DD-αAMG on QPACE 2: A case study

Tilo Wettig

Department of Physics
University of Regensburg

QCDNA 2016, Edinburgh

arXiv:1412.2629, 1512.04506, 1601.03184, Lattice 2016
(work done by Peter Georg, Simon Heybrock, Daniel Richtmann)

Outline

Outline

- objective: use an iterative method to solve Dirac equation

$$Du = f$$

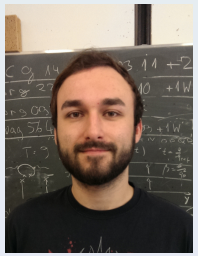
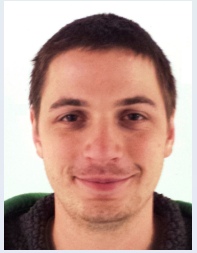
- condition number of D increases like $(am)^{-1}$
  $\rightarrow$ use a preconditioner M with $M^{-1} \approx D^{-1}$
- using

$$DM^{-1}Mu = DM^{-1}v = f$$

  we solve for v with preconditioned matrix DM^{-1} (smaller condition number) and obtain $u = M^{-1}f$
- error $e_n = u^* - u_n$ can formally be written as a linear combination of the eigenmodes of D
  $\rightarrow$ find a preconditioner that reduces the contributions of these eigenmodes to the error

Introduction

- adaptive multigrid appears to be the method of choice to precondition D for $a, m \to 0$ Brannick et al. 0707.4018, Babich et al. 1005.3043
- we use the Wuppertal version (DD-αAMG) Frommer et al. 1303.1377, github.com/DDalphaAMG
  - domain-decomposition (DD) based smoother reduces contributions of high modes to error Lüscher hep-lat/0310048
  - coarse-grid correction (CGC) reduces contributions of low modes
  - relation to inexact deflation (Lüscher 0706.2298) understood
- optimized implementation of Wuppertal code on QPACE 2
  - work done mostly by Simon Heybrock, Daniel Richtmann, Peter Georg

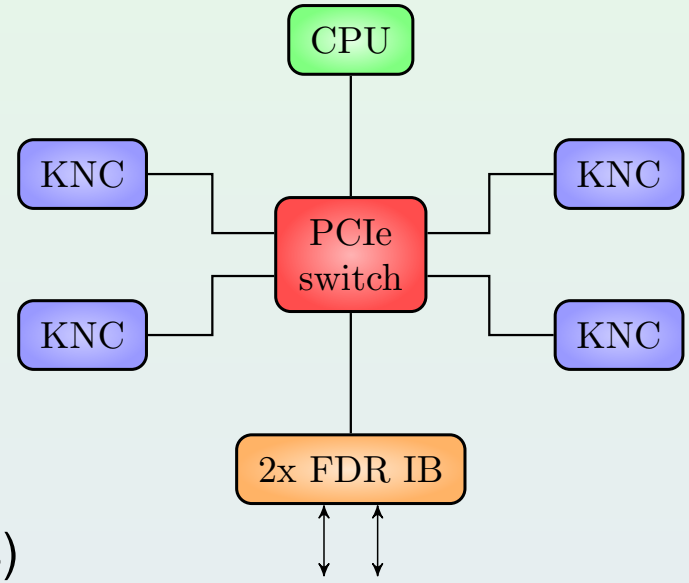




    with support from Matthias Rottmann
  - currently only 2-level MG is optimized (sufficient for our current lattices)
  - multi-level MG will be optimized in near future (needed for future lattices)

Outline

QPACE 2: High-level design

- machine consists of identical compute nodes:
  - 4 Intel Xeon Phi a.k.a. Knights Corner (KNC) coupled via PCIe switch (PLX 8796)
  - weak CPU (Xeon E3-1230L v3) for root complex functionality (PCIe master)
  - dual-FDR Infiniband card (Mellanox Connect-IB) (network b/w consistent with LQCD requirements)

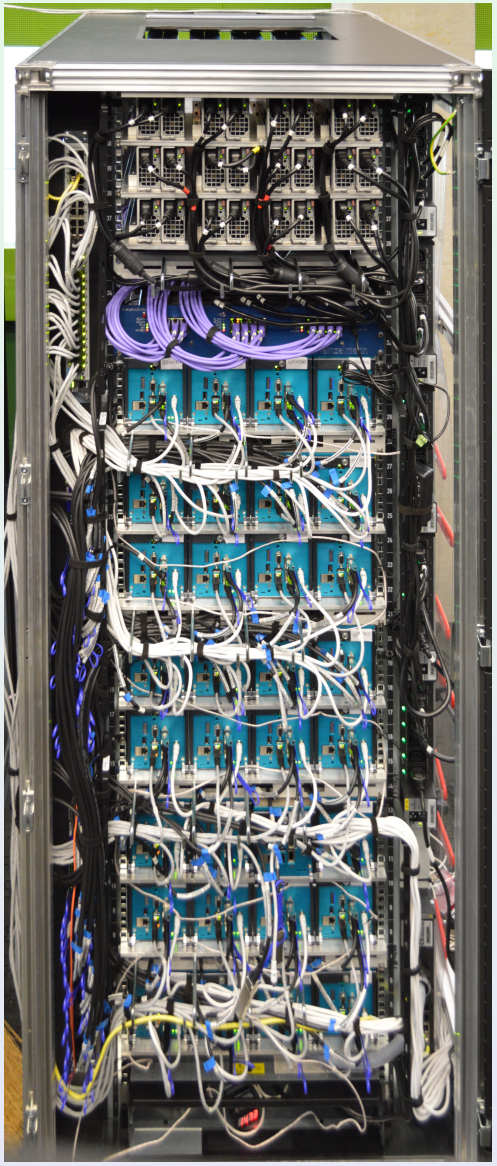


- communication
  - within node via PCIe (8 GB/s between each KNC and PCIe switch)
  - out of node via Infiniband (13.5 GB/s per node)

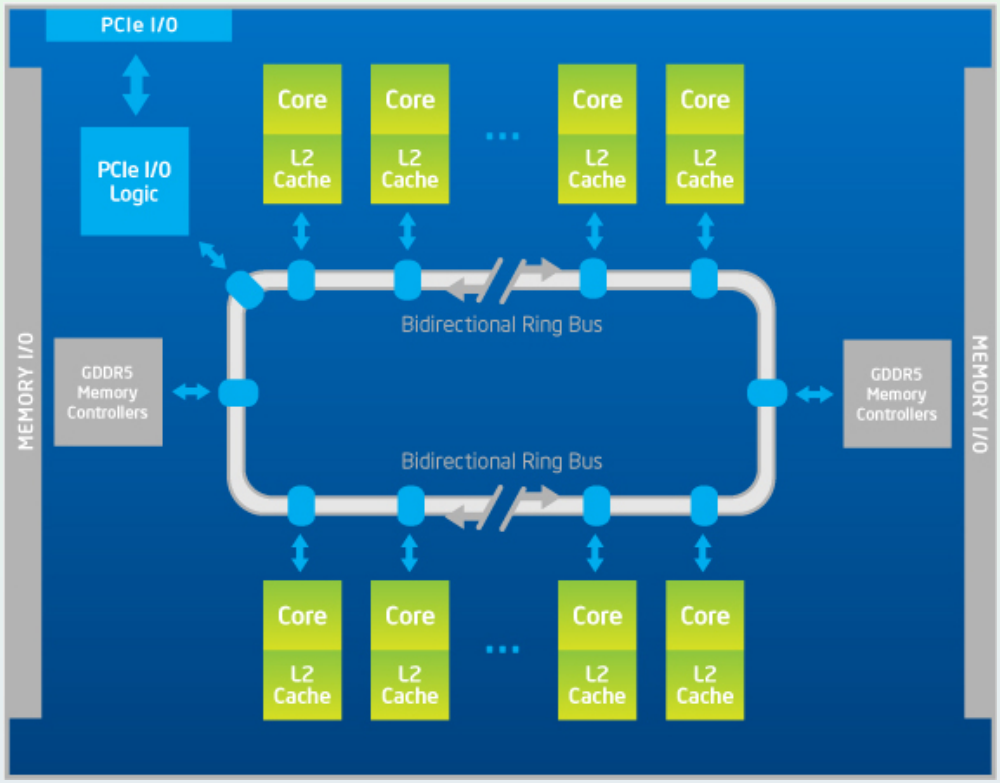
QPACE 2: System design

rack design:

- standard 19” rack (height 42U)
- 64 bricks (256 KNCs) in 24U
- rest for PSUs, switches (4x IB and 3x GigE), management/login server
- 310 TFlop/s DP peak per rack (KNCs only) at $\sim$ 75 kW



Xeon Phi / Knights Corner



source: intel.com

- QPACE 2 uses version 7120X: 61 cores @ 1.238 GHz
- 512-bit wide SIMD units (one per core)
- 16 GB memory (170 GB/s sustained)
- 512 kB private L2 cache per core (unified with distributed tag directory)
- peak DP performance 1.2 TFlop/s

Outline

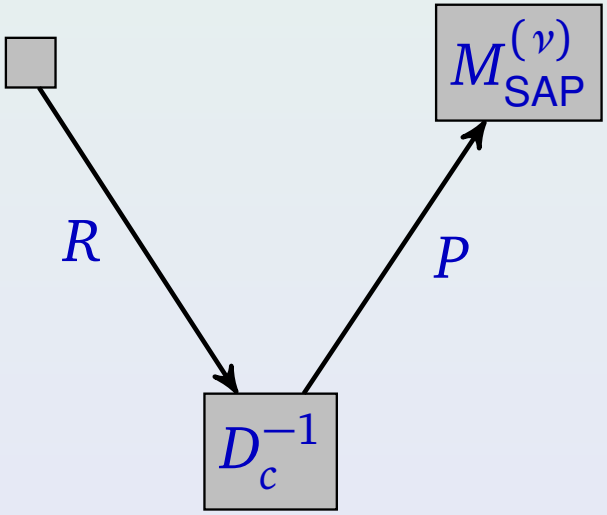
Overview of DD-αAMG

- Dirac operator is Wilson clover
- lattice volume V is divided into N_{block} blocks with $V_{\text{block}} = V/N_{\text{block}}$
  - each block consists of two aggregates that contain the left- and right-handed spinor components, respectively
- outer solver: FGMRES with deflated restarts Frommer et al. 1204.5463
- MG method consists of two parts
  - setup
  - solve (application of MG preconditioner in every FGMRES iteration)

Algorithm 1: MG preconditioner (V-cycle)

Input: right-hand side y
Output: approximate solution x of $Dx = y$

1 apply coarse-grid correction to y (Alg. 2)
2 apply smoother to y, with result from coarse-grid correction as starting guess (Alg. 3)
3 set x to result of smoother



Algorithm 2: Coarse-grid correction

Input: right-hand side y
Output: approximate solution x of $Dx = y$

1 restrict vector y from fine to coarse grid:

$$y_c = Ry$$

2 Coarse-grid solve to low precision using FGMRES with even/odd preconditioning:

$$x_c \approx D_c^{-1} y_c$$

3 prolongate solution vector from coarse to fine grid:

$$x = Px_c \quad \text{with} \quad P = R^\dagger$$

- x should approximate low-mode content of true solution
- this is accomplished by a suitable prolongation operator P (computed in setup phase), which also determines $D_c = RDP$

Algorithm 3: Smoother (DD)

Input: right-hand side y, starting guess $x^{(0)}$
Output: approximate solution $x^{(\nu)}$ of $Dx = y$

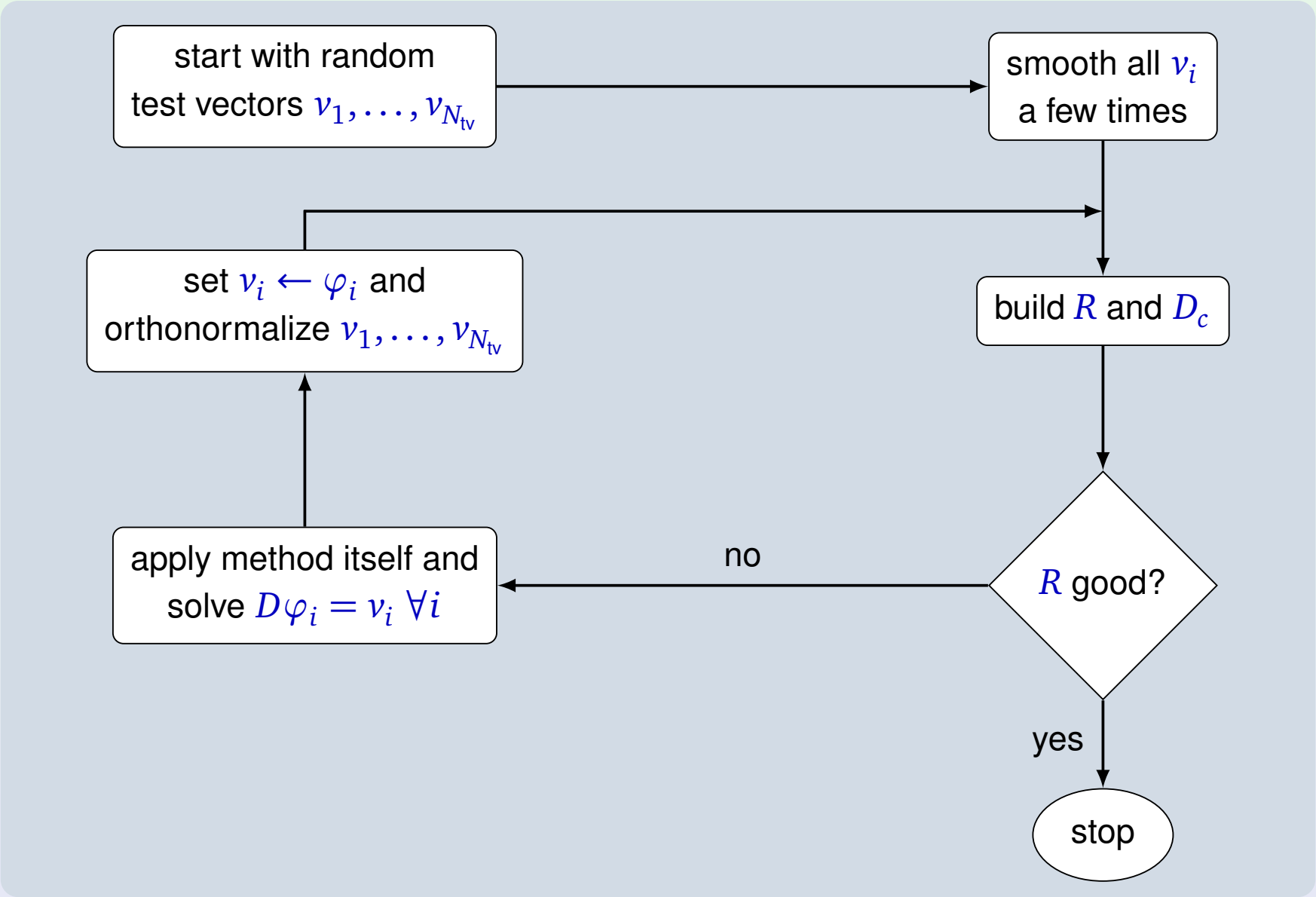
1 split lattice into blocks
2 write $D = B + Z$ with $B =$ couplings within blocks and $Z =$ couplings between blocks
3 **for** $n = 1$ **to** ν **do**
4 $\quad x^{(n)} = x^{(n-1)} + B^{-1}(y - Dx^{(n-1)})$ *// simplified; in practice SAP is used*

- output should approximate high-mode content of true solution
- inversion of B done by minimal residual (MR)
  - choose block size so that block solve runs from cache
  - even/odd preconditioning on blocks

MG setup

- empirical observation: the $O(V)$ low Dirac eigenmodes are locally coherent Lüscher 0706.2298
- aim of MG setup: construct a subspace (of dimension N_{tv}) that approximates the "near-null space"
  - define a set of test vectors $\{v_j\}$ ($j = 1, \ldots, N_{\text{tv}}$)
  - start with random vectors and apply an iterative process through which high-mode components are successively damped
- setup is expensive and can dominate execution time if only few solves are done
  - important to optimize setup
  - for case of few solves, could make setup less optimal
    $\rightarrow$ setup cheaper, solve more expensive, faster overall Osborn 1011.2775

MG setup flow diagram



source: Matthias Rottmann

Algorithm 4: MG setup

Input: none

Output: restriction operator R and coarse-grid operator D_c

// Initial setup:

1 set N_{tv} test vectors to random starting vectors

2 **for** $k = 1$ **to** 3 **do**

3 update each test vector by applying smoother with $\nu = k$, with starting guess 0

4 setup of restriction and coarse-grid operator (Alg. 5)

5 normalize the test vectors

// Iterative refinement:

6 **for** $i = 1$ **to** N_{setup} **do**

7 update each test vector by applying V-cycle (Alg. 1)

8 setup of restriction and coarse-grid operator (Alg. 5)

Algorithm 5: Setup of restriction and coarse-grid operator

Input: test vectors $\{v_j\}$
Output: restriction operator R and coarse-grid operator D_c
// Setup of restriction operator:
1 **for** $i = 1$ **to** N_{block} **do**
2 **foreach** $h = \ell, r$ **do**
3 set R_i^h to $N_{\text{tv}} \times 6V_{\text{block}}$ matrix having in its rows the vectors $v_j^\dagger$ restricted to aggregate A_i^h
4 run Gram-Schmidt on the rows of R_i^h
// Setup of coarse-grid operator:
5 compute

$$D_c = RDR^\dagger$$

Outline

General strategy

- select most suitable algorithm for given hardware
- adapt data layout to make optimal use of hardware (cache and vector units) and to minimize data movement (memory and network)
- identify main contributors to wall-clock time and optimize them
  - vectorization
  - cache management and prefetching
  - intra-core threading
  - inter-core parallelization
  - multi-node implementation
- some extra tricks
  - half-precision storage (for some objects)
  - multiple RHS
  - communication latency hiding
  - software prefetching

Miscellanei

- SIMD implementation
  - originally based on intrinsics for Intel compiler
  - now same API for intrinsics and auto-vectorization by compiler
  - real and imaginary parts are not mixed in same register
- threading
  - lattice is decomposed into pieces that are assigned to individual threads
  - done with OpenMP
  - we use persistent threads with synchronization points
    (better performance than fork-join)
- multi-node communication
  - originally based on Intel MPI
  - now being replaced by high-performance comms library pMR (Peter Georg)

Vectorization: the threefold way

- before working on DD-αAMG, Simon Heybrock developed optimized DD-based solver for 512-bit SIMD/KNC Heybrock et al. 1412.2629
  - used as smoother on fine grid in DD-αAMG
  - vectorization by site-fusing (combine several lattice sites in SIMD unit)
  - SOA data layout (domain boundary data also in AOS)
  - `https://rqcd.ur.de:8443/hes10653/mic-qcd-solver`
- remaining parts of DD-αAMG are easier to vectorize:
  - number of components that can be treated on same footing contains factor of N_{tv} (on fine grid) or $2N_{\text{tv}}$ (on coarse grid)
  - choose this factor to be an integer multiple of SIMD length N_{SIMD} $\rightarrow$ perfect use of SIMD unit
  - if not an integer multiple, part of SIMD unit is wasted in last iteration
- when multiple RHS are present: better to vectorize over RHS (requires change in data layout)

Current status

- 2-level DD-αAMG fully vectorized
- most optimizations completed
- multiple RHS implemented in iterative setup phase (coarse grid)
- several improvements in comms
  - threading of copying from/to comm buffers
  - MRHS implementation leads to fewer and larger messages
  - MPI replaced by pMR for halo exchange on coarse grid

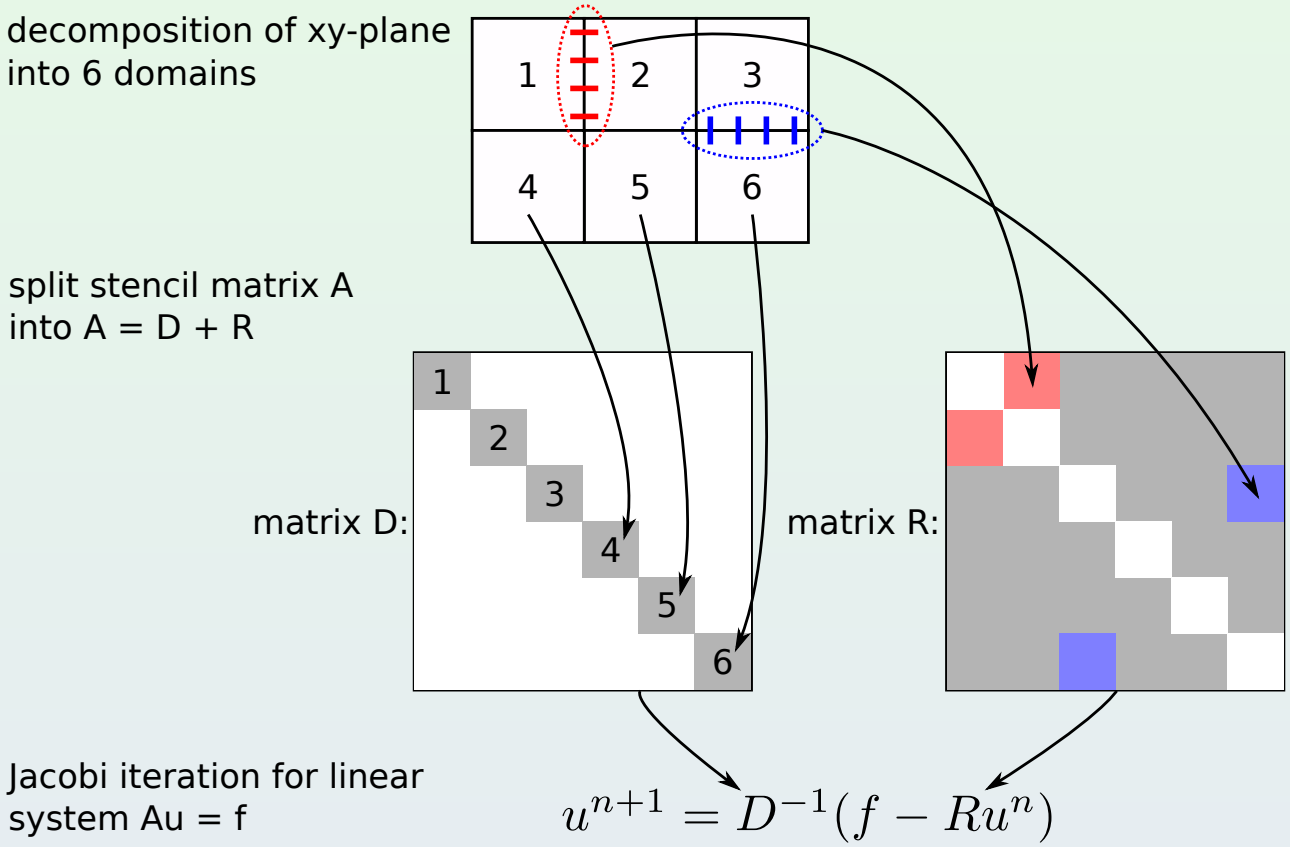
TBD

- DD-preconditioning of FGMRES on coarse levels
- multiple RHS for smoother in setup phase
  - vectorization over test vectors
- multiple RHS for smoother on fine grid (for analysis)
  - vectorization over RHS
- optimized DD-based smoother on coarse levels (for multi-level algorithm)
  - SRHS: vectorization over test vectors
  - MRHS: vectorization over RHS
- further optimization of comms
  - pMR also for global sums
  - replace all performance-relevant MPI calls by pMR calls
    (MPI could be still be used as a provider within pMR)
- How to optimally map coarse grid(s) to machine partition?

Outline

Outline

Domain Decomposition



- main idea: Schwarz (1870), Lüscher hep-lat/0310048
  - subdivide lattice into domains and reorder indices → block-diagonal + rest
  - inversion only on domains (no communication required, ideally from cache)
  - rest (application of R) needs comms but does not occur frequently

→ less communication, better latency tolerance, more cache reuse

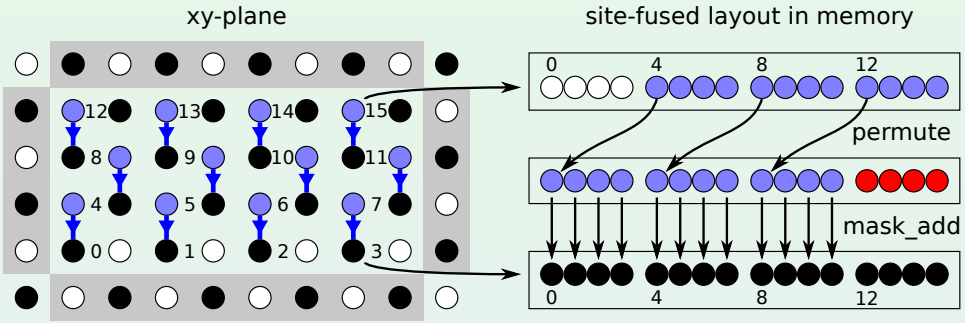
Data layout and mapping to SIMD

- goals:
  - avoid loading cache lines that are only partially needed
  - use all SIMD elements
  - avoid instruction overhead due to permutations (for complex arithmetic)
- our solution:
  - structure-of-array (SOA) format, i.e., all 24 floating-point components of a spinor are stored in 24 separate registers and cache lines
  - this leads to "site fusing": 16 lattice sites in one 512-bit register (for SP) in our case: 4×4 sites per register in x and y direction
- computation of hopping terms:
  - straightforward in z and t direction
  - in x and y, use permute/mask → wastes 12.5% (25%) of SIMD units in x (y)
  - in site-fused dimensions, hopping terms between domains would give large overhead:
    - need to load cache line with neighbor's boundary data
    - but this cache line contains extra data that are not needed

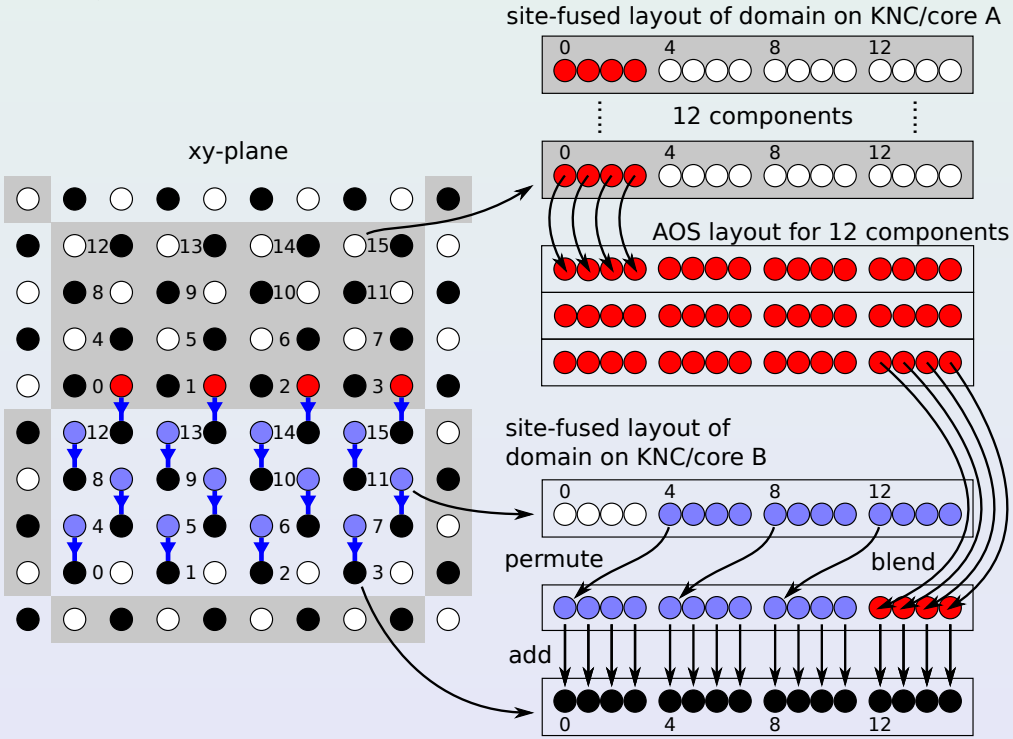
    → additionally store boundary data in array-of-structure (AOS) format

Permuting/masking and repacking of boundary data

- permuting/masking:



- repacking of boundary data:



Cache management and prefetching

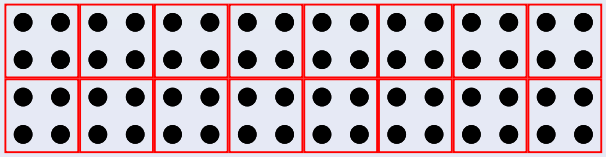
- one domain per core since L2 is not shared
- cache size (512 kB/core) restricts domain size to 8×4^3 (in SP)
- KNC can do up/down-conversion between half/single on load/store
  $\rightarrow$ store (some) domain data in half precision
  $\rightarrow$ reduced working set and reduced bandwidth requirements
  - to ensure stability, spinors are kept in single precision
  - gauge links and clover matrices in half precision
  - no noticeable impact on iteration count of outer solver
- prefetching:
  - no L1 hardware prefetcher
  - L2 hardware prefetcher only for streaming access
  - compiler-generated software prefetches often not good enough
  - manual L1 and L2 prefetches essential (using intrinsics)
  - fine-tuning of prefetches rather time-consuming

Intra-core threading

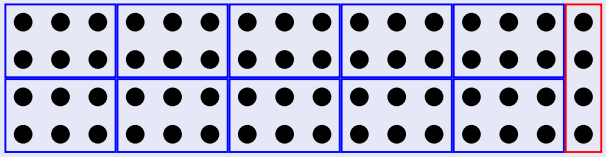
- need at least two threads per core for full pipeline utilization
- we assign threads to alternating time slices within domain
- we see no significant differences between two or four threads per core:
  - two threads: more stalls due to latency of L1 or L2 misses
  - four threads: working set exhausts L1 size
    $\rightarrow$ threads evict each other's data more frequently

Inter-core parallelization

- using OpenMP
- recall: one domain per core
- synchronization between cores only necessary after MR block solve (MR = inversion on domains)
  $\longrightarrow$ cost of barrier has no significant impact
- load-balancing issues with standard lattice sizes (2^n) on 60 cores (some cores would be unused)
  - simple issue but significant impact on performance
  - possible workarounds:
    - use prime factors of 3 and 5 in lattice sizes (for new lattices)
    - non-uniform partitioning of the lattice
      example: processors with 6 cores each, 4×16 lattice



$2 \cdot 8 = 16$ processors
$(16 \cdot 2 = 32$ cores unused$)$



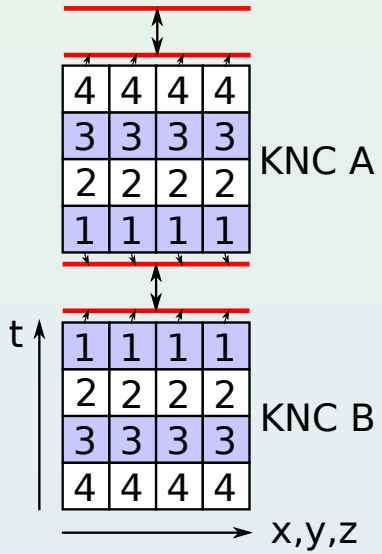
$2 \cdot 5 + 1 = 11$ processors
(2 cores unused)

Multi-node implementation

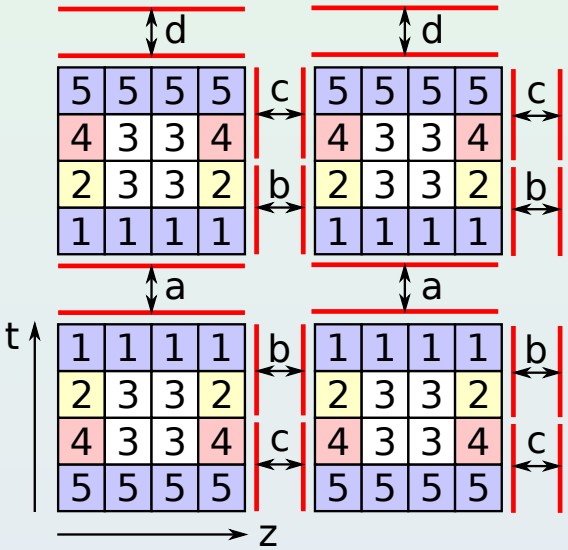
- could have each thread issue its own MPI calls, but:
  - typically high overhead for MPI calls from several threads
  - message sizes too small for efficient network utilization
- better:
  - combine surface data of all domains and communicate them using a single thread
  - needs explicit on-chip synchronization
- hiding communication behind computation is important (even for DD)
  - standard method (divide local volume into interior and surface) does not work for us since most domains would be on the surface
  - instead, send boundary data when half of them are ready

Communication latency hiding for DD

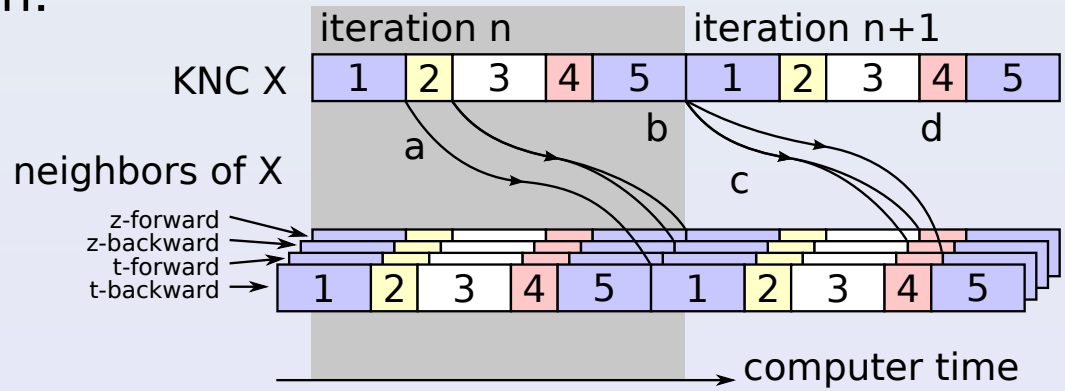
- boxes represent domains, numbers represent order of execution, small letters represent order of communication
- bad:



good:



- linear representation:

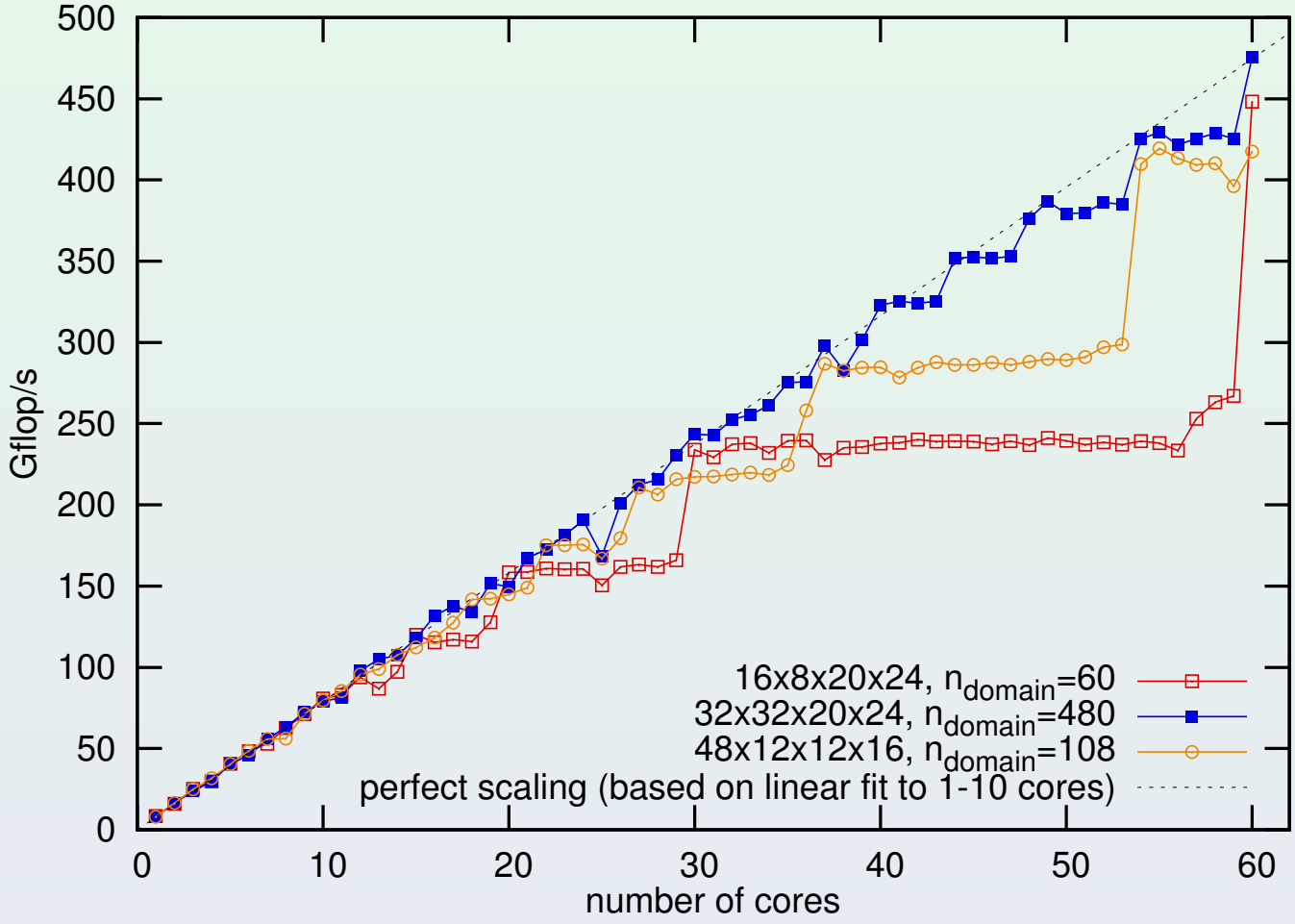


Results: Single core

- theoretical performance model predicts 56% of peak = 20 GFlop/s/core
- actual MR performance $\sim$ 12 GFlop/s on single core
  main culprit (VTune): stalls due to outstanding L1 prefetches
- optimal number of MR iterations (for minimal time-to-solution) is only 4$\sim$5
  $\rightarrow$ other parts of Schwarz method contribute significantly
  $\rightarrow \sim$ 8 GFlop/s/core
- single-core performance in GFlop/s:

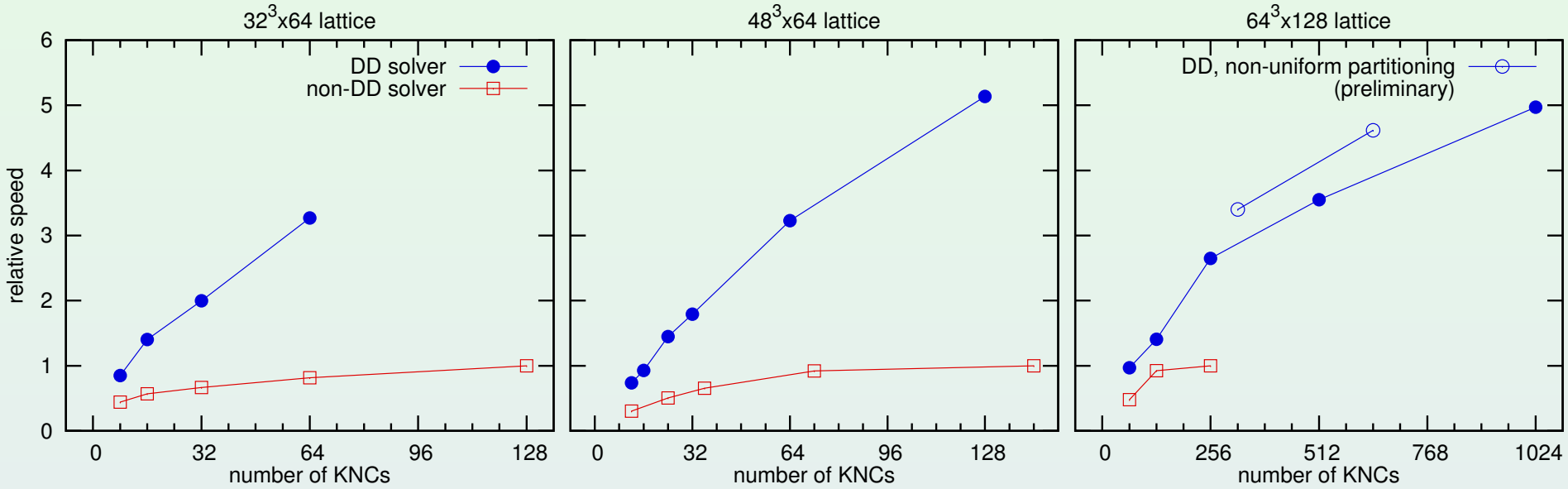
	MR iteration		DD method	
	single	half	single	half
no software prefetching	5.4	7.9	4.1	5.9
L1 prefetches	9.2	11.8	5.8	7.7
L1+L2 prefetches	9.1	11.8	6.3	8.4

Results: Single KNC



- almost perfect scaling (except for load imbalance):
  - cores can work independently during MR inversion
  - almost no competition for memory access since MR runs from cache

Results: DD strong scaling (on Stampede)



- $m_\pi = 290$ MeV, 150 MeV (QCDSF), SU(3) point ($\sim$ 800 MeV) (USQCD)
- results normalized to minimum time-to-solution for non-DD solver (BiCGstab)
- DD strong-scales to more nodes (also better for equal number of nodes)
- performance drop for large number of nodes:
  - overlapping communication with computation becomes harder/impossible
  - message sizes are smaller $\rightarrow$ less efficient network communication
- max. number of nodes is determined by local volume:
  - if domains too small $\rightarrow$ DD less efficient
  - we don't split domains over cores (no shared L2)

Outline

Application of restriction operator (single RHS)

- compute $y_c = Ry$ with $R = \mathrm{diag}(R_1^\ell, R_1^r, \ldots, R_{N_{\text{block}}}^\ell, R_{N_{\text{block}}}^r)$
- $\dim(R_i^{\ell,r}) = N_{\text{tv}} \times 6V_{\text{block}}$ → vectorize over row index:

Algorithm 6: SIMD implementation of Ry

```
for i = 1 to N_block do
    foreach h = ℓ, r do
        set (y_c)_i^h = 0 in SIMD vectors (real and imaginary part)   // dim(y_c)_i^h = N_tv
        for n = 1 to 6V_block do                                      // work on aggregate A_i^h
            load real and imaginary part of column n of R_i^h into SIMD vectors
            broadcast real and imaginary part of corresponding element of y into
            SIMD vectors
            increase (y_c)_i^h by complex fused multiply-add (corresponding to 4 real
            SIMD fmadds)
        write (y_c)_i^h to memory
```

- row index of R_i^h runs in SIMD vector (latter contains column of R_i^h if $N_{\text{tv}} = N_{\text{SIMD}}$)

Application of restriction operator (single RHS)

$$\left((y_c)_i^h\right)_m = \sum_n (R_i^h)_{mn} (y_i^h)_n \qquad \begin{cases} h = \ell, r \\ i = 1, \ldots, N_{\text{block}} \\ m = 1, \ldots, N_{\text{tv}} \\ n = 1, \ldots, 6V_{\text{block}} \end{cases}$$

SIMD FMA:

$$\begin{array}{c} \left((y_c)_i^h\right)_1 \\ \hline \left((y_c)_i^h\right)_2 \\ \hline \left((y_c)_i^h\right)_3 \\ \hline \vdots \end{array} \quad += \quad \begin{array}{c} (R_i^h)_{1n} \\ \hline (R_i^h)_{2n} \\ \hline (R_i^h)_{3n} \\ \hline \vdots \end{array} \quad * \quad \begin{array}{c} (y_i^h)_n \\ \hline (y_i^h)_n \\ \hline (y_i^h)_n \\ \hline \vdots \end{array}$$

Prolongation (single RHS)

- similar to restriction but with $R \rightarrow P = R^\dagger$
- aspect ratio of rectangular matrix is reversed
  $\rightarrow$ now column index of $P_i^{\ell/r}$ ($=$ row index of R) runs in SIMD vector
- at the end, require additional sum over elements in SIMD vector
  $\rightarrow$ prolongation somewhat less efficient than restriction

Setup of coarse-grid operator

- $D_c = RDP$ in detail:

$$\begin{pmatrix} D_c^{\ell\ell} & D_c^{\ell r} \\ D_c^{r\ell} & D_c^{rr} \end{pmatrix}_{ij} = \begin{pmatrix} R_i^\ell & 0 \\ 0 & R_i^r \end{pmatrix} \begin{pmatrix} D_{ij}^{\ell\ell} & D_{ij}^{\ell r} \\ D_{ij}^{r\ell} & D_{ij}^{rr} \end{pmatrix} \begin{pmatrix} P_j^\ell & 0 \\ 0 & P_j^r \end{pmatrix}$$

  - i and j are equal or nearest neighbors, and run from 1 to N_{block}
  - $(D_c)_{ij}$ computed for $i = j$ and forward neighbors
  - for backward neighbors use $(D_c)_{ji}^{hh} = (D_c)_{ij}^{hh\dagger}$ and $(D_c)_{ji}^{hh'} = -(D_c)_{ij}^{h'h\dagger}$ $(h \neq h')$
  - D_c is stored in half precision (reduces memory capacity and bandwidth requirements, no impact on algorithmic performance)
  - we also store $(D_c)_{ji}$ since transpose is expensive in SIMD
- first compute $D_{ij}^{hh'} P_j^{h'}$, i.e., sparse matrix applied to multiple vectors (the N_{tv} columns of $P_j^{h'}$) $\rightarrow$ vectorize over column index (Alg. 7)
- application of R to result = restriction with multiple RHS ($\#\text{RHS} = N_{\text{tv}}$) $\rightarrow$ vectorize over RHS

Algorithm 7: SIMD implementation of $D_{ij}^{hh'} P_j^{h'}$

```
for x ∈ block i do
    set output = 0 in SIMD vectors (real and imaginary parts)
    foreach μ ∈ {±1, ±2, ±3, ±4} do
        if x + μ̂ ∈ block j then
            load real and imag. parts of the 6 rows of P_j^{h'} corresponding to x+μ̂
            into SIMD vectors
            broadcast real and imag. parts of the 9 elements of SU(3) link U_μ(x)
            into SIMD vectors
            increase output by complex fmadd (1 + γ_μ)^{hh'} U_μ(x)^† P_j^{h'}(x + μ̂)
    if i = j and h = h' then
        load real and imaginary parts of the 6 rows of P_i^h corresponding to x into
        SIMD vectors
        broadcast real and imaginary parts of the clover matrix elements C^{hh}(x)
        into SIMD vectors
        increase output by complex fmadd C^{hh}(x) P_i^h(x)
```

Restriction with multiple RHS

$$\left((y_c)_i^h\right)_m^{(j)} = \sum_n (R_i^h)_{mn} (y_i^h)_n^{(j)} \qquad \begin{cases} h = \ell, r \\ i = 1, \ldots, N_{\text{block}} \\ m = 1, \ldots, N_{\text{tv}} \\ n = 1, \ldots, 6V_{\text{block}} \\ j = 1, \ldots, \#\text{RHS} \end{cases}$$

SIMD FMA:

$$\begin{array{c} \left((y_c)_i^h\right)_m^{(1)} \\ \hline \left((y_c)_i^h\right)_m^{(2)} \\ \hline \left((y_c)_i^h\right)_m^{(3)} \\ \hline \vdots \end{array} \quad += \quad \begin{array{c} (R_i^h)_{mn} \\ \hline (R_i^h)_{mn} \\ \hline (R_i^h)_{mn} \\ \hline \vdots \end{array} \quad * \quad \begin{array{c} (y_i^h)_n^{(1)} \\ \hline (y_i^h)_n^{(2)} \\ \hline (y_i^h)_n^{(3)} \\ \hline \vdots \end{array}$$

Application of coarse-grid operator

- $(D_c)_{ij} \neq 0$ only if i and j are equal or nearest neighbors then $(D_c)_{ij}$ is dense and stored in memory
- $\dim(D_C)_{ij} = 2N_{\text{tv}}$
- vectorization can be done as in the restriction (but different approach/ data layout depending on whether we have SRHS or MRHS)

Gram-Schmidt on aggregrates

- needed to orthonormalize rows of R_i^h
- we do not use modified Gram-Schmidt:
  - classical Gram-Schmidt easier to vectorize, and needs fewer globals sums
  - stability of Gram-Schmidt process not an issue in preconditioner
- we use block Gram-Schmidt method
  - obtains better cache reuse and thus saves memory bandwidth
- vectorization done as before: merge same components of the N_{tv} test vectors in the SIMD vectors
- disadvantage: axpy operations and dot products waste parts (on average one half) of the SIMD vectors

BLAS-like linear algebra on coarse grid

- needed for FGMRES on coarse grid
- SRHS:
  - data layout change on coarse grid would be needed to utilize SIMD unit
  - this change would propagate to other parts of code
    $\longrightarrow$ not done since impact on performance is not dominant
  - temporary workaround: de-interleave real and imaginary parts on the fly to do SIMD computation
- MRHS: no such issues

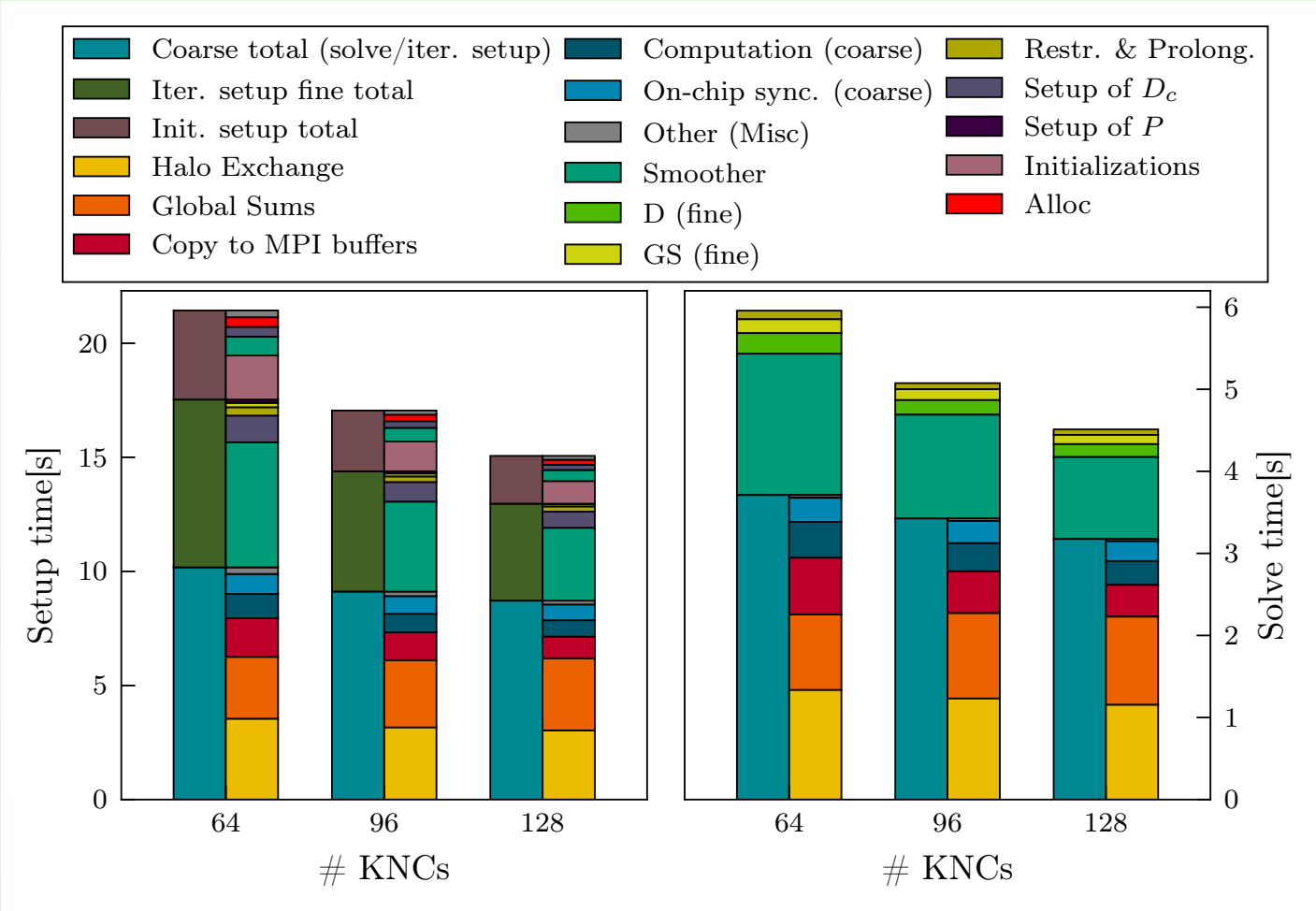
SIMD speedup of DD-αAMG

- speedup factor w.r.t. original Wuppertal code:

MG component	Restrict.	Prolong.	D_c setup	$(D_c)_{i \neq j}$	$(D_c)_{ii}$	GS on aggr.
SIMD speedup	14.1	8.6	19.7	20.2	19.5	10.8

- single core on a single KNC, lattice size $= 8^4$ (does not fit in cache)

Strong scaling of DD-αAMG (single RHS implementation)



- CLS lattice: $48^3 \times 96$, $\beta = 3.4$, $m_\pi = 220$ MeV, $a = 0.086$ fm
  (small lattice size chosen intentionally to see breakdown of strong scaling)
- after optimizations, off-chip communication now dominant

Outline

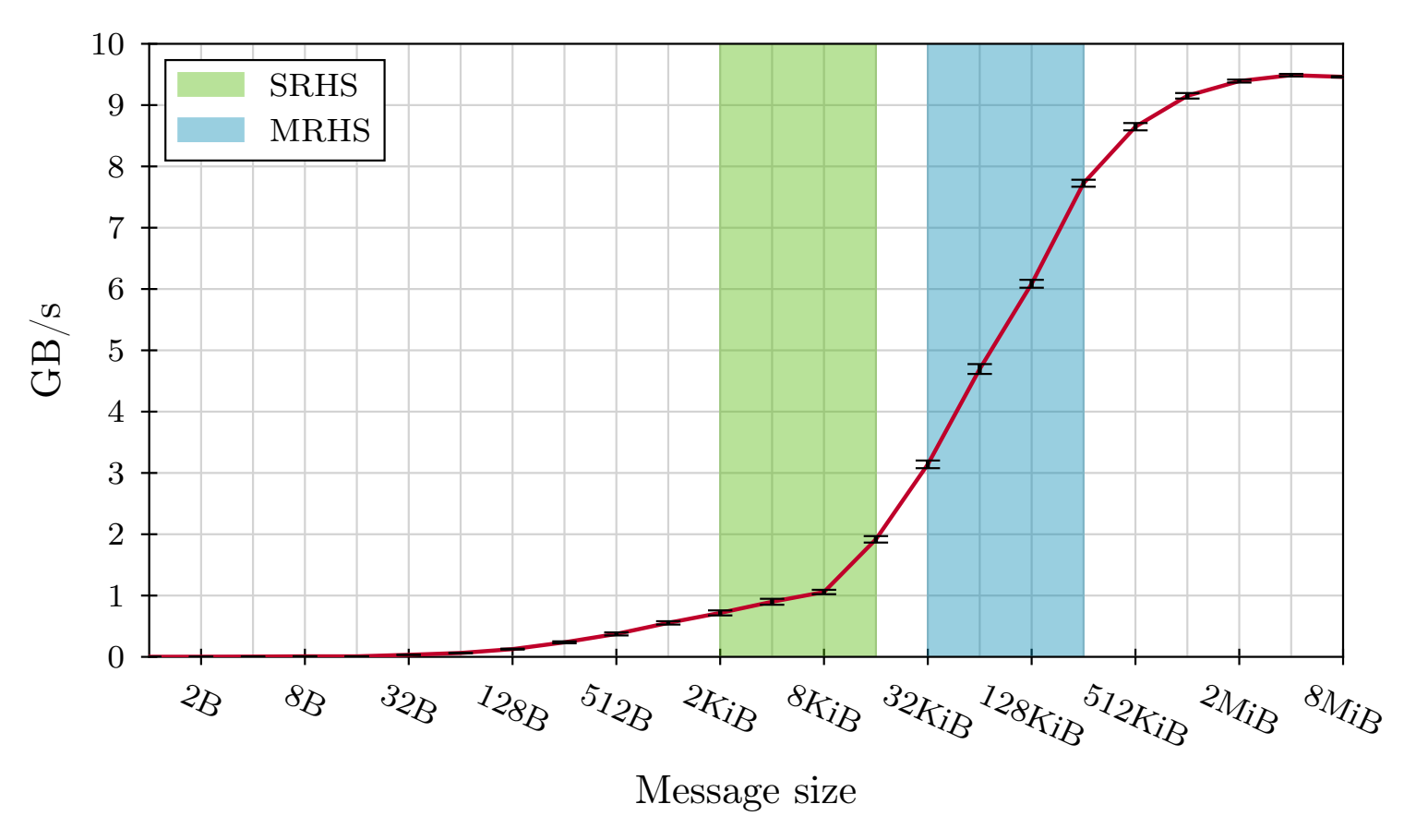
Multiple RHS and optimization of communications (Daniel Richtmann)

- small message sizes imply inefficient network utilization
  $\longrightarrow$ alleviated by multiple RHS (fewer and larger messages)
- so far: V-cycle applied to test vectors in setup sequentially (SRHS)
  $\longrightarrow$ message size on coarse grid:

$$S_\mu = \prod_{\nu=0,\,\nu\neq\mu}^{3} \frac{(\text{local lattice})_\nu}{(\text{domain size})_\nu} \cdot \frac{2N_{\text{tv}}}{2} \cdot 8 \text{ Byte}$$

- now: apply V-cycle to N_{SIMD} test vectors simultaneously (MRHS)
  $\longrightarrow$ message size increases by factor of $N_{\text{SIMD}} = 16$

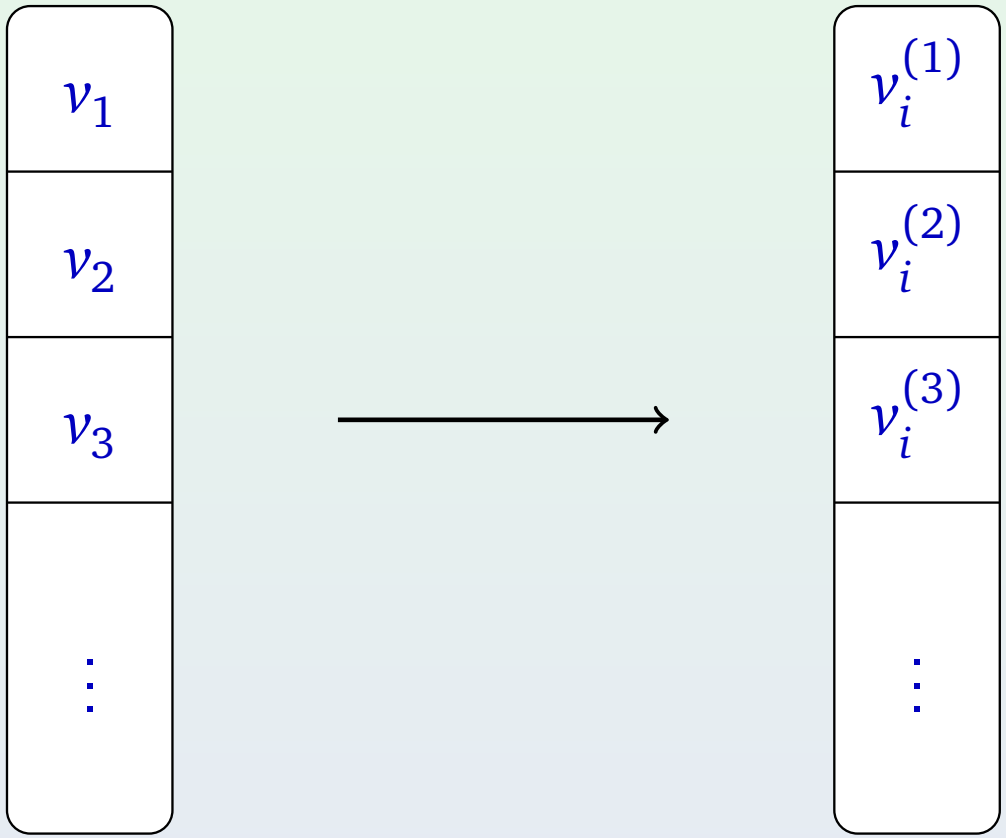
Impact of MRHS on effective network bandwidth



network bandwidth between two KNCs in QPACE 2 via FDR InfiniBand

MHRS: Change in data layout and mapping to SIMD

- change in data layout:



- yields more natural mapping to SIMD and performance gains
  - BLAS-like linear algebra (e.g., vector adds) vectorized trivially
  - de-interleaving overhead eliminated
  - no data dependencies of individual entries in registers
    $\longrightarrow$ reduction operations over elements in register no longer needed

Further benefits of MRHS

- arithmetic intensity of dense complex matrix-vector multiplication
  - SRHS: $\sim$ 32 Byte/cycle per core $\rightarrow$ 2377 GB/s on KNC
  - MRHS: $\sim$ 2 Byte/cycle per core $\rightarrow$ 149 GB/s on KNC

  KNC memory bandwidth is about 170 GB/s

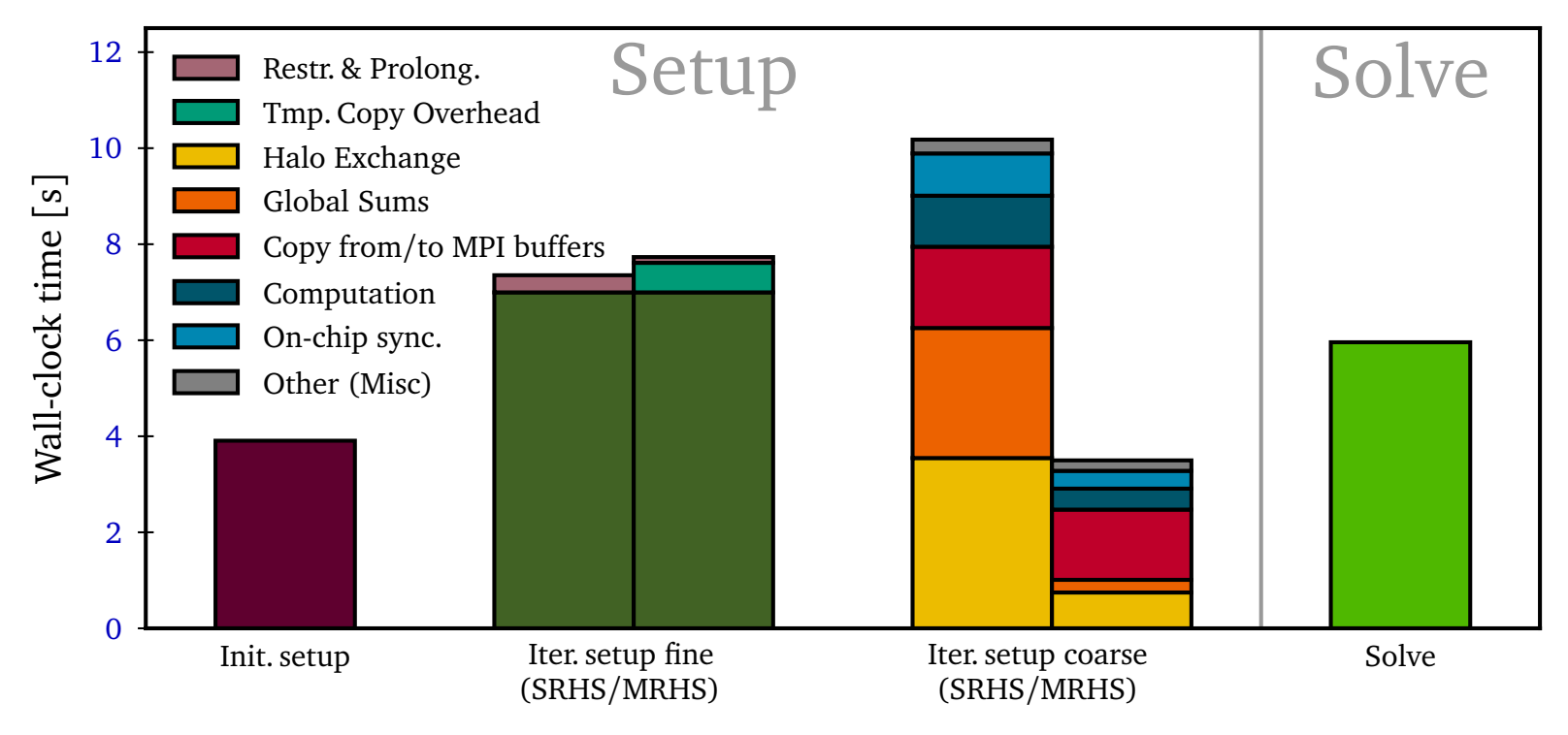
  $\rightarrow$ MRHS no longer memory-bandwidth bound
- fewer calls to barriers $\rightarrow$ less synchronization overhead

MRHS performance gains

projection operators	2.9x
coarse-grid computation	2.4x
on-chip synchronization	2.7x
halo exchange	4.7x
global sums	10.3x
coarse-grid total	2.9x

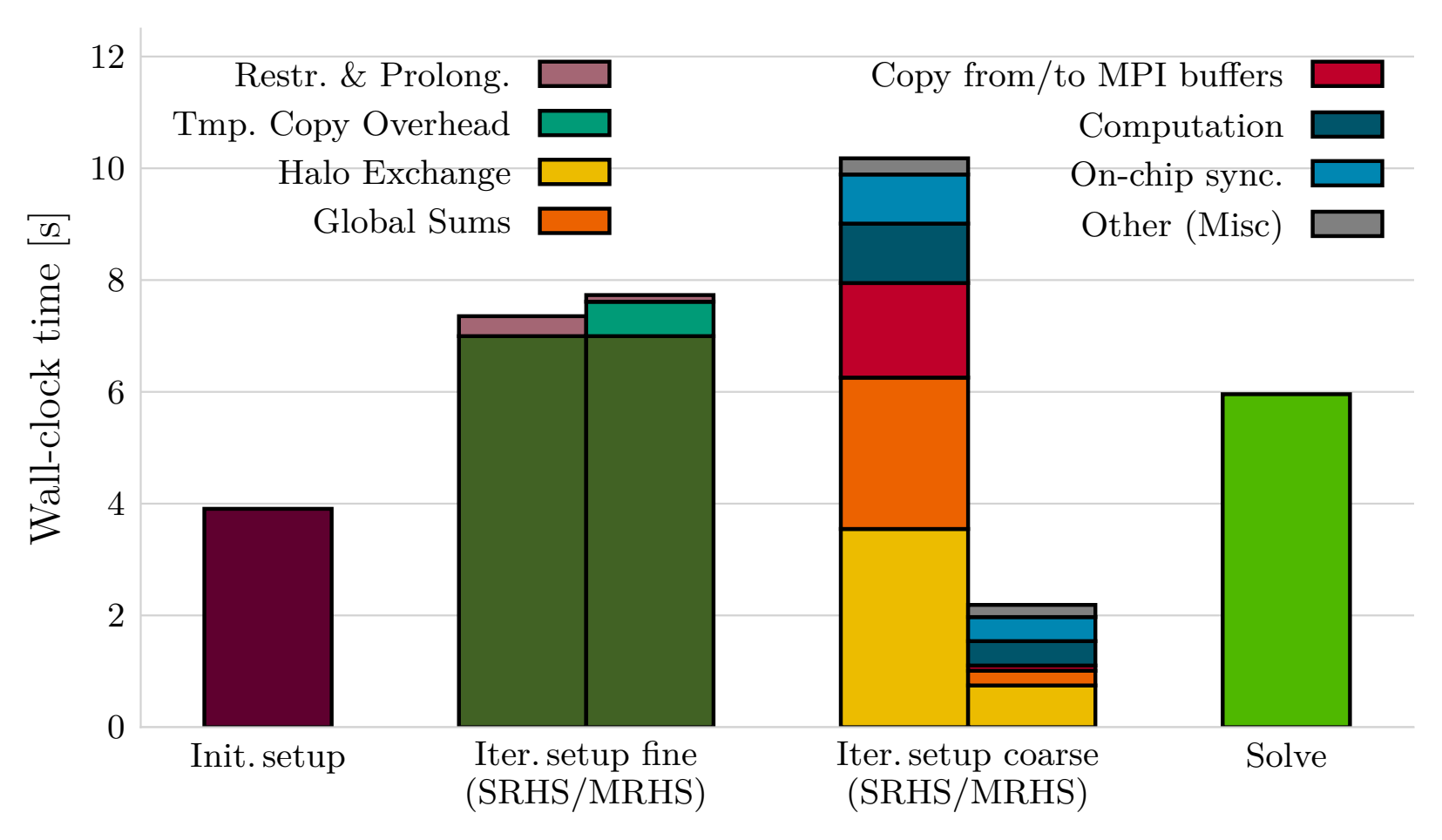
Impact of MRHS on wall-clock time

- so far, MHRS implemented in coarse-grid solve and projection operators
- MRHS implementation of smoother in progress
- results on 64 KNCs (with parameters tuned for SRHS setup):



- here, copy operations to/from comm. buffers were not threaded yet

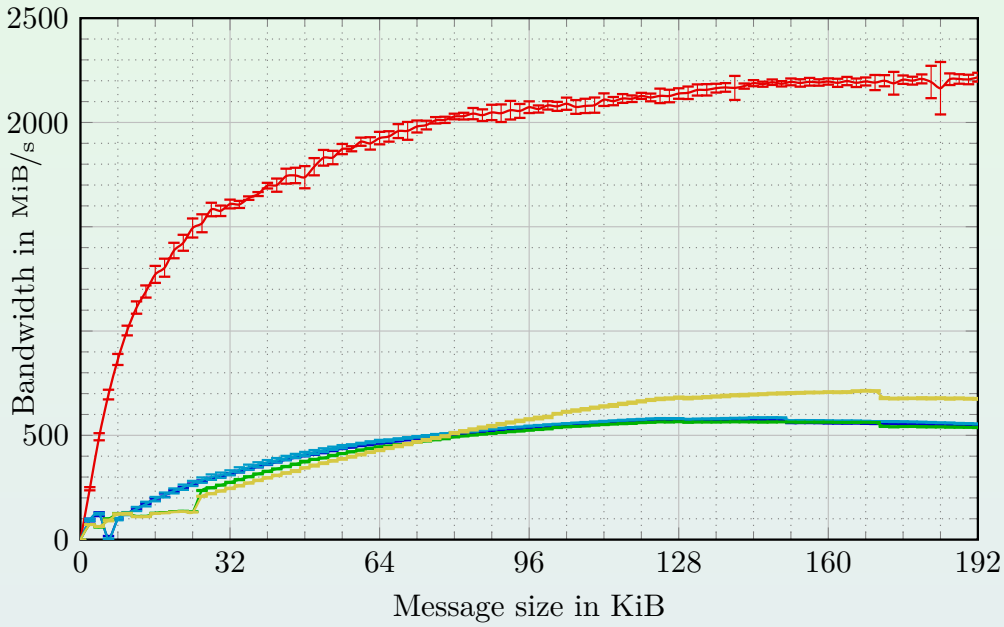
MRHS + threading of copy to/from comm. buffers



Further comms optimization: pMR

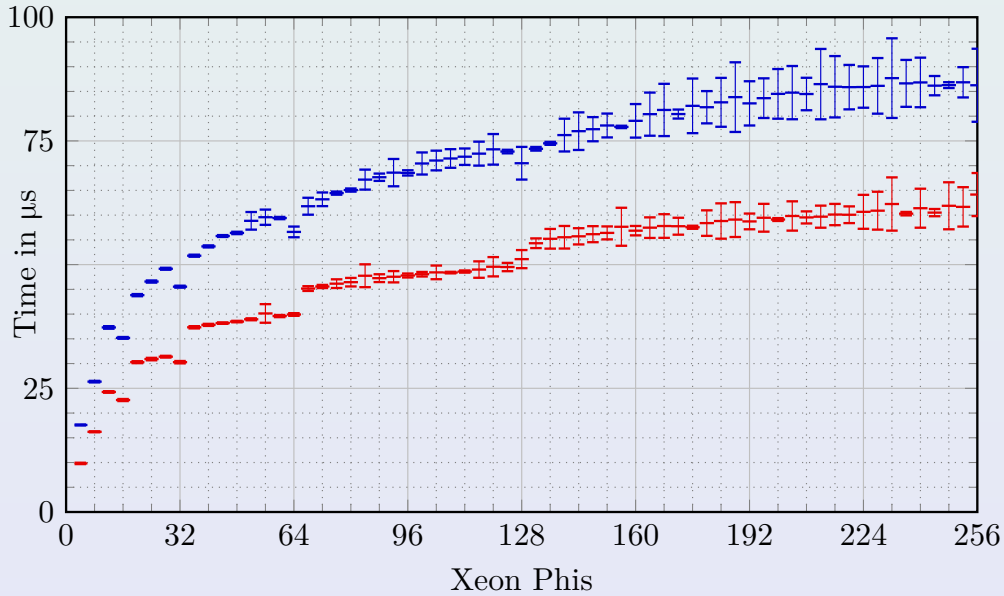
- switch from MPI to high-performance communications library pMR for performance-relevant parts
  - for details see poster
  - persistent, one-sided communication (RDMA)
  - written in C++11
  - separate code for supported network providers (IB verbs, Linux CMA)
  - supports exotic network topologies
  - so far only implemented for halo exchange on coarse grid
- check out now from `https://rqcd.ur.de:8443/gep21271/pmr` will be put on github for contributions

Synthetic benchmarks



4d halo exchange on 224 KNCs

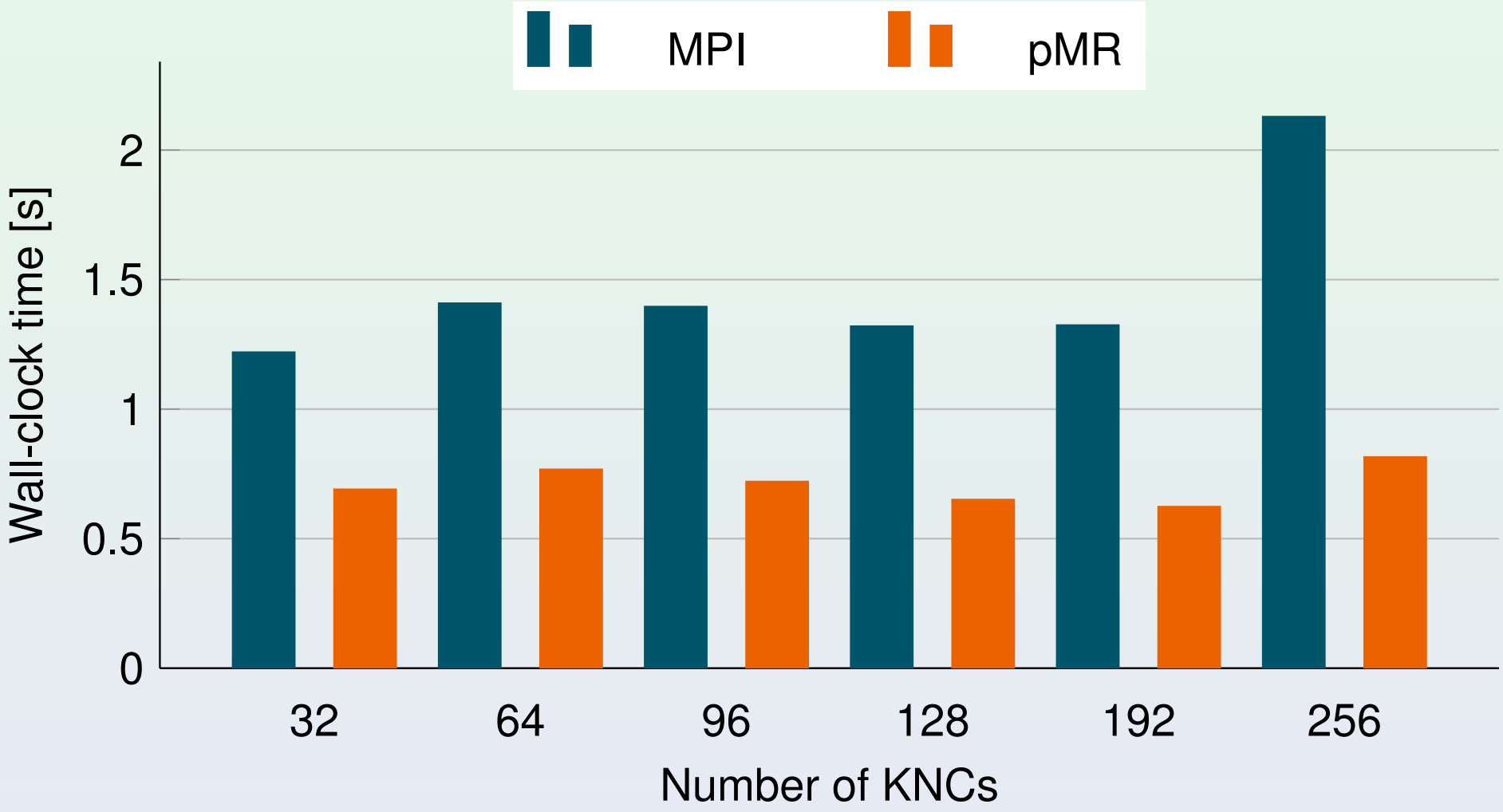
- red: pMR
- blue: Intel MPI (DAPL, port 1)
- cyan: Intel MPI (DAPL, port 2)
- green: Intel MPI (OFA, one port)
- yellow: Intel MPI (OFA, two ports)



global sum

- red: pMR
- blue: Intel MPI

Halo exchange on coarse grid for one solve



- CLS lattice: $48^3 \times 96$, $\beta = 3.4$, $m_\pi = 220$ MeV, $a = 0.086$ fm
  (small lattice size chosen intentionally to see breakdown of strong scaling)

Outline

From KNC to KNL

- porting has started very recently, so no real results yet (only synthetic benchmarks)
- KNL supports different memory and cluster modes
  - MCDRAM: flat, cache, hybrid
  - cluster: quadrant, SNC-4 (and others)
- two vector units per core (instead of one)
  - memory bandwidth went up by about 2.5x
  - but only 16GB fast memory (MCDRAM)
- barriers still slow: O(10,000) cycles for 64 cores with two threads each
- cores can now do hardware prefetching
  $\rightarrow$ software prefetching efforts should be eliminated (or reduced?)
- half precision:
  - Xeon Phi does not have HP arithmetic instructions, but storing some objects in HP is still beneficial (memory capacity/bandwidth)
  - KNC ISA has up/down conversion on load/store, no longer present on KNL
  - instead, use combination of AVX-512 intrinsics

Outline

Conclusions

- DD-αAMG is a good target for SIMD architectures
- optimized 2-level implementation on QPACE 2/KNC mostly completed
  - most important missing piece: MRHS for smoother
- high-performance comms library pMR
- TBD:
  - optimize multi-level implementation
  - port to QPACE 3/KNL
  - optimal mapping of coarse level(s) to machine partitions

Induced QCD with two bosonic flavors

Tilo Wettig

in collaboration with Bastian Brandt and Robert Lohmayer

Department of Physics
University of Regensburg

QCDNA 2016, Edinburgh

arXiv:1411.3350, arXiv:1511.08374

Outline

Outline

Strong-coupling expansion

- action of lattice QCD (sum over colors and flavors implied):

$$S = S_{\text{YM}} + S_f$$
$$= -\frac{\beta}{N_c} \sum_p \operatorname{Re}\operatorname{tr} U_p + \sum_x \left[\bar{\psi}_x \psi_x - \kappa \sum_{\pm\mu} \bar{\psi}_{x+\mu} (1+\gamma_\mu) U_{x\mu} \psi_x \right]$$

  with $\beta = 2N_c/g^2$ and $\kappa = 1/(2m+8)$
- lattice perturbation theory is an expansion about the $g = 0$ limit
- there is another limit about which perturbation theory can be set up: $g = \infty$ or, equivalently, $\beta = 0$

  $\longrightarrow$ strong-coupling expansion
  - can be done to almost arbitrary orders by computing group integrals (doable since at $\beta = 0$ gauge fields appear only linearly, while $U_p \sim U^4$)
  - leads to novel simulation algorithms
  - known since the 1970s
  - still of interest in cases where standard Monte Carlo simulations are not possible (e.g., sign problem at $\mu \neq 0$)

Induced QCD

- to use the strong-coupling techniques also away from $\beta = 0$, one should linearize the plaquette action
- this can be done, e.g., by rewriting the plaquette term as an integral over suitable auxiliary fields to which U couples linearly ("induced QCD")
- several ways to do this have been proposed in the 1980s and 1990s:
  - Bander 1983 (auxiliary scalar fields)
  - Hamber 1983 (auxiliary Wilson fermions)
  - Hasenfratz-Hasenfratz 1992 (like Hamber + 4-fermion interaction)
  - Kazakov-Migdal 1993 (adjoint scalars, no YM limit but useful for large N_c)
- in the cases that reproduce YM theory in the continuum limit, this requires taking the combined limit $N_f \to \infty$ and $\kappa \to 0$ such that $N_f \kappa^4 = \text{const} \propto \beta$ (with N_f = number of auxiliary flavors)
  $\to$ inconvenient (need many auxiliary flavors and extrapolation)
- new idea by Budczies-Zirnbauer (2003) requires only small number of auxiliary bosons and no extrapolation

Outline

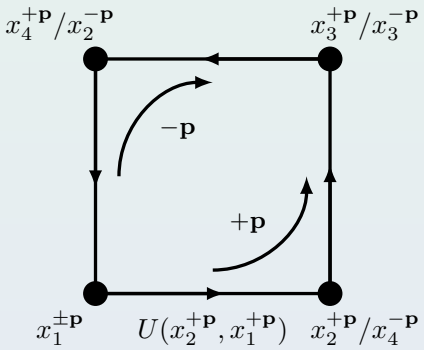
Outline

Budczies-Zirnbauer action

- Budczies-Zirnbauer (math-ph/0305058) propose a "designer action"

$$S_{\mathrm{BZ}}[\varphi,\bar{\varphi},U] = \sum_{b=1}^{N_b}\sum_{\pm\mathbf{p}}\sum_{j=1}^{4}\Big[m_{\mathrm{BZ}}\bar{\varphi}_{b,\mathbf{p}}(x_j^{\mathbf{p}})\varphi_{b,\mathbf{p}}(x_j^{\mathbf{p}}) - \bar{\varphi}_{b,\mathbf{p}}(x_{j+1}^{\mathbf{p}})U(x_{j+1}^{\mathbf{p}},x_j^{\mathbf{p}})\varphi_{b,\mathbf{p}}(x_j^{\mathbf{p}})\Big]$$

  - the φ are auxiliary boson fields and carry an (oriented) plaquette index
  - j labels the points of the plaquette
  - the second term corresponds to the hopping of the φ around the plaquette



- integrating out the φ yields the weight factor

$$\omega_{\mathrm{BZ}}[U] = \prod_p \left|\det\left(m_{\mathrm{BZ}}^4 - U_p\right)\right|^{-2N_b}$$

  - product is over unoriented plaquettes
  - $U_p \sim U^4$ is the usual product of the four links around the plaquette

Conventional pure gauge limit

- write weight factor as

$$\omega_{\mathrm{BZ}}[U] \sim \exp\left\{-2N_b \,\mathrm{Re} \sum_p \mathrm{tr} \log\left(1-\alpha_{\mathrm{BZ}} U_p\right)\right\}$$

  with $\alpha_{\mathrm{BZ}} = m_{\mathrm{BZ}}^{-4}$ (allowed range is $m_{\mathrm{BZ}} > 1$ and thus $0 < \alpha_{\mathrm{BZ}} < 1$)

- expand in small α_{BZ} :

$$S_{\mathrm{BZ}}^{\mathrm{eff}}[U] = -2N_b \alpha_{\mathrm{BZ}} \sum_p \mathrm{Re}\,\mathrm{tr}\, U_p + O(\alpha_{\mathrm{BZ}}^2)$$

- this gives the usual Wilson plaquette action if

$$\alpha_{\mathrm{BZ}} \to 0\,,\; N_b \to \infty \quad \text{such that} \quad \beta = 2N_c N_b \alpha_{\mathrm{BZ}} \quad \text{fixed}$$

- however, they can do better

Nontrivial pure gauge limit for U(N_c)

- first consider $d = 2$ and gauge group U(N_c)
- then one can prove that in the limit

$$\alpha_{\mathrm{BZ}} \to 1$$

the BZ theory has a continuum limit that coincides with YM theory, provided that $N_b > N_c$

Outline

Proof of continumm limit

- for $N_b \geq N_c$, weight factor approaches δ-function for $\alpha_{\text{BZ}} \to 1$

$$\lim_{\alpha_{\text{BZ}} \to 1} \frac{\langle f \rangle}{\langle 1 \rangle} = f(\mathbb{1}), \quad \langle f \rangle = \int_G dU_p \, f(U_p) \left| \det\left(1 - \alpha_{\text{BZ}} U_p\right) \right|^{-2N_b}$$

  proof by group theory (character expansion) and some algebra

- thus fluctuations from unity are strongly suppressed
  $\longrightarrow$ diverging correlation length
  $\longrightarrow$ continuum limit (same reasoning as for Wilson action)

- the key was the minus sign in the exponent (resulting from bosons)

- BZ say that the bound $N_b \geq N_c$ is optimal:
  - δ-function is obtained if all irreps r of the group occur in the character expansion with coefficient $c_r = d_r$ (Peter-Weyl theorem)
  - for $N_b < N_c$ BZ say some irreps are missing $\longrightarrow$ no δ-function
  - we will later relax this bound for non-integer N_b
    (we find that irreps are not missing but $c_r \neq d_r$)

Equivalence of continuum limit with YM

- proof by character expansion of the weight function (a class function):

$$\omega(U) = \sum_r c_r \chi_r(U) \quad \text{with} \quad c_r = \int dU\, \omega(U) \chi_r(U^{-1})$$

- for $N_b \geq N_c + 1$, the expansion coefficients have a Taylor expansion in $(1 - \alpha_{\text{BZ}})^2$
- in the limit $\alpha_{\text{BZ}} \to 1$, first nontrivial term leads to boundary-value partition function for a 2-dim. area

$$\Gamma(\mathcal{U}) = \sum_r d_r\, \chi_r(\mathcal{U}) \exp\left\{ -\frac{\mu}{2} \left[\text{Cas}_2(r) + \frac{B_1}{B_2} q(r)^2 \right] \right\}$$

  with $\mu \sim (1 - \alpha_{\text{BZ}})^2$
- this was the starting point of Witten's combinatorial treatment of 2d YM $\to$ equivalence established Witten, Commun. Math. Phys. 141 (1991) 153
- $N_b = N_c$ is a special case (non-renormalizable theory with Cauchy distribution), which does not persist for $d > 2$

From $d = 2$ to $d > 2$

- proof becomes a conjecture
- increasing the number of dimensions also increases the collective behavior of the link variables (more transverse gluons)
- this enhances the universal properties of the microscopic theory
- thus, if we start with a microscopic theory in the YM universality class for $d = 2$, this theory should remain in the YM universality class for $d > 2$

Outline

Spurious sign problem

- the weight factor $\prod_p \left|\det\left(m_{\mathrm{BZ}}^4 - U_p\right)\right|^{-2N_b}$ has no sign problem
- however, the action with auxiliary bosons does: S_{BZ} is generically complex since in

$$\sum_{b=1}^{N_b}\sum_{\pm\mathbf{p}}\sum_{j=1}^{4}\bar{\varphi}_{b,\mathbf{p}}(x_{j+1}^{\mathbf{p}})U(x_{j+1}^{\mathbf{p}},x_j^{\mathbf{p}})\varphi_{b,\mathbf{p}}(x_j^{\mathbf{p}})$$

  the imaginary parts of the terms containing the positively and negatively oriented links do not cancel per configuration, but only after integration
- this can be fixed easily by noting

$$\begin{aligned}\left|\det\left(m_{\mathrm{BZ}}^4 - U_p\right)\right|^2 &= \det\left(m_{\mathrm{BZ}}^4 - U_p\right)\left(m_{\mathrm{BZ}}^4 - U_p^\dagger\right)\\ &\sim \det\left(\bar{m} - \left(U_p + U_p^\dagger\right)\right)\end{aligned}$$

  with $\bar{m} = m_{\mathrm{BZ}}^4 + m_{\mathrm{BZ}}^{-4}$

Modified designer action

- a modified action without sign problem is thus

$$S_B[\varphi,\bar{\varphi},U]=\sum_{b=1}^{N_b}\sum_{p}\sum_{j=1}^{4}\Big[m\bar{\varphi}_{b,p}(x_j^p)\varphi_{b,p}(x_j^p)-\bar{\varphi}_{b,p}(x_{j+1}^p)U(x_{j+1}^p,x_j^p)\varphi_{b,p}(x_j^p)$$
$$-\bar{\varphi}_{b,p}(x_j^p)U(x_j^p,x_{j+1}^p)\varphi_{b,p}(x_{j+1}^p)\Big]$$

with $\bar{m}=m^4-4m^2+2$

- now p labels unoriented plaquettes, and we have only half the number of bosons compared to BZ
- the resulting weight function is

$$\omega[U]=\prod_p\Big[\det\Big(1-\frac{\alpha}{2}\big(U_p+U_p^\dagger\big)\Big)\Big]^{-N_b}$$

with $\alpha=2/\bar{m}$ (allowed range is $0<\alpha<1$)

Outline

Outline

Bounds on N_b: δ-function property

- in the formulation with auxiliary boson fields, N_b is integer
- however, in the weight function $\det\left[1-\frac{\alpha}{2}\left(U_p+U_p^\dagger\right)\right]^{-N_b}$ N_b can be non-integer
- we find that the weight function goes to a δ-function for $\alpha \to 1$ if

$$N_b \geq N_c - \frac{1}{2} \qquad \text{for } \mathrm{U}(N_c)$$
$$N_b \geq N_c - \frac{5}{4} \qquad \text{for } \mathrm{SU}(N_c)$$

  - proof by character expansion: recall

$$\omega(U) = \sum_r c_r \chi_r(U) \quad \text{with} \quad c_r = \int dU\, \omega(U) \chi_r(U^{-1})$$

  - δ-function is obtained if $c_r = d_r$ for all r
  - to compute the integrals for the expansion coefficients, we use an exponential parameterization $U = e^{i\sqrt{\gamma}H}$ with $\gamma = 2(1-\alpha)/\alpha$
  - details are lengthy (essentially, one needs to compute the LO terms in γ)
  - we find $c_r \neq d_r$ if N_b is below the bound

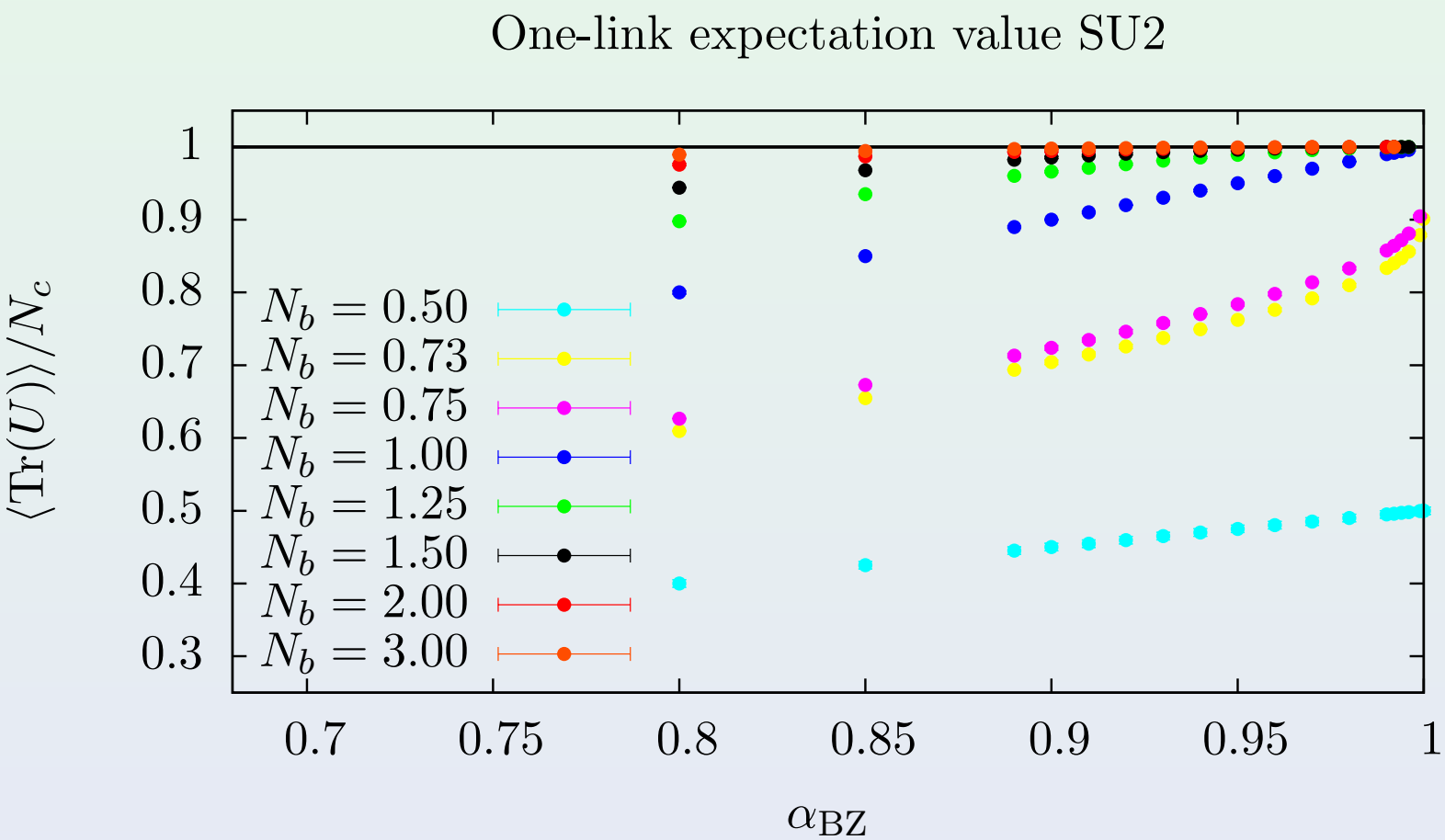
Numerical checks

- for historical reasons, the following is for the BZ action (for which the refined bounds are the same)
- to check the δ-function property, consider an expectation value

$$\langle f \rangle = \frac{1}{Z} \int dU \, \det|1 - \alpha_{\mathrm{BZ}} U|^{-2N_b} f(U)$$

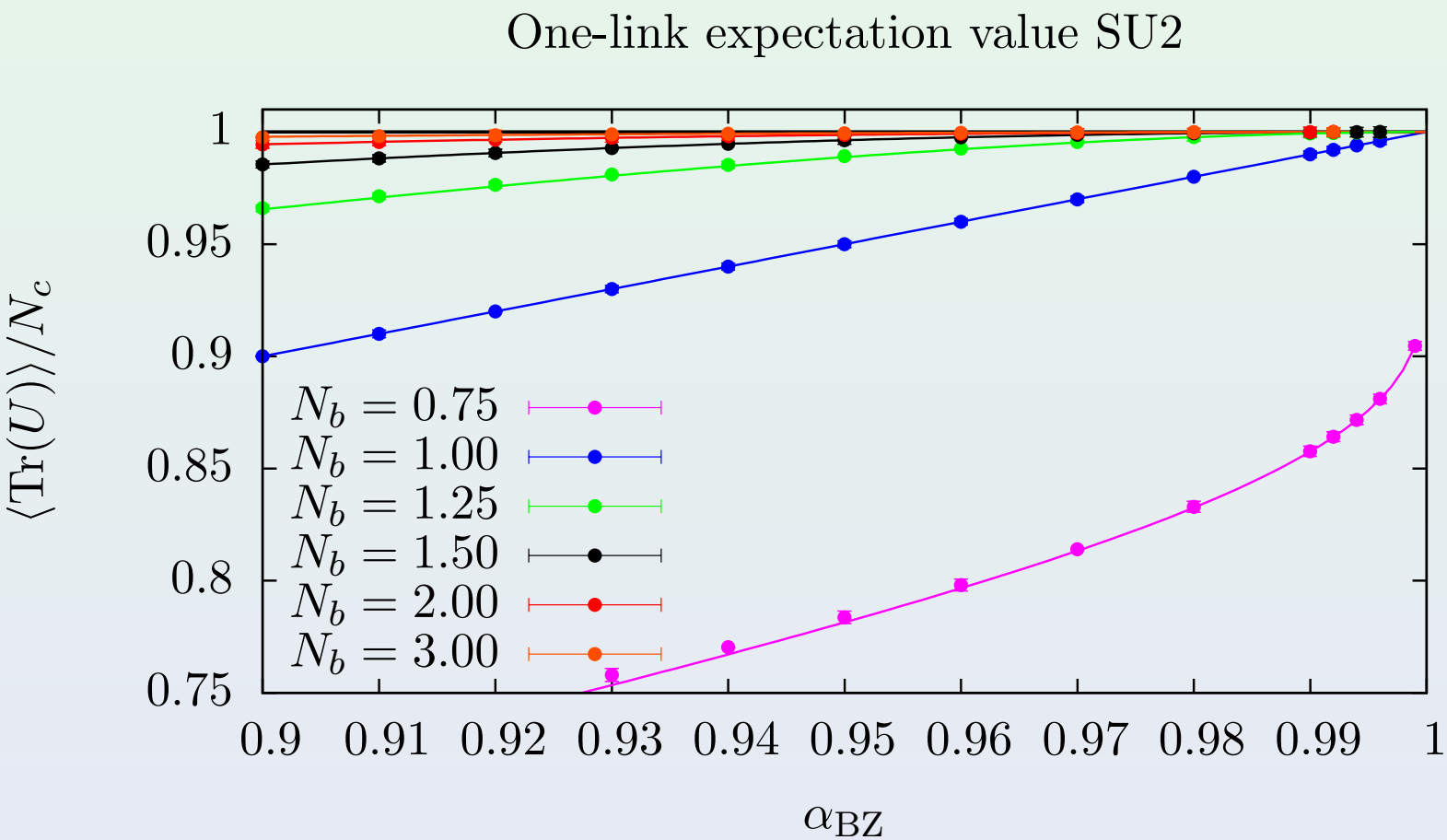
  for some test function $f(U)$
- for $\alpha_{\mathrm{BZ}} \to 1$ we should have $\langle f \rangle \to f(\mathbb{1})$

One-link expectation value for SU(2)



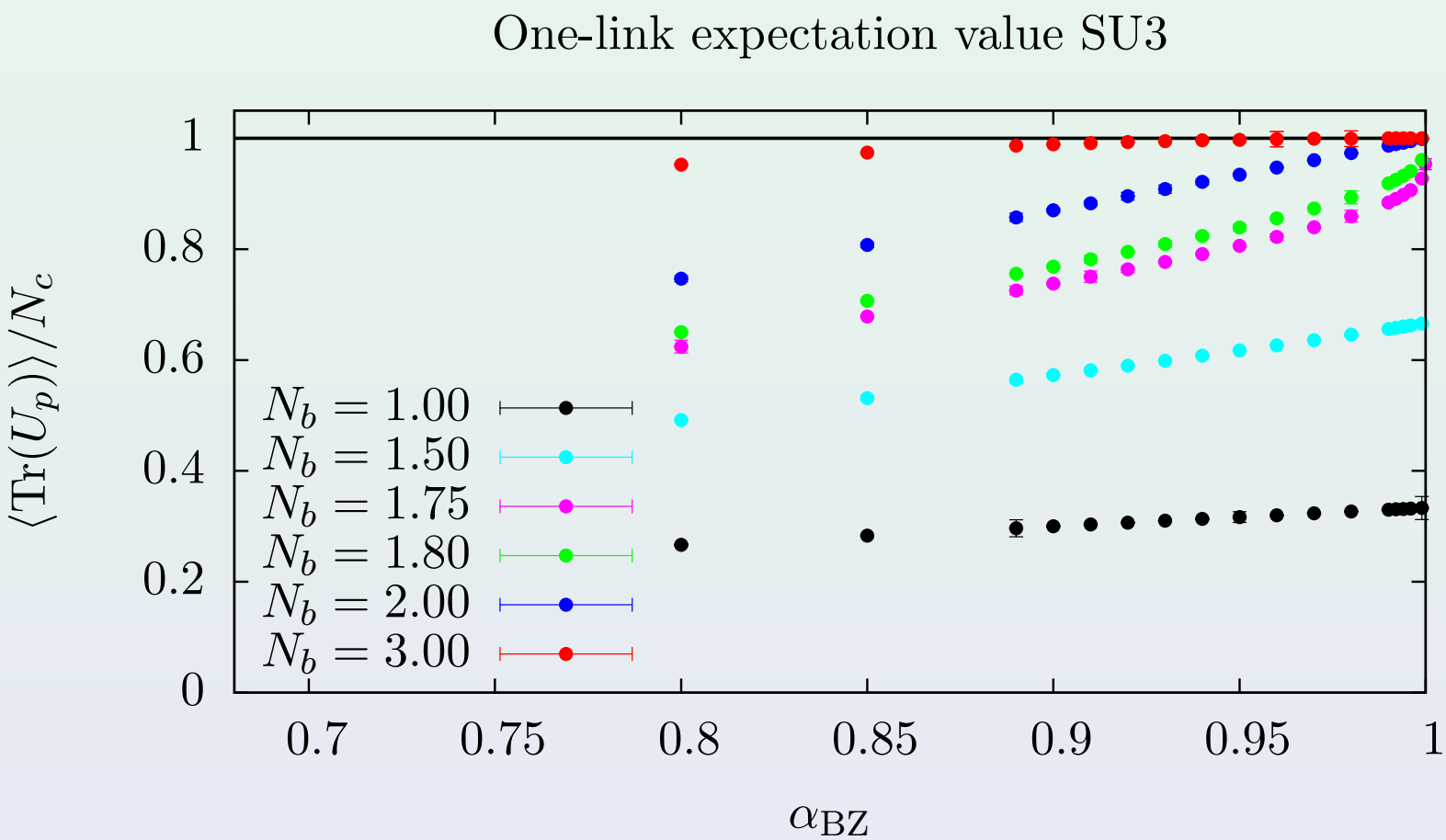
theoretical bound on N_b is $N_c - \frac{5}{4} = \frac{3}{4}$

Zoom in on $\alpha_{\mathrm{BZ}} \to 1$



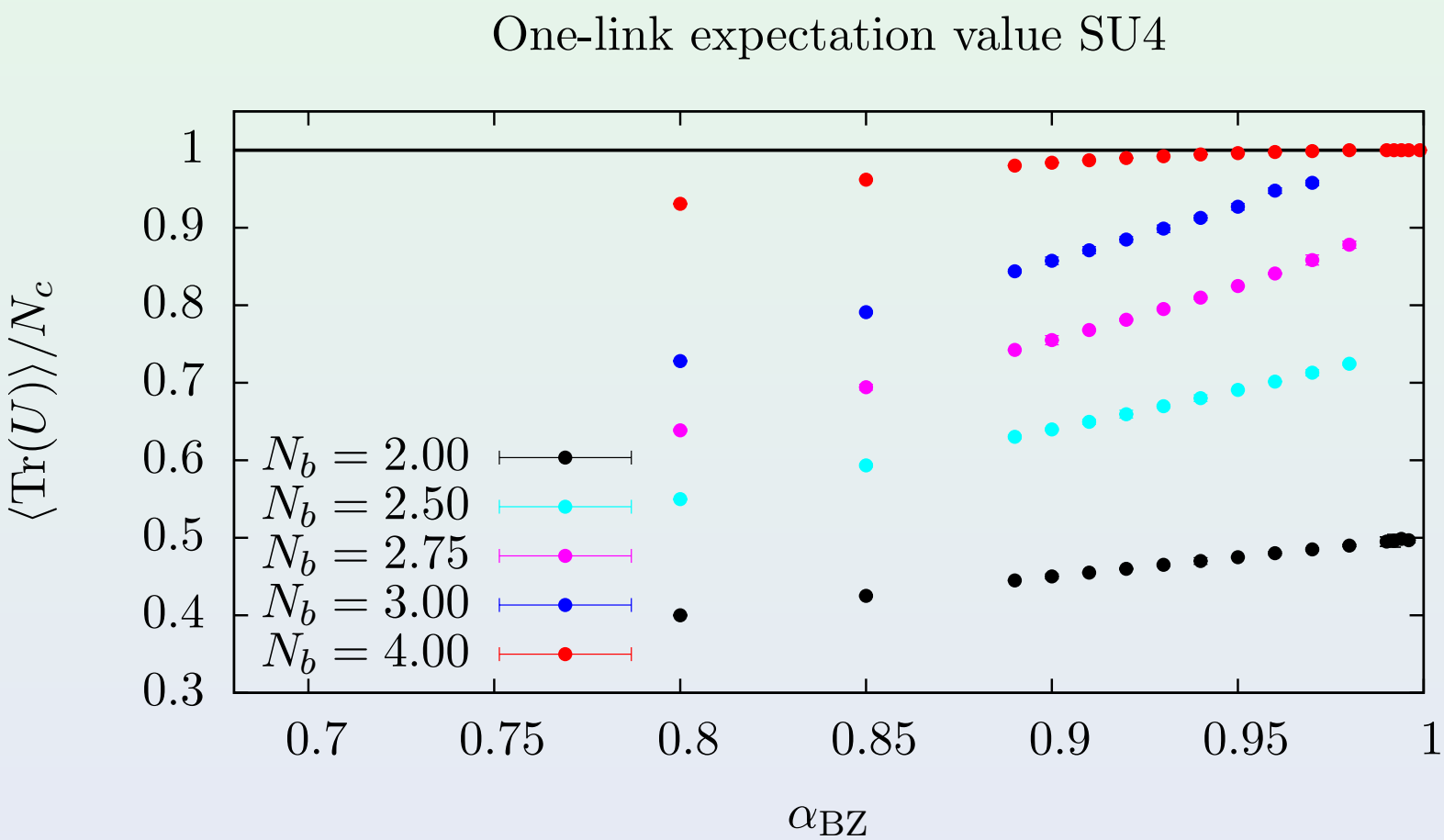
lines are analytical results for small $1 - \alpha_{\mathrm{BZ}}$ from character expansion

One-link expectation value for SU(3)



theoretical bound on N_b is $N_c - \frac{5}{4} = \frac{7}{4} = 1.75$

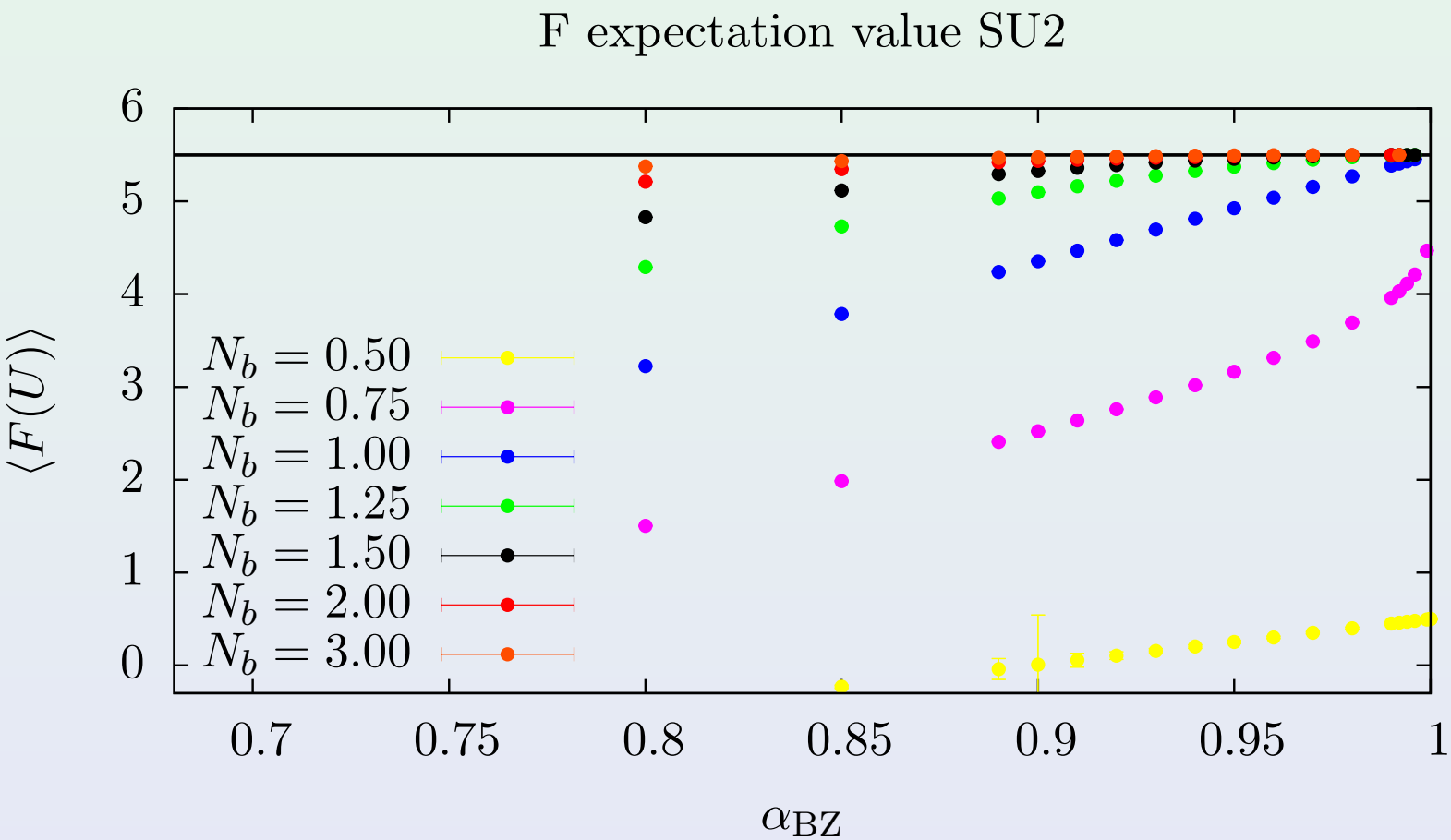
One-link expectation value for SU(4)



theoretical bound on N_b is $N_c - \frac{5}{4} = \frac{11}{4} = 2.75$

Other test functions

- e.g., $F(U) = \mathrm{tr}(U + 6U^2 - 1.5U^3)$



theoretical bound on N_b is $N_c - \frac{5}{4} = \frac{3}{4}$

Outline

Bounds on N_b: Nature of continuum limit

- again use character expansion of the weight function
- now need to compute NLO terms in $\gamma \to 0$ expansion of c_r
- we get the correct boundary-value partition function $\Gamma(\mathcal{U})$ if

$$N_b \geq N_c + \frac{1}{2} \qquad \text{for } \mathrm{U}(N_c)$$

$$N_b \geq N_c - \frac{3}{4} \qquad \text{for } \mathrm{SU}(N_c)$$

  i.e., with these bounds the continuum limit of the induced theory is equivalent to YM (in $d = 2$)
- for $d > 2$ numerical evidence suggests that these bounds can be relaxed (but cannot be weaker than those for the δ-function)

Outline

Perturbation theory

- goal: connect parameters of standard theory (β) and induced theory (α)
  - must be done numerically (i.e., non-perturbatively), but close to the continuum limit we should get some idea from perturbation theory
- normally one would expand in powers of $(1-\alpha)$ at fixed N_b, but there are two problems with the action

$$S_{\text{eff}} = -N_b \sum_p \operatorname{tr} \log\left(1 - \frac{\alpha}{2}\left(U_p + U_p^\dagger\right)\right)$$

  - expansion about $U_p = \mathbb{1}$ for $\alpha \to 1$ has zero radius of convergence (expansion of log about 0)
  - even if we ignore this convergence issue, formal expansion does not allow for saddle-point analysis

Large-N_b perturbation theory

instead, we approach the cont. limit by large-N_b pert. theory for fixed α

- write $\log(1-\frac{\alpha}{2}(U_p+U_p^\dagger)) = \log(1-\frac{\alpha}{2(1-\alpha)}(U_p+U_p^\dagger-2)) + \text{const}$
  leading-order expansion then gives definition of coupling in induced theory:

$$\frac{1}{g_I^2} = \frac{N_b\alpha}{2(1-\alpha)} \qquad \text{(i.e., large } N_b \longleftrightarrow \text{small } g_I\text{)}$$

- an n-loop calculation yields

$$\frac{1}{g_W^2} = \frac{1}{g_I^2}\left[1 + c_1(\alpha)g_I + c_2(\alpha)g_I^2 + \ldots\right] \qquad (*)$$

- however: expansion of the log converges only for $\alpha \leq \frac{1}{3}$, thus $(*)$ does not apply directly to continuum limit $\alpha \to 1$ with N_b fixed
- if we ignore this problem, $(*)$ can be rewritten as

$$\beta = \frac{b_{-1}}{1-\alpha} + b_0 + b_1(1-\alpha) + \ldots$$

with coefficients b_n that depend on N_b (and on N_c and the dimension d)

Outline

Outline

Basic setup

- consider first the cheapest nontrivial case: SU(2) in $d = 3$
- do simulations in standard Wilson pure gauge theory (WPG) and in the induced pure gauge theory (IPG)
  - use one observable to match the couplings β and α
  - then compare other observables in both theories
- observables for our first tests:
  - $T = 0$: quantities connected with the $\bar{q}q$ potential
  - $T \neq 0$: transition temperature and order of the transition
- simulation details:
  - WPG: standard mixture of heatbath and overrelaxation updates
  - IPG: local Metropolis with links evolving in ε-ball
  - computation of $\bar{q}q$ potential: Lüscher-Weisz algorithm JHEP 0109 (2001) 010
  - scale setting: Sommer parameter r_0 NPB 411 (1994) 839

First step: matching of α and β

- matching via Sommer scale r_0:
  for each α, find $\beta(\alpha)$ such that $r_0(\alpha)_{\text{IPG}} = r_0(\beta)_{\text{WPG}}$
- then fit coefficients in

$$\beta(\alpha) = \frac{b_{-1}}{1-\alpha} + b_0 + b_1(1-\alpha) + \ldots$$

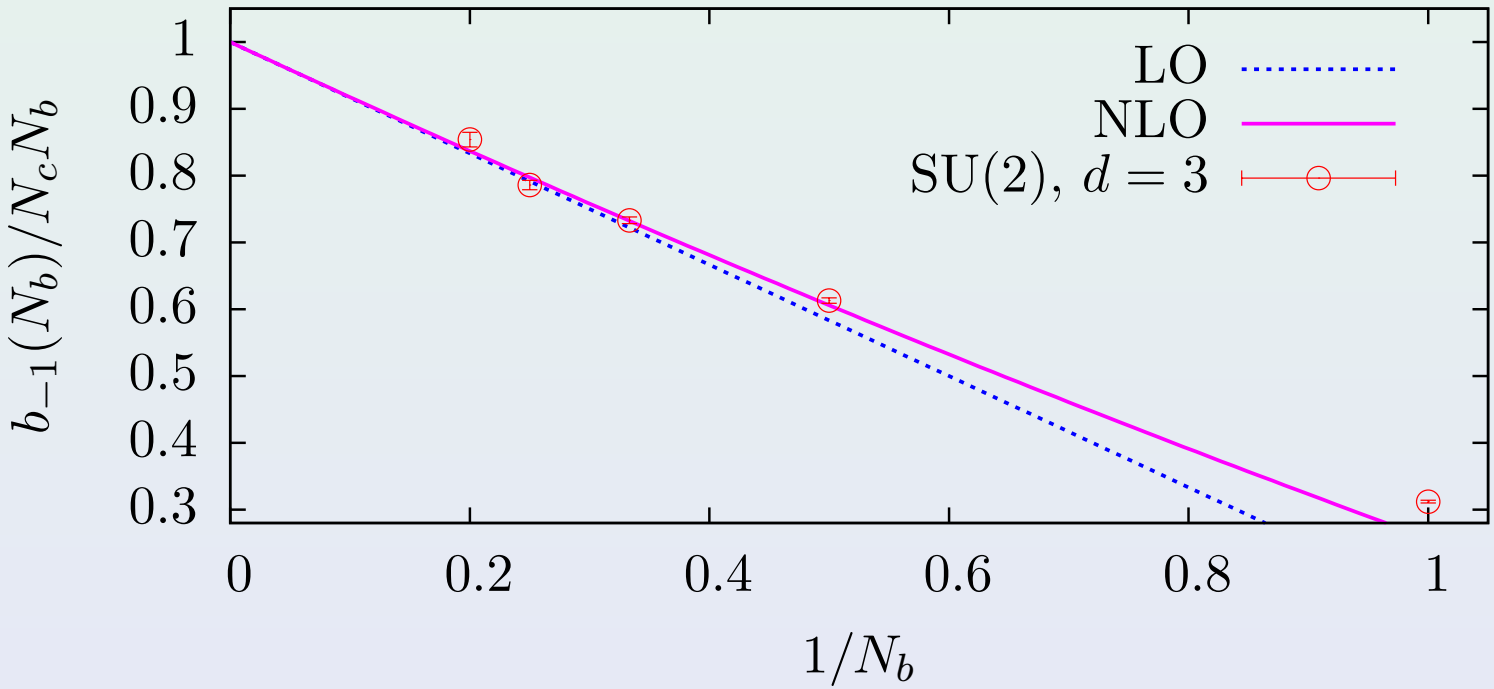
- numerical results:

N_b	b_{-1}	b_0	b_1
1	0.623(4)	−1.78(11)	3.59(69)
2	2.453(14)	−2.76(38)	0.99(5)
3	4.399(29)	−4.43(16)	−0.17(21)
4	6.286(52)	−6.01(23)	−0.52(25)
5	8.54(11)	−8.99(41)	0.45(38)

Comparison to large-N_b perturbation theory

- 2-loop result for $N_c = 2$ in $d = 3$:

$$\frac{b_{-1}(N_b)}{N_c N_b} = 1 - \frac{5}{6N_b} + \frac{0.0908283}{N_b^2} + O(N_b^{-3})$$



so $(*)$ seems to hold even outside its formal range of applicability

Outline

Second step: static potential at similar lattice spacings

- compare to high-precision results obtained with Wilson action

  Brandt, PoS EPS-HEP (2013)

- at large distances R, the energy levels of the $\bar{q}q$ bound state are well described by an effective string theory

  Nambu, PLB 80 (1979) 372, Lüscher-Symanzik-Weisz, NPB 173 (1980) 365,

  Polyakov, NPB 164 (1980) 171

- potential in effective string theory for the flux tube ($d = 3$):

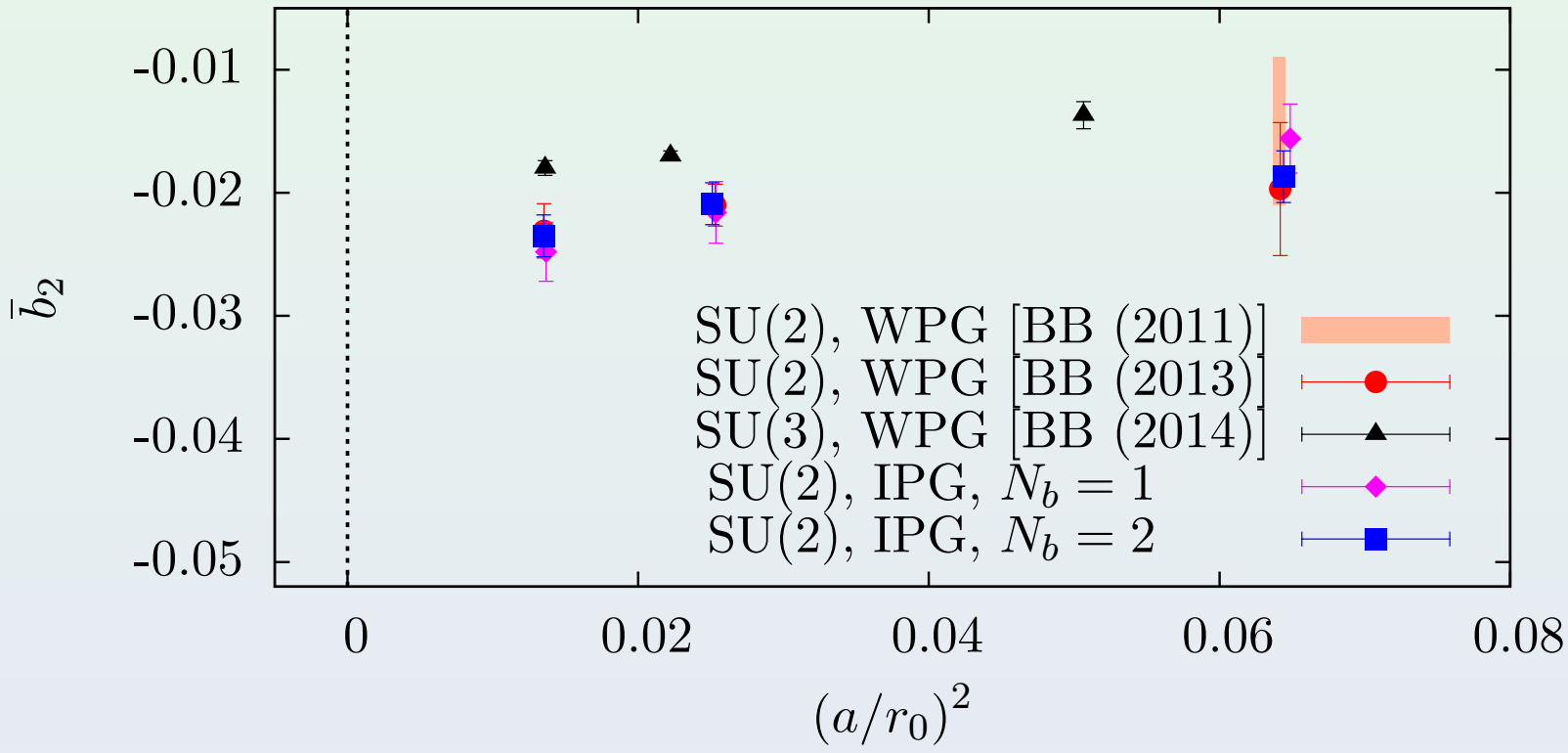
  Aharony et al. JHEP 0906 (2009), JHEP 1012 (2010), JHEP 1101 (2011), JHEP 1305 (2013)

$$V(R) = \sigma R \sqrt{1 - \frac{\pi}{12\sigma R^2}} - \bar{b}_2 \frac{\pi^3}{60\sqrt{\sigma^3} R^4}$$

  two non-universal parameters: σ and $\bar{b}_2$ (boundary coefficient)

- $\sqrt{\sigma} r_0$ is the same in both theories (by construction)
- agreement of $\bar{b}_2$ means that the potential is identical up to 4–5 significant digits

Results for $\bar{b}_2$



excellent agreement

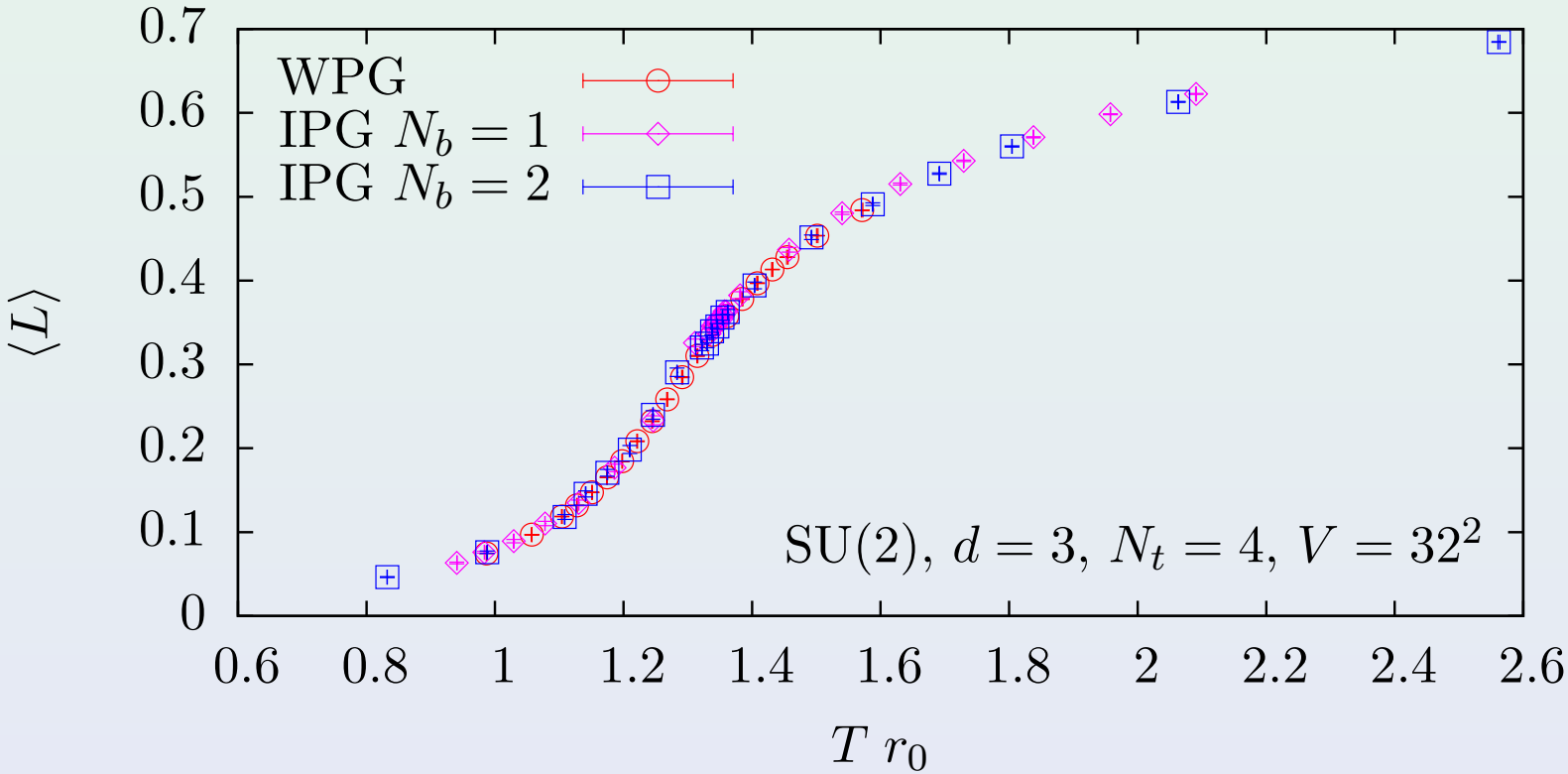
Outline

Finite-T properties

- will look at finite-T transition
- SU(2) in $d = 3$: second-order phase transition in the 2d Ising universality class Engels et al. NPPS 53 (1997)
- first tests at $N_t = 4$ (and some results at $N_t = 6$)
- scale setting via r_0 and the matching between α and β obtained at $T = 0$

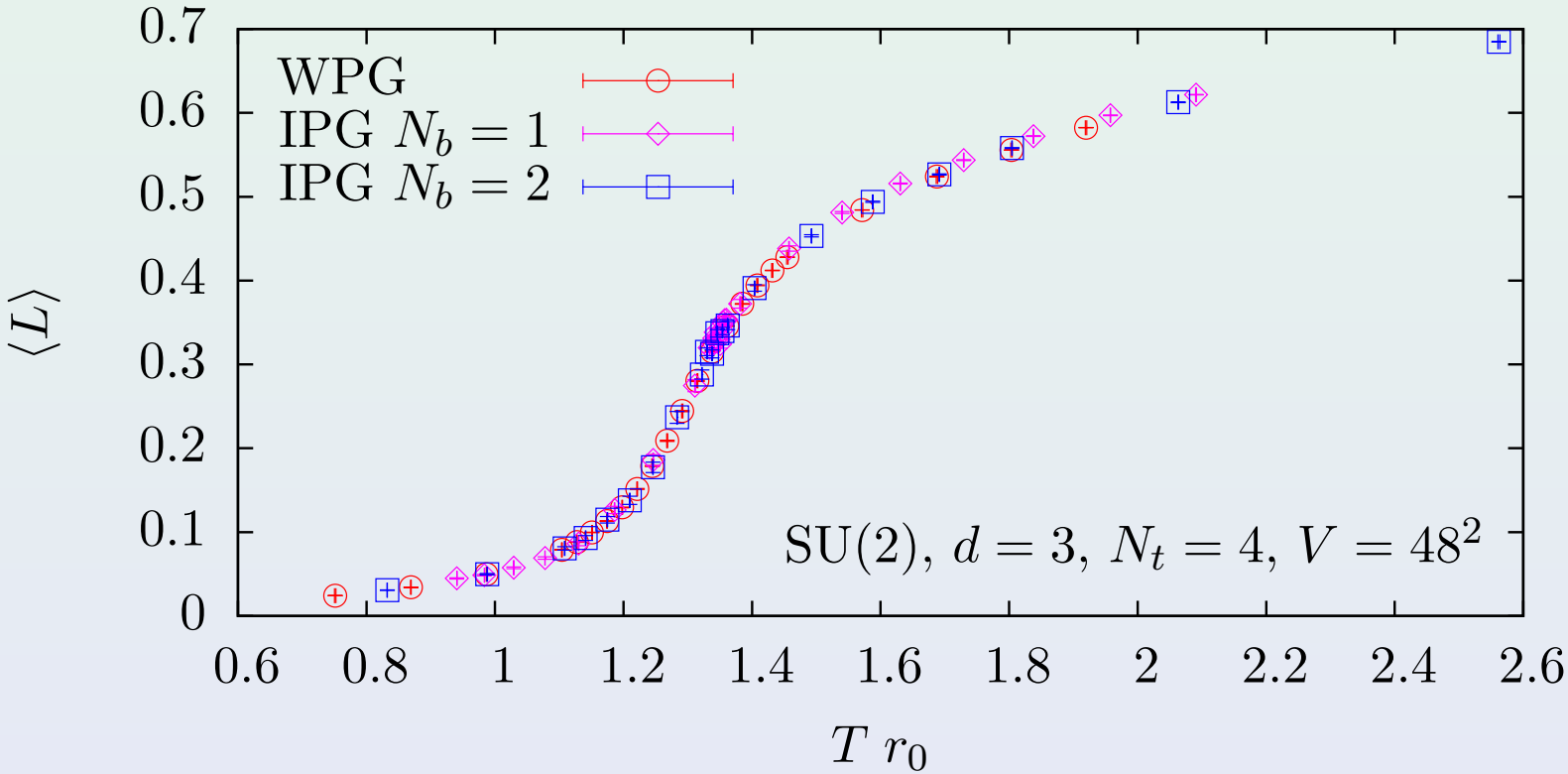
Phase transition at $N_t = 4$

Polyakov loop expectation value:



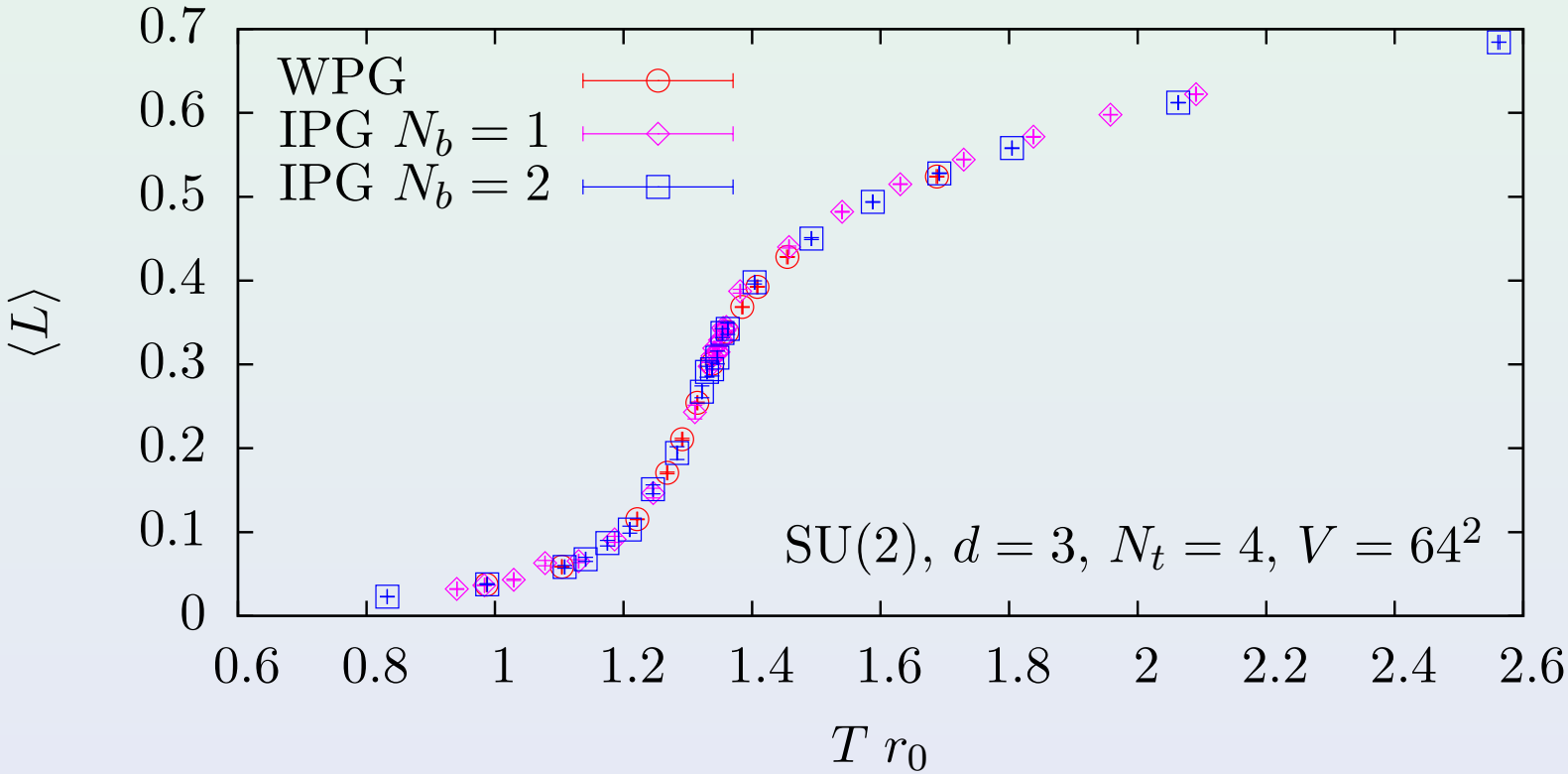
Phase transition at $N_t = 4$

Polyakov loop expectation value:



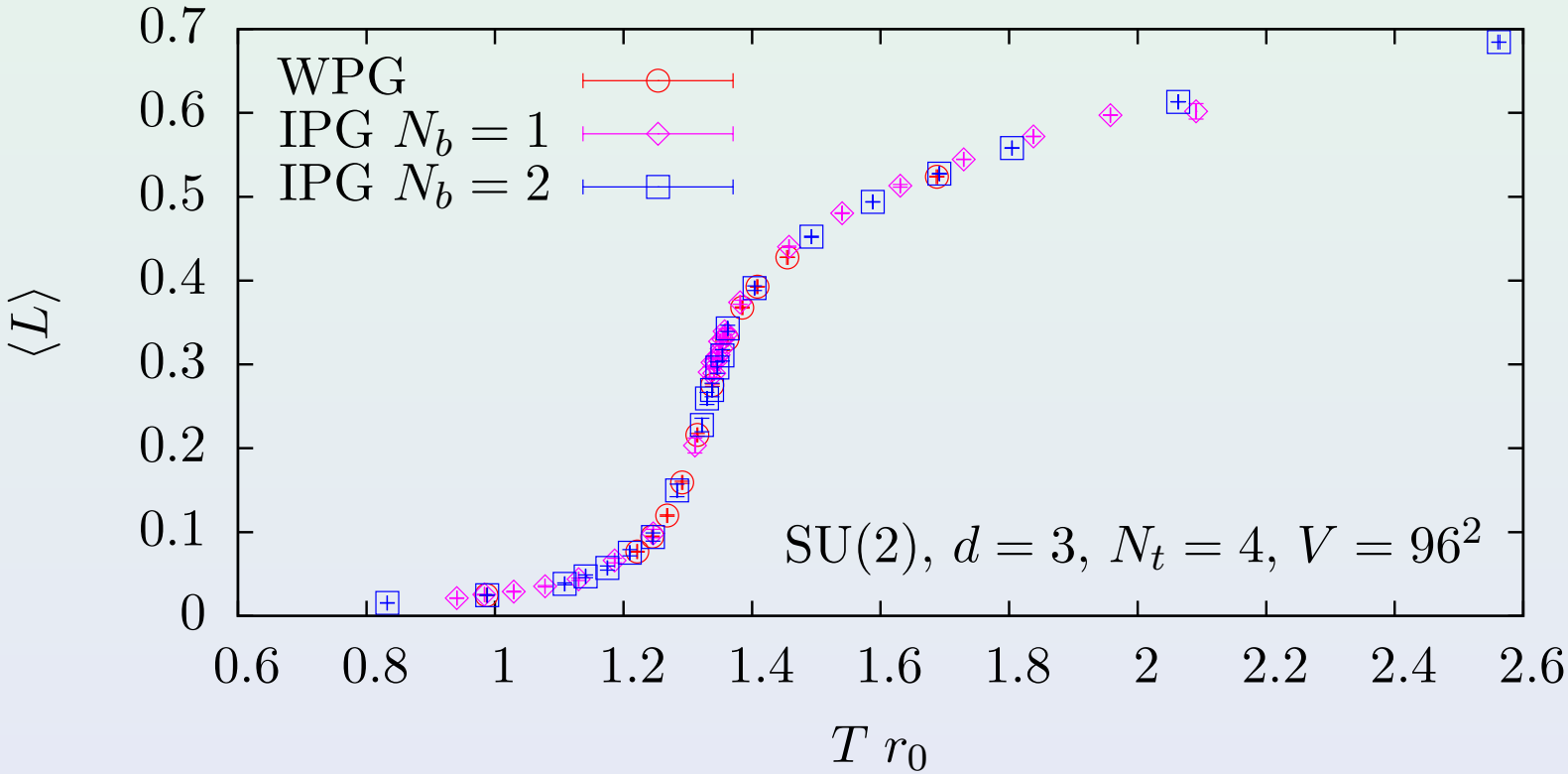
Phase transition at $N_t = 4$

Polyakov loop expectation value:



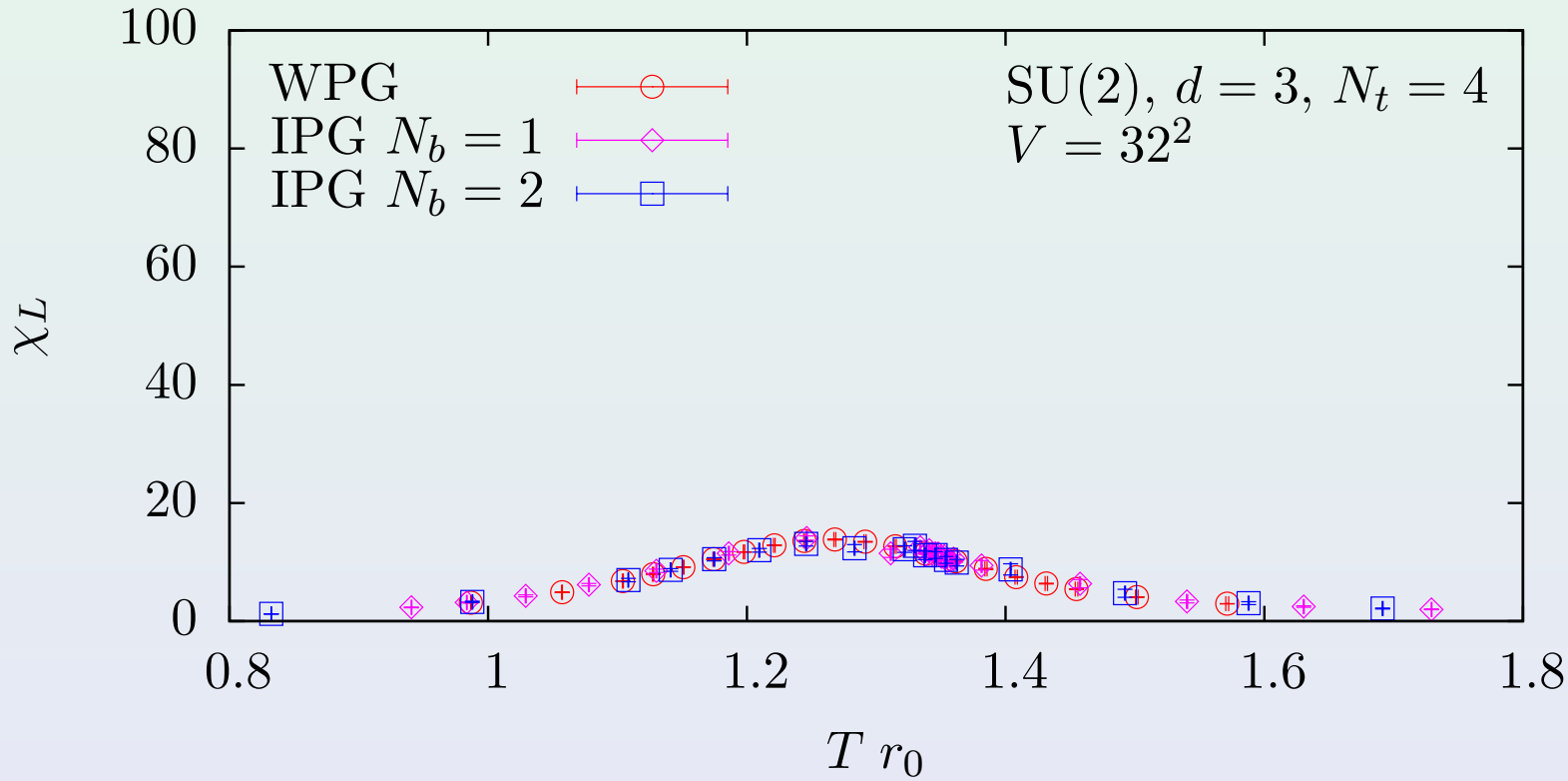
Phase transition at $N_t = 4$

Polyakov loop expectation value:



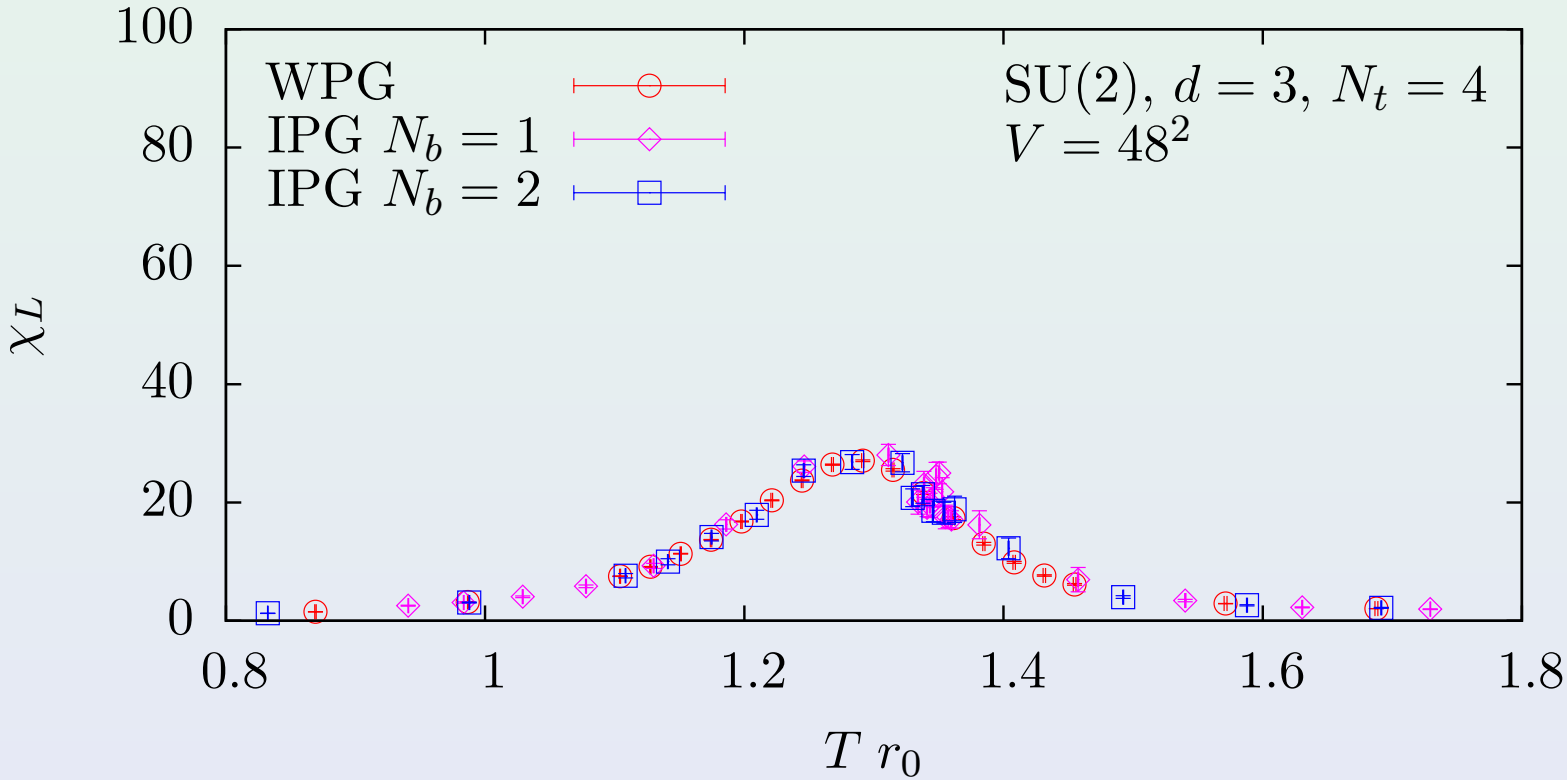
Phase transition at $N_t = 4$

Polyakov loop susceptibility $\chi_L \sim \langle |L|^2 \rangle - \langle |L| \rangle^2$:



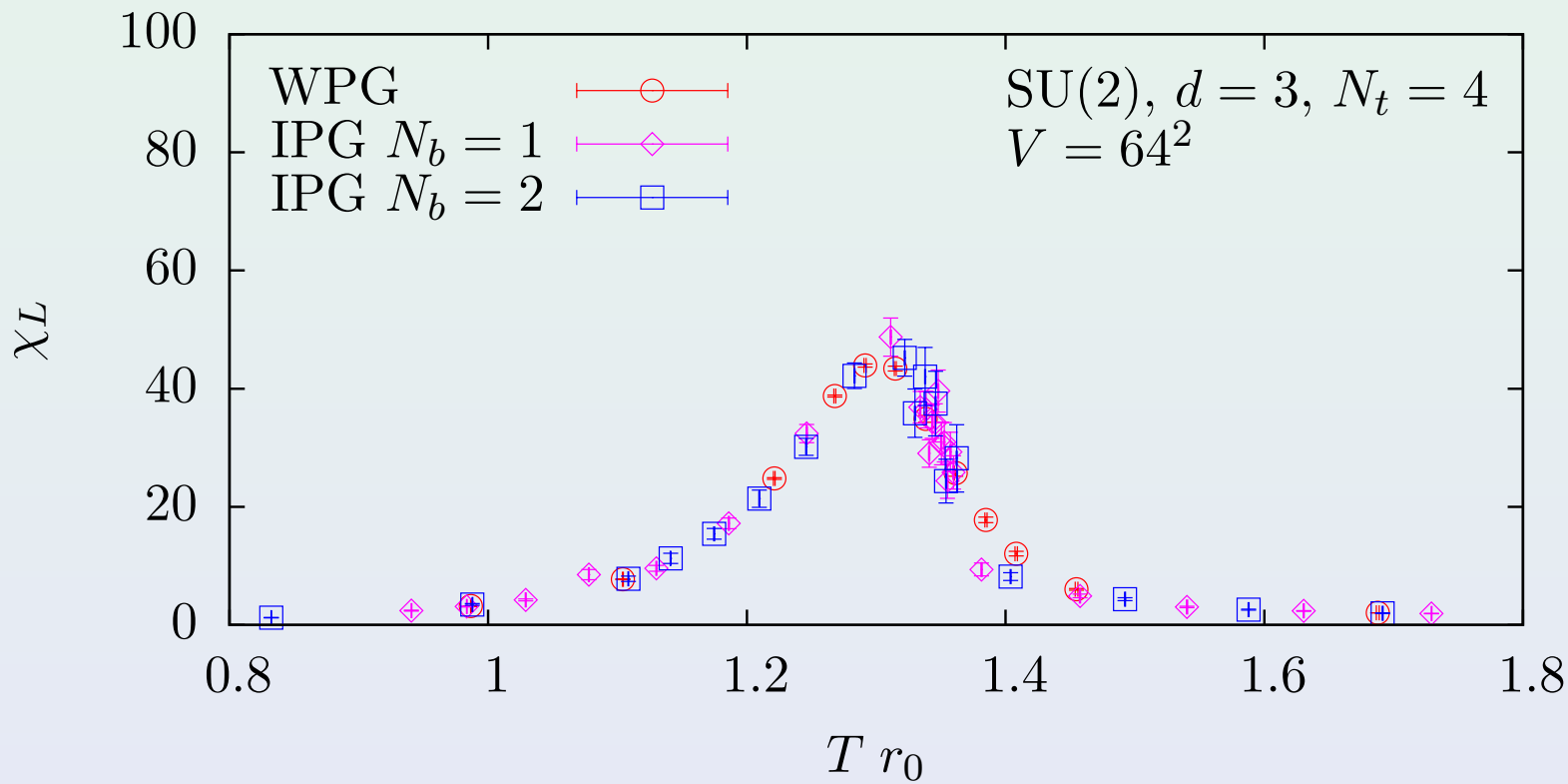
Phase transition at $N_t = 4$

Polyakov loop susceptibility:



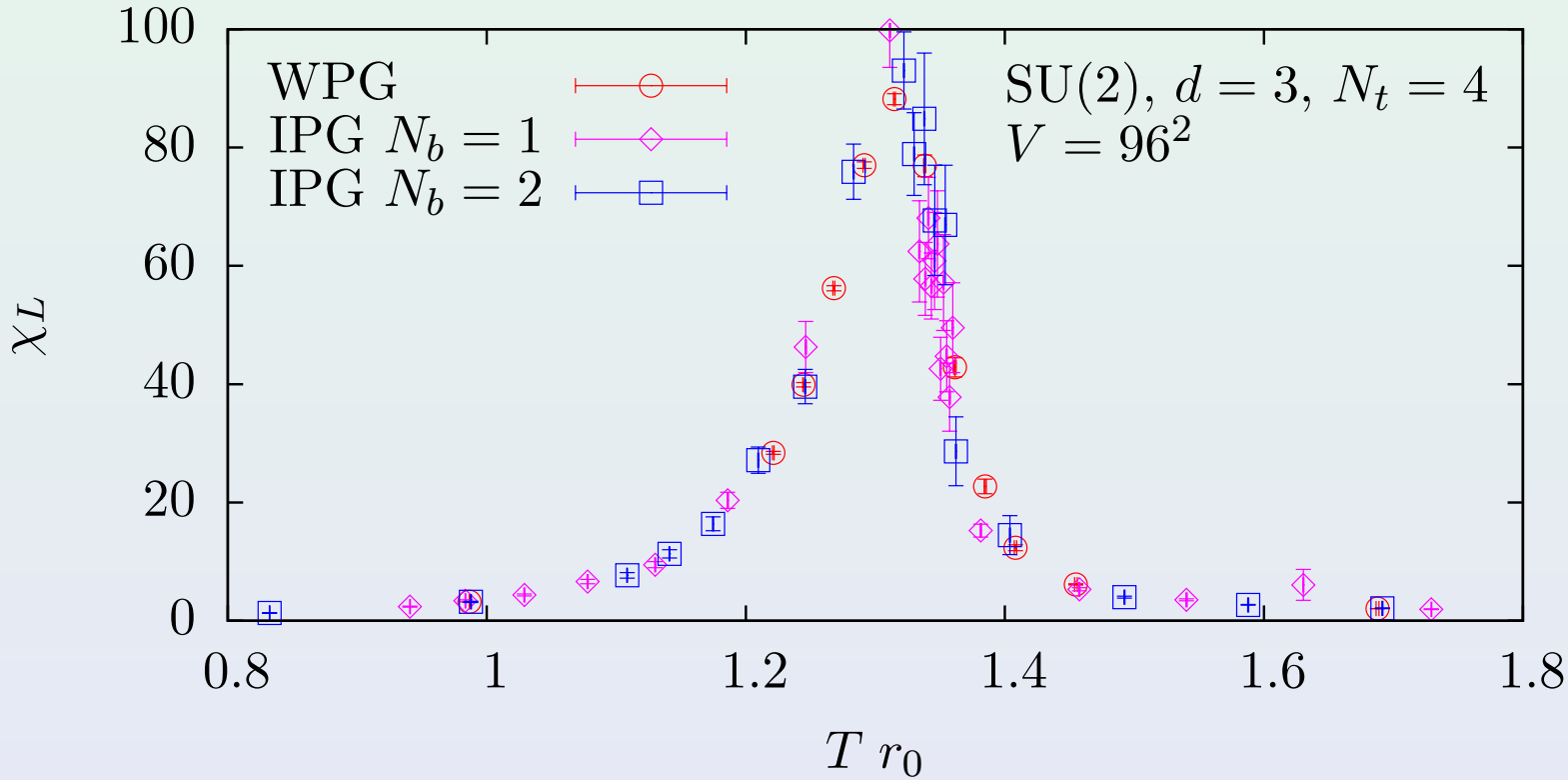
Phase transition at $N_t = 4$

Polyakov loop susceptibility:



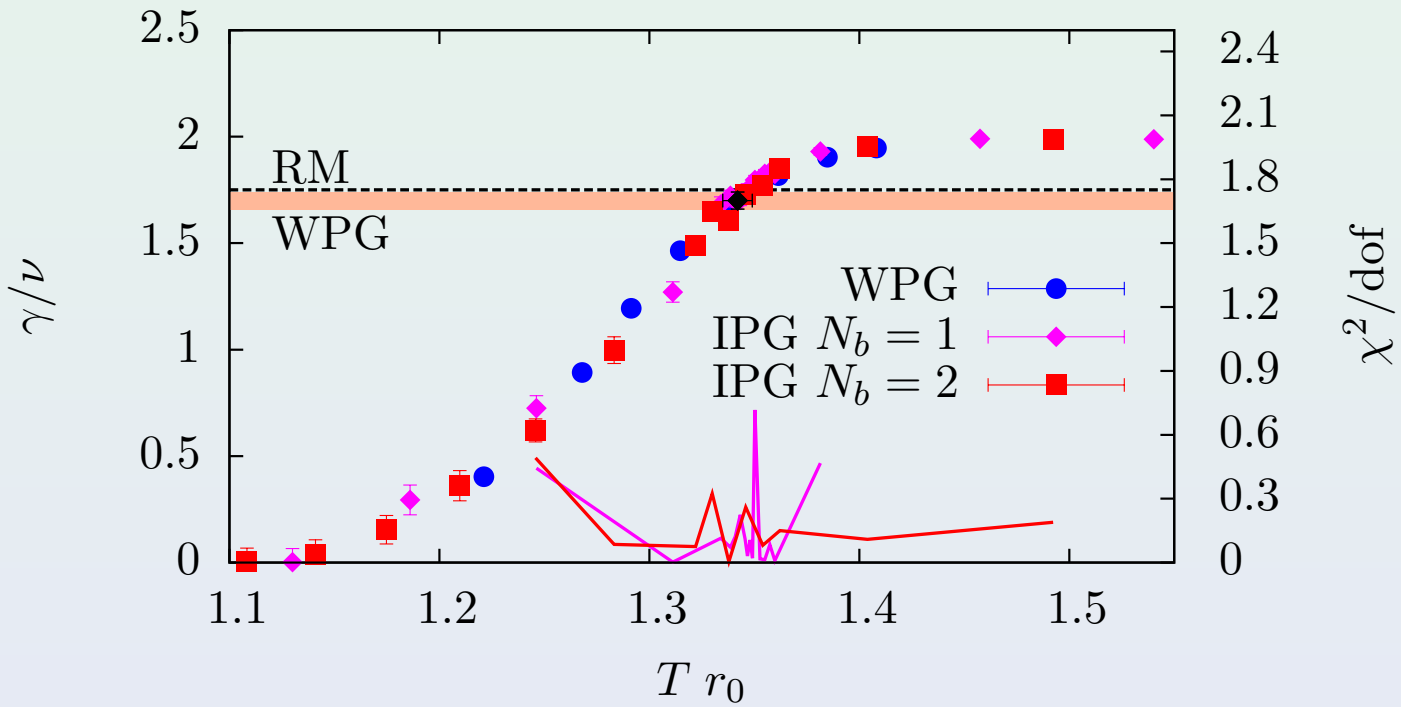
Phase transition at $N_t = 4$

Polyakov loop susceptibility:



Critical exponents at $N_t = 4$

fit: $\ln \chi_L = C + \frac{\gamma}{\nu} \ln N_s$ (valid only at phase transition, deviations otherwise)
phase transition identified by smallest χ^2/dof



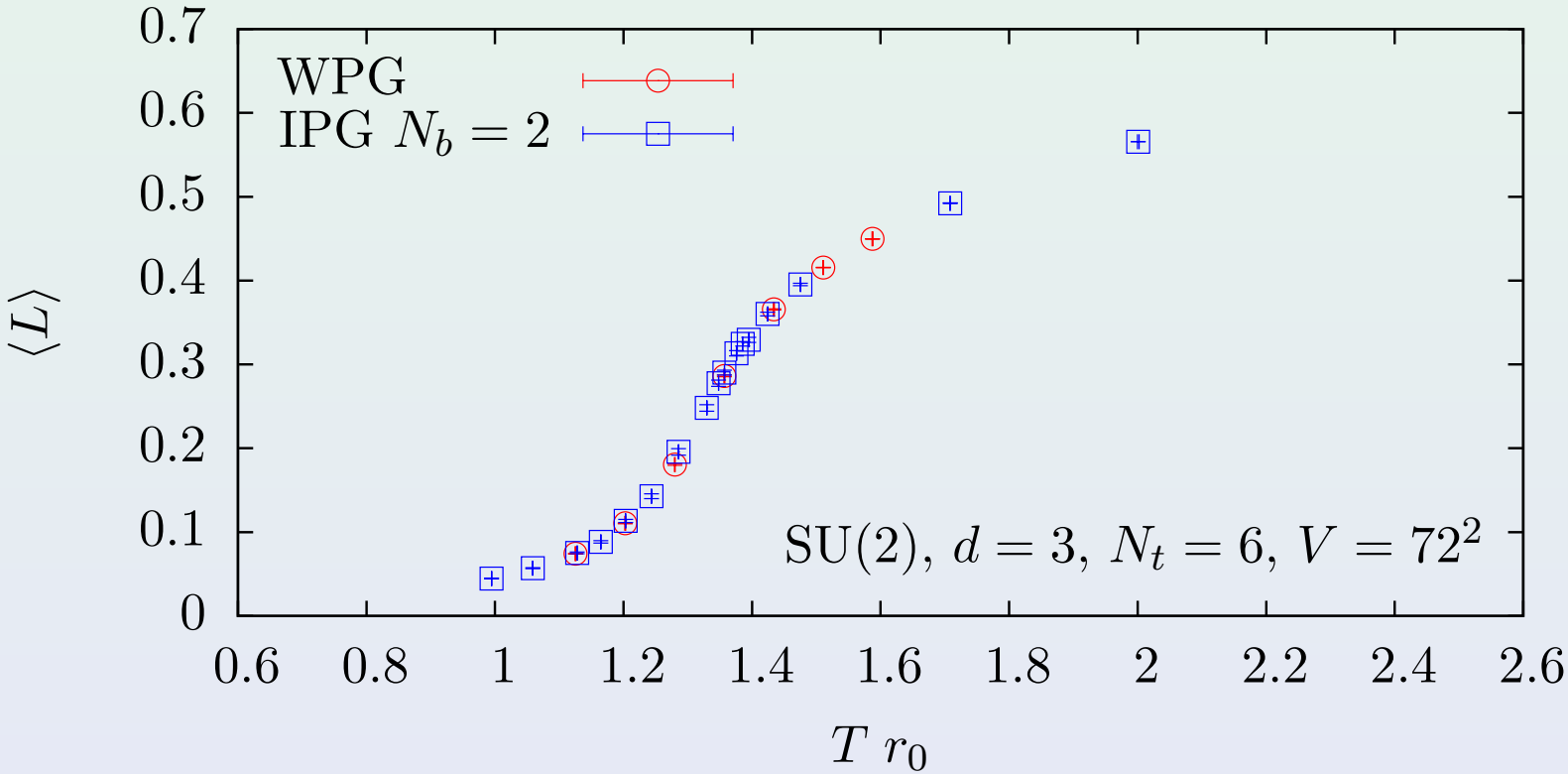
black point (WPG): $\gamma/\nu = 1.70(4)$

Engels et al. NPPS 53 (1997)

for IPG we obtain $\gamma/\nu = 1.69(4)$ at $T_c r_0 = 1.34(2)$

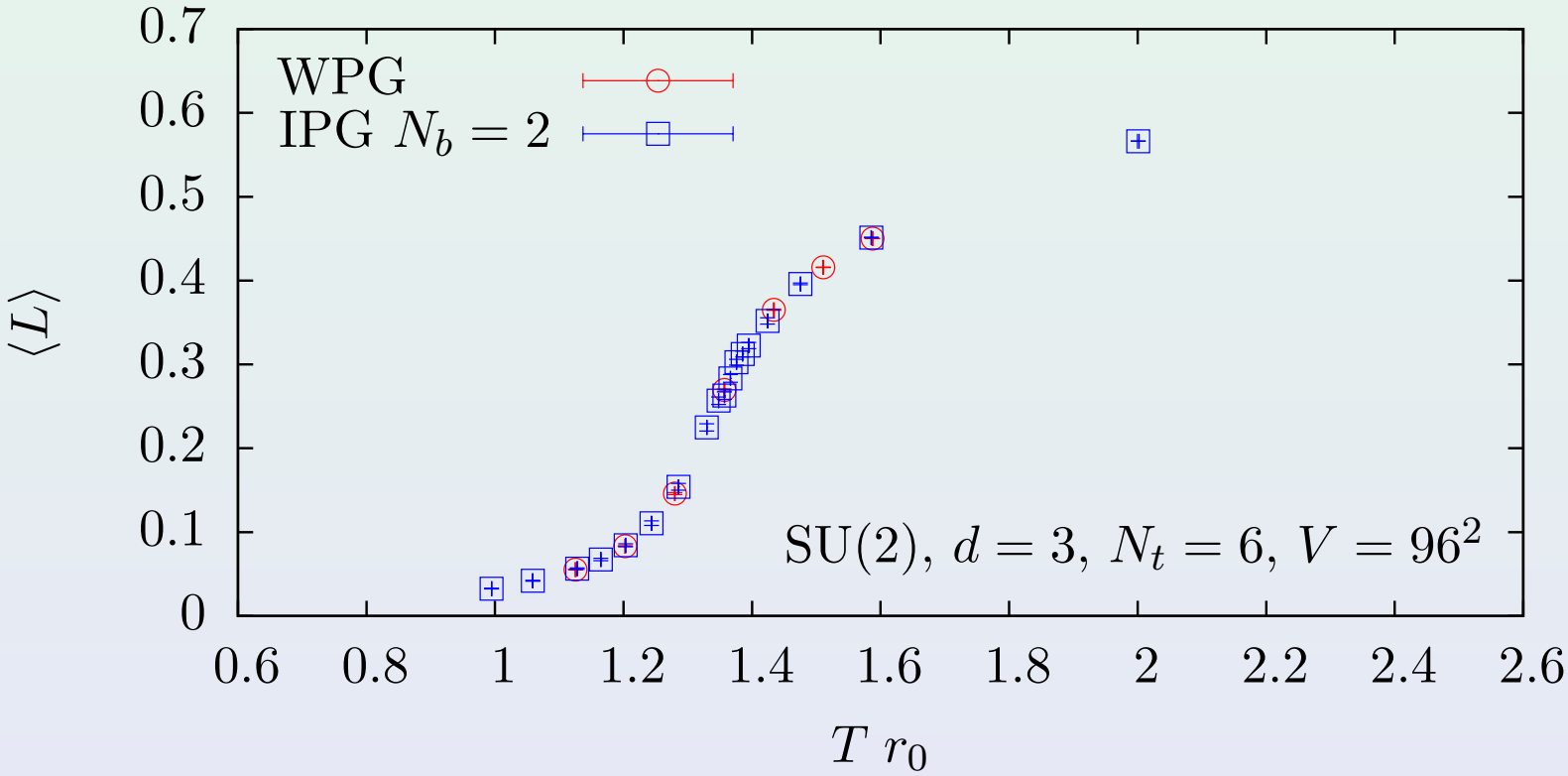
Phase transition at $N_t = 6$

Polyakov loop expectation value:



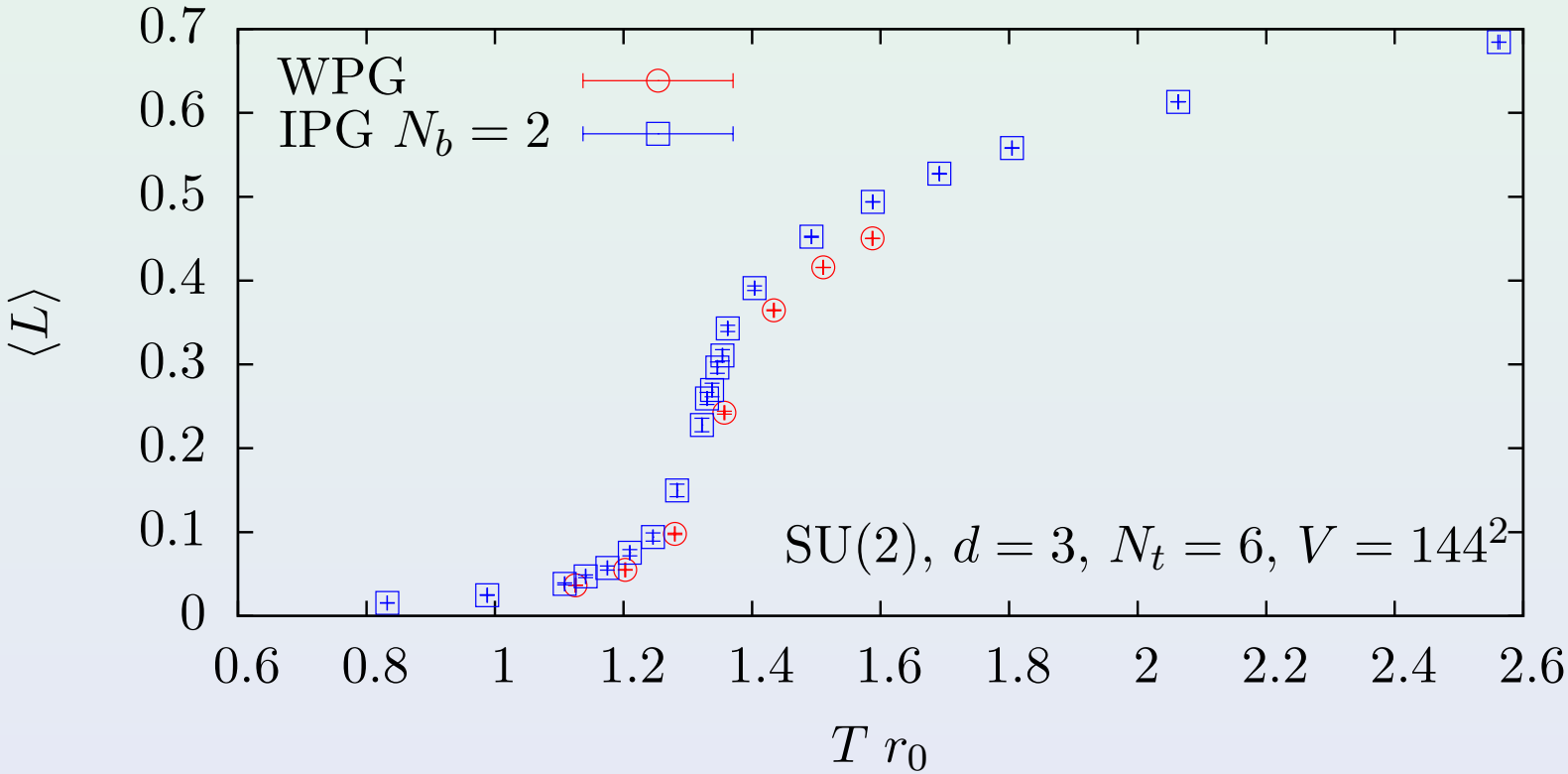
Phase transition at $N_t = 6$

Polyakov loop expectation value:



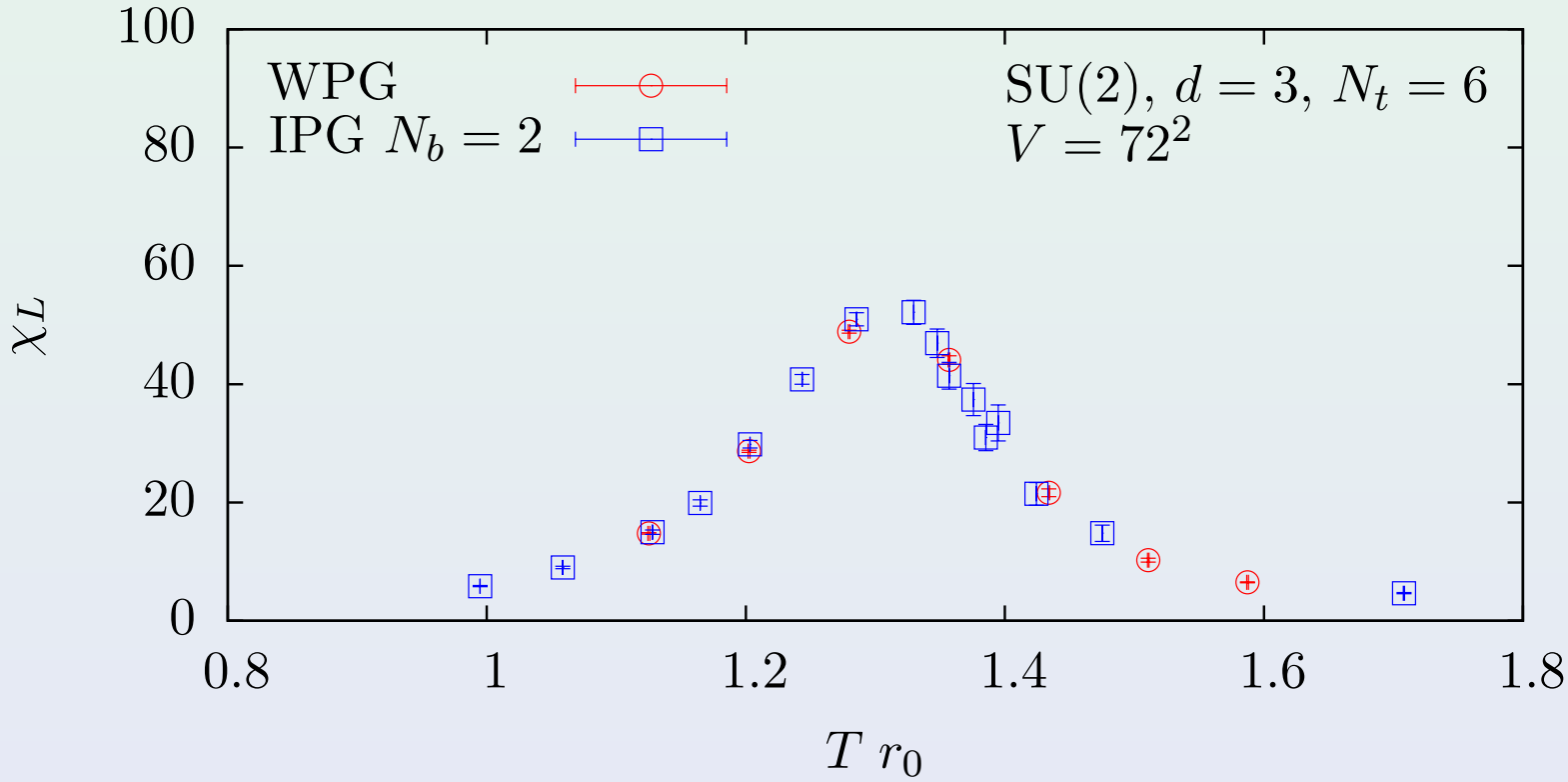
Phase transition at $N_t = 6$

Polyakov loop expectation value:



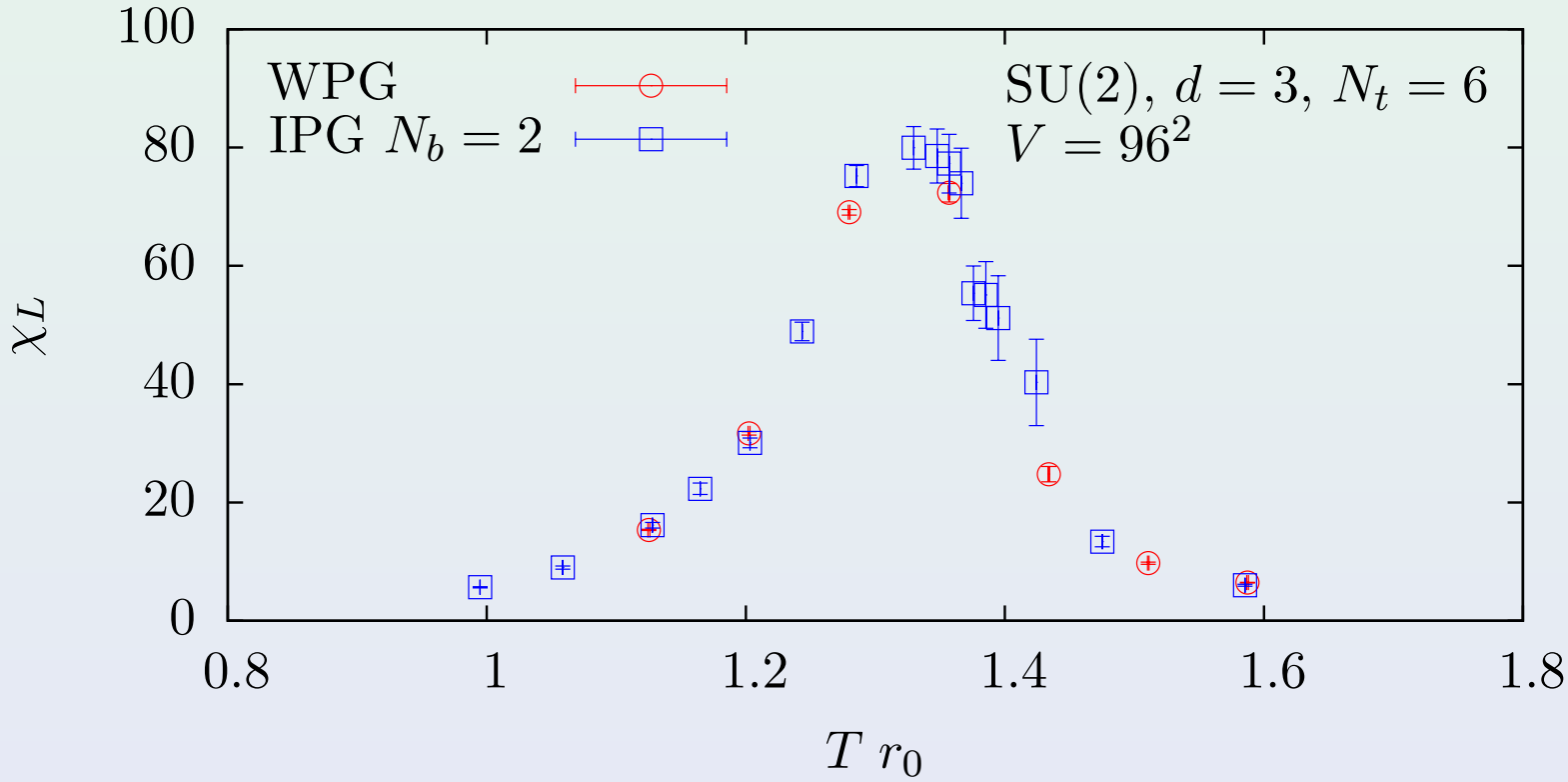
Phase transition at $N_t = 6$

Polyakov loop susceptibility:



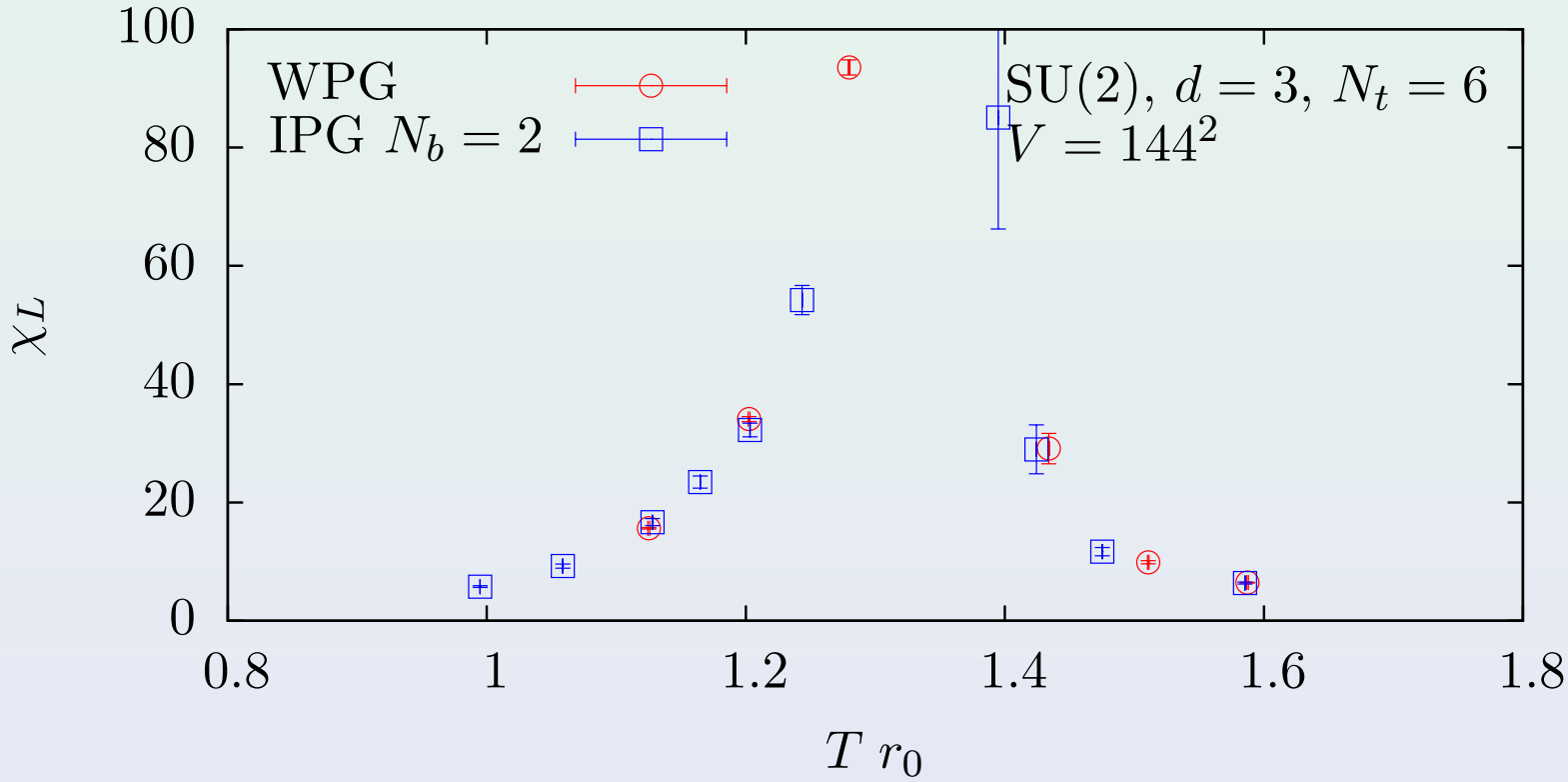
Phase transition at $N_t = 6$

Polyakov loop susceptibility:



Phase transition at $N_t = 6$

Polyakov loop susceptibility:



Outline

Dual representation

- we have seen, by a combination of analytical and numerical arguments, that the induced pure gauge theory has the same continuum limit as the Wilson pure gauge theory
- now: instead of integrating out the auxiliary bosons, integrate out the gauge fields to arrive at a theory involving only auxiliary bosons (and later also fermions)
- goal: construction of new simulation algorithms, possibly solving some sign problems
- start from

$$S_B[\varphi,\bar{\varphi},U] = \sum_{b=1}^{N_b}\sum_{p}\sum_{j=1}^{4}\Big[m\bar{\varphi}_{b,p}(x_j^p)\varphi_{b,p}(x_j^p) - \bar{\varphi}_{b,p}(x_{j+1}^p)U(x_{j+1}^p,x_j^p)\varphi_{b,p}(x_j^p) - \bar{\varphi}_{b,p}(x_j^p)U(x_j^p,x_{j+1}^p)\varphi_{b,p}(x_{j+1}^p)\Big]$$

Integration over gauge fields

- write partition function as a product of integrals

$$Z = \int D\varphi D\bar{\varphi}\, \mathcal{F}[\varphi,\bar{\varphi}] \prod_{x,\mu} \int dU_\mu(x)\, e^{\frac{1}{2}\,\mathrm{tr}\left\{U_\mu(x)A_\mu(x)[\varphi,\bar{\varphi}]+U_\mu^\dagger(x)A_\mu^\dagger(x)[\varphi,\bar{\varphi}]\right\}}$$

$$= \int D\varphi D\bar{\varphi}\, \mathcal{F}[\varphi,\bar{\varphi}] \prod_{x,\mu} \mathcal{I}_{x,\mu}[\varphi,\bar{\varphi}]$$

with

$$\mathcal{F}[\varphi,\bar{\varphi}] = \exp\left\{-\sum_{b=1}^{N_b}\sum_{p}\sum_{j=1}^{4} \bar{m}\bar{\varphi}_{b,p}(x_j)\varphi_{b,p}(x_j)\right\}$$

$$A_\mu(x)[\varphi,\bar{\varphi}] = 2\sum_{b=1}^{N_b}\sum_{\mu\neq\nu}\Big[\varphi_{b,\bar{p}(x,\mu,\nu)}(x_{\bar{j}(\mu,\nu,0,1)})\bar{\varphi}_{b,\bar{p}(x,\mu,\nu)}(x_{\bar{j}(\mu,\nu,0,0)})$$

$$+\,\varphi_{b,\bar{p}(x-\hat{\nu},\mu,\nu)}(x_{\bar{j}(\mu,\nu,1,1)})\bar{\varphi}_{b,\bar{p}(x-\hat{\nu},\mu,\nu)}(x_{\bar{j}(\mu,\nu,1,0)})\Big]$$

Integration over gauge fields

- need to solve integral $\mathcal{I} = \int dU\, e^{\mathrm{tr}(UA+U^\dagger A^\dagger)}$
- some results known in literature for $\mathrm{U}(N_c)$ and $\mathrm{SU}(N_c)$ (sometimes restricted to special cases like small N_c or $A \propto \mathbb{1}$)

  e.g., Brower-Rossi-Tan, PRD 23 (1981), Lenaghan-Wilke, Nucl.Phys. B 624 (2002)
- result for $\mathrm{SU}(N_c)$:

$$\mathcal{I} \sim \sum_{\nu=0}^{\infty} \varepsilon_\nu \cos\left(\nu\theta\right) \frac{\det\left[\lambda_i^{j-1} I_{\nu+j-1}\left(\lambda_i\right)\right]}{\Delta\left(\lambda^2\right)}$$

  - ε_ν: Neumann's factor ($\varepsilon_{\nu=0} = 1$ and $\varepsilon_{\nu>0} = 2$)
  - $\exp(i\theta)$: complex phase of $\det(A)$
  - λ_i^2: eigenvalues of the $N_c \times N_c$ matrix $AA^\dagger$
  - $\Delta(\lambda^2)$: Vandermonde determinant
  - I: Bessel function of the first kind
- looks difficult, but sum over ν converges very rapidly

Full QCD

now add fermionic fields, e.g., with a staggered-type action:

$$S_f = \sum_x m_q \bar{\psi}(x)\psi(x) + \sum_{x,\mu} \left[\bar{\psi}(x)\alpha_\mu(x)U_\mu(x)\psi(x+\hat{\mu}) + \bar{\psi}(x+\hat{\mu})\tilde{\alpha}_\mu(x)U_\mu^\dagger(x)\psi(x) \right]$$

and perform the following steps (first two as in Karsch-Mütter, NPB 313 (1989)):

- expand weight factor in the Grassmann variables
- integrate out Grassmann variables
  $\longrightarrow$ constraints on the "dual variables" (occupation numbers)
- integrate out gauge fields
  - possible since they still appear linearly in the exponent
  - pre-exponential factors obtained as derivatives w.r.t. components of A

Full QCD

- result:

$$Z = \sum_{\{n,k,\ell_b,\ell_q\}} \prod_x \omega_x \prod_b \omega_b \prod_{\ell_b} \omega_{\ell_b} \int D\bar{\varphi} D\varphi \prod_{\ell_q} \omega_{\ell_q}[\varphi,\bar{\varphi}] \mathcal{F}[\varphi,\bar{\varphi}] \prod_b \mathcal{I}_b[\varphi,\bar{\varphi}]$$

  - monomer terms: $\omega_x = \frac{N_c!}{n_x!}(2am_q)^{n_x}$ with $n_x \in \{0,\ldots,N_c\}$
  - dimer terms: $\omega_b = \frac{(N_c-k_b)!}{N_c!k_b!}$ with $k_b \in \{0,\ldots,N_c\}$
  - baryon loops ℓ_b: ω_{ℓ_b} depends on loop geometry
  - quark loops ℓ_q: $\omega_{\ell_q}[\bar{\varphi},\varphi]$ depends on loop geometry **NEW**
- ω_{ℓ_b} and ω_{ℓ_q} are not positive definite $\rightarrow$ sign problem (to be solved)
- future work: find a smarter way to do this (Howe duality à la BZ?)
  - currently we would generate many configurations whose contribution to gauge-invariant quantities is zero after averaging over gauge fields
  - should generate only configurations with nonzero contributions after averaging

Outline

Summary and outlook

- promising (and economical) new approach to induced QCD
- sign problem in original BZ proposal can be eliminated easily
- correct continuum limit (YM) can be shown analytically in $d = 2$
- bounds on N_b vs N_c refined, also for $\mathrm{SU}(N_c)$
- perturbation theory to connect parameters of WPG and IPG
- numerical tests for $\mathrm{SU}(2)$ in $d = 3$ show good agreement with Wilson gauge action for both $T = 0$ and $T \neq 0$
- same conclusions from preliminary results for $\mathrm{SU}(3)$ in $d = 4$
- dual representation via integration over gauge fields
  - fermions can also be integrated out
  - resulting theory contains only auxiliary boson fields
  - partition function can be written in terms of monomers, dimers, baryon loops and quark loops
  - dual representation currently has a sign problem (solvable?)
  - if solvable: new simulation algorithms possible