

Master Thesis Project Proposal (30hp):

Fault-Tolerant Skeleton Program Execution in Distributed Heterogeneous Parallel Systems

High-level parallel programming aims to abstract challenging aspects of parallel and heterogeneous systems for non-expert programmers. *Algorithmic skeletons* is an interface approach based on computational patterns, such as *map*, *reduce*, and stencil operations. These patterns can be instantiated by providing a custom operator ("user-function"), which is then applied to a supplied dataset in parallel according to the particular pattern semantics. Skeleton programming frameworks and libraries such as *SkePU* implement skeletons as C++ constructs and provide "back-ends" for parallelism in multi-core CPUs, GPU accelerators, and multi-node clusters. The skeletons are typically provided as libraries (e.g., in Spark) or, in the case of SkePU, as a framework with both library and a custom compiler toolchain. The SkePU library is implemented in modern C++ and involves template metaprogramming. SkePU (<https://skepu.github.io>) is a long-term open-source effort at PELAB, Linköping University. Recently, we have added a distributed stream-processing framework atop SkePU, called *SkePU-Streaming*. It provides a two-layer programming model where the top layer specifies the topology and metadata of a complex stream-processing pipeline, while the actual stream processing tasks are defined in plain SkePU in the lower layer.

So far, SkePU and SkePU-Streaming rely on fault tolerance mechanisms in the underlying hardware and system software layers, and do not implement any fault tolerance mechanisms on their own. This is good enough for many less critical or short-lived application scenarios where occasional failures are quickly resolved by re-executing the program elsewhere, but for use in a high-reliability scenario (e.g., real-time automotive or avionics applications) this is not sufficient.

Task This project will investigate ways to establish fault-tolerance in SkePU-Streaming program execution on top of unreliable hardware, both within a node and in multi-node scenarios. Different fault models (node failure, silent data corruption) and hardening techniques known from the literature (e.g., modular redundancy, checkpointing, data storage redundancy, ECC) shall be investigated and at least two of them prototypically implemented for SkePU and the overheads experimentally evaluated. We are particularly interested in (skeleton-specific?) configurability options of the fault-tolerance mechanisms that allow to adaptively explore cost-reliability trade-offs.

Prerequisites TDDD56 Multicore and GPU Programming (mandatory), TDDD38 Advanced Programming in C++ (mandatory), TDDD25 Distributed Systems (recommended), TDDE31 Big Data Analytics (recommended), TDDE65 Programming of parallel computers (recommended). Linux programming skills.

Contact Christoph Kessler, August Ernstsson, Sajad Khosravi (first.last@liu.se)

Open thesis projects: <https://www.ida.liu.se/~chrke55/exjobb/open-exjobb-projects.shtml>