Abstract— This research paper aims at exploiting efficient ways of implementing the N-Body problem. The N-Body problem, in the field of physics, predicts the movements and planets and their gravitational interactions. In this paper, the efficient execution of heavy computational work through usage of different cores in CPU and GPU is looked into; achieved by integrating the OpenMP parallelization API and the Nvidia CUDA into the code. The paper also aims at performance analysis of various algorithms used to solve the same problem. This research not only aids as an alternative to complex simulations but also for bigger data that requires work distribution and computationally expensive procedures. Index Terms— N-Body, All-Pairs, Barnes-Hut, Parallelization, OpenMP, CUDA I. INTRODUCTION The N-Body problem, in physics, aims at predicting the individual motions of a group of celestial bodies interacting gravitationally [1]. Applications include un derstanding of the motion of the Sun, planets, visible stars etc. Simulations are important in anticipating certain behaviors in science, especially physics [2]. There is much room for improving the effectiveness of execution of these simulations by parallelizing the same with utilities such as OpenMP and Nvidia CUDA; involving the work distribu tion among N-processors (OpenMP) or GPUs (CUDA). The N-Body problem simulation is a relevant application to show the improvement in the simulation speed by evenly distributing the nodes (planets) through a finite number of processor cores. This report documents the development of All-Pairs and Barnes-Hut algorithms used to solve the N-Body problem. The process of creation of these algorithms in serial and then in parallel along with the problems associated with this process have been elucidated. Each algorithm is tested and the results are compared to draw conclusions of their strengths and weaknesses. The All Pairs algorithm is a naive approach to solving the N-Body problem, with O(N 2) runtime; involving the calculation of the force acting on every body with respect to every other body. The Barnes-Hut algorithm, on the other hand, optimizes the N-Body problem with O(N log(N)) runtime. In simple terms, the Barnes-Hut algorithm uses a quad tree. For 2D version of the algorithm, we recursively divide N-bodies into groups by storing them in a quad-tree [3]. The root of the tree represents a space cell with all the bodies in the system. The tree is built by adding particles to initially empty root cell, subdividing the cell into four children when it contains more than one body. As a result, the internal nodes of the tree have more than one body while the leaves are just single bodies. The tree is adaptive; it extends to more levels in regions with high particle densities. To compute the force, the center of mass is considered and an approximation based on the the ratio of the length of a side of the cell to the distance of the body from the center of mass of the cell; l D < θ, where θ is a constant that usually range from 0.5 to 1.2 [3]. This paper is structured as follows: Section II provides details on the literature survey. Section III explains the parallelization of the N-Body problem. In Section IV the details about experimental results have been provided; fol lowed by conclusions and references. Appendix provides the performance statistics of the Barnes-Hut algorithm in OpenMP on galactic datasets chosen from Princeton

University [4]; with number of bodies ranging from 5 to 30,002. II. LITERATURE SURVEY A. Background The N-Body problem is concerned with the interactions between celestial bodies [5], where every body in the given system of bodies is affected by every other body. The creation of galaxies, the effects of black holes and even the search for the dark matter are concerned with the N B