1J-01

# HPCクラスタを用いた協調フィルタリングアルゴリズムの並列化

持田 (市村) 春嘉 †
北陸先端科学技術大学院大学 †

井口 寧 ‡
北陸先端科学技術大学院大学 ‡

## 1 はじめに

協調フィルタリング [神 08] は産業界においてレコメンドエンジン構築アルゴリズムとして広く利用されている手法である。この手法の問題点として膨大な類似度計算を要する点があり、特に類似ユーザーを用いるレコメンドにおいては全てのユーザーの組み合わせについて類似度計算を行わなければならない。協調フィルタリングアルゴリズムの処理時間短縮に向け、北陸先端科学技術大学院大学所有の KAGAYAKI 計算機クラスタを使用し並列化を行った。

## 2 協調フィルタリング

### 2.1 協調フィルタリングの概要

協調フィルタリングはレコメンドエンジン実装の 1 つである。ユーザー×アイテムの評価値行列に対し、列もしくは行同士の類似度から k 近傍点を算出し、k 近傍点のユーザーもしくはアイテムの情報を用いてレコメンドを行う。例えばユーザーに対するレコメンド時には、レコメンドを行いたいユーザー $a$ の k 近傍のユーザー達 $b_k$ を抽出し、それらのユーザーが高く評価しているが、まだレコメンドを行いたいユーザーが評価していないアイテムをレコメンドする。

### 2.2 類似度計算

上述した k 近傍のユーザーを抽出するため、類似度計算が必要である。類似度計算とは、ベクトル空間上の点同士の距離を何らかの距離尺度によって算出する操作であり、距離尺度としてはユークリッド距離や角度を用いる事ができる。本研究では以下で定義されるコサイン類似度を類似度計算で使用した。

$$\cos(\mathbf{a}, \mathbf{b_i}) = \frac{\mathbf{a} \cdot \mathbf{b_i}}{\|\mathbf{a}\|\|\mathbf{b_i}\|} = \frac{\sum_{j=1}^{n} a_j b_{ij}}{\sqrt{\sum_{j=1}^{n} (a_j)^2}\sqrt{\sum_{j=1}^{n} (b_{ij})^2}} \quad (1)$$

ただし、$\mathbf{a}, \mathbf{b_i}$ は、協調フィルタリングの文脈においては、 $\mathbf{a}$ がレコメンドを行いたいユーザーのベクトル、$\mathbf{b_i}$ がその他のユーザー $i$ のベクトルとなる。

## 3 先行研究

協調フィルタリングアルゴリズムを高速化する先行事例として、アルゴリズム自体を改変して高速化を行う事例と、アルゴリズムは同一であるが並列処理・分散処理を用いて高速化を行う事例がある。前者としては貪欲法を用いて k 近傍グラフの構築を高速化する試み [PPLJ14]、後者としては Hadoop を用いて分散処理を行うことで処理全体を高速化する試み [ZS10] などがある。本稿では後者のアプローチと同様、アルゴリズム自体の改変ではなく分散処理を利用した高速化を目指した。

## 4 実装

本稿では協調フィルタリングアルゴリズム中の類似度計算部分に並列化を適用した。類似度計算の処理回数は膨大であるがそれぞれの処理は独立であるため、並列化に適している。並列化の実装には python の MPI ラッパーである mpi4py[DF21] を用い、ブロッキング通信、ノンブロッキング通信、集団通信での実装を行った。MPI 実装は OpenMPI4.1.1 を用いた。

## 5 評価

### 5.1 評価環境

評価は北陸先端科学技術大学院大学所有の KAGAYAKI 計算機クラスタを用いて行った。KA-

Parallelization of Collaborative Filtering Algorithms Using HPC Clusters
† Haruka Mochida(Ichimura), Japan Advanced Institute of Science and Technology
‡ Yasushi Inoguchi, Japan Advanced Institute of Science and Technology

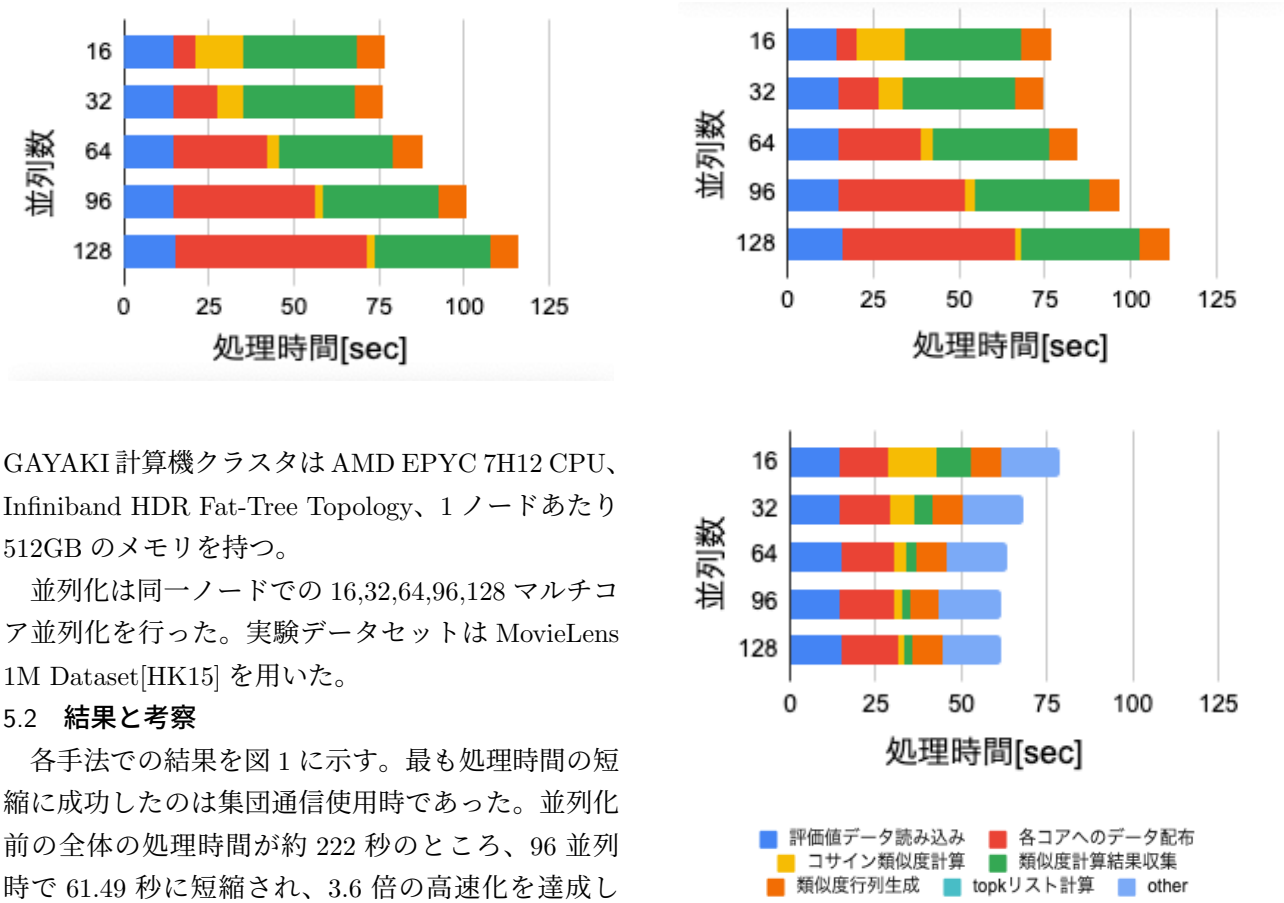


図 1　ブロッキング通信 (左)、ノンブロッキング通信 (右上段)、集団通信 (右下段) を用いた各並列数での並列化時の処理時間内訳

GAYAKI 計算機クラスタは AMD EPYC 7H12 CPU、Infiniband HDR Fat-Tree Topology、1 ノードあたり 512GB のメモリを持つ。

並列化は同一ノードでの 16,32,64,96,128 マルチコア並列化を行った。実験データセットは MovieLens 1M Dataset[HK15] を用いた。

### 5.2　結果と考察

各手法での結果を図 1 に示す。最も処理時間の短縮に成功したのは集団通信使用時であった。並列化前の全体の処理時間が約 222 秒のところ、96 並列時で 61.49 秒に短縮され、3.6 倍の高速化を達成した。並列数に対して高速化効率が高くない要因としては、全体の処理時間に対する非並列化部分の割合の大きさがあり、並列化後の処理時間の約 38 %をデータ読み込み等の非並列化部分が占めていた。

ブロッキング通信、ノンブロッキング通信使用時は、各コアへのデータ配布にかかる時間は並列数の増加に応じて増加していくが、コサイン類似度計算にかかる時間は並列数に応じて減少するという同様の傾向が見られた。今回使用したデータ量では、並列数を増やすことによるコサイン類似度計算時間の減少効果よりも各コアへのデータ配布にかかる時間の増加効果の方が大きく、全体の処理時間が最も短くなるのは 32 並列時となった。

## 6　結論

HPC クラスタを用いた並列化により、集団通信利用時の 96 並列で 3.6 倍の高速化を達成した。

## 参考文献

[DF21]　L Dalcin and Y Fang. mpi4py: Status update after 12 years of development. *Comp. in Science Engineering*, 23(4):47–54, 2021.

[HK15]　F Harper and J Konstan. *ACM Trans. on Interactive Intelligent Systems*, 5(4), dec 2015. Publisher Copyright: © 2015 ACM.

[PPLJ14]　Y Park, S Park, S Lee, and W Jung. Fast collaborative filtering with a k-nearest neighbor graph. pages 92–95, 2014.

[ZS10]　Z Zhao and M Shang. User-based collaborative-filtering recommendation algorithms on hadoop. pages 478–481, 2010.

[神 08]　神嶌敏弘. 推薦システムのアルゴリズム（２）. **人工知能学会誌**, 23(2):89–103, 2008.