From 11 to 14.4 PFLOPs: Performance Optimization for Finite Volume Flow Solver

Panagiotis E. Hadjidoukas
phadjido@mavt.ethz.ch

Babak Hejazialhosseini
hbabak@mavt.ethz.ch

Diego Rossinelli
diegor@mavt.ethz.ch

Petros Koumoutsakos
petros@ethz.ch

Chair of Computational Science, D-MAVT ETH Zürich, 8092 Switzerland

ABSTRACT
CUBISM-MPCF is a compressible, two-phase flow solver that has performed unprecedented flow simulations, employing 13 trillion computational elements to study cavitation collapse of a cloud composed of 15'000 bubbles. The code had been deployed on 1.6 million cores of the Sequoia IBM BlueGene/Q supercomputer, reaching initially 11 PFLOPs, corresponding to 55% of its nominal peak performance. This paper reports, for the first time, the techniques used to extend the performance of the code by 30% reaching 14.4 PFLOPs on BlueGene/Q systems. The achieved 72% of the peak performance constitutes to date the best performance for flow simulations in supercomputer architectures. Our techniques take advantage of the underlying hardware capabilities and were applied through all levels in the software abstraction aiming at full exploitation of the inherent instruction/data-, thread- and cluster-level parallelism. The software advances by two to three orders of magnitude the state-of-the-art both in terms of solution time and geometric complexity of the flow. We believe that the present methods are relevant to all grid based solvers and as such they may serve to enhance the capabilities across different areas of simulation based science.

Keywords
High performance computing, flow simulations, supercomputers

1. INTRODUCTION
Vehicles operating with liquid fluids are the most dominant form of transportation and they account for more than 20% of the world's energy resources. Their energy efficient operation is of paramount importance as further reduction in CO₂ emissions requires improving the efficiency of internal combustion 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. In such conditions, liquid fuel can undergo vaporization and subsequent re-condensation in the combustion chamber. Clusters of vapor bubbles incepted in such flow conditions are referred to as ‘cloud cavitation’. Their collapse induces pressure peaks up to two orders of magnitude larger than the ambient pressure [10]. When such pressures are exerted on solid walls they can cause material erosion of the combustion chamber and limit the lifetime of the fuel injectors. The damaging effects of cloud cavitation collapse are also detrimental to the operation of marine propellers and turbomachinery yet they can be harnessed in medical applications ranging from kidney lithotripsy to drug delivery [7].

Realistic simulations require two phase flow solvers capable of capturing interactions between multiple deforming bubbles, pressure waves and shocks and their interaction with turbulent flow fields. CUBISM-MPCF is a high throughput software (ACM Gordon Bell Prize 2013) [8] that addresses challenges critical to flow simulations in terms of floating point operations, memory traffic and storage capacity. The software has been designed to take advantage of the features of the IBM BlueGene/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. Furthermore, the software introduces a first of its kind efficient wavelet based compression scheme, in order to decrease the I/O time and the footprint of the simulations. The scheme delivers compression rates up to 100 : 1 and takes less than 1% of the total simulation time.

As collapsing bubbles cover about 50% of the computational domain, we chose a uniform resolution over an adaptive mesh refinement [2] or a multi resolution technique [11] for the discretization of this flow field. By performing simulations that resolve collapsing clouds with up to 15’000 bubbles, CUBISM-MPCF improved by two orders of magnitude the previous state of the art, set by Adams and Schmidt [1]. Considering uniform resolution solvers, simulations of noise propagation of jet engines were performed on Sequoia using similar number of computational elements but with significantly lower performance in terms of time to solution and percentage of the peak [3]. Regarding performance, an earlier version of the present software achieved 30% of the nominal peak on 47k cores of Cray XE6 Monte Rosa [4] for studies of shock-bubble interactions.

In this work, we first discuss our key software design decisions for addressing simulation challenges with regard to floating point operations and memory traffic. Then, we present and evaluate optimization techniques that allowed us to improve the initial performance of CUBISM-MPCF from 55% to 72% of the theoretical peak on BGQ systems, which is translated to the increase from 11 to 14.4 PFLOP/s on Sequoia. These techniques take advantage of the underlying hardware capabilities and were applied at the three
2. EQUATIONS AND DISCRETIZATION

Caviton dynamics involve a complex interplay of physical processes associated with compressibility, convective and viscous dissipation. We simulated cavitation in inviscid, compressible, two-phase flows using a finite-volume discretisation of the governing Euler equations. The evolution of density, momenta and the total energy of the flow is described with the following system of equations:

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0, \\
\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho uu + p I) = 0, \\
\frac{\partial (\rho E)}{\partial t} + \nabla \cdot ((E + p) u) = 0.
\]

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

\[
\frac{\partial \phi}{\partial t} + u \cdot \nabla \phi = 0,
\]

where \( \phi = (\Gamma, \Pi) \) with \( \Gamma = 1/(\gamma - 1) \) and \( \Pi = \gamma p_v/(\gamma - 1) \). The specific heat ratio \( \gamma \) and the correction pressure of the mixture \( p_v \) are coupled to the system of equations (1) through a stiffened equation of state of the form \( \Gamma p + \Pi = E - 1/2 \rho u^2 \).

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. The DT kernel computes the time step that is obtained by a global data reduction of the maximum characteristic velocity. The RHS kernel evaluates 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 velocity and pressure \([6]\) while their zero jump conditions across the contact discontinuities are maintained by reconstructing special functions of the specific heat ratios and correction pressures. Quantities at the cell boundaries are reconstructed through the fifth-order Weighted Essentially Non-Oscillatory (WENO) scheme \([5]\). In order to advance the system, we compute the numerical fluxes by using the HLLE (Harten, Lax, van Leer, Einfeldt) scheme \([12]\). The evaluation of RHS requires information exchange of adjacent subdomains due to the WENO scheme. The RHS evaluation includes five stages/microkernels: a conversion stage from conserved to primitive quantities (CONV), a spatial reconstruction (WENO) using neighboring cells, evaluation of the numerical flux (HLLE) at the cell boundaries, summation of the fluxes (SUM) and a final stage for writing back the results (BACK).

3. HARDWARE PLATFORM

Our target platform was the IBM Blue Gene/Q supercomputer. This platform is based on the BGQ compute chip which is equipped with 16 symmetric cores operating at 1.6 GHz. A per-core Quad floating-point Processing Unit implements the QPX instruction set and has a SIMD-width of 4. Each core supports 4 hardware threads, offering a maximum concurrency of 64 on a single BGQ node. A 16 KB L1 data cache is shared across the hardware threads of a single core. Each core accesses the shared L2 data cache through a crossbar. L2 is organized in 16 slices of 2 MB and memory addresses are scattered across these slices. An L1 cache prefetching unit aims at hiding possible latencies from the L2 data cache and DDR memory.

Table 1 summarizes the main performance features of a single BGQ node. Node boards consist of 32 compute nodes and are grouped in 32 to form a rack, with a nominal compute performance of 0.21 PFLOP/s. BGQ nodes 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. Due to the relatively low ridge point of the platform, kernels that exhibit operational intensities higher than 7.3 FLOP/off-chip Bytes are compute-bound.

4. SOFTWARE LAYOUT

CUBISM-MPCF is designed to minimize compulsory memory traffic by using low-storage time stepping schemes that reduce the overall memory footprint and high-order spatiotemporal discretization schemes that decrease the total number of steps. In its current version, the solver employs a third-order low-storage TVD Runge-Kutta time stepping scheme, combined with a fifth order WENO scheme. To avoid degradation of operation intensity, we employ data reordering and cache-aware techniques. Data reordering is achieved by grouping the computational elements into 3D blocks of contiguous memory, organized in an AoS format. To effectively operate on blocks we consider SIMD-friendly temporary data structures, in SoA format, that allow for extensive use of vector intrinsics In addition, to increase temporal locality we employ computation reordering techniques when evaluating the RHS.

CUBISM-MPCF is written in C++ and parallelized using the MPI and OpenMP programming models. It is conceptually decomposed into three layers: cluster, node, and core. The cluster layer is responsible for the domain decomposition and the inter-rank information exchange based on MPI. The computational domain is decomposed into subdomains across the ranks in a cartesian topology with a constant sub-domain size. The subdomains are further decomposed into constant-sized blocks of data, which are divided into halo
5. PERFORMANCE IMPROVEMENTS

Fig 1 summarizes the initial overall performance (ALL) of CUBISM-MPCF as well as the performance of its individual kernels (RHS, DT, UP), obtained on 1, 24 and 96 BGQ racks. From 1 to 24 racks, the performance of the RHS kernel decreases from 60% to 57% of the peak. The DT kernel exhibits a 2% performance loss while the UP kernel, which does not involve any communication, remains unaffected. Another 2% loss in the performance is observed for the RHS kernel on the 96 BGQ racks, achieving 11 PFLOP/s.

The optimization techniques described in this section allowed us to improve by 31% the sustained performance of our simulations. The highest sustained peak performance reached 14.43 (previously 10.99) PFLOP/s, which corresponds to 72% (previously 55%) of the nominal peak of Sequoia, the IBM BGQ system at the Lawrence Livermore National Laboratory.

We did not introduce any algorithmic changes in our software but instead, we focused on its fine-tuning and the exploitation of the BGQ hardware. Our improvements result from addressing three major performance challenges: computation/communication overlap, memory management, and load imbalance.

### Computation/Communication overlap

Despite the use of non-blocking MPI calls and the adequate processing time of inner blocks before calling MPI_Waitall(), the cluster layer of the initial version suffered a 5% performance loss from 1 to 96 BGQ racks mainly due to inefficient communication/computation overlap observed for a large number of compute nodes.

Initially, we modified our code to post all the asynchronous calls before the MPI_Isend() calls. We managed to eliminate communication overheads by activating the asynchronous progress communication at the PAMID layer of the BGQ MPI implementation. This mechanism leads to practically zero overhead for the non-blocking point-to-point communication in the RHS kernel. In addition to setting the appropriate environment variables, we had to apply a new workloop for the rank/thread configuration of 1/64 on each BGQ node. More specifically:

- The main thread issues MPI_IRECV/ISSEND calls for the halo blocks.
- It then encounters a parallel region with 63 OpenMP threads, where each thread processes a single inner block. The number of assigned blocks per thread can be adjusted to allow for full overlap of communication with computation.
- The main thread calls MPI_Waitall() after this parallel region.
- The rest of the inner blocks and the halo ones are processed by all 64 OpenMP threads in a subsequent parallel region using a dynamically scheduled for loop.

The necessity of this scheme is attributed to the non-preemptive scheduling of software threads, which does not allow for core oversubscription. Due to the ‘free’ hardware thread that advances MPI communications in the background, the total communication time of MPI_Waitall() is negligible. In turn the performance loss of the RHS kernel at the cluster layer is minimized and becomes practically independent of the number of compute nodes used for the simulation.

### Memory management

In order to reduce L1 Data Cache Misses (DCM) and register spilling, we applied the following techniques:
1. Linear stream prefetching of data was activated with the confirmed mode and depth equal to one (default is two). A stream is confirmed when there are two L1 cache misses within 128 bytes.

2. Deactivation of compiler-based unrolling of a loop that invokes the QPX-based WENO kernel. The initially used `pragma unroll(4)` compiler directive was affecting performance due to register spilling.

3. Faster loading of ghost data both at the node and cluster layer due to more efficient unpacking of the received data. This was achieved by extensive use of the built-in `__bcopy()` function.

**Load imbalance**

The load imbalance at the cluster layer was originally manifested by significant times spent at MPI_Allreduce. Besides the varying times of MPI_Waitall, load imbalance was introduced by the intra-node scheduling scheme for block processing and the boundary conditions at the cluster level. We made the following code modifications:

1. Besides fine tuning of computation/communication overlap, the adopted block processing workflow improves load balancing because blocks are distributed evenly to the OpenMP threads of the two parallel regions mentioned above. For instance, for a typical cubic subdomain of \(16^3 = 4096\) blocks per node, 63 blocks are initially processed and then 4033 blocks are dynamically distributed among 64 threads. In contrast, the previously used workflow first assigns 2744 inner blocks and then 1352 halo blocks to 64 threads, increasing the possibility that some OpenMP threads to become idle.

2. Boundary conditions are enforced by using loop unrolling and copying memory with the `__bcopy()` function. This minimizes per-block overheads and combined with the faster loading of ghost data leads to more uniform block-processing time across the compute nodes and, thus, in better load balancing.

**Additional fine tuning options**

We observed minor performance improvements (<0.5%) in our solver for the following options:

- Use of the same optimization flag ("-O3") for all the three layers of the software: the core layer was previously compiled with "-O5".
- Decrease of the stack size of OpenMP threads from 1MB to 512KB.

**Numerical accuracy**

In addition to these advances, we extended the level of accuracy in our simulations, by introducing an additional second pass in the Newton-Raphson scheme used for the computation of the reciprocals. The previously employed single-pass scheme achieves 13.79 PFLOP/s of peak performance for the RHS Kernel. The two-pass scheme increases the computational intensity with respect to the single-pass, at the expense of slightly higher time-to-solution (12%). This, combined with the better exploitation of memory subsystem and the communication and load imbalance advances allowed us to reach 14.43 PFLOP/s for the RHS kernel.

### 6. PERFORMANCE RESULTS

We compiled CUBISM-MPCF with the same software stack and version of the IBM XL C/C++ compiler (v12.1) and used the IBM Hardware Performance Monitor (HPM) Toolkit for BGQ for measuring performance figures.

**Cluster layer**

We repeated the initial runs for identical cloud simulations and problem sizes, using 4096 blocks on each compute node with 32\(^2\) computational elements in each block. Performance measurements for the fraction of the peak and the achieved PFLOP/s on the 96-rack Sequoia BGQ system (98304 nodes, 1572864 cores) are presented in Table 2 (top and bottom respectively). In summary, the overall performance has been improved from 10.14 to 13.1 PFLOP/s for the two-pass scheme and to 11.3 PFLOP/s for the single-pass scheme.

On the 96 racks of Sequoia, our simulations operate on \(N_v=13.2\) trillion grid points and therefore, each core processes 3.89 million points every 15.2 seconds. By dividing the time per step with the number of points per core, we compute that the normalized time is equal to \(T_{w}=1.81\). \(T_{w}\) was introduced by Bermejo-Moreno et al. [3] to characterize the performance of their Hybrid solver and the best values they achieved for their turbulence simulations on Sequoia range between 16.3 and 39.0, while the achieved performance did not exceed 6.4% of the peak. By projecting the performance of [1] on the BGQ platforms and assuming perfect scaling, we compute that \(T_w=29.7\).

**Node layer**

We assess the performance of the node layer by performing simulation runs 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.

The percentage of the peak achieved by the node layer is depicted in Table 3. We observe that the overall performance of both schemes for the node layer, both in fraction of the peak and time per simulation step, is close to the corresponding performance achieved for the cluster layer on the 96 racks. We observe 0.6% and 5.6% absolute performance loss for the RHS and DT kernels, while UP is not affected as it involves local computations. The minimal performance loss for RHS demonstrates the effectiveness of our computation-communication overlap scheme. With regard to DT, the difference between the two layers is that the cluster layer performs an overall MPI reduction operation. In addition, load imbalance during the processing of blocks cannot be fully eliminated due to different boundary conditions. The loss in the overall performance is close to that of the RHS kernel, while the time required for a single simulation step increases increases by 0.3 seconds for both cases.

The third row in Table 3 shows the performance of a scheme where both reciprocal/divisions and square roots computations are performed using the native QPX-based `vec_swdiv()` and `vec_swsqrt()` functions. This scheme increases the performance of the DT kernel but delivers lower performance to RHS, which affects both the overall performance and mostly the time to solution. Based on the observed performance and accuracy of our simulations, we concluded that `vec_swdiv()` follows the two-pass scheme.
Table 2: Performance in fraction of the peak (top) and improvements in attained PFLOP/s (bottom) as well as time to solution for the initial and the updated version of the software and two accuracy levels.

<table>
<thead>
<tr>
<th></th>
<th>ALL</th>
<th>RHS</th>
<th>DT</th>
<th>UP</th>
<th>TtS  (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial (single-pass)</td>
<td>50.4%</td>
<td>54.6%</td>
<td>4.9%</td>
<td>2.4%</td>
<td>18.3</td>
</tr>
<tr>
<td>Updated (single-pass)</td>
<td>61.1%</td>
<td>68.5%</td>
<td>10.2%</td>
<td>2.3%</td>
<td>15.2</td>
</tr>
<tr>
<td>Updated (two-pass)</td>
<td>64.8%</td>
<td>71.7%</td>
<td>13.2%</td>
<td>2.3%</td>
<td>17.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Initial (single-pass)</th>
<th>Updated (single-pass)</th>
<th>Updated (two-pass)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TtS  (sec)</td>
<td>10.14</td>
<td>10.99</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>+1.16</td>
<td>+2.80</td>
<td>+1.07</td>
</tr>
<tr>
<td></td>
<td>+2.96</td>
<td>+3.44</td>
<td>+1.67</td>
</tr>
</tbody>
</table>

Table 3: Achieved performance of the node layer.

<table>
<thead>
<tr>
<th></th>
<th>ALL</th>
<th>RHS</th>
<th>DT</th>
<th>UP</th>
<th>TtS  (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial (single-pass)</td>
<td>61.9%</td>
<td>69.1%</td>
<td>15.8%</td>
<td>2.3%</td>
<td>14.9</td>
</tr>
<tr>
<td>Updated (single-pass)</td>
<td>65.5%</td>
<td>72.3%</td>
<td>19.9%</td>
<td>2.3%</td>
<td>16.7</td>
</tr>
<tr>
<td>Updated (two-pass)</td>
<td>64.9%</td>
<td>71.1%</td>
<td>20.4%</td>
<td>2.3%</td>
<td>17.6</td>
</tr>
</tbody>
</table>

Core layer

We evaluate the performance of the WENO kernel, the most time consuming stage of the RHS. Table 4 shows the performance of the two accuracy schemes (single and two-pass) for the reciprocal and for both of the QPX non-fused and fused WENO implementations. The fused WENO implementations reach 77.6% and 80.5% of the peak performance of the BGQ core (12.8 GFLOP/s), which are within 1% of their maximum theoretical performance as defined by their density of FMA operations. Kernel fusion improves the performance of WENO by 8% and 22% with respect to the attained GFLOP/s and processor cycles respectively.

Simulations

We initialize the simulation with spherical bubbles modeling the state of the cloud right before the beginning of collapse, while radii of the bubbles are sampled from a lognormal distribution corresponding to a range of 50-200 microns. For the bubble distributions, we choose a resolution such that the smallest bubbles are still resolved with 50 points per radius. Material properties, $\gamma$ and $\rho_c$, 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, 1000 bar for pure liquid, to 6.59 and 4096 bar for pure vapor. We chose a CFL of 0.3, leading to a time step of 1 ns for a total of 40'000 steps. The simulations were performed in mixed precision: single precision for the memory representation of the computational elements and double precision for the computation.

Recent simulation results are presented in Fig. 2 providing an unprecedented level of detail for the evolution of bubble interfaces and pressure isosurfaces in a collapsing cloud of cavities over a solid wall. In Fig. 2(left) we visualize the liquid/vapor interface as well as the pressure field and the solid wall for a simulation at $t = 0.6$. We also monitor the maximum pressure in the flow field and on the solid wall, the equivalent radius of the cloud ($\sqrt{3V_{\text{Vapor}}/4\pi}$). 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. At a later stage, the highest pressure is recorded over the solid wall to be about 20 times larger than the ambient pressure (Fig. 2, right). 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 the same figure) undergoes an expansion after $t = 0.6$ implying that some packets of vapor grow larger, indicating bubble rebound, before undergoing their final collapse.

7. CONCLUSION AND OUTLOOK

We have presented CUBISM-MPCF, 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 to classical flow solvers by contemporary supercomputers, namely the imbalance between the compute power and the memory bandwidth as well as the limited I/O bandwidth. The present flow simulations on 1.6 million cores of Sequoia achieve an unprecedented 14.4 PFLOP/s corresponding to 72% 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 consider that the techniques reported here in can be adopted to enhance the performance of all finite volume and finite difference flow solvers that employ uniform grid sizes. Our goal is to enhance this software with wavelet adapted grids for multiresolution flow simulations [9]. 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.

Acknowledgments

We wish to thank Dr. A. Curioni (IBM), Dr. C. Bekas (IBM), Dr. A. Bertsch (LLNL) and Dr. S. Futral (LLNL) for their valuable help in conducting the experiments on Sequoia, and Dr. R. Walkup (IBM) for providing us the HPM toolkit. This work was supported by DOE INCITE and PRACE Awards.
Table 4: Achieved performance of the WENO kernel (per-core).

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Version</th>
<th>Performance (GFLOP/s)</th>
<th>Peak fraction [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-pass</td>
<td>Baseline</td>
<td>9.22</td>
<td>72.0%</td>
</tr>
<tr>
<td></td>
<td>Fused</td>
<td>9.93</td>
<td>77.6%</td>
</tr>
<tr>
<td>Two-pass</td>
<td>Baseline</td>
<td>9.62</td>
<td>75.2%</td>
</tr>
<tr>
<td></td>
<td>Fused</td>
<td>10.30</td>
<td>80.5%</td>
</tr>
</tbody>
</table>

Figure 2: (Left) Volume and isosurface rendering of the pressure (high/low in yellow/orange) and the interfaces of the bubble (translucent white) at late stages of the collapse of an array of clouds. (Right) Temporal evolution of the maximum pressure in the field and on the solid wall and temporal evolution of the normalized equivalent radius of the cloud (solid blue line).

8. REFERENCES