ハイパフォーマンス
コンピューティング 91－13
（２００２．８．２１）

階層的マスタワーカ方式による BMI 固有値問題の Grid 計算

夏目　亘
東京工業大学

合田　憲人
東京工業大学 /
科学技術振興事業団,さきがけ研究 21

二方　克昌
科学技術振興事業団

数値最適化の一つである BMI 固有値問題を高速計算するにあたって，複雑な Grid 計算環境下において従来のマスタワーカ方式では用意した計算資源に対する性能のスケーラビリティを得ることは難しい．そこで本稿はこのマスタワーカ方式が持つマスタ処理や通信のボトルネックを取り除くために，階層的マスタワーカ方式を提案する．また，それを各所の PC クラスタ上に実装し，様々な計算機環境における提案手法の有効性を示す．

Grid Computing scheme for the BMI Eigenvalue Problem with Hierarchical Master-Worker Paradigm

Wataru Natsume
Tokyo Institute of Technology

Kento Aida
Tokyo Institute of Technology /
PRESTO, JST

Yoshiaki Futakata
JST

The parallel algorithm with master-worker paradigm to solve the BMI Eigenvalue Problem, which is one of optimization problems, has been proposed. However, this algorithm has a problem for scalability of performance because of performance bottleneck on a master and communication overhead. This paper proposes new algorithm with hierarchical master-worker paradigm to improve the scalability and evaluates effectiveness of the proposed algorithms.

1. はじめに

BMI（Bilinear Matrix Inequality）固有値問題は，双線形行列の最大固有値を最小化する解を求めることを目的とした数値最適化問題の一つである。この問題は，ヘリコプター旋回動作制御をはじめとした多くの制御システムの解析設計やオペレーションズリサーチ等の分野で用いられるが，莫大な計算時間が必要であるため，求解時間の短縮が切望されている．これに関して，分枝限定法とマスタワーカ方式を用いた並列解法が[1]によって提案されている．しかし[1]の手法ではネットワーク上に存在する多くの計算資源を利用できる Grid 計算環境においては性能のスケーラビリティの面で限界が見られる．そこで，本稿はこの[1]によって提案されたアルゴリズムを用い，従来の単純なマスタワーカ方式に階層化を施す手法を提案する．また，それを実際の Grid 上に実装した上で性能を評価した結果，様々な計算環境において提案手法の有効性が示された．

以降，2 節では BMI 固有値問題及びマスタワーカ方式について，3 節では本稿が提案する階層的マスタワーカ方式について，4 節ではその性能評価を述べ，最後の 5 節でまとめと今後の課題を説明する．

2. BMI 固有値問題解法

本節では BMI 固有値問題について説明すると共に，従来のマスタワーカ方式による問題の並列解法について概説する．

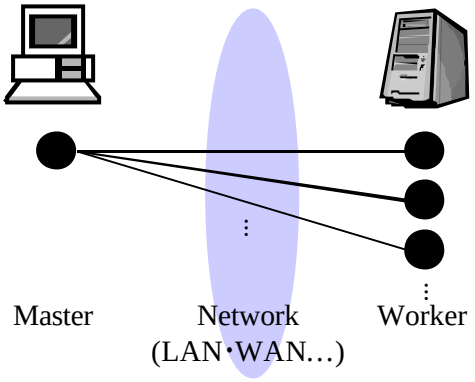

図 1マスタワーカ方式

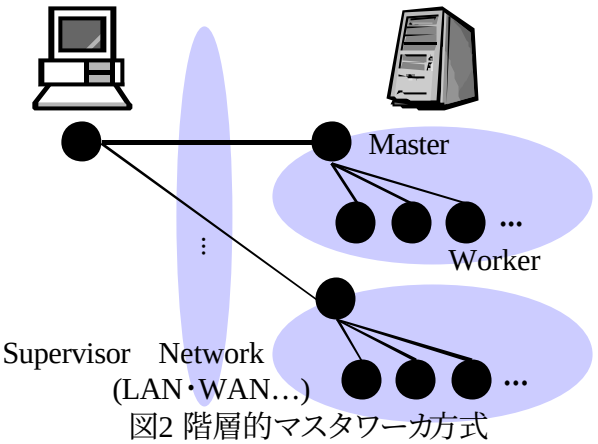

図2 階層的マスタワーカ方式

2.1 BMI 固有値問題

BMI 固有値問題とは以下の式で表される双線形行列関数 F(x,y)

$$F(x,y)=F_{00}+\sum_{i=1}^{n_x}x_iF_{i0}+\sum_{j=1}^{n_y}y_jF_{0j}+\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}x_iy_iF_{ij}$$

$$x=(x_1,\ldots,x_{n_x})^T,\ y=(y_1,\ldots,y_{n_y})^T$$

の最大固有値を最小化するベクトル変数 x,y を求める問題であり, 以下のように定義される.

$$\Phi(B)=\min\Lambda(x,y),$$

$$\Lambda(x,y)=\overline{l}\{F(x,y)\},(x,y)\in B$$

ここで $\overline{l}\{F(x,y)\}$ は与えられた x,y に対する行列 F(x,y)の最大固有値を, B は x,y の探索空間を表す.

BMI 固有値問題は NP 困難な問題であるが, 分枝限定法を用いて $\Phi(B)$ の上限と下限を ε 近傍に大域収束させることによって, 有限時間内で問題を解く手法が提案されている[2][3].

2.2 マスタワーカ方式による並列解法

2,1 節で述べた分枝限定法による BMI 固有値問題の解法を PC クラスタや Grid 計算環境上で高速に実行するために, マスタワーカ方式を用いた並列解法が提案されている[1]. 本手法では, 一つのマスタと複数のワーカが以下の手順で分枝限定法を実行する.

i. マスタはワーカに初期データ及び探索ツリーの子問題に相当する計算ノードを渡し, ワーカは分枝操作により複数の計算ノードを作りマスタに返す.
ii. マスタは受け取った計算ノードの下界値と, そのときの最小上界値である暫定値から探索不要な計算ノードを取り除く限定操作を行う.
iii. 残った計算ノードの中から最小下界値をもつ計算ノードを選択し, 現在計算をしていないワーカに送る.
iv. ワーカは, 受け取った計算ノードに対し分枝操作行う. 次に, 新たに生成された計算ノードの上界値と下界値を計算し, 限定操作を行った上で残った計算ノードをマスタに返す.

以降, ii. iii. iv. を暫定値が大域収束するまで繰り返す.

2.3 マスタワーカ方式の問題点

2.2 節のマスタワーカ方式では, 複数のワーカに対してマスタは一つである. そのため, Grid 計算環境下において多くのワーカを確保できたとしても, それらを統括する唯一のマスタが処理全体のボトルネックとなってしまい, ワーカ数が多い場合に性能が低下する可能性がある. また, Grid 計算環境下においては, マスタとワーカが広域ネットワーク（WAN）により接続されるため, 実行環境によっては, ネットワークのバンド幅の制限やネットワークの混雑具合が性能を大きく低下させる要因となる. さらに, PC クラスタ等の計算資源はプライベートネットワーク内に存在することが多い. このような計算資源を利用するためには, SSH のポートフォワーディングを用いる等何らかの通信の中継を行う必要があるが, このような処理による通信性能の低下も全体の性能低下の要素となる.

3. 階層的マスタワーカ方式

2.3 節で述べた問題を解決する方法として, 本稿では従来のマスタワーカ方式に階層化を施し, 新たに図 2 に示すようなスーパーバイザ,

表 1 Grid 計算環境

	CPU	Mem	OS	台数
東工大クライアントマシン（横浜市）	Pentium4(1.9GHz)	512MB	Linux	1
東工大 PC クラスタ（横浜市）	Pentium3(1GHz) Dual	1GB	Linux	16
京大 PC クラスタ（京都市）	Pentium3(650MHz)	320MB	Linux	1
	Athron(1.3GHz)	768MB	Linux	13
	AthronMP(1.2GHz) Dual	1GB	Linux	6
東工大 Titech Grid クラスタ（目黒区）	Pentium3(1GHz)	512MB	Linux	2
	Pentium3(1.4GHz) Dual	512MB	Linux	192

マスタ, ワーカを定義することによって複雑な Grid 計算環境下に柔軟に対応する手法を提案する. 従来の一つのマスタが行っていた処理を複数のマスタが分散して行う. また, その際にマスタを計算資源が設置されているサイト内のネットワークに配置することによって, ワーカ群とのネットワーク的な距離を最小限に抑え, SSH ポートフォワーディング等を介さない高速な通信を可能にする.

3.1 実行手順

階層的マスタワーカ方式ではワーカは従来のワーカに等しく, 与えられた計算ノードに対する分枝操作とそれに伴う探索空間の上界値と下界値の計算及び限定操作を行う. マスタは複数台設定することができ, 従来の単独のマスタ処理を分散して行う. 各々は複数のワーカを統括し, それぞれが独立に計算ノードの管理, 及び計算ノードに対する限定操作と選択操作を行う. 最後に, スーパーバイザは各マスタが持つ計算結果（暫定解や最小下界値等）を定期的に（例えば 2 秒間隔で）集計し, その集計結果を再び各マスタに配布することにより, 計算の促進とマスタ間の同期を取る.

全体的な計算手順を以下に示す.

i. スーパーバイザは初期データを各マスタに送る.
ii. 一番目のマスタはワーカ上での分枝操作により利用可能な計算資源の数に見合った計算ノードを生成する.
iii. スーパーバイザは一番目のマスタから複数の計算ノードを受け取り, 他のマスタに分配する.
iv. 各マスタは受け取った計算ノードに対し 2.2 節の動作 ii iii iv を繰り返す.

このようにマスタは iv を行う一方で, 上記のようなスーパーバイザからの要求にも応える

3.2 負荷分散

3.1 節の方法において, スーパーバイザが単純に計算ノードを各マスタに割り振るだけでは, あるマスタは同時処理可能な量以上の計算ノードを持て余し, 他方では計算ノードが限定操作によって枝刈りされ, アイドリングしているといった状況に陥る. それでは用意した計算資源分の並列計算性能を得ることは難しい. そこで提案手法ではスーパーバイザが, 定期的にマスタと通信を行い, 各マスタ上の待ちキュー内で計算待ちの計算ノード数を得て, アイドリングに近いマスタへ, 計算ノードを持て余しているマスタから計算ノードを適宜移動させる. これによりマスタ間の負荷分散を実現する.

実際には, その待ちキューに持つ計算ノードの数が, 閾値を下回ったマスタが存在する場合, スーパーバイザは他のマスタに計算ノードを要求する. 要求されたマスタは, ある条件を満たす *Nodes* 個の計算ノードをスーパーバイザに渡す.

提案手法ではこの条件を変化させることによって, 計算ノードを持て余しているマスタは積極的に計算ノードを他に回すかどうかを調整できる. 積極的に回さないように, 例えば待ちキューの半分以上は回さないように設定すると, 他のマスタがアイドリングに近い状況にあるにも関わらず, あるマスタには多くの計算ノードが待ちキューに残っているという状況になり, 全体としての性能は落ちる. 一方で, この条件を待ちキューにあるだけ積極的に回すように変更すると, 計算ノードが不必要にマスタ間を移動し, その間に各マスタが連鎖的なアイドリングを起こしてしまう. そこでそれらの折衷案として本稿では以下のような条件を採用し, 比較した結果良好な負荷分散性能を得ている.

$$Nodes < (qsize + nServer) * (mServer - 1) / mServer$$

qsize:そのマスタがもつ待ちキューに残っている計算ノードの数
nServer:そのマスタが統括するワーカの数
mServer マスタの数

4. 性能評価

本節では,提案手法を Ninf を用いて実装し, Grid 実験環境上で実行した結果について述べる.

Ninf は Grid 計算環境上で GridRPC を実現するミドルウェアの一つである[4][5]. Ninf では NinfExecutable と言われる実行プログラムを Ninf サーバ上に登録しておけば, クライアントからの NinfCall によってプログラムが起動され, 実行及び計算結果をクライアントに返すという一連の処理を容易に実現できる.

4.1 実験環境

本実験で用いる実験環境は, 東京工業大学すずかけ台キャンパス（神奈川県横浜市）に設置されたクライアントマシンと PC クラスタ, 京都大学に設置された PC クラスタ, 及び東京工業大学学術国際情報センター（東京都目黒区）に設置された Titech Grid クラスタである. 表１に各マシンの仕様を示す. ただし, 今回使用したクラスタ群は, 利用に関してスケジュール管理を行っておらず, 他のユーザもクラスタを頻繁に利用しているような状況下における実行結果である. 時間帯や状況によっては同じ条件での実行であっても数十秒単位の実行時間差が生じる. よって, この実行結果は本来の Grid 環境下に近いデータであるといえる.

また, 性能評価に用いた問題はヘリコプターの旋回動作制御問題[6]と, 一様な乱数によって双線形行列関数 F(x,y)を生成したランダム問題である.

4.2 階層的マスタワーカ方式による効果

東工大のクライアントマシンと京大の PC クラスタを用いて, 従来のモデルと階層化を施したモデルに同じランダム問題を逐次計算させた. 以下はその実行時間と, 各ポイントにおける実行時間計測から, 実行時間に含まれる通信時間を概算した結果である.

実行時間（通信にかかる時間）
従来モデル … 1807sec（668.4sec）
階層化モデル … 1149sec（ 12.1sec）

従来モデルのマスタとワーカ間の通信時間には大まかに言って, 東工大内の LAN, 東工大から京大までの WAN（SSH ポートフォワーディングを介した）と京大内の LAN によるものが含まれる. 階層化モデルでは, マスタを京大のクラスタ内に置く事によって全体の実行時間に影響する通信を京大内の LAN による通信だけに抑えることができる. 階層化モデルにおけるスーパーバイザとマスタの通信には WAN を介しているが, この通信は計算結果の集計等のためであって計算の本質ではないため, 多少の遅延があっても実行時間に対しほとんど影響を与えない.

表 2 マスタワーカ方式と
階層的マスタワーカ方式の比較

	マスタワーカ方式		階層的 マスタワーカ方式	
Worker 台）	Exectime（sec）	Speed_up	Exectime（sec）	Speed_up
1	1719	1.0	1523	1.0
2	884	1.9	529	2.9
4	444	3.9	264	5.8
6			203	7.5
8	235	7.3	152	10.0
14			102	14.9
16	165	10.4	81	18.8
24	158	10.9		
28	151	11.4		
32			65	23.4
54			51	29.9
*74			36	42.3

この結果から, 従来モデルと階層化モデルは, 含まれる通信時間の差が直接実行時間に影響していることがわかる. よってこの通信時間を最小限に抑えられる階層化は, 大きなメリットがあることを示している.

次に従来のマスタワーカ方式と階層的マスタワーカ方式の比較を行った. ヘリコプター問題（係数行列サイズ 8×8, $n_x = 10, n_y = 2$）と東工大 PC クラスタと京大 PC クラスタを用いて, ワーカの数（Worker）を変化させたときの性能を計測結果を表 2 に示す. Worker の台数は両 PC クラスタから同じ台数のマシンを含む. 例えば 8 台の行は, 東工大 PC クラスタから 4 台と, 京大 PC クラスタからの 4 台それぞれをワーカとして実行した結果である. また, 階層的マスタワーカ方式においてはワーカと別に, マスタを各 PC クラスタ内に一台ずつ配置している. どちらの方式も台数が増えるに

従って比較的スムーズに性能が伸びている. しかし，マスタワーカ方式はワーカ台数が 16 台に近づくあたりからワーカとの通信時間がボトルネックとなり始め，性能向上が飽和する. そして，32 台になると正しい計算結果が得られなくなる．この原因は，マスタとワーカを SSH のポートフォワーディングを用いていることで, 単位時間当たりの通信量が多くなりすぎてバッファあふれを起こし, ワーカから正しい計算結果を得られなくなるからと考えられる. 一方, 階層的マスタワーカ方式は最大ワーカ 54 台で 29.9 倍の速度向上が見られた. ヘリコプター問題はローカルな環境でのマスタワーカ方式では，速度向上の限界が 27 倍前後にあることがわかっているため, 階層化によるマスタ処理の多重化と, マスタとワーカ間の通信時間削減が実際に性能を向上させていることがわかる．参考として，東工大（横浜市）にもう一つマスタを置き，20 台のワーカを追加した 74 台環境下では 42 倍の速度向上が確認されている.

4.3 階層的マスタワーカ方式のスケーラビリティ

次に階層的マスタワーカ方式を, 更に大きなランダム問題（係数行列サイズ 24×24, $n_x=6, n_y=6$）を用意し，マスタ（及びワーカ）の数が増えた環境における性能を評価する. 表 3 にはそれぞれのマスタ台数に対する実行結果と, その際の各マスタのアイドリング時間が示してある．特に示してない行は東工大 Titech Grid クラスタ内に指定台数のマスタが置かれ，それぞれが 16 台づつのワーカを統括する．ここで言うアイドリング時間というのは, マスタ上における分枝限定法の枝刈り処理の結果, 待ちキュー及び統括するワーカ上に計算ノードが無くなって他のマスタからノードを回してもらうのを待っている時間の合計である. つまりこの時間が少ないほどマスタ間の負荷分散がうまくいっていることを示す.

実行時間を見るとマスタ（及びワーカ）が増えるに従い実行時間は改善され, マスタ数の増加に伴う性能低下といった傾向は特に見られない. アイドリング時間に関して今回用いた問題においては，計算を開始してから 40 秒弱の間は, 分枝限定法の限定操作が効果的に働くことによって計算ノードの数が大きくならず, 一番目のマスタ（と統括するワーカ間）だけで計算が行われ, 他のマスタにまわす計算ノードが無い状態である. よって二番目以降のマスタのアイドリング時間は比較的大きい. これを差し引いた時間が実際に負荷分散アルゴリズムの性能によって影響される部分である. それを考慮すると本稿の負荷分散アルゴリズムは比較的効果的に働いていることが確認できる. ただし, 必ずしもアイドリング時間の削減が実行時間の減少につながるとは限らない.

表 3 階層的マスタワーカ方式のスケーラビリティ

Master（台）	Worker（台）	Exectime（sec）	Speed_up	Idling（sec）
1	1	15418	1.0	
1	16	1013	15.2	
2	32	615	25.1	(2 35)
3	48	456	33.8	(5 38 36)
4	64	375	41.1	(2 39 40 35)
5(*1)	80	337	45.8	(6 43 42 41 37)
6(*2)	96	293	52.6	(0 41 41 45 43 65)

*1 東工大 PC クラスタ内に一台
*2 東工大 PC クラスタと京大 PC クラスタ内に一台づつ

4.4 マスタ／ワーカの構成

最後にマスタとワーカの数を様々な台数に設定した際の性能を表 4 に示す. 表の下へ行くほどワーカの台数とマスタの数が増えるように並んでおり, それぞれ実行時間も改善されている. 同じワーカ台数の行を比較するとほぼ同等の性能を示している. マスタの数が多いほうがマスタの処理が分散されるが, 負荷の偏りが影響し性能が相殺しているからと考えられる. また, 各マスタの統括するワーカ数を多少不均一にした環境を再現すべく用意したマスタ 5 台の 3 つの行を見ても, 同じワーカ台数の行と比べて実行時間は同等で, 台数に対する性能が確保されている.

最後のマスタ 4 台，各ワーカ 64 台づつの環境では逆に性能が落ちているが, これは各マスタが受け持つワーカが多いため, 従来のマスタワーカ方式のマスタのようにそれぞれのマスタのノード管理や通信処理が重くなり, ボトルネックとなっていること, 及びワーカ数が多いためにワーカ間の暫定値更新が遅れるためと考えられる．これは今後，負荷状況に応じてマスタが使用するワーカ数を調整するアルゴリズムを組み込むことで性能低下を回避できる可能性がある.

表 4 マスタ／ワーカの構成と性能

Master（台）	Worker の所属（各台）	Worker 数（台）	Exec_time（sec）	Speed_up	Idling（sec）
1	c*1	1	15418	1.0	
2	c*32 c*32	64	357	43.2	(0 37)
4	c*16 c*16 c*16 c*16	64	375	41.1	(1 47 53 48)
5	c*32 c*8 c*8 c*8 dc*8	64	362	42.6	(2 43 42 47 49)
3	c*32 c*32 c*32	96	271	56.9	(4 35 35)
5	c*64 c*8 c*8 c*8 c*8	96	278	55.4	(0 53 45 47 49)
2	c*64 c*64	128	217	71.1	(0 38)
4	c*32 c*32 c*32 c*32	128	210	73.4	(0 40 43 41)
5	c*64 c*16 c*16 c*16 c*16	128	224	68.8	(0 42 48 42 43)
3	c*48 c*48 c*48	144	206	74.8	(0 41 44)
3	c*64 c*64 c*64	192	181	85.2	(0 41 41)
4	c*48 c*48 c*48 c*48	192	179	86.1	(0 44 44 51)
4	c*64 c*64 c*64 c*64	256	233	66.2	(0 50 60 58)

*Worker の所属の列における一つ一つの項はマスタの統括するワーカの台数とマシンの所属を示す.
c…Titech Grid クラスタ　dc…東工大 PC クラスタ

5. まとめ

本稿では, 従来の単純なマスタ方式に階層化を施すことにより, 唯一のマスタによるボトルネック, 及び通信遅延による２つの問題を取り除く階層的マスタワーカ方式を提案した. また, 提案手法を実際の Grid 実験環境下に実装したところ, 他のユーザが頻繁に利用している環境でも良好な性能向上を実現することができ, 提案手法の有効性が確認された.

今後著者らは, 計算環境, 特にネットワークやマシンの負荷状態等に応じて, ワーカの数を自動的に調整する仕組みや, 静的に定められている各種パラメタや条件式(マスタ間の負荷分散アルゴリズム等)を動的に変化させる仕組みを取り入れることにより, 複雑な Grid 計算環境に柔軟に対応していく予定である.

謝辞

本研究を進めるにあたって, PC クラスタの利用に関してご協力頂いた京都大学の藤沢克樹助手, また Titech Grid クラスタ上での性能評価にご協力頂いた東京工業大学学術国際情報センターに感謝いたします.

参考文献

[1]合田 憲人, 二方 克昌, 原 辰次, “並列分散計算システム上での BMI 固有値問題解法“, 情報処理学会論文誌: ハイパフォーマンスコンピューティングシステム, Vol.42 No.SIG1 2 (HPS4) Nov.2001

[2]H.Fujioka and K.Hoshijima : Bounds for the BMI Eigenvalue Problem, 計測自動制御学会論文集 vol.33, No7, pp.616-621(1997)

[3]Fukuda,M. and Kojima.M.:Branch-and-cut algorithms for the bilinear matrix inquality eigenvalue problem, Computational Optimization and Applications, Vol.19, No.1.pp.79-105(2001)

[4]M.Sato, H.Nakada, S.Sekiguchi, S.Matsuoka, U.Nagashima, and H.Takagi : Ninf : Network based Information Library for a Global World-Wide Computing Infrastructure, Proc. Of HPCN '97(LNCS-1225), pp.491-502(1997)

[5]Ninf: Network Infractracture for Global Computing
http://ninf.apgrid.org

[6]Keel, L.H., Bhattachayya, S.P. and Howze, J.W.: Robust control with structured perturbations, IEEE Trans. Auto. Contr., Vol.33, No.1, pp.68-78(1988)