高性能コンピューティング特論

講義メモ（１）―科学技術計算と並列処理－

1. High Performance Computing

- 超並列処理システムが実用的に稼働するためには、ある程度の粒度を持つ並列処理単位と、十分な並列度が必要。
- 一般的なアプリケーションでは、必ずしもその要請が満たされない場合が多い。
- 大規模科学技術計算は多くの場合、双方の要請を満たす。また、それほど広範囲ではないが、ある程度の需要は常に存在する。（常に高性能化が要求される分野）
- 超並列処理システムは High-Performance Computing との親和性が良い。
- 実際に超並列処理システムを HPC において効率的に運用するための各種アルゴリズムが存在する。
- 多くの科学技術計算において、マトリクス演算は重要であるが、場合によっては超並列システムを最初から考慮し、アルゴリズムそのものを並列化することもある。これにより、一般に、超並列システムにおける効率は高くなる。

2. 大規模科学技術計算とその処理

- 代表的な大規模科学技術計算の例
  - 半導体（デバイス）シミュレーション
    ⇒移流拡散方程式
  - 計算化学、材料設計
    ⇒分子動力学(MD)、分子軌道法(MO)
  - 数値予報、数値風洞、計算流体力学
    ⇒Navier-Stokes 方程式
  - 構造解析
    ⇒有限要素法、応力解析
  - 計算物理、量子力学（第一原理計算)、天文
    ⇒QCD、波動方程式、電磁流体方程式、多体問題
- 問題とその解法
  - 偏微分方程式：
    ナビアストークス、移流拡散、熱伝導、ポアソン、シュレンディンガー、ボルツマン
  - 離散化：
    差分法、有限要素法、境界要素法、粒子法、モンテカルロ法

- 連立一次方程式の解法：
  ガウス消去法、コレスキー分解、スカイライン法、ガウスザイデル法、ヤコビ法、SOR 法、共役勾配法、マルチグリッド法、領域分割法

・ 代表的な方程式

- 熱伝導方程式

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}$$

- 移流拡散方程式

$$\frac{\partial u}{\partial t} + div(-k\nabla u + u\vec{v}) = f$$

- Poisson 方程式

$$\nabla^2 u = f$$

- Navier-Stokes 方程式

$$\rho\left[\frac{\partial V}{\partial t} + grad(\frac{V^2}{2}) + (rotV)\times V\right] = -grad(p + \rho q^2) + \mu\nabla^2 u$$

- 波動方程式

$$\frac{\partial^2 u}{\partial t^2} = \alpha^2 \frac{\partial^2 u}{\partial x^2}$$

3. 偏微分方程式の離散化（差分法）

偏微分方程式の要素としての 1 階微分、2 階微分を離散表現する。このために、連続空間における微分を、離散空間における差分に置き替える。

問題空間として、1 次元空間上の変数 u の差分を考える。空間座標 x は h 刻みに離散化されるものとする。ここで座標 $x0$ の h 近傍における Taylor 展開を考える。

$$u(x_0 + h) = u(x_0) + \frac{du}{dx}\bigg|_{x=x_0} h + \frac{1}{2}\frac{d^2u}{dx^2}\bigg|_{x=x_0} h^2 + \frac{1}{6}\frac{d^3u}{dx^3}\bigg|_{x=x_0} h^3 + \cdots$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} u^{(n)}(x_0) h^n$$

ここで 1 階微分を求めるために、上式の 2 次以降の項を省略する。

$$u(x_0 + h) = u(x_0) + u'(x_0)h$$

$$u'(x_0) = \frac{1}{h}\left[u(x_0 + h) - u(x_0)\right]$$

これを**前進差分**という。また、h に $-h$ を代入し、

$$u'(x_0) = \frac{1}{h}\left[u(x_0) - u(x_0 - h)\right]$$

これを**後退差分**という。さらに、両者の平均を取り、

$$u'(x_0) = \frac{1}{2h}\left[u(x_0 + h) - u(x_0 - h)\right]$$

これを**中心差分**という。

次に、2 階微分を求めるために、最初の式の 3 次以降の項を省略する。

$$u(x_0 + h) = u(x_0) + u'(x_0)h + \frac{1}{2}u''(x_0)h^2$$

同様に変形し、前進形及び後退形が求まる。

$$u''(x_0) = \frac{2}{h^2}\left[u(x_0 + h) - u(x_0) - u'(x_0)h\right] \qquad \text{（前進差分）}$$

$$u''(x_0) = \frac{2}{h^2}\left[u(x_0 - h) - u(x_0) + u'(x_0)h\right] \qquad \text{（後退差分）}$$

さらに上記 2 式の平均を取り、1 階微分を消去することにより中心形が求まる。

$$u''(x_0) = \frac{1}{h^2}\left[u(x_0 + h) - 2u(x_0) + u(x_0 - h)\right] \qquad \text{（中心差分）}$$

以上の離散化は、偏微分に関しても同様に有効である。このようにして偏微分方程式の微分項を離散化する方法を差分法と呼ぶ。

以上を全空間上の点に適応する。h 離れた離散点の座標を、序数 i をつけて x_i と表し、$u(x_i)$ を u_i と表すことにする。また、空間刻み幅 h を Δx と示すことにする。すると、例えば u の 2 階微分の中心差分形式は以下のように表すことができる。

$$u_i'' = \frac{1}{\Delta x^2}\left[u_{i+1} - 2u_i + u_{i-1}\right]$$

以上は空間微分に関する離散化であったが、時間微分項を含む方程式の場合、同様に時間に関しても離散化を行なう。そして、時間発展する変数を求めるために、時間微分項を積分する。この場合、積分の方法としては、単純な Euler 法、または精度を高めるために Runge-Kutta 法等が用いられる。

4. 陽解法と数値解析的安定性

例として以下の方程式の時間発展解を求める差分式を考える。

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

右辺の空間差分は前出の中心差分を用いる。左辺に関しては Euler 法を適用する。Euler 法では時間微分を使って変数 u を以下のような積分で表す。

$$u(t_0 + \Delta t) = u(t_0) + \frac{\partial u}{\partial t}\Delta t$$

ただし、Δt は時間刻みである。変数 u を時間発展的に求めるには、現在の u の時間微分を

用い、積分することによって次の時刻の u を求める、という作業を繰り返すことになる。このように、現在の時刻の変数から、次の時刻の変数を求めていく作業の繰り返しによって、変数値を計算していく方法を**陽解法**と呼ぶ。これに対し、次の時刻で各空間変数が満たすべき関係を連立方程式で表現し、その解を求めることによって現在の解を求める方法を**陰解法**と呼ぶ。

これらを用いると離散化された方程式より変数 u は以下のように表される。ただし、u_i^k という表現は、時刻 k における、座標 i の変数を表す。

$$u_i^{k+1} = u_i^k + \frac{\partial u_i}{\partial t}\Delta t$$
$$= u_i^k + \frac{\Delta t}{\Delta x^2}(u_{i+1}^k - 2u_i^k + u_{i-1}^k)$$

ここで注意すべきは、Δx 及び Δt の刻み幅の選択である。両者は密接に関係しており、例えば Δx を決定した時に、どのような Δt を選ぶかは非常に重要である。この方程式の場合、両者には以下の関係があることがわかっている。

$$\Delta t < \frac{1}{2}(\Delta x^2)$$

この関係を満たさないような大きな Δt を選択した場合、解が安定することは保証されない。具体的には、大きい Δt を使って上の計算ステップを実行すると、おそらく解は発散する。Δt をある程度以下に押さえることによって解の発散が防止される性質を、陽解法における数値解析的安定性と呼ぶ。

上記条件に基づいて計算量を見積ると深刻な問題にぶつかる。空間離散の精度を上げるために Δx を小さくすると、その 2 乗に比例して Δt を小さくしなければならなくなる。従って、全体の計算コストは、Δx の小ささの 3 乗に比例する。従って、空間精度を上げることは計算時間に非常に大きく影響してくる。

5. 陽解法と超並列処理

陽解法を用いることは、数値解析的安定性の問題を常に含んでおり、計算量の急激な増加を招く可能性がある。しかし、陽解法の各ステップにおける計算には**空間的並列性**という大きな魅力がある。すなわち、1 ステップの計算における各空間点の計算は、完全に並列処理することができ、例えば 2 次元問題で各次元を 1/100 に離散化したとしても、全体で 10000 の並列性が存在することになる。

また、直線状（1 次元の場合）、または平面格子状（2 次元の場合）といった、メッシュ構造のプロセッサ空間に対し、問題空間を自然にブロック分割してマッピングすることにより、隣接点同士間のみによる交信でデータ交換を実現できるという利点がある。

さらに、より細かい空間刻みを採用したい場合は、1 プロセッサ当たりの点数を増やすこ

とにより、プロセッサの逐次処理単位（粒度）を上げることが可能となる。全点のデータは同時に交換されるため、プロセッサ間でデータをブロック化し、まとめて交換することも可能で、その点では通信の粒度も上げることができる。

以上のように、偏微分方程式の陽解法による求解は、超並列処理システムにとって非常に高い親和性を持つ。数値解析的安定性の問題を各プロセッサの持つ高速性によってある程度克服できれば、陽解法は非常に有効な手段となる。

6. 陰解法と超並列処理

陰解法の場合、各空間点の計算は連立方程式の解を求める作業に帰着される。これは陽解法のような単純並列性を持たないため、並列処理には工夫が必要となる。しかし、数値解析的安定性の問題を逃れることができるという大きな魅力がある。

そこで、大規模連立 1 次方程式を効率的に解くという問題も、超並列処理システムにおいて非常に重要な問題となる。密行列に関しては、ガウス消去法が有効であるが、一般的な問題では行列は帯行列のような構造を持った疎行列、あるいは非零要素がランダムに分散する、一般疎行列になる場合が多い。こういう問題では、計算量を削減するために、直接法（例えばガウス消去法）は用いられない。なぜならば、計算を繰り返すことにより、行列中の当初零要素であった部分に非零要素が現れてくるからである（これを **fill-in** と呼ぶ)。

これに対し、反復法と呼ばれる方法（ヤコビ法、ガウス・ザイデル法、SOR 法、共役勾配法等）では、行列の形式を変形しないため、fill-in が発生せず、また各反復における並列性が一般に行列の非零要素数程度あるため、大規模行列の場合は超並列計算機による効率的な処理も期待できる。