The Journal of Systems and Software 138 (2018) 37–51

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Empirical validation of cyber-foraging architectural tactics for surrogate provisioning

Fahimeh Alizadeh Moghaddam[a,b,*], Giuseppe Procaccianti[a], Grace A. Lewis[a], Patricia Lago[a]

[a] Software and Services Research Group, Vrije Universiteit Amsterdam, The Netherlands
[b] Systems and Network Engineering Research Group, University of Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 15 June 2016
Revised 14 November 2017
Accepted 21 November 2017
Available online 22 November 2017

Keywords:
Software engineering
Software architecture
Cyber-foraging
Energy efficiency
Resilience
Software sustainability

ABSTRACT

Background: Cyber-foraging architectural tactics are used to build mobile applications that leverage proximate, intermediate cloud surrogates for computation offload and data staging. Compared to direct access to cloud resources, the use of intermediate surrogates improves system qualities such as response time, energy efficiency, and resilience. However, the state-of-the-art mostly focuses on introducing new architectural tactics rather than quantitatively comparing the existing tactics, which can help software architects and software engineers with new insights on each tactic.
Aim: Our work aims at empirically evaluating the architectural tactics for surrogate provisioning, specifically with respect to resilience and energy efficiency.
Method: We follow a systematic experimentation framework to collect relevant data on Static Surrogate Provisioning and Dynamic Surrogate Provisioning tactics. Our experimentation approach can be reused for validation of other cyber-foraging tactics. We perform statistical analysis to support our hypotheses, as compared to baseline measurements with no cyber-foraging tactics deployed.
Results: Our findings show that Static Surrogate Provisioning tactics provide higher resilience than Dynamic Surrogate Provisioning tactics for runtime environmental changes. Both surrogate provisioning tactics perform with no significant difference with respect to their energy efficiency. We observe that the overhead of the runtime optimization algorithm is similar for both tactic types.
Conclusions: The presented quantitative evidence on the impact of different tactics empowers software architects and software engineers with the ability to make more conscious design decisions. This contribution, as a starting point, emphasizes the use of quantifiable metrics to make better-informed trade-offs between desired quality attributes. Our next step is to focus on the impact of runtime programmable infrastructure on the quality of cyber-foraging systems.

1. Introduction

In 2014, the number of mobile users exceeded the number of desktop users globally, which was about 1.7 billion users (Bosomworth, 2015). Consequently, many computation tasks are migrated to handheld devices as mobile apps. Statistics provided by "The Statistics Portal" forecast approximately 269 billion mobile app downloads for 2017, which is around 20% more than the previous year (Statistica., 2013). Although handheld devices are often selected as the main target for consumers and app developers, they are still limited in resources in terms of computational power and battery life.

The importance of extended device battery life has motivated software architects to introduce *Mobile Cloud Computing* solutions, in which the cloud takes charge of compute- and data-intensive tasks. Although these solutions significantly help to address resource limitations, a number of prerequisites need to be met. For example, a reliable Internet connection must exist between the handheld device and the cloud, which is not necessarily guaranteed in resource-scarce environments. Resource-scarce environments usually lack stable environmental conditions. *Cyber-foraging* has been introduced to enable resource-limited devices to benefit from available external resources in such environments with dynamic conditions.

A number of cyber-foraging tactics have been identified and categorized in Lewis et al. (2014; 2016) to help software architects select the best tactics to meet system requirements. In this

* Corresponding author at: De Boelelaan 1081a, Vrije Universiteit Amsterdam, The Netherlands.
E-mail addresses: f.alizadehmoghaddam@vu.nl (F. Alizadeh Moghaddam), g.procaccianti@vu.nl (G. Procaccianti), g.a.lewis@vu.nl (G.A. Lewis), p.lago@vu.nl (P. Lago).

https://doi.org/10.1016/j.jss.2017.11.047
0164-1212/

work we particularly focus on the "Surrogate Provisioning" tactics from an experimentation point of view. We study to what extent the cyber-foraging architectural tactics for surrogate provisioning impact system resilience and energy efficiency. Our findings guide software architects and software engineers to trace the impact of their design decisions with scientific insights concluded from quantifiable metrics. Our main contributions are:

- we provide a detailed description of cyber-foraging tactics for surrogate provisioning;
- we present a runtime optimization algorithm to support surrogate provisioning tactics and describe a proof-of-concept implementation;
- we show the systematic design and execution of our experimentation approach applied to surrogate provisioning, which can be reused for validating other cyber-foraging architectural tactics;
- we report on the execution and the results of our empirical experimentation aimed at quantifying the impact of the cyber-foraging tactics for surrogate provisioning on resilience and energy efficiency in a controlled environment;
- we provide an evaluation of the cyber-foraging tactics for surrogate provisioning, emphasizing trade-offs with respect to different system qualities.

This paper is organized as follows: Section 2 presents an overview of the cyber-foraging architectural tactics. Section 3 focuses on the surrogate provisioning tactics and how online optimization algorithms play a role in the system. In Section 4 we describe the scope of the experimentation using the goal, research questions, and metrics. Section 5 provides details of the planning steps from different perspectives such as context selection, variable selection, hypothesis formulation, subject selection, experiment design, and instrumentation. The steps taken to execute the experiments are explained in Section 6. In Sections 7 and 8 we present and discuss our results. Section 9 discusses the implications of our findings for software architecture. In Section 10 we describe the possible threats to validity and their mitigation. Section 11 discusses related work. Finally, Section 12 concludes the paper and outlines the research direction for our future work.

2. Background

Cyber-foraging is a mechanism that leverages cloud servers, or local servers called surrogates, to augment the computation and storage capabilities of resource-limited mobile devices while extending their battery life (Satyanarayanan, 2001). There are two main forms of cyber-foraging (Flinn, 2012; Lewis and Lago, 2015a; Sharifi et al., 2012). One is computation offload, which is the offload of expensive computation in order to extend battery life and increase computational power. The second is data staging to improve data transfers between mobile devices and the cloud by temporarily staging data in transit on intermediate, proximate nodes. While cyber-foraging can take place between mobile devices and cloud resources, our focus is on systems that use intermediate, proximate surrogates.

The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both (Bass et al., 2012). Software architectures are created because a system's qualities, expressed as functional and non-functional requirements, can be analyzed and predicted by studying its architecture.

One of the main challenges of building cyber-foraging systems is the dynamic nature of the environments that they operate in. For example, the connection to an external resource may not be available when needed or may become unavailable during a computation offload or data staging operation. As another example, multiple external resources may be available for a cyber-foraging system but not all have the required capabilities. Adding capabilities to deal with the dynamicity of the environment has to be balanced against resource consumption on the mobile device so as to not defeat the benefits of cyber-foraging. Being able to reason about the behavior of a cyber-foraging system in light of this uncertainty is key to meeting all its desired qualities, which is why software architectures are especially important for cyber-foraging systems.

Given the potential complexity of cyber-foraging systems, it would be of great value for software architects to have a set of reusable software architectures and design decisions that can guide the development of these types of systems, the rationale behind these decisions, and the external context/environment in which they were made; this is called *architectural knowledge* (Kruchten et al., 2006; Lago and Avgeriou, 2006). One way to capture architectural knowledge is in the form of *software architecture strategies*.

We define a *software architecture strategy* as the set of architectural design decisions that are made in a particular external context/environment to achieve particular system qualities. Software architecture strategies are codified as architectural tactics that can be reused in the development of software systems. We define *architectural tactics* as design decisions that influence the achievement of a system quality (i.e., quality attribute) (Bass et al., 2012).

Software architecture strategies for cyber-foraging systems are therefore the set of architectural design decisions, codified as reusable tactics, that can be used in the development of cyber-foraging systems to achieve particular system qualities such as resource optimization, fault tolerance, scalability and security, while conserving resources on the mobile device (Lewis, 2016).

In previous work we conducted a systematic literature review (SLR) on architectures for cyber-foraging systems (Lewis et al., 2014; Lewis and Lago, 2015a). The common design decisions present in the cyber-foraging systems identified in the SLR were codified into functional and non-functional architectural tactics (Lewis and Lago, 2015a; 2015b). Functional tactics are broad and basic in nature and correspond to the architectural elements that are necessary to meet cyber-foraging functional requirements. Non-functional tactics are more specific and correspond to architecture decisions made to promote certain quality attributes. Non-functional tactics have to be used in conjunction with functional tactics.

A cyber-foraging system must have at a minimum the following combination of functional tactics:

- Computation Offload and/or Data Staging tactics to provide cyber-foraging functionality.
- A Surrogate Provisioning tactic to provision a surrogate with the offloaded computation or data staging capabilities.
- A Surrogate Discovery tactic so that the mobile device can locate a surrogate at runtime.

Then, based on additional functional and non-functional requirements, such as fault tolerance, resource optimization, scalability/elasticity, and security, complementary tactics are selected.

The work in this paper focuses on surrogate provisioning tactics. We compare the different surrogate provisioning tactics from an architectural point of view with respect to their resilience and energy efficiency.

3. Surrogate provisioning tactics

3.1. Tactics description

To be able to use a surrogate for cyber-foraging, it has to be provisioned with the offloaded computation and/or the computational elements that implement the offloaded computation or en-