

マルチドメインクラウド資源管理フレームワークの実証実験

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 〒 305-8568 茨城県つくば市

E-mail:

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

あらまし マルチドメインクラウドの資源管理フレームワークとして、資源管理システム、アプリケーション実行管理システム、分散モニタリングシステムからなる GridARS を開発している。本研究では、既存システムを拡張し、相互運用性、機能性の高い GridARS システムを開発し、大規模環境でその有用性を評価した。また、GLIF2010 および SC10 において実証実験を行い、GridARS を用いてマルチドメインクラウド上に自動的にユーザの要求を満たす仮想インフラを構築し、そのモニタリングを行うことに成功した。

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, 実証実験

Demonstration of Resource Management Framework for Multi-domain Cloud

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) Tsukuba, Ibaraki 305-8568, Japan

E-mail:

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

Abstract We have been developing the GridARS resource management framework for multi-domain cloud. GridARS consists of resource management, distributed monitoring, and application execution management systems. In this work, we extend GridARS to be an interoperable and high-functional system and investigate the scalability of the developed system. At GLIF2010 and SC10, we succeeded in demonstration that GridARS automatically constructed a user requested virtual infrastructure, distributed over multi-domain cloud, and provide monitoring information of the infrastructures.

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

1. はじめに

クラウドとは、一般にデータセンターで管理されている計算機、ストレージやその上に構築される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。しかしながら、クラウドサービスで必要とされるデータ、アプリケーションライブラリ、ハードウェア等の資源は、必ずしも一つのデータセンター内や資源プロバイダ(ドメイン)によって管理されているわけではない。よって、付加価値の高いサービスを実現するには、性能が保証された各データセンター内の

資源とデータセンター間のネットワークを確保し、その上に仮想インフラを構築する技術が必要となる。

我々は、資源管理フレームワーク GridARS を開発している[1],[2]。マルチドメインクラウド上に仮想インフラを構築・提供するため、GridARS は資源管理システム(RMS)、アプリケーション実行管理システム(AEM)、分散モニタリングシステム(DMS)で構成されている[1],[3],[4]。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等の資源を提供するためのウェブサービスインタフェース GNS-WSI v. 3 (GNS-WSI3)[5]に基づき、ユーザの要求に応じてマルチドメ

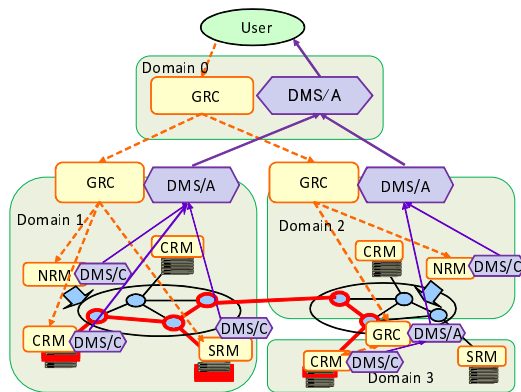


図 1 GridARS フレームワークの概要

Fig. 1 Overview of the Gridars framework.

インクラウドの資源を予約を前提として確保する。AEM は RMS が確保した資源上にアプリケーションの実行環境である仮想インフラを構築する。DMS は構築された仮想インフラの利用状況を収集して仮想インフラのユーザに提供する。

我々は、WSRF (Web Services Resource Framework) [6] の参照実装である Globus Toolkit 4.0 (GT4) を用いて開発した GT4 版と、Apache CXF [7] および Jetty [8] を用いて実装した PureWS 版の RMS および DMS を開発し、その基本性能を示した [2]。しかしながら、GT4 版は相互運用性が低い、PureWS 版では機能性が低いという問題があり、大規模環境での検証も不十分であった。また、AEM との連携はとれておらず、マルチドメインクラスタ上の自動的な仮想インフラ構築は実現できていなかった。

本研究では、PureWS 版の機能を拡張することで、相互運用性及び機能性の高いマルチドメインクラウド資源管理フレームワークを開発した。また、AEM と連携して自動的な仮想インフラの構築、モニタリングを行えるようにした。PureWS 版の拡張では、永続化機能、セキュリティ機能、モニタリング可視化機能を実装した。AEM との連携では、IP アドレスや VLAN タグ ID の割り当て、仮想インフラからのモニタリング情報を取得する機能を実装した。

評価では、PureWS 版を用いてクラスタ環境で 10 ドメイン 20RM に対する予約処理およびモニタリング処理を行った。予約処理は、SSL 通信をする場合でも 2.3 秒で、モニタリング処理は 10 ドメイン × 1001 資源のモニタリング情報取得を 1.8 秒で行えることを示す。また、国際会議 GLIF2010 および SC10 において 2 つのネットワークドメイン、4 つの計算機ドメインの環境で RMS、AEM、DMS の連携実験を行い、仮想インフラの自動的な構築およびモニタリングに成功したので報告する。

2. GridARS 資源管理フレームワーク

GridARS は予約を前提として計算機やストレージ、ネットワークの帯域等、多様な資源を確保・提供するための資源管理フレームワークである。図 1 にその概要を示す。GridARS は資源管理システム (RMS)、アプリケーション実行管理システム (AEM)、分散モニタリングシステム (DMS) で構成される。図

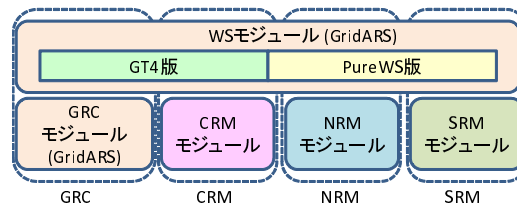


図 2 RMS のモジュール構成。

Fig. 2 RMS modules.

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。AEM は、CRM 内にある。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、AEM、DMS の概要を述べる。

2.1 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。図 2 に、RMS のプログラム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 モジュールでは、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する。RM モジュールは、既存資源管理システムのためのラップであり、GNS-WSI3 のインタフェースを提供する RM モジュールを介して各資源を実際に管理する。既存の資源管理システムには、予約機能を持つバッチキューイングシステムなどを用いる。

2.2 アプリケーション実行管理システム AEM

AEM は、RMS で確保された資源上に仮想インフラを構築する [3]。AEM は RMS と連携して、予約時刻に確保された計算機の IP アドレス、拠点間のパスの VLAN タグ ID 等を収集し、各計算機およびネットワークを仮想化する。仮想化では、Linux コンテナ [9] の起動、SSH の key 配備、ファイルシステムの設定、VLAN の設定等を行い、性能が保証されたセキュアな仮想インフラを構築する。構築した仮想インフラでは、一般のクラスタ向けに書かれたプログラムを改変することなく実行することができる。また、事前に指定されたアプリケーションジョブの自動実行も可能である。

2.3 分散モニタリングシステム DMS

DMS は、仮想インフラの利用状況を RMS、AEM と連携してモニタリングし、仮想インフラを要求したユーザに提供する。この情報をもとに、ユーザは資源の所在やドメインに配慮することなくアプリケーションの安定稼働の確認や、余剰資源の解放、負荷分散等を行うことができる。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。DMS/Aは、RMSと連携して予約IDをキーとして階層的な予約ツリーをたどり、関連するDMS/Cのモニタリング情報を収集してユーザに提供する。

3. PureWS版GridARSの機能拡張

RMSおよびDMSには、WSRFの参照実装であるGT4で開発したGT4版と、Apache CXFおよびJettyで開発したPureWS版がある[2]。GT4版はGT4のセキュリティやWSRF機能が利用できるものの、他のウェブサービス実装と相互運用性が低いという問題があった。一方、PureWS版は一般的なウェブサービス実装であり、相互運用性は高いものの機能が不十分であった。本研究では、PureWS版に対して永続化、セキュリティ、モニタリング可視化の機能を実装し、相互運用性および機能性を備えた資源管理フレームワークを開発した。

3.1 永続化

永続化をサポートしないと、予約に関するプロパティ情報は障害等でウェブサーバを再起動すると失われてしまう。また、完全に障害復旧には、データの復旧だけでなく、各処理の復旧まで行う必要がある。よって、RMSに対してJDK6の標準RDBであるDerbyを用いてプロパティ情報をDBに格納して永続化を実現するとともに、再起動時はDBから情報を取り出して予約処理を復旧するようにした。具体的には、各予約に対して“予約ID”、“予約ユーザ名”、“関連RM予約ID”をDBに格納しておき、再起動時に次の手順で復旧する。(1)全予約の予約ID、予約ユーザ名、関連RM予約IDをDBから取得する。(2)各関連RMに対してプロパティ情報を再取得する。(3)必要な予約情報をメモリ上の予約管理ハッシュに格納したのち、各予約ステータスごとの処理を復旧させる。

3.2 セキュリティ

セキュリティ機能をサポートする、JettyのSSLを用いたセキュア通信機能、認証機能をRMS、DMSで利用できるようにした。セキュア通信では、PureWS版GRCおよびRM(サーバ)側では(s1)鍵ファイルを作成し、指定ディレクトリに格納しておく、(s2)セキュア通信を行うURLをプロパティファイルで設定する、(s3)関連ライブラリ、プロパティファイルなどをWAR(Web Application Archive)としてまとめ、Jetty(ウェブサーバ)に配備する、(s4)JettyでSSLのポート番号、鍵ファイルのパス、パスフレーズの設定を有効にする。クライアント側では、(c1)プロパティファイルで鍵ファイル名、パスフレーズを設定して、(c2)クライアント用Jarファイルを作成し、(c3)予約実行時にSSLを利用するためのオプションを指定する。これにより、SSLを使うかどうか指定することができる。

ユーザ認証は、ダイジェスト認証、クライアント証明書による認証が考えられるが、まずはベーシック認証を利用できるようにした。サーバ側では、予めユーザ名、パスワード、ロール名を設定しておき、Jettyにデプロイする。クライアント側では、実行時にユーザ名、パスワードを指定する。

3.3 モニタリング可視化

DMSでは、仮想インフラのモニタリング情報を収集するもの

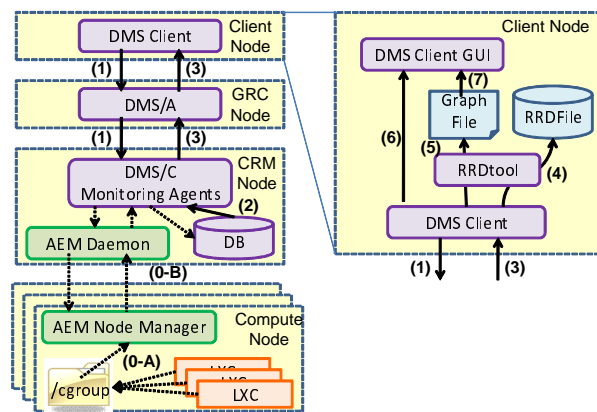


図3 モニタリング情報の可視化とAEMとの連携。

Fig. 3 Monitoring visualization and cooperation with AEM.

の、その可視化ツールは提供していなかったため、RRDtool[10]を用いたモニタリング可視化機能を実装した。RRDtoolは、時系列データをグラフ化するためのツールである。

図3の右側にRRDtoolを用いたモニタリング可視化の概要を示す。図中のDMS Clientは通常のDMSクライアントであり、(1)の処理でモニタリング情報を要求し、(3)で要求した情報を取得する。モニタリング可視化機能として、(4)~(7)の処理を新たに実装した。(4)では、(3)で得た情報をもとにRRDtoolを操作し、モニタリング情報ごとに“時刻”と“値”の時系列データとしてRRD(ラウンドロビンDB)ファイルに格納する。(5)では、RRDtoolを操作してモニタリング情報ごとのグラフファイルを作成する。(6)でGUIを提供するDMS Client GUIに対して表示するグラフの更新を指示し、(7)でグラフファイルを再読み込みすることによりグラフ表示を更新する。

4. AEMとの連携

SC09の実験[2]では、AEMとの連携は実現できていなかったため、本研究では、RMS、AEM、DMSを連携させて仮想インフラの構築およびモニタリングの自動化を実現する。

4.1 RMSとAEMの資源管理連携

AEMは、ユーザに対して1つの計算機クラスタのように仮想インフラを提供する必要がある。そのため、AEMでは異なるドメインの計算機に対して同じセグメントのIPアドレスを設定し、各拠点間はVLANで接続する[3]。この際、IPアドレスとVLANタグIDの調整が必要となる。

よって、RMSではIPアドレスやVLANタグIDも予約する資源の1つと考え、適切なものを選択するようにした。具体的には、GRCが各CRMやNRMから利用可能なIPアドレスレンジとVLANタグIDの情報を取得し、資源予約の際にIPアドレスレンジおよびVLANタグIDを合わせて予約する。

IPアドレスは、予約時刻に各CRMでIPアドレスレンジから最終的なIPアドレスが決定される。このような情報を各ドメインのAEM間で共有する必要があるため、RMSに予約時刻に動的情報を各ドメインから収集、分配する処理のトリガ部分も実装した。

4.2 AEM と DMS のモニタリング連携

構築された仮想インフラは物理計算資源からアイソレートされているため、Ganglia [11] 等のモニタリングツールではその利用状況は取得できない。よって、AEM と連携して仮想インフラの計算資源モニタリング情報を取得するようにした。

図 3 の左側に AEM と DMS のモニタリング連携の概要を示す。DMS/C Monitoring Agents (MA) は DMS/C でモニタリング情報収集を司るエージェントで、その 1 つとして計算資源情報を収集するエージェントがある。AEM Daemon (AEMD) は各計算サイト(ドメイン)のヘッドノードで稼働し、CRM と連携して AEM Node Manager (AEMNM) 群を管理する。AEMNM が各計算ノードで仮想インフラ構築処理を行っている。

モニタリングの手順は次のようになる。(0-A) AEMNM が仮想インフラを構成する Linux コンテナの cgroup 管理ディレクトリに書き込まれた情報を読むことで、各計算資源の情報を定期的に収集する。(0-B) MA が AEMD を介して定期的に情報を収集する。収集した情報は DMS/C が DB に格納する。(1)DMS Client がモニタリング情報を要求すると、(2)DMS/C は予め収集されていた情報を DB から読み出し、(3) その情報を DMS Client に返す。

5. 大規模環境を想定した評価

マルチドメインクラウド環境での GridARS の有効性を調査するため、改良した PureWS 版 GridARS を用いて 10 ドメイン 20RM に対して 1001 資源の予約およびモニタリングを行う際の処理時間を計算機クラスタ環境で評価した。

5.1 評価環境

10 ドメイン 20RM を用いた評価を行うため、AIST Green Cloud Server (AGC) の 1 シャーシ上に 1 つの GRC と 10 のドメインを用意し、1 ドメインに NRM および CRM を 1 つずつ用意した。また、DMS は GRC に DMS/A、各ドメインに 1 つずつ DMS/C を配備した。AGC は 8 シャーシからなり、1 シャーシは 16 台のブレードサーバ Dell PowerEdge M610 (クアッドコア Intel Xeon E5540×2 (2.53GHz, 8MB キャッシュ)、48GB メモリ, CentOS5.4) で構成されている。シャーシ内ネットワークは 10Gb Ethernet スイッチで接続されている。

評価では、ユーザが GRC に要求して GRC が 20 の RM に対して予約要求を送信する場合、ユーザが DMS/A に要求して DMS/A から 10 の DMS/C に対してモニタリング要求を送る場合の所要時間をそれぞれ測定した。予約処理では SSL 通信の有無、モニタリング処理では SSL 通信の有無およびデータ圧縮の有無による比較を行う。各ケースにおいて 50 回の実行時間の平均値を示す。

5.2 RMS の評価結果

図 4, 5 に各予約手続きの処理時間を示す。横軸はドメイン数を表す。実験では、CRM に対して 1000CPU の予約、NRM に対して 1 パスの予約を行っている。予約手続きは、create, reserve, getCommandStatus(getCS), commit オペレーションの順に行われ、それぞれ予約手続きの初期化、予約要求の送信、予約処理状態の確認、予約の確定処理を表す [2]。getCS は予約

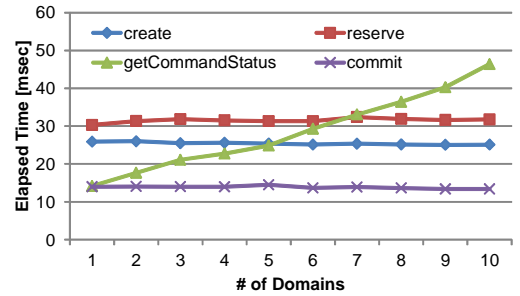


図 4 各予約手続きの処理時間 (SSL なし)。

Fig. 4 Elapsed times of reservation operations w/o SSL.

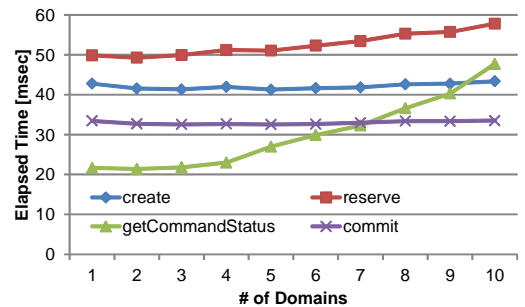


図 5 各予約手続きの処理時間 (SSL あり)。

Fig. 5 Elapsed times of reservation operations w/ SSL.

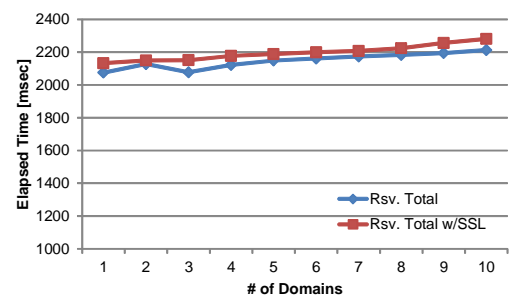


図 6 総予約処理時間の比較。

Fig. 6 Comparison of total reservation times.

準備ができるまで複数回実行されるが、ここでは 1 回の平均応答時間を表す。図 4 は SSL 通信を行わない場合、図 5 は SSL 通信を行った場合の結果を示す。

図 4 では、getCS 以外の処理時間がドメイン数によらずほぼ一定で、13.6-32.4[msec] となっていた。一方、getCS はドメイン数が増えるにつれ処理時間が増加した。これは、reserve が実行されると GRC がマルチスレッドで配下の RM に対して create や reserve の処理を始めるため、ドメイン数が増えるにつれて GRC の負荷が高くなり、直後に実行される getCS の処理時間が長くなってしまったためである。図 5 でも、同様の傾向が見られた。また、SSL 通信の有無で比較すると SSL 通信有の方が 20[msec] 程度処理時間が長くなっていった。

図 6 に総予約処理時間を SSL 通信の有無で比較した結果を示す。予約処理では、全 RM が予約可能状態になるまでクライアント-GRC 間、GRC-RM 間で getCS を指定した間隔で複

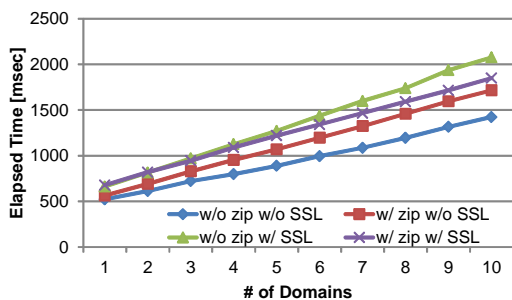


図 7 getInformation 処理時間の比較 .

Fig.7 Comparison of the getInformation operation elapsed times.

数回実行する．実験ではそれぞれ 1 秒と設定しており，各 RM での予約処理が 1 秒以内の場合で最大 2 秒がインターバルに費やされる．グラフから，総予約時間はインターバルを含めても SSL の有無やドメイン数によらず 10 ドメイン資源予約が 2.3 秒以内と高速に処理できることが示せた．

5.3 DMS の評価結果

モニタリングでは，モニタリング初期化手続きと仮想インフラの利用状況を取得するモニタリング手続きがある．初期化手続きでは，予約ツリーを辿る作業以外は RMS 同様数十 msec で処理可能であり，予約ツリーを辿る作業はバックグラウンドで処理されるのでユーザに対する応答時間に影響しない [2]．よって，評価ではモニタリング手続きに関する比較を行う．

モニタリング手続きは，getInformation オペレーションで行われる．クライアントが getInformation を呼び出すと，DMS/A が関連する DMS/C からモニタリング情報を収集して提供する．この際，マルチドメインから大規模な仮想インフラを構築した場合にモニタリング情報が膨大になり，getInformation の応答時間が長くなることが懸念される．よって，SSL の有無だけでなく，圧縮の有無による応答時間の影響を調査する．圧縮は，CXF の提供する機能を利用して送信前に圧縮，受信後に解凍する．モニタリング情報は，各ドメインから 1000 台分の計算機の情報と 1 つのパスの情報を収集し，最大 10010 資源の情報（非圧縮時は約 15MB，圧縮時は 500KB）を送信した．

図 7 に，応答時間の比較結果を示す．図中の zip は圧縮，SSL は SSL を表し，w/および w/o はその有無を表す．SSL がない場合で比較すると，圧縮しない方が応答時間が短いことが分かる．これは，通信量を小さくしたことによる通信時間の削減より，圧縮・解凍にかかるオーバーヘッドの方が大きいためである．一方，SSL 通信を行う場合では，圧縮通信を行った方が高速に処理できる．これは，圧縮のオーバーヘッドより暗号化するデータの縮小効果の方が大きいためで，SSL 通信では圧縮が非常に有効であることが分かる．また，ドメイン数が増えるにつれて総受信量が増えるため応答時間は長くなるが，SSL を用いるケースでも圧縮機能を用いれば 10010 資源のモニタリングが 1.8 秒程度で行えることが示せた．

6. 実証実験

国際会議 GLIF2010 および SC10 において RMS, AEM,

DMS の連携実験を行った．

6.1 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 8 のような環境を想定した．図 8 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 と連携して適切な資源を選択・確保するようにした．想定した環境はクラスタ内に構築することにした．サイト間の通信遅延の制御には，ハードウェアネットワークエミュレータ GtrcNET-1 [12] を用いた．GtrcNet-1 では，VLAN ごとのネットワークのモニタリングも行った．各計算サイトは，ヘッドノード 1 台と計算ノード 4 台で構成され，それぞれバッチキューイングシステムの Sun Grid Engine [13] と予約機能を付加するプラグインスケジューラ PluS [14] を用いた．GRC 及び 6 つの RM は，図 8 の破線で表したように 4 台のヘッドノードに配備した．

6.2 実証実験結果

実証実験では，ポータルからユーザの要求を入力し，ユーザの要求に応じて GridARS が自動的にマルチドメインクラウドの資源を確保し，予約時刻に仮想インフラを構築し，そのモニタリング情報を提供することに成功した．

図 9 にポータル画面を示す．ポータル画面では，計算サイト数，CPU 数，サイト間の帯域からなる資源要求と，デッドラインと予約時間からなる時間要求の入力のみで，計算機とその間のネットワークが自動的に割り当てられたことを示している．

図 10，図 11 に予約時刻における管理者およびユーザ用のモニタリング結果を示す．いずれも左上は現在の資源の確保状況を表し，その下は計算機およびネットワーク帯域の予約表を示す．図 10 では，赤で示された他のユーザの予約状況が出力されているのに対し，図 11 では黄色の予約したユーザの情報のみ出力されているのが分かる．これにより，DMS で予約したユーザの情報のみをフィルタして提供できることが分かる．さらに，図 11 の下部分には構築された仮想インフラを構成する計算機の負荷やメモリ使用量，ネットワークの使用帯域と累積転送量が表示されている．仮想インフラでは，Top500 で知られる HPL ベンチマークプログラムを仮想インフラ上で 1 分のインターバルをおいて繰り返し実行しており，その利用状況のモニタリングができることが示せた．また，クラスタ環境で汎用的に利用されているプログラムを改変することなく，仮想インフラ上で実行できることを確認した．

7. 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 [15] ~ [17]．OSCARS [15]，ORCA [16] および Open DRAC [17] は，ネットワーク資源の予約のためのフレームワークであり，ネットワーク機器のマネージメントまでを対象とした資源管理システムである．モニタリングシステムでは，perfSONAR [18] が複数のネットワークテストベッドで既にデプロイされており，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てクライアン

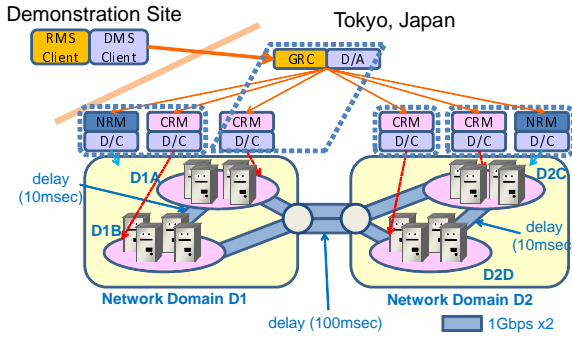


図 8 想定する広域環境

Fig. 8 Assumed distributed resource environment.

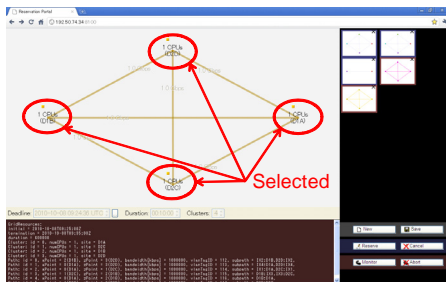


図 9 ポータル画面（予約結果）

Fig. 9 Portal view.

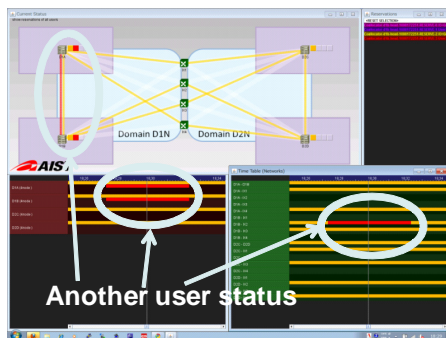


図 10 管理者用モニタリング結果

Fig. 10 Administrator view of monitoring

トに提供する。perfSONAR は、ネットワーク資源専用で物理ネットワークのモニタリングツールである点、情報を集中 DB に格納する点で DMS とは異なる。また、多様な資源の確保、仮想インフラの構築、モニタリングまでを全て自動化した資源管理フレームワークは GridARS の他にはない。

8. まとめ

本研究では、PureWS 版を拡張し、相互運用性及び機能性の高いマルチドメインクラウド資源管理フレームワーク GridARS を開発した。大規模評価実験では、SSL 通信を行った場合でも高速に予約、モニタリング処理が行えることを示した。また、GLIF2010 および SC10 でマルチドメインクラウド環境での GridARS の実証実験を行い、仮想インフラの自動的な構築およびモニタリングに成功した。

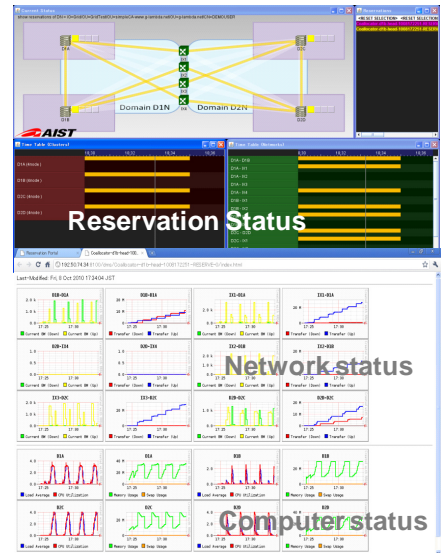


図 11 ユーザー用モニタリング結果

Fig. 11 User view of monitoring

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」、新世代ネットワークサービス基盤構築技術に関する研究開発、および科研費（21700047）の助成を受けたものである。

文 献

- [1] A. Takefusa et al, "GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources," Proceedings of 13th Workshop on Job Scheduling Strategies for Parallel Processing, pp.152-168, June 2007.
- [2] 竹房ほか, "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," 電子情報通信学会技術研究報告, pp.247-252, March 2010.
- [3] 中田ほか, "ネットワーク帯域予約と os 仮想化機構を用いた分散アプリケーション実行環境," コンピュータシステム・シンポジウム論文集, pp.51-58, 2009.
- [4] A. Takefusa 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, March 2009.
- [5] G-lambda プロジェクト, "<http://www.g-lambda.net/>".
- [6] S. Graham and A. Karmarkar and J. Mischkin and I. Robinson and I. Sedukhin, ed., Web Services Resource 1.2 (WS-Resource), OASIS, April 2006.
- [7] Apache CXF, "<http://cxf.apache.org/>".
- [8] Jetty, "<http://www.eclipse.org/jetty/>".
- [9] LinuxContainers, "<http://lxc.sourceforge.net/>".
- [10] RRDtool, "<http://oss.oetiker.ch/rrdtool/>".
- [11] Ganglia Monitoring System, "<http://ganglia.info/>".
- [12] Y. Kodama et al, "GNET-1: Gigabit Ethernet Network Testbed," Proc. IEEE International Conference on Cluster Computing Cluster2004, pp.185-192, Sept. 2004.
- [13] Sun Grid Engine, "<http://gridengine.sunsource.net/>".
- [14] H. Nakada et al, "An Advance Reservation-based Computation Resource Manager for Global Scheduling," Proc. the 3rd International Workshop on Grid Computing and Applications, June 2007.
- [15] OSCARS, "<http://www.es.net/oscars/>".
- [16] ORCA, "<http://nicl.cod.cs.duke.edu/orca/>".
- [17] Open DRAC, "<http://www.opendrac.org/>".
- [18] PerfSONAR, "<http://www.perfsonar.net/>".