P,Q 比が変更可能な ScaLAPACK のコスト見積もり関数の開発

Development of the Cost Estimated Function for the ScaLAPACK can change ratio of P and Q

斉藤 宏樹
Hiroki SAITO

Abstract: To execute the parallel and distributed applications effciently on a Computional Grid, we need to select appropriate machines that meet the characteristics of applications. In this paper, the linear algebra library, ScaLAPACK, is targeted and optimum scheduling is discussed. In this study, developed the Cost Estimated Function for the ScaLAPACK can change ratio of P and Q. Compared to the result of measured time, estimated time is large and optimun parameters of P and Q differed.

1　はじめに

グリッド上の計算資源を利用して，効率的に分散・並列アプリケーションを実行するためには，アプリケーションの特性に合わせて，適切な計算資源を選び出す必要がある．現在，このようなスケジューリング問題を目的とした研究が多くなされている [1]．

本研究では並列線形ライブラリ ScaLAPACK[2] の一関数を対象にしたスケジューリングについて検討する．グリッド上の計算資源を利用するアプリケーションの開発・実行を行うための技術やツールを提供する Grid Application Development Software（GrADS） Project[3] は，ScaLAPACK の実行時間を見積もる関数を開発し，YarKhan らはその関数を利用したスケジューリング手法の実験を行っている [4]．しかし，GrADS Project が開発したコスト見積もり関数は，実行プロセスへの行列の割り当て方が限定されており，最適な実行時間を求めることができない．

そのため，本研究ではグリッドの計算資源を利用した最適な ScaLAPACK のスケジューリング手法を開発することを目標に，行列の割り当て方を決定するパラメータ，$P \times Q$ の値を自由に変更できる見積もり関数の開発を行う．

2　ScaLAPACK

2.1　概要

ScaLAPACK（Scalable LAPACK）は，共有メモリ用の線形計算ライブラリである LAPACK（Linear Algebra PACKage）を PVM や MPI を使用して分散並列実行できるように開発された分散メモリ用の高性能数値計算ライブラリであり，密行列の分解，線形方程式，線形最小二乗問題，固有値問題，特異値問題を解くことが可能である．通信層に BLACS（Basic Linear Algebra Communication Subprogram）を使用しており，係数行列をプロセスグリッド (P, Q) に基づいて複数のプロセスにブロックサイクリックに分割することが可能で，LU 分解・QR 分解を分散して実行することができる．

Fig. 1 に $N \times N$ の係数行列を $(P, Q) = 2 \times 3$ のプロセスグリッドに基づいてブロックサイクリックにデータ分割する例を示す．まず係数行列を一つのプロセスに割り当てられる最小の行列サイズ，ブロックサイズ (NB) に分割する．そして行方向に 2 つのプロセスを，列方向には 3 つのプロセスをプロセスグリッド 2×3 に従って，繰り返し順に割り当てることで各プロセスへのデータ分割を行う方法である．この時，ScaLAPACK のパラメータ N，NB，(P, Q) の値とプロセスを割り当てる計算資源の性能や数によって，行列の演算処理時間や通信時間が変化して実行時間が変動することになる．

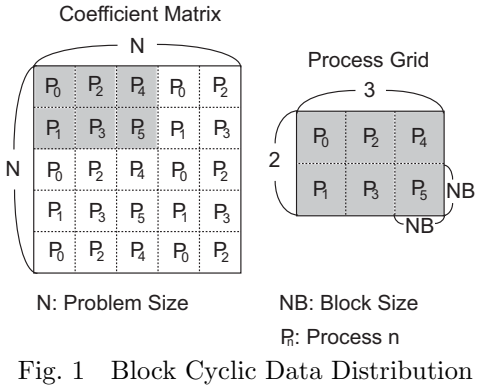

Fig. 1　Block Cyclic Data Distribution

2.2　PDGESV ルーチンのアルゴリズム

ScaLAPACK には数値計算を行うための多数のルーチンが用意されているが，本研究では GrADS Project のコスト見積もり関数と同様に，LU 分解によって密行列連立一次方程式を解く PDGESV ルーチンを対象とし，その実行時間を予測する．

以下に P.Q 比を (P,Q)=(2,3) とした場合を例に，PDGESV ルーチンのアルゴリズムを示す．

1. ブロック化 LU 分解を実行 (PDGETRF)

(a) 部分行列 LU 分解フェーズ (PDGETF2：LEVEL 2 BLAS)

i. ピボットの探索 (PDAMAX：ベクトルの最大要素を探索)

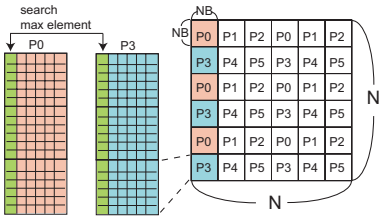

Fig. 2　PDAMAX

ii. ブロック列において行交換 (PDSWAP：ベクトル交換)

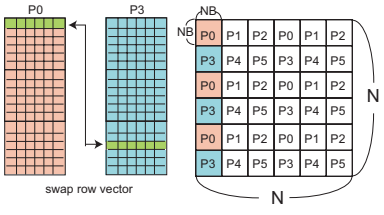

Fig. 3　PDSWAP

iii. ブロック列 (L) の計算 (PDSCAL：ベクトルをスカラー倍)

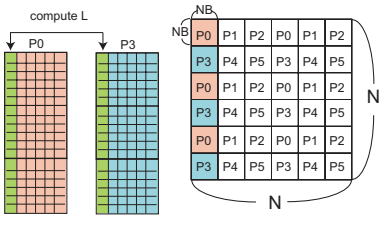

Fig. 4　PDSCAL

iv. ブロックパネル (U) の更新 (PDGER：ランク 1 の行列更新)

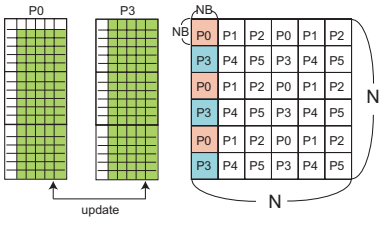

Fig. 5　PDGER

v. 参照する列の要素を右に一つ進めて，(i) に戻る．NB 回繰り返すと終了．

vi. ピボット情報の送信 (IGEBS2D：row プロセスへ BroadCast)

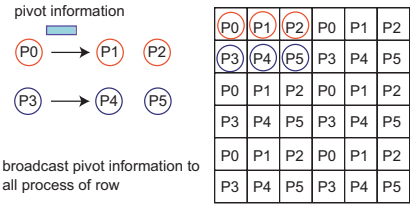

Fig. 6　Broadcast pivot information

(b) 各ブロック列において行交換 (PDLASWP：複数ベクトルの交換)

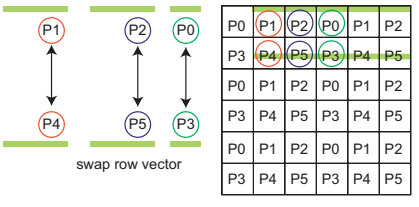

Fig. 7　PDLASWP

(c) ブロック行 (U) の更新フェーズ

i. 分解済みブロックパネル (L) の送信 (row プロセスへ BroadCast)

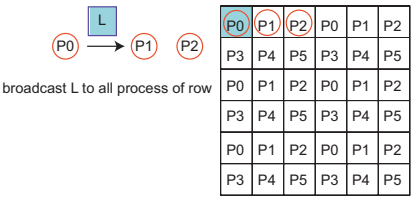

Fig. 8　BroadCast

ii. ブロック行 (U) の更新 (PDTRSM：LEVEL 3 BLAS)

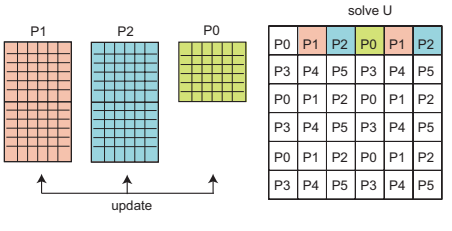

Fig. 9　PDTRSM

(d) 未更新行列の更新フェーズ

i. 分解済みブロック (L) の送信 (column プロセスへ BroadCasr) j

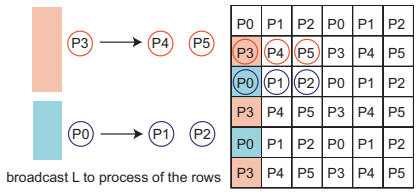

Fig. 10　BroadCast

ii. 分解済みブロック (U) の送信 (row プロセスへ BroadCast)

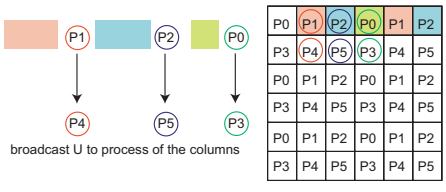

Fig. 11　BroadCast

iii. 未更新行列の更新 (PDGEMM：行列積の計算)

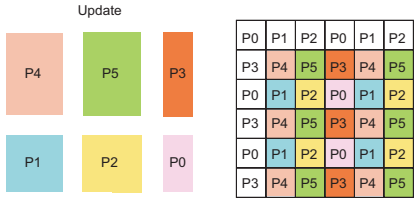

Fig. 12　PDGEMM

(e) ブロックを一つ右へ進めて，a に戻る．N/NB 回繰り返すと終了．

2. 求解 (PDGETRS) を実行

なお，ブロック化 LU 分解の演算量 $O(N^3)$ に対して求解の演算量は $O(N^2)$ となっている．

3　P,Q 比が変更可能な ScaLAPACK のコスト見積もり関数の作成

GrADS Project が開発した ScaLAPACK のコスト見積もり関数は，密行列連立一次方程式を解く PDGESV ルーチンの実行時間を予測する．しかしこの関数は，P,Q 比 (P, Q) のパラメータ P が「1」で固定されており，一次元のプロセスグリッドにおけるシミュレーションしか行えない．一次元のプロセスグリッドでは他のプロセスが計算待ちになる時間が長くなるため，通常はロードバランスが良く性能が引き出せる二次元のプロセスグリッドを適用する．

そのため本研究では，P の値も変更することが可能な見積もり関数の開発を行う．見積もるコストは演算コストと通信コストであり，実行時間に強く影響を与える主要な部分のみを見積もる．

3.1　演算コストの見積もり

演算コストは，行列の更新に必要な浮動小数点の演算量（Flops）で見積もる．見積もるステップは以下の 3 つであり，それぞれの演算コストは Table 1 より見積もることができる．

1. ブロックパネル (U) の更新 (PDGER：ランク 1 の行列更新)
2. ブロック行 (U) の更新 (PDTRSM：LEVEL 3 BLAS)
3. 未更新行列の更新 (PDGEMM：行列積の計算)

Table 1　Flops in each Level of BLAS

BLAS	Flops
Level 1	$2n$
Level 2	$2n^2$
Level 3	$2n^3$

まずブロックパネル (U) の更新 (PDGER：ランク 1 の行列更新) に必要な演算コストは，式 (1) で計算できる．

$$\sum_{k=1}^{NB-1} 2((N/P)-k)(NB-k) \tag{1}$$

続いて，ブロック行 (U) の更新 (PDTRSM：LEVEL 3 BLAS) に必要な演算コストは，式 (2) で計算できる．

$$2(NB)\times(N/Q)\times(NB) \tag{2}$$

未更新行列の更新 (PDGEMM：行列積の計算) に必要な見積もりコストは，式 (3) で計算できる．

$$2(N/P)\times(N/Q)\times(NB) \tag{3}$$

これらのコストを (N/NB) 回足し合わせて，PDGESV ルーチンの演算コストとを見積もる．式 (4) にその式を示す．

$$\begin{aligned}\sum_{j=1}^{N/NB}\{\sum_{k=1}^{NB-1}\{2((N-(j-1)\times NB)/P)-k)(NB-k)\}\\ +2(NB)\times((N-j\times NB)/Q)\times(NB)\\ +2((N-j\times NB)/P)\times((N-j\times NB)/Q)\times(NB)\}\end{aligned} \tag{4}$$

3.2 通信コストの見積もり

通信コストは，行列の更新に必要なピボット情報やベクトル情報の通信量（Byte）で見積もる．見積もるステップは，以下の 5 つである，

1. ピボット情報の送信 (IGEBS2D：row プロセスへ BroadCast)
2. 各ブロック列において行交換 (PDLASWP：複数ベクトルの交換)
3. 分解済みブロックパネル (L) の送信 (row プロセスへ BroadCast)
4. 分解済みブロック (L) の送信 (row プロセスへ BroadCast)
5. 分解済みブロック (U) の送信 (column プロセスへ BroadCast)

「2」のステップは一対一の通信であり，それ以外のステップは BroadCast で通信する．BroadCast による通信コストの見積もりと一対一による通信コストの見積もりについて説明する．

3.2.1 BroadCast による通信コスト

ScaLAPACK は通信レイヤに BLACS を使用しており，通信トポロジーを複数の方式から選択することができる．本研究で開発する関数は,「Split-ring」方式を選択した際の通信コストを見積もる.「Split-ring」方式による BroadCast 通信の流れを Fig. 13 に示す.

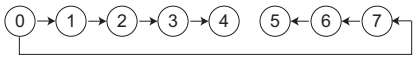

Fig. 13 Split ring broadcast

開発する関数では Split-ring による通信コストを，全プロセスへデータが送信されるまでの通信コストで見積もるのでは無く，ある特定のプロセス（プロセス番号 0）の送受信が完了するまでの通信コストで見積もる．Fig. 14 に，プロセス数が 3 である場合の通信コストの見積もりを示す.

Fig. 14 は，ブロックパネル分解後のピボット情報の送信に要する通信コストを例としている．ピボット情報の BroadCast は，分解したプロセスが送信元となって行う．そのため，分解したプロセスが異なると，見積もる通信コストも異なる．Fig. 14 では，分解プロセスが 0〜2 となる各場合の見積もりを示している．なお，C_1 が BroadCast 通信の最初に発生する通信コストであり，

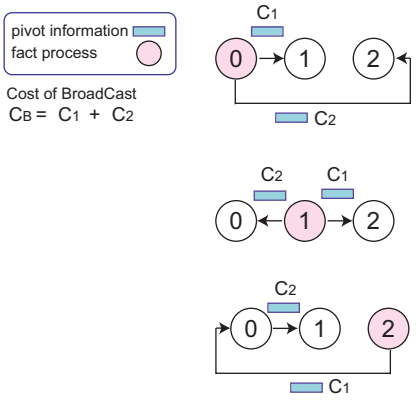

Fig. 14 Cost Estimated of Communication in three processes

C_2 が次に発生する通信コストである．Fig. 14 を見ると，全プロセスへの送信コストを見積もっていると捉えることも可能であるが，作成した関数では，常にプロセス 0 が送受信を完了するまでの時間を見積もっている．プロセス数 4，5 の通信コストを Fig. 15，Fig. 16 に示す.

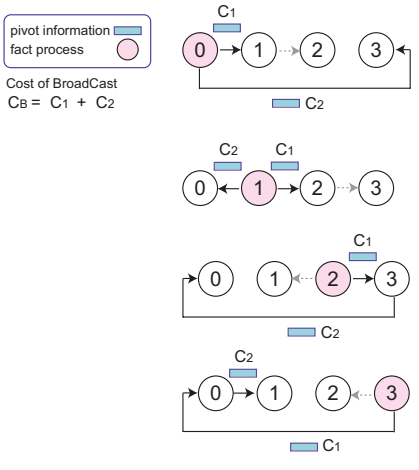

Fig. 15 Cost Estimated of Communication in four processes

Fig. 14 から Fig. 16 より，プロセス数と分解プロセス番号によって通信コスト C を加える回数が異なることがわかる．これは，プロセス番号 0 の送受信が完了した時点で見積もりを終了しているためである.

通信コスト C の値については，ステップによって異なる．まず，ピボット情報の送信では int 型（ピボット情報）の一次元要素を送信するので，C は式 (5) となる.

$$NB \times 4 \tag{5}$$

分解済みブロックパネル (L) の送信では，double 型（行列要素）の二次元要素を送信するので，C は式 (6)

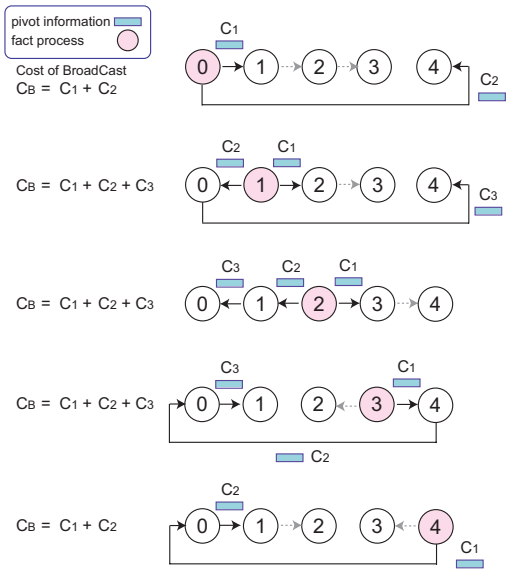

Fig. 16　Cost Estimated of Communication in five processes

となる．

$$NB \times NB \times 8 \tag{6}$$

分解済みブロック (L) の送信では，double 型（行列要素）の二次元要素を送信するので，C は式 (7) となる．

$$N/P \times NB \times 8 \tag{7}$$

そして分解済みブロック (U) の送信で，double 型（行列要素）の二次元要素を送信するので，C は式 (8) となる．

$$N/Q \times NB \times 8 \tag{8}$$

これらの通信コストを column プロセス方向と row プロセス方向とに分けて見積もることで，PDGESV ルーチンの BroadCast 通信コストを見積もる．式 (9)，式 (10) にそれらを示す．なお，S は BroadCast 時に足し合わせる通信コストの回数を示しており，プロセス数と分解プロセス番号によって決定される値である．

$$BcastCost_{col} = \sum_{j=1}^{N/NB} \sum_{k=0}^{S} \{(N - j \times NB)/Q \times NB \times 8\} \tag{9}$$

$$\begin{aligned} BcastCost_{row} &= \sum_{j=1}^{N/NB} \sum_{k=0}^{S} \{NB \times 4 + NB \times NB \times 8 \\ &+ (N - j \times NB)/P \times NB \times 8\} \end{aligned} \tag{10}$$

3.2.2　一対一による通信コスト

各ブロック列における行交換では，一対一による通信が発生する．一回の行交換で double 型（行列要素）の一次元要素を送受信するので，その通信コストは式 (11) となる．

$$N/Q \times 8 \times 2 \tag{11}$$

行交換が発生する回数とプロセスを正確にシミュレーションすることは不可能であるため，作成した関数では乱数を用いて確率により見積もる．分解プロセスが所有する行列中に最大要素が無い場合，行列交換は発生する．そして各プロセスが所有する配列は，プロセスグリッド P,Q 比によってほぼ均等に割り当てられている．ゆえに行列交換が発生する確率は，1 から分解プロセスが所有する行の割合 $1/P$ を引いたものになる．

よって，$1 - 1/P$ の確率で式 (11) の通信コストが発生することになる．発生した回数を R とすると，PDGESV ルーチンの一対一による通信コストは，式 (12) となる．

$$SendRecvCost = \sum_{k=1}^{R} N/Q \times 8 \times 2 \tag{12}$$

3.3　実行時間の見積もり

PDGESV ルーチンの実行時間は，これまでに示した演算コストと通信コストを各フェーズを担当する計算資源の性能で割ることで算出するようにした．なお演算時間の見積もりは式 (13) で行っている．

$$\frac{FactCost(Flops)}{Peak(\%)/100 \times Filtering(Flops/Hz) \times Clock(Hz)} \tag{13}$$

式 (13) の Filtering は，クロック周波数を浮動小数点演算回数に変換する処理で，CPU の演算ユニットの数で決定される．Peak は，用いた BLAS と CPU とキャッシュのアーキテクチャによって決定される，DGEMM ルーチンを実行する際のピーク性能である．コスト見積り関数ではピーク性能で計算されることを想定して，FactCost(分解・更新に要する演算回数) を浮動小数点演算のピーク性能値で割ることで，見積もり時間 (sec) を算出している．

4　数値実験

本研究室にある二つのクラスタ，Gregor と Xenia を用いて ScaLAPACK を実行し，シミュレータによる見積もり時間と実測値について比較検討する．また，並列・分散プログラムの性能測定ツール Paradyn[5] を使用して ScaLAPACK の CPU 時間 (演算時間) や通信による送受信量を測定し，見積もり値との比較を行う．なお，

本研究では計算資源情報の収集に NWS[6] を，BLAS の高速化に ATLAS[7] と High Performance BLAS[8] を使用した．

4.1 Paradyn

Paradyn は並列・分散プログラムの性能測定ツールであり，実行アプリケーションに動的に挿入されて性能を測定する．基本的に Paradyn は測定するアプリケーションのソースコードの修正を必要とせず，特別なコンパイラも必要としない．実行アプリケーションのバイナリファイルされあれば測定可能で，測定するデータもユーザが自由に決定でき，測定のオーバーヘッドが小さい特徴もある．MPI もサポートしており，BroadCast に要した時間や Sendrecv に要した時間も計測できる．

4.2 NWS

NWS(Netowork Weather System) は，ネットワークおよび計算資源の情報を周期的にモニタリングし，それらの計測値に基づいて将来の値を予測する分散システムである．NWS には 4 種類のプロセスが存在し，NameServer, Sensor，PersisitentState，Forecaster がある．本実験ではこれらのうち Forecaster プロセスを起動させず，一定の間隔 (10 秒) で現時点でのネットワークのバンド幅とレイテンシ，CPU 利用率 (新たに起動するプロセスに対する) のみを収集するようにした．

Table 2 に NWS によって収集した計算資源情報を示す．CPU 利用率については個々のノードについて，ネットワークバンド幅，レイテンシに関しては，全ノード間について計測した．なお，各クラスタで用いたネットワークは 100BASE Ethernet である．

Table 2　Dynamic Information by NWS

CPU Available	[0.0-1.0]
Bandwidth(Mbps)	-
Latency(μsec)	-

4.3 ATLAS

BLAS を高速化するために Gregor クラスタにおいては ATLAS を用いた．ATLAS(Automatically Tuned Linear Algebra Software) は，アーキテクチャに合わせて最適な BLAS を自動生成するソフトウェアである．Level3 の BLAS を使用した DGEMM ルーチンに対して，Pentium(I〜III) のアーキテクチャは約 75%のピーク性能を示しているが，本報告では GrADS Project で用いられていた値 60%を用いた．

4.4 High Performance BLAS

BLAS を高速化するために Xenia クラスタにおいては High Performance BLAS を用いた．各アーキテクチャに合わせて最適な BLAS ライブラリが用意されており，DGEMM の対ピーク性能効果が示されている．Xenia のアーキテクチャに合わせて PentiumIV，L2/L3 Cache 512KB 用の「libgoto_p4_512-r0.94.so.gz」を用いた．DGEMM のピーク性能は 87.5%である．

4.5 実験環境

実験に用いた各クラスタの静的情報を Table 3 に示す．

Table 3　Static Information of Gregor and Xenia

Cluster	Gregor	Xenia
Clock Frequency(MHz)	1000	2400
FLOPS/Clock Frequency(Hz)	1.0	2.0
DGEMM Peak performance(%)	60	87.5
Memory(MB)	512	512
Number of Nodes	30	48

4.6 ScaLAPACK のパラメータ

Table 4，Table 5 に Gregor クラスタと Xenia クラスタで用いた ScaLAPACK のパラメータを示す．Gregor クラスタを用いた実験ではプロセスグリッドを P × Q=30 になるように設定し，Xenia クラスタでは 48 になるように設定した．1 ノードで 1 プロセスを実行した．

Table 4　parameters of ScaLAPACK in Gregor

Problem Size(N)	9600
Block Size(NB)	80
Process Grid(P,Q)	(1 × 30),(2 × 15),(3 × 10),(5 × 6) (6 × 5),(10 × 3),(15 × 2),(30 × 1)

Table 5　parameters of ScaLAPACK in Xenia

Problem Size(N)	11520
Block Size(NB)	80
Process Grid(P,Q)	(1 × 48),(2 × 24),(3 × 16),(4 × 12) (6 × 8),(8 × 6),(12 × 4),(16 × 3) (24 × 2),(48 × 1)

4.7 実験結果

Gregor クラスタと Xenia クラスタで得られた実験結果を示す．

4.7.1 Gregor クラスタ

Fig. 17 にプロセスグリッド ($P \times Q$) を変更して ScaLAPACK とシミュレータを実行した結果を示し，それぞれの実測値と見積もり値を比較する．また Fig. 18 に見積もり値の誤差を示す．

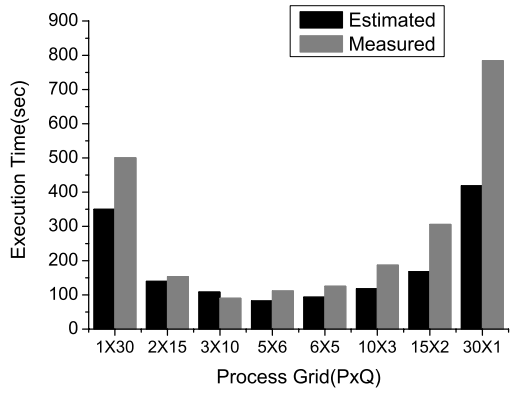

Fig. 17　Estimated execution time and Measured execution time in Gregor

Fig. 17 から，各プロセスグリッドにおける実行時間の実測値と見積り値の傾向を見ると，実測値の最小値が 3×10 の場合に対して見積り値は 5×6 となっており，最適なプロセスグリッドの値が異なっていることが確認できる．他のプロセスグリッド間の実測・見積り時間の大小関係は同じである．

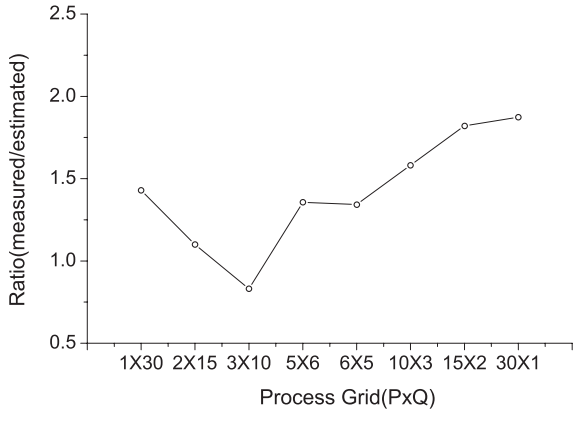

Fig. 18　Ratio of measured execution time to estimated execution time in Gregor

Fig. 18 は，各プロセスグリッドにおける見積り値に対する実測値の割合を示しており，Ratio の値が 1.0 に近いほど見積もり値が実測値に近いことを示している．P の値が 3 から大きくなるにつれて誤差が大きくなっていることが確認できる．P の値が大きくなると行列要素の交換を行う通信が多く発生する特徴があるが，その通信量の測定に誤差があると考えられる．

Fig. 19 に P,Q 比を変化させた際の ScaLAPACK の CPU 時間 (演算時間) について，Paradyn による実測値の計測結果とシミュレータによる見積もり値の比較結果を示す．

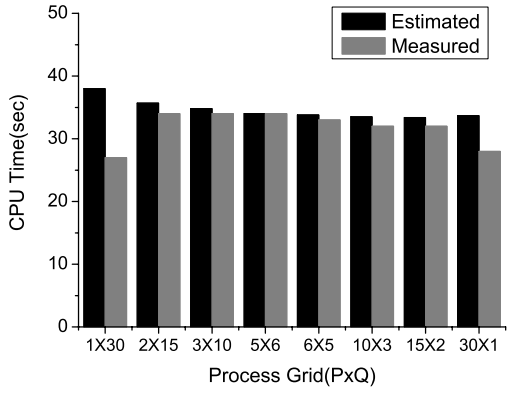

Fig. 19　Estimated cpu time and Measured cpu time in Gregor

Fig. 19 の結果から，行列分解・更新にようする演算時間の誤差は小さいことが確認できる．次に ScaLAPACK の P,Q 比を変化させ，全プロセスおよび 1 プロセスの送受信量を Paradyn によって計測した結果を Fig. 20 に示す．なお，Fig. 20 にはシミュレータによる送受信量の見積もり値の結果も示す．

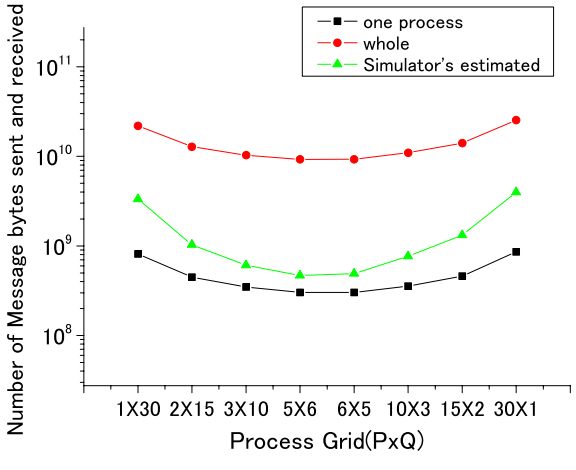

Fig. 20　Number of Message bytes sent and received in Gregor

Fig. 20 より，シミュレータで見積もった送受信量は，1 プロセスの値よりも大きく，全プロセスの値よりも小さいことがわかる．ScaLAPACK は実行時間を決定す

るプロセスが一意に決定されないため，各プロセス間の送受信量を考慮する必要があり，1 プロセスまたは全プロセスの送受信量のみを見積もっても予測できない．そのため作成したシミュレータは平均的な送受信量を見積もるように作成している．Fig. 20 の結果を見ると，各プロセスグリッドの値における送受信量の大小関係は一致していることが確認できる．

4.7.2 Xenia クラスタ

Fig. 17 にプロセスグリッド ($P \times Q$) を変更して ScaLAPACK とシミュレータを実行した結果を示し，それぞれの実測値と見積もり値を比較する．また Fig. 22 に見積もり値の誤差を示す．

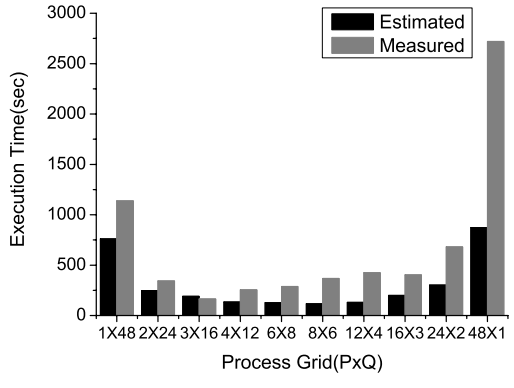

Fig. 21 Estimated execution time and Measured execution time in Xenia

Fig. 21 から，各プロセスグリッドにおける実測値と見積り値の傾向を見ると，実測値の最小値が 3×16 の場合に対して見積り値は 8×6 となっており，最適なプロセスグリッドの値が異なっていることが確認できる．他のプロセスグリッド間の実測・見積り時間の大小関係についても異なっている．

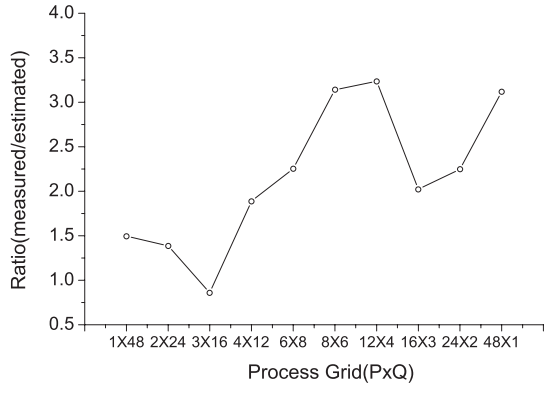

Fig. 22 Ratio of measured execution time to estimated execution time in Xenia

Fig. 22 を見ると，Gregor クラスタでの実験と同様に P の値が 3 より大きくなると誤差が大きくなっており，Gregor クラスタの実験より誤差が大きくなっている．

5 まとめ

本研究では P,Q 比が変更可能な ScaLAPACK のコスト見積もり関数を作成し，クラスタ環境における ScaLAPACK の実行時間を見積もり，実測値と比較した．その結果, P の値が大きい場合に実測値との誤差が大きくなることが確認された．演算時間については Paradyn で計測した結果，実測値との誤差が小さいことが確認された．各 P,Q の値における送受信量の増減の傾向については，見積もり値と実測値で等しいことが確認されたが，最適な P,Q の値が異なる原因についてさらに調査する予定である．

参考文献

1) Francine D. Berman, Rich Wolsky, Silvia Figueria, Jennifer Schopf, and Gary Shao. Application-level scheduling on distributed heterogeneous networks. In ACM, editor, Supercomputing '96 Conference Proceedings: November 17-22

2) ScaLAPACK Project http://www.netlib.org/scalapack/

3) Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennaart Johnsson, Ken Kennedy, Carl Kesselman, John Mellor-Crummey, Dan Reed, Linda Torczon, and Rich Wolski. The GrADS Project: Software support for high-level Grid application development *The international Journal of High Performance Computing Applications 327-344,November, 2001.*

4) Asim YarKhan, Jack J. Dongarra Experiments with Scheduling Using Simulated Annealing in Grid Environment, Workshop on Grid Computing, June 7, 2002.

5) Paradyn Parallel Performance Tools http://www.cs.wisc.edu/~paradyn/

6) Rich Wolski,Neil T. Spring, and Jim Hayes. The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing *Future Generation Computer Systems 757-768, 1999.*

7) Automatically Tuned Linear Algebra Software http://math-atlas.sourceforge.net/

8) High-Performance BLAS by Kazushige Goto http://www.cs.utexas.edu/users/flame/goto/