11 PFLOP/s Simulations of Cloud Cavitation Collapse

Diego Rossinelli[1], Babak Hejazialhosseini[1], Panagiotis Hadjidoukas[1],
Costas Bekas[2], Alessandro Curioni[2], Adam Bertsch[3], Scott Futral[3],
Steffen J. Schmidt[4], Nikolaus A. Adams[4] and Petros Koumoutsakos[1]
[1]Professorship for Computational Science, ETH Zürich, Switzerland
[2]IBM Research Division, Zürich Research Laboratory, Switzerland
[3]Lawrence Livermore National Laboratory, U.S.A.
[4]Institute of Aerodynamics and Fluid Mechanics, TU München, Germany

ABSTRACT

We present unprecedented, high throughput simulations of cloud cavitation collapse on 1.6 million cores of Sequoia reaching 55% of its nominal peak performance, corresponding to 11 PFLOP/s. The destructive power of cavitation reduces the lifetime of energy critical systems such as internal combustion engines and hydraulic turbines, yet it has been harnessed for water purification and kidney lithotripsy. The present two-phase flow simulations enable the quantitative prediction of cavitation using 13 trillion grid points to resolve the collapse of 15'000 bubbles. We advance by one order of magnitude the current state-of-the-art in terms of time to solution, and by two orders the geometrical complexity of the flow. The software successfully addresses the challenges that hinder the effective solution of complex flows on contemporary supercomputers, such as limited memory bandwidth, I/O bandwidth and storage capacity. The present work redefines the frontier of high performance computing for fluid dynamics simulations.

Categories and Subject Descriptors

[**Peak performance, Time-to-solution**]

1. INTRODUCTION

Today's advances in supercomputers enable engineers to design effective solutions for some of the most pressing challenges of our century such as energy and the environment. Fluid dynamics are a critical aspect in the design and operation of such engineering systems. Here, we develop software for the simulation of cloud cavitation collapse, a phenomenon of enormous economic and ecological impact as it pertains to the erosion of liquid-fuel injectors, hydropower turbines and ship propellers. With 20% of the world's energy resources currently spent in transportation by vehicles running on liquid fuels, their proper operation is of paramount importance. In particular, further reduction in CO_2 emissions requires improving the efficiency of internal combustion (IC) engines which in turn implies high-pressure fuel injection systems. Precise fuel injection control and enhanced fuel-air mixing implies high liquid fuel injection pressures (e.g. 2500 bar for Diesel engines with combustion chamber pressure of 50 bar). Under such conditions, liquid fuel can undergo vaporization and subsequent recondensation in the combustion chamber resulting in shock waves with pressure peaks up to two orders of magnitude larger than the ambient pressure [67]. When such shock waves impact on solid walls they can cause material erosion of the combustion chamber and limit the lifetime of the fuel injectors. Similarly, the prevention of cavitation is essential for the design of ship propellers, cryogenic pumps in the aerospace industry [75], pipelines and turbines. On the other hand, the destructive potential of cavitation can be harnessed, as already realized by the mantis shrimp to stun its prey, to remove kidney stones [50], purify water, and to develop innovative drilling and emulsification techniques [13].

Currently, the quantitative prediction of cavitation phenomena is severely limited. Cavitation, in particular as it occurs in realistic conditions, presents a formidable challenge to experimental and computational studies due to its geometric complexity and the disparity of its spatiotemporal scales. Consequently, engineers usually employ simplified models to account for its effects in their designs. Simulations of cloud cavitation collapse are very demanding in terms of numbers of operations, system size and memory traffic. They require two phase flow solvers capable of capturing interactions between multiple deforming bubbles, traveling pressure waves, formation of shocks and their impact on solid walls, over a multitude of spatiotemporal scales. In this paper, we integrate accurate numerical methods with effective software that harnesses the power of modern supercomputers, to present unprecedented simulations of cloud cavitation collapse.

Cloud cavitation collapse involves bubbles covering about 50% of the computational domain, while pressure gradi-

SC13 November 17-21, 2013, Denver, CO, USA
ACM 978-1-4503-2378-9/13/11
http://dx.doi.org/10.1145/2503210.2504565

ents propagate and interact with flow structures throughout the flow field. As further discussed and justified in Section 7 we chose a uniform resolution over an Adaptive Mesh Refinement (AMR) [7] or multi resolution technique [76] for the discretization of this flow field. We focus on the performance and time to solution of finite volume uniform resolution solvers that enable simulations with an unprecedented 13 trillion computational elements evolving over 10'000 to 100'000 time steps. Such simulations resolve collapsing clouds with up to 15'000 bubbles, a two order of magnitude improvement over the current state of the art.

Our high throughput software for the simulation of cloud cavitation collapse on supercomputers had to overcome the following challenges:

- I/O bandwidth. Unravelling the fast dynamics of the cloud collapse requires large numbers of time steps as well as massive numbers of grid points to capture the multiple bubble interactions. These requirements impose high demands in terms of storage and I/O bandwidth. The serialization to file of the simulation state would involve I/O operations on Petabytes of data, leading to an I/O time that would overwhelm the actual simulation effort.

- Operational intensity. In a finite volume solver, the evaluation of the fluxes can be regarded as a product of a large, sparse and data-dependent matrix with a large state vector containing the flow quantities. This type of operation is not expected to perform at large fractions of the nominal peak performance. As suggested by the roofline model [79], in order to fully utilize the available computing power, software must feature computational kernels with operational intensities[1] of at least 10 FLOP /Byte. While numerical schemes for compressible flow simulations can involve a large amount of operations, devising compute kernels exhibiting such high operational intensities is challenging.

- FLOP/instruction density. Current computing hardware is designed to execute relatively high FLOP/instruction density text. Cores are meant to perform a smaller amount of data accesses compared to the number of floating point operations. Algorithms in this context must employ data reordering to enforce more spatial locality, however, at the cost of increased data shuffling and integer instructions. This in turn leads to a decrease in the FLOP/instruction density of the compute kernels [31, 4, 41].

- Software productivity. The last, and arguably the most important challenge, is the current gap between performance and productivity in particular as it manifests itself in flow simulation software. The current lack of productivity in simulation software is considered to be the most detrimental bottleneck in computational science [18, 46, 81] especially due to the fast evolution of hardware.

The present work extends the design and implementation of CUBISM-MPCF which was shown to reach 30% of the nominal peak performance on Cray supercomputers employing up to 250 billion computational elements for gas dynamics simulations [33]. The software has been extensively reengineered to take advantage of the novel features of the the IBM Blue Gene/Q (BGQ) platform and to simulate cavitation collapse dynamics using up to 13 trillion computational elements. The performance of the software is shown to reach an unprecedented 11 PFLOP/s on 1.6 million cores corresponding to 55% of the peak on the 20 PFLOP/s Sequoia supercomputer. These results massively improve the current state of the art in flow simulations and open new frontiers for Computational Fluid Dynamics. Furthermore, the software introduces a first of its kind efficient wavelet based compression scheme, in order to decrease the I/O time and the disk footprint of the simulations. The scheme delivers compression rates up to 100 : 1 and takes less than 1% of the total simulation time.

The paper is organized as follows: in Section 2 we discuss the current state of the art and in Section 3 we present the governing equations, their numerical discretization and their computer implementation. The computing platforms on which we performed simulations of cloud collapse are described in Section 4, while in Section 5 and Section 6 we describe our algorithmic and implementation innovations, respectively. The results of the production simulations are presented in Section 7, and in Section 8 we present a detailed analysis of the performance. We summarize our findings in Section 9 along with a discussion on the implications of the present work on future supercomputing architectures.

2. CURRENT STATE OF THE ART

The prevention and exploitation of the destructive capacity of cavitation for engineering and medical applications ranging from pressurized diesel injection systems to extracorporeal shock wave lithotripsy, has been the focus of an ever increasing number of investigations over the last 50 years [49, 29, 5, 30, 14, 10, 9]. However, current estimates of cavitation phenomena are largely based on the theory of single bubble collapse as developed a century ago by Lord Rayleigh [61], and further extended by Gilmore [25] and Hickling and Plesset [35]. These studies have demonstrated the importance of compressibility in the later stages of collapse.

However, modeling cavitation on the basis of the spherical collapse of an isolated bubble entails a number of idealized assumptions and is not likely to be representative of real world problems. Such models must be enhanced to account for the presence of other bubbles and/or solid walls, which lead to the asymmetric deformation of individual collapsing bubbles [11].

The spatiotemporal scales and the violence of cavitation hinder the experimental acquisition of information that can elucidate its driving mechanisms. Experimental investigations have also largely addressed single bubble collapse [5, 74] and report quantities such as average bubble radii, while they estimate the damage potential through measurements of surface pits inside tubes and over flat surfaces [56, 30, 40, 21]. Similarly cavitation presents a formidable challenge to simulations. Blake et al. [9] studied the single bubble asym-

[1]The ratio between floating point operations and off-chip Byte traffic.

metric collapse using a boundary integral method. Johnsen and Colonius [44] have investigated the potential damage of single collapsing bubbles in both spherical and asymmetric regime with axisymmetry assumption for a range of pulse peak pressures in shock-induced collapse. Lauer et al. [51] have studied collapses of arrays of cavities under shockwaves using the sharp interface technique of Hu et al. [37].

The current state of the art in the numerical investigation of cloud cavitation are those by Schmidt et al. [68] and Adams and Schmidt [2] using a cluster of 125 vapor bubbles inside a pressurized liquid at 40 bar. To the best of our knowledge there has never been a large scale study of cavitation on supercomputer architectures.

Flow simulations on supercomputing architectures however have a long history. We note in particular a number of impressive flow simulations that range from the work of Gropp et al. [27] to those performed using AMR techniques [7]. Their use has been facilitated by powerful open source software such as Chombo [58] and FLASH [22]. Chombo [58] is a block-structured AMR package which features a number of solvers for time dependent systems of PDEs with the support for compressible, two-phase flows and problems with solid boundaries as well as for elliptic PDEs. Its performance has been shown only for elliptic problems by Wen et al. [77] to scale up to 8000 processors. FLASH [22] is a solver for hyperbolic system of equations on adaptive grids constructed by PARAMESH and it has been shown to scale up to 33k cores for terascale simulations of compressible turbulence [19]. Raptor developed by Greenough et al. [26] has been used for both uniform resolution and AMR-based simulations of compressible flows with two-phases (see [60]). The reported performance for uniform resolution simulations on a 32k-core IBM BG/L achieved 3 TFLOP/s, corresponding to 3.3% of the peak performance. For fluid-structure interaction (FSI) problems, an AMR-based solver called Uintah by Berzins et al. [8, 55] is shown to scale up to 256k cores. A high performance AMR-based solver (Nyx) has been recently introduced for hydrodynamics (and astrophysics) problems demonstrating a weak scaling efficiency of 70% on 50k cores [3] of a Cray XE6.

In the realm of uniform resolution solvers, the most recent landmark is the simulation of noise propagation of jet engines on 1 million cores on the Sequoia supercomputer by the Center for Turbulence Research at Stanford University [57]. However the percentage of the peak performance was not reported. To the best of our knowledge, the highest fraction of the peak for uniform resolution solvers was reached by an earlier version of the present software, achieving 30% of the nominal peak on 47k cores of Cray XE6 Monte Rosa [33] to study shock-bubble interactions [34].

Data dumps of large scale simulations of cavitation requires efficient data compression algorithms to alleviate I/O bottlenecks by decreasing the I/O footprint of the simulation. The large, dynamic disparity of achievable compression rates hinders the native support of data compression by high performance I/O libraries such as ADIOS [53], HDF5 [20] and PnetCFD [52]. ISOBAR [66] represents the state of the art that performs asynchronous data transfer to the dedicated I/O nodes. It was shown to achieve compression rates between 1.9X and 52.5X for three scientific datasets of less than 1 GB, on up to 2048 nodes of the Cray XK6 Jaguar cluster. We are not aware of data compression techniques for the large scale data associated with high performance computing for flow simulations, as is the case in this work.

Roofline model. The high performance techniques developed herein were guided by the roofline performance model [79]. This model reflects the disparity between the peak performance and the DRAM memory bandwidth of the underlying hardware as well as the operational intensity. Given a compute kernel with an operational intensity of 0.1 FLOP/B – for a machine with 200 GFLOP/s and 30 GB/s – the maximum achievable performance is estimated by $\min(200, 0.1 \cdot 30) = 3$ GFLOP/s. Any kernel with an operational intensity less than the "ridge point", 6.7 FLOP/B in this case, cannot achieve peak performance and is thus memory-bound.

3. EQUATIONS AND DISCRETIZATION

Cavitation dynamics are mainly dictated by the compressibility of the flow while viscous dissipation and capillary effects take place at orders of magnitude larger time scales [10]. Accordingly, we present a solver for the simulation of inviscid, compressible, two-phase flows by discretizing the corresponding Euler equations. The system of equations describing the evolution of density, momenta and the 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. Each simulation step involves three kernels: DT, RHS and UP (Figure 1 (left)). The DT kernel computes a time step that is obtained by a global data reduction of the maximum characteristic velocity. The RHS kernel entails the evaluation of the Right-Hand Side (RHS) of the governing equations for every cell-average. The UP kernel updates the flow quantities using a Total Variation Diminishing (TVD) scheme. Depending on the chosen time discretization order, RHS and UP kernels are executed multiple times per step.

The spatial reconstruction of the flow field is carried out on primitive quantities (velocity and pressure) in order to minimize spurious oscillations across the interface [1, 43]. The zero jump conditions for pressure and velocity across the contact discontinuities are maintained by reconstructing special functions of the specific heat ratios and correc-

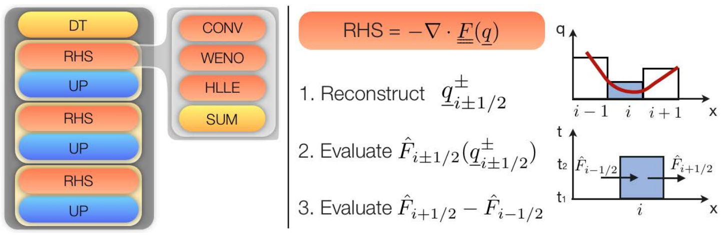
DT
RHS
UP
RHS
UP
RHS
UP
CONV
WENO
HLLE
SUM
RHS = $-\nabla \cdot \underline{\underline{F}}(\underline{q})$
1. Reconstruct $\underline{q}^{\pm}_{i\pm1/2}$
2. Evaluate $\hat{F}_{i\pm1/2}(\underline{q}^{\pm}_{i\pm1/2})$
3. Evaluate $\hat{F}_{i+1/2} - \hat{F}_{i-1/2}$
q
x
$i-1$
i
$i+1$
t
t_2
t_1
$\hat{F}_{i-1/2}$
$\hat{F}_{i+1/2}$
i
x

Figure 1: Compute kernels of the software (left), colored according to their operational intensity (blue: low, red: high), and computational stages for the RHS evaluation (right).

Table 1: BlueGene/Q supercomputers.

Name	Racks	Cores	PFLOP/s
Sequoia	96	$1.6 \cdot 10^6$	20.1
Juqueen	24	$6.9 \cdot 10^5$	5.0
ZRL	1	$1.6 \cdot 10^4$	0.2

Table 2: BQC performance table.

Cores	16, 4-way SMT, 1.6 GHz
Peak performance	204.8 GFLOP/s
L2 peak bandwidth	185 GB/s (measured)
Memory peak bandwidth	28 GB/s (measured)

tion pressures [45]. Quantities at the cell boundary are reconstructed through a Weighted Essentially Non-Oscillatory scheme (WENO) [42] that is regarded as a non-linear data dependent spatial stencil. In order to advance the RHS, we compute and sum the numerical fluxes by the HLLE (Harten, Lax, van Leer, Einfeldt) scheme [78]. We emphasize that the evaluation of the RHS requires "ghosting", i.e. the information exchange of adjacent subdomains due to the WENO scheme. The stages of the RHS evaluation are depicted in Figure 1. They involve a conversion stage from conserved to primitive quantities (CONV), a spatial reconstruction (WENO) using neighboring cells (only two in the sketch, top right) and evaluation of the numerical flux (HLLE) at the cell boundaries (bottom right) and summation of the fluxes (SUM).

4. EXPERIMENTAL SETUP

The target platform of this work is the IBM Blue Gene/Q supercomputer (BGQ), that is based on the BGQ compute chip (BQC)[31]. Table 1 reports the nominal peak performance of the BGQ installations we used.

The BQC is designed for optimal price/performance, energy efficiency and reliability [73]. This chip features enhanced versions of the A2 processor core [39], and its design is highly optimized for aggregate compute throughput performance. A BQC features 16 symmetric cores operating at 1.6 GHz (and 2 extra cores: one redundant and one for the OS), where each core supports 4 hardware threads, reaching a concurrency level of 64. The BQC core features a Quad floating-point Processing Unit (QPU) that implements the QPX instruction set, which provides a full set of arithmetic operations including fused multiply-add instructions, and a variety of operations for shuffling and selecting data. QPX instructions have a SIMD-width of 4, and are natively designed for double precision computation. The BQC core natively supports 4-way fine-grained Simultaneous Multi-Threading (SMT). Within a thread, dispatch, execution and retirement of the instructions is performed in-order. Each hardware thread has dedicated resources: the instruction buffer, the decode and the dependency units. Within a cycle, the active hardware thread can issue one general-purpose instruction and one QPX instruction.

The 4-way SMT system is designed for hiding the back-end latencies of the execution unit. Each core features a 4-way set-associative L1 data cache, 16 KB wide, which is shared across the hardware threads. Each core accesses the L2 cache through a crossbar, the central connecting structure between all of the processing units. The L2 data cache is 16-way set-associative, write-back, 32 MB wide, and is organized in 16 slices of 2 MB. The L2 memory controllers feature a programmable hash function to scatter the memory addresses across the slices. In order to hide possible latencies from the L2 data cache and DDR memory, the BGQ features an L1 cache prefetching unit.

Table 2 characterizes the performance of a single BQC. BQCs are placed in a five-dimensional network topology, with a network bandwidth of 2 GB/s for sending and 2 GB/s for receiving data, respectively. Node boards are organized in a group of 32 to form a rack, with a nominal compute performance of 0.21 PFLOP/s. Each rack features additional BQC nodes for I/O, with an I/O bandwidth of 4 GB/s per node. We note that the BQC has a relatively low ridge point and therefore kernels that exhibit operational intensities higher than 7.3 FLOP/off-chip Bytes are compute-bound.

In addition we consider two other computing platforms at Centro Svizzero di Calcolo Scientifico (CSCS), Switzerland: Monte Rosa and Piz Daint. Monte Rosa is a Cray XE6 featuring compute nodes with 2P AMD CPUs based on the Bulldozer micro-architecture. The nominal peak performance of one node is 540 GFLOP/s and the measured peak memory bandwidth is 60 GB/s, aggregate. The ridge point is thus located at 9 FLOP/Byte. Piz Daint is a Cray XC30 and its compute nodes are based on the Sandy Bridge micro-architecture. The nominal peak performance and measured peak memory bandwidth are 670 GFLOP/s and 80 GB/s, respectively, with a ridge point located at 8.4 FLOP/Byte.

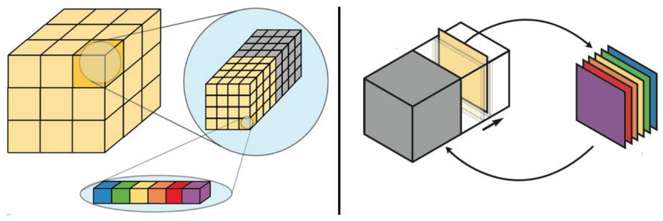
Figure 2: Grid blocks containing cells in AoS format (left), and the AoS/SoA conversion during the evaluation of the RHS (right).

Table 3: Potential gain due to data-reordering.

	RHS	DT	UP
Naive	1.4 FLOP/B	1.3 FLOP/B	0.2 FLOP/B
Reordered	21 FLOP/B	5.1 FLOP/B	0.2 FLOP/B
Factor	15X	3.9X	1X
Max. gain	5.2X	3.9X	-

5. KEY DECISIONS

The present software addresses simulation challenges in terms of floating point operations, memory traffic and storage capacity. The following design choices were made so as to overcome the limitations in memory and I/O bandwidth experienced by current supercomputers.

Minimize the compulsory memory traffic. A significant algorithmic decrease in the total memory traffic has a direct impact in terms of time to solution. To this effect we implemented:

- Low-storage time stepping schemes, to reduce the overall memory footprint.
- High-order spatiotemporal discretization schemes, to decrease the total number of steps.

In addition, a high order spatial discretization increases the per-step FLOP count, potentially allowing kernels to reach higher performance. Its main drawback however is the increased stencil size, which in turn can degrade the operational intensity. This issue can be resolved by performing data reordering [38, 54, 28] and cache-aware techniques [71] as described in the next paragraph. In this work we employ a third-order low-storage TVD Runge-Kutta time stepping scheme [80], combined with a fifth order WENO scheme [42]. We opt for a discretization scheme that exhibits a spatial order higher than the temporal one so as to better capture the sharp pressure gradients associated with the shock waves.

Maximize FLOP/Byte and FLOP/instructions. For the considered platforms, the roofline model predicts that the performance of the kernels is likely to be memory-bound. Hence we seek computational schemes that exhibit higher operational intensities. We employ:

- Data reordering techniques, to increase spatial locality.
- SIMD-friendly memory layouts for vectorization.
- Computation reordering to increase temporal locality.
- Take advantage of platform-specific instructions.

Data reordering is achieved by grouping the computational elements into 3D blocks of contiguous memory, and reindexing the blocks with a space-filling curve [33, 28]. Blocks are organized in an AoS format as it provides us with the flexibility to easily change or extend the layout of the computational elements (Figure 2, left). Table 3 reports the potential gain on BGQ due to data reordering. Here, we consider blocks of 32 elements in each direction, with extra memory necessary for the evaluation of the RHS (Figure 2, right, gray area).

To effectively operate on blocks we consider "data-slices", i.e. SIMD-friendly temporary data structures that lay out a multiple-of-four number of elements. These data structures are 32- or 16-byte aligned and have a small memory footprint. Their SoA format renders the computation amenable to vectorization.

The computation reordering takes place when evaluating the RHS. As depicted in Figure 2 (right), the kernel operates on 2D slices in the z-direction and performs directional sweeps to evaluate the x-, y- and z-fluxes. The evaluation of the RHS is then performed by exclusively using the ring buffers, and it is put back in the temporary area of the block. When possible, we "micro-fuse" different kernel substages together, i.e. we mix the instructions coming from subsequent computational patterns, so as to further increase temporal locality[2].

Three QPX instructions are particularly useful here: interlane permutations, fused multiply-add instructions and conditional selections, mostly used at the WENO and HLLE stages. The substages of every kernel benefit to different extents, from fused multiply-add instructions.

Efficient wavelet-based data compression. The large number of computational elements employed in the present simulations impose severe requirements on I/O time and storage. A critical component of this work is the development of efficient wavelet based data compression schemes to address this issue. A highly parallel implementation of the compression schemes was necessary so as to have a negligible impact on the total simulation time. Our design relies on the following features:

- Data dumps performed only for p and Γ quantities.
- Fourth-order interpolating wavelets, on the interval.
- Lossy compression: detail coefficients are decimated.
- Lossless encoding of the decimated data.
- In-place transform, decimation and encoding.
- Data compression of only one quantity at a time.
- Parallel granularity corresponding to one block.
- A dedicated decimation buffer for each thread.
- A global buffer per rank for the encoded data.

Data dumps are performed exclusively on p and Γ as they represent the main quantities of interest for the study and visualization of the cloud collapse dynamics. Wavelets enable data de-correlation while the separability of their associated filters leads to a very efficient forward wavelet transform

[2]Micro-fusion allows us also to manually perform common subexpression eliminations that are not visible to the compiler.

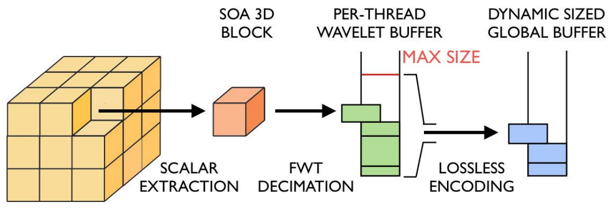

Figure 3: Data flow of the compression scheme.

(FWT). We employ fourth-order interpolating wavelets [17, 59] as they provide a balanced trade-off between compression rate and computational cost, and can be readily 4-way vectorized. In terms of accuracy, it is guaranteed that the decimation will not lead to errors larger than the threshold ϵ. The significant detail coefficients are further compressed by undergoing a lossless encoding with an external coder, here the ZLIB library [23]. Alternatively efficient lossy encoders can also be used such as the zerotree coding scheme [72] and the SPIHT library [48]. The "on the interval" property [12] allows us to consider individual blocks as independent datasets, enabling the parallel transform of all the blocks. We perform all the compression substages in-place to minimize the memory overhead. By processing one quantity at a time we enforce it to be at most 10% of the simulation footprint. Instead of encoding the detail coefficients of each block independently, we concatenate them into small, per-thread buffers and we encode them as a single stream. The detail coefficients of adjacent blocks are expected to assume similar ranges, leading to more efficient data compression. This process is depicted in Figure 3.

6. SOFTWARE

The software is built upon a core of computational techniques that reflect the strategies described in Section 5. During the software development, the primary objectives were set to high productivity/low development time. We embraced the agile development principles [70] across a team of 3 programmers. The software was initially developed in less than 9 person months, and it has been extended and optimized on BGQ platforms with a 7 man months effort. Software reusability and flexibility were achieved by adopting a set of software design patterns [24]. The short development time is reflected in the number of lines, which is approximately 20'000. The code is spread in about 100 C++ classes with the largest class containing 1200 lines of code.

Abstraction layers. The software is conceptually decomposed into three layers: *cluster, node, and core.* This organization increases reusability, allows to perform layer-specific optimizations and has been shown to provide with an adequate level of abstraction for rapidly prototyping new simulations. The *cluster layer* is responsible for the domain decomposition and the inter-rank information exchange. The computational domain is decomposed into subdomains across the ranks in a cartesian topology with a constant subdomain size. The cluster layer dispatches the prepared blocks for computation to the node layer. The *node layer* is responsible for coordinating the work within the ranks. 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 fetched from a global buffer. The node layer relies on the core layer for the execution of the compute kernels. The *core layer* is responsible for the execution of the compute kernels, namely RHS, UP, SOS and FWT. This layer is the most critical in terms of performance.

Enhancing rank-level parallelism. The parallelism across the cluster is achieved with the MPI library. During the evaluation of the RHS, blocks are divided in two parts: halo and interior. Non-blocking point-to-point communications are performed to exchange ghost information for the halo blocks. Every rank sends 6 messages to its adjacent neighbors, the corresponding message size ranges between 3 MB and 30 MB. Given these message sizes, we expect to observe high communication bandwidth. While waiting for the messages, the rank dispatches the interior blocks to the node layer. For the target platform the time spent in the node layer is expected to be one order of magnitude larger than the communication time. MPI parallel file I/O is employed to generate a single compressed file per quantity. Since the size of the compressed data changes from rank to rank, the I/O write collective operation is preceded by an exclusive prefix sum. After the scan, each rank acquires a destination offset and, starting from that offset, writes its compressed buffer in the file.

Enhancing Thread-Level Parallelism (TLP). We rely on the OpenMP standard to take advantage of the TLP. We enforce optimal thread-data affinity through a depth-first thread placement layout. In order to hide potential imbalances during the evaluation of the RHS, we enforce a dynamic work scheduling and a parallel granularity of one block. The overhead incurred by this scheduling is expected to be amortized by the work per block, since it takes in the order of 10 ms (per thread). Work imbalance is also expected to appear during the wavelet-based compression due to the block-dependent work, and in the DT, due to the scalar reduction. These imbalances are however expected to have a less severe performance impact compared to those of the RHS evaluation.

Enhancing Data-Level Parallelism (DLP). We rely on explicit vectorization to effectively enforce the data-level parallelism. The benefits of this choice have been previously investigated in [15, 33]. The RHS, UP, DT kernels were written so as to accommodate explicit vectorization with QPX intrinsics. Due to its spatial access pattern and computational irregularities, the RHS is not straightforward to vectorize: it involves AoS/SoA conversions, data reshuffling for stencil operations and conditional branches. Explicit vectorization has been also applied to the three substages of the FWT kernel, namely the one-dimensional filtering, the $x-y$ transpositions of slices, and the $x-z$ transpositions of the entire dataset. During the wavelet transform, these stages

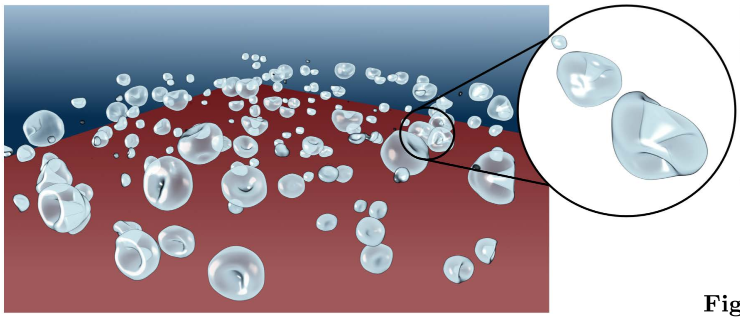

Figure 4: Overview of asymmetric deformations of the bubbles towards the center of the cluster.

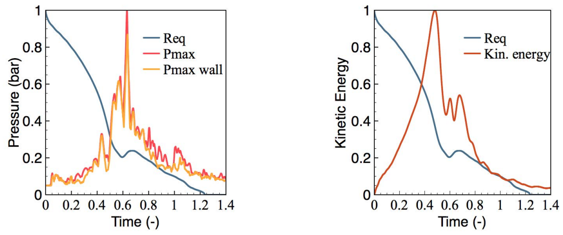

Figure 5: Temporal evolution of the maximum pressure in the flow field and on the solid wall (left), the kinetic energy of the system (right), and temporal evolution of the normalized equivalent radius of the cloud (solid blue line).

are repeatedly performed in order to construct the levels of the multiresolution analysis. While the vectorization of the transpositions are straightforward, the vectorization of the filtering poses difficulties due to the filter irregularity at the data boundaries. These are resolved by processing four y-adjacent independent data streams. This techniques is subject to an overhead due to the additional 4 x 4 transpositions necessary to operate concurrently on the four data streams.

Enhancing Instruction-Level Parallelism (ILP). The ring buffers employed in the RHS evaluation are designed to minimize the memory consumption and maximize the temporal locality. Although one slice fits in the L1 data cache (it takes about 6 KB, i.e. 40×40 scalars, single precision) the ring buffer does not fit, as it contains 6 slices. As we need to maintain seven ring buffers (one per flow quantity), we have an aggregate overhead of 250 KB per thread, totaling about 16 MB per node. This footprint is big enough and may encounter problems in fitting into the L2 data cache. The micro-fusion of the RHS substages alleviates this problem. The FWT kernel is expected to show caching issues due to $x-z$ data transpositions at the finest multiresolution levels. We perform however only two such "dangerous" transpositions: one at 128 KB and one at 16 KB. The other FWT stages are expected to be cache-friendly.

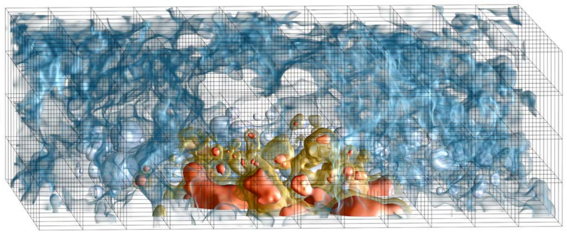

Figure 6: Domain decomposition, visualization of pressure field (low to high: translucent blue, yellow, red) and liquid/vapor interface (white) at early stages.

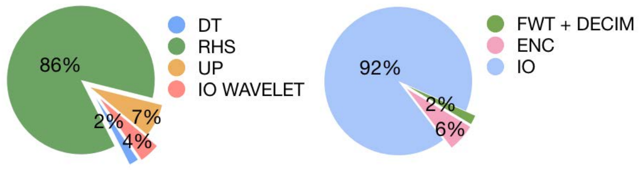

Figure 7: Time distribution of the simulation (left) and the "IO WAVELET" stage (right).

7. SIMULATIONS

We initialize the simulation with spherical bubbles modeling the state of the cloud right before the beginning of collapse. Radii of the bubbles are sampled from a lognormal distribution [30] corresponding to a range of 50-200 microns. Typical shock-bubble systems have been shown to require a resolution of 50-150 points per radius (p.p.r.), depending on the numerical approach (see [51, 32, 44]). For the bubble distributions considered in the present work, we choose a resolution such that the smallest bubbles are still resolved with 50 p.p.r.

Material properties, γ and p_c, are set to 1.4 and 1 bar for pure vapor, and to 6.59 and 4096 bar for pure liquid. Initial values of density, velocity and pressure are set to 1 kg/m^3, 0, 0.0234 bar for vapor and to 1000 kg/m^3, 0, 100 bar to model the pressurized liquid. The total simulated physical time is around 40μs. Figure 4 depicts the bubble deformation after 20μs. For the clouds considered in this work, no significant change in the integral quantities of the flow is observed beyond this time. We chose a CFL of 0.3, leading to a time step of 1ns for a total of 40'000 steps. Due to the native support for double precision computation on the BGQ, the simulations were performed in mixed precision: single precision for the memory representation of the computational elements and double precision for the computation.

The target physical system is assembled by piecing together the simulation units and keeping the same spatial resolution. The physical system is then decomposed it into subdomains and mapped to MPI ranks, as depicted in Figure 6. Every simulation unit is a cube of 1024^3 grid cells i.e. 32 blocks per dimension, and contains 50-100 bubbles. A single simulation unit requires around 30 hours of wall-clock time (about 35'000 steps) on one BGQ rack (0.2 PFLOP/s). Since the

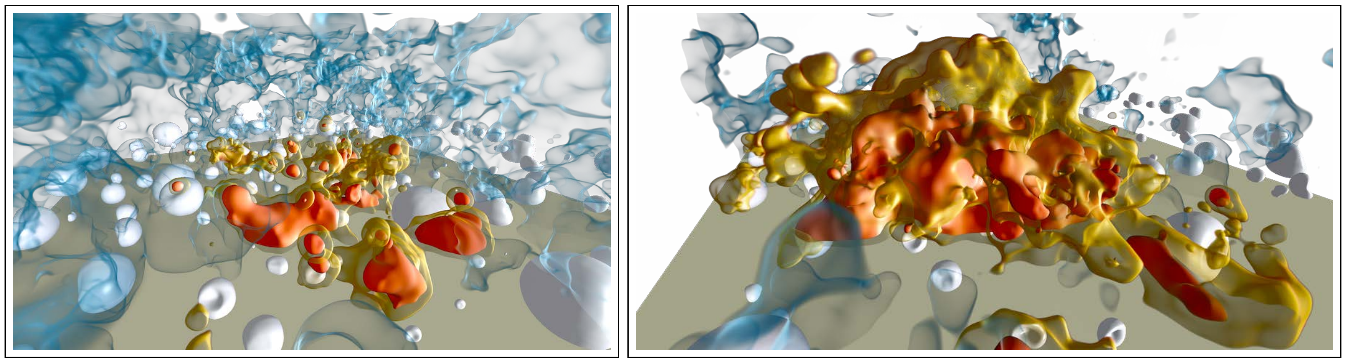

Figure 8: Volume/isosurface rendering of the pressure/interface at time $t = 0.3$ (left) and $t = 0.6$ (right). White isosurfaces identify the bubbles and orange/blue denote high/low pressure regions.

spatiotemporal resolution of larger clouds is close to those within a simulation unit, for larger simulations we do not observe a significant change in time-to-solution. The compressed data dumps are carried out every 100 steps. This extra time is included in the wall-clock time.

In Figure 8 we present visualizations of the liquid/vapor interface as well as the pressure field and the solid wall for a 9-unit simulation at $t = 0.3$ (left) and 0.6 (right). We monitor the maximum pressure in the flow field and on the solid wall, the equivalent radius of the cloud ($\sqrt[3]{3V_{\text{vapor}}/4\pi}$) and the kinetic energy of the system. At $t = 0.3$, we observe initial asymmetric deformation of the bubbles while a few bubbles have undergone the final stage of their collapse. At $t = 0.6$ a large number of bubbles have collapsed with larger collective pressure hot spots within the flow field. Shortly after this time we observe the highest values of the kinetic energy inside the flow field (Figure 5, right). At a later stage, the highest pressure is recorded over the solid wall to be about 20 times larger than the ambient pressure (Figure 5, left). We consider that this pressure is correlated with the volume fraction of the bubbles, a subject of our ongoing investigations. We also observe that the equivalent radius of the cloud (blue line in Figure 5) undergoes an expansion after $t = 0.6$ implying that some packets of vapor grow larger, indicating bubble rebound, before undergoing their final collapse.

For this simulation, the decimation threshold for visualization dumps was set to 10^{-2} for p and 10^{-3} for Γ. The observed compression rates were in the range of 20-10 : 1 for pressure and 150-100 : 1 for Γ, during the entire simulations. The total uncompressed disk space is 7.9 TB whereas the compressed footprint amounts to 0.47 TB. The compression rate for p is lower than Γ as p shows more spread spatiotemporal scales and exhibits less correlation. The left graph of Figure 7 illustrates how the wall-clock times of a simulation step and associated data compression are distributed among the kernels. Due to the high compression rate and the parallel I/O, data dumps take only up to 4% of the total time. The right side of Figure 7 shows that, within a data dump, 92% of the time is spent for parallel I/O, while 2% and 6% of the time is spent on the wavelet compression and encoding stages, respectively. Table 4 reports the work imbalance[3] for the three stages of the wavelet-based compression. As FWT is applied to every data block, imbalance for this stage is exclusively introduced by the decimation process, which is data-dependent. The encoding exhibits higher imbalance because it strongly depends on the volume of the data produced by the wavelet transform. However the encoding amounts only to 6% of the compression time and as such it does not have a significant impact.

Table 4: Work imbalance in the data compression.

	DEC	ENC	IO
Gamma	30%	390%	5%
Pressure	22%	2100%	15%

We remark here that AMR and multiresolution techniques [7, 6, 76, 58, 65, 16, 69] are known to provide advantages in time to solution. Thresholds considered in wavelet- and AMR-based simulation are usually set so as to keep the L_∞ (or L_1) errors below $10^{-4} - 10^{-7}$ [36, 76, 47, 62, 63]. Here, these thresholds lead to an unprofitable compression rate of 1.15 : 1 at best, by considering independently each scalar field, and 1.02 : 1 by considering the flow quantities as one vector field. This demonstrates that AMR techniques would not have provided significant improvements in terms of time to solution for this flow.

Throughput. On the 96 racks of Sequoia, the simulations operate on 13.2 trillion points, taking 18.3 seconds to perform a simulation step, reaching a throughput of 721 billion points per second. By projecting the #cells/second throughput of [68] on the BGQ platforms and assuming perfect scaling, the present software outperforms the current state-of-the-art by a factor of 20X. In terms of time to solution, we estimate a similar improvement.

8. RESULTS AND DISCUSSION

We assess the software performance for simulations of cloud cavitation collapse (see Section 7). Our main contributions are summarized as follows:

- Peak performance: 11 PFLOP/s, i.e. 55% of the nom-

[3]computed as $(t_{\max} - t_{\min})/t_{\text{avg}}$

Table 5: Achieved performance.

	RHS	DT	UP	ALL
1 rack [% of peak]	60%	7%	2%	53%
24 racks [% of peak]	57%	5%	2%	51%
24 racks [PFLOP/s]	2.87	0.23	0.12	2.55
96 racks [% of peak]	55%	5%	2%	50%
96 racks [PFLOP/s]	10.99	0.98	0.49	10.14

inal peak on 1.6 million cores.

- Time to solution: 20X improvement over the current state of the art.

We also note that we improved the state of the art in the geometric complexity of the flow, by simulating the evolution of 15'000 bubbles a 100X improvement over the current state of the art. The software involves 20'000 lines of code with 16 man-months of development overall.

We discuss the challenges we addressed in achieving such a performance and we identify the bottlenecks that prevented us from reaching higher fractions. The measurements on BGQ platforms collected on the total weighted GFLOP/s were reported by IBM Hardware Performance Monitor (HPM). The executables were generated with IBM XL C/C++ compiler for BGQ v12.1.

Cluster layer. Table 5 shows the performance of the individual kernels as well as the overall, obtained by carrying out different simulations on up to 96 BGQ racks. On 24 racks, the RHS kernel achieves about 57% of the peak while the performance loss compared to the 1 BGQ rack is approximately 5%. A smaller loss of %2 is observed for the DT kernel, while the UP kernel is not significantly affected. Another 2% loss in the performance is observed for the RHS kernel on the 96 BGQ racks, achieving 11 PFLOP/s. The software scales efficiently on all configurations, reaching an overall performance (i.e. over the total execution time) of more than 50% of the peak. On all platforms, the compressed data dumps take 4%-5% of the total simulation time for frequent data dumps, e.g. every hundred simulation steps, introducing an overall performance degradation of 6%-8%. For less frequent dumps, e.g. every 500 steps or more, we expect a performance degradation of 1% or less.

Node layer. We assess the performance of the node layer by performing simulation runs on a single BGQ chip. This layer includes the ghost reconstruction across the blocks but it avoids any inter-node communication and synchronization overheads due to MPI. Table 6 depicts the percentage of the peak achieved by the node layer. We observe a 2% node-to-cluster performance degradation for all but the DT kernel, which experiences a major performance loss due to the involved global scalar reduction. This has however a negligible impact to the overall performance as it corresponds to 2% of the total wall-clock time. Figure 9 (left) reports the weak scaling study and illustrates the performance (in GFLOP/s) of the three kernels. We observe good scaling for the RHS and DT kernels and lower for the UP kernel, caused by low FLOP/B ratios. Figure 9 (right) depicts the performance

Table 6: Node-to-cluster performance degradation.

	RHS	DT	UP	ALL
1 rack [% of peak]	60%	7%	2%	53%
1 node [% of peak]	62%	18%	3%	55%

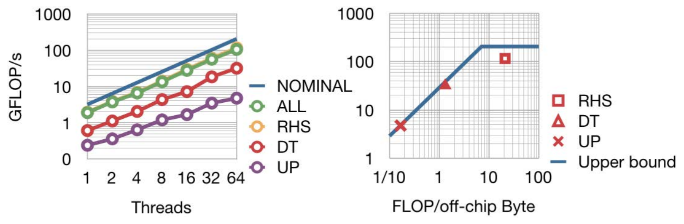

Figure 9: Performance of the node layer.

of the three kernels with respect to the roofline model. The overall performance is close to the one of the RHS kernel, which takes 89% of the total wall-clock time.

Core layer. The core layer performance is assessed by evaluating the individual kernels. Table 7 depicts the per-core measured performance for the C++ and the QPX implementations of the kernels. We observe that the QPX version of the RHS kernel reaches 65% of the nominal peak performance, indicating that the performance loss of the node layer (62%) due to the ghost reconstruction is about 3%. For the DT and UP kernels, the performance across the two layers is practically identical. The performance of the FWT kernel reaches about 10% of the peak. We also observe that the explicit vectorization with QPX intrinsics radically improves the performance of all but the UP kernel. As the RHS kernel takes up to 90% of the total simulation time, it is worthwhile to estimate a performance upper bound. According to the roofline model and the RHS operational intensity of 21 FLOP/Byte, the performance of the RHS kernel is expected to reach 100% of the nominal peak. This performance upper bound is optimistic as it discounts the FLOP/instruction density.

A second approach for estimating the upper bound of the RHS kernel is based on nominal instruction issue bandwidth. The RHS kernel is composed of five stages/microkernels: CONV, WENO, HLLE, SUM and BACK. We analyze the compiler-generated assembly of these QPX micro-kernels to estimate their FLOP/instruction density. As we seek for an upper bound here, we count as "FLOP" also the instructions for permutation, negation, conditional move, and comparisons. We then divide the FLOP count by the total amount of QPX instructions, excluding loads and stores. Table 8 depicts the theoretical upper bound of the perfor-

Table 7: Performance of the core layer.

	RHS	DT	UP	FWT
C++ [GFLOP/s]	2.21	0.90	0.30	0.40
QPX [GFLOP/s]	8.27	1.96	0.29	1.29
Peak fraction [%]	**65%**	15%	2%	10%
Improvement	3.7X	2.2X	-	3.2X

Table 8: Performance estimations based on the issue rate.

Stage	Weight	FLOP/instr	Peak
CONV	1%	1.10 x 4	55%
WENO	83%	1.56 x 4	78%
HLLE	13%	1.30 x 4	65%
SUM	2%	1.22 x 4	61%
BACK	<1%	1.28 x 4	64%
ALL	100%	1.51 x 4	**76%**

Table 9: Performance of the WENO kernel

	Baseline	Fused
Performance [GFLOP/s]	7.9	9.2
Peak fraction [%]	62%	72%
GFLOP/s improvement	-	1.2X
Time improvement	-	1.3X

mance of each stage based on the nominal instruction issue bandwidth. This model indicates that the maximum achievable performance of the RHS kernel is 76% of the nominal peak, whereas the WENO stage could achieve up to 78% of the peak. It is impossible to achieve higher peak fractions as the FLOP/instruction density is not high enough. We examine the performance gain by applying micro-fusion at the WENO kernel, the most time consuming stage of the RHS. Table 9 presents the performance results for both of the QPX non-fused and fused WENO implementations. We observe a gain of 1.1X and 1.3X with respect to the attained GFLOP/s and processor cycles respectively. The overall gain of micro-fusing the RHS kernel is 28% in cycles and 16% in GFLOP/s.

8.1 Performance portability

Part of the strategies and implementation techniques presented here have been assessed on Cray platforms [33], where the software was previously shown to achieve 30% of the nominal peak, using smaller problem sizes and a machine with less peak performance. Nevertheless, we benchmark the new version of the software and we evaluate its performance portability. The software can be compiled straightforwardly on Intel/AMD platforms by including a header file that performs the QPX/SSE conversion via macroinstructions. The development of the conversion was rather effortless, except for two intrinsics: the QPX permutation function, which is significantly more flexible than the SSE data shuffle and the QPX absolute value function, which does not have a counterpart in SSE. We have performed production simulations of cloud cavitation collapse on Piz Daint and Monte Rosa at CSCS. The nominal peak performance of the available computational resources are 0.34 PFLOP/s and 0.28 PFLOP/s, respectively, corresponding to approximately 1.5 BGQ racks. As reported in Table 10, the techniques discussed in Sections 5 and 6 are pertinent to both Cray platforms as well. The results indicate a performance improvement of 1.3X on both AMD and Intel platforms compared to [33]. It is worth to mention that the nominal peak on Piz Daint requires the use of AVX intrinsics, therefore the current QPX/SSE conversion will not provide the maximum achievable performance.

Table 10: Measured performance on the CSCS platforms, per node.

	RHS	DT	UP
Piz Daint [GFLOP/s]	269	118	13
Piz Daint [% of peak]	**40%**	18%	2%
Monte Rosa [GFLOP/s]	201	86	10
Monte Rosa [% of peak]	**37%**	16%	2%

9. CONCLUSION AND OUTLOOK

We have presented CUBISM-MPCF[4], a large-scale compressible, two-phase flow simulation software designed for studies of cloud cavitation collapse. The software is built upon algorithmic and implementation techniques that address the challenges posed by contemporary supercomputers, namely the imbalance between the compute power and the memory bandwidth as well as the limited I/O bandwidth and storage capacity. The present flow simulations on 1.6 million cores of Sequoia at LLNL, achieve an unprecedented 11 PFLOP/s corresponding to 55% of its peak. The simulations employ 13 trillion computational elements to resolve 15'000 bubbles improving by two orders of magnitude the state of the art in terms of geometric complexity. We devise a novel data compression scheme that leads to a 10-100X improvement in terms of I/O time and disk space, and takes less than 1% of the total simulation time. These achievements narrow drastically the gap between hardware performance and its effective utilization for flow simulations.

While the focus of this work was on IBM Blue Gene/Q platforms, we demonstrated performance portability across AMD and Intel micro-architectures. This indicates that the devised techniques are generalizable and likely to remain valuable for future systems. The proposed cache-awareness strategy, together with the devised DLP and ILP features, can be readily applied on current and next generation many-core accelerators. For such platforms, as already demonstrated on single-node simulations [64], most of the modifications will likely concern the enhancement of the TLP and in particular work decomposition and thread-cooperation schemes.

More investigations are necessary to identify optimal block sizes for future systems. Furthermore it is not clear whether such a two-level hierarchical indexing would provide adequate locality. Due to the I/O bottlenecks of current many-core accelerators, we envision the development of intra-node techniques to enforce computation/transfer overlap that hide potentially long latencies. Extensions aimed at targeting next generation platforms are expected to be eased by the present software design, which favors high flexibility.

We envision that large scale simulations of cloud cavitation collapse will enhance engineering models and form the foundation for complete simulations of high performance fuel injection systems. On-going research in our group focuses on coupling material erosion models with the flow solver for predictive simulations in engineering and medical applications. Ultimately, we consider that this work represents a successful step in closing the performance-productivity gap.

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

10. ACKNOWLEDGMENTS

We wish to thank Dr. Olivier Byrde, Adrian Ulrich, Teodoro Brasacchio, from the Brutus cluster at ETH Zurich, and Dr. Maria Grazia Giuffreda and Dr. Mauro Bianco from CSCS, for their exceptional support. We would like to express our gratitude to Dr. Michael Bergdorf for valuable discussions during the last few years. We thank the Jülich Supercomputing Center and the IBM T.S. Watson team for their valuable support in the development phase of this work. This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID s70.

11. REFERENCES

[1] R. Abgrall and S. Karni. Computations of compressible multifluids. *Journal of Computational Physics*, 169(2):594 – 623, 2001.

[2] N. Adams and S. Schmidt. Shocks in cavitating flows. In C. F. Delale, editor, *Bubble Dynamics and Shock Waves*, volume 8 of *Shock Wave Science and Technology Reference Library*, pages 235–256. Springer Berlin Heidelberg, 2013.

[3] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. V. Andel. Nyx: A massively parallel amr code for computational cosmology. *The Astrophysical Journal*, 765(1):39, 2013.

[4] AMD Inc. *Software Optimization Guide for the AMD 15h Family*, 2011.

[5] T. B. Benjamin and A. T. Ellis. The collapse of cavitation bubbles and the pressures thereby produced against solid boundaries. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 260(1110):221–240, 1966.

[6] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64 – 84, 1989.

[7] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53(3):484–512, 1984.

[8] M. Berzins, J. Luitjens, Q. Meng, T. Harman, C. A. Wight, and J. R. Peterson. Uintah: a scalable framework for hazard analysis. In *Proceedings of the 2010 TeraGrid Conference*, TG '10, pages 3:1–3:8. ACM, 2010.

[9] J. R. Blake, M. C. Hooton, P. B. Robinson, and R. P. Tong. Collapsing cavities, toroidal bubbles and jet impact. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 355(1724):537–550, 1997.

[10] C. E. Brennen. *Cavitation and bubble dynamics.* Oxford University Press, USA, 1995.

[11] C. E. Brennen. An introduction to cavitation fundamentals. Technical report, In: WIMRC Forum 2011 – Cavitation: Turbo-machinery & Medical Applications, 2011.

[12] A. Cohen, I. Daubechies, and P. Vial. Wavelets on the interval and fast wavelet transforms. *Applied and Computational Harmonic Analysis*, 1(1):54 – 81, 1993.

[13] A. Cucheval and R. Chow. A study on the emulsification of oil by power ultrasound. *Ultrasonics Sonochemistry*, 15(5):916 – 920, 2008.

[14] J. Dear and J. Field. A study of the collapse of arrays of cavities. *J. Fluid Mech*, 190(409):172, 1988.

[15] N. G. Dickson, K. Karimi, and F. Hamze. Importance of explicit vectorization for cpu and gpu software performance. *Journal of Computational Physics*, 230(13):5383 – 5398, 2011.

[16] M. O. Domingues, S. M. Gomes, O. Roussel, and K. Schneider. Space-time adaptive multiresolution methods for hyperbolic conservation laws: Applications to compressible euler equations, Sep 2009.

[17] D. Donoho. Interpolating wavelet transforms, 1992.

[18] S. Faulk, E. Loh, M. L. D. Vanter, S. Squires, and L. Votta. Scientific computing's productivity gridlock: How software engineering can help. *Computing in Science Engineering*, 11(6):30–39, 2009.

[19] R. T. Fisher, L. P. Kadanoff, D. Q. Lamb, A. Dubey, T. Plewa, A. Calder, F. Cattaneo, P. Constantin, I. T. Foster, M. E. Papka, S. I. Abarzhi, S. M. Asida, P. M. Rich, C. C. Glendenin, K. Antypas, D. J. Sheeler, L. B. Reid, B. Gallagher, and S. G. Needham. Terascale turbulence computation using the flash3 application framework on the ibm blue gene/lsystem. *IBM Journal of Research and Development*, 52(1-2):127–136, 2008.

[20] Folk M., Cheng A. and Yates K. Hdf5: A file format and i/o library for high performance computing applications. In *Proceedings of Supercomputing*, 1999.

[21] J. P. Franc and M. Riondet. Incubation time and cavitation erosion rate of work-hardening materials. In *The proceeding of the Sixth International Symposium on Cavitation, CAV2006*, 2006.

[22] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. Flash: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131(1):273, 2000.

[23] J.-l. Gailly and M. Adler. Zlib compression library. 2004.

[24] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design patterns: Abstraction and reuse of object-oriented design. In O. Nierstrasz, editor, *ECOOP' 93 — Object-Oriented Programming*, volume 707 of *Lecture Notes in Computer Science*, pages 406–431. Springer Berlin / Heidelberg, 1993.

[25] F. R. Gilmore. The collapse and growth of a spherical bubble in a viscous compressible liquid. Technical Report 26-4, California Institute of Technology, 1952.

[26] J. A. Greenough, B. R. De Supinski, R. K. Yates, C. A. Rendleman, D. Skinner, V. Beckner, M. Lijewski, and J. Bell. Performance of a block structured, hierarchical adaptive mesh refinement code on the 64k node ibm bluegene/l computer. *Computer*, pages 1–12, 2005.

[27] W. Gropp, D. Kaushik, D. Keyes, and B. Smith. High-performance parallel implicit CFD. *Parallel Computing*, 27(4):337–362, 2001.

[28] F. Günther, M. Mehl, M. Pögl, and C. Zenger. A cache aware algorithm for pdes on hierarchical data structures based on space filling curves. *SIAM Journal on Scientific Computing*, 28(5):1634–1650, 2006.

[29] F. G. Hammitt. Damage to solids caused by cavitation. *Philosophical Transactions of the Royal*

Society of London. Series A, Mathematical and Physical Sciences, 260(1110):245–255, 1966.

[30] I. Hansson, V. Kedrinskii, and K. A. Morch. On the dynamics of cavity clusters. *Journal of Physics D: Applied Physics*, 15(9):1725, 1982.

[31] R. Haring, M. Ohmacht, T. Fox, M. Gschwind, D. Satterfield, K. Sugavanam, P. Coteus, P. Heidelberger, M. Blumrich, R. Wisniewski, A. Gara, G.-T. Chiu, P. Boyle, N. Chist, and C. Kim. The ibm blue gene/q compute chip. *Micro, IEEE*, 32(2):48–60, 2012.

[32] N. A. Hawker and Y. Ventikos. Interaction of a strong shockwave with a gas bubble in a liquid medium: a numerical study. *Journal of Fluid Mechanics*, 701:59–97, 2012.

[33] B. Hejazialhosseini, D. Rossinelli, C. Conti, and P. Koumoutsakos. High throughput software for direct numerical simulations of compressible two-phase flows. *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 16:1–16:12, 2012.

[34] B. Hejazialhosseini, D. Rossinelli, and P. Koumoutsakos. 3D shock-bubble interactions at mach 3. *Physics of Fluids (Gallery of Fluid Motion)*, 2013.

[35] R. Hickling and M. S. Plesset. Collapse and rebound of a spherical bubble in water. *Physics of Fluids*, 7(1):7–14, 1964.

[36] M. Holmström. Solving hyperbolic pdes using interpolating wavelets. *SIAM Journal on Scientific Computing*, 21(2):405–420, 1999.

[37] X. Y. Hu, B. C. Khoo, N. A. Adams, and F. L. Huang. A conservative interface method for compressible flows. *Journal of Computational Physics*, 219(2):553–578, Dec 10 2006.

[38] Y. Hu, A. Cox, and W. Zwaenepoel. Improving fine-grained irregular shared-memory benchmarks by data reordering. In *Supercomputing, ACM/IEEE 2000 Conference*, pages 33–33, 2000.

[39] IBM. *A2 Processor User's Manual for Blue Gene/Q*, 2012.

[40] T. Ikeda, S. Yoshizawa, M. Tosaki, J. S. Allen, S. Takagi, N. Ohta, T. Kitamura, and Y. Matsumoto. Cloud cavitation control for lithotripsy using high intensity focused ultrasound. *Ultrasound in Medicine & Biology*, 32(9):1383 – 1397, 2006.

[41] Intel Corporation. *Intel® 64 and IA-32 Architectures Optimization Reference Manual*. Intel Corporation, 2009.

[42] G. Jiang and C. Shu. Efficient implementation of weighted ENO schemes. *Journal of Computational Physics*, 126(1):202–228, Jun 1996.

[43] E. Johnsen and T. Colonius. Implementation of WENO schemes in compressible multicomponent flow problems. *Journal of Computational Physics*, 219(2):715–732, Dec 10 2006.

[44] E. Johnsen and T. Colonius. Numerical simulations of non-spherical bubble collapse. *Journal of Fluid Mechanics*, 629:231–262, 5 2009.

[45] E. Johnsen and F. Ham. Preventing numerical errors generated by interface-capturing schemes in compressible multi-material flows. *Journal of Computational Physics*, 231(17):5705 – 5717, 2012.

[46] D. Kelly. A software chasm: Software engineering and scientific computing. *Software, IEEE*, 24(6):120–119, 2007.

[47] N. K. R. Kevlahan and O. V. Vasilyev. An adaptive wavelet collocation method for fluid-structure interaction at high reynolds numbers. *SIAM Journal on Scientific Computing*, 26(6):1894–1915, 2005.

[48] B.-J. Kim and W. Pearlman. An embedded wavelet video coder using three dimensional set partitioning in hierarchical trees (spiht). In *Data Compression Conference, 1997. DCC '97. Proceedings*, pages 251–260, 1997.

[49] R. T. Knapp. Recent investigations of the mechanics of cavitations and cavitation damage. *Trans. ASME*, 77, 1955.

[50] T. Kodama and K. Takayama. Dynamic behavior of bubbles during extracorporeal shock-wave lithotripsy. *Ultrasound in Medicine & Biology*, 24(5):723 – 738, 1998.

[51] E. Lauer, X. Y. Hu, S. Hickel, and N. A. Adams. Numerical investigation of collapsing cavity arrays. *Physics of Fluids*, 24(5):052104, 2012.

[52] J. Li, W.-k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale. Parallel netcdf: A high-performance scientific i/o interface. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, SC '03. ACM, 2003.

[53] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin. Flexible io and integration for scientific codes through the adaptable io system (adios). In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, CLADE '08, pages 15–24. ACM, 2008.

[54] J. Mellor-Crummey, D. Whalley, and K. Kennedy. Improving memory hierarchy performance for irregular applications using data and computation reorderings. *International Journal of Parallel Programming*, 29(3):217–247, 2001.

[55] Q. Meng and M. Berzins. Abstract: Uintah hybrid task-based parallelism algorithm. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 1431–1432, 2012.

[56] K. Mørch. Energy considerations on the collapse of cavity clusters. *Applied Scientific Research*, 38:313–321, 1982.

[57] A. Myers. Stanford researchers break million-core supercomputer barrier, January 2013.

[58] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Mondiano, D. B. Serafini, and B. Van Straalen. Chombo software package for amr applications design document. Technical report, Lawrence Berkeley National Laboratory, 2003.

[59] R. Prosser. Resolution independent lifted interpolating wavelets on an interval. 2009.

[60] D. Ranjan, J. H. J. Niederhaus, J. G. Oakley, M. H. Anderson, J. A. Greenough, and R. Bonazza. Experimental and numerical investigation of shock-induced distortion of a spherical gas inhomogeneity. *Physica Scripta Volume T*,

132(1):014020, Dec. 2008.

[61] L. Rayleigh. Viii. on the pressure developed in a liquid during the collapse of a spherical cavity. *Philosophical Magazine Series 6*, 34(200):94–98, 1917.

[62] S. J. Reckinger, D. Livescu, and O. V. Vasilyev. Adaptive wavelet collocation method simulations of Rayleigh-Taylor instability. *Physica Scripta*, T142, DEC 2010. 2nd International Conference and Advanced School on Turbulent Mixing and Beyond, Abdus Salam Int Ctr Theoret Phys, Trieste, ITALY, JUL 27-AUG 07, 2009.

[63] S. M. Reckinger, O. V. Vasilyev, and B. Fox-Kemper. Adaptive volume penalization for ocean modeling. *Ocean Dynamics*, 62(8):1201–1215, AUG 2012.

[64] D. Rossinelli, B. Hejazialhosseini, D. Spampinato, and P. Koumoutsakos. Multicore/multi-gpu accelerated simulations of multiphase compressible flows using wavelet adapted grids. *SIAM J. Scientific Computing*, 33(2), 2011.

[65] O. Roussel, K. Schneider, A. Tsigulin, and H. Bockhorn. A conservative fully adaptive multiresolution algorithm for parabolic pdes. *Journal Of Computational Physics*, 188(2):493–523, Jul 2003.

[66] E. R. Schendel, S. V. Pendse, J. Jenkins, D. A. Boyuka, II, Z. Gong, S. Lakshminarasimhan, Q. Liu, H. Kolla, J. Chen, S. Klasky, R. Ross, and N. F. Samatova. Isobar hybrid compression-i/o interleaving for large-scale parallel i/o optimization. In *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, pages 61–72, New York, NY, USA, 2012. ACM.

[67] 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.

[68] Schmidt et al. Assessment of the prediction capabalities of a homogeneous cavitation model fir the collapse characteristics of a vapour-bubble cloud. In *WIMRC 3rd International Cavitation Forum*, Coventry, U.K., 2011.

[69] K. Schneider and O. V. Vasilyev. Wavelet methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 42(1):473–503, 2010.

[70] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, 1st edition, 2001.

[71] J. Sermulins, W. Thies, R. Rabbah, and S. Amarasinghe. Cache aware optimization of stream programs. *SIGPLAN Not.*, 40(7):115–126, June 2005.

[72] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *Signal Processing, IEEE Transactions on*, 41(12):3445–3462, 1993.

[73] I. B. G. team. Design of the ibm blue gene/q compute chip. *IBM Journal of Research and Development*, 57(1/2):1:1–1:13, 2013.

[74] Y. Tomita and A. Shima. Mechanisms of impulsive pressure generation and damage pit formation by bubble collapse. *Journal of Fluid Mechanics*, 169:535–564, Aug. 1986.

[75] Y. Utturkar, J. Wu, G. Wang, and W. Shyy. Recent progress in modeling of cryogenic cavitation for liquid rocket propulsion. *Progress in Aerospace Sciences*, 41(7):558 – 608, 2005.

[76] O. Vasilyev and C. Bowman. Second-generation wavelet collocation method for the solution of partial differential equations. *Journal of Computational Physics*, 165(2):660–693, DEC 10 2000.

[77] T. Wen, J. Su, P. Colella, K. Yelick, and N. Keen. An adaptive mesh refinement benchmark for modern parallel programming languages. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, SC '07, pages 1–12. ACM, 2007.

[78] B. Wendroff. Approximate Riemann solvers, Godunov schemes and contact discontinuities. In Toro, EF, editor, *Godunov Methods: Theory and Applications*, pages 1023–1056, 233 Spring St, New York, NY 10013 USA, 2001. London Math Soc, Kluwer Academic/Plenum Publ.

[79] S. Williams, A. Waterman, and D. Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52:65–76, 2009.

[80] J. Williamson. Low-Storage Runge-Kutta Schemes. *Journal of Computational Physics*, 35(1):48–56, 1980.

[81] G. Wilson. Where's the real bottleneck in scientific computing? *American Scientist*, 2006.