バタフライの中間に冗長なデータ交換を行い ジッタの影響を緩和する集団通信アルゴリズム

松本英樹[1] 須田礼仁[1,2]
[1] 東京大学情報理工学系研究科 [2] JST CREST

1. はじめに

ジッタの影響により, 大規模計算機における並列計算のスケーラビリティが制限されるという問題がある[1]. 使用するプロセッサの増加に伴い, ジッタの影響は大きくなる. ジッタの影響を強く受ける allreduce について[2], ジッタの影響を緩和するアルゴリズム[3]がある. [3]の手法は, バタフライという allreduce アルゴリズムにおける, 通信の独立性を生かした方法である. 従来のデータ交換を終えた後, さらにデータ交換を複数回行い, 早く届いたデータを採用することでジッタの影響を緩和することができる. ところがこの手法はプロセッサ数が 2^{14} 程度以上で効果が小さくなる.

本論文では, バタフライの途中で冗長なデータ交換をし, 最後に再び冗長なデータ交換を行うことにより性能を改善する.

2. 既存手法

[3]では, 従来のバタフライによるデータ交換を終えた後に,冗長なデータ交換を行う. プロセッサ数が 4 の場合を図 1 に示す. 4 つのプロセッサがそれぞれのデータを交換するときの,左端のプロセッサが受け取るデータの流れをあらわした図である. バタフライは 2 段目までのデータ交換で終了する. [3]では,さらに 3 段目の冗長なデータ交換を行い, 青の矢印と橙の矢印で示した 2 通りの経路のうち,早く届いた方のデータを採用する.

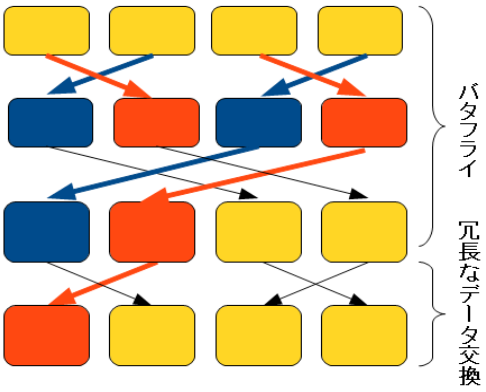

図 1:左端のプロセッサが受け取るデータの経路

ネットワークレイテンシ 1.0e-7 秒, ネットワークバンド幅 1.0e-9 秒, メモリバンド幅 1.0e-10 秒, ノイズの周期 1.0e-2 秒, ノイズの長さ 1.0e-4 秒の場合,プロセッサ数が 2^{14} を超えると, 図 3 の緑の折れ線が示すように[3]の提案手法もノイズの影響を大きく受ける.

3. 提案手法

本論文では, バタフライの最後だけでなく, バタフライの途中にも冗長なデータ交換を行う. プロセッサ数が 2^K のときバタフライの K/2(小数点以下切捨て)回目のデータ交換を終えた後に冗長なデータ交換を 1 回以上行い, 早く届いたデータを採用する(以下, 中間追加という). その後, バタフライの続きを行う. それを終えると[3]と同様にさらに冗長なデータ交換を 1 回以上行う. 冗長なデータ交換はいずれもバタフライと同じ通信パターンで行う. 擬似コードを図 3 に示す. ここで K はバタフライのデータ交換の段数, M は中間追加の回数, L は最後のデータ交換の回数である. (D+1)回目から(D+M)回目までのデータ交換が中間追加である. また, (K+M+1)回目から(K+M+L)回目までのデータ交換が最終追加である.

```
1 : for (i = 1; i <= 2 * (K + M + L); i++)
2 :   if i が偶数
3 :       計算時間の処理
4 :   else // 通信時間の処理
5 :       for プロセッサ p
6 :           src = プロセッサ p にデータを送信するプロセッサ番号
7 :           recv = etime [i − 1 ][ src ] + 通信時間 + ジッタ
8 :           D = K / 2
9 :           if ((i < 2 * D)
10 :                または (2 * (D + M) < i かつ i < 2 * (K + M)))
11 :               //バタフライのデータ交換
12 :               etime[i][p] = max(etime[i-1][p], recv)
13 :          else // 中間追加および最終追加のデータ交換
14 :               etime[i][p] = min(etime[i-1][p], recv)
```

図 2 : 提案手法の擬似コード

4. シミュレーションによる予備評価

[3]と同じシミュレータを用いて評価した. 図 3 は, 各手法の平均実行時間を表したグラフである. 提案手法は最終追加を最大 20 回行う. また提案手法(M)は中間追加を M 回行うことを意味するものとする. プロセッサ数が 2^{14} 個を超えると[3]の手法ではジッタの影響が大きいが, 提案手法(1)の場合は[3]の手法よりもジッタの効果を小さくできている. さらに提案手法(2)の場合, 提案手法(1)より効果が大きく, プロセッサ数が 2^{18} よりも小さければ, ジッタの影響を緩和できることがわかる.

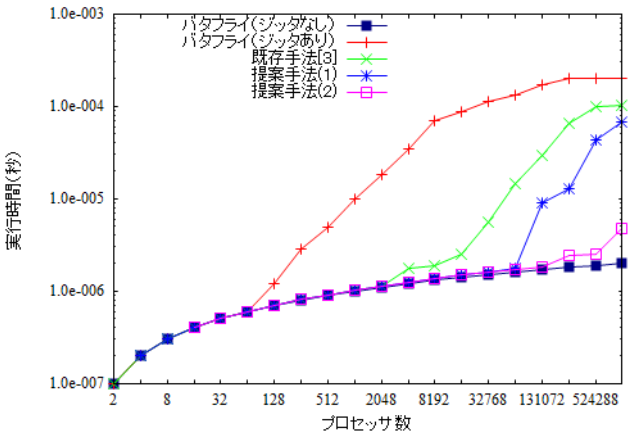

図 3:各手法の実行時間

この結果より, バタフライの最後だけでなく, 途中で冗長なデータ交換を行うことで性能の改善が可能であると言える. 今後の課題は, 中間追加と最終追加をそれぞれ何回行えばよいかを調べたい.

参考文献

[1]Petrini, F. et al.: The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q, *SC03*, pp. 55-, (2003).
[2]Beckman, P. et al.: The Influence of Operating Systems on the Performance of Collective Operations at Extreme Scale, *Cluster06* , Vol. 0, pp. 1-12 (2006).
[3]松本英樹, 須田礼仁, ジッタの影響を緩和する集団通信アルゴリズム, 情報処理学会 第 137 回 HPC 研究会(2012)