Методика. Сформулирован алгоритм „колонии больших и малых муравьев“ со, своего рода, стратегией „прыжок через яму“. При этом большие муравьи могут переносить большее количество феромона и более склонны к ошибкам.

Результаты. Впервые использован алгоритм „колонии больших и малых муравьев“ для увеличения скорости схождения. Затем, при использовании концепции „прыжок через яму“, был реализован поиск путей в более широком диапазоне, где стратегия „малого прыжка“ позволяет более чем одному муравью выбрать альтернативный путь, а стратегия „большого прыжка“ позволяет ставить преграды на пути, отмеченном феромоном, и вынуждать муравьев выбирать другой путь. Результаты экспериментов показали, что модифицированный алгоритм всегда достигает оптимального результата в отличие от MMAS-алгоритма.

Научная новизна. Изучен новый „муравьиный алгоритм“ и рассмотрена эффективность идеи, ранее не обсуждавшейся.

Практическая значимость. Предложенный алгоритм может использоваться для решения многих задач, особенно в сочетании с каким-либо детерминированным алгоритмом для реализации быстрой общей оптимизации.

Ключевые слова: *муравьиный алгоритм, „задача коммивояжера“, Max-Min муравьиная система, задача недетерминированной полиномиальной сложности, глобальный поиск, феромоны, стратегия „прыжок через яму“*

Рекомендовано до публікації докт. техн. наук В.І. Корнієнком. Дата надходження рукопису 21.11.14.

Heng Yang

Nanyang Normal University, Nanyang, China, e-mail: yheng_sir@sina.cn

A REPLACING STRATEGY BASED LEAST RECENTLY USED ALGORITHM IN STORAGE SYSTEM

Хен Ян

Наньянский педагогічний університет, м. Наньян, провінція Хенань, КНР, e-mail: yheng_sir@sina.cn

СТРАТЕГІЯ КЕШУВАННЯ, ЗАСНОВАНА НА АЛГОРИТМІ ВИТІСНЕННЯ ДАВНО НЕВИКОРИСТАНИХ ЕЛЕМЕНТІВ У ПАМ'ЯТІ СИСТЕМИ

Purpose. Big data is a very large vocabulary we have heard recently. No matter for business or personal users, there is always a lot of important data to store. When we use the storage system, we often want the system to respond quickly enough to reach a state of no delay. This is a big challenge to the storage system. Scientists have already done many researches on this topic and they found that the use of cache in the storage system can improve the performance of storage system greatly.

Methodology. Cache algorithm is a hot research field in the current storage area. Least Recently Used algorithm (LRU) is a commonly used cache replacement algorithm.

Findings. Since the new hash value that appears atop of the stack needs to adjust the stack even if the visited page is already in memory, this takes much time. We need a better cache replacement algorithm to improve the performance.

Originality. A new replacing strategy based on the LRU algorithm has been developed; it is called Improved LRU algorithm (ILRU). It can increase the hit rate when more users suddenly have a higher access to the unfamiliar page. We determine whether the hash value is added to a page or not through searching the access hash value in the LRU queue.

Practical value. The test results show that the design of ILRU algorithm can improve the performance comparing to traditional LRU algorithm. At the same time, ILRU algorithm has higher hit rate than FIFO algorithm.

Keywords: *replacing strategy, least recently used algorithm, ILRU, hit rate, performance*

Introduction. The development of internet is very rapid. According to the U.S. Department of Commerce survey, the data on internet traffic is going to be double on average every one hundred days. The fast development brings not only the tremendous business opportunities but also creates a big problem to storage system. People will have high requirements to the storage system performance.

In recent years, people have used the cache to increase the performance of the storage system. The main function of the storage system is as a cache between the high speed equipment and the low speed device. A good cache replacement algorithm may be designed to improve the hit rate of the content. When the content of the request is hit, it can be obtained directly from the cache, which can reduce the response time of the request [1].

In order to improve the hit rate of the cache system, we should design different cache algorithms to manage the data and make out the elimination decision when the buffer space is full. The cache algorithm generally determines which data needs to be cached and the data blocks are selected when the space is full.

LRU algorithm is the most commonly used cache algorithm. LRU is the abbreviation of Least Recently Used, which services for the virtual memory management. For the virtual page storage, internal and external information is replaced by the page. When we need a page to be placed in the external memory, we transfer it to the memory. In order to keep the size of the original space, we also move less the process execution. The present of LRU algorithm is based on the assumption that the pages used frequently in the previous instructions may be used frequently in the following instructions [2]. On the contrary, the page that has not been used for a long time most likely will not be used in the near future. This is the famous local principle. Therefore, we only need to find the least used page to call out of memory at every time of exchange. The drawback of LRU algorithm is obvious. When there are more users to access the unfamiliar page suddenly, the system will produce a higher miss rate. That is why it is necessary to improve the LRU algorithm.

In the research, the ILRU algorithm has been developed which can increase the hit rate when more users suddenly have a higher access to the unfamiliar page. We determine whether the hash value is added to a page or not through searching the access hash value in the LRU queue.

Findings of the research:

• After the analysis of the disadvantage of LRU algorithm, the ILRU algorithm has been designed to improve the hit rate of cache in the storage system.

• Experimental results show that ILRU algorithm has improved the performance of the system significantly.

Related work. According to the cache level, the cache algorithm can be divided into two categories: single level cache and multi levels cache. In the single level cache structure, the most commonly used algorithm is LRU and LFU (Least Frequently Used) algorithm. Based on these two algorithms, there were created many improved algorithms, such as LIRS [3], LF-RU [4], ARC [5] and other algorithms. Multi-Queue [6], LRU-K [7], 2Q [8] and other algorithms are applied in multi levels cache structure. There are also some algorithms suitable for only special application scenarios. DULO [9] algorithm is customized for the database type.

The LRU-K cache replacement algorithm is based on the LRU to improve, in which K represents the number of recent visits. The algorithm records the access information about the last K times of each data block. In order to reduce the space complexity of the algorithm, the algorithm sets a history retention period. When the time information access of a data block is over the history retention time, it will be discard. When the data is eliminated, the block, which is the most distant from the last K times visit, will be selected. The algorithms generally use the heap to maintain the data. The elements in the heap are built in accordance with the last K access time. When a data block is accessed, the access time of the data block is moved. At the same time, the system will add the current time and readjust the heap. We only need to select the elements on the top of the heap in elimination, so that the time complexity of LRU-K algorithm is log (N) level.

LRU algorithm is a special case of K equal to one. The LRU-K algorithm can resist the cache pollution caused by file scanning when K is greater than one. When the file is scanned, the numbers of accesses to the data block have changed the current access time, which cannot brush out the elements in the current data cache. The LRU-K algorithm hit rate curve is shown in Fig.1.

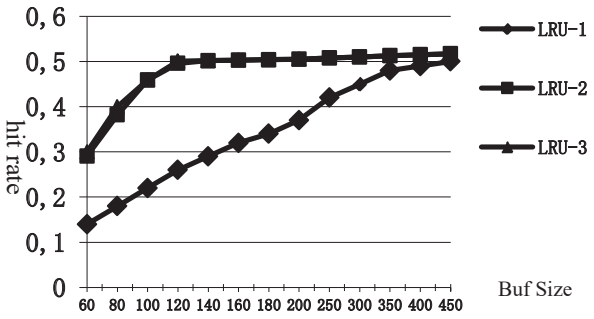

Fig. 1. LRU-K algorithm hit rate curve

In the LRU-K algorithm, the time cost of the Log (N) level is reached because the system uses a heap to maintain the data. The time cost of 2Q algorithm is constant time and the performance is similar to that of the LRU-2 algorithm. Another advantage is that it is not required to adjust the parameters of the LRU-K algorithm.

The 2Q algorithm only allows hot data blocks to enter the cache. The figure of 2Q algorithm is shown in Fig.2. From the Fig. 2, we can see that 2Q algorithm contains two queues, A1 and Am queue. A page is accessed into the A1 queue for the first time when the queue is a FIFO queue. If this page is accessed again in the A1 queue, then the page is treated as hot data pages to enter the Am queue, which is also a LRU queue. If a page has not been accessed in the A1 queue, this page is likely to be a cold data page. When it gets to the head of the queue, it will be deleted.

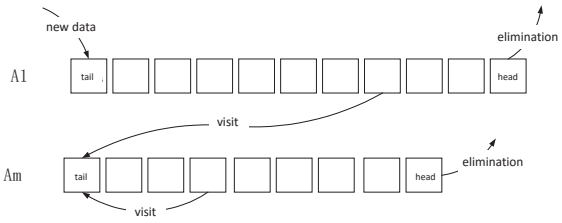

Fig. 2. The figure of 2Q algorithm

The ILRU algorithm. ***The Design of the Algorithm.*** According to the ILRU algorithm, the system creates an ILRU queue to score the data and related information. When a user accesses a page, the corresponding hash value will be calculated. The hash value will be searched in the ILRU queue. If the associated hash value does not hit in memory, a new hash value is inserted into the header of the ILRU1 queue. If the associated hash values are hit in memory, the system will find its position in the queue. In the ILRU1 queue, the associated hash value memory block is upgraded to the ILRU2 queue in the linked list. If the hit hash value of the memory block is in the ILRU2, then find its position in the queue, to move it to the head of the ILRU2 queue. The schematic diagram of IL-RU algorithm is shown in Fig. 3.

The advantages of the ILRU algorithm:

(1) The calculation of hash value. The calculation of hash value is calculated mainly according to some information in data block. Even a very good algorithm will produce conflict. In order to reduce the conflict, we use the MDS algorithm and time hash algorithm to generate two keys. The two key values for the unique identification greatly reduce the possibility of hash value conflict.

(2) Due to the use of two LRU queues, and the second level of the LRU queue is an upgrade of the first LRU queue. When a large number of users access read and write requests, it results in a large number of new hash keys. It will be replaced in the ILRU1 queue and will not lead to the entire ILRU queue refresh, so that it will appear suddenly higher miss rate.

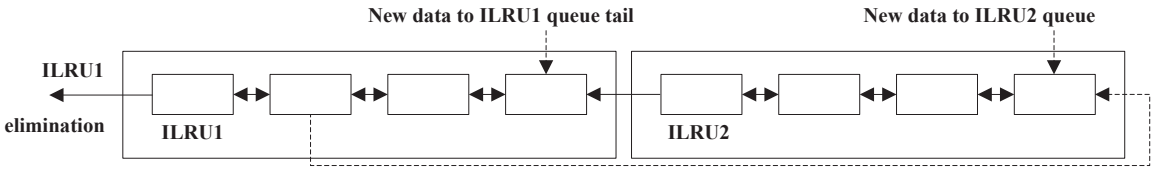

Fig. 3. The schematic diagram of ILRU algorithm

We can see from Fig. 3 that each hash value node contains two pointers, prew and next, mean the front of the node and after the node, respectively.

The whole data linked lists create two similar LRU queues, which show as the form of a data linked list in memory. The LRU queue only achieves the sorting relationship between the data pointers.

Because the data list is a two-way list, which is very easy to delete and insert data. Therefore, the hash value is a very small cost for a new connection and the insertion of the virus related information into a linked list.

Algorithm Description. The ILRU algorithm is implemented is shown in Fig. 4. Let us describe each step in detail:

(1) A new request. When a request arrives, it will be resolved out of the relevant information.

(2) Calculated hash values. According to specific data, the hash value is calculated by MDS algorithm.

(3) Search according to the hash value in the list. Look the hash value as the key, search the relevant memory block in the list.

(4) Judge whether the search is successful or not. If you find it, then go on.

(5) Check the ILRU queue. We should check the value of the ILUR queue based on the hash table.

(6) Judge whether the search result is in ILRU1 queue or not. If it is in ILRU1, then continue; otherwise go to step (9).

(7) Join in ILRU2 queue. Upgrade to LRU1 queue to the ILRU2 queue.

(8) According to the information we, process the message. According to finding the hash value, we take out the corresponding information and deal with it according to the specific circumstances.

(9) Move to the queue header. The system search the hash value. If it is in the ILUR2 queue, it will be moved to the queue header.

Experimental results. It was found that the hit rate of the 0 sequences is a hash value of the common resource. We find that except the beginning of the hash table, we have had a miss. The later hit rate reached 100%, and IL-RU algorithm can keep a high hit rate. We can see the hit situation schematic diagram of ILRU algorithm in the table.

Compared with the traditional LRU algorithm, the ILRU algorithm is higher. The hit rates of the ILRU algorithm and traditional LRU algorithm are shown in Fig. 5.

In the previous test, the ratio of the cache was 50%. In order to test the effect of the cache size, it was tested in four groups. The results are shown in Fig. 6. The percentage of the cache in the test was 0, 25, 50 and 75%, respectively. The occupancy rate is 0, which indicates that the RAM is all ILRU cache zone. The reason why there is no test 100% is that the RAM is written in the buffer.

As we can see from the Fig. 6, when the proportion of cache increases from 0 to a certain range, the cache system to write the most frequent data. This part of the number of data is written in a very many times higher than others in the RAM, so the performance of the system will increase.

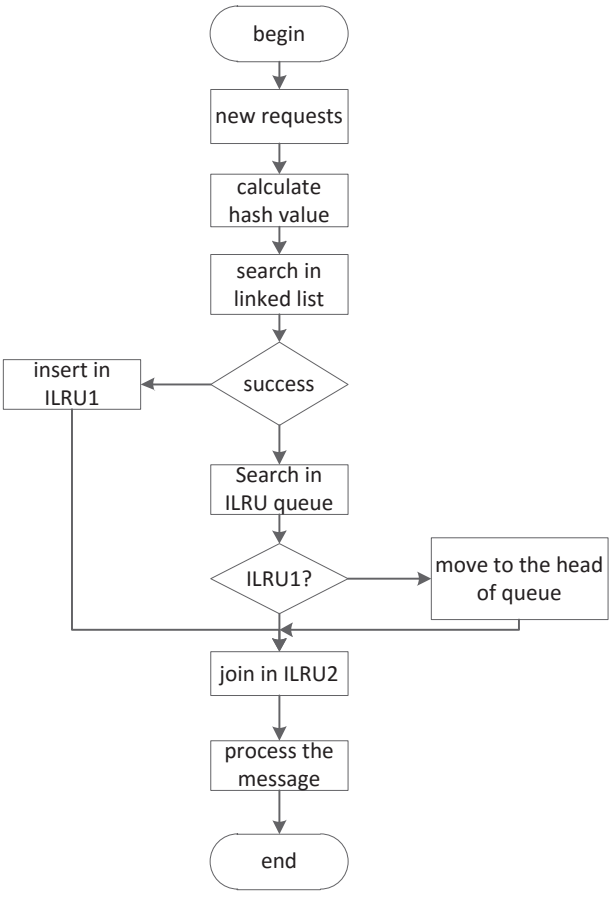

Fig.4. The flow chart of ILRU

ISSN 2071-2227, Науковий вісник НГУ, 2015, № 6

Table

Comparisons of PSNR Parameters

Access hash sequence	7	0	1	2	0	3	0	4
	7	0	1	2	2	3	3	4
ILRU1		7	0	1	1	2	2	3
			7	0				
					0	0	0	0
S NR								
	×	×	×	×	∨	×	∨	×

The performance of the system decreased when the ratio is from 50 to 75%. This is because the ILRU cache decreases, which affects the system's hit rate, and the write cache already has much write frequent data.

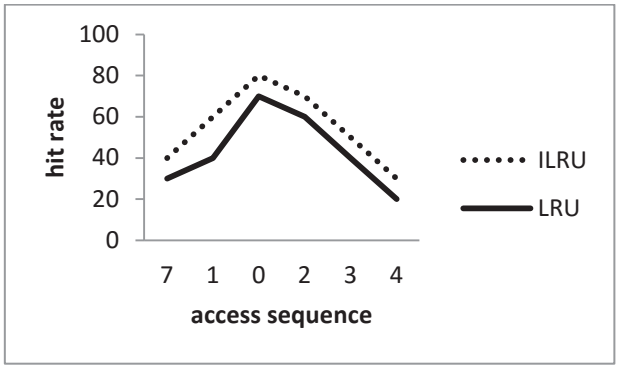

Fig.5. The hit rate of ILRU algorithm and traditional LRU algorithm

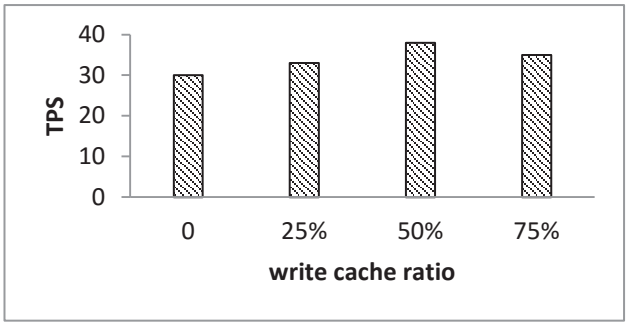

Fig.6. Different cache size effect on the performance

Conclusions. When there are more users to access an unfamiliar page suddenly, the system will produce a higher miss rate. This is why it is necessary to improve the LRU algorithm. In this research the ILRU algorithm has been developed, which can increase the hit rate when more users suddenly have a higher access to an unfamiliar page. We determine whether the hash value is added to a page or not through searching the access hash value in the LRU queue. The experimental results show that the algorithm can achieve the goal of the system performance improvement.

Acknowledgements. This work was supported by project 2015LGYB28 of Jiujiang University.

References / Список літератури

1. Gantz, J. and Reinsel, D. (2011), "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east", *IDC iView: IDC Analyze the Future*, pp. 1–16.

2. Xu, Z., Ning, W., Vassilios, G.V. and Michael, P.H. (2015), "A distributed in-network caching scheme for P2P-like content chunk delivery", *Computer Networks*, vol. 91, no.14, pp. 577–592.

3. Hamilton, T., Brian, D., Jules, W., Russell, K., Jonathan, P. and Douglas, C.S. (2014), "Aniruddha Gokhale. DRE system performance optimization with the SMACK cache efficiency metric", *Journal of Systems and Software*, vol. 98, pp. 25–43.

4. P. Julian B. and F. Sagayaraj F. (2015), "Improving the performance of a proxy cache using very fast decision tree classifier", *Procedia Computer Science*, vol. 48, pp. 304–312.

5. Nicaise C.F., Philippe, N., Giovanni, N. and Don, T. (2014), "Performance evaluation of hierarchical TTL-based cache networks", *Computer Networks*, vol. 48, pp. 304–312.

6. O'neil, E.J., O'neil, P.E. and Weikum, G. (2012), "The LRU-K page replacement algorithm for database disk buffering", *Proc. of the Conf. on ACM SIGMOD Record*, NY, USA, pp. 297–306.

7. Shasha, D. and Johnson, T. (2010), "2Q: A low overhead high performance buffer management replacement algoritm", *Proc. of the 20th International Conference on Very Large Databases*. Copenhagen, Danmark, pp. 439–450.

8. Wenjia, N., Gang, L., Endong, T., Xinghua, Y., Liang, C., Zhong, Z.S. and Song, C. (2014), "Interaction relationships of caches in agent-based HD video surveillance: Discovery and utilization", *Journal of Network and Computer Applications*, vol. 37, pp. 155–169.

9. Jiang, S., Ding, X. and Chen, F. (2006), "DULO: an effective buffer cache management scheme to exploit both temporal and spatial locality", *Proc. of the 4th USENIX Conference on File and Storage Technologies*. California, USA, pp. 8–17.

Мета. Останнім часом ми все частіше чуємо про супермасиви даних (даних великого об'єму). Як у ділових, так і в особистих цілях користувачам необхідно зберігати багато важливої інформації. При використанні системи зберігання даних ми хочемо, аби система відповідала на запит досить швидко, без затримок. Для системи зберігання даних – це дуже складне завдання. До теперішнього часу вчені провели дуже багато досліджень у цьому напрямі та з'ясували, що використання кеш (швидкодіючої буферної пам'яті) у системі збері-

гання даних дозволяє значно поліпшити її продуктивність.

Методика. У сфері оперативного зберігання інформації велика увага приділяється дослідженню алгоритмів кешування. Найбільш популярним є алгоритм витіснення на основі найбільш давнього використання (LRU – Least Recently Used).

Результати. Оскільки нове значення хеш-функції, що переміщується на вершину стека, вимагає внести зміни до стека, навіть якщо відвідувана сторінка вже знаходиться в пам'яті, це займає багато часу. Для поліпшення продуктивності необхідний покращений алгоритм кешування.

Наукова новизна. Розроблена нова стратегія заміщення на основі алгоритму LRU, названа покращеним алгоритмом ILRU. Покращений алгоритм може підвищити частоту успішних звернень, коли користувачі різко підвищують кількість звернень до незнайомої сторінки. Алгоритм визначає, чи привласнене значення хеш-функції сторінці шляхом пошуку у LRU черзі.

Практична значимість. Результати тестування показали, що покращений алгоритм ILRU підвищує продуктивність у порівнянні з традиційним алгоритмом LRU. До того ж, покращений алгоритм ILRU має вищу частоту успішних звернень, ніж алгоритм типу "першим прийшов − першим вийшов".

Ключові слова: *стратегія заміщення, алгоритм LRU, покращений алгоритм ILRU, частота успішних звернень, продуктивність*

Цель. В последнее время мы все чаще слышим о супермассивах данных (данных большого объёма). Как в деловых, так и в личных целях пользователям необходимо хранить много важной информации. При использовании системы хранения данных мы хотим, чтобы система отвечала на запрос достаточно быстро, без задержек. Для системы хранения данных – это очень сложная задача. До настоящего времени ученые провели очень много исследований в этом направлении и выяснили, что использование кэш (быстродействующей буферной памяти) в системе хранения данных позволяет значительно улучшить ее производительность.

Методика. В сфере оперативного хранения информации большое внимание уделяется исследованию алгоритмов кэширования. Наиболее популярным является алгоритм вытеснения на основе наиболее давнего использования (*LRU – Least Recently Used*).

Результаты. Поскольку новое значение хэш-функции, перемещающееся на вершину стека, требует внести изменения в стек, даже если посещаемая страница уже находится в памяти, на это уходит много времени. Для улучшения производительности необходим улучшенный алгоритм кэширования.

Научная новизна. Разработана новая стратегия замещения на основе алгоритма LRU, называемая улучшенным алгоритмом ILRU. Улучшенный алгоритм может повысить частоту успешных обращений, когда пользователи резко повышают количество обращений к незнакомой странице. Алгоритм определяет, присвоено ли значение хэш-функции странице путем поиска в LRU очереди.

Практическая значимость. Результаты тестирования показали, что улучшенный алгоритм ILRU повышает производительность по сравнению с традиционным алгоритмом LRU. К тому же, улучшенный алгоритм ILRU имеет более высокую частоту успешных обращений, чем алгоритм типа "первым пришёл − первым вышел".

Ключевые слова: *стратегия замещения, алгоритм LRU, улучшенный алгоритм ILRU, частота успешных обращений, производительность*

Рекомендовано до публікації докт. техн. наук В.В. Гнатушенком. Дата надходження рукопису 20.11.14.