TECHNISCHE UNIVERSITÄT DRESDEN

cfaed CENTER FOR ADVANCING ELECTRONICS DRESDEN

Composable Operating Systems Group Barkhausen Institut

## Task Description for Grosser Beleg Thesis

for: **Lisza Zeidler**

Major: Diplom Informatik, 2018
Matriculation Nr.:

Title: **A Python Integration for Ohua**

Ohua is a parallelizing compiler that transforms sequential stateful applications into (micro-)service-based programs for micro-kernel-based operating systems and cloud infrastructures. A central point of Ohua's programming model is that it integrates into many existing imperative languages. However, the current implementation only delivers an integration for the Rust programming language. Rust is well-known in the systems programming community but an important target for Ohua is the cloud where other languages are favored.

One of the most popular programming languages used by developer for cloud application is Python. Python is generally interpreted as an untyped language. This deminishes one of the main benefits of Ohua: a type-safe compilation for a program that executes in a distributed fashion. Due to Python's popularity push in the recent years, gradual type support was added.

This thesis shall provide an integration of Python for the Ohua compiler that is meant to serve two goals. At first, it delivers the foundation for a future cloud backend. Secondly, the task focuses on providing a *multi-processing* backend. Python has a global interpreter lock and as such cannot provide parallel programming via shared memory. Developers have to fall back to programming with processes which is tedious and error-prone. The Ohua integration for Python developed in this thesis removes this burden of Python developers.

In particular, this "Grosser Beleg" Thesis shall include the following tasks. The student shall:

1. Become familiar with the integration infrastructure of Ohua and the "language-python"[1], the Haskell package for parsing and producing Python code
2. Define the Python subset that maps to Ohua's programming model of state threads.
3. Implement this mapping as a frontend in Ohua.
4. Implement a language backend for Ohua.
5. Implement the according architectural backend for the multiprocessing library of Python[2].
6. Evaluate the implementation using existing benchmarks.

| | |
|---|---|
| Advisor: | Dr. rer. nat. Andrés Goens, Dr.-Ing. Sebastian Ertel |
| 1. Examiner: | Prof. Dr.-Ing. Jeronimo Castrillon |
| 2. Examiner: | Dr.-Ing. Sebastian Ertel |

| | | | |
|---|---|---|---|
| Issued: | 22.07.2021 | Turn in by: | 9.12.2021 |

Prof. Dr.-Ing. Jeronimo Castrillon

Dr.-Ing. Sebastian Ertel

[1] https://hackage.haskell.org/package/language-python

[2] https://docs.python.org/3/library/multiprocessing.html