Intel KNL を使ってみて：格子 *QCD* での例

Issaku Kanamori (Hiroshima Univ.)

第 2 回 HPC-Phys 勉強会理研和光, 2018 年 12 月 1 日

Ref.: I.K. and H.Matsufuru, arXiv:1710.07226 (Lattice 2017), arXiv:1712.01505 (CANDAR'17),

arXiv:1811.00893 (ICCSA 2018)

本題に入る前に...

今回参加されてまだメーリングリストに加入されていない方、もしよろしければご登録ください。

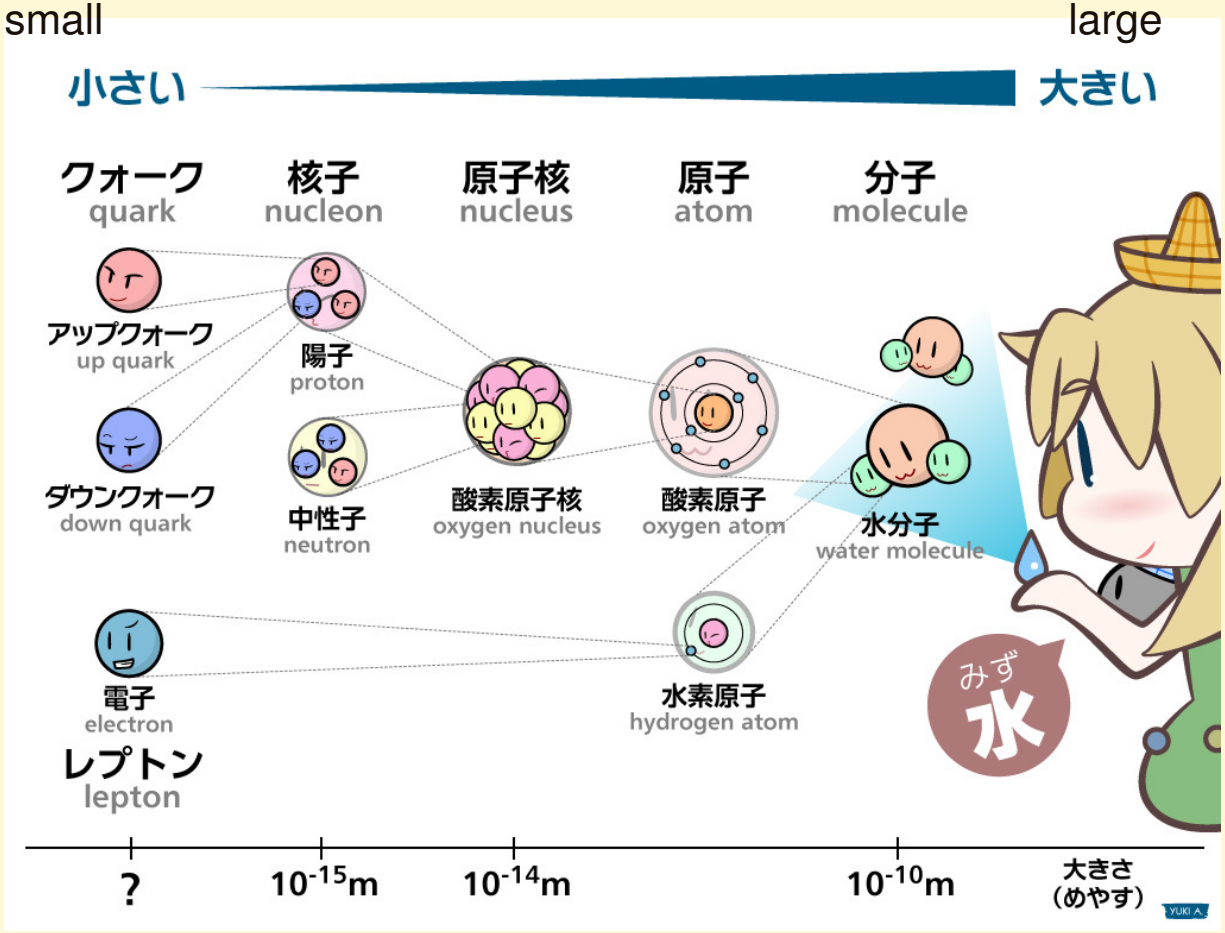
- フロントエンド: 金森までご連絡下さい
- バックエンド: 松古さん

Outline

1. Lattice QCD （のボトルネック）: 疎行列の掛け算
2. Intel Xeon Phi Knights Landing
3. 実装
4. ベンチマークの結果
5. まとめ

水分子の中を見ていくと...

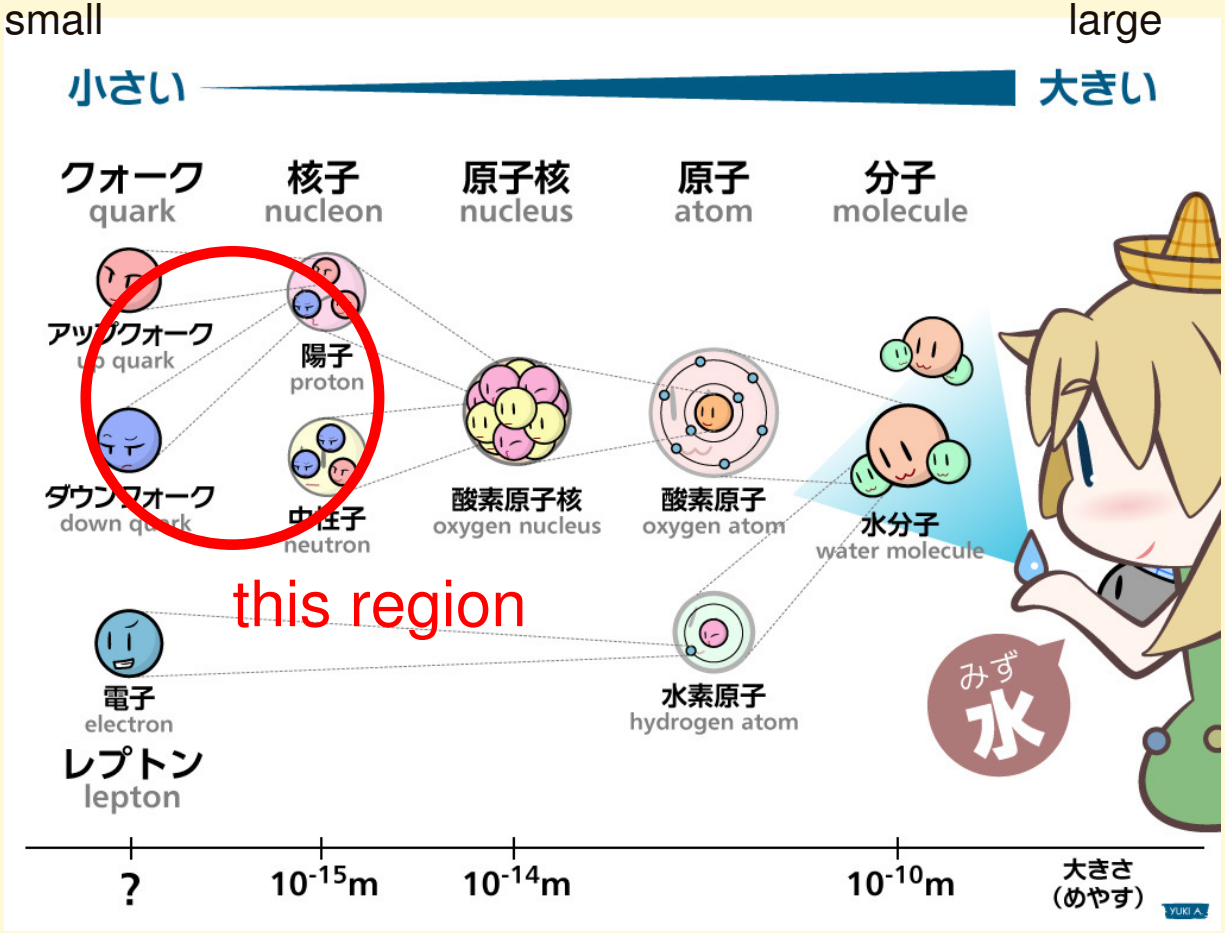
small large



http://higgstan.com

水分子の中を見ていくと...

small large



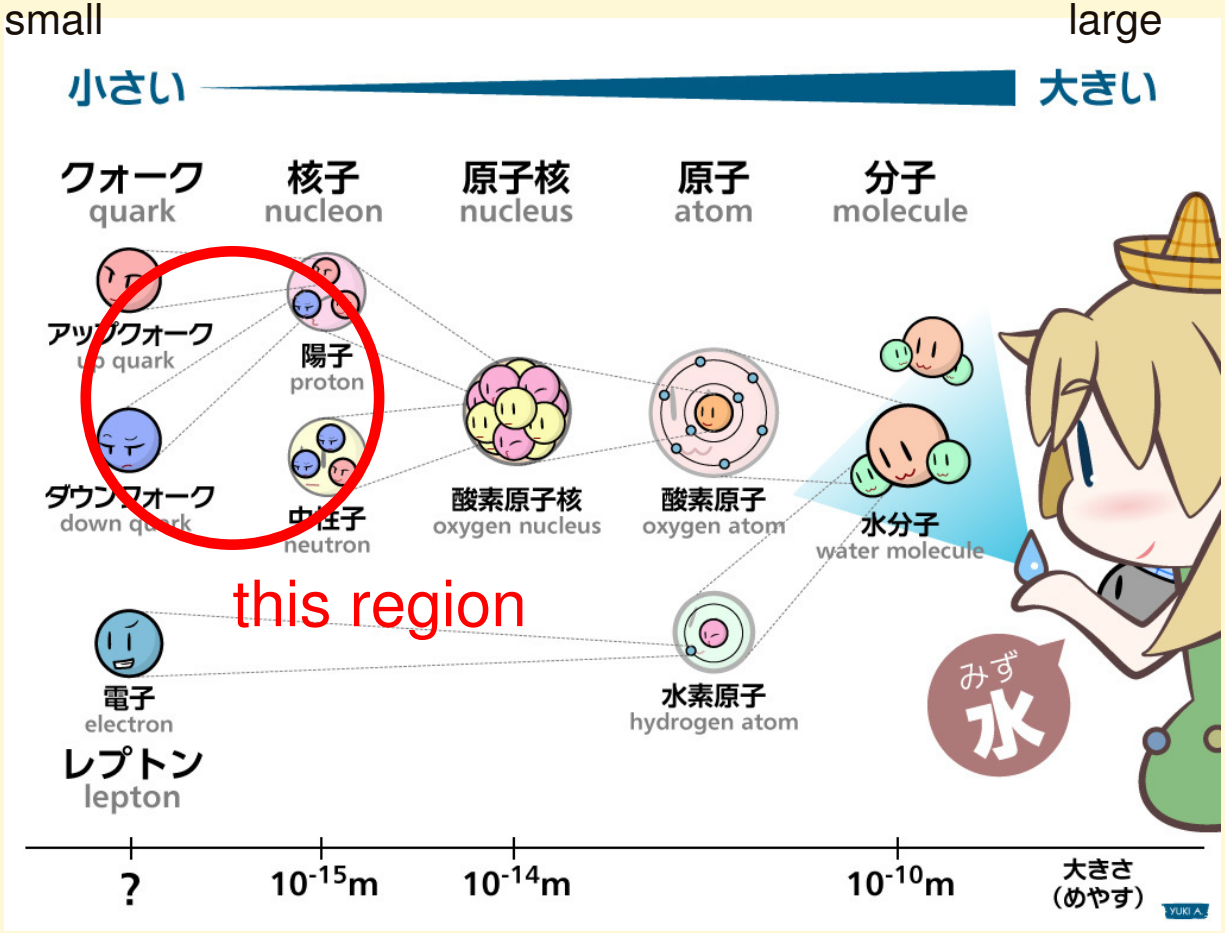
http://higgstan.com

興味の対象

核子の中/核子間

水分子の中を見ていくと...

small large

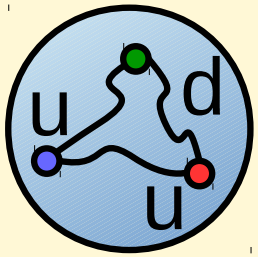


http://higgstan.com

興味の対象

核子の中/核子間

proton



made of 3 quarks
+ gluons

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory
- 核子、様々なバリオンや中間子の質量を再現 etc.

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory
- 核子、様々なバリオンや中間子の質量を再現 etc.
- 新物理の尻尾を捕まえるには、QCD の寄与の精密な計算が必要

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory
- 核子、様々なバリオンや中間子の質量を再現 etc.
- 新物理の尻尾を捕まえるには、QCD の寄与の精密な計算が必要
- 理論は分かっても解けない

$$\partial_\mu F^{\mu\nu} - igA_\mu F^{\mu\nu} + igF^{\mu\nu}A^\nu = -g\bar{\psi}\gamma^\nu\psi$$
$$(i\partial_\mu\gamma_\mu - igA_\mu\gamma^\mu - m)\psi = 0$$
$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - ig(A_\mu A_\nu - A_\nu A_\mu)$$

coupling g: 低エネルギーで強結合、非線形

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory
- 核子、様々なバリオンや中間子の質量を再現 etc.
- 新物理の尻尾を捕まえるには、QCD の寄与の精密な計算が必要
- 理論は分かっても解けない

$$\partial_\mu F^{\mu\nu} - igA_\mu F^{\mu\nu} + igF^{\mu\nu}A^\nu = -g\bar{\psi}\gamma^\nu\psi$$
$$(i\partial_\mu\gamma_\mu - igA_\mu\gamma^\mu - m)\psi = 0$$
$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - ig(A_\mu A_\nu - A_\nu A_\mu)$$

coupling g: 低エネルギーで強結合、非線形

- 系を（うまく）離散化すれば（=紫外正則化）非摂動的に定義できる: Lattice QCD

K.Wilson, Phys.Rev. D10 (1974) 2445-2459

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory
- 核子、様々なバリオンや中間子の質量を再現 etc.
- 新物理の尻尾を捕まえるには、QCD の寄与の精密な計算が必要
- 理論は分かっても解けない

$$\partial_\mu F^{\mu\nu} - igA_\mu F^{\mu\nu} + igF^{\mu\nu}A^\nu = -g\bar{\psi}\gamma^\nu\psi$$
$$(i\partial_\mu\gamma_\mu - igA_\mu\gamma^\mu - m)\psi = 0$$
$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - ig(A_\mu A_\nu - A_\nu A_\mu)$$

coupling g: 低エネルギーで強結合、非線形

- 系を（うまく）離散化すれば（=紫外正則化）非摂動的に定義できる: Lattice QCD K.Wilson, Phys.Rev. D10 (1974) 2445-2459
- 経路積分 $\simeq$ 統計力学系 $\simeq$ モンテカルロでアンサンブル平均!

Quantum ChormoDyanamics (QCD)

- クォークとグルーオンの相互作用
- 理論は確立: SU(3) gauge theory
- 核子、様々なバリオンや中間子の質量を再現 etc.
- 新物理の尻尾を捕まえるには、QCD の寄与の精密な計算が必要
- 理論は分かっても解けない

  $\partial_\mu F^{\mu\nu} - igA_\mu F^{\mu\nu} + igF^{\mu\nu}A^\nu = -g\bar{\psi}\gamma^\nu\psi$

  $(i\partial_\mu\gamma_\mu - igA_\mu\gamma^\mu - m)\psi = 0$

  $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - ig(A_\mu A_\nu - A_\nu A_\mu)$

  coupling g: 低エネルギーで強結合、非線形
- 系を（うまく）離散化すれば（=紫外正則化）非摂動的に定義できる: Lattice QCD K.Wilson, Phys.Rev. D10 (1974) 2445-2459
- 経路積分 $\simeq$ 統計力学系 $\simeq$ モンテカルロでアンサンブル平均!
- 「シミュレーション」といっても...

  $\times$ 系の時間発展を追う

  ◯ 多数の仮想世界（＝場の配位）を用意して、測定する

  cf. （量子力学）何度も実験で測定して期待値を求める

ボトルネック

Dirac 方程式 $D|\psi\rangle = |b\rangle$ を解くこと: 線形 solver

$$\int \mathcal{D}A\mathcal{D}\psi\mathcal{D}\overline{\psi}\exp[-S_{\text{gauge}} - \int d^4x\overline{\psi}D\psi] = \int \mathcal{D}A \det D[A]\exp[-S_{\text{gauge}}]$$

- 配位生成: $\det D = \int \mathcal{D}\phi\mathcal{D}\overline{\phi}\exp[-\overline{\phi}D^{-1}\phi]$ の評価が必要: 1 配位の生成に $O(10)$ 回の solver
- 測定: $\langle\psi(x)\overline{\psi}(y)\rangle = D^{-1}(x,y)$ 配位毎に $O(10)$ 回の solver が必要

京での具体例

- 配位生成時間の 60% が solver
- 測定時間の 80% が solver

Dirac 演算子 D は 1 階微分演算子: $\gamma_\mu(\partial_\mu - iA_\mu) - m$

⇒疎行列

cf $\partial_x f(x)$⇒
$$\begin{pmatrix} f(2)-f(1) \\ f(3)-f(2) \\ f(4)-f(3) \\ f(5)-f(4) \\ .. \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 & 0 & \cdots \\ 0 & -1 & 1 & 0 & \cdots \\ 0 & 0 & -1 & \ddots & \\ \vdots & & & & \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ f(3) \\ \vdots \end{pmatrix}$$

疎行列に対する solver : Krylov 空間 solver (CG 法など)

$D|x\rangle = |b\rangle \Rightarrow |x\rangle = c_0|b\rangle + c_1 D|b\rangle + c_2 D^2|b\rangle + \cdots$ の形で解を探す

⇒D の掛け算が最も重要!

D の詳細

D の離散化はいろいろある: symmetry, discretization error, computational cost,...

Wilson type, Clover type, Domainwall type,...

D の詳細

D の離散化はいろいろある: symmetry, discretization error, computational cost,...

Wilson type, Clover type, Domainwall type,...

Wilson type: 1368 flop/site, 1.12 byte/flop

$$D_W(x,y) = (m_0+4)\delta_{x,y} - \frac{1}{2}\sum_{\mu=1}^{4}\left[(1-\gamma_\mu)U_\mu(x)\delta_{x+\hat{\mu},y} + (1+\gamma_\mu)U_\mu^\dagger(x-\hat{\mu})\delta_{x-\hat{\mu},y}\right]$$

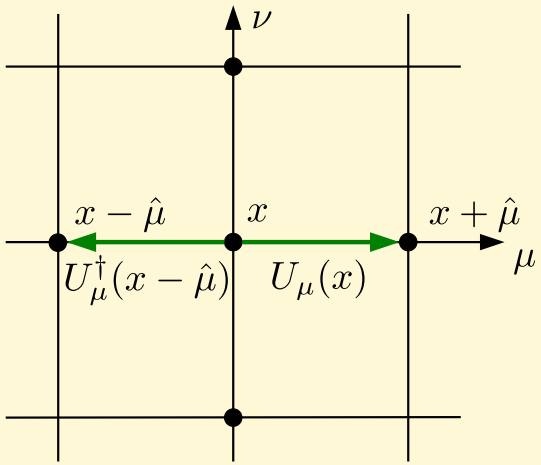
D の詳細

D の離散化はいろいろある: symmetry, discretization error, computational cost,...

Wilson type, Clover type, Domainwall type,...

Wilson type: 1368 flop/site, 1.12 byte/flop

$$D_W(x,y) = (m_0+4)\delta_{x,y} - \frac{1}{2}\sum_{\mu=1}^{4}\left[(1-\gamma_\mu)U_\mu(x)\delta_{x+\hat{\mu},y} + (1+\gamma_\mu)U_\mu^\dagger(x-\hat{\mu})\delta_{x-\hat{\mu},y}\right]$$



4 次元格子上での 9 点ステンシル計算 ($\mu = x, y, z, t$)

$\psi_{i\alpha}(x)$: 各格子点に複素 12 成分

$i = 1, 2, 3$ (カラー)

$\alpha = 1, ..., 4$ (スピン + クォーク/反クォーク)

$U_\mu(x)_{ij}$: 3×3 複素行列 $\in$ SU(3) [gluons]

$(\gamma_\mu)_{\alpha\beta}$: 4×4 成分 permutation 行列

m_0: real number [クォーク質量]

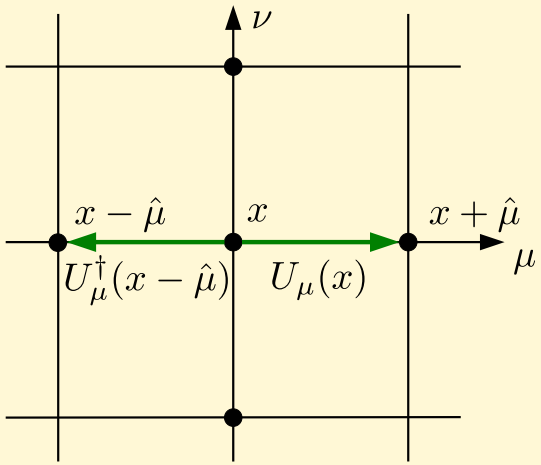
D の詳細

D の離散化はいろいろある: symmetry, discretization error, computational cost,...

Wilson type, Clover type, Domainwall type,...

Wilson type: 1368 flop/site, 1.12 byte/flop

$$D_W(x,y) = (m_0+4)\delta_{x,y} - \frac{1}{2}\sum_{\mu=1}^{4}\left[(1-\gamma_\mu)U_\mu(x)\delta_{x+\hat{\mu},y} + (1+\gamma_\mu)U_\mu^\dagger(x-\hat{\mu})\delta_{x-\hat{\mu},y}\right]$$



4 次元格子上での 9 点ステンシル計算 ($\mu = x, y, z, t$)

$\psi_{i\alpha}(x)$: 各格子点に複素 12 成分

$i = 1, 2, 3$ (カラー)

$\alpha = 1, ..., 4$ (スピン + クォーク/反クォーク)

$U_\mu(x)_{ij}$: 3×3 複素行列 $\in$ SU(3) [gluons]

$(\gamma_\mu)_{\alpha\beta}$: 4×4 成分 permutation 行列

m_0: real number [クォーク質量]

Clover type: 1944 flop/site, 0.94 byte/flop

離散化誤差が Wilson より少ない

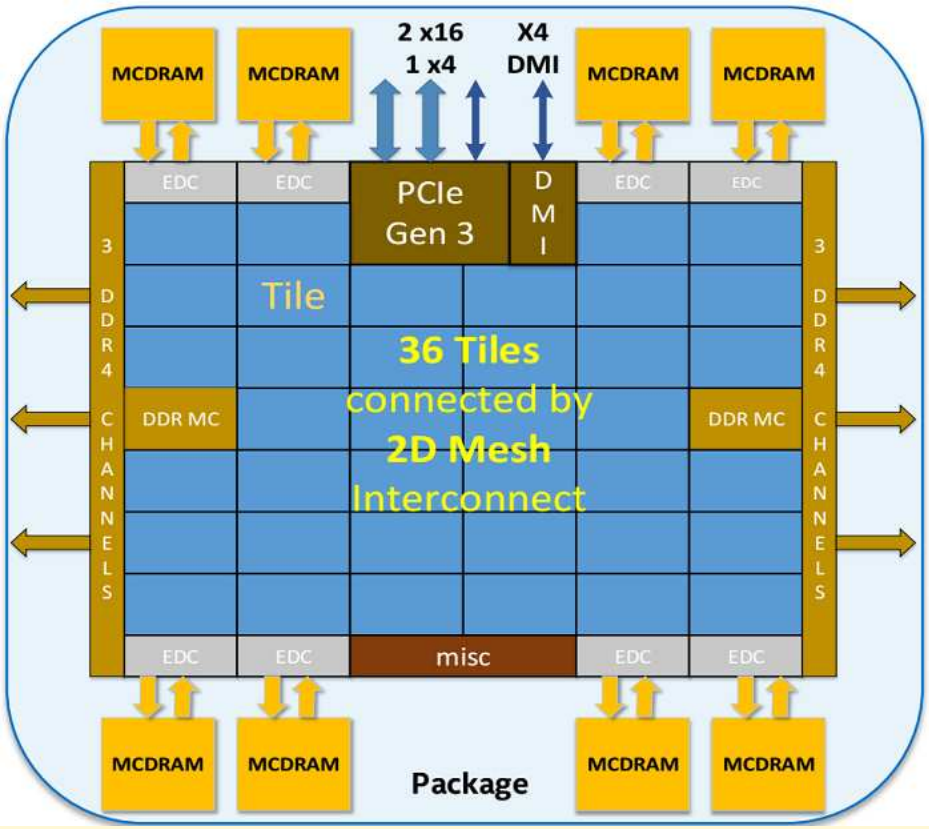
$$D_{\mathrm{clov}}(x,y) = D_{\mathrm{W}} + F(x)\delta_{x,y}, \quad F(x) = \begin{pmatrix} 6\times 6\ \text{Hermite} & 0 \\ 0 & 6\times 6\ \text{Hermite} \end{pmatrix}$$

Intel Xeon Phi KNL

Intel Xeon Phi Knights Landing (KNL)

既に受注終了

- Xeon Phi シリーズ第 2 世代
- Many core: 最大 72 コア
- 4-way hyper-threading
- 2 コア = 1 タイル
  L2 cache を共有
- 512-bit long SIMD vector:
  AVX-512 instruction set
- 16GB MCDRAM
  ($>$ 490 GB/s, B/F$\sim$ 0.2 at OFP)
  cache, flat, hybrid
- SIMD register は 32 個
- L1: 32kB + 32kB
- L2: 1MB (タイル内で共有)



Ref: J. Jeffers, J. Reinders, A. Sodani, “Intel Xeon Phi Processor High Performance Programming Knights Landing Edition” (Elsevier, 2016)

図は https://software.intel.com/en-us/articles/intel-xeon-phi-processor-7200-family-memory-management-optimizations から

使った計算機: Oarkforest-PACS

http://jcahpc.jp/ofp/ofp_intro.html

- Host: 最先端共同 HPC 基盤施設 (筑波大＆東大)
  http://www.cc.u-tokyo.ac.jp/system/ofp/index-e.html
- 25 PFlops: 3 TFlops [double] × 8208 nodes
  14th in Top 500 (2nd in Japan), 2018 年 11 月
- PRIMERGY CX1640 M1 by Fujitsu:
  Intel Xeon Phi 7250 68C 1.4GHz (KNL) + Intel Omni-Path
  network topology: Full-bisection Fat Tree
- 2017 年 4 月から本格運用開始

実装

実装: メニュー

- code framework
- data layout
- AVX-512 intrinsics
- task assignment
- prefetch

実装: メニュー

- code framework
- data layout
- AVX-512 intrinsics
- task assignment : skip
- prefetch : skip

実装

- code framework
  Bridge++ コードセット データ構造拡張版
  C++ object oriented, MPI + OpenMP

実装

- code framework
  Bridge++ コードセット データ構造拡張版
  C++ object oriented, MPI + OpenMP

- AVX-512 intrinsics
  お見せするベンチマークは `Simd` directory in Grid library から wrapper を借用　(Boyle et al., 2016 `https://github.com/paboyle/Grid`)

  （⇒ 最新のコードは自前の wrapper で書き換えてます、パフォーマンスは同等です）

実装

- code framework
  Bridge++ コードセット データ構造拡張版
  C++ object oriented, MPI + OpenMP

- AVX-512 intrinsics
  お見せするベンチマークは `Simd` directory in Grid library から wrapper を借用　(Boyle et al., 2016 `https://github.com/paboyle/Grid`)

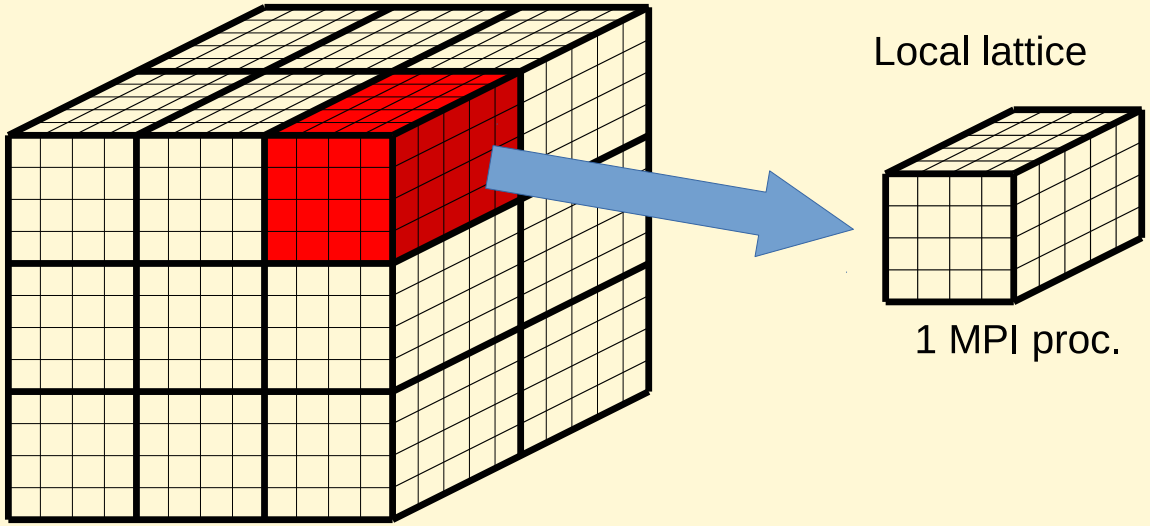
  （⇒ 最新のコードは自前の wrapper で書き換えてます、パフォーマンスは同等です）
- 行列積には manual prefetch も適用 (主に L2)

実装: メニュー

- code framework
- data layout
- AVX-512 intrinsics
- task assignment
- prefetch

実装: メニュー

- code framework
- data layout
- AVX-512 intrinsics
- task assignment
- prefetch



SIMD

single instruction, multiple data: 一つの演算命令が複数のデータに作用
AVX-512: データは 512bits = 16 実 = 8 複素数変数（単精度の場合）
どの自由度を simd で扱うか？

- color : 3
- spin: 4 or 2 [フェルミオン場のみ]
- 格子点: たくさん ⇐ これ

データ構造

外側 内側

$$\texttt{vec}[\texttt{site\_outer}]\underbrace{[\texttt{spin}][\texttt{color}]}_{\text{内部自由度}}\underbrace{[\texttt{site\_inner}]}_{\text{SIMD 自由度}}$$

$$\texttt{site\_idx} = (8 \text{ or } 4) * \texttt{site\_outer} + \texttt{site\_inner}$$

（複素数の実部/虚部の自由度が一番内側）

実装: data layout, 2 types

`vec[site_outer][spin][color][site_inner]`

site 自由度の SIMD 化

512-bit SIMD = 8 （単精度）/4 （倍精度） 個の複素数どう詰める？

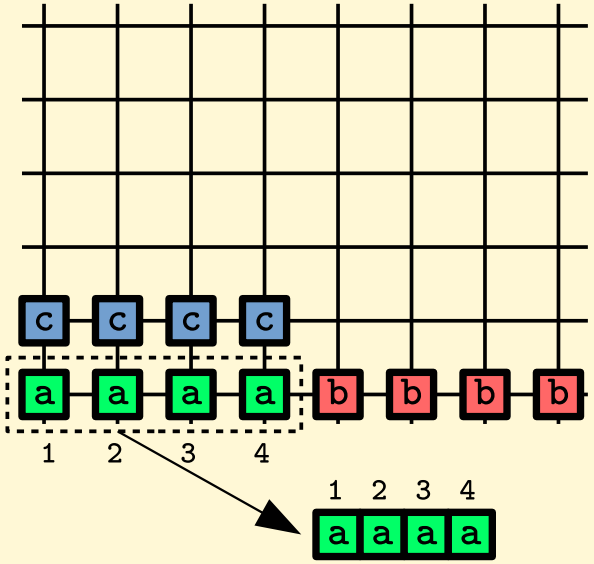
実装: data layout, 2 types

`vec[site_outer][spin][color][site_inner]`

site 自由度の SIMD 化

512-bit SIMD = 8 （単精度）/4 （倍精度） 個の複素数どう詰める？

layout 1



x (=most inner) coordinate

bridge++ w/o SIMD の素朴な拡張

(local lattice size)$_x = 4n$ or $8n$:

less flexible ⇒no MPI in x dir.

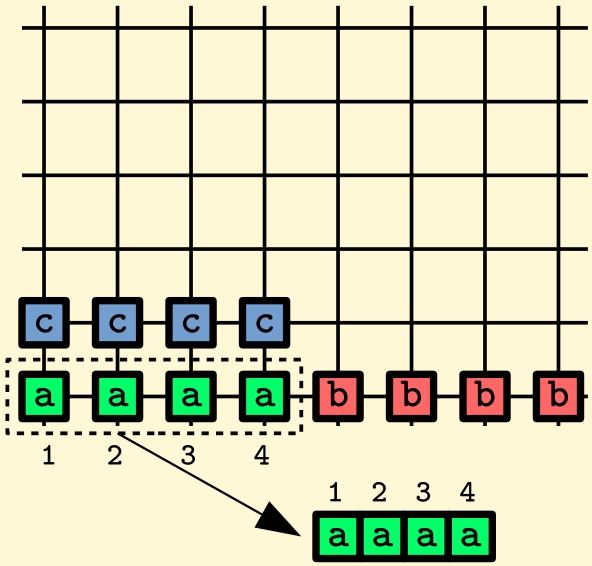
実装: data layout, 2 types

`vec[site_outer][spin][color][site_inner]`

site 自由度の SIMD 化

512-bit SIMD = 8 （単精度）/4 （倍精度） 個の複素数どう詰める？

layout 1



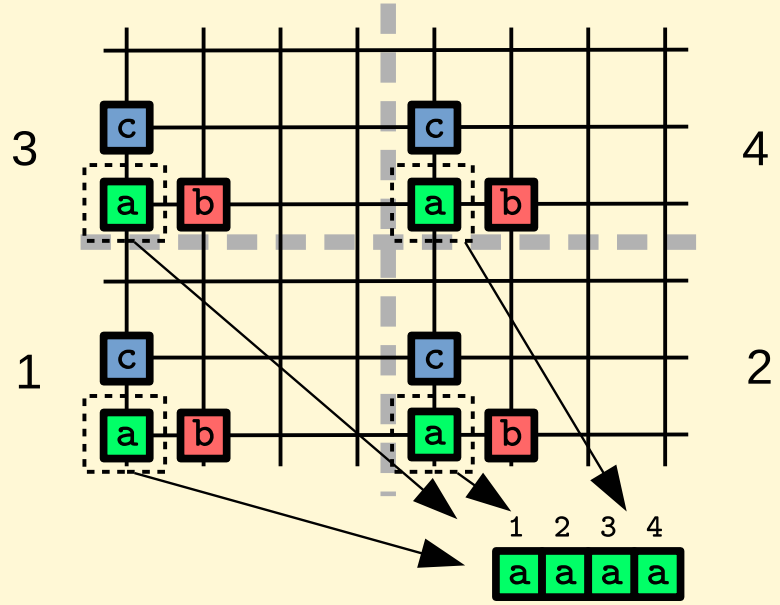
x (=most inner) coordinate

bridge++ w/o SIMD の素朴な拡張

(local lattice size)$_x = 4n$ or $8n$:

less flexible ⇒no MPI in x dir.

layout 2 SIMD lane = subdomain



subdomains in a local lattice

cf. Grid library

隣接格子点へのアクセスが容易 — simd

内部をいじる必要なし (境界は除く)

flexible local lattice size

ベンチマークの結果

KNL

- Intel compiler 18.0.1 with `-O3 -ipo -no-prec-div -xMIC-AVX512`
- MCDRAM: cache mode
- KMP_AFFINITY=compact (unset if thread/core=1)

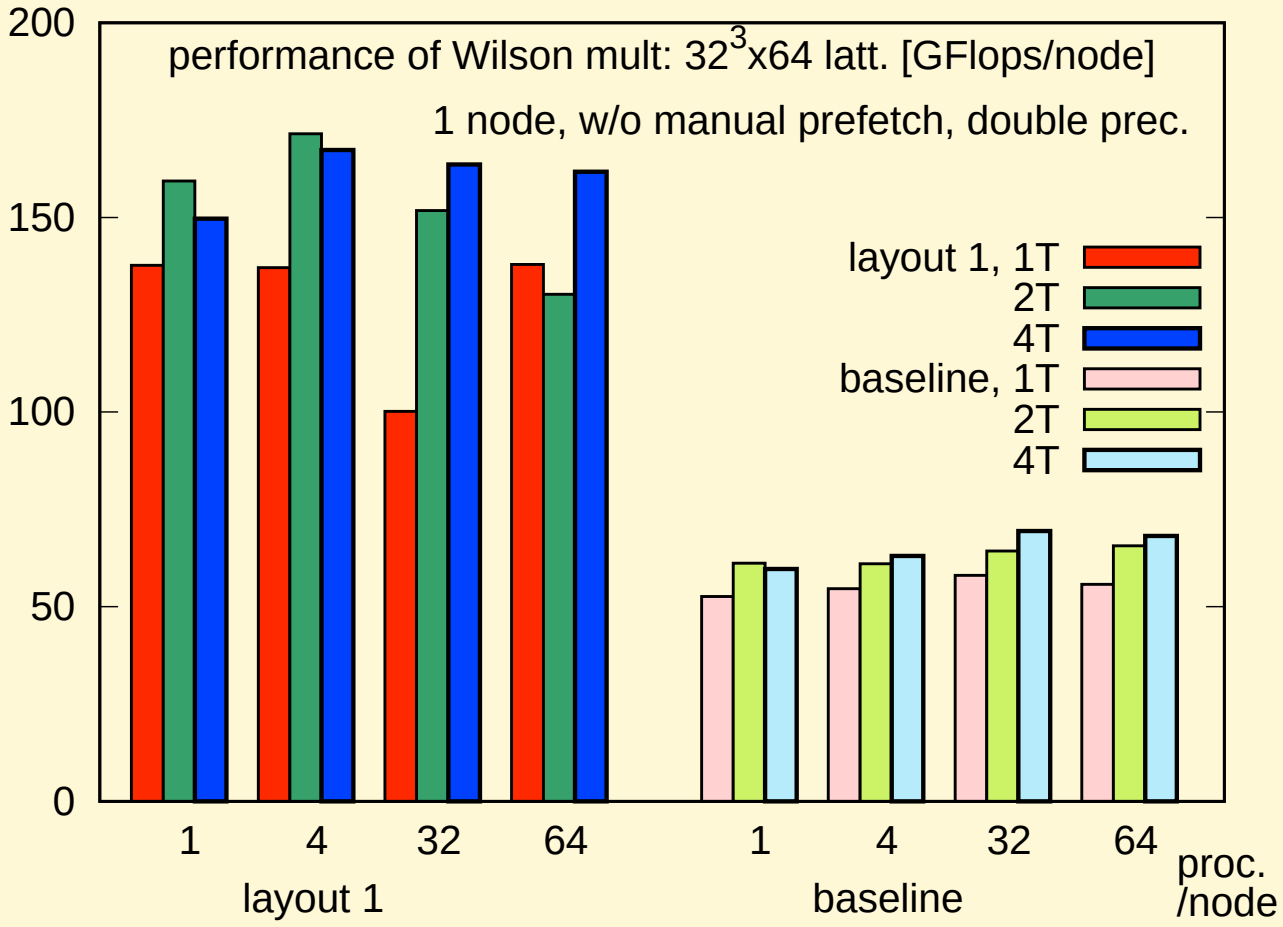
Skylake-SP (the same code as KNL)

- Intel compiler 18.0.0 with `-ipo -O3 -no-prec-div -fp-mpdel fast=2 -xHost`
- KMP_AFFINITY=compact

boundary data copy is enforced even if with single MPI proc.

データ構造と simd 変数の効果: 1 node

グラフ: I.K. and H.Matsufu ICCSA2018 (doi:10.1007/978-3-319-95168-3_31) より



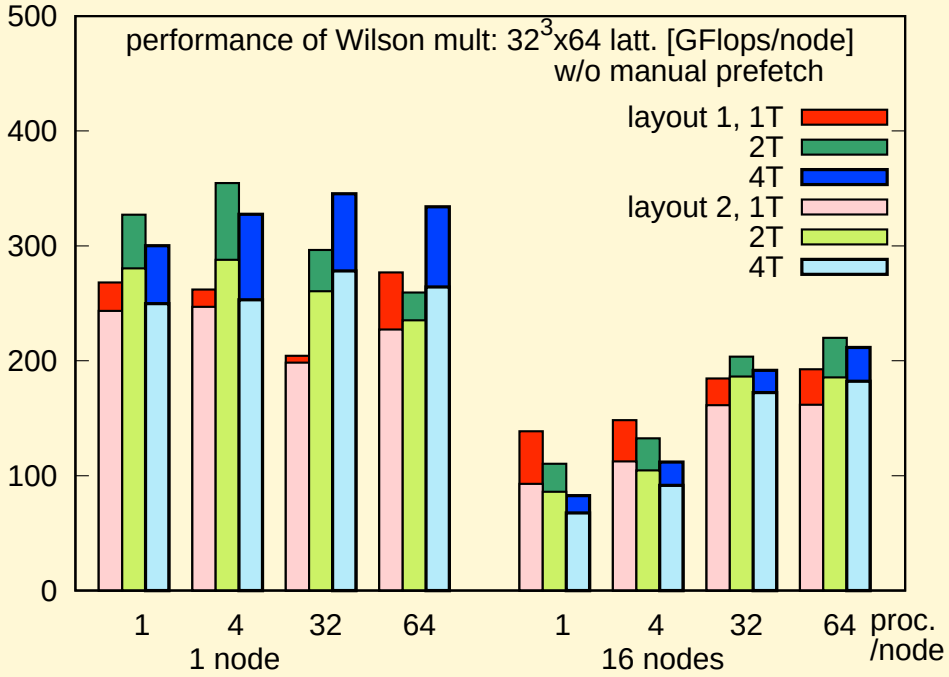
better

SIMD 変数の利用が重要

- layout 1: SIMD 変数使用 `__m512` (w/ wrapper)
- baseline: SIMD 変数不使用 (compiler が使ったかも)

データ構造/データプリフェッチ: Wilson-type mult, 1 or 16 nodes

single prec., $32^3 \times 64$ latt.　グラフ: I.K. and H.Matsufu ICCSA2018 (doi:10.1007/978-3-319-95168-3_31) より

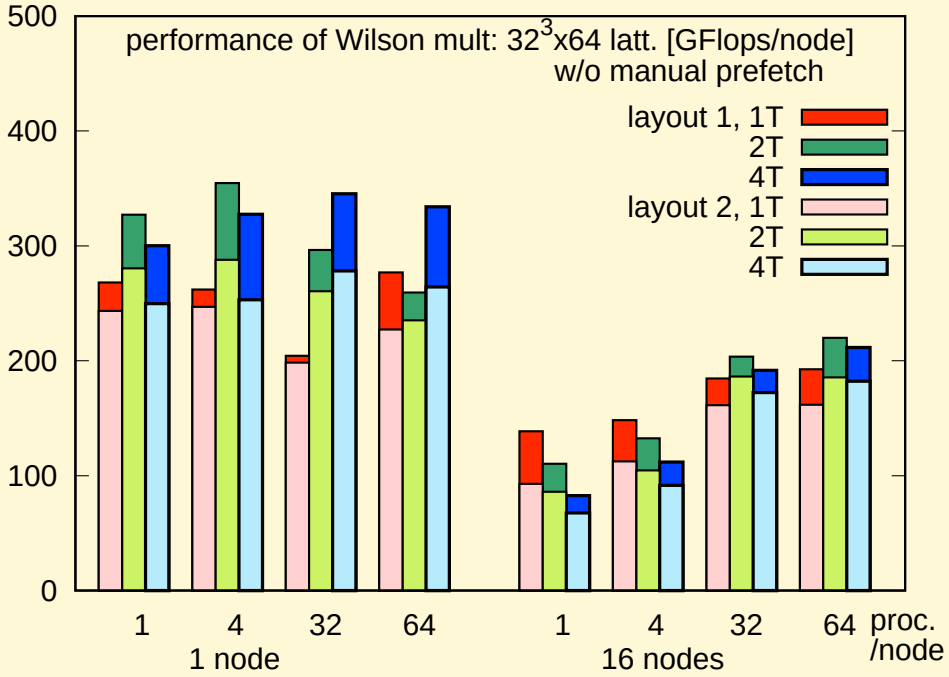


Layout 1 (濃) vs. 2 (淡)

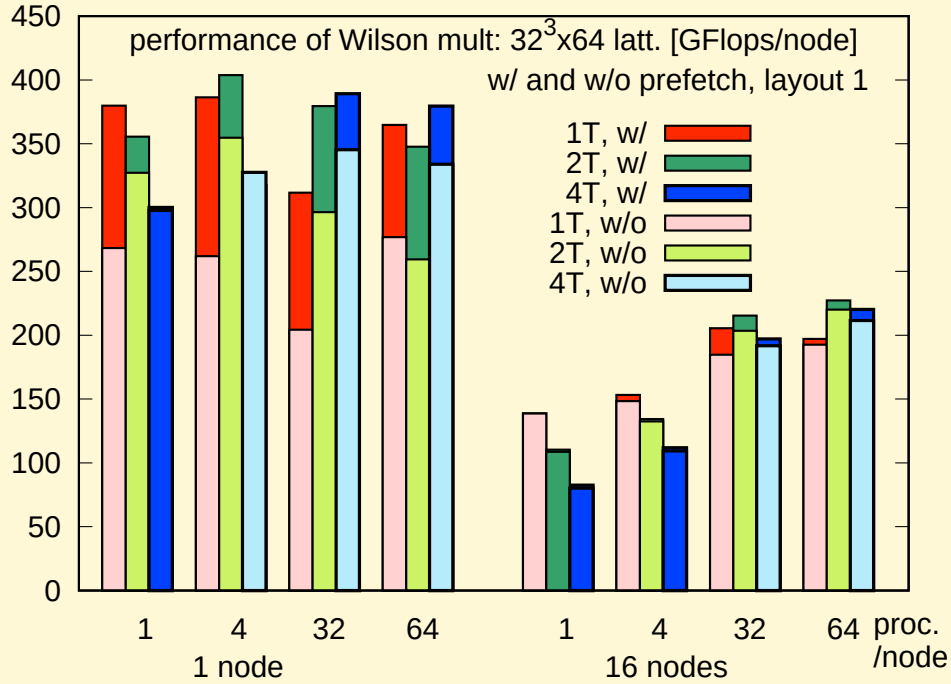
Layout 1 の方がいい

データ構造/データプリフェッチ: Wilson-type mult, 1 or 16 nodes

single prec., $32^3 \times 64$ latt.　グラフ: I.K. and H.Matsufu ICCSA2018 (doi:10.1007/978-3-319-95168-3_31) より



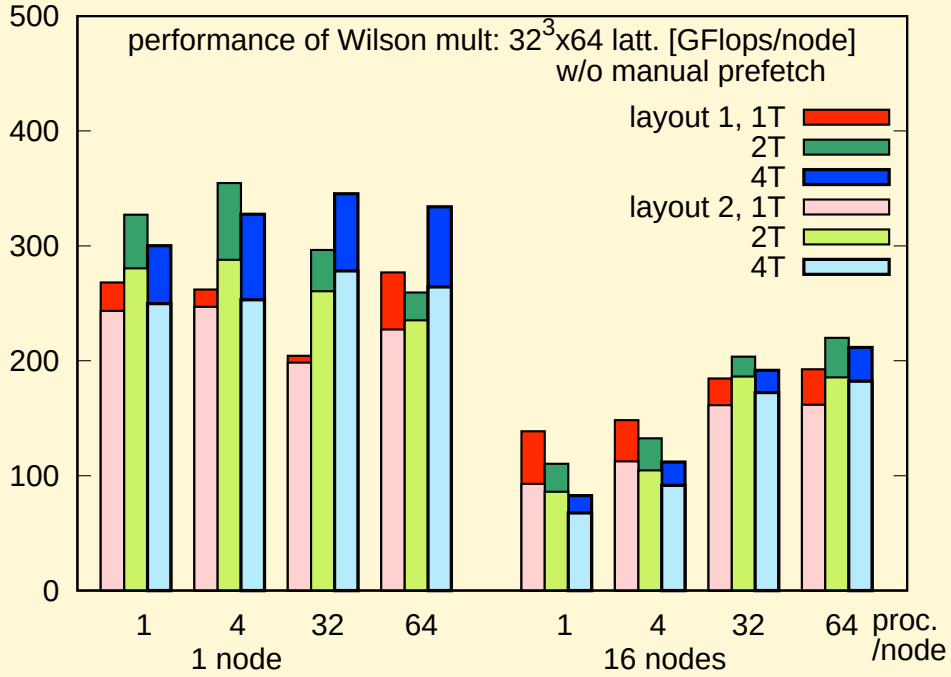
Layout 1 (濃) vs. 2 (淡)
Layout 1 の方がいい



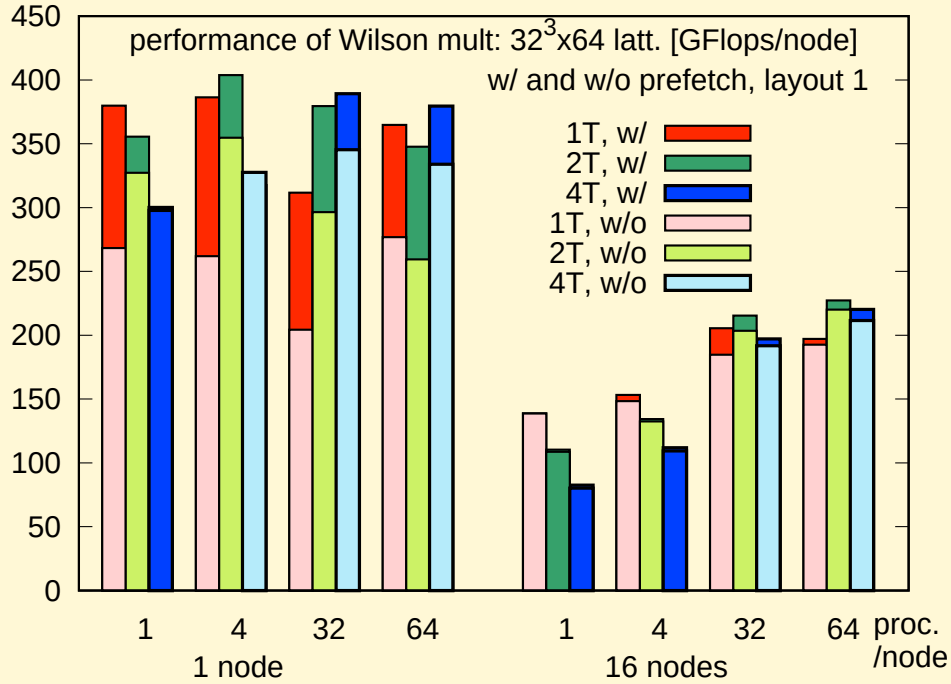
w/ (濃) vs. w/o (淡) manual prefetch (layout 1)
10% 以上向上（16 nodes だと効果は小さい）

データ構造/データプリフェッチ: Wilson-type mult, 1 or 16 nodes

single prec., $32^3 \times 64$ latt.　グラフ: I.K. and H.Matsufu ICCSA2018 (doi:10.1007/978-3-319-95168-3_31) より



Layout 1 (濃) vs. 2 (淡)
Layout 1 の方がいい



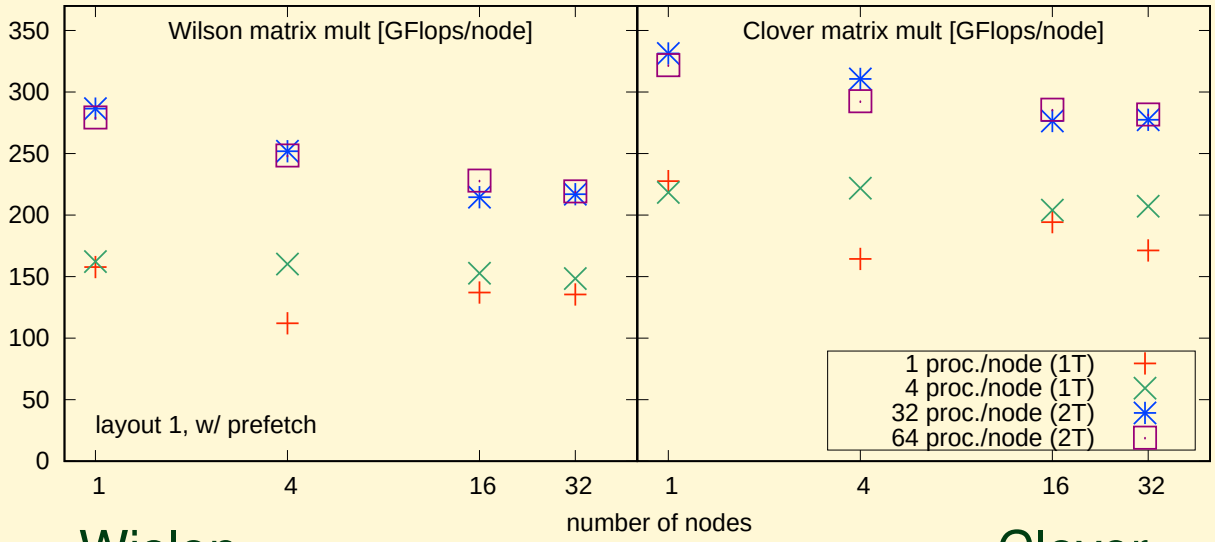
w/ (濃) vs. w/o (淡) manual prefetch (layout 1)
10% 以上向上（16 nodes だと効果は小さい）

local volume が小さく (最小: 32×4^3)、通信を隠蔽しきれていない

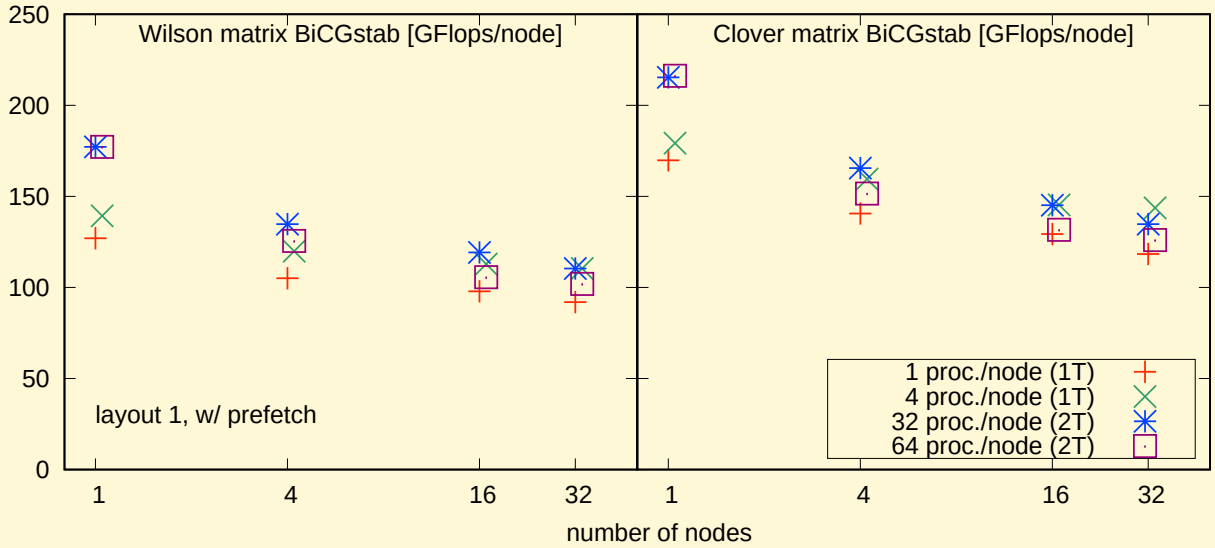
Weak Scaling: D の mult と BiCGStab solve

グラフ: I.K. and H.Matsufu ICCSA2018 (doi:10.1007/978-3-319-95168-3_31) より

$32^3 \times 16$ latt./node, layout 1, w/ prefetch, single prec.



Wislon Clover



Mult : GFlops/node

- Wilson: $\gtrsim 220$
- Clover: $\gtrsim 280$

roofline from MCDRAM BW

Wilson:424/ Clover:506

Solve : GFlops/node

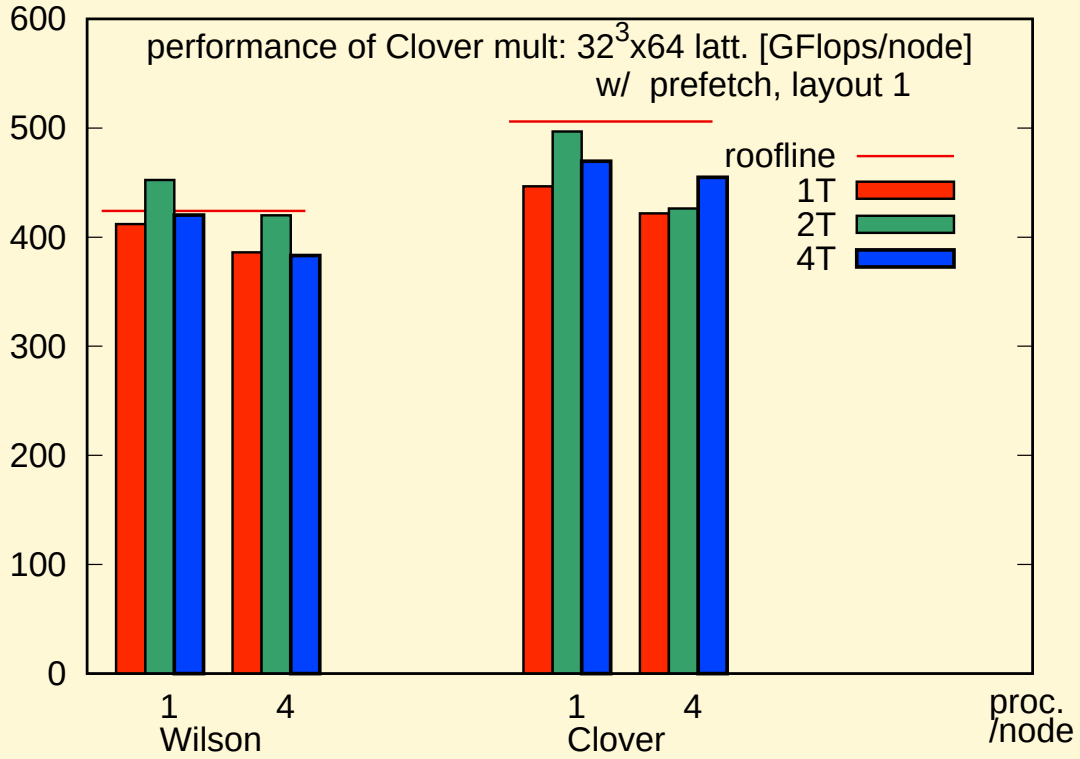
- Wilson: $\gtrsim 110$
- Clover: $\gtrsim 140$

linear algebra requires more BW

1 mpi rank/core か 1 mpi rank/tile がよい

（1 ノードで余分な通信を切った場合）

これまでのベンチマーク: ノード分割していない方向にも必ず MPI 通信（ノード内で通信バッファのコピー）を実行
余分なコピーを省いた場合 — 1 node でどこまで速くできるか

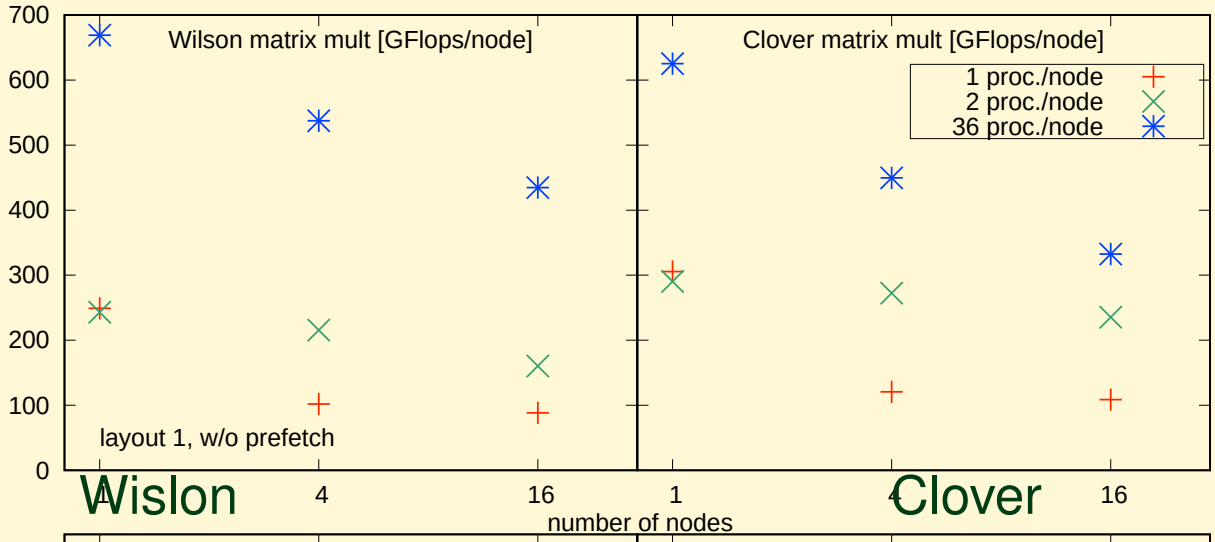


Wilson mult ∼ 450 GFlops (7.5%)
Clover mult ∼ 500 GFlops (8.3%)
さらに高速化するには、データ再利用の工夫が必要

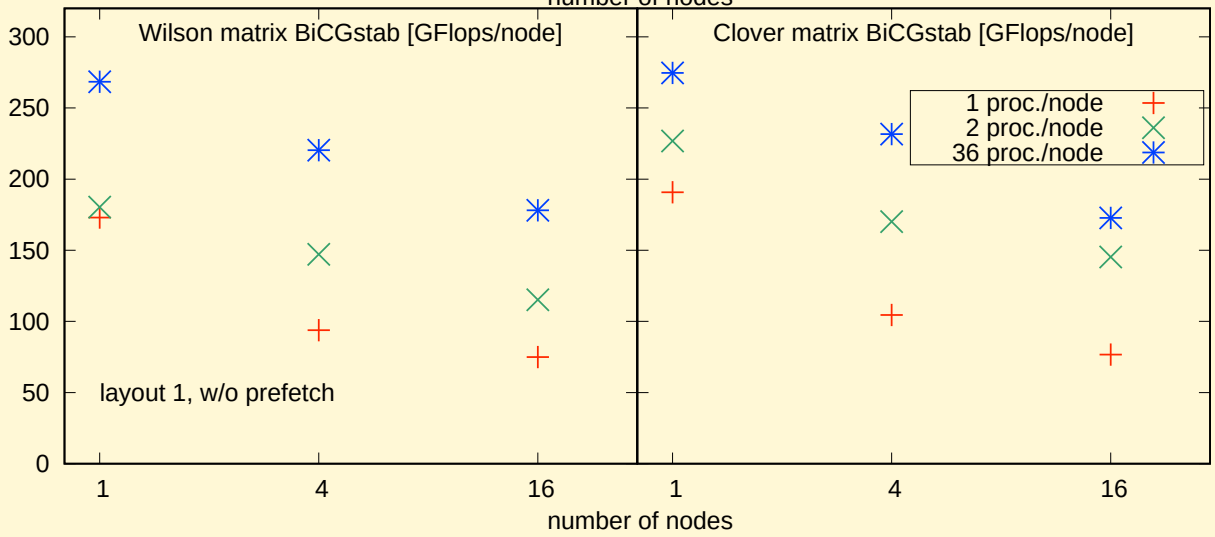
参考: Skylake-SP の場合（九大 ITO）

グラフ: I.K. and H.Matsufu ICCSA2018 (doi:10.1007/978-3-319-95168-3_31) より

$32 \times 16 \times 12 \times 12$ latt./node, layout 1, w/o prefetch, single prec.



Wislon Clover



Mult : GFlops/node

Wilson: $\gtrsim 440$

Clover: $\gtrsim 330$

roofline from DDR4 BW

Wilson:228/ Clover:273

Solve : GFlops/node

Wilson: $\gtrsim 180$

Clover: $\gtrsim 170$

linear algebra requires more BW

- キャッシュの効果! メモリ BW roofline より速い
  local lattice が大きい時: mult はだいたい roofline 通り
- fastest: 1 mpi rank/core

その他の **tips**

- 1 次元 index から 4 次元座標を出す整数演算: 遅い（割り算、余り）
  ($i = x + N_x y + N_x N_y z + N_x N_y N_z t$)
  - 1 次元 index はスレッド並列化で便利
  - リストベクトルで書き換え中
- プロファイラを活用すべし
  Intel vtune amplifier でプリフェッチを入れる場所（$\simeq$ 実行に時間がかかる命令）を探した

まとめ

まとめ（個人の感想です）

Intel Xeon Phi KNL を使ってみました

- SIMD 意識したデータ構造が必須
- instrinsic を使えば確実に SIMD 命令を出してくれる（はず）
  (データ構造 + 適切な # pragma でもある程度行ける場合もある)
- SIMD 変数への wapper を使えば、かなり楽をできる
  明示的な load/store も省ける
- MCDRAM は癖がある — L2 へのプリフェッチが有効、キャッシュ効果によるハイパースケーリングは期待できなさそう
- 1 core/MPI か 2 core/MPI が速い