Network Partitioning in Scientific Simulations - A Case Study

Hélène C. Coullon and Rob H. Bisseling

High-level parallel programming, or implicit parallel programming is one of the most important research domains to bring the use of parallel architectures within easy reach for non computer scientists. The large domain of scientific computations has, a priori, no requirement limits for performance and parallelism, and is one of the primary domains that need implicit parallel programming solutions. Among all possible scientific computations, simulations based on *partial differential equations* (PDEs) are particularly time-consuming. In this context, the high-level parallel programming library SkelGIS has been implemented. SkelGIS [2,3] is an implicit parallelism, and a C++ header-only, library to solve mesh-based PDEs in parallel while preserving a sequential programming style.

Numerical methods to solve partial differential equations (PDEs) discretize both time and space to run a simulation on a computer. The discretization of the space domain is called a mesh. Parallelization of a simulation implies the need for a good load balancing of the mesh among the processors, and for a minimization of the communication volume during computations, which could be modeled as a graph partitioning problem. In some specific simulations, as for example in artery blood-flow or river water-flow simulations, a network is created to represent the domain with two different types of elements: nodes and edges. A network could be considered as a general graph where computations are carried out on both edges and nodes and where communications are needed from nodes to edges and from edges to nodes, possibly at different time steps. Thus, parallelization of such applications raises a specific graph partitioning problem, where both edges and nodes handle computations and communications at each time iteration.

Partitioning of simulations with several computation supersteps such as our network simulation can, in principle, be done by invoking a multi-constraint hypergraph partitioner [4] as PaToH for example. Our approach is different: we use the single-constraint partitioner Mondriaan [5] but make sure to satisfy both constraints of load balancing, while minimizing the communication. Two different methods have been explored: the first one, named *single-partitioning method*, is composed of two steps: (1) the communication superstep from nodes to edges is translated to a hypergraph partitioning problem [1], to distribute the edges, and (2) a heuristic is applied to distribute the nodes of the network, taking into account the distribution of the edges; the second one, named *double-partitioning method*, is decomposed in three steps: (1) the communication step from nodes to edges is translated to a hypergraph partitioning problem to distribute the edges, (2) the communication step from edges to nodes is translated to a hypergraph partitioning problem to distribute the nodes, and (3) a permutation problem is solved to match both distributions. We expect performance results for the poster presentation.

References

[1] Ü. V. Çatalyürek and C. Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE TPDS*, 10(7):673–693, 1999.

[2] H. Coullon, M.-H. Le, and S. Limet. Parallelization of Shallow-Water Equations with the Algorithmic Skeleton Library SkelGIS. In *ICCS*, pages 591–600, 2013.

[3] H. Coullon and S. Limet. Algorithmic skeleton library for scientific simulations: SkelGIS. In *HPCS*, pages 429–436, 2013.

[4] B. Uçar and C. Aykanat. Partitioning sparse matrices for parallel preconditioned iterative methods. *SIAM J. Sci. Comput.*, 29(4):1683–1709, 2007.

[5] B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Rev.*, 47(1):67–95, 2005.