分散アルゴリズムについて

山下　雅史

1. はじめに

計算機が互いに結合され，計算機システムが複雑化するにしたがって，さまざまな並列計算機システム向けのアルゴリズム論が展開されてきており，この並列アルゴリズム特集でも，共有メモリ結合型やハイパーキューブ等のトポロジーを持つメッセージ交換型のマルチプロセッサシステムをはじめ，スーパーコンピュータやシストリックアレー等の並列計算機システム用のアルゴリズムが紹介されている．ある特定の並列計算機システム用のアルゴリズムを論じるには，その並列計算機システムの特徴を反映した抽象的な並列計算機械を定義し，その上で問題解決を議論することになるが，このとき，

1. この並列計算機械のことを十分に知っている，
2. この並列計算機械を自由に使用できる，
3. 問題を解くために必要なデータはすでに揃っている，

等は自明な仮定として採用される．これらは，集中型システム上の計算の典型的な特徴である．

しかし実際には，これらの項目を仮定できないような計算機システムも存在する．たとえば，ＴＣＰ／ＩＰと呼ばれるプロトコルで結合されている広域（計算機）ネットワークを考えよう．これには世界中の15万台を超えるワークステーションが参加している．このように巨大なネットワークであるが，１つのシステムに組織し，利用することもしばしば行なわれる．たとえば，われわれは，この上に構築された電子メールシステムを利用することで，世界中のどの地点にもほとんど瞬時にメールを送りつけることができる．このネットワーク上で，たとえば，一定時間内に流れたある話題に関するメッセージ数を計数したいとしよう．このとき，

1. このネットワークの現在の姿について正確に知ることはできないし，
2. 各計算機はそれぞれ異なる方針で運営されており，それらを自由に利用することもできないし，また，
3. 問題を解決するのに必要なデータ（計算機が送信したメッセージ履歴）は，それぞれの計算機に分散している．

広域ネットワークのように，システム全体の状況をある特別な計算機（プロセス）が把握し，制御できないようなシステムを分散（型）システムと言い，分散システム用のアルゴリズムを分散アルゴリズムと呼ぶ．

2. 問　　題

並列アルゴリズムと分散アルゴリズムの問題設定の相違を，基本的なグラフ問題である，最小生成木問題（minimum spanning tree problem）を例にとって説明しよう．コスト付き無向グラフ $G=(V,E,c)$ が与えられたときに，コスト最小の生成木 $T=(V,E_T)$ を抽出する問題が最小生成木問題である．ここに，$c(e)(e\in E)$ は枝 e のコストを表わし，生成木 T のコスト $c(T)$ は

$$c(T)=\sum_{e\in E_T} c(e)$$

で与えられる．

最小生成木問題の並列アルゴリズムを議論する場合には，入力 G を与えることは，それを解くための計算機械 M と，その上で G がどのように保持されているか（あるいはどのように入力されるか）決めることである．計算終了時に，結果である T は，ある特定のメモリ（群）に指定した要領で格納されている（あるいは，計算終了までにある順序で出力する）．すなわち，アルゴリズムの設計者は，

1. M を熟知し，
2. M を自由に利用でき，
3. G を完全に知っている，

と仮定されている．

分散アルゴリズムにとっても，最小生成木問題は基本的な問題である．たとえば，ある計算機 w がネットワーク N に参加するすべての計算機にあるメッセージを伝える問題である，放送問題（broadcast problem）を考える（問題を解決したいと思い，アルゴリズムを起動する

やました　まさふみ　広島大学　工学部　第２類
〒724 東広島市鏡山町

計算機wを始動計算機（initiator）と呼ぶ．ネットワークNのトポロジーを$G=(V,E)$で表現する．ここで，Vは計算機の集合であり，$(u,v)\in E$は，計算機uとvが直接メッセージを交換できる，すなわち，uとvの間には双方向の通信リンクが存在する，ことを示す．次のようにして，放送問題を解くことができる．

1. 始動計算機wは接続するすべてのリンクに，伝えたいメッセージaを送信する．
2. メッセージaをリンクlから受信した計算機は，以前にaを受信したことがないならば，lを除く接続しているすべてのリンクにaを送信する．

このアルゴリズムでは最悪の場合$\Omega(|E|)$回のメッセージ送信が必要となる．分散システムではメッセージ通信は高価であると考えられているので，できれば$O(|V|)$回のメッセージ送信でこの問題を解決したい．

このために，$T=(V,E_T)$をGの生成木とし，各計算機uはuに接続するTの枝を認識していると仮定しよう（uはT全体を知っている必要はない）．このとき，上のアルゴリズムで，メッセージaの送信をTの枝だけに限定しても，放送問題は解決でき，しかも，メッセージ数は$O(|V|)$ですむ．なお，この例では，最小生成木がとりわけ必要ではなく，どの生成木でもかまわないのだが，たとえば，枝のコストとして通信にかかる費用を考えれば，最小生成木を用いる放送は，費用最小の放送という意味を持つ．

以上のような動機から最小生成木問題を考察しようとすると，問題例は次のように与えられるだろう．解くべき問題例であるコスト付きグラフ$G=(V,E,c)$は，その問題を解くアルゴリズムが実行される分散システムのトポロジーである．また，各計算機uは，最初，uに局所的な情報しか持たない．

並列アルゴリズムと分散アルゴリズムとは，ともに複数の計算機上のアルゴリズムという点で類似のアルゴリズム論のように錯覚しがちであるが，並列アルゴリズムでは，問題を（問題とは関係のない）並列計算機を利用して解決するのに対して，分散アルゴリズムでは（自分がその一員である），ネットワーク全体に関する問題を局所的な情報を交換して解決する，という意味で全く異なるものである．

“良い分散アルゴリズムがないのなら，どうして分散して解く必要があるのか？”ある研究会で耳にした質問である．分散アルゴリズムは，並列アルゴリズムのように，分散（並列化）して問題を解くためではなく，（すでに）分散されてある問題を解決するために開発されるのである．

本解説では，分散システム，分散問題や分散アルゴリズムの定義には立ち入らないので，これらの定義はたとえば，文献［1］を参考にされるようお願いする．

3. 時　間

分散システムは，いくつかの自律的なプロセッサとプロセスをサポートするデータ蓄積装置，かつ／またはデータベースシステムから構成されるもので，全体として1つの目標を達成するように協調動作するシステムである，と定義される［5］．プロセス間の情報の交換は通信ネットワークを介するメッセージ通信によって行なわれる．メッセージ転送遅延は常に存在し，有限であるが変化する．分散システムには，通信遅延が局所的な計算時間に比較して十分に長く，しかもそれを事前に見積もることが困難である，という特性がある（高速LANの上に作られた分散システムではこの性質が近似的にしか成立しないこともある）．通信遅延が存在するので，分散システムに属するすべてのプロセスがある時点における分散システム全体の状態を共通に知ることはできない．この点が分散アルゴリズムの設計と，対象の全体像が常に把握できる集中型システムで実行される通常のアルゴリズムの設計との本質的な相違である．

たとえば，あるプロセスiが，分散システムがデッドロックしているかどうか確認したいと考えたとしよう．このためには，分散システム全体の状態（大域状態）がわかればよい（大域状態を求める問題は大域的スナップショット問題（global snapshot problem）と呼ばれる）．もしも，分散システムを十分長い間止めることができるなら，この問題は簡単に解決できる．しかし，これは，不可能ではないとしても，全く現実的な解法ではない．分散システムを止めることなしに，プロセスiが他のプロセスの局所状態を回収し，それから大域状態を復元しようとすると，さまざまな時刻の局所状態を組み合わせることになり，“正しい”大域状態が復元できる保証はない[1]．これが，大域的スナップショット問題の困難な所である．このような問題は動的分散問題と呼ばれており，分散アルゴリズムの実行中に問題例が刻々変化する．

大域的スナップショット問題に戻って，各プロセスj

1）“正しい”という用語の正確な定義は意外に難しい．文献［3］を参照せよ．

が，大域的スナップショットを取る時刻Tをすでに知らされていると仮定しよう．このとき，各プロセスjが時刻Tの局所状態をiに伝えることで，問題が解決できるように思われる．しかし，プロセスjが参照する時計C_jが正確でない限り，問題が解決したことにはならない．そして，時計合わせは，非常に困難で，多くの場合，不可能な仕事なのである．たとえば，ある親時計Cに各局所時計C_jを合わせるというような単純な方法はうまくいかない．通信遅延が無視できず，しかもその値を事前に見積もることができないので，Cを管理するプロセスからCの現時刻tを送信してもらい，通信遅延の見積を加えてC_jにセットするというような方法は取れない[2]．

時間の役割は，

1. 一連の事象が起きた順序を定める規準を提供し，
2. 2つの事象の生起間隔を測る尺度を提供すること，

である．前段で述べたように，分散システムでは共通の時計を各プロセスが参照することが困難であり，したがって，(2)の目的で時間を提供することは困難である．(1)だけの目的で時間を提供する時計を論理時計（logical clock）という．もう少し，詳しく説明しよう．プロセスの実行過程を（生起した）事象の列で表現する．事象の集合の間に，次のようにして自然な半順序関係→が定義でき，これを前後関係（happened before relation）と呼ぶ．

1. 事象e_1とe_2が同じプロセスの事象ならば，e_1がe_2より先に生起したとき，かつそのときに限り，$e_1 \rightarrow e_2$.
2. 事象e_1と事象e_2がある同一のメッセージの送信および受信事象であれば$e_1 \rightarrow e_2$.
3. 事象e_1, e_2およびe_3に関して$e_1 \rightarrow e_2$，かつ$e_2 \rightarrow e_3$ならば$e_1 \rightarrow e_3$.

すなわち，考えられる限り最弱の前後関係から導かれる半順序関係が→である．プロセスjの時計C_jで計ったjの事象eが生起した時刻を$C_j(e)$で表現し，事象eがプロセスjの事象ならば$C(e)=C_j(e)$と定義することで，大域時計Cを定義する．Cは条件“$e_1 \rightarrow e_2$ならば$C(e_1)<C(e_2)$”を満たすならば論理時計である．論理時計によって，各プロセスは事象の前後関係について，矛盾なく知ることができる．Lamport［3］は，多くの目的の場合，分散システムの時計が果たす役割の本質的な部分が論理時間の提供であることを論じている．

論理時計は以下のようにして実現できる．各プロセスjは，

1. 受信事象以外の事象が生起した直後にC_jに1を加える．
2. • メッセージを送信するときには，タイムスタンプ$ts=C_j$をメッセージに付加して送信する．
   • タイムスタンプtsを持つメッセージを受信したときには，C_jを$\max\{C_j, ts\}+1$まで進める．この受信事象は更新された時刻C_jに生起したとする．

4. 知　識

分散アルゴリズムの対象となる問題は分散システムの大域的な知識にかかわる問題であり，ある問題を解くアルゴリズムの実行過程は，その問題を解決するために必要な知識を効率よく収集する過程と考えられる．よいアルゴリズムであればあるほど，少量の良質の知識を手際よく交換することによって問題を解決する．

2つのプロセスiとjを結ぶ通信リンクlは，信頼性が低く，メッセージがその中で消えるかもしれないと仮定する．通常のアルゴリズム論では，抽象計算機械は正常に作動すると仮定されている．丸め誤差等を考慮したアルゴリズムは研究されているが，たまに故障するRAMやPRAM上のアルゴリズムはあまり聞かない．一方，分散システムにはハードウェアやソフトウェアに起因する多くの故障の可能性が内在しており，しかもそれらの多くには再現性がないのでデバッグもままならないのが現実である．紙面の制約から解説する余裕がないが少々の故障に影響されることなく正しく問題を解決する耐故障アルゴリズム（fault tolerant algorithm）の検討は，この分野の重要なテーマの1つとなっている［4］．

簡単のために，リンクlの通信遅延が常に1秒であると仮定し，プロセスiがjに事実ϕを確実に伝えようと考えたとする．このためのアルゴリズムは以下のようなものであろう．

1. iは，jからメッセージackが到着するまで，ϕを運ぶメッセージを2秒間隔で繰り返し送信する．

2) 以下の2条件は時計を“近似的”に合わせるための十分条件である．
1. 任意の2つのプロセスの局所時計の1秒（ticking rate）の差異は定数で押さえられる．
2. 任意の2つのプロセスにおいて，それらの間の通信遅延のゆらぎは定数で押さえられる．

2. jはiからのメッセージが到着するたびに，*ack*を返送する．

このアルゴリズムRELIABLE-SENDによって，iはϕの送信を確実なものにできる．

もう少し複雑な問題を考えてみよう．iが事実ϕをjに送信し，iは“jがϕを知っていること”を，一方jは“iが“jがϕを知っているということ”を知っていること”を，それぞれ確信したいと考えたとする．このような状況は，iとjが同じデータを参照しているとき，iがそのデータに変更を加えようとして，jに了解を求めるときなどに現われる．プロセスhがある事実Ψを知っていることを$K_h\Psi$で表現することにする．このとき，

1. $K_j\phi$をiの知識にし，
2. $K_iK_j\phi$をjの知識とする，

ことが目的となる．

故障が起こらないというシナリオに沿って進んでみよう．iはϕ（運ぶメッセージ）を送信する．jはϕを受信し，$K_j\phi$を返送する．iは$K_j\Psi$を受信する．このとき，iは$K_j\phi$を確信できるが，$K_j\Psi$がiに到達したことをjは確信できないから，jは，$K_iK_j\phi$を確信できない．そこで，iは$K_j\Psi$を受信したことを伝えるために$K_iK_j\phi$をjに送信する．jは$K_iK_j\phi$を受信し，目的を達成できる．ここで，$K_h\Psi$とΨは異なる知識であることに注意してほしい．すなわち，たとえば$K_iK_jK_i\phi$と$K_i\phi$は異なる知識であり，混同は許されない．

なお，メッセージが脱落することがあっても最終的にはjが$K_iK_j\phi$を知ることができるようにするためにはメッセージ$\phi, K_j\phi, K_iK_j\phi$の送信を確実にする必要があり，このため，たとえば，アルゴリズムRELIABLE-SENDを適用する．したがって，*ack*どメッセージなをiとjの間で交換する必要があるが，省略する．

このように，問題を解決するために必要不可欠な知識を理論式で具体的に表現し，それらをメッセージに対応させることによって，アルゴリズムの構成を試みるのは，有力なアプローチの1つであり，このようにして作られたアルゴリズムを知識ベースプロトコルと呼ぶ．知識ベースプロトコルが用いる，$K_h\Psi$という形の式を扱う様相論理は知識論理（knowledge logic）と呼ばれ，上に述べたアルゴリズムの構成の側面を含め，さまざまな側面からの研究が進められている．知識論理では，$K_h\Psi \Rightarrow \Psi$，すなわち，成立しない事実を知っていることはない，ことを自然な公理として採用しているが，これを公理として採用しない様相論理体系として（正しくない事実の），思い込みも含めて扱う論理体系である信念論理（belief logic）がある［2］．

参考文献

［1］ 萩原，増澤：分散アルゴリズム，情報処理学会誌 31, 9 (1990) 1245-1256.

［2］ Halpern, J. Y., “Reasoning about knowledge : an overview”, *Proc. of the 1986 Gonference on Theoretical Aspects of Reasoning About Knowledge* (ed. J. Halpern) Monterey, Calfornia (March 19-22, 1986) Morgan Kaufmann, 1986.

［3］ Lamport, L., “Time, clocks and ordering of events in a distributed system”, *Comm. ACM 21*, 7 (1978) 558-564.

［4］ Lamport, L., Shostak, R. E., and Pease, M., “The Byzantine generals problem”, *ACM Transactions on Programming Language and Systems 4*, 3 (1982) 382-401.

［5］ Sloman, M., and Kramer, J. “Distributed Systems and Computer Networks”, Prentice-Hall International, Hemel Hempstead, Hertfordshire, U. K. (1987).（齊藤監訳：分散システムと計算機ネットワーク，丸善株式会社 (1988)）.