An Adaptive Callback Cache Access for Wireless Internet

Yang Xiao
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
e-mail: yangxiao@ieee.org

Hui Chen
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
e-mail: hchen2@memphis.edu

Abstract—We propose a two-level adaptive callback access mechanism for wireless data access. In the first level, cache size in a mobile termination is adaptively adjusted based on update-to-access-ratio. In the second level adaptation, when an object is updated at the server, whether to send the object directly or an invalidation message, adaptively depends on the object size. We analytically model cost function, and the optimal cache size and the optimal adaptation threshold value are obtained simultaneously. Both simulations and analytical results are used to study the performance.

Keywords—Cache, Callback, Wireless Networks

I. INTRODUCTION

Wireless Application Protocol (WAP) and iSMS have been developed to support wireless Internet applications [1-3]. A mobile user can access Internet web applications through a client/server model. Cache can also be used in the client mobile terminal to buffer frequently/recently used data objects sent from the server. For some applications, a strongly consistent data access protocol must be exercised between the client and the server [3], such as poll-each-read (PER) and callback (CB). In the PER mechanism, if the data object that a client requests is not in the cache, the client requests it from the server; otherwise, the client always asks the server to check its validity; if it is valid, the server informs the client and the client uses the data object; if it is not valid, the server sends the updated data object to the client to replace the out-of-date data object in the client cache. In the CB mechanism, when a data object is to be changed at a server, the server informs the client; and the client marks the data object in the cache as invalid and the space can be reclaimed to accommodate other data objects.

In this paper, we have three observations for the CB scheme: a) when the update-to-access-ratio (UAR) is very large so that objects in the cache are always obsolete such as stock information, the cache should not be used, and therefore the cache size should be set to zero; b) when the UAR is zero so that every object in the cache is always valid, the cache size should be set to the maximum physical cache size of the mobile terminal; and c) when a data object is updated at the server and its size is very small, the data object instead of an invalidation message should be sent back to the client.

Therefore, we propose an adaptive access mechanism, called optimal callback with two-level adaptation. We provide an analytical model and study optimality of the cache size and the object size threshold simultaneously, and U-threshold defined in the next section is obtained analytically. Simulations are conducted to validate the analytical model.

II. CACHE ACCESS MECHANISMS

A. Push, Invalidation, and Fetch

There are several CB schemes. The scheme described in Introduction as well as in [3] is referred as to Invalidation scheme in this paper. We define another callback scheme called Push scheme, in which when a data object is updated at the server, the object is sent to the client instead of an invalidation message. The Invalidation scheme and the Push scheme are two special cases of the callback scheme. We further define another scheme called Fetch scheme, in which the cache is disabled. Specifically, when a client needs a data object, it always request the data object from the server that sends the data object to the client; when an object is updated at the server, and no further procedure is needed.

B. Optimal callback with two-level adaptation

In the proposed optimal callback scheme, we consider both cache size and object size. Assume that the maximum physical cache size in a mobile terminal is M in terms of the number of objects it can hold. The range of the cache size is [0, M]. In the proposed scheme, the cache size is adaptively adjusted based on update-to-access-ratio (UAR), defined as the average number of updates per data object access. One extreme case is that when the UAR is very large so that objects in the cache are always obsolete, the cache should not be used, and therefore the cache size should be set to zero. In other words, the Fetch scheme should be used. Note that the Fetch scheme can be treated as a special case when the cache size is zero. Another extreme case is that when the UAR is zero, the cache size should be M. Under other situations, the cache size is dynamically changed between 0 and M. Define U-threshold of the UAR for any object, a particular important threshold, as a UAR value, beyond which the object should be not cached at all. If all objects’ UARs are beyond U-threshold, the cache should be disabled, i.e., the optimal cache size is zero.
In the second-level adaptation, when an object is updated at the server, it is sent directly to the client if the object size is smaller than a threshold, called Push Threshold ($T$); otherwise, an invalidation message is sent to the client. The proposed scheme is an adaptive scheme between the Invalidation scheme and the Push scheme from this aspect.

Our major goal is to minimize the total traffic involved between the server and the client per data object access with an optimal cache size $K$ and an optimal $T$ value simultaneously, based on the value $U$ in later sections.

III. ANALYTIC MODELS

Based on [3], we obtain an analytic model for the proposed adaptive CB cache mechanism under Least Recently Used (LRU) replacement policy. Then, we formulate a cost function.

A. An analytic model

We have following assumptions: 1) sizes of mobile objects follow a density distribution $g(x)$ ($x>0$); 2) a client can hold $K$ objects in its local cache regardless object sizes; 3) accesses to a data object $O_k$ ($k=1,...,N$) follow Poisson distribution with rate $\mu_k$, where $\mu_k = \mu$ ($k=1,...,N$); 4) Interarrival times between updates of a data object $O_k$ ($k=1,...,N$) follow a general distribution with the density function $f_k(t)$ with the mean $1/\lambda_k$, and the Laplace transform $f_k*(s) = \int_0^\infty f_k(t)e^{-st}dt$, where $\lambda_k = \lambda$ and $f_k(t)=f(t)$ for $(k=1,...,N)$.

A data object $O_j$ in the cache of the client is moved to the top of the cache when $O_j$ is accessed. Consider Fig. 1, where the current access to $O_j$ occurs at time $t_0$, the previous access to $O_j$ occurs at time $t_0$, the previous update of $O_j$ occurs at time $t_1$, and the next update of $O_j$ occurs at time $t_2$. Consider another object $O_j$ in Fig.1, where the previous access to $O_j$ occurs at time $t_3$, the next access to $O_j$ occurs at time $t_4$, the previous update of $O_j$ occurs at time $t_5$, and the next update of $O_j$ occurs at time $t_6$. We have $\min \{t_0, t_1, t_2, t_3\} < t_4 < \min \{t_5, t_6, t_7\}$. Therefore, $t_0$ and $t_3$ have exponential distributions with mean $1/\mu$, and $t_1$ and $t_2$ have same general distributions with mean $1/\lambda$. $T_0$ has the same distribution as $\tau_3$ based on the excess life theorem [4] and the memoryless property of the exponential distribution. $T_1$ and $T_2$ have density functions $r_{T_1}(t)$ and $r_{T_2}(t)$, respectively, where $r_{T_1}(t) = r_{T_2}(t) = \lambda^t_s f(x)dx$ and their Laplace transforms $r^*_1(s) = r^*_2(s) = \frac{\lambda}{s}[1-f^*(s)]$; $\tau^*_1$ and $\tau^*_2$ have distribution functions $R_{\tau^*_1}(t)$ and $R_{\tau^*_2}(t)$, respectively, where $R_{\tau^*_1}(t) = R_{\tau^*_2}(t) = R(t)$. Let $h(t)$ be the probability that the cache ranking of $O_j$ is affected by an access to $O_j$ under the condition $\tau_0 = t$. Let $P_{Adaptive\_CB}$ be the probability that an effective cache hit occurs at time $t_3$ when the adaptive CB scheme is adopted, respectively. We have

\[ h(t) = 1 - e^{-\mu t} - Pr(L > T) \int_{\tau^*_3 = 0}^{\tau^*_3} \mu e^{-\mu \tau^*_3} R(\tau^*_3) d\tau^*_3 \] (1)

\[ P_{Adaptive\_CB} = \sum_{k=0}^{K-1} \int_0^{\infty} \left[ \beta(\tau_0)^k \int_0^{\infty} \left[ 1 - \beta(\tau_0)^k \right]^{N-k-1} \right] d\tau_0 \] (2)

\[ C_{Adaptive\_CB} = \begin{cases} 1 & \text{if } R(T) \leq T \land R(T) \geq \mu \tau_0 \\ 0 \end{cases} \] (3)

\[ C_{Adaptive\_CB} = \begin{cases} 1 & \text{if } R(T) \leq T \land R(T) \geq \mu \tau_0 \\ 0 \end{cases} \] (4)

B. Cost function

The request message, the acknowledgement messages, and the invalidation message are denoted as $L_{Req}$, $L_{ACK}$, and $L_{Inv}$, respectively. An object size $L$ is a variable value. Let $M(L)$ be a cost when the message size is $L$. Let $E(U)$ denote the average number of updates of an object per data access to this object. Let $E(L)$ denote the mean value of object sizes. Let $E(L_{Inv})$ denote the mean value of object sizes that are smaller than $T$. Let $E(U)$ denote the average number of updates of an object per data access to this object. The cost function $C_{Adaptive-CB}$ per access event is

\[ C_{Adaptive\_CB} = \begin{cases} 1 & \text{if } R(T) \leq T \land R(T) \geq \mu \tau_0 \\ 0 \end{cases} \] (5)

\[ P_{Adaptive\_CB} = \begin{cases} 1 & \text{if } R(T) \leq T \land R(T) \geq \mu \tau_0 \\ 0 \end{cases} \] (6)

\[ C_{Adaptive\_CB} = \begin{cases} 1 & \text{if } R(T) \leq T \land R(T) \geq \mu \tau_0 \\ 0 \end{cases} \] (7)

\[ C_{Adaptive\_CB} = \begin{cases} 1 & \text{if } R(T) \leq T \land R(T) \geq \mu \tau_0 \\ 0 \end{cases} \] (8)

Our goal is to minimize $C_{Adaptive\_CB}$ with both cache size $K$ the Push threshold $T$.

D. Extreme cases: Push and Invalidation

In this subsection, we consider two extreme/special cases. Let $\beta_{Inv}(t)$ and $\beta_{Push}(t)$ denote $\beta(t)$ values for the Invalidation scheme and the Push scheme, respectively. Let
\( P_{inv} \) and \( P_{Push} \) denote effective hit ratio values for the Invalidation scheme and the Push scheme, respectively. Let \( C_{inv} \) and \( C_{Push} \) denote costs values per data object access for the Invalidation scheme and the Push scheme, respectively.

\[
\beta_{Push}(t) = \left( 1 - e^{-\mu t} \right) \tag{10}
\]

\[
P_{Push} = K/N \tag{11}
\]

\[
C_{Push} = \left( 1 - \frac{K}{N} \right) \left[ L_{Req} + E(L) \right] + U \left[ E(L_{low}) + L_{ACK} \right] \tag{12}
\]

\[
\beta_{Inv}(t) = \frac{\mu}{\mu + \lambda} \left( 1 - e^{-(\mu + \lambda) t} \right) \tag{13}
\]

\[
P_{inv} = \sum_{k=0}^{K-1} \sum_{n=0}^{N} \binom{N}{n} U^{n}(1+U)^{N-n} \tag{14}
\]

\[
C_{inv} = \left[ 1 - \frac{1}{N} \sum_{k=0}^{K-1} \sum_{n=0}^{N} \binom{N}{n} U^{n}(1+U)^{N-n} \right] \left[ L_{Req} + E(L) \right] + U \left[ L_{inv} + L_{ACK} \right] \tag{15}
\]

E. Fetch and U-threshold

For the Fetch scheme, the cost function is very simple, and is given as follows.

\[
C_{Fetch} = L_{Req} + E(L) \tag{16}
\]

U-threshold is defined as a U value when the Fetch scheme is better than the adaptive CB scheme in (9). Therefore, when \( U \) is larger than U-threshold, the Fetch scheme is adopted, i.e., the cache is disabled or the cache size is zero. To obtain U-threshold, we can numerically solve the equation:

\[
C_{Fetch} = C_{Adaptive-CB} \cdot
\]

IV. OPTIMALITY ANALYSIS

A. Optimal Push threshold

Assume that data object sizes follow a uniform distribution as follows.

\[
g(x) = \begin{cases} \frac{1}{2a}, & x \in [E(L) - a, E(L) + a] \\ 0, & \text{otherwise} \end{cases} \tag{17}
\]

\[
C_{Adaptive-CB} = \left\{ 1 - A(T) + B(T) \frac{E(L) + a - T}{2a} \right\} \left[ L_{Req} + E(L) \right] + \left[ \frac{L_{inv} E(L) + a - T}{2a} + T + E(L) - a \right] + \frac{T - E(L) + a}{2a} + L_{ACK} \tag{18}
\]

If we let \( \frac{\partial C_{Adaptive-CB}}{\partial T} = 0 \), we can obtain the optimal threshold \( T_{Optimal} \) explicitly. Other distributions such as constant and exponential distributions are included in the journal version [7] of this paper.

B. Optimal cache size

Since the cache size \( K \) has a finite range \([0, M]\). For a given Push threshold and a given UAR, we can obtain the optimal \( K \) value by comparing \( M+1 \) values of the cache size. One extreme case is that when the UAR is very large, the Fetch scheme should be used since objects are always obsolete. Note that the Fetch scheme can be treated as a special case when the cache size is zero. Another extreme case is that when the UAR is zero, the cache size should be \( M \). In other words, the optimal \( K \) value is a function \( F(T(U)) \) of the Push threshold and UAR as follows.

\[
K_{Optimal}(T,U) = \begin{cases} 0, & U \geq U_{Threshold} \\ M, & \text{otherwise} \end{cases} \tag{19}
\]

The function \( F \) can be achieved by comparing the finite number of cases.

C. Optimizing both the cache size and the Push threshold

The major goal of this paper is to show that we can optimize \( T \) and \( K \) at the same time. We adopt the following steps. Step1: For each fixed \( K \) value within \([0, M]\), obtain the optimal cost \( C_{Optimal}(K) \) with the optimal Push threshold \( T_{Optimal}(K) \). Step2: Compare the above \( M+1 \) values to find the minimum cost \( C_{Optimal} \) and corresponding \( T_{Optimal} \) and \( K_{Optimal} \).

\[
C_{Optimal} = \min_{0 \leq K \leq M} \{ C_{Optimal}(K) \} \tag{20}
\]

\[
T_{Optimal} = T_{Optimal}(k_{i}), \text{ where } C_{Optimal}(k_{i}) = \min_{0 \leq K \leq M} \{ C_{Optimal}(K) \} \tag{21}
\]

\[
K_{Optimal} = k_{i}, \text{ where } C_{Optimal}(k_{i}) = \min_{0 \leq K \leq M} \{ C_{Optimal}(K) \} \tag{22}
\]

V. PERFORMANCE EVALUATION

We have the following parameters unless otherwise stated: \( N=100, M=70, K=70, \mu=1 \), and \( L_{inv} = L_{Req} = L_{ACK} = 30 \).

A. Simulation validations

Fig. 2 shows costs of simulations and analytic results where \( L_{inv} = L_{Req} = L_{ACK} = E(L)/2 = 60 \) and \( K=50 \). The simulation adopts discrete event simulation using C++. Fig. 2 indicates that analytical results match simulation results exactly for both the Push scheme and the Invalidation scheme.

B. Optimal Push threshold

We study a constant distribution, a uniform distribution and an exponential distribution for object sizes in this subsection. Fig. 3a, 3b, and 3c show the costs under a constant
distribution, a uniform distribution, and an exponential distribution, respectively, where \(E(L)=30\). Fig. 3a shows the cost over the object size, and Fig. 3b and 3c show the cost over the Push threshold. Fig. 3a, 3b, and 3c all indicate that when \(U\) is larger, the cost is higher since there are more updates. Fig. 3a shows that when the object size is small, the Push scheme is better and when the object size is large, the Invalidation scheme is better. Fig. 3b and 3c show that the cost decreases as the threshold \(T\) increases when \(T\) is small, and increases as \(T\) increases when \(T\) is large. Therefore, an optimal \(T\) value exists for both Fig. 3b and Fig. 3c. Fig. 3b and 3c also shows that the shapes of curves are similar under uniform and exponential distributions of object sizes.

In Fig. 3a, a larger \(U\) value makes the cost caused by data object updates weight more than the cost caused by data object accesses, and therefore, causes either the Push scheme or the Invalidation scheme better than another based on the object size. With some simple calculations, \((C_{\text{push}}-C_{\text{inv}})/C_{\text{inv}}\) could be as high as (±) 30-40%. Therefore, a better design is necessary.

Fig. 4 shows \(T_{\text{optimal}}\) values over \(U\) under a constant distribution, a uniform distribution, and an exponential distribution, respectively, where \(T_{\text{optimal}}\) is the optimal Push threshold and \(E(L)=30\). Fig. 4a shows that when \(U\) is not very large (<1.5), \(T_{\text{optimal}}\) increase as \(U\) increases; and when \(U\) is large, \(T_{\text{optimal}}\) decreases slowly as \(U\) increases since a data object in cache is more likely to be obsolete very often and the replaced objects have less usage. For the uniform distribution and the exponential distribution, the curves (Fig. 4b and Fig. 4c) are similar to Fig. 4a except that \(T_{\text{optimal}}\) changes slower with \(U\) value.

Fig. 5 shows costs over the object size under a constant distribution, a uniform distribution, and an exponential distribution, respectively, where \(U=1\). For all distributions, the cost increases as the object size increases. Fig. 5a compares the costs of the Push scheme, the Invalidation scheme, and the proposed optimal adaptive CB scheme; when the object size is small, the Push scheme is better than the Invalidation scheme, and when object size is large, the Invalidation scheme is better than the Push scheme; and the optimal adaptive scheme is the best scheme. Fig. 5b and Fig. 5c compare costs with different Push thresholds and the optimal Push threshold, and indicates that the optimal threshold achieves the minimum cost.

C. Optimize both cache size and Push threshold

This section is omitted due to the limited space. Please see the journal version [7] of this paper.

D. \(U\)-threshold

This section is omitted due to the limited space. Please see the journal version [7] of this paper.

VI. CONCLUSIONS

In this paper, we propose an optimal callback with two-level adaptation scheme for wireless data access. We analytically model a cost function as the total traffic involved between a server and an MT per data object access, and the optimal \(T\) value and the optimal cache size are obtained to minimize the cost function. An important threshold called \(U\)-threshold, is defined as an update-to-access-ratio (UAR) value, beyond which, the object should not be cached. We have results as follows.

- Analytical results match simulation results exactly.
- The cost is minimized with an optimal cache size and an optimal Push threshold.
- \(U\)-threshold increases as the UAR increases, and \(U\)-threshold is an important threshold and concept.
- As object size increases, costs of all schemes increase.
- The optimal \(T\) (with an optimal increases) increases first and then decreases as UAR increases.
- For different distributions of object sizes, effects of the proposed algorithm are almost similar.

REFERENCES
