New parallelization model of sequential Monte Carlo analysis with prediction-correction computing

Eiji Tomiyama
Research Organization for Information Science & Technology (RIST)
1-5-2, Minatojimamimami, Chuo, Kobe, Hyogo 650-0047, JAPAN
+81-78-599-9556
tomiyama@rist.or.jp

Hiroshi Koyama
Research Organization for Information Science & Technology (RIST)
1-5-2, Minatojimaminami, Chuo, Kobe, Hyogo 650-0047, JAPAN
+81-78-599-9552
hkoyma@rist.or.jp

Katsumi Hagita
National Defense Academy
1-10-20, Hashirimizu, Yokosuka 239-8686, JAPAN
+81-46-841-3810
hagita@nda.ac.jp

ABSTRACT
Monte Carlo (MC) search with Metropolis judgment has been used widely to solve inverse problems; in particular, reverse Monte Carlo (RMC) analysis has been shown to make a possible configuration of particles in terms of the structure factor $S(q)$ obtained by X-ray scattering experiments. In general, MC search proceeds sequentially, because each judgement of the MC trial depends on its previous state. In the present study, we have developed a new “SimpleRMC” parallel code for RMC analysis for a system with millions of particles. In this code, two hotspots are identified and optimized for the histogram $h(r)$ and difference of histogram $\Delta h(r)$ calculations. For the histogram kernel, we achieved 339 TFlops performance (32.3% of the peak) on 8,192 nodes of the K computer using a 33,554,432-particle system. This newly developed “prediction–correction” method improves both the parallel performance of the $\Delta h(r)$ calculation and its elapsed time.

Categories and Subject Descriptors
D.1.3 [PROGRAMMING TECHNIQUES]: Concurrent Programming - Parallel programming

General Terms
Experimentation

Keywords
Reverse Monte Carlo, histogram calculation, parallel algorithm, prediction–correction

1. INTRODUCTION
Recently, polymer nanocomposites (PNCs) filled with nanoparticles have been used widely as highly functional industrial materials, including functional film, tires, and automotive parts. The morphology (i.e., 3D configuration) of nanoparticles is an important key controlling the mechanical properties of PNCs. In general, it is difficult to observe the real spatial structure of nanoparticles in polymer matrices. However, information about their configurations can be through analysis of their scattering spectra in inverse space based on small-angle X-ray scattering. For a monodisperse system, the scattering spectrum corresponds to a structure factor $S(q)$. Using this $S(q)$, reverse Monte Carlo (RMC) analysis [1] is able to provide a possible configuration of nanoparticles using sequential Monte Carlo (MC) search. Here, we present performance results of our newly developed code “SimpleRMC,” which can conduct RMC analysis for a system with millions of particles.

2. METHODS
The RMC method calculates the structure factor $S(q)$ using a radial distribution function $g(r)$. The variables $S(q)$ and $g(r)$ can be described by the histogram $h(r)$ of all particle pairs as a function of pair distance $r$, as follows:

$$g(r) = \frac{h(r)}{4\pi r^2 \rho}, \quad S(q) = 4\pi \rho \int dr r^2 (g(r) - 1) \frac{\sin(qr)}{qr} + 1,$$

where $\rho$ is the number density of the nanoparticles. The difference of the error sum $\Delta\chi^2$ must be calculated at every MC step. The difference of the radial distribution function $\Delta g(r)$ and the structure factor $\Delta S(q)$, defined as follows, are necessary to evaluate $\Delta\chi^2$:

$$\Delta g(r) = \frac{\Delta h(r)}{4\pi r^2 \rho}, \quad \Delta S(q) = 4\pi \rho \int dr r^2 (\Delta g(r)) \frac{\sin(qr)}{qr}.$$ 

Therefore, the performance of each MC sampling strongly depends on the calculation of $\Delta h(r)$.

The floating point operations of $h(r)$ and $\Delta h(r)$ are $O(N^2)$ and $O(N)$, respectively, where $N$ is the number of particles. Although $h(r)$ appears only once as an initial estimate in this algorithm, the numerical cost of $h(r)$ is still high. Therefore, in practice, a high-performance calculation is required for $h(r)$, as described in detail in Section 3.1. Conversely, a $\Delta h(r)$ calculation can be conducted for each trial move as part of a “sequential” MC method. Here, we propose a “prediction–correction” method that allows these sequential trials to be calculated in parallel for $M$ trials. The detailed algorithm is presented in Section 3.2.

3. RESULTS AND DISCUSSION
We have developed a new code named “SimpleRMC” that includes both “sequential” and “prediction–correction” MC methods. This code was designed to allow hybrid parallelization with MPI and automatic thread parallelization. We have incorporated two tuning procedures into the code. Using a 4,194,304-particle system, the histogram $h(r)$ calculation kernel executed in 104.7 s on 128 nodes of the K computer. The corresponding performance was 5.98 TFlops (i.e., 36.5% of the...
3.1 Optimization for histogram kernel

Although the histogram calculation is \(O(N^2)\), the kernel loop remains a memory bottleneck. The kernel loop consists of 29 floating point operations (Figure 1), including one square root and three aint built-in functions. We note that “aint” is the fastest of the round-to-integer operations on the Fujitsu Fortran compiler K-1.2.0-15. Demand memory access in this loop is 16 bytes. Thus, the required byte per flop ratio (B/F) is 0.55, which is slightly higher than the 0.36 for the SPARC64 X11fx CPU on the K computer.

We modified the original kernel loop for the single instruction multiple data (SIMD) hardware with fused multiply–add (FMA) support in terms of instruction-level parallelization. To prevent indirect memory access, we divided the kernel loop into two loops: distance computation for all particle pairs (Loop1), and histogram reduction on each thread (Loop2). Note that there is only one floating-point operation in Loop2. We achieved 51% of the theoretical peak in Loop1 and 36.5% in both loops when 75% of the instructions were SIMD vectorized. Additionally, 191.5 KB of the histogram reduction was performed once at the end of the operation with MPI Allreduce, although this contributed only 0.9 ms of the elapsed time. We achieved a kernel performance of 339 Tflops (i.e., 32.3% of the peak) using 8,192 nodes for a 33,554,432-particle system.

![Figure 1. Histogram calculation kernel](image)

3.2 Advanced parallel computing of difference of histogram

We have proposed a new parallel “prediction–correction” method to help reduce MPI communications. This method calculates \(\Delta h(r)\) for \(i = 1\) to \(M\) simultaneously, where each \(\Delta h(r)\) consists of an independent particle move. The excluded volume (EV) is calculated before each \(\Delta h(r)\) calculation to prevent unphysical conditions for \(\Delta h(r)\). In the 4,194,304-particle system, approximately 13% of the 100,000 trials satisfied the EV conditions. Next, approximately 60% of the satisfied trials were accepted in terms of the \(\chi^2\) judgment, which is calculated for \(M\) trials simultaneously. This concept is illustrated on the right-hand side of Figure 2.

In general, increasing \(M\) reduces the time required to obtain the solution. However, the correction of \(\Delta h(r)\) requires the cost of communication in addition to the cost of \(O(M^2)\). Thus, it is intuitive that an optimal value of \(M\) exists. The elapsed times benchmarks for several values of \(M\) are presented in Table 1. \(M = 64\) was found to require the shortest interval for the 4,194,304-particle system, with 22% and 44% of the elapsed time for communication and the “sequential” method, respectively. For these benchmarks, most Message sizes are smaller than 11 KB. Therefore, the communication cost depends on the count of the MPI function call (Table 2). Number of the MPI call is reduced to \(1/M\). Consequently, the calculation time of \(\Delta h(r)\) for \(M = 64\) was 2.9 s (14%) shorter than that of the “sequential” method.

![Figure 2. Flowchart of the “sequential” (left) and “prediction–correction” (right) methods](image)

**Table 1. Elapsed time of the \(\Delta h(r)\) calculation on 24 nodes**

<table>
<thead>
<tr>
<th>(M)</th>
<th>Sequential</th>
<th>M = 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>20.6 s</td>
<td>7.66 s</td>
</tr>
<tr>
<td>16</td>
<td>20.9 s</td>
<td>3.06 s</td>
</tr>
<tr>
<td>64</td>
<td>19.3 s</td>
<td>1.46 s</td>
</tr>
<tr>
<td>128</td>
<td>17.7 s</td>
<td>14.37 s</td>
</tr>
</tbody>
</table>

**Table 2. Time and count of collective communication**

<table>
<thead>
<tr>
<th>MPI function</th>
<th>Sequential</th>
<th>M = 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alldistribute</td>
<td>39,927</td>
<td>13,921</td>
</tr>
<tr>
<td>Beast</td>
<td>3.40 s</td>
<td>14.37 s</td>
</tr>
<tr>
<td></td>
<td>153,236</td>
<td>14.37 s</td>
</tr>
</tbody>
</table>

Using this “prediction–correction” method, efficient parallel calculation is possible for larger numbers of particles than using traditional methods. We also benchmarked a 33,554,432-particle system on 256 nodes of the K computer; the elapsed time with \(M = 32\) was 22.4 s, which is faster than that of the \(M = 64\) run (24.7 s).

4. CONCLUSIONS

We have developed RMC analysis code for a huge monodisperse system. In particular, we identified two hotspots: histogram \(h(r)\) and difference of histogram \(\Delta h(r)\) calculations. Our code achieved high performance in \(h(r)\) calculation. Moreover, the performance of \(\Delta h(r)\) reached a maximum for an optimal value of \(M\), indicating that our “prediction–correction” method is very useful for large-particle systems as well as large parallel computers.

5. ACKNOWLEDGMENTS

The performance analysis and optimization were partially performed in the “users tuning support program” at RIST. KH is partially supported by “Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures” and “High Performance Computing Infrastructure” (ID: jh140004-NA02, hp130062, hp140191). The authors thank Messrs. T. Noguchi and T. Harayama at RIST, Dr. T. Honda and Mr. A. Takayanagi at ZEON Corporation (hp140238), and Drs. T. Sone and T. Tominaga at JSR Corporation (hp140239) for their useful discussions.

6. REFERENCES