D-05

大規模生体モデルを対象とする Flint シミュレーションコードの並列生成

Parallel generation of a simulation code from a large scale biophysical model for Flint

中村 亜希†　吉川 禎†　置田 真生†　萩原 兼一†
Aki Nakamura　Tadashi Yoshikawa　Masao Masao　Kennichi Hagihara

1. はじめに

生理学分野において，生体機能の解析を目的とした研究が盛んである．その研究のひとつに，生体機能を数理モデル化し，計算機上でシミュレーションする研究がある．シミュレーションの対象は，細胞 1 つから組織，臓器など多階層にわたり，また特定の機能に限定されない．そのため相互に作用しあう複数の生体機能を統合的に扱うことのできる生体シミュレーションが求められている．

多階層かつ多機能な生体モデルの統合開発環境 PhysioDesigner [1][2] は，PHML（Physiological Hierarchy Markup Language）により記述された生体モデル（以降，PHML モデル）[3] を入力とするシミュレータとして Flint [4][5] を提供する．PHML は XML に基づく記述言語で，汎用的な生体機能を，階層構造を持つ数理モデルとして記述できる．Flint は PHML モデルを対象として，複数の要素プロセッサ（以降，PE）を用いた並列計算によりシミュレーション時間の短縮を図る，汎用生体シミュレータである．

Flint によるシミュレーションは，PHML モデルからシミュレーションコード（以降，SC）の生成および生成した SC の実行に分かれる．Flint は，SC の生成および実行ともに，実行時間の短縮を目的として並列処理を利用する．SC の生成において，Flint は PHML モデル内の数式を C 言語の命令に変換する．そのためにまず，モデルから抽出した数式についてデータオブジェクト（以降，DO）を特定の単一 PE が生成する．DO は数式の情報や自身が依存する他 DO の情報などを保持する．ここで，正しいシミュレーション結果を得るためには，数式の依存関係に則って，SC 内での計算順序を決定する必要がある．そのため，Flint は DO を頂点，DO 間の依存関係を辺とする細粒度の依存グラフを構築する．このグラフを部分グラフに分割して，各 PE へ通信によって分配し，以降を並列処理する．グラフの分割により頂点間の依存関係が切断される場合には，切断辺を SC 内の通信命令へと変換する．各 PE は担当する部分グラフ内の DO の依存関係を考慮して，計算順序，通信のタイミングを決定する．DO の持つ数式を SC 内の命令に変換することにより，モデル内の数式が表す各種変数値の時系列を求める SC を生成する．生成した SC を並列実行することで，時間発展型シミュレーションの結果が得られる．この細粒度グラフを介して SC を生成する手法を以降では FGM と表す．

Flint の SC 生成では，モデルの解析から分割を行うまでが単一 PE による逐次処理となっている．大規模なモデルを入力した場合，この逐次処理が性能ボトルネックとなり，多数の PE を用いても台数効果は低い．生理学の発展にしたがって，生体モデルは複雑化かつ大規模化の傾向にあり，SC 生成に要する時間が問題となると予測している．

そこで本研究では，DO の生成前に，PHML の階層構造を利用し，モデルを粗粒度なグラフと解釈して分割する手法（以降，HGM）を提案する．Flint は，DO の依存グラフ生成において，中間表現として二階層の依存グラフを生成する．HGM では，このうち抽象度が高く粗粒度な上階層を分割し，DO の生成以降を並列処理する．HGM は FGM より早い段階で並列化するため，SC 生成における並列処理の割合が増加し，SC 生成時間が削減できる．HGM を実現するには，グラフ分割の改変と部分グラフの SC への変換の 2 つの改編が必要となる．前者については，吉川らの報告がある[6]．本稿では後者について報告する．

部分グラフを SC へ変換するにあたり，2 つの課題がある．まず，（1）SC 内の通信命令の生成手段を新規に構築しなければならない．FGM では通信命令を生成するため，DO の依存関係を基に，SC 実行時の通信相手となる PE を全 PE に対して問い合わせる．しかしながら，HGM において各 PE は割り当てられた部分グラフに属する DO のみ生成するため，他の PE で生成される DO の情報，通信すべき DO を問い合わせることができない．次に，（2）HGM では FGM と比較して SC の実行における通信が増加し，通信パターンが複雑化する傾向にある．

本稿では，この 2 つの課題をそれぞれ以下の方針で解決した．（1）に対して，各 PE は担当する部分グラフに対し，切断辺で隣接する袖領域の DO も部分グラフに含める．追加した袖領域の DO には通信に必要な情報を付与する．通信命令を生成する際には，他の PE に問い合わせることなく，この袖領域の DO を通信命令に変換する．また（2）に対しては通信パターンの複雑化による実行性能の低下を避けるため，HGM においては積極的に冗長計算を追加することで通信を削減する．

2．Flint

本節では，本研究において対象とする PHML モデルおよびそのシミュレータである Flint について述べる．

2.1　入力：PHML モデル

2.1.1　PHML

PHML モデルは多様な生体機能とそのダイナミクスを記述できる．そのうち，Flint が対象とするのは常微分方程式（以降，ODE）で表現された生体機能の時間変化のシミュレーションである．PHML では，生体モデルをモジュールおよび依存関係の集合として表現する．PHML モデルの主な構成要素を以下に示す．

- モジュール

†大阪大学大学院情報科学研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

モジュール（以降，MD）は生体部品のかたまりを表し，ポート，Physical Quantity（以降，PQ）および他の MD を内包する．MD はカプセル化されており，内部の PQ および MD は外部の MD から直接参照できず，ポートを介して参照する．

- ポート
  ポートは MD の特徴量を表し，外部の MD から参照できる．その値は，内部の PQ およびポートの転送（あるいは集約）として記述する．
- Physical Quantity
  PQ は生体機能が持つ物理量を表し，その値を定数，変数，代数関数，ODE および条件文を用いて記述する．
- 辺
  辺は PQ およびポート間の参照を表す．全ての辺は有向であり，1 つの辺につき 1 つのデータ依存を意味する．

カプセル化により入れ子になった MD 群は木構造を構成する．以降，この木構造を構成する MD 群を MD 木と呼ぶ．一般的な PHML モデルにおいて，多くの依存関係は MD 木で閉じている．本章では，本研究において対象とする PHML モデルおよびそのシミュレータである Flint について述べる．

2.1.2 ステップ内依存およびステップ間依存

オイラー法およびルンゲ=クッタ法を用いて計算する場合，辺 $\langle y,x\rangle$ が表すデータ依存は，ステップ内依存とステップ間依存の 2 種類に分類できる．この依存の種類は，y の値の計算式が ODE であるか否かによって決定する．y が ODE でない場合，同一のタイムステップ t_q の一連の計算において，y を計算した後に x を計算する依存関係が生じる．このような依存をステップ内依存と呼び，ステップ内依存を表す辺の集合を E_f と表す．一方で y が ODE の場合，x および y の間に t_q 内における依存は存在しない．この理由は，オイラー法の計算方法による．例えば，x の計算式が ODE $\frac{dx}{dy}=f(y)$ である場合，これは

$$\begin{cases} t_q = t_{q-1} + \Delta t \\ x = x_{q-1} + \Delta t \cdot h(y_{q-1}) \end{cases}$$

と計算する．t_q における x の計算には，t_{q-1} における y の値を用いる．ルンゲ＝クッタ法についても同様である．このような依存をステップ間依存と呼び，ステップ間依存を表す辺の集合を E_o と表す．データ依存の種類によって，SC 実行時における通信タイミングの制約が異なる．$\langle y,x\rangle \in E_f$ が切断された場合，y の計算後から同一タイムステップ内の x の計算前までに y の値を通信する必要がある（ステップ内通信と表す）．

一方，$\langle y,x\rangle \in E_o$ が切断された場合，次のタイムステップの開始までに y の値を通信すれば十分である（ステップ間通信と表す）．ステップ間通信はタイムステップの最後に一括化するなどの最適化が可能となる．

2.2 Flint

2.2.1 FGM

Flint の処理手順は 2 つの処理に分かれる．まず入力として与えられた PHML モデル p を読み込んで SC s を生成する（図 1）．

次にその s をコンパイル，実行することでシミュレーション結果を得る．以降 p から s の生成を $g(p)$，s の実行を

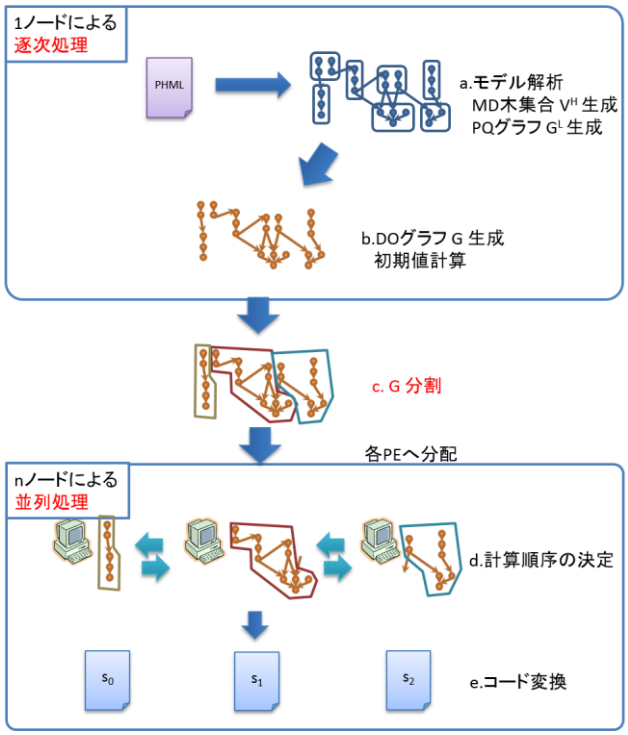

図 1　FGM の処理手順

$e(s)$ と表す．まず $g(p)$ について述べる．Flint は p から SC 生成のための依存グラフ G を生成する（図 1 b）．G の頂点はデータオブジェ（DO）と呼ばれるオブジェクトであり，自分が担当する PQ，どの DO が担当する PQ を参照するかという情報を持つ．初期値計算や計算順序の決定，SC 変換は DO について実行される．G を生成するにあたり，p から粒度の異なる二階層の依存グラフが中間表現として生成される（図 1 a）．まず p に記述されている MD の構造を解析し，MD 木毎にまとめた MDForest を生成する．次に MDForest の各 MD から PQ を抽出する．抽出した PQ 群と p に記述されている辺情報から，PQ を頂点とする粒度の細かなグラフ G^L を生成する．G^L の各頂点について DO を生成することで，G が完成する．完成した G について初期値計算を行い，並列 SC 変換のために G を分割する（図 1 c）．この初期値計算までの処理は特定の単一 PE（以降，ルート PE）上で逐次処理する．

ここで $e(s)$ において，各 PE は割り当てられた部分グラフに属する PQ 群を計算する．したがって，分割時の切断辺は s における通信に変換される．一般に並列処理において通信は性能ボトルネックとなりうるため，切断辺が少なく，かつ部分グラフの頂点が均衡するような分割手法として，multilevel k-way partitioning algorithm [7] を用いる．

分割によって得られた部分グラフを $G_0, G_1, \ldots, G_n$ $(n = |PE| - 1)$ と表す．分割後，通信によりルート PE から各 PE へ部分グラフを分配する．各 PE において G_k $(0 \le k \le n)$ に対するトポロジカルソート[8] を実行し，計算および通信のスケジュールを決定する（図 1 d）．通信のスケジュールを決定するにあたり，通信を必要とする DO は，生成時に与えられた ID を基にして全 PE に通信相手を問い合わせる．決定したスケジュールをもとに SC s_k $(0 \le k \le n)$ へ変換する（図 1 e）．ODE で表される PQ 値の計算にはオイラー法あるいはルンゲ=クッタ法を用いる．

最後に $e(s)$ について述べる．各 s_k を 1 つの PE に割り当て，並列に実行する．s_k には割り当てられた PQ 群の，オ

イラー法（ルンゲ=クッタ法）による 1 タイムステップの計算内容が記述されている．$e(s)$ はユーザが指定したステップ幅（単位時間），ステップ数で実行し，各 PQ の時系列を出力する．この出力が時間発展型シミュレーションの結果となる．

2.2 FGM における課題

FGM には以下の 2 つの問題がある．

（1） 逐次処理部による性能ボトルネック

（2） ルート PE の主記憶容量の不足

（1） について，DO グラフ G を分割するまでの処理はルート PE による逐次実行であり，性能ボトルネックとなっている．また（2） について，2.2.1 節で述べた通り，FGM では，ルート PE による DO の生成処理で大量に DO を生成するため，主記憶の消費が急増する．現在，fsk と呼ばれる心室筋細胞モデルを FGM で SC 生成すると，もっとも高速な 8 並列でもおよそ 2 時間要し，ルート PE では約 60GB の主記憶を消費する．大規模化に伴い生成すべき DO の数が増加すると，将来的にはルート PE の主記憶容量が不足し，SC 生成が不可能となる．

本章では，本研究において対象とする PHML モデルおよびそのシミュレータである Flint について述べる．

3． Hierarchical Graph Mediation

HGM（Hierarchical Graph Mediation）は並列 SC の生成において，粒度の粗い MD 木の集合 V^H を分割し，DO 生成から SC 変換までの処理を分散する手法である．以下の 3.1 節では HGM の概要について説明する．また，3.2 節において，HGM を実現する上での課題について述べる．

3.1 HGM の概要

HGM による Flint の処理手順を図 2 に示す．

まず，入力されたモデル p を解析し，MD を抽出して，MD 木の集合 V^H を生成する（図 1 a）．MD 間の依存関係の多くが木内で閉じている事を利用し，MD 木単位で分割することにより，分割時に切断する辺の数を削減する．

次に V^H を分割し，通信により分割情報を全 PE に渡す（図 1 b）．分割情報とは，全 MD 木の ID 及びそれを処理する PE の情報である．このとき，通信すべき情報を分割情報のみとするため，予め V^H を全 PE で生成する．各 PE は分割情報から全 MD の担当 PE のリストを生成する．

PQ のグラフ $G^L = (V^L, E^L)$ を生成中，このリストをもとに，割り当てられた MD に所属する PQ 群 ${V^L}_k \subseteq V^L$ $(0 \leq k \leq n)$ を確認する（図 1 c）．G^L をもとに DO を生成する際，割り当てられた PQ $v \in {V^L}_k$ について DO を生成し，$G_k = (V_k, E_k)$ を作成する（図 1 d）．以降各 PE は G_k について初期値計算，計算順序の決定を行い，${s'}_k$ へ変換する（図 1 e）．

以上より，HGM では FGM よりも並列処理部が増加するため，逐次処理部による性能ボトルネックが軽減できる．また，各 PE は割り当てられた部分グラフの DO を生成するため，ルート PE の負荷を分散できる．

3.2 HGM における課題

HGM の実装において，以下の 2 つの課題がある．

（1） 生成された並列 SC s' の実行性能の低下

（2） s' における通信命令の生成に既存手法を適用できない

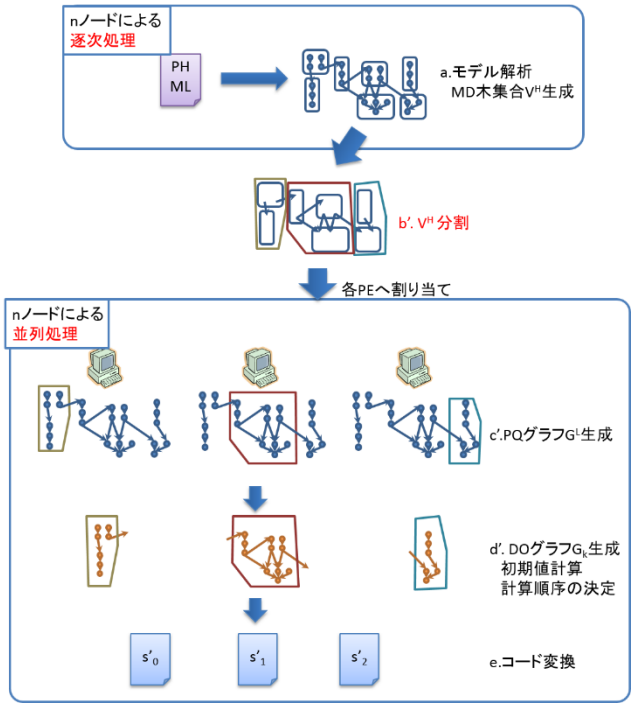

図 2 HGM の処理手順

まず（1） について，分割対象が G から V^H となることで，s' の実行性能が損なわれる可能性がある．これは，MD 木を跨ぐ依存関係が存在するためである．FGM では粒度の小さな G について，切断辺が少なく，かつ部分グラフの頂点数が均衡する分割により s の最適化を施している．一方，HGM ではG よりも粒度の粗い V^H を分割するため，切断辺の増加および頂点数の偏りが生じる．その結果，FGM と比較して s' の実行性能が低下する．

次に（2） について，DO は，参照すべき PQ を担当する DO の ID を持つ．FGM では，参照する DO の ID を基に，全 PE に対して s' 実行時における通信先を問い合わせる．しかし，HGM において各 PE は割り当てられた部分グラフに属する DO のみを生成するため，参照する DO の ID を持たない．したがって，既存実装の通信スケジューリングを適用できなくなる．HGM を実装するにあたり，s' における通信命令の生成手法を変更する必要がある．

（1） の性能低下を軽減する分割手法については，共同研究者である吉川ら[6] が担当している．この分割手法では，節で述べたステップ内通信を削減し，ステップ間通信のみとする．これにより通信のタイミングをステップ終了時に一括して実行することが可能となる．

3.3 通信命令の生成手法

HGM では ${G^L}_k$ に袖領域 $S({G^L}_k) = \{v \mid v \in V^L \cap \overline{{V^L}_k}, \langle u, v\rangle \cup \langle u, v\rangle \in E,\ u \in {V^L}_k\}$ に属する頂点を追加し，この追加された頂点を基に通信命令の生成処理を実装する．$v \in S({G^L}_k)$ から生成された DO に通信に必要な情報を持たせることにより，各 PE は FGM のように通信先を問い合わせることなく独立に通信命令を生成する．

4. 冗長計算

4.1 冗長計算

HGM では FGM と比較して，SC 実行時における通信回数が増大する傾向が強い．これは，高階層 V^H の分割が，

低階層 G^L においては複数の辺切断に相当するためである．特に切断辺が $e \in E_f$ の場合は，大規模並列環境における実行で問題となるステップ内通信の通信パターンが複雑化し，SC 生成および生成された SC の実行における処理時間増大の原因となる．

これを避けるため，HGM では冗長計算の追加によるスップ内通信の削減を基本方針とする．冗長計算とは，切断辺で隣接する頂点 $v \in S({G^L}_k)$ および v を計算するために必要な頂点を，v を担当しない PE においても計算することである．現在の Flint も冗長計算の機能を備えている．ただし，それが通信コストの大きい環境における効率化オプションであることに対し，HGM では SC 実行時の通信をステップ間通信のみとするため，積極的に冗長計算を利用する．通信を各ステップの最後に一度だけとすることで，祖粒度並列処理を実現し，SC 実行時間の増大を避ける．

4.2 冗長計算の実装

HGM では 3.2 で述べた通信同様，既存手法で冗長化される DO を PE 間で共有することができない．よって，分割する際，冗長計算する場合には，その冗長化された MD を担当する PE の情報も分割情報として全 PE に与える．これにより，冗長化された MD に属する頂点は担当する各 PE の ${V^L}_k$ に所属する．

また，冗長な DO を生成する際には，分割情報から同じ DO を持つ PE の情報を持たせる．これは，3.3 で DO から通信命令を生成する際，その DO が通信先にも冗長化され，通信する必要がない場合があるためである．DO が冗長化されている場合には，それが通信先にも冗長化されているかを確認し，通信すべきかを判断する．

5. 実験

本節では以下の観点から提案手法を評価する．

- SC 生成時間
- SC 生成において 1PE が要する主記憶消費量

実験には 48 コアの共有メモリ計算機 1 台（AMD Opteron（tm）Processor 6174 2.2Ghz 12-core4，主記憶 256GB）を用いる．OS は Ubuntu 12.04.1，コンパイラは gcc 4.6.3，MPI のバージョンは OpenMPI 1.4.3 を利用している．使用したモデルは心室筋細胞モデル bm64（MD 木数：約 20 万，DO 数：約 1890 万），および同じく心室筋細胞モデル fsk（MD 木数：約 46 万，DO 数：約 3959 万）である．この 2 つのモデルは心室筋細胞の膜電位ダイナミクスを再現したモデルである．これらの違いは，bm64 の MD 木は 99%以上が入出力にステップ内依存を含まない独立したものであるのに対し，fsk は全ての MD 木がステップ内依存で接続している．

5.1 シミュレーションコード生成時間

図 3 に bm64 の SC 生成時間，図 4 に fsk の SC 生成時間をそれぞれ示す．各モデルでの HGM を適用した場合の生成時間に関して，HGM において並列化される処理（DO 生成及び初期値計算）の処理時間が分割数に応じて短縮できている．

表 1 に bm64 の並列化効率，表 2 に fsk の並列化効率を示す．並列化効率を比較すると，どの分割数においても性能は向上しており，最大で 7 倍の性能向上が見られる．また，HGM では bm64 の分割数 2 および分割数 4，fsk の全ての分割数で並列化効率が 1 を越えた．これは，分割対象のグラフ頂点が DO から MD 木へと変化し，頂点数がおよそ 1/90 程度に減少したことにより，分割時間が短縮されたためである．

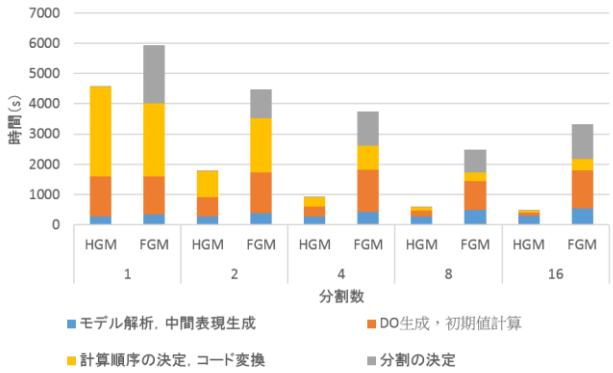

図 3　コード生成時間（bm64）

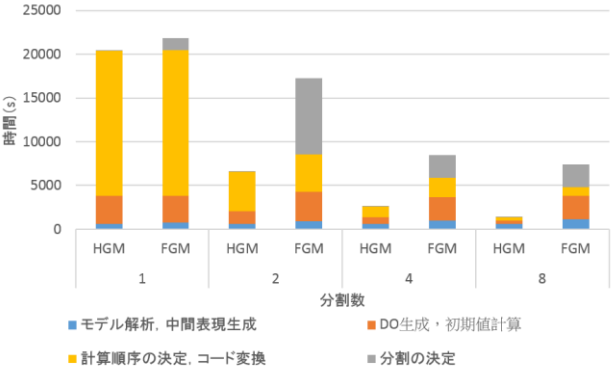

図 4　コード生成時間（fsk）

表 1　並列化効率（bm64）

分割数	1	2	4	8	16
HGM	1.00	1.28	1.24	0.96	0.60
FGM	1.00	0.49	0.29	0.22	0.08

表 2　並列化効率（fsk）

分割数	1	2	4	8
HGM	1.00	1.55	1.97	1.86
FGM	1.00	0.63	0.65	0.37

正しく動作するモデルの試行錯誤のためには，繰り返し SC を生成しシミュレーションすることが必要となる．FGM では fsk の SC 生成に 2 時間以上要していたが，HGM では 30 分未満となり，より実用的な時間で Flint を使用可能となった．

5.2 主記憶消費量

bm64 および fsk の SC 生成時の 1PE の主記憶最大消費量を図 5，および図 6 に示す．既存手法では，ルート PE が分割数によらず全 DO を生成するため，主記憶の消費量はほぼ一定となる．既存手法，HGM のどちらも，SC 生成処理における 1/3 程度の主記憶を中間表現の生成に消費する．これを差し引き，DO の消費する主記憶量に着目すると，

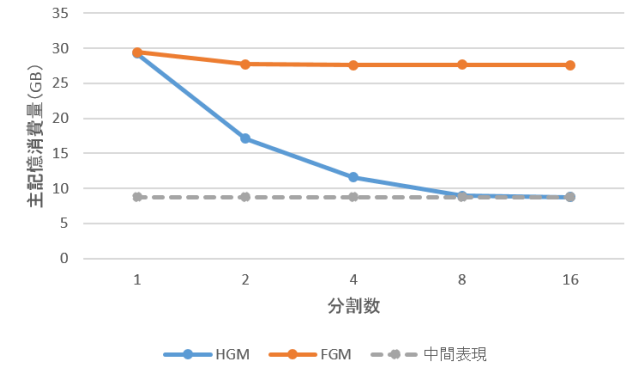

図 5　1PE が消費する最大主記憶量（bm64）

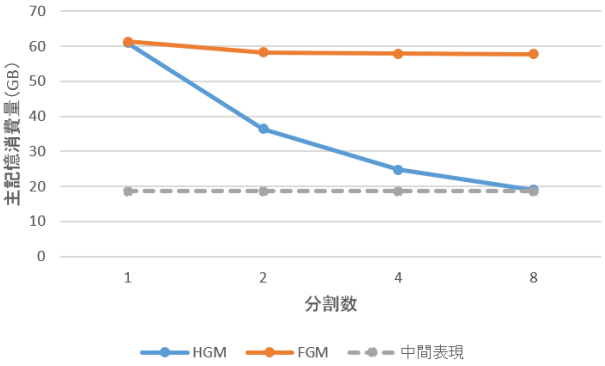

図 6　1PE が消費する最大主記憶量（fsk）

HGM では分割による DO の生成数に伴い 1PE が消費する最大時の主記憶量を削減できている．

HGM により，1PE が消費する主記憶量を 2/3 程度削減することに成功した．しかし，今後もモデルの大規模化は進み，いずれは中間表現の生成の段階で主記憶量が不足する．今後の課題として，分割後に生成する PQ グラフG^L に関しても，各 PE は担当する頂点のみに限定した部分グラフを生成することが必要となる．

6. まとめ

本研究は，生体シミュレータ Flint において，大規模な生体モデルからのシミュレーションコード生成を高速化する手法 HGM を開発した．Flint は，中間表現として，階層ごとに粒度の異なる二階層グラフを入力モデルから生成する．HGM では粗粒度グラフを分割し，その情報を用いて細粒度グラフからプロセスごとに処理すべき頂点を限定する．

このとき，細粒度グラフにおいて切断される辺は並列シミュレーションにおける通信へと変換する必要がある．この通信命令の変換処理を新規に実装する．分割された部分グラフに切断辺で隣接する頂点を袖領域として追加し，その袖領域に属する頂点に通信に必要な情報を付与する．通信命令の変換処理は袖領域の頂点に対して実行する．

また，HGM では冗長計算を積極的に利用することで，コード内の通信を削減し，シミュレーション実行時間の増大を避ける．

HGM を適用したコード生成について，従来の Flint におけるコード生成と比較した．その結果，並列化した処理においてコード生成時間，主記憶消費量ともに台数効果を向上し，HGM の適用によりスケールアウトが実現できることを確認した．

今後の課題として，中間表現である細粒度グラフの生成を部分グラフに限定し，主記憶の消費量をさらに削減する必要がある．

謝辞

本研究の一部は科学研究費補助金（若手研究（B）26730035，新学術領域研究（研究領域提案型）25136711）および文部科学省「医・工・情報連携によるハイブリッド医工学産学連携拠点整備事業」の支援による．

参考文献

[1] PhysioDesigner (2013). http://physiodesigner.org/.

[2] Asai, Y., Abe, T., Okita, M., Okuyama, T., Yoshioka, N.,Yokoyama, S., Nagaku, M., Hagihara, K. and Kitano, H.:Multilevel Modeling of Physiological Systems and Simulation Platform: PhysioDesigner, Flint and Flint K3Service, *Applications and the Internet (SAINT), 2012IEEE/IPSJ 12th International Symposium on*, pp. 215-219 (2012).

[3] PhysioDesigner.org: About Physiological Hierarchy ML(PHML), http://physiodesigner.org/phml/index.html.

[4] Heien, E., Okita, M., Asai, Y., Nomura, T. and Hagihara,K.: insilicoSim: an Extendable Engine for Parallel Heterogeneous Biophysical Simulations, *Proceedings of the 3rd International Conference on Simulation Tools and Techniques* (2010).

[5] 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. 99, No. PrePrints (2013).

[6] [6] 吉川　禎，中村亜希，置田真生，安部武志，浅井義之，北野宏明，野村泰伸，萩原兼一：汎用生体モデルの高速な並列シミュレーションのための階層的依存グラフの自動分割，情報処理学会研究報告，No. 2014-HPC-143 (2014). 7pages.

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

[8] Reynolds, R. G.: An introduction to cultural algorithms, *Proceedings of the third annual conference on evolutionary programming*, World Scientic, pp. 131-139 (1994).