Acceleration of Diagrammatic Determinantal Quantum Monte Carlo Calculations using GPUs

Markus Schmitt[1,2], Iain Bethune[2], Patrick Haase[1], Thomas Pruschke[1]
[1]Georg-August-Universität Göttingen, Institut für Theoretische Physik
[2]The University of Edinburgh, EPCC

April 2 2014

Motivation

Diagrammatic Determinantal Quantum Monte Carlo (DDQMC) algorithms are used to solve quantum impurity models such as the Anderson model. The calculation of acceptance rates and observables during the Monte Carlo walk involves linear algebra operations whose computational expense increases with decreasing temperature. Thus, the lower boundary of the treatable temperature range is limited by the available compute capacity. In order to make use of GPUs as cheap and powerful accelerators parts of a DDQMC code (*CT-INT*, [GML+11]) were ported to CUDA [Sch13].
All performance numbers presented here were obtained using one NVIDIA C2050 card for the accelerated code and a 2.67GHz Intel Xeon processor for the serial parts.

DDQMC

The *CT-INT* algorithm [GML+11] is based on splitting the Hamiltonian into a non-interacting and an interacting part

$$H = H_0 + H_I$$

and expanding the partition function in a series of diagrams

$$Z = tr\left(e^{-\beta H}\right) = tr\left(e^{-\beta H_0} T_\tau \exp\left(-\int_0^\beta d\tau H_I(\tau)\right)\right)$$

$$= \sum_k \int_0^\beta .. \int_{\tau_{k-1}}^\beta \underbrace{i^{2k} tr\left(e^{-\beta H_0} H_I(\tau_k)..H_I(\tau_1)\right) d\tau_1..d\tau_k}_{w_c}$$

$$= \sum_c w_c$$

with configurations

$$c = (k, \tau_1, ..., \tau_k) \equiv (k, \vec{\tau}) \ .$$

Via Wick's theorem one obtains

$$w_c = Z_0 \frac{(-U)^k}{k!} \det M_k^\uparrow \det M_k^\downarrow d\tau_1 ... d\tau_k \ ,$$

where $Z_0 = Tr\left(e^{-\beta H_0}\right)$ and

$$(M_k^\sigma)_{ij} = G_\sigma^0(\tau_i - \tau_j)$$

with $G_\sigma^0(\tau)$ the non-interacting Green's function. The idea of DDQMC is now to stochastically sample the diagram series using a Metropolis-Hastings algorithm with updates

$$(k, \tau_1, ..., \tau_k) \to (k+1, \tau_1, ..., \tau_k, \tau_{k+1})$$
$$(k, \tau_1, ..., \tau_k) \to (k-1, \tau_1, ..., \tau_{l-1}, \tau_{l+1}, ..., \tau_k)$$

in the configuration space and acceptance rates

$$R = \alpha_k(\beta) \frac{\det M_{k+1}^\uparrow \det M_{k+1}^\downarrow}{\det M_k^\uparrow \det M_k^\downarrow} \ ,$$

which can be implemented computationally efficiently by the observation that

$$M_{k+1}^\sigma = \begin{pmatrix} M_k^\sigma & Q \\ R & s \end{pmatrix} \ \Rightarrow \ \frac{\det M_{k+1}^\sigma}{\det M_k^\sigma} = s - R(M_k^\sigma)^{-1} Q \ .$$

In order to obtain the interacting Green's function the observable

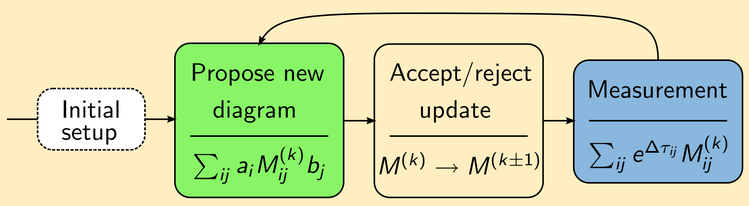
$$\tilde{G}_\sigma^c(\omega_n) = \sum_{ij} e^{i\omega_n(\tau_i - \tau_j)} \left((M_k^\sigma)^{-1}\right)_{ij}$$

is *measured* in the course of the MC calculation for $\mathcal{O}(10)$ frequencies ω_n.

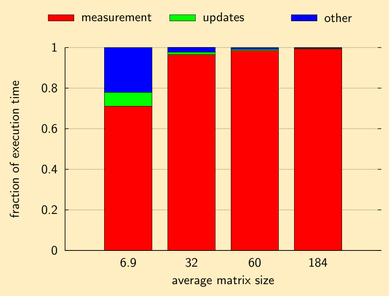
Remark: As the acceptance rate depends on the temperature via $\alpha_k(\beta)$, the average perturbation order or the average dimension of M occurring in the MC calculation are temperature dependent, respectively.

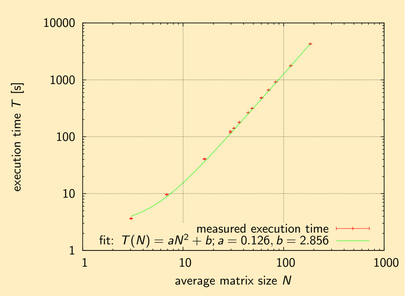
Performance of the serial code

The core of the code is a common MC loop:



Computational expense and scaling:





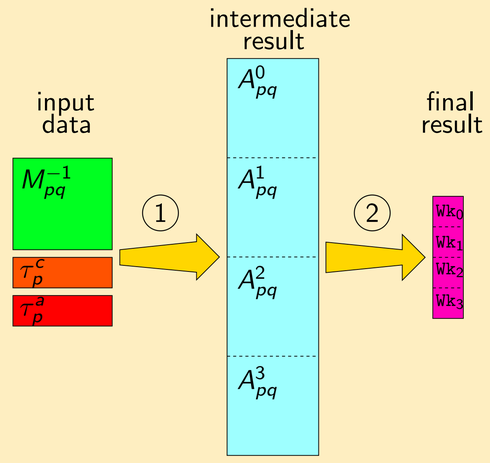
The compute time is almost completely spent on the measurement, which leads to quadratic scaling of the execution time with increasing average perturbation order (= average matrix size). Thus, one can expect considerable performance enhancements when this part of the code is efficiently parallelised.

CUDA kernels

For the measurement

$$Wk_n = \sum_{pq} e^{i\omega_n(\tau_p - \tau_q)} \left((M_k^\sigma)^{-1}\right)_{pq}$$

needs to be evaluated. This maps nicely onto the subsequent calling of two CUDA kernels , ① and ②:



① $A_{pq}^w = \texttt{expFun}(w, \tau_p - \tau_q) \times M_{pq}^{-1}$

② $\texttt{Wk}_w = \sum_{pq} A_{pq}^w$

Crucial for obtaining any speedup is at this point that the small amount of input data (M^{-1}) is multiplied on the device by a factor of $\mathcal{O}(10)$ such that the massive thread parallelism offered by the GPU can be made use of.

Memory transfer optimisation

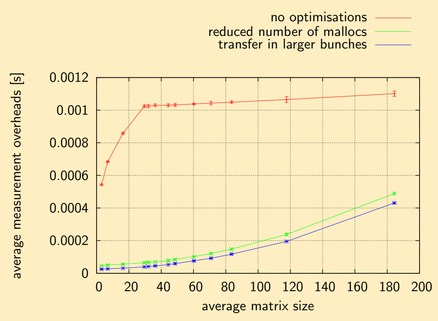
In the initial implementation device memory was allocated and freed before and after each measurement, respectively. Multiple little chunks of data were transferred for each measurement.
Profiling showed that

- 75% of the overheads are `cudaMalloc` or `cudaFree`
- only 12% of the available PCIe bandwidth is used in average.

Optimisations applied to improve the performance:

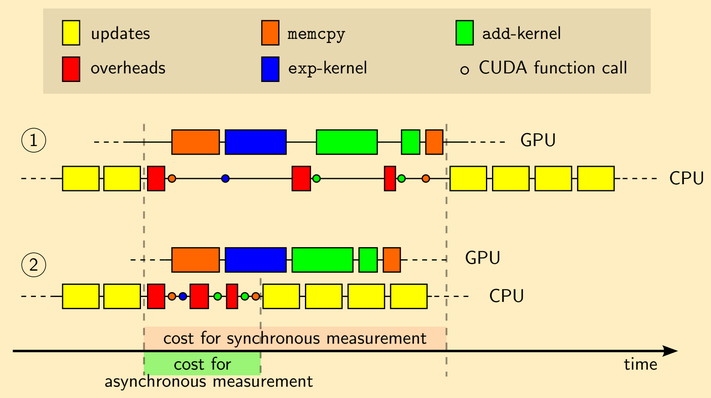
- Reduced number of malloc/frees (on device)
- Transfer data in larger chunks to increase bandwidth

Thereby performance was significantly improved:

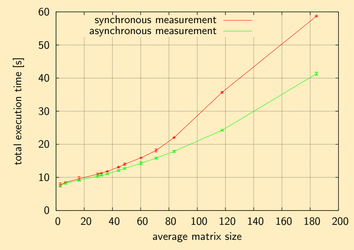


Asynchronous measurements

The updates are independent of preceding measurements. Thus, the measurement and subsequent updates can be performed in parallel on GPU and CPU, respectively.

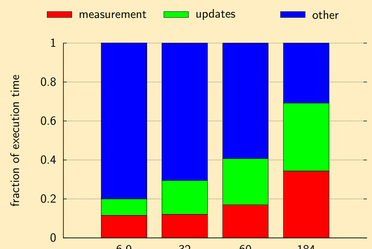


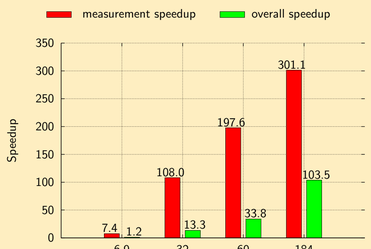
Asynchronous execution of the measurements cuts the total execution time by almost one third on the largest matrix size.



Results

- Shares in execution time and overall speedup





- Contributions from individual optimisations

optimisation	speedup
none (serial code)	1.00
kernel implementation	44.3
optimised kernels	1.19
optimised memory transfer	1.38
asynchronous measurement	1.42
aggregate	**103.5**

Summary

- The **measurement** of the DDQMC algorithm is **well suited for acceleration on GPUs**.
- **Avoiding redundant memory allocations** and **performing measurements asynchronously** improved the performance of the accelerated code considerably.
- An **overall speedup of 103.5**× was achieved on the whole code for the largest matrix size.

References

- NVIDIA CUDA. `https://developer.nvidia.com/category/zone/cuda-zone`.
- E. Gull, A.J. Millis, A.I. Lichtenstein, A.N. Rubtsov, M. Troyer, and P. Werner. Continuous-time Monte Carlo methods for quantum impurity models. *Rev. Mod. Phys.*, 83:349–404, May 2011.
- M. Schmitt. Diagrammatic Determinantal Quantum Monte Carlo Calculations on GPUs. Masters thesis, School of Physics and Astronomy, University of Edinburgh, 2013.