ハイパフォーマンス
コンピューティング 58－5
（１９９５．１０．１８）

並列処理数値シミュレーションのための スケーラビリティ検討方法

折居茂夫
日本原子力研究所　計算科学技術推進センター
〒113　東京都文京区本駒込2-28-8　理化学研究所駒込分所
E-mail : orii@koma.jaeri.go.jp

科学技術分野の数値シミュレーションを並列処理で行う場合、シミュレーションの問題の性質、規模、計算スキーム、アルゴリズム、プログラミング方法が、スケーラビリティに影響を与える。並列計算がシミュレーションプログラム中の並列性を利用するためである。更に、並列計算機のアーキテクチャ、性能仕様、言語等が並列計算機特有のオーバーヘッドを生み、並列性と密接に関係し、スケーラビリティの検討を困難にしている。本研究の目標は、並列処理時間のモデルを作り、数値シミュレーションコードのスケーラビリティ検討方法を確立することである。時間測定と共にこのモデルを使う時、並列性とオーバーヘッドを精度良く表すことができる。本論文では、この方法を示し、並列化コレスキープログラムを用いてVPP500, Paragonでのスケーラビリティ検討を行った。

A Method of Analyzing Scalability for Parallel Processing Numerical Simulation

Shigeo Orii
Japan Atomic Energy Research Institute
Center for Promotion of Computation Sci. & Eng.
2-28-8, Honkomagome, Bunkyou-ku, Tokyo 113, Japan

In case of parallel processing for a numerical simulation in the field of science and technology , there are some factors contributing to scalability, such as a property and size of the problem, calculation scheme, algorithms, and programming style of the simulation code. The reason is that parallel calculations make use of parallelisms in the simulation code. In addition, the architecture, the performance and the language of parallel computers spawn specific overheads which relate closely to the parallelisms and result in the cause of difficulty in analyzing scalability. The target of this study is to establish the method of analyzing scalability of numerical simulation codes by making a model of parallel processing time. The model, combined with measurement of the time, accurately describes the parallelisms and the overhead. In this paper, the modeling methodology and two examples of scalability analyses are shown with a parallelized Cholesky program on VPP500 and Paragon.

1．はじめに

並列処理は、問題やｼﾐｭﾚｰｼｮﾝｺｰﾄﾞの中にある並列性を利用して性能向上を図る。しかし並列処理の特有のｵｰﾊﾞｰﾍｯﾄﾞとのﾊﾞﾗﾝｽの上に全体の性能が決まる。

このように相殺する効果がある場合、ｱﾙｺﾞﾘｽﾞﾑの処理時間をﾓﾃﾞﾙ化することによって現象の理解が容易になり、最適化のために役立つことが、ﾍﾞｸﾄﾙ計算機時代から知られている[1]。並列計算機においても、処理時間の理想化したﾓﾃﾞﾙ化を行って並列計算機の特性を研究すること[2]、数値計算ｱﾙｺﾞﾘｽﾞﾑとして意味のあるｶｰﾈﾙをﾓﾃﾞﾙ化して性能評価を行うこと[3]が行われてきた。

これらの研究にも関わらず、実際のｼﾐｭﾚｰｼｮﾝｺｰﾄﾞのｽｹｰﾗﾋﾞﾘﾃｨの検討では、未だ並列化作業を行って時間測定するのが現状である。この主な原因の一つは、ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞのような大きなｺｰﾄﾞの処理時間のﾓﾃﾞﾙ化が困難なためである。

しかし、数値ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞのｽｹｰﾗﾋﾞﾘﾃｨを、系統的、定量的に検討するためには、この問題を避けて通ることはできない。

そこで本論文では、３章でｼﾐｭﾚｰｼｮﾝｺｰﾄﾞの処理時間のﾓﾃﾞﾙを作り、任意の並列計算機のｽｹｰﾗﾋﾞﾘﾃｨを検討する方法について述べる。４章では、ｶﾗﾑ・ｺﾚｽｷｰ法を用いた検討例をVPP500, Paragonについて示す。

2．研究のモチーフ

本研究のﾓﾁｰﾌは、処理時間のﾓﾃﾞﾙ化によって、少ない測定で広い範囲の「問題の規模」と「ﾌﾟﾛｾｯｻ数」に対し、ｽｹｰﾗﾋﾞﾘﾃｨを検討することである。

また複数の研究者が作成したﾓﾃﾞﾙをｱｾﾝﾌﾞﾙして利用したい。そこで、多くの計算機に適用でき、あるがままの問題とｼﾐｭﾚｰｼｮﾝｺｰﾄﾞが記述できること、及び他の計算機に無い特定の測定機能を使わないでﾓﾃﾞﾙﾊﾟﾗﾒｰﾀを決定できることを原則としたﾓﾃﾞﾙ化方法を作成することにした。

本方法でﾓﾃﾞﾙ化しﾊﾟﾗﾒｰﾀを測定すれば、ｶｰﾈﾙﾚﾍﾞﾙの並列化方法と並列計算機のｽｹｰﾗﾋﾞﾘﾃｨの検討ができる。このﾚﾍﾞﾙをｼﾐｭﾚｰｼｮﾝｺｰﾄﾞまで拡張したい。

その時、ﾓﾃﾞﾙとﾊﾟﾗﾒｰﾀを手掛かりにして、ﾀｰｹﾞｯﾄ数値ｼﾐｭﾚｰｼｮﾝに対し、ｽｹｰﾗﾋﾞﾘﾃｨを持つ並列計算機を設計することも可能と考える。

3．スケーラビリティ検討方法

3.1　処理時間のﾓﾃﾞﾙ化

数値ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞに基づいた処理時間tのﾓﾃﾞﾙ化は、問題の規模ｎ、ﾌﾟﾛｾｯｻ数Ｎを用い次のように仮定して行う。

- 1ﾌﾟﾛｾｯｻの処理時間t_{1pe}は四則演算の計算回数f(n)に比例する。これを式(1)に示す。
- 並列処理時間t_{npe}はt_{1pe}に比例する。これを式(2)に示す。
- 通信処理時間t_cは通信回数g(n,N)に比例する。これを式(3)に示す。

ここに、問題の規模ｎとはﾒｯｼｭ数等を表す量である。t_0とEは、定数と比例定数である。特にEは効率を表すこととし、1ﾌﾟﾛｾｯｻの最大性能R_a(flops)、通信最大性能R_c(Byte/秒)、Ｎで規格化する。

$$t_{1pe} = t_{s0} + f(n) \cdot R_a^{-1} \cdot E_s^{-1} \qquad (1)$$

$$t_{npe} \fallingdotseq t_{p0} + t_{1pe} \cdot N^{-1} \cdot E_p^{-1} + t_{povh} \qquad (2)$$

t_{povh}は並列ｵﾊﾞｰﾍｯﾄﾞで次式で表わされる。

$$t_{povh} \fallingdotseq t_c + t_{cyn} + t_f + t_j$$

ここに、t_{cyn}は同期時間、t_fとt_jはﾀｽｸ生成と消滅時間である。

$$t_c \fallingdotseq t_{c0} + g(n,N) \cdot R_c^{-1} \cdot E_c^{-1} \qquad (3)$$

ここに、t_{c0}は通信立ち上がり時間、E_cは通信効率である。

f(n)、g(n,N)は、ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞのｺｰﾃﾞｨﾝｸﾞより導出し、f(n)は、四則演算の数のみを数える。また、内部関数の呼び出しは、四則演算と同等に数えることとする。

ｺｰﾄﾞ全体の並列実行時間$(T_{npe})^T$は、並列処理

のｶｰﾈﾙの個数をk_p、逐次処理のそれをk_sとする時、式(4)で表せる。

$$(T_{npe})^T=\sum_{k=1}^{k_p}(T_{npe})_k + \sum_{k=1}^{k_s}(T_{1pe})_k \qquad (4)$$

また、ｺｰﾄﾞ全体の逐次処理時間$(T_{1pe})^T$は、式(5)となる。

$$(T_{1pe})^T=\sum_{k=1}^{k_p}(T_{1pe})_k + \sum_{k=1}^{k_s}(T_{1pe})_k \qquad (5)$$

ここに、$T_{1pe}=\sum_{i=1}^{i_{max}}(t_{1pe})_i$、$T_{npe}=\sum_{i=1}^{i_{max}}(t_{npe})_i$、

で、i_{max}はdoﾙｰﾌﾟ等のｲﾀﾚｰｼｮﾝ回数である。

3.2 ｽｹｰﾗﾋﾞﾘﾃｨ決定要因とﾓﾃﾞﾙﾊﾟﾗﾒｰﾀ

数値ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞのｽｹｰﾗﾋﾞﾘﾃｨ決定要因は、次のa.～c.で示される３つの大項目中に分類できる。式(1)～(3)を用いたモデル化よって、要因が細かく区別できるようになる。

a.ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞの並列性に関するもの
- 並列化率（式(6)に示す） (P)

b.並列ｵｰﾊﾞｰﾍｯﾄﾞに関するもの
- ﾌﾟﾛｾｯｻの使用効率 (E_p)
  (ﾛｰﾄﾞﾊﾞﾗﾝｽ、共有ﾒﾓﾘｺﾝﾃﾝｼｮﾝ、等)
- 並列立ち上がり時間 (t_{p0})
- 通信時間 (t_c)
- 同期時間 (t_{cyn})
- ﾀｽｸ 生成、消滅時間 (t_f,t_j)

c.１ﾌﾟﾛｾｯｻの処理に関するもの
- 演算器の使用効率 (E_s)
  (複数演算器の稼働率等、ﾒﾓﾘｱｸｾｽ競合、ｷｬｯｼｭﾐｽﾋｯﾄ等)
- 演算立ち上がり時間 (t_{s0})
  (ﾒﾓﾘ・ｷｬｯｼｭｱｸｾｽの立ち上がり等)

3.3 測定によるﾓﾃﾞﾙﾊﾟﾗﾒｰﾀの決定

測定は、ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞに経過時間を測定する時計を挿入して行う。ﾓﾃﾞﾙﾊﾟﾗﾒｰﾀE、t_0等は、問題の規模nに対してﾌﾟﾛｾｯｻ数Nを変えて経過時間測定を行い、式(1)、(2)、(3) を最小自乗法でﾌｨｯﾄさせて求める。

3.4 スケーラビリティ検討

(1)E, t_o, t_{povh}の検討

Eは、計算機のﾎﾟﾃﾝｼｬﾙをどのぐらい使用しているかを示す目安となる。全体に対して、Eが小さい箇所、t_o, t_{povh}が大きい箇所に着目する。ｺｰﾃﾞｨﾝｸﾞ、計算ｱﾙｺﾞﾘｽﾞﾑレベルの最適化が大きく寄与している場合がある。

(2)並列化率の検討

次式(6)の並列化率Pから、そのｼﾐｭﾚｰｼｮﾝｺｰﾄﾞが持つ並列性を調べることができる。倍率1/(1-P)が、目安となる。

$$P = \sum_{k=1}^{k_p}(T_{1pe})_k \;/\; (T_{1pe})^T \qquad (6)$$

(3)並列処理時間検討

ﾌﾟﾛｾｯｻ数Nに対する並列処理時間T_{npe}を検討する。ﾘﾆｱﾘﾃｨ、ﾀｰｹﾞｯﾄ性能に必要なﾌﾟﾛｾｯｻ数を検討する。

また、t_{povh}を検討することにより、その問題とｼﾐｭﾚｰｼｮﾝｺｰﾄﾞに対する、並列化ｱﾙｺﾞﾘｽﾞﾑ、並列計算機のｱｰｷﾃｸﾁｬ、通信性能仕様と演算性能のﾊﾞﾗﾝｽ、等が検討できる。

ﾘﾆｱﾘﾃｨが無い場合、項目(2)、(1)の順に遡ることによって原因を特定または、絞ることができる。

4. ｽｹｰﾗﾋﾞﾘﾃｨ検討例

4.1 検討対象

図－１に示すｶﾗﾑ・ｺﾚｽｷｰ法のLU分解部を並列化して、そのｽｹｰﾗﾋﾞﾘﾃｨをVPP500、Paragonについて検討した。ﾌﾟﾛｸﾞﾗﾑはFORTRANで書き、全ての浮動小数点の計算は単精度で行った。

```
parameter(n=5000)
real*4 rL(n,n),s(n)
  do 10 j=1,n
```

```
      do 11 i=1,n
         s(i)=0.
   11 continue
      do 20 k=1,j-1
      do 20 i=j,n
         s(i)=s(i)+rL(j,k)*rL(i,k)
   20 continue
      tj=rL(j,j)-s(j)
      rL(j,j)=tj/sqrt(tj)
      do 30 i=j+1,n
         ti=rL(i,j)-s(j)
         rL(i,j)=ti/sqrt(tj)
   30 continue
   10 continue
```

図－1　ｶﾗﾑ･ｺﾚｽｷｰ法のLU分解部

並列化は、cyclic手続き分割をdo 20の外側のﾙｰﾌﾟに対して行う。配列sとrLは重複ﾛｰｶﾙ配列とし、ﾃﾞｰﾀ分割は行わないｱﾙｺﾞﾘｽﾞﾑとした。各ﾌﾟﾛｾｯｻが計算結果sをdo 30で使用するため、20 continue直後で全ﾌﾟﾛｾｯｻで総和計算を行う。

do 20のﾓﾃﾞﾙ化は、do 10を考慮して行い、式(7)、(8)となる。

$$(T_{1pe})_{do10,20} = \sum_{j=1}^{n} \sum_{k=1}^{j-1} \sum_{i=j}^{n} (t_{1pe})_{do20} = (T_{1pe})_{do20} \cdot n \qquad (7)$$

$$(T_{1pe})_{do20} = 1/2\,(n-1) \cdot (t_{s0})_{do20} + 1/3(n^2-1) \cdot R_a^{-1} \cdot (E_s)_{do20}^{-1}$$

$$(T_{npe})_{do10,20} = \{(t_{p0})_{do10} + (T_{1pe})_{do20} N^{-1} \cdot (E_p)_{do20}^{-1} + t_{povh}\} \cdot n \qquad (8)$$

また、do 10を考慮してdo 30をﾓﾃﾞﾙ化すると式(9)となる。

$$(T_{1pe})_{do10,30} = n \cdot (t_{s0})_{do30} + (n^2+n) \cdot R_a^{-1} \cdot (E_s)_{do30}^{-1} \qquad (9)$$

4.2　VPP500における検討

VPP500の並列化は、図－1に次の追加を行う。

```
      parameter(N$=16)
!xocl processor pe(N$)
!xocl spread do/(pe,index=1:n,part=cyclic)
      do 20 k=1,j-1
      ・・・・・
   20 continue
!xocl end spread sum(s)
```

図－2　VPP500の並列化ﾌﾟﾛｸﾞﾗﾑ

総和計算sum(s)はﾄﾗﾝｽﾎﾟｰｽﾞ転送を用いて行われ[4]、ﾈｯﾄﾜｰｸにｸﾛｽﾊﾞｰｽｲｯﾁを使用しているため、式(10)のようにﾓﾃﾞﾙ化する。

$$(T_c)_{do10} = 2\,\{(t_{c0})_{do20} + (N-1)/N \cdot [(4_{byte})\,n/N] \cdot R_c^{-1} \cdot (E_c^{-1})_{do20}\}\ N \cdot n \qquad (10)$$

ここで、右辺の係数2は、総和計算にﾄﾗﾝｽﾎﾟｰｽﾞ転送が2回必要なことを示す。また、大括弧の右にあるnは、総和計算が、do 10のｲﾀﾚｰｼｮﾝでn回行われることを示す。その隣のNは、n/N個の4byteﾃﾞｰﾀをN個のﾌﾟﾛｾｯｻに転送することを表す。また、(N-1)/Nは自ﾌﾟﾛｾｯｻへは転送を行わないことを表す。また、VPP500の演算速度が1.6Gflopsであるのに対し、転送速度が100Mﾃﾞｰﾀ/sec（単精度）のため、演算時間は無視できると仮定した。

全体の処理時間は、次式(11)、(12)となる。

$$(T_{1pe})^T = (T_{1pe})_{do10,20} + (T_{1pe})_{do10,30} \qquad (11)$$

$$(T_{npe})^T = (T_{npe})_{do10,20} + (T_{1pe})_{do10,30} + (T_c)_{do10} \qquad (12)$$

経過時間測定のため、ｻｰﾋﾞｽｻﾌﾞﾙｰﾁﾝの時計

gettodをdo 20、do 30の前後に挿入した。またsumの時間を測定するため、sumを行わない同一の処理をするdoﾙｰﾌﾟを作成し、それらのdo 20の時間差をsumの処理時間とした。測定は、ﾙｰﾌﾟnを500から5000まで変え、またNを1から16まで変えて行った。ｺﾝﾊﾟｲﾗは、VPP FORTRAN77 EX/VPP V12L20を用い、ｺﾝﾊﾟｲﾙｵﾌﾟｼｮﾝは、-Wxを用いた。またsumの演算を最適化するためのﾜｰｸ領域を、実行時ｵﾌﾟｼｮﾝ-Wl,-Pg2000で確保した。

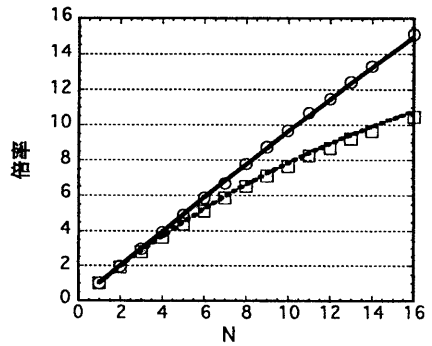
測定した結果に最小自乗法を用い、式(7)～(10)の定数を求めた結果を表－１に示す。表中のE_sは、この計算がVPP500の１ﾌﾟﾛｾｯｻのﾎﾟﾃﾝｼｬﾙを46～60％を引き出していることがわかる。尚、計算にはR_a=1.6Gflops, R_c= 400 MB/秒を用いた。

表－１　VPP500のﾓﾃﾞﾙﾊﾟﾗﾒﾀｰ値

条件	n=5000	条件	N=1
$(E_p)_{do20}$	0.999	$(E_s)_{do20}$	0.605
$(t_{p0})_{do20}$	40 μs	$(t_{s0})_{do20}$	0 μs
$(E_c)_{do20}$	0.493	$(E_s)_{do30}$	0.46
$(t_{c0})_{do20}$	1.10μs	$(t_{s0})_{do30}$	1.7 μs

式(6)から得られる並列化率はほぼ100％である。do 30の計算時間が無視できるためである。

図－３に倍率を示す。○は$(T_{1pe})_{do10,20}/(T_{npe})_{do10,20}$、□は$[(T_{1pe})_{do10,20}+(T_{1pe})_{do10,30}]/[(T_{npe})_{do10,20}+(T_{1pe})_{do10,30}+(T_c)_{do10}]$である。この並列化ｱﾙｺﾞﾘｽﾞﾑの性能が、総和計算の性能に依存していることが判る。尚、図中の記号が実測値、線がﾓﾃﾞﾙからの計算値である。



図－３　VPP500のスケーラビリティ

４．３　Paragonにおける検討

Paragonの並列化ﾌﾟﾛｸﾞﾗﾑを図－４に示す。図－１に次の変更、追加を行う。iamは自ﾌﾟﾛｾｯｻ番号、nodesはﾌﾟﾛｾｯｻ数である。l_r,l_cは各々、列と行のﾌﾟﾛｾｯｻ数の$\log_2$をとったものである。tgdsumで、全ﾌﾟﾛｾｯｻ間の総和計算を行う。尚、tmpは総和計算のためのﾜｰｸ配列である。

```
      dimension tmp(n)
      iam=mynodes()
      nodes=numnodes()
      istat=nx_app_rect(lr,lc)
       ・ ・ ・ ・ ・
      do 20 k=iam+1, j-1, nodes
      do 20 i=j,n
       ・ ・ ・ ・ ・
   20 continue
      call tgdsum(s,n,tmp)
```

図－４　Paragonの並列化ﾌﾟﾛｸﾞﾗﾑ

総和計算を行うｻﾌﾞﾙｰﾁﾝtgdsumの作成に当たっては、ﾒｯｼｭﾈｯﾄﾜｰｸにおける衝突を避けるため、ﾈｯﾄﾜｰｸの列方向と行方向の順にﾂﾘｰ 状に転送を行う。ﾃﾞｰﾀは、ﾂﾘｰ の頂点の１ﾌﾟﾛｾｯｻに集めながら総和計算され、計算結果は来た道を逆戻りする形で配布される。これを、式(13)のようにﾓﾃﾞﾙ化する。

$$(T_c)_{do10} = \{ 2[(t_{c0})_{tgdsum} + n(l_r+l_c) \cdot (4_{byte}) \cdot R_c^{-1} \cdot (E_c^{-1})_{tgdsum}] + (t_{s0})_{tgdsum} + n(l_r+l_c) \cdot R_a^{-1} \cdot (E_s^{-1})_{tgdsum} \} n \quad (13)$$

ここに、$N=2^{lr}+2^{lc}$である。

ここで、右辺の第一、二項は転送時間を表し、係数２はﾃﾞｰﾀ転送が往復行われることを表す。

また第三、四項の総和計算は、往路のみ行われる。また、大括弧の右にあるnは、総和計算が、do 10のｲﾀﾚｰｼｮﾝでn回行われることを示す。

測定のため、ｻｰﾋﾞｽ関数dclockをdo 20、do 30、tgdsumの前後に挿入した。時間測定は、ﾙｰﾌﾟnを500から2500まで変え、またNを1から64まで変えた。ｺﾝﾊﾟｲﾗは、PGFTN(Version Rel 4.5)を用い、ｺﾝﾊﾟｲﾙｵﾌﾟｼｮﾝは、-nx -Mvectを用いた。
-Mvectはﾍﾞｸﾄﾙ計算を行うｵﾌﾟｼｮﾝであり、tgdsumには、必須であった。

測定した結果に、式(7)～(9)及び(11)をﾌｨｯﾃｨﾝｸﾞさせた結果を表－２に示す。表中のE_sは、この計算でParagonの１ﾌﾟﾛｾｯｻの効率が0.4～34%とばらついていることがわかる。尚、計算にはR_a=0.1Gflops（単精度）、R_c=200MB/秒を用いた。

表－２　Paragonのﾓﾃﾞﾙﾊﾟﾗﾒｰﾀ値

条件	n=2500	条件	N=1
$(E_p)_{do20}$	0.983	$(E_s)_{do20}$	0.338
$(t_{p0})_{do20}$	0.40 μ s	$(t_{s0})_{do20}$	0.94 μ s
$(E_c)_{tgdsum}$	0.113	$(E_s)_{do30}$	0.00380
$(t_{c0})_{tgdsum}$	0 μ s	$(t_{s0})_{do30}$	.0017 μ s
$(E_s)_{tgdsum}$	0.0684		
$(t_{s0})_{tgdsum}$	0 μ s		

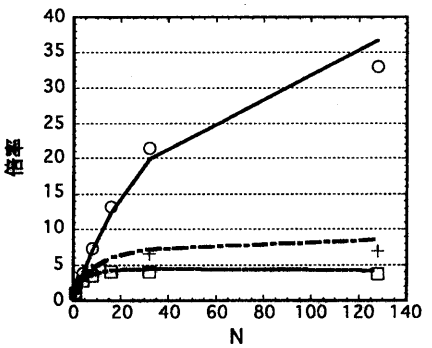
式(6)の並列化率Pは90%で、倍率1/(1-P)は10倍である。

図－５に倍率を示す。○は$(T_{1pe})_{do10,20}$/$(T_{npe})_{do10,20}$、＋は$[(T_{1pe})_{do10,20}+(T_{1pe})_{do10,30}]$/$[(T_{npe})_{do10,20}+(T_{1pe})_{do10,30}]$、□は$[(T_{1pe})_{do10,20}+(T_{1pe})_{do10,30}]$ / $[(T_{npe})_{do10,20}+(T_{1pe})_{do10,30}+(T_c)_{do10}]$である。$(E_s)_{do\ 30}$が極端に低いことが並列性に影響を与え最大性能を10倍にしている様子、更にtgdsumにおける総和計算がｽｹｰﾗﾋﾞﾘﾃｨに影響し、採用した並列ｱﾙｺﾞﾘｽﾞﾑとの相性を悪くしていることがわかる。尚、図中の記号が実測値、線がﾓﾃﾞﾙからの計算値である。

５．議論

ﾓﾃﾞﾙはﾛｰﾄﾞﾊﾞﾗﾝｽ等の効果を取り入れる必要がある。また、１ﾌﾟﾛｾｯｻで実行できないｺｰﾄﾞの、E_s, t_{s0}の定義の方法の検討が必要である。これらは、各ｱｰｷﾃｸﾁｬの計算機上で、ｼﾐｭﾚｰｼｮﾝｺｰﾄﾞを並列化することによって研究する予定である。

意味があり、精度の高いﾓﾃﾞﾙを作成するためには、ｱﾌﾟﾘｹｰｼｮﾝ側の視点を持った人の協力が必要である。



図－５　Paragonのスケーラビリティ

６．謝辞

Paragonの総和計算ｻﾌﾞﾙｰﾁﾝは、日本原子力研究所、計算科学技術推進ｾﾝﾀｰの大田敏郎氏作成のものを利用した。感謝したい。

〔参考文献〕

[1]Orii S., Fully Vectorizable Particle-Mesh Model Codes on the FACOM VP-100/200, Proc. ANS/ENS /JAES Int. Topical Meeting, Knoxville, Tennessee, April 9-10,Vol.1, pp.22 (1985)

[2]Hockney R., Curington I., $f_{1/2}$:A parameter to characterize memory and communication bottlenecks, Parallel Computing Vol. 10, pp.277-286 (1989)

[3]Formella A., Muller S., Paul W., Bingert A., Isolating the Reasons for the Performance of Parallel Machines on Numerical Programs, Automatic Paralllelization, Wieweg Publishing (1994)

[4]岡田、坂本、浅井：VPP500ｼｽﾃﾑのｿﾌﾄｳｪｱ、電子情報通信学会論文誌 D-I Vol. J78-D-I, No.2, pp.149-161 (1995)