Parallel Processing Letters
Vol. 28, No. 1 (2018) 1850003 (12 pages)

DOI: 10.1142/S0129626418500032

World Scientific
www.worldscientific.com

Fast Approximate Evaluation of Parallel Overhead from a Minimal Set of Measured Execution Times

Siegfried Höfinger*
VSC Research Center, TUit, Vienna University of Technology
Wiedner Hauptstraße 8-10, A-1040 Vienna, Austria
Department of Physics, Michigan Technological University
1400 Townsend Drive, 49931 Houghton, MI, USA
siegfried.hoefinger@tuwien.ac.at; shoefing@mtu.edu

Thomas Ruh
Institute for Materials Chemistry, Vienna University of Technology
Getreidemarkt 9/165, A-1060 Vienna, Austria

Ernst Haunschmid
VSC Research Center, TUit, Vienna University of Technology
Wiedner Hauptstraße 8-10, A-1040 Vienna, Austria

Received 30 December 2017
Accepted 6 February 2018
Published 29 March 2018

ABSTRACT

Porting scientific key algorithms to HPC architectures requires a thorough understanding of the subtle balance between gain in performance and introduced overhead. Here we continue the development of our recently proposed technique that uses plain execution times to predict the extent of parallel overhead. The focus here is on an analytic solution that takes into account as many data points as there are unknowns, i.e. model parameters. A test set of 9 applications frequently used in scientific computing can be well described by the suggested model even including atypical cases that were originally not considered part of the development. However, the choice about which particular set of explicit data points will lead to an optimal prediction cannot be made a-priori.

Keywords: Performance evaluation; parallel processing; HPC; parallel overhead; MPI; a-posteriori assessment.

*Corresponding author.

1. Introduction

High Performance Computing (HPC) is increasingly seen as an integral part of modern scientific research and many examples have been given to date [1–5] reaching from theoretical predictions of exotic states of matter [1, 5] over analyzing CERN data in the discovery of the Higgs boson [2] or genetic data in deciphering the human genome [4] to worldwide collaborations on the detection of gravitational waves [3]. In many instances a single key algorithm forms the basis of all in-depth scientific discovery and, consequently, needs to be ported to HPC architectures for efficient computation of large scale problems and detailed analysis of complex data sets. However, such a transformation of scientific code to an efficient HPC implementation is all but a trivial task. This is due in part to the complex nature of current state-of-the-art HPC systems, e.g. the dynamic behaviour of core components [6], the increasing complexity of CPU architectures [7] and the need for scalable performance using standard methods of parallel programming [8]. Moreover, significant limitations arise from the required communication between individual parallel tasks, where the introduced overhead needs to be kept at an absolute minimum in order to achieve sustained scalability to large numbers of parallel processing cores [9, 10]. Especially with respect to this latter constraint, several quality tools have been developed in the past that aid in quantifying parallelization overhead and thus ease the process of porting scientific programs [11–13].

Studying parallel overhead has been a research focus for many years [14–22]. One early proposal was given with the logP model [14, 15] where four parameters, latency, overhead, gap (reciprocal communication bandwidth) and number of processors were introduced to analyze a given algorithm. A core design goal was to find a balance between overly simplistic and overly specific models. Application to MPI [16] and several extensions respecting large messages [17, 16] and contention effects [18] were described. A more abstract framework with tunable complexity but still practical timing requirements has been provided with PERC [19]. More recent trends in hybrid MPI/OpenMP programming were taken care of by a combination of application signature with system profiles [20]. Along similar lines application-centric performance modeling [21] was described based on characteristics of the application and the target computing platform with the objective of successful large-scale extrapolation. Similar predictions could be made with the help of runtime functions within the SUIF infrastructure [22].

For many practical applications, prior to any more sophisticated analysis, it is already considered helpful preliminary information if one could get a quick overview on the subject with simplest methods, i.e. without having to switch on profiling flags, linking in additional libraries, embedding in analysis tools etc. Such a method need not be perfect [7], it merely should serve for a qualitative assessment to monitor critical situations where the consultation of more advanced tools [11–13] is becoming a strongly advisable recommendation. We have previously presented such a method [23] where a simple record of execution times for varying numbers of

parallel processing cores could be used to estimate the parallel overhead incurred. This approach was specifically applicable to a particular class of scientific applications frequently used on current HPC platforms. Here we want to briefly summarize the basics of this approach, augment it with two more examples from the scientific community and describe a systematic way to determine model-critical parameters.

2. Basic model

The time to solution, t_n, for a particular application executed on some HPC platform using n cores was proposed as [23],

$$t_n = t_n^{Amdahl}\left[1 + \frac{b(n-1)}{(1+c-b)n + (b+c+c^2)}\right] \tag{1}$$

where b and c are application- as well as problem-size-specific parameters and the initial term is the classic relation of Amdahl [24],

$$t_n^{Amdahl} = f_s t_1 + \frac{(1-f_s)t_1}{n} \tag{2}$$

with f_s the sequential fraction of t_1 that cannot be parallelized. Parameters b and c could be determined from a fit to measured execution times with f_s obtained as a by-product. Once the fit was showing a good match to the data set, f_s, b and c could be plugged into a second relation accounting for τ_n, the parallel overhead affecting the application,

$$\tau_n = \frac{t_n^{Amdahl} b(n-1)}{(1+c-b)n + (b+c+c^2)} \tag{3}$$

Owing to its simplicity, Eq. (1) can also be used in reverse to determine model parameters, c, f_s and b from a triple of known explicit data points, $\{(n_x, t_x), (n_y, t_y), (n_z, t_z)\}$. Since the number of possible triples is growing rapidly with the number of available data points, it is interesting to examine which of the available combinations would lead to an optimal representation. The hope is to replace the fitting procedure used in [23] with an analytic approach based on a suitable choice of three explicit execution times. Given the already established data sets considered in [23], these data can now be re-evaluated and general trends unveiled as well as potential numerical issues identified.

2.1. *Determination of model parameters c, f_s and b*

Respecting the non-linear character of Eq. (1) after a series of trivial algebraic manipulations we obtain the following 4th-order polynomial in c,

$$\begin{aligned} 0 = {} & (\gamma_1^{xz}\delta_1^{xy} - \gamma_1^{xy}\delta_1^{xz})\, c^4 + (\gamma_2^{xz}\delta_1^{xy} + \gamma_1^{xz}\delta_2^{xy} - \gamma_2^{xy}\delta_1^{xz} - \gamma_1^{xy}\delta_2^{xz})\, c^3 \\ & + (\gamma_3^{xz}\delta_1^{xy} + \gamma_2^{xz}\delta_2^{xy} + \gamma_1^{xz}\delta_3^{xy} - \gamma_3^{xy}\delta_1^{xz} - \gamma_2^{xy}\delta_2^{xz} - \gamma_1^{xy}\delta_3^{xz})\, c^2 \\ & + (\gamma_3^{xz}\delta_2^{xy} + \gamma_2^{xz}\delta_3^{xy} - \gamma_3^{xy}\delta_2^{xz} - \gamma_2^{xy}\delta_3^{xz})\, c + (\gamma_3^{xz}\delta_3^{xy} - \gamma_3^{xy}\delta_3^{xz}) \end{aligned} \tag{4}$$

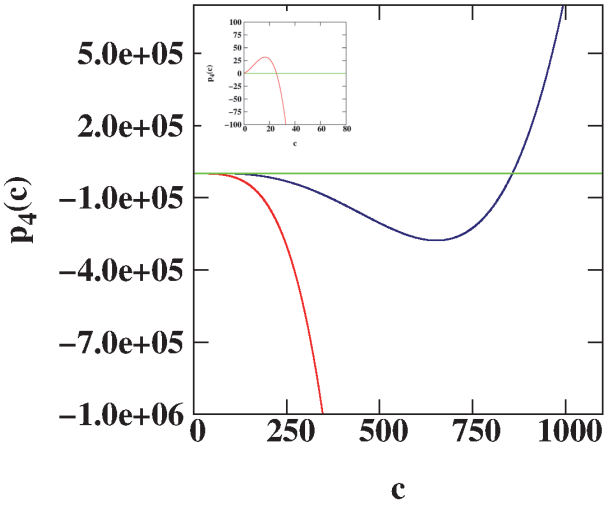

Fig. 1. (color online) Characteristic samples of the 4th-order polynomial in c given in Eq. (4) where all coefficients are scaled by $(\gamma_3^{xz}\delta_2^{xy} + \gamma_2^{xz}\delta_3^{xy} - \gamma_3^{xy}\delta_2^{xz} - \gamma_2^{xy}\delta_3^{xz})^{-1}$. Different choices of explicit triplets of execution times give rise to $p_4(c)$ of either strongly increasing (blue) or decreasing (red) trend with at least one particular zero always detectable in the interval $[10, 1000]$.

with individual coefficients defined as follows,

$$
\begin{aligned}
\gamma_1^{xy} &= t_1 t_y n_y - t_1 t_x n_x + t_x t_y n_x {n_y}^2 - t_x t_y {n_x}^2 n_y \\
&\quad - t_1 t_y {n_y}^2 + t_1 t_x {n_x}^2 + 2 t_1 t_x n_x n_y - 2 t_1 t_y n_x n_y - 2 {t_1}^2 n_y + 2 {t_1}^2 n_x \\
\delta_1^{xy} &= -t_1 t_y n_y + t_1 t_x n_x + t_1 t_y n_x n_y - t_1 t_x n_x n_y \\
&\quad + t_1 t_y {n_y}^2 - t_1 t_x {n_x}^2 - t_1 t_y n_x {n_y}^2 + t_1 t_x {n_x}^2 n_y + 2 {t_1}^2 n_y - 2 {t_1}^2 n_x \\
\gamma_2^{xy} &= -2 t_x t_y {n_x}^2 n_y + 2 t_x t_y n_x {n_y}^2 - t_1 t_y n_x n_y \\
&\quad + t_1 t_x n_x n_y + t_1 t_y n_y - t_1 t_x n_x - 3 t_1 t_y n_x {n_y}^2 \\
&\quad + 3 t_1 t_x {n_x}^2 n_y - t_1 t_y {n_y}^2 + t_1 t_x {n_x}^2 - 2 {t_1}^2 n_y + 2 {t_1}^2 n_x \\
\delta_2^{xy} &= -t_1 t_y n_y + t_1 t_x n_x + t_1 t_y {n_x}^2 n_y - t_1 t_x n_x {n_y}^2 \\
&\quad + t_1 t_y {n_y}^2 - t_1 t_x {n_x}^2 - t_1 t_y {n_x}^2 {n_y}^2 + t_1 t_x {n_x}^2 {n_y}^2 \\
&\quad + 2 {t_1}^2 n_y - 2 {t_1}^2 n_x - 2 {t_1}^2 {n_x}^2 n_y + 2 {t_1}^2 n_x {n_y}^2 \\
\gamma_3^{xy} &= -t_x t_y {n_x}^2 n_y + t_x t_y n_x {n_y}^2 + t_1 t_y n_x n_y - t_1 t_x n_x n_y \\
&\quad - 3 t_1 t_y n_x {n_y}^2 + 3 t_1 t_x {n_x}^2 n_y \\
\delta_3^{xy} &= -t_1 t_y n_x n_y + t_1 t_x n_x n_y + t_1 t_y {n_x}^2 n_y - t_1 t_x n_x {n_y}^2 \\
&\quad + t_1 t_y n_x {n_y}^2 - t_1 t_x {n_x}^2 n_y - t_1 t_y {n_x}^2 {n_y}^2 + t_1 t_x {n_x}^2 {n_y}^2 \\
&\quad - 2 {t_1}^2 {n_x}^2 n_y + 2 {t_1}^2 n_x {n_y}^2
\end{aligned}
\tag{5}
$$

and characteristic shapes of the polynomial shown in Fig. 1.

Because individual coefficients in the polynomial of Eq. (4) are rather large, we usually divide the entire equation by the coefficient corresponding to the 1st-order term, i.e. the prefactor of c. In so doing we expect to identify at least one particular zero in the interval $[10, 1000]$ (see Fig. 1) with $c \in \mathbf{R}^+$ and straightforward detection by means of bisection [25]. Of particular note is the bivalent character of the curvature of either rapidly increasing, or decreasing trend (see blue and red graphs in Fig. 1). Having obtained a solution for c, the next model parameter can be determined using

$$f_s = -\frac{\gamma_1^{xy} c^2 + \gamma_2^{xy} c + \gamma_3^{xy}}{\delta_1^{xy} c^2 + \delta_2^{xy} c + \delta_3^{xy}} \tag{6}$$

where it should be noted that equivalent expressions in terms of pairs xz or yz should lead to identical solutions. Finally, the third parameter can be derived from the following relation,

$$\begin{aligned} b = -\frac{1}{t_x n_x - t_x {n_x}^2 - 2 t_1 n_x} \big[& \left(t_x n_x - t_1 + f_s t_1 - f_s t_1 n_x \right) c^2 \\ & + \left(t_x {n_x}^2 + t_x n_x - t_1 n_x - t_1 + f_s t_1 - f_s t_1 {n_x}^2 \right) c \\ & + \left(t_x {n_x}^2 - t_1 n_x + f_s t_1 n_x - f_s t_1 {n_x}^2 \right) \big] \end{aligned} \tag{7}$$

where, again, similar expressions hold for n_y, t_y and n_z, t_z.

3. HPC platforms

Study systems were run on the *Vienna Scientific Cluster*, version 3 (VSC-3) [26]. VSC-3 consists of 2020 compute nodes, all of them equipped with dual socket 8 core Intel Xeon CPUs (E5-2650v2, 2.6 GHz, Ivy Bridge) and interconnected by a dual-rail Infiniband QDR-80 network. Standard node memory is 64 GB, optionally available are nodes with 128 GB or 256 GB. The system features a rather unconventional cooling infrastructure, i.e. *Liquid Immersion Cooling* [27] where hardware components are fully immersed in mineral oil.

4. Results

Extending our previous study [23] we first present data for two more applications frequently used in science and research. These are WIEN2k [28] and NWCHEM [29]. Times to solution, t_n, as a function of numbers of processing cores, n, are shown in Fig. 2 (red squares and green discs, also see Table 1). As can be seen, WIEN2k belongs exactly to the same class of applications that was subject to our previous study where the focus was on compute-bound applications of significant arithmetic intensity. In contrast, NWCHEM exhibits a remarkably different profile and shows a strong rise in t_n even for small numbers of processing cores, where t_n starts to exceed the single-core execution time, t_1, at core counts of $n > 250$. The reason for such a drastic qualitative change may lie in the fact that NWCHEM implements

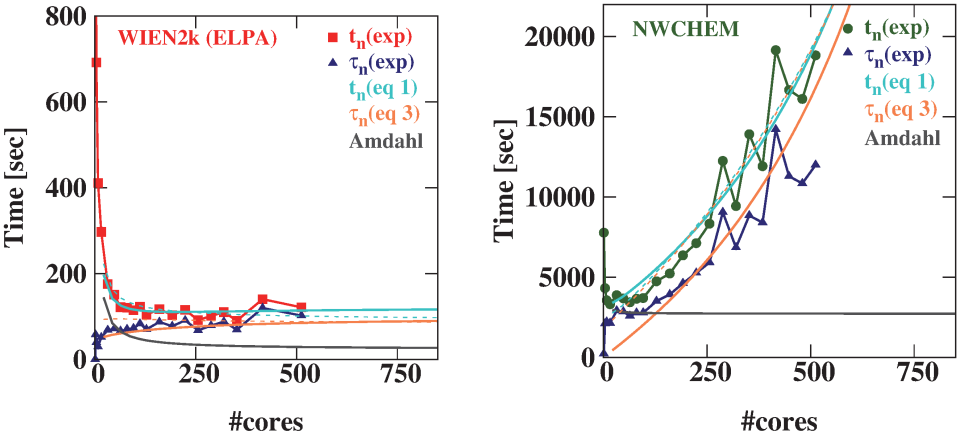

Fig. 2. (color online) Recorded times to solution, t_n, (red squares and green discs, also see Table 1, columns 1, 2 and 5) as a function of numbers of cores, n, operating in parallel for applications WIEN2k [28] and NWCHEM [29]. Very large initial times corresponding to very small core counts have been truncated for better visual clarity. Best fitting the data with GNUPLOT [35] yields parameters b and c (implicitly also f_s) via Eq. (1) where the original data is reasonably well approximated (solid line in cyan). In addition, an estimate can be provided for the parallel overhead using Eq. (3) (orange line). The estimate matches the *allinea/MAP*-derived [13] mean parallel overhead rather well (compare orange line to triangles in blue, respectively Table 1, columns 3 and 6). Significant deviation from Amdahl's Law [24] is seen already for small core counts (compare cyan to grey line). Optional analytic determination of c, f_s and b (as discussed here) from a triple of known data points are shown as dashed lines.

a PGAS [30–33] model, in particular the Global Arrays Toolkit [34]. It is surprising that the suggested relation of Eq. (1) appears to be working even for such applications that never had been considered target of the original development.

4.1. *Overall assessment of the explicit computation of model parameters c, f_s and b*

Previous [23] and current measurements of t_n as a function of n are considered for analytical computations of c, f_s and b following Eqs. (4), (6) and (7). From a set of k data points all possible triple combinations $\{(n_x, t_x), (n_y, t_y), (n_z, t_z)\}$ are examined. There are a number of $\binom{k}{3}$ such triples where it should be noted that the single core execution time, t_1, is not a member of the k-group but needs to be provided in addition to it. Initially, the root-finding problem described in Sec. 2.1 should deliver a single solution for c. This is then used in Eq. (6) to determine f_s. Two pairs of data points with largest differences in core counts are taken into account for this second task and the smaller f_s value is carried on if either solution is perceived meaningful, i.e. $f_s > 0$. The final task then is to use these two parameters, c and f_s, together with Eq. (7) to determine b. All three possibilities are considered and a mean value of b is calculated only in case all individual solutions turn out to be reasonable, i.e. $b > 0$. The general numerical behaviour of

Table 1. Exe-times, t_n, and MPI overhead, τ_n^{MPI} (determined with *allinea/MAP* [13]) for applications WIEN2k [28] and NWCHEM [29].

	WIEN2k			NWCHEM		
n	t_n [sec]	τ_n^{MPI} [sec]	$\frac{\tau_n^{MPI}}{t_n}$	t_n [sec]	τ_n^{MPI} [sec]	$\frac{\tau_n^{MPI}}{t_n}$
1	2652.6	0.0	0.000	7780.3	225.6	0.029
2	1250.4	57.5	0.046	–	–	–
4	691.6	39.4	0.057	4322.1	2104.9	0.487
8	411.2	29.2	0.071	3559.2	2213.8	0.622
16	297.0	51.4	0.173	3319.2	2137.6	0.644
32	175.9	68.4	0.389	3877.4	2908.1	0.750
48	150.6	71.4	0.474	3680.7	2845.2	0.773
64	121.7	66.3	0.545	3438.5	2582.3	0.751
80	120.3	70.5	0.586	3660.4	2770.9	0.757
96	114.9	72.4	0.630	3705.0	2782.5	0.751
112	123.3	83.2	0.675	–	–	–
128	104.2	70.2	0.674	4747.0	3508.0	0.739
160	117.3	87.4	0.745	5230.7	3923.0	0.750
192	103.4	77.6	0.750	6363.1	4619.6	0.726
224	115.8	90.9	0.785	7122.5	5263.5	0.739
256	91.4	67.7	0.741	8335.0	5917.9	0.710
288	101.2	78.9	0.780	12245.9	9049.7	0.739
320	110.3	88.0	0.798	9434.8	6849.7	0.726
352	90.2	69.3	0.768	13913.5	8849.0	0.636
416	140.4	119.9	0.854	19151.7	14210.6	0.742
448	–	–	–	16676.2	11289.8	0.677
480	–	–	–	16114.1	10844.8	0.673
512	121.4	102.2	0.842	18831.3	11995.5	0.637

Table 2. Evaluation of explicit calculations of parameters c, f_s and b via Eqs. (4), (6) and (7) for applications WIEN2k [28], NWCHEM [29], HPL [36], GROMACS [37], AMBER [38],GREMLIN [39, 40], VASP [41], QUANTUM ESPRESSO [42] and LAMMPS [43].

Application	#Triples Considered	#Misses Comp(c)	#Misses Comp(f_s)	#Misses Comp(b)
WIEN2k	1140	238	781	77
NWCHEM	1330	916	0	2
HPL	1140	768	251	11
GROMACS	2024	221	1642	105
AMBER	1771	354	1274	117
GREMLIN	35	13	20	2
VASP	816	147	601	64
QESPRESSO	1540	281	1148	72
LAMMPS	1540	610	809	15

this approach is summarized in Table 2. As becomes clear immediately, the major problem is finding a suitable solution for f_s. This is partially due to the anticipated value of f_s close to zero for strong-scaling applications which for many triples will result in a pair of f_s-solutions slightly larger and smaller than zero, hence will be

discarded following the aforementioned procedure. The interesting outlier in this respect is NWCHEM which underlines the observed characteristics of non-ideality in the strong-scaling regime with an anticipated sizeable fraction of sequential code. Given the autonomous nature and efficiency of the present approach, an analytical determination of such critical parameters as f_s seems to be of considerable advantage.

4.2. *Optimal choice of three data points*

For each of the successful computations of parameters c, f_s and b (see previous Sec. 4.1) we can express the quality of the approximation in terms of the root mean square deviation (RMSD) from the measured execution times,

$$RMSD = \sqrt{\frac{1}{k}\sum_{n}^{k}\left[t_n(exp) - t_n(app)\right]^2} \tag{8}$$

where $t_n(exp)$ are the k experimentally observed data points and $t_n(app)$ their corresponding approximations resulting from Eq. (1). By ranking all the solutions according to RMSD we can screen for the most optimal triplet of explicit data points leading to the best match to the observed data. Moreover, different applications can be compared to each other and their respective optimal combinations of explicit data points can be analyzed for an emerging pattern perhaps common to all applications. Top-2-ranked solutions together with their underlying triplet of numbers of cores are summarized in Table 3. Top-results corresponding to WIEN2k and NWCHEM are also indicated as dashed lines in Fig. 2 and can be compared

Table 3. Top ranked solutions for c, f_s and b using Eqs. (4), (6) and (7) and corresponding selection of three explicit data points resulting in minimal RMSD from experimentally observed times to solutions, t_n, for a range of applications (see Table 2).

Application	n_x	n_y	n_z	f_s	b	c
WIEN2k	48	192	320	0.0029	22.435	23.530
	48	192	224	0.0041	21.630	23.111
NWCHEM	32	128	320	0.0103	3.952	2.943
	32	128	512	0.0075	3.890	2.876
HPL	384	768	1296	0.0001	92.001	155.776
	384	640	896	0.0007	82.083	154.994
GROMACS	48	112	688	0.0011	42.627	56.077
	48	112	320	0.0017	40.825	55.129
AMBER	80	320	352	0.0021	43.217	45.096
	40	256	352	0.0019	38.099	40.440
GREMLIN	–	–	–	–	–	–
VASP	80	192	512	0.0009	48.679	54.238
	32	112	768	0.0016	28.513	32.549
QESPRESSO	160	416	768	0.0018	78.453	77.214
	128	160	1024	0.0033	86.608	85.555
LAMMPS	320	512	1024	0.0037	28.132	27.439
	320	512	768	0.0030	27.932	27.117

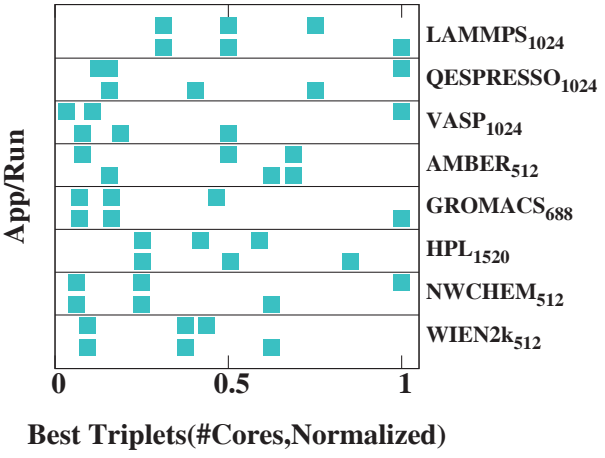

Fig. 3. Triplets of explicit data points leading to optimal solutions for c, f_s and b. Maximum numbers of cores considered are indicated as subscripts to the name of the application on the right axis. Various ranges have been normalized into the interval $[0, 1]$ for better direct comparison with 1 representing the maximum core count.

to the plain GNUPLOT [35] based fitting. As has already been outlined in [23] because of $\lim_{n\to\infty} \frac{\tau_n}{t_n} < 1$ we infer that $b < c + 1$, hence a condition that almost all of the top-2-ranked results do indeed reflect (see final two columns in Table 3). It is interesting to observe that applications similar to NWCHEM with increasing trend in t_n deliver solutions very close to the limit (see for example b, c values in Table 3 for applications QUANTUM ESPRESSO and LAMMPS). Thus the rather strong deviation of τ_n (see dashed orange line in the right panel of Fig. 2) may indicate a certain degree of systematic bias for this type of applications. No solution could be determined for application GREMLIN [39, 40] because of the limited number of available data points (see Table 2).

In general, it is difficult to advise on an optimal choice of three explicit data points. Results of Table 3 are also graphically illustrated in Fig. 3 where individual ranges of considered numbers of cores have been normalized into the interval $[0, 1]$ and the largest value of n (corresponding to normalized 1) is given as subscript to the name of the application. Not only do optimal core counts scatter over the entire range, but also are second and top ranked solutions pretty different to each other for individual applications. Thus it appears that compilation of a small data set similar to the example given in Table 1 with subsequent combinatorial examination of all possible triplets is probably the most general approach to determine optimal solutions for c, f_s and b.

5. Conclusion

Parallel execution times, t_n, can be well described by Eq. (1) for a great variety of different applications. This even includes atypical cases as shown here on the

example of NWCHEM [29] that exhibit deterioration of execution times with increasing numbers of parallel processing cores. Model parameters c, f_s and b can be efficiently computed from a set of 3 explicit data points with f_s, the fraction of sequential code, being the most difficult to determine. Knowledge of these parameters facilitates an approximate estimation of τ_n, the parallel overhead affecting the application (see Eq. (3)). No general recommendation can be made for an a-priori choice of explicit data points most appropriate for an optimal determination of these parameters.

Acknowledgments

This study is based on work supported by the VSC Research Center funded by the Austrian Federal Ministry of Science, Research and Economy (bmwfw). The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

References

[1] M. Seager, Metallic Liquid Hydrogen, https://str.llnl.gov/str/JanFeb05/Seager.html, 2005.
[2] T. Whyntie and J. Coles, Number-crunching Higgs boson: Meet the world's largest distributed computer grid, http://theconversation.com/number-crunching-higgs-boson-meet-the-worlds-largest-distributed-computer-grid-38696, 2015.
[3] E. Seidel, Supercomputing Gravitational Waves at NCSA, http://insidehpc.com/2016/06/second-recorded-gravitational-wave-event-analysed-using-illinois-supercomputer, 2016.
[4] J. Craig Venter *et al.*, The Sequence of the Human Genome, http://science.sciencemag.org/content/291/5507/1304, 2016.
[5] P. Vranas, T. Luu and R. Soltz, Quark Theory and Today's Supercomputers: It's a Match, https://str.llnl.gov/str/JanFeb08/soltz.html, 2008.
[6] S. Perarnau, R. Thakur, K. Iskra, K. Raffenetti, F. Cappello, R. Gupta, P. H. Beckman, M. Snir, H. Hoffmann, M. Schulz and B. Rountree, *Distributed Monitoring and Management of Exascale Systems in the Argo Project*, Volume 9038 (Springer, 2015), pages 173–178.
[7] S. Williams, A. Waterman and D. Patterson, Roofline: An insightful visual performance model for multicore architectures, *Commun. ACM* **52**(4):65–76, 2009.
[8] *MPI*: *A Message-Passing Interface Standard Version 3.1*, High Performance Computing Center Stuttgart (HLRS), Stuttgart, 2015.
[9] D. J. Appelhans, T. Manteuffel, S. McCormick and J. Ruge, A low-communication, parallel algorithm for solving PDEs based on range decomposition, *Numer. Linear Algebr.* 2016.
[10] A. Grama, A. Gupta, G. Karypis and V. Kumar, *Introduction to Parallel Computing* (Addison Wesley, 2003), 2nd edition.
[11] D. Böhme, M.-A. Hermanns and F. Wolf, Scalasca, in *Entwicklung und Evolution von Forschungssoftware*, Rolduc, November 2011, Volume 14 of Aachener Informatik-Berichte, Software Engineering, Shaker, 2012, pages 43–48.
[12] J. Vetter and C. Chambreau, mpiP: Lightweight, Scalable MPI Profiling, version 3.4.1, http://mpip.sourceforge.net, 2014.

[13] Allinea Software Ltd., MAP, release 6.0.6, http://www.allinea.com/products/map, 2016.

[14] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E.Schauser, E. Santos, R. Subramonian and T. von Eicken, LogP: Towards a realistic model of parallel computation, in *PPOPP '93 Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ACM, 1993, pages 1–12.

[15] T. Kielmann, H. E. Bal and K. Verstoep, Fast measurement of LogP parameters for message passing platforms, in *IPDPS '00 Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, Springer, UK, 2000, pages 1176–1183 .

[16] K. Al-Tawil and C. A. Moritz, Performance modeling and evaluation of MPI, *J. Parallel Distrib. Comput.* **61**:202–223, 2001.

[17] A. Alexandrov, M. F. Ionescu, K. E. Schauser and C. Scheiman, LogGP: Incorporating long messages into the LogP model — One step closer towards a realistic model for parallel computation, in *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, ACM, 1995, pages 95–105.

[18] C. A. Moritz and M. I. Frank, LogGPC: Modeling network contention in message-passing programs, *IEEE Trans. Parallel Distrib. Syst.* **12**(4):404–415, 2001.

[19] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia and A. Purkayastha, A framework for performance modeling and prediction, in *SC '02*: *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, IEEE, 2002.

[20] L. Adhianto and B. Chapman, Performance modeling of communication and computation in hybrid MPI and OpenMP applications, *Simul. Model. Pract. Theory* **15**(4):481–491, 2007.

[21] K. J. Barker, K. A. Hoisi, D. J. Kerbyson, M. Lang, S. Pakin and J. C. Sancho, Using Performance modeling to design large-scale systems, *Computer* **42**(11):42–49, 2009.

[22] M. Kühnemann, T. Rauber and G. Rünger, A source code analyzer for performance prediction, in *18th International Parallel and Distributed Processing Symposium 2004*, IEEE, 2004.

[23] S. Höfinger and E. Haunschmid, Modelling parallel overhead from simple run-time records, *J. Supercomput.* 2017.

[24] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in *Proceedings AFIPS '67* (*Spring*) *Joint Computer Conference*, ACM, 1967, pages 483–485.

[25] P. L. DeVries, *A First Course in Computational Physics* (John Wiley & Sons, 1994).

[26] Vienna Scientific Cluster, generation 3, http://vsc.ac.at/systems/vsc-3, 2015.

[27] Green Revolution Cooling, http://www.grcooling.com, 2015.

[28] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka and J. Luitz, *WIEN2k: An Augmented Plane Wave + Local Orbitals Program for Calculating Crystal Properties* (Institute of Materials Chemistry, Getreidemarkt 9/165-TC, A-1060 Vienna, Austria, 2016), 16.1 edition.

[29] M. Valiev, E. J. Bylaska, N. Govind, K. Kowalski, T. P. Straatsma, H. J. J. van Dam, D. Wang, J. Nieplocha, E. Apra, T. L. Windus and W. A. de Jong, NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations, *Comput. Phys. Commun.* **181**(9):1477–1489, 2010.

[30] M. Prugger, L. Einkemmer and A. Ostermann, Evaluation of the partitioned global address space PGAS model for an inviscid Euler solver, *Parallel. Comput.* **60**:22–40, 2016.

[31] F. Blagojevic, P. Hargrove, C. Iancu and K. Yelick, PGAS10, in *Hybrid PGAS Runtime Support for Multicore Nodes*, *4th Conference on Partitioned Global Address Space Programming Models*, 2010.

[32] G. Almasi, P. Hargrove, G. Tanase and Y. Zheng, PGAS11, in *UPC Collectives Library 2.0*, *5th Conference on Partitioned Global Address Space Programming Models*, 2011.

[33] K. Fürlinger, C. Glass, J. Gracia, A. Knüpfer, J. Tao, D. Hünich, K. Idrees, M. Maiterth, Y. Mbedheb and H. Zhou, Dash: Data structures and algorithms with support for hierarchical locality, in *Euro-Par 2014*, 2014.

[34] J. Nieplocha, B. Palmer, V. Tipparaju, M. Krishnan, H. Trease and E. Apra, Advances, applications and performance of the global arrays shared memory programming toolkit, *Int. J. High Perform. Comput. Appl.* **20**(2):203–231, 2006.

[35] H. B. Bröker, J. Campbell, R. Cunningham, D. Denholm, G. Elber, R. Fearick, C. Grammes, L. Hart, L. Heckingu, P. Juhász, T. Koenig, D. Kotz, E. Kubaitis, R. Lang, T. Lecomte, A. Lehmann, A. Mai, B. Märkisch, E. A. Merritt, P. Mikulík, C. Steger, S. Takeno, T. Tkacik, J. V. der Woude, J. R. V. Zandt, A. Woo, and J. Zellner, *gnuplot 4.6 — An Interactive Plotting Program*, 2014.

[36] A. Petitet, C. Whaley, J. Dongarra, A. Cleary and P. Luszczek, HPL 2.1 – A portable implementation of the high-performance Linpack benchmark for distributed-memory computers, http://www.netlib.org/benchmark/hpl/hpl-2.1.tar.gz, 2012.

[37] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess and E. Lindahl, GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers, *SoftwareX*, **1-2**:19–25, 2015.

[38] D. A. Case, T. E. Cheatham-III, T. Darden, H. Gohlke, R. Luo, K. M. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang and R. J. Woods, The amber biomolecular simulation programs, *J. Comput. Chem.* **26**(16):1668–1688, 2005.

[39] S. Höfinger, O. Steinhauser and P. Zinterhof, Performance analysis and derived parallelization strategy for a SCF program at the Hartree Fock level, *Lect. Notes Comp. Sci.* **1557**:163–172, 1999.

[40] R. Mahajan, D. Kranzlmüller, J. Volkert, U. H. Hansmann and S. Höfinger, Detecting secondary bottlenecks in parallel quantum chemistry applications using MPI, *Int. J. Mod. Phys. C* **19**(1):1–13, 2008.

[41] G. Kresse and J. Hafner, Ab initio molecular dynamics for liquid metals, *Phys. Rev. B* **47**(1):558–561, 1993.

[42] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. D. Corso, S. de Gironcoli, S. Fabris, G. Fratesi, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari and R. M. Wentzcovitch, QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials, *J. Phys.*: *Condens. Matter* **21**(39):395502, 2009.

[43] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, *J. Comput. Phys.* **117**(1):1–19, 1995.