Parallel Computing: On the Road to Exascale
G.R. Joubert et al. (Eds.)
IOS Press, 2016

doi:10.3233/978-1-61499-621-7-767

High throughput simulations of two-phase flows on Blue Gene/Q

Panagiotis HADJIDOUKAS, Diego ROSSINELLI, Fabian WERMELINGER, Jonas SUKYS, Ursula RASTHOFER, Christian CONTI, Babak HEJAZIALHOSSEINI, and Petros KOUMOUTSAKOS [1]
Chair of Computational Science, Department of Mechanical and Process Engineering, ETH Zürich, Switzerland

Abstract. CUBISM-MPCF is a high throughput software for two-phase flow simulations that has demonstrated unprecedented performance in terms of floating point operations, memory traffic and storage. The software has been optimized to take advantage of the features of the IBM Blue Gene/Q (BGQ) platform to simulate cavitation collapse dynamics using up to 13 Trillion computational elements. The performance of the software has been shown to reach an unprecedented 14.4 PFLOP/s on 1.6 Million cores corresponding to 72% of the peak on the 20 PFLOP/s Sequoia supercomputer. It is important to note that, to the best of our knowledge, no flow simulations have ever been reported exceeding 1 Trillion elements and reaching more than 1 PFLOP/s or more than 15% of peak. In this work, we first extend CUBISM-MPCF with a more accurate numerical flux and then summarize and evaluate the most important software optimization techniques that allowed us to reach 72% of the theoretical peak performance on BGQ systems. Finally, we show recent simulation results from cloud cavitation comprising 50000 vapor bubbles.

Keywords. Petaflop computing, Flow simulations, Cavitation

Introduction

Modern supercomputers enable engineers to design effective solutions for the energy and the environment where fluid flows play a key role. Cloud cavitation collapse can induce pressure peaks up to two orders of magnitude larger than the ambient pressure [1]. It is detrimental to the lifetime of high pressure injection engines and ship propellers and instrumental to kidney lithotripsy and drug delivery [2]. The simulation of cavitation requires two-phase flow solvers capable of capturing interactions between deforming bubbles, formation of shocks and their impact on solid walls, over a multitude of spatiotemporal scales. Moreover, simulations of bubble cloud collapse are very demanding in terms of numbers of operations, system size and memory traffic.

CUBISM-MPCF [3] is a compressible, two-phase finite volume solver capable of simulating and study cavitation collapse of clouds composed of up to 50000 bubbles. It is designed for multi-core clusters and highly optimized on Blue Gene/Q (BGQ) platforms, where it has managed to achieve up to 72% of the peak performance, to date the

[1] Corresponding Author: petros@ethz.ch

best performance for flow simulations in supercomputer architectures. A set of software techniques take advantage of the underlying hardware capabilities and concern all levels in the software abstraction aiming at full exploitation of the inherent instruction/data-, thread- and cluster-level parallelism.

1. Models and numerical approach

Cavitation dynamics are mainly governed by the compressibility of the flow while viscous dissipation and capillary effects take place at orders of magnitude larger time scales. We therefore solve inviscid, compressible, two-phase flows by discretizing the corresponding Euler equations. The system of equations describing the evolution of density, momentum and total energy of the flow reads:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0, \\ \frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}^T + p\mathbb{I}) &= \mathbf{0}, \\ \frac{\partial E}{\partial t} + \nabla \cdot ((E+p)\mathbf{u}) &= 0. \end{aligned} \tag{1}$$

The evolution of the vapor and liquid phases is determined by another set of advection equations:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \tag{2}$$

where $\phi = (\Gamma, \Pi)$ with $\Gamma = 1/(\gamma - 1)$ and $\Pi = \gamma p_c/(\gamma - 1)$. The specific heat ratio γ and the correction pressure of the mixture p_c are coupled to the system of equations (1) through a stiffened equation of state of the form $\Gamma p + \Pi = E - 1/2\rho|\mathbf{u}|^2$. We discretize these equations using a finite volume method in space and evolving the cell averages in time with an explicit time discretization.

Finite Volume Method. The governing system (1) and (2) can be recast into the conservative form

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} + \frac{\partial \boldsymbol{H}}{\partial z} = \boldsymbol{R}, \tag{3}$$

where $\boldsymbol{Q} = (\rho, \rho\mathbf{u}, E, \phi)^T$ is the vector of conserved variables and $\boldsymbol{F}, \boldsymbol{G}, \boldsymbol{H}$ are vectors of flux functions

$$\boldsymbol{F} = \begin{pmatrix} \rho u_x \\ \rho u_x^2 + p \\ \rho u_y u_x \\ \rho u_z u_x \\ (E+p)u_x \\ \phi u_x \end{pmatrix}, \quad \boldsymbol{G} = \begin{pmatrix} \rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ \rho u_z u_y \\ (E+p)u_y \\ \phi u_y \end{pmatrix}, \quad \boldsymbol{H} = \begin{pmatrix} \rho u_z \\ \rho u_x u_z \\ \rho u_y u_z \\ \rho u_z^2 + p \\ (E+p)u_z \\ \phi u_z \end{pmatrix}, \quad \boldsymbol{R} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \phi(\nabla \cdot \mathbf{u}) \end{pmatrix}.$$

The last term in $\boldsymbol{R}$ is due to rewriting (2) in conservative form [4]. The method of lines is applied to obtain a semi-discrete representation of (3), where space continuous operators are approximated using the finite volume method. The time continuous system of ordinary differential equations

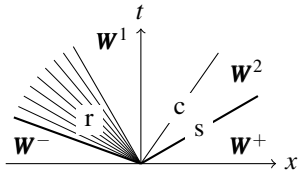
$$\frac{d\boldsymbol{V}(t)}{dt} = \mathscr{L}(\boldsymbol{V}(t)), \tag{4}$$

is then solved using a low-storage third-order Runge-Kutta scheme [5]. $\boldsymbol{V} \in \mathbb{R}^{7N}$ is a vector of conserved cell average values at each center of the N cells in the domain. The evaluation of the discrete operator $\mathscr{L}\colon \mathbb{R}^{7N} \rightarrow \mathbb{R}^{7N}$ requires the solution of a Riemann problem at each cell face in the computational domain.

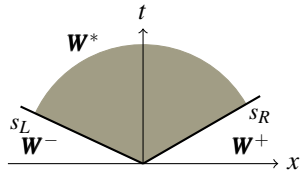
Three computational kernels are considered to perform one step. First, the admissible time step, Δt, is determined by adhering to the CFL condition. We call this kernel "DT" and execute it once per step. The second stage requires the evaluation of the right-hand side in (4). The explicit third order time integration requires the evaluation of $\mathscr{L}$ three times per step. The computational kernel for this task is called "RHS". Finally, updating the state of each of the solution variables in the domain is performed by calling the kernel "UP" once each step. RHS is bounded by the maximum performance of a BGQ node, while the DT and UP are memory bound [3]. In addition, RHS, DT and UP correspond approximately to 90%, 2% and 8% of the total simulation time, respectively (with I/O disabled).

Improved Numerical Flux. The RHS solves a Riemann problem on each cell interface in order to determine the numerical flux $\mathscr{F}^n(\boldsymbol{W}^-, \boldsymbol{W}^+)$ at discrete time t^n. Here $\boldsymbol{W} = (\rho, \mathbf{u}, p, \phi)^T$ is a vector of primitive variables. The two states $\boldsymbol{W}^-$ and $\boldsymbol{W}^+$ are the initial states for the Riemann problem. They are reconstructed from cell center values to the cell faces, using a third- or fifth-order weighted essentially non-oscillatory (WENO) scheme [6]. We use primitive variables for reconstruction to prevent numerical instabilities at interfaces [4]. The Riemann problem is then approximated at each cell face by assuming a single constant state $\boldsymbol{W}^*$ instead of the state $\boldsymbol{W}^1$, obtained from the exact wave structure, cf. Figure 1. This approximate Riemann solver is due to Harten, Lax and van Leer [7] and known as HLLE, where the wave speed estimates, s_L and s_R in Figure 1(b) are computed based on the work of Einfeldt [8].

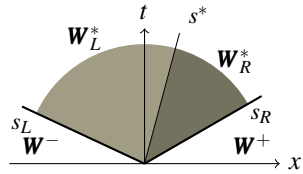
In the context of this work, the accuracy of the approximate Riemann problem has been improved by explicitly modeling the contact discontinuity, labeled by "c" in Figure 1(a), by implementing the HLLC numerical flux [9]. The HLLC approximation assumes two constant intermediate states, separated by a contact discontinuity, propagating with a speed s^*, cf. Figure 1(c). Separating the contact discontinuity by two states allows for sharper resolution of material interfaces, which is important for multicomponent flows. The single state in the HLLE approximation is diffusive over the whole wave structure and consequently produces larger amounts of numerical viscosity in flow regions with multiple components. Estimates for s_L and s_R are again obtained from Einfeldt [8], which results in comparable approximations for shock and expansion waves for both, HLLE and HLLC solvers.



(a) Wave patterns for a specific exact solution to the Riemann problem.

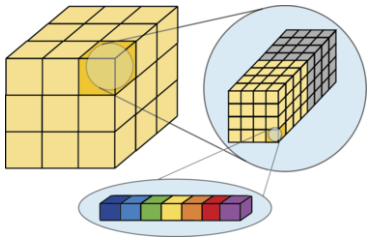


(b) HLLE approximation to the exact solution of (a).

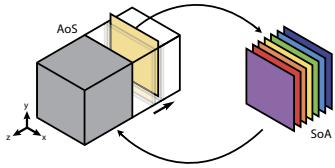


(c) HLLC approximation to the exact solution of (a).

Figure 1. (a) Exact solution of the Riemann problem for the 1D Euler equations with intermediate states $\boldsymbol{W}^j$ for $j = 1, 2$. "r" denotes a continuous rarefaction wave, "c" a contact discontinuity and "s" a shock wave. The initial states $\boldsymbol{W}^-$ and $\boldsymbol{W}^+$ are obtained from a WENO reconstruction. (b) HLLE approximation to the exact solution (a) using a single intermediate state $\boldsymbol{W}^*$. (c) HLLC approximation to (a) using two intermediate states $\boldsymbol{W}^*_L$ and $\boldsymbol{W}^*_R$.



(a) Array of structures hierarchy for a memory block.



(b) Conversion to structure of arrays for the evaluation of RHS on the core layer.

Figure 2. (a) Memory layout for a computational block in CUBISM-MPCF. The data is stored in an array of structures (AoS) format in memory. Each cell provides enough space to store the vector of conserved variables $\boldsymbol{Q}$, illustrated by the different colors. (b) At the core layer, the AoS layout is reformatted into a structure of arrays (SoA) layout to exploit data parallelism. Note that a second copy of each block (gray) is stored in memory due to the low-storage time integrator.

2. Optimization techniques and software

CUBISM-MPCF is decomposed into three layers: *cluster*, *node* and *core* to maximize the reusability. The *cluster layer* is responsible for the domain decomposition and the inter-rank information exchange. We divide the data into blocks that are formed by 32^3 cells. Each cell is represented by the vector of conserved variables $\boldsymbol{Q}$, which we consecutively store in memory. The 3D block structure is represented in linear memory by $\boldsymbol{V}$ for a total of N computational cells. This results in an array of structures (AoS) format for the data in main memory. A second copy for each block is required due to the low-storage Runge-Kutta scheme. See Figure 2(a) for a visualization of a block.

The data is further decomposed into subdomains across the ranks in a cartesian topology with a constant subdomain size. The cluster layer dispatches the blocks for computation to the *node layer*, which is responsible for coordinating the work within each rank. The work associated to each block is exclusively assigned to one thread. To evaluate the RHS of a block, the assigned thread loads the block data and ghosts into a per-thread dedicated buffer. For a given block, the intra-rank ghosts are obtained by loading fractions of the surrounding blocks, whereas for the inter-rank ghosts data is

Table 1. Performance in fraction of the peak as well as time-to-solution (TtS) for the two accuracy levels on 96 BGQ nodes [10].

Accuracy	ALL	RHS	DT	UP	TtS (sec)
1 NR iteration	61.1%	68.5%	10.2%	2.3%	15.2
2 NR iterations	64.8%	71.7%	13.2%	2.3%	17.0

fetched from a global buffer. The node layer relies on the *core layer* for the execution of the compute kernels on each block. In order to exploit the data parallelism, the AoS layout is reformatted into a structure of arrays (SoA) layout, expressed by $2D$ slices suited for the required stencil computations in RHS, see Figure 2(b) for a visualization. The data conversion is a substage of RHS with about 1% execution time relative to the execution time of RHS (cf. Table 8 in [3]). The conservative variables stored in each block are read once by DT, UP and RHS, where the same number is written back by UP and RHS. Further micro-kernels are called within RHS, requiring further memory accesses that correspond mostly to cache hits due to the design of the blocks to fit the cache. A roofline plot of the three kernels on BGQ is shown in Figure 9 in [3].

CUBISM-MPCF employs a number of optimization techniques such as data reordering, vectorization by using QPX intrinsics, computation reordering and code fusion. These techniques are applied to increase the FLOP/Byte and FLOP/instruction ratios [3]. In addition, the code exploits mechanisms supported by the BGQ hardware, such as data prefetching, optimized memory copy functions and computation-transfer overlap using asynchronous progress communication. Finally, the software implements two levels of accuracy for floating point divisions by employing one or two iterations in the Newton-Raphson (NR) scheme. A detailed description of these techniques can be found in [10].

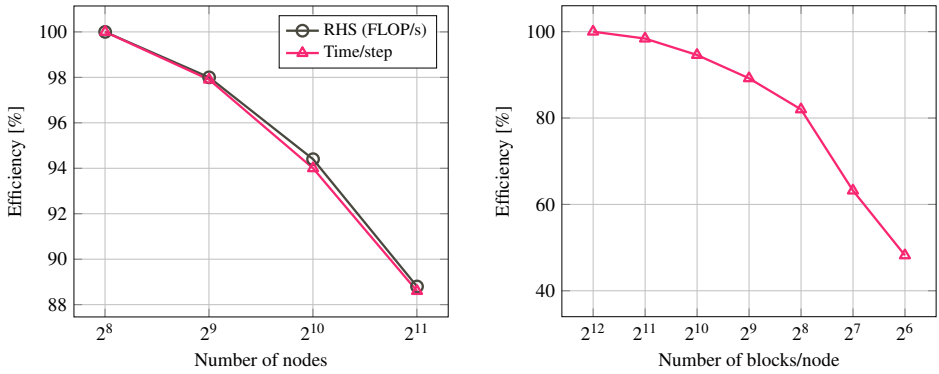
3. Results

In this section we present for the first time, performance results that concern the strong scaling efficiency of our code. We also demonstrate the effect of asynchronous progress communication and discuss, in detail, about the different levels of code fusion. Finally, we evaluate the performance of the improved approximate Riemann solver.

We compiled our code with the IBM XL C/C++ compiler (v12.1) and used the IBM Hardware Performance Monitor (HPM) Toolkit for measuring performance figures. We use $16^3 = 4096$ blocks per compute node with 32^3 computational elements in each block. Performance measurements from a previous publication are presented in Table 1. It is important to note that due to the efficient computation/transfer overlap employed in CUBISM-MPCF, these results are valid for practically any number of BGQ nodes [10].

Strong scaling. Figure 3(a) depicts the strong scaling efficiency of the cluster layer from 256 to 2048 BG/Q nodes, ranging the number of blocks per node from 4096 to 512. We observe an efficiency of 89% on 2048 nodes and that both metrics, overall time/step and achieved performance of RHS, exhibit similar scaling results.

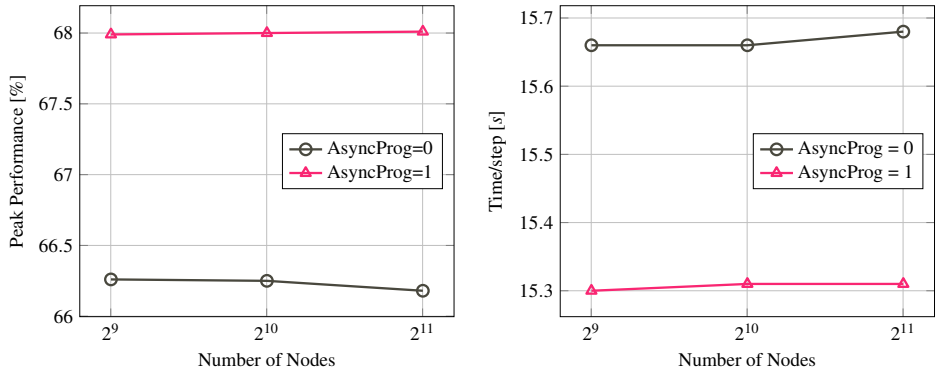
Figure 3(b) shows how the cluster layer scales with respect to the number of blocks per node. In this case, we range the number of blocks from 64 to 4096, running our experiments on 512 BGQ nodes. For down to 512 blocks, the results coincide with those in Figure 3(a). The efficiency remains above 82% for 256 blocks and drops down to 63%



(a) Efficiency with respect to achieved performance of RHS and time required per simulation step.

(b) Efficiency on 512 nodes for decreasing number of blocks per node.

Figure 3. (a) Strong scaling efficiency of the cluster layer of CUBISM-MPCF. The efficiency is illustrated for both floating point operations of RHS and time per Runge-Kutta step. (b) Efficiency for fixed number of nodes (512) and decreasing number of blocks that are processed by each node.



(a) RHS

(b) Time-to-solution

Figure 4. Effect of asynchronous progress communication on (a) peak performance and (b) time-to-solution.

and 48% for 128 and 64 blocks per node, due to underutilization of processor cores and inefficient computation/communication overlap. For instance, the 64 blocks, arranged in a 4x4x4 subdomain, correspond to 27 inner and 37 halo blocks that are processed in two phases by the 64 total OpenMP threads of each node. Therefore, the majority of threads remains idle.

Computation/transfer overlap. To overlap communication with computation, we use non-blocking point-to-point communications to exchange ghost information for the halo blocks and then process the inner blocks before calling `MPI_Waitall()`. To take full advantage of this mechanism and eliminate communication overheads, however, we had to activate the asynchronous progress communication at the PAMID layer. In addition, we had to modify the workflow in the RHS kernel as follows: after issuing the MPI calls for the ghost data, we create an OpenMP parallel region with 63 threads, each processing a single inner block. After this parallel region, we call `MPI_Waitall()` and then process

Table 2. Levels of micro-kernel fusion.

Level	Description
0	No fusion, 8 micro-kernels.
1	As Level 0 with reduced accuracy of WENO smoothness indicators.
2	Fusion of energy flux with wave speed estimator, as well as fusion of mass flux with the diagonal entry of the momentum flux, 6 micro-kernels.
3	As Level 2 with reduced accuracy of WENO smoothness indicators.
4	Aggressive fusion of all micro-kernels into a single macro-kernel with code transformations and reduced accuracy of WENO smoothness indicators.

the rest of the blocks using 64 threads [10]. In Figure 4 we observe the gain in achieved performance (left) and time-to-solution (right) on up to 2 BGQ racks, attributed to the negligible communication time of `MPI_Waitall()` due to the available hardware thread that advances MPI communications.

Effect of code fusion. The evaluation of RHS requires the computation of the numerical flux $\mathscr{F}^n(\boldsymbol{W}^-, \boldsymbol{W}^+) \in \mathbb{R}^7$ for each of the 7 components of the conserved variables $\boldsymbol{Q}$ based on the HLLE or HLLC scheme. In addition, wave speed estimates for s_L and s_R must be computed based on [8]. This results in 8 micro-kernels for each spatial direction in $3D$ space.

We have implemented and tested several depths of code fusion, which are summarized in Table 2. Code fusion reduces redundant accesses to main memory and increases instruction level parallelism. On the other hand, pressure on the register file increases, which might cause register spills. In addition to fusing the low-level micro-kernels of RHS, reduction of complexity for the WENO smoothness indicators allows for further code optimizations.

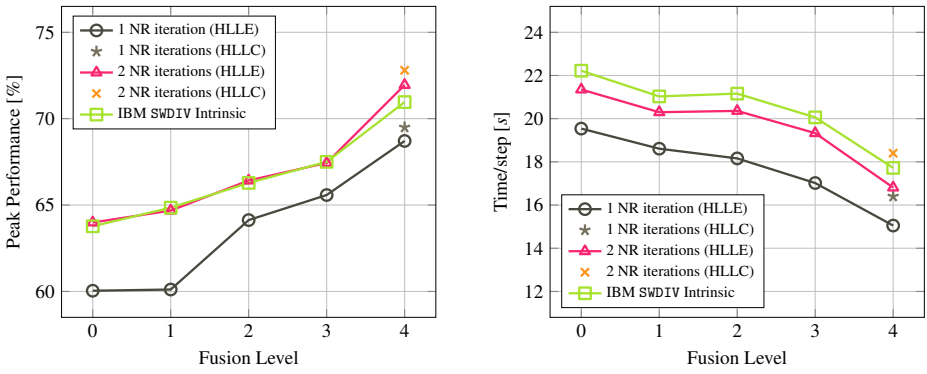
We assess the performance of these optimizations at the node layer by performing simulations on a single BGQ chip. Although this layer performs ghost reconstruction across the blocks, it completely avoids explicit communication and synchronization overheads due to MPI. In contrast to the cluster layer, the 4096 blocks are dynamically scheduled to the 64 OpenMP threads using a single parallel for loop.

Figure 5 shows the results for the different levels of code fusion on a single BGQ node, where we further distinguish between the IBM `SWDIV` intrinsic and our NR implementation for floating point divisions. Figure 5(a) illustrates the achieved peak performance, which we managed to increase by **8**% in absolute value when going from the initial version to the one with the most aggressive optimizations. Figure 5(b) shows time-to-solution based on our optimizations.

Improved numerical flux. The increased complexity of the HLLC algorithm further improves the fraction of the peak for RHS to **69.5**% and **72.8**% for one and two NR iterations, respectively, on a single BGQ node. Despite the additional logic required to distinguish between the two constant states $\boldsymbol{W}_L^*$ and $\boldsymbol{W}_R^*$ in the HLLC algorithm, we observe an improvement of about 1% in the achieved performance using the HLLC approximate Riemann solver over HLLE. The improvement is due to a slightly better density of FMA instructions in the QPX implementation of HLLC. This, however, at the expense of higher execution time, which is increased by **9**% compared to the version that uses the HLLE scheme, due to the increased number of floating point operations. Table 3 com-

Table 3. Measurements for time to solution and RHS performance on a single BGQ node for one and two NR iterations, respectively, for the HLLE and HLLC solvers. On 2048 BGQ nodes, RHS using the HLLC solver achieves 68.8% and 72.1% of peak performance for one and two NR iterations, respectively.

	One NR iteration		Two NR iterations	
	HLLE	HLLC	HLLE	HLLC
Time/step [s]	15.0	16.4	16.8	18.4
RHS [% peak]	68.7%	69.5%	72.0%	72.8%



(a) Peak performance

(b) Time-to-solution

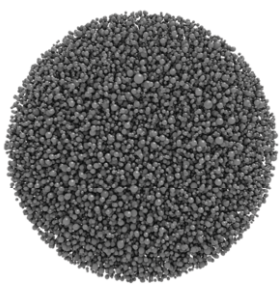
Figure 5. Achieved peak performance (a) and time-to-solution (b) for different code fusion levels on a single BGQ node. The plots compare three variants for the computation of floating point divisions. We distinguish between the NR scheme for floating point divisions with either one or two iterations, as well as the IBM intrinsic `SWDIV`. See Table 2 for a description of the different fusion levels.

pares measurements for time to solution and RHS performance on a single BGQ node for the HLLE and HLLC solvers.

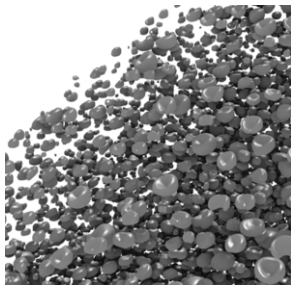
4. Simulations

We present recent results from cloud cavitation comprising 50000 vapor bubbles. Figure 6(a) displays a random spherical cloud of vapor bubbles with an initial radius of $r_c = 8\,\text{mm}$ and cloud interaction parameter of $\beta = 120$ at time $t = 0$. The bubble radii range from $50\,\mu\text{m}$ to $200\,\mu\text{m}$. The domain is discretized using 4096^3 cells. The simulation was scheduled on 2 BGQ racks and required 25000 time steps.

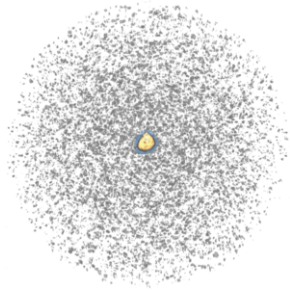
The collapse of bubbles is initiated at the most outer shell of the cloud and further propagates towards the cloud center. Instantaneous bubble shapes of these initial collapses are depicted in Figure 6(b). At the final stage of the collapse, a strong pressure shock is emitted from the center of the cloud, see Figure 6(c). The maximum pressure, an indicator for the damage potential, exceeds the ambient pressure by a factor of 25 for this particular cloud. The pressure amplification relative to the ambient pressure p_∞ is shown in Figure 7(a). Highest Mach numbers are observed prior to the final collapse of the cloud. During that time, the collapsing bubbles are located at the outer shells of the spherical cloud with re-entrant micro-jets focused towards the cloud center. The non-spherical collapse generates high velocity jets, which penetrate through the bubbles. These jets are associated with large kinetic energy, which in turn cause high Mach num-



(a) Spherical cloud with 50000 bubbles at time $t = 0$.

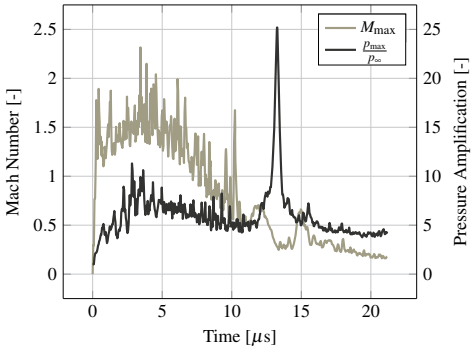


(b) Close-up of the first collapsing bubbles at the most outer shell.

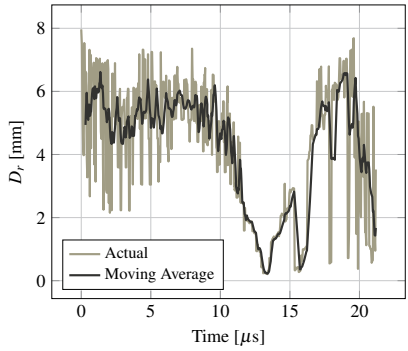


(c) Peak pressure in the cloud center at the final stage of the collapse.

Figure 6. (a) Initial state of the cloud. (b) Close-up view of the outer bubbles during the collapse process. (c) Emerging pressure wave after the final collapse of the cloud. Pressure regions are depicted in yellow (15–25 times larger than the initial ambient pressure) and blue (10–15 times larger). Refer to the online version for colors.



(a) Maximum Mach number and pressure amplification factor.



(b) Radial distance of maximum pressure from the cloud center.

Figure 7. (a) Maximum Mach number and pressure amplification during the simulation. (b) Occurrence of maximum pressure measured by radial distance from the cloud center.

bers. As the bubble collapse propagates closer towards the cloud center, the jet velocities become weaker and concentrate at the center of the cloud. As the total kinetic energy decreases, the potential energy increases due to conservation of energy. Hence the sharp decline of the maximum Mach number just prior to the emission of the strong pressure shock from the cloud center, cf. Figure 7(a).

Figure 7(b) illustrates the radial distance, D_r, of the maximum pressure, measured from the cloud center. For the first collapsing bubbles, the maximum pressure randomly jumps to radial distances further away from the cloud center. As collapses progress, the maximum pressure obeys a structured information propagation with a signal velocity of approximately $1405\,\mathrm{m/s}$, which is lower than the speed of sound in water at standard conditions due to the disruptive vapor cavities. The minimum radial distance of the maximum pressure is observed when the pressure is found to be the global maximum, see Figure 7(a) and 7(b). The emission of a strong pressure shock at the final collapse stage causes the radial distance of the maximum pressure to increase. As the radial distance in-

creases, the structure in the information propagation is lost and maximum pressure locations continue to jump randomly between spherical shells. A notable difference to the initial phase of the collapse is that the pressure maximum in the post cloud collapse regime rapidly jumps back somewhere close to the cloud center. Similar pressure rebounds are observed in spherical collapse of a single bubble.

5. Conclusions

The employment of algorithmic and software techniques and the exploitation of underlying hardware capabilities have enabled CUBISM-MPCF to reach 72% of the theoretical peak performance of BGQ platforms, allowing us to perform simulation studies of cloud cavitation collapse at unprecedented scale. Our approach is generally applicable and can be adopted to enhance the performance of all uniform grid based solvers [2]. We are currently enhancing our software with wavelet adapted grids for multiresolution flow simulations.

Acknowledgements. We wish to thank Dr. R. Walkup (IBM Watson) for providing us the HPM toolkit. We acknowledge support by the DOE INCITE project CloudPredict and the PRACE projects PRA091 and Pra09_2376.

References

[1] D. P. Schmidt and M. L. Corradini. The internal flow of Diesel fuel injector nozzles: A review. International Journal of Engine Research, 2(1):1–22, 2001.

[2] Tetsuya Kodama and Kazuyoshi Takayama. Dynamic behavior of bubbles during extracorporeal shockwave lithotripsy. Ultrasound in medicine & biology, 24(5):723–738, 1998.

[3] Diego Rossinelli, Babak Hejazialhosseini, Panagiotis Hadjidoukas, Costas Bekas, Alessandro Curioni, Adam Bertsch, Scott Futral, Steffen J Schmidt, Nikolaus Adams, Petros Koumoutsakos, et al. 11 pflop/s simulations of cloud cavitation collapse. In High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for, pages 1–13. IEEE, 2013.

[4] Eric Johnsen and Tim Colonius. Implementation of WENO schemes in compressible multicomponent flow problems. Journal of Computational Physics, 219(2):715 – 732, 2006.

[5] JH Williamson. Low-storage Runge-Kutta schemes. Journal of Computational Physics, 35(1):48–56, 1980.

[6] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. Journal of computational physics, 115(1):200–212, 1994.

[7] A. Harten, P. Lax, and B. van Leer. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. SIAM Review, 25(1):35–61, 1983.

[8] B. Einfeldt. On Godunov-type methods for gas dynamics. SIAM Journal on Numerical Analysis, 25(2):294–318, 1988.

[9] E. F. Toro, M. Spruce, and W. Speares. Restoration of the contact surface in the HLL-Riemann solver. Shock Waves, 4(1):25–34, 1994.

[10] Panagiotis Hadjidoukas, Diego Rossinelli, Babak Hejazialhosseini, and Petros Koumoutsakos. From 11 to 14.4 pflops: Performance optimization for finite volume flow solver. In 3rd International Conference on Exascale Applications and Software (EASC 2015), pages 7–12, 2015.

[2] The software can be downloaded from GitHub, https://github.com/cselab/CUBISM-MPCF.