Web サービス技術を基盤とする GridRPC システムの評価

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

Grid 上のミドルウェアである GridRPC は科学技術計算に多く用いられる。しかし、それの GridRPC システムが独自のプロトコルを利用しているため，インタオペラビリティが重要な課題となっている．一方，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．そのため，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) ビジネスアプリケーションを念頭としたこれらの仕様が 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．本研究では，SOAP と WSDL を基盤とする GridRPC を実装し，評価した．その結果，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り，WSDL の仕様の拡張が必要であることが分かった．

Evaluating Web Service Based Implementations of GridRPC

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

GridRPC is a class of Grid middleware for scientific computing. Interoperability has been an important issue, because current GridRPC systems each employ its own protocol. Web services, where XML-based standards such as SOAP and WSDL are expected to see widespread use, could be the medium of interoperability; however, it is not clear 1) if XML-based schemas have sufficient expressive power for GridRPC, and 2) whether performance could be made sufficient. Our experiments 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. However, encoding of various features of GridRPC proved to be somewhat difficult due to WSDL limitations. The results show that GridRPC systems can be based on Web technologies, but there needs to be work to extend WSDL specifications.

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，高レベルで簡便な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そのひとつである．これは，既存の GridRPC システムは，それぞれが独自のプロトコルを採用してることに起因する．そのため，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．

一方，コンピュータ関連の各分野において XML を用いたアプリケーションデータの連携，標準化や電子文書

† 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

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[3]，WSDL[4]，UDDI[5] 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．こらの標準技術を採用することで，GridRPC においてもインタオペラビリティを確保できると考える．

しかし，Web サービスに用いられている技術を GridRPC に適用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．また，XML を用いることによる性能低下も問題となる．

本研究では，SOAP，WSDL を基盤とする GridRPC を構築し，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が，いくつかの性能改善手法を適用し性能を改善出来た．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．

SOAP SOAP (Simple Object Access Protocol)[3] 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)[4] 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一般的である．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．

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う．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を用いる．

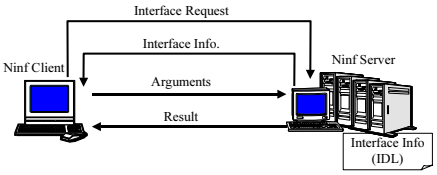

図 1 Ninf システム図

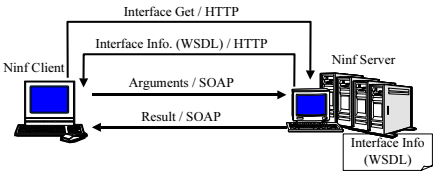

図 2 SOAP 基盤 Grid RPC システム図

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 (Ninf on SOAP) のシステム図である．インタフェース情報は WSDL で記述し，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う．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 ファイルを取得し，実際のパラメタの交換には SOAP を用いる．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し，サーバに送信する．結果も同様に SOAP メッセージとして返送する．

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 ライブラリ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

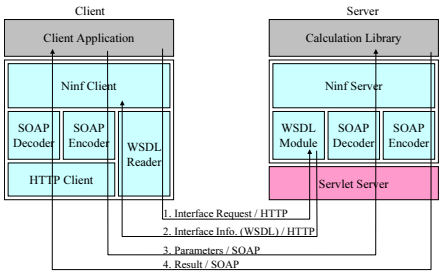

図 3 Ninf on SOAP システム概要

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．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コストとも大きくなる．6) 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．例えば，Ninf IDL では，パラメタ間の依存管理を記述でき，配列サイズの指定を行える．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．

3. Ninf on SOAP システム

Web サービス技術を用いた GridRPC システムとして，Ninf on SOAP を構築した．図 3 は Ninf on SOAP のシステム図である．すべてのモジュールは Java で実装されている．また，性能向上のため，独自のシリアライザ，デシリアライザを用いる．

3.1 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，そのメッセージを登録されている 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 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 ライブラリである WSDL4J を利用する．

SOAP デシリアライザ クライアントから送られてきた SOAP メッセージを WSDL によって記述されたインタフェース情報に従いデシリアライズする．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[7] 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[8] 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 パーサでは XML データの受信と同時に解析を行なうことが可能なため，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．

Invocator 計算ライブラリの呼び出しを行う．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 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．

3.2 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．その後，そのインタフェース情報を用いて 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．

4. 性能評価と改善

単純な行列の積を行なう計算ライブラリを用いて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．各ノードは 640MB のメモリを搭載したデュアル Pentium III 800MHz ノードで，100Base-TX イーサーネットスイッチで接続されている．主なソフトウェア環境は，Linux

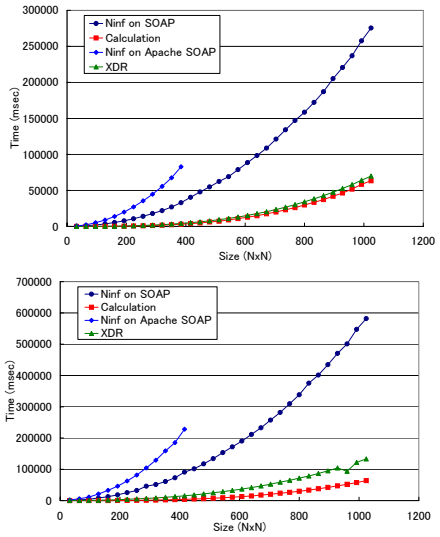

図 4 Ninf on SOAP の性能 (上段: LAN，下段: WAN)

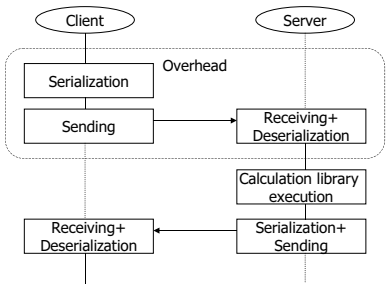

図 5 Execution flow

2.2.19，IBM Java 1.3.0，Jakarta Tomcat 3.2.3 である．LAN 環境における評価に於いては，サーバ，クライアントともに同一 LAN 上のノードを用いた．WAN に置ける評価では，PrestoII のノードをサーバとして用い，クライアントは産業技術総合研究所のノードを用いた．クライアントは，SPARC 333MHz×6 の Sun Ultra-Enterprise マシンで，960MB のメモリを搭載する．主なソフトウェア環境は，Solaris 5.7，Sun Java 1.3.0，Jakarta Tomcat 3.2.3 である．

図 4 は LAN 環境と WAN 環境に於ける結果である．比較のため，Java 用の SOAP ライブラリ Apache SOAP 上にナイーブに実装したシステムである Ninf on Apache SOAP の性能も示してある．それに比べ，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大きい．

オーバヘッドの原因を特定するため，各処理フェーズごとの分析をした．図 5 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各フェーズ (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 6 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．WAN 環境においてもシリアライズ，デシリアライズのオーバヘッドは大きいが，メッセージサイズの増大によりデータ転送のにかかる時間が大部分を占めている．

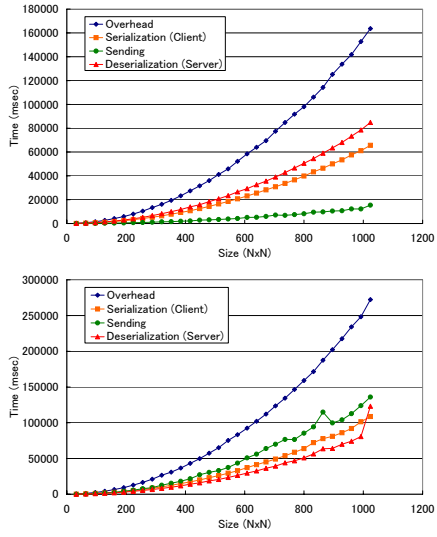

図 6 オーバヘッドの解析 (上段: LAN，下段: WAN)

以下では，この分析結果をもとにさまざまな性能改善手法を適用し，性能の改善を行う．

性能改善 1: HTTP Content-Length ヘッダフィールド

オーバヘッドの原因の一つとして，図 5 に示すように，シリアライズとデシリアライズの処理が並列でない点が挙げられる．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 7 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

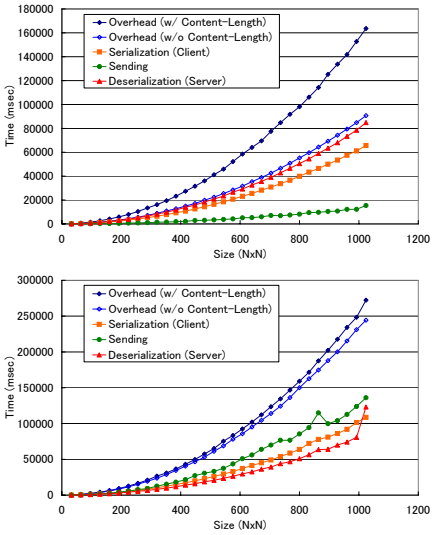

図 7 HTTP Content-Length の有無によるオーバヘッドの変化 (上段: LAN，下段: WAN)

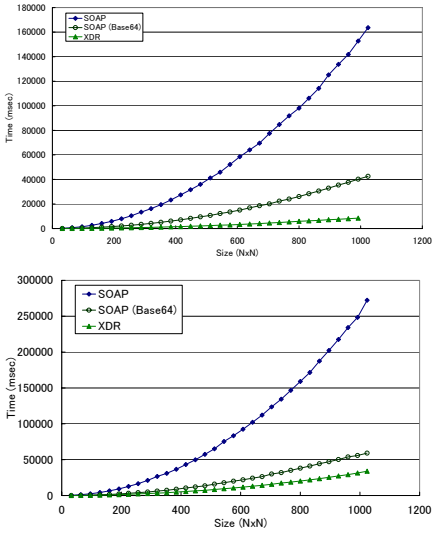

図 8 エンコーディングの違いによるオーバヘッドの変化 (上段: LAN，下段: WAN)

て，オーバヘッドが約 45%低減されている．WAN 環境においても，LAN 環境に程ではないがオーバヘッドの低減が見られる．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ることが分かったが，この方法は RFC で定められる仕様に反する．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．考えられる方法として，1) メッセージの長さとして最大長を取っておき，不足分をスペーで埋める方法，2) HTTP 1.1 の機能である Chunked Transfer Coding を利用し，メッセージを長さ付の断片に分割する方法がある．

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用した．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 8 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ものである．純粋な SOAP を用いた場合に比べ，LAN 環境，WAN 環境の双方に於いてオーバヘッドの約 75%が削減されたことが分かる．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と考えられる．

5. 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，In/Out 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．

6. 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．6) 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記述性の評価は行なっていない．これに続く 9) 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．10) 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．

7. 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 への適用性を評価した．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) 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.
4) 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.
5) UDDI.org: UDDI Technical White Paper (2000).
6) 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).
7) Simple API for XML (SAX): http://www.saxproject.org/.
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) 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).
10) 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).