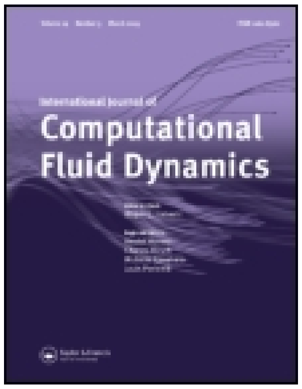

International Journal of Computational Fluid Dynamics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcfd20

# Parallel Implementation of DSMC Using Unstructured Mesh

J.-S. Wu , K.-C. Tseng & T.-J. Yang
[a] Department of Mechanical Engineering , National Chiao-Tung University , 1001 Ta Hsueh Road, Hsinchu, Taiwan

Published online: 10 Oct 2011.

To cite this article: J.-S. Wu , K.-C. Tseng & T.-J. Yang (2003) Parallel Implementation of DSMC Using Unstructured Mesh , International Journal of Computational Fluid Dynamics, 17:5, 405-422, DOI: 10.1080/1061856031000081497

To link to this article: http://dx.doi.org/10.1080/1061856031000081497

*International Journal of Computational Fluid Dynamics*, October 2003 Vol. 17(5), pp. 405–422

# Parallel Implementation of DSMC Using Unstructured Mesh

J.-S. WU*, K.-C. TSENG and T.-J. YANG

*Department of Mechanical Engineering, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan*

*(Received 7 March 2002; In final form 8 December 2002)*

The parallel implementation of the Direct Simulation Monte Carlo (DSMC) method on memory-distributed machines using unstructured mesh is reported. Physical domain decomposition is used to distribute the workload among multiple processors. A high-speed driven cavity flow is used as the benchmark problem for the validation of the parallel implementation. Three static partitioning techniques including simple coordinate partitioning, two-step partitioning (JOSTLE) and multi-level partitioning (METIS) are used for static domain decomposition, respectively. A cell renumbering technique is used to improve the memory management efficiency. Results of parallel efficiency show that two-step partitioning using JOSTLE performs the best, with 63% up to 25 processors, due to better load balancing among the processors. The powerful computational capability of the parallel implementation is demonstrated by computing a 2-D, near-continuum, hypersonic flow over a cylinder as well as a 3-D hypersonic flow over a sphere, respectively, both using 25 processors. Results compare reasonably well with previous simulated and experimental studies.

*Keywords*: Parallel computing; Direct simulation Monte Carlo; Unstructured mesh; Partitioning; Driven cavity; Load balancing

## INTRODUCTION

The highly developed Computational Fluid Dynamics (CFD) using the Navier–Stokes equations has permitted the prediction of complex thermal and fluid problems of engineering and scientific interest. However, there are limitations to the applicability based on these equations. In some flow regimes, the Navier–Stokes equations fail to represent the gas dynamics behavior and the particle nature of the matter must be taken into account. One of these is the rarefied gas flow, which the mean free path becomes comparable with, or even larger than, the characteristic length of flows. Such flows can be characterized by the Knudsen number, $\mathrm{Kn} = \lambda/L$, where $\lambda$ is the mean free path, and $L$ is the characteristic length. The Knudsen number is usually used to indicate the degree of rarefaction. Traditionally, flows are divided into four regimes as follows (Schaff and Chambre, 1958): $\mathrm{Kn} < 0.01$ (continuum), $0.01 < \mathrm{Kn} < 0.1$ (slip flow), $0.1 < \mathrm{Kn} < 3$ (transitional flow) and $\mathrm{Kn} > 3$ (free molecular flow). As the Kn increases, the rarefaction becomes important and even dominates the flow behavior. Hence, the N–S-based CFD techniques are often inappropriate for higher Kn flows, such as slip flow, transitional flow and free molecular flow.

The understanding of rarefied gas dynamics (high Knudsen number flows) has played or has begun to play an important role in several research disciplines. The examples include the spacecraft re-entry (Nance, 1995), the plume impingement from control thrusters on satellite (Boyd *et al.*, 1996; Kannenberg, 1998), the pumping characteristics of high vacuum pump (Lee and Lee, 1996a,b), the low-pressure plasma-etching and chemical vapor deposition (LPCVD) (Plimpton and Bartel, 1993), the computer hard disk slider air bearing (Alexander *et al.*, 1994) and the micro-electro-mechanical-system (MEMS) (Piekos and Breuer, 1996; Nance *et al.*, 1998; Wu and Tseng, 2001), to name a few. Each of these distinct applications of high Knudsen number flows is now of practical scientific and engineering importance.

*Corresponding author. E-mail: chongsion@cc.nctu.edu.tw

ISSN 1061-8562 print/ISSN 1029-0257 online © 2003 Taylor & Francis Ltd
DOI: 10.1080/1061856031000081497

While the Boltzmann equation may be more suitable for approximating high Knudsen number flows, attempts to solve it numerically have met with much less success than the Navier–Stokes equations due to the higher dimensionality (up to seven) of the Boltzmann equation and the difficulties of modeling the integral collision term. To circumvent the difficulty of a direct numerical solution of the Boltzmann equation, an alternative method known as Direct Simulation Monte Carlo (DSMC) was proposed by Bird in 1963, and the associated monograph was published in 1994 (Bird, 1994). In the past, DSMC was attacked on the basis that its solution does not represent the true solution of the Boltzmann equation although it was argued by Bird that the same assumptions applied to both methods. Not until about two decades ago was Nanbu (1982) able to demonstrate mathematically that the DSMC method is equivalent to solving the Boltzmann equation, and more recently, Wagner (1992) proved that the solution of the DSMC method converges to the Boltzmann equation.

Most applications of DSMC applied structured grids (Bird, 1994) in the physical space. It is much easier to program the code using structured grids; however, it requires tremendous problem specific modification. To alleviate such restriction, an unstructured grid system may be one of the best choices, although it may be computationally more expensive. Considering the advantages using unstructured grids, its use in DSMC is highly justified. For example, Boyd's group (Boyd *et al.*, 1996; Kannenberg, 1998) applied such a technique to compute the thruster plume produced by a spacecraft and found that the results were very satisfactory. In addition, Piekos and Breuer (1996) and Wu and Tseng (2001) used the similar gridding technology to compute several typical microscale flows.

The DSMC method has become a widely used computational tool for the simulation of gas flows in which molecular effects become important. The advantage of using a particle method under these circumstances is that a molecular model can be applied directly to the calculation of particle collisions, while the continuum methods use macroscopic averages to account for such effects. Therefore, particle methods can predict these effects with much higher accuracy. Also, it is the only viable tool for analyzing the gas flows in the transitional regime. Nevertheless, the main drawback of such a direct physical method is its high computational cost. That is why the DSMC method was only used for analyzing high Knudsen number flows (or transitional flows). For low Knudsen number gas flows near the continuum regime, the computational cost is prohibitively high, even with today's most advanced computer. Hence, it is important to increase the computational speed to extend the application range of the DSMC method.

Since the DSMC method is a particle-based numerical method, the movement of each particle is inherently independent of each other. The coupling between particles is only made through collision in the cells. Hence, as compared with other N–S equation-based numerical schemes, the DSMC method is nearly 100% in parallelism. Therefore, the implementation of the DSMC method on the parallel processors is highly justified.

In the past, several studies on parallel implementation of DSMC have been published (Furlani and Lordi, 1988; Ota and Tanaka, 1991; Nance *et al.*, 1994; Ota *et al.*, 1995; Matsumoto and Tokumasu, 1997) using static domain decomposition and structured mesh. Message passing was used to transfer molecules between processors and to provide the synchronization necessary for the correct physical simulation. Results showed reasonable speedup and efficiency could be obtained if the problem is sized properly to the number of processors.

Recently, Boyd's group (Dietrich and Boyd, 1996; Kannenberg, 1998) designed a parallel DSMC software named MONACO. In this code, unstructured grids were used to take advantage of the flexibility of handling complex object geometry. In addition, a new data structure was proposed to meet the specific requirement related to workstation hardware while maintaining high efficiency on supercomputers. Timing results showed the performance improvement on workstations and the necessity of load balancing for achieving high performance on parallel computers. A coordinate partitioning technique was used for domain decomposition. The authors also suggested that only decomposition along streamlines should be chosen to keep communication at a minimum. However, it may not always be possible for realistic flow problems. Maximum 32 processors of IBM SP-1 were used throughout the research.

Until very recently, a dynamic load balancing technique, using Stop At Rise (SAR) based on a degradation function, was used in conjunction with the parallel implementation of the DSMC method (Robinson, 1998; LeBeau, 1999). Parallel efficiency up to 90% was reported for 128 processors for flow over a sphere (LeBeau, 1999), although it is not clearly reported how they implemented the dynamic load balancing. However, a data mapping between the local (in each sub-domain) and the global (whole domain) cell numbers was required, which became very costly for a large simulation otherwise (Robinson, 1998).

In the current study, we will concentrate on static domain decomposition, which is much easier to implement, as compared with dynamic domain decomposition. However, the design of the current parallel implementation will take into account the future implementation of dynamic domain decomposition. Static decomposition methods fall into two categories as pointed out by Robinson (1998). These are geometry-based and graph-based. Geometry-based methods use spatial (or coordinate) information of mesh to partition the domain. These methods are usually simple and fast but provide poor $E_c$ (edge cut) and poor load balancing. On the other hand, graph-based methods provide better

$E_c$ and load balancing if some careful measure is imposed. Many physical problems can be expressed within the framework of *graph theory*, such as discrete optimization problems and matrix reordering. One of the advantages in expressing the problem in terms of a graph is that each of the edges and vertices can be assigned a *weight* to account for the specific numerical application. For example, in DSMC, the vertex (cell centers) can be weighted with the particle numbers with all edges having unitary weight. Thus, domain decomposition in DSMC can become very efficient by taking advantage of the success in graph partitioning. Related descriptions and reviews of graph partitioning can be found in Kernighan and Lin (1970), Barnard and Simin (1994), Hendrickson and Leland (1994), Karypis and Kumar (1995), Vanderstraeten and Keunings (1995), Walshaw *et al.* (1995), Vanderstraeten *et al.* (1996) and Robinson (1998), and are not repeated here. There are, however, two graph partitioners available in the public domain worthy of mentioning, and they are described in the following.

METIS (Karypis and Kumar, 1995), developed at the University of Minnesota, was a variant of the multi-level scheme of graph partitioning. The idea of multi-level schemes draws from the multi-grid techniques, which are used to speed up the convergence in CFD problems. This algorithm successively contracts the original graph into several levels and computes the Fiedler vector on the coarsest contracted graph. This vector is then interpolated between the levels of the graph with some local refinement at each level. Reported performance was impressive in terms of *cpu* time. Another variant of the multi-level scheme, JOSTLE (Vanderstraeten and Keunings, 1995; Walshaw *et al.*, 1995; Vanderstraeten *et al.*, 1996), uses an initial domain decomposition (generated by greedy partitioning) and successively adjusts the partition by moving vertices lying on partition boundaries. In this method, vertex shedding is localized since only the vertices along the partition boundaries are allowed to move, not the vertices anywhere in the domain. Hence, this method possesses a high degree of concurrency and has the potential of being applied in the dynamic domain decomposition in the event of load imbalance across the processor array. In the current study, both METIS and JOSTLE are used to produce required domain decomposition for the DSMC simulation.

Based on previous reviews, the development of the parallel DSMC method did not take the advantage of the great success in graph partitioning. Considering DSMC, a truly dynamic load balancing technique is required because the load (approximately proportional to the particle numbers) in each sub-domain changes frequently, especially during the transient period. However, such implementation is definitely not an easy task to accomplish. As a first step, we instead use the static decomposition using METIS (Karypis and Kumar, 1995) and JOSTLE (Walshaw *et al.*, 1995), respectively, considering the particle weight in each sub-domain obtained from the simulation with very few simulated particles (e.g. less than 10%).

The objectives of the current study are summarized as follows.

1. To complete a two-dimensional parallel DSMC code using quadrilateral unstructured mesh, using a high-speed driven cavity flow as the benchmark problem, using a mesh renumbering technique to relieve the memory burden.
2. To utilize and compare the parallel performance of different techniques of static domain decomposition, including the simple coordinate partition without considering the workload in each sub-domain, the multi-level scheme (METIS) and the two-step method (JOSTLE) by considering the estimated particle weighing in each sub-domain.
3. To apply the parallel DSMC implementation to compute a 2-D realistic, near-continuum hypersonic flow over a cylinder as well as a 3-D hypersonic flow over a sphere and compare with previous experimental and DSMC data available in the literature.

The paper begins with descriptions of parallel DSMC method. Results of parallel performance are then considered using a high-speed cavity flow and applying to a 2-D near-continuum hypersonic flow over a cylinder and a 3-D hypersonic sphere flow in turn.

## THE PARALLEL DSMC METHOD

The DSMC method is a particle method for the simulation of gas flows. The gas is modeled at the microscopic level using simulated particles, each of which represents a large number of physical molecules or atoms. The physics of the gas are modeled through uncoupling of the motion of particles and collisions between them. Mass, momentum and energy transport are considered at the particle level. The method is statistical in nature. Physical events such as collisions are handled probabilistically using largely phenomenological models, which are designed to reproduce real fluid behavior when examined at the macroscopic level.

### Conventional DSMC Method

Since Bird (1994) has documented in detail the conventional DSMC method in his monograph, it is only briefly described here. Important steps of the DSMC method include setting up the initial conditions, moving all the simulated particles, indexing (or sorting) all the particles, colliding between particles, and sampling the molecules within cells to determine the macroscopic quantities. This method is essentially a computer simulation of gas molecular dynamics and depends heavily upon pseudo-random number sequences

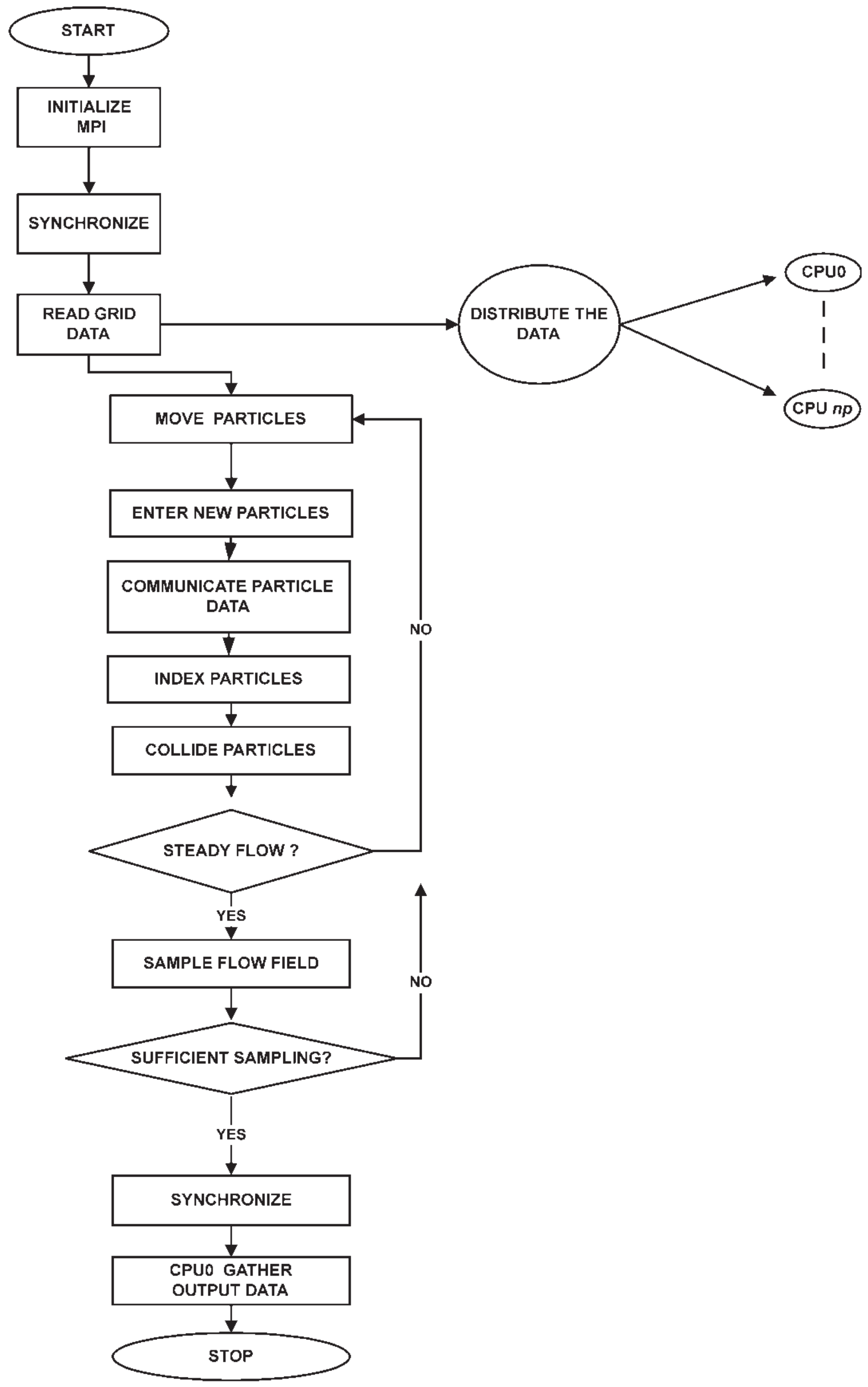

FIGURE 1 Simplified flow chart of the parallel DSMC method for np processors.

for simulating the statistical nature of the underlying physical processes. The data variables are randomly accessed from computer memory. Thus, it is very difficult to vectorize the DSMC code. However, since the movement of each particle and the collision in each cell is treated independently, this makes DSMC perfectly suitable for parallel computation, which is introduced next.

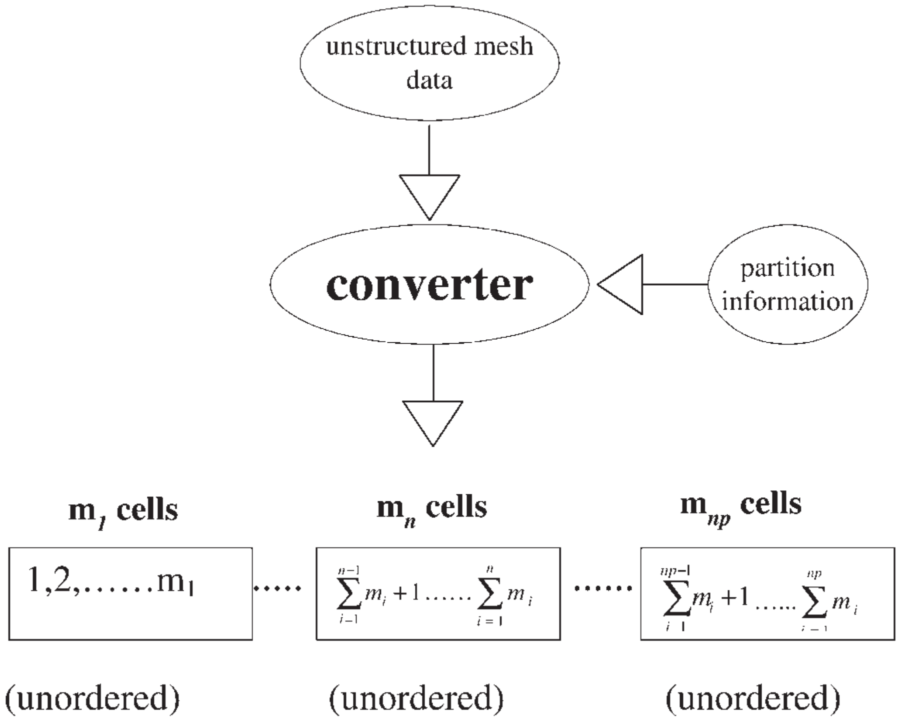

FIGURE 2 Schematic diagram of the proposed cell numbering scheme. $m_n$ is the number of cells in the $n$th processor, where the starting and ending cell number are $\sum_{i=1}^{n-1} m_i + 1$ and $\sum_{i=1}^{n} m_i$, respectively. np is the total number of processors.

### Parallel DSMC Method

The DSMC algorithm is readily parallelized through the physical domain decomposition. The cells of the computational grid are distributed among the processors. Each processor executes the DSMC algorithm in serial for all particles and cells in its own domain. Parallel communication occurs when particles cross the domain (processor) boundaries and are then transferred between processors. Figure 1 shows a simplified flow chart of the parallel DSMC method proposed in the current study. Note that CPUs are numbered from 0 to np in the figure.

In this method, a new approach of handling the cell-related data is proposed. A fully-unstructured quadrilateral (or triangular) mesh is first constructed using the advancing front method by a commercial code, *HyperMesh*™ (Liu, 2002). Then, a preprocessor (or "converter") is used to convert the fully-unstructured mesh data into the *globally sequential but locally unstructured* mesh data for each processor in conformation with the partitioning information from graph partitioners (e.g., METIS or JOSTLE), as schematically presented in Fig. 2. For example, the partition information from JOSTLE (based on the estimated weights in each cell) provides the cell numbers ($m_n$ for the $n$th subdomain, where $n = 1$ to np) and mapping of cells in each partitioned subdomain. After the conversion, the cells in each subdomain are renumbered such that the corresponding global starting and ending cell numbers for the nth subdomain are $\sum_{i=1}^{n-1} m_i + 1$ and $\sum_{i=1}^{n} m_i$, respectively. In each processor, the cell numbering is unordered (unstructured), but both the starting (smallest) and ending (largest) cell numbers increase sequentially with processor number. We term this as "globally sequential but locally unstructured". Thus, in each processor the memory is only needed to record the starting and ending cell numbers. The mapping between global and local cell data, however, can be easily obtained by a simple arithmetic operation due to this special cell-numbering design. The required array size for cell-related data is approximately the same as the cell numbers in each subdomain, rather than the total cell numbers in the simulation domain as in Robinson (1998). For example, if there are one million cells totally in the simulation with 100 processors, each processor will only require to store the array on the size of 10,000. The memory cost reduction will be at least 100 times in this case. This simple conversion dramatically reduces the memory cost otherwise required for storing the mapping between the local cell number in each processor and the global cell number in the computational domain if unstructured cells are used (Dietrich and Boyd, 1996; Kannenberg, 1998; Robinson, 1998). In addition, a processor neighbor-identifying array is created for each processor from the output of the converter, which is used to identify the surrounding processors for those particles crossing the inter-processor boundaries during simulation. From our practical experience, the maximum numbers of processor neighbor are less than 10;

therefore, the increase of memory cost due to this processor neighbor-identifying array is negligible. The resulting *globally sequential but locally unstructured* mesh data is then imported into the parallel DSMC code.

After reading the mesh data on a master processor (cpu 0), the mesh data are then distributed to all other processors according to the designated domain decomposition. All the particles in each processor then start to move as in the sequential DSMC algorithm. The particle-related data are sent to a buffer and are numbered sequentially when hitting the inter-processor boundary (IPB) during its journey within a simulation time step. After all the particles in a processor are moved, the destinating processor for each particle in the buffer is identified via a simple arithmetic computation, owing to the previously mentioned approach for the cell numbering, and are then packed into arrays. Considering communication efficiency, the packed arrays are sent as a whole to their surrounding processors in turn based on the tagged numbers recorded earlier. Once a processor sends out all the packed arrays, it waits to receive the packed arrays from its surrounding processors in turn. This "send and receive" operation serves practically as a synchronization step during each simulation time step. Received particle data are then unpacked and each particle continues to finish its journey for the remaining time step. The above procedures are repeated twice since there might be some particles that cross the IPB twice during a simulation time step.

After all particles on each processor have come to their final destinations at the end of a time step, the program then carries out the indexing of all particles and the collisions of particles in each computational cell in each processor as usual in a sequential DSMC code. The particles in each cell are then sampled at the appropriate time.

High parallel efficiency can only be achieved if communication is minimized and the computational load is evenly distributed among processors. To minimize the communication for static domain decomposition, the boundaries between subdomains should lie along the streamlines of the flow field (Dietrich and Boyd, 1996) as mentioned previously; however, it is nearly impossible to achieve this partition for most practical flows. Fortunately, the advancement of networking speed has reduced the communication time between processors to a tolerable amount. For the DSMC algorithm, the workload (or equivalently particle numbers) in each processor changes frequently, especially during the transient period of a simulation, while the workload attains a roughly constant value during steady-state sampling. Thus, a truly dynamic (or adaptive) domain decomposition technique is required to perfectly balance the workload among the processors. However, as a first step towards this objective, we have instead adopted the static domain decomposition method to provide the partitioned sub-domains. In addition to the simple coordinate partition, we use two state-of-the-art graph partitioning tools, JOSTLE

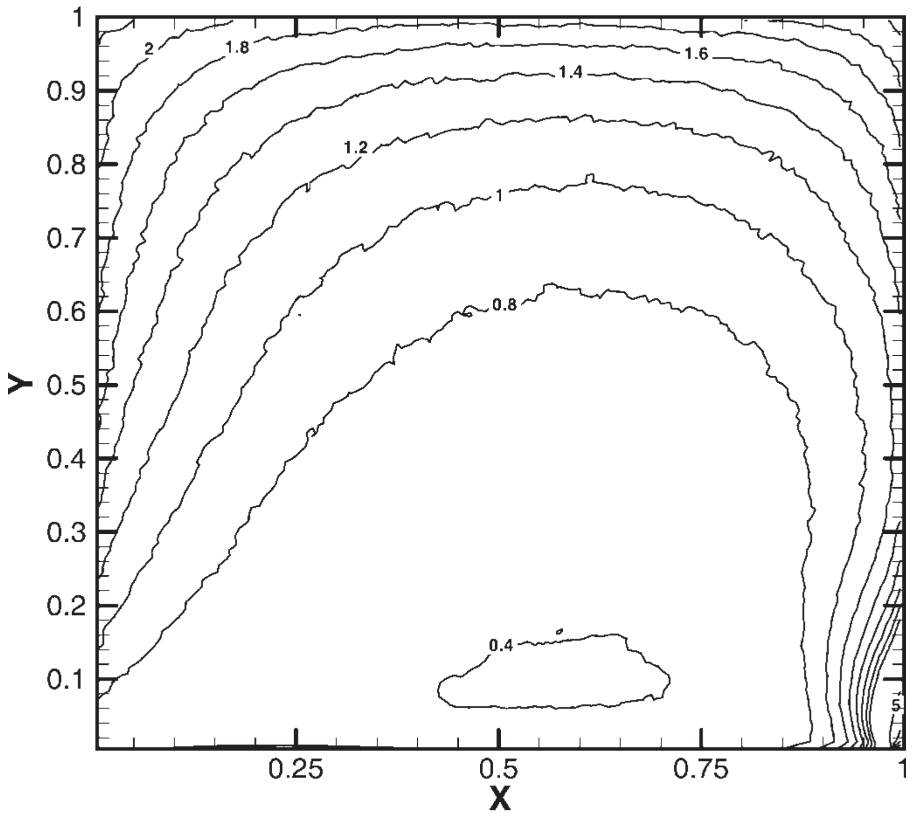

FIGURE 3 Normalized density contour of a high-speed driven cavity flow ($Kn_\infty = 0.04$, 10,000 cells, 130,000 particles).

(Walshaw *et al.*, 1995) and METIS (Karypis and Kumar, 1995), by considering the estimated particle weight on each vertex (cell center). This represents the first application of the graph partition techniques in DSMC to the best of the author's knowledge. The weight on each vertex is estimated by running the sequential DSMC code with about 10% or fewer (for some cases only 3% is used) of the total particle number, which is used for a real parallel simulation. The variations of weight on vertices obtained this way can be shown later to be roughly equivalent to those obtained by running all particles (100%).

The current parallel code, in the Single Program Multiple Data (SPMD) paradigm, is implemented on the IBM-SP2 machines (distributed memory system) using message passing interface (MPI) to communicate information between processors. Therefore no code modification is essentially required to adapt to other parallel machines (e.g. IBM-SP2, PC-clusters) with similar distributed memory system once they use the same MPI libraries for data communication.

### Parallel Performance Evaluation

Two parameters often used to measure the performance of the parallel implementation are *speedup* and *efficiency*. The same applies to the parallel DSMC method as well. These two parameters are defined as follows.

Speedup is defined as the ratio of the required running time for a particular application using *one* processor to that using *N* processors, i.e.

$$\text{Speedup} = \frac{t_1}{t_N}. \quad (1)$$

Efficiency is then defined as

$$\text{Efficiency} = \frac{\text{Speedup}}{N} \quad (2)$$

and is just the ratio of the true speedup to the ideal speedup, *N*, and hence its value lies between zero and one.

Although the DSMC possesses nearly 100% parallelism (except for initialization and final output), both the values of speedup and efficiency are expected to be lower than the ideal values due to the load imbalance and communication as mentioned previously.

## BENCHMARK TESTS

### Benchmark Problem

In order to test the proposed parallel DSMC algorithm, we use a 2-D, high-speed driven cavity flow as the benchmark problem (Robinson, 1998). This benchmark problem is a 2-D variation of the familiar Rayleigh problem, consisting of a square box whose wall temperatures are all set to 300 K and diffusive (100% full thermal accommodation), and lower wall velocity is eight times the most probable speed based on wall temperature. Initial gas temperatures within

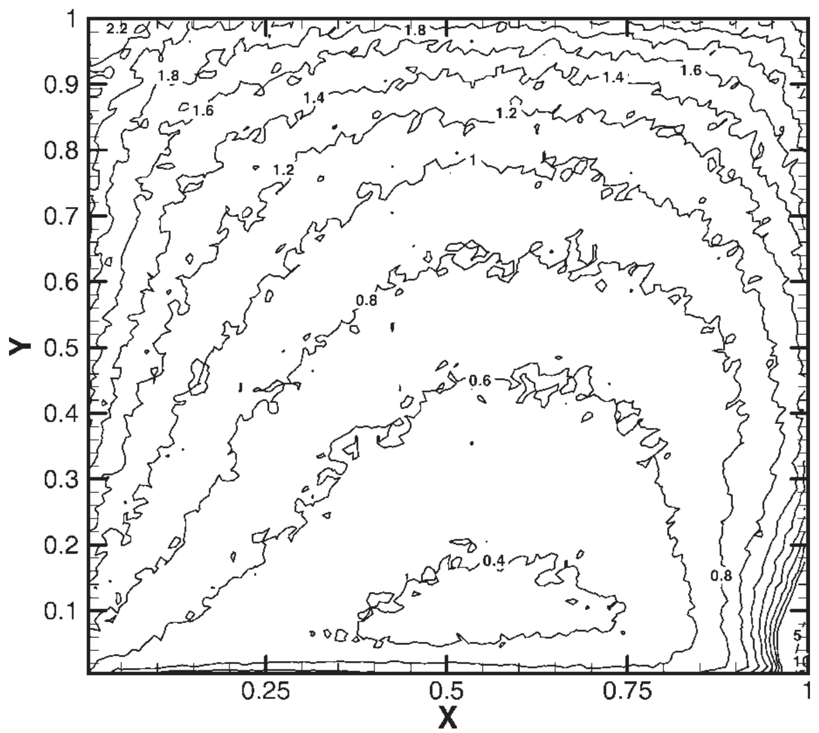

FIGURE 4 Estimated normalized density contour of a high-speed driven cavity flow ($Kn_\infty = 0.04$, 10,000 cells, 10,000 particles).

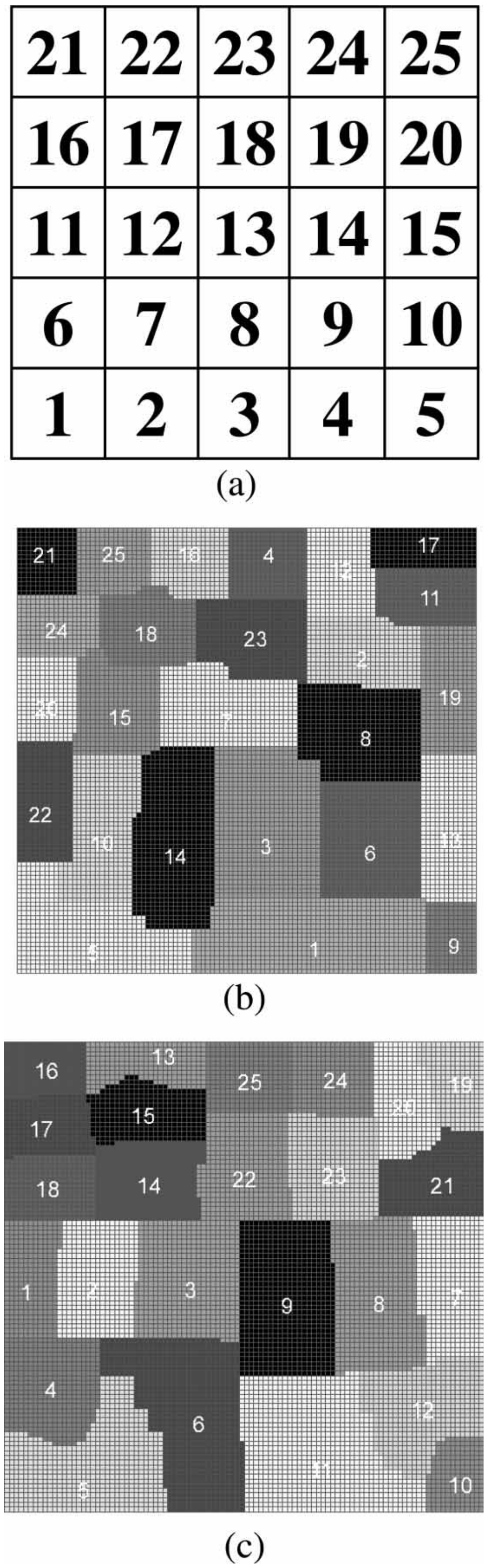

FIGURE 5 Static domain decomposition using three different partition schemes for 25 processors: (a) simple coordinate; (b) JOSTLE; (c) METIS.

the enclosure are set to 300 K, which is the same as the wall temperature. The corresponding Knudsen number is based on the initial uniform mean free path and length of the square box equal to 0.04.

## Results of the Sequential Code

In total 10,000 uniform quadrilateral cells and 130,000 particles are used in the serial simulation of the high-speed driven cavity flow. The time step is chosen as smaller compared with mean collision time. Results of normalized density plots are shown in Fig. 3. Note that the dimensions in $x$ and $y$ directions are also normalized with the width of the cavity. The results show that an ultra high-density region, up to 26.2 times the initial density, appears at the very right-hand bottom corner due to the high-speed moving plate at the bottom of the cavity. Also the densities at the two top corners are higher than the initial value. In addition, most of the region above the moving plate is rarefied as compared with the initial value since the particles are "entrained" to collide with the moving plate. The above sequential results are used in the baseline solution for the parallel computation discussed later.

## Domain Decomposition

In the current study, three static domain decomposition methods are used to partition the computational domain. These include the simple coordinate partition without considering the particle weights in each subdomain, the two-step multi-level method (JOSTLE) and the multi-level scheme (METIS), both considering the approximate particle weights in each subdomain estimated by running the sequential code with 10% of the total particles or fewer. A typical normalized density contour with averaged one particle per cell is shown in Fig. 4. The general trend is similar to that obtained by 100% particles; however, the finer structure exhibits large differences. Thus, we would expect that the load imbalance will be worsened for increasing processor numbers due to this difference. The results exhibit large statistical scatters as expected; however, by comparing Figs. 3 and 4 the approximation seems justified. For example, we can obtain the approximate particle weights with about $10^5$ particles or fewer, if the real simulation particle number is on the order of $10^6$, which is still considered very time-consuming on modern workstations or personal computers. The resulting domain decompositions are described as follows.

Typical results of domain decomposition for 25 processors using simple coordinate, JOSTLE and METIS schemes are presented in Figs. 5a–c, respectively. For simple coordinate partition (Fig. 5a), each subdomain possesses the same number of cells. IPBs are straight, which makes the handling of particle tracking across the IPB more efficient. However, the load imbalancing is expected to be the worst. Note that the resulting numbering in each subdomain remains the same for future discussion. A typical graph partition resulting from JOSTLE is shown in Fig. 5b for 25 processors. For example, the processor 9 on the right-hand bottom corner possesses far fewer cells than its adjacent

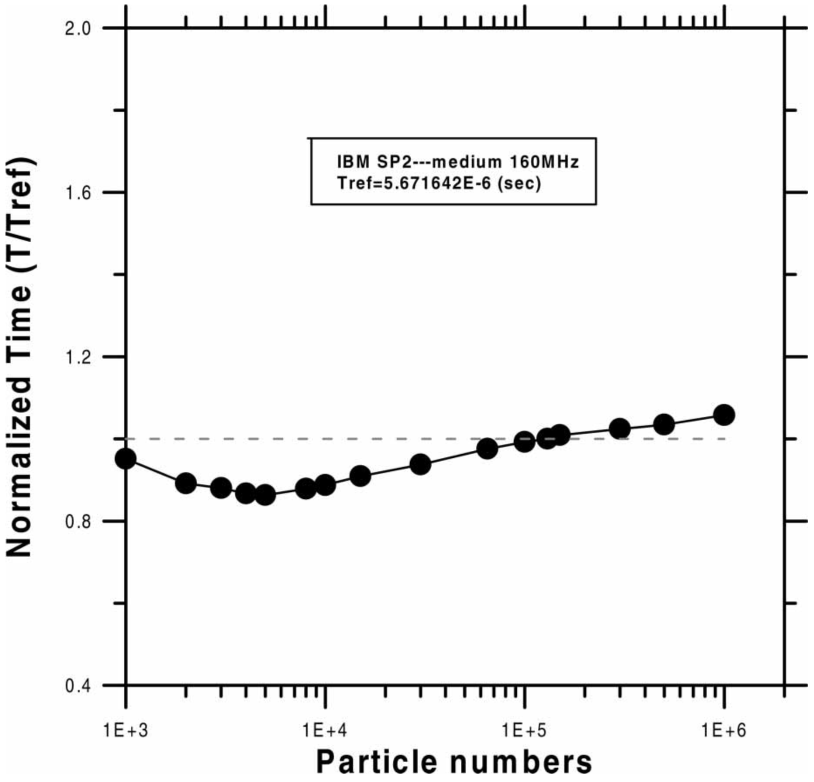

FIGURE 6 Normalized computational time per particle at different particle numbers ($T_{\rm ref} = 5.671642\mathrm{E} - 6$ s for 130,000 particles) (uniform flow, $20 \times 20$ cells).

processor 1 due to the ultra high density in the corner. Another typical graph partition resulting from METIS for 25 processors is presented in Fig. 5c. Obviously, the resulting partition looks very different as compared with that of JOSTLE. In general, the number of resulting edge cuts from the JOSTLE is smaller than that from METIS due to the two-step method used to smooth the IPBs. Comparing the results of three different partitioning schemes, we would expect that the JOSTLE provides the best partitioning quality for the DSMC computation considering the communication cost required for parallel computation.

## Parallel Performance Evaluation

### *Cache Effects due to Different Particle Numbers*

Before proceeding to the discussion of the parallel performance results, the cache effects of the superscalar workstation (IBM-SP2 in the current study) are magnified by running the sequential code with a uniform flow over a uniform mesh ($20 \times 20$ cells) with different particle numbers. It is well recognized that the simulation time is approximately proportional to the particle numbers if the cell number remains the same and is relatively small as compared with total particle numbers. Results of normalized computational time per particle (to the time per particle required by $1.3\mathrm{E} + 5$ particles) are presented in Fig. 6 for the particle numbers ranging from $1\mathrm{E} + 3$ to $1\mathrm{E} + 6$. Minimum normalized computational time per particle occurs at a particle number of about 5000. Results clearly demonstrated the phenomenon of *cache misses* as particle number increases. The normalized computational time per particle increases from 0.86 to 1.08 as the particle number increases from $5\mathrm{E} + 3$ to $1\mathrm{E} + 6$. Interestingly, the normalized value for $1\mathrm{E} + 3$ particles increases instead up to 0.95, which should be attributed to the overhead required for loading the particle data from RAM into the cache. Thus, the parallel performance is expected to deteriorate as the particle number on a processor decreases down to 1000 as compared with 5000. This will somehow restrict the scalability of the parallel DSMC method on superscalar machines. The concept of *cache misses* is, however, important in explaining some parallel performance results later.

### *Speedup and Efficiency*

The results of speedup and efficiency of the parallel DSMC computation for fewer than or equal to 25 processors, using different partitioning techniques, are presented in Figs. 7 and 8, respectively. For the simple coordinate partition, the speedup (efficiency) of 25 processors is 13.6 (54.5%), while it increases to 15.6 (62.5%) for the two-step method (JOSTLE) and 15.5 (62.2%) for the multi-level scheme (METIS). Note that the percentage number appearing in the parenthesis right after the speedup represents the corresponding parallel efficiency. For processor numbers

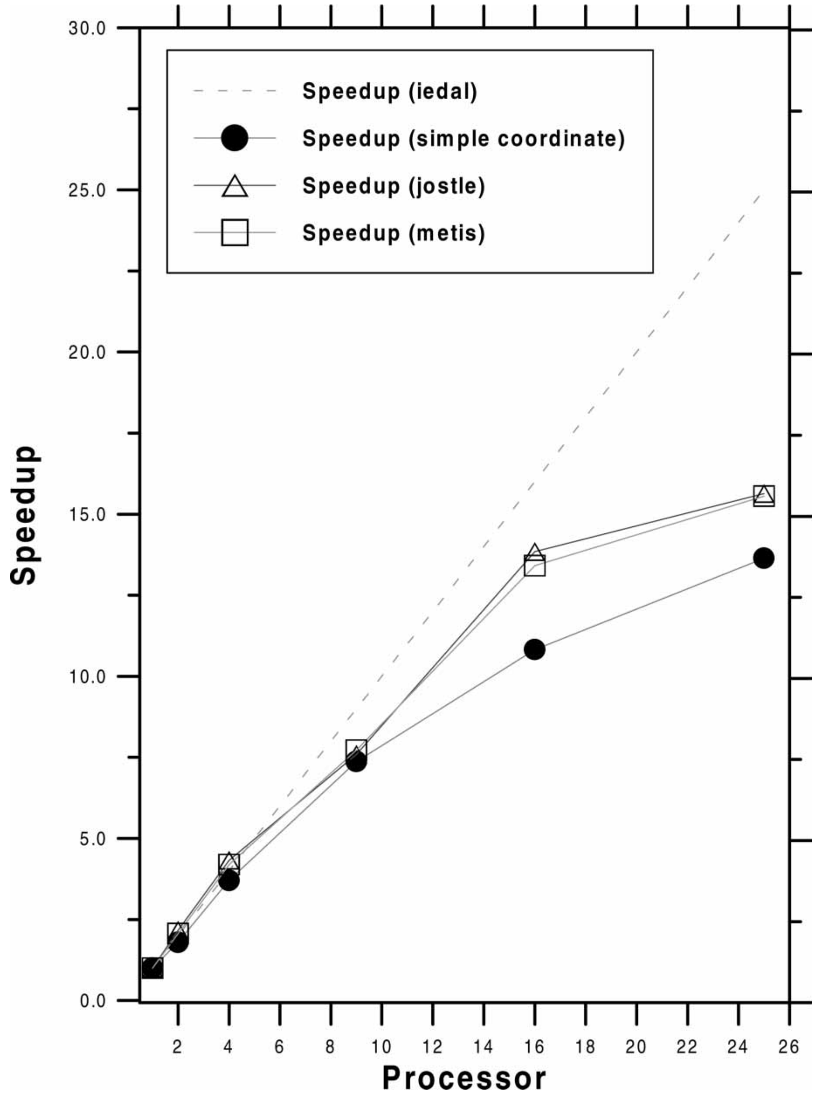

FIGURE 7 Speedup of parallel DSMC computation for the high-speed driven cavity flow ($\leq$ 25 processors) using different partitioning techniques.

fewer than or equal to 4, super-linear speedup occurs clearly, as shown in both Figs. 7 and 8, using METIS and JOSTLE. This is attributed to the cache effects discussed earlier, which make the computational time per particle less as the particle number in each CPU decreases. As the processor number increases more than 4, this super-linearity disappears due to the increasing ratio of communication to computation, and load imbalance among processors. It seems that the speedup, as shown in Fig. 7, begins to level off for processor numbers greater than 16, especially for the results using JOSTLE and METIS. This means that the scalability of currently proposed static partition using estimated number density becomes worse for increasing processor numbers due to the incorrect finer structure mentioned earlier. The reasons causing the deteriorating parallel speedup and efficiency for higher processor numbers can be demonstrated by the detailed analysis of the workload variations among the processors, as described in the following.

### *Workload and Communication Analysis*

The workload and communication variation among processors plays an important role in affecting the parallel performance. Figure 9 presents the particle number (at steady state), workload and communication for 25 processors using the simple coordinate partition. Note that the communication time actually includes the idle time. All quantities are normalized to the actual workload of first processor in each partition. As stated previously, the actual workload strongly correlates with the steady-state particle number in each processor. The same trend applies to other partitions. This proves that the computational time is approximately proportional to the particle number. Note that the communication time

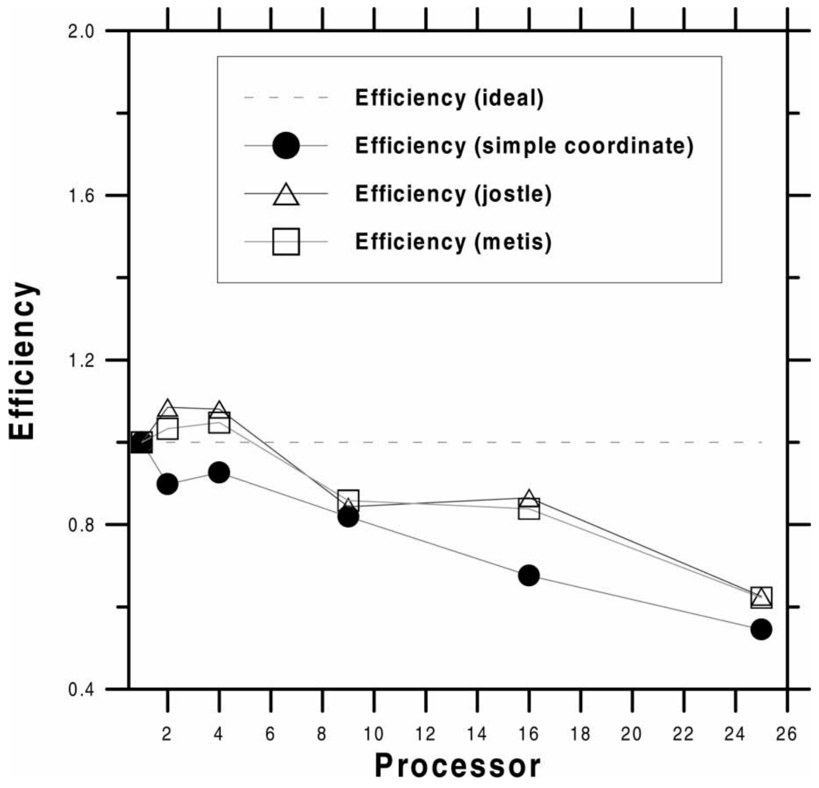

FIGURE 8 Efficiency of parallel DSMC computation for the high-speed driven cavity flow (≤ 25 processors) using different partitioning techniques.

becomes large for the processors with fewer particles due to the large idle time included.

Similar results are shown in Fig. 10 (25 processors) for two-level partitioning using JOSTLE. The actual workload distributes quite uniformly (0.75–1.0) among the processors and hence the communication (0.05–0.35 of the workload with respect to processor one) is small and relatively uniform as compared with the case of simple coordinate partition. There is still one processor (cpu 1) doing too much work, which causes the increase of communication time (including idle time) for other processors. This is not unexpected since the partition is based on a very rough estimation of particle weighing by running with $\leq 10\%$ particles. Nevertheless, the overall parallel performance using JOSTLE is superior to that using the simple coordinate partition. Results for partition using METIS are not shown here, but we found the similar trend for the workload and communication analysis; however, the overall parallel performance is a little worse than that using JOSTLE due to the jig-jagged like IPB produced using METIS.

## APPLICATION TO A REALISTIC FLOW

The proposed parallel DSMC implementation has been verified by the 2-D cavity flow stated previously with 25 processors or fewer. We have concluded that the JOSTLE partition is the best considering the parallel performance. Thus, to demonstrate the powerful capability of the current parallel implementation, we have applied it using the JOSTLE partition to compute a 2-D realistic, near-continuum hypersonic flow over a cylinder (Bütefsch, 1969; Koura and Takahira, 1996) as well as a 3-D, hypersonic rarefied gas flow past a sphere (Russell, 1968; Liu, 2002), respectively. The results are then compared with previous simulations (Koura and Takahira, 1996; Liu, 2002) and experimental studies (Russell, 1968; Bütefsch, 1969). Note that the discussion of physics of flow field is brief since we are only interested in demonstrating the powerful computational capability of the current DSMC implementation in rarefied gas dynamics.

### 2-D Cylinder

#### *Flow and Simulation Conditions*

Flow conditions are the same as those of Koura and Takahira (1996) and represent the experimental conditions of Bütefsch (1969). For completeness, they are briefly described here as follows: VHS nitrogen gas, free-strem Mach number $M_\infty = 20$, free-stream number density $n_\infty = 5.1775 \times 10^{19}\,\text{particles/m}^3$, free-stream temperature $T_\infty = 20\,\text{K}$, fully thermal accommodated and diffusive cylinder wall with $T_w/T_0 = 0.18$, where $T_w$ and $T_0$ are the wall and stagnation temperatures, respectively. A temperature-dependent rotational energy exchange model of Parker (Nanbu, 1982) is used to model the diatomic nitrogen gas. The resulting Knudsen number based on the free-stream condition is 0.025. A total of

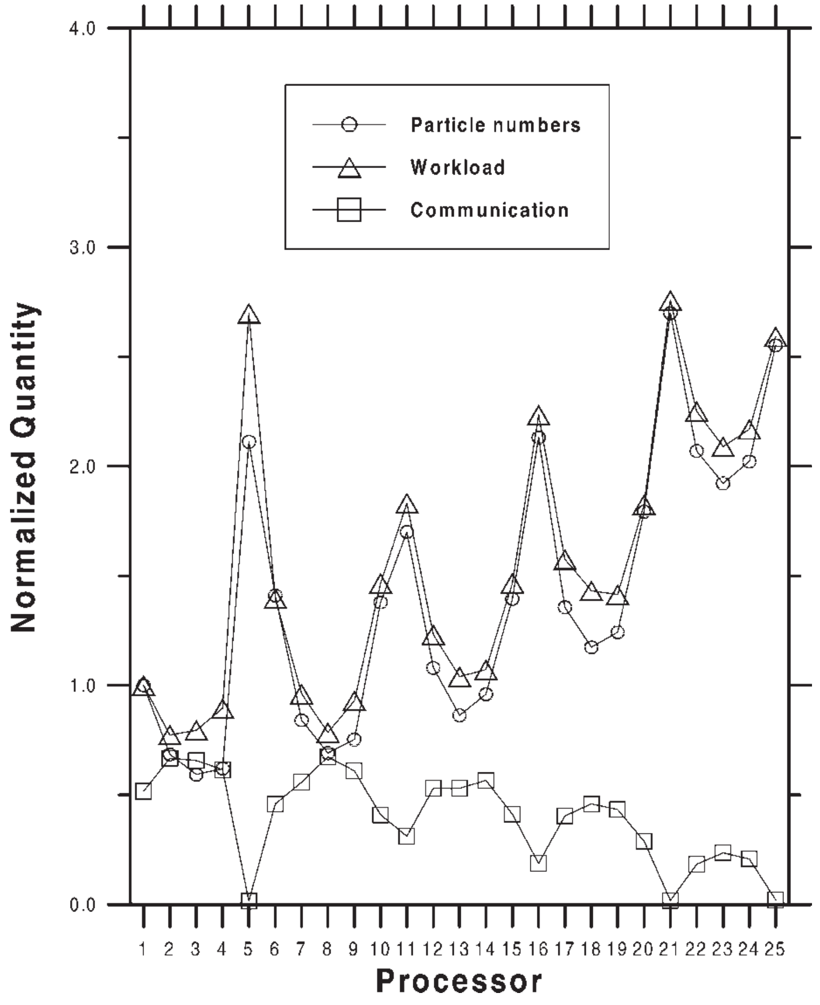

FIGURE 9 Workload distribution for the high-speed driven cavity flow using simple coordinate partition (25 processors).

32,319 quadrilateral cells and about 2 million particles are used for the simulation. Computational mesh distribution is shown in Fig. 11, which shows higher mesh density near the stagnation region and lower mesh density in the wake region. The domain decomposition using JOSTLE for 25 processors, estimated using approximately 32,000 particles (less than 1.6% of the final total particle number), is shown in Fig. 12. It clearly shows that the relatively smaller subdomains near the stagnation region and larger subdomains in the wake region.

### *Properties Along the Stagnation Line*

Results of normalized number density ($n/n_\infty$), and normalized translational and rotational temperatures ($(T - T_\infty)/(T_o - T_\infty)$) along the stagnation line are presented in Figs. 13 and 14, respectively. Note that the subscripts "o" and "∞" represent stagnation and free-stream properties, respectively. Previous experimental data of Bütefsch (1969) and the DSMC data of Koura and Takahira (1996) are also included in these figures. Note that the data of Koura and Takahira (1996) were obtained using the conventional DSMC method. First, for the density ratio, the present data agree well with both the available experimental data and simulated data in front of and behind the cylinder. The maximum density ratio (21.965) of the present study is, however, larger than that (~16) of Koura and Takahira (1996) due to the very refined mesh in the stagnation region. Second, for the translational temperatures along the stagnation line, the current simulation data compare reasonably well with those of Koura and Takahira (1996) (no experimental data are available). In addition, the present simulated data are in good agreement with those of Koura and Takahira (1996) in the wake region. Finally, for the rotational temperatures along the stagnation line, our data agree reasonably well with both the experimental (Bütefsch, 1969) and the simulated DSMC (Koura and Takahira, 1996) data.

## 3-D Sphere

### *Flow and Simulation Conditions*

Flow conditions past a 3-D sphere are the same as those of Liu *et al.* (2002) and represent the experimental

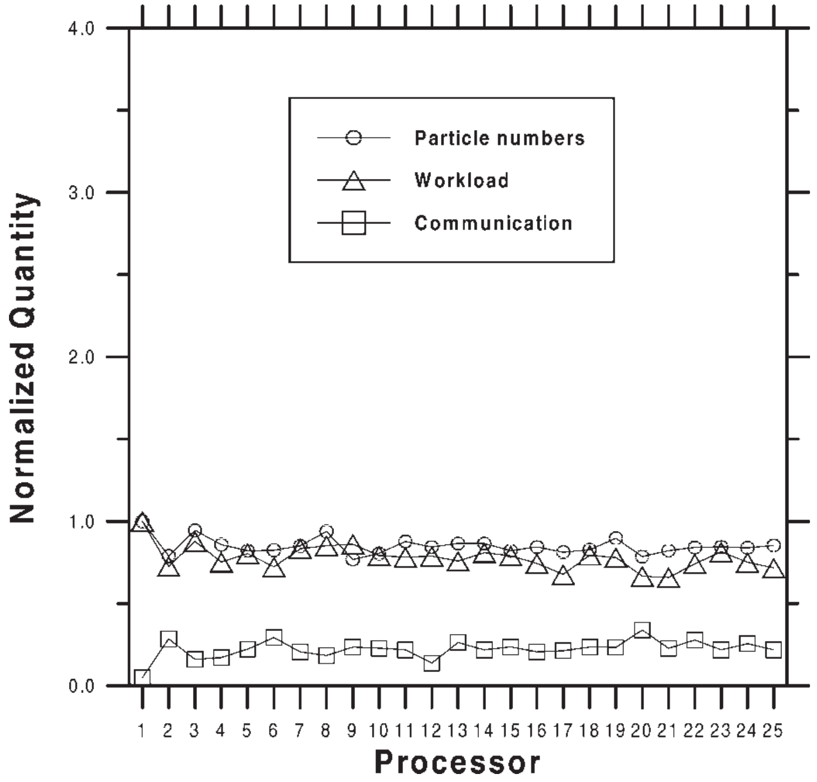

FIGURE 10 Workload distribution for the high-speed driven cavity flow using two-level partition (25 processors) (JOSTLE).

conditions of Russell (1968). The flow conditions are briefly described as follows: the sphere diameter $D = 1.28 \times 10^{-2}$ m, VHS nitrogen gas, free-stream Mach number $M_\infty = 4.2$, mean free path at free-stream $\lambda_\infty = 1.325 \times 10^{-3}$ m$^3$, fully thermal accommodated and diffusive sphere wall with $T_w/T_0 = 1$, where $T_w$ and $T_0$ are the wall and total temperatures and equal to 300 K, respectively. The resulting Knudsen number based on the free-stream condition ($\lambda_\infty/D$) is 0.1035. A total of 132,634 tetrahedral cells and about 2 million particles are used for the simulation. Computational mesh distribution and the surface processor distribution are shown in Figs. 15 and 16, respectively.

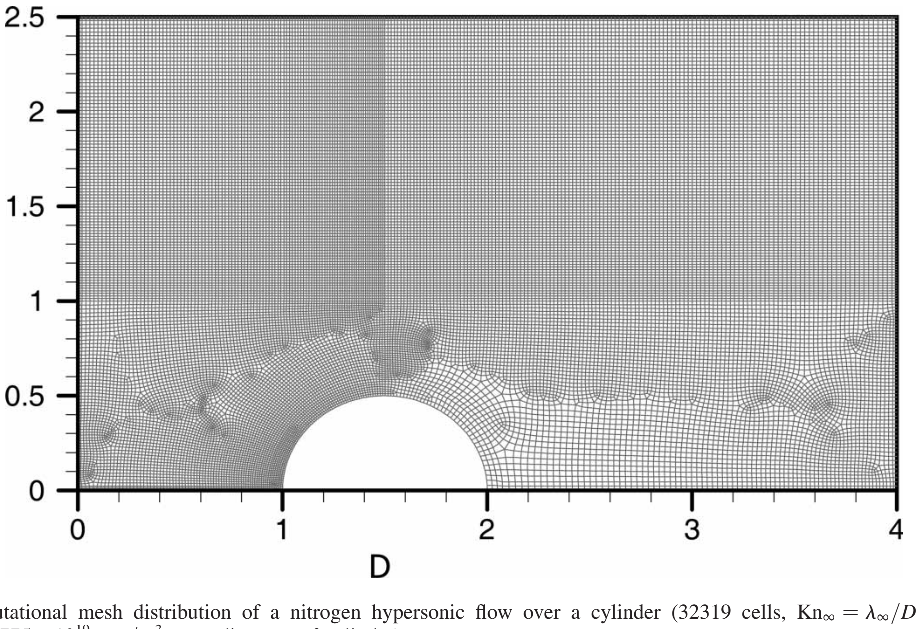

FIGURE 11 Computational mesh distribution of a nitrogen hypersonic flow over a cylinder (32319 cells, $\mathrm{Kn}_\infty = \lambda_\infty/D = 0.025$, $M_\infty = 20$, $T_\infty = 20$ K, $n_\infty = 5.1775 \times 10^{19}$ part/m$^3$, $D =$ diameter of cylinder).

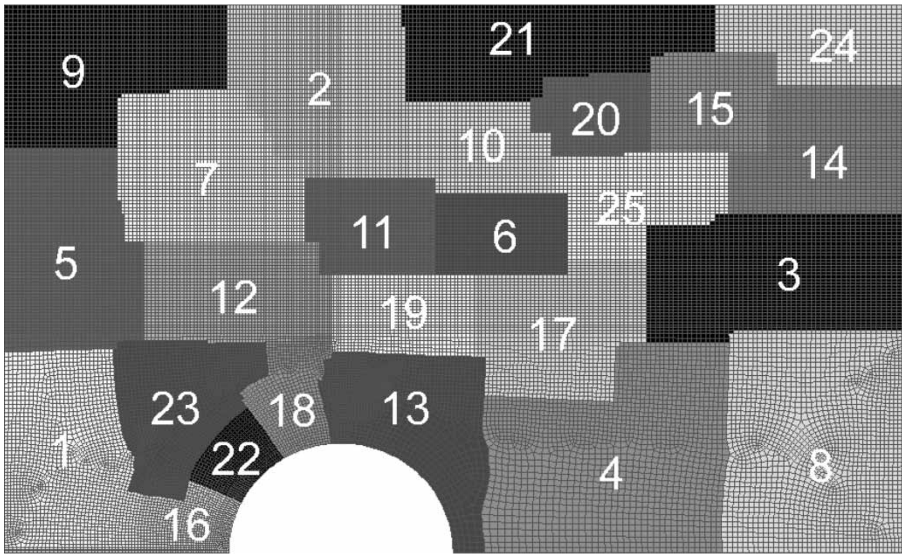

FIGURE 12 Domain decomposition for a hypersonic flow over a cylinder using JOSTLE (25 processors; $Kn_\infty = 0.025$).

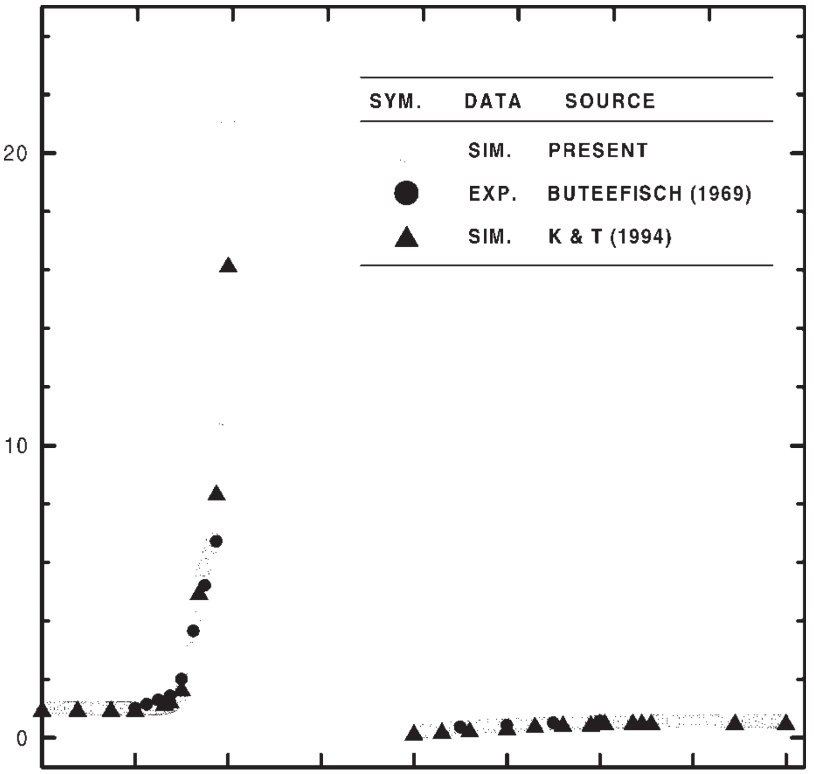

FIGURE 13 Normalized density along the stagnation line of a hypersonic flow over a cylinder (25 processors; $Kn_\infty = 0.025$).

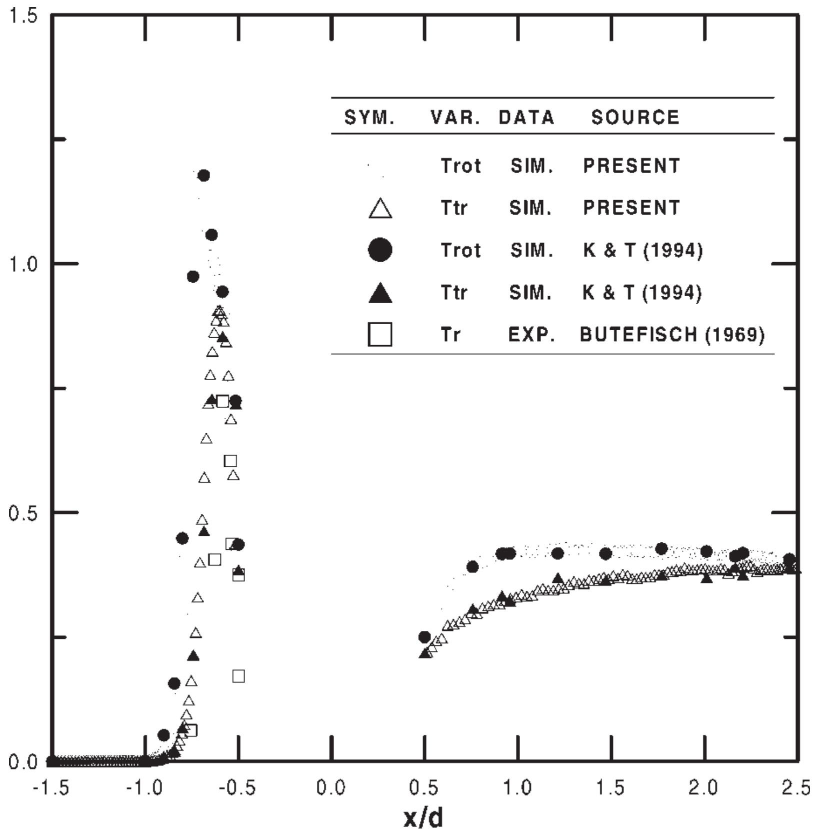

FIGURE 14 Normalized translational and rotational temperatures along the stagnation line of a hypersonic flow over a cylinder (25 processors; $Kn_\infty = 0.025$).

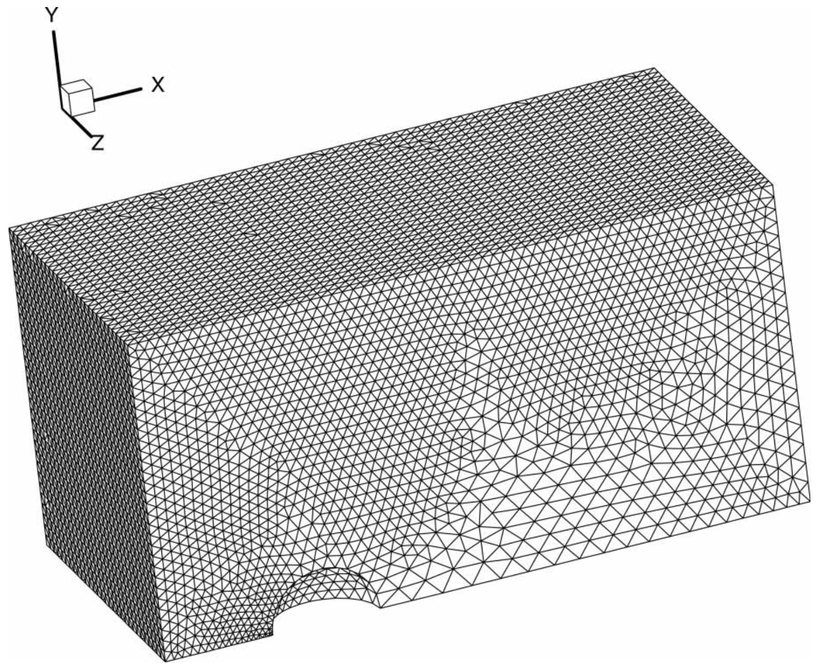

FIGURE 15 Mesh distribution for a hypersonic flow over a 3-D sphere (132,634 cells, $Kn_\infty = \lambda_\infty/D = 0.1035$, $M_\infty = 4.2$, $T_w = T_0 = 300\,K$, $D = 1.28 \times 10^{-2}$ m, $D =$ diameter of sphere).

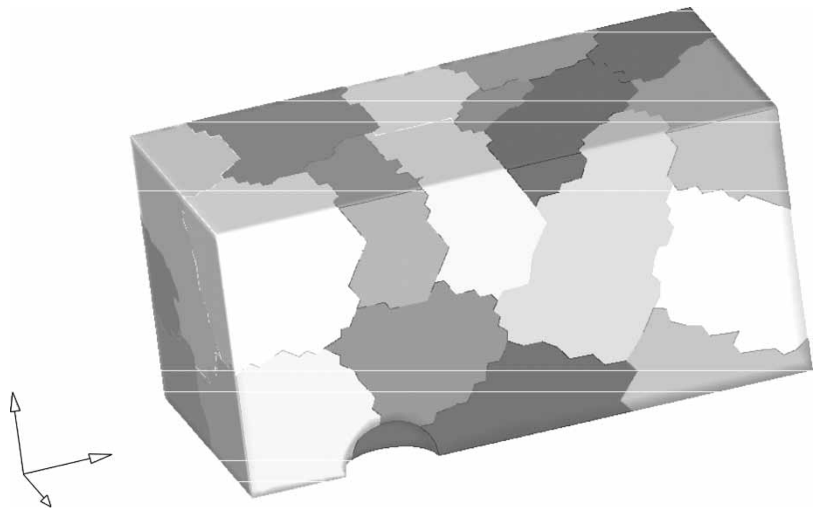
FIGURE 16 Surface processor distribution for a hypersonic flow over a 3-D sphere (25 processors; $Kn_\infty = 0.1035$).

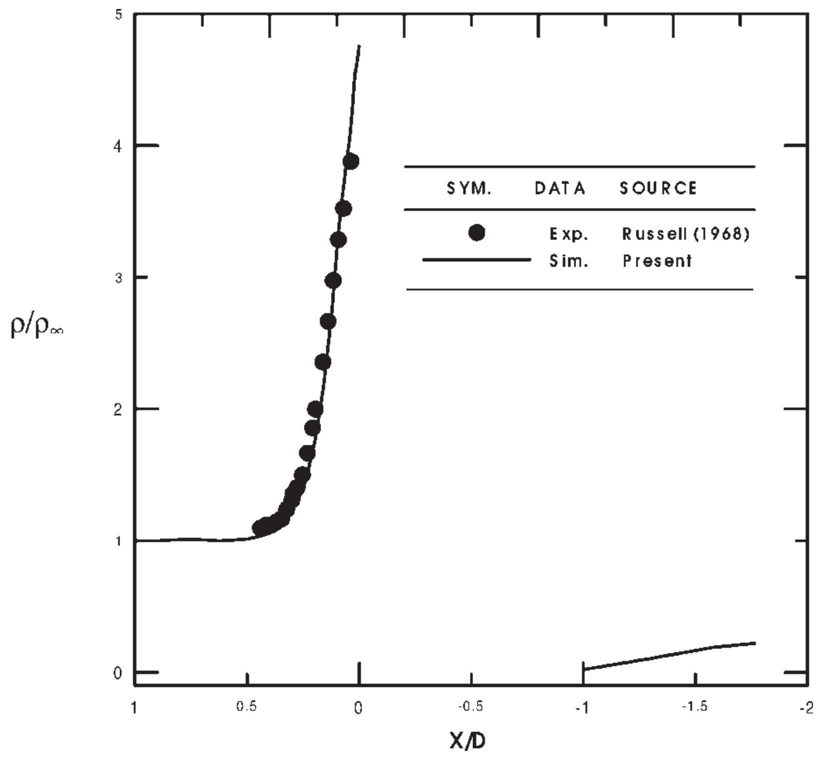

FIGURE 17 Normalized centerline density distribution for a hypersonic flow over a sphere (25 processors; $Kn_\infty = 0.1035$).

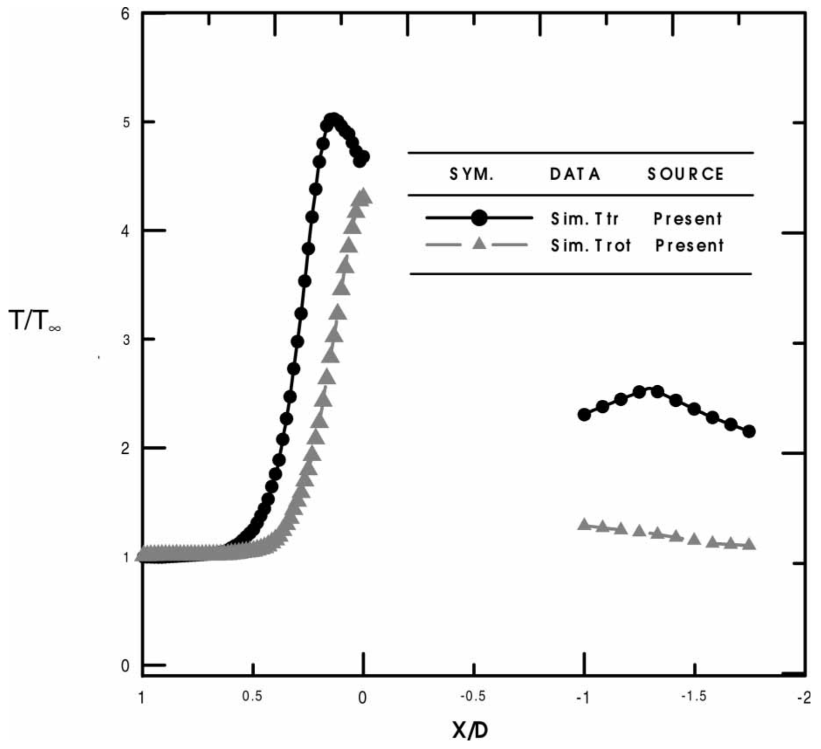

FIGURE 18 Normalized centerline temperature distribution for a hypersonic flow over a sphere (25 processors; $Kn_\infty = 0.1035$).

### *Properties Along the Stagnation Line*

Results of normalized centerline density ($\rho/\rho_\infty$), and normalized translational and rotational temperatures ($T/T_\infty$) distribution are presented in Figs. 17 and 18, respectively. Previous experimental data of Russell (1968) for the density ratio are also included in Fig. 17 for comparison. Note that there were no experimental data available behind the sphere. First, for the density ratio, the present simulation data agree very well within experimental uncertainties with the available experimental data in front of the sphere. The maximum density ratio of the present study is 4.8 and there are no experimental data behind the sphere. Second, the normalized temperatures along the stagnation line are shown in Fig. 18, although no experimental data are available. There exists strong thermal non-equilibrium in the bow-shock region in front of the sphere and in the wake region due to the strong rarefaction caused by the sphere. In addition, parallel efficiency is about 60% using 25 processors for this 3-D test case.

## CONCLUSIONS

In the current study, a parallel DSMC algorithm using unstructured mesh, which can be implemented on memory distributed parallel machines with MPI for the data communication, is proposed and tested via a high-speed driven cavity flow problem. Different static decomposition methods are utilized to distribute the workload among processors. These include the simple coordinate partitioning, the multilevel scheme and the two-step method, with the latter two methods considering the estimated particle weights. The results of parallel performance for these methods are then compared. Finally, a 2-D realistic flow and a 3-D hypersonic sphere flow are chosen to demonstrate the computational power of the current parallel implementation of the DSMC method.

In summary, the major findings of the current research are listed as follows.

1. The use of a *globally sequential but locally unstructured* approach reduces the memory cost dramatically, which would be otherwise expensive due to the mapping between the global and local cell data.
2. The study has demonstrated that significant speedups for multiprocessors (e.g. 14 for 16 processors using JOSTLE) can be obtained over a single-processor processing; however, the speedup begins to level off as the processor number is larger than 16 (e.g. 16 for 25 processors using JOSTLE).
3. A cheap method of estimating the particle weights on vertices, estimated by reduced total particle numbers (less than 10%), has been proved to do the job fairly well for the high-speed driven cavity problem ($\leqq 16$ processors in current case) as compared with the complicated dynamic load balancing technique.

4. By considering the particle weight on each vertex (cell center), the speedups of the two-step method (JOSTLE, Walshaw *et al.* (1995)) are slightly better than those of the multilevel scheme (METIS, Karypis and Kumar (1995)) up to 25 processors due to the smoother partitioning boundaries provided by JOSTLE. As expected, the performance of simple coordinate partitioning represents the worst case.
5. For fewer processors ( $\leqq$ 4 in driven cavity flow problem), super-linear appears due to cache effect are demonstrated. It is overwhelmed by load imbalancing and communication as the processor number increases.

The parallel method presented in this study utilizes static domain decomposition to distribute the workload among processors. It is well known that the workload (particle number) on each sub-domain changes very frequently in a DSMC simulation, especially during the transient stage. Thus, a truly adaptive domain decomposition method is required to dynamically distribute the workload among the processors and is currently in progress.

## Acknowledgement

This investigation was supported by the National Science Council of Taiwan, Grant No. NSC91-2212-E-009-045. The authors would like to express their sincere thanks for the computing resources provided by the National Center for High Performance Computing of the National Science Council of Taiwan.

## References

Alexander, F.J., Garcia, A.L. and Alder, B.J. (1994) “Direct simulation Monte Carlo for thin-film bearings”, *Physics of Fluids* **6**, 3854–3860.
Barnard, S.T. and Simon, H.D. (1994) “A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems”, *Concurrency Practice and Experience* **6**(2), 101–117.
Bird, G.A. (1994) Molecular Gas Dynamics and the Direct Simulation of Gas Flows (Oxford University Press, New York).
Boyd, I.D., Jafry, Y. and Beukel, J.W. (1996) “Particle simulation of helium microthruster flows”, *Journal of Spacecraft and Rockets* **31**, 271–281.
Bütefsch, K. (1969) Investigation of Hypersonic Non-equilibrium Rarefied Gas Flow Around a Circular Cylinder by the Electron Beam Technique, Rarefied Gas Dynamics II (Academic Press, New York), pp 1739–1748.
Dietrich, S. and Boyd, I.D. (1996) “Scalar and parallel optimized implementation of the direct simulation Monte Carlo method”, *Journal of Computational Physics* **126**, 328–342.
Furlani, T.R., Lordi, J.A. (1988) Implementation of the Direct Simulation Monte Carlo Method for an Exhaust Plume Flowfield in a Parallel Computing Environment, AIAA Paper No. 88–2736.
Hendrickson, B., Leland, R. (1994) The Chaco Users Guide: Version 2.0., Technical Report SAND 94-2692, Sandia National Laboratories.
Kannenberg, K.C. (1998) “Computational method for the direct simulation Monte Carlo technique with application to plume impingement”, Ph.D. Thesis, Cornell University (Ithaca, NY).
Karypis, G., Kumar, V. (1995) Metis: Unstructured Graph Partitioning and Sparse Matrix Ordering, Version 2.0 User Manual, Minneapolis MN55455, Computer Science Dept, University of Minnesota, U.S.A.
Kernighan, B.W. and Lin, S. (1970) “An efficient heuristic procedure for partitioning graphs”, *Bell Systems Technical Journal* **49**, 291–308.
Koura, K., Takahira M. (1996) Monte Carlo Simulation of Hypersonic Rarefied Nitrogen Flow Around a Circular Cylinder, 20th International Symposium on Rarefied Gas Dynamics, 1236–1242.
LeBeau, G.J. (1999) “A parallel implementation of the direct simulation Monte Carlo method”, *Computer Methods in Applied Mechanics and Engineering* **174**, 319–337.
Lee, Y.K. and Lee, J.W. (1996a) “Direct simulation of compression characteristics for a simple drag pump model”, *Vacuum* **47**, 807–809.
Lee, Y.K. and Lee, J.W. (1996b) “Direct simulation of pumping characteristics for a model diffusion pump”, *Vacuum* **47**, 297–306.
Liu, H., Fan, J., Shen, C. (2002) Validation of a Hybrid Grid Scheme of DSMC in Simulating Three-Dimensional Rarefied Gas Flows, 23rd International Symposium on Rarefied Gas Dynamics.
Matsumoto, Y. and Tokumasu, T. (1997) “Parallel computing of diatomic molecular rarefied gas flows”, *Parallel Computing* **23**, 1249–1260.
Nanbu, K. (1982) “Direct simulation scheme derived from the Boltzmann equation. I. Monocomponent gases”, *Journal of the Physical Society of Japan* **49**(5), 2042–2049.
Nance, R.P. (1995) “Monte Carlo simulation of three-dimensional hypersonic flows on parallel architectures”, Master Thesis, North Carolina State University.
Nance, R.P., Wilmoth, R.G., Moon, B., Hassan, H.A., Saltz, J. (1994) Parallel Solution of Three-dimensional Flow Over a Finite Flat Plate, AIAA Paper No. 94–0219.
Nance, R.P., Hash, D.B. and Hassan, H.A. (1998) “Role of boundary conditions in Monte Carlo simulation of microelectromechanical systems”, *Journal of Thermophysics and Heat Transfer* **12**, 447–449, Technical Notes.
Ota, M. and Tanaka, T. (1991) “On speedup of parallel processing using domain decomposition technique for direct simulation Monte Carlo method”, *The Japan Society of Mechanical Engineering (B)* **57**(540), 2696–2700.
Ota, M., Taniguchi, H. and Aritomi, M. (1995) “Parallel processings for direct simulation Monte Carlo method”, *The Japan Society of Mechanical Engineering (B)* **61**(582), 496–502.
Piekos, E.S. and Breuer, K.S. (1996) “Numerical modeling of micromechanical devices using the direct simulation Monte Carlo method”, *Transaction ASME Journal of Fluids Engineering* **118**, 464–469.
Plimpton, S., Bartel, T. (1993) Parallel Particle Simulation of Low-Density Fluid Flows, U.S. Department of Energy Report No. DE94–007858.
Robinson, C.D. (1998) “Particle simulations on parallel computers with dynamic load balancing”, Ph.D. Thesis, Imperial College of Science, Technology and Medicine (UK).
Russell, D. (1968) “Density disturbance ahead of a sphere in rarefied supersonic flow”, *The Physics of Fluids* **11**, 1679–1685.
Schaff, S. and Chambre, P. (1958) Chapter H, Fundamentals of Gas Dynamics (Princeton University Press, Princeton, NJ).
Vanderstraeten, D. and Keunings, R. (1995) “Optimized partitioning of unstructured finite element meshes”, *International Journal of Numerical Methods in Engineering* **38**(3), 433–450.
Vanderstraeten, D., Farhat, C., Chen, P.S., Keunings, R. and Zone, O. (1996) “Aretrofit based methodology for the fast generation and optimization of large-scale mesh partitions: beyond the minimum interface size criterion”, *Computer Methods in Applied Mechanics and Engineering* **133**, 25–45.
Wagner, W. (1992) “A convergence proof for bird's direct simulation Monte Carlo method for the Boltzmann equation”, *Journal of Statistical Physics* **66**(3/4).
Walshaw, C., Cross, M., Everett, M.G., Johnson, S. and McManus, K. (1995) “Partitioning and mapping of unstructured meshes to parallel machine topologies”, In: Ferreira, A. and Rolim, J., eds, Proc Irregular Parallel Algorithms for Irregularly Structured Problems (Springer **Vol. 980 of LNCS**, pp 121–126.
Wu, J.S. and Tseng, K.C. (2001) “Analysis of micro-scale gas flows with pressure boundaries using direct simulation Monte Carlo method”, *Computers and Fluids* **30**, 711–725.