03-01063

複数携帯端末を利用した分散協調アプリケーションの設計・開発に関する研究

東　野　輝　夫　　大阪大学大学院情報科学研究科教授

概　要

有線・無線混在型のネットワーク環境で分散協調アプリケーションを開発する際に、電池駆動の PDA や携帯端末などに CPU 負荷の高いプログラムをロードすると、電池の消耗により携帯端末の使用に支障が生じる可能性がある。また、携帯端末にロードできるプログラム容量の制限などから、一時にたくさんのモジュールをロードできない場合もある。そこで、ホーム（ゲートウェイ）サーバを用意し、ほとんどの処理はサーバマシンで実行し、入出力結果のみ携帯端末に送り、携帯端末上では I/O のためのプログラムのみを走らせるという実装を考える。しかし、その場合、どのようにサーバに処理を割り振るか、どこまでを I/O 処理として端末に割り振るか、といった問題はアプリケーションごとに考慮する必要がある。

本研究では、Java 言語で書かれた（複数オブジェクト群からなる）プログラムと、各オブジェクトのサイズや実行時の CPU 負荷、オブジェクトの呼び出しに必要なデータ量や各オブジェクトの入出力処理に必要な通信量や通信時間、携帯端末にロード可能なプログラム容量とオブジェクトの呼び出し関係などをもとに、与えられた Java プログラムと同じ動作をする複数サーバマシンと複数携帯端末のプログラム群を自動導出するプログラム変換法を考案した。まず、提案手法ではシミュレーションによりあらかじめ基準となる統計情報を収集する。Java のプログラムを 1 台のマシンで複数回実行し、各モジュール間の通信量や、各モジュールで消費される CPU 時間などを計測する。次に端末側のリソース制限や、どういったコストを最小化するかを示した目的関数を与え、SA（Simulated Annealing）法を用いて目的関数の値が最小になるよう分割を行う。提案手法を実用的なアプリケーションに適用し、分割のコストを比較することで、手法の有用性を確かめた。

1　はじめに

有線・無線混在型のネットワーク環境で分散協調アプリケーションを開発する際に、電池駆動の PDA や携帯端末などに CPU 負荷の高いプログラムをロードすると、電池の消耗により携帯端末の使用に支障が生じる可能性がある。また、携帯端末にロードできるプログラム容量の制限などから、一時にたくさんのモジュールをロードできない場合もある。一方、各携帯端末とは別にホーム（ゲートウェイ）サーバを用意し、すべての処理をサーバマシンで実行し、入出力結果のみ携帯端末に送り、携帯端末上では I/O のためのプログラムのみを走らせるという実装も考えられるが、携帯端末上でアニメーションを表示する場合などでは、フレームの更新時間間隔ごとにサーバから携帯端末に画像データを転送する必要が生じ、サーバと携帯端末間の通信量が膨大になる。特に携帯端末が携帯電話の場合、通信遅延が大きくなると共に、通信コストが増大するなどの問題が生じる。

そこで、本研究では、Java 言語で書かれた（複数オブジェクト群からなる）プログラムと、各オブジェクトのサイズや実行時の CPU 負荷、オブジェクトの呼び出しに必要なデータ量や各オブジェクトの入出力処理に必要な通信量や通信時間、それぞれの端末にロード可能なプログラム容量とオブジェクトの呼び出し関係、サーバと携帯端末間の通信コスト、などをもとに、与えられた Java プログラムと同じ動作をする複数サーバマシンと携帯端末のプログラム群を自動導出するプログラム変換法を考案した。変換に際しては、通信コストの最小化や、遅延時間の最小化などを目的関数にして、変換目的に応じたプログラムを導出できるよう実装を行った。

自動導出を行うためには、既存の Java アプリケーションに対して、分割の際の基準となる通信量や通信回数などの統計情報が必要となる。Java による遠隔メソッド呼び出しに関しては、性能解析と最適化に関する研究[4]や、実時間システムの性能向上ための性能計測やスケジューリングに関する研究[5,6]、また、RMI をより効率的に実装するための研究[7]など様々な研究がなされている。しかし、これらの手法は分散アプリケーションと

して記述されたプログラム群の実環境における性能計測に基づいているため、アプリケーションの分割のための性能評価を目的とする本研究では用いることが出来ない。

またこのような統計的な評価としては、プログラムのソースコードを解析して行う手法も存在するが、本研究においては手法の簡単化のため、複数回のシミュレーションに基づく性能計測を用いる。シミュレーションは対象となる Java アプリケーションのソースコードに、メソッド呼び出しの際にそれらの通信量や回数を記録する処理を機械的に挿入することで行う。修正を加えたプログラムを様々な状況を考慮して各々一定回数動作させることで、各モジュールに関する統計情報を収集する[1,2]。

次に得られた統計情報に基づき、サイズなどの制約を満たしつつ、要求された性能が最良となるような分割を求める。この分割問題は NP 困難であり、現実的な時間で最適解を求めることは困難であるため、ヒューリスティックアルゴリズムを用いて近似解を求めることとした。近似解法の代表的なものとしては、Kernighan と Lin により提案された Min-Cut 法[8]とその改良版[9]等が提案されている。またこの Min-Cut 法に基づくグラフ分割アルゴリズムと遺伝的アルゴリズム（GA 法）を組み合わせた方法[10]などが提案されている。さらに、 SA（Simulated Annealing）法[11,12]や変数に対する線形時間解法[13]なども提案されている。本研究では十分な時間をかけて計算することで比較的良い解が得られることが知られているため、SA法を採用した。

本問題を定式化し、SA を用いて実装を行った後、例題に適用して通信量等の性能を最小とするような分割を求めた。総検索を行い最適解を求めた場合と比較して、本手法は実行時間を大幅に短縮することができ、また、分割後の性能についても、幅広い探索を行うことで最適解と同程度の解を得ることができた。

2　提案手法の概要

提案手法では既存のアプリケーションをクラス単位で分割することで、端末および複数サーバのそれぞれで動作させるプログラムを導出する。各端末に配置されたモジュール群は互いに Java RMI により連携し、与えられたアプリケーションと同様に動作する。

2.1　提案手法で対象とするモデル

本手法を適用するアプリケーションは以下の条件を満たすものとする。

・アプリケーションを構成するクラスのうち、分割対象となる全てのクラスのソースコードが利用可能である
・クラス間の連携は共有変数などを介さず全てメソッド呼び出しにより行われる
・各クラスはメソッド呼び出し終了後にその引数として渡されたデータ構造に対する参照を保持しない
・メソッドの引数、返り値としては単純なデータ構造のみが用いられ、全て値渡しで行われると仮定できる

まず、ソースコードに性能計測コードを追加してシミュレーションを行うため、プログラムのソースコードが利用可能である必要がある。ソースコードが得られないクラス群は分割の対象とせず、サーバ、端末の両側に配置する（標準 API に関しては明示的に割り当てることなくサーバ、端末双方で常に利用可能であるためソースコードは必要ではない）。次に、対象とするアプリケーションは全てのクラス間の連携がメソッド呼び出しに基づいて行われる必要がある。共有変数などが利用されている場合には、それらをアクセスメソッドにより置き換えることで対応する。同様に、メソッド呼び出し後に、その引数として与えられたオブジェクトに対する参照を保持し利用することも分割後の動作に支障をきたすため認めない。ただし、そのような関係がクラス間に存在する場合でも、それらの複雑な協調動作を行うクラス群を内包するラッパークラスを新たに定義するなどして統合することで提案手法を適用することはできる。また、ここではシステムの単純化のため全てのメソッド呼び出しは値渡しで行われるものとし、RMI で送受信不可能なソケットなどの通信端点といった環境に接続する特殊なオブジェクトは引数、返り値として用いられないものとする。

2.2　提案手法が想定する実行環境

分割されたアプリケーションは Java RMI を利用して端末、サーバ間の連携を実現するため、実行環境においてサーバ、端末間で双方向にRMIが利用できるものと仮定する。

現在一般に利用可能な携帯電話などの環境においては、RMI は利用できず、通信も携帯端末側からの Pull 型のデータ通信に限られている。しかし、次世代携帯端末の標準規格である WAP2.0 においては、サーバ側からの Push 型の通信も規定されているため、そういったサービスを利用して擬似的に RMI を実装することで提案

手法が利用可能である。また、現在でも、au の携帯電話で提供されている ezplus 環境では、C-mail と呼ばれるショートメールサービスを利用することで小さなデータサイズに限られるがサーバから端末に対する Push 型の通信が行える。それをトリガとして利用することでサーバ側から携帯端末側への RMI のエミュレーションも擬似的に実装可能である。

3　分割の最適化のための統計情報見積もり手法

提案手法では、分割の基準となるデータを求めるため、与えられたアプリケーションに必要となる統計情報を求めるためのコードを挿入し、一定時間動作させることでシミュレーションを行う。

3.1　最適化項目と統計量

まず、本研究において分割時に最適化する項目を以下のように定める。

・通信量
・遅延時間
・携帯端末における消費電力

ここで、それぞれの項目に関わる統計量を以下のようにして求めるものとした。通信量に関しては、分割した場合に生じる端末、サーバ間の通信量により評価を行う。遅延時間に関しては、データ送受信のオーバーヘッドにより遠隔メソッド呼び出しを行う回数に強く依存すると考えられるため、端末、サーバ間の遠隔メソッド呼び出し回数により評価する。消費電力は端末において実行されるコード量に依存するため、ここでは、携帯端末で消費される CPU 時間に依存すると仮定する。また、一般に携帯端末はサーバに比べてきわめて非力であるため、CPU 時間を大量に消費するモジュールをサーバに置くことで、アプリケーション全体の高速化も図れる。提案手法においてはこれらの 3 項目の評価値に対して重み付け係数を設けその値を調整することで、より目的に添った最適化が行えるものとした。

以下、各統計量の見積もり方法について述べる。

3.1.1　通信量の見積もり

ある 2 つのオブジェクトが別々のマシンに置かれている場合、それらの間でのメソッド呼び出は遠隔メソッド呼び出しによって実行されるものとする。本研究では、このような遠隔メソッド呼び出しによって生じる通信量を 1 回のアプリケーションの実行について積算した値をそれぞれ通信量として用いる。本研究で対象としているのは一般的な単一のマシン上で動作する Java アプリケーションであるため、実際に遠隔メソッド呼び出しが行われるわけではなく、そのようにして実行されると仮定してその場合の統計量を評価する。

3.1.2　遅延時間の見積もり

ここでは遅延時間は主に通信オーバーヘッドにより発生すると仮定し、遠隔メソッド呼び出しの回数を用いて評価する。実際には通信量や回線速度に応じて変化するため、実環境に応じて通信量の計数との重み付けを調整する必要がある。

3.1.3　CPU 時間の見積もり

CPU 時間は各クラスに関してそのクラスに属するメソッドの総実行時間を計測することで見積もりを行う。各メソッド呼び出し時にその時刻を記録し、呼び出しが終了した（もしくは、そのメソッド内から標準 API 以外の他のクラスのメソッドが呼び出された場合にはその呼び出しが発生した）時刻との差分を累計することでそのメソッドの属するクラスの CPU 時間を見積もる単純な手法を用いた。ただし、この手法では実装の簡単化のため、実験環境において他に負荷が無く、また、対象アプリケーションもマルチスレッド化されていないと仮定している。このような仮定が成り立たない場合には、Extensible Java Profiler[14]のようなプロファイリング用のツールを用いた詳細な解析が必要となる。

3.2　統計情報収集コードの追加

分割の際に基準とする統計情報を収集するため、評価の対象となる Java アプリケーションのソースコードに

対して以下のような変更を加える。

1．まず、統計情報を記録し、アプリケーションの終了時に結果を出力するためのデータ統計用クラスを定義する。このクラスは次のような要素を持つ。
   ○統計量を保存しておくためのデータ領域
   ○統計量を更新するためのメソッド
   ○結果をファイルなどに出力するためのメソッド
2．統計情報を更新するメソッドの呼び出し文を挿入する。
3．プログラムの終了直前の部分（例えば main メソッドの最後）に結果を出力するメソッドの呼び出し文を挿入する。

4　モジュール配置問題の定式化

本研究で対象とするモジュール配置問題を以下のように定式化する。

入力：

・モジュールの集合 $M=\{m_1,\ldots,m_k\}$
・各モジュールの使用メモリサイズ $S:M\rightarrow N$
・端末の集合 $P=\{p_1,\ldots,p_l\}$
・各端末のメモリ容量 $m:P\rightarrow N$
・分割において配置が固定されるモジュールの集合 $M_{S1},\ldots,M_{Sl}\subset M$
・モジュール間の通信量 $T:M\times M\rightarrow N$
・モジュール間の呼び出し回数 $C:M\times M\rightarrow N$
・各モジュールの実行時間 $Q:M\rightarrow N$
・各端末の処理能力計数 $H_1,\ldots,H_l\in N$
・目的関数の係数 $K_1,K_2,H_1,\ldots,H_l\in N$

出力：

・以下の制約条件を満たし、目的関数値を最小にするような M の分割 $D:M\rightarrow \mathrm{P}$

制約条件：

・$^{\forall}P\in\mathbf{P}:\ \sum_{D(m)\in P}S(m)\leq m(P)$

・$^{\forall}i:m\in M_{si}\rightarrow D(m)=P_i$

目的関数：

・$f(D)=K_1\sum_{D(m_1)\neq D(m_2)}T(m_1,m_2)+K_2\sum_{D(m_1)\neq D(m_2)}C(m_1,m_2)+\sum_{1\leq i\leq l}H_i\sum_{D(m)\in P_i}Q(m)\ \mathrm{m}\in Mc$

ここでは、簡単化のため $T(m_1,m_2)$ は m_1 から m_2 への通信量とその逆向きの通信量の合計としている。ただし、必要であるならばそれぞれの向きを別として扱うことも出来る。C も同様に m_1 から m_2 へとその逆向きのメソッド呼び出しの回数とした。$H_1\sim H_l$ は各端末の処理能力を表す係数とした。目的関数においては、各端末に割り当てられたモジュールによる CPU 使用時間を H_i で重みを付けて積算している。これにより処理能力の低い端末ほど大きな値になるよう設定することで、その端末に割り当てられるモジュールによる CPU 時間消費を減らす効果が得られる。携帯端末における消費電力を最小にしたい場合にも該当する H_i の値を他より大きな値に設定することで目的を達成できる。K_1,K_2 は、それぞれ通信量の係数、呼び出し回数の係数、であり、通信量を最小にしたい場合には K_1、通信遅延を最小にしたい場合には K_2 をより大きな値に設定すればよい。

ここで、このモジュール割り当て問題の複雑さについて論じる。頂点集合 $V=v_1,\ldots,v_{2n}$、辺集合 $E\subseteq V\times V$ により定義される無向グラフ $G=(V,E)$ を、切断コスト

$$W(V_1,V_2)=$$

が最小となるよう要素数 n の 2 つの集合 $V_1,V_2(|V_1|=|V_2|=n,V_1\cup V_2=V,V_1\cap V_2=\phi)$ に分割するグラフ分割問題は NP 困難問題であることが知られている[15]。以下、このグラフ分割問題から本研究で対象とするモジュール割り当て問題へと多項式時間で帰着可能であることを示すことで、本モジュール割り当て問題が NP 困難問題

である事を証明する。

略証：任意のグラフ分割問題のインスタンス $G=(V,E)$ を、

・$M=V\cap\{v_c\}$

・$^{\forall}v\in M: S(v)=1$

・$\mathbf{P}=\{P_1,P_2\}$

・$S(P_1)=|V|/2+1$

・$S(P_2)=|V|+1$

・$M_{S1}=\{v_c\}$

・$T(v_1,v_2)=\begin{cases}1 & (v_1,v_2)\in E\\ |E|+1 & v_1=v_c \text{ or } v_2=v_c)\in E\\ 0 & \text{othewize}\end{cases}$

・$H_1,\ldots,H_l=0, K_1=1, K_2=0$, 他のパラメータは 0 もしくは ϕ

として本問題に帰着する。元のグラフ分割問題における分割 $\{V_1,V_2\}$ のコスト $W(V_1,V_2)$ に対し、この分割問題から帰着したモジュール配置問題の分割 $D(^{\forall}m\in V_1\cap\{v_c\}: D(m)=P_1, ^{\forall}m\in V_2: D(m)=P_2)$ による目的関数値は、分割 D において P_i に配置されるモジュール数を $\#P_i(D)=|\{m: D(m)=P_i\}|$ と示すとすると、定義より $F(D)=\#P_2(D)\times(|E|+1)+W(V_1,V_2)$ となる。ここで、任意の G の分割に対してコスト W は、$W(V_1,V_2)\le|E|$ であるため、任意の分割 D' と D'' に対して、$\#P_1(D')<\#P_1(D'')\rightarrow F(D')<F(D'')$ が成立する。よって、帰着したモジュール配置問題の最適解 D は、$\#P_1(D)$ の値が最小であるような分割となり、P_1 に配置されるモジュール数は高々 $m_c=|V|/2+1$ 個であるため、そのモジュール数は制約条件より $\#P_2(D)=|M|-S(P_1)=|V|/2$ となる。その際の目的関数値は $F(D)=(|V|/2)\times(|E|+1)+W(V_1,V_2)$ であるため、$W(V_1,V_2)$ が最小となる分割が最適解となる。よって、この最適解から G の最適な分割が得られる。これらの帰着は多項式時間で可能であるため、本モジュール配置問題は NP 困難問題である。 □

5 SA を用いた分割法

本問題は NP 困難問題であり実用的な時間で最適解を求めることが困難と考えられるため、ヒューリスティックアルゴリズムを用いて近似解を求める。ここでは SA（Simulated Annealing）法による近似解法を用いた。

SA 法では、解候補を繰り返し改善することで最適解へと近づけることを目指す。現在の解候補の近傍を探索し、ランダムに新たな解候補を選択する。新たな解候補が元の解候補より改善されていれば解候補を置き換える。解が改善されない場合でもある確率により置き換えを行う。最初はこの確率を大きくしておき、少しずつ減らしていく。確率の減らし方を緩やかにし、十分な回数の試行を行うことで比較的良い解が得られることが知られている。

5.1 提案手法における SA アルゴリズム

提案手法は以下のアルゴリズムにより実装した。

1．初期温度 T_0 の設定と初期解の生成

2．以下のことを一定回数繰り返す（これをループ回数と呼ぶ）

① 現在の割り当てから割り当て候補を生成

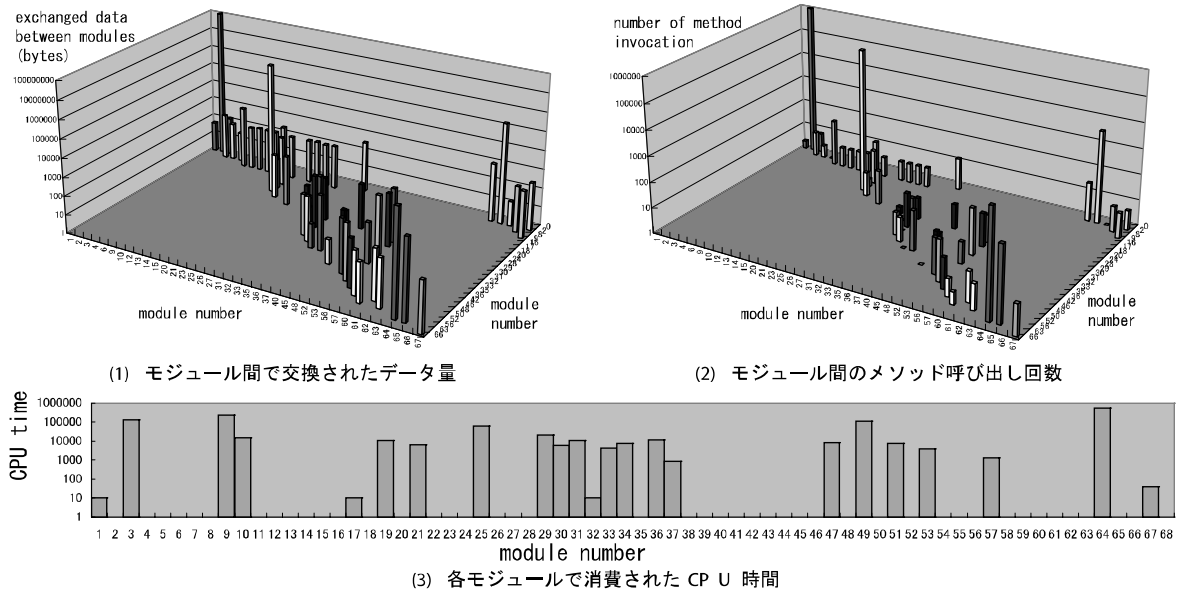
② 現在の割り当ての評価値 $F(D)$ と割り当て候補の評価値 $F(D')$ の計算を行い、その差 $\Delta C=F(D')-F(D)$ を計算する。

③ 計算した評価値が現時点で最小ならば、最小評価値をその値に更新する。また、そのときのモジュールの割り当ても記憶しておく。

④ $\Delta C<0$ なら、割り当て候補を現在の割り当てとする。そうでない場合も確率 $e^{(-\Delta C/T)}$ で割り当て候補を現在の割り当てとする。

3．以下のように温度を一定の割合で下げ、2．へ。

$T_{k+1}=kT_k$（k: 定数（冷却係数））



(1) モジュール間で交換されたデータ量　(2) モジュール間のメソッド呼び出し回数

(3) 各モジュールで消費された CP U 時間

図１　モジュール間のインタラクションおよび各モジュールで消費される CPU 時間

４．温度が最終温度 T_{fin} 未満になれば、処理を終了し、評価値が最小のときのモジュールの割り当てを出力する。そうでなければ２．へ。

またここで、解候補の生成は以下の方針で行った。

・モジュールを１つ選択。

・そのモジュールをいずれか別の端末に移動。

・移動先端末のメモリサイズを超過した場合には、移動先端末から任意の１モジュールを選択、交換し、メモリ制限を満たす場合は解候補とする。交換後もサイズ超過した場合には候補を破棄する。

６　本手法の例題への適用

既存 Java アプリケーションに対して我々のシステムを適用した。対象アプリケーションは既存の画像編集ツールで、68個のクラスからなり、それらの平均サイズは 5KB であった。まず、このアプリケーションに対して統計情報収集手法を適用した結果を**図１**に示す。それぞれのグラフは、各クラス間の⑴通信量と⑵呼び出し回数（全く呼び出しの無かった組み合わせはグラフから省略）、⑶それぞれのクラスの実行にかかった CPU 時間、を示している。

次に、モジュール割り当て手法を適用した。ここでは、予備実験により SA 法のパラメータを下記のように定めた。

・初期温度：$T_0 = 400.0$

・最終温度：$T_{fin} = 1.0$

・冷却係数：$k = 0.95$

手法の有用性を評価するため、$l = 2$ として上記アプリケーションを二分割する実験を行った。いくつかの目標関数のパラメータ H_1, H_2, K_1, K_2 に対して、携帯端末のメモリ制限を変化させ実験を行った。結果を**表１**に示す。１列目は通信量の係数 K_1 のみを用い、他の係数 H_1, H_2, K_2 を0とした実験結果である。この場合、通信量はクライアントのメモリ容量制限に応じて劇的に変化している。２列目は同様に通信回数の最適化を行った結果である。この場合、サーバ、端末間の通信回数が削減されていることが分かる。３列目は端末の一方を携帯端末と見なし、そちら側での CPU 時間消費を最小化した結果である。この場合、携帯端末側で使用される CPU 時間が利用可能なメモリの増加と共に減少している。

以上の結果より、我々の手法は現実に携帯端末で動作させるアプリケーションの設計に対しても有効であると言える。ただし、いくつかの場合で SA 法が局所最適解に陥っている。局所最適解を回避するため、SA 法に基づくアルゴリズムのさらなる改善が必要である。

表１　例題アプリケーションの分割結果

memory of client (KB)	amount of comm. (bytes)	number of method invocation	CPU time of client (ms)
	$H_1=0$, $H_2=0$, $K_1=1$, $K_2=0$		
45	234014	4162	134199
50	234702	4174	134209
55	22382	1018	704093
57	4784	134	694691
60	2242	73	709080
	$H_1=0$, $H_2=0$, $K_1=0$, $K_2=1$		
45	234014	4162	134199
50	234314	4173	134209
55	234314	4173	134209
60	3462	108	691717
	$H_1=1$, $H_2=0$, $K_1=0$, $K_2=0$		
45	238198	4297	363280
50	236322	4236	378151
55	236090	4230	378151
60	235294	4218	378151

７　まとめ

本研究では、クライアントサーバ間の通信量、実行時間、携帯端末の消費電力などができるだけ小さくなるような、Java アプリケーションの分割を行うための統計情報評価法の提案を行った。また提案方法で求めた統計情報を用いて、通信量や実行時間、消費電力などができるだけ小さくなるように各クラス（モジュール）をクライアント側とサーバ側に配置する方法の提案を行った。また、例題に適用して総検索法と比較することにより提案手法の有効性を評価した。

今後の課題としては、様々な携帯アプリケーションに本手法を適用することである。GPS を用いた位置情報に基づく歩行者ナビゲーションシステムなどへの応用を検討している[3]。また、モジュールの配置方法において、できるだけ局所解に陥らないような効率のよい初期配置の取り方やモジュールの交換手法を検討していく予定である。

研究成果

[1] 浦田繁玄, 田中邦明, 梅津高朗, 中田明夫, 東野輝夫 : シミュレーションと SA を用いた Java プログラムの分割と携帯端末上での実装の一手法, ソフトウェア工学の基礎 X (日本ソフトウェア科学会 FOSE 2003), 近代科学社, vol. 29, pp.239-250 (2003)

[2] Takaaki Umedu, Shigeharu Urata, Akio Nakata and Teruo Higashino : “Automatic Decomposition of Java Program based on Simulation and its Implementation on Mobile Terminals”, *Proc. of the 19th Int. Conf. on Advanced Information Networking and Applications (AINA2005)*, pp.544-549 (March 2005)

[3] Yoshitaka Nakamura, Guiquan Ren, Masatoshi Nakamura, Takaaki Umedu, Teruo Higashino : “Personally Customizable Group Navigation System using Cellular Phones and Wireless Ad-hoc Communication”, *Proc. of the 2005 IEEE Int. Conf. on Multimedia & Expo (ICME2005)*, CD-ROM (July 2005)

参考文献

[4] Matjaz B.J., Ivan R., Marjan H., Alan P.S. and Simon N.: Java 2 Distributed Object Models Performance Analysis, Comparison and Optimization, *Proc. of the 7th Int. Conf. on Parallel and Distributed Systems (ICPADS'00)*, pp.239-246 (2000)

[5] Kalogeraki V., Melliar-Smith P.M. and Moser L.E.: Using Multiple Feedback Loops for Object Profiling, Scheduling and Migration in Soft Real-Time Distributed Object Systems, *Proc. of the 2nd IEEE Int. Symp.*

on Object-Oriented Real-Time Distributed Computing, pp.291-300 (1999)

[6] Flores A.P., Nacul A., Silva L., Netto J., Pereira C.E. and Bacellar L.: Quantitative Evaluation of Distributed Object-Oriented Programming Environments for Real-Time Applications, *Proc. of the 2nd IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing*, pp.133-138 (1999)

[7] Kono K. and Masuda T.: Efficient RMI, Dynamic Specialization of Object Serialization, *Proc. of the 20th Int. Conf. on Distributed Computing Systems (ICDCS 2000)*, pp.308-315 (2000)

[8] Kernighan B.W. and Lin S.: An Efficient Heuristic Procedure for Partitioning Graphs, *Bell Syst. Tech. J.*, vol.49, no.2, pp.291-307 (1970)

[9] Krishnamurthy B.: An Improved Min-Cut Algorithm for Partitioning VLSI Networks, *IEEE Trans. Computers*, vol.33, no.5, pp.438-446 (1984)

[10] Bui T.N. and Moon B.R.: Generic Algorithm and Graph Partitioning, *IEEE Trans. Computers*, vol.45, no.7, pp.841-855 (1996)

[11] Sait S.M. and Youssef H.: 組合せ最適化アルゴリズムの最新手法 基礎から工学応用まで, 白石洋一訳、丸善㈱出版事業部（2002）

[12] Johnson D.S., Aragin C., McGeoch L., and Schevon C.: Optimization by Simulated Annealing, An Experimental Evaluation, Part1, Graph Partitioning, *Operations Research*, vol.37, pp.865-892 (1987)

[13] Fiduccia C.M. and Mattheyses R.M.: A Linear-Time Heuristic for Improving Network Partitions, *Proc. of the 19th Design Automation Conference*, pp.175-181 (1982)

[14] Sebastien Vauclair : Extensible Java Profiler,//http://ejp.sourceforge.net/

[15] Garey M.R. and Johnson D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman (1979)

〈発 表 資 料〉

題　　名	掲 載 誌 ・ 学 会 名 等	発 表 年 月
シミュレーションと SA を用いた Java プログラムの分割と携帯端末上での実装の一手法	ソフトウェア工学の基礎X（日本ソフトウェア科学会 FOSE 2003），近代科学社，vo129, pp.239-250	2003年11月
Automatic Decomposition of Java Program based on Simulation and its Implementation on Mobile Terminals	Proc. of the 19th Int. Conf. on Advanced Information Networking and Applications (AINA 2005). pp. 544-549	2005年 3 月
Personally Customizable Group Navigation System using Cellular Phones and Wireless Ad-hoc Communication	Proc. of the 2005 IEEE Int. Conf. on Multimedia & Expo (ICME2005), CD-ROM	2005年 7 月