HPC Predictions of Primary Atomization with SPH: Challenges and Lessons Learned

Samuel Braun, Lars Wieth, Thilo Dauch, Marc Keller, Geoffroy Chaussonnet,
Corina Höfler, Rainer Koch, Hans-Jörg Bauer
Institut für Thermische Strömungsmaschinen
Karlsruhe Institute of Technology
Karlsruhe, Germany
samuel.braun@kit.edu

***Abstract*—Up to now, numerical predictions of fuel preparation in the context of aircraft engines were not feasible due to the enormous computational costs. Thanks to the steadily improving availability of High Performance Computing (HPC) resources, such simulations have come into reach. In this paper we present the transition to such large scale simulations and the associated implications. An exemplary 3D simulation of a planar prefilming airblast atomizer, which is modeled by 1.2 billion particles, helps to illustrate the workflow, necessary code modifications and possible ways to assess the simulation data. Several aspects, which are prerequisites for successfully conducting the simulation and a beneficial usage of the results, are discussed. Most importantly, a massive improvement of the serial code performance has been achieved by changing the data structure and by ensuring a cache efficient order of the particle interactions. A reduction of the memory requirement during the pre- and post-processing steps has been realized using disassembled datasets. This allows the handling and analysis of large computational domains on standard desktop computers. By applying the α-shape algorithm, a descriptive and memory-saving visualization of the liquid surface is possible.**

I. Introduction

Proper atomization of the liquid fuel in aircraft engines is the key issue for controlling emissions. At the same time, any numerical prediction of the spray formation process was prohibitive due to the enormous computational costs which are associated to air assisted atomization. The spatial and temporal scales to be resolved cover at least 4 orders of magnitude. The lower bound of the length scales is defined by the smallest droplets with about $1\,\mu\mathrm{m}$ in diameter, the upper bound is defined by the geometry of the atomizer, which is in the order of $1\,\mathrm{cm}$. Recently, first attempts have been made to numerically predict an atomizer configuration which is representative for aircraft applications [12]. These simulations are based on the Finite Volume approach, the multi-phase flow is modeled by the Volume of Fluid (VoF) method. The simulation did reflect many experimentally observed features, however, the VoF method inherits some disadvantages. Most importantly, the predicted droplet sizes strongly depend on the choice of the volume fraction, which is used to distinguish between gaseous and liquid phase. Furthermore, despite a moderate number of cells of approximately 80 million, the computational runtime was rather long.

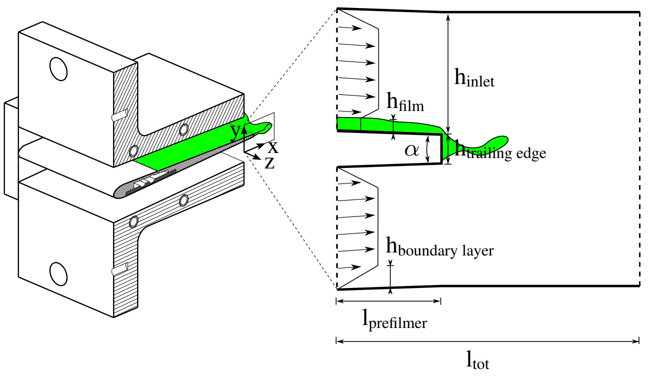


Fig. 1: Planar prefilming airblast atomizer, experimental setup (left) and numerical abstraction.

The SPH method shows a great potential to a more efficient use of the computer hardware. Applying a weakly compressible, explicit formulation leads to very low memory requirements and an excellent computing to communication ratio, when it comes to parallelization. Concerning multi-phase flows, the Lagrangian description avoids the complex interface tracking or reconstruction, which is necessary in grid based Eulerian methods. In the present paper we describe the application of SPH to problem sizes of more than 1 billion particles. Compared to today's largest Molecular Dynamics or CFD simulations, this represents only a medium sized problem. However, in order to achieve reasonable computing runtimes, a highly efficient use of HPC resources is indispensable. Furthermore, the pre- and post-processing tools must handle big data sizes, which in general do not fit on standard desktop computers.

The structure of the paper is as follows. In section II we introduce the setup of the computational domain, which is used to perform the prediction of primary atomization as well as scalability tests. In section III we briefly summarize our considerations regarding the optimization of the serial code performance. Especially the trade-off between vectorization and cache optimization is addressed. Furthermore, a detailed examination of the strong scalability of our code is discussed. In section IV an overview is given on how large data files can

TABLE I: Fluid properties and geometrical dimensions

Property	**air**	**liquid**
$\rho\,[\mathrm{kg\,m^{-3}}]$	1.0	770.0
$\mu\,[\mathrm{Pa\,s}]$	1.8×10^{-5}	1.56×10^{-3}
velocity@inlet $[\mathrm{m\,s^{-1}}]$	$0.0 - 50.0$	0.617
$\sigma_{\text{air–liquid}}\,[\mathrm{N\,m^{-1}}]$	0.0275	
contact angle$_{\text{wall–liquid}}$ $[^\circ]$	60.0	
h$_{\text{trailing edge}}$ $[\mathrm{\mu m}]$	230.0	
h$_{\text{inlet}}$ $[\mathrm{mm}]$	3.0	
h$_{\text{film}}$ $[\mathrm{\mu m}]$	80.0	
h$_{\text{boundary layer}}$ $[\mathrm{mm}]$	0.5	
l$_{\text{tot}}$ $[\mathrm{mm}]$	6.0	
l$_{\text{trailing edge}}$ $[\mathrm{mm}]$	2.0	
depth$_{\text{lateral}}$ $[\mathrm{mm}]$	4.0	
$\alpha\,[^\circ]$	4.29	
$dx\,[\mathrm{\mu m}]$	5.0	

be handled and assessed even on standard desktop hardware. A very important, yet overlooked subject is the visualization of the particle data. We present our strategy for visualizing liquid surfaces. Furthermore, the assessment of the spray properties is described. In section V we present the 3D simulation of an experimentally investigated airblast atomizer. The modeling approaches, the required computational resources as well as the physical findings are illustrated and discussed. In section VI the results are summarized.

II. Numerical Setup

The numerical setup for the scalability tests and for the spray prediction is derived from an experimentally investigated generic planar atomizer, which is a two-dimensional abstraction of an annular airblast atomizer [5]. Figure 1 and Table I give an overview of the geometric features of the experimental setup, the fluid properties and the numerical model. The experimental setup consists of an airfoil shaped prefilmer, which is exposed to an air stream. A liquid film is applied to the upper side of the prefilmer surface. High aerodynamic forces pull the liquid film to the trailing edge, where the liquid is accumulated and forms flapping ligaments. Finally it detaches from the prefilmer lip. The detachment and disintegration of the ligaments is called primary atomization.

The domain of the numerical predictions covers the region in the vicinity of the trailing edge. At the inlet, the air velocity is prescribed by a piecewise linear profile. The liquid phase enters the domain with a constant velocity. On top and on bottom the computational domain is confined by static walls. In two dimensions, the computational domain consists of roughly 1.5×10^6 particles with an inter-particle spacing of $dx = 5\,\mathrm{\mu m}$. During preliminary parametric studies, this spatial resolution has been found to be the coarsest acceptable for the given flow configuration. The 2D domain was used for the performance tests and for generating an initial solution for the 3D simulation. Here, an established 2D flow state is extruded in spanwise direction, with the lateral faces being confined by periodic boundary conditions. The lateral extent of $4\,\mathrm{mm}$ yields 1.2×10^9 particles.

Concerning the physical models, we apply the density equation as proposed by Hu and Adams [9], in order to handle the interfacial discontinuities. Viscosity is modeled by the formulation of Szewc [13]. The interaction between the liquid phase and the prefilmer lip plays an important role. Therefore, wall wetting effects have are taken into account using the extended surface tension model proposed by Wieth et al. [14]. Walls are represented by fixed wall particles.

III. Computational Performance

A. Serial Optimization

Our SPH code framework as presented previously [2] showed a decent scalability (c.f. Fig. 3, gray line). Therefore, in order to further accelerate simulations, the serial performance had to be improved. Within this paper we distinguish between hard optimizations and soft optimizations. Hard optimization denote a real code improvement like e.g. vectorization or more cache friendly algorithms, whereas soft optimization denotes the use of e.g. a less expensive kernel, smaller smoothing lengths or relaxed CFL conditions. In the following, both optimization types are discussed.

In order to identify the problematic passages of the code, different instrumentation and sampling tools like e.g. gprof [6] have been applied. Furthermore, the code has been inspected in order to find unnecessary or redundant calculations. Those simple optimizations, as e.g. described by Hager and Wellein [7], already resulted in measurable performance gains. However, in order to massively improve the serial performance, at a first glance vectorization seemed to be the solution of choice. Modern processors usually have special registers, which allow to execute SIMD instructions (Single Instruction Multiple Data). Depending on the data type, 4 or more identical simple operations can be performed on multiple data objects within one processor cycle. Using the SIMD instruction sets is generally called vectorization. However, vectorizing particle-interaction calculations is not straightforward, since the particles are scattered across the three dimensional space (even if they are sorted), whereas memory is one dimensional. In order be able to perform vectorized calculations, the data has to be packed, e.g. in groups of 4, as described by Alonso [1]. However, this data shuffling (gather/scatter operations) generates a large overhead and the symmetry of the interactions can not be exploited. For Molecular Dynamics simulations, Páll and Hess [10] proposed a remedy by creating spatial particle clusters. However, the overhead introduced by their method is only compensated, if the interactions are based on very simple operations. This is not the case for SPH. Our implementations of vectorized interaction loops were faster than the non-vectorized ones, however, the overhead due to data preparation compensated this effort in terms of a global view.

As vectorization was not suitable to substantially improve the serial performance, we examined the code concerning its cache efficiency. Cache means different levels of buffer memory, which have higher access bandwidths and a lower latencies than the main memory. However, the closer these cache levels are located to the processing units, the smaller they are. The usage of cache is beneficial, if data is to be reused. This is the case for particle interactions, as every particle does interact with a certain number of other particles and, therefore, has to be reread from memory during the interaction calculations. The two most important modifications for maximizing cache efficiency are the change of the data layout and a spatial sorting of the particles. In the original code, a particle has been stored in a derived data structure, containing all particle attributes. The data of all particles has been stored in a list of particle structures. This type of data layout is called array of structures (AoS). When it comes to the calculation of e.g. the density, only the particle location and its mass are of interest. However, the cache-line, i.e. the data block containing the needed information, which is loaded to a faster cache level or to the processor registers, is flooded with unnecessary supplementary data, like e.g. velocities, pressures, IDs et cetera. This causes the cache to be replaced and reloaded very frequently. A remedy consists of switching to a so called structure of arrays (SoA) data layout. Here, the particle attributes of all particles are stored in separate arrays. When performing calculations on the data, only the required data of the particles under concern is loaded. The caches have to be refreshed much less frequently, more data fits into the fast, low latency memory buffers. The switch from AoS to SoA required every single line of source code to be revised.

A further important step for improving the cache efficiency is to increase the spatial locality of data. This is achieved by sorting the particles in such a way, that particles assigned to the same cell of the list search are positioned adjacent in memory, as described e.g. by Alonso [1]. However, instead of applying an expensive sorting algorithm, we modified the particle deletion routine which is carried out at the end of each time step. This deletion step is required anyway, as particles might have crossed an outlet boundary or they might have left the computational sub-domain. The deletion consists of copying the particles to be kept into a new SoA-container and deleting the old container. Instead of accessing the particles sequentially, they are accessed according to their assignment to the search cubes. Consequently, the particles are sorted automatically with no extra cost. Additional particle sorting, e.g. the application of a Morton order (Lebesque curve) for the data traversal, did not show any improvements at all. Switching to the SoA layout and sorting the particles with respect to their position in the search grid increased the code performance by approximately a factor of 2 to 3.

As vectorization has been excluded from our optimization considerations, we could exploit the symmetry of the particle interactions. Identical to the findings described by Alonso [1], the gain in terms of speedup was approximately 50 %.

Our last consideration for increasing the cache efficiency

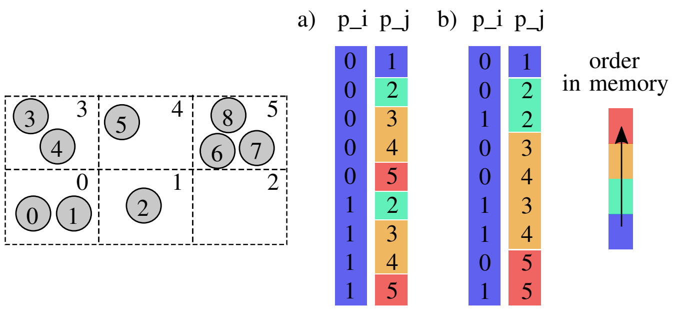


Fig. 2: 9 spatially sorted particles (gray circles) are assigned to 6 cells of the search grid (left). The corresponding Verlet lists for the particle interaction pairs are depicted on the right. Comparison of a) particle-centered and b) cube-centered creation of the lists. Particles within a colored block are adjacent in memory.

was to improve the temporal locality of data. This means, that operations on the same data objects should be temporarily close to each other. This increases the probability, that the cache line, which holds the required data, is still persistent in the fastest cache. We addressed the demand for temporal locality by modifying the creation of the Verlet lists.

In the original code version, the creation of the Verlet lists was particle-centered. This is illustrated in Fig. 2a, where the interactions of particle i and particle j are stored within the integer arrays `p_i` and `p_j`. The colors of the blocks denote the index of the containing search cubes. Because the particles are sorted spatially, the color also indicates the order of the particle data in memory. Particle-centered creation of the Verlet list means, that first all interaction partners of particle 0 are determined, before proceeding to particle 1 as regarded center particle. If during the interaction calculations the center particle `p_i` changes from particle 0 to particle 1, the memory controller has to jump (eventually far) backwards in order to fetch the `p_j` particle data. This renders the currently loaded cache obsolete and the entire cache line has to be replaced.

In Fig. 2b the result of a cube-centerd creation of the Verlet list is depicted. A cube-centered creation means, that all interactions between particles of adjacent search cubes (e.g. cube 0 and cube 1) are determined, before proceeding to the next search cube (e.g. 0 and cube 3). When performing the interaction calculations, `p_i` jumps forward and backward in memory. However, the maximum width of the jumps corresponds to the number of particles inside a search cube. It is very likely, that the corresponding data is contained in a fast cache level. Concerning the `p_j` data, there are no far reaching backward jumps anymore. The maximum backward jump is also defined by the number of particles within one search cube. Due to the fact, that the cache is loaded in blocks in forward direction with a certain size, chances are high that there are only very few cache misses, once the first particle attribute of a new memory block has been loaded. Depending on the problem size, the performance gain of using this temporal locality aware approach is in the order of $50\,\%$.

Concerning the soft optimizations, switching from a quintic kernel with a smoothing length of $h = 1.0 \cdot dx$ ($dr = 3.0 \cdot dx$) to the Wendland kernel with $h = 1.3 \cdot dx$ ($dr = 2.6 \cdot dx$) resulted in a massive improvement of the simulation speed. This is due to the fact, that the Wendland kernel function can be realized without conditional branching and, therefore, is computationally less expensive. Furthermore, the reduced kernel cut-off radius naturally reduces the computational costs. Finally, as h is bigger, larger time steps are possible, as long as the flow is limited by the numerical speed of sound. A further increase of the time step size has been realized by relaxing the factors of the CFL conditions.

The overall performance gain of the serial code optimizations is in the order of 5 with respect to the number of computed time steps per hour. With respect to the physical time, which can be computed per hour, the performance gain is above a factor of 10.

B. Parallel Performance

Strong scalability tests have been performed on the cluster ForHLR I[1], which is equipped with 2 Deca-Core Intel® Xeon® E5-2670 v2 processors per compute node. Each of the 512 thin nodes is connected via an InfiniBand 4X FDR interconnect. The operating system is RHEL Server release 6.7 with the 2.6.32 Linux kernel. The SPH code has been compiled with the GNU Compiler Collection 4.9.3 with auto-vectorization enabled, Profile Guided Optimization (PGO) and using fast math optimizations. The computational domain for the scalability tests consists of the 2D domain, described in section II. It has been decomposed into 1 to 1000 sub-domains, assuring an even distribution of particles. The termination criterion of the scalability tests was 1 hour wall clock time. This renders a scalability test over 3 orders of magnitude easily feasible, however, temporal changes of the computational load may falsify the results. The speedup has been calculated using the computed time steps within one hour. In order to assess the performance of the SPH code, comparative simulations with commonly used multi-phase solvers have been conducted, applying the same initial and boundary conditions. The number of cells of the Cartesian grid corresponds to the number of particles. The two tools to be compared to SPH were the VoF solvers of the CFD toolkits OpenFOAM® 2.3.0 (interFoam) and another commercially available CFD software. Turbulence modeling has been disabled.

It is to be emphasized that the following results might not be representative for the maximum achievable performance of the commercial software and OpenFOAM. Both codes have been used to the best of the author's knowledge. The multi-phase solver settings correspond to production run settings, which are typically used at our institute. Furthermore, small domain sizes of only 1.5×10^6 cells are usually not considered to be run on more than 2 nodes.

In Fig. 5 the speedup and the parallel efficiency are depicted for the original SPH implementation (gray line), the serially optimized version, the commercial software and OpenFOAM. The reference speed was defined by the performance of one compute node with 20 cores. The termination criterion is one hour wall clock time. OpenFOAM shows a decent scalability till 40 cores (2 nodes), where it especially profits from the fast intra-socket communication. The serial performance of the commercial software is questionable, however, it scales very well till 5 nodes. However, above 100 cores, saturation of the speedup is reached for both grid based tools. Furthermore, the efficiency shows a severe breakdown. A further increase of computational resources would not accelerate the simulation anymore. The parallel efficiency of the old SPH version does not show any degradation at all. The efficiency of the new SPH code starts to show a slight decrease for core numbers greater than 200. However, there is no sudden performance drop.

At a low number of processors, OpenFOAM seems to outperform the other codes. The old SPH version seems to perform better than the new version. Therefore, in order to assess the real world performance of the different codes, the actual reduction of the simulation time is considered. In Fig. 4 the simulated physical time, which can be computed with one hour wall clock time (left) or by spending one CPU-hour (right) is depicted over the number of cores. The simulations with the commercial solver used a fixed time step size of $2.5 \times 10^{-8}\,\mathrm{s}$, SPH and OpenFOAM used an adaptive time stepping with mean time increments of $1.6 \times 10^{-8}\,\mathrm{s}$ (SPH) and $4.3 \times 10^{-8}\,\mathrm{s}$ (OpenFOAM), respectively. Looking at the grid based methods and the old SPH code, the commercial software clearly outperforms the other codes within a wide range of cores. Due to its perfect scalability, the old SPH code was able to catch up with OpenFOAM at 100 cores and with the commercial code at 400 cores.

The serially optimized SPH code outperforms all other codes by far. At the best, OpenFOAM achieves $0.19\,\mathrm{ms}$ physical time per hour wall clock time and the commercial code achieves $0.63\,\mathrm{ms}$ per hour. SPH achieves $6.2\,\mathrm{ms}$ per hour using 400 cores. By using 1000 cores, the SPH code achieves $12.4\,\mathrm{ms}$ per hour. The simulation could even be accelerated further, as the speedup has not yet reached saturation. The physical time, which can be computed per CPU-hour, can be interpreted as a measure for the cost- or energy-effectiveness of the code. Even at 1000 cores, SPH shows a better cost-benefit ratio than the other codes at their respective optimal number of cores.

In order to check, whether the use of the wall clock time as termination criterion substantially affects the speedup results, further scalability tests have been run with the serially optimized SPH code. Using a fixed number of time steps (20 000) as termination criterion, a time dependent computational load will not affect the results. The speedup has been determined

[1]Forschungshochleistungsrechner ForHLR (Phase) I
http://www.top500.org/system/178424
http://www.bwhpc-c5.de/wiki/index.php/ForHLR_Phase_I_-_Hardware_and_Architecture

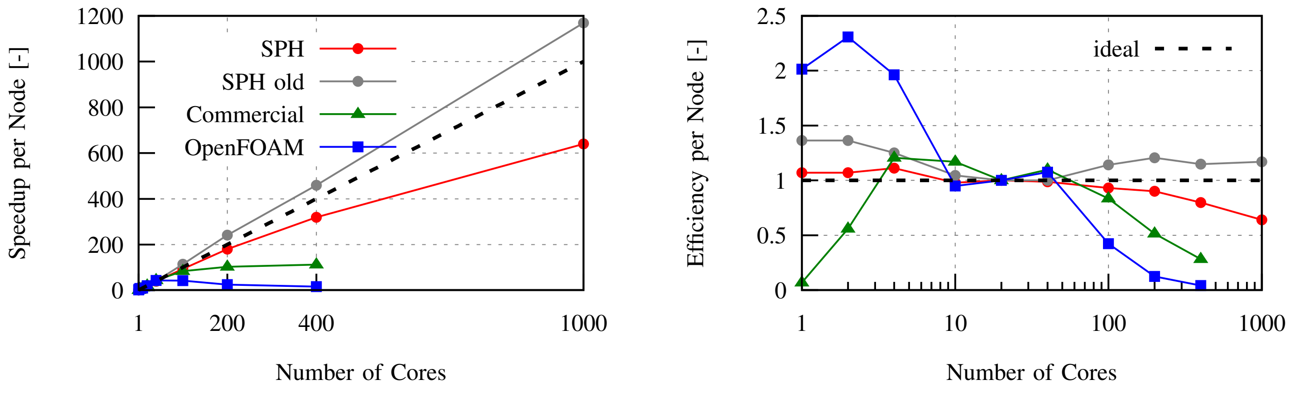


Fig. 3: Speedup and parallel efficiency for SPH, OpenFOAM and a commercial solver. The graphs are normalized by the performance of a single node with 20 cores.

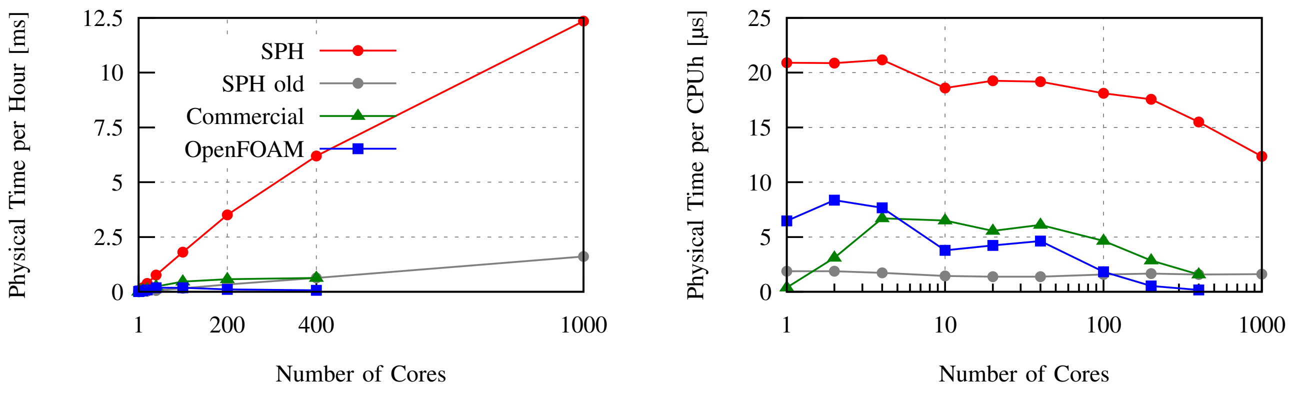


Fig. 4: Physical time per hour and per CPU-hour for SPH, OpenFOAM and a commercial solver.

evaluating the required runtimes. In Fig. 5 the resulting parallel efficiencies are compared. The red line and the gray line result from the wall clock time termination, the blue line and the green line result from the time step based termination. The wall clock time for 20 000 time steps varied between 12 h for 1 core and 97 s for 1000 cores. The reference performance is given by 1 node. The green line has been obtained starting with an established flow regime as initial condition. The other graphs are obtained with initially all particles at rest.

The effect of the fast intra-socket communication is clearly perceptible for the step based termination criterion. For the time based termination, this effect is reduced. This could be due to an increased computational load at the beginning of the simulation. As the established flow and the non-established flow (green and blue line) show a very similar behavior, this increased computational load must be attributed to a non-physically determined time dependent load. One explanation could be the required time for memory allocation, which mainly takes place at the beginning of the simulation. The SoA data layout of the new SPH implementation requires more arrays to be allocated, than the old AoS approach. Therefore, the old implementation shows a better efficiency at an intra-socket basis. Using more than one node, all three scalability tests with the new SPH implementation show an identical behavior. The efficiency increase for large numbers of cores of the old implementation is most likely due to cache effects. The smaller the sub-domains, the more data fits into the small but fast lower cache levels. This compensates the increasing effort for communication.

In summary it can be stated, that the new SPH implementation shows a massively improved performance. In contrast to the old implementation, the inter-processor communication cannot be hidden by the bad serial performance anymore. However, the code still shows a decent strong scalability over 3 orders of magnitude, with sub-domain sizes ranging from 1.5×10^6 down to 1500 particles.

IV. Pre- and Post-Processing

A. Data Handling

Usually, computational domains consisting of more than $n_{\text{particle}} = 1 \times 10^8$ particles cannot be handled using regular desktop computers, due to the limited main memory. However, it would be convenient to create the computational domain and to do the post-processing apart from HPC facilities. Therefore, we modified all our pre- and post-processing tools in such a way, that the particle attributes are loaded and processed

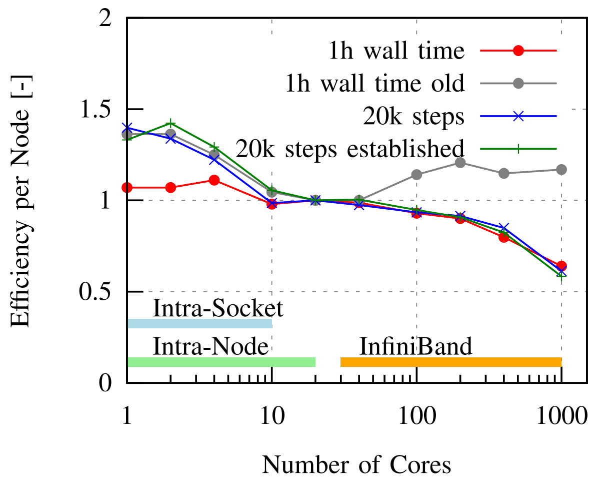
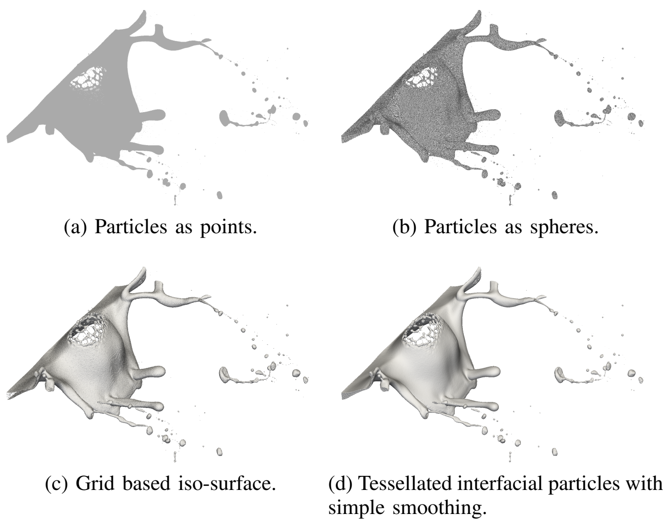


Fig. 5: Comparison of different scalability termination criteria. The graphs are normalized by the performance of a single node with 20 cores.



(a) Particles as points.
(b) Particles as spheres.
(c) Grid based iso-surface.
(d) Tessellated interfacial particles with simple smoothing.

Fig. 6: Different visualization types for a three dimensional liquid structure.

separately. Consequently, at most 3 float or double arrays (position) and one boolean array (bookkeeping, filtering) with the length of n_{particle} have to be kept in memory simultaneously. This approach is easily applicable using the binary data format H5Part [8]. A further advantage of H5Part is the fact, that the size of one file is not limited and that read and write accesses can be performed close to the hardware speed limits [3]. After the modification of the tools, the 3D simulation to be presented in the following (1.2 billion particles, 62 GB per time step) could be pre-processed (domain creation, domain decomposition) and post-processed (domain reconstruction, filtering, spray analysis) on a desktop computer with 24 GB RAM (XServer not running). However, during the post-processing of the data, it became obvious, that for future simulations with more than 10 000 cores, a per-node data output should be considered, instead of the current per-process output. This will reduce the number of necessary file accesses.

B. *Visualization*

In order to understand the breakup process, an appropriate visualization of the liquid phase is required. However, most post-processing tools rely on computational meshes or tessellated data. The direct depiction of particles generally compromises the visual sensation of depth or creates a wrong impression of surface texture. This can be seen in Figs. 6a and 6b, where a three dimensional liquid structure is visualized by particles. Triangulated surfaces improve the sensation of depth substantially. Furthermore, the data sizes to be handled are reduced dramatically, if only the surface mesh is visualized. The depiction of the liquid surface in Fig. 6c has been obtained by interpolating the particle data on a Cartesian grid and by extracting a certain iso-surface level. This approach is computationally very inexpensive, but it generates visible grid artifacts at the inclined surfaces.

By the α-shape algorithm [4], the liquid surface is directly tessellated. The utmost particles serve as supporting points of the tessellating triangles. Figure 6d has been obtained by applying this algorithm and additionally performing a slight surface smoothing. Despite the higher computational effort with respect to a interpolation based iso-surface reconstruction, the α-shape algorithm seems to be the most appropriate method for the visualization of phase interfaces.

In order to further facilitate the sensual comprehensibility and, therefore, increase the benefit of 3D simulation data, the upcoming virtual reality (VR) methods are a great opportunity. Currently, our simulation results can be read and displayed on mobile devices in combination with Google Cardboard. More advanced VR systems will be applied in the near future.

C. *Spray-Analysis*

When doing simulations of atomizing devices, the quantities of interest are mainly the spray characteristics, such as droplet size distributions, breakup frequencies and trajectories. These characteristics can be used for quantitative comparisons with experimental findings, but they also may serve as initial and boundary conditions for combustion simulations. In order to extract the spray characteristics, the droplets and detached ligaments have to be detected. Hence, we implemented a cluster detection tool, which is based on the Connected Component Labeling (CCL) technique proposed by Rosenfeld and Pfaltz [11]. The first step of droplet recognition consists of discarding the gaseous phase and the walls, which would not be possible using grid based methods. For the given test case, the amount of data can be reduced by approximately 99 %. In order to perform the CCL algorithm, a connectivity list for the remaining particles has to be build. As the computational domain is only sparsely filled, a tree based neighbor search algorithm is applied. By only taking into account a 10-connectivity, this can be done with a minimum computational effort. The CCL algorithm basically assigns the same ID to every element inside a connected cluster of particles. Connectivity is assumed, if the inter-particle distance is smaller than 1.5 times the mean particle spacing. As a result, every detected cluster of particles can be distinguished by its ID and it can be characterized by e.g. the center of gravity, the number of associated particles, a bounding box, a representative volume and velocity and a deformation index. By replacing all particles of a cluster by a single representative droplet, the data size is further reduced from several gigabytes to a couple of kilobytes. A subsequent statistical analysis is then easily feasible.

V. Simulation of a Generic Atomizer

A. Computational details

The 3D computational domain described in section II has been split into 2560 sub-domains. With the given number of 1.2 billion particles for the entire domain, this yields roughly 5×10^5 particles per sub-domain. The simulations have been performed on the compute cluster ForHLR I, which is described in section III-B. During the first 490 hours a quintic interpolation function with a smoothing length of 5 µm has been applied. Within 305 522 time steps 3.17 ms physical time could be simulated. Afterwards, a Wendland kernel with a smoothing length of 6.5 µm has been implemented and used, due to its lower computational costs. Within 960 hours, 790 160 time steps have been simulated. This corresponds to 11.43 ms of physical time. On average 43 000 time steps could be calculated within 2 days. Using the Wendland kernel, the computational effort for one particle iteration is 8.5×10^{-6} CPUs/step/particle.

Altogether, 1 095 682 time steps have been calculated within 60 days. This corresponds to 3.71×10^6 CPUh. 1113 time steps have been dumped to the file system. The data size of one time step is approximately 62 GB, the entire simulation data has a size of 69 TB.

B. Results

In Fig. 7 a snapshot of a breakup event is represented. Only a slice of the gaseous phase is depicted, where the colors correspond to the velocity magnitude. The image shows the instant in time, when a bubble shaped liquid structure starts to burst. This feature of the atomization process has also been experimentally observed. It is phenomenological related to the bag breakup regime of secondary atomization [5]. The present simulation is one of the first numerical simulations ever to correctly capture this type of atomization process. The accumulation of liquid at the trailing edge, the periodic flapping of the attached liquid and the breakup of these ligaments can be predicted with 2D simulations. However, only 3D simulations are able to capture those bubble shaped structures. The high temporal resolution of the simulation allows for a thorough investigation of the breakup process, which even phenomenologically is not yet understood in detail.

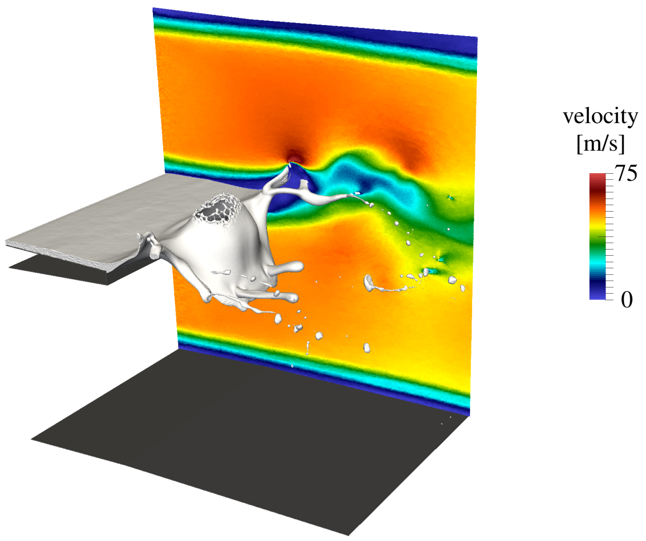


Fig. 7: 3D view of the computational domain. Only a slice of the gaseous phase is depicted, the coloring denotes the velocity magnitude. The upper wall is not depicted.

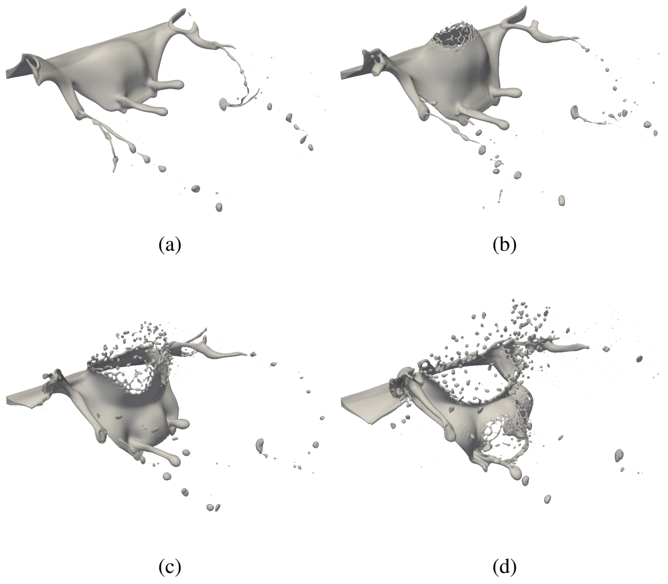


Fig. 8: Sequence of a bag breakup event.

In Fig. 8 a time series of such a breakup event is depicted. The time increment between two consecutive images is 74.5 µs, which corresponds to 5 times the data output increment. Beside the bag breakup, a Rayleigh breakup of elongated ligaments can be observed. These elongated ligaments either stem from the remaining rims of the bubble structures or are directly pulled out of the liquid film. In Fig. 9 experimentally obtained consecutive top view snapshots are compared to the simulation results. Both, bag breakup processes and Rayleigh breakups can be identified. At a first glance, the qualitative comparison reveals very similar breakup structures and length

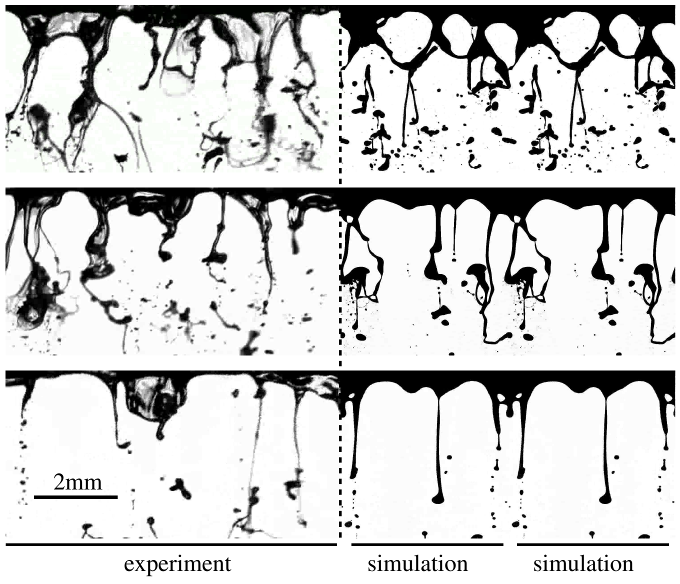


Fig. 9: Comparison of experiment (left half) and simulation. The displayed experimental section is $8\,\mathrm{mm} \times 4\,\mathrm{mm}$. The simulation data is duplicated in span-wise direction. Images of the experimental investigations: courtesy of S. Gepperth.

scales. At the same time, it is clearly visible that the very thin skin of the bursting bubbles (estimated thickness: $1\,\mu\mathrm{m}$) could not be predicted by the simulations, due to the restricted spatial resolution of approximately $5\,\mu\mathrm{m}$. Quantitative comparisons of the experiment and the simulation are currently ongoing. However, it is worthwhile to mention, that the comparison of the droplet spectra have to be limited to droplets larger than $14\,\mu\mathrm{m}$ in diameter, as this corresponds to the spatial resolution of the high speed camera used for image recording. Furthermore, statistically significant statements will not be conceivable, due to the limited amount of simulated breakup events.

VI. Conclusion

The successful prediction of primary atomization inherently requires massive computing power. SPH offers the great opportunity, to efficiently make use of the available resources. Depending on the regarded physical problem, SPH can outperform commonly used grid based methods by far. However, in order to achieve the highest possible computing effectiveness, a relatively detailed understanding of the computer hardware is necessary, which helps to estimate the implications of the implemented algorithms. In the present paper we conclude, that cache aware algorithms seem to outweigh the advantages of vectorization. The Lagrangian nature of SPH not only facilitates multi-phase handling, but also can be exploited for memory efficient pre- and post-processing. By separately handling the particle attributes, even very large simulation data sets can be conveniently processed on standard desktop computers. Concerning the visualization of multi-phase flows involving several length scales, the tessellation of the phase interfaces is advantageous with respect to a reasonable sensation of depth. The algorithm of choice seems to be the α-shape method. Further advantages of using tessellated surfaces are the massive data reduction and the possibility to reuse the mesh with sophisticated rendering software.

Primary atomization is now within the range of CFD simulations. We successfully predicted the bag breakup regime of an air assisted atomizer. 1.2 billion seems to be the lowest possible number of particles required for physically meaningful results of this 3D multi-scale phenomenon. Quantitative comparisons to experiments are currently ongoing. However, statistically significant quantitative comparisons will not be feasible due to the limited amount of predicted breakup events.

Acknowledgment

This work was performed on the computational resource ForHLR Phase I funded by the Ministry of Science, Research and the Arts Baden-Württemberg and DFG ("Deutsche Forschungsgemeinschaft") within the framework program bwHPC.

References

[1] José Manuel Domínguez Alonso. *DualSPHysics: Towards High Performance Computing using SPH technique*. PhD thesis, University of Vigo, September 2014.

[2] S. Braun, L. Wieth, R. Koch, and H.-J. Bauer. A Framework for Permeable Boundary Conditions in SPH: Inlet, Outlet, Periodicity. *10 th International SPHERIC Workshop*, 2015.

[3] Suren Byna, A. Uselton, D. Knaak Prabhat, and Y He. Trillion particles, 120,000 cores, and 350 TBs: Lessons learned from a hero I/O run on Hopper. In *Cray User Group meeting*, 2013.

[4] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *Information Theory, IEEE Transactions on*, 29(4):551–559, 1983.

[5] S. Gepperth, R. Koch, and H.-J. Bauer. Analysis and comparison of primary droplet characteristics in the near field of a prefilming airblast atomizer. In *ASME Turbo Expo 2013: Turbine Technical Conference and Exposition*. American Society of Mechanical Engineers, 2013.

[6] Susan L Graham, Peter B Kessler, and Marshall K Mckusick. Gprof: A call graph execution profiler. In *ACM Sigplan Notices*, volume 17, pages 120–126. ACM, 1982.

[7] Georg Hager and Gerhard Wellein. *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010.

[8] Mark Howison, Andreas Adelmann, E Bethel, Achim Gsell, Benedikt Oswald, et al. H5hut: A high-performance I/O library for particle-based simulations. In *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, pages 1–8. IEEE, 2010.

[9] X.Y. Hu and N.A. Adams. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, 213(2):844–861, 2006.

[10] Szilárd Páll and Berk Hess. A flexible algorithm for calculating pair interactions on SIMD architectures. *Computer Physics Communications*, 184(12):2641–2650, 2013.

[11] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966.

[12] B. Sauer, A. Sadiki, and J. Janicka. Numerical analysis of the primary breakup applying the embedded dns approach to a generic prefilming airblast atomizer. *The Journal of Computational Multiphase Flows*, 6(3):179–192, 2014.

[13] K. Szewc. Smoothed Particles Hydrodynamics – the implementations of the incompressibility. *Copernican Letters*, 1(0):141–154, 2010.

[14] L. Wieth, S. Braun, R. Koch, and H.-J. Bauer. Modeling of liquid-wall interaction using the meshless Smoothed Particle Hydrodynamics (SPH) method. *26th Annual Conference on Liquid Atomization and Spray Systems, Bremen, Germany*, 2014.