Replacement Method based on Access Spatiotemporal Locality in a Heterogeneous Distributed Cluster-based Caching System for WebGIS

Rui Li*, Xinxing Wang and Yanping Lin

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, Hubei, 430079, PR China

Abstract: Community user access of a WebGIS is characterized by intensity and popularity. The requested geospatial data have the characteristics of spatial and temporal locality. This paper proposed an expression for the replacement feature by balancing spatial and temporal locality as well as long-term and short-term popularity in tile access to ensure that the replacement process can not only optimize global access but also adapt to the access pattern changes. Then, using the Hash function and linear linked chains to provide cooperative cache management in a heterogeneous cluster-based caching system speeds up the query and replacement process of tiles and improves the performance of the cluster-based cache service. Experimental results revealed that the proposed method obtained a higher cache hit rate and a good average response performance for a heterogeneous distributed cluster-based cache system, providing service to more users and thus increased its service capacity.

Keywords: Access pattern, Cache, Hash, Linear liked chain, Replacement, Spatiotemporal.

1. INTRODUCTION

As the development of Web Geographic Information Systems (WebGIS) has progressed, user activity on such systems has increased [1]. High levels of user access to WebGIS entail some social community law and access repeatability and the accessed hotspot geospatial data exhibits spatiotemporal locality [2-5]. A distributed cluster-based caching system (DCCS) can cache accessed hotspots in the cluster-based cache servers, reducing the database I/O bandwidth and the response time for large-scale user access, thereby providing a scalable WebGIS service [6]. DCCS is one of the most effective service-accelerating methods. However, the cache capability in DCCS is limited. When the cache is filled by outdated hotspot data, the new popular hotspot data cannot be cached. Thus, low storing-value data in the cluster-based cache system must be deleted for free storage space for new hotspot caching. This method is called cache replacement, and it directly impacts DCCS's performance in terms of cache utilization, cache hit ratio, response delay, and so on. Thus, cache replacement is the key method to improve the performance of a cluster-based WebGIS service.

Some relevant studies have been conducted on cache replacement for Web pages, which can be divided into three types: 1) methods based on the locality principle, such as; Least Recently Used (LRU) [7], Least Frequently Used (LFU) [8], First In, First Out (FIFO) [7], and their variants; 2) methods based on the size of cached data, such as; Size-based Replacement [9] and its varieties, Greedy dual-size [10], and LRU-MIN [11]; and 3) methods based on specific accessed content, such as the Weight method based on translating time cost, data size, and the latest access time [12], Hybrid-G [13], Lowest Relative Value (LEV) [14] and Size-Adjust LRU [15]. Many existing applications still use LRU as their replacement strategy, such as Google [16] and NASA [17]. However, geospatial data in WebGIS have specific spatial and temporal features in access patterns, which differ from Web pages, and are stored primarily in tiles based on a pyramid model. The tiles in each layer have the same size since there are multiple tiles in a browsing window while a user roams in WebGIS. Thus, the methods mentioned above cannot directly be used in cache replacement for geospatial data.

In the WebGIS research domain, some methods of replacement have been proposed, which can be classified into two types. One type involves replacing tiles with the lowest access probabilities, which are computed through system analysis or training [18, 19]. This requires large volumes of statistics and probability computations because there are large numbers of tiles in the WebGIS. These methods cannot adapt quickly to changes in the access patterns. Thus, such methods cannot be used efficiently in the WebGIS. The other type of method uses statistics of the interval access time for tiles for a single client and replaces tiles with higher interval values in the client cache [20]. Such methods cannot be used to achieve collaboration among heterogeneous cluster-based multi-cache servers.

Some studies have shown that community user access to geospatial data has spatial and temporal locality [2-5, 21]. Temporal locality of access to tile means that the latest accessed tile has a higher probability that it will be accessed again. The temporal locality is embodied in the access time interval or access frequency. Spatial locality of access to tile means that the tiles that are spatial neighbors have adjacent access time, that is, when a tile is accessed, both the tile and its neighboring tiles, which are in the same local area, have a

*Address correspondence to this author at the LIESMARS, Wuhan University, Wuhan, Hubei, 430079, PR China; Tel: +86 27 68778247; Email: Ruili@whu.edu.cn

higher probability of being accessed again in the next moment. The spatial locality of the accessed tile is embodied in the adjacency between the accessed tiles. However, the relationship between spatial locality and temporal locality of tile access is associated. Access to tiles also has the characteristic of long-term and short-term popularity. Thus, this paper analyzed and considered the spatial locality and temporal locality of tile access and proposed a way to express the accessed hotspot popularity and its features of spatial-temporal locality and access stability by balancing the long-term and short-term features, not only to keep the cached objects relatively stable but also to adapt to hotspot changes and to reduce the frequency of replacement operations. The paper then proposed a cluster-based cache replacement method with a collaboration style for heterogeneous DCCS to improve cache hit rate and cluster-based service efficiency.

2. EXPRESSION OF ACCESS SPATIAL-TEMPORAL LOCALITY FOR GEOSPATIAL DATA

Geospatial data are generally stored as tiles and thus this paper used a tile as a cache unit. Zipf's law of tile access states that access to a tile is uneven when users carry out roaming in the WebGIS. The access probability of a tile and its access rank follows a power-law distribution [2-5]. The law further indicates that a tile that has frequently been accessed in the past has a high probability of being requested again in the near future [5]. Thus, the probability of a tile being accessed again can be simplified as being in direct proportion to its long-term popularity (the total number of times a tile is accessed) [20]. Further, if a tile has a higher access frequency, its neighboring tiles will likewise have a higher probability of being accessed. Thus, the total number of times a tile is accessed can reflect spatial distribution of the tile's access with geography features, that is, access to spatial locality. Zipf's law reflects the long-term access popularity of a tile, which can be used for an effective cluster-based cache replacement mechanism [22].

LRU reflects the short-term popularity of tile access. It considers that the probability of a tile again being accessed is inversely proportional to the interval between the tile's access time and current time. Thus, the access probability of tiles is ranked according to LRU in the descending order, and the rank is determined by the latest access time of a tile. Tiles accessed more recently are ranked higher and tiles accessed earlier are ranked lower. Since the rank depends on the latest access time, LRU ignores the long-term access of tiles, which could lead to instability in the replacement. As we observed from the access logs in the actual WebGIS, a tile's access interval time is always dynamic. Thus, we used access interval time to reflect the temporal locality and short-term popularity of the tile's access, and calculated the access interval time to reflect the long-term access popularity and spatial locality. Thus, taking into account both spatial and temporal locality, and both long-term and short term popularity, we proposed an algorithm, Sum of Tile Access Times per Interval (Stat), as shown in (1):

$$stat(i)=\begin{cases}\mathrm{tat}\left(\mathrm{i}-1\right)+\mathrm{tat}\left(\mathrm{i}\right)=\sum_{k=2}^{i}\mathrm{tat(k)},\mathrm{i}\geq 2\\ 0,\mathrm{i}=1\end{cases}\tag{1}$$

with

$$tat\left(k\right)=\frac{accessTimes\left(k\right)-1}{accessTime\left(k\right)-accessTime(1)}$$

Equation (1) shows that *tat(k)* is the average access time in a unit time for *k*-th access, that is, the *k*-th access frequency. The value of *tat(k)* is related to the total number of times the tile is accessed and the current access time, and reflects the long-term access characteristic as a Zipf distribution and access spatial distribution. It considers two access spatial factors: the spatial distance between the current accessed tile and the tile in cache, and the difference of spatial distance between the current accessed tile and the tile in cache. *stat(i)* is the accumulated value of access times in a unit time under the *i*-th access time. It reflects the temporal locality and considers two temporal factors: the interval time between current access time and the previous accessed time of a tile, and the difference between the previous intervals.

$$\begin{aligned}&\mathrm{accessTime}\left(\mathrm{k}\right)-\mathrm{accessTime}\left(1\right)\\&=\sum_{j=2}^{k}\left(\mathrm{accessTime}\left(\mathrm{j}\right)-\mathrm{accessTime}\left(\mathrm{j}-1\right)\right)\\&=\sum_{j=2}^{k}\Delta\mathrm{accessTime(j)}\end{aligned}\tag{2}$$

Equation (2) shows that the *i*-1 previous accesses are all involved in the operation for *stat(i)*. Thus, both the total number of times a tile is accessed and each access to a tile are determined based on the value of stat.

To reduce the complexity of the Stat algorithm and eliminate the uneven distribution of locality for spatial access, stat can be shortened to (3) and (4), where *i* is the *i*-th access:

$$\mathrm{stat}\left(\mathrm{i}\right)=\sum_{k=2}^{i}\Delta t^{-1}=stat(i-1)+\Delta t^{-1}\tag{3}$$

with

$$\Delta\mathrm{t} = \mathrm{accessTime(i)} - \mathrm{accessTime(i\text{-}1)}\tag{4}$$

Thus, the Stat algorithm considers that the access probability *p* is in direct proportion to the total number of times a tile is accessed and is inversely proportion to the interval time. As (5)

$$\begin{aligned}&p\sim \mathrm{totalAccessTimes}\\&and\\&p\sim\Delta t^{-1}\end{aligned}\tag{5}$$

Equation (3) accumulates the reciprocal of each Δt value for the value of stat. The interval time between the adjacent access points is used to replace the average value of multi-access frequency. It can reflect the uneven access in an actual WebGIS. The more a tile is accessed, the higher is the stat value of the tile. Moreover, the shorter the interval time between the two adjacent accesses, the higher is the stat value of the tile, as shown in Equation 3. Furthermore, the higher the stat value, the higher is the probability that the tile will be accessed again. In this way, the stat value of a tile which is not accessed for a long time will gradually decrease. Thus, the stat value indicates the cached value of a tile; therefore, a tile with a lower stat value can be replaced. This method helps to quickly identify the tile with the lower cached value and to reduce the replacement frequency.

3. COLLABORATIVE REPLACEMENT METHOD IN A HETEROGENEOUS DCCS

3.1. Cache Index

A pyramid model for tiles is a valid method for storing and managing geospatial data in a multi-resolution hierarchy model. The idea is that by a block-and-layer operation, different resolution layers are generated by resampling from raw data. A layer of data is mapped onto a specified number of pixels in a block to generate a tile matrix. A tile with coordinates (*tx*, *ty*, ℓ)is on the matrix on the ℓ-th layer, in line with *tx* and row *ty*. The client application calculates the coordinates of the center tile of the current browsing view based on its longitude and latitude, and it then requests the tile by providing its coordinates (*tx*, *ty*, ℓ) to the server. The request format is similar to URL=http://WebGIS_ server_address/tile. aspx?L= ℓ&X=tx&Y=ty&.

A high-efficiency cache index should be built for the DCCS in order to carry out operations such as creating query, updating, and deletion for cache management. When the number of cached tile achieves the replacement threshold value, the cache index can help to implement the cache replacement algorithm. By considering a tile as a unit, this paper built an index for caching tiles based on the pyramid model. As shown in Fig. (**1**), the Hash function and linear linked chains were used to build a cache index *CacheIndex*. The triplet coordinates (*tx*, *ty*, ℓ) of tile as the key variable were mapped to a table entry *h* (0 <=*h* <=H) using the Hash function. When mapping conflict was observed, the tiles with the same Hash value were stored in the same linear linked chain. Thus, the index can complete a query operation with Time Complexity *O(1*n)*(where n is the length of the linear linked chain that connects with table entry *h*) and locate the requested tile in the DCCS quickly.

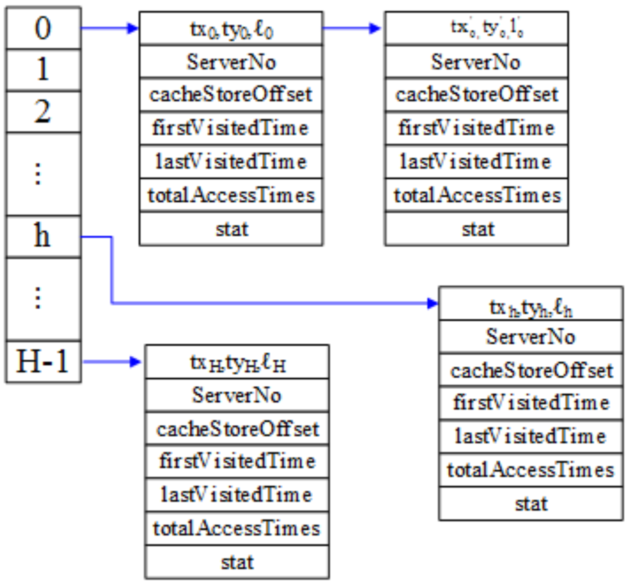

Fig. (1). Cache index.

In the linear linked chain, each node is an array *Tilecached* with size 7. *Tilecached[0]* is the coordinate of tile (*tx*, *ty*, ℓ), *Tilecached[1]*, as the Server No, is the identifier of the cache server in which the tile (*tx*, *ty*, ℓ) is cached, and *Tilecached[2]* is the store offset in the cache. *Tilecached[1]* and *Tilecached[2]*can help to locate the cached tile in the cluster-based cache, in order to quickly obtain the tile data and return the tile to the user, reducing the response delay. *Tilecached[3]*, as the first Visited Time, records the first request time for tile (*tx*, *ty*, ℓ). *Tilecached[4]*, as the last Visited Time, records the latest request time for tile (*tx*, *ty*, ℓ). *Tilecached[5]*, as the total Access Times, records the total access times for tile (*tx*, *ty*, ℓ), and *Tilecached[6]* records the latest stat value for the latest access of tile (*tx*, *ty*, ℓ), which is the replacement attribute value. In each linear linked chain of table entry *h*, each node is sorted in the descending order by stat value. The end node has the lowest stat value. Thus, during cluster-based cache replacement, only the end node of each linear linked chain is compared and the tile with the lowest stat value is replaced. This can reduce the search time in the replacement process.

Because the cluster-based servers are heterogeneous, each server has a different cache capacity(CC) and service processing capacity(SPC, the capacity that the number of requests the sever can process in a unit time), as shown in Fig. (**2**). We should setup another two-dimensional index, *ServerCaching[n][4]*, to record the caching state of each server for cache management and replacement. N is the number of cluster-based cache servers, *ServerCaching[i][0]*, as cacheSize, is the cache capacity of server *Si*. *ServerCaching[i][1]*, as the cached Size, is the used cache size of server *Si*. *ServerCaching[i][2]* is the SPC of server *Si*. *ServerCaching[i][3]*, as currentServiceRequest, is the number of requests that the server currently processes (Current Service Request, CSR).

3.2. Replacement flow and Collaboration in DCCS

For a set of DCCS servers *S*={*Si*,1≤*i*≤N}, each server has a different SPC and CC, as shown in Fig. (**2**). The cluster supervisor manages and harmonizes cluster-based servers to ensure that the DCCS is available and scalable. Based on the simplest management rule and the different capacity of each server, considering both cached tiles and non-cached tiles, the basic idea of DCCS collaboration for the replacement method is that the server with the highest SPC value processes more tile requests and cache more tiles as its cache capability allows in order to achieve load balancing for heterogeneous DCCS and optimal performance of cluster-based service response. Service flow is shown in Fig. (**3**), and is explained below.

Step 1. After receiving the request of tile (*tx*, *ty*, ℓ), the cluster supervisor computes *h*-value for the tile (*tx*, *ty*, ℓ) based on the Hash function. The *h*-th linear linked chain is retrieved connected with the table entry *h* for tile (*tx*, *ty*, ℓ). If the tile (*tx*, *ty*, ℓ) is found, this is known as a cluster cache hit and the Tilecached node of the tile (*tx*, *ty*, ℓ) from *h*-th linear linked chain is returned. According to *Tilecached[1]* (ServerNo.) and *Tilecached[2]* (cacheStoreOf f set), tile (*tx*, *ty*, ℓ) is located in the cluster-based cache and the tile data is returned to the user followed by the modification of *Tilecached[4]* (currentTime) and the stat value in *Tilecached[6]* based on Equ.3. The node of tile (*tx*, *ty*, ℓ) is

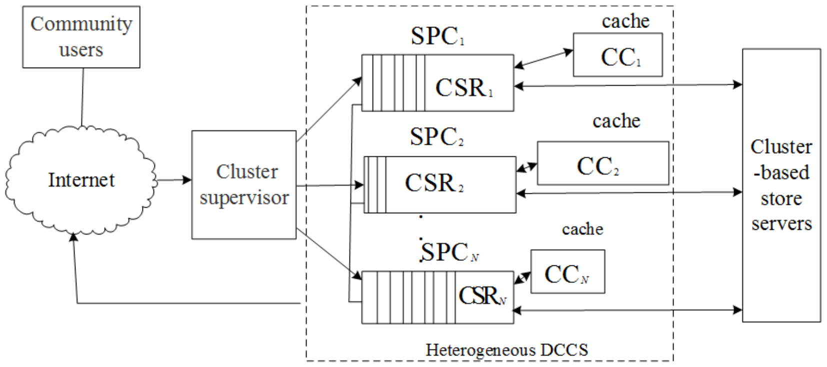

Fig. (2). Heterogeneous DCCS.

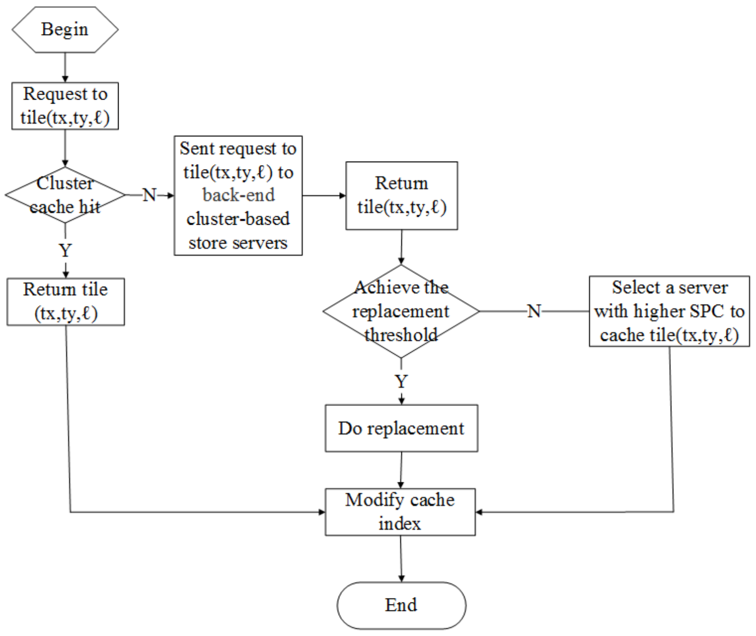
Begin
Request to tile(tx,ty,ℓ)
Cluster cache hit
N
Y
Sent request to tile(tx,ty,ℓ) to back-end cluster-based store servers
Return tile(tx,ty,ℓ)
Return tile (tx,ty,ℓ)
Achieve the replacement threshold
N
Select a server with higher SPC to cache tile(tx,ty,ℓ)
Y
Do replacement
Modify cache index
End

Fig. (3). Replacement flow and collaboration in a Heterogeneous DCCS.

moved to the correct location in the *h*-th linear linked chain. If the retrieval fails, then a "no cluster-cache hit" occurs, after which step 2 is followed.

Step 2. A request is sent for the tile (*tx*, *ty*, ℓ) to the back-end cluster-based store servers to retrieve the tile (*tx*, *ty*, ℓ) and return the data to the cluster supervisor and the user.

Step 3. Supervisor judges are clustered whether the DCCS has reached the replacement threshold. If it has, the stat values of the end node of each linear linked chain are compared in the CacheIndex, to obtain the node with the lowest stat value and to replace the new arriving tile (*tx*, *ty*, ℓ) with the outdated tile (*tx'*, *ty'*, ℓ'). Following this, the *CacheIndex* is maintained by deleting the node of tile (*tx'*, *ty'*, ℓ') and inserting the node for tile (*tx*, *ty*, ℓ) into the correct linear linked chain based on its Hash value.

Step 4. If the DCCS does not reach the replacement threshold, the cache server is selected with the highest value of left SPC (the value is calculated by SPC-CSR) which has space for caching tile (*tx*, *ty*, ℓ). After this, the node of the tile (*tx*, *ty*, ℓ) is inserted into the correct linear linked chain based on its Hash value.

4. SIMULATION AND RESULT ANALYSIS

To simplify the simulation to verify the advantages of cache replacement methods, we used 90-m global Shuttle Radar Topography Mission (SRTM) terrain data, with tiles of size 128×128. In the simulation, 12 distributed cluster-based caching servers were connected using a 1,000-Mbps switch to form a fast Ethernet. A cluster supervisor with sufficient processing power was placed at the entrance of the distributed system to prevent forwarding bottlenecks. The requests to tiles can be expressed as a Poisson distribution [23] in the networked systems. Thus, in this simulation, tile requests were 100,000 following a Poisson distribution. The simulations used the replacement method proposed in this paper, and compared it with the classic methods, such as FIFO [7], LFU [8], LRU [7], and TAIL (Tile Access average Interval time Longest) [20].

The cache size in the DCCS is an important efficiency factor for a distributed cache replacement strategy. The relative size of the cache (RSC) is the ratio of the cache size to the total size of the tiles requested. Therefore, simulations, in which RSC were varied, were carried out to compare the

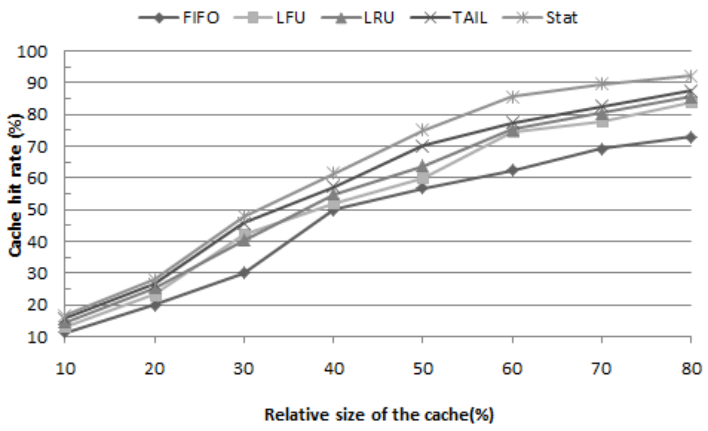

Fig. (4). Comparison of cache hit rates.

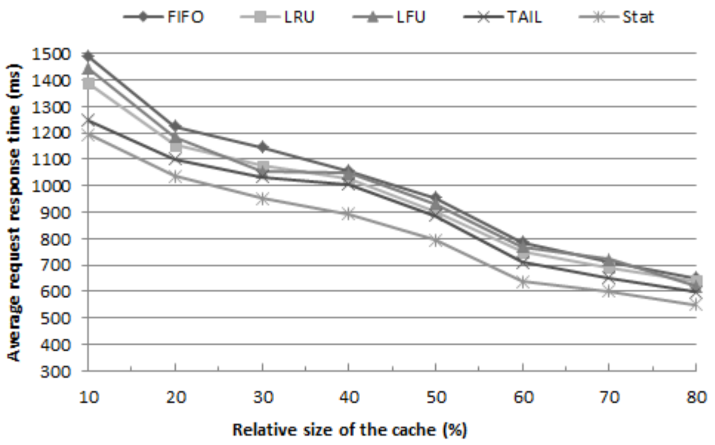

Fig. (5). Comparison of average request response time.

cache replacement performance in terms of the cache hit rate and average request response time.

4.1. Cache Hit Rate (CHR)

CHR is an important indicator to verify the efficiency of a cache replacement method, which reflects the availability of cache replacement. CHR is the ratio of the direct response by a cluster-based cache for the tile requests to the total number of tile requests. Fig. (**4**) shows the CHR of FIFO, LFU, LRU, TAIL and Stat using different RSC. It indicates that cache hit rate increased approximately linearly with the cache size. The CHR of Stat increased rapidly compared to the other methods when the RSC was between 40% and 70%. FIFO and LRU take temporal locality into account while LFU takes spatial locality into account; thus, they both performed more weakly than TAIL and Stat, which consider both temporal locality and spatial locality. When RSC was lower (10%-30%), the CHR of Stat was around 5% higher than TAIL; while when the RSC was between 40% and 70%, the CHR of Stat was around 10% higher than TAIL. This shows that the replacement frequency was higher under lower CHR and Stat and TAIL both reflected the average access frequency in the short term, so they showed little difference in the CHR. When the RSC increases, Stat reflected long-term accumulated access frequency and access stationarity, while TAIL only reflected average access frequency for the short-term. Thus, Stat considered both temporal locality and spatial locality, while balancing the short-term and long-term access popularities.

4.2. Average Response Time (ART)

ART can reflect the advantages of the DCCS, and different cache replacement methods have different influences on the performance of a DCCS. From Fig. (**5**), it can be observed that the ART of the five methods decreased as the cache size increased. Stat's ART was 15% to 19% lower than FIFO, 10% to 15% lower than LRU, 10% to 17% lower than LFU, and 4% to 11% lower than TAIL. This shows that Stat provides more advantageous service performance than the other three methods for the large-scale users. Stat can balance different capacities of heterogeneous servers and use them to the best of their capacities. Furthermore, considering the long-term access characteristics and access spatial and temporal locality, Stat showed lower replacement frequency, reducing operations in the cache. Using the Hash function and linear linked chain to manage the cache, Stat accelerated the retrieval process and reduced the response time, which means that the DCCS can serve more users and increase its service capacity.

CONCLUSION

Access to geospatial data not only has the characteristics of spatial and temporal locality, but also has the features of long-term and short-term popularity in the WebGIS. This paper proposed an expression for replacement feature by balancing the temporal locality and spatial locality of access to tiles, embodying both long-term popularity and short-term popularity of access to tiles. Since cluster-based cache servers in a heterogeneous DCCS have different cache capacities and service processing capacities, this paper proposed a collaboration method for cache replacement in the DCCS, which used Hash function and linear linked chain for cache management and quick replacement. In future work, we will study the access pattern of spatial transfer based on time series during roaming for community users to find a more precise expression for spatial-temporal locality and to improve the performance of the replacement method. However, such an investigation should include large amounts of data from user access logs.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 41371370).

REFERENCES

[1] J.H. Gong, "Man-Earth Relationships Based on Virtual Geographic Environments," In: *6th National Conference Cartography GIS Conference*, Wuhan, Hubei, China, 2006.

[2] D. Fisher, "How We Watch the City: Popularity and Online Maps," *Workshop Imaging City, ACM Computer-Human Interaction*, San Jose, CA, USA, 2007.

[3] Q. Li, Y. Zheng, X. Xie, Y.K. Chen, W.Y. Liu, and W.Y. Ma, "Mining User Similarity Based on Location History," *16th ACM SIGSPATIAL International Conference on Geographic Information Systems*, Irvine, CA, USA, 2008.

[4] N. Talagala, S. Asami, D. Patterson, B. Futernick, and D. Hart, "The art of massive storage: a web imagearchive," *IEEE Computer Society*, vol. 33, no. 11, pp. 22-28, 2000.

[5] R. Li, R. Guo, Z. Q. Xu and W. Feng, "A prefetching model based on access popularity for geospatial data in a cluster-based caching system," *International Journal Geographic. Information Sciences*, vol. 26, no. 10, pp. 1831-1844, 2012.

[6] G. Barish, and K. Obraczke, "Workd Wide Web Caching: Trends and Techniques," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 178-184, 2000.

[7] A. Dan, and T. Don, "An Approximate Analysis of The LRU and FIFO Buffer Replacement Schemes," *Proceedings ACM SIGMETRICS Performance Evaluation Review*, USA, vol. 18, no. 1, pp. 143-152, 1990.

[8] D. Lee, J. Choi, J.H. Kim, S.H. Noh, S.L. Min, Y. Cho, and C.S. Kim, "On The Existence of A Spectrum of Policies That Subsumes The Least Recently Used (LRU) and Least Frequently Used (LFU) Policies," In: *ACM SIGMETRICS Performance Evaluation Review*, 1999, pp. 134-143.

[9] S. Podlipnig, and B. Laszlo, "A survey of Web Cache Replacement Strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374-398, 2003.

[10] P. Cao, and I. Sandy, "Cost-Aware WWW Proxy Caching Algorithms," *USENIX Symposium on Internet Technologies and Systems*, Montery, CA, USA, vol. 12, no. 97, p. 18, 1997.

[11] S. Williams, M. Abrams, C.R. Standridge, G. Abdulla, and E.A, "Removal policies in network caches for world-wide web documents," *Proceedings ACM SIGCOMN96*, vol. 26, no. 4, pp. 293-305, 1996.

[12] J. C. Bolot and P. Hoschka, "Performance Engineering of the WWW: Application to Dimensioningand Cache Design," In: *5th International WWW Conference*, 1996.

[13] K.Y. Wong, "Web cache replacement policies: a pragmatic approach," *IEEE Network,* vol. 20, no. 1, pp. 28-34, 2006.

[14] P. Cao, and I. Sandy, "Cost-aware WWW proxy caching algorithms," In: *Proceedings of the Usenix Symposium on Internet Technologies and Systems*, Monterey, CA, USA, 1997.

[15] K. Cheng, and Y. Kambayashi, "LRU-SP: A Size-Adjusted and Popularity-Aware LRU Replacement Algorithm for Web Cching," *24th Annual International Computer Software and Applications Conference*, 2000, pp. 48-53.

[16] M.N.K. Boulos, "Web GIS in Practice III: Creating A Simple Interactive Map of England's Strategichealth Authorities Using Google Maps API, Google Earth KML, and MSN Virtual Earth Map Control," *International Journal of Health Geographics,* vol. 4, p. 22, 2005.

[17] D.G. Bell, F. Kuehnel, C. Maxwell, R. Kim, K. Kasraie, T. Gaskins, P. Hogan, and J. Coughlan, "NASAWorld Wind: Opensource GIS for Mission Operations," In: *IEEE Aerospace Conference*, 2007, pp. 1-9.

[18] Y.K. Kang, K.C. Kim, and Y.S. Kim, "Probability-based tile prefetching and cache replacement algorithms for web geographical information systems," *Databases and Information Systems*, vol. 2151, pp. 127-140, 2001.

[19] F. Wang, "*Design and Implementation of Web-Based GIS for Forest Fragmentation Analysis,*" Diss. West Virginia University, 2002.

[20] H. Wang, Z.W. Yu, W. Zeng, and S.M. Pan, "The research on the algorithm of spatial data cache in network geographic information service," *Acta Geodaetica et Cartographica Sinica*, vol. 38, no. 4, pp. 348-355, 2009.

[21] D.J. Unwin, "GIS, spatial analysis and spatial statistics," *Human Geography*, vol. 20, no. 4, pp. 540-551, 1996.

[22] L. Shi, Z. Gu, W. Lin, and Y. Shi, "Quantitative analysis of Zipf 's law on web cache," *LNCS*, vol. 3758, pp. 845-852, 2005.

[23] R. Li, Y.F. Zhang, Z.Q. Xu, and H.Y. Wu, "A load-balancing method for network GISs in a heterogeneous cluster-based system using access density," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 528-535, 2013.

Received: April 02, 2015 Revised: May 23, 2015 Accepted: June 06, 2015