Machine Learning for Chemogenomics on HPC in the ExCAPE Project

Tom Vander Aa and Tom Ashby
IMEC,
Leuven, Belgium
email:*firstname.lastname*@imec.be

Yves Vandriessche
Intel corp.
Belgium
email:yves.vandriessche@intel.com

Vojtech Cima, Stanislav Böhm and Jan Martinovič
IT4Innovations, VŠB – Technical University of Ostrava,
Ostrava, Czech Republic
email:*firstname.lastname*@vsb.cz

***Abstract*—The ExCAPE project is a Horizon 2020 project to advance the state of the art of machine learning (ML) implementations on supercomputing hardware. We have adopted bioactivity predictions for chemogenomics as a challenging use-case to drive development. In this paper, we will give an overview of the challenges in ExCAPE to use supercomputing efficiently. We will touch on three key examples dealing with efficient ML workflow execution, support for multi-task learning using matrix factorization methods and the challenges originating from the large and very sparse datasets in ExCAPE.**

***Index Terms*—Machine Learning, High-Performance Computing, Collaborative Filtering, Distributed Task Scheduling**

I. Introduction and Context

Traditional users of High Performance Computing (HPC) have mostly been concerned with simulation of physics of one type or another and at various different scales. In the last decade, a new breed of user of very large machines has appeared, those concerned with Big Data. Carrying out simulations is mostly about doing large amounts of computation to observe the behavior of a sophisticated model with few parameters. Big Data problems, by contrast, usually deal with less sophisticated models but with many more parameters, and try to choose the model parameters by analyzing large amounts of data with relatively little associated computation. Folk wisdom in this field states that the ability to capture and analyze more data is more valuable than making more sophisticated models, and this works well when data is cheap and easy to get. However, there are problems in this area for which the data are very expensive to generate. In this case, it becomes important to be able to use more sophisticated models to be able to squeeze as much knowledge as possible out of the data. Such problems are at the juncture of HPC and Big Data in that they have large data sets to analyze, yet should exploit more sophisticated models through computation to make the most of the available data.

The ExCAPE project [1] is about how to tackle such problems. The core of the project is about mathematics and software and how they work on HPC machines. However, to be able to advance the state of the art it helps to have a concrete problem to tackle. For this, we take the chemogenomics problem, that of predicting the activity of compounds in the drug discovery phase of the pharmaceutical industry, leading to the project name *Exascale Compound Activity Prediction Engines* (ExCAPE). Making such predictive models belongs to the field of Machine Learning.

More general than chemogenomics, we want to find methods and systems that can tackle large and complex machine learning problems. This will require algorithms and software that make efficient use of the latest HPC machines. Creating these, along with preparing the data to give the system something to work on, is the main work of the project.

Many interesting open challenges need to be overcome to be able to run machine learning efficiently at scale on HPC hardware in all cases. In the following sections of this paper we explain three very relevant and interesting example challenges, namely:

- how to execute machine learning workflows with many dependent tasks (Section II);
- how to take advantage of the fast interconnect (like inifiniband) typically only found on true HPC hardware for multi-node machine learning tasks (Section III);
- how to support very large but sparse datasets (Section IV)

More details can be found in the referenced documents.

II. Efficient Workflows using HyperLoom

Solutions for scheduling problems on HPC systems exist when dispatching tasks (computational units) of known duration and resource requirements. However, real-world applications such as those in machine learning, encompass tasks with no requirement annotations in addition to the overhead of handling massive amounts of data only manageable by large-scale distributed environments. To address these challenges, we developed HyperLoom [2]. HyperLoom is a platform for defining and executing pipelines in large-scale distributed environments. Unlike other scheduling systems, HyperLoom is specifically tailored to work efficiently on high-performing computing (or HPC) systems and offers a user-friendly representation of tasks as acyclic computational graphs. An example of such a scientific pipeline is shown in Figure 1.

Our resulting pipelines for both synthetic and real-world use cases are successfully distributed leveraging HPC resources covering up to a hundred thousand tasks, across ten or more physical compute nodes. We analyzed HyperLoom performance for both synthetic and real test cases scaling up to hundreds of thousands tasks and tens of physical computational nodes. HyperLoom significantly outperforms Dask/Distributed

 ISBN: 978-1-61208-567-8

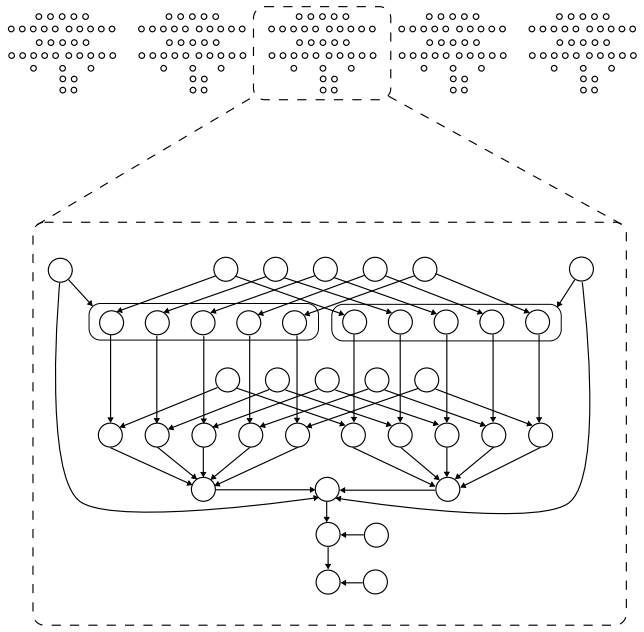

Figure 1. Example of a scientific pipeline visualized as a directed acyclic graph.

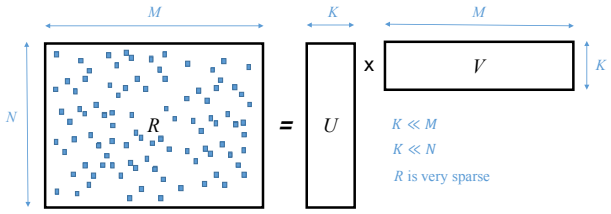

Figure 2. Low-rank Matrix Factorization

ranging from 6.3× to 2.2× better performance for two test cases [2].

III. Scalable Matrix Factorization using Asynchronous Communication

While single-task learning methods for compound-activity prediction result in many small tasks, which can easily be solved using HyperLoom, multi-task learning methods combine and solve multiple tasks at the same time [3], by exploiting commonalities and differences across tasks. This generally results in improved prediction accuracy but also in much fewer and much larger tasks. Such tasks need to be parallelized themselves to be able to run them efficiently.

Matrix Factorization (MF) is a class of multi-task methods that have been successfully used. As sketched in Figure 2, the idea of these methods is to approximate the compound-activity matrix R as a product of two low-rank matrices U and V such that $R \approx U \times V$. In this way, U and V are constructed from the known ratings in R, which is usually very sparsely filled. The recommendations can be made from the approximation $U \times V$, which is dense.

We have built a high-performance distributed implementation [4] of the popular and effective matrix factorization algorithm called Bayesian Probabilistic Matrix Factorization (BPMF [5]). We have shown that load balancing and low-overhead asynchronous communication are essential to achieve good parallel efficiency, clearly outperforming more common synchronous approaches like GraphLab [6]. The achieved speed-up allowed us to speed up machine learning for drug discovery on an industrial dataset from 15 days for the initial Julia-based version to 5 minutes using the distributed version with Intel Threading Building Blocks (TBB) and the Global Address Space Programming Interface (GASPI) [7].

IV. Support for Large Binary Sparse Matrices

There is one class of routines that is ubiquitous for all involved Machine Learning algorithms on ExCAPE-like data: sparse linear algebra. In many cases, these operations dominate the runtime of the computation, which means that even small optimizations translate into large efficiency and performance gains.

Unlike dense linear algebra routines, where ready-made heavily optimized libraries are available, the performance of sparse routines is determined by the nature of the data, which makes it impossible to make a routine that is optimal in general.

Sparse matrix-vector (SPMV) was identified as a key bottleneck low-level operation is the ExCAPE ML-algorithms. The two main workloads we are analyzing are matrix-factorization and neural network training on ExCAPE's activity and the compounds' fingerprint data.

After optimization [8] of those SPMV routines on ExCAPE data [9] we can say that:

- Linear algebra optimizations on the Compressed Storage of Rows (CSR) format performs best from those formats tested, but
- although the Coordinate Format (COO) is less efficient, it gets close to CSR by sorting non-zero elements by Hilbert-order [10], making it a good choice in case the matrix contents are changing dynamically.
- Optimizing for binary matrix data and parallel task imbalances yields a 2.7× speedup over the more general `mkl_scsrgemv()` Intel Math Kernel Library (MKL) SparseBlas routine.
- Operating on multiple right-hand sides obtains another 2.4× proportional speedup due to both vectorization and improved cache locality.

V. Conclusions

This short paper provided insight into what is needed to run large scale machine learning efficiently on HPC hardware for three key examples. We have indicated the importance of efficient *i)* workflow execution, *ii)* support for multi-task learning, and *iii)* low lever sparse algebra routines. Solutions for all three will need to be combined for the project to succeed.

 ISBN: 978-1-61208-567-8

ACKNOWLEDGMENTS

This work is partly funded by the European project ExCAPE with reference 671555 and by the IT4Innovations infrastructure, which is supported from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center – LM2015070".

REFERENCES

[1] The ExCAPE Consortium, "ExCAPE: Exascale Compound Activity Prediction Engine," http://excape-h2020.eu/, retrieved: June 2017.

[2] V. Cima *et al.*, "HyperLoom possibilities for executing scientific workflows on the cloud," in *Proceedings of the CISIS 2017 : The 11th International Conference on Complex, Intelligent, and Software Intensive Systems*, 2017.

[3] R. Caruana, "Multitask learning," in *Learning to learn*. Springer, 1998, pp. 95–133.

[4] T. Vander Aa, I. Chakroun, and T. Haber, "Distributed bayesian probabilistic matrix factorization," in *ICCS 2017: International Conference on Computational Science*, June 2017.

[5] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proceedings of the International Conference on Machine Learning*, vol. 25, 2008, pp. 880–887.

[6] Y. Guo, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke, "Benchmarking graph-processing platforms: A vision," in *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '14. New York, NY, USA: ACM, 2014, pp. 289–292. [Online]. Available: http://doi.acm.org/10.1145/2568088.2576761

[7] D. Grünewald and C. Simmendinger, "The GASPI API specification and its implementation GPI 2.0," in *7th International Conference on PGAS Programming Models*, vol. 243, 2013, pp. 243–248.

[8] Y. Vandriessche and T. Vander Aa, "ExCAPE deliverable D2.6: Simulation report 1," Tech. Rep., 2017.

[9] J. Sun *et al.*, "ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics," *Journal of Cheminformatics*, vol. 9, no. 1, p. 17, dec 2017. [Online]. Available: http://jcheminf.springeropen.com/articles/10.1186/s13321-017-0203-5

[10] A. N. Yzelman, D. Roose, and K. Meerbergen, "Sparse matrix-vector multiplication: parallelization and vectorization," in *High Performance Parallelism Pearls: Multicore and Many-core Programming Approaches*, J. Reinders and J. Jeffers, Eds. Elsevier, 2014, ch. 27, p. 20.

 ISBN: 978-1-61208-567-8