An Evaluation of Productivity and Performance in the EPEEC project

Tom Vander Aa <tom.vanderaa@imec.be>, ExaScience Life Lab at imec, Belgium
Antonio Peña <antonio.pena@bsc.es>, Barcelona Supercomputing Centre, Spain

The EPEEC European Project [1] is developing parallel programming environments to cope with heterogeneous exascale supercomputers and turn them into manageable platforms for domain application developers. One of its main goals is to combine high-performance with high programmer productivity.

To achieve this, EPEEC builds on and integrates existing European technology such as automatic generation of directives with the Parallelware tools [2], task-based programming with OmpSS [3] and asynchronous communication using GASPI [4]. EPEEC extends these technologies by providing an integrated set of programming guidelines for parallel applications [6], improving support for tasking on accelerators and support for eventually consistent collectives, amongst many other.

SMURFF, (Scalable Matrix Factorization Framework [5]) is one EPEEC's driver application. SMURFF implements a set of high-performance matrix factorization methods that are use for recommender systems (RS). SMURFF is used in many domains ranging from marketing to drug discovery [7], on extremely large datasets and by users that have little experience with low-level optimizations for HPC systems. These last two items make support for high-productivity high-performance programming crucial.

In this poster we will present the productivity and performance improvements we were able to achieve by applying EPEEC's guidelines on the SMURFF application. We will show the performance-productivity trade-offs using two types of examples:

1. *Small change in code, large impact in productivity or performance.* In some examples a small change in the code (e.g., adding a single #pragma) can result in a significant performance improvement. We show examples of what the user can expect from the automatically added directives by the parallelization tools, and examples from adding manual directives for accelerator off-loading.
2. *Comparison of different application codes*. Sometimes it is needed to rewrite a significant part of the application to evaluate two or more EPEEC programming models. In this case we compare effort versus performance of different parallel versions of the same application code.

Acknowledgments

This work has been supported by the EU H2020 FET-HPC project EPEEC (contract #801051).

References

1. EPEEC: https://epeec-project.eu/
2. Parallelware Technology: https://www.appentra.com/parallelware/
3. The OmpSS Programming Model: https://pm.bsc.es/ompss
4. GASPI: http://www.gaspi.de/
5. SMURFF: https://github.com/ExaScience/smurff
6. EPEEC Programming Guidelines: https://epeec-project.eu/results/programming-guidelines
7. Sturm, N., Mayr, A., Le Van, T. et al. Industry-scale application and evaluation of deep learning for drug target prediction. J Cheminform 12, 26 (2020). https://doi.org/10.1186/s13321-020-00428-5