Implementation of Twisted Mass Fermion operator in QUDA

A. Strelchenko[a], D. Alexandrou[b], G. Koutsou[b], A. Vaquero[b]

a. Scientific Computing Division @ Fermilab
b. CaSToRC @ The Cyprus Institute

Lattice 2013, Mainz, August 2

Outline

- Brief overview of the project
- Twisted mass fermions in QUDA
- Performance summary
- Conclusion

Brief overview of the project

- TM development was partly supported by Cyprus RPF
  - grant ΠΡΟΣΕΛΚΥΣΗ/ΠΡΟΝΕ 0308/09
- and *PRACE-1IP*
  - WP-7.5-1IP, grants RI-211528 and FP7-261557
  - A.S. *et. al.*, *PRACE* white-paper 2012
- Degenerate TM support since QUDA-0.3.1
- Non-degenerate TM support since QUDA-0.5.0
- Code is currently in production
  - next report by Alejandro Vaquero

Twisted Mass fermions in QUDA: general info

- The Twisted Mass fermions [*Frezotti, Rossi, 2004*]:

$$\not{D}_{TM} = \not{D}_W I_f + i\mu\gamma_5\tau_3 + \epsilon\tau_1,$$

where:

$\not{D}_W$ is the Wilson-Dirac matrix

μ, ϵ - the twisted mass parameters

- Available resources:
  - tmLQCD (*C.Urbach*, Fri 9G & *B. Kostrzewa*, Fri 10G)
  - a number of custom codes (e.g.: *M. Bach et. al.*, Tue 4G)

Twisted Mass fermions in QUDA

- TM fermion operator:
  - optimizations (e.g., cache blocking),
  - multi-GPU (4-dim splitting),
  - autotuning framework
- Non-degenerate TM operator peculiarities:
  - more complecated case due to off-diagonal elements
  - more resource-demanding: register spilling is more severe

Twisted Mass fermions in QUDA

- Twisted mass code inherits Wilson Dslash code design:
  - separate internal/external CUDA kernels for Multi-GPU
  - C++ interface to GPU kernels
  - packing routines for Multi-GPU communications

- End-user configuration/setup:
  - TM stuff is enabled via options:

./configure ... –enable-[ndeg-]twisted-mass-dirac ...

  - Dslash type:

QudaDslashType dslash_type = QUDA_TWISTED_MASS_DSLASH;

  - TM parameters and flavor type:

inv_param.mu = $\bar{\mu}$;
inv_param.epsilon = $\bar{\epsilon}$;

inv_param.twist_flavor = QUDA_TWIST_NONDEG_DOUBLET; //or QUDA_TWIST_[PLUS/MINUS]

Twisted Mass in QUDA: GPU kernels

- Asymmetric preconditioning:

  $a)$ $(R_{ee} - \kappa^2 \not{D}_{eo} R_{oo}^{-1} \not{D}_{oe}) \psi_e = b_e - \not{D}_{eo} R_{oo}^{-1} b_o;$

  $b)$ $\psi_o = R_{oo}^{-1} (b_o - \not{D}_{oe} R_{ee}^{-1} \psi_e).$

  - Required additional new kernels:

  $R^{-1} \not{D}_W, \quad (R - \kappa^2 \not{D}_W)$

  - ... and their 'daggered' analogs:

  $(R^{-1})^\dagger \not{D}_W^\dagger, \quad (R^\dagger - \kappa^2 \not{D}_W^\dagger)$

- Symmetric preconditioning:

  $a')$ $(I_{ee} - \kappa^2 R_{ee}^{-1} \not{D}_{eo} R_{oo}^{-1} \not{D}_{oe}) \psi_e = R_{oo}^{-1} (b_e - \not{D}_{eo} R_{oo}^{-1} b_o);$

  - ... and four more kernels for this case:

  $\not{D}_W R^{-1}, \quad (I - \kappa^2 R^{-1} \not{D}_W), \quad \not{D}_W^\dagger (R^{-1})^\dagger, \quad (I - \kappa^2 \not{D}_W^\dagger (R^{-1})^\dagger)$

Twisted Mass in QUDA: GPU kernels (cont.)

- Flavor matrix (degenerate case):

$R^{\pm} = (1 \pm i2\kappa\mu\gamma_5)$;

- Flavor matrix (non-degenerate case):

$R = (I + i2\kappa\bar{\mu}\gamma_5\tau^3 - 2\kappa\bar{\epsilon}\tau^1)$;

- Spinor face packing kernels:
  - standard Wilson packing: $\psi_{face} \rightarrow P^{\pm}_{\mu}\psi_{face}$
  - modified version for TM: $\psi_{face} \rightarrow P^{\pm}_{\mu}R^{-1}\psi_{face}$

Twisted Mass in QUDA: C++ interface

- Extra classes to provide TM-specific functionality

```
// Full twisted mass
class DiracTwistedMass : public DiracWilson {

  protected:
      double mu;
      double epsilon;
       ...
  public:
      DiracTwistedMass(const DiracTwistedMass&);
      DiracTwistedMass(const DiracParam&, const int );
      virtual ~DiracTwistedMass();
       ...
};

// Even-odd preconditioned twisted mass
class DiracTwistedMassPC : public DiracTwistedMass {

  public:
      DiracTwistedMassPC(const DiracTwistedMassPC&);
      DiracTwistedMassPC(const DiracParam&, const int );
      virtual ~DiracTwistedMassPC();
       ...
};
```

HW/SW configuration

- NVIDIA GTX Titan summary:
  - GK110 architecture (cc3.5), 2688 CUDA cores
  - 1.5(?)/4.5 TFlops double/single performance
  - 6 GB memory, 288 GB/secs BW
- NVIDIA K20M summary:
  - GK110 architecture (cc3.5), 2496 CUDA cores
  - 1.17/3.52 TFlops double/single performance
  - 5 GB memory, 208 GB/secs BW

- GTX Titan SW environment:
  - Linux: Scientific Linux v 6.3
  - nvcc-5.5, gcc-4.4.6
- JLab k20m cluster SW environment:
  - Linux: RH v 6.3
  - nvcc-5.0, gcc-4.6.3
  - MPI: mvapich2-1.8

Performance summary: single GPU

- Preconditioned Dslash $(32^3 \times 64)$: $R_{ee} - \kappa^2 \not{D}_{eo} R_{oo}^{-1} \not{D}_{oe}$

- reconstruction 12

Prec. (GFlops)	Wilson	Deg. TM	Non-deg TM
double	125	116	126
single	401	415	516
half	732	759	879

- reconstruction 8

double	68	70	94
single	472	487	567
half	829	858	624

Performance summary: degenerate TM fermions

CG solver (symmetric vs. asymmetric preconditioning):

- $L = 48, T = 96$ lattice with $\kappa = 0.156361$, $\mu = 0.0015$
- double-single mixed precision (tol=1e-6), reconstruction 8

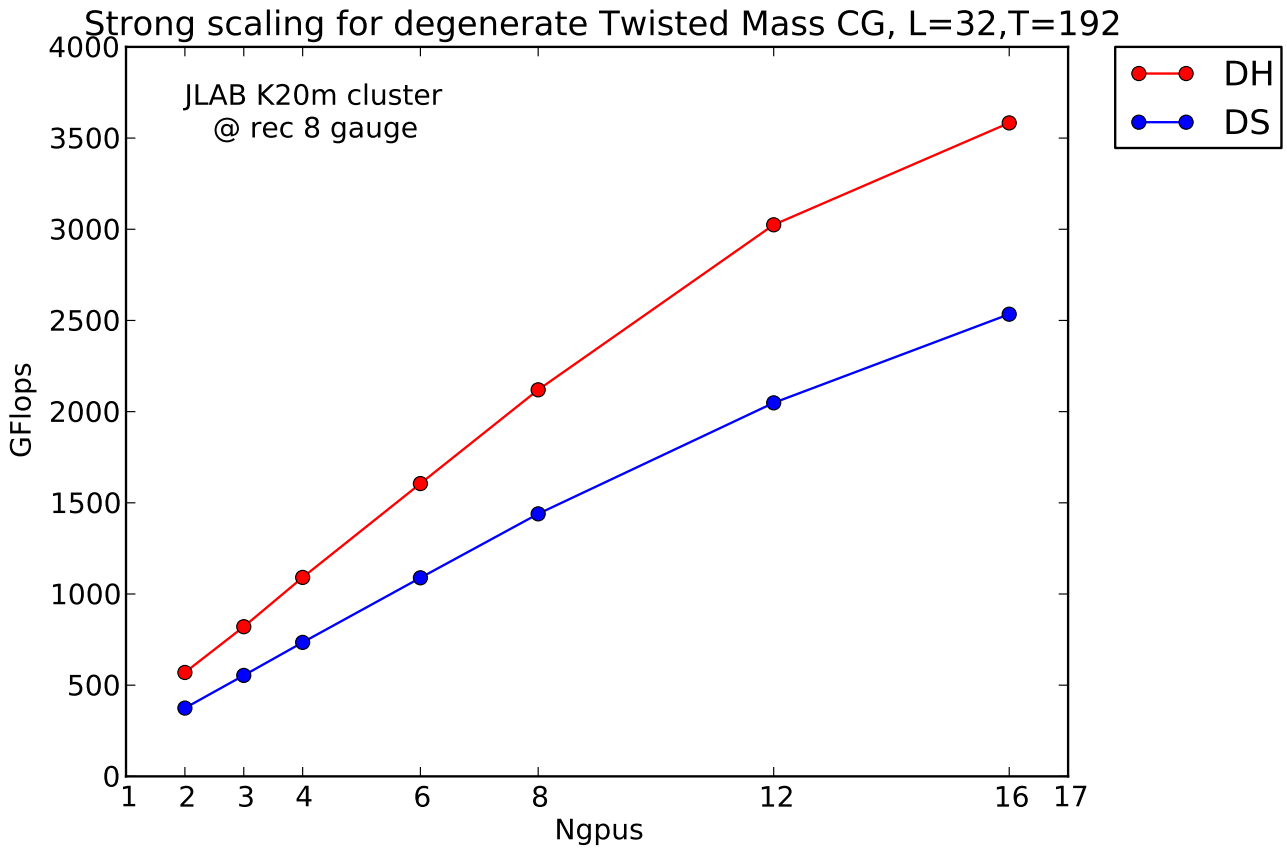
N GPUs	Asymm. (iter/secs)	Symm. (iter/secs)	Speedup
4	3130 / 126.28	3131 / 132.72	5%
6	3111 / 84.99	3111 / 89.72	6%
8	3169 / 71.93	3169 / 79.47	10%

- double-half mixed precision (tol=1e-4), reconstruction 8

N GPUs	Asymm. (iter/secs)	Symm. (iter/secs)	Speedup
4	149 / 4.00	149 / 4.25	6%
6	156 / 2.79	156 / 2.94	5%
8	172 / 2.42	172 / 2.59	7%

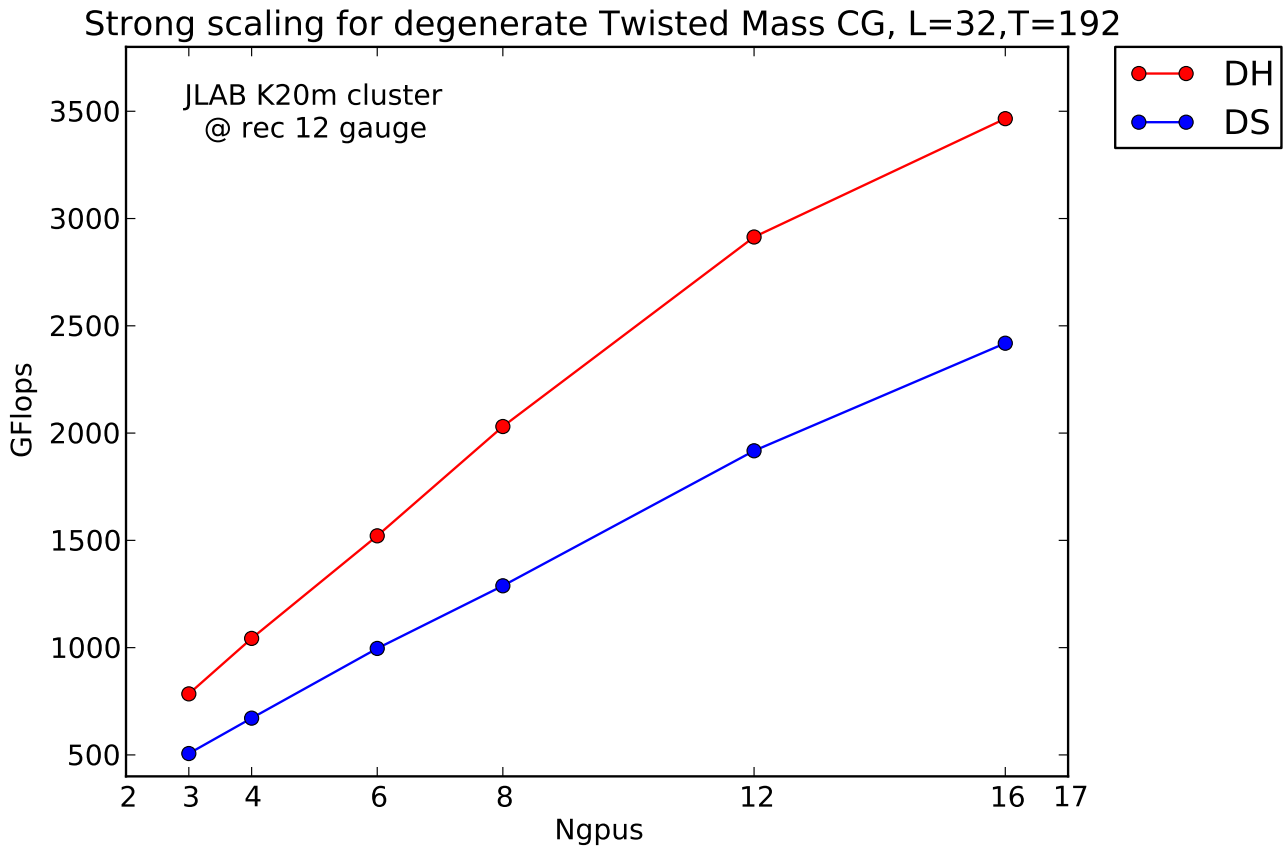
Performance summary: degenerate TM fermions

Strong of the degenerate twisted mass CG solver, reconstr. 8



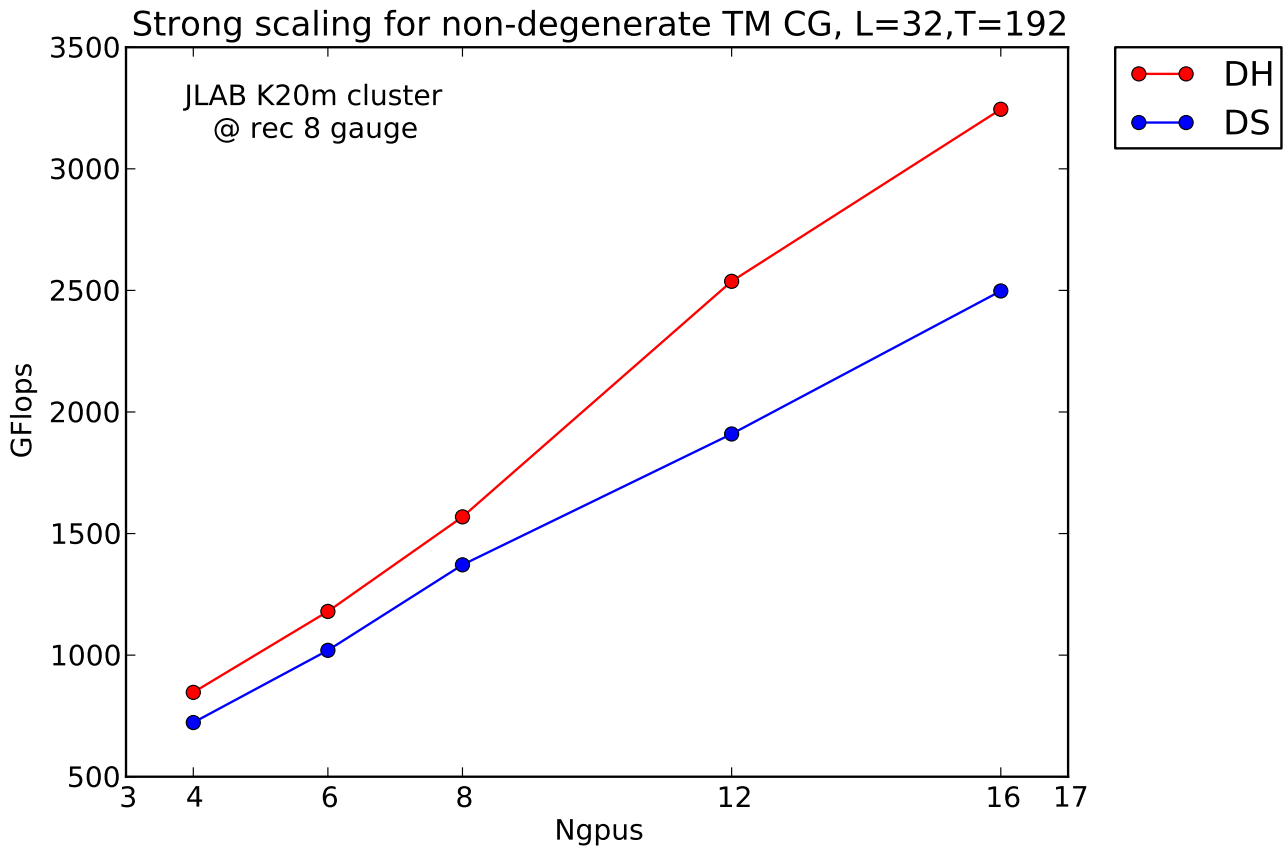
Performance summary: degenerate TM fermions

Strong of the degenerate twisted mass CG solver, reconstr. 12



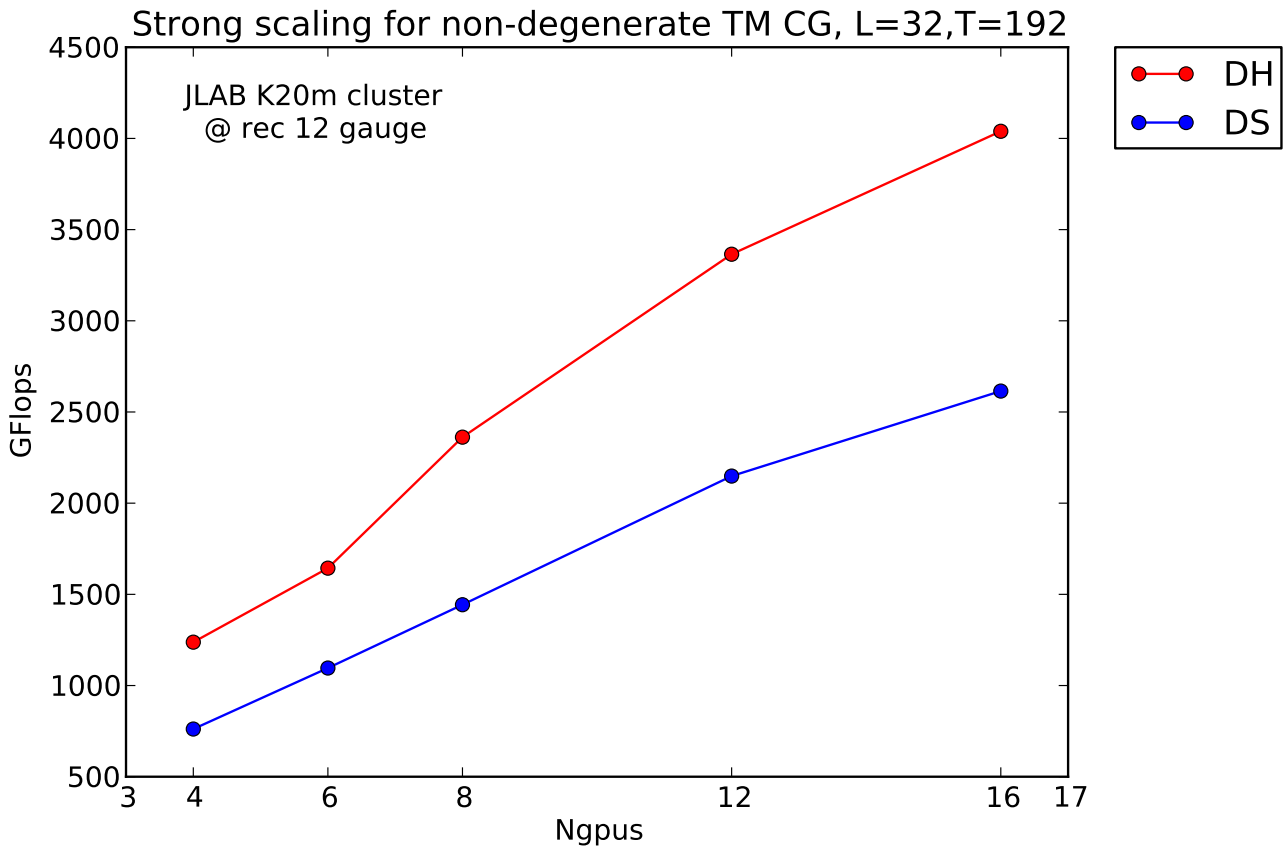
Performance summary: non-degenerate TM fermions

Strong of the non-deg. twisted mass CG solver, reconstruction 8



Performance summary: non-degenerate TM fermions

Strong of the non-deg. twisted mass CG solver, reconstruction 12



Conclusion

- Twisted mass fermions in QUDA:
  - full twisted mass fermion action
  - asymmetric preconditioning is slightly faster
  - reconstr. 8 works better for the degenerate case
- Future work:
  - integration of the disconn. stuff (report by A.V.)
  - more solvers (e.g., deflated solvers)
  - I/O libraries (e.g., Lemon library)
  - integration into tmLQCD

Thank you!