УДК 004.042

Р. С. Клейменов, Т. А. Ліхоузова

ПРОБЛЕМИ КЕШУВАННЯ ДАНИХ ПРИ ВИКОРИСТАННІ МОВИ ПРОГРАМУВАННЯ JAVA

Анотація: Більшість існуючих комп’ютеризованих систем, реалізованих на мові програмування Java, для роботи з інформацією використовують широко розповсюджені SQL бази даних, для взаємодії з якими використовують стандартну реалізацію SQL інтерфейсу й мають велику кількість запитів до малозмінних даних. Це призводить до нераціонального використання інтернет-трафіку, зменшення швидкості доступу до необхідної інформації та збільшення навантаження на сервери баз даних. Кешування малозмінних даних високонавантаженої системи допомогло б вирішити дану проблему, але стандартна реалізація SQL інтерфейсу мови програмування Java не має такої функціональності. В статті проведено порівняльний аналіз основних моделей кешування. Отримані дані показали, що є потреба в у більш простому у використані кеші, що має інтеграцію зі стандартним SQL інтерфейсом Java.

Ключові слова: кешування, веб-додаток, інвалідація, Java.

Опис проблеми

Однією з найголовніших характеристик більшості комп’ютеризованих систем є швидкість доступу до необхідної для кінцевого користувача інформації за найкоротший проміжок часу за певним критерієм та зменшення навантаження на системи, з котрими дана комп’ютеризована система взаємодіє. Слід зазначити, що кінцевим користувачем може бути як людина, так й інший інформаційний додаток, що використовує результати роботи даної системи [1,2].

Розглядаючи більшість існуючих комп’ютеризованих систем, реалізованих на мові програмування Java, можна помітити, що багато з них для роботи з інформацією використовують широко розповсюджені SQL бази даних [3,4]. Для взаємодії з цими базами даних використовують стандартну реалізацію SQL інтерфейсу й мають велику кількість запитів до малозмінних даних (каталоги продукції, шаблони документів, опис товарів, конфігурації систем, таблиці доступу і багато інших), що призводить до нераціонального використання інтернет трафіку (у випадку коли БД знаходиться на іншому сервері), зменшення швидкості доступу до необхідної інформації та збільшення навантаження на сервери баз даних.

Кешування малозмінних даних високонавантаженої системи допомогло б вирішити дану проблему, але стандартна реалізація SQL інтерфейсу мови програмування Java не має такої функціональності, в зв’язку з цим постає задача розробити та реалізувати в рамках даної мови програмування кеш, що дозволив би зменшити навантаження на SQL базу даних інформаційної системи та збільшити швидкість доступу до необхідної інформації.

 ISSN 1560-8956

Огляд існуючих рішень

Що ж таке кеш, які аналоги існують та які відмінності між ними? Кеш – проміжний буфер з швидким доступом, що містить інформацію, яка може бути необхідна з найбільшою ймовірністю. Доступ до даних в кеші здійснюється швидше, ніж вибірка вихідних даних з більш повільної пам'яті або віддаленого джерела, однак його обсяг істотно обмежений у порівнянні зі сховищем вихідних даних [5].

На даний момент одним з відомих кешів для Java є OSCache [6]. Основним його призначенням є кешування динамічного контенту, тобто зображень, бінарізованих і PDF файлів, а також інтернет сторінок в цілому.

Переваги:

- наявність персистентного кешу;
- підтримка кластеризації;
- підтримка фреймворку hibernate;
- підтримка кешування web-частини;
- можливість кешувати звичайні Java об’єкти.

Недоліки:

- веб-орієнтоване кешування;
- частина функціональності являється не безкоштовною;
- немає інтеграції з SQL інтерфейсом Java;
- залежність від сторонніх бібліотек;
- фреймворк-орієнтована підтримка;
- обмеження яке вимагає наявність Servlet 2.3 / JSP 1.2 сумісного контейнера;
- відсутність широкого розповсюдження у комп’ютеризованих додатках.

Наступним кешем на ринку IT індустрії є SwarmCache [7]. SwarmCache є простим, але ефективним розподіленим кешем. Для ефективної взаємодії з будь-яким числом хостів в локальній мережі він використовує IP Multicast. Цей кеш спеціально розроблений для роботи у кластерних середовищах для веб-додатків що використовують бази даних. Такі програми, як правило, мають більшу кількість операцій читання ніж операцій запису, що дозволяє SwarmCache збільшити продуктивність додатку. SwarmCache використовує JavaGroups для управління і комунікації з розподіленої кеш-пам'яттю. Для даного кешу існують інтерфейси для взаємодії з такими популярними фреймворками як Hibernate та JPOX.

Концепція SwarmCache досить проста. Кожен сервер ініціалізує свій власний менеджер. Для кожного типу об'єкта що сервер бажає закешувати, він створює кеш і додає його менеджеру. Менеджер приєднується до multicast групи і обмінюється даними з іншими менеджерами в групі. Всякий раз, коли об'єкт віддаляється з кешу, менеджер сповіщає всі інші менеджери в групі. Ці менеджери гарантують, що об'єкт буде видалений з їх відповідних кешів. Результатом є те,

що сервери не будуть мати в своєму кеші застарілої версії об'єкта, який був оновлений або видалений на іншому сервері.

Переваги:

- підтримка різних стратегій кешування;
- наявність IP multicast;
- розподілений кеш;
- підтримка декількох сучасних фреймворків;
- має досить інтуїтивний інтерфейс.

Недоліки:

- об’єктно-орієнтоване кешування;
- відсутня інтеграція з стандартним SQL інтерфейсом Java;
- залежність від сторонніх бібліотек;
- фреймворк-орієнтована розробка;
- складний у налаштуванні;
- відсутність широкого розповсюдження у комп’ютеризованих додатках;
- відсутність транзакційності;
- наявність тільки розподіленого кешування.

Також на ринку можна помітити JBoss TreeCache – деревовидний, розподілений, транзакційний кеш який є основою для багатьох фундаментальних служб кластеризації сервера додатків JBoss [8]. Може бути використаний в якості окремого кешу або навіть в якості об’єкто-орієнтованого сховища даних. Може бути вбудований в інші J2EE-сумісні сервера, такі як BEA WebLogic або IBM WebSphere, контейнери сервлетів, таких як Tomcat, або навіть в Java додатки, які не виконуються всередині сервера додатків.

Переваги:

- підтримка різних стратегій кешування;
- наявність персистентного кешу;
- наявність розподіленого кешу;
- наявність транзакційності;
- наявність ліцензії що забезпечує підтримку виробника.

Недоліки:

- надмірна функціональність;
- необхідність придбання дуже дорогої ліцензії;
- орієнтованість на сервера додатків;
- відсутність інтеграції з стандартним SQL інтерфейсом Java;
- залежність від сторонніх бібліотек;
- складність у налагодженні;
- необхідність у потужних серверах;
- не інтуїтивний інтерфейс;
- складність впровадження.

 ISSN 1560-8956

Найбільш використовуваним на даний момент кешем є Ehcache [9]. Це поширений Java-розподілений кеш з відкритим вихідним кодом для кешування загального призначення, використовується як в Java EE так і в фреймворках.

Ehcache може реалізовувати кілька різних стратегій кешування, наприклад - LFR (кешування виходячи з частоти використання) або FIFO, також можна контролювати кеш на рівні окремих об'єктів (задаючи схему інвалідації об'єктів в кеші - за часом життя або часу простою). Додаток підтримує стандартний протокол роботи з кешем - JSR107 JCACHE.

Частина функцій Ehcache реалізуються через сторонні компоненти (наприклад, розподілені кеші). Розподілене кешування може застосовуватися для забезпечення роботи на кластері або в інших розподілених схемах, використовується вбудований (з версії 1.2) механізм RMI на основі протоколу TCP. Синхронізація та/або реплікація кешів між вузлами може бути як загальною, так і локальною, для окремих кешів за своєю схемою, асинхронною або синхронною.

Однією з унікальних функцій Ehcache є «персистентний кеш», який дозволяє зберігати стан об'єктів навіть після перезавантаження віртуальної java-машини. Крім цього, система кешування підтримує консоль управління JMX, що дозволяє інтегрувати її в систему управління будь-яким додатком, використовуючи тільки стандартні протоколи і можливості.

До переваг даного кешу відноситься підтримка різних стратегій кешування, підтримка механізму RMI, наявність персистентного кешу, підтримка консолі JMX, наявність розподіленого кешу, підтримка декількох сучасних фреймворків. Але, більшість з цих переваг є надмірними й майже ніколи не використовуються, основна частина його функціональності платна, дуже неефективна і складна інтеграція через сторонні фреймворки з SQL інтерфейсом Java, жорстка залежність від сторонніх бібліотек та фреймворк-орієнтовна розробка.

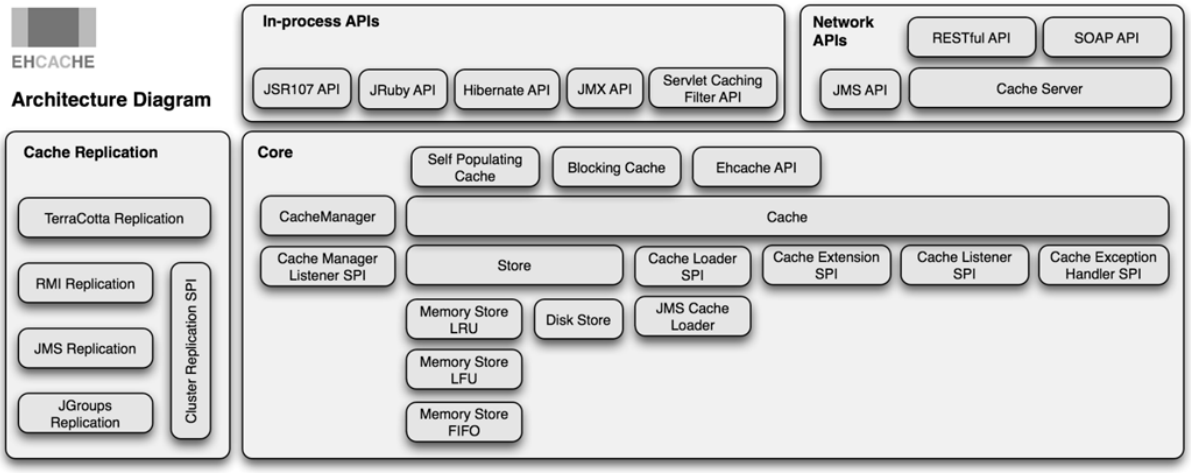

Рис.1. Архітектура Ehcache

Висновки

Приведений вище аналіз демонструє, що існує потреба у більш простому у використані кеші, що має інтеграцію зі стандартним SQL інтерфейсом Java, який би не потребував великих інвестицій.

Вирішенням проблеми є створення кешу малозмінних даних на основі SQL інтерфейсу мови програмування Java який повинен відповідати наступним вимогам:

- простота у використанні;
- інтуїтивно зрозумілий інтерфейс;
- простота у налагодженні;
- можливість використовувати кеш на будь-якій сучасній платформі;
- кеш повинен забезпечувати компактне збереження даних;
- кеш повинен зменшити навантаження на БД;
- кеш повинен зменшити час отримання малозмінної інформації у комп’ютеризованій системі.

Список використаних джерел

1. *Verma D.C.* Content Distribution Networks: An engineering approach / Wiley Inter-Science, 2002.

2. *Чумаченко П.В.* Мережі доставки контенту // П.В.Чумаченко, Т.А.Ліхоузова, О.І.Лісовиченко / Міжвідомчий науково-технічний збірник «Адаптивні системи автоматичного управління» № 2(29), 2016 – с.78-89

3. https://www.rockvalleycollege.edu/webadmin/upload/Top-10-Java-Performance-Problems.pdf

4. https://habrahabr.ru/post/201612/

5. https://en.wikipedia.org/wiki/Cache_(computing)

6. https://java-source.net/open-source/cache-solutions/oscache

7. http://swarmcache.sourceforge.net/

8. http://docs.jboss.org/jbosscache/1.2.0/TreeCache.html

9. http://www.ehcache.org/