2004－HPC－99　(18)
2004／7／31

マルチクラスタ環境での MMDL 漸近最適スケジューリング

須　田　礼　仁†

MMDL (Multi-Master Divisible Load) モデルは著者が DLT (Divisible Load Theory) をデータ再分散や動的負荷分散に向けて拡張したものである。本稿では、複数のクラスタをクロスバースイッチで接続した二階層のマルチクラスタ環境における MMDL 問題に対する漸近最適なスケジューリングのアルゴリズムを報告する。ここでは各クラスタ内の通信性能は均一（クラスタに依存してもよい）であり、クラスタ間通信性能も均一で、どのクラスタのクラスタ内通信の性能よりもクラスタ間通信の性能の方が低いシステムを仮定する。また、通信は full duplex で、処理の結果は収集しなくてよいとした。プロセッサ数 P に対し提案アルゴリズムは $O(P \log P)$ の計算量で漸近最適スケジュールを求める。本稿は MMDL に通信の非一様性を取り入れた最初の報告となる。

An Asymptotically Optimum Scheduling Algorithm for an MMDL Problem on Multi-Cluster Environments

Reiji SUDA†

MMDL (Multi-Master Divisible Load) model is an extension of DLT (Divisible Load Theory) applicable to data redistribution and dynamic load balancing. This paper proposes an asymptotically optimum scheduling algorithm for an MMDL problem on multi-cluster systems that consist of several clusters connected by a crossbar switch. The communication performance within each cluster is assumed to be uniform (but may be dependent on the cluster), and the inter-cluster communication is also uniform but its performance is lower than any intra-cluster communication. The proposed algorithm assumes full duplex communications and no collection of the results, and computes an asymptotically optimum schedule in $O(P \log P)$ where P is the number of processors. This paper is the first report of MMDL with heterogeneous network.

1. はじめに

MMDL (Multi-Master Divisible Load) は著者が前報告[1] において、データと処理の再分散のための一つのモデルとして提案したものである。従来の DLT (Divisible Load Theory)[2] はマスター・ワーカー型の並列処理をモデル化したもので、非常にシンプルなモデルでありながら、ヘテロ環境を含む各種の条件の下で興味深い理論的な結果を多く導き、またいくつかの応用も行われてきたものであるが、マスター・ワーカーの枠組みは多様な並列処理のあり方からするとあまりにも適用範囲が狭いという感は否めなかった。MMDL は DLT のシンプルさを受け継ぎながら、マスター・ワーカーという制約を解消したモデルであり、データ再分散や動的負荷分散の基礎理論として役立ちうる枠組みになったと考えている。

前報告では、この MMDL の提案とともに、計算性能がヘテロでもよいクラスタ環境における漸近最適スケジューリングアルゴリズムを提案した。本報告では、複数のクラスタを接続したマルチクラスタ環境での漸近最適スケジューリング問題について報告する。

† 東京大学 情報理工学系研究科 コンピュータ科学専攻
Dpt. of Computer Science, Grad. Schl. of Information Science and Technology, the University of Tokyo

2. マルチクラスタ上の MMDL モデル

2.1 タスクのモデル

今回扱う MMDL タスクのモデルは前報告のものとほぼ同じである。複数のプロセッサがあり、初期状態ではプロセッサ P_i に x_i の量のタスクが割り当てられている（x_i は 0 でもよい）。タスクは相互に依存がなく独立に処理することができる。また、どのプロセッサもどのタスクでも処理することができる。タスクは任意のサイズに切り分けることができ、タスクを移送する場合のメッセージサイズはタスクサイズに比例する。今回は前報告とは異なり、処理の結果の収集は行わないものとする。

2.2 システムのモデル

今回扱うシステムのモデルは二階層のマルチクラスタである。各クラスタはクロスバースイッチにより接続されたプロセッサからなる。クラスタ内の通信のスループットはどのプロセッサ間でも同じであり、クラスタ C_j 内では単位サイズのタスクの移送に β_j の時間がかかるとする。通信の立ち上がり遅延は無視する。プロセッサの計算性能は異なってもよく、プロセッサ P_i 上で単位サイズのタスクを処理するために γ_i の時間がかかるものとする。

クラスタ間の通信は一つのクロスバースイッチにより行われる。クラスタ間の通信のスループットはどのクラスタ

間でも同じであり、単位サイズのタスクの移送に β の時間がかかるとする。通信の立ち上がり遅延は無視する。各クラスタのクロスバースイッチの口の一つがクラスタ間通信のクロスバースイッチに接続されていると考える。

各プロセッサは通信と処理をオーバーラップすることができると仮定する。プロセッサ P_i がクラスタ内通信を行っている間に処理を行う場合の単位タスクあたりの所要時間を $\gamma_i^{(I)}$, クラスタ間通信を行っている間に処理を行う場合のそれを $\gamma_i^{(X)}$ とする。また、通信は full duplex を仮定し、クラスタ内通信とクラスタ間通信を同時に行っている間に処理を行う場合の単位タスクあたりの所要時間を $\gamma_i^{(D)}$ とする。但し single port とし、送信を複数、あるいは受信を複数同時に行うことはできないものとする。

クラスタ内通信性能 β_j とクラスタ間通信性能 β の関係はスケジューリングのアルゴリズムに大きく影響する。特に、すべての β_j が等しく、β が β_j の $1/k$ (k は 1 以上の整数) の場合が最も簡単である。この場合、プロセッサ間通信はすべて同じ性能になるので「リレーなしの制約」(前報告参照) のもとで漸近最適解が構築できる。そのアルゴリズムは前報告にあるものを比較的すなおに拡張したものとなる。

今回は上記の簡単な場合は紙数の都合で省略し、不均一な通信の場合について考察を進めることにする。すなわち、クラスタ内通信性能 β_j はクラスタによって異なってもよいとし、クラスタ間通信性能については $\beta \geq \beta_j$ がすべてのクラスタ C_j に対して成り立つことを仮定する。クラスタ間通信の方がクラスタ内通信よりも性能が低い場合であり、厳密ではないが SMP クラスタの近似としても使える可能性がある。

2.3 定常ラウンド スケジューリングの制約

次に、スケジューリングの方式について制約を設ける。今回は通信が不均一であるため、リレーなしの制約を設けると漸近最適性を保証することができない。そこで定常ラウンドの制約のみをスケジューリング方式の制約とする。

定常ラウンド方式のスケジューリングでは、ラウンドと呼ばれる同一のパターンの通信や処理の組み合わせを (最初と最後を除いて) 反復する。これにより、問題を定常状態にある各ラウンドのスケジューリングに帰着させることができる。また、あるラウンドの処理に必要な通信は前のラウンドで行うことにする。これにより、ラウンド内の処理は通信に依存せず、自由なスケジューリングで実行することができることになる。

タスクが任意のサイズに分割でき、通信と処理の時間がタスクサイズに比例し、通信の立ち上がり時間を無視する、という仮定から、ラウンド内のスケジューリングはタスクサイズを定数倍しても不変である。従って、ラウンド数が R のときに一ラウンドの所要時間は T/R と書くことができる。$R \to \infty$ において最初と最後のラウンドの影響が無限に小さくなり、全体所要時間が T に漸近する。定常ラウンドのスケジュールが最適であれば、T は全体の所要時間の下限となる。

実際には通信には一定の立ち上がり時間がかかる。立ち上がり時間がメッセージサイズに依存しないとすると、一ラウンドの所要時間はある定数 α を用いて $T/R + \alpha$ と書くことができるはずである。最初のラウンドは通信だけ、最後のラウンドは計算だけを行うことを考慮すると、全体の所要時間は $T + T/R + \alpha R$ でおさえられる。これは $R = \sqrt{T/\alpha}$ のときに最小値 $T + 2\sqrt{T\alpha}$ となる。ここで α は定数であるから、$T \to \infty$ の極限において T に漸近する。すなわち、定常ラウンドの最適解が得られれば、通信の立ち上がり時間も込みにして、全体として漸近最適解を構成することができる。

3. 必要条件：最小ラウンド時間

定常ラウンドの仮定により、ラウンド数を固定したときに一ラウンドの所要時間を最小化する問題を解けばよい。通信の立ち上がり時間を無視すればラウンド数 R は任意にとってよいから、$R = 1$ とする。このとき最小定常ラウンド時間 T は全体の所要時間の下限でもある。

まず、ラウンド時間 T 以内に必要な処理と通信を完了することができる必要条件を求め、その条件を満たす最小の T を求める。

3.1 リレーなしの仮定のもとでの分析

今回はリレーなしの制約がないが、まずリレーがないものと仮定して問題を分析することは有益である。ここでリレーとは、あるプロセッサ P_i が他のプロセッサ P_j からタスクを受け取り、さらに別のプロセッサ P_k にタスクを送出するようなスケジューリングを言う。

まず、$x_i\gamma_i > T$ となるプロセッサは他のプロセッサに処理を委託しないとタスク x_i を完了することができない。マスターワーカーモデルの連想から、このようなプロセッサをマスタープロセッサと呼ぶことにする。リレーがない場合、マスタープロセッサは他のプロセッサ (複数かもしれない) にタスクを送るが、受け取ることはない。プロセッサ P_i の属するクラスタ内のプロセッサに送るタスク総量を $-y_i^{(I)}$, 他のクラスタのプロセッサに送るタスク総量を $-y_i^{(X)}$ とする。

逆に $x_i\gamma_i < T$ となるプロセッサは他のプロセッサからタスクを受け取り処理をする余裕がある。このようなプロセッサをワーカープロセッサと呼ぶことにする。リレーがない場合、ワーカープロセッサは他のプロセッサ (複数かもしれない) からタスクを受け取るが、送出することはない。プロセッサ P_i の属するクラスタ内のプロセッサから受け取るタスク総量を $y_i^{(I)}$, 他のクラスタのプロセッサから受け取るタスク総量を $y_i^{(X)}$ とする。

プロセッサ P_i が手持ちのタスク量 x_i をすべて自分で処理すると $x_i\gamma_i$, クラスタ内のプロセッサにできるだけ委託すると $x_i\beta_j\gamma_i^{(I)}/(\beta_j + \gamma_i^{(I)})$, クラスタ外のプロセッサにできるだけ委託すると $x_i\beta\gamma_i^{(X)}/(\beta + \gamma_i^{(X)})$ の時間がかかる。ラウンド時間 T はこれら 3 つの値の最小値よりも

小さくすることはできない。すなわち、

$$t_i = \min\left\{x_i\gamma_i, \frac{x_i\beta_j\gamma_i^{(I)}}{\beta_j+\gamma_i^{(I)}}, \frac{x_i\beta\gamma_i^{(X)}}{\beta+\gamma_i^{(X)}}\right\}$$

とおくと、

$$T \geq \min\{t_i\}$$

を満たさなければならない。

3.1.1 リレーのないマスタープロセッサ

プロセッサ P_i がマスタープロセッサであるとする。マスターがクラスタ内の他のプロセッサに仕事を委託する意味があるのは

$$\gamma_i > \beta_j\gamma_i^{(I)}/(\beta_j+\gamma_i^{(I)})$$

の場合に限る。そこで、以下ではこの条件を満たさない場合には $y_i^{(I)}=0$ であるとする。同様に、クラスタ外のプロセッサに仕事を委託する意味があるのは

$$\gamma_i > \beta\gamma_i^{(X)}/(\beta+\gamma_i^{(X)})$$

の場合に限る。そこで、この条件を満たさない場合には $y_i^{(X)}=0$ であるとする。ラウンド時間 T が前段落の条件 $T \geq \min\{t_i\}$ を満たす場合、プロセッサ P_i がマスターであるためには上記のいずれかの条件を満たしていなければならない。

マスターが他のプロセッサに委託するタスクの量は必要最小限でよい。この条件は

$$y_i^{(I)}\mu_i^{(I)} + y_i^{(X)}\mu_i^{(X)} = T/\gamma_i - x_i$$

と表現することができる。ここで

$$\mu_i^{(I)} = \beta_j\left(\frac{1}{\beta_j}+\frac{1}{\gamma_i^{(I)}}-\frac{1}{\gamma_i}\right)$$

$$\mu_i^{(X)} = \beta\left(\frac{1}{\beta}+\frac{1}{\gamma_i^{(X)}}-\frac{1}{\gamma_i}\right)$$

である。前段落の仮定から $y_i^{(I)} \neq 0$ なら $\mu_i^{(I)} > 0$, $y_i^{(X)} \neq 0$ なら $\mu_i^{(X)} > 0$ である。

ここで $\mu_i^{(I)} < \mu_i^{(X)}$ の場合、プロセッサ P_i はクラスタ内からタスクを受け取るよりも、クラスタ外からタスクを受け取った方が効率よく処理ができることになる。これは直感に反するし、スケジューリング上も難しい問題を提起するので、今回は $\mu_i^{(X)} \leq \mu_i^{(I)}$ がすべてのプロセッサ P_i について成り立つと仮定する。

また、通信時間について考えると

$$-y_i^{(I)}\beta_j - y_i^{(X)}\beta \leq T$$

が必要条件となる。

3.1.2 リレーのないワーカープロセッサ

ワーカープロセッサについても同様に分析をすることができる。受け入れることができるタスク量は自分の持っていたタスク量とあわせて T 以内で処理できる量である必要がある。これは

$$y_i^{(I)}\omega_i^{(I)} + y_i^{(X)}\omega_i^{(X)} \leq T/\gamma_i - x_i$$

と書くことができる。ここで

$$\omega_i^{(I)} = \beta_j\left(\frac{1}{\beta_j}-\frac{1}{\gamma_i^{(I)}}+\frac{1}{\gamma_i}\right)$$

$$\omega_i^{(X)} = \beta\left(\frac{1}{\beta}-\frac{1}{\gamma_i^{(X)}}+\frac{1}{\gamma_i}\right)$$

である。自然な仮定として $\gamma_i \leq \gamma_i^{(I)}$ および $\gamma_i \leq \gamma_i^{(X)}$ と考えられるので、$\omega_i^{(I)} > 0$ および $\omega_i^{(X)} > 0$ が常に成立する。また、同じ仮定の下で $\mu_i^{(I)} \leq \omega_i^{(I)}$ および $\mu_i^{(X)} \leq \omega_i^{(X)}$ が成立する。

さらに、マスターと同様の考察により、$\omega_i^{(I)} \leq \omega_i^{(X)}$ がすべての P_i に対して成り立つと仮定する。これはマスターのところで導入した $\mu_i^{(X)} \leq \mu_i^{(I)}$ と同値である。

通信時間の制約は

$$y_i^{(I)}\beta_j + y_i^{(X)}\beta \leq T$$

である。

3.1.3 クラスタ内のタスク収支

以上の結果から、

$$\bar{y}_i^{(I)} = \begin{cases} (T/\gamma_i - x_i)/\mu_i^{(I)} & \text{for master processors} \\ (T/\gamma_i - x_i)/\omega_i^{(I)} & \text{for worker processors} \end{cases}$$

と定義すると、マスター・ワーカーともに

$$y_i^{(I)} \leq \bar{y}_i^{(I)}$$

を満たさなければならないことがわかる。そこで

$$\bar{Y}_j^{(I)} = \sum_{P_i \in C_j} \bar{y}_i^{(I)}$$

を定義する。すると、$\bar{Y}_j^{(I)} < 0$ なるクラスタは、クラスタ内でタスクのやり取りを行っても処理しきれないタスクが残るので「マスタークラスタ」となる。また、$\bar{Y}_j^{(I)} > 0$ となるクラスタは、クラスタ内でタスクのやり取りを行っても余裕のあるワーカープロセッサが残るので「ワーカークラスタ」となる。

3.2 クラスタ間通信とリレー

クラスタ間通信は均一な性能なので、クラスタ単位で見るとリレーなしの仮定を置くことができる（但し $\mu_i^{(X)} \leq \mu_i^{(I)}$ および $\omega_i^{(I)} \leq \omega_i^{(X)}$ が必要である）。この仮定により、マスタークラスタ内のすべてのプロセッサに対して $y_i^{(X)} \leq 0$ であり、ワーカークラスタ内のすべてのプロセッサに対して $y_i^{(X)} \geq 0$ であると仮定することができる。

以下、まずマスタークラスタを仮定する。すなわち $\bar{Y}_j^{(I)} < 0$ である。ここで

$$Y_j^{(I)} = \sum_{P_i \in C_j} y_i^{(I)}$$

を定義すると、クラスタ内でのタスク移送の過不足がなくなるために $Y_j^{(I)} = 0$ としなければならない。他のクラスタに送出するタスク総量は

$$Y_j^{(X)} = \sum_{P_i \in C_j} y_i^{(X)}$$

であるが、これは最小限に抑えたい。この問題を、$y_i^{(I)} =$

$\bar{y}_i^{(I)}$ および $y_i^{(X)} = 0$ からスタートして、$y_i^{(I)}$ を増やし $y_i^{(X)}$ を減らすことにより達成する、という見方をすることにする。

3.2.1 マスタープロセッサのクラスタ間通信

マスタープロセッサのタスク量に関する制約は

$$y_i^{(I)}\mu_i^{(I)} + y_i^{(X)}\mu_i^{(X)} = T/\gamma_i - x_i$$

であった。上の式から、$y_i^{(I)} = \bar{y}_i^{(I)} + d_i$ とすると

$$-y_i^{(X)} = d_i\mu_i^{(I)}/\mu_i^{(X)}$$

を満たすことがわかる。

ここで、できるだけ少なく $Y_j^{(X)}$ を減らすことにより $Y_j^{(I)}$ を増やすということを考えると、$\mu_i^{(I)}/\mu_i^{(X)}$ が小さいプロセッサほど有利である。すなわち、$\mu_i^{(I)}/\mu_i^{(X)}$ が小さいものから順にマスタープロセッサをソートしておき、(a) 通信時間が T を超えない、(b) $y_i^{(I)} \leq 0$ を満たす、(c) $Y_j^{(I)} \leq 0$ を満たす、という 3 条件を満たす限り、クラスタ内通信 $y_i^{(I)}$ をクラスタ間通信 $y_i^{(X)}$ に移し変えればよい。

3.2.2 ワーカープロセッサのリレー

今回は通信が一様でないため、リレーが有利になる可能性がある。まず、マスタークラスタ中のワーカープロセッサによるリレーを考える。これは、ワーカープロセッサが本来処理できる以上の量をクラスタ内のマスタープロセッサから受け取り、処理しきれないタスクをクラスタ外のプロセッサに送出するというものである。あるマスターが非常に多くのタスクを持っており、性能の低いクラスタ間通信ではタスクを送出しきれないような場合に、一旦ワーカーに送ってそこから他のクラスタに送出することにより処理が可能となるような場合において有効である。

ワーカーがクラスタ内からの受信とクラスタ外への送信とを同時に行っている間の処理性能は $\gamma_i^{(D)}$ とする。今回は full duplex という仮定を設けているが、効率的なスケジューリングを可能にするためには、クラスタ内通信とクラスタ間通信が相互に干渉を起こさないことが望ましい。ここで、クラスタ間通信とクラスタ内通信を同時に行っている時間を τ とすると、タスク処理能力に関する制約条件が

$$\begin{aligned} x_i + & y_i^{(I)} + y_i^{(X)} \leq \\ & \tau/\gamma_i^{(D)} + (y_i^{(I)}\beta_j - \tau)/\gamma_i^{(I)} + (y_i^{(X)}\beta - \tau)/\gamma_i^{(X)} \\ & + (T - y_i^{(I)}\beta_j - y_i^{(X)}\beta + \tau)/\gamma_i \end{aligned}$$

となる。ここで τ の係数の和が 0 になれば、クラスタ間通信とクラスタ内通信とのスケジューリングは相互に影響なく自由に行うことができることになる。これは「一貫した計算性能低下」(前報告も参照)

$$\frac{1}{\gamma_i^{(D)}} = \frac{1}{\gamma_i} - \frac{1}{\gamma_i^{(I)}} - \frac{1}{\gamma_i^{(X)}}$$

を仮定することにより達成される。いささか都合のよい仮定であるが、これがなければ漸近最適スケジューリングを構築するのは非常に難しくなる。以下、これを (マスターを含む) すべてのプロセッサに仮定する。

さて、ワーカーのリレーでは $y_i^{(I)} > 0$ かつ $y_i^{(X)} < 0$ である。このときタスク処理量の制約式は

$$y_i^{(I)}\omega_i^{(I)} + y_i^{(X)}\mu_i^{(X)} \leq T/\gamma_i - x_i$$

となる。先ほどと同様に $y_i^{(I)} = \bar{y}_i^{(I)} + d_i$ とおくと

$$-y_i^{(X)} \geq d_i\omega_i^{(I)}/\mu_i^{(X)}$$

が得られる。

ここでワーカーの場合には $\mu_i^{(X)} \leq 0$ となる場合がありうるが、それはクラスタ外にタスクを送出すると内部通信まで増えてしまうことを意味する。従って、そのようなプロセッサはクラスタ間通信の候補に挙げる意味がない。そのようなプロセッサを除いた上で、$\omega_i^{(I)}/\mu_i^{(X)}$ が小さいプロセッサから優先的にリレーを行えばよい。

この係数はマスターの場合の $\mu_i^{(I)}/\mu_i^{(X)}$ と同じ意味を持っているから、マスター・ワーカーの両方をあわせてそれぞれの係数でソートしておき、小さいものから順に選択するのがよい。

3.2.3 マスタープロセッサのリレー

さらに、マスタープロセッサについてもリレーを考えることができる。マスタープロセッサ P_i のクラスタ内通信がすべてクラスタ間通信に置き換えられ、$y_i^{(I)} = 0$ になったとする。この時点でまだ P_i に通信をする余裕があれば、クラスタ内の他のマスターからタスクを受け取り、それをクラスタ外のプロセッサに送信することが可能である。これまでと同様に分析すると、$\omega_i^{(I)}/\mu_i^{(X)}$ が小さいものが有利であることがわかる。これはワーカーによるリレーとまったく同じ結果である。

以上をまとめると、マスターのクラスタ内通信からクラスタ間通信への変換、ワーカープロセッサのリレー、マスタープロセッサのリレーのすべてを係数 $\mu_i^{(I)}/\mu_i^{(X)}$ あるいは $\omega_i^{(I)}/\mu_i^{(X)}$ の小さいものから順に適用してゆけばよいことが分かる。このようにして、最小の $-Y_j^{(X)}$ で $Y_j^{(I)} = 0$ を達成するような $y_i^{(I)}, y_i^{(X)}$ が求められる。

以上はマスタークラスタの場合であったが、同様にしてワーカークラスタに対しては $Y_j^{(I)} = 0$ を達成するような最大の $Y_j^{(X)}$ を求めることができる。この最大値を $\bar{Y}_j^{(X)}$ とおくと、実際のマスタークラスタへのタスク流入量 $Y_j^{(X)}$ は $Y_j^{(X)} \leq \bar{Y}_j^{(X)}$ を満たす必要がある。

3.3 システム全体のタスク収支

以上のようにして、ラウンド時間 T が与えられると $Y_j^{(I)} = 0$ および $Y_j^{(X)} \leq \bar{Y}_j^{(X)}$ を必要条件とするような $\bar{Y}_j^{(X)}$ を、各クラスタに対して求めることができる。

ここでマスタークラスタに対して $-Y_j^{(X)}\beta > T$ であれば、クラスタ間通信時間が不足して実行不能になる。各クラスタ C_j について $-Y_j^{(X)}\beta = T$ となる T が求まるので、これを T_j とする。するとラウンド時間 T は

$$T \geq T_L = \min\{T_j, t_i\}$$

を満たさなければならない。

ワーカークラスタでも同様で、$\bar{Y}_j^{(X)}\beta > T$ であれば、クラスタ間通信が遅いために実際には $\bar{Y}_j^{(X)}$ だけのタスクを受け入れることができない。そこで

$$\bar{Y}_j = \min\{\bar{Y}_j^{(X)}, T/\beta\}$$

を定義すると、実際に受け入れられるタスク量は $Y_j^{(X)} \leq \bar{Y}_j$ で抑えられる。マスタークラスタに対して $\bar{Y}_j = Y_j^{(X)}$ とすれば、上の式はマスタークラスタに対しても成立する。

もし $T = T_L$ において $\bar{Y} = \sum_j \bar{Y}_j \geq 0$ であれば、マスタークラスタからあふれるタスクをワーカークラスタですべて吸収することができる。従って、この T_L が最小ラウンド時間となる。

もし $T = T_L$ において $\bar{Y} < 0$ であれば、マスタークラスタからあふれるタスクをワーカークラスタで吸収しきれない。$\bar{Y}$ は T に関して連続な単調増加関数なので、$\bar{Y} = 0$ となる T が存在し、これが最小ラウンド時間となる。

3.4 最小ラウンド時間の計算の計算量

次に、最小ラウンド時間 T を求めるために必要な計算量を考える。ひとつ T を与えると、各クラスタ C_j に対して $\bar{Y}_j$ が求まるが、この計算のためには (a) 各プロセッサについて $\bar{y}_i^{(I)}$ を計算、(b) $\bar{Y}_j^{(I)}$ を求め、マスタークラスタかワーカークラスタかを判定、(c) 係数 $\mu_i^{(I)}/\mu_i^{(X)}$ および $\omega_i^{(I)}/\mu_i^{(X)}$ でプロセッサをソートし、$y_i^{(I)}$ と $y_i^{(X)}$ を順次更新してゆく、という処理が必要である。ここで (c) のソートは最初に一度だけ行えばよいので、T が与えられたときに $\bar{Y}_j$ を求める計算の計算量はクラスタ内のプロセッサ数を $|C_j|$ とすると $O(|C_j|)$ であることがわかる。すべてのクラスタについてこれを行うと、$\bar{Y}$ の計算に必要な計算量はプロセッサ数を P とすると $O(P)$ となることがわかる。

次に $\bar{Y} = 0$ となる T を求めるアルゴリズムを考える。ここで $\bar{Y}$ は区分的線形関数であることを利用する。まず折れ点として各プロセッサがマスターとワーカーの間で切り替わるラウンド時間と、各クラスタがマスターとワーカーの間で切り替わるラウンド時間をすべて数え上げる。前者の個数は $O(P)$ で数え上げも $O(P)$ の計算量でできき、後者も $O(P)$ で数え上げは $O(P\log P)$ の計算量でできる（前報告のアルゴリズムを改変する）。これらの折れ点を $O(P\log P)$ の計算量でソートして、二分法で $\bar{Y} = 0$ なる T を 2 つの折れ点間に追い込めば、$O(P\log P)$ の計算量ですむ。

この 2 つの折れ点間では、プロセッサとクラスタのマスター・ワーカー分類が一定となるので、折れ点は 3.2 節のアルゴリズムにおいて $\bar{Y}_j^{(X)}$ を求める際のプロセッサのリストのどこまでが選ばれるかが切り替わるところで生じる。これは $O(P)$ 個であり $O(P)$ の時間で求めることができる。これらの折れ点を $O(P\log P)$ の計算量でソートして、二分法で $\bar{Y} = 0$ なる T を 2 つの折れ点間に追い込むのに $O(P\log P)$ の計算量ですむ。

この 2 点間では $\bar{Y}$ は線形なので、$\bar{Y} = 0$ なる T は $O(1)$ で求められる。結局、最小ラウンド時間は $O(P\log P)$ の計算量で求めることができることがわかる。

4. 十分条件：スケジューリング

以上で、スケジューリングが可能となるための各種の必要条件をすべて満たす最小のラウンド時間 T が計算できる。これに対して、このラウンド時間 T で実際にスケジューリングができることを示せば、定常ラウンドの最適スケジューリングを与えたことになる。

あるラウンドの計算に必要となる通信は前のラウンドで行うことにする。従って、処理と通信スケジューリングに干渉はない。また、リレーは 2 つの通信からなっているが、必要であればこれらを異なるラウンドで行うことにする。これにより、通信間の依存性もラウンド内では考慮する必要がなくなる。すなわち、必要な量の通信がラウンド内にスケジューリングできれば十分である。

4.1 クラスタ間通信スケジューリング

まず、クラスタ単位で通信スケジューリングを行う。クラスタ間通信は性能が一様でリレーなしが仮定されているため、前報告の通信スケジューリングアルゴリズムがそのまま使える。

次に、クラスタ間通信のプロセッサ間通信へのマッピングを行う。クラスタ内でどのプロセッサがどれだけクラスタ間通信を行うかはあらかじめ決められているので、クラスタ内通信のスケジューリングに先立って行うことにさえすれば、任意に行うことができる。クラスタのクラスタ間通信量をクラスタ内のプロセッサのクラスタ間通信量に合わせて分割し、通信時間帯とともに割り当ててやればよい。

4.2 クラスタ内通信スケジューリング

次に、クラスタ内通信のスケジューリングについて考える。以下マスタークラスタを仮定するが、ワーカークラスタについてもまったく同様である。

マスタークラスタの場合、クラスタ外への通信はタスクの送出のみである。今回は full duplex と一貫した計算性能低下を仮定しているため、ワーカープロセッサ（リレーを行うマスタープロセッサを含む、以下同様）の受信スケジューリングはクラスタ間通信の影響を受けない。一方で、マスタープロセッサ（リレーを行うマスタープロセッサを除く、以下同様）はクラスタ間通信を行っている場合、single port の仮定を置いているため、その時間帯だけクラスタ内通信のスケジューリングができない。しかし、以下に示すように、この制約を満たすようなスケジューリングは常に可能である。

4.2.1 マスタープロセッサの通信スケジューリング

まず、制約の強いマスタープロセッサについて通信スケジューリングを行う。

クラスタ内ネットワークはクロスバースイッチを仮定し

ているため、複数の通信が同時に行われることがある。ある時点で同時に行われている通信（送受信のペア）の数を、クラスタ内通信の「並列度」と呼ぶことにする。並列度は時刻によって異なりうるが、ラウンド中で並列度が最大になる時刻（このときの並列度を「最大並列度」とよぶ）と、並列度が最小になる時刻（このときの並列度を「最小並列度」とよぶ）で、その並列度の差が 1 以下となるような、マスタープロセッサの通信スケジュールを以下のように構成する。

通信スケジューリングは各マスタープロセッサに対して順次決めてゆく。その際、あるマスタープロセッサ（とそれ以前のマスタープロセッサ）に対する通信スケジューリングが決められたところで定義される、部分的なクラスタ内通信スケジュールにおいて、最大並列度と最小並列度の差が常に 1 以下となるようにする。

それぞれのマスタープロセッサに対する通信スケジューリングを行う際に、まず、ラウンド中の時間帯に対して次のように優先順位を決める。(a) 並列度が最小並列度よりも大きい時間帯には最低の優先度を与える。(b) 並列度が最小並列度に一致する時間帯のうち、まだスケジューリングされていないマスタープロセッサがクラスタ間通信の時間帯に重なる時間帯には高い優先度を与える。さらに、まだスケジューリングされていないマスタープロセッサが複数ある場合、先にスケジューリングされるマスタープロセッサのクラスタ間通信の時間帯に重なる時間帯ほど高い優先順位を与える。このようにして決まった優先度の高い時間帯から順次クラスタ内通信時間を割り振る（ただし、クラスタ間通信を行っている時間帯にはクラスタ内通信時間を割り振ることはできない）。

あるマスタープロセッサがクラスタ間通信を行う時間帯は、それ以前のマスタープロセッサの通信スケジューリングにおいて最大の優先度を持っている。従って、このプロセッサのクラスタ間通信の時間帯の内に最小並列度の時間帯が含まれているとすると、クラスタ間通信を行わない時間帯の並列度はすべて最小並列度に一致しなければならない。従って、これらの時間帯すべてにクラスタ内通信をスケジューリングしても、並列度の差は 1 以下に保たれる。クラスタ間通信の時間帯の中に最小並列度の時間帯が含まれていない場合は、最小並列度の時間帯に優先的に通信がスケジューリングされるので、並列度の差は 1 以下に保たれる。

4.2.2 ワーカープロセッサの通信スケジューリング

次に、並列度の差が 1 以下であるようなクラスタ内通信（送信）スケジュールが与えられたとき、これを受信できるようなワーカープロセッサの通信スケジューリングができることを示す。

一貫した計算性能低下と full duplex の仮定のおかげで、ワーカープロセッサのクラスタ内通信のスケジューリングには制約がない。そこで、ワーカープロセッサを適当な順序でならべて、順に受信のスケジューリングを決めてゆく。その際、まだ受信スケジューリングが決まっていないクラスタ内通信の集合を考え、それにおける最大並列度の時間帯に優先的に通信をスケジューリングする。これにより、各ワーカープロセッサの通信スケジューリングが決まったところで、並列度の差が最小である 1 以下に保たれるのは明らかである。

以上のようにして、すべてのプロセッサに対するクラスタ間通信、クラスタ内通信のスケジューリングが決定できる。上記の説明では通信時間帯だけが決まっていて通信相手が決まっていないが、任意の時刻で送信者数と受信者数はバランスしているので、通信相手は適当にマッチングさせればよい。以上のスケジューリングのアルゴリズムの計算量は $O(P)$ である。このスケジューリング方式は、前報告の「区間交差マッチング」に比べると通信の回数が相当に増えるが、漸近最適性には影響しない。

5. まとめと考察

本報告では、マルチクラスタ上での MMDL 問題の一つを取り上げ、それに対する漸近最適スケジューリングを導いた。今回の仮定のポイントは (a) 複数のクラスタをひとつのクロスバースイッチで接続した二階層マルチクラスタ、(b) クラスタ内通信は（クラスタごとに決まった）一様な性能、(c) クラスタ間通信も一様だが、どのクラスタ内通信よりも遅い、(d) 各プロセッサの通信は single port の full duplex、(e) 結果の収集はしない、のようにまとめられる。提案したアルゴリズムは上記の問題設定に対して $O(P \log P)$ の計算量で漸近最適スケジュールを求めることができる。なお、送信時と受信時で $\gamma_i^{(I)}$ や $\gamma_i^{(X)}$ が同じであるという仮定は用いていない。

Half duplex あるいは結果の収集が必要な場合には、本稿で論じたような手法で決めたラウンド時間内にスケジューリングができないような場合が生じるらしい。これには通信と計算がオーバーラップできないプロセッサのモデルも含まれる（full duplex と一貫した計算性能低下の仮定に矛盾する）。このことは、通信性能のヘテロ性というものが MMDL のような非常にシンプルなモデルにおいても扱いにくいものであるということを示している。

謝　　辞

本研究は文部科学省の 21 世紀 COE プロジェクトと科学研究費補助金による。

参 考 文 献

1) 須田礼仁、「Multi-master divisible load model における漸近最適スケジューリング」、情報処理学会研究報告 2003-ARC-157/2003-HPC-97, 2004 年 3 月, pp. 97–102.
2) T. G. Robertazzi, *Ten Reasons to Use Divisible Load Theory*, IEEE Computer, Vol. 36, No. 5, May 2003, pp. 63–68.