3K-3

GPGPUによるユーザベース協調フィルタリングの高速化

†常盤 俊充　　†小柳 滋
†立命館大学 情報理工学部

1 はじめに

近年, ビッグデータの効率的な利活用に関する研究開発が盛んに行われている. そのような中, 従来までの処理能力に限界のあるCPUを用いた処理だけでは, 大量のデータを処理する際に時間がかかるという問題が存在する. そこで、画像処理専用プロセッサであるGPUを画像処理以外の汎用計算目的で用いるGPGPUという概念が生まれる.

また, Amazonに代表される巨大なECサイトでは協調フィルタリングと呼ばれるレコメンド技術を用いてビッグデータを処理し, ユーザに商品を推薦することを行っている.

本研究では, 協調フィルタリング技術の1つである, ユーザベース協調フィルタリングをGPUを用いて並列化することで処理の高速化を図り, ユーザベース協調フィルタリングの性能の向上に努める.

2 ユーザベース協調フィルタリング

本章では, 協調フィルタリングの中でも最も古典的とされるユーザベース協調フィルタリングについて述べる. ユーザベース協調フィルタリングは, ユーザの類似性に基づいてアクティブユーザ (推薦対象ユーザ) に対して推薦するアイテムを決定する手法のことである. ユーザベース協調フィルタリングのアルゴリズムは以下の2.1〜2.5のようになる.

2.1 アクティブユーザとの類似度計算

ユーザの類似性を算出する指標として類似度を用いる. 類似度の算出方法として, 今回は最も一般的な手法として知られているピアソン相関係数を用いる.

全ユーザ集合をU, 全アイテム集合をIとし, $x \in U, y \in U, i \in I$の関係があると仮定する. この時, ユーザ$x$とユーザ$y$の類似度$sim_{x,y}$を表す計算式は以下の(1)のようになる.

$$sim_{x,y} = \frac{\sum_{i \in I(x) \wedge I(y)} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i \in I(x) \wedge I(y)} (x_i - \overline{x})^2 \sum_{i \in I(x) \wedge I(y)} (y_i - \overline{y})^2}} \quad (1)$$

ここで, $I(x)$はユーザxが保持している全アイテム情報, $i \in I(x) \wedge I(y)$はユーザxとユーザyが共通して保持しているアイテムiを示す. また, x_iはあるユーザxのアイテムiに対する評価, $\overline{x}$はあるユーザxの評価の平均値を示す. 類似度$sim_{x,y}$がとる値の範囲は$-1 \leq sim_{x,y} \leq 1$である.

2.2 高類似度ユーザ抽出

2.1で計算した類似度$sim_{x,y}$から類似度の高いユーザを抽出する.

2.3 アクティブユーザの未評価アイテムの予測値計算

アクティブユーザにアイテムを推薦するための指標として予測値を用いる. アクティブユーザaに対する未評価アイテムi'の予測値$P(a_{i'})$を表す計算式は以下の(2)のようになる.

$$P(a_{i'}) = \frac{\sum_{x \in U(i')} (x_{i'} - \overline{x}) sim_{a,x}}{\sum_{x \in U(i')} |sim_{a,x}|} \quad (2)$$

ここで, ユーザxは2.3で抽出した高類似度ユーザを指す.

2.4 高予測値アイテム抽出

2.3で計算した予測値$P(a_{i'})$から予測値の高い未評価アイテムを抽出する.

2.5 アイテムを推薦

2.4で抽出した高予測値の未評価アイテム上位N個をアクティブユーザへ推薦する.

3 実装・評価

3.1 実装

CUDA上で実装する並列処理を施したユーザベース協調フィルタリングのフローチャートを図1に示す. ここではデバイス側の4つの並列処理の実装方法について説明する.

3.1.1 類似度計算

デバイス側に確保したユーザ数分の配列に対して, 同一数のスレッドを動作させて並列に類似度を算出する.

Speed up of user-based collaborative filtering by GPGPU
†Toshimitsu Tokiwa　†Shigeru Oyanagi
†College of Information Science and Engineering, Ritsumeikan University

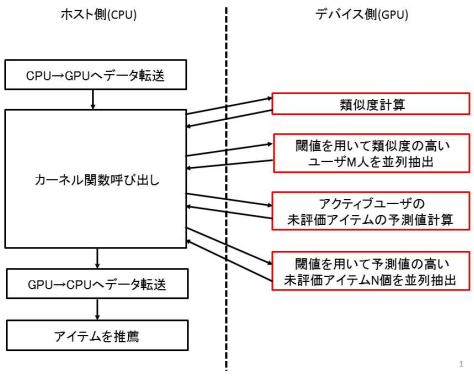


図 1: CUDA によるユーザベース協調フィルタリング

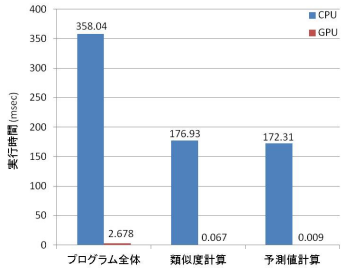
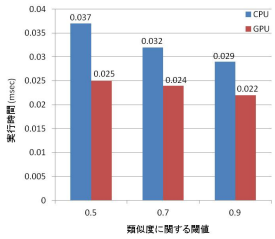


図 2: 全体時間と類似度計算時間と予測値計算時間

表 1: 評価環境

CPU	Intel Core i5-2500
GPU	NVIDIA GeForce GTX 580
CUDA	Version5.0



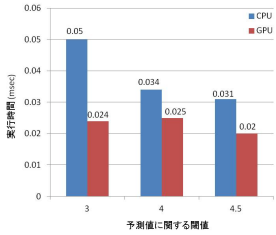


図 3: 高類似度ユーザ抽出時間　図 4: 高予測値アイテム抽出時間

3.1.2 高類似度ユーザ抽出

類似度が $-1 \leq sim_{x,y} \leq 1$ であることを利用して, 高類似度だと想定される閾値を設定する. アクティブユーザを除いたユーザ数の配列を確保し, この閾値を超えたユーザを並列抽出する.

3.1.3 予測値計算

デバイス側に確保したアイテム数分の配列に対して, 同一数のスレッドを動作させて並列に予測値を算出する.

3.1.4 高予測値アイテム抽出

高予測値だと想定される閾値を設定する. アイテム数の配列を確保し, この閾値を超えたアクティブユーザの未評価アイテムを並列抽出する.

3.2 評価

CPU と GPU での実行時間の相違を確認するために, C 言語でユーザベース協調フィルタリングを実装し, CUDA 実装のものと比較を行った. サンプルデータとして, MovieLens の 100 万件のデータセットを使用した. なお, プログラムの実行時間の計測に使用した評価環境は表 1 の通りである.

図 2 はプログラム全体の実行時間と類似度計算時間と予測値計算時間を示したグラフである. C 言語実装と比較した場合, CUDA 実装ではプログラム全体が 133 倍の高速化, 類似度計算が 2640 倍の高速化, 予測値計算が 19145 倍の高速化に成功した.

図 3 は高類似度ユーザ抽出時間, 図 4 は高予測値アイテム抽出時間を示したグラフである. C 言語実装と比較した場合, 高類似度ユーザ抽出が閾値 0.5 の時には 1.48 倍の高速化, 高予測値アイテム抽出が閾値 3 の時には 2.08 倍の高速化に成功した.

4 おわりに

本研究では, ユーザベース協調フィルタリングを C 言語実装したものと, CUDA 実装したもので実行時間の比較を行った. 100 万件のデータセットを使用した場合, 処理全体で 133 倍の高速化, 類似度計算で 2640 倍の高速化, 高類似度ユーザ抽出で最大 1.48 倍の高速化, 予測値計算で 19145 倍の高速化, 高予測値アイテム抽出で最大 2.08 倍の高速化に成功した.

今後の課題として, 類似度計算時のユーザ 1 人当たりの総和演算でさらに並列処理を行うことや, 予測値計算時の未評価アイテム 1 個当たりの総和演算の中でさらに並列処理を行い, 高速化を図ることが考えられる.

参考文献

[1] Toby Segaran 著, 當山仁健・鴨澤眞夫 訳:『集合知プログラミング』, オライリー・ジャパン, (2008)

[2] GroupLens : ”MovieLens”, http://grouplens.org/datasets/movielens/