікується, що модель затримки описує загальну поведінку системи без вказівки часу надходження та обробки даних, то моделюємо кожен оператор DSP як чергу G/G/. Іншими словами, існує загальний розподіл як між прибуттям, так і часом обслуговування даних в операторах, в той час як може бути кілька каналів, підготовлених для обслуговування виконання даного оператора. На рисунку 2.1 показаний застосунок DSP з чотирма операторами, які послідовно перетинаються. Вихід оператора o розглядається як введення даних оператору a, і тому час між прибуттям оператора a інтерпретується як час отриманих даних від оператора o. Крім того, оператор a моделюється як черга з трьома паралельними каналами (тобто = 3) і його вихід спрямований на оператор b. Легко помітити, що обробка може бути потенційно прискорена за рахунок збільшення паралелізму. певних операторів, особливо якщо оператор є досить вузьким місцем.

Формула Аллена-Кунніна (A-C), вперше описана в [4], може бути застосована для приблизної оцінки системного часу даних у черзі із загальним розподілом як часу прибуття, так і часу служби [19, 67]. Примітно, що формула A-C була розроблена з використанням обчислювальних методів оцінки без будь-яких формальних доказів.

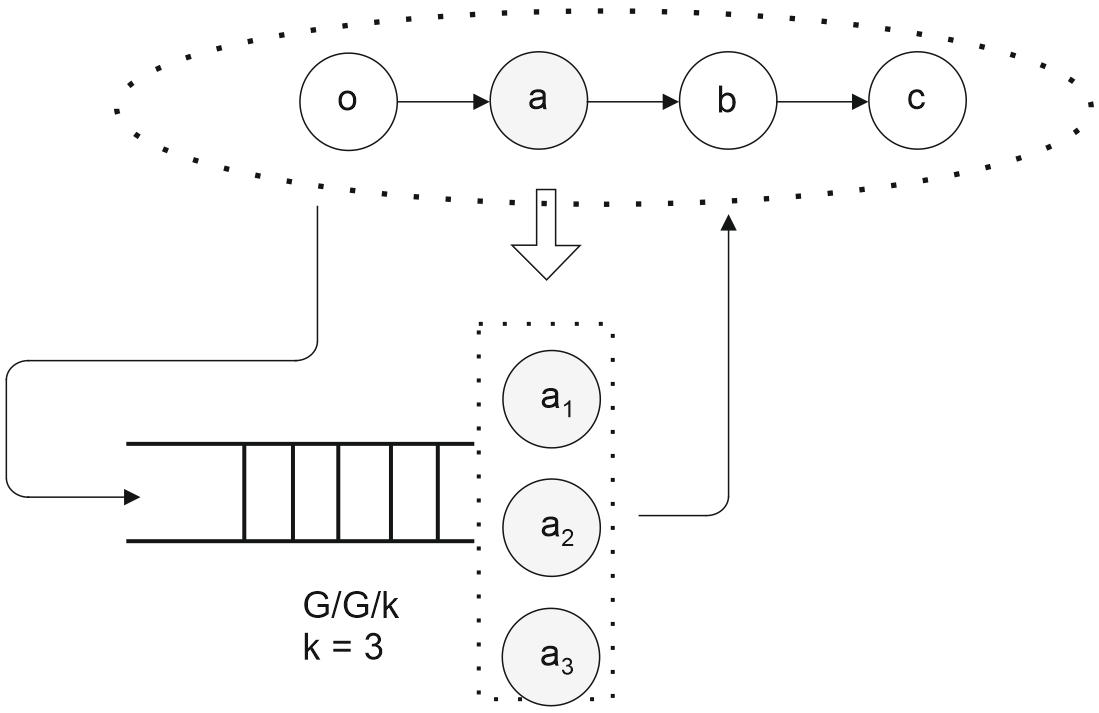


Рисунок 2.1 – Модель черги G/G/k

Тим не менш, це часто забезпечує відмінне наближення до середнього часу очікування клієнтів у чергах G / G /. Звіт Tanner підтверджує, що результати, отримані за формулою A-C, були в межах 10% від фактичних значень у більшості сценаріїв [29]. Таким чином, використовуємо наближення Аллена-Кунніна для оцінки часу очікування кортежу даних у черзі оператора. Введемо позначення, що використовуються при визначенні моделі затримки оператора: μ - середня швидкість обслуговування кортежу даних; ТА - середній час прибуття кортежу даних; ТВ - середній час обслуговування кортежу даних; λ - середня швидкість обслуговування кортежу даних; ρ - використання СМО; $^{M\mu}$ - розрахункова затримка оператора. Нехай ТА і ТВ будуть середнім часом між прибуттям і середнім часом обслуговування кортежів, тоді середня швидкість прибуття λ і швидкість обслуговування μ розраховуються як взаємність кортежів, тоді середня швидкість прибуття λ і швидкість обслуговування μ розраховуються як взаємність ТА і корелюють відповідно. Крім того, рівень використання ρ черги з декількома каналами може бути обчислений за допомогою формули (2.3).

$$\lambda=\frac{1}{T_A} \tag{2.1}$$

$$\mu=\frac{1}{T_B} \tag{2.2}$$

В даному випадку M представляє кількість каналів для обробки даних [67]. Промоделюємо окремий оператор як чергу G/G/замість черги G/G/1, оскільки кілька каналів можуть точно описати підтримку паралельної обробки даних в операторах DSP. Збільшення кількості каналів M може знизити рівень використання СМО. Це також правильно в контексті потокової обробки даних, оскільки трафік оператора може бути потенційно пом'якшилися, маючи застосункові екземпляри.

$$\rho=\frac{\lambda}{M_\mu} \tag{2.3}$$

Отримані таким чином математичні моделі надають змогу здійснювати моделювання системи та обробляти експерименти з моделлю для оцінювання точності запропонованих варіантів розв’язання задачі.

2.3 Моделювання системи та результати експериментів з моделлю

Визначатимемо пропускну здатність, оскільки загальна кількість кортежів може оброблятися через певний інтервал, і тому пропускна здатність топології обчислюється з оцінок пропускної здатності для окремих операторів. Крім того, модель пропускної здатності, побудована для оператора, вимагає швидкості походу даних як входу для відображення динамічної зміни робочого навантаження, яку відчуває даний оператор. Здійснюватимемо ідентифікацію оціночної величини за допомогою окремого символу, щоб відрізнити його від спостережуваного стану системи. Нехай це рівень паралелізму (кількість екземплярів) оператора, тоді пропускна здатність оператора - це загальна кількість кортежів, оброблених усіма екземплярами. Розбиття даних дозволяє підвищити пропускну здатність в DSP, оскільки екземпляр не є локально реалізованим як єдиний потік. Маючи кілька потоків, обсяг даних може бути оброблений протягом фіксованого інтервалу, може

бути значно покращений, особливо при розміщенні потоків у географічно розподілених хостах. Потім розрахункова пропускна здатність оператора, позначена як $\bar{R}$, буде виведена шляхом додавання чисел у всіх екземплярах. Крім того, існують два основних фактори, які визначатимуть пропускну здатність екземпляра оператора: (1) обсяг даних, що надійшли, і (2) обчислювальна здатність цільового хоста. Іншими словами, обсяг даних, що надходять до екземпляра оператора, обчислює очікувану кількість кортежів, які будуть оброблені даним екземпляром оператора, тоді як базова ємність обмежує максимальну кількість кортежів, які можуть бути оброблені протягом наступного проміжку часу. Нехай U перераховує весь набір операторів, тоді загальна кількість кортежів, які, як очікується, надійдуть у екземплярі оператора h протягом наступного проміжку часу, позначається як $\bar{A}$. Розподіл кортежу даних між екземплярами є відносно стабільною, а дисперсія розподілу ключів у потоках даних з плином часу з обсягу цієї роботи. На рисунку 2.2 показаний приклад сценарію, в якому оператор c має два оператори (тобто a і b), а рівень паралелізму - три. Якщо припустити, що дані, перетворені з оператора a, спрямовані лише на перший екземпляр c_1, тоді як b рівномірно віддає належне його виходу іншим двом екземплярам c_2 і c_3. Тоді $d(a, c_1) = 1$ і $d(b, c_2) = d(b, c_3) = 1$. Отже, очікувана кількість кортежів, що надходять в c^1, оцінюється як $\bar{R}\bar{T}_a$, і $1 \cdot \bar{R}\bar{T}_b$ як для c_2, так і для c_3.

Розрахункова сума передачі оператора висхідного потоку використовується замість розрахункової пропускної здатності $\bar{R}$, оскільки в деяких випадках оброблені кортежі даних повинні бути агреговані перед передачею. Отже, оператор нижнього потоку отримує менший обсяг даних, ніж обсяг даних, що випускається рівнем вище. Однак співвідношення між RT і R узгоджується з ідентичним розподілом ключа в потоках з плином часу. Нехай O перераховує колекцію операторів у даній топології, то загальна пропускна здатність в кінцевому підсумку оцінюється шляхом додавання пропускної здатності окремих операторів. У міру розгляду обсягу передачі даних оцінка загальної пропускної здатності повинна починатися з першого оператора, який отримує дані із зовнішніх джерел і виконує оцінку ітераційно, поки кожен оператор не буде вивчений.

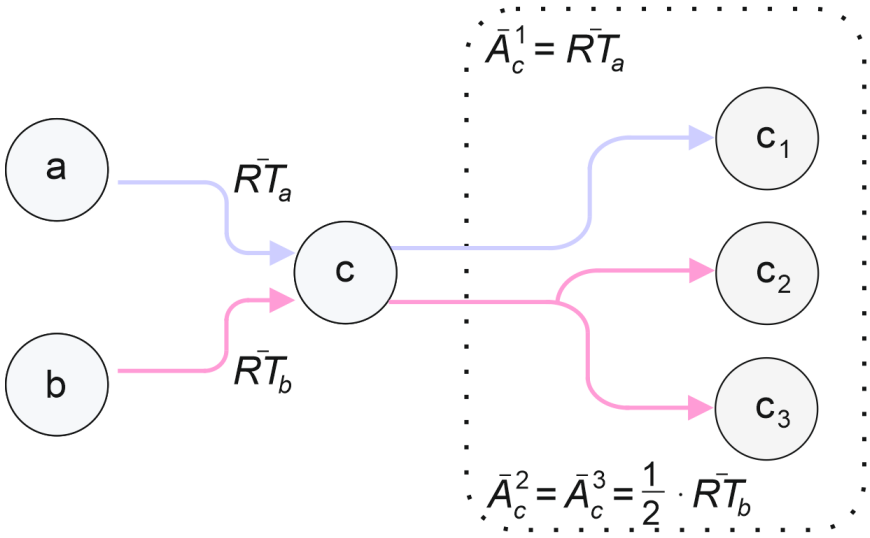


Рисунок 2.2 – Приклад оцінки обсягу даних прибуття для екземпляра оператора

Основною метою автоматичного масштабування є динамічне коригування стратегії розміщення операторів щодо постійно мінливого попиту на ресурси. Грунтуючись на загальних моделях, запропонованих для пропускної здатності і затримки, розглянемо модельний автомасштаб, щоб вирішити розміщення оператора для даної топології з урахуванням різної завантаженості та обчислювальної потужності.

Оскільки однією з основних вимог автомасштабу є незалежність від моделей навантаження та адаптації до постійно мінливого попиту на ресурси, пропонується ввести MAPE у розробці процедури автоматичного масштабування [91]. Цикл MAPE складається з чотирьох етапів, і він розпочинається з монітора, який працює як фонова служба. Решта етапів в циклі – це аналіз, планування і виконання. Більш конкретно, процес моніторингу, по суті, необхідний для захоплення поведінки системи під час виконання. DSP–застосунки, як правило, абстрагуються як DAG, де як вершини, так і краї мають критичний вплив на обробку, і, отже, процес моніторингу повинен збирати інформацію щодо 1) часу виконання операторів; 2) обсяг даних, що обробляються операторами; 3) обсяг даних, що передаються між кожною парою операторів. Розміщення операторів також збирається для

інтерпретації обчислювальних потужностей, передбачених для кожного оператора. Крім того, коефіцієнти зміни часу між прибуттям та обробкою включаються, оскільки вони приймаються як вхідні для оцінки часу очікування. Щоб увімкнути оновлення стану, етап моніторингу реалізується на основі способу розсувного вікна. Моделі пропускної здатності та затримки, розглянуті в попередньому підрозділі, потім застосовуються на етапі аналізу. Іншими словами, система виконує оцінки пропускної здатності та затримки, приймаючи дані, зібрані з монітора, як введення моделі. Наприклад, на рисунку 2.3 зображені перші дві фази, монітор і аналіз запропонованого авто масштабувальник на основі моделі. Він показує оцінку затримки для оператора a. Збираємо час між прибуттям та час обслуговування під час моніторингу фази, час очікування, а також загальну затримку, яку оператора можна оцінити на етапі аналізу. Він схожий на механізм оцінки пропускної здатності. Потім результати оцінки доставляються на етап планування, на якому буде визначено рішення про розміщення.

Для подальшого підвищення точності результатів запропонованих моделей впроваджена процедура само реконфігурації для дослідження зовнішніх порушень. Без великих обчислювальних затримок система дізнається про помилку оцінки з виходу системи в той час і розглядає цю помилку як порушення в часі. Потім фаза планування визначає оптимальне рішення для повторного виділення операторів на існуючі вузли після оцінки результатів оцінки відповідно до цільової метрики ефективності. Вузол і хост, який служив меті виконання, можна обміняти в нашому дослідженні. Алгоритми, які будуть обговорюватися в цьому розділі, є основними компонентами етапу планування. Зокрема, є два кроки, пов'язані з плануванням. Спочатку ми об'єднуємо хости відповідно до їх підготовлених ресурсів (наприклад, ядер процесора та пам'яті). Враховуючи, що попит на ресурси в застосунку DSP може варіюватися від оператора до оператора, він спрямований на призначення операторів вказаному типу хостів, який відповідає його вимогам. Іншими словами, перший алгоритм визначає розмір хоста для кожного оператора, посилаючись на цільову пропускну здатність і результати оцінки при розміщенні його в інший тип хостів. Тоді обраний тип хостів для кожного оператора можна розглядати як

керівництво для інструктаж розміщення для окремих операторів. Проблему вузького місця, з якою будь-який оператор може зіткнутися під час виконання, замовивши операторів відповідно до їх передбачуваної затримки. Він завжди спрямований на визначення пріоритетів операторів, які відчувають більш тривалу затримку. На другому кроці вводиться інший алгоритм для вибору певного хоста для кожного екземпляра оператора щодо доступності ресурсів і запропонованого типу (з кроку 1). Виводом цього алгоритму є зіставлення між екземплярами операторів і запропонованими хостами. Зрештою, останній етап циклу MAPE (тобто фаза виконання) полягає у виконанні фактичних дій перепланування відповідно до зв'язку відображення, отриманого з фази планування. В результаті розглянуто поширену неоднорідність хмарних систем у прийнятті рішень щодо автоматичного масштабування та розміщено оператори для хостів, які мають точну здатність служити цілі виконання. Крім того, динамічний характер DSP вказує на невизначеність і складність щодо моделей надходження даних. В результаті попит на ресурси окремих операторів з часом може коливатися. Включені пропускна здатність і затримки моделей, різниця між передбачуваним результатом і очікуваним стандартом може бути захоплена під час виконання. Мета авто масштабу полягає в тому, щоб звести до мінімуму такий розрив для всього набору операторів, уникаючи при цьому втрати ресурсів. Розглянемо детальніше описаний алгоритм 1. Він описує механізм знаходження типу хоста кандидата для оператора. Мережеві хости, які в основному служили для виконання завдань, класифікуються відповідно до їх обчислювальних можливостей (наприклад, ядра процесора та пам'ять). Даний алгоритм передбачає надання доступних категорій для хостів та визначає пов'язані з ними ресурси. Потім визначається бажаний рівень пропускної здатності для оператора. Хоча рівень пропускної здатності цілі може бути налаштований користувачами, визначаємо потрібну пропускну здатність як вихід системи при виконанні даної програми з достатнім запасом ресурсів. В результаті деградація пропускної здатності буде в першу чергу викликана обмеженням потужностей, що зберігаються для окремих операторів. Потім цей алгоритм перевіряє оператора і профілів його, призначається певний тип хоста. Вона

повертає тип хоста кандидата, вибраний для оператора, і позначає результат. Час виконання розглядається як ключовий фактор для оцінки пропускної здатності екземпляра оператора. Таким чином, інформація про ємність хоста по суті потрібна в функції пропускної здатності. Абсолютне значення різниці фіксується, щоб потенційно уникнути втрати ресурсів, вибравши хости, які мають точну здатність для задоволення вимог обробки. Мінімальне відхилення, а також відповідний тип, записуються на кожному раунді ітерації. Зрештою, кандидат повертається для інформування авто масштабувальник з вибраним типом хоста. Зверніть увагу, що кандидат залишається таким, яким він є, якщо існуючий тип вже є оптимальним варіантом. В результаті алгоритм відображає оператор до хоста типу кандидата, потужність якого тісно пов'язані з попитом на обробку. Очікується, що такий хост оброблятиме потоки даних на вхідному трафіку на цільовому рівні. Найголовніше, що потреби перераховуються щоразу, коли потрібно прийняти рішення, оскільки швидкість по прохання даних операторів може змінюватися з плином часу, інструктований обраним типом хостів для операторів. Другим кроком на етапі планування є розміщення операторів топології в хости з усвідомленням їх затримки. Зокрема, процедура розміщення вимагає збору результатів, отриманих з алгоритму 1. Нехай O позначає весь набір операторів у даній топології T, а H_T підтримує тип кандидатів хостів для операторів в O. Крім того, списки доступних хостів, а також категорій, до яких вони належать відповідно до їх пов'язаної потужності. Є ще дві змінні C і P, які вказують на існуючі і запропоновані схеми розміщення відповідно. Обидва вони зберігають дані як ключове значення пари, в яких ключ ідентифікує екземпляр оператора, а значення - ідентифікатор хоста. Щоб визначити цільовий хост для конкретного оператора, система повинна бути проінформована про рівень використання ресурсів кожного активного хоста щодо фактичного навантаження T. Функціональний ресурс визначається для надання можливості отримання доступності ресурсу, і для цього потрібні два параметри: набір залучених хостів і запропонована схема розміщення. Припускаючи, що фонова служба продовжує працювати, щоб зібрати останній статус кожного хоста в A, система не зможе зафіксувати нещодавно запропоновану зміну розміщення,

оскільки фактичний повторний розподіл ще не виконується. Однак хост був обраний для екземпляра оператора теоретично зайнятий, і, отже, він більше не доступний для розміщення інших екземплярів. Таким чином, очікується, що оновлення доступності ресурсу для довідки кожного разу, коли необхідно прийняти рішення для забезпечення ефективності пропонованого розміщення. Крім того, оператор, оснащений значними ресурсами, не може повністю використовувати свої підготовлені ресурси, якщо один з його операторів відчуває затримки під час виконання. Для уникнення цього сортуємо розрахункову затримку операторів у порядку зменшення. В результаті розміщення для операторів завжди визначає пріоритети операторів, які відчувають більш високу затримку.

Алгоритм 2 передбачає детальні процедури розміщення операторів для топології T. з усвідомленням затримки. Весь набір операторів O топології T, доступні хости A та номінований тип хоста для операторів H_T необхідні для розміщення. Процедура розміщення операторів ітерує оператори і замовляється відповідно до їх затримки. Під час кожної ітерації операторів обраний тип хоста кандидата може бути отримана з виходу алгоритму 1. Отже, ідентифікатор екземпляра, а також отриманий хост кандидата записуються на P як нещодавно запропонована схема. Метод поверне null, якщо такого хоста немає з достатньою доступністю даного типу. В результаті існуюче розміщення буде застосовано до запропонованої схеми, щоб уникнути вартості перенастроювання. Крім того, доступність ресурсів повинна своєчасно реагувати на зміну розміщення, як обговорювалося вище, щоб уникнути потенційних конфліктів розміщення інших екземплярів оператора в хост, поки він був повністю зайнятий. Таким чином, метод викликається відразу після зміни розміщення.

Точність оцінки пропускної здатності та моделей затримки з огляду на різноманітне розташування операторів може мати застосунок DPS і відповідну дисперсію щодо моделей трафіку в застосунку. Розглядатимемо три основні структури і будь-яке складне застосування DSP може бути розкладене на такі форми. Крім того, впроваджена топологія, яка читає твіти в режимі реального часу та обчислює трендові теми протягом певних часових інтервалів, щоб ще більше

виправдати ефективність запропонованого авто масштабувальник на практиці. Оцінка моделей ілюструє узгодженість результатів оцінки з виходом системи. По суті, пропускна модель має менше 1% відхилень порівняно з пропускною здатністю системи. У той час як модель затримки має не більше 1,65%, 1,76% і 3,12% відхилень від затримки системи трьох топологій відповідно. Крім того, запропонований авто масштабувальник оцінюється відповідно до схеми планування за замовчуванням Apache Storm щодо пропускної здатності та затримки. Результати показали, що макет DSP застосунків відіграє значну роль у продуктивності системи. Програми мають різноманітний рівень підвищення продуктивності від запропонованого авто масштабу, і додатково, як усвідомлення затримки оператора, це може допомогти підвищити загальну продуктивність системи, порівнявши його зі стратегією без сортування операторів за їх спостережуваною затримкою.

Для експерименту цієї частини роботи було виділено шість вузлів супервайзера, які класифікуються на три групи, а кількість ядер процесора, обсяг оперативної пам'яті і кількість вузлів кожної категорії показані в таблиці 2.1. Крім того, системний монітор реалізований за підтримки API REST, що надаються інтерфейсом Storm [8]. Зокрема, є три набори експериментів, які були проведені з метою дослідження:

– для оцінки точності оцінки щодо пропускної здатності і затримки;

– оцінки ефективності запропонованого авто масштабування - кільця до різноманітних структур застосунків DSP;

– оцінки ефективності запропонованого авто–масштабувальника щодо роботи з даними в режимі реального часу.

Перші два набори експериментів були виконані на трьох мікро–бенчмаркових застосунках DSP і були спрямовані на запис рівня затримки і пропускної здатності протягом приблизно однієї години. Третій експеримент обійшов задану топологію з даними в режимі реального часу, що потрапляють в неї протягом п’яти хвилин. Пропускна здатність і затримка захоплені і порівнюються з існуючою стратегією Storm.

Таблиця 2.1 – Ресурси та кількість вузлів, підготовлених для кожного типу

Клас	ЦП - кількість ядер	Оперативна пам’ять, GB	Номер типу топології
1	8	32	1
2	12	48	2
3	16	64	3

Три топологічні форми для потокових застосунків використовуються і реалізуються як топології Storm для оцінки. Оскільки будь-які складні програми DSP можуть бути додатково розкладені на ці основні структури, результати оцінки дозволяють нам досліджувати ефективність запропонованих моделей та авто масштабування щодо різноманітного макета топологій. Ці топології включають в себе той же набір операторів. Мекети архітектур тестових наборів представлені на рисунку 2.3.

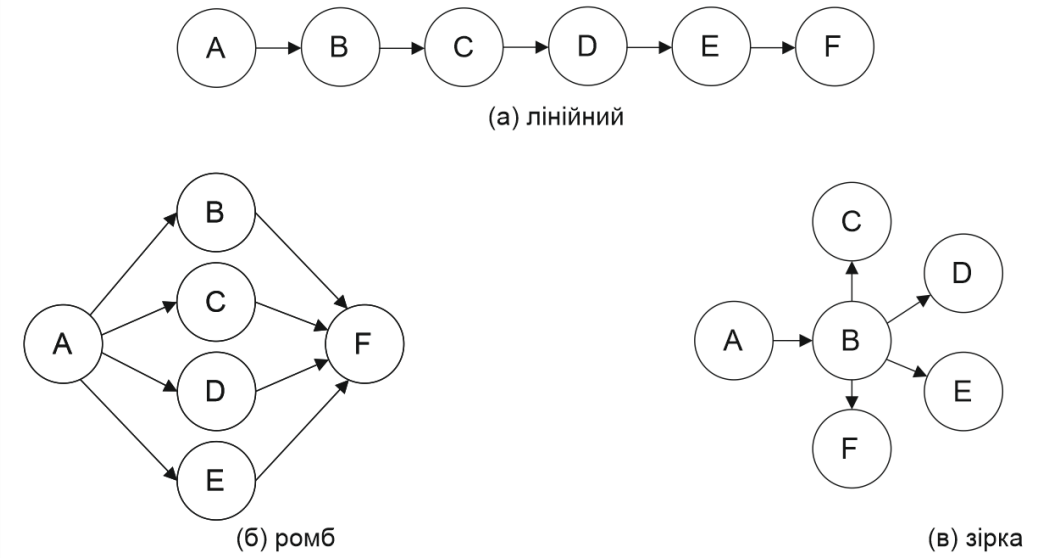


Рисунок 2.3 – Макети тестових застосунків

Storm дозволяє екземпляри, пов'язані з певним оператором, потенційно підвищуючи продуктивність системи та підтримуючи розділення даних. Тому, визначаємо рівень паралелізму чотири для всіх операторів, хоча рівень кожного оператора може бути переналаштований під час обробки для подальшої оптимізації

продуктивності. В результаті для проведеня дослідження була розроблена реальна програма для обчислення популярних тем, що стосуються останніх твітів протягом певних часових інтервалів. Ця топологія складається з чотирьох операторів в лінійній формі. В результаті, n слів в кінцевому підсумку повертаються, щоб повідомити про популярні теми протягом останнього настроюється проміжку часу (наприклад, хвилин, годин або днів). Є три цілі оцінки: 1) обґрунтувати точність оцінки моделей та пропускної здатності, особливо при роботі з постійно мінливим навантаженням на роботу; 2) оцінити ефективність запропонованого авто–скалера в продуктивності системи; 3) обґрунтувати ефективність стратегії, що усвідомлює затримку, у розміщенні операторів ДСП. Зафіксовані вхідні сліди застосунків DSP і відстежено їх результати оцінки пропускної здатності протягом 5 хвилин. Хоча дані випадковим чином потрапляють в застосунки, як показано в вхідних слідах топологій, модель пропускної здатності показала свою здатність регулювати оцінки на основі вхідного трафіку. Результати оцінки для тестування застосунків порівнювалися з виходом системи реального часу. Відповідно до семантики оператора, вихід системи, як очікується, буде відповідати тенденції введення. Оскільки швидкість фіксування даних поступово збільшується, як рівень пропускної здатності системи, так і результати оцінки були збільшені. Максимальне відхилення під час спостереження становить близько 5 хвилин, коли розрахункова пропускна здатність на 0,3% перевищує вихід системи. З тих пір така різниця поступово зменшується. Тому, оцінка пропускної здатності для лінійних топологія може точно передбачити загальну тенденцію. Вхідні дані відчувають різке зростання протягом перших п'яти хвилин і продовжують зростати, тому що введення даних має тенденцію до стабільності. Помічено, що пропускна здатність системи, як і оцінки моделі, узгоджуються з вхідними слідами. Оцінки відхилилися від пропускної здатності системи з менш ніж 1% під час виконання, і тому модель пропускної здатності забезпечує точні оцінки.

Для створення описових текстових інформаційних моделей зазвичай використовують природні мови. Поряд з природними розроблені і використовуються формальні мови: системи числення, алгебра відношень, мови

програмування та ін. Основна відмінність формальних мов від природних полягає в наявність у формальних мов не тільки жорстко зафіксованого алфавіту, але і строгих правил граматики та синтаксису. За допомогою формальних мов будують інформаційні моделі певного типу – формально-логічні моделі. При вивченні нового об'єкта спочатку зазвичай будується його описова модель, потім вона формалізується, тобто документується з використанням математичних формул, геометричних об'єктів тощо. Процес побудови інформаційних моделей за допомогою формальних мов називають формалізацією. Моделі, побудовані з використанням математичних понять і формул, називають математичними моделями. Модель повинна враховувати якомога більше число факторів. Однак реалізувати таке положення складно особливо в слабкоструктурованих системах. Тому найчастіше прагнуть створювати моделі досить простих елементів, з урахуванням їх мікро- і макрозв’язків. Це дозволяє отримувати безпосередньо адекватні результати.

Згідно з результатами оцінки модель працює більш бажано для двох складніших топологій порівняно з лінійною топологією. Щоб оцінити модель затримки, створену за допомогою черги G/G/ і формула A-C, було порівняно результати оцінки з рівнем затримки в реальному часі та іншою моделлю, побудованою з чергою G/G/1 та формулою Кінгмана [41]. Оскільки паралелізація окремих операторів розглядалася при з'єднанні затримки, запропонована модель затримки перевершила модель, побудовану за допомогою черги G/G/1 та формула Кінгмана з точки зору середньої точності оцінки. Розрахункова затримка зідно запропонованої моделі затримки для лінійної топології була на 1,65% вище, ніж спостереження системи. З тих пір оцінки мають загальну тенденцію наближення до системного рівня. Розглядаючи чутливість топології в лінійній формі, поставлена модель G/G/ з формулою A-C миттєво реагує на дисперсію навантаження. Середні відхилення, що відчуваються моделлю G/G/1, становлять 4,2% від затримки системи, тоді як запропонована модель G/G/ має менш ніж 1% відхилення. На рисунку показано, що G/G/1 надмірно реагує на вхідну дисперсію та обчислює затримку, яка перевищує фактичний вихід через відсутність розгляду паралелізм

рівня операторів. Запропонована модель затримки привела до аналогічних результатів оцінки складної топології з максимальним відхиленням 1,76% (на 5 хвилині) від затримки системи. Однак результати оцінки поступово узгоджуються з виходом системи потім. Знову ж таки, модель, побудована з чергою G / G / 1 і формулою Кінгмана, як правило, надмірно реагує на вхідну дисперсію для топології алмазів і завжди відчуває розширену затримка в порівнянні з виходом системи. Зокрема, запропонована модель, побудована з чергою G/G/ та формулою A-C, зазнала в середньому менше 1% відхилень. в той час як черга G/G/ мала відхилення 3,5% від рівня затримки системи. Крім того, запропонована модель затримки спостерігала максимальне відхилення топології зірок і на 3,12% відхилялася від затримки системи. Оцінки в кінцевому підсумку були близькі до системного рівня в кінці експерименту. Аналогічно, модель, побудована з чергою G / G / 1 і формулою Кінгмана, спостерігала нижчу продуктивність з розширеною затримкою і більш високими відхиленнями від рівня затримки в режимі реального часу. У той час як запропонована модель затримки зазнала відхилень на 1,4%, модель, побудована з чергою G/G/1, мала відхилення від виходу системи в середньому на 6,2%.

Запропонована модель затримки перевершує модель, побудовану з G/G/1 чергою і формулою Кінгмана. Ключовою причиною є розгляд декількох сервісних каналів, збережених для індивідуальних операторів. Таким чином, запропонована модель затримки може забезпечити бажані оцінки для різноманітних топологій з різним макетом, за умови варіацій навантаження. Запропонований авто-масштабувальник був оцінений відповідно до стратегії планування Storm згідно стратегії Round-robin щодо пропускної здатності та затримки. Стратегія планування за замовчуванням Apache Storm призначена для рівномірного розподілу виконавців між робочими процесами, тому навантаження може бути рівномірно розподілене між залученими хостами. Три топології були розгорнуті в кластері Storm і виконані близько п'яти хвилин з випадковим шаблоном отримання даних. Рівень пропускної здатності був формалізований як обсяг виводу на вхідний кортеж даних. Без зміни семантики топології, більш високе співвідношення вказує

на підвищену здатність обробки. Рівень пропускної здатності запропонованого авто масштабувальник на основі моделі завжди був вищим, ніж планування Storm, який триває 5 хвилин. Тим не менш, також помічено, що авто-скалятор зазнав розширеної затримки, ніж планування Storm. Розширений рівень відповідав зростанню швидкості потрапляння даних, і тому запропонований авто масштабувальник не в змозі впоратися зі збільшенням навантаження на топології, які мають лінійну компоновку операторів. Щоб вирішити це питання, було проведено подальші дослідження щодо моделі передачі даних з DSP.

Деградація пропускної здатності стала результатом повторної конфігурації авто-масштабувальник, оскільки кілька операторів зобов'язані повторно розподілятися відповідно до етапу планування циклу MAPE в автомасштабі. Щоб перенести операторів, певні процеси потрібно призупинити, доки операції не буде повністю перенесено до щойно вибраних хостів. Очікується, що він перевершить графік Storm щодо рівня пропускної здатності після декількох раундів конфігурацій, поки не буде знайдено оптимальне розміщення операторів. На відміну від цього, затримка авто-масштабувальник перевершувала планування Storm з приблизно 50% поліпшенням під час виконання. Тому, запропонований модельний авто-масштабувальник особливо ефективний у забезпеченні низько латентної обробки з топологіями, які мають складну компоновку операторів. Топологія зірки спостерігала погіршення рівня пропускної здатності і вона, як правило, збільшувалася після цього. Подібно до складної топології, пропускна здатність топології зірок також зазнала коливань під час виконання в результаті повторного виділення оператора. Крім того, затримка, поставлена авто масштабувальником, в основному перевершила графік Storm під час виконання. Отже, потік все ще може відчувати затримку, коли навіть кожен оператор знаходиться на хості з достатніми обчислювальними можливостями. Тим часом, затримка була значно покращена для топології зірок, оскільки окремим операторам дозволяється виконувати паралельно. На відміну від цього, лінійна топологія вимагає послідовної обробки, і затримка зв'язку може домінувати над загальною затримкою. Таким чином, розгляд мережі та передачі даних потрібно для

оптимізації продуктивності DSP далі. Авто масштабувальник для топології був розроблений у формі лінійної, і вона була додатково оцінена щодо пропускної здатності та затримки при розгортанні за допомогою автоматичного масштабувальника. Примітно, що співвідношення між вихідним і вхідним обсягом даних для топології вивчається для інтерпретації рівня пропускної здатності.

2.4 Висновки до другого розділу

Оцінки, зроблені на затримку і пропускну здатність трьох тестових застосунків, показали високу точність виходу моделі при роботі з топологією в основних формах. Очікується, що пропускна здатність і затримка приносять користь будь-яким системам, які можуть бути розкладені на різні топології. Крім того, модельний авто-масштабувальник перевершує метод планування справедливості на рівні пропускної здатності, в той час як лінійні топології спостерігають розширену затримку для запропонованого авто-масштабувальника через відсутність розгляду міжоператорського трафіку та умов мережі. Зведення середовище забезпечує еластичність ресурсів, дозволяючи ефективно автоматично масштабувати ресурс для певних застосунків. З огляду на невизначеність і складності, викликані обробкою потоку, було запропоновано модельний підхід для оцінки пропускної здатності і затримки, а також розробки рішень масштабування відповідно до результатів оцінки. Цей розділ представляє дві математичні моделі, що стосуються пропускної здатності та затримки, і описує модельний авто масштабувальник, призначений для розподілених застосунків обробки потоків. Зокрема, модель затримки була розроблена на основі теорії черг, яка моделює окремих операторів як станції технічного обслуговування G/G/. Крім того, було оцінено щодо ресурсної потужності та різноманітного навантаження. Хоча моделювання в першу чергу було включено на етапі аналізу, два алгоритми визначення типу хоста операторів і розміщення операторів були описані як основні компоненти етапу планування. Розділ оцінки надає обґрунтування моделі за допомогою трьох тестових застосунків з різними формами, оскільки макет

топології може вплинути на результати оцінки запропонованих моделей. Крім того, авто масштабувальник був оцінений відповідно до стратегії планування Storm з трьома тестовими застосунками, а також топологією, яка отримує твіти в режимі реального часу як введення даних. Як показали результати, моделі забезпечують бажані оцінки, а авто масштабувальник підвищив рівень затримки системи та пропускної здатності для різних типів топології.

3 МЕТОД РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ ДЛЯ ОБРОБКИ РОЗПОДІЛЕНИХ ПОТОКІВ ДАНИХ

3.1 Класифікатор потоків та онлайн-монітор

Вимоги до якості обслуговування (QoS), пов'язані з паралельними запущеними запитами, можуть варіюватися від програми до програми. Таким чином, важко гарантувати виконання QoS (наприклад, наскрізний час відгуку) для збору заявок. Розглядаючи динамічний характер окремих потоків та різноманітний попит на ресурси операторів пропонується метод динамічного планування для управління ресурсами для розподілених DSP щодо їх очікувань QoS. Планувальник розроблений як система управління, і він сформульований як задача оптимізації, де набір функцій витрат визначається для досягнення наступних цілей: а) зменшити суми інцидентів порушення QoS у всіх застосунках; б) підвищити рівень використання процесора і (в) уникнути застосункових витрат, викликаних частими переносами.

Велика кількість даних, висока швидкість трафіку даних і різноманітність формату даних, що беруть участь в потоках даних, традиційна система старіння бази даних (СКБД) або моделі пакетної обробки не здатні обробляти безперервні інформаційні потоки майже в режимі реального часу [7, 11, 51]. Тому системи обробки потоків були запропоновані з основною метою щодо відхилення результатів обробки в більш коротку затримку, поки система поповнювалася найсвіжішою інформацією [36]. Такі системи, також відомі як застосунки Big Data, можна знайти в областях, починаючи від фінансового ринку, розумних міст, кібербезпеки та управління охороною здоров'я, а типовим індексом продуктивності таких систем є середній час реагування на зовнішні події [7, 41, 61, 68]. Apache Storm [8], як один з найпопулярніших застосунків для обробки потоків даних, був широко вивчений і застосований практиками галузі. Він підтримує швидке виконання необмежених потоків даних і надає послуги з еластичністю ресурсів. Планувальник Apache Storm вирішує, як розподіляти ресурс для представлених заявок на базових кластерних хостах, а планувальник за замовчуванням,

реалізований в Storm, спрямований на рівномірне розподілення виконавців, які виконують фактичні завдання серед існуючих хостів [8]. Незважаючи на просту реалізацію і легкі накладні витрати, введені таким дизайном планувальника, він не враховує доступність окремих хостів. Іншими словами, кластерні хости, підготовлені для DSP, також можуть бути зайняті деякими іншими програмами. Навіть без неоднорідності типів застосування характеристики окремих систем можуть бути надзвичайно диференційовані один від одного. Як наслідок, метод планування, спрямований на справедливий розподіл операторів DSP, не здатний гарантувати угоди QoS. Хоча справедливий розподіл намагається збалансувати розподіл ресурсів у спільному середовищі [58], це призведе до серйозної деградації продуктивності, коли рівні застосування QoS безпосередньо сильно відрізняються в різних застосунках [52].

З огляду на поширену практику, яка розгортає програми в хмарних екосистемах, DSP зазвичай конкурують за поставки ресурсів на такій платформі з декількома орендарями. Хоча деякі програми DSP дуже чутливі до затримок і вимагають, щоб їх дані оброблялися миттєво, наприклад, в порядку мілісекунд, є заявки, які є такими, що мають відгук протягом більш тривалого часу відгуку, наприклад, в порядку годин або навіть днів [13]. Щоб вирішити різноманітність застосунків DSP в очікуваннях QoS, визначаємо набір пріоритетних черг для класифікації застосунків відповідно до їх чутливості до затримки. У запропонованому методі планування основні ресурси управляються індивідуально для кожної черги пріоритетів, щоб потенційно уникнути конкуренції між різними типами застосунків. Крім того, загальна продуктивність може бути гарантована, в той час як витрати на підтримку базової інфраструктури зведені до мінімуму з усвідомленням конкретних вимог QoS. Розроблена система управління для збору системних і прикладних показників, моделювання поведінки системи та оптимізації продуктивності представлена на рисунку 3.1 концептуальною схемою, де наявні кожен модуль з їх очікуваними функціями та ролями. При безперервному потраплянні даних в систему для обробки потоки даних попередньо обробляються в класифікаторі системи.

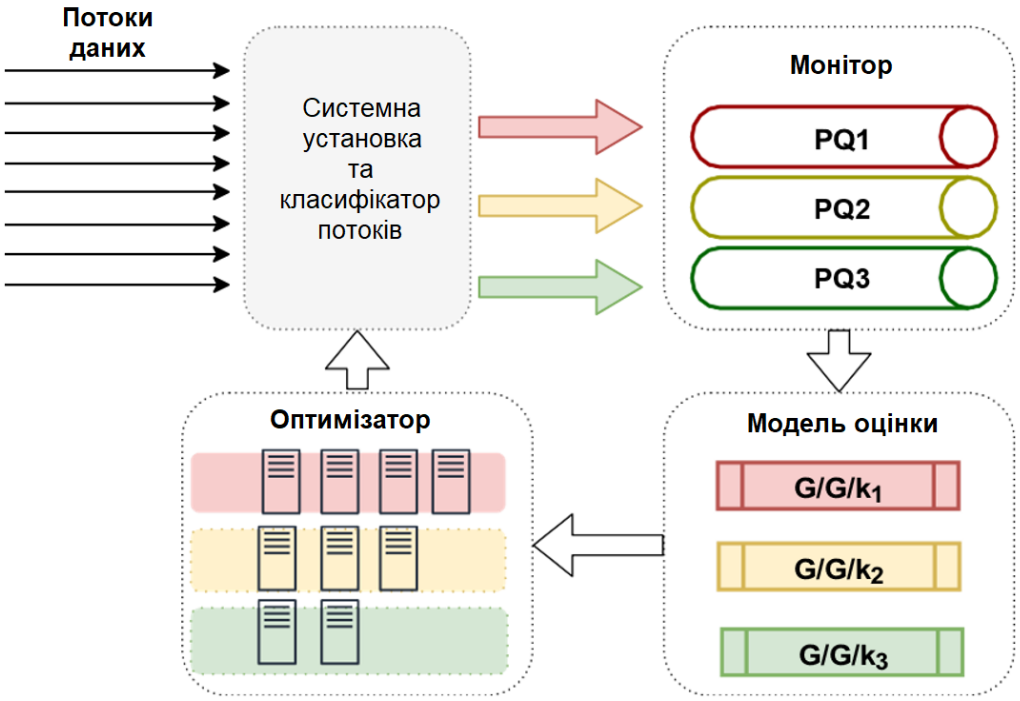


Рисунок 3.1 – Концептуальне схема розробленої системи управління

Зокрема, класифікатор відстежує застосунок, який належить даному кортежу даних, оскільки заявки були позначені пріоритетами на момент їх подання в систему. Припускаємо, що мітки надаються кінцевими користувачами для визначення очікуваного часу відгуку. Потім кортеж даних буде спрямований у відповідні черги, які збереглися для його застосування, в той час як кілька серверів зберігаються для розміщення застосунків у кожній черзі. На другому етапі монітор збирає стан системи (наприклад, доступність ресурсів) і показники застосунків (наприклад, швидкість походу даних, пропускна здатність системи) під час виконання. Очікується, що такий процес моніторингу з простою реалізацією і пов'язаний з мінімальними накладні витрати. Оцінювач моделі застосовує теорію черги для моделювання кожної пріоритетної черги як багатоканальної станції технічного обслуговування та оцінює затримку доставки для прогнозування порушень QoS. Нарешті, оптимізатор відноситься до набору об'єктивних функцій і має на меті знайти оптимальне рішення щодо кількості ресурсів, які будуть підготовлені для кожної пріоритетної черги. Програми, призначені ідентичним чергам, потім надаватимуть спільний доступ до ресурсів, виділених їм. Як

динамічний характер DSP застосунків, навантаження може бути змінюватися час від часу, і, отже, ресурси, що надаються для кожної черги буде динамічно коригуватися відповідно до вимог QoS і уникнути порушення інцидентів. Увесь операційний часовий горизонт ділиться на рівні інтервали, і рішення потрібно приймати на початку кожного інтервалу. Класифікатор потоку спрямовує вхідні потоки даних до відповідної черги пріоритетів, розглядаючи їх атрибути QoS. Політика QoS підтримує $Q \geq 2$ різних класів, а $Q = 3$ у прикладі, показаному на рисунку 3.1. Три класи позначені як низькі, середні та високі і позначаються 1, 2 і 3 відповідно. На практиці може бути визначено більше класів для підтримки дрібнозернистого контролю варіацій QoS. Модуль онлайн-моніторингу збирає, аналізує та інтерпретує час прибуття та запитаний час обслуговування потокових застосунків на пріоритетній основі черги. Він надає результати аналізу щодо часу прибуття даних та часу обробки заявок у кожній черзі до відповідних моделей. Крім того, також збирається кількість хостів, зарезервованих для кожної черги, і доступність ресурсів залученого хоста в часі. Модель оцінювач моделює пріоритетні черги як багатоканальні СМО і виконує системну оцінку для кожної черги, посилаючись на отримане прибуття час і час служби. Результати оцінки інтерпретують очікувану подію під заданою кількістю ресурсів і вказують на можливі випадки порушення QoS в конкретних чергах. Після фіксації зв'язків між потрібними показниками продуктивності (наприклад, час очікування та використання процесора) та приблизними значеннями, модуль оптимізатора має на меті призначити найкращий параметр для платформи обробки. Зокрема, оптиміст повинен вирішити обмежену проблему мінімізації та визначає кількість ресурсів, пов'язаних із кожною чергою пріоритетів, де відзначена, і набір хостів, які потрібно призначити. Нарешті, динамічний планувальник коригує розподіл ресурсів і повторно відображає потокові програми до хостів, підключених до певних черг, як інструктовано модулем оптимізатора. У той же час весь цей цикл моніторингу, моделювання, оцінки та оптимізацій процесу повторюється.

На початку класифікатор направляє дані в певну групу хостів для процесу, і монітор продовжує збирати всю відповідну інформацію щодо можливостей

ресурсів та застосунків. Припускаючи, що група потокових застосунків працює на цільовій платформі процесу, ділячись базовими ресурсами у форматі віртуальних машин. Для вирішення різноманітних вимог QoS серед цих DSP, програми класифіковані на основі їх рівня чутливості до латентності. Класифіковані програми виділяються на черги, які мають аналогічні вимоги до обробки. Зокрема, кожна черга пріоритетів пов'язана зі значенням індексу, що вказує на верхню межу затримки, яка має бути задоволена для всіх застосунків, що належать до нього. В результаті, програми кластеризовані, посилаючись на їх вимоги до обробки і виділяються купі хостів, збережених для даної черги. Наприклад, черга з більш високим пріоритетом означає, що очікуваний час відгуку пов'язаних з ними DSP відносно коротший, і лише кілька порушень QoS є прийнятними. Як початковий етап системи управління, онлайн-монітор отримує системні показники та метрики застосування для фіксації динамічної зміни робочого навантаження та доступності ресурсів. Оскільки однією з цілей запропонованого планувальника є максимізація використання ресурсів, очікується, що він консолідує виконавців, використовуючи мінімальну кількість хостів. Отже, рівень використання та доступність залучених хостів обов'язково необхідні для процесу моніторингу. Монітор отримує показники застосунків протягом усього періоду виконання та аналізує час прибуття та обслуговування для кожної черги.

Таким чином, розроблений метод та система управління, яка представлена концептуальною схемою містять класифікатор задач, що надходять на обробку, і є основою для реалізації.

3.2 Динамічне планування

Динамічне програмування (ДП) - метод оптимізації, пристосований до операцій, в яких процес прийняття рішення може бути розбитий на етапи (кроки). Такі операції називаються багатокроковими. Початок розвитку ДП відноситься до 50-м р XX в. Воно пов'язане з ім'ям Р. Веллман. Якщо моделі лінійного

програмування можна використовувати в економіці для прийняття великомасштабних планових рішень в складних ситуаціях, то моделі ДП застосовуються при вирішенні завдань значно меншого масштабу, наприклад при розробці правил управління запасами, що встановлюють момент поповнення запасів і розмір поповнює замовлення; при розробці принципів календарного планування виробництва і вирівнювання зайнятості в умовах мінливого попиту на продукцію; при розподілі дефіцитних капітальних вкладень між можливими новими напрямками їх використання; при складанні календарних планів поточного і капітального ремонту складного обладнання та його заміни; при розробці довгострокових правил заміни тих, що вибувають з експлуатації основних фондів і т.п. Оскільки поставлені завдання з розподілу обчислювальних ресурсів відносяться до такого класу задач, тільки виконуються автоматично, то вони можуть бути адаптовані до розглядуваної задачі з обробки потоків даних. В реально функціонуючих великих економічних системах щотижня потрібно приймати мікроекономічні рішення. Моделі ДП цінні тим, що дозволяють на основі стандартного підходу з використанням ПК при мінімальному втручанні людини приймати такі рішення. І якщо кожне взяте окремо таке рішення малоістотно, то в сукупності ці рішення можуть мати великий вплив на прибуток. Розглянемо загальну постановку задачі ДП. Розглядається керований процес, наприклад економічний процес розподілу коштів між підприємствами, використання ресурсів протягом ряду років, заміни обладнання, поповнення запасів і т.п. В результаті управління система (об'єкт управління) S перекладається з початкового стану 50 в стан s. Припустимо, що управління можна розбити на η кроків, тобто рішення приймається послідовно на кожному кроці, а управління, що переводить систему S з початкового стану в кінцеве, являє собою сукупність η покрокових управлінь. Позначимо через X k управління на k-м кроці (до = 1, 2, ..., η). Змінні X k задовольняють деяким обмеженням і в цьому сенсі називаються допустимими (X до може бути числом, точкою в "- мірному просторі, якісною ознакою). Нехай X (X ,, X 2, ..., X п) - управління, що переводить систему 5 зі стану s0 в стан s. Позначимо

через s k стан системи після / " '- го кроку управління. Отримуємо послідовність станів s0, st, ..., s k _ v s k, ..., s n _ -, яку зобразимо кружками на рисунку 3.2[90]:

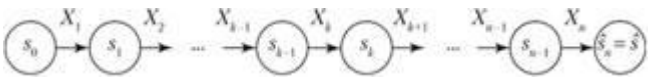


Рисунок 3.2 – Послідовність станів

Показник ефективності даної керованої операції - цільова функція залежить від початкового стану і управління[90]:

$$\mathrm{Z} = F(S_0, X) \qquad (3.1)$$

Припустимо, що:

1. Стан s k системи в кінці до -го кроку залежить тільки від попереднього стану s k, і управління на кожному кроці X до (і не залежить від інших попередніх станів і управлінь)[90]. Е го вимога називається "відсутністю післядії". Сформульоване положення записується у вигляді рівнянь:

$$S_к = \varphi_к(S_{к-1}, X_к), к = 1,2, \dots, n, \qquad (3.2)$$

які називаються рівняннями станів.

2. Цільова функція (3.1) є адитивною від показників ефективності кожного кроку[90]. Позначимо показник ефективності ⅛-го кроку через

$$\mathrm{Z}_к = \int_к (S_{к-1}, X_к), к = 1,2, \dots, n, \qquad (3.3)$$

тоді отримаємо

$$Z = \sum_{к=1}^{n} \int_к (d_{k-1}, X_k) \qquad (3.4)$$

Завдання покрокової оптимізації (задача ДП) формулюється так: визначити таке допустиме управління X, що переводить систему S зі стану s_0 в стан при якому цільова функція (3.4) приймає найбільше (найменше) значення.

Виділимо особливості моделі ДП:

– завдання оптимізації інтерпретується як п-кроковий процес управління;

– цільова функція дорівнює сумі цільових функцій кожного кроку;

– вибір управління на k-му кроці залежить тільки від стану системи до цього кроку, не впливає на попередні кроки (немає зворотного зв'язку);

– стан s k після к-го кроку управління залежить тільки від попереднього стану s k ↑ і управління X до (відсутність післядії);

– на кожному кроці управління X до залежить від кінцевого числа керуючих змінних, а стан s k - від кінцевого числа параметрів (сенс зауваження 5 стане зрозумілим із розглянутих нижче прикладів).

Існують різні способи вирішення подібних завдань, що застосовуються в залежності від виду функцій, обмежень, розмірності і т.п. Розглянемо обчислювальну схему ДП, яка виявиться байдужою до способів завдання функцій і обмежень. Обчислювальна схема пов'язана з принципом оптимальності і використовує рекурентні співвідношення.

Динамічний планувальник фіксує поведінку системи реального часу, посилаючись на дані, зібрані системним монітором або обчислені оцінювачами моделі. Потім він вирішує проблему оптимізації, щоб знайти рішення, яке вводить мінімальні витрати щодо налаштованих коефіцієнтів. Вивід планувальника інформує систему з кількістю віртуальних машин, які будуть виділені для кожної пріоритетної черги щодо вимог QoS та рівня використання ресурсів. Процедура планування потребує на початку увесь список віртуальних машин і класифікованих застосунків у форматі пріоритетних черг. Крім того, максимальна кількість віртуальних машин, дозволених для кожного відповідного черги забезпечує очікування QoS з точки зору затримки окремих програм DSP. Вивід ініціалізується як null і зберігається як структура карти, в якій ключем є індекс черги пріоритетів, а значення дає список віртуальних машин, з якими потрібно зв'язати. Процедура планування починається зі збору відповідної інформації (наприклад, максимальна кількість віртуальних машин, які дозволяє мати черга, доступні віртуальні машини) кожного у встановлених пріоритетних чергах. Зокрема, функція знаходить

максимальну кількість віртуальних машин, що дозволяє мати, і вибирає N ідентичних віртуальних машин. Функція має на меті перерахувати будь-яку можливу комбінацію віртуальних машин щодо обмежень. Щоб виправдати ефективність всіх можливих рішень, відповідні витрати (наприклад, порушення QoS, обчислювальні та комутаційні витрати) обчислюються за допомогою відповідних функцій. Зокрема, метод обчислює порушення QoS, а також вартість ресурсів та вартість перемикання. Загальна вартість даного вибору віртуальних машин порівнюється з розглянутими рішеннями і повторюється до тих пір, поки не буде знайдено мінімальне значення після ітерації всього набору. Після того, як система отримує S як рішення щодо планування, вона потім готує систему до готовності для виконання коригувань. Система налаштування зазвичай включає в себе підготовки нових віртуальних машин або звільнення існуючих віртуальних машин для черг і перенесення пов'язаних потоків та їх операторів до вказаних віртуальних машин.

Таким чином, удосконалення в динамічному планувальнику стосується врахування оцінки витрат, що є однією з характеристик ефективності запропонованих рішень.

3.3 Метод розподілу обчислювальних ресурсів

Операційна система не тільки надає користувачам і програмістам зручний інтерфейс до апаратних засобів комп'ютера, але і є механізмом, який розподіляє ресурси комп'ютера. Обчислювальними ресурсами називаються можливості, що забезпечуються компонентами обчислювальної системи, що витрачаються (займані) в процесі її роботи.

Типи обчислювальних ресурсів:

- процесорний час;
- пам'ять (оперативна і віртуальна);
- місце на жорсткому диску (постійна пам'ять);
- пропускна здатність мережі.

Ресурси розподіляються між процесами. Процес (задача) - програма в стадії виконання. Програма - це статичний об'єкт, що представляє собою файл з кодами і даними. Процес - це динамічний об'єкт, який виникає в операційній системі після того, як користувач або сама операційна система вирішує «запустити програму на виконання», тобто створити нову одиницю обчислювальної роботи. Управління ресурсами обчислювальної системи з метою найбільш ефективного їх використання є призначенням операційної системи.

Ресурси - обсяг роботи або термін експлуатації, на який розраховується машина, будинок тощо. Після вичерпання ресурсу безпечна робота пристрою не гарантується, йому потрібен капітальний ремонт або заміна.

Ресурсом є будь-який компонент ПК і надані їм можливості: центральний процесор, оперативна або зовнішня пам'ять, зовнішній пристрій, програма тощо. Ресурси поділяються на чотири види.

Види ресурсів персонального комп'ютера: апаратні ресурси, файлові ресурси, програмні ресурси, мережеві ресурси

Апаратні ресурси - це системний блок, периферійні пристрої, будь-яке обладнання, підключене до комп'ютера.

Файлові ресурси - це файли і папки, а також вся файлова система.

Програмні ресурси - це все програми встановлені в комп'ютері. Часто називають програмним забезпеченням (ПЗ). Програмне забезпечення підрозділяється на два види: системне і прикладне ПЗ.

Мережеві ресурси - ресурси доступні за коштами ЛВС. Як правило, це ресурси інших комп'ютерів доступні за локальної або глобальної мережі.

Комп'ютерна мережа - апаратне і програмне об'єднання двох і більше комп'ютерів з виділенням мережевих ресурсів. Для зв'язку комп'ютерів в комп'ютерну мережу можуть бути використані наступні апаратні засоби: модем; мережева карта; мережевий кабель; мережеві комутатори; WI-FI адаптер; бездротове обладнання; маршрутизатор; мережеві екрани тощо.

Мережевими ресурсами можуть бути:

– обладнання (тобто апаратні ресурси іншого ПК або мережеві пристрої), наприклад мережевий принтер;

– інформація (тобто файли і папки іншого комп'ютера), наприклад інформація в Інтернеті, або на сервері;

– програмне забезпечення (встановлений на іншому комп'ютері). [2]

Головним елементом комп'ютера є мікропроцесор - електронна схема, що виконує всі обчислення і обробку інформації. Коли доводиться виконувати багато математичних обчислень, до основного мікропроцесора додають математичний співпроцесор.

Мікропроцесор - процесор (пристрій, що відповідає за виконання арифметичних, логічних операцій і операцій управління, записаних в машинному коді), реалізований у вигляді однієї мікросхеми або комплекту з декількох спеціалізованих мікросхем (на противагу реалізації процесора у вигляді електричної схеми на елементної базі загального призначення або у вигляді програмної моделі). Перші мікропроцесори з'явилися в 1970-х і застосовувалися в електронних калькуляторах, в них використовувалася двійковій-десяткова арифметика 4-х бітних слів. Незабаром їх стали вбудовувати і в інші пристрої, наприклад термінали, принтери і різну автоматику. Доступні 8-бітові мікропроцесори з 16-бітної адресацією дозволили в середині 1970-х створити перші побутові мікрокомп'ютери.

Довгий час центральні процесори створювалися з окремих мікросхем малої і середньої інтеграції, що містять від декількох одиниць до декількох сотень транзисторів. Розмістивши цілий ЦПУ на одному чіпі сверхбольшой інтеграції вдалося значно знизити його вартість. Незважаючи на скромний початок, безперервне збільшення складності мікропроцесорів призвело до майже повного старіння інших форм комп'ютерів, в даний час один або кілька мікропроцесорів використовуються в якості обчислювального елемента в усьому, від найдрібніших вбудованих систем і мобільних пристроїв до величезних мейнфреймів і суперкомп'ютерів. З початку 1970-х широко відомо, що зростання потужності мікропроцесорів слідує закону Мура, який стверджує що число транзисторів на

інтегральній мікросхемі подвоюється кожні 18 місяців. В кінці 1990-х головною перешкодою для розробки нових мікропроцесорів стало тепловиділення (TDP) через витоки струму та інших факторів. Деякі автори відносять до мікропроцесорах тільки пристрої, реалізовані строго на одній мікросхемі. Таке визначення розходиться як з академічними джерелами, так і з комерційною практикою (наприклад, варіанти мікропроцесорів Intel і AMD в корпусах типу SECC і подібних, такі як Pentium II - були реалізовані на декількох мікросхемах). В даний час, у зв'язку з дуже незначною присутністю мікропроцесорів, які не є процесорами, в побутовій лексиці терміни «мікропроцесор» і «процесор» практично рівнозначні.

Однією з важливих задач операційної системи є управління наявними в її розпорядженні ресурсами (основний пам'яттю, пристроями введення-виведення, процесором), а також їх розподіл між різними активними процесами. При розробці стратегії розподілу ресурсів необхідно брати до уваги такі фактори.

Рівноправність. Зазвичай бажано, щоб всім процесам, які претендують на якийсь певний ресурс, надавався до нього однаковий доступ. Особливо це стосується завдань, що належать до одного і того ж класу, тобто завдань з аналогічними вимогами до ресурсів.

Диференціація відгуку. З іншого боку, може знадобитися, щоб операційна система по-різному ставилася до завдань різного класу, які мають різні запити. Потрібно спробувати зробити так, щоб операційна система виконувала розподіл ресурсів відповідно до цілим набором вимог. Операційна система повинна діяти в залежності від обставин. Наприклад, якщо якийсь процес очікує доступу до пристрою введення-виведення, операційна система може спланувати виконання цього процесу так, щоб якомога швидше звільнити пристрій для подальшого використання іншими процесами.

Ефективність. Операційна система повинна підвищувати пропускну здатність системи, зводити до мінімуму час її відгуку і, якщо вона працює в системі поділу часу, обслуговувати максимально можливу кількість користувачів. Ці вимоги дещо суперечать один одному; нагальною проблемою дослідження

операційних систем є пошук потрібного співвідношення в кожній конкретній ситуації.

Завдання управління ресурсами і їх розподілу типова для досліджень операційних систем; тут можуть застосовуватися математичні результати, отримані в цій області. Крім того, важливо вимірювати активність системи, що дозволяє стежити за її продуктивністю і вносити корективи в її роботу.

Операційна система підтримує кілька черг, кожна з яких є просто списком процесів, які очікують своєї черги на використання якогось ресурсу. В короткострокову чергу заносяться процеси, які (або, принаймні, основні частини яких) знаходяться в основній пам'яті і готові до виконання. Вибір чергового процесу здійснюється короткостроковим планувальником, або диспетчером. Загальна стратегія полягає в тому, щоб кожному знаходиться в черзі процесу давати доступ по черзі; такий метод називають циклічним (round-robin). Крім того, процесам можна привласнювати свої пріоритети. У довгостроковій черзі перебуває список нових процесів, які очікують можливості використовувати процесор. Операційна система переносить їх з довгострокової черзі в короткострокову. У цей момент процесу необхідно виділити певну частину основної пам'яті. Таким чином, операційна система повинна стежити за тим, щоб не перевантажити пам'ять або процесор, додаючи в систему занадто багато процесів. До одного і того ж пристрою введення-виведення можуть звертатися кілька процесів, тому для кожного пристрою створюється своя черга. І тут операційна система повинна вирішувати, якому процесу надати місце, що звільнилося, зокрема пристрій введення-виведення в першу чергу.

Під час переривання управління переходить до обробника переривань, який є частиною операційної системи. В силу своєї функціональності процес може звернутися до деякого сервісу операційної системи, наприклад до драйверу пристрою введення-виведення. При цьому відбувається виклик обробника звернень до сервісів, який стає точкою входу в операційну систему. Незалежно від того, чи відбулося переривання або звернення до сервісу, після його обробки планувальник вибере з короткостроковій черзі процес для виконання.

Модуль оптимізації враховує набір функцій і обмежень для перерахування кількості віртуальних машин, пов'язаних з кожною чергою. Система управління отримує результати оцінки від оцінювача моделі і інформує динамічний планувальник з похідними конфігураціями. Маршрути класифікатора потоку подають потоки до певної черги пріоритетів таким чином, щоб відставання від очікувань затримки, гарантували при цьому продуктивність кожної пріоритетної черги. Оптимізатор, як очікується, повністю використовує обчислювальні можливості в той час як консервативно резервувати ресурс для застосунків в нижніх чергах пріоритетів не потрібно. Іншими словами, він уникає безперервного масштабування або масштабується для застосунків з високим рівнем до затримок. Така стратегія потенційно може зарезервувати ресурси для застосунків, які менш терпимі до затримок, особливо коли вони відчувають раптове збільшення навантаження. Отже, проблема оптимізації полягає у визначенні методу розподілу ресурсів, який відповідає очікуванням затримки, використовуючи меншу ємність ресурсів. Більш конкретно, є три основні цілі, які повинні бути задоволені модулем оптимізатора: (1) зменшити загальну кількість інцидентів порушення QoS у всіх пріоритетних чергах; (2) підвищити рівень використання процесора активних хостів шляхом об'єднання завдань у меншу кількість хостів; і (3) мінімізувати витрати, викликані частими повторними налаштуваннями.

Обчислювальна вартість це кількість обчислювальних ресурсів в черзі. Сконцентруємо на кількості віртуальних машин і ядер процесора, призначених кожній черзі, в той час як функція може бути додатково розширена, щоб включити введення-виведення, мережі та інших видів ресурсів. Вартість ресурсу представляє ціну за одиницю за використання ядра процесора. Це загальна кількість ядер процесора, приєднаних до виділених віртуальних машин. На практиці, різні значення можуть бути застосовані, якщо неоднорідний кластер бере участь. Примітно, що він відрізняється від того, що він явно забезпечує кількість ядер процесора, в той час як дає кількість віртуальних машин, пов'язаних з чергою. Що стосується неоднорідності ресурсів у хмарі то, віртуальні машини можуть бути підготовлені з різним розміром, а розмір може бути в подальшому динамічно

налаштованим. Таким чином, розглядаємо в моделі витрат, щоб забезпечити точну оцінку обчислювальної вартості з точки зору ресурсів процесора, зайнятих пріоритетних черг. Комутаційна вартість бере участь в об'єктивній функції, щоб уникнути частих рекламних виправдань. В результаті надмірної реакції на будь–яку спостережувану зміну навантаження, часте коригування ресурсів може призвести до непотрібних витрат системних ресурсів. Як правило, існує затримка між спостереженням за зміною навантаження і проведеною реакцією, регулювання з високою частотою може не відповідати попиту на ресурси і навіть викликати серйозну деградацію системи. Тому, моделюємо вартість перемикання як штрафну функцію для нормалізації модифікації, внесеної з поточних конфігурацій. Запропонований планувальник гарантовано буде більш консервативним у прийнятті помірних змін для розподілу ресурсів, і він особливо ефективний у боротьбі з коливаннями навантаження. Іншими словами, набір траєкторій виводиться з порівняння між існуючим станом системи і очікуваним виходом. Замість того, щоб рухатися безпосередньо до цільового результату, перший крок траєкторії розглядається як бажана дія, яка вводить менші витрати на перемикання. Отже, під час виконання буде застосовано ряд плавних коригувань і прогресивно підходить до оптимального рішення. Мінімізуючи витрати на розробку оптимального розміщення для операторів, необхідно, також, виконати обмеження щодо рівня використання черг, кількість віртуальних машин може бути збережена для кожної черги та суми коефіцієнтів. Наприклад, встановлення більш високої ваги для вартості ресурсів полягає в тому, щоб поставити більш строге правило на забезпечення віртуальної машини порівняно з іншими видами витрат. Відповідно, оптимізатор звертається до значення вартості ресурсу, виводячи оптимальні рішення з мінімальною пропозицією ресурсу для черг, що буде супроводжуватися певними порушеннями QoS або досить частими коригуваннями в якості цін і пропускною здатністю. Ядра процесора для кожного оператора можуть бути розширеними, щоб мінімізувати навантаження на інші мережі і оптимізувати роботу інших віртуальних машин.

3.4 Експерименти, тестування програм і черги

Для початку проведемо серію експериментів для вивчення кореляції міжоператорського зв'язку та мережевого трафіку. Застосуємо контроль трафіку Linux, щоб імітувати конкуренцію за пропускну здатність серед декількох аплікацій DSP. Контроль трафіку дозволив нам маніпулювати пакетами, що доставляються за бажаною швидкістю. Розглядаючи два вузли для виконання топології «SlidingWindow», було здійснено оцінку його середньої затримки після введення різноманітного рівня затримок між двома вузлами і збереження інших конфігурацій незмінними (тобто частота прийому даних, рівень паралелізму оператора і т.д.). Застосований планувальник Storm Even, щоб рівномірно розмістити операторів існуючим працівникам і призначити двох систем на вузол. Макет топології та розміщення кожного екземпляра оператора відповідно показані на рисунку 3.2.

Затримка між двома системами була введена як 5, 10, 20, 50 і 100м для імітації різноманітного рівня затримки, викликаної міжоператорним рухом великої групи процесів. Середня затримка була зібрана для кожної конфігурації після запуску протягом 10 хвилин. Підвищена затримка, викликана введеною затримкою, була проілюстрована в відсоток. Згідно з рисунком 3.2, топологія пережила не більше 3% деградації затримки, коли введена затримка становить менше 20мс.

Однак вона стала до 10% і наблизилася до 20%, коли затримки були встановлені на 50ms і 100ms відповідно. На практиці або тимчасове велике навантаження, або обмежені мережеві ресурси можуть призвести до тривалих затримок мережі, а отже, розміщення операторів без усвідомлення міжоператорського зв'язку призведе до субоптимальності продуктивності. Досліджувався, також, вплив трафіку застосунків на пропускну здатність топології. Оскільки міжоператорський трафік може відбуватися тільки на цих двох вузлах, основною причиною спостережуваної деградації пропускної здатності буде збільшення кількості трафіку.

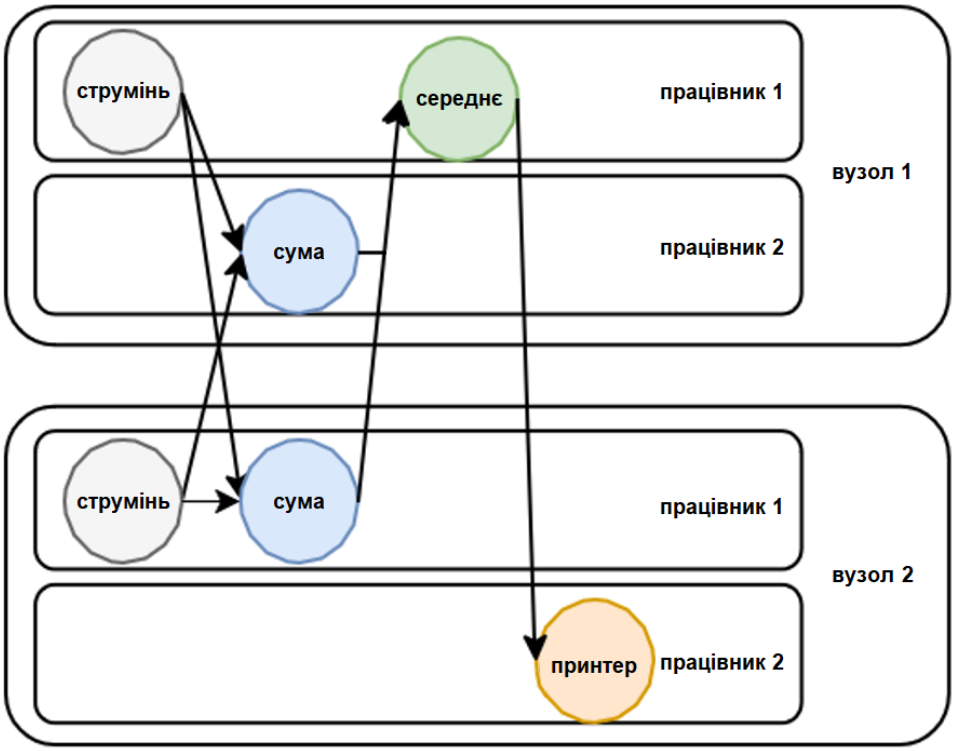


Рисунок 3.3 – Розміщення топології

3.5 Висновки

В результаті було оцінено ефективність запропонованого планувальника в Apache Storm з трьома тестовими застосунками, що проникають в режимі реального часу. Програми охоплюють цілий ряд різних макетів операторів і різноманітний рівень толерантності до затримки, щоб забезпечити різноманітність застосунків DSP, які розглядаються в наших оцінках. Ефективність запропонованого рішення виправдана щодо наступних аспектів: кількість порушень QoS, що виникають у пріоритетних чергах (DSP-застосунках); середнє використання процесора базової платформи; 3) вплив налаштування коефіцієнтів об'єктивної функції. Відповідно до порушень QoS, з якими стикаються програми тестування, можна тим самим виправдати ефективність запропонованого планувальника, який в першу чергу спрямований на гарантію дисперсії очікувань QoS у ряді застосунків DSP. Крім того, відстеження рівня використання процесора дозволяє обґрунтувати ефективність запропонованого методу зниження

споживання ресурсів шляхом об'єднання завдань у меншу кількість вузлів. Нарешті, налаштування коефіцієнтів об'єктивної функції дозволяє обґрунтувати ефективність запропонованого планувальника у відповідь на особливі інтереси користувачів (наприклад, менш часте коригування ресурсів, висока толерантність до порушень QoS). Запропонований планувальник QoS показує свою ефективність у динамічному розподілі ресурсів для застосунків DSP з пріоритетними чергами впровадження відповідно до оцінки результатів. Планувальник особливо добре працює для DSP, які чутливі до затримок, оскільки ці програми мають найвищий пріоритет під час виконання та зберігаються з достатніми ресурсами, що стосуються різкого збільшення запитів користувачів. Крім того, планувальник QoS підвищує рівень використання базових ресурсів, оскільки комутаційна вартість розглядається як основний компонент об'єктивної функції.

4 ПРОГРАМНА СИСТЕМА РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ ДЛЯ ОБРОБКИ РОЗПОДІЛЕНИХ ПОТОКІВ ДАНИХ

4.1 Дослідження застосування парадигми оброблення потоків, орієнтованої на трафік

За визначенням Поста і Тюрінга – це обчислювальний процес, який виконується деякою моделлю обчислювача, яка зконструйована в рамках точних математичних понять. Таке визначення алгоритму дає змогу, по-перше, класифікувати алгоритми за типом моделі обчислювача, по-друге, порівнювати між собою різні алгоритми шляхом їх еквівалентного перетворення в алгоритми, що задані на стандартній моделі обчислювача, такій, як машина Тюрінга, по-третє, конструювати такі моделі обчислювачів, які найбільш придатні для ефективного задавання того чи іншого класу алгоритмів.

Визначення алгоритму, стосується, перш за все, нейманівської моделі обчислювача. Така модель характеризується тим, що вона виконує послідовно в строгому порядку команди, які змінюють стан пам’яті обчислювача, від початкового до кінцевого. На такій моделі виконуються операторні, імперативні алгоритми.

В функціональних алгоритмах обчислення виражаються як множина тотожностей від функцій. При цьому результати функцій обчислюються тільки тоді, коли для них є відповідні вхідні дані. В функціональному алгоритмі немає обмежень на порядок обчислення функцій крім порядку, який задається залежностями за даними.

Модель обчислювача, що реалізує функціональний алгоритм, прийнято представляти у вигляді графа. В такій моделі вершини графа представляють функції або оператори алгоритму, а дуги – залежності по даних і керуванню. Реалізація алгоритму на такій моделі представляє собою передачу даних у напрямку дуг і обхід вершин у відповідності до їхньої інцидентності, наявності даних на їхніх входах або за іншими правилами. На відміну від імперативних алгоритмів, тут порядок виконання операторів є нестрогим, а паралелізм алгоритму

заданий явним чином. Далі розглянемо кілька відомих графових моделей функціональних алгоритмів щоб показати їхні основні властивості.

Найбільш відомою графовою моделлю обчислень є мережа Петрі. Граф цієї мережі складається з вершин-позицій і вершин-переходів, які попарно зв'язані дугами. Виконання алгоритму в цій моделі керується наявністю міток в вершинах-позиціях, які асоціюються з даними. Якщо кількість міток на входах вершини-переходу є достатньою, то вона спрацьовує і відповідна мітка з'являється на її виході. Виконання алгоритму складається в серії спрацьовувань переходів в довільному порядку, аж поки не залишиться умов для таких подій. Мережі Петрі використовуються для моделювання роботи асинхронних систем, як наприклад, протоколів обміну даними. Хоча існує достатня кількість робіт, присвячених застосуванню цих мереж в проектуванні обчислювальної апаратури, вони не набули великого розповсюдження в цій галузі. Основна причина цього є та, що дуже важко аналізувати мережу Петрі, визначаючи її безпечність, відсутність блокувань та інші властивості.

В обчислювальній моделі *мережі процесів Кана* асинхронні процеси взаємодіють між собою через послідовні буфери даних, названих потоками, які підкоряються дисципліні перший зайшов в буфер – перший вийшов, тобто FIFO. В цій моделі вершина процесу може виконувати ввід даних в довільному порядку. Але після читання даного процес блокується, поки не буде в наявності нове дане, тобто процес запускається по оператору wait on data, а саме дане вилучається. В загальному випадку, якщо буфери FIFO обмеженої глибини, ця мережа може бути заблокованою, причому неможливо встановити заздалегідь, чи відбудеться таке блокування. Тому коректне виконання алгоритму може бути гарантоване лише при динамічному складанні розкладу.

На відміну від мережі Петрі, в мережі процесів Кана стан даних в потоках не залежить від розкладу виконання процесів, тобто потоки даних в моделі є детермінованими. Ця мережа дала поштовх для народження багатьох інших моделей обробки потоків даних. Наприклад, *граф потоків даних* (ГПД) характеризується тим, що дуги мають розмітку, яка вказує, яка глибина буфера

FIFO, скільки міток заходить в дугу й виходить з неї за один запуск акторів на її кінцях, вершини можуть мати різні правила спрацьовування, де також включаються умовні спрацьовування.

Для класифікації моделей обробки потоків даних розглянемо визначення потоку даних і його параметри.

Всі моделі обчислювачів потоків даних мають ту загальну особливість, що їх компоненти – вузли, оператори, актори або процеси – з'єднані між собою потоками даних. Часто потік даних називають сигналом. В загальному випадку, *потік даних* – це засіб передачі скінченної або в граничному випадку, нескінченної кількості впорядкованих даних між компонентами обчислювальної моделі.

Обчислювальні моделі відрізняються між собою способом виконання потоків даних. Спосіб такого виконання залежить від певних властивостей або ознак потоку, тобто від того: чи потік однонаправлений, чи двонаправлений; чи у потока одне джерело даних, чи кілька джерел; чи з потока зчитує дані один процес, чи кілька процесів; як в потоці дані відрізняються за часом виникнення, споживання; як в потоці дані відрізняються між собою за порядком, як наприклад, в буфері FIFO; чи дозволяється в потоці дублювати дані або їх перезаписувати; якщо потік є чергою, чи дозволяється перевірка наявності даних в потоці, чи ні; чи розглядає модель час як безперервну сутність, коли сигнал – це функція від часу, чи як дискретні події, коли відлік сигналу представлений парою момент часу (номер такту) – значення, чи взагалі процеси розвиваються безвідносно до часу, коли сигнал – це впорядкована множина відліків; якщо потік є чергою, то чи кількість переданих даних в потік за певний проміжок часу дорівнює кількості спожитих даних, чи ні, або ця кількість змінюється динамічно; чи джерело й приймач даного звертаються до потоку одночасно [20].

Останні дві приведені вище властивості потоку даних характеризують його існування в часі. Розрізняють синхронні й асинхронні потоки та відповідно – синхронні й асинхронні моделі обчислень. В асинхронному потоці момент передачі даного є довільним і він допускає довільну затримку між завантаженням даного в потік і його споживанням. Це дає можливість будь-яким чином вибирати моменти

завантаження й споживання даних або спрацьовування відповідних акторів при складанні розкладу виконання алгоритму. В синхронному потоці момент передачі даного, або групи даних зв’язаний з деякою регулярно повторюваною подією, наприклад, сигналом синхросерії, тобто цей момент визначається дискретно. Якщо такий потік не має буферної пам’яті, то завантажене в нього дане повинне бути спожите негайно або з затримкою, що не перевищує періоду синхросерії. Синхронізм потоку може досягатись за допомогою логічних умов, тобто це особливий випадок асинхронного потоку. Зате аналіз синхронних потоків і складання для них розкладу значно спрощується в порівнянні з асинхронними потоками. Розрізняють моделі з синхронізацією по синхросерії та моделі з рандеву. В останніх джерело й приймач звертаються до потоку одночасно через механізм синхронізації з квотуванням.

Модель мережі процесів Кана характеризується тим, що вона абстрагована від часу й тому є потенційно асинхронною. Завдяки цьому вона може задавати алгоритми на більш високому рівні абстракції та бути трансформованою в асинхронну або синхронну модель, наприклад, при синтезі обчислювача. Крім того, ця модель має потоки у вигляді черги, тому може задавати алгоритми, в яких кількість даних в черзі змінюється динамічно. Наприклад, мережа процесів Кана дозволяє динамічно змінювати кількість переданих даних в потік. Через це глибина буфера потоку може бути необмеженою.

Таким чином, різні моделі обробки потоків даних, перш за все, можна класифікувати за ознаками їх потоків. Також їх можна класифікувати за будовою їхніх процесів, акторів, операторів, типом розкладу.

Розроблений метод має на меті мінімізувати кількість між операторного зв'язку, що відбувається у з'єднаннях з високою затримкою. Він об'єднує оператори, які вимагають широкої передачі даних і намагається розмістити отримані більші одиниці в мінімальну кількість хостів. Іншими словами, похідні розділи будуть виділені на один хост, якщо є один оснащений достатньою місткістю. В іншому випадку ці оператори, як очікується, будуть розміщені в хостах з меншою затримкою, щоб уникнути розширеної затримки.

Загальна архітектура системи, представлена на рисунку 4.1, складається з трьох абстрактних шарів. Нижній шар містить інфраструктурну інформацію щодо мережевих хостів. Хости розміщуються в тривимірній системі координат, в якій їх затримка мережі і доступність ресурсів своєчасно відображаються від їх розташування і відстаней між ними. Очікується, що окремі DSP зіставляться в мережних зв'язках з коротшою затримкою, і тому хости кластеризуються на основі їх спостережуваної затримки. Результат кластеризації динамічно оновлюється відповідно до зміни трафіку. Тим часом верхній шар збирає інформацію на рівні застосунків, який містить, але не обмежується затримкою операторів, попитом на ресурси та між операторним трафіком. Між прикладним шаром і інфраструктурним шаром вводимо шар розділу, який переводить метрики операторів в перегородки після застосування оператора злиття.

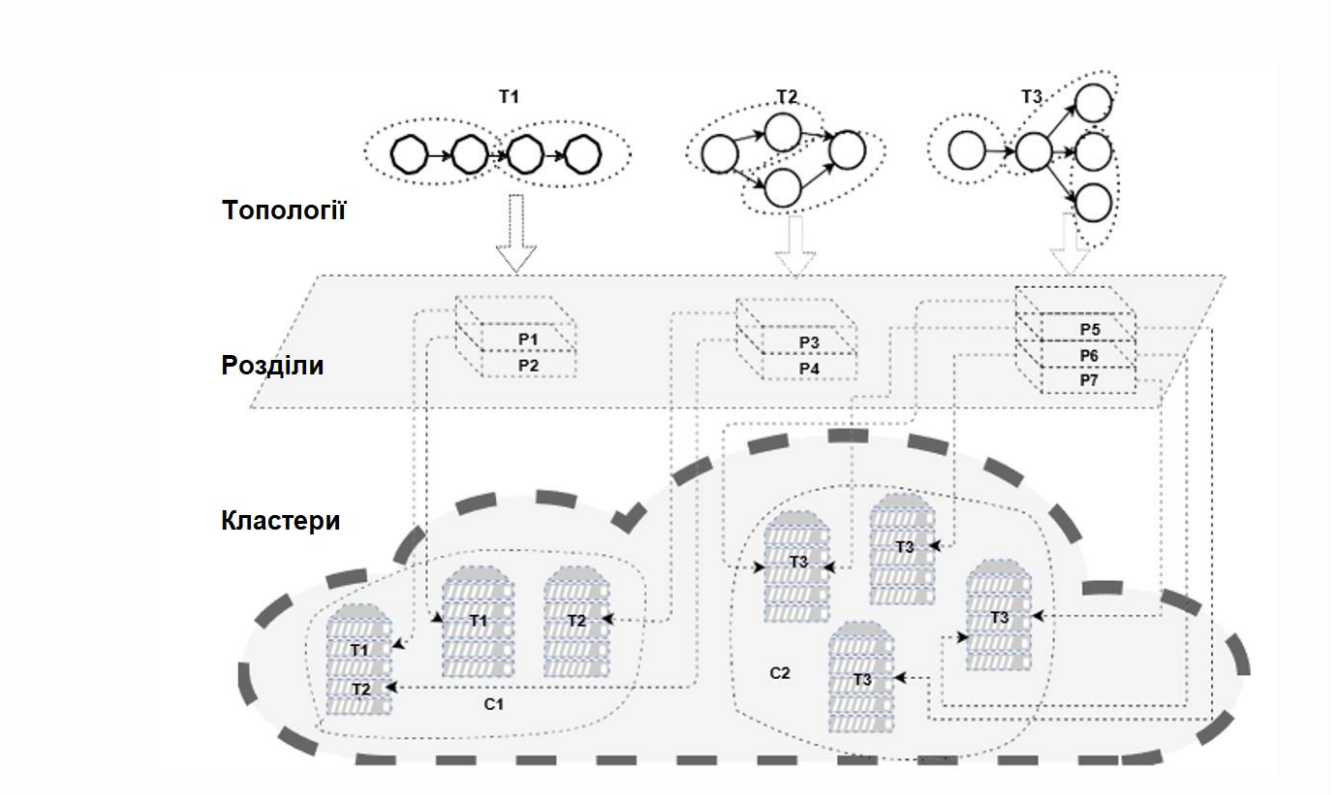


Рисунок 4.1 – Концептуальна схеми процесів, яка орієнтована на трафік та розподіл

Розроблений метод визначає зв'язок зіставлення між похідними розділами та кластерними хостами для прискорення обробки потоків. Зокрема, очікується, що тори, що належать до одного розділу, будуть розміщені в одному і тому ж хості,

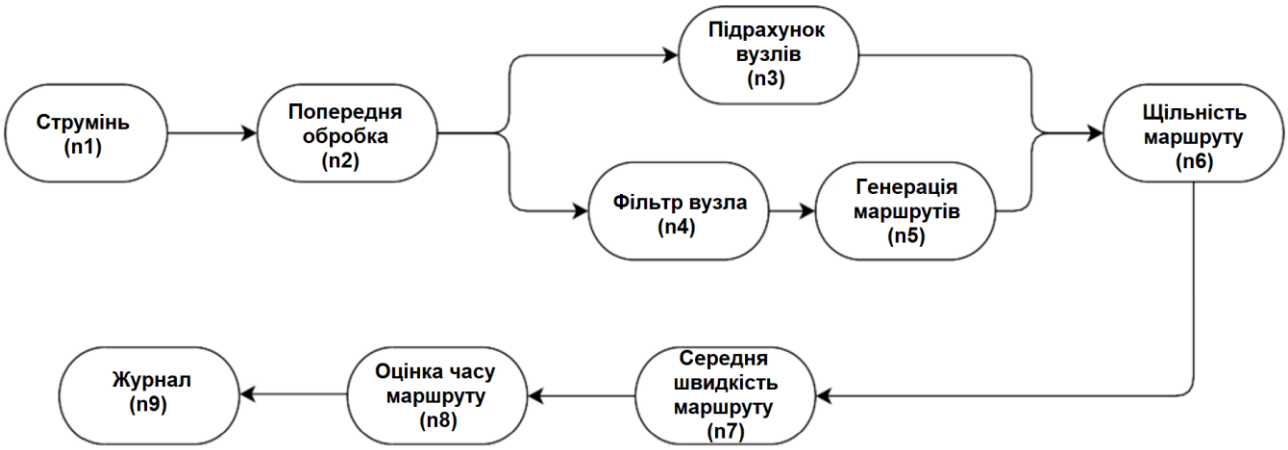
щоб уникнути непотрібної пропускної здатності, споживаної між операторним зв'язком. Кожного разу, коли він не може виявити такий хост зі значною ємністю, розділи будуть додатково розділені, поки всі похідні компоненти не можуть бути встановлені принаймні в один з активних хостів. Крім того, він спрямований на виділення операторів, що приходять з тієї ж топології в рамках конкретного кластера, щоб уникнути високої затримки зв'язку в застосунках DSP.

4.2 Розробка, реалізація та оцінка прототипу

Розроблений застосунок реалізовано згідно розробленого методу, орієнтованого на трафік і розподіл, щоб розмістити оператори на кластері Storm. Цей прототип можна розглядати як попередній крок для інформування в режимі реального часу. Крім того, він демонструє ефективність методу у вирішенні реальних сценаріїв. Розглянемо питання щодо проектування та реалізації прототипу.

Топологія для системи, яку розроблено, складається з дев'яти вузлів. На рисунку 4.2 показана топологічна форма програми та її оператори, як відправна точка безперервних потоків даних, Сервер отримує, як правило, відповідні дані із зовнішніх джерел (наприклад, датчиків, розподілених застосунків). Іноді він допомагає з попередньою нормою даних і гарантує, що наступні оператори можуть споживати потоки даних. Оскільки очікується, що прототип буде моделювати дані в режимі реального часу, вводимо чергу повідомлень, щоб кешувати нещодавно спостережувані пішохідні дані, а не використовувати кількість пішоходів безпосередньо на перехрестях. Зокрема, черга повідомлень приймає оригінальні дані, зібрані з датчиків, як вхідні дані. Динамічно розраховуємо середню кількість даних, що спостерігаються за хвилину на кожному перехресті, щоб чергою підштовхнути дані з отриманою швидкістю до топології. Іншими словами, сервер визначається як споживач черги повідомлень, тоді як кортеж вхідних даних

позначено ідентифікатором розташування. Крім того, введення черги повідомлень дозволяє системі призупинити обчислення, коли очікується серйозна деградація.



4.2 – Топологія дизайну оцінки часу

Обов'язково потрібно розуміти множину даних між будь-якою парою перехресть, щоб оцінити середній час обробки. Тому, потрібно попередньо обробити потоки даних і забезпечити, щоб дані з ідентичного датчика були спрямовані в той же пункт призначення. Крім того, враховуючи потенційні вигоди, які можна отримати від розпаралелізації даних, три операції (наприклад, підрахунок вузлів, фільтр вузлів і генерація маршруту) можна розглядати як область розпаралелювання. В результаті він дозволяє паралельне виконання і призводить до прискореної обробки. Зокрема, припускаючи, що для виконання паралельної області збереглися кілька вузлів (складається з 3, 4 і 5), визначаємо оператор, який реплікує вхідні дані і розподіляє вхідні дані на наступні паралельні канали. Багато потоковий оператор дозволяє уникнути непотрібного споживання пропускної здатності, зберігаючи паралелізм між географічно розподіленими вузлами. Таким чином, застосовано багато потоковий синтез оператора, щоб потенційно зменшити між вузловий трафік і підтримано багатоканальну станцію обслуговування для похідних розділів. Він представляє відповідний розділ після прикладного операторного синтезу і позначає його рівень паралелізму, потім змодельований як черга G/G/, а його оператори розміщуються з усвідомленням

ресурсної ємності (посилаючись на алгоритми). Оператори споживатимуть однаковий набір даних. Зокрема, операторний вузол-підрахунок підраховує загальну кількість на перехресті через налаштовувані часові проміжки. Індивідуальне значення дозволяє проводити аналіз та оцінку, посилаючись на конкретне часове вікно історичних даних. Тим часом вузол-фільтр відфільтровує дані, що спостерігаються на новому між секційному переході, і направляє оновлений список перехресть. Запропонований прототип підтримує оновлення інформації про перетин за допомогою вузла-фільтру, оскільки нові датчики можуть бути встановлені на більш пізньому етапі, і нещодавно зібрані дані будуть відповідно записані. Оператор маршрут-генерація потім визначає прилеглі перехрестя та визначає маршрути на основі інформації безпосередньо про карту попереднього завантаження.

Поєднуючи загальну кількість даних на кожному спостережуваному перехресті та оновлений набір маршрутів, оператор розраховує щільність маршруту. Щоб бути конкретним, він обчислює середнє число вздовж певного шляху протягом проміжку часу.

4.3 Розміщення операторів із визначенням стану

Розподі́лені обчи́слення (розподілена обробка даних) — спосіб розв'язання трудомістких обчислювальних завдань з використанням двох і більше комп'ютерів, об'єднаних в мережу.

Розподілені обчислення є окремим випадком паралельних обчислень, тобто одночасного розв'язання різних частин одного обчислювального завдання декількома процесорами одного або кількох комп'ютерів. Тому необхідно, щоб завдання, що розв'язується було сегментоване — розділене на підзадачі, що можуть обчислюватися паралельно. При цьому для розподілених обчислень доводиться також враховувати можливу відмінність в обчислювальних ресурсах, які будуть доступні для розрахунку різних підзадач. Проте, не кожне завдання можна

«розпаралелити» і прискорити його розв'язання за допомогою безпосередньо розподілених обчислень.

Різниця між HTC і HPC. HPC (англ. High Performance Computing, Високопродуктивні обчислення). HPC-системи зазвичай виконують близькозв'язані[уточнити] паралельні завдання, які має сенс запускати на обчислювальній системі зі з'єднаннями, що мають досить невеликі значення латентності. HTC (англ. High Throughput Computing, Обчислення високої пропускної здатності). HTC-системи, навпаки, призначені для незалежних, послідовних завдань, виконання кожної з яких можна планувати незалежно на великій кількості обчислювальних ресурсів, що входять безпосередньо в різні адміністративні організації.

Щоб знизити витрати часу, пов'язані з латентністю, потрібно розробляти алгоритми, що вимагають менше пересилань даних, так як це є мірою складності системи, а також групувати запити і відповіді; використовувати інформацію, розташовану «близько» в гіпермережі; кешувати, запитувати заздалегідь і дублювати інформацію (при цьому не варто забувати, що дані мають властивість застарівати); переміщати дані на ПК, де виконуються обчислення; виконувати обчислення там, де зберігаються дані. Це вимагає вирішення виникаючих питань, пов'язаних з безпекою та використанням приватних ресурсів та сервісів.

Дуже важливо враховувати топологію і розміщення даних, оскільки їх розмір може бути великим, і великі витрати на зв'язок будуть понесені для міграції таких операторів та даних. Крім того, передача станів через мережу споживає пропускну здатність і затримує обробку, що може ще більше погіршити загальну продуктивність. Для вирішення цієї проблеми розроблений метод, який враховує особливості топології і даних, актуалізується на цих особливостях і на враховує обов’язково вартість міграції. Зокрема, запропонований метод розглядає можливі місця розташування, отримані від схем управління ресурсами з різними показниками QoS, і обчислює вартість міґрації, викликану операторами. Він спрямований на вибір того, який має мінімальну вартість міграції серед доступних варіантів. В результаті запропонований метод підтримує системну інтеграцію з

різними схемами управління ресурсами, забезпечуючи мінімізацію витрат, викликаних міграцією станів. Розробка стратегії розміщення інтегрована з стратегією, що стосується трафіку та розподілів. Розроблений метод може бути інтегрований з альтернативними рішеннями. Моделювання розміщення оператора та станів враховують обсяги даних і топологію. Оскільки розміщення оператора, як правило, інтерпретується як розробка перез'єднання між операторами та хостами, то моделюємо розміщення операторів як двійкову матрицю для опису існуючого зв'язку між вузлами та їх розміщеними операторами. Представимо можливі схеми розміщення, отримані з стратегії управління ресурсами (наприклад, обізнаність QoS), а потім кожна включена також може бути представлена у вигляді бінарної матриці, яка показує розподіл операторів на активних вузлах. Крім того, припускаємо, що завжди є кілька потоків, які обробляються паралельно, і система надається хостами для обслуговування виконання. операторів в цілому. Нехай O(0, 1, ... −1) однозначно ідентифікує операторів і H(h0, h1, ... h−1) перераховує доступні хости. Потім існуюче розміщення E, а також можливі рішення про розміщення ∈ P змодельований як двійкова матриця з рядками і стовпці, що зображено відповідно на рисунку 4.3.

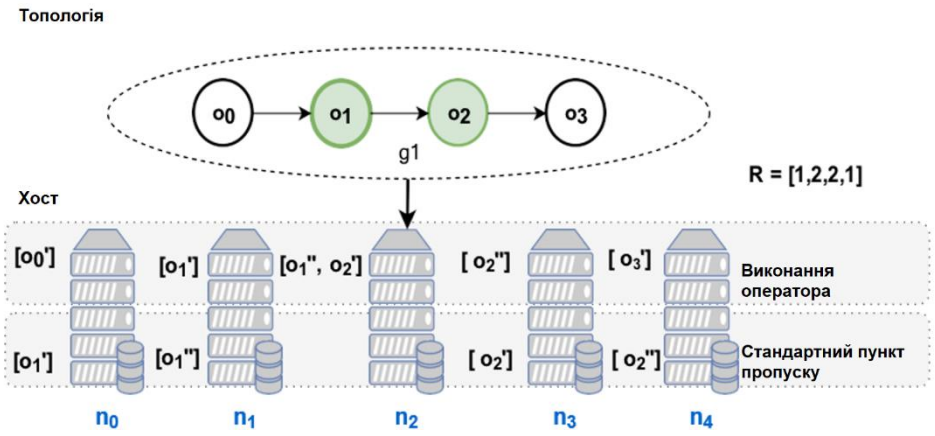


Рисунок 4.3 Приклад розміщення та моделі державної місцевості

За допомогою вищезазначеного матричного представлення існуючих та можливих схем розміщення можна зробити швидку оцінку витрат щодо коригувань, які необхідно зробити, особливо коли беруть участь оператори.

Враховуючи, що в DSP можуть бути різні цілі, а вимоги до обробки варіюються від застосування до заявки, запропоновано в розробленому методі обчислення вартості міграції від існуючого розміщення до запропонованих рішень про розміщення. Модель витрат підтримує інтеграцію зі схемами управління ресурсами щодо показників QoS або умов мережі, і розглядає стани як критичний показник для розробки ефективної стратегії розміщення операторів.

У той час як метод розміщення на основі трафіку та синтезу підвищує продуктивність обробки систем з великим навантаженням та різноманітними моделями трафіку, він в першу чергу фокусується на розміщенні для операторів без прив'язки до певних вузлів. Інтегруючи цей метод з обчисленням витрат на міграцію, очікується, що він забезпечить ефективне рішення для розміщення загальних станів потоків даних. Частково система координат була застосована для пошуку робочих вузлів і захоплення затримок мережі в режимі реального часу. Розширивши метод розміщення на основі трафіку та злиття з оцінкою витрат, що стосуються станової міграції та даних, включаємо використання ресурсів та витрати на повторну конфігурацію при прийнятті рішень про розміщення. В результаті формулюємо розміщення загальних станів потоків даних як задачу оптимізації з трьома основними цілями, які повинні бути задоволені: мінімізувати витрати на міграцію станів через вузли; мінімізувати вартість передачі даних між операторами; уникнення витрат, безпосередньо за допомогою викликаних частими повторними налаштуваннями.

Відібравши дві топології, які були включені в проект запуску системи з метою оцінки і макет стану показаний на рисунку 4.3 розглядаємо чотири оператори, які лінійно підключені, а саме сервер, часткова сума, принтер і загальний. У той час як сервер і принтер обробляють вхідні дані безпосередньо (відомі як оператори), визначаються як станові оператори, які по суті необхідні для регулярних контрольно-пропускних пунктів. Відповідно до стратегії управління станами система, її елементи створюються, якщо є хоча б один оператор, а також створюється додатковий потік для пересилання кортежу контрольно-пропускного пункту. Крім того, повідомлення про визнання записуються як сигнал успішної

обробки. Іншими словами, додатковий елемент сеситеми і потік дозволяють підтримувати стан і забезпечують узгодженість шляхом передачі кортежів контрольної точки. Крім того, топологія має аналогічний макет, за винятком останнього оператору. Таким чином, станова топологія має оператори стану. Така відмінність дозволяє обґрунтувати ефективність запропонованого методу в роботі з різноманітним співвідношенням станових операторів в застосунках DSP. Для подальшого вивчення кореляції між швидкістю прийому і відповідною продуктивністю змінимо сервер в обох топологіях, щоб дозволити їм налаштувати частоту обробки даних. Зокрема, встановлюємо частоту як 1 кортеж / секунду і 1000 кортежів / секунду відповідно.

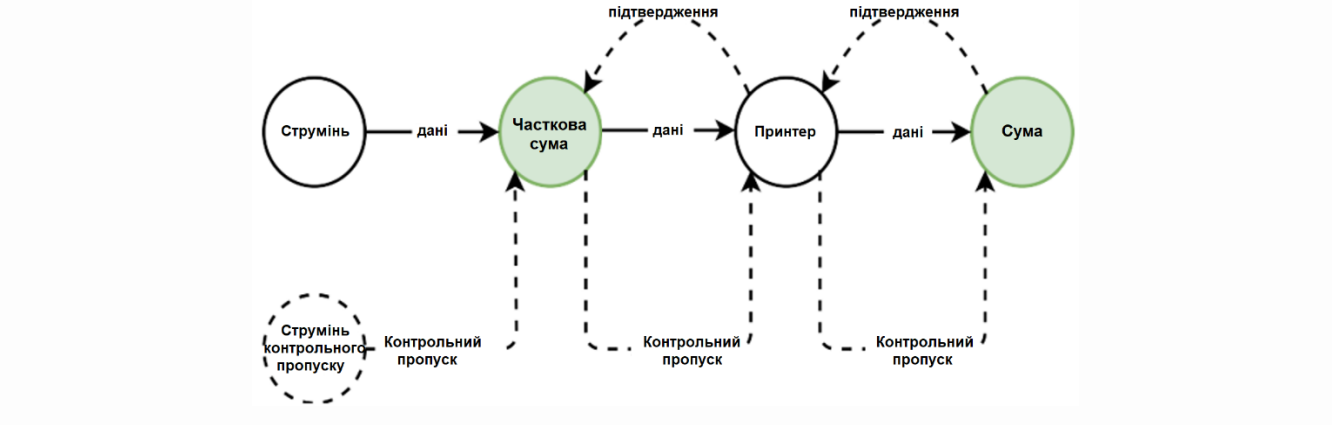


Рисунок 4.4 – Макет топології

Спочатку було розгорнуто першу топологію і зібрано метрики застосунків протягом виконання 20 хвилин. Сервер розпочинається з швидкості обробки 1 кортеж / секунду. На рисунках 4.3 і 4.4. показана затримка системи і пропускна здатність станової топології над виконанням. Лінії на рисунку 4.3 зображують рівень затримки, а складені області на рисунку 4.4 показують різницю пропускної здатності між розробленим методом і функцією балансування. Зокрема, оскільки обидва методи були розпочаті з єдиною схемою планування, то вони мають подібну поведінку. Після того, як система має тенденцію стабілізуватися і була зібрана відповідна інформація, функція перебалансування, а також запропонований метод, була застосована з 2-ї хвилини. Різке збільшення затримки в обох методах було в основному викликано міграцією операторів, що, відповідно, призводить до

різноманітного рівня деградації пропускної здатності. Зокрема, в той час як обидва методи зазнали тривалої затримки через міграцію, середня затримка була дещо покращена після того, як запропонований метод був значно покращений, а пропускна здатність була значно збільшена порівняно з функцією перебалансування. В основному це викликано обізнаністю про розміщення станів операторів і зниженням витрат на міграцію.

4.4 Висновки до четвертого розділу

Розроблено прототип для підтримки аналізу в режимі реального часу та їх поведінки. Даний сценарій має різноманітну модель між операторного зв'язку, а його ефективність сильно залежить від базового стану мережі. Розроблений метод інтегрували з трафіком та розділеннями, щоб розмістити операторів прототипу. Результати оцінки показують ефективність застосування методу трафік-мережі до прототипу, особливо з точки зору підвищення рівня. Ефективна система аналізу поведінки може розробити оптимальну маршрутизацію шляху.

ВИСНОВКИ

У ході досліджень було досліджено та розроблено метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних.

У першому розділі за результатами проведеного аналізу щодо розподілу обчислювальних ресурсів для обробки розподілених потоків даних було встановлено межі предметної області дослідження, визначено поняття потокової обробки, а також виділено недоліки у відомих рішеннях щодо організації кластеру та його складових компонентів, стратегіях розподілу та групування даних, керуванні станами та обробка даних, керуванні ресурсами при обробці даних та особливостях схем планування, що впливають на досягнення ефективності.

У другому розділі представляється дві математичні моделі, що стосуються пропускної здатності та затримки, і описує модельний авто масштабувальник, призначений для розподілених застосунків обробки потоків. Зокрема, модель затримки була розроблена на основі теорії черг, яка моделює окремих операторів як станції технічного обслуговування G/G/.

У третьому розділі в результаті було оцінено ефективність запропонованого планувальника в Apache Storm з трьома тестовими застосунками, що проникають в режимі реального часу. Програми охоплюють цілий ряд різних макетів операторів і різноманітний рівень толерантності до затримки, щоб забезпечити різноманітність застосунків DSP, які розглядаються в наших оцінках.

У четвертому розділі розроблено прототип для підтримки аналізу в режимі реального часу та їх поведінки. Даний сценарій має різноманітну модель між операторного зв'язку, а його ефективність сильно залежить від базового стану мережі. Розроблений метод інтегрували з трафіком та розділеннями, щоб розмістити операторів прототипу.

В результаті проведених досліджень та завдань з усунення недоліків відомих підходів було [95] розроблено метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних з метою досягнення ефективності обробки. Розроблений метод розподілу обчислювальних ресурсів для обробки розподілених

потоків даних дає змогу досягти ефективності обробки, на відміну від відомих методів, за рахунок інтегрованого поєднання схем планування та балансування навантаження між обчислювальними ресурсами. Практична цінність роботи полягає в можливості реалізації розробленого методу розподілу обчислювальних ресурсів для обробки розподілених потоків даних та практичному застосуванні розроблених систем, що в результаті дало можливість покращити ефективність обробки на 3-5%. Для перевірки ефективності запропонованих рішень було розроблено систему згідно розробленого методу та проведено з нею експерименти.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Daniel J Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag Maskey, Alex Rasin, Esther Ryvkina, et al. The design of the borealis stream processing engine. In Cidr, volume 5, pages 277–289, 2005.

2. The city of melbourne, walking plan 2014–17 [Електронний ресурс]. - Режим доступу: http://participate. melbourne.vic.gov.au/walkingplan. November 2014.

3. Daniel J Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Con-vey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. The Journal The International Journal on Very Large Data Bases (VLDB), 12(2):120–139, 2003.

4. Arnold O Allen. Probability, statistics, and queueing theory. Academic Press, 2014.

5. Henrique C. M. Andrade, Buğra Gedik, and Deepak S. Turaga. Introduction to stream processing, pages 33–74. Cambridge University Press, 2014.

6. Henrique CM Andrade, Buğra Gedik, and Deepak S Turaga. Fundamentals of Stream Processing: Application Design, Systems, and Analytics. Cambridge University Press, 2014.

7. Leonardo Aniello, Roberto Baldoni, and Leonardo Querzoni. Adaptive online scheduling in storm. In Proceedings of the 7h ACM international conference on Distributed event-based systems, pages 207–218, 2013.

8. Apache Software Foundation. Storm, an open source distributed real-time com- putation system. [Електронний ресурс]. - Режим доступу: http://storm.apache.org/, 2016.

9. Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Itaru Nishizawa, Justin Rosenstein, and Jennifer Widom. Stream: the stanford stream data manager (demonstration description). In Proceedings of the ACM SIGMOD international conference on Management of data, pages 665–665, 2003.

10. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. Computer networks, 54(15):2787–2805, 2010.

11. Ron Avnur and Joseph M. Hellerstein. Eddies: Continuously adaptive query processing. In SIGMOD International Conference on Management of Data, pages 261–272, NY, USA, 2000. ACM.

12. Ron Avnur and Joseph M Hellerstein. Eddies: Continuously adaptive query processing. In ACM sigmod record, volume 29, pages 261–272. ACM, 2000.

13. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In Proceedings of the twenty first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages 1–16, 2002.

14. Brian Babcock, Mayur Datar, and Rajeev Motwani. Load shedding for aggregation queries over data streams. In Proceedings. 20^h IEEE international conference on data engineering, pages 350–361, 2004.

15. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In ACM SIGOPS operating systems review, volume 37, pages 164–177, 2003.

16. Michael Batty. Cities and complexity: understanding cities with cellular automata, agent-based models, and fractals. The MIT press, 2007.

17. Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future generation computer systems, 28(5):755–768, 2012.

18. Alain Biem, Eric Bouillet, Hanhua Feng, Anand Ranganathan, Anton Riabov, Olivier Verscheure, Haris Koutsopoulos, and Carlos Moran. Ibm infosphere streams for scalable, real-time, intelligent transportation services. In Proceedings of the ACM SIGMOD International Conference on Management of data, pages 1093–1104, 2010.

19. Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S Trivedi. Queueing networks and Markov chains: modeling and performance evaluation with

computer science applications. John Wiley & Sons, 2006.

20. Dhruba Borthakur. The hadoop distributed file system: Architecture and design. Hadoop Project Website, 11(2007):21, 2007.

21. Luca Bruno and Fiammetta Venuti. The pedestrian speed-density relation: modelling and application. Proceedings of Footbridge, 2008.

22. Stefan Buchmueller and Ulrich Weidmann. Parameters of pedestrians, pedestrian traffic and walking facilities. IVT Schriftenreihe, 132, 2006.

23. Rajkumar Buyya. Economic-based distributed resource management and scheduling for grid computing. arXiv preprint cs/0204048, 2002.

24. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5^h utility. Future Generation computer systems, 25(6):599–616, 2009.

25. Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications'qos. IEEE Transactions on Cloud Computing, 3(4):449–458, 2014.

26. Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, and Matteo Nardelli. Optimal operator replication and placement for distributed stream processing systems. ACM SIGMETRICS Performance Evaluation Review, 44(4):11–22, 2017.

27. Valeria Cardellini, Francesco Lo Presti, Matteo Nardelli, and Gabriele Russo Russo. Optimal operator deployment and replication for elastic distributed data stream processing. Concurrency and Computation: Practice and Experience, 30(9):e4334, 2018.

28. Valeria Cardellini, Francesco Lo Presti, Matteo Nardelli, and Gabriele Russo Russo. Auto-scaling in data stream processing applications: A model-based reinforcement learning approach. In Workshop on New Frontiers in Quantitative Methods in Informatics, pages 97–110. Springer, 2017.

29. Don Carney, Uğur Çetintemel, Alex Rasin, Stan Zdonik, Mitch Cherniack, and Mike Stonebraker. Operator scheduling in a data stream manager. In Proceedings

of the 29^h international conference on Very large data bases-Volume 29, pages 838–849. VLDB Endowment, 2003.

30. Eddy Caron, Frédéric Desprez, and Adrian Muresan. Forecasting for Cloud computing on-demand resources based on pattern matching. PhD thesis, INRIA,2010.

31. Eddy Caron, Frédéric Desprez, and Adrian Muresan. Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. Journal of Grid Computing, 9(1):49–64, 2011.

32. Raul Castro Fernandez, Matteo Migliavacca, Evangelia Kalyvianaki, and Peter Pietzuch. Integrating scale out and fault tolerance in stream processing using operator state management. In Proceedings of the ACM SIGMOD international conference on Management of data, pages 725–736, 2013.

33. Chialin Chang, Bongki Moon, Anurag Acharya, Carter Shock, Alan Sussman, and Joel Saltz. Titan: a high-performance remote-sensing database. In Proceed-ings 13^h IEEE International Conference on Data Engineering, pages 375–384,1997.

34. Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In NSDI, volume 8, pages 337–350, 2008.

35. Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Prac- tical, distributed network coordinates. ACM SIGCOMM Computer Communication Review, 34(1):113–118, 2004.

36. Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR), 44(3):15, 2012.

37. Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In ACM SIGCOMM Computer Communication Review, volume 34, pages 15–26, 2004.

38. Marco Danelutto and Massimo Torquati. Structured parallel programming with“core” fastflow. In Central European functional programming school, pages 29–

75. Springer, 2013.

39. Miyuru Dayarathna and Srinath Perera. Recent advancements in event processing. ACM Computing Surveys (CSUR), 51(2):33, 2018.

40. Marcos Dias de Assuncao, Alexandre da Silva Veith, and Rajkumar Buyya. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. Journal of Network and Computer Applications, 103:1–17, 2018.

41. Tiziano De Matteis and Gabriele Mencagli. Keep calm and react with foresight: Strategies for low-latency & energy-efficient elastic data stream processing. In SIGPLAN Symp. on Principles & Practice of Parallel Programming, pages 1–12, NY, USA, 2016. ACM.

42. Tiziano De Matteis and Gabriele Mencagli. Proactive elasticity and energy awareness in data stream processing. Journal of Systems and Software, 2016.

43. Tiziano De Matteis and Gabriele Mencagli. Elastic scaling for distributed latency-sensitive data stream operators. In 25^h Euromicro IEEE International Conference on Parallel, Distributed and Network-based Processing (PDP), pages 61–68, 2017.

44. Tiziano De Matteis and Gabriele Mencagli. Parallel patterns for window-based stateful operators on data streams: an algorithmic skeleton approach. International Journal of Parallel Programming, 45(2):382–401, 2017.

45. Tiziano De Matteis and Gabriele Mencagli. Proactive elasticity and energy awareness in data stream processing. Journal of Systems and Software, 127:302–319, 2017.

46. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008.

47. David J DeWitt and Jim Gray. Parallel database systems: The future of high performance database processing. Technical report, University of Wisconsin Madison Department of Computer Sciences, 1992.

48. Jianbing Ding, Tom Z. J. Fu, Richard T. B. Ma, Marianne Winslett, Yin Yang, Zhenjie Zhang, and Hongyang Chao. Optimal operator state migration for

elastic data stream processing. ArXiv, abs/1501.03619, 2015.

49. György Dósa. The tight bound of first fit decreasing bin-packing algorithm is FFD(I) 11/9OPT(I)+ 6/9. In International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies, pages 1–11. Springer, 2007.

50. Xavier Dutreilh, Aurélien Moreau, Jacques Malenfant, Nicolas Rivierre, and Isis Truck. From data center resource allocation to control theory and back. In Cloud Computing (CLOUD), IEEE 3^{d} International Conference on, pages 410–417, 2010.

51. Leila Eskandari, Zhiyi Huang, and David Eyers. P-scheduler: adaptive hierarchical scheduling in apache storm. In Proceedings of the Australasian Computer Science Week Multiconference, page 26. ACM, 2016.

52. M Reza Hoseiny Farahabady, Hamid R Dehghani Samani, Yidan Wang, Albert Y Zomaya, and Zahir Tari. A qos-aware controller for apache storm. In Network Computing and Applications (NCA), 15^{h} IEEE International Symposium on, pages 334–342, 2016.

53. Yi-Hsuan Feng, Nen-Fu Huang, and Yen-Min Wu. Efficient and adaptive stateful replication for stream processing engines in high-availability cluster. IEEE Transactions on Parallel and Distributed Systems, 22(11):1788–1796, 2011.

54. Ian Foster and Carl Kesselman. The Grid 2: Blueprint for a new computing infrastructure. Elsevier, 2003.

55. Martin Fowler. Patterns of enterprise application architecture. Addison-WesleyLongman Publishing Co., Inc., 2002.

56. Lawrence D Frank, James F Sallis, Terry L Conway, James E Chapman, Brian E Saelens, and William Bachman. Many pathways from land use to health: associations between neighborhood walkability and active transportation, body mass index, and air quality. Journal of the American planning Association, 72(1):75–87, 2006.

57. Tom ZJ Fu, Jianbing Ding, Richard TB Ma, Marianne Winslett, Yin Yang, andZhenjie Zhang. Drs: dynamic resource scheduling for real-time analytics over fast streams. In 35^{h} IEEE International Conference on Distributed Computing Systems,

pages 411–420, 2015.

58. Ron Gabor, Shlomo Weiss, and Avi Mendelson. Fairness enforcement in switchon event multithreading. Trans. Archit. Code Optim., 4 (3), 2007.

59. Buğra Gedik. Partitioning functions for stateful data parallelism in stream processing. The Journal The International Journal on Very Large Data Bases (VLDB), 23(4):517–539, 2014.

60. Bugra Gedik, Henrique Andrade, Kun-Lung Wu, Philip S Yu, and Myungcheol Doo. Spade: the system s declarative stream processing engine. In Proceedings of the ACM SIGMOD international conference on Management of data, pages 1123–1134, 2008.

61. Bugra Gedik, Scott Schneider, Martin Hirzel, and Kun-Lung Wu. Elastic scaling for data stream processing. IEEE Transactions on Parallel and Distributed Systems, 25(6):1447–1463, June 2014.

62. Bugra Gedik, Scott Schneider, Martin Hirzel, and Kun-Lung Wu. Elastic scal- ing for data stream processing. IEEE Transactions on Parallel & Distributed Systems, (1):1–1, 2014.

63. Billie Giles-Corti and Robert J Donovan. The relative influence of individual, social and physical environment determinants of physical activity. Social science& medicine, 54(12):1793–1812, 2002.

64. Zhenhuan Gong, Xiaohui Gu, and John Wilkes. Press: Predictive elastic resource scaling for cloud systems. In IEEE International Conference on Networkand Service Management, pages 9–16, 2010.

65. Michael I Gordon, William Thies, and Saman Amarasinghe. Exploiting coarse-grained task, data, and pipeline parallelism in stream programs. ACM SIGARCH Computer Architecture News, 34(5):151–162, 2006.

66. Ryan Greene-Roesel, Mara Chagas Diogenes, David R Ragland, and Luis An- tonio Lindau. Effectiveness of a commercially available automated pedestrian counting device in urban environments: Comparison with manual counts. 2008.

67. Donald Gross. Fundamentals of queueing theory. John Wiley & Sons, 2008.

68. Vincenzo Gulisano, Ricardo Jimenez-Peris, Marta Patino-Martinez, et al. Streamcloud: An elastic and scalable data streaming system. IEEE Trans- actions on Parallel and Distributed Systems, 23(12):2351–2365, 2012.

69. Rui Han, Li Guo, Moustafa M Ghanem, and Yike Guo. Lightweight resource scaling for cloud applications. In Cluster, Cloud and Grid Computing (CCGrid),12^h IEEE/ACM International Symposium on, pages 644–651, 2012.

70. John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1):100–108, 1979.

71. Thomas Heinze, Valerio Pappalardo, Zbigniew Jerzak, and Christof Fetzer. Auto-scaling techniques for elastic data stream processing. In 30^h IEEE International Conference on Data Engineering Workshops, pages 296–302, 2014.

72. Nikolas Roman Herbst, Nikolaus Huber, Samuel Kounev, and Erich Amrehn. Self-adaptive workload classification and forecasting for proactive resource pro- visioning. Concurrency and computation: practice and experience, 26(12):2053–2078, 2014.

73. Thomas Hiessl, Vasileios Karagiannis, Christoph Hochreiner, Stefan Schulte, and Matteo Nardelli. Optimal placement of stream processing operators in the fog. In 3^d IEEE International Conference on Fog and Edge Computing (ICFEC), pages 1–10, 2019.

74. Martin Hirzel, Robert Soulé, Scott Schneider, Buğra Gedik, and Robert Grimm. A catalog of stream processing optimizations. ACM Computing Surveys, 46(4):46, 2014.

75. Mohammad Reza Hoseiny Farahabady, Young Choon Lee, and Albert Y. Zomaya. Pareto-optimal cloud bursting. IEEE Transactions on Parallel and Distributed Systems, 25 (10):2670–2682, 2014.

76. Xiaoran Huang, Marcus White, and Mark Burry. A pedestrian-centric design strategy: melding reactive scripting with multi-agent simulation. In Proceedingsof the Symposium on Simulation for Architecture and Urban Design,

page 34. Society for Computer Simulation International, 2017.

77. Patrick Hunt, Mahadev Konar, Flavio Paiva Junqueira, and Benjamin Reed. Zookeeper: Wait-free coordination for internet-scale systems. In USENIX an- nual technical conference, volume 8, page 9. Boston, MA, USA, 2010.

78. Bart Jacob, Richard Lanyon-Hogg, Devaprasad K Nadgir, and Amr F Yassin. Apractical guide to the ibm autonomic computing toolkit. IBM Redbooks, 4(10), 2004.

79. Navendu Jain, Lisa Amini, Henrique Andrade, Richard King, Yoonho Park, Philippe Selo, and Chitra Venkatramani. Design, implementation, and evaluation of the linear road bnchmark on the stream processing core. In Proceedings of the ACM SIGMOD international conference on Management of data, pages 431–442, 2006.

80. Brendan Jennings and Rolf Stadler. Resource management in clouds: Survey and research challenges. Journal of Network and Systems Management, 23(3):567–619, 2015.

81. Anthony D JoSEP, RAnDy KAtz, AnDy KonWinSKi, LEE Gunho, DAViD PAttERSon, and ARiEL RABKin. A view of cloud computing. Communicationsof the ACM, 53(4), 2010.

82. Apache Kafka. A high-throughput distributed messaging system. URL: kafka. apache. org as of, 5(1), 2014.

83. Evangelia Kalyvianaki, Themistoklis Charalambous, and Steven Hand. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In Proceedings of the 6^h ACM international conference on Autonomic computing, pages 117–126, 2009.

84. James M Kaplan, William Forrest, and Noah Kindler. Revolutionizing data center energy efficiency. Technical report, Technical report, McKinsey & Company, 2008.

85. George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing, 20(1):359–392, 1998.

86. George Karypis and Vipin Kumar. A software package for partitioning un-

structured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. University of Minnesota, Department of Computer Science and Engineering, Army HPC Research Center, Minneapolis, MN, 1998.

87. Rohit Khandekar, Kirsten Hildrum, Sujay Parekh, Deepak Rajan, Joel Wolf, Kun-Lung Wu, Henrique Andrade, and Buğra Gedik. Cola: Optimizing stream processing applications via graph partitioning. In ACM/IFIP/USENIX Inter- national Conference on Distributed Systems Platforms and Open Distributed Processing, pages 308–327. Springer, 2009.

88. Abbas Eslami Kiasari, Zhonghai Lu, and Axel Jantsch. An analytical latency model for networks-on-chip. IEEE Transactions on Very Large Scale Integration(VLSI) Systems, 21(1):113–123, 2013.

89. John FC Kingman. On queues in heavy traffic. Journal of the Royal Statistical Society: Series B (Methodological), 24(2):383–392, 1962.

90. Paweł Koperek and Włodzimierz Funika. Dynamic business metrics-driven re- source provisioning in cloud environments. In International Conference on Parallel Processing and Applied Mathematics, pages 171–180. Springer, 2011.

91. YongChul Kwon, Magdalena Balazinska, and Albert Greenberg. Fault-tolerant stream processing using a distributed, replicated file system. Proceedings of theVLDB Endowment, 1(1):574–585, 2008.

92. Boduo Li, Yanlei Diao, and Prashant Shenoy. Supporting scalable analytics with latency constraints. Proceedings of the VLDB Endowment, 8(11):1166 – 1177, 2015.

93. Teng Li, Zhiyuan Xu, Jian Tang, and Yanzhi Wang. Model-free control for dis-tributed stream data processing using deep reinforcement learning. Proceedings of the VLDB Endowment, 11(6):705–718, 2018.

94. Shengchao Liu, Jianping Weng, Jessie Hui Wang, Changqing An, Yipeng Zhou, and Jilong Wang. An adaptive online scheme for scheduling and resource enforcement in storm. IEEE/ACM Transactions on Networking, 2019.

95. Дрозд А. І., Форкун Ю.В. Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних // Зб. наук. пр. наукової конференції «АПКН-

2021». Хмельницький ХНУ. – 2021. – С. 319 - 320. [Електронний ресурс]. - Режим доступу: https://kn.khmnu.edu.ua/page.aspx?r=3&p=6

ДОДАТОК А

(Обов’язковий)

ТЕЗИ НАУКОВОЇ РОБОТИ

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XIII Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2021»

15-16 жовтня 2021

Хмельницький 2021

УДК 004.4

Дрозд А. І., Форкун Ю. В.

Хмельницький національний університет

МЕТОД РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ ДЛЯ ОБРОБКИ РОЗПОДІЛЕНИХ ПОТОКІВ ДАНИХ

Розглянуто та пропонуються до розробки метод розподілу обчислювальних ресурсів при обробці розподілених потоків даних. В задачах обробки розподілених потоків даних суттєво втрачається ефективність при отриманні остаточного результату через недостатню узгодженість при розподіленні ресурсів. Тому, цьому питанню приділяється увага дослідників. Розроблений метод покращує ефективність обробки розподілених потоків даних.

The development of the distribution of computing resources in the processing of distributed data streams is considered and proposed for development. In the tasks of processing distributed data streams, the efficiency in obtaining the final result is significantly lost due to insufficient consistency in the allocation of resources. Therefore, this issue is the focus of researchers. The developed method improves the efficiency of processing distributed data streams.

Покращення рівня інформатизації суспільства створює широкі можливості для залучення обчислювальних ресурсів користувачів в різних суттєво складніших задач, ніж ті для яких планувалось їх використовувати. На сьогодні актуальним напрямом досліджень з метою ефективного використання обчислювальних ресурсів є їх застосування для обробки розподілених потоків даних. При організації роботи з розподіленими потоками даних виникає проблема організації ефективних розподілених обчислень. Від цієї ефективності залежать очікувані результати.

Метою роботи є розробка методу розподілу обчислювальних ресурсів при обробці розподілених потоків даних.

Сенсоризація реального світу дає можливість моніторингу в реальному часі та інтерактивного контролю в широкому діапазоні областей (наприклад, військових, охорони здоров'я та техніки). Дані, які спочатку генеруються з географічно розподілених датчиків, зазвичай обробляються з великим обсягом [1], і очікується, що вони оброблятимуть безперервні потоки даних майже в режимі реального часу [2, 3]. Однак широко застосовувані традиційні рішення з управління даними (наприклад, системи керування реляційними базами даних) не здатні задовольнити зростаючу потребу в миттєвій обробці, що стосується зростаючого попиту на базові обчислення, а також на ємність зберігання та непередбачувані запити користувачів [4]. Тому пропонується механізм обробки потоків для роботи з потоками даних «на

льоту» за допомогою безперервних запитів [5], а для роботи з необмеженими потоками даних було запропоновано кілька фреймворків обробки потоків даних. Вони в першу чергу спрямовані на надання результатів обробки з низькою затримкою та високою пропускною здатністю. У той час як менша затримка гарантує, що кортежі даних можуть бути оброблені миттєво та направлені на подальшу обробку, більш високий рівень пропускної здатності свідчить про те, що система здатна обробляти величезну кількість даних. Для прискорення обробки потоку або підвищення рівня пропускної здатності були застосовані методи, включаючи переупорядкування/розділення оператора, розпаралелізацію конвеєра.

Таким чином, розробка розробки методу розподілу обчислювальних ресурсів при обробці розподілених потоків даних покращує ефективність обробки потоків даних.

Перелік посилань

1. Henrique CM Andrade, Buğra Gedik, and Deepak S Turaga. Fundamentals of Stream Processing: Application Design, Systems, and Analytics. Cambridge University Press, 2014.
2. Leonardo Aniello, Roberto Baldoni, and Leonardo Querzoni. Adaptive online scheduling in storm. In Proceedings of the 7 ACM international conference on Distributed event-based systems, pages 207–218, 2013.
3. Apache Software Foundation. Storm, an open source distributed real-time com- putation system. http://storm.apache.org/, 2016.
4. Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Keith Ito, Itaru Nishizawa, Justin Rosenstein, and Jennifer Widom. Stream: the stanford stream data manager (demonstration description). In Proceedings of the ACM SIGMOD international conference on Management of data, pages 665–665, 2003.
5. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. Computer networks, 54(15):2787–2805, 2010.

Дані про авторів:

ПІБ автора	Телефон	Email
Дрозд Андрій Ігорович	+380982718334	andriydrozdit@gmail.com
Форкун Юрій Вікторович	+380503764019	forkun@ridne.net

ДОДАТОК Б

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних

Автор роботи:
Студент групи ІПЗм-20-1
Дрозд А. І.
Керівник роботи:
к.т.н. Медзатий Д. М.

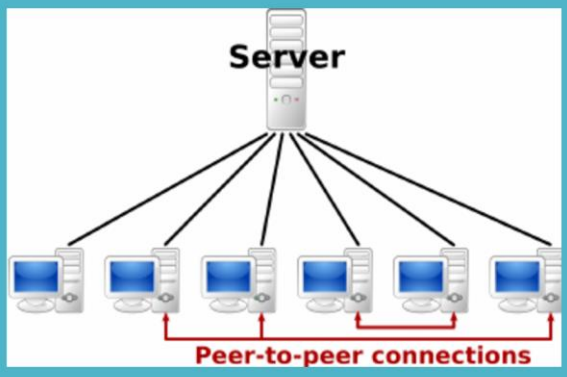
Мета роботи: розробити метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних з метою досягнення ефективності обробки

Об'єктом дослідження є обчислювальних ресурсів для обробки розподілених потоків даних.

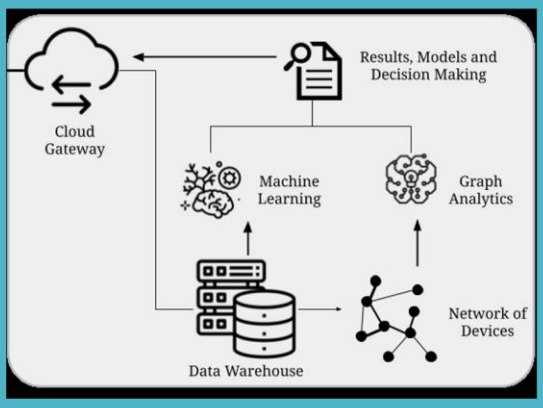
Наукова новизна роботи: розроблений метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних дає змогу досягти ефективності обробки, на відміну від відомих методів, за рахунок інтегрованого поєднання схем планування та балансування навантаження між обчислювальними ресурсами.

Для досягнення поставленої мети необхідно:
1) розробити модель затримки оператора;
2) розробити модель пропускної здатності системи з врахуванням особливостей проаналізованих недоліків в цій предметній області;
3) здійснити моделювання системи та результати експериментів з моделлю;
4) розробити метод та алгоритми розподілу обчислювальних ресурсів для обробки розподілених потоків даних;
5) здійснити реалізацію запропонованого рішення.

Загальна практика управління ресурсами в хмарі для проведення зростаючих великомасштабних обчислювальних завдань в областях, починаючи від науково-дослідних, медичних, фінансових та інженерних, обчислювальна інфраструктура Grid і peer-to-peer (P2P) систем в основному необхідні для обміну обчислювальними, мережевими та складськими ресурсами, географічно розподіленими між установами або організаціями.

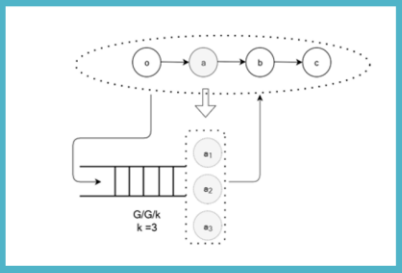


SIS архітектура

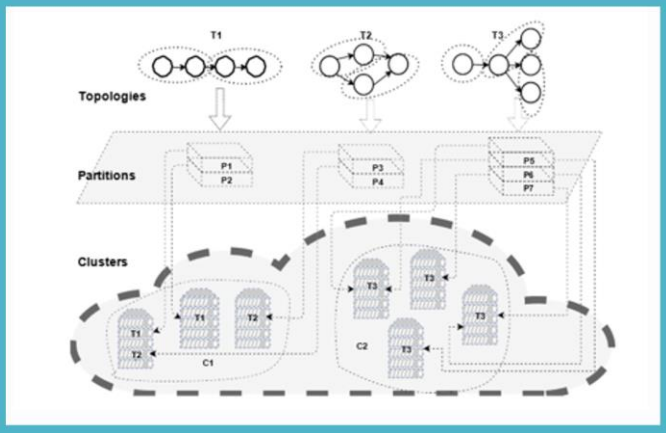


Предикативна аналітика

Легко помітити, що обробка може бути потенційно прискорена за рахунок збільшення паралелізму. певних операторів, особливо якщо оператор є вузьким місцем. Модель черги G/G/k.



Концептуальна схеми процесів, яка орієнтована на трафік та поділ



Висновок

В результаті проведених досліджень та завдань з усунення недоліків відомих підходів було розроблено метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних з метою досягнення ефективності обробки. Розроблений метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних дає змогу досягти ефективності обробки, на відміну від відомих методів, за рахунок інтегрованого поєднання схем планування та балансування навантаження між обчислювальними ресурсами. Практична цінність роботи полягає в можливості реалізації розробленого методу розподілу обчислювальних ресурсів для обробки розподілених потоків даних та практичному застосуванні розроблених систем, що в результаті дало можливість покращити ефективність обробки на 3-5%. Для перевірки ефективності запропонованих рішень було розроблено систему згідно розробленого методу та проведено з нею експерименти.

Sat Dec 04 15:21:35 EET 2021, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

ID: 98035 Назва: Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних Додано в БД: 2021-12-04 Автора: А.І. Дрозд Керівники: Д.М. Медзатий Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	155438	1184	5593 (4%)	63 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1009522658

Дата перевірки:
04.12.2021 16:16:46 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.12.2021 16:18:20 EET

ID користувача:
100005589

Назва документа: дипломна_робота_Дрозд_Андрій1_2

Кількість сторінок: 75 **Кількість слів:** 18224 **Кількість символів:** 141917 **Розмір файлу:** 1.31 MB **ID файлу:** 1009533639

12.1%
Схожість

Найбільша схожість: 2.4% з джерелом з Бібліотеки (ID файлу: 1009532628)

9.94% Джерела з Інтернету	289	Сторінка 77
4.88% Джерела з Бібліотеки	92	Сторінка 81

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	9

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник студент групи ІПЗм-20-1 Дрозд А.І.

Тема Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг дипломної роботи:

Кількість листів креслень 0 ; кількість сторінок записки 89

1.Короткий зміст ДР та прийнятих рішень Представлена робота присвячена актуальній темі в області обробки потоків даних, і складається з наступних розділів: вступ, аналіз розподіленої паралельної обробки даних, розгляд моделей системи, метод та програма системи розподілу обчислювальних ресурсів для обробки розподілених потоків даних, висновок, додатки.

2. Висновок про відповідність ДР поставленому завданню Магістерська кваліфікаційна робота виконана у відповідності до завдань із дотриманням усіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі студент провів детальний аналіз предметної області, дослідив існуючі методи обробки розподілених потоків, описав їх особливості, переваги та недоліки, зробив висновок як можна удосконалити існуючі рішення. В другому розділі проведено аналіз моделей системи затримки оператора та пропускної здатності системи для покращення оптимізації. У третьому та четвертому розділах проведено тестування та проектування майбутньої системи.

4. Позитивні сторони роботи До позитивних сторін роботи слід віднести глибоке дослідження існуючих методів обробки розподілених потоків даних, детально обгрунтування як позитивних так і негативних аспектів кожного запропонованого метода.

5. Негативні сторони роботи В ході рецензування виявлено неповноцінний опис у пунктах 2.3, 3.2.

6. Оцінка графічного оформлення та пояснювальної записки роботи Представлені матеріали роботи є правильно виконаними та логічно структуровані, що відображає послідовність виконання поставлених завдань. Проте матеріал мав декілька стилістичних та орфографічних помилок, які згодом були виправлені. Таким чином оформлення пояснювальної записки та графічного оформлення заслуговує оцінки «добре».

7. Відгук про роботу в цілому Зміст представленої роботи виглядає гармонічно, в повній мірі розкриває суть проблеми і підходи для вирішення поставленої проблеми за допомогою покращення існуючого методу. Результатом проведених досліджень стали відповідні висновки і конкретні пропозиції, щодо вдосконалення методу.

8. Інші зауваження _____

9. Оцінка дипломної роботи Робота заслуговує оцінки «добре», а її автор – присвоєння кваліфікації «магістра» з інженерії програмного забезпечення.

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи)
Партико Валерій Володимирович,
зав. каф. автоматизації та комп'ютерно-інтегрованих технологій
д.т.н., професор

"6" 12 2021 р. _____
(підпис)

Завідувачу кафедри інженерії програмного
забезпечення проф. Бедратюку Л. П.
здобувача вищої освіти
Дрозд Андрій Ігорович
факультет ІТ, 2 курс, група ІПЗм-20-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

23.11.2021 р.
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Метод розподілу обчислювальних ресурсів для обробки розподілених потоків даних»
Автор: Дрозд Андрій Ігорович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Медзатий Дмитро Миколайович, кандидат технічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посилакання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 8,46% і адресується до 176 джерел, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь дипломної роботи.

Керівник _____ Д. М. Медзатий

Гарант ОП _____ О. М. Яшина

Завідувач кафедри _____ Л. П. Бедратюк