Distributed software development governed by security-constrained data flow

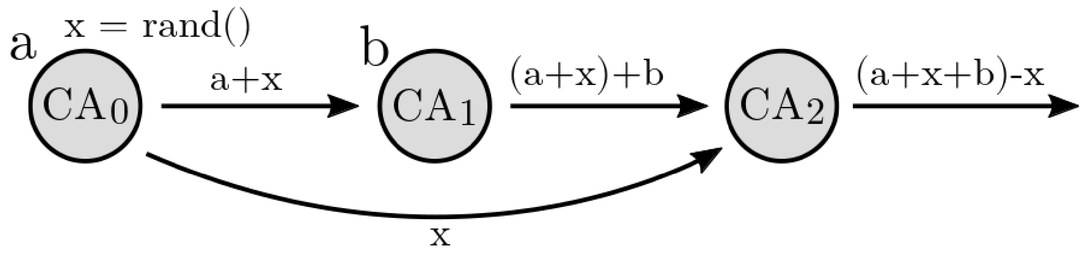
Rudolf Lovrenčić, mag. ing. comp.
(rudolf.lovrencic@fer.hr)
mentor: Assoc. Prof. Dejan Škvorc, PhD
University of Zagreb Faculty of Electrical Engineering and Computing

1. Introduction

Cloud computing is increasingly used for personal and business purposes, both as passive data storage and as a place for data processing, e.g. running applications. However, a certain number of organizations are unwilling or unable to take advantage of cloud computing due to trust issues. If the cloud is used not only for data storage, but also for data processing, then the data privacy can be violated not only in the repository if the data is not encrypted, but also during the processing. Encrypting the data before outsourcing them to the cloud increases their security, but at the same time prevents them from being processed in the cloud. This research aims to provide a paradigm for outsourcing both the data and the code to the cloud in a way that preserves data privacy, while still enabling their processing outside the organization.

2. Problem Description

To be able to distribute the program logic, protocols for performing computations without revealing the operands or the results of the operations to the participants must be introduced. In order to minimize performance degradation, the granulation of software fragmentation should be optimized based on the user-provided security requirements and the set of available hosts. The optimization process decides which operations are computed using distributed patterns.

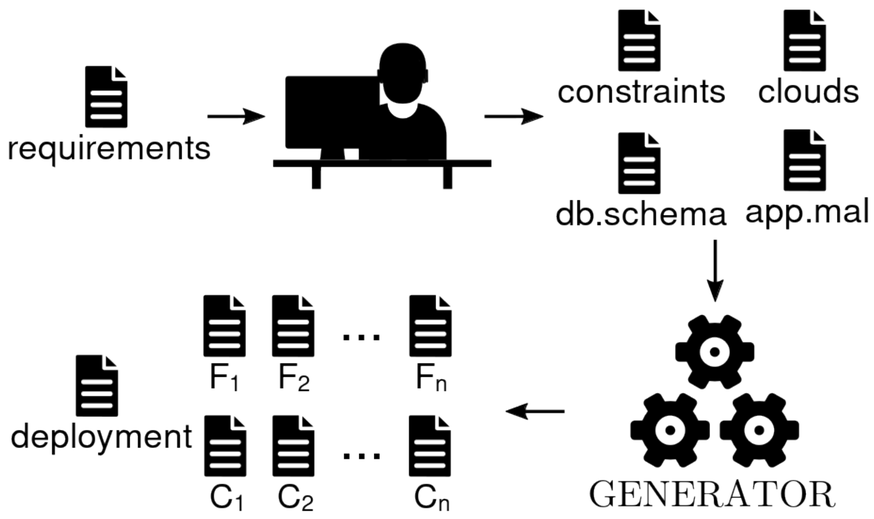


Distributed addition pattern in which no component learns both operands

3. Methodology

The generator translates the application from a monolithic form, which simplifies the development, into a distributed form with embedded security-constrained data flow optimized for the given deployment environment. Generation consists of three phases:

1) Parse the environment description, security constraints, dababase schema and program logic.
2) Determine the optimal mapping of data to the provided hosts.
3) Generate a distributed application in the target language.

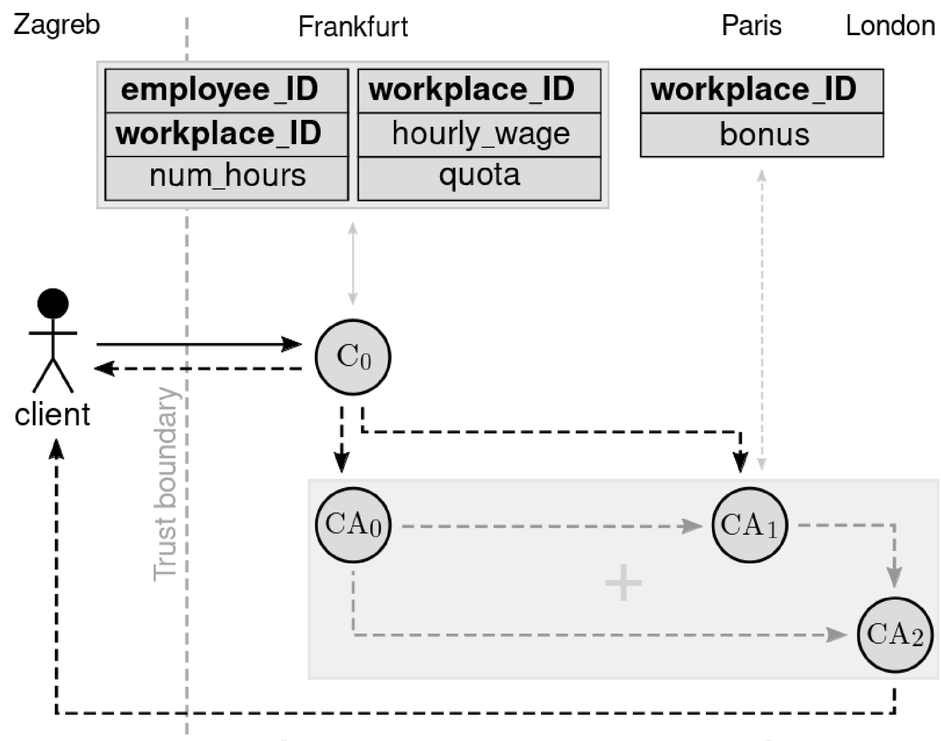


Development of a distributed application with security-constrained data flow

During the optimization procedure, the generator decides which operations should be performed using the distributed patterns and which should remain monolithic, ensuring a minimal performance penalty.

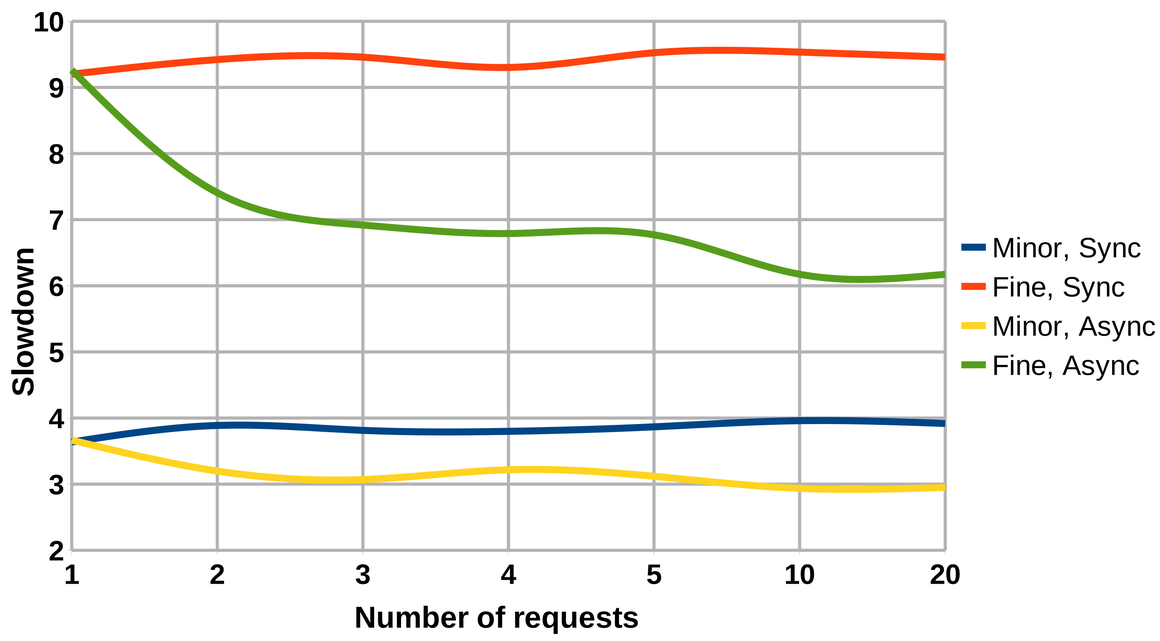
4. Results

Performance testing has been performed on a trivial application that calculates the payout of company's employees. Relative to the monolithic application, two additional variations of the application have been analyzed: **i)** *fine split,* where each piece of data is stored on a separate location, **ii)** *minor split,* where only the bonus is separated.



A "minor split" version of the demo application

Two versions of the client were also considered: **a)** *synchronous client* that waits for the response before sending another request, and **b)** *asynchronous client* that sends all requests immediately.



5. Conclusion

The performance analysis indicates that the paradigm is suitable for small-scale applications or critical parts of larger cloud systems. Results also imply that the performance impact is lower in applications dominated by computation because the introduced communication cost is relatively small. A tool for automatic generation of distributed logic based on the monolithic logic, environment description, and security constraints has been introduced. The tool reduces the complexity of development significantly and separates security requirements from the application logic.

6. Project Acknowledgement

This research is co-sponsored by the European Regional Development Fund through research grant KK.01.2.1.01.0109.