Adaptive Load Balance Techniques in Parallel Rarefied Gas Simulations

S. Antonov and F.–J. Pfreundt
Institute for Industrial Mathematics
Kaiserslautern
Germany

J. Struckmeier
Department of Mathematics
University of Kaiserslautern
Germany

Abstract

The paper presents some adaptive load balance techniques for the simulation of rarefied gas flows on parallel computers. It is shown that a static load balance is insufficient to obtain a scalable parallel efficiency. Hence, two adaptive techniques are investigated which are based on simple algorithms. Numerical results show that using heuristic techniques one can achieve a sufficiently high efficiency over a wide range of different hardware platforms.

1 Introduction

Particle or Monte–Carlo methods are efficient numerical tools to predict rarefied gas flows around re–entry bodies. Besides the well–known DSMC approach developed by Bird [5], the Finite–Pointset Method as described in [8] has been widely used to investigate rarefied gas flows [3], [4], [10]. Moreover, is is known, that – in general – Monte–Carlo methods can be implemented more easy on parallel architecture then other methods, like, e.g., FEM–computations. The same holds for particle schemes for rarefied gas simulations and severel authors already investigated parallel versions of the classical serial codes [2], [11], [14]. To achieve an appropriate parallel efficiency, it is necessary to introduce adaptive load balance concepts. The main reason for this is the strong variation of the macroscopic density within the spatial domain. In the current paper we present some load balance concepts which result in very efficient parallel particle schemes for rarefied gas flows.

The paper is organized as follows: in Section 2 we briefly describe the mathematical equation which describes rarefied gas flows and explain the main ideas how to simulate this equation using particle methods. Moreover, we indicate how to implement those schemes on a parallel architecture and why an adaptive load balance becomes necessary to obtain scalable parallel schemes. Section 3 deals with some heuristic methods to obtain a sufficiently accurate load balance and especially we focuse on two different approaches: the so–called Min–Max–Update as well as an Streamline Method. Heuristic methods are necessary due to the high–dimensionality of the underlying optimization problem. The following section is devoted to a wide range of applications and includes a comparison of various hardware platforms. A conclusion of the current paper is given in Section 5.

2 Rarefied Gas Simulation by Particle Methods

2.1 Particle Methods

Rarefied gas flows are described by the well–known Boltzmann equation introduced by Ludwig Boltzmann in 1872, a nonlinear transport equation which defines the time evolution of the density function of the gas ensemble in the phase space. In this section we consider the Boltzmann equation on the spatial domain $\Omega \subset I\!R^3$ defined as the nonlinear transport equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f = \frac{1}{\varepsilon} Q(f)$$

for the density function $f = f(t, x, v)$ of the gas ensemble, together with the initial condition

$$f(0, x, v) = \overset{o}{f}(x, v)$$

and appropriate boundary conditions on $\partial\Omega$, like inflow and outflow conditions or some scattering conditions of the form

$$|(v,n)| f(t,x,v) = \int\limits_{(v',n)<0} R(v,v',t,x) |(v',n)| f(t,x,v') \mathrm{d}v' \quad (v,n) > 0$$

where $R(v \to v'; t, x)$ is an appropriate gas–surface scattering kernel. The collision operator – here written down for the simplest case of a monoatomic gas – is expressed in the form

$$Q(f) = \int\limits_{I\!R^3} \int\limits_{S^2_+} k(|v - v_*|, n) \{ f' f'_* - f f_* \} \mathrm{d}n \mathrm{d}v_*$$

where we used the notations

$$\begin{aligned} f' &= f(t,x,v') \qquad & v' &= v - (v - v_*, n)n \\ f'_* &= f(t,x,v'_*) \qquad & v' &= v_* + (v - v_*, n)n \end{aligned}$$

Besides the simple monoatomic case, one may consider a rarefied gas mixture consisting of M different species, where the different species might carry internal energies and the whole system might undergo chemical reactions, like dissociation–recombination reactions [9].

Remark 2.1
It turns out, that concerning the parallelization the particle schemes for monoatomic gas and gas mixtures with chemical reactions are very similar. Hence, we will restrict ourselve in the first part of the paper to the monoatomic Boltzmann equation. Nevertheless, the results presented in Section 4 include computations with real gas effects.

The most efficient numerical tools to simulate the Boltzmann equation are Particle Methods (also called Monte–Carlo Methods), where the solution $f = f(t, x, v)$ is approximated at every time $t \geq 0$ by a finite set of particles in the phase space $\Omega \times I\!R^3$. A finite set of particles is defined as the set $\{(\alpha_1, x_1, v_1), ..., (\alpha_n, x_n, v_n)\}$, where $(x_i, v_i) \in \Omega \times I\!R^3$, $i = 1, ..., n$ and α_i, $i = 1, ..., n$ are the weights of the particles. Both particle characteristics, i.e. points in the phase space as well as the weights might depend on time

$$(x_i, v_i) = (x_i(t), v_i(t)) \quad \alpha_i = \alpha_i(t) \quad i = 1, ..., n$$

and the principle of a particle scheme is to derive a time evolution on the level of the discrete approximation by particles. It is out of the scope of the paper to describe in detail the derivation of such particle schemes. Hence, we restrict ourselve to the main steps of the particle method:
to derive the time evolution of the particles, one uses a splitting method based on a discrete time step Δt, i.e. over the time interval $[n\Delta t, (n+1)\Delta t)$ one solves the two equation

$$\frac{\partial f}{\partial t} + v\nabla_x f = 0 \tag{2.1}$$

and

$$\frac{\partial f}{\partial t} = Q(f) \tag{2.2}$$

1. The Free Flow of Particles
Equation (2.1) can be solved by the method of characteristics, which leads to the simple free flow condition on the level of the particle approximation, i.e.

$$x_i((n+1)\Delta t) = x_i(n\Delta t) + \Delta t v_i(n\Delta t) \quad v_i((n+1)\Delta t) = v_i(n\Delta t) \tag{2.3}$$

If the particle trajecory defined by (2.3) intersects with the boundary $\delta\Omega$, one has to consider the corresponding boundary condition.

2. Collision Process between Particles
To obtain the particle scheme for equation (2.2) one replaces the exact collision operator $Q(f)$ by a spatially smoothed operator $Q^{\Delta x}(f)$. Based on a partition $\bigcup\limits_{k\in\mathcal{K}} Z_k^{\Delta x} = \Omega$ of the spatial domain, such that $\operatorname{diam} Z_k^{\Delta x} \leq \Delta x$, and a smoothing kernel $\beta^{\Delta x}$ given by

$$\beta^{\Delta x}(x, x_*) = \sum_{k\in\mathcal{K}} \frac{\mathcal{X}_{Z_k^{\Delta x}}(x)\mathcal{X}_{Z_k^{\Delta x}}(x_*)}{Vol(Z_k^{\Delta x})}$$

the collision operator $Q(f)$ is replaced by

$$Q^{\Delta x}(f) = \int\limits_{\Omega}\int\limits_{\boldsymbol{R}^3}\int\limits_{S_+^2} \beta^{\Delta x}(x, x_*) k(|v - v_*|, n)\{f' f'_* - f f_*\} \mathrm{d}n \mathrm{d}v_* \mathrm{d}x_*$$

with the notations

$$f' = f(t, x, v') \quad f'_* = f(t, x_*, v'_*) \quad etc.$$

Due to the special form of the smoothing kernel $\beta^{\Delta x}$, the problem is reduced to a system of spatial–homogeneous Boltzmann equations on the cell system $Z_k^{\Delta x}$ and one continues applying some discretization techniques for the spatial–homogeneous Boltzmann equation. Especially the spatial coordinate of each particle remains constant during this collision step.

2.2 Particle Methods on Parallel Computers

A parallel version of particle methods for rarefied gas simulations is obtained by dividing the spatial domain Ω into k non–overlapping subdomains, where k is the number of processors. Then, each processor computes the time evolution of the those particles belonging to the own subdomain. Communication routines between processors handle the particles which leave subdomains during the free flow condition – the first part of the splitting approach – whereas no communication is necessary during the second step – the collisions between particles – because the spatial coordinates remain constant.
The following two criteria should be used to obtain an appropriate partition on the processors:

- each processor should use nearly the same CPU–time to compute the time evolution of the particles belonging to the processor,
- the communication between the processors should be as small as possible.

Because rarefied flow simulations are time–dependent simulations, it is in general not possible to fulfill both criteria using a static partition of the computational domain. The following results (Table 1) are obtained on a nCUBE 2S parallel system using a static partition for the rarefied gas flow around the EXPRESS re–entry capsule (see Section 4.1).

Table 1. Parallel Efficiency with Static Load Balance

Proc.	CPU [m:s]	Speedup	Effic.	lbc
64	10:13	23.2	36.3	2.61
32	11:15	21.1	65.9	1.48
16	21:25	11.1	69.2	1.45
8	35:20	6.7	83.9	1.20
4	67:19	3.5	88.1	1.16

Remark 2.2
Concerning the definition of the speedup factor, the efficiency as well as the load balance coefficient (lbc), we refer the reader to Section 4, where we present detailed numerical investigations.

Due to the shockfront around the re–entry body, the density field is strongly inhomogeneous and this results in an insufficient load balance. A–priori information on the regions with high density are not accurate enough to obtain a–priori partitions which overcome this effect. A further reason is, that in transient computations the number of simulation particles does not remain constant, but increases due to the formation of the shockfront. Moreover, solving larger problems a static partition may result in a memory overflow on certain processors. Hence, adaptive load balance techniques are absolutely necessary to obtain efficient parallel codes.

3 Adaptive Load Balance Techniques

As mentioned in the last section, it is necessary to use adaptive load balance concepts to obtain efficient particle schemes which are scalable with respect to the number of processors as well as the flow characteristics. In the case of rarefied flow simulations, an adaptive load balance technique might be implemented by changing the partition of subdomains on the processors within the transient computation. To perform this partition it is appropriate to introduce the so–called Load Balance Units (lbu's) which are defined as the smallest subdomains that may be exchanged between processors. Here, the lbu's consists of a certain number of cells $Z_k^{\Delta x}$ used to perform the collision process between particles. This concept was already used in [11], where the load balance units consists of sticks along the flow direction.

Remark 3.1
The repartition of the domain should be implemented in the particle scheme after the first step of the splitting approach. Due to the repartitioning a communication step between the processors becomes necessary and this can be done in combination with the exchange of particles due to the free flow condition.

With the concept of lbu's, the repartition problem may be formalized as follows: Let k be the number of processors $\{p_1, ..., p_k\}$, p the number of lbu's $\{g_1, ..., g_p\}$, then a partition of lbu's on the set of processors is given by a mapping $P : \{1, ..., p\} \to \{1, ..., k\}$, such that the lbu g_i belongs to processor p_j, if $P(i) = j$. Moreover, let t_i the CPU–time used to compute to solution on lbu g_i. Then, the CPU–time t^j on processor p_j is given by the formula

$$t^j = \sum_{i=1}^{p} t_i \delta_{jP(i)}$$

where

$$\delta_{kl} = \begin{cases} 0 & \text{if } k \neq l \\ 1 & \text{else} \end{cases}$$

and the averaged CPU–time t_{av} is defined as

$$t_{av} = \frac{1}{k} \sum_{i=1}^{k} t^i$$

Now, let $\mathcal{P}$ the set of all mappings $P : \{1, ..., p\} \to \{1, ..., k\}$ and for given $\varepsilon > 0$ we define the set $\mathcal{P}_\varepsilon$ of all $P \in \mathcal{P}$ such that

$$\max_{i=1,...k} |t_{av} - t^i| < \varepsilon$$

First of all, partitions out of $\mathcal{P}_\varepsilon$ ensure an appropriate load balance with respect to the parameter ε and if we perform a repartition of the spatial domain, we should choose an element out of $\mathcal{P}_\varepsilon$. Moreover, we like to choose a partition which results in a low communication between the processors. This is formalized by considering the flux F_{ij} of particles from lbu g_i to g_j. Then, the total flux of particles between the processors based on the partition $P \in \mathcal{P}$ is given by

$$F^{\mathcal{P}} = \sum_{i=1}^{k} \sum_{j,l=1}^{p} F_{ij}\delta_{iP(j)}(1 - \delta_{iP(l)})$$

If we denote by P' the actual partition of Ω, then it is moreover necessary to consider the communication effort to pass from P' to P. This costs $K_{P' \to P}$ can be expressed by

$$K_{P' \to P} = \sum_{i=1}^{k} \sum_{j=1}^{p} \delta_{iP'(j)}(1 - \delta_{iP(j)})$$

If we denote by $\overset{o}{K}$ the bound on the costs $K_{P' \to P}$, we finally obtain the following minimization problem:
for given $\varepsilon, \overset{o}{K}$ and the partition P' find the minimizer on the set $\mathcal{P}_\varepsilon$ of the functional F_P, under the restriction that $K_{P' \to P} \leq \overset{o}{K}$.

Remark 3.2
Instead of using the CPU–time of each processor, one might use some estimates on the CPU–times, like the number of particles each processor has to simulate. This estimate turns out to be accurate as long as the gas–surface interaction requires only a small part of the total CPU–time, e.g., if the collisions between particles are the dominating part.

It is obvious, that this is a difficult optimization problem. Hence, one has to use some heuristic algorithms to obtain a computationally cheap solution which is close to the optimal solution. Two heuristic concepts are described in the following:

- the Min–Max–Update
  A similar technique was already used in [11], based on the number of particles on each processor,

- the Streamline Approach
  this technique uses the concept of streamlines in order to reduce the communication between the processors.

3.1 Min–Max–Update

The Min–Max–Update is a simple idea to obtain a repartition using the actual partition P, but does not consider the effort for the communication between processors. Some lbu's are simple moved to other processors, such that the actual partition P' becomes an element of $\mathcal{P}_\varepsilon$ for some small $\varepsilon > 0$. This algorithm is described as follows:

Min–Max–Update

1) *Let P be the actual partition and define for each processor a switch $s_i \in \{0,1\}$, such that $s_i = 0$ by*

$$s_i = \begin{cases} 0 & \text{if } \sum\limits_{j=1}^{p} \delta_{iP(j)} \leq 1 \\ 1 & \text{else} \end{cases}$$

i.e. the switch s_i is "off" if the number of lbu's belonging to processor p_i is less than 2.

2) *perform the following loop at most k times*

   a) *determine the processors l_{max} and l_{min} with*

$$t^{av} - t^{l_{max}} = \max\{t_{av} - t^1, ..., t_{av} - t^k\}$$

   and

$$t^{av} - t^{l_{min}} = \min\{t_{av} - t^1, ..., t_{av} - t^k\}$$

   If $s_{l_{max}} = 0$ or $s_{l_{min}} = 0$ then go to to 3)

   b) *take the last lbu with index p in the list of lbu's of the processor l_{max}.*
   If $t_p \geq t^{l_{max}} - t^{l_{min}}$ then
   goto 3)
   else
   move this lbu to the end of the list of lbu's of the processor
   and define $t^{l_{max}} := t^{l_{max}} - t_p$, $t^{l_{min}} := t^{l_{min}} + t_p$
   if $t^{l_{min}} < t_{av}$ and $t^{l_{max}} > t_{av}$ then goto a)

   c) *if $t^{l_{max}} \leq t_{av}$ then put $s_{l_{max}} = 0$*
   if $t^{l_{min}} \geq t_{av}$ then put $s_{l_{min}} = 0$
   goto a)

3) *stop the load balancing step.*

Remark 3.3
This algorithm only uses the CPU–times of the set $\{g_1, ..., g_p\}$ of lbu's to obtain an appropriate partition and the effort to perform this load balance step is neglectable in comparison with the global CPU–time. As already metioned, one might use instead of the CPU–times the number of particles belonging to the single lbu's.

Numerical experiments show that this simple strategy yields a sufficient parallel efficiency as long as the communication between processors is sufficiently fast. We refer the reader to the numerical results presented in Section 4. Similar to the load balance technqiues investigated in [11], the final partition turns out to be randomly distributed over the computational domain Ω.

Remark 3.4
If the number of lbu's is of the same order as the number of processors, it might be useful to enlarge the Min–Max–Update described above using a binary exchange of lbu's along two processors.

3.2 The Streamline Approach

The Min–Max–Update described in the previous subsection yields a sufficiently accurate algorithm to obtain an appropriate load balance and works well if $p/k \gg 1$ and the size of a lbu is sufficiently large as in the 3D–case. Nevertheless, in this approach there exists no strategy to involve an estimate on the communication between processors which is necessary to handle the free flow condition of the splitting approach.

The idea to include some control on the communcation is to arrange the subdomains, i.e. the lbu's belonging to one processor, along the streamlines of the flow. If the lbu's of one processor are located on a streamline, one may expect to obtain a low amount of communication because most of the particles will follow the streamlines.

Given a time–dependent velocity field $u : I\!R_+ \times \Omega \to I\!R^3$ on the spatial domain Ω, a streamline is defined as a curve on Ω such that the tangent is instantaneously everywhere parallel to the velocity field u. Especially, if the velocity field is time–independent, the streamlines remain constant in time.

The Streamline Approach discussed in the following turns out to be an efficient load balance strategy for two–dimensional as well as axisymmetric computations where the computational domain has an inflow boundary. Hence, we restrict ourselve to this situation: first we define for each lbu g_i the corresponding mean velocity $v_i \in I\!R^2$ which is obtained by taking the arithmetic average over all particle velocities belonging to the lbu g_i. Moreover, we denote by g'_j, $j = 1, ..., p^*$, those lbu's which have an intersection with the inflow boundary of the domain Ω. Then, the Streamline Approach to perform a repartitioning P' is defined as follows:

Streamline Approach

1) **Initialization**
put the inflow lbu's g'_j, $j = 1, ..., p^*$ *on a stack* S *and set* $s_i = 0$ *for all* $i = 1, ..., p$ *to indicate wether the lbu* g_i *already belongs to some processors.*
put $m = 1$ *and set* $t^i = 0$ *for all* $i = 1, ..., k$ *and assume that the averaged CPU–time* t_{av} *with respect to the actual partition* P *is given*

2) **Following the Streamlines**
determine the lbu's belonging to processor p_m *by performing the following steps*

a) *take lbu* g'_{j_*} *from top of the stack* S*, if the stack is empty go to 3)*

b) *If* $t_{j_*} + t_m \le t_{av}$ *then*
put $P'(j_*) = m \quad s_{j_*} = 1 \quad t^m = t^m + t_{j_*}$ *and goto b)*
else *put* $m = m + 1$, *if* $m > k$ *then go to 3) else go to b)*

c) consider the velocity vector $v_{j*} = (v^x_{j*}, v^y_{j*})$
if $|v^x_{j*}| > |v^y_{j*}|$ *then*
choose the next lbu g_k *in x–direction with respect to* g_{j*}
if their exist no lbu go to a)
if $s_k = 0$ *then*
put $j_* = k$ *and go to b)*
else
go to a)
else
choose the next lbu g_k *in y–direction with respect to* g_{j*}
If their exist no lbu go to a)
put the next lbu in x–direction with respect to g_{j*} *on top of the stack S*
if $s_k = 0$ *put* $j_* = k$ *and go to b)* *else go to a)*

3) **Distributing the rest**
determine the lbu's for which P(i) is undefined and put these lbu's on a stack S'

a) determine the processor $p_{l_{min}}$ *with minimal CPU–time* $t^{l_{min}}$

b) take lbu g''_{j*} *from top of the stack* S' *and put*

$$P(j_*) = l_{min} \quad t^{l_{min}} = t^{l_{min}} + t_{j*}$$

if the stack is empty go to 4)

c) If $t^{l_{min}} < t_{av}$ *go to b)*
else go to a)

4) stop the load balancing step.

Remark 3.5
The decision 2.c) to proceed aligning the lbu's in x or y–direction approximates the technique to pass along a virtual streamline running through the lbu's. The reason to put the lbu located in x–direction on top of the stack S, if one proceeds in y–direction, is to allow the next processors to follow a streamline from this point on. The Streamline Approach is graphically illustrated in Fig. 1.

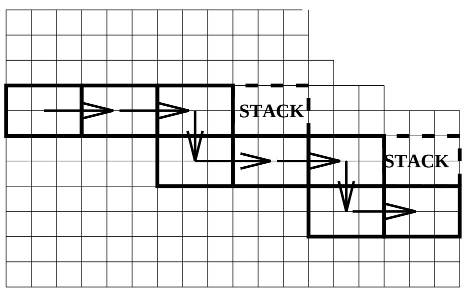

Figure 1: Streamline Approach in two dimensions

4 Numerical Results

In this section we present some rarefied gas simulations using various hardware platforms and compare the parallel efficiency of the adaptive load balance techniques described in the previous section. Especially, we consider the following re–entry configurations:

1. the EXPRESS capsule (Section 4.1)
2. a Deltawing configuration (Section 4.2)
3. the ARD capsule (Section 4.3)

The following hardware platforms are used to evaluate the parallel efficiency:

Table 2. Hardware Platforms

System	Proc.	Type	Mb
INTEL Paragon	256	INTEL 860	32
CRAY T3D	128	DEC Alpha 150 MHz	64
nCUBE 2S	64	nCUBE 25MHz	8
IBM SP2	32	Power 2 66 MHz	128
Pentium PC–cluster	8	Pentium 90 MHz	64
HP–9000/K	2	PA–7200 100 MHz	256

The numerical results are obtained using the Finite–Pointset–Method (FPM) as described in [8] and [12]. The implementation of this particle method on parallel computers (using the MPI communication library [7]) as well as the investigation of adaptive load balance techniques was done in the framework of the German HPCN–initiative ParBoSS [1].

In order to quantify the parallel efficiency of rarefied gas simulations the following notations are used in this section: The speedup factor using n processors is defined as the ratio of the total CPU–time T^n on n processors to the CPU–time T^1 on a single node, i.e

$$Sp = \frac{T^1}{T^n}$$

Moreover, we define the parallel efficiency by

$$E = \frac{Sp}{n}$$

where n is the number of processors.

Remark 4.1
If the problem size is too large to run the simulation on a single node, we use an estimate on the single–node CPU–time. This estimate is obtained by summing up the CPU–times of the single processors without the communication time as well as without the additional effort to perform the load balancing steps. Tests against the exact single–node CPU's show, that the estimated values for the speedup are even lower than the exact ones. For parallel computers with Cache memory, where one expects even an super–linear speedup factor, the single–node CPU–time is estimated using the minimal number of processors for which the simulation can be performed. Especially, the theoretic speedup values on this machines is in general underestimated.

The Load Balance Coefficient (lbc) is defined as the ratio between the maximal CPU–time over the single processors divided by the averaged CPU–time, i.e.

$$lbc = \frac{t_{max}}{t_{av}}$$

where both CPU–times of the single processors are given as the total CPU–time minus the communication time and the time to perform the adaptive load balancing.

4.1 The EXPRESS Capsule

The EXPRESS capsule configuration (see Fig. 2) includes two pitot pressure probes: one at the (geometric) stagnation point as well as one along the side line.

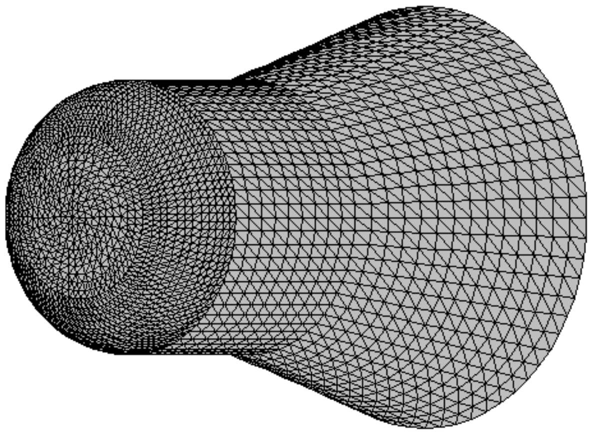

Figure 2: EXPRESS capsule configuration

The aim of the project is to measure atmospheric data during the re–entry of the test configuration. Due to the geometry, one is able to consider an axisymmetric configuration if the angle of attack is equal to zero. The main purpose of rarefied gas simulations is the prediction of the flow conditions inside the pressure probe given the outer flowfield around the capsule. We refer the reader to the results given in [6].

Results on the parallel efficiency of a particle scheme for this configuration using a nCUBE 2S system with up to 64 nodes are shown in Table 3a, b and c. The simulation uses about 60.000 particles to obtain the steady state solution after 800 time steps for a Knudsen number of 0.0961.

The spatial domain around the EXPRESS capsule is divided into $60 \times 32 = 1920$ spatial cells and each cell represents one load balance unit. At time $t = 0$ each cell is filled with 28 particles which carry besides translational energy also rotational energy. The Mach number of the flow is equal to 27 with a freestream temperature of 192.6 K.

Table 3a. Parallel Efficiency with Static Load Balance

Proc.	CPU [m:s]	Speedup	Effic.	lbc
64	10:13	23.2	36.3	2.61
32	11:15	21.1	65.9	1.48
16	21:25	11.1	69.2	1.45
8	35:20	6.7	83.9	1.20
4	67:19	3.5	88.1	1.16

Table 3b. Parallel Efficiency using Max–Min–Update

Proc.	CPU [m:s]	Speedup	Effic.	lbc
64	7:19	32.4	50.7	1.67
32	9:38	24.6	76.9	1.22
16	17:26	13.6	85.0	1.14
8	32:18	7.3	91.7	1.06
4	60:08	3.9	98.6	1.007

Table 3a. Parallel Efficiency using Streamline Approach

Proc.	CPU [m:s]	Speedup	Effic.	lbc
64	6:15	38.0	59.3	1.230
32	8:46	27.1	84.6	1.035
16	16:15	14.6	91.2	1.019
8	31:17	7.6	94.7	1.012
4	60:59	3.9	97.2	1.010

Remark 4.2
The adaptive load balance method, i.e. the repartioning of the lbu's, is performed in every 20th time step of the transient computation. One may even use an adaptive method here in order to reduce the number of load balance steps if the flow becomes stationary.

As shown in the tables, for low numbers of processors, the static load balance technique works reasonable well although the Min–Max–Update as well as the Stramline Approach already give better results. If the number of processors is enlarged, e.g., from 32 to 64 nodes, the parallel efficiency of the static partition drops dramatically down: the CPU–time using 32 and 64 nodes remains nearly constant. On the other hand, both adaptive load balance concepts still produce an increasing speedup factor, where the Streamline Approach gives better results and the difference to the Min–Max–Update increases with the number of processors. Similar results using an adaptive load balance technique are obtained using different parallel system: Table 4 shows some results using the Streamline Approach as adaptive load balance technique.

Table 4. Parallel Efficiency using Streamline Approach

System	Proc.	CPU [m:s]	Speedup	Effic.	lbc
CRAY T3D	64	1:50	36.5	57.0	1.42
	32	2:29	27.0	84.3	1.05
	16	4:39	14.0	87.3	1.02
	8	9:02	7.4	92.6	1.04
	4	17:39	3.8	94.8	1.03
	2	34:01	1.97	98.4	1.008
	1	66:57	-	-	-
nCUBE 2S	64	6:15	38.0	59.3	1.230
	32	8:46	27.1	84.6	1.035
	16	16:15	14.6	91.2	1.019
	8	31:17	7.6	94.7	1.012
	4	60:59	3.9	97.2	1.010
IBM SP2	32	1:23	19.8	61.7	1.13
	16	2:06	13.0	81.1	1.03
	8	3:59	6.9	85.7	1.008
	4	7:10	3.8	95.4	1.004
	2	13:52	1.97	98.6	1.004
	1	27:20	-	-	-
PC	8	8:10	6.65	83.1	1.03
	4	14:05	3.86	96.4	1.02
	2	27:07	2.003	100.15	1.005
	1	54:19	-	-	-
HP	2	17:33	1.88	94.2	1.009
	1	33:04	-	-	-

The speedup factors given in Table 4 show, that although applying an adaptive load balance technique, the parallel efficiency drops down when the number of processors is increased. This indicates that the problem size becomes too small using a large number of nodes, i.e. the problem size has even to fit the computational power to obtain a reasonable parallel efficiency.
As an example, one may pass to a full three–dimensional computation using the same mesh size. Introducing a further spatial dimension increases the problem size by one order of magnitude. The results on the parallel efficiency for the three–dimensional calculation using the Min–Max–Update are given in Table 5.

Table 5. Parallel Efficiency using Min–Max–Update

System	Proc.	CPU [m:s]	Speedup	Effic.	lbc
CRAY T3D	128	17:49	120.6	94.2	1.05
	64	34:10	62.9	98.3	1.02
	32	68:55	31.2	97.4	1.009
	16	136:39	15.7	98.3	1.005
nCUBE 2S	64	139:37	60.8	94.9	1.02
	32	273:48	31.0	96.8	1.01
IBM SP2	32	34:40	32.3	100.9	1.01
	16	68:53	16.3	101.6	1.007
	8	138:50	8.1	100.8	1.005
	4	282:38	3.96	99.0	1.004

The parallel efficiency remains reasonable high even if the number of processors in increased.

Remark 4.3
Numerical experiments show the Streamline Approach gives less efficient results for three–dimensional computations. This is due to the fact, that the approximation of real streamlines by the technique given in Section 3.2 is not accurate enough. Hence, one looses the advantage of a lower effort for the communication and, at the same time, the computational effort applying the Streamline Approach is much higher than in the Min–Max–Update.

4.2 Deltawing Configuration

Hypersonic rarefied flow simulations around a deltawing configuration are classical test problems for rarefied gas simulation techniques. Several authors already investigated this configuration using various simulation tools [3], [4]. The actual configuration is shown in Fig 3.

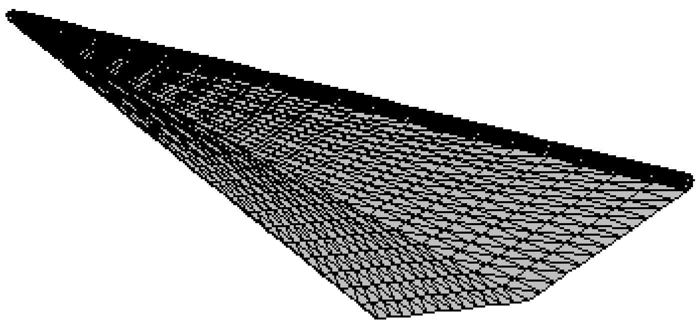

Figure 3: Deltawing configuration

Although one expects a high communication between the single processors, the Min–Max–Update gives a quite reasonable parallel efficiency. Table 6 gives some results using various hardware platforms as well as various numbers of processors. The simulation is performed at a Knudsen number of 0.1 and a Mach number of 8.9 on a $20 \times 64 \times 20$ grid and about 1.000.000 particles using 300 time steps. One lbu consists of 80 cells, i.e. the spatial domain consists of 320 lbu's. Moreover, the load balance steps are performed in every 10th time step of the transient computation.

Table 6. Parallel Efficiency using Min–Max–Update

System	Proc.	CPU [m:s]	Speedup	Effic.	lbc
Paragon	256	2:06	182.0	71.1	1.15
	128	3:18	115.5	90.3	1.08
	64	6:09	62.2	97.11	1.03
	32	12:39	30.2	94.4	1.02
CRAY T3D	128	1:28	115.6	90.3	1.06
	64	1:50	61.4	95.9	1.02
	32	2:29	31.2	97.5	1.01
	16	4:39	15.8	98.9	1.007
	8	9:02	7.9	99.0	1.005
	4	42:54	3.9	98.8	1.003
nCUBE 2S	64	10:54	57.2	89.3	1.02
	32	20:46	30.0	93.7	1.02
	16	40:24	15.4	96.4	1.007
IBM SP2	32	3:04	33.9	106.1	1.02
	16	5:47	18.0	112.4	1.01
	8	11:50	8.8	109.8	1.005
	4	24:41	4.2	105.5	1.005
	2	50:33	2.06	103.0	1.002
	1	103:54	-	-	-
Pentium	8	24:41	7.38	92.3	1.02
	4	47:45	3.82	95.4	1.007
HP	2	56:36	1.99	99.5	1.002
	1	112:41	-	-	-

Two special points have to mentioned

- the IBM SP2 system gives a speedup which is for all nodes greater then the theoretical value. This is due to the increasing efficiency of the cache system,
- the INTEL Paragon drops dramatically done going from 128 to 256 nodes. This indicates that the current problem is too small using a high number of nodes.

4.3 ARD Capsule

The ARD capsule configuration (see Fig. 4) was investigated in the framework of the ESA–project "Development of Prediction Tools for Re–Entry Capsule Aerothermodynamics". The aim of the project is to study the influence of chemical reactions on the aerothermodynamic characteristics of re–entry configurations. The current simulations use a 5 species air model consisting of N_2, O_2, N, O and NO together with 19 dissociation and exchange reactions where the rate coefficients are based on the Dunn–Kang model (recombination reactions are neglected).

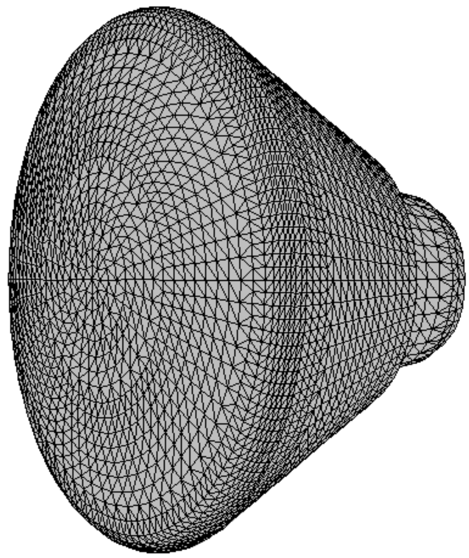

Figure 4: ARD capsule configuration

The modelling of the chemical cross sections as well as the implementation of chemical reactions in a particle scheme are based on the work given in [9] and [13]. Numerical results are obtained for two altitudes at 94 and 103 km. For a detailed description we refer the reader to [13].

As already mentioned, the aim of the simulations presented in the following is to study the influence of chemical reactions in rarefied gas flows. One expects that chemical reactions become more important as the altitude of the re–entry configuration decreases because of the higher collision frequency at lower altitudes. Hence, two computations are performed for both altitudes at 94 and 103 km. The first simulation uses a gas mixture consisting of nitrogen and oxygen molecules with mole fractions of 0.756 and 0.244. The second simulation includes the formation of nitrogen and oxygen atoms as well as nitrogenoxyd due to chemical reactions (dissociation and exchange reactions).

All simulations include an angle of attack of 20 degree such that the full three-dimensional ARD–configuration has to be used, although the ARD–capsule is defined as an axisymmetric configuration. The physical parameters used for both altitudes are summarized in Table 7. Moreover, the boundary condition at the surface of the ARD–capsule is given by complete accomodation with diffusive reflection at a wall temperature of 600 K for the translational as well as rotational energies. The vibrational energies remain unchanged during the interaction. The corresponding numerical parameters as well as the parallel efficiency using different architectures are given in Table 8. In all simulations the Min–Max–Update is applied to obtain a reasonable performance on parallel computers and the load balance steps are performed in every 20th time step.

Table 7. Physical Parameters for the ARD–capsule

H [km]	ρ_∞ [kg/m^3]	λ_∞ [m]	Kn	V_∞ [m/s]	T_∞ [K]
94	$1.459 \cdot 10^{-6}$	0.05	0.0178	7768	193
103	$2.945 \cdot 10^{-7}$	0.26	0.0929	7758	224

In Tables 7 and 8 we used the following abbrevations:

H	altitude of the re–entry configuration
ρ_∞	freestram density
λ_∞	freestream mean free path
Kn	Knudsen number
V_∞	freestream velocity
T_∞	freestram temperature
CPU	total CPU–time in hours and minutes
$\mathbf{N}_{part}$	total number of simulation particles

Table 8. Numerical Parameters for the ARD–capsule

H	Chem.	Hardware	Proc.	CPU	Speedup	Effic.	$\mathbf{N}_{part}$
94	no	Cray T3D	128	4:39	118.1	92.3	$9.2 \cdot 10^6$
94	yes	Cray T3D	128	8:58	119.6	93.4	$11.3 \cdot 10^6$
103	no	nCUBE 2S	64	6:43	55.9	87.3	$1.6 \cdot 10^6$
103	yes	Cray T3D	32	3:16	30.8	93.2	$1.7 \cdot 10^6$

Moreover, the simulation includes an adaptive grid refinement in order to cover the large density deviations over the spatial domain. In all simulations a reasonable parallel efficiency is achieved using the Min–Max–Update. Finally, the incluence of chemical reactions on the aerothermodynamic characteristics of the ARD–configuration are given in Table 9.

Table 9. Aerothermodynamic Characteristics of the ARD–capsule

H [km]	Chem.	CA	CN	Cm	CD	CL	Ch
94	no	1.432	0.169	-0.079	1.403	-0.330	0.210
94	yes	1.439	0.170	-0.080	1.411	-0.332	0.160
103	no	1.535	0.253	-0.088	1.529	-0.286	0.400
103	yes	1.535	0.254	-0.089	1.530	-0.286	0.382

where

CA	axial force coefficient
CN	normal force coefficient
Cm	pitching moment based on the stagnation point
CD	drag coefficient
CL	lift coefficient
Ch	heat transfer coefficient

5 Conclusion

In order to achieve a reasonable parallel efficiency of rarefied gas simulations using particles methods it is necessary to introduce appropriate adaptive load balance techniques. Due to the high–dimensionality of the underlying optimization problem, it is moreover necessary to develop heuristic methods. Two such approaches are presented in the current investigation. Introducing the concept of load balance units (lbu's) one might use a simple Min–Max–Update between the single processors based on the CPU–times as well as a more sophisticated methods where on tries to align the load balance units along streamlines (Streamline Method).

Numerical experiments show that the latter one yields a reasonable load balance for axisymmetric (as well as two–dimensional) simulations. Using an even simpler technique, like the Min–Max–Update, may even increase the parallel efficiency in three–dimensional computations. Moreover, the results obtained are nearly independent of the underlying hardware platforms.

The numerical results presented in the paper include three different re–entry configurations as well as a wide range of different parallel systems and the results clearly indicate the efficiency of rarefied gas simulations on parallel computers when using an adaptive load balance strategy. Nevertheless, the results even show the well–known fact that the efficiency is correlated with the size of the investigated problem. The results presented in the paper use adaptive load balance step after a fixed number of time steps. Techniques how to include an adaptive method to perform the repartitioning in order to reduce the number of load balance steps if the flow becomes stationary are currently under investigation.

Acknowledgement

The main part of this research was funded by the German Minister for Research and Technologies under the contracts 01–IR–408–A–0 and 03–NE7KAI. Moreover, the research was supported by the European Space Agency under the contract n° 11171/94/F/WE "Development of Prediction Tools for Re–Entry Capsule Aerothermodynamics" (ARD capsule).

References

[1] S. Antonov, M. Hartig, F.J. Pfreundt, W. Sack, K. Steiner, J. Struckmeier, A. Schüller and G. Koppenwallner, "BMBF–Verbundprojekt ParBoSS: Mathematik – Parallelisierung – Anwendungen", in: G. Wolf and R. Krahl (Eds.), Statustagung des BMBF HPSC 95, DLR (1995).

[2] S. Dietrich and I.D. Boyd, "Load Balancing for the DSMC Method in a Parallel Implementation", preprint, 1994.

[3] F. Gropengießer, H. Neunzert, J. Struckmeier and B. Wiesen, "Rarefied Gas Flow Around a 3D–Deltawing", in: Desideri, J.A.; Glowinski, R. and Periaux, J. (Eds.), Hypersonic Flows for Reentry Problems, Volume II, Springer , 1067–1078, 1991.

[4] F. Gropengießer, H. Neunzert, J. Struckmeier and B. Wiesen, "Rarefied Gas Flow Around a 3D Delta Wing", in: Abgrall, R.; Desideri, J.A.; Glowinski, R.; Mallet, M. and Periaux, J. (Eds.), Hypersonic Flows for Reentry Problems, Vol. III, Springer Verlag, 976–986, 1992.

[5] G.A. Bird, "Molecular Gas Dynamics and the Direct Simulation of Gas Flows", Clarendon Press, Oxford, 1994.

[6] A. Hartmaier and H.M. Urbassek, "Transition Flow in an Impact Pressure Probe: Relevance of gas–wall interaction", preprint, Dept. of Physics, University Kaiserslautern, 1996.

[7] MPI, "Message–Passing Interface Standard", Message Passing Interface Forum, May 1994.

[8] H. Neunzert and J. Struckmeier, *Particle Methods for the Boltzmann Equation*, Acta Numerica 1995, Cambridge, 417–457.

[9] W. Sack, "Modelling and Numerical Simulations of Chemically Reacting Flows in Rarefied Gases" (in German), PhD–thesis, University Kaiserslautern, 1995.

[10] J. Struckmeier, "Development of Prediction for Re–Entry Capsule Aerothermodynamics", Internal Reports, ESA contract n° 11171/94/F/WE, 1995–1996.

[11] J. Struckmeier and F.J. Pfreundt, "On the efficiency of simulation methods for the Boltzmann equation on parallel computers", *Parallel Computing* **19**, 103–119 (1993).

[12] J. Struckmeier and K. Steiner, "A Comparison of Simulation Methods for Rarefied Gas Flows", *Physics of Fluids A* **7**, 2876–2885 (1995).

[13] K. Steiner and J. Struckmeier, "Modelling and Simulation of Chemically Reacting Rarefied Gas Flows", in preparation.

[14] R.G. Wilmoth, "Application of a Parallel DSMC Method to Hypersonic Rarefied Flows", AIAA 91–0772, 1991.

Jens Struckmeier
Department of Mathematics
University of Kaiserslautern
P.O. Box 3049
67653 Kaiserslautern
Germany
email: struckm@mathematik.uni-kl.de