2004－HPC－99　(8)
2004／7／30

Webサービスを用いた最適化計算システムの構築

廣安 知之 †，三木 光範 †，狩野 浩一 ††，下坂 久司 ††

† 同志社大学工学部　†† 同志社大学大学院

近年，IT 資源をネットワークで結合し統合利用する Grid が注目を集めている．膨大な計算資源を構築し使用するという利用方法も考えられるが，サービスとサービスを柔軟に効率よく結合し，新しいシステムを構築するための利用が今後増加していくものと考えられる．最適化計算は問題に応じて最適器と解析器を選択する必要がある．そのため，最適化計算のシステムを構築する際に，Grid 技術を利用して構築することが有効であると考えられる．本研究では，その前段階として，web サービスを利用して最適化計算システムを構築する際の指針を提示する．その指針に沿って最適化計算システムのプロトタイプを構築した．構築システムでは，ユーザーがさまざまな最適器と解析器およびその他のツールを選択し，それらを連携させるだけで最適化計算システムを構築できる．また，構築したシステムが稼動することを確認し，数値実験によりシステムの基本性能を測定した．

Development of Optimization Problem Solving System using Web Service

Tomoyuki HIROYASU †　Mitsunori MIKI †　Koichi KANO ††　Hisashi SHIMOSAKA ††

† Knowledge Engineering Dept., Doshisha University

†† Graduate School of Engineering, Doshisha University

Recently, the Grid that connects IT services by network and creates a unified system has been paid attention. One of the Grid utilities is developing a huge computational resource. The other is to construct a new system effectively and quickly using existing services and Grid technology. The number of this utilization is getting bigger. To perform optimization calculation by computational simulation, optimizer and analyzer should be selected for an optimization problem. Therefore, when an optimization calculation system is developed for an optimization problem, using the Grid is effective and useful. In this paper, as a preliminary research, we propose some directions of developing methods of optimization calculation system using web services. In the proposed system, users select their own optimizer, analyzer, and tools from the list of them. Then, they connect selected modules easily and construct the problem oriented system effectively. The developed system is tested on the cluster and confirmed the system operation. At the same time, the overhead and calculation time is examined.

1　はじめに

近年ネットワーク技術の進歩により，分散した計算資源や情報資源を統合する Grid[1] が高く注目されている．Grid に関する研究ではアプリケーション作成のための Grid ミドルウェアが多数開発されており，代表的な Grid ミドルウェアに Globus Toolkit[2] が挙げられる．最近ではビジネス分野での利用から，Web サービス [3] の標準技術を拡張した Grid サービスを定義するために OGSA(Open Grid Services Architecture)[4] が用いられている．今後，OGSA の実装が下位層に位置する OGSI(Open Grid Services Infrastructure) に代わり，新たに WSRF(WS-Resource Framework)[5] が提案されたことで，Grid と Web サービスの標準技術が同一のものとなる．

本研究では，Grid 環境における最適化計算システムの構築を目標とし，Web サービスを用いた最適化計算システムの構築について述べる．最適化計算システムには，対象問題の解析を行う解析アプリケーション，最適化処理を実行する最適化アプリケーション，対象問題の目的関数値を保存するデータベースアプリケーションなど様々な構成要素が考えられる．そのため，特定のアプリケーションに依存しない汎用的なアプリケーション連

携の仕組みが必要となってくる．そこで提案システムでは，ユーザがアプリケーションを複数選択し，選択したアプリケーションの連携方法を記述するのみで，最適化計算システムを構築できる簡便な仕組みを提供する．また，特に最適化アプリケーションに焦点をあて，提案システム上のアプリケーション連携を利用する，新たなアプリケーションを構築するための仕組みも提供する．さらに，実際に提案システムを用いて，最適化計算システムを構築し，提案システムの有効性を検討する．

2 最適化計算システム

2.1 システム概要

最適化計算とは，ある制約条件の下で目的とする関数の値を最小化あるいは最大化する変数を決定する計算のことである．

本論文では，最適化計算システムとして，次探索点の決定を行う Optimizing Service と，解析計算を行い目的関数や制約条件の値を決定する Analyzing Service の 2 つから構成されるシステムを想定する．この最適化計算システムでは，Optimizing Service において解析処理が必要になった際，Analyzing Service を呼び出すことにより最適化計算を実行する．最も単純な最適化システムには，Optimizing Service および Analyzing Service にそれぞれ最適化及び解析アプリケーションを 1 つずつ割り当てたシステムである．しかし，現実の最適化システムでは，最適化計算の種類によって Fig. 1 に示すように，解析計算に複数のアプリケーションを連携させて利用することが有効であったり，解析処理に何らかの最適化計算結果が必要となる場合が数多く存在する．そのため，本論文で提案する最適化計算システムでは，次のような機能が必要となる．

- Analyzing Service としてエンドユーザが任意のアプリケーション連携を構成できる機能
- 構成したアプリケーション連携を Optimizing Service に指示し，解析処理として利用できる機能

解析処理に最適化計算結果を利用することを考慮すると，提案システムにおいてアプリケーションの種類を区別しないことが望ましいと考えられる．また一般的に，提案システムで利用される各アプリケーションの開発者が異なることを考慮すると，

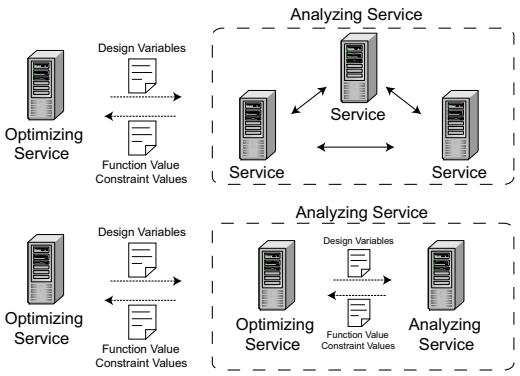

Fig. 1 最適化計算システム

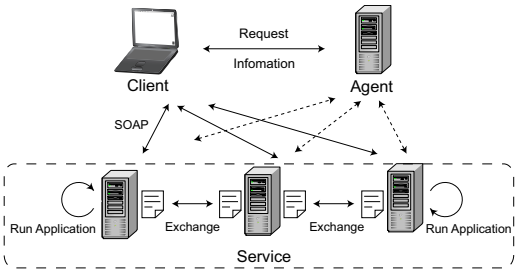

Fig. 2 提案システム

これらのアプリケーション間の情報交換には，ファイルを介して行うことが現実的である．

このことから，本研究で提案する最適化計算システムでは，アプリケーションを連携させ最適化計算システムを構築する Client，アプリケーションの実行やアプリケーション間で入出力ファイルを交換するための機能を提供する Service 群，各 Service の情報を一元管理する Agent の 3 つの主要コンポーネントから構成される．提案システムの概要を Fig. 2 に示す．現時点では，Agent の実装を行っておらず，Client は何らかの方法で利用する Service の情報を取得しているものとする．以下の節では，上記の機能を満たし，提案システムを構築する仕組みについて述べる．

2.2 基本機能

提案システムにおいて各 Service は，アプリケーションの実行やアプリケーション間で入出力ファイルを交換のために，4 つの基本機能を提供する．これらの基本機能は Web サービスの標準技術である SOAP-RPC[6] で実装されている．以下に 4 つの基本機能の詳細を示す．

- Receive Files
  SOAP-RPC の呼び出し側からのファイルを受信し，アプリケーションの入力ファイルとする．

- Run Application
  各 Service が管理するアプリケーションを実行する．
- Return Files
  SOAP-RPC の呼び出し側にアプリケーションの実行によって得られた出力ファイルを返信する．
- Send Files
  SOAP-RPC の呼び出し側が指定した他の Service に，アプリケーションの実行によって得られた出力ファイルを送信する．

Receive Files 機能，Run Application 機能，Return Files 機能はそれぞれ 1 回の SOAP-RPC によって実装される．また，Send Files 機能は，2 回の SOAP-RPC を用いて実装している．つまり 1 回目の SOAP-RPC でファイルの送信先を指定し，2 回目の SOAP-RPC で送信先の Receive Files 機能を利用してファイルを送信する．Client はこれらの基本機能を組み合わせ，適切な順序で利用することによりアプリケーションの連携を行う．

2.3 設定ファイル

提案システムにおいて Client は，特定の書式に従って設定ファイルを記述することにより，アプリケーションの連携を指示する．設定ファイルに記述する内容を以下に示す．

- 実行する基本機能の総数 (NUMBER_OF_FUNCTION)
- 実行する基本機能の種類 (FUNCTION)
- 基本機能の呼び出し側の Service(FROM)
- 基本機能の呼び出し先の Service(TO)
- 送受信を行うファイルの数 (NUMBER_OF_FILES)
- 呼び出し側の Service が送受信するファイル名 (FROM_FILES)
- 呼び出し先の Service が送受信するファイル名 (TO_FILES)

2.4 Application Programming Interface

提案システムではアプリケーション開発者が容易にアプリケーション連携を利用し，新たなアプリケーションを開発するための API を提供する．Optimizing Service に属するアプリケーションでは，次に示す API を用い，Client が指定する任意のアプリケーション連携を解析処理として利用する．

2.4.1 Coordinate クラス

Coordinate クラスは，連携を行うアプリケーションの存在する Service の様々な情報を保持する変数やアプリケーションの連携を行うための各種メソッドを提供する．以下に示す API を利用するためには，Coordinate クラスを宣言し，インスタンスを生成する必要がある．

2.4.2 初期化

Coordinate クラスには連携を行う Service の名前や URL といった情報を保持する変数群が存在し，この変数値を用いて SOAP-RPC による通信を実現する．そのため，まず変数群に Service の情報を渡し初期化する必要がある．初期化には以下の API を用いる．

```
int initialize(String service_info);
```

2.4.3 設定ファイルの指定

Optimizing Service では設定ファイルを指定する API を用いて，Client から送信されたアプリケーション連携の設定ファイルを利用する．

```
int configure(String configfile);
```

2.4.4 アプリケーション連携の実行

Optimizing Service において解析処理が必要になるたびに，以下の API を呼び出し，アプリケーション連携を実行する．

```
int run();
```

2.5 最適化計算システムの構築

上で述べた様々な仕組みを用いて，最適化計算システムの構築を行う．エンドユーザが提案システムを用いて最適化計算システムを構築するためには，次に示す手順で行う．

1. 現在利用可能な Service の情報を Agent から取得し，利用する Service を選択する．
2. 基本機能を組み合わせ，各 Service をどのように連携させるかを決定する．
3. アプリケーション連携の実行に必要なアプリケーションの入力ファイルを用意する．
4. アプリケーション連携を実行し，実行結果を取得する．

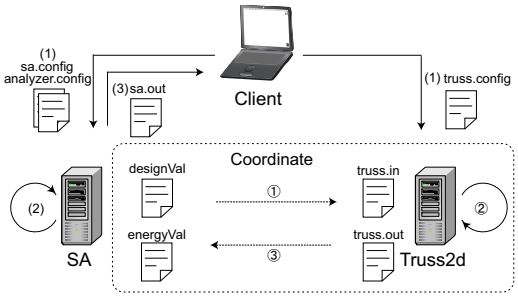

Fig. 3　SA と Truss2d の連携

2.5.1　アプリケーションの選択

本節では Fig. 3 に示す最適化システムの構築例を前節の手順に従って述べる．ここでは，トラス構造を SA を利用して最適化することを想定する．

まず，Client は Agent から得た利用可能な Service の情報をもとに，連携を行う Optimizing Service と Analyzing Service を選択する．

Analyzing Service

Analyzing Service に，トラス構造物立体構造解析アプリケーション (Truss2d) を選択する．Truss2d は入力ファイルとして truss.confing と truss.in を必要とし，出力ファイルとして truss.out を出力する．truss.config には解析の初期設定が記述されており，解析を行うトラス構造物の基本構造が定義される．また，truss.in には解析を行うトラス構造物の各部材の断面積の値が記述されており，これが設計変数値に対応する．truss.out には解析によって得られた目的関数値と制約条件値が記述される．

Optimizing Service

Optimizing Service に，Simulated Anealing(SA)[7] を選択する．SA は入力ファイルとして sa.config を必要とし，出力ファイルとして sa.out を出力する．sa.config には最適化の初期パラメータが記述され，sa.out には最適化の結果が記述される．また，SA は解析処理を行う際に設計変数をファイル designVal に出力し，目的関数値を energyVal から入力する．この際，前述の API に設定ファイル (analyzer.config) を指定することで，任意のアプリケーション連携を可能とする．

Fig. 3 の例では，解析アプリケーション Truss2d との連携を行う．ここで Truss2d と SA の連携を考える際，designVal は Truss2d の入力ファイル truss.in と，energyVal は Truss2d の出力ファイルである truss.out と同じ書式で記述されている必要がある．提案システムにおいて，入出力ファイルの書式を統一する機能は必要不可欠であるが，現時点ではそのような機能を提供しておらず，今後の課題とする．

```
$NUMBER_OF_FUNCTION   4
$FUNCTION    1 1 2 3
$FROM          Client Client Client Client
$TO            SA Truss2d SA SA
$NUMBER OF FILES    2 1 0 1
$FROM_FILE    saparameter.txt analyzer.txt
$TO_FILE       sa.config analyzer.config
$FROM_FILE    trussparameter.txt
$TO_FILE       truss.config
$FROM_FILE
$TO_FILE
$FROM_FILE    saresult.txt
$TO_FILE       sa.out
```
Fig. 4　設定ファイル (Client-SA)

```
$NUMBER_OF_FUNCTION   3
$FUNCTION    1 2 3
$FROM          SA SA SA
$TO            Truss2d Truss2d Truss2d
$NUMBER OF FILES    1 0 1
$FROM_FILE    designVal
$TO_FILE       truss.in
$FROM_FILE
$TO_FILE
$FROM_FILE    energyVal
$TO_FILE       truss.out
```
Fig. 5　設定ファイル (SA-Truss2d)

2.5.2　設定ファイルの記述

次に選択したアプリケーションの連携方法を，2.3 節で述べた形式で設定ファイルに記述する．Fig. 3 に示す例では，Client が最適化計算システムを実行させるための設定ファイルと SA が Truss2d との連携を行うための設定ファイル (analyzer.config) を記述する必要がある．Fig. 4 に最適化システムを実行するための設定ファイルを，Fig. 5 に SA が分析器として Truss2d を用いるための設定ファイルを示す．

2.5.3　必要なファイルの用意

アプリケーション連携を行うために必要な入力ファイルを作成する．Client は，Truss2d の入力ファイルの 1 つである truss.config と SA の入力ファイルである sa.config を作成しなければならない．Truss2d の入力ファイルの 1 つである truss.in は SA の出力ファイルである sa.out を Truss2d の入力ファイルとして用いるため，Client が作成する必要はない．

2.5.4　アプリケーション連携の実行

2.4 節に述べた API を用いて Fig. 6 のような実行プログラムを作成し，アプリケーション連携を実行する．

```
public class Client(){
  public static void main(String args[]){
    Coordinate coordinate;

    public Client(){
      coordinate = new Coordinate();
    }

    coordinate.initialize(args[0]);
    coordinate.configure(args[1]);
    coordinate.run();
  }
}
```

Fig. 6　Client プログラム

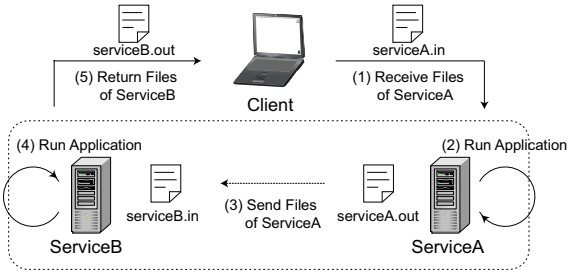

Fig. 7　アプリケーションの連携

3　数値実験

3.1　基本機能の実行時間

本実験では，提案システムにおけるアプリケーション連携の基本性能の評価を行う．基本性能の評価は，提案システムが提供する各 API および Service の各基本機能の実行に要するオーバヘッドを測定する．

3.1.1　実験概要

本実験で用いるアプリケーション連携を Fig. 7 に示す．ServiceA 及び B は，入力ファイルに記述されている内容をコピーして，ファイルに出力するサービスである．本実験では基本機能の実行時間の測定を目的とするため，入出力ファイルのサイズは最小とする．

本実験において，Client は Fig. 6 に示す Client プログラムを実行し，アプリケーションの連携を行う．ServiceA に対して 3 回，ServiceB に対して 2 回の計 5 回の SOAP-RPC を行う．また，ServiceA は ServiceB に対して 1 回の SOAP-RPC を行う．本実験では，提案システムの性能を正確に測定するために，Table 1 に示すクラスタ環境を用い，Client，ServiceA，ServiceB にそれぞれ 1 ノードずつ割り当てた．

Table 1　実験環境

CPU	Pentium4 2.4GHz
Memory	DDR-SDRAM 1GB
Network	Ethernet 100Mbps
OS	Debian GNU/Linux 2.4.18

Table 2　API の実行時間

API	Execute Time (ms)
initialize	19.2
configure	15.8
run	144.3

3.2　実験結果

前節で述べた環境下で，試行回数 100 回における各 API の平均実行時間を Table 2 に，各基本機能の平均実行時間を Table 3 に示す．

Table 2 より，初期化と設定ファイルの指定に 36ms，Client が基本機能を 5 回実行するのに 144ms 必要なことが分かる．また Table 3 より，アプリケーションの実行よりもファイルを送受信する基本機能のほうが多くの時間を必要とすることが分かる．これはファイルを送受信する際に SOAP with Attachments という SOAP の添付ファイル形式への変換を行うためである．また Send Files 機能の実行に多くの時間がかかるのは，1 回目の SOAP-RPC で送信先の Service を指定し，2 回目の SOAP-RPC で送信先の Receive Files 機能を呼び出すためである．

3.3　最適化計算システムの実行時間

前節の基本機能の評価に基づき，提案システムで最適化計算システムを構築した場合の性能を測定する．

3.3.1　実験概要

本実験では，Fig. 3 に示す最適化計算システムを用いて，提案システムで構築した最適化計算システムの性能を評価する．Client は SOAP-RPC により Service の基本機能を 4 回実行することによって最適化計算を実現する．また，SA は Service の基本機能を 3 回実行することによって 1 回の解析処理を実現する．前節と同様に，Fig. 6 に示す Client プログラムを実行し，API 及び Client から実行される Service の基本機能の実行に必要な時間を測定する．実験環境は Table 1 に示すクラスタ環境を用い，Client 及び提案システムが提供する SA，Truss2d にそれぞれ 1 ノードずつ割り当てた．

Table 3　基本機能の実行時間

Function	Execute Time (ms)
Receive File of ServiceA	34.7
Run Application of ServiceA	10.7
Send File of ServiceA	54.7
Run Application of ServiceB	9.6
Return File of ServiceB	32.8

Table 4　API の実行時間

API	Execute Time (ms)
initialize	2.01
configure	1.45
run	5022.72

3.3.2　実験結果

試行回数 100 における各 API の平均実行時間を Table 4 に，Client が最適化計算を実行する際に用いる SA の各基本機能の平均実行時間を Table 5 に示す．

実験結果より，SA のアプリケーションを実行する Run Application が 50194ms と 99%以上の時間を占めていることが分かる．本実験において Client が最適化計算を完了する時間のうち，SA の Run Application 機能の実行時間以外は固定的に必要な時間であり，全体に占める割合も比較的小さい．このため，Run Application 機能に要する時間を見積もることができれば，最適化計算に必要な時間も推測することができる．

本実験で用いた SA による最適化計算では，1 回の最適化計算に 320 回の解析計算を必要とする．1 回の解析計算は SA が Truss2d の Receive Files 機能，Run Application 機能，Return Files 機能を実行することにより実現される．実際に Fig. 1 に示す 1 ノードを用い，提案システムを用いずに SA と Truss2d を実行したところ，25821ms で最適化計算を終了した．これに提案システムのオーバヘッドを加えた時間が実行の見積もり時間である．Receive Files 機能と Return Files 機能に約 33ms，Run Application 機能に約 10ms，API の一連の処理に約 140ms かかることを考慮すると，SA のアプリケーションのオーバヘッドは約 24460ms となる．以上より，本実験により構築した最適化計算システムの見積もり時間は約 50140ms となる．50434ms と比べて多少の誤差が見られるが，比較的正しい値だといえる．

Table 5　基本機能の実行時間

Function	Execute Time (ms)
Receive File of Truss2d	83
Receive File of SA	118
Run Application of SA	50434
Return File of SA	63

4　終わりに

本研究では，最適化計算システムを構築するために，アプリケーション間の情報交換にはファイルを用い，各 Service にファイルによる情報交換やアプリケーションの実行を行う基本機能を付加することで，Web サービスを用いた汎用的なアプリケーション連携の仕組みを提案した．実際に提案システムを用いて最適化計算システムを構築し，数値実験により提案システムが稼動することを確認した．

今後の課題は，Service の情報を管理する Agent の実装，入出力ファイルの記述形式を統一するための機能の追加，Globus Toolkit 3 を利用して提案システムを実装することでトランザクションや認証等を考慮したセキュアで信頼性の高いシステムの構築を目指すことである．

参考文献

1) Carl Kesselman Ian Foxter. *The Grid : Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1998.

2) Ian Foster, and et.al. *Globus : A metacomputing infrastructure toolkit* International Jounal of Supercomputer Applications, Vol.11, No.2, 1997

3) Web Service. http://www.w3c.org/TR/xmlbase

4) Jeffrey M.Nick Ian Foster, Carl Kesselman and Steven Tuecke. *DRAFT document : The Physiology of the Grid : An Open Grid Services Architecture for Distributed Systems Integration.* June 2002.

5) WS-Resource Framework. http://www.globus.org/wsrf/

6) SOAP-RPC. http://www.w3.org/TR/2003/RFC-soap12-part2-20030624/

7) S.Kirkpatrick, C.D.Gelatt, Jr.M.P.Vecchi. *Optimization by Simulated Annealing.* Science, Vol220, No.4598, pp.671-680, 1983.