小特集

並列・分散処理のためのスケジューリング理論

須田礼仁*

ABSTRACT This paper provides a brief introduction to task scheduling algorithms, especially those with theoretical performance guarantee, for parallel and distributed processing. Some results of classical task models and divisible load models are shown. Also some of our research results on asymptotically optimum load redistribution algorithms are presented.

1. はじめに

最近の急速なマルチコアCPUの普及には目を見張るものがあり，並列処理の新しい時代の幕開けを感じるところである．ゲームやグラフィクス用のプロセッサは内部に多数の演算装置を持っており，その計算能力を科学技術計算に応用するための研究も精力的に行われている．また，宇宙科学や分子動力学などを主たるターゲットとして，汎用のコンピュータに特殊な超高性能演算装置を付加するアーキテクチャも注目を集めている．他方，たゆまぬ努力に支えられ，計算機ネットワークの性能は堅調に伸びており，これを背景としてグリッドに代表される広域的な分散計算システムも実証段階に入りつつある．消費電力の問題もあってCPUクロック周波数の劇的な伸びは期待できないため，今後，計算性能の向上の大半は並列アーキテクチャの高度化に依存するものと思われる．

単体プログラムの並列化は高性能な科学技術計算の基本であり，将来においてもその重要性は変わらないであろう．しかし上述のように並列アーキテクチャが多様化し，CPU内から，チップ内，筐体内，システム内，計算センター内，そしてグローバルネットワークに至る，さまざまな階層においてそれぞれの並列性が展開されると，MPIやOpenMPを用いて単体プログラムを並列化するだけでは，計算システムが持つ能力を利用しきれなくなる．アーキテクチャの並列性階層に応じて，ソフトウェアでも階層的な並列性を見出して利用することが必要となっているのである．

近年，並列処理においてタスクレベルの並列性に関する研究をよく目にするようになった．その背景として，グリッド技術の発展，マルチフィジックスなどソフトウェアの階層性の深まり，単体プロセッサ性能の向上による単体プログラム性能に対する満足度の高まりなどが考えられる．上述のような並列アーキテクチャの展開を考えると，今後は高性能プログラムに欠かせない並列化階層のひとつとして，タスクレベル並列性はその地位を確立してゆくものと思われる．

タスクレベルの並列処理の基本的な要素技術がスケジューリングである．タスクのスケジューリングとは，各タスクをどのプロセッサに割り当てて，どのような順序で実行するかを決定することである．本稿ではタスクスケジューリングのアルゴリズムについて，理論的な側面に重点をおいて紹介する．

本稿は21世紀COEプログラム「情報科学技術戦略コア」の3つの融合プロジェクトのひとつである「超ロバスト計算原理プロジェクト」の成果の一部にもとづいている．我々は，従来の並列処理の多くが仮定していたプロセッサの均一性や占有使用の仮定を外し，多様化する並列アーキテクチャに対応できる並列処理手法の開発を目指してきた．本稿でも非一様なプロセッサからなるシステムに適用できる手法を取り上げる．前述のような並列アーキテクチャの多様化を考慮すると，本稿で紹介するような非一様なプロセッサと実効性能の動的な変化に適応できるようなタスクスケジューリングアルゴリズムが今後は重要になってくるであろう．

Scheduling Theory for Parallel and Distributed Processing. By *Reiji Suda* (Graduate School of Information Science and Technology, The University of Tokyo).

* 東京大学大学院情報理工学系研究科

2. スケジューリング理論

並列処理の本格的な広まりが比較的最近であるのに比べると，タスクスケジューリングの研究は意外と長い歴史を持っている．*Graham*らによる1979年のサーベイ[1]は初期の研究の集大成であり，その後の研究の出発点ともなった重要な論文であるが，すでに非一様なプロセッサを対象とした並列プログラムのタスクスケジューリングについてかなりの研究成果があったことがわかる．その後も着実に研究は発展しており，限られた紙数ではその広がりと深さを垣間見ることすら難しい．ここでは著者が昨年まとめる機会を得たサーベイ[2]にもとづきその一端を紹介する．本章では紙数の節約のために論文の引用をしないので，興味を持たれた読者は上記のサーベイを参照していただきたい．

第i番目のタスクを第p番目のプロセッサで処理したときの所要時間をt_{ip}とする．これは何らかの手段で知られているとする．所要時間t_{ip}がプロセッサpによらないとき，identicalなプロセッサであるという．これは均一なプロセッサからなる並列システムに対応する．不均一な場合は次のように2通りに分かれる．所要時間が$t_{ip} = w_i / s_p$と書けるとき，uniformなプロセッサであるという．このときw_iはタスクiの仕事量，s_pはプロセッサpの速度と考えればよい．所要時間が一般の場合は，unrelatedなプロセッサであるという．所要時間t_{ip}を要素とする行列を考えると，その階数が1の場合がuniformであり，そのような制約がない場合がunrelatedである．このuniform, unrelatedという用語が不自然に思われる読者もおられると思うが，スケジューリング分野では定着しているので本稿でもこのまま採用する．

タスクの実行順序に何の制約もないとき，独立であるという．割り当て変数x_{ip}を，タスクiをプロセッサpに割り当てるとき$x_{ip} = 1$，それ以外のとき$x_{ip} = 0$と定義すると，独立タスク問題のスケジュール長は$C = \max_{1 \le p \le m} \{\Sigma_{i=1}^{n} x_{ip} t_{ip}\}$となる．ここで$m$はプロセッサ数，$n$はタスク数である．この値はプロセッサごとの計算時間の合計の最大値であり，タスクの配送や通信のコストはまったく入っていない，いわばもっとも簡単な問題設定である．しかし，独立タスクでidenticalなプロセッサの問題ですら強NP困難であることが古くから知られており，大規模問題に対して最適解を得ることは現実的ではない．

よって研究のほとんどは近似的な最適化を目指すことになる．ここで性能の指標となるのが最悪性能比という概念である．最短スケジュール長をC_{opt}，あるアルゴリズムにより得られる近似解のスケジュール長をC_{app}とするとき，その比C_{app} / C_{opt}の上限をそのアルゴリズムの最悪性能比という．すなわち最悪性能比がσの場合，どんな問題に対しても最適解のσ倍以下のスケジュール長となることが保証されている．Uniformなプロセッサでの独立タスク問題に対しては，任意の$\varepsilon > 0$に対して最悪性能比が$1 + \varepsilon$以下となる多項式時間アルゴリズムが構成できる．しかしこの場合は$1/\varepsilon$に関して指数的な計算時間がかかるため，高い精度の近似解を求めることは現実的ではない．プロセッサ数mを定数とすると，$1/\varepsilon$に関しても多項式オーダーとなるアルゴリズムが構成できるが，今度はプロセッサ数mに関して指数的な時間がかかるので，大規模な並列処理には向かない．Unrelatedなプロセッサではさらに問題が難しく，最悪性能比が3/2未満の多項式時間近似アルゴリズムは存在しないこと(P≠NPであれば)が証明されている．

Uniformなプロセッサでは，LPTと呼ばれる簡単なアルゴリズムの最悪性能比が1.584以下であることが知られており，最悪性能比が1.382以下のMULTIFITというアルゴリズムも十分実用的である．Unrelatedなプロセッサでは最悪性能比が2のアルゴリズムが知られているが，アルゴリズムは相当複雑である．簡単なアルゴリズムとしては，*Davis*と*Jaffe*による最悪性能比が$2\sqrt{m}$(mはプロセッサ数)のアルゴリズムがある．

タスクの処理を途中で中断して，他のプロセッサに移送して残りの処理を行うことができる場合，プリエンプティブなタスクという．プリエンプティブな独立タスクの問題に対しては，プロセッサがuniformでもunrelatedでも最適解が多項式時間で求められる．

このほか，タスクの実行に先行制約があるDAGスケジューリング，実行時にタスクの情報が与えられるオンラインスケジューリング，個々のタスクが複数のプロセッサを同時に利用するマルチプロセッサタスクなど，多様な問題設定のもとで研究が展開されており，非常に興味深い多数のアルゴリズムが提案されている．

しかし，これらの研究における問題設定には，ある重要な要素が欠けている．それは通信時間である．たとえタスクが独立であっても，最初から理想的な場所にタスクが存在するとは限らないので，タスクを処理プロセッサに配送するために通信時間がかかるはずである．Identicalなプロセッサに対しては通信を考慮したスケジューリングの研究もあるが，不均一なプロセッサに対しては，理論的な保証のある結果はほとん

ど存在しない.

3. Divisible Load Theory

3.1 Divisible loadのモデル

プロセッサが不均一な問題に対して通信を考慮したスケジューリングアルゴリズムが少ないのは，通信を含まなくても十分問題が難しいことがひとつの原因と考えられる．それでは通信を考慮する代わりに，タスクのモデルを最大限簡単にしたらどうなるであろうか．その答えのひとつがdivisible loadと呼ばれるタスクモデルである．なお，定着した和訳がなく，綴ると長く読みにくいので，以下では比較的通用する略語であるDLT(Divisible Load Theory)を用いることにする．なお，前述のサーベイにもDLTの紹介がある．

DLTのモデルでは，タスクは任意のサイズに分割することができ，それぞれの部分タスクは任意のプロセッサで独立に処理することができる．これは，計算内容が同一の独立なタスクが非常に多数ある場合を近似していると考えればよい．タスクが均一なので，プロセッサの不均一性はuniformに限られる．非常に単純なタスクモデルであるが，さまざまな応用問題に適用されていたり，類似の結果が繰り返し再発見されたりしていることから，研究する価値が十分にあるモデルであると言えるだろう．

DLTの研究の多くでは，マスタ・ワーカ(マスタ・スレーブともいう)が仮定されている．すなわちタスクは最初すべてマスタと呼ばれる1台のプロセッサにあって，そこからワーカと呼ばれる他のプロセッサに配送されて処理される．マスタ自身が処理をするかどうか，通信と処理が同時に実行できるか否か，マスタとワーカをつなぐネットワークがどのようなトポロジーになっているか，処理の結果をマスタに返す必要があるかどうか，通信の処理の所要時間をどのようにモデル化するか，などの問題設定は多様なものが提案され解かれている．なお，本稿では，マスタ自身も処理を行う，通信と処理は同時にできない，各プロセッサは複数の通信を同時にはできない，すべてのプロセッサが直接通信できる，処理の結果は集めなくてもよい，と仮定する．

3.2 DLTによるマスタ・ワーカのスケジューリング例

図1に簡単な例を示す．ワーカは2台で，マスタとワーカの処理速度は同じとする．(a)は，タスクを3等分し2台のワーカに順次送るスケジュールを示している．すると，先に送られたワーカは他の2台より早く処理が開始できるため，処理の終了も早くなる．他の2台が処理を続けている間このワーカはアイドル状態であり，無駄がある．そこで(b)では，タスクの分割サイズを工夫してすべてのワーカが同時に終了するようにしている．通信時間と計算時間がタスクのサイズに比例する場合は，(b)のようなスケジューリングの最適解は簡単に求めることができる．ところが(c)では，タスクをさらに2等分し，2回に分けてワーカに送信している．この場合，あとから受信するワーカが(b)のスケジュールより早く処理を開始できるので，さらにスケジュール長が短くなる．このように，マスタが同じワーカに複数回通信する方式をマルチラウンドと呼んでいる．この例で見るように，一般にはマルチラウンドにしないと最短スケジュール長は達成されない．

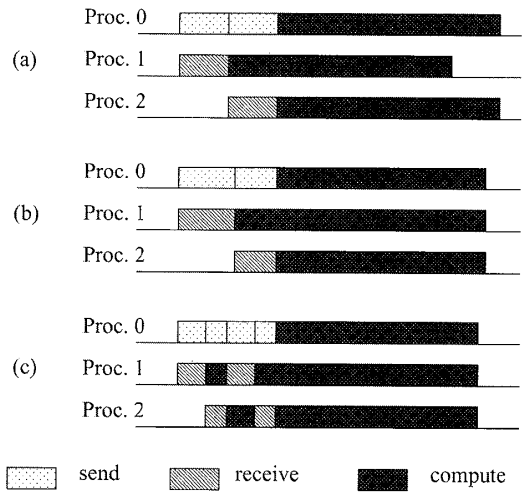

図1　マスタと2台のワーカでタスクを分配する3つのスケジューリング

図1の例の場合，通信時間と計算時間がタスクのサイズに比例すると仮定すると，タスクを細かく分けて通信回数を増やすほどスケジュール長が短くなり，通信回数が無限大の極限が最適解となる．このような非現実的な解が最適になるのは，小さなタスクの通信時間や計算時間がいくらでも小さくなるというモデルに問題がある．通信時間や計算時間のモデルとしては，(定数項)＋(タスクサイズに比例する項)というアフィンモデルの方が現実に近い．ところがアフィンモデルでは，DLTの最適スケジュールを求める問題はNP困難であることが知られている．

3.3 漸近最適スケジューリング

これに対し*Beaumont*らは論文3)などにおいて，**図2**のような規則的な構成法により，最適に近いスケジュールが求められることを示した．以下では我々が

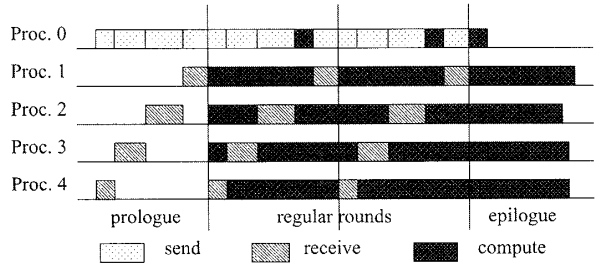

図2　定常ラウンドを反復するスケジューリング

行った，より簡明な解析を紹介する．

スケジュールの最初と最後以外は，定常ラウンドと呼ばれる同一のスケジュールを反復する．各定常ラウンドでワーカはタスクの受信と処理を行うが，処理するタスクは前のラウンドで受信しておくとする．スケジュールの最初と最後にこのルールを形式的に適用すると，スケジュールの最初(プロローグ)は最初の定常ラウンドで処理するタスクの通信だけ，最後(エピローグ)は最後の定常ラウンドで受信したタスクの処理だけとなる．

次に，定常ラウンドのスケジュールを以下のように決定する．通信時間と計算時間はアフィンモデルであるが，一旦定数項を忘れて，所要時間がタスクサイズに比例するモデルを考える．その上で，定常ラウンドの時間 t を与えて，ワーカが処理するタスクと受信するタスクのサイズが一致するという条件のもとで，時間 t の間にできるだけ多くのタスクを処理するようなスケジュールを作成する．紙数の都合で詳しい説明を省略するが，通信性能と計算性能がプロセッサごとに異なっている場合を含めて，これは簡単な線形計画問題になる．(計算性能によらず)通信バンド幅が広いワーカに優先的に仕事を割り当てるのが最適であることが証明できるので，プロセッサ数を m とすると $O(m \log m)$ の計算量でスケジュールを決めることができる．ここでは定数項を無視しているので，タスク量と時間を任意の定数倍しても最適性が保存される．そこで適当な定数倍をとれば，最初マスタにあるタスクすべてを処理できるような定常ラウンドの時間 T を求めることができる．定数項が実際に 0 の場合，このスケジュールを k 等分したスケジュールを定常ラウンドとするスケジュールを作成すれば，$k \to \infty$ の極限が最適スケジュールとなる．この極限のときのスケジュール長は T で，明らかにこれはアフィンモデルにおけるスケジュール長の下限となっている．

さて，定数項を復活させてアフィンモデルに戻ると，復活した定数項の分だけスケジュール長は伸びることになる．上記のように k 等分した定常ラウンドの時間は，定数項がなければ T/k であるので，定数項が復活するとある定数 A を用いて $T/k+A$ と表せる．この A は定常ラウンドに含まれる通信や計算の回数で決まる定数なので，k にはよらない．プロローグとエピローグの時間も $T/k+A$ でおさえられるので，全体のスケジュール長 C_{app} は $(T/k+A)(k+1)=T+T/k+A(k+1)$ 以下となる．

よって $k=\lfloor\sqrt{T/A}\rfloor$ とすると $C_{app} \leqq T+T/\sqrt{T/A}+A(\sqrt{T/A}+2) \leqq T+2\sqrt{AT}+2A$ が得られる．前述の通り最適スケジュール長を C_{opt} とすると $T \leqq C_{opt}$ なので，性能比 $C_{app}/C_{opt} \leqq 1+2\sqrt{A/T}+2A/T$ となる．タスク量が増加すると T は大きくなるが A は定数なので，$T \to \infty$ の極限で $C_{app}/C_{opt} \to 1$ となる．これを「漸近最適」と呼んでいるが，マシンを固定してタスク量だけを増やしているので，前述の最悪性能比とは異なる概念である．しかし，最適解との違いを定量的に評価できることは重要で，実用的にも価値がある．

3.4　データ再分散の漸近最適スケジューリング

さて，近年では並列システムが非占有的に用いられることも多い．そのようなシステムでは，他のユーザの影響で実効性能が動的に変化する．実効性能の変化率がある程度緩やかであれば，データの再分散による適応が有効である．

従来，データの再分散はそれに引き続く計算と分離して考えられていた．データ再分散の通信コストは純粋なオーバーヘッドとされ，計算負荷の完全均衡化を実現する最小コストの再分散が「最適な再分散」と考えられていた．しかし，この意味で最適化をすると，図1の(a)のような解が導かれる．図1の(b)および(c)のように，再分散の通信と引き続く計算とを一体的に最適化することにより，より効率的なデータ再分散が可能になるはずである．

我々はこのような考えにもとづき，データ再分散の問題をDLTでモデル化し，マルチラウンドの漸近最適スケジュールを導くことを提案した[4]．前述のDLTとの大きな違いは，マスタ・ワーカではなく，初期状態でタスクが各プロセッサに分配されており，任意のプロセッサの間で通信が行われ得るという点である．

我々は，通信バンド幅が均一な(計算性能は不均一でもよい)場合には，プロセッサ数 m に対し $O(m \log m)$ の計算量で漸近最適スケジュールが構築できることを示した．通信性能を均一と仮定したのは，これが不均一な場合には，あるプロセッサがタスクの受信と送信

を両方行うタスクのリレーが必要となり，問題の複雑さが高まるためである．しかし，内部的に通信性能が均一なクラスタを，異なる通信性能のネットワークで接続した，階層的なシステムの場合には，漸近最適解がやはり $O(m \log m)$ の計算量で求められる．さらに，**図3**のようにプロローグにも処理を導入し，各ラウンドのサイズを可変として，それを最適化する手法を提案した．これにより，漸近性能のみならず，現実的な問題サイズでも非常に良好なスケジュールを得られるようになった．

我々のこれらの手法の大きな特徴は少ない計算量である．実効性能の動的変化に追従するためのデータ分散を応用として念頭においているため，スケジューリングの計算量は低く抑えておきたいからである．しかも漸近最適性は確保しており，計算途中でスケジュール長の下限も得られているので，得られた近似解の品質(どのぐらい最適解に近いか)も定量的に評価できる．

また今回は紙数の都合で説明できなかったが，既存のデータ再分散手法の多くは通信スケジュールを具体的に作成していないのに対し，我々は衝突が生じないような通信のスケジュールをあたえている点も重要である．このため我々の手法では，予測性能と実性能が非常に近い．この意味では，性能の安定性が重要な，埋め込みシステムや実時間処理においても有用かもしれない．また，ここで利用している通信スケジューリング手法は，不均一なプロセッサ上での行列計算など，さまざまな場面で有用であることもわかっている．

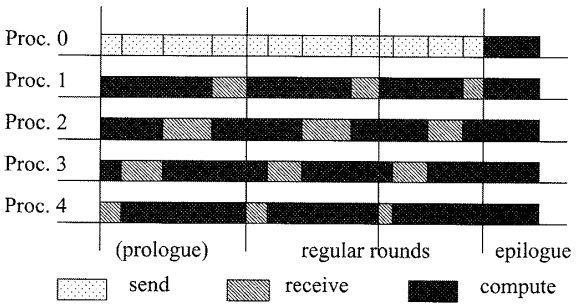

図3　図2のスケジューリングの改良

4. まとめ

本稿では，非一様なプロセッサからなる並列処理システムに使えるタスクスケジューリングアルゴリズムを，特に理論的な性能の保証が得られているものに着目して紹介した．古典的なモデルにおけるスケジューリング理論の成果の一端と，divisible load モデルのいくつかの結果しか紹介できなかったが，この分野の深さと応用上の重要性を少しでも感じていただければ幸いである．しかし一方で理論的な研究は理論に都合のよい問題設定と評価方法に限られる傾向にある．今後はもっと応用よりの視点で問題設定と評価方法を見直しつつ，研究をさらに展開させたいと考えている．

謝辞

COE プログラムにおいて本研究をお支えいただき，本稿の執筆をお勧めいただいた，杉原厚吉先生と原辰次先生に感謝を申し上げます．また多くの貴重なコメントをいただいた小柳義夫先生と，小柳研究室・須田研究室のメンバーに感謝いたします．本研究は文部科学省21世紀COEプログラムおよび科学研究費の援助を受けています．

参考文献

1) R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. R. Rinnooy Kan: Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of Discrete Mathematics, **5**, 287/326 (1979)
2) 須田：ヘテロ並列計算環境のためのタスクスケジューリング手法のサーベイ，情報処理学会論文誌：コンピューティングシステム，47-SIG18(ACS16)，92/114 (2006)
3) O. Beaumont, A. Legrand and Y. Robert: Scheduling divisible workloads on heterogeneous platforms, Parallel Computing, **29**-9, 1121/1152 (2003)
4) R. Suda and S. Tomi: Task redistribution scheduling using multi-master divisible load model, Proc. PDCS 2006, 160/165 (2006)