Fire in a Storage: From Simulation to Virtual Environments

Andreas Gerndt, Malcolm Hutson, Paul Fung, Carolina Cruz-Neira

(1) Louisiana Immersive Technologies Enterprise, Lafayette, LA, USA

Abstract

Carrying out commercial projects for industrial customers in the field of Computational Fluid Dynamics (CFD) simulations and interactive exploration of the results is always a challenge. The project time is usually limited, accurate simulation is a complex task with a lot of unknowns, and interactive CFD post-processing with very large-scale datasets is almost impossible on contemporary visualisation computers even with the availability of efficient rendering hardware. In this case study, we describe which steps we performed and which challenges we had to solve in order to apply a complex fire simulation in a warehouse and to enable interactive post-processing in virtual environments. This comprises our approach to analyse and evaluate the chosen CFD solver FDS on Itanium-based supercomputers, our enhancement of FDS to be more efficient and generate data we could use directly for the post-processing, and the way we visualised data from different sources for real-time exploration in immersive virtual environments. Furthermore, we present our software layer VRFlowVis that is based on VR Juggler, VistaFlowLib, and VTK, which offers a simplified interface for application developers to bring their projects into cluster-based as well as multi-pipe rendering environments.

Keywords: Flow Simulation, Distributed Post-Processing, Modelling, Virtual Reality

1 Introduction

Companies offering storage to third parties must prove by law that their buildings were constructed by following fire prevention and protection requirements. For that purpose, prototype setups of racks in real buildings are set fire in order to test the efficiency of e.g. the fire protections systems such as sprinkler systems. Such real accidental fire evaluations are very expensive and time intensive. Sometimes, the whole process must be repeated to show that modifications of the building or the sprinkler system improved the fire extinguishing.

An alternative solution might be the replacement of real fire experiments with computer simulations. The case study discussed in this paper deals with an accidental fire in a warehouse and shows all steps from simulation pre-processing to interactive exploration of large-scale results in virtual environments. In this particularly interdisciplinary project, we collaborated intensively with experts in the field of numerical fire simulations, with fire marshals, and of course, with the customer.

First of all, we had to understand how fire can be simulated to find an accurate solver for our CFD problem. There are three principal approaches to model fire. The analytical model just uses basic equations to define fire development. It is well suited for early stage of fire but has some considerable limitations. The model covers only simple setups and important fire properties such as extinctions or flashovers are not supported. The zone model, instead, allows more complex fire simulations. Two or more distinct horizontal layers are specified, containing

regions of uniform concentrations. However, this improved model cannot model fire in small enclosures properly. The third model, the field model, overcomes this limitation. It is based on numerical solvers of transient conservation equations of mass, momentum, and energy and specifies further start and boundary conditions. The disadvantage of this model is its complexity which requires an accurate preparation and evaluation of the pre-processing phase performed by qualified individuals. While the analytical model can present results almost immediately and even the zone model yield results in the order of seconds, the field model needs hours or days for the computation. (Rein et al, 2006)
The following sections are structured as follows. Section 2 describes the Fire Dynamics Solver (FDS) used for the fire simulation. Section 3 summarises our evaluation with a multi-block example case and some simulation aspects of our rack storage fire. The geometrical 3D modelling, performed modification of FDS, and the used post-processing algorithms for an interactive analyse are presented in Section 4. Finally, we present the core features of our developed Virtual Reality (VR) framework before we conclude the main issues and point out future steps.

2 Fire Dynamics Solver

The Fire Dynamics Solver (FDS) is a field model developed by National Institute of Standards and Technology (NIST). It is a widely accepted open-source tool written in Fortran aimed at fire simulations in buildings (McGrattan et al, 2007). A combination of large eddy simulations (LES) and the mixture fraction model is used to solve the Navier-Stokes equations (NSE), which couples several transient conservation equations mentioned before (Rein et al, 2006).
FDS is stable, reliable, and highly accurate when only one regular grid is used for the flow field definition. The simulation run time depends then on the grid size, the number of input parameters, the boundary conditions, and the flow field complexity. Not all configurations will converge. Therefore, in general one has to spend a considerable amount of time to find a “good” solution by analysing why a configuration did not yield an expected result, stops too early because of numerical instabilities, or does not terminate at all. To speed up simulation iterations, FDS is being enhanced to support parallelisation. This can only be carried out if the domain is subdivided into multiple grid blocks. Then, each processor computes the fire dynamics in one grid independently. As each block is assigned to its own process, the number of parallel processes is restricted to the number of available blocks.
The multi-block domain decomposition does not require fully matching grid blocks, which makes it more flexible. Otherwise, this could result in so-called T-vertices at grid boundaries, introducing numerical inaccuracies (Grosso et al, 1996). Furthermore, as we would find out in discussions with FDS users and developers, pressure is only solved locally which, in the multi-block case, can result in a lost of global conservation of mass. Analysing FDS results, discontinuities at the boundaries of neighboured blocks can be detected, even if some modelling constraints (McGrattan et al, 2007) are considered. An improvement might be achieved by overlapping the grids, but the current recommendation is to avoid too many grids. This recommendation, however, restricts our goal of massive parallel computation. As this is still ongoing work, we expect a considerable improvement of multi-block computations in future FDS versions. In order to evaluate the current accuracy, we analysed multi-block results using a Townhouse test-case offered by NIST (cf. Section 3.2).

2.1 Smokeview

The developers of FDS also offer a post-processing toolkit called Smokeview. Actually, FDS carries out the time-consuming post-processing tasks during the simulation run. Therefore, Smokeview's primary job is to visualise the pre-computed quantities as a time series. Special features include smoke visualisation and transparent rendering of other fire attributes. Although it also offers a stereo mode, this does not work on our clustered or multi-pipe visualisation systems. Interactive exploration approaches and immersive analysis are not offered. (Forney, 2007)

3 Fire Simulation

3.1 Computing Environment

At our facility, an SGI Altix 350 cluster with 11 nodes, each equipped with 32 Itanium-2 (Madison 9M) 1.5 GHz processors and 32 GB main memory, is installed. All nodes are connected via Infiniband and are operated by SUSE Linux Enterprise Server 10, which is a 64-bit operating system. The second supercomputer available at LITE is an SGI Altix 4700 with 80 Itanium-2 (Montecito) 1.6 GHz dual-core processors. This system, called Beast, comes with the tremendous total amount of 4.1 Terabytes shared main memory. The latter system should even be sufficient for in-memory computation of most common simulation sizes currently in use.

3.2 Pre-Evaluations

We installed and compiled the then current version 4 of FDS on the Itanium-based 64-bit OS systems. Unfortunately, it turned out that the simulation with that solver version was not reliable. Therefore, a then current beta-version of the successor (version 5, release candidate 6) was used.

In order to prove that our 64-bit Itanium-2 systems are capable to carry out the fire simulation accurately with FDS, we started at first with an evaluation phase. Instead of using an early input data file from our commercial project, a prototype test case offered by FDS was used. It describes a fire in a townhouse and composed of 4 full-matching blocks. We furthermore sub-divided each block once more which yielded a second test input file with 8 blocks. This could be used to evaluate the run-time behaviour and the accuracy of the parallel FDS version on the Altix supercomputers.

As expected, the method by which the underlying parallelisation toolkit (MPI) was configured and compiled has a particular influence on the runtime performance of the solver. We used MPICH by Argonne National Laboratories (v2.1.0.5p4) which is an MPI-2 implementation (Gropp et al, 1999) and comes with different communication devices. The default device uses the TCP/IP socket protocol. One alternative, which is of interest for our systems, is a device called Nemesis that communicates via shared memory.

MPICH as well as FDS were compiled using Intel's compiler suite version 9.1.039. It offers optimisations for the Itanium processors series, inter-procedural code optimisation, and auto-parallelisation capabilities for Fortran code. The latter functionality parallelises the code independently to the Message Passing approach. We also considered using OpenMP which is offered by these Intel compilers, but due to the compactness of the FDS code and in order to avoid unwanted side effects on the results, we did not use OpenMP.

The simulation time for the townhouse fire was set to 300 seconds. The simulation produced 3000 equidistant snapshots and stored them on the file server. We used Smokeview to compare the results we received from 4 block and 8 block decompositions. We could identify smaller deviations especially in the living room and the staircase area (cf. Figure 1). We discussed this result with our customer and fire simulation consultants and found that this was still satisfactory considering the complexity and the simulation's length of time.

3.3 Configuration and Measurement

For runtime evaluation, we carried out the simulation under eight major configurations. Besides a simulation run with the standard sequential FDS version, we also compared parallel versions with different optimisation flags and different MPI libraries, all of which were linked statically

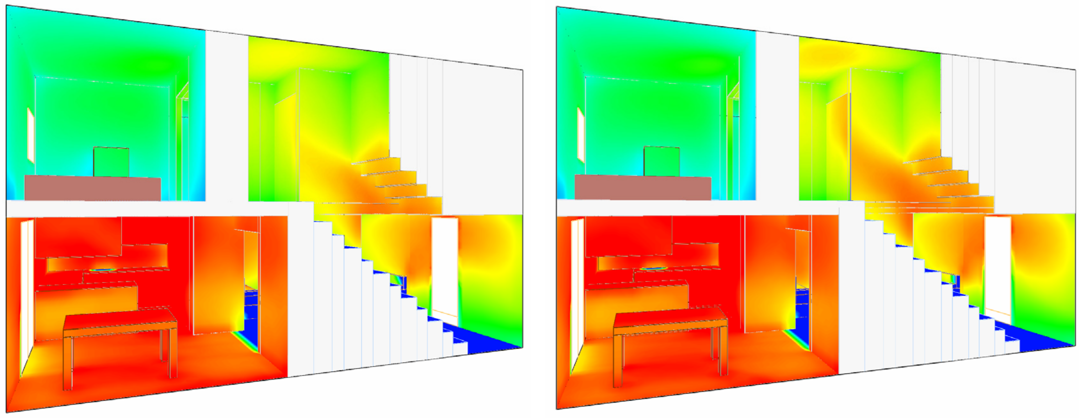

Figure 1. Fire simulation result (wall temperature) in a townhouse flow field at frame 1247 with a domain decomposition of 4 blocks (left) and 8 blocks (right).

to the solver. All results were analysed visually to ensure solution consistency. Using all possible optimisation features mentioned above yielded the most satisfying speed-up. The runtime measurements for these optimised versions linked to different MPI libraries are depicted in Table 1:

Table 1. Wall Clock and Processor Core Time in Seconds.

	Wall Clock	Core Time
Sequential Runtime	90589.63	90571.4
Altix 350, TCP/IP Communication, 4 Processes	35205.05	127582.0
Beast, TCP/IP Communication, 4 Processes	39007.40	138134.4
Altix 350, TCP/IP Communication, 8 Processes	23296.97	138030.6
Beast, TCP/IP Communication, 8 Processes	20465.07	108208.8
Beast, Nemesis Communication, 8 Processes	14121.84	112205.5

In this table, the wall clock time and the processor core time are rated. The wall clock time is the time elapsed between the time the solver was started and time it terminated. In comparison, the core time measures the sum of the time of all processors used to compute the flow field. As the processors can run in parallel, it shows the efficiency of the parallelisation, balancing problems, and communication overhead. In an optimum parallel simulation run, the core time would be the number of processors multiplied by the wall clock time.

The sequential runtime of 25 hours and 10 minutes offers the basis for the efficiency evaluation of the parallel runs. An efficiency of 1.0 would yield in a wall clock time of 6 hours, 17 minutes and 3 hours, 8 minutes for 4 and 8 processors, respectively. The Altix 350 cluster using the TCP/IP-based MPI library still needed 9:44h and 6:28h, respectively. Even Beast with this communication device did not near this goal: 10:50h and 5:50h, respectively, were not close to the estimated optimum. Message passing via shared memory, however, showed a considerably improved result. With Nemesis, the measured 3:55h was much closer to the optimum of 3:08h for 8 processors.

For the customer, the core time could be of higher interest as this is the measurement of supercomputer usage when considering costs. The diagram in Figure 2 shows the time a single processor took and the summed up core time of all processors. Because of the lack of interprocess communication, the sequential run required the least core time. The next best result shows Beast with 8 processors using TCP/IP followed by Beast with 8 processors and the Nemesis MPI device.

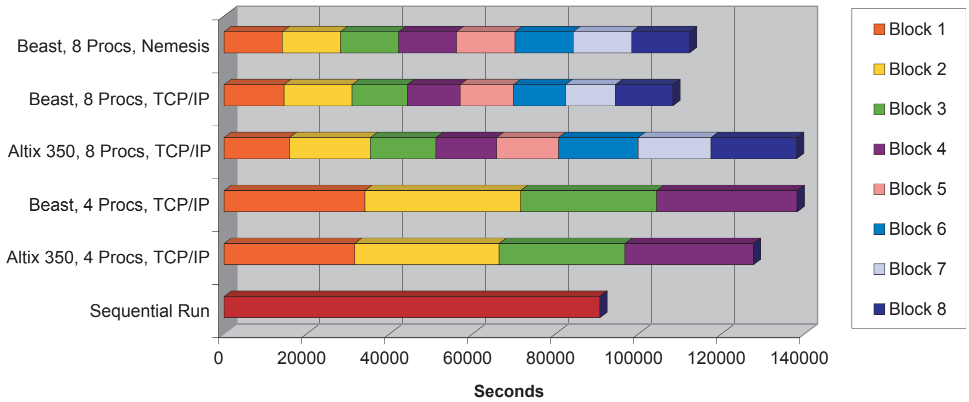

Figure 2. Core Time for different optimised simulation runs.

Investigating the used time for each processor core also facilitates the analysis of balancing behaviour of the parallelised flow solver. Figure 3 shows that Message Passing over TCP/IP yields balancing problems introduced by network latencies. Shared memory communication, instead, does not show such delays. As demonstrated by the wall clock time, Nemesis does not have this further overhead. Its maximum single core time is near to the wall clock time measured. This differs from the TCP/IP versions. Therefore, we found that the best setup considering the trade-offs of costs-speed ratio for fire simulations based on FDS was the shared-memory supercomputer Beast using MPICH2 configured to communicated with the Nemesis device.

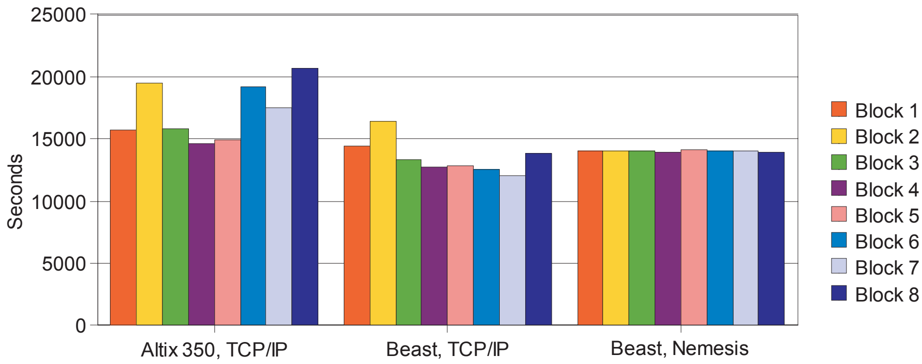

Figure 3. Chart depicting Core Time balancing for 8 blocks.

3.4 Fire in a Storage Simulation

For the storage fire simulation, different numbers of blocks with varying grid resolutions were used for the multi-block domain decomposition in order to reduce the computational cost and to improve the accuracy of simulation results. Eventually, we selected a flow field sub-division using 39 blocks. The resolution of the 25 inner blocks, consisting of rack geometry details important for the simulation, is considerably higher than of the 14 surrounding blocks. As FDS only supports rectilinear grids, this block arrangement leads to non-matching cells between the boundary of coarser and finer resolved grids. This can yield numerical artefacts, which could be depicted later on by post-processing approaches. The inner blocks consist of 2,621,512 cells and 2,647,406 vertices, whereas the surrounding blocks only show a summed up resolution of 243,216 cells and 280,539 vertices. In total, 2,684,499 vertices were used for the storage fire simulation. After discarding coincident grid points, 2,494,372 vertices remained. The

determination of this domain decomposition required some iterative steps as the cells of the inner blocks had to comprise the geometry details used for the simulation.
The simulation of 420 seconds, one snapshot per second, with seven scalar fields and one vector field produced an output dataset of approximate 56 GB. Only one sprinkler device below the ceiling was included in this first simulation setup. The wall clock time for this simulation run on Beast with Nemesis was 2 days, 10 hours, 14 minutes, and 3 seconds. For debugging purposes, we also used a solver version compiled with debug information. With that version, we had to wait over 13 days for the simulation results. Especially for an in-depth analysis of core and wall clock time, FDS produces very informative log files.

4 Interactive Post-Processing

4.1 Rack Rendering

In order to visualise a more sophisticated interior of the storage, we modelled the storage rack with Maya by Autodesk (Derakhshani, 2006) instead of using the simplified geometry defined in the input file (cf. Fig. 3, left). This model is based on blueprints and photos taken during a visit in one of the storage buildings of the customer (cf. Fig. 3, right). Later on, all single components, like the vertical and horizontal bars, the cross bars, the shelves, the catwalks, and the boxes were exported to a polygonal file format which could be used directly by our VR application.

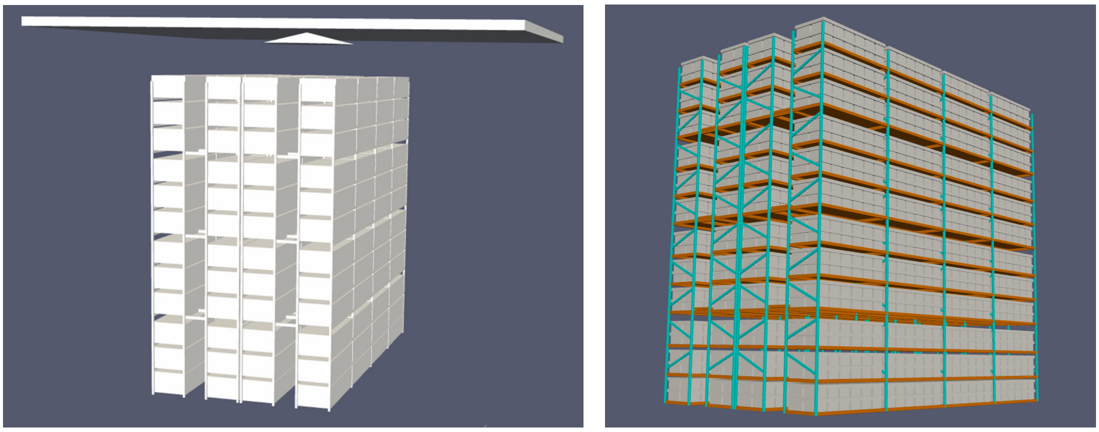

Figure 3. Geometry definition by the input file for the FDS simulation (left) and the high-resolution model designed with Maya (right).

4.2 Grid-based Output

FDS, as released, provides relatively few numerical outputs. Most of the produced result data is stored in a proprietary format which can directly be used by Smokeview. The input file already specifies which kind of post-processing algorithms are supposed to be carried out on simulation results. Thus, data available for each frame consists of, for instance, pre-computed isosurfaces, boundary values, cutplanes, particles trajectories, and more. This does not allow interactive exploration of fire simulations by changing post-processing parameters like the position of the cutplane or the isovalue which specifies an isosurface. The simulation grid with all computed quantities for a more interactive post-processing approach is not stored by default.
However, it is possible to enable the creation of Plot3D output files which is a rather old format defined by NASA (Buning and Steger, 1985). Furthermore, this format is fairly limited. It does not contain additional Meta information about the simulation and can only store five quantity component fields. These cannot be named so an external application does not have the ability to identify what is stored in the result files. By default, Plot3D interprets these quantities as

density, momentum vector components, and stagnation energy. In contrast, if not differently specified, FDS stores temperature, velocity vector components, and heat release rate per unit area (HRRPUV) in Plot3D solution files. Smokeview can use those files for a more grid-based analysis while the grid definition is taken from the input file. For external use, Plot3D grid files can be created as well.

Nevertheless, if one wishes to analyse different quantities than those stored in those result files, the time-consuming simulation has to be restarted. Therefore, it is highly recommended to store all simulated values in a data format which offers some more meta-information like scalar and vector field names so that all simulation results are available later on for an interactive post-processing. This was the reason to replace FDS's Plot3D export filter by a VTK format writer. VTK is the native format of the Visualization Toolkit (Schroeder et al, 2006), which is widely used, and can be converted easily in other currently common CFD data formats. Moreover, VTK is the basic post-processing toolkit used in the VR framework VRFlowVis described in Section 5. While this approach demands a higher storage memory requirement, it offers the ability to work with the produced data in a vastly interactive environment.

4.3 Fire Analysis

In general, the most import quantity for the fire analysis is heat release rate (HRR). In order to emphasise the development of the fire over time, we computed opaque isosurfaces (orange) with an iso-value of 1000 kW and semi-transparent isosurfaces (yellow) with an iso-value of 100 kW which yielded a good insight in the fire distribution. A further important aspect of the application was the activation of the sprinkler system. The activation start point was set to 74 degrees Celsius. As soon as this threshold is exceeded, the sprinkler goes off. Once more, this can be visualised with isosurfaces (blue) but now using the temperature scalar field with an iso-value of 74 degrees. A rendering snapshot is depicted in Figure 4:

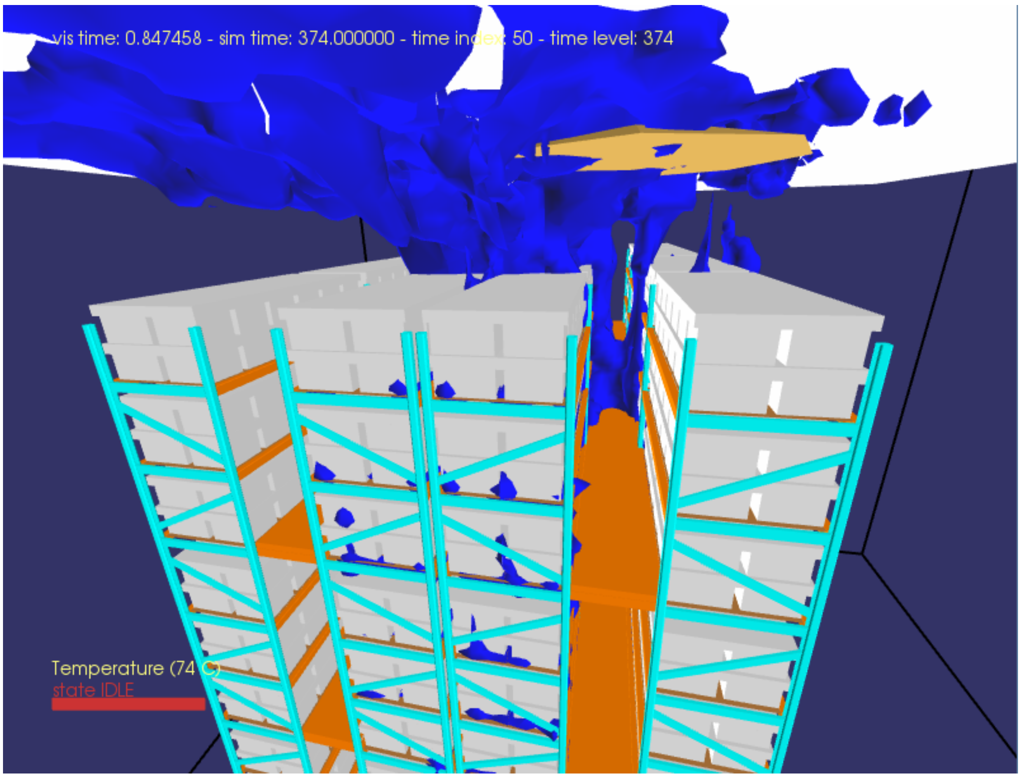

Figure 4. Sprinkler devices (one depicted by the cone below the ceiling) are activated at a temperature threshold of 74 degrees Celsius (blue isosurface).

Additional to these primary visualisation objects, we added the possibility to compute cutplanes which can be placed interactively within the entire dataset, and streamlines which aim at the interactive investigation of the velocity field behaviour.

5 VR Framework

5.1 Visualisation Environments

Multiple immersive VR venues are available at our facility. A 6-sided CAVE-like display system, called the Total Immersive Space (TIS), is driven by an HP SVA (Scalable Visualization Array) V.1.1.1 architecture. This cluster consists of 6 nodes with Fast Ethernet and Infiniband interconnects. Each node is equipped with two Opteron 246 processors (2 GHz) and 2 Gbyte main memory. NVidia Quadro FX 4500 graphics cards are installed that support OpenGL 2 and enhanced shader technologies.
The other VR venues, a 175 seat theatre with a 40 foot curved screen, a smaller collaboration room with a three channel front-projection screen, and a conference room with a 2-channel back-projection display, are all driven by 3 SGI Prism systems with 14 graphics pipes in total. These Itanium-2 based visualisation computers have up to 16 processors and 16 GByte shared main memory. To compose a computer graphics image on these displays, multi-pipe rendering with edge blending is used.

5.2 VRFlowVis

The dataset size of around 56 GByte is far too much to be processed on a common visualisation system used for virtual environments. To enable interactive exploration of CFD datasets, we integrated Viracocha (Gerndt et al, 2004) into our framework. It relieves the visualisation frontend by moving heavy I/O load and time-consuming post-processing tasks to dedicated supercomputer systems which generally offers multi processors for parallelism and huge main memory to handle large-scale datasets. In our case, we used Beast as backend as well as for the original simulation. This is based on the assumption that a system which was able to compute a simulation is almost always appropriate for the post-processing of the results as well.
The corresponding visualisation library to Viracocha is VistaFlowLib (Schirski et al, 2003) which is a toolkit to manage time-dependent visualisation objects. The post-processing results are transmitted from Viracocha and arranged in discrete time data arrays. A continuous visualisation time loop determines which bin of this container needs to be rendered. The core component of this time-dependent visualisation object selection is a time mapper. This time mapper also allows cutting out a certain time range of the simulation which should be considered for post-processing and visualisation. In our case, we detected that the first seconds do not change much and considering every fifth second is sufficient enough for an accurate analysis. All this could be specified by some few parameters of the time mapper so that finally only 60 time steps were used for efficient post-processing and rendering. This number is small enough to assign one processor at the backend (Beast) to one time-step of the simulation dataset so that all time steps can be processed at the same time in parallel.
Including VTK as the basic toolkit for post-processing and data structures, a powerful set of libraries was collocated. The processing, serialisation, transmission, and rendering of VTK objects are already encapsulated by Viracocha and VistaFlowVis. To assure the correct operation of our VR framework in multi-pipe environments as well as in cluster-based systems, we incorporate VR Juggler (Cruz-Neira et al, 2002), a widely used basic toolkit for virtual environment applications, into the design. The combination of data distribution in clustered environments by VR Juggler and VistaFlowLib (Kuhlen et al, 2005) worked seamlessly. The challenge, however, was the dynamic rendering of visualisation objects as display lists in multi-pipe environments as multi-pipe is not supported by VistaFlowLib yet.

To combine all the benefits of the basic rendering and visualisation libraries and to offer a straightforward interface for domain specific rapid application development, we designed a further software layer called VRFlowVis. It allows the dynamic handling of so-called visualisation components consisting of a data source, the data, and data sinks (optimised data renderers). As a source, the results of the backend can be used, but loading of pre-computed visualisation data or disposing application defined VTK pipeline segments is also allowed. This is utilised in the fire simulation visualisation. Post-processing results come from the backend, the simulated storage rack components are loaded directly from file server, and additional data like position and functional shape of sprinkler are analysed using the simulation input file to generate some more VTK objects.

An advantage of VRFlowVis's visualisation components is that each can be selected separately so that a user can perform component specific tasks. For instance, visibility toggling is possible to get a better insight into the scene whenever it becomes cluttered or important aspects are occluded. For the analysis of the storage fire, it was very helpful the hide individual boxes on the shelves to analyse the propagation of the fire through the rank (cf. Figure 5).

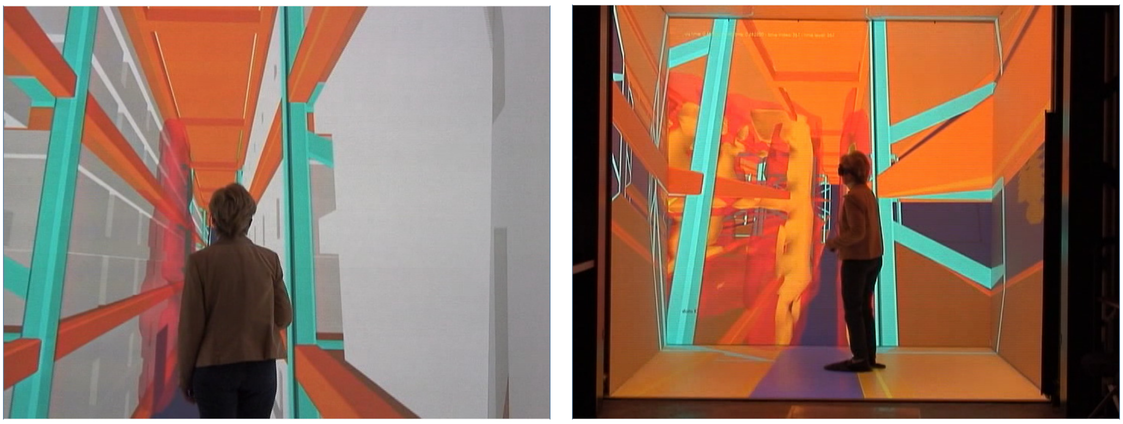

Figure 5. Immersive fire exploration (Heat Release Rate 100 kW and 1000 kW) in a CAVE-like virtual environment (left: with boxes, right: hidden boxes).

5.3 Interactivity and GUIs

In order to navigate through the scene and to offer basic functionalities to manipulate selected visualisation components in immersive environments, the wireless wand device, which is a part of the installed Intersense tracking system, can be used. Because of the complexity of the application, further settings are only available by the keyboard control. Unfortunately, this is less efficient in virtual environments. For this purpose, it is possible to employ a CORBA-based graphical user-interface (GUI) on a tablet-PC which can be carried even in wireless environments. As VR-Juggler allows the scalability of the application from laptop to CAVE-like systems, the application offers the choice to apply best fitting interaction devices. Thus, the keyboard approach can be used whenever the user works at a desktop while also allowing tablet-PCs come into play when working in wireless environments.

6 Conclusion and Future Work

In this paper, we presented a case study on how to manage the entire workflow chain from domain decomposition to interactive post-processing of a fire simulation in a storage warehouse. We showed that paying attention to the compilation of a solver can reduce customer's expense and speed-up the computation considerably, but the main challenge was to bring the huge dataset into interactive virtual environments. The solution was a distributed

framework, combined with some powerful libraries in the field of post-processing, visualisation, and Virtual Reality.
In this first implementation, we only applied one set of parameters for the fire simulation. Also, just one sprinkler device was involved. In the future, we plan to vary the input configuration for comparison purposes. Interactive comparison of different simulation results in an immersive environment is not trivial, but we believe that the visualisation component approach can help to integrate first approaches quickly. In this context, it is also important to compare simulation results with experimental measurements. Beside videos of the real fire test, a recorded series of a grid of measurement points was made available to us. But at present, it is much too early to conclude that our simulation can replace real fire tests. For this purpose, further simulations and fire tests must be performed which can then be compared utilising our framework.

7 Acknowledgements

The authors wish to thank the Virtual Reality Group, RWTH Aachen University, Germany who kindly made available the VistaFlowLib and Viracocha toolkits for evaluation purposes.

8 References

Buning, P. G. and Steger, J. L. (1985): Graphics and Flow Visualization in Computational Fluid Dynamics, In *Proceedings, 7th Computational Fluid Dynamics Conference, AIAA-1985-1507*, Cincinnati, OH, USA.

Cruz-Neira, C. and Bierbaum, A. and Hartling, P. and Just, C. and Meinert, K. (2002): VR Juggler - An Open Source Platform for Virtual Reality Applications. In *Proceedings, 40th AIAA Aerospace Sciences Meeting and Exhibit 2002*, Reno, NV, USA.

Derakhshani, D. (2006): *Introducing Maya 8 – 3D for Beginners*. Wiley Publishing.

Forney, G. P. (2007): User's Guide for Smokeview Version 5 - A Tool for Visualizing Fire Dynamics Simulation Data. NIST Special Publication 1017-1.

Gerndt, A. and Hentschel, B. and Wolter, M. and Kuhlen, T. and Bischof, C. (2004): VIRACOCHA: An Efficient Parallelization Framework for Large-Scale CFD Post-Processing in Virtual Environments. In *Proceedings, Supercomputing 2004*, Pittsburgh, PA.

Gropp, W. and Lusk, E. and Thakur, R. (1999): *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press.

Grosso, R. and Schulz, M. and Kraheberger, J. and Ertl, T. (1996): Flow Visualization for Multiblock Multigrid Simulations", In *Proceedings, Eurographics Workshop on Virtual Environments and Scientific Visualization*, Monte Carlo, Monaco, Springer, 296-307.

Kuhlen, T. and Gerndt, A. and Assenmacher, I. and Hentschel, B. and Schirski, M. and Wolter, M. and Bischof, C. (2005): Analysis of Flow Phenomena in Virtual Environments – Benefits, Challenges and Solutions. In *Proceedings, 11. International Conference on Human Computer Interaction, HCII 2005*, Las Vegas, NV, USA.

McGrattan, K. and Hostikka, S. and Floyd, J. (2007): Fire Dynamics Simulation (Version 5) – Users's Guide. NIST Special Publication 1019-5.

Rein, G. and Bar-Ilan, A. and Fernandez-Pello, A. and Alvares, N. (2006): A Comparison of Three Models for the Simulation of Accidental Fires. Journal of Fire Protection Engineering, Vol. 16, No. 3, 183-209.

Schirski, M. and Gerndt, A. and van Reimersdahl, T. and Kuhlen, T. and Adomeit, P. and Lang, O. and Pischinger, S. and Bischof, C. (2003): ViSTA FlowLib - A Framework for Interactive Visualization and Exploration of Unsteady Flows in Virtual Environments. In *Proceedings, 7th International Immersive Projection Technologies Workshop, and 9th Eurographics Workshop on Virtual Environments*, ACM Siggraph, Zurich, Switzerland, 77-85.

Schroeder, W. and Martin, K. and Lorensen, B. (2006): *The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics*. 4th edition, Kitware Inc.