マルチコアプロセッサ上でのハーモニッククラスタリングを用いた基本周波数解析の並列処理

Parallel Fundamental Frequency Analysis Using Harmonic Clustering on Multicore Processors

鈴木 涼介† Ryosuke Suzuki
吉田 明正† Akimasa Yoshida

1 はじめに

本稿では，ピアノ音等のWAVEデータに対して短時間フーリエ変換(STFT)による周波数解析を行い，倍音成分を考慮して基本周波数を決定し[1, 2]，音高データを出力するシステムの並列化手法を提案する．本システムでは，ハーモニッククラスタリング手法を用いて，倍音成分を含む音響信号から基本周波数を正確に求めているが，計算量が極めて大きいため並列処理による高速化が必要不可欠となっている．そこで，本稿では，Javaマルチスレッドを用いて，ハーモニッククラスタリングの並列化を実現し，マルチコアプロセッサ上で並列処理による性能評価を行っている．

2 基本周波数解析システムの構成

本システムの構成を図1に示す．まず，入力音響信号を短時間フーリエ変換により周波数変換する．次に周波数成分のクラスタリングを行い基本周波数を推定した後，最後に音高を抽出する．最も計算量の多い箇所であるハーモニッククラスタリングにはマルチコアプロセッサ上で並列処理を行い高速化を実現する．

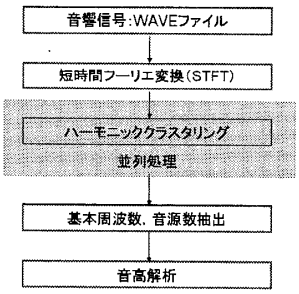

図1 基本周波数解析の手順.

3 ハーモニッククラスタリングとAIC

基本周波数解析にはフーリエ変換の他に亀岡らによるハーモニッククラスタリング(Harmonic Clustering)[3, 4]を用いる．これは窓関数や分析区間内におけるピッチ変化の影響により左右に広がったスペクトルが観測され，この周波数軸上に広がったスペクトルの形状を正規分布で最適近似するものである．音源数を判定するには，AIC（赤池情報量基準）を導入し最適なモデルの選択基準とする．

3.1 ハーモニッククラスタリングの定式化

複数（K個）のクラスタ群を用いてEMアルゴリズムにより解く．倍音クラスタ群kのn倍音に対する重心を$\mu_k+\log n$とし，クラスタ群kにおいて上限（ナイキスト周波数の対数）まで取りえるクラスタ重心数を$N(k)$とする．また，クラスタごとの重みをω_n^kとし，クラスタ帰属度$p_n^k(n)$と，平均$\mu_k+\log n$，分散σとした正規分布$g(x|\mu_k+\log n,\sigma^2)$を以下のように定義する．

$$p_n^k(x)=\frac{\omega_n^k\cdot g(x|\mu_k+\log n,\sigma^2)}{\sum_{k=1}^{K}\sum_{n=1}^{N(k)}\omega_n^k\cdot g(x|\mu_k+\log n,\sigma^2)} \tag{1}$$

$$g(x|\mu_k+\log n,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{\{x-(\mu_k+\log n)\}^2}{2\sigma^2}\right\} \tag{2}$$

さらに，距離関数$\varphi(x,\mu_k+\log n)$を

$$\varphi(x,\mu_k+\log n)=\log\frac{\omega_n^k}{\sqrt{2\pi\sigma^2}}-\frac{\{x-(\mu_k+\log n)\}^2}{2\sigma^2} \tag{3}$$

と定義することにより，クラスタリング評価関数

$$D=\sum_{k=1}^{K}\sum_{n=1}^{N(k)}\int_{-\infty}^{\infty}\omega_n^k\cdot\varphi(x,\mu_k+\log n)\cdot p_n^k(x)\cdot f(x)dx \tag{4}$$

を得る．この評価関数はEMアルゴリズムによる混合正規分布のパラメータの最尤推定におけるQ関数と同値となるため，この評価関数を局所最大化するμ_k，ω_n^kを求めることにより，基本周波数が解析される．

3.2 音源数推定AIC

まず，同時発音されるピッチ数は12個以下として12個のクラスタ群を用意する．EMアルゴリズムによりモデルパラメータの最尤推定値を求め，AICを算出する．AICは$-2\times$(モデルの最大対数尤度)$+2\times$(モデルの自由パラメータ数)で与えられ，このAICが最小となるモデルを最良のモデルと考える．

3.3 音源数とクラスタ重心μ_kの導出

前節までに述べたハーモニッククラスタリングとAICを用いて，音源数とクラスタ重心を導出する手順を以下に示す．

Step1 あるオクターブ範囲内における12平均律音階の基本周波数数値を基本クラスタ重心とする．つまり初期のKは12となる．

Step2 以下の反復計算により最尤パラメータを求める．但し，$F=\int_{-\infty}^{\infty}f(x)dx$とする．

$$\overline{\mu}_k=\frac{\sum_{n=1}^{N(k)}\int_{-\infty}^{\infty}(x-\log n)p_n^k(x)f(x)dx}{\sum_{n=1}^{N(k)}\int_{-\infty}^{\infty}p_n^k(x)f(x)dx} \tag{5}$$

$$\overline{\omega}^k=\frac{1}{FN(k)}\sum_{n=1}^{N(k)}\int_{-\infty}^{\infty}p_n^k(x)dx \tag{6}$$

†東邦大学理学部情報科学科
Department of Information Science, Toho University

Step3 AIC を算出する．AIC が上昇した時点で終了する．

Step4 倍音クラスタ群を順次削除し，残った K を新たな K とし **Step2** に戻る．

4 マルチコアプロセッサ上での並列処理

前章の音源数とクラスタ重心 μ_k を導出する際に，AIC が最小となるまで **Step2** の反復計算部分を繰り返し行わなければならないので計算量がとても多くなってしまう．そこで **Step2** の計算を並列化し高速化を図る手法を本章で述べる．

4.1 マルチコアプロセッサ上で並列化

マルチコアプロセッサ上で並列化する場合，コア数と同数のスレッドを用意する．ループ並列処理やリダクション処理 [6] を行うときには，必要に応じてスレッドごとのローカル変数や，全スレッドにおける共有変数を用いる．本手法では μ_k+log$n$，ω_n^k，x，$f(x)$ を共有変数とし，その他をローカル変数とした．また，全スレッドでの同期が必要となる箇所，例えば図 2 のように異なる計算（計算 A と計算 B）の間では，バリア同期を導入している．バリア同期を導入することにより，スレッド生成は各コアで 1 回にしており，スレッド生成のオーバーヘッドが軽減できる．

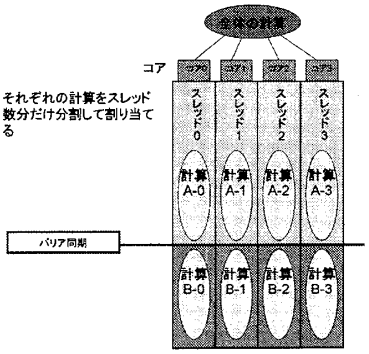

図 2 マルチコアプロセッサ上での同期．

4.2 Java マルチスレッドによるバリア同期の実装

本手法では，スレッド間でバリア同期を取る際には synchronized メソッドを用い，synchronized メソッドの中で wait() メソッド，notifyAll() メソッドを導入する [5]．wait() メソッドとはスレッドをウェイトセットに入れるメソッドで，notifyAll() メソッドとはウェイトセットにいる全てのスレッドをウェイトセットから取り出すメソッドである．これらのメソッドを使用し，他のスレッドの計算が終了するまでウェイトセットに入り，最後のスレッドがウェイトセットに入ったら全てのスレッドを起こす動作が実現される．

4.3 ハーモニッククラスタリングの Java 実装

本節では，第 3 章および第 4 章のハーモニッククラスタリングの Java マルチスレッド実装について述べる．入力音響信号をフーリエ変換し，周波数変換された離散化数値を得る．この得られた数値が前章の式の x となる．次に μ_k，ω_n^k に初期値を与える．今回の性能評価では，A3 を一番低い音とした 12 平均律音階を初期値とする．つまり，$\mu_k = 220 \times 2^{\frac{k}{12}}$ となる．**Step2** の式 (5)，式 (6) の反復計算を行う．ここで，4.1 節で述べたスレッド

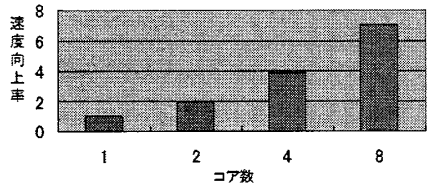

図 3 ハーモニッククラスタリングの並列処理．

をコア数分だけ作成し，クラスタ帯域をコア数に分割する．式 (2)，式 (3)，および式 (1) の分子の計算ではループ並列処理を行い，総和計算や積分計算が含まれている式 (1) の分母，式 (4)，式 (5)，式 (6) ではリダクション処理を行い，バリア同期コードを伴う Java マルチスレッドで実装した．

5 マルチコアプロセッサ Xeon 上での性能評価

本性能評価では，Intel Xeon クアッドコアプロセッサ（E5430）2 台からなる DELL PowerEdge 2900 III（計 8 コア）を用いる．各プロセッサは 2.66GHz の動作周波数で，6MB の L2 Cache を 2 つ搭載しており，メモリは 8GB，OS は Red Had Enterprize Linux5，Java 処理系は JDK1.6 となっている．

作成したシステムのコードは javac でコンパイルし，PowerEdge 2900 III の JVM 上で実行した．JVM では -Xint オプションをつけ，JIT コンパイルは適用していない．入力信号はモノラル音響信号 (C4-E4-G4) を用い，スペクトル解析はサンプリング周波数 44.1kHz，フレーム長を 93ms，フレームシフト 6ms とし，窓関数は Hamming 窓関数を用いて STFT を行う．

並列処理による実行結果は，図 3 に示すように 4 コアの速度向上率は 3.85 倍，8 コアでは 7.03 倍となった．それゆえ，ハーモニッククラスタリングに対して高速化が図られており本手法の有効性が確かめられた．

6 おわりに

本稿では，マルチコアプロセッサ上でのハーモニッククラスタリングを用いた基本周波数解析の並列処理手法を提案した．本手法では，ハーモニッククラスタリングを用いた基本周波数解析に Java マルチスレッドによる並列処理を適用しており，高速処理が可能となる．本手法では，機種依存性のない Java による実装を行ったが，今後は OpenMP による実装を行うことでキャッシュ最適化による更なる速度向上が期待できる．

参考文献

[1] 宮崎洋光，永井啓之亮，水谷孝一．楽器演奏音からの音調抽出．日本音響学会音楽音響研究会資料，MA98-16，pp.9-14，1998.

[2] Curtis Roads，青柳龍也，小阪直敏，平田圭二，堀内靖雄．コンピュータ音楽．東京電機大学出版局，2001.

[3] 亀岡弘和，西本卓也，嵯峨山茂樹．ハーモニッククラスタリングによる多重音信号音高抽出における音源数とオクターブ位置推定．情報処理学会論文誌，No.48，pp.27-32，2003.

[4] 亀岡弘和，西本卓也，篠田浩一，嵯峨山茂樹．ハーモニッククラスタリングによる多重音の基本周波数推定．情報処理学会論文誌，No.82，pp.29-34，2003.

[5] 結城浩．Java 言語で学ぶデザインパターン入門．ソフトバンクパブリッシング，2006.

[6] Wolfe, M. High performance compilers for parallel computing. Addison-Wesley Publishing Company, 1996.