

1 Introduction

As part of James E. Avery’s efforts to develop an efficient screening pipeline for fullerenes and potentially fulleroids we will in this report detail our efforts in porting parts of the xtb program by Grimme et al to SYCL code. The goal is a highly optimised and fully lockstep-parallel implementation of the electronic structure calculations from the GFN2-xTB method. A previous thesis by la Cour provides an efficient lockstep-parallel implementation of a forcefield method for computing geometric structures of fullerenes, which we will take to be our input.

The mentioned screening pipeline would enable the search of entire isomer spaces for fullerenes with certain properties such as a low lowest energy state indicating a stable isomer.

A fullerene is a molecule consisting only of carbon atoms connected in 12 pentagons and enough hexagons to create a hollow structure. As we increase the number of atoms the isomer space quickly grows leading to very slow search times. We aim to provide a quick and relatively accurate method for discarding large unpromising parts of the isomer space before searching with more accurate methods.

Fulleroids are essentially an extension to fullerenes where we allow n-gons instead of only penta- and hexagons, as long as we can still create a closed shape.

After a literature review we settled on the Geometry, Frequency, Noncovalent, extended Tight Binding (GFNn-xTB) family of methods as they are relatively accurate and quite fast at predicting electronic structures to a reasonable accuracy. We hope that this inherent speed will aid in getting good though-put after the transformation to a lockstep-parallel version. The GFNn-xTB methods come in iterations. GFN1 is the first and lays the ground work for the later iterations. It does however rely on element pairwise specific constants. In GFN2 this has been changed in favour of only element specific parameters. GFN0 is a more approximate and faster version of GFN2. And GFN-FF takes this trade-off further as this is a forcefield method which is parametrised using the insights (and parameters) gained from the other GFN iterations.

Forcefield methods save on computing all the pairwise interactions between atoms in a molecule and instead use efficient rules to lump atoms together in predictable clumps which then interact with other clumps. This can save tremendous effort.

Specifically GFN2 seemed most promising for our purposes as it is more accurate than GFN0 and simpler than GFN1, and if it is not fast enough would be relatively easy to then implement GFN0. GFN-FF was not considered suitable due to us wanting to see if it could be fast enough without defaulting to a forcefield method.

Lockstep-parallelisation is a paradigm best suited for GP-GPU. It takes advantage of the fact that GPUs operate more efficiently when all the cores are doing the same operations in a predictable fashion. This essentially is a step beyond data parallelism. We are not only operating on the same data across

cores, but also doing the exact same steps. This means no conditionals with a data dependent evaluation. It is fine to have a loop that runs five times, opposed to say `data[coreId]` times.