Towards Universal Mobile Caching[*]

Ganesh Santhanakrishnan, Ahmed Amer, Panos K. Chrysanthis
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
{ganesh, amer, panos}@cs.pitt.edu

ABSTRACT

In the context of mobile data access, data caching is fundamental for both performance and functionality. For this reason there have been many studies into developing energy-efficient caching algorithms suitable for specific mobile environments. In this papers, we present a novel caching policy, *Universal Mobile Caching* (UMC), which is suitable for managing object caches in structurally varying environments, and which is self-optimizing for changing workloads. UMC is based on a simple set of basic criteria which reflect a spectrum of possible caching policies. UMC has demonstrated the ability to provide caching benefits in the on-demand retrieval of web documents for the mobile web, wherein multiple levels of intervening caches can create adverse workloads for other general caching schemes. When considering the energy expended in servicing cache misses, UMC consistently demonstrated savings on the order of 10% to 15%. These energy savings are solely due to local per-node behavior, and do not include the potential reduction of power consumption, to less than half its normal levels, achievable due its enabling more effective multi-hop data transmission.

Categories and Subject Descriptors

C.2.8 [**Mobile Computing**]: Mobile Environments, Miscellaneous; H.3.m [**Information Storage and Retrieval**]: Miscellaneous—*Caching, Web Caching, Adaptive Caching*; D.4.3 [**Operating Systems**]: File Systems Management—*Distributed File Systems*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

Adaptive Caching, Web Caching, Greedy-Dual Algorithms, Mobile Web

[*]Supported in part by the National Science Foundation awards ANI-0325353 and ANI-0123705.

1. INTRODUCTION

One of the most prevalent techniques for improving storage system, network, and device performance is caching. In mobile environments, caching takes on an added importance when it can contribute to a greater reduction in the consumption of constrained resources such as power and network bandwidth, and by allowing disconnected operation. There have been many studies into developing a *better* caching algorithm, focusing on both improving the choice of item to replace, as well as developing techniques to model data access behavior and prefetch data. Recently, research efforts have produced caching policies that, in addition to optimizing a specific performance metric, attempt to automate policy parameter tuning. These efforts hope to effectively eliminate the need for an administrator or programmer to select a particular parameter, and thus allow the algorithm to adapt to the observed workload. Such adaptive and self-optimizing caching algorithms offer another advantage when we consider mobile environments, where users of mobile devices should not be expected to tune their devices in response to workload changes. The nature of the workload can vary dramatically depending on the current position of a mobile node in relation to other nodes and stations, and also depending on the current location and context of the mobile user. Such adaptive and self-optimizing caching algorithms offer another advantage when we consider mobile environments, where users of mobile devices should not be expected to tune their mobile devices and the nature of the workload can vary dramatically depending on the current position of a mobile node in relation to other caches on mobile or stationary nodes.

In a mobile environment, caching is beneficial both for reducing the access time to remote data, and for avoiding the power and resource consumption involved in retrieving such data. Caching is particularly effective for referential data that is unlikely to change frequently, and yet comes from a base too large to replicate across nodes. This is the environment created with the mobile web, and will form the main application example we consider in this work. Another advantage of caching data locally to mobile nodes is the ability to retrieve data from a nearby node, rather than from a more distant base station. In wireless environments, simply retrieving data using multiple short-range transmissions provides a reduction in overall energy consumed. And yet, this very behavior results in a structure that renders effective caching challenging. Depending on the current position of a mobile node, it is possible that the workload it observes is the result of misses in multiple intervening caches, a condition that even in wired environments is normally believed to render basic caching schemes useless without relying on inter-cache communication [23].

In this work we demonstrate how an adaptive self-optimizing caching scheme can still yield benefits in such a mobile multistaged environment (Figure 1). While such an environment may appear to be more likely in an infrastructureless wireless network setting (*e.g.*, mobile ad-hoc networks), we make no such assumption, as the need for multi-hop transmission can also arise with mobile infrastructure-based environments (*i.e.*, with backbone of base stations), for example, due to interference and dead spots. For our experiments we assume a general architectural model that can apply to both infrastructure and ad-hoc environments, where data may be retrieved through multiple hops to a base-station or satisfied from the local caches of mobile peers.

In our work we assume the on-demand broadcasting of web documents, and caches which depend solely on the observed data accesses to inform the caching policy. One of the earliest works to identify problems in future mobile wireless computing environments was Imielinski *et al* [5]. As early as 1995 they observed that energy constrained mobile hosts will need to communicate over a data transmission medium connecting to different data servers at different times. They also demonstrated that caching of frequently accessed data items will be an important technique that will reduce contention on narrow bandwidth, wireless channels. Unlike caching schemes, such as PIX [1], where a periodic broadcast of data is used (acting as a cache on the air), or schemes that employ opportunistic scheduling and prefetching, such as that of Aksoy *et al* [2], we consider only the demand-based retrieval of web documents in the mobile web. Our work also differs in its focus from caching schemes that exploit semantic knowledge, such as the processing of location-sensitive queries [19]. In this manner, we hope to focus on the most basic and general form of caching algorithms and largely emphasize the impact of the adaptive policy. In the best of our knowledge, no other adaptive policy exists in the context of mobile computing.

Our proposed caching scheme, called *Universal Mobile Caching*, is suitable for managing object caches in structurally varying environments, efficient enough to be implemented with minimal computational costs, and can be easily extended to capture all possible criteria influencing the performance of a cache replacement scheme. Our caching algorithm aims to be "Universal" at three different levels. The first is its ability to adapt to arbitrary workloads, as can arise with multiple mobile caches or due to fluctuations in user behavior. The second sense of "universal" comes from the algorithms aim to capture a wide (and eventually exhaustive) spectrum of possible caching policies within a simple and practical algorithm based on a small collection of observable workload traits, which we refer to as "base criteria." If used by itself on a mobile device, our novel caching scheme will give better performance, both in terms of energy and response time. The second advantage of UMC (the third dimension of universality) is in the event of data sharing between caches of mobile peers. UMC has the ability to adapt itself, thereby acting synergetically and cooperatively with other caching schemes. This is important, particularly in mobile computing, since cooperation may entail considerable energy costs and communication overheads.

The paper is structured as follows. We discuss background in Section 2. After that we go on to discuss our assumed system and power model in Section 3. In Section 4, we describe our Universal Mobile Caching policy in detail. After that we present results for trace-driven tests of the policy using varied workloads in Section 5. We conclude in Section 6 with a summary of our contribution.

2. BACKGROUND

Providing an effective and self-contained caching policy that is useful in the face of varying and unpredictable data access workloads, while simultaneously avoiding excessive complexity in the policy itself, is a challenging proposition, and a novel contribution of our work. We achieve this goal by providing a self-optimizing caching policy, based on our prior experiences in the application of machine learning techniques to caching problems, while ensuring that our algorithm is based on a minimal, simple, and efficiently computed set of basic criteria. To this end, our work builds upon the state of the art in adaptive caching, and we now introduce a brief background to the development of adaptive and self-optimizing caching schemes.

Simple Caching, Simple Criteria: The Least-Normalized-Cost replacement (LNC) [18] policy inserts the documents into a priority queue with a priority key. The problem with this policy is that it has tunable parameters which are workload dependent. Another policy known as Low Interference Recency Set (LIRS) [10] maintains a variable size LRU stack whose LRU page is the L(lirs)-th page that has been seen at least twice recently, where L(lirs) is a parameter. As suggested in the paper, the setting of L(lirs) to 1% of the cache size will be good for Independent Reference Model (IRM) workloads. But it does not perform as well for LRU Stack Depth Distribution (SDD) workloads.

The Frequency based replacement (FBR) [20] policy combines both the frequency of access and recency of access. It divides the LRU list into three sections, and maintains a counter for every document in the cache. The algorithm has several tunable parameters. In order to prevent cache pollution from stale pages with high reference counters, all the reference counters must be periodically rescaled. A class of policies known as Least Recently/Frequently Used (LRFU) [13] were shown to subsume the Least Recently Used (LRU) and Least Frequently Used (LFU) policies. These policies have an update rule which is a form of exponential smoothing that is widely used in statistics. It effectively balances both the LRU policy and LFU policies. There is an adaptive version of the policy known as Adaptive Least Recently/Frequently Used (ALRFU) [14] policy. This policy basically has a mechanism to dynamically adjust the parameter used in the basic LRFU policy. Again, both these LRFU schemes require tunable parameters. There has also been considerable work in the context of multiversion data broadcast, i.e., data broadcast in which more than one value is broadcast per data item, which is essential to support applications that require access to data sequences and are memory and power constrained [22]. Another adaptive cache invalidation algorithm for client/server mobile environments was proposed by Jin *et al* [12]. The proposed Bit-sequences algorithm uses adaptable mechanisms to adjust the size of the invalidation report to optimize the use of a limited communication bandwidth while retaining the effectiveness of cache invalidation. An interesting work that presents an energy-efficient cache invalidation method for a wireless mobile computer is Grouping with Cold Update-set Retention (GCORE) [24]

Self-Optimizing Caching Policies: A recently proposed self-tuning caching policy is known as the Adaptive Replacement Cache (ARC) [17]. It maintains two variable sized LRU lists. It captures both recency and frequency of access well, and provides an elegant, efficient and effective mechanism for combining them. It is intended as a page replacement policy, and so it does not consider the different delays in fetching a document, and variable object sizes, which are important factors in general object caching, as with Web content caching.

Our general approach of combining more than one constituent is most similar to Adaptive Caching using Multiple Experts (ACME) [3]. ACME uses a mixture of arbitrary policies which are treated as experts. Machine learning algorithms are applied to combine the recommendations of the different policies based on their ordering. We differ from ACME in two ways: by providing a more direct evaluation of each element's relative importance, and allowing the evaluation of an arbitrary selection of performance criteria. By using GD-Size variants as component algorithms we are able to proportionally scale the credits based on the normalized cost functions. ACME restricts policy information to orderings in exchange for a greater generality in terms of applicable policies. This means that GD-GhOST introduced in [21] produces a new evaluation function instead of a switching among policies or a rigid mixed-weighting policy. The Greedy-Dual * (GD-*) Web caching algorithm [11] is known to be superior to many other Web cache replacement policies, but our approach differs in that we attempt to optimize a user-specified combination, or selection, of performance goals.

As with GD-* we also differ from ACME in our ability to incorporate arbitrary combinations or selections of performance criteria. Finally, a motivating factor for these dynamic schemes, as for multi-queue algorithms [25], is to adapt policies automatically for cases where multiple caches can have adverse interactions. A recent work that addresses harmful cache interactions is that of Wong and Wilkes [23], where demotions were used as a mechanism to explicitly ensure cache exclusivity. Demotions require communication between these multiple caches, an assumption we do not make.

GD-GhOST & Universal Caching: Combining multiple algorithms can be performed by combining a multi-expert machine learning algorithm with virtual caches, and using different replacement algorithms for each virtual cache. This approach was first presented with ACME (Adaptive Caching using Multiple Experts) [3]. For machine learning algorithms, such as “weighted majority” and “share” [9], the term expert refers to any mechanism for offering an answer to the question. It can be any arbitrary algorithm for predicting the correct value, or even a sample answer *per se*. For cache replacement, the answer we seek is the identity of the object in the cache with the least likelihood of subsequent future access. The actual answer offered by the full algorithm is in fact a weighted combination of each expert's individual answer, and the contribution of the machine learning algorithm is to provide a mechanism to update these weights based on the relative performance of the individual experts. For ACME, the individual experts were full replacement algorithms, applied to virtual caches, and their performance was estimated based on the observed performance of these virtual caches. GD-GhOST (Goal-Oriented, Self-Tuning) caching is based on a combination of several Greedy Dual-Size variants [6], and attempts to satisfy a given performance goal using a fully adaptive combination of these individual component algorithms [21]. As with GD-GhOST algorithm, our proposed Universal Mobile Caching moves away from the use of arbitrary collections of component algorithms, and focuses on a reduced number of complimentary algorithms. Universal Mobile Caching, which we describe in the next section, offers further performance improvements, but its most notable feature is that it relies on a set of three simple properties of each cached object. These properties, or base criteria, were derived from the intuitions expressed in greedy-dual algorithms, and can be directly calculated or estimated for each cached object.

3. SYSTEM & POWER MODEL

We now discuss our system model, and the data transmission and power consumption behavior that we will be evaluating. We are assuming a model of layered caches among the mobile nodes, as illustrated in Figure 1. A mobile host in a lower layer communicates with any of the mobile hosts in the layer above it. More specifically in Figure 1, the mobile hosts *L2A1* and *L2A2* can communicate with their parent mobile host *L1A*, but not the base station directly. This can due to the use of an infrastructureless network, or due to wireless interference and dead spots necessitating such multi-hop communication. While earlier works have explicitly considered the benefits and implementation of multi-hop strategies, it is important to note that local caches within such schemes are either non-cooperative, or require explicit communication to act well together. The overheads of explicit communication are easy to overlook. In contrast, our policy allows the multiple local caches to coexist by autonomously self-tuning the local policy, which is independent of any workload changes caused by intervening local caches.

We have assumed a generic model of data transmission in the wireless domain. Basically, the energy consumed in the transmission of data is related to the volume of data and number of retrieval operations by the following equation:

$$Energy = K + [Power * VolumeOfMisses * Distance^2] \qquad (1)$$

In this equation, the constant K accounts for various costs associated with the network, the term Power stands for the power costs, the term Volume Of Misses stands for the volume of data misses at the cache and the term Distance stands for the distance of the mobile hosts. In practice, the average estimated power required for data transmission in terms of the volume of data transferred will vary depending on the model of wireless interface employed. Typical values include 1.875 Watts at 1 Mb/s to 3.12 Watts at 4.2 Mb/s [15,16]. Using this model we are able to estimate a reasonable range of values for the energy costs of data transmission, and we model the energy consumed for the wireless transmission of data to the local cache at each node. The exact values for the energy costs are computed as per Equation 1.

As we are considering the mobile web, the local cache at each node may in fact be subject to a workload that originates from the cache misses of other nodes' caches. For this reason, we model the miss and byte-miss behavior of caches that observe workloads filtered through zero or more preceding caching layers. The ideal result would be to observe our policy showing better cache performance than competing object caching algorithms, and to observe a positive impact on the energy consumed in wireless data transmission throughout the system. To test this hypothesis, we subject the caches to filtered workloads, and compare the energy consumed in the wireless transmission of data for our universal caching policy to that consumed using GDSF and GD-GhOST caching.

Such a relative comparison, conducted across a range of power consumption values, allows us to largely eliminate the contributions of any assumed constants and demonstrate the relative gains in energy conservation resulting from a more effective caching policy.

4. UNIVERSAL MOBILE CACHING

We now present our initial proposal for a *Universal Mobile Caching* (UMC) policy, which offers a self-optimizing replacement algorithm that is usable for general object caching, and yet is based on the simplest of base criteria. As discussed above, prior work has demonstrated the feasibility of constructing adaptive caching algorithms, but such efforts have either been restricted to page replacement and limited criteria, as with ARC [17], or else based on combinations of arbitrary replacement algorithms, as with ACME [3]. In contrast, Universal Mobile Caching uses a very simple set of object properties to select which objects (of varying size) will be removed from the cache.

Greedy-Dual (GD) cache replacement algorithms are based on ranking objects by their cost of retrieval (*H*-values), and replacing the objects with the lowest retrieval cost. To account for variable

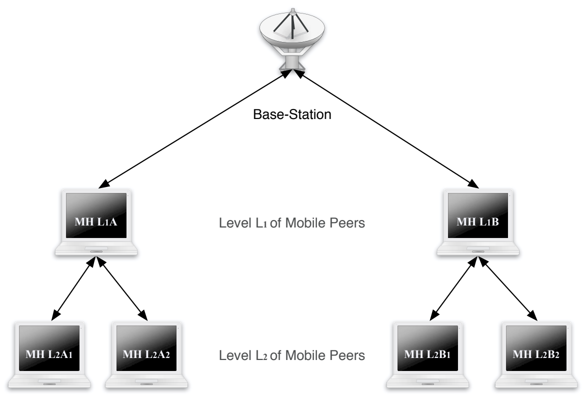


Figure 1: Data Transmission model of mobile hosts.

page sizes the retrieval cost is divided by the page size in Greedy Dual-Size (GD-Size) algorithms. For an object p, the H-value is calculated as:

$$H(p) = \frac{cost(p)}{size(p)} \tag{2}$$

The division by the page size accounts for the intuition that replacing larger pages frees up more space in the cache than removing smaller ones (the same intuition behind "SIZE" replacement that evicts the largest objects first). Different variants of GD-Size use different estimates of retrieval cost. For example, GD-1 assumes all objects have an equal replacement cost ($cost(p) = 1$), while GD-Size Frequency (GDSF) assumes the cost of replacement is proportional to the frequency of access of the object ($cost(p) = frequency(p)$). Yet another variant, GD-Size Packets, estimates the cost of retrieval as proportional to the number of packets that must be sent to transfer the object.

All variants of GD-Size algorithms are essentially providing different estimates for the same basic formula for evaluating the merit of retaining objects in the cache. This formula can be expressed as:

$$\begin{aligned} L(p) &: \text{Access-likelihood of } p \\ C(p) &: \text{Cost of retrieving } p \\ B(p) &: \text{Benefit from evicting } p \\ H(p) &= \frac{L(p) \cdot C(p)}{B(p)} \end{aligned} \tag{3}$$

All cache replacement algorithms can be interpreted as different simplifications and estimates of this formula. The benefit from evicting an object is best described by the cache capacity freed by such an eviction. This does not differ among the GD-Size variants. But from Equation 3, it should be apparent that increases in the replacement cost of an object, weighted by the likelihood of future access, are equivalent to *expected* costs incurred by evicting an object. The variations among the GD-Size variants appear in their estimates of this expected replacement cost. More importantly, we suggest that there is no cache replacement policy that does not perform an estimate of Equation 3 when deciding what to evict.

The Universal Mobile Caching algorithm draws on this small set of object properties (likelihood $L(p)$, cost $C(p)$, and benefit $B(p)$), and directly combines them to provide a self-optimizing and potentially comprehensive caching policy. Assuming we have three accurate estimators for these three properties, a possible mechanism to vary their combination is to provide a weight for each property. Unfortunately, it would be much simpler to automatically adjust such weights if these properties were a sum of "expert" values, and not a product. This is easily achieved by taking the log-sum of the product in Equation 3, giving us Equation 4.

$$H(p) = w_L \cdot log(L(p)) + w_C \cdot log(C(p)) - w_B \cdot log(B(p)) \tag{4}$$

In the following Section we see that using logarithms has a distinct impact on algorithm performance. For Equation 4 to be the basis of an adaptive caching algorithm, we need to select estimators of likelihood, cost, and benefit, as well as a mechanism to update the weights.

The choice of estimators can be debated, but for our experiments we selected the simplest possible set. The benefit from evicting object p was taken as the size of the object, while the cost of retrieval was also estimated as the size of the object. It should be pointed out that such conflicting uses of the same value are perfectly reasonable. If the cost of retrieving object p outweighs the benefit from evicting it, this can be reflected by increasing w_C relative to w_B. It should also be made clear that the estimator need not be an accurate estimate, merely that it should be proportional to the value being estimated.

Estimating likelihood of future access was done using two different estimators: frequency of access, and recency of last access. Because these two criteria were used as two distinct experts, we avoid any need to balance the mix of the two criteria (as is done manually in LRFU algorithms, and automatically in ARC [17]).

Similar to ACME and GD-GhOST, Universal Mobile Caching combines individual components using a master-algorithm approach [3]. Based on the performance of the component criteria, we distribute the credits for each criteria in the following manner.

$$Credit_c = (Perf_o - Perf_i) \cdot Credit_p$$

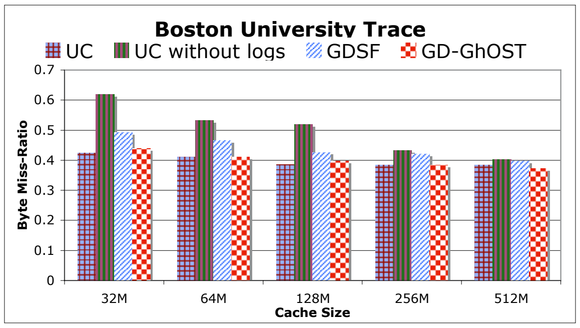
where:

$Credit_c$	:	The current credits for the criteria.
$Credit_p$	:	The previous credits for the criteria.
$Perf_o$	:	The overall combination/selection of hit ratio and byte hit ratio.
$Perf_i$	:	Criteria i's performance based on the combination/selection of hit ratio and byte hit ratio.

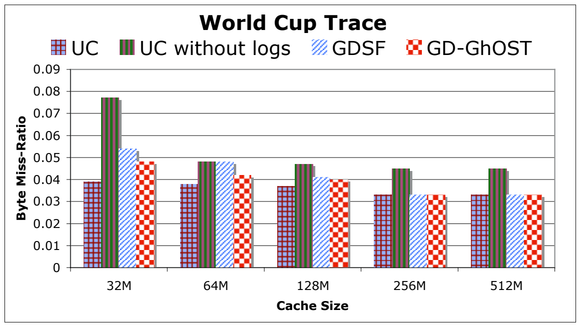
The credits are distributed among the criteria every time we evaluate their relative performance. This is not done with every access, but at a variable interval. There is no need to manually set this interval, as it's automatically adjusted based on variations in relative algorithm performance. If the workload is such that there is a consistent combination of criteria to maximize performance, then the period is automatically lengthened. As updates are needed more rapidly, the period is automatically reduced. This on-line update of credits ensures that at any instant in time, we are most likely to follow the leader among the criteria, for the metrics that are considered most important. When the best performing criterion degrades in performance, the redistribution of credits ensures that it does not degrade the overall performance. We have actually observed that Universal Mobile Caching can follow the best performance of the component criteria, and frequently exceeds it.

5. EXPERIMENTAL RESULTS

We conducted simulation-based experiments on real-world traces to evaluate the Universal Mobile Caching policy and the effect of web-cache filtering effects on it, particularly in comparison to GDSF and the adaptive, yet more complex, GD-GhOST policy.

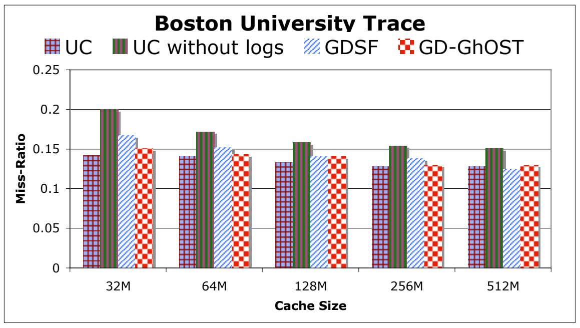


(a) *Boston University Trace*

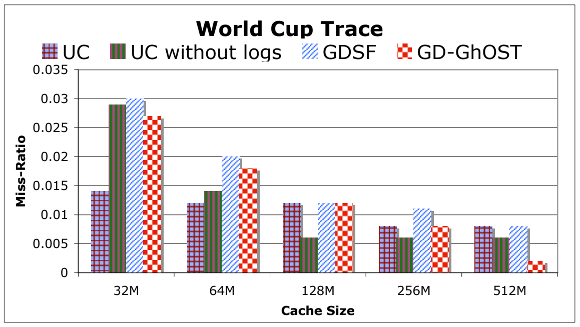


(b) *World Cup 98 Trace*

Figure 2: Average byte miss ratio results for varied cache sizes.



(a) *Boston University Trace*



(b) *World Cup 98 Trace*

Figure 3: Average miss ratio results for varied cache sizes.

We tested the ability of Universal Mobile Caching to exhibit performance similar to the best policy for the selected performance metric. Specifically we evaluated *hit ratios*, *byte hit ratios*, and *estimated power consumed for data transmission.*

We also present the impact of cache filtering to demonstrate the resilience of our Universal Mobile Caching policy to the harmful cache interactions observed by Wong and Wilkes [23], and its ability to provide performance benefits in spite of such an adverse workload.

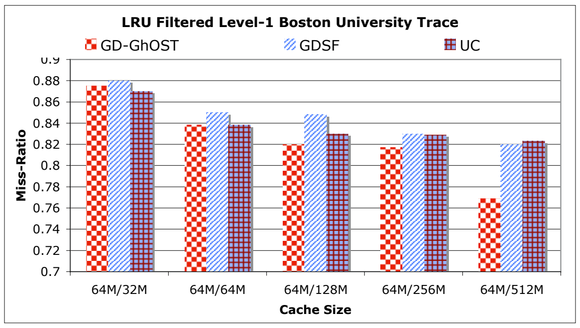
5.1 Workload Description

Experiments were conducted using traces run against a simulated cache. The three Greedy Dual-Size algorithms (GD-Size(1), GD-Sizepackets), and GDSF) were implemented, along with the GD-GhOST and Universal Mobile Caching policies. For GD-GhOST and Universal Mobile Caching the credits for the components (algorithms or criteria) were updated on-line using the proportional weighted averaging described in Section 4. In other words, no warm-up or training period was allowed for the adaptive caches.

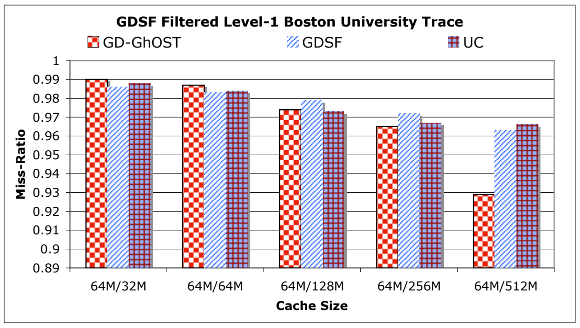
For our tested workloads, the best performing Greedy Dual-Size algorithm was GDSF, to which we restrict our graphed results for the sake of clarity. For Universal Mobile Caching, the core criteria were implemented using their own estimators and each operating on its own ghost cache. This was a compromise to simplify testing and comparison with GD-GhOST, but ultimately the goal will be to completely eliminate the need for ghost caches by evaluating criteria values in relation to future accesses. The criteria were also implemented without using their log based components. We refer to the Universal Caching policy thus implemented as the Universal Caching policy without logs. The credits for the individual replacement criteria were updated on-line using the proportional weighted averaging described in Section 4. As described above, the trial period for updating the credits was dynamically adjusted and required no *a priori* settings. We tested with different cache sizes and using both client and proxy web traces. Specifically, we used traces from Boston University [8] and the 1998 World Cup [4]. The Boston University traces contain records of the HTTP requests and user behavior of a set of Mosaic clients running in the Boston University Computer Science Department, spanning from 21 November 1994 through 8 May 1995. During the data collection period a total of 9,633 Mosaic sessions were traced, representing a population of 762 different users, and resulting in 1,143,839 requests for data transfer. There were 5-32 workstations. The World Cup 98 data set consists of all the requests made to the 1998 World Cup Web site between April 30, 1998 and July 26, 1998. During this period of time the site received 1,352,804,107 requests. In Figures 2 through 7 the criss-cross pattern represents the Universal Mobile Caching policy, the slanted lines to the left represent the GDSF policy and the checkerboard pattern represents the GD-GhOST policy. In Figures 2 and 3 the solid pattern represents the Universal Mobile Caching policy without the logarithmic components.

5.2 Basic Caching Results

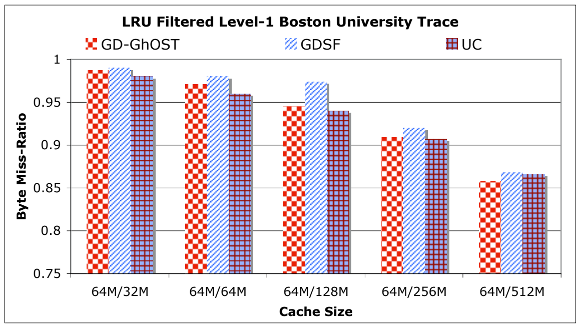
We present the byte miss ratios in Figure 2 and the miss ratios for



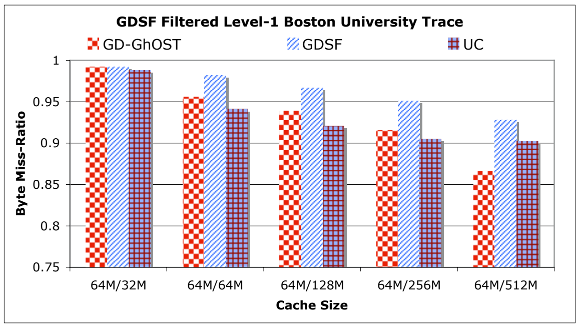
(a) *Miss ratios for LRU filtered level-1 Boston University Trace*



(a) *Miss ratios for GDSF filtered level-1 Boston University Trace*



(b) *Byte Miss ratios for LRU-filtered level-1 Boston university Trace*



(b) *Byte Miss ratios for GDSF-filtered level-1 Boston University Trace*

Figure 4: Web cache filtering results for LRU filtered level-1 Boston University Trace.

Figure 5: Web cache filtering results for GDSF filtered level-1 Boston University Trace.

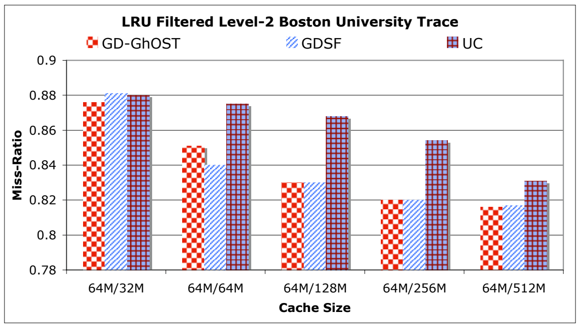
the Universal Mobile Caching policy in Figure 3. These results are for cache sizes of 32 MB, 64 MB, 128 MB, 256 MB and 512 MB. We compare the performance of the Universal caching policy based Equation 4 (with logs), to that of the Universal Caching policy without logs (Equation 3), the GD-GhOST policy [21], and the Greedy Dual-Size Frequency policy [7].

We see that our Universal Mobile Caching policy (with logs) based on Byte Hit-Ratio, performs on par with the other competing variants exceeding them on several occasions. When compared based on Hit-Ratio, it performs among the top two policies while performing better than others on more than a couple of occasion's. The Universal Mobile caching policy not based on logs performs slightly poorly than that based on logs.

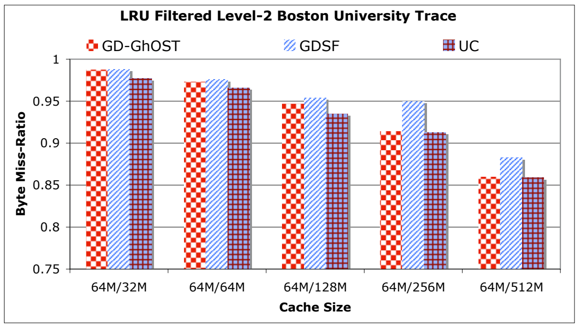
The striking result from Figure 2 is that our Universal Mobile Caching policy consistently outperforms GDSF policy in terms of byte hit ratios. From Figure 3, we see that for simple hit ratios, where all object misses are equal regardless of object size, Universal Caching performs only slightly worse for the largest of cache sizes. While the overall value ranges differ for the two workloads, it is important to note that the trends are consistent regardless of whether the workload is from a server, or from a client-side proxy.

5.3 Mobile Web Cache Filtering

Recognizing that in most configurations, the request path from clients to servers can involve multiple caches, at least two at browser/proxy and at the server, that is why we performed experiments with web cache filtering. The Web cache filtering effects described here were simulated as follows: We allowed the Boston University and World Cup 98 traces to pass through a cache employing a specific policy. Then we ran the filtered trace through the cache employing the GD-GhOST policy, its constituents and the Universal Mobile Caching policy. For clarity and space we present the results for the Boston University trace, and limit the constituents to the GDSF policy, which was the best performaing GD-Size variant for the workloads tested. While the results presented here are representative, we felt that filtering the Boston University traces would most realistically reflect a server-caching scenario where client requests have passed through prior caching levels. We performed the simulation for the mobile web cache filtering effects using varying cache sizes. We experimented with LRU filtering caches varying from size 32 MB to 512 MB, and target caches varying across the same range. Without loss of generality, we fixed the cache size of the filtering cache as 64 MB since we felt that it was representative. We also experimented with the GD-Size(1), GD-Size(Packets), and GDSF filtered caches of size 64 MB using cache sizes of 32 MB, 64 MB, 128 MB, 256 MB, and 512 MB. We also experimented with two levels of cache filtering employing the same policies, and using the same cache sizes as described above.

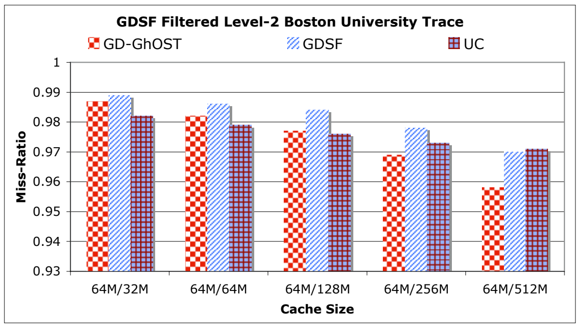


(a) *Miss ratios for LRU filtered level-2 Boston University Trace*

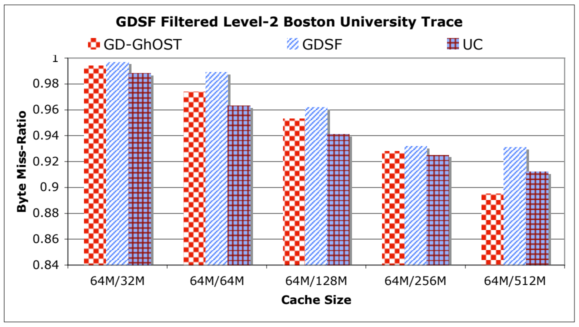


(b) *Byte Miss ratios for LRU-filtered level-2 Boston University Trace*

Figure 6: Web cache filtering results for LRU filtered level-2 Boston University Trace.



(a) *Miss ratios for GDSF filtered level-2 Boston University Trace*



(b) *Byte Miss ratios for GDSF filtered level-2 Boston University Trace*

Figure 7: Web cache filtering results for GDSF filtered level-2 Boston University Trace.

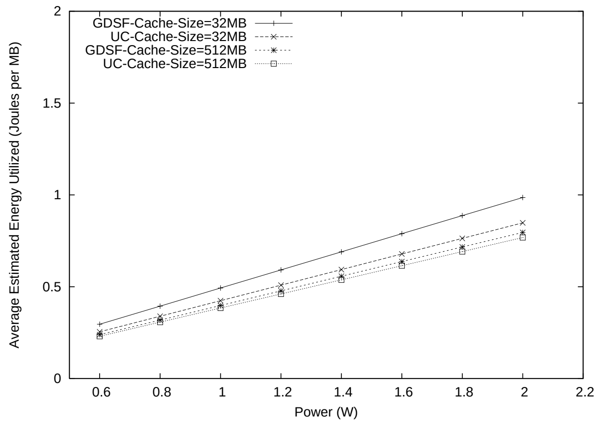
We compare the results for byte miss ratios for the Universal Mobile Caching, GD-GhOST, and GDSF policies. Results for both levels of filtered workloads are shown in Figures 4, 5, 6 and 7. The miss ratios are generally much higher under these conditions, but this is true regardless of policy, and the Universal Caching results are again very encouraging. Our policy consistently outperforms GDSF and the GD-GhOST policy for small and medium sized caches, while performing only slightly worse for only the largest cache capacities where miss rates are exceedingly low.

We now present the power results for our Universal Mobile Caching policy compared against the GDSF policy. We have incorporated the parameters as mentioned in our data transmission and power consumption model. We have assumed power consumption values for data transmission rates as suggested by Lorch *et al* [15] and Mahesri *et al* [16]. The power consumption values for the data transmission rates range from 1.875 W at 1 Mbps, 2.55 W at 2.9 Mbps and 3.12 W at 4.2 Mbps. We have shown the average estimated energy utilized for a range of power values with the aforementioned three values as the baseline.

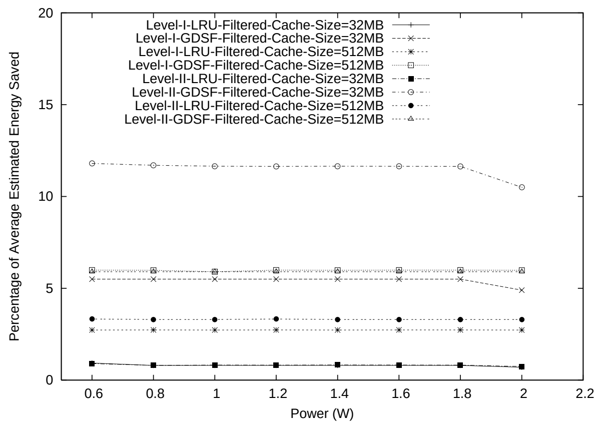
In figure 8(a), we show the plot of Average Estimated Energy utilized in terms of Joules per Megabit against the Power usage in terms of Watts. We show the results for the unfiltered Boston University trace comparing GDSF and Universal Mobile Caching policy for cache sizes of 32 MB and 512 MB. We can clearly see that the energy utilized for Universal Mobile Caching policy for cache size of 32 MB is much less than that of the energy utilized for Greedy Dual Size Frequency Caching policy for cache size of 32 MB. The difference between the energy utilized for the Universal Mobile Caching policy for cache size of 512 MB and the energy utilized for the GDSF caching policy for cache size of 512 MB is negligible.

In figure 8(b), we show the plot of the percentage of average estimated energy saved for the filtered Boston University Trace through two different levels of filtering, as well as employing two different filtering policies of Least Recently Used (LRU) and GDSF. We show the percentage savings when employing the Universal Mobile Caching policy as opposed to that of the Greedy Dual Size Frequency caching policy. We can see here that there is a significant energy savings, which is also consistent over the cache sizes from 32 MB through 512 MB.

By using our universal mobile caching policy we are able to achieve a savings of up to 12% in terms of energy used to transmit data. We should also point out that this does not include the potential halving of such energy due to the enabling of multi-hop transmission, but reflects only the energy saved at the individual nodes due to the reduced volume of data that needs to be transmitted. As such, we offer a very conservative estimate of the energy savings achievable by employing universal mobile caching in mo-



(a) *Average Estimated Energy Utilized for Boston University Trace*



(b) *Percentage of Average Estimated Energy Saved for Level-I and Level-II filtered Boston University Trace*

Figure 8: Power Savings Results for Boston University Trace

bile web environments.

6. CONCLUSIONS

Our major contribution in this paper is a Universal Mobile Caching policy that demonstrates its ability to use a set of simple criteria as a basis for auto-tuning the caching policy, and as a result is capable of achieving resource savings in the most adverse multi-level caching scenarios. In summary, we can see that Universal Mobile Caching performs very well, achieving the highest byte hit ratios among the best competing algorithms, and even against a far more complex self-optimizing algorithm. The comparison was based on web workloads, but they differed greatly in their nature due to the imposition of different prior levels of caching. In a mobile web environment, with multiple levels of intervening caches, our policy was able to demonstrate consistent power savings over a range of possible power consumption models.

7. REFERENCES

[1] ACHARYA, M. F., AND ZDONIK, S. Dissemination-based data delivery using broadcast disks. *IEEE PCM 2*, 6 (1995).

[2] AKSOY, D., FRANKLIN, M. J., AND ZDONIK, S. B. Data staging for on-demand broadcast. In *Proc. of the 27th Intl. Conf. on Very Large Data Bases* (2001).

[3] ARI, I., AMER, A., GRAMACY, R., MILLER, E. L., BRANDT, S., AND LONG, D. D. E. Adaptive Caching using Multiple Experts. In *Proc. of the Workshop on Distributed Data and Structures* (2002).

[4] ARLITT, M., AND JIN, T. 1998 World CupWebSite Access Logs - Available at http://www.acm.org/sigcomm/ita/.

[5] BARBARA, D., AND IMIELINSKI, T. Sleepers and workaholics: Caching strategies in mobile environments. *VLDB J. 4*, 4 (1995).

[6] CAO, P., AND IRANI, S. Cost aware WWW proxy caching algorithms. In *Proc. of USENIX symp. on Internet Technologies and Systems* (1997).

[7] CHERKASOVA, L. Improving WWW proxies performance with Greedy-Dual-Size-Frequency caching policy. In *HP TR* (1998).

[8] CUNHA, C. A., BESTAVROS, A., AND CROVELLA, M. E. Characteristics of WWW Client Traces. In *Boston University Dept. of Computer Science, TR-95-010* (1995).

[9] HERBSTER, M., AND WARMUTH, M. K. Tracking the best expert. In *Proc. of the 12th Intl. Conf. on Machine Learning* (1995).

[10] JIANG, S., AND ZHANG, X. LIRS: an efficient Low Inter-reference Recency Set replacement policy to improve buffer cache performance. In *Proc. of the 2002 ACM SIGMETRICS Intl. Conf. on Measurement and modeling of computer systems* (2002).

[11] JIN, S., AND BESTAVROS, A. GreedyDual* web caching algorithm: Exploiting the two sources of temporal locality in web request streams. In *Intl. Journal on Computer Communications,* (2001).

[12] JING, J., ELMAGARMID, A., HELAL, A. S., AND ALONSO, R. Bit-sequences: an adaptive cache invalidation method in mobile client/server environments. *MONET 2*, 2 (1997).

[13] LEE, D., CHOI, J., KIM, J., NOH, S., MIN, S., CHO, Y., AND KIM, C. LRFU: A spectrum of policies that subsumes the Least Recently Used and Least Frequently Used policies. In *IEEE TOC* (1990).

[14] LEE, D., CHOI, J., KIM, J.-H., NOH, S. H., MIN, S. L., CHO, Y., AND KIM, C. S. On the existence of a spectrum of policies that subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) policies. In *Proc. of ACM SIGMETRICS Conf.* (1999).

[15] LORCH, J. R., AND SMITH, A. J. Software strategies for portable computer energy management. *IEEE PCM 5*, 3 (1998).

[16] MAHESRI, A., AND VARDHAN, V. Power consumption breakdown on modern laptop. In *Proc. of the 4th workshop on Power-Aware Computer Systems* (2004).

[17] MEGIDDO, N., AND MODHA, D. S. ARC: a self-tuning, low overhead replacement cache. In *Proc. of FAST* (2003).

[18] PETER SCHEUERMANN, JUNHO SHIM, R. V. A case for delay-conscious caching of web documents. In *Computer Networks and ISDN Systems* (1997).

[19] REN, Q., AND DUNHAM, M. H. Using semantic caching to manage location dependent data in mobile computing. In *Proc. of the 6'th Annual Intl. Conf. on Mobile computing and networking* (2000).

[20] ROBINSON, J. T., AND DEVARAKONDA, M. V. Data cache management using frequency-based replacement. In *Proc. of ACM SIGMETRICS Conf.* (1990).

[21] SANTHANAKRISHNAN, G., AMER, A., CHRYSANTHIS, P. K., AND LI, D. GD-GhOST: A goal-oriented self-tuning caching algorithm. In *Proc. of the 19th Symp. on Applied Computing (SAC)* (2004).

[22] SHIGILTCHOFF, O., CHRYSANTHIS, P. K., AND PITOURA, E. Adaptive multiversion data broadcast organizations. *Information Systems 29*, 6 (2004).

[23] WONG, T. M., AND WILKES, J. My cache or yours? Making storage more exclusive. In *Proc. of the USENIX Annual Technical Conf.* (2002).

[24] WU, K.-L., YU, P. S., AND CHEN, M.-S. Energy-efficient mobile cache invalidation. *Distrib. Parallel Databases 6*, 4 (1998).

[25] ZHOU, Y., PHILBIN, J. F., AND LI, K. The Multi-Queue replacement algorithm for second level buffer caches. In *Proc. of USENIX Annual Technical Conf.* (2001).