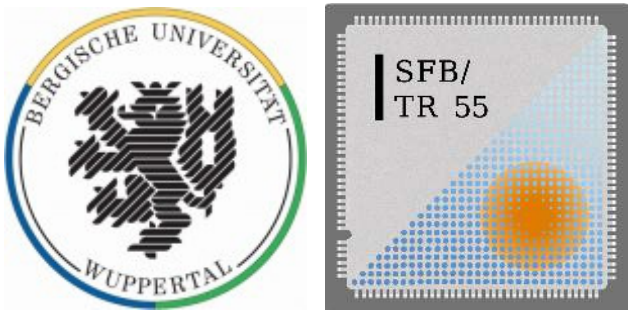
Optimization of the Brillouin operator on the KNL architecture

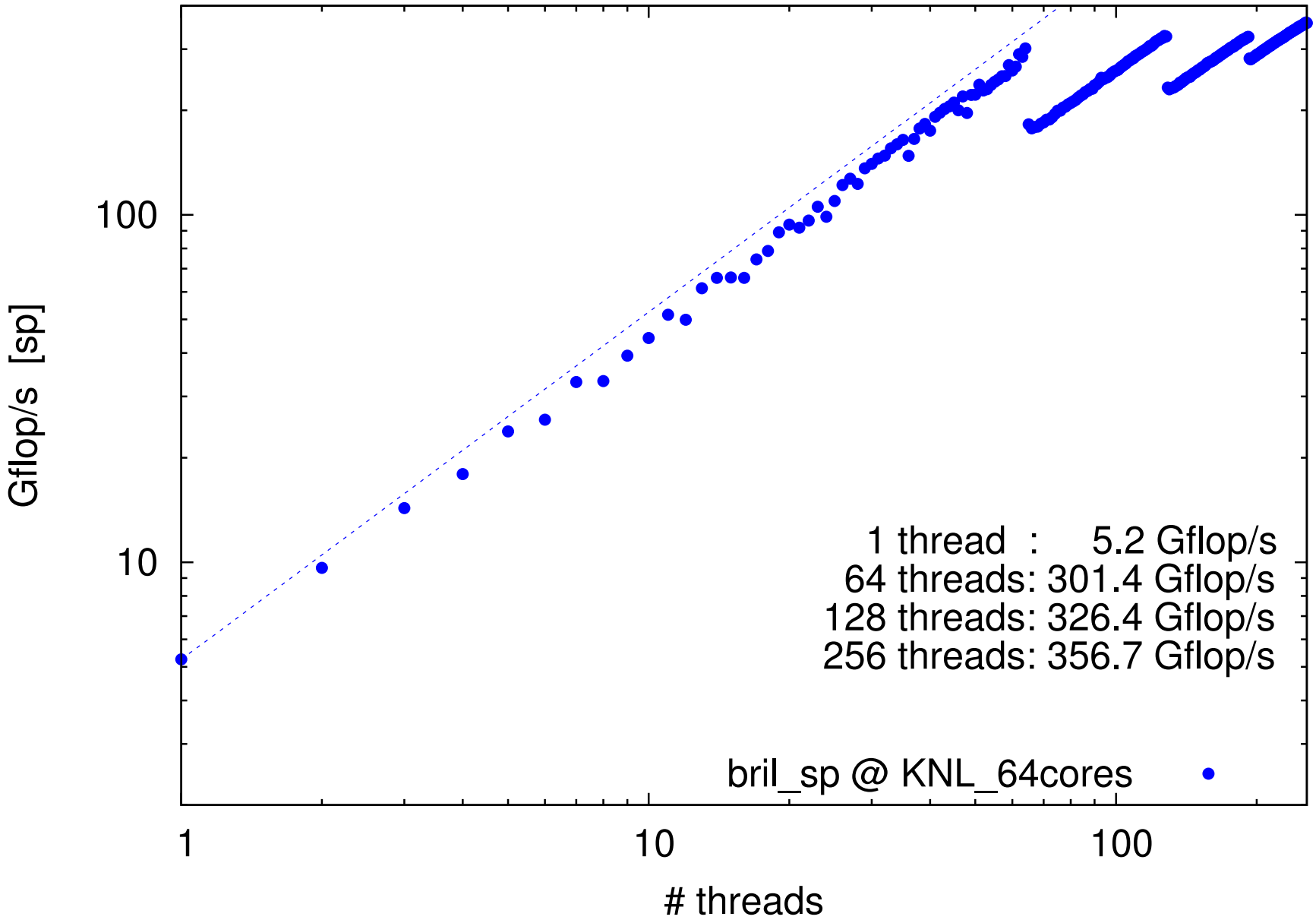
Stephan Dürr



University of Wuppertal
Jülich Supercomputing Center

Lattice 2017 – Algorithms and Machines – 19. June 2017

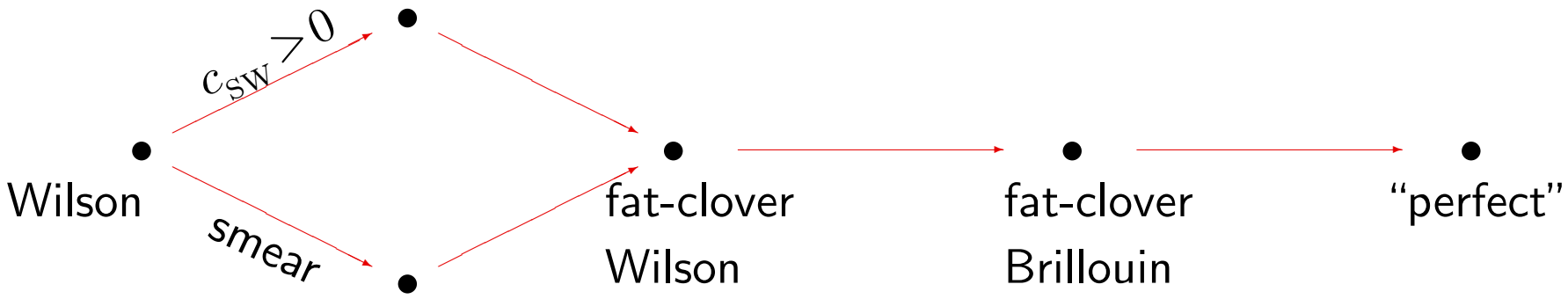
Brillouin matrix-times-vector thread-scaling on Intel KNL



Talk Overview

- What is the Brillouin operator ?
- Code suite design guidelines
- Brillouin kernel details
- Timing results on KNL
- Timing results on Core i7
- Wilson kernel details
- Timing results for Wilson

Dirac operator roadmap (pedestrian perspective)



Wilson Dirac operator:

$$D_{\rm W}(x,y) = \sum_\mu \gamma_\mu \nabla^{\rm std}_\mu(x,y) - \frac{a}{2}\triangle^{\rm std}(x,y) + m_0\delta_{x,y} - \frac{c_{\rm SW}}{2}\sum_{\mu<\nu}\sigma_{\mu\nu}F_{\mu\nu}\delta_{x,y}$$

Brillouin Dirac operator:

$$D_{\rm B}(x,y) = \sum_\mu \gamma_\mu \nabla^{\rm iso}_\mu(x,y) - \frac{a}{2}\triangle^{\rm bri}(x,y) + m_0\delta_{x,y} - \frac{c_{\rm SW}}{2}\sum_{\mu<\nu}\sigma_{\mu\nu}F_{\mu\nu}\delta_{x,y}$$

$\sigma_{\mu\nu} = \frac{\rm i}{2}[\gamma_\mu,\gamma_\nu]$, $F_{\mu\nu}$ the hermitean clover-leaf field-strength tensor, new $m_0, c_{\rm SW}$

Brillouin operator definition

$$(\rho_1,\rho_2,\rho_3,\rho_4)\equiv(64,16,4,1)/432,\quad (\lambda_0,\lambda_1,\lambda_2,\lambda_3,\lambda_4)\equiv(240,-8,-4,-2,-1)/128$$

$$\begin{aligned}
a\nabla^{\rm iso}_\mu(n,m) &= \rho_1\,[\delta_{n+\hat\mu,m}-\delta_{n-\hat\mu,m}]\\
&+ \rho_2\sum\nolimits_{\nu(\neq\mu)}[\delta_{n+\hat\mu+\hat\nu,m}-\delta_{n-\hat\mu+\hat\nu,m}]\\
&+ \rho_3\sum\nolimits_{\rho\neq\nu(\neq\mu)}[\delta_{n+\hat\mu+\hat\nu+\hat\rho,m}-\delta_{n-\hat\mu+\hat\nu+\hat\rho,m}]\\
&+ \rho_4\sum\nolimits_{\sigma\neq\rho\neq\nu(\neq\mu)}[\delta_{n+\hat\mu+\hat\nu+\hat\rho+\hat\sigma,m}-\delta_{n-\hat\mu+\hat\nu+\hat\rho+\hat\sigma,m}]\\
a^2\triangle^{\rm bri}(n,m) &= \lambda_0\,\delta_{n,m}+\lambda_1\sum\nolimits_{\mu}\delta_{n+\hat\mu,m}+\lambda_2\sum\nolimits_{\nu\neq\mu}\delta_{n+\hat\mu+\hat\nu,m}\\
&+ \lambda_3\sum\nolimits_{\rho\neq\nu\neq\mu}\delta_{n+\hat\mu+\hat\nu+\hat\rho,m}+\lambda_4\sum\nolimits_{\sigma\neq\rho\neq\nu\neq\mu}\delta_{n+\hat\mu+\hat\nu+\hat\rho+\hat\sigma,m}
\end{aligned}$$

8 terms at 1-hop: 1!= 1 paths, so nothing to be done

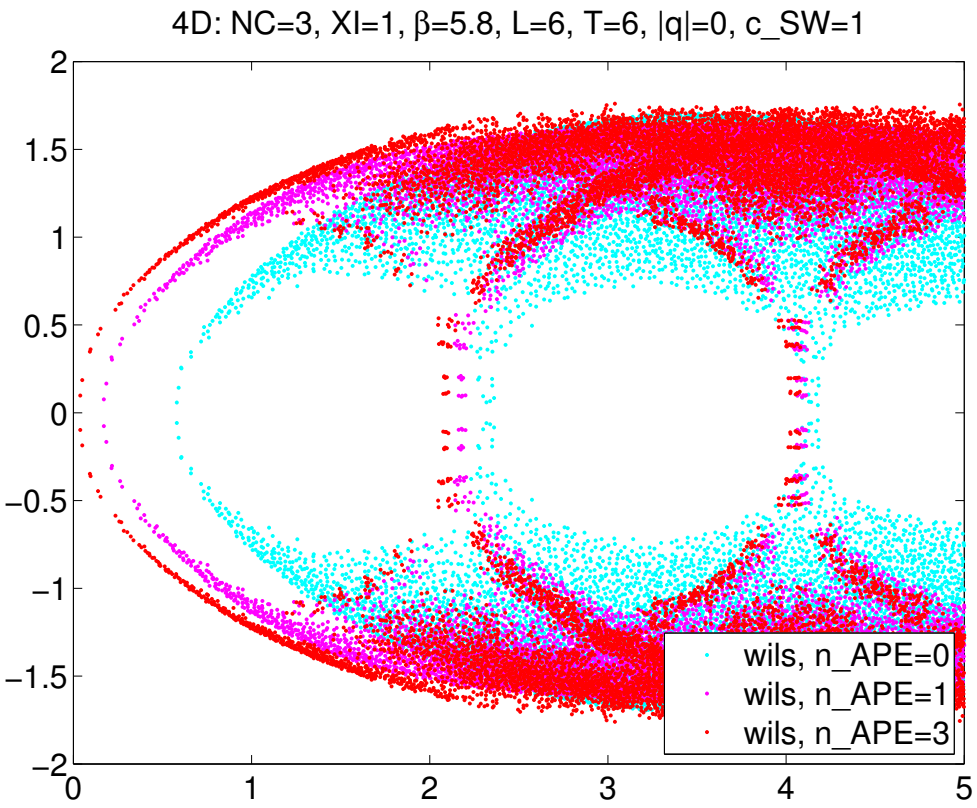
24 terms at 2-hop: 2!= 2 paths, so $\delta_{n+\hat\mu+\hat\nu,m}$ gets factor $\frac{1}{2}[V_\mu(n)V_\nu(n+\hat\mu)+\mu\leftrightarrow\nu]$

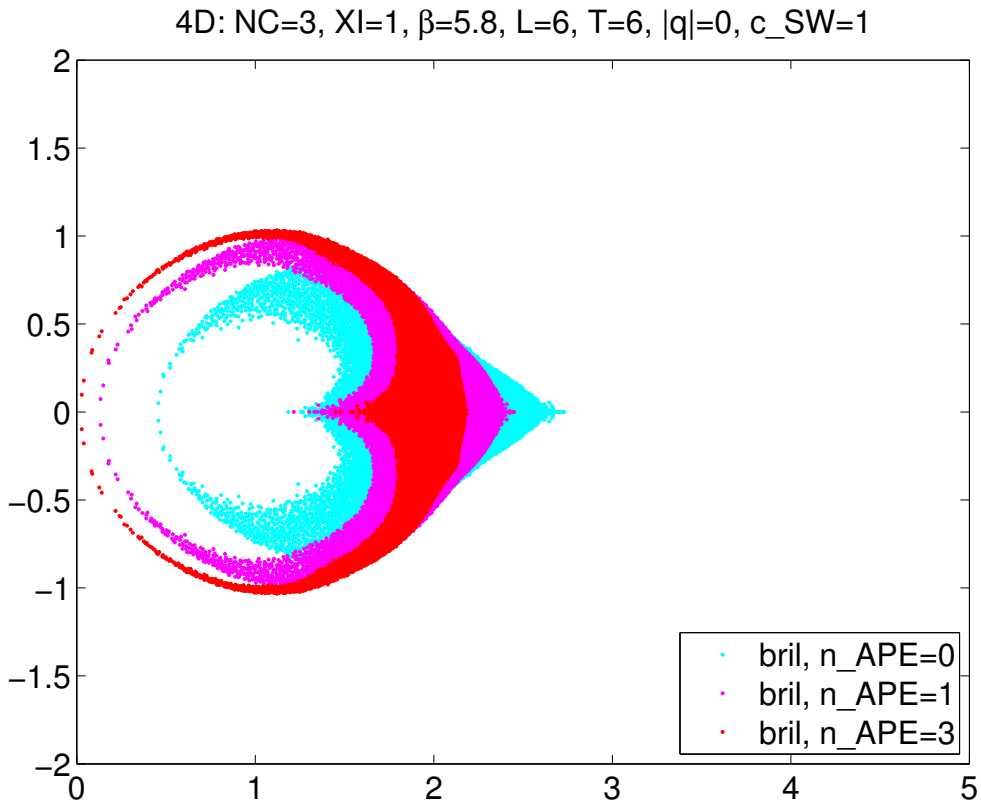
32 terms at 3-hop: 3!= 6 paths, to be gauge-averaged

16 terms at 4-hop: 4!=24 paths, to be gauge-averaged

(paths must be averaged in order to maintain γ_5-hermiticity; precompute $\mathtt{V}\mapsto\mathtt{W}$)

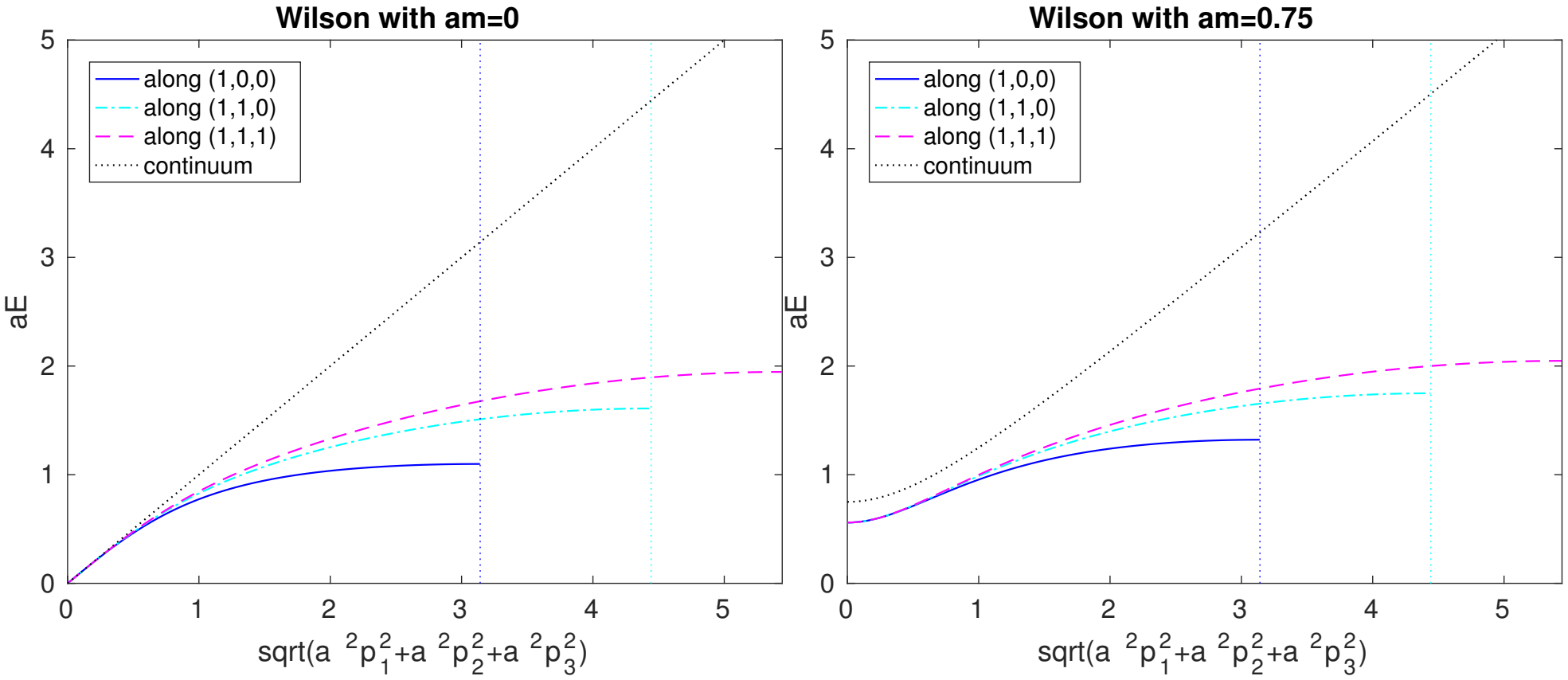
- **Wilson versus Brillouin eigenvalue spectrum**





- fat-clover strategy drastically reduces $am_{\rm res}$ or $\frac{1}{2\kappa_{\rm crit}} - 4$, both for W and B
- clover-term in Brillouin leads to "fuzzying" of 15 doubler branches at $\mathrm{Re}(z) = 2$
- for details, see S. Dürr, G. Koutsou, arXiv:1701.00726

- **Dispersion relation for Wilson operator**

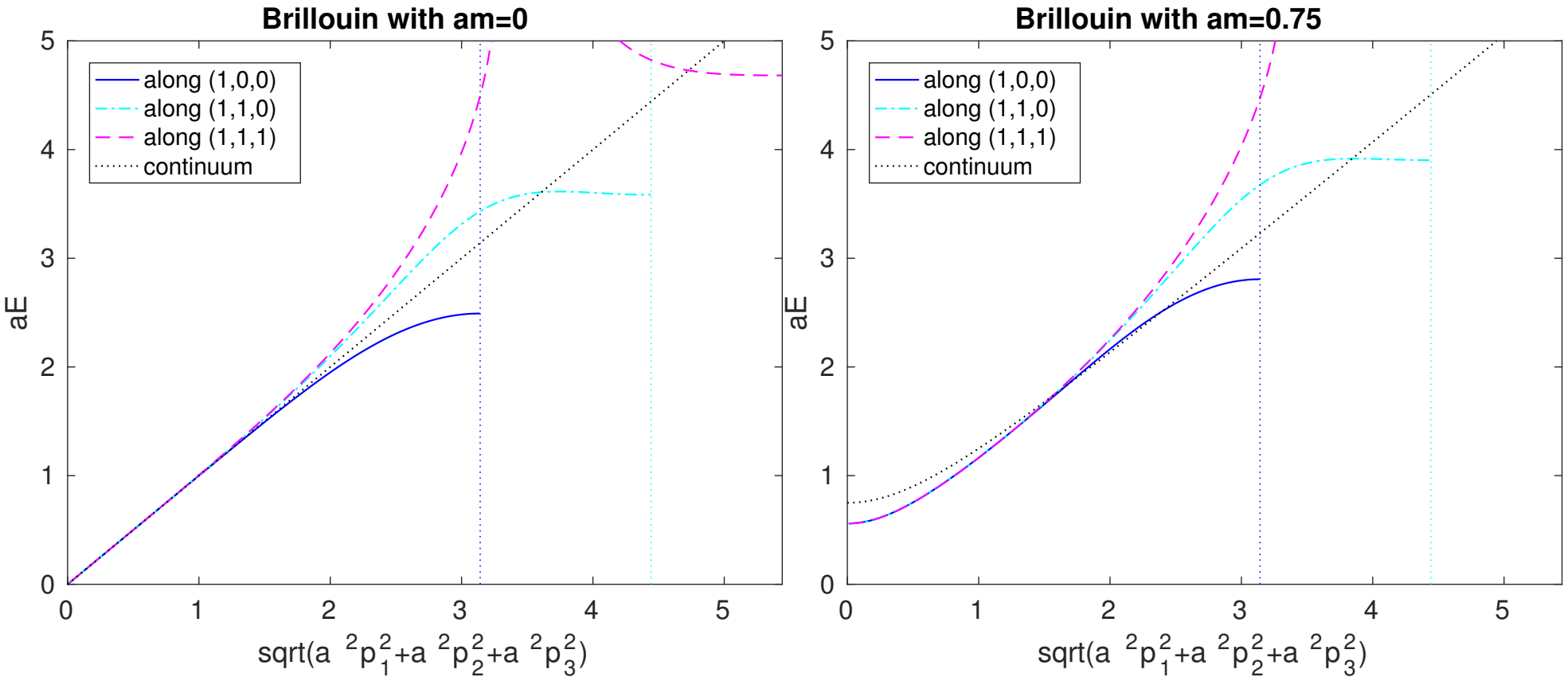


⊖ strong deviation from continuum for any $a|\mathbf{p}| > 1$

⊖ strong rotational symmetry breaking for any $a|\mathbf{p}| > 1$

⊖ strong effect of $am \not\ll 1$, even at $\mathbf{p} = \mathbf{0}$

- **Dispersion relation for Brillouin operator**

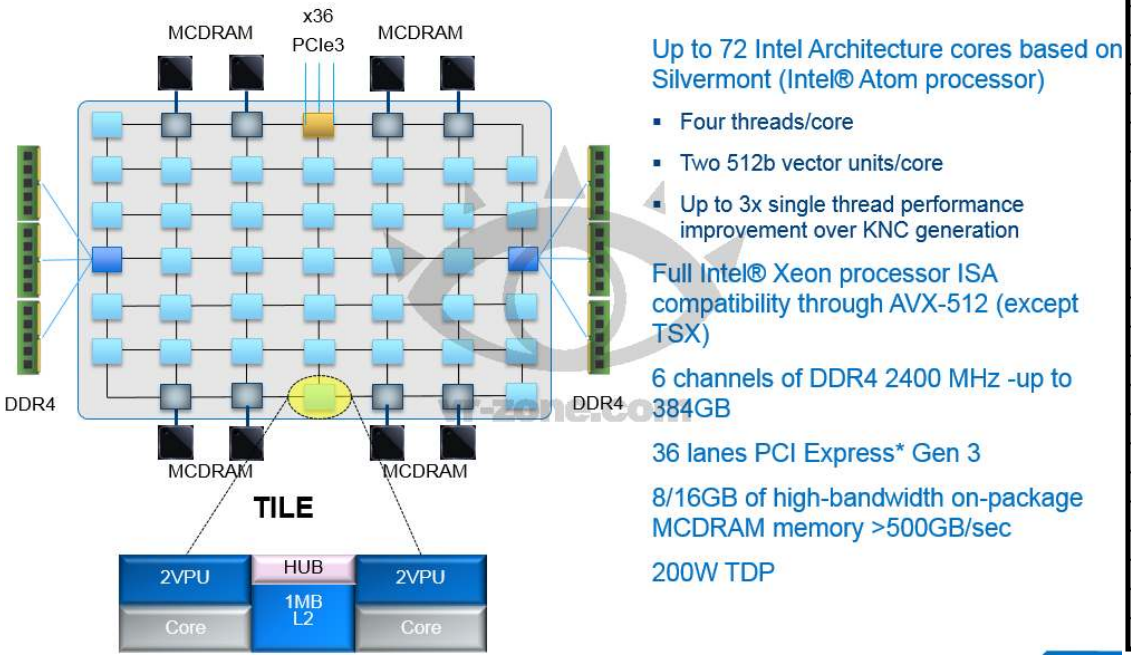


⊕ mild deviation from continuum up to $a|\mathbf{p}| \simeq 2$

⊕ mild rotational symmetry breaking up to $a|\mathbf{p}| \simeq 2$

⊖ strong effect of $am \not\ll 1$, especially at $\mathbf{p} = \mathbf{0}$

Intel KNL: architecture overview



	Knights Landing (Estimate)	Knights Corner	Haswell Client	Ivy Bridge Server	Unit
Process Node	14	22	22	22	nm
Base Frequency	1.4	1.238	3.6	2.7	GHz
TDP	300	300	82	130	W
Cores	72	61	4	12	
DP FLOP/core	32	16	16	8	FLOP/cycle
DP GFLOPs/core	44.8	19.8	57.6	21.6	GFLOP/s
L1D Capacity	32	32	32	32	KB
L1D Read	128	64	64	32	B/cycle
L1D Read BW	179.2	79.2	230.4	86.4	GB/s
L2 Capacity/core	256	512	256	256	KB
L2 Read/core	64	64	64	32	B/cycle
L2 Read BW/core	89.6	79.2	230.4	86.4	GB/s
L3 Capacity/core	2	NA	2	2.5	MB
L3 Read/core	32	NA	32	32	B/cycle
L3 Read BW/core	44.8	NA	115.2	86.4	GB/s
Total DP GFLOP/s	3225.6	1208.288	230.4	259.2	GFLOP/s
Total L1 Read BW	12.902	4.833	0.922	1.037	TB/s
Total L2 Read BW	6.451	4.833	0.922	1.037	TB/s
Total L3 Read BW	3.226	NA	0.128	0.384	TB/s
Total LLC Capacity	144	30.5	8	30	MB

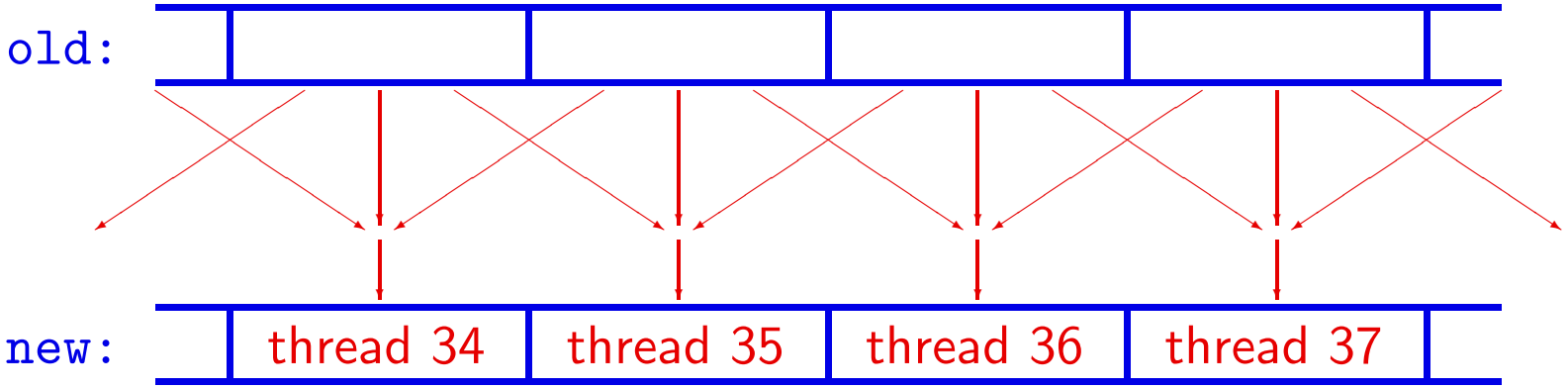
- hard to make near-perfect use of SIMD capability, hence expect $\ll 3\,\mathrm{T\,flop/s}$
- `numactl --membind 1 ./test_knl < test_knl.inp` to access MCDRAM
- `SCHEDULE(static)` and shared-variable initialization with same `NUM_THREADS`

Code suite guidelines

N_c colors and N lattice sites imply sparse $N_c 4N \times N_c 4N$ matrix $[4 + m_0 = (2\kappa)^{-1}]$:

$$D_{\mathrm{W}}(x,y) = \frac{1}{2}\sum_{\mu}\left\{(\gamma_\mu - I)U_\mu(x)\delta_{x+\hat{\mu},y} - (\gamma_\mu + I)U^\dagger_\mu(x-\hat{\mu})\delta_{x-\hat{\mu},y}\right\} + \frac{1}{2\kappa}\delta_{x,y}$$

- **How to avoid write-collisions among threads**



OMP threads over out-vector space-time index (collect in private variable `site`)

- **Fortran loop/slot ordering ("column major memory layout")**

Declare `old/new` vector as complex array of size `(1:Nc,1:4,1:Nv,1:Nx*Ny*Nz*Nt)`:

* color (`1:Nc`) innermost/first ⟵ unroll
* spinor (`1:4`) next ⟵ unroll
* rhs-idx (`1:Nv`) next ⟵ SIMD via OMP pragma
* site (`1:Nx*Ny*Nz*Nt`) outermost/last ⟵ distribute among OMP threads

- **Overall smearing strategy**

Use same kind of smearing in covariant Nabla and Laplacian and clover term.

Introduce smeared gauge field `V(Nc,Nc,4,Nx,Ny,Nz,Nt)` besides original U_ρ, and evaluate Wilson/Brillouin operator on V_ρ.

For Brillouin operator precompute off-axis averages of V_ρ, and assemble the 40 relevant elements per site in auxiliary object `W(Nc,Nc,40,Nx,Ny,Nz,Nt)`.

For $c_{\rm SW} \neq 0$ precompute $F_{\mu\nu}$ from V_ρ, i.e. create `F(Nc,Nc,6,Nx,Ny,Nz,Nt)`.

- **Mini-summary**

Avoid derived data types, i.e. stick to multidimensional complex arrays.
Leave index computations to compiler, except for site index `n` $\in$ `[1:Nx*Ny*Nz*Nt]`.

`app_wils` does `old(Nc,4,Nv,Nx*Ny*Nz*Nt)` $\mapsto$ `new(Nc,4,Nv,Nx*Ny*Nz*Nt)` using `V(Nc,Nc,4,Nx,Ny,Nz,Nt)` and possibly `F(Nc,Nc,6,Nx,Ny,Nz,Nt)` as argument.

`app_bril` works identically, except that `V` is traded for `W(Nc,Nc,40,Nx,Ny,Nz,Nt)`.

- **Compilation (ifort $\geq$ 17.2)**

```
ifort [...] -qopenmp -O2 -xmic-avx512 -align array64byte -o test_knl test_knl.f90
ifort [...] -qopenmp -O3 -xcore-avx2                     -o test_knl test_knl.f90
```

• Wilson kernel without shrink/expand trick

```
PARAMETERS: Nx,Ny,Nz,Nt, Nc,Nv, sp,dp, i_sp,i_dp !!! known at compile time

!$OMP PARALLEL DO DEFAULT(private) FIRSTPRIVATE(mass) SHARED(old,new,V) SCHEDULE(static)
do l=1,Nt; l_plu=modulo(l,Nt)+1; l_min=modulo(l-2,Nt)+1
do k=1,Nz; k_plu=modulo(k,Nz)+1; k_min=modulo(k-2,Nz)+1
do j=1,Ny; j_plu=modulo(j,Ny)+1; j_min=modulo(j-2,Ny)+1
do i=1,Nx; i_plu=modulo(i,Nx)+1; i_min=modulo(i-2,Nx)+1
   n=(((l-1)*Nz+(k-1))*Ny+(j-1))*Nx+i
   !!! direction 0 gets factor -1/2*(-8I)=4*I and mass
   site(:,:,:)=(4.0+mass)*old(:,:,:,n)
   !!! direction -4 gets factor 1/2*(-I-gamma4)
   tmp=0.5*transpose(conjg(V(:,:,4,i,j,k,l_min))); nsh=n+(l_min-l)*Nz*Ny*Nx; ...
   !!! direction -3 gets factor 1/2*(-I-gamma3)
   tmp=0.5*transpose(conjg(V(:,:,3,i,j,k_min,l))); nsh=n+(k_min-k)*Ny*Nx   ; ...
   !!! direction -2 gets factor 1/2*(-I-gamma2)
   tmp=0.5*transpose(conjg(V(:,:,2,i,j_min,k,l))); nsh=n+(j_min-j)*Nx      ; ...
   !!! direction -1 gets factor 1/2*(-I-gamma1)
   tmp=0.5*transpose(conjg(V(:,:,1,i_min,j,k,l))); nsh=n+(i_min-i)         ; ...
   !!! direction +1 gets factor 1/2*(-I+gamma1)
   tmp=0.5*               V(:,:,1,i    ,j,k,l)  ; nsh=n+(i_plu-i)         ; ...
   !!! direction +2 gets factor 1/2*(-I+gamma2)
   tmp=0.5*               V(:,:,2,i,j    ,k,l)  ; nsh=n+(j_plu-j)*Nx      ; ...
   !!! direction +3 gets factor 1/2*(-I+gamma3)
   tmp=0.5*               V(:,:,3,i,j,k    ,l)  ; nsh=n+(k_plu-k)*Ny*Nx   ; ...
   !!! direction +4 gets factor 1/2*(-I+gamma4)
   tmp=0.5*               V(:,:,4,i,j,k,l    )  ; nsh=n+(l_plu-l)*Nz*Ny*Nx; ...
end do ! i=1,Nx
end do ! j=1,Ny
end do ! k=1,Nz
end do ! l=1,Nt
!$OMP END PARALLEL DO
```

Each block ... is replaced by $\psi(.) \mapsto V_\mu(.)\psi(.)\gamma_\mu^{\rm tr}$ with $\psi \in \mathbf{C}^{3\times 4}$, e.g. for $\mu = -4, -3$:

```
!!! direction -4 gets factor 1/2*(-I-gamma4)
tmp=0.5*transpose(conjg(V(:,:,4,i,j,k,l_min))); nsh=n+(l_min-l)*Nz*Ny*Nx
!$OMP SIMD PRIVATE(full)
do idx=1,Nv
   forall(col=1:Nc,spi=1:4) full(col,spi)=sum(tmp(col,:)*old(:,spi,idx,nsh))
   site(:,1,idx)=site(:,1,idx)-full(:,1)-     full(:,3) !!! transpose(gamma4)=     0     0     1     0
   site(:,2,idx)=site(:,2,idx)-full(:,2)-     full(:,4) !!!                       0     0     0     1
   site(:,3,idx)=site(:,3,idx)-full(:,3)-     full(:,1) !!!                       1     0     0     0
   site(:,4,idx)=site(:,4,idx)-full(:,4)-     full(:,2) !!!                       0     1     0     0
end do
!$OMP END SIMD

!!! direction -3 gets factor 1/2*(-I-gamma3)
tmp=0.5*transpose(conjg(V(:,:,3,i,j,k_min,l))); nsh=n+(k_min-k)*Ny*Nx
!$OMP SIMD PRIVATE(full)
do idx=1,Nv
   forall(col=1:Nc,spi=1:4) full(col,spi)=sum(tmp(col,:)*old(:,spi,idx,nsh))
   site(:,1,idx)=site(:,1,idx)-full(:,1)+i_sp*full(:,3) !!! transpose(gamma3)=     0     0     i     0
   site(:,2,idx)=site(:,2,idx)-full(:,2)-i_sp*full(:,4) !!!                       0     0     0    -i
   site(:,3,idx)=site(:,3,idx)-full(:,3)-i_sp*full(:,1) !!!                      -i     0     0     0
   site(:,4,idx)=site(:,4,idx)-full(:,4)+i_sp*full(:,2) !!!                       0     i     0     0
end do
!$OMP END SIMD
```

- OMP-pragmas for shared-memory parallelization over space-time indices
- OMP-pragmas for SIMD-pipelining over Nv right-hand-sides
- color-spinor muliplication in `forall(col=1:Nc,spi=1:4)` means unrolling
- implied-do in `site(:,1:4,idx)=...` means unrolling

- **Wilson kernel with shrink/expand trick**

Operator $\frac{1}{2}(1 \mp \gamma_\mu)$ is a projector $\forall \mu$ (only 2 eigenvectors with non-zero eigenvalue):

```
! [V,D]=eig(gamma4trsp-eye(4)) ==> V*sqrt(2)=( +1 00 +1 00 ) D=( -2          )
!                                            ( 00 +1 00 +1 )   (    -2       )
!                                            ( -1 00 +1 00 )   (       00    )
!                                            ( 00 -1 00 +1 )   (          00 )
! [V,D]=eig(gamma4trsp+eye(4)) ==> V*sqrt(2)=( +1 00 +1 00 ) D=( 00          )
!                                            ( 00 +1 00 +1 )   (    00       )
!                                            ( -1 00 +1 00 )   (       +2    )
!                                            ( 00 -1 00 +1 )   (          +2 )
```

```
!!! direction -4 gets factor 1/2*(-I-gamma4)
tmp_min=0.5*transpose(conjg(V(:,:,4,i,j,k,l_min))); nsh_min=n+(l_min-l)*Nz*Ny*Nx
!!! direction +4 gets factor 1/2*(-I+gamma4)
tmp_plu=0.5*                V(:,:,4,i,j,k,l    )  ; nsh_plu=n+(l_plu-l)*Nz*Ny*Nx
!$OMP SIMD PRIVATE(red_min,red_plu,half_min,half_plu)
do idx=1,Nv
   red_min(:,1)=old(:,1,idx,nsh_min)+     old(:,3,idx,nsh_min)
   red_min(:,2)=old(:,2,idx,nsh_min)+     old(:,4,idx,nsh_min)
   red_plu(:,1)=old(:,1,idx,nsh_plu)-     old(:,3,idx,nsh_plu)
   red_plu(:,2)=old(:,2,idx,nsh_plu)-     old(:,4,idx,nsh_plu)
   forall(col=1:Nc,spi=1:2) half_min(col,spi)=sum(tmp_min(col,:)*red_min(:,spi))
   forall(col=1:Nc,spi=1:2) half_plu(col,spi)=sum(tmp_plu(col,:)*red_plu(:,spi))
   site(:,1,idx)=site(:,1,idx)-  half_min(:,1)-  half_plu(:,1)
   site(:,2,idx)=site(:,2,idx)-  half_min(:,2)-  half_plu(:,2)
   site(:,3,idx)=site(:,3,idx)-  half_min(:,1)+  half_plu(:,1)
   site(:,4,idx)=site(:,4,idx)-  half_min(:,2)+  half_plu(:,2)
end do
!$OMP END SIMD
```

Brillouin kernel details

```
PARAMETERS: Nx,Ny,Nz,Nt, Nc,Nv, sp,dp, i_sp,i_dp !!! known at compile time

mask=[0.0_sp,1.0_sp,4.0_sp,16.0_sp,64.0_sp]/432.0_sp !!! note: declared as real(kind=sp),dimension(0:4)
!$OMP PARALLEL DO COLLAPSE(4) DEFAULT(private) FIRSTPRIVATE(mass,mask) SHARED(old,new,W) SCHEDULE(static)
do l=1,Nt
do k=1,Nz
do j=1,Ny
do i=1,Nx
   n=(((l-1)*Nz+(k-1))*Ny+(j-1))*Nx+i
   site(:,:,:)=cmplx(0.0,kind=sp) !!! note: site is Nc*4*Nv
   !!! visit all 81 sites within hypercube (distances 0 to 4 in taxi-driver metric)
   dir=0
   do go_l=-1,1; lsh=modulo(l+go_l-1,Nt)+1
   do go_k=-1,1; ksh=modulo(k+go_k-1,Nz)+1
   do go_j=-1,1; jsh=modulo(j+go_j-1,Ny)+1
   do go_i=-1,1; ish=modulo(i+go_i-1,Nx)+1
      dir=dir+1 !!! note: dir=(go_l+1)*27+(go_k+1)*9+(go_j+1)*3+go_i+2
      ...
   end do ! go_i=-1,1
   end do ! go_j=-1,1
   end do ! go_k=-1,1
   end do ! go_l=-1,1
   !!! plug everything into new vector
   do idx=1,Nv
      new(:,:,idx,n)=site(:,:,idx)
   end do ! idx=1,Nv
end do ! i=1,Nx
end do ! j=1,Ny
end do ! k=1,Nz
end do ! l=1,Nt
!$OMP END PARALLEL DO
```

The block ... is replaced by (chiral representation of γ-matrices):

```
select case(dir)
   case(01:40); tmp=W(:,:,dir,i,j,k,l)
   case(   41); tmp=color_eye() !!! note: yields Nc*Nc identity matrix
   case(42:81); tmp=conjg(transpose(W(:,:,82-dir,ish,jsh,ksh,lsh)))
end select
absgo_ijkl=go_i*go_i+go_j*go_j+go_k*go_k+go_l*go_l
fac=0.125_sp/2**absgo_ijkl               !!! note: for 1/2 times Brillouin Laplacian
if (absgo_ijkl.eq.0) fac=fac-2.0_sp-mass !!! note: correction for go_i=go_j=go_k=go_l=0
fac_i=go_i*mask(absgo_ijkl)              !!! note: for isotropic derivative in x-direction
fac_j=go_j*mask(absgo_ijkl)              !!! note: for isotropic derivative in y-direction
fac_k=go_k*mask(absgo_ijkl)              !!! note: for isotropic derivative in z-direction
fac_l=go_l*mask(absgo_ijkl)              !!! note: for isotropic derivative in t-direction
nsh=(((lsh-1)*Nz+(ksh-1))*Ny+(jsh-1))*Nx+ish
!$OMP SIMD PRIVATE(full)
do idx=1,Nv
    forall(col=1:Nc,spi=1:4) full(col,spi)=sum(tmp(col,:)*old(:,spi,idx,nsh))
    site(:,1,idx)=site(:,1,idx)-fac*full(:,1)+cmplx(-fac_j,-fac_i)*full(:,4)+cmplx(+fac_l,-fac_k)*full(:,3)
    site(:,2,idx)=site(:,2,idx)-fac*full(:,2)+cmplx(+fac_j,-fac_i)*full(:,3)+cmplx(+fac_l,+fac_k)*full(:,4)
    site(:,3,idx)=site(:,3,idx)-fac*full(:,3)+cmplx(+fac_j,+fac_i)*full(:,2)+cmplx(+fac_l,+fac_k)*full(:,1)
    site(:,4,idx)=site(:,4,idx)-fac*full(:,4)+cmplx(-fac_j,+fac_i)*full(:,1)+cmplx(+fac_l,-fac_k)*full(:,2)
end do ! idx=1,Nv
!$OMP END SIMD
```

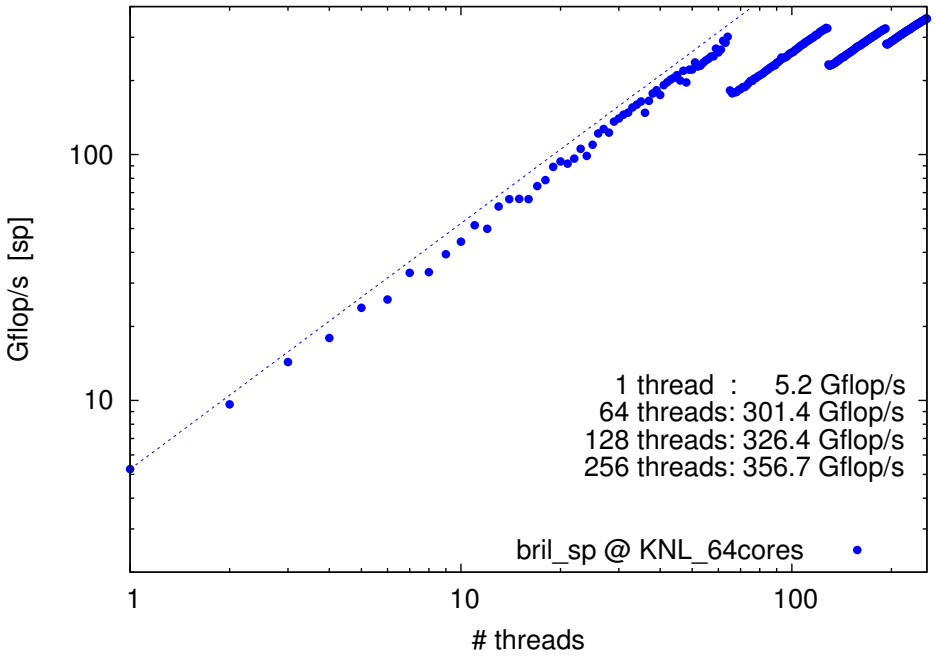
- OMP-pragmas for shared-memory parallelization over space-time indices
- OMP-pragmas for SIMD-pipelining over Nv right-hand-sides
- color-spinor muliplication in `forall(col=1:Nc,spi=1:4)` means unrolling
- implied-do in `site(:,1:4,idx)=...` means unrolling

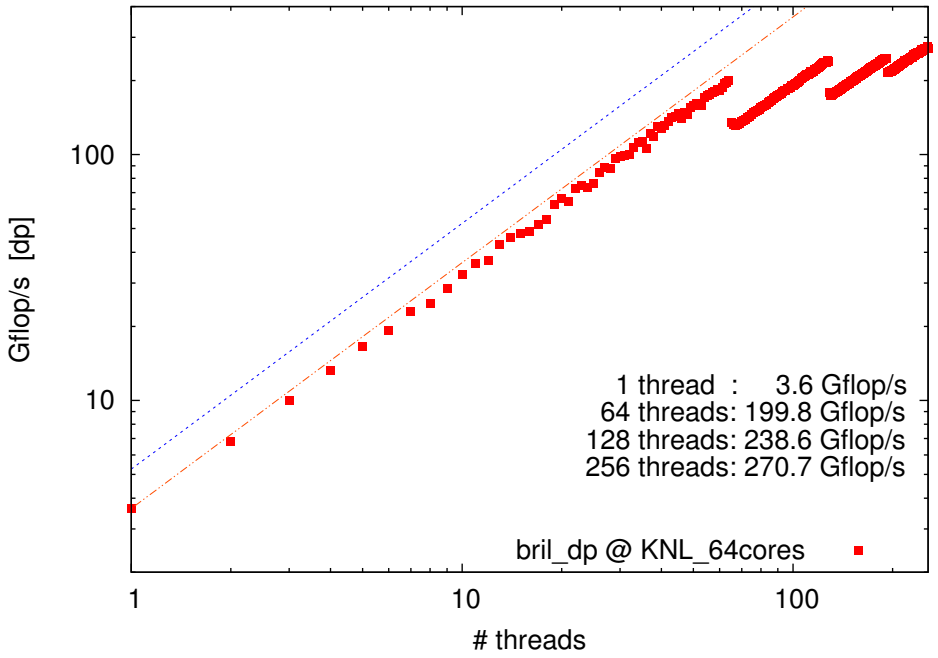
Brillouin operator timings

- **Brillouin on KNL (64 cores)**

Volume $32^3 \cdot 64$ fixed, $N_c = 3$ fixed, $N_v = 4N_c$ fixed, all performances in Gflop/s.

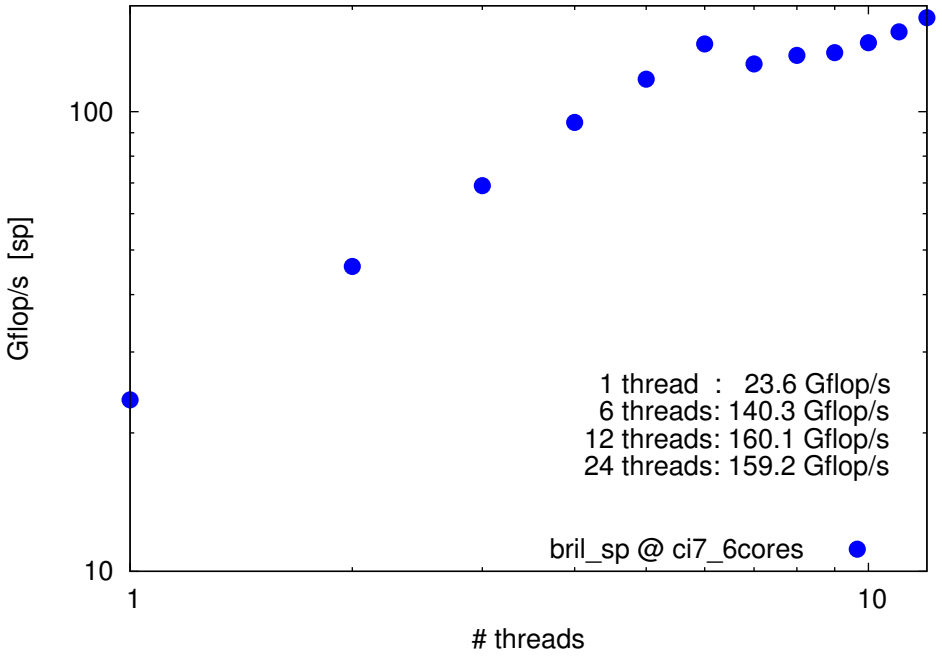
	$N_{\rm thr} = 1$	$N_{\rm thr} = 64$	$N_{\rm thr} = 128$	$N_{\rm thr} = 256$	$N_{\rm thr} = 512$
sp	5.2	301.4	326.4	356.7	357.1
dp	3.6	199.8	238.6	270.7	271.8

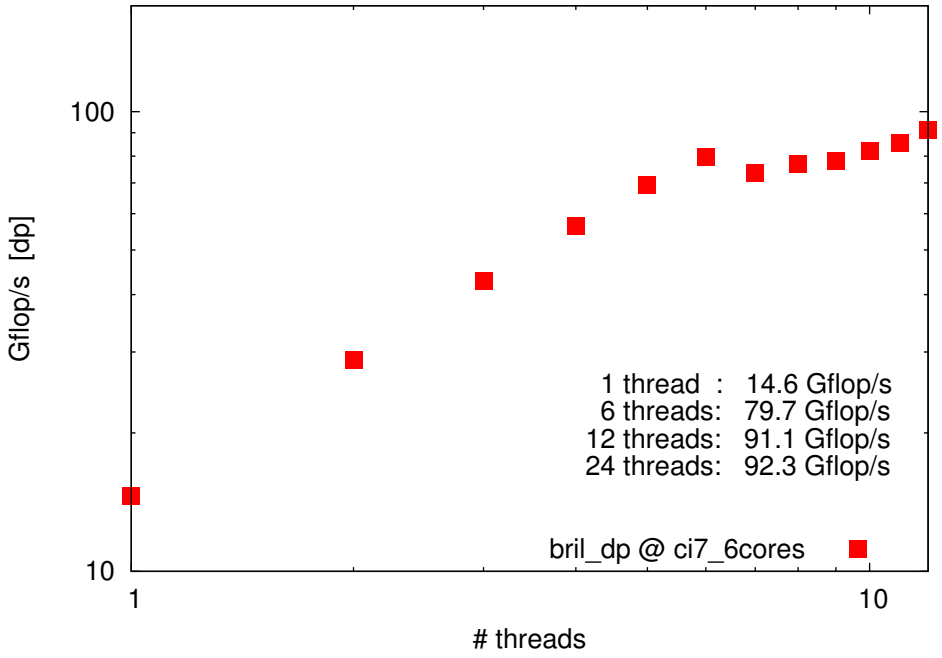




- **Brillouin on Core i7 (6 cores, same parameters)**

	$N_{\rm thr}=1$	$N_{\rm thr}=2$	$N_{\rm thr}=4$	$N_{\rm thr}=6$	$N_{\rm thr}=12$	$N_{\rm thr}=24$
sp	23.6	46.0	94.8	140.3	160.1	159.2
dp	14.6	28.9	56.3	79.7	91.1	92.3





- **Brillouin performance ratios**

KNL 64: 5.2/2.6 Tflop/s peak [sp/dp]: 357/272 Gflop/s mean 6.8 /10.4% [sp/dp]
Broadw: 690/345 Gflop/s peak [sp/dp]: 160/ 92 Gflop/s mean 23.2/26.7% [sp/dp]

- **Dependence on `Nx*Ny*Nz*Nt` (on KNL)**

$N_c = 3$ fixed, $N_v = 4N_c$ fixed, $T = 2L$ scales, with sp-performance in Gflop/s:

	$L = 12$	$L = 16$	$L = 20$	$L = 24$	$L = 32$
128 threads	335	328	339	340	321
256 threads	350	350	360	359	357

- **Dependence on `Nv` (on KNL)**

Volume $24^3 \cdot 48$ fixed, $N_c = 3$ fixed, with sp-performance in Gflop/s:

	$N_v = 2N_c$	$N_v = 4N_c$	$N_v = 8N_c$	$N_v = 32N_c$
128 threads	194	341	526	533
256 threads	209	360	558	588

- **Dependence on `Nc` (on KNL)**

Volume $24^3 \cdot 48$ fixed, $N_v = 4N_c$ scales, with sp-performance in Gflop/s:

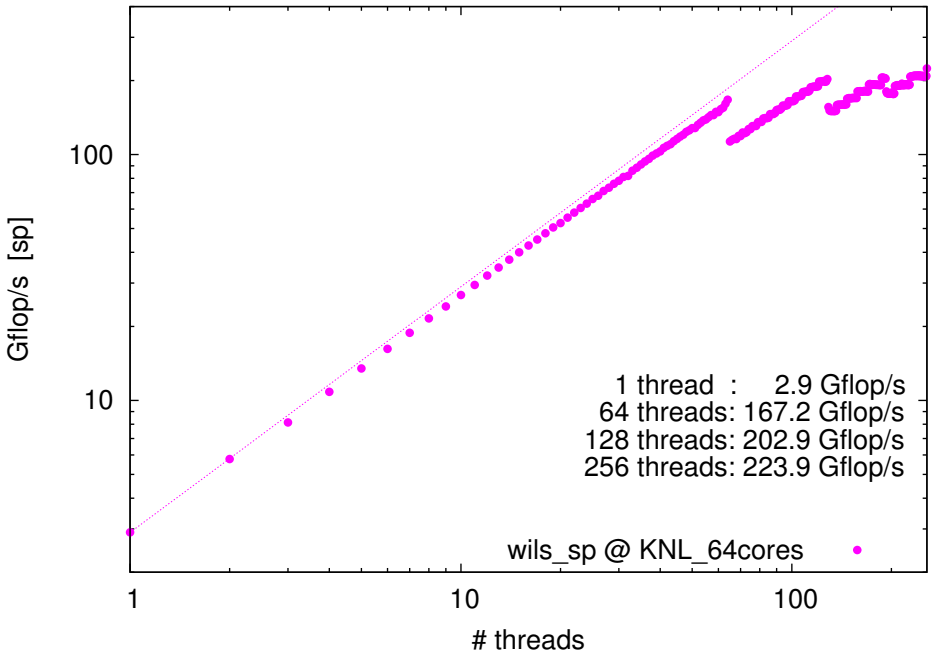
	$N_c = 2$	$N_c = 3$	$N_c = 4$	$N_c = 5$	$N_c = 6$
128 threads	344	341	552	444	634
256 threads	356	360	641	488	779

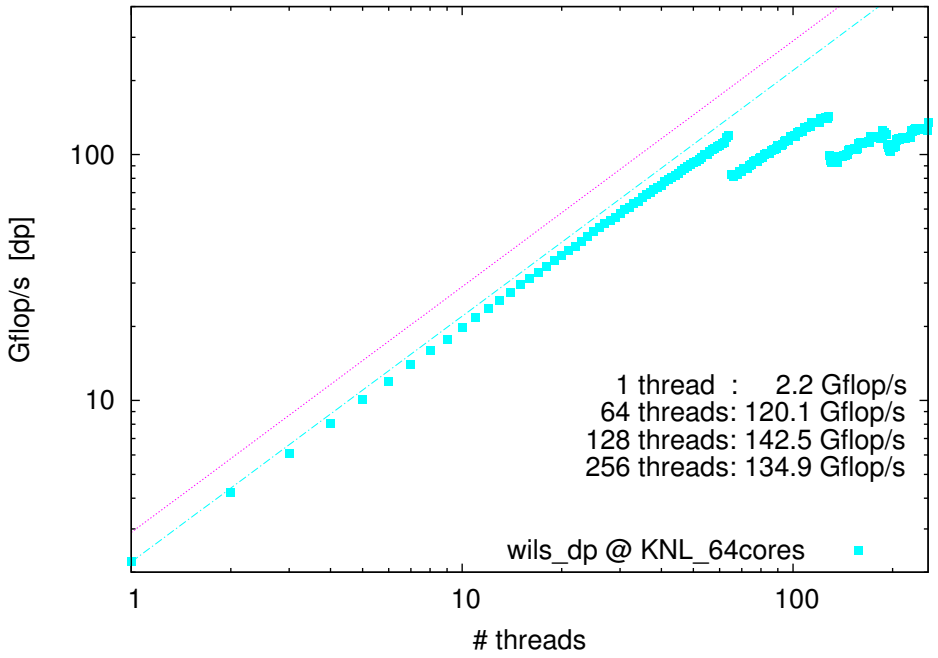
Wilson operator timings

- **Wilson on KNL (64 cores)**

Volume $32^3 \cdot 64$ fixed, $N_c = 3$ fixed, $N_v = 4N_c$ fixed, all performances in Gflop/s.

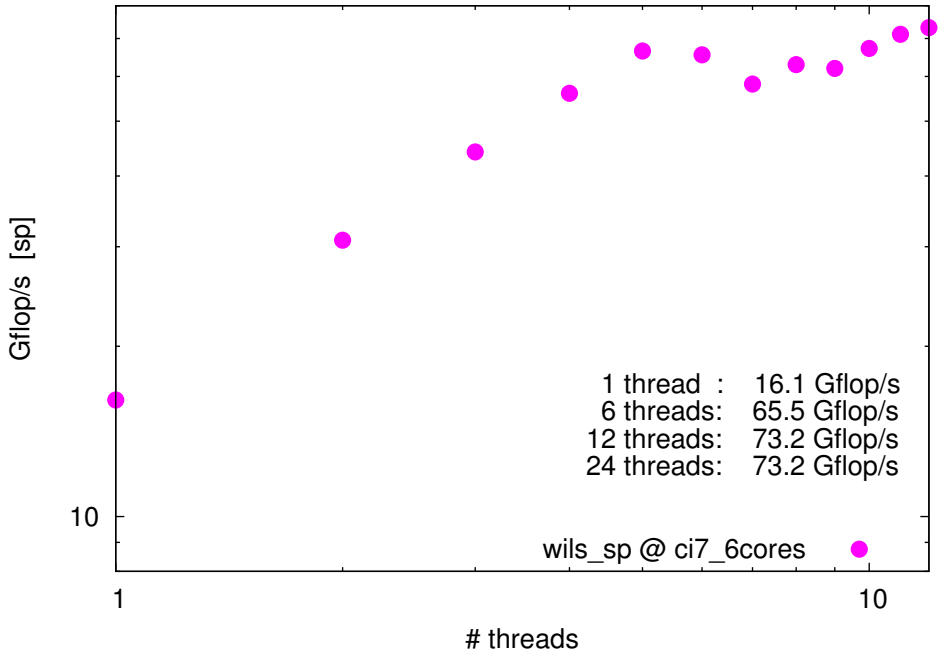
	$N_{\rm thr} = 1$	$N_{\rm thr} = 64$	$N_{\rm thr} = 128$	$N_{\rm thr} = 256$	$N_{\rm thr} = 512$
sp	2.9	167.2	202.9	223.9	225.2
dp	2.2	120.1	142.5	134.9	134.2

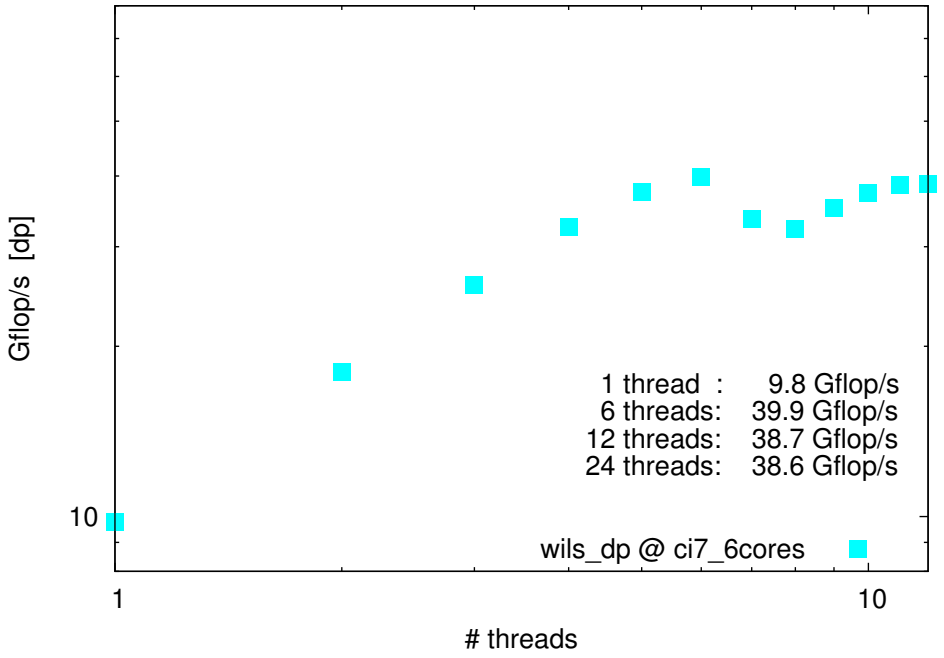




- **Wilson on Core i7 (6 cores, same parameters)**

	$N_{\rm thr}=1$	$N_{\rm thr}=2$	$N_{\rm thr}=4$	$N_{\rm thr}=6$	$N_{\rm thr}=12$	$N_{\rm thr}=24$
sp	16.1	30.8	56.0	65.5	73.2	73.2
dp	9.8	18.0	32.5	39.9	38.7	38.6

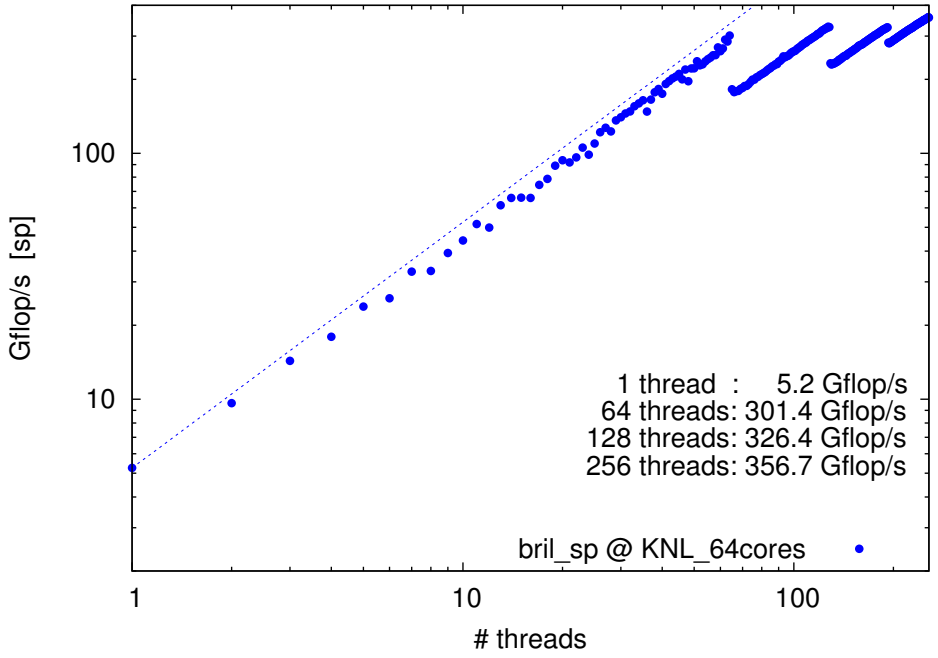


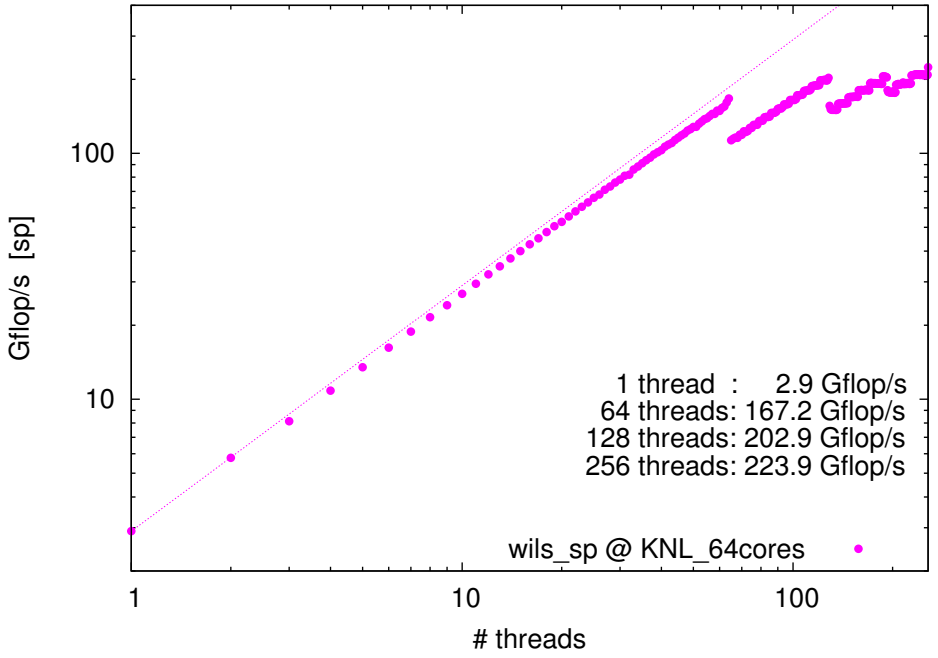


- **Wilson performance ratios**

KNL64 : 5.2/2.6 Tflop/s peak [sp/dp]: 225/135 Gflop/s mean 4.3 / 5.2% [sp/dp]
Broadw: 690/345 Gflop/s peak [sp/dp]: 73 / 40 Gflop/s mean 10.6/11.6% [sp/dp]

Summary





- KNL: 360 Gflop/s [sp] for $N_c = 3, N_v = 4N_c$ and $N_{\rm thr} = 256$ is $\sim 6.8\%$ of peak
- core : 160 Gflop/s [sp] for $N_c = 3, N_v = 4N_c$ and $N_{\rm thr} = 012$ is $\sim 23.2\%$ of peak

- SIMD over number of right-hand-sides, explicit unrolling of color/spinor
- OpenMP parallelization over space-time manifold, avoids write(read) collisions
- no MPI parallelization (yet?)

O(1000) lines of Fortran 2008 [bril,wils,clover,scalp,CG] code: durr(AT)itp.unibe.ch

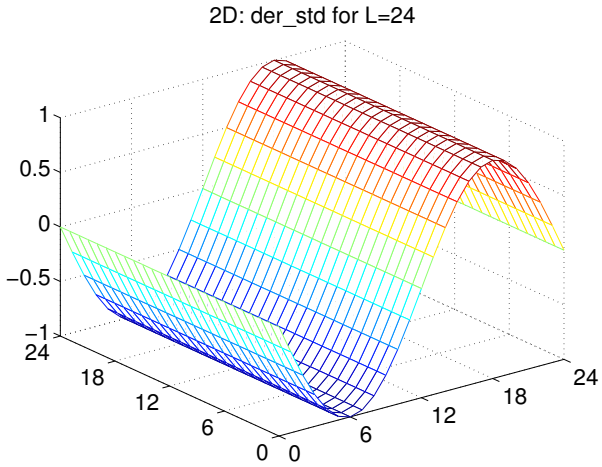
BACKUP SLIDES

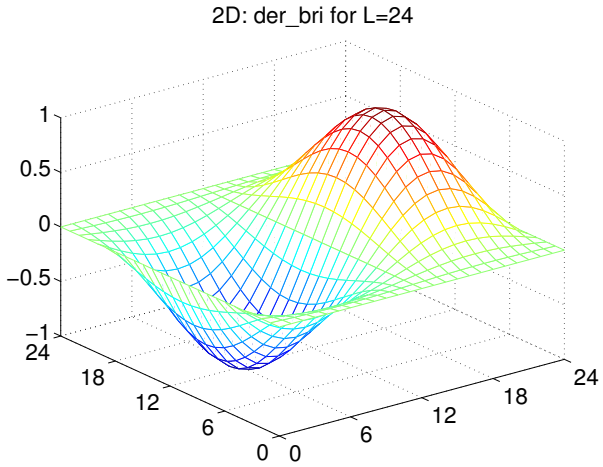
- **3 options for (covariant) Nabla**

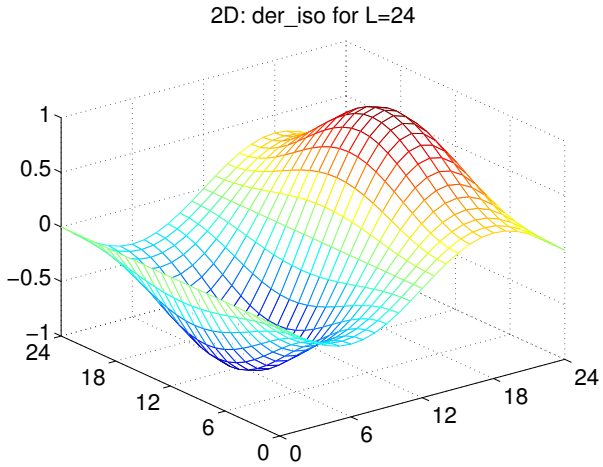
Standard Derivative: $\hat{\nabla}_x = \mathrm{i}\sin(k_1)$

Brillouin Derivative: $\hat{\nabla}_x = \mathrm{i}\sin(k_1)[\cos(k_2)+1][\cos(k_3)+1][\cos(k_4)+1]/8$

Isotropic Derivative: $\hat{\nabla}_x = \mathrm{i}\sin(k_1)[\cos(k_2)+2][\cos(k_3)+2][\cos(k_4)+2]/27$







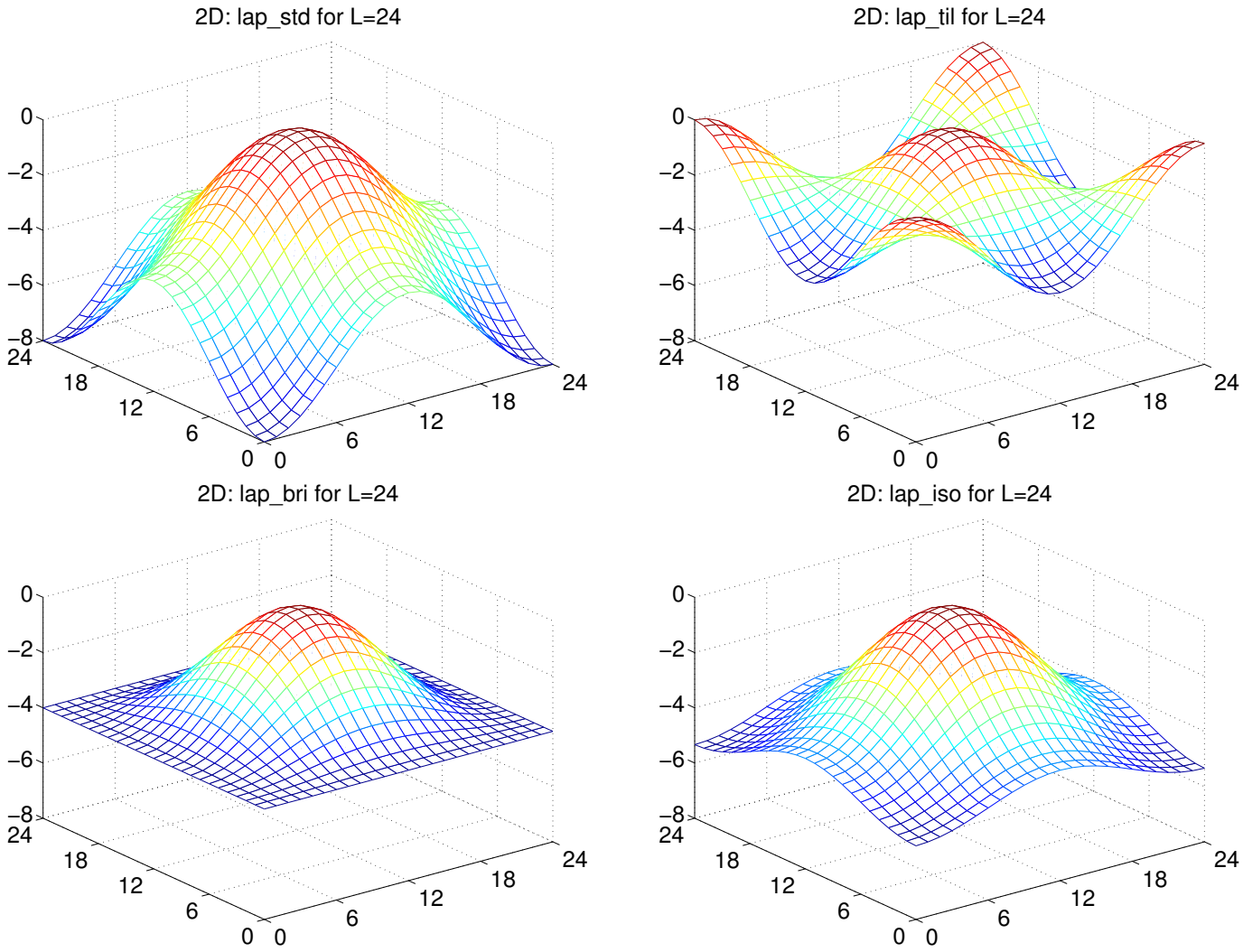
- **4 options for (covariant) Laplacian**

Standard Laplacian: $\hat{\triangle} = 2\cos(k_1) + 2\cos(k_2) + 2\cos(k_3) + 2\cos(k_4) - 8$

Tilted Laplacian: $\hat{\triangle} = 2\cos(k_1)\cos(k_2)\cos(k_3)\cos(k_4) - 2$

Brillouin Laplacian: $\hat{\triangle} = 4\cos^2(k_1/2)\cos^2(k_2/2)\cos^2(k_3/2)\cos^2(k_4/2) - 4$

Isotropic Laplacian: $\hat{\triangle} = [2c_1c_2c_3c_4 + 7c_1c_2c_3 + \ldots + 20c_1c_2 + \ldots + 25c_1 + \ldots - 250]/54$



- **Selection Procedure**

All 12 options live on $[-1:1]^4$ hypercube (81 sites in 4D, ultralocal). Test them systematically (eigenvalues, dispersion relation). Combination $(\triangle^{\mathrm{bri}}, \nabla^{\mathrm{iso}})$ prevails.

- **Wilson flop count**

For matrix-times-vector operation we must (per site):
(i) spin-project (from 4 to 2) for each direction $\longrightarrow 12 \cdot 8 = 96$ flops
(ii) SU(3)-multiply (spin-reduced) for each direction $\longrightarrow 6 \cdot 22 \cdot 8 = 1056$ flops
(iii) accumulate $8 + (4+m_0)1$ contributions to out-spinor $\longrightarrow 24 \cdot 9 = 216$ flops

All together 1368 flops per site.

- **Brillouin flop count**

For matrix-times-vector operation we must (per site):
(i) SU(3)-multiply (spin-full) for each direction $\longrightarrow 12 \cdot 22 \cdot 80 = 21120$ flops
(ii) multiply with fac_i/.../fac: $24 \cdot (54+54+54+54+81) = 7128$ flops
(iii) accumulate 81 contributions to out-spinor $\longrightarrow 24 \cdot 80 = 1920$ flops

All together 30168 flops per site.

- **Mini-summary**

The Brillouin-to-Wilson ratio of flops is **22.1**
The Brillouin-to-Wilson ratio of memory traffic is **8.9** (next slide)
The Brillouin-to-Wilson ratio of computational intensity is **2.5**.

- **Wilson memory traffic**

For matrix-times-vector operation we must (per site, with #rhs=1):
(i) read one sp-spinor for each direction $\longrightarrow 24 \cdot 9 = 216$ floats
(ii) read one sp-gauge matrix `V` per direction $\longrightarrow 18 \cdot 8 = 144$ floats
(iii) write one sp-spinor $\longrightarrow 24$ floats

All together 384 floats of traffic per site, i.e. 1536 bytes in sp (1.12 bytes/flop).
With 12 rhs this changes to 216+12+24=252 floats from/to memory per site.

- **Brillouin memory traffic**

For matrix-times-vector operation we must (per site, with #rhs=1):
(i) read one sp-spinor for each direction $\longrightarrow 24 \cdot 81 = 1944$ floats
(ii) read one sp-gauge matrix `W` per direction $\longrightarrow 18 \cdot 80 = 1440$ floats
(iii) write one sp-spinor $\longrightarrow 24$ floats

All together 3408 floats of traffic per site, i.e. 13632 bytes in sp (0.45 bytes/flop).
With 12 rhs this changes to 1944+120+24=2088 floats from/to memory per site.

- **Mini-summary**

The Brillouin-to-Wilson ratio of memory traffic is 3408/384=**8.9** or 2088/252=**8.9**.
Worst case scenario assumed, i.e. everything to be read afresh, i.e. nothing in cache.