超並列惑星磁気圏電磁流体シミュレーションに向けた隣接通信の効率化

深沢 圭一郎†‡　　梅田 隆行§　　南里 豪志†‡

惑星磁場の勢力範囲である磁気圏は非常に巨大であり、一般にプラズマを流体として扱う電磁流体（MHD）方程式を用いてシミュレーションが行われる．MHD シミュレーションは一般の流体に電磁場を考慮したシミュレーションになるため、並列計算時には隣接通信が主に用いられる．我々が現在ターゲットにしている解像度での惑星磁気圏シミュレーションでは超並列計算が必須であるが、10,000 並列近くでは、並列化効率が 5%程度下がることが確認されている．そのため、隣接通信を高効率に行うことが求められている．そこで我々はシミュレーション中の計算ステップをまたいで通信をまとめることで、通信量を減らさずに通信回数を減らす手法を考え、その性能を、実アプリケーションを用いて評価した．その結果、通信回数を半分にすることで並列化効率が約 2%上昇し 99%になることを確認した．また新しい手法では計算量が多くなる場合や並列数が増えても並列化効率の劣化が今までの手法に比べて少ない事も確認された．惑星磁気圏の計算では大領域を長時間計算することから、この違いは非常に効果的であると考えられる．

Efficient near-neighbor communication for massively parallel magnetohydrodynamics simulation of planetary magnetosphere

KEIICHIRO FUKAZAWA†‡, TAKAYUKI UMEDA§, and TAKESHI NANRI†‡

The planetary magnetosphere, which is an area of influence in its intrinsic magnetic field, has a huge configuration and is simulated using the magnetohydrodynamics (MHD) equations. MHD simulation consists of the neutral fluid dynamics including the magnetic and electric field thus the near-neighbor communication is dominated in the parallel computing basically. It is necessary to perform the massively parallel calculation for our targeted resolution of simulation, however, it is found that the parallel efficiency decreases with almost 10,000 cores. Thus it is required to do near-neighbor communication efficiently in such parallel calculation. In this study we introduce the efficient communication method to pack the communication over the multiple calculation steps to our MHD simulation model and evaluate its performance. As the result, we obtain the 2% increase of parallel efficiency and it reaches 99% to reduce the number of communication to half. In addition we found that the new communication method is robust to the degradation of parallel efficiency in increasing the number of parallel calculation processes. These are effective for the simulation of planetary magnetosphere because it requires the large number of gird and long calculation time.

† 九州大学情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University
‡ 独立行政法人科学技術振興機構, CREST
JST, CREST
§ 名古屋大学太陽地球環境研究所
Solar Terrestrial- Environment Laboratory, Nagoya University.

1. Introduction

惑星磁気圏は太陽風プラズマと惑星磁場の相互作用によって形作られており、非常に巨大な構造をもつ．そのため、その相互作用を調べるにはプラズマを流体近似した電磁流体（MHD）近似が用いられている．簡単に言えば、一般の流体に電磁場を考慮した方程式になっており、MHD 方程式は式(1)のようになっている．

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\nabla \cdot (\vec{v}\rho) + D\nabla^2 \rho \\
\frac{\partial \vec{v}}{\partial t} &= -(\vec{v} \cdot \nabla)\vec{v} - \frac{1}{\rho}\nabla P + \frac{1}{\rho}\vec{J} \times \vec{B} + g + \frac{\mathbf{\Phi}}{\rho} \\
\frac{\partial P}{\partial t} &= -(\vec{v} \cdot \nabla)P - \gamma P \nabla \vec{v} + D_P \nabla^2 P \\
\frac{\partial \vec{B}}{\partial t} &= \nabla \times (\vec{v} \times \vec{B}) + \eta \nabla^2 \vec{B} \\
\vec{J} &= \nabla \times (\vec{B} - \vec{B}_d)
\end{aligned} \qquad (1)$$

上から、連続の式、運動方程式、圧力変化の式（エネルギーの式）、最後が磁場の誘導方程式となる [1]．ここで ρ はプラズマの密度、$\vec{v}$ は速度、P はプラズマ圧力、$\vec{B}$ は磁場、$\vec{J}$ は電流密度、D = Dp は拡散係数、g は重力加速度、$\mathbf{\Phi} \equiv \mu N^2 \vec{v}$ は粘性、γ = 5/3 は 3 次元の比熱定数、η は電気抵抗である．$\vec{B}_d$ は惑星の固有磁場を示す．このように中性流体に加え、電磁場を解くため、方程式の計算量自体も多い．我々はこの MHD 方程式を用いて惑星磁気圏の研究を行ってきた．シミュレーション結果の例として Fig. 1 に土星磁気圏のプラズマ対流と沿磁力線電流の関係を示す．ここでは磁気圏内のプラズマ対流が渦構造を持ち、その渦構造が沿磁力線電流を生み、オーロラ発光と関係していることを示している [1]．このような計算ではグローバル 3 次元磁気圏を精度良く解かなければならず、計算量は非常に多くなる．

電磁流体は領域分割を行うことで並列化しやすいため、これまでは大規模並列計算に対応することができた．しかしながら、我々の一つの目標である、マクロとミクロの遷移領域（MHD 近似の限界領域）を計算するためには、0.01 R_S（R_S は土星半径 = 60,300 km）の格子幅、$10{,}000^3$ 程度のグリッドが必要であり、その計算量の多さから今まで以上の超大規模並列計算が必要と想定される．例えば、今後 10 年程度 CPU の周波数がそれほど上がらないと仮定すると、$10{,}000^3$ グリッドを現実的な時間で計算するためには 200 万コアを利用する必要がある．しかしながら、東京大学情報基盤センターHA8000 システムを利用し、8192 並列の計算を行ったところ並列化効率が 1024 並列と比べ約 5%低下すること

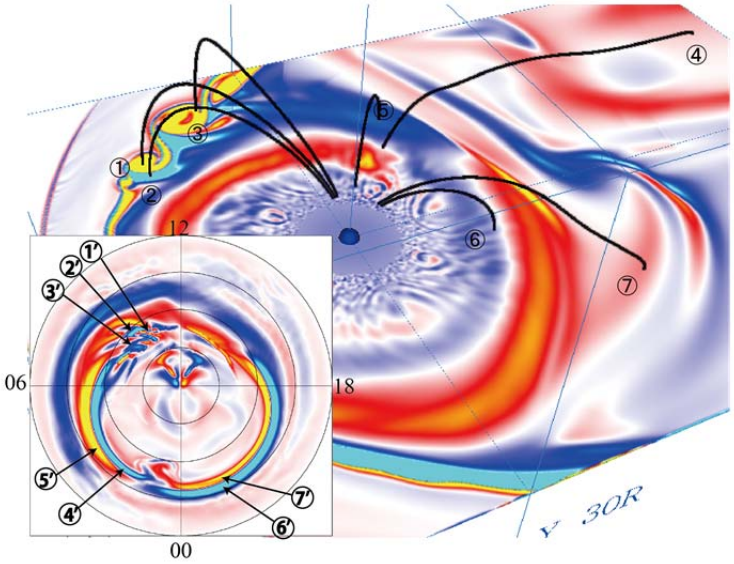

Fig. 1. 赤道面における渦度と極域沿磁力線電流（左下図）の関係．番号は磁力線によってその領域がつながっていることを示す．

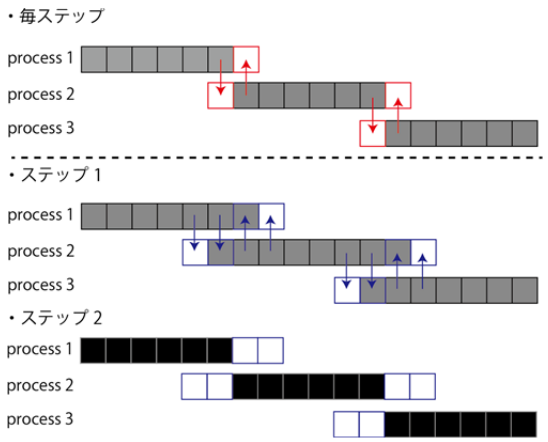

Fig. 2. プロセス間の隣接通信モデル（1 次元）．上図は従来のモデル、下図は 2 ステップに 1 回通信する場合のモデル．

がわかった [2]．そのため超大規模並列計算における並列化効率を上げることが課題になっている．

前述のように基本的に我々の計算は流体計算のため、並列化では隣接通信が大部分を占めている．またプロセス毎の計算負荷を均等にしており、並列化に伴う追加の計算もほとんど無いため、並列数が増大することに伴う並列化効率の劣化も隣接通信によって決まっていると言っても良い．そこで本論文は実アプリケーションである惑星磁気圏 MHD シミュレーションにおける隣接通信の性能を調べ、並列化効率の向上手法を議論する．

2. Improved Communication Model

シミュレーションコードは我々が開発した MHD 方程式を差分化し、惑星磁気圏を実際に計算しているコードを用いた [1]．このコード内で利用される隣接通信を評価し、並列化効率の向上を行う．

一般に各プロセスで 1 ステップ計算するには以下のように t_{total} かかる．

$$t_{total} = t_{mhd} + t_{comm} \quad (2)$$

ここで t_{comm} は隣接通信時間、t_{mhd} は MHD 計算にかかる時間であり、系が十分大きいため、これら以外の時間は無視できると仮定する．それぞれ以下のように定義できる．

通信時間を簡単な線形式と考えると、隣接通信時間 t_{comm} は

$$t_{comm} = \alpha x + \beta \quad (3)$$

となる．ここで α、β は正の定数であり、それぞれ通信量にかかる係数、遅延時間（同期時間含む）に対応する．x はメッセージ量（通信量）を表す．一方、MHD シミュレーション自体の計算時間 t_{mhd} は、

$$t_{mhd} = \gamma n \quad (4)$$

ここで γ は定数で、n は計算グリッド数を示す．

式(3)より、通信時間の短縮化は通信量（αx）を減らすか、遅延時間（β）を減らすことが考えられる．並列化に伴う通信量自体は計算手法、並列化手法を変更しない限り削減することは出来ない．そのため遅延時間を減らすことで通信時間を削減することを考える．我々は今までもこの遅延時間を少なくするために、バッファを自分で用意し、データをまとめて通信する手法を用いており、それにより並列化効率も上がっている．詳しく説明すると、毎ステップ 4 回ある通信により転送されるデータを一つのバッファにすべて格納し、そのバッファ全体を転送することで通信回数を 1 回に削減していた [3]．今回本研究では、このような単一計算ステップ内の削減では無く、複数計算ステップをまたいで通信回数の削減を行い、遅延時間を減らす．

数値計算において、並列化に伴い通信が必要となる基本的な理由は、あるグリッド上の値を計算するときに周辺グリッド上の情報が必要となるからである．そのため周辺グリッド上の情報が自プロセスに無い場合、その情報を別プロセスからもらってくる必要がある．実際のコードでは、何グリッド上のデータが必要になるかは計算手法の精度に依って異なるが、例えば、1 つ必要な場合は、基本的に 1 ステップに 1 グリッド分の自グリッド周辺情報をやりとりする必要がある．図で示すと、Fig.2 上図のように赤色グリッドの値を別プロセスから自プロセスに転送し、その値を使用して計算を行い、灰色に色づけされたグリッド上の値を更新する．そして更新されていない赤色グリッドの値を再び別プロセスから転送するという流れを繰り返す．

このような場合、ある計算ステップの 1 回の通信で 2 つ分の周辺情報をやりとりすると、次のステップでは通信の必要が無い．Fig.2 の下図 ではまず青色グリッドの値（2 グリッド分）を別プロセスから自プロセスに転送している．その後、計算を行うと灰色のグリッド上の値が更新される．次のステップではステップ 1 で更新された灰色部分のグリッドを使用し、計算を行うとステップ 2 の黒色のグリッ

Table 1. 実験 A,B における通信量(1=1,446,857 Byte)の変化

	計算量 (1=3,351,011,328 grids)					
	0.25	0.5	1	2	3	4
A	0.25	0.5	1	2	3	4
B	1	1	1	1	1	1

ド上の値が更新される．そして、更新されていない青色グリッド上の値を再度転送するという動作を繰り返す．

この時、10 ステップ計算するときにかかる時間は下記の t_{10mod} のように見積もられ、

$$t_{10mod} = 10\gamma n + 5(2\alpha x + \beta) \quad (5)$$

となる．また、従来の通信方法だと t_{10} のようになる．

$$t_{10} = 10 \times t_{total} = 10(\gamma n + \alpha x + \beta) \quad (6)$$

これらから通信時間の短縮が見積もることができる．式(6)と(5)の差をとると、

$$t_{10} - t_{10mod} = 5\beta \quad (7)$$

となり、遅延時間分通信にかかる時間が削減される．

このように通信回数を減らせば、遅延時間分が短縮される報告はすでにある [4]．しかしながら、実アプリケーションのように通信と計算が組み合わさった環境では、式(1)～(7)のような理想的な見積通りの動作になるとは限らず、実際に計測してみる必要がある．

3. Performance Measurements

本研究では、我々が実際に惑星磁気圏 MHD シミュレーションを行っているコード [1]を利用して、実計算において、どれだけ並列化効率上昇に効果があるかを調べる．計測する手法は、比較の対象となる通常の隣接通信を行った場合と、本研究で提案する手法による 2 種類（通信を 2 計算ステップに 1 回と 3 計算ステップに 1 回）の隣接通信である．現状我々は Flat MPI が最も並列化効率が出ているため、並列化には MPI だけを利用し、下記のように *mpi_sendrecv* を用いて隣接通信を行う．

1)1 計算ステップに 1 回の通信の場合（通常）

mpi_sendrecv(f(1, 1, ks, m), n2, mpi_real, left, 100, f(1, 1, ke+1, m), n2, mpi_real, right, 100, mpi_comm_world, status, err)

2)2 計算ステップに 1 回の通信の場合（改良版 1）

*mpi_sendrecv(f(1, 1, ks, m), n2*2, mpi_real, left, 100, f(1, 1, ke, m), n2*2, mpi_real, right, 100, mpi_comm_world, status, err)*

3)3 計算ステップに 1 回の通信の場合（改良版 2）

*mpi_sendrecv(f(1, 1, ks, m), n2*3, mpi_real, left, 100, f(1, 1, ke-1, m), n2*3, mpi_real, right, 100, mpi_comm_world, status, err)*

このように通信部分の書き換えは配列の始点をずらし、メッセージ量を変更するだけでよく、コードの書き換えコストも低い．

計測には九州大学情報基盤研究開発センター高性能演算サーバ（Fujitsu PRIMERGY RX200S6）96 コア（8 ノード）を利用した．このシステムは Intel Xeon X5670 (2.93GHz)で構成され、ノード間は InfiniBand 4x QDR(4GB)×1 によって結合されている．計測方法としては 64 ステップ分の計算を行い、その通信時間（*sendrecv* にかかる時間）を 64 で割ることで 1 ステップ分の通信時間とする．この通信時間は各プロセスにおいて、*sendrecv* の前後で時間を測り、その差をとり、さらにプロセス間で平均をとっている．通信の性質を調べるためにそれぞれ下記 A、B の実験をおこなった．

A）通信量を変化させ、通信時間がどう増えるか調べる．

B）通信量は等しくし、計算量を変化させ通信時間がどうなるかを調べる．

今回の計測では直交格子、3 次元空間(*x, y, z*)において、*z* 方向に 1 次元領域分割を用いて並列化したコードを利用している．この時、隣接通信されるのは *xy* 平面上のデータであるため、通信量を増加させる場合は、*x*、*y* 方向のグリッド数を増やし、通信量を固定し、計算量を増やす場合は、*z* 方向のグリッド数を増やしている．

実験は各 10 回行い、著しく遅いものは除外し、平均値を利用した．各実験のパラメータは Table 1 の通りである．

Fig. 3 に A の結果を、Fig. 4 に B の計測結果をそれぞれ示す．各図とも横軸が計算量、縦軸が通信時間を示す．

Fig. 3 では A の場合、計算量に比例して通信量も増大するが、その通信量に比例して通信時間が増大しているのが想定通りに見て取れる．しかしながら、式 (3) を考えると、傾き（α）は 3 種類の通信結果において等しいはずだが、計算量が増えるほどにグラフ同士の差が開くことから異なっていることがわかる．また、遅延時間に当たる切片（β）は 3 計算ステップに 1 回の通信の場合（改良版 2）では、近似直線を考えると負になり（$\beta = -0.00003$）、β が正の定数であるという仮定とは異なった結果になっている．通信時間の削減効果としては、式(7)で期待した遅延時間の削減だけでは無く、通信量の係数である傾きも、通常の方法より今回導入した手法の方が明らかに良くなっており、通信量が増えるとその差が大きくなることがわかった．また 2 計算ステップに 1 回では、式(5)、(6)から考えると、遅延時間 β は通常の 1/2 になる．また、3 計算ステップに 1 回かでは β は 1/3 になる．それらを考慮すると、計算量が 4 の場合は、この見積もり通りの結果になっているが、それより小さい計算量では、2 計算ステップに 1 回と 3 計算ステップに 1 回で差が見られない．我々が目標としている超大規模並列ではプロセス辺りの計算量がそれほど多くないと予想されることから、2 ステップに 1 回の手法で十分と考えられる．

一方、B では通信量は変わらないが、Fig. 4 のように計算量が増えると小さな差だが通信時間が増えてきている（Fig. 3 と Fig. 4 は縦軸のスケールが異なる）．式(3)の見積もりでは、各グラフは計算量に依って変化せず、横一直線となると考えられる．これは計算量が増えることにより、プロセス毎の計算時間が微妙に異なってくるため、通信時にプロセス間での同期をとる時間が増加しているからだと考えられる．このような結果は単純な通信だけの実験では見ることが出来ず、実際のアプリケーションで計算しながら、実験をしていることで初めて出てくる現象である．

また Fig. 4 おいて、計算量が増えていく時の通信時間増加の傾きが通常の手法と比べて今回導入した手法では明らかに小さく、通信時間改善の効果がよく見える．これは通信に伴う同期の回数が減ったことから来ていると考えられる．また、B の場合でも 2 計算ステップに 1 回と 3 計算ステップに 1 回ではそれほど差が見えない．同期の影響を考えれば傾きが異なるべきだが、Fig. 4 からその違いは見えない．3 計算ステップに 1 回通信のグラフはあまり直線的に変化しておらず、傾きが一定とは言いがたい．そのため、計算量を更に増やすなどの実験を行う必要がある．A の場合と同じく通常に比べ、β の値が 2 計算ステップに 1 回の通信では 1/2、3 計算ステップに 1 回では 1/3 となるが、計算量が 2 未満の時は

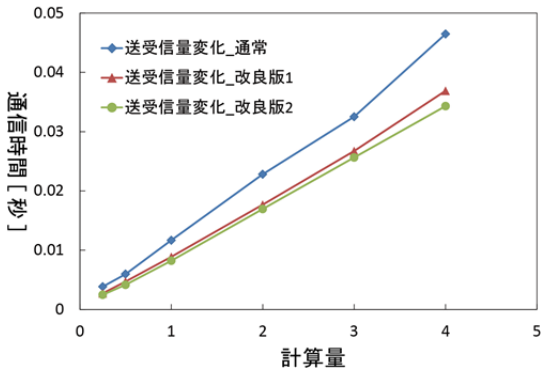

Fig. 3.　通信量を変化させた場合の各手法による通信時間の変化（実験 A）．グラフの通常は 1 計算ステップに 1 回の通信、改良版 1 は 2 計算ステップに 1 回の通信、改良版 2 は 3 計算ステップに 1 回の通信による結果を示す．

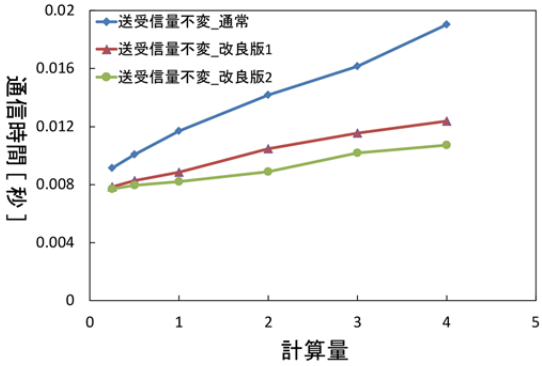

Fig. 4.　通信量を固定させた場合の各手法による通信時間の変化（実験 B）．グラフの通常は 1 計算ステップに 1 回の通信、改良版 1 は 2 計算ステップに 1 回の通信、改良版 2 は 3 計算ステップに 1 回の通信による結果を示す．

その見積もりの差は見えない．2 以上からは見積もり通りの結果になっている．

B の結果から例えば京コンピュータの全プロセッサコアを利用した場合の 70 万並列程度を考えると、同期による遅延が更に大きくなると想定され、今回のように通信回数自体を減らすことが、式(3)で考えるような理想的な遅延時間だけの削減では無く、通信時間全体の削減に有用であると考えられる．

4. Discussions and Summary

本研究では、シミュレーションを行う計算ステップ中で通信回数を間引くという手法を提案し、実際のシミュレーションコードに導入することにより、実アプリケーションレベルで評価を行った．理論的には我々の MHD コードでは隣接通信が主であり、この通信を高効率に行うことが超大規模並列計算時代では重要である．Fig. 5 に 1 計算ステップに 1 回の通信時間と今回の実験 A、B で行った 2 計算ステップに 1 回、3 計算ステップに 1 回の通信時間の差（削減通信時間）をそれぞれ示す．今回の結果により Fig. 5 に示すように 1 通信時間当たり最大で 0.012 秒近く時間の改善があった．

また、並列数を増やした場合、768 並列で 2 ステップに 1 回の通信で約 0.03 秒の改善が見られ、並列数が増大すると今回の手法がより効果的であることも確認している．我々が対象としている惑星磁気圏では今回と全く同じコードを使用し、より大きい計算量、並列数で長時間計算を行っている．そのため、絶対値で見ればわずかな通信時間の改善に見えるが、実際のシミュレーション全体の計算時間は約 8%短縮されており、1 年間の計算で 1 ヶ月近い時間の短縮になると見積もることができた．

実際に並列化効率がどの程度上昇したかを調べると、通常の通信方法の場合が 94～98%（計算サイズが大きいほど効率は上がるため幅がある）だったものが、今回の手法では 97～99%に上昇している．この手法はコードの書き換えも非常に簡単であり、通信回数を減らすことによってわずかに計算量が増えるが（1 グリッド境界分）、我々が想定している計算では 1 プロセス当たり(100, 100, 100)の配列であるため、ほぼ誤差の範囲内である．また今回の実験は 1 次元領域分割の結果だが、2 次元、3 次元領域分割の場合でも、各分割次元で通信回数が増

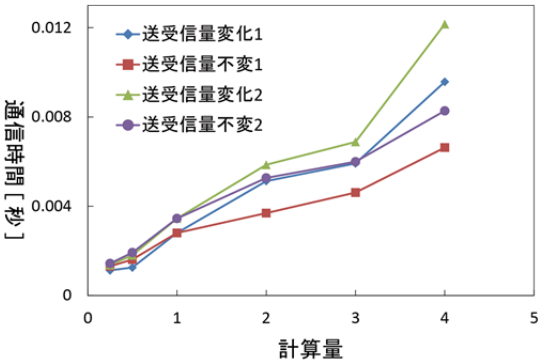

Fig. 5. 本研究で提案した手法による通信の改善時間．各グラフの 1 は 1 計算ステップに 1 回の通信結果と 2 計算ステップに 1 回の通信結果の差、2 は 1 計算ステップに 1 回の通信結果と 3 計算ステップに 1 回の通信結果の差を示す．

えるだけであり、本研究の結果をそのまま適応できる．さらに流体計算であれば、容易に本研究と同手法をコードに組み込むことが可能であり、並列化効率の改善も強く見込まれる．

また、複数 GPU で並列計算を行う場合、GPU 間の通信が遅いため、通信を減らすことが求められるが、この手法は有効な方法と言える．

今後は、非同期隣接通信、物理 CPU、ノードの配置による並列化などを考慮し、隣接通信の高効率化を行っていく．

参 考 文 献

[1] Fukazawa, K., T. Ogino, and R.J. Walker, A Magnetohydrodynamic Simulation Study of Kronian Field-Aligned Currents and Aurora, J. Geophys. Res., revised, 2011.

[2] Fukazawa, K., T. Umeda, Performance measurement of magnetohydrodynamic code for space plasma on the typical scalar type supercomputer systems with the large number of cores, The International Journal of High Performance Computing Applications, revised, 2011.

[3] Fukazawa, K., T. Umeda, T. Miyoshi, N. Terada, Y. Matsumoto, and T. Ogino, Performance measurement of magneto-hydro-dynamic code for space plasma on the various scalar type supercomputer systems, IEEE Trans. Plasma Sci. 38, 9.2254, 2010.

[4] Subramoni, H., M. Koop, and D. K. Panda, Designing Next Generation Clusters: Evaluation of InfiniBand DDR/QDR on Intel Computing Platforms, 17th Annual Symposium on High-Performance Interconnects (HotI'09), August 2009.