多数の遊休PC上での分散ゲーム木探索

金田憲二
東京大学 / PRESTO, JST

1 はじめに

多数の動的に増減する遊休PC上で分散ゲーム木探索を行なうコンピュータ将棋について述べる．

ゲーム木探索は人工知能研究において最も重要なトピックのひとつであり，かつ計算インテンシブな計算でもあるため，並列化・分散化によって性能向上を目指す研究が数多く存在する [2].

並列･分散ゲーム木探索においてより良い性能を得るためには，当然，より多数の計算資源を用いて探索を行うことが望ましい．近年では，多数の計算資源を低コストで得るために，家庭や会社などの遊休PCの利用が注目を浴びてきている（例，SETI@home [3], United Devices [5]）．

そこで，遊休なPCを利用して分散ゲーム木探索を行なうシステムを設計・実装する．このシステムの特徴として，動的にPCが増減する中でも探索を実行し続けることが可能となっている．さらに，そうしたPCの増減を中央集権的な管理なしに行なう．このシステムの性能を最大で720台のノートPCを用いて測定した．最大8倍の性能向上となった．

この発表概要の構成を以下に示す．2節ではシステムの概要について述べる．3節では実装の詳細について述べる．4節では行なった実験について述べる．最後にまとめと今後の課題について述べる．

2 システムの概要

この節では，システムの概要について述べる．まず，我々のシステムが用いる分散ゲーム木探索アルゴリズムについて説明する．次に，動的なPCの追加・削除をどのようにシステムが扱うかについて述べる．

2.1 分散ゲーム木探索アルゴリズム

分散ゲーム木探索とは，ゲーム木を複数のPCが協調しながら探索することである．分散ゲーム木探索における重要な問題のひとつとして，同じ盤面を重複して探索するのをいかに避けるかということがある．ゲーム木が実際にはグラフ構造であるような探索の場合，同じ状態の盤面に複数回到達することは少なくない．そのため，以前の探索した結果を保持する局面表（transposition table）を用意し，一度探索した盤面を再度探索するのを避けることが，性能向上のために重要となる．

ゲーム木探索を分散化する際には，この局面表をどのように複数のPC間で共有・管理するかが問題となる．我々の分散ゲーム木探索は，[2] の提案する手法に基づいている．この手法は，ある盤面を探索するPCが常に同じになるように，探索を各PCへと割り当てる．同じ盤面が重複して複数回現れる場合は，常に同じPC上でその盤面は探索される．よって，個々のPCがローカルに局面表を保持するだけで，盤面の重複探索を回避することができる．

2.2 PCの追加・削除への対処

ゲーム木を探索している最中に，PCを追加・削除することが可能となっている．これにより,「遊休PCのみ計算に参加し，遊休でなくなったPCは計算から脱退する」といったことが可能となる．

計算の参加･脱退は以下のようにして行なわれる．新たにマシン x が計算に参加するとする．このとき，x は既に計算に参加しているPC群にそのことを通知する．その参加通知を受け取ったPCは，x に盤面の探索を割り振るようになる．

また，既に計算に参加しているマシン y が，計算から脱退するとする．y は，まずそのことを他のPC

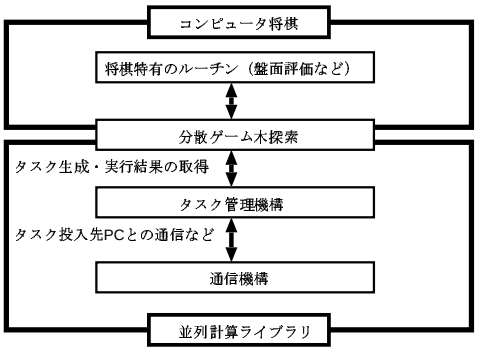

図 1: システムの構成

に通知する．通知を受け取った PC は，それ以降 y に盤面探索を割り振らないようにする．そして，y は自分に割り振られている未探索の盤面が存在する場合，それを他の PC に再割り当てし，脱退処理を完了する．

以上の述べたように，PC の追加・削除は，管理サーバのような中央集権的な機構を必要としない．各々の PC が自律して動作することによって実現される．

3 実装の詳細

この節では，システムの実装について述べる．主に，2.2 節で述べた PC の追加・削除処理をどのように実装するかについて説明する．

3.1 システムの構成

まず，システムの構成について述べる．我々のシステムは，今回は将棋を対象として実装されており，並列計算ライブラリと，コンピュータ将棋部分からなる（図 1 参照）．

並列計算ライブラリは，DAG 型の依存関係 のもつタスクを並列に実行することができる．Phoenix [1, 4] という枠組みを通信機構として利用し，動的な PC の追加・削除を扱う．

コンピュータ将棋部分は，激指 [6] を元にしている．激指の逐次ゲーム木探索ルーチンを，我々の作成した並列計算ライブラリを用いるように変更し，将棋の分散ゲーム木探索を実装している．

以下では，それぞれの機構の詳細について説明する．

3.2 通信機構

この通信機構は，タスクの投入や実行結果の送信の際に用いられるものであり，Phoenix [1, 4] を元としている．Phoenix は，仮想ノード名と呼ばれる論理的な名前を提供する．この仮想ノード名で宛先を指定することにより，PC 間で通信することができる．Phoenix は動的な計算資源の増減に対処するため以下のような特長をもつ：

- 仮想ノード名と物理的な PC との対応を，プログラマが指定することができる．
- その対応を動的に変更することができる．
- 一つの PC に複数の仮想ノードを対応させることができる．

以上の機能を用いて「新たに計算に参加した PC は既に参加している PC から仮想ノード名の一部を委譲してもらい，計算から脱退する PC は自分に割り当てられた仮想ノード名を他の PC に委譲する」ようにすると，動的に PC が増減する中でも，常に一定の仮想ノード名（i.e., 通信先）が存在するようになる．詳しくは後述するが，これにより動的に PC が増減する中での並列タスク実行が可能となる．

[1, 4] で述べられているものとは，以下の点で実装が異なる．今回はローカルサブネット内での動作を想定しているため，各 PC は自分に割り当てられた仮想ノード名を UDP のブロードキャストによって他の全 PC に通知する．基本的には，自分に割り当てられた仮想ノードが変更された際に，新たに割り当てられた仮想ノード集合をブロードキャストする．こうして UDP によるブロードキャストを利用することによって，TCP で全対全にコネクションを生成することによるオーバヘッドを回避できる．

3.3 タスク管理機構

これは，タスクの並列実行のための機構であり，遠隔 PC へのタスクの投入・実行結果の取得などを行う．基本的には，一つのタスクに一つの盤面の探索が対応する．動的に PC が増減する中でも実行を続けられるように，3.2 節で述べた仮想ノード名を以下のように利用している．

まず，各タスクに仮想ノード名を ID として割り当てる．これは探索する盤面の状態から一意に定まるようにする．そして，タスクは自分の ID である仮想ノード名を現在担当している PC へと投入・実行されるようにする．例えば，ID が x であるタスクは，仮想ノード x が現在割り当てられている PC へと投入される．これは，タスクの ID を宛先として通信することによって実現される．同じ盤面には同じ仮想ノード名が割り当てられるため，(仮想ノード名の割り当てが変更されない限り）2.1 節で述べたように同じ PC 上で実行されることが保証される．また，仮想ノード名宛で通信を行なうため，たとえ物理的な PC 数が増減しても，タスクの投入先の PC が存在しないといったこともない．

PC が参加・脱退する時には，3.2 節で述べられたように仮想ノードを委譲することによって，計算に参加している PC のみにタスクが投入されるようになる．脱退する PC がもつ未実行のタスクは，仮想ノード名と共に同時に割り当て先へと委譲される．

3.4 分散ゲーム木探索・将棋特有のルーチン

コンピュータ将棋における枝刈りや盤面評価関数などは，激指のルーチンをそのまま用いている．オリジナルの激指では再帰的に関数を呼び出すことによってゲーム木の探索を行なっている．基本的には，その関数呼び出しを我々のライブラリが提供する並列タスク実行 API 呼び出しに置き換えることによって，分散ゲーム木探索を実現する．

3.5 そのシステムの起動・設定のためのツール群

以上に述べたものの他に，多数の PC を使用する実験を簡便に行えるようにするためのツールを開発した．まず，ソフトウェアのローダを作成した．これは，指定されたファイルをサブネット内の全 PC にコピーする機能と，指定されたコマンドをサブネット内の全 PC で実行する機能をもつ．UDP のブロードキャストを用いて実装される．

また，ユーザレベルで動作可能な分散ファイル共有システムも開発した．`open`，`close` などのシステムコールをトラップし，その引数を変更することによって実現される．

4 実験

この節では，720 台のノート PC を用いて行なった性能測定について述べる．

4.1 実験環境と内容

実験には，720 台の東芝 TECRA（CPU: PentiumM 1.3GHz，Memory: 512MB) を用いた．各 PC は 100Base-T のネットワークで結合されている．

この環境において，将棋の中盤の盤面から深さ 12 と 14 で探索を行ない，探索時間を計測した．

4.2 実験結果と考察

図 2 は実験結果を示している．深さ 12 の時は，最大で 7 倍の性能向上（96 ノード）となった．深さ 14 の時は，最大で 8 倍の性能向上（192 ノード）となった．しかし，オリジナルの逐次の激指と比べてと，4 から 15 倍の性能低下であった．

このオリジナルと比較しての性能低下や性能がスケールしなかったことの原因として，以下のことが考えられる．

分散化によるオーバヘッド 逐次では関数呼び出しによって行なわれていた探索に，分散化によってタスクの生成・通信といった処理が必要となる．特に，タスクの粒度が細かい時には，こうした分散化によるオーバヘッドが大きくなる．

探索盤面数の増加 逐次と比較して分散版では枝狩りの効率が悪くなり，探索盤面数が増加してしまっている（深さ 12 で約 2 倍，深さ 14 で約 7 倍）．

小さい並列度 相互に依存関係のなく並列に探索可能な盤面の最大数を，並列度と定義する．表 1 は深さ 6，8，10 の探索における並列度の値を示している．これにより，PC の台数と比べて並列実行可能な盤面数が少なく，探索がスケールしないことが分かる．

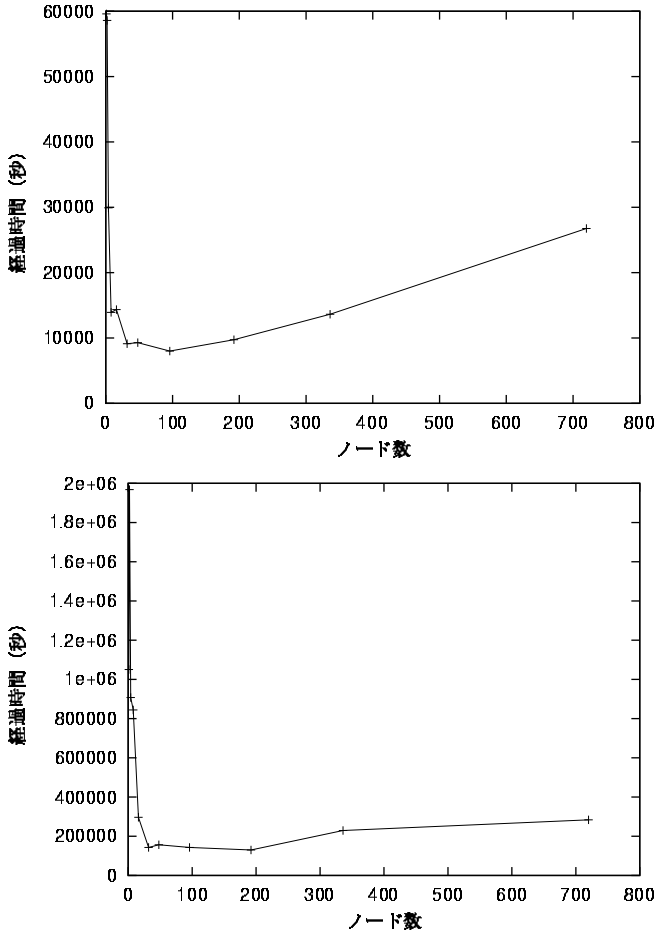

図 2: 実験結果：深さ 12 と深さ 14 における探索時間

表 1: 並列度

探索の深さ	並列度
6	18
8	61
10	162

5 おわりに

多数の動的に増減する遊休 PC 上で分散ゲーム木探索を行なうコンピュータ将棋を設計・実装した．今後の課題としては，まず，動的負荷分散の導入やタスクのスケジューリング方式・粒度の改良による性能向上が挙げられる．次に，耐故障アルゴリズムを考案し，信頼性の向上を目指すことが挙げられる．また，広域環境での動作・性能評価も今後の課題である．

参考文献

[1] Kenji Kaneda, Kenjiro Taura, and Akinori Yonezawa. Routing and Resource Discovery in Phoenix Grid-Enabled Message Passing Library（掲載予定）. In *proceedings of 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, 2004.

[2] Akihiro Kishimoto and Jonathan Schaeffer. Distributed game-tree search using transposition table driven work scheduling. In *proceedings of 31st International Conference on Parallel Processing (ICCP)*, 2002.

[3] SETI@home. http://setiathome.ssl.berkeley.edu/.

[4] Kenjiro Taura, Kenji Kaneda, and Toshio Endo. Phoenix: a Parallel Programming Model for Accommodating Dynamically Joininig/Leaving Resources. In *proceedings of 9th ACM SIGPLAN symposium on Principles and practice of parallel programming (PPoPP)*, pp. 216–229, 2003.

[5] United Devices Inc. ^{TM}. http://www.ud.com/.

[6] 将棋プログラム「激指」. http://www.logos.t.u-tokyo.ac.jp/~gekisashi/.