Computational Performance of a Parallelized Three-Dimensional High-Order Spectral Element Toolbox

Christoph Bosshard[1], Roland Bouffanais[2], Christian Clémençon[3], Michel O. Deville[1], Nicolas Fiétier[1], Ralf Gruber[1], Sohrab Kehtari[4], Vincent Keller[1], and Jonas Latt[1]

[1] Laboratory of Computational Engineering, École Polytechnique Fédérale de Lausanne, STI – ISE – LIN, Station 9, CH–1015 Lausanne, Switzerland

[2] Massachusetts Institute of Technology, 77 Massachusetts Avenue, Room 5-326, Cambridge MA 02139, USA

[3] DIT, École Polytechnique Fédérale de Lausanne, CH–1015 Lausanne, Switzerland

[4] Swiss Federal Institute of Technology Zurich, HG J 47, Rämistrasse 101, 8092 Zurich, Switzerland

Abstract. In this paper, a comprehensive performance review of an MPI-based high-order three-dimensional spectral element method C++ toolbox is presented. The focus is put on the performance evaluation of several aspects with a particular emphasis on the parallel efficiency. The performance evaluation is analyzed with help of a time prediction model based on a parameterization of the application and the hardware resources. A tailor-made CFD computation benchmark case is introduced and used to carry out this review, stressing the particular interest for clusters with up to 8192 cores. Some problems in the parallel implementation have been detected and corrected. The theoretical complexities with respect to the number of elements, to the polynomial degree, and to communication needs are correctly reproduced. It is concluded that this type of code has a nearly perfect speed up on machines with thousands of cores, and is ready to make the step to next-generation petaflop machines.

1 Introduction

The **Spec**tral **u**nstructured E**l**ements **O**bject-**O**riented **S**ystem (SpecuLOOS) is a toolbox written in C++ [1]. SpecuLOOS is a spectral and mortar element analysis toolbox for the numerical solution of partial differential equations and more particularly for solving incompressible unsteady fluid flow problems [2]. The main architecture choices and the parallel implementation were elaborated and implemented by Van Kemenade and Dubois-Pèlerin [3]. Subsequently, SpecuLOOS' C++ code has been further developed, see [4].

It is well known that spectral element methods [5] are easily amenable to parallelization, as the domain decomposition into spectral elements can be made to correspond in a natural way to an attribution to parallel nodes [6].

The numerous references previously given and the ongoing simulations based on SpecuLOOS highlight the achieved versatility and flexibility of this C++ toolbox. Nevertheless, ten years have passed between the first version of SpecuLOOS' code and the present time and tremendous changes have occurred at both hardware and software levels. Fast dual DDR memory, RISC architectures, 64-bit memory addressing, compilers improvement, libraries optimization, libraries parallelization, and increase in inter-connecting switch performance are all among the SpecuLOOS improvements.

Here we discuss adaptation of SpecuLOOS to thousands of multi-core nodes. Performance measurements on a one-core node, on a cluster with hundreds of nodes, and on a 4096 dual-core BlueGene/L are presented. The obtained complexities are compared with theoretical predictions. First results show that small cases gave good complexities, but huge cases gave poor efficiencies. These results led to the detection of a poor parallel implementation. Once corrected, the complexity of SpecuLOOS corresponds to the theoretical one up to the 8192 cores used.

Test case description

The test case belongs to the field of CFD and consists in solving the 3D Navier–Stokes equations for a viscous Newtonian incompressible fluid. Based on the problem at hand, it is always physically rewarding to non-dimensionalize the governing Navier–Stokes equations which take the following general form

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} \mathbf{u} = -\boldsymbol{\nabla} p + \frac{1}{\mathrm{Re}} \boldsymbol{\Delta} \mathbf{u} + \mathbf{f}, \qquad \forall (\mathbf{x}, t) \in \Omega \times I, \tag{1}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0, \qquad \forall (\mathbf{x}, t) \in \Omega \times I, \tag{2}$$

where $\mathbf{u}$ is the velocity field, p the reduced pressure (normalized by the constant fluid density), $\mathbf{f}$ the body force per unit mass and Re the Reynolds number

$$\mathrm{Re} = \frac{UL}{\nu}, \tag{3}$$

expressed in terms of the characteristic length L, the characteristic velocity U, and the constant kinematic viscosity ν. The system evolution is studied in the time interval $I = [t_0, T]$. Considering particular flows, the governing Navier–Stokes equations (1)–(2) are supplemented with appropriate boundary conditions for the fluid velocity $\mathbf{u}$ and/or for the local stress at the boundary. For time-dependent problems, a given divergence-free velocity field is required as initial condition in the internal fluid domain.

The test case corresponds to the fully three-dimensional simulation of the flow enclosed in a lid-driven cubical cavity at the Reynolds number of 12 000 placing us in the locally-turbulent regime. It corresponds to the case denoted under-resolved DNS (UDNS) in Bouffanais *e.a.* [7]. The reader is referred to Bouffanais *e.a.* [7] for full details on the numerical method and on the parameters used.

The complexity is proportional to the total number of elements E in the three dimensional space. Each element is transformed to a cube. Since the Gauss–Lobatto–Legendre basis functions of degrees $N = N_x = N_y = N_z$

$$h_j(r) \;=\; -\frac{1}{N(N+1)}\frac{1}{L_N(\xi_j)}\frac{(1-r^2)\,L'_N(r)}{(r-\xi_j)}, \qquad -1 \le r \le +1, \quad 0 \le j \le N, \tag{4}$$

are orthonormal, the complexity for the pressure is $(N-1)^3$, while the complexity for the velocity is $(N+1)^3$. During the computations, the variables are frequently re-interpolated between the collocation, and operation which has a leading complexity of N^4, due to the tensorization of the implied linear operations. At large values of N, these re-interpolations therefore dominate the total computation time. It should be remarked that from a complexity standpoint, a term like $(N-1)^3$ is equivalent to a term like $(N+1)^3$. In the following, a term $N-1$ has been applied systematically to read the complexity from experimental performance curves, while the notation of the equations is simplified by use of the term N.

The CPU time T of the SpecuLOOS spectral code can then be estimated to

$$T(N_1, N_{CG}, E, N) = a_1 N_1 N_{CG} E N^{a_3}\,, \tag{5}$$

where N_{CG} is the number of conjugate gradient steps, N_1 is the number of time steps, a_1 is found through simulation, and $3 < a_3 < 4$.

2 Complexity on One Node

First, we run SpecuLOOS on one node without any communication using the Couzy preconditioner [8]. One time iteration step of SpecuLOOS is divided into three main parts: (1) computes the tentative velocity (through a Helmholtz equation solved by preconditioned conjugate gradient method), (2) computes the pressure (through a sophisticated conjugate gradient), and (3) corrects the tentative velocity. The relative importance of each of these three components depends on the parameters in a given simulation. With a fine discretization of the time axis, the second part becomes dominant, and takes as much as 90 % of the total CPU time in the example described below.

Table 1 presents the results of SpecuLOOS on one node of an Intel Xeon cluster. The CPU time measurements, $T_{CG,(1\ iter)}$, for one time step and one CG iteration step ($N_1 = N_{CG} = 1$) are used to minimize $\sum (T - T_{CG,(1\ iter)})^2$, where

$$T(1, 1, E, N) = a_1 E^{a_2} N^{a_3}\,, \tag{6}$$

Table 1. SpecuLOOS on one node of an Intel Xeon cluster. The number of conjugate gradient iterations N_{CG} is an average value over all time steps for the pressure.

N_1	E	N	T_{exec} $[s]$	N_{CG} # iter	T_{CG} $[s]$	$\frac{T_{CG}}{T_{exec}}$	$T_{CG,(1\ iter)}$ $[s]$
1	256	8	40.1	198	32.8	0.818	0.17
1	256	10	119.3	247	103.8	0.870	0.42
1	512	6	43.2	205	33.9	0.785	0.17
1	512	8	116.4	268	106.7	0.917	0.40
1	512	10	394.3	344	342.3	0.868	1.00
1	1024	6	105.2	259	83.4	0.793	0.32
1	1024	8	311.0	339	265.4	0.853	0.78

This minimization procedure gives the scaling law.

$$T(1,1,E,N) = 2.01 \cdot 10^{-6} E^{0.97} \cdot N^{3.3} \,. \tag{7}$$

One realises that this complexity law corresponds well to the theoretical one, (Eq. 5).

Generally, the number of iteration steps is not known. If in the optimization procedure one includes N_{CG} in the parameters E and N,

$$T(N_1,\cdot,E,N) = 1.15 \cdot 10^{-5} \cdot N_1 \cdot E^{1.30} \cdot N^{4.19} \,. \tag{8}$$

As a consequence, the estimated number of $N_{CG,est}$ is

$$N_{CG,est} = 5.72 \cdot E^{0.33} \cdot N^{0.89} \,. \tag{9}$$

This prediction is also close to the expected theoretical complexity $N_{CG,theo}$:

$$N_{CG,theo} \approx E^{\frac{1}{3}} \cdot N \,. \tag{10}$$

The same type of studies have been made for a diagonal precoditioner varying the polynomial degree. The complexity found is $N_{CG,est} \approx N^{1.47}$. Thus, for $N \geq 12$ the diagonal preconditioner is faster, while for $N < 12$ the Couzy preconditioner is faster. Since we treat cases with $N = 12$ or smaller, we concentrate on the Couzy preconditioner.

3 Wrong Complexity on the BlueGene/L

The SpecuLOOS code has been ported to the IBM BlueGene/L machine at EPFL with 4096 dual core nodes. Since interelement communication is not that important, all the cores can be used for computation. Table 2 presents the results obtained with the original version of the SpecuLOOS code, before important adaptations described below where performed. One element is running in a core. The polynomial degree is fixed to N=12 for the velocity components, and

Table 2. SpecuLOOS on the BlueGene/L machine up to $P = 8192$ cores. The number of elements per core have been fixed to one. The polynomial degree for the pressure is equal to $N - 2 = 10$.

N_1	$E = E_x E_y E_z$	N	P	$\frac{\#\ elem}{node}$	T_{exec}
1	8x8x16	12	1024	1	17.22
1	8x16x16	12	2048	1	29.91
1	16x16x16	12	4096	1	57.05
1	16x16x32	12	8192	1	140.50

to $N - 2$=10 for the pressure. The resulting complexity given by the pressure computation is illustrated on Table 2, and is identified as

$$T \approx E^2. \tag{11}$$

This result shows that the complexity of the original parallel code is far away from the $E^{1.3}$ law, which is expected for theoretical reasons and verified numerically in a serial program execution, eq. 8. The reasons for this bad result could be identified as follows. An IF instruction over all elements had been introduced in the code, in order to identify the attribution of elements to computational nodes dynamically, at each iteration of the conjugate gradient method. Such an instruction is typical for rapid corrections in a code, which are made to parallelize a program rapidly without realising its impact on future program executions. This did not affect the CPU time for less than $P = 100$ nodes, but became dominant for $P > 1000$. This IF instruction has now been replaced by the use of a precomputed list, pointing to the elements which are active on a core.

4 Fine Results on the BlueGene/L

The corrected SpecuLOOS code has been rerun again on the BlueGene/L machine. The effect of the communication has been studied, and results are presented in Fig. 1. The ideal speedup is presented in the upper straight line. The measurements for the case of one element per core are presented through the circular markers. Inter-element communication takes a very small amount of time in the case of 1024 elements, and about 12% of the total time for 8192 cores. If the number of elements per core is increased, the communication/computation decreases, and the computation is closer to ideal scalability.

The measurements in Table 3 show that the symmetric cases with E=16x16x16 take less iteration steps than non-symmetric cases. The complexity laws $T \approx E^a$ for the unsymmetric cases give exponents of a_2=(1.35, 1.28, 1.36, 1.34) for $P =$ (1024, 2048, 4096, 8192). These exponents correspond very well to the expected one, i.e. to a_2=1.3, eqs. 6 and 8. This tells us that the present version of the SpecuLOOS code is well scalable on the BlueGene/L up to 8192, and will for bigger cases, scale to petaflop machines.

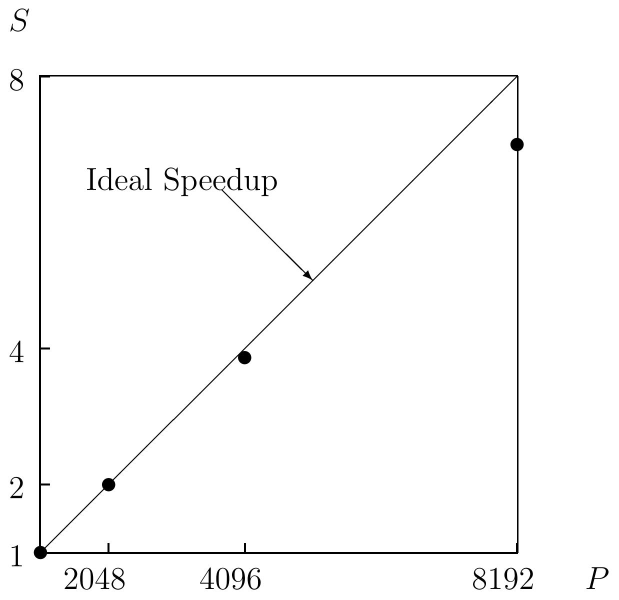

Fig. 1. Speedup S on BlueGene/L as a function of P for the case E=16x16x32, Table 3. The 12% loss in speedup for P=8192 is due to communication between the cores.

Table 3. SpecuLOOS on the BlueGene/L machine up to $P = 8192$ cores. The polynomial degree for the pressure is equal to $N - 2 = 10$. The number of elements per core vary from 1 to 8. The number of iteration steps is an average value, with a clear drop for $E_x = E_y = E_z$.

P	$E = E_x E_y E_z$	$\frac{E^3}{P}$	N_{it}	$T[sec]$
1024	8x8x16	1	689	29.9
1024	8x16x16	2	966	82.8
1024	16x16x16	4	912	155.3
1024	16x16x32	8	1463	494.2
2048	8x16x16	1	950	42.1
2048	16x16x16	2	912	79.1
2048	16x16x32	4	1461	249.9
4096	16x16x16	1	912	41.9
4096	16x16x32	2	1463	128.4
4096	16x32x32	4	1925	331.2
8192	16x16x32	1	1463	70.7
8192	16x32x32	2	1958	179.1

5 Conclusions

The performance review presented in this paper for the high-order spectral and mortar element method C++ toolbox, Speculoos, has shown that good performances can be achieved with relatively common internode network communication systems, available software and hardware resources—small commodity clusters with non-proprietary compilers installed on it.

The parallel implementation of Speculoos based on MPI has shown to be efficient. Reasonable scalability and efficiency can be achieved on commodity clusters. The results support the original choices made in Speculoos parallel implementation by keeping it at a very low-level.

One of the goal of this study was to estimate if Speculoos could run on a massively parallel computer architecture comprising thousands of computational units, specifically on the IBM Blue Gene machine at EPFL with 4'096 dual core units. After detection and correction of a poor implementation choice in the parallel version, perfect scalabilities on up to 8192 cores have been detected.

The present version of the SpecuLOOS code is well scalable on the BlueGene/L up to 8192, and will for bigger cases, even scale to petaflop machines.

Acknowledgments

This research is being partially funded by a Swiss National Science Foundation Grant (No. 200020–101707) and by the Swiss National Supercomputing Center CSCS, whose supports are gratefully acknowledged.

References

1. The OpenSPECULOOS project, `http://sourceforge.net/projects/openspeculoos/`
2. Van Kemenade, V.: Incompressible fluid flow simulation by the spectral element method, Tech. rep., "Annexe technique projet FN 21-40'512.94", IMHEF–DGM, Swiss Federal Institute of Technology, Lausanne (1996)
3. Dubois-Pèlerin, Y., Van Kemenade, V., Deville, M.: An object-oriented toolbox for spectral element analysis. J. Sci. Comput. 14, 1–29 (1999)
4. Bouffanais, R.: Simulation of shear-driven flows: Transition with a free surface and confined turbulence, EPFL, Thèse no. 3837 (2007)
5. Deville, M.O., Fischer, P.F., Mund, E.H.: High-order methods for incompressible fluid flow. Cambridge University Press, Cambridge (2002)
6. Fischer, P.F., Patera, A.T.: Parallel spectral element solution of the Stokes problem. J. Comput. Phys. 92, 380–421 (1991)
7. Bouffanais, R., Deville, M.O., Leriche, E.: Large-eddy simulation of the flow in a lid-driven cubical cavity. Phys. Fluids 19, Art. 055108 (2007)
8. Couzy, W., Deville, M.O.: Spectral-element preconditioners for the Uzawa pressure operator applied to incompressible flows. J. Sci. Comput. 9, 107–112 (1994)