XMLベースGridRPCシステムの構築と評価

白砂 哲[†] 中田秀基[††,†]
松岡 聡[†,†††] 関口智嗣[††]

GridRPCは科学技術計算に多く用いられるGrid上のミドルウェアであるが，それぞれのGirdRPCシステムが独自のプロトコルを利用しているため，インテオペラビリティが重要な課題となっている．Webサービスの分野では，SOAPやWSDLといったXML基盤の標準仕様が用いられており，広く使用されることが期待されている．GridRPCにおいてもこれらの仕様を用いてインタオペラビリティを確保することが可能であると考えられるが，1) XML基盤のこれらの仕様がGridRPCに適した記述力を有しているか，2) コストが高いXMLを用いて十分な性能を得ることができるか，などが明らかではない．本研究において，SOAPとWSDLを基盤とするGridRPCを実装し，実験した結果，これらの技術を用いることは有用であることが分かった．SOAP基盤のGridRPCのナイーブな実装においては大きなオーバヘッドが大きいが，いくつかの性能向上を行なうことにより，本来のバイナリ転送に近い性能が得られた．一方，配列パラメタの扱いなどのGridRPC特有なさまざまな機能を実現することは，WSDLの制限により困難であることが分かった．

Construction and Evaluation of XML-based GridRPC systems

Satoshi Shirasuna,[†] Hidemoto Nakada,[††,†]
Satoshi Matsuoka[†,†††] and Satoshi Sekiguchi [††]

GridRPC is a class of Grid middleware for scientific computing. Interoperability has been an important lingering issue, because current GridRPC systems each employ its own private protocol. Web services, where XML-based interoperability schema standards such as SOAP and WSDL are expected to see widespread use, could be the medium of interoperability; however, it is not clear if 1) XML-based schemas have sufficient expressive power for GridRPC, as it embodies various sophisticated mechanisms not present in traditional RPCs, and 2) whether performance could be made sufficient, since there are inherent high costs with XML. Our experiments implementing GridRPC features with SOAP and WSDL-based modules indicate that the use of such technologies are more promising than previously reported. Although a naive implementation of SOAP-based GridRPC has severe performance overhead, application of a series of optimizations improves performance to be almost competitive with the original binary transport. However, encoding of various features of GridRPC including its comprehensive handling of array arguments proved to be somewhat difficult due to WSDL limitations.

1. はじめに

広域ネットワークを基盤としたGridコンピューティングが大規模科学技術計算の分野などで盛んに行なわれている．Ninf[1)]，NetSolve[2)]に代表されるGridRPCシステムは，Grid環境におけるRPC基盤のシステムであり，高レベルで使いやすいプログラミングモデルを提供している．これらは，セキュリティ，リソース管理，スケジューリングなどのGridサービスの複雑さを隠蔽し，また，並列処理を可能にする．さらに，サーバ側におけるIDLの管理や科学技術計算に特化したIDLの採用により，クライアント側にメモリ透過なAPIを提供する．そのため，GridRPCはさまざまな分野においてGrid上でのミドルウェアとして広く用いられている．

しかし，現在GridRPCシステムはフォールトトレランス，スケーラビリティ，スケジューリングなどのいくつかな研究課題を抱えている．GridRPCシステム間やその他のサービスとのインタオペラビリティも研究課題のひとつであり，標準化の必要がある．既存のGridRPCシステムは，それぞれが独自のプロ

† 東京工業大学
Tokyo Institute of Technology
†† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
††† 科学技術振興事業団
Japan Science and Technology Corporation

トコルを採用しており，あるシステムのクライアントからは異なるシステムのサーバ資源を利用できない．NetSolve–Ninf ブリッジ[3] などにより相互接続も行なわれているが，この方法では，1) それぞれのシステムに共通する機能のみが利用できる，2) プロトコル変換によりパフォーマンスが低下する，3) すべての GridRPC システム間のブリッジを作るのは現実的ではない，などの問題がある．

一方，コンピュータ関連の各分野において XML を用いてアプリケーションデータの連携，標準化や電子文書作成などが行なわれている．この標準化の流れは Web サービスの分野で顕著であり，SOAP[4]，WSDL[5]，UDDI[6] などの XML 基盤のプロトコル標準仕様が制定され用いられている．こらの標準技術を GridRPC でも使用することで，インタオペラビリティを確保できる．

しかし，Web サービスに用いられている技術を GridRPC に適応するにはいくつかの技術的課題がある．まず，ビジネスアプリケーションで多く用いられる Web サービス用の仕様に GridRPC に適した記述力があるか否かが明らかではない．例えば，Ninf や NetSolve の IDL では，参照渡しで渡される配列に対し，計算に必要な部分のデータを転送することで，仮想的な共有メモリを実現している．また，IDL をサーバ側で管理することで，クライアントには透過的な API を提供する．また，XML を用いることによる性能低下も問題である．例えば，SOAP では，配列データを要素ごとに XML 要素として記述するため，データサイズが増大するだけでなく，シリアライズ，デシリアライズにかかるこすとも大きくなる．7) では，これらのオーバヘッドはかなり大きいと報告されている．

本研究では，Web 技術が GridRPC システムの基盤として使用できるか否かを評価するため，SOAP，WSDL を基盤とする GridRPC を構築し，評価した．その結果，ナイーブな実装においてはオーバヘッドがかなり大きいことが分かったが，いくつかの性能改善手法を適応し性能を改善できた．しかし，現在の WSDL 仕様の制限から，上述の GridRPC のさまざまな機能の実現は困難であり，科学技術計算用の IDL として用いるためにはいくつかの拡張が必要となることが分かった．

2. SOAP 関連技術と GridRPC システム

インターネットにおいて Web は広く用いられる重要なインフラであるため，そのインフラを用いてさまざまなサービスが展開されている．Web サービスと称されるこれらのサービスを標準化するため，いくつかの XML 基盤の仕様が作成され，またそれを利用するためのツール群が整備されつつある．ここでは，本研究に関連のある仕様の概要を述べる．

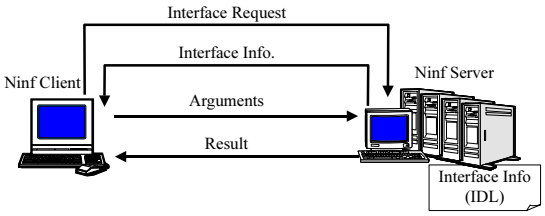

図 1 Ninf システム図

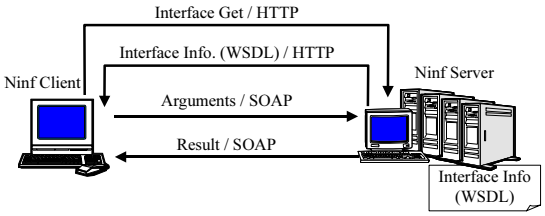

図 2 SOAP 基盤 Grid RPC システム図

SOAP SOAP (Simple Object Access Protocol)[4] は，分散環境においてメッセージ交換のための仕様である．言語やプラットフォームに非依存であり，HTTP が通信レイヤとして用いられることが多い．本研究では，SOAP を GridRPC のメッセージ交換レイヤに用い，その性能を評価する．

WSDL WSDL (Web Service Definition Language)[5] は，Web サービスのインタフェース情報を記述するための仕様である．WSDL はいくつかの Web サービスへのバインディングを規定しているが，SOAP とともに用いられることが多い．本研究では，WSDL を GridRPC の IDL として用い，WSDL が科学技術計算用の IDL としての十分な記述力を有しているかを評価する．

本研究では，これらの規格を GridRPC に用い，インタオペラビリティの高いシステムの開発を行なう．

図 1 は現在の GridRPC システム (Ninf) のシステム図である． 計算ライブラリのインタフェース情報は Ninf IDL を用い記述する．クラアント側での IDL 管理の繁雑さを省き，透過的な API をユーザに提供するため，IDL の管理はすべてサーバ側で行なわれる．そのため，クライアントはインタフェース情報を実行時にサーバより取得する．インタフェース情報の取得，計算ライブラリの呼び出しは XDR を用いた独自プロトコルで行なわれる．

それに対し，図 2 が本研究で提案する XML 技術を基盤とする GridRPC システムのシステム図である．インタフェース情報は WSDL で記述され，インタフェース情報の交換は現在の Ninf と同様に実行時に行なわれる．クライアントは，計算ライブラリの実行前に HTTP Get を用いて WSDL ファイルを取得する．実際のパラメタの交換には SOAP が用いられる．呼び出しに必要なライブラリ名，パラメタは SOAP

メッセージにエンコードされ，サーバに送信される．結果も同様に SOAP メッセージとして返送される．

この SOAP を基盤とした GridRPC システムには以下の利点があると考えられる．

標準技術 標準的な仕様を使用することで，システムの開発，およびサーバ/クライアントの作成に際に XML ライブラリや SOAP 用ツール群などの既存のツールが利用可能である．また，HTTP を SOAP の通信レイヤとして用いることで，HTTP 上の既存のセキュリティ技術が利用可能である．

インタオペラビリティ SOAP 基盤の GridRPC 間でのインタオペラビリティが達成できる．また，SOAP 基盤の一般 Web サービスを GridRPC のクライアントから利用することも可能になる．

ファイアウォール・フレンドリ 通信基盤に HTTP を用いているため，多くのファイアウォール内のホストとも通信できる．ファイアウォールを導入している組織は多くあり，既存の GridRPC を導入する際に大きな障害となっている．しかし，SOAP を基盤とした GridRPC では，ファイアウォールの設定を変更せずに導入できる．また，プライベートネットワーク内のクライアントも一般的な HTTP プロキシを用いることでサーバに接続できる．

一方，SOAP，WSDL を用いることで以下の技術的課題が考えられる．

パフォーマンスの低下 XML ベースの SOAP を用いることで速度低下が問題となる．これは，XML 表現を用いることでデータが 10 倍以上になってしまうことや，XML のシリアライズ/デシリアライズのコストが高いことに起因する．7) では，科学技術計算に用いるためのいくつかの RMI プロトコルを比較している．これによると，SOAP における性能は，Java RMI，Nexus RMI と比較して，数十倍の性能低下が見られる．本研究では，この性能低下が何に起因しているのかを分析し，性能の改善を行なう．

SOAP，WSDL の記述力 SOAP，WSDL はもともとビジネスアプリケーションを念頭において作成されたものであるため，GridRPC のような科学技術計算向けアプリケーションに用いるのに十分な機能を持っているか評価が必要である．現在の GridRPC システムである Ninf，NetSolve の IDL では，Grid 上の分散環境を透過に扱うため，CORBA に代表されるビジネスアプリケーション IDL では記述できない機能を提供している．例えば，Fortrun，C，C++ のような言語においては，配列サイズを動的に取得することはできない．しかし，Grid 環境上において参照渡しに相当するメモリ透過なプログラミングモデルを提供するためには，適切な範囲のメモリを読み込み，転送する必要がある．この実現のため，Ninf IDL では，パラメタ間の依存関係を記述でき，この機能を用いて配列サイズの指定を行なえる．また，大きな配列のすべてを転送するのを防ぐため，配列のストライド転送や部分配列の転送をサポートしている．これらの機能を SOAP，WSDL の枠組でどのように実現できるのかは明らかではない．

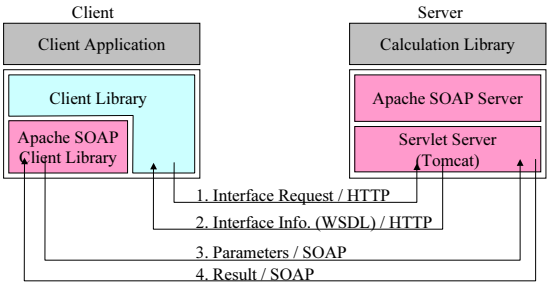

図 3 Ninf on Apache SOAP

3. 第 1 プロトタイプ: Ninf on Apache SOAP

Apache SOAP は，Java 用の SOAP ライブラリであり，広く用いられている．Apache SOAP は，SOAP のサーバに接続するためのクライアントライブラリと Servlet として動作する Apache SOAP サーバから成る．サービス作成者は Java や Java Script を用いて SOAP サービスを実装し，Web インタフェースか独自のサービス記述言語によりサービスをサーバに登録する．クライアント作成者は Apache SOAP API を用い，サーバに接続する．Apache SOAP では，実行時にインタフェース情報を取得する手段を提供していないため，クライアント作成者はサーバ名，パラメタ名，型情報などを明示的に記述する必要がある．

我々は，XML 技術基盤の GridRPC の第 1 プロトタイプとして，Ninf on Apache SOAP を構築した．Ninf on Apache SOAP は Apache SOAP をメッセージレイヤとして用い，API は既存の Ninf と同等のものを提供している．図 3 は，Ninf on Apache SOAP の概要である．Apache SOAP は WSDL をサポートしていないため，IDL の管理を行なう WSDL モジュールを実装した．Ninf on Apache SOAP は予備評価のための第 1 プロトタイプであるため，Apache SOAP に起因する 1) 複数の Out パラメタの非サポート，2) In/Out パラメタの非サポート，といった制限がある．すべてのモジュールは Java で実装されている．

サーバ側の実装 GridRPC システムのサーバには，直接 Apache SOAP サーバを用いる．計算ライブラリは，Apache SOAP サーバに登録する．その際，Java のクラスファイルより生成する WSDL ファイルもともに登録する．

表 1　PrestoII cluster の仕様

CPU	Pentium III 800MHz
Memory	640MB
Motherboard	MSI MS-6120N
Network Card	DEC 21140AF(Planex)
OS	Linux 2.2.19
Java	IBM Java 1.3.0
Servlet	Jakarta Tomcat 3.2.3

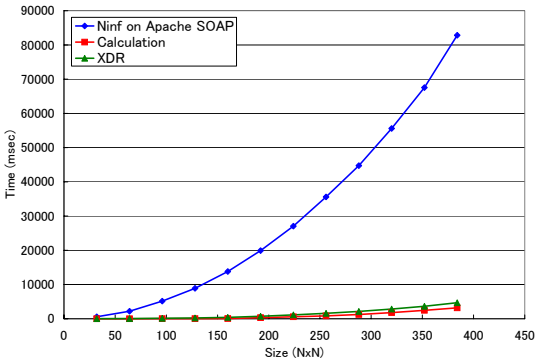

図 4　Ninf on Apache SOAP の性能

クライアント側の実装　Ninf on Apache SOAP のクライアントライブラリは，Apache SOAP ライブラリをメッセージ通信のレイヤに用いる．GridRPC の呼び出しでは，まずクライアントプログラムから渡された URL からインタフェース情報が記述された WSDL ファイルを取得する．そしてそのインタフェース情報を用いてパラメタをシリアライズし，SOAP メッセージとしてサーバに送信する．サーバ側での計算終了後，返送された SOAP メッセージを再び Apache SOAP ライブラリを用いてデシリアライズし，クライアントプログラムに返す．この呼び出しは既存の GridRPC と同等の API を用いて行なわれているため，Apache SOAP ライブラリはユーザには隠蔽される．

単純な行列の積を行なう計算ライブラリを用いて Ninf on apache SOAP の性能評価を行なった．評価には，東京工業大学松岡研究室の PrestoII クラスタのノードを用いた．各ノードの仕様と評価に用いたソフトウェア環境を表 1 に示す．評価では，同一 LAN 上のクライアント，サーバを用いた．

図 4 が，性能評価の結果である．横軸は配列のサイズ，縦軸は実行にかかる時間を示す．比較のため，グラフには，Java で実装された XDR 基盤の GridRPC の性能と，計算ライブラリの実行自体にかかる時間を含む．グラフによると Ninf on Apache SOAP にかかる時間は計算時間よりもはるかに長く，XDR 基盤の GridRPC と比べてもオーバヘッドが大きい．

オーバヘッドが大きい理由として，Apache SOAP の実装がサイズが大きい XML データに適していないことがあげられる．Apache SOAP では SOAP メッセージの解析に DOM[8] パーサを用いている．DOM パーサは XML データ構造を DOM オブジェクトのツリーとしてメモリ上に構築するため，解析前にすべての XML データを受信する必要がある．そのため，Apache SOAP は SOAP メッセージの受信が終了するまでデシリアライズの処理を開始できない．また，DOM ツリーをメモリ上に構築するため，大量のメモリを必要とする．実際に，これは XML のテキスト表現の何倍もの大きさとなる．

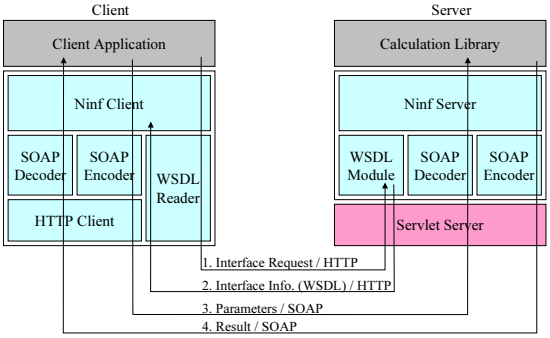

図 5　Ninf-on-SOAP Architecture

以上に述べた Apache SOAP の実装上の制限から，Apache SOAP を用いたシステムの性能向上は難しく，性能向上を行なうためにはメモリ上にデータを保持しない SAX パーサの採用や，受信と同時にデータの解析を行なうシステムを構築する必要があると考える．

4. 第 2 プロトタイプ: Ninf on SOAP

性能改善のため，第 2 プロトタイプ，Ninf on SOAP を構築した．Ninf on SOAP では速度向上のため，独自のシリアライザ，デシリアライザを用いる．また，複数の Out パラメタや In/Out パラメタをサポートする．

図 5 は Ninf on SOAP のシステム図である．第 1 プロトタイプと同様にすべてのモジュールは Java で実装されている．

4.1 Ninf on SOAP サーバ

Ninf on SOAP サーバは Servlet として動作する．Ninf on SOAP クライアントから SOAP メッセージを受信し，そのメッセージを WSDL ファイルに記述されているインタフェース情報に基づきデシリアライズする．その後，適切な計算ライブラリを呼び出す．計算ライブラリの実行終了後，結果を SOAP メッセージにシリアライズし，クライアントに返送する．

WSDL モジュール　サーバは登録された WSDL ファイルの読み込みを行ない，インタフェース情報を取得する．WSDL ファイルのパーズと解析に

は，Java 用の WSDL ライブラリ，WSDL4J を用いる．WSDL ファイルには以下の情報が含まれる．

- 計算ライブラリのクラス名
- 計算ライブラリのメソッド名
- パラメタの名前，型情報，In/Out モード
- パラメタの順序

SOAP デシリアライザ　SOAP デシリアライザはクライアントから送られて生きた SOAP メッセージをデシリアライズする．SOAP メッセージは WSDL によって記述されたインタフェース情報に従い解析される．XML の解析には，性能改善とメモリ使用量の低減のために SAX パーサ[9] を用いる． SAX パーサはイベントドリブンなパーサで XML データ全体をメモリ上に保持する必要がない．そのため，DOM パーサと比較してデシリアライズの速度が向上する．さらに，SAX パーサは XML データの受信と同時に解析を行なえるため，SOAP メッセージの受信とデシリアライズを同時に行なえる．SOAP デシリアライザは WSDL ファイル内に記述されたパラメタの順序に従い，デシリアライズしたパラメタをベクトル内に保存する．

Invocator　Invocator は計算ライブラリを実行する．現在のプロトタイプ実装においては，呼び出せるのは Java のメソッドみであるが，将来は C，C++，Fortran などの関数の呼び出し可能にする．その際，配列データは参照渡しで渡される．In/Out モードや型の情報は WSDL で記述されたものを用いる．

SOAP シリアライザ　SOAP シリアライザは，Java のメソッドの実行後，Out パラメタ，In/Out パラメタをシリアライズし，クライアントに SOAP メッセージとして返送する．HTTP Response メッセージにおいては，Content-Length ヘッダフィールドをつけることは必須ではなため，Ninf on SOAP サーバでは，Content-Length ヘッダを省略する．これにより，シリアライズと送信を同時に行なえ，性能が向上する．この詳細は 5 節で述べる．

4.2 Ninf on SOAP クライアント

Ninf on SOAP のクライアントライブラリは既存の Ninf クライアントと同等な単純で透過的な API を提供する．GridRPC の呼び出しの際に，クライアントは指定された URL から WSDL ファイルを取得し，インタフェース情報を得る．クライアントは，そのインタフェース情報を用い，In パラメタ，In/Out パラメタをシリアライズし，SOAP メッセージとしてサーバに送信する．サーバ側での計算ライブラリの実行後，クライアントはサーバから返送された Out パラメタ，In/Out パラメタを受信し，インタフェース情報に従いデシリアライズする．

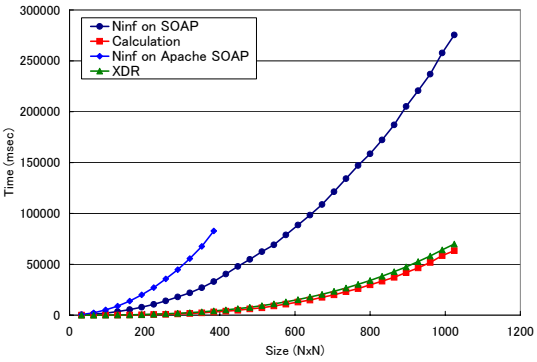

図 6　Ninf on SOAP の性能

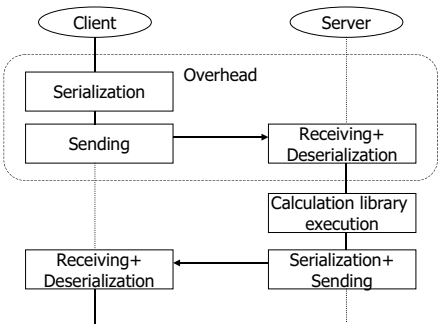

図 7　Excecution flow

SOAP メッセージの送受信，WSDL ファイルの読み込みなどの処理はサーバ側と同様である．SOAP デシリアライザも同様に SAX パーサを用いている．異なる点は，HTTP Post においては，Content-Length ヘッダフィールドが必須であるため，クライアントは SOAP メッセージをメモリ上に構築しメッセージ長を計算した後に，送信を行なう．

5. 性能評価と改善

Ninf on SOAP の性能評価を 3 節で述べた環境において評価した．図 6 が結果である．比較のため，グラフには第 1 プロトタイプ (Ninf on Apache SOAP) の性能も示してある．第 1 プロトタイプに比べ，性能は大幅に向上したが，XDR を用いるシステムに比べオーバヘッドはかなり高い．

オーバヘッドの原因を特定するため，どの処理に時間がかかっているのかを調べる分析を行なった．図 7 は Ninf on SOAP の実行の流れを示す．点線に囲まれた部分が計算ライブラリ実行前のオーバヘッドである．このオーバヘッド内のそれぞれのフェーズ (シリアライズ，メッセージ送信，デシリアライズ) にかかる時間を測定した．図 8 に示される結果によると，LAN 環境においては，メッセージサイズの増大にかかわら

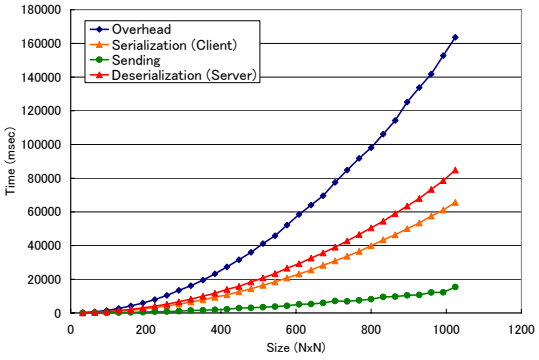

図 8 オーバヘッド解析

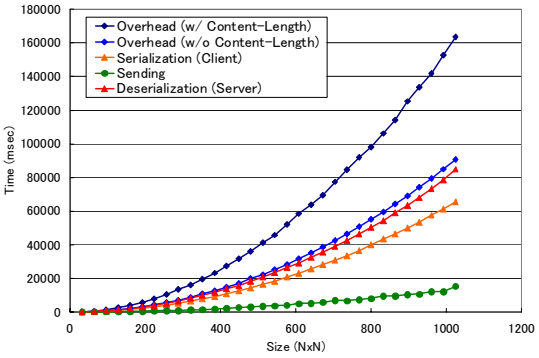

図 9 HTTP Content-Length ヘッダの有無によるオーバヘッド

ず送信時間は比較的少なく，シリアライズ，デシリアライズが大きなオーバヘッドの原因となっている．

この分析結果をもとにさまざな性能改善手法を適応し，性能の改善を行なった．

性能改善 1: HTTP Content-Length ヘッダフィールド

オーバヘッドが大きい原因の一つは，図 7 に示されるようにシリアライズとデシリアライズが並列で行なわれていないことである．前述したように，これは HTTP Post では HTTP Content-Length ヘッダフィールドが必須であるため，SOAP メッセージをメモリ上に構築し，サーバに送信する前に HTTP Content-Lenght ヘッダを付加する必要があるためである．しかし，SOAP メッセージは XML で記述されているため，HTTP サーバは XML タグのペアの対応を取ることでメッセージの終端を検知できる．そのため，RFC に違反する方法ではあるが，Content-Length ヘッダフィールドを省略し，シリアライズとデシリアライズを並列して行なうことが可能である．

この性能改善手法においてどの程度のオーバヘッドが低減されるのかの評価を行なった．図 9 によると，Content-Length ヘッダフィールドを付加する場合のオーバヘッドは，おおよそシリアライズとデシリアライズの処理時間の和であるのに対し，Content-Length ヘッダフィールドを省いた場合のオーバヘッドは，おおよそデシリアライズの処理時間に等しい．結果として，オーバヘッドが約 45%低減されている．この結果は，シリアライズとデシリアライズの処理の並列化は，SOAP を基盤とする GridRPC においては極めて効率的であることを示す．

HTTP Content-Length ヘッダフィールドを省略することで，パフォーマンスの向上が得られたが，この方法は RFC で定められる仕様に反する．しかし，HTML の次バージョンである XHTML や SOAP，WSDL など XML 基盤のデータが HTTP 上で交換されることは多く，今回の場合のように Content-Length ヘッダフィールドを省くことで恩恵を得られる場合は多いと考える．今後の課題として，RFC に従ったシリアライズとデシリアライズの並列化の実現方法を評価する．第 1 の方法は，SOAP メッセージの長さをシリアライズの処理前に計算し，その後，実際にシリアライズを行なう方法がある．この方法では，SOAP メッセージの長さの計算が新たなオーバヘッドになる可能性がある．第 2 の方法としては，XML におけるそれぞれの要素の長さを推定する方法がある．この時，数値型の XML 表現の長さは固定ではないため，最大長を取っておいて，短い場合にはスペースで埋める．この方式では，シリアライズとデシリアライズの処理は並列化できるが，スパースな配列データなどではサイズが増大する可能性がある．

性能改善 2: Base64 エンコーディング

SOAP は XML 基盤のプロトコルであるため，メッセージサイズが本来のデータサイズに比べて膨大になる．また，XML のシリアライズとデシリアライズのコストも高い．SOAP では，バイナリデータを転送するために，Base64 エンコーディングをサポートしている．ここでは，性能改善の手段として，配列データをバイナリデータとして扱い，Base64 エンコーディングを適応した．今まで配列のそれぞれの要素が XML 要素として表されていたものが，配列の XDR 表現が Base64 エンコードされて送られることになる．

図 10 は，いくつかのエンコーディングにおけるオーバヘッドを比較した物である．Base64 を適応することで，純粋な SOAP を用いた場合に比べ，オーバヘッドの約 74%が削減されたことがわかる．これは，配列データを Base64 エンコードすることで，XML タグの数が大幅に減少し，シリアライズ，デシリアライズの速度が向上したためである．

性能改善のまとめ

図 11 は，それぞれの性能改善とその組合せを適応した場合の性能結果を示す．ナイーブな実装である Ninf on Apache SOAP 比べ，かなりの性能改善が行なわれたことが分かる．実際に，XDR 形式を用いたもと比較してもそれほどオーバヘッドは大きくなく，実用

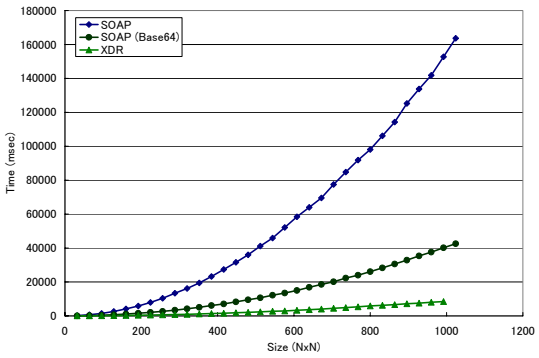

図 10 Base64 エンコーディングによるオーバヘッド

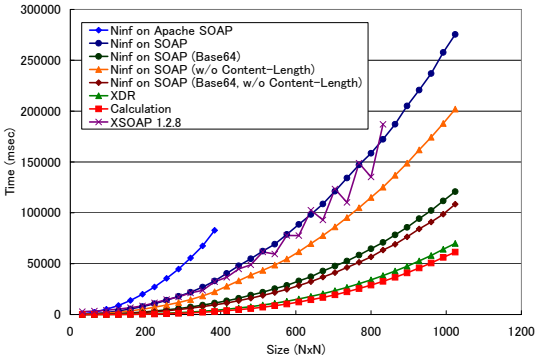

図 11 Overheads of different encodings

可能であると考える．

我々の実装と他の実装と比較するため，インディアナ大学で開発されている SOAP 基盤の RMI システム，XSOAP[10] との比較を行なった．XSOAP は多次元配列が未サポートであるため，行列の記述には 1 次元配列を用いた．性能評価の結果は，我々の性能改善を行なっていない Ninf on SOAP とほぼ等しいことが分かった．

6. SOAP，WSDL の記述力

ここで，SOAP や WSDL が科学技術計算用の IDL の機能を十分に表現できるかについて考察する．

配列サイズ記述の欠如 WSDL は配列サイズを記述する手段を提供していない．科学技術計算で用いられる IDL においては，配列サイズは他の引数に依存して記述する必要がある．特に C や Fortran などの配列サイズ取得する手段がない言語においては，重要な機能である．そのため，WSDL が配列サイズを他の引数に依存して記述する手段を提供することは重要である．

部分配列，配列のストライド 同様に数値計算を行なう際には，部分配列や配列のストライド転送など，配列の一部分のみを必要とすることが多い．SOAP では Partially Transmitted Array や Sparse Array などのデータ型をサポートしているため，これらのデータ型を表現することは可能である．しかし，WSDL の仕様ではこれらのデータを表現する仕組みを提供していない．

インタオペラビリティ WSDL ではパラメタの順序を operation の parameterOrder 属性として表現する．Ninf on SOAP はこの属性を用い，パラメタの順序を指定する．しかし，parameterOrder 属性はオプションであるため，parameterOrder 属性を用いていない WSDL を用いるシステムとの接続では，パラメタの順序を決定できない．現在，Ninf on SOAP では，parameterOrder 属性がない場合，In パラメタ，Out パラメタの順序で処理を行なう．また，それぞれのパラメタの順序は WSDL に記述されている順序に従う．

その他にも，WSDL では，配列の記述法など同一のデータを表現する方法が複数存在するなどの問題もある．

7. 関 連 研 究

RPC の最適化を行なう研究は数多く存在する．しかし，それらは直接本研究の XML を用いる枠組には適用できない．11) では，GridRPC システムを CORBA と定性的，定量的に比較している．それぞれのシステムでは，通信はバイナリで行なっているため，通信速度は同等である．しかし，GridRPC システムでは，クライアントや IDL の記述が容易であるといった利点がある．7) では，さまざまな RMI プロトコルの比較を行なっており，SOAP 基盤の RMI は Java RMI や Nexus RMI に比べて遅いといった結果が得られている．この論文では，通信の効率だけを測定しており，最適化や科学技術計算における表現力などの評価は行なっていない．これに続く 12) では，SOAP 基盤の RMI システムである SoapRMI(XSOAP の前バージョン) の実装がなされている．SoapRMI では，SOAP に適した XML パーサである XML Pull Parser を用いている．これは，一般的な RMI の実装であり，本研究で対象としている GridRPC とは異なる．13) では，XML 基盤の GridRPC システムを提案している．このシステムは WSDL に似た XML を用い，インタフェース情報を記述する．実際のデータ転送にはパフォーマンスに優れるバイナリ形式を用いている．

8. まとめと今後の課題

本研究では，Web サービスで用いられる XML 技術の GridRPC への適応性を評価した．結果として，ナイーブな実装では大きなオーバヘッドが生じることが分かった．しかし，シリアライズとデシリアライズ

の処理の並列化や，Base64 エンコーディングによる XML タグの削減などの性能改善を行なうことにより，かなりの性能向上が達成でき，バイナリ形式を用いるシステムとの性能差が減少した．一方，WSDL を科学技術計算用の IDL として用いる場合，パラメタ間の依存関係の記述，部分配列，配列のストライドの記述ができないことが分かった．これらを解決するためには，WSDL 仕様の拡張が必要である．

今後の課題として，我々はさらなる性能改善を行なう．性能改善の方法のひとつは，GridRPC に最適化された XML パーサの開発である．GridRPC においては，計算ライブラリの呼び出しの前にインターフェース情報が交換される．そのため，そのインタフェース情報を用い，交換されるメッセージに適したパーサを動的に生成することで性能を向上できる．これらのパーサの生成時間は，実行時間の長い計算ライブラリの呼び出しの場合は，相対的に短いため大きなオーバヘッドとはならない．また，実行時間が短く，繰り返し呼ばれる計算ライブラリでは，キャッシュを行なう．

参 考 文 献

1) Nakada, H., Takagi, H., Matsuoka, S., Nagashima, U., Sato, M. and Sekiguchi, S.: Utilizing the Metaserver Architecture in the Ninf Global Computing System, *High-Performance Computing and Networking '98, LNCS 1401*, pp. 607–616 (1998).
2) Casanova, H. and Dongarra, J.: Applying NetSolve's Network-Enabled Server, *IEEE Computational Science & Engineering*, Vol. 5, No. 3, pp. 57–67 (1998).
3) Nakada, H. and SatoshiMatsuoka, S.S.: Bridging Ninf and NetSolve (1997).
4) Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., rystyk Nielsen, H., Thatte, S. and Winer, D.: Simple Object Access Protocol (SOAP) 1.1, W3C Note (2000). http://www.w3.org/TR/2000/NOTE-SOAP-20000508.
5) Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S.: Web Services Description Language (WSDL) 1.1, W3C Note (2001). http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
6) : UDDI Technical White Paper, Technical report, uddi.org (2000).
7) Govindaraju, M., Slominski, A., Choppella, V., Bramley, R. and Gannon, D.: Requirements for and Evaluation of RMI Protocols for Scientific Computing, *Proc. of SuperComputing 2000* (2000).
8) Hors, A. L., Hegaret, P. L., Wood, L., Nicol, G., Robie, J. and Champion, M.: Document Object Model (DOM) Level 2 Core Specification Version 1.0, W3C Recommendation (2000). http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113.
9) Simple API for XML (SAX): http://www.saxproject.org/.
10) Indiana University: XSOAP toolkit (aka SoapRMI). http://www.extreme.indiana.edu/soap/.
11) Suzumura, T., Nakagawa, T., Matsuoka, S., Nakada, H. and Sekiguchi, S.: Are Global Computing Systems Useful? - Comparison of Client-Server Global Computing Systems Ninf, NetSolve versus CORBA, *Proc. of International Parallel and Distributed Processing Symposium* (2000).
12) Slominski, A., Govindaraju, M., Gannon, D. and Bramley, R.: Design of an XML based Interoperable RMI System: SoapRMI C++/Java 1.1, *Proc. of The 2001 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2001)* (2001).
13) Widener, P., Eisenhauer, G. and Schwan, K.: Open Metadata Formats: Efficient XML-Based Communication for High performance Computing, *Proc. of the 10th IEEE International Symposium on High Performance Distributed Computing-10 (HPDC-10)*, pp. 371–380 (2001).