計算機シミュレーションの並列処理

西原功修
（大阪大学レーザー核融合研究センター）
坂上仁志
（姫路工業大学工学部情報工学科）
新宮哲，川口正仁
（関西日本電気ソフトウェア株式会社）

概　要

本講演では、粒子シミュレーションコード、流体シミュレーションコードの並列処理について述べる。並列計算機としてはベクトル共有メモリー型、ベクトル分散メモリー型、スカラー分散メモリー型を考え、また、並列化手法としては領域分割、粒子分割について具体的なプログラムを例にとり、その検討結果、測定結果について報告する。ただし、本原稿では、高密度プラズマ粒子コード「SCOPE」の並列処理（領域分割）のみを記した。

1．はじめに

個々の粒子間力が重要となるクーロン多体系のダイナミックスシミュレーションを行う３次元強結合プラズマ粒子コード「SCOPE(Strongly COupled Plasma particlE code)」の並列処理について概説する。具体的にはスカラー超並列分散メモリ型のIntel社製Paragon XP/Sとベクトル高並列共有メモリ型のNEC社製SX-4による並列処理について述べる。この2台の並列計算機はスカラー⇔ベクトル、分散メモリ⇔共有メモリという異なったハードウェア構成となっている。Paragon XP / S については最大512台（メモリ：128MB×512PE=64GB）のプロセッシング・エレメント（PE）、SX-4については32台（メモリ：8GB）のプロセッサ（CPU）を用いて並列性能評価を行った。テストを行った計算機のカタログ最大演算性能は、Paragon XP / S ：75MFLOPS×512PE=38.4GFLOPS, SX-4：2GFLOPS×32CPU=38.4GFLOPSとなっている。

「SCOPE」は粒子コードであるが、次節で述べるように、計算の手法上ほとんどの計算は領域分割で計算を行っており、並列処理も領域分割を採用した。第3節では、SX-4、Paragon XP / S,それぞれについて並列化の手法を述べる。SX-4では、オリジナルのプログラムにほとんど手を加えることなくマクロタスク機能のみを用いて並列化を行うことを試みた。Paragon XP / Sについては、分散メモリ用に手を加え、比較的高い並列化を実現した。

並列処理性能は、プログラムの並列化の程度や並列化に伴うオーバーヘッドの割合に大きく依存するが、それぞれの計算機について、プロセッサの台数の増加とともにどれだけ計算処理時間が早くなるかを計測し、それらの依存性を調べた（第4節）。

2. 高精度3次元粒子コード「SCOPE」の概要

ここでは並列化に必要なコードの概要を説明する（詳細は参考文献［1］）。理想プラズマの粒子シミュレーションでは、個々の粒子の運動より集団的な振る舞いが重要となる。従って、近接粒子間力を抑制する工夫がなされ、有限の大きさの粒子と空間格子を使用するPIC(Particle-In-Cell)法［2］が開発されている。PIC法では、粒子の電荷を空間格子上に割り付け、ポワソン方程式を解いて空間格子上の電荷密度から空間格子上の電場を求め、その電場を粒子に割り付けることにより、他の粒子からの力を計算している。一方、強結合プラズマのように個々の粒子間力が重要となる問題については、各粒子について他の全ての粒子からの力を直接加算する分子動力学的手法が開発されている。しかしながら、後者の手法では計算量が粒子数の2乗に比例して増大するため、計算手法やハードウエアそのものに対する工夫がなされてきた。「SCOPE」ではHockey,Eastwood［3］により提案されたPPPM(Particle-Particle,Particle-Mesh)法を用いている。すなわち近接粒子間では、分子動力学のシミュレーションと同様に、個々の粒子間力を加算（PP法）し、十分に離れた粒子間では、PICシミュレーションで行う粒子 - 格子法（PM法）を用いる。この方法によって、計算量は粒子数に比例してしか増大せず、近接粒子間力をとり入れた大きな空間スケールのプラズマ・ダイナミックスのシミュレーションが可能となる。

PPPM法の計算精度は、PM法の導入に伴う近似計算の誤差によって決まる。このため「SCOPE」では、従来のPPPM法と異なり、個々の粒子間力を計算するPP領域の大きさを決める荒い空間格子（以下PP格子と呼ぶ）に加え、PM法で用いる細かい空間格子（以下PM格子と呼ぶ）を導入している。また、粒子電荷の空間格子への割り付けや、格子上の電場の粒子への割り付けには3次スプライン補間を用い、ポワソン方程式の計算には4次精度の空間差分を導入している。その結果、PM格子の1辺の長さが、PP格子の半分の長さ程度で十分な高精度計算を実現している（詳細は参考文献［1］を参照）。

具体的には、Fig.1に示すように（図示する困難さを避けるため、ここでは2次元格子を用いて説明する）、PP格子（実線）とPM格子（実線と破線）を導入し、図中のi粒子に働く他の粒子からの力は次の2つに分けて計算する。同一のPP格子にあるj_1粒子と、隣接する26個のPP格子内に含まれるj_2粒子からのクーロン力は、それぞれの粒子位置を用いて計算を行い加算する。このハッチ領域外のj_3粒子からの力は、その粒子電荷をPM格子上に割り付け、ポワソン方程式を解くことによって求めた空間格子点上の電場をi粒子に割り付けることにより求める。従って、i粒子に仂く力は

$$F_i = \sum_{j=j_1,j_2} \frac{e_i e_j}{\left|r_{ij}\right|^3} r_{ij} + e_i \sum_m E_m\left(x_i\right) \qquad (1)$$

と書ける。ここで、r_{ij}は、 i, j 粒子の位置をX_i, X_jとすると $r_{ij} = X_i - X_j$である。また、上式の第2項は j_3 粒子からの力をPM格子上で定義された電場E_mを用いて、$X = X_i$ での電場を3次のスプライン関数で補間したものである。

このように「SCOPE」では、粒子データ(i粒子の位置 $X_i(t)$ と速度 $V_i(t)$;)と、PM格子データ（m 格子上の空間電荷 ρ_m、ポテンシャル Φ_m、電場E_m）とがある。PPPM法での計算は、PIC法と同様、

（P-1）：PM格子上の電場E_mからi粒子への電場の割り付け、
（P-2）：i粒子の運動方程式の時間積分、
（P-3）：i粒子の位置からPM格子上の空間電荷 ρ_mへの割り付け

と、

（M）：PM格子上の空間電荷 ρ_mからポワソン方程式を解いてPM格子上の電場E_mを求める計算

に大別できる。前者の (P-1)～(P-3) の計算は粒子番号 i についての計算であり、後者の（M）の計算は格子番号mについての計算である。

テストに用いたプログラムはイオンの1成分強結合プラズマで、その主な入力パラメータは Table 1の通りである。ここでイオン結合定数は、

$$\Gamma = \frac{e^2}{akT} \quad , \quad \frac{4}{3}\pi a^3 = \frac{1}{n}$$

で定義され、また、e，n，Tはそれぞれ電荷、イオン数密度、温度であり、aはイオン球半径である。1成分強結合プラズマはイオン結合定数でのみ特性づけられる。

3 PPPM法の並列化

各粒子に対する計算、即ち上記 (P-1)～(P-3) の計算は、すべて粒子背番号についてのDOループ計算となっている。PPPM法では、通常のPIC法と異なり、各PP格子ごとにそのPP格子に含まれる粒子に対して（1）式の力F_i を求め、運動方程式の時間積分を行う。並列化を行う以前の「SCOPE」においても、粒子の背番号は、系全体での背番号だけでなく、各PP格子ごとに背番号が与えらており、最内殻DOループは各PP格子内での粒子番号となっている。したがって、並列計算は、基本的にはシミュレーション実空間をPP格子に領域分割し、各PP格子での粒子に対する計算（P）を並列計算機の各プロセッサで行う方法が最も簡単である。この並列化手法を採用することによって、例えば共有メモリ型のSX-4 では数行のマクロタスク組み込みサブルーチンのCALL文を挿入するだけで並列化が可能であった。

一方、計算（M）の部分、すなわち、PM格子上で定義された空間電荷から空間格子上の電場を求める計算は、PM格子に関する計算であって、並列化には異なっ

た手法を用いなければならない。しかしながら、テストプログラムでは ポワソン方程式（１３点差分方程式）の解法に高速フーリェ変換（FFT)を採用しているが、以下に述べるように、この部分のコストは低く、テスト段階ではこの部分の並列化は行っていない。並列処理部分は、プロセッサ台数の増加に比例して高速に処理されるが、このような非並列化部分があると、この部分の計算に要する時間のため、全体の処理はプロセッサ台数に比例した高い効率が得られない。非並列化部分が多い程、またプロセッサ台数が多い程この傾向は顕著となる。しかし、「SCPOE」を1台のプロセッサで実行した場合の非並列計算部分の計算時間が全体の計算時間に占める割合は、Fig.2に示すように、SX-4、およびPragon XP / S で、それぞれ0.30％と0.02％にすぎない。したがって、粒子に対する計算部分のみを並列化しても十分に高い効率が期待できる。非並列化部分の割合が、SX-4とPragon XP / S で、10倍以上も異なるのは、前節の (P-3) の部分、すなわち、粒子電荷をPM格子上空間電荷に割り付ける部分(通常PIC法ではこの部分の計算量が非常に多い)を、SX-4では並列化しなかったことに起因する。なお、オリジナルプログラムでは、(P-3) の部分は、系全体での粒子背番号を用いた計算になっており、SX-4でこの部分を並列化しなかったのは、後述するように、オリジナルのプログラムに手を加えることなくマクロタスク機能のみを用いて並列化することを試みたかったためである。

3.1　共有メモリ型の並列化

まず、並列化が容易である共有メモリ型について述べる。ＰＰ格子で領域分割し、(P-1),(P-2) の計算を行うサブルーチン群をタスクの単位としてマクロタスク並列処理を行う。この手法の特徴は、並列化を行ってもベクトル長が短くならずにSX-4のベクトル性能を生かすことができることである。各タスクが処理するＰＰ格子は、あらかじめテーブルを作成して割り振る。タスク生成のオーバーヘッドを出来るだけ少なくするため、子タスクの生成（PTFORK）は１度だけ行う(Fig.3参照)。Fig.3の主な記号は以下の処理に対応する。

INT: 初期データの読み込みなどの初期処理
G to L: 系全体の粒子背番号で定義された粒子データを各ＰＰ格子の粒子番号に対応するデータに移項する。
MOTION: 粒子の運動方程式の時間積分（（1）式, (P-1),(P-2) を含む）
CHGASN: 粒子位置からPM格子上の空間電荷を求める（P-3）
POISSON: ポワソン方程式を解き PM格子上の電場を求める（M）

Fig.3の Ｇ to Ｌ とMOTION の計算は並列処理であり、プロセッサの台数がPP格子の数より少ない場合には、各プロセッサはこの部分を繰り返し計算することになる。

PM格子上の空間電荷を求めるCHGASNの計算を行うには、全ての粒子位置が更新されていなければならない。したがって、CHGASNの計算の前に同期制御が必要であり、この 同期制御にはバリア同期制御（PBSYNC）を用いる。また、親タスクで POISSON の計算を行い PM 格子上の電場が求まるまで子タスクの計算を

抑制しなければならない。このため POISSON の後で再び同期制御を行う。なお、共有メモリ型では、全タスクで共有するグローバルコモン変数と各タスクに固有なローカルコモン変数とに振り分ける必要がある。このコモン変数の変更以外は、オリジナルのプログラムにほとんど手を加えることなく、（PTFORK）,(PTJOIN),（PBSYNC）のマクロタスク組み込みサブルーチンを挿入するだけで並列化が可能であった。

3.2　分散メモリ型の並列化

分散メモリ型では、全てのPEに同じプログラムをロードする（Fig.4参照）。各PEは担当するPP格子内の粒子の運動を必要に応じて同期をとりながら並列に計算処理を行う。オリジナルのプログラムでは、粒子の背番号は各PP格子内だけでなく系全体の背番号が与えられており、SX-4の並列化で述べたように、系全体の背番号で定義された粒子データをPP格子内の粒子番号に対応するデータへ移項する処理（G to L）を行っていた。分散メモリ型では各PEのメモリ（128MB）が大きくないことと、このデータ移項を行った場合には、そのデータ送信量が膨大な量となるので、各PEは担当するPP格子に必要なデータのみを持つように変更した。一般には、PP格子の数よりPEの台数の方が少いと考えられるから、各PE は複数個のPP格子の計算を行うことになる。したがって、各PE の分散メモリには

(1) 分担する複数個のPP格子内に含まれる粒子の位置と速度、
(2) 対応するPP格子に隣接する26個のPP格子内の粒子の位置
(3) 対応するPP格子に含まれるPM格子上の電場と空間電荷密度

が含まれる。この変更に伴い、オリジナルプログラムのG to Lを、

PP to PE：各PE が担当する全てのPP格子の粒子データから計算するPP格子のデータに移項する。

に変更した。また、この変更に伴い、粒子位置からPM格子上の空間電荷を求める計算（CHGASN）も、各PEで担当するPP格子に対応するPM格子に対してのみ並列計算を行うように変更した。その結果、(P-3) の計算も並列化することができた。

PPPM法では、先に述べたように、（P-1）～（P-3）の各粒子の対する計算だけでなく、（M）の計算、すなわち、PM格子上の空間電荷からPM格子上の電場を求める計算が必要である。このポワソン方程式の解法にFFTを用いているので、その計算には全PM格子上の空間電荷が必要となる。したがって、分散メモリーの場合には、共有メモリーの場合と異なり、各 PE が計算した空間電荷をお互いに送受信する必要がある。一つの PE でポワソン方程式を解く場合には、求ったPM格子上の電場を再び各PEに送信する必要が生じる。この2回の送信を1回にするため、本テストプログラムでは、各PEが計算した空間電荷を全てのPEに送信（broadcast）し（Fig.4のcsend（3））、全てのPEが同じポワソン方程式を解くようにした。

その他の PE 間のデータ送受信について説明する（Fig.4参照）。データ送受信や同期制御にはPragon 固有の nxライブラリを用いた。あるPP格子に含まれる粒子の運動を求めるには、そのPP格子に隣接する26個のPP格子内の粒子位置データを必要とするが、そのデータを送受信するのが、Fig.4の（csend（1））である。

実際には（csend）コマンドだけでなく（crecv）コマンドで同期を取りながらこの送受信を行った。

各時間ステップごとに運動方程式を積分して、各ＰＰ格子の境界を横切る粒子については、その粒子の背番号、位置、速度の情報を対応するPEに送信しなければならない。このデータの送受信は、各ＰＰ格子内の全ての粒子について運動方程式を時間積分した後に、対応するPEごとに送信する粒子数の情報を含めて行う（csend（2））。時間きざみ（△ｔ）は十分小さく、境界を横切る粒子は隣接ＰＰ格子にしか移動しない。しかしながら、Paragon のPEは２次元メッシュ結合であるから、隣接する26個のＰＰ格子を担当するPEは隣接していないものが必ず含まれる。したがって、これらのデータ転送も複数のPEを経て行われることになる。また、上記の粒子移動データの受信は、各PEの一時データ配列に格納されるため、

APPEND：一時データ配列を各PP格子データに移項する

を設けた。このように分散メモリ型では3回のデータ送受信が発生し、そのたびに同期制御を伴うことになる。

4　並列処理性能

Table 1に示すように、10ステップの計算を行い、SX-4については最大32台のCPUを用い、またParagon XP/Sについては最大512台のPE を用いて、プロセッサ台数の増加とともにどれだけ計算処理時間が早くなるか、その加速率を計測した。なお、512台のPEを開いた場合には、1台のPEが１つの格子を分担することになる（Table 1）。実際のシミュレーションでは、粒子数や格子数がもっと大きく、また数万ステップの計算を行うので、加速率の評価には以下の処理時間を除いた。SX-4については子タスクの生成に要する時間、Paragonについては各PEにプログラムをロードする時間、さらに粒子の初期データの読み込み時間や、計算結果が正しいかどうかをチェックするための計算とその出力に要する時間を除いた。例えば、SX-4で32台のCPUを用いた場合については、評価対象経過時間は3.74秒であり、上記処理を含めた全経過時間は4.74秒である。またParagon で 512台のPE を用いた場合については、それぞれ18.86秒と、197.50秒である。SX - 4に用いたプログラムの方が並列化の程度が低いにもかかわらず、「SCOPE」のテストでは、評価対象経過時間で、SX - 4が５倍速い結果となった。なお、SX - 4での上記処理を含めた全ての計算での演算性能は11GFLOPSである。また、SX-4のプログラムのベクトル化率は99.2%で、平均ベクトル長は115である。

SX-4とParagon XP/Sについて、加速率とその評価対象経過時間をFig.2-5、とFig.6に示す。SX-4の場合は、最大16CPUまで、またParagon XP/Sでは128PEまで、ほぼプロセッサ台数に比例したスケーラブルな加速率が得られている。それぞれの計算機での最大プロセッサ台数においても、SX-4では32CPUで24.9倍、Paragon XP／Sでは512PE で359.1倍の高い加速率が得られた。SX-4では加速率の増加が32CPUですでに線形な伸びから低下しているのは (P-3)の計算、すなわち、PM格子上の空間電荷計算（CHGASN）を並列化しなかったことも原因と考えられるが詳しくは後述する。

並列プログラムの実行時間の加速率は、プログラムの並列化の程度(並列化率α)と、並列化に伴うオーバーヘッドの割合(β)の特性に大きく依存する。実効的な加速率nは、プロセッサ台数をKとすると、

$$n = \frac{1}{1-\alpha+\frac{\alpha}{K}+\beta} \quad (2)$$

となる。(1-α)は非並列計算部分の処理、α/Kは並列計算部分の処理の割合である。Fig.7に、並列計算部分の計算時間、非並列計算部分の計算時間、同期制御の待ち合わせ時間、データ送受信時間の割合を示す。なお、これらの時間の割合いはプログラム中の測定場所や測定方法に依存し、大体の目安として考える方がよい。

まず並列処理部分と非並列(逐次)処理部分の計算時間の占める割合が、1台のプロセッサの場合に比べて、それぞれのプロセッサ台数でどのように変化するかを考察する。並列化率αを、Fig.2に示すように並列処理部分が一台のPEでの経過時間の中に占める割合で定義すると、並列計算部分の処理時間と非並列計算部分の処理時間との比は、理想的には

$$\frac{\alpha/K}{1-\alpha} \quad (3)$$

となる。SX-4、Paragon XP/S に対して、それぞれα=0.99705、および、α=0.99977を(3)式に使うと、比は、それぞれ、10.56、および、8.49となる。一方、比の実測値は11.40、および、8.75である。これは各プロセッサが処理する粒子数のバラつきのためと考えられる。バラつきがあると、並列処理を行っても処理時間は最大の粒子数を処理するプロセッサの処理時間によって決まる。テストデータでは、平均粒子数が45.625に対してPP格子の粒子数の最大値と最小値は、55と40程度ものバラつきがあった。なお、並列計算部分の処理時間と非並列計算部分の処理時間との比について、理想的な場合とのずれをParagonXP/S とSX-4 について比較すると、ParagonXP/S については8.75/8.49=1.03、SX-4 については11.40/10.56=1.08となっている。Paragon XP/Sに比して、SX-4 が32台のCPUの割りに並列処理時間が長い。この原因を同定することは困難であるが、次のことが原因として考えられる。Paragon では、並列処理時間の測定をnode(0)でのみ行ったためである。このときには、後述するように、粒子数のバラつきによる並列計算部分の処理時間の増加の一部が送受信に含まれてしまう。今1つは、テストに用いた各計算機の最大構成がParagon XP/Sが800PE, SX-4は32 CPUであり、SX-4ではテストCPU台数が最大CPU構成に等しく、OS の負荷等が関係していたかもしれない。

並列化に伴うオーバーヘッドが無い(β=0)、理想的な場合の加速率を、先に定義したαを用いて(2)式より求めると、SX-4 / 32 CPUでは29.3倍、Paragon

XP / S / 512 PEでは、458.2倍の加速率が得られることになる。しかし、実際には並列化に伴うオーバーヘッドがプロセッサ台数の増加とともに無視できなくなり、Fig.5,6に示した様にプロセッサ台数が多くなると加速率の伸びが減少してくる。この並列化に伴うオーバーヘッドを評価するために、SX-4とParagonについて同期待ち合わせ、データ送受信に要する時間を検討する。共有メモリ型のSX-4ではデータの送受信を必要としないが、3.1項で述べたように同期制御が必要である。その同期待ち合わせ時間が32CPU では経過時間の2.8％となる。この同期制御に要する時間は各プロセッサが処理する粒子数のバラつきに関係する。しかし、先に述べた粒子数の大きなバラつきの割りには、同期制御時間が2.8％と比較的小さい。この原因には、色々な理由（例えば、27個のPP格子内の粒子数のバラつきは比較的少ないなど）があるが、SX-4ではベクトル計算を行うので、粒子数の増加に比べて計算時間はあまり増加しないことが考えられる。一方、Paragonでは同期制御の占める割合が1.8％と低く評価されている。これは、同期制御を行う gsync (1) の前に実行するデータ送受信（csend (2)）に同期型の受信（crecv）を用いていることにより、粒子数のバラつきにより発生する必要な同期制御がこの crecv 実行時に取られているためである。したがって、この1.8％の同期制御負荷は低く見積もられていることになる。Paragon XP / S / 512 PEでは、データ送受信の負荷の増加が著しい。Paragon XP / S では3回のデータ送受信を行っているが、それらのデータ量と処理時間との関係は興味深い。Fig.4のcsend (1) で各プロセッサが関係する26個のPEに送信するデータ量は、3.3×26KB程度であり、また、csend (3) で各PEが全てのPE に送信するデータ量は1×511 KB 程度である。csend (2) のデータ量は各PP格子を横切る粒子数に比例するのでその量は不明であるが、cend (1) に比べてそのデータ量は非常に少ない。一方、（csend （1）, (2), (3)）の全処理に占める割合は、それぞれ、2.7％、7.2％、13.1％となっている。（csend （1））に比べて（csend （3））のデータ量は非常に多いが、処理時間が比較的短いのは全PEに送信するbroadcastが非常に高速に処理できているからである。なお、（csend （2））の送信データ量は（csend （1））に比べ少ないはずであるが、この負荷が高いのは、すでに述べたように、各PP格子の粒子のバラつきによる同期制御が含まれているためである。（2）式のオーバーヘッドの割合（β）を厳密に定義することは困難であるが、1台のプロセッサで計算した場合の全経過時間に対する並列処理での送受信、同期制御に要する時間と考えることができる。そうするとSX-4 / 32CPUではβ＝0.11％、ParagonXP / S / 512PEではβ＝0.069％にしかすぎない。この小さなオーバーヘッドでもプロセッサ台数の増加とともに並列処理部分が高速に処理されるため、Fig.2-7に示すように、512台ものPEを使用した場合には、データ送受信、同期制御に要する時間は全経過時間の32.5％を占めることになる。

最後に、Paragon XP/S 用のプログラムで、（P-3）の計算（CHGSAN）を並列化しなかった古いバージョンでの結果を比較の意味で少し述べる。加速率はFig.6の三角で示すように、215であった。また、1台のPEでの非並列計算部分は0.16％であり、またこのバージョンでの（csend （1））は、データ量が約3.3KBであり、

このデータを全PEに送信（broadcast）するので3.3×511KBのデータが送られる。512PEでの並列計算、非並列計算、同期制御、送受信は、それぞれ、38.11%、34.46%、0.79%、26.65%であった。このように、非並列計算の部分と送受信データのわずかな増加により、並列処理部分の占める割合が約半分に減少し、512台ものMPP（Massively Parallel Processing）では並列性能が低下する。

5 まとめ

PPPM法を用いた3次元粒子コード「SCOPE」について計算アルゴリズムの特性を生かした領域分割並列処理手法を開発した。ベクトル高並列（32 CPU）共有メモリ型のSX-4とスカラー超並列（512PE）分散メモリ型のParagon XP/S のいずれの場合にもスケーラブルな並列性能を得ることが出来た。SX-4 についてはオリジナルプログラムに数行のマクロタスクライブラリを挿入するだけで、高い並列加速率を実現した。また、並列化によってベクトル長が短くならずにベクトル性能を生かすことを可能とした。Paragon XP/Sについては、分散メモリであるために発生する通信量を減らすこと、また並列計算部分を増加させること等のプログラム構造の改善を行い、nxライブラリを用いて高い並列加速率を512台のPEまで実現できることを実証した。

今後の予定として、ポワソン方程式の計算並列化、PP格子内粒子数のバラつきによる同期制御待ち合わせ時間の抑制、また、並列処理の標準となりつつあるMPI（Message Passing Interface）ライブラリの使用などを考えている。

参考文献

[1] K.Nishihara: Kakuyugo Kenkyu **66**, 253(1991).

[2] C.K.Birdsall and A, B. Langdon: *Plasma Physics via Computer Simulation* (McGraw-Hill, New York, 1985).

[3] R.W.Hockney and J.W.Eastwood: *Computer Simulation using Particles* (Adam Hilger, New York, 1988)

（西原 功修）

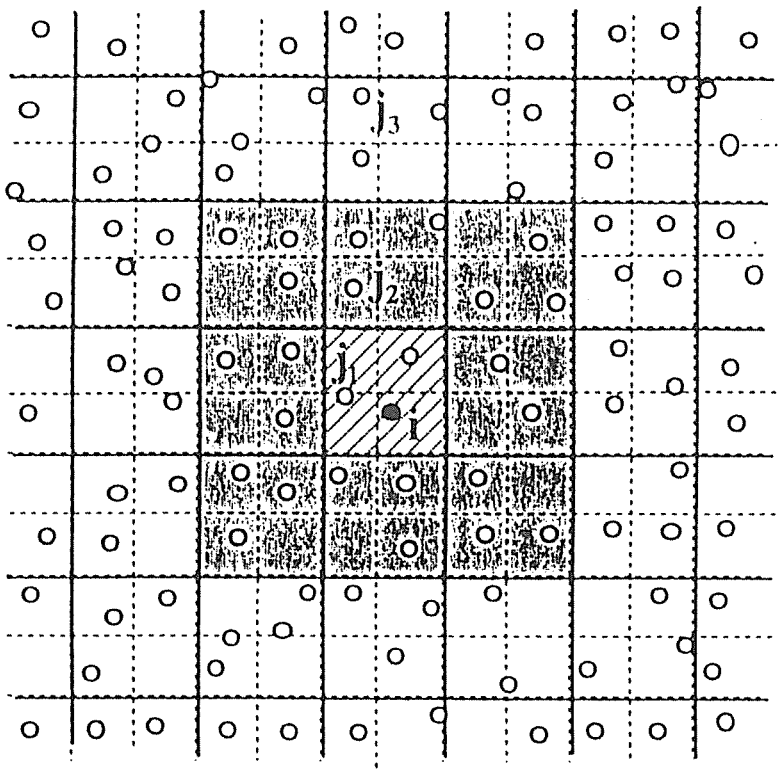

Fig. 1 Prticle-Particle meshes (solid line) and Prticle-Mesh meshes (solid and dotted lines) in SCOPE.

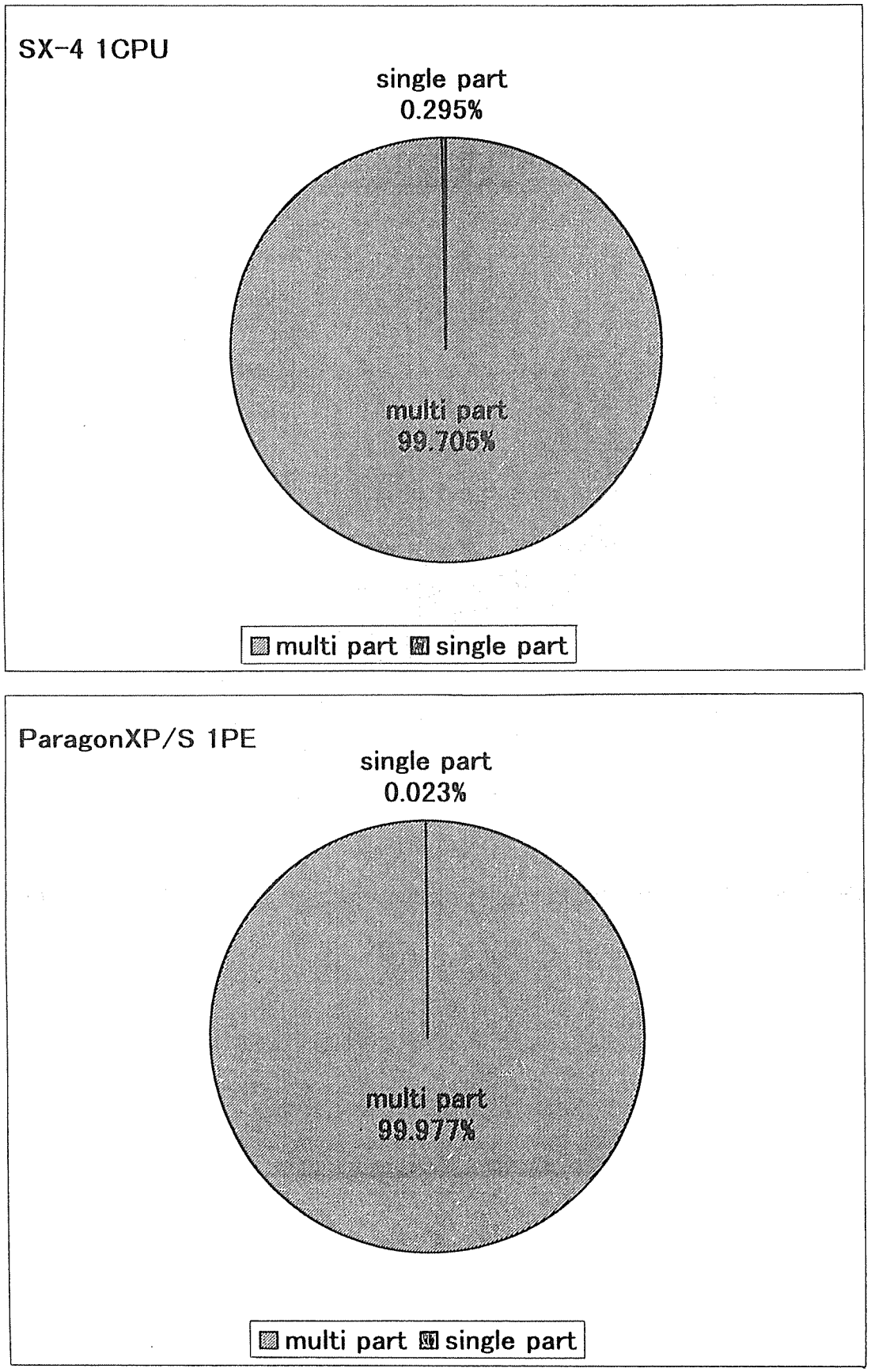

Fig. 2 Cost distribution in serial processing for SX-4 (a) and Paragon XP/S (b).

Table 1. physical and computing parameters

ion coupling constant Γ	10
total partile number N	23, 360
PP latice number	8×8×8
PM latice number	16×16×16
maximum processing element number K	32(SX - 4) / 512(paragon)
time steps	10

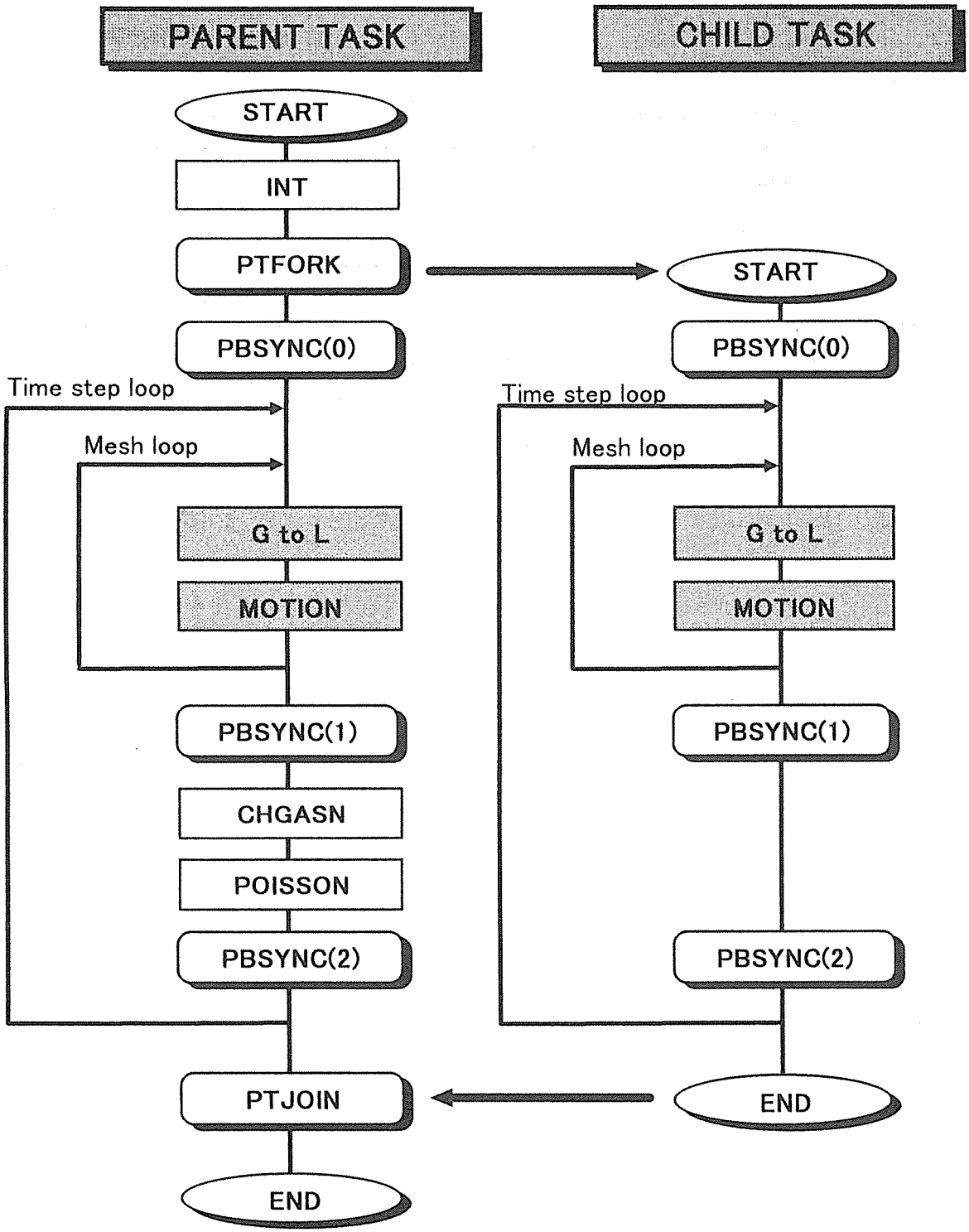

Fig. 3 Flow chart of parallelized SCOPE using macro tasking in SX-4.

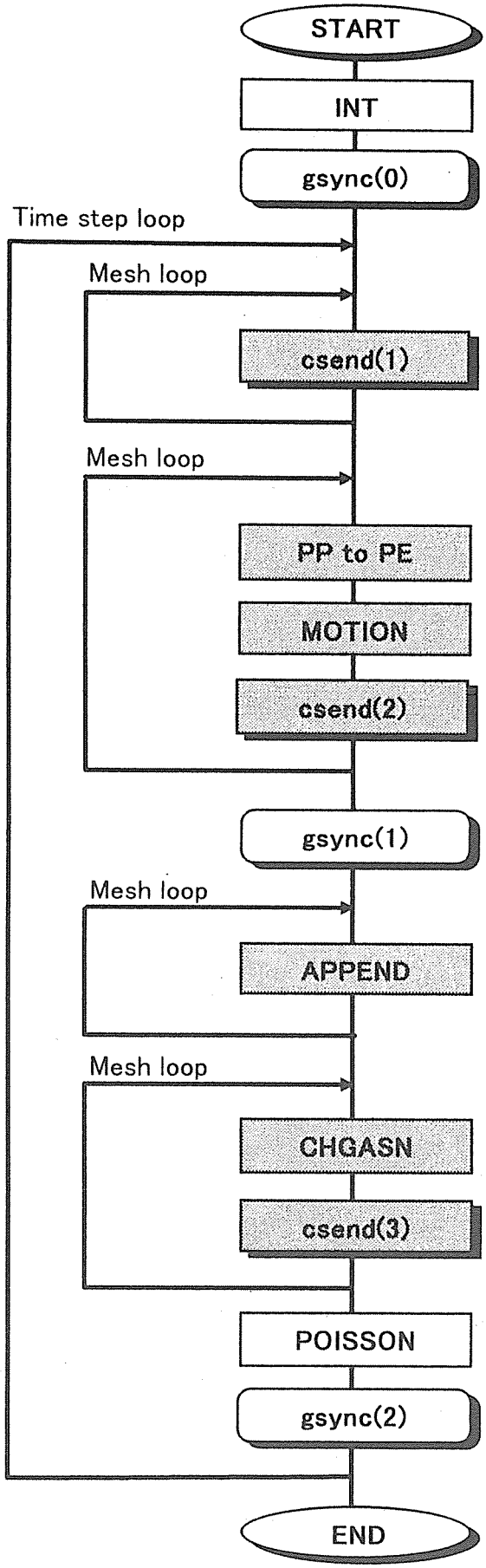

Fig. 4 Flow chart of paralleized SCOPE using nx library in Paragon XP/S.

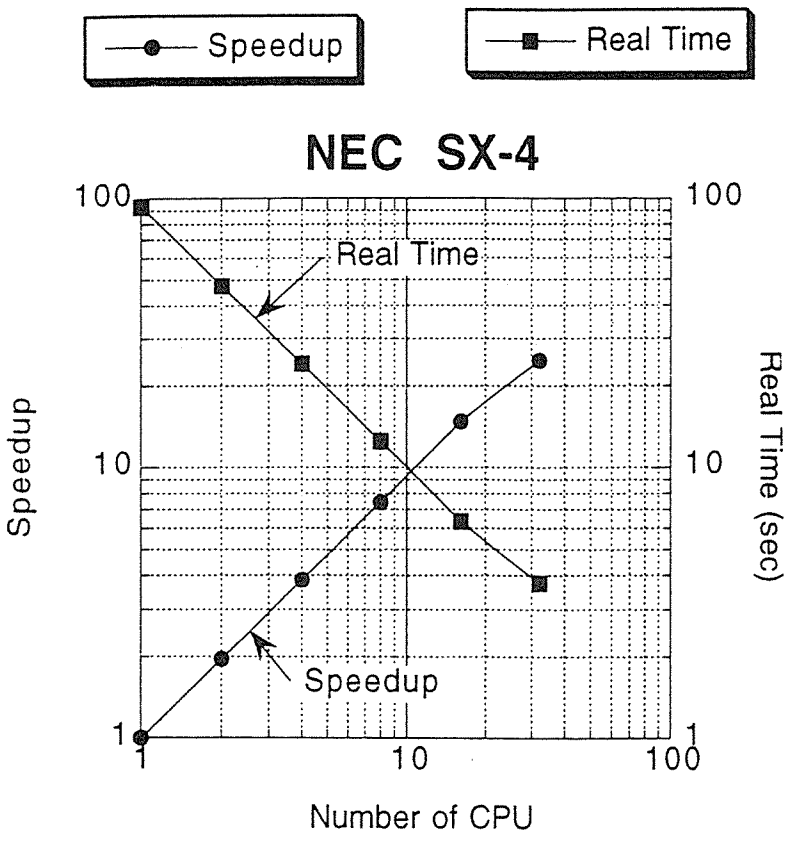

Fig. 5 Speed up ratio and real time as a function of number of processing elements in SX-4.

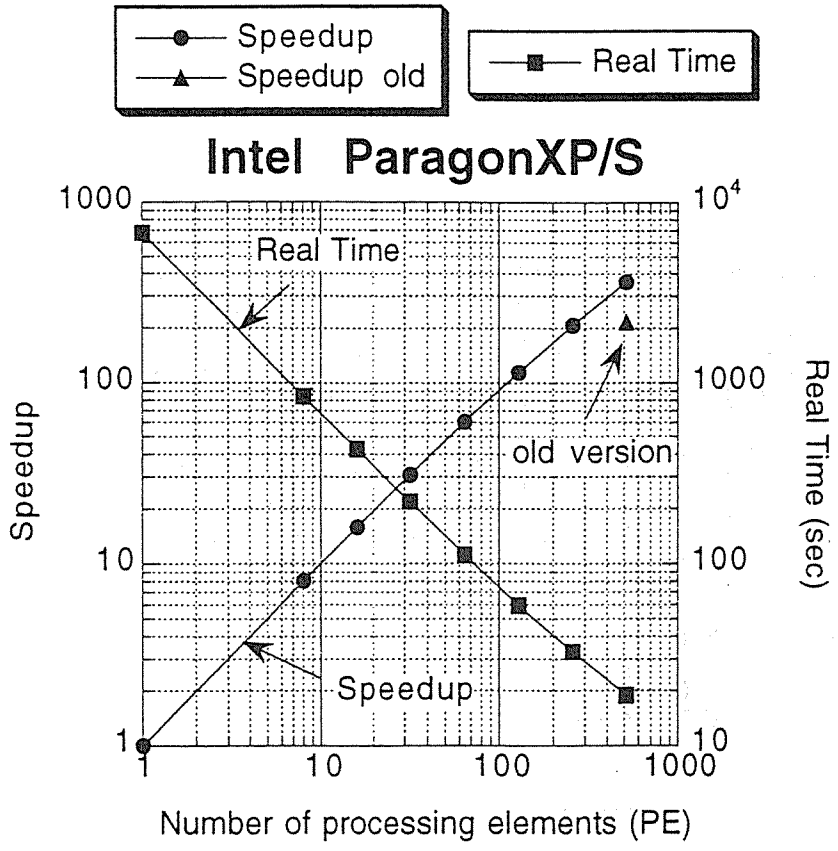

Fig. 6 Speed up ratio and real time as a function of number of processing elements in Paragon XP/S

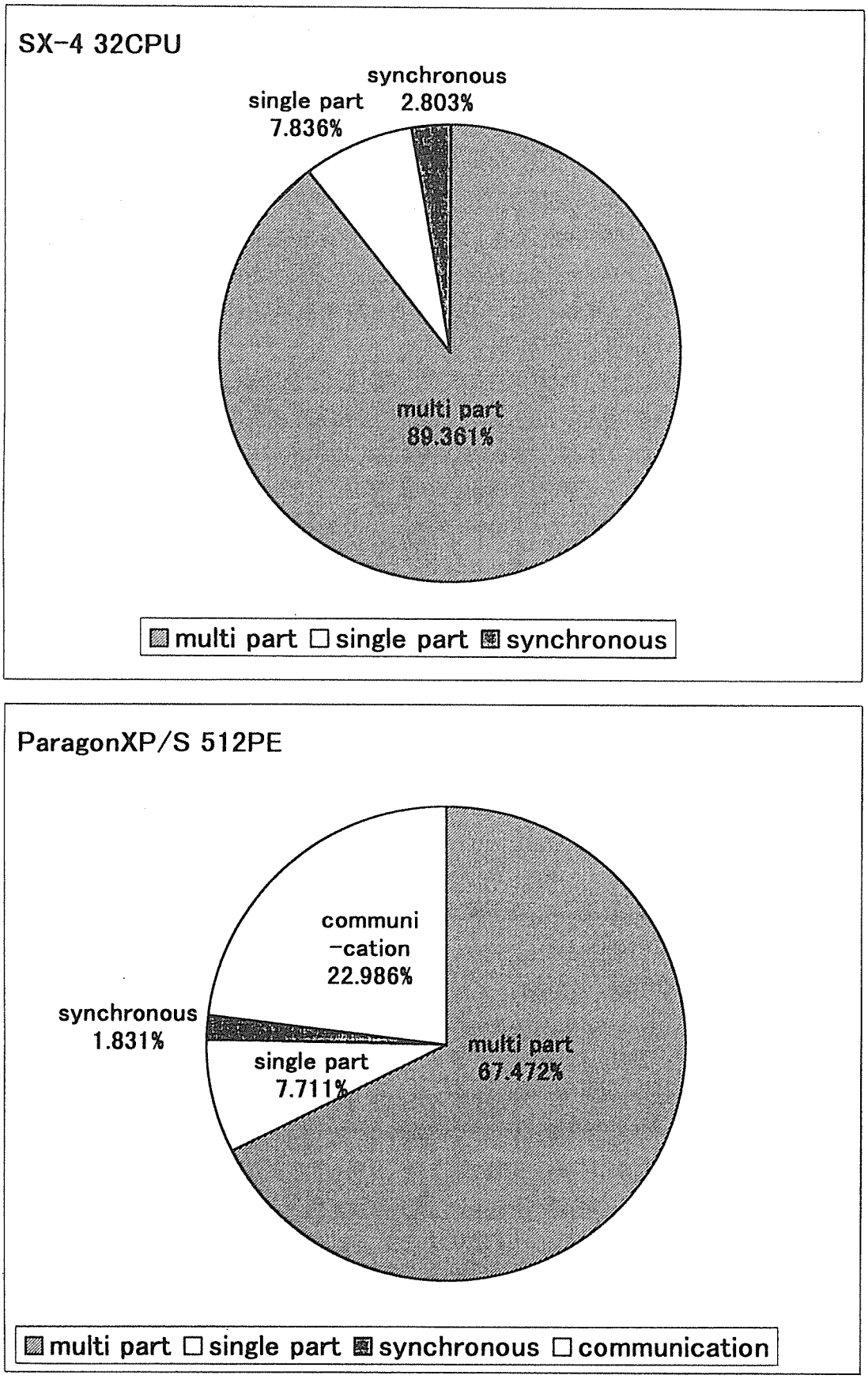

Fig. 7 Cost distribution in parallel processing for SX-4 32PE (a) and Paragon XP/S 512PE (b)