解説

# コンピュータ将棋における クラスタ並列探索

Cluster-parallel Game Tree Search in Computer Shogi

伊藤英紀

## Abstract

2012年1月に将棋電王戦が行われ，筆者の開発した将棋ソフト「ボンクラーズ」が米長邦雄元名人に勝利した．2012年現在の将棋ソフトの実力は人間のトップ棋士をも越えるほどである．最近のソフトの棋力向上の原動力の一つがクラスタ並列探索である．$\alpha\beta$探索のクラスタ並列化は技術的な困難が多く，少し前までほとんど成功例がなかったが，近年ボンクラーズがこの分野の技術を先導・開拓し，棋力向上を実現した．本稿ではコンピュータ将棋におけるクラスタ並列探索技術の過去の事例を振り返り，その困難さについて述べた後，ボンクラーズの実装を中心に技術のポイントを幾つか解説していく．

キーワード：コンピュータ将棋，並列処理，$\alpha\beta$探索，PVSplit，YBWC

## 1. はじめに

2012年1月に第1回将棋電王戦が行われ，筆者の開発した将棋ソフト「ボンクラーズ」が米長邦雄永世棋聖・元名人に勝利した[1]．ボンクラーズはまた，2011年末にインターネット上の将棋対局サイトである「将棋倶楽部24」に参戦し，プロを含む高段者を相手に，棋力の指標であるレーティングにおいて歴代1位の3,364を記録し，プロをも上回る実力を証明した[2]．2012年現在の将棋ソフトの棋力は，人間のトップ棋士（名人）のそれに匹敵するか，若しくは対局条件によっては人間を越えるレベルに達しているものと考えられる．

5年前の2007年には当時のトップソフトであった「ボナンザ」[3]が当時の渡辺竜王に敗れている．この時点ではソフトの棋力はトッププロよりかなり下であったが，この5年の間に急速に進歩した．

ボンクラーズの棋力のベースとなっている技術は大きく三つに分けられる：

（a） コンピュータチェス由来の探索技術（※スレッド並列探索を含む）
（b） 機械学習による評価関数の精緻化
（c） クラスタ並列探索

（a）はコンピュータチェスの世界で開発されてきた技術をそのまま取り入れたものである．将棋とチェスはゲームとしての性質が似ており，チェスの探索技術の一部はそのまま将棋にも使用できる．（b）は上述のボナンザで保木[4]が開発した技術である．（c）は筆者が新規に開発した技術であり，本稿は（c）に焦点を当てていく．なお，コンピュータ将棋固有の技術としてはこのほかに長手数詰将棋探索[5]や実現確率探索[6]等があるが，ボンクラーズではこれらの技術は使用していない．

将棋の棋力は深く読むことで向上するため，並列化によって探索を高速化し，限られた時間内でより多量に手を読むことは不可欠である．コンピュータ将棋の基本アルゴリズムである$\alpha\beta$探索[用語]のクラスタ並列化は技術的な困難が多く，少し前までほとんど成功例がなかったが，2009年から筆者らがこの分野に取り組みはじめ，2011年5月の世界コンピュータ将棋選手権[7]で優勝して本技術の実効性を示した．

本稿では以下，2. 過去の事例，3. 並列処理の課題，4. ボンクラーズの実装，5. まとめ，の順に述べていく．

伊藤英紀 （株）富士通研究所ITシステム研究所
Eiki ITO, Nonmember (IT Systems Laboratories, Fujitsu Laboratories Ltd., Kawasaki-shi, 211-8588 Japan).
電子情報通信学会誌 Vol.96 No.2 pp.117-123 2013年2月

## 2. 過去の事例

並列処理を分類すると，大きく密結合並列と疎結合並列に分けられる．後者はクラスタ並列とも呼ぶ．この分類は，計算ノード間でメモリを共有できるか否かに着目したものである．

メモリを共有する前者では，プログラミングモデルとしてスレッドが使用でき，この場合はスレッド並列とも呼ぶ．スレッド並列はコンピュータチェスでも古くから利用されていて実績があり，ソースを公開している強豪チェスソフトのCrafty[8]も使用している．Craftyのスレッド並列手法をほぼそのままの形で活用している将棋ソフトもボナンザ[9]，YSS[10]をはじめ多い．スレッド並列の最大の欠点は，並列化の規模に限度がある点である．メモリを共有する関係上，ほとんどきょう体内の並列化に限定され，何百台をも接続するような構成は難しい．

クラスタ並列の方は，ネットワークで接続するため，規模の面では何千台だろうが接続するだけなら可能である．ただしこちらはスレッドが使えないため，MPI（Message Passing Interface）のようなメッセージ通信のライブラリを用いて並列化を実現する必要があり，スレッドよりは難易度が格段に上がる．このため，少し前までは将棋ソフトでクラスタ並列を実現して大会に参加したものは筆者の知る限り皆無であった．

コンピュータチェスにおいてクラスタ並列で使えるアイデアとしては，論文レベルでは例えばAPHID[11]などの例はある．しかし，APHIDが実際のコンピュータチェスに使われてトップレベルの棋力を実現したわけではない．論文に書ける（学会的な意味でアイデアの新規性がある）ことと実際のシステムで棋力を向上させることの間にはギャップがあり，後者の方がはるかに難しい．論文に書かれたアイデアで，実際のコンペティションに耐え得る性能を出せたクラスタ並列技術はこれまた筆者の知る限り皆無である．

チェスの世界で人間のチャンピオンを破ったことで有名なDeep Blueは，実装方法の詳細は分かっていないが，論文から推察するにクラスタ並列を使用していたと思われる．しかしながらDeep Blueの作者自身は文献(12)においてその並列性能に関して「(並列効率が)8〜12%．改善の余地がある．」と述べており，かなり並列性能は低かったことがうかがわれる．

論文を離れて大会に目を向けると，例えば2009年のComputer Olympiadのコンピュータチェス部門では，二つのソフトがクラスタ並列を使用している．そのうちの一つ，Rybkaは優勝しており，もう一つのDeep Sjengも上位に入っている．ただしいずれも並列処理方式の詳細が論文等で公開されてはいないようである．

コンピュータ将棋においては，2009年春の時点ではクラスタ並列を検討している開発者すら恐らくいなかったと思われる．その中で筆者がまず2009年5月に検討を開始し[13]，プロトタイプが動いたところで状況を2010年1月にブログに公開した[14]．それに触発されたのか，その後2010年2月にはGPS将棋チームもクラスタ並列の開発を始めた[15]．2011年の大会でボンクラーズが優勝するとクラスタ並列への注目度が上がり，2012年の大会では5チームがクラスタ並列を使用するまでになっている．3年前は誰も見向きもしていなかった分野がここまで隆盛するに至ったのは，先導してきた立場としては誠に喜ばしいと思っている．

## 3. ゲーム木探索における並列処理の課題

コンピュータ将棋でもコンピュータチェスでも，局面をノード，指し手を枝とするグラフを考える．これをゲーム木と呼ぶ．将棋やチェスにおけるゲーム木探索の基本アルゴリズムは，$\alpha\beta$探索及びその改良であるPVS(用語)探索である[16]．PVS探索は並列処理に限らず適用可能なものである．並列化のアルゴリズムとしてはPVSplit(用語)やYBWC（Young Brothers Wait Concept）[16]が従来多く使われてきた．

計算ノードが例えば10台あると，1台のときと比較して処理速度は上がるだろうか？　ある種の計算，例えば行列計算や物理シミュレーションでは，ほぼ10倍になる．1,000次元の行列計算をするのに100次ずつに分割して10台で並列計算すれば，多少のオーバヘッドはあっても，速度は恐らく9.$x$倍にはなるだろう．

ところがコンピュータ将棋ではこうはいかない．単純なやり方で並列化をしたのでは，効率が2倍にもならないこともあり得る．なぜか？　例えば図1のような状況である．一番上のノード（ルート）の局面で，可能な指し手が幾つかある．それらの手を指した後のノードを一つずつ探索するのだが，このときそれらの子ノードを探索するのにかかる計算量が，行列計算等と違って一定でなく大きくばらつくのである．図の数字はサンプルだが，2倍や3倍でなく100倍以上の差が出ることも頻繁にある．単純な並列化の例として，例えば各子ノードに計算ノードを1台ずつ割り当てたとしよう．すると，軽い子ノードに当たった計算ノードはすぐに処理が終わってしまう一方，重い子ノードに当たった計算ノードは延々と処理を続け，その間軽いのを担当していた計算ノードは待っているだけ，といった状況になる．このために，10台でも速度2倍にもならない，ということになってしまう．

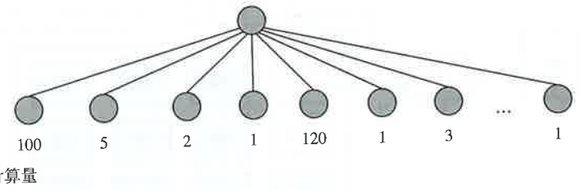


**図1　計算量がばらつくゲーム木**　単純に子ノード一つに計算ノード1台とすると幾つかの重いノードに負荷が集中し，並列化の効果が出ない．

並列化の効率が落ちる原因を，同期・通信・探索の3種のオーバヘッドに分類する考え方もある[17]．同期オーバヘッドとは，ある計算ノードが他の計算ノードの処理を待つ必要があるため，待つ間処理が進まないことである．上記の例はこの同期オーバヘッドである．通信オーバヘッドとは，ネットワークを介してノード同士が通信するため，通信にかかる時間である．メモリを共有するスレッド並列と比較すると，クラスタ並列では通信オーバヘッドが桁違いに大きい．これもクラスタ並列が難しい大きな要因である．

探索オーバヘッドというのは，ゲーム木探索でかなり特徴的な現象である．これは，「並列で探索することによって，必要な計算量そのものが増える」ことをいう．例えば，ある局面を1台のノードで探索すると，計算量が100かかるとする．これを2台で並列処理すると，1台目が80，2台目が70，つまり合わせて150の計算が必要になる，というような現象が生じる．つまりこの例だと，1台のときよりもトータル計算量が1.5倍になっている．これが探索オーバヘッドである．探索オーバヘッドの発生要因としては，トランスポジションテーブルの非共有，更新した$\alpha$値の伝搬遅延，$\beta$カットによるキャンセル発生，等がある．

2012年の世界コンピュータ将棋選手権におけるクラスタ並列利用チームのクラスタ構成概略を表1にまとめた．（筆者はこの大会ではプログラム名を「ボンクラーズ」から"Puella $\alpha$"と改名して参加した．）数百コア，数千コアのチームもあるが，コア数に150倍の差があっても勝敗的には大きな差はなく，台数に見合った処理向上は得られていないことが分かる．クラスタ並列の技術は，進展してきているとはいえまだまだ発展途上である．

**表1　第22回世界コンピュータ将棋選手権におけるクラスタ並列利用チームのクラスタ構成概略**（出典：コンピュータ将棋協会ウェブサイト http://www.computer-shogi.org/wcsc22）　勝敗は，Bonanzaと大槻将棋は二次予選のみ，他は二次予選と決勝の合計．

| チーム名 | ソケット数 | コア数 | 勝敗 |
|---|---|---|---|
| GPS将棋 | 804 | 3,224 | 13勝3敗 |
| Puella $\alpha$ | 4 | 22 | 12勝4敗 |
| ponanza | 9 | 70 | 11勝5敗 |
| Bonanza | 47 | 288 | 5勝4敗 |
| 大槻将棋 | 8 | 48 | 4勝5敗 |

## 4. ボンクラーズの実装

### 4.1　ハード及びソフト環境

ここからボンクラーズの解説に入っていくが，最初に簡単に使用している環境について述べておく．開発は個人の趣味として行っているため，自腹でそろえることが可能で自宅に置ける範囲の機材しか使えない．2012年の大会では，自作PCを中心に普通のデスクトップPC 4台（計22コア）の構成であった．総額40万円程度．毎年，全部は無理だが少しずつCPUを最新・最速に近いものにアップデートしており，ここ数年は大体年に20万円程度かけている．ただし電王戦のときは会社（富士通）の協力が得られ，ブレードサーバ（72コア搭載）を借用し使用させてもらった．

ソフトウェアの方は，Ubuntu Linux上でC++で開発している．クラスタ並列のメッセージ通信にはオープンソースのopenMPIライブラリを使用．その他使用しているツール名を挙げると，icc，idb，vim，make，vtune等．いずれも個人で非商用で使用する分には無料のものばかりである．インタフェースはCUIで，普通の端末でシェルからコマンドを起動する．

### 4.2　開発の経緯

そもそもボンクラーズ開発開始のきっかけだが，2009年までは筆者はFPGAで将棋ハードウェアを開発して

### ■ 用語解説

**$\alpha\beta$探索**　コンピュータ将棋・チェスの基本アルゴリズム．最善の1手だけ選べればよいとする考えから，最善ではないと判明した部分木の探索を省くことで計算量を削減する手法．

**PVS（Principal Variation Search）**　$\alpha\beta$探索の改良版．PV（最善応手列）上のノードのみ普通に$\alpha\beta$探索を行い，2番目以降の子ノードでは$\alpha\beta$幅を限界まで小さく設定することによって$\alpha\beta$探索より更に平均探索量を下げる手法．

**PVSplit**　$\alpha\beta$系探索（PVS含む）の並列化手法．PV上の最初の子ノードは並列化せずに探索を行い，$\alpha$値が確定したところで2番目以降のノードは並列に探索する．

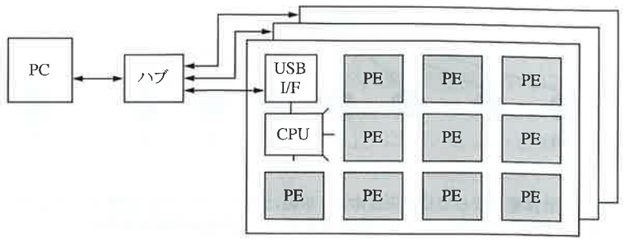


**図2　将棋ハードエンジンの大規模並列構成想定図**　PE=Processing Element. ここではASICチップを想定し，数コアが載るとする．ボード数枚で100コア超の構成が可能と予想される．

いた．こちらについては本稿の本題ではないので詳細は割愛するが，興味のある方は筆者の開発ブログ[18]に開発状況を記してあるので参照頂きたい．さてそのFPGA将棋では，市販のFPGA試作ボードを用い，1チップのFPGAに将棋エンジン1コアを載せて動作するところまで2009年の時点で進んでいた．次にやりたかったのは，その将棋エンジンを複数用意し，並列処理することであった．システム構成としては図2のようなものを想定していた．

しかしながら前述したように，当時クラスタ並列の技術は事実上存在しなかったため，まずこの技術を開発する必要があった．クラスタ並列探索では通常，1台がマスタとなって全体の進行を管理し，他がスレーブとなって$\alpha\beta$探索を繰り返し実行する．このスレーブ側については，FPGAでもソフトでも，並列アルゴリズムという観点からは実はどちらでも余り変わりはない．FPGAを扱うのは何かと手間がかかるため，並列アルゴリズムを開発する間はFPGAでなくソフトでいく方が開発効率が良いだろうと考えた．ではそのソフトはどうしようかと考えたとき，当時ソースが公開されていた幾つかの将棋ソフトの中で，棋力もかなり高く，コードも比較的読みやすいボナンザを選んだ，という次第である．もっともこの選択はかなり恣意的なもので，率直にいって並列化という観点からすると，別にボナンザでなくてもよかった．当時の考えとしては，いずれFPGAで置き換えるまでの代替にすぎなかったのである．

毎年5月に世界コンピュータ将棋選手権があるため，コンピュータ将棋開発の典型的なサイクルとしては，5月の大会終了直後から次の年の大会に向けて1年で次バージョンを開発となる．2009年5月まではFPGAに注力していて，その後ボンクラーズ（=「ボナンザベースのクラスタ並列」から命名）の開発を始めて，2010年，2011年，2012年と三つのバージョンを開発してそれぞれ大会に参加した．このうち，スレーブ側の方はボナンザソースから出発しており，その後いろいろと独自の工夫を加えたとはいえ8割方はボナンザの資産を流用している．マスタ側がクラスタ並列のアルゴリズムを実装する部分だが，ここは実は三つのバージョン全てが全く違う．つまり，毎年ほぼゼロから作り直している．

### 4.3　2010年バージョン

2010年版，つまり最初のバージョンでは，元々の想定が図2のような構成だったため，次の2点を前提とした．

- 数百コアでの並列化をターゲットにする
- スレーブ側の処理はできるだけ単純化すること

この2番目の前提の理由だが，当初のハードウェア実装としてFPGAを想定したため，FPGAの制約が幾つかある．FPGAでは搭載メモリ量に制限があり，余り大きなプログラムは載せることができない．実際，FPGA将棋の開発をしていた2008年には，プログラムを書いたもののメモリに乗り切らずコンパイル時にエラーが出たこともあった．そのためにこの制限を設けた．

この二つの制約を基に開発を進め，約1年かけて一応動作するものはできたが，残念ながら並列化の効果を出せるまでには至らなかった．つまり，クラスタ並列をしない1ノードのプログラムと，3ノード並列で動くプログラムとを対戦させたところ，並列にしてもほとんど強くなっていない，という状況であった．もっともこのときは，クラスタ並列が初めてということもあり，「まずはちゃんと動くものを作ろう」というレベルで，習作のつもりで正直余り強くなることも期待していなかったという面もあった．ただこのときに各種のデータを取るなどして，次版以降の改善のための参考情報は多く得られた．

### 4.4　2011年バージョン

2010年5月からまた次版の構想を練り始めた．このときまず考えたのは，4.3に挙げた二つの前提は捨てよう，ということ．将来FPGA/ASICを目指すとはいっても結構遠い先だろうから，まずは目先の大会で良い成績を残そう，そのためには現実的な構成であるPC数台ということを前提として，そこに最適化して作り込んでいくべきではないか，と思い直した．それならば数百コアは当面不要だし，スレーブ側のメモリの心配も不要となる．

2010年版の観察結果から，同期待ちが一番のボトルネックであることが分かっていた．その対策として，PVSplitをやめる，との結論に至った．

図3のようなゲーム木があるとき，PVSplitでは$A$の完了後$B$，$C$，$D$は並列に実行する．だが$B$，$C$，$D$全てが完了するまで（つまり，$P$が完了するまで）$Q$，$R$，$S$は実行しない．この方法では，PVの1段ごとに同期オーバヘッドが生じてしまう．そこで2011年版ボンクラーズでは，$B$，$C$，$D$と$Q$，$R$，$S$を並列に実行できるようにした．例えば$A$が値100を返したとする．（以下，表記上Negamaxを使う．つまり，深さ1ごとに符号を反転させる．）$P$の値は未確定だが，$B$，$C$，$D$が$\alpha$を更新しないとすると$-100$になるはずである．そこで$P$の値$=-100$を"仮定"して$Q$，$R$，$S$を探索する．このときこの"$-100$"は，$P$，$Q$，$R$，$S$の列においてまだ$\alpha$ではないが，$B$，$C$，$D$が完了すれば$\alpha$になる「であろう」値である．「(今は$\alpha$ではないが）いずれ$\alpha$になる値」という意味で，これを筆者は「$\alpha$の子供」="Puella $\alpha$"と呼んでいる．2012年版のプログラム名称はここからきている．以下では，この値を$\alpha'$とダッシュを付けて表記する．

もちろん，$P$が完了する前に$Q$，$R$，$S$を投機的に実行することになるので，$B$や$C$で$\alpha$値の更新があると$Q$，$R$，$S$はキャンセルした後リトライが必要になる．制御はかなり複雑になるが，ハッシュテーブルや反復深化等の技法により，PV以外での$\alpha$更新の確率は低く抑えることができるので，実行効率は向上する．

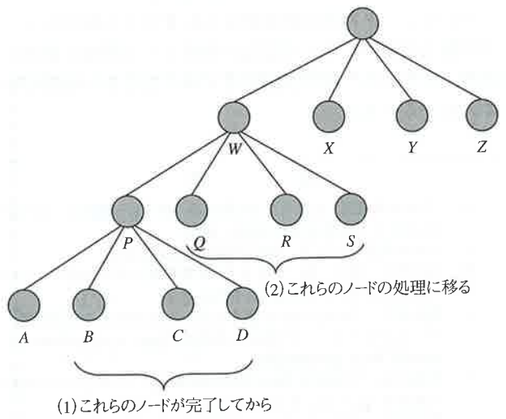


**図3　PVSplitの説明**　各列において，左端のノードが終了（=$\alpha$値が確定）しないと2番目以下のノードを探索しない．ボンクラーズではこの制約を取り払った．

逆に，1列上，例えば$Q$で更新があると，$B$，$C$，$D$は（まだ終了していなければ）$\beta$カットと同じ扱いになり，キャンセルできる．しかし2列上の，例えば$X$で更新があると，単純にキャンセルするわけにはいかない．2011年版では，2列までの並列実行を許したが，3列以上は並列実行しないようにした．$B$，$C$，$D$が実行終了しないと，$X$，$Y$，$Z$は開始しない．この場合「下の列で更新→上の列はキャンセル＆リトライ．上の列で更新→下の列はキャンセル」となり，比較的単純な制御で済む．

この改良は大きな効果を上げ，2011年の選手権では優勝することができた．このときはPC 3台/16コア構成で出場した．他にはGPS将棋とBonanzaがそれぞれ832コア，132コアのクラスタ構成で出場したが，それらを抑えて勝利し，クラスタ並列技術の最先端であることを実証した．

2012年1月の電王戦は，細かいチューン等はしたものの，基本的には2011年版に基づいたバージョンで臨み，米長元名人に勝利した．

なお，「$\alpha$確定前に投機的に上のノードを実行する」というアイデアだけならば，DTS[19]の論文で類似の考えが大分昔に出ている．しかしながらこのアイデアは，文献(19)の著者が現在開発しているチェスソフト(Crafty）では使われておらず，棋力向上に結び付かなかったことがうかがわれる．またそもそも，DTSはメモリ共有を前提としており，クラスタ並列には使えない．いずれにせよ，クラスタ並列で投機的実行を実装してコンペティションレベルに耐える高性能（棋力）を実現したソフトは，筆者の知る限りボンクラーズが世界初である．

### 4.5　2012年バージョン

2011年版はかなりの成功であったが，それでも細かく動作を分析するとまだ同期待ちがかなり多く見られた．主な原因の一つは王手のケースである．王手のかかった局面では合法手の数が著しく減るため，2列並列実行をしても実質1列しかないような状態になり，同期待ちが発生していた．これに対処するため，2011年版では制御の複雑化を嫌って避けていた3列以上並列実行に踏み込むことにした．図3の例でいうと，$B$，$C$，$D$から$X$，$Y$，$Z$まで全て同時に（投機的に）実行する制御に変えた．

これに伴い，2011年版までにはなかった新たなケースに対処が必要となった．いくつか例を挙げると：

図4を例にとる．今$A$は完了して値100を返したものとし，$B$，$C$，$D$，$Q$，$R$，$S$，$X$，$Y$，$Z$を同時実行しているとする．ここで$X$が値80を返したとしよう．このとき$P$-$S$の列では，$\alpha'=100$，$\beta=80$となっている．$\alpha'$

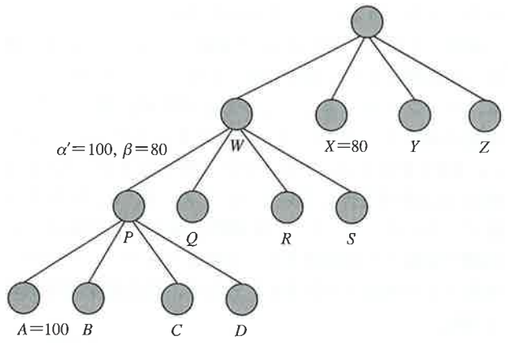


図4　サスペンドのケース　*P*-*S*の列において，Puella $\alpha$（図中では$\alpha'$と表記）が$\beta$を超えるため，$\beta$カットが予想されるので*Q*，*R*，*S*は実行すべきでない．

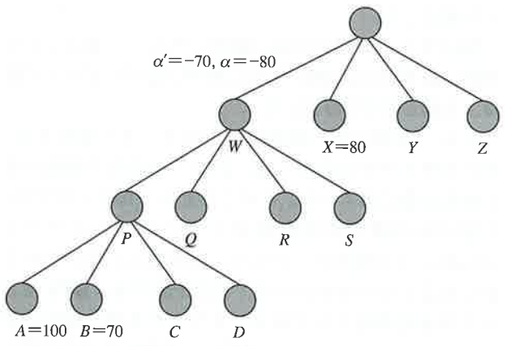


図5　$\alpha$と$\alpha'$が同時に出現するケース　*W*-*Z*の列において，$\alpha'$を使うべきか$\alpha$を使うべきか？一長一短があり判断が難しい．自己対戦等で検証する必要がある．

がいずれ$\alpha$になれば$\beta$カットすることになり，*Q*，*R*，*S*を実行するのは恐らく無駄になる．この理由から，*Q*，*R*，*S*は「サスペンド」する必要がある．つまり，即キャンセルはできない（*B*，*C*，*D*で更新があれば必要となる可能性がある）が，恐らく不要のまま終了する．サスペンドは2011年版にはなく，新たに必要になった機能である．また，*X*が返ったときに例えば*Q*を実行していたとすると，*Q*のジョブ自身はどうするのか，という問題もある．続行すべきか，それとも即キャンセルすべきか？　これは一長一短があり，数多くの自己対戦から判断すべきものだが，時間やリソースの関係でそこまで判断できていないというのが実情である．

もう一つ例を挙げる．図5のように，二つ離れた列で共に$\alpha$更新があり，下の列の方が良い値のケースを考える．このとき*W*-*Z*の列では$\alpha'$が−70，$\alpha$が−80である．*Y*，*Z*で使う（仮の）$\alpha$としてどちらを使うべきか？という問題が生じる．これも2011年版では生じない問題である．$\alpha'$を使うのは投機的な実行になるので，キャンセル＋リトライの危険性がある．しかし一方，$\alpha\beta$探索の特性上，より大きい値を$\alpha$として使用する方が探索の効率が上がるという事情があり，その面からは$\alpha'$を使う方が望ましい．これも一長一短で，どちらがよいのか簡単には判断がつかない．現状は，十分な自己対戦ができずに暫定的なコードになってしまっているのが実情である．

以上のように幾つか問題を抱えながらも，限られた自己対戦によって実装上の決断を行った結果，2012年版が2011年版におよそ2勝1敗ペースで勝ち越すようになった．2012年の大会ではGPS将棋に次ぐ準優勝という結果だったが，コア数に150倍の差があることを考慮すると，台数当りの並列化の効率としてはPuella $\alpha$が世界最高水準にあるといって差し支えないだろう．

## 5. ま　と　め

コンピュータ将棋におけるクラスタ並列探索の課題について述べ，それらを解決する実装の一つとしてボンクラーズ・Puella $\alpha$の手法を紹介した．2011年版のボンクラーズ＋今年（2012年）1月現在のハードウェアで既に，電王戦では元名人に勝利し，また将棋倶楽部24ではプロ棋士をも上回る実力を示した．その後，ソフト面では2012年版が2011年版に大きく勝ち越すまでに改良が進み，またハード面でも2012年春にインテル社からIvy Bridgeベースのサーバプロセッサがリリースされ，処理速度・コア数共に向上している．これらの事実から，現時点において既にコンピュータが名人を上回っているとの推測は当然成り立つし，また実際にそう考えている関係者も多い．しかしその推測が正しいかは，実際に対局して検証する以外にすべがない．

コンピュータ将棋の開発者はこれまで「人間のトッププロ棋士に勝つ」ことを最大の目標として開発を進めてきた．どうやら，次の目標を考えねばならない時期がきているようである．

## 文　献


（1）“米長永世棋聖　将棋ソフトに敗れる「慎重に…裏目」，”読売新聞2012年1月15日朝刊39面，2012.
（2）将棋倶楽部24サイト，http://www.shogidojo.com/
（3）保木邦仁，渡辺　明，ボナンザVS勝負脳—最強将棋ソフトは人間を超えるか，角川書店，2007.
（4）保木邦仁，“局面評価の学習を目指した探索結果の最適制御，”ゲームプログラミング2006発表資料，http://www.geocities.jp/bonanza_shogi/gpw2006.pdf
（5）脊尾昌宏，“共謀数を用いた詰将棋の解法，”コンピュータ将棋の進歩2，pp. 1-21，共立出版，1998.
（6）鶴岡慶雅，“将棋プログラム「激指」，”コンピュータ将棋の進歩4，pp. 1-17，共立出版，2003.
（7）“コンピュータ将棋　進化，”朝日新聞2011年5月17日夕刊7面，2011.
（8）R. Hyatt, Crafty ソースコード，http://www.craftychess.com/crafty-23.4.zip
（9）保木邦仁，ボナンザソースコード，http://gi.cs.uec.ac.jp:10140/bonanza_v6.0.zip
（10）山下　宏，“Crafty解析メモ，”http://www32.ocn.ne.jp/~yss/csa0307.txt
（11）M. Brockington, “APHID game-tree search,” Advances in Computer Chess 8, 1996.
（12）H. Campbell, J. Hoarn, and F. Hsu, “Deep blue,” Artif. Intell., vol. 134, pp. 57-83, 2002.
（13）伊藤英紀，“FPGA将棋開発の軌跡，”コンピュータ将棋協会誌，vol. 21, pp. 3-6, 2009.
（14）伊藤英紀，“ボンクラーズ，デビュー，”筆者の開発ブログ，http://aleag.cocolog-nifty.com/blog/2010/01/post-875f.html
（15）田中哲郎，“将棋プログラムの大規模並列実行，”第13回プログラミングおよびプログラミング言語ワークショップ資料，2011.
（16）M. Brockington, “A taxonomy of parallel game-tree search algorithms,” ICCA Journal, vol. 19, no. 3, pp. 162-174, 1996.
（17）R. Feldmann, P. Misliwietz, and B. Monien, “Studying overheads in massively parallel min/max-tree evaluation,” Proc. 6th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 94-103, 1994.
（18）伊藤英紀，“A級リーグ指し手1号，”筆者の開発ブログ，http://aleag.cocolog-nifty.com/
（19）R. Hyatt, “The DTS high-performance parallel tree search algorithm,” Online publication, http://www.cis.uab.edu/info/faculty/hyatt/search.html


（平成24年7月18日受付　平成24年9月5日最終受付）


伊藤　英紀（いとう　えいき）

1985東大・理・情報科学卒．1988同大学院修士課程了．同年富士通株式会社入社．以来，CPUハードウェア設計，ファウンドリ/ASICのマーケティング等に従事．現在(株)富士通研究所在籍．趣味でコンピュータ将棋を開発し，2011世界コンピュータ将棋選手権優勝．