Productivity and Software Development Effort Estimation in HPC

Sandra Wienke
IT Center & Chair for High Performance Computing, RWTH Aachen University, Aachen, Germany

***Abstract*—Relying vast HPC investments on an informed decision making process is important when serving the ever increasing demands for computational power. Providing a quantitative metric to compare and evaluate HPC systems in procurement processes, I define a productivity model with predictive power that focuses on the number of simulation application runs and the total cost of ownership (TCO). As part of TCO, the software development costs determined by efforts to parallelize, port or tune simulation codes for novel HPC setups must be predicted. Since approaches from mainstream software engineering does not directly suite HPC needs, I set up a methodology to estimate software development effort in HPC. Proof of concepts are mostly based on real-world HPC setups at RWTH Aachen University.**

I. Introduction

The ever increasing demands for computational power tighten the reigns on available budgets. Constraints on expenses for acquisition and electrical power must be met, yielding continuously-increasing hardware and software complexity that application developers have to deal with. Thus, the need to make informed decisions on how to invest available budgets is inherent and growing in HPC procurement projects. To predict and compare the cost effectiveness of various hardware environments in HPC centers, a quantitative metric is required.

While LINPACK Flop/s have been a prominent metric to compare different HPC systems, it emphasizes number-crunching machines and ignores today's numerous scientific HPC applications that suffer from memory bottlenecks. Metrics such as Flop/s per watt or Flop/s per dollar (of acquisition) followed when discussing power wall and budget constraints. However, they only pick up part of the picture that HPC centers are concerned with in procurement processes which are expenses for hardware, software, maintenance, infrastructure, energy, programming effort, as well as the value that researchers get from the HPC system in terms of scientific output. The latter further incorporates the challenge of defining the value of scientific output in a useful way.

In my thesis, I use *productivity* as overarching figure of merit that is widely used in economics and defined as ratio of output units to input units. I setup a productivity model with predictive power for ex-ante procurement evaluations. For a comprehensive HPC center perspective, I define the number of simulation application runs as output value, and the total cost of ownership (TCO) as input. This model allows to reflect real-world multi-job environments and has been shown to be applicable in HPC setups at RWTH Aachen University.

As part of a sound TCO model, costs for development efforts to parallelize, port and tune simulation codes to efficiently exploit the HPC system (to be bought) must be considered. While these efforts have been taken for granted heretofore, today's continuously-growing hardware and software complexity push corresponding costs to the fore. Lately, the German science advisory council has recommended to incorporate these into the future German HPC funding landscape. However, embracing software development efforts into the TCO model and, hence, into the productivity metric to evaluate cost-effective HPC centers, HPC managers face the challenge to estimate these efforts for informed procurement decisions. Although software cost estimation models are popular in mainstream software engineering (SE), a direct transfer to performance-critical HPC environments is often not feasible.

As part of my thesis and based on several HPC projects, it is shown that the parametric software cost model COCOMO II from SE is not directly applicable to HPC environments. As direct consequence, I introduce a methodology to estimate software development efforts needed for HPC activities. I model the relationship of development effort to performance in a so-called performance life-cycle, and parametrize it with respect to the numerous impact factors on effort. For the latter, I identify key drivers using ranking surveys, and investigate two major factors. Since data collection on human subjects is essential for reliable statistical results, I target at a community effort and provide methods and tools for that.

Thus, the following contributions stand out:

- A productivity model with predictive power that is applicable for evaluations in HPC procurements: It focuses on runtime of simulation codes while considering total ownership costs of the HPC center, including costs for energy and development effort to parallelize, port or tune the respective codes. This enables especially an overarching figure of merit for real-world multi-job setups in HPC centers.
- A methodology to estimate software development effort in HPC as part of the TCO equation in my productivity model: I establish the concept of performance life-cylces that describes the relationship of application performance and effort needed to achieve this performance. I further investigate systematically the many-fold dependence on various impact factors such as the pre-knowledge of the developer. I provide methods to collect corresponding human-subject based data and show proof of concepts.

The remainder of this work is structured as follows: In

SC Doctoral Showcase; Denver, Colorado, USA; November 2017

Section II, I introduce my productivity figure of merit for single and multi application setups and present its sensitivity towards errors in assumptions. In Section III, I describe my methodology on development effort estimation that introduces a performance life-cycle, and identifies and quantifies impact factors on effort. Finally, I conclude in Section IV.

II. Productivity

My research targets at supporting informed decision making in HPC procurement processes at German university HPC centers by providing a quantitative, understandable and real-world-applicable metric to evaluate potential HPC setups.

My approach is based on the productivity metric that is prevailing in economics to measure the value of products or programs. Basically, productivity Ψ can be expressed as ratio of units of output to units of input [1], or as cost effectiveness ratio given as value over costs:

$$\Psi = \frac{\text{output}}{\text{input}} = \frac{\text{value}}{\text{cost}} \tag{1}$$

Applying this figure of merit to university HPC centers, I define the *value* as scientific output. The challenge lies in the numerical quantification of science and research with their intangible character. While different approaches exist, e.g, Apon et al. [2] use the number of publications or amount of funding, and Hyperion expresses the value of HPC as return on investment (ROI) [3], I use the *number of simulation application runs* that an HPC center of size n (number of compute nodes) can perform over the system's lifetime τ:

$$\text{output} := \sum \text{application runs} = \sum_i r_{app,i}\,(n,\tau) \tag{2}$$

with $r_{app,i}$ the number of executions of application i.

As denominator of productivity, I define total ownership costs of the HPC environment to incorporate all arising costs. I sum up substantial one-time costs C^{ot} and annual costs C^{pa}:

$$\text{input} := \text{TCO}\,(n,\tau) = C^{ot}(n) + C^{pa}(n)\cdot\tau. \tag{3}$$

Combining (1), (2) and (3), my definition of productivity Ψ is as follows:

$$\Psi(n,\tau) = \frac{\sum r_{app,i}(n,\tau)}{\text{TCO}(n,\tau)}. \tag{4}$$

Other productivity models have been mainly proposed as part of the HPCS program [4] and published in [5], such as the $(\text{SK})^3$ synthesis model. However, with respect to available publications, existing models have only rarely been shown to be applicable in real-world HPC procurement setups due to their complex structure, little predictive power, or neglect of job mix setups. Instead, I show that my productivity model given in (4) is suitable for various HPC use cases based on real-world data taken from RWTH Aachen University. For example, to compare CPU- and accelerator-based systems [6] (see abstraction in Fig. 1), argue for two-phase procurements vs. single-phase purchases [7] (schematic view in Fig. 2) or investigate various operational concepts of GPU-based systems [8].

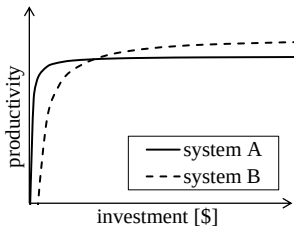

Fig. 1. Productivity for comparison of different systems

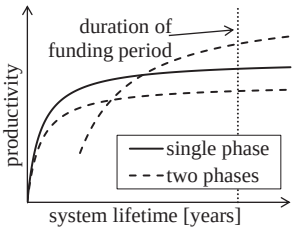

Fig. 2. Productivity for comparison of single- and 2-phase procurements

A. Value: Number of Application Runs

Taking the number of application runs as value is based on the supposition that each *scientific output* of a simulation application delivers insights into the corresponding scientific field, no matter whether the result represents a scientific success or a scientific nonachievement. In addition, it is an easily quantifiable and understandable metric that encapsulates predictability using performance models.

As abstraction of a typical cluster's job mix, I present a model for a single application first before composing it to multiple simulation codes. For a single application i, the number of runs $r_{app,i}$ are computed by dividing the overall time that the system is available by the application's runtime $t_{app,i}$ [7]:

$$r_{app,i}(n,\tau) \sim \frac{\alpha\cdot\tau}{t_{app,i}(n)} \tag{5}$$

where α describes the system availability rate in percent and addresses maintenance periods or unreliability of the system. In my thesis, I also introduce a quality weighting factor $q_{app,i}$ that accounts for the benefit of large-scale runs, i.e., the increase of data set resolution in favor of shorter runtimes.

For the composition to a job mix amount of applications, the runtime (prediction) of all simulation codes exploiting the cluster is needed. Since this could amount to several hundred, I propose a reduction to m "relevant" applications. These can be found, for instance, by a particular small set of applications that will be part of the call for tenders (usually in the order of tens). Another approach is based on previous cluster statistics that reveal the applications that occupied most compute time, e.g., by core-h. At RWTH Aachen University, the top 15 HPC projects consumed $50\,\%$ of the compute resources in 2015, and the top 35 projects took $75\,\%$. From this latter approach, a capacity weighting factor p_i for each application can be derived so that holds:

$$\sum_i^m r_{app,i}(n,\tau) = \sum_i^m \left(p_i \cdot \frac{q_{app,i}(n_i)\cdot\alpha\cdot n\cdot\tau}{n_i\cdot t_{app,i}(n_i)} \right) \tag{6}$$

with n_i the number of nodes that application i is running on.

B. Cost: Total Cost of Ownership

For refining (3), I continue to concentrate on an HPC center's view. While my main categorization of costs divides TCO into one-time costs and costs per anno, I further break it down on a *per node* and *per node type* basis that is motivated by differences, e.g., in CPU- and accelerator-type systems.

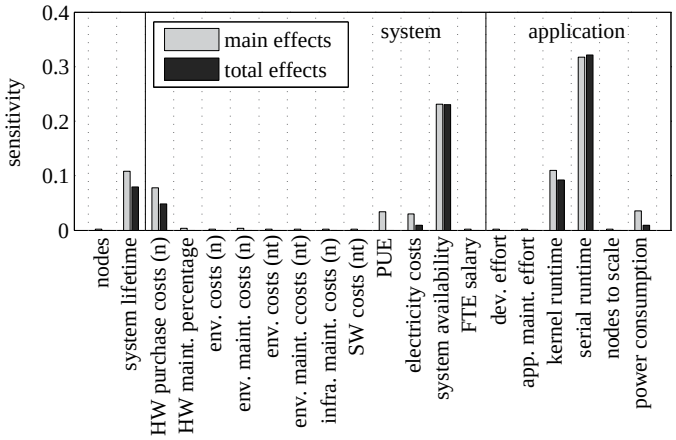

Fig. 3. Exemplary sensitivity analysis based on data from RWTH Aachen University. Main effects (percentage) result from varying the specific parameter alone. Total effects cover the output variance by parameter interaction.

One-time costs comprise expenses, e.g., for hardware and infrastructure acquisition, for setup and installation of an appropriate operating system and environment, as well as for development efforts for HPC activities. Annual costs cover expenditures for maintenance of hardware and the cluster's environment, amortization of building the data center, software and compiler licensing, energy and the maintenance of the simulation code. More details and real-world quantifications for these parameters taken from RWTH Aachen University can be found in [6], [7], [8].

For the incorporation of multiple applications into the TCO model, efforts for development and maintenance of all m simulation codes are simply summed up. The application-dependent part of the energy costs must be expressed accordingly to the capacity-based approach of distributing applications across the whole HPC cluster and, hence, include the applications' capacity weighting factors.

C. Sensitivity Analysis

When predicting and quantifying all parameters of the productivity model, several assumptions must be taken. Since these are generally prone to errors, I evaluate my model using a variance-based uncertainty and global sensitivity analysis. Taking the simulation-based approach by Saltelli et al. [9] and a respective Matlab implementation in the SAFE Toolbox [10], I investigate small-scale and large-scale HPC setups, comparison capabilities of different node types, as well as, output variances due to my reduction approach with respect to the relevant applications only.

Within the given conditions, results show that my productivity metric can be used to compare different HPC setups and that the reduction to a limited number of relevant applications is reasonable. Furthermore, my model is robust against several uncertainty factors since I identify only few performance-related well-understood parameters (main effects in Fig. 3) that must be accurately predicted to minimize the variance in productivity.

III. Software Development Effort

My research aims at estimating software development effort in HPC as part of the productivity metric and to evaluate impact factors on effort, e.g., the parallel programming model used. Tackling the ambiguous definition of *effort*, I consider development effort for HPC activities (only), i.e., parallelizing, porting or tuning simulation applications.

While software cost models are widely used in SE, it is not straightforward to transfer them to performance-critical environments as typical in HPC: obtaining the last percentage points of performance is often a laborious task that is not part of mainstream SE models. For instance, in previous publications [11], [12], we showed that the parametric software cost model COCOMO II [13] is not directly applicable to HPC projects. Besides the generally scarce research in this area, most other works that investigate effort in HPC focus on comparing various programming models in terms of effort or lines of code – often neglecting any other impact factors.

As first systematic approach (to the best of my knowledge) to investigate efforts needed for HPC activities, I introduce a methodology on development effort estimation in HPC [14], [15]. It is based on a so-called performance life-cycle, and methodically identifies and quantifies various impact factors.

A. Performance Life-Cycle

For establishing an effort estimation model in HPC, I follow the basic concept of COCOMO II's approach and setup an algorithmic model whose parameters are determined by regression analysis – as soon as sufficient data sets are collected. Noteworthy, I move from lines of code (used in COCOMO II) to performance as base metric. Thus, I define a function that describes the relationship between performance and effort needed to achieve this performance in dependence on several impact factors – the *performance life-cycle*:

$$\text{effort} = S \cdot f(\text{performance})^R + Q. \tag{7}$$

Here, the function $f(\text{performance})$ can follow different effort-performance relationships, e.g., easily expressed as 80-20 rule, or more complex relationships that are closer to real-world experiences and may depend on various milestones. An abstracted example of such a relationship that has been revealed from human-subject research is shown in Fig. 4. The parameters S, R and Q in (7) represent different impact factors on effort. As a starting point, I assume the following composition of cost drivers (similar to COCOMO II): $S = s_0 \cdot \prod_{j=1}^{n_s} s_j$ and $R = r_0 + \sum_{k=1}^{n_r} r_k$. The factor Q accounts for one-time efforts with $Q = \sum_{l=1}^{n_q} q_l$.

B. Impact Factors on Effort

Since software development effort is dependent on numerous factors, my methodology further evaluates which aspects contribute to S, R and Q in (7). For a systematic factor identification, I developed ranking surveys that I distribute, e.g., to students in HPC classes or professionals in hackathons. Participants rank given factors relatively to each other in terms of higher and lower impact. Additionally, they can suggest

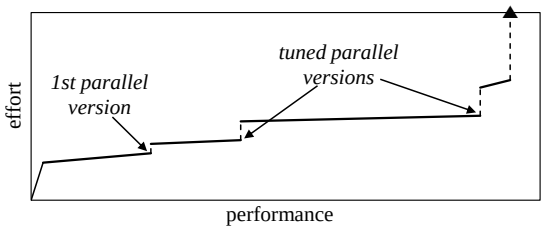

Fig. 4. Exemplary performance life-cycle with different milestones: at the beginning, performance is reached with little effort; at the end, lots of effort is needed to achieve more performance.

further impact factors. Statistical assessment of survey results (Friedman test, Wilcoxon rank test with Holm correction) can reveal factors that are unique or should be grouped or ignored in future studies. Results gathered from 44 participants show currently that the top five factors are: *pre-knowledge on hardware & parallel programming model*, *code work*, *pre-knowledge on numerical algorithm used*, *parallel programming model & compiler/ runtime system* and *performance*.

After identification, the impact factors need to be quantified to be served into the performance life-cycle. Focusing on the key drivers first, I started to investigate the developers' pre-knowledge and the parallel programming model in combination with the numerical algorithm used. To quantify the pre-knowledge of developers, I use so-called knowledge surveys [16] that are actually applied to evaluate intellectual development in, e.g., classrooms. Knowledge surveys ask participants to rate their confidence in answering questions instead of solving tasks. This way, many questions can be dealt with in short time to achieve statistical relevance. Proof of concepts are presented in [14]. For the (methodological) quantification of the impact of the parallel programming model, I follow a pattern-based approach. It assumes that a parallel programming model is especially suitable for certain parallel patterns which categorize the given application algorithm. In a community-wide one-time effort corresponding mappings could be determined.

C. Data Collection

The investigation of the performance life-cycle and corresponding impact factors must be based on sufficient data sets to be statistically reliable. To support the collection of effort-performance pairs in human-subject research, the tool *EffortLog* [14] has been developed. It enables the tracking of effort and performance data with interval-based questionnaires. Moreover, created material is publicly available [17] to foster a community effort for further data collection.

IV. Conclusion

My research focuses on methodologies and models to foster informed HPC procurements in Germany. My first research highlight is a productivity figure of merit of HPC centers that is applicable to real-world multi-job setups. It also contributed to the tender call for the RWTH Compute Cluster *CLAIX* in 2016. My second research highlight covers a methodology for predicting software development efforts in HPC. Overall, my established models prepare, e.g, university institutes for acquiring their own (productive) cluster partitions. In future, I will continue to collect data sets for effort evaluations and investigate conditional refinements of my productivity model.

References

[1] T. J. Coelli, D. P. Rao, C. J. O'Donnell, and G. E. Battese, "An Introduction to Efficiency and Productivity Analysis, 2nd Edition," *Interfaces*, vol. 37, no. 2, pp. 198–199, 2007.

[2] A. Apon, S. Ahalt, V. Dantuluri, C. Gurdgiev, M. Limayem, L. Ngo, and M. Stealey, "High Performance Computing Instrumentation and Research Productivity in U.S. Universities," *Journal of Information Technology Impact*, vol. 10, no. 2, pp. 87–98, 2010.

[3] E. C. Joseph, S. Conway, and C. Dekate, "Creating Economic Models Showing the Relationship Between Investments in HPC and the Resulting Financial ROI and Innovation? and How It Can Impact a Nation's Competitiveness and Innovation," IDC, Tech. Rep., 2013.

[4] J. Dongarra, R. Graybill, W. Harrod, R. Lucas, E. Lusk, P. Luszczek, J. Mcmahon, A. Snavely, J. Vetter, K. Yelick, S. Alam, R. Campbell, L. Carrington, T.-Y. Chen, O. Khalili, J. Meredith, and M. Tikir, "DARPA's HPCS Program: History, Models, Tools, Languages," in *Advances in COMPUTERS High Performance Computing*, ser. Advances in Computers, M. V. Zelkowitz, Ed. Elsevier, 2008, vol. 72, pp. 1–100.

[5] J. Dongarra and B. R. De Supinski, Eds., *International Journal of High Performance Computing Applications*. Thousand Oaks, CA, USA: Sage Publications, 2004, vol. 18(4).

[6] S. Wienke, D. an Mey, and M. S. Müller, "Accelerators for Technical Computing: Is It Worth the Pain? A TCO Perspective," in *Supercomputing*, ser. Lecture Notes in Computer Science, J. M. Kunkel, T. Ludwig, and H. W. Meuer, Eds. Springer Berlin Heidelberg, 2013, vol. 7905, pp. 330–342.

[7] S. Wienke, H. Iliev, D. an Mey, and M. S. Müller, "Modeling the Productivity of HPC Systems on a Computing Center Scale," in *High Performance Computing*, ser. Lecture Notes in Computer Science, J. M. Kunkel and T. Ludwig, Eds. Springer International Publishing, 2015, vol. 9137, pp. 358–375.

[8] F. P. Schneider, S. Wienke, and M. S. Müller, "Operational Concepts of GPU Systems in HPC Centers: TCO and Productivity," in *15th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platform (HeteroPar)*, 2017, accepted.

[9] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis: The Primer*. John Wiley & Sons, 2008.

[10] F. Pianosi, F. Sarrazin, and T. Wagener, "A Matlab toolbox for Global Sensitivity Analysis," *Environmental Modelling & Software*, vol. 70, pp. 80–85, 2015.

[11] M. Nicolini, J. Miller, S. Wienke, M. Schlottke-Lakemper, M. Meinke, and M. S. Müller, *Software Cost Analysis of GPU-Accelerated Aeroacoustics Simulations in C++ with OpenACC*. Cham: Springer International Publishing, 2016, pp. 524–543.

[12] J. Miller, S. Wienke, M. Schlottke-Lakemper, M. Meinke, and M. S. Müller, "Applicability of the Software Cost Model COCOMO II to HPC Projects," *International Journal of Computational Science and Engineering*, 2017, accepted.

[13] B. Boehm, C. Abts, A. W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, "COCOMO II - Model Definition Manual, Version 2.1," University of Southern California, Tech. Rep., 2000.

[14] S. Wienke, J. Miller, M. Schulz, and M. S. Müller, "Development Effort Estimation in HPC," in *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2016, pp. 107–118.

[15] S. Wienke, T. Cramer, M. S. Müller, and M. Schulz, "Quantifying Productivity—Towards Development Effort Estimation in HPC," Poster at the International Conference for High Performance Computing, Networking, Storage and Analysis (SC15), 2015.

[16] E. Nuhfer and D. Knipp, "The Knowledge Survey: A Tool for All Reasons," *To Improve the Academy*, vol. 21, pp. 59–78, 2003.

[17] S. Wienke, "Development Effort Methodologies," Chair for High Performance Computing, RWTH Aachen University, http://www.hpc.rwth-aachen.de/research/tco, 2016.