Vol.2017-HPC-161 No.6
2017/9/19

グラフ分割ライブラリ METIS と リストスケジューリングを応用した負荷分散の改善

西川 直樹[1]　置田 真生[1]　萩原 兼一[1]

概要：プログラムを並列化するとき，計算の割当にグラフを用いる手法がある．計算を規定するグラフを分割し，部分グラフをそれぞれ 1 つの実行ユニットに割り当てる．グラフの頂点および切断辺はそれぞれ計算および通信に対応するため，部分グラフの頂点の重みの和が均衡し，かつ切断辺の重みの和が少ない分割が望ましい．このような分割を得る手法として METIS が広く使われている．しかし，METIS k-way partitioning を利用してグラフを分割すると，頂点の重みの均衡化に失敗する場合がある．この結果に従って計算を割り当てると，負荷の不均衡により性能が低下する．そこで本研究では，負荷の均衡という制約下で通信を削減する割当手法を提案する．具体的には，k-way partitioning を用いてグラフを粗粒化し，リストスケジューリングを応用して実行ユニットに割り当てる．提案手法を適用した結果，k-way partitioning のみを用いた場合と比較して最大 4.2 倍の速度向上を達成した．

キーワード：自動並列化，コード生成，MPI

Naoki Nishikawa[1]　Masao Okita[1]　Kenichi Hagihara[1]

1. はじめに

一般に，MPI 並列プログラムの実行時間は，プロセッサ・エレメント（以降，PE と表記）に割り当てられた計算の量や通信量に依存する．PE 間の計算の量に差があると，計算の量が大きい PE の処理が終了するまで他の PE に待ち時間が生じる．また，通信は並列処理におけるオーバヘッドとなる．したがって，実行時間の削減には計算負荷の均衡および通信量の削減が必要である．

プログラムを並列化する時，計算の割当にグラフを用いる手法がある [1], [2]．計算を規定するグラフを分割し，得られた部分グラフをそれぞれの PE に割り当てる．グラフの頂点および切断辺をそれぞれ計算処理および通信処理としたとき，理想的な分割は部分グラフの頂点の重みの和がほぼ等しく，切断辺の重みが最小となる分割である．しかし，このような分割を得る問題は NP 困難 [3] であるため，一般には近似アルゴリズムを用いる．

グラフを用いて並列化するソフトウェアの 1 つに，Flint[4] がある．生体モデル記述言語である PHML（Physiological Hierarchy Markup Language）[5] を入力としてシミュレーションコード（以降，SC と表記）を出力する．Flint は生体モデルを数式の依存グラフとして解釈し，グラフ分割を行う．得られた部分グラフに対応する処理をそれぞれの PE に割り当てた MPI プログラムを生成する．

Flint は，グラフ分割にグラフ分割ライブラリ METIS[6] に含まれる multilevel k-way partitioning algorithm[7], [8]（以降，part_kway）を用いる．しかし，part_kway を用いて特定の生体モデルに基づくグラフを分割すると，部分グラフ間で頂点の重みの和が 2 倍以上異なる場合がある．すなわち SC の実行における PE 間の計算負荷の格差が大きい．

これまでのところ，Flint を利用して行ったシミュレーションは計算時間の占める割合が非常に大きく，通信時間は高々数%である．したがって，性能向上のためには通信の最小化よりも負荷の均等化が重要である．

そこで本研究では，負荷均衡を重視し，その下で通信を削減する割当手法を提案する．提案手法を Flint に実装し，生体モデルのシミュレーションに適用して実際の性能を評価する．

以降では，2 節で必要な知識と諸定義を述べる．3 節で既存実装における問題点を示し，4 節で提案手法を説明す

[1] 大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻
Department of Computer Science,Graduate School of Information Science and Technology,Osaka University

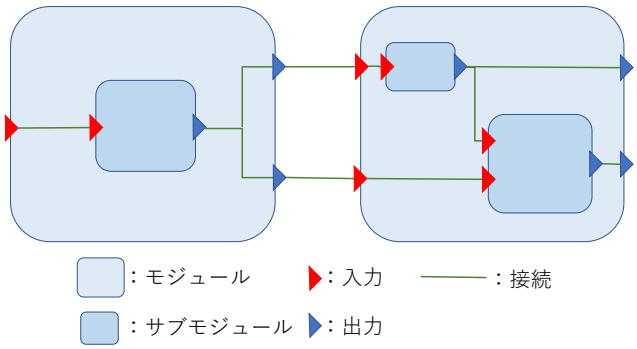

図 1　PHML モジュール・ネットワークの例

る．5 節で評価実験を行い，結果を考察する．6 節で関連研究について述べ，最後に 7 節で本稿をまとめる．

2. 準備

2.1 表記

重みつき非多重有向グラフを $G = (V, E, q, w)$ と表記する．V および E はそれぞれ頂点および辺の集合を表す．$e_{i,j} \in E$ は，$v_i \in V$ を始点とし $v_j \in V$ を終点とする辺を表す．$q : V \to \mathbf{R}$ は頂点の重みを表し，以降では $q(v_i)$ を単に q_i と表記する．同様に，辺の重み $w : E \to \mathbf{R}$ に対し $w(e_{i,j})$ を $w_{i,j}$ と表記する．$X \subseteq V$ に対する要素の重みの総和を $\theta(X)$ で表す．

グラフ G に対し V の分割 $P = \{P_1, P_2, \cdots, P_n\}$ を得ることをグラフ分割といい，$\mathrm{part}(G, n, P)$ と表す．辺 $e_{i,j} \in E$ について $v_i \in P_k, v_j \in P_l \neq P_k$ ならば切断辺といい，P に対する G の全ての切断辺の集合を $C_G(P)$ と表す．

2.2 汎用生体シミュレータ Flint

Flint は，PHML で記述された生体モデル M と並列度 n を入力とし，MPI で並列化された SC を生成する．1 つの PHML モデルは，連立一階常微分方程式からなる系を表現する．SC はこれら常微分方程式の初期値問題をオイラー法で数値的に解く．SC をコンパイルし，シミュレーションのステップ数を指定して実行することで M の時間発展を得る．

PHML では，生体モデルを階層的モジュールのネットワークとして記述する（**図 1**）．モジュールは生体組織あるいは生理機能を表し，複数の常微分方程式および初等関数の方程式（以降，これらを合わせて数式と表記）をカプセル化したものである．さらに，モジュール内にサブモジュールを内包できる．一方，ネットワークはモジュール間の作用を表す．モジュールは複数の入力と複数の出力を持ち，入出力を介して接続するモジュール間には数式の依存関係がある．なお，1 組のモジュール間に複数の作用がある場合は多重接続となる．

MPI 並列化のため，Flint はグラフ分割を利用して PE へタスク割当を決定する．まず，M をモジュールを頂点とし，接続を有向辺とする $G = (V, E, q, w)$ で表現する．接続の多重度を辺の重みとし，モジュールが内包する数式の総数を頂点の重みとする．次に，part_kway を用いて $\mathrm{part}(G, n, P)$ を実行する（以降ではこれを $\mathrm{part\_kway}(G, n, P)$ と表記）．最後に，P の要素をラウンドロビンで PE に 1 つずつ割り当てる．

Flint は $P_i \in P$ のそれぞれをプログラムに変換し，まとめて 1 つの $\mathrm{SC}(P)$ として出力する．全ての $v \in P_i$ に含まれる数式を抽出し，依存関係をもとにスケジュールを決定する．この際，切断辺 $e \in C_G(P)$ に基づいて必要な通信命令を挿入する．複数の通信命令を一括化できるよう，フェーズ単位で挿入個所を決定する [4]．スケジュールに従って数式を計算処理（代入文）に変換し，$\mathrm{SC}(P)$ を得る．

2.3 グラフ分割の有用性の指標

本研究においてグラフ分割 $\mathrm{part}(G, n, P)$ の有用性を評価するための指標を定義する．$\mathrm{SC}(P)$ の実行時間が短いことが望ましい．実行時間を決定する要素として通信量と負荷分散の 2 点に注目する．

まず，通信量を評価するために次式の $\mathrm{cut}_G(P)$ を定義する．

$$\mathrm{cut}_G(P) \equiv \sum_{e_{i,j} \in C_G(P)} w_{i,j} \tag{1}$$

通信量は切断辺の重みと正の相関があり，小さいことが望ましい．すなわち，$\mathrm{cut}_G(P)$ が 0 に近いほど優れたグラフ分割である．

次に，負荷分散の評価には以下の $\mathrm{fairness}(P)$ を用いる．

$$\mathrm{fairness}(P) \equiv \frac{\max\{\theta(X), X \in P\}}{\mathrm{average}\{\theta(X), X \in P\}} \tag{2}$$

P_i を割り当てられた PE の計算負荷は，$\theta(P_i)$ と正の相関がある．並列プログラムの実行時間は最も遅い PE に依存するため，$\theta(P_i)$ の最大値を用いる．最大値と平均値が等しいとき，すなわち $\mathrm{fairness}(P) = 1$ のとき理想的な負荷分散である．

以上を用いて，$\mathrm{SC}(P)$ の実行時間 $T(P)$ を次の (3) でモデル化する．

$$T(P) = T_{\mathrm{calc}}(P) + T_{\mathrm{com}}(P) + z_1 \tag{3}$$

$$T_{\mathrm{calc}}(P) = z_2 \times \mathrm{fairness}(P) \tag{4}$$

$$T_{\mathrm{com}}(P) = z_3 \times \mathrm{cut}_G(P) \tag{5}$$

z_1 および z_2, z_3 は正の実数であり，$T_{\mathrm{calc}}(P)$ および $T_{\mathrm{com}}(P)$ はそれぞれ計算時間および通信時間を表す．$T_{\mathrm{com}}(P)$ については通信パターンを考慮したより詳細なモデルがあり得るが，本稿では相対的に通信時間が短いため単純な線形モデルを用いる．$T(P)$ を最小化するようなグラフ分割が有用である．

表 1　part_kway(G, n, P) の fairness(P)

モデル G	$n=2$	$n=4$	$n=8$	$n=16$	$n=32$
1x301	1.00	1.01	1.01	1.01	1.06
fsk-epi	1.00	1.01	1.02	1.02	1.03
rT-L	1.13	1.20	1.16	1.23	1.30

表 2　rT-L を入力とした part_kway$(G, 4, P)$

	$\theta(P_i)$	切断辺数（入次）	切断辺数（出次）
P_1	9,593,000	17,583	139,050
P_2	8,360,722	14,318	119,479
P_3	5,226,464	12,491	94,315
P_4	8,769,736	15,341	127,019

表 3　表 2 に対する SC(P) の実行時間内訳（単位は秒）

	$T_{\mathrm{calc}}(P_i)$	$T_{\mathrm{com}}(P_i)$	$T(P_i)$
P_1	2,655.34	4.63	2,776.88
P_2	2,330.32	345.96	2,776.76
P_3	1,423.68	1,292.03	2,776.46
P_4	2,400.67	269.32	2,776.77

3. multilevel k-way partitioning を用いたコード生成の問題点

part_kway は多制約グラフ分割アルゴリズム [7] の実装であり，METIS の一部として公開されている．part_kway(G, n, P) は，$\mathrm{cut}_G(P)$ ができるだけ小さく，かつ fairness$(P) \leq \tau$ となる分割をヒューリスティックに求める．fairness(P) の上限 τ は利用者が任意に指定できる．多制約とは頂点の重みが多次元であることを意味し，次元ごとに fairness(P) の上限を指定できる．

しかし，特定のグラフを入力とした場合に fairness$(P) > \tau$ となる場合がある．**表 1** に part_kway（$\tau = 1.1$）を用いた場合の fairness(P) を計測した結果を示す．利用したモデルの詳細は 5 節（**表 4**）に後述する．その他のモデルと比較して，rT-L だけ n に関わらず fairness(P) が τ を上回る．1 例として，rT-L における part_kway$(G, 4, P)$ の結果を表 2 に示す．$\theta(P_3)$ だけが他と比較して小さく，fairness(P) 増大の原因となる．

fairness(P) が比較的大きい場合，PE の遊休状態が発生する．実際に rT-L から生成した SC(P)（$n = 4$）の PE ごとの実行時間を**表 3** に示す．実行環境は後述する**表 5** のマルチコア環境である．各行は P_i を割り当てた PE の処理時間の内訳を表す．なお，通信処理は同期待ち時間を含む．P_1 の計算処理が全体の 96%を占める一方，P_3 の計算処理は 51%に過ぎない．P_3 の通信処理はほとんどが同期待ち時間であるため，P_3 を割り当てた PE は全体のおよそ半分で遊休状態にある．

その結果，計算が支配的な状況下で SC(P) の実行性能が低下する．ここで計算が支配的な状況とは，$T_{\mathrm{calc}}(P) \gg T_{\mathrm{com}}(P)$ となる実行環境および入力モデル，並列度 n の組み合わせを指す．例えば，高速通信が可能な環境で $\mathrm{cut}_G(P)$ が十分小さい場合が挙げられる．計算が支配的な状況では，通信はほぼ無視できるため，遊休状態が性能低下に及ぼす影響が大きい．part_kway(G, n, P) は cut(P) が小さい点で有用であるが，特定のモデルに対して fairness(P) が比較的大きい点が問題となる．

アルゴリズム 1 part_fair(G, n, P)　　$\alpha = 0.02, \varepsilon = 1.01$

```
m ← 1
x ← 0
while nm ≤ |V| do
  x ← x + 1
  part_lsch(G, n, m, H_x)
  if fairness(H_x) < 1 + α then
    P ← H_x
    return
  else if x > 2 then
    if fairness(H_{x-2})/fairness(H_{x-1}) < ε and fairness(H_{x-1})/fairness(H_x) < ε then
      P ← H_{x-2}
      return
    end if
  end if
  m ← 2^x
end while
P ← H_x
```

アルゴリズム 2 part_lsch(G, n, m, P)

```
part_kway(G, nm, Q)
sort Q s.t. i < j → θ(Q_i) ≤ θ(Q_j)
initialize P s.t. ∀P_i ∈ P, P_i = ∅
for (i = 1 ··· nm) do
  select P_k s.t. P_k ∈ arg min θ(P_i)
  P_k ← P_k + Q_i
end for
```

4. 提案手法

計算が支配的な状況における $T(P)$ の削減を目的として，fairness(P) の最小化という制約の下で $\mathrm{cut}_G(P)$ を削減する分割手法 part_fair(G, n, P) を提案する．fairness(P) を最小化する手法として，Graham のリストスケジューリング [11]（以降，LSCH と表記）を応用する．全ての P_i が空である状態から始め，頂点を重みの大きいものから順に，その時点で $\theta(P_i)$ が最小となる P_i に頂点を割り当てる．ただし，この方法は切断辺を全く考慮しないため，$\mathrm{cut}_G(P)$ が比較的大きい．

$\mathrm{cut}_G(P)$ を減ずるため，G を粗粒化したグラフの頂点に対して LSCH を適用する．辺の数が減少するように粗粒化することで，LSCH 適用時に切断辺を考慮しなくても $\mathrm{cut}_G(P)$ の減少を期待できる．ここで，粗粒化の程度（以降，粗粒化度と表記）に依存して fairness(P) および $\mathrm{cut}_G(P)$ がトレードオフの関係にあることが実験的に分かっている．粒度が大きいほど $\mathrm{cut}_G(P)$ は減少する傾向にある一方，fairness(P) は増大する傾向にある．

表 4　実験に用いる生体モデル

モデル名	主要モジュール	ネットワーク	$\|V\|$	$\|E\|/\|V\|$	$\theta(V)$	ステップ数
1x301	視神経細胞 [9]	直線	302	15.76	37,933	100,000
fsk-epi	心筋細胞 [10]	トーラス体	455,890	1.74	39,585,395	1,000
rT-S	視神経細胞 [9]	2 層メッシュ	571	2.44	161,659	100,000
rT-L	視神経細胞 [9]	2 層メッシュ	39,522	7.16	31,949,922	1,000

表 5　実験環境

	マルチコア CPU	CPU クラスタ
CPU	Xeon E5-4640 v2	Xeon E3-1230
コア数	10×4	4
動作周波数	2.2 GHz	3.2 GHz
ノード数	1	64
ネットワーク	無し	1.0 Gbps Ethernet
主記憶	1,024 GB	16 GB
OS	CentOS 6.6	CentOS 5.6
コンパイラ	gcc 4.9.3	gcc 4.1.2
MPI	OpenMPI 1.8.1	OpenMPI 1.4.3

そこで提案手法では反復法を用いて粗粒化度を決定する．提案手法 part_fair(G, n, P) のアルゴリズムをアルゴリズム 1 に示す．m は粗粒化度を調整する変数で，小さいほど粒度が大きい．H_x を x 回目の反復時に得られた分割とし，fairness(H_x) が収束するまで，m を増大しながら分割 part_lsch(G, n, m, H_x)（アルゴリズム 2）を繰り返す．part_lsch(G, n, m, H_x) では，part_kway を用いて G を粗粒化したのち，LSCH を適用して分割 H_x を決定する．

反復の終了条件は以下のいずれかを満たす時である．

$$\text{fairness}(H_x) < 1 + \alpha \tag{6}$$

$$\frac{\text{fairness}(H_{x-2})}{\text{fairness}(H_{x-1})} < \varepsilon \wedge \frac{\text{fairness}(H_{x-1})}{\text{fairness}(H_x)} < \varepsilon \tag{7}$$

$$mn > |V| \tag{8}$$

条件 (6) は fairness(H_x) が十分小さいことを意味する．今回の実装では経験的に $\alpha = 0.02$ とした．条件 (7) は H_x の収束を意味し，$\varepsilon = 1.01$ とする．H_{x-1} との比較だけで判断すると極小値に陥る可能性が高いため，H_{x-2} とも比較することでより適切な解を得る．なお，この条件を満たす場合は H_x ではなく H_{x-2} を解とする．その理由は，fairness(H_x) がほぼ同等であれば粒度が大きいほど，すなわち x が小さいほど $\text{cut}_G(H_x)$ が小さいためである．また，mn が頂点数 $|V|$ より大きくなった場合，それ以上粒度を小さくできないため反復を終了する（条件 (8)）．m と fairness(P) は概ね負の相関があるため，反復終了時の H_x を解とする．

なお，part_fair の入出力は part_kway の互換であるため，Flint への適用は容易である．part_kway の呼び出しを置換するだけでよい．

5. 評価実験

以下の観点から，提案手法を評価する．

表 6　Zoltan パラメータ

LB_METHOD	GRAPH
GRAPH_PACKAGE	ParMETIS
LB_APPROACH	Partition
PARMETIS_METHOD	PartKway
GRAPH_SYMMETRIZE	TRANSPOSE
SCATTER_GRAPH	0
IMBALANCE_TOL	1.1

表 7　part_kway(G, n, P) および part_fair(G, n, P) の比較

モデル G	n		part_kway	part_fair	m
1x301	4	fairness(P)	1.01	1.01	1
		$\text{cut}_G(P)$	110	110	
	32	fairness(P)	1.06	1.06	1
		$\text{cut}_G(P)$	994	994	
fsk-epi	4	fairness(P)	1.01	1.01	1
		$\text{cut}_G(P)$	29,766	29,766	
	32	fairness(P)	1.03	1.00	2
		$\text{cut}_G(P)$	93,918	124,790	
rT-S	4	fairness(P)	1.21	1.01	8
		$\text{cut}_G(P)$	688	1,356	
	32	fairness(P)	4.84	1.08	16
		$\text{cut}_G(P)$	2,954	3,986	
rT-L	4	fairness(P)	1.20	1.00	16
		$\text{cut}_G(P)$	119,466	155,300	
	32	fairness(P)	1.30	1.02	16
		$\text{cut}_G(P)$	731,440	1,093,432	

- 分割結果
- SC 生成時間
- SC 実行時間

提案手法では $\text{cut}_G(P)$ が増大する場合があるため，通信に要する時間の異なる 2 つの環境で評価した．通信に要する時間が比較的大きい CPU クラスタ環境および高速な通信を期待できるマルチコア CPU 環境である．それぞれの詳細を表 5 に示す．

入力となる生体モデルは，規模とネットワーク構造の異なる 4 つのモデルを使用した（表 4）．規模の異なるモデルを評価に用いる理由は，$T_{\text{calc}}(P)$ に対する $T_{\text{com}}(P)$ の比が異なる場合を検証するためである．1x301 および fsk-epi のネットワーク構造はそれぞれ単純な直線および心室を模したトーラス体であり，隣接するモジュール間のみ接続がある．一方，rT-S および rT-L は 2 次元格子状に配置したモジュールに対し，近傍にあるほど高確率で接続が存在するネットワーク構造 [12] である．

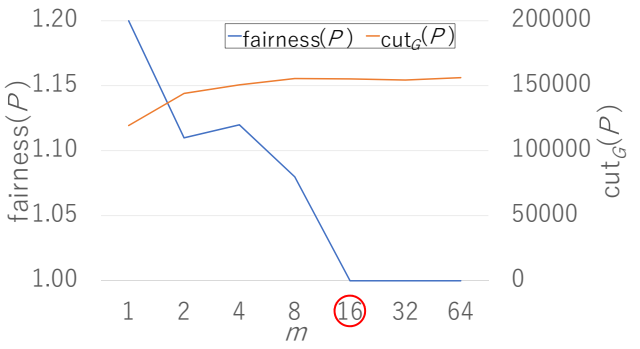

図 2 part_lsch$(G, 4, m, P)$ における m と fairness(P) および cut$_G(P)$ の関係（対象：rT-L，赤丸は part_fair が選択した m の値）

図 3 part_lsch$(G, 32, m, P)$ における m と fairness(P) および cut$_G(P)$ の関係（対象：1x301，赤丸は part_fair が選択した m の値）

Flint は，動的負荷分散ライブラリ Zoltan[13] を介して part_kway を実行する．使用したバージョンは Zoltan 3.2 および ParMETIS 3.1（Zoltan 同梱版）である．Zoltan の設定を**表 6** に示す．IMBALANCE_TOL が fairness(P) の上限 τ を表す．記載のないパラメータは標準値を用いた．

5.1 分割結果

fairness(P) および cut$_G(P)$ について，提案手法 part_fair を part_kway と比較した（**表 7**）．fairness(P) に関して，モデルの違いおよび分割数 n に関わらず，part_fair は常に part_kway 以下の値を示す．part_kway の場合と比較して fairness(P) が小さく，かつその場合に限り，cut$_G(P)$ は増大する．すなわち，提案手法は fairness(P) に改善の余地があるとき，cut$_G(P)$ の増大と引き替えに fairness(P) を削減できることを示した．

表 7 が示すように，2 つの例外を除いて，part_fair の結果は fairness$(P) \leq 1.02$ に収束した．収束の例を**図 2** に示す．fairness(P) の極小値（$m = 2$）を避けて適切な m を選択することを確認できる．例外の 1 つ，1x301 を対象とした $n = 32$ の場合には，収束するものの fairness$(P) = 1.06$ と比較的大きい（**図 3**）．fairness(P) にほとんど差がない中で，cut$_G(P)$ が最小となる $m = 1$ を選択できた．もう一方の例外，rT-S を対象とした $n = 32$ の場合には，m が最大値でも fairness(P) が収束しなかった（**図 4**）．この場合にも fairness(P) が最小となる m を選択できた．以上のように，提案手法は cut$_G(P)$ の過度な増大を避けつつ，fairness(P) を最小化することを確認できた．

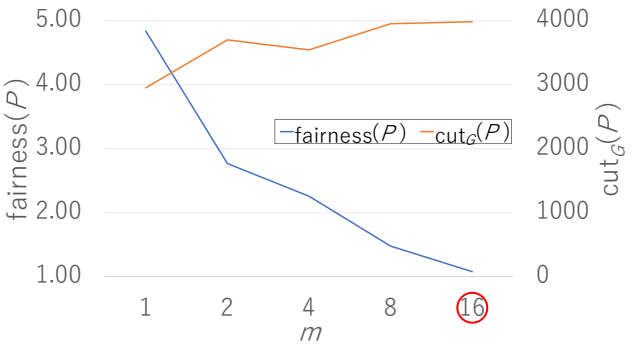

図 4 part_lsch$(G, 32, m, P)$ における m と fairness(P) および cut$_G(P)$ の関係（対象：rT-S，赤丸は part_fair が選択した m の値）

表 8 分割時間および SC 生成全体の処理時間（単位は秒）

		分割時間			全体時間
モデル G	n	既存手法	提案手法	反復 x	提案手法
1x301	4	0.02	0.02	1	10.73
	32	0.03	0.08	4	84.69
fsk-epi	4	9.53	11.23	1	1,234.83
	32	12.05	17.32	2	528.03
rT-S	4	0.03	0.05	4	3.85
	32	0.04	0.08	5	1.60
rT-L	4	4.58	8.46	5	1,081.51
	32	6.18	10.90	5	435.50

fairness$(P) > 1.02$ となる両場合について，その原因はともに n に対して $|V|$ が小さいためである．q の平均が 100 以上あるため，どのように割り当てても fairness(P) の削減は難しい．さらなる削減には頂点を分割するなどの工夫が必要である．

5.2 SC 生成時間

提案手法による分割に要する時間を評価する．既存手法を用いた場合の分割時間と，提案手法を用いた場合の分割時間および SC 生成全体の処理時間を**表 8** に示す．分割時間には，グラフ分割に要する時間だけでなく前処理および後処理の時間を含む．提案手法は part_kway を x (≥ 1) 回繰り返すため，全ての場合で既存手法を上回る分割時間を要する．分割時間が 0.1 秒未満とごく短い場合を除くと，part_kway と比較して最大 1.8 倍であった．

表 9　SC(P) 実行時間（単位は秒）

モデル G	n		マルチコア CPU		CPU クラスタ	
			part_kway	part_fair	part_kway	part_fair
fsk-epi	32	$T(P)$	363.84	363.55	205.96	204.35
		$T_{\mathrm{calc}}(P)$	352.92	348.94	162.85	156.11
		$T_{\mathrm{com}}(P)$	1.15	5.64	11.32	12.71
rT-S	4	$T(P)$	648.19	487.72	561.49	540.41
		$T_{\mathrm{calc}}(P)$	608.47	449.12	490.11	364.42
		$T_{\mathrm{com}}(P)$	8.68	12.80	29.05	166.40
	32	$T(P)$	412.97	151.30	1,665.06	396.59
		$T_{\mathrm{calc}}(P)$	283.64	82.52	319.57	75.73
		$T_{\mathrm{com}}(P)$	106.32	40.56	1,335.78	316.12
rT-L	4	$T(P)$	2,776.88	2,370.51	1,813.90	1,486.22
		$T_{\mathrm{calc}}(P)$	2,655.34	2,247.14	1,700.89	1,373.98
		$T_{\mathrm{com}}(P)$	4.63	5.77	5.75	20.94
	32	$T(P)$	368.05	312.48	257.63	193.89
		$T_{\mathrm{calc}}(P)$	332.14	288.47	174.54	122.59
		$T_{\mathrm{com}}(P)$	7.03	7.36	9.71	21.55

また，表 8 が示すように，SC 生成時間のうち分割処理が占める割合は非常に小さい．part_fair による分割時間の増大は，SC 生成時間の高々 1.08%である．このことから，提案手法による分割時間の増大は SC 生成時間に影響を与えないことを確認できた．

5.3　SC 実行時間

提案手法による SC 実行の高速化を評価する．part_kway および part_fair をそれぞれ用いて生成した SC(P) の実行時間を**表 9** に示す．両手法で分割結果が等価である 1x301（$n = 4, 32$）および fsk-epi（$n = 4$）については省略する．fsk-epi に対しては $T(P)$ がほぼ同等であり，rT-S および rT-L に対しては提案手法の $T(P)$ が小さかった．

既存実装において fairness(P) が最適値に近い値である fsk-epi（$n = 32$）は，part_fair による fairness(P) の削減量が小さいため $T_{\mathrm{calc}}(P)$ は高々 4%の削減だった．一方 $T_{\mathrm{com}}(P)$ は最大 4.91 倍に増大したが，$T(P)$ のうち $T_{\mathrm{calc}}(P)$ の占める割合が大きいため，全体としては part_fair と part_kway の差はなかった．このことから，今回の実験環境と生体モデルでは $\mathrm{cut}_G(P)$ の増大による不利益が十分小さいことを確認できた．

既存実装において fairness(P) が比較的大きい rT-S では，part_fair による fairness(P) の削減量が大きいため $T_{\mathrm{calc}}(P)$ は最大 76%削減した．同様に，rT-L も part_fair による fairness(P) の削減量が大きいため $T_{\mathrm{calc}}(P)$ は最大 30%削減した．$T_{\mathrm{calc}}(P)$ の占める割合が大きく，かつ $T_{\mathrm{calc}}(P)$ の削減量が大きいため，$T(P)$ を 15%〜76%削減した．ただし，CPU クラスタ環境における rT-S の $n = 4$ の場合は，T_{com} 増大の影響が大きく，$T(P)$ の削減率は 3.7%だった．以上から，提案手法は計算が支配的な状況において有効である．

注目すべき点として，rT-S の $n = 32$ の場合，$\mathrm{cut}_G(P)$ が増大したにも関わらず $T_{\mathrm{com}}(P)$ が減少した．特に CPU クラスタ環境においては $T_{\mathrm{com}}(P)$ を 76%削減した．このように，$\mathrm{cut}_G(P)$ の増大は必ずしも $T_{\mathrm{com}}(P)$ の増大を引き起こさない．

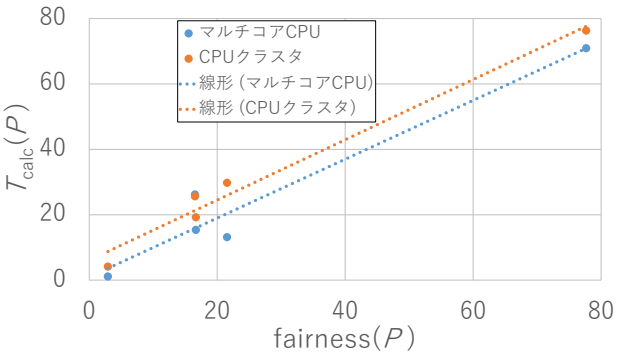

図 5　$T_{\mathrm{calc}}(P)$ および fairness(P) の相関

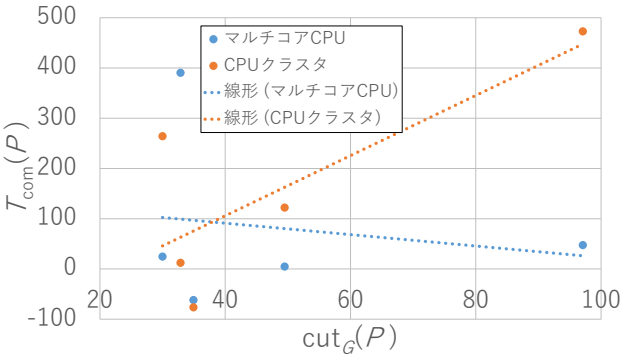

図 6　$T_{\mathrm{com}}(P)$ および $\mathrm{cut}_G(P)$ の相関

5.4　考察

分割結果と SC 実行時間の相関関係を確認する．まず，提案手法による，既存実装に対する fairness(P) の削減率および $T_{\mathrm{calc}}(P)$ の削減率を**図 5** に示す．マルチコア CPU 環境および CPU クラスタ環境における相関係数はそれぞれ 0.97 および 0.99 だった．すなわち，fairness(P) および $T_{\mathrm{calc}}(P)$ にはかなり強い相関があり，グラフ分割の指標として適切である．

次に，提案手法による，既存実装に対する $\mathrm{cut}_G(P)$ の増大率および $T_{\mathrm{com}}(P)$ の増大率を**図 6** に示す．CPU クラスタ環境における相関係数は 0.77 と強い相関があり，式 (5) の単純なモデルでも有用である．一方で，マルチコア CPU 環境おける相関係数は-0.18 と負の相関がある．すなわち，式 (5) の $T_{\mathrm{com}}(P)$ のモデルが不十分であり，$\mathrm{cut}_G(P)$ はグラフ分割の指標として有用でない．$T_{\mathrm{com}}(P)$ のより正確なモデル化およびグラフ分割指標の改善は今後の課題である．

また，rT-S の $n = 32$ の場合における $T_{\mathrm{com}}(P)$ の減少の原因を分析する．この原因は，（S）通信の負荷分散が改善されたためである，と推測した．rT-S のネットワーク構造では，頂点に接続する辺の数はおよそ一定であるため，fairness(P) の改善により通信の負荷分散も改善される．（S）

を検証する．rT-S の $n = 32$ における，part_kway(G, n, P) および part_fair(G, n, P) による P の中で頂点の重みの和が最も大きい要素をそれぞれ X および Y とする．この時 X および Y に対する切断辺の重みの和はそれぞれ 286 および 126 であり，X 対する Y の割合は 44%だった．マルチコア CPU および CPU クラスタにおける $T_{\mathrm{com}}(P)$ の削減率はそれぞれ 62%および 76%だった．この結果から，(S) が原因であるとは断定できない．更なる究明は今後の課題である．

6. 関連研究

グラフ分割に基づく並列処理における負荷分散の研究として，藤森らの分散グラフ処理におけるグラフ分割の最適化 [2] がある．分散グラフ処理では，入力となるグラフを分割して得られた部分グラフを各計算機に割り当て，各計算機が自身に割り当てられた部分グラフに対する処理を実行する．処理の実行中，切断辺では計算機間で通信が発生するため，切断辺が少ないほど処理の実行中の通信コストは小さくなる．また，グラフデータに対する分析処理の多くは，処理する辺の数が多いほどタスク量が増大するため，各計算機に割り当てられる部分グラフの辺の数を均衡させることによって処理時間の偏りを削減できる．

藤森らは，切断辺の最小化と部分グラフの辺の数の均衡化を両立することで，分散グラフ処理における分析処理を高速にするグラフ分割手法を提案した．グラフ分割を 2 つのステップに分け，最終的に計算機台数と同じ数のグラフに分割する．初めのステップでは，モジュラリティを最大化するようにグラフを計算機台数よりも多いクラスタ数にクラスタリングする．次のステップでは，初めのステップで得られたクラスタ群から辺の数の多いクラスタを計算機台数選択し，選択したクラスタに隣接するクラスタの中で辺の数が最小となるクラスタを順に結合することで部分グラフの辺の数を均衡させる．

本研究との類似点は，グラフ分割の前半に行う粗粒化の方針にある．藤森らの手法はモジュラリティを最大化，すなわち異なるクラスタ間が疎となるネットワークを得ることで，切断辺を削減する．提案手法は part_kway を用いることで，同様に切断辺の少ない粗粒化を実現する．

本研究との相違点は，粗粒化度の決定方法にある．藤森らの手法では，粗粒化度を利用者が予め決定する．一方，提案手法は粗粒化度を反復法を用いて自動的に決定する．

7. まとめ

本研究では，グラフを用いて並列化されるプログラムの，計算が支配的な状況における実行時間の削減を目的として，負荷均衡という制約の下で通信を削減する割当手法を提案した．具体的には，METIS k-way partitioning algorithm を用いて通信が減少するように粗粒化し，リストスケジューリングを応用して負荷が均衡するように PE に割り当てる．粗粒化度は反復法によって自動で決定する．

手法を適用して Flint でプログラムを生成し，負荷分散と通信量を計測した．結果，負荷分散に改善の余地があるとき，通信量の増大と引き換えに計算の負荷を均等にした．負荷分散に改善の余地があるときに対して，各 PE の計算の量の平均値に対する最大値の比が最大で 4.84 だったものが最大で 1.08 となり，計算の負荷が均衡した．この時適切な粗粒化度を選択でき，過度な通信の増大を回避した．

プログラムの生成時間のうち，グラフ分割処理に要する時間は最大で 1.8 倍となった．しかし，グラフ分割処理に要する時間はプログラムの生成時間の高々 1.08%であり，プログラムの生成時間の増大は全体と比べ無視できる範囲であった．

生成プログラムの実行時間は，マルチコア CPU 実行では 1.17 倍〜2.73 倍，CPU クラスタ実行では 1.10 倍〜4.20 倍の速度向上が得られた．

今後の課題として以下の 2 つが挙げられる．1 つ目は，通信の最小化である．本研究では計算が支配的な状況を対象としたが，通信により時間を要する環境にも対応する．2 つ目は，通信処理時間のより正確なモデル化である．

謝辞 本研究の一部は，科学研究費補助金（基盤研究（A）JP15H01687，若手研究（B）JP26730035）の支援による．

参考文献

[1] Tanaka, M. and Tatebe, O.: Workflow Scheduling to Minimize Data Movement using Multi-constraint Graph Partitioning, *Proceedings of the 12th IEEE/ACM International Symposium on Culster, Cloud and Grid Computing (CCGrid 2012)*, pp. 65–72 (online), DOI: 10.1109/CCGrid.2012.134 (2012).

[2] 藤森俊匡，塩川浩昭，鬼塚 真：分散グラフ処理におけるグラフ分割の最適化，情報処理学会論文誌：データベース (TOD72)，Vol. 9, No. 4, pp. 46–56 (2016).

[3] 永持 仁：グラフの最小分割問題に対するアルゴリズム，電子情報通信学会論文誌 D，Vol. J86-D1, No. 2, pp. 53–68 (2003).

[4] Okuyama, T., Okita, M., Abe, T., Asai, Y., Kitano, H., Nomura, T. and Hagihara, K.: Accelerating ODE-based Simulation of General and Heterogeneous Biophysical Models using a GPU, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 8, pp. 1966–1975 (2014).

[5] PhysioDesigner.org: About Physiological Hierarchy ML (PHML), `http://physiodesigner.org/phml/index.html` (accessed on 2017-2-13).

[6] Karypis Lab: ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering, `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview` (accessed on 2017-08-07).

[7] Karypis, G. and Kumar, V.: Multilevel Algorithms for Multi-constraint Graph Partitioning, *Proceedings of the 1998 ACM/IEEE Conference on Supercomputing (SC 98)*, San Jose, CA, pp. 1–13 (1998).

[8] Schloegel, K., Karypis, G. and Kumar, V.: Parallel Multilevel Algorithms for Multi-constraint Graph Partitioning, *Proceedings of 6th International European Conference on Parallel Processing (Euro-Par 2000)*, Berlin, Heidelberg, pp. 296–310 (2000).

[9] Traub, R. D., Contreras, D., Cunningham, M. O., Murray, H., LeBeau, F. E., Roopun, A., Bibbig, A., Wilent, W. B., Higley, M. J. and Whittington, M. A.: Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts, *Journal of Neurophysiology*, Vol. 93, No. 4, pp. 2194–2232 (2005).

[10] Haraguchi, R., Ashihara, T., Namba, T., Tsumoto, K., Murakami, S., Kurachi, Y., Ikeda, T. and Nakazawa, K.: Transmural dispersion of repolarization determines scroll wave behavior during ventricular tachyarrhythmias, *Circulation Journal*, Vol. 75, No. 1, pp. 80–88 (2011).

[11] Graham, R. L.: Bounds on Multiprocessing Timing Anomalies, *SIAM Journal on Applied Mathematics*, Vol. 17, No. 2, pp. 416–429 (1969).

[12] Ohtsu, S., Nomura, T., Uno, S., Maeda, K., Hayashida, Y. and Yagi, T.: A large scale simulation of excitation propagation in layer 2/3 of primary and secondary visual cortices of mice, *Proceedings of the 37th Annual International Conference of Engineering in Medicine and Biology Society (EMBC 2015)*, Milano, Italy, pp. 3957–3960 (2015).

[13] Sandia National Laboratories: Zoltan: Parallel Partitioning, Load Balancing and Data-Management Services, `http://www.cs.sandia.gov/zoltan/Zoltan.html` (accessed on 2017-08-20).