7 | Conclusions

The last few years have seen a huge increase in computational power in the form of special purpose hardware and new supercomputers. This is the direct result of the increasing amount of parallelism available in current day computer chips. However, in order to use this computational power, the user — or better, the developer — is forced to rethink the design and implementation of algorithms. Without taking advantage of the available multi-core technology there will hardly be any advantage of buying a new computer. We see this trend in the Central Processing Unit (CPU) and the Graphics Processing Unit (GPU), but also for example in the mobile phone industry where quad-cores are the current day standard and octo-cores are slowly being introduced.

This thesis presents how we can benefit from the available processing power of these many-core chips, in our case GPUs, when performing astrophysical simulations. This can either be by implementing expensive, but accurate, algorithms such as direct N-body methods (see Chapter 2) or by taking it a step further and transforming the hierarchical Barnes-Hut tree-code method into a version that is suitable for many-core architectures (see Chapters 3, 4 and 6). The resulting simulation codes have a performance that is one to two orders of magnitude higher than previous versions. This allows for new kinds of science and wider parameter searches. For example, the work in Chapter 5 is the result of hundreds of simulations, while other works about the same topic usually do not perform more than a dozen simulations.

In this work we kept the direct N-body method and the tree-code method strictly separate, but in the future it might be beneficial to make use of both methods or make one of the methods part of a larger (existing) code. We will discuss this and more in the next paragraphs.

7.1 BRIDGE; Combining direct and hierarchical N-body methods

The difference between particle numbers used in collisional and collisionless methods has, because of their difference in scaling complexity, over the years only increased. Depending on the problem, scientists either choose for high precision direct N-body methods or for large particle numbers using approximation methods like the tree-code. Recently, however, methods have been introduced that try to combine the best of both worlds: the high

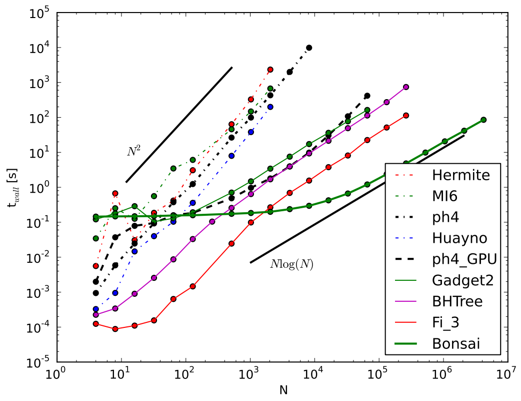


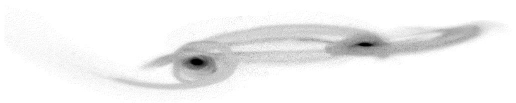
Figure 7.1: Performance comparison of a suite of N-body codes. These codes are included in the AMUSE software package. Visible are direct N-body codes that scale as $O(N^2)$ and hierarchical codes that scale as $O(Nlog(N))$. (Figure taken from (Portegies Zwart et al. 2013))

accuracy of direct methods and the speed of tree-codes. In the `BRIDGE` algorithm by Fujii et al. (2007) a direct N-body method is combined with a tree-code to integrate the evolution of star clusters (which requires direct N-body methods) embedded in their host galaxy (which requires an approximation method because of the large number of particles). This allows for detailed simulations in the area of interest while still being able to use large particle numbers. Since the method is based on two well known algorithms it is possible to use the methods presented in this thesis to accelerate `BRIDGE` with GPUs.

With simulation codes becoming more complex and containing more advanced features it becomes difficult to add new physics to existing codes without breaking other parts of the codes. This is a common problem in computational sciences and software development in general. Ideas often start off simple, but when something works you want to extend it, which complicates matters. In AMUSE (Portegies Zwart et al. (2009)) a different approach is taken. Codes that are written for different specific purposes are combined into one framework. This simplifies the development of the separate software products. The other advantage is that you can combine simulation codes that have support for GPUs with codes that do not and therefore still have the speed advantage of using GPUs. With AMUSE it is possible to use the same script using different simulation codes and thereby have the choice between speed, accuracy or available hardware. An example of this is shown in Fig. 7.1 where the execution speed of a set of N-body integration codes is demonstrated. The figure shows the results of 4 direct N-body codes (`Hermite, PhiGRAPE, Huayno and ph4`) and 3 tree-codes (`Gadget2, Octgrav, Bonsai`). Clearly visible is the difference in speed and scaling between the direct codes($\mathcal{O}(N^2)$ scaling) and the tree-codes ($\mathcal{O}(N \log N)$ scaling).

7.2 The future

With focus shifting to more complex methods and algorithms we see the advantage of the versatility of GPUs and the shift from fixed function methods in the early 90s (like the GRAPE) to programmable chips like GPUs. Even though Field Programmable Gate Arrays (FPGAs) have been around for decades, their programming is difficult and expensive,



certainly compared to chips that are programmable by software. It is much easier to develop and acquire chips like GPUs, since they can be bought in many consumer computer stores. The availability and price makes the GPU one of the most attractive high performance computing devices currently available. It is of course still possible to develop faster chips that require less energy if you make them dedicated, but the development cost and specialized knowledge to build a chip that is competitive against the multi-billion dollar gaming industry is higher than a university research team can afford (Makino and Daisaka (2012)).

Also, simulation algorithms become more advanced and incorporate different techniques to overcome, for example, the painful $\mathcal{O}(N^2)$ scaling. An example of this is the `Pikachu` code by Iwasawa et al. (in preparation). In the `BRIDGE` method one has to indicate which particles will be integrated using the direct algorithm and which particles with the tree-code algorithm when the initial conditions are created. The `Pikachu` code improves on this by dynamically deciding which particles can be integrated using a tree-code and which need direct N-body integration.

Even though approximation methods (tree-codes, FMM and Particle Mesh) are much faster than direct N-body methods, they do not reach the same level of accuracy. With the increase in computational power, direct N-body methods will always be used for new simulations with increasing N either to compare to previous results (e.g. performed with approximate methods) or for new science. The same is valid for the methods used to improve the performance of direct N-body simulations (block time-steps, neighbour schemes, etc.; see Section 1.3). These all have an influence on the precision. Although the difference is smaller than the difference between direct methods and approximation based methods it still might be of influence, especially considering the chaotic nature of the N-body problem (Miller (1964); Goodman et al. (1993)). Therefore, with the increased compute performance we will not only perform simulations with larger N, but also much more detailed simulations with relatively small N to validate previously obtained results. Simulations of globular clusters using high precision shared time-step algorithms are still far out of reach, but one day we will have the computational power to perform exactly this kind of simulation.

The increasing availability of GPUs in supercomputers and in small dedicated GPU clusters shows the potential, increased usage and the faith of researchers in GPUs over the last few years. And especially with the installation of GPUs in ordinary desktop computers, as is done, for example, at the Leiden Observatory, this computational power is available at everyone's fingertips without having to request time on expensive supercomputers.

However, as we demonstrated in Chapter 6, supercomputers are not obsolete. For many scientific questions we can increase the problem size indefinitely and by doing so we will run out of the available resources of our desktop computer and small scale clusters. At that point we have to transition to supercomputers. To make this transition as easy as possible for the user it is fundamental that supercomputers represent architectures that are popular in desktop and cluster-sized machines. This allows researchers to develop and optimize their single and multi-node implementations on their local hardware and then try to scale this up to thousands of nodes. This scaling is not trivial, but if you already have an optimized single-node implementation you only have to focus on the multi-node aspect of your code.