Parallel machines have limited communication capabilities and a large spread on memory access times. Therefore, the performance of parallel programs is constrained by the amount of data they move and the layout of the data. For good scalability, programs should be designed to minimize data movement on the machine in addition to having ample parallelism. But tuning a program to minimize communication on a new machine involves considerable effort in terms of understanding the arrangement of caches and connections on the machine. One way to avoid repeating this tedious task is to develop a portable analysis for the locality and parallelism of parallel programs with relevance to costs on a broad class of machines.

This thesis addresses the problem of portable analysis by designing a program-centric metric for locality and parallelism of nested-parallel programs – a metric based solely on the program structure without reference to machine parameters such as processors, caches and connections. We show that this metric is relevant by constructing a provably-good locality preserving scheduler that maps programs to parallel memory hierarchies, represented by a tree of caches. The scheduler also achieves good load balance. We prove performance bounds on the communication costs, running time and space requirements of this scheduler in terms of the program-centric analysis. This demonstrates the feasibility and practicality of this approach since parallel memory hierarchies are a good approximation for a broad class of machines.

Further, we illustrate some simple guidelines to build algorithms that are good according to our metrics. Using these guidelines, we construct optimal parallel algorithms for several common algorithmic problems. These algorithms also scale well in practice. Emphasizing the separation between algorithms and schedulers, we built a framework for an empirical comparison of schedulers. We engineered a practical variant of our scheduler for parallel memory hierarchies in this framework and demonstrated performance improvements over some state-of-the-art schedulers on a certain class of algorithms.