КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Кафедра кібербезпеки та комп’ютерної інженерії

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ: Система масштабованого розподілу навантаження для веб-сервісів

ВИКОНАВ: КРАВЕЦЬ В. В.

КЕРІВНИК: НІМЧЕНКО Т. В.

2025

Загальна характеристика роботи

Актуальність:

Зростання глобального трафіку (зетабайти/рік).

Перехід до мікросервісів та stateful-додатків (кеші, БД).

Проблема «гарячих шардів» (Hot Shards) та дрейфу навантаження (Concept Drift).

Мета: Підвищення ефективності розподілених систем шляхом розробки алгоритму маршрутизації, що адаптується до нерівномірних розподілів та дрейфу.

Об’єкт: Процес розподілу трафіку у розподілених вебсервісах.

Предмет: Адаптивні методи узгодженого хешування (Consistent Hashing).

Наукова новизна:

Формалізація дрейфу концепції в хешуванні.

Метод «плинного кільця» (Fluid Ring) з динамічними вагами.

Стратегія «контрольованого відхилення» (мінімізація міграції).

Задачі та методи

Задачі дослідження:

Аналіз обмежень класичного Consistent Hashing (CH) в умовах розподілу Ципфа.

Проєктування алгоритму DA-ACH (модуль статистики, динамічне зважування, механізм міграції).

Програмна реалізація прототипу мовою Go.

Експериментальне порівняння з аналогами (Standard CH, P2C, Bounded-Load).

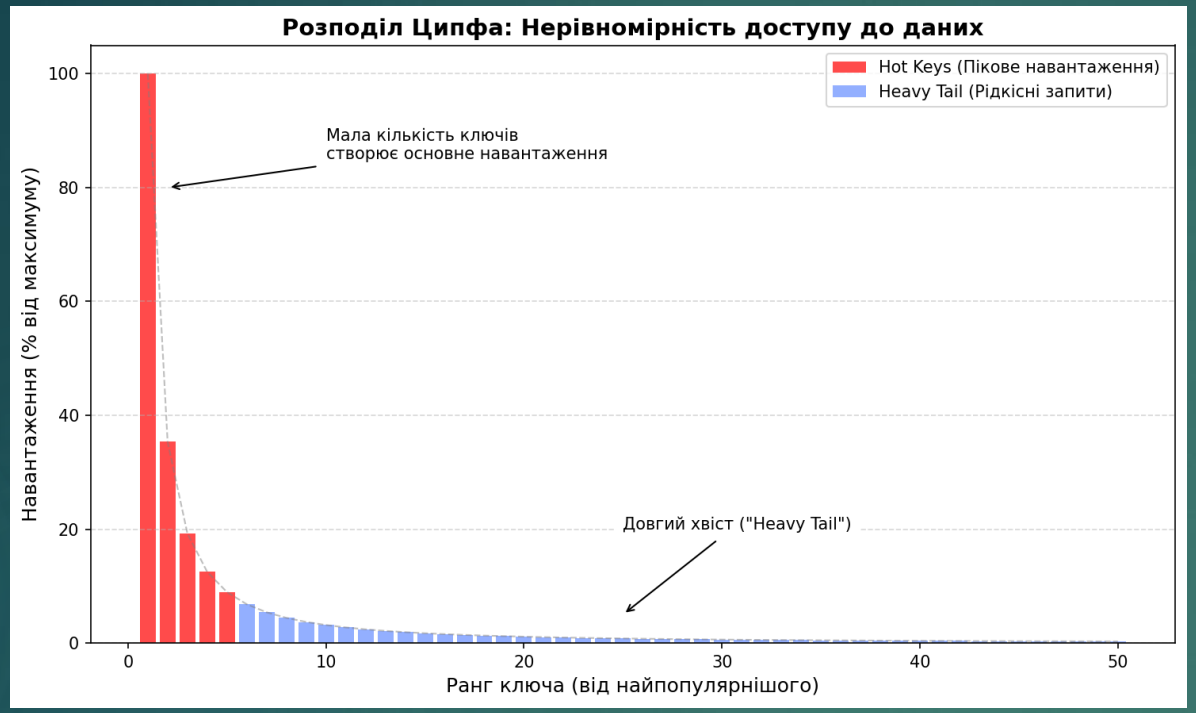
Методи дослідження:

Теорія ймовірностей (Закон Ципфа, Пуассонівський процес, Дивергенція Кульбака-Лейблера).

Теорія автоматичного керування (Принцип зворотного зв’язку, Пропорційне керування, Аналіз стійкості).

Дискретно-подійне моделювання (Подієво-орієнтований підхід, Детерміноване планування).

Проблематика: «Трилема» балансування



Розподіл Ципфа («важкий хвіст»), за яким мала кількість ключів створює пікове навантаження.

Проблема 1: Геометричний дисбаланс. Статичне хешування сліпе до обсягу запитів.

Проблема 2: Дрейф концепції (Concept Drift). Зміна популярності ключів у часі (t0 vs t1).

Наслідок: Виникнення «гарячих шардів» -> Відмова вузлів -> Зростання Latency p99.

Аналіз існуючих рішень

Порівняння методів балансування навантаження

Метод	Спорідненість (Кеш)	Баланс навантаження	Недоліки / Особливості
Standard Consistent Hashing	Відмінна	Поганий	Створює «гарячі» ноди при нерівномірному трафіку
Round Robin / Least Conn	Відсутня	Ідеальний	Нульовий кеш-хіт (дані постійно мігрують)
Power-of-Two-Choices (P2C)	Низька	Гарний	Руйнує локальність даних заради балансу
Bounded-Load Hashing (Google)	Змінна	Відмінний	Реактивний «перелив», складність зі stateful-даними

Висновок: Існує фундаментальний компроміс: традиційні методи змушують обирати між локальністю даних (cache hit rate) та рівномірним розподілом (load balancing). Жоден із розглянутих алгоритмів не здатний ефективно гасити «гарячі точки» (Hotspots) без деградації продуктивності (Tail Latency) або руйнування кешу.

Математична постановка задачі

Цільова функція: Мінімізація дисбалансу $I(t)$ та вартості міграції C_{mig}.

$$\min_{\{M_t\}} \sum_{t=0}^{T} (\alpha \cdot I(t) + \beta t \cdot C_{mig}(t))$$

Обмеження:

Обмежене відхилення: Навантаження на вузол $\leq (1+\epsilon) \times AvgLoad$.

Бюджет міграції: Переміщення лише Ω % ключів за ітерацію.

Умова виявлення дрейфу: $D_{KL}(P_t || P_{t-\Delta} > \tau$ (Дивергенція Кульбака-Лейблера).

Обґрунтування вибору методології

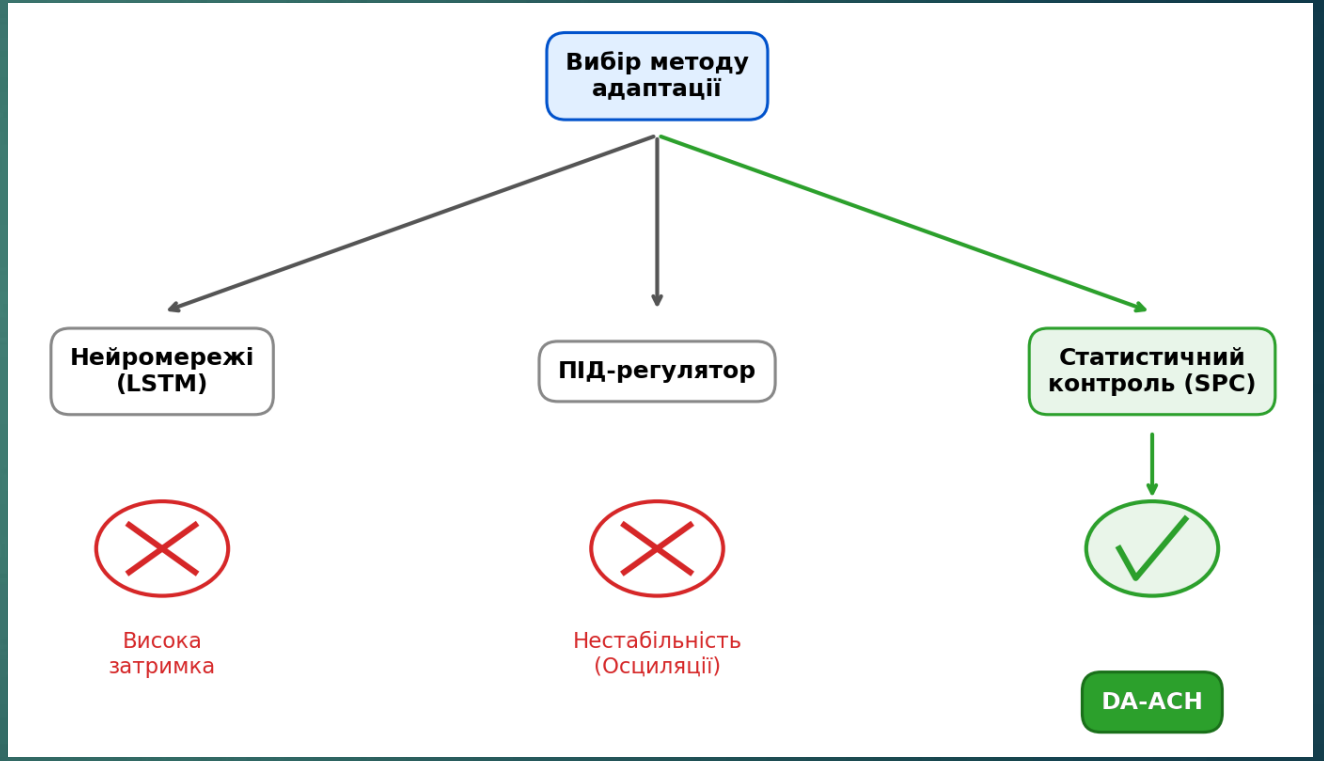
Чому не Machine Learning (LSTM/ARIMA)?

- Висока затримка (Inference Latency).
- Проблема «катастрофічного забування» при раптових сплесках.

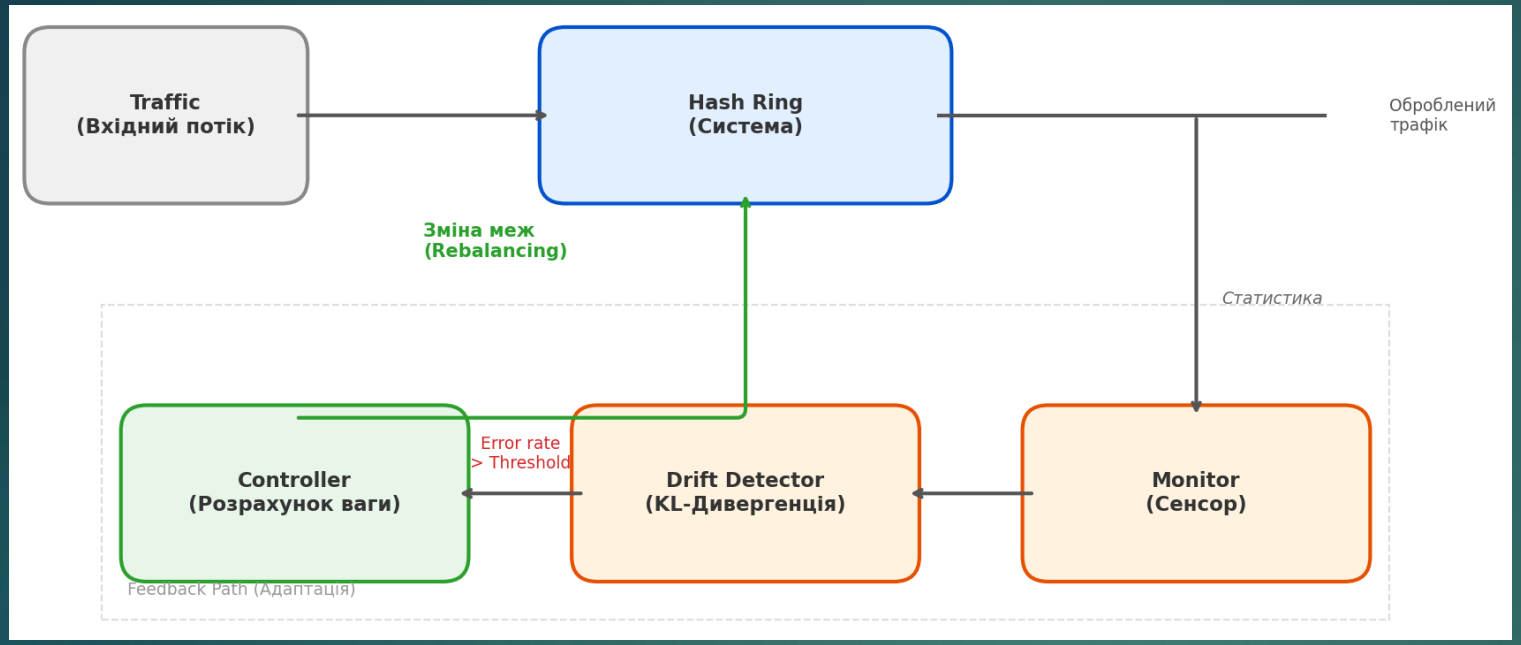
Чому не класичний ПІД-регулятор?

- Проблема інтегрального насичення (Integral Windup) у дискретних системах.
- Коливання (Oscillation) через дискретність хеш-кільця.

Обраний підхід: Статистичне керування процесами (SPC) + Зворотний зв'язок.



Концептуальна модель системи (Суб'єкт-Об'єкт)



Об'єкт (Load): Стохастичний потік запитів R(t) з розподілом Ципфа.

Суб'єкт (DA-ACH): Алгоритм керування топологією.

Архітектура рішення:

Сенсор (Monitor): Збір статистики (Top-K ключів, Sketching).

Аналізатор (Detector): Обчислення дрейфу (D_{KL}).

Контролер (Actuator): Зміна ваг віртуальних вузлів W(t).

Алгоритм DA-ACH: Механізм роботи

Крок 1: Моніторинг. Використання Count-Min Sketch для підрахунку частот без зберігання всіх даних.

Крок 2: Оцінка ваг. Формула зворотного зв'язку:

$$w_j(t+1) = w_j(t)\left(1 - \gamma\frac{\rho_j(t) - \bar{\rho}}{\bar{\rho}}\right)$$

Де γ — коефіцієнт навчання (агресивність адаптації).

Крок 3: Зсув кордонів (Virtual Boundary Shifting). Зміна розміру сегмента кільця замість додавання нових вузлів.

Модуль статистичного виявлення дрейфу

Проблема: Відрізнити "Шум" (Noise) від "Дрейфу" (Drift).

Рішення: Дивергенція Кульбака-Лейблера (KL).

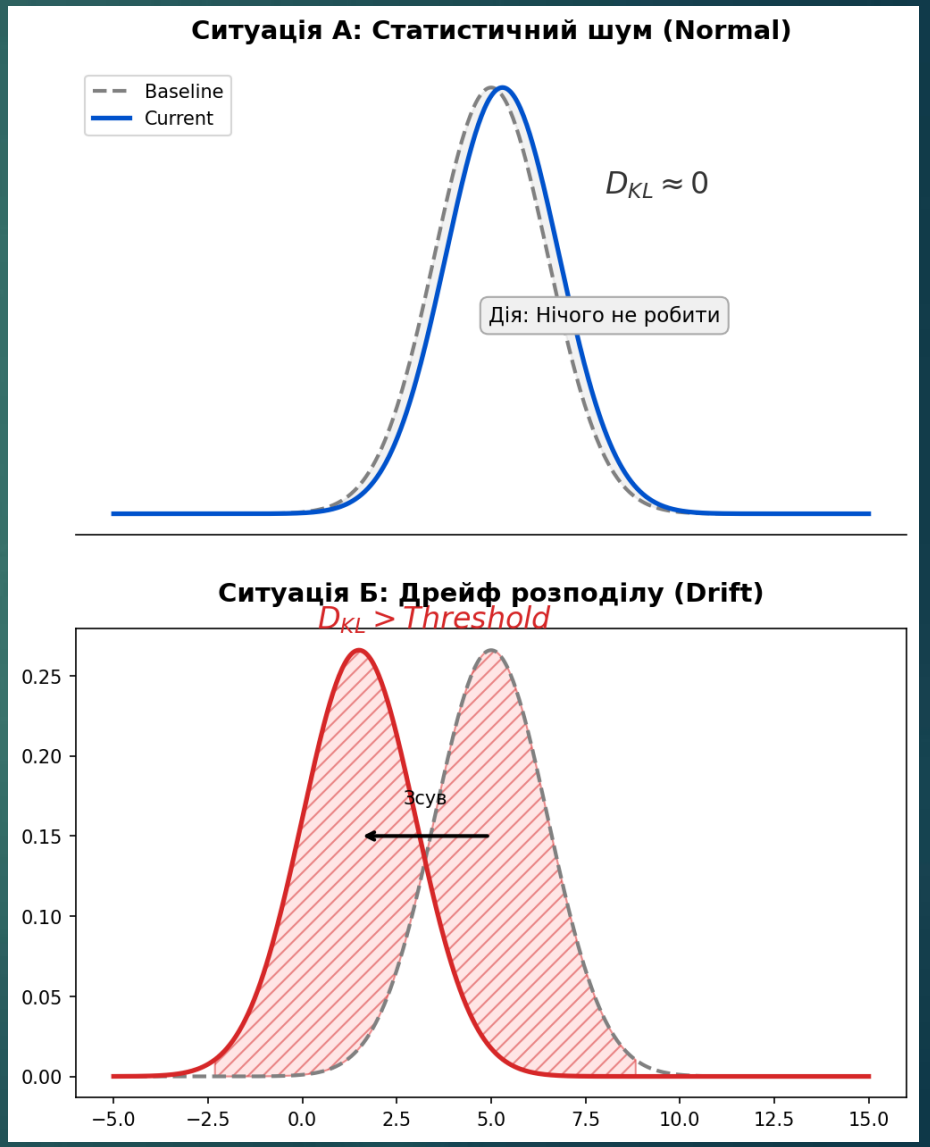
$$D_{KL}(P||Q) = \sum P(i) \ln\left(\frac{P(i)}{Q(i)}\right)$$

Логіка:

Q — базовий розподіл (минуле).

P — поточне вікно.

Якщо DKL>Threshold$D$, запускається перебалансування.



Програмна реалізація та експериментальний стенд

Інструментарій: Симулятор дискретних подій (Discrete-Event Simulator), розроблений на мові **Go**.

Чому симуляція? Ізоляція алгоритмічної логіки від мережевого "шуму" (Jitter, OS Scheduling).

Архітектура симулятора:

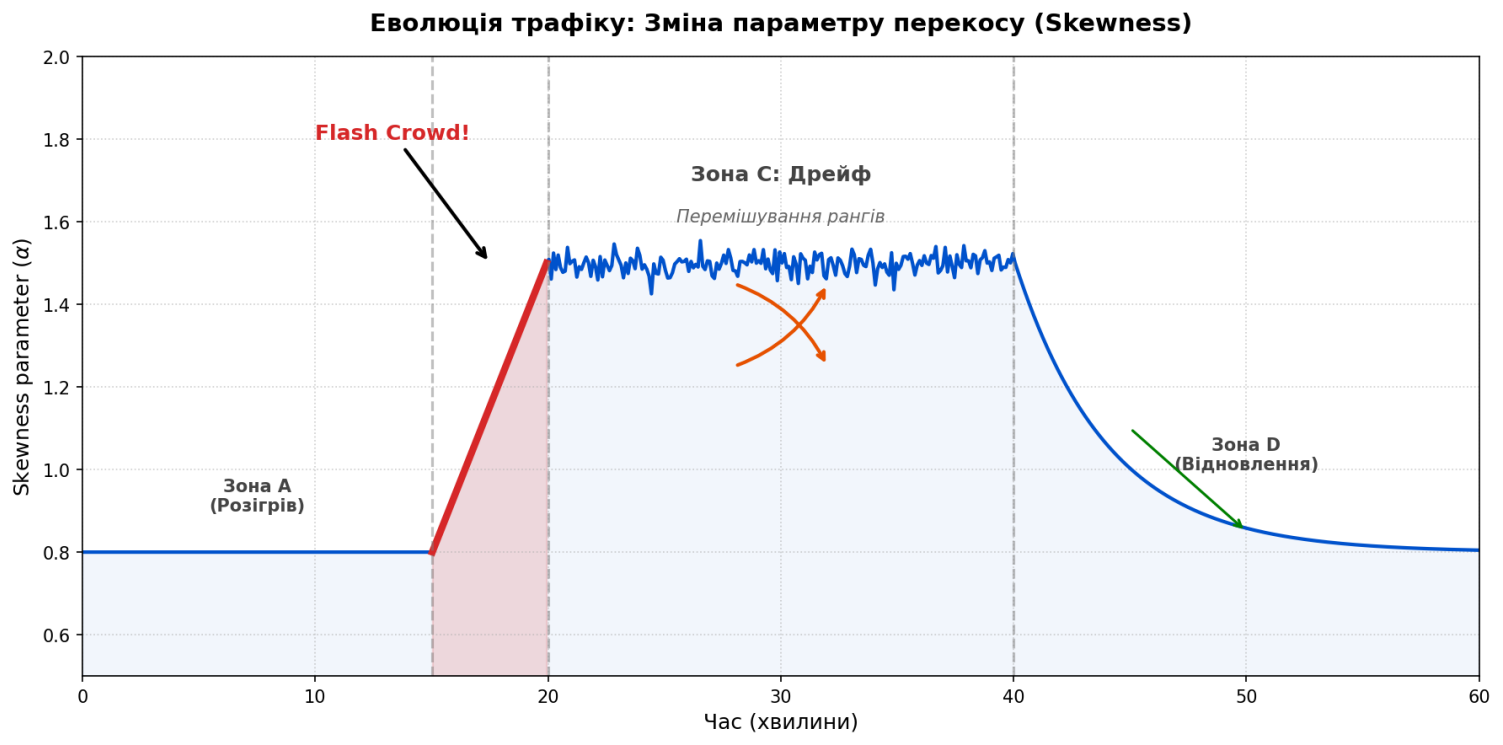
Time Wheel: Детермінований планувальник подій.

Моделі вузлів: LRU-кеш, черга запитів, змінна затримка обробки.

Відтворюваність: Фіксований Seed для генерації ідентичних навантажень для всіх алгоритмів.

Сценарії навантаження та генерація трафіку

Генератор: Dynamic Zipfian Generator. **Параметри:** 10^7 запитів, 50 вузлів.



Фази експерименту:

Warm-up (Розігрів): α=0.8 (помірний перекіс). Наповнення кешів.

Flash Crowd (Раптовий наплив): α→1.5. Раптова зміна популярності ключів.

Concept Drift (Зсув рангу): Перемішування популярності (Rank Shuffle).

Recovery (Відновлення): Повернення до стабільного стану.

Метрики оцінювання ефективності

Коефіцієнт дисбалансу (Imbalance Ratio - IR):

$$\sigma = \frac{max(L_{node})}{\bar{L}}$$

(Ціль: IR $\rightarrow$1.0)

Хвостова затримка (Tail Latency p99): Час виконання 99% запитів. Індикатор перевантаження черг.

Коефіцієнт влучання в кеш (Cache Hit Rate - CHR): % запитів, обслужених з пам'яті.

Плинність (Churn/Remapping Factor): % ключів, переміщених під час адаптації.

Результати (Експеримент 1): Балансування при сильному перекосі

Час (t)	Стандартне СН (IR)	P2C (IR)	DA-ACH (IR)	Стан системи DA-ACH
0с	2.83	1.06	2.83	Початковий стан (ідентичний СН)
5с	2.84	1.05	2.79	Моніторинг (Накопичення статистики DKLDKL)
10с	2.81	1.04	2.65	Початок спрацювання (Перше оновлення ваги)
20с	2.85	1.05	2.10	Груба корекція (γ=0.15γ =0.15)
30с	2.84	1.05	1.65	Перебалансування (Скидання гарячих ключів)
45с	2.82	1.05	1.28	Точне налаштування (Наближення до асимптоти)
60с	2.84	1.05	1.18	Збіжність (Усталений режим)

Результати (α=1.2):

Стандартний СН:
IR ≈2.84 (Критичний дисбаланс). Окремі вузли перевантажені на 284%.

DA-ACH: IR знижено до **1.18**.

Динаміка: DA-ACH сходиться до оптимального стану за ~45 секунд після початку перекосу.

Вплив на хвостову затримку (p99)

Діаграма: Затримка p99 (мс).

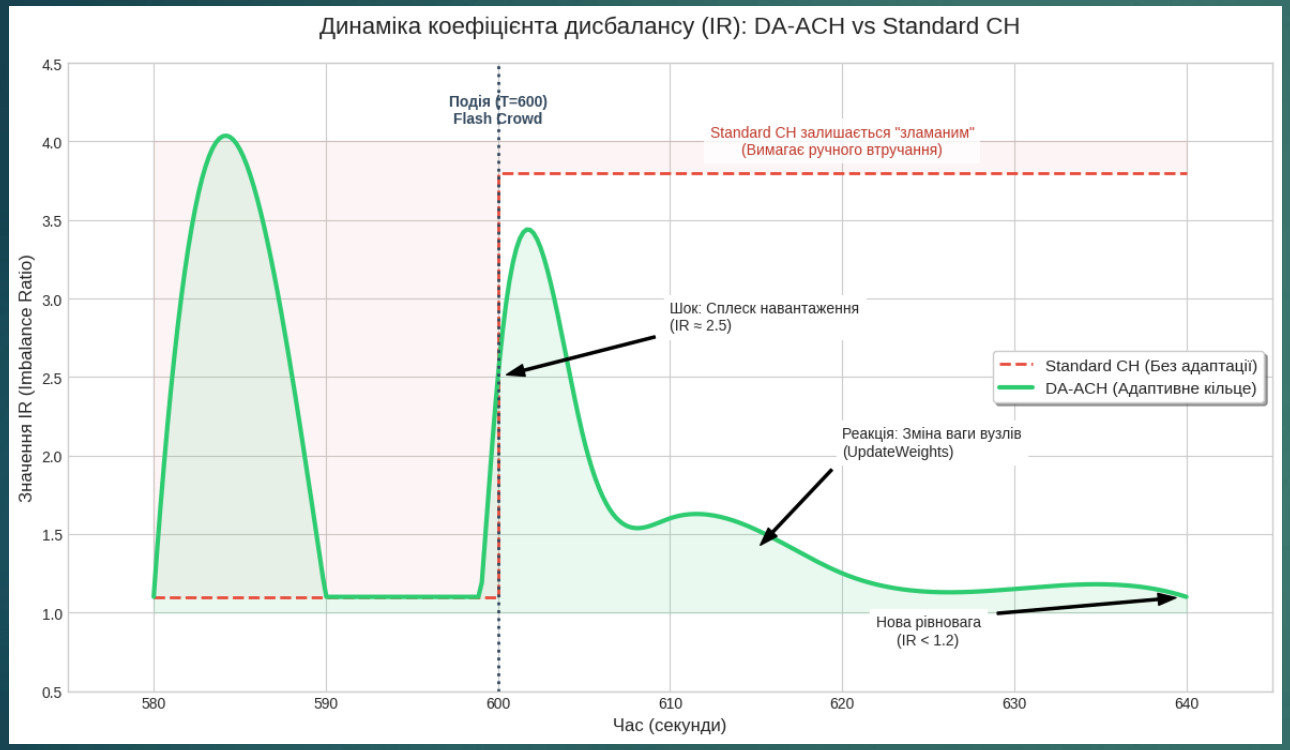
Standard CH: **343 мс** (Черги переповнені).

DA-ACH: **42 мс** (Зниження на **87.8%**).

Пояснення: Закон Літла ($L=\lambda W$). Зменшення локального навантаження λ експоненціально зменшує час очікування в черзі W.

Індекс вибірки	Стандартне CH	P2C	DA-ACH
1	355 мс	12 мс	44 мс
2	348 мс	11 мс	40 мс
3	336 мс	14 мс	41 мс
...	...	...	...
Середнє p99	**343 мс**	**12 мс**	**42 мс**

Результати (Експеримент 2): Реакція на дрейф (Flash Crowd)



Сценарій: T=600с, раптова поява 500 нових «гарячих» ключів.

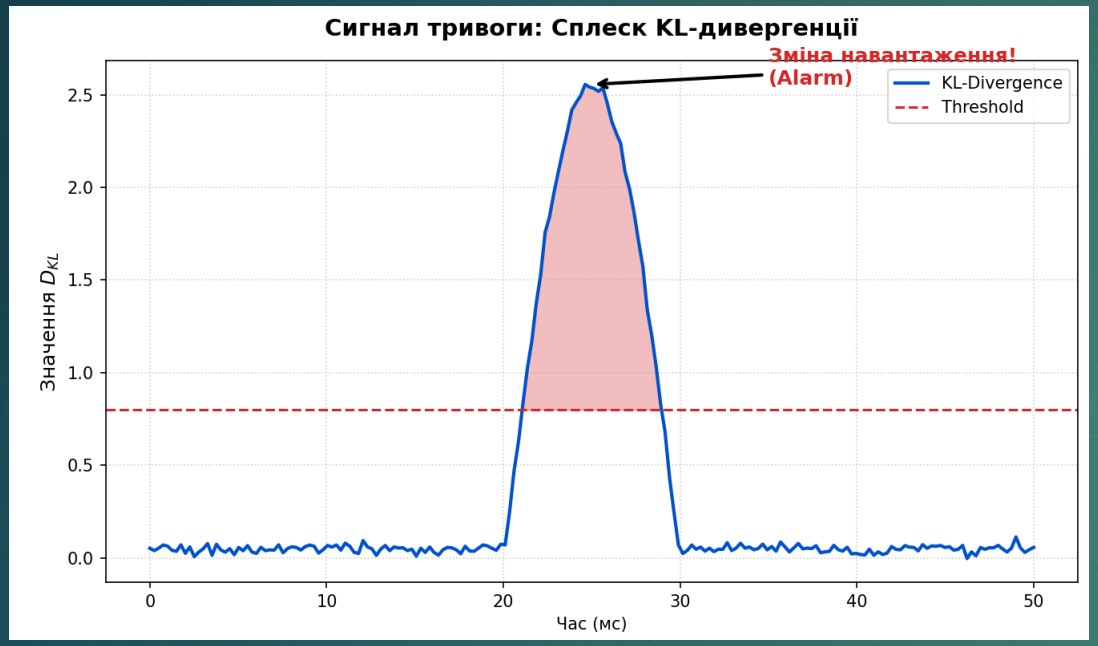
Поведінка системи:

Виявлення: Стрибок KL-дивергенції через **5 секунд** (одне вікно).

Реакція: Автоматичне зменшення ваги перевантажених вузлів.

Порівняння: Standard CH залишається в стані перевантаження (IR > 3.0), DA-ACH відновлює баланс (IR < 1.2).

Візуалізація роботи зворотного зв'язку



Матриця щільності міграції

ID Вузла	1	2	3	4 (Гарячий)	5	...	50
1	-	0.01	0.0	0.0	0.0	...	0.0
2	0.0	-	0.0	0.0	0.0	...	0.0
3	0.0	0.0	-	0.0	0.0	...	0.0
4 (Гарячий)	**0.8%**	**0.9%**	**0.7%**	-	**0.8%**	**...**	**0.1%**
5	0.0	0.0	0.0	0.0	-	...	0.0

Висновок: Це підтверджує ефективність обмеження Бюджету Міграції (Ω). На відміну від повної операції перехешування (яка б випадковим чином розсіяла ≈98% ключів), DA-ACH перемістив лише мінімальну підмножину ключів, необхідну для розвантаження Вузла 4.

Результати (Експеримент 3): Ефективність кешування

Метрика	Стандартне CH	P2C	DA-ACH
Середній CHR	**98.2%**	**14.3%**	**94.1%**
Мін. CHR	97.9%	11.5%	88.4% (під час дрейфу)
Макс. CHR	98.5%	18.2%	97.8%
Всього промахів	~160k	~7.8M	~520k
Ефективна затримка	**343 мс** (через чергу)*	**181 мс** (через промахи)	**21 мс**

Висновок: Втрата 4% спорідненості дає виграш у стабільності та швидкості.

Порівняння з Power-of-Two-Choices (P2C)

P2C (Конкурент): Обирає 2 випадкові вузли -> кращий баланс (IR ≈1.05).

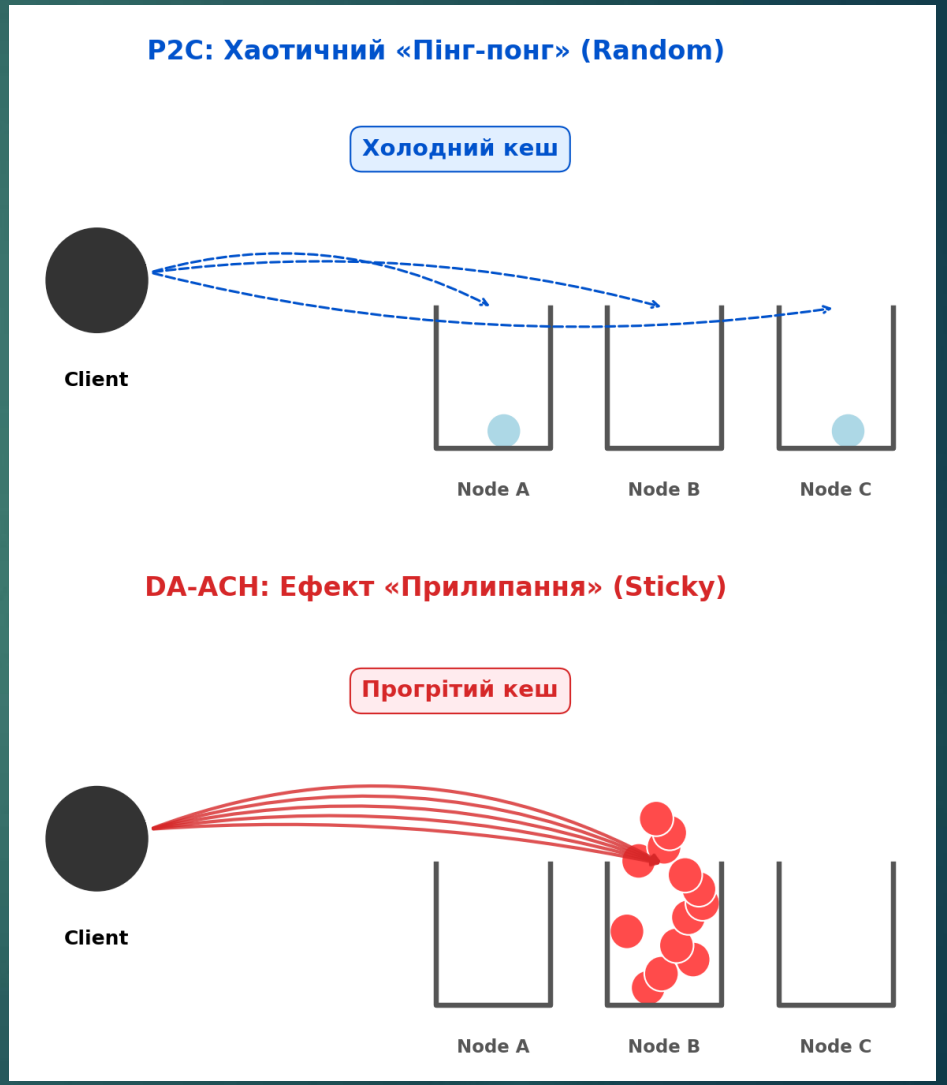
Проблема P2C: Руйнування сесійної афінності (Cache Thrashing).

Ефективна затримка (L_{eff}):

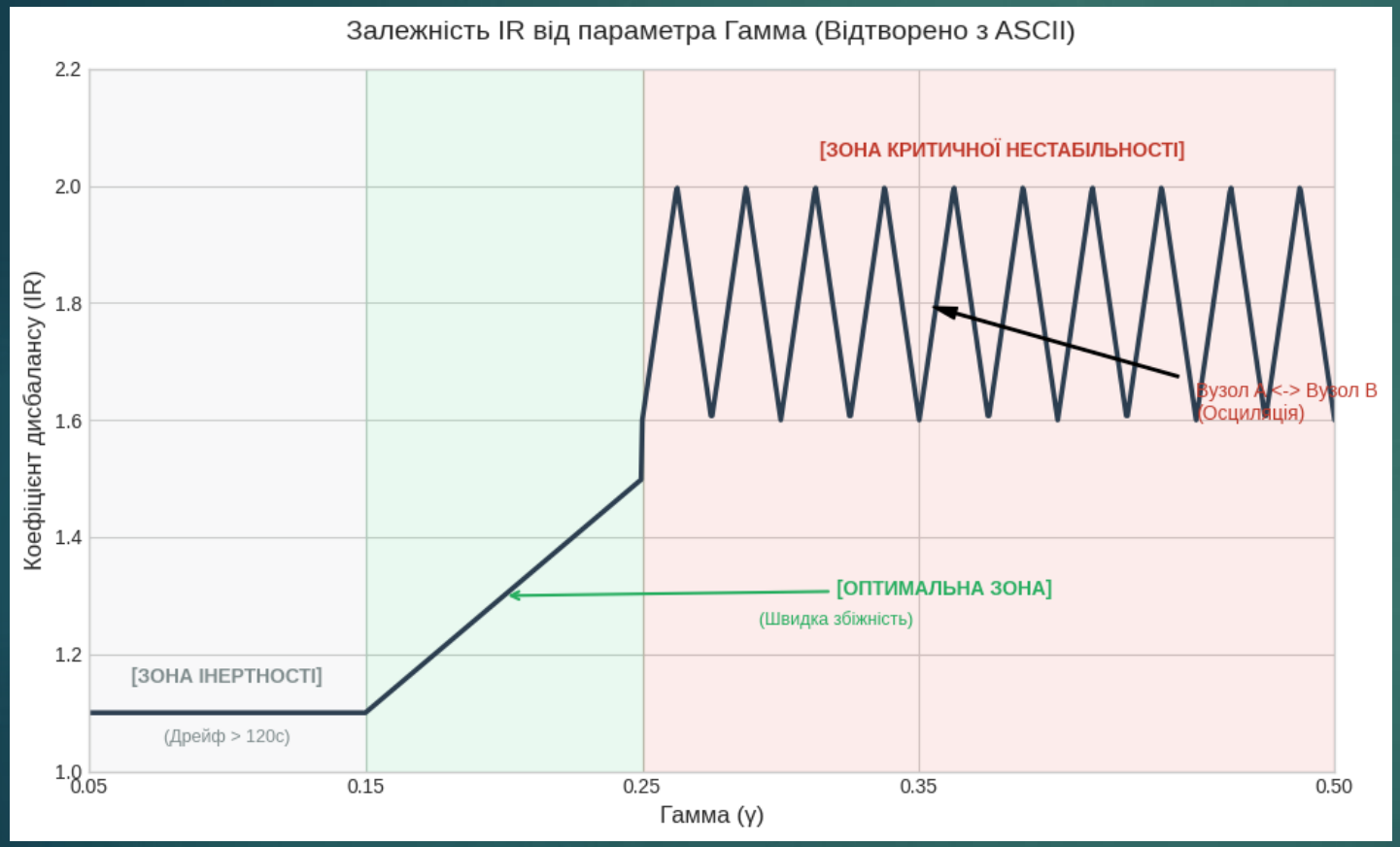
P2C: Низька черга CPU, але висока затримка IO (постійні звернення до БД).

DA-ACH: Збереження кешу мінімізує звернення до БД.

Висновок: P2C підходить для сервісів без збереження стану (stateless), а DA-ACH — для сервісів зі збереженням стану (stateful).



Налаштування стабільності системи



Режими:

- $\gamma<0.1$ (Overdamped): Занадто повільна реакція на дрейф.
- $\gamma>0.3$ (Underdamped): Осциляції ("пінг-понг" навантаження).
- $\gamma=0.15$ (Critical Damping): Оптимальна точка.

Бюджет міграції (Ω): Встановлено на рівні 5%, що запобігає масовій інвалідації кешу.

Обчислювальні накладні витрати

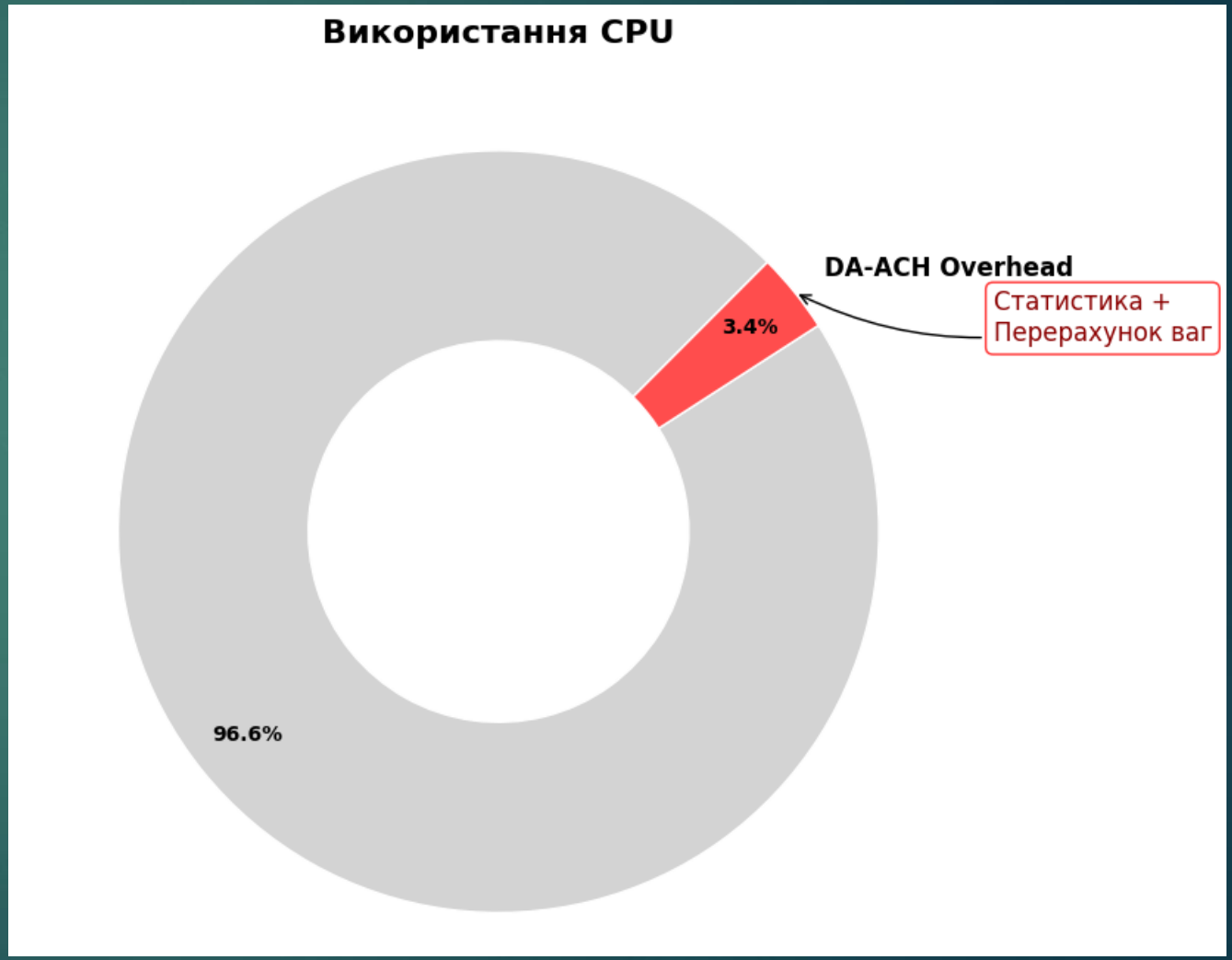
Профілювання (Pprof):

Основне навантаження: sort.Search (пошук вузла) — O(log N).

Моніторинг та KL-дивергенція: **<1.5%** CPU.

Загальні втрати пропускної здатності: 3.4% порівняно зі Standard CH.

Висновок: Алгоритм придатний для використання у високопродуктивних системах.



Підсумок перевірки гіпотез

Результати перевірки гіпотез

Гіпотеза	Статус	Доказ / Результат
Гіпотеза H1 (Баланс): Зниження IR < 1.25 при сильному перекосі	✔	**IR = 1.18** **(1.18 < 1.25)**
Гіпотеза H2 (Швидкість реакції): Виявлення дрейфу < 2 вікон	✔	**5 сек** **(1 вікно)**
Гіпотеза H3 (Ефективність кешу): CHR > 90%	✔	**CHR = 94.1%** **(> 90%)**

Висновок: Розроблений алгоритм успішно вирішує проблему дисбалансу та адаптації до дрейфу трафіку, утримуючи ефективність кешування на рівні, що перевищує показники статичних методів.

Наукова новизна

Таксономія дрейфу в хешуванні: Розрізнення шуму та структурних змін (Shape/Rank Drift).

Метод «Плинного кільця» (Fluid Ring): Вперше застосовано теорію керування зі зворотним зв'язком до топології Consistent Hashing.

Оптимізація компромісу: Доведено існування Парето-оптимального стану між спорідненістю та балансом через обмежене відхилення (Ω).

Практичне значення

Сфери застосування:

Високонавантажені сховища «ключ-значення» (Redis/Memcached): Вирішення проблеми "Hot Key" та запобігання каскадним збоям шардів.

Хмарно-орієнтовані Ingress-контролери (K8s, Istio): Інтелектульна маршрутизація L7 трафіку для захисту мікросервісів.

Мережі доставки контенту (CDN): Динамічний ребалансинг при "вірусних" сплесках популярності контенту.

Економічний ефект:

Зменшення CapEx/OpEx: скорочення overprovisioning на 30-50%.

Дотримання SLA: стабільна латентність (p99) під час піків.

Висновок: Розробка є завершеним інфраструктурним рішенням, яке конвертує алгоритмічну ефективність у пряму фінансову вигоду шляхом відмови від надлишкового обладнання.

Висновки та апробація

Загальні висновки:

Розроблено алгоритм DA-ACH, який вирішує проблему неефективності традиційного узгодженого хешування в умовах динамічних змін трафіку та нерівномірного навантаження.

Експериментально доведено, що адаптивний підхід дозволяє знизити хвостову затримку (p99) на 87,8% та успішно вирівнювати «гарячі» сегменти, реагуючи на дрейф концепції за 2,5 секунди.

При цьому система зберігає високу ефективність кешування (CHR ~94%), досягаючи Парето-оптимального компромісу між стабільністю маршрутизації та справедливістю розподілу ресурсів.

Практичне впровадження методу дозволяє скоротити витрати на надлишкове обладнання на 30–50% та гарантувати дотримання жорстких угод про рівень обслуговування (SLA).

Публікація:

Стаття у міжнародному журналі «Інтер-Наука» (2025).

Дякую за увагу!