Motivation
Large scale parallel programs take a long time to develop and are hard to get right. Following common parallel design patterns like task pools and pipelines offers a way to reduce the structural complexity of the program. Still, a reasonable performance is not guaranteed, even when following well known design patterns, an issue only observable once the whole application is running on a large scale.
To circumvent this issue, we introduced a framework to estimate the throughput of a program following such a parallel design pattern by using the throughput of the used atomic functions and a composition operator. This way, a developer can model the atomic functions of their application single-threaded and estimate the performance of the complete program with a reasonable relative error.

Task
There already exist an implementation for the parallel design patterns task pool, pipeline, map, reduce, mapreduce and for the general design patterns of composition and iteration. The implementation and the associated operators are checked for correctness regarding the throughput of the application. The aim of this theses is to establish (a) new operators that work analogously to the already defined one, working on metrics like latency, number of threads, etc. and (b) to bring the operators in the style of a calculus, formulizing them precisely on the way.

- Introduce composition operators for latency
- Introduce composition operators for thread count
- Define a calculus that formulizes the operators as judgments

Requirements
- C++ (level of SPP is sufficient)
- Parallel Programming (level of MTC++ is sufficient)
- Understanding of a calculus (level of MoSeS is sufficient)

Contact
Fabian Czappa <czappa@cs.tu-darmstadt.de>

References