The Framework for Study of Caching Algorithm Efficiency.

© Thanh Hung Ngo
Don State Technical University
nthungla@yahoo.com

Mosab Bassam Y. Al Zgool
Don State Technical University
mosab2000@yahoo.com

PhD supervisor: Michael V. Grankov

Abstract.

In this paper we offer several models of reference sequences (traces of references) using Markov chains for testing of the replacement policies in caching systems. These models enable the generations of traces with the repeated subsequences of references, which are of great interest for study of forecasting methods in caching systems. Furthermore, we offer the scheme of the program stand, where these models have been realized, and result of the experiments, which have been carried out with its help.

Index terms: program stand, model of reference sequences, Markov chains, traces with repeated subsequences.

1 Introduction

The most popular method to study the replacement policies in caching systems is its testing by the program simulating caching system. In this approach different traces are given to the input of the program and the cache-hit rate is registered as the result provided cache-system. The more the value of this rate is, the more efficient the investigated policy performs.

Specific characters of the functioning of the information system often render a big influence upon behavior of the reference sequences. So testing the policy, have been specially designed for a certain information system, requires the traces, reflecting specifics of the system. There are two ways of achieving the reference sequences: by means of logging the accesses being executed in the system for a long-lasting time (e.g. collecting reference sequences) [1, 2] and by means of trace generating programs [3, 4]. The traces having been achieved using the first method more really reflect the specifics of the system, but their achievement requires a significant computing and material resources. The achievement of traces using program-generator is faster and cheaper. The main problem of the last method is the development of mathematical model, describing the information system.

Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2008

In this paper we offer several models of reference sequences using Markov chains [5]. Besides, we offer the scheme of the program stand, where these models have been realized, and the result of the experiments, which have been carried out with its help.

2 Models of reference sequences

2.1 Model with One Markov Chain (*MOMC*)

The model is widely used for modeling of different queuing systems, including caching system. *MOMC* is specified by the triple (S, A, π), where:

1. $S = \{s_1, s_2, \dots, s_N\}$ - the set of N objects in the system. Objects may be the papers in paper caching systems or the objects in the object caching systems;
2. $A = \{a_{ij}\}$ - the object transition probabilities. The value of element a_{ij} of matrix equals to the probability of event, when the system, having accessed the object s_i, accesses the object s_j. If mark the object having been accessed at the moment t as q_t, then:

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i),$$

$$a_{ij} \geq 0, \sum_{j=1}^{N} a_{ij} = 1, 1 \leq i, j \leq N \;;$$

3. $\pi = (\pi_1, \pi_2, \dots, \pi_N)$ - the initial object distribution.

The generation of traces by *MOMC* follows the algorithm:

1. Input S, A, π.
2. Input the count of references L (length of trace).
3. Initialize value for the variable O, which is the string logging the references to the object, as an empty string.
4. Initialize value for the variable t, which is the current count of references:

$$t = 1.$$

5. Choose the initial object q_t from the set S according to the initial object distribution π.
6. WHILE $t \leq L$

7. Let $q_t = s_k$, append s_k to the string O:

$$O = O + s_k .$$

8. Choose the next object according to the matrix of object transition probabilities A and the current object s_k.

9. Increase the value of the current count of references t by one:

$$t = t + 1 .$$

10. END-WHILE.

11. Save the achieved reference string O as the desired trace.

Special occasions of implementing *MOMC.* Let's consider the following realizations of *MOMC*:

1. $a_{ij} = \frac{1}{N}, \forall i, j : 1 \le i, j \le N$.
In this case we achieve the model describing the system where the accesses to the objects obey the uniform law.

2. The absolute cyclic traces (*ACT*).
Generation of the *ACT* is made by picking up the matrix of transition probabilities as in table 1.

i\j	1	2	3	4
1	0	0	0	1
2	0	0	1	0
3	1	0	0	0
4	0	1	0	0

Table 1: matrix of cyclic transitions.

A fragment of *ACT*, achieved by using the matrix in table 1, looks like: …, 2, 3, 1, 4, 2, 3, 1, 4, …. The *ACT* is of interest to study the anomalies such as Belady's anomaly.

3. The cyclic traces with random "noise" (*CTRN*).
Replacements of one or more unit element (the element with value equaling to 1) of matrix in table 1 by the value $e^{-\lambda x}$, and zero elements in the corresponding rows by the value $\left((1 - e^{-\lambda x})/(N-1)\right)$ result in inclusion of a random "noise" in cyclic traces ($0 \le x \le \frac{\ln(N)}{\lambda}$ and $0 < \lambda$). When $x = 0$, we achieve the *ACT* as in occasion 2. Increasing the value of x, the indeterminacy in behavior of traces increase. When $x = \frac{\ln(N)}{\lambda}$, we achieve the traces, in which references to the objects obey the uniform law (as in occasion 1). Thus the first and the second occasions are just the special cases of the third occasion.
For example we have brought into the matrix in table 1 the noise with the following parameters $(\lambda = 7, x = 0.0198)$ and have generated traces with its help. The matrix of transition probabilities now looks like in table 2. Some of generated traces are as follow:
(1, 4, 2, 3, 4, 2, 3, 4, 4, 2, 3, 1, 4, 2, 4);
(2, 3, 1, 2, 2, 3, 1, 4, 3, 1, 4, 2, 4, 1, 4).

i\j	1	2	3	4
1	0.043	0.043	0.043	0.871
2	0.043	0.043	0.871	0.043
3	0.871	0.043	0.043	0.043
4	0.043	0.871	0.043	0.043

Table 2: matrix of cyclic transitions with random "noise".

Because of one's simplicity the *MOMC* is not able to describe the functioning principles of the real systems. To specify the more complicated mechanisms it is used the hidden Markov model.

2.2 Model with Hidden Markov Chain (*MHMC*)

MHMC is specified by the quintuple (S, V, A, B, π), where:

1. $S = \{s_1, s_2, \dots, s_U\}$ - the set from U users of the information system;

2. $A = \{a_{ij}\}$ - the user transition probabilities. If mark the user, which has got access to the system at the moment t, by q_t, then:

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i),$$

$$a_{ij} \ge 0, \sum_{j=1}^{U} a_{ij} = 1, 1 \le i, j \le U ;$$

3. $V = \{v_1, v_2, \dots, v_N\}$ - the set from N objects of the system;

4. $B = \{b_{ij}\}$ - probabilities of choosing object for the users. The i-th row is the object choice probability distribution for the i-th user. The value of element b_{ij} equals to the probability of event, when the current user $q_t = s_i$ chooses the object v_j. If mark this object as o_t, then:

$$b_{ij} = P(o_t = v_j \mid q_t = s_i),$$

$$b_{ij} \ge 0, \sum_{j=1}^{N} b_{ij} = 1, 1 \le i \le U, 1 \le j \le N ;$$

5. $\pi = (\pi_1, \pi_2, \dots, \pi_N)$ - the initial user distribution.

The generation of traces by *MHMC* follows the algorithm:

1. Input S, V, A, B, π.

2. Input the count of references L (length of trace).

3. Initialize value for the variable O, which is the string logging the references to the object, as an empty string.

4. Initialize value for the variable t, which is the current count of references: $t = 1$.

5. Choose the initial user q_t from the set S according to the initial user distribution π.

6. WHILE $t \le L$

7. Let $q_t = s_k$.

8. Choose the current object o_t according to the matrix B and the current user s_k.

9. Let $o_t = v_l$, append v_l to the string O:

$$O = O + v_l .$$

10. Choose the next user according to the matrix of user transition probabilities A and the current user s_k.

11. Increase the value of the current count of references t by one: $t = t + 1$.

12. END-WHILE.

13. Save the achieved reference string O as the desired trace.

The second model more naturally reflects the relationship between the elements of the information system: each user accesses to the objects due to his logic (his distribution law).

Both of the described models have disadvantages. The first one is not taken into account the presence of the users, and the second one is not able to model the cyclic traces. For elimination of these restrictions we offer the two-level model of Markov chains.

2.3 Two-Level Model of Markov Chains (*TLMMC*)

TLMMC is specified by the six $(S, V, A_S, A_V, \pi_S, \pi_V)$, where:

1. $S = \{s_1, s_2, \ldots, s_U\}$ - the set from U users of the information system;

2. $V = \{v_1, v_2, \ldots, v_N\}$ - the set from N objects of the system;

3. $A_S = \{a_{ij}\}$ - the user transition probabilities. If mark the user, which has got access to the system at the moment t, by q_t, then:

$$a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i),$$

$$a_{ij} \geq 0, \sum_{j=1}^{U} a_{ij} = 1, 1 \leq i, j \leq U ;$$

4. $A_V = (A^1, A^2, \ldots, A^U)$ - matrix of object transition probabilities for the users, where:

$A^k = \{a^k_{ij}\}$ - matrix of object transition probabilities for the k-th user. Value of the element a^k_{ij} equals to the probability of event, when the k-th user, having chosen the i-th object, chooses the j-th object. If mark the object, being chosen by the k-th user at his r-th choice, as o_r, then

$$a^k_{ij} = P(o_{r+1} = v_j \mid ((o_r = v_i) \wedge (q_t = s_k))),$$

$$a^k_{ij} \geq 0, \sum_{j=1}^{N} a^k_{ij} = 1, 1 \leq i, j \leq N, 1 \leq k \leq U ;$$

5. $\pi_S = (\pi_1, \pi_2, \ldots, \pi_U)$ - the initial user distribution;

6. $\pi_V = (\pi^1, \pi^2, \ldots, \pi^U)$ - the initial object distributions for the users, where:

$\pi^k = (\pi^k_1, \pi^k_2, \ldots, \pi^k_N)$ - the initial object distribution for the k-th user. $1 \leq k \leq U$.

The generation of traces by *MHMC* follows the algorithm:

```
1. Input: $S$, $V$, $A_S$, $A^1$, $A^2$, …, $A^U$, and $\pi_S, \pi^1, \pi^2, \ldots, \pi^U$.
2. Input the count of references $L$ (length of trace).
3. Initialize value for the variable $O$, which is the string logging the references to the object, as an empty string.
4. Initialize value for the variable $l$, which is the current count of references: $l = 1$.
5. Initialize value for the variable $t$, which is the count of the user turns: $t = 1$.
6. Choose the initial user $q_t$ from the set $S$ according to the initial user distribution $\pi_S$.
7. WHILE $l \leq L$
8.     Let $q_t = s_k$.
9.     Initialize value for the variable $r$, which is the session count of references for the current user: $r = 1$.
10.    Choose the initial object $o_r$ according to the distribution $\pi^k$.
11.    Generate the number $z$, which is the count of references will be made by the current user.
12.    WHILE $(l \leq L)$ and $(r \leq z)$
13.        Let $o_r = v_m$, append $v_m$ to reference string: $O = O + v_m$.
14.        Increase the value of $l$ by one: $l = l + 1$.
15.        Choose the next object $o_{r+1}$ according to the matrix $A^k$ and the current object $o_r$.
16.        Increase the value of $r$ by one: $r = r + 1$.
17.    END-WHILE
18.    Choose the next user $q_{t+1}$ according to matrix $A_S$ and the current user $q_t = s_k$.
19.    Increase the value of $t$ by one: $t = t + 1$.
20. END-WHILE.
21. Save the achieved reference string $O$ as the desired trace.
```

Let's consider an example illustrating the usage of *TLMMC*. Set value for the parameters of *TLMMC* as below:

$$S = \{1, 2\}; V = \{1, 2, 3, 4\}; \pi_S = (0.5, 0.5);$$

$$\pi^1 = (0.5, 0.5, 0.5, 0.5); \pi^2 = (0.5, 0.5, 0.5, 0.5);$$

$$A_S = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}; \quad A^1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix};$$

$$A^2 = \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

One of the traces achieved with help of the model looks like: 4, 2, 3, 1, 3, 1, 4, 2, 4, 2, 3, 1, 1, 2, 2, 3, 1, 4, 1, 1, 2, 3, 1, 4, 2, 1, 4, 2. In this trace the absolute cyclic trace (…, 1, 4, 2, 3, 1, 4, 2, 3, …) has been distorted and fragmented into subsequences due to the random turns of the users and the random number of references having been made in their turns. These distortions create more difficulties for the methods identifying the subsequences in the references sequence. So the model *TLMMC* is of great interest to investigate the prediction methods in caching system [6, 7].

3 The scheme of program stand

The program stand has been written in programming environment Delphi 7. It has following components (fig. 1):

1. Trace - generator.
   Generate traces by the one of different models.
2. Realization block of replacement policies.
3. Cache simulator.
   Simulate the performance of cache-system with the chosen policy and chosen traces model.
4. Analysis block
   Analyze the result of performance of cache-system.

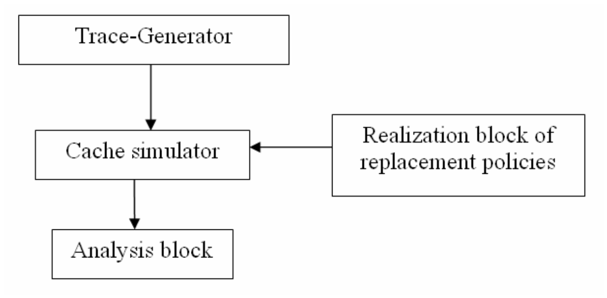

Figure 1: The components of the program stand.

4 Experiments

Functioning of the program stand has been illustrated by benchmark analysis of performance of different replacement policies: Random, LRU and LFU. The result of the experiment is showed in fig. 2.

In the investigation it has been used the model *TLMMC* with the following parameters:

$$S = \{1, 2\}; V = \{1, 2, \ldots, 100\}; \pi_S = (.5, .5);$$
$$\pi^1 = (.01, .01, \ldots, .01); \pi^2 = (.01, .01, \ldots, .01);$$

$$A_S = \begin{pmatrix} .5 & .5 \\ .5 & .5 \end{pmatrix};$$

Matrix A^1 is specified as matrix of cyclic traces with a random "noise" (looks like the matrix in tab. 2). When $x = 0$ the absolute cyclic trace, being generated according to this matrix, looks like:

$$\begin{pmatrix} 1,2,3,4,87,35,7,99,9,76,13,12,11,14,15,34,78, \\ 30,74,20,29,5,23,64,80,100,79,28,21,53,6,26, \\ 33,32,44,17,81,65,51,73,41,42,43,31,48,46, \\ 47,22,49,45,96,52,18,54,55,94,19,60,90,25, \\ 92,62,16,24,40,63,67,68,69,75,71,72,56,61, \\ 70,10,77,86,38,58,37,82,83,84,27,36,50,88, \\ 89,59,57,91,93,39,95,85,97,98,8,66 \end{pmatrix}$$

Matrix A^2 determines for the second user the equiprobable transition from an object to any objects, including the transition to it self, i.e. $a_{ij}^2 = 1/100, \forall i, j: \quad 1 \le i, j \le 100$.

As magnitude of the "noise" coefficient it has been used three values: $x = 0$; $x = 0.1054$ and $x = 4.6052$. In all cases $\lambda = 1$. When $x = 0$, in generated traces present the repeated subsequences. This fact has negatively influenced upon performance of Random and LRU, but not for LFU. In this case it is registered the slight advantage of Random over LRU and significant advantage of LFU over the rest. Increasing the magnitude of x, the indeterminacy in generation of the traces rapidly increases. This change positively influences upon performance of Random and LRU, and equalizes the performance of all policies.

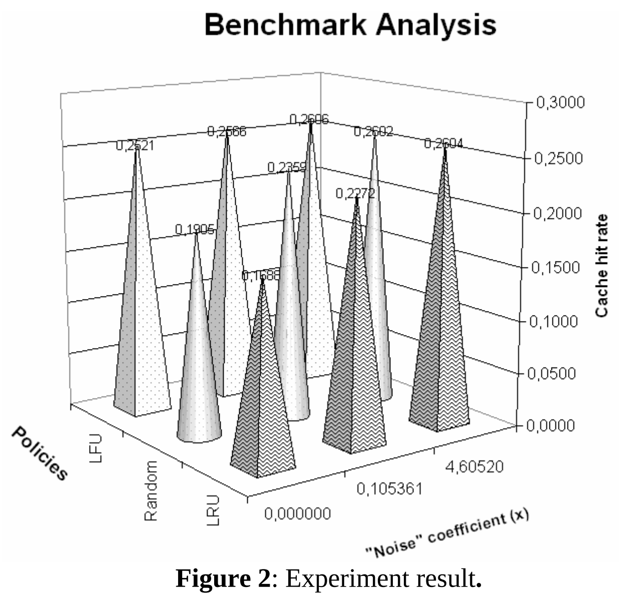

Figure 2: Experiment result.

5 Conclusions

The result of the demonstration experiment shows urgency and relevance of the offered traces models and also program stand for study of different replacement policies. The *TLMMC* presents a great interest to study the forecasting method in cache-system. The traces generated by this model were applied to testing the decomposition method of the relations in database systems with the criteria of increasing the cache-hit rate [8]. Herein, when the magnitude of x reaches its maximum value, the experimental cache-hit rate completely agrees with the corresponding rate theoretically has been achieved in that paper.

References

[1] Min Xu, Vyacheslav Malyugin, Jeffrey Sheldon, Ganesh Venkitachalam, Boris Weissman. ReTrace: Collecting Execution Trace with Virtual Machine Deterministic Replay. // Third Annual Workshop on Modeling, Benchmarking and Simulation, held in conjunction with the 34th Annual International Symposium on Computer Architecture, June 2007. http://www.xhfamily.com/x/files/MoBS07_replay_itrace.pdf

[2] Hervé Touati, Alan Jay Smith. Reducing and Manipulating Complex Trace Data. // Software - Practice and Experience. Vol. 21, June 1991, 639–655.

[3] Eeckhout L., De Bosschere K., Neefs H. Performance analysis through synthetic trace generation. // Performance Analysis of Systems and Software, 2000. ISPASS. IEEE International Symposium on Volume, Issue, 2000.

[4] Germán Galeano Gil, Juan A. Gómez Pulido, and Juan M. Sánchez Pérez. Tool for the Analysis and Memory-Trace Generation of DOS Executable Files. // Microprocessors and Microsystems, Volume 22, Issue 7, 25 January 1999, Pages 389-393.

[5] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. // Proceedings of the IEEE, VOL. 77, NO. 2, Feb 1989.

[6] Fei Guo and Yan Solihin. A Prediction Model for Alternative Cache Replacement Policies. // http://www.ece.ncsu.edu

[7] Thomas M. Kroeger, Darrell D. E. Long. Predicting File System Actions from Prior Events. // Proceedings of the USENIX 1996 Annual Technical Conference, pages 319-328, January 1996. http://citeseer.ist.psu.edu/kroeger96predicting.html

[8] Thanh Hung Ngo, Michael V. Grankov. New Object Function for Vertical Partitioning in Database System. // the present colloquium.