Intel Xeon Phi (Knights Landing) のパフォーマンス評価の一例

東京大学大学院　新領域創成科学研究科

松田和高，森田直樹，奥田洋司

2017年1月30日　第33回FrontISTR研究会

目次

- 背景と目的
- KNLのアーキテクチャ
- メモリモードとクラスタモード
- STREAM triadによる性能評価
- FrontISTRによる性能評価
- まとめ
- 参考文献

背景と目的

背景

- 計算機の単体性能向上において，動作周波数の向上が限界に達する
  - 消費電力の増大，発熱の限界
- 2004年ごろから計算機はマルチコア化，並列処理の流れ
  - 現在のスパコンも並列処理で性能向上を実現
- 近年はMulti-Channel DRAM(MCDRAM)のような高速メモリがCPUに搭載される

目的

- XeonPhi 「Knights Landing」 (以後KNL)のアーキテクチャ理解
- STREAMベンチマーク，並列有限要素法ソルバー FrontISTRを用いた性能評価

KNLのアーキテクチャ

KNLにはDDR4と積層メモリ (MCDRAM) が搭載

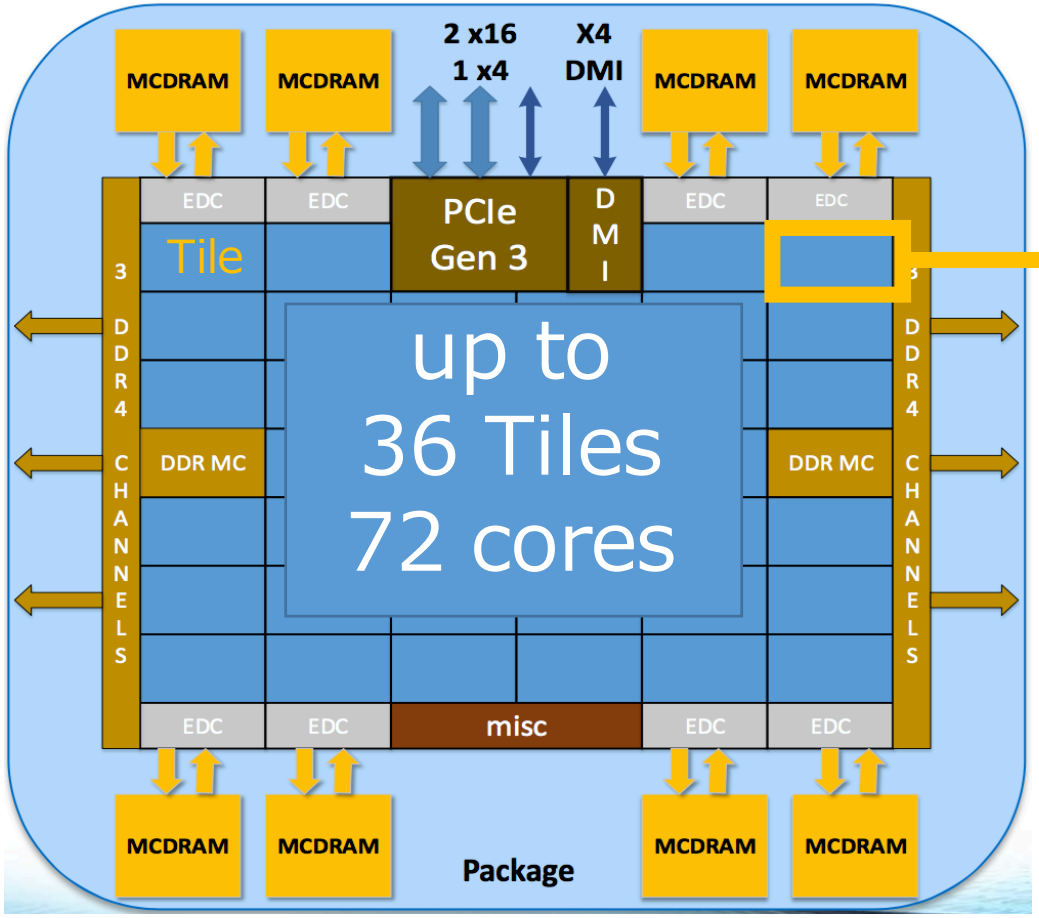

図1-1 KNL Package[1]

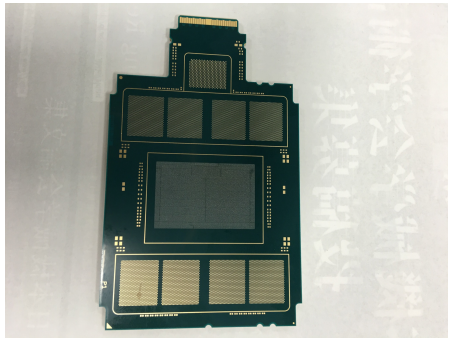

図1-2 KNLのプリント基板

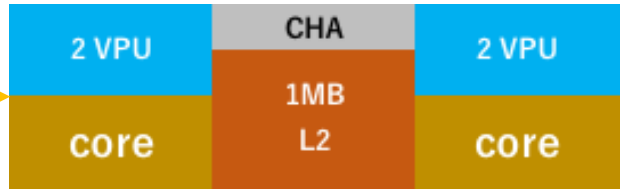

図1-3 Tileの概観[2]

- VPU(vector processing unit)
  512bit register
- CHA(Caching/Home Agent)
  機能は後で詳しく紹介
- DDR4のメモリサイズ
  64GB×6channelにより最大384GBまで搭載可
- MCDRAMのメモリサイズ
  2GB×8個で計16GBが搭載

[1] Avinash Sodani, *Intel Xeon Phi Processor "Knights Landing" Architectural Overview,* 2015.

[2] Avinash Sodani, *Knights Landing (KNL):2nd Generation Intel Xeon Phi Processor*, 2015.

メモリモードとクラスタモード

メモリモード，クラスタモードという合計９通りの使い方がある

メモリモード
(DDR4，MCDRAMの使い方を決める)

- Flat
- Cache
- Hybrid (※)

クラスタモード
（タイル，メモリ間の通信方法を決める）

- All to All (※)
- Quadrant/Hemisphere
- SNC(Sub-NUMA Clustering) (※)

2つのモードはBIOSで設定し，rebootすることで変更できる

(※)Oakforest-PACSでは選択不可

メモリモード

メモリ (DDR4，MCDRAM) の使い方に３つのモードがある

- Flat
- Cache
- Hybrid

Flatモード

- MCDRAMはNUMA ノードとして使われる
- MCDRAMの使用は ユーザーが制御

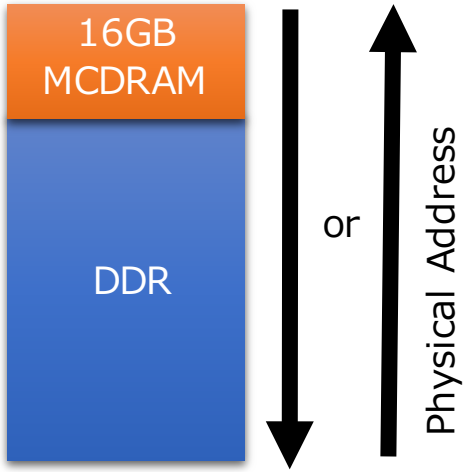

図2-1 Flatモードの概要

Cacheモード

- MCDRAMをLast Level Cache(LLC)として使用
- ハードウェアがMCDRAMの使い方を制御

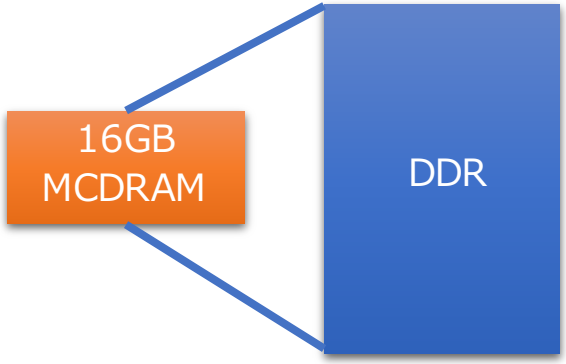

図2-2 Cacheモードの概要

Hybridモード

- FlatモードとCacheモードの組み合わせ
- メモリ比率は設定可能

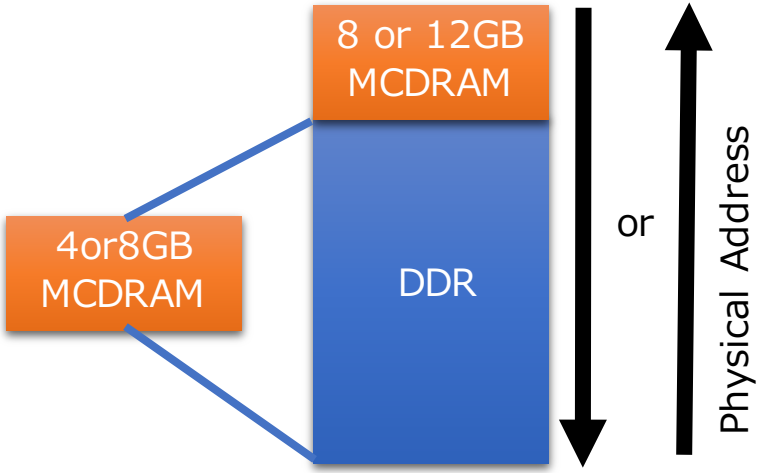

図2-3 Hybridモードの概要

メモリモード変更はBIOSで設定し，rebootすることで有効になる

クラスタモード

タイル，メモリ間の通信方法に
大きく３つのモードがある

- All to All
- Quadrant/Hemisphere
- SNC(Sub-NUMA Clustering)

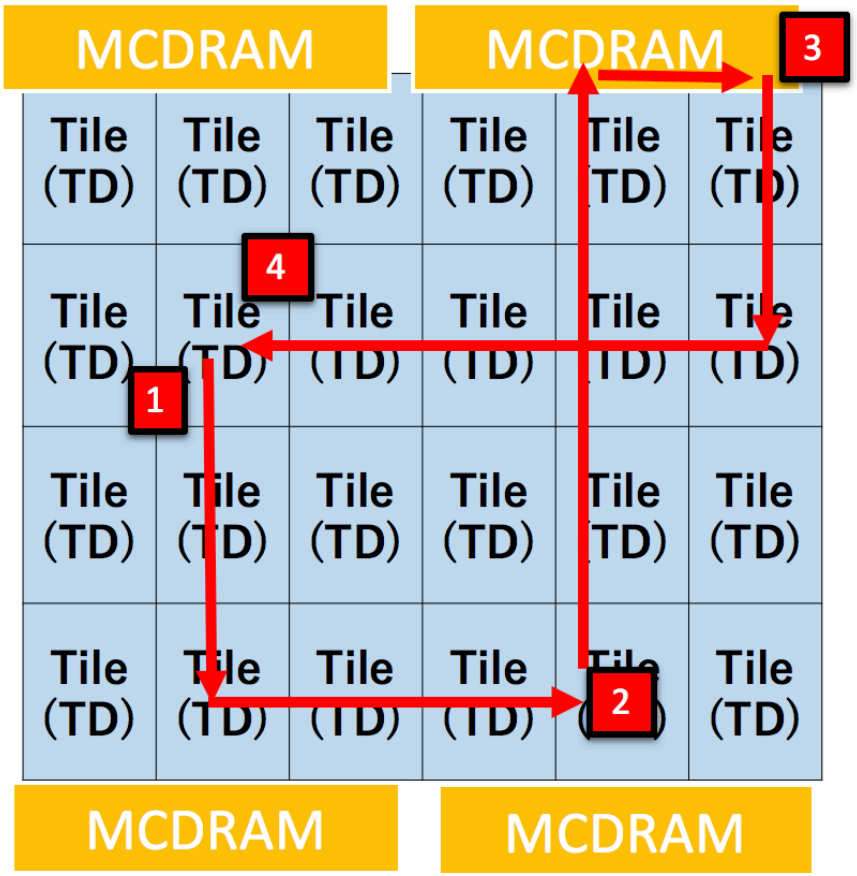

図3 L2 miss時のメモリアクセスの流れ

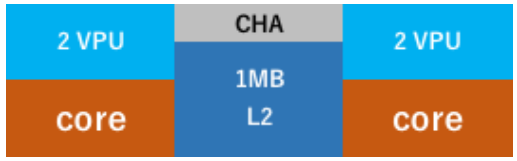

図1-3 Tileの概観

CPUコアは，欲しいデータが
どこのタイル(L2キャッシュ)にあるかを
Caching/Home Agent(CHA)に確認

L2 missした時に
クラスタモードの性能差がでる

Typical Read L2 miss
1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory send the data to the requestor

クラスタモード: All to All

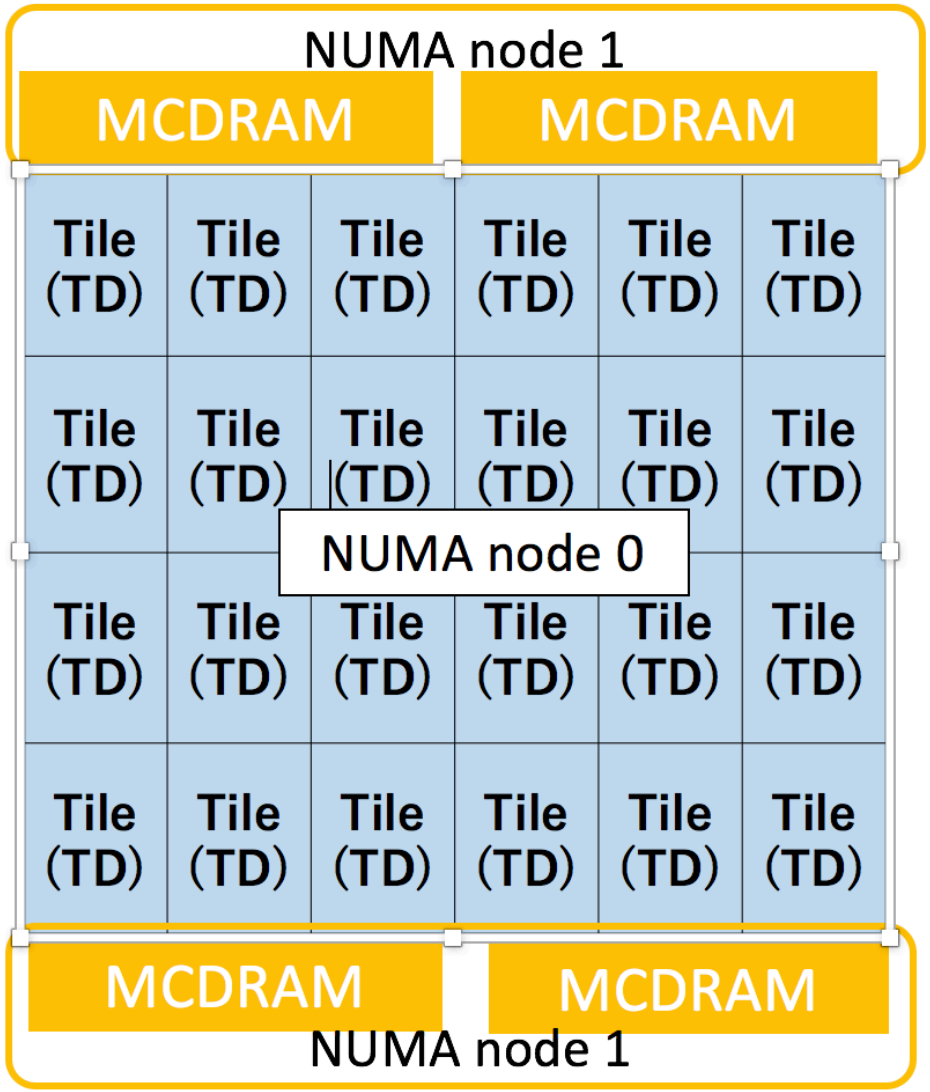

図4 All to Allの概念

NUMA（Non-Uniform Memory Access）

Tileのディレクトリとメモリに affinityがない

Tileのディレクトリ配置とメモリ通信はハードウェアに自動で任せる。

DDR4 $numactl --membind=0 ./a.out

MCDRAM $numactl --membind=1 ./a.out

3つのモードのうち一番メモリパフォーマンスが悪い

(※)Oakforest-PACSでは選択不可

クラスタモード: Quadrant/Hemisphere

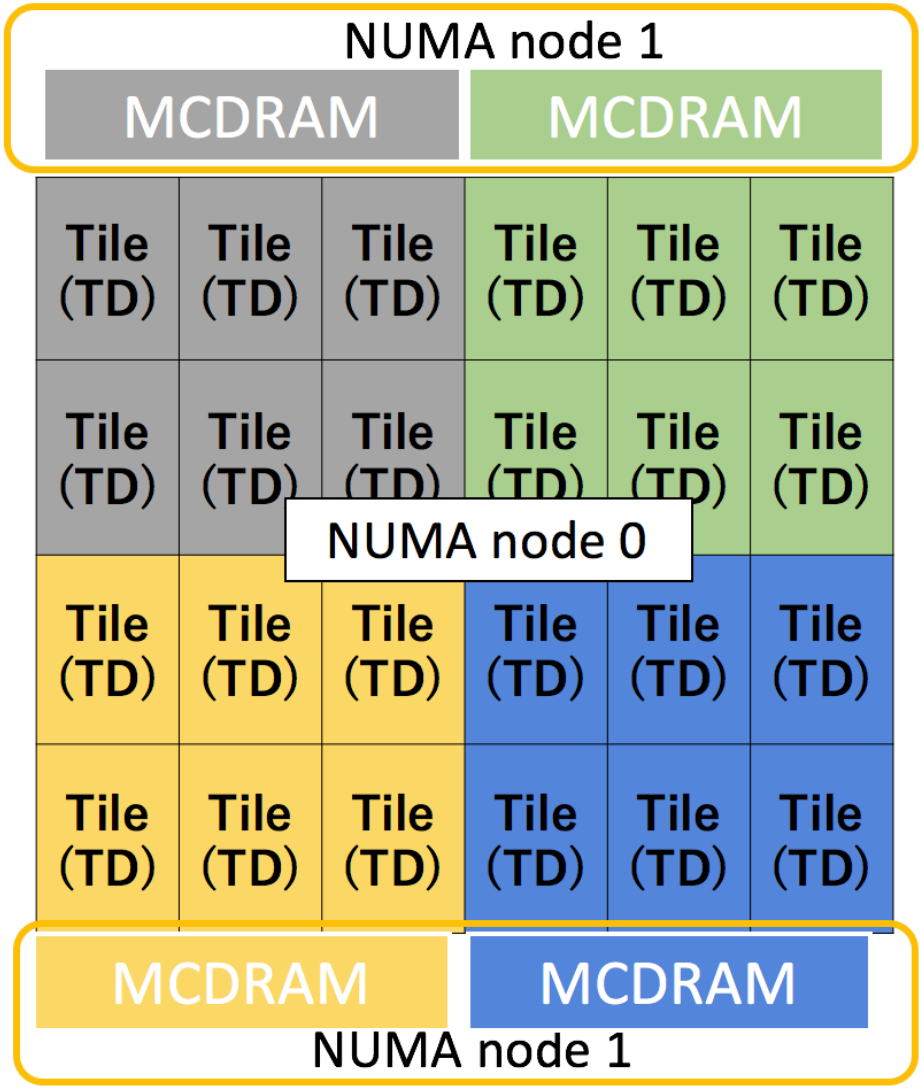

図5 Quadrantの概念

仮想的に４つまたは２つの象限に分割

同じ象限にTileのディレクトリとメモリが配置されるよう，アドレスをハッシュする

All to Allよりは
メモリパフォーマンスが良い

クラスタモード: SNC-4/SNC-2

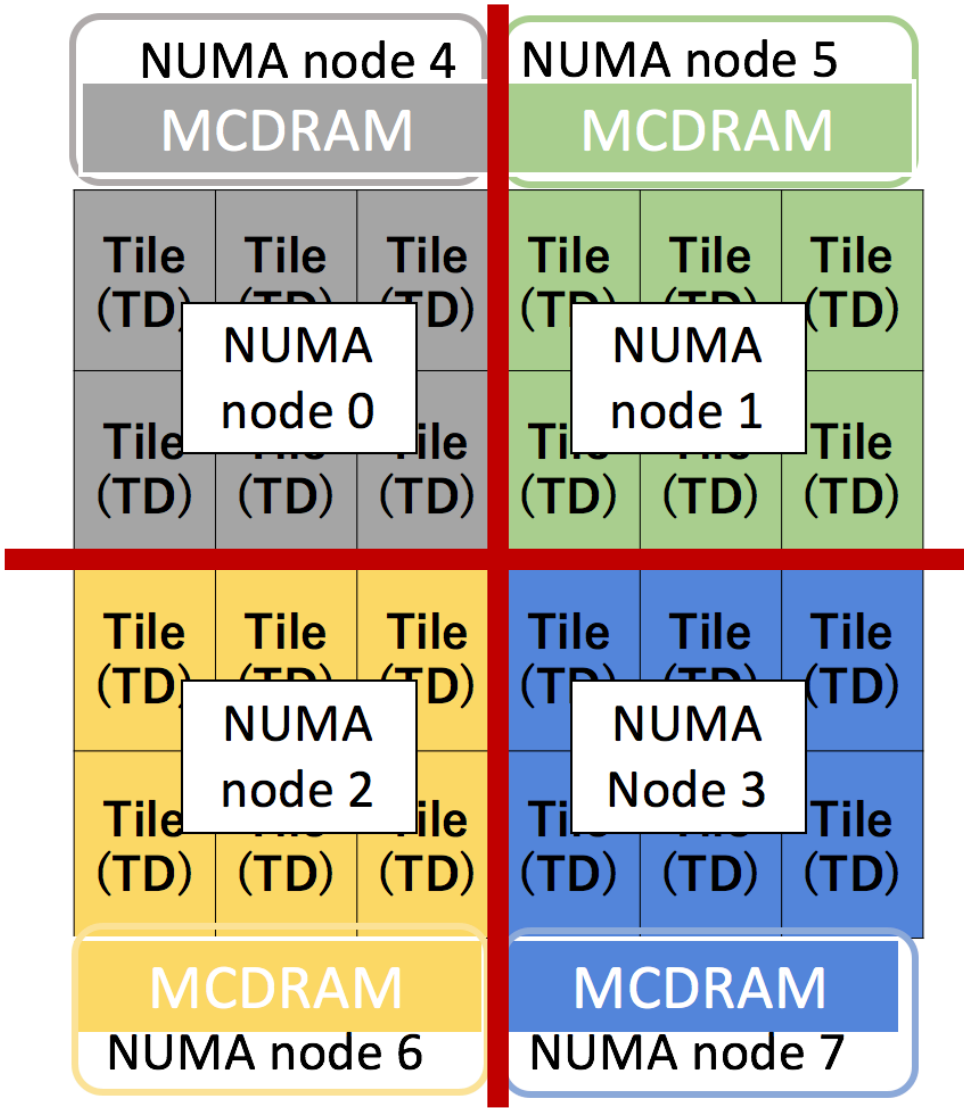

図6 SNC-4の概念

４つまたは２つのNUMAに分割

Intelが公開しているシェルスクリプトなどを利用して，
各NUMAに所属してる
タイルディレクトリとメモリを
bindさせることが可能

メモリパフォーマンスが
最も良い

(※)Oakforest-PACSでは選択不可

ソースコード命令によるMCDRAM利用方法

Intelの資料[3]より抜粋

Option B.1: Using Memkind Library to Access MCDRAM

Allocate 1000 floats from DDR

```
float   *fv;

fv = (float *)malloc(sizeof(float) * 1000);
```

Allocate 1000 floats from MCDRAM

```
#include <hbwmalloc.h>

float   *fv;

fv = (float *)hbw_malloc(sizeof(float) * 1000);
```

Allocate arrays from MCDRAM and DDR in Intel® Fortran Compiler

```
c      Declare arrays to be dynamic
       REAL, ALLOCATABLE :: A(:), B(:), C(:)

!DEC$ ATTRIBUTES FASTMEM :: A

       NSIZE=1024
c
c      allocate array 'A' from MCDRAM
c
       ALLOCATE (A(1:NSIZE))
c
c      Allocate arrays that will come from DDR
c
       ALLOCATE  (B(NSIZE), C(NSIZE))
```


[3] Shuo Li, Karthik Raman, Ruchira Sasanka, Andrey Semin, Enhancing Application Performance using Heterogeneous Memory Architectures on the Many-core Platform, 2016,

DDR4とMCDRAMの特徴

WciL: Worst case interrupt Latency

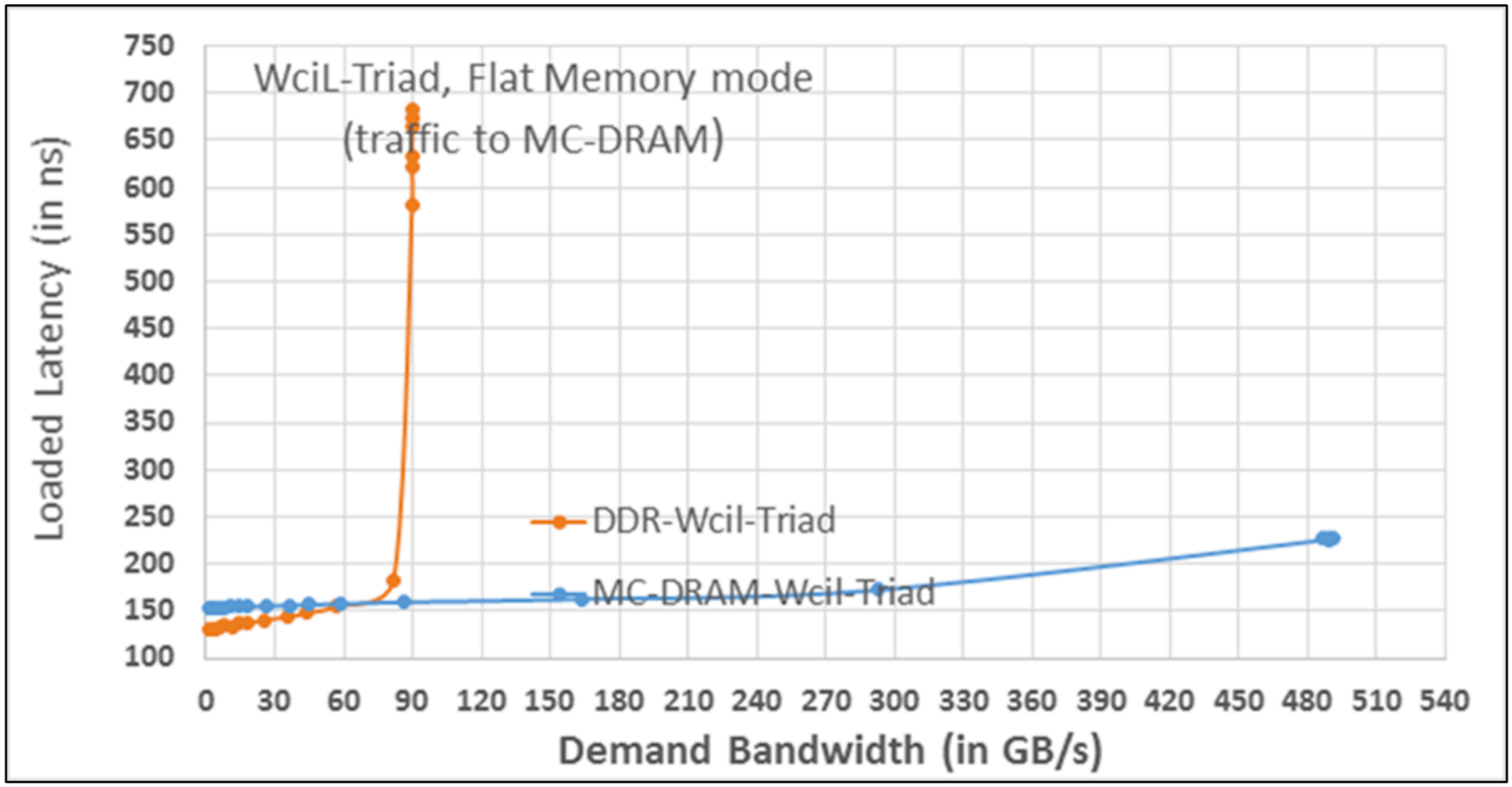

図7　要求メモリサイズとレイテンシの関係[3]

[3] Shuo Li, Karthik Raman, Ruchira Sasanka, Andrey Semin, Enhancing Application Performance using Heterogeneous Memory Architectures on the Many-core Platform, 2016,

KNL搭載機材の性能とFX10，FX100との比較

表1 KNL搭載機材及びFX10, FX100との比較

	Oakforest-PACS		FX10	FX100
搭載CPU	Xeon Phi Processor 7250		SPARC64 IXfx	SPARC64 XIfx
理論演算性能[Gflops]	3046.4		236.5	1126.4
コア数	68		16	32
スレッド数	272		16	32
動作周波数[GHz]	1.4		1.848	2.2
Flops/Clock	32[4]		8[5]	16[6]
メモリサイズ[GB]	DDR4	MCDRAM	32	32
	96	16		
理論メモリ速度[GB/s]	115	490	85	240
Hardware Byte/Flop	0.0377	0.161	0.359	0.213

理論演算性能 ＝ コア数 × 動作周波数×Flops/Clock

Hardware Byte/Flop ＝ 理論メモリ速度 / 理論演算性能

[4] David Kanter, Knights Landing Details, 2014.

[5] 東京大学情報基盤センタースーパーコンピューティング部門, ”第2章 FX10 スーパーコンピュータシステムについて”.

[6] 富士通株式会社 次世代テクニカルコンピューティング開発本部, ”FUJITSU Supercomputer PRIMEHPC FX100 次世代技術への進化”, 2014.

STREAM triadによる性能評価

STREAM triadによる性能評価の概要

KNL機材の性能評価をする上で，STREAM triad測定を実施

長さがSTREAM_ARRAY_SIZEのdouble型(8[Byte])配列a, b, cと
double型変数scalarで積和演算を行い，メモリスループットを測定するプログラム

表2 STREAM triadのカーネル部分

```
#pragma omp parallel for
for (j=0; j<STREAM_ARRAY_SIZE; j++)
        a[j] = b[j]+scalar*c[j];
#endif
```

カーネル部分よりSTREAM triadの要求Byte/Flopを求める

double型scalarはレジスタに乗ることが期待できるので
double型配列a, b, cと演算(Flop)が2つという事実に注目すれば良い

$$\text{STREAM triadの要求Byte/Flop} = \frac{8[\text{Byte}]\times 3}{2} = \frac{24}{2} = 12.0$$

STREAM triadの対ピーク性能，演算速度限界値

STREAM triadの要求Byte/Flopが12.0より

- STREAM triadの対ピーク性能[%]

$$= \frac{\text{Hardware Byte/Flop}}{\text{STREAM triadの要求Byte/Flop}} \times 100 = \frac{\text{Hardware Byte/Flop}}{12.0} \times 100$$

- STREAM triadの演算速度限界値[Gflops]

= 理論演算性能[Gflops]× STREAM triadの対ピーク性能[%] / 100

表3 KNL搭載機材と，FX10, FX100の比較 (STREAM triadの対ピーク性能，演算速度限界値)

	Oakforest-PACS		FX10	FX100
	DDR4	MCDRAM		
理論演算性能[Gflops]	3046.4	3046.4	236.5	1126.4
理論メモリ速度[GB/s]	115	490	85	240
Hardware Byte/Flop	0.0377	0.161	0.359	0.213
STREAM triadの対ピーク性能[%]	0.314	1.34	2.99	1.78
STREAM triadの演算速度限界値[Gflops]	9.57	40.9	7.08	20.0

Oakforest-PACSの性能評価条件

使用コンパイラ: Intel C++ compiler 17.0.1

Intelから発表されている
STREAM計測最適化条件[7]を遵守

測定におけるコンパイルオプション

```
-mcmodel medium -shared-intel -O3 -xMIC-AVX512
-DSTREAM_ARRAY_SIZE=67108864 -DOFFSET=0
-DNTIMES=10 -qopenmp -qopt-streaming-stores always
–lmemkind
```

stream.c ソースコード内のmalloc関数をhbw_malloc関数へ変更

メモリスループット評価に関して,
スレッド数と要求メモリサイズを変化させる

[7] Karthik Raman, *Optimizing Memory Bandwidth in Knights Landing on Stream Triad*, 2016-6-20.

Oakforest-PACSのメモリスループット評価①

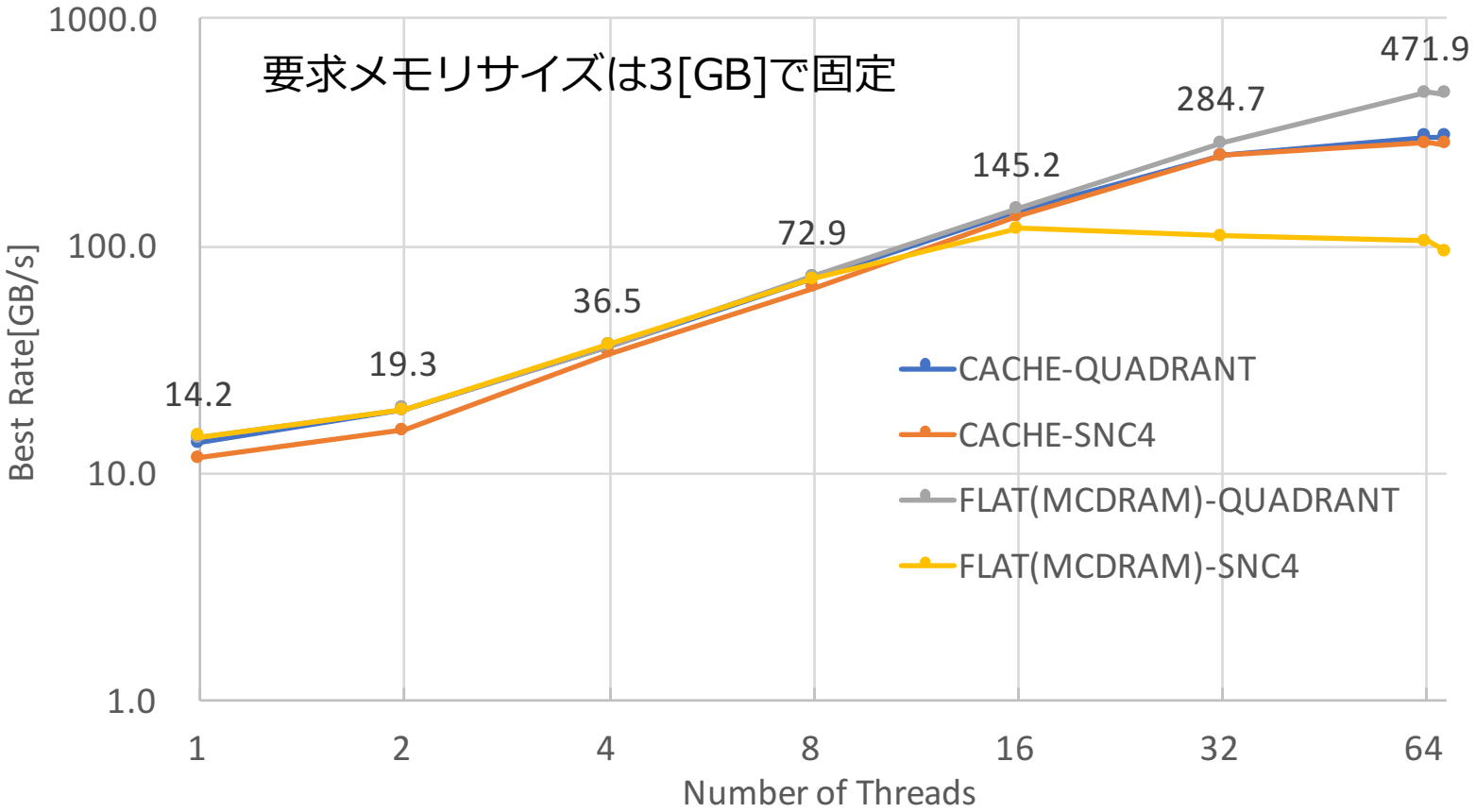

図8 thread数を変化させたときのSTREAM triad実行結果

FLAT(MCDRAM)-QUADRANTにおいて
Intel側が主張するMCDRAMの実性能値[7](475～490[GB/s])に近い値が確認できた

[7] Karthik Raman, *Optimizing Memory Bandwidth in Knights Landing on Stream Triad*, 2016-6-20.

STREAM triad実行時の演算速度算出

次に，STREAM triad実行時の演算速度から評価を行う

表2 STREAM triadのカーネル

```
#pragma omp parallel for
for (j=0; j<STREAM_ARRAY_SIZE;  j++)
                a[j] = b[j]+scalar*c[j];
#endif
```

要求メモリサイズ3[GB]
STREAM_ARRAY_SIZE ＝ 134,217,728　より

総演算数は
2×134,217,728 [Flop]≒0.27[Gflop]

$$\text{STREAM triad実行時の演算速度[Gflops]} \fallingdotseq \frac{0.27[\text{Gflop}]}{\text{elapsed time[sec]}}$$

STREAM triad実行時の演算速度評価

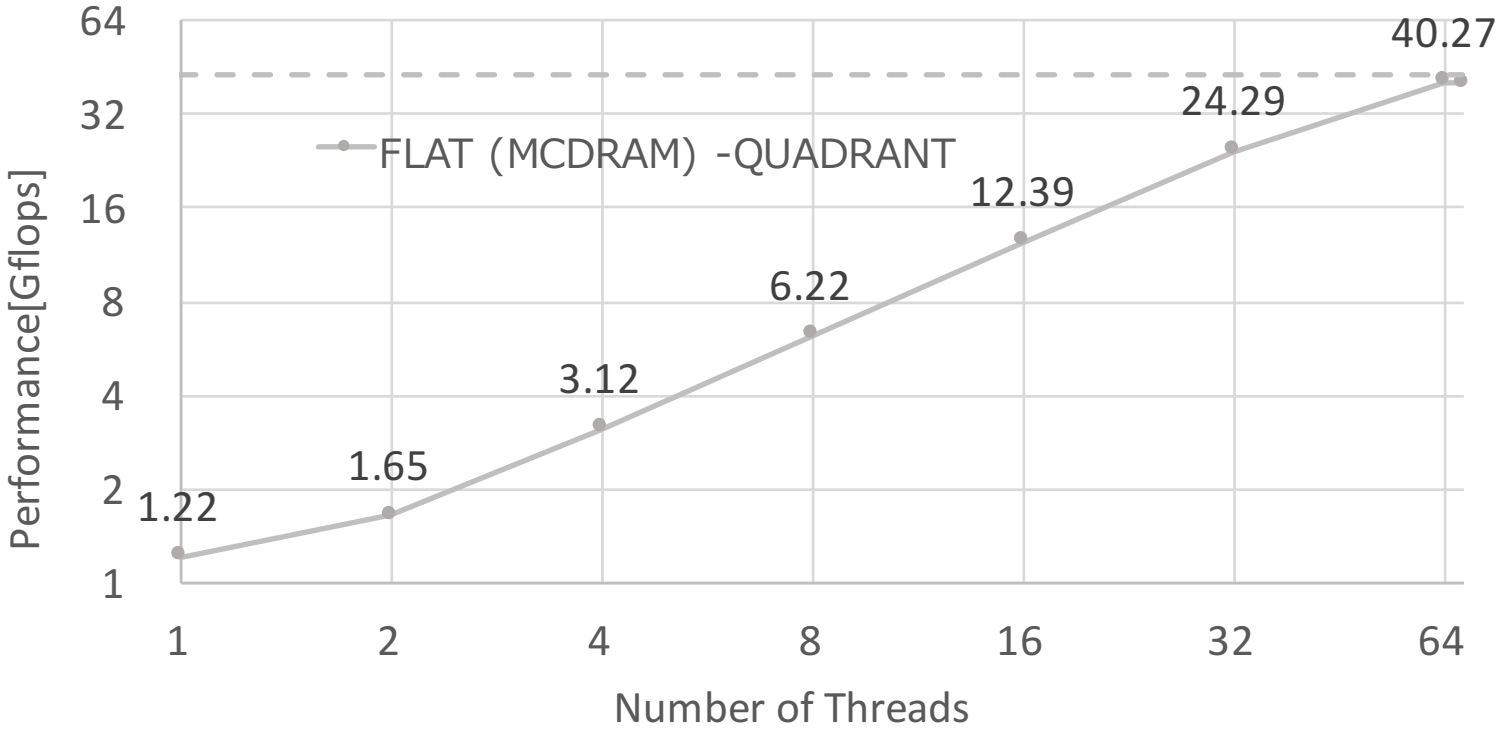

図9 STREAM triad実行時の演算速度

表4 OFPの理論演算性能と
STREAM triadの対ピーク性能, 演算速度限界値

	Oakforest-PACS	
	DDR4	MCDRAM
理論演算性能[Gflops]	3046.4	3046.4
STREAM triadの 対ピーク性能[%]	0.319	1.41
STREAM triadの 演算速度限界値[Gflops]	9.72	43.0

最大で40.27[Gflops] (64 thread)
演算速度限界値の約93.7%

Oakforest-PACSのメモリスループット評価②

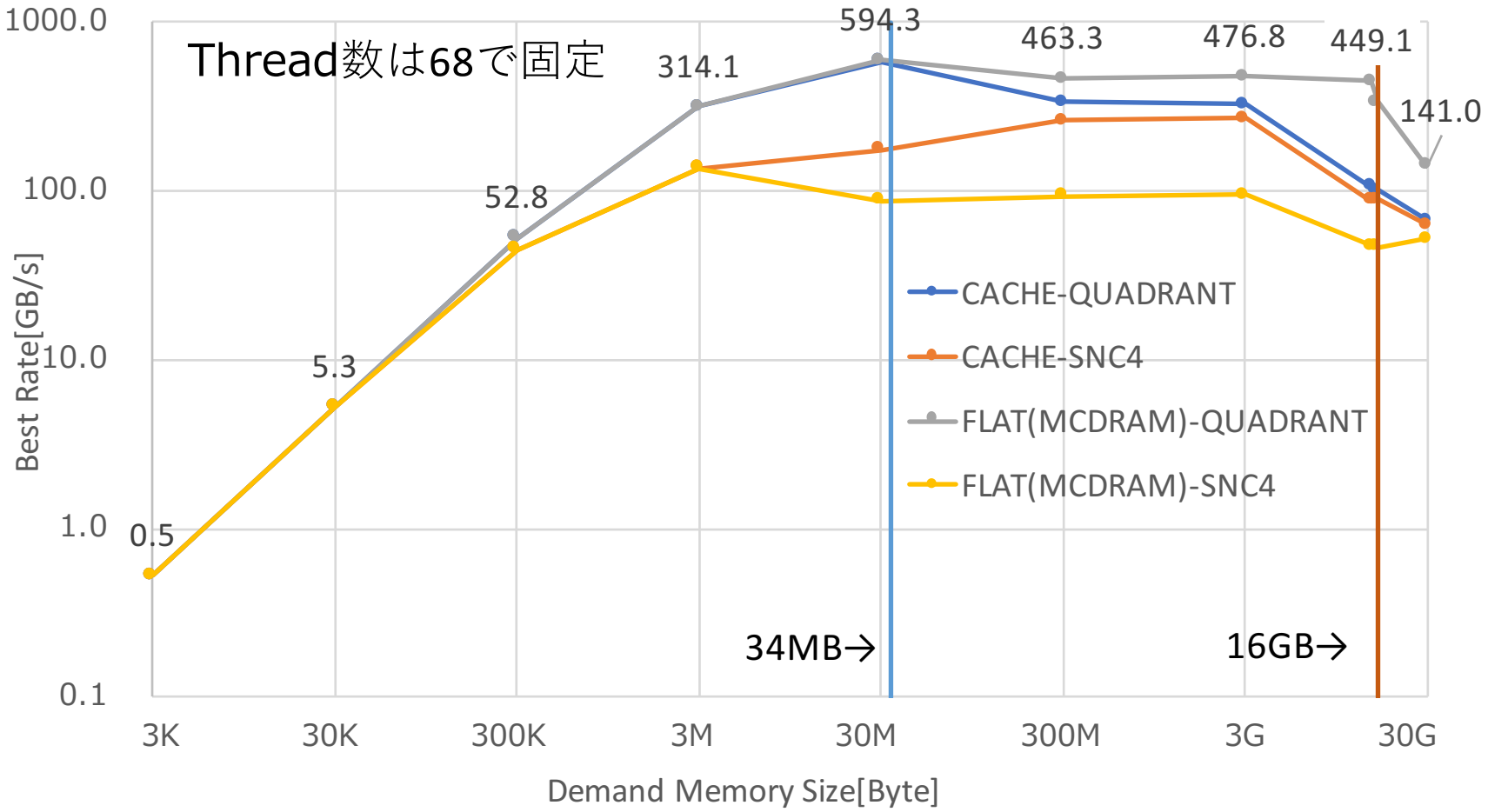

図10 要求メモリサイズを変化させたときのSTREAM Triad実行結果

要求メモリサイズが
16GB以下の場合　⇒　要求メモリサイズに比例して測定値が上昇
16GBを超えた場合 ⇒　DDR4が使われるため，性能が下降

Oakforest-PACSのメモリスループット評価③

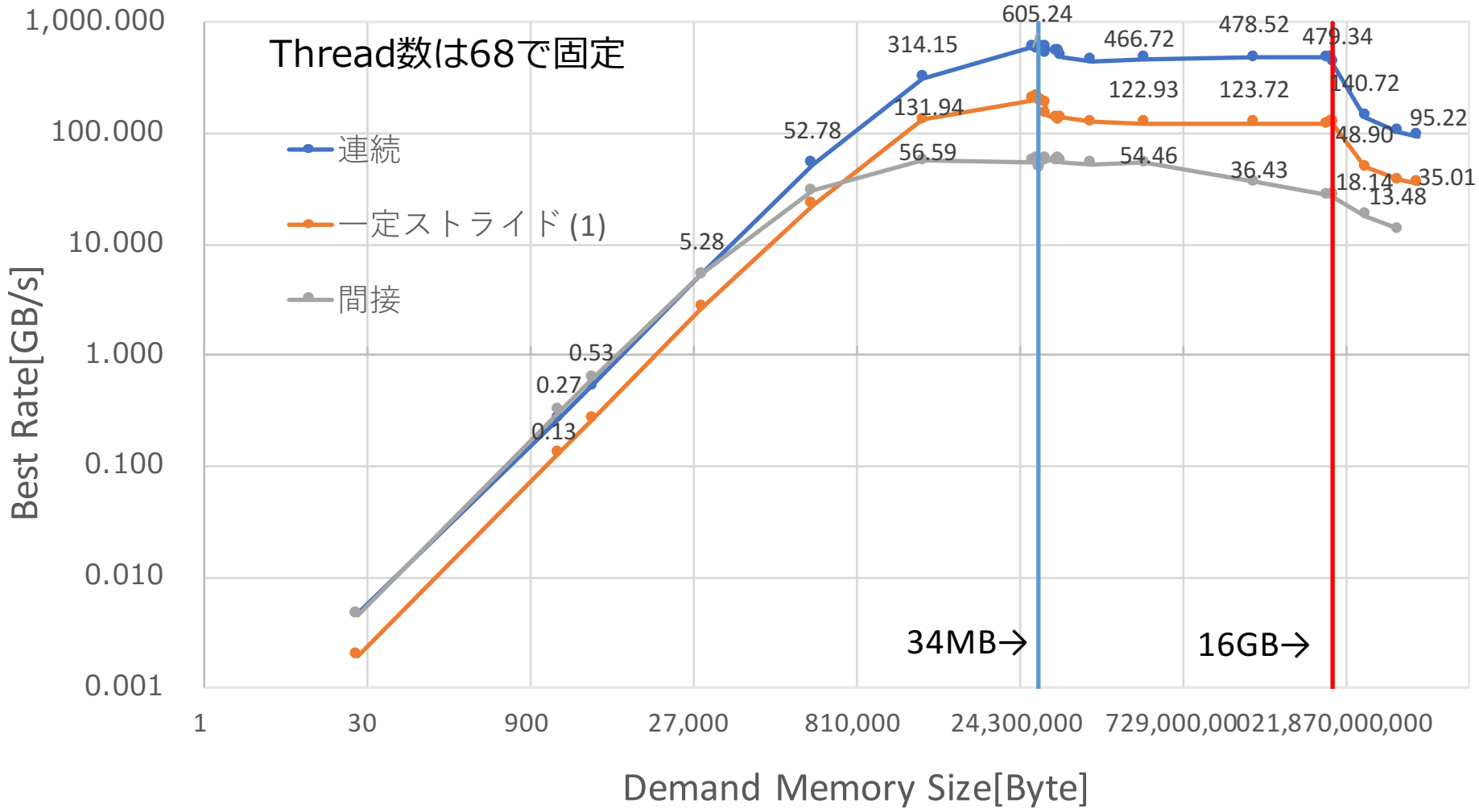

図11 配列要素へのアクセス方法を変化させたときのメモリ性能

配列要素へのアクセスが連続でない時，メモリ性能値がさがる

Oakforest-PACSのメモリスループット評価④

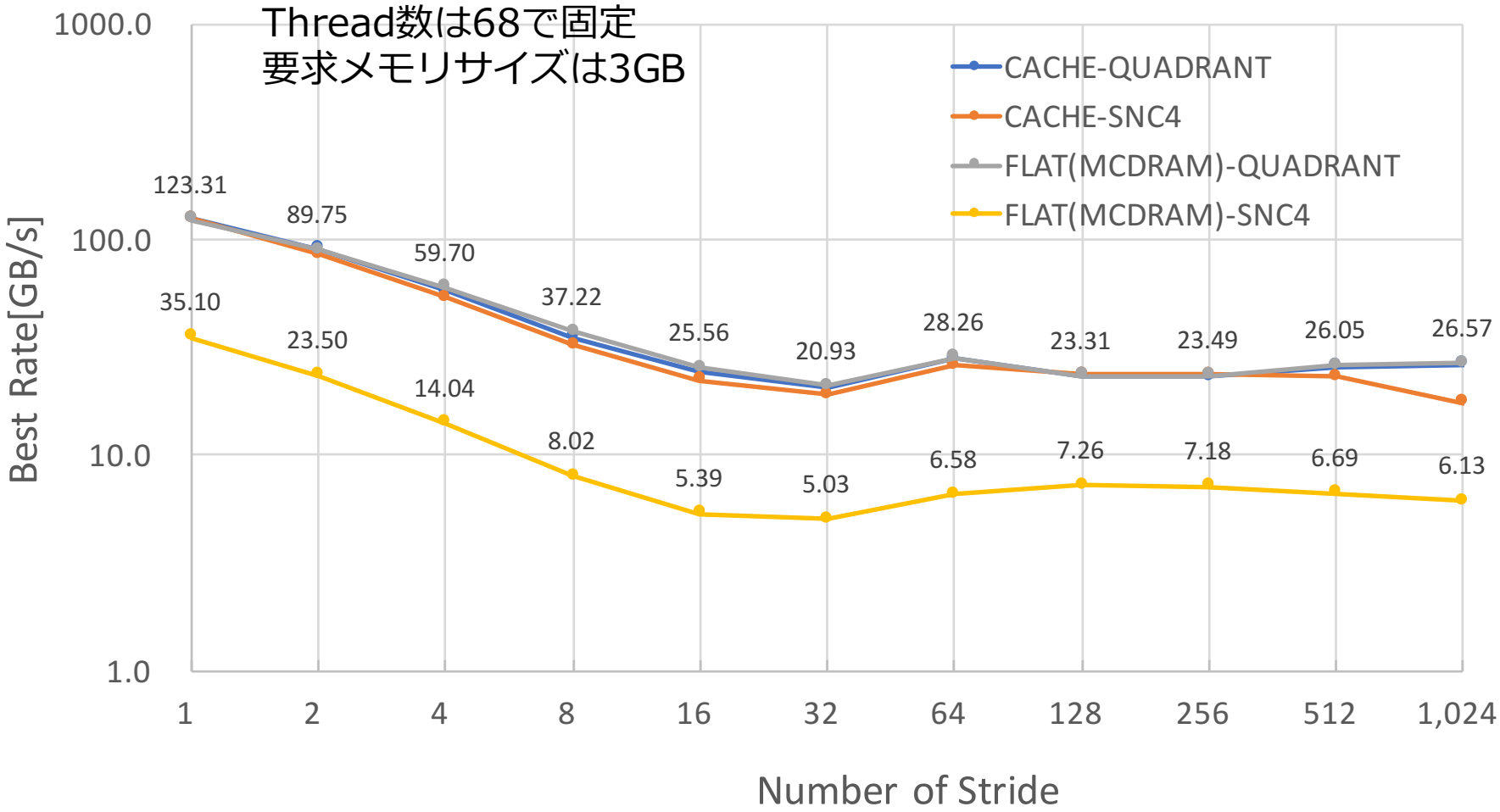

図12 ストライド数を変化させたときのメモリ性能

ストライド数を32まで増加した時，メモリ性能値は直線的に下がる

FrontISTRによる性能評価

FrontISTR 実行例： Solid-100

使用コンパイラ: Intel Fortran compiler 17.0.1

コンパイルオプション

```
-O2 –qopenmp -xMIC-AVX512 -lmemkind
-mcmodel=medium -shared-intel
-qopt-streaming-stores=always
```

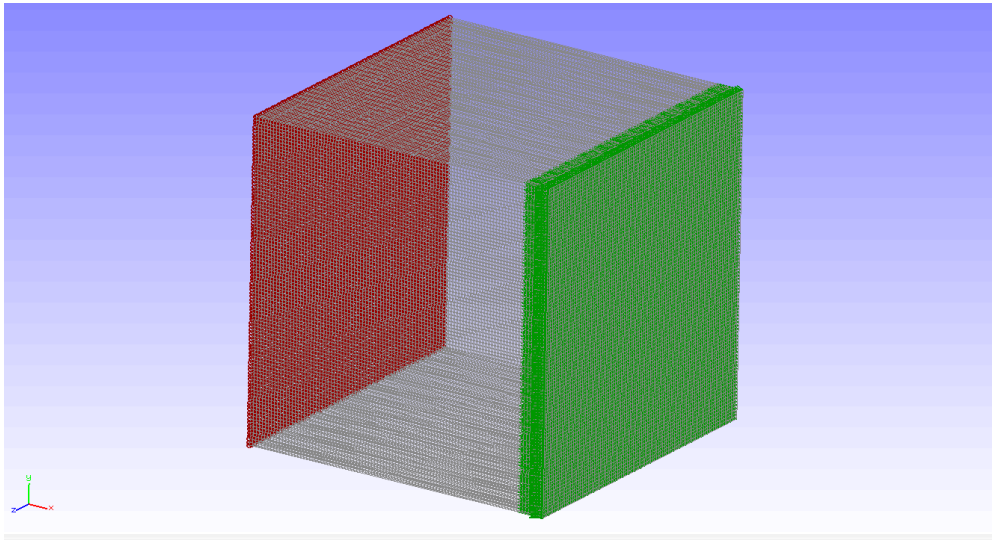

図13 Solid-100のメッシュ図

節点数: 1,000,000
要素数: 970,299
自由度数: 3,000,000
共役勾配法の前処理: 対角スケーリング
必要メモリサイズ (全体): 2.15[GB]
必要メモリサイズ (ベクトル): 22.9[MB]

クラスタモードは
QUADRANTに固定

FrontISTR
実行の流れ

(MPI用前処理 入力メッシュファイル分割)

↓

ファイル入力

↓

全体剛性行列の作成

↓

solver/precond
solver/matvec
solver/communication

解が収束する
まで反復

↓

解析結果ファイル出力

SpMVの要求Byte/Flop

表5 FrontISTRのホットスポット(CSR形式によるSpMV)

```
do blockNum  = 0 , numOfBlockPerThread  - 1
 blockIndex = blockNum * numOfThread  + threadNum
 do i = startPos(blockIndex), endPos(blockIndex)
   X1= X(3*i-2)
   X2= X(3*i-1)
   X3= X(3*i  )
   YV1= D(9*i-8)*X1 + D(9*i-7)*X2 + D(9*i-6)*X3
   YV2= D(9*i-5)*X1 + D(9*i-4)*X2 + D(9*i-3)*X3
   YV3= D(9*i-2)*X1 + D(9*i-1)*X2 + D(9*i  )*X3
   jS= indexL(i-1) + 1
   jE= indexL(i  )
    do j= jS, jE
      in  = itemL(j)
      X1= X(3*in-2)
      X2= X(3*in-1)
      X3= X(3*in  )
      YV1= YV1 + AL(9*j-8)*X1 + AL(9*j-7)*X2 + AL(9*j-6)*X3
      YV2= YV2 + AL(9*j-5)*X1 + AL(9*j-4)*X2 + AL(9*j-3)*X3
      YV3= YV3 + AL(9*j-2)*X1 + AL(9*j-1)*X2 + AL(9*j  )*X3
    enddo
    jS= indexU(i-1) + 1
    jE= indexU(i  )
    do j= jS, jE
      in  = itemU(j)
      X1= X(3*in-2)
      X2= X(3*in-1)
      X3= X(3*in  )
      YV1= YV1 + AU(9*j-8)*X1 + AU(9*j-7)*X2 + AU(9*j-6)*X3
      YV2= YV2 + AU(9*j-5)*X1 + AU(9*j-4)*X2 + AU(9*j-3)*X3
      YV3= YV3 + AU(9*j-2)*X1 + AU(9*j-1)*X2 + AU(9*j  )*X3
    enddo
   Y(3*i-2)= YV1
   Y(3*i-1)= YV2
   Y(3*i  )= YV3
 enddo
enddo
```

カーネル部分

SpMVの要求Byte/Flopを求める

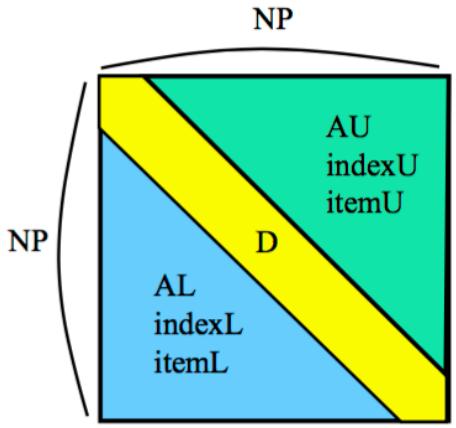

図14 係数行列に関する内部表現[8]

X, Yの要素数は3NP
indexL, indexUの要素数はNP
⇒キャッシュに乗ることが期待できる

疎行列の非ゼロ要素数に合わせて
大きな配列となる
AL, AU, itemL, itemUについて考慮

float型配列(8byte):	AL, AU
Integer型配列(4byte):	itemL, itemU

$$\text{SpMVの要求Byte/Flop} = \frac{(8[\text{Byte}]\times 9+4[\text{Byte}]\times 1)\times 2}{(2\times 9)\times 2} = \frac{76}{18} \approx 4.22$$

[8] FrontISTR研究会, ” HEC-MW における 重要な変数のデータ格納形式”, 2006-04-28.

FrontISTRの対ピーク性能，演算速度限界値

SpMVの要求Byte/Flopが4.22より

- ForntISTRの対ピーク性能[%]

$$= \frac{\text{Hardware Byte/Flop}}{\text{SpMVの要求Byte/Flop}} \times 100 = \frac{\text{Hardware Byte/Flop}}{4.22} \times 100$$

- FrontISTRの演算速度限界値[Gflops]

= 理論演算性能[Gflops]× FrontISTRの対ピーク性能[%] / 100

表6 KNL搭載機材と，FX10, FX100の比較 (FrontISTRの対ピーク性能，演算速度限界値)

	Oakforest-PACS		FX10	FX100
	DDR4	MCDRAM		
理論演算性能[Gflops]	3046.4	3046.4	236.5	1126.4
理論メモリ速度[GB/s]	115	490	85	240
Hardware Byte/Flop	0.0377	0.161	0.359	0.213
FrontISTRの対ピーク性能[%]	0.893	3.82	8.51	5.05
FrontISTRの演算速度限界値[Gflops]	27.2	116	20.1	56.9

FrontISTR DDR4とMCDRAMの性能比較

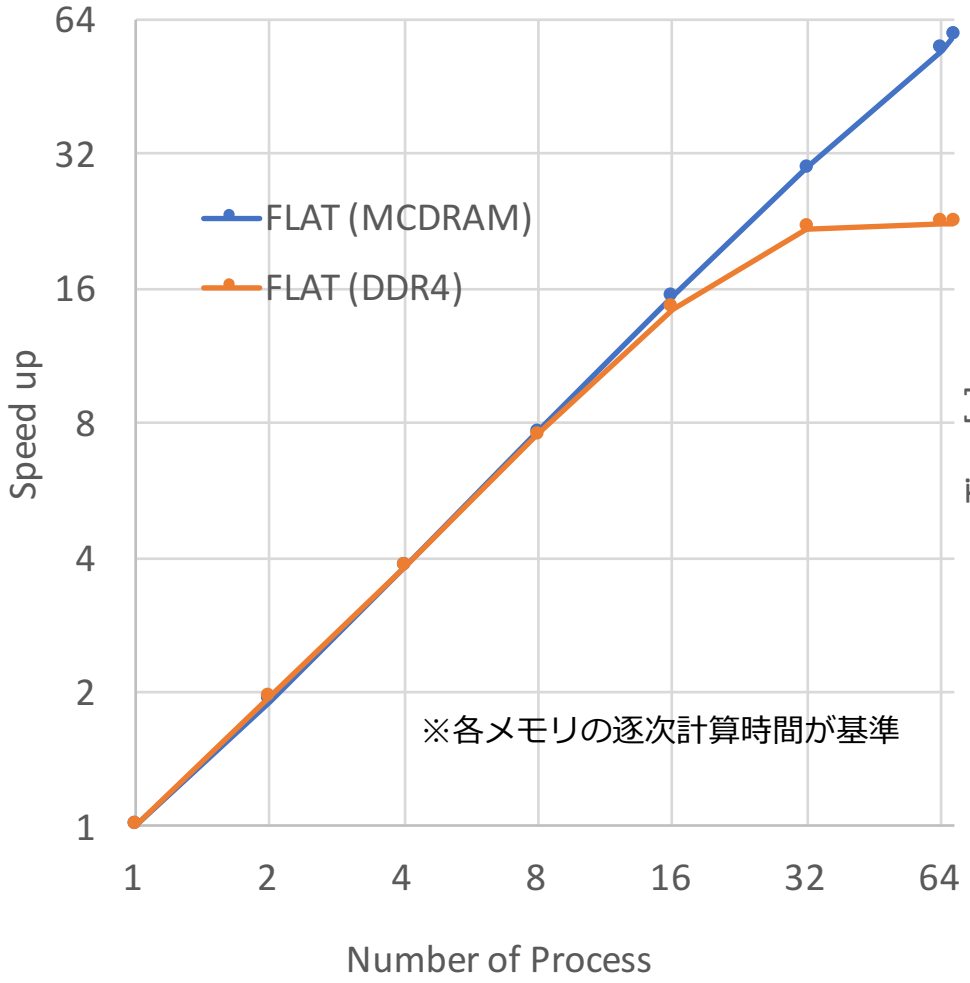

図15-1 MPI並列時のスケーリング性能

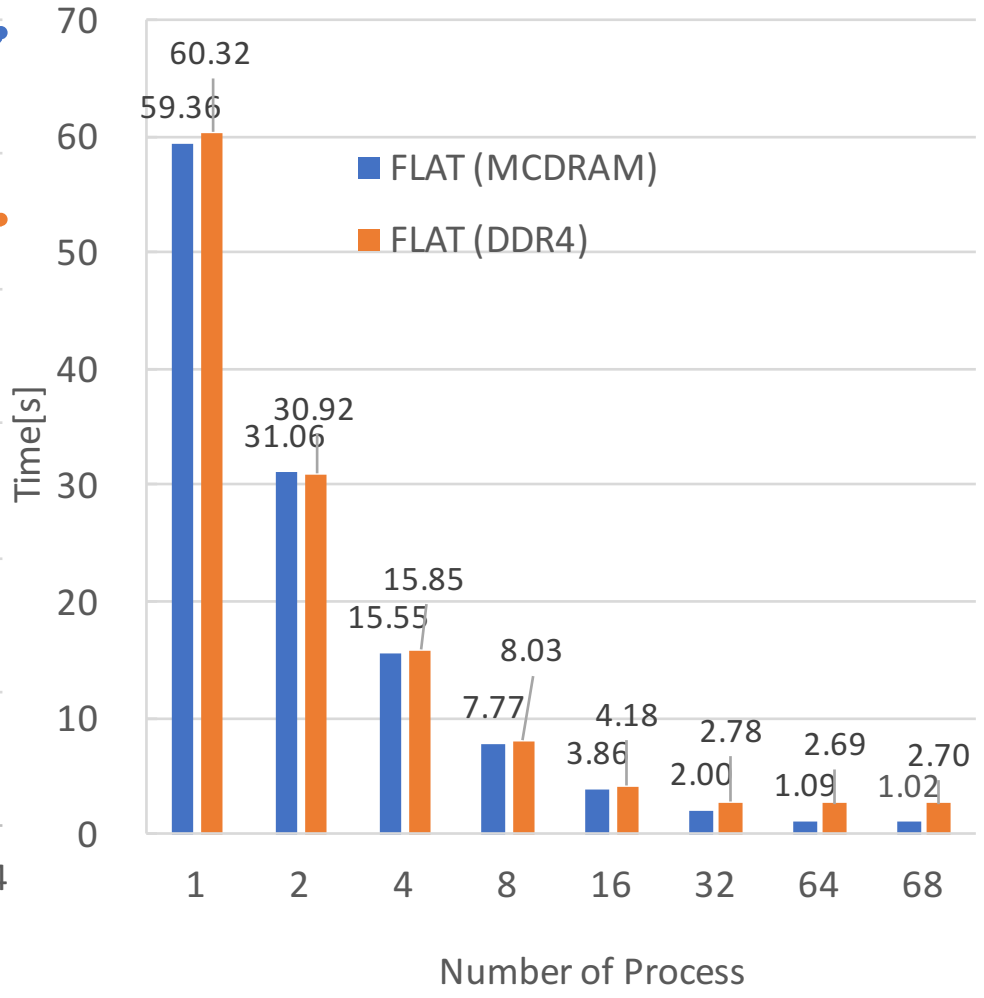

図15-2 MPI並列時の実行時間

DDR4使用時は，32並列で性能向上が頭打ち

FrontISTR メモリモードの性能比較

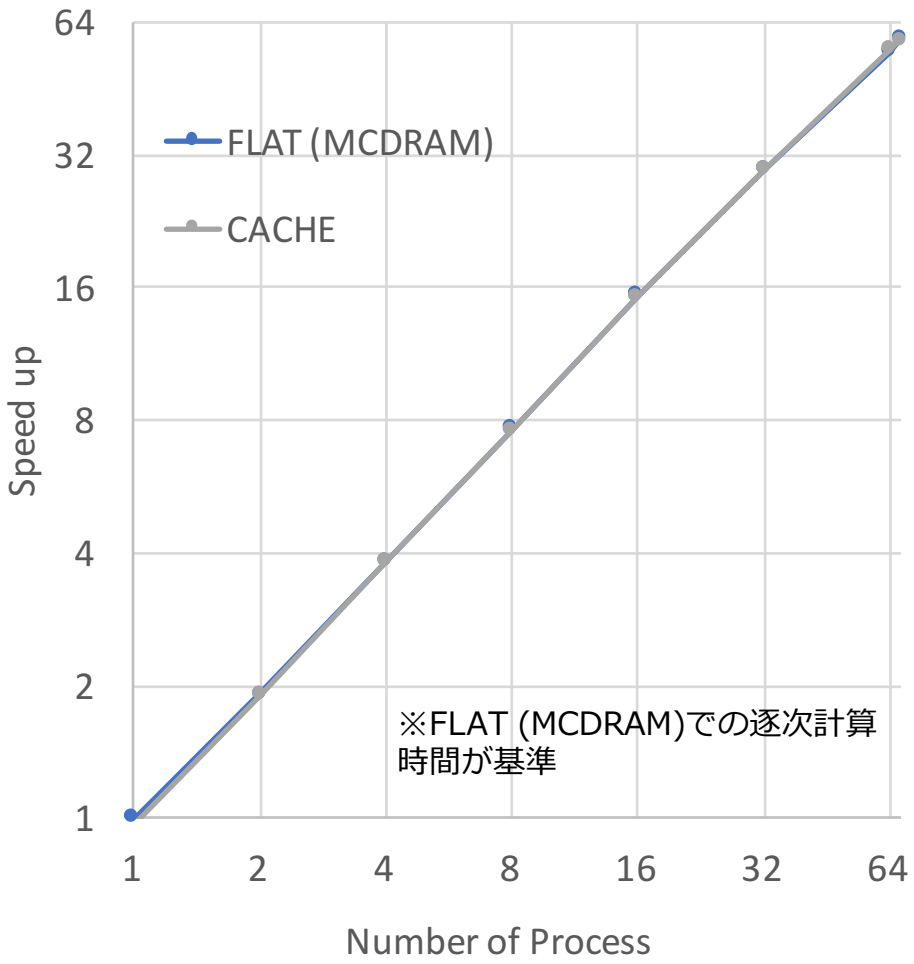

図16-1 MPI並列時のスケーリング性能

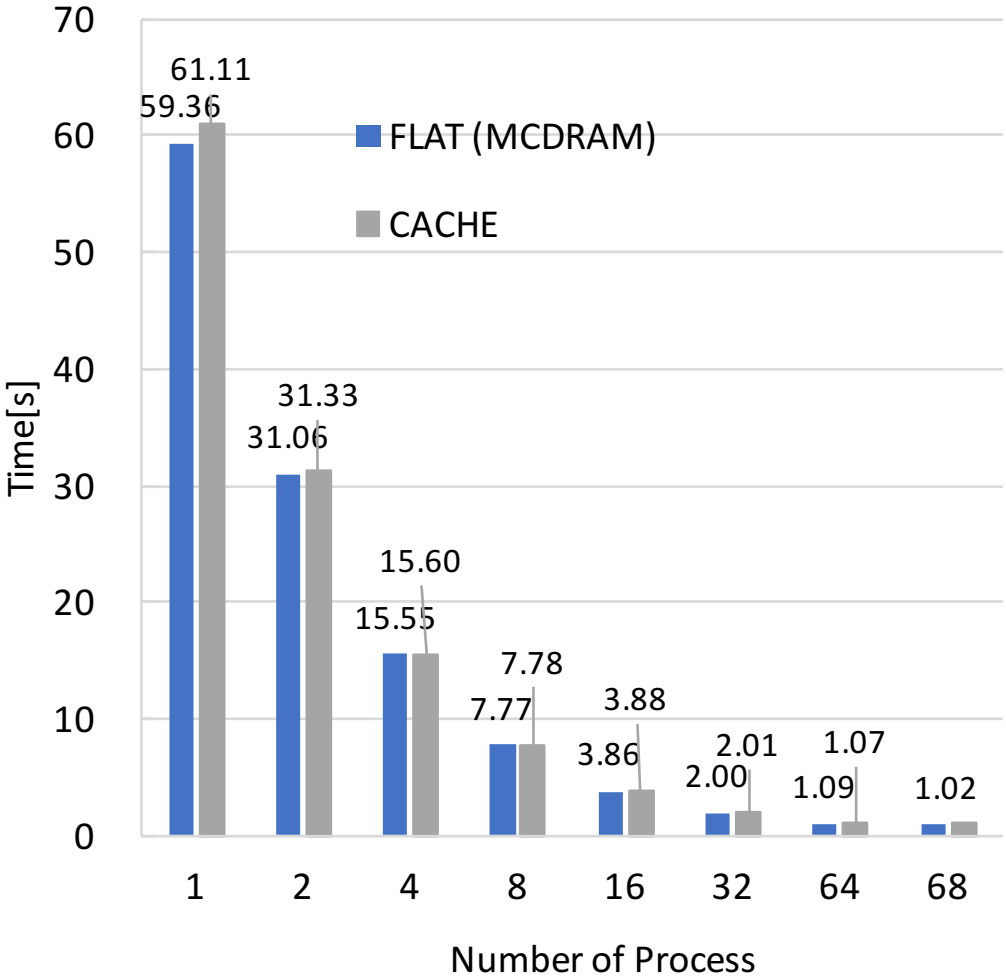

図16-2 MPI並列時の実行時間

わずかな差で，FLAT (MCDRAM) のほうが性能が高い

FrontISTR実行時の演算性能算出

FrontISTRの総演算数は疎行列の非ゼロ要素数(Num of NZ)と
CG法の反復回数(iteration)で決まる

Solid-100では

Num of NZ	256,172,328
iteration	100に固定

非ゼロ要素１つに2flop(積和演算)が行われるので

総演算数[flop] = (Num of NZ)×2×iteration

よって

$$\text{FrontISTR実行時の演算性能[Gflops]} = \frac{(\text{Num of NZ})\times 2\times \text{iteration}}{\text{solver}_{\downarrow}\text{matvec time[sec]}}$$

FrontISTR実行時の演算性能

表7 OFPの理論演算性能とFrontISTRの対ピーク性能, 演算速度限界値

	Oakforst-PACS	
	DDR4	MCDRAM
理論演算性能[Gflops]	3046.4	3046.4
FrontISTRの 対ピーク性能[%]	0.893	3.82
FrontISTRの 演算速度限界[Gflops]	27.2	116

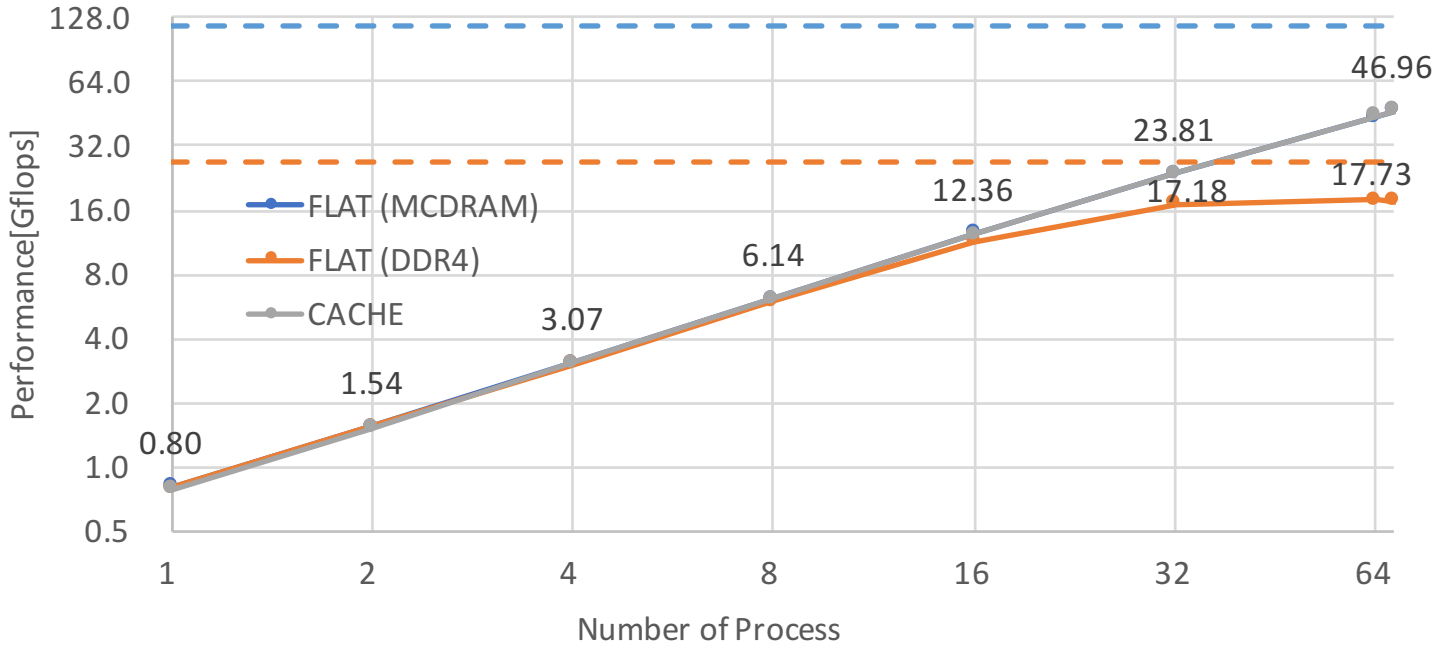

図17 MPI並列時の演算性能

FrontISTRの演算速度限界値に対し，最大で
- FLAT (DDR4): 約65.18%
- FLAT (MCDRAM): 約40.48%

出ている

まとめ

KNLはハードウェアとしてDDR4，MCDRAMを搭載したCPU
メモリモード，クラスタモードという合計９通りの使い方がある

メモリモード
(DDR4，MCDRAMの使い方を決める)

- Flat
- Cache
- Hybrid (※)

クラスタモード
（タイル，メモリ間の通信方法を決める）

- All to All (※)
- Quadrant/Hemisphere
- SNC(Sub-NUMA Clustering) (※)

(※)Oakforest-PACSでは選択不可

- STREAMによるOakforest-PACS性能測定
  - Flat(MCDRAM)-Quadrantという条件下でintelの主張する475[GB/s][7]に近い性能を確認した
  - メモリアクセスが間接参照のとき，連続参照に比べ約10[%]以下まで性能が下がることを確認した
- FrontISTRによるOakforest-PACS性能測定
  - DDR4では32並列で性能向上が頭打ちなのに対し，MCDRAMは68並列まで性能向上できることを確認した
  - MCDRAM使用時， FrontISTRの演算速度限界値が116[Gflops]なのに対し，実性能値として46.96[Gflops]という結果がでた。割合として約40.48%の性能値である

[7] Karthik Raman, *Optimizing Memory Bandwidth in Knights Landing on Stream Triad*, 2016-6-20.

参考文献

[1] Avinash Sodani, *Intel Xeon Phi Processor "Knights Landing" Architectural Overview,* 2015.

[2] Avinash Sodani, *Knights Landing (KNL):2nd Generation Intel Xeon Phi Processor*, 2015.

[3] Shuo Li, Karthik Raman, Ruchira Sasanka, Andrey Semin, Enhancing Application Performance using Heterogeneous Memory Architectures on the Many-core Platform, 2016.

[4] David Kanter, Knights Landing Details, 2014, http://www.realworldtech.com/knights-landing-details/, (2016-11-21 accessed).

[5] 東京大学情報基盤センタースーパーコンピューティング部門, "第2章 FX10 スーパーコンピュータシステムについて", http://www.cc.u-tokyo.ac.jp/system/fx10/fx10-tebiki/chapter2.html, (2016-11-21 accessed).

[6] 富士通株式会社 次世代テクニカルコンピューティング開発本部, "FUJITSU Supercomputer PRIMEHPC FX100 次世代技術への進化", 2014.

[7] Karthik Raman, *Optimizing Memory Bandwidth in Knights Landing on Stream Triad*, 2016-6-20.

[8] FrontISTR研究会, " HEC-MW における 重要な変数のデータ格納形式", 2006-04-28, http://www.multi.k.u-tokyo.ac.jp/FrontISTR/150424/(2)HECMW_import_variables150424.pdf, (accessed 2016-11-23).

[9] James Jeffers,James Reinders, Avinash Sodani, *Intel Xeon Phi Processor High Performance Programming, Second Edition: Knights Landing Edition*,Morgan Kaufmann, 2016.