Adaptive Power-Aware Cache Management for Mobile Computing Systems

Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
gcao@cse.psu.edu

ABSTRACT

Prefetch can be used to reduce the query latency and improve the bandwidth utilization of cache invalidation schemes. However, prefetch consumes power. In this paper, we propose a power-aware cache management to address this issue. Based on a novel *prefetch-access ratio* concept, the proposed scheme can dynamically optimize performance or power based on the available resources and performance requirements. Simulation results show that our solution not only improves the cache hit ratio, the throughput, and the bandwidth utilization, but also reduces the query delay and the power consumption.

Keywords

Power-aware, invalidation report, caching, mobile computing.

1. INTRODUCTION

Caching frequently accessed data items on the client side is an effective technique to improve performance in a mobile environment. Classical cache management strategies may not be suitable for mobile environments due to the disconnection and mobility of the mobile clients. Barbara and Imielinski [1] proposed a cache solution which is suitable for mobile environments. In this approach, the server periodically broadcasts an *invalidation report (IR)* in which the changed data items are indicated. Rather than querying the server directly regarding the validation of cached copies, the clients can listen to these IRs over the wireless channel, and use them to validate their local cache. The IR-based solution is attractive because it can scale to any number of clients who listen to the IR. However, the IR-based solution has some major drawbacks such as long query latency and low bandwidth utilization. In our previous work [3], we addressed the first problem with a UIR-based approach. In this approach, a small fraction of the essential information (called updated invalidation report (UIR)) related to cache invalidation is replicated several times within an IR interval, and hence the client can answer a query without waiting until the next IR. However, if there is a cache miss, the client still needs to wait for the data to be delivered. To increase the cache hit ratio and reduce the bandwidth consumption, clients intelligently prefetch the data that are most likely used in the future. However, prefetch consumes power. In this paper, we propose a power-aware cache management to address this issue. Based on a novel *prefetch-access ratio* concept, the proposed scheme can dynamically optimize performance or power based on the available resources and performance requirements. Compared to previous schemes, our solution not only improves the cache hit ratio, the throughput, and the bandwidth utilization, but also reduces the query delay and the power consumption.

2. POWER-AWARE CACHE MANAGEMENT

2.1 Efficiently utilize the bandwidth

To improve the cache hit ratio, clients prefetch data that may be used in the near future. To save power, clients may only wake up during the IR broadcasting period, and then how to prefetch data becomes an issue. As a solution, after broadcasting the IR, the server first broadcasts the id list of the data items whose data values will be broadcast next, and then broadcasts the data values of the data items in the id list. Each client should listen to the IR if it is not disconnected. At the end of the IR, a client downloads the id list and finds out when the interested data will come and wakes up at that time to download the data. With this approach, power can be saved since clients stay in the doze mode most of the time; bandwidth can be saved since the server may only need to broadcast the updated data once. Since prefetching also consumes power, it is very important to identify which data should be included in the id list. Since the server does not maintain any information about the clients, it is very difficult, if not impossible, for the server to identify which data is hot. To save broadcast bandwidth, the server does not answer the client requests immediately; instead, it waits for the next IR interval. After broadcasting the IR, the server broadcasts the id list of the data items that have been requested during the last IR interval. In addition, the server broadcasts the values of the data items in the id list.

2.2 An Adaptive Prefetch Approach

Since prefetching also consumes power, we investigate the tradeoff between performance and power, and propose an adaptive scheme to efficiently utilize the power.

Each client may have different available resources and performance requirements, and these resources such as power may change with time. For example, suppose the battery of a laptop lasts three hours. If the user is able to recharge the battery within three hours, power consumption may not be an issue, and the user may be more concerned about the performance aspects such as the query latency. However, if the user cannot recharge the battery within three hours and wants to use it a little bit longer, then power consumption becomes a serious concern. To address this issue, the system monitors the power level. When the power level drops below a threshold, power consumption becomes the primary concern. If query latency is more important than power consumption, the client should al-

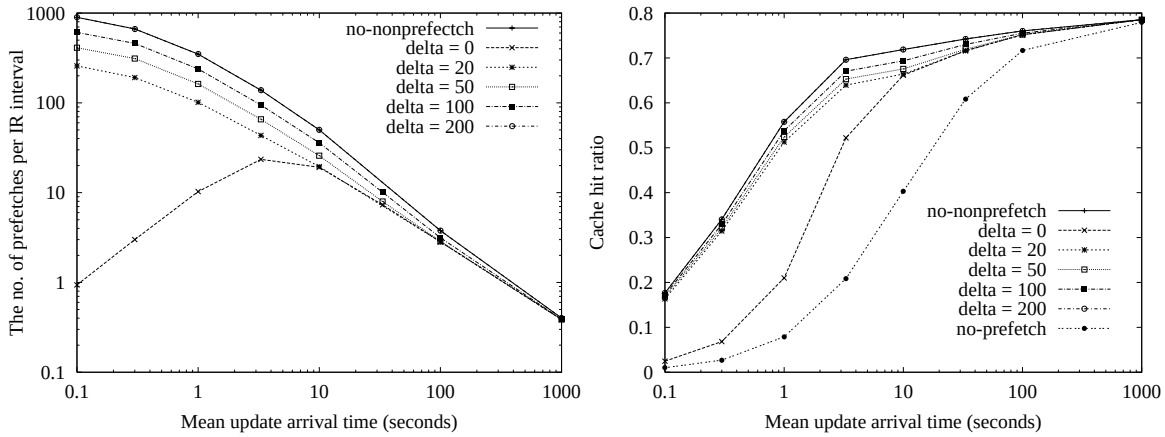

Figure 1: The effects of δ

ways prefetch the interested data. However, when the power drops to a threshold, the client should be cautious about prefetching.

There are two solutions to reduce the power consumption. As a simple solution, the client can reduce its cache size. With a smaller cache, the number of invalid cache entries reduces, and the number of prefetches drops. Although small cache size reduces prefetch power consumption, it may also increase the cache miss ratio, thereby degrading performance. In a more elegant approach, the client marks some invalid cache entries as *non-prefetch* and it will not prefetch these items. Intuitively, the client should mark those cache entries that need more power to prefetch, but are not accessed too often.

The adaptive prefetch approach: In order to implement the idea, for each cached item, the client records how many times it accessed the item and how many times it prefetched the item during a period of time. The *prefetch-access ratio (PAR)* is the number of prefetches divided by the number of accesses. If the PAR is less than 1, prefetching the data is useful since the prefetched data may be accessed multiple times. When power consumption becomes an issue, the client marks those cache items which have $PAR > \beta$ as *non-prefetch*, where $\beta > 1$ is a system tuning factor. The value of β can be dynamically changed based on the power consumption requirements. For example, with a small β, more energy can be saved, but the cache hit ratio may be reduced. On the other hand, with a large β, the cache hit ratio can be improved, but at a cost of more energy consumption. Note that when choosing the value of β, the uplink data request cost should also be considered.

When the data update rate is high, the PAR may always be larger than β, and clients cannot prefetch any data. Without prefetch, the cache hit ratio may be dramatically reduced and resulting in poor performance. Since clients may have a large probability to access a very small amount of data, marking these data items as pre-fetch may improve the cache hit ratio and does not consume too much power. Based on this idea, when $PAR > \beta$, the client marks δ number of cache entries which have high access rate as $prefetch$.

Since the query pattern and the data update distribution may change over time, clients should measure their access rate and PAR periodically and refresh some of their history information. Assume N^x_{acc} is the number of access times for a cache entry d_x. Assume $N^x_{c\_acc}$ is the number of access times for a cache entry d_x in the current evaluation cycle. The number of access times is calculated by

$$N^x_{acc} = (1 - \alpha) * N^x_{acc} + \alpha * N_{c\_acc}$$

where $\alpha < 1$ is a factor which reduces the impact of the old access frequency with time. Similar formula can be used to calculate PAR.

3. PERFORMANCE EVALUATION

In order to evaluate the efficiency of various invalidation algorithms, we develop a model which is similar to that employed in [2, 3]. It consists of a single server that serves multiple clients. The database can only be updated by the server whereas the queries are made on the client side. Figure 1 shows the effects of $delta$ on the number of prefetches and the cache hit ratio. As can be seen, the no-prefetch scheme, which does not prefetch data, has the lowest cache hit ratio, whereas the no-nonprefetch scheme has the highest cache hit ratio, but the highest number of prefetches. As $delta$ changes, the number of data items to be marked as prefetch changes, and resulting in a tradeoff between cache hit ratio (delay) and the number of prefetches (power).

4. CONCLUSIONS

Based on a novel *prefetch-access ratio* concept, we presented an adaptive power-aware cache management scheme for mobile environments. Simulation results verified that our scheme can keep the advantage of prefetch with low power consumption.

5. REFERENCES

[1] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies for Mobile Environments," *ACM SIGMOD*, pp. 1–12, 1994.

[2] G. Cao, "On Improving the Performance of Cache Invalidation in Mobile Environments," *ACM/Baltzer Mobile Networks and Application (MONET)*, to appear.

[3] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," *IEEE Transactions on Knowledge and Data Engineering*, to appear (A preliminary version appeared in ACM MobiCom'00).