Semantic Caching for Web-Based Spatial Applications

Sai Sun and Xiaofang Zhou

School of Information Technology and Electrical Engineering
The University of Queensland, Australia
{sunsai, zxf}@itee.uq.edu.au

Abstract. Client-side caching of spatial data is an important yet very much under investigated issue. Effective caching of vector spatial data has the potential to greatly improve the performance of spatial applications in the Web and wireless environments. In this paper, we study the problem of semantic spatial caching, focusing on effective organization of spatial data and spatial query trimming to take advantage of cached data. Semantic caching for spatial data is a much more complex problem than semantic caching for aspatial data. Several novel ideas are proposed in this paper for spatial applications. A number of typical spatial application scenarios are used to generate spatial query sequences. An extensive experimental performance study is conducted based on these scenarios using real spatial data. We demonstrate a significant performance improvement using our ideas.

1 Introduction

Geographic applications have been used extensively in the Web environment, providing online customized digital maps and supporting map-based queries. However, the overall potential of these applications has yet to be achieved due to the conflict between large size and high complexity of spatial data and relative low transmission speed of the Internet [14]. The conflict is even more serious in the mobile environment which suffers from low bandwidth and low-quality communications (such as frequent network disconnections). Caching pertinent and frequently queried data on the client side is an effective method to improve system performance, since it can reduce network traffic, shorten the response time and lead to better scalability by reusing local resources [4]. Client-side semantic data caching has great potential in spatial database systems. It is enabled by improved processing capacity on the client-side. About ten years ago, clients are still low-end workstations. Caching is only done on the server side aiming to avoid disk traffic in typical commercial relational databases [6]. However, with rapid growth of client-side processing capacity, complex processing required for spatial caching can now be done at an acceptable speed on the client side. Spatial queries are usually related to each other semantically. Statistical analysis shows that spatial queries submitted by a client in a limited time interval have great semantic correlation. Users of 'location-aware' websites usually interact

with a map to zoom in or out and pan on the client side by clicking on spatial objects to request further information. Progressive queries of multiresolution spatial databases, which are used extensively for decision support systems, data mining and interactive systems, usually begin with a general scope of parameters that are iteratively refined in both location and precision until a final satisfactory result is obtained. Semantic caching of highly correlated spatial data on the client side is helpful to improve spatial query response time. Table 1 shows a sample of a typical set of queries submitted by a client. Query 0 begins from a large query scope (query window size 49% = 70%*70%) and low resolution level (Resolution 13). Query 1, 2 and 3 keep zooming in (Refining window as well as increasing resolution level). To Query 3, the query window area is 4% of the whole map and resolution level is 22. Query 4 pans the query window.

Table 1. A typical set of queries in Web-Applicaton

QueryID	Query Window (x, y, width, height)	Query Resolution
0	0.1, 0.1, 0.7, 0.7	13
1	0.36, 0.13, 0.5, 0.5	15
2	0.42, 0.19, 0.4, 0.4	18
3	0.47, 0.28, 0.2, 0.2	22
4	0.53, 0.31, 0.2, 0.2	22

Compared with standard data, spatial data have some notable properties. Spatial data tend to be very large and have complex structures; this makes data organization a significant factor to consider. A good organization method should compress and cluster data reasonably as it can reduce the size of the storage and also the time of accessing database, and increase the utilization of client cache. In this paper, we explore different spatial data organization strategies based on single resolution, multi-representation and multi-resolution ideas. We propose a new data organization method named *Bit_Map* which balances data between the opaque and the fragment ways and allows multi-resolution data access. Our experiments shows that *Bit_Map* is more flexible and needs less data than *SDO_GEOMETRY* — the method used in Oracle DBMS, as well as The multi-representation one. It is especially suitable for semantic caching as it also maximizes data reuse.

In this paper, we investigate the problem of multi-resolution spatial query trimming which is much more complex than aspatial query trimming. We focus on *Window Query* because it is the basic geometric selection and can be served as building blocks for more complex spatial operations. We also discuss about caching coalescence and propose three schemes (reconstruction scheme, fragment scheme1 and fragment scheme2). Our experiments show that the performance of fragment scheme1 is better than or comparable to the other two schemes in all scenarios.

The remainder of this paper is organized as follows. In section 2, we analyze problems combining with related works. Our approach is introduced in section

3. Section 4 describes experiments and provides experiment results. We conclude this paper in section 5.

2 Background

2.1 Semantic Cache

Past research on semantic caching focuses mainly on relational database. It includes: a) query folding [9], [5], which check the satisfiability, equivalence or implication relationship between a query and a given set of data, then further decide whether the query is rewriteable according to given data and how to rewrite. Note that not all queries can be rewritten. In fact, most research in semantic caching is just based on simple query such as 'select object from table where $x > a$ and $x < b$'. b) caching coherency strategy, which is used to ensure that cached data are consistent with those stored in server [1]. c) caching coalescence strategy, which decides how to re-organize cache regions after new data are fetched from database. d) caching replacement policy [4], [2].

Recently, semantic caching in mobile computing environment has received more attention [10], [12], [3], [8], [13]. Most of them focus on location-dependent information services (LDIS) and take the movement of mobile client and the valid region of data into account. Till now, in our best literature review, we have not found any work done to solve the above issues we discussed in Section 1.

2.2 Data Organization

In traditional spatial systems, spatial data are organized based on geometry. Here all information of a spatial object is stored as one record in a database. Consequently geometry elements are opaque to applications and only one resolution level is available to be accessed - the highest level. The advantage of this technique is that the information of the exact geometry can be accessed directly and no extra step of reconstruction is needed. However, to solve the Query 0 in Table 1, 49% data of the whole digital map need to be accessed from database. If local cache is large enough to save the result, no data need to be fetched to answer Query 1 to Query 4. However, usually local cache can not provide such a large space and data have to be abandoned which causes a great waste. Another problem of this model is that the response time of Query 0 is dispensable long. If users finally find the following queries need not be executed, too many unnecessary data had been fetched. One improvement is to use multiple representations of spatial objects. Previous work in this area aims to exploit the benefit of essentially pre-computing of a query result. An object can have up to n representations, denoted as $O_1, O_2, \ldots, O_n$. Each representation contains all vertexes of an equal or lower resolution level. Thus all points in O_i also exist in $O_{(i+1)}$. A representation with a maximum resolution n contains all data of the lower resolutions. The multi-representation technique is, intuitively, superior in terms of data retrieval speed for single queries. However the total replication scheme

of multi-representation defeats the purpose of progressively iterating queries as data in different resolutions are not associated, so duplicated data at a low resolution will be retrieved again when a higher resolution is required. Thus, this model can not well utilize the benefit of cache. Our proposed Bit_Map scheme solves this problem by only storing additional information for every resolution level except the base level. Every representation at higher resolution δ than the base level need to be constructed by combining data of base level and additional information equal to or less the than δ. Therefore, to execute queries in table 1, data of base level and additional information equal to or less than Resolution 13 will be accessed from database to answer Query 0. In the following queries, only additional data in the refined area need to be fetched from database and combined with cached data for construction. The disadvantage of this model is that extra time jis needed for reconstruction. However, noticing that transmission usually cost much more time than CPU computing, the extra time can be ignored.

3 Our Approach

In this section, we introduce our approaches. We focus on three issues, 1) Data Organization, 2) Window Query trimming, 3) Caching Re-organization.

Figure 1 is a three-layer client-server architecture. The client maintains a semantic cache C locally, which begins from empty. When users submit a request, the client formulate a window query Q according to the request, then process query trimming based on Q and C. If C can not fully answer Q, missing data will be fetched from database via web server. Then the client re-organize local cache. Finally, the result of the query is rendered to the user. Additional refinement may be processed in web server and client side.

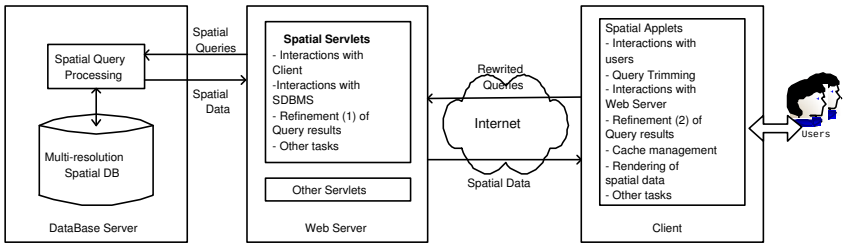

Fig. 1. System Architecture

3.1 Our Database Organization – *Bit_Map*

An object at lower resolution can be considered as an approximation of the object at higher resolution, which still maintains important features of this object but

its data complexity is simpler and it needs less storage. We use z-ordering to fragment spatial objects into series of resolutions. Z-ordering is a method using a space filling curve to transfer two dimensional data into one dimensional data. Starting from the fixed map size, space is iteratively decomposed into four same-size subspaces, named as Peano cells. Each Peano cell is labelled with a unique number that defines its position in the total z-order, which is called z-value of this Peano cell [7], as shown in Figure 2(a). Because the decomposition and encoding is iterative, each child Peano cell's z-value contains its father Peano cell's z-value as a prefix. The longer a z-value is, the smaller the Peano cell is. When a z-value is long enough(usually 22-28 digits), the Peano cell is small enough to represent a point. Thus, we can use Peano cells to represent spatial objects at different resolution [11]. Figure 2(b) shows an approximation at lower resolution. With additional information, we can extend shadowy Peano cells in Figure 2(b) to smaller Peano cells in Figure 2(c) with more details and achieve an approximation at higher resolution.

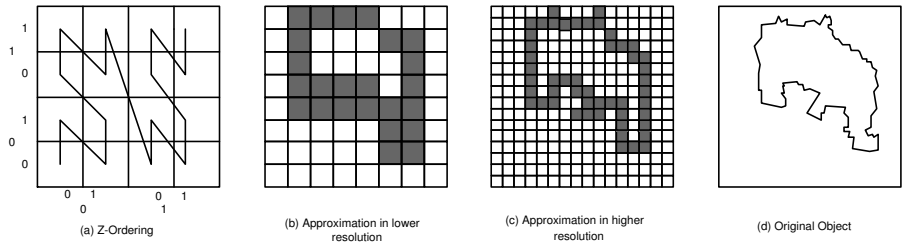

Fig. 2. Using Z-ordering to fragment spatial objects into series of resolutions

Bit_Map scheme chooses a certain resolution level as the base level. Each geometry point at the base level is stored as a variable array. For higher levels, additional information is provided to extend the approximation to the finer polygon with more details. Thus we use relation $R(ObjectID,\ BLData, AData, Delta)$ to describe Bit_Map, where $BLData$ stores the approximations at the base level; $AData$ stores the additional information to reconstruct data at higher resolution, $Delta$ denotes the resolution of data. A spatial object i will have n tuples as $(i, bldate, \emptyset, bl)$, $(i, \emptyset, a_1 data, a_1)$, ..., $(i, \emptyset, a_{(n-1)} data, a_{(n-1)})$ in R. To an arbitrary O_δ, all data with resolution level equal to or less than δ need to be fetched to reconstruct the spatial object O at resolution δ. Figure 3 is an example about the data in Bit_Map. At base level bl, we have a point '12002330' and it expands to two new points in level a_1 ('120023302' and '120023303'). '120023302' further expands to three new points in level a_2. They are '1200233020', '1200233022' and '1200233023'. Note that if the change in the number of points between adjacent resolution levels is small, several continuous levels can be integrated into a single level. Bit_map provides a reasonable, natural method to cluster points to different resolution levels. Because this method uses the common prefix to

construct data iteratively, the reduplication between points is well avoided. In our experiments, data stored with this method occupy 21.501MB, whereas data stored as geometry class of Oracle 9i need 33.438MB. If data is stored by multi-representation technology, same series of resolutions need 152.313MB.

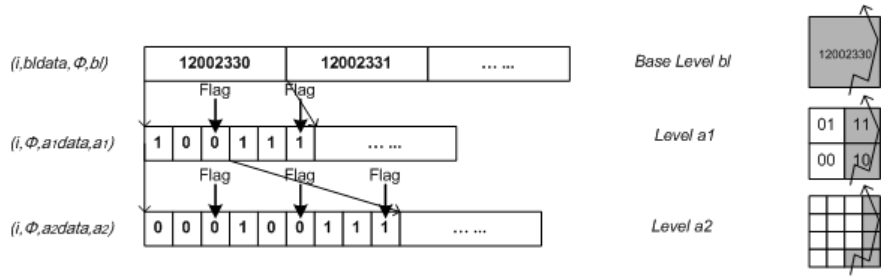

Fig. 3. Bit Map

The algorithm to create $R(ObjectID, BLData, AData, Delta)$ is:

1)Choose a proper z-value length (n) as the highest resolution level, and encode spatial objects at this level as $Z(ObjectID, PointID, z, delta)$, where z is the z-value for the point, *delta* is the resolution value given to the vertex, which is calculated as the number of digits of the longest common prefix between the two end points of the line.

2)Choose resolution bl as base resolution level and constructing $BLData$ at resolution bl. To a spatial object o, using the following SQL to choose points:

$select\ substring(z, 1, bl)$

$from\ Z$

$where\ ObjectID = o\ and\ delta \leq bl$

$order\ by\ PointID$

Thus, the $BLData$ of o is $(o, bldata, \emptyset, bl)$, where $bldata$ is the digital sequence of $substring(z, 1, bl)$.

3)Constructing $AData$ at resolution a_i.

$select\ substring(z, a_i - 1, a_i)$

$from\ Z$

$where\ ObjectID = o\ and\ delta = a_i$

$order\ by\ PointID$

The collection of $substring(z, a_i - 1, a_i)$ is the additional information $a_i data$. Because it is impossible to estimate the exact number of how many Peano cell at a_i will be extended from its father Peano cell at a_{i-1}, we add a binary bit to flag it (0 - the next Peano cell is extended from the same father Peano cell; 1 - the end of extension in the same father Peano cell and the next cell is extended from a different father Peano cell). Thus, the $ADATA$ is $(i, \emptyset, a_1 data, a_1)$, ..., $(i, \emptyset, a_{(n-1)} data, a_{(n-1)})$

3.2 Query Trimming

In this study, we focus on *Window Query*, the most common query in spatial database. Dealing with more complex spatial queries is an important direction of our future research. A window query Q of a multi-resolution spatial relation R, described as $Q\langle W_Q, \delta\rangle$, finds all objects at resolution level δ with at least one point in common with window W_Q. Under the above data organization scheme Bit_Map, we can define the following constraint formula to describe a semantic region or a general query:

Definition 1. *Given a spatial relation R(ObjectID, BLData, AData, Delta), a constraint formula, denoted as* $\langle W, \delta_1, \delta_2\rangle$ *(*$\delta_1 \leq \delta_2$ *and* $\delta_1 \geq bl$*), describes the data (at resolution* δ_1 *to* δ_2*) of spatial objects intersecting window* W *on* R*. When* $\delta_1 = bl$*, it describes the data answering the window query* $\langle W, \delta_2\rangle$*.*

For a general query $Q\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$ and a semantic region $S\langle W_S, \delta_1^S, \delta_2^S\rangle$, Table 2 gives a conclusion of query trimming, where P denotes Probe query and R denotes Remainder Query.

Table 2. Multi-resolution Window Query trimming

	W_S Disjoint W_Q	W_S contains W_Q	W_Q contains W_S	W_S intersects W_Q
$\delta_1^S > \delta_2^Q$ or $\delta_2^S < \delta_1^Q$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$
$\delta_1^S \leq \delta_1^Q$ and $\delta_2^S \geq \delta_2^Q$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$ R: $\emptyset$	P:$\langle W_S, \delta_1^Q, \delta_2^Q\rangle$ R:$\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q \bigcap W_S, \delta_1^Q, \delta_2^Q\rangle$ R:$\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$
$\delta_1^S > \delta_1^Q$ and $\delta_2^S < \delta_2^Q$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q, \delta_1^S, \delta_2^S\rangle$ R:$\langle W_Q, \delta_1^Q, \delta_1^S - 1\rangle$ $+\langle W_Q, \delta_2^S + 1, \delta_2^Q\rangle$	P:$\langle W_S, \delta_1^S, \delta_2^S\rangle$ R:$\langle W_S, \delta_1^Q, \delta_1^S - 1\rangle$ $+\langle W_S, \delta_2^S + 1, \delta_2^Q\rangle$ $+\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q \bigcap W_S, \delta_1^S, \delta_2^S\rangle$ R:$\langle W_Q \bigcap W_S, \delta_1^Q, \delta_1^S - 1\rangle$ $+\langle W_Q \bigcap W_S, \delta_2^S + 1, \delta_2^Q\rangle$ $+\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$
$\delta_1^S \leq \delta_1^Q$ and $\delta_2^S < \delta_2^Q$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q, \delta_1^Q, \delta_2^S\rangle$ R:$\langle W_Q, \delta_2^S + 1, \delta_2^Q\rangle$	P:$\langle W_S, \delta_1^Q, \delta_2^S\rangle$ R:$\langle W_S, \delta_2^S + 1, \delta_2^Q\rangle$ $+\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q \bigcap W_S, \delta_1^Q, \delta_2^S\rangle$ R:$\langle W_Q \bigcap W_S, \delta_2^S + 1, \delta_2^Q\rangle$ $+\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$
$\delta_1^S > \delta_1^Q$ and $\delta_2^S \geq \delta_2^Q$	P:$\emptyset$ R:$\langle W_Q, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q, \delta_1^S, \delta_2^Q\rangle$ R:$\langle W_Q, \delta_1^Q, \delta_1^S - 1\rangle$	P:$\langle W_S, \delta_1^S, \delta_2^Q\rangle$ R:$\langle W_S, \delta_1^Q, \delta_1^S - 1\rangle$ $+\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$	P:$\langle W_Q \bigcap W_S, \delta_1^S, \delta_2^Q\rangle$ R:$\langle W_Q \bigcap W_S, \delta_1^Q, \delta_1^S - 1\rangle$ $+\langle W_Q \bigcap \neg W_S, \delta_1^Q, \delta_2^Q\rangle$

3.3 Caching Re-organization

After missing data is transmitted to client, data will be reconstructed for rendering and cache need to be re-organized. In this section, we propose three re-organizing schemes in client cache.

Reconstruction Scheme As the name suggests, local cache stores reconstructed data. This scheme can avoid repeated reconstruction, but because reconstruction always companies with decompression, this method needs more storage. As shown in Figure 4(a), given three semantic regions S_A, S_B, S_C in cache, δ_1 of S_A, S_B, S_C is 12; δ_2 of S_A, S_B, S_C are 13, 22, 16 respectively; the query is 'to find all polygons intersect window W at resolution level 16'. After executing this query, A and C are decomposed into two parts, *Part 1* is the intersection with Q and *Part2* the difference from Q. Part 1 of A, C, all data of B and ship data are coalesced together as the data of new region. Figure 4(b) shows the regions after Q. Note that δ_2 of S_B is larger than δ_2^Q, data in shadow area is at higher resolution than query needed. After repetitious queries, data in regions may be much more than necessary.

Fragment Scheme 1 In this scheme, data are organized according to Bit Map. The coalescence method is same as reconstruction scheme. The disadvantage of this scheme is that data may be reconstructed duplicately. But it avoids the two problem of reconstruct scheme. Figure 4(c) shows the regions after query. Note region B is divided into two parts in this scheme, data of resolution 12 - 16 and data of resolution 17 - 22. The latter part of data forms Region 6.

Fragment Scheme 2 In this scheme, data are still organized according to Bit Map. But the coalescence method is different. In this scheme, existing regions are not changed. Query window is divided according to difference between existing regions as figure 4(d). This scheme may create numerous too small regions which is easy to cause volatile of accessing database.

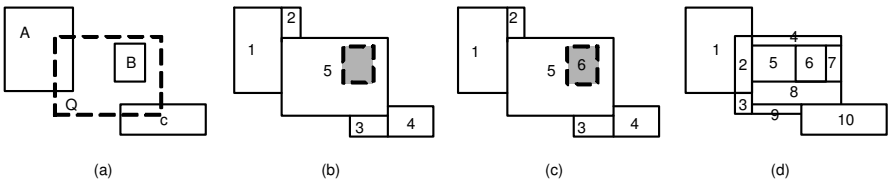

Fig. 4. Three schemes of client cache

4 Experiments and Results

In this section, we investigate the performance of various cache strategies on spatial queries. The data used for experiments are from California SEQUOIA polygon dataset. It contains 20,137 objects which are composed of 2,635,065 points. We have produced tests on various aspects of cache scheme to check the efficiency of semantic caching for web-based spatial applications and to determine

the most efficient semantic caching scheme. These aspects include: 1) size of cache in client side; 2) three different data organizations, SR (single resolution), MP (multirepresentation), BM (our proposed method); 3) For BM, three different re-organizing schemes in client side as discussed in Section 3.3. The primary measurement we use is the amount of transmission as it affects the response time dominantly for spatial query. Other measurements such as time of accessing database, time of data processing are also used. Note that data processing include query trimming, cache region management and data refinement etc. The results of each test are obtained by running 30 groups of queries. Each group begins with an empty cache and performs a window query with the following actions in succession: zoom in, zoom in, zoom in, pan, pan, pan, zoom out, pan, zoom out, pan, zoom in, zoom in, pan, pan, pan. The location of the first window and the extent of zoom in, zoom out, pan is randomly generated. The area of the largest query window is 36% of the whole map, the smallest query window is 0.5%.

4.1 Three Different Data Organizations

We first study the performance of the three data organizations in database: SR (Single Resolution), MP (Multi-representation) and BM (Bit Map). In our experiments, data stored in SR require 33.438MB for storage and 152.313MB in MP, 21.501M in BM. All data in client cache are organized as polygons. The y-axis of figures in Figure 5 (a), (b), (c) and (d) are the time of accessing database, the time of processing data, the amount of transmission and the total response time respectively; the x-axes of the figures are the size of cache (0 means no cache). From figure 5(a) we can see that our proposed data model BM always has the least database accessing time, about 20% lower than that of SR and MP takes less time than SR. However, the difference of data accessing time between three data models is not so significant compared to the amount of data transmitted. Figure 5(b) demonstrates the time cost for data processing in client side. BM spends more time than SR and MP in data processing because it needs extra time for reconstruction. But data processing time has little effect on the whole performance as it is much less than the time cost for accessing database, which is nearly 10 times of the processing time. The most significant difference is the amount of data transmitted. Spatial data is more complex than aspatial data, there needs at least 20 MB data transmission in only 16 queries for SR. As SR always accesses data at the highest resolution, the amount is far more than other models. The amount of data transmitted for BM is the lest because it avoid the repeat farthest. Figure 5(d) shows the total response time with a quite high transmission speed at 512Kbps on the internet. Compared to Figure 5(c), we can see that the trend of three models are very similar in the amount of data transmission and total response time. That is because data transmission is the most time consuming stage for spatial queries which occupies around 95% of the total response time for SR, 80% for MP and 60% for BM respectively even in such a high transmission speed. Moreover, with limited transmission speed, the effect of other factor except the size of data transmission can even

be ignored. Compared to two other kinds of data organization, BM achieves a great performance for spatial web-application.

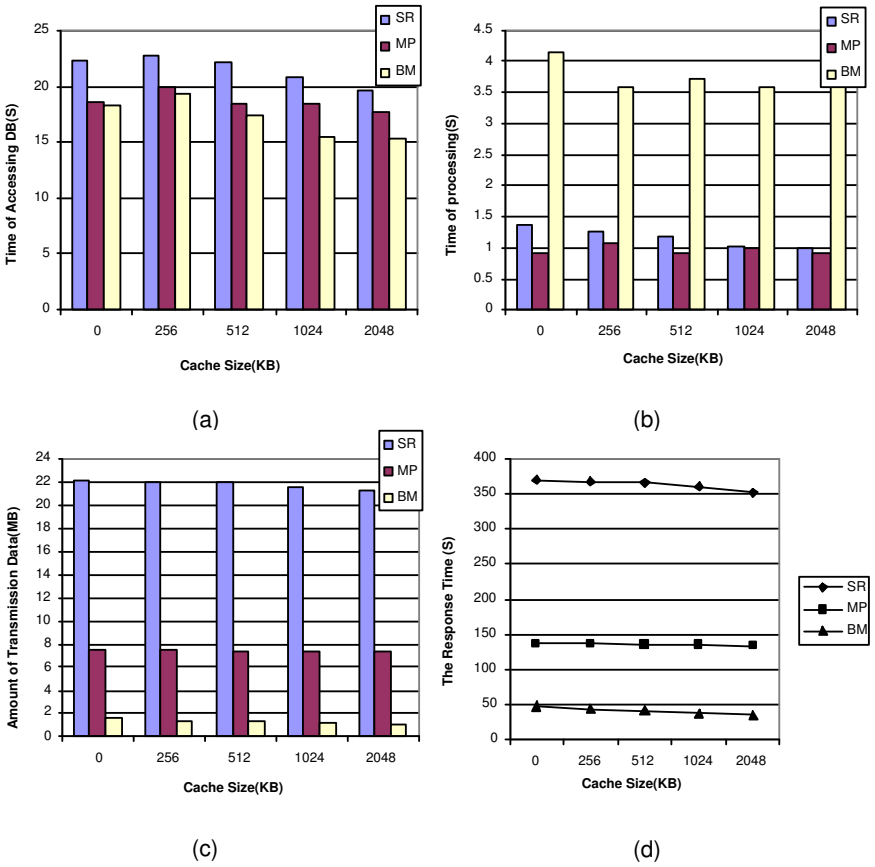

Fig. 5. Comparing SR, MP and BM

4.2 Different caching schemes under BM

We have also studied the performance of different caching schemes under BM which includes RS (reconstruction scheme), $F1$ (Fragment Scheme 1) and $F2$ (Fragment Scheme 2). From figure 6 we can see that $F1$ is the best scheme under various cache size with the least time of database accessing and the least amount of data transmission. Data processing only takes one fourth of the time of database accessing which is similar as in Figure 5. The total response time is mainly affected by the time of data transmission and the time of database access. To sum up, $F1$ is the best scheme for the whole performance with the

least response time while RS is the worst. The response time for all curves tends to decrease with the cache size increasing. Moreover, data accessing becomes more important and has influence on the overall performance of all schemes under BM because of the minimum size of data transmission.

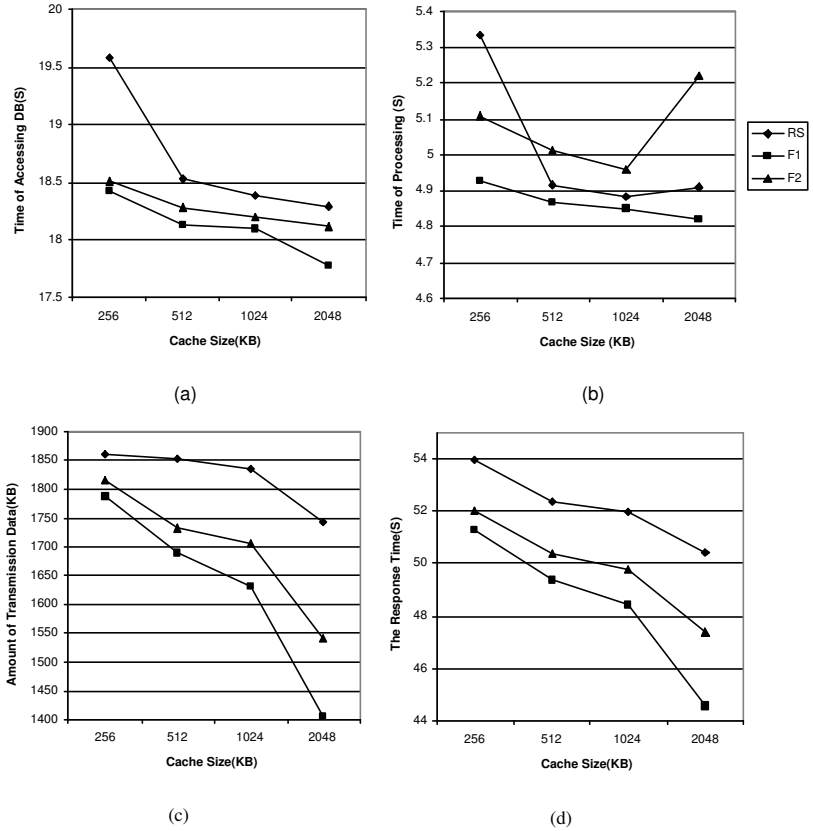

Fig. 6. Different schemes under BM

5 Conclusion

In this paper, we have investigated the effect of semantic caching on the performance of web based spatial applications. We have focused on three problems: data organization, query trimming and caching re-organization. By comparing three major measurements: database accessing time, data processing time and the amount of data transmission for three different data models under various client cache sizes, we found that with limited network transmission speed

(less than 512Kbps), the amount of data transmission is the most dominant factor that affects the response time. Among the three models, BM outperforms the other two by a significant margin, which further improves with the cache size increasing. In order to achieve the optimal caching strategy, we have also constructed three different schemes for BM and tested their performance. Experimental results demonstrated that scheme $F1$ performs better than RS and $F2$ under various cache sizes.

Acknowledgment: The work reported in this paper has been partially supported by grant DP0345710 from the Australian Research Council. We thank Sham Prasher and David Horgan for many helps received from them during this project.

References

1. J. Cai, K.-L. Tan, and B. C. Ooi. On incremental cache coherency schemes in mobile computing environments. In *ICDE*, 1997.
2. B. Y. Chan, A. Si, and H. V. Leong. Cache management for mobile databases: Design and evaluation. In *ICDE*, 1998.
3. X. Chen, Y. Chen, and F. Rao. An efficient spatial publish/subscribe system for intelligent location-based services. In *2nd International Workshop on Distributed Event-Based Systems*, 2003.
4. S. Dar, M. J. Franklin, B. Jonsson, D. Srivastava, and M. Tan. Semantic data caching and replacement. In *VLDB*, 1996.
5. J. Gryz. Query folding with inclusion dependencies. In *ICDE*, 1998.
6. A. M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. *The VLDB Journal*, 5(1):35–47, 1996.
7. J. Orenstein and T.H.Merrett. A class of data structures for associative searching. In *PODS*, pages 181–190, 1984.
8. W.-C. Peng and M.-S. Chen. Mining user moving patterns for personal data allocation in a mobile computing system. In *Proceedings of the 29th International Conference on Parallel Processing*, 2000.
9. X. Qian. Query folding. In *ICDE*, 1996.
10. Q. Ren and M. H. Dunham. Using semantic caching to manage location dependent data in mobile computing. In *Proceedings of the 6th annual International Conference on Mobile Computing and Networking*, pages 210 – 221, 2000.
11. S. Sun, S. Prasher, and X. Zhou. A scaleless data model for direct and progressive spatial query processing. In *The First International Workshop on Conceptual Modeling for GIS*, 2004.
12. J. F. Yao and M. H. Dunham. Caching management of mobile dbms. *Integrated Computer-Aided Engineering*, 8(2):151–169, 2001.
13. B. Zheng, J. Xu, and D. L. Lee. Cache invalidation and replacement strategies for location-dependent data in mobile environment. *IEEE Transactions on Computers*, 51(10):1141–1153, 2002.
14. X. Zhou, S. Prasher, S. Sun, and K. Xu. Multiresolution spatial databases: Making web-based spatial applications faster. In *Proceedings of Asia-Pacific Web Conference 2004*, pages 36–47, 2004.