Acta Technica *62* (2017), No. 4A, 753–762

An improved strategy of map cache prefetching

Hongxin Ma[2,3], Hui Qi[2,5], Xiaoqiang Di[2], Jinqing Li[2], Fengrong Zhang[4]

Abstract. Map cache system has a good effect on accelerating the response speed of the dynamic vehicle navigation system and reducing the amount of network transmission data, while the cache prefetch can further improve the performance of the cache system. In this paper, we deeply study the cache prefetch strategy and propose the heuristic map tiles prefetch strategy based on road network analysis, which first analyzes the front road network where the vehicle will travel and identifies the front intersection so as to predict the trajectory of the vehicle, and then combines the result and that predicted by heuristic prefetch strategy to produce more accurate prediction. The experiments show that the prefetch strategy based on road network analysis can further improve the performance of the cache system than the pure heuristic prefecth strategy, significantly reducing the amount of map tiles needed by the vehicle navigation system.

Key words. Map tiles, vehicle navigation system, cache prefetch, heuristic cache prefetch.

1. Introduction

Based on the wireless communication network, the dynamic vehicle navigation system realizes the information exchange and sharing between the vehicle and the vehicle, the vehicle and the road as well as the vehicle and the traffic information center, which can provide more services, reduce service costs and improve the user experience and traffic conditions [1,2]. However, the performance of wireless network influences and restricts the development of dynamic vehicle navigation system. In practical applications, the 4G LTE network can provide downlink rates of 30.8Mbps, uplink rates of 19.4Mbps, and TCP handshake delay of 127ms at 95% of cases [3].

[1] Acknowledgement-This work is supported in part by the National Social Science Fund of China under Grant No.17BSH135.

[2] Workshop 1 - School of Computer Science and Technology, Changchun University of Science and Technology

[3] Workshop 2 - Training Department, Aviation University Air Force, Changchun, China

[4] Workshop 3 - Northeast Normal University,Changchun, China.

[5] Corresponding author:Hui Qi; e-mail: qihui@cust.edu.cn

In contrast, the flash memory commonly used in mobile devices has a write delay of less than 1 ms, a read delay of less than 0.3ms and a bandwidth of more than 2GB/s[4]. Thus, the performance of today's wireless networks is far lower than that of local storage. According to the principle of memory architecture, local storage is usually used as a cache for other computers on the network.

Literature [5] designed a cache system for dynamic navigation systems. It helps to improve the map display at the client, but it won't be helpful enough if it caches only the map tiles needed every time. The map display will be improved more significantly if the cache system can forecast the map tiles in possible demand next time and cache them in advance [6].

Suppose the vehicle stands in the tile a_{ij} at the time t. The tiles to be cached in the first option will be:

$$A_1^{(t)} = \begin{pmatrix} a_{(i+2)(j-2)} & \cdots & a_{(i+2)(j+2)} \\ \vdots & \ddots & \vdots \\ a_{(i-2)(j-2)} & \cdots & a_{(i-2)(j+2)} \end{pmatrix}$$

The tiles to be cached in the second option will be:

$$A_2^{(t)} = \begin{pmatrix} a_{(i+1)(j-1)} & \cdots & a_{(i+1)(j+1)} \\ \vdots & \ddots & \vdots \\ a_{(i-1)(j-1)} & \cdots & a_{(i-1)(j+1)} \end{pmatrix}$$

Suppose the vehicle moves from the tile a_{ij} to the tile $a_{i(j+1)}$ at a future moment t$'$. The tiles for display ($A_d^{(t')}$) are all contained in $A_1^{(t)}$. In the first option, the navigation system can display the map without waiting for the cache system to fetch the $A_2^{(t')}$. However, in the second option, the navigation system cannot display the map before $A_2^{(t')}$is fetched by the cache system, as not all tiles of the $A_d^{(t')}$ are contained in $A_2^{(t)}$. Therefore, the cache system with prefetch will perform better than the cache system withoutprefetch (cache on demand).

Now suppose there is another cache design (the third option), a cache system that has an ideal forecasting capacity and caches the current and next required tiles every time. According to this hypothesis, in the third option, the tiles to be cached at t will be:

$$A_3^{(t)} = A_d^{(t)} \cup A_d^{(t')} \begin{pmatrix} a_{(i+1)(j-1)} & \cdots & a_{(i+1)(j+1)} \\ \vdots & \ddots & \vdots \\ a_{(i-1)(j-1)} & \cdots & a_{(i-1)(j+1)} \end{pmatrix}$$

In summary, the prefetch option with accurate forecasting is the best. As such, how to improve the forecast accuracy is the focus of prefetch strategy research and this paper. Next, a survey of map tiles prefetch strategies will be introduced. Then the prefetch strategy proposed by this paper, a heuristic prefetch strategy based on road network analysis, will be described and finally verified through a series of tests.

2. Related work

In [8], a location-aware prefetching mechanism was proposed, which was intended for European waterway network system. Since the users of that system often travel on main rivers, their routes can be forecast by means of river information as well as ship speed and direction. In [9], a prefetch algorithm based on Hilbert curves was proposed. It maps two-dimensional Euclidean distances into one-dimensional space via Hilbert curves, thus reducing the prefetched data to some extent. This algorithm is based on the fact that the user access pattern has a spatial locality, that is, it is more possible for the user to request the objects spatially adjacent to the currently accessed object. The forecasting of user behavior is completely based on spatial distance. The objects closer to the currently accessed object are more possible to get access to. For the objects with the same distances, correct forecasting is impossible. In [10], two prefetch approaches were recommended. The first approach is probability-based to forecast the tiles for possible future access by considering the probability of transition between two neighboring tiles and the position of screen center in the current tile. The second approach is based on the movement features of the first k steps. It takes the movement sequence of the first k steps as the current state and models the state transition through the Markov chain in order to predict the tiles for next access. The two approaches are probability-based in nature needing a large amount of data statistics to determine the transition probability, and therefore more suitable for server caching. The [11] improved the probability-based prefetch approach recommended in [10], and presented a Markov prefetch model based on Zipf distribution. The transition probability matrix in that model is no longer targeted at all the tiles, but focused on the tiles with frequent user access in line with the characteristics of Zipf distribution. Thus it has effectively downsized the transition probability matrix and reduced the storage space. In [10], a heuristic prefetch strategy was presented to predict the tiles for future request in accordance with the movement features and cache hit ratio of the first k steps. Compared with the second approach in [10], this strategy is more time-related and more suitable for regular navigation application.

The above approaches, except for that in [8], take no account of the geographic background of map data. Even in [8], only the river information has been considered. But there is another approach, other than the above ones. According to that approach, geographic background information has driven the user's access to map data and thus shall be taken into consideration in forecasting the data access behavior. In , a prediction model was provided to determine hot spots in the map. That model mainly uses geographic background information, including the information on residential area, main traffic, coastline and points of interest, to output a Boolean vector set. In the approach in was improved by determining the access probability of map tiles through an ordinary least square regression model, which can automatically decide on useful geographic background according to the information on user access. In , the model in was improved into a model with an artificial neural network that can also be used for cache replacement . The above prefetch methods based on geographic background are usually used for server caching and also suitable for

clients' map roaming.

3. Heuristic prefetch strategy

The prefetch strategy proposed by this paper is based on the strategy in [12]. To better describe this strategy, the following variables shall be defined:

d: Capacity of historical movement list. During each movement into a tile, the algorithm records the row and column indexes of the tile on a first-in-first-out (FIFO) linked list.

h_e: Linked list of column indexes. Whenever the vehicle enters a tile, the column index of the tile shall be stored into the linked list. If the capacity of the linked list is d before the data insertion, the new element shall be inserted at the tail after the deletion of the head element.

h_s: Linked list of row indexes.

hitRatio: Linked list of cache hit ratios. This list records the cache hit ratio for every movement.

With the above variables, the following variables can be calculated:

$$\text{Easting} = \int_{i=1}^{d-1} \left(h_e\left[i+1\right] - h_e\left[i\right]\right) \times hitRatio[i+1]$$

$$\text{Southing} = \int_{i=1}^{d-1} \left(h_s\left[i+1\right] - h_s\left[i\right]\right) \times hitRato[i+1]$$

Two types of information are contained in Easting and Southing respectively: prediction accuracy and stability of Easting or Southing movement in the first d-1 steps. Based on the values of Easting and Southing, the next movement direction can be forecast. It can be found through observation that the values of Easting and Southing have something to do with the value of d. If the movement in the first d-1 steps is stable, the values of Easting and Southing will increase with the value of d. This will bring some trouble to prediction, as the thresholds of Easting and Southing may vary with the value of d. Therefore, Easting and Southing shall be normalized and rewritten as follows:

$$\text{Easting} = \frac{\int_{i=1}^{d-1} \left(h_e\left[i+1\right] - h_e\left[i\right]\right) \times hitRatio[i+1]}{\int_{i=1}^{d-1} \left(h_e\left[i+1\right] - h_e\left[i\right]\right)}$$

$$\text{Southing} = \frac{\int_{i=1}^{d-1} \left(h_s\left[i+1\right] - h_s\left[i\right]\right) \times hitRatio[i+1]}{\int_{i=1}^{d-1} \left(h_s\left[i+1\right] - h_s\left[i\right]\right)}$$

If the denominator is 0, this equation will be false so that Easting or Southing can be directly equal to 0. If Easting ≥ 0.5, the vehicle will move eastward into the right tile. If Easting < -0.5, the vehicle will move westward into the left tile. If $-0.5 \leq$ Easting < 0.5, the vehicle will move neither eastward nor westward and the tile column index to enter will remain unchanged. Similarly, if Southing ≥ 0.5,

the vehicle will move southward into the tile below. If Southing < -0.5, the vehicle will move northward into the tile above. If $-0.5 \leq$ Southing < 0.5, the vehicle will move neither southward nor northward and the tile row index to enter will remain unchanged.

After the tile to enter is forecast, the tile to be prefetched can be determined by the following rule:

Suppose the tile set cached at t is $S^{(t)}$ and the central tile is a_{ij}. Based on the values of Easting and Southing, the row and column indexes of the central tile (possibly $a_{ij'}$ or $a_{i'j}$or $a_{i'j'}$) at t$'$ can be predicted as i$'$ $= i - \lfloor Southing + 0.5 \rfloor$ and j$'$= j+ $\lfloor Easting + 0.5 \rfloor$ respectively. Once the central tile at t$'$ is determined (forecast), the tile set needed by the system at t$'$ will be:

The tile set to be prefetched at t will be $A_d^{(t')} - S^{(t)}$.

4. Heuristic prefetch strategy based on road network analysis

4.1. Prefetch algorithm

The prefetch strategy in [12] considers only historical movement and historical prediction, imposing no restrictions on movement conditions. This strategy can achieve high forecast accuracy when the user moves regularly (rather than totally random map roaming). If the user moves regularly under limited conditions, the consideration of such conditions can further improve the forecast accuracy.

The application of onboard navigation is a good fit for the above hypothesis. The navigated vehicle usually moves along the road. If there is no road in a tile, the movement from a neighboring tile to this tile won't occur. If there is no road between the tile a_{ij} and its neighbor $a_{i'j'}$, the movement from a_{ij} to $a_{i'j'}$ won't occur. It can be seen that the user's movement under navigation is limited by road network. Therefore, adding road network analysis to heuristic prefetch strategy will be helpful for improving the prefetch performance.

4.2. Prefetch algorithm

1. Determine the row and column indexes i and j of the tile where the vehicle is located at the time t, in order to obtain the road network data for that tile.

2. Determine the starting point and direction of road network search in accordance with the current road information and the driving direction, and search the road network data of the current tile for intersections.

3. If no intersection is found, the row and column indexes i$'$ and j$'$ of the tile to be entered at t$'$ (t$'$ $> t$) can be determined through searching along the road and then the tile set needed at t$'$ can be forecast, as shown below:

   $A_d^{(t')} = \{a_{xy} | i' - 1 \leq x \leq i' + 1, j' - 1 \leq y \leq j' + 1$

4. Otherwise (an intersection is found), the tile set to be entered at t$'$ can be

determined through searching along the road behind the intersection to forecast the needed tile set $A_{d1}^{(t')}$. On the other hand, the row and column indexes i′ and j′ of the central tile can be forecast according to the values of Easting and Southing, and the needed tile set $A_{d2}^{(t')}$ can be calculated as per the equation (1). The two tile sets can be combined as: $A_d^{(t')} = A_{d1}^{(t')} \cap A_{d2}^{(t')}$.

It is at the step 4 that heuristic prefetch strategy is applied to the above algorithm. That is to say, this strategy is needed only when an intersection is found, otherwise the central tile at t′ can be determined by searching along the road. Furthermore, the step 4 is not to directly use the heuristic prefetch, but to combine the forecast results of heuristic prefetch and road network search. The key of this algorithm is road network search, which is also a difficulty due to the complexity of road network structure, especially in terms of intersection identification.

4.3. Road network structure

Road network structure, the basis of road network search and intersection identification, is to be briefly described in this section. Road network is a graph structure composed of nodes and road segments, as shown in Fig. 1. Nodes are divided into shape nodes (such as the solid nodes n_1 and n_2 to connect not more than two road segments) and intersection nodes (such as the hollow nodes n_3 and n_4 to connect more than two road segments). A road segment often has a direction, which decides the traveling direction of a vehicle. In addition, the concept of road link has been introduced. The road link L is a road segment sequence $<s_i, s_j, \cdots, s_k>$, where the terminal node of one road segment is the initial node of the next segment. Except for s_i (initial node) and s_k (terminal node), all the nodes are shape nodes. In a road network model, a wider road is often expressed by two or more road links, such as $<s_3, s_1>$or $<s_2, s_4>$. But a narrower road is expressed by only one directionless road link, such as $<s_7>$, $<s_8>$ or $<s_9>$, where a vehicle can run in two ways.

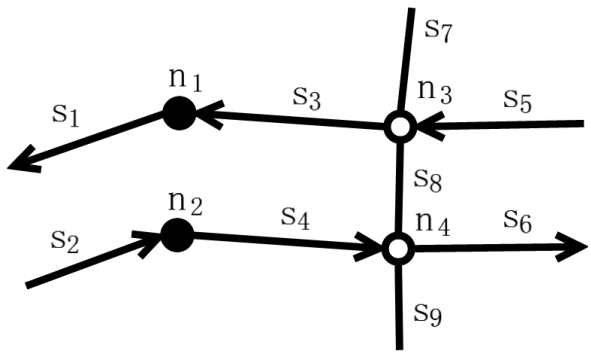

Fig. 1. Road network structure

4.4. Intersection identification

According to the definition of road network structure, every intersection node is connected to more than 2 road segments. But in practice, a node connected to more than 2 road segments may not be an intersection node. Suppose that in the Fig. 1, the vehicle is running on the road segment s_2 in the given direction at the time t. Then it will be very easy to identify the front intersection as n_4 according to the road network structure. If n_4 is beyond the current tile, the vehicle will move eastward along the road link $<s_2, s_4>$ as per the prefetch algorithm in this paper. The central tile at $t^{'}$ can also be determined to finally figure out the tile to be prefetched at t.

The road network structure in Fig. 2 is more complex. Suppose the vehicle is running on the road segment s_2 in the given direction. The front node n_2 is shown as an intersection, but in fact, it is not. That is to say, as long as the vehicle doesnot make a big steering maneuver (for example, from s_2 via n_2 and $n_1$to s_1), it will still stick to the road link $<s_2, s_4>$. In this case, the tile to be prefetched is the same as in the above case. Thus, the decisive intersection is still n_4.

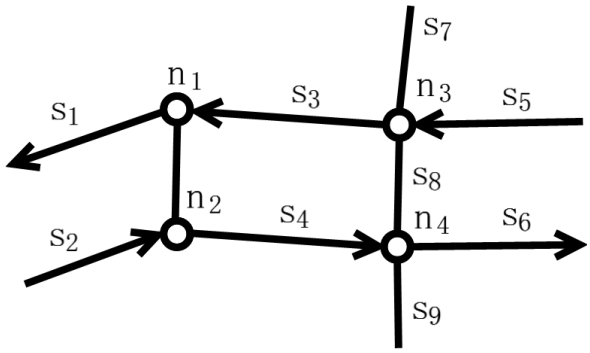

Fig. 2. Road network structure with true and false intersections

As shown in Fig. 2, the most important thing in the road network search is to look for true intersection nodes such as $n_3$and n_4, or to eliminate false nodes such as n_1 and n_2. According to the study, the difference between these two types of intersection nodes is that n_3 and n_4 can force the vehicle into a new direction —a far cry from the original — for a distance that is normally longer than that between n_3 and n_4 or between n_1 and n_2.

In this paper, an algorithm has been designed to tell true intersection nodes from false ones. The goal of this algorithm is not to mistake a true intersection node for a false one given the complexity of road network structure. To better describe this algorithm, several variables shall be included:

n:the intersection node to be identified;

h_n: the direction of the n-linked road segment, where n is the road segment end;

h_{ni}: the direction of the n-linked road segment, where n is the road segment origin and i is the road segment number;

s_{ni}: the road segment starting from n, where i is the road segment number.

The steps to implement the algorithm are as follows:

S1:To determine the direction of search according to the vehicle direction and the road segment direction.

S2:To search the road network until the intersection node n is found.

S3:To expand the node n. Expand all the road segments starting from n, calculate the angles between all the road segment directions and h_n, and record h_{nx}, the direction of the road segment with a smallest angle.

To test the algorithm, a field test has been done. In this paper, 135 intersections were tested, including 97 true intersections. The test results show that this algorithm has correctly identified all the true intersections. In this test, 3 false intersections are identified as true intersections, but this does not affect the subsequent work of the cache system, because the goal of this algorithm is not to identify the true intersection as false intersection.

5. Analysis of test results

In this section, the prefetch strategy proposed by this paper will be compared with that proposed in [12] through a field test, mainly in terms of two performance indexes, namely cache hit ratio and the number of the requested map tiles. The former index reflects the response speed of navigation system, while the latter index reveals the data resource the system needs. To speed up the system response, data can be increased, like in the "full prefetch" strategy. This, however, is far from a good solution, as the growth in data will result in the increase in network bandwidth resource, storage resource and computing resource, which, in turn, may slow down the system response easily. Therefore, a good prefetch strategy shall achieve a higher cache hit ratio with fewer data.

The prefetch algorithm proposed by this paper (hereinafter referred to as "algorithm 1") and that proposed in [12] (hereinafter referred to as "algorithm 2") are used to test the data sets and calculate the cache hit ratio and the number of the requested tiles. As a result, 4 groups of test results can be yielded, including the cache hit ratio H_1 of algorithm 1, the cache hit ratio H_2 of algorithm 2, the number of the requested tiles M_1 of algorithm 1, and the number of the requested tiles M_2 of algorithm 2. Through the analysis of $H_1 - H_2$and $M_1 - M_2$, the performance of the two algorithms can be compared.

Test the data set 1 with the two algorithms. Map the value of every component in $\triangle H$ ($\triangle H = H_1 - H_2$) to the two-dimensional coordinate system, as shown in Fig. 3(a). It is not hard to see that $\triangle H > 0$ (because two of its components are > 0 and the others are $= 0$). As a matter of fact, the average of H_1 components is 0.957, the average of H_2 components is 0.950, and therefore the average of $\triangle H$ components is 0.007. The cache hit ratio of algorithm 1 is 0.007 (or 0.73%) higher than that of algorithm 2 on average. Then map the value of every component in $\triangle M$ ($\triangle M = M_1 - M_2$) to the two-dimensional coordinate system, as shown in Fig. 3(b). The vertical coordinates of the nodes are positive or negative, but the negatives are more and bigger (in terms of absolute value) than positive ones. In fact, the sum of M_1 components is 193, the sum of M_2 components is 201, and therefore the sum of $\triangle M$ components is -8. The requested tiles in algorithm 1 are 8 tiles (or 3.98%)

fewer than in algorithm 2.

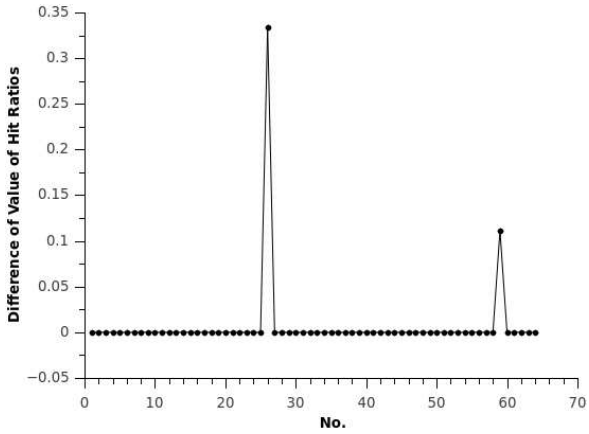

Fig. 3. Difference between algorithms 1 and 2 in testing the data set 1.

Test the data set 2 in the same way. Map the value of every component in $\triangle$H to the two-dimensional coordinate system, as shown in Fig. 4(a), where one component is > 0 and the others are $= 0$. As a matter of fact, the average of H_1 components is 0.947, the average of H_2components is 0.940, and therefore the average of $\triangle$Hcomponents is 0.007. The cache hit ratio of algorithm 1 is 0.007 (or 0.74%) higher than that of algorithm 2 on average. Then map the value of every component in $\triangle$M to the two-dimensional coordinate system, as shown in Fig. 4(b). It is clear that the negative components in Fig. 4(b) are more and bigger (in terms of absolute value) than the positive ones. In fact, the sum of M_1 components is 123, the sum of M_2 components is 147, and therefore the sum of $\triangle$M components is -24. The requested tiles in algorithm 1 are 24 tiles (or 16.33%) fewer than those in algorithm 2.

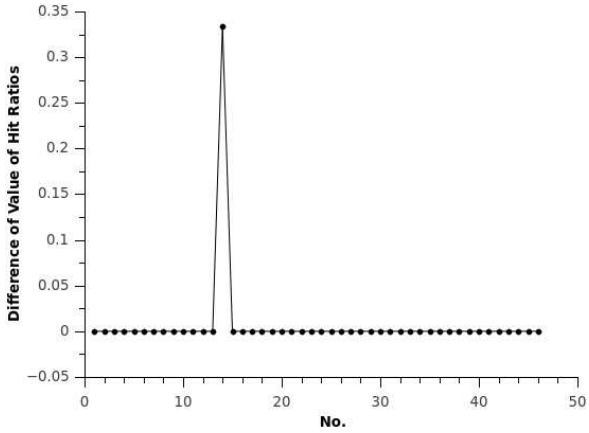

Fig. 4. Difference between algorithms 1 and 2 in testing the data set 2.

6. Conclusion

This paper studies the cache prefetch strategies, and proposes the heuristic prefetch strategy based on road network analysis — a strategy that uses the results of road network analysis to correct the forecast results of heuristic prefetch strategy. According to the test results, the prefetch strategy proposed by this paper is superior to a pure heuristic prefetch strategy, effectively reducing the data volume needed by cache system without sacrificing the cache hit ratio. Meanwhile, since road network analysis is usually realized through a navigation module, this prefetch strategy can make use of the calculated results of the navigation module. Therefore, as far as the whole navigation system is concerned, the application of this strategy won't introduce additional computation overhead.

References

[1] J. Wang, Y. Wang, M. Yun, X. Yang : *Development of Urban Road Network Traffic State Dynamic Estimation Method.* Mathematical Problems in Engineering (2015),1–10.

[2] X. Wang, L. Peng, T. Chi, M. Li, X. Yao, J. Shao: *A Hidden Markov Model for Urban-Scale Traffic Estimation Using Floating Car Data.*PLOS ONE *10* (2016), 1–10.

[3] J. Huang: *An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance.* in Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, New York, NY, USA, (2013),363–374.

[4] G. Wu, P. Huang, X. He: *Reducing SSD access latency via NAND flash program and erase suspension.*Journal of Systems Architecture*60* (2014),345–356.

[5] L. Xiong, Z. Xu, H. Wang, S. Jia, L. Zhu : *Prefetching Scheme for Massive Spatiotemporal Data in a Smart City.*INTERNATIONAL JOURNAL OF DISTRIBUTED SENSOR NETWORKS (2016),11–11.

[6] H. Zhao,B. Shneiderman: *Colour—coded pixel—based highly interactive Web mapping for georeferenced data exploration.*International Journal of Geographical Information Science *19* (2005) 413–428.

[7] H. Kirchner, R. Krummenacher, D. Edwards. may, T. Risse: *A Location-aware Prefetching Mechanism, in 4th International Network Conference (INC 2004), University of Plymouth, Plymouth.* Sadhana (2004), 453–460.

[8] D. -J. Park, H. -J. Kim: *Prefetch Policies for Large Objects in a Web-enabled GIS Application.*Data Knowl.Eng.*37* (2001),65–84.

[9] D. Lee, J. Kim, S. Kim, K. Kim, K. Yoo-Sung, J. Park: *Adaptation of a Neighbor Selection Markov Chain for Prefetching Tiled Web GIS Data.* in Advances in Information Systems *2457* (2002),231–222.

[10] R. Li, R. Guo, Z. Xu, W. Feng: *A prefetching model based on access popularity for geospatial data in a cluster-based caching system.* International Journal of Geographical Information Science *26*, (2012),1831–1844.

Received November 16, 2016