High-Performance Computing Group
Prof. Dr. Christian Plessl
Computer Science Department and
Paderborn Center for Parallel Computing

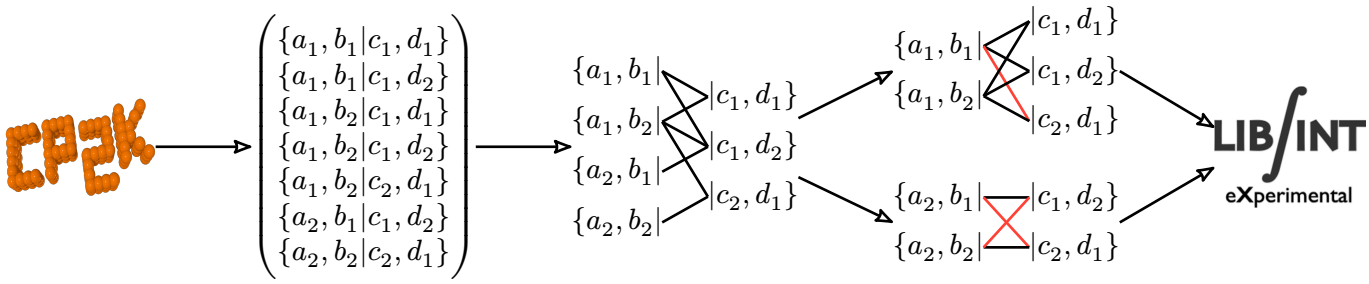
Grouping and Scheduling Electron Repulsion Integral Quartets for GPU acceleration

Master Thesis

At a glance

- Construct a bipartite graph from input data generated by CP2K
- Efficiently find and extract almost-complete subgraphs
- Implement a CPU-GPU offloading scheme for computing on the subgraphs
- No prior knowledge of quantum chemistry is required nor expected

Electron Repulsion Integrals (ERIs) are an essential quantity for ab initio molecular dynamics packages such as CP2K. Even small simulations containing only a few atoms already require the computation of billions of ERIs. Their computation is therefore a desirable target for GPU acceleration. Here, the LibintX library has recently shown promising performance results on GPU using a matrix-multiplication based approach, but unfortunately the input format of LibintX is not compatible with the input generation process of CP2K:
LibintX takes a list of $\{ab|$ shell pairs and a list of $|cd\}$ shell pairs as input, and computes the outer product between the two lists, i.e. each element in $\{ab|$ is combined with each element in $|cd\}$. CP2K, on the other hand, generates a list of quartets $\{ab|cd\}$ – a quartet is a container for multiple ERIs – which usually does not consist of all pair products. The distinction between the two formats is especially visible when treating a quartet list as a bipartite graph, where the two pair lists are the nodes on either side:



The pair lists extracted from the quartet list contain four and three elements, respectively, whereas the number of connections in the bipartite graph matches the size of the quartet list. Submitting the two extracted pair lists directly to LibintX would compute 12 quartets even though only 7 are required. If instead the graph is split into two subgraphs, then only 3 unnecessary quartets are computed by LibintX. The generated subgraphs, however, are smaller than the original graph and are therefore processed less efficiently by LibintX. This trade-off also has to be considered when generating the subgraphs.

When choosing this thesis, your task will be to solve the graph problem described above on CPU. Ideally, as few CPU cores as possible should be required to keep the GPU(LibintX) busy. You will also have to thoroughly benchmark LibintX to correctly estimate the cost of a subgraph.

Further Reading:
LibintX: https://doi.org/10.1021/acs.jpca.3c04574, https://doi.org/10.1063/5.0217001

Contact:
Johannes Menzel, Office: X1.125, Phone: 05251 60-1741, Mail: johannes.menzel@uni-paderborn.de

For information on theses offered by the HPC group or PC2 please visit:
https://en.cs.uni-paderborn.de/hpc/teaching/open-theses