Commun. Comput. Phys.
doi: 10.4208/cicp.030316.310716sw

Vol. **20**, No. 4, pp. 1106-1126
October 2016

COMPUTATIONAL SOFTWARE

DASHMM: Dynamic Adaptive System for Hierarchical Multipole Methods

J. DeBuhr[1,*], B. Zhang[1], A. Tsueda[2], V. Tilstra-Smith[3] and T. Sterling[1]

[1] *Center for Research in Extreme Scale Technologies, School of Informatics and Computing, Indiana University, Bloomington, IN, 47404, USA.*
[2] *College of Arts and Sciences, Loyola University Chicago, Chicago, IL, 60660, USA.*
[3] *Department of Physics and Mathematics, Central College, Pella, IA, 50219, USA.*

Received 3 March 2016; Accepted (in revised version) 31 July 2016

Abstract. We present DASHMM, a general library implementing multipole methods (including both Barnes-Hut and the Fast Multipole Method). DASHMM relies on dynamic adaptive runtime techniques provided by the HPX-5 system to parallelize the resulting multipole moment computation. The result is a library that is easy-to-use, extensible, scalable, efficient, and portable. We present both the abstractions defined by DASHMM as well as the specific features of HPX-5 that allow the library to execute scalably and efficiently.

AMS subject classifications: 15A06, 31C20, 68N19

Key words: Barnes-Hut method, fast multipole method, Laplace potential, ParalleX, runtime software.

Program summary

Program title: DASHMM

Nature of problem: Evaluates the Laplace potentials at N target locations induced by M source points.

Software license: BSD 3-Clause

CiCP scientific software URL: `http://www.global-sci.com/code/dashmm-0.5.tar.gz`

Distribution format: .gz

Programming language(s): C++

*Corresponding author. *Email address:* `jdebuhr@indiana.edu` (J. DeBuhr)

Computer platform: x86_64

Operating system: Linux

Compilers: GCC 4.8.4 or newer; icc (tested with 15.0.1)

RAM:

External routines/libraries: HPX-5 2.1.0 or later

Running time:

Restrictions: Currently only supports shared memory, but library will be extended to multiple nodes

Supplementary material and references: `https://www.crest.iu.edu/projects/dashmm/` `http://hpx.crest.iu.edu/`

Additional Comments:

1 Introduction

Multipole methods are a key computational kernel in a wide variety of scientific applications spanning multiple disciplines. However, these applications are emerging as scaling-constrained when using conventional parallelization practices. Emerging, dynamic task management execution models can go beyond conventional programming practices to significantly improve both efficiency and scalability for algorithms exhibiting irregular and time-varying executions. Multipole method calculations are an example of an irregular application that would benefit from the use of a dynamic adaptive runtime system approach. In this paper, we present DASHMM, a library leveraging the power and expressibility of an experimental runtime system to improve the efficiency and scalability of multipole method calculations, to solve ever more challenging end-user problem domain applications.

The Barnes-Hut (BH) [5] method and the Fast Multipole Method (FMM) [11], both exemplars of multipole methods, are used extensively in applications [7, 10, 19, 23]. The fundamental building blocks of these methods are quite similar, and so it is worth constructing a generalization of multipole methods that encompasses both BH and FMM. In practice, many variations and improvements on the original forms of these methods are used; a library that provides only a few rigidly defined methods would be of little use. Instead, DASHMM outlines a set of abstractions that are general enough to allow both BH and the FMM to be implemented as well as many variations on the theme of multipole methods. With the abstractions defined, the task is to effectively parallelize the resulting general multipole method.

Considerable prior effort has been exerted on multipole methods. The first parallel FMM algorithm is introduced by Greengard and Gropp in its 2D uniform version on shared memory architecture [9]. Zhao and Johnsson studied the parallelization of 3D uniform FMM on the connection machine [34]. The parallelization strategy follows