Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерної інженерії та системного програмування

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Розподілені системи на основі стратегій призначення завдань для змінних робочих навантажень
Назва теми

КвРКІ.170157.17.03.14 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія»*
Назва

Виконав: студент IV курсу, група КІ–17–3 _____ О.С.Шевчук
Підпис Ініціали, прізвище

Керівник _____ А.О.Нічепорук
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної
Інженерії та системного
Програмування _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«24» червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Комп'ютерної інженерії та системного програмування

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123Комп'ютерна інженерія

Освітня програма освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 11" 012021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Шевчук Олегу Сергійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи)Розподілені системи на основі стратегій призначення завдань для змінних робочих навантажень

Керівник проекту (роботи)А.О.Нічепорук , д.т.н., доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2021 р. № 11

2. Строк подання студентом проекту (роботи) на кафедру 07.06.2021 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідженняпредметноїобласті та постановка задачі

Ефективні стратегії розподілу завдань для розподілених систем із сильно змінним навантаженням

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

Модель

TAPTF

Модель TAPTF–WCs 4 хостами

Процес Пуассона

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІСП		
Антиплагіат	Нічепорук А.О., доцент кафедри КІСП		

7. Дата видачі завдання «_11_»_____01_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	11.01.2021	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначенняоб'єкта та предмета дослідження	01.02.2021	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2021	виконано
4	Робота над розділом 2 – моделювання та проєктуваннямультипроцесорної системи	01.04.2021	виконано
5	Робота над розділом 3 – програмно - апаратнареалізаціямультипроцесорної системи	30.04.2021	виконано
6	Оформленняпояснювальної записки згідновимог	31.05.2021	виконано
7	ПопереднійзахистВКР	02.06.2021	виконано
8	ЗахистВКР на засіданні ЕК	Червень 2021 року	

Студент _____ О.С. Шевчук
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ А.О.Нічепорук
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 170157.17.03.14 ПЗ	Пояснювальна записка	72		
			Графічні матеріали			
2		КвРКІ 170157.17.03.14 Е8	Модель TAPTF	1		
3		КвРКІ 170157.17.03.14 Е8	Модель TAPTF-WC	1		
4		КвРКІ 170157.17.03.14 Е8	Процес Пуассона	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 170157.17.03.14 ВП		
Розробив		Шевчук			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Нічепорук				У	
Н. контр.		Лисенко				ХНУ, КІ-17-3	
Затв.		Говорущенко					

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Розподілені системи на основі стратегій призначення завдань для змінних робочих навантажень».

Автор роботи: Шевчук Олег Сергійович.

Керівник роботи: Нічепорук Андрій Олександрович.

Пояснювальна записка: 72 с., 6 рис., 9 табл., 4 дод., 43 джерела.

Графічна частина: 7 презентаційних слайдів.

ЕФЕКТИВНІ СТРАТЕГІЇ РОЗПОДІЛУ ЗАВДАНЬ ДЛЯ РОЗПОДІЛЕНИХ СИСТЕМ ІЗ СИЛЬНО ЗМІННИМ НАВАНТАЖЕННЯМ

Метою роботи є сторення нового підходу до призначення завдань у розподіленій системі, TAPTF.

У цій роботі було проаналізовано різні політики планування, що використовуються в обчислювальних системах, і їх вплив на загальні метрики масового обслуговування. Та створена нова політика призначення завдань для пакетних та супер-обчислювальних кластерів, яка називається "Task Assignment based on Prioritising Traffic Flows" (TAPTF)

Підпис студента Дата

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

FIFO – First-In-First-Out

LIFO – Last-In-First-Out

RSS – Random Selection for Service

SJF – Shortest Job First

PS – Processor Sharing

SRPT – reduced instruction set computer

PSJF – Pre-emptive Shortest-Job-First

LAS – Least-Attained-Service

TAPTF – Task Assignment based on Prioritying Traffic Flows

TAGS – Task Assignment based on Guessing Size

JSWQ – Join Shortest Weighted Queue

TAPTF-WC – Task Assign- ment based on Prioritising Traffic Flows with Work-Conserving Migration

TAGS-WC – Task Assignment based On Guessing Size - Work Conserving.

ВСТУП

Попит на обчислювальні ресурси в мережі постійно зростає для багатьох доменів програм. Користувачі пакетних та наукових обчислювальних кластерів ставлять все більший попит на ці ресурси, оскільки вони намагаються аналізувати та вирішувати більші та складніші проблеми в різних сферах, таких як інженерія, науки про життя та аерокосмічна промисловість. Такі проблеми можуть мати дуже мінливі вимоги щодо обчислювальних ресурсів, які їм потрібні. Що стосується так званих інтернет-додатків, користувачі роблять більший попит на веб-серверах та кластерах, які надають ці послуги, оскільки задовільнення більшої частини щоденних потреб виконується в режимі он-лайн. Ці послуги включають електронну комерцію, Інтернет-банкінг, соціальні мережі та придбання та обмін різними форматами мультимедійних файлів. Попит на ці послуги (і необхідні ресурси) часом може бути тимчасовим, часті випадки "спалахів", де попит різко зростає. Попит, який вони вимагають на обчислювальні ресурси, також дуже мінливий, що ставить унікальні вимоги до дизайнерів систем, які хочуть надати адекватні послуги якомога більшій кількості клієнтів.

Метою данної роботи є аналіз та покращення продуктивності розподілених систем при сильно змінних робочих навантаженнях. Основний акцент якої, на поліпшенні продуктивності розподілених систем шляхом створення політик призначення завдань, які відповідають потребам сучасних обчислювальних навантажень. Значна частина минулих досліджень у галузі призначення завдань (або планування) передбачає, що навантаження є менш мінливим, причому розподіл послуг, як правило, характеризується експоненціальним розподілом. Широкі недавні дослідження показали, що сучасні обчислювальні навантаження дуже різноманітні, і розподіли, що їх представляють, є «важкими». Такі розподіли характеризуються численними дрібними завданнями (або запитами) із невеликими вимогами до послуг, і небагато великих завдань із непропорційно великими вимогами до послуг. Ці характеристики роблять розумне розподіл завдань надзвичайно складним для ефективного використання ресурсів. У світлі цих висновків зосередження зусиль відбувається на аналізі, математичному

моделюванні та вдосконалення політики розподілу завдань за таких дуже змінних навантажень.

У данній роботі розглянуті деякі важливі існуючі політики розподілу завдань для розподілених систем. Проаналізовані їх сильні та слабкі сторони, особливо при вирішенні реалістичних, дуже змінних навантажень. У другому і третьому розділі буде створений і реалізований, при різних умовах, новий принцип розподілу завдань.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Політика планування

Дисципліна обслуговування (або планування) визначає порядок та порядок обслуговування завдань у черзі. Залежно від характеру та характеристик комп'ютерної системи, може застосовуватися лише попереднє планування (описане в розділі 1.1.1) підтримується. Це означає, що після того, як завдання виконується, воно не може бути перерване (тобто попередньо вимкнене) із сервісу, скоріше воно повинно виконуватися до завершення. В якості альтернативи може бути доступне попереджувальне планування (описане у Розділі 1.1.2), де завдання, яке обслуговується, може бути перерване вхідним або існуючим завданням у черзі. Залежно від характеру робочого навантаження, яке зазнає обчислювальна система, вибір політики планування може мати значний вплив на середні показники ефективності завдань, а також на дисперсію і, як наслідок, нашу впевненість у цих показниках . Врешті-решт, буде обрано політику планування відповідно до домену проблеми, апаратних та програмних обмежень (наприклад, операційної системи) відповідної обчислювальної системи.

1.1.1 Політика непередбачуваного планування

Попереджувальні політики планування диктують, як тільки завдання перебуває в службі, воно не може бути перерване будь-яким іншим надходить завданням або наявним завданням у черзі обслуговування. Це вигідно, оскільки воно все ще дозволяє маніпулювати порядком виконання завдань очікування, в той час як витрат, пов'язаних із виведенням та виведенням з експлуатації завдань (а також підтримкою та збереженням інформації про стан), уникається. З аналітичної точки зору, системи масового обслуговування без попереджувального планування зазвичай легше формулювати та оптимізувати.

First-In-First-Out (FIFO). Як випливає з назви, First-In-First-Out, сервісні завдання виконуються в тому порядку, в якому вони надходять. У літературі це часто згадується як "Хто прийшов, хто першим обслужив" (FCFS). Великий обсяг існуючих результатів масового обслуговування (наприклад, формула Поллачека-Хінчина) передбачає дисципліну планування FIFO. Інтуїтивно це вважається найбільш справедливою політикою планування, коли завдання винагороджується за прибуття раніше, ніж інше завдання, за рахунок обслуговування до нього.

Last-In-First-Out - (LIFO). За дисципліною "Останній-перший-вихід", останнє завдання, яке входить в чергу, є першим кандидатом на службу. В іншому випадку це згадується як "Прийшов-прийшов-першим обслужив" (LCFS) у літературі планування. Політики LIFO є аналогічними структурам даних стеку, де елементи додаються до стеку та видаляються (тобто вискакують) з верхньої частини стека, а не з нижньої. Можна було б уявити, що існують потенційні можливості, особливо при великих навантаженнях, для того, щоб завдання тривалий час томилися внизу стека. Справді, це виявляється часом - поки середні показники масового обслуговування для FIFO та LIFO еквівалентні, другий момент (і, відповідно, дисперсія) політики LIFO гірші, оскільки вони мають більшу залежність від навантаження системи .

Random Selection for Service - (RSS). Випадковий вибір вибирає завдання, що очікують на чергу, для обслуговування абсолютно випадковим чином. Тобто кожен клієнт у черзі має 1 / n шансів бути обраним для обслуговування. Результати черги для дисципліни планування RSS виявились однаковими з результатами планування FIFO . Всі дисципліни непередбачуваного планування, які не використовують розміри завдань (такі як FIFO, LIFO та RSS), мають однаковий розподіл за кількістю завдань у системі . Отже, (згідно з формулою Поллачека-Хінчина), очікувані (середні) показники черги цих політик еквівалентні.

Shortest Job First - (SJF). Shortest Job First, інакше його називають Shortest Job Next (SJN), вибирає завдання для обслуговування з усіх завдань, що очікують у черзі, що має найменшу вимогу до служби (наприклад, час виконання). Дисципліна планування SJF бажана, оскільки вона максимізує пропускну

здатність - тобто кількість завдань, які можна виконати до завершення за певний проміжок часу. Однак тривалий час існували думки, що політики SJF можуть спричинити голод служби для завдань із більшими вимогами до послуг, якщо постійно надходять менші завдання. Крім того, його додатки обмежені, оскільки для цього потрібні знання (або точні оцінки) вимог до послуг усіх завдань, що очікують у черзі.

1.1.2 Попередня політика планування

Політики попереджувального планування дозволяють завданням, що надходять (або завданням, що вже в черзі), попереджати або переривати завдання, яке зараз обслуговується, незалежно від того, завершено воно чи ні. Поточне завдання, яке обробляється, виводиться з експлуатації в обмін на інше. Такі методи часто можуть зменшити розбіжність у часі очікування завдань, залежно від конкретної застосовуваної політики попереджувального планування. Наприклад, завдання меншого розміру може перервати тривале завдання, тому воно не очікує непропорційно багато часу для обслуговування. Однак додаткові накладні витрати виникають за рахунок заміни незавершених завдань на роботу та поза ними та підтримання відповідної інформації про стан, щоб вони з часом могли відновитись.

Processor Sharing - (PS). Дисципліна планування спільного використання процесора (інакше відома як розподіл часу) надає квантовий сервіс для кожного завдання, яке чекає в черзі круглим способом. Черги спільного використання процесора мають ряд бажаних властивостей. Вони вважаються справедливими, оскільки всі завдання мають однакове очікуване уповільнення . Крім того, середній час, який витрачає завдання в системі, залежить лише від середнього розподілу розміру завдання, а не від інших вищих моментів. Інтуїтивно ці результати мають сенс, оскільки дисципліна обміну процесорами вирішує багато проблем, пов’язаних із чергою FCFS. Оскільки завданням надається фіксований зріз послуги, більші завдання не можуть блокувати менші завдання за ним у черзі протягом тривалих періодів часу, як це може відбуватися в політиках

непередбачуваного планування. Це також допомагає меншим завданням, не вимагаючи, щоб час обслуговування був відомий заздалегідь, оскільки для завершення обслуговування їм потрібна лише невелика кількість фрагментів.

Shortest Remaining Processing Time - (SRPT). Дисципліна планування найкоротшого часу обробки - це переважна політика, яка спочатку обслуговує завдання із найменшим часом, що залишився. Це вимагає, щоб вимога про послугу була відома заздалегідь для ефективного використання. Давно доведено, що дисципліна SRPT є оптимальною для мінімізації середнього часу відгуку в черзі . Незважаючи на це, його використання не є широко розповсюдженим через загальне сприйняття (і вчення), що воно віддає перевагу коротшим завданням, а довші завдання може спричинити збиток або навіть померти від голоду . Це сприйняття несправедливості політикою SRPT щодо більш масштабних завдань було оскаржене в останніх дослідженнях . Автори аналізують дисципліну планування SRPT, використовуючи стандартний аналіз черг M / G / 1, порівнюють результати з політикою планування PS, яка вважається справедливою, оскільки забезпечує однакове очікуване уповільнення для всіх завдань. Бансал та ін. стверджують, що уявлення про те, що планування SRPT є несправедливим, є в основному необгрунтованим, і демонструють, що при помірному навантаженні системи (незалежно від розподілу завдань за розміром) усі завдання віддають перевагу плануванню SRPT перед PS. При більшому навантаженні це спостереження все ще справедливо для розподілу розмірів завдань з важкими хвостами. У світлі цих висновків політика планування SRPT відтоді впроваджується на веб-сервері Просто змінивши порядок розміщення веб-запитів там, де це було заплановано, автори продемонстрували покращену затримку та середній час відповіді, маючи при цьому мінімальний негативний вплив на більші запити.

Pre-emptive Shortest-Job-First (PSJF). Політика попереднього планування найкоротших робочих місць, як і непередбачувальний еквівалент, має на меті виконати завдання з найкоротшою вимогою до послуги. Однак PSJF гарантує таку поведінку в будь-який час, дозволяючи завданням, які надходять, попереджати завдання, яке зараз обслуговується, якщо воно менше. PSJF забезпечує майже

оптимальний час відгуку на завдання в одній черзі, близький до часу політики SRPT, який, як відомо, є оптимальним, як описано раніше. Однак для певних класів завдань (наприклад, великих завдань) PSJF може їх дискримінувати та забезпечувати непередбачувані результати роботи в черзі. Це не дивно, оскільки PSJF завжди віддає перевагу меншим завданням, аніж більшим.

Pre-emptive Last-Come-First-Serve (PLCFS). Попереджувальна політика останнього прибуття - це просто переважний аналог описаного раніше LCFS. У будь-який час політика PLCFS забезпечує обслуговування останнього прибулого завдання. Будь-які запущені завдання за необхідності будуть виключені з обслуговування новим завданням. Як і обмін процесорами, PLCFS виявився справедливим і забезпечує передбачуваний час обслуговування всіх завдань, незалежно від їх розміру.

Least-Attained-Service (LAS). Дисципліна планування найменш досягнутої служби - це переважна політика, яка не вимагає знання вимог до обслуговування завдання. LAS працює, надаючи службі завдання в черзі, яка отримала найменше послуг загалом, порівняно з усіма іншими завданнями очікування. Завдання, що прибувають, завжди попереджають поточне завдання в службі і обробляються до прибуття наступного завдання або до отримання ним такої кількості послуг, що еквівалентна попередньо виконаному завданню, залежно від того, яке відбувається першим. Політика LAS є привабливою, оскільки було показано, що вона забезпечує середній час відгуку, порівнянний із часом, досягнутим SRPT, і при цьому не вимагає знання розмірів завдань, необхідних. У цьому ж дослідженні також було показано, що LAS забезпечує кінцевий середній час відгуку на більшість завдань (до 99-го процентиля) для завдань при перевантаженні $\rho = 2{,}0$. Подальша робота продемонструвала, що LAS (і варіанти LAS) є дуже ефективними, оскільки політики планування через вузькі місця в мережах з комутацією пакетів значно покращують планування FIFO за різних умов навантаження.

1.2 Індекс навантаження

Доступність та використання ресурсів, що надаються комп’ютером, зазвичай характеризуються у вигляді індексу навантаження. Індекс навантаження може містити показник використання одного ресурсу в комп'ютерній системі, наприклад, навантаження на центральний процесор. Крім того, це може бути комбінований показник безлічі ресурсів і показників, таких як завантаження процесора, використання пам’яті, підкачка диска та кількість запущених процесів. Ця інформація може бути використана для ідентифікації вузла, який завантажений злегка, що робить кандидатом можливість міграції завдань туди з вузла з великим навантаженням (обговорено в Розділі 1.3) або визначення цільового вузла для динамічної політики розподілу навантаження (обговорюється у розділі 1.4). У розділі 1.2.1 вказані типи мір та метрик, які зазвичай використовуються для побудови індексів навантаження в комп'ютерних системах. Розділ 1.2.2 описує деякі методи, що використовуються для ефективного та своєчасного розповсюдження цієї інформації серед розподілених систем.

1.2.1 Вимірювання індексу навантаження

Тип вимірювань навантаження та метрики, що використовуються в обчислювальних системах, дуже залежать від операційної системи та області застосування. Робота Феррарі та Чжоу намагається це формалізувати, враховуючи уявлення про те, що правильний індекс навантаження для даного домену програми (і суміш процесів, що складає навантаження) повинен гарантувати, що час відгуку на завдання має бути функцією індексу навантаження . У цьому дослідженні автори розглядають використання лінійної комбінації середньої довжини черги (наприклад, фонових процесів) та коефіцієнтів, що описують потреби в ресурсах для наступного завдання, яке очікує призначення, щоб представити сприймане навантаження на кожній машині. Метою обчислення індексу таким чином є забезпечення мінімізації часу відгуку для даної програми (з конкретними потребами в ресурсах). Врешті-решт це означає, що індекси

обчислюються для кожної програми за потреби, враховуючи різні вимоги до ресурсів різних додатків.

1.2.2 Інтерпретація та розповсюдження індексу навантаження

Мітценмахер розглядає, наскільки корисною є інформація про старе завантаження, вибираючи, куди направляти щойно надходять завдання в розподіленій системі . Враховуючи, що, як правило, неможливо мати миттєву глобальну інформацію про навантаження постійно доступною, Міценмахер досліджує використання інформації про навантаження, яка лише періодично оновлюється. Враховуючи, що навантаження може швидко змінюватися, часто незрозуміло, який є найбільш підходящим способом використання старої інформації про навантаження. Автор широко моделює та оцінює різні моделі для старої інформації про навантаження. Міценмахер розглядає теоретичний підхід дошки оголошень - де глобальна інформація централізовано розміщується на дошці оголошень. Однак насправді ця інформація може містити стару (і неточну) інформацію. Щоб вибрати, де призначається вхідне завдання, можна випадково вибрати d серверів (тобто підмножину), перевірити інформацію про їх завантаження на дошці оголошень і призначити завдання серверу з найменшим навантаженням. В якості альтернативи можна перевірити навантаження всіх серверів, і завдання, призначене серверу з найменшим навантаженням.

Надання інформації на дошці оголошень актуальною, підмножина підходу працює добре і має нижчі накладні витрати, ніж перевірка навантаження всіх серверів. У більш централізованих розподілених системах, де можлива централізована дошка оголошень з усією інформацією про завантаження, призначення завдань найменш завантаженому серверу може забезпечити кращі результати, і насправді є оптимальним з математичної точки зору у багатьох ситуаціях. Потрібно враховувати спосіб розповсюдження та інтерпретації навантаження в моделі дошки оголошень. Міценмахер пропонує модель періодичного оновлення, де дошка оголошень оновлюється новою інформацією про завантаження кожні T секунди. Таким чином, час оновлення буде 0, T, 2T,. . ..

Час між цими оновленнями вважається фазою, причому фаза і закінчується в момент iT. Для цього типу підходу існують дві ключові проблеми - кількість серверів, які враховуються (наприклад, усі сервери або підмножина) при виборі місця призначення завдань, і тривалість фази оновлення T. Якщо інтервал оновлення T залишається коротким, то вибір найкоротшої черги (відповідно до наявної в даний час інформації про завантаження) працює добре - оскільки в інтервалі надходить мало нових завдань, а стара інформація про завантаження залишається відносно точною. Зі збільшенням T може спостерігатися екземпляр поведінки стада, оскільки завдання присвоюються тій самій невеликій підмножині серверів, які здаються незначно завантаженими, що врешті-решт призводить до перевантаження. У таких випадках (у міру того як T збільшується і наближається до ∞), просто присвоєння випадкових завдань серверам зазвичай працює ефективніше. В якості альтернативи може бути використана модель безперервного оновлення, де дошка оголошень постійно оновлюється, але постійно відстає на T секунд від справжнього глобального стану. Це моделює загальний сценарій, коли існує затримка передачі між тим, коли інформація про навантаження розповсюджується, і коли вона доступна для використання завданнями. Як результат, завдання використовують інформацію, яка стара T секунд, коли роблять вибір за призначенням.

Було досліджено два сценарії - де інтервал безперервного оновлення T є фіксованою константою, і де T замінюється на X, випадковим експоненційно розподіленим значенням. У кожному випадку спостерігаються суттєво різні результати. У випадку фіксованого T, призначення завдань найменш завантаженому серверу працює погано, навіть при відносно невеликих значеннях T. Вибір у випадковому порядку серед невеликої підмножини найменш завантажених серверів пройшов значно кращі результати в широкому діапазоні значень T. Використання випадково розподіленого інтервалу оновлення X дало напрочуд гарні результати як для найкоротшої черги, так і для підмножин найкоротших призначень черги. Мітценмахер пояснює це тим, що завдання, що надходять у систему приблизно одночасно, матимуть різні погляди на систему і, отже, робитимуть різний вибір. Це дозволяє уникнути поведінки стада, яка

зазвичай зустрічається при інших підходах балансування навантаження, та покращує використання та продуктивність. Також коротко розглядається окрема модель оновлення, де сервери оновлюють інформацію про завантаження в різний, незалежно розподілений час. Однак було встановлено, що ця модель працює аналогічно стандартній моделі безперервного оновлення. З роботи Міценмахера випливає кілька цікавих результатів, але ми зазначаємо, що вона обмежена системами, де прибуття та розподіл послуг експоненційно розподіляються (тобто M / M / c). Як такі, ці висновки не можуть бути безпосередньо зіставлені з системами з дуже змінними надходженнями або розподілом послуг. Дійсно, висновки могли б суттєво відрізнятися через дуже мінливий характер важкозавантажених навантажень та їхню схильність розбалансувати навантаження в розподілених системах.

1.3 Міграція процесів

Процес - це конструкція операційної системи, яка представляє екземпляр запущеної комп’ютерної програми. З процесом зазвичай пов'язані інші ресурси, що характеризуються даними, стеком програм, вмістом реєстру та іншими дескрипторами стану, залежно від операційної системи, на якій він працює. Міграція процесу - це акт передачі процесу між двома логічними сутностями (джерелом і призначенням) під час виконання. Ці сутності можуть бути різними процесорами на одній машині або різними фізичними вузлами в мережі. Якщо міграція має зберегти роботу, інформація про стан також повинна передаватися під час міграції процесу. Часто використовується термін завдання як узагальнення концепції процесу. Механіка збереження роботи залежить від операційної системи та деталей реалізації, але її можна описати загалом наступним чином:

1) на основі певного критерію міграції запит на міграцію видається віддаленому вузлу. Після узгодження між вихідним та кінцевим вузлом запит на міграцію приймається;

2) процес призупинено до його виконання та відокремлюється від вихідного вузла. Зараз воно перебуває в мігруючому стані;

3) повідомлення, що надходять, тимчасово стоять у черзі, доки канали зв'язку не можуть бути перенаправлені на вузол призначення та успішно доставлені;

4) витягується стан процесу, який залежно від деталей реалізації може складатися з вмісту пам'яті та реєстру, стану зв'язку (наприклад, відкритих файлів) та контексту ядра;

5) процес відтворений на вузлі призначення в готовності до передачі відповідної інформації про стан;

6) інформація про стан передається і асоціюється з нещодавно відтвореним процесом на вузлі призначення;

7) необхідно створити посилання для переадресації, щоб направити зв'язок із нещодавно відтвореним процесом на вузлі призначення;

8) щойно відтворений процес відновлює виконання з моменту, коли він був зупинений на вихідному вузлі.

1.4 Класична політика розподілу навантаження

Проблема оптимального розподілу завдань у розподіленій системі була добре дослідженою протягом багатьох років. Більшість так званих `` класичних " підходів були створені за припущенням марківського процесу обслуговування (наприклад, експоненціально розподілений час обслуговування). Багато з цих політик все ще широко використовуються через їх спрощений характер та простоту реалізації. У цьому розділі досліджуються деякі з цих політик, оцінюємо їх сильні та слабкі сторони та визначаємо деякі відомі результати.

1.4.1 Random and Round Robin

Класичні політики призначення завдань, такі як Random і Round-Robin , традиційно використовуються в розподілених системах і досі широко використовуються для багатьох доменів додатків. Відповідно до політики «Випадкове» завдання з однаковою ймовірністю статично присвоюються кожному внутрішньому серверу. Використовуючи політику Round-Robin,

завдання призначаються серверам циклічно. Обидві політики вирівнюють очікувану кількість завдань на кожному сервері і часто використовуються в якості базової лінії для порівняння з іншими політиками розподілу завдань. Завдання призначаються без урахування навантаження кожного хоста або розподілу розмірів завдань. Незважаючи на це, Random та Round-Robin все ще часто використовуються у багатьох середовищах планування (швидше за все, завдяки простоті впровадження). Раніше було показано, що і Random, і Round-Robin мають схожі експлуатаційні характеристики. Зважені варіанти Random і Round-Robin є популярними політиками призначення завдань, зокрема, коли розподілена система містить неоднорідні хости. У цих випадках зважування призначення завдання, щоб більш потужні хости отримували більшу частку завдань, може призвести до значного поліпшення порівняно зі стандартними політиками Random та Round-Robin. Політики розподілу випадкових навантажень також поєднуються з динамічними політиками, де завдання призначається найменш завантаженому серверу, вибираючи лише з випадкової підмножини доступних серверів

1.4.2 Динамічні

Динамічні політики розумно призначають завдання на основі подання поточного навантаження на кожному хості. Підхід LLF (Least-Loaded-First) призначає серверу завдання із найменшим обсягом роботи, що намагається досягти миттєвого балансу навантаження. Залишилася робота може бути апроксимована довжиною черги (Найкоротша черга) , або припускаючи, що вимога до обслуговування завдань відома апріорі, сукупна робота, що залишається в черзі (Найменша робота-Залишилася). Підтримуючи баланс навантаження, час очікування в черзі може бути зменшений.

Відомо, що балансування навантаження мінімізує середній час відгуку у типі розподіленої системи, який ми розглядаємо в цій дисертації. Динамічна політика демонструє хорошу ефективність, оскільки розподіл обсягу завдань стає більш рівномірним, але коли воно наближається до емпірично виміряних робочих

навантажень (де $\alpha \approx 1$) та сильно варіюється, нові підходи, засновані на розмірах, працюють краще. Незважаючи на це, існує низка застережень. По-перше, „найкраща” продуктивність не завжди досягається балансуванням навантаження, особливо якщо вас цікавлять різні показники продуктивності, такі як середнє уповільнення. По-друге, балансування навантаження не завжди є практичним, оскільки часто ви залежать від приблизних показників навантаження, таких як довжина черги. При дуже мінливих робочих навантаженнях (де різниця між «малими» та «великими» завданнями може бути величезною) дуже ймовірно, що довжина черги може бути ненадійним показником фактичного навантаження на хості. По-третє, існує додаткова складність та накладні витрати на оновлення та розповсюдження інформації про навантаження. Таким чином, легко уявити, наскільки поганою політикою є залежати лише від кількості завдань у черзі на кожному сервері та впливу на продуктивність, який може виникнути в результаті використання такої інформації для вибору призначення завдань .

1.4.3 Центральна черга

Політика Central-Queue зберігає завдання в черзі до диспетчера, поки хост не перебуває в режимі очікування. Така політика виявилася еквівалентною політиці щодо найменшої роботи, що залишається, показуючи, що еквівалентну ефективність можна отримати без будь-якого попереднього знання про розмір завдання . Однак, хоча обидва демонструють настільки ж хороші показники за експоненціального навантаження, ефективність політики Центральної черги однаково низька за більш реалістичних умов навантажень з великими хвостами. Нещодавно було запропоновано два варіанти політики Центральної черги - Викрадення циклу з негайною відправленням (CS-ID) та Викрадення циклу з Центральною чергою (CS-CQ). CS-ID негайно відправляє завдання на внутрішній сервер, тоді як CS-CQ зберігає завдання в центральній черзі диспетчера, поки хост не перебуває в режимі очікування. Обидві політики оцінюються на основі виділеної політики (подібно до політики на основі розміру, викладеної в наступному розділі). У виділеній політиці один хост «присвячений»

обслуговуванню всіх коротких робіт, тоді як інший хост обслуговує довгі роботи. Як CS-ID, так і CS-CQ дотримуються подібної домовленості, але можуть викрадати цикли з простою хоста, якщо такі є (і розумно це робити). Наприклад, якщо надходить коротке завдання і короткий хост зайнятий, поки довгий хост не працює, коротке завдання може бути відправлене на довгий хост для поліпшення використання. Як CS-ID, так і CS-CQ демонструють покращення щодо виділеної політики у багатьох областях (особливо для коротких завдань). Застосування цих політик обмежується доменами, де відоме апріорне знання розміру завдань, а у випадку з CS-CQ, між диспетчером та серверними серверами повинен бути постійний зворотний зв'язок, щоб повідомити диспетчера про холостий хост.

1.4.4 Обмеження

Політики призначення завдань, перераховані вище, не підходять для розподілу завдань із великою мінливістю. Очевидним є те, що всі показники залежать від другого моменту розподілу потреби у послузі. Що стосується динамічної політики (наприклад, „Найменша робота, що залишається)", всі показники залежатимуть від квадратичного коефіцієнта варіації . Такі розподіли, як Ерланг, вважаються малими дисперсіями, тоді як розподіли, такі як Парето та Гіперекспоненціальний, вважаються дуже мінливими.

1.5 Ефективність при важких робочих навантаженнях

Розглядаючи продуктивність розподіленого кластера серверів, де коефіцієнт прибуття - Пуассон, а розподіл часу обслуговування - за обмеженим розподілом Парето. Встановлено p (найбільше завдання) рівним 107, і k (найменше завдання), щоб зберегти середнє розподіл E (X), встановлене на рівні 3000. Ці параметри використовуються для того, щоб зосередити увагу на ефекті зміни дисперсії у розподілі часу обслуговування. Політика Random просто виконує випадкове розділення потоку прибуття і не приділяє уваги навантаженню кожного заднього кінцевого вузла при призначенні завдань. Навантаження системи змінюється,

показуючи продуктивність, де ρ = 0,3 (низьке навантаження), ρ = 0,5 (помірне навантаження) і ρ = 0,7 (велике навантаження). Зі збільшенням варіації (коли α зменшується) очікуваний час очікування та уповільнення швидко зростають. Крім того, масштаб вдосконалення, який показує динамічна політика порівняно із випадковою політикою, дещо зменшується із збільшенням навантаження системи.

Зі збільшенням навантаження системи є менша ймовірність простою хоста, навіть за умови довільної політики. Як такий динамічна політика має менше можливостей для вдосконалення. Спостерігається значне поліпшення показаної політики призначення завдань на основі розміру (TAGS), особливо в умовах великих і екстремальних варіацій розміру завдання. Такі політики за своєю природою зменшують різницю розмірів завдань на кожному хості, розподіляючи робоче навантаження між кожним хостом. Це призводить до групування завдань однакового розміру разом за чергами кожного хоста, відповідно зменшуючи дисперсію на кожному хості та покращуючи такі показники продуктивності, як середній час очікування та уповільнення.

1.6 Постановка задачі

Ураховуючи все вище сказане, очевидно, що існуючі політики слабо адаптовані до змінних умов. Метою данної роботи – є створення нового підхіду до розподілу навантаження, який називається TAPTF. Робота виконуватиметься у три етапи:

1) пошук інформації та аналіз наявних рішень;
2) проектування системи; розробка нового підхіду до розподілу навантаження;
3) реалізація розподілу навантаження з урахованням міграції.

1.7 Висновки

У цьому розділі було досліджено різні політики планування, що використовуються в обчислювальних системах, і їх вплив на загальні метрики

масового обслуговування. Також розглянулись різні методи, що використовуються для вимірювання та розповсюдження інформації про навантаження в комп'ютерній системі, роль міграції процесів у розподілених системах, визначаючи, як механізм працює в сучасних розподілених системах, і за яких обставин доцільно використовувати. У розділі 1.4 розглядаються класичні політики розподілу завдань, включаючи статичні політики, такі як «Random and Round Robin», а також Динамічні та Політики центральної черги.

2 ПРИЗНАЧЕННЯ ЗАВДАННЯ НА ОСНОВІ ПРІОРИТЕТНОСТІ ПОТОКУ РУХУ

2.1 Запропонована модель – TAPTF

У цьому розділі було запропоновано нову політику призначення завдань під назвою TAPTF - Task Assignment based on Prioritying Traffic Flows - для вирішення обмежень існуючих підходів у роботі з певними класами трафіку. Детально викладено мотивацію підходу TAPTF. Виділено відмінності між ним та існуючими моделями та пояснено важливі особливості, методи, що використовуються TAPTF для підвищення ефективності. Данний розділ надає концептуальне уявлення про запропоновану модель. Важливі параметри, пов'язані з даною моделлю, визначення та обчислення.

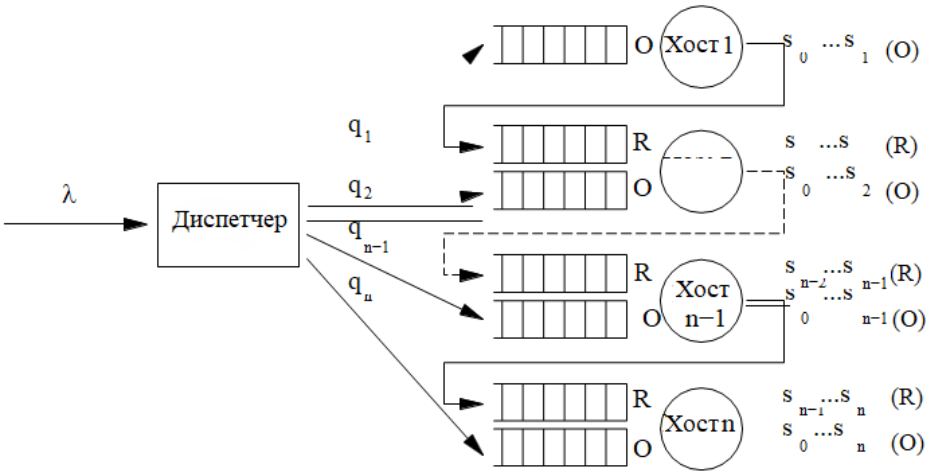


Рисунок 2.1 - Ілюстрація моделі TAPTF

2.1.1 Мотивація

Підхід TARS, хоч і здавався протилежно інтуїтивним у багатьох відношеннях, виявився дуже ефективною політикою призначення завдань для розподілених систем. Таким чином, TAGS забезпечує чудову точку порівняння для будь-якої нової політики призначення завдань, що працює за подібних

обмежень. Політика TAGS має ряд бажаних властивостей - найважливішим є те, що вона не передбачає жодних попередніх знань щодо вимог до обслуговування вхідних завдань, зберігаючи при цьому хорошу продуктивність.

Політика TAGS чудово працює в реалістичних умовах, що змінюються, використовуючи важкохвосту природу, що відповідає багатьом обчислювальним робочим навантаженням. Незважаючи на це, TAGS може спричинити значний надлишок у фонових хостах - марно оброблена обробка, яку завдає завдання (і відповідне навантаження, розміщене на хості), коли воно було розміщено в неправильній черзі і згодом перезапущено після перевищення обмеження обробки, пов'язаного з хостом. Неправильно призначене завдання карається шляхом зупинки, розміщення його в кінці черги чергового хоста та перезапуску з нуля (після досягнення передньої частини цієї черги). Ці недоліки виправдані тим, що за самою природою розподілу важкого робочого навантаження завдання, які караються, можуть поглинути додатковий час очікування та обробки для загального блага. Тим не менше, це марнотратно, але як можна підвищити ефективність, зберігаючи при цьому хороші показники? Треба розглянути три ключові області:

1) зменшення дисперсії завдань, що мають однакову чергу;

2) зменшення штрафу за даремно оброблену (надлишкову) обробку даних на фонових хостах - викликанезавдання, які не завершують свою обробку вчасно і перезапускаються на іншому хості («передача даних»);

3) зменшення покарання за перезапущені завдання (враховуючи, що завдання потенційно може бути перезапущено кілька разів).

Для вирішення цих ключових питань була сформульована політика TAPTF. З рисунка 2.1 можна побачити введення подвійних черг на кожному хості - звичайної (O) черги та черги перезапуску (R). Завдання можуть увійти в систему на будь-якому хості. Наплив завдань, призначених кожному хосту, контролюється дробом qi. Ці доповнення дозволяють моделі TAPTF бути гнучкими, забезпечуючи засоби для маніпулювання поведінкою моделі TAPTF в залежності від характеристик робочого навантаження.

2.1.2 Прийоми

У Розділі 2.1.1 було виявлено ряд недоліків моделі TAGS, які потрібно було усунути. Таким чином, TAPTF був розроблений з метою вдосконалення цих ключових областей. Обґрунтування методів, які TAPTF використовує для усунення недоліків існуючих підходів, викладено наступним чином.

Зменшення дисперсії. Бажано зменшити ефект варіації розміру завдання, що суттєво згубно впливає на ефективність у міру збільшення мінливості. TAPTF зменшує різницю в розмірах завдань, що мають однакову чергу, за рахунок використання подвійних черг (звичайної (O) черги та черги перезапуску (R)) та міграції, намагаючись згрупувати завдання подібного розміру разом. Це робиться для того, щоб мінімізувати шанс короткого завдання застрягти за довгим завданням в одній черзі.

Зменшення кількісті передач. Надлишок - додаткову роботу, створену перезапуском багатьох завдань з нуля - потрібно звести до мінімуму. TAPTF намагається зменшити кількість "передач", розмістивши якомога більше завдань у найбільш підходящій черзі (тобто їх кінцевому пункті призначення) насамперед - зменшення штрафу як за хости, так і за завдання. Це досягається двома взаємопов’язаними способами. По-перше, маніпулюючи часткою завдань (qi), що надсилається кожному хосту, що надає подальший ефект збільшення кількості завдань, які правильно призначені для відповідного хоста - тобто де вони можуть виконуватися до завершення. По-друге, причина того, що він може увійти в систему (і, можливо, завершити) на будь-якому хості, полягає у тому, що гранична межа нижньої межі кожної звичайної (O) черги становить k, найменший можливий розмір завдання. У системі TAGS завдання, яке потрібно обробити на хості i (наприклад, його розмір знаходиться між si − 1 і si), має перенести з хосту 1 на хост i. У TAPTF для того самого завдання існує ймовірність qi, що вона буде направлена туди (ідеальний вибір), і ймовірність qi + qi + 1 + ... + qn, що вона буде призначена хосту i або вище - де воно не буде піддаватися жодним передачам. Це стає більш важливим, оскільки варіація розміру завдання зменшується.

Зменшення ефекту передачі. Поки кількість передач скорочується, вони все одно будуть відбуватися. Знаючи це, можливе зведення до мінімуму згубний вплив передач, які трапляються (зокрема, на час очікування та уповільнення), переносячи перезапущені завдання через черги перезапуску до кінцевого пункту призначення. Залежно від того, чи метою є оптимізація загальних показників ефективності або справедливість до покараних завдань , завдання можна було швидко відстежити до кінцевого пункту призначення, надавши їм пріоритет обслуговування на кожному хості (над завданнями в звичайній черзі, отриманими від диспетчера). Зверніть увагу, що поведінка за замовчуванням є протилежною, коли завдання в звичайній черзі мають пріоритет служби над завданнями в черзі перезапуску.

Кожен хост у нашій розподіленій системі є чергою M / G / 1 FCFS. Видно, що всі показники ефективності залежать від E (X2), другого моменту розподілу розміру завдання. E (X2) пропорційний дисперсії розміру завдань, що мають спільну чергу. Навіть після того, як буде узагальнено формулу Поллачека-Хінчина на черги пріоритетів (вимагаються реалізацією подвійної черги). Було зроблено висновок, що зменшення розбіжностей у сервісних вимогах завдань на кожному хості може поліпшити продуктивність, зменшуючи ймовірність того, що менше завдання застряє за значно довшим завданням.

2.1.3 Концептуальний вигляд моделі TAPTF

Як видно на малюнку 2.1, завдання надходять до центрального диспетчера, слідуючи процесу Пуассона зі швидкістю λ. Диспетчер призначає завдання (в порядку "Перший-у-першому") одному з n хостів (скажімо, Host i, де $1 \leq i \leq n$) навмання з імовірністю qi. З метою аналізу було зауважено, що потік надходження до хоста і також є процесом Пуассона зі швидкістю λqi.

Завдяки важким характеристикам розподілу завдань за розмірами було припущено, що розподіл розмірів завдань (тобто розподіл послуг) слідує за обмеженим розподілом Парето B (k, p, α). Кожному хосту в розподіленій системі

присвоюється "відсікання" (si). Зокрема, завдання обробляються на хостах із наступними умовами:

1) черга O хосту i має справу лише із завданнями, розміри яких знаходяться в діапазоні [k, si], $1 \leq i \leq n$;

2) черга R хосту i має справу лише із завданнями, розміри яких знаходяться в діапазоні [si − 1, si], $1 < i \leq n$, де $k = s0 < s1 < s2 < s3 < \ldots < sn = p$. Ці граничні значення можна обчислити для мінімізації певних вимірюваних величин, таких як середній час очікування або середній час уповільнення.

Кожен хост (крім хосту 1) забезпечує дві черги, звичайну чергу та чергу перезапуску (позначаються O та R відповідно). Всі завдання в чергах O та R подаються за принципом " First- Come-First-Served " (FCFS). Завдання, надіслані певному хосту від диспетчера, приєднуються до черги O цього хосту. Після того, як завдання перемістилося в передню чергу, воно може почати оброблятися. Якщо час обробки завдання на даному хості перевищує призначений ліміт відсікання, завдання зупиняється та переміщується до черги перезапуску (R), що належить наступному хосту. Цей процес повторюється до тих пір, поки ці завдання не завершаться у кінцевому (правильному) місці призначення. Завдання, що очікують в черзі O, мають пріоритет обслуговування над тими, що знаходяться в черзі R на даному хості. Однак завдання, яке обслуговується з черги R, не буде попередньо вивільнене з обслуговування при надходження завдання в чергу O на даному хості. Це поведінка політики TAPTF за замовчуванням.

Одним із способів, що модель TAPTF відрізняється від TAGS, є фіксовані межі нижнього розміру на кожному хості ($k = s0$), так що всі завдання з розмірами, меншими або рівними фіксованій точці відсікання, можуть бути потенційно оброблені на конкретному хості. Це означає, що завдання можна відправити на будь-який хост , спочатку без першого відправлення на хост 1 (згідно з підходом TAGS), зберігаючи властивість, що попит на завдання не відомий апріорі. Крім того, TAPTF використовує подвійні черги на кожному хості, щоб пришвидшити потік коротших завдань, дозволяючи швидко обробляти менші завдання в звичайній черзі та мігруючи більші завдання з місця, дозволяючи їм об'єднуватися в черги перезапуску у наступних хостів.

2.1.4 Математичні попередні дані для моделі TAPTF

У цьому розділі буде визначено та обчислено всі важливі параметри, пов'язані з моделлю TAPTF. Основна мета - використовувати їх для визначення оптимальних точок відсікання.

Таблиця 2.1 - Позначення моделі TAPTF

n	Кількість хостів у системі
$B(k, p, a)$	Обмежений розподіл розміру завдання Парето
k	Нижня межа розподілу розміру завдання
p	Верхня межа розподілу розміру завдання
$f(x)$	Функція щільності ймовірності для B (k, p, α)
a	Параметр
s_i	Обрізання розміру завдання для хосту i
q_i	Частка завдань, відправлених хосту i
λ	Коефіцієнт надходження завдань до системи
ρ	Навантаження системи
p_i	Частка завдань, кінцевим пунктом призначення є або хост i, або його попередники
h_{iO}	Частка завдань, які відвідують звичайну (O) чергу хосту i
h_{iR}	Частка завдань, які відвідують чергу перезапуску (R) хосту i
h^{0}_{iO}	Частка завдань, кінцевим пунктом призначення яких є хост i звичайна (O) черга
h^{0}_{iR}	Частка завдань, кінцевим пунктом призначення яких є хост i черга перезапуску (R)
$E(X^{j}_{iO})$	j-й момент розподілу завдань, остаточний пункт призначення - звичайна (O) черга хосту i
$E(X^{j}_{iR})$	j-й момент розподілу завдань, остаточний пункт призначення - черга перезапуску (R) хосту i

Кінець таблиці 2.1 - Позначення моделі TAPTF

$E(hostX_{iO}^{j})$	j-й момент розподілу завдань, які витратив час у звичайній (O) черзі хосту i
$E(hostX_{iR}^{j})$	j-й момент розподілу завдань, які витратив час у черзі перезапуску (R) хосту i
λ_{iO}	Швидкість прибуття до звичайної черги (O) хосту i
λ_{iR}	Швидкість прибуття в чергу перезапуску (R) хосту i
ρ_{iO}	Завантаження в звичайну (O) чергу хоста i
ρ_{iR}	Завантаження в черзі перезапуску (R) хоста i
$E(hostW_{iO})$	Час очікування завдання в звичайній (O) черзі хоста i
$E(hostW_{iR})$	Час очікування завдання в черзі перезапуску (R) хоста i
$E(W_{iO})$	Час очікування завдання, кінцевим пунктом призначення є звичайна (O) черга хоста i
$E(W_{iR})$	Час очікування завдання, кінцевим пунктом призначення якого є черга перезапуску (R) хосту i
$E(S_{iO})$	Очікуване уповільнення в звичайній (O) черзі хоста i
$E(S_{iR})$	Очікуване уповільнення в черзі перезапуску (R) хоста i

(де k = s0 <s1 <s2 <... <sn − 1 <sn = p), що відповідає мінімальному середньому часу очікування або уповільнення для завдань, що надходять у розподілену систему. Варто зазначити, що наведені нижче результати зводиться до результатів, отриманих для політики TAGS, коли q1 = 1 і немає звичайних черг. Позначення моделі TAPTF наведено в таблиці 4.1.

У моделі TAPTF можливо обрати пріоритетність завдань, що знаходяться в черзі перезапуску (зокрема, "передачі даних", що надходять від хоста над нею), над завданнями в звичайній черзі. Як варіант, є можливість обрати, щоб завдання в звичайній черзі (які отримуються безпосередньо від диспетчера) мали пріоритет послуги перед завданнями в черзі перезапуску (що фактично стає чергою з

низьким пріоритетом). Останній є поведінкою моделі TAPTF за замовчуванням, оскільки забезпечує найкращу продуктивність.

Наступний набір результатів, що стосується очікуваного часу очікування завдань у звичайному хосту I (O) та перезапуск (R) черг, спирається на деякі ключові факти:

1) завдання в черзі O мають пріоритет обслуговування над завданнями в черзі R;

2) завдання, яке обслуговується в черзі R, не буде попереджене від служби завданням, яке згодом надходить у чергу O;

3) у межах кожної черги завдання обробляються на основі FCFS;

4) необхідно припустити, що завдання, які надходять у черги R, утворюють процеси Пуассона.

Як раніше обговорювалося, потоки надходження в черги O - це процеси Пуассона. Це не так у випадках, коли надходять у черги R, які (як зазначає Гарчоль-Балтер) є менш розривчастими (тобто більш рівномірно випадковими), ніж у процесі Пуассона. Якщо розміри завдань розподіляються експоненціально, тоді результат черги буде Пуассоном, але не інакше.

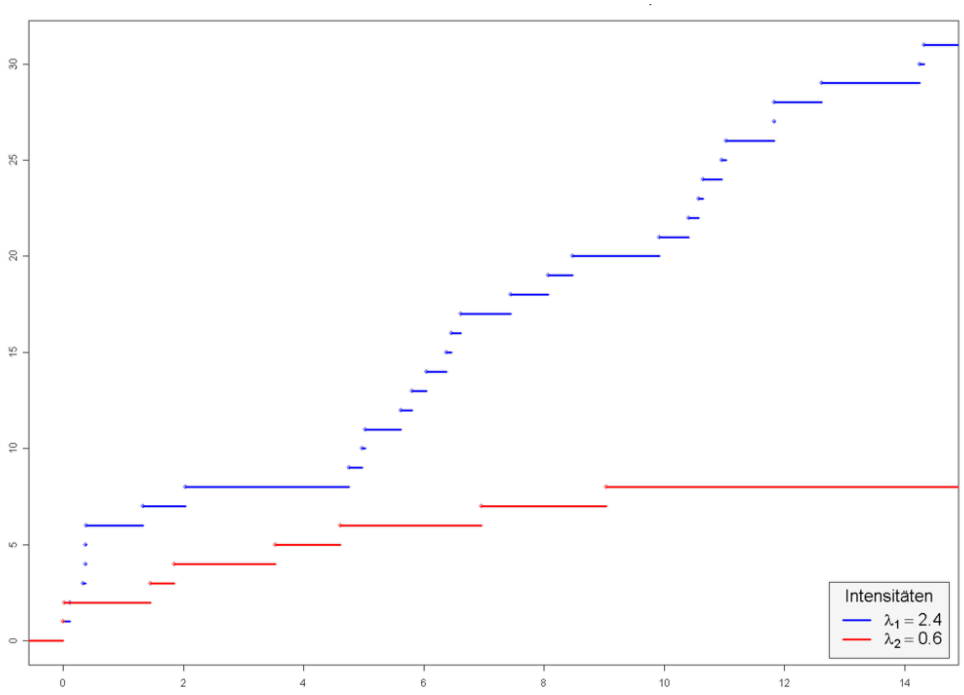


Рисунок 2.2 - Процес Пуассона

2.1.5 Вибір границь

Як і більшість політик, що базуються на розмірах, ефективність TAPTF критично залежить від вибору використовуваних границь. Обмеження стосуються діапазону розмірів, пов'язаного з кожним хостом. Обмеження можна вибрати для оптимізації середнього часу очікування або середнього уповільнення. Для того, щоб оптимізувати середній час очікування, навантаження має бути більш рівномірно збалансовано між хостом. Для оптимізації середнього уповільнення застосовуються методи розбалансування навантаження, особливо в умовах великих варіацій розміру завдання. Було вирішено оптимізувати як середній час очікування, так і, що ще важливіше, середнє уповільнення, оскільки бажано, щоб затримка завдань була пропорційною вимозі до обслуговування.

Межі для TAPTF вибираються так само, як і підхід TAGS. Граничні значення для TAPTF та TAGS є функцією розподілу розміру завдання (у даному випадку визначається обмеженим Парето B (k, p, α)) та швидкості надходження завдання у розподілену систему, λ. Ці параметри можна визначити, спостерігаючи розподілену систему протягом певного періоду. Оптимальний середній час очікування та середнє уповільнення для випадку двох та трьох хостів можна отримати для TAGS, враховуючи вище зазначені параметри, вирішивши оптимальні значення граничних границь за допомогою математики . Для чотирьох і більше хостів відсікання потрібно налаштовувати вручну, але таких самих результатів можна досягти, дотримуючись деяких простих принципових правил.

Тепер проблема оптимізації визначена, є можливість обрати оптимізацію середнього значення час очікування (описаний проблемою I), середнє уповільнення (описане у проблемі II) або їх поєднання.

Як описано вище, вибір обмежень залежить від мінливості розміру завдання. Чим нижче параметр α, тим вища мінливість і менший відсоток завдань, що становить 50% навантаження. TAGS (і згодом TAPTF, які можуть поводитися як TAGS, встановивши q1 = 1,0, коли є розумним), можуть використовувати це

властивість важкохвостого розподілу шляхом запуску всіх (або переважної більшості) (малих) завдання на першому хості, залишаючи їх під легким і помірним навантаженням, тоді як найбільші завдання фільтруються, щоб з часом їх обробляти останні хости.

Оскільки мінливість зменшується (α збільшується), більше немає можливості так легко використовувати важкохвосту нерухомість. Середній розмір завдань, які вважаються «малими», повільно збільшується із збільшенням α. Таким чином, потрібно відповідно обирати обмеження, а також маніпулювати частиною завдань, призначених останнім хостам. Використовуючи важкохвосту власність, обробляючи більші завдання на останніх хостах, але не потрібно розбалансовувати навантаження до наскільки була можливість, коли мінливість була вищою ($\alpha \leq 1$). Коли α наближається до 2.0, розмір завдання варіація нижча, і інші хости повинні почати тягнути свою вагу, щоб підтримувати хороший середній час очікування та уповільнення. TAPTF використовує ці знання для забезпечення кращих показників у цих сферах.

2.2 Аналітичне порівняння

Для того, щоб оцінити корисність підходу TAPTF, було проведено аналітичне порівняння з TAGS та Random. Випадковий рівень включений як базовий рівень, тоді як TAGS забезпечує найкращу точку порівняння, оскільки він працює за схожими обмеженнями до TAPTF, де не передбачається апріорне знання вимог щодо обслуговування завдання. Ці підходи оцінювались у різних умовах та їх ефективність порівнювалась з використанням найважливішої з метрик. Було враховано діапазон значень α, від 0,5 до 2,0, що демонструє широкий діапазон варіацій розміру завдання, від екстремальних змін розміру завдання ($\alpha \approx 0{,}5$) до низьких варіацій розміру завдання ($\alpha \approx 2{,}0$), і все між ними. Щоб спостерігати ефект зміни дисперсії, середнє значення обмеженого розподілу Парето встановлюється на рівні 3000, а максимальне значення p встановлюється на рівні 107. Для того, щоб зберегти середнє значення фіксованим, мінімальне значення k змінюється при зміні параметра α.

Кожне значення α оцінювалось для різних системних навантажень (ρ) - 0,3 (низьке навантаження), 0,5 (помірне навантаження) та 0,7 (велике навантаження). Ці порівняння були проведені для двох та трьох хост-систем,

Таблиця 2.2 - Розподіл завдань у TAPTF - 2 хоста, ρ = 0,3

α	q_1	q_2
1.0	0.98	0.02
1.1	0.97	0.03
1.2	0.96	0.04
1.3	0.95	0.05
1.4	0.93	0.07
1.5	0.90	0.10
1.6	0.87	0.13
1.7	0.84	0.16
1.8	0.80	0.20
1.9	0.76	0.24
2.0	0.73	0.27

(a) E(W)

α	q_1	q_2
1.0	1.00	0.00
1.1	0.99	0.01
1.2	0.99	0.01
1.3	0.98	0.02
1.4	0.96	0.04
1.5	0.94	0.06
1.6	0.91	0.09
1.7	0.88	0.12
1.8	0.84	0.16
1.9	0.80	0.20
2.0	0.76	0.24

(a) E(S)

після чого вже не було змоги знайти оптимальні значення si із доступними обчислювальними ресурсами. Це не є великою проблемою саме по собі, як зазначалося в попередніх дослідженнях, оскільки розподілена система n-хоста (де n> 2) із системним навантаженням ρ завжди може бути влаштована таким чином, щоб забезпечити продуктивність, яка є порівнянним або навіть кращим, ніж найкраща продуктивність двох чи трьох хост-систем (де n кратно двом або трьом відповідно) із завантаженням системи ρ. Наприклад, система з 4 хостами (з двома підсистемами, що містять по 2 хости в кожному) буде поводитися ідентично стандартній системі з 2 хостами. Тобто робочі характеристики однієї підсистеми в цьому сценарії будуть такими ж, як і вся система 2 хоста. Те саме стосується системи 6 хостів (з двома підсистемами, що містять 3 хости) та стандартної системи 3 хостів. Це справедливо для будь-якої політики призначення завдань.

Аналітичне порівняння було проведено в Mathematica 5.0, використовуючи математичні попередні етапи. Узагальнена математична модель TAPTF також використовується для моделювання поведінки TAGS шляхом встановлення q1 = 1,0 (і згодом q2 ... qn дорівнює 0) - заперечення подвійних черг та множинних точок входу і змушуючи його поводитися ідентично тегам. Для кожного сценарію визначаються оптимальні граничні значення щодо середнього часу очікування та середнього сповільнення як для TAPTF, так і для TAGS, використовуючи функцію NMinimize в Mathematica для отримання найкращого (і найбільш справедливого) порівняння. NMinimize здійснює чисельний пошук значень si у кожному екземплярі, що створюють локальні мінімуми для очікуваного часу очікування, E (W) та очікуваного середнього уповільнення, E (S).

Вибір параметрів qi має значний вплив на ефективність політики TAPTF. У випадку з 2 хостами, є можливість чисельно шукати комбінації значень si та qi, які дають найкращі результати (тобто локальні мінімуми) для очікуваного часу очікування та уповільнення. У випадку з 3 хостами, потрібно налаштувати параметри qi вручну. Це не так проблематично, як здається, оскільки ми можемо використовувати свою інтуїцію щодо необхідного розподілу завдань, а також майже оптимальних результатів, отриманих у 2 сценаріях хосту, для керівництва вибором.

Політики призначення завдань, які передбачають апріорне знання розмірів завдань, не оцінюються в цьому розділі, оскільки спонукає більш песимістичний (і менш обмежувальний) погляд на розподілений (кластер) модель, де ця інформація не є гарантовано доступною. Політики щодо найменшої роботи та центральної черги (які, як виявилося, еквівалентні) опущені з двох основних причин. По-перше, ці політики не відповідають припущенням проблемної області, обговореним у Розділі 2.1.1. По-друге, хоча багато хто вважає цю політику підходящою для умов незначних та помірних коливань, попередня робота показала лише помірне збільшення результативності порівняно з випадковою за аналогічною оцінкою, що проводилась у цій главі. Це ж дослідження показало, що алгоритм TAGS перевершує політику щодо найменшої роботи, що залишається, майже у всіх розглянутих сценаріях (як з малими, так і з великими варіаціями). Таким чином, доцільно зосередити увагу на TAGS.

В інтересах чітких та значущих результатів, порівняння середнього часу очікування та середнього уповільнення виконується з використанням відповідних політик TAPTF та TAGS, оптимізованих для цього показника. Випадкова політика включена в якості базової лінії для порівняльних цілей у кожному випадку.

2.2.1 Два хости

У цьому розділі представлено аналітичне порівняння TAGS та TAPTF у системі, що розподілена з двома хостами. Результати показують показники продуктивності для завантаження системи 0,3. Результати для TAPTF показуються лише там, де вони кращі за TAGS, оскільки TAPTF може зменшити до TAGS (і досягти однакових показників), як описано в Розділі 2.1 та Розділі 2.2.

З даного аналізу політика TAGS досягає кращого середнього часу очікування та уповільнення в умовах від екстремальних до великих коливань (де α становить від 0,5 до 1,0). Сфери, де політика TAPTF вдосконалюється щодо TAGS, виділені на графіках. Можна помітити, що в умовах помірних та низьких коливань (де α становить від 1,1 до 2,0) політика TAPTF досягає кращих показників щодо середнього часу очікування та уповільнення.

Таблиця 2.3 - Розподіл завдань у TAPTF - 2 хоста, ρ = 0,7

a	q_1	q_2
0.9	1.00	0.00
1.0	0.99	0.01
1.1	0.98	0.02
1.2	0.95	0.05
1.3	0.91	0.09
1.4	0.86	0.14
1.5	0.81	0.19
1.6	0.77	0.23
1.7	0.73	0.27
1.8	0.70	0.30
1.9	0.67	0.33
2.0	0.64	0.36

(а) E(W)

a	q_1	q_2
0.9	0.97	0.03
1.0	0.95	0.05
1.1	0.92	0.08
1.2	0.88	0.12
1.3	0.84	0.16
1.4	0.79	0.21
1.5	0.76	0.24
1.6	0.72	0.28
1.7	0.69	0.31
1.8	0.67	0.33
1.9	0.64	0.36
2.0	0.62	0.38

(б) E(S)

Це збільшення продуктивності можна пояснити використанням подвійних черг та призначенням завдань усім серверам (або їх підмножині), а не подачею всіх завдань на перший хост, згідно з підходом TAGS. З таблиці 2.3 помітно, що коли варіація збільшується (і α зменшується), TAPTF наближається до поведінки, подібної до TAGS, для оптимальної роботи. Можливо побачити, коли оптимізовано час очікування (де α = 1,0), майже всі завдання (98%)

відправляються на хост 1. У міру подальшого збільшення варіацій (де α становить від 0,5 до 1,2) поведінка, схожа на TAGS, дає найкращі результати. І навпаки, коли варіація зменшується, доцільно призначати деякі завдання другому хосту. Оскільки варіація зменшується (і α наближається до 2.0), є змога дозволити собі призначити більше завдань другому хосту.

Політика TAPTF покращує TAGS у більшому діапазоні сценаріїв варіації завдань, ніж це було при низькому навантаженні (при цьому TAPTF демонструє нижчий середній час очікування та уповільнення, коли α становить від 0,9 до 2,0). Можна помітити, що TAGS суттєво страждає від високого навантаження на систему.

Можна спостерігати збільшену частку завдань, відправлених другому хосту для того, щоб підтримувати вищу ефективність, ніж політика TAGS.

Таблиця 2.4 - Розподіл завдань у TAPTF - 3 Хости

a	q_1	q_2	q_3
0.9	1.0	0.0	0.0
1.0	1.0	0.0	0.0
1.1	0.9	0.1	0.0
1.2	0.8	0.2	0.0
1.3	0.75	0.25	0.0
1.4	0.7	0.3	0.0
1.5	0.6	0.4	0.0
1.6	0.6	0.4	0.0
1.7	0.6	0.3	0.1
1.8	0.6	0.3	0.1
1.9	0.5	0.4	0.1
2.0	0.5	0.4	0.1

(а) ρ = 0.3

a	q_1	q_2	q_3
0.9	0.95	0.05	0.0
1.0	0.9	0.1	0.0
1.1	0.8	0.2	0.0
1.2	0.75	0.25	0.0
1.3	0.7	0.3	0.0
1.4	0.7	0.3	0.0
1.5	0.6	0.4	0.0
1.6	0.6	0.3	0.1
1.7	0.5	0.4	0.1
1.8	0.5	0.4	0.1
1.9	0.5	0.3	0.2
2.0	0.5	0.3	0.2

(б) ρ = 0.5

a	q_1	q_2	q_3
0.9	0.95	0.05	0.0
1.0	0.9	0.1	0.0
1.1	0.8	0.2	0.0
1.2	0.8	0.2	0.0
1.3	0.7	0.3	0.0
1.4	0.6	0.3	0.1
1.5	0.6	0.3	0.1
1.6	0.5	0.3	0.2
1.7	0.5	0.3	0.2
1.8	0.5	0.3	0.2
1.9	0.5	0.3	0.2
2.0	0.4	0.3	0.3

(в) ρ = 0.7

2.2.2 Три хости

У цьому розділі представлено аналітичне порівняння TAGS та TAPTF у розподіленій системі із трьома хостами. TAPTF працює краще у великому діапазоні значень α, демонструючи покращені показники щодо середнього часу очікування та уповільнення, коли α становить від 1,1 до 2,0. У міру зменшення варіацій значна кількість завдань відправляється другому хосту (позначається q2), а коли α наближається до 2.0, більше завдань відправляється на третій і кінцевий господар (позначається q3). Кінцевий хост у системі TAGS зазвичай обробляє лише файл Найбільші завдання - із зменшенням варіацій ця практика виявляється поганою, що продемонстрували вищі показники TAPTF. TAPTF (де α становить від 1,1 до 2,0), показує постійні системні навантаження, тоді як TAGS демонструє різке зростання. Коли α наближається до 2.0, політика TAGS виробляє значне надлишкове навантаження, що є тривожним знаком за такої низької швидкості надходження в розподілену систему. Середній час очікування та уповільнення при помірному навантаженні системи (ρ = 0,5). Оскільки навантаження на систему зростало, TAPTF демонструє покращення у більшому діапазоні значень α (де α становить від 0,9 до 2,0).

Варто зазначити, що оптимальні значення граничних значень (si) для TAGS не можуть бути знайдені для значень α 0,5 або 0,6, що свідчить про те, що неможливо (або, принаймні, це не обчислювально) знайти обмеження, які могли б зберегти навантаження нижче 1,0 на кожному хості. Різке збільшення навантаження (і відповідного перевищення) можна спостерігати, коли α наближається до 2,0, тоді як TAPTF підтримує постійне навантаження на тій же площі. При великому навантаженні системи (ρ = 0,7) . Подібні проблеми, що виникають при навантаженні системи 0,5, виникали при знаходженні граничних значень для багатьох значень α за трьох хостів, ρ = 0,7 сценарій для TAGS. Тобто було неможливо знайти оптимальні відсікання, які б задовольняли вимозі, що навантаження має бути нижче 1,0 на всіх хостах. Щоб впоратися зі збільшеним

навантаженням системи, більша частка завдань призначається другому і третьому хосту в середньому, щоб впоратися. Дійсно, коли α дорівнює 2,0, кожному внутрішньому хосту виділяється досить рівна частка вхідних завдань (q1 = 0,4, q2 = 0,3 і q3 = 0,3). Знову можна помітити, що в міру збільшення навантаження системи діапазон значень α, де TAPTF перевершує TAGS, все ще настільки ж великий - де α становить від 0,9 до 2,0.

2.3 Висновки

У цьому розділі представлений новий підхід до призначення завдань у розподіленій системі, TAPTF (Призначення завдань, засноване на визначенні пріоритетів потоків руху). TAPTF - це гнучка політика, яка усуває недоліки існуючих підходів (описаних раніше в цьому розділі) до призначення завдань. TAPTF продемонстрував покращену ефективність (як середній час очікування, так і середнє уповільнення) у ключових областях, де страждають TAGS та випадкові політики. Найважливіше те, що TAPTF демонструє покращену продуктивність при низьких до великих варіаціях розміру завдання та великому навантаженні системи за рахунок зменшення надлишку, пов’язаного з великою кількістю перезавантажень, та інтелектуального контролю над припливом завдань до кожного внутрішнього вузла. Було виявлено, що для двох та трьох сценаріїв хосту, що в міру навантаження системи збільшується діапазон параметрів α, де було показано покращення, і величина цього вдосконалення зростала. Беручи до уваги, що TAPTF може охоплювати найкращі характеристики існуючих підходів, а також вдосконалювати їх у критичних сценаріях великого навантаження трафіку та дуже змінних розмірів завдань. TAPTF є гідною політикою розподілу навантаження в середовищах, де не є попереднім і розміри завдань невідомі апріорі.

3 ПРИЗНАЧЕННЯ ЗАВДАННЯ З МІГРАЦІЄЮ, ЩО ЗБЕРІГАЄ РОБОТУ

3.1 Кластери веб-серверів

Щоб вирішити проблему підвищення ефективності роботи великих веб-сайтів, пропонується багато розподілених архітектур для поліпшення роботи користувачів - наприклад, часу відгуку на отримання файлу або пропускної здатності. Ці архітектури намагаються забезпечити розширювані ресурси для вирішення питання масштабованості - як обслуговувати зростаючу кількість користувачів та доступну для них пропускну здатність. Оскільки пропускна здатність мережі збільшується удвічі швидше, ніж пропускна здатність сервера та динамічно генерується вміст, що становить більший відсоток запитуваного веб-вмісту, сторона сервера буде вузьким місцем зараз і в майбутньому. Архітектору таких систем доступно багато варіантів вирішення цих проблем масштабності.

Одним із них є масштабування, вдосконалення вже доступного вам вузла. Цього можна досягти двома способами. По-перше, існує поняття апаратного масштабування, де за необхідності до вашого вузла додається більше ресурсів (таких як диск, пам’ять та процесор). Це може бути корисним як тимчасове рішення, але воно не дуже масштабне (враховуючи постійне збільшення Інтернет-трафіку) і не вирішує питання надійності, якого може досягти набір серверів, що пропонують дзеркальні послуги. Інша техніка називається розширенням програмного забезпечення, де вносяться покращення продуктивності та ефективності вузла на рівні програмного забезпечення. Цього можна досягти, зробивши операційну систему більш ефективною, зменшивши накладні витрати на рівні застосування веб-сервера або вдосконалення планування запитів політика. Знову ж таки, це може досягти лише обмеженого приросту продуктивності і не вирішує питання масштабованості та надійності. Щоб впоратися зі зростаючим попитом та забезпечити певний рівень відмовостійкості та надмірності, відбувається маштабування, додаючи додаткові вузли до архітектури нашого веб-сервера.

Одним із широко застосовуваних методів є масштабне масштабування, розміщення вузлів у різних географічних місцях. Це має ряд бажаних властивостей, таких як пропонування швидших локальних дзеркал (залежно від місцезнаходження користувача) та забезпечення масштабного резервування, усуваючи будь-яку точку відмови. Незважаючи на це, дуже часто вирішення проблеми розподілу навантаження в таких архітектурах здійснюється за допомогою механізму DNS, намагаючись розумно маршрутизувати запити під час фази вирішення адреси. На жаль, це виявляється дуже чітким підходом через кешування інформації DNS на локальних серверах імен і навіть на самому клієнті. Вимірювання показали, що A-DNS контролює лише невеликий відсоток запитів - до 5% від загальної кількості запитів, що надходять у систему. Таким чином, це не цілком ефективний спосіб розподілу навантаження в розподіленій системі веб-сервера. Для більш ефективного розподілу навантаження в розподіленій системі веб-серверів необхідний більш детальний контроль, враховуючи емпіричні докази щодо характеристик робочого навантаження та знання негативних наслідків для продуктивності, які може спричинити невдалий вибір завдання. Таким чином, було розглянуто поняття локального масштабу, де була сформована локальна група внутрішніх веб-серверів, які потенційно можуть обслуговувати будь-який запит, з відповідальністю за призначення запитів, розміщених на інтерфейсному веб-сервері або диспетчерському пристрої, або навіть самі внутрішні веб-сервери (із запитами, що транслюються або багатоадресно передаються на кожен сервер).

Було вирішено зосередитись на політиці розподілу завдань, яка відповідає місцевим масштабним архітектурам. Вони є найкращим механізмом для ефективного розподілу робочого навантаження в розподіленій системі веб-серверів, забезпечуючи найбільший контроль над відправленням або маршрутизацією вхідних запитів. Існує ряд різних механізмів, які можуть становити локальний кластер веб-серверів.

Однією із загальних схем є віртуальний веб-кластер, де клієнтам видно одну IP-адресу (віртуальний IP, VIP). Цей IP не призначений певному інтерфейсному серверу або пристрою скоріше він спільний для кожного вузла сервера. Інша

домовленість - це традиційна локально розподілена веб-система, де кожен вузол сервера має унікальну IP-адресу, яку бачать усі клієнти. Основна увага зосереджена на веб-системі на базі кластерів, яка має ряд бажаних функції, які роблять його привабливим для віртуального веб-кластера або розподіленої веб-системи.

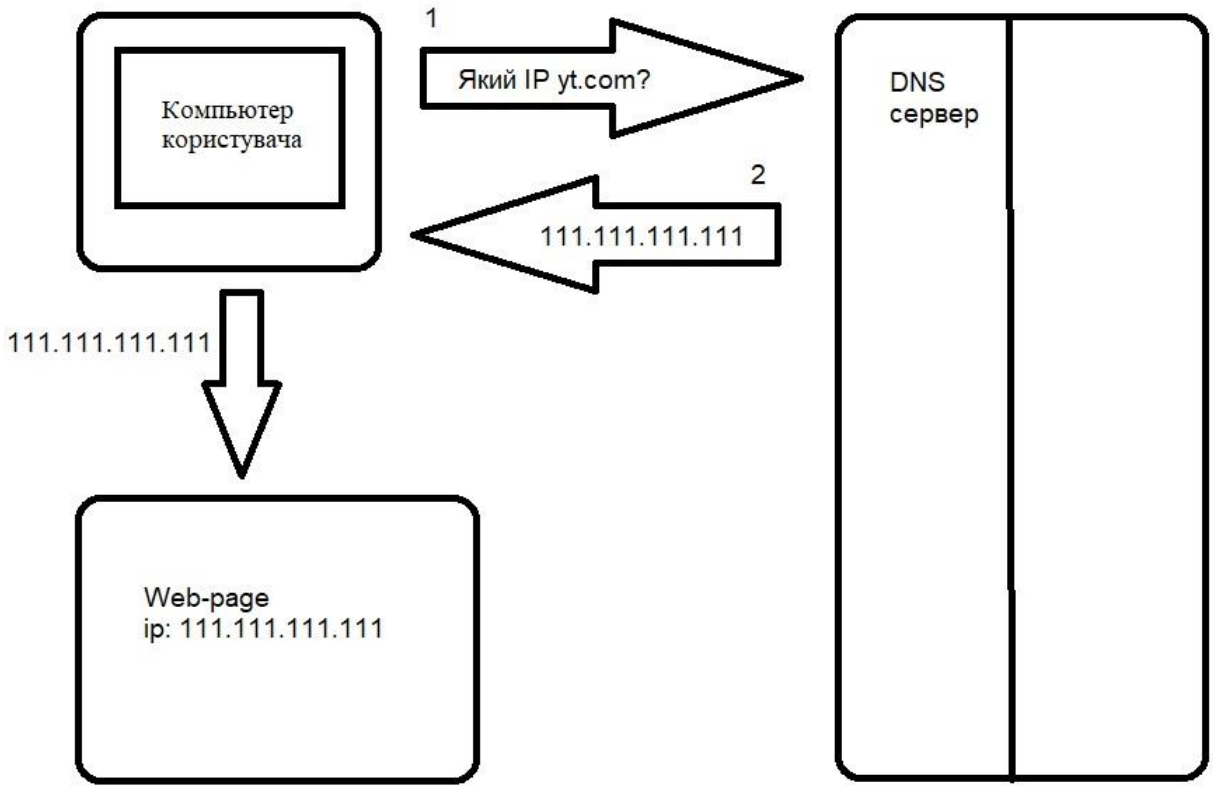


Рисунок 3.1 - Принцип роботи механізму DNS

Кластерна домовленість має єдину IP-адресу і, отже, єдину контактну точку. Він може забезпечити чіткий контроль маршрутизації запитів. Найголовніше, що він не вимагає спеціальної реконфігурації клієнтів або серверів, добре інтегрується з поточними протоколами, стандартами та клієнтами. Це має вирішальне значення, оскільки дуже бажано зробити будь-які складні маршрутизацію та перенаправлення запитів прозорими для клієнта. Ця архітектура робить можливим маршрутизацію другого рівня за допомогою

перенаправлення HTTP на рівні програми, або через передачу TCP або сплайсинг на рівні протоколу IP . Ефективність цих методів значно покращилася за останні 7 років до того рівня, коли накладні витрати на ресурси кластеру є мінімальними.

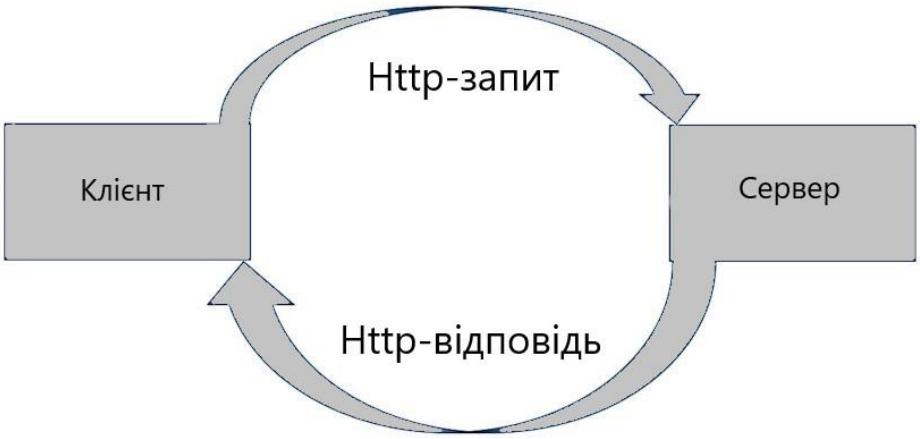


Рисунок 3.1 - Принцип роботи HTTP

Архітектура, що містить централізований диспетчер з маршрутизацією другого рівня, забезпечує ідеальну платформу для реалізації моделі TAPTF, що зберігає роботу, TAPTF-WC, представленої в цій главі.

3.2 Пов’язана робота

Дійсно, багато комерційних рішень для балансування навантаження в Інтернеті залежать від традиційних методів розподілу навантаження, а також від політики призначення найкоротшої черги-першої. У розділі 1 було показано, що ці політики працюють погано при високоінтенсивних, дуже мінливих робочих навантаженнях, вказуючи на те, що потрібні сучасні методи, що стосуються негативних характеристик цих робочих навантажень. Дві останні політики, які були запропоновані спеціально для вирішення питань балансування навантаження на кластерних веб-серверах, - EQUILOAD та ADAPTLOAD. Відповідно до політики EQUILOAD, внутрішні сервери постійно контролюють вхідне робоче навантаження, яке вони отримують, і періодично повторно узгоджують свою згоду щодо розміру запитів, які їм слід розподілити. Запропонована методологія,

яка характеризує робоче навантаження в Інтернеті, оснащуючи їх розподілами фазового типу, які дуже нагадують вихідний розподіл. Характеристика може бути виконана як в режимі офлайн (наприклад, на повній трасі), так і в Інтернеті (періодично перевіряючи навантаження, побачене до цього часу, та коригуючи його відповідність). Політика ґрунтується виключно на розподіл розмірів вхідних завдань.

EQUILOAD не є справді адаптивним - особливі події можуть суттєво змінити відносну популярність документів веб-сервера, через що обрані межі перестануть бути оптимальними. Чітка політика планування повинна враховувати швидкість прибуття вхідних завдань та розподіл їхніх потреб у послугах, і що будь-які зміни в спостережуваній пошкодженості середньої швидкості прибуття повинні спричинити зміну параметрів політики для прийняття до нової швидкості прибуття. Таким чином була сформульована політика ADAPTLOAD. ADAPT-LOAD використовує історію робочого навантаження для адаптації меж. Моделювання вказує, що знання про кінцеве навантаження може бути використано як хороший показник майбутньої поведінки. Політика ADAPTLOAD досліджує останні K запити, щоб побудувати дискретні гістограми даних, необхідні для визначення меж розподілу наступних K запитів. K не повинен бути ні занадто малим (оскільки ADAPTLOAD потребує статистично значущої вибірки), ні занадто великим (оскільки він повинен адаптуватися до коливань). Якщо значна частина робочого навантаження складається з кількох популярних файлів, можливо, не вдасться вибрати N чітких меж (тобто для кожного сервера), і при цьому переконатися, що кожен інтервал відповідає рівному обсягу навантаження, отриманого на кожному сервері. Таким чином, введено імовірнісні межі для боротьби з цією проблемою. Імовірність ρі присвоюється кожній граничній точці si, виражаючи частину запитів на розмір файлу si to обслуговуватися сервером i. Решта 1 - пі запитів для цього розміру файлу обслуговується сервер i + 1 або додаткові сервери. ADAPTLOAD порівнюється із Найменшою зваженою чергою (JSWQ), де “довжина кожної черги в системі зважується за розміром запитів, що стоять у черзі” - по суті підхід з найменшою роботою, що залишився. При низькому навантаженні JSWQ працює краще, ніж

ADAPTLOAD, який може направити запит на сервер, який зайнятий, навіть коли сервер не працює доступний (через попередньо обчислені межі). У періоди перехідних перевантажень ADAPT-LOAD перевершує JSWQ, досягаючи нижчого середнього уповільнення та швидшого повернення до прийнятних рівнів перевантаження. ADAPTLOAD керує послідовно невеликими уповільненнями майже для всіх класів запитів. Як і інші підходи, що базуються на розмірах, EQUILOAD та ADAPTLOAD обмежуються програмами, де розміри завдань відомі заздалегідь.

3.3 Концептуальний вигляд моделі TAPTF-WC

Розглядаючи розширення політики TAPTF, що називається TAPTF-WC - Призначення завдання, засноване на визначенні пріоритетів потоків трафіку з міграцією, що зберігає роботу. TAPTF має ряд бажаних характеристик , що робить його ідеальною базою, на якій можна будувати політику призначення завдань, що зберігає роботу. На відміну від TAPTF (і TAGS), де завдання перезапускаються з нуля, якщо вони перевищують межу на даному хості, TAPTF-WC зберігає кожну частину роботи, яку він виконує на даному хості. Після переходу на новий хост, TAPTF-WC відновлює роботу з того місця, де він припинив обробку перед міграцією. Це узгоджується з розподіленими системами, які підтримують міграцію, що зберігає роботу.

У TAPTF-WC прибуття завдань диспетчеру відбувається за процесом Пуассона зі швидкістю λ. Потім диспетчер призначає завдання кожному з n хостів, називають їх Host i, $1 \leq i \leq n$, випадковим чином із ймовірністю qi. Використовуючи добре відому властивість процесу Пуассона, потік прибуття до Хоста і також є процесом Пуассона зі швидкістю λqi. Розміри завдань (розподіл послуг) слідують за обмеженим розподілом Парето B (α, k, p). Наступне присвоєння навантажень хостам:

1) завдання, які виконуються в черзі O хоста i, - це ті, чий початковий (оригінальний) розмір знаходяться в діапазоні [k, si], а залишковий розмір <si;

2) завдання, які виконуються в черзі R хосту i (де 1 <i ≤ n) - це ті, початкові (оригінальні) розміри яких знаходяться в діапазоні [si − 1, si], а залишковий розмір <si - si − 1 , де k <s1 <s2 <s3 <. . . <sn = p.

Мета полягає в обчисленні граничних значень (значень si) з метою мінімізації критично важливих показників ефективності, таких як середній час очікування або середнє уповільнення. Як і TAPTF, кожен хост, крім хосту 1, вміщує дві черги, звичайну (O) чергу та чергу перезапуску (R). Завдання, надіслані хосту з диспетчера, приєднуються до черги O. Якщо залишковий розмір завдання не потрапляє в правильний діапазон (тобто він перевищує обмеження обробки si, пов'язане з деяким хостом i), вони переміщуються до черги R, що належить наступному хосту по рядку. На відміну від стандартного підходу TAPTF, будь-яка обчислювальна робота, виконана на даному хості, зберігається, коли вона переміщується до черги перезапуску (R) наступного хоста. Цей процес повторюється до тих пір, поки ці завдання не зможуть завершитися. Завдання, що зберігаються в черзі O (отримані безпосередньо від диспетчера), мають пріоритет обслуговування перед тими, що знаходяться в черзі R. Однак завдання, яке обслуговується в черзі R, не буде попередньо скасовано з обслуговування при надходження завдання в чергу O. Всі завдання в чергах R та O подаються за принципом "Хто прийшов, хто першим". Вищевказана модель відрізняється від TAGS тим, що ми встановили межі розмірів завдань на кожному хості таким чином, що всі завдання з залишковими розмірами, меншими або рівними фіксованій точці відсічення, обробляються хостом. Це означає, що завдання може бути відправлене будь-яким хостам спочатку без попереднього відправлення на хост 1, як у TAGS з метою збереження майна, про який попит на роботу не відомий апріорі. TAPTF-WC (як TAPTF) використовує подвійні черги на кожному хості, щоб пришвидшити потік коротших завдань. На малюнку 3.2 показана система TAPTF-WC для 4 хостів. «Оригінальний розмір» стосується початкового розміру завдань, які будуть виконуватися на цьому хості. "Залишилося" - це завдання із залишковим часом обробки, меншим за граничний показник, який буде завершено на цьому хості.

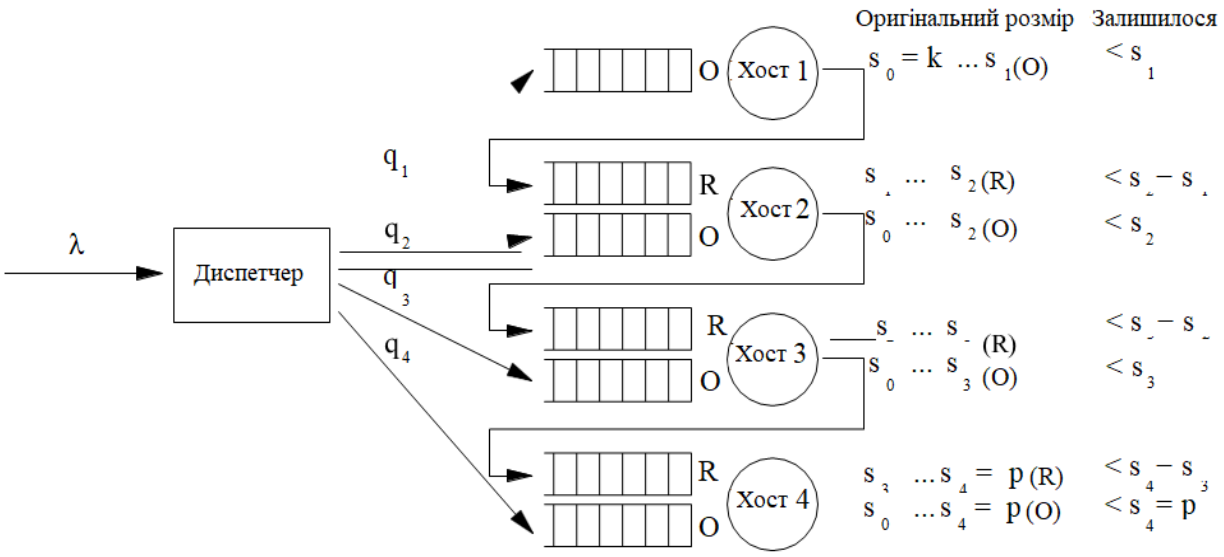


Рисунок 3.2: TAPTF-WC з 4 хостами

3.3.1 Вибір границь

Подібно до TAPTF та інших політик, заснованих на розмірах, вибір цих обмежень, що використовуються в TAPTF-WC, може мати величезний вплив на продуктивність розподіленої системи. Принципи пошуку найкращих граничних рівнів такі самі, як описано в нашому попередньому аналізі TAPTF. На оптимальні граничні показники для TAPTF-WC впливають розподіл розміру завдання (Bounded Pareto B (k, p, α)) та швидкість надходження завдання у розподілену систему, λ. Враховуючи вищезазначені параметри та використовуючи математичні попередні етапи, ми можемо спробувати отримати оптимальні граничні значення (si) для кожного з наших хостів у системі TAPTF-WC.

Оптимальний середній час очікування та середнє уповільнення для випадку двох і трьох хостів можна отримати, використовуючи модель, шляхом вирішення оптимальних значень граничних значень (si) та дробових часток (qi) з використанням математики. У випадку з трьома хостами si можна знайти чисельно, але ци повинні бути налаштовані вручну. Для чотирьох і більше хостів усі параметри (si та qi) потрібно було б налаштувати вручну. Чим менший параметр α, тим вища мінливість і менший відсоток завдань, що становить 50%

навантаження. TAGS (і згодом TAPTF, який міг працювати ідентично TAGS, встановлюючи q1 = 1,0 при розумності) може використовувати цю властивість важкохвостого розподілу, виконуючи всі (малі) завдання на першому хості, залишаючи їх під легким і помірним навантаженням, тоді як найбільші завдання відфільтровані, щоб врешті-решт оброблятись останні господарі. Незважаючи на діяльність за нових припущень, політика TAPTF-WC часто доцільна діяти подібним чином - особливо при надзвичайно змінних робочих навантаженнях. Таким чином, коли TAPTF-WC встановлює q1 = 1,0, він ефективно поводиться як економічна версія політики TAGS. TAGS у своєму первісному вигляді не враховували міграції, що зберігають роботу, оскільки вони найбільше підходили для пакетних та суперобчислювальних засобів, де міграція, що зберігає роботу, гарантовано не буде доступною і не буде використана з міркувань продуктивності. Цей особливий випадок TAPTF-WC ми позначимо як TAGS-WC (Призначення завдання на основі розміру вгадування - Збереження роботи).

3.4 Аналітичне порівняння

Було проведено аналітичне порівняння підходу TAPTF-WC з політикою TAGS, що зберігає роботу (TAGS-WC), щоб встановити ефективність відповідної політики. Це забезпечило найкращу точку порівняння, оскільки обидві політики мають однакові обмеження (тобто відсутність апріорних знань про необхідність обслуговування завдання, відсутність переваги, доступна міграція, що зберігає роботу). Саме ці обмеження виключають пряме EQUILOAD / ADAPTLOAD та розмір. Крім того, кожну політику збереження роботи порівнювали з початковою формою (де робота не зберігається) - TAGS до TAGS-WC та TAPTF до TAPTF-WC. Основна зацікавленість в кількісному визначенні вигод, які міграція, що зберігає роботу, може забезпечити, якщо вона доступна. Ці підходи оцінювались у різних умовах та їх ефективність порівнювалась із використанням найважливішої з метрик - середнього часу очікування та середнього уповільнення. Був розглянутий великий діапазон значень α, від 0,5 до 2,0. Кожне значення α оцінювалось для різних системних навантажень (ρ) - 0,3 (низьке навантаження),

0,5 (помірне навантаження) та 0,7 (велике навантаження). Показники продуктивності були розраховані для цих сценаріїв завантаження як для двох, так і для трьох хост-систем.

Таблиця 3.1: Розподіл завдань у TAPTF-WC - 2 хоста, ρ = 0,3

a	q_1	q_2
1.0	0.98	0.02
1.1	0.97	0.03
1.2	0.96	0.04
1.3	0.95	0.05
1.4	0.93	0.07
1.5	0.91	0.09
1.6	0.88	0.12
1.7	0.84	0.16
1.8	0.81	0.19
1.9	0.77	0.23
2.0	0.74	0.26

(а) $E\{W\}$

a	q_1	q_2
1.0	1.00	0.00
1.1	0.99	0.01
1.2	0.99	0.01
1.3	0.98	0.02
1.4	0.96	0.04
1.5	0.94	0.06
1.6	0.91	0.09
1.7	0.88	0.12
1.8	0.84	0.16
1.9	0.80	0.20
2.0	0.77	0.23

(б) $E\{S\}$

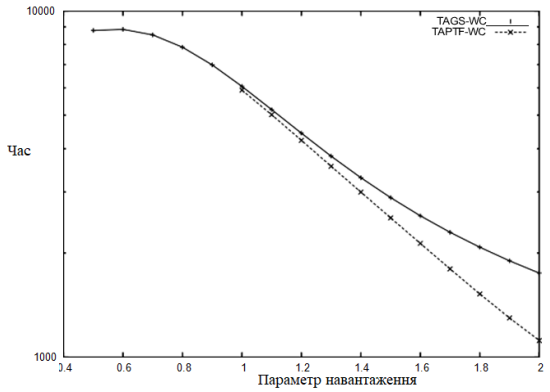
Узагальнена математична модель TAPTF-WC також використовується для моделювання поведінки політики TAGS, що зберігає роботу, TAGS-WC, встановлюючи q1 = 1,0 (і згодом q2 ... qn рівним 0). Це призводить до заперечення подвійних черг та кількох точок входу, що використовуються політикою TAPTF-WC, і змушує її вести себе як економічна версія існуючої політики TAGS. Знайдено оптимальні граничні значення щодо середнього часу очікування та середнього уповільнення як для TAPTF-WC, так і для TAGS-WC за допомогою функції NMinimize в Mathematica, щоб отримати найкраще (і справедливе) порівняння політик призначення завдань. Це досягається знаходженням значень si у кожному екземплярі, які дають місцеві мінімуми для очікуваного часу очікування E (W) та очікуваного середнього уповільнення E (S). Вибір розподілу завдань у диспетчера також є критичним, де qi контролює частку завдань, призначених Host і від диспетчера. У випадку двох хостів, були знайденні найкращі чисельні значення на кожному хості як для si, так і для qi, що призводить до ідеальних умов для нашої політики призначення завдань, TAPTF-

WC, для E (W) або E (S). Для випадку з трьома хостами визначені оптимальні значення si, але налаштувати параметри qi повинні вручну. Використовуючи результати з обох випадків та експериментів, щоб знайти хороші параметри. У трьох випадках приймання були використані окремі набори параметрів qi, що призвело як до хороших E (W), так і E (S). З метою чітких та значущих результатів усі порівняння середнього значення час очікування та середнє уповільнення, які виконуються в цьому розділі, використовують відповідні політики TAPTF-WC та TAGS-WC, оптимізовані для цього показника. Графіки часу очікування та уповільнення представлені в шкалі журналу для горизонтальної осі.

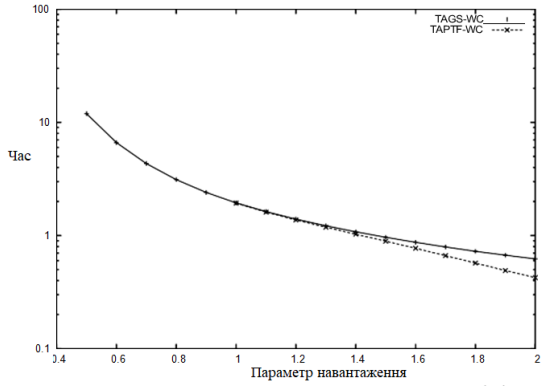
3.4.1 Два хости

У цьому розділі розглянуто аналітичне порівняння TAGS-WC та TAPTF-WC у системі розподілених двох хостів. Результати для TAPTF-WC відображаються лише там, де вони перевершують TAGS-WC, оскільки TAGS-WC тепер є підмножиною поведінки політики TAPTF-WC, яка може бути прийнята, коли це розумно для кращої роботи. З таблиці 3.1 помітно, що оптимальні значення параметрів дотримуються подібних тенденцій, що демонструються політикою TAPTF. Дійсно, значення qi майже ідентичні для моделей TAPTF, що не зберігають роботу та не зберігають роботу за цим сценарієм. Зі збільшенням варіації менша кількість завдань спрямовується на другий хост, поки врешті-решт жодні завдання не надсилаються безпосередньо на цей хост, прагнучи до поведінки, яку ми виконали як TAGS-WC (тобто модель збереження роботи TAGS). З малюнків 5.3 (а) та (б) очевидне розумне покращення очікуваного часу (від $\alpha = 1,1$) та уповільнення (від $\alpha = 1,3$) для політики TAPTF-WC щодо втілення TAGS-WC за низької системи навантаження 0,3. Масштаб цього вдосконалення розширюється із зменшенням мінливості, наближаючись до $\alpha = 2,0$. Рисунки 5.3 (в) та (г) зображують покращення, яке може досягти економія міграції (якщо вона доступна) за час очікування та уповільнення політики TAGS. З графіків спостерігається, що поліпшення ледь відчутно - це очікується при такому

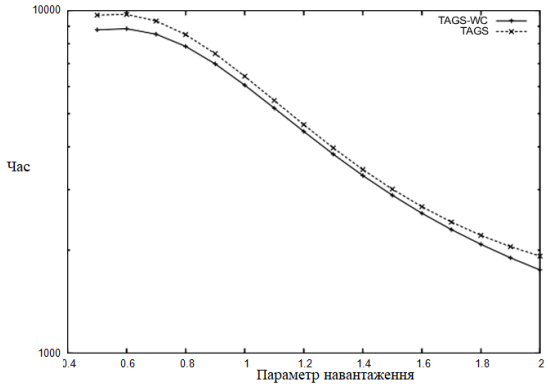
низькому навантаженні системи, оскільки надлишок, що генерується стандартною політикою, мало впливає, оскільки швидкість прибуття та очікувана довжина черги низькі. Вибір застосовуваної політики призначення завдань є менш критичним при такому низькому навантаженні. Рисунки 5.3 (д) та (е) протиставляють ефективність політики TAPTF та її економічні зміни, що позначаються як TAPTF-WC. Як і політики TAGS, є незначна різниця у продуктивності двох варіацій TAPTF через низьке навантаження системи в поєднанні з тим, що політика TAPTF була розроблена в першу чергу для зменшення кількості надлишку. Це було досягнуто за допомогою декількох методів, що мінімізують кількість передач, коли завдання перезапускаються з нуля.



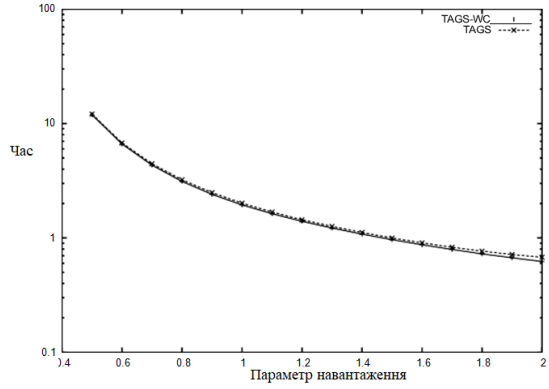
(а) E(W)



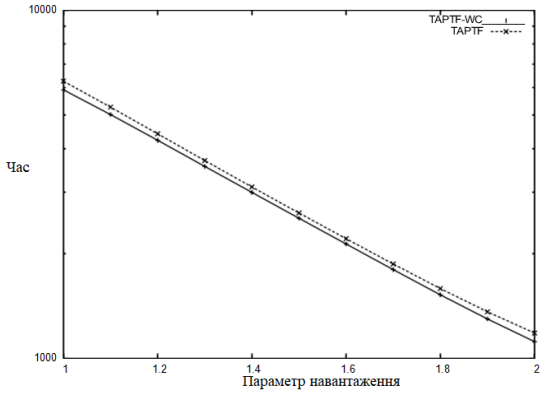
(б) E(S)



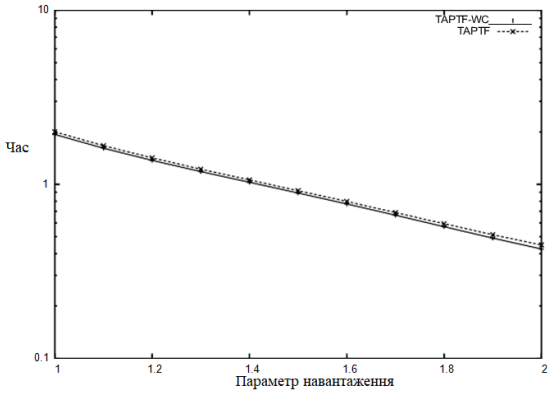
(в) E(W) - TAGS



(г) E(S) - TAGS



(д) E(W) – TAPTF



(е) E(S) - TAPTF

Рисунок 3.2: Продуктивність розподіленої системи з двома хостами із завантаженням системи 0,3. Очікуваний час очікування та уповільнення зображено в (а) та (б) для політик збереження роботи, оптимізованих для цих відповідних показників. У (в), (г), (д) та (е) порівнюються версії TAGS та TAPTF, що зберігають роботу та не зберігають роботу.

У таблиці 5.4 показано оптимальні значення параметрів qi для випадку двох хостів при навантаженні системи 0,5. Другому хосту призначено більше завдань, ніж це було при навантаженні системи 0,5. Як і очікувалось, із збільшенням варіації (α наближається до 0) все менше завдань призначається другому хосту, доки врешті хост не отримає всі завдання від диспетчера. Політика TAPTF-WC демонструє покращену ефективність очікуваного часу очікування (від α = 1,1) та уповільнення (від α = 1,2) порівняно з політикою TAGS-WC. Величина цього збільшення продуктивності зростає, коли α наближається до 2,0, з покращенням приблизно в два рази часу очікування та уповільненням, коли α = 2,0. Знову ж таки, як це було при низькому навантаженні системи 0,3, ступінь вдосконалення політики TAGS-WC порівняно з політикою TAGS при помірному навантаженні системи 0,5 невелика. Незважаючи на збільшення навантаження системи, цього все ще недостатньо, щоб істотно змінити показники продуктивності. Характеристики графіку здебільшого однакові як для очікуваного часу очікування, так і для уповільнення, без помітної різниці в продуктивності. Як було описано раніше, стандартна політика TAPTF вже має заходи щодо зменшення обсягу та ефекту надмірної обробки, яка існує через міграцію, що не

зберігає роботу, що пояснює подібність результатів. Дві тенденції, які спостерігалися раніше, продовжуються. По-перше, із збільшенням навантаження системи більше завдань призначається безпосередньо другому хосту (для заданого значення α) від диспетчера. По-друге, коли варіація збільшується (а α зменшується), другому хосту призначається менше завдань, доки врешті-решт ніякі завдання не призначаються безпосередньо там від диспетчера. Оскільки α зменшується (з α = 1,1), ми можемо спостерігати збільшення приросту продуктивності TAPTF-WC порівняно з TAGS-WC як за очікуваним часом очікування, так і за сповільненням. Коли α досягає 2,0, TAPTF-WC демонструє значне покращення продуктивності приблизно в 2,5 рази за очікуваний час очікування та уповільнення. При високому навантаженні системи 0,7 помітна різниця в продуктивності між стандартною політикою TAGS та політикою збереження роботи TAGS, TAGS-WC. Надмірна обробка, створена політикою TAGS, тепер має помітно несприятливий вплив на продуктивність.

Таблиця 3.2: Розподіл завдань у TAPTF-WC - 2 хости, ρ = 0,5

a	q_1	q_2
0.9	0.98	0.02
1.0	0.97	0.03
1.1	0.96	0.04
1.2	0.94	0.06
1.3	0.92	0.08
1.4	0.89	0.11
1.5	0.86	0.14
1.6	0.82	0.18
1.7	0.79	0.21
1.8	0.76	0.24
1.9	0.73	0.27
2.0	0.70	0.30

(а) E(W)

a	q_1	q_2
0.9	1.00	0.00
1.0	0.99	0.01
1.1	0.99	0.01
1.2	0.98	0.02
1.3	0.96	0.04
1.4	0.94	0.06
1.5	0.90	0.10
1.6	0.87	0.13
1.7	0.83	0.17
1.8	0.79	0.21
1.9	0.76	0.24
2.0	0.73	0.27

(б) E(S)

Коли α = 2,0, існує коефіцієнт приблизно 3,5 різниці між TAGS і TAGS-WC, що свідчить про згубний ефект, який надлишок обробки викликає, коли швидкість надходження вище і очікувана довжина черги збільшується. Як і

очікувалось, показники все ще залишаються однаковими з тих самих причин, наведених раніше в цьому розділі.

3.4.2 Три хости

Як і в попередньому розділі, результати для TAPTF-WC показуються лише там, де вони перевершують TAGS-WC. На таблиці 3.3 показано параметри qi, знайдені (шляхом експериментів) для випадку трьох хостів при низькому навантаженні системи 0,3. Ці параметри є більш грубими, ніж ті, що були знайдені (за допомогою чисельної оптимізації) у двох випадках хосту. Однак подібні тенденції можна спостерігати.

Таблиця 3.3: Розподіл завдань у TAPTF-WC - 2 хости, ρ = 0,7

a	q_1	q_2
0.9	0.97	0.03
1.0	0.96	0.04
1.1	0.93	0.07
1.2	0.89	0.11
1.3	0.85	0.15
1.4	0.82	0.18
1.5	0.78	0.22
1.6	0.75	0.25
1.7	0.72	0.28
1.8	0.69	0.31
1.9	0.67	0.33
2.0	0.66	0.34

(а) E(W)

a	q_1	q_2
0.9	1.00	0.00
1.0	0.99	0.01
1.1	0.98	0.02
1.2	0.96	0.04
1.3	0.92	0.08
1.4	0.88	0.12
1.5	0.84	0.16
1.6	0.80	0.20
1.7	0.76	0.24
1.8	0.73	0.27
1.9	0.70	0.30
2.0	0.68	0.32

(б) E(S)

Можна побачити, що більшість завдань призначено хосту 1, тоді як менші пропорції призначені хостам 2 та 3 відповідно. Зі збільшенням варіацій більшість завдань надсилається безпосередньо на хост 1, з меншою часткою на хост 2, і жодним завданням не призначається хост 3. Незважаючи на низьке навантаження, ми все ще можемо спостерігати суттєві вдосконалення цих показників у широкому діапазоні значень α, від великих коливань, де α = 0,9, до нижчих

варіацій, коли α наближається до 2,0. Коли α = 2,0 TAPTF-WC покращується на TAGS-WC в 2 рази як очікуваний час очікування, так і уповільнення. Незважаючи на низьке навантаження на систему (0,3), політика TAGS щодо збереження роботи TAGS-WC демонструє чітке покращення продуктивності у всіх досліджуваних областях. Політика TAGS-WC покращує стандартну політику TAGS в середньому приблизно в 1,5 рази, як очікуваний час очікування, так і уповільнення. На таблиці 3.4 показано параметри qi, знайдені (знову ж таки шляхом експериментів) для випадку трьох хостів, при помірному навантаженні системи 0,5. У кількох випадках (як і очікувалося) було виявлено, що для даного значення α другий хост отримує більшу частку завдань, ніж це відбувалося під навантаженням системи 0,3.

Таблиця 3.4: Розподіл завдань у TAPTF-WC - 3 хости, ρ = 0,5

a	q_1	q_2	q_3
0.9	0.95	0.05	0.0
1.0	0.90	0.10	0.0
1.1	0.80	0.20	0.0
1.2	0.75	0.25	0.0
1.3	0.70	0.30	0.0
1.4	0.60	0.40	0.0
1.5	0.60	0.30	0.1
1.6	0.60	0.30	0.1
1.7	0.60	0.30	0.1
1.8	0.60	0.30	0.1
1.9	0.60	0.30	0.1
2.0	0.60	0.30	0.1

З таблиць спостергігається стабільний фактор покращення в порівнянні з політикою TAPTF-WC в 1,3 рази очікуваного часу очікування. Це в усьому діапазоні значень α, від α = 0,9 (сильно мінлива) до α = 2,0 (мала варіація). Аналогічно, коефіцієнт поліпшення в 1,4 рази можна побачити в очікуваному уповільненні, якщо порівнювати TAPTF-WC з TAPTF у тому ж діапазоні значень α. Параметри qi, знайдені для системи трьох хостів з великим навантаженням системи 0,7, зображені у таблиці 3.5. Протиставляючи параметри qi, що

використовуються тут, з параметрами, що використовуються для менших навантажень (до трьох хостів), помітна делікатна тенденція, коли вигідно відправляти більше завдань на другий і третій хости, коли зростає навантаження системи. Очевидна вигода може бути помітна як у часі очікування, так і особливо в уповільненні, починаючи від умов великих коливань (α = 0,9) і розширюючись за величиною, коли α наближається до 2. Як це було у випадку великого навантаження (для трьох господарів) у попередній главі для TAGS TAGS-WC не може працювати в певних умовах навантаження. Зокрема, коли α> 1,6, не вдається знайти відповідних відсікань, які утримують навантаження нижче 1 у всіх хостів. Коли α = 1,6 TAPTF-WC має приблизне покращення порівняно з TAGS-WC в часі очікування та уповільнення коефіцієнта в 2,2 рази.

Таблиця 3.5: Розподіл завдань у TAPTF-WC - 3 хости, ρ = 0,7

a	q_1	q_2	q_3
0.9	0.95	0.05	0.0
1.0	0.9	0.1	0.0
1.1	0.8	0.2	0.0
1.2	0.7	0.3	0.0
1.3	0.6	0.4	0.0
1.4	0.6	0.3	0.1
1.5	0.6	0.3	0.1
1.6	0.6	0.3	0.1
1.7	0.6	0.3	0.1
1.8	0.6	0.3	0.1
1.9	0.5	0.3	0.2
2.0	0.5	0.3	0.2

Як час очікування, так і уповільнення підтримують відносно рівну реакцію, а додавання міграції збереження роботи дозволяє TAGS-WC обслуговувати ширший діапазон варіацій робочого навантаження, ніж це міг TAGS. TAPTF та його варіант збереження роботи, TAPTF-WC, для трьох хостів при високому навантаженні системи 0,7. Покращення часу очікування та особливо уповільнення зростало із збільшенням навантаження на систему. Коли α = 2, коефіцієнт

покращення 1,6 для часу очікування та 1,84 уповільнення для політики TAPTF-WC порівняно зі стандартною політикою TAPTF.

3.5 Міграція фіксованих витрат

Варіюючи фіксовані витрати на міграцію та відновлення, вивчаючи сценарії, де $\gamma s = \gamma d = 750$, 1500 та 3000. З часу очікування, зображеного на малюнку 5.14 (a), це помітно коли $\alpha \geq 1{,}2$, будь-які прибутки, отримані завдяки політиці збереження роботи TAGS, враховуються фіксована вартість міграції, при цьому оригінальна модель TAGS працює незначно краще, ніж будь-яка інша моделей TAGS-WC на основі фіксованої вартості. І навпаки, коли α наближається до 0,5, міграція, що зберігає роботу, демонструє незначне покращення, навіть коли виникають витрати.. Оскільки вартість міграції для TAGS-WC зростає, уповільнення поступово погіршується. Існує лише незначна різниця у продуктивності у всіх випадках, причому як час очікування, так і уповільнення поступово погіршуються із збільшенням фіксованих витрат на міграцію. Навіть за найнижчих фіксованих витрат на міграцію ($\gamma d = \gamma s = 750$), оригінальна модель TAPTF (яка перезапускає завдання з нуля) працює краще у всіх випадках.

Щодо моделей TAGS-WC, заснованих на витратах, уповільнення значно зростає із зменшенням варіації розміру завдання, особливо там, де фіксована вартість міграції висока ($\gamma d = \gamma s = 3000$). Спостерігається незначна різниця між ними, оскільки час очікування та уповільнення зростає поступово із зростанням вартості міграції. У цьому випадку модель витрат TAGS-WC показана лише там, де $\gamma d = \gamma s = 750$. Не вдається знайти параметри для $\gamma d = \gamma s = 1500$ або 3000, оскільки навантаження не може утримуватися нижче 1 на кожному хості. Однак все ще можемо спостерігається триваюча тенденцію, коли модель, заснована на витратах TAGS-WC, страждає в міру зменшення варіації розміру завдання, оскільки все більше і більше завдань переносяться з огляду на те, що навантаження потрібно розподіляти більш рівномірно, щоб навантаження було менше 1 на кожному хості. Оскільки понесені витрати на кожну з цих міграцій, час очікування швидко погіршується. Однак із збільшенням варіації штраф TAGS

за перезапуск завдань з нуля набагато перевищує фіксовану вартість міграції завдання економічно. Є кілька сюрпризів, враховуючи час очікування та уповільнення для варіантів TAPTF. Однак ми зазначаємо, що із збільшенням варіації розміру завдання оригінальна модель TAPTF має найвищий час очікування та уповільнення порівняно з TAPTF-WC та кожним із варіантів, що базуються на фіксованих витратах.

3.5.1 Пропорційна міграція витрат

Пропорційні витрати на міграцію та відновлення змінюються для вивчення сценаріїв, де βs = βd = 0,25, 0,5 та 0,75. Це прирівнюється до покарання на джерелах та хостах призначення 25%, 50% та 75% від початкової потреби в завданні відповідно.. Як для часу очікування, так і для уповільнення, у міру зростання витрат на одиницю міграції, показники збільшуються пропорційно. Розбіжність збільшується із збільшенням варіації розміру завдання та збільшенням витрат на одиницю продукції. Це має інтуїтивний сенс, оскільки, як правило, виникає більше великих завдань, а оскільки витрати на міграцію пропорційні розміру завдання, збільшується накладні витрати на перенесення цих завдань. Результати поступово погіршуються, оскільки витрати на міграцію на одиницю зростають як для часу очікування, так і для уповільнення. Знову спостерігаються певні розбіжності, оскільки розподіл обсягу завдань стає більш мінливим у поєднанні зі збільшенням вартості міграції на одиницю. Однак це менш виражено, ніж те, що спостерігалося для варіантів TAGS. Враховуючи, що витрати на міграцію пропорційні розміру завдання, порядок виконання різних варіантів TAGS та TAPTF у порівнянні між собою залишається незмінним. Відповідно до того, що сталося під цим навантаженням системи з фіксованою вартістю міграції, оптимальні налаштування для TAGS-WC з пропорційними витратами на міграцію можуть бути обчислені лише там, де βs = βd = 0,25. У цьому конкретному випадку результати можна було отримати лише там, де $0,9 \leq \alpha \leq 2,0$. Це пов'язано з неможливістю утримувати навантаження на обох хостах нижче 1 в інших випадках. Результати, отримані для TAGS-WC, де βs = βd = 0,25,

стабільно гірші, ніж TAGS і TAGS-WC майже у всіх випадках. Поєднання високого навантаження та високих витрат на міграцію на одиницю товару починає брати своє значення на показаних варіантах, заснованих на витратах TAPTF-WC. Це особливо випадок, коли змінюється розмір завдання. Дійсно, оскільки більші завдання виконуються регулярніше, вартість міграції цих завдань висока

3.6 Висновки

У цьому розділі представлені економічні розширення політики TAPTF, TAPTF-WC. Також представлені економічні варіанти політики призначення завдань TAGS (позначені як TAGS-WC) як особливий випадок TAPTF-WC. Змодельовано TAPTF-WC (і TAGS-WC), де витрати на міграцію були незначними, фіксованими та пропорційними первинному сервісному запиту завдання. Там, де витрати на міграцію були незначними, TAPTF-WC демонстрував стабільно хорошу продуктивність у широкому діапазоні сценаріїв розподілу завдань завдяки своїй гнучкій природі, розподіляючи роботу на кількох хостах, коли було розумно, та відокремлюючи короткі потоки завдань від великих за допомогою подвійних черги. Завдання переносяться економічно, зменшуючи покарання, пов’язане з перенесенням завдань, яке міститься в існуючих політиках, таких як TAGS та TAPTF, які перезапускають завдання під час перенесення. Це робить політику TAPTF-WC добре придатною для середовищ, де доступні засоби економії міграції, що зберігають роботу, такі як кластери веб-серверів.

Порівнюючи TAGS та TAPTF з відповідними варіантами збереження роботи, TAGS-WC продемонстрував більші покращення. Це було пов’язано з більш марнотратним характером політики TAGS за сценаріїв великого навантаження системи та більш помірних коливань обсягу завдань, що було виправлено шляхом додавання економії роботи (TAGS-WC). Підхід TAGS-WC корисний в умовах великих до екстремальних змін, оскільки ідеально підходить (за задумом) до таких сценаріїв. І навпаки, TAPTF-WC добре підходить для умов великих до помірних та низьких коливань, а також високих навантажень системи.

Це пояснюється його здатністю справлятися як із дуже мінливими робочими навантаженнями, утримуючи малі та великі завдання від перешкоджання один одному через подвійні черги та міграцію, так і помірно мінливі робочі навантаження, використовуючи простої хостів, оскільки завдання стають більш рівномірними. Таким чином, політика TAPTF-WC (що охоплює TAGS-WC) добре підходить для розподілу навантаження в широкому діапазоні різних навантажень в середовищах, де розміри завдань апріорі невідомі, і доступна мізерна економія роботи, що зберігає незначну вартість.

ВИСНОВКИ

Ця робота представляє собою розробку нової політики розподілу завдань для розподілених завдань. Було проаналізовано різні політики планування, що використовуються в обчислювальних системах, і їх вплив на загальні метрики масового обслуговування. Також розглянулись різні методи, що використовуються для вимірювання та розповсюдження інформації про навантаження в комп'ютерній системі, роль міграції процесів у розподілених системах.

Далі була створена нова політика призначення завдань для пакетних та супер-обчислювальних кластерів, яка називається "Task Assignment based on Prioritising Traffic Flows" (TAPTF), забезпечуючи гнучку та високоефективну політику призначення завдань, і зменшив деякі накладні витрати на обробку, що було проблематично при існуючій техніці, TAGS. TAPTF продемонстрував хорошу продуктивність у широкому діапазоні робочих навантажень та сценаріїв завантаження системи. В останньому розділі було розглянуто практичне застосування нової політики при міграції з збереженням роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Osogami T., Harchol-Balter M. Necessary and sufficient conditions for representing general distributions by coxians. *Computer Performance: Modelling Techniques and Tools (LNCS 2794)*. 2003 , page 182–199 .

2) Pai S., Druschel P., and Zwaenepoel W. Flash: An efficient and portable Web server. *In Proceedings of the USENIX 1999 Annual Technical Conference*, 1999.

3) Harchol-Balter M., Crovella M. E., and Murta C. D. *On choosing a task assignment policy for a distributed server system*. Journal of Parallel and Distributed Computing, 2004. Page 204–228.

4) Ferrari D. A study of load indices for load balancing schemes. Technical Report CSD-85-262, UC Berkeley, 1985.

5) Feldmann A. and Whitt W. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. In INFOCOM ’97: Proceedings of the INFOCOM ’97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Soci- eties. Driving the Information Revolution, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7780-5, page 1096–1104 .

6) Ferrari D. and Zhou S. *A load index for dynamic load balancing*. In ACM ’86: Proceedings of 1986 ACM Fall joint computer conference, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press. ISBN 0-8186-4743-4, page 684–690.

7) Adams K. and Agesen O. *A comparison of software and hardware techniques for x86 virtual- ization*. In ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-451-0, page 2–13 .

8) Amazon.com, Inc. Amazon Elastic Compute Cloud (Amazon EC2), 2007. URL: http://aws.amazon.com/ec2 (дата звернення: 15.05.2021).

9) Arlitt M. and Jin T.. *Workload characterization of the 1998 World Cup web site*. IEEE Network, May/Jun 2000, page 30-37.

10) Wolff R. W. *Stochastic Modeling and the Theory of Queues*. Prentice-Hall, 1989.

11) Aversa L. and Bestavros A.. *Load balancing a cluster of web servers using distributed packet rewriting*. In Proceedingss of the 19th IEEE International Performance, Computing and Communications Conference (IPCCC), February 2000, page 30-37.

12) Wierman A. and Harchol-Balter M. *Classifying scheduling policies with respect to higher moments of conditional response time*. In SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pages 229–240, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-022-1.

13) Bill Devlin L., Jim Gray and Spix G. *Scalability terminology, farms, clones, partitions, and packs, racs and raps*. Technical Report MS-TR-99-85, Microsoft Research, 1999.

14) Broberg J., Tari Z., and Zeephongsekul P. Task assignment based on prioritising traffic flows. In Higashino T., editor, Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS 2004). Springer-Verlag, 2005. ISBN 3-540-27324-7.

15) Broberg J., Tari Z., and Zeephongsekul P.. Task assignment with work-conserving migration. Journal of Parallel Computing (Special Issue on Performance Evaluation of Networks for Parallel, Cluster and Grid Computing Systems), 2006. To appear.

16) Tanenbaum A. *Modern Operating Systems*. Prentice Hall, 1995.

17) Varga A. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'01)*, 2001.

18) Bienia C. Benchmarking modern multiprocessors. Princeton University, 2011.145.

19) Wolff R. W. Stochastic Modeling and the Theory of Queues. Prentice-Hall, 1989.

20) Stallings W. High-Speed Networks and Internets: Performance and Quality of Service. Prentice-Hall, 2002.

21) Sozaki S. and Ross R. Approximations in finite capacity multiserver queues with Poisson arrivals. *Journal of Applied Probability*, 1978, pages 826–834.

22) Anuradha, Dande. Simulation of Multiprocessor System Scheduling. MS thesis. 2014, URL: https://trepo.tuni.fi/handle/123456789/22144 (дата звернення: 18.05.2021).

23) Silberschatz A. and Galvin P. *Operating System Concepts (Fifth Edition)*. Addison-Wesley Publishing Company, 1998.

24) Schrage L.. A proof of the optimality of the shortest remaining service time discipline. *Operations Research*, pages 670–690, 1968.

25) Ross S. M. *Simulation*. Academic Press, 2001.

26) Ross S. M. *Introduction to Probability Models*. Academic Press, 2002.

27) Pai V. S., Druschel P., and Zwaenepoel W. Flash: An efficient and portable Web server. In Proceedings of the USENIX 1999 Annual Technical Conference, 1999.

28) Nelson R. D. and Philips T. K. An approximation for the mean response time for shortest queue routing with general interarrival and service times. *Performance Evaluation Review* , 1993, pages 123–139.

29) Mitzenmacher M. How useful is old information? *IEEE Transactions on Parallel and Distributed Systems*, 2000, pages 6–20.

30) Milojicic D. S., Douglis F., Paindaveine Y. Process migration. ACM Computing Surveys, 2000, pages 241–299.

31) Kleinrock L. *Queueing Systems Volume 1: Theory.* John Wiley and Sons., 1975.

32) Kleinrock L. Queueing Systems Volume 2: Computer Applications. John Wiley and Sons., 1975.

33) Kendall D. Stochastic processes occurring in the theory of queues and their analysis by the method of the embedded markov chain. *Annals of Mathematical Statistics*, pages 338–354, 1953.

34) Harchol-Balter M., Crovella M. E., and Murta C. D. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing*, 1999, pages 204–228.

35) Gross D. and Harris C. M. *Fundamemtals of Queueing Theory*. Wiley-Interscience, 1998.

36) Downey A. B. Lognormal and Pareto distributions in the internet. *Computer Communica-tions*, 2005, pages 790–801.

37) Ferrari D. A study of load indices for load balancing schemes. Technical Report CSD-85-262, UC Berkeley, 1985.

38) Crovella M. E., Taqqu M. S., and Bestavros A. *Heavy-tailed probability distributions in theWorld Wide Web*. Birkhauser Boston Inc., Cambridge, MA, USA, 1998b. ISBN 0-8176- 3951-9.

39) Conway R. W., Maxwell W. L., and Miller L. W. *Theory of Scheduling*. Courier Dover Publications, 1967. ISBN 0486428176.

40) Cobham A. Priority assignment in waiting line problems. *Journal of the Operations Research Society*, 1953, pages 70–76.

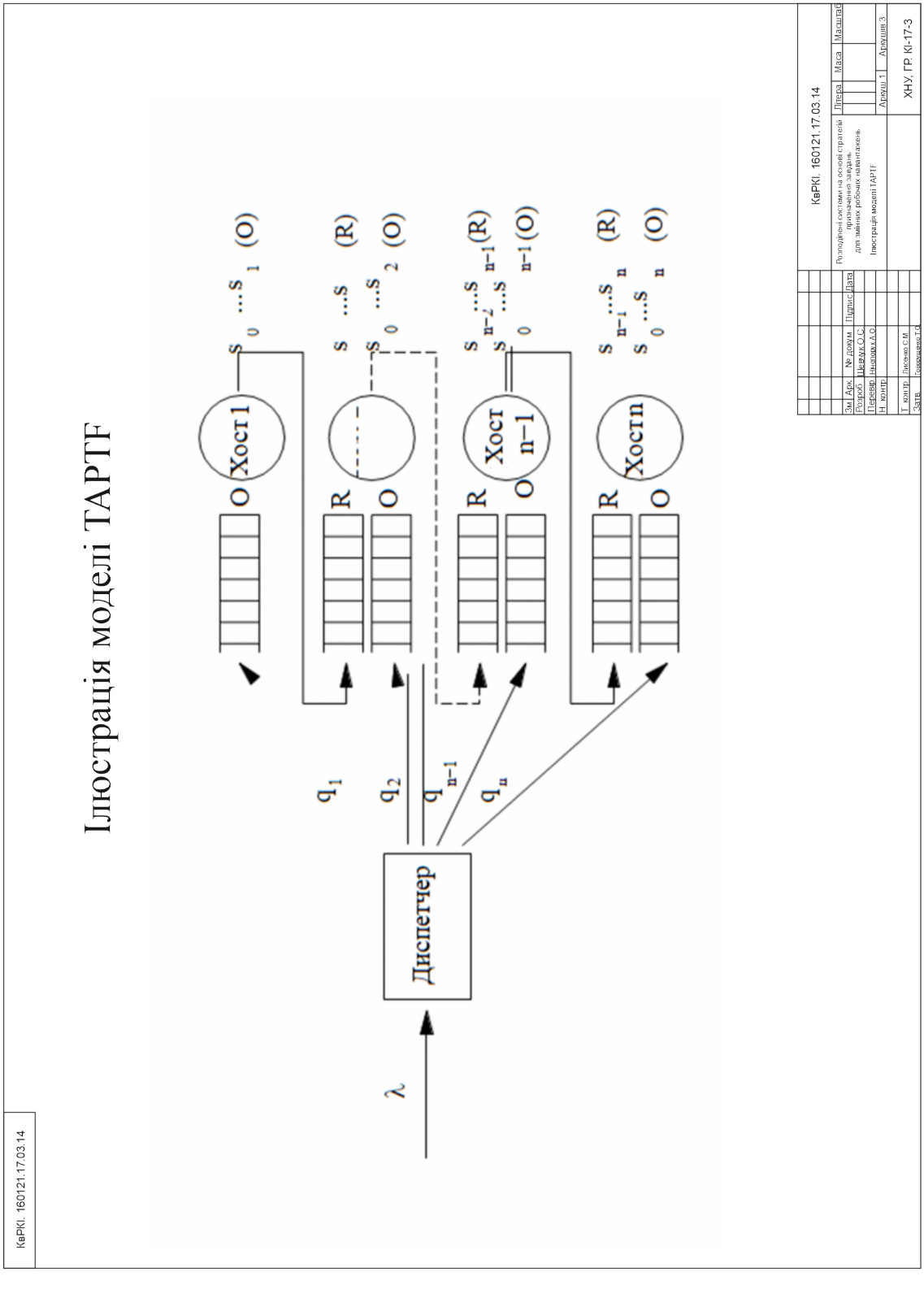
41) Cisco Systems. Scaling the internet web servers, 1997. URL http://www.cisco.com/warp/ public/cc/pd/cxsr/400/tech/scale wp.pdf. (дата звернення: 15.05.2021).

42) Cabrera L.-F. The influence of workload on load balancing strategies. In *Proceedings of theWinter USENIX Conference*, 1986, pages 446–458.

43) Brockwell P. J. and Davis R. A. *Time series: theory and methods*. Springer-Verlag New York, Inc., New York, NY, USA, 1986. ISBN 0-387-96406-1.

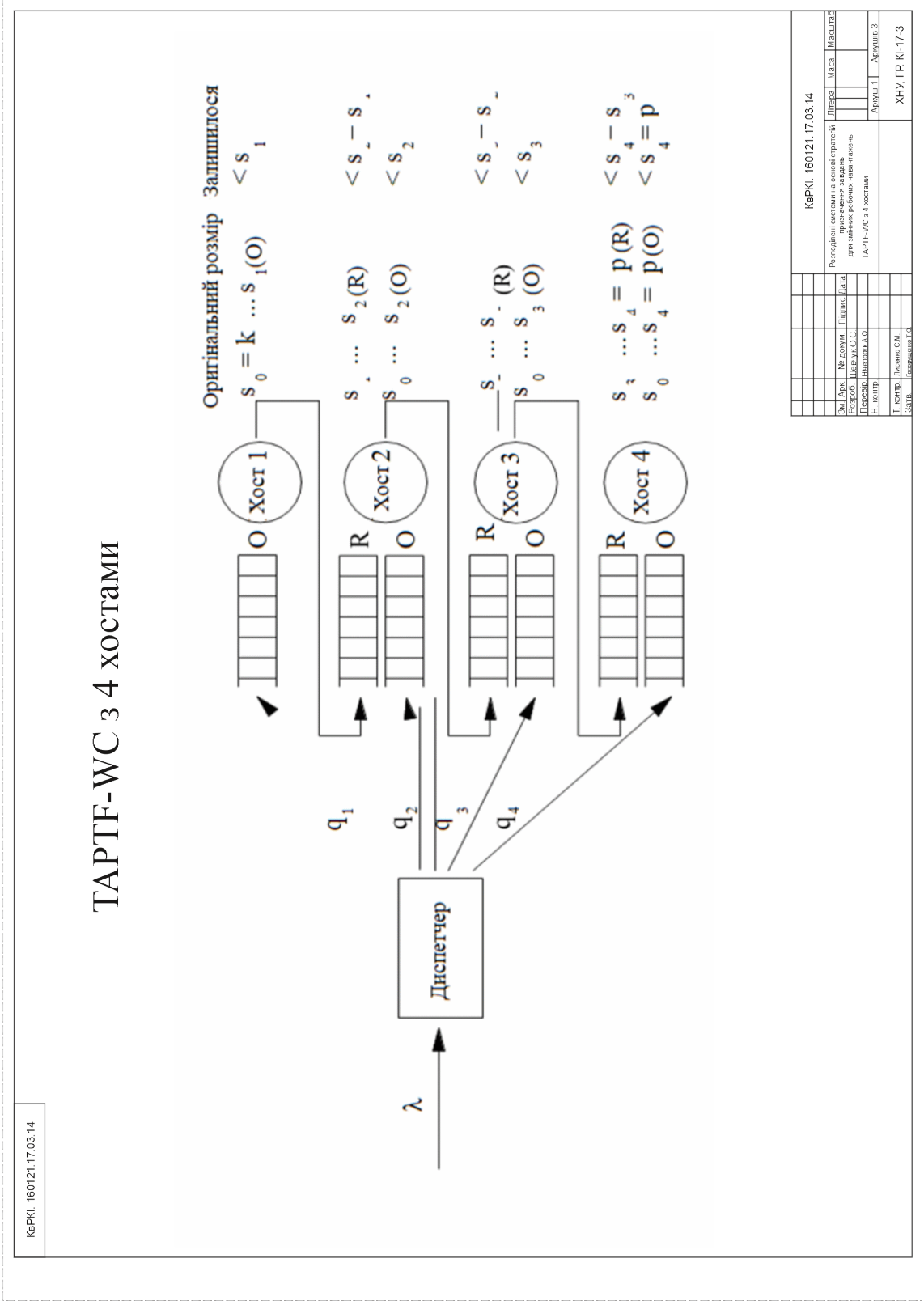
Додаток А

Копія креслення «Модель TAPTF»



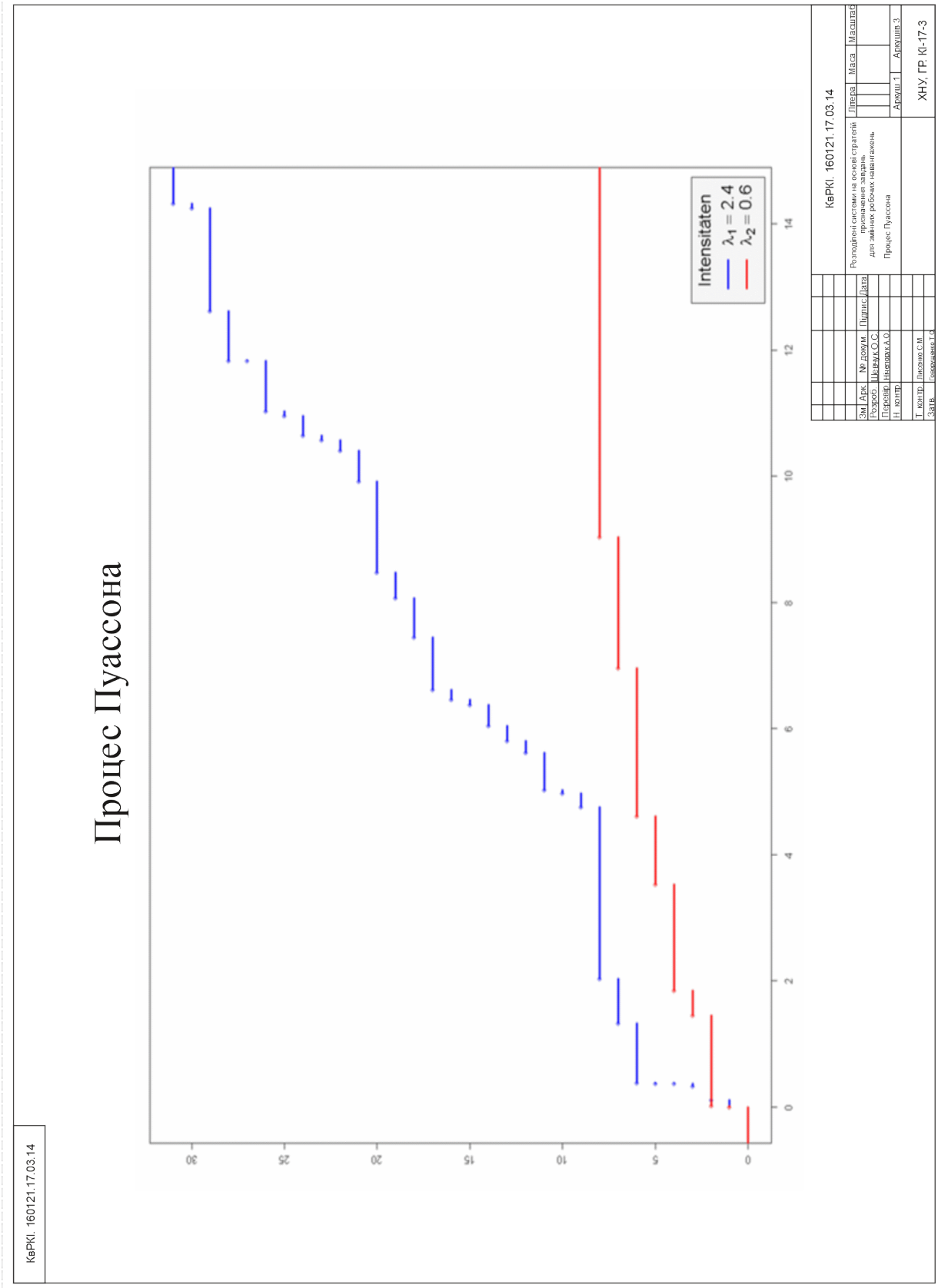
Додаток Б

Копія креслення «Модель TAPTF-WC з 4 хостами»



Додаток В

Копія креслення «Процес Пуассона»



Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 3.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 11%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 94913 Название: Розподілені системи на основі стратегій призначення завдань для змінних робочих навантажень Добавлено в БД: 2021-06-21 Авторы: Шевчук О.О. Руководители: Нічепорук А.О. Консультанты: Оппоненты:	Символы	Лексемы	Символы	Лексемы
	94559	710	2813 (3%)	28 (4%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1008337216

Дата перевірки:
21.06.2021 08:22:19 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.06.2021 08:23:20 EEST

ID користувача:
100005591

Назва документа: **Шевчук_Розподілені системи на основі стратегій призначення завдань для змінних робоч...**

Кількість сторінок: **60** Кількість слів: **15010** Кількість символів: **113292** Розмір файлу: **919.00 KB** ID файлу: **1008408033**

9%
Схожість

Найбільша схожість: **7.51%** з джерелом з Бібліотеки (ID файлу: **1008214695**)

0.14% Джерела з Інтернету	34	Сторінка 62
8.86% Джерела з Бібліотеки	77	Сторінка 62

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	31

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник Шевчук Олег Сергійович
Тема Розподілені системи на основі стратегій призначення завдань для змінних робочих навантажень
Спеціальність 123 Комп'ютерна інженерія

Обсяг кваліфікаційної роботи:

кількість листів креслень 3 ; кількість сторінок записки 76

1. Короткий зміст КП та прийнятих рішень В рамках кваліфікаційної роботи було розроблено нова політика призначення завдань для пакетних та супер-обчислювальних кластерів, яка називається "Task Assignment based on Prioritising Traffic Flows" (TAPTF), забезпечуючи гнучку та високоефективну політику призначення завдань, і зменшив деякі накладні витрати на обробку, що було проблематично при існуючій техніці, TAGS.

2. Висновок про відповідність КП дипломному завданню Кваліфікаційна робота у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому, теоретичному, розділі кваліфікаційної роботи якісно та в повній мірі розглянуті методи вирішення поставленої задачі, був проаналізований кожен аспект, який стосується теми кваліфікаційної роботи. У наступному розділі було проведено порівняльну характеристику модулів, що можуть використовуватись в системі. Рішення про використовувані модулі обґрунтовано. У основні частинні роботи було представлено новий підхід до призначення завдань у розподіленій системі, TAPTF. В загальному усі розділи відповідають завданню та містять сучасні методи вирішення поставлених завдань.

4. Позитивні сторони проекту Кваліфікаційна робота відповідає сучасним вимогам до проектування мультипроцесорних систем. Для проектування системи були вукористані сучасні програмно-апаратні рішення. Окремо можна виділити підняття питання про «вузьке місце» таких систем, та спроектовано і протестовано рішення для усунення такого недоліку.

5. Негативні сторони проекту Надмірна деталізація в плані питання вибору процесора. Добре було б детальніше розглянути питання масштабованості та організації системи. Вказаний недолік не зменшує позитивне враження від роботи.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконане відповідно до суті кваліфікаційної роботи. У першому листі відображено загальну схему моделі TAPTF У другому листі детально показано модель TAPTF-WC з 4 хостами». Третій лист представляє собою схему Процесу Пуассона.

7. Відгук про роботу в цілому В загальному кваліфікаційна робота заслуговує схвальних відгуків. Весь матеріал роботи структурований, чіткий та послідовний. Усі розділи кваліфікаційної роботи йдуть у вірній послідовності, що дозволяє чітко розуміти викладений матеріал в рамках даної роботи. Графічний матеріал дозволяє наочно побачити принцип побудови, та методи покращення розподілених систем.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Розглянувши позитивні та негативні сторони представленого дипломного проекту, можна зробити висновок, що він заслуговує оцінку «задовільно»

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) доцент кафедри автоматизації, комп'ютерно-інтегрованих технологій та телекомунікацій, к.т.н. Чешун Віктор Миколайович

« 23 » червня 2021 р. _____ (підпис)

ХНУ. Зам. 54, тир. 6000, 2007.

Завідувачу кафедри КІСП
д-р.техн.наук, проф. Говорущенко Т. О.

Шевчук Олег Сергійович
ПІБ здобувача вищої освіти

ФПКТС, 4 курсу, групи КІ-17-3

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 (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) 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.

23.06.2021
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

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

Назва: Розподілені системи на основі стратегій призначення завдань для змінних робочих навантажень

Автор: Шевчук Олег Сергійович

Спеціальність: 123 – Компютерна інженерія та програмування

Освітня програма: освітньо-професійна

Науковий керівник: Нічепорук А.О., к.т.н, 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.	

Підтвердження:

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

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з більш ніж 10 джерелами на один фрагмент речення;
4) серед запозичень знаходяться загальновідомі терміни, скорочення та визначення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає _9%_ і адресується до _111_ першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ А.О. Нічепорук

Гарант ОП _____ С.М. Лисенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко