Optimizing Wilson-Dirac operator and linear solvers for KNL

Thorsten Kurth, Balint Joo, Dhiraj Kalamkar, Aaron Walden Karthikeyan Vaidyanathan

IXPUG 2016 Frankfurt, Germany
June 23, 2016
**Lattice QCD**

- Straightforward way to solve QCD in non-perturbative regime with quantifiable uncertainties
- QCD is discretized on space-time grid with millions of DoF

\[
Z_E = \int DU_\mu D\psi D\bar{\psi} \exp \left( -S_g[U] - \int_{\mathbb{R}^4} d^4 x \bar{\psi}(x) D[U] \psi(x) \right)
\]

\[
= \int DU_\mu D\phi D\phi^\dagger \exp \left( -S_g[U] - \int_{\mathbb{R}^4} d^4 x \phi^\dagger(x) D[U]^{-\frac{1}{2}} \phi(x) \right)
\]

- Most time is spent in solving \((A - D\Phi)\psi = \chi\)
- Optimizing the solvers as well as \(D\Phi\psi, A\psi\) is important
Dirac Operators

- Wilson Operator

\[ \mathcal{D}(x, y) = \sum_{\mu=0}^{3} U_\mu(x)(1 - \gamma_\mu)\delta_{y,x+\hat{\mu}} + U_\mu^\dagger(x - \hat{\mu})(1 + \gamma_\mu)\delta_{y,x-\hat{\mu}} \]
Dirac Operators

- Wilson Operator

\[ \mathcal{D}(x, y) = \sum_{\mu=0}^{3} U_{\mu}(x)(1 - \gamma_{\mu}\delta_{y, x+\hat{\mu}}) + U_{\mu}^{\dagger}(x - \hat{\mu})(1 + \gamma_{\mu}\delta_{y, x-\hat{\mu}}) \]

sparse, only NN coupling
Dirac Operators

- Wilson Operator

\[ D(x, y) = \sum_{\mu=0}^{3} U_{\mu}(x)(1 - \gamma_{\mu}) \delta_{y, x+\hat{\mu}} + U_{\mu}^\dagger(x - \hat{\mu})(1 - \gamma_{\mu}) \delta_{y, x-\hat{\mu}} \]

very sparse Dirac matrices, implemented as functions
Dirac Operators

- Wilson Operator

$$\mathcal{D}(x, y) = \sum_{\mu=0}^{3} U_{\mu}(x)(1 - \gamma_{\mu})\delta_{y, x+\hat{\mu}} + U_{\mu}^\dagger(x - \hat{\mu})(1 + \gamma_{\mu})\delta_{y, x-\hat{\mu}}$$

Unitary 3x3 complex matrices (store 6 complex numbers)
Dirac Operators

- **Wilson Operator**

\[ \mathcal{D}(x, y) = \sum_{\mu=0}^{3} U_{\mu}(x) (1 - \gamma_{\mu}) \delta_{y,x+\hat{\mu}} + U_{\mu}^\dagger(x - \hat{\mu}) (1 + \gamma_{\mu}) \delta_{y,x-\hat{\mu}} \]

- **Clover Term**

\[ A(x) = (N_d + m) - i \frac{1}{8} c_{sw} \sigma_{\mu\nu} F_{\mu\nu}(x), \quad F_{\mu\nu}(x) = \frac{-i}{8} (Q_{\mu\nu}(x) - Q_{\nu\mu}(x)) \]

\[ \sigma_{\mu\nu} \equiv [\gamma_{\mu}, \gamma_{\nu}] \]

Gattringer, Lang: Quantum Chromodynamics on the Lattice
Arithmetic Intensity (Naive)

- Overview over Dslash key ingredients
  - links (U-matrices): 3x3 complex, unitary
  - spinors: 4x4 complex
  - 9-point stencil in 4D

- Thus:
  - read 8 neighboring spinors, 8 links, write central spinor

naive intensity: 0.92 flop/byte
Optimization Potential in Dslash

\[
\mathcal{D}(x, y) = \sum_{\mu=0}^{3} U_\mu(x)(1 - \gamma_\mu)\delta_{y,x+\hat{\mu}} + U_\mu^\dagger(x - \hat{\mu})(1 + \gamma_\mu)\delta_{y,x-\hat{\mu}}
\]

- even-odd Schur-preconditioning
  \[M_{oo} = A_{oo} - D_{oe}A_{ee}^{-1}D_{eo}\]
- streaming in t-direction: 7-of-8 neighbor reuse
- (temporal blocking)
- no reuse of links due to EO, but use 12-reconstruction

Smelyanski, Vaidyanathan, Joo, et. al.: High-Performance Lattice QCD for Multi-core Based Parallel Systems using a Cache-Friendly Hybrid Threaded-MPI Approach
Optimization Potential in Clover

- Clover Term:
  - $A$: in general 12x12 complex
  - Appropriate choices of $\gamma_{\mu}$: becomes block-diagonal, $w/2$ 6x6 hermitian blocks
  - $L^* D L$ decomposition: 12 reals + 30 complex numbers total per site
  - Inverse term $A^{-1}$ will be precomputed and stored (has similar structure)
- $A$ costs 504 FLOPS
- Data: 288B (clover term) + 2x96B (spinors) = 480B
- $A_l \approx 1$
Performance Bounds

- $R =$ no. of reused input spinors
- $r = 0$ streaming, $r = 1$ "read-for-write"
- $B_r =$ read bandwidth
- $B_w =$ write bandwidth
- $G =$ size of Link
- $S =$ size of Spinor

$$F = \frac{1320}{8G/B_r + (8 - R + r)S/B_r + S/B_w}$$
Performance Bounds

- $R$ = no. of reused input spinors
- $r = 0$ streaming, $r = 1$ „read-for-write“
- $B_r$ = read bandwidth
- $B_w$ = write bandwidth
- $G$ = size of Link
- $S$ = size of Spinor

\[
F = \frac{1320}{8G/B_r + (8 - R + r)S/B_r + S/B_w}
\]

<table>
<thead>
<tr>
<th>R</th>
<th>12-compress</th>
<th>AI (SP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no</td>
<td>0.92</td>
</tr>
<tr>
<td>0</td>
<td>yes</td>
<td>1.06</td>
</tr>
<tr>
<td>7</td>
<td>no</td>
<td>1.72</td>
</tr>
<tr>
<td>7</td>
<td>yes</td>
<td>2.29</td>
</tr>
</tbody>
</table>
QPhiX Data Layout

- partial SoA layout (AoSoA)
- pack ngy chunks of length soa from different y-coordinates
  - vec = vector length
  - soa = SoA-length
  - ngy = vec/soa
  - x-extent Lxh must be divisible by soa, block size by must be divisible by ngy
  - load-unpack/pack-store is faster than gather
- gauge fields constant, pre-gather ngy chunks
- padding helps alignment (after xy-planes)

Figure by Joo

B. Joo, Kalamkar, D., Vaidyanathan K. et. al.: Lattice QCD on Intel(R) Xeon Phi(tm) Coprocessors, ISC 2013
QPhiX Blocking

- 3.5D blocking
  - vectorize in x and y, block in y and z, stream in t
- How to assign blocks to cores, maintaining load balancing the same time?
  - multi-phase block allocation:
    - more blocks than cores: round-robin allocation
    - more cores than blocks: split in T to make more blocks than cores

Nguyen, A.D., Satish, N., Chhugani, J., Kim, C., Dubey, P.: 3.5-d blocking optimization for stencil computations on modern cpus and gpus, SC 2010
BLAS operations

• pool BLAS operations to maximize cache reuse
• use functor-based approach inspired by Kokkos

CopyFunctor( typename Geometry<FT,V,S,compress>::FourSpinorBlock* res_,
        const typename Geometry<FT,V,S,compress>::FourSpinorBlock* src_ );

AXPYFunctor(const AXPYFunctor<FT,V,S,compress>& rhs);

Norm2Functor(const typename Geometry<FT,V,S,compress>::FourSpinorBlock* x_);

XMYNorm2Functor(typename Geometry<FT,V,S,compress>::FourSpinorBlock* res_,
        const typename Geometry<FT,V,S,compress>::FourSpinorBlock* x_,
        const typename Geometry<FT,V,S,compress>::FourSpinorBlock* y_);
Test system: KNL-CPU

• Configuration A:
  ‣ Intel Knight’s Landing B0 7250 parts
  ‣ 68 cores@1.4 Ghz
• Configuration B:
  ‣ Intel Knight’s Landing B0 7210 parts
  ‣ 64 cores@1.4Ghz
• Common features:
  ‣ 4 HT per core, two 512bit VPU’s
  ‣ 16 GB MCDRAM
  ‣ 96 GB DDR
• Used configuration in both cases: quad-flat
• use KMP_PLACE_THREADS=1s<Nc>c<Nt>t
Results: single node KNL (A)

Wilson Dslash

GFLOPS

1 threads
2 threads
4 threads

SOA 4
DDR
MCDRAM

SOA 8
Haswell Dual Socket
DDR
MCDRAM
MCDRAM + Sfw + Hw Prefetch
MCDRAM + Sfw - Hw Prefetch

SOA 16
DDR
MCDRAM

V=32^4, by.bz=4, pxy=pxyz=1
Results: single node KNL (A)

Wilson CG

SOA 4
DDR
MCDRAM
SOA 8
Haswell Dual Socket
DDR
MCDRAM
MCDRAM + Sfw + Hw Prefetch
MCDRAM + Sfw - Hw Prefetch
SOA 16
DDR
MCDRAM

GFLOPS

V=32^4, by=bx=4, pxy=pxyz=1
Results: single node KNL (A)

Wilson BiCGStab

GFLOPS

V=32^4, by=bz=4, pxy=pxyz=1
Results: thread scaling (B)

Note: we did not tune layout parameter for optimal performance at given number of threads.
AVX512 does not contain AVX2 intrinsics except for _mm_prefetch
no gather intrinsics used, data properly packed
extensive use of fused multiply-adds
Comparison: AVX512 vs. AVX2 (B)

Relative Speedup of AVX512 over AVX2

consistent 20% speedup in all our kernels
**Test system: weak scaling**

- **Knight’s landing:**
  - KNL configuration B
  - Intel(R) OPA Host Fabric Interface: Series 100 ASIC (B0 silicon)
  - Intel(R) OPA Switch: Series 100 Edge Switch - 48 port (B0 silicon)
  - Intel MPI 5.1.2 (not most optimal choice, OpenMPI 10.0.1.50 works better according to Intel)

- **Haswell:**
  - NERSC Cori Phase 1, Cray XC
  - Dual socket Haswell, 32 cores@2.3 Ghz
  - 128 GB DDR
  - Cray Aries Interconnect with dragonfly topology
**Weak Scaling HSW vs. KNL (B)**

\[ V_{\text{socket}} = 32^4, \ bx = by = 8, \ pxy = pxyz = 0, \ SOA = 8, \ 32 \text{ threads/socket} \]
Weak Scaling HSW vs. KNL (B)

\[ V_{\text{socket}} = 32^4, \ bx = by = 8, \ pxy = pxyz = 0, \ \text{SOA} = 8, \ 32 \ \text{threads/socket} \]

additional communication in z-direction
Weak Scaling HSW vs. KNL (B)

\[ V_{\text{socket}}=32^4, \ bx=by=8, \ pxy=pxyz=0, \ SOA=8, \ 32 \text{ threads/socket} \]

additional communication in z-direction
Conclusions

- single node optimizations of QPhiX for Intel XeonPhi Knight’s Landing microarchitecture
- good thread scaling
- sustained max single-node performance of 505 GFLOPS/s (Dslash) from MCDRAM
- good weak scaling up to 16 KNL sockets, 3.5-4x improvement over 16 HSW sockets
- QPhiX is ready for Intel Knight’s Landing
Outlook

- AI of 2.29, MCDRAM BW of 450 GB/s: not yet at our performance limit (cache misses?)
- extend scaling study to O(1K) sockets
- measure strong scaling
- explore one-sided communication routines
- explore different SNC modes with MPI+OpenMP
- integrate into USQCD stack, i.e. plug in QDP-JIT as backend
Thank you
Backup
QPhiX SMT Threads & Prefetching

- maximize L1 coherence, i.e. loop over vectors in y and z
- prefetch spinor of next site
- prefetch links of next site
- prefetch y and z neighbors if they are off-core

Figure by Joo
Weak Scaling Cori Phase I

\( V_{\text{local}} = 16^4: \ bx = by = 4, \ pxy = pxyz = 0, \ SOA = 8, \ 32 \ \text{threads/socket} \)

\( V_{\text{local}} = 32^4: \ bx = by = 8, \ pxy = pxyz = 0, \ SOA = 8, \ 32 \ \text{threads/socket} \)