The Journal of Supercomputing, 13, 211–232 (1999)

Modelling of a Utility Boiler Using Parallel Computing

P. J. COELHO, P. A. NOVO AND M. G. CARVALHO

Instituto Superior Técnico, Mechanical Engineering Department, Technical University of Lisbon, Av. Rovisco Pais, 1096 Lisboa Codex, Portugal

Abstract. A mathematical model for the simulation of the turbulent reactive flow and heat transfer in a power station boiler has been parallelized. The mathematical model is based on the numerical solution of the governing equations for mass, momentum, energy and transport equations for the scalar quantities. The k-ε model and the conserved scalar/prescribed probability density function formalism are employed. Radiative heat transfer is calculated using the discrete ordinates method. The code has been fully parallelized using the spatial domain decomposition approach and MPI. Calculations were performed using an IBM-SP2. It is shown that the computational requirements are reduced and the parallel efficiency increases if the mean temperature and density are calculated *a priori*, and stored. The role of the different parts of the code on the parallel performance is discussed. A speedup of 5.9 is achieved using 8 processors.

Keywords: boilers; computational fluid dynamics; turbulent reactive flows; radiative heat transfer; discrete ordinates; parallel processing

Nomenclature

A_x, A_y, A_z	Area of a control volume normal to the x, y and z directions, respectively.
C_p	Specific heat at constant pressure
C_μ	Constant of the turbulence model
C_1, C_2, C_D	Constants of the turbulence model
f	Mixture fraction
h	Specific enthalpy
I	Radiation intensity
k	Turbulent kinetic energy
M	Molar mass
n	Number of iterations
p	Pressure; number of processors
$p(f)$	Probability density function
R_o	Universal gas constant
s	Coordinate along the direction of a radiation beam
S	Speedup
t	Time
T	Temperature
u_j	Velocity component in j direction
V	Volume

x_j	Coordinate along j direction
Y_j	Mass fraction of species j
γ	Weighting factor
ε	Dissipation rate of turbulent kinetic energy; parallel efficiency
κ	Absorption coefficient of the medium
η	Direction cosine
μ	Dynamic viscosity; direction cosine
ξ	Direction cosine
ρ	Density
σ_k, σ_ε	Constants of the turbulence model
σ_t	Constant of the turbulence model
ϕ	Arbitrary scalar variable

Subscripts

b	Blackbody
i, j	Directions
in	Incoming
iter	Iteration
p	Number of processors
P	Grid node
x, y, z	Directions

Superscripts

o	Standard state
$\bar{(\ )}$	Reynolds average
$\tilde{(\ )}$	Density-weighted average
$(\ )''$	Density-weighted fluctuation

1. Introduction

The demand for electricity in the world has increased enormously during this century. Most of the electricity is produced in thermal or nuclear power plants, the remaining fraction being satisfied by water power resources or renewable energy sources. But the main fraction is generated at thermal power plants which produce electricity by conversion of the thermal energy released by the combustion of fossil fuels, such as coal, fuel-oil or natural gas, in the boiler. A schematic of a simple vapor power cycle in a thermal power plant is shown in Fig. 1. The heat released from the fuel combustion in the combustion chamber of the boiler vaporizes the water which circulates in the tubular panels that form the walls of the boiler. The steam rises to the drum where the moisture is removed. Then, it is superheated in the convection section of the boiler and sent to the high pressure turbine where it is expanded enabling the conversion of part of its thermal energy into mechanical energy. The alternator is connected to the turbine and converts the mechanical energy into electrical energy. The expanded steam returns to the convection section of the boiler where it is reheated, and then sent to the low pressure

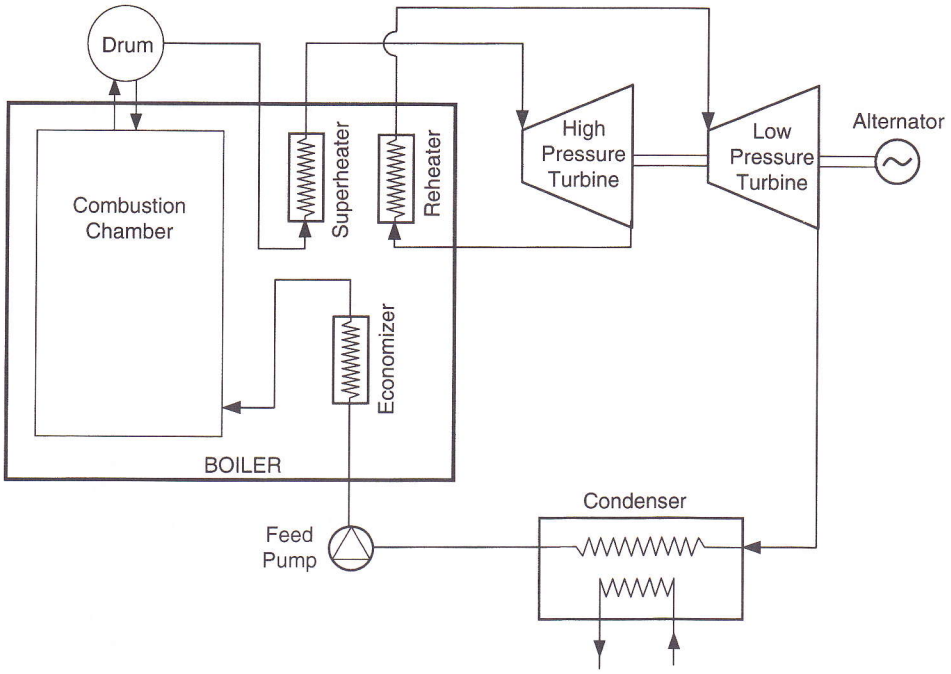

Figure 1. A vapor power cycle.

turbine. After expansion in this turbine the steam is liquefied in the condenser, compressed and preheated in the economizer before it returns to the combustion chamber to restart the cycle.

The boiler is one of the key components of thermal power plants. It is an expensive equipment and therefore it is essential that its design and operation ensure an efficient combustion, with the pollutant emissions below the maximum legislated levels, and allow the burning of a wide range of fuels without affecting the time life of the equipment. Mathematical models of the combustion chamber of the boiler can provide a useful help to face these challenges.

The modelling of the combustion chamber of a utility boiler is a difficult and time consuming task due to the complexity and interaction between different physical phenomena. The practical usefulness of the results of a model depends on their accuracy and on the cost of the computational simulation. It is generally recognized that currently used models have shortcomings and need to be improved. The numerical accuracy can be improved by using high resolution schemes for the discretization of the convective terms, employing body fitted coordinates and seeking for grid independent solutions. In some problems this last issue is particularly difficult to deal with. In fact, a typical utility boiler contains several burners with a complex geometry, including swirling vanes, and relevant characteristic dimensions (e.g., the atomizer in fuel-oil fired boilers) may be three or four orders of magnitude smaller than the overall dimensions of the boiler. As a consequence,

the total number of control volumes needed to solve accurately such a problem would require prohibitively large computing times. Therefore, coarser grids and simplifications in the geometrical details and physical models are currently employed. The physical accuracy requires improvements on the turbulence modelling and near wall region treatment, combustion modelling, radiation and radiative properties modelling. The interaction between turbulence, combustion and radiation enhances the difficulty of the problem.

Virtually any advance on the numerical or physical accuracy of a model implies an increase of the computational requirements. This constitutes a strong motivation for the development of more powerful solution techniques. Several different numerical techniques have been developed aiming at a reduction of the computing time of a simulation. These include faster solvers for the algebraic sets of equations, faster solution algorithms, adaptive techniques, domain decomposition techniques (overlapping grids, multi-block grids with discontinuous grid lines along the interfaces of neighbouring blocks, local grid refinement), and multigrid. Some of these techniques may be coupled together. On another hand, developments in the hardware have lead to the spreading of vector and parallel computers. The use of this hardware generally requires modifications of the traditional sequential codes to allow full exploitation of the vector or parallel features of such machines.

The present paper addresses the development and application of a parallel code to the mathematical modelling of a power station boiler of the Portuguese Electricity Utility. A similar parallelization development applied to a coal-fired boiler is in progress at the University of Stuttgart [1–3]. The parallelization of fluid flow codes has been widely investigated, and the combustion community has also been active on this subject. However, the parallelization of radiative heat transfer codes has been very limited. The present work deals with a turbulent reactive flow with radiative heat transfer. It constitutes one of the first works in which all the physical models (turbulence, combustion, radiative heat transfer) have been parallelized.

The mathematical model is described in the next section, and is followed by the description of the parallel implementation. The results are presented and discussed in section four. The paper ends with a summary of the main conclusions.

2. The Mathematical Model

The mathematical model is based on the numerical solution of the governing equation for mass, momentum and energy and transport equations for scalar quantities. In steady state, and for high Reynolds numbers, the density weighted average form of the governing equations for turbulent flows may be written in Cartesian coordinates as

- Mass

$$\frac{\partial}{\partial x_j}\left(\rho \tilde{u}_j\right) = 0 \tag{1}$$

- Momentum

$$\frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_i\tilde{u}_j\right) = -\frac{\partial}{\partial x_j}\left(\bar{\rho}\widetilde{u_j''u_j''}\right) - \frac{\partial \bar{p}}{\partial x_i} + \bar{\rho}g_i \tag{2}$$

- Scalars

$$\frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j\tilde{\phi}\right) = -\frac{\partial}{\partial x_j}\left(\bar{\rho}\widetilde{u_j''\phi''}\right) + \bar{S}_\phi \tag{3}$$

The energy equation may be written in the form of Eq. (3) with the stagnation enthalpy h in the place of ϕ.

The Reynolds stresses $\widetilde{u_i''u_j''}$ and the turbulent scalar fluxes $\widetilde{u_j''\phi''}$ are determined by means of the k-ε eddy viscosity/diffusivity model:

$$-\bar{\rho}\widetilde{u_i''u_j''} = \mu_t\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}\right) - \frac{2}{3}\left(\rho k + \mu_t\frac{\partial \tilde{u}_k}{\partial x_k}\right)\delta_{ij} \tag{4}$$

$$-\bar{\rho}\widetilde{u_j''\phi''} = \frac{\mu_t}{\sigma_t}\frac{\partial \tilde{\phi}}{\partial x_j} \tag{5}$$

and

$$\mu_t = C_\mu \bar{\rho}k^2/\varepsilon \tag{6}$$

This model involves the solution of transport equations for the turbulent kinetic energy, k, and its dissipation rate, ε:

$$\frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j k\right) = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right) - \bar{\rho}\widetilde{u_i''u_j''}\frac{\partial \tilde{u}_i}{\partial x_j} - \frac{\mu_t}{\rho^2}\frac{\partial \bar{\rho}}{\partial x_j}\frac{\partial \bar{p}}{\partial x_j} - \bar{\rho}\varepsilon \tag{7}$$

$$\frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j\varepsilon\right) = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x_j}\right) + C_1\frac{\varepsilon}{k}\left(-\bar{\rho}\widetilde{u_i''u_j''}\frac{\partial \tilde{u}_i}{\partial x_j} - \frac{\mu_t}{\rho^2}\frac{\partial \bar{\rho}}{\partial x_j}\frac{\partial \bar{p}}{\partial x_j}\right) - C_2\bar{\rho}\frac{\varepsilon^2}{k} \tag{8}$$

The constants of the model are $C_\mu = 0.09$, $C_1 = 1.44$, $C_2 = 1.92$, $\sigma_k = 1.0$, $\sigma_\varepsilon = 1.3$ and $\sigma_t = 0.7$ [4].

Combustion modelling is based on the assumptions that the reaction rates are very fast compared to the mixing rates, and that the mass diffusion and thermal diffusion coefficients are equal. This allows the determination of the instantaneous thermochemical state of the gaseous mixture in terms of conserved scalars. In the

case of an adiabatic flow, a single conserved scalar, generally taken as the mixture fraction, is sufficient to specify the relationships between instantaneous values of temperature, density, mass fractions and mixture fraction. In the present work these state relationships were derived from a chemical equilibrium code [5]. Complete equilibrium was assumed up to a critical value of the mixture fraction, f_c. Adiabatic mixing of the mixture at f_c and pure fuel was assumed for mixture fractions greater than f_c [6]. The value of f_c was selected to match the maximum measured CO mass fraction.

In boilers the radiation plays a dominant role and there is no unique relation between enthalpy and mixture fraction. Thus, the energy conservation equation is also solved for, and the ideal gas law is used to close the governing set of equations. The enthalpy and the temperature are related by

$$h = \sum_i Y_i h_i = \sum_i Y_i \left(h_i^0 + \int_{T^0}^{T} C_{p,i}(T^*)\, dT^* \right) \tag{9}$$

and the enthalpy of each species may be expressed as a polynomial in T.

The turbulent fluctuations are taken into account by assuming that the mixture fraction has a clipped Gaussian probability density function. The mean values of the chemical species concentrations, temperature and density may be found from integration of the product of the instantaneous values by that probability density function over the mixture fraction range

$$\tilde{\phi} = \int_0^1 \phi(f) p(f)\, df \tag{10}$$

$$\overline{\phi} = \bar{\rho} \int_0^1 \frac{\phi(f)}{\rho(f)} p(f)\, df \tag{11}$$

$$\bar{\rho} = \left[\int_0^1 \frac{p(f)}{\rho(f)}\, df \right]^{-1} \tag{12}$$

Transport equations are solved to compute the mixture fraction and its variance as follows

$$\frac{\partial}{\partial x_j}\left(\bar{\rho} \tilde{u}_j \tilde{f} \right) = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_t} \frac{\partial \tilde{f}}{\partial x_j} \right) \tag{13}$$

$$\frac{\partial}{\partial x_j}\left(\bar{\rho} \tilde{u}_j \widetilde{f''^2} \right) = \frac{\partial}{\partial x_j}\left(\frac{\mu_t}{\sigma_t} \frac{\partial \widetilde{f''^2}}{\partial x_j} \right) + \frac{2\mu_t}{\sigma_t}\left(\frac{\partial \tilde{f}}{\partial x_j} \right)^2 - C_D \frac{\bar{\rho}\varepsilon}{k} \widetilde{f''^2} \tag{14}$$

where C_D is a constant of the model which is usually set to 2.0.

The calculations performed in this work using Eq. (10), (11) and (12) to obtain the mean density and temperature at every iteration will be referred to as case 1.

However, it is possible to perform the integrations *a priori* for a range of mean mixture fraction, variance of mixture fraction and enthalpy values. The computed mean temperature and density are stored for all the possible combinations and used to obtain the temperature and density during the CFD (computational fluid dynamics) calculations without the need to use Eq. (10), (11) and (12) again. The calculations carried out in this way will be identified as case 2.

The discrete ordinates method [7, 8] was used to calculate the radiative heat transfer. In the case of a grey non-scattering medium the radiative heat transfer equation (RTE) may be written as

$$\frac{dI}{ds} = -\kappa I + \kappa I_b \tag{15}$$

In the discrete ordinates method this equation is solved for a set of $n(n+2)$ directions, yielding the so-called S_n approximation. These directions span the total solid angle range of 4π around a point in space. Along a direction $\vec{s}_i$ the RTE reads as

$$\xi_i \frac{\partial I_i}{\partial x} + \eta_i \frac{\partial I_i}{\partial y} + \mu_i \frac{\partial I_i}{\partial z} = -\kappa I_i + \kappa I_b \tag{16}$$

where ξ_i, η_i and μ_i are the direction cosines of that direction. If this equation is discretized using the finite-volume approach, a relationship between the volume average radiation intensity, I_{P_i}, and the radiation intensities entering (subscript in) and leaving a control volume is obtained:

$$I_{P_i} = \frac{kV\gamma I_b + |\xi_i| A_x I_{x_{in,i}} + |\eta_i| A_y I_{y_{in,i}} + |\mu_i| A_z I_{z_{in,i}}}{kV\gamma + |\xi_i| A_x + |\eta_i| A_y + |\mu_i| A_z} \tag{17}$$

The parameter γ relates the incoming and the outgoing radiation intensities to the volume average intensity and it was set to one, according to the step scheme. Additional details of the method, including the derivation of these equations, the boundary conditions treatment and the solution algorithm may be found in references [7–9].

The S_4 approximation and the level symmetric quadrature satisfying odd moments [10] were employed in the present work. The control volumes in the ash-pit and nose regions of the boiler which lie outside of the physical domain were dealt with as proposed in [11]. The radiant superheaters which are suspended from the top of the combustion chamber were simulated as baffles without thickness, as described in [12].

The absorption coefficient of the medium is required to integrate the radiative transfer equation. The weighted sum of grey gases model extended to account for soot was used to calculate the absorption coefficient of the medium. The soot particles are too small to scatter, and the scattering from soot agglomerates was neglected. The soot mass fraction was determined from the solution of a transport

equation. The soot formation model of Khan and Greeves [13] and the soot oxidation model of Magnussen and Hjertager [14] were employed.

The mean value of the fourth power of temperature which appears in the radiative transfer equation (RTE) and the mean value of the source term of the soot mass fraction transport equation were determined via integration of the instantaneous values over the mixture fraction range (Eq. 11). Although it is possible to store the integrated values, as explained above for the density and the temperature, the present calculations did not use this option. In fact, the radiation and the soot mass fraction equation were only solved every 10 iterations of the main iterative loop. Therefore, the increase of the computational time associated with the integrations is not too high.

The governing equations were discretized over a cartesian, non-staggered grid using a finite volume/finite difference method. The SIMPLE solution algorithm is used. The algebraic sets of discretized equations are solved using the Gauss-Seidel line-by-line iterative procedure, except the pressure correction equation which is solved using a preconditioned conjugate gradient method. Both the discretization procedure and the solution method are well known and a full description may be found, e.g., in [15].

3. Parallel Implementation

The parallel implementation is based on a domain decomposition strategy which is implemented in a standard message passing library (MPI). This standard has now been implemented widely and the code is therefore easily portable across hardware ranging from workstation clusters, through shared memory modestly parallel servers to massively parallel systems. Within the domain decomposition approach the computational domain is split up into a number of subdomains, each one being assigned to a processor. Each processor runs its own code dealing with a sub-section of the global grid and communicates and synchronizes its actions with those of other processors by exchanging messages.

The global domain is partitioned into subdomains and in general the three-dimensional grid of points is mapped onto a logical three-dimensional array of processors. If n_x, n_y and n_z correspond to the number of processors along the x-, y- and z-axis, respectively, the total number of processors will be $p = n_x{}^* n_y{}^* n_z$. The global and the partitioned domains are schematically shown in Fig. 2 for the case $p = 8$ and $n_x = n_y = n_z = 2$. When the domain has been partitioned the neighbour processor topology has to be determined for communication purposes. Once this logical mapping has been established, neighbouring processors with common boundaries can be found. In general each processor will need to know its 6 neighbours. Processors on global boundaries will have fewer neighbours.

In order to ease the implementation of the computations a buffer of halo points is added to the rectangular domains to accommodate the finite volume computational stencils on both real and subdomain boundaries. These halo points are shown in Fig. 3 for one of the subdomains of a two-dimensional problem with $n_x = n_y = 4$. Along a given coordinate direction the information in the first and in

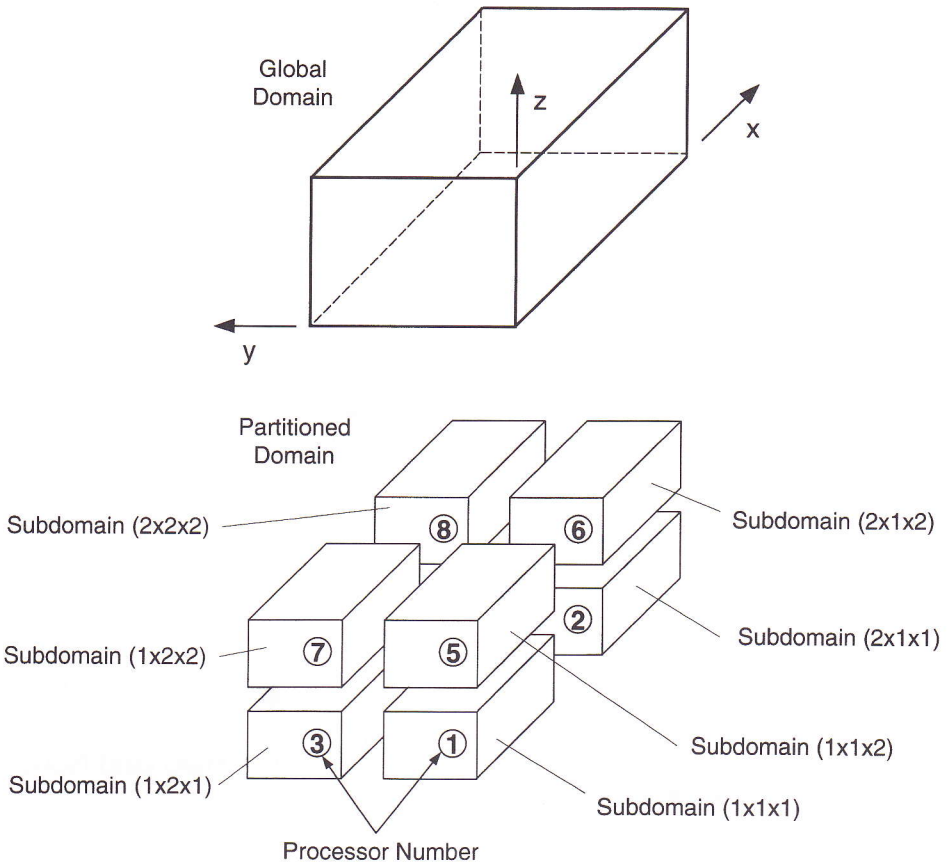

Figure 2. Global and partitioned domains.

the last planes of active grid points are sent to the halo region of the neighbouring subdomains. In each iteration of the solution algorithm the transfer of the halo data must be effected prior to each call to solve a governing equation. This ensures the correct coupling of the local solutions into the global solution. The calculation of the various source terms and finite difference coefficients is performed explicitly for each grid point and can proceed trivially in parallel. The tri-diagonal matrix algorithm and the pre-conditioned conjugate gradient solvers are parallelised by restricting the scope of the back substitution steps to points in the subdomain, using field values at active points on adjacent processors in neighbouring subdomains as boundary conditions. The halo data transfer between neighbouring processors is achieved by a pair-wise exchange of data. This transfer proceeds in parallel.

As well as this nearest neighbour communications, the processors need to communicate global data such as values of residuals which need to be accumulated,

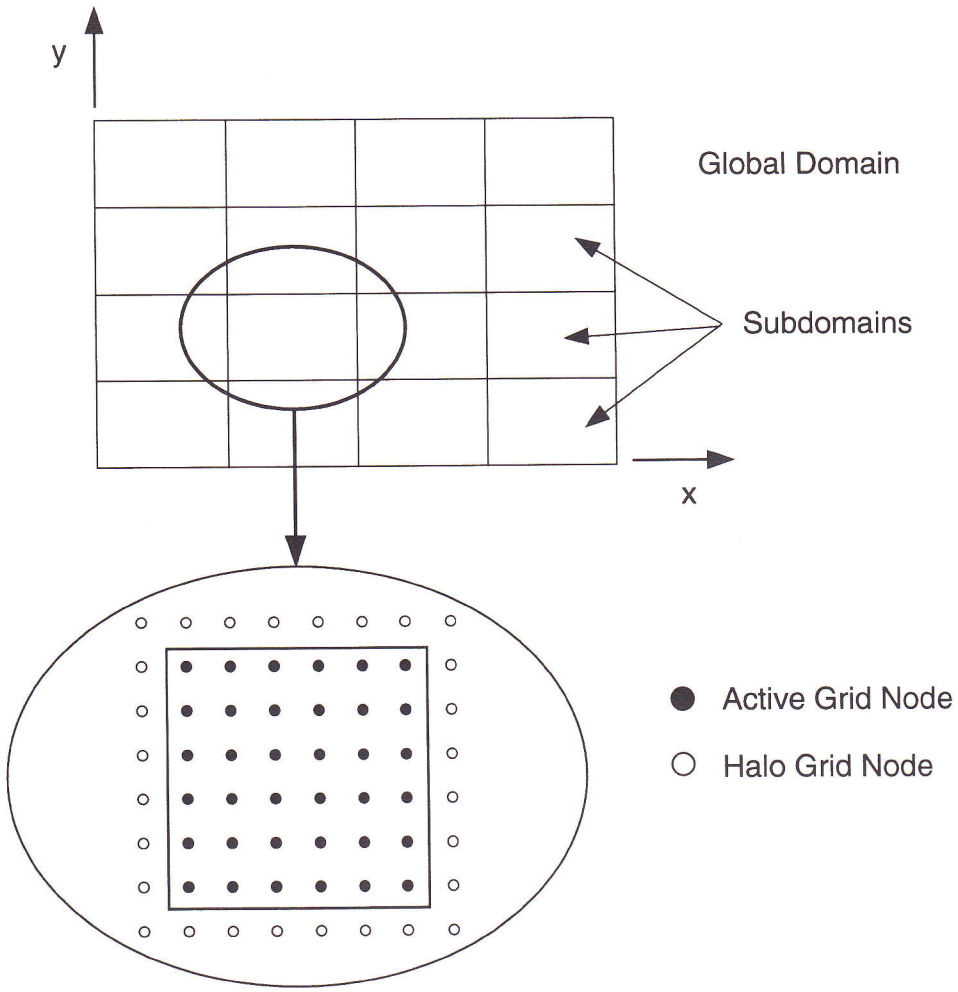

Figure 3. Active and halo grid nodes of a subdomain of a two-dimensional problem mapped onto 16 processors ($n_x = n_y = 4$).

or maximum or minimum values determined or values broadcast. These global operations are now available in standard message passing interfaces.

The code is divided into three parts: the *reader*, the *pre-processor* and the *core-code*. The *reader* package is concerned with the input of data, generation of boundary condition data structures, and initialization of the dependent variables and properties fields. The *pre-processor* splits up the global data amongst the necessary number of files. Each file is assigned to a unique processor. Neither the *reader* nor the *pre-processor* are parallelized. The *core-code* starts by barrier synchronizing all processors, and issuing MPI calls to determine the number of processors being used and the identity of its particular process. The input data

required in each processor is read before entering the main iteration loop. In this loop the segregated solver runs through all the governing equations, solves the radiative heat transfer equation with a certain frequency (every 10 iterations in the calculations presented below) and updates the properties, e.g., the density.

While the parallelization of the fluid flow equations has been extensively addressed in the literature, the parallelization of the radiation model is a comparatively unexplored subject. A study on the parallelization of the discrete ordinates model has recently been carried out and is fully reported in [16]. In that study which was restricted to radiation problems two different parallelization methods were implemented and compared, the angular decomposition and the spatial domain decomposition. It was found that the most straightforward and efficient way to parallelize the discrete ordinates method is angular decomposition. In this method the total number of directions along which the RTE is solved is divided into a number of subsets equal to the number of processors. Each processor performs the calculations for the whole domain but treats only a certain subset of directions. For example, if the S_4 approximation is used, as in the present work, then 24 directions are considered and the direction cosines of those directions are given in [9]. If 8 processors are used, each processor deals with 3 directions, and solves the RTE along these 3 directions along the whole domain. This means that the grid data (coordinates of the grid nodes and cell faces, dimensions of the control volumes), the temperature of the gas and wall boundaries, the absorption coefficient of the medium and the emissivity of the walls for the whole domain must be available to every processor.

The fluid flow equations are generally parallelized using the spatial domain decomposition method. In this way, in distributed memory machines only the dependent variables at the grid nodes of the subdomain assigned to a processor reside on the memory of that processor. Therefore, the angular decomposition parallelization of the discrete ordinates method is not suitable for implementation in a computational fluid dynamics code parallelized using a domain decomposition approach, since this parallelization strategy requires that data from the whole domain are available to every processor. The only efficient way of using the angular decomposition for the RTE coupled to a CFD problem parallelized using domain decomposition would be in a problem of small size enabling the data from the whole domain to fit into the memory of every processor. Hence, the spatial domain decomposition was used in the present work, since it enables the governing transport equations and the RTE to be parallelized in a similar fashion, and it can be used regardless of the problem size.

4. Results and Discussion

The model was applied to the combustion chamber of a power station boiler of the Portuguese Electricity Utility schematically shown in Fig. 4. It is a natural circulation drum fuel-oil fired boiler with a pressurized combustion chamber, parallel passages by the convection zone and preheating. The boiler is fired from three levels of four burners each, placed on the front wall. Vaporization of the fuel is

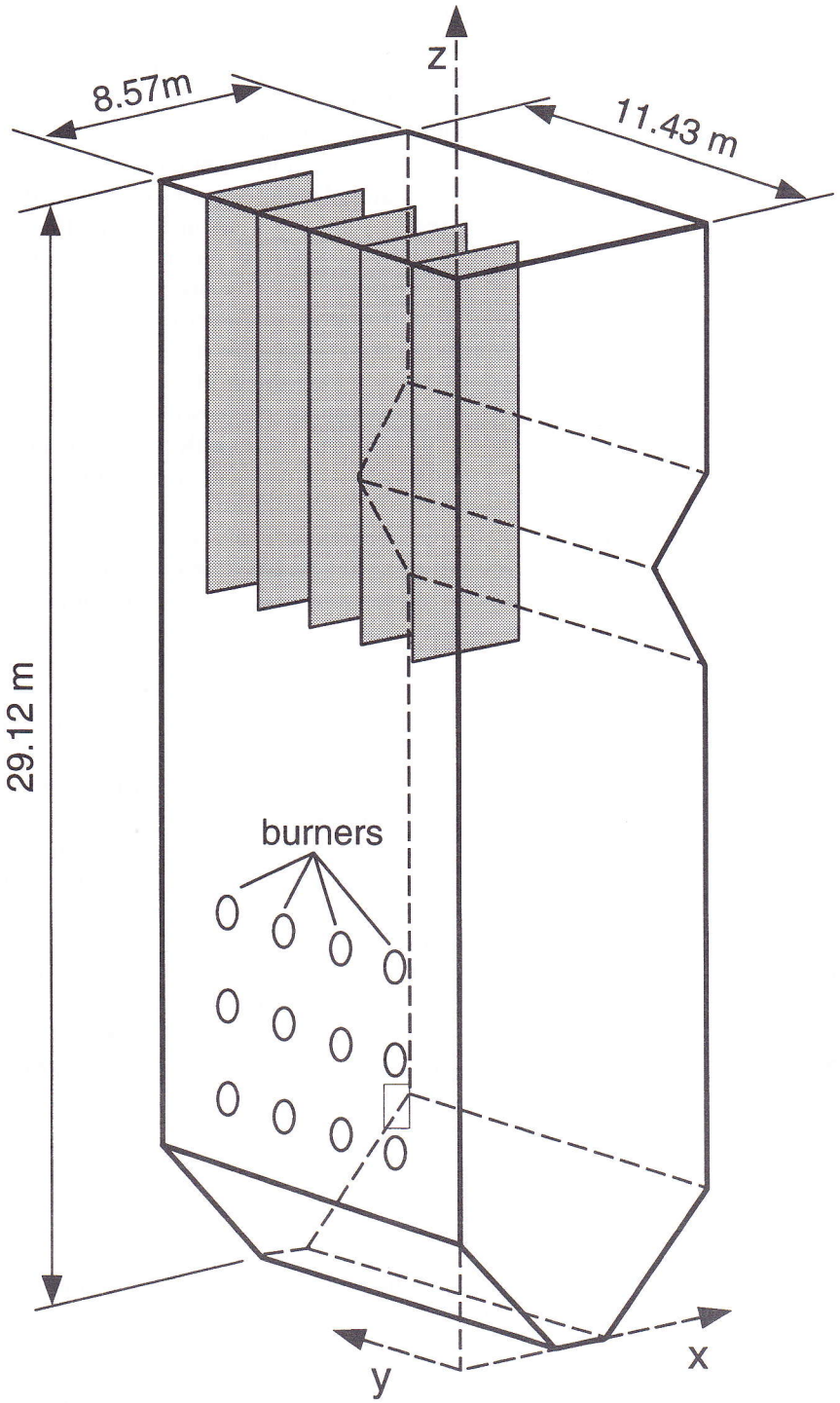

Figure 4. Schematic of the boiler.

assumed to occur instantaneously. At maximum capacity (771 ton/h at 167 bar and 545 °C) the output power is 250 MWe. This boiler has been extensively investigated, both experimentally and numerically. Detailed predictions using a sequential code, measurements, and comparisons between predictions and measurements may be found elsewhere [17–20]. The present work is exclusively concentrated on the parallel performance of the code.

The calculations were performed on an IBM-SP2 using a mesh with $16 \times 39 \times 68$ grid nodes and up to 8 processors. The grid is partitioned in such a way that the number of grid nodes allocated to each subdomain is approximately equal. In the case of 8 processors, the subdomains have $8 \times 20 \times 34$ or $8 \times 19 \times 34$ active control volumes, plus one additional layer of grid nodes on each boundary (halo data region). The convergence criterion demands that the sum of the absolute values of the normalised residuals of mass and momentum decrease below 10^{-4}. The radiative calculations, are performed every 10 iterations of the fluid flow calculations, and a maximum of 10 iterations are allowed for the radiative calculations every time they are called. In fact, due to the iterative nature of the solution algorithm, a tight convergence of the RTE is not needed.

The parallel performance of the code will be examined by means of the efficiency, ε, the efficiency per iteration, $\varepsilon_{\text{iter}}$, and the speedup, S. These quantities are defined as follows:

$$\varepsilon = \frac{t_1}{p * t_p} \tag{18a}$$

$$\varepsilon_{iter} = \frac{t_1/n_1}{p * t_p/n_p} \tag{18b}$$

$$S = t_1/t_p \tag{18c}$$

where p denotes the number of processors, t_i is the time required to achieve convergence using i processors, and n_i is the corresponding number of fluid flow iterations.

The decrease of the efficiency that occurs when the number of processors increases is due to three factors: the degradation of the convergence rate, the time required to exchange data among the processors, and the uneven distribution of the computational load among the processors, i.e., load imbalance. The efficiency per iteration is a consequence of the last two factors, and of the degradation of the convergence rate in the solution of the RTE, but does not reflect the increase of the number of fluid flow iterations.

The number of iterations and the total time required to achieve convergence are given in table 1 for both cases 1 and 2, and for 1, 2, 4, 6 and 8 processors. In case 1 the mean density and temperature are determined at every iteration of the solution algorithm using Eq. (10)–(12), while in case 2 these quantities are computed *a priori*, stored in tabular form, and interpolated from the stored data during the CFD calculations. In both cases the number of iterations increases with the

Table 1. Parallel performance of the code.

	Case 1					Case 2				
Number of processors	1	2	4	6	8	1	2	4	6	8
Number of iterations	1033	1033	1098	1097	1092	1019	1016	1102	1098	1092
Total time (s)	35556	20769	12208	8938	6623	24454	13555	7724	5575	4159
Efficiency	—	85.6%	72.8%	66.3%	67.1%	—	90.2%	79.1%	73.1%	73.5%
Efficiency per iteration	—	85.6%	77.4%	70.4%	70.9%	—	89.9%	85.6%	78.8%	78.8%
Speedup	—	1.7	2.9	4.0	5.4	—	1.8	3.2	4.4	5.9

number of processors, but not monotonically. The complex interaction between different phenomena and the non-linearity of the governing equations may be responsible for the non-monotonic behaviour, which has also been found in other studies. The total time decreases from case 1 to case 2, especially when p increases. The efficiency and the speedup are smaller for case 1 than for case 2. In both cases a slightly higher efficiency was achieved using 8 processors instead of 6. This is related to the partitions employed ($2 \times 3 \times 1$ and $2 \times 2 \times 2$ for 6 and 8 processors, respectively) and to the number of grid nodes in the halo region. In the case of 6 processors the z direction which has the maximum number of grid nodes (68) was not partitioned. This enabled an even distribution of grid nodes among the processors, but originated large halo regions with the consequent increase of the communication time.

Although the speedup is what really matters from the user point of view, since it indicates how much faster the code runs using p processors instead of one, it is useful to investigate more deeply the reasons behind the departure of the actual speedup from the ideal linear speedup. If these reasons are well understood it may be possible to improve the speedup by modifying the parts of the code responsible for the decrease of speedup. To achieve this goal we will now look at the distribution of the total time by the different parts of the code. This distribution is given in Fig. 5 for both cases 1 and 2, and for 1 and 8 processors. The tasks considered are the solution of the momentum equations (u, v, w); the solution of the pressure correction equation and the correction of velocities and pressure (p'); the solution of the turbulent kinetic energy equation (k) and the dissipation rate equation (ε); the solution of other transport equations, including energy, mixture fraction, mixture fraction variance, and soot mass fraction equations; the calculation of the mean density, temperature, fourth power of temperature, source term of the soot mass fraction equation, absorption coefficient of the medium and viscosity (mean properties); and the solution of the RTE (radiation).

In case 1 the calculation of the mean properties is the most time consuming task, accounting for 42.9% of the total time using 1 processor and for 48.9% using 8 processors. These times are dramatically reduced in case 2. This clearly shows that the calculation of the mean density and temperature via Eq. (10–12) is a computationally demanding task. As expected, the time required by all the other tasks

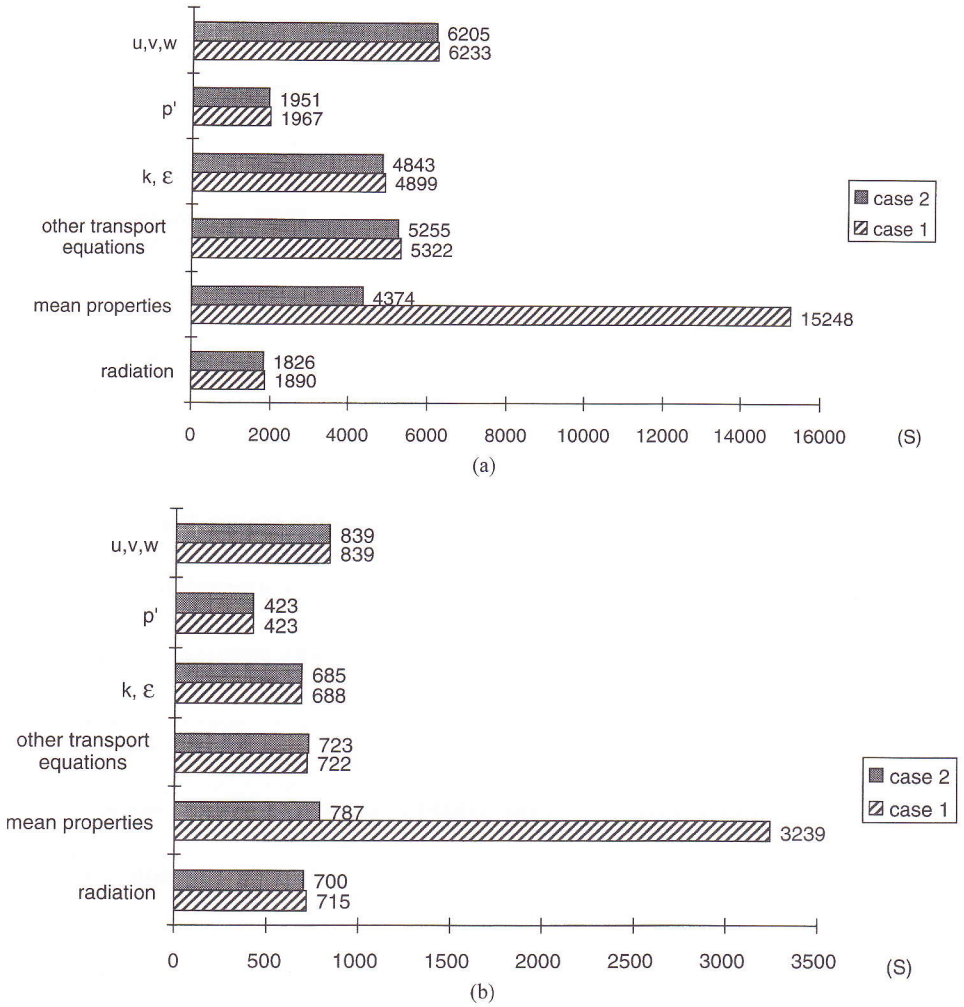

Figure 5. Calculation times per task. a) One processor, b) Eight Processors

changes only marginally from case 1 to case 2, regardless of the number of processors. It is also obvious from Fig. 5 that the radiation task is the least demanding one using one processor, but it is a time consuming task using eight processors.

A better understanding of the role of the different tasks on the parallel performance is obtained by calculating the efficiencies per task. These are calculated via Eq. (18a) using the time per task rather than the total time. The results for case 2 are summarised in table 2. The results for case 1 are not shown since they are similar, except the partial efficiency for the calculation of mean properties which is higher in case 2 than in case 1. Table 2 shows that there are three tasks controlling the speedup. In fact, the pressure correction, the mean properties and

Table 2. Efficiencies per task in case 2.

Number of processors	2	4	6	8
Momentum equations	97.9%	93.4%	91.3%	92.4%
Pressure correction equation	77.3%	65.6%	58.0%	57.5%
k and ε equations	94.6%	90.0%	86.3%	88.0%
Other transport equations	94.7%	92.6%	89.8%	91.0%
Mean properties	89.0%	74.2%	72.8%	69.5%
Radiation	66.7%	42.3%	40.6%	32.6%

the radiation have much lower efficiencies than the remaining tasks, regardless of the case and number of processors. Let us now examine the reason for this behaviour.

All the three factors mentioned earlier for the decrease of efficiency with the increase of the number of processors coexist. But, depending on the task, one of these factors may be dominant. The key factor responsible for the low efficiency of the pressure correction equation is the large time required for communications among processors. This task requires much more communications than the remaining ones. In fact, the calculation of the coefficients of the discretized pressure correction equation requires that neighbouring processors exchange the pressure gradients in x, y and z directions, and the inverse of the main diagonal coefficients of the momentum equations, i.e., six arrays. In the preconditioned conjugate gradient solver another array needs to be exchanged in every iteration of that solver. The pressure correction field is exchanged before correcting the velocity and the pressure fields according to the SIMPLE algorithm. Finally, the corrected velocity and pressure fields, as well as the mass flow rates at the cell faces, are exchanged between neighbouring processors (seven arrays). In addition, there are several global operations: communication of the residual, reference pressure, and two inner products at every iteration of the conjugate gradient solver. Therefore, the communication time is an important fraction of the time spent in this task.

As far as the radiation is concerned, the main factor responsible for the decrease of the efficiency with the increase of the number of processors is the degradation of the convergence rate. This degradation is illustrated in Fig. 6 which shows the number of iterations used to solve the RTE whenever the radiation subroutine is called. These results have been obtained for case 2, but case 1 has an identical behaviour. If only one processor is used, the number of iterations rapidly drops from 8, at the beginning, to 3 where it stabilizes with a few oscillations. When the number of processors increases, so does the number of iterations, which drops during the course of the fluid flow equations loop, but slower than for one processor. Considering a complete run, the total number of iterations performed in the radiative calculations increases markedly with the number of processors, justifying the low efficiencies attained. This is consistent with previous work on this subject [16]. The size of the problem studied in [16] is smaller than the present one, but it was shown there that the number of iterations required to achieve convergence of the RTE is independent of the problem size. In the problem considered

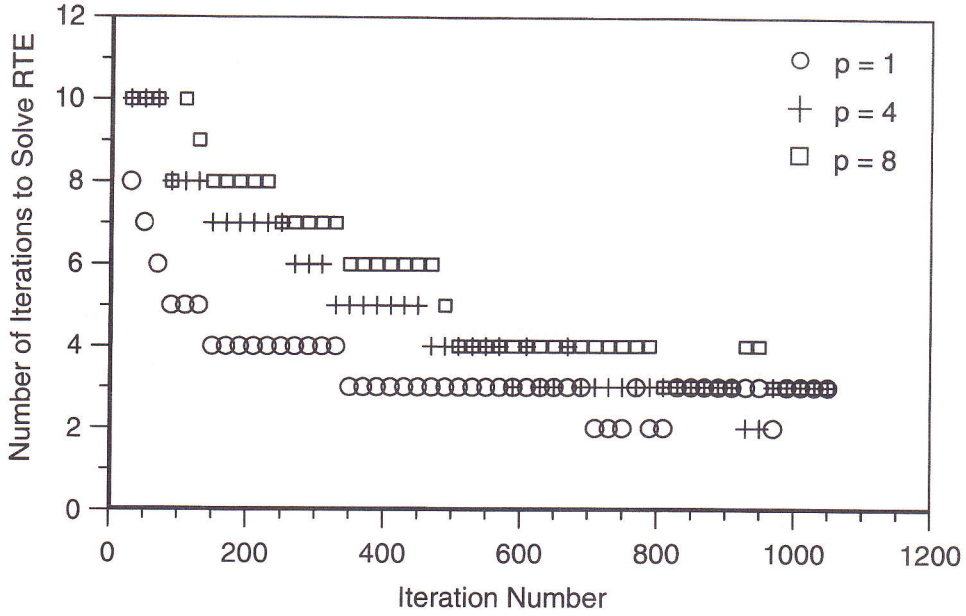

Figure 6. Number of iterations required to solve the radiative transfer equation per fluid flow iteration.

there, convergence was achieved in 8, 10, 14 and 19 iterations for 1, 2, 4, and 9 processors, respectively. The number of iterations will decrease if the initial guess is closer to the actual solution, but it will remain larger for a larger number of processors, according to the present findings.

It is interesting to note that the same parallelization strategy, the spatial domain decomposition, has a minor adverse influence on the convergence rate of the fluid flow equations, but a major influence on the convergence of the RTE. In fact, in the present problem the convergence of the fluid flow equations is achieved in about one thousand iterations. Owing to the non-linearity of the governing equations, only a weakly converged solution of the momentum equations, for example, is needed at each iteration, since the coefficients of the equations change every iteration. As a consequence, the data transfer at the interfaces of neighbouring processors does not cause a significant delay in the achievement of a converged solution. On the other hand, the convergence of the RTE is achieved in about 10 iterations. Moreover, in the case of a black enclosure an iterative procedure is not needed at all. In such an enclosure the iterative procedure is only needed to update the boundary conditions, but the information propagates exactly from a boundary to an opposite one in the case of a single processor. In the case of several processors the propagation of information from a boundary to an opposite one requires as many iterations as the number of processors between the two boundaries. This has a major impact on the convergence rate, as illustrated above.

Finally, let us investigate the low efficiency associated with the calculation of the mean properties. Figure 7 shows the time required to exchange data between

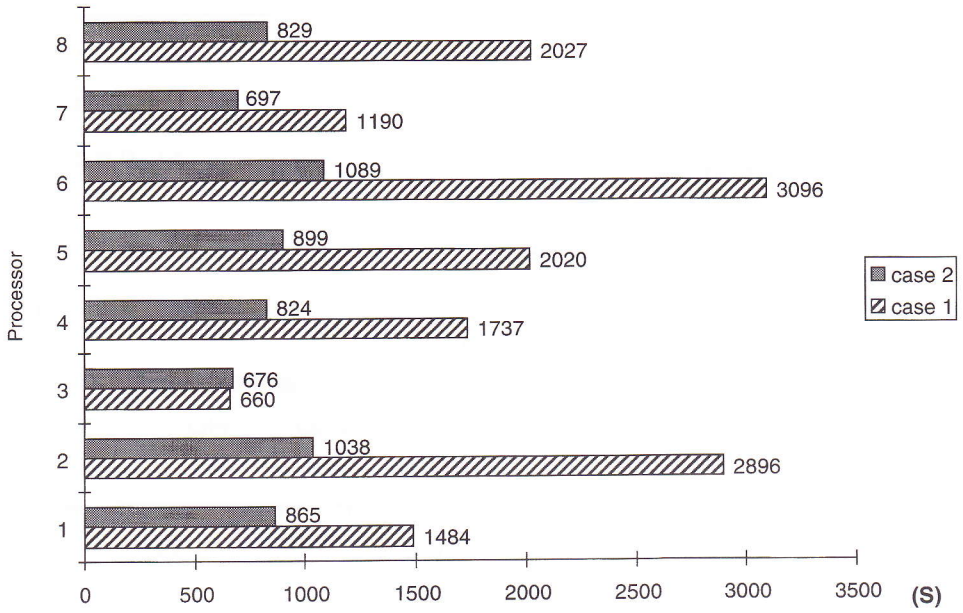

Figure 7. Time spent by each processor to exchange halo data with the neighbouring processors.

neighbouring processors, referred to as halo data time, for 8 processors and for both cases 1 and 2. The time required for global communications is not shown because it is typically one order of magnitude smaller. It can be seen that there is a large disparity of values from processor to processor, and that the halo data time is larger in case 1 for all but one processor. The dispersion of values is an indication of load balancing problems. These are expected in all tasks, since the number of grid nodes is not exactly the same in all the processors. In fact, the grid partitioning is that illustrated in Fig. 2, and so the subdomains assigned to processors 1, 2, 5 and 6 have $8 \times 19 \times 34$ active grid nodes while the subdomains assigned to the remaining processors have $8 \times 20 \times 34$ active grid nodes. But the dispersion would be small if this were the only reason, because the distribution of grid nodes among the processors is not far from uniform.

More insight into the problem is provided by the results summarized in table 3. They concern to case 2, but similar results could be presented for case 1. Both the global (t_{global}) and halo (t_{halo}) time spent in communications are listed, the second being dominant, as stated above. The longest halo time (1089s) is 61% larger than the shortest halo time (676 s). Table 3 also shows the halo time associated with the mean properties calculation ($t_{\text{halo, props}}$). It corresponds to the time required to transfer the density between neighbouring processors, after its calculation for all the grid nodes assigned to a processor. The difference $t_{\text{halo}} - t_{\text{halo, props}}$ is also given. Although t_{halo}, props is relatively small compared to t_{halo}, there are very large differences at $t_{\text{halo,props}}$ from processor to processor. The longest $t_{\text{halo, props}}$ (370 s) is 11.6 times larger than the shortest (32 s). This is a consequence of major

Table 3. Communication times per processor for case 2 using 8 processors.

Processor	1	2	3	4	5	6	7	8
$t_{\text{global}}(s)$	129	135	98	98	118	131	82	81
$t_{\text{halo}}(s)$	865	1038	676	824	899	1089	697	829
$t_{\text{halo, props}}(s)$	146	341	32	179	200	370	85	221
$t_{\text{halo}} - t_{\text{halo, props}}(s)$	719	697	644	645	699	719	612	608
Number of grid nodes with large fluctuations	2449	761	3436	1885	1835	510	2771	1521

load balancing problems in the mean properties calculation. The other subroutines exhibit much smaller load balancing problems, as can be seen from the values of $t_{\text{halo}} - t_{\text{halo, props}}$, which range from 608 s to 719 s, i.e., a ratio of only 1.18.

The computational load in the mean properties calculation is dominated by the time required to perform the integration of instantaneous values over the pdf range (Eq. 10–12). This integration is only performed if the local mixture fraction variance exceeds a prescribed tolerance, here taken as 10^{-4}. If the mixture fraction variance is smaller, it is assumed that the mean value of a property coincides with its instantaneous value. The number of grid nodes where the converged mixture fraction variance exceeds the specified tolerance is shown in table 3. It is easy to see that it changes quite significantly from processor to processor, and there is an obvious correlation between this number and the value of $t_{\text{halo, props}}$. As the number of grid nodes where turbulent fluctuations increases, $t_{\text{halo, props}}$ decreases.

This behaviour is illustrated in Fig. 8. A processor with a small number of grid nodes with large fluctuations rapidly finishes the integrations and is ready to communicate with the neighbours. A processor with a large number of grid nodes with large fluctuations spends more time to perform the integrations, but once these are finished it can communicate with the neighbours and does not need to wait for them. Therefore, the smallest value of $t_{\text{halo, props}}$ corresponds to the actual time needed for communications, while the higher values include the time during which the processors are idle. The load imbalance occurs both for cases 1 and 2, but it is more critical for case 1 since this case involves the integration of more quantities. The load imbalance would be eliminated if all the integrals were calculated *a priori* and stored, at the expense of an increase of the memory requirements.

The influence of the grid size on the parallel performance of the code is illustrated in Fig. 9 which shows the speedup as a function of the number of processors for three different grids in case 2. These are identified as a standard grid which is that used so far ($16 \times 39 \times 68$ grid nodes), a coarse grid with one half of the grid nodes ($8 \times 39 \times 68$), and a fine grid with twice as many nodes as the standard one ($32 \times 39 \times 68$). It can be seen that the speedup increases with the grid size for a fixed number of processors, as expected.

A reviewer has asked how the performance is expected to change for many more processors. Although this question cannot be addressed using the IBM-SP2 due to the unavailability of many more processors, some recent results were obtained

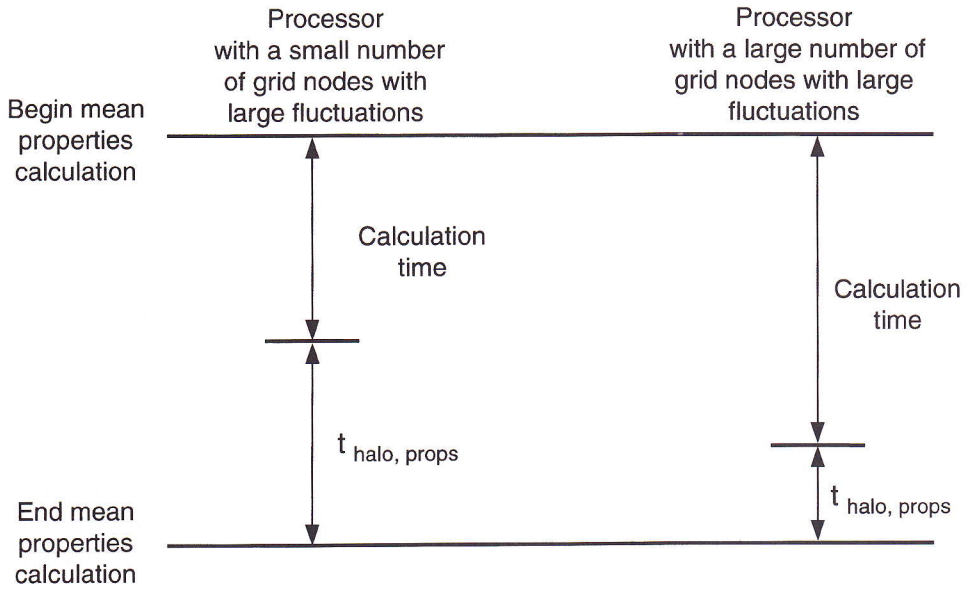

Figure 8. Time spent in calculations and in data exchange between neighbouring processors in the subroutine for the calculation of the mean properties.

using a Cray T3D and they are reported in [21]. They were obtained using a grid with 64 000 grid nodes. Neglecting the radiation and calculating *a priori* the mean density and temperature (case 2), the efficiency obtained for the first 80 iterations is 90.0%, 76.8% and 60.5% for 8, 32 and 128 processors, respectively. If the radiation is accounted for and the calculations proceed up to convergence, then the relative efficiency is 89.5% and 70.7% for 64 and 128 processors, respectively. This relative efficiency was calculated taking as a reference a solution for 32 processors. In general, it is found that the role of radiation becomes increasingly more important with the increase of the number of processors.

5. Conclusions

A power station boiler of the Portuguese Electricity Utility was simulated using a parallel code. The calculations were performed in an IBM-SP2 using 1, 2, 4, 6 and 8 processors. A speedup of 5.9 has been achieved using 8 processors, and a grid with about 42,000 grid nodes. It was shown that the storage of the mean density and temperature as a function of the mixture fraction and its variance allows a significant reduction of the computing time and an improvement of the parallel efficiency. Three tasks were identified as the main responsible for the drop of the efficiency with the increase of the number of processors. One is the solution of the pressure correction equation and the correction of the velocity and pressure fields which requires a large amount of communications. Another one is the solution of

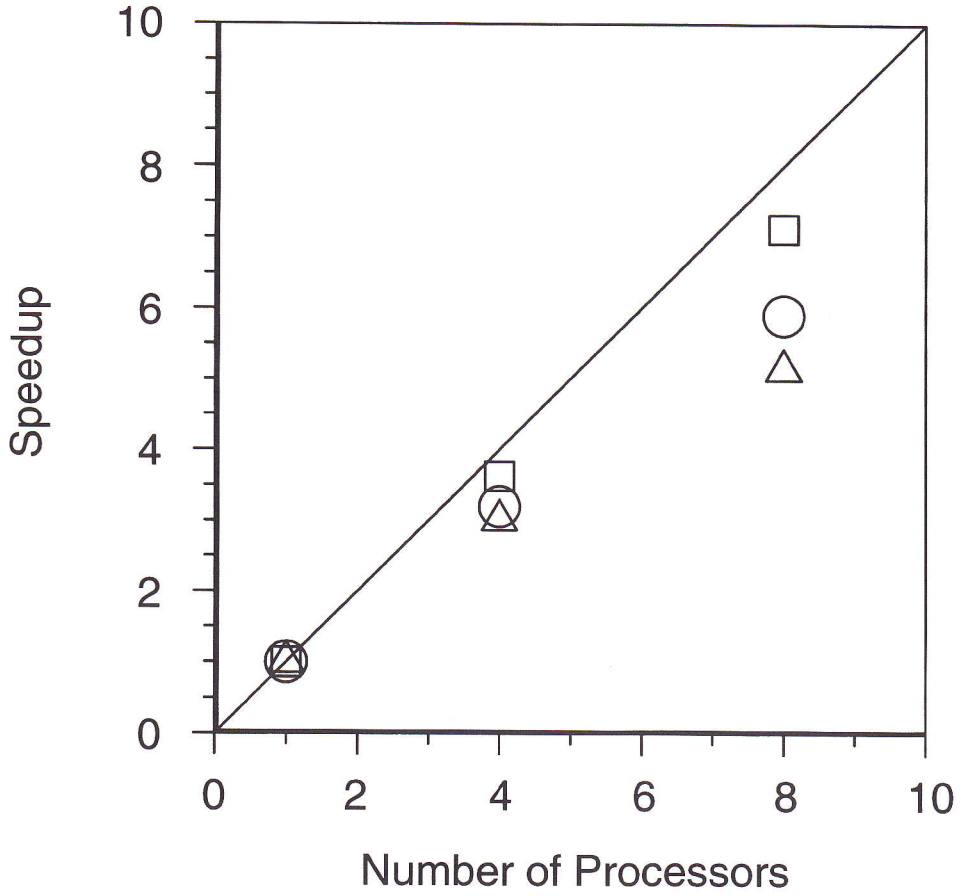

Figure 9. Speedup as a function of the number of processors and grid size (△—coarse grid, ○—standard grid, □—fine grid).

the radiative heat transfer equation whose convergence rate is highly influenced by the number of processors. The remaining one is the calculation of the mean properties via an integration of instantaneous values over the pdf range. It presents load balancing problems due to the uneven distribution of the grid nodes with large turbulent fluctuations among the processors.

Acknowledgments

The financial support of the European Commission within the framework of the project HP-PIPES of the workprogramme ESPRIT is acknowledged. The calculations were performed at an IBM-SP2 of CLRC (Council for the Central Laboratory of the Research Councils) of the Daresbury Laboratory, U.K. The permission to use their computational facilities is also acknowledged.

References

1. B. Risio, R. Schneider, U. Schnell and K. R. C. Hein, "HPF-Implementation of a 3D-Combustion Code on Parallel Computer Architectures Using Fine Grain Parallelism", in "Parallel Computational Fluid Dynamics: Algorithms and Results Using Advanced Computers," P. Schiano, A. Ecer, J. Periaux and N. Satofuka (Eds.), Elsevier Science, pp. 124–131 (1997).
2. B. Risio, J. Lepper, U. Schnell and K. R. G. Hein, "Microtasking Versus Message Passing Parallelization of the 3D-combustion Code AIOLOS on the NEC-SX4," Proc. Parallel CFD'97, May 19–21, Manchester (1997).
3. J. Lepper, R. Rühle, U. Schnell and K. R. G. Hein, "Performance Comparison of the Cray T3E/S12 and the NEC SX/4-32 for a Parallel CFD-code Based on Message Passing," Proc. Parallel CFD' 97, May 19–21, Manchester (1997).
4. B. E. Launder and D. B. Spalding, "The Numerical Computation of Turbulent Flows," Computer Methods in Applied Mechanics and Engineering, Vol. 3, pp. 269–289 (1974).
5. S. Gordon and B. V. McBride, "Computer Program for Calculation of Complex Chemical Equilibrium Compositions," Rocket Performance, Incident and Reflected Shocks and Chapman-Jouget Detonations', NASA SP-273 (1971).
6. S.-M. Jeng, L. D. Chen and G. M. Faeth, "The Structure of Buoyant Methane and Propane Flames," 19th Symposium (Int.) on Combustion, The Combustion Institute, pp. 349–358 (1982).
7. B. G. Carlson and K. D. Lathrop, "Transport Theory—The Method of Discrete Ordinates," in "Computer Methods in Reactor Physics," Gordon & Breach, New York (1968).
8. W. A. Fiveland, "Discrete-ordinates Solutions of the Radiative Transport Equation for Rectangular Enclosures," J. Heat Transfer, Vol. 106, pp. 699–706 (1984).
9. M. F. Modest, "Radiative Heat Transfer," Mc Graw-Hill (1993).
10. W. A. Fiveland, "The Selection of Discrete Ordinate Quadrature Sets for Anisotropic Scattering," HTD-160, ASME, pp. 89–96 (1991).
11. J. C. Chai, H. S. Lee and S. V. Patankar, "Treatment of Irregular Geometries using a Cartesian-coordinates-based Discrete-ordinates Method," HTD-Vol. 244, Radiative Heat Transfer: Theory and Applications, pp. 49–54 (1993).
12. P. J. Coelho, J. M. Gonçalves, M. G. Carvalho and D. N. Trivic, "Modelling of Radiative Heat Transfer in Enclosures with Obstacles," Int. J. Heat and Mass Transfer, Vol. 41, No. 4–5, pp. 317–326 (1998).
13. I. M. Khan and G. Greeves, "A Method for Calculating the Formation and Combustion of Soot in Diesel Engines," in "Heat Transfer in Flames," N. H. Afgan & J. M. Beer (Ed.), Scripta Book Co., chapter 25 (1974).
14. B. F. Magnussen and B. H. Hjertager, "On Mathematical Modelling of Turbulent Combustion with Special Emphasis on Soot Formation and Combustion," 16th Symposium (Int.) on Combustion, The Combustion Institute, pp. 719–728 (1977).
15. S. V. Patankar, "Numerical Heat Transfer and Fluid Flow," Hemisphere Publishing Corporation, New York (1980).
16. J. Gonçalves and P. J. Coelho, "Parallelization of the Discrete Ordinates Method," Numerical Heat Transfer. Part B: Fundamentals, Vol. 32, No. 2, pp. 151–173 (1997).
17. J. Cassiano, M. V. Heitor, A. L. N. Moreira and T. F. Silva, "Temperature, Species and Heat Transfer Characteristics of a 250 MWe Utility Boiler," Combustion Science and Technology, Vol. 98, pp. 199–215 (1994).
18. M. G. Carvalho, P. J. Coelho, A. L. N. Moreira, A. M. C. Silva and T. F. Silva, "Comparison of Measurements and Predictions of Wall Heat Flux and Gas Composition in an Oil-fired Utility Boiler," 25th Symposium (Int.) on Combustion, the Combustion Institute, pp. 227–234 (1994).
19. P. J. Coelho and M. G. Carvalho, "Evaluation of a Three-dimensional Model for the Prediction of Heat Transfer in Power Station Boilers," Int. J. Energy Research, Vol. 19, pp. 579–592 (1995).
20. P. J. Coelho and M. G. Carvalho, "Evaluation of a Three-dimensional Model of a Power Station Boiler," J. Eng. Gas Turbines and Power, Vol. 118, No. 4, pp. 887–895 (1996).
21. P. J. Coelho, "Influence of the Discretization Scheme on the Parallel Efficiency of a Code for the Simulation of a Utility Boiler," VECPAR'98, 3rd Int. Meeting on Vector and Parallel Processing, June 21–23, Porto, Portugal (1998).