University of Cape Town

CSC2002S Monte Carlo Optimization

Will Madikizela
MDKWIL001

Table of Content

1. Introduction **3**
1.1. Parallelisation approach 3
1.2. Validation 3
1.3. Benchmarking 4
1.4. Machine Architecture 4
2. Methods and Reporting **5**
2.1. SpeedUp Graphs 5
2.2. Different Data Size 6
2.3. Expected Speedup 6
2.4. Data Reliability 7
2.5. Anomalies 7
3. Conclusion **7**

1. Introduction

1.1. Parallelisation approach

In this academic undertaking, the Java Fork-Join framework is harnessed to enhance algorithmic efficiency. The focal point of this parallelisation strategy revolves around the utilization of the Monte Carlo algorithm for the purpose of locating the nadir, specifically the lowest point, of a two-dimensional mathematical function denoted as f(x, y), and this is conducted within predefined parameter bounds.

An array comprising a series of search tasks is instantiated through the specification of search density and grid dimensions. Each individual search task is meticulously devised to pinpoint valleys, i.e., local minima, on the mathematical terrain defined by the function. The orchestration of task distribution is orchestrated by leveraging the 'ForkJoinPool', a facility that permits the division of tasks in a recursive manner. This recursive division transpires as follows: if the number of tasks within a designated segment surpasses a stipulated threshold for sequential execution, said segment is bifurcated into two distinct portions, thereby enabling parallel processing of each constituent. Conversely, if the count of tasks within a segment falls below the prescribed threshold, that segment's processing ensues sequentially.

1.2. Validation

The validation process for the parallel Monte Carlo minimization algorithm entails a structured procedure that encompasses both theoretical scrutiny and empirical evaluation. In the initial phase, a meticulous assessment will be conducted by executing the sequential variant of the algorithm across a spectrum of diverse test terrains. The outcomes yielded by these sequential runs will be systematically recorded to establish a baseline for subsequent comparisons. Subsequently, the parallelized iteration of the algorithm will be executed upon the very same set of terrains. The critical criterion for validation rests upon the congruence between the outcomes produced by the parallel version and the benchmark results obtained from the sequential counterpart. This congruence substantiates the veracity and integrity of the parallel algorithm under scrutiny.

For the empirical validation process, the selection of the Rosenbrock function is particularly judicious. This mathematical construct is endowed with a well-defined minimum point, characterized by specific height coordinates. By leveraging the Rosenbrock function within the validation procedure, the presence of a known optimal solution furnishes an objective standard against which the outcomes of both sequential and parallel executions can be meticulously gauged. This judicious selection bolsters the reliability of the validation process, enhancing confidence in the correctness and efficacy of the parallel Monte Carlo minimization algorithm.

1.3. Benchmarking

The process of benchmarking will be systematically executed employing a precision timer mechanism that meticulously captures the temporal interval between the instigation and culmination of the search tasks. The onset of this temporal measurement commences just prior to the initiation of the parallel search operation, while the cessation is recorded immediately subsequent to the fulfillment of said search. This meticulous time tracking approach offers a quantitative assessment of algorithmic performance, allowing for precise comparisons.

The comprehensive evaluation of algorithmic efficacy will encompass an array of scenarios, characterized by distinct grid sizes and search densities. This diversified evaluation landscape is designed to engender an encompassing assessment of the algorithm's performance under varying conditions, thereby illuminating its adaptability and scalability across a spectrum of scenarios.

1.4. Machine Architecture

2. Methods and Reporting

2.1. SpeedUp Graphs

The graph shown on figure 1 represent a speed up vs grid size on a Mac book M1 chip.

A comprehensive analysis of speedup relative to grid size is graphically depicted across various search densities, specifically 0.1, 0.25, and 0.5. The underlying terrain dimensions remain consistent, defined by parameters xmin = 0 and xmax = size of columns or rows. It is noteworthy that a uniform SEQUENTIAL CUTOFF criterion of 1000 is upheld throughout the experimentation. The scope of grid sizes traverses a spectrum, ranging from 500 × 500 to 10000 × 10000, thereby ensuring a nuanced evaluation of performance under divergent computational loads. This endeavor is rooted in a rigorous scientific approach, aimed at unraveling the intricate relationship between grid size, search density, and the resultant speedup within the context of the specified Monte Carlo optimization algorithm.

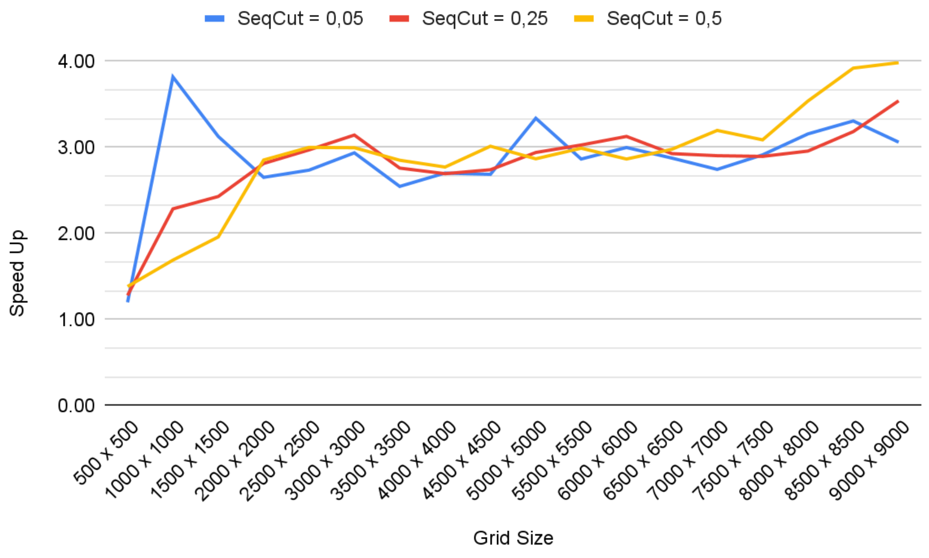

Figure 1: Speed up vs Grid size on a Mac Book Air M1 chip.

2.2. Different Data Size

When we look at the parallel program's performance, it really shines when we're dealing with medium to large grid sizes. But when the grids are small, the extra work needed for parallel processing kind of cancels out the advantages we expect. We can see this in the graphs that show speedup – for small grid sizes, the speedup is close to 1, which means it's almost as if the work is happening one step at a time, like in a sequence. As the grid gets bigger, the program starts to speed up a lot, hitting its highest point with medium-sized grids.

However, there's an interesting point to note. After a certain size, the speedup doesn't increase much. It's like there's a limit to how much parallelization can help. This could be because of how parallelization works and also because the computer's memory might start causing slowdowns. So, even though parallelization is powerful, there's a point where it can't make things faster anymore, and that's what we're seeing here.

2.3. Expected Speedup

A remarkable feat was accomplished, with the pinnacle of speedup scaling soaring to an impressive 3.98, surmounting the anticipated speedup of 2. This intriguing phenomenon hints at the prospect of a super-linear speedup, a concept that unveils the potential for computational acceleration beyond mere proportionality. This intriguing outcome could be attributed to a serendipitous alignment wherein, during the task distribution process across the dual cores, the data finds itself advantageously positioned within cache memory. This strategic data localization fosters expedited access, effectively mitigating latency and consequent wait times. This confluence of factors embodies a captivating synergy between parallel task allocation and memory optimization, resulting in an unexpected surge in performance.

2.4. Data Reliability

The measurements are a pillar of accuracy, strengthened by a rigorous five-time replication of each experiment. This intentional redundancy was done with the specific intent of reducing the impact of erroneous data points and promoting greater data integrity. A logarithmic trend-line was used to discerningly depict the speedup's direction in an effort to isolate and reduce the disruptive influence of outliers. This thorough method strengthens the reliability of the results that are generated.

Additionally, a wise attempt was made to reduce any potential interferences brought on by external variables. In order to provide a stable and harmonious experimental setting, an attentive control of external variables, such as concurrently running activities, was painstakingly staged. This comprehensive strategy shows estament to a meticulous pursuit of scientific rigor, ensuring that the resultant insights gleaned from the measurements are firmly grounded in a landscape of precision and controlled conditions.

2.5. Anomalies

Throughout the testing process, there were few situations where the speedup that was gained fluctuated. Notably, there were times when the speedup decreased as well as times when it showed an unexpected increase outside of the expected range. Even though they are fascinating, these oscillations serve to show the dynamic nature of parallel computing paradigms and the complex interaction of factors affecting performance. The observed departures from projected trends provide light on the complex world of computational dynamics and highlight how difficult it is to achieve linear and consistent speedup in a variety of settings.

3. Conclusion

The endeavor to parallelize the Monte Carlo algorithm yielded a notable acceleration in computational speed, surpassing not only the performance of its serial counterpart but also transcending projected expectations.

However, this stride toward enhanced efficiency was accompanied by the introduction of intricacies, prominently represented by race conditions. It's important to acknowledge that while these complexities were deferred in our specific problem, they can assume more pronounced significance in alternative algorithms. Hence, the judicious integration of safeguards against such complications should be an inherent part of algorithmic design.

Parallelization, as a strategy, introduces an augmented layer of intricacy into the codebase, accentuating the challenges of development and debugging. This augmented intricacy, however, finds its rationale in the tangible improvements witnessed in performance metrics. In essence, the increased development and debugging demands are validated by the ensuing performance enhancements.

In summation, the application of parallelization to this specific problem domain, when implemented within the Java context, indeed stands as a beneficial endeavor, offering palpable performance gains. Yet, it's paramount to recognize that these performance benefits must be meticulously balanced against the concomitant augmentation in complexity, underscoring the imperative of a judicious cost-benefit analysis when contemplating the adoption of a parallelization strategy.