Efficient SDH Computation In Molecular Simulations Data

Yi-Cheng Tu [1], Shaoping Chen [2], Sagar Pandit [3], Anand Kumar [4], and Vladimir Grupcev [5]

[1,4,5] Department of Computer Science and Engineering, University of South Florida, Tampa, Florida - 33620
[2] Department of Mathematics, Wuhan University of Technology, 122 Luosi Road, Wuhan, Hubei, 430070, P. R. China
[3] Department of Physic, University of South Florida, Tampa, Florida - 33620

{[1]ytu, [4]akumar8, [5]vgrupcev}@cse.usf.edu, [2]chensp@whut.edu.cn, [3]pandit@cas.usf.edu

ABSTRACT

Analysis of large particle or molecular simulation data is integral part of the basic-science research community. It often involves computing functions such as point-to-point interactions of particles. Spatial distance histogram (SDH) is one such vital computation in scientific discovery. SDH is frequently used to compute Radial Distribution Function (RDF), and it takes quadratic time to compute using naive approach. Naive SDH computation is even more expensive as it is computed continuously over certain period of time to analyze simulation systems.

Tree-based SDH computation is a popular approach. In this paper we look at different tree-based SDH computation techniques and briefly discuss about their performance. We present different strategies to improve the performance of these techniques. Specifically, we study the *density map* (DM) based SDH computation techniques. A DM is essentially a grid dividing simulated space into cells (3D cubes) of equal size (volume), which can be easily implemented by augmenting a Quad-tree (or Oct-tree) index. DMs are used in various configurations to compute SDH continuously over snapshots of the simulation system. The performance improvements using some of these configurations is presented in this paper. We also present the effect of utilizing computation power of Graphics Processing Units (GPUs) in computing SDH.

Categories and Subject Descriptors

I.6.6 [**Computing Methodologies**]: Simulation and Modeling—*Simulation Output Analysis*

Keywords

Molecular Simulation, Radial Distribution Function, Spatial Distance Histogram, Structural Biology, Protein Structure

ACM-BCB'12, October 7-10, 2012, Orlando, FL, USA.
ACM 978-1-4503-1670-5/12/10

1. INTRODUCTION

Particle/Molecular simulations (PS/MS[1]) are computer simulations in which the basic components of large systems interact with one another under certain postulated empirical forces for certain duration of time [2]. Simulation techniques are primarily applicable in the modeling of complex chemical and biological systems that are beyond the scope of theoretical models. MS are applied in various scientific domains such as material sciences, biomedical sciences, and biophysics. Large scale celestial structure formation is another example from astrophysics where N-body simulations are predominantly used [5, 7].

Quantitative analysis of large simulation data is important for scientific discoveries. Statistical properties of particles or some complex functions on particles' coordinates are often computed to find interesting patterns [5]. Interactions between pairs of particles and their statistical properties are of special interest to scientists. Computation of these quantities for N particles require $O(N^2)$ computations using brute-force approach. In this paper, we focus on one such analytical function: the Spatial Distance Histogram (SDH), which asks for a histogram of the distances of all pairs of particles in the simulated system. SDH is often used to compute Radial Distribution Function (RDF). Computation of thermodynamic characteristics of a system require RDF. The RDF is important in the computation quantities like total pressure and energy, otherwise these cannot be calculated.

In this paper we look at different tree-based SDH computation techniques and discuss briefly about their performance. We present different strategies to improve the performance of these techniques. Specifically, we study the *density map* (DM) based SDH computation techniques. A DM is essentially a grid dividing simulated space into cells (3D cubes) of equal size (volume), which can be easily implemented by augmenting a Quad-tree (or Oct-tree) index. DMs are used in various configurations to compute SDH continuously over snapshots of the simulation system. The performance improvements using some of these configuration is presented in this paper. We also present the idea of utilizing computation power of Graphics Processing Units (GPUs) in computing SDH. Characteristics of the GPU system are briefly discussed along with challenges of implementation.

[1]We use MS and PS interchangeably.

2. SDH BASICS

An algorithm to compute SDH based on a data structure called *density map* (DM) is presented here. A DM is essentially a grid dividing simulated space into cells (3D cubes) of equal size (volume), which can be easily implemented by augmenting a Quad-tree(or Oct-tree) index [10]. To generate a DM of higher resolution, we divide each cell of this grid again into equally sized cells. We use a region Quad-tree to organize different density maps of the simulation data. A tree node represents each cell of the DM, so a density map is essentially the collection of all nodes on one level of the tree. Count of points in each cell are stored in the corresponding tree node. Nodes with zero count are removed from the tree, as they do not carry any particles. The height of the tree (denoted as h) is determined in a way such that the average number of points in all possible leaf nodes is no smaller than a predefined threshold β. To be specific, we have

$$h = \lceil \log_{2^d} N/\beta \rceil$$

This is necessary to ensure that the DM-based algorithms do not enter the quadratic-time computation state.

The histogram of distances between all pairs of particles is built by traversing the tree, starting at a specific level, while processing pairs of cells in each level. Focal point of this algorithm is a procedure named ResolveTwoTrees. To resolve two cells A and B (with total particle counts n_a and n_b, respectively) of a DM we first read the coordinates of the two cells and compute the minimum and maximum distances between these two cells. These distances are called **range**. If the range falls into a required distance range of a histogram bucket i, we say A and B are *resolvable*. In this case, we increment the count of bucket i by $n_a \times n_b$. If the two cells are not resolvable we can follow different strategies to compute their histogram (Figure 1):

1. Resolve all cell pairs formed between the children of A and B in the Quad-tree.

2. Apply heuristics to distribute the number of distances into buckets covered by the minimum and maximum distances (range) between cells.

The ResolveTwoTrees procedure is applied on all pairs of cells of the starting DM. The starting DM can be a level in which cell diagonal is less than or equal to the histogram bucket width.

It is easy to see that no matter how small the cells are in a density map, non-resolvable cell pairs always exist. Therefore, when we reach the lowest level of the tree (case 1 above), we have to calculate all point-to-point distances of the particles in the unresolved cells. The heuristics (case 2) can be applied whenever we decide to stop traversing the tree.

Existence of chemical bonds and/or inter-particle forces in natural systems often tend to spread out the particles evenly in sub-regions [2, 3].[2] Also the particles do not move randomly in the system. Group of particles tend to move in a smooth trajectory path. Therefore, there is a potential to use the spatial and temporal uniformity for efficient histogram computation. Utilizing the spatiotemporal uniformity present in sub-regions of simulation space, captured by cells of DM, can improve the SDH computation time while applying the heuristics. Also, computation time can be significantly improved by employing powerful GPUs. In the following section we briefly look into the details of all such strategies.

[2]SDH computation becomes a trivial task if whole simulation system is uniform.

Algorithm ResolveTwoTrees
Input: Cells A, B; Counts n_a, n_b; Heuristic
Output: Distance Histogram H

```
1  if A and B are resolve into bucket i
2      H[i] = H[i] + n_a n_b
3  else if Heuristic then Apply heuristic and update H
4  else if A and B are leaf nodes
5      Compute point-to-point distances between points of
6        A and B
7      Update distances in H
8  else for each child a of A
9          for each child b of B
10             ResolveTwoTrees(a, b)
```

Figure 1: Resolving sub-trees of density map tree

3. COMPUTATION STRATEGIES

The ResolveTwoTrees procedure is the basic computation unit before applying any of the strategies listed above. We discuss various strategies that can be applied to compute the SDH efficiently.

Basic Techniques.

In current state-of-the-art SDH computation techniques space-partitioning trees such as *kd*-trees, as reported in [6, 11], cluster the data. The clusters are treated as basic processing units to build the histogram. The key idea in such methods is to process all the particles in a tree node (cluster) as a whole, rather than processing particle-to-particle distances.

A DM-based SDH (DM-SDH) algorithm designed using Quad-tree data structure is presented by Tu *et al.* [11]. They have studied the algorithm running time theoretically and presented formal proof of DM-SDH along with experimental results on real MS data sets. It has been proved that the running time for DM-SDH is $\Theta(N^{\frac{3}{2}})$ for 2D data and $\Theta(N^{\frac{5}{3}})$ for 3D data. Case 1 explained in previous section is essentially the DM-SDH algorithm.

Basic Techniques with Heuristics.

A histogram by itself is an approximation of the underlying data distribution, an approximate histogram generated from a given dataset will still be useful in statistical sense. In many cases a coarse SDH will greatly help fine-tune the simulation programs. An approximate SDH algorithm (ADM-SDH), with running time not related to the data size N was also introduced in [11]. This algorithm follows the idea of `ResolveTwoTree` with an exception that the sub-trees rooted at non-resolvable cells are not traversed when a desired error bound is achieved. The distances $n_a n_b$ are distributed over the range of buckets choosing one of the following strategies:

- All distances into one bucket that is chosen randomly,
- Evenly distributed into all buckets of overlap range, or

- According to proportions of range overlapping over each bucket of the histogram.

All these heuristics take constant time to compute the solution for two cells. Running time of ADM-SDH is influenced by a guaranteed error bound as well as by the histogram bucket size w. A thorough analysis of error bounds and the performance of ADM-SDH is presented in a recent work [4].

SDH is often computed on consecutive snapshots (called **frames**) of the simulation system. The basic techniques have to be started afresh on each new frame to get the SDH. Algorithms can be significantly improved if spatiotemporal uniformity of the simulation system is utilized. Existence of chemical bonds and/or inter-particle forces in natural systems often tend to spread out the particles evenly [2, 3]. As a result particles are found to be uniformly distributed in localized regions of the simulation space. Such uniform regions can be assumed as single components in SDH computation while introducing very small errors.

Heuristics Utilizing Spatiotemporal Uniformity.

Uniform distribution of particles in localized regions leads us to new heuristic for SDH computation. As shown in [1] utilizing this type of uniformity does not introduce significant errors in the output histogram. Exploiting uniform region property makes the running time of the algorithm independent of the bucket width w – such dependency is the main drawback of existing algorithms. We generate the statistical distribution of the distances between uniform regions via Monte Carlo simulations, and put the $n_a n_b$ distances into overlapping range of histogram buckets based on this distribution. This is similar to the heuristics discussed above.

Temporal similarity among neighboring frames is another property that can be combined with uniform regions. Given a frame f_0 and its SDH, we can obtain the SDH of next frame f_1 by dealing only with the regions that do not exhibit temporal similarity between the two frames, while ignoring regions that are similar. We designed an *incremental algorithm* [1] that can quickly compute SDH of a frame from the SDH of a base frame that was obtained using traditional single-frame algorithms.

A combination of temporal similarity and spatial uniformity properties can improve SDH computation. Temporal similarity introduces small errors when cells with very small changes are eliminated from the computation. If this property is to be ignored, the uniform property can still be applied for efficient SDH computation.

Computations Using GPUs.

Parallel processing is an obvious strategy to reduce computing time in our problem. GPUs are massively parallel systems and are more effective than general purpose CPUs, especially for algorithms like ours that process large blocks of data in parallel. Large number of threads can be created on the GPU multi-processors. These multi-processors execute threads in SIMD (Single Instruction Multiple Data) fashion [9]. Thread creation and context-switch times are very low for the GPUs as compared to the threads on CPUs.

We develop an algorithm to take advantage of GPU's processing power. The cells of a chosen DM are placed in the GPU global memory such that consecutive threads can read consecutive cells through coalesced access mechanism [8]. Every single GPU thread processes a pair of cells in the DM. A distinct pair of cells is processed by each thread. Each block (group of threads) executing on a multiprocessor processes different portions of the density map. Thus, parallel processing can help improve performance of the algorithm.

Global memory of the GPU can cause performance issues due to its limited speed of access. Density map can be placed in the shared memory to get better access speeds. Each thread can access distinct pairs of cells from the shared memory for SDH computation. One major obstacle in implementing this idea is the limited size of shared memory. The algorithm design and implementation should address this probelm.

4. CONCLUSIONS

In this paper we discussed different tree-based SDH computation algorithms. The basic exact solution performs better than quadratic-time algorithms. Approximate version of the basic solution can promise error bounds and improve computation time. However, the real performance improvement was observed when spatiotemporal uniformity of the simulation data was utilized. Quad-tree was efficiently used to take advantage of the data locality and statistical data distribution properties. Therefore, spatiotemporal uniformity based algorithms showed significant performance improvements. We further improved the performance of these algorithms with the use of GPUs. In future, the GPUs can play important role in computation of multi-body correlation functions and other complex analytical functions which are important to the scientific community.

5. REFERENCES

[1] A. Kumar *et al.* Distance histogram computation based on spatiotemporal uniformity in scientific data. In *EDBT*, pages 288–299, 2012.

[2] M. Allen. *Introduction to Molecular Dynamics Simulation*, volume 23. John von Neumann Institute of Computing, NIC Seris, 2003.

[3] Andrey Omeltchenko *et. al.* Scalable I/O of large-scale molecular dynamics simulations: A data-compression algorithm. *Computer physics communications*, 131(1–2):78–85, 2000.

[4] S. Chen, Y.-C. Tu, and Y. Xia. Performance analysis of a dual-tree algorithm for computing spatial distance histograms. *The VLDB Journal*, 20, 2011.

[5] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From Algorithms to Applications*, volume 1. Academic Press, Inc., 2nd edition, 2001.

[6] A. G. Gray and A. W. Moore. N-body problems in statistical learning. In *NIPS*, pages 521–527, 2001.

[7] D. Landau and K. Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 2005.

[8] NVIDIA. CUDA C Best Practices Guide, Ver. 4, 2011.

[9] NVIDIA. CUDA C Programming Guide, Ver. 4, 2011.

[10] J. A. Orenstein. Multidimensional tries used for associative searching. *Information Processing Letters*, 14(4):150–157, 1982.

[11] Y.-C. Tu, S. Chen, and S. Pandit. Computing distance histograms efficiently in scientific databases. In *ICDE*, pages 796–807, 2009.