QCDGPU: an open-source OpenCL tool for Monte Carlo lattice simulations on heterogeneous GPU cluster

V. I. Demchik, N. V. Kolomoyets

Dnepropetrovsk National University
Dnepropetrovsk, Ukraine

GPUinHEP'14

September 10, 2014

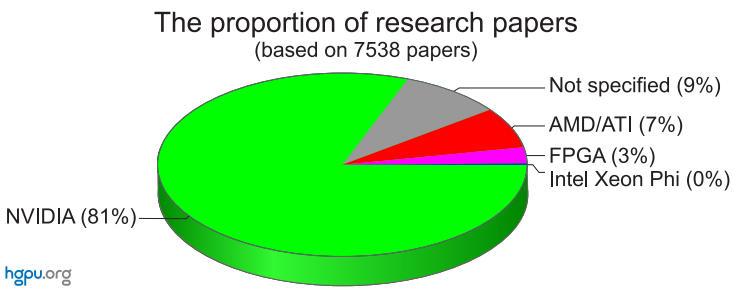
QCDGPU is a multi-GPU open-source package for lattice Monte Carlo simulations of pure SU(N) gluodynamics in external chromomagnetic field at finite temperature and O(N) model.

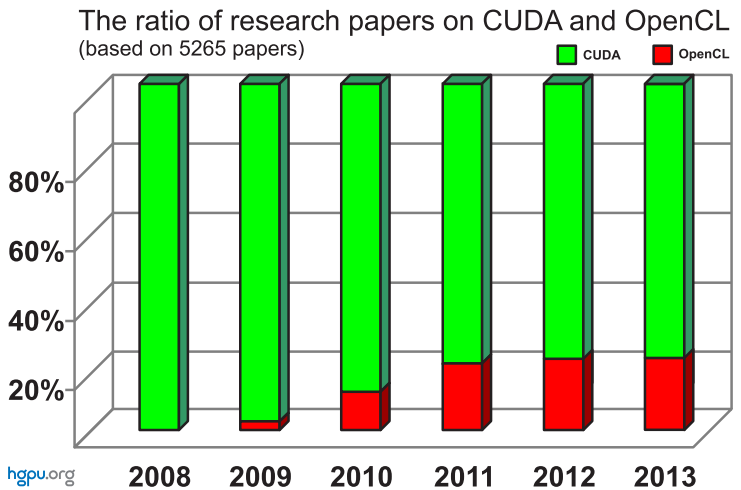
The code is implemented in OpenCL, tested on AMD and NVIDIA GPUs, AMD and Intel CPUs and may run on other OpenCL-compatible devices.

The package contains minimal external library dependencies and is OS platform-independent.

Why OpenCL?

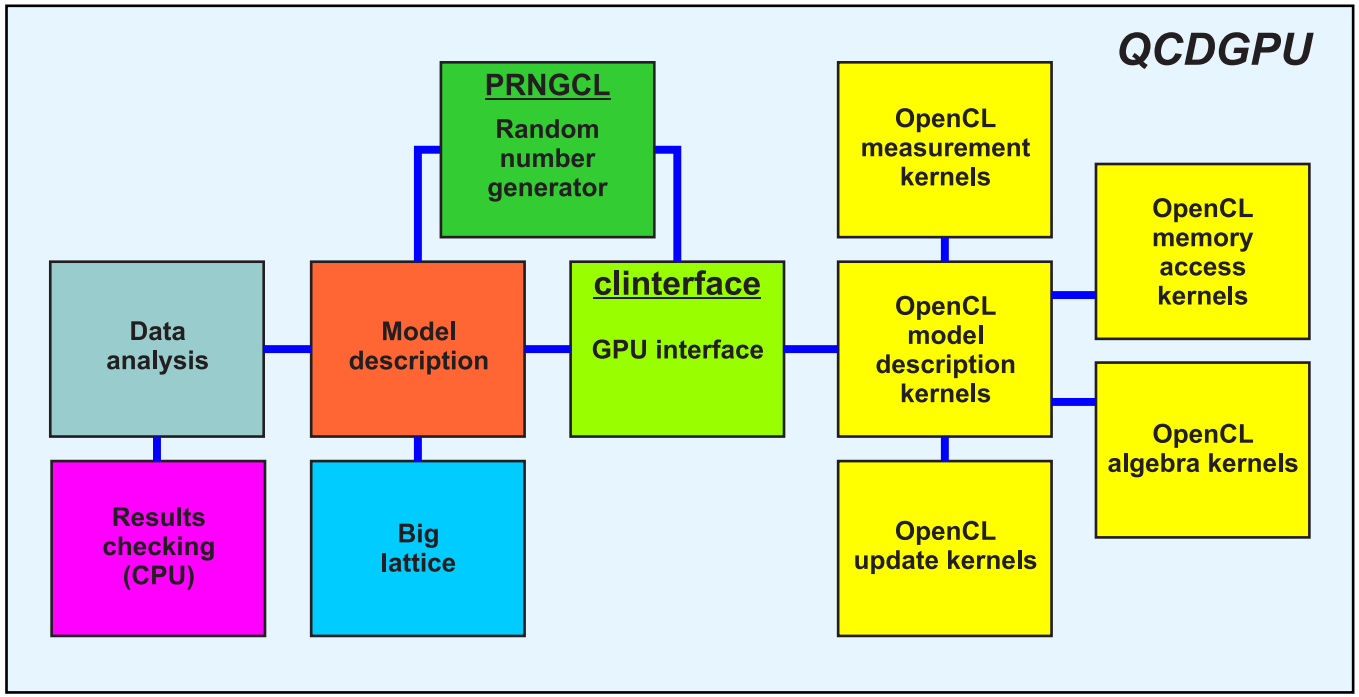
main reason - portability





Currently $\sim 20\%$ of all scientific researches using GPU accelerators are performed with OpenCL

Package structure



1. CLInterface module
2. Model description module
3. PRNGCL module
4. Big lattice module
5. CPU results checking module

CLInterface

Provides the interaction of host code with computing devices

- Device initialization/finalization
- Programs handling
  - .bin, .inf-files
  - MD5-hash
- Kernels handling
- Buffers handling
- Profiling
- Device errors control

Provide simulation of SU(N) and O(N) models

$$\texttt{index} = y + z \cdot N_2 + t \cdot N_2 N_3 + x \cdot N_2 N_3 N_4$$

SU(2) matrix:

$$m = a_0 I + i a_i \sigma_i$$

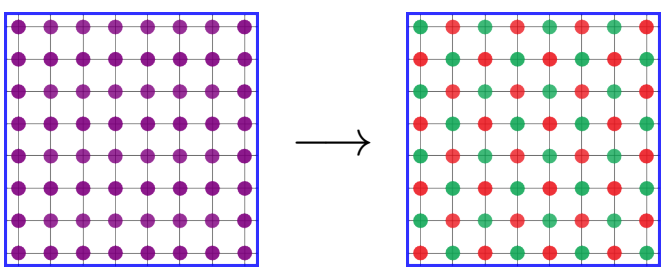
```
m.uv1 = (a0, a3, a2, a1);
```

O(N) model:

```
v.uv1 = (hgpu_double) a;
```

SU(3) matrix:

$$m = \begin{pmatrix} \vec{u} \\ \vec{v} \\ \vec{w} \end{pmatrix}, \quad \vec{w} = \vec{u}^* \times \vec{v}^*$$



Pseudo heat bath algorithm

Metropolis algorithm

Pseudorandom numbers

Module for PRNs production is based on PRNGCL library.
Generators available:

1. RANLUX
2. RANMAR
3. RANECU
4. XOR7, XOR128
5. PM
6. MRG32k3a

Random seeding for initialization – various PRNG threads are initialized by different seed tables.

```
PRNG = RANLUX3                    RANDSERIES = 12
```

`RANDSERIES = 0` corresponds to system time.

Single/double precision PRNGs.

`https://github.com/vadimdi/PRNGCL`

Performs simulation with predefined numbers instead of PRNs

```
  if(GID < SITES)
  {
    f1 = sin((0.001+10.0/SITES)*GID)/2.0;
    f2 = cos((0.001-10.0/SITES)*GID)/3.0;
    f3 = sin((0.001-18.0/SITES)*GID)/4.0;
    f4 = sqrt(1.0-f1*f1-f2*f2-f3*f3);

    matrix.uv1 = (hgpu_float4) (f1, f3, f4, f2);
...
  }
```

```
#ifdef GID_UPD
prns[GID].x = (float) index;
#endif
        mU     = lattice_heatbath_2(&staple,&m0,&bet,prns);
```

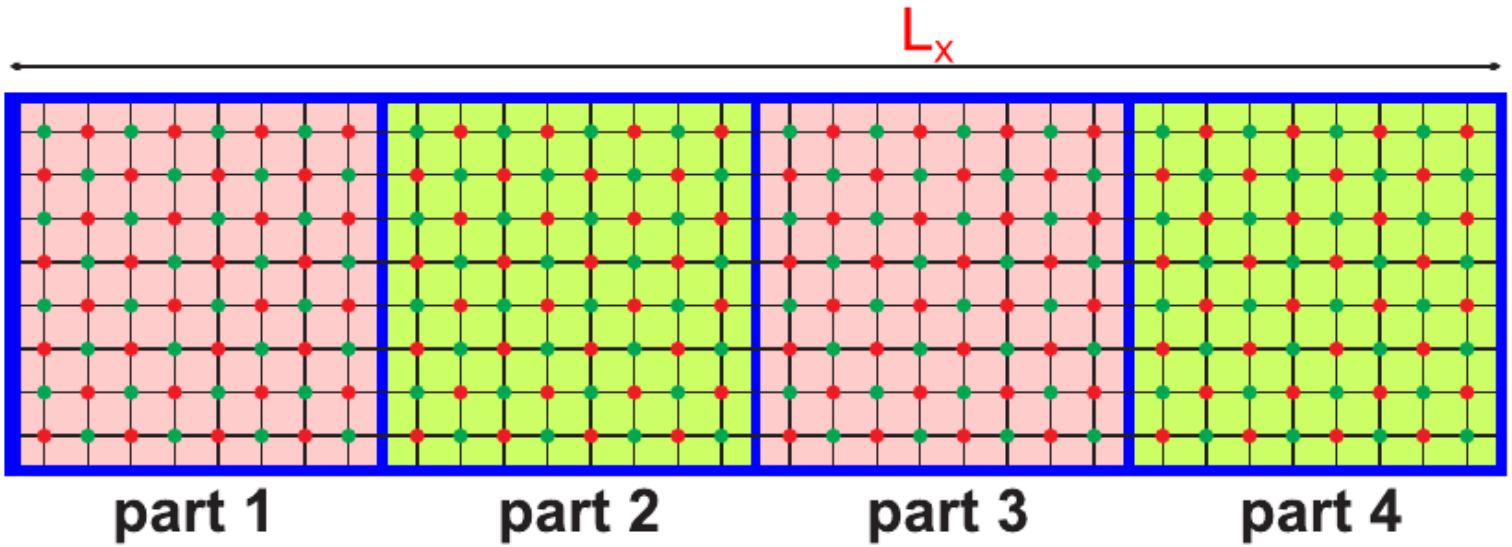
```
#ifdef GID_UPD
        gid = prns[(*indprng)].x;

rnd.x = (hgpu_float) fabs(sin((0.005*(1 + NHIT)+270.0/SITES)*gid));
rnd.y = (hgpu_float) fabs(cos((0.005*(1 + NHIT)+ 60.0/SITES)*gid));
rnd.z = (hgpu_float) fabs(sin((0.005*(1 + NHIT)-150.0/SITES)*gid));
rnd.w = (hgpu_float) fabs(cos((0.005*(1 + NHIT)-380.0/SITES)*gid));
#else
        rnd.x = (hgpu_float) prns[(*indprng)].x;
        rnd.y = (hgpu_float) prns[(*indprng)].y;
        rnd.z = (hgpu_float) prns[(*indprng)].z;
        rnd.w = (hgpu_float) prns[(*indprng)].w;
        (*indprng) += PRNGSTEP;
#endif
```

```
#ifdef GID_UPD
delta = cosrnd * cosrnd / bdet;
flag = true;
#endif
...
c.uv1.x = 1.0 - delta;
```

Big lattice

Second level checkerboard scheme



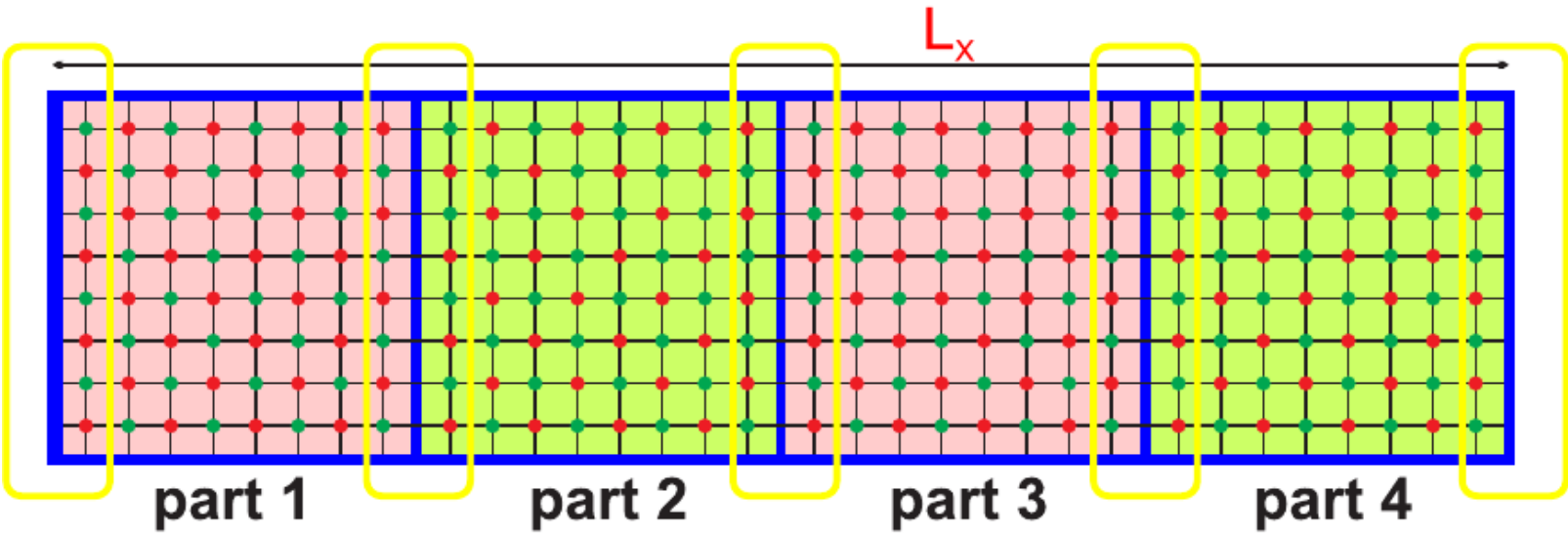
4D lattice $L_x \times L_y \times L_z \times L_t$

Finite temperature: $L_s > L_t$

Latice is devided in X direction

Big lattice

Second level checkerboard scheme



4D lattice $L_x \times L_y \times L_z \times L_t$

Finite temperature: $L_s > L_t$

Latice is devided in X direction

For updating parts the boundary elements should be accounted

Performance results

QCDGPU was tested on NVIDIA GeForce GTX 560 M (Windows 7), NVIDIA GeForce GTX 560 Ti,AMD Radeon HD 7970 (OpenSUSE 12.2), HD 6970, HD 5870 (OpenSUSE 11.4)

Timings of 1 working sweep for O(4), SU(2) and SU(3) models in single- and double-precision (seconds)

Model	Device	Parts	Lattice	Single	Double
O(4)	GTX560M	1	16^4	0.08	0.12
			24^4	0.42	0.6
			32^4	1.31	1.87
		4	48^4	7.79	11.11
	GTX560Ti	1	32^4	0.72	0.99
		4	48^4	4.64	6.10
	HD 5870	2	32^4	0.96	1.56
	HD 6970	2	32^4	0.73	1.3
	HD 7970	2	32^4	0.66	0.95
		12	72^4	20.35	36.2
	HD 7970+GTX560Ti	4	32^4	0.72	0.96
		8	48^4	2.97	4.1
SU(2)	GTX560Ti	1	44^4	0.95	1.14
	HD 5870	1	30^4	0.22	0.24
	HD 6970	1	30^4	0.20	0.22
	HD 7970	1	48^4	4.64	6.10
SU(3)	GTX560Ti	1	32^4	0.56	1.17
			36^4	0.90	1.86
	HD 6970	1	24^4	0.24	0.23
	HD 7970	1	28^4	0.13	0.22

"Hot" start, RANLUX PRNG (luxury level 3). For O(4) model NHIT = 100 tries were used to update each lattice site (this provides acceptance rate up to 50%). For SU(2) and SU(3) models NHIT = 10 and reunitarization were used. One bulk sweep was performed to decorrelate configurations to be measured.

QCDGPU performance results comparison

Cardoso and Bicudo reported in [1] the following timings (seconds per single sweep) on lattice 8^4 for SU(2) and SU(3) models (one lattice sweep with reunitarization, without any measurements).
The authors [1] used NVIDIA GeForce GTX 580 and NVIDIA CUDA

	SU(2)		SU(3)	
	single	double	single	double
[1]	$6\text{x}10^{-4}$	$7\text{x}10^{-4}$	$2\text{x}10^{-3}$	$4\text{x}10^{-3}$
GTX 560 Ti	$6\text{x}10^{-4}$	10^{-3}	$1.3\text{x}10^{-3}$	$4.9\text{x}10^{-3}$
HD 7970	$2\text{x}10^{-3}$	$2.6\text{x}10^{-3}$	$3.4\text{x}10^{-3}$	$5.1\text{x}10^{-3}$

* For large lattices AMD HD 7970 demonstrates better performance than NVIDIA GTX 560 Ti

[1] N. Cardoso and P. Bicudo: Generating SU(Nc) pure gauge lattice QCD configurations on GPUs with CUDA and OpenMP. *Comput. Phys. Commun.* 184: 509-518, 2013

Conclusions

1. new package QCDGPU for Monte Carlo simulations of SU(N) gluodynamics in external field and O(N) model on OpenCL-compatible devices is developed
2. QCDGPU allows to carry out the simulations for very big lattices in single- or multi-GPU mode
3. simulations can be run with single or double precision
4. QCDGPU allows to investigate most popular LGT models in N-dimensional space-time
5. current version of QCDGPU uses trivial parallelization scheme to distribute computing
6. built-in mechanism for saving the computational state while achieving some conditions allows to interrupt long calculations without threat of data loss and to continue them on another available device

Conclusions

1. new package QCDGPU for Monte Carlo simulations of SU(N) gluodynamics in external field and O(N) model on OpenCL-compatible devices is developed
2. QCDGPU allows to carry out the simulations for very big lattices in single- or multi-GPU mode
3. simulations can be run with single or double precision
4. QCDGPU allows to investigate most popular LGT models in N-dimensional space-time
5. current version of QCDGPU uses trivial parallelization scheme to distribute computing
6. built-in mechanism for saving the computational state while achieving some conditions allows to interrupt long calculations without threat of data loss and to continue them on another available device

Plans for future:

1. implement the automatic adjustments of package start-up parameters via built-in profiling mechanism, as well as additional micro-benchmarks
2. realization of RHMC algorithm for including fermionic fields on a lattice
3. running of one simulation on several hosts at the same time on the base of MPI, which is very important in case of accounting of fermionic fields

Conclusions

1. new package QCDGPU for Monte Carlo simulations of SU(N) gluodynamics in external field and O(N) model on OpenCL-compatible devices is developed
2. QCDGPU allows to carry out the simulations for very big lattices in single- or multi-GPU mode
3. simulations can be run with single or double precision
4. QCDGPU allows to investigate most popular LGT models in N-dimensional space-time
5. current version of QCDGPU uses trivial parallelization scheme to distribute computing
6. built-in mechanism for saving the computational state while achieving some conditions allows to interrupt long calculations without threat of data loss and to continue them on another available device

Plans for future:

1. implement the automatic adjustments of package start-up parameters via built-in profiling mechanism, as well as additional micro-benchmarks
2. realization of RHMC algorithm for including fermionic fields on a lattice
3. running of one simulation on several hosts at the same time on the base of MPI, which is very important in case of accounting of fermionic fields

Thank you for your attention!