TEMPORAL ASPECTS-BASED REPLACEMENT IN MEDIA OBJECT CACHES

Hagen Höpfner, Andreas Thenn and Maximilian Schirmer
Mobile Media Group, Bauhaus-Universität Weimar, Bauhausstr. 11, 99423 Weimar, Germany

Keywords: Caching, Cache Replacement, Temporal Aspects.

Abstract: Caching is an appropriate and well-known approach for reducing data transmissions in distributed information systems by creating and maintaining redundant data. As cache memory is limited and cached data might get outdated, it is impossible to store everything forever. If a cache is full, a replacement strategy decides on the cache entry that needs to be replaced by new data. There exist various strategies utilising different indicators for making this decision. Almost all of them do not take the content and the context of the systems' users into account. In this paper, we present two novel cache replacement strategies called TA and aTA that utilize temporal aspects included in media objects such as websites, specified by the user, or learnt from her or his behaviour. Our evaluation results show that, in the used application scenario, TA and aTA outperform classical replacement schemes like LRU.

1 INTRODUCTION AND MOTIVATION

Data transmissions are the most cost-intensive and time-consuming subtasks in most distributed information systems. Since wireless networks get more and more important, energy requirements also justify the need for reducing the number of unnecessarily repeated transmissions. Besides replication and hoarding, caching is an appropriate and widely-used approach for this purpose (Höpfner et al., 2009). The basic idea is to implicitly store received data as close as possible to the data processing subsystem. Well-known examples are caches in Web-based systems (Wang, 1999). Web browsers, e.g., cache Web objects such as images, videos, or HTML documents in the file system of the device running the browser. The browser reuses this locally available data if the user requests the same Web object again. Hence, a repeated transmission is not necessary. Certain research questions result from this simple idea:

(1) Which data should be or needs to be cached?

(2) What happens if cached data gets outdated?

(3) How to reuse cached data?

(4) What happens if the cache is full?

Although all questions are important in this paper, we only address the fourth one. For focussing purposes, we assume as "answers" to the other three questions that (1) all received objects are cached, (2) there are no updates on cached data, and (3) cache entries are identified by an identifier and reused in an atomic manner. Question four results from the fact that cache memory is limited. In case of a full cache, the system has to decide on removing or replacing, respectively, cached data. This decision is made by a cache replacement strategy. Various approaches have been researched in the past. In general, they differ in the parameters they factor into the decision. FIFO (first-in-first-out) (Tanenbaum, 2007), e.g., replaces the oldest cache entry, while LRU (Bennett and Kruskal, 1975) replaces the least recently used one. Hence, time or recency of accesses are taken into account. Almost all strategies published until today ignore the contents of cached data. This, of course, is necessary if a strategy aims for being as general as possible. However, in user-centered information systems, generic approaches might thwart the users' expectations, as the subjective importance of a cached object for the user might change over time. For example, information about a meeting taking place tomorrow is very important before and right after the meeting. Therefore, it should stay in cache. If one uses the system the day after the meeting, the subjective importance decreased and the data might be replaced.

In this paper, we analyse various temporal aspects of a user-based access to media objects. This includes

Höpfner H., Thenn A. and Schirmer M..
TEMPORAL ASPECTS-BASED REPLACEMENT IN MEDIA OBJECT CACHES.
DOI: 10.5220/0003799700730082
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 73-82
ISBN: 978-989-8565-08-2

temporal aspects within media objects, as well as user preferences (e.g. prefer information for weekends) and user behaviour. Based on these temporal aspects, we developed two novel cache replacement strategies, called TA (temporal aspects-based replacement) and aTA (adaptive TA). Both were implemented and successfully evaluated.

The remainder of this paper is structured as follows: Section 2 presents an application scenario that motivates the research. Section 3 discusses the related work. Section 4 introduces and classifies the temporal aspect considered in our work. Section 5 describes the novel replacement strategies. Section 6 presents the evaluation scenario and the evaluation results. Section 7 summarizes the paper with a conclusion and outlines future research directions.

2 APPLICATION SCENARIO

Today is Monday, August 29, 2011. Anna and Irene utilise smartphones for organising their spare time activities. Both installed the WeIS app, which accesses the WeIS Web service. WeIS is the so called "Weimar Information System". It collects and integrates information about events (e.g., cinema schedules, concert information) from various Web sources. The WeIS app caches received data on the smartphone. As Anna and Irene have different interests, they browse the WeIS content in a different manner. Anna prefers to go to the cinema on weekends and therefore frequently reads movie descriptions and watches trailers of movies that will be shown on the next weekend. She did this a week ago, too. Hence, Annas cache contains movie information and trailers that were relevant for August 27 and August 28 as well as those that are relevant for September 3 and September 4. Obviously, upcoming events are more important than passed ones. She also had a short look at the cinema schedule for next Wednesday. As Anna specified a "weekend data caching", the Wednesday information is not cached. However, in case of a full cache, if Anna reads the cinema schedule for the week after this week, the information about last week's movies is replaced rather than this week's information. If she uses the app again on September 5, the August information became less important, and so on. Irene is more keen on going to concerts, no matter if they take place on weekends or not. Hence, all data (band biographies, audio files, etc.) accessed by Irene is cached on her smartphone. Furthermore, she frequently reads the newsletters of her favourite artists using WeIS. If the cache is full, the importance of the cache entries depends on the event date in relation to the current date. Outdated events are less important than recent ones, and recent events are more important than future ones. Hence, Irene's app replaces the less important concert information. As shown by both examples, the importance of cached data depends on the temporal aspects of the cached information as well as on the current date. Furthermore, importance changes over time.

3 RELATED WORK

As already briefly mentioned in Section 1, our research falls into the area of distributed information systems (dIS) in general. More precisely, our work is related to redundant data management approaches in dIS. According to the classification presented in (Höpfner et al., 2009), there exist three categories: (1) replication, (2) hoarding, and (3) caching. They differ in their approach to specifying the chosen data, their possibility of using redundant data without having a connection to the master copy, their required software components, the potential dynamics of the data, and their handling of updates to the redundant data. For details on (1) and (2), we refer to (Höpfner et al., 2009). The basic idea of caching is to reuse implicitly received data. If a user requests data for the first time, the client keeps processed data locally. This copy is used later on for answering new requests on the same information. There are various cache related issues to be considered. In this paper, we introduce new replacement strategies that are necessary to shrink a cache if cache memory is full. The literature introduced many different approaches to decide about the cached data items that are replaced by new incoming data (Wang, 1999; Podlipnig and Böszörmenyi, 2003; Romano and ElAarag, 2008). Almost all of them try to replace data that will most likely be useless in the future, taking one of the following indicators into account: time point t_i of the last access to data item i, the time T_i since the last access, and the access frequency f_i. Other indicators are the size s_i of a data item, the costs c_i of retransmitting it, or the access latency l_i.

There are only a few approaches that use the semantics of the cached data or the context of the system's usage. The LSR strategy (Calsavara, 2003) replaces the least semantically related object but does not take temporal aspects into account. Even more enhanced caching strategies like location-aware semantic caches (Ren and Dunham, 2000) and their replacement strategies (i.e., FAR) do not analyse cached data in this regard. In contrast to this, the cache replacement strategies introduced in this paper take the temporal relevancy of and within data items into account.

4 TEMPORAL ASPECTS

The term "media object" used in our work extends the definition of "Web objects" given in (Podlipnig and Böszörmenyi, 2003). The term "Web object" subsumes "*all possible objects (HTML pages, images, videos, etc.) that can be stored in a proxy cache.*" Media objects are Web objects plus other remotely accessible documents like emails, calendar items, etc. Temporal aspects of a media object result from explicit meta information specifications (e.g., well-defined date information of an event) or from the contents of the object (e.g., a date mentioned in the text of an HTML page). In the following we, refer to explicit temporal aspects as *external*. There are various approaches for harvesting temporal aspects from the contents of a media object (Lienhart, 1999; Saquete and Martínez-Barco, 2000; Morita et al., 2000; Martínez-Barco et al., 2002). They are also built into common desktop applications. Apples Mail.app, e.g., analyses emails and allows to create a calendar entry if a date is found in the text of the email. However, these information retrieval approaches are out of the scope of this paper. We assume, that there is a proper way to harvest such *internal* temporal aspects.

As the importance of a media object is a subjective measure, we have to take the user into account. He or she might use an *explicit* definition of the importance, e.g., a certain user might be more interested in events taking place on weekends. In addition, one can analyse the user's access behaviour to cached media objects. If he or she reuses media objects that have a temporal relevancy within a week, the system should keep such objects in cache. We call such temporal aspects *implicit*. Please note, that implicit temporal aspects are *not* comparable to the access statistic used, e.g., for LRU. It changes over time an does not refer to a single media object, while standard LRU analyses accesses per object.

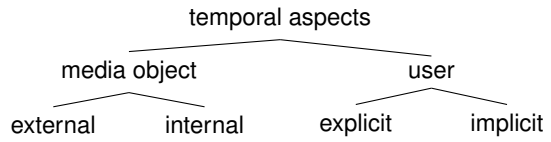

Figure 1: Taxonomy of caching relevant temporal aspects.

Figure 1 illustrates a taxonomy of the considered temporal aspects. The application scenario (cf. Section 2) contains all of them:

External: The temporal aspect of a movie trailer is not explicitly given by the movie file itself. However, it results from the cinema schedule that is easily wrapped from the well-structured website of the cinema.

Internal: Artists might inform their fans on upcoming concerts in their newsletters. As newsletters are mostly unstructured, this information needs to be retrieved from the plain text.

Explicit: Anne configured her WeIS app in a way that it caches only those data relevant for weekends.

Implicit: Irenes WeIS app analyses her access behaviour and "recognises" that she most frequently accesses information that becomes relevant one month after the access date.

For the rest of the paper, we assume that each cached media object has a defined validity date.

5 REPLACEMENT STRATEGIES

The temporal aspect-based replacement strategies presented in this paper are, to a certain degree, subjective. We assume that outdated media objects (validity date is a thing of the past) are less important then current media objects and those with a validity date in the near future. Furthermore, we assume that media objects becoming obsolete later in future are less important than those that become obsolete in near future. However, all of them are more important than already outdated media objects.

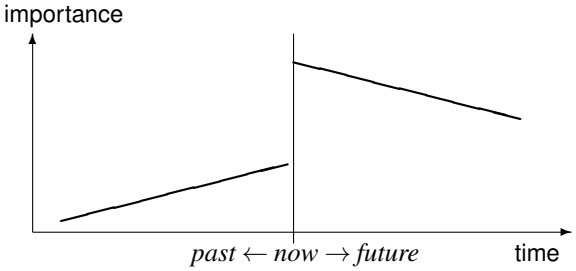

Figure 2: Validity date based importance of media objects.

5.1 TA Replacement

As illustrated in Figure 2, the importance of a cached media object depends on the current point in time and therefore changes over time. Let $\mathcal{M}$ be the set of all media objects in the cache. The expiration date of an object $m \in \mathcal{M}$ is given as $exp(m)$ and the importance of m is given as $imp(m)$. At a certain point in time (now) the set $M^P \subseteq \mathcal{M}$ contains all outdated media objects ($\forall m \in M^P | exp(m) < now$) and the set $M^F \subseteq \mathcal{M}$ contains all not outdated media objects ($\forall m \in M^F | exp(m) \geq now$). Consequently,

$M^P \cap M^F = \emptyset$ and $M^P \cup M^F = \mathcal{M}$ must hold. Hence, a correct importance-based order of the cached media objects is given if

- $\forall m,n \in M^P((imp(m) < imp(n)) \leftrightarrow (exp(m) < exp(n))) \vee |M^P| < 1$,
- $\forall m,n \in M^F((imp(m) \leq imp(n)) \leftrightarrow (exp(m) > exp(n))) \vee |M^F| < 1$, and
- $\forall m \in M^P \forall n \in M^F (imp(m) < imp(n))$

hold. We are not interested in absolute importance values, but rather in sorting the replacement queue of the media objects according to the relative importance among cached objects. Similar to the well-known LRU queue, the least important media object represents the head of the queue and would be removed if a replacement is required. The queue is ordered according to the importance defined above. The closer a media object is placed at the tail of the queue, the more important it is. However, in contrast to the LRU queue, our TA queue also changes over time, even if no objects are referenced in between. Hence, materialising the TA queue would be inefficient as time passes by and reordering would become a frequent task even if no new media objects appear. Therefore, we decided to realize the TA queue as a virtual queue that is evaluated only if a replacement is required (cf. Figure 3[1]).

	object reference	expiration date	imp.
M^P	C	6.2.2011	1
M^F	A	10.2.2011	2
	D	8.2.2011	3
	B	8.2.2011	4

Figure 3: Virtual TA queue on February 7th, 2011.

Media objects are simply stored in the file system. The virtual TA queue is a table that maintains object references (absolute file names) and expiration dates. For illustration purposes, we added an importance value in Figure 3. At this, object C is least important ($imp(C) = 1$), as it is already outdated. If the cache would have to store a new media object on February 7th, 2011, C would be replaced. If more space would be required, TA would replace A that is not outdated, but refers to the most future point in time (cf. Figure 2). Figure 4 illustrates the virtual TA queue two days later. C is still outdated. However, now B and D are outdated, too, and/or would be replaced rather than A.

Algorithm 1 implements the TA cache replacement strategy. It is triggered by new incoming media

[1]For illustration purposes, we use dates as temporal values throughout the paper. However, the implementation and the evaluation also take time values into account.

	object reference	expiration date	imp.
M^P	C	6.2.2011	1
	D	8.2.2011	2
	B	8.2.2011	3
M^F	A	10.2.2011	4

Figure 4: Virtual TA queue on February 9th, 2011.

objects in case of a full cache. As shown in Figure 3 and Figure 4, the virtual TA queue is a table. Formally it is the set $TAQ = \{(ref(m), exp(m)) | m \in \mathcal{M}\}$ of tuples of the form $(ref(m), exp(m))$. At this, $ref(m)$ is a reference to the media object $m \in \mathcal{M}$ and $exp(m)$ is the aforementioned expiration date. Furthermore, $size(m)$ returns the memory space required for caching media object m (i.e., m's file size).

Algorithm 1: TA cache replacement.

Precond.: // new object would fit into cache
Input: TAQ // virtual TA queue
n // new media object
now // current date/time
rm // required memory
Output: R // set of discarded objects' references

```
01  def TA_replacement(TAQ, n, now, rm):
02      R = ∅ // reset replacement set
03      fs = rm − FCM // initiate loop variable
04      nmo = FALSE // flag to avoid useless scans
05      while fs > 0 do:
06          date = now; cand = NULL
07          if nmo == FALSE
08              for each (x,y) ∈ TAQ do:
09                  if y < date ∧ x ∉ R then:
10                      date = y; cand = x fi
11              done
12          fi
13          if cand ≠ NULL then:
14                  R = R ∪ {cand}
15              fs = fs − size(cand)
16          else // no more outdated objects
17              nmo = TRUE
18              for each (x,y) ∈ TAQ do:
19                  if y ≥ date ∧ x ∉ R then:
20                      date = y; cand = x fi
21              done
22              R = R ∪ {cand}
23              fs = fs − size(cand)
24          fi
25      done
26      return(R)
```

Algorithm 1 returns the set $R = \{ref(m) | m \in \mathcal{M}\}$ of references to those media objects that shall be

replaced. In a subsequent step, the respective media objects are removed from cache and queue, and the new object is cached and added to the queue. The precondition is checked by comparing $size(n)$ to the guaranteed (predefined) cache memory space. If a media object is bigger than the cache, the object is not cached at all. The basic principle of the algorithm is as follows: We first calculate the memory required for caching the new object n by subtracting the amount of free cache memory FCM from the amount of required memory rm (Line 3) that is determined using $size(n)$ in the function call. Then we loop through the TA queue and first try to collect enough memory by marking outdated objects for replacement (Lines 07-15). If no more outdated objects can be found and if we need more free space, we start selecting those objects that expire in the future (Lines 16-24). The algorithm is guaranteed to terminate and does not require file access to the cached objects.

Example and Threats to Validity. The following example (cf. Figure 5) extends the Figure 3 by showing the file sizes. We assume, that cache memory is limited to 20MB.

object reference	expiration date	size
C	6.2.2011	4MB
A	10.2.2011	8MB
D	8.2.2011	3MB
B	8.2.2011	1MB

Figure 5: Example TA replacement on February 7th, 2011.

Anna accesses a 10MB video file (E) that expires on March 1st, 2011. Obviously, 10MB is less than 20MB. Hence, the precondition is fulfilled. The cache uses $4+8+3+1 = 16$MB. 4MB are free that are not sufficient for caching the new file. We have to free $10-4 = 6$MB. The TA algorithm first selects outdated objects for replacement. On February 7th, 2011 only C is outdated. Removing C would free 4MB, which is not sufficient either (4MB from C plus 4MB free space sum up to 8MB while 10MB are required). Hence, TA replacement now selects A for replacement and, as enough memory was collected, returns $R = \{C,A\}$. Both objects are removed from cache and queue and E is added. Figure 6 illustrates the resulting virtual TA queue.

object reference	expiration date	size
D	8.2.2011	3MB
B	8.2.2011	1MB
E	1.3.2011	10MB

Figure 6: Example TA replacement result on February 7th, 2011.

If Irene would have accessed E two days later (cf. Figure 4) then either C, B, and D or C and D would have been removed (depending on the order "within" TAQ). This issue results from the fact that our naive algorithm does not intelligently differentiate between cached objects with the same expiration date. In order to overcome this issue, we experimented with two different approaches (cf. Section 6). As basic behaviour, we extended Algorithm 1 in a way that all "candidates" are removed. For this example this means that C, B, and D are always selected. The second variant utilises the idea of LRU. If two objects have the same expiration date, then we first select the one that was least recently used. The LRU counter was simply added to the TA queue.

Another issue illustrated by the example is the fact that the new object expires later than the at latest expiring cached object. Actually, this conflicts with our definition of importance. However, as Irene intentionally accessed this new object and as time-based importance is strongly subjective, anyway, we decided to insert objects independent of their expiration date into the cache.

5.2 aTA Replacement

The TA replacement considers only external and internal temporal aspects. Section 4 also introduced explicit and implicit temporal aspects that result from users preferences and behaviour. Irene (cf. Section 2) might be interested in concerts that take place within a week from now earliest, while Anna might be more interested in this week's cinema information. Hence, TA cache replacement would reflect Anna's behaviour but not Irene's. The aTA replacement strategy also considers such implicit temporal aspects. The basic idea is to analyse access time frames, so called "zones".

As shown in Figure 7, accesses to objects are grouped, based on their expiration date in relation to the current time t. In our experiments, we decided to use six zones, three historical zones and three future zones. However, the number of zones as well as their definition is subject to future research.

During the runtime of the system object accesses are analysed and counted. The zones in Figure 7 correspond to the following access behaviour:

Zone 1: access to objects expired earlier than one week before *now*.

Zone 2: access to objects expired one week but not earlier than 24 hours before *now*.

Zone 3: access to objects expired within 24 hours before *now*.

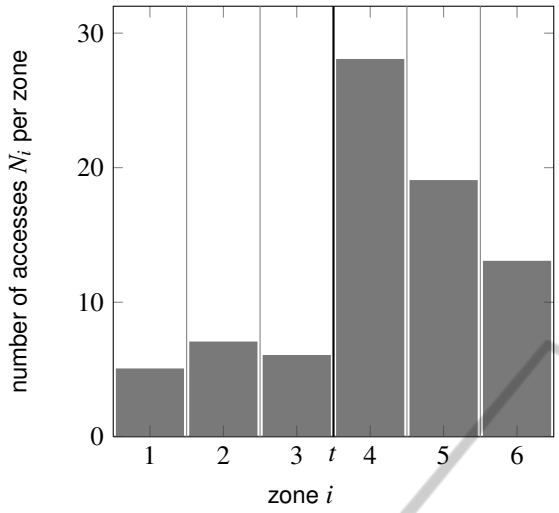

Figure 7: Example access distribution for 6 zones.

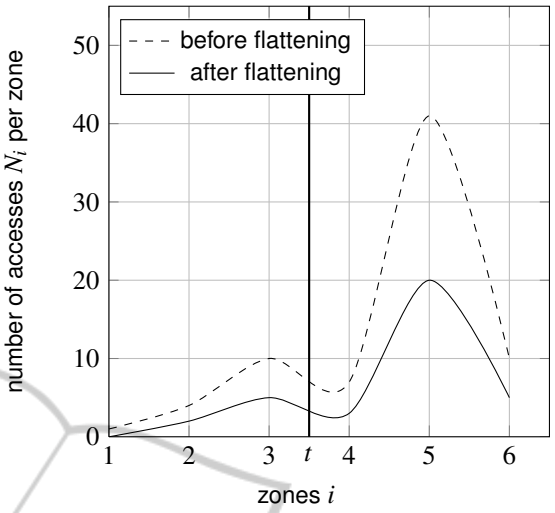

Figure 8: Smoothing of the aTA importance function.

Zone 4: access to objects that will expire within 24 hours from *now*.

Zone 5: access to objects that will expire in one week but not earlier than 24 hours from *now*.

Zone 6: access to objects that will expire later than one week from *now*.

Using this approach allows to define an importance function that reflects the user's behaviour rather than the static definition illustrated in Figure 2. However, objects of the same zone are ordered according to the TA importance function. The example data shown in Figure 7, that results from our experiments, illustrates that this particular user is more interested in media objects of zone 2 than in those of zone 3. However, simply counting accesses is not sufficient as the counters would grow with each access. In order to avoid this phlegm, we halve and round down all zone counters after 50 accesses each and therefore flatten the importance curve (cf. Figure 8).

Figure 8 illustrates a different user behaviour than Figure 7. Here, the user accesses data that expires in one week but not earlier than one day from now more often than those data expiring within the next 24 hours. Therefore, the importance function reflects Irene's interest in concerts taking place in a week from now. If the system has to cache new data and if the cache is full, objects from the least important zone are replaced according to Algorithm 2. Besides the virtual TA queue, we need a function lfz that calculates the least frequently used[2]zone and a function $map(n)$ that assigns a media object n to a particular zone. The zones are represented as the set $\mathcal{Z} = \{(i,c)\}$ of tuples of the form (i,c) with $i,c \in \mathbb{N}$. At this, i is the zone number and c is the (flattened) number of accesses to this zone. In our implementation[3], the mapping is done as follows. $24h$ means one day (24 hours), $168h = 24h \cdot 7$ means one week.

$$map(n) = \begin{cases} 1 & \text{if} \quad exp(n) < now - 168h \\ 2 & \text{if} \quad now - 168h \leq exp(n) \ldots \\ & \quad \cdots < now - 24h \\ 3 & \text{if} \quad now - 24h \leq exp(n) < now \\ 4 & \text{if} \quad now \leq exp(n) \leq now + 24h \\ 5 & \text{if} \quad now + 24h < exp(n) \ldots \\ & \quad \cdots \leq now + 168h \\ 6 & \text{if} \quad exp(n) > now + 168h \end{cases}$$

The $lfz(\mathcal{Z})$ functions is a lookup function that returns the value of i of the tuple $(i,c) \in \mathcal{Z}$ where $\forall (i',c') \in \mathcal{Z} : c \leq c'$ holds.

zone (i)	number of accesses (c)	
1	0	(1)
2	2	(4)
3	5	(10)
4	3	(7)
5	20	(41)
6	5	(10)

Figure 9: Example: zone based access frequency.

Figure 9 corresponds to Figure 8 and presents an example for the zone set. The rightmost column,

[2]Please note that this approach is not comparable to the LFU replacement strategy (Tanenbaum, 2007). LFU considers the access frequency of objects rather than the grouping of temporal accesses that are independent of the accessed objects.

[3]Please note that other zone definitions might be possible. As Algorithm 2 only uses this function, the algorithm can be adapted by providing another mapping function.

which has been included into the figure for illustration purposes only, contains the access values for the unflattened curve. As one can see, the zone importance does not change if flattening is used.

Algorithm 2: aTA cache replacement.

Precond.: // new object would fit into cache
Input: *TAQ* // virtual TA queue
n // new media object
now // current date/time
rm // required memory
Output: *R* // set of discarded objects' references

```
01  def aTA_replacement(TAQ, n, now, rm):
02      R = ∅ // reset replacement set
03      fs = rm − FCM // initiate loop variable
04      CZ = ∅ // remember cleaned zones
05      while fs > 0 do:
06          tTAQ = ∅ // reset tTAQ
07          cm = 0 // cleanable memory per zone
08          zone = lfz(Z − CZ)
09          for each (x,y) ∈ TAQ do:
10              if map(x) == zone then
11                  cm = cm + size(x)
12                  tTAQ = tTAQ ∪ {x,y}
13              fi
14          done
15          if tTAQ ≠ ∅ then
16              R' =TA_repl.(tTAQ, n, now, cm)
17              R = R ∪ R'
18              fs = fs − cm
19          fi
20          CZ = CZ ∪ {zone}
21      done
22      return(R)
```

Algorithm 2 works in two stages. In stage 1, a temporary virtual TA queue *tTAQ* is generated (Lines 6-14). It represents those objects that belong to the least recently used zone. Furthermore, stage 1 calculates the amount of memory that can be freed by replacing all objects from the selected zone. This value is stored in *cm*. Stage 2 utilises *cm* and calles Algorithm 1 based on *tTAQ* (Line 16). We iterate through both stages until enough memory can be freed. Stage 2 is only entered if the selected zone contains objects (Line 15).

Example. Given the example discussed in the previous subsection, the virtual TA queue illustrated in Figure 5, as well as the zone definition from Figure 9. On February, 7th 2011 object *C* belongs to zone 3, object *A* belongs to zone 5, and the objects *D* and *B* belong to zone 4. We need 10MB for caching object *E*. Again, we have 4MB free cache memory. Hence, 6MB must be freed. The aTA algorithm first determines the least frequently used zone, which is zone 1 (cf. Figure 9, 0 accesses). As no object belongs to zone 1, zone 2 (2 accesses) is analysed. This zone does not contain objects either. Hence, zone 4 (3 accesses) is used next. Zone 4 contains *B* (1MB) and *D* (3MB). As we have to free 6MB, both objects are selected using the TA algorithm. The next least important zone is zone 3 or zone 6. It does not matter, which one is selected, because both were used equally often. As zone 6 does not contain any objects, zone 3 is used anyway. Hence, *A* is selected for replacement, too.

Under the assumption that the access frequency history represents Anna's behaviour, *B*, *C*, and *D* would be replaced. The resulting virtual TA queue after this replacement is illustrated in Figure 10.

object reference	expiration date	size
A	10.2.2011	8MB
E	1.3.2011	10MB

Figure 10: Example aTA replacement result on February 7th, 2011.

5.3 Cache Filters

Explicit temporal aspects are indicators for a manual individualisation of the caching behaviour, independent of the used replacement strategy. Remember the application scenarios presented in Section 2. Anna is more interested in events taking place on weekends. We utilise this information and cache only those media objects that are relevant for weekend events (cf. Figure 11).

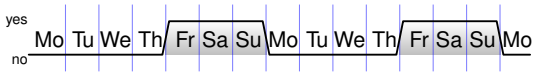

Figure 11: Example for manually defining cache contents.

6 EVALUATION

The performance of cache replacement strategies is measured in form of the ratio of the number of cache hits $N_{CH} \in \mathbb{N}$ to the number of requests $N_Q \in \mathbb{N}$. A cache hit occurs if a request can be answered using cached objects. Otherwise a cache miss indicates that the requested object is not in the cache. The following formula is used in order to calculate the cache hit rate $H_{TA} \in \mathbb{R}$ for the TA replacement strategy:

$$H_{TA} = \frac{N_{CH}}{N_Q}$$

Obviously, the value must be in the interval $[0,1]$, while 1 indicates best performance (all requests are answered by the cache) and 0 indicates worst performance (no requests are answered by the cache). We evaluated our strategies in comparison to the LRU strategy. The respective LRU cache hit rate is denoted as H_{LRU}. According to this naming convention, H_{aTA} denotes the cache hit rate for the aTA-algorithm. H_{TA+LRU} names the cache hit ratio for the combination of TA and LRU.

6.1 Prototype

As mentioned in Section 1, the importance of cached data is of a subjective nature. Therefore, evaluation is subject to a user study. We implemented the mobile information system introduced in Section 2 using the WebOS plattform (Allen, 2009) (cf. Figure 12). The WeIS-client requests data from a server running CouchDB (Anderson et al., 2010). We manually filled the server database and cleaned up the respective expiration dates. The prototype also covers cache coherency mechanisms that are out of the scope of the paper but necessary for future applications.

Figure 12: Screenshot of the WeIS evaluation prototype.

6.2 Experimental Setup

We implemented four different caches that ran in parallel and utilised one of the following replacement strategies:

LRU: A standard least recently used replacement strategie is used as reference implementation as.

TA: The TA implementation is based on Algorithm 1 presented in Section 5.1. As an extension, we decided to replace all media objects with the same expiration date if more alternatives exist.

TA+LRU: The TA+LRU implementation is based on Algorithm 1 presented in Section 5.1, too. In the case that media objects have the same expiration date we aditionally used LRU in order to select the replacement candidate.

aTA: The aTA+LRU implementation is based on the Algorithm 2 presented in Section 5.2. In stage 2, the TA+LRU implementation is used.

We conducted a rather small user study with 5 users that used WeIS for 14 days. During this time and due to privacy issues, we asked the users to actively submit at least 10 generated reports through the app. In order to guarantee that replacement is used, we limited cache memory to 512 KBytes. Furthermore, in a preprocessing phase, we guaranteed comparable object sizes among provided media objects. This was necessary as larger objects would cleanup the "small" cache completely. However, in productive environments, the capacity of modern smartphones supersedes this assumption as well as the cache memory limit.

6.3 Evaluation Results

The numbers shown in Figure 13 illustrate that our test persons used the app differently. Especially, there was no improvement over a default LRU replacement for $user_4$ and $user_5$. However, Figure 14 shows that TA, TA+LRU, as well as aTA performed just as bad as or even better than LRU. The relatively low cache hit rates result from the fact, that users were free to use the system and mostly accessed uncached data. However, the trend shows, that our temporal approach reflects user behaviour better than the access statistic used in LRU.

Figure 15 supports this hypothesis, too. It shows the average cache hit rate in relation to the measured LRU result. Interestingly, TA is better than TA+LRU. Please remember that our TA implementation replaces all replacement candidates in case of equal expiration dates. Due to the nature of the used application scenario, this means that all objects

	$user_1$	$user_2$	$user_3$	$user_4$	$user_5$
LRU	59	44	8	23	11
TA	70	53	9	23	11
TA+LRU	70	47	9	23	11
aTA	56	54	9	23	11
Σ	266	215	56	94	69

Figure 13: Absolute number of cache hits.

	$user_1$	$user_2$	$user_3$	$user_4$	$user_5$
H_{LRU}	.222	.205	.143	.245	.160
H_{TA}	.263	.247	.161	.245	.160
H_{TA+LRU}	.263	.219	.161	.245	.160
H_{aTA}	.211	.251	.161	.245	.160

Figure 14: Cache hit rates.

(images, texts, videos) belonging to the same event are replaced at once. In other words, this means that more memory is freed in one replacement step. Hence, TA+LRU keeps rather unimportant data in cache at the point of replacement. As the importance value might change over time, these remaining objects might become more important than those that are more frequently used. If a replacement request selects a more frequently accessed but, from a temporal point of view, less important media object, cache hit ratio decreases with upcoming requests.

replacement strategy	cache hit rate in relation to LRU
LRU	100%
TA	110.36%
TA+LRU	107.49%
aTA	105.44%

Figure 15: Cache hit rate improvement in relation to LRU.

Furthermore, we analysed the number of cached objects for $user_2$. As illustrated in Figure 16, starting with report 5, less objects remain in cache with LRU. The reason for this results from the fact that LRU kept certain larger files in cache that were replaced right before this report. TA in contrast replaced those files in an earlier stage at the point the corresponding event information was replaced. However, a more detailed analysis of this behaviour is subject to future research and requires a more detailed and more frequent data acquisition with more users.

Figure 17 compares the TA implementation to the aTA implementation using the reports of $user_1$. Obviously, TA outperforms aTA. The reason for this is the fact that aTA requires some time to learn the user's behaviour. Starting with report 6, one can see a permanent increasing aTA cache hit rate. Hence, the learned

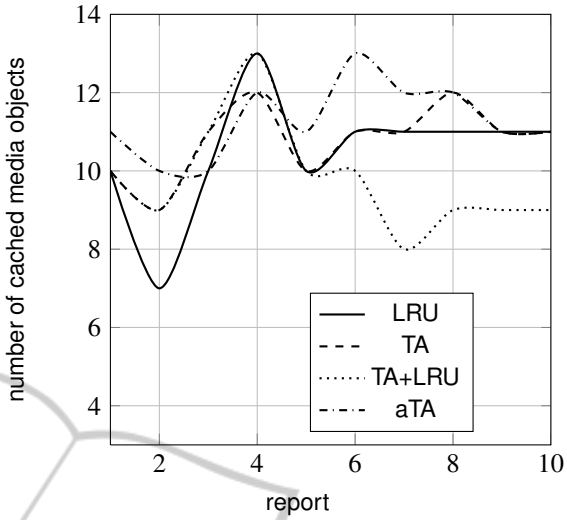

Figure 16: Cache space usage of $user_2$.

importance function becomes better. However, aTA must adapt the zones to the current user behaviour. If it changes, aTA cache hit rate would decrease dramatically as the wrong zone would always be selected for replacement (until the zones are corrected again).

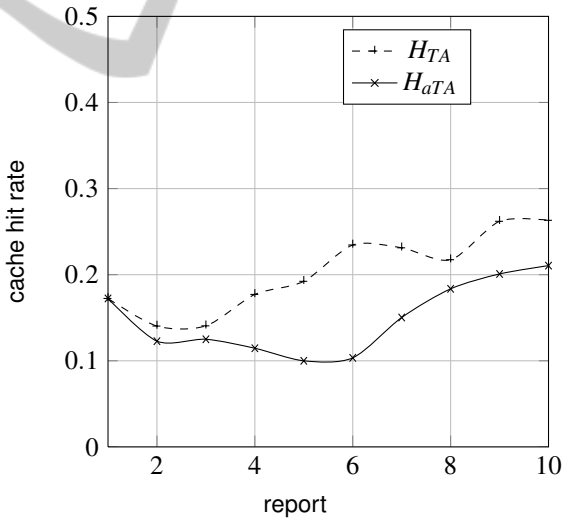

Figure 17: Development of the cache hit rates for $user_1$.

7 SUMMARY, CONCLUSIONS AND OUTLOOK

In this paper, we introduced two new cache replacement strategies, TA and aTA. Both utilise temporal aspects of accessing media objects in order to select the replacement candidate(s) in case of a full cache.

We first analysed and classified the temporal aspects. Then we used them in order to define importance functions describing the temporal aspect-based (subjective) importance of media objects. Explicit and implicit temporal aspects that result from the media objects themselves were used for the TA replacement. Implicit temporal aspects that result from user's behaviour were used for aTA. As the evaluation results show, both approaches outperform a standard LRU replacement. However, we are aware of the fact that there are plenty of other replacement strategies we should analyse in comparison to TA and aTA. However, before doing this, we will take a deeper look into the zone definition and the zone numbers of aTA. Furthermore, we plan to use dynamic zone definitions instead of fixed ones.

A more general research question results from the data preprocessing. For our experiments, we manually tagged the media objects. In order to adapt our approaches to other applications, one has to find proper ways to harvest time and date information from arbitrary media objects. Our prototype contains rudimentary support for Websites. However, not all dates mentioned on a Website have to be expiration dates. Another open issue is the support for durations of validity. We currently only support expiration dates and assume that validity starts at the first time an object is accessed before caching. A more detailed study on time/date ranges could improve temporal aspect based caching approaches.

Last but not least, we are planning a more detailed evaluation including some of the mentioned future works and involving more users for a longer period of time. Therefore, we are working on porting WeIS to Apple's iOS and Google's Android platform and also include the cache filtering. The latter was not evaluated during the experiments presented in this paper, but might improve the cache hit ratio dramatically if used in a proper way.

ACKNOWLEDGEMENTS

We want to thank all students of the Mobile Media Group of the Media Department at the Bauhaus-Universität Weimar, who participated in the conducted evaluation.

REFERENCES

Allen, M. (2009). *Palm webOS*. O'Reilly Media, 1st edition. http://shop.oreilly.com/product/9780596155261.do.

Anderson, J. C., Lehnardt, J., and Slater, N. (2010). *CouchDB: The Definitive Guide*. O'Reilly Media, 1st edition. http://guide.couchdb.org/editions/1/en/index.html.

Bennett, B. T. and Kruskal, V. J. (1975). LRU stack processing. *IBM Journal of Research and Development*, 19:353–357.

Calsavara, A. (2003). The least semantically related cache replacement algorithm. In *LANC'03 Proceedings of the 2003 IFIP/ACM Latin America conference on Towards a Latin American agenda for network research*, pages 21–34. ACM Press.

Höpfner, H., Mansour, E., and Nicklas, D. (2009). Review of Data Management Mechanisms on Mobile Devices. *it – information technology*, 51(2):79–84.

Lienhart, R. (1999). Abstracting home video automatically. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pages 37–40. ACM Press.

Martínez-Barco, P., Saquete, E., and Muñoz, R. (2002). A Grammar-Based System to Solve Temporal Expressions in Spanish Texts. In Ranchhod, E. and Mamede, N. J., editors, *Advances in Natural Language Processing — Proceedings of the Third International Conference, PorTAL 2002*, volume 2389 of *Lecture Notes in Computer Science*, pages 709–719, Berlin / Heidelberg. Springer.

Morita, M., Lethelier, E., Yacoubi, A. E., Bortolozzi, F., and Sabourin, R. (2000). An HMM-based Approach for Date Recognition. In *Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems (DAS 2000), December 10-13, 2000, Rio de Janeiro, Brazil*, pages 233–244.

Podlipnig, S. and Böszörmenyi, L. (2003). A Survey of Web Cache Replacement Strategies. *ACM Computing Surveys*, 35(4):374–398.

Ren, Q. and Dunham, M. H. (2000). Using semantic caching to manage location dependent data in mobile computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 210–221. ACM Press.

Romano, S. and ElAarag, H. (2008). A quantitative study of recency and frequency based web cache replacement strategies. In Ahmad, A. and Bragg, A., editors, *CNS '08: Proceedings of the 11th communications and networking simulation symposium*, pages 70–78. ACM Press.

Saquete, E. and Martínez-Barco, P. (2000). Grammar specification for the recognition of temporal expressions. In *MT 2000: Online-Proceedings of the International Conference on Machine Translation and Multulingual Applications in the new Millenium*. The British Computer Society. paper 21, available online: http://www.mt-archive.info/BCS-2000-Saquete.pdf.

Tanenbaum, A. S. (2007). *Modern Operating Systems*. Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition. http://www.pearsonhighered.com/product?ISBN=0136006639.

Wang, J. (1999). A survey of web caching schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, 29(5):36–46.