Rodrigo Duro, F., García Blas, J., Higuero, D., Pérez, O., Carretero, J. (2015). CoSMiC: A hierarchical cloudlet-based storage architecture for mobile clouds. *Simulation Modelling Practice and Theory,* 50, pp. 3-19.

DOI: 10.1016/j.simpat.2014.07.007

CoSMiC: A hierarchical Cloudlet-based Storage Architecture for Mobile Clouds

Francisco Rodrigo Duro[a], Javier Garcia Blas[b], Daniel Higuero[c], Oscar Perez[b], Jesus Carretero[b]

[a]*Computer Architecture and Communication Area, University Carlos III, Avda. Universidad, 30, 28911, Madrid, Spain, frodrigo@arcos.inf.uc3m.es, +34916246260*
[b]*Computer Architecture and Communication Area, University Carlos III, Avda. Universidad, 30, 28911, Madrid, Spain*
[c]*Stratio, Avda. Europa, 26, 28224, Madrid, Spain*

Abstract

Storage capacity is a constraint for current mobile devices. Mobile Cloud Computing (MCC) is developed to augment device capabilities, facilitating to mobile users store/access of a large dataset on the cloud through wireless networks. However, given the limitations of network bandwidth, latencies, and devices battery life, new solutions are needed to extend the usage of mobile devices. This paper presents a novel design and implementation of a hierarchical cloud storage system for mobile devices based on multiple I/O caching layers. The solution relies on Memcached as a cache system, preserving its powerful capacities such as performance, scalability, and quick and portable deployment. The solution targets to reduce the I/O latency of current mobile cloud solutions. The proposed solution consists of a user-level library and extended Memcached back-ends. The solution aims to be hierarchical by deploying Memcached-based I/O cache servers across all the I/O infrastructure datapath.

Keywords:
Storage, I/O caching, Multi-level, Mobile Computing, Cloud

1. Introduction

A drastic increase in the number of applications and amount of digital contents such as pictures, songs, movies, and home films one the hand and the limited storage capacity of mobile devices on the other hand, decelerate

usability of mobile devices. While PCs are able to locally store a huge amount of data, smartphones space is limited to few gigabytes, which are mostly occupied by system files, user applications, and personal data [1].

Some recent examples of mobile cloud storage are Apple's iCloud, Google Drive, and Dropbox [2]. These solutions allow users of mobile devices to synchronize their application data such as photos, iTunes music, calendars, email, and messages. Although there is steady growth in mobile storage capacity, the ever increasing appetite of users for high-resolution videos and images promises the increasing popularity of cloud storage [3]. Given the current popularity of cloud computing and the current growing usage of mobile devices since the release of iPhone and Android, two approaches were instantly taken. First, adapting existing cloud services to mobile usage. Second approach is to use nearby mobile devices to collaborate in a common task.

Mobile Cloud Computing (MCC), as defined by Liu et al. [3] "is a model for elastic augmentation of mobile device capabilities via ubiquitous access to cloud storage and computing resources". An extended definition was proposed by Sanaei et al. [4] "mobile cloud computing is a rich mobile computing technology that leverages unified elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility to serve a multitude of mobile devices anywhere, anytime through the channel of Ethernet or Internet regardless of heterogeneous environments and platforms based on the pay-as-you-use principle". The objective of mobile cloud computing proposed solutions so far is to extend the capabilities of mobile devices, specially on their weakest areas: computing power, battery life, mobile network bandwidth and latency, and storage capacity.

Another problem addressed in cloud computing environments is data management. Currently, one of the techniques to optimize I/O systems in cloud environments is to decouple the virtual instances from the storage resources [5]. Moving information between different domains has never been a simple task. First, it is costly to deploy virtual machines that need to process a huge amount of data. Second, data access between geographically dispersed infrastructures is significantly increasing the latency perceived by users. Existing cloud computing tools tackle only specific problems, such as parallelized processing on massive data volumes [6] or large data storage [7]. However, these tools provide little support for mobile clouds, where data access is mainly limited by the network bandwidth and latency.

Recently, Abolfazli et al. [8] argued that there are open challenges waiting

to be resolved in the MCC research area. Among of them, we highlight seamless ubiquity, context awareness, and resource scheduling due to the following reasons. First, current mobile infrastructures have to ensure connectivity in all possible scenarios. Second, given the huge amount of data generated by smartphones, data must be allocated as close as possible in order to reduce transfer latencies. Third, an adequate usage of the resources is completely necessary for reducing system peaks and overheads.

This work aims to present the architecture of a hierarchical storage solution for large scale mobile cloud systems. The storage solution fills the latency gap between mobile devices and the final cloud-based storage systems. We present a cloudlet-based cache storage infrastructure, namely CoSMiC. Our solution could be used to deploy storage in-a-box systems, on all the levels of the datapath hierarchy. Mobile applications benefit from this solution by improving data locality, reducing application execution times, and saving money and battery life in mobile devices due to the use of Wireless Local Area Network (WLAN) connections instead of Wireless Wide Area Networks (WWAN).

The contributions of this work are the following. First, we present a cloudlet-based hierarchical storage system that reduces data access latency of current large scale mobile cloud infrastructures. Second, the proposed solution could be easily deployed on heterogeneous and low power computational systems, including clusters and clouds. Third, using both configurable hash and address algorithms included in Memcached [9] client library, the CosMiC front-end is completely decoupled from the I/O servers, resulting in an increase of scalability. Fourth, CoSMiC allows system monitoring, taking into account the usage of Memcached statistics with extended metrics.

The remainder of this paper is organized as follows. Section 2 reviews related work and background. Section 3 presents the design details of CoSMiC. In Section 4 we present some possible scenarios and deployment examples. In Section 5, we show our evaluation results. Finally, we conclude and discuss about future uses of CoSMiC in Section 6.

2. Related work

Mobile devices have very limited resources, being their main weak points computing power, storage space, and battery life. To augment computing power and improve battery life, highly compute demanding applications are offloaded to the cloud. In order to achieve this objective, several solutions

have been presented, some of them, based on the use of Hadoop [10, 11] over virtual machines and focused in determining cost-benefits of the offloading, considering also the data transfer required before and after computation. CloneCloud [12] and MAUI [13] are examples of compute offloading in distant fixed clouds, while Hyrax [14] and Phoenix [15] propose a solution for offloading between nearby mobile devices.

In the following subsection we present and compare previous works related to CoSMiC. We will focus on traditional immobile cloud solutions for storage and hybrid-based approach for MCC. Finally we discuss about similar works that rely on distributed caches as a storage infrastructure.

2.1. Immobile cloud resources for storage

The most common approach for mobile cloud storage is the adaptation of already known and highly used cloud storage services like Apple iCloud, Google Drive, Microsoft SkyDrive, or Dropbox[16], and cloud storage back-ends such as Amazon S3 [17] and Windows Azure Storage [18].

Currently, one of the most used cross-platform solution is Dropbox, which offers platform-independent storage, applications for almost every mobile and desktop platforms, secured data with AES-256 encryption, and highly-reliable Amazon S3 as storage system back-end. Also, to minimize the impact of synchronization, it uses binary-delta encoding functions to only upload the changes on each file. Apple iCloud, Google Drive, and Microsoft SkyDrive are leading alternatives and have the advantage of being embedded into their respective operating systems, but as their main negative point, they are cross-platform restricted and lack some of the features offered by Dropbox.

Cloud storage systems are directly related with distributed file systems, which usually offer the file management infrastructure (back-end). In fact, cloud storage can be seen as the evolution of distributed file systems for domestic users in front of typically business/research oriented distributed file systems. Similar to our proposed solution, the Ceph [19] distributed file system is currently growing in popularity. In order to avoid metadata access bottlenecks, Ceph takes advantage of a distributed metadata cluster architecture based on Dynamic Subtree Partitioning [20]. The file system is partitioned by delegating authority for subtrees of the hierarchy to different metadata servers. This solution allows to distribute the workload across the metadata hierarchy, which is fully independent. Our approach deals with metadata bottlenecks by distributing data and metadata across every available Memcached nodes. Another similarity between Ceph and our proposed

solution is the mechanism for obtaining and addressing data objects. It is not necessary to store the addresses of the blocks or objects used by each file on the metadata as done in traditional file systems. Ceph eliminates allocation lists entirely. Instead, file data are striped onto predictably named objects, while a special-purpose data distribution function assigns objects to storage nodes or storage devices. This allows any party to calculate (rather than look up) the name and location of objects comprising a file's contents. This solution eliminates the need of maintaining and distributing a list of objects, simplifying the design of the system, and reducing the metadata cluster workload. Ceph and GFS[21] save all the necessary information about the file to metadata servers and access directly from the I/O nodes without any further metadata access. Ceph and GFS follow a similar replication scheme, in which data object replicas are distributed through the cluster, at least one copy in the same rack and another outside of the rack containing the original data.

2.2. Hybrid mobile cloud solutions

The 'cloudlet' concept, classified as "inmobile or fixed local cloud" by Abolfazli et al. [1] was introduced by Satyanarayanan et al., [22] as "a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and available for use by nearby mobile devices". The objective of this kind of solutions is to present a hybrid alternative between resource-restricted ad-hoc mobile clouds and geographically distant public cloud services. As shown in Figure 1, cloudlets are usually situated close to the mobile devices and connected with them via WLAN (e.g. Wi-Fi) and access to feature-rich public cloud services.

Cloudlets are the new trend over ad-hoc mobile cloud computing because they offer some advantages impossible to achieve by mobile devices. First, cloudlets computing capabilities are higher than mobile devices. Second, cloudlets do not need to be mobile and can be plugged, giving them the advantage of infinite power source and better high performance wired internet connections. Main advantages of cloudlets over mainstream public cloud services are due to the proximity of the cloudlets: mobile users can take advantage of a lower latency in their requests, and an extended battery life result of the use of Wi-Fi instead of WWAN (3G/HSDPA/LTE) [13, 23]. Given all these advantages, Sakr et al. [24] determined that cloudlets are one of the main research directions to achieve a better quality of service in

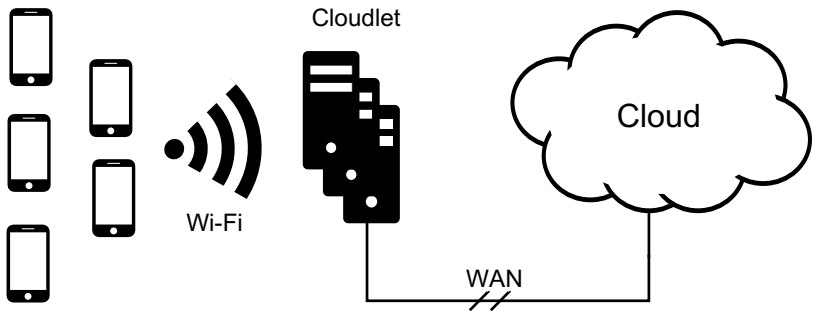

Figure 1: Representation of a cloudlet infrastructure. Cloudlet nodes are resource-rich and have high bandwidth/low connection access to the Internet. They are accessible from mobile clients through high speed and low-energy WLAN networks such as Wi-Fi and are designed to reduce latency related problems in mobile cloud environments due to WAN/WWAN connections.

MCC architectures. We classify the hybrid mobile cloud solutions in four categories: infrastructure, computation, storage, and energy efficiency.

Infrastructure. Next-Generation Hotspot (NGH) infrastructure [25] is a solution proposed by Cisco to improve the functionality given by adding Hotspots. The objective of this solution is to alleviate the load over the wireless cells, deploying new hotspots in highly populated areas. CoSMiC shares with NGH the objective of alleviating WWAN radio stations offering users WLAN connection. However, NGH offers hotspots without any kind of storage purposes or specifically cloud-oriented functionalities. CosMic can take advantage of NGH by improving Wi-Fi usage and confidence in mobile end-users. These users should be more confident to use our system when deployed within CoSMiC covered areas. SAMI (Service-based Arbitrated Multi-tier Infrastructure for mobile cloud computing) [26] addresses the latency problem by using a service-oriented architecture (SOA) and involving the Mobile Network Operators (MNO) to deploy authorized dealers of services in highly populated areas. CosMiC shares with SAMI the objective of reducing both latency and WWAN networks congestion. We also share the approach of deploying in social locations such as shopping malls, airports, or commercial buildings. The main differences are its service orientation, and its dependence from MNOs: first, CosMiC focuses on Storage as a Service (SaaS), while SAMI alleviates and manages services on behalf of the cloud providers, directly in the MNOs; second, CoSMiC is MNO-independent and is deployed on top of network providers, while allows to offer customizable services. As

described in Section 3, instead of using a SOA infrastructure, CosMiC relies on Memcached client libraries, available for several programming languages. Abolfazli et al. [27] present a market-oriented MCC architecture (MOMCC) that faces the problem with a different approach. Based on a SOA, the objective of this solution is to exploit nearby mobile devices to alleviate the WAN latency, usually perceived by mobile users of cloud computing. While the objectives pursued by this approach are shared with CoSMiC (reduce both WAN latency and congestion on radio base stations), the MOMCC is completely different. The use of nearby mobile devices improves availability of low-latency services, but it also incurs in some problems: battery life on mobile host devices, unpredictable availability of close devices offering this specific service, MNO's mandatory involvement, and costly deployment of the solution.

Computation. Several solutions for computation offloading in nearby compute nodes has been presented in recent years (e.g. Cloudlets[22], MAUI[13], and mobile volunteer computing [28]). Soyata et al. [29] present Cloud-Vision, a face recognition solution that relies on a cloudlet architecture. In Cloud-Vision, the costly execution process is distributed to multiple cloudlet with higher computation capabilities. However, authors do not comment how data are transferred between mobile devices and cloudlets and its corresponding cost. CoSMiC could help these applications to reduce data transfers in the cloudlet infrastructure. Sanaei et al. [30] introduced the hybrid pervasive for MCC (HPMCC). HPMCC relies on a multi-tier MCC infrastructure based on the proximity of the involved devices. HPMCC paradigm targets to make a better usage of mobile devices and services by more transparent and context-aware MCC-based solutions. Authors propose that security and privacy methods can be offloaded from mobile devices to other entities like MNO.

Storage. Storage Cloudlets approach has been explored in a less deeply way than computing offloading approaches. Xu et al. [31] pointed out that the transmission of large data items should occur within a tight user-machine interaction loop. MoCa (Mobile Collaboration Architecture) [32] is not specialized in cloud storage, but it shares with our solution its focus on mobile clients paired with the closest possible proxy that offers the service wanted by the client, instead of using directly the servers providing this service. MoCa was released in 2004, and while its algorithm to discover the closest proxy, based on RADAR [33], is really powerful, the complete solution is not focused on MCC, lacking most of the advantages offered by this new tech-

nology. Phoenix [15] provides a transient distributed data storage system for mobile devices that leverages opportunistic use of cloud of nearby mobile computing devices in ad-hoc manner. In Phoenix, mobile devices cooperate as a P2P network, distributing data replicas in order to ensure data availability. However, while Phoenix is currently working in one-hop networks, CoSMiC relies on a completely different approach: First, CoSMiC aims to reduce battery consumption by reducing data transfers, mobile devices do not collaborate in the storage architecture; Second, performance is obtained by caching data in the cloudlet-based storage nodes; Third, CoSMiC uses a single datapath, which avoids expensive coherence protocols.

Energy. Finally, the energy efficiency of data transfers is highly penalized by the latency and low bandwidth, as shown by Miettinen and Nurminen [34], and Balasubramanian et al. [35]. eTime [23] proposes an energy-efficient strategy for data transfers between mobile devices and the cloud based on Lyapunov optimization. The objective is to take advantage of periods with good connectivity to prefetch data while delaying transmissions in cases of bad connectivity, when possible. The proposed solution relies on the improvements in battery life under good conditions. Their results demonstrate that prefetching is a powerful way to improve battery life and should be considered for hybrid mobile cloud storage solutions.

Our solution stands out in the state-of-the-art of MCC in four novel features. First, it has been specifically designed for two of the main problems in the current MCC storage scenarios: latency and battery life. Second, these problems are targeted by a multi-level hierarchy for caching objects. Third, the solution aims to adapt to any existent network topology being possible to deploy any number of caching level with any number of CoSMiC storage nodes in each level. Fourth, CoSMiC has been conceived around the novel concept of cloudlet to achieve a trade-off solution between pure mobile storage and pure remote cloud storage.

2.3. Distributed cache memory solutions

Nowadays there are multiple alternatives for distributed cache memory solutions. Most of them rely on key-value store system such as Couchbase [36], Redis [37], Cassandra [38], and Memcached [9]. These solutions provide a distributed NoSQL database, providing scalability and performance.

Memcached is an open source distributed memory object caching system typically used in web environments to reduce the latency of web requests.

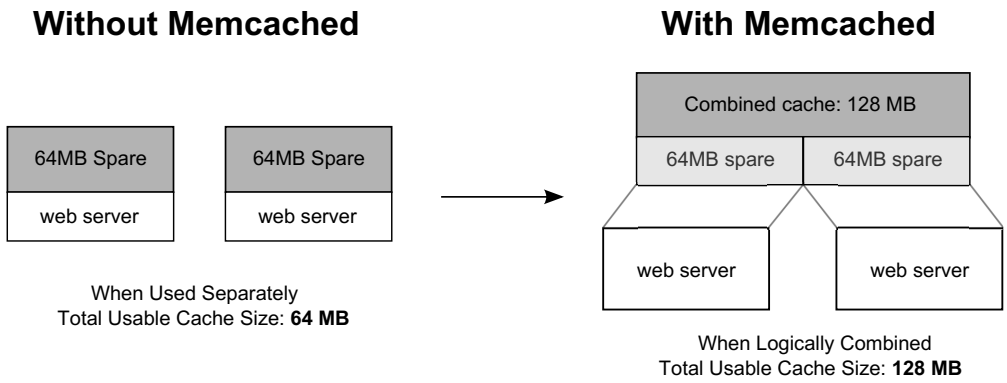

Figure 2: Spare memory fragments can be used as an unique cache space in Memcached. With Memcached, all of the servers share the same virtual pool of memory. A given item is always stored and always retrieved from the same location without needing any communication between clients or between servers.

Memcached is an in-memory key-value storage system based on the formation of a unique caching space among the involved servers (as shown in Figure 2). In this way, a common cache view is maintained through the use of a Distributed Hash Table (DHT). Memcached can take advantage of unused memory in the system and easily aggregate it to the cache pool.

One of the main benefits of using Memcached as a caching system is the decoupling of the potential clients, and the decoupling among the servers that are part of the DHT. Each client maintains a list of the available servers, and redirects its requests to them based on the hash of the requested items. By default, the server is chosen applying MD5 over the block key and calculating the modulo with the number of servers, as shown below:

$$\text{Destination Server} = \text{MD5}(key) \quad \% \quad \text{No. Servers} \tag{1}$$

The hash and selection functions are easily modifiable at client-side, permitting a high level of customization of the load balancing. It is even possible to select a consistent hashing like in Ketama [39].

By default, Memcached allows to store key-value pairs with up to 1 MiB raw data (up to 128 MiB in experimental mode) indexed with up to 250 bytes keys. Memcached was designed with the objective of caching data related to web applications, it can store any kind of data inside the value associated to each key.

On the server side, each server divides its available memory in different

slabs. Each slab stores key-value pairs within a size range in order to reduce memory fragmentation. In case that no free space is available in a slab, a Least Recently Used (LRU) policy is applied to evict items from the cache. Additionally, a slab can be resized if near ones are empty. As the architecture does not involve any centralized coordinator, Memcached is a highly scalable architecture, typically used in high-throughput scenarios, where the objective is to reduce the latency of user requests. Memcached is currently used in high-demanding web sites such as YouTube, Twitter, and Wikipedia [40].

Relative to high performance I/O and non-persistent storage, we present some approaches that have to be taken into account during the design of a cloud storage solution. Nahanni Memcached [41] is a solution totally applicable to our proposed cloud storage system in a multi-tenant environment. Nahanni Memacached uses mechanisms of shared memory between virtual machines running on the same host to avoid communications over sockets, achieving much better performance. It is important to remark that there are previous works using Memcached in a distributed file system. Wang et al. [42] benefit from Memcached by promoting read throughput in a massive file system, in which small files are predominant. In contrast, our proposed solution supports any file size given that a multiple-block storage system is implemented on top of Memcached.

3. CoSMiC: Storage cloudlet in-a-box

A comparison of a classical MCC model and CoSMiC infrastructure is shown in Figure 3. Classical solutions access to cloud services via wireless connection, using an antenna (WLAN or WWAN). A datapath through the ISP's network infrastructure to the cloud storage service is then established. Our solution proposes deploying multiple number of cache levels before the ISPs network infrastructure, reducing the latency produced by requests through WANs, following cloudlet's spirit.

CoSMiC reduces the connection latency and improves battery life through the use of Wi-Fi instead of 3G, but it is applied to storage adding hierarchies to the caching system, permitting the deployment in a hierarchical way. The number of cache levels and the number of servers on each level is adaptable depending on the characteristics of the infrastructure where the deployment will be done.

CosMiC relies on Memcached as a distributed cache system, using the distributed cache as a virtual I/O device where file blocks are mapped into

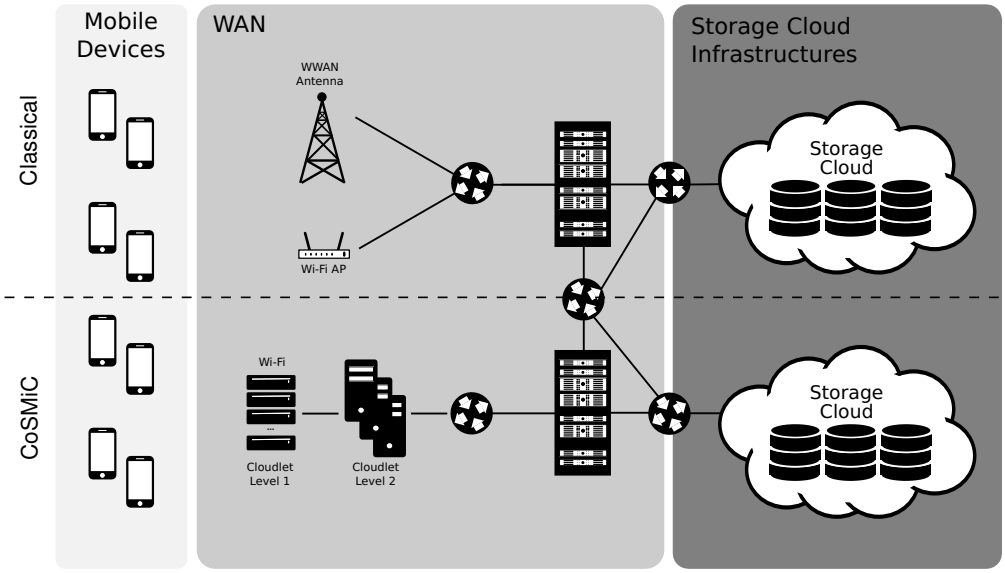

Figure 3: CoSMiC architecture. On top current network path of user requests to cloud storage services. At the bottom, hierarchical storage cloudlet solution deployed between mobile device clients and storage cloud services.

key-value items. We have included two new features on top of the initial design of Memcached. First, evicted items are saved in a persistent storage. Our generic design permits to use any persistent storage available through the use of plugins (e.g., local file system, Lustre, NoSQL databases, cloud storage back-ends like Amazon S3, etc.). Second, our solution can build hierarchical cache systems by connecting different levels of Memcached back-ends, using a new dedicated plugin. Its generic design permits to easily adapt to existing hardware and software configurations. Portability is guaranteed due to the large number of the developped libraries for Memcached [1].

In the following subsections we detail both hardware and software architecture details of CoSMiC.

3.1. Software architecture

The proposed software architecture is composed of two main components: an user-level file library and the CoSMiC back-ends. The user library maps

[1]List of available Memcached client libraries for multiple programming languages at https://code.google.com/p/memcached/wiki/Clients.

I/O file operations of client applications into item requests. CoSMiC back-ends receive the requests managing the access to the persistent storage or next level of the hierarchy. The next subsections describe in detail the file mapping mechanisms, the structure of the user library, and the internals of the CoSMiC back-ends.

3.1.1. User-level file library

Mobile applications employ the CoSMiC library in order to access files across the CoSMiC hierarchy. The layered design (see Figure 4) provides an interface designed for any kind of mobile application. The general-purpose interface (libCosmic) manages the mapping of POSIX-like file I/O operations into Memcached key-value pair requests (more details in Subsection 3.1.2). Requests are then sent through an unmodified Memcached client library to a previously defined CoSMiC back-end pool, based on Memcached back-ends.

CoSMiC is also able to deal with key-value items directly given that, as shown in Figure 3, it takes advantage of the Memcached library. This approach increases CoSMiC portability by supporting current applications that take advantage of NoSQL-based data accesses.

3.1.2. File Mapping

CoSMiC maps file blocks into key-value pairs. Figure 5 shows the transformation of a file into a collection of key-value pairs. First, we create a metadata key-value pair whose key corresponds with the file name and the value stores the associated metadata. The metadata contains a unique identifier of the file and its size among other common metadata. The file identifier is generated during the creation of the file, by applying a SHA-512 hash over the file name [43]. This assures that every identifier has the same length, and provides a good distribution of keys over the global key space. Each data block is stored using a key-value pair whose key corresponds to the hash of the file identifier and block offset, and the value contains the raw data.

In order to calculate the destination back-end node, the user-level file library concatenates the unique ID with the block offset, resulting in a string with format $> ID > offset$. As next step, an MD5 is applied to this string. As last step, the server is selected by the Memcached client library. This calculation does not suppose a great overhead, given that primitives such as MD5 and SHA are commonly used in the mobile community. In our case, two calculations are needed, file hash (one-time SHA to calculate file ID) and

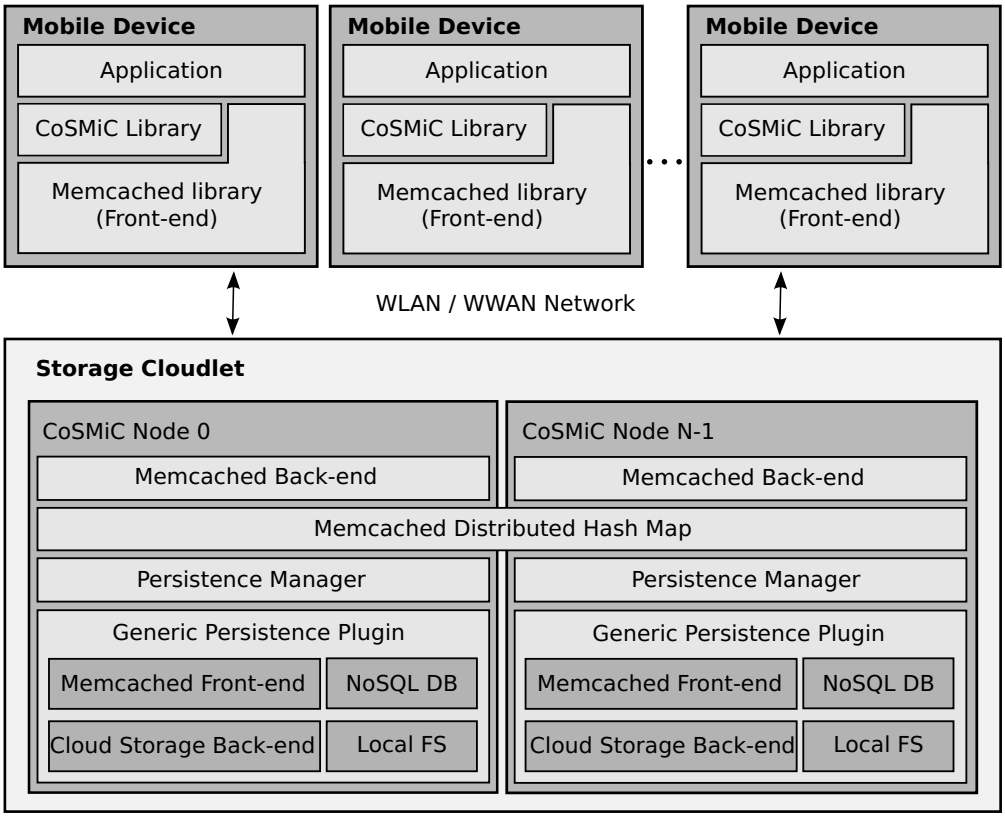

Figure 4: CoSMiC architecture is divided in two main components: a user-level library for applications and CoSMiC back-ends. The user-level library is layered and offers POSIX-like interface. CoSMiC back-ends improve Memcached servers' functionality by adding a persistence manager with different plugins, easily implementable and switchable.

server hash (MD5). Both hashes are performed over small keys (around 250 bytes) simplifying the calculation.

It is important to highlight that unlike in typical file systems, where it is necessary to store pointers to the data blocks, in our solution the keys required to access them can be calculated on demand per file. CoSMiC minimizes the metadata requests for the location of a particular data block, and it is capable of treating equally data and metadata blocks as key-value items, fully distributing them among all the CoSMiC nodes. The fully distribution of metadata blocks instead of the use of a centralized metadata server, reduces possible bottlenecks produced by heavy metadata accesses.

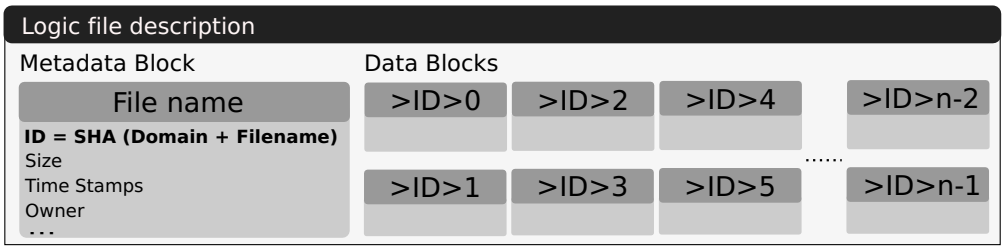

Figure 5: CoSMiC file description. In the CoSMiC client library, a file is composed by one item that contains the metadata and several data blocks. Data block are accessible by the ID metadata field, stored in the metadata block.

3.1.3. Storage back-end

CoSMiC back-ends leverage the design of Memcached while introducing new capabilities. In Memcached, items evicted by the LRU eviction policy become unrecoverable. As shown in Figure 4, we have extended the existing features of CoSMiC by introducing a Persistence Manager layer that is in charge of storing these items before becoming unrecoverable, by moving them to any of the available Generic Persistence Plugins (GPP).

The GPP layer transfers evicted items to another level of the hierarchy or to any kind of storage sub-system, such as cloud-based storage solutions like Amazon S3 [17] and Windows Azure Storage [18], or even to a local file system or NoSQL database for private systems. In case of using a persistent file system, data blocks are stored as a regular file. For totally local deployments, metadata can be stored in Berkeley DB, a NoSQL-based database similar in functionality to Memcached. CoSMiC offers the possibility to use another layer of CoSMiC back-ends, resulting in a hierarchical structure and allowing the deployment of storage cloudlets with multiple cache levels. Due to the layered design of the architecture, it could be even possible to implement a GPP for popular cloud storage services, like Dropbox, and caching this kind of accesses in our proposed system.

Additionally, in order to increase Memcached back-end performance, we have included three new optimizations. First, a dirty byte has been added to the Memcached item struct. Only items marked as dirty need to be stored before eviction. Second, a Preemptive Eviction Module (PEM) has been implemented. The items that will be evicted soon (last elements of the LRU) are stored into the next layer of the hierarchy but preserved in the current layer for locality reasons, marked as clean. This action simplifies the eviction

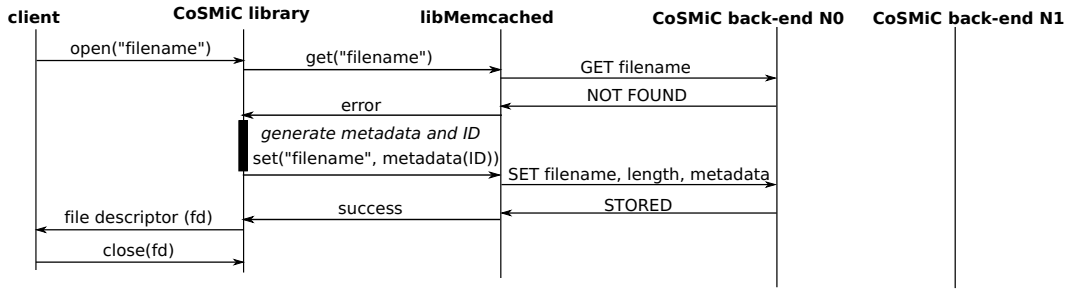

Figure 6: Message passing protocol for file creation in CoSMiC.

when needed, items that stay in clean condition will be deleted without any further action because they are already stored in lower layers, and their state should be dirty if they have been modified after the preemptive eviction. Third, we have implemented a buffering system for the evicted items. Items are copied to the buffer and flushed in an asynchronous way in low load conditions. This buffering system can be configured varying the number of active buffers. Items are only stored in one buffer based on a hash function, resulting in a quicker search and a better throughput performance by a better use of the socket transferring multiple blocks in parallel, one for each buffer.

3.1.4. Message passing protocol

Figure 6 shows a file creation in CoSMiC. When a file is created, first the CoSMiC library/front-end request an item with the file name. If this item does not exist, the file metadata object is created as a new item in the CoSMiC back-end node. Then, the CoSMiC library assigns a file descriptor to this file. It is important to note that CoSMiC allows to isolate metadata by configuring the CoSMiC library by using another CoSMiC level for only storing persistent metadata.

An example of write and read accesses protocol is shown in Figure 7. In case of write accesses, clients open the file obtaining a file descriptor. A write operation is mapped into two consecutive calls of Memcached. One write request of 6000 bytes is translated to two request of 4000 bytes each, assuming a block size of this length. We highlight that a large block size can incur in a transfer overhead. Then, the libMemcached library distributes requests across the CoSMiC back-end. Finally, the CoSMiC library updates the file metadata (file length and modified date). Other file operations such as *lseek* of *close* do not involve message exchange.

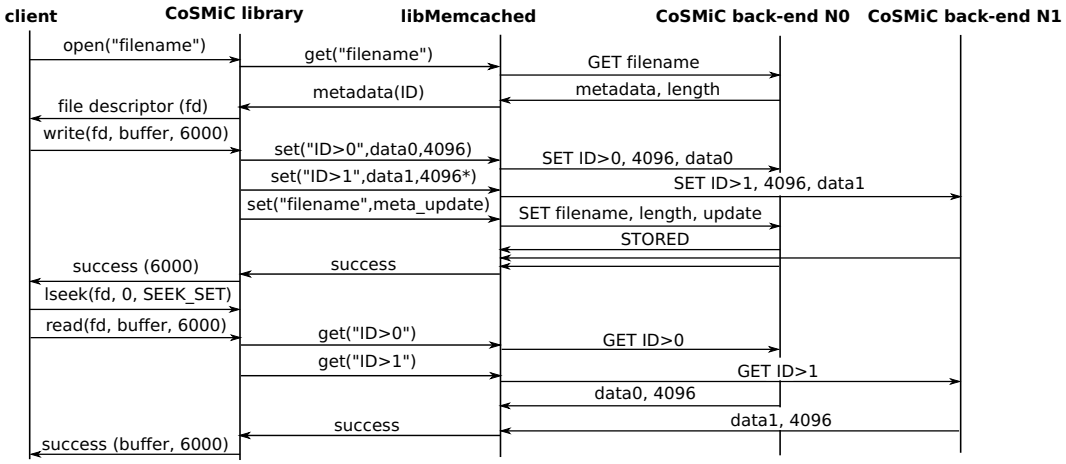

Figure 7: Message passing protocol for file edtion (write and read) in CoSMiC. The CoSMiC library translates POSIX call to Memcached primitives.

3.2. Hierarchical datapath

Nowadays, storage systems are one of the main bottlenecks in cloud systems. Furthermore, it is greatly accepted that large scale storage systems will be necessarily hierarchical [44]. This is done by organizing the memory spaces in a complex hierarchy, moving data to local caches as fast as possible and throwing it to slower devices in an asynchronous way [45]. This hierarchical structure should be constructed with decoupling in mind, which consists of splitting and isolating application devices, forwarding nodes, and storage nodes as much as possible.

As can be seen in Figure 8, the system offers the possibility to be structured hierarchically. The objective of this approach is to offer as much flexibility as possible. If it is possible to deploy the system with every node at the same cache level, it can be done. If the topology of the network is not optimum for this kind of deployment (e.g., different bandwidths/latencies in different segments of the network), the deployment can be done exploiting the characteristics of the network. Any number of nodes can be deployed in each one of the hierarchy levels and any number of hierarchy levels can be used in the architecture. As the number of levels deployed increase, the latency to access to the back-end layer is higher. However, the objective is to reduce the impact of WAN access latency using the cloudlet caching nodes, while asynchronously flushing items to lower levels of the cache, being imperceptible by the clients.

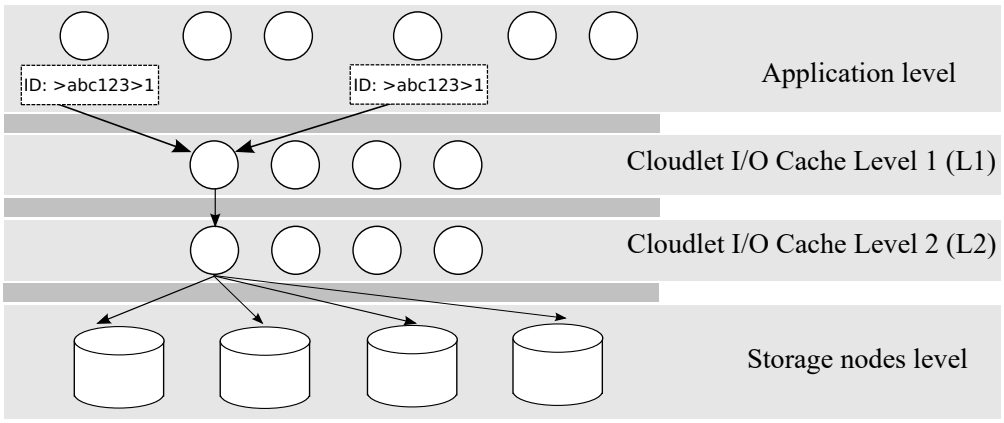

Figure 8: CoSMiC hierarchy. The flexibility and easy deployment of our proposed solution permits any number of cache levels, each one involving any number of nodes seen as a unique caching space. Due to the different Generic Persistence plugins available, there are a lot of possibilities for the storage back-end in the last level of the hierarchy.

Figure 8 represents a hierarchical deployment of CoSMiC. The hierarchical deployment of CoSMiC is both single-copy and single-path, only one copy of the item is maintained per level and items are mapped to a specific I/O node, respectively. In this way it is not necessary count with expensive coherence and consistence protocol, improving the performance. In Figure 8, two devices access the same item, therefore, requests are mapped to the same I/O node. Then, this node caches and forward this item to he next level of the hierarchy up to reach the final storage system.

Another capability offered by the hierarchical deployment is to set-up top levels of the hierarchy, with as many little servers as possible, spread over the desired area, without needing extreme power, cooling, or space requirements. Every one of this storage in-a-box nodes can be connected to near storage servers as next cache level. This new level reduces limitations with improved features: unlimited power sources, high-bandwidth network and enhanced computing capabilities. In addition to improved capabilities, this level can be deployed in a properly configured room close to the first level nodes. Even different domains can be deployed in different areas of the same building.

3.3. File domains

The file system could be deployed into multiple domains thus to separate files and resources based on their intended usage. This separation provides

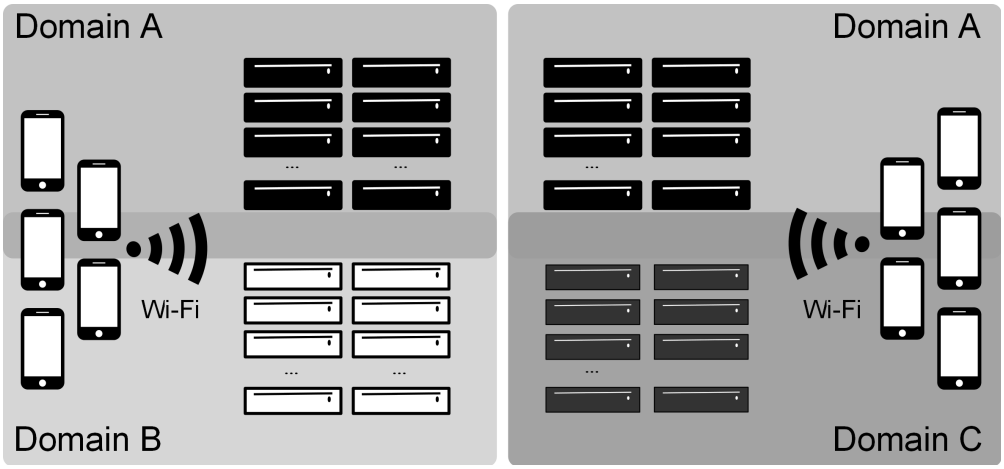

Figure 9: Different file domains can be deployed in different areas based on content-needing of a specific context. Domains can be shared between different areas.

simplicity to the user, who only has to worry about a specific subset of files. Arranging files by domain also lets the system to apply blanket access privileges to files in that domain, preventing unauthorized users from changing files intentionally or inadvertently. In CoSMiC, a file domain is defined as pool of Memcached back-ends. Figure 9 represents a CoSMiC infrastructure composed by three different domains. Different domains can be deployed on the same area. Also it is possible to offer the same domain in different distant areas.

3.4. Deployment

CoSMiC supports multiple scenarios. The concept storage cloudlet in-a-box refers to its flexibility and easy deployment. CoSMiC can be installed into multiple hardware architectures, even in really simple computers such as PlugComputers, Intel NUCs, or ARM-based mini PCs (such as Raspberry Pi and ODROID). The storage cloudlet is composed by a cluster of low consumption devices, as explained previously. This approach has various benefits: (1) it increases the Memcached cache capacity by adding additional nodes; (2) load-balancing in case of peaks; (3) CoSMiC back-end nodes can share an Internet address (NAT) or use a public range.

As previously stated, CoSMiC back-end nodes run without information about any other node in the same caching level. CoSMiC back-end nodes are configurable via command line. Block size can be configured from 1 KByte

to 64 MBytes. The GPP can be configured to work in local persistence mode (persistence data path) or hierarchic mode. This mode should be configured with the IPs of the CoSMiC back-end nodes deployed in the next caching level. CoSMiC back-end nodes are fully retro-compatible with Memcached, including all the possible command line arguments such as port mapping, monitoring, slabs configuration, etc.

Clients configuration is straightforward. Client only need the IPs of every CoSMiC back-end node inside the first level of their specific domain. In order to facilitate the IPs configuration, a discovery service can be deployed for an specific domain. Block size is also configurable in the CoSMiC clients library through the use of environment variables.

4. Application scenarios

In this section we will offer different scenarios that can take advantage of CoSMiC, centering the explanation in the infrastructure of the different possible deployments.

4.1. Public places

The first scenario that we address it is a large building or a crowded place. Examples of this scenario are airports, universities, stadiums, concert halls, cinema halls, theaters, and shopping malls. Even outdoor spaces with a minimum network and electric infrastructure, like music festivals, campus or crowded events, can be considered part of this scenario. As an example, we will focus our explanation of this scenario in shopping malls because of its characteristics specially favorable for our proposed solution.

Figure 10 shows an example of deployment in this scenario. Different zones of the scenario are separated by dashed lines, and represent different physical zones of the scenario. In a shopping mall could be seen as a stores zone and a restaurants zone, and in a university could be seen as different departments. Mobile clients are able to know which servers are in each zone and which domains are available in each zone by a discovery service based on the currently connected Wi-Fi access point. As the figure shows, different caching levels can be deployed. In this case, two levels have been considered enough for a shopping mall, but is possible to deploy any number of levels. Any number of nodes can be deployed in any of the levels of the hierarchy and in any of the zones of the scenario, however, the number of nodes should be dependent on the system load.

In the first layer the nodes can be servers, personal computers, or low power computers, as defined previously in our storage cloudlet in-a-box concept. It is recommendable to have a wired LAN connection between caching layers but, as can be seen on the top of the figure, is not mandatory, as nodes can connect to the next caching level through a WLAN connection. The second level of the system can be deployed in large computers in a room specially qualified for that matter, it can even be possible to use a data center if it is available. This second level should be well-connected to Internet through a high-bandwidth connection in order to retrieve and store objects in a cloud storage back-end, like Amazon S3 or Microsoft Azure.

It is extremely important that mobile devices connect with the storage cloudlets through a WLAN connection (such as Wi-Fi or even Bluetooth can be considered in very special scenarios). The use of Wi-Fi has two main advantages for the users over WWAN connections: the latency and bandwidth are improved and the battery life boosted. If the requested data is cached, the response should be faster than WWAN connection, while not cached data response should be similar to WWAN. The increase of latency through the different caching levels should be compensated by the better wired WAN connection of the building.

The shopping mall example is specially interesting in this scenario, because, as said by Abolfazli et al. [27], "the number of computers in public places such as shopping malls, cinema halls, airports and coffee shops is rapidly increasing. These machines are hardly performing tense computational tasks and are mostly playing music, showing advertisement, or performing lightweight applications". Our system can take advantage of these underutilized systems deploying instances of CoSMiC on them. In addition, due to the lightweight computing requirements of our proposed solution, CoSMiC nodes can be used to offer other services, like custom cloud services for customers or computing offload for nearby mobile devices.

An essential stakeholder of this scenario is frequently forgotten. Apart from the service provider (e.g. shopping mall, airport, etc.) and the mobile clients, it is essential to analyze the importance of the Internet connection provider, also known as Mobile Network Operator (MNO). It is easy to explain how our proposed approach helps mobile device users and storage cloudlet owners.

On the one hand, mobile device users improve their bandwidth and latency over WWAN data access and save battery life and money due to the use of WLAN instead of contracted data plan. On the other hand, the owner

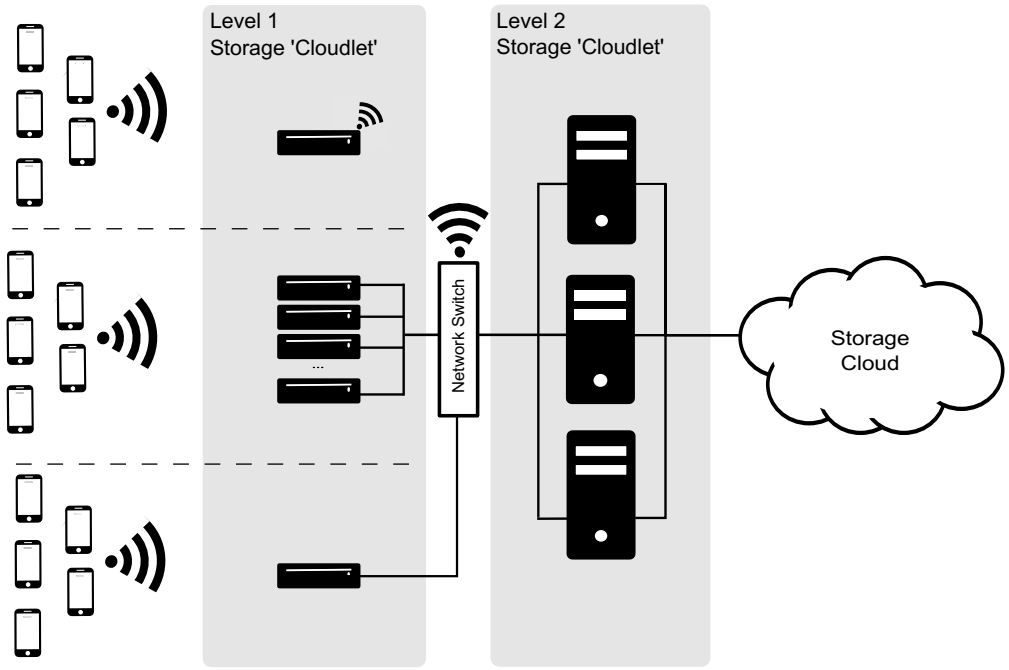

Figure 10: CoSMiC infrastructure deployed on a public place. Cluster of mobile devices connect to one or more storage cloudlet-base nodes (level 1), which forwards and cache I/O blocks to the the next hierarchy level (level 2). Finally, data blocks, are stored in a cloud-based storage infrastructure.

of the storage cloudlet has a lot of benefits with a small cost of deployment and ownership, with the ability of reuse existing infrastructure: attracting new clients, offering better or customized cloud services, and, facilitating the Internet usage habits of their clients. It could even be possible to propose special offers to frequent customers, offering other benefits for using the system, in a similar way to Four Square offers[2]. It can be even possible to offer some services based on the context of the user, in a similar way as proposed by Apple with their iBeacons[3] but using nearest Wi-Fi access point instead of Bluetooth.

But, as said before, MNOs are usually forgotten in this kind of scenarios, while being a major participant. Figure 11 shows a simplified vision of a

[2]http://www.foursquare.com
[3]http://support.apple.com/kb/HT6048

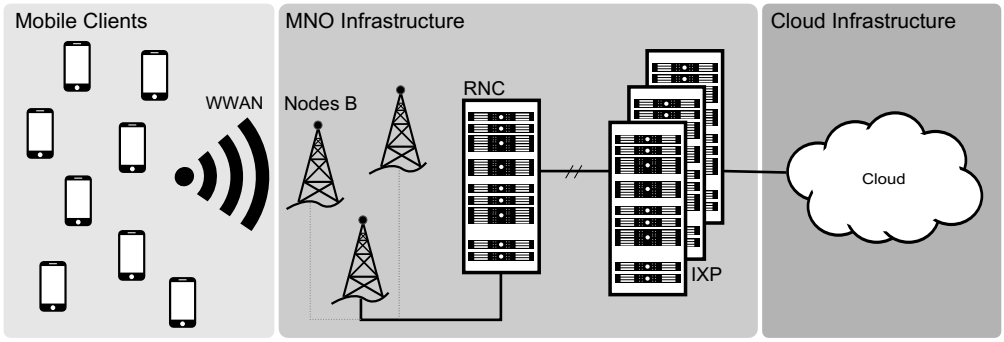

Figure 11: Simplified vision of a Mobile Network Operator infrastructure (MNO). The MNOs offer connectivity from mobile devices to geographically distributed cloud storage services or back-ends. We have used the UMTS approach where mobile devices connect to a Node B operated by a Radio Network Controller (RNC). The request is routed through a land infrastructure and, if necessary, data is exchanged with other Internet providers on the Internet Exchange Nodes (IXP).

typical MNO infrastructure for mobile data networks. Requests from mobile devices are sent to the nearest Node B. A Node B is a radio base station for UMTS connections. Nodes B have a very basic functionality and are controlled by Radio Network Controllers (RNC). A small number of near Nodes B share the same RNC. This RNC routes user requests through a wired network, in a similar way as done in WAN connections, so apart from the possible caching mechanisms present in the RNC, from that point on, its behavior is similar to WAN connections like xDSL, FTTH or cable. The last interesting item in the schema are the Internet Exchange Points (IXP), the last point of the infrastructure of MNOs. In these IXPs, different Network Operators (mobile or not) exchange data between their networks. This traffic exchange is the most expensive traffic for MNOs because it happens outside of their own networks.

In order to reduce costs at the Internet Exchange Points, MNOs try to cache as much data as possible inside their own networks. Most popular contents are popular in large areas, even in entire countries, like social networks, news pages, public administration websites, etc. But a lot of contents are popular only in small areas: local news pages, viral YouTube videos only in a specific city, etc. This kind of content is easily cached near its consumption, for example, in RNCs. But another kind of content popularity is possible: context-specific contents. Popular contents consumed in a shopping mall or a university could be slightly different from the most consumed contents in

the area covered by the RNC (100 meter to 8 km areas).

Storage cloudlets can cache this context-specific content and offer two main advantages to NMOs. First, less requests are done to Nodes B near highly populated areas with storage cloudlets deployed. If contents are cached, requests are immediately resolved by storage cloudlets alleviating the stress in near cells. This approach targets one of the problems presented by Sanaei et al. [26, 46]: "a large number of such helping nodes are needed to alleviate ever increasing wireless traffic", referring wall-conected Wi-Fi hotspots similar to our proposed solution. Second, requests are served by the storage cloudlets, minimizing context-specific contents requests on the Internet Exchange Points, saving money to MNOs. The concept of increasing the density of antennas in a specific crowded area instead of increasing their range is currently being explored by pCell [4], in a similar way to our caching solution in small crowded areas instead of large areas covered by antennas deployed by MNOs.

To the best of our knowledge, this is a new business model not approached until now and presents a win-win situation for every participant of the scenario. As said before, not only mobile device users and storage cloudlet owners benefit from our proposed solution, MNOs can benefit too. It is even possible that MNOs propose to share storage cloudlets' costs due to their own economic interests, storage cloudlets should be cheaper than wireless connection cells.

One of the current problems of mobile devices platforms is the heterogeneity, forcing developers to implement different versions of their applications for each of the existent mobile devices platforms such iPhone, Android, Windows Phone or Blackberry. The solution to this problem proposed by Sanaei et al. [4] is developing web applications fully independent of any platform and has been the path followed by latest released platforms such as Firefox OS or Tizen. Our proposed solution is specially useful applied to web application loads, because a lot of elements can be cached for different user of the same web application: templates, images, updates, etc.

4.2. Limited connectivity

The second suggested scenario shows in a deeply way the possibilities of our proposed solution. Flexibility and easy deployment are key factors when

[4] http://www.artemis.com/pcell

the connectivity is limited by unusual conditions. Nowadays it might look like an unrealistic scenario but, as pointed by Satyanarayanan et al. in the "disaster relief" scenario [47], even in the future could appear scenarios where extreme conditions lead to a limited connectivity scenario. The disaster relief scenario brings up the possibility of a disaster disabling land and wireless WAN networks, leaving areas without any kind of connectivity.

Our proposed limited connectivity scenario is, to the best of our knowledge, new on the MCC research field. As seen in the last three years, mobile devices and cloud services have become a really powerful tool for social movements. Examples are the Arab Spring [48], Occupy Wall Street or protest movements in Spain, Greece, and Italy. The importance of these movements were awarded by the Time magazine as "person of the year" in 2011. In the most extreme cases Internet access was blocked by the government, for example in Egypt on January 2011 [49]. Another example of Internet blocks is the censorship in China, where connection blocks outside of the country have been frequently reported [50]. Even on tube lines of big developed cities in the present, some line segments do not have 3G connectivity. Our solution can be used to offer multimedia related contents to the mobile devices as it is currently done in TVs all around the stations and trains.

As shown in Figure 12, our solution aims to solve the aforementioned problems based on the flexibility and easy deployment. Using low-cost nodes, our solution is capable of deploying a cloud storage infrastructure for mobile devices, while sharing any kind of contents. There are multiple possible configurations mixing WLAN or WWAN. Connection between clients and cloudlets can be configured over WLAN or WWAN depending on the available networks. The connection between hierarchy levels can be configured over WLAN, LAN, WWAN or WAN connections depending on the currently active connections, being wired connections the best option. At the same level, it is not necessary to do any connection between servers. As presented on Subsection 3.4, our solution can be deployed in a huge variety of platforms, including low-power solutions. In case of a power outage, these nodes can be powered by diesel generators.

In the case of an emergency disabling of the Internet infrastructure in a huge area, storage cloudlets with any number of nodes can be deployed in hospitals, medical camps, or any other place where victims are congregated. These storage cloudlets could be used to share information about patients, condition, and missing people. In a similar way, in areas where connectivity is limited due to government restrictions, storage cloudlets can be used in

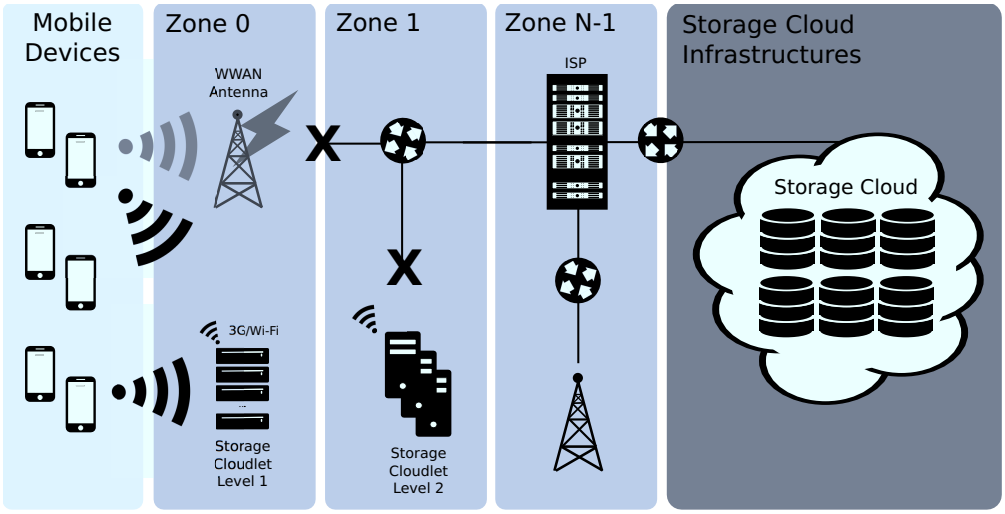

Figure 12: Limited connectivity scenario. WWAN connectivity in zone zero is down. Clients can connect through Wi-Fi (or 3G when available) to a local storage cloudlet infrastructure. The storage cloudlets can be extended to new zones when infrastructure starts being available, bypassing damaged paths.

congregation areas. Even citizens could share their own resources in their houses to add storage capacity to the cloud. As said before, in extreme conditions, there is no need of a special infrastructure, only enough nodes of our storage cloudlet in-a-box solution and a power source (can be a diesel generator or any other kind of portable and/or autonomous power source).

Porting Memcached to mobile platforms could lead to deploy the storage cloudlets over mobile devices, needing only a subscription system to configure the initial infrastructure and start offering the service. We have not implemented this approach because it is only desired in extreme cases due to the low performance, limited battery, and limited storage capacity of these devices. This approach could bring support to the storage layer of an ad-hoc mobile cloud instead of the proposed cloudlet-oriented solution.

5. Evaluation

To test the feasibility and performance of CoSMiC, we have setup a testbed in one of the scenarios mentioned above as a possible target: a university. The test consisted in a group of up to 40 students retrieving documents of different sizes from Amazon cloud. The students used smartphones to get

Table 1: Evaluation parameters.

Parameter	**Value**
Block size (request)	250 Bytes
Block size (response)	256 KBytes
3G network bandwidth	3.5 Mbps
WiFi network bandwidth	150 Mbps
LAN network bandwidth	1 Gbps
Memcached back-ends cache memory	2.0 GBytes
Amazon network bandwidth	20 Mbps
Amazon EBS Disk Performance	90 MBytes/sec
Thread per Memcached back-end	1

the documents using two alternative infrastructures: CoSMiC and 3G-based connection across MNOs.

Our university group is a real testbed that is significant for CoSMiC, as the students access data from the cloud, but following different interests, showing that the effect of the cache varies depending on the behavior of the users, affecting also latency and scalability of the system as shown below.

The experiments have been carried out in ARM-based nodes equipped with four Freescale i.MX6 cores (Cortex-A9) and 2 GBytes of RAM. The storage cloudlet is composed by four nodes, running CoSMiC storage cloudlets (based on Memcached version 1.4.13) with up to 2 GBytes of cache over a Linux kernel 3.0.35. In all the experiments, the files are divided in blocks of 256 KBytes. Clients run a synthetic benchmark in Android-based mobile devices. The final storage solution corresponded with an Amazon EC2 instance and data was stored in an EBS data volume. Table 1 summarizes the parameters used for the testbed. We compare CoSMiC with a baseline case which consists of a direct 3G-based connection to our Amazon instance. In all the graphs, we show the average result of five executions.

5.1. Cache effect

We have evaluated the effect of round-trip delay (RTT) of one read request for different hit ratios. Figure 13 plots the RTT of a read request for the baseline case composed by a 3G connection and CoSMiC. As the cache hit ratio increases, the RTT perceived by users reduces significantly given that read request are responded by the cloudlet network.

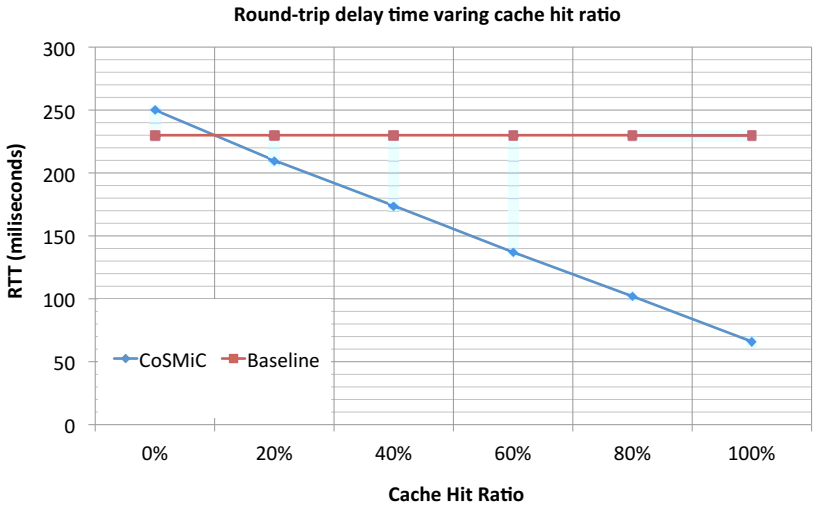

Figure 13: Round-trip delay time comparison for multiple CoSMiC hit ratios.

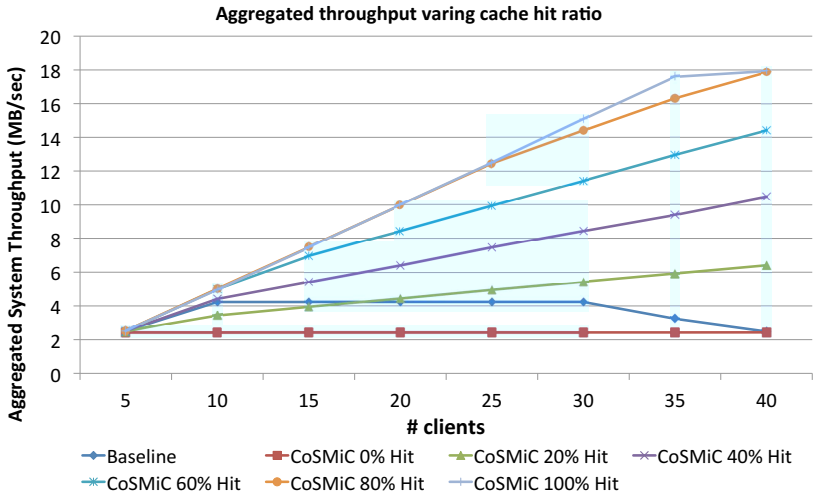

Figure 14: Aggregated system throughput for 5 up to 40 concurrent clients, which access a file of 512 KBytes.

5.2. System throughput

In this subsection, we evaluated CoSMiC in terms of the aggregated system throughput for an increasing number of clients. Figure 14 plots the aggregated throughput varing the number of clients. In this experiement, each client requests a file of 512 KBytes. We observe that for 35 and 40 concurrent clients, CoSMiC uses the maximum system thoughput, limited by

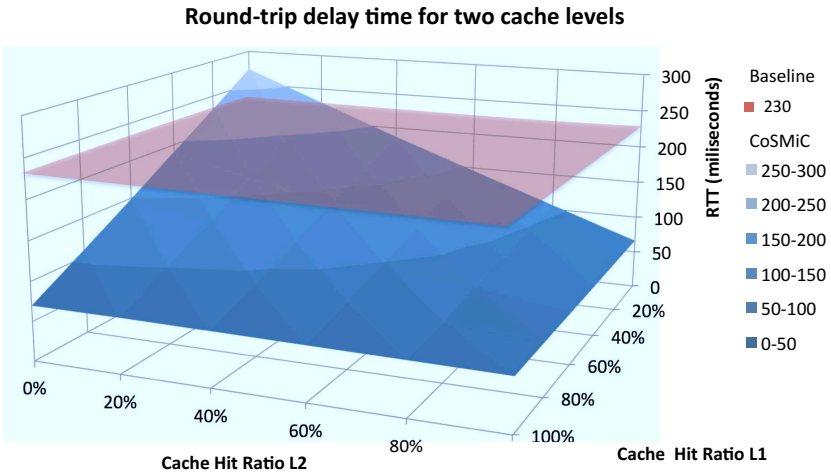

Figure 15: Round-trip delay time evaluation for two CoSMiC zones/levels. Cache L1 corresponds with the first CoSMiC level and L2 the second.

the Wi-Fi network bandwidth. We also observe that the 3G-based baseline case reduces its performance for 30 or more concurrent transfers. This case is specially unfair for crowed areas.

5.3. Multiple level cache

In this subsection, we evaluate the effects of multiple levels of CoSMiC in terms of RTT. A new CoSMiC back-end node has been included between the previous cloudlet and the WAN network. This node counts with an Intel Xeon E5640 with 64 GBytes of RAM and a gigabit network interface.

Figure 15 plots the RTT for two levels of CoSMIC and different hit ratios. As we observe, CoSMiC performs slower for low hit ratios, given the fact that more hops are involved in the data transfer. However, as hit ratios increase in both levels, CoSMiC outperforms the baseline solution in all the cases. An extra CoSMiC level alleviates the latency effects of multiple levels.

6. Conclusion

In this work we have presented, CoSMiC, a hierarchical cloudlet-based storage infrastructure for mobile clouds. CoSMiC offers flexibility and easy deployment, key features for MCC, specially for the proposed scenarios: public places and limited connectivity. Its generic design allows to deploy it in both heterogeneous hardware and network infrastructures, even for extreme

condition scenarios. As pointed by Sanaei et al. [26], one of the main problems of the MCC architectures is the heterogeneity due to the diversity of hardware architectures used by mobile devices in contrast to classical computers. The portability of our solution permits deploying it in any kind of node, due to the portability of Memcached, which even allows to use CoSMiC as a bridge between mobile devices and cloud services. Moreover, CoSMiC supports a complete single-path/single-copy file hierarchy, enhancing the possibilities of deployment in any kind of infrastructure. This hierarchy supports any number of levels and any number of cloudlet nodes in each level, providing decoupling between clients and servers and making our solution highly scalable and amenable for dynamic deployment of storage cloudlets in-a-box.

We have also presented a new business model involving MNOs, not used until now for mobile cloud storage services deployed in public places, and presented a win-win situation for every participant of the scenario. This model has also been proven valid when MCC is affected by extremely limited connectivity due to restricted connection situations instead of network failures, showing that our proposed solution is capable of addressing a lot of MCC problems related to storage, while benefiting to many participants in typical scenarios.

Evaluation made shows that CoSMiC can reduce the application's RTT significantly. CoSMiC permits to increase the system throughput when the number of clients increases.

Due to the possibilities offered by our solution, we have a good number of ideas for ongoing and future works. A possible enhancement would be to use the underutilized computational resources can be used to perform additional operations in order to optimize both storage and I/O activity, like implementing data encryption to improve security. This functionality could be implemented easily as an additional service in two different ways. First, by protecting data at the device level, at the cost of battery life, and second, ciphering data at the first cloudlet level making computation offloading (following the model proposed by Sanaei et al. [30]). Another enhancement could be data deduplication, that could be implemented prior to the persistent storage layer, reducing network traffic between layers and traffic to the cloud storage back-end. This approach could be complemented with replication of the most popular objects, as in CloudScale [51] and Scarlett [52], to avoid bursty loads by distributing queries to various nodes. Smart replication solutions could be provided by using CoSMiC data access statistics and the approach of pool nodes and shadow nodes proposed by Zhang et al. [53] for

weblets. Finally, we plan to extend CoSMiC to support other NoSQL-base distributed caches like Cassandra.

References

[1] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, R. Buyya, Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges, IEEE Communications Surveys and Tutorials 16 (1) (2014) 337–368.

[2] A. Khan, M. Othman, S. Madani, S. Khan, A survey of mobile cloud computing application models, IEEE Communications Surveys Tutorials 16 (1) (2014) 393–413.

[3] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, B. Li, Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications, Wireless Communications, IEEE 20 (3) (2013) 14–22.

[4] Z. Sanaei, S. Abolfazli, A. Gani, R. Buyya, Heterogeneity in mobile cloud computing: Taxonomy and open challenges, IEEE Communications Surveys Tutorials 10 (1) (2013) 369–392.

[5] D. Petcu, G. Macariu, S. Panica, C. Crciun, Portable cloud applications-from theory to practice, Future Generation Computer Systems 29 (6) (2013) 1417 – 1430, Including Special sections: High Performance Computing in the Cloud and Resource Discovery Mechanisms for P2P Systems.

[6] H.-c. Yang, A. Dasdan, R.-L. Hsiao, D. S. Parker, Map-reduce-merge: simplified relational data processing on large clusters, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, ACM, 2007, pp. 1029–1040.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, R. E. Gruber, Bigtable: A distributed storage system for structured data, ACM Trans. Comput. Syst. 26 (2) (2008) 4:1–4:26.

[8] S. Abolfazli, Z. Sanaei, A. Gani, F. Xia, L. T. Yang, Rich mobile applications: Genesis, taxonomy, and open issues, Journal of Network and Computer Applications 40 (0) (2014) 345–362.

[9] B. Fitzpatrick, Distributed caching with memcached, Linux J. 2004 (124) (2004) 5–.

[10] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM 51 (1) (2008) 107–113.

[11] T. White, Hadoop: The Definitive Guide, original Edition, O'Reilly Media, 2009.

[12] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: Elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys '11, ACM, New York, NY, USA, 2011, pp. 301–314.

[13] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: Making Smartphones Last Longer with Code Offload, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys '10, ACM, New York, NY, USA, 2010, pp. 49–62.

[14] E. E. Marinelli, Hyrax: Cloud Computing on Mobile Devices using MapReduce, Ph.D. thesis, School of Computer Science Carnegie Mellon University (Sep. 2009).

[15] R. Panta, R. Jana, F. Cheng, Y. Chen, V. Vaishampayan, Phoenix: Storage using an autonomous mobile infrastructure, IEEE Transactions on Parallel and Distributed Systems 24 (9) (2013) 1863–1873.

[16] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, A. Pras, Inside dropbox: Understanding personal cloud storage services, in: Proceedings of the 2012 ACM Conference on Internet Measurement Conference, IMC '12, ACM, New York, NY, USA, 2012, pp. 481–494.

[17] M. R. Palankar, A. Iamnitchi, M. Ripeanu, S. Garfinkel, Amazon s3 for science grids: a viable solution?, in: Proceedings of the 2008 international workshop on Data-aware distributed computing, ACM, 2008, pp. 55–64.

[18] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, J. Haridas, C. Uddaraju, H. Khatri, A. Edwards, V. Bedekar, S. Mainali, R. Abbasi, A. Agarwal, M. F. u.

Haq, M. I. u. Haq, D. Bhardwaj, S. Dayanand, A. Adusumilli, M. McNett, S. Sankaran, K. Manivannan, L. Rigas, Windows azure storage: A highly available cloud storage service with strong consistency, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP '11, ACM, New York, NY, USA, 2011, pp. 143–157.

[19] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, C. Maltzahn, Ceph: A Scalable, High-performance Distributed File System, in: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, OSDI '06, USENIX Association, Berkeley, CA, USA, 2006, pp. 307–320.

[20] S. A. Weil, K. T. Pollack, S. A. Brandt, E. L. Miller, Dynamic metadata management for petabyte-scale file systems, in: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, SC '04, IEEE Computer Society, Washington, DC, USA, 2004.

[21] S. Ghemawat, H. Gobioff, S.-T. Leung, The google file system, SIGOPS Oper. Syst. Rev. 37 (5) (2003) 29–43.

[22] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for vm-based cloudlets in mobile computing, IEEE Pervasive Computing 8 (4) (2009) 14–23.

[23] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, etime: Energy-efficient transmission between cloud and mobile devices, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 195–199.

[24] S. Sakr, A. Liu, D. Batista, M. Alomari, A Survey of Large Scale Data Management Approaches in Cloud Environments, IEEE Communications Surveys and Tutorials 13 (3) (2011) 311–336.

[25] CISCO, The Future of Hotspots: Making Wi-Fi as Secure and Easy to Use as Cellular (2012).
URL `http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns673/white_paper_c11649337.html`

[26] Z. Sanaei, S. Abolfazli, A. Gani, M. Shiraz, SAMI: Service-Based Arbitrated Multi-Tier Infrastructure for Mobile Cloud Computing, in: 1st IEEE International Conference on Communications in China Workshops (ICCC), 2012, pp. 14–19.

[27] S. Abolfazli, Z. Sanaei, M. Shiraz, A. Gani, MOMCC: market-oriented architecture for mobile cloud computing based on service oriented architecture, in: 1st IEEE International Conference on Communications in China Workshops (ICCC), IEEE, 2012, pp. 8–13.

[28] G. Huerta-Canepa, D. Lee, A virtual cloud computing provider for mobile devices, in: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond, MCS '10, ACM, New York, NY, USA, 2010, pp. 6:1–6:5.

[29] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, W. Heinzelman, Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture, in: 2012 IEEE Symposium on Computers and Communications (ISCC), 2012, pp. 59–66.

[30] Z. Sanaei, S. Abolfazli, A. Gani, F. Xiam, Hybrid pervasive mobile cloud computing: Toward enhancing invisibility, INFORMATION 16 (11) (2013) 8145–8156.

[31] Y. Xu, S. Mao, A survey of mobile cloud computing for rich media applications, Wireless Communications, IEEE 20 (3) (2013) 46–53.

[32] V. Sacramento, M. Endler, H. K. Rubinsztejn, L. dos S. Lima, K. M. Gonalves, F. N. Nascimento, G. A. Bueno, MoCA: A Middleware for Developing Collaborative Applications for Mobile Users, IEEE Distributed Systems Online 5 (10).

[33] P. Bahl, V. Padmanabhan, RADAR: an in-building RF-based user location and tracking system, in: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 2, 2000, pp. 775–784 vol.2.

[34] A. P. Miettinen, J. K. Nurminen, Energy Efficiency of Mobile Clients in Cloud Computing, in: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, Berkeley, CA, USA, 2010.

[35] N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy consumption in mobile phones: A measurement study and implications for network applications, in: Proceedings of the 9th ACM SIGCOMM

Conference on Internet Measurement Conference, IMC '09, ACM, New York, NY, USA, 2009, pp. 280–293.

[36] M. C. Brown, Getting Started with Couchbase Server, O'Reilly Media, Inc., 2012.

[37] T. Macedo, F. Oliveira, Redis Cookbook, O'Reilly Media, Sebastopol, 2011.

[38] A. Lakshman, P. Malik, Cassandra: a decentralized structured storage system, ACM SIGOPS Operating Systems Review 44 (2) (2010) 35–40.

[39] J. Zhang, G. Wu, X. Hu, X. Wu, A distributed cache for hadoop distributed file system in real-time cloud services, in: 2012 ACM/IEEE 13th International Conference on Grid Computing (GRID '12), 2012, pp. 12–21.

[40] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using wikipedia-based explicit semantic analysis., in: IJCAI, Vol. 7, 2007, pp. 1606–1611.

[41] A. Wolfe Gordon, P. Lu, Low-Latency Caching for Cloud-Based Web Applications, in: Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB '11), Athens, Greece, 2011.

[42] C.-M. Wang, C.-C. Huang, H.-M. Liang, Asdf: An autonomous and scalable distributed file system, in: Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID '11, IEEE Computer Society, Washington, DC, USA, 2011, pp. 485–493.

[43] L. Marques, C. J. Costa, Secure deduplication on mobile devices, in: Proceedings of the 2011 Workshop on Open Source and Design of Communication, OSDOC '11, ACM, New York, NY, USA, 2011, pp. 19–26.

[44] J. Dongarra, P. Beckman, T. Moore, Aerts, The International Exascale Software Project roadmap, Internatinal Journal in High Performance Computing with Applications 25 (1) (2011) 3–60.

[45] F. Isaila, J. G. Blas, J. Carretero, R. Latham, R. Ross, Design and evaluation of multiple-level data staging for blue gene systems, IEEE

Transactions on Parallel and Distributed Systems 22 (6) (2011) 946–959.

[46] P. Li, Y. Fang, On the throughput capacity of heterogeneous wireless networks, IEEE Transactions on Mobile Computing 11 (12) (2012) 2073–2086.

[47] M. Satyanarayanan, Mobile computing: The next decade, in: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing, MCS '10, ACM, New York, NY, USA, 2010, pp. 5:1–5:6.

[48] E. Stepanova, The role of information communication technologies in the arab spring, Implications beyond the Region. Washington, DC: George Washington University (PONARS Eurasia Policy Memo no. 159).

[49] BBC News, Egypt severs internet connection amid growing unrest (2011).
URL http://www.bbc.co.uk/news/technology-12306041

[50] C. Thompson, Google's China Problem (and China's Google Problem), New York Times.
URL http://www.nytimes.com/2006/04/23/magazine/23google.html

[51] Z. Shen, S. Subbiah, X. Gu, J. Wilkes, Cloudscale: elastic resource scaling for multi-tenant cloud systems, in: Proceedings of the 2nd ACM Symposium on Cloud Computing, SOCC '11, ACM, New York, NY, USA, 2011, pp. 5:1–5:14.

[52] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, E. Harris, Scarlett: coping with skewed content popularity in mapreduce clusters, in: Proceedings of the sixth conference on Computer systems, EuroSys '11, ACM, New York, NY, USA, 2011, pp. 287–300.

[53] X. Zhang, A. Kunjithapatham, S. Jeong, S. Gibbs, Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing, Mobile Networks and Applications 16 (3) (2011) 270–284.