An Optimal Distributed Load Balancing Algorithm for Homogeneous Work Units

[Extended Abstract]

Akhil Langer
Department of Computer Science
University of Illinois at Urbana-Champaign, USA
alanger@illinois.edu

ABSTRACT

Many parallel applications, for example, Adaptive Mesh Refinement [1] simulations, need dynamic load balancing during the course of their execution because of dynamic variation in the computational load. We propose a novel tree-based fully distributed algorithm for load balancing homogeneous work units. The proposed algorithm achieves perfect load balance while doing minimum number of migrations of work units.

Categories and Subject Descriptors

G.1.0 [**Mathematics of Computing**]: Parallel algorithms

General Terms

Algorithms, Performance

Keywords

Distributed load balancing; adaptive mesh refinement; complexity analysis; scaling; high performance computing

1. THE DISTRIBUTED LOAD BALANCING ALGORITHM

The proposed algorithm is a spanning-tree based algorithm that is inspired by the distributed algorithm for constructing balanced spanning trees of process subgroups in parallel applications [2]. We form a spanning tree of all the processes, where each vertex of a tree corresponds to a process. The algorithm does two passes over the spanning-tree - the upward pass and the downward pass.

Upward pass: Processes send the count of work units in their subtree (of which the process itself is the root) to the parent process. At the end of the upward pass, each process has the count of work units in each of its child subtrees.

Downward pass: During this pass, migration decisions and work unit migrations take place. The scheme balances the tree while minimizing the number of migrations. The number of work units in the subtree and the size of the subtree are used to compute the number of work units per process in a perfectly balanced state. The root process and the child subtrees are categorized as either *work supplier* or *work consumer*. Each vertex V of the tree performs a "matchmaking" step, ensuring that each of *work supplier* is assigned one or more of *work consumer* that can absorb the excess work units of the supplier within their subtrees. If V itself needs some work units it requests work units from suppliers for itself. Similarly, if V itself has excess work, it tags itself as a work supplier. A vertex V concludes its role by calling the balancing step on the work suppliers if V was a work receiver or the work receivers if V was a work supplier. This alternation is done in order to ensure that the list of work suppliers and work receivers does not grow long, and in order to minimize the number of messages received by any vertex. Additionally, alternation also helps in early assignment of work units to their final destination processes. This makes it possible to migrate work units concurrently with the downward pass of the algorithm.

Our implementation is based on Charm++ [3], which is an object-based asynchronous message driven parallel programming paradigm.

ICS'14, June 10–13 2014, Munich, Germany.
ACM 978-1-4503-2642-1/14/06.
http://dx.doi.org/10.1145/2597652.2600108 .

2. REFERENCES

[1] Akhil Langer, Jonathan Lifflander, Phil Miller, Kuo-Chuan Pan, , Laxmikant V. Kale, and Paul Ricker. Scalable Algorithms for Distributed-Memory Adaptive Mesh Refinement. In *Proceedings of the 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2012). To Appear*, New York, USA, October 2012.

[2] Akhil Langer, Ramprasad Venkataraman, and Laxmikant Kale. Scalable Algorithms for Constructing Balanced Spanning Trees on System-Ranked Process Groups. In Jesper TrÃd'ff, Siegfried Benkner, and Jack Dongarra, editors, *Recent Advances in the Message Passing Interface*, volume 7490 of *Lecture Notes in Computer Science*, pages 224–234. Springer Berlin / Heidelberg, 2012.

[3] Laxmikant Kale, Anshu Arya, Nikhil Jain, Akhil Langer, Jonathan Lifflander, Harshitha Menon, Xiang Ni, Yanhua Sun, Ehsan Totoni, Ramprasad Venkataraman, and Lukasz Wesolowski. Migratable Objects + Active Messages + Adaptive Runtime = Productivity + Performance A Submission to 2012 HPC Class II Challenge. Technical Report 12-47, Parallel Programming Laboratory, November 2012.