

A summary of DWF Inverters

Bálint Joó *Jefferson Lab*
Chulwoo Jung *BNL*
James Osborn *BU*
Andrew Pochinsky *MIT*

January 17, 2007

1 Introduction

In this document we collect information about Domain Wall Fermion inverters in use in SciDAC on BlueGene computers. The aim is to provide as much information and performance comparison as we can. We consider the level 2 inverters in Chroma, the MIT level 3 inverter, the inverter in CPS, and the inverter over QDP/C.

2 Chroma

2.1 Overview

Chroma works over a level 2 inverter routine into which we can plug various kinds of linear operators. The linear algebra is done using BLAS within QDP++, which on the BlueGene is mostly supplied by the *bagel_qdp* library. The numerically intensive parts of the various 5D operators make use of the Dslash and vector Dslash routines. On the BlueGene these are supplied by the *bagel_wilson_dslash* library. Both *bagel_qdp* and *bagel_wilson_dslash* require the use of the BAGEL assembly generator package. Chroma can also utilize the the *CG-DWF* package in inversions used in propagator calculations.

2.2 Data Layout Comments

The BAGEL Wilson Dslash has the same layout as QDP++ in 2 color checkerboarded mode. So the Fermion Field Layout agrees between BAGEL and Chroma. The gauge field needs to be packed (transposed) to meet BAGEL's needs.

2.3 Operators Supplied

The following operators are available:

- Regular Shamir Domain Wall
- Moebius Domain Wall,
- Continued Fraction Chiral Operator

- Partial Fraction Chiral Operator

2.4 Preconditioning

All the above operators are available in unpreconditioned form and in even odd preconditioned form where the preconditioning is only applied in 4D. The operators are preconditioned according to the scheme

$$\begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ M_{oe}M_{ee}^{-1} & 1 \end{pmatrix} \begin{pmatrix} M_{ee} & 0 \\ 0 & M_{oo} - M_{oe}M_{ee}^{-1}M_{eo} \end{pmatrix} \begin{pmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{pmatrix}$$

The preconditioned operator is: $M_{oo} - M_{oe}M_{ee}^{-1}M_{eo}$. Typically, the Vector Dslash routine is used in the M_{eo} and M_{oe} and the M_{ee} and M_{oo} are implemented in QDP++ as is the M_{ee}^{-1} .

In the CG-DWF the preconditioning is internal and only the solution to the unpreconditioned system is currently exposed.

2.5 Restrictions

We are aware of the following restrictions:

- Coupling to BAGEL
- Assembler Restriction: BAGEL produces assembler for use by GNU as. This is not currently directly consumable by the xlc toolchain on the BG/L (there was a hacked version floating around that could produce code for xlc at one point)
- Precision Restriction: The *bgl* code generator in BAGEL currently produces code so that client routines must use double precision.
- Layout Restriction: BAGEL Wilson Dslash requires at least two sublattice dimensions to be even
- Layout Restriction: QDP++ 2 Checkerboard layout requires that the first dimension of the local lattice be even.

2.6 Conventions

Normalisation of the fermion fields is unity. The Domain Wall height, should float between 0 and 2.

2.7 Known Issues:

I have tried to use the sloppy precision feature, which uses single precision for the half spinors internally. This resulted in a massive speed increase, but unfortunately produced the wrong result. A fully double precision build works fine tho.

2.8 Performance Benchmarks

The following performance was observed on the MIT BG/L, on a $28^3 \times 64$ lattice, with $L_s = 16$:

The speed of 234.8 Gflops on $\frac{1}{2}$ tower is approximately 8.3% of peak.

	Flop Counting Formula	$\frac{1}{2}$ tower (1024 cores)
Global Volume		28x28x28x64
Local Volume		14x7x7x2
Virtual topology		2x4x4x32
Performance/core (Mflops)	chroma	229
Total performance (Gflops)	chroma	234.8

Table 1: DWF Inverter Benchmarks from Chroma Level 2 Inverter on MIT BG/L. (Propagator Inversion, Double Precision, Shamir DWF Operator)

3 CG-DWF

3.1 Overview

The CG-DWF package provides a level 3 inverter for the unpreconditioned Shamir Domain Wall operator. Internally it uses an even odd preconditioned Conjugate Gradients (on the Normal Equations). The code makes use of mostly vector types (and a very few lines of inline assembler) and gcc provided arithmetic on these types (this is different from compiler intrinsics). The package can be built on SSE, BG/L and AltiVec architectures.

3.2 Operators Supplied

The following operators are available:

- Shamir Domain Wall

3.3 Data Layout Comments

User must support indexing functions for reading and writing from/to the external Gauge and Fermion fields. The user then supplies lattice layout information to the package, which will then use the indexing reader functions to import the gauge and fermion fields into its own internal layout. On exit, the resulting fields must be saved, which use the indexing writer functions to fill out the external data structure from the internal one.

3.4 Conventions

The domain wall height floats between 0 and 1, Fermion field normalisation is unity.

3.5 Preconditioning

The preconditioning is internal and follows the following scheme

$$\begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} = \begin{pmatrix} M_{ee} & 0 \\ M_{oe}M_{ee}^{-1} & M_{oo} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 - M_{oo}^{-1}M_{oe}M_{ee}^{-1}M_{eo} \end{pmatrix} \begin{pmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{pmatrix}$$

The preconditioned operator is: $1 - M_{oo}^{-1}M_{oe}M_{ee}^{-1}M_{eo}$.

In the CG-DWF the preconditioning is internal and only the solution to the unpreconditioned system is currently exposed.

3.6 Restrictions

We are aware of the following restrictions:

- Coupling to special version of GCC for BG/L. This version of gcc, cannot create executables due to its internal use of newlib. However it can easily compile to object format and the linking step can be done with another compiler.
- Precision Restriction: none - both single and double precision are supported. Single and double precision solvers can be in scope at the same time.
- Layout Restriction: The global size of each dimension must be even, but no restriction on local sublattice size except that total number of sites in the 5D lattice must be divisible by 4 (for single precision) or 2 (for double precision).
- Layout Restriction: If used from within QDP++ or Chroma: QDP++ 2 Checkerboard layout requires that the first dimension of the local lattice be even.

3.7 Performance Benchmarks

The following performance was observed on the MIT BG/L, on a $28^3 \times 64$ lattice, with $L_s = 16$:

	Flop Counting Formula	$\frac{1}{2}$ tower (1024 cores)	Full tower (2048 cores)
Global Volume		64x28x28x28	64x28x28x28
Local Volume		4x7x7x7	2x7x7x7
Virtual topology		16x4x4x4	32x4x4x4
Performance/core (Mflops)	chroma	206.6	190.8
Total performance (Gflops)	chroma	211.5	390.6
Total Performance (Gflops)	AVP	230	424.9

Table 2: Single Precision DWF Propagator Inversion using the CG-DWF from within Chroma. Shamir DWF Operator

Initially we did not understand why the numbers from chroma were so much below what Andrew’s independent tests produced. Eventually we just discovered that we counted flops using different formulae. Chroma undercounted flops compared with Andrew’s formula. The difference is probably due to using a different preconditioning style. Retrofitting the counting from Andrews independent test into chroma allowed us to reach much closer numbers. We included both ways of counting in table 2 above.

The speed of 211 – 230 Gflops on $\frac{1}{2}$ tower is approximately 7.5%-8.2% of peak. The speed of 424.9 Gflops on a full tower is approximately 7.6% of peak. These tests were carried out running the inverter from within Chroma on the MIT BG/L using the single precision version of the inverter.

For comparison, the following hard scaling numbers were presented by Andrew in Boston (for comparison) with an independent benchmark of his own:

Here the Block refers to the number of nodes (each node has 2 cores)/ The sublattice sizes agree with the chroma measurements, but the position of the short dimension is different which can have a performance effect (and appears to do so)

Block/mode/precision	Sublattice	Total Performance (Gflops=1024 Mflops)
512/vn/float	7x7x7x4	242.8
1024/vn/float	7x7x7x2	489.3

Table 3: Extracted rows from independent Hard Scaling Test Table

4 CPS

4.1 Overview

The CPS code uses Pavlos Vranas' Dslash routine for BG/L. Both Pavlos' Dslash and the Vector Dslash assembler routines from BAGEL for QCDOC are included in the CPS code and so the BAGEL Wilson Dslash package does not need to be separately installed.

4.2 Operators Supplied

The following operators are available:

- Shamir DWF, 5D-preconditioned CG, multimag inverter

4.3 Data Layout Comments

Written to work with CPS from the get go. No major packing of the fields is required except that the gauge fields is converted to CPS wilson fermion order (checkerboarded, odd first).

4.4 Conventions

The domain wall height floats between 0 and 2 (1.8 for most of actual measurements). Fermion field normalisation is $\frac{1}{\sqrt{2\kappa}}$ where κ is defined in a 5D way:

$$\kappa = (5 - M)$$

where M is the Domain Wall height. The internal operator is:

$$D = 1 - \kappa D_5$$

where D_5 is a simple 5 dimensional vector Dslash build up of the usual 4 dimensional Dslash-es.

4.5 Preconditioning

The fields are "preconditioned in 5D", which alternates between odd and even 4D slices. The preconditioned operator is

$$\tilde{D} = 1 - \kappa^2 D_5^{oe} D_5^{eo}$$

this is a Schur Decomposition where $D_5^{ee} = (D_5^{ee})^{-1} = 1$

An advantage of this preconditioning is that one does not need to calculate M_{ee}^{-1} terms and so some overhead is saved.

4.6 Restrictions

We are aware of the following restrictions:

- CPS restriction: Sublattice size must be even in all dimensions (Pavlos' assembly can deal with odd size lattices)
- Can only use "natural" machine topology. Rotation of axis is allowed.
- Only double precision assemblies available.

4.7 Performance Benchmarks

Flop Counting Formula	512 nodes (1024 core), double precision		
Global Volume		32x32x32x8x16	32x32x32x64x16
Local Volume		4x4x4x4x16	8x4x4x16
Virtual topology		8x8x8x2	8x8x8x2
CG Performance/core (Mflops)	CPS	354	276
MInv Performance/core (Mflops)	CPS	343	262

Table 4: DWF Propagator Inversion using CPS, with Pavlos Vranas' optimized code . Shamir DWF Operator.

The benchmark was run with the environment variable `BGL_APP_L1_SWOA=1`, which turns on store without allocate for all memory regions. CPS counts 1368 flops per checkerboard sites.

5 QOPQDP (Level 3 over QDP/C)

5.1 Overview

QOPQDP is a library of level 3 routines built on top of QDP/C. It currently provides routines for an improved gauge force, asqtad inverter and force and Wilson inverter in addition to the domain wall inverter discussed here. It requires the SciDAC QDP/C, QLA, QIO and QMP libraries and is portable to any platform supporting those. It therefore also makes use of any platform specific optimizations provided by those libraries.

5.2 Operators Supplied

The following domain wall operators are currently available:

- Shamir Domain Wall

Adding new operators is fairly simple since it is written entirely in QDP.

5.3 Data Layout Comments

The internal data layout comes from QDP/C and QLA. There are routines which allow QDP fields to be directly converted to/from level 3 fields. There is also an interface that allows passing fields in a "raw" format that the user must copy their data to/from.

5.4 Conventions

It currently uses the same conventions as CG-DWF, but can easily be modified to other conventions.

5.5 Preconditioning

It currently uses the same preconditioning as CG-DWF, but again can easily be modified.

5.6 Restrictions

We are aware of the following restrictions:

- **Layout Restriction:** The global 4D lattice size must be even in all directions. The only restriction on L_s is that $L_s \geq 2$ (including odd values).
- **Precision Restriction:** None - both single and double precision are supported. Single and double precision solvers can be in scope at the same time.
- **QLA Restriction:** Good performance on BG/L requires a new version of QLA which is still under development. The new version needs XLC v8 to generate the double hummer code. The new QLA is currently available for testing only. It does however seem to be fully functional and once the test suite is working on BG/L and all routines have passed, it should be ready for production.
- **QMP Restriction:** The best performance on BG/L requires a new version of QMP which is still under development. However depending on the lattice size used, the MPI version may be sufficient. The new QMP does not have any known bugs, but it is not yet complete. It is complete enough for many applications, but may not work yet for certain combinations of application, lattice size, etc. The current code could be made production ready fairly soon if needed, but completing it and doing the necessary testing and performance tuning will still take a while.
- **Documentation Restriction:** The documentation is very poor, mainly because there isn't any.

5.7 Performance Benchmarks

The following performance was observed on a half tower (1024 cores) of the BU BG/L with a $28^3 \times 64$ lattice at $L_s = 16$:

	Flop Counting Formula	single precision	double precision
Global Volume		28x28x28x64	28x28x28x64
Local Volume		7x7x7x4	7x7x7x4
Virtual topology		4x4x4x16	4x4x4x16
Performance/core (Mflops)	JCO	409	267
Total Performance (Gflops)	JCO	419	273

Table 5: DWF Propagator Inversion using QOPQDP. Shamir DWF Operator

Notes:

- The benchmarks make use of the current development versions of QLA and QMP mentioned above.

- The inverter was run from a benchmark program in the QOPQDP package, not from within real application code.
- The inverter has several parameters that can be changed by the user to help tune the performance. The benchmark program runs through essentially all combinations of the parameters to find the best ones. The numbers reported are for that one and should be reproducible in application code if the same parameters are set.
- The benchmark was run with the environment variable `BGL_APP_L1_SWOA=1`, which turns on store without allocate for all memory regions. This generally provides some improvement in the codes I have tested. This attribute could be put directly in the QDP memory allocations (and in QMP) if it turns out to be undesirable for the rest of the application.

The single precision performance above is 14.6% of peak and the double precision performance is 9.5% of peak.