Accelerate Markov Chain Monte Carlo on a CUDA-enabled GPU

Webster Bei Yijie, Junyu Liang, Yanming Lai

Abstract

Markov Chain Monte Carlo (MCMC) is a technique very commonly used in Bayesian inference. Markov Chain Monte Carlo can be used to generate samples from a given distribution, and it can also be used as an optimization algorithm that finds a good configuration (or a good value for the parameter to be optimized) for a given value function. However, for real-world problems, these methods become prohibitively expensive due to the sheer amount of computations that need to be done. Among the various MCMC algorithms, Metropolis-Hastings is one of the most commonly used. This Project focuses on expediting Metropolis-Hastings sampling by exploiting parallelism that exists in the algorithm with Graphics Processing Unit (GPU). We will divide and map MCMC computation process onto multiple processing cores to increase the throughput of samples generated from the Markov process.

Objectives

Metropolis-Hastings is one of the most widely used Markov Chain Monte Carlo algorithms that provides a framework for performing several different tasks. To list some, two common applications of the Metropolis-Hastings algorithm are Bayesian parameter estimation and data sampling according to an unnormalized distribution function. While Metropolis-Hastings for parameter estimation is far less parallelizable due to the sequential dependence between states, the Metropolis-Hastings that is purposed to generate data samples from an unnormalized distribution function, however, is perfectly parallelizable. Therefore, this project will mainly focus on this variant of the Metropolis-Hastings algorithm. Our objective is to implement a variant of Metropolis-Hastings in CUDA that is able to achieve high throughput.

Methods

The Algorithm

Given an unnormalized high dimensional distribution function f (v) (where v is a vector), we want to generate vectors (data samples) u such that the empirical distribution vectors u generated through this process resemble the given distribution function.

The Idea

Each state (data sample) will have certain probability to mutate into another state determined by the distribution function. The mutation and probability of the mutation being accepted as a new state are only dependent on the current state. Since each state can result in multiple independent mutations and there is no retro-dependence, the sampling process at each stage can be viewed as completely independent of each other. Therefore, multiple sampling process can be started independently.

Performance Metric

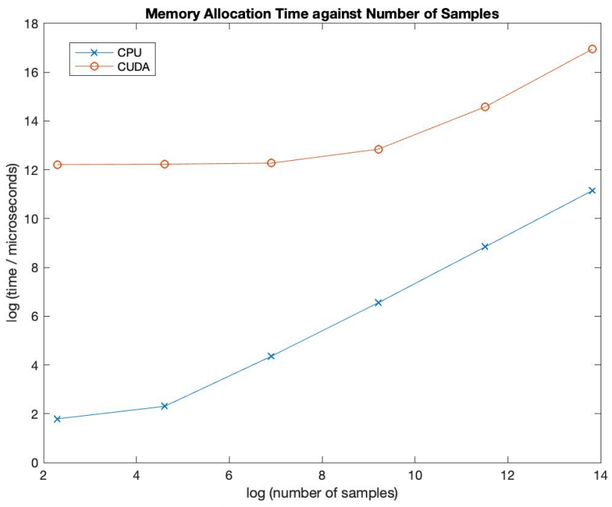
The performance of Metropolis-Hastings sampling algorithm on CPU and GPU will be compared using “number of data points generated per second” as the metric.

Other Factors

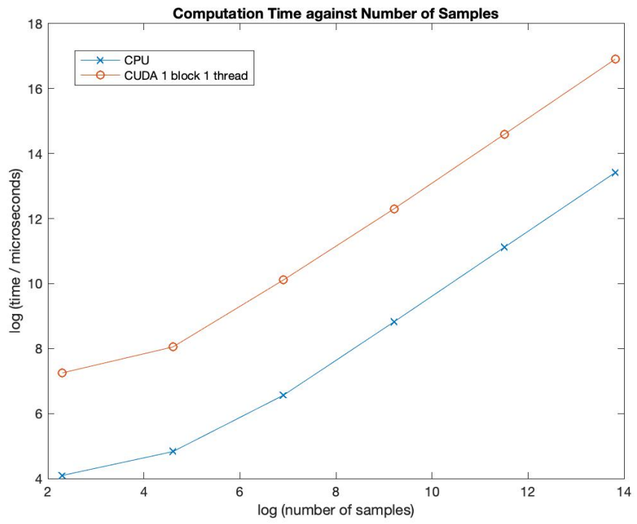
Resource utilization will be studied under different configurations (e.g cuda threads, block configurations). Bottleneck of the system will be analyzed.

Results

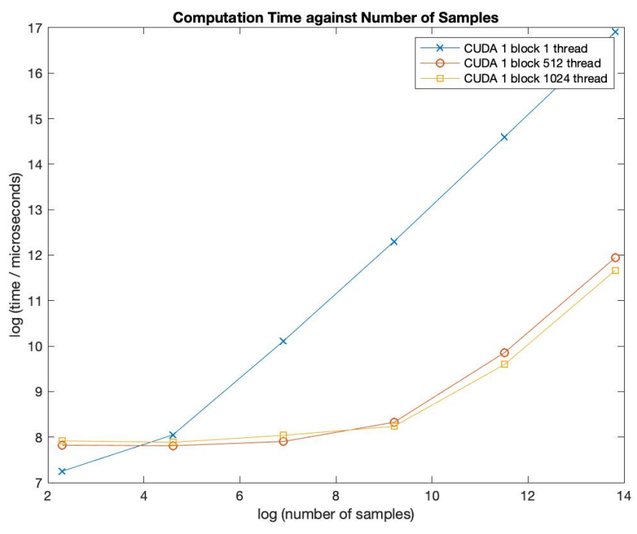
Shared memory allocation incurs significant overhead

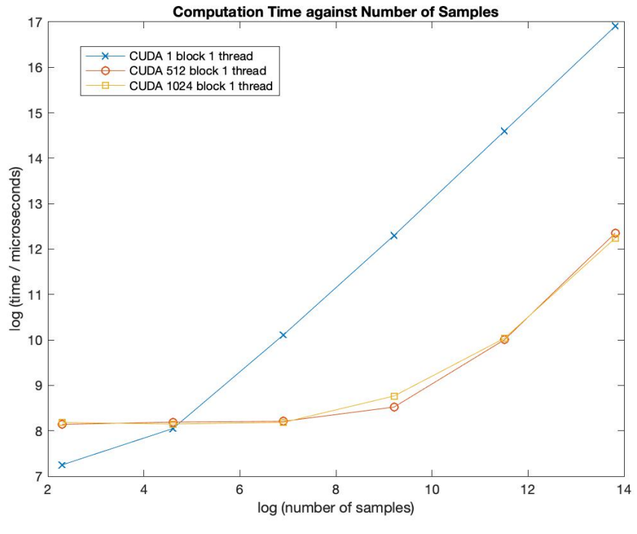


The time taken to perform the actual sampling process is significantly faster on CPU given that only one thread and one block is used on GPU.

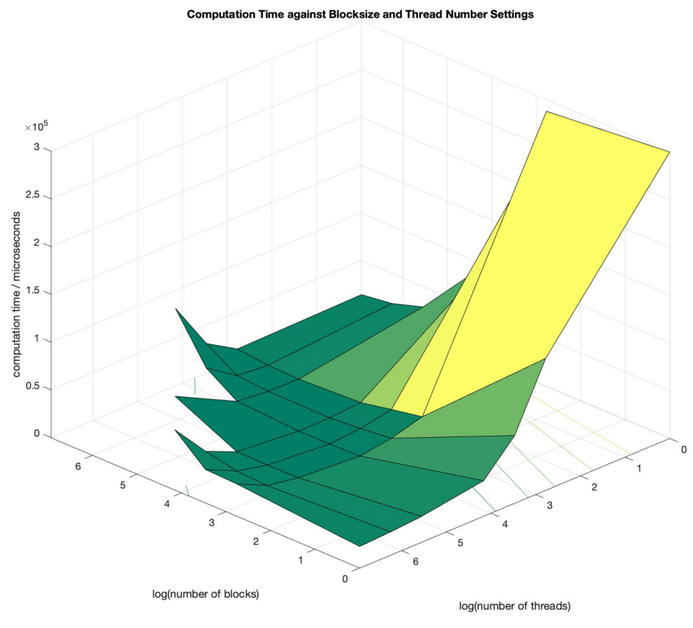


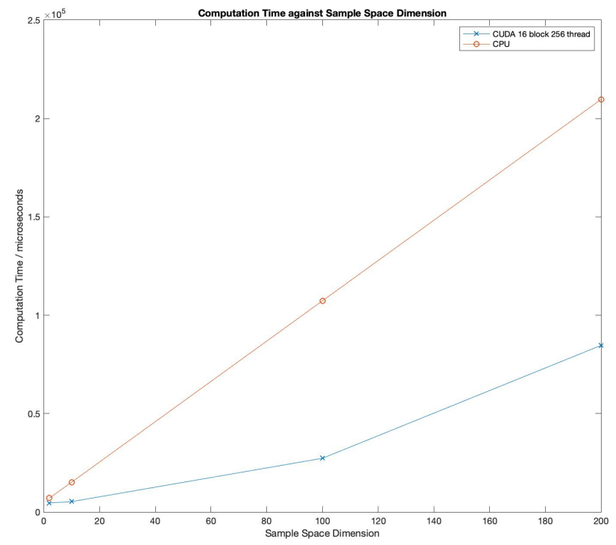
Neither increasing the number of blocks or increasing the number of threads alone achieve significant performance improvements beyond certain point.





Grid search for the optimal block size and number of threads configuration. When either setting is too large, the global shared memory access time becomes bottleneck.





Conclusion

Our experiments have shown that we can utilize parallelism at each MCMC iteration and speed up the process by orders of magnitude on a GPU than a CPU. However, it is important to note some trade-offs using parallelism and GPUs, such as overheads in memory allocation and data transfer. Also, as block size and concurrency increase, global memory access within the GPU becomes the bottleneck for performance. Despite some limitations, we argue that MCMC computations can be accelerated significantly with CUDA-enabled GPUs.

References

Craiu, R., Rosenthal, J., and Yang, C. (2009), “Learn From Thy Neighbour: Parallel-Chain and Regional Adaptive MCMC,” Journal of the American Statistical Association, 104 (408), 1454–1466. [617]

Jacob, P., C. Roberts, and M. Smith (2010). Using Parallel Computation to Improve Independent Metropolis-Hastings Based Estimation. Journal Of Computational And Graphical Statistics 20, 1–18.

Maclaurin, D., & Adams, R. P. (2014). Firefly Monte Carlo: Exact MCMC with subsets of data. In N. L. Zhang, & J. Tian (Eds.), Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014 (pp. 543-552).

Matias Quiroz, Robert Kohn, Mattias Villani & Minh-Ngoc Tran (2019) Speeding Up MCMC by Efficient Data Subsampling, Journal of the American Statistical Association.

Mykland, P., Tierney, L., and Yu, B. (1995), “Regeneration in Markov Chain Samplers,” Journal of the American Statistical Association, 90, 233–241. [617]

Robert, C., and Casella, G. (2004), Monte Carlo Statistical Methods (2nd ed.), New York: Springer-Verlag. [616, 618,619,623]