Towards Java-based Data Caching for Mobile Information System Clients

Alexandru Caracas, Iulia Ion, Mihaela Ion, Hagen Höpfner
{*alexandru.caracas*|*iulia.ion*|*mihaela.ion*}*@i-u.de, hoepfner@acm.org*
International University in Germany
School of Information Technology
Campus 3, 76646 Bruchsal, Germany

Abstract: Mobile information system require client side caching of data for various reasons. Caching issues are discussed in many scientific publications that mostly lack an usable implementation. Nearly all mobile devices support Java's mobile information device profile (MIDP). In this paper we present our implementation of an MIDP-implemented cache for mobile phones.

1 Introduction and Motivation

Information systems with mobile clients have to handle many difficulties that do not exist in non-mobile environments. One of the most important issues is the wireless connectivity. Wireless networks are either slow, expensive and widely available (e.g. GSM, HSCSD, GPRS) or fast, less expensive but not accessibly everywhere (e.g. WLAN, UMTS). However, even with a fast and cheep wireless "always-on"-connection client side data caching is necessary. On the one hand over air data transmissions are energy-intensive and reduce the operating time of the device. On the other hand physical effects like shielding, reflection, refraction, etc. make it is impossibles to guarantee a permanent availability. So, data caching is an essential issue in mobile information systems (MIS). There are many publications on caching, hoarding or replicating data in MIS. However, almost all of them lack an usable implementation and evaluations are mostly of a purely scientific nature based on system simulations. In this short paper we present a Java-based implementation of a client side cache that allows the storage and offline usage of data received by a mobile phone via web-services.

2 Related Work

Our paper touches various research areas. However, client site data storage in mobile information system is its major topic. Based on the classification of replication, hoarding and caching in [HTKR05] one can identify major caching issues in our work. We implicitly store data on the mobile client that is explicitly queried and reuse it.

In related papers caching is very often realized in a semantic manner [LLS99]. We do not partially reuse stored data by applying filter queries but assume that the data structure is understood by the application. So, semantic caching techniques are not necessary here. Furthermore, clients do not allow modifications of cached data. So, synchronization techniques as presented e.g. in [GHOS96] are not required.

There are some projects that aim to implement a system similar to our proof of concept presented in Section 4. The German Railway - *Deutsche Bahn* (DB) offers a static solution for train schedules[1]. Mobile users have the possibility to download a personal schedule between two stations by specifying the start and destination. However, there is no guarantee that the data is up to date, moreover the user is presented only with the trains to one particular destination.

Another approach is by *Actuan Mobile* who offers the complete dynamic schedule for the New Jersey Transit rail system. The application[2] is offered both as a mobile browser solution and J2ME application. The mobile application which is available for download is quite complex and offers extensive features. However, the schedule information is obtained directly by querying on line servers for each request.

3 Caching on a Java-capable Mobile Device

Caching is a common approach for storing and reusing received data in order to reduce volume of data transfered. From a database perspective caching means creating redundant data that needs to be maintained by the system. Major issues of such approaches are:

Coherency: the cache software has to guarantee that stored data is up to date. Especially in mobile environments this is not trivial. We decided to reuse an special attribute that specifies how long a data item is valid. If this date expires the item is removed from the cache and re-requested with the next query.

Replacement: Caches can use only a limited amount of memory. If this memory is full the software has to decide about which cached data item should be replaced by a new received data item. There are many approaches discusses in database literature (e.g. [EN04]). We implemented a standard FIFO replacement strategy.

Lookup: In order to reuse cached data, the system must be able to find cached data. As discussed in Section 2 semantic approaches are suggested for mobile clients. In contrast to them we use unique identifiers instead of query indexes and request complete data items. This can be assumed as a full table wise read only replication where the table name is used for cache lookup. So, neither semantic compensation nor query rewriting are necessary.

In the following we briefly present how to access an information system server via webservices from an MIDP-capable mobile device and introduce the Record Management Store that was used for storing data on the client.

[1] `http://persoenlicherfahrplan.bahn.de`

[2] `http://www.actuanmobile.com/documentation/train\_schedule\_manual-online-version-v0\_1.pdf`

Web services are accessed from mobile devices (J2ME clients) using the standard JSR 172: J2ME Web Services Specification [EY03]. The specification provides two new capabilities to the J2ME platform: (a) access to remote SOAP/XML based web services, and (b) parsing XML data

The specification allows for seamless integration of the web service programming paradigm in the mobile realm. At a high level, the architecture has three elements [Or04]: (1) A network-aware application which includes a JSR 172 stub generated with the J2ME Wireless Toolkit from the WSDL XML file. The stub uses the JSR 172 runtime to communicate with the network. The runtime has a Service Provider Interface which allows the stub to perform RPCs. (2) The wireless network and the Internet, and communication and data-encoding protocols (binary protocols, HTTP, SOAP/XML, etc.). (3) A web server (the service producer) offering access to back-end data and applications.

Method invocation follows the syncronous request-response model of client-server interaction. A mobile client invokes a request to a remote web service. The web service processes the request and sends back the result. The client then receives the data. The JSR 172 stub and runtime handle the encoding of the method and arguments, serialization, sending the request and receiving the response, decoding, and de-serialization, keeping all these transparent to the application.

Some mobile devices lack the implementation of this specification. For such clients, a method of accessing a web service is by sending SOAP messages and parsing the replies. However, this method is now obsolete and is prone to XML processing errors.

We used the MIDP Record Store Management for storing data on the client. A record store is identified by its name. A record store contains a number or records accessible by their index. A MIDlet application can persistently store data using the provided framework. A more detailed description of the MIDP record management store is given in [Gh02, Gi04].

Web services are typically bundled with XML-documents. However, saving XML data on a mobile phone is not feasible due to the limited ammount of storage. Hence, we need a method for transforming XML data into a more compact representation that can be easily stored and accesed from a mobile phone. In fact, we need a mechanism to transform XML provided by the web service into data that can be stored within an MIDP Record Store.

Data in XML format is hierarchical and it can be generally mapped into the Java Record Management Store while keeping an efficient index structure. The pragmatic method is to map each XML data record to a Record.

We use the name of the document as the key for the record store. Furthermore, we assume, the semantic of the data is understood by the application. In this respect, a simple binary serialization is required.

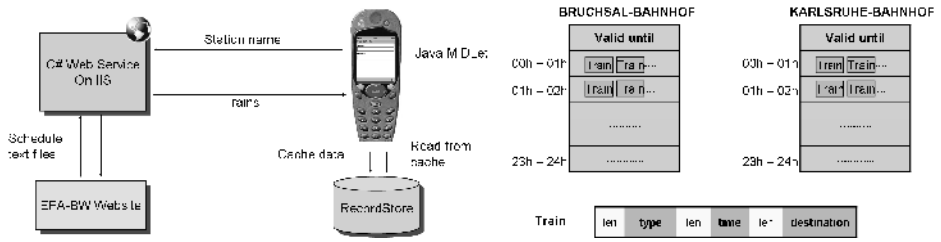
4 Evaluation

As mentioned in Section 2 we evaluated our ideas and implementation in context of a train schedule. The mobile application (see Figure 1(a)) connects to a web service to download

the latest schedules for a given station. The web service sends the schedule as a XML document including trains with their respective type, departure time, final destination, etc.

The MIDlet application caches the data into the persistent storage of the Record Store. If the user queries for the next trains for a given station, the application first checks whether the information is already present in cache. If the information is cached and the schedule is still up-to-date then the next departing trains are presented to the user. Otherwise, the MIDlet queries the Web Service for the latest schedule for the given station and caches the received response.

The data structure used for persisting the train schedule data on the mobile client tries to minimize the amount of information as well as having a structure allowing for efficient and fast lookups. The data structure is based on the capabilities of the MIDP Record Store described in the previous section.



(a) Train schedule application architecture

(b) Train schedule data structure

Figure 1: Implementation details

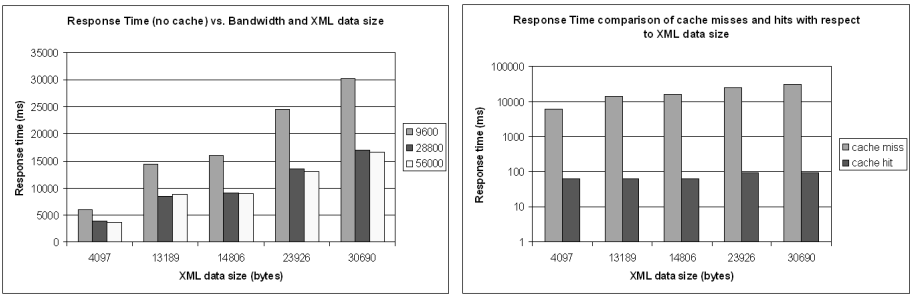
Figure 1(b) graphically depicts the data structure used for caching the train schedule information. We show this for two example train stations, Bruchsal and Karlsruhe. Each of the train stations is saved in a separate record in the record store.

For measuring the performance of our application we used the mobile emulator present with the J2ME Wireless Toolkit. The tests we performed represent the performance of the application for various parameters of the VM emulator speed and network throughput emulation.

Figure 2(a) illustrates for various XML data sizes the response time for answering a query if no caching is used. In fact, this means that all data needs to be accessed from the server. Figure 2(b) the response time in case of a cache miss to the case of a cache hit and shows, that answering a query locally is much faster than receiving the data over network. It is interesting to see, that the influence of the size of nearly requested data disappears if data can be provided locally.

5 Summary, Conclusions and Outlook

In this short paper we presented a proof on concept for caching data on an Java MIDP-capable mobile device. We discussed the implementation, illustrated the cache related



(a) (b)

Figure 2: Cache performance

decisions we made for the prototype and presented first evaluation results. However, regarding a full cache support for mobile phones, this paper is only a first step. In fact, for our implementation nearly all caching issues that are currently discussed in journals and on conferences were substituted by pragmatic approaches. Nevertheless, the results show that caching on mobile phones is really possible and has a huge potential to support the integration on mobile devices into information systems.

The next steps of our work are the implementation of some more efficient caching strategies and some more detailed evaluations.

References

[EN04] Elmasri, R. and Navathe, S. B.: *Fundamentals of Database Systems*. Addison Wesley. 4th. 2004.

[EY03] Ellis, J. and Young, M.: *J2ME Web Services 1.0*. Sun Microsystem, Inc. Santa Clara, CA, USA. Final draft. October 2003. Available at `http://jcp.org/aboutJava/communityprocess/final/jsr172/index.html`.

[Gh02] Ghosh, S. J2ME record management store – Add data storage capacities to your MIDlet apps. Online article at IBM developerWorks. May 2002. Available at `http://www-128.ibm.com/developerworks/library/wi-rms/`.

[GHOS96] Gray, J., Helland, P., O'Neil, P. E., and Shasha, D.: The Dangers of Replication and a Solution. In: Jagadish, H. V. and Mumick, I. S. (Eds.), *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*. volume 25 of *SIGMOD Record*. pp. 173–182. New York, NY, USA. June 1996. ACM Press.

[Gi04] Giguere, E. Databases and MIDP, Part 1: Understanding the Record Management System. Online article at Sun Developer Network. February 2004. Available at `http://developers.sun.com/techtopics/mobility/midp/articles/databaserms/`.

[HTKR05] Höpfner, H., Türker, C., and König-Ries, B.: *Mobile Datenbanken und Informationssysteme — Konzepte und Techniken*. dpunkt.verlag. Heidelberg, Germany. July 2005. in German.

[LLS99] Lee, K. C. K., Leong, H. V., and Si, A.: Semantic query caching in a mobile environment. *ACM SIGMOBILE Mobile Computing and Communications Review*. 3(2):28–36. April 1999.

[Or04] Ortiz, C. E. Introduction to J2ME Web Services. Online article at Sun Developer Network. April 2004. Available at `http://developers.sun.com/techtopics/mobility/apis/articles/wsa/`.