Vol.2015-HPC-152 No.18
2015/12/17

次世代 Power システムにおける NVLink と GPU を利用したアプリケーションの性能予測

土井淳[†1]

概要：IBM の次世代 Power システムと NVIDIA の次世代 GPU である Pascal は，NVLink と呼ばれる高速な独自ネットワークで接続される．NVLink は従来の PCI express 接続に比べて数倍のバンド幅を持ち，GPU のワークロードをオフローディングする際に必要となる，ホスト-GPU 間のデータ転送速度が大幅に改善されることが見込まれる．実際の NVLink を搭載したシステムが登場する前に，実アプリケーションがどの程度 NVLink の恩恵を受けることができるのかを知るために，本研究では例として格子 QCD，HPCG ベンチマーク，並列 1 次元 FFT の 3 つのアプリケーションについてそれぞれのアプリケーションの演算とデータ転送の特性から性能予測を行う．

キーワード：GPU, GPGPU, POWER, NVLink, Lattice QCD, HPCG, FFT

1. はじめに

エクサスケールコンピューティングに向けて，GPU, MIC や FPGA などのアクセラレーターを用いたスパコンシステムが電力効率の高さから主流となりつつあるが，アクセラレーターと CPU の間でのデータ転送が，アプリケーションの速度向上のボトルネックとなる場合がある．現在のシステムでは PCI express を用いてアクセラレーターを接続しているが，このバンド幅はメモリバンド幅に比べて遅くアプリケーションの性能を引き出すには最適化の工夫が必要となることが多い．

IBM の Power システム[1]は，米国の次世代スパコンプロジェクトである CORAL[2]に向けて，NVIDIA の Tesla GPU をアクセラレーターとして搭載することが決まっている．POWER プロセッサーと GPU は，新しい高速なインターコネクト技術である NVLink[3]を用いて相互接続される．NVLink は，PCI express に比べて最大で数十倍のバンド幅を持ち，これにより CPU-GPU 間のデータ転送によるボトルネックを解消できる可能性がある．CORAL は 2 世代後のシステムであるが，次世代の Power システムと NVIDIA の Pascal アーキテクチャーの GPU から，NVLink の搭載が予定されている．

本報告では，NVLink 実機導入前に，NVLink によるデータ転送速度の向上がどの程度アプリケーションの性能向上に貢献できるかを知るために，アプリケーションとして，(1)格子 QCD, (2)HPCG ベンチマーク, (3)並列 1 次元 FFT, の 3 つを取り上げ，それぞれのアプリケーションの演算とデータアクセス，データ転送の特性から性能予測を行う．

2. NVLink 概要

NVLink は，NVIDIA 社によって提案されたインターコネクト技術[3]で，GPU チップ間および CPU-GPU 間を高速に接続するための新しいインターフェースである．現在の GPU は，PCI express バスを介して，GPU 間もしくは CPU と GPU 間を接続しデータ転送を行っているが，それとは独立したインターコネクトネットワークを用いることで，より効率の良いデータ転送を行うことができるようになる．NVLink は次世代の GPU である Pascal アーキテクチャーより採用される予定である．また，IBM の POWER プロセッサーは，NVLink を CPU 側で初めて採用する予定であり，これにより，ワークロードオフローディングに必須である CPU-GPU 間のデータ転送速度が飛躍的に向上されることが期待される．

NVLink は，図 1 に示すように，リンクあたり 20GB/s の転送速度を持ち，チップあたり 4 本の双方向リンクを持つ．このように CPU ソケットあたり 1 つの GPU を接続する場合は，80GB/s の双方向通信が可能となる．また，図 2 のようにソケットあたりの GPU 数を増やした構成の場合，GPU 同士も相互に接続したようなネットワークを形成することが可能であり，それぞれの接続に使用したリンクの数の分の転送速度で双方向通信が可能である．POWER プロセッサー以外の CPU の場合，CPU-GPU 間の NVLink 接続はできないため，従来通り PCI express での接続となるが，ソケットあたり複数の GPU を構成する場合，GPU 間の相互接続は NVLink によって高速なインターコネクトネットワークを構成することが可能である．

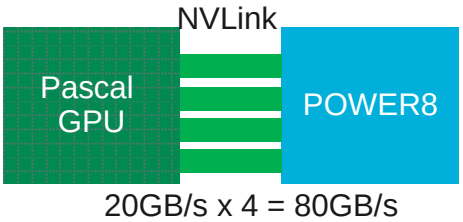
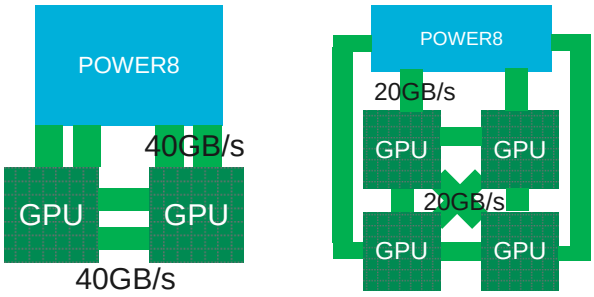


図 1　NVLink による POWER8 プロセッサーと Pascal GPU の高速接続

†1 日本アイ・ビー・エム株式会社　東京基礎研究所
IBM Research – Tokyo



(a) 2GPU 構成　　(b) 4GPU 構成

図 2　ソケットあたり 2GPU および 4GPU 構成にした場合のインターコネクトネットワークと転送速度

3. 格子 QCD の性能予測

3.1 格子 QCD 概要

格子 QCD（Quantum Chromodynamics）は強い力の場の理論を離散化してコンピュータシミュレーションで解ける形にしたもので，世界中の多くのスパコンシステムにおいて実行されるアプリケーションの一つである．格子 QCD は，様々な物理現象を再現したり理論を証明したりするのに役立っており，計算機能力の向上により，未解明や未発見の事象についてのコンピューター上での再現が望まれている．

格子 QCD は 4 次元時空間を格子上に離散化した問題として扱い，格子点間の相互作用を用いて CG 法などを用いて線形方程式を解く．相互作用を計算する方法はいくつかあるがここでは良く使われる Wilson-Dirac 演算子を用いる．Wilson-Dirac 演算子は，式 1 に示されるような形で，スピノル場について，ゲージ場の影響を，隣接格子点の 4x3 のスピノル行列と 3x3 のゲージ行列を乗じることで求める 9 点ステンシル計算である．

$$D(n)=\delta(n)-\kappa\cdot\sum_{\mu=1}^{4}\left\{(1-\gamma_\mu)U_\mu(n)\delta(n+\hat{\mu})+(1+\gamma_\mu)U_\mu^{\dagger}(n-\hat{\mu})\delta(n-\hat{\mu})\right\} \quad (1)$$

また，SU(3)対象性を利用して，4x3 のスピノル行列は，半分のハーフスピノルと呼ばれる 2x3 行列に変換することでゲージ行列の乗算と，分散メモリ並列化時の境界部分の交換時のデータ転送を半分にできることが知られている．式 2 は X 軸の正の方向の例を示している．

$$(1-\gamma_1)U_1(n)\delta(n+\hat{1},m)=\begin{pmatrix}U_1(n)\cdot(s_1+i\cdot s_4)\\U_1(n)\cdot(s_2+i\cdot s_3)\\-i\cdot U_1(n)\cdot(s_2+i\cdot s_3)\\-i\cdot U_1(n)\cdot(s_1+i\cdot s_4)\end{pmatrix}=\begin{pmatrix}U_1(n)\cdot h_1\\U_1(n)\cdot h_2\\-i\cdot U_1(n)\cdot h_2\\-i\cdot U_1(n)\cdot h_1\end{pmatrix} \quad (2)$$

Wilson-Dirac 演算子の分散メモリ並列化では，4 次元格子を各プロセスに分割して処理を行う．分割された格子の境界部分では，互いのプロセス間で境界部分のデータを交換する必要がある．GPU に処理をオフロードする場合，境界部分のデータ交換時に，CPU-GPU 間でデータ転送が必要となる．

3.2 Wilson-Dirac 演算子の実行モデル

あらかじめすべての必要なデータが GPU のメモリ上にあるとき，GPU を使用して Wilson-Dirac 演算子を計算するときの GPU 上での計算と CPU-GPU 間およびプロセス間のデータ転送の流れを図 3 に示す．

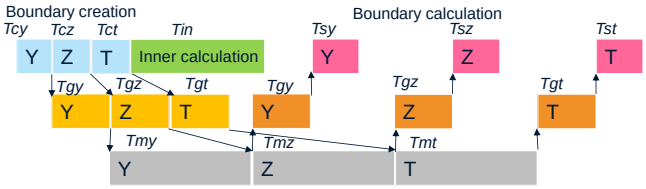


図 3 Wilson-Dirac 演算子の計算とデータ転送の流れ．上から GPU 上の計算時間，CPU-GPU 間の転送時間，MPI 通信時間を表し各段で依存性の無い処理は重ね合わせられる．矢印はデータの流れを表す

図 3 では，Y,Z,T 軸方向に格子を分割した場合を考える．それぞれの軸について，正負の両方向とも処理，転送が必要であるがここでは 1 つにまとめて考える．各軸ごとに次の 3 つのステップで処理を行う．

(1) 境界部分についてハーフスピノルを生成する
(2) 境界部分を CPU に転送し MPI で通信し，GPU に戻す
(3) 境界部分のハーフスピノルを使って計算する

また，これらの処理と平行して中心部分の格子について Wilson-Dirac 演算子を計算する．

モデルを単純にするため，それぞれの処理同士は同時に行わないものとし，図 3 の順番通りに処理するとする．(ただし，CPU-GPU 間の転送については，逆方向の転送同士は同時に実行できるとする) このとき，Wilson-Dirac 演算子の全体の実行時間は，図 3 に示すそれぞれの時間を用いて，次の式で求めることができる．

$$\begin{aligned}
T_{y1} &= T_{cy}+T_{gy}\\
T_{z1} &= max(T_{y1},T_{cy}+T_{cz})+T_{gz}\\
T_{t1} &= max(T_{z1},T_{cy}+T_{cz}+T_{ct})+T_{gt}\\
T_{y2} &= T_{y1}+T_{my}+T_{gy}\\
T_{z2} &= max(T_{y2},T_{z1}+T_{mz})+T_{gz}\\
T_{t2} &= max(T_{z2},T_{t1}+T_{mt})+T_{gt}\\
T_{y3} &= max(T_{y2},T_{cy}+T_{cz}+T_{ct}+T_{in})+T_{sy}\\
T_{z3} &= max(T_{y3},T_{z2})+T_{sz}\\
T_{Dirac} &= max(T_{z3},T_{t2})+T_{st} \quad (3)
\end{aligned}$$

それぞれの時間は，計算処理については，ルーフラインモデル[4]を用いて，計算量とデータアクセスの比率から経過時間を予測する．図 4 に，GPU におけるルーフラインモデルをプロットしたものを示す．次世代 GPU である Pascal の詳細な性能値は 2015 年 11 月時点では確定していないので，推定値として現行機種である Tesla K40 に対して，演算性能を 3 倍，メモリバンド幅を 4 倍としてモデルを設定した．

また，CPU-GPU 間，MPI によるノード間のそれぞれのデータ転送時間は，それぞれのピークバンド幅の 80%を有効バンド幅として，転送するデータ量から時間を予測値とし

て求める．これらは残りのアプリケーションの性能予測においても同様である．

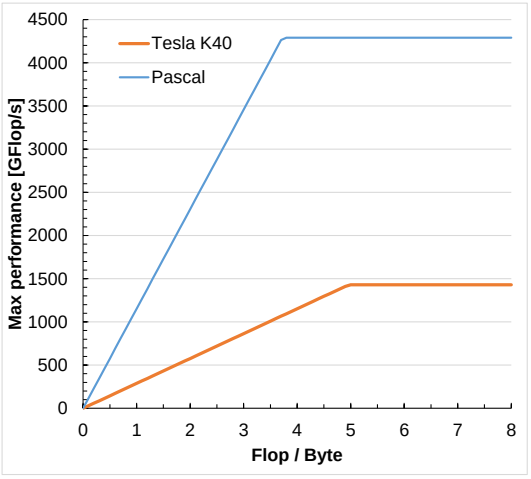


図 4　GPU のルーフラインモデル（倍精度），現行機種 Tesla K40 と，次世代機種である Pascal（予想値）を比較

Wilson-Dirac 演算子の計算時間をルーフラインモデルで求めるにあたり，境界部分のハーフスピノル生成，内部の演算，境界部分の演算それぞれについて，格子点あたりの計算量を表 1 に，データアクセス量を表 2 にそれぞれまとめる．なお，ここでは，理論値ではなく実際の実装上の数値をまとめている．これは，ゲージ行列の SU(3)対称性を利用することで，3x3 のゲージ行列のうち，3x2 成分のみをロードし，実行時に残りの成分を演算で求めることで，データアクセスを減らす手法を用いているためである．これにより，データアクセスを減らせる代わりに，42 flops の計算量が増えているが，GPU では計算量の方が圧倒的に高速であるので結果として性能が向上する[5].

表 1 Wilson-Dirac 演算子の格子点あたり計算量

[flops]	(X),Y,Z 軸	T 軸
ハーフスピノル生成	正: 12 負: 186	正: 0 負: 174
内部の演算 (正+負)	420	372
境界部分の演算	正: 198 負: 24	正: 186 負: 12

表 2 Wilson-Dirac 演算子の格子点あたりデータアクセス量

[bytes]	(X),Y,Z 軸	T 軸
ハーフスピノル生成	正: 288 負: 384	正: 192 負: 288
内部の演算 (正+負)	768	576
境界部分の演算	正: 384 負: 288	正: 288 負: 192

また，境界部分のデータ転送に必要なデータ量は，格子点あたり各方向毎に 96 バイトである．これに各方向の境界部分の体積を乗じたものが転送されるデータの総量となる．

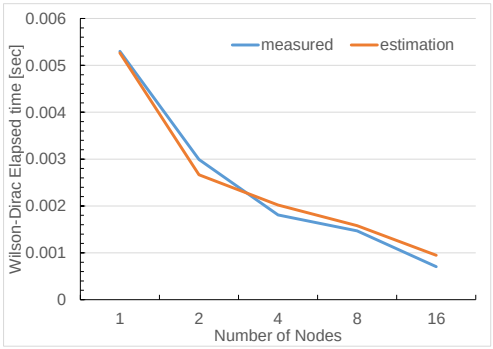
3.3 実機による性能測定と性能予測の比較

性能予測モデルがどの程度正確かを知るために，実機を用いて予測値と実測値を比較する．本報告で使用するクラスターの仕様を表 3 にまとめる．以降のアプリケーションについても同様のクラスターを使用する．

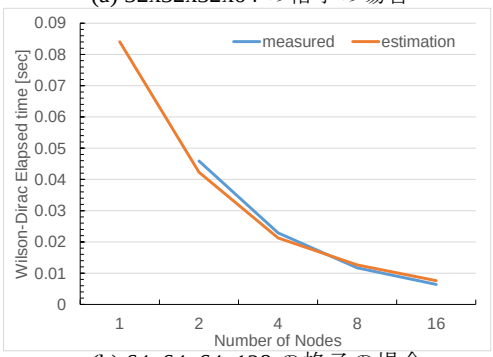
表 3 実測に使用するクラスター

モデル名	Lenovo NeXtScale nx360 M4
CPU	Intel Xeon E5-2667 v2
ノードあたりソケット数	2
GPU	NVIDIA Tesla K40
ノードあたり GPU 数	2
ネットワーク	Infiniband FDR

Wilson-Dirac 演算子の性能測定では，BiCGStab 法を用いた線型方程式ソルバー内で呼ばれる Wilson-Dirac 演算子について，平均実行時間を測定した．ここでは，演算には GPU のみを使用し，GPU1 枚当たり 1 つの MPI タスクを割り当てることとする．ここでは，CPU-GPU 間のデータ転送には PCI express gen.3 を利用し，バンド幅は 16GB/s とする．また，ノード間は Infiniband FDR で接続され，バンド幅は 7GB/s とする．この条件で性能予測を行い，実測値と Strong scaling について比較した結果を図 5 に示す．



(a) 32x32x32x64 の格子の場合



(b) 64x64x64x128 の格子の場合

図 5 Tesla K40 をノードあたり 2 枚搭載した Infiniband FDR 接続のクラスターにおける Wilson-Dirac 演算子の性能予測値と実測値の演算時間の比較（Strong scaling）

格子サイズが比較的小さい(a)の場合，ノード数を増やし

ていくと性能が頭打ちして下がっていく様子が性能予測値でも再現されており，実測値でもほぼ同じような結果が得られていることが分かる．この結果から，この性能予測モデルの再現度は比較的高いと分かる．

3.4 Wilson-Dirac 演算子の NVLink による性能向上予測

次世代 GPU である Pascal アーキテクチャーを用いた場合の Wilson-Dirac 演算子の実行時間の予測値について，POWER プロセッサーを用いて NVLink 接続を行う場合と，それ以外のプロセッサーを用いて PCI express（gen.3）接続を行う場合を比較した．また，ノード間の接続はこちらも 1 世代進化させて Infiniband EDR 接続を想定する．このとき，ノードあたり 1 つの Infiniband カードの場合と 2 枚使用する場合のそれぞれについて比較した．

ここでは，ノードあたり 2 つの CPU と 2 枚の GPU を搭載し，NVLink は CPU と GPU 間のみを接続するとする．つまり CPU-GPU 間の転送速度は 80GB/s とする．Infiniband EDR によるノード間の通信は，12.5GB/s とし，2 枚挿しの場合は 25GB/s とする．Wilson-Dirac 演算子の実行時間の予測値について，16 ノード，32GPU を使用した場合について，2 つのサイズの格子についての比較をそれぞれ表 4 および表 5 にまとめる．

表 4 Wilson-Dirac 演算子の実行時間予測（32x32x32x64）

	Tesla K40 Infiniband FDR	Pascal Infiniband EDR		Infiniband EDR x2	
	PCIe	PCIe	NVLink	PCIe	NVLink
Time[ms]	0.95	0.57	**0.49**	0.33	**0.26**
Speed up [%]			**14.94**		**28.62**

表 5 Wilson-Dirac 演算子の実行時間予測（64x64x64x128）

	Tesla K40 Infiniband FDR	Pascal Infiniband EDR		Infiniband EDR x2	
	PCIe	PCIe	NVLink	PCIe	NVLink
Time[ms]	7.58	4.54	**3.95**	2.65	**2.06**
Speed up [%]			**14.94**		**28.62**

Pascal 世代のシステムにおいて Wilson-Dirac 演算子は NVLink によって 15～30%程度の性能向上が得られることが予測される．特にノード間の通信速度を上げると NVLink による性能向上の恩恵が多く得られることが分かる．これは図 3 で，MPI による通信時間が減ると，CPU-GPU 間のデータ転送時間が多く見えるようになるためである．

4. HPCG ベンチマークの性能予測

4.1 HPCG ベンチマーク概要

HPCG ベンチマーク[6]は，スパコンシステムの性能を比較するための新しいベンチマークであり，将来的には TOP500[7]における LINPACK ベンチマークと置き換わる指標として使用される可能性のあるベンチマークの一つである．HPCG ベンチマークでは，CG 法を用いて疎行列で構成される線形方程式を解く問題についての実効性能を測定する．具体的には 3 次元構造格子についての 27 点ステンシル計算を疎行列ベクトル積で計算している．HPCG ベンチマークの CG 法では，収束を早めるために前処理を行っており，Gauss-Seidel 法を用いたマルチグリッド法が採用されている．HPCG ベンチマークにおいて計算時間の多くは，このマルチグリッド法による前処理が占める．本報告では，HPCG ベンチマークにおけるマルチグリッド法について，性能予測を行う．

4.2 マルチグリッド法の GPU による実装

HPCG ベンチマークにおけるマルチグリッド法では，Gauss-Seidel 法を採用しているため，そのままでは計算の依存関係から並列化を行うことが困難である．Gauss-Seidel 法の並列化には，(1)超平面を利用する方法，(2)マルチカラー法を用いる方法，の 2 つが良く用いられるが，(1)では並列度が限られるため GPU での計算には向かないため(2)のマルチカラー法を用いる．一般的に 3 次元構造格子における 27 点ステンシルをカラーリングする場合 8 色でカラーリングすると隣接格子点間で必ず違う色に分けられることが知られているが，HPCG ベンチマークでは 3 次元構造格子であることを利用した最適化はできないため，ベクトル要素をノードとし疎行列で接続されたグラフと見なし，グラフ上のノードのカラーリング問題を解くことで，各カラー毎に並列化を行うことで GPU 上で計算を行う．

マルチグリッド法の計算を GPU で行う際，あらかじめ疎行列は GPU のメモリ上に転送しておく．HPCG ベンチマークでは問題サイズを自由に選択できるので，すべてのデータが GPU メモリ上に展開できる大きさを選択する．つまり，CG 法で使用するすべてのベクトルも GPU 上にあるものとし，CPU-GPU 間のデータ転送は，ステンシル計算において，袖領域となる部分に対応するベクトルデータを隣接プロセスと交換する処理を行うときに発生する．

4.3 マルチグリッド法の実行モデル

HPCG ベンチマークにおけるマルチグリッド法の実装では，計算とデータ転送の処理は重ね合わせずに逐次実行するものとする．マルチグリッド法による計算は次に示す擬似コードのように細かいグリッドから粗いグリッドにかけて再帰的に処理を行う．

```
kernel ComputeMG
    if coarsest level then
        ComputeGaussSeidel
    else
        ComputeGaussSeidel
        ComputeSPMV
        ComputeRestriction
        ComputeMG for coarse level
        ComputeProlongation
        ComputeGaussSeidel
    endif
end kernel
```

これらの処理はすべて逐次実行され，マルチグリッド法の実行時間はすべての処理の合計時間となる．なお，HPCG

ベンチマークでは，4 段階のレベルで異なるサイズのグリッドを用いる．これらの処理のうち，ComputeGaussSeidel および，ComputeSPMV において，袖領域の交換を行う．図 6 にこの様子を示す．共に計算を行う前にベクトルについて袖領域を交換する．

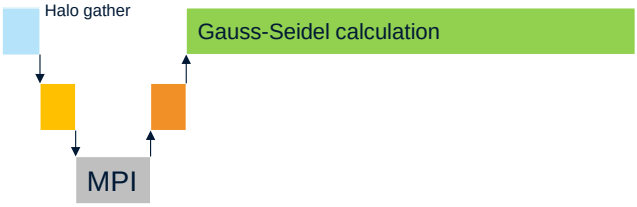


図 6　Gauss-Seidel 法における袖領域交換のモデル．上から GPU 上での処理，CPU-GPU 間のデータ転送，MPI によるプロセス間データ転送を表す．SPMV も同様

また，ルーフラインモデルを用いて性能予測を行うのに必要な，各処理の格子点あたりの計算量とデータアクセス量を表 6 にまとめる．ここでは，27 点ステンシル計算であることを前提に，各格子点あたり 27 成分の疎行列要素を参照するものとする．

表 6 マルチグリッド計算の格子点あたりの計算量とデータアクセス量

	計算量 [flops]	データアクセス [bytes]
ComputeGaussSeidel	104	1088
ComputeSPMV	54	540
ComputeRestriction	1	28
ComputeProlongation	1	20

4.4 実機による性能測定と性能予測の比較

HPCG ベンチマークを実行して，マルチグリッド計算にかかった時間の平均を測定し，性能予測値と比較する．問題サイズは，プロセスあたり 128x128x128 とし，GPU あたり 1MPI タスク（ノードあたり 2MPI タスク）として実行する．図 7 にこのときの実測値と予測値の比較を示す．

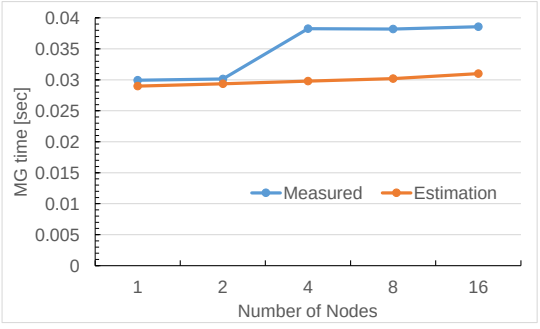


図 7 Tesla K40 をノードあたり 2 枚搭載した Infiniband FDR 接続のクラスターにおける HPCG ベンチマークのマルチグリッド計算の性能予測値と実測値の演算時間の比較（Weak scaling）

4 ノード以上になった場合に，性能劣化が見られる（恐らく実装上の問題と思われる）が予測性能の傾向と実測値の傾向はおおむね一致しているといえる．

4.5 マルチグリッド計算の NVLink による性能向上予測

Pascal 世代の環境において，POWER プロセッサーと Pascal の NVLink 接続と，それ以外のプロセッサーの PCI express 接続を想定した HPCG ベンチマークにおけるマルチグリッド計算の性能予測を行う．格子 QCD の場合と同じ条件で，16 ノードを使用した場合について比較を行ったところ，表 7 のように，NVLink によって 6%程度の性能向上が得られることが分かった．

表 7 マルチグリッド計算の実行時間予測（16 ノード使用，プロセスあたり 128x128x128）

	Tesla K40 Inifiniband FDR PCIe	Pascal Inifiniband EDR PCIe	Pascal Inifiniband EDR NVLink	Pascal Inifiniband EDRx2 PCIe	Pascal Inifiniband EDRx2 NVLink
MG time [sec]	0.0310	0.0087	**0.0082**	0.0083	**0.0078**
Speed up [%]			**6.128**		**6.444**

マルチグリッド計算において，データ転送にかかる時間が計算処理時間に比べて比較的小さいために，性能向上がそれほど大きくは無いという結果になった．

5. 並列 1 次元 FFT の性能予測

5.1 並列 1 次元 FFT 概要

FFT（高速フーリエ変換）は多くの科学技術計算に利用される重要な計算の一つである．大規模な 1 次元 FFT 自体が使用される機会はあまり多くは無いが，HPC Challenge ベンチマーク[8]の一つのベンチマークとして利用され，スパコンの性能を知る上で重要なアプリケーションである．また，並列化された 2 次元や 3 次元 FFT の性能の傾向は並列 1 次元 FFT と近いため，それらの性能を知る上でも重要である．

1 次元 FFT を分散メモリ並列化する場合，6-Step FFT[9]という手法が広く用いられ，HPCC ベンチマークでも採用されている．一般的に 1 次元 FFT を分散メモリ並列化する場合は，1 次元配列を np 個のプロセスで分割して保持するが，FFT では全体の配列を参照する必要があるため，各プロセスで効率良くデータを参照できるようにするために，1 次元配列を 2 次元配列とみなしてそれぞれの軸方向に FFT 計算を行う．

式(4)は通常の 1 次元 FFT を表すが，これを式(5)のように 2 次元配列の形に置き換える．式(5)は，3 回の配列の転置，2 回の FFT 計算，1 回のひねり係数の乗算の 6 ステップで計算を行うことができ，これを並列化すると図 8 に示すように，転置の部分に 3 回の全対全通信が必要になる．

$$y_k = \sum_{j=0}^{n-1} x_j w_n^{jk} \quad , \quad w_n = e^{-2\pi i/n} \tag{4}$$

$$y(k_2,k_1) = \sum_{j_1=0}^{n_1-1}\sum_{j_2=0}^{n_2-1} x(j_1,j_2) w_{n_2}^{j_2 k_2} w_{n_1 n_2}^{j_1 k_2} w_{n_1}^{j_1 k_1} \tag{5}$$

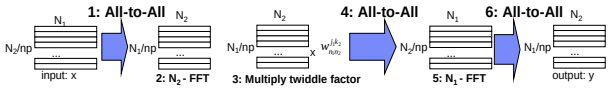


図 8　6-step FFT による並列 1 次元 FFT の計算

5.2 GPU による並列 1 次元 FFT の実装

ここでは，FFT 計算はすべて GPU 上で行うとする．あらかじめ GPU のメモリ上に FFT を行う分散された 1 次元配列がある状態で処理を始めるものとし，分散配列全体が GPU メモリ内に収まるサイズであるとする．また，ひねり係数の入った配列も同様に GPU のメモリ上にあらかじめ置く．この状態で，全対全通信の前後に CPU-GPU 間のデータ転送が発生する．

ローカルな FFT 計算には，cuFFT[10]を利用し，それ以外の転置などの処理は別途 GPU 用のカーネルプログラムを用意する．

5.3 並列 1 次元 FFT の実行モデル

本報告では，6-step FFT の処理は逐次行うものとし，データ転送と計算は重ね合わせないものとする．このときの 6-step FFT の処理の流れを図 9 にまとめる．並列 1 次元 FFT の実行時間は，これらの時間をすべて足したものになる．

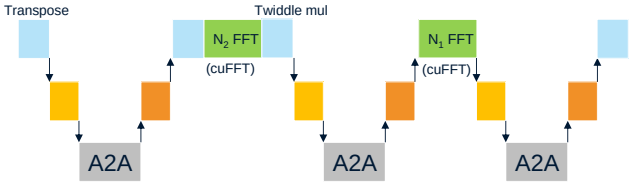


図 9 GPU を利用した 6-step FFT のモデル．上から GPU 上での計算時間，CPU-GPU 間のデータ転送時間，MPI による全対全通信の時間を表す

配列の長さ n の 1 次元 FFT 計算の計算量は式(6)で，データアクセス量は式(7)で求める．（ただし n は 8 のべき乗とする）

$$F = 5n \cdot \log n / \log 2 \quad (6)$$

$$B = 32n \cdot \log n / \log 8 \quad (7)$$

また，ひねり係数の乗算は，配列要素あたり 6 flops であるとし，要素あたり 48 バイトのデータアクセスがある．

全対全通信は，Infiniband 上で行う場合，配列サイズが十分大きいとすると最大で式(8)に示すバンド幅になる．

$$A2A_{bw} = \frac{Nn \cdot Bw}{Nppn \cdot (Nn-1)} \quad (8)$$

Nn は使用するノード数，Bw はネットワークバンド幅（80%を実効バンド幅とする），Nppn はノードあたりのプロセス数を表す．図 10 は，ノードあたり 2 プロセスを使用した場合の全対全通信のバンド幅の測定値と式(8)による最大バンド幅を比較したものである．

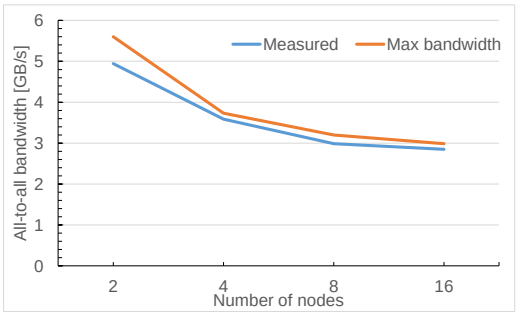


図 10 Infiniband FDR 接続のクラスターにおけるノードあたり 2 プロセスのときの全対全通信のバンド幅

5.4 実機による性能測定と性能予測の比較

1 次元 FFT を，前進，後退の順に繰り返し実行した場合の実効性能について，ノードあたり 2 プロセス（プロセスあたり 1GPU）を用いて測定を行い性能予測モデルと比較した．ここでは，使用するノード数によって配列サイズを変えているため実行時間ではなく，実効性能を用いて評価する．測定結果との比較を図 11 に示す．この結果から，実測値は最大性能までは届いてはいないが，性能の傾向としてはおおむね正しく，最大性能を予測するには問題ないといえる．

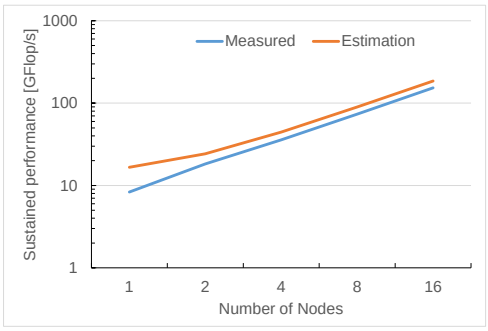


図 11 Tesla K40 をノードあたり 2 枚搭載した Infiniband FDR 接続のクラスターにおける並列 1 次元 FFT の実効性能と性能予測による最大性能値の比較

5.5 並列 1 次元 FFT の NVLink による性能向上予測

Pascal 世代の環境において，POWER プロセッサーと Pascal の NVLink 接続と，それ以外のプロセッサーの PCI express 接続を想定した並列 1 次元 FFT の性能予測を行う．ここでも，ノードあたり 2GPU を想定して，16 ノードを使用した場合の最大性能値の予測値求める．このとき，十分に大きな配列サイズについて 1 次元 FFT を行うとした場合の最大性能比較を表 8 にまとめる．NVLink 接続によって，9 割程度まで性能が向上することが予測される．

表 8 並列 1 次元 FFT の予測最大性能値の比較（16 ノード，ノードあたり 2GPU 使用）

	K40	Pascal			
	Infiniband FDR	Infiniband EDR		Infiniband EDRx2	
	PCIe	PCIe	NVLink	PCIe	NVLink
Performance [Gflop/s]	185.90	278.46	**427.13**	376.83	**712.39**
Speedup [%]			**53.39**		**89.05**

6. まとめ

3 つのアプリケーション，格子 QCD，HPCG ベンチマーク，並列 1 次元 FFT について，性能予測モデルを作り，実機による性能比較により，モデルの妥当性を検証した．作成した性能予測モデルにより，IBM の次世代 Power システムにおいて Pascal アーキテクチャの GPU を NVLink によって接続した場合の，PCI express 接続と比べた性能向上を予測した．この結果，(1)格子 QCD では 2～3 割程度，(2)HPCG ベンチマークでは 6%程度，(3)並列 1 次元 FFT では 8～9 割程度，の性能向上が得られることが予測できた．それぞれのアプリケーションで CPU-GPU 間のデータ転送の割合が異なるため，このように性能向上比は異なるが，いずれのアプリケーションにおいても，NVLink がアプリケーションの性能向上に貢献することを確認できた．

本報告では 3 つのアプリケーションについて検証したが，今後はさらに多くのアプリケーションについての性能予測と，実機による検証を行っていきたい．また，NVLink によって CPU-GPU 間のデータ転送が高速化される場合に更なる最適化手法が考えられるかについても検討したい. 特に，CPU と GPU 双方による協調動作を行う場合についても NVLink による性能向上が見込まれる．本報告では GPU のみを使用したが，今後は CPU と GPU の双方を使用した場合についても検証したい.

参考文献

1) IBM Power Systems
http://www-06.ibm.com/systems/jp/power/
2) CORAL Collaboration,
https://asc.llnl.gov/CORAL/
3) NVIDIA NVLink High-Speed Interconnect
http://www.nvidia.com/object/nvlink.html
4) S. Williams et al., Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM* 52, 4 (April 2009), 65-76.
5) M. A. Clark et al. Solving Lattice QCD systems of equations using mixed precision solvers on GPUs, Comput. Phys. Commun. 181, 1517, 2010.
6) HPCG Benchmark
http://www.hpcg-benchmark.org/
7) TOP500 Supercomputing Site
http://www.top500.org/
8) HPC Challenge Benchmark
http://icl.cs.utk.edu/hpcc/
9) D. Takahashi et al., “A Blocking Algorithm for Parallel 1-D FFT on Clusters of PCs,” *Proc. 8th International Euro-Par Conference (Euro-Par 2002), Lecture Notes in Computer Science*, No. 2400, 2002, pp. 691-700.
10) cuFFT
https://developer.nvidia.com/cufft