数理解析研究所講究録
第880巻 1994年 185-192

汎用超並列計算機の上での大規模数値計算
－ ＱＣＤ＿ＴＡＲＯ(+) プロジェクトの経験から －

山形大学・教育学部　中村純　(Atsushi NAKAMURA)

要旨

ベクトル型のスーパー・コンピュータしか使ったことのなかった素人のユーザーが経験した超並列計算の報告をする。単純な幾何学的分割とメッセージ・パッシングを使ってプログラミングを行なった。並列計算の概要を簡単に与えたのち、我々が遭遇したいくつかの問題（ゲージ系に固有の問題、繰り込み変換に伴う問題等）とその解決の試みを紹介する。現在の並列計算のユーザーから見た問題点を報告する。

１．序

並列計算機が新しいスーパー・コンピュータとして注目を集めている。計算機の専門家の間では、並列コンパイラー等の研究が意欲的に進められているが、実用化までの道はまだ遠いらしい。

しかし、我々のような大規模科学計算を進めているユーザーにとって、魅力的なピーク性能を持ったハードが既に存在しているのに、手をこまねいているのは辛い。ベクトル型スーパーコンピュータの出現によって、シミュレーションが科学を進める道具となることは確立しつつあるが、系のサイズ、統計精度が限られた結果が出始めると、これらを克服して、決定的な結果を出したくなる。(*) 比較的廉価で高いパーフォマンスが得られる並列計算機が素朴なプログラムで使えるものなら、どんどん計算を始めたい。

(+) ＱＣＤ on Thousand cell ARray processor for Omnipurposes. 千のセルを持った多目的プロセッサーの上でのＱＣＤ。ＱＣＤはQuantum Chromo-Dynamics（量子色力学）の略で、物質の究極の構成子と現在考えられているクォークとグルオンを支配する力学。

(*) われわれの取り扱うモデルは４次元空間 (x,y,z,t)の離散化に基づいている。今一辺に（わずか）３０点取ったとしても、10^6 の格子点が必要になる。各点にあるグルオン場は、４成分を持つベクトルで、各成分は３×３複素行列で表される。モンテカルロ計算で一点の更新に１００回の浮動少数点演算が現われるとして（実際はもっと多い）、（わずか）10^4 回のモンテカルロ・ステップを実行すると、１０００ギガの計算が必要となる。一辺の点の数を倍にすれば、格子点は１６倍になる。またモンテカルロ計算の統計誤差を半分にしようとすれば、４倍のステップ数が必要になる。

ここで報告するアプリケーション（格子ＱＣＤ）は、１つの計算に数千時間かかるので、ピーク性能に少しでも近い値を得ることが重要となる。また、これまでベクトル計算機で行なわれてきた計算が、超並列計算機を使っているからという理由で制限をうけては面白くない。

以上のような背景をもって進んでいる我々のプロジェクトでの経験を報告する。

２．幾何学的分割とメッセージ・パッシング

時空間の関数の微分方程式で表されるモデルなら、時空間を離散化することによって、差分方程式になる。対象の物体をいくつかの部分に分けて、それぞれを並列計算機を構成しているプロセッシング・エレメント（ＰＥ）に割り当てる。差分方程式なので、１つの点を計算するのに、その近傍の点が必要になる。境界の点に対しては、近傍の点が同じＰＥの上に無いことがある（図１）。その時は、必要な点の上での値をｓｅｎｄ，ｒｅｃｅｉｖｅで他のＰＥから持ってくる。これで並列プログラムができあがる。専門家にはうけないが、ユーザーにはこれが一番わかりやすい。

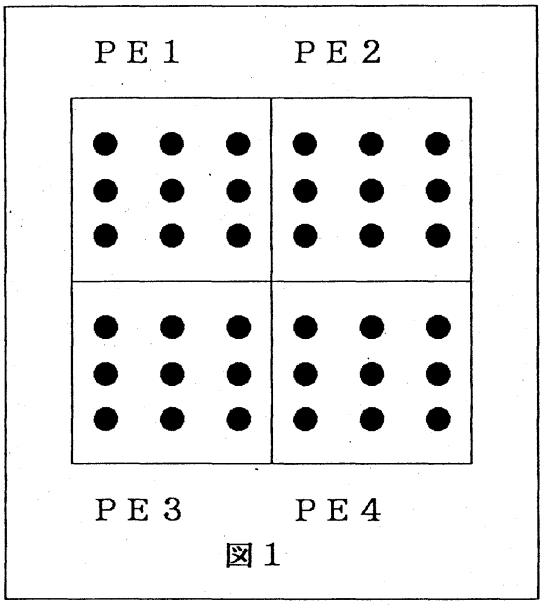


図１

３．ゲージ理論

ゲージ理論の特徴は、基本的な変数が点の上ではなく、点を結ぶリンクの上で定義されており、相互作用がリンクをつないでできるループで与えられていることである。図２において、a,b,c,d は２次元平面上の格子点を表し、$U(a,b)$等が２つの格子点を結ぶリンクの上で定義されたＳＵ（３）行列（３×３ユニタリー行列で行列式が１のもの）であるとき、相互作用は

$$\mathrm{Tr}\ U(a,b)\ U(b,c)\ U(d,c)^{+}\ U(a,d)^{+}$$

という形をとる。

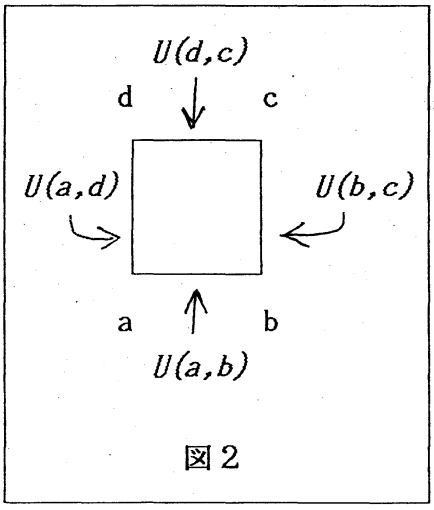


図２

我々の問題は空間３次元・時間１次元の４次元空間（x,y,z,t)を離散化したものである。計

算を進めている超並列計算機、富士通AP1000の要素は２次元配列なので、(+) ４次元空間を２次元面の上で分割しなければならない。一番簡単なのはｘとｙの部分を分割することである（図３）。尤もらしく「ｘｙパーティション」となずける。このときゲージ系の特徴のため、前後左右だけでなく、斜め方向のＰＥ間の通信も必要になる。(*) 現在の世界のトップレベルのスケールの問題

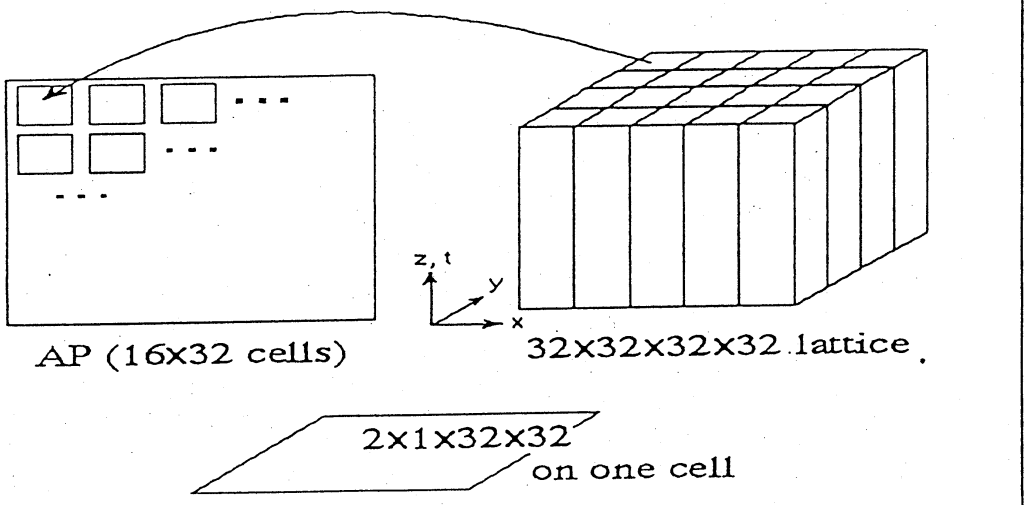


図３

は(x,y,z,t) 方向の点が３２×３２×３２×３２のオーダのものである。一方５１２個のＰＥの構成は１６×３２なので、さきほどの「ｘｙパーティション」の分割の方法では、１つのＰＥの上にのっているのは２×１×３２×３２となる。

４．繰り込み変換

繰り込み変換では、近傍の値を平均して新しく疎視化した系をつくっていく（図４）。つまり、３２×３２×３２×３２という系から、１６×１６×１６×１６を作り、更にそれから８×８×８×８という系を作りというように進んでいく。先程の「ｘｙパーティション」では、１６×１６×１６×１６という大きさの系を１６×３２個のＰＥの上に乗せることはできない。そこでｚ方向

(+) 現在Ｎ－ｊｉｇｅｎというライブラリーが用意され、ソフト的に任意次元の構成として扱うことができるようになった。

(*) ループにそって行列の積を計算するさい、まずコの字型の部分を計算してから転送を行なえば、斜め方向のＰＥ間の通信は必要がなくなる。広島大学・日置慎治氏のご指摘による。

も分割することにして「ｘｙｚパーティション」と呼ぶことにした（図５）。

まず、全格子をｚ軸に垂直な方向に８つにスライスする。一方５１２個の

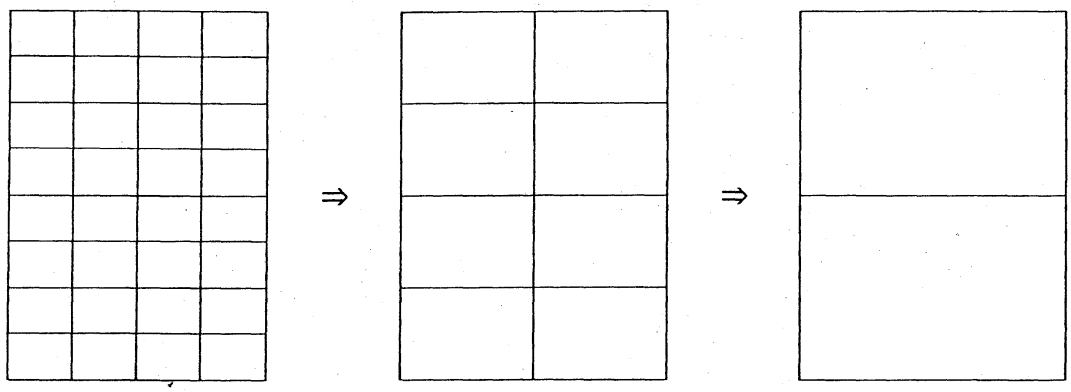


図４

ＰＥを６４個づつのクラスターに分割する。（プログラミングにおける仮想的な分割。ハード的には何も意味しない。）各クラスターをスーパーＰＥと呼ぶことにする。ｘｙｚパーティションでは、ＰＥ全体でみると複雑な通信が必要で、ｘｙパーティションの時のように、近傍のＰＥとの通信だけというようにはいかなくなる。

格子には周期境界条件が課してあるので、ｚ＝１とｚ＝z_{Max}（＝３２）が

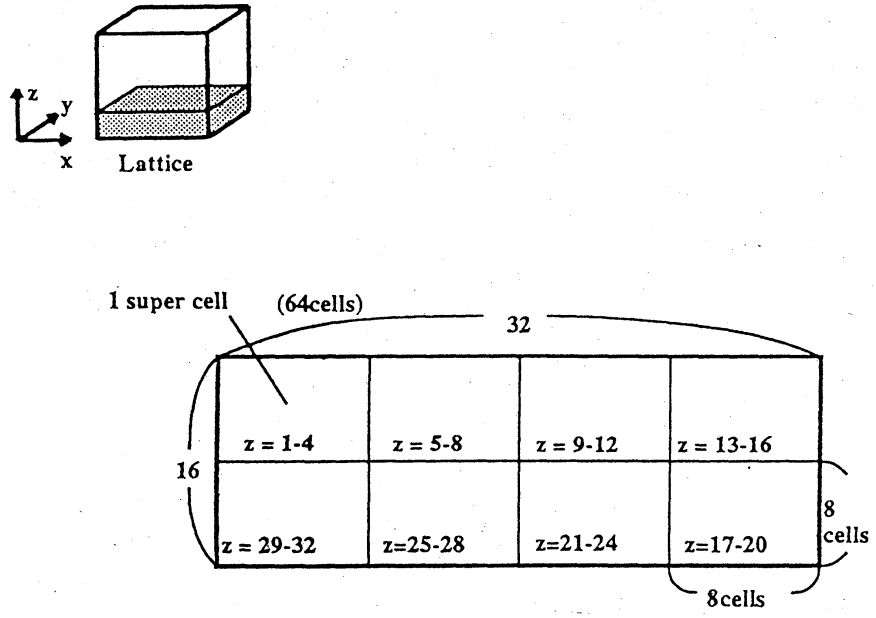


図５

少しでも近くなり、ＰＥ間の通信路の距離が小さくなるように、図５のようにスーパーＰＥを配置した。(*)

４．並列計算におけるＧＡＴＨＥＲ／ＳＣＡＴＴＥＲ

我々のようなタイプの計算における並列の問題というのはいったい何なのかを考えてみる。我々はシステムをＰＥに分割している。そのため、物理的な系で見たときある点の右隣の点が欲しいといっても、その点は同じＰＥの上にいるとは限らない。そこで通信の必要が起こる。言い換えると、系全体を表すグローバルな変数（Ｘ，Ｙ，Ｚ，Ｔ）を使ったとき、例えばｙ方向の隣の点は（Ｘ，Ｙ＋１，Ｚ，Ｔ）とすぐ分かるのだが、ＰＥの上のローカルな変数（ｘ，ｙ，ｚ，ｔ）で見たとき物理的にその点からみてある関係にある点の変数を持って来いと言われたとき、通信が起こる可能性がある。我々のタイプの計算では、通信は本質的にこの場合しか起こらない。変数をＰＥにどのように分割したか、ユーザーは知っているのだから、上のような場合だけを１つのルーティンにまとめておけばあとは並列計算のことは忘れてしまってもよい。問題は、１つのＰＥ上でローカルなインデックスｉを使って記述される

$$Y(i) = X(M(i))$$

というオペレーションが現れ、Ｍ（ｉ）という点が同じＰＥ上にない場合である。このとき、通信をしてよそからデータを持ってきてくれるルーティンを作り、ＸＧＡＴＨＥＲと名づけた（図６）。上の逆の機能を持ったルーティンをＸＳＣＡＴＴＥＲと呼んでいる。

$$Y(M(i)) = X(i)$$

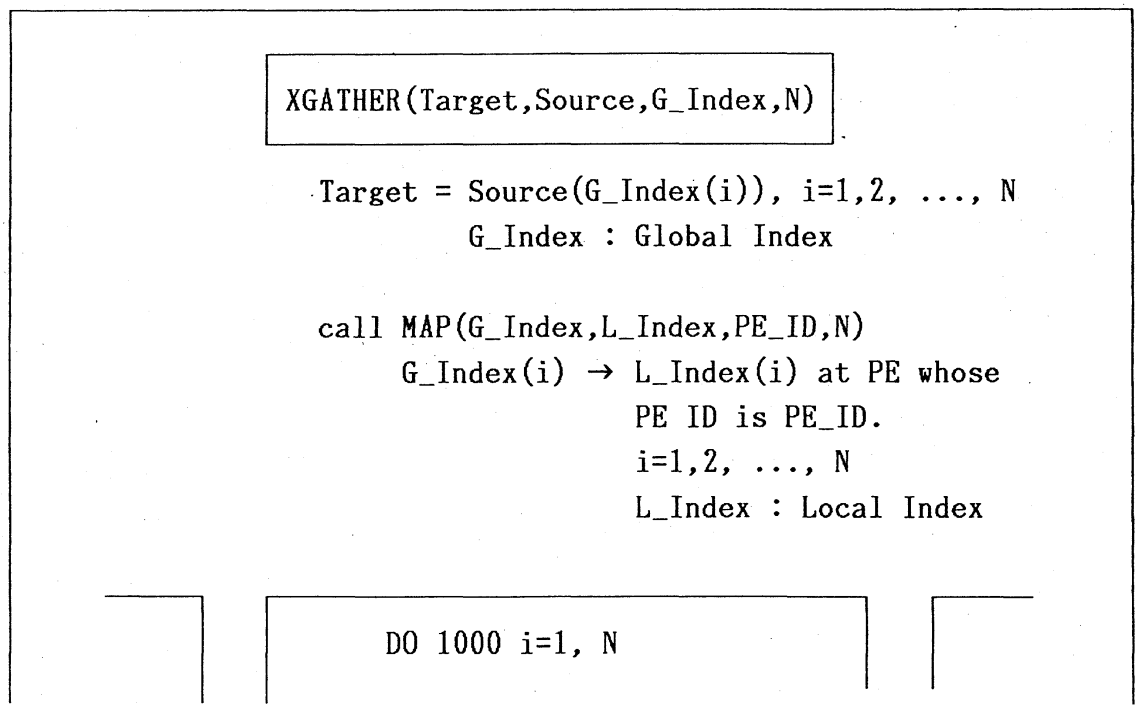
(*) 図５の配置は最善の解ではない。通信路の距離の総和を最短にするものは、下図のように配置した場合に実現される。ここで１は格子のｚ＝１－４が置かれるＰＥ、２はｚ＝５－８が置かれるＰＥ等々である。富士通計算科学研究センター柴田一哉氏、金澤宏幸氏のご指摘による。

```
１２３４　１２３４　１２３４　１２３４　１２３４　１２３４　・・・・
５６７８　５６７８　５６７８　５６７８　５６７８　５６７８　・・・・

１２３４　１２３４　１２３４　１２３４　１２３４　１２３４　・・・・
５６７８　５６７８　５６７８　５６７８　５６７８　５６７８　・・・・

・・・・
```

前節で述べたように、繰り込み変換の計算では、格子のＰＥへの配置は複雑なものになり、このＸＧＡＴＨＥＲ／ＸＳＣＡＴＴＥＲはたいへん強力なツールとなった。しかし以下のような問題点を持っている。

i) グローバルなインデックスは、表にすると巨大になり、各ＰＥに持たせることができなかった。そこで、配置のルールからそのたびに計算で求めている。このためＸＧＡＴＨＥＲ／ＸＳＣＡＴＴＥＲはたいへん時間を喰うルーティンとなってしまった。

ii) 図６に示したように、これらのルーティンは
① 必要なデータを持っているＰＥに、欲しいデータのローカル・インデックスを送る。
② 他のＰＥからの要求を受け取る。
③ 他のＰＥへ要求されたローカル・インデックスを持ったデータを送る。
④ 他のＰＥから送られてきたデータを受け取る。
他のＰＥからのメッセージを受け取るためには、Ｒｅｃｅｉｖｅを発行しなければならないので、②の段階でどれだけの回数の要求がくるか知っていなければならない。裏タスク等の技巧を使わずにプログラムを書いたため、他のＰＥに要求するデータの数は、自分が要求されるデータの数と等しいということを仮定してしまった。このため、完全に一般性をもったｇａｔｈｅｒ／ｓｃａｔｔｅｒルーティンにはなっていない。



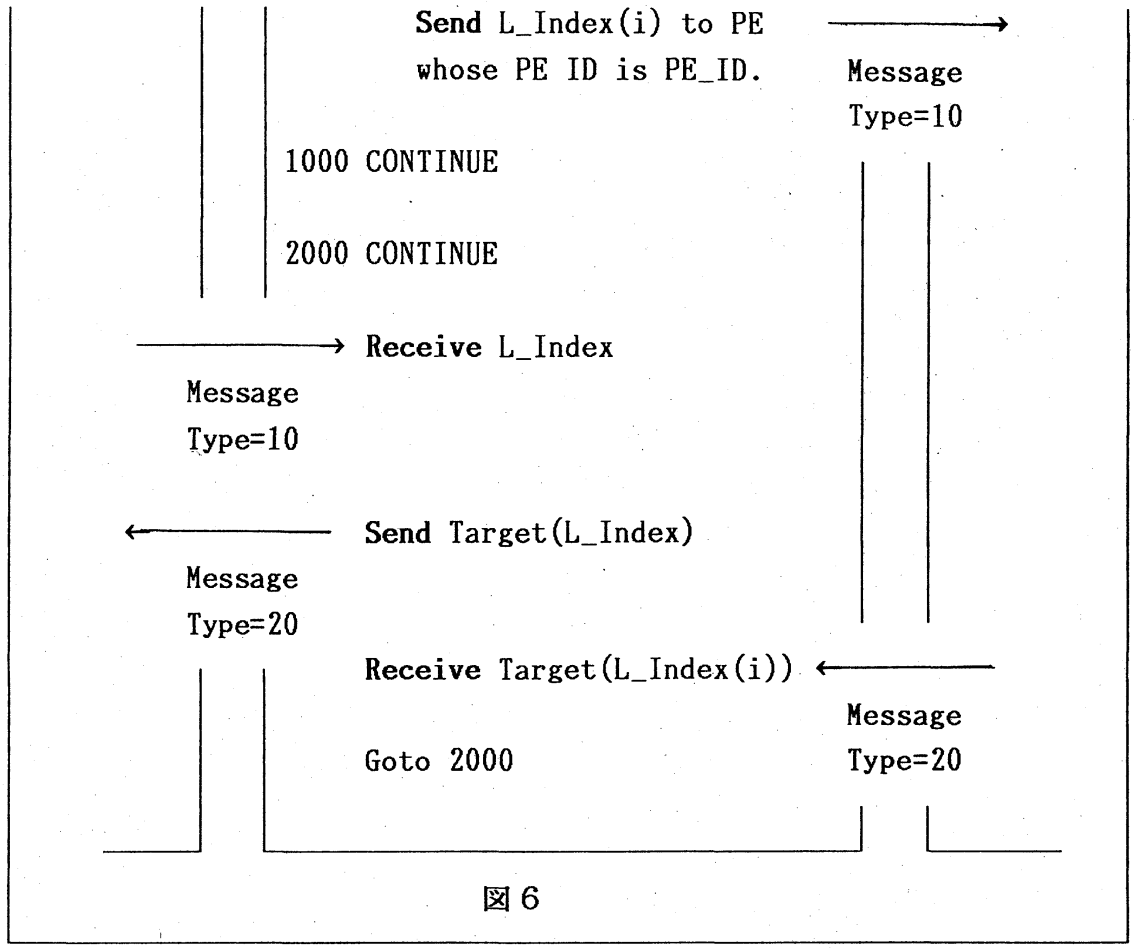


図６

５．結論

幾何学的な分割による並列化が可能であるような大規模問題では、超並列計算機は大変コスト・パーフォマンスのよいマシーンである。実用にあたってユーザーの観点から気が付いた点を、順不同であるが以下にあげてみる。。

１）並列プログラミングの経験が浅いためか、人間が本質的に並列に考えることが困難なのか、バグが発生したとき、発見がむずかしい。新しいタイプのプログラミング・スタイルあるいはデバッガーが必要なのかもしれない。（デバッガーで各ＰＥ上の状態を追うことができても、多数のＰＥの動きを把握するのは容易ではない。

２）並列計算機ごとに並列化のストラテジー、プログラムを変えなければいけないのはつらい。たとえば、ＶＰＰ型の並列プログラムとＡＰ１０００型の並列プログラムではまったく頭を切り替えなければならない。また、ベクトル演算装置がＰＥについているものとスカラーではストラテジーがま

ったく違う。

3）ＡＰ１０００のグローバル・サム（全ＰＥ上の和）を計算するライブラリーはたいへん有用であった。このように並列計算において非常に有用になるものを抽出していってライブラリー化していくことは、並列計算機の普及には大きな意味をもつであろう。

謝辞

本報告は富士通（株）・田子精男、奥田基、藤崎正英、川添明美、ハイデルベルグ大学・I.O.Stamatescu、スイス連邦工科大学・Ph.deForcrand 、ベルリン計算センター・H.C.Hege、広島大学・宮村修、日置慎治、高石哲弥、福井大学・橋本貴明の諸氏との共同研究に基づくものです。

参考文献

K.Akemi 他、 "QCD on Parallel Computer AP1000",
Nucl.Phys. B(Proc.Suppl.)26, 1992, 644-646.

K.Akemi 他、 "Quantum Choromodynamics Simulations on a Nondedicated Highly Parallel Computer",
プレプリント YAMAGATA-HEP-93-11
(Comp.Phys.Comm. に投稿中)

K.Akemi 他、 "Scaling Study of Pure Gauge Lattice QCD by Monte Carlo Renormalization Group Method",
Phys.Rev.Letters 71, 1993, 3063-3066.