Parallelizing G4Lifetime/GrROOT Software for the Lifetime Group at NSCL/FRIB

Roy Salinas

October 27, 2021

CMSE 822
Parallel Computing
Date Performed: Fall 2021

Abstract

The structure of the things around us are macroscopically interesting, yielding discussions and endeavors that drive the incessant pursuit of knowledge. Supplementary, the microscopic nature of the universe, i.e. nuclei and their properties, sheds light on complex microcosms that the universe is built upon. The Lifetime group at the NSCL/FRIB focuses on a particular facet of discovery; spectroscopy of exotic nuclei far from stability via in-beam gamma and particle spectroscopy. The Lifetime group utilizes simulations to better understand the complexities seen in nuclear structure via experimental observables such as lifetimes of excited states, transition probabilities, and deformation parameters. The process in converting raw data to appropriate data structures, and performing simulations, require a considerable amount of time which renders the physicists to the will of the machine. I propose to ameliorate this situation by utilizing multi-threaded approaches to the current c++ simulation/calibrating software of the Lifetime group to reduce computation time.

1 Parallelization Strategies

The parallelization strategy that I will pursue would be that of relying less on the programmer (myself), and more on the compiler (i.e. built in functionalities) to assure there are barriers and "waits" in place, to prevent things such as data races, unintelligible error codes, and time efficiency. Software I intend to utilize for this consists of:

- OpenMP
- OpenMPS
- CUDA
- Single threads, futures, and utilization of mutex locks
- Hybrid parallelization
  - MPI with OpenMP
  - MPI with Cuda

The selection of software was made by considering the accessibility (and challenges) of parallelizing scripts in the simulation/calibration software, the edification facet of this project, and the efficacy of this endeavor to benefit the Lifetime group.

2 Benchmark and optimization

The benchmark in parallelizing the simulation/calibration software will be two fold: Calibration and Simulation.

2.1 Calibration

To preface the problem, the calibration process consists of the following:

i) Read in large amounts of raw data

ii) Process data and apply calibration

iii) Create output files with calibration on data

The bottle neck occurs in step ii), where the code is serialized and takes a significant amount of time to apply the calibration to all of the raw data. The serialized calibration for 1 raw data file will serve as the initial benchmark to compare with a parallelized version of the calibration software.

2.2 Simulation

Similar to the calibration process, the simulation consists of:

i) Set parameters for the kind of transition we are looking at

ii) Run simulation

iii) Create simulation file

The bottle neck occurs in step ii), where the code is serialized. A serialized simulation will serve as the initial benchmark to compare with the parallelized version of the simulation

It is worth noting that the parallelized/serialized code will not be run on my personal machine bur on the nscl servers (particularly fishtank). The benefit in running the software on the servers that NSCL/FRIB provides lies in the increased amount of CPUs. The comparison with the benchmark will serve to highlight the following faces on the simulation/calibration: time, memory usage, readability. The last comparison serves as an investment towards future improvements that will not be made by me, as this is the first time anyone in the group attempts to speed up these processes.