Revision: "PNBsolver: A Domain-Specific Language for Modeling Parallel N-body Problems"

Ferosh Jacob, Md Ashfakul Islam, Weihua Geng,
Jeff Gray, Susan Vrbsky, Brandon Dixon,
and Purushotham Bangalore

June 26, 2016

In this document, we explain in detail how we revised the paper addressing the comments and concerns of the reviewer. The five main reviewer comments and the corresponding modifications are described in the following subsections.

1 Analytic results

Reviewer description: "The paper lacks ANALYTIC results. The implementation/simulation validation is not enough to warrant the publication of this paper. The authors should add analytic analysis of the proposed solution."

Paper modifications: We included a detailed analysis of tree code algorithm showing how the efficient implementation of the tree code algorithm can be leverage for N-body problems. The changes made in the paper are listed below.

1. We added the following line to the end of **Section 2. N-body problems and tree code algorithm**.

   In this section, the analytical formulation of tree code algorithm is reviewed, explaining how the efficient implementations of this algorithm can be reused in N-body computations.

2. Rewrote the second paragraph in **2.1. Tree code algorithm** as following.

   The tree code algorithm has proved its efficiency for N-body problems [18]. It reduces the computational cost of these problem from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log(N))$. The tree code algorithm is explained in two perspectives: 1) Analytical formulation; and 2) Numerical implimentation. In the

analytical formulation, the important concepts of multipole expansion, moments as well as the novel recurrence relation idea for computing multi-indexed Taylor expansion are introduced. Error and computational cost analysis are provided.

3. Included a new subsection, **2.1.1. Tree code algorithm: analytical formulation** explaining the analytical formulation of tree code algorithm. The newly added subsection is given below.

2.1.1. Tree code algorithm: analytical formulation

In a system consisting of N bodies, the interaction of body i with other $N-1$ bodies is given in Equation 1.

$$I_i = \sum_{j=1, j\neq i}^{N} M_j G(x_i, y_j) \tag{1}$$

According to the tree code algorithm, for bodies within a cell/cluster (one of the subunits of the 3-D cube) satisfying the maximum acceptance criterion (MAC) [22] in Equation 2 for $\theta < 1$, their interaction with body i can be calculated more efficiently with controllable errors.

$$\frac{r_c}{R} < \theta \tag{2}$$

In Equation 2, r_c represents the location of the cell center and R represents the distance between the ith body and the cell center. On satisfying the MAC, the more efficient particle-cluster interaction between body i and bodies inside the cell c is given as $I_{i,c}$ in Equation 3 from [19, 23, 24]. Here p represents the ORDER of the Taylor expansion. Note that k, x, and y are 3-D vectors.

$$I_{i,c} \approx \sum_{||k||=0}^{p} \frac{1}{k!} D_x^k G(x_i, y_c) \sum_{y_j \in c} M_j (y_j - y_c)^k \tag{3}$$

$$\approx \sum_{||k||=0}^{p} a^k(G(x_i, y_c)) m_c^k \tag{4}$$

Where $a^k(G(x_i, y_c)) = \frac{1}{k!} D_x^k G(x_i, y_c)$ is the kth order Taylor coefficients and m_c^k is the kth order moment of the cell c.

In Equation 3, for each k, the Taylor coefficients need to be computed for every target body using recurrence relations. Recurrence relations are usually kernel-dependent. For example, let $a^0(G(x, y)) = 1/|x - y|$ be the Coulombic interaction, the higher order Taylor coefficients can

be computed recurrently by

$$||k|||x-y|^2 a^k - (2||k||-1)\sum_{i=1}^{3}(x_i-y_i)a^{k-e_i} + (||k||-1)\sum_{i=1}^{3}a^{k-2e_i} = 0 \tag{5}$$

where e_i's are standard 3-D unit vectors. It worths noting that Tausch developed a kernel-independent recurrence relation for wider application of the Cartesian multipole expansion including fast multipole method and tree code [25].

The moment of the cell, on the contrary, is only evaluated once for each cell, i.e. the moment is a property associated with each cell. From an implementation perspective, to implement this algorithm for any interaction, programmers only need to choose one of our included interactions or provide the Taylor coefficients for user-specified interactions.

Equation 3 also reveals the error and computational cost of the tree code. The error is essentially $\mathcal{O}(\theta^{(p+1)})$ due to the Taylor expansion to the order p. The total computational cost is $\mathcal{O}(p^3 N \log N)$ (a little bit more specified than $\mathcal{O}(N \log N)$), where p^3 is from the summation over the magnitude of 3-D vector k from 0 to p, N is the number of target particles and $\log N$ is the level of trees.

2 DSL implementation effort

Reviewer description: "Please discuss also the effort to implement PNBSolver. This is an important issue since trade-off between developing a DSL and its later usage should be contrasted with classic GPL approach. How quickly similar DSLs can be developed?"

Paper modifications: We added a new subsection **4.4 PNBsolver Implementation Effort** to explain the implementation and source code details. The newly added subsection is given below.

4.4 PNBsolver Implementation Effort

Table 1: Code analysis of PNBsolver

Code type	Language	files	blank	comment	code
Generated	Java	2	868	468	2572 (87%)
Manual	Java	9	73	4	385 (13%)

The source code for the PNBsolver is publicly available on github[1]*. After we finalized the design, the core implementation was finished in a weekend.*

[1]PNBsolver source code, `https://github.com/feroshjacob/PBNsolver`

The implementation has 2957 lines of code (estimated by CLOC[2]) distributed in 11 Java files. The details of the manual and generated code (generated from the grammar using the ANTLR generator tool) is given in the Table 3. As shown in the table, 87% code was generated and only 385 lines of code was handwritten for implementing the PNBsolver.

3 Related work

Reviewer description: "PNBSolver is a tiny DSL. It can be probably replaced with powerful annotation system. For example as described in: Walter Cazzola, Edoardo Vacchi. @Java: Bringing a richer annotation model to Java. Computer Languages, Systems & Structures, Volume 40, Issue 1, April 2014, Pages 2-18. Please discuss pros and cons of such approach in the related work section."

Paper modification: We added the following line to the subsection, **7.3. Modeling in parallel programming**.

@Java [49] extends the Java Annotation model and enables programmer to define custom annotations in Java code and such annotations can be evaluated dynamically. PNBsolver differ from @Java because it is not tightly bound to any general purpose language and users only have to familiarize with PNBsolver semantics to use PNBsolver.

4 Typo errors

Reviewer description: "Typo on page 16: PNBSolver to to Li − > PNBSolver to Li"

Paper modification: We fixed the typo.

5 Images in the paper

Reviewer description: "Will figures appear in color? Namely, current figures are not very readable on black ink printer."

Paper modification: None, Yes the current figures are in color. We will check with the publisher whether we have to convert once the paper is accepted.

[2]Count Lines of Code (CLOC), `http://cloc.sourceforge.net`