

Master Thesis Project Proposal (30hp):

Load Balancing for Skeleton-Based Distributed Stream Processing

High-level parallel programming aims to abstract the complexities of parallel and heterogeneous systems away from non-expert programmers. Algorithmic skeletons are one such approach, based on computational patterns such as map, reduce, and stencil, each instantiated with a custom user function applied to a dataset in parallel according to the pattern's semantics. SkePU-Streaming is a distributed stream-processing framework developed atop SkePU, a high-level parallel programming framework that uses algorithmic skeletons. SkePU-Streaming employs a two-layer programming model: the upper layer specifies the topology and metadata of the stream-processing pipeline, while the lower layer defines individual processing tasks using SkePU constructs. SkePU implements skeletons as C++ constructs with back-ends targeting multi-core CPUs, GPU accelerators, and multi-node clusters. This framework is developed as a long-term open-source project at PELAB, Linköping University.

In streaming applications, variations in computational load, such as increased throughput requirements or changes in output quality, can result in load imbalance across system resources. Such imbalances may cause significant queuing delays or communication buffer overflows. An effective load-balancing mechanism must monitor system status, predict near-future load based on current conditions and trends, and determine when and how to migrate or replicate tasks to rebalance load on computational resources.

This project targets load balancing at two distinct levels. At the local level, each node balances its workload using available resources, such as hybrid execution across multiple SkePU back-ends, or by coordinating with neighboring nodes for locally optimal task mapping through low-cost, distributed, decentralized migration. At the global level, a central load balancer monitors the overall system status and makes global load-balancing decisions. Local balancing responds to short-term load fluctuations with minimal migration cost, while global balancing addresses longer-term, system-wide fluctuations that local decisions alone cannot resolve.

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

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