

複数ドメイン環境で QoS を保証するクラウドのための資源管理フレームワーク

竹房あつ子[†] 中田 秀基[†] 高野 了成[†] 柳田 誠也[†] 大久保克彦[†]
工藤 知宏[†] 田中 良夫[†]

[†] 産業技術総合研究所 情報技術研究部門 〒 101-0021 東京都千代田区外神田 1-18-13 秋葉原ダイビル 11 階
E-mail:

{atsuko.takefusa,hide-nakada,takano-ryousei,seiya.yanagita,ookubo-k,t.kudoh,yoshio.tanaka}@aist.go.jp

あらまし クラウド環境で QoS を保証した環境を提供するには、計算資源と計算資源間のネットワークの両方を同時に確保・提供する必要がある。また、これらの資源群は広域に分散し、複数ドメインから提供されるため、その資源のモニタリング情報の収集も課題となる。我々は、資源管理フレームワーク GridARS を開発しており、GridARS RMS により複数ドメインから提供される資源を選択・確保し、GridARS DMS でそのモニタリングを行う。本研究では、多様な資源のために規定された予約インタフェース GNS-WSI3 を採用した RMS および DMS を、Globus Toolkit 4.2 および Apache CXF を用いて開発した。システムの有効性を示すため、エミュレーション環境を構築して RMS および DMS の基本性能を調査し、各システムが非常に高速に処理できることを示す。また、RMS と DMS の連携実験を SC09 国際会議で行ったので報告する。

キーワード 資源管理, モニタリング, ウェブサービス, クラウド, QoS

Resource Management Framework for QoS-guaranteed Cloud over Multi-domain Resources

Atsuko TAKEFUSA[†], Hidemoto NAKADA[†], Ryousei TAKANO[†], Seiya YANAGITA[†], Katsuhiko OHKUBO[†], Tomohiro KUDOH[†], and Yoshio TANAKA[†]

[†] Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST) 1-18-13 Soto-kanda, Chiyoda-ku, Tokyo 101-0021, Japan

E-mail:

{atsuko.takefusa,hide-nakada,takano-ryousei,seiya.yanagita,ookubo-k,t.kudoh,yoshio.tanaka}@aist.go.jp

Abstract Resource management and monitoring are key issues for constructing a QoS-guaranteed Cloud over multi-domain resources. We have been working on development of the GridARS resource management framework, which mainly consists of a resource management system RMS and a distributed monitoring system DMS. This paper describes development of new versions of GridARS RMS and DMS, based on the GNS-WSI3 Web services-based resource reservation interface. We have developed the GT4 and PureWS versions using Globus Toolkit 4.2 and Apache CXF, respectively. We evaluate the basic performance of developed RMS and DMS with experiments on our emulated environment and show their fast operation times. We also report our SC09 demonstration in cooperation with RMS and DMS.

Key words Resource management, Monitoring, Web Services, Cloud, QoS

1. はじめに

クラウドとは、計算機やストレージ等のハードウェアやそれ

らを仮想化するプラットフォーム及びその上で動くアプリケーションを、インターネットを介して“サービス”として提供するものである。これらのサービスを実際に処理するハードウェア

ア資源は、各資源提供者側（ドメイン）で管理されているため、複数ドメインから提供される資源を組み合わせるクラウドサービスを提供するには、以下のような課題がある。

- 異なるドメイン間のネットワークも資源として扱い、その性能を保証する。
 - サービスに必要な性能を満たす資源の組み合わせを適切に探索する。
 - 複数ドメインから提供される複数の資源を、統一されたインタフェースで確保する。
 - 確保された資源を仮想化し、1つのインフラとしてユーザに提供する。
 - 確保された資源の利用状況を収集し、ユーザに提供する。
- ここで、“ユーザ”はハードウェア資源の利用者、すなわちクラウドサービスの利用者、またはクラウド上に独自のサービスを構築して第三者にその提供するクラウドサービス提供者のことを指す。

これらの課題を解決するため、我々は資源管理フレームワーク GridARS を開発している [1]。GridARS は、予約を前提として複数ドメインから提供される資源を、性能を保証するクラウドのインフラとして提供することを目的としたものである。GridARS は、主に資源の組み合わせを適切に選択して確保する資源管理システム (RMS) [2], [3] と、確保した資源の利用状況を収集する分散モニタリングシステム (DMS) [4] からなる。RMS は、G-lambda プロジェクト [5] で規定されたネットワーク帯域を提供するためのウェブサービスインタフェース GNS-WSI [1] v. 2 を採用し、WSRF (Web Services Resource Framework) [6] の参照実装である Globus Toolkit 4.0 (GT4) を用いて開発され、国際回線を利用した実証実験で用いられている [1], [7]。しかしながら、GT4 はセキュリティや WSRF の機能のサポートがあるものの他のウェブサービスの実装とインターオペラビリティがない。また、計算機などネットワーク以外の資源にも適用可能なインタフェース GNS-WSI v. 3 (GNS-WSI3) [5] が新たに規定されたため、GNS-WSI3 を採用した RMS の開発が必要である、RMS と DMS の実環境での連携はまだ行われていない、という課題もある。

本研究では、GNS-WSI3 を採用した GridARS RMS および DMS の GT4 版と PureWS 版を開発した。GT4 版では GT4、PureWS 版では Apache CXF [9] および Jetty [8] を用いた。開発した GNS-WSI3 ベースの GridARS の有効性を示すため、計算機クラスタ環境に広域ネットワークを想定したエミュレーション環境を構築し、RMS および DMS の基本性能を調査した。実験から、RMS および DMS の各手続きが非常に高速に処理できることを示す。また、SC09 国際会議において PureWS 版 RMS と DMS の連携実験を行ったので報告する。

2. GridARS 資源管理フレームワーク

2.1 GridARS フレームワークの概要

我々は、予約を前提としたネットワークの帯域を含む多様な資源を管理・提供するためのフレームワーク GridARS を開発している。図 1 に GridARS の概要を示す。GridARS は主に資

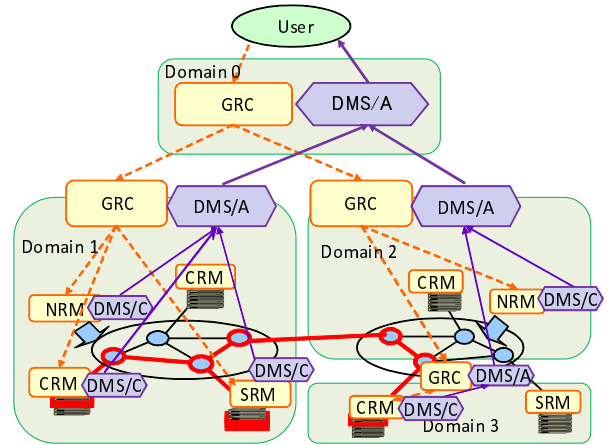


図 1 GridARS フレームワークの概要
Fig.1 Overview of the Gridars framework.

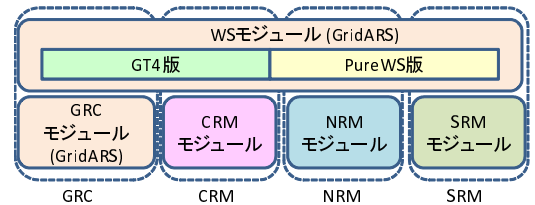


図 2 RMS のモジュール構成。
Fig. 2 RMS modules.

源管理システム (RMS) と分散モニタリングシステム (DMS) で構成される。図 1 の GRC, CRM, NRM, SRM は RMS のモジュールを表し、DMS/A および DMS/C は DMS のモジュールを表す。また、図中の Domain は個々の組織などを表し、4つのドメインがある場合を示している。各ドメインは、資源を直接または間接的に管理する。以下に RMS および DMS の概要を述べる。

2.2 資源管理システム RMS

RMS は、グローバル資源コーディネータ (GRC) と各資源を管理する資源マネージャ (RM) で構成される。CRM, NRM, SRM はそれぞれ計算機、ネットワーク、ストレージの RM を表している。GRC はユーザの資源要求を満たす資源を適切に選択し、RM と連携して選択した資源群を確保し、ユーザに提供する。RM は各資源を管理するものであり、バックエンドにバッチキューイングシステムのような既存の資源管理システムを用いている。

図 2 に、RMS のモジュール構成を示す。GridARS では、ウェブサービスインタフェースモジュールと、GRC/RM モジュールを提供する。GRC/RM モジュールは、GridARS の提供する Java の SPI (Service Provider Interface) を実装することで、資源提供者が自由に開発することもできる。GridARS の GRC では、資源の割り当てを最適化問題にモデル化した手法を用いており、ユーザの要求を満たす適切な資源の組み合わせを選択することができる [3]。

2.3 分散モニタリングシステム DMS

DMS は、RMS で確保された資源群の稼働時の資源利用状況

をモニタリング情報として資源を確保したユーザに提供する。この情報をもとに、ユーザは確保した資源の安定稼働の確認や、余剰資源の解放や高負荷資源の負荷分散のための再構成を行うことができる。GRC 及び各 RM は再構成を行うための手段を提供し、具体的な再構成方針はユーザが決定する。モニタリング情報は各ドメイン内で管理されており、集中 DB を持たない、どの情報を提供するかは各ドメインで決定できる、という特徴がある。

DMS は DMS/C (コレクタ) と DMS/A (アグリゲータ) で構成される。DMS/C では、実際に資源情報をモニタリングし、各ユーザに対してどの情報を提供するかを決定 (認可) するモジュールである。DMS/C では、提供する情報ごとにモニタリングエージェントを複数登録することができ、各エージェントが SNMP [10] や, Iperf [11], ping 等のツールを用いて実際の資源の利用状況を収集する。また、標準の認可手法である XACML [12] により、ドメインごとのポリシーに基づいて提供する情報をフィルタリングすることができる。

DMS/A は、RMS と連携して予約のツリーをたどり、関連する DMS/C のモニタリング情報を集めてユーザに提供するためのモジュールである。DMS がモニタリングを要求したユーザの分散する資源情報を取得するには、ユーザ確保した資源の予約 ID を用いて RMS の管理する資源予約情報を取得し、関連する DMS/A または DMS/C に問い合わせる。これを階層的に繰り返し、確保した全資源のモニタリング情報を収集することができる。

2.4 前バージョンの GridARS の課題

1. 節で述べたように、実証実験 [1], [7] で用いられた前バージョンの GridARS には、以下のような課題がある。

インターオペラビリティ 前バージョンは、Globus Toolkit 4 (GT4) を用いて開発されている。GT4 は、WSRF の参照実装であるとともに、GSI(Grid Security Infrastructure) と呼ばれるセキュリティ基盤をサポートしている。一方、GT4 は Apache AXIS [13] を拡張して実装されており、他のウェブサービスの実装との互換性がなく、ユーザコミュニティが限定されてしまう。GNS-WSI3 への対応 前バージョンの RMS は、NRM にネットワークの資源予約のためのウェブサービスインタフェース GNS-WSI v. 2(GNS-WSI2) を採用し、GRC や CRM に対しては、GNS-WSI2 の拡張を行ったものを用いていた。そのため、GRC や各 RM のインタフェース (WSDL) が統一されておらず、GRC や RM の構成が限定されていた。その後、ネットワーク以外の資源の予約も可能にした GNS-WSI v. 3(GNS-WSI3) が規定されたため、GNS-WSI3 を採用し、GRC や RM の構成の制約を解消するフレームワークが必要である。

RMS と DMS の連携 DMS はプロトタイプ実装のみであり、実際の資源や RMS との連携の動作確認が必要である。

3. GNS-WSI3 ベース GridARS の開発

本研究では、GNS-WSI3 の参照実装となる RMS と、その RMS と連携してモニタリングサービスを提供する DMS を開発した。GNS-WSI はインタフェースの規定のみで、その実装

表 1 GNS-WSI3 の主なオペレーション

Table 1 GNS-WSI3 operations.

オペレーション	機能	入力	出力
create	初期化	-	RsvEPR
reserve	資源予約要求	RsvEPR, 時刻, 資源	CmdEPR
modify/ modifyAll	一部 / 全資源の修正要求	RsvEPR, 時刻, 資源	CmdEPR
release/ releaseAll	一部 / 全資源の解放要求	RsvEPR, 資源 ID	CmdEPR
commit	コマンドの実行	CmdEPR	-
abort	コマンドの破棄	CmdEPR	-
getResource- Property	プロパティ値の取得	プロパティ名	プロパティ値

に何を用いるかは規定していないため、機能性とインターオペラビリティの双方を考慮して GT4 を用いた GT4 版と一般的なウェブサービスソフトウェアの一つである Apache CXF を用いた PureWS 版を開発した。

3.1 RMS のインタフェース: GNS-WSI3 の概要

GNS-WSI は、ネットワークの帯域予約のための共通ウェブサービスインタフェースを規定することを目的として G-lambda プロジェクトで提案された。GNS-WSI では、WSDL (Web Services Description Language) により、2 拠点を結ぶパスの帯域の予約、帯域や予約時刻の修正、および予約したパスの解放と、利用可能なパスに関する問い合わせのためのオペレーションを提供している。また、GNS-WSI v. 2 では、WSRF の仕様に基づくとともに、2 相コミットを採用して分散する複数 RM へのトランザクション処理を可能にした。

GNS-WSI v. 1 および v. 2 は、いずれもネットワーク資源の予約手続きを規定したものであるが、v. 3 では対象とする資源を抽象化することにより、多様な資源を扱うことが可能な予約インタフェースとした。すなわち、予約対象資源を抽象型 ReservationResources_Type として定義し、インタフェースの定義 (WSDL) では、ReservationResources_Type の予約、修正、解放のオペレーションを規定した。また、ネットワークや計算機、ストレージ等、予約対象となる個々の資源については、ReservationResources_Type を継承した Network/Compute/StorageResources_Type を別途定義し、各 GRC 及び RM で関連する資源に対する処理を実装するようにした。また、v. 2 以前では 1 度のオペレーションで扱えるのは 1 つの資源 (パス) のみだったのに対し、GNS-WSI3 では複数の資源を扱えるようにした。これらの改良により、GNS-WSI3 では、GRC や RM を自由に構成することができるようになった。

表 1 に GNS-WSI3 の主要なオペレーションを示す。図中の RsvEPR, CmdEPR は、初期化した予約手続きの ID, 各予約 / 修正 / 解放要求コマンドの ID を表す。表 1 のうち、getResourceProperty は WSRF で規定されているオペレーション、それ以外は GNS-WSI3 独自のオペレーションである。この他に、独自の例外処理を行うことを目的として、プロパティのうち予約

表 2 DMS の主なオペレーション

Table 2 DMS operations.

オペレーション	機能	入力	出力
create	初期化	RsvEPR	MonEPR
configure	収集情報, 時刻の指定	MonEPR, 時刻, 情報リスト	-
getInformation	指定した 情報の取得	MonEPR(, 時刻)	モニタリング 情報

処理状態 ReservationStatus 及びコマンド実行状況 Command-Status を取得するためのオペレーション (getReservationStatus , getCommandStatus) は別途定義している .

GNS-WSI はオペレーションのブロッキングを避けるためにポーリングベースで設計されているため, 通常の予約手続きをする場合には, ユーザは create (初期化), reserve (予約要求), getCommandStatus (予約手続きの状況の確認), commit (予約の確定) の手続きが必要となる .

3.2 DMS のインタフェース

DMS は, DMS/A と DMS/C で共通のインタフェースを利用しており, DMS/A および DMS/C も自由に階層化して構成できる .

表 2 に DMS の主なオペレーションを示す . create はモニタリング要求を受け付けるオペレーションであり, RMS での予約に利用した RsvEPR を引数として渡し, このモニタリング情報の ID となる MonEPR をユーザに返す . 各 EPR は EndpointReferenceType で表され, EPR 内に EPR を発行したサービスのエンドポイント情報が含まれている . よって, DMS はそのユーザが予約した際に利用した GRC および RMS の所在を知ることができ, 予約 ID である RsvEPR によって関連する予約情報も取得できるため, それを階層的に繰り返すことで複数のドメインを跨る場合でも予約のツリーを辿ることができる .

configure はどの情報をいつから集めるのかを指定するコマンドである . DMS では, configure 要求の際に初めて予約ツリーを辿る作業を行い, 以降は下位の DMS と直接やりとりする . DMS のオペレーションは基本的にブロッキングで処理されるが, 予約ツリーを辿る作業には時間がかかるため, DMS/A の configure のみをノンブロッキングで処理することにした . また, getInformation は指定した情報を取得する際に用いる .

3.3 GT4 版と PureWS 版の実装

GT4 版では Globus Toolkit 4.2 を用いて実装した . GT4 は, WSRF の参照実装であり, WSRF の仕様で規定されているオペレーションが提供されている . 例えば, getResourceProperty や手続きの Lifetime および Notification に関するオペレーションやメカニズムがある . また, 認証, 認可, 権限委譲の仕組みを提供する GSI や SSL もサポートしている . よって, GT4 版では予約に必要なオペレーションおよび型の定義のみを行い, 定義したウェブサービスを実装するだけで, WSRF の機能および GSI を利用することができる .

しかしながら, GT4 は通常のウェブサービスとのインターオペラビリティがなく利用者のコミュニティが小さい, バグへの

対応も Globus プロジェクト依存となってしまうという問題もある . 例えば, GRC では複数 RM への同時アクセスを実現するためにマルチスレッドを用いているが, GT4.0 および GT4.2 で実装した GRC では初回のマルチスレッドでのアクセス時にたびたび例外が発生してしまい, その原因が特定できていない . 現在の GT4 版では, この部分をシングルスレッド化することにより対応したが, GRC において複数 RM へのアクセスをシリアルに行くと遅延が大きくなってしまうため, バグの修正が待たれる .

一方, PureWS 版ではウェブサービスコンテナとして Apache CXF 2.2.3, ウェブサーバとして Jetty 7.0.0 を用いた . Apache CXF は一般に用いられているウェブサービスソフトウェアとインターオペラビリティがあり, Jetty は Java で実装された軽量のウェブサーバとして知られている . しかしながら, WSRF の仕様で規定されているオペレーションや, GSI に代わるセキュリティパッケージは提供されていない . よって, 現 PureWS 版では getResourceProperty のみを実装し, 他の機能は今後順次実装していくこととした . また, セキュリティ機能は未対応であり, 認証をサポートする OpenSSO [14] などのセキュリティパッケージの適用を検討している .

RMS では, GRC/RM モジュールも GT4 版と PureWS 版を開発した . GridARS は WS モジュールと GRC/RM モジュールからなり, 理論的には GT4 版と PureWS 版の WS モジュールのみを開発すれば, 下位の GRC/RM モジュールは同じものを利用することができると思われる . しかしながら, WSDL から Java のオブジェクトに変換するツールが GT4 と CXF で異なり, GT4 版では XML スキーマで “XXX.Type” と定義された型が “XXX.Type” という名の Java のクラスに変換されるのに対し, PureWS 版では “XXXType” となり, コードの流用ができない . よって, 同様の機能をもつ GRC/RM モジュールをそれぞれ用意した . DMS においても, GT4 版および PureWS 版の WS モジュールおよび DMS/A, DMS/C モジュールを開発した .

4. 実 験

新たに開発した GridARS RMS および DMS の有効性を示すため, 計算機クラスタ内に広域環境をエミュレートする環境を構築し, RMS および DMS の基本性能を調査するとともに, SC09 において RMS と DMS を連携させる実証実験を行った .

4.1 エミュレーション環境

実験では, 広域環境で複数ドメインから資源が提供される, 図 3 のような環境を想定した . 図 3 では, 4 つの計算機サイトと 2 つのネットワークドメインからなり, それぞれ 4 つの CRM と 2 つの NRM が資源を管理する . また GRC を 1 つ用意し, GRC がこれら 6 つの RM と連携してユーザの要求を満たす資源を選択・確保する . 遅延は異なるネットワークドメイン間では 100 msec とし, ドメイン内の異なるサイト間は 10 msec とした . また, ドメイン間の接続は 1Gbps×4 本とし, その他は 1Gbps×2 本とした .

この環境を想定したエミュレーション環境をクラスタ内に構

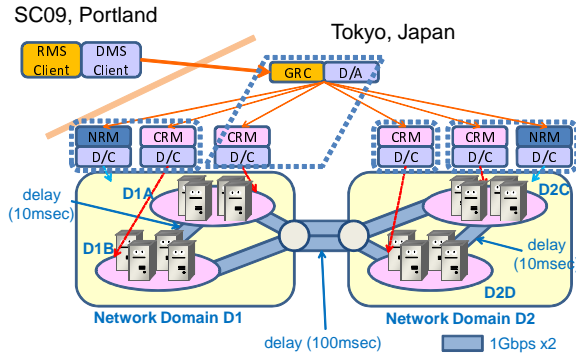


図 3 想定する広域環境

Fig. 3 Assumed distributed resource environment.

表 3 PureWS 版と GT4 版のオペレーション所要時間の比較

Table 3 The comparison of operation elapsed times using the PureWS version versus the GT version.

オペレーション	オペレーション処理時間 [msec]		
	PureWS 版	GT4 版	GT4/GSI 版
create	14.9	15.4	132.2
reserve	24.0	40.5	165.6
getCS	13.6	13.8	145.9
commit	13.8	15.2	157.1

築した。計算サイト間のネットワークの遅延やスループットの制御には、ハードウェアネットワークエミュレータ GNET-1 [15] を用いた。D1A/D1B/D2C/D2D には、それぞれヘッドノード 1 台、計算ノード 4 台からなる計算機サイトを構築し、サイト内の計算機管理にはバッチキューイングシステムの Sun Grid Engine [16] と予約機能を付加するプラグインスケジューラ PluS [17] を用いた。GRC 及び 6 つの RM は、図 3 の破線で表したように 4 台のヘッドノードに配備した。計 20 台の計算機の性能はすべて同じで、CPU Intel Core2 Quad Q9550(2.83GHz)、OS CentOS 5.3、kernel 2.6.18 x86_64、メモリ 4GB となっている。

4.2 RMS の基本性能

RMS の基本性能の評価では、PureWS 版と GT4 版において 3.1 節で示した予約手続きに必要なオペレーションの処理時間の比較と、PureWS 版において GRC と複数 RM と連携する際の処理時間を示す。

表 3 に、PureWS 版と GT4 版のクライアント-RM 間のオペレーションの処理時間を msec で示す。この実験では、クライアントと RM は同じ計算機上に設定した。PureWS 版ではセキュリティなし、GT4 版では GSI を利用しない場合と GSI を用いた場合 (GT4/GSI 版) の結果を示す。getCS は getCommandStatus の結果を表す。

PureWS 版と GT4 版で比較すると、reserve の手続きでは GT4 版の方が処理時間が長くなるものの、他のオペレーションではほぼ同程度であることが分かる。また、いずれもオペレーションの所要時間は 50msec に満たず、非常に高速であることが示された。GT4 版と GT4/GSI 版の比較では、GT4/GSI 版

表 4 複数ドメイン環境での PureWS 版 RMS の処理時間

Table 4 The PureWS version RMS operation elapsed times in multi-domain environment.

オペレーション	オペレーション処理時間 [msec]					
	RMx1	RMx2	RMx3	RMx4	RMx5	RMx6
create	14.7	13.3	12.7	12.4	12.3	13.4
reserve	22.9	21.2	19.8	20.1	20.2	21.9
getCS	12.2	11.0	10.1	10.0	10.3	10.7
commit	11.9	10.2	9.6	10.1	11.2	9.8

では認証、SSL 通信を行っているものの、そのオーバーヘッドはいずれのオペレーションにおいても 150msec 以下であった。

次に、表 4 に PureWS 版において、GRC と複数 RM 間で連携する際に要する時間を比較する。この実験では、reserve 以降のオペレーション (create 以外) においてクライアントが GRC に要求を送り、GRC がその要求に関連する 1~6 つの RM に対して要求を送る際に要する時間を調査する。クライアント及び GRC は同一マシンに設定し、6 つの RM は図 3 の 4 つのサイトのヘッドノードとなる 4 つのマシンに設定した。表 4 の結果から、create 以外のオペレーションの処理時間も RM の台数によらずほぼ同程度であることが分かる。これは、GRC において複数 RM に対する要求をマルチスレッド化して同時に処理しているためである。

4.3 PureWS 版での DMS の基本性能

DMS の基本性能の評価では、PureWS 版を用いて GRC と複数 RM と DMS が連携する際の処理時間を示す。表 5 に DMS の主なオペレーションの処理時間を示す。C-DMS/C は、RM および DMS/C のみを用いてクライアントが DMS/C にモニタリング要求した際の結果であり、クライアント、RM、DMS/C が 1 つの計算機上で動作している。DMS/Cx1 ~ DMS/Cx6 は、GRC、DMS/A と 1~6 つの RM、DMS/C を用い、クライアントが DMS/A を介して DMS/C に対してモニタリング要求した際の結果を表す。

create は受け付けた DMS での処理のみ、configure 以降は階層的な処理が行われる。また、3.2 節で述べたように DMS/A の configure 処理はノンブロッキングで行っているため、configure(bg) はバックエンドで実際に階層的な configure 処理が終了するまでに要した時間を表す。

表 5 から、RMS 同様、create、getInformation とともに 15msec 以下、configure も 100msec 程度で処理できており、非常に高速であることが分かる。一方、configure(bg) では予約ツリーを辿る作業を行っているため、1sec 以上かかっていた。ただし、バックグラウンドで処理されているのでユーザに対する応答時間には影響がない。また、RMS 同様 DMS/C の台数によらず、各オペレーションが処理できることが示された。

4.4 SC09 での実証実験

GridARS の RMS と DMS が連携し、資源の確保および確保した資源のモニタリングができることを示すため、2009 年 11 月に米国ポートランドで開催された SC09 国際会議で実証実験を行った。実験では PureWS 版 RMS と DMS を用い、図 3 の

表 5 複数ドメイン環境での PureWS 版 DMS の処理時間

Table 5 The PureWS version DMS operation elapsed times in multi-domain environment.

オペレーション	オペレーション処理時間 [msec]						
	C-DMS/C	DMS/Cx1	DMS/Cx2	DMS/Cx3	DMS/Cx4	DMS/Cx5	DMS/Cx6
create	11.4	14.3	13.4	13.8	13.4	13.9	13.1
configure	38.5	68.1	85.7	115.7	116.2	119.9	116.1
configure(bg)	–	1078.1	1095.3	1126.9	1128.2	1132.6	1125.5
getInformation	3.1	11.1	11.3	11.6	13.3	10.7	13.0

ようにクライアントを SC09 の会場の計算機に設定し、広域環境には評価実験で用いたエミュレーション環境を利用した。

実験では、まず RMS のコマンドラインインタフェース (CLI) から、抽象的な資源要求を入力し、GRC のスケジューラで適切な計算機サイトおよびネットワークのパスを選定し、関連する RM に対して資源確保手続きを行って必要な資源を確保した。次に、DMS の CLI から RMS から取得した RsvEPR を用いてモニタリング要求を送り、関連する RMS および DMS が連携してモニタリング情報収集エージェントを起動しておく。その後、予約時刻に getInformation コマンドで各 RM で収集された情報が DMS/A を介して取得できることを確認した。

5. 関連研究

ネットワーク資源を含む多様な資源を扱う資源管理システムが複数開発されている [18]~[20]。VIOLA [18] は、インタフェースに WS-Agreement [21] を採用して GT4 を用いて実装している。ORCA (Open Resource Control Architecture) [19] および Nortel 社により開発された DRAC (Dynamic Resource Allocation Controller) [20] は、ネットワーク機器のマネジメントまでを対象としている。

モニタリングシステムでは、perfSONAR [22] がいくつかのネットワークテストベッドにおいて既にデプロイされている。perfSONAR では、分散するネットワークルータから得られる情報をウェブサービスを介して収集し、クライアントに提供する。perfSONAR は、ネットワーク資源専用のモニタリングツールである点、モニタリング情報を集中 DB に格納する点で DMS とは異なる。

6. まとめ

本研究では、新たに規定された GNS-WSI3 を採用した GridARS RMS および DMS の GT4 版と PureWS 版を開発した。開発した RMS および DMS を用いて広域ネットワークを想定したエミュレーション環境で各オペレーションの所要時間を調査し、各オペレーションの処理時間が非常に短いことを示した。また、RM が複数ある環境においても、RM 数に依存することなく高速に処理できることを示した。また、SC09 国際会議における PureWS 版 RMS と DMS の連携実験について報告した。

今後は、PureWS 版のセキュリティ機能の追加を行うとともに、実際の広域分散環境での実験を行う。

謝 辞

本研究の一部は、情報通信研究機構 (NICT) の委託研究「ダイナミックネットワーク技術の研究開発」の助成を受けたものである。

文 献

- [1] Takefusa, A. et al.: G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS, *Future Generation Computing Systems*, Vol. 22(2006), pp. 868–875 (2006).
- [2] Takefusa, A. et al.: GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources, *Proc. 13th Workshop on Job Scheduling Strategies for Parallel Processing* (2007).
- [3] 竹房ほか: 性能を保証する分散実行環境のためのオンラインコアレーション手法, コンピュータシステム・シンポジウム論文集, pp. 51–58 (2009).
- [4] Takefusa, A. et al.: Design of a Domain Authorization-based Hierarchical Distributed Resource Monitoring System in cooperation with Resource Reservation, *Proc. HPC Asia 2009*, pp. 77–84 (2009).
- [5] G-lambda: <http://www.g-lambda.net/>.
- [6] S. Graham et al.(ed.): *Web Services Resource 1.2 (WS-Resource)*, OASIS (2006).
- [7] Thorpe, S. et al.: G-lambda and EnLIGHTened: Wrapped In Middleware Co-allocating Compute and Network Resources Across Japan and the US, *Proc. GridNets2007* (2007).
- [8] Jetty: <http://www.eclipse.org/jetty/>.
- [9] Apache CXF: <http://cxf.apache.org/>.
- [10] J.D. Case and M. Fedor and M.L. Schoffstall and C. Davin: Simple Network Management Protocol(SNMP) (1990).
- [11] Iperf: <http://sourceforge.net/projects/iperf/>.
- [12] T. Moses(ed.): *eXtensible Access Control Markup Language (XACML) Version 2.0*, OASIS (2005).
- [13] Apache Axis: <http://ws.apache.org/axis/>.
- [14] OpenSSO: <http://planets.sun.com/OpenSSO/>.
- [15] Kodama, Y. et al.: GNET-1: Gigabit Ethernet Network Testbed, *Proc. IEEE International Conference on Cluster Computing Cluster2004* (2004).
- [16] Sun Grid Engine: <http://gridengine.sunsource.net/>.
- [17] Nakada, H. et al.: An Advance Reservation-based Computation Resource Manager for Global Scheduling, *Proc. the 3rd International Workshop on Grid Computing and Applications* (2007).
- [18] Barz, C.: Dynamic allocation of network resources in VIOLA, *VIOLA workshop* (2005).
- [19] ORCA: <http://nicl.cod.cs.duke.edu/orca/>.
- [20] DRAC: <http://www.nortel.com/drac/>.
- [21] Andrieux, A. et al.: Web Services Agreement Specification (WS-Agreement), *OGF Document Series*, GFD.107 (2007).
- [22] PerfSONAR: <http://www.perfsonar.net/>.