Noname manuscript No.
(will be inserted by the editor)

Generic Context Adaptation for Mobile Cloud Computing Environments

Gabriel Orsini · Dirk Bade · Winfried Lamersdorf

Received: 12 May 2017 / Accepted: XX XX 2017

Abstract Markets for mobile applications offer myriads of apps ranging from simple to quite demanding ones. The latter are on the rise since every new generation of smartphones is equipped with more resources (CPU, memory, bandwidth, energy) to even allow resource-demanding services like speech- or face recognition to be executed locally on a device. But compared to their stationary counterparts, mobile devices remain comparatively limited in terms of resources. Because of this, current approaches aim at extending mobile device capabilities with computation and storage resources offered by cloud services or other nearby devices. This paradigm, known as Mobile Cloud Computing (MCC), is challenged by the dynamically changing context of mobile devices, which developers are required to take into account to decide, e.g., which application parts are when to offload. To rise to such and similar challenges we introduce the concept of Generic Context Adaptation (GCA), a data mining process that facilitates the adaptation of (mobile) applications to their current and future context. Moreover, we evaluate our approach with real usage data provided by the Nokia Mobile Data Challenge (MDC) as well as with *CloudAware*, a context-adaptive mobile middleware for MCC that supports automated and context-aware self-adaptation techniques.

Keywords Mobile Cloud Computing · Mobile Edge Computing · Context Adaptation · Context Awareness

G. Orsini
Distributed Systems Group
Department of Computer Science
University of Hamburg
Vogt-Koelln-Str. 30
22527 Hamburg
Tel.: +49-40-42883-2140
Fax: +49-40-42883-2328
E-mail: orsini@informatik.uni-hamburg.de

D. Bade
E-mail: bade@informatik.uni-hamburg.de

W. Lamersdorf
E-mail: lamersd@informatik.uni-hamburg.de

1 Introduction

Throughout the last decade, mobile devices became constant companions allowing to access information and services anytime and anywhere. Due to the ongoing miniaturization and cost-reduction of constituents, these devices are nowadays powerful enough to support lots of our everyday routines. Along with that trend, vendors like Google, Apple, and Microsoft built up marketplaces to easily distribute any kind of application to further extend the possibilities of such devices. The users in turn got accustomed, demand for even more sophisticated applications and are willing to invest in even more feature-rich and powerful devices, leading to a positive spiral of supply and demand in terms of mobile computation power at hands.

The mentioned situation has led to an increasing demand for a system support that is able to exploit the potentials of spontaneous interaction and therefore needs to be able to dynamically adapt to the quickly and constantly changing context of mobile ad-hoc scenarios. At present, many of the proposed solutions provide only limited context awareness or just specific prediction capabilities that moreover often require configuration by the developers. Consequently, we present the concept of *Generic Context Adaptation* (GCA), where we combine context awareness features with machine learning to serve as a prediction engine for arbitrary context attributes in mobile environments. In this way,

more generic and flexible scenarios that go beyond than just adaptation to the current context but to a future context become possible. The contributions in this paper can be summarized as follows:

- The concept of the GCA process that is able to predict arbitrary context attributes to anticipate the quickly changing context of mobile applications.
- A simulation of the developed GCA process that is based on realistic context information provided by the Nokia Mobile Data Challenge (MDC) campaign.
- An evalulation of the GCA process in a Mobile Cloud Computing (MCC) offloading scenario for image processing.

The remainder of this paper is structured as follows: Section 2 introduces the foundations of context awareness and MCC. Afterwards, Section 3 describes typical application scenarios, whose general requirements are matched with the related work, presented in Section 4. Subsequently, the concept of GCA is presented and evaluated in Section 5. At the end, we summarize our findings and give prospects for future work in Section 6.

2 Background

In this section we will describe the concept of MCC and context awareness being the two main foundations for the remainder of our work.

2.1 Mobile Cloud Computing and related Paradigms

MCC tries to weaken the restrictions of mobile applications by offering centralized resources to augment mobile devices. According to (Dinh et al, 2011) it is defined as the integration of cloud computing into the mobile environment to overcome obstacles related to performance (e.g., battery life, storage and bandwidth), environment (e.g., heterogeneity, scalability and availability), and security (e.g., reliability and privacy). We agree with this definition, extending the consideration of environmental restrictions to the more general problem of context adaptation. An early definition mentioned in the context of MCC which especially covers the aspect of mobility is the term *cyber foraging*. Coined by Satyanarayanan in 2001 (Satyanarayanan, 2001) it is described as:

> *"...construed as 'living off the land', [...] The idea is to dynamically augment the computing resources of a wireless mobile computer by exploiting wired hardware infrastructure."*.

Introducing the concept of cloudlets as an intermediate layer between mobile devices and cloud resources, cyber foraging is expected to further improve latency and execution speed.

A similar, but more recent definition that is more focused on the edges of a network is the so-called *(Mobile) Edge Computing*. As an evolution to mostly centralized resource augmentation strategies like MCC, mobile edge computing (MEC) or simply egde computing moves further in terms of decentralization. MEC tries to move the major part of remote operations from central resources directly into the surrounding infrastructure and so includes the logical extremes of a network. To do so, it replicates parts of an application's business logic onto nearby devices. Coined in 2004 by Akamai (Davis et al, 2004) it was first used to describe the topology of their content delivery networks that was used to cache often-requested contents at the logical edges of the network, but today the definition is used to provide more complex services by using cloud computing principles in a pay-as-you-go manner. Popular examples providing resource augmentation to increase the mobile devices computation power are *MAUI* (Cuervo et al, 2010), *CloneCloud* (Chun et al, 2011) and *ThinkAir* (Kosta et al, 2012).

2.2 Context Adaptation

Mobile devices are continuously faced with changes in their physical as well as logical environment. This might open up new opportunities (e.g., access to new services), but also challenges the execution continuity of running applications (e.g., need for service rebindings). In order not to bother the user with a decreasing quality of service or possible re-configuration requirements while on the move, applications and services ideally adapt themselves to their current context, as defined by Salber et al. (Salber et al, 1999) as:

> *"Environmental information or context covers information that is part of an application's operating environment and that can be sensed by the application. This typically includes the location, identity, activity and state of people, groups and objects."*

If applications or services make use of such environmental information, collected by either physical or virtual sensors, they are called context-aware. According to Dey and Abowd this definition includes the ability of a system to provide relevant information and/or services to the user where relevancy depends on the user's task (Dey and Abowd, 1999). This process of information extraction for adaptation purposes can be as

trivial as receiving GPS coordinates to show the user's location on a map, or as complex as employing the process of knowledge discovery in databases (KDD) to the vast amount of sensor data available on current smartphones.

For example, in order to distinguish if the user is running, walking or sitting, this can include employment of various sensors of a smartphone, data selection (choosing the right sensors and adjusting their resolution to acquire the right amount of data), preprocessing (data cleansing and transformations), modelling (finding appropriate models and fitting them to the problem) and deployment (using an efficient implementation that performs well on mobile devices) (Yuan et al, 2014). More detailed specifications of context adaptation introduce further distinctions (Geihs, 2008), namely:

- parametric adaptation
- compositional adaptation
- anticipated or non-anticipated adaptation

Parametric adaptation describes the adjustment of an application's control flow initiated by changes in the context. This criterion is often associated with typical examples of context awareness like location-based recommendations for electronic tourist guides, which we assign to the domain of classical context-aware research. In contrast, compositional adaptation refers to the replacement of certain application components at runtime, e.g. exchange a TCP[1]-based communication adapter against a Bluetooth GATT[2] adapter once the communication channel changes. The third criterion, anticipation, describes the condition in which future alternatives of context states are known a priori, or, in the case of non-anticipation, can at the earliest be determined at runtime.

The last two criteria can be crucial ones for future MEC scenarios, because respective applications need to adapt the composition of their constituent components when offloading tasks into the network. And for the developers it is hardly possible to foresee all imaginable future context states of a highly dynamic environment that require such an offloading decision (Wei and Chan, 2013). We therefore consider these kinds of adaptations the main requirement for resource augmentation in environments with intermittent connectivity like opportunistic computing. Hence, we describe context adaptation for MEC as:

> *"The ability of an MEC application to react to current and future connectivity states and re-evaluate the non-anticipated deployment strategy along with the available resources accordingly by using compositional adaptation."*

Recent MEC solutions consider context adaptation only as a minor factor to allow offloading parts of the computation to a surrogate, while we consider it the essential criterion to enhance the user experience, which will be reflected upon in Section 5.

3 Application Characteristics

To further illustrate the idea of MCC, three application scenarios are exemplified in the following. Common to this scenarios is the fact, that their distribution into the infrastructure is not static, but dependent on the context which is why they were chosen and, in a generalized manner, will serve as a base for the evaluation presented in Section 5.

Machine Learning as a Service: If an application is able to forecast a user's future activities, it can better adapt its behavior to the current needs, e.g. defer tasks until appropriate resources are available. To do so, an application must first learn about the user's behavior and daily routines. Therefore, context data must be collected continuously and be used to train prediction models. This not only involves quite huge amounts of data, but is also a computational intensive task. Hence, the context data can be uploaded from the mobile device into the cloud where prediction models are built or updated and only estimates of short term activity predictions are returned to the application afterwards.

Realtime Audio Fingerprinting: Modern smartphones are equipped with capabilities that allow their use as small media libraries containing huge amounts of music and video files. Acoustic fingerprinting is one popular method to identify songs using only short (and possibly noisy) fragments thereof. Suppose you are in a club, a really nice song is playing and you want to know what song that is. You record a few seconds and calculate an acoustic fingerprint which is broadcasted thereafter to all other smartphones nearby. On these devices the fingerprint is compared with the music files in the local media library and if a match is found, the file or at least the meta-data is sent back to you.

Sensor Data Analysis: Myriads of sensors have already been woven into our everyday live. Often, these are used to monitor certain conditions in the surrounding and communicate their observations to some kind of base station which resides at the edge of the sensor network and acts as a gateway in order to e.g. further

[1] Transmission Control Protocol, core protocol for internet-based communication

[2] Generic Attribute Profile, contains common Bluetooth operations and framework

disseminate and finally analyze the data. Typical scenarios include CCTV surveillance, environmental monitoring to detect wildfires, tsunamis, earthquakes, etc. as well as crowd-sourced weather information. If all the raw observation data (e.g., the live video stream) would be transferred to some central point of processing the required amount of bandwidth and central processing capabilities would be extremely high. One solution is to reduce the amount of data as early as possible (i.e. at the edge, see Figure 1) by using filters, aggregators or some kind of analysis techniques to infer higher-level information. For example, instead of transmitting video streams from hundreds of surveillance cameras to a central operating center, edge nodes could already perform a face recognition and only forward relevant information. This way, the processing load is shifted to the edge right where the data originates from and valuable bandwidth in the rest of the network is saved.

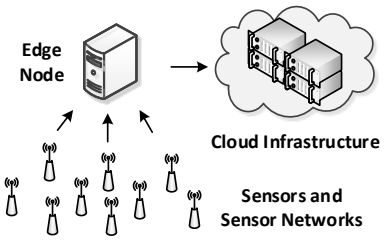

Fig. 1 Application Scenario: Edge Processing of Sensor Data

4 Related Work

From the middleware perspective, research on context-aware software has mostly addressed mechanisms that best support dynamic adaptation for specific use-cases. Early approaches in the field of self-adaptation were focused on the integration of context handling and the applications' business logic, whereas current solutions try to separate business logic from the context awareness features, to allow both parts of the implementation to be handled and replaced separately. Apart from some early approaches that explicitly handle the context some modularized approaches like the *Mobile Gaia* (Chetan et al, 2005) have emerged, which provided raw sensor data to its applications.

Notable current approaches from the domain of context-aware mobile middleware have been presented by Preuveneers and Barbers (Preuveneers and Berbers, 2007). They present a context-aware middleware for mobile devices that is component-based and self-adaptive. Nevertheless, there are no program-ming-level concepts that ease the development. Another prominent approach that aims at providing a self-adaptive middleware is presented in *MADAM* (Mikalsen et al, 2006), and later in *MUSIC* (Rouvoy et al, 2009). Here, applications are assembled by a component composition process and context awareness is achieved by exchanging the components' implementations with others having the same functional behavior. Still, adaptation rules need to be defined by the developer (e.g. using annotations). Further well-known solutions include *PACE* (Henricksen et al, 2005), *SOCAM* (Gu et al, 2004) and *CAMPUS* (Wei and Chan, 2013) whereas the last one represents the most current approach of an automatized reasoning by using the applications' context.

However, it can be concluded that in most context-aware systems, the adaptation logic of an application is implemented at the time of development. Such an approach has limited flexibility and poses a high burden for developers as it is almost impossible to foresee all conceivable context states, especially in mobile environments with a quickly changing context.

Summarizing the previous findings, we conclude that several solutions have been proposed to contribute to the fields of context adaptation. However, there is no ready-to-use solution, as none of the current solutions is able to provide context adaptation capabilities on a broad range of context attributes. As a consequence, we present the concept of Generic Context Adaptation, a holistic approach to tackle the challenges of both domains: computation offloading and context adaptation through a generic context adaptation process.

5 Generic Context Adaptation

In January 2009, the Nokia Research Center Lausanne (NRC), the Idiap Research Institute, and the EPFL initiated the creation of a large-scale mobile data research. This included the design and implementation of the Lausanne Data Collection Campaign (LDCC), an effort to collect sensor data from smartphones created by almost 200 volunteers in the Lake Geneva region over 18 months (Laurila et al, 2013). To our knowledge it is the largest dataset that contains information about mobile devices as well as application usage statistics, which is why we chose the Nokia MDC dataset to derive the following information as an input to a simulation of the prediction performance for several context attributes as well as an evaluation of a previously presented MCC-offloading solution (Orsini et al, 2015):

- GSM/WiFi/Bluetooth state (on/off), discovered MAC addresses and GSM cells, signal strength of WiFi as well as GSM cells, extended with our own mea-

surements to get an assumption about the available bandwidth.
- General information about the mobile device itself: time since the last user interaction, silent mode switch status, charging state, battery level, free memory.
- Date, time, location, calendar events, average and predicted remaining duration of stay at the current location.
- Reasoned attributes: remaining duration of stay at the same WiFi access point or GSM cell, user is at home/work, traveling, moving, resting.
- Application usage data: The applications the user interacts with and the specific screens of these applications.

5.1 Generic Context Adaptation Process

The previously presented data serves as input to our data mining process called *Generic Context Adaptation* (GCA) that is tailored to forecast any of the mentioned attributes by predicting it using the remaining provided attributes. More formally, GCA represents a data mining process that provides three types of predictions:

- A binary classifier (e.g. for predicting the future availability of a WiFi),
- a multi-class classifier (e.g. to predict a bandwidth range) and
- a regression mode to forecast real-valued attributes like the execution time of an offloaded task.

This way, historical data that has been collected by the mobile device can be used to forecast a future value of a certain context attribute. To perform such a prediction the GCA process needs to be provided with the historical input data. It is furthermore required to choose the context attribute to be predicted and the forecast horizons (the number of time intervals to look ahead into the future) to initiate the learning phase. The resulting predictive model can then directly be used to forecast the chosen context attribute. The data used in the GCA process is mainly the same as in the MDC and is just converted to a representation that is suitable to be presented to machine learning algorithms.

5.1.1 General design decisions

To be used together with a mobile middleware like CloudAware (Orsini et al, 2015), the GCA process is designed to be executed on a mobile device. As the preprocessing is performed in SQL and the application of the data mining models is performed in Java, the process is able to run on any (mobile) device that provides a Java virtual machine. Nevertheless, the training of the data mining models is computationally intensive and is currently intended to run on more powerful cloud resources, whereupon the trained model is transferred to the mobile device, to be used for predictions. We currently use an interval of five minutes to collect the required sensor data to not drain the battery too much. Furthermore, we reduced the available preprocessing- and machine learning algorithms to provide a lightweight implementation of the GCA process, that is able to run on mobile devices and hence is used for the evaluation presented in Section 5.2 and 5.3.

5.1.2 Preselection of suitable data mining algorithms

During the selection of suitable algorithms we focused on classifiers and regression techniques that have been successfully used in the domain of context reasoning. For example, the training of an artificial neural network is considered too heavyweight, both in terms of the runtime as well as in terms of the amount of data that would be required.

In Lim and Dey (2010) the authors have surveyed which classifiers are commonly used in scenarios where context data needs to be predicted. This evaluation was refined in Perera et al (2014) and the following classifiers were found to be the ones that where most often used:

- Decision Trees (15 %)
- Rule-based Systems (54 %)
- Hidden Markov Models (13 %)
- Naive Bayes (13 %)
- Support Vector Machines (4 %)
- k-Nearest Neighbor (2 %)

Together with the aforementioned decision for the interval-based modeling we selected the following algorithms and respective configurations for the reasoning in the GCA process:

- k-Nearest Neighbor (k-NN), where the value of k is chosen during the training
- M5 Decision Trees, using the CHAID-algorithm
- Naive Bayes
- Logistic Regression, using a polynomial kernel
- Support Vector Machines (SVM) or Support Vector Regression (SVR), using an rbf kernel
- An adaptive boosting (AdaBoost) for each of the learners

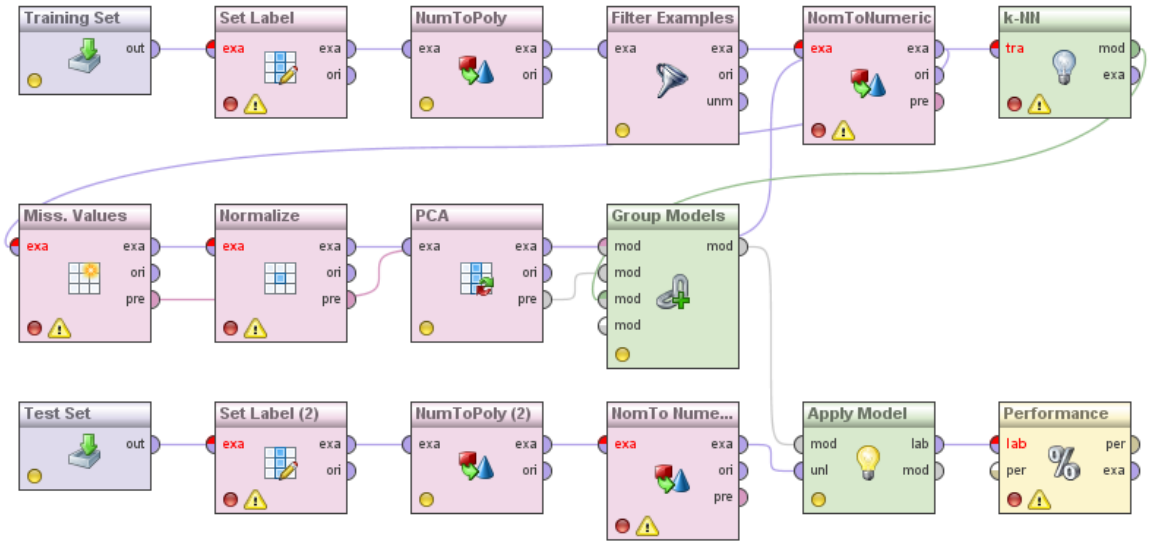

Fig. 2 GCA training and validation process

5.1.3 The GCA model selection process

To achieve both, a good prediction performance as well as a lightweight model (in terms of training time), the model selection process and the training process are separated in GCA. The model selection process aims at finding the right data mining algorithm, suitable hyperparameters (if applicable) and a suitable set of principal components for a certain context attribute and for an appropriate amount of data required for training. It is not intended to run on the mobile device as it performs the following tasks:

- The training data is filtered for missing attributes, which are then, if the attribute is at least complete to a minimum degree of 90%, filled with the attributes' average.
- A simple time series prediction for the prediction target is performed and added to the feature set.
- To reduce the runtime of subsequent steps a principal component analysis (PCA) is applied, where the first five principal components are used.
- Training and test sets are split via linear sampling while the shares are 80% training- and 20% test set.
- A preselection of the principal components is performed (adding features iteratively), as some of the chosen algorithms perform better with only the most relevant principal components.
- Where applicable, a parameter optimization is performed and the parameters are saved for the later training.
- Each of the learners is trained and its prediction accuracy, the hyperparameters, the input attributes used and the amount of data available for training are saved to a database.

5.1.4 The GCA training process

With the information from the model selection process about which learner has the best performance for a certain amount of data, the GCA training process now builds the requested prediction models. The training process performs the same preprocessing as the model selection process, but then selects the hyperparameters, input attributes and the information about which learner achieved the best performance from the database. This configuration is then trained with all available data, as exemplified in Figure 2. As a classification performance criterion we chose the F1-score, as the accuracy score is often uninformative in unbalanced sample sets. As a regression performance criterion we chose the mean absolute deviation (MAD) as the cost of a false prediction is considered equal for over- and for underestimation of the predicted target.

Furthermore, to face the obstacle that at the beginning of the simulation there is no training data, we use simple averages as long as we collected enough statistics (around 15 samples) before we switch to the output of the real prediction model. After an increase of the available data, the training set is further reduced to be more tailored to the specific prediction target, more precisely we are switching from weekly to weekday-specific models as we found that many predictions of context attributes benefit from this type of preprocessing. Figure 3 shows the weekly forecasting performance for the context attribute "charging state", showing that the described approach allows forecasts, even with no or just a small amount of training data available.

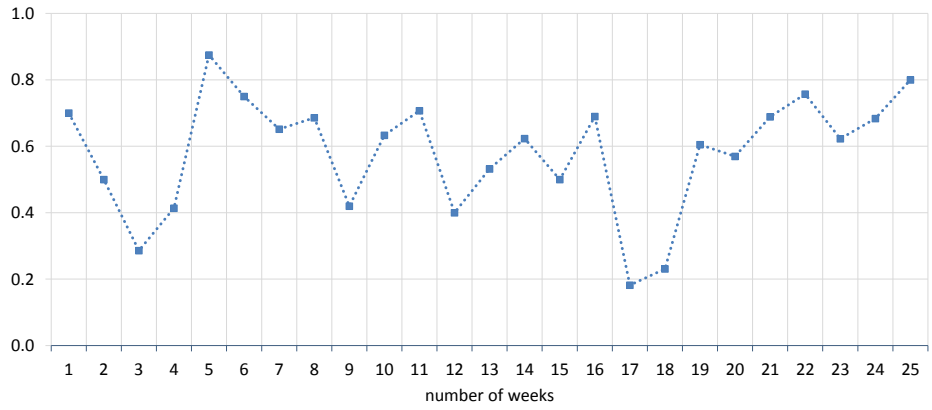

Fig. 3 F1 score for weekly forecasts

5.1.5 Learnings and Results

While designing the GCA process we chose to carry out exploratory data analysis on the MDC dataset to find relevant correlations between the prediction targets and the remaining features that served as inputs to the machine learning. Next, we combined some of these basic input variables into combined features to serve as an additional input to the machine learning. Furthermore, we normalized this set of features and carried out a PCA to reduce theamount of data to ensure a fast processing of the subsequent machine learning steps. This way, the developer is only required to provide the GCA process with a set of sensor data and by choosing a prediction target apart from this, the process is hands-free and does not require any further parameterization as, for example, hyperparameters of some of the learning algorithms are adjusted automatically.

Regarding the prediction horizon and the quality of the predictions our approach is able to discover both, periodic patterns that are expected in the next hours or days, as well as closer events that depend on current changes of the mobile devices context.

5.2 Evaluation with the Nokia MDC dataset

The top row diagrams in Figure 4 show that many of the discovered patterns in a mobile device's context follow time- or location-dependent patterns, hence adding an extensive modeling of time and location and even providing an estimation about when the user is expected to leave his current location highly improved the forecasting performance, shown in the bottom row of Figure 4. Here it can be seen that for binary classifications (left diagram) a good F1 score is achieved that slowly decreases for wider prediction horizons. The same holds for the regression performance shown in the right diagram.

5.3 Evaluation with the CloudAware Mobile Middleware

As a second evaluation scenario we chose an improved version of CloudAware (Orsini et al, 2015), an MCC/MEC offloading solution which differs from previous or similar approaches in the domain of MCC and context-adaptive mobile middleware by its primary design goal to support ad-hoc and short-time interaction with not only centralized resources, but also nearby devices. How CloudAware faces these restrictions and which general assumptions motivate specific design decisions has been described in previous work (Orsini et al, 2015) while the focus of this work are the results that have been achieved by employing the concept of GCA to improve CloudAware's offloading decision, i.e. when will offloading be beneficial considering the predicted future context.

The evaluation is performed by picking users that provided data for at least 18 months from the Nokia MDC dataset and by using the provided context attributes to design an event-based simulation that considers the connectivity (i.e. latency, intermittent connectivity and round trip times) of a mobile device as well as the battery drain, charging state, the limited computation power of a mobile device as well as the energy that is required to compute and to send or receive a specific amount of data via a specific interface (WiFi or GSM). The developed sample application applies different image filters to pictures taken with a smartphone's camera. It produces 20 different types of tasks. For each task the execution time, its variance, as well as input and output sizes (i.e. the amount of data that needs to be transferred in case of an offloading) have been measured and linked to real application events, that have been recorded in the MDC dataset.

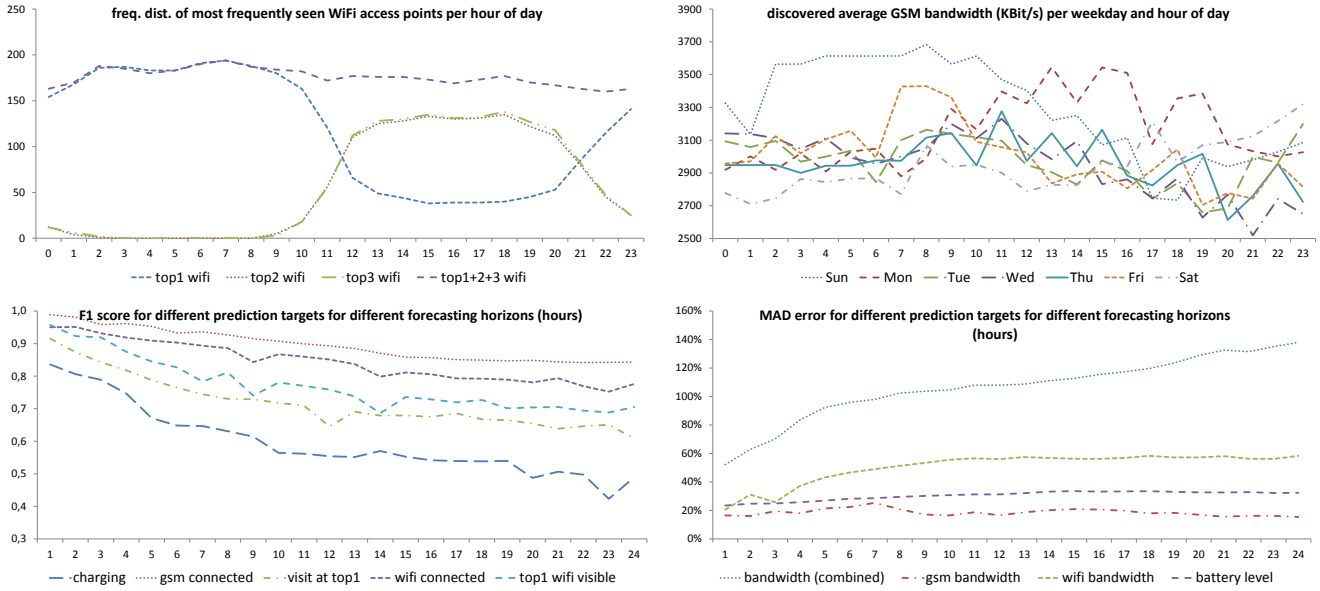

Fig. 4 Nokia MDC data characteristics (top) and evaluation of the prediction performance (bottom)

5.3.1 Simulation of the infrastructure

The used infrastructure consists of a Samsung Galaxy S5, representing the mobile device. The devices battery level found in the MDC dataset is considered a baseline, while the prototyped sample application generates an additional battery drain, therefore all tasks could only be successfully executed if the mobile device would be connected to an energy source at all times. The distinctive energy consumption of the sample application has been measured by measuring the amount of energy consumed, directly at the battery of the mobile device. Theoretically, similar measurements could be acquired by using a software like PowerTutor [3] but the corresponding power profile was unavailable. Furthermore, we consider the time and energy that is required to apply the trained GCA model as well as the energy that is required to communicate with the infrastructure via WiFi or GSM. Hence, it would be possible for developers to benchmark their application by using the described software-based approach.

The cloud server is simulated by measurements from and to a virtual server running at a german cloud service provider. To reflect the MEC scenario we furthermore assume cloudlets with the performance of an up-to-date desktop computer to be available at the three most frequently visited locations of each user. Furthermore, due to their limited resources, these cloudlets are assumed to be overloaded at certain times of the day, resulting in longer execution times or even timeouts that lead to unsuccessful offloading.

[3] http://ziyang.eecs.umich.edu/projects/powertutor/

5.3.2 Scheduling strategy and optimization goal

For each of the of the offloadable tasks, it is decided whether to offload this task to a cloud-server or a cloudlet by predicting its probability to be executed successfully, meaning that the result is returned to the mobile device. Therefore, we consider all tasks equally important and hence execute them by their invocation time, with the exception that already computed tasks may return to the mobile device with a higher priority. The offloading decision is furthermore influenced by choosing the alternative with the minimum invocation time and if equal the lowest energy consumption. We defined seven different scenarios which differ in the used offloading strategy:

Mobile: All tasks are being executed on the mobile device. The success rate in this scenario is limited to 86% as the additional energy that is consumed by the sample application produces additional periods of time in which the mobile device has run out of energy.

Cloud and Cloudlet: All tasks are being executed on the respective surrogate. In both scenarios, the success rate is limited, as sending tasks to and receiving tasks from the surrogate consumes additional energy by using the GSM or Wi-Fi interface of the mobile device.

CloudAware (CA): Tasks are being assigned to the cloud, the cloudlet or the mobile device based on the aforementioned scheduling strategy. Even if this strategy appears equal to the previous one, the higher bandwidth and lower latency to the Cloudlets (the cloud

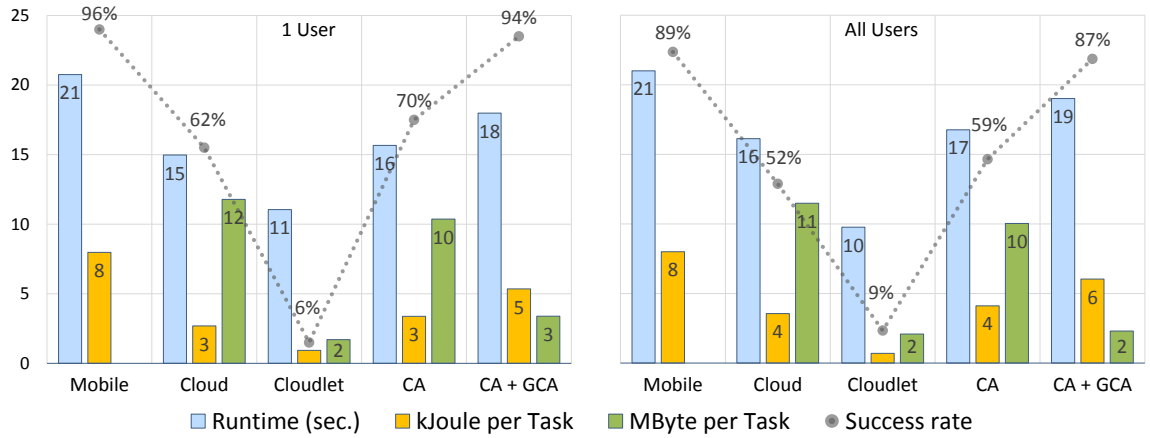

Fig. 5 CloudAware evaluation results for a single application

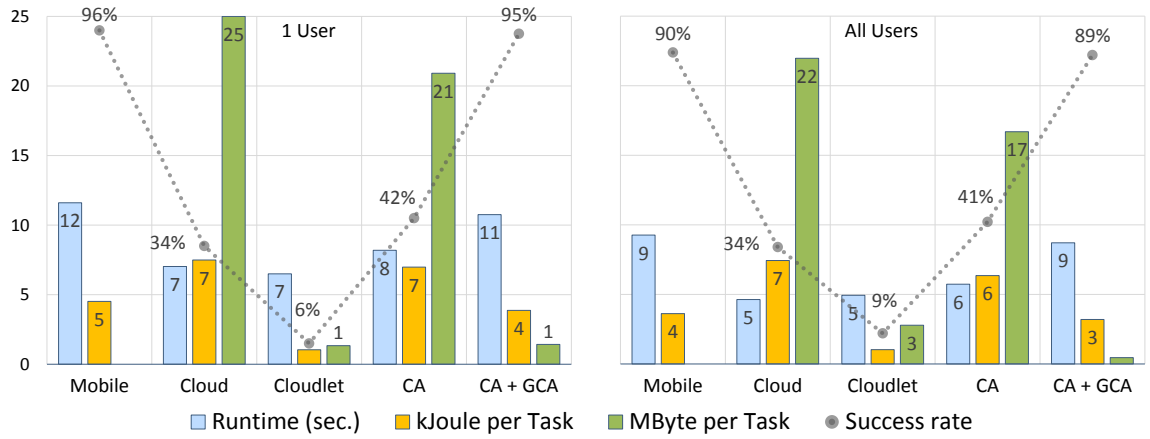

Fig. 6 CloudAware evaluation results for a mix of applications

servers are connected through a WAN-connection in our simulation) can result in higher execution speeds.

CloudAware supported by GCA (CA+GCA): Same as CloudAware scenario but the context manager of the CloudAware middleware provides predictions about the probability for a specific task to be successfully executed on a surrogate and furthermore predicts the expected time to execute. This information is used to decide about the offloading target.

5.3.3 Evaluation

For the evaluation of the trained models we perform a round-based training with a step-size of a week (we assume a weekly re-training of the models sufficient as it depicts a good balance between the availability of a significant amount of new data and the up-to-dateness of the prediction model).

Figure 5 reflects the results of our simulation for a single application, both for a single user as well as for an averaged statistic that includes different users and their specific usage patterns. It can be seen that the local execution (Mobile) for the single user, reflecting the baseline, allows an execution of 96% of the simulated tasks while achieving an average execution time of 21 seconds. The cloud-based execution results in a success rate of 62% while achieving an average execution time of 15 seconds. As expected, the execution time is lower but due to the limited network connectivity of the mobile device, the success rate is lower as well. The "Cloudlet" version of this scenario allows even lower execution speeds of 11 seconds, but the success rate is only 6% due to the limited reliability of the Cloudlet (see simulation setup).

Comparing these baselines to the standard offloading cases in mobile cloud computing (CA) and mobile edge computing (MEC Speedup) it can be seen that offloading based on a simple cost function to reduce the execution time is not always able to improve the service quality of mobile users as the success rates are not always higher, which is due to the case that intermittent connectivity sometimes prevents the successful completion of an offloading task.

Compared to this, it can be concluded that already the baseline version of CloudAware provides a speedup of 24% while executing 70% of the tasks. Using the GCA-supported version of CloudAware further increases the average energy consumption to 94% while it is the only scenario maintaining a lower execution times as well as energy consumption compared to the local execution. In terms of energy usage it can be concluded that offloading is not always beneficial if energy and bandwidth, next to speed, is included in the offloading decision. Here, the scenario "CA + GCA" is able to achieve the best trade-off between all evaluated scenarios as the consumed bandwidth and energy are both similar (bandwidth) or lower (energy), compared with the traditional MCC- or Cloud-based scenarios.

To proove the general applicability we designed an even more complex scenario, containing different applications and a higher variance, especially regarding execution times and the amount of state that is transferred. Figure 6 shows that the presented approach of GCA is capable to discover these patterns and allows this approach to be used for a broad range of application scenarios.

To summarize, it can be concluded that it is often not beneficial to just rely on past execution statistics and that the use of machine learning can provide decision support in context-dependent offloading tasks.

6 Conclusion

Augmenting the resources of mobile handheld devices can significantly improve their usability in areas such as health care, mobile learning, entertainment, and daily routine. Nevertheless, this type of augmentation can only be achieved by proper context adaptation and besides several efforts have already been directed towards context awareness in mobile middleware, the surveyed solutions are often domain-specific. Additionally scalability Xue and Deters (2016) and security AlShahwan et al (2016) are still areas of active research. Furthermore, the requirement for additional configuration prevents their immediate use in the domain of MEC where a quick, efficient and dynamic context-adaptation is necessary in order to even anticipate future context scenarios.

While there exists lots of work on designing context-aware mobile services that are able to predict specific context attributes it was the purpose of this paper to present a completely generic support for relevant mobile cloud interaction scenarios through context adaptation. Furthermore, it is often difficult to integrate context adaption features into applications as it is complex to achieve good context predictions for a multitude of configurations, devices and context states that cannot be foreseen by the developers.

The presented approach of Generic Context Adaption provides a hands-free solution for developers to allow context adaption without having to deal with context at all, but leaving the possibility to do so. Apart from basic services like link- and connectivity prediction the presented approach is able to predict arbitrary context attributes that are time- and location-dependent or correlate with other monitored context features. We evaluated the presented concept based on a real-world smartphone application and realistic context data provided by the Nokia MDC dataset and demonstrated that this scenario can benefit to a considerable degree from anticipation of future context states.

Acknowledgements Parts of the research in this paper used the MDC Database made available by Idiap Research Institute, Switzerland and owned by Nokia.

References

AlShahwan F, Faisal M, Ansa G (2016) Security framework for restful mobile cloud computing web services. J Ambient Intelligence and Humanized Computing 7(5):649–659

Chetan S, Al-Muhtadi J, Campbell R, Mickunas MD (2005) Mobile gaia: a middleware for ad-hoc pervasive computing. In: Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE, IEEE, pp 223–228

Chun BG, Ihm S, Maniatis P, Naik M, Patti A (2011) CloneCloud: elastic execution between mobile device and cloud. In: Proceedings of the 6. European Conference on Computer Systems, pp 301–314

Cuervo E, Balasubramanian A, Cho Dk, Wolman A, Saroiu S, Chandra R, Bahl P (2010) Maui: making smartphones last longer with code offload. In: Proceedings of the 8th international conference on Mobile systems, applications, and services, ACM, pp 49–62

Davis A, Parikh J, Weihl WE (2004) Edgecomputing: Extending enterprise applications to the edge of the internet. In: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters

Dey AK, Abowd GD (1999) Towards a better understanding of context and context-awareness. In: HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, Springer-Verlag, London, UK, pp 304–307

Dinh HT, Lee C, Niyato D, Wang P (2011) A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Comm and Mobile Computing 13(18):1587–1611

Geihs K (2008) Selbst-adaptive software. Informatik-Spektrum 31(2):133–145

Gu T, Pung HK, Zhang DQ (2004) A middleware for building context-aware mobile services. In: Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th, IEEE, vol 5

Henricksen K, Indulska J, McFadden T, Balasubramaniam S (2005) Middleware for distributed context-aware systems. In: On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, Springer, pp 846–863

Kosta S, Aucinas A, Hui P, Mortier R, Zhang X (2012) Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: INFOCOM, 2012 Proceedings IEEE, IEEE, pp 945–953

Laurila JK, Gatica-Perez D, Aad I, Blom J, Bornet O, Do, Trinh Minh Tri, Dousse O, Eberle J, Miettinen M (2013) From big smartphone data to worldwide research: The mobile data challenge. Pervasive and Mobile Computing 9(6):752–771

Lim BY, Dey AK (2010) Toolkit to support intelligibility in context-aware applications. In: Proceedings of the 12th ACM international conference on Ubiquitous computing, ACM, pp 13–22

Mikalsen M, Paspallis N, Floch J, Stav E, Papadopoulos GA, Chimaris A (2006) Distributed context management in a mobility and adaptation enabling middleware (madam). In: Proceedings of the 2006 ACM symposium on Applied computing, ACM, pp 733–734

Orsini G, Bade D, Lamerdorf W (2015) Computing at the mobile edge: Designing elastic android applications for computation offloading. In: 8th Joint IFIP Wireless and Mobile Networking Conference (WMNC), IEEE Explore Washington/DC, USA, p 8

Perera C, Zaslavsky A, Christen P, Georgakopoulos D (2014) Context aware computing for the internet of things: A survey. IEEE Communications Surveys & Tutorials 16(1):414–454

Preuveneers D, Berbers Y (2007) Towards context-aware and resource-driven self-adaptation for mobile handheld applications. In: Proceedings of the 2007 ACM symposium on Applied computing, ACM

Rouvoy R, Barone P, Ding Y, Eliassen F, Hallsteinsen S, Lorenzo J, Mamelli A, Scholz U (2009) Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments. In: Software engineering for self-adaptive systems, Springer

Salber D, Dey AK, Abowd GD (1999) The context toolkit: aiding the development of context-enabled applications. In: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, New York, NY, USA, CHI '99, pp 434–441

Satyanarayanan M (2001) Pervasive computing: vision and challenges. Personal Communications, IEEE 8(4):10–17

Wei EJ, Chan AT (2013) Campus: A middleware for automated context-aware adaptation decision making at run time. Pervasive and Mobile Computing 9(1):35–56

Xue Y, Deters R (2016) Towards horizontally scalable apps. Journal of Ambient Intelligence and Humanized Computing 7(4):465–473

Yuan B, Herbert J, Emamian Y (2014) Smartphone-based activity recognition using hybrid classifier. In: Proceedings of the 4th International Conference on Pervasive and Embedded Computing and Communication Systems