УДК 004.421.2 : 004.75

Магістрант Жовтанюк М. В., д-р. ф. Молчанов О. А.

Національний технічний університету України
«Київський політехнічний інститут імені Ігоря Сікорського»

СПОСІБ ПОБУДОВИ РОЗПОДІЛЕНОГО ФІЛЬТРА БЛУМА З КОНСИСТЕНТНИМ ХЕШУВАННЯМ

Abstract

Zhovtaniuk Maksym, master student; Oleksii Molchanov, PhD
A method for constructing a distributed bloom filter with consistent hashing

This work addresses membership testing with Bloom filters when their size exceeds the memory capacity of a single node. The paper proposes a distributed Bloom filter based on consistent hashing that partitions a single bit array across multiple nodes and routes each query to the corresponding shard-level filter. The approach preserves the target false positive rate (FPR) by performing membership checks within a single shard without aggregating responses from multiple nodes. The proposed design enables horizontal scalability and enhances fault tolerance by distributing data and load across nodes, thereby reducing the impact of individual failures on overall availability.

Вступ

Стрімке зростання обсягів даних у сучасних інформаційних системах та платформах обробки даних створює підвищені вимоги до швидких і ефективних щодо використання пам’яті структур даних для перевірки належності елементів до множини. Фільтр Блума [1], як класична ймовірнісна структура, забезпечує сталий час доступу та керований відсоток хибно-позитивних відповідей (FPR). Втім, у випадку великих обсягів даних бітовий масив часто перевищує можливості одного вузла пам’яті, що ускладнює централізоване розміщення без втрати продуктивності та доступності.

Наявні підходи до використання фільтра Блума у розподілених системах мають суттєві обмеження. Монолітне розміщення на одному сервері створює єдину точку відмови та впирається в ліміти оперативної пам’яті. Тому актуальною є побудова розподіленого рішення, яке зберігає цільовий FPR, забезпечує сталий доступ і підтримує горизонтальне масштабування.

У цій роботі запропоновано теоретично обґрунтований спосіб побудови розподіленого фільтра Блума на основі консистентного

хешування. Ключовою ідеєю є розподілення єдиного бітового масиву між вузлами кластера та маршрутизація запитів до відповідного шард-фільтра, що дає змогу зберегти заданий FPR без агрегування відповідей, знизити хвостову затримку доступу та досягти гнучкого горизонтального масштабування системи.

Постановка задачі

Метою дослідження є теоретичне обґрунтування способу побудови розподіленого фільтра Блума на основі консистентного хешування, який зберігає цільовий відсоток хибно-позитивних відповідей (FPR), забезпечує доступ до єдиного шард-фільтра під час перевірки належності та підтримує горизонтальне масштабування системи. Завдання полягає у формалізації моделі розподілення єдиного бітового масиву між вузлами, визначенні параметрів фільтра Блума, аналітичній оцінці впливу кількості вузлів на латентність і пропускну здатність.

Термінологія

Фільтр Блума (Bloom filter). Імовірнісна структура даних для перевірки належності елемента до множини з фіксованою ймовірністю хибнопозитивної відповіді. Складається з бітового масиву розміру m та k хеш-функцій, вставка встановлює k бітів, перевірка тестує ті самі позиції. За класичної моделі фільтра Блума позитивні відповіді перевірки належності множині з певною імовірністю можуть бути хибними, тоді як негативні — завжди коректні.

Хибнопозитивні відповіді, FPR (False Positive Rate). Імовірність того, що фільтр Блума помилково заявить про належність елемента, якого фактично немає в множині.

Горизонтальне масштабування (horizontal scaling). Збільшення пропускної здатності та/або доступного обсягу пам’яті шляхом додавання нових вузлів (scale-out), на відміну від вертикального масштабування (scale-up), де посилюються ресурси окремого вузла.

Консистентне хешування (consistent hashing). Метод відображення ключів на вузли кластера через «кільце» ідентифікаторів із віртуальними вузлами, що забезпечує стабільний розподіл навантаження та обмежує перерозподіл ключів при зміні складу кластера.

Шард-фільтр — це окрема частина єдиного фільтра Блума, закріплена за певним вузлом. Вона містить свій бітовий масив і обслуговує лише ключі, що потрапили до неї за консистентним хешуванням.

Аналіз наявних рішень для вирішення проблеми перевірки належності у розподілених системах

Проблема класичного підходу побудови фільтра Блума полягає в тому, що зі зростанням очікуваної кількості елементів і зменшенням цільового FPR розмір бітового масиву швидко виходить за межі пам’яті одного вузла. Це створює єдину точку відмови, ускладнює оновлення та перебудову фільтра.

На практиці часто використовують базу даних Redis з модулем RedisBloom [2]. Кожен фільтр Блума зберігається як окремий ключ у Redis. Якщо Redis працює в кластерному режимі, дані автоматично розподіляються по так званих ділянках кластера. Для масштабування створюють кілька фільтрів, які розподіляються по різних вузлах, тим самим розподіляючи навантаження.

Альтернативним способом до класичного підходу перевірки належності за допомогою фільтра Блума з збереженням у оперативній пам’яті є використання операцій дискових. При такому підході фільтр представлено як файл, що відображений на пам’ять (mmap) [3]. Це дає змогу зберегти цільові параметри структури даних, але в той ж час збільшує затримку відповіді, що недопустимо в високонавантажених системах реального часу.

Опис запропонованого способу

Для вирішення проблеми горизонтального масштабування фільтра Блума запропоновано спосіб розподілення структури даних за допомогою консистентного хешування. Підхід полягає в створенні кільця ідентифікаторів із вузлами, кожному сегменту кільця відповідає шард-фільтр як частина єдиного логічного бітового масиву. Для перевірки належності ключ хешується й проєктується в точку на кільці, після чого маршрутизується до найближчого за годинниковою стрілкою віртуального вузла. Усі операції вставки та перевірки виконуються в одному шарді, що зберігає цільовий FPR і дає можливість горизонтально розподіляти пам’ять і навантаження між вузлами. Як хеш-функцію використано MurmurHash [4] — швидку некриптографічну функцію з хорошою «лавинною» властивістю, коли мала зміна ключа суттєво змінює хеш, що дає рівномірний розподіл значень, між віртуальними вузлами на кільці.

Результати експериментальних досліджень

Проведено навантажувальне тестування з параметрами: тривалість — 60 секунд, частота — 500 запитів/секунду. Для стандартного підходу побудови фільтра Блума отримано результати p99 — 1.305 мс (рис 1).

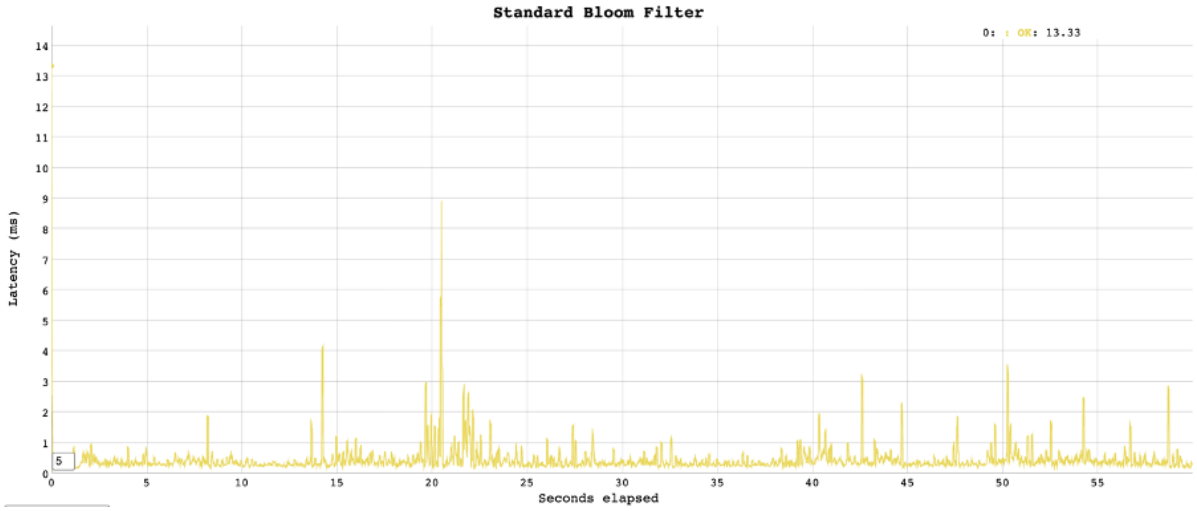

Рис. 1. Результати роботи стандартного фільтра Блума

Тоді як для запропонованого способу розподіленої системи з консистентним хешуванням p99 — 1.787 мс (рис. 2).

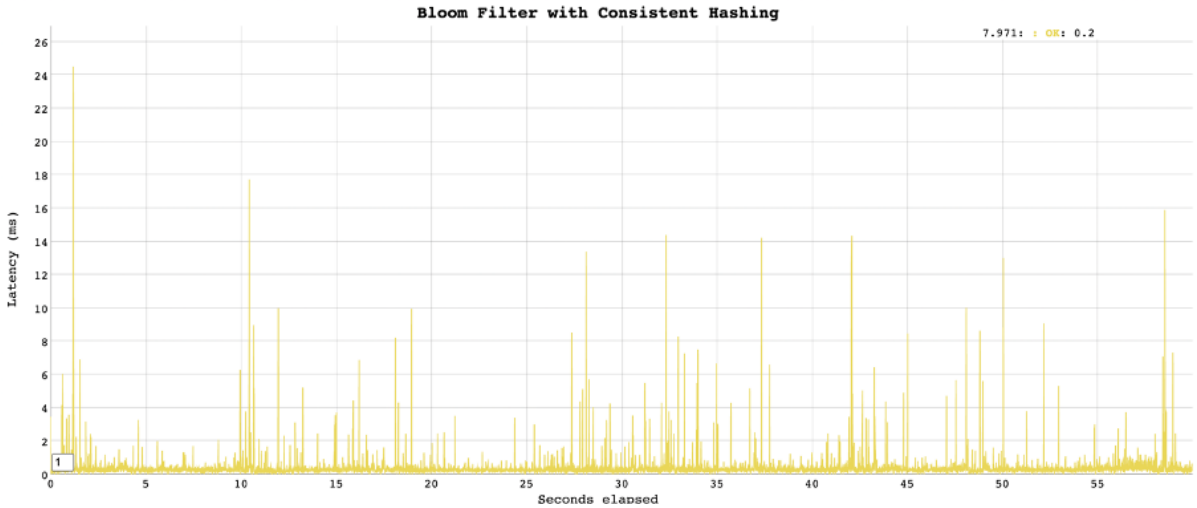

Рис. 2. Результати роботи системи розподіленої системи фільтра Блума

Зростання p99 пов’язане з додатковими операціями обчислення хешу ключа та пошуком потрібного вузла на кільці. Їх виконання додає десятки–сотні мікросекунд, проте залишається некритичним, оскільки MurmurHash3 є високопродуктивним, а адресація виконується один раз на запит. Запропонований спосіб дозволяє розміщувати шард-вузли на різних фізичних серверах, що підвищує відмовостійкість системи завдяки рознесенню навантаження та локалізації збоїв окремих хостів без впливу на цілісність логічного фільтра.

Висновки

У роботі запропоновано спосіб розподілення фільтра Блума за допомогою консистентного хешування з формуванням кільця вузлів. Такий підхід забезпечує доступ до єдиного шард-фільтра для вставки та перевірки, зберігає цільовий FPR, дає змогу горизонтально масштабувати обсяг пам’яті й навантаження, а також розміщувати шард-вузли на різних фізичних серверах для підвищення відмовостійкості. Практичні вимірювання показали невелике зростання p99-латентності через адресацію на кільці, яке є некритичним завдяки високій швидкодії обраної хеш-функції і тому, що адресація виконується один раз на запит.

Надалі доцільно зосередитися на питаннях відмовостійкості: процедурах відновлення даних вузла після збою, алгоритмах перебалансування кільця під час зміни складу кластера, а також додати підтримку додавання та видалення вузлів із гарантованою цілісністю фільтра.

Література

1. Bloom B. H. Space/Time Trade-offs in Hash Coding with Allowable Errors // Communications of the ACM. – 1970. – Vol. 13, № 7. – P. 422–426. DOI: https://doi.org/10.1145/362686.362692
2. Jones A. Redis Unlocked: Advanced Techniques and Strategies for Efficient Data Management. – 2025. – Sec. 3.5. URL: https://www.google.com.ua/books/edition/Redis_Unlocked_Advanced_Techniques_and_S/c-87EQAAQBAJ?hl=en&gbpv=0 (дата звернення: 12.10.2025).
3. mmap(2) — Linux manual page // man7.org. – URL: https://man7.org/linux/man-pages/man2/mmap.2.html (дата звернення: 20.10.2025).
4. SMHasher // GitHub. – URL: https://github.com/aappleby/smhasher (дата звернення: 20.10.2025).