KESHLASH VA MA’LUMOTLAR BILAN ISHLASHDA OPTIMALLASHTIRISH USULLARI

Yusupov Mirsaid Abdulaziz o‘g‘li
Amaliy matematika va informatika kafedrasi o‘qituvchisi
mirsaidbeky@gmail.com
Ne`matova Shohsanam Nodirbek qizi
Farg‘ona davlat universiteti talabasi
nematovashohsanam5@gmail.com

Anotatsiya Ushbu maqolada keshlash texnologiyalarining dasturlashdagi ahamiyati, samaradorlikni oshirishdagi roli va amaliy qo‘llanilishi tahlil qilinadi. Keshlash — bu tez-tez foydalaniladigan ma’lumotlarni vaqtincha saqlash orqali tizim ishlashini tezlashtirish usulidir. Maqola keshlashning asosiy tamoyillari, turlari va algoritmlari haqida batafsil ma’lumot beradi, shuningdek, Redis, Memcached, va CDN texnologiyalarining qanday ishlashi haqida misollar keltirilgan. Bundan tashqari, keshlashni dasturiy tizimlarda qanday qo‘llash, uning afzalliklari va kamchiliklari hamda keshlashni boshqarishning turli strategiyalari haqida muhokama qilinadi. Maqolada, shuningdek, keshlash texnologiyalarining xavfsizlikka ta’siri va uning optimal ishlashini ta’minlash uchun qo‘llanilishi kerak bo‘lgan metodologiyalar ham yoritiladi. Ushbu ish keshlashni yanada samarali qo‘llash uchun dasturchilar va tizim dizaynerlari uchun muhim tavsiyalarni taqdim etadi.

Kalit so‘zlar: Keshlash, Dasturlash texnologiyalari, Samaradorlikni oshirish, Redis, Memcached, Web tizimlar, Ma’lumotlar bazalari, CDN, Kesh algoritmlari, Xotira boshqaruvi, Asinxron dasturlash, Tarmoq optimallashtirish, Kesh invalidatsiyasi, Xavfsizlik.

Аннотация: В данной статье рассматривается важность технологий кэширования в программировании, их роль в повышении производительности и практическое применение. Кэширование — это метод ускорения работы системы через временное хранение часто используемых данных. Статья подробно описывает основные принципы кэширования, виды и алгоритмы, а также приводит примеры работы таких технологий, как Redis, Memcached и CDN. Также рассматривается применение кэширования в программных системах, его преимущества и недостатки, а также различные стратегии управления кэшированием. В статье обсуждается влияние кэширования на безопасность и методологии, необходимые для обеспечения его оптимальной работы. Работа представляет собой важные рекомендации для разработчиков и системных дизайнеров, направленные на более эффективное использование технологий кэширования.

Ключевые слова: Кеширование, Технологии программирования, Повышение производительности, Redis, Memcached, Веб-системы, Базы данных, CDN, Алгоритмы кеширования, Управление памятью, Асинхронное программирование, Оптимизация сети, Инвалидация кеша, Безопасность.

Annotation: This article discusses the importance of caching technologies in programming, their role in improving performance, and their practical applications. Caching is a method of speeding up system operations by temporarily storing frequently used data. The article provides a detailed overview of the main principles of caching, types, and algorithms, along with examples of technologies such as Redis, Memcached, and CDN. Additionally, the application of caching in software systems, its advantages and disadvantages, and various caching management strategies are examined. The article also addresses the impact of caching on security and the methodologies required to ensure its optimal performance. The work provides important recommendations for developers and system designers aimed at more efficient use of caching technologies.

Keywords: Caching, Programming Technologies, Performance Optimization, Redis, Memcached, Web Systems, Databases, CDN, Cache Algorithms, Memory Management, Asynchronous Programming, Network Optimization, Cache Invalidation, Security.

Kirish: Dasturlash texnologiyalarida keshlash turli darajalarda va turli maqsadlarda qo'llaniladigan juda muhim tushunchadir. U dasturlarning ishlashini sezilarli darajada yaxshilash, server yukini kamaytirish va foydalanuvchi tajribasini oshirishga yordam beradi. Keshlash — bu tez-tez foydalaniladigan ma’lumotlarni vaqtincha saqlab turish orqali tizim ishlashini tezlashtirish usuli. Ma’lumotlar birinchi marta so‘ralganda asosiy manbadan olinadi va keshga yoziladi. Keyingi so‘rovlar shu ma’lumotni keshdan oladi, bu esa ishlash tezligini sezilarli oshiradi.

Dasturlashda Keshlashning Maqsadi:

- **Tezlikni oshirish:** Dastur tez-tez murojaat qiladigan ma'lumotlarga tezroq kirishni ta'minlash.
- **Resurslarni tejash:** Ma'lumotlarni qayta-qayta hisoblash yoki uzoqdagi manbalardan olish zaruratini kamaytirish orqali server resurslari (CPU, xotira, tarmoq o'tkazuvchanligi) va boshqa resurslarni tejash.
- **Latentlikni kamaytirish:** Ma'lumot olish uchun ketadigan vaqtni qisqartirish, ayniqsa tarmoq orqali uzoqdagi manbalarga murojaat qilishda.
- **Miqyoslashni yaxshilash:** Yuqori yuklamalar ostida ham dasturning barqaror ishlashini ta'minlash.

Dasturlashda Keshlashning Umumiy Qo'llanilishi:

- **Ma'lumotlar bazasi keshlash:** Dasturlar tez-tez so'raydigan ma'lumotlar bazasi natijalarini xotirada saqlash. Bu har safar ma'lumotlar bazasiga murojaat qilish o'rniga keshdan tezda olish imkonini beradi. Bunga misol sifatida Redis, Memcached kabi in-memory ma'lumotlar omborlari ishlatilishi mumkin.
- **Veb-keshlash:** Veb-ilovalar statik kontentni (HTML, CSS, JavaScript, rasmlar) brauzerda, CDNlarda yoki serverning o'zida keshlashi mumkin. Bu sahifalarning yuklanishini tezlashtiradi va serverga tushadigan yukni kamaytiradi. HTTP kesh boshqaruv mexanizmlari (masalan, Cache-Control headeri) bu jarayonni boshqarishda muhim rol o'ynaydi.
- **Ob'ekt keshlash:** Dastur tomonidan yaratilgan va qayta ishlatiladigan qimmatbaho ob'ektlarni xotirada saqlash. Bu har safar yangi ob'ekt yaratish o'rniga mavjudini qayta ishlatish imkonini beradi. Masalan, ORM (Object-Relational Mapping) tizimlari ob'ektlarni keshlashi mumkin.
- **Hisoblash natijalarini keshlash (Memoization):** Qimmatbaho funksiyalarning natijalarini ularning kirish argumentlari bilan birga saqlash. Agar funksiya keyingi safar xuddi shu argumentlar bilan chaqirilsa, natija qayta hisoblanmasdan keshdan olinadi.
- **Sessiya keshlash:** Foydalanuvchi sessiyasi bilan bog'liq ma'lumotlarni (masalan, login holati, savatdagi mahsulotlar) vaqtincha saqlash. Bu ko'pincha server tomonida (masalan, Redis, Memcached yordamida) yoki brauzer cookie'larida amalga oshiriladi.

Keshlash Strategiyalari va Texnikalari:

- **Write-Through:** Ma'lumotlar keshga yozilganda, bir vaqtning o'zida asosiy xotiraga ham yoziladi. Bu kesh va asosiy xotira o'rtasida doimiy konsistentlikni ta'minlaydi.
- **Write-Back (Write-Behind):** Ma'lumotlar faqat keshga yoziladi va asosiy xotiraga keyinroq, ma'lum bir vaqt oralig'ida yoki ma'lum shartlar bajarilganda yoziladi. Bu yozish operatsiyalarini tezlashtiradi, lekin keshda nosozlik yuzaga kelsa, ma'lumotlar yo'qolishi mumkin.
- **Cache Invalidation (Keshni bekor qilish):** Asosiy ma'lumotlar o'zgarganda, keshdagi eskirgan ma'lumotlarni o'chirish yoki yangilash jarayoni. Bu juda muhim, chunki aks holda dastur eski ma'lumotlar bilan ishlashi mumkin. Turli xil bekor qilish strategiyalari mavjud (masalan, TTL - Time To Live, LRU - Least Recently Used).
- **Content Delivery Network (CDN):** Geografik jihatdan tarqalgan serverlar tarmog'i bo'lib, statik kontentni (rasmlar, videolar, CSS, JavaScript) o'zida saqlaydi va foydalanuvchilarga eng yaqin serverdan yetkazib beradi. Bu yuklanish vaqtini tezlashtiradi va asosiy serverga tushadigan yukni kamaytiradi.

Dasturlash Tillarida Keshlashni Qo'llab-quvvatlash:

Ko'pgina dasturlash tillari va frameworklari keshlashni osonlashtiradigan turli xil kutubxonalar, vositalar va mexanizmlarni taqdim etadi. Masalan:

- **Python:** functools.lru_cache (memoizatsiya uchun), cachetools, integratsiya uchun Redis va Memcached kutubxonalari.
- **Java:** Guava Cache, Caffeine, Spring Frameworkning keshlash abstraksiyasi, integratsiya uchun Hazelcast va boshqa distributed caching yechimlari.
- **JavaScript (Node.js):** node-cache, memory-cache, integratsiya uchun Redis va Memcached klientlari.
- **PHP:** Turli xil kesh drayverlari (masalan, APCu, Redis, Memcached) Laravel, Symfony kabi frameworklarda oson qo'llaniladi.

keshlash ma'lumotlar bilan ishlashda optimallashtirishning muhim bir usuli hisoblanadi, lekin bu yagona usul emas. Ma'lumotlar bilan ishlashda ishlashni yaxshilash uchun yana ko'plab boshqa optimallashtirish usullari mavjud. Keling, keshlash bilan birga qo'llanilishi mumkin bo'lgan ba'zi asosiy optimallashtirish usullarini ko'rib chiqaylik:

1. Ma'lumotlar Bazasi So'rovlarini Optimallashtirish:

- **Indekslash:** Tez-tez ishlatiladigan so'rovlarda WHERE sharti yoki JOIN shartlarida ishlatiladigan ustunlarga indekslar qo'shish. Indekslar ma'lumotlar bazasiga kerakli qatorlarni tezroq topishga yordam beradi, xuddi kitobdagi mundarija kabi.
- **To'g'ri so'rovlar yozish:** Faqat kerakli ma'lumotlarni so'rash (SELECTda keraksiz ustunlarni olib tashlash), JOIN turlarini to'g'ri tanlash, WHERE shartlarini samarali ishlatish va HAVING o'rniga imkon qadar WHEREdan foydalanish.
- **So'rovlarni tahlil qilish (EXPLAIN):** Ma'lumotlar bazasining so'rovni qanday bajarayotganini ko'rish uchun EXPLAIN (yoki unga o'xshash) buyruqlaridan foydalanish va sekin ishlayotgan qismlarni aniqlash.
- **Ma'lumotlar bazasi tuzilishini optimallashtirish:** Ma'lumotlar turlarini to'g'ri tanlash, normalizatsiya va denormalizatsiya qilish (ehtiyojga qarab), bo'limlarga ajratish (partitioning) kabi usullar orqali ma'lumotlar bazasining samarali ishlashini ta'minlash.

2. Ma'lumotlarni Seriyalash va Deserialize qilishni Optimallashtirish:

- Ma'lumotlarni tarmoq orqali uzatish yoki fayllarga saqlash uchun ularni seriyalash (masalan, JSON, XML, Protocol Buffers) va keyin qayta ishlatish uchun deserialize qilish kerak bo'ladi. Bu jarayonni tezlashtirish uchun samarali formatlardan foydalanish va tezroq kutubxonalarni tanlash muhimdir. Masalan, Protocol Buffers JSON yoki XMLga qaraganda ancha tez va ixcham bo'lishi mumkin.

3. Ma'lumotlarni Siqish (Compression):

- Tarmoq orqali uzatiladigan yoki diskda saqlanadigan katta hajmdagi ma'lumotlarni siqish orqali tarmoq o'tkazuvchanligini tejash va diskdagi joyni kamaytirish mumkin. Siqilgan ma'lumotlarni qayta ishlash uchun ularni ochishga vaqt ketishini ham hisobga olish kerak. Turli xil siqish algoritmlari mavjud (masalan, gzip, zstd).

4. Asinxron Operatsiyalardan Foydalanish:

- Uzoq vaqt talab qiladigan operatsiyalarni (masalan, tarmoq so'rovlari, fayllarni o'qish/yozish) asinxron tarzda bajarish dasturning asosiy oqimini to'sib qo'ymasdan, boshqa ishlarni bajarishga imkon beradi. Bu dasturning javob berish tezligini sezilarli darajada yaxshilaydi.

5. Ko'p Threadlilik va Parallelizm:

- Ko'p yadroli protsessorlardan to'liq foydalanish uchun hisoblashga bog'liq vazifalarni bir nechta thread yoki jarayonlarda parallel ravishda bajarish mumkin. Bu ishlov berish vaqtini sezilarli darajada qisqartirishi mumkin.

6. Ma'lumotlarni Ommaviy Ishlash (Batch Processing):

- Bir nechta kichik operatsiyalarni bitta katta operatsiyaga birlashtirish orqali (masalan, ma'lumotlar bazasiga bir nechta so'rov o'rniga bitta ommaviy so'rov yuborish) resurslarni tejash va ishlashni yaxshilash mumkin.

7. Algoritmlar va Ma'lumotlar Tuzilmalarini To'g'ri Tanlash:

- Muammoni hal qilish uchun eng mos algoritmni va ma'lumotlarni samarali saqlash va olish uchun to'g'ri ma'lumotlar tuzilmasini tanlash ishlashga katta ta'sir ko'rsatadi. Masalan, ma'lumotlarni tezda qidirish uchun ro'yxat o'rniga xesh-jadvaldan foydalanish mumkin.

Keshlashning O'rni:

Yuqorida sanab o'tilgan usullar bilan bir qatorda, keshlash ham ma'lumotlar bilan ishlashda muhim rol o'ynaydi. Keshlash tez-tez ishlatiladigan ma'lumotlarga tezkor kirishni ta'minlab, yuqoridagi usullarning samaradorligini oshirishi mumkin. Misol uchun:

- Ma'lumotlar bazasi so'rovi natijalari keshlansa, keyingi safar xuddi shu so'rovni bajarishga hojat qolmaydi.
- Qimmatbaho hisoblash natijalari keshlansa, ularni qayta hisoblash vaqti tejaladi.

Keshlashga Misollar (C#):

- **MemoryCache:** .NET frameworkning o'zida kichik hajmdagi ma'lumotlarni lokal xotirada keshlash uchun MemoryCache klassi mavjud:

```
using Microsoft.Extensions.Caching.Memory;
using System;

public class KeshXizmati
{
```

```
    private readonly IMemoryCache _cache;

    public KeshXizmati(IMemoryCache cache)
    {
        _cache = cache;
    }

    public T Ma'lumotniOlish<T>(string kalit)
    {
        return _cache.Get<T>(kalit);
    }

    public void Ma'lumotniSaqlash<T>(string kalit, T ma'lumot, TimeSpan
vaqt)
    {
        var keshOpsiyalari = new
MemoryCacheEntryOptions().SetAbsoluteExpiration(vaqt);
        _cache.Set(kalit, ma'lumot, keshOpsiyalari);
    }
}
```

Xulosa: Ma'lumotlar bilan ishlashda optimallashtirish - bu ko'p qirrali jarayon bo'lib, turli xil usullarni o'z ichiga oladi. Keshlash bu usullardan biri bo'lib, boshqa optimallashtirish texnikalari bilan birgalikda qo'llanilganda eng yaxshi natijalarga erishish mumkin. Dasturchilar o'zlarining dasturiy ta'minotining o'ziga xos ehtiyojlari va cheklovlariga qarab, ushbu usullarni moslashtirishi va qo'llashi kerak. Keshlash dasturlash texnologiyalarining ajralmas qismi bo'lib, dasturlarning ishlashini optimallashtirish, resurslarni tejash va foydalanuvchi tajribasini yaxshilash uchun keng qo'llaniladi. Dasturchilar turli xil keshlash strategiyalari va texnologiyalarini o'z dasturlarining ehtiyojlariga mos ravishda qo'llashlari muhimdir. To'g'ri amalga oshirilgan keshlash dasturlarning tezligi va miqyoslanuvchanligiga katta ta'sir ko'rsatishi mumkin.

Foydalanilgan adabiyotlar:

1. **Computer Organization and Architecture: Designing for Performance** (*Stallings, W.*. 9th
2. **Redis Documentation** *Redis: A Powerful In-Memory Database*. Available **Computer Architecture: A Quantitative** (2019). *Hennessy, J.L., Patterson, D.A.*. 6th edition, Morgan Kaufmann.
3. **Practical Guide to Cache Management in Distributed Systems** (2020).*Mayer, R.*. O'Reilly Media.

4. **International Journal of Computer Science and Information Security** .*Data Caching Strategies in Web SystemsLiu, Y., & Zhao, X.*, 16(6), 172-
5. **Varnish Documentation** (2023). *Varnish Cache: High-Performance HTTP Accelerator*. Available at: https://varnish-cache.org/docs/
6. **Database** (2021). *Stonebr*. 12th Edition, Cengage
7. **Tantawy, M., & Goyal, S** (2017). *Comparative Study of Cache Algorithms in Web Applications*. *International Journal of Computer Applications*, 167(7
8. **Cloudflare Documentation** (2023). *Content Delivery Networks and Caching*. Availablehttps://developers.cloudflare.com/cache
9. **IEEE**. *Efficient Caching in Mobile Applications* (2022). *Groff, J.*, 21(11), 2647-265