

i アプリによる数値計算実アプリケーションの実装例

程 暁 紅^{†1,*1} 沼野 なぎさ^{†1,*2}
高 田 雅 美^{†1} 城 和 貴^{†1}

携帯電話に代表される携帯情報端末は、待機状態における余剰計算リソースを有しており、これらを集約することができれば新たな計算リソースの創出につながる。携帯情報端末のハードウェア技術革新は日進月歩であり、数年前の PC のスペックに迫るような携帯情報端末が出現することは必然である。本論文では、これら携帯情報端末を分散処理システムとして構築していくための一事例として、携帯電話に対する実アプリケーション・プログラムの移植方法と移植した結果の実行性能について報告する。その結果、現状の携帯電話の言語処理系には問題があるものの、近い将来携帯情報端末による分散処理システムの構築が実現可能であることを示す。

Implementation of a Practical Number Crunching Application for Cellular Phones

GYOKUKOU TEI,^{†1,*1} NAGISA NUMANO,^{†1,*2}
MASAMI TAKATA^{†1} and KAZUKI JOE^{†1}

PDAs such as cellar phones have surplus computing resource when they are in a standby mode. If such surplus computing resource are to be consolidated, we will have a new class of computing resource. The hardware technologies for PDA are innovated day by day, so a highly specialized PDA that is almost equivalent to existing PCs will appear in the near future. In this paper, we propose a distributed processing system model by many PDAs. For the sake of a case study, we explain the porting methods of a real application program to a cellra phone and show some performance results of the ported application. As a result, we find the current language specification for the cellar phone has a restriction, but we are sure that the restriction will disappear in a very near future so that the distributed processing systems by many PDAs are realizable.

1. はじめに

日々進歩する携帯情報端末は、他の携帯情報端末をはじめ屋外や電車、自動車等、あらゆる時間や場所でネットワークに接続可能であり、ユビキタスネットワーク社会の構築に貢献している。一方で、多くの携帯情報端末は、計算資源という観点からは十分に活用されておらず、待機状態の余剰計算リソースが存在する。これらの余剰計算リソースを集約し、分散処理システムを構築することができれば、携帯情報端末の新たなシーズとなりうる。

本研究では、携帯情報端末の中でも利用者の多い携帯電話に着目する。特に、利用者数が圧倒的に多い¹⁾ NTT DoCoMo 社の携帯電話を対象とする分散処理システムを検討する。なお、他の携帯電話においても、NTT DoCoMo 社同様、計算資源を有しており、本研究において検討するシステムを構築することは可能である。

NTT DoCoMo 社の携帯電話²⁾ には、Java の実行環境の一種である DoJa³⁾ を用いた Java 仮想マシンが搭載されており、i アプリ⁴⁾ という名のアプリケーションを動作させることが可能である。この i アプリを用いた分散処理をすることによって、大規模な科学技術計算を行うことが可能である。すでに我々は、島モデルを用いた分散遺伝的アルゴリズム (GA: Genetic Algorithm)^{5),6)} を i アプリとして開発することによって、科学技術計算が可能なることを確認している⁷⁾。また、FFT や、AGM (算術幾何平均法) による円周率計算¹⁴⁾ を i アプリで実装し、基本的な数値計算が十分可能であることも報告している⁸⁾。

我々の最終的な目標は、携帯電話等の小規模な端末で構成される大規模な分散処理システムや Grid であるが、大規模なシステムを構築する前に、携帯電話による実際の数値計算アプリケーションを実装し、その実現可能性を検証することが必要である。そこで本論文では、代表的な分子動力学アプリケーション GROMACS⁹⁾ を i アプリで実装するための手法について報告する。GROMACS は、一般的な計算機の余剰計算リソースを活用するために開発された Folding@home¹⁰⁾ で用いられているアプリケーションである。この GROMACS を携帯電話で実行させることによって、Folding@home と同等のシステムを構築していくことが我々の最終的な目標である。

^{†1} 奈良女子大学大学院人間文化研究科

Graduate School of Humanities and Sciences, Nara Women's University

*1 現在、株式会社エルゼウスソリューションズ

Presently with elZeus solutions corporation

*2 現在、NEC システムテクノロジー株式会社

Presently with NEC System Technologies, Ltd.

以下、2 章において、i アプリを利用可能な携帯電話のスペックについて紹介した後、3 章では、一般的な PC を用いた分散処理システムとして知られている Folding@home で用いられている GROMACS について説明する。4 章では、GROMACS を i アプリとして実装する手法について報告し、5 章では実装結果と実装により判明した問題点について述べる。

2. 携帯電話の能力

2.1 i アプリ

i アプリ⁴⁾は、2001 年に NTT DoCoMo 社²⁾の携帯電話として発売された 503i シリーズから搭載されているアプリケーションである。このアプリケーションは、DoJa と呼ばれる Java 実行環境の一種の下で作成されたプログラムを実行するための Java 仮想マシンが搭載されており、DoJa で作成されたアプリケーションを携帯電話にダウンロードさせることで実行することができる。

2.2 DoJa

DoJa とは、Java の J2ME CLDC¹²⁾をベースの言語仕様として構築された処理系である。そのため、通常の計算機で開発する際に用いる J2SE¹³⁾のクラスが使用できないことがある。たとえば、DoJa-3 シリーズまでは、浮動小数演算が不可能であった。しかし、2005 年の初夏に公開された DoJa-4.0 において、CLDC1.1 をサポートするように機能拡張されたため、浮動小数演算が可能となっている。

2.3 スペックの制限

携帯電話でシステムの開発を行う場合、ハードウェア・スペックの制限を考慮する必要がある。本論文では、DoJa-4.0 を用いて i アプリの開発を行う。

表 1 は、DoJa-4.0 が使用可能な代表機種種のスペックを表す。使用できるメモリは 8 MB～10 MB である。そのため、この容量内でプログラムを実行する必要がある。

プログラム・サイズに関しては、DoJa を用いて開発されたアプリケーションは Jar ファイルに保存されるため、実質的に Jar ファイルの上限がプログラム・サイズの上限と見なせる。

表 2 は、DoJa の各バージョンに対応する機種名と保存可能な Jar ファイルのサイズを表す。表 2 より、DoJa-4.0 の対応機種では、Jar ファイルのサイズが 100 KB である。DoJa-1.0 の 10 KB に比べれば、非常に大きくなっているが、一般的な計算機と比較すると依然として制限が大きい。最新の DoJa-5.0 では 1 MB となっており、本論文で報告する実装よりも規模の大きいものが実装可能となっている。この傾向は今後も続く想定され、本論文で述べるような数値計算分野への適用が容易になる方向といえる。

表 1 DoJa-4.0 を実行可能な機種種のスペック

Table 1 Hardware specification of mobile phones with DoJa-4.0.

機種名	CPU	メモリ (ヒープ容量)
N901iS	200 MHz	9,216 KB
P901iS	200 MHz	9,216 KB
F901iS	216 MHz	3,000/5,000 KB
D901iS	216 MHz	3,000/5,000 KB
SH901iS	216 MHz	8,000 KB
N902i	400 MHz	9,218 KB
P902i	400 MHz	8,192 KB
F902i	400 MHz	4,500/3,500 KB
D902i	400 MHz	4,500/3,500 KB
SH902i	400 MHz	10,752 KB

表 2 DoJa の Jar ファイルサイズの上限

Table 2 Upper limit size of DoJa Jar files.

環境	機種	Jar サイズ
DoJa-1.0	503i	10 KB
DoJa-2.0	504i	30 KB
DoJa-3.0	505i, 506i	30 KB
DoJa-3.5	900i	100 KB
DoJa-4.0	901i	100 KB
DoJa-4.1	902i	100 KB
DoJa-5.0	903i	1,024 KB

2.4 余剰計算リソースを用いた分散処理システムの可能性

i アプリに対応した携帯電話の契約台数は非常に多く、その大部分は待機状態である。つまり、携帯電話の計算能力の大部分が余剰資源として放置されているという状況である。また、前節で述べたように、i アプリを搭載した携帯電話は、新機種開発とともに、性能やメモリ容量が飛躍的に向上している。そのため、一昔前の PC に近い性能を持っている。大浦はガウスルジャンドル公式をさらに改良した AGM (算術幾何平均法) を用いて円周率を計算するアルゴリズムを報告しているが¹⁴⁾、我々は DoJa-4.0 を用いてこの方法を実装し、円周率を 32 万桁まで求めたことをすでに報告している⁸⁾。また、FFT を実装し問題なく実行することも確認している。これらの実装はいずれも単体の携帯電話用の実装したものであるが、携帯電話の余剰計算リソースを集約し分散システムとして活用することができれば、大規模な分散処理が可能となり、ハイエンドの計算機に劣らない処理能力となると考えられる。

分散処理を利用するアプリケーション分野は多岐にわたり、遺伝子解析、気象予測、暗号解読等の大規模な計算でしばしば用いられている。これらの計算では、扱うデータが膨大であるため、多数の処理系にデータを割り当てて処理をすることが多い。先に述べたように携帯電話はPCに比べ圧倒的に利用台数が多く、その大部分が余剰計算リソースとして眠っているのが現状である。ならば多数の携帯電話を、大規模な分散処理システムとして構築しようというのが本研究の着想である。本研究に着手したところに比べ、モバイル・プロセッサの性能向上には目を見張るものがあることは前節で述べたとおりであるが、プロセッサ本体の性能が向上しても、分散処理システムを構築するには問題点も多い。

そもそも分散処理を行うにはデータ通信が必須である。大規模な問題に適用するならば、多数に分割されたとはいえ、携帯電話個々のデータ通信量もある程度の量を見積もらなければならない。つまり、計算は高速にできても、データ通信に時間がかかったり、データ通信の際に発生するパケット料金という経済的コストの問題があったりした。本研究に着手した 2000 年当時は 64 Kbps のバンド幅しかなく、実用的な分散処理は絶望的に思えたが、2001 年に始まった第 3 世代移動通信サービスでは 384 Kbps に高速化され、平成 20 年 4 月時点での第 3.5 世代では 7.2 Mbps であり、第 4 世代では 1 Gbps が目標とされている¹⁵⁾。また、データ通信の際に発生するパケット料金に関しても、近年パケット料金定額化の傾向が顕著になっており、分散処理を行ううえでのデータ通信の問題は、利用環境のレベルで改善されている。

以上述べてきたように、携帯電話はハードウェア仕様の観点からも、通信インフラの観点からも、大規模な分散処理システムの構成要素となりうる可能性を見せはじめている。大規模な分散処理システムの例として、Folding@home¹⁰⁾がある。Folding@homeは、2000年10月より開始された分散処理プロジェクトで、現在、全世界において100万個以上のCPUが参加している。この対象となるCPUは、一般的なPCのCPUである。そこで、本論文では、携帯電話用のFolding@homeを開発するための第一歩として、そこでの単体主要計算部分を受け持つGROMACSに着目し、単体の携帯電話にGROMACSを実装する。

3. GROMACS

GROMACS (Groningen Machine for Chemical Simulations)⁹⁾ は、オランダの Groningen 大学で開発された分子動力学シミュレーションプログラムであり、GNU パブリックライセンスの下で利用できるフリーソフトである。

GROMACS は、何百万もの粒子に対して、ニュートンの運動方程式、準ニュートン法、Polak-Ribiere による共役勾配法、最急降下法等¹¹⁾を用いた分子動力学を適用し、各粒子

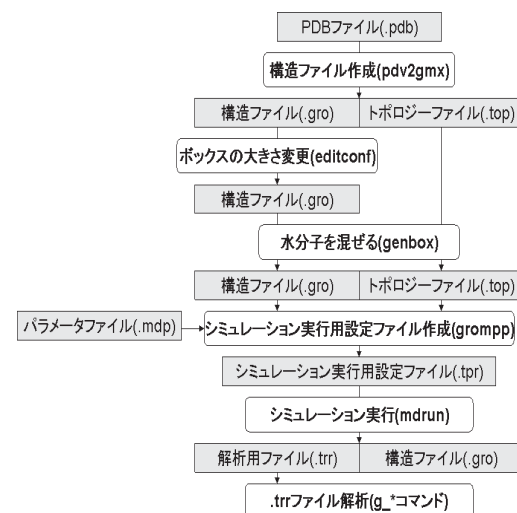


図 1 GROMACS 実行の流れ図
Fig.1 The execution flow in GROMACS.

の動きをシミュレートするソフトウェアパッケージである。主たる利用例としては、たんぱく質や脂質といった複雑に結合された生化学分子のシミュレーションが知られている。また、通常では大規模な計算を必要とする非結合な相互作用を短時間で計算可能なため、非生物学的なものを対象とする多くの研究でも利用されている。

GROMACS のソースコードは主に C 言語で構成されており、その規模は約 20 万行に及ぶ。ソースコードに対するアルゴリズム・レベルでの最適化がなされており、非常にパフォーマンスが高い。また、MPI によって並列化を行うことも可能となっている。

図 1 は、GROMACS の実行を行うコマンドと作成されるファイルとをあわせた流れ図である。GROMACS は以下の流れでシミュレーションを行う。ここで、括弧内はその動作を行うための関数を表す。

1. 構造ファイルの作成 (pdv2gmx)
2. ボックスの大きさの変更 (editconf)
3. 水分子の混入 (genbox)
4. シミュレーション実行用設定ファイルの作成 (grompp)
5. シミュレーションの実行 (mdrun)

6. シミュレーション結果の解析 (g-*)

シミュレーションを行う mdrun の流れは以下のとおりである。

- a) 初期化
- b) データの読み込み
- c) 各種制限
- d) QM/MM (Quantum Mechanics/Molecular Mechanics) 法の適用
- e) (必要時) PPPM (Particle-Particle Particle-Mesh) 法の適用
- f) シミュレーション方法の選択
 - 分子動力学法 (do_md)
 - Polak-Ribiere による共役勾配法 (do_cg)
 - ニュートン法 (do_nm)
 - 準ニュートン法 (do_lbfgs)
 - 最急降下法 (do_steep)
- g) 原子へ変換
- f) のシミュレーション方法のうち、分子動力学法を行うための関数 do_md は、以下の流れを繰り返す。
 - (1) 初期化
 - (2) 原子に働く力の計算
 - (a) 各原子の近接原子の探索
 - (b) 力の計算
 - (i) 非結合分子間に働く力の計算
 - (ii) 結合分子間に働く力の計算
 - (c) クーロン力の計算
 - (3) 原子の速度と座標データの更新
 - (4) ステップを進め、(1) へ

4. i アプリへの移植方法

4.1 C 言語と DoJa

本論文で移植対象とする GROMACS は C 言語で開発されている。これを i アプリに移植するためには、C 言語から DoJa で動作する JAVA プログラムに変換する必要がある。以下の項では、移植の際の諸問題とその解決方法について述べる。

4.1.1 ポインタ

一般に C 言語を用いてプログラム開発する場合、ポインタの使用率が高く、GROMACS においても例外ではない。しかしながら、ポインタの利用は DoJa には言語仕様のレベルで含まれていない。ただし、言語処理系の内部的な扱いとして、DoJa でもポインタという概念は存在している。たとえば、間接参照ができるデータ型であれば、DoJa においても、いわゆるポインタとして関数にアドレスを引き渡すことができる。しかし、DoJa で用いることができるポインタの概念と C 言語での概念では違いがあり、GROMACS で利用されているポインタ利用を DoJa で実現するには次のようにした。

GROMACS での最も多いポインタ利用である、関数に配列と配列のインデックス値を指定してアドレスを引き渡す場合、C 言語では `&array[index]` とすればよいが、DoJa では、配列名とインデックス値を別々に送らなければならない。特に注意が必要なのは、関数引渡しでポインタを用いる場合である。C 言語では、引渡し先の関数内で、特に指定しなかった場合、配列ポインタのインデックス値は 0 とされている。一方、DoJa には、この定義がない。そのため、DoJa へ移植を行う場合、引き渡されたインデックス値 0 を関数内の添数に加える必要がある。

また GROMACS でしばしば行われているアドレスの演算に関しては、DoJa には機能がないため、当該アドレスを配列に変換しインデックス値を利用して疑似的に行う。さらに基本データ型変数をポインタとして扱う場合は参照データ型に変換してから使用する必要がある。

4.1.2 構造体

DoJa では、クラスを用いることによって構造体や共用体を代用することが可能である。しかし、構造体や共用体の機能とクラスの機能は同じではない。そのため、1 つの構造体に対して 1 つのクラスを作成する必要がある。この際、カプセル化の観点からは好ましくないが、変数に `public` 修飾を行い、簡易に値の入れ替えを行えるようにする。

この方法で移植を行う場合、クラスファイル数が増えるため、ファイル容量が大きくなるという問題が生じる。また後ほど述べるように実行性能に影響を与えることが判明している。

4.1.3 typedef と extern の特殊な使われ方

C 言語の `typedef` は定義済みの型に対して別の名を定義し、`extern` はあるファイルの中で変数の型宣言に使われた場合、その変数を他の複数のファイルで利用可能なグローバル変数として定義する。

図 2 は、GROMACS で用いられている `typedef` と `extern` の関係図である。これらを用

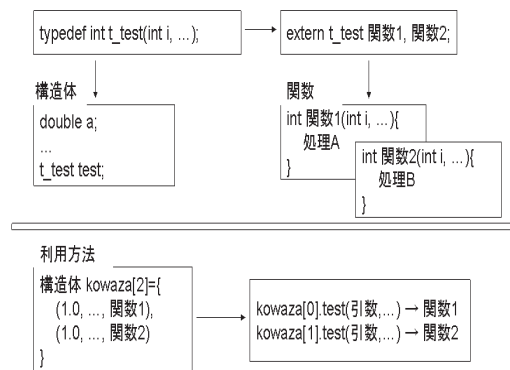


図 2 typedef と extern の利用
Fig. 2 How to use typedef and extern.

いることにより、関数を柔軟に呼び出すことが可能となっている。

たとえば、typedef で関数 `t_test` を宣言し、ある構造体の中にその関数 `t_test` の名で変数 `test` と宣言する。さらに、extern で関数 1 と関数 2 を関数 `t_test` であると定義づける。このように定義された関数は、構造体のデータを作成するときに関数名を代入することによって利用可能となる。使用方法としては、構造体から変数 `test` を呼び、引数を与えることによって、変数 `test` に代入されている関数名の関数が実行できる。

これらを DoJa で実装する場合、構造体の代わりに作成されたクラスの中に関数を作成し、変数 `test` を文字列として扱う。このことにより、変数 `test` を利用した条件分岐が可能となり、呼び出す関数を変えることができる。

4.2 メモリ使用量の削減

メモリ使用量を必要最低限にとどめるために、無駄な変数や配列を宣言しないことと、ガーベジコレクションを適切に行うことがあげられる。無駄な変数や配列を宣言しないために、変数の再利用を行うことによって、あらかじめ確保されるメモリ量を減らすことが可能である。ガーベジコレクションとは、プログラムが動的に確保したメモリ領域のうち、不要となった領域を開放することである。J2SE および DoJa では、使用しなくなった変数や配列に `null` を代入し、`System.gc()` を実行することによって当該メモリ部分を開放できる。

4.3 Jar ファイルのサイズの縮小

携帯電話で扱うことができる Jar ファイルのサイズは、一般的な計算機と比べ小さい。そ

のため、DoJa-4.0 を用いて開発を行う際、プログラムソースを簡潔にするとともに、オブファスケット処理¹⁶⁾を行うべきである。

オブファスケットとは、本来、クラス名や変数名を極端に短い名前に変換し、可読性を下げることによって、プログラムの不正改竄を阻止することを目的として開発されたものである。一方、オブファスケットを使うと、短い名前を用いるため、Class ファイルが軽くなる。Jar ファイルのサイズに上限がある携帯電話では、Class ファイルの容量の削減が重要である。また、i アプリをダウンロードする際の通信量を節約するという点においても有効である。なお、開発支援ソフトとして、オブファスケット処理専用のツール¹⁷⁾も開発されている。

オブファスケットを導入する際には、オブティマイズに留意しなければならない。オブティマイズとはコードの最適化のことであり、より効率的なコードを生成するために必要なものである。オブファスケットにはオブティマイザが組み込まれており、これもまた軽量化につながるものであるが、意図しない動作にコードが変更されてしまうことがある。そのため、オブティマイズを行うときには、意図どおりの動作が実行されているか確認することが必要となる。意図しない動作となってしまった場合には、オブティマイズを行わないよう指定できるツールを使うべきである。

本実装においてオブファスケットを使うことで、5 章で述べるように不要なクラスファイル読み込み時間の削減が観測されている。

4.4 計算結果の表示と保存

携帯電話を用いた演算の結果の表示ならびに保存方法として、画面への出力、スクラッチパッドの利用、サーバへの転送による保存があげられる。

携帯電話の画面上に文字を配置するには、Panel のコンポーネントである Label を使用する方法と TextBox を使用する方法がある。Label を使用する場合は、1 つの Panel に対して表示できる Label の数が決められており、N901i^{*1}では 127 Label である。通常のフォントサイズで 1 行 39 文字を表示することができるため、Label では約 5,000 文字までの表示が可能である。これ以上 Label を利用するようにコーディングすると、エラーが生じ、i アプリが停止してしまう。一方、TextBox は、本来、ユーザからの文字入力を行うためのシステムであるが、あらかじめ文字を入力することも可能なため、表示にも使うことができる。TextBox を使うことによって、1 万文字まで表示可能となる。ただし、1 万文字以降の

*1 本段落で説明している様々な制限は N901i で実行した場合に発見されたものであり、DoJa-4.0 の仕様ではない。エミュレーター上での実行では、これらの制限は確認されていない。

文字列は切り捨てられる。切り捨てられた文字列はエラー処理されるわけではない。このため、1 万文字を超える大量のデータの結果表示には不向きである。いずれの方法も計算途中のデータの表示に使うものである。

スクラッチパッドを用いてデータを保存する場合、テキストデータかバイナリデータで保存できる。しかし、スクラッチパッド内のデータはその携帯電話内部でしか利用できず、分散処理システムには適していない。

サーバにデータを保存する場合、i アプリ独自の API である `HttpConnection` と `OutputStream` を使用することにより比較的容易に実行可能となる。この際、サーバではデータ受信のために、CGI や Java 等によるサーバプログラムを待機させておく必要がある。現在 NTT DoCoMo 社の携帯電話の通信には W-CDMA が採用されており、通信速度は上りで 384 Kbps である。これは、ADSL の上り 12 Mbps に比べると非常に遅い。ただし、下りに関しては平成 20 年 4 月現在、7.2 Mbps と高速化されている。また、i アプリで 1 度に送信できる容量は 80 KB 以下である。そのため、この制限以上のデータを送信する場合、データ分割が必要となる。受信にも制限があり、150 KB を超える場合には分割しなければならない。

画面への表示やスクラッチパッドは、演算途中の結果保存としては利用できる。しかし、他の携帯電話やサーバで演算結果を利用できないため、携帯電話を用いた分散コンピューティングでは、ネットワークを介したサーバでのデータ保存が適している。

4.5 算術関数の欠如

C 言語では標準ライブラリの中に算術関数がある。指数関数演算として `exp`、自然対数演算として `log`、べき乗演算として `pow` 等が提供されている。しかし DoJa では多くの算術関数が実装されていない。特に GROMACS の移植に関して、指数関数演算、自然対数演算、べき乗演算の実装が必要になったため、この 3 つの関数の実装を行う。

整数によるべき乗演算関数を作成し、それを自然対数演算で使用する。小数によるべき乗演算関数は、指数関数演算関数と自然対数演算関数を使い実装する。これらにより、DoJa においても C 言語における `exp`、`log`、`pow` 関数の演算が可能となる。

5. 実装と考察

GROMACS はソースコード 20 万行以上に及ぶ巨大なプログラムであり、1 台の携帯電話に実装できるものではないが、分散システムとしての利用であれば、GROMACS の計算の核の部分の実装ができれば、その可能性を示せると考える。本研究ではプロトタイプと

表 3 mdrun におけるアルゴリズムの使用メモリ量

Table 3 Memory sizes required by several mdrun algorithms.

アルゴリズム	メモリ量
MD 法 (md)	350.2 KB
最急下降法 (steep)	377.8 KB
BFGS (lbfgs)	579.0 KB
ニュートン法 (nm)	15,467.1 KB

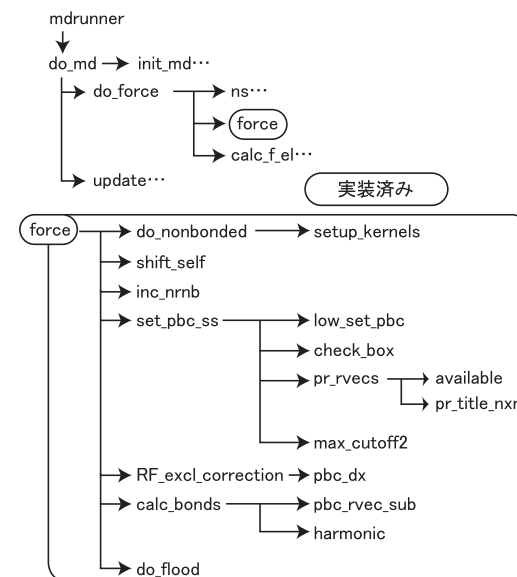


図 3 分子動力学法のコールグラフ

Fig. 3 Call-graph of MD.

して、mdrun の一部関数である `do_md` の移植を進めている。`do_md` は、MD 法のシミュレーションを行うための関数である。関数 `do_md` を選択したのは、GROMACS で最も重要である分子動力学を行う関数であり、表 3 で示されるように MD 法のアルゴリズムが最も使用メモリ数が少ないからである。前章の方針に従って実装したものは図 3 で示すように、関数 `force` 以下の 445 個の関数約 2 万 2 千行である。

この関数 `force` を動作させるために使用したメモリは 158.9 KB となった。DoJa への移

表 4 force 関数の実行時間 (ms)
Table 4 Execution time of force (ms).

試行数	PC (C)	PC (DoJa)	携帯電話
1	4	125	1,429
2	3	125	1,450
3	3	125	1,447
4	2	125	1,483
5	9	141	1,427
6	3	141	1,440
7	3	125	1,412
8	3	125	1,425
9	3	125	1,393
10	8	125	1,469
平均	4.1	128.2	1,437.5

植前が 234.6 KB であり, DoJa-4.0 で使用可能なメモリが 8 MB~10 MB であることから, 余裕を持って移植ができたといえる. 一方, Jar ファイルの容量は 52 KB となっている. これは, DoJa-4.0 の Jar ファイル上限が 100 KB であることから, 容量内に収まっている. DoJa-5.0 では Jar ファイル上限が 1 MB になったとはいえ, 他の関数も実装しなければならないことを考えると, さらなる軽量化が必要となる可能性がある.

表 4 は実装した関数 force の実行時間を, PC ならびに携帯電話で実行した時間を示している. 使用した PC は, CPU が intel (R) Core (TM) SoloCPU 1.05 GHz, メモリが 512MB のものである. 表 4 の PC (C) は, 普通に PC で GROMACS を実行させた時間である. PC (DoJa) は, PC (C) と同じ PC を用い, 携帯電話用に移植した GROMACS を DoJa エミュレータ上で実行させた時間を表す. 携帯電話は, N901iS において DoJa 用に移植した GROMACS を実行させた場合の時間である. 与えたデータ^{*1}は, 647×3 の二次元配列である.

PC 上での DoJa エミュレータと携帯電話実機での実行時間差が 10 倍になっているが, これが携帯電話実機での実行の妥当な処理時間なのかどうか疑問が残る. そこで実機とエミュレータの簡単なベンチマーク・テストを行い分析を試みた.

図 4 は, ループ文でインクリメントを単純に繰り返し処理時間を計測したものである. 図 4 では, 携帯電話実機での処理性能の方が 3 倍以上良いことが見てとれる. このことから, 演算によって計算時間が増加しているとは考えがたい.

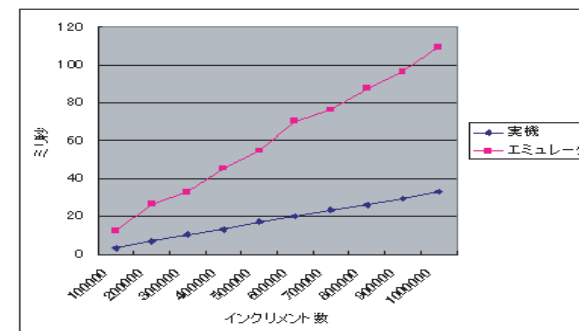


図 4 インクリメント実行時間
Fig. 4 Execution time for increment.

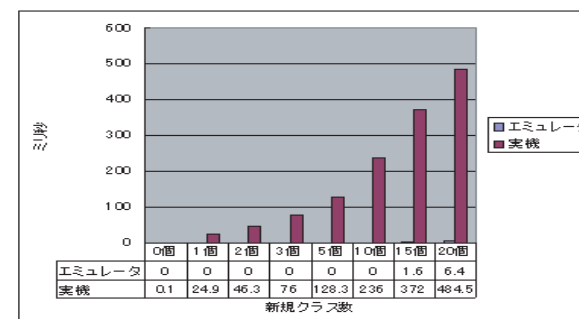


図 5 新規クラス読み込み時間
Fig. 5 Reading time for new classes.

この処理時間の差異がどこに原因するかを考察した結果, 前章の移植方法のいずれかに問題があるとした考えられない. 計算時間への影響が出そうな移植方法は, DoJa 版 GROMACS でのクラスの多用が考えられる. そこで, クラスの読み込みに関して時間を計測してみたのが図 5 である. 図 5 は, 新規クラスの読み込み^{*2}数を変化させて計測したものである. 新規クラスが増えるほど, 携帯電話実機では所要時間が増えていく. 新規クラスを 1 回読み込ませたものは 24.9 ms で, 以後新規クラスを読み込むたびに約 25.0 ms の処理時間を要して

*1 このデータは GROMACS のサンプル・データで, 水分子を表している.

*2 単純に.class ファイルのロードを行う.

いる。一方、エミュレータでは新規クラスの読み込み時間は携帯電話実機の約 $\frac{1}{100}$ となっている。このことから、実機の DoJa では、新規クラスを読み込むことが性能劣化の直接の原因となっていると考えられる。

実際、DoJa に移植された GROMACS では構造体をクラスで書き換えたものが 68 カ所あり、この読み込みに $68 \times 25 \text{ ms} = 1,700 \text{ ms}$ かかる。これは全体の実行時間の平均 1,437.5 ms を超えているが、その理由はオブファスケット等による最適化を施した結果、不必要なクラスファイル読み込みが削除されたためと推測され、実際の force 実行時間は計測できないと結論づける。新規クラスの読み込みに長時間かかることは、DoJa の実装上の問題であり、近い将来解決されると期待したい。

6. ま と め

本論文では、携帯情報端末の普及にともない増加している余剰計算リソースの活用を目的とする、携帯情報端末による分散処理システムのモデルを提案した。この一環として、携帯電話を対象とし、余剰計算リソースを集約して分散処理を行うために、分子動力学シミュレーションプログラム GROMACS の主要計算部分を i アプリに移植した。GROMACS は C 言語で開発されているため、i アプリとして動作させるには DoJa への移植が必要となる。移植に際して、メモリ容量、結果の保存、通信、表示に関する制限あるため、本論文で移植手法についてまとめた。

実装された GROMACS の計算の核の部分の実行性能を検証するために、オリジナルの GROMACS、PC のエミュレータによる移植された GROMACS、携帯電話実機による移植された GROMACS をそれぞれ実行したところ、携帯電話実機での実行時間が予想よりもはるかに長いことが判明した。そこで追加実験を行った結果、DoJa への移植に多用した C 言語の構造体のクラスとしての利用に問題があることが判明した。これは DoJa（今回の実装では 4.0）の実装上の問題であり、改善が期待されるものである。

今後携帯情報端末はデータ通信速度が Gbps に移行し、数 MB～数十 MB のメモリが使える、マルチコアやマルチスレッドの省電力プロセッサが使われるようになると予想される。さらに燃料電池の利用拡大により携帯情報端末の余剰計算リソース集約のニーズが現実のものになると考える。本論文では携帯電話で GROMACS すべてを実行できることを示してはいないが、その可能性を実証したという意味は大きいと考える。今後の研究課題はより新しい携帯電話を使った GROMACS の完全移植であり、それを利用した分散処理システムの構築である。

参 考 文 献

- 1) i モード契約数. www.nttdocomo.co.jp/corporate/ir/finance/
- 2) NTT 公式サイト. www.nttdocomo.co.jp/
- 3) i アプリコンテンツの概要.
www.nttdocomo.co.jp/service/imode/make/content/iappli/about/
- 4) i モードコンテンツ. www.nttdocomo.co.jp/service/imode/make/
- 5) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley (1989).
- 6) Tanese, R.: Distributed Genetic Algorithms. *Proc. 3rd Intl. Conf. on Genetic Algorithms*, pp.434–439 (1989).
- 7) 高田雅美, 柴山智子, 渡辺知恵美, 庄野 逸, 城 和貴: i アプリを用いた数値計算の可能性, 情報処理学会論文誌: 数理モデル化と応用, Vol.46, SIG2 (TOM11), pp.47–55 (2004).
- 8) 程 暁紅, 渡辺知恵美, 城 和貴: i-appli における浮動小数点の演算拡張, 情報処理学会関西支部大会 C-10, pp.171–174 (2004).
- 9) GROMACS グローニンゲン大学. www.gromacs.org/
- 10) Folding@home スタンフォード大学. folding.stanford.edu/
- 11) Scales, L.E.: *Introduction to non-linear optimization*, Springer-Verlag, New York (1985).
- 12) Sun J2ME 規格. java.sun.com/javame/index.jsp
- 13) Sun J2SE 規格. java.sun.com/javase/index.jsp
- 14) 大浦拓哉: 円周率公式の改良と高速多倍長計算の実装, 日本応用数理学会論文誌, Vol.9, No.4 (1999).
- 15) NTT ドコモレポート No.40. www.nttdocomo.co.jp/
- 16) Sun Developer Network. java.sun.com/developer/J2METechTips/2002/tt0226.html
- 17) ProGuard. proguard.sourceforge.net/

(平成 20 年 3 月 27 日受付)

(平成 20 年 7 月 6 日再受付)

(平成 20 年 7 月 23 日採録)



程 曉紅

昭和 48 年生。平成 17 年奈良女子大学大学院人間文化研究科情報科学専攻修士課程修了。平成 20 年奈良女子大学大学院人間文化研究科複合現象科学専攻修了。博士（情報科学）を同大学より取得。ユビキタスコンピュータの活用に関する研究に従事。



沼野なぎさ

昭和 58 年生。平成 20 年奈良女子大学大学院人間文化研究科情報科学専攻修士課程修了。同年 NEC システムテクノロジー株式会社に入社。ステレオマッチングシステムのソフトウェア開発に従事。



高田 雅美（正会員）

昭和 52 年生。平成 13 年奈良女子大学大学院人間文化研究科情報科学専攻修士課程修了。平成 16 年奈良女子大学大学院人間文化研究科複合領域科学専攻修了。博士（理学）を同大学より取得。平成 16 年独立行政法人科学技術振興機構戦略的創造事業の委嘱研究員として、京都大学大学院情報学研究科数理工学専攻数理解析分野にて従事。平成 18 年奈良女子大学大学院人間文化研究科助手。平成 19 年国立大学法人奈良女子大学大学院人間文化研究科助教。数値計算ライブラリの開発、分散メモリ環境を対象とする並列プログラムの開発に関する研究に従事。



城 和貴（正会員）

大阪大学理学部数学科卒業。日本 DEC、ATR 視聴覚研究所（日本 DEC より出向）、（株）クボタ・コンピュータ事業推進室で勤務。平成 5 年奈良先端科学技術大学院大学情報科学研究科博士前期課程入学。平成 8 年博士（工学）を同大学院大学より取得。平成 8 年同大学院大学情報科学研究科助手。平成 9 年和歌山大学システム工学部情報通信システム学科講師。平成 10 年同学科助教授。平成 11 年奈良女子大学理学部情報科学科教授、現在に至る。情報処理学会論文誌数値モデル化と応用編集副委員長。