New Approaches to Fault-Tolerant Computing

Anders Hansson, Gabriel Istrate, Christian Reidys, Allon G. Percus, CCS-3; Stefan Boettcher, Bruno T. Goncalves, Emory University; Shiva P. Kasiviswanathan, Carnegie Mellon University

Fault tolerance in a parallel and distributed computing architecture relies on rapidly matching processes with resources across a network. This problem, called load balancing, is largely unsolved at the scale needed to support routine petaflop computing. Load balancing is closely related to the discrete optimization problem of graph partitioning. Imagine that each process is represented by a node in a network and that communication between processes is represented by a link. Mathematically, the communication patterns of the processes form a directed acyclic graph, shown in Fig. 1. The goal in graph partitioning is to partition the graph into a certain number of subcomponents, with as few links as possible connecting them. When one imposes additional global constraints, such as the "balanced partition" requirement that all subcomponents be the same size, the problem rapidly becomes computationally intractable as the graph size increases.

Load-balancing problems may be studied under the following generative model: create a dependency graph by placing, independently and with some probability p, a link between each pair of nodes. Reaching a detailed understanding of the solutions to the balanced partitioning problem, even on simplified random instances of this sort, is a prerequisite to modeling appropriately structured graphs. We have studied characteristics of how these solutions "cluster" in solution space, for the case of partitioning the graph into two balanced subcomponents. It follows from standard random graph arguments that given n nodes, if $pn < 2 \log 2$, processes can almost always be partitioned perfectly without any links connecting the subcomponents. We have shown that in this case, given a straightforward definition of distance in solution space, all solutions lie within a single "cluster" where each solution is close to others [1].

This has significant implications for the analysis of graph partitioning using methods from statistical physics. The existence of a single cluster enables a key tool for algorithmic analysis, known as *replica symmetry,* to be used for this problem. Our study has resulted in a new algorithm, known as core peeling [2], that reliably finds near-optimal solutions in time scaling quadratically in n (Fig. 2). Thus, it makes the problem computationally tractable. We expect that a theoretical analysis of the core peeling algorithm could improve what is currently the best theoretical bound on the optimal solution, dating from the 1980s.

Additionally, we have developed a load-balancing algorithm for computer networks that is based on a local load-redistribution protocol, redistributing a given processor's data packets according to the value of a "load function" at neighboring nodes [3]. Our analysis shows that under certain system assumptions, the algorithm functions at close to optimal network capacity. Since the approach is robust to a large class of update schedules, it can

provide a methodology for near-optimal dynamic redistribution of computational tasks—an instrumental tool for fault-tolerant network computing.

For more information contact Allon G. Percus at percus@lanl.gov.

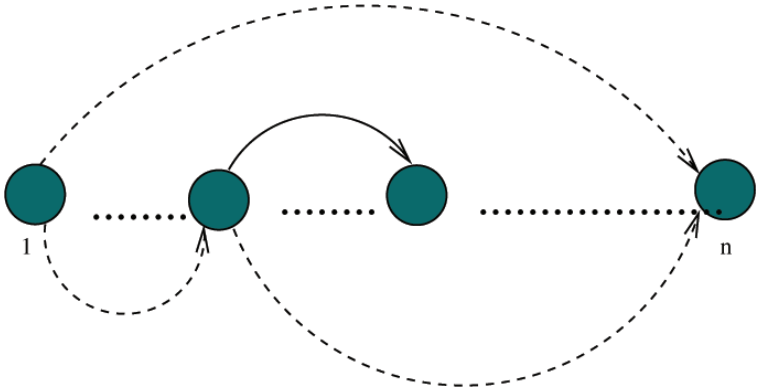

Fig. 1.
Communication patterns on **n** *processes represented as directed acyclic graph.*

[1] G. Istrate, et al., "The Cluster Structure of Minimum Bisections of Sparse Random Graphs," Los Alamos National Laboratory report LA-UR-06-6566 (2006).
[2] B.T. Goncalves, et al., "The Core Peeling Algorithm for Graph Bisection: An Experimental Evaluation," Los Alamos National Laboratory report LA-UR-06-6863 (2006).
[3] A. Hansson and C. Reidys, "A Class of Load Balancing Network Routing Protocols," in *Proceedings of European Modeling and Simulation Symposium;* part of International Mediterranean Modeling Multiconference, Barcelona, Spain, Oct. 2006, pp. 37–42. Invited Paper. Also available as Los Alamos National Laboratory report LA-UR-06-5281 (July 2006).

Funding Acknowledgements
Weapons Supported Research,
Computer Science Research Foundation.

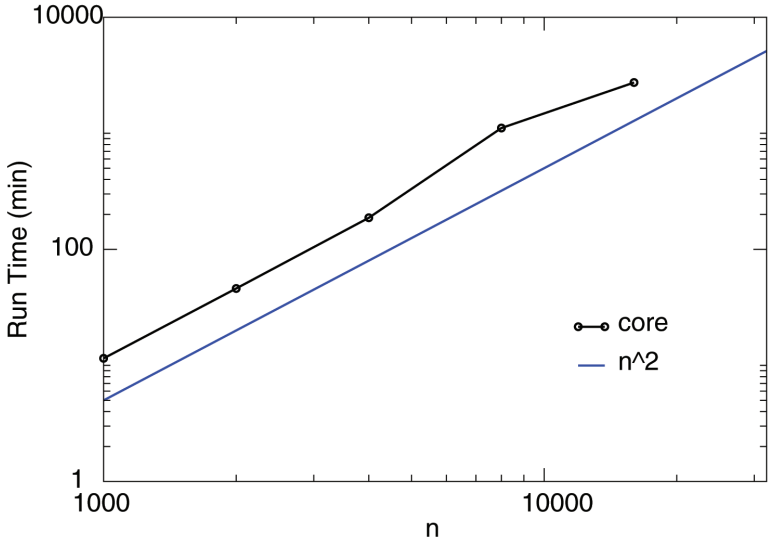

Fig. 2.
Scaling of running time for core peeling algorithm, compared with $\mathbf{n^2}$.