УДК 519.687.7

МОДЕЛЬ МЕНЕДЖЕРА КЕШ-ПАМ'ЯТІ НА ОСНОВІ НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Д. М. Вдовичинський[1, a], А. М. Родіонов[1]

[1] *Національний технічний університет України «Київський політехнічний інститут»*

Анотація

У даній роботі проведено огляд основних сучасних алгоритмів роботи з кеш-пам'яттю, описано математичну модель менеджера кеш-пам'яті(кеш-контролеру) та застосування методів навчання з підкріпленням для вирішення задачі заміщення сторінок.

Ключові слова: кеш-пам'ять, кеш-контролер, навчання з підкріпленням

Вступ

Швидке збільшення темпів використання в сучасному світі новітніх інформаційних технологій вимагає від розробників як пошук нових технічних рішень, так і розробку ефективних технологій, які забезпечують зберігання, аналіз і обробку значних масивів різноманітної інформації.

На даний час усе більшу роль відіграють технології, які забезпечують зберігання, аналіз та обробку даних. В процесі обробки даних часто виникають проблеми забезпечення продуктивності обчислень. Сучасні системи працюють з об'ємами даних, розмір яких значно перевищує обсяг доступної оперативної та кеш-пам'яті – швидкої пам'яті невеликої місткості, яка розташована між процесором і основною пам'яттю і зберігає найбільш часто використовувані дані і команди.

Великі об'єми даних неможливо одночасно розмістити в оперативній та кеш-пам'яті, тому більша частина даних під час роботи системи залишатиметься на зовнішніх носіях, що зводить продуктивність системи до мінімуму. Практично єдиним ефективним способом зменшення кількості звертань до зовнішніх носіїв є механізми кешування даних, реалізовані кеш-контролером, основною задачею якого є прогнозування звернень процесора до пам'яті.

Кеш контролер займається виконанням задачі заміщення сторінок – задачі керування пам'яттю комп'ютера, що полягає у розподілі між швидкою та повільною пам'яттю сторінок(даних) відносно частоти запитів на сторінки.

Метою цієї роботи є аналіз ефективності використання алгоритмів методу навчання з підкріпленням при вирішені задачі заміщення сторінок.

1. Модель менеджера кеш-пам'яті

Для порівняння алгоритмів кешування буде використана модель менеджера кеш-пам'яті, зображена на рис. 1.

[a] chinskiy93@gmail.com

Рис. 1. Модель менеджера кеш-пам'яті

Кожен новий процес містить невпорядкований набір комірок пам'яті з повтореннями, у яких міститься інформація, для виконання процесу. Процес передається у диспетчер завдань, який послідовно вимагає від кеш-контролера отримати певну комірку пам'яті. Після отримання усіх комірок пам'яті процес завершує свою роботу і створюється новий процес.

У кеш-контролері використовується алгоритм кешування, який за обраним алгоритмом керує кеш-пам'яттю, додаючи чи забираючи комірки пам'яті із сховища даних. Також у кеш-контролері, окрім алгоритму кешування здійснюється підрахунок кеш-влучення, частоти зміни кеш-пам'яті та інші статистичні дані.

Сховище даних містить впорядкований набір комірок пам'яті без повторень, яке містить дані для виконання процесів.

Перевагою даної моделі є лише зміна алгоритму кешування у кеш-контролері. Усі додаткові елементи, як, наприклад, лічильник «*віку*» у алгоритмі LRU, будуть знаходитись у кеш-контролері.

2. Алгоритми кешування

На теперішній час існує велика кількість алгоритмів кешування, основними з яких є наступні: [1]

- **LRU** (англ. Least Recently Used) – в даному алгоритмі витісняються елементи, які довше за інші не використовувались. Для реалізації цього алгоритму використовується лічильник «*віку*» – числове значення, яке показує на час останнього звернення до елементу. Алгоритм LRU простий в реалізації, але, очевидно, що його можливості прогнозування досить обмежені;

- **MRU** (англ. Most Recently Used) – алгоритм протилежний до LRU, так як витісняються елементи до яких відбулось останнє звернення. Цей алгоритм інколи є більш ефективним за LRU, наприклад у випадках циклічного сканування великих наборів даних;
- **LFU** (англ. Least Frequently Used) – алгоритм, заснований на підрахунку числа звернень до кожного елемента, при якому витісняються елементи, які використовуються рідше за інші. Для реалізації потрібно збереження частоти звертань до кешу;
- **SLRU** (Segmented LRU) – модифікація LRU. Кеш-пам'ять умовно поділяється на 2 сегмента: незахищений та захищений.
  Після першого запиту елемент додається в незахищений сегмент, а після кеш-влучення, алгоритм пересуває елемент в захищений сегмент. При цьому всередині кожного сегменту відбувається заміщення по алгоритму LRU, але видалення зазнають лише елементи з незахищеного сегменту.
  Коли відбувається витіснення елементу із захищеного сегменту, він додається в незахищений сегмент з максимальним значенням лічильника «*віку*».
  Такий підхід дозволяє раніше популярним елементам залишатись в кеш-пам'яті протягом довгого періоду, поки елементи, запит до яких здійснився лише один раз, видаляються з незахищеного сегменту кеш-пам'яті.

Новизна елементів – характеристика об'єкту, яка використовується відповідно до стратегії заміщення у алгоритмах LRU, MRU, SLRU, у той час як LFU використовує як таку характеристику частоту звертань до елементу.

3. Методи навчання з підкріпленням

Навчання з підкріпленням – група методів машинного навчання, у яких алгоритм самостійно намагається знайти оптимальну стратегію.

При цьому способі навчання запам'ятовується відповідність між станами $s \in S$ і діями $a \in A(s)$, які об'єкт управління (агент) повинен виконати в тій чи іншій ситуації. Загальна схема навчання з підкріпленням показана на рисунку 2 [2].

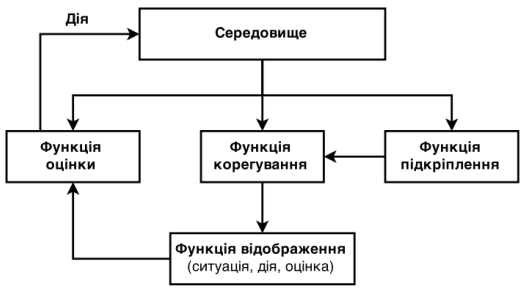

Рис. 2. Принцип роботи систем, які використовують навчання з підкріпленням

На кожному з кроків агент отримує деяке уявлення про стан середовища s та змінює свої правила в результаті отримання нового досвіду, вибирає певну дію. Під час наступного кроку, як частина відповіді на дію, агент отримує числове підкріплення і переводить себе в наступний стан.

Роботу методів навчання з підкріпленням обґрунтовано з теоретичної точки зору для випадків, коли модель системи можна подати у вигляді Марковського процесу прийняття рішень. Для цього потрібне виконання марковської властивості, що виражається рівністю формул (1) та (2):

$$P\{s_{t+1} = s', r_{t+1} = r \\ |s_t, a_t, r_t, s_{t-1}, a_{t-1}, ..., r_1, s_0, a_0\} \quad (1)$$

$$P\{s_{t+1} = s', r_{t+1} = r|s_t, a_t\}, \quad (2)$$

де s – стан системи, r – винагорода, a – дія. У роботі [3] автори використовували Марковський процес прийняття рішень до вирішення задачі заміщення сторінок.

У нашому випадку агент – кеш-контролер, який контролює вміст даних в кеш-пам'яті, а середовище – уся інша частина моделі.

На кожному дискретному кроці t агент отримує стан середовища $s_t \in S$, де S – множина станів середовища. Так як в стан можуть входити як зовнішні, так і внутрішні параметри, до стану системи входять наступні параметри:

$$s_t = \{array_t, i_t\},$$

де $array_t$ – розміщення даних у комірках пам'яті на даному кроці, i_t – наступна комірка, яку хоче отримати диспетчер завдань.

Відповідно до вибраного стану агент обирає дію $a_t \in A(s)$, де $A(s)$ – множина можливих дій, доступних із стану s_t. Множина можливих дій з стану s має вигляд

$$a_t = \{a_0, a_1, ...a_{len(array)}\},$$

де a_0 – відсутність дії на даному кроці, $a_1, ...a_{len(array)}$ – заміщення комірки пам'яті елементом, який хоче отримати диспетчер завдань.

Після виконання дії a_t, модель переходить в новий стан s_{t+1} та отримує винагороду(підкріплення) r_t.

$$r_t = [r_{cache\ hit}|r_{cache\ miss}] + [r_{action}|r_{no-action}]$$

r_t – число, що є безпосередньою оцінкою середовищем дії a_t. Кеш-контролер отримує велику позитивну винагороду за кожне кеш-влучення($r_{cache\ hit}$), та маленьку позитивну винагороду за відсутність змін у кеш-пам'яті($r_{no-action}$), відповідно велику негативну за кеш-промах($r_{cache\ miss}$), та маленьку негативну за зміну комірок кеш-пам'яті(r_{action}).

Задача агента – максимізувати сумарну винагороду. Для максимізації сумарної винагороди агенту потрібно знайти оптимальну функцію виграшу. В моделі немає конкретного обмеження на кількість кроків, тому буде зручно використовувати модель середньої винагороди, в якій агент хоче максимізувати границю середньої винагороди, так як дана модель не робить різниці між найближчою та віддаленою винагородою:

$$\lim_{n\to\infty} E\{\frac{1}{h}\sum_{t=0}^{h} r_t\}, \gamma \in (0,1]$$

Кеш-контролер повинен обрати стратегію для отримання винагороди.

Стратегія – функція $\pi : S \times A \to [0,1]$, де $\pi(s,a)$ – ймовірність вибору дії $a \in A(s)$ в стані s. В даному випадку стратегія є детермінованою тому в кожному стані існує дія, яку агент обере з ймовірністю одиниця, дія, яка обирається позначається $\pi(s)$.

Функція очікуваного виграшу $V^\pi(s)$ – чисельна величина, яку очікує отримати агент при умові, що агент дотримується стратегії π, та починає роботу з стану s:

$$V^\pi(s) = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

Також можна розглянути іншу функцію виграшу для якої відома перша дія a, яка обирається агентом, який знаходиться в стані s.

$$Q^\pi(s,a) = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

Для отримання максимальної винагороди агенту потрібно знайти оптимальну стратегію π. Знання оптимальної функції виграшу дозволяє побудувати оптимальну стратегію. В основі наступних методів лежить обчислення оптимальної функції виграшу[4]:

- **TD-методи**
  Дані методи не потребують повної інформації про зовнішнє середовище і використовують результати безпосередньої взаємодії агента і середовища, а також наявні оцінки для обчислення нових оцінок.
  Знаходячись в стані s, агент здійснив дію a та перейшов в стан s', отримавши при цьому винагороду r. Тоді оцінку $V(s)$ значення $V^\pi(s)$ можна оновити наступним чином:
  $$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s)).$$
  Значення $V^\pi(s')$ невідомо, тому для оцінки загальної суми винагород, яку отримає агент, використовується наявна оцінка для стану s':
  $$R = r + \gamma V(s').$$
  Таким чином, оцінка $V(s)$ зсувається в сторону нової оцінки на долю α від різниці двох оцінок. Починаючи з нульових значень $V(s)$, агент на кожному кроці використовує вказане правило для оновлення значень $V(s)$, отримуючи при цьому все більш точні оцінки значень функції виграшу.
- **Q-навчання**
  Для знаходження оптимальної стратегії метод апроксимує значення оптимальної функції виграшу Q^*.
  Метод Q-навчання оновлює на кожному кроці оцінки значень функції виграшу Q^π поточної стратегії π, використовуючи наступну формулу:
  $$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma * max_a Q(s_{t+1},a) - Q(s_t,a_t)),$$
  де s_t – стан на кроці t, a_t – дія, яка відбулася на кроці t, r_{t+1} – отримана винагорода, s_{t+1} – стан, в який перейшли.
  Покращення даної стратегії проводиться шляхом вибору із даного стану s дії, яка максимізує оцінку Q.
  Агент в процесі своєї роботи покращує свою поточну стратегію, обираючи дії, які призводять до найбільшого виграшу, і в той же час на кожному кроці покращує оцінки, які використовуються при виборі дії.

У описаних методах обчислення оптимальної функції виграшу вважається, що агент містить оцінки значень функції виграшу для кожного стану чи пари стан-дія. Найпростіший спосіб зберігання даної інформації – таблиця, але такий спосіб збереження інформації неможливий за рахунок великої кількості станів, і внаслідок великого розміру таблиці.

Для вирішення даної проблеми значення функції виграшу можна апроксимувати за допомогою нейронної мережі. На вхід нейронної мережі подаються числа, які описують стан(чи пару стан-дія), а значення на виході використовується в якості оцінки виграшу для даного стану(чи пари стан-дія).

Для оновлення значень оцінок використовується навчання нейронної мережі. В якості алгоритму навчання можна використати метод зворотнього поширення помилки. При цьому прикладом для навчання мережі є пара з вхідних даних та нової оцінки.

Висновки

У даній роботі була розглянута теоретична база для побудови моделі кеш-контролера на основі методів навчання з підкріпленням для вирішення задачі заміщення сторінок кеш-пам'яті, та в майбутньому отримання експериментальної оцінки та порівняння отриманих результатів з оцінками сучасних алгоритмів. Оцінка ефективності застосування цих методів буде проведена із застосуванням описаної моделі менеджера кеш-пам'яті.

Перелік використаних джерел

1. Caching Strategies to Improve Disk System Perfomance / R. Karedla, J. S. Love, B. G. Wherry — IEEE Computer, Vol.27, 1994. — pp. 38-46.
2. Artificial Intelligence: A Modern Approach / S. J. Russell, P. Norvig — Prentice Hall 1995. — pp. 596-600.
3. Cache Performance of Priority Metrics for MDP Solvers / D. Wingate, K. D. Seppi — AAAI Workshop on Learning and Planning in Markov Processes. — 2004. — pp. 103-106.
4. Р. С. Саттон, Э. Г. Барто. Обучение с подкреплением / пер. с англ. — М. : БИНОМ. Лаборатория знаний, 2011. — 399 с.