# 4 大規模クラスタシステムでの実行 ―GPS 将棋の試み―

田中 哲朗
東京大学情報基盤センター

金子 知適
東京大学大学院総合文化研究科

## 大規模クラスタ化の動機

大関や横綱に昇進した力士は直後の場所で活躍できないことが多いとよく言われる．昇進直後はスポンサーなどに呼ばれたり，取材を受けるのに忙しく稽古の暇が取れないことが原因であると言われるが，世界コンピュータ将棋選手権でも同様の傾向があることが知られている．

連続優勝は 2000 ～ 01 年の IS 将棋以降はなく，ここ 5 年は，

2004 年優勝 YSS → 2005 年 4 位
2005 年優勝激指 → 2006 年 5 位
2006 年優勝 Bonanza → 2007 年 4 位
2007 年優勝 YSS → 2008 年 4 位
2008 年優勝激指 → 2009 年 6 位

と優勝ソフトのシード権 (3 位以内) 獲得失敗が続いていた．

力士と違って，コンピュータ将棋の場合は「稽古不足で弱くなる」ことはないが，年々レベルアップする中では「伸びが小さい」だけでもすぐに抜かれてしまうからであろう．

GPS 将棋は昨年の第 19 回世界コンピュータ将棋選手権で初優勝を遂げた．運にも恵まれたが，2003 年の初参加以来，実現確率探索，機械学習による評価関数のパラメータ調整など有望そうな技術を取り入れるとともに独自の工夫も取り入れていった地道な努力が実を結んだ優勝と言えるだろう [1]．

最近難しくなってきた連続優勝を達成すべく，この 1 年間，GPS 将棋に評価関数の精度の向上，詰将棋の並列化などさまざまな改良を加えてきた．しかし，コンピュータ将棋の連続対戦場である Floodgate[☆1] では，GPS 将棋は Bonanza, 激指等の他の強豪プログラムに対しては負け越すことが多かった．

- 比較的遅いマシン（Opteron 248, 2.2GHz）で走らせた逐次プログラムを Floodgate に常駐させている．
- 評価関数の調整などは一手 30 秒の持ち時間で行っているので，持ち時間 1 局 15 分の Floodgate での強さとはマッチしていない可能性がある．

などの事情はあるが，他の強豪プログラムが急激な勢いで進歩していることは明らかだった．

そこで，GPS 将棋の棋力の大幅な向上を実現するために，従来から用いていたマルチスレッドによる並列化に加えて大規模クラスタ環境での並列化を適用することを試みた．本稿では並列化の詳細と，作成したシステムによる世界コンピュータ将棋選手権参加で得られた知見を述べる．

## クラスタ上でのゲーム木探索

主記憶を共有するマルチコアマシン上でスレッドを複数用いて，ゲーム木探索を効率的に行う手法としては PVS（Principal Variation Splitting）[2] およびその改良のさまざまな手法が考案され，コンピュータ将棋の分野でも広く使われるようになっている．

☆1 http://wdoor.c.u-tokyo.ac.jp/shogi/floodgate.html

一方，主記憶を共有しないクラスタ環境のゲーム木探索への適用は遅れている．これは，マルチスレッドによる並列化と比較して，以下のような不利な点があるためと考えられる．

- **通信遅延**
  プロセッサと主記憶上との通信と比較して，別プロセッサ間の通信はスループットが数桁小さいが，通信遅延の方はさらに差が大きい．そのため，全体の性能を落とさないためには，細かい通信ではなく，まとまった単位での通信を行う必要がある．
- **トランスポジションテーブル共有の困難**
  合流がある木探索の場合は，トランスポジションテーブルを共有しないと同じノードを重複して探索する可能性がある．トランスポジションテーブルの参照，更新は高頻度で発生するので通信のみで実現するとオーバヘッドが大きい．
- **負荷分散の困難**
  通信遅延が大きいため，実行すべきタスクを持たないプロセッサがあり，別のプロセッサに実行可能なタスクが余っていても，タスクの割当ては瞬時には行われない．
- **不要な探索の即時中断の困難**
  枝刈りの結果，あるノード以下の木を探索することが不要になることもあるが，探索の中断と別のタスクへの割当てが瞬時に行われない．

研究レベルでは，これらの弱点を克服するための有望なアルゴリズムがいくつか発表されているが，多くのプログラムが開発されているコンピュータチェスの世界でもクラスタ並列化は一般的にはなっていない．

一方，将棋プログラムの疎結合並列計算機上の実行例は少ないながらも存在している．1997 年 2 月に開催された第 7 回コンピュータ将棋選手権には「スーパー将棋」が SR2201 というスーパーコンピュータの 8 プロセッサ構成で参加している．これは root の子供のみを並列に探索するという簡単な並列化を行ったもので，予選 2 勝 5 敗という結果に終わっている．

また，東京大学大学院情報理工学研究科の戦略ソフトウェア創造人材養成プログラムの一貫で 720 台のノート PC を接続して，将棋プログラム「激指」を使った並列探索を試みた事例がある．これはある程度のスピードアップは得られたものの，並列化のオーバヘッドが大きく，1 台で実行する逐次プログラムよりも遅い結果しか得られなかったし，一局を通じて対戦可能なプログラムは作られなかった [3]．

複数のマシンを「逐次で行う木探索を高速に実行する」目的以外に利用した例としては，第 7，8 回コンピュータ将棋選手権に登場した S1.2，S1.3 がある．これは 2 台の計算機を用意して 1 台には通常探索を実行させ，残りの 1 台には詰みの有無だけを探索させるというものである．また，第 19 回世界コンピュータ将棋選手権での文殊 [4] および，第 20 回世界コンピュータ将棋選手権での「Bonanza Feliz」で使われた合議アルゴリズムも複数マシンの有効な利用法として注目を集めた．

なお，第 20 回世界コンピュータ将棋選手権では「ボンクラーズ」もクラスタ環境での並列化を実行している．Web ページによる解説☆2 を見る限りでは，本稿で述べる GPS 将棋のクラスタ並列化よりは本格的な並列化を試みているようである．大規模クラスタ環境で実行したときにどの位強くなるのか今後の展開を期待したい．

## GPS 将棋のクラスタ並列化

一般には台数に比例するスピードアップを並列化の目標とすることが多いが，今回は，台数の平方根に比例するスピードアップしか実現できない並列化モデルを用いた．

木探索に関しては，「実質的なスピードアップがマシン数の平方根になる高速化は容易」と言われることが多い．これは，以下の考察に基づくものである．

- 木のルートから決められた深さまですべてのノー

☆2 「インサイド・ボンクラーズ」http://aleag.cocolog-nifty.com/blog/2010/01/post-6445.html

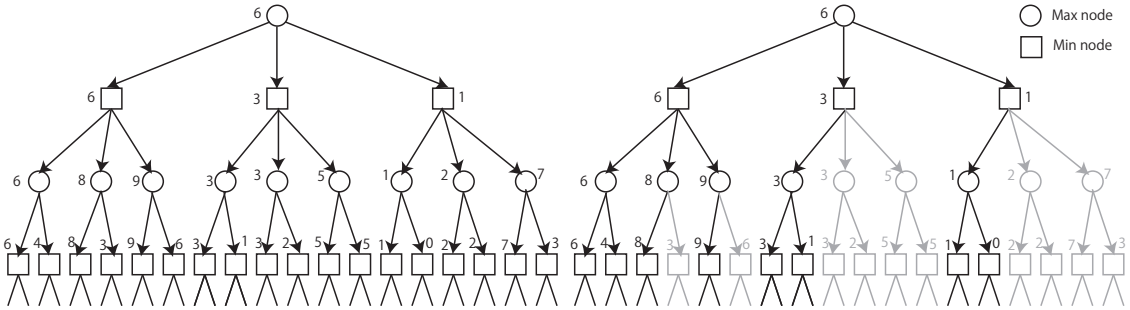


図-1　単純なミニマックス法の並列実行とアルファベータ法との比較

ドを展開して，それより深い木探索部分を別プロセスで並列に実行するという方法で容易に並列化することができる．

- このとき，決められた深さまでの展開ノードはその深さまでの単純なミニマックス法の探索ノード数と等しい．
- 理想的なアルファベータ法の探索ノード数は単純なミニマックス法の探索ノード数の平方根となる．

**図-1** の左側に，この方法で深さ3まで展開してその下を並列に探索する木の例を示す．この図では深さ3の18個のノードを別プロセスの逐次プログラムで探索することになる．

この木は，左の子供ほど良い手になるように並べているが，この順でアルファベータ法で探索を行ったときに探索する部分を図の右側の黒で，不要になる部分を灰色で示す．この図では，18個のノードのうち8個のノードしか有効な探索をしていないということが分かる．展開を打ち切る深さが深くなるにつれて，有効な探索を行うノードの数はノード数の平方根に収束するので，無駄になる割合が大きくなっているわけである．

実際のゲーム木の探索では，

- ゲームの局面は木ではなく合流やサイクルのあるグラフだが，逐次探索ではトランスポジションテーブルにより合流を扱える．トランスポジションテーブルを共有せずに並列化するだけでは合流は扱えない．
- アルファベータ法では適切な探索ウィンドウ（アルファベータ法における$\alpha$値，$\beta$値の組）が設定

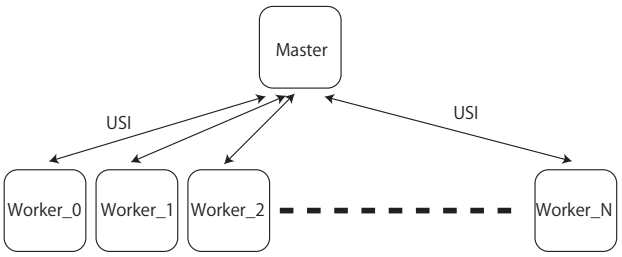


図-2　マスタ・ワーカーモデルによる実現

されているが，この並列化では探索ウィンドウが$[-\infty, \infty]$となっているので，そのノード以下の探索ノード数が増える．

ことがあるので，台数の平方根のスピードアップが得られない可能性はある．一方で，実際の探索が理想的なアルファベータ法ではないために，台数の平方根以上のスピードアップが実現される可能性もある．

このモデルに基づく並列化であるが，実際に将棋に適用するのはそれほど自明ではない．平均分岐数80の将棋でトップレベルで全幅探索を用いると，深さ2で展開しても$80^2$=6400プロセスが必要になるためである．

本稿における並列化では，実装を容易にするために**図-2** のようなマスタ・ワーカーモデルを用いるとともに，必要なプロセス数を抑えるための工夫を行っている．以下に概要を述べる．

- 一手ごとの探索時間は局面の進行度と持ち時間，経過時間，手数などから決定する．
- 短い時間（制限時間1秒）で，Multi PV（rootでPV以外のノードの探索のウィンドウ幅を広めにして，PVよりも少し劣る手も求める）で探索し

（以下では presearch と呼ぶ），そのときの rank（順序）に応じて，子供ノードにリソース（ワーカー群）の数を調整して割り当てる．

- presearch 中に別ワーカーを使って並列に詰み探索も行う．
- 残り時間が少ない場合は実現確率のみで候補手を生成する．
- 親から渡される「ワーカー群」のサイズが 1 になったら 1 ワーカーで残り時間いっぱいそのノードを探索させる．
- 「残りの手」は 1 ワーカーで探索を行う．

以上に対応するマスタ部分の擬似コードを**図 -3** に示す．

擬似コード中で関数 distribute として現れている子供へのワーカー群の割当は root では，$\frac{1}{4}$，$\frac{1}{4}\times\frac{3}{4}$，$\frac{1}{4}\times\left(\frac{3}{4}\right)^2$，...非 root では $\frac{1}{2}$，$\left(\frac{1}{2}\right)^2$，$\left(\frac{1}{2}\right)^3$，... のように rank に応じて台数を減らす形で行っている．**図 -4** に木の探索の際にワーカーを割り当てていく様子を示す．

これは，**表 -1** のように，presearch での rank が高い手が最終的に選択される確率が高く，rank に応じて選択確率が指数関数的に下がってきているという観察に基づいている．

```
int search(board,workers,time_left){
  if workers の数が 1
    return workers[0].search(board,time_left)
  end
  if time_left が 10 秒以上
    # ks に候補手を入れる
    Thread.new(ks = workers[0].presearch(board))
    Thread.new(cm = workers[1].has_checkmate(board,1 秒 ) )
    Thread.join()
    if cm が checkmate
      return +INF
    end
  else
    # 時間がないので実現確率のみで候補手生成
    ks = workers[0].gen_moveprobabiity(board)
  end
  if ks が 0 手
    # 負けの場合もあるが一応時間をかけて読んでみる.
    return workers[0].search(board,time_left-1 秒 )
  else if ks が 1 手で forced move
    return
      -search(board.do_move(ks[0]),time_left-1 秒 ,workers)
  end
  d=Array.new(ks.size+1)
  # workers( ただし workers[0] 以外 ) を子供たちに分ける.
  d=workers.distribute(ks)
  v=Array.new(ks.size+1)
  # その他の手を読む
  Thread.new(v[ks.size]=
    workers[0].search_other(ks,board,time_left-1 秒 ))
  for i=0 to ks.size
    Thread.new(v[i]=
      -search(board.do_move(ks[i]),d[i],time_left-1 秒 ))
  end
  Thread.join()
  return max(v) # 子供の手で最大の値を返す
}
```

図 -3　マスタプログラムの擬似コード

マスタ・ワーカー構成で実現する場合，

- マスタとワーカープログラムの起動，マスタ・ワーカー間の通信はどのようにして行うか？
- マスタとワーカーをそれぞれどのようなプログラミング言語で実装するか？

などの実装上の選択肢がある．

今回は，

- ワーカープログラムはクラスタ並列用に開発せずに，GPS 将棋の USI プロトコル GUI 用エンジンプログラムである gpsusi をそのまま用いる．
- マスタとワーカーの間の通信は同一マシン内では pipe，別マシンでは ssh を使ったストリーム通信上の USI プロトコルに基づく通信．

とした．

ここで使われた USI（Universal Shogi Interface）[☆3] はチェスプログラムの GUI プログラムと思考プログラムの間の通信に使われる UCI（Universal Chess Interface）を参考に Tord Romstad 氏が提案した将棋プログラムの GUI プログラムと思考プログラムの間の通信プロトコルであり，以下のような特徴がある．

- 通信に使われるのは 7-bit ASCII のみ．
- 通信内容は行単位で parse 可能．短くするために

☆3　http://www.glaurungchess.com/shogi/usi.html

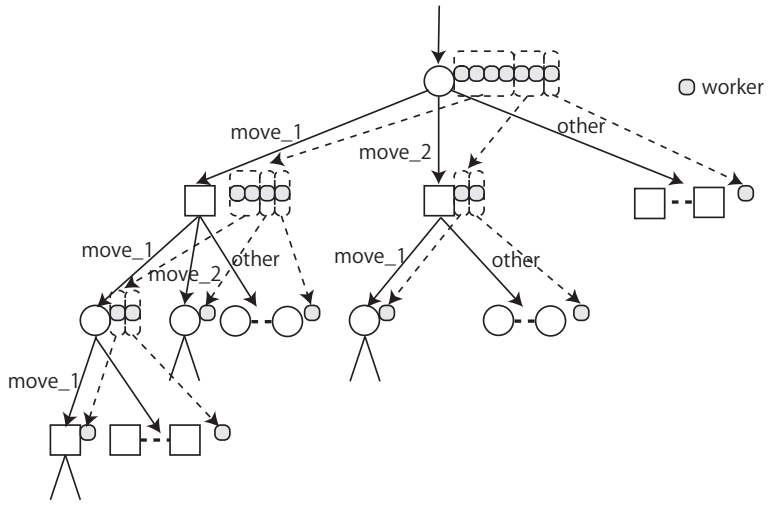


図 -4　rank による子供へのワーカー群の割当て

| 順位 | 0 | 1 | 2 | 3 | 4 | 5 | その他 |
|---|---|---|---|---|---|---|---|
| 選択確率 (%) | 46.3 | 21.4 | 12.0 | 6.65 | 4.70 | 2.78 | 5.93 |

表 -1　presearch による順位と最終的な選択確率

エンコーディング上工夫されている.

- 非同期 (asynchronous) な通信を必要とするコマンドも存在.

USI プロトコルに対応した GUI プログラムとしては,「将棋所」☆4,「プチ将棋」☆5, BCMShogi ☆6 など多数存在する. 特に「将棋所」は独自の USI 拡張をしていて, GPS 将棋もそれの最低限部分を実装した実行プログラム gpsusi を使っていた. 今回はそれに加えて, gpsusi にいくつかの拡張コマンドを実装した.

マスタプログラムは Ruby で記述した. これは以下の理由による.

- Floodgate の将棋サーバで使われている将棋盤クラス等が再利用できる.
- 実験を行うには, コンパイル時間のかからず, USI の parse が自然に書けるインタプリタ言語が適している.
- マルチスレッドプログラミングが容易に行える.

☆4 http://www.geocities.jp/shogidokoro/
☆5 http://www.geocities.jp/shogi_depot/
☆6 http://home.arcor.de/Bernhard.Maerz/BCMShogi/

一方で,

- スクリプト言語の中でも実行速度は遅い.
- 変数名のミス等はコンパイル言語ではコンパイル時に検出されるが, インタプリタなので, 実行するまで検出されない.

などのデメリットも当然ある. マスタプログラムの Ruby プログラムのうち, 今回のクラスタ並列化で新たに書いたコードは 1500 行程度である.

## 評価

予備実験として, Xeon X5365 (3GHz) × 2 = 8 core のサーバを使って,

- 8 スレッド使ったスレッド並列版
- 1 スレッド 8 ワーカーを使ったクラスタ並列版

を指定局面から 30 秒の秒読み先読みなしで 40 回戦わせたところ, 前者が 26 勝 14 敗と勝ち越した. 一方,

- 4 スレッド使ったスレッド並列版
- 1 スレッド 8 ワーカーを使ったクラスタ並列版

で同じ条件で対戦させたところ, 20 勝 20 敗と互角の結果を得た.

## 〈4〉大規模クラスタシステムでの実行

このことから，マシン 1 台ごとにスレッド並列版をワーカーとして 1 つ動かして，それをまとめたクラスタ並列構成にするという方針がまとまった．

今回は各ワーカーの能力が異なるヘテロな環境で実行したが，

- あるノードを探索する際には，まず，1 番速いワーカーで presearch を 2 番目に速いワーカーで詰み探索を行う．
- 複数の候補が見つかったあとで，「その他の手」は 1 つのワーカーでしか実行できないので，一番速いワーカーを使う．他の候補手には，2 番目に速いワーカーからラウンド・ロビンで割り当てていく．

というアルゴリズムで割当てを行った．

第 20 回世界コンピュータ選手権には，以下の構成で参加した．

- マスタ Xeon X5365（3GHz）8core 8 thread, ruby プログラムのみを動かす．
- ワーカー
  - Xeon X5570 (2.93GHz) × 2, 8core, 16 thread
  - Xeon X5470 (3.33GHz) × 2, 8core, 8 thread
  - Opteron 2376 (2.3GHz) × 2, 8 core, 8 thread
    4 台
  - Opteron 280 (2.4GHz) × 2, 4 core, 4 thread
  - Core 2 duo (2GHz), 2 core, 2 thread 307 台

構成の中で，最も台数の多い 307 台のマシン（Apple 社 iMac）の設置されている演習室 2 室のうちの 1 室の写真を**図 -5** に示す．省電力のために端末は入力がないまま一定時間経過すると，ディスプレイスリープ状態に入る（ssh で使用する分には問題ない）はずだが，図を見ると何台かはディスプレイ表示状態になっている．

この構成での動作を確認するために，人間用の問題集「ラクラク次の一手 2」[5] の問題 216 問を一手 20 秒で解かせてみた．結果は 216 問中正解が 196 問となり，Xeon X5570 × 2 のスレッド並列版で一手 30 秒の制限時間で解かせたときの正解数である 186 問よりも改善されていた．

図 -5　使用した演習室の 1 つ

次に，Xeon X5570 × 2 のスレッド並列版と上記構成から該当マシンのみを除いたクラスタ並列版を，持ち時間 25 分で指定局面からの連続対戦をさせたところ，途中でバグが出たのを除くと 13 連勝となった．

iMac は他のサーバ機と違って誤り訂正機能のないメモリを使っているし，台数も多いので，計算の二重化や故障時の再構成を考慮することが望ましい．サーバとの通信プログラムは思考プログラムと分けて，思考プログラムのプロセスがエラー等で落ちたときに，通信プログラムが再起動するような仕組みを組み込む構想はあったが，リモート参加用に新たに作った対戦プログラムにそこまで組み込む時間がなかったので，314 台のうち 1 台でも異常動作したらその場で負けが確定するという厳しい状況で決勝戦に望んだ．

結果的に，7 局無事に落ちずに対戦を終えることができたのは，運が良かったとしか言いようがない．**表 -2** に示すように，結果は 5 勝 2 敗で 3 位に終わった．

選手権の 7 局での読み筋の中の PV（Principal Variation）の長さ（静止探索，詰み探索を含まない通常探索で到達するノードの深さ）をグラフにしたものを**図 -6** に示す．4 つのグラフはそれぞれ以下を示す．

**WCSC2010**　今回のクラスタ版による 7 局の PV の長さ

| | 1回戦 | 2回戦 | 3回戦 | 4回戦 | 5回戦 | 6回戦 | 7回戦 |
|---|---|---|---|---|---|---|---|
| 対戦相手 | 芝浦将棋 | 激指 | YSS | 習甦 | Bonanza Feliz | ボンクラーズ | 大槻将棋 |
| GPSの手番 | 後手 | 先手 | 後手 | 先手 | 後手 | 先手 | 先手 |
| 勝敗 | 勝 | 勝 | 勝 | 負 | 勝 | 負 | 勝 |

表-2　世界コンピュータ将棋選手権決勝でのGPS将棋の対戦

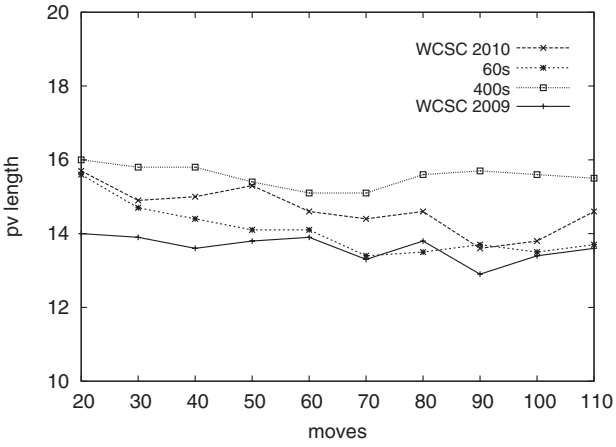


図-6　PVの長さの比較


【68手目3三桂　まで】

図-7　対ボンクラーズ戦

**60s**　Xeon X5570 × 2のスレッド並列で一手につき，60秒打ち切りで7局の各局面を考えさせたときのPVの長さ(選ばれた手はWCSC2010とは一致しない)

**400s**　Xeon X5570 × 2ののスレッド並列で一手につき，400秒打ち切りで7局の各局面を考えさせたときのPVの長さ（選ばれた手はWCSC2010とは一致しない)

**WCSC2009**　昨年度の決勝7局（Xeon X5570 × 2のスレッド並列)のPVの長さ

WCSC2010は最大35秒の探索でも，WCSC 2009より平均一手位PVが長いこと(厳密には違う局面を読んでいるので直接比較はできない)，60sよりはPVが長いことが分かる．一方，400sよりも探索したノードの総数が数倍多いはずだが，無駄なノードを多数探索しているため，PVの長さは400sには及んでいない．

明らかな失着としてプロ棋士から指摘を受けた局面について読みの内容を確認してみる．

**図-7**は第6戦ボンクラーズ戦の68手目の局面である．この局面は駒得もあり7一角成としてGPS将棋が優勢と言われていた．ここでは，60s, 400s共に7一角成を選ぶが(400sはその後で4三金2四飛の展開を読んでいる)，クラスタ版は2四飛車と仕掛ける手を選んで一気に形勢をそこねてしまった．

クラスタ版はPVとして「2四飛, 2三歩，3四飛」で評価値は431を返していた．400sの「7一角成，4三金右，2四飛」の読み筋と合わせて，クラスタ版の読みを解析して，ワーカー1つで担当するまで木を展開してみると，以下のようになった．

```
第1候補 7一角成
 + 第1候補 8六歩
 | 第2候補 9一飛
 | 第3候補 4三金右
  + 第1候補 8二馬
  | 第2候補 7五歩
  | 第3候補 7七金右
  | その他　2四飛車 △ 365
第2候補 3五歩
第3候補 6四歩
第4候補 1六歩
第5候補 9六歩
第6候補 7五歩
第7候補 3三桂成
第8候補 3三桂不成
第9候補 7七銀
第10候補 2四飛
 + 第1候補 2三歩
  + 第1候補 2九飛車
  | 第2候補 3三桂成
  | その他　3四飛車 ○ 431
 + 第2候補 2三銀
 + 第3候補 2三金
 + その他　3一玉
...
その他 6九金
```

このように，2四飛はrootで第10候補の手であり，「2四飛, 2三歩，3四飛」は探索深さが十分でないところで，得られた評価値を使っていることが

分かる．平均してある程度深く読めても，勝負どころで探索深さが足りないと致命的なミスになるということで，将棋というゲームの怖さを改めて実感した．

## 今後の大規模クラスタシステム

本稿で述べた大規模クラスタシステムは，「探索効率を極限まで高める」というものではなく，「簡単な枠組みでリソースをつぎ込めばつぎ込むほど強くなることを実証する」のが目的で試作されたものである．結果的には2敗して優勝を逃したことで，目的を果たせず残念な結果に終わった．ただ，今回の結果で判明した弱点も，探索深さが足りないときに再探索を行うなどの改良である程度は克服できると期待される．

gpsusi以外であっても，USIにgpsusi同様の拡張をした思考プログラムであれば，マスタ部分を利用して同等の並列化が行えるはずである．また，今回のクラスタ並列化は通信量(および頻度)が少ないので，一般参加型の分散コンピューティングで適用できる可能性もある．

今回作成したシステムは，マスタ部分のRubyプログラム，ワーカー部分のgpsusi共にGPS将棋のソースレポジトリ☆7から入手可能である．手元に空いているクラスタがある方はぜひお試しいただきたい．

**参考文献**


1) 金子知適：コンピュータ将棋の新しい波：3. 最近のコンピュータ将棋の技術背景とGPS将棋，情報処理，Vol.50, No.9, pp.878-886 (Sep. 2009).
2) Marsland, T. A. and Popowich, F. : Parallel Game-tree Search, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.PAMI-7, No.4, pp.442-452 (1985).
3) 金田憲二：多数の遊休PC上での分散ゲーム木探索，第1回大域ディペンダブル情報基盤シンポジウム，http://web.yl.is.s.u-tokyo.ac.jp/~kaneda/pub/kaneda-coe04-abst.pdf (2004).
4) 伊藤毅志：コンピュータ将棋の新しい波：4. 合議アルゴリズム「文殊」 単純多数決で勝率を上げる新技術，情報処理，Vol.50, No.9, pp.887-894 (Sep. 2009).
5) 日本将棋連盟書籍編：ラクラク次の一手2 基本手筋集，日本将棋連盟 (2003).


(平成22年5月31日受付)

☆7 http://gps.tanaka.ecc.u-tokyo.ac.jp/cgi-bin/viewvc.cgi/trunk/gpsshogi/?root=gpsshogi


**田中　哲朗（正会員）** ktanaka@tanaka.ecc.u-tokyo.ac.jp

1965年生．1987年東京大学工学部卒業．1992年同大学院博士課程修了．博士（工学）．現在同大情報基盤センター准教授．2009年よりゲーム情報学研究会主査．

**金子　知適（正会員）** kaneko@graco.c.u-tokyo.ac.jp

東京大学大学院総合文化研究科助教．2008年よりゲームプログラミングワークショップ共同プログラム委員長．