TSUBAME 共同利用 平成 25 年度 学術利用 成果報告書

利用課題名 ポストペタ時代の大規模並列数値計算のための技術開発

英文：Research and Development on Large Scale Parallel Numerical Computations in the Post-Peta Era

須田 礼仁
Reiji Suda

東京大学 情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo
http://sudalab.is.s.u-tokyo.ac.jp/~reiji/

邦文抄録（300 字程度）

本研究では，ポストペタスケールの計算機での大規模数値計算を目標として，手法の開発を行っている．いくつかのテーマに取り組んでいるうちから，本稿では，複数 GPU が搭載されているときに，CPU と GPU の間でのデータ転送の所要時間を最小にする転送方式を決定するアルゴリズムについて報告する．この問題は線形計画問題として定式化できるが，双対問題を考えることで，幾何学的な解釈をすることができる．これにより，データ転送時間を最小にする転送方式は，たかだか 2 つのシステム状態（データ転送の組み合わせ）により実現されることが示された．

英文抄録（100 words 程度）

We are developing several methods toward large scale numerical computations on post-peta scale supercomputers. Among several works we achieved, this report considers a problem of finding data transfer schedule between CPU and multiple GPUs, and gives an algorithm that gives a schedule with minimal data transfer time for given data sizes. This problem can be solved via linear programming, but we present a geometric interpretation of the problem, which leads us a theorem that at most 2 system states (i.e. combination of types of data transfer) realizes minimal time schedule.

Keywords: 5つ程度

Scheduling, multi-GPU system, GPU-CPU data transfer, optimal schedule, linear programming.

背景と目的

ポストペタ時代の大規模並列計算機として，GPU のようなアクセラレータを搭載した並列計算機は有力なアーキテクチャである．しかしながら，電力消費における課題，超大規模並列化プログラミングのための手法の必要性，強スケーリングのための新たなアルゴリズム開発，プログラマビリティの課題など，多くの課題が浮かび上がってきている．これらの課題の解決をめざし，我々はソフトウェアとアルゴリズムの視点で取り組みを進めている．特に大規模並列数値計算を主なターゲットとして，並列化手法やアルゴリズム開発を行っている．

概要

平成 25 年度には，以下の成果を挙げた．(1) 超大規模離散最適化問題の超並列 GPU 実装を実現した．(2) TSUBAME のように 1 ノードに複数 GPU が搭載されているシステムに対して，CPU と GPU の間のデータ転送の最適な手法を開発した．(3) また，(2) の成果に基づいて，divisible load model により，複数 GPU を用いた漸近最適なタスク分割・スケジューリング手法を提案した．(4) 将来の超大規模並列計算機に向けて，Krylov 部分空間法の集団通信の回数の削減が求められており，block Krylov と Chebyshev 基底を併用した Block Chebshev Basis CG (BCBCG) を提案した．(5) 数値的に限りなく安定な Krylov 部分空間を生成することをめざし，Arnoldi 過程を通信削減で実現するアルゴリズムを開発した．(6) GPU を用いて多倍長精度の行列計算基本アルゴリズムを実装し，高速化を達成した．(7) 将来の超大規模並列計算機における細粒度並列化に向けて，OS ジッタの影響を緩和する集団通信アルゴリズムの開発・実装を行った．(8) 今後一層問題となる，電力最適化に向けて，オンライン自動チューニングのアルゴリズムを開発・実装した．

これらすべてについて詳細を報告することは困難なので，本稿では，上記の内，(2) の複数 GPU に対するデータ転送の最適アルゴリズムについて詳述する．

CPU と GPU とのデータ通信は，GPU プログラミングにおける重要なテーマの一つである．近年では，1 つのマザーボードに複数の GPU を搭載することができるようになっており，TSUBAME 2.0/2.5 もそのような構成になっている．また，複数ストリームを利用することにより，CPU から GPU へのデータ転送（以下，CPU 側から見て，「送信」と呼ぶ）と，GPU から CPU へのデータ転送（以下，「受信」と呼ぶ）も同時に実行できる．

例えば，送受信を同時に行うと，送信だけや受信だけとは異なるスループットが得られる．また，複数 GPU に同時にデータ転送をすると，その影響も受ける．これらの要因のため，CPU と GPU の間のデータ転送の最適な方法は必ずしも自明ではない．本稿ではこの問題に対する解を提示する．

さて，各 GPU について見ると，

- N: データ転送をしていない
- S: CPU→GPU のデータ転送（送信）
- R: GPU→CPU のデータ転送（受信）
- SR: 送受信の双方向同時転送

という 4 つの状態がある．従って，n 台の GPU があれば，システム全体として 4^n 通りの状態がある．CPU と GPU の間で，あるデータ量を転送しようとしたときに，この 4^n 通りの状態のどれをどのように組み合わせれば，最大のスループットが得られるかが，課題である．

以下では，簡単のため 2 台の GPU の場合について説明する．GPU のうち 1 台を G0，もう 1 台を G1 と呼ぶ．システム全体としてのデータ転送状態は 16 通りあるが，G0 の状態と G1 の状態を / で区切って並べて示す．例えば S/N は G0 に送信のみしている状態，SR/SR は両方の GPU ともに送受信同時に行っている状態である．システム状態が q のとき，Gi への送信のスループットを $S i_q$，受信のスループットを $R i_q$ と書く．

また，本稿では，2 つの GPU は同じ仕様で，同じ接続になっているものとする．例えば，G0 だけに送信するときのスループット $S0_{S/N}$ は，G1 だけに送信するときのスループット $S1_{N/S}$ に等しく，G0 と G1 に同時に送信するときには両方に同じスループットで転送される（すなわち $S0_{S/S} = S1_{S/S}$）などとする．

さらに，簡単のため，2 つの GPU へのデータ転送要求は等しいとして，合計の送信データサイズを X_S，合計の受信データサイズを X_R とする．従って，各 GPU には，それぞれの半分，つまり $X_S/2$, $X_R/2$ のデータが送信・受信されることになる．

さて，各システム状態 q で時間 T_q だけデータを転送することを考える．このとき，G0 に送信されるデータサイズは $X_{S0} = \Sigma_q T_q S0_q$, G1 から受信するデータサイズは $X_{R1} = \Sigma_q T_q R1_q$ などとなる．またデータ転送の所要時間は $T = \Sigma q\ Tq$ である．このモデルではバンド幅のみ考慮しており，転送のスタートアップ時間などは考慮していない．このモデルによれば，転送スループットはスケジューリングの順序によらず，各システム状態での時間のみ決定すればよい．所要時間最小でデータを送る問題は

Minimize T

subject to $X_{S0} = X_{S1} = X_S/2$, $X_{R0} = X_R1 = X_R/2$

という線形計画問題になる．これを「所要時間最小化問題」と呼ぶことにする．

上記の線形計画問題は，通常の方法で容易に解くことができるが，以下では幾何的な解釈によりその特徴を説明する．

まず，本稿でおいている仮定として，両 GPU に同じだけデータ転送するため，対称なデータ転送は同じ時間かけると制限しても最適解が得られることが証明できる．すなわち $T_{S/N} = T_{N/S}$, $T_{R/N} = T_{N/R}$, $T_{SR/R} = T_{R/SR}$ などの 6 つの等式が成り立つ．最適解では自明に $T_{N/N} = 0$ なので，あわせて 7 つの等式が得られ，もともと 16 あった自由度のうち，9 つだけが残る．

さて，システム状態 S/N と N/S は同じ時間かけるので，これらをペアにして，一つのシステム状態 S/N として改めて定義する．転送時間 $T_{S/N}$ およびスループット $S_{S/N}$ も，2 つのシステム状態のペアに対して定義する．このようにすることで，「すべてのシステム状態において 2 つの GPU に等しいデータ量が転送される」という簡単化がなされる．すなわち，各システム状態 q は，2 つ

の GPU に対する合計送信スループット S_q と合計受信スループット R_q の 2 つのパラメタで完全に特徴づけられる．これを 2 次元平面上の点 $P_q = (S_q, R_q)$ に対応付けよう．この対応付けにより，9 つのシステム状態が平面上の 9 つの点に対応する．

次に，所要時間最小化問題の双対問題を考える．これは，データ転送の合計時間 T を固定したときに，データ転送量（X_S, X_R）が最大となるデータ転送方法を決める問題である．ただし，「データ転送量が最大である」とは，所要時間が T となる他のどのようなスケジュールでも，「データ送信量が X_S 以下」または「データ受信量が X_R 以下」のどちらか（または両方）が成立することをいう．すなわち，2 つの目的関数 X_S, X_R に対するパレート最適解が「データ転送量が最大」である．これを「データ転送量最大化問題」と呼ぶことにする．一般性を失うことなく T = 1 としてよい．

さて，最初に，2 つのシステム状態 q1 と q2 のみを使う場合を考察しよう．合計所要時間が 1 であるから，状態 q1 に時間 a，状態 q2 に時間 1 − a を費やすとしよう（$0 \leqq a \leqq 1$）．このとき転送データ量は

$X_S = a\, S_{q1} + (1 - a)\, S_{q2}$,

$X_R = a\, R_{q1} + (1 - a)\, R_{q2}$

となる．この点 (X_S, X_R) は，図 1 に示すように，点 P_{q1} と点 P_{q2} を結ぶ線分上にある．$0 \leqq a \leqq 1$ を適当に選ぶことで，線分上の任意の点を取ることができる．これを「状態 q1 と状態 q2 の凸結合」と呼ぶことにする．

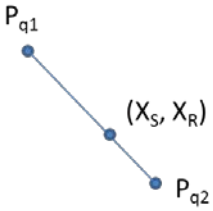

図 1. 状態 q1 と状態 q2 の凸結合．

次に，3 つのシステム状態 q1, q2, q3 を使う場合を考える．このとき，先と同様に考えると，点 P_{q1}, P_{q2}, P_{q3} が作る三角形の内部を取ることができることがわかる．図 2 (a) のようになっていれば，q1 と q3 の凸結合が右上の辺，すなわちパレート最適解に対応しており，q2 を含むどのような通信スケジュールよりもデータ転送量が多い．すなわち，(a) では状態 q2 は使う意味がない．一方，図 2 (b) のようになっていれば，q1 と q3 の凸結合よりも，q1 と q2 の凸結合，あるいは q2 と q3 の凸結合のほうが右上にあり，データ転送量が多い．すなわち，q1 と q3 の凸結合，あるいは 3 つの状態すべてを使った凸結合よりも，q1 と q2，あるいは q2 と q3 の 2 状態の凸結合がよい．

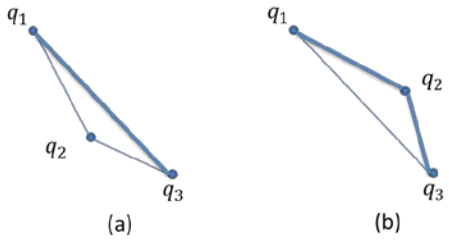

図 2. 3 つの状態の凸結合．

これを一般化すると，図 3 のようになる．すべての状態を使って時間 1 の間に転送できるデータ量は，各状態に対応する点から構成される凸包の内部または境界上の点に対応している．従って，転送データサイズが最大になる転送方法は，右上凸包の頂点または辺上の点に対応している．

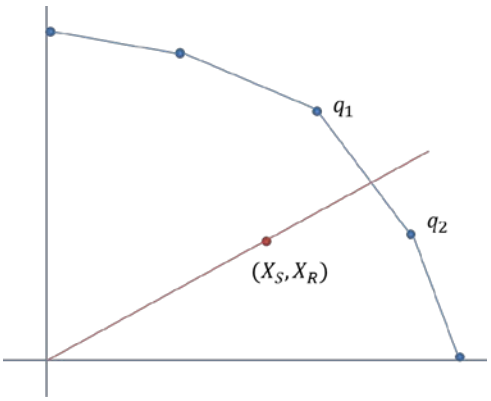

図 3. データ転送量最大化問題の解．
赤い線と凸包の交点から，所要時間最小のスケジュールに用いるシステム状態 q1, q2 がわかる．

所要時間最小化問題の解は，データ転送量最大化問題の解でもなければならない．従って，図 3 の右上凸包の頂点または辺上の点である．図 3 には，データ転

送量（X_S, X_R）が与えられたときに，所要時間を最小にする 2 つの状態 q1, q2 の求め方が示されている．すなわち，(X_S, X_R) と原点を通る直線と，右上凸包との交点を求め，その辺の両側の頂点に対応する状態を用いればよい．交点の位置から，状態 q1 および q2 で通信に要する時間が計算できる．

以上が所要時間最小化問題に対する幾何的な解法である．

GPU の台数が増えても同様で，システム状態の数が増えることに伴って点の数は増えるが，やはり 2 次元平面上の凸包の問題に帰着される．ここから重要な結論が 1 つ導かれる．

定理：対称 GPU システムに対する所要時間最小のデータ転送スケジュールは，1 つまたは 2 つのシステム状態からなる．

ただし，1 つのシステム状態は対称性により再定義される前の複数のシステム状態を含んでいる可能性がある．しかし，もともと 4^n 個あったシステム状態のうち，ほんのわずかだけを用いるのが最適であることが示されたことになる．

結果および考察

本研究により，複数 GPU に対して最小時間でデータ転送するスケジューリングが明らかになった．また，その幾何学的な解釈と，そこから導かれる性質について説明した．

我々はこの成果をもとにして，divisible load model に基づくタスク分割とパイプライン型のスケジューリングアルゴリズムを提案している．そこでは，（本稿では無視した）通信やカーネル起動のオーバーヘッドを考慮に入れても，タスクサイズが無限に大きい極限において，理論的な最適解に限りなく近づくという，漸近最適性を証明することができている．これらの成果は論文にまとめて近く投稿する予定である．

まとめ、今後の課題

本プロジェクトではいくつかのテーマに取り組んでいるが，本稿では複数 GPU に対するデータ転送の最適なアルゴリズムについて詳しく報告した．ここでは対称な GPU に対する対称なデータ転送しか論じていないが，非対称な GPU システムや非対称なデータ転送についても考えていく必要がある．また，divisible load model はタスクが一様であると仮定しているが，非一様なタスクも現実にはあるので，そのようなモデルに対する解も求めてゆきたい．