# Gauge fixing using overrelaxation and simulated annealing on GPUs

Mario Schröck[1] and Hannes Vogt[2]

[1]Institut für Physik, FB Theoretische Physik, Universität Graz, 8010 Graz, Austria
[2]Institut für Theoretische Physik, 72076 Tübingen, Germany

Lattice 2012
Cairns, June 25, 2012

## Motivation

- the QCD action is invariant under local gauge transformations

$$g(x)U_\mu(x)g^\dagger(x+\hat{\mu})\,, \quad g(x) \in SU(3)$$

- in order to study gauge noninvariant quantities one has to "fix the gauge", i.e., choose a particular transformation $g(x)\,\forall x$
- gauge fixing often demands the largest part of computer time when extracting gauge dependent observables from gauge configurations
- the relaxation algorithm for gauge fixing is strictly local and thus perfectly suited to be accelerated by GPUs
- overrelaxation and stochastic relaxation overcome the problem of critical slowing down

## Our program: overview

Code design

- CUDA C++
- use of template classes for general, reusable code

Algorithms

- overrelaxation
- stochastic relaxation
- simulated annealing, see [Bali et al., Phys. Rev. D54 (1996)]

Gauges

- Landau gauge $\partial_\mu A_\mu = 0$
- Coulomb gauge $\partial_i A_i = 0$ (max. each time-slice separately)
- Maximally abelian gauge

## Gauge fixing on the lattice

- the continuum Landau gauge condition is equivalent to maximizing

$$F_g[U] = \sum_\mu \sum_x \mathfrak{Re}\ \mathrm{tr}\left[U_\mu^g(x)\right]$$

- relaxation algorithm: optimize $F_g[U]$ *locally*

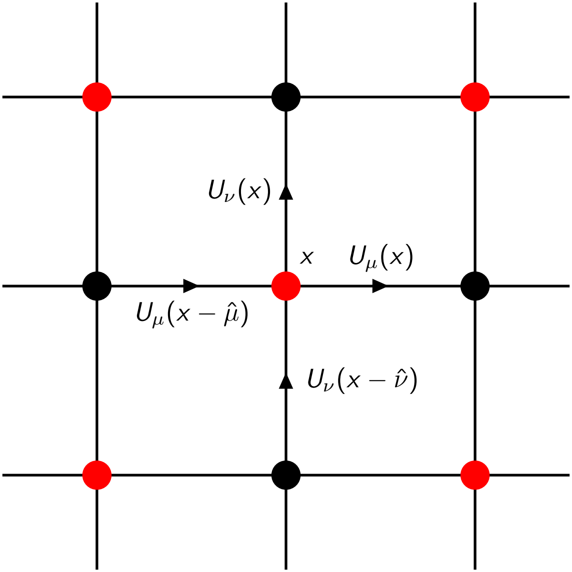
$$\mathfrak{Re}\ \mathrm{tr}\left[g(x)K(x)\right] \to \text{max.}$$

  with $K(x) := \sum_\mu U_\mu(x)g^\dagger(x+\hat{\mu}) + U_\mu^\dagger(x-\hat{\mu})g^\dagger(x-\hat{\mu})$
- in SU(2) local optimum given by

$$g(x) = K(x)^\dagger / \det K(x)^\dagger$$

  for larger SU(N) operate in SU(2) subgroups
- overrelaxtion: replace $g(x) \to g^\omega(x), \quad \omega \in [1,2)$
- stochastic relaxation: replace $g(x) \to g^2(x)$ with probability $p$
- gauge precision: $\theta \approx \frac{1}{V}\sum_x \left|\partial_\mu A_\mu(x)\right|^2$

## Checker board decomposition



## Overrelaxation algorithm

```
 1: while precision $\theta$ not reached do
 2:   for sublattice = RED, BLACK do
 3:     for all x of sublattice do
 4:       for all SU(2) subgroups do
 5:         $g(x) \to \sum_\mu \left\{ U_\mu^\dagger(x) + U_\mu(x-\hat{\mu}) \right\}$      $\to$ 60 Flop
 6:         $g(x) \to g^\omega(x)$, project to SU(2)                                          $\to$ 19 Flop
 7:         for all $\mu$ do
 8:           $U_\mu(x) \to g^\omega(x) U_\mu(x)$                                               $\to$ 84 Flop
 9:           $U_\mu(x-\hat{\mu}) \to U_\mu(x-\hat{\mu}) g^\omega(x)^\dagger$                    $\to$ 84 Flop
10:         end for
11:       end for
12:     end for
13:   end for
14: end while
```

in total 751 Flop per site and SU(2) subgroup iteration

$\Rightarrow$ 2253 Flop/site for SU(3).

## NVIDIA GeForce GTX 580

| | |
|---|---|
| architecture | Fermi |
| Compute Capability | 2.0 |
| # SMs[1] | 16 |
| # total CUDA cores | 512 |
| device memory | 1.5 GB |
| memory bandwith | 192.4 GB/s |
| ECC available | no |
| L2 cache | 768 KB |
| L1 cache / SM | 16 KB or 48 KB |
| shared memory / SM | 16 KB or 48 KB |
| 32-bit registers / SM | 32768 |
| max. registers / thread | 63 |

[1]Streaming Multiprocessor

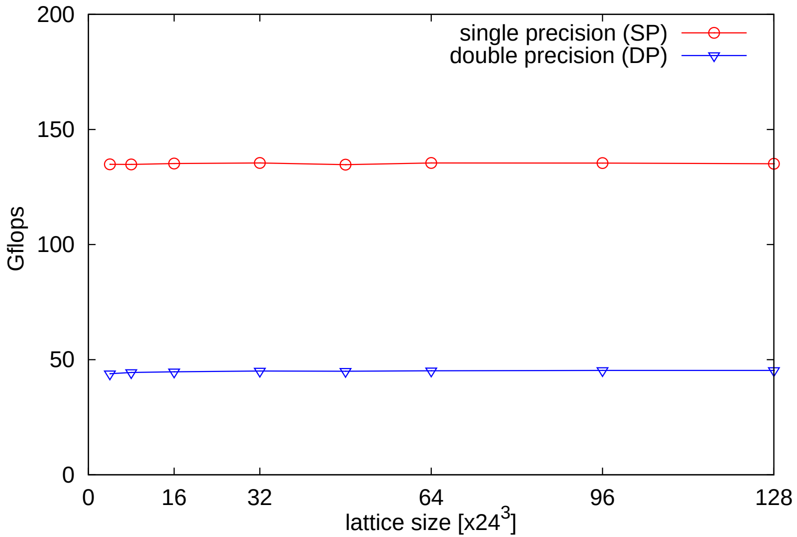
## First attempt

Taking the GPU hardware properties into account...

- assign one thread to each lattice site
- memory coalescing: rearrange gauge field into black and red sublattices, choose the site index running faster than color and Dirac indices
- prefetch data from global to local memory
- reduce memory traffic: reconstruct the third line of each SU(3) matrix instead of prefetching it

see also e.g. [Babich et al., Comp. Phys. Comm. 181 (2010)]

## Performance of the first attempt



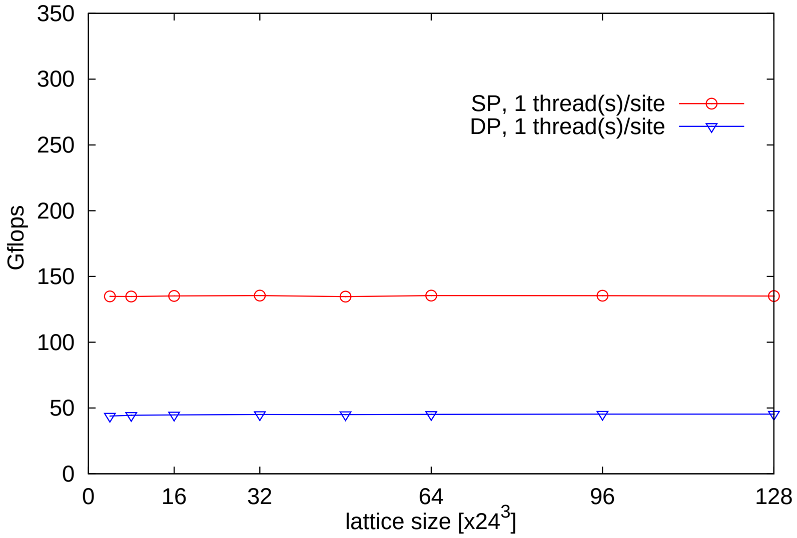
## Analysis & optimization

- at the beginning *and* end of each iteration step, each thread needs the eight neighbor gauge links:
  data volume $8 \times 18$ reals/site $=$ 144 reals/site
- performance is restricted by the register limit of 63 per thread: the standard one-thread-per-site strategy results in many register spills to global memory which negatively effects the (bandwith bound) performance
- optimization approach: avoid register spills by adopting a finer parallelization granularity

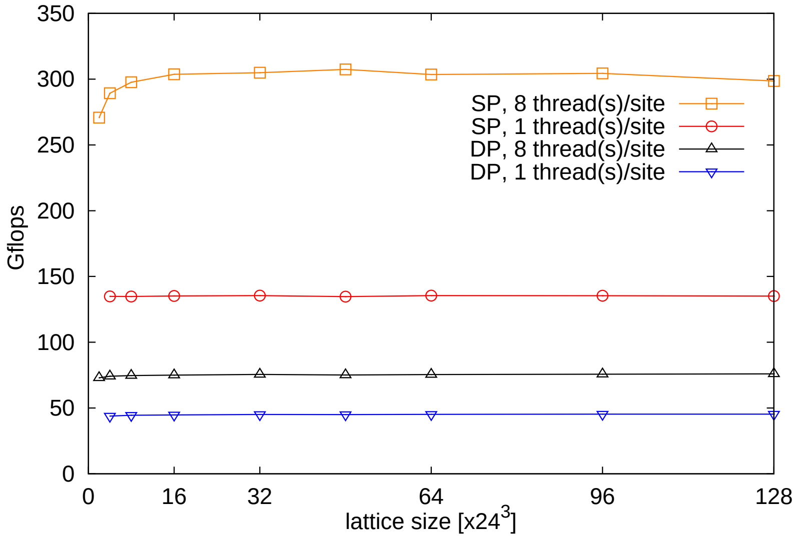
## Eight-threads-per-site strategy

- we assign eight threads to each lattice site, i.e., each thread handles only one neighbor gauge link
- only 18 registers per thread needed
- we start thread blocks of size $8 \times 32 = 256$ in order to be able to avoid warp divergences
- the gauge transformation is then accumulated in shared memory: $g(x) \in SU(2)$ (subgroup iteration) can be stored as 4 reals, thus $4 \times 32 = 128$ reals or 512 bytes per thread block
- moreover we limit the register usage to 32 to achieve a higher occupancy

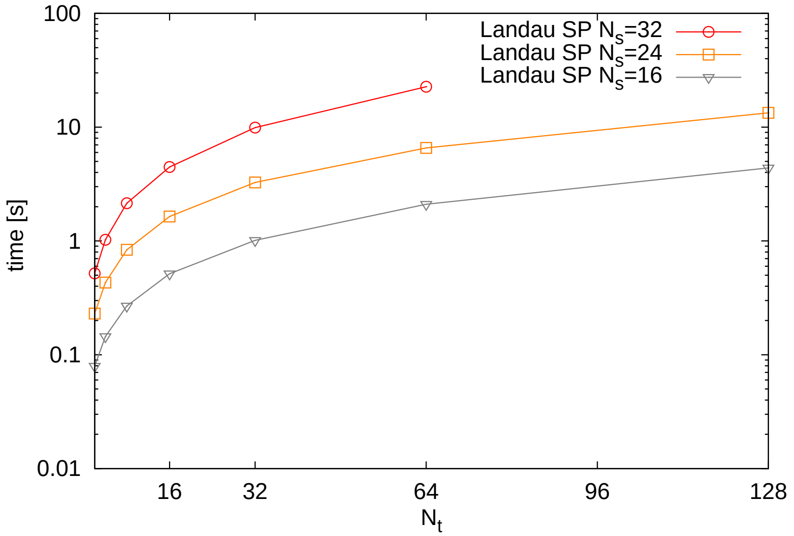
# Performance



## Performance



## Time needed for 1000 iterations on $N_s^3 \times N_t$ lattices in SP



## Speedup over CPU

Comparison to the performance of

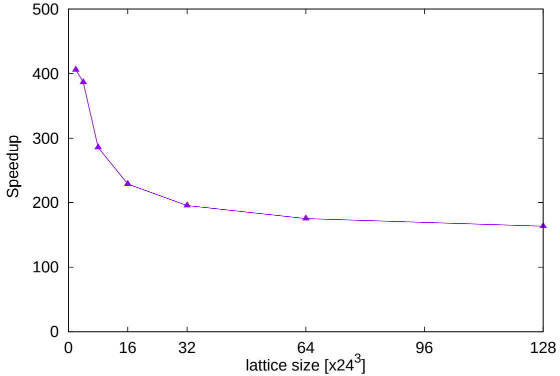
- the overrelaxation algorithm of the FermiQCD library[2]
- run in parallel (MPI) on all four cores of the
- Intel Core i7-950 (“Bloomfield”) @ 3.07GHz

[2]`www.fermiqcd.net`

## Speedup over CPU

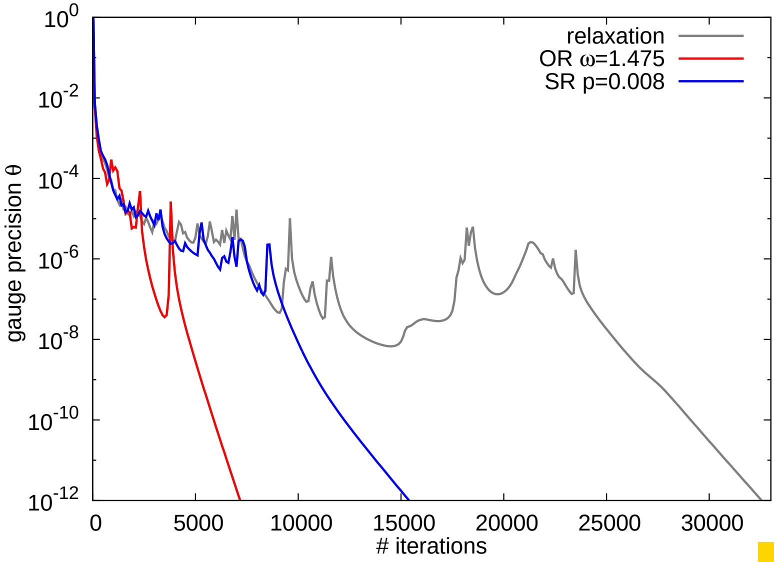
Comparison to the performance of

- the overrelaxation algorithm of the FermiQCD library[2]
- run in parallel (MPI) on all four cores of the
- Intel Core i7-950 ("Bloomfield") @ 3.07GHz



[2] www.fermiqcd.net

## Number of iterations on a random $\beta = 6.1$, $32^4$ lattice



## Summary

We offer a very efficient implementation of

- Landau,
- Coulomb
- and maximally abelian gauge fixing

using the

- simulated annealing,
- overrelaxation
- and stochastic relaxation algorithms.

Performance highlights using the eight-threads-per-site strategy in single precision:

- 300 Gflops for Landau gauge fixing with the overrelaxation algorithm
- time needed to fix a $\beta = 6.1$ $32^4$ lattice of the order of one minute
- two orders of magnitude speedup over FermiQCD run on the Intel Core i7 quad core CPU