The multicore revolution in microprocessor design has resulted in a critical need for software to run efficiently and reliably across multiple cores if we are to continue the usual performance gains that are desired by industry and researchers. This work presents a multicore parallelism framework that exposes the discrete optimization problem of software parallelism. An increasingly important problem when parallelizing an algorithm in a network-on-a-chip environment is to hide the data flow latency. Our technique, which is currently implemented on the Cell Broadband Engine, is to have a runtime interpreter on each high performance computational unit which executes incoming bytecode instructions from a controlling computational unit. The bytecodes provide a more abstract method of expressing parallelism which is largely decoupled from the details of specific hardware. This frees us to focus on the algorithm we are trying to parallelize and use heuristic techniques to optimize the data flow, in particular to prevent latency.

This presentation has two aims: to define and explain the objective and constraints to experts in optimization but not processor design, and to give initial performance numbers for our heuristic techniques on benchmarks taken from linear algebra, magnetic resonance image reconstruction problems.