Porting DDalphaAMG solver to K computerPorting DDalphaAMG solver to K computer

Issaku Kanamori and Ken-Ichi Ishikawa (Hiroshima Univ.)

Background: Algebraic Multigrid Solver

Linear Solver for the Dirac eq.: a bottleneck of the Lattice QCD
Critical slowing down: as the quark becomes lighter, and the lattice becomes finer, it takes more and more time to solve
multigird solver: very mild slowing down

application to QCD: R. Babich, J. Brannick, R.C.Brower et al. PRL 105 (2010) 201602

idea: solving the equation in low mode space (=coarser lattice)

made of:
coarse lattice part (coarse grid solver) efficient for low mode
fine lattice part (smoother + outer solver) efficient for high mode

building a good coarse grid operator is important: adaptive method

1. initial random vectors $|i\rangle$ $(n \sim 20)$
2. initial coarce op. with $|i\rangle$
3. $D^{-1}|i\rangle$: low mode rich, gives improved $|i\rangle$
4. improve the coarse op. with the improved $|i\rangle$
5. go back to 3.

pos: fast even with physical quark mass
coarse grid solver can have a coarser gird : multigrid
con: complicated, some parameters to tune

K computer and details of the tuning

http://www.r-ccs.riken.jp/jp/k/about.html

available since 2012 --- not new any more
3rd in the latest HPCG (1st till last November)
SPARC64 VIIIfx processors, Tofu interconnect
128 GFlops/node
256 registers/core (128 bit), no single prec. arithmetics
[1 register = 1 complex number, rich intrinsics for complex arithmetics]
hardware counter: easy to check the efficiency from the portal

Tuning: intrinsics + lots of register variables
matrix mult on vectors (clover term, coarse grid op.), hop of Dirac op.
clover term: upper/lower half of the clover (6 real + 15 cmpl.) + input (6 cmpl.) + output (6 cmpl) and working variables on the register
coarse grid operator: 4x4 blocking
4= chirality (2) x 2 from test vecor (# test vector must be even)

DDalphaAMG

https://github.com/DDalphaAMG

A. Frommer, K. Kahl, S. Krieg, B. Leder and M. Rottmann, SIAM J. Sci. Comput. 36 (2014) A1581

adaptive aggregation based domain decomposed algebraic multigrid method
smoother: multiplicative Schwartz Alternative Procedure
coarse grid solver: even odd preconditioned GMRES
outer solver: FGMRES

test vectors (low mode rich)
$|\lambda_i\rangle (i = 1, \ldots, n; n \sim 20)$

domain decomposition
each domain (X) contains $\sim 4\times 4\times 4\times 4$ lattice sites

basis for projection in domain X

$$|\lambda_i(X, s=+/-)\rangle \equiv \begin{cases} \frac{1\pm\gamma_5}{2}|\lambda_i\rangle & (x \in X) \\ 0 & \end{cases}$$

make a coarse operator

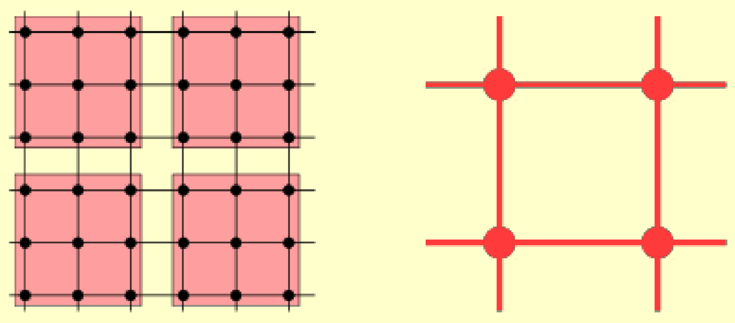
$$D \Rightarrow D_{\text{coarse}}(s,i,X;t,j,Y) = \langle \lambda_i(s,X)|D|\lambda_j(t,Y)\rangle$$

projection of the source to the coarse grid

$$|b\rangle \Rightarrow b_{\text{coarse}}(i,s,X) = \langle \lambda_i(s,X)|b\rangle$$

prolongation of the solution to the fine grid

$$|x'\rangle = \sum_{i,s,X} x_{\text{coarse}}(i,s,X)|\lambda_i(s,X)\rangle \Leftarrow x_{\text{coarse}}(i,s,X)$$



12 d.o.f/site	$2n$ d.o.f/site	vector (site)
$D : (3\times 3)\otimes(4\times 4)$	$D_{\text{coarse}} : 2n \times 2n$	Dirac op. (site, link)

Appendix: code example

clover term mult (fine lattice) uses 33 register variables

data layout: $|\underbrace{r_{00}\ r_{11}}_{1\ \text{simd}}|r_{22}\ r_{33}|r_{44}\ r_{44}|r_{01}\ i_{01}|r_{02}\ i_{02}|\cdots$

```
static inline void K_site_clover_2spin_float( float *eta, const float *phi, float *clover ) {

    register _fjsp_v2r8 in0,in1,in2,in3,in4,in5;
    register _fjsp_v2r8 out0,out1,out2,out3,out4,out5;

    register _fjsp_v2r8 clov0,clov1,clov2;
    register _fjsp_v2r8 clov01,clov02,clov03,clov04,clov05,
      clov12,clov13,clov14,clov15,
      clov23,clov24,clov25,
      clov34,clov35,
      clov45;
    register _fjsp_v2r8 diag0,diag1,diag2,diag3,diag4,diag5;

    // load input spinor
    register ssu3ferm2* in_ptr=(ssu3ferm2*)phi;
    load_ferm2(in,in_ptr);

    // load clover term
    register ssu3clov2* clov_ptr=(ssu3clov2*)clover;
    load_clov2(clov, clov_ptr);

    // extract diagnaols (=they are real)
    diag0 =_fjsp_unpacklo_v2r8(clov0,clov0);
    diag1 =_fjsp_unpackhi_v2r8(clov0,clov0);
    diag2 =_fjsp_unpacklo_v2r8(clov1,clov1);
    ...

    // 1st line
    rmult(out0,diag0,in0);
    accum_cmult(out0, clov01, in1);
    accum_cmult(out0, clov02, in2);
    accum_cmult(out0, clov03, in3);
    accum_cmult(out0, clov04, in4);
    accum_cmult(out0, clov05, in5);

    // 2nd line
    ...
```

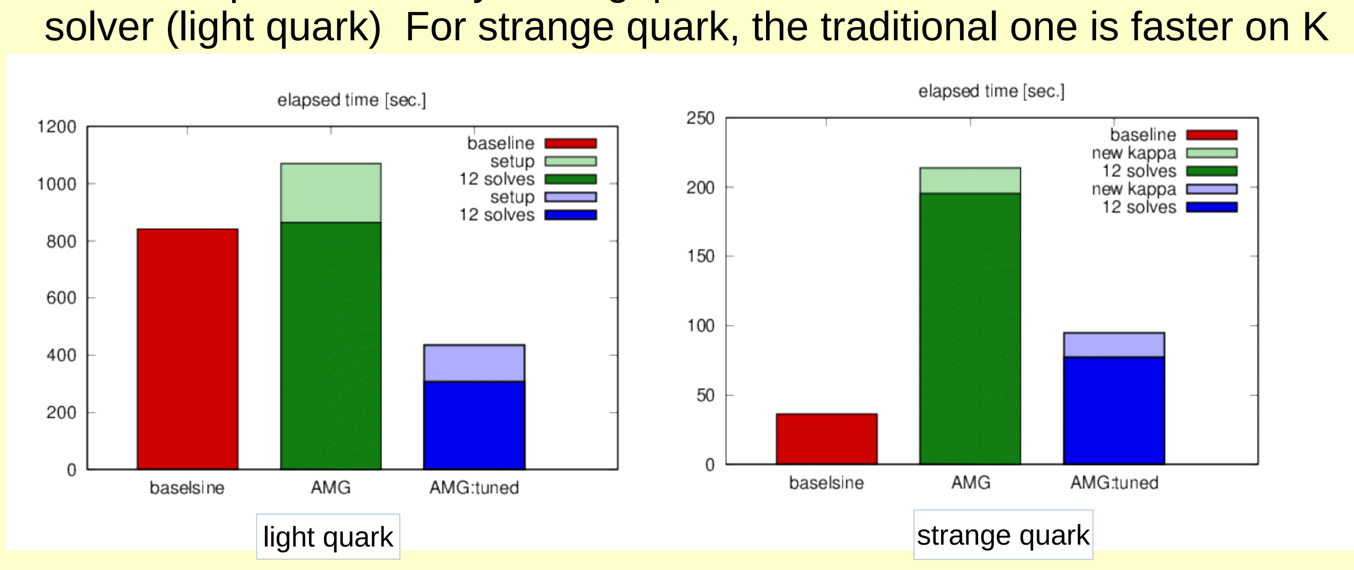
some macros

```
#define load_ferm2( d, s ){                  \
  d ## 0 =  _fjsp_stod_v2r8(s->y[0]);    \
  d ## 1 =  _fjsp_stod_v2r8(s->y[1]);    \
  ...                                     \
  d ## 5 =  _fjsp_stod_v2r8(s->y[5]);    \
}

#define store_ferm2( s, d ){                 \
    s->y[0] = _fjsp_dtos_v2r4(d ## 0 ); \
    s->y[1] = _fjsp_dtos_v2r4(d ## 1 ); \
    ...                                   \
    s->y[5] = _fjsp_dtos_v2r4(d ## 5 ); \
  }

// c=a*b as real
#define rmult(c,a,b){       \
  c=_fjsp_mul_v2r8(a,b);    \
}

// c+=a*b
#define accum_cmult(c,a,b){                          \
    c=_fjsp_madd_cp_v2r8(a,b,c);                     \
    c=_fjsp_nmsub_cp_neg_sr12_v2r8(a,b,c);      \
}
```

complex mult add with only 2 intrinsics

Benchmark results

even with poor efficiency, throughput is better than the well tuned traditional solver (light quark) For strange quark, the traditional one is faster on K



Configuration: n_f=2+1 clover, 96^4 lattice, m_{pi} = 146 MeV, 1/a=2.33 GeV [PACS]

2048 nodes, 2 level method

baseline: well tuned solver for K [efficiency: 22%]*
mixed precision nested BiCGstab, where the single precision solver uses Domain decomposition (block size=12x12x12x12, NSAP=5).
The solver inside the domain uses SSOR method with sub-blocking.
cf. K.-I.Ishikawa et al [PACS collaboration]., PoS LATTICE2015(2016) 075

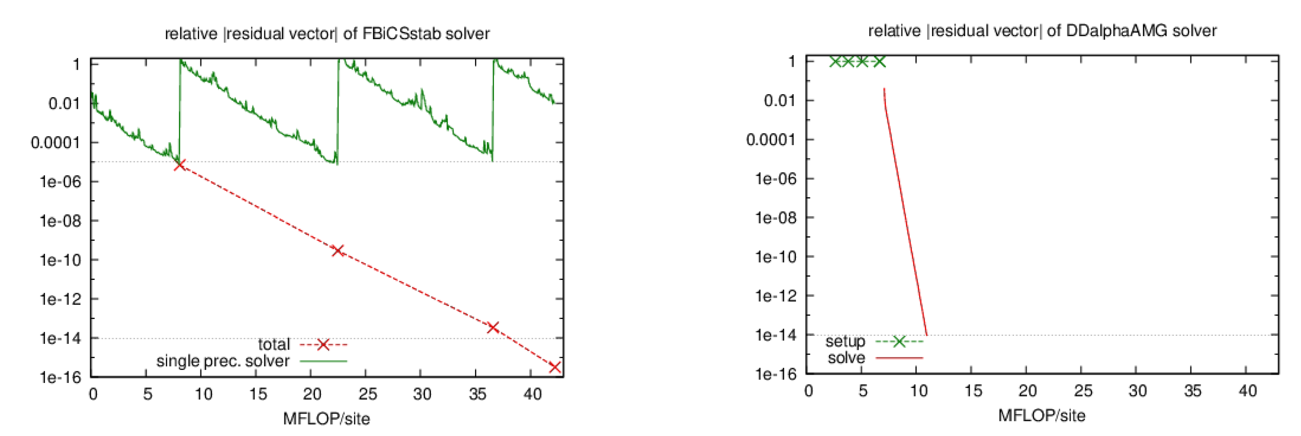
AMG(before tuning) [efficiency 3.0%]*

AMG: tuned [effcicency 5.3%]* THIS WORK
cf. https://github.com/i-kanamori/DDalphaAMG/tree/K/
block size=4x4x4x4, NSAP=4, # testvector=16
local volume: $12\times 12\times 12\times 24 \Rightarrow 3\times 3\times 3\times 6$
site d.o.f : $12 \Rightarrow 32$

*by hardware counter (contains contractions for meson spectroscopy + I/O); theoretical value is 10%-20% lower

OMP threading: 8 threads/node, different stragegy
inside a domain (12x12x12x12 sites) vs. domains (3x3x3x3 domains)
baseline DDalphaAMG

Convergence history vs. FLOP/site:
DDalphaAMG is 8x better for 1 propagator



42 MFLOP/site
(504 MFLOP/site for 12 solves)

setup 4.3 MFLOP/site
solve 6.7 MFLOP/site
(58.3 MFLOP/site for 1 setup + 12 solve)

Acknowledgements to...
Priority Issue 9 to be Tackled by Using Post K Computer, JICFuS
K computer (hp170229,hp180178)