Enabling P2P Cooperative WMS Proxy Caching and Prefetching in an Educational Environment

Jeffrey A. Bergamini and Dr. Michael Haungs

Abstract. Given the great demand and promise for educational use of GIS, real time access to massive remote geospatial datasets for pedagogical purposes would be immensely useful to educators and students. However, such access has remained elusive. In other work, we have demonstrated that a P2P distributed system of client-side proxies can address the challenges posed by the interactive, multiplicative, and exploratory nature of classroom GIS, and we described this system at a high level. In this paper, we present the details of several novel techniques that enable P2P cooperative caching and prefetching of OGC WMS data in an educational lab environment, via an implicit and flexible pyramid tiling scheme, a query smoothing heuristic, and statistical prediction. The techniques are standards-compliant and client-transparent, and provide dramatic improvement in user response times while reducing impact on remote WMS servers.

Keywords: WMS, P2P, caching, prefetching, tiling, education

1 Introduction

Massive publicly available image-based geospatial datasets for use with GIS are increasingly offered over the Internet. Several high-profile examples include NASA JPL's OnEarth[1], the GLOBE Visualization Project[2], and TerraServer[3] from Microsoft and USGS. The Open Geospatial Consortium (OGC)[4] has standardized re-

[1] onearth.jpl.nasa.gov
[2] www.globe.gov
[3] terraserver-usa.com
[4] www.opengeospatial.org

mote access to these data through web services via the Web Map Service (WMS) standard.

Educators have great interest in using GIS for pedagogical purposes. GIS offers opportunities for enhanced study and research "in any and every discipline" [1]. The various difficulties of local data storage arouse particular enthusiasm for using remote datasets [2, 3]. However, current access methods fail to address the interactive, multiplicative and exploratory nature of classroom GIS. The lab's inherent quantity and concurrency of requests results in the common and problematic occurrence of excessive response times and denial of service [4].

In other work, we have given a high-level overview of *GeoTorrent*, a system we have developed to overcome these serious limitations of traditional, direct access to remote WMS servers in an educational lab setting [4]. GeoTorrent enables practical, responsive use of remote WMS data in this setting (or other similar situations). It provides a distributed network of client-side proxies that cooperatively manage WMS connections in order to optimize user response times. GeoTorrent works alongside and transparently to WMS-compatible GIS client software, assembling a shared distributed cache, intelligently prefetching data, and offering aggregated service information.

This paper presents the details of several novel standards-compliant and client-transparent techniques for peer-to-peer cooperative caching and prefetching of geospatial data:

- *WHoMPyTS*, a conceptual model of WMS data, allows us to define an implicit pyramid tiling scheme, enabling flexible, client-neutral tile-based cooperative caching and prefetching.
- *WMSmooth* is an algorithm for dynamically analyzing and modifying incoming WMS queries, making them suitable for caching and prefetching.
- Two prefetching methods, *New Neighbors* and *Prevailing Wind*, allow for the prediction and prefetching of WMS data based on statistical analysis of incoming requests.

The remainder of the paper is structured as follows: Section 2 explains WHoMPyTS and the need for an implicit tiling system. Section 3 details the WMSmooth algorithm and how it allows WHoMPyTS, and caching and prefetching in general, to function. Section 4 discusses the two prefetching methods. Section 5 evaluates the proposed techniques. Sections 6 and 7 review related and future work. Some conclusions are drawn in Section 8.

2 WHoMPyTS: Implicit WMS Pyramid Tiling

Normative WMS imposes no practical limitations on the image size and/or bounding box of a request[5]. This presents a rather large problem with regard to the cooperative caching and prefetching of WMS data: Peer group clients are likely to generate

[5] The bounding box must lie within the geographic area covered by the requested layer(s).

requests that are similar, but not identical, and are therefore very difficult to cache and prefetch effectively. Request tiling is a common solution in other situations, but WMS does not explicitly support it. However, we have been able to exploit the cooperative proxy model to extend the benefits of tiling to WMS. In doing so, we create a new twist on the classic tiled pyramid storage system, which we have dubbed "WHoMPyTS".

2.1 WMS Tiling: Benefits and Challenges

Tiling, or offering identically-sized data images along regular, predefined boundaries, affords many benefits to both servers and clients. Databases can be optimized to serve tiles quickly by either exploiting their native tiled storage or proactively caching requests [5]. Tiling clients facilitate user "browsing" capabilities to users, since users need not re-request the entire screen image when the view is slightly changed (e.g. while scrolling).

Traditional WMS clients (e.g. ArcGIS[6], uDig[7]) do not support tiling, since WMS has no explicit support for it. Web-based client/server GIS systems (and their underlying databases) that are built with speed and scalability in mind generally employ tiling. Examples include Google Maps[8], TerraServer[9], and worldKit[10]. Some of these can be modified to communicate with WMS servers, but one serious drawback remains: They make multiple simultaneous requests for relatively small "tiles", each of which is processed by the WMS server as a normal, arbitrarily defined request, resulting in poor performance (see 5.3).

However, the unique properties of the cooperative proxy model allow us to create a new, tiling efficient twist on the data storage model, and to apply it to WMS in a standards-compliant manner.

2.2 Unique Properties of Tiled Pyramid Storage in a Cooperative Proxy Cache

The benefits of tiling result from data being stored in a "tiled pyramid" structure, frequently used to store massive geospatial datasets [6]. The tiled pyramid approach involves generating identically-sized tile images at multiple levels of resolution from a large, high-resolution database (see Figure 1). Spatial database management systems (SDBMS) that support WMS servers (e.g. TerraService, OnEarth) generally use some variation of this idea to store their data, and clients make requests for these pre-computed tiles.

A cooperative WMS caching / prefetching system has a goal similar to that of a SDBMS supporting a normal WMS server: fast, efficient and complete service of a very large archive of spatial information. As such, we want to be able to use

[6] esri.com/arcgis
[7] udig.refractions.net
[8] google.com/apis/maps
[9] terraserver-usa.com
[10] worldkit.org

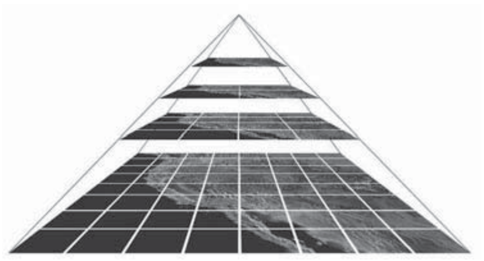

Fig. 1. Classic tiled pyramid (courtesy [7])

tiling. As explained in 2.1, WMS tiling is unsupported in a normal client / server model. However, several fundamental differences exist between a P2P cooperative WMS proxy cache (PCWPC) like GeoTorrent in an educational lab environment and a standard geospatial database:

External Data. A PCWPC's data exists elsewhere, and the pyramid structure can be built on demand as data is requested or prefetched. Consequently, the pyramid benefits from increased flexibility in shape and structure, and can be built to maximize its utility to peers in the PCWPC group.

Client Uniformity. WMS servers can make no assumptions about the clients requesting data or their possible tiling schemes. A PCWPC in the educational environment can assume that multiple instances of a single tiling client are in use and, therefore, tailor its caching and prefetching to that client's tiling scheme.

Request Locality. WMS servers fulfill a general purpose. They can assume nothing about the locality and frequency of the data requested of them, which makes caching difficult. A PCWPC in an educational lab serves a specific purpose, viz. caching and prefetching lab data. Students exhibit similar access patterns and work with a subset of the data offered by any given data source. The likelihood of repetitive and localized requests makes caching and prefetching much more feasible.

The combination of these differences makes it possible to use a distributed pyramid tiling storage system in a PCWPC like GeoTorrent. However, pyramid tiling normally relies on predefined tile boundaries. We may be able to allow clients to define those boundaries, but we need a standard way of analyzing them in order to cache and prefetch tiles. Luckily, we can exploit part of the WMS standard to allow this.

2.3 WGS 84 Standard in WMS

In order to "maximize interoperability among servers", the WMS specification requires that all WMS providers support the WGS 84 coordinate reference system (CRS) [8]. Its use in WMS is simple. For example, if the desired image includes the entire surface of Earth, the bounding box (`bbox`) section of a WMS query using the WGS 84 CRS looks like this:

```
bbox=-180,-90,180,90
```

The presence of a universal CRS means a PCWPC can use it to define and analyze client tile boundaries. We propose the imposition of a simple and flexible rule on clients, that elegantly allows peer client requests to build a tiled pyramid-like cache structure throughout the PCWPC group.

2.4 WGS 84 Hollow Mosaic Pyramid Tiling Scheme (WHoMPyTS)

Implicit WMS tiling and a pyramid-like storage structure can be guaranteed in a P2P cooperative proxy cache through the following simple rule:

WHoMPyTS Rule 1 *For requests to be cached and prefetched, clients must request images of regular (but arbitrary) size on regular (but arbitrary) WGS 84 boundaries.*

This rule is the essence of what we have dubbed the WGS 84 Hollow Mosaic Pyramid Tiling Scheme, or WHoMPyTS. WHoMPyTS has several important characteristics that distinguish it from a traditional pyramid tiling scheme (see Figure 2):

- Unlike the classic tiled pyramid, WHoMPyTS starts out as a loosely defined "hollow" pyramid. The above rule governs the structure of the pyramid. The pyramid data takes form on demand, as client requests are received and predicted data is prefetched.
- Unlike a traditional pyramid tiling scheme, WHoMPyTS does not predefine a PCWPC's own tile boundaries, but instead lets clients choose them (and change them), forming a client-driven "mosaic" structure for the pyramid. Note that this involves no extra or non-OGC standard communication between clients and the PCWPC.
- A WHoMPyTS pyramid consists of only the subset of data that is useful to the group.

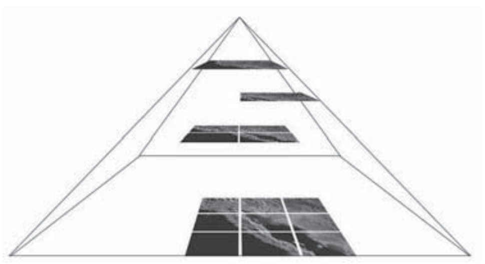

Fig. 2. A WHoMPyTS pyramid

It bears mention that clients need not be aware of WHoMPyTS in order to take advantage of it. As long as a client follows the WHoMPyTS rule, a PCWPC such as GeoTorrent can fully cache and prefetch the requests. Common sense and a preliminary survey of pre-existing tiling-based clients indicate that most clients already follow the WHoMPyTS rule[11]. It is also worth mentioning that non-tiling clients can still request normally through a WHoMPyTS-aware PCWPC, as stipulated by the WMS specification.

Section 4 explains how a PCWPC prefetching system can verify that clients are following the WHoMPyTS rule.

3 WMSmooth

GeoTorrent's cooperative proxy WMS caching and prefetching is triggered by and dependent on receiving WMS `GetMap` requests from clients. Each time a request is received, GeoTorrent checks its local and group caches to see if it can return a cached or prefetched version, etc. The details behind caching are described in [4], and Section 4 discusses our prefetching methods. The main requirement enabling both caching and prefetching in any WMS proxy is the ability to make exact matches between client requests and cached or prefetched responses.

One might assume that if clients follow the WHoMPyTS rule, making exact matches should not be a problem. That is, if a PCWPC node has sensed the square tile and boundary size that a given client is using, it should be able to:

1. Easily match a client request to a cached response
2. Easily add to and subtract from query bounding boxes in order to form prefetch requests

However, in practice this is not necessarily true, primarily due to floating point inaccuracies in calculation of bounding boxes, both on the client and the PCWPC. For example, the Google Maps client has a fairly bad record of generating regular and / or repeatable bounding boxes in its queries. This is due entirely to floating point errors. The client uses a formula to calculate the WGS 84 boundaries of each tile, resulting in bounding box corner values that extend to 7-13 decimal places (e.g. a latitude of `37.85750715625203`). These numbers often have mantissas much longer than necessary for their respective bounding boxes and image sizes. If we use exact matching with these numbers to predict and prefetch tiles and check for cache hits, floating point errors completely prohibit any prefetch cache hits (see Section 5.1) and undermine normal cache hits as well.

Since a PCWPC has no control over the way clients derive their tile bounding boxes, it must have some way of finding a "close enough" match rather than an absolute one, in order to effectively cache and prefetch. We propose a simple but elegant algorithmic solution, which we call "WMSmooth".

The idea behind WMSmooth is that a PCWPC may discard, upon receiving a client request, unnecessary and potentially inaccurate portions of the mantissas of the four bounding box elements, at least for the purpose of cache lookup. In this way, a client request for a tile that is fundamentally identical to one that has been cached (normally or through prefetching) will actually generate a cache hit, even if some of the insignificant digits of its bounding box corners do not exactly match.

[11] We refer to all those mentioned so far, further documented in [4].

3.1 Mantissa-rounding Algorithm

WMSmooth refactors WMS requests using an algorithm based on the geographic extent of a request with respect to its image size (in pixels). The formula determines the number of significant digits in bounding box mantissas. To begin, we calculate the degrees per pixel in both dimensions:

```
xDegreesPerPixel := longitudinalExtent / imageWidth
yDegreesPerPixel := latitudinalExtent / imageHeight
```

Given the measurements of degrees per pixel, for each dimension we can figure out the number of decimal places necessary to maintain (pixel-wise) the same image. We use the following function:

```
function bBoxSigDigits (degreesPerPixel):
   sigDigits := 0
   testPrecision := 1
   while (degreesPerPixel > testPrecision):
         testPrecision *:= .1
         sigDigits++
   return sigDigits
```

When this formula is applied to both of the degrees-per-pixel measurements, we get the longitudinal (‘x’) and latitudinal (‘y’) number of significant mantissa digits.

3.2 Query Modification

In GeoTorrent, each time a node receives a client request that uses WGS 84, WMSmooth modifies the query string “on the fly” to reflect only the number of significant digits in the corners of the bounding box. For example, the original `bbox` element of a query may be:

```
-121.640625,37.3002752813443,-121.46484375,37.43997405227057
```

Given a request for this geographical extent in a 256x256 pixel image, WMSmooth alters `bbox` to be:

```
-121.641,37.300,-121.465,37.440
```

A CPWPC can use the original `bbox` to request the tile from the remote server but perform cache and prefetch lookups based on the WMSmooth `bbox`. This avoids any inadvertent (though unlikely) changes in the resulting image generated by the server.

3.3 Complexity and Performance Penalty

Note that the result of the WMSmooth mantissa-rounding formula is equivalent to the mathematical expression $\lfloor |\log_{10}[degreesPerPixel]| \rfloor$. We minimize the performance penalty of calculating this upon receiving each WMS query by using our own "shortcut" version, viz. the WMSmooth rounding algorithm. Since we only care about the integer portion of the logarithm, and can disregard the sign, the entire algorithm's complexity can be reduced to $o(\log_{10}[n])$. This is a definite gain over general-purpose algorithms for computing the logarithm of a real number [9]. We demonstrate the minimal experimental effect on performance in Section 5.

4 Prefetching Methods

A PCWPC's ability to decrease user response time correlates to its ability to predict and prefetch data. Peer nodes can use the WHoMPyTS and WMSmooth guidelines detailed in Sections 2 and 3, combined with statistics collected on incoming client requests, to predict and prefetch likely future client requests. Each incoming WMS request can be analyzed for prefetching, and if the analysis results in any predicted requests, they can (for example) be added to a prefetch queue.

4.1 Sensing WHoMPyTS

For a PCWPC to perform prefetching, it must first verify that clients are following the WHoMPyTS rule. Since the rule is quite simple, so is the verification.

Peer nodes can maintain a record of the number of incoming client requests for a given combination of bounding box extent and image size. Once a PCWPC becomes aware of multiple queries with the same combination, prefetching can be enabled on that level of the "hollow mosaic pyramid".

4.2 New Neighbors Prefetching

In the GeoTorrent PCWPC, we offer two prefetching methods. The first and simplest is called "New Neighbors" prefetching (NNP). NNP involves exactly what its name implies: As each request is received and passes the WHoMPyTS test, GeoTorrent analyzes the request and calculates each of the eight surrounding tiles. Those not already cached by the node / group or scheduled for prefetching are added to the front of the prefetch queue.

4.3 Prevailing Wind Prefetching

The other, slightly more complicated method of prefetching is called "Prevailing Wind" prefetching (PWP). PWP models the idea of a prevailing wind measurement. A PCWPC can keep statistics (a "wind vane") on the overall direction of movement in the last N requests (GeoTorrent uses 20). PWP prioritizes the eight surrounding tiles of each request so that the tiles in the prevailing direction are processed first.

5 Evaluation

Preliminary experiments show promising results for our techniques. We review some of them.

5.1 Efficacy of WMSmooth

Table 1 describes a quick study of WMSmooth's functionality. We performed a simple scripted experiment using the Google Maps client to view and browse an area of TerraServer's UrbanArea layer involving roughly 100 256x256 pixel tiles (recorded from a student's browsing session). We ran the experiment in two different conditions, averaging the results over ten runs for each condition. The first condition used GeoTorrent without WMSmooth (unmodified requests), and the other included WMSmooth; both used New Neighbors prefetching. We observed several important results:

- Without WMSmooth (and, by extension, WHoMPyTS), prefetching did not function **at all**. Analysis of the individual queries showed that this was due to slight floating point differences in the queries' bounding boxes.
- WMSmooth also improved the normal cache hit ratio for the same reason. This implies that the client's formula for generating tile bounding boxes was not entirely deterministic, or at least not absolutely repeatable.

Table 1. WMSmooth increases cache and prefetch rates.

	Request Analysis	
Response Type	**Unmodified**	**WMSmooth**
Uncached	67%	51%
Cached	33%	38%
Prefetched	0	11%

WMSmooth query modification adds an element of overhead to GeoTorrent's query processing, but the delay is insignificant. Informal measurements on student workstations show that the WMSmooth calculations add roughly only 10 milliseconds of processing time per request, on average. The benefits afforded outweigh this inconsequential amount of overhead.

5.2 Prefetching Mode Evaluation

Figure 3 describes the results of several experiments we performed to compare and evaluate the performance of the two prefetching algorithms. We measured the change in GeoTorrent's overall response time to clients vis-à-vis the prefetching mode. We ran the Google Maps client along two different scripted paths, each starting at the same point and involving roughly 400 requests, to generate requests for TerraServer's

DOQ layer. One path followed a straight line to the north, and the other explored in a zigzag pattern. No previous cache existed. Each experiment was repeated 10 times, and the results were averaged.

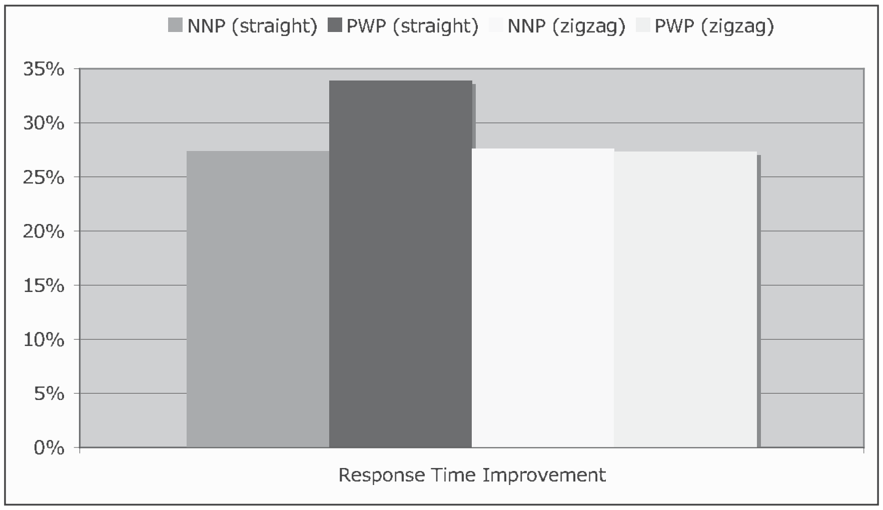

Fig. 3. Comparison of prefetching modes

Prefetching in general improved overall response times by roughly 25% to 35%. The performance of New Neighbors mode (NNP) was stable between the two paths as expected, since its behavior is independent of incoming requests. Prevailing Wind (PWP) did show improvement in the straight line path, also as expected, since its "wind vane" allowed GeoTorrent to queue the northern neighbors for immediate prefetching.

5.3 Scalability and Overall Performance

To get an idea of the collective performance of our techniques in their intended cooperative proxy environment, we ran several scalability and load test experiments with GeoTorrent. Figure 4 illustrates the measured scalability and overall performance of GeoTorrent's combination of the WHoMPyTS rule, WMSmooth, and New Neighbors prefetching in a simulated educational environment. We ran GeoTorrent on twenty student workstations with unique Internet IP addresses and had each workstation make requests chosen randomly from a list of WMS requests. We simulated different loads by varying the concurrency of requests across a range from 1 to 20.

Our results show that when WMSmooth is applied to requests made in accordance with the WHoMPyTS rule, a PCWPC can indeed provide dramatic reductions in response time, and that GeoTorrent's performance is scalable (up to at least 400

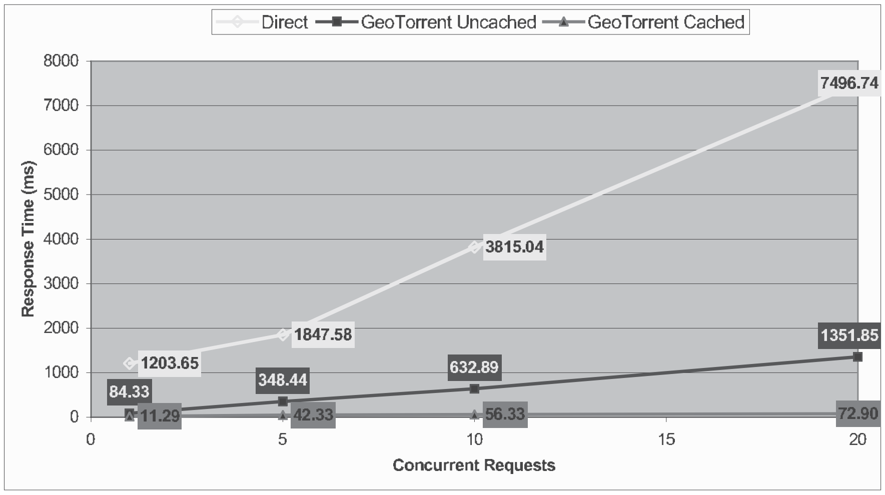

Fig. 4. The combination of WHoMPyTS, WMSmooth, and prefetching provide dramatic, scalable improvements in response time.

concurrent, distributed group requests in this example). Note also that the caching enabled by our techniques ensures that no request made within the cooperative group is ever made to an external server more than once, drastically reducing the impact of the group on the server. Similar experiments in [4] review these results in further detail.

6 Related Work

We review some of the more pertinent literature on prefetching / path prediction and cooperative proxy caching.

6.1 Prefetching (Path Prediction)

The rise of mobile cell phones and hand-held computers in recent years has spurred interest in practical path prediction algorithms. However, they tend to focus more on regularity of movement based on daily or weekly patterns [10]. They also must take into account some things that don't necessarily apply to our situation, such as velocity of travel or road paths [11]. Others take into account other aspects of actual first-person human movement (e.g. frequented locations) [12]. We needed a more generalized approach based solely on trends in recent access, perhaps better met by our prefetching models.

6.2 Cooperative [Proxy] Caching

A fair amount of research exists on cooperative proxy caching, including applications, limitations, and suggestions. Most work on cooperative caching focuses on web proxies [13, 14, 15, 16, 17]. We focus on several of the most applicable and interesting.

The Squirrel peer-to-peer web cache [13] caches web pages among a group of machines by building a "scalable distributed hash-table" among the group. Squirrel associates web pages with a given hash ID and uniformly distributes hash IDs among group members, for fault tolerance purposes. As a result, each message passes through an average of three nodes. The goal of interactivity in the educational environment precludes routinely routing messages among multiple peers, and node failures are unlikely within the span of a lab period.

Other work attempts to avoid Squirrel's hash key assignment model. One such approach is modeled, oddly enough, after the hoarding mechanisms of squirrels (the mammals) [15]. This work proposes a multi-agent system where peers work independently, but according to heuristics that result in a balanced resource allocation. They conclude that a "sniffing and burying" algorithm, where a peer "sniffs" several random possible locations before deciding to place a resource, results in a well-balanced allocation. We chose a reactive model instead, prioritizing immediate response over long-term balancing. Some work suggests that cooperative proxy caching offers only negligible benefit for user response time, at least in web caches, due to the number of cache misses [16]. However, our situation differs from standard web caching in that the peer group should repeatedly and reliably access the same or similar data.

General purpose web proxy caches (e.g. Squid [14]) serve a similar purpose. However, due to the application-specific challenges we address with WHoMPyTS, WMSmooth, prefetching, etc. (as well as the educational environment's infrastructural limitations), normal proxies are not suitable.

7 Future Work

Our results are encouraging, and merit further exploration. We propose several avenues.

7.1 Tiling Hints

Databases supporting WMS servers often already store data in a tiled fashion [4], so it would be nice if clients and/or a CPWPC could take advantage of those preexisting tile boundaries, making requests along them and thereby increasing performance. Optional tile boundaries could be advertised in a WMS Capabilities document, given a standard notation.

7.2 Enhanced Prefetching

Many GIS clients incorporate the idea of “zoom levels”, i.e. a user zooms in and out to predefined view levels. If a prefetching algorithm could sense patterns among these levels and how clients are using them, it could potentially prefetch data at nearby levels, anticipating and speeding up zooming operations.

Another possible prefetching enhancement could be an extension of the Prevailing Wind mode. PWP currently only prioritizes the requesting of a tile’s immediate neighbors based on the current “wind vane” measurement. Another method might extrapolate based on the “strength” of the wind and fetch tiles further in this direction.

8 Conclusion

We have presented several novel techniques that enable cooperative proxies to effectively cache and prefetch WMS data, using the example of GeoTorrent in the educational lab environment. We have introduced WHoMPyTS, which allows us to define an implicit and flexible pyramid tiling scheme. We have discussed WMSmooth, a query smoothing algorithm that enables caching and prefetching through the WHoMPyTS rule. We have proposed two methods of data prefetching based on statistical prediction, viz. the New Neighbors and Prevailing Wind models.

Our techniques provide dramatically reduced user response times, while greatly reducing impact on remote WMS servers. When applied in a cooperative proxy system, they enable the interactive use of remote WMS servers by a group of clients (e.g. in an educational lab). Use of these massive, rich, and publicly available datasets for educational purposes has previously been impractical. This work makes such use a reality and can do so for other environments as well.

9 Acknowledgments

We would like to thank the California State University for supporting this work.

References

1. Peggy Ann Brown. Cultivating community from the classroom. *American Forests*, 111(1), June 2005.
2. Thomas R. Baker. Internet-based GIS mapping in support of K-12 education. *The Professional Geographer*, 57(i1):44–47, February 2005.
3. Barbara Parmenter. GIS in the classroom. *Learning & Leading with Technology*, 28(i7):10, April 2001.
4. Jeffrey A. Bergamini and Michael Haungs. Geotorrent: Optimizing GIS web services for interactive educational use. *Proceedings of the UCGIS 2006 Summer Assembly*, June 2006.

5. Tom Barclay, Jim Gray, Eric Strand, Steve Ekblad, and Jeffrey Richter. Terraservice.net: An introduction to web services. Technical report, Microsoft Research, June 2002.
6. Tom Barclay, Jim Gray, and Don Slutz. Microsoft terraserver: a spatial data warehouse. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 29(2), May 2000.
7. Michael Potmesil. Maps alive: Viewing geospatial information on the WWW. *Proceedings of the Sixth International World Wide Web Conference*, April 1997.
8. Open Geospatial Consortium Inc. OpenGIS web map service (WMS) implementation specification (OGC 04-024) version 1.3. august 2004.. http://www.opengeospatial.org/specs.
9. Jonathan M. Borwein and Peter B. Borwein. *π and the AGM: A Study in Analytic Number Theory and Computational Complexity*. Wiley, 1987.
10. George Y. Liu and Gerald Jr. Maguire. Efficient mobility management support for wireless data services. *Proceedings of the 45th IEEE Vehicular Technology Conference, Chicago, IL*, July 1995.
11. Hassan Karimi and Xiong Liu. A predictive location model for location-based services. *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, November 2003.
12. Daniel Ashbrook and Thad Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5), October 2003.
13. Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: a decentralized peer-to-peer web cache. *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, July 2002.
14. Squid web proxy cache. (website). http://www.squid-cache.org.
15. Sergio Camarlinga, Ken Barker, and John Anderson. Multiagent systems for resource allocation in peer-to-peer systems. *Proceedings of the winter international synposium on Information and communication technologies*, 2004.
16. Sandra Dykes and Kay Robins. Limitations and benefits of cooperative proxy caching. *IEEE Journal on Selected Areas in Communications*, September 2002.
17. Prakash Linga, Indranil Gupta, and Ken Birman. A churn-resistant peer-to-peer web caching system. *Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems: in association with 10th ACM Conference on Computer and Communications Security*, 2003.