HoGent

Faculteit Bedrijf en Organisatie

Research of caching strategies in mobile native applications using external data services

Frederik De Smedt

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Joeri Van Herreweghe
Co-promotor:
Jens Buysse

Instelling: —

Academiejaar: 2015-2016

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Research of caching strategies in mobile native applications using external data services

Frederik De Smedt

Scriptie voorgedragen tot het bekomen van de graad van
Bachelor in de toegepaste informatica

Promotor:
Joeri Van Herreweghe
Co-promotor:
Jens Buysse

Instelling: —

Academiejaar: 2015-2016

Tweede examenperiode

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Voorwoord

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Contents

Chapter 1

Introduction

The definition of mobile has changed a lot in the last few years and the expectations of mobile applications continue to rise. Mobile applications should work on a myriad of devices and screen resolutions, and should take battery efficiency into account. Furthermore is an application no longer really considered mobile if it isn't available without a stable internet connection. This makes research about caching in mobile applications really interesting since it can solve multiple problems. Caching of the results of expensive operations, e.g. an internet connection, will allow the system to avoid many of these operations, reducing battery consumption, and can these results be used when a (temporary) connection loss would occur.

Lots of IT companies already implement such a system:

- The Facebook-app still shows several posts and pictures when you have no internet connection;
- Third-party API's, such as Picasso (Android) and SDWebImage (iOS) retrieve images from the web and have an embedded caching system;
- Twitter allows you to change account settings offline, which are later synchronized with the backend when a connection is established[1].

[1]In this example we are talking more about persistent data, however it is used to store temporary data that can later be forgotten once it is synchronized with the backend. Just like a cache containing business data.

1.1 Problem definition and research questions

However mobile caching is becoming more and more common, there is no real framework on which they all rely. Instead they each have to think about how to implement caching and have to invest in discovering caching strategies and inventing a good caching implementation.

They can decide to ignore the caching problem and focus on the business logic and the actual functional requirements of the project. The chances that the project is able to be completed within the deadline increases, as they have less work to do. However, if there is a non-functional requirement that requires some sort of caching or if the users of the application are complaining about problems that come along (e.g. lots of traffic or battery usage), you will have to implement the caching system afterwards. Which might force them to redesign parts of the system (both frontend and backend) or forces you to do this in an incomplete way, introducing lots of bugs that usually come along in caching systems. Every platform already has some infrastructure designed to allow caching, e.g. Guava and a local lightweight database. Yet there is more to caching than simply storing information, developers should think about what data is eligible for caching and how this can improve their application. They should be able to do this with an efficient method, customized to the needs of the application considering all possible events that might occur and how they might be able to handle this. These ideas and strategies should then be implemented in both the backend and frontend, however the backend implementation could be reduced to an almost non-existing implementation, since it is not merely about the native application itself, but about the data flow between the different systems.

Because of these reasons, we will try to answer the following questions in this article:

- How can a caching strategy efficiently store and fetch data in a native mobile application?
- What life-cycle events should be considered and how can they react to these events?
- How can the cache be synchronized with the external data service (backend)?

Chapter 2

Methodology

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

Chapter 3

Basics of caching

A cache can be defined as a (fast) memory structure designed to reduce costs and improve speeds of computer operations. The term caching is widely used in computer science, from the CPU that uses different multi-level caches, to web caches concerning the reuse of already fetched HTTP pages (Fielding, 1999) and all of them are designed to improve performance. Even low-end devices such as microcontrollers can benefit on the usage of a cache (Hruska, 2016). How this memory structure improves the performance is dependent on the use case in which it used. It could keep the results of long operations or use tokens to check whether data is outdated.

3.1 Basic cache performance metrics

The presence of a cache by itself will not cause a performance increase. It is integrated in a software application that uses a cache. The application then checks whether certain data is present in the cache and if not, does everything it needs to do in order to retrieve this data from a third party. In most examples a cache can be represented by a collection of key/value pairs, to which the application requests a value based on a key. When the cache contains the data the application is requesting, it is commonly known as a *cache hit*. Inversely, when the cache does not contain the requested data, it is known as a *cache miss*. The relationship between the total amount of requests and the total amount of hits is the *hit ratio*. $R = h/n$ where R is the hit ratio, h is the total amount of cache hits and n is the amount of cache requests. These terms are, among others, important factors determining the performance of a cache. A common misconception is that a high hit ratio results in better cache performance, yet this is only true considering hit ratios. If the hit ratio were the only factor, the caches "performance" would be proportional to the size of the cache and a cache of 10Tb always yield better performance than a cache of 20Mb (Wulf and McKee, 1995). This statement can easily be proven wrong when considering other factors such as speed, as the 20Mb cache will need far less time to produce a result than the cache of 10Tb.

3.2 Paging

However caching is of high importance with lots of aspects in IT, caching is what it is today thanks to some specific problems that have received lots of attentions the last decades. Perhaps the most important problem, considering the field of caching, is the problem with *virtual memory* (Denning, 1970). Virtual memory maps virtual memory addresses, provided by an operating system to a program, to actual hardware memory addresses. This way, the operating system can manage its memory resources and decide where to store and retrieve data from programs using virtual memory. This was considered a very elegant way to increase programming productivity, as programmers could use a high-level API to retrieve and store data instead of needing to talk directly to the processor, which would also make them machine dependent.

The operating system can assign priorities to memory addresses and decide whether it should be stored on a fast or slow memory. The memory of an idle process could for example be persisted on slow memory, i.e. hard drive, so that the faster memory, i.e. RAM, can be used for memory of currently active processes. The moment that the idle process wakes up, it can still use the virtual memory pointers, as if the data has never moved. When a process tries to access a piece of data, currently stored on the hard drive, the operating system will produce a *page fault*. This event will cause the *page*, a block of data, containing the desired data on the hard drive to be transfered to somewhere in RAM memory.

The memory transfer due to a page fault is however very expensive as the hard drive is a lot slower than RAM memory and the processor itself. Therefore it is important to have a good paging strategy, an algorithm that decides which pages will be persisted to the hard drive and which pages will remain in RAM memory. The optimal algorithm will only persist the pages that will not be used for the longest period of time.

The paging problem is very similar to finding the optimal caching strategy, RAM memory is the cache, the hard drive is non-cached memory and we are looking for an algorithm such that only the least interesting memory is removed from the cache. The principle ultimately solving the paging problem is therefore one of the fundamental principles not only in pages and virtual memory, but all of caching, this principle is called the *locality principle*.

3.3 Locality principle

3.3.1 What is the locality principle

When people are executing a process, such as preparing dinner, they always tend to gather everything they need before or during the process. If it were a bigger meal the entire meal is divided in a set of subprocesses. When preparing lasagne it could be divided in preparing the tomato sauce, creating each layer using previously prepared ingredients, etc. The principle of dividing a large task in a set of smaller subtasks is called *divide and conquer*. When creating the lasagne layers, we would first collect all supplies needed (tomato sauce, lasagne sheets, etc) and put them somewhere close, e.g. on the counter, once that is done, the layers are assembled. This is more practical and time saving than to continually walk a relatively long distance each time you need the tomato sauce.

Something similar has been discovered with programs by Denning (2005) when analyzing memory usage of multiprogramming systems using virtual memory in 1967. He formalized this idea through the years and called it the *locality principle* also commonly known as *locality of reference*. He found that programs intrinsically use a very small part of data when performing arbitrary algorithms. In 1980 he formally defined a relationship between the processor, executing the instructions, and the data/object it uses called the *distance* and is denoted as $D(x,t)$ where x is the object and t is the current time. The relevance of this object to the processor at a certain time is based on whether the distance is less than a certain threshold:

$$D(x,t) \leq T$$

where T is the threshold and D(x,t) is the distance function. The threshold and the distance are embedded in a, usually 2 dimensional, spatial-temporal coordinate space where the spatial dimension is the memory address of the object and the temporal dimension is the time when it was accessed.

3.3.2 Interpreting spatial-temporal distances

Because the distance function D is operating in a spatial-temporal space, it is based on the definition and interpretation of the two different dimensions. The easiest way to understand how these dimensions can be interpreted is to look at them seperately. Once that is understood, you simply need a function $m : S \times T \to \Theta$ where S is the set representing the spatial dimension, T is the set representing the temporal dimension and Θ is the set representing the spatial-temporal space.

Temporal distance, the distance in time that can be expressed in several ways:

- time since last reference
- time until next reference
- access time to get the object
- a combination of the above or others

The time since the last reference to object x can be known by storing the references used by the processor and the time when the reference happened in a list. The time until next reference might yield better results when trying to create an effective and efficient cache since it is known when the object will be used again. However this should require a learning agent that can try and calculate when the next reference will happen based on the reference history. The time unit used should be a relative unit such as the execution of an instruction, this is better than an absolute unit such as seconds because it will ignore external factors that shouldn't affect the distance, e.g. other processes running.

Spatial distance, the distance to the location of the object can be expressed in several ways as well:

- the amount of hops in a network required to retrieve the contents
- the memory on which the data is stored, e.g. objects stored in RAM have a smaller spatial distance than objects stored on a hard drive
- the number of memory addresses between the object and the addresses currently being used or pointed to
- a combination of the above or others

Intuitively, one could see a relationship between the spatial and temporal locality. When the spatial distance to an object is greater, the temporal distance usually also increases because it will need to cover more ground to retrieve the object. When the temporal distance is greater the spatial distance will usually increase because of pacification of the object: it will be stored on slower memory due to the lack of interaction. When checking the relevance of objects to the processor are checked, with $D(x,t) \leq T$, only some objects will pass the inequality, the set of these objects is called the *working set*. It are the objects in the working set that would, among others, be candidates for caching.

3.4 Caching algorithms

An *LRU-cache* (Least-Recently-Used Cache) is an efficient way of caching objects based on their temporal locality, by storing new objects to the cache while overriding the oldest cache lines/elements when the cache is full (Al-Samarraie, nd). There is currently no *trivial* cache mechanism focusing on spatial locality, this would be a cache keeping objects predicted to be accessed due to some spatial locality/localities. *Bélady's algorithm* is considered the most optimal caching algorithm available, as it discards the objects from which he knows it will not be needed for the longest period of time (Al-Samarraie, nd). Yet there are no claims whether this is the most optimal caching algorithm considering a dynamic cost for retrieving uncached data.

Chapter 4

Conclusion

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

Bibliography

Al-Samarraie, N. A. (n.d.). And page replacement algorithms: Anomaly cases. Technical report, Baghdad college of Economics Sciences.

Denning, P. J. (1970). Virtual memory. *ACM Computing Surveys (CSUR)*, 2(3):153–189.

Denning, P. J. (2005). The locality principle. *Communications of the ACM*, 48(7):19–24.

Fielding, R. (1999). Hypertext transfer protocol. Technical report, W3C.

Hruska, J. (2016). How l1 and l2 cpu caches work, and why they're an essential part of modern chips. http://www.extremetech.com/extreme/188776-how-l1-and-l2-cpu-caches-work-and-why-theyre-an-essential-part-of-modern-chips.

Wulf, W. A. and McKee, S. A. (1995). Hitting the memory wall: implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24.

List of Figures

List of Tables