ビートトラッキングシステムの並列計算機への実装 — AP1000によるリアルタイム音楽情報処理 —

後藤真孝[†,††] 村岡洋一[†]

本稿では，並列計算機の音楽情報処理への応用例として，音楽音響信号に対してリアルタイムにビートを認識するシステムの，並列計算機への実装について述べる．本システムでは質が異なる複数の処理を同時に実行する必要があるため，計算機全体でデータパラレル処理を行う実装は適切でない．本実装では，並列計算機を構成するプロセッサをグループに分け，各グループに対し，処理の質に応じた4種類の並列処理方式（パイプライン処理，データパラレル処理，コントロールパラレル処理，分散協調処理）を適用する．そしてこれらのグループ間でデータを流し，計算機全体ではパイプライン状にデータを処理する．本システムは実際に並列計算機AP1000上でリアルタイムに動作し，ロック・ポップス30曲中27曲に対してビートトラッキング可能であった．

Parallel Implementation of a Beat Tracking System — Real-time Musical Information Processing on AP1000 —

MASATAKA GOTO[†,††] and YOICHI MURAOKA[†]

This paper presents parallel implementation of a real-time beat tracking system that processes musical acoustic signals and recognizes temporal positions of musical-beats. Parallel processing is necessary to perform this computationally-intensive task in real time. We apply four kinds of parallelizing techniques to execute heterogeneous processes simultaneously. The processes are first pipelined, and then each stage of the pipeline is implemented with data/control parallel processing, pipeline processing and distributed cooperative processing. Our system has been implemented on the Fujitsu AP1000. In our experiment, it correctly tracked beats in 27 out of 30 popular songs.

1. はじめに

ビートトラッキングとは，人間が音楽に合わせて手拍子を打つように，曲のビート（4分音符）の位置を認識する技術である．ビートは西洋音楽を理解するうえで基本的な概念のひとつであり，ビートトラッキングの実現は，音楽聴取過程を計算機上で実現する第一段階として重要である．さらに，音楽音響信号からリアルタイムにビートトラッキングする技術は，多くのマルチメディアアプリケーションにおいても望まれている[1]．しかし，従来のビートトラッキングに関する研究の多くは，MIDI信号か少数の楽器で演奏された音響信号を対象にしており[2)~7)]，ドラムスを含む複数の楽器で演奏された音響信号を扱っていなかった．

本研究で実現するシステムは，ロック・ポップスの主にドラムスがビートを刻む曲を対象とし，複数の楽器で演奏された音響信号に対してリアルタイムにビートを認識・予測する．主要な処理は，複数の手がかりを抽出する周波数解析と，ビートの様々な解釈の可能性を同時に追跡するビート予測である．これらの処理は計算量が多いため，リアルタイムに実行するには並列処理するのが適している．

従来の並列計算機の応用例の多くでは，計算機全体でデータパラレル処理を行っていた．しかし，本研究が対象とするビートトラッキングのように，質が異なる複数の処理を同時に実行する必要がある場合には，この実装法は適切でない．

本実装では多様な処理を並列に実行するために，まず並列計算機を構成するプロセッサを複数のグループに分け，各グループに対してシステムを構成する処理単位を直接割り当てる．割り当てられた処理の質はそれぞれ異なるため，その質に応じた異なる並列処理方式（パイプライン処理，データパラレル処理，コント

† 早稲田大学 理工学部
School of Science and Engineering, Waseda University
†† 日本学術振興会 特別研究員
A Research Fellow of the Japan Society for the Promotion of Science

ロールパラレル処理，分散協調処理）を各グループ内で適用する．そして，これらのグループ間でデータを送受信することで，計算機全体としてはパイプライン状にデータを処理する．

我々は，本システムを富士通の分散メモリ型並列計算機 AP1000（64 プロセッサ構成）上に実装した．本実装では，あるプロセッサグループが同期をとっているときに，別のグループが処理を続ける必要がある．しかし，AP1000 の OS が提供する S-Net によるバリア同期は，すべてのプロセッサで明示的に同期をとる必要があるため利用できない．そこで，同期が不要なプロセッサは処理を続けることができるように，ステータスハードウェアを利用して時間確認可能な同期機構を実現した．

実装したシステムに対し，市販の Compact Disc (CD) の音響信号を入力して実験した結果，30 曲中 27 曲に対して正しいビートをリアルタイムにトラッキングできた．実装したシステムのターンアラウンドタイムは 127.71 msec であったが，つねに予測結果（次のビートの時刻）を出力するというビートトラッキングの処理の性質から，この遅延時間は問題にならなかった．

2. 処理モデル

本システムは，市販の CD などから得た複数の楽器音を含む音響信号を入力とし，4 分音符に相当するビートを認識する．その際，ビートの存在する時刻（ビート時刻）を認識するだけでなく，そのビートが強拍か弱拍か（ビートタイプ）も判断する．そして，ビート時刻，ビートタイプ，現在のテンポの 3 つから成るビート情報を，入力された音楽に合わせて出力する．

本システムは，入力に対して以下の仮定を行う．

- 入力曲は 4/4 拍子であり，テンポは 70〜180 M.M.（4 分音符/分）の間で，曲中を通じてほぼ一定である．
- バスドラム（BD）が強拍（1，3 拍目），スネアドラム（SD）が弱拍（2，4 拍目）で鳴る確率が高い．

これらの仮定は，ロック・ポップスの多くの曲に当てはまる．

このような音響信号に対するビートトラッキングを実現するには，主に次のような課題がある．(1) 音響信号波形にはビートに直接関係のないエネルギーピークが多数あるため，ピーク検出した後に閾値処理をする単純な手法では，ビートをとらえることはできない．(2) 対象とする音響信号には様々な楽器音が混在しており，MIDI の場合には容易に得られる各音の発音時刻を，正確に求めることは困難である．(3) ビートは音楽に対して人間が知覚する概念であり，ビートが実際の信号に直接対応するとは限らない．音響信号がないところにビートが存在する場合もある．このような曖昧さがあるためにビートを一意に解釈することはできず，つねに複数の解釈の可能性がある．(4) 4 分音符の長さやビートタイプの判断は音楽的知識を必要とし，一般には難しい．

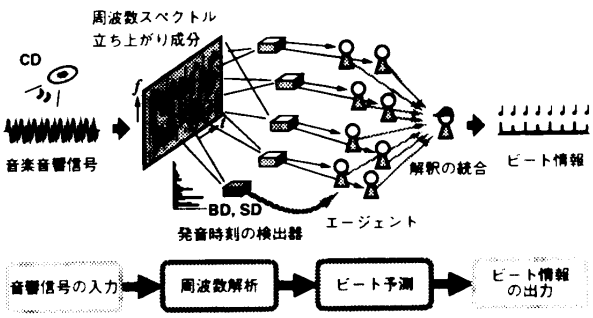

図 1 処理モデル
Fig. 1 Processing model.

以上を解決するために，複数の手がかりを抽出し，複数の解釈の可能性を同時に追跡する．まず，ビートに直接対応する特定の手がかりがないため，周波数解析で複数の手がかりを抽出する．具体的には，各周波数帯域ごとの発音時刻と，主要なドラム音（BD と SD）の発音時刻を検出する．次に，ビート予測において，複数のエージェントがビートの様々な解釈の可能性を並列に調べる．各エージェントは，発音時刻をそれぞれ異なった戦略により解釈し，次のビートの時刻を予測した後に，その解釈の確信度を自己評価する．最終的に出力するビートの時刻は，最も確信度の高いエージェントの解釈に基づいて決定する．これにより，あるエージェントの解釈がはずれても，他のエージェントが正しく解釈している限り，ビートを見失わずにトラッキングできる．また，ロック・ポップスに特化した音楽的知識として，BD と SD の出現位置に関する性質を用いることで，4 分音符の長さを推定しビートタイプを判断する．

本システムの処理モデルの概念図を**図 1** に示し，処理の流れを**図 2** に示す．まず，**音響信号の入力**で A/D 変換された音響信号に対して**周波数解析**を行い，各周波数帯域ごとの発音時刻と，BD と SD の発音時刻を検出する．これは様々な視点から検出する複数の発音時刻の検出器が行う．次に，**ビート予測**の各エージェントが，過去に得られた発音時刻からビートの間隔を求め，次のビート時刻の予測とビートタイプの判断を

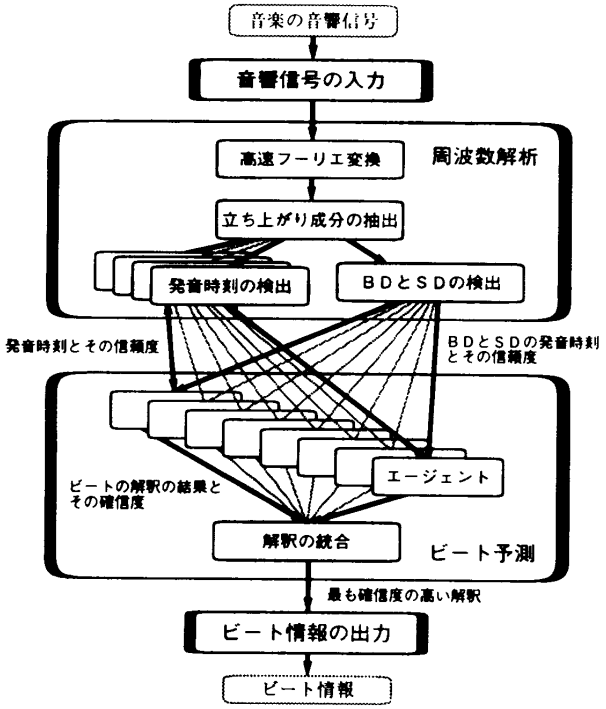

図2 処理の流れ
Fig. 2 Overview of our beat tracking system.

行う．そして，全エージェントの解釈を統合してビート情報を生成する．最後にビート情報の出力が，ネットワークを通じて他のアプリケーションプログラムへとビート情報を送信する．

以下では，主要な処理である周波数解析とビート予測について述べる．

2.1 周波数解析

複数の発音時刻の検出器が様々な視点から手がかりを抽出する．まず，高速フーリエ変換で得られた周波数スペクトルに対し，立ち上がり成分の抽出を行う．この結果に対し，発音時刻の検出器☆1が，様々な感度・周波数帯域において発音時刻を検出する．また，これ以外にBDとSD専用の発音時刻の検出器もある．

以上の発音時刻の検出結果は，ビート予測の各エージェントへと送られる．

2.1.1 高速フーリエ変換（FFT）

A/D変換された音響信号に対してFFTを行い，周波数スペクトル（パワースペクトル）を得る．これにより，各周波数成分のパワーの時間変化が得られる．観測区間（WindowSIZE）を時間軸方向に単位時間（ShiftSIZE）ずつずらしながらFFTを適用する☆2．

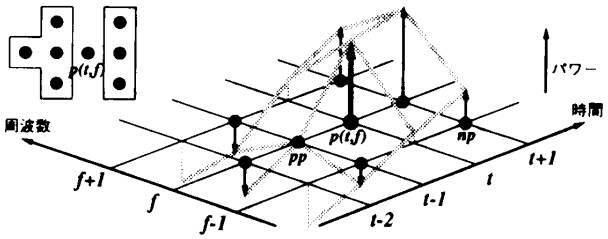

図3 立ち上がり成分の抽出
Fig. 3 Extracting an onset component.

2.1.2 立ち上がり成分の抽出

スペクトルのパワーが確実に増加し続けている周波数成分を，立ち上がり成分として抽出する．多くの場合，発音したときの鳴り始めの周波数成分がこの抽出で得られる．

FFTの結果として得られるスペクトル中の時刻t，周波数f☆3におけるパワーを$p(t,f)$としたとき，式(1)を満たす$p(t,f)$を立ち上がり成分と見なす（**図3**）．ただし，ppとnpは式(2)，式(3)で与えられるものとする．

$$\begin{cases} p(t,f) > pp \\ np > pp \end{cases} \tag{1}$$

$$pp = \max(p(t-1,f), p(t-1,f\pm1), p(t-2,f)) \tag{2}$$

$$np = \min(p(t+1,f), p(t+1,f\pm1)) \tag{3}$$

$p(t,f)$が立ち上がり成分であるとき，その立ち上がりの度合$d(t,f)$は，式(4)により求める．

$$d(t,f) = \max(p(t,f), p(t+1,f)) - pp \tag{4}$$

2.1.3 発音時刻の検出

複数の発音時刻の検出器が，それぞれ異なるパラメータにより発音時刻を求める．各検出器は，2.2節で述べるエージェントの組に対応しており，検出結果を対応する組へ送信する（図1，図5参照）．

発音時刻は以下のように得る．まず，各時刻における立ち上がり成分の合計値$D(t)$を，式(5)により求める．次に，その時間変化のピーク時刻とピーク値を，平滑化微分を用いたピーク検出により求める☆4．こうして得られたピーク時刻を発音時刻とする．

$$D(t) = \sum_f d(t,f) \tag{5}$$

それぞれの検出器は，検出感度，検出周波数帯域と

☆1 現在の実装では，発音時刻の検出器の数は15個である．

☆2 現在の実装では，WindowSIZEは1024点（46.44 msec），ShiftSIZEは256点（11.61 msec）であり，周波数分解能は21.53 Hz，時間分解能は11.61 msecになる．

☆3 ただし，t，fは整数値をとり，$1t$，$1f$はそれぞれ時間分解能，周波数分解能に相当する．

☆4 Savitzky と Golay の2次多項式適合による平滑化微分を用いたピーク検出により，ノイズの影響を平滑化で減らした後に極大値を与える時刻を検出する．

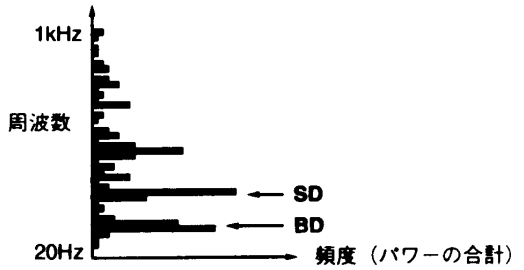

図4 ヒストグラム中のBDとSDの特徴周波数
Fig. 4 The characteristic frequencies of BD and SD.

いう2つのパラメータを持つ．検出感度は，平滑化微分における平滑化の時間幅であり，平滑化幅が小さいほど感度が高くなる．検出周波数帯域は，式(5)中の$\sum$における周波数帯域の指定で，これにより制限された帯域の発音時刻を得ることができる．

2.1.4 BDとSDの検出

立ち上がり成分の抽出結果から，現在の曲のBDとSDの音に対応する特徴周波数を自動的に獲得し☆，これらの発音時刻を検出する．立ち上がり成分が検出された時刻において，その周波数軸方向のピーク周波数がBDかSDの特徴周波数に一致した場合，BDまたはSDがその時刻に発音したと判断する．

特徴周波数は以下のように獲得する．まず，立ち上がり成分が存在する時刻において，周波数軸方向のピークを求め，そのヒストグラムを作成する（図4）．その際，ヒストグラムには各ピークの立ち上がりの度合$d(t,f)$を加える．そして，ヒストグラム中で最も周波数の低いピークをBD，それより周波数が高く最もパワーの大きいピークをSDの特徴周波数と見なす．

2.2 ビート予測

周波数解析の結果を，複数のエージェントがそれぞれ異なった戦略により解釈し，次のビート時刻を予測する．各エージェントが出力する解釈の結果は，次のビート時刻，そのビートタイプ，ビートの間隔の3つから成る（図5）．この結果は，解釈の統合へと集められ，最も確信度の高い解釈が選択される．

すべてのエージェント☆☆は，2つずつの組に分けられる．同じ組の2つのエージェントは協調し合いながら，両者が同じビートの間隔で，互いにビートの間隔の1/2ずれた時刻を予測する．これにより，一方が4分音符の裏拍を予測し始めても，他方が正しい表拍を予測できる．各エージェントの組は，2.1.3項で述べたように，それぞれ対応する検出器から発音時刻を受けとる（図5）．

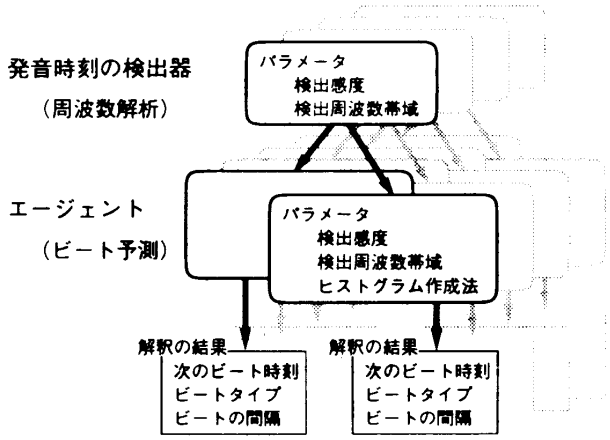

図5 発音時刻の検出器とエージェント
Fig. 5 Onset-time finders and agents.

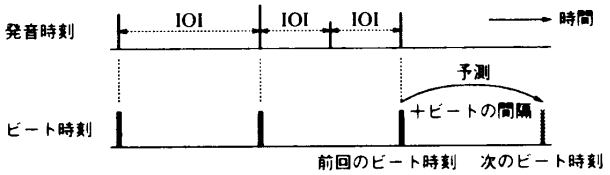

図6 次のビート時刻の予測
Fig. 6 Beat prediction.

エージェントは，ビートを解釈する際の戦略を変更する3つのパラメータ（検出感度，検出周波数帯域，ヒストグラム作成法）を持つ．パラメータの値の設定は，各エージェントの組ごとに異なり，同じ組の2つのエージェントだけが同じ値をとる．検出感度と検出周波数帯域は，発音時刻の検出器内の対応するパラメータを制御し，エージェントが受けとるビートの手がかりの質を調整する．ヒストグラム作成法は，発音時刻の間隔（IOI: inter-onset-interval）のヒストグラムを作成する方法を変更する．

エージェントは自己評価した確信度が長期間低い場合に，これらのパラメータを，他のエージェントの現在のパラメータの状態を考慮して自己修正する．具体的には，他のエージェントのパラメータとは異なる値で，最も確信度の高いエージェントのパラメータに近付くように，パラメータの値を調整する．

2.2.1 エージェントの動作

各エージェントは発音時刻からビートの間隔を求めて次のビート時刻を予測する．そして，予測したビートに対してビートタイプを判断し，その解釈の確信度を自己評価する．

次のビート時刻の予測は，前回のビート時刻にビートの間隔を加えて行う（図6）．ビートの間隔は，IOIのヒストグラムの最大ピークから求める．もし前回の

☆ 一般に，これらのドラム音の音色は曲ごとに異なるため，特徴周波数を事前に与えることはできない．
☆☆ 現在の実装では，エージェントの数は30個である．

ビート時刻とほぼ一致する発音時刻がある場合には，その発音時刻にビートの間隔を加える．

次に，予測した各ビート時刻に対し，BD と SD の発音時刻の検出結果からビートタイプを判断する．ロック・ポップスの多くの曲ではこれらがある程度交互に鳴るという性質を用いて，次のビートに BD と SD のどちらが予測されるかを判断する．もし次のビートが BD だと予測すれば，そのビートタイプを強拍とし，SD の場合には弱拍とする．

各エージェントは解釈の確信度を，1) ビート時刻と発音時刻との一致の度合と，2) ビートタイプの規則性の度合の両者に基づいて自己評価する．これは，それぞれの評価値に基づいて確信度を増減することで行う．1) の評価では，過去に予測したビート時刻に発音時刻が一致していれば確信度を上げ，一致していなければ確信度を下げる．ただし，ビート時刻の間の 8 分音符や 16 分音符に相当する時刻に発音時刻が一致している場合には，確信度を少し上げる．一方 2) の評価では，ビートタイプとして強拍と弱拍が交互に規則正しく検出されるほど確信度を上げる．これにより，4 分音符に相当する適切なビートの間隔を持つ解釈が選択されやすくなる．

2.2.2 解釈の統合

すべてのエージェントから集められた解釈の結果は，ビート時刻とビートの間隔が同じものどうし，グループ分けされる．各グループにおいてグループ内の全解釈の確信度を合計し，そのグループの確信度とする．そして，最も確信度の高いグループ内の最も確信度の高い解釈を選択する．

3. 並列化手法

2 章で述べた処理モデルから，質が異なる多様な処理を，並列計算機上でリアルタイムに実行する必要があることが分かる．このような処理を並列化するために，計算機全体で時分割処理するのではなく，プロセッサを図 2 の各処理ごとのグループに分けて，計算機全体でパイプライン処理する．その際，各グループはパイプラインの各ステージに対応し，グループ間でデータを流すことでパイプラインを形成する．各グループへその処理の並列度に応じてプロセッサを割り当てることで，アイドル状態となるプロセッサを削減できる．各プロセッサは，つねに割り当てられた同種の処理を実行するので，処理の切替えのオーバヘッドも小さい．

本システムの並列化では計算機全体のパイプライン処理に加え，さらに 4 種類の異なる並列処理方式を組み合わせる．パイプライン中の 4 つのステージにおいて，それぞれの処理の質に応じてパイプライン処理，データパラレル処理，コントロールパラレル処理，分散協調処理を適用する（**図 7**）．

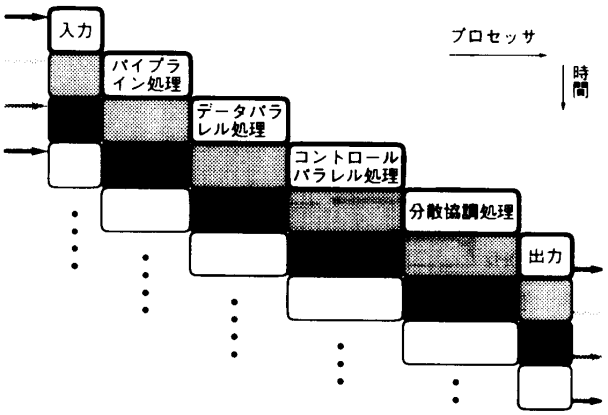

図 7 異なる並列処理方式を組み合わせたパイプライン処理
Fig. 7 Pipeline processing with various parallelizing techniques.

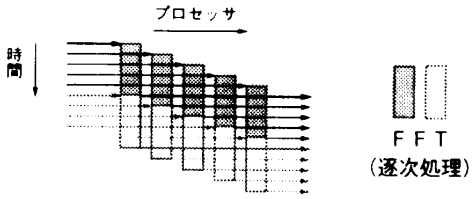

図 8 パイプライン処理（5 プロセッサ）
Fig. 8 Pipeline processing (5 processors).

以下では，これら 4 つの並列処理方式について順番に述べる．なお，現在の実装は並列計算機 AP1000 に対して行っており，各ステージのプロセッサ数は，総プロセッサ数（64 台）を考慮して決められている．

3.1 パイプライン処理（FFT）

FFT（2.1.1 項）を 5 台のプロセッサがパイプライン処理する．FFT 自体は逐次処理で行うが，各プロセッサが単位時間（ft: フレーム時間）ずつずれて FFT を実行することで，パイプライン状に処理する．現在の実装では 1 ft は 11.61 msec であり，1 回の FFT（1024 点）の計算はおよそ 4.2 ft かかる．そこで，5 台のプロセッサが FFT を行うことにより，毎 ft ごとに結果を得ることができる（**図 8**）．

3.2 データパラレル処理（立ち上がり成分の抽出）

立ち上がり成分の抽出（2.1.2 項）を 8 台のプロセッサがデータパラレル処理する．周波数スペクトルの全周波数帯域を 8 等分してプロセッサへ割り当てる．そして，各プロセッサは自分の担当する帯域の立ち上がり成分を抽出する（**図 9**）．この処理は DOALL 型ループで実現でき，全プロセッサでデータパラレル処理す

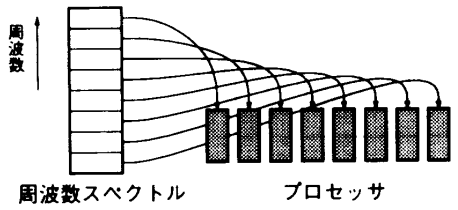

図9 データパラレル処理（8プロセッサ）
Fig. 9 Data parallel processing (8 processors).

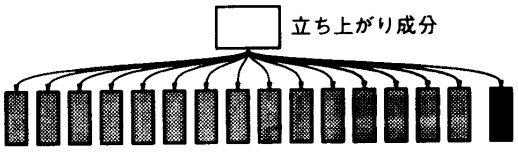

図10 コントロールパラレル処理（16プロセッサ）
Fig. 10 Control parallel processing (16 processors).

ることも可能である．今回は1ft以内に処理が終了すれば十分であるので，8台のプロセッサで並列化した．

3.3 コントロールパラレル処理（発音時刻の検出・BDとSDの検出）

発音時刻の検出（2.1.3項）およびBDとSDの検出（2.1.4項）を，16台のプロセッサがコントロールパラレル処理する．実装した処理モデルでは，15個の通常の発音時刻の検出器と1個のBDとSD専用の検出器があり，これらを各プロセッサに1つずつ割り当てる（図10）．そして，立ち上がり成分の抽出結果という同一のデータに対し，16台のプロセッサが，異なるアルゴリズムやパラメータで別々に発音時刻を検出する．この各プロセッサの処理は1ft以内に終了する．

3.4 分散協調処理（エージェント・解釈の統合）

ビート予測のエージェント（2.2.1項）および解釈の統合（2.2.2項）を，31台のプロセッサが分散協調処理する．実装した処理モデルでは，30個のエージェントと1個の解釈の統合処理があり，これらを各プロセッサに1つずつ割り当てる（図11）．プロセッサ間でエージェントの解釈の結果や現在のパラメータの状態を通信し合い，協調して処理を進める．

4. AP1000への実装

3章で述べた並列化手法により，本システムを富士通の分散メモリ型並列計算機AP1000上に実装した（図12）．実装したAP1000は，64台のセルと呼ばれる要素プロセッサで構成される．本実装では，これらのセルを図12のように7つのグループに分けた．これらのグループが，計算機全体のパイプラインを構成する．グループを表す長方形の右下に，そのグループに割り当てられたセルの数を示す．矢印はグループ間の大局的なデータの通信を表しており，主に入力から出力へとU字型にデータが流れる．実際の通信量を，矢印の横に示す．

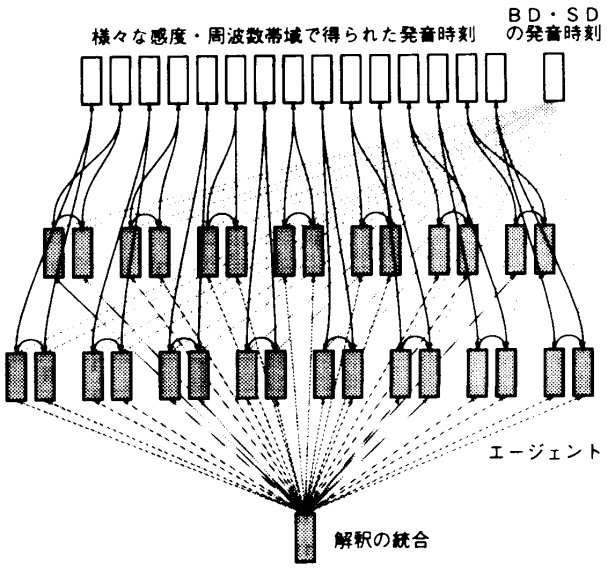

図11 分散協調処理（31プロセッサ）
Fig. 11 Distributed cooperative processing (31 processors).

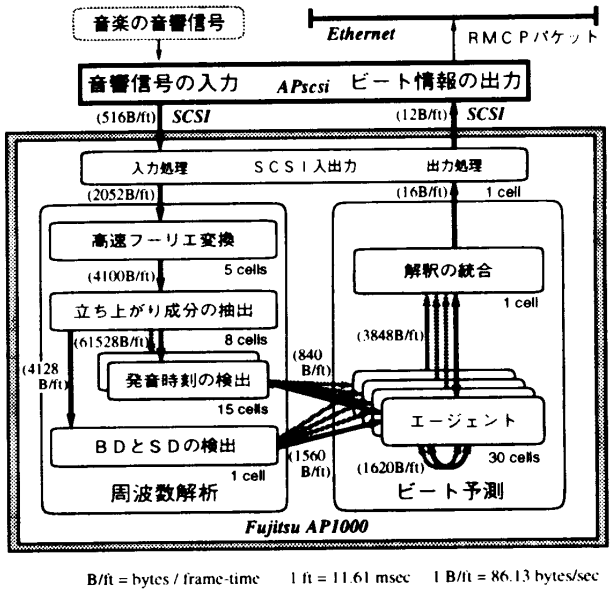

図12 AP1000への実装
Fig. 12 Implementation on the AP1000.

4.1 入出力

音響信号の入力とビート情報の出力は，セルの1つとSCSIによって接続された入出力用ワークステーション（Sun SPARC Station 10: 以下APscsiと呼ぶ）により行う（図12）．このSCSI接続を行うために，セルの1つにSCSIインタフェースを持つオプションハー

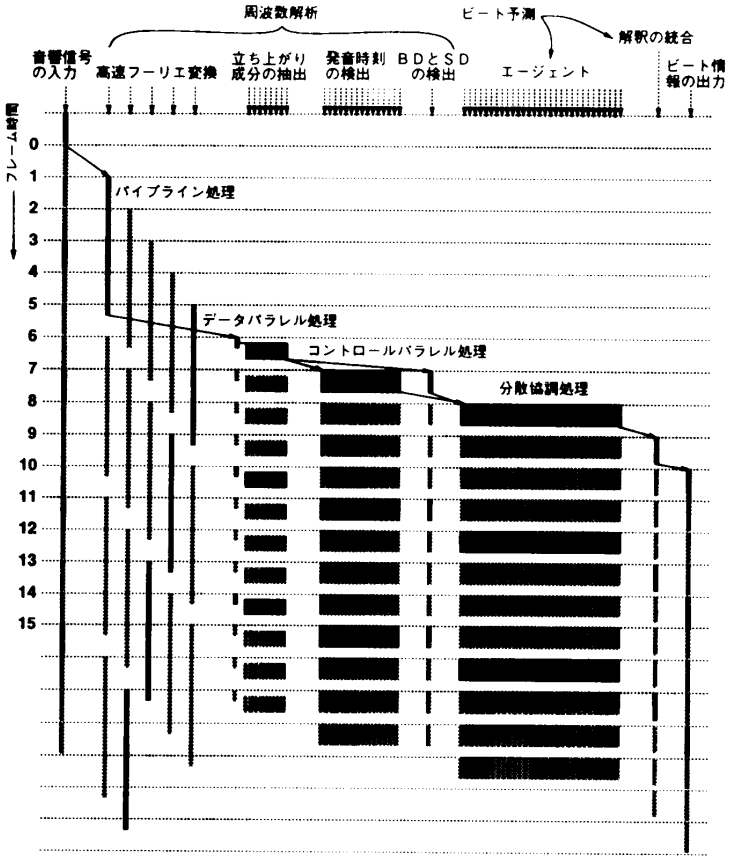

図13 タイミングチャート
Fig. 13 Timing chart.

ドウェア（富士通DDVボード）を追加した．

4.1.1 音響信号の入力

APscsiは音響信号を22.05 kHz，16 bit，モノラル（1 channel）でA/D変換する．この結果を256点ごとにブロック化して，SCSI入出力を行うセルへ送信する．1 ftはこの1ブロックの時間に相当する．

4.1.2 ビート情報の出力

APscsiは，SCSI入出力を行うセルからビート情報を受信し，RMCP（Remote Music Control Protocol）パケットとして，予測されたビート時刻にEthernet上へ送信（ブロードキャスト）する．RMCPは，MIDIとLANを融合した分散協調システム（RMCPシステム）におけるサーバ・クライアント間の通信プロトコルであり，シンボル化された音楽情報をネットワークを通じて伝送するために設計された[8]．

RMCPパケットとしてブロードキャストすることにより，アプリケーションの実装が容易になるだけでなく，複数のアプリケーションでビート情報を同時に利用できる．すでに，音楽に合わせてドラム音や手拍子の音を出力したり，音楽に同期したCG（コンピュータグラフィックス）を生成する数種類のRMCPサーバを実装した．

4.2 時間管理のための同期機構

計算機全体のパイプライン処理と主要なデータ通信のタイミングを**図13**に示す．この時間管理は，APscsiがA/D変換する際に，一定個数（256点）のデータが揃うまでブロックすることを利用して行う．AP1000側は，APscsiから転送されてきたデータによるデータ駆動方式で動作し，1ブロックを受信するたびに同期をとる（11.61 msec間隔）．図13の各横線が同期をとる時刻を表す．この同期は，APscsiからデータを受信する時刻に，各セルの処理のタイミングを合わせるために行う．

本実装では，あるプロセッサグループが同期をとっているときに，同期が不要な別のグループは処理を続ける必要がある．しかし，AP1000のOSが提供するS-Netによるバリア同期は，すべてのセルで明示的に同期をとる必要があるため利用できない．そこで，ステータスハードウェアを利用して時間確認可能な同期機構を実現する．

S-Netは，バリア同期を行う同期ハードウェア以外にステータスハードウェアを備えている．これは，全セルからの8ビットのステータスの論理和をハードウェアで求め，その結果を各セルが参照できる機構で

ある．そこで，この8ビットのうち上位3ビットを本来のステータス用（0～7），下位5ビットを同期用ステータス（0～31）として割り当てる．

同期は以下のように行う．SCSI入出力を行うセルがマスターとなり，APscsiからデータを受信するたびに同期用ステータスをインクリメントする．他のすべてのセルはスレーブとなり，同期用ステータスの変化を検出して同期を行う．こうして同期をとる必要のあるセルだけが変化を監視して同期し，必要のないセルはそのまま処理を続けることができる．たとえばFFTを行うセルは，同期用ステータスを5ftずつ飛ばして確認する．

さらに本同期機構では，同期用ステータスの値が予期していた値よりも進んでいると，自分の処理が遅れていることが分かる．しかし，現在の実装では，AP1000の処理能力は本システムの計算負荷よりも十分に高く，正常動作しているときに処理が遅れることはない．そのため，処理が通常より遅れた場合には，異常が起きたと判断してシステムは停止する．

4.3 ターンアラウンドタイム

現在の実装では，AP1000全体のパイプラインをデータが通過する時間（ターンアラウンドタイム）は11ft（127.71 msec）である．つまり，ある時点の出力結果には，最短で127.71 msec前の入力までが反映される．この遅延時間は，入力を直接加工して音響信号を出力する処理（リバーブなどのエフェクト処理）では問題になるが，本システムのように過去のビートを認識して，つねに予測結果（次のビート時刻）を出力する場合には問題にならない．たとえば，180 M.M.の最も短い4分音符の長さでも333.33 msecあるため，予測時に前回のビート時刻周辺の情報は利用できる．

ただし，これはターンアラウンドタイムが333.33 msecでもよいことを意味するのではない．たとえば，発音時刻を検出するには，その時刻の前後100～200 msecの情報を平滑化微分の処理が必要とする．また，ターンアラウンドタイムが短いほど，より新しい情報を利用して予測が行える．

5. 実験結果

本システムが入力に対して仮定した条件を満たす曲に対する，システムの動作を確認する実験を行った．実験では，市販のCDからサンプリングしたロック・ポップス30曲のモノラルの音響信号を入力した．これらの曲はテンポが78～168 M.M.と広い範囲から選び，21の異なるアーティストによって演奏されたものである．

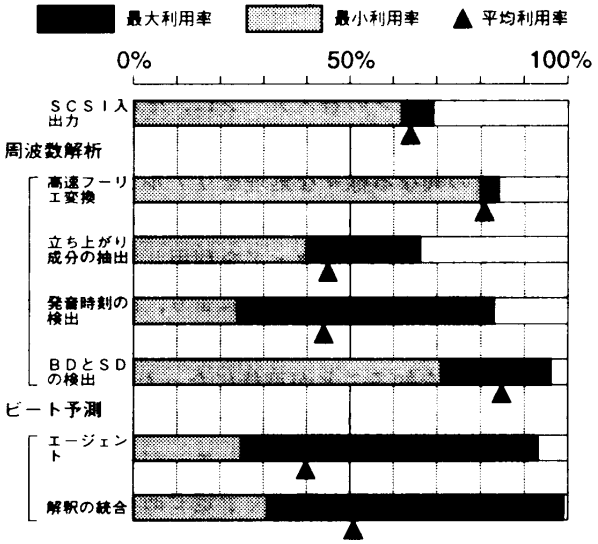

図14 セル利用率
Fig. 14 Cell utility.

実験の結果，30曲中27曲に対して正しいビートをリアルタイムに出力できた．本実験では，BDとSDがほぼ定常的に鳴り出してから6小節以内に，適切なビート時刻とビートタイプを出力した曲を正しいと判定した．ただしビート時刻は，システムの出力結果を人間が知覚して適切かどうかの判定をし，定量的には，人間が手作業で調査したビート時刻から2ftの誤差範囲内のものを適切であるとした．

誤認識した3曲では，ビート時刻は適切に得られていたが，ビートタイプを誤っていた．これは，BDとSDの特徴周波数を正しく獲得できなかったためか，ドラムスの変則的なリズムが一時期続いていたのが原因であった．

本実験におけるセル利用率の変動を**図14**に示す．図中には，各処理の最大利用率，最小利用率，平均利用率が示されている．これから，曲の状況などによって利用率が大きく変動していることが分かる．また，本実装をプラットフォームとして今後システムを拡張していく予定であるが，利用率の面からもその余地が残されていることが分かる．一般に並列処理研究の立場からは，利用率が高いほど望ましいと考えられるが，本研究のような応用の立場からは，今後の拡張のために利用率が飽和していないことが望ましい．

6. おわりに

本稿では，音楽音響信号に対するビートトラッキングシステムの並列化手法と，並列計算機AP1000への実装について述べた．本システムでは，並列処理することにより，複数の手がかりをリアルタイムに抽出し，ビートの様々な解釈の可能性を同時に追跡できる

ようになった.

本実装では，並列計算機全体でパイプライン処理を行い，パイプラインの各ステージに，それぞれの処理の質に応じた4種類の並列処理方式を適用した．これにより，質の異なる様々な処理をリアルタイムに実行することが可能になった．また，AP1000に実装する際に，ステータスハードウェアを利用して時間確認可能な同期機構を実現した．これにより，同期が不要なプロセッサは処理を続けることができ，各プロセッサは処理の遅れを知ることが可能になった．

今後は，ドラム音の検出手法や予測処理の改良により入力の制約を減らすとともに，エージェント間の協調・統合の方式も改良していく予定である．またビートトラッキングの技術は，ビデオ/オーディオ編集システムやリアルタイムCGなど，多くのマルチメディアアプリケーションに有効である．我々はすでに本システムを応用して，音楽のビートに合わせて踊る仮想のCGダンサーをリアルタイムに表示するシステムを実現したが[1]，今後は他のマルチメディアシステムへの応用も行っていきたい．

謝辞 AP1000のSCSI入出力の実装について協力していただいた富士通研究所の稲野 聡氏に深く感謝いたします．また，AP1000の実行環境を提供していただいた富士通研究所 並列処理研究センターに感謝いたします．

参考文献

1) Goto, M. and Muraoka, Y.: A Beat Tracking System for Acoustic Signals of Music, *Proc. Second ACM International Conference on Multimedia*, pp.365–372 (1994).
2) Dannenberg, R.B. and Mont-Reynaud, B.: Following an Improvisation in Real Time, *Proc. 1987 International Computer Music Conference*, pp.241–248 (1987).
3) Allen, P.E. and Dannenberg, R.B.: Tracking Musical Beats in Real Time, *Proc. 1990 International Computer Music Conference*, pp.140–143 (1990).
4) Rosenthal, D.: Machine Rhythm: Computer Emulation of Human Rhythm Perception, Ph.D. Thesis, Massachusetts Institute of Technology (1992).
5) Desain, P. and Honing, H.: Advanced Issues in Beat Induction Modeling: Syncopation, Tempo and Timing, *Proc. 1994 International Computer Music Conference*, pp.92–94 (1994).
6) Schloss, W.A.: On The Automatic Transcription of Percussive Music—From Acoustic Signal to High-Level Analysis, Ph.D. Thesis, CCRMA, Stanford University (1985).
7) 片寄晴弘：音楽感性情報処理に関する研究，博士論文，大阪大学 基礎工学部 (1991).
8) 後藤真孝，橋本裕司：MIDI制御のための分散協調システム—遠隔地間の合奏を目指して—，情処研報音楽情報科学，93-MUS-4-1, Vol.93, No.109, pp.1–8 (1993).

(平成7年7月13日受付)
(平成8年3月12日採録)

後藤 真孝（学生会員）

1993年早稲田大学理工学部電子通信学科卒業．現在同大学院博士後期課程在学中．日本学術振興会特別研究員．音楽情報処理，並列処理，インタラクティブシステムなどに興味を持つ．1992年jus設立10周年記念UNIX国際シンポジウム論文賞受賞．1993年NICOGRAPH'93 CG教育シンポジウム最優秀賞受賞．電子情報通信学会，人工知能学会，日本ソフトウェア科学会，日本音楽知覚認知学会，日本神経回路学会，ICMA各会員．

村岡 洋一（正会員）

1965年早稲田大学理工学部電気通信学科卒業．1971年イリノイ大学電子計算機学科博士課程修了．Ph.D.この間，Illiac-IVプロジェクトで並列処理ソフトウェアの研究に従事．同学科助手ののち，日本電信電話公社（現NTT）電気通信研究所に入所．1985年より早稲田大学理工学部教授．現在同大学情報科学研究教育センター所長，図書館副館長．IEEE Computer Society Computational Science and Engineering Magazine, International Journal of High Performance Computing, Parallel Processing Letter誌などのEditor．並列処理，マンマシンインタフェースなどに興味を持つ．「コンピュータアーキテクチャ」（近代科学社）など著書多数．