PARALLEL MODEL OF FDS USED FOR A TUNNEL
FIRE SIMULATION

P. Weisenpacher, L. Halada, J. Glasa, and V. Šipková

Institute of Informatics, Slovak Academy of Sciences,
841 04 Bratislava, Slovakia

Abstract

Numerical modeling plays a significant role in many research areas filling in the gap between experimental and theoretical approach. During the past few years, rapid advances in computing hardware and technologies have allowed simulations of the most challenging and complex scientific problems. One of such discipline is applying the Computer Fluid Dynamics (CFD) theory in simulations of the spread of fire. Fire processes represent a very complex phenomenon consisting of the combustion, radiation, turbulence, fluid dynamics and other physical and chemical processes. In this paper, we describe the investigation of the Fire Dynamics Simulator (FDS) used for the simulation of burning in a short road tunnel. The FDS system was compiled to evaluate two parallel variants: the pure MPI, and the hybrid MPI+OpenMP, both executed on a quad-core compute cluster. We have analyzed several runs of the fire simulation which differ from each other in the form of computational domain decomposition, and in the number of processors assigned to the program execution.

1 Introduction

Recent large-scale catastrophic events, such as extensive forest fires, fires in high rise buildings, in large-scale parking spaces, or in tunnels, cause enormous material damages and sometimes lead to tragic losses of human lives. The preparedness to alternative rescue procedures against the extreme events is an inevitable prerequisite to the mitigation and prevention of devastating consequences of such events. Over the last decade, the computer simulation of a fire has become an effective tool for the preparation of rescue work in the given environment, as well as for the fire suppression. They enable to visualize the spread of the fire and its basic parameters, such as the temperature, air velocity, heat release rate, smoke transfer. In some cases, they facilitate to verify the strategy and effectiveness of the fire suppression which may be very useful for the fireman staff.

Fire processes are very complex phenomenon consisting of the combustion, radiation, turbulence, fluid dynamics and other physical and chemical processes. Consequently, applying the Computer Fluid Dynamics (CFD) theory requires a good understanding of all these issues, as well as, the qualified specification of all input parameters relevant to burning materials, in order to define the impact of the environment and correct initial and boundary conditions. Moreover, the existing software tools for making fire simulations are composed of many computational procedures based on a discretization of relevant equations, and it is important to know limitations of all these numerical procedures. In spite of that, a computer simulation of a fire is the most economical and feasible method to research and analyze fire processes in a given environment with changing conditions. A very critical factor affecting the possibility of the wider use of the fire computer simulation is the computational complexity. The realization
of the fire simulation arose in high buildings, car parks, or in tunnels on a single-processor can take many days. Therefore, the employment of high performance parallel computing is the only effective way to ensure the real usability of computation- and/or data-intensive CFD tools. At present a number of tunnels is under the course of construction and the length and complexity of tunnel systems is increasing. It is evident that the traditional empirical approach to design fire protection and security system components is inadequate. There are already quite a lot of publications dealing with the computer simulation of fires in tunnels comparing results with the corresponding full-scale fire experiments.

In [1], fourteen full-scale fire tests were carried out during 2000-2001 in the Second Benelux Tunnel near Rotterdam. Tests were designed to assess the tenability conditions for escaping motorists from the road tunnel in case of a fire, and to examine the effect of mitigating measures on these conditions. In particular, effects of the detection system, mechanical ventilation and sprinklers were also investigated. In papers [2, 3, 4, 5] numerical simulations of different fires in tunnels by CFD software were presented. The objective of these simulations was to quantify temperature distribution, smoke movement from a burning vehicle in tunnel and its velocity, and so on. Extensive studies about the influence of the ventilation rate on the fire growth and peak heat release rates in tunnel fires have been published in [6, 7]. Five possible means of the fire spread from a vehicle to vehicle, often over large distance, such as the flame impingement, flame spread, spontaneous ignition, fuel transfer (including flow of burning liquid fuels and fire brands) and explosion are analyzed. Relatively little research has been devoted to these mechanisms. Therefore, research of all aspects of these mechanisms, as well as the study of the appropriate ventilation and its impact on the fire size, fire spread and fire growth is inevitable.

2 FDS Overview

The Fire Dynamics Simulator (FDS) [8], is a Computational Fluid Dynamics (CFD) model of fire-driven fluid flow. FDS solves numerically a form of the Navier-Stokes equations appropriate for the low-speed, thermally-driven flow with an emphasis on the smoke and heat transport from fires. Throughout its development, FDS has been aimed at solving practical fire problems in fire protection engineering, while at the same time providing a tool to study the fundamental fire dynamics and combustion.

The FDS model has been designed to run on a variety of platforms and OS, either serially or in parallel, with or without multi-threading. FDS uses MPI (Message-Passing Interface) [9] to allow running a single FDS job on a computing cluster. The main idea is that the FDS domain is broken up into multiple meshes, and then the flow field in each mesh is computed as an individual process on the individual computer. MPI handles the transfer of information between these processes. Usually, each mesh is assigned its own MPI process, although it is possible to assign multiple meshes to a single process. In this way, large meshes can be computed on dedicated processors, while smaller meshes can be grouped together in one process running on a single processor. FDS supports a two-level parallelization strategy that first decomposes each computational domain into meshes for distributed memory, and then, within each mesh the multi-threading is applied for shared memory. FDS uses the OpenMP
library [10] to implement the multi-threading.

2.1 FDS Mathematical Model

FDS model solves a form of conservation equations for low speed, thermally driven flow. Smoke and heat transfer from fires is the main concern of this program system, which also includes thermal radiation, pyrolysis, combustion of the pyrolysis products, flame spread and fire suppression by sprinklers. The basic equations for the conservation mass, species, momentum and energy are the following:

\[
\begin{align*}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= \dot{m}_b'' \\
\frac{\partial}{\partial t} (\rho Y_\alpha) + \nabla \cdot (\rho Y_\alpha \mathbf{u}) &= \nabla \cdot (\rho D_\alpha \nabla Y_\alpha + \dot{m}_\alpha'' + \dot{m}_{b,\alpha}'') \\
\frac{\partial}{\partial t} (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) + \nabla p &= \rho \mathbf{g} + \mathbf{f}_b + \nabla \cdot \tau_{ij} \\
\frac{\partial}{\partial t} (\rho h_s) + \nabla \cdot (\rho h_s \mathbf{u}) &= D_p \frac{\partial}{\partial t} + \dot{q}''' - \dot{q}''_b - \nabla \cdot \dot{q}'' + \epsilon
\end{align*}
\]

(1)

Here \(\dot{m}_b'' = \sum_\alpha \dot{m}_{b,\alpha}''\) is the production rate of species by evaporating droplets or particles, \(\rho\) is the density, \(\mathbf{u} = (u, v, w)\) is the velocity vector, \(Y_\alpha\), \(D_\alpha\), and \(m_\alpha''\), \(\dot{m}_{b,\alpha}''\) are the mass fraction, diffusion coefficient and mass production rate of \(\alpha\)-th species per unit volume, respectively. \(p\) is the pressure, \(\mathbf{f}_b\) the external force vector, \(\tau\) the viscous stress tensor, and \(h_s\) is the sensible enthalpy. The term \(\dot{q}'''\) is the heat release rate per unit volume from a chemical reaction, and \(\dot{q}''_b\) is the energy transferred to the evaporating droplets. The term \(\dot{q}''\) represents the conductive and radiative heat fluxes. To these four equations another two equations, namely, the pressure equation and equation of state are added:

\[
\begin{align*}
\nabla^2 H &= -\frac{\partial}{\partial t} (\nabla \cdot \mathbf{u}) - \nabla \cdot F \\
p &= \frac{\rho RT}{W}
\end{align*}
\]

(2)

The value \(H\) represents the total pressure divided by the density. \(R\) is universal gas constant, \(T\) temperature, and \(W\) molecular weight of the gas mixture. Thus, we have for six unknowns, all functions of three spatial dimensions and time: the density \(\rho\), three components of the velocity \(\mathbf{u} = (u, v, w)\), the temperature \(T\) and the pressure \(p\), the set of six equations. These equations must be simplified in order to filter out sound waves, which are much faster than the typical flow speed. The final numerical scheme is an explicit \textit{predictor-corrector finite difference} scheme, which is second order accurate in space and time. The flow variables are update in time using an explicit second-order \textit{Runge-Kutta} scheme. This part of the algorithm is highly suitable to the parallel computation.

Boundary conditions are prescribed on the walls and vents. All input data for a simulation are required in the form of a text file in the prescribed format, which describes the coordinate system, geometry of the domain and its location in the given coordinates, mesh resolution obstacles, boundary conditions, material properties and different simulations parameters. Important limitations of the program is that the domain and all obstructions representing real
objects, which can burn, heat up, conduct heat etc., should be rectangular, conforming with the underlying grid.

The overall computation can either be treated as a Direct Numerical Simulation (DNS) or as a Large Eddy Simulation (LES). The description of the numerical schemes used for the solution all equations is completely introduced in [11]. It is well known, that the accuracy of simulating fires highly depends on the resolution of the grid size. For this reason, many researchers investigate effects of the grid size on the different fire characteristics, such as the flame height, radiative heat fluxes, temperature distribution, and so on. It was observed for different grid sizes, the predicted flame heights for 20, 30 and 38 cm pool fires increase with the increasing the ratio $D^*/\Delta d$, where $D^*$ is a characteristic diameter and $\Delta d$ is the grid size, (i.e. with decreasing grid size) and would approach to a converged value. The another serious problem is the computation of the total pressure value $H$ which fulfills the following equation:

$$\nabla^2 H = -\frac{\partial \nabla \cdot u}{\partial t} - \nabla \cdot f$$ (3)

This equation solved by numerical scheme gives the Poisson equation, which is solved in FDS by a direct FFT-based solver that is part of a library CRAYFISHPAK [12]. The precision of this solution is very important and has marked an influence on simulation outputs.

Parallel versions of FDS make possible to simulate a fire in large areas, (buildings, garages, long tunnels, etc.). However, decomposing a large computational domain onto tenths and more sub-domains appears as a problem which must be seriously analyzed. Especially, coupling of the pressure solver across the mesh boundaries in a multi-mesh simulation must be tested and verified. Experiments presented in this paper confirm that the different scalability of the computational domain has an impact on the computational value of the temperature, and consequently, also on another output values.

3 Scenario of Fire in Tunnel

We have constructed a two-lane road tunnel model with dimensions 10 m x 180 m x 7 m (width x length x height, xyz) with two fans located on the tunnel ceiling at the distance 50 m and 140 m from the left entrance of the tunnel. The scheme of the tunnel is outlined in Fig. 1. The Cartesian coordinate system xyz chosen for the tunnel was x [-5.0, 5.0], y [0.0, 180.0] and z [-5.0, 2.2] measured in meters. The fire source in the simulation was represented by burning of a flammable liquid in a pool with dimensions 2 x 3 m placed in the distance 92 m from the left entrance of the tunnel, 1.1 m above the floor level, with the Cartesian coordinates x [0.0, 3.0] m, y [92.0, 94.0] m, z [1.1, 1.1] m. The maximum heat release rate per unit area (HRRPUA) of the fire was 1666.67 kW and the total heat release rate (HRR) was 10 MW. During the simulation, the fire did not spread along the tunnel, no other flammable obstacles were included in the simulation.

The total duration of the simulation was 150 s. The initial air temperature in the whole tunnel was set to 20°C. The dynamics of the fire source and the tunnel ventilation was simulated as follows. At the beginning (t = 0 s), both fans started to blow the air with the velocity of 5 m/s in y direction. At t = 40 s, the fire started with linearly increasing power, so
that it achieved the above mentioned maximum HRR at $t = 45$ s and then it was not changed until the end of the simulation. 10 s after the fire appeared, i.e. at $t = 50$ s, both fans started to increase their power linearly achieving the maximal air velocity of 20 m/s at $t = 55$ s, which was not changed until the end of the simulation.

Various control devices were installed inside the tunnel in order to record mean values of gas phase quantities (soot volume fraction, visibility, temperature and carbon monoxide mass fraction) inside small testing cube-like volumes placed under the ceiling of the tunnel and at the level of human head (see Fig. 1). The slices of gas temperatures, oxygen and carbon monoxide mass fractions were also recorded for several planes. The wall temperature of the tunnel ceiling was detected above the fire.

4 Running on the Cluster

To evaluate the simulation we ran a number of experiments which were carried out on the HP Blade Cluster at the Institute of Informatics, Slovak Academy of Sciences, Bratislava. This cluster consists of 16 compute nodes, each comprising of two quad-core 2.93 GHz processor Intel Xeon X5570 with 8 MB cache. Each node contains 24-48 GB of RAM. Nodes are connected by the Infiniband interconnection network with the bandwidth of 40 Gbit/s per link and direction. Simulations were realized through the FDS version 5.5.3. For comparison purposes we configured three parallel variants: a pure MPI [9], pure OpenMP [10] and hybrid MPI+OpenMP. For the compilation we used GNU v. 4.1.2 and 4.4.0 (gcc, gfortran, OpenMP) and the Open MPI v. 1.4.

In order to automate the process of simulation runs with a minimum of human intervention, for each configuration we have developed a fds-manager script (written in Shell) which is responsible for the completion of the following actions:

- It accepts the input arguments specifying the input file and required running configuration: the number of nodes, the number of cores, and eventually the number of MPI
processes and number of threads. Arguments are passed to the subsequent operations.

- It produces the corresponding *job-submission* script which serves as an input to the Portable Batch System (PBS) [13].
- It provides for the execution of the FDS simulation using the previously generated *job-submission* script.

To submit and start a batch job on a compute cluster the `qsub` command is used, to which the *job-submission* script is passed as argument. Typically, the *job-submission* script for the PBS batch server is a Shell-script including PBS commands. A number of options on `qsub` allow the specification of attributes which may affect the behavior of the job execution. When running the pure MPI version of the FDS, MPI processes are allocated by *slots-strategy* (default), where one MPI process is mapped per processor-core. When running the hybrid MPI+OpenMP version, MPI processes are enforced to be allocated so, that each MPI process is mapped per processor which has a sufficient number of cores available, that must be reserved to fork OpenMP threads.

5 Performance Results

In this section we summarize the most significant results obtained from simulations of the tunnel fire scenario described in section 3.

<table>
<thead>
<tr>
<th>Simulation</th>
<th>Mesh Division Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M</td>
<td>1 mesh: 180 m/10 cm</td>
</tr>
<tr>
<td>1M-omp</td>
<td>1 mesh: 180 m/10 cm</td>
</tr>
<tr>
<td>3Ma</td>
<td>3 meshes: 60 m/10 cm</td>
</tr>
<tr>
<td>3Mb</td>
<td>1 mesh: 48 m/10 cm; 1 mesh: 60 m/10 cm; 1 mesh: 60 m/10x10-14x10 cm</td>
</tr>
<tr>
<td>4M</td>
<td>4 meshes: 45 m/10 cm</td>
</tr>
<tr>
<td>8Ma</td>
<td>8 meshes: 22.5 m/10 cm</td>
</tr>
<tr>
<td>9M</td>
<td>1 mesh: 7.5 m/10 cm; 7 meshes: 22.5 m/10 cm; 1 mesh: 15 m/10 cm</td>
</tr>
<tr>
<td>10M</td>
<td>1 mesh: 60 m/10 cm; 8 meshes: 7.5 m/5 cm; 1 mesh: 60 m/10 cm</td>
</tr>
<tr>
<td>24M</td>
<td>24 meshes: 7.5 m/10 cm</td>
</tr>
<tr>
<td>48M</td>
<td>48 meshes: 3.75 m/10x5x10 cm</td>
</tr>
<tr>
<td>8Mb-omp</td>
<td>1 mesh: 64 m/10 cm; 2 meshes: 8 m/5 cm; 4 meshes: 9 m/5 cm; 1 mesh: 64 m/10 cm</td>
</tr>
</tbody>
</table>

Several instances of the simulation were performed which differ from each other in the way how the whole computational domain was divided into particular computational meshes and/or in the size of cells inside the mesh. Descriptions of the meshes corresponding to individual simulations are listed in Tab. 1. Each computational mesh was assigned to one
CPU core. The characteristic fire diameter is $D^* = 2.4\text{ m}$. Therefore, the mesh resolution used in all simulations is fine.

Table 2: Tunnel fire simulations characteristics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1M</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>12.96</td>
<td>33041</td>
<td>377.2</td>
<td>375.8</td>
<td>1.00</td>
</tr>
<tr>
<td>1M-omp</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>12.96</td>
<td>32881</td>
<td>247.1</td>
<td>247.0</td>
<td>1.53</td>
</tr>
<tr>
<td>3Ma</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>12.96</td>
<td>31905</td>
<td>172.4</td>
<td>169.5/165.6</td>
<td>0.73</td>
</tr>
<tr>
<td>3Mb</td>
<td>3</td>
<td>-</td>
<td>3</td>
<td>12.96</td>
<td>31358</td>
<td>150.9</td>
<td>149.8/145.5</td>
<td>0.83</td>
</tr>
<tr>
<td>4M</td>
<td>4</td>
<td>-</td>
<td>4</td>
<td>12.96</td>
<td>32839</td>
<td>130.9</td>
<td>128.6/125.5</td>
<td>0.72</td>
</tr>
<tr>
<td>8Ma</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>12.96</td>
<td>31277</td>
<td>68.2</td>
<td>67.2/63.2</td>
<td>0.69</td>
</tr>
<tr>
<td>9M</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>12.96</td>
<td>30947</td>
<td>69.8</td>
<td>68.9/-</td>
<td>0.68</td>
</tr>
<tr>
<td>10M</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>12.96</td>
<td>60727</td>
<td>313.6</td>
<td>309.6/289.4</td>
<td>0.80</td>
</tr>
<tr>
<td>24M</td>
<td>24</td>
<td>-</td>
<td>24</td>
<td>12.96</td>
<td>31206</td>
<td>32.6</td>
<td>32.0/26.9</td>
<td>0.48</td>
</tr>
<tr>
<td>48M</td>
<td>48</td>
<td>-</td>
<td>48</td>
<td>12.96</td>
<td>68759</td>
<td>63.2</td>
<td>61.5/53.8</td>
<td>0.50</td>
</tr>
<tr>
<td>8Mb-omp</td>
<td>8</td>
<td>4</td>
<td>32</td>
<td>39.17</td>
<td>60054</td>
<td>203.8</td>
<td>199.2/194.1</td>
<td>1.48</td>
</tr>
</tbody>
</table>

Various mesh divisions used have a direct impact on the computational load and performance. Basic characteristics of all simulation variants are summarized in Tab. 2. Columns 2-4 specify the number MPI processes, OpenMP threads, and CPUs employed. In the 5th and 6th column, the total number of rectangular cells and the maximal number of cells per mesh is given. The column 7 contains the number of time steps needed for the simulation of 150 s. Recall that all processors compute in a fully synchronized manner, i.e., the same value of time step is used in each processor for the integration of the system of PDEs in a given parallel cycle. Hence, the difference in the number of parallel steps between particular variants can only reflect the difference in the size of the actual time steps used in the integration. Therefore, the time step value for 10M, 48M and 8Mb-omp is approximately one half of the minimal value for the rest of simulations. Consequently, the total execution time of these simulations (columns 8 and 9) is proportionally larger. The time step is directly connected to the mesh resolution, which for 10M, 48M and 8Mb-omp is exactly doubled at least in one direction, compared to the rest of simulations. Taking into account these considerations, we can assume that the overall simulation time depends on the maximal cells number per mesh ($N_{\text{max}}$) and minimal resolution used ($Res_{\text{min}}$) as $t_{\text{sim}} = c \cdot N_{\text{max}} / Res_{\text{min}}$. Column 10 depicts the ratio $c(1M) / c$, which roughly characterizes the efficiency of parallelization from the time savings point of view.

Fig. 2 shows the smoke development at the 50th, 57th, 88th and 150th second of the simulation 1M. It demonstrates that smoke was pushed away from the tunnel region on the left of the fire source by the ventilation. Mesh boundaries have the same effect as permeable barriers and slow down the heat and smoke transfer slightly. At the device point 3, there is a visible lag in temperature increase depending on number of meshes of particular simulations.
Figure 2: Smoke development

(see Fig. 3). The maximum lag is about 1 s (in the distance about 27 m from the fire source). Although this lags affect the simulation precision, their effect is not critical.

Some quantities show different behavior (see Fig. 4). The sequential 1M behavior seems to have larger fluctuations in some phases of the burning, but the shape of the curve seems to be smoother than in the case of parallel simulations. It is probably due to numerical approach used to resolve the processes at the boundaries of the meshes. The simulation 48M used very fine division of the computational domain into numerical meshes. However, the impact of this division on overall simulation results was not considerable. All simulations provide reasonable rough estimates of relevant quantities.

6 Conclusion

We have run several implementations of the FDS simulation of the smoke transfer in a road tunnel. Though our main goal was to investigate the impact of partitioning the computational domain on the simulation results reliability, we struggled along with to find a FDS scenario exhibiting the best performance. FDS simulations show the realistic smoke transfer behavior and confirm that also parallel versions of FDS are able to provide results with a reasonable precision even for a larger number of meshes. From the obtained results (see columns 8 and 10 in Tab. 2) is apparent, that the MPI version turned out to be the most efficient. In comparison with the sequential code it shows a speedup of 2-11 times, depending on the domain division. The performance improvement of the OpenMP implementation is not so significant, a run using 8 threads offers a speedup of 1,5 times. However, a combination of both parallelization strategies can be beneficial, it requires additional testing. Also further research is needed to evaluate the precision of parallel fire scenarios, especially the influence of numerical approach describing the processes on mesh boundaries.
Figure 3: The impact of the mesh division on temperature time behavior lags

Figure 4: The impact of the mesh division on overall simulation results
Acknowledgements. The authors would like to thank to our colleagues M. Dobručký, J. Astaloš and P. Slížík for their technical support during FDS simulations. This paper was partially supported by Slovak Scientific Research Agencies APVV (project APVV-0532-07), VEGA (project VEGA 2/0216/10) and ASFEU (project OPVV CRISIS ITMS 26240220060).

References


[12] CRAYFISHPAK - A vectorized form of the elliptic equation solver FISHPAK, originally developed at the National Center for Atmospheric Research (NCAR) in Boulder, Colorado.