ゲーム木の並列探索のための分散共有ハッシュ機構の設計と実現

長島紀子[†,☆] 中山泰一[†] 野下浩平[†]

ゲーム木を探索する場合，異なる枝に同一局面が発生することが頻繁に起きる．この探索の重複を避けるため，局面表（ハッシュ表）に局面を登録し，計算結果を再利用するという手法がよく用いられる．ゲーム木の並列探索するときにも，上記のハッシュ表をプロセッサ間で共有できれば，異なるプロセッサ上で計算した結果が利用でき，計算時間の短縮が期待できる．本研究では，まず，分散メモリ型並列計算機（NEC Cenju-3）上に分散共有ハッシュ機構を設計・実現した．続いて，具体的なゲーム木探索問題としてオセロゲームの先手必勝後手必勝の決定問題を取り上げ，分散共有ハッシュ機構の評価実験を行った．実験の結果，たとえば 7×5 盤のオセロゲームでは，分散共有ハッシュ機構の導入により約 30% 計算時間が短縮されることが示された．

Design and Implementation of a Distributed Shared Hashing Mechanism for Searching Game-Trees in Parallel

NORIKO NAGASHIMA,[†,☆] YASUICHI NAKAYAMA[†] and KOHEI NOSHITA[†]

This paper proposes a distributed shared hashing mechanism for searching game-trees in parallel on distributed-memory computers. If a parallel program for solving game problems uses hash (transposition) tables shared among all processors and if the overhead caused by interprocessor communications is relatively small, then its computation time can be reduced. We have designed and implemented a distributed shared hashing mechanism on the distributed-memory multiprocessor 'NEC Cenju-3', and have solved $M \times N$ Othello (Reversi) problems. The experimental results show that the computation time has been considerably reduced when our hashing mechanism is employed; e.g. for 7×5 Othello problem, it has been reduced by 30%.

1. はじめに

チェス，将棋，囲碁などのゲーム木を探索する場合に並列処理を行うことが注目されている．先日人間のチャンピオンに勝ち越したチェスプログラム DeepBlue においても，専用の並列処理ハードウェアを用いて並列計算を行っている．また，筆者らはこれまでに，ネットワークにより結合された UNIX ワークステーション群を利用して，具体的なゲーム木として詰将棋問題を並列に探索し，並列計算による性能向上を示してきた[7),8)]．

ゲーム木を探索する場合，異なる枝に同一局面が発生することが，頻繁に起きる．この探索の重複を避けるために，局面表（transposition table）に局面を登録し，のちに同一局面が発生したときに，局面表の結果を再利用するという手法がよく用いられる．局面表はハッシュ表を用いて実現する．

本研究では，ゲーム木の並列探索に，上記のハッシュ表をプロセッサ間で共有することに着目した．ハッシュ表の共有により，異なるプロセッサ上で計算した結果を利用でき，計算時間の短縮を期待できる．

そこで，まず，分散メモリ型並列計算機上に，ハッシュ表をプロセッサ間で共有するための機構（分散共有ハッシュ機構）を設計・実現した．分散メモリ型並列計算機において，プロセッサ間でデータを共有し，あらゆる並列プログラムを正しく動作させるためには，厳密なデータの一貫性の制御が求められ，大きなオーバヘッドがかかる．これに対し，ゲーム木の探索におけるハッシュ表に登録するデータの特徴より，ゲーム木の並列探索は，緩いデータの一貫性で済ませることができ，小さなオーバヘッドで分散共有ハッシュ表を導入できる．

続いて，具体的なゲーム木探索問題として，オセロ

† 電気通信大学電気通信学部情報工学科
Department of Computer Science, The University of Electro-Communications
☆ 現在，日本電気株式会社第三コンピュータソフトウェア事業部
Presently with NEC Corporation

ゲームの先手必勝後手必勝の決定問題を取り上げ，この分散共有ハッシュ機構を評価する実験を行った．この実験の結果，分散共有ハッシュ機構を導入することにより，導入しない場合と比較して，大きな性能向上を得た．たとえば，7×5 盤のオセロゲームの先手必勝後手必勝の決定問題においては，導入しない場合には18時間要する実行時間が，導入した場合には13時間となり，約 30% 実行時間の短縮となった．

2. 分散共有ハッシュ機構の提案

ゲーム木において，別の枝に同一局面が発生することは，頻繁に起こる．そこで，ゲーム木の探索では，局面をハッシュ表（局面表）に登録しておき，同一局面に対する再計算を防ぐ方法が，一般的に用いられている．これにより大幅に計算時間の短縮を行うことができる．また，同一局面に限らず類似局面についてもハッシュ表に格納されているか検索を行い，その情報を利用することにより，計算時間の短縮を図ることが可能であると報告されている[4),5)]．

ここで，互いに独立したハッシュ表を備えたプロセッサ $(P_x, P_y, P_z, \ldots)$ が，**図1** のような探索木を並列に探索する場合を考える（ただし，D は同一の部分木）．

ハッシュ表の検索により，各プロセッサ単位では，探索の重複はない．しかし，全体を1つの探索木とすると，プロセッサ間で探索の重複が行われている．

この重複を避けるためには，プロセッサ間でハッシュ表を共有すればよい．これにより，計算時間の短縮につながる可能性がある．

2.1 ハッシュ機構の性能向上の見積り

本分散共有ハッシュ機構は，プロセッサ間通信を用いることにより，プロセッサ間でハッシュ表を共有し，大規模なハッシュ空間を実現している．すなわち，検索，登録要求はプロセッサ間通信を必要とする．よって，「ある局面の結果を得るための時間（試行時間）」は，「プロセッサ間の通信時間（C）」「局面の結果を得るためのゲーム木の探索時間（M）」「ヒット率（α）」の3つの要素から推定することが可能である．実際には C は，通信時間のほか検索時間を含む待ち時間である．分散共有ハッシュ機構を使用した場合の1回の平均試行時間は次のようになる．

$$C + (1-\alpha) \times M$$

分散共有ハッシュ機構を使用しない場合の平均試行時間は M である．したがって，以下の関係を満たす C, M, α の下であれば，本分散共有ハッシュ機構の導入は，性能向上を得るといえる．

$$C + (1-\alpha) \times M < M$$
$$\Leftrightarrow \frac{C}{\alpha} < M$$

たとえば，C, M, α が $C = 5\,\mathrm{msec}$, $M = 100\,\mathrm{msec}$, $\alpha = 0.1$ のとき，ゲーム木の並列探索は分散共有ハッシュ機構を導入することにより性能向上を得ると見積もることができる．

2.2 データの一貫性

汎用の分散メモリ型並列計算機において，多数のプロセッサがメモリを共有するためには，一般的に，厳密な一貫性（sequential consistency）[3)]の保持が必要であり，オーバヘッドはきわめて大きなものとなる．

そこで，本研究では，次にあげるゲーム木探索のためのハッシュ表へ登録するデータの特徴に注目した．

特徴1 ある局面に対する情報は，どのプロセッサが計算しても同一の結果を得る

特徴2 ハッシュ表に，必要な情報がない場合には，改めて計算すれば，どのプロセッサでも正確な結果を得ることができる

この特徴により，ゲーム木の探索に用いられるハッシュ機構は緩いデータの一貫性で済み，オーバヘッドを小さくすることができる．

3. 分散共有ハッシュ機構の構成

分散メモリ型並列計算機上で分散共有ハッシュ機構をシステム・ソフトウェアとして設計，構成する．システムは，ゲーム木の探索をする「ユーザ・プログラム」とプロセッサ間で共有するハッシュ表を提供する「システム・プログラム」から成る．システムの概略を**図2** に示す．

システム・プログラムが，同時に動作している全プロセッサ上にハッシュ表を作成し，システム全体として大規模なハッシュ表を実現している．ユーザ・プログラムとシステム・プログラムはコンテクスト切替えによるオーバヘッドを防ぐため，メモリ領域を共有す

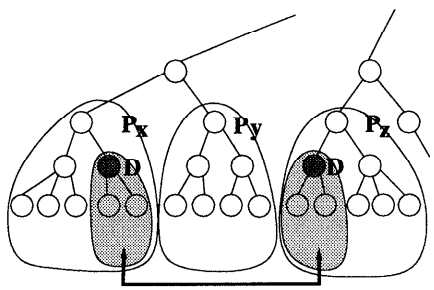


図1 プロセッサ間の同一・類似局面
Fig. 1 Identical (or similar) positions generated by different processors.

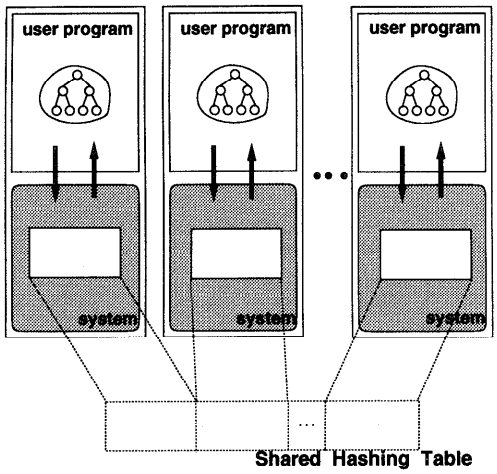


図2 システムの概略
Fig. 2 Basic structure of our distributed shared hashing mechanism.

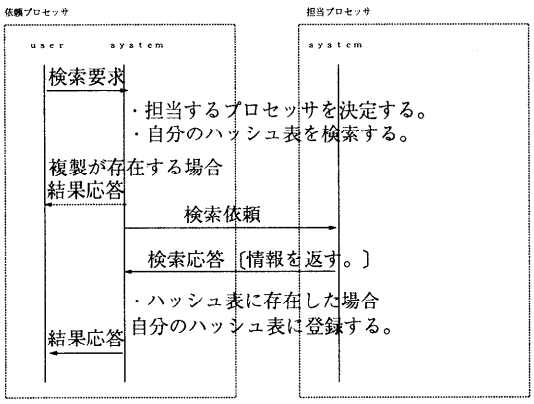


図3 複製データを持つ方式
Fig. 3 Mechanism using the replication of remote hash tables.

る軽量プロセスとして実現し，並行に動作している．

ユーザ・プログラムの要求はシステム・プログラムによって処理される．要求を依頼したプロセッサを「依頼プロセッサ」，実際の処理を担当するプロセッサを「担当プロセッサ」とする．

- 探索要求
  1 ハッシュ・キー（key）から担当するプロセッサを決定する．
  2 担当プロセッサへデータを送信して，「検索」を依頼する．
  3 担当プロセッサからの検索応答を受け，ユーザ・プログラムに結果応答する．
- 登録要求
  1 keyから担当するプロセッサを決定する．
  2 担当プロセッサへデータを送信して，「登録」を依頼する．

  ただし，依頼プロセッサと担当プロセッサが同一の場合は，依頼プロセッサのハッシュ表の検索およびハッシュ表への登録を行うものとする．

この基本的な処理方法により，ハッシュ表の検索時に最新の情報を得ることができる．

実現したシステムでは，システム・プログラムに依頼を出すときにユーザ・プログラムはサスペンドするようにしている．なお，システム・プログラムは，他のプロセッサに依頼を出している間もアイドルにはならず，他のプロセッサからの依頼を受け付けて処理を行う．

4. 性能向上のための方法

前章において述べた処理方法を基本にして，緩い一貫性を保つ，検索および登録要求の処理方法を示す．

4.1 データの複製を保持する方式

他のプロセッサへ要求を依頼する場合には，データの送受信によるオーバヘッドがかかる．

そこで，このオーバヘッドを削減するために，依頼プロセッサの持つハッシュ表に，データの複製を保持する方式を提案する．

すなわち，ハッシュ表を検索するときには，**図3**に示すように，まず依頼プロセッサ自身で複製を保持しているかどうかを調べる．複製データが存在しないときに，はじめて担当プロセッサに依頼を出す．なお，複製用として特に領域の確保はせず，依頼プロセッサのハッシュ表を複製データの格納にも用いることにした．

データの複製を保持する方式により，検索時に依頼プロセッサの持つハッシュ表に複製データが存在するならば，ユーザ・プログラムに速いレスポンスで検索結果を返すことができる．また，2.2節において述べた，ゲーム木探索の特徴より，複製データの無効化の処理，複製データをどのプロセッサ所有しているかの管理をする必要がなく，複製データの一貫性を保持できている．

4.2 タイムアウトを利用する方式

前節の処理方法を採用した場合においても，依頼プロセッサと担当プロセッサが異なる場合には，検索応答時間にはプロセッサ間通信によるオーバヘッドがかかる．

これを防ぐために，「タイムアウト時間」の設定ができるようにする．担当プロセッサへ検索を依頼してか

ら，設定された「タイムアウト時間」を過ぎても担当プロセッサからの検索応答がない場合，担当プロセッサからの応答を待たずに，依頼プロセッサは情報がなかったと判断し，ユーザ・プログラムに「情報が存在しない」と結果応答する．

以上の処理方法を採用する場合においても，情報が存在しないと応答したとしても，前述のとおり各プロセッサが改めてその局面を計算すればよいので，ユーザ・プログラムの動作へ支障を来すことはない．これによりシステム・プログラムの検索応答のかかり過ぎによるユーザ・プログラムのサスペンドを防ぐことができる．

4.3 プライオリティを利用する方式

本分散共有ハッシュ機構を用いてハッシングを行う場合，大規模なハッシュ表を実現しているので，ゲーム木の探索回数を大幅に減少できる．2.1 節において述べたように,「プロセッサ間の通信時間（C）」「局面の結果を得るためのゲーム木の探索時間（M）」「ヒット率（α）」が，

$$\frac{C}{\alpha} < M$$

の関係式を満たすとき，本分散共有ハッシュ機構導入による性能向上が得られると見積もることができる．しかし，M が小さ過ぎると，性能向上を得ることができない．

そこでプライオリティを利用して，M が大きなものだけを共有するようにする．すなわち，M の大きい局面（重要な局面）には高いプライオリティをつけ，共有ハッシュ表を利用する．これに対し，M の小さい局面（重要でない局面）には低いプライオリティをつけ，共有ハッシュ表を利用しないで局面の探索を行う．中間的なプライオリティの局面については，前節で述べた，タイムアウト時間付きで共有ハッシュ表を利用する．

なお，分散共有ハッシュ機構に限らず，一般にハッシュ機構において，ハッシュ表が溢れたとき，登録したデータすべてを消去してしまうと，データがすべて無駄になってしまう．そこで，本分散共有ハッシュ機構では，ハッシュ表が溢れたときに全体を空にするのではなく，プライオリティを利用した GC を行う．プライオリティの高い，利用価値の高いデータをできるだけ残すようにする．

5. ゲーム木による性能向上実験

2 章から 4 章で述べてきた分散共有ハッシュ機構を設計・実現し，分散メモリ型並列計算機上で，評価実験を行った．

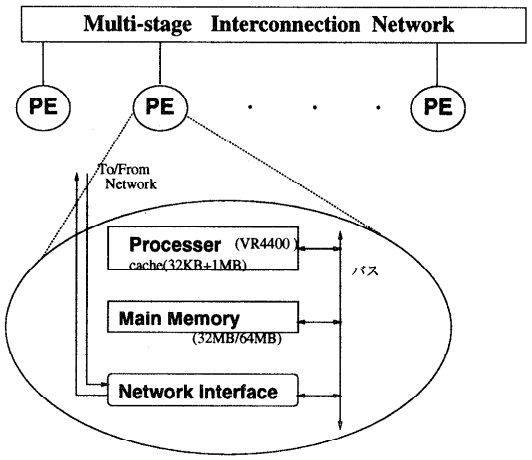


図 4 Cenju-3 のハードウェア構成図
Fig. 4 Hardware configuration of 'NEC Cenju-3'.

5.1 計算環境

分散メモリ型並列計算機 Cenju-3（NEC）を使用した．Cenju-3 は，マイクロプロセッサ V_R4400 を要素プロセッサとし，疎結合型で実装されている．最大接続可能 PE 数は 256 である．ネットワークには多段接続網を使用して，40 MB/sec を実現している[2]．ハードウェアの概略図を**図 4** に示す．

5.2 実験対象

ゲーム木の問題として「$M \times N (M \geq N)$ 盤のオセロゲームの先手必勝後手必勝の決定」を取り上げる．

ゲームの探索木の初期盤面から数手を展開して，各プロセッサに 1 つずつ部分木を割り当てる．これらの部分木を，各プロセッサで α-β 探索（および着手順序付け）を行う．また，各プロセッサでは，16 バイトで表された局面を key にしてハッシュ表にその局面の計算結果を登録し検索することを行う．6 × 5 盤では 25,000 エントリ，7 × 5 盤では 100,000 エントリのハッシュ表を用意した．

表 1 に示す 6 種類の構成についての総実行時間の比較を行った．プライオリティ付けには木の深さを利用した．すなわち，プライオリティを利用する方式では，木の浅いところの局面でのみ共有のハッシュ表を利用している．

5.3 実験結果

各構成（非共有，および，共有 A〜E）における総実行時間を 6 × 5 盤，7 × 5 盤についてそれぞれ**表 2**，**表 3** に示す．

- 6 × 5 盤：PE 数 16 として実行した．非共有構成では 48 分要したのに対して，共有 E 構成では

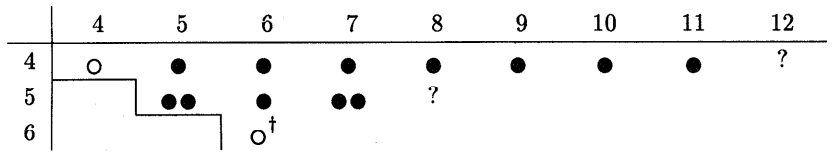


図5 $M \times N$ 盤オセロゲームの勝敗表．●：先手勝ち，○：後手勝ち，†：Feinstein[1]による．なお，奇数 × 奇数盤では2つの初期局面があるため，2つの結果が得られる．

Fig. 5 Results of $M \times N$ Othello problems.

表1 実験に用いた6種類の構成

Table 1 6 types of system configuration used in the experiment.

構成	要求の処理方式
非共有	(プロセッサごとの局所的なハッシュ表は持つ)
共有A	厳密なデータの一貫性制御
共有B	基本方式
共有C	複製を持つ
共有D	プライオリティ
共有E	共有C＋共有D

表2 6×5盤の総実行時間

Table 2 Computation time for 6 × 5 Othello problem.

構成	総実行時間 (sec)
非共有	2872
共有A	3003
共有B	2903
共有C	2857
共有D	2433
共有E	2366

表3 7×5盤の総実行時間

Table 3 Computation time for 7 × 5 Othello problem.

構成	総実行時間 (sec)
非共有	64203
共有A	63379
共有B	61703
共有C	61189
共有D	65638
共有E	47640

39分となり，9分の総実行時間の短縮となり，約20%の性能向上となった．

- 7×5盤：PE数20として実行した．非共有構成は18時間要したのに対し，共有E構成は13時間となり，5時間の総実行時間の短縮となり，約30%の性能向上になった．

ハッシュ表を共有しないハッシュ機構（非共有構成）と比較して，分散共有ハッシュ機構の導入は，優れた性能向上を示した．これより，本分散共有ハッシュ機構は，プロセッサ間の通信によるオーバヘッドの削減を実現し，大規模なハッシュ表を所有する効果を発揮することができるといえる．

さらに，「6×5盤」を対象とした実験と比べ，「7×5盤」を対象にした実験は，「非共有構成と比較して共有E構成の性能向上率」が良いことが分かる．なお，共有E構成におけるハッシュ表のヒット率は，「6×5盤」と「7×5盤」でほぼ同じであった．よって，「7×5盤」では，ハッシュ表を共有したことで，より多くの計算時間を必要とする局面について結果の再利用ができたと考えられる．探索木の規模が大きいほど，本分散共有ハッシュ機構導入により，総実行時間の短縮につながることが予測される．

6. まとめ

本論文では，非常に多くの計算時間を必要とするゲーム木探索問題を並列に解く際に，分散共有ハッシュ機構を導入することを提案した．

具体的なゲーム木の問題として「$M \times N (M \geq N)$盤のオセロゲームの先手必勝後手必勝の決定」について，実並列計算機上で実験を行った．その結果，分散共有ハッシュ機構の導入により，大幅な総実行時間の短縮が確かめられた．また，大規模な探索木になるほど，分散共有ハッシュ機構の性能が発揮できるという見積りが得られた．

また，具体的なゲーム木として，オセロゲームに限らず，チェス，将棋，囲碁など様々なゲームの並列探索においても，分散共有ハッシュ機構の導入により，総実行時間の短縮につながることが期待される．

なお，筆者らが求めた$M \times N$盤のオセロゲームに関する結果は**図5**のとおりである[6]．ただし，6×6盤はFeinsteinの報告による[1]．

謝辞 分散メモリ型並列計算機Cenju-3の利用環境をご提供いただいた，NEC C&Cメディア研究所並列処理センターに感謝いたします．

参考文献

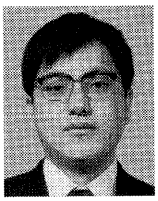
1) Feinstein, J.: Amenor Wins World 6 × 6 Championships!, *British Othello Federation's Newsletters* (1993).

2) 広瀬哲也，細見岳生，丸山 勉，加納 健：並列コンピュータ Cenju-3 のプロセッサ間通信方式とその評価，情報処理学会論文誌，Vol.37, No.7, pp.1378–1387 (1996).
3) Lamport, L.: How to Make a Multiprocessor Computer that Correctly Executes Multiprocess Programs, *IEEE Trans. Comput.*, Vol.C-28, No.9, pp.690–691 (1979).
4) Levy, D.N.L. (Ed.): *Computer Games I*, Springer-Verlag (1988).
5) 松原 仁（編著）：コンピュータ将棋の進歩，共立出版 (1996).
6) 長島紀子：ゲーム木の探索のための分散共有ハッシュ機構，電気通信大学大学院電気通信学研究科情報工学専攻修士論文 (1997).
7) Nakayama, Y., Akazawa, T. and Noshita, K.: A Parallel Algorithm for Solving Hard Tsume-Shogi Problems, *ICCA Journal*, Vol.19, No.2, pp.94–99 (1996).
8) 中山泰一，赤澤忠文，野下浩平：ゲーム木の並列探索のための分散的実行管理機構，電子情報通信学会論文誌，Vol.J79-D-I, No.9, pp.572–575 (1996).

(平成 9 年 10 月 29 日受付)
(平成 10 年 4 月 3 日採録)

赤澤 紀子（正会員）

旧姓長島．1972 年生．1995 年電気通信大学情報工学科卒業．1997 年同大学院電気通信学研究科情報工学専攻博士前期課程修了．1997 年 4 月より，日本電気（株）第三コンピュータソフトウェア事業部勤務．インターネットソフトウェアの開発に従事．



中山 泰一（正会員）

1965 年生．1988 年東京大学工学部計数工学科卒業．1993 年同大学院工学系研究科情報工学専攻博士課程修了．工学博士．1993 年 4 月より，電気通信大学情報工学科助手．現在，同学科講師．オペレーティング・システム，並列・分散処理，ゲーム・プログラミングに関する研究に従事．日本ソフトウェア科学会，電子情報通信学会，IEEE, CSA, ICCA 等会員．

野下 浩平（正会員）

1943 年生．東京大学工学部計数工学科卒業．現職：電気通信大学情報工学科教授．工学博士（東京工業大学）．専門：アルゴリズム解析，組合せゲーム．