Towards Reconfigurable HPC Component Models

Christian Perez
Avalon, LIP
Univ. Lyon, Inria, CNRS, ENS Lyon, UCBL
Lyon, France
christian.perez@inria.fr

Vincent Lanore
LBBE, UMR 5558
Univ. Lyon, Université Claude Bernard Lyon 1, CNRS
F-69622 Villeurbanne, France
vincent.lanore@univ-lyon1.fr

INVITED TALK EXTENDED ABSTRACT

A. Introduction

High performance computing (HPC), as other domains, is facing growth of the complexity of both hardware and software. Hardware complexity can be observed in the diversification of computing units (multi-core, many-core, GPGPU, FPGA, etc.) and, more recently, in storage elements (SSD and then SVRAM). Many of these new technologies stem from frequency scaling limitations and energy consumption issues. Hence, not only do applications have to be continuously adapted to varied and changing hardware configurations, but also energy-related concerns encourage runtime adaptation to use as little resources as possible.

Software complexity on the other hand stems from the need to simulate more and more complex phenomena. For example, large physics codes can be coupled to perform complex multi-physics simulations. Complex HPC applications can reach many hundreds of thousands of lines of finely optimized code. Such applications are very tough to develop, to maintain and to adapt to new hardware. Runtime adaptation is particularly complex and usually raises high development costs.

Software engineering techniques aims at easing the development and adaptation of applications. Thus, they have the potential to substantially lower development costs. Performance, however, is paramount for HPC, which restricts suitable software engineering technologies to those with very low performance overhead.

B. Reconfiguration in Component Models

Component-based programming is a programming paradigm that is known to improve software reuse and separation of concerns [1]. It consists in writing applications as *assemblies* of well-encapsulated pieces of software called *components*. A framework specifying component nature and assembly rules is called a *component model*. A component assembly can be used as a high-level application model for the purpose of adaption, hiding the low-level details inside the components.

While static adaption of a component application consists in generating an adapted assembly [2], runtime adaptation (also called *reconfiguration*) presents some specific challenges. Assembly transformations have to be specified somehow and then performed on a running application without breaking anything. This problem becomes even more difficult in HPC, where overheads have to be kept low, and where centralized control is anathema to large-scale scalability.

There exist some component models built specifically for HPC such as CCA [3] and L2C [4]. These models however do not provide any assistance for reconfiguration and thus let the entire reconfiguration burden to application developers.

There are some reconfigurable component models outside of HPC such as FRACTAL [5] SOFA2 [6], or Pycots/Coqcots [7]. They are also interesting reconfiguration oriented DSL for component models such as FScript [8] for FRACTAL. However, they rely on centralized reconfiguration control. While this approach greatly simplifies getting correct transformations, its centralization affects performance too negatively to be compatible with HPC.

A reconfigurable HPC component model would need to both provide an easy way to specify and perform correct assembly transformations, and achieve the performance of hand-coded reconfiguration in HPC applications.

C. DirectMod and DirectL2C

We have started to deal with the problem of high-performance reconfiguration of component assemblies by specifying the *DirectMOD* formal component model [9, 10] and by providing *DirectL2C* [9], its C++ implementation. *DirectMOD* allows the specification and execution of low-overhead assembly transformations with fully distributed control. DirectL2C is a proof of concept implementation of DirectMOD on top of L2C. We have used *Adaptive Mesh Refining* (AMR), a common optimization requiring runtime adaptation, as a use-case to illustrate that it is possible to combine software engineering concepts and high performance.

Keywords-HPC, component models, reconfiguration.

978-1-5386-7879-4/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCS.2018.00038

ACKNOWLEDGMENT

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

BIOGRAPHIES

CHRISTIAN PEREZ received a PhD in computer science at ENS Lyon in 1999. He is an Inria researcher since 2000, and a senior Inria researcher since 2011. He is leading the Avalon research team since 2012. His topics of research include parallel and distributed programming models and resource management. He published more than 60 papers in international journals, conferences, and workshops.

VINCENT LANORE received his PhD in computer science at ENS de Lyon in 2015 under the direction of Christian Perez on the topic of reconfigurable component models for HPC. He has been a temporary researcher at CNRS since 2016 in the context of the CONVERGENOMIX bioinformatics project. His research interests include software engineering for scientific computing, bioinformatics, and statistical inference.

REFERENCES

[1] C. Szyperski. "Component Software: Beyond Object-Oriented Programming (2nd ed.)". Addison-Wesley Longman Pub. Co., Inc., Boston, MA, USA. 2002.

[2] V. Lanore, C. Pérez, J. Richard. "Towards Application Variability Handling with Component Models: 3D-FFT Use Case Study". UnConventional High Performance Computing 2015, Aug 2015, Vienne, Austria. Springer, Euro-Par 2015: Parallel Processing Workshops, 9523 (61), pp.12, 2015.

[3] B. A. Allan and al., "A Component Architecture for High-Performance Scientific Computing," International Journal of High Performance Computing Applications, 2006.

[4] J. Bigot, Z. Hou, C. Pérez, and V. Pichon, "A low level component model easing performance portability of HPC applications," Computing, 2013.

[5] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani. "The fractal component model and its support in Java". Software: Practice and Experience, 36(11-12):1257–1284, 2006.

[6] T. Bures, P. Hnetynka, and F. Plasil. "Sofa 2.0: Balancing advanced features in a hierarchical component model. In Fourth Intl Conference on Software Engineering Research, Management and Applications. pp 40–48, Aug 2006.

[7] J. Buisson, E. Calvacante, F. Dagnat, E. Leroux, and S. Martinez. "Coqcots & Pycots: non-stopping components for safe dynamic reconfiguration". In the 17th Intl ACM Sigsoft Symp. on Component-Based Software Engineering, page 1, Lille, France, June 2014.

[8] P.-C. David, T. Ledoux, et al. Safe dynamic reconfigurations of Fractal architectures with FScript. In Proceeding of Fractal CBSE Workshop, ECOOP, volume 6, 2006.

[9] V. Lanore. "On Scalable Reconfigurable Component Models for High-Performance Computing ". PhD of ENS Lyon. 2015. http://www.theses.fr/2015ENSL1051.

[10] V. Lanore, C. Pérez. "A Reconfigurable Component Model for HPC". In The 18th Intl ACM Sigsoft Symposium on Component-Based Software Engineering (CBSE 2015), p 10, May 2015, Montréal, Canada.