Multipart SU(N) mode in QCDGPU package for Monte Carlo lattice simulations

V. I. Demchik, N. V. Kolomoyets

Dnepropetrovsk National University
Dnepropetrovsk, Ukraine

Perspectives of GPU Computing in Science

September 26, 2016

Outline

Lattice approach

Lattice QCD – one of non-perturbative approaches to QCD

Continuous space-time $\rightarrow$ Discrete lattice

Discretized operators

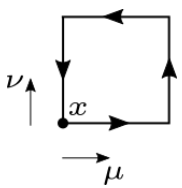
Gluonic fields – SU(N) matrices on the lattice links

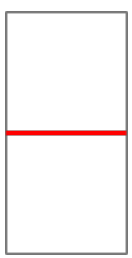
$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}U \, \mathcal{O}[U] e^{-S[U]} \quad \rightarrow \quad \langle \mathcal{O} \rangle \approx \frac{1}{N} \sum_{U_n} \mathcal{O}[U_n]$$

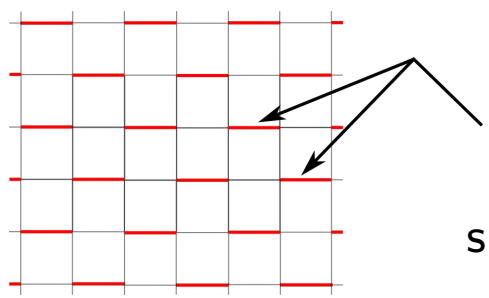
$$\mathcal{Z} = \int \mathcal{D}U \, e^{-S[U]}, \quad \int \mathcal{D}U = \prod_{x,\mu} \int dU_\mu(x), \quad U_n \propto e^{-S}$$

Lattice Wilson action:

$$S_W = \beta \sum_{\square} \left[1 - \frac{1}{N} \mathrm{Re}\,\mathrm{Tr} U_{\square} \right],$$







may be updated simultaneously

QCDGPU in general

QCDGPU is a multi-GPU open-source package for lattice Monte Carlo simulations of SU(N) gluodynamics at finite temperature and O(N) models.

It is implemented in C/C++/OpenCL and OpenMP (in multipart mode), tested on AMD and NVIDIA GPUs, AMD and Intel CPUs and may run on other OpenCL-compatible devices.OS independent.

It allows investigation of phenomena in external chromomagnetic field (implemented through the "twisted boundary conditions").

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

QCDGPU in general

QCDGPU is a multi-GPU open-source package for lattice Monte Carlo simulations of SU(N) gluodynamics at finite temperature and O(N) models.

It is implemented in C/C++/OpenCL and OpenMP (in multipart mode), tested on AMD and NVIDIA GPUs, AMD and Intel CPUs and may run on other OpenCL-compatible devices;OS independent.

It allows investigation of phenomena in external chromomagnetic field (implemented through the "twisted boundary conditions").

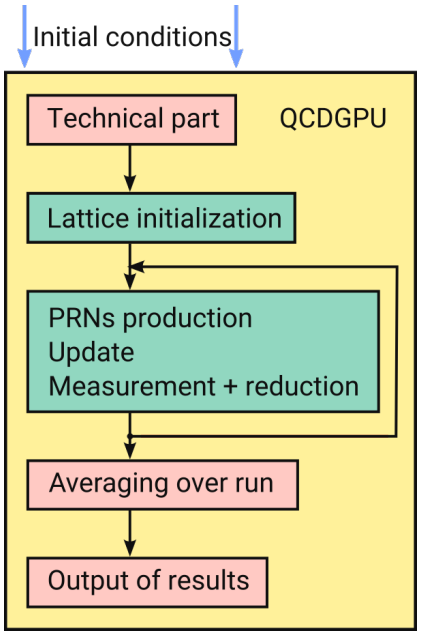
`https://github.com/vadimdi/QCDGPU`

Objectives for the lattice decomposition:

- overcome limited GPU memory
- decrease of computational time

  execution on several devices simultaneously

Package structure

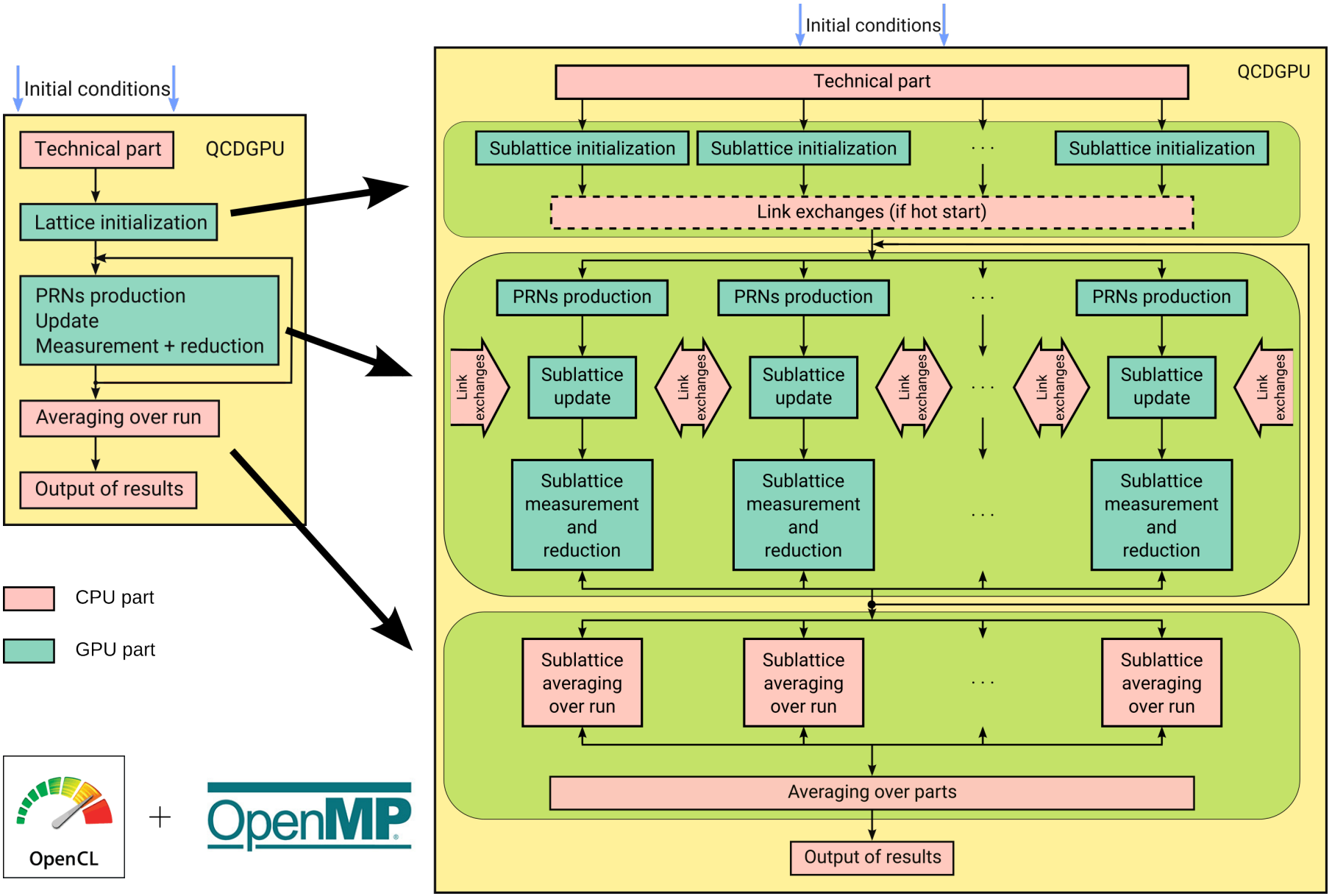


CPU part

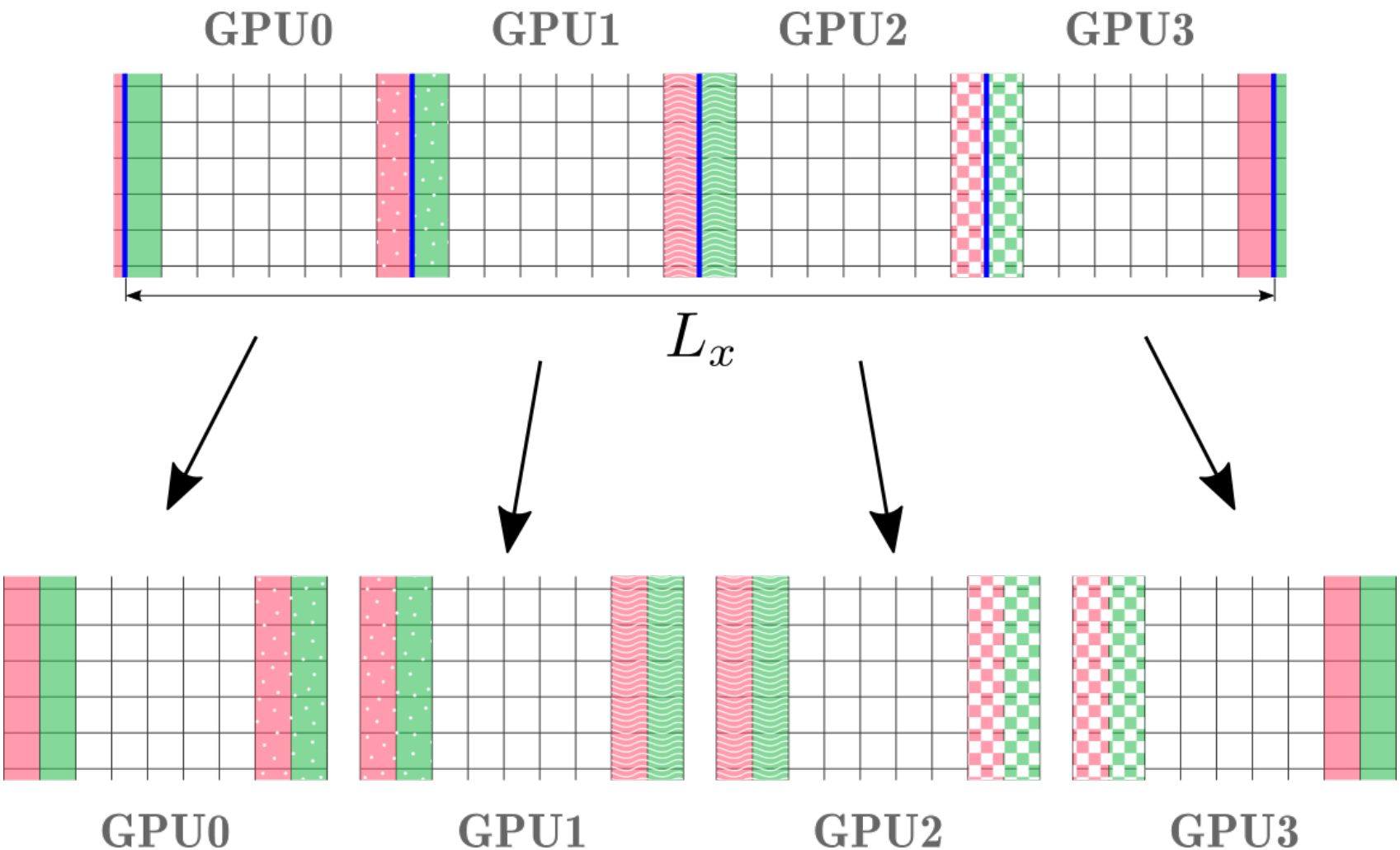
GPU part

OpenCL

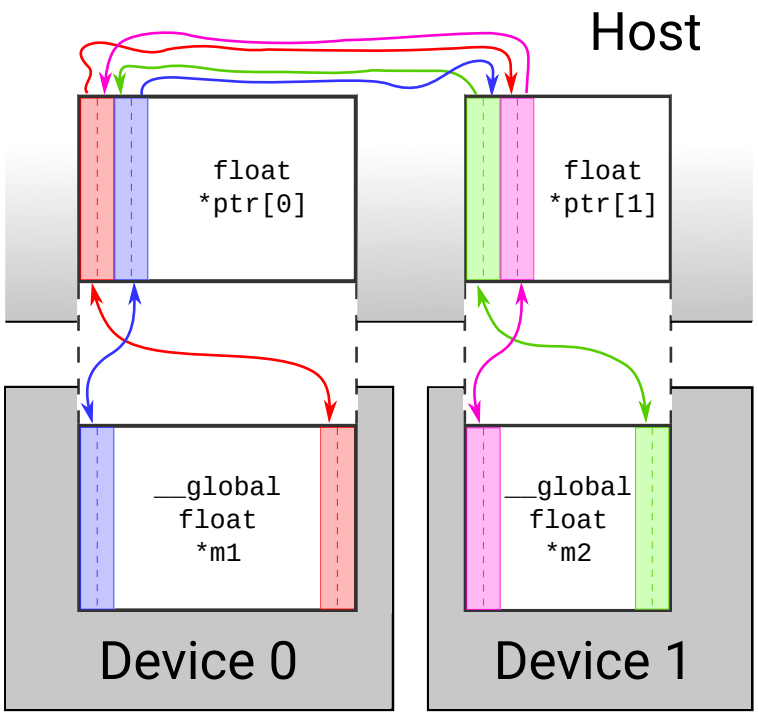
Package structure



Lattice decomposition

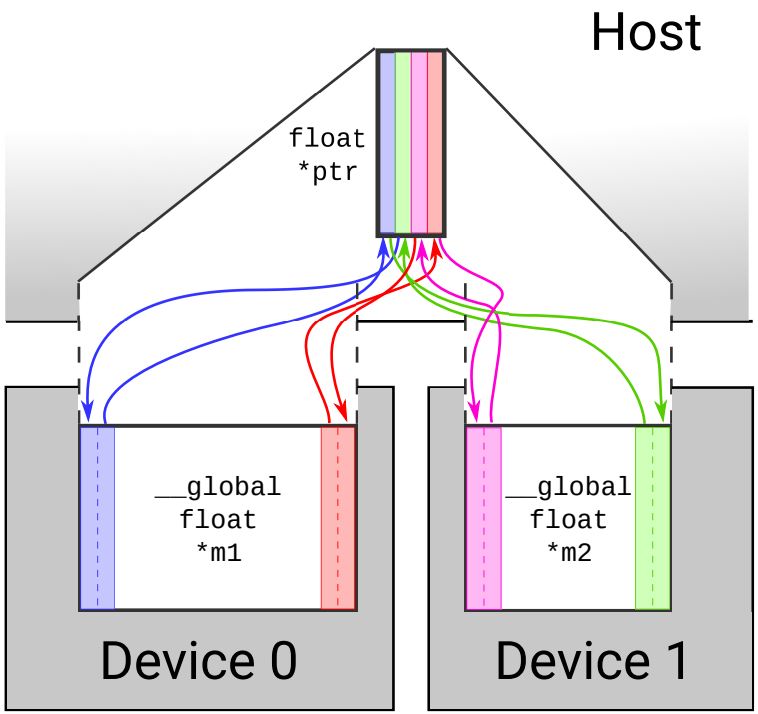


Edges exchange



map / unmap

/* currently fails on AMD GPUs
because of driver bug */



read / write

Pseudorandom numbers

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

1. RANLUX (with several luxury levels)
2. RANMAR
3. RANECU
4. XOR7, XOR128
5. PM
6. MRG32k3a

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

PRNG is initialized with with one integer number – `RANDSERIES`.

At each lattice part, PRNG is initialized with own `RANDSERIES`.

Single/double precision PRNGs are implemented.

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

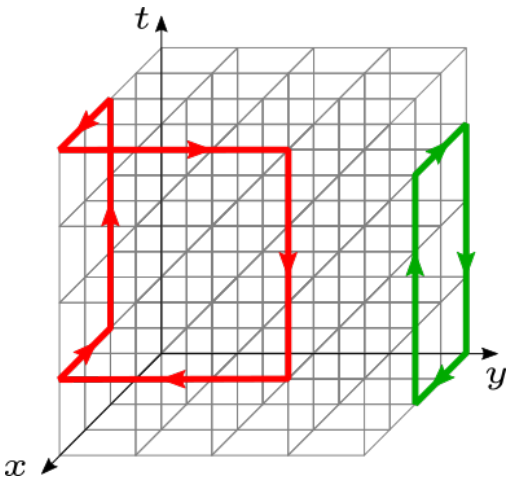
Measured quantities

- plaquette
- Wilson action (also coordinatewise)
- components of tensor of SU(N) electromagnetic field $F^a_{\mu\nu}$

[arXiv:1212.6185v1[hep-lat]]

- Polyakov loop L (also coordinatewise), L^2, L^4
- Wilson loop

The Wilson loop



Continuous theory:

$$W_{\mathcal{L}} = \mathrm{Tr}\left[\mathcal{P}\exp\left(i\oint_{\mathcal{L}} A_\mu dx^\mu\right)\right]$$

Lattice theory:

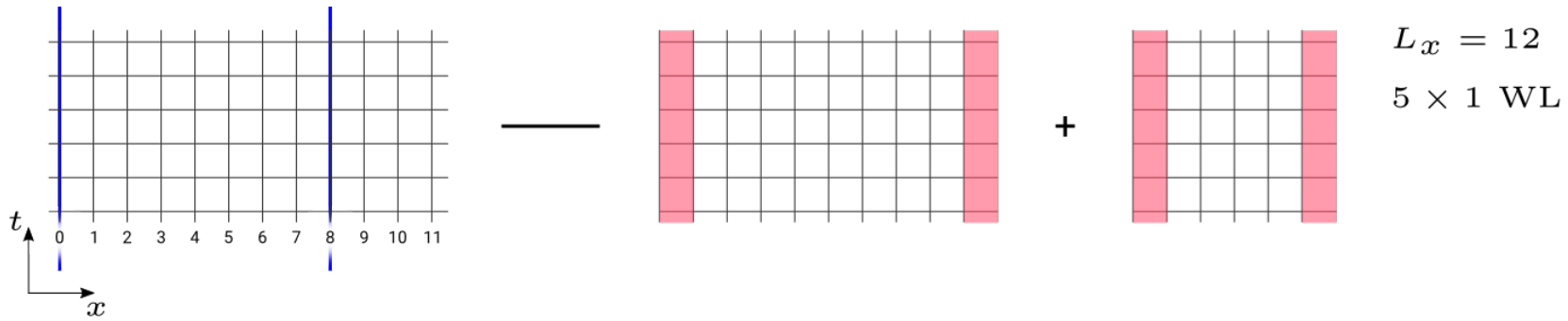
$$W_{\mathcal{L}} = \mathrm{Tr}\left[\prod_{(x,\mu)\in\mathcal{L}} U_\mu(x)\right]$$

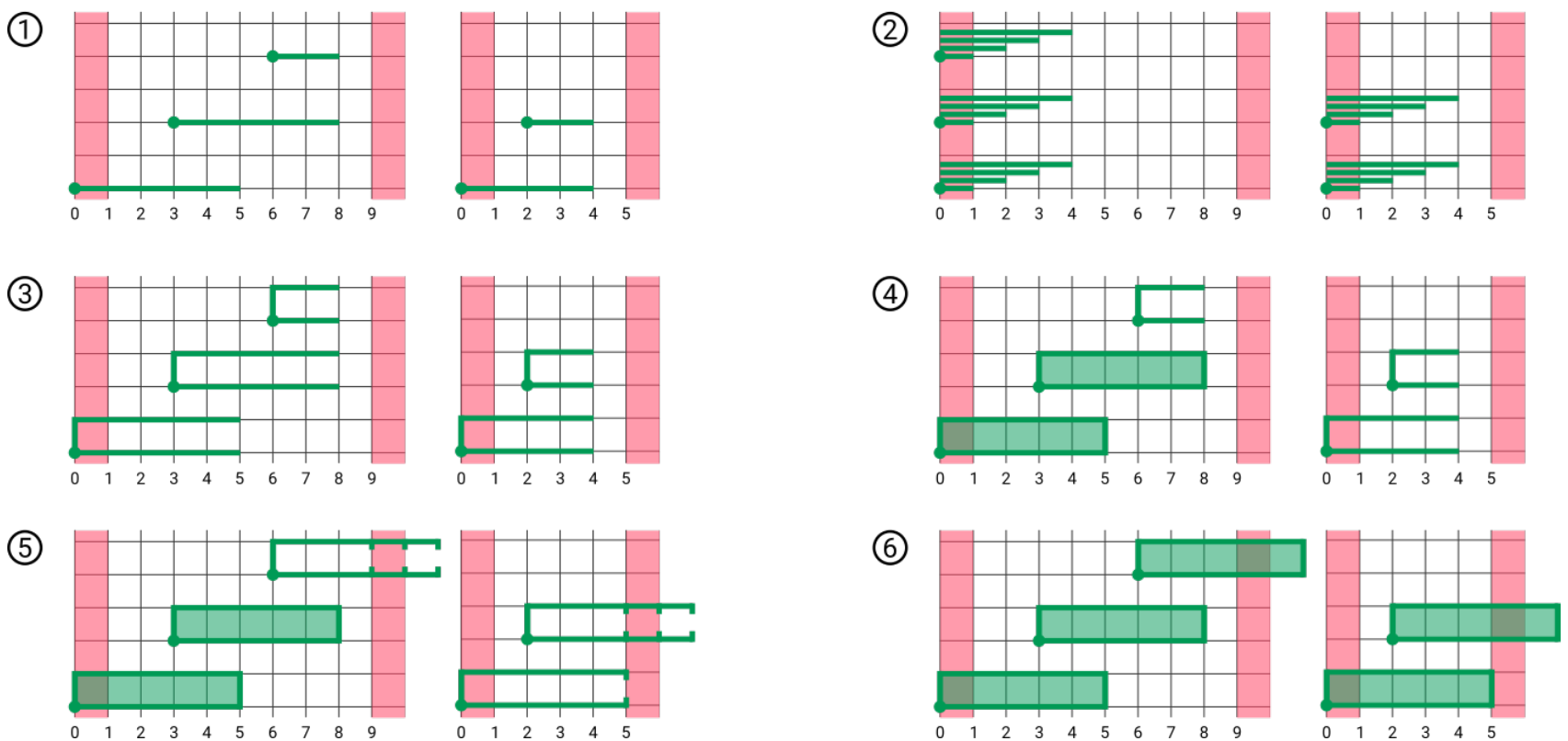
Wilson loops: planar, non-planar

Usage of the Wilson loop:

- indicates the confinement/deconfinement phases
- string tension
- lattice spacing

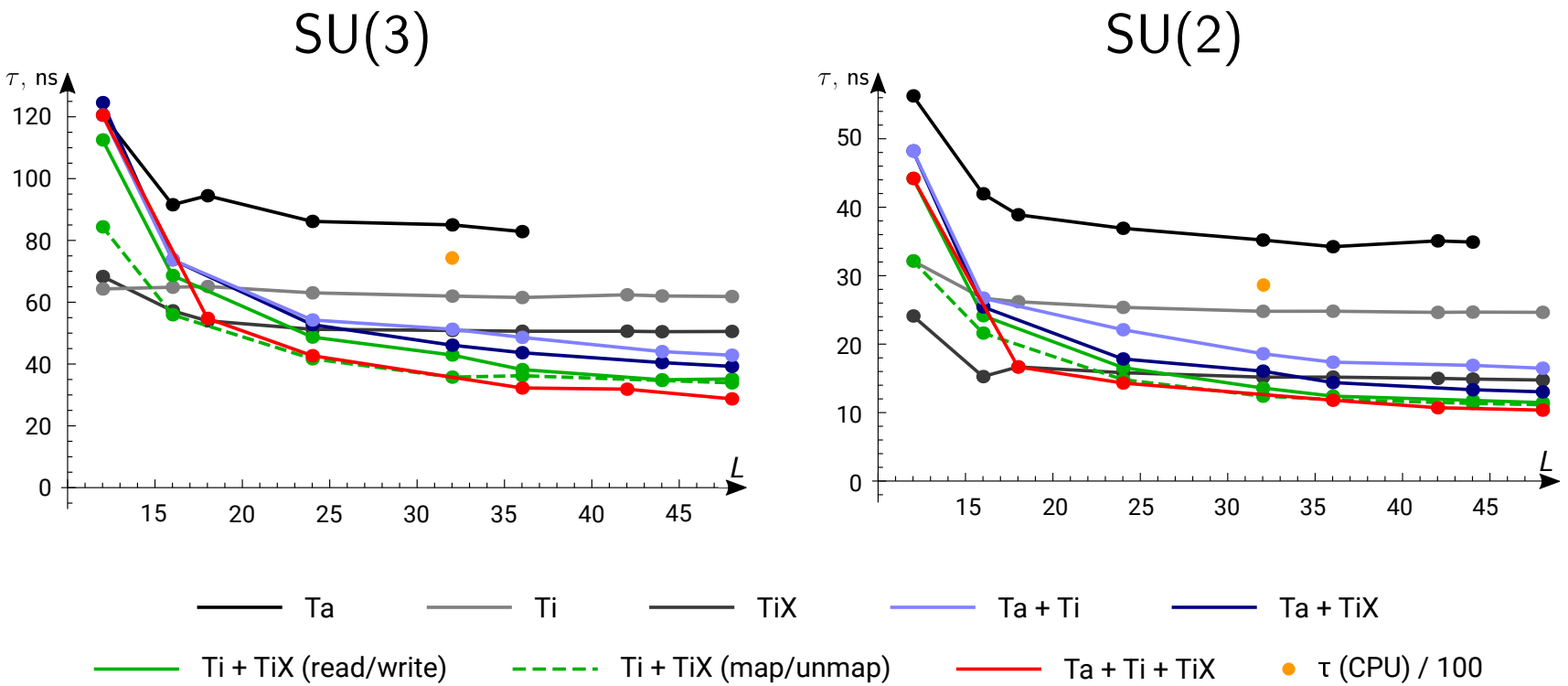
Implementation of the Wilson loop





Performance results

L^4 lattice
τ = (time for 1 lattice update + 1 measurement of S_W)/($N_d \cdot L^4$)



"Cold" start, RANLUX PRNG (luxury level 3, single precision). NHIT = 10, reunitarization is used, double precision.
1 lattice part / device; sizes of the parts are chosen to provide better performance.

Ta – AMD Radeon HD7990 (Tahiti)
Ti – NVIDIA GeForce GTX TITAN
TiX – NVIDIA GeForce GTX TITAN X

[HPC Village] http://openwall.info/wiki/HPC/Village

Conclusions

Multipart mode in SU(N) part of QCDGPU package is implemented using OpenMP:

1. several parts may be processed in parallel on several devices as well as sequentially on a single device
2. two multithread variants of exchange of the edge links are implemented:
   - through map/unmap: is faster but currently isn't supported by AMD
   - through read/write: is some slower but is supported by all GPU manufacturers
3. measurement of usual and particular (like components of $F_{\mu\nu}$ tensor) quantities is implemented in multipart mode. In particular, measurement of the Wilson loop is implemented
4. the $\sim 28.5 - 33\%$ acceleration is obtained in parallel run on two GPUs and $\sim 36 - 43\%$ acceleration on three GPUs for SU(3) model; for SU(2) model these numbers are $19.5 - 24.5\%$ and $22.3 - 29.8\%$, correspondingly.

Conclusions

Multipart mode in SU(N) part of QCDGPU package is implemented using OpenMP:

1. several parts may be processed in parallel on several devices as well as sequentially on a single device
2. two multithread variants of exchange of the edge links are implemented:
   - through map/unmap: is faster but currently isn't supported by AMD
   - through read/write: is some slower but is supported by all GPU manufacturers
3. measurement of usual and particular (like components of $F_{\mu\nu}$ tensor) quantities is implemented in multipart mode. In particular, measurement of the Wilson loop is implemented
4. the $\sim 28.5 - 33\%$ acceleration is obtained in parallel run on two GPUs and $\sim 36 - 43\%$ acceleration on three GPUs for SU(3) model; for SU(2) model these numbers are $19.5 - 24.5\%$ and $22.3 - 29.8\%$, correspondingly.

Plans for future:

- implementation of fermionic fields

Conclusions

Multipart mode in SU(N) part of QCDGPU package is implemented using OpenMP:

1. several parts may be processed in parallel on several devices as well as sequentially on a single device
2. two multithread variants of exchange of the edge links are implemented:
   - through map/unmap: is faster but currently isn't supported by AMD
   - through read/write: is some slower but is supported by all GPU manufacturers
3. measurement of usual and particular (like components of $F_{\mu\nu}$ tensor) quantities is implemented in multipart mode. In particular, measurement of the Wilson loop is implemented
4. the $\sim 28.5 - 33\%$ acceleration is obtained in parallel run on two GPUs and $\sim 36 - 43\%$ acceleration on three GPUs for SU(3) model; for SU(2) model these numbers are $19.5 - 24.5\%$ and $22.3 - 29.8\%$, correspondingly.

Plans for future:

- implementation of fermionic fields

Thank you for your attention!