OR木探索向けの粒度自動調整型動的負荷分散方式の試作と評価

川村 正祥† 中島 克人†
†東京電機大学大学院 工学研究科

1. はじめに

組合せ最適化問題の解を並列に探索する場合，探索空間を探索木で表現すると，探索部分木をジョブとしてプロセッサマッピングする方式が一般的と言える．しかし探索部分木のそれぞれの大きさが著しく異なる様なアンバランスツリーの場合，探索部分木を負荷分散の単位(以下，粒度)として静的にプロセッサマッピングしても負荷の均等化は困難になる．そうした背景から多くの負荷分散手法が提案されてきた[1,2,3]が，問題の規模や計算機環境を固定した上での負荷分散手法が多く，汎用性は高くなかった．

そこで問題の規模や計算機環境に依存せずにプロセッサマッピング時の粒度を動的に決定する，汎用性の高い指標を提案する．そして提案指標を組込んだ粒度自動調整機構を OR 並列の探索が可能なナンバープレース問題に適用したので，その評価結果を報告する．

2. 粒度自動調整の指標

2.1 並列処理の基本方針

探索部分木の生成と分配など負荷分散の制御と実装が容易であるマスター・ワーカモデルを用いる．マスター・ワーカモデルは一台のマスタープロセッサがジョブを生成し，複数台のワーカプロセッサからの要求に応じてジョブを分配する動的負荷分散方式である．マスタープロセッサは，ジョブとなる探索部分木を，大元の探索木の根からある深さ(以下，レベル)まで幅優先探索することで作成し，ジョブキューにそれらを一旦貯めた後，ジョブ要求してきたワーカプロセッサに分配する．ワーカプロセッサからの要求がない間は，自らもジョブを担当する．

2.2 提案指標

粒度が小さ過ぎると通信や同期のための負荷分散オーバヘッドが増大してプロセッサ稼働率が下がり，粒度が大き過ぎると全体の計算終了時にアイドルプロセッサが生じることにより，やはりプロセッサ稼働率の低下を招く．それゆえ高いプロセッサ稼働率を維持するために粒度を適度の大きさに調整すべきであるが，探索木がアンバランスツリーであると，ジョブとする部分木の探索時間，即ち，粒度の事前予想は困難である．そこで，それまでの実績，すなわち，それまでの部分木探索時間と負荷分散時の通信時間の割合(以下，オーバヘッド率)，およびジョブキュー内の部分木数を基に粒度自動調整を行う方法を提案する．粒度調整の際の指標は次のようにする．

○粒度を大きくする際の指標
探索部分木数がプロセッサ台数より多く，オーバヘッド率が一定の割合(C_L[%])を超えた時

○粒度を小さくする際の指標
探索部分木数がプロセッサ台数以下で，オーバヘッド率が一定の割合(C_S[%])以下の時

C_L と C_S の値は，得たいプロセッサ稼働率の目安としてユーザが指定する．なお，ここでの粒度調整とは，兄弟関係にある未探索の部分木を分岐前の親のレベルにまとめたり，それぞれを子のレベルに分解したりすることである．

A Dynamic Load Balancing Scheme with Automatic Grain-size Control for OR-tree Search Problem and Its Evaluation
Masayoshi Kawamura†, Katsuto Nakajima†
†Graduate School of Engineering, Tokyo Denki University

2.3 粒度の調整量

探索部分木のレベルをユーザが期待するプロセッサ稼働率になるまで一度に調整する方式が考えられるが，探索時間は，部分木の大きさのみならず，探索木の場所にも深さにも依存する可能性があるため，過度の調整は再調整を招く場合がある．そこで一度に調整するレベル数は平均分岐数を参考に小さめの固定値(n)にしておく．

3. 粒度自動調整機構の実装と評価

3.1 評価対象

評価対象としてナンバープレース問題を選択した．これは，M 行×M 列の方眼上の幾つかのマスに予め数字が配置された「問題」に対して，下記の 3 つの制約条件を下に空マスに数字を配置するパズルである．

- 同じ列に 1~M の数字を重複なしで配置
- 同じ行に 1~M の数字を重複なしで配置
- m×m のブロック内に 1~M の数字を重複なしで配置(ただし m は M の平方根とする)

今回は表 1 に示した複数解を持つ問題を用いた．本問題における解探索では，数字を 1 つ配置することが探索木上で探索を 1 レベル深めることに相当する．ここで，ある空マスに対して配置できる数字が複数ある場合は，それらの数字の配置後の部分探索空間は互いに独立であり OR 並列の探索が可能になる．なお，数字を配置したマスが増えるにつれ，すなわち，探索木のレベルを深めるにつれて探索部分木数は増大する．しかし配置できる数字は減少し，最後の空マスに着手する前に配置できる数字が無くなる場合，すなわち，探索に失敗する場合が増大する．

表 1　ナンバープレース問題の性質

	問題 1	問題 2	問題 3
問題サイズ	9×9	9×9	9×9
空マスの個数	62	58	61
解の個数	88	7660	12826
全探索ノード数	9,688,532	4,551,288	7,153,741
探索木の平均分岐数	1.31	1.38	1.40

3.2 粒度自動調整の実装

問題の探索木の平均分岐数は 1.3～1.4（表 1）と大きくはないが，粒度の変り過ぎを避けるため，今回は 1 回の調整レベル数 n=1 とする．

粒度を大きくするときは，ジョブキュー内で最も深いレベルの部分木のみを対象とし，それらの兄弟関係を調べ，兄弟全部が揃っている場合に限り，それらをそれらの親の部分木にまとめる．これを可能にするために，各部分木はレベル値に加えて「先祖リスト」を保持している．先祖リストには，その部分木の親の ID や，自分が何人兄弟の何番目かという情報が含まれている．

粒度を小さくするときは，ジョブキュー内の探索部分木の全てを 1 つ下のレベルに分解する．単純な方法を用いているのはジョブ数不足を早く解消するためである．

粒度自動調整に至る手順は次のとおりである．

(a) ワーカプロセッサは，マネージャプロセッサからプロセッサマッピングされた探索部分木の解探索を行い，見つけた解は，その都度マネージャプロセッサに報告する．解探索が終了したら探索に費やした時間とジョ

ブを要求してから受け取るまでの通信時間をマネージャプロセッサに報告する

(b) マネージャプロセッサは各ワーカプロセッサから受け取った探索時間と通信時間のそれぞれの合計からオーバヘッド率を計算する

(c-1)マネージャプロセッサはジョブキュー内の探索部分木数がプロセッサ台数より多く，かつオーバヘッド率が C_L[%]以上ならば粒度を大きくする

(c-2)マネージャプロセッサはジョブキュー内の探索部分木数がプロセッサ台数以下，かつオーバヘッド率が C_S[%]以下ならば粒度を小さくする

3.3 評価結果

表1の問題を逐次探索，粒度調整なしの並列探索，粒度調整ありの並列探索の3パターンで評価した．評価環境は，表2に示したPCクラスタで1，2，4，8台のプロセッサを使用した．なお各問題の逐次探索の実行時間は，問題1，2，3のそれぞれに対し約285秒，約139秒，約226秒であった．

表2 評価環境

プロセッサ / メモリ	Intel Celeron 2.6GHz / 256MByte
OS / 通信ライブラリ	Red Hat Linux9 / MPICH 1.2.5
ネットワーク	Gigabit Ethernet

表3に逐次探索の実行時間に対する台数効果と粒度調整の回数を示す．表中の L は粒度を大きくした回数，S は小さくした回数，B はマネージャプロセッサが初期生成した探索部分木数である．

表3から，各問題共，粒度調整の有無に関わらずほぼ線形の台数効果が得られているが，問題2の8台並列探索を除いて，粒度調整を行った方がより良い台数効果が得られたことが分かる．粒度を大きくした回数は事前の予想に反し，各問題共 1~2 回程度と余り多くない．粒度を小さくした回数はプロセッサ台数の増加につれて多くなっているので，探索終盤でのアイドルプロセッサの発生を抑止できていることが読み取れる．なお，問題3-2の8台並列でのスーパーリニアの台数効果はキャッシュメモリの増大効果によるものと考えられる．ちなみに，マネージャプロセッサが粒度調整に費やした時間はマネージャプロセッサの実行時間の約0.17%であった．

3.4 考察

表4および表5は，8台並列でのプロセッサの稼働率と粒度調整時の探索木のレベルおよび平均分岐数である．

表4より，問題1，2では粒度調整による稼働率の向上が見られるが，問題3-1，3-2ではわずかな低下が観察される．これはマネージャプロセッサが粒度調整のために探索に割ける時間が減少したためと考えられる．

表5より，粒度を大きくしたときの分岐数は全体の平均と同等かやや大きく，粒度を小さくしたときの分岐数は平均で1前後しかないことから，この問題の探索木は総じて横に狭く縦に長かったことが分かる．また，粒度を小さくするために探索木のレベルを1つ下げてもジョブキュー内のジョブが増えず，このレベル下げがマネージャプロセッサ内で繰り返されている．問題2では，粒度調整によって稼働率が向上してはいる(表4)ものの，台数効果が悪化した(表3)．これは，マネージャプロセッサでの頻繁な粒度調整のために，全体として探索効率（データ参照の局所性など）が低下したためと考えられる．

表3 逐次探索の実行時間に対する台数効果と粒度調整の回数

	問題1				問題2			
	C_L：10%, C_S：10%, 初期生成部分木数 B：3000				C_L：10%, C_S：10%, 初期生成部分木数 B：3000			
台数	あり(*)	L	S	なし(*)	あり(*)	L	S	なし(*)
1	1			1	1			1
2	1.83	1	0	1.81	1.84	1	0	1.84
4	3.76	2	2	3.74	3.78	1	3	3.69
8	7.67	2	7	7.60	7.61	1	3	7.64

	問題3-1				問題3-2			
	C_L：10%, C_S：10%, 初期生成部分木数 B：3000				C_L：5%, C_S：10%, 初期生成部分木数 B：1500			
台数	あり(*)	L	S	なし(*)	あり(*)	L	S	なし(*)
1	1			1	1			1
2	1.92	1	0	1.92	1.92	1	0	1.92
4	3.95	1	3	3.91	3.97	1	2	3.96
8	7.96	1	12	7.90	8.10	1	7	8.02

L／S：粒度を大きく／小さくした回数

表4 8台並列でのプロセッサの稼働率

	問題1	問題2	問題3-1	問題3-2
あり(*)[%]	94.30	92.30	94.07(52.75)	93.91(51.33)
なし(*)[%]	94.19	91.75	94.22(54.09)	93.91(51.36)

(*)：粒度調整の有無，( )内はマネージャプロセッサの稼働率を示す

表5 8台並列での粒度調整時の探索木のレベルと平均分岐数

		問題1	問題2	問題3-1	問題3-2
粒度を大きくしたとき	探索木のレベル	8~9	6	15	10
	平均分岐数	1.25	1.72	1.41	1.59
粒度を小さくしたとき	探索木のレベル	8~14	5~7	14~25	9~15
	平均分岐数	1.16	0.74	1.04	0.90

4. 結論と今後の課題

結論として，提案指標による粒度調整は台数効果や稼働率に関して効果があり，マネージャプロセッサの負荷もそれほど大きくないことが確認できた．しかし，今回の問題は探索木の分岐数が小さいことから，粒度を大きくする調整はあまり効果を発揮できず，小さくする調整は何度も繰り返すことによりマネージャプロセッサに負担をかけ，総じて自動調整の効果は限定的であった．

以上のことから，粒度の調整量の検討が今後の大きな課題である．今回の評価では調整量を探索木の1レベルに固定したが，動的に調整量を決定するのが理想的である．これは，粒度調整時の部分木の分岐数を計量することにより実現可能である．ただし，ジョブとなる部分木の深さ，つまり，ジョブの実行時間も考慮に入れて調整量を判断しなくてはならない．

今回の評価に使用した問題は平均分岐数が予想以上に小さかった．今後はもう少し横広のものを含め，色々な探索木形状を持つ問題で評価と検討を行う予定である．

参考文献

[1] 髙木 允，田村 慶一，周藤 俊秀，北上 始，“並列 Modified PrefixSpan 法における動的負荷分散手法”，情報処理学会研究報告-数理モデル化と問題解決，pp.9~15，2004年6月

[2] 古市 昌一，瀧 和男，市吉 信行，“疎結合並列計算機上での OR 並列問題に適した動的負荷分散方式とその評価”，情報処理学会研究報告-計算機アーキテクチャ，pp.73~81，1989年11月

[3] 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(トランザクション)ハイパフォーマンスコンピューティングシステム，pp.132~141，2001年11月