High-performance GPGPU OpenCL simulation of quantum Boltzmann equation

Petr F. Kartsev

NRNU MEPhI (Moscow Engineering-Physics Institute)
115409, Kashirskoe sh. 31, Moscow, Russia

Institute of Laser and Plasma technologies (LaPlas)

Dept. 70 “Physics of Solid State and Nanosystems”

IWOCL / SYCLcon 2020 DIGITAL, April 27-29

Outline of my talk

1) *Fast processes in physics*
2) *What is Quantum Boltzmann Equation*
3) *Specific physical problem*
4) *Problem analysis*
5) *Development of OpenCL solver and optimizations*
6) *Examples of solver application*
7) *Performance benchmarks and discussion*

High-performance GPGPU OpenCL simulation of quantum Boltzmann equation
(Petr F. Kartsev, NRNU MEPhI)

The work is supported by the Russian Foundation for Basic Research, Grant No. 17-29-10024.

The topic of this work is the numerical simulation of fast processes in solid state physics.

Physical problems with fast processes are, for example:

- photoinduced electrons and holes in semiconductors;
- optical field in laser physics;
- Quantum state of superconductor and Bose-Einstein condensation.

High-performance GPGPU OpenCL simulation of quantum Boltzmann equation
(Petr F. Kartsev, NRNU MEPhI)

Solid state is quantum system.

Standard approach to simulate fast processes in quantum systems is **kinetic equations**.

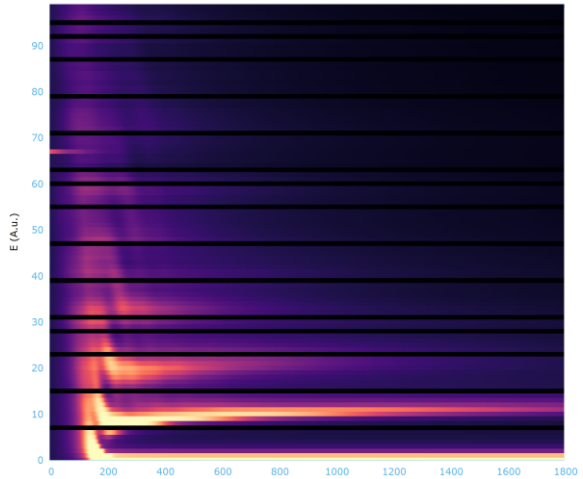
Example:

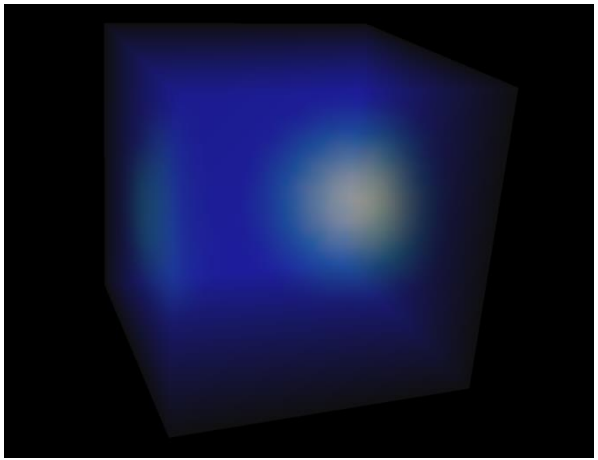
$$\frac{dn_1}{dt} = \frac{2\pi}{\hbar} U_0^2 \sum_{234} [(n_1 + 1)(n_2 + 1 + \delta_{12}) n_3 (n_4 - \delta_{34}) -$$
$$- n_1 (n_2 - \delta_{12})(n_3 + 1)(n_4 + 1 + \delta_{34})] \delta_{\varepsilon_1 + \varepsilon_2, \varepsilon_3 + \varepsilon_4} \delta_{1+2, 3+4},$$

From [*P F Kartsev and Kuznetsov I O. 2017. Simulation of the weakly interacting Bose gas relaxation for cases of various interaction types. Journal of Physics: Conference Series 936 (2017), 012055.* https://doi.org/10.1088/1742-6596/936/1/012055] also presented at IWOCL'2017

High-performance GPGPU OpenCL simulation of quantum Boltzmann equation
(Petr F. Kartsev, NRNU MEPhI)

We use kinetic equations to study relaxation processes … two typical pictures:





- [*P F Kartsev. 2017. Effective simulation of kinetic equations for bosonic system with two-particle interaction using OpenCL. In Proceedings of IWOCL'17, Toronto, Canada, May 16-18, 2017. ACM Press.* *https://doi.org/10.1145/3078155.3078185*]
- [*P F Kartsev and Kuznetsov I O. 2017. Simulation of the weakly interacting Bose gas relaxation for cases of various interaction types. Journal of Physics: Conference Series 936 (2017), 012055.* *https://doi.org/10.1088/1742-6596/936/1/012055*]

But sometimes kinetic equations are not enough

They neglect higher-order correlations

for example $\langle n_1 n_2 \rangle \neq \langle n_1 \rangle \langle n_2 \rangle$

(Usually, the average of the product is not the product of averages)

but these correlators can be essential

for some complex phenomena under study.

“We need to go deeper”

Quantum Boltzmann equation (QBE)

QBE is the universal approach of theoretical physics to describe the behavior of complex many-particle quantum system. It is based on the differential equation for the so-called `density matrix'.

[see for example *I. A. Shelykh et al., Physical Review B* ***76*** *(2007), 155308, https://doi.org/10.1103/PhysRevB.76.155308*]

$$i\hbar \frac{d\rho}{dt} = [H;\rho]$$

It generates the infinite chain of interconnected time-dependent differential equations for particle numbers and various correlators of increasing order.

Limiting the maximal order of correlators, we can cut the chain of equations and arrive to closed system of equations which can be solved.

This work: weakly-interacting Bose gas

[1] *I. A. Shelykh et al., Physical Review B* ***76*** *(2007), 155308,* https://doi.org/10.1103/PhysRevB.76.155308

We solve the QBE for the weakly-interacting Bose gas formulated in [1] :

$$\frac{dN_{\mathbf{k}}}{dt} = -\frac{1}{\hbar}\mathrm{Im}\sum_{k',q} V_{\mathbf{k},\mathbf{k}',\mathbf{q}} A_{\mathbf{k},\mathbf{k}',\mathbf{q}}, \tag{1}$$

$$\frac{dA_{\mathbf{k},\mathbf{k}',\mathbf{q}}}{dt} = \frac{i}{\hbar}(\varepsilon_{\mathbf{k}'+\mathbf{q}} + \varepsilon_{\mathbf{k}-\mathbf{q}} - \varepsilon_{\mathbf{k}} - \varepsilon_{\mathbf{k}'})A_{\mathbf{k},\mathbf{k}',\mathbf{q}} + F_1 + F_2 + F_3, \tag{2}$$

where $\mathbf{k}$ is the momentum of a particle, $\hbar$ is Planck constant (later we take $\hbar = 1$), $N_{\mathbf{k}}$ and $A_{\mathbf{k},\mathbf{k}',\mathbf{q}}$ are particle numbers (one index) and two-particle correlators (three indices), $\varepsilon_{\mathbf{k}}$ are particle energies and $V_{\mathbf{k},\mathbf{k}',\mathbf{q}}$ are interparticle interaction coefficients.

Equations, pt.2

Terms $F_1..F_3$ in equations:

From [*I. A. Shelykh et al., Physical Review B* ***76*** *(2007), 155308,* *DOI 10.1103/PhysRevB.76.155308*]

the terms depend again on our variables N, A so we have a closed system of equations.

Many variables and many equations.

They should be solved numerically.

$$F_1 \simeq \frac{i}{\hbar} A_{\mathrm{k},\mathrm{k}',\mathrm{q}} \sum_{\mathrm{k}''} \left[V_{\mathrm{k}-\mathrm{q},\mathrm{k}'',0} + V_{\mathrm{k}'+\mathrm{q},\mathrm{k}'',0} - V_{\mathrm{k},\mathrm{k}'',0} - V_{\mathrm{k}',\mathrm{k}'',0} \right] N_{\mathrm{k}''}$$

$$F_2 \simeq \frac{i}{\hbar} V_{\mathrm{k},\mathrm{k}',\mathrm{q}} [N_{\mathrm{k}} N_{\mathrm{k}'} (N_{\mathrm{k}-\mathrm{q}} + N_{\mathrm{k}'+\mathrm{q}} + 1) - N_{\mathrm{k}-\mathrm{q}} N_{\mathrm{k}'+\mathrm{q}} (N_{\mathrm{k}} + N_{\mathrm{k}'} + 1)],$$

$$F_3 \simeq \frac{i}{\hbar} \sum_{\mathrm{q}'} V_{\mathrm{k},\mathrm{k}',\mathrm{q}'} \left[(N_{\mathrm{k}-\mathrm{q}} + N_{\mathrm{k}'+\mathrm{q}} + 1) A_{\mathrm{k},\mathrm{k}',\mathrm{q}-\mathrm{q}'} - (N_{\mathrm{k}} + N_{\mathrm{k}'} + 1) A_{\mathrm{k}-\mathrm{q}',\mathrm{k}'+\mathrm{q}',\mathrm{q}-\mathrm{q}'} \right]$$
$$+ \frac{i}{\hbar} \Big[\sum_{\mathrm{k}'' \neq \mathrm{k}'+\mathrm{q}} V_{\mathrm{k}'',\mathrm{k}-\mathrm{q},\mathrm{q}} N_{\mathrm{k}} A_{\mathrm{k}'',\mathrm{k}',\mathrm{q}} + \sum_{\mathrm{k}'' \neq \mathrm{k}-\mathrm{q}} V_{\mathrm{k}'+\mathrm{q},\mathrm{k}'',\mathrm{q}} N_{\mathrm{k}'} A_{\mathrm{k}',\mathrm{k}'',\mathrm{q}} - \sum_{\mathrm{k}'' \neq \mathrm{k}'} V_{\mathrm{k},\mathrm{k}'',\mathrm{q}} N_{\mathrm{k}-\mathrm{q}} A_{\mathrm{k}''+\mathrm{q},\mathrm{k}',\mathrm{q}} - \sum_{\mathrm{k}'' \neq \mathrm{k}} V_{\mathrm{k}'',\mathrm{k}',\mathrm{q}} N_{\mathrm{k}'+\mathrm{q}} A_{\mathrm{k},\mathrm{k}''-\mathrm{q},\mathrm{q}} \Big].$$

Main obstacle:

huge amount of variables and calculations

For example, for lattice L x L x L (3D) or L x L (2D)

- N_k : array size $V=L^3$ (3D) or L^2 (2D),
- $A_{kk'q}$: array size $V^3=L^9$ (3D) or L^6 (2D)

Amount of calculations: $\sim V^4=L^{12}$ (3D) or L^8 (2D)

8 x 8 x 8: two 2GB arrays and 1.6×10^{12} FLOPs per step

10 x 10 x 10 : two 15GB arrays and 3.2×10^{13} FLOPs per step

And it is only for one step, while usually 10^4 and more needed.

So this simulation is extremely slow and practically impossible!

This work:
we are trying to make it possible.

see Proceedings:

Petr F. Kartsev. 2020. High-performance GPGPU OpenCL simulation of quantum Boltzmann equation. In *International Workshop on OpenCL (IWOCL '20), April 27–29, 2020, Munich, Germany*. ACM, New York, NY, US, 2 pages. https://doi.org/10.1145/3388333.3388664

Lets’ see what we can do!

0. Actually, **smaller systems are not bad and also can be useful**:

we can simulate smaller systems

and then extrapolate the result to larger size :

L=4 -> 6 -> 8 -> …

Main steps to improve the performance

1. We should modify the equations:

Choosing a simpler interaction model
(keeping the physics intact):

$V_{kk'q} = V_0 = const$

a) gives ***F_1=0,***

b) ***F_3*** *can be calculated without summation:*
these sums can be calculated beforehand (~V^3)

$$\mathrm{Asum}^{(0)}_{\mathrm{k}',\mathrm{q}} \equiv \sum_{\mathrm{k}} A_{\mathrm{k},\mathrm{k}',\mathrm{q}},\ \mathrm{Asum}^{(1)}_{\mathrm{k},\mathrm{q}} \equiv \sum_{\mathrm{k}'} A_{\mathrm{k},\mathrm{k}',\mathrm{q}},$$

I.e. we lowered amount of calculations from ~V^4 to ~V^3

- several orders lower. Much better scaling!

$$F_1 \simeq \frac{i}{\hbar} A_{\mathrm{k},\mathrm{k}',\mathrm{q}} \sum_{\mathrm{k}''} \left[V_{\mathrm{k}-\mathrm{q},\mathrm{k}'',0} + V_{\mathrm{k}'+\mathrm{q},\mathrm{k}'',0} - V_{\mathrm{k},\mathrm{k}'',0} - V_{\mathrm{k}',\mathrm{k}'',0} \right] N_{\mathrm{k}''}$$

$$F_2 \simeq \frac{i}{\hbar} V_{\mathrm{k},\mathrm{k}',\mathrm{q}} [N_{\mathrm{k}} N_{\mathrm{k}'} (N_{\mathrm{k}-\mathrm{q}} + N_{\mathrm{k}'+\mathrm{q}} + 1) - N_{\mathrm{k}-\mathrm{q}} N_{\mathrm{k}'+\mathrm{q}} (N_{\mathrm{k}} + N_{\mathrm{k}'} + 1)],$$

$$F_3 \simeq \frac{i}{\hbar} \sum_{\mathrm{q}'} V_{\mathrm{k},\mathrm{k}',\mathrm{q}'} \left[(N_{\mathrm{k}-\mathrm{q}} + N_{\mathrm{k}'+\mathrm{q}} + 1) A_{\mathrm{k},\mathrm{k}',\mathrm{q}-\mathrm{q}'} - (N_{\mathrm{k}} + N_{\mathrm{k}'} + 1) A_{\mathrm{k}-\mathrm{q}',\mathrm{k}'+\mathrm{q}',\mathrm{q}-\mathrm{q}'} \right]$$
$$+ \frac{i}{\hbar} \Big[\sum_{\mathrm{k}'' \neq \mathrm{k}'+\mathrm{q}} V_{\mathrm{k}'',\mathrm{k}-\mathrm{q},\mathrm{q}} N_{\mathrm{k}} A_{\mathrm{k}'',\mathrm{k}',\mathrm{q}} + \sum_{\mathrm{k}'' \neq \mathrm{k}-\mathrm{q}} V_{\mathrm{k}'+\mathrm{q},\mathrm{k}'',\mathrm{q}} N_{\mathrm{k}'} A_{\mathrm{k}',\mathrm{k}'',\mathrm{q}} - \sum_{\mathrm{k}'' \neq \mathrm{k}'} V_{\mathrm{k},\mathrm{k}'',\mathrm{q}} N_{\mathrm{k}-\mathrm{q}} A_{\mathrm{k}''+\mathrm{q},\mathrm{k}',\mathrm{q}} - \sum_{\mathrm{k}'' \neq \mathrm{k}} V_{\mathrm{k}'',\mathrm{k}',\mathrm{q}} N_{\mathrm{k}'+\mathrm{q}} A_{\mathrm{k},\mathrm{k}''-\mathrm{q},\mathrm{q}} \Big].$$

Several steps to improve performance

1. We modified the equations,

lowered amount of calculations from ~V^4 to ~V^3

2. Now it's time to develop the program…

Several steps to improve performance

1. Lowered amount of calculations from ~ $32V^4$ to ~ $20V^3$

2. Develop the program using **OpenCL for GPU accelerator:**

- GPU has more TFLOPS than CPU,
- higher memory bandwidth
- fast local memory, *etc…*

Calculations are repeated many times,
so we use the Production-CL library (IWOCL'2017)

P F Kartsev. 2017. Production-CL library for iterative scientific calculations. In Proceedings of IWOCL'17, Toronto, Canada, May 16-18, 2017. ACM Press. https://doi.org/10.1145/3078155.3078162

Several steps to improve performance

1. Modified equations:

We lowered amount of calculations from ~V^4 to ~V^3

2. Applied GPGPU / OpenCL

3. **What about OpenCL optimizations?**

Development of OpenCL solver

1. Modified equations

2. Applied GPGPU / OpenCL

3. **We use standard OpenCL optimizations:**

- reduction in local memory,
- arrays with staggered offsets to avoid bank conflicts,
- choose reasonable grid dimensions.

Development of OpenCL solver

1. Modified equations

2. Applied GPGPU / OpenCL

3. OpenCL optimizations

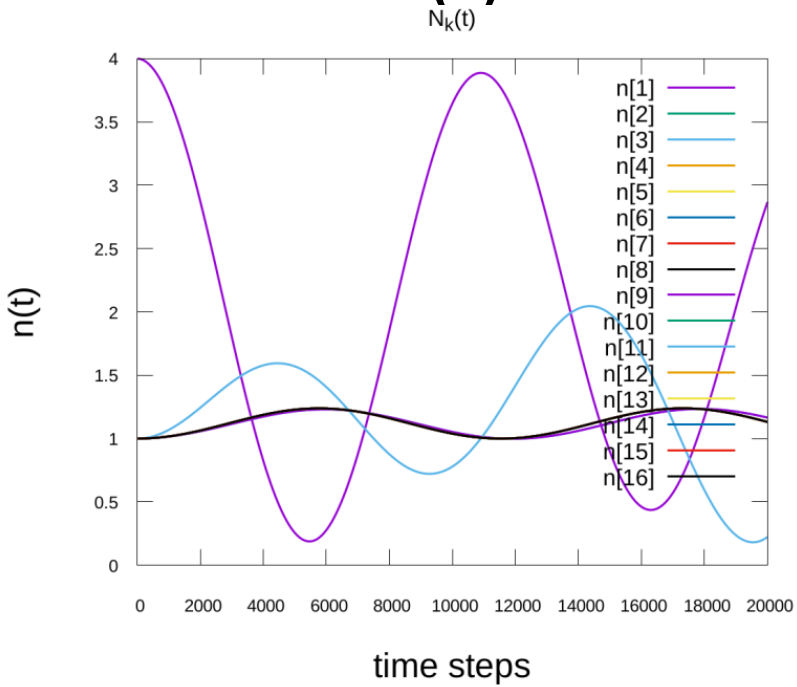
4. Testing?

5. Benchmarks

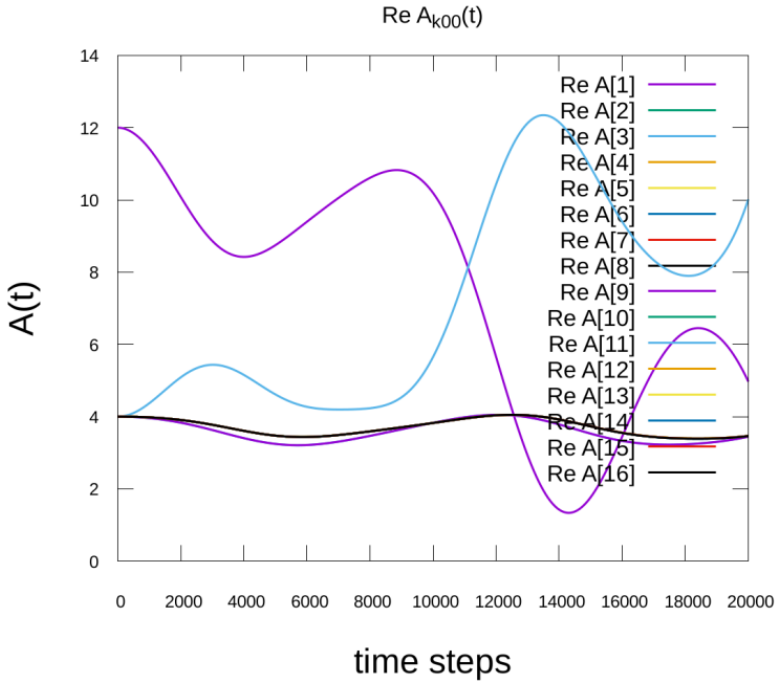
Testing: example N1

4x4 2D Bose system with $E_k=0$, $N_k=1$, $N_0=4$:

N(t)



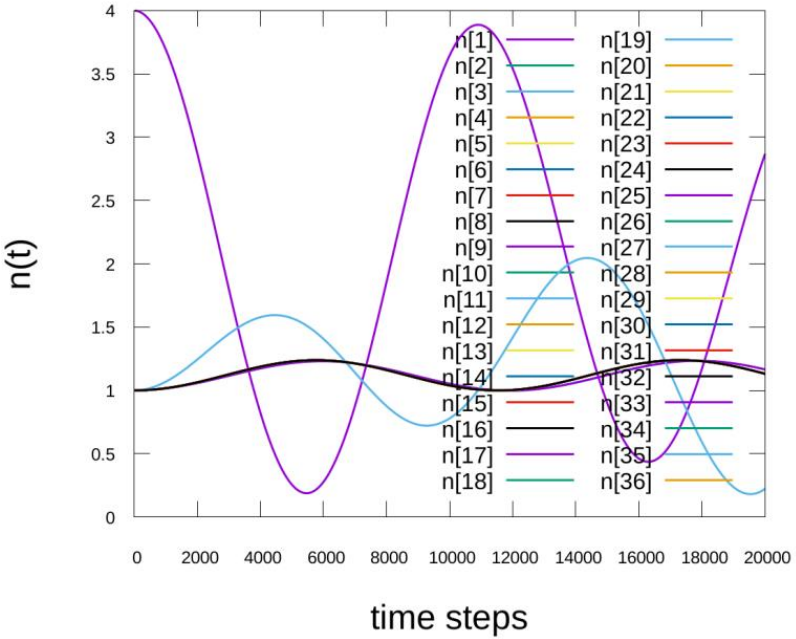
Re A(t)



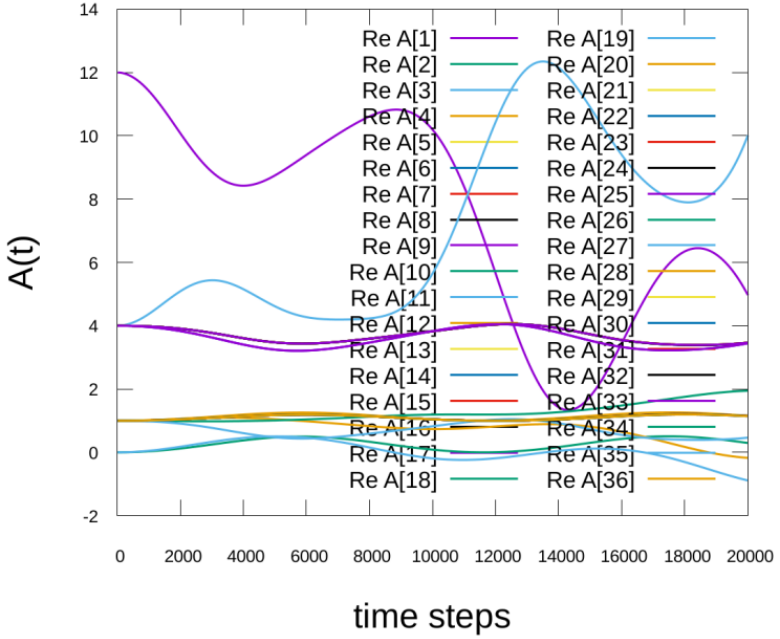
Testing: example N2

6x6 2D Bose system with $E_k=0$, $N_k=1$, $N_0=4$:

N(t)



Re A(t)



5. And finally, benchmarks

Performance benchmarks

We measure time needed to make single Euler step
for the system of differential equations

$$d\mathbf{N}/dt = R1(\mathbf{N},\mathbf{A})$$
$$d\mathbf{A}/dt = R2(\mathbf{N},\mathbf{A})$$

Lower is better, usable values should be lower than approx. 1 second

We checked OpenCL @CPU vs GPU, different generations,
and also did serial Fortran-90 implementation

What we expect:

1) GPU speed-up over CPU: how much? 20x ? 100x ? will see.
2) The problem is memory-bound, which means that the effect of GPU architecture and generation should be insignificant , and maybe even the number of CPU cores.
3) The effects of runtime overheads and optimizations for specific accelerators.

Details of test systems and accelerators

GPU:

- NVidia GTX 1080 Ti
- Nvidia GTX Titan Black
- AMD Radeon HD Fury X
- AMD Radeon HD 7970

OS:

- Windows 7 (x86_64)
- Debian Linux 10 (x86_64)

OpenCL runtimes:

- Intel, Nvidia CUDA, AMD, POCL

CPU:

- Intel i7-4790 (4 cores, 8 threads), 32 GB RAM
- AMD Threadripper 1950X (16 cores, 32 threads), 64 GB RAM

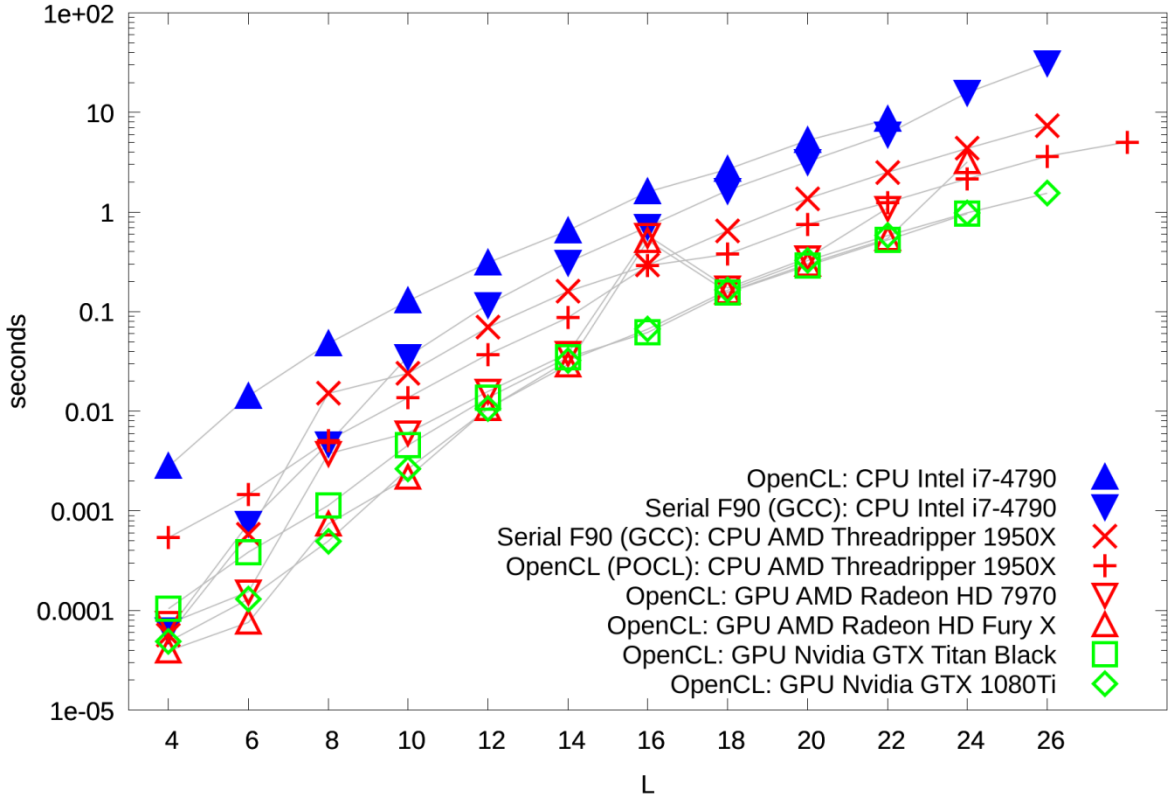
Serial CPU code for comparison: Fortran90,

compiler: **Gfortran 8.3.0**,

compilation keys:

-O3 -march=native -mavx

Benchmark: performance for 2D problem

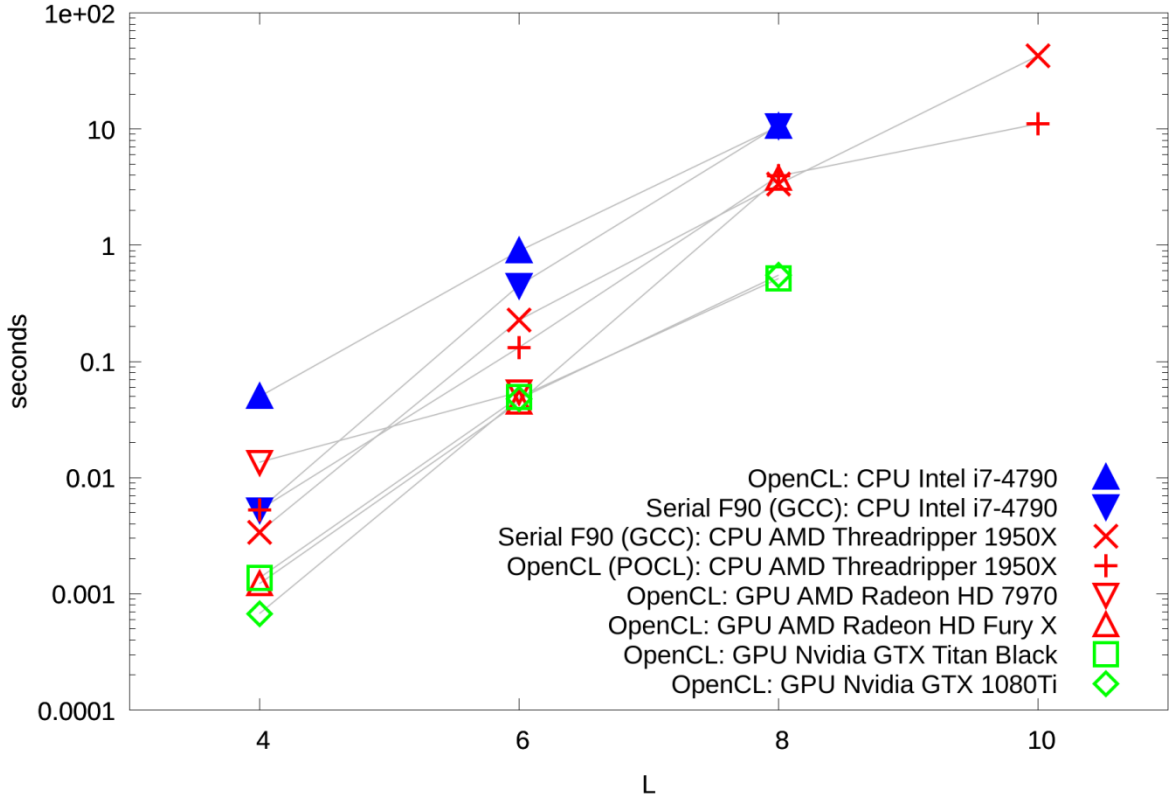


Time in seconds,
logarithmic scale,
lower is better.
Graphs are not very different!

1) GPU: ~30x speed-up
2) Times for all GPUs are mostly the same
3) All sizes fitting RAM (up to L=26) are OK (calc. time not exceeding 1 second)

Note: L=28 with 16+ GB RAM was possible only on CPU: F90 or POCL runtime

Benchmark: performance for 3D problem



Time in seconds, logarithmic scale, lower is better.

1) GPU: ~30x speed-up
2) All sizes fitting RAM (up to L=8) are OK (calc. time not exceeding 1 second)

Note 1: L=10 with 30+ GB RAM was possible only on CPU: F90 or POCL runtime

Note 2: On CPU, OpenCL code can be slower than serial F90 version – probably due to large runtime overheads and insufficient optimization

Results

- We developed the GPGPU OpenCL solver for Quantum Boltzmann Equation (QBE) able to simulate bosons on finite lattice.
- The solver performance allows us to simulate large enough systems of sizes up to 8x8x8 and 26x26, with sub-second time for single calculation step, which is good enough for practical study.
- Optimizations include not only OpenCL-specific tricks but also modification of the initial mathematical problem.
- The solver will be used in our research to study fast processes in various non-equilibrium quantum systems.

Thanks for Your attention!

Questions:
PFKartsev@mephi.ru

Study in MEPHI:
(Moscow Engineering-Physics Institute)
HPC, numerical simulation, laser physics, solid state physics, theoretical physics *etc.*

www: https://eng.mephi.ru

also https://studyinrussia.ru/en/study-in-russia/universities/mephi/