

WSRFに基づく情報サービスのXACMLによるアクセス制御

竹房 あつ子[†] 中田 秀基[†] 柳田 誠也,^{†,††}
工藤 知宏[†] 田中 良夫[†] 関口 智嗣[†]

グリッドでは資源やジョブに関する情報の提供は重要な機能のひとつであり、クラスタ監視ツールのGangliaやGlobus Toolkit 4のMDS4などの情報サービスが実現されている。しかしながら、複数の仮想組織に属するユーザが資源を共有する場合、全情報をすべてのユーザに開示することは好ましくない。本研究では、認可モデルとポリシー記述言語の標準として提案されているXACMLを用い、サイト毎に情報開示ポリシーを定義して、各ユーザからの細粒度の情報アクセス制御を可能にする、WSRFに基づく情報サービスシステムを提案する。本研究で開発したプロトタイプでの予備実験から、(1)情報収集に関するオーバーヘッドはWSRF/GSIによるもので占められ、認可決定の時間は無視できる、(2)複数サイトから情報収集する場合も、手続きを並行して行うとその所要時間は許容できる、(3)XACMLのポリシー数が多い場合も、判定に要する時間は全体の処理時間に対して十分小さいことが確認できた。

Access Control using XACML for WSRF-based Information Service

ATSUKO TAKEFUSA,[†] HIDEMOTO NAKADA,[†] SEIYA YANAGITA,^{†,††}
TOMOHIRO KUDOH,[†] YOSHIO TANAKA[†] and SATOSHI SEKIGUCHI[†]

It is an important issue to provide information on Grid resources and job status. The Ganglia cluster monitoring tool and MDS in Globus Toolkit 4 can provide such information service. However, all the information should not be disclosed for all users, who might belong to different virtual organizations. We propose a WSRF-based information service system, which enables various policy definitions by XACML, a standard of authorization model and a policy description language, and fine-grain access control of information. The experiments using our prototype system show: (1) The overhead of XACML authorization decision process is negligible, while that of WSRF/GSI is dominant, (2) Parallel information aggregation from multiple sites makes the overhead acceptable, and (3) Even for a large number of policies, the XACML overhead is still negligible over the total processing time.

1. はじめに

グリッドでは、複数の組織から提供される広域に分散する資源を用いて、高性能計算を行うことができる。この際、利用している計算機やネットワーク等の資源の状態や、ユーザのジョブの実行状況の監視要求が非常に大きい。従来、SSH等により割り当てられた計算ノードに直接ログインし、このような情報を直接取得することができたが、大規模計算を行う場合には計算ノード数が非常に多くなってしまったり、そもそも全ノードに個人のローカルアカウントが作られていない場合もあり、手作業での監視には限界がある。

これに対して、グリッドに関するツールキットを提供しているGlobus Toolkit 4¹⁾では、WSRF(Web Ser-

vices Resource Framework)²⁾に基づく汎用情報サービスをMDS4(GT 4.0 WS Monitoring and Discovery System)として提供している。また、Ganglia³⁾では管理下の計算ノード群の負荷やメモリ使用量等の資源情報を、WebベースまたはMDS4を介してWSRFベースで公開することができる。これらの情報サービスシステムでは、基本的に全てのユーザに対して全ての情報を公開している。しかしながら、複数の仮想組織に属するユーザが資源を共有する場合、特に商用サービスとして多様なユーザに資源を共有させている場合、実行プロセス情報等を全てのユーザに開示することは好ましくない。個々のユーザやユーザの所属する仮想組織等により、開示する情報を細粒度で制御する必要がある。

本研究では、認可モデルとポリシー記述言語の標準として提案されているXACML(eXtensible Access Control Markup Language)⁴⁾を用い、各サイト毎に情報開示ポリシーを定義して、各ユーザからの細粒度の

[†] 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology (AIST)

^{††} 数理技研 SURIGIKEN Co., Ltd.

アクセス制御を可能にする、情報サービスシステムを提案する。本システムは、各サイトで情報を収集して XACML で定義したポリシーに基づく認可決定により情報提供する InformationCollector(IC) と、複数サイトの認可された情報を統合してユーザーに提供する InformationAggregator からなり、いずれも WSRF に基づくインタフェースを提供している。

Globus Toolkit 4 の WS Core とサンマイクロシステムズの提供する XACML の参照実装を用いて、提案する情報サービスシステムのプロトタイプを開発した。プロトタイプを用いて予備的に評価した結果から、XACML による認可決定の時間はポリシー数が多い場合も十分小さく、複数 IC から情報収集する場合もその所要時間は許容できることを示す。

2. XACML

2.1 ポリシ言語の選択

アクセス制御のモデルは古くから研究されており様々な手法が提案されている。しかしこれらの多くは制御対象が限定され、十分な記述力、汎用性があるとは言いがたい。たとえば、代表的なアクセス制御ポリシーとして、UNIX のファイルへのアクセスをユーザごとに細粒度制御する POSIX ACL⁵⁾ があるが、アクセス対象がファイルに限定され、アクセスするユーザ名以外の情報を利用してポリシーを規定することができない。複雑で階層的な構造を持つ実世界情報へのアクセス制御ポリシーを柔軟に記述するためには、より高度で柔軟な認可モデルと階層的な構造を持つポリシー記述言語が必要となる。

このような構造を持つ言語の一つとして XML を基盤とした XACL (XML Access Control Language)⁶⁾ がある。XACL は IBM が規定した、XML のアクセス制御を行う仕様で、ポリシーの制御対象に関しては十分な記述力を持つ。しかし、XACL で規定している制御動作 (Action) は、読み込み、書き込み、生成、削除の 4 つに制限されており、汎用性が十分ではない。

これに対して、標準化団体の OASIS において、XACL を拡張して定義されたものが XACML である。XACML は、多様な資源へのアクセス制御を実現することのできる柔軟なポリシー仕様となっている。XACML は標準規格として多くの組織で採用されているため、今後も評価エンジンや周辺ツールが広く提供されることを期待することができる。XACML は汎用で柔軟なポリシー記述が可能である反面、ポリシーの記述が煩雑であるが、これはポリシー記述ツールの発展、普及によって解決すると思われる。よって、本システムでは認可モデルおよびポリシー記述言語として XACML を用いる。

2.2 XACML 認可モデル

XACML の認可モデルを図 1 に示す。主なモジュール

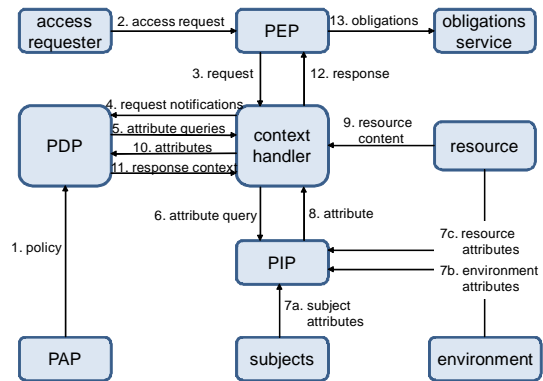


図 1 XACML 認可モデル

Fig.1 The XACML authorization model.

ルを以下に示す。

PIP Policy Information Point. 属性情報を提供する。

PAP Policy Administration Point. ポリシ (ルールのセット)、またはポリシーセットを生成する。

PDP Policy Decision Point. 適用可能なポリシーを評価し、認可決定を下す。

PEP Policy Enforcement Point. 認可決定をもとにアクセス制御を実行する。

Context Handler (CH) コンテキストを処理するシステムであり、決定要求を XACML 形式へ、認可決定を応答形式へ変換する。

Obligation Service PEP のポリシー実行に際して、指定された仕事を同時に行う。

認可の手順は以下の通りである。

1. PAP が PDP で利用可能なポリシーまたはポリシーセットを生成する。
2. リクエストが PEP にアクセス要求を送る。
3. PEP はその要求を CH に送る。
4. CH は要求を XACML 形式に変換して PDP へ送信する。
5. PDP は subject, resource, action, environment に関する属性情報を CH に要求する。
6. CH は各属性情報を PIP に要求する。
7. PIP は要求された属性情報を取得する。
8. PIP はその属性情報を CH に送信する。
9. CH は必要に応じてコンテキストに resource を含める。
10. CH は要求された属性情報を PDP に送り、PDP はそれを用いて該当するポリシーを評価する。
11. PDP は CH に認可決定を送る。
12. CH はその認可決定を PEP の応答形式に変換して、PEP に送信する。
13. PEP は責務を果たす。すなわち、アクセスが許可されれば PEP はその資源へのアクセスを許可し、そうでなければアクセスを否認する。

```

<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <Subject>
    <Attribute
      AttributeId="属性識別子"
      DataType="属性の型">
    <AttributeValue>属性値</AttributeValue>
  </Subject>
  <Resource>
    <Attribute
      AttributeId="属性識別子"
      DataType="属性の型">
    <AttributeValue>属性値</AttributeValue>
  </Resource>
  <Action>
    <Attribute
      AttributeId="属性識別子"
      DataType="属性の型">
    <AttributeValue>属性値</AttributeValue>
  </Action>
</Request>

```

図 2 認可要求の構成

Fig. 2 Description of an authorization request.

```

<?xml version="1.0" encoding="UTF-8"?>
<Policy
  PolicyId="ポリシ識別子"
  RuleCombiningAlgID="アクセス可否の判定方法">
  <Description>ポリシの説明</Description>
  <Target>ポリシを適用する要求</Target>
  <Rule>判定規則</Rule>
</Policy>

```

図 3 ポリシの構成

Fig. 3 Description of an authorization policy.

2.3 XACML のコンテキスト

XACML のコンテキストには、認可要求、ポリシまたはポリシセット、認可決定応答がある。上述した認可モデルでは、認可要求を受け取ると、ポリシおよびポリシセットに従って PDP で認可決定を行っている。以下に各コンテキストの構成を説明する。

2.3.1 認可要求

認可要求は図 2 のように構成されている。<Request> タグの下に、誰が、何に対して、何を行いたいかを、それぞれ<Subject>、<Resource>、<Action>タグで記述する。

2.3.2 ポリシおよびポリシセット

ポリシの構成を図 3 に示す。このポリシを適用する要求は、<Target>タグで記述することができる。<Target>タグの下には、<Subject>、<Resource>、<Action>を記述することができ、これらは認可要求における各タグと対応している。<Rule>タグでは、判定規則を記述することができる。ひとつのポリシファイルの中で<Rule>を複数記述することもできる。その場合、<Policy>タグ内の RuleCombiningAlgID 属性におい

```

<?xml version="1.0" encoding="UTF-8"?>
<PolicySet
  PolicySetId="ポリシセット識別子"
  PolicyCombiningAlgID="アクセス可否の判定方法">
  <Description>ポリシセットの説明</Description>
  <Target>ポリシセットを適用する要求</Target>
  <Policy>ポリシ</Policy>
  <PolicyIdReference>既存ポリシの ID
</PolicyIdReference>
</PolicySet>

```

図 4 ポリシセットの構成

Fig. 4 Description of an authorization policy set.

```

<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <Result>
    <Decision>認可決定結果</Decision>
  </Result>
</Response>

```

図 5 認可決定応答の構成

Fig. 5 Description of an authorization decision response.

て、複数の規則を用いた場合の判定方法を指定することができる。その方法として、deny-overrides, permit-overrides, ordered-permit-overrides, ordered-deny-overrides, first-applicable がある。

ポリシセットの構成を図 4 に示す。ポリシセットは複数のポリシをまとめたものであり、ある要求に対して複数のポリシを同時に適用する場合に利用する。<PolicySet>内の PolicyCombiningAlgId 属性を用いることで、複数のポリシを用いた場合の判定方法をポリシの場合と同様に指定することができる。ポリシセットにおけるポリシの記述方法として、<PolicySet>タグ下の<Policy>および<PolicySet>タグで直接ポリシを記述する方法と、<PolicyIdReference>または<PolicySetIdReference>で既存ポリシまたはポリシセットの ID を指定する方法がある。

2.3.3 認可決定応答

図 5 に認可決定応答の構成を示す。<Result>タグ下の<Decision>で認可決定結果を記述する。PDP は認可決定結果として、Permit (許可)、Deny (否認)、Indeterminate (評価過程で構文エラーが発生)、NotApplicable (適用するルールがなかった) のいずれかを返さなければならない。

3. WSRF に基づく情報サービスシステム

3.1 WSRF の概要

本節では、本システムで基盤として用いた WSRF に関して簡単に述べる。一般に Web Services におけるサービスは明示的に内部状態を持たず、クッキーなどのセッション管理手法を用いて、暗黙裏に内部状態を表現する。内部状態の表現方法に標準はなく、外部

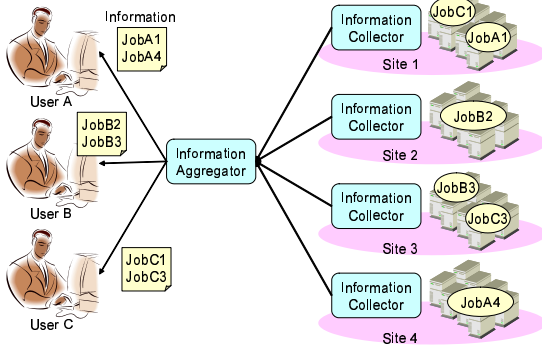


図 6 情報サービスシステムモデル

Fig. 6 The system model of the information service.

からのアクセスもできないため、相互運用性に問題があった。この問題を解決するために Global Grid Forum(現 Open Grid Forum)で提案され、OASIS で標準化された規格が WSRF である。WSRF では、サービスと明確に分離され、サービスから参照される内部状態であるリソースと呼ばれる概念を導入する。サービスとリソースの ID をペアで扱うことで、内部状態をもつサービスを実現することができる。リソースは複数のリソースプロパティから構成される。リソースプロパティはキーとそれに対応する値のペアである。以下本稿で、リソースはこの WSRF におけるリソースを指す。

3.2 情報サービスシステムの概要

図 6 に提案する情報サービスシステムモデルを示す。本システムは複数サイトに分散する各ユーザの情報を統合して提供する InformationAggregator(IA) と、各サイトにおいて認可された情報を提供する InformationCollector(IC) からなり、いずれも WSRF インタフェースを提供する。IA では、ユーザから情報取得要求を受け取ると、関連するサイトの IC にそのユーザの権限を移譲して問い合わせる。各 IC は、問い合わせを受けたユーザの属性情報とユーザが望む情報のアクセスポリシーとを照らし合わせ、認可されると要求された情報を提供する。IA は各 IC から取得した情報を統合し、要求したユーザに送信する。ただし、IC は特定の IA の管理化におかれるわけではなく、問い合わせのあった複数の IA に対して情報を提供する。

例えば、図 6 において User A が実行中である JobA1 と JobA4 の実行状況に関する情報の取得要求を送ると、IA は Site 1 と Site 4 に情報取得要求を送信する。* 各サイトでジョブを実行中のユーザにのみ情報を提供するというポリシーを設定している場合、Site 1 と Site 4 において User A のジョブが実行中であるた

* JobA1 と JobA4 の実行サイトはユーザにとって既知であることを前提としている。この情報はメタスケジューラなどの、提案情報サービス外のシステムから与えられる。

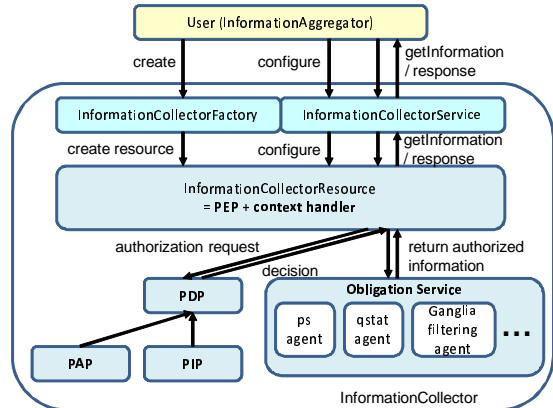


図 7 InformationCollector システムアーキテクチャ

Fig. 7 The InformationCollector system architecture.

め、そのジョブの実行状況に関する情報を IC が提供する。IA がこれらの情報をまとめて User A に提供する。

3.3 InformationCollector アーキテクチャ

提案システムでは、IC でのみ認可決定手続きを行っており、他のユーザの実行状況等の認可されていない情報がサイト外に流出しないように設計されている。図 7 に IC のシステムアーキテクチャを示す。IC は XACML の認可モデルに基づいており、WSRF インタフェースモジュール (InformationCollectorFactory, InformationCollectorService)、ステートフルサービスインスタンスモジュール (InformationCollectorResource(ICR)), 認可決定関連モジュール (PDP, PAP, PIP), Obligation Service モジュールで構成される。ICR は PEP と context handler の役割も担っている。

WSRF インタフェースモジュールは、情報サービスの受付と、その後の各サービスオペレーションの窓口となる。ICR は各情報サービス要求のサービスインスタンスであり、各要求に関する情報を管理すると共に、認可決定関連モジュールへの認可決定要求と、その認可決定に基づく Obligation Service モジュールからの認可情報の取得を行う。Obligation Service モジュールでは、認可された情報を提供する。本システムでは、様々な情報を取得するために各情報に対応したエージェントを用意し、各エージェントが動的または静的な情報を提供する。

各サイトの管理者は、Obligation Service モジュールに開示したい情報を収集するエージェントをあらかじめ登録し、その情報へのアクセスを制御するためのポリシーを PAP 下に用意することができる。ユーザは登録されている情報の中から取得したい情報を指定して、認可されればそれら情報を得ることができる。

3.3.1 提供する情報

本システムでは、CPU クロックやメモリサイズ、計算ノード数など、資源構成等の静的な情報の他に、図 7

で示すように `ps` や `w` など UNIX コマンドから得られる結果, TORQUE⁷⁾ や GridEngine⁸⁾ 等のバッチキューイングシステムのキューの情報, Ganglia のモニタリング情報等の動的な情報を定期的に収集して提供することを想定している。

提供する情報のセマンティクス, シンタックスは本システムを用いるコミュニティ全体で何らかの合意がとられているものとする。

3.4 サービスオペレーション

IA と IC が提供する主要なサービスオペレーションを次に示す。いずれも Factory サービスの `create` オペレーションによりサービスインスタンスである `InformationAggregatorResource(IAR)`, `ICR` を生成し, 生成したリソースに対して各サービスオペレーションを実行する。

3.4.1 InformationAggregator のサービスオペレーション

表 1 に IA のサービスオペレーションを示す。それぞれオペレーション名, オペレーションの機能, 入力, 出力を示す。 `configure` では, どここのサイト・ホストからどのような情報を取得したいか (Ganglia の資源情報など), またいつからいつまでの情報が欲しいかを, あらかじめ登録する。 `startAggregate` により, `configure` で指定した情報の収集を開始する。情報収集期間中に, `getInformation` または `subscribeAggregate` オペレーションを用いて情報を取得することができる。 `getInformation` では, このオペレーションが実行されるまでに収集した情報のうち, 認可されたものを返す。 `subscribeAggregate` では, WS-Notification⁹⁾ に基づき, 認可された指定情報を IC が取得するたびに, ユーザ側に通知される。 `unsubscribeAggregate` で通知に基づく情報収集を停止する。最後に `stopAggregate` で各 Collector での情報収集を停止する。

3.4.2 InformationCollector のサービスオペレーション

表 2 に IC のサービスオペレーションを示す。それぞれ, オペレーション名, 機能, 入力, 出力と XACML に基づく認可判定の有無を示している。 `configure` では, サイト内のホスト名, 取得する情報名, 時刻に関する要求を登録する。 IA 同様, `startCollect`, `stopCollect` により, 指定された情報の収集の開始, 停止を行う。また, 情報収集期間中に, `getInformation`, `subscribe` を用いることで情報を取得することができ, `unsubscribe` で通知に基づく情報収集を停止する。指定された情報の提供にあたり, `getInformation` が呼ばれるたびに, また通知ベースで情報を提供するたびに XACML による認可判定が行われる。これは, 時刻やポリシーの変更により, 判定条件が変更しても対応できるようにするためである。

3.5 情報取得プロセス

IA と IC の連携による情報取得プロセスを図 8 に示

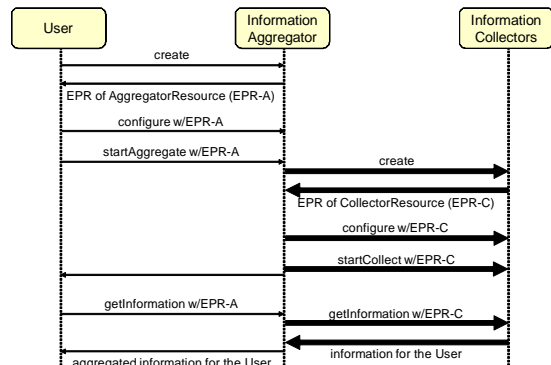


図 8 情報取得プロセス

Fig. 8 Protocol sequence of information aggregation.

す。ユーザは IA に対して `create` で情報取得受付要求を送ると, そのインスタンスである IAR へのポインタである Endpoint Reference (EPR-A) が送られる。ユーザは EPR-A を用いて `configure` コマンドで取得したい情報セットを設定する。その後 `startAggregate` を実行すると, IA から IC に対して情報収集処理が開始される。IA から関連する複数の IC に対して並行して同時に手続きが進められる。各 IC への手続きでは, IA の場合と同様に `create` で情報取得受付要求を送り, そのインスタンス ICR へのポインタ EPR-C を受け取ると, それを用いて `configure` 要求を送信する。その後, `startCollect` によりサイト内での情報収集処理を開始する。各サイトでの情報収集が開始されると, ユーザは `getInformation` または `subscribeAggregate` コマンドにより, 情報取得要求を IA に送信する。IA は IC に対して `getInformation` または `subscribe` により, 各サイトで認可された情報を収集して提供する。

3.6 提案情報サービスの動作概要

提案情報サービスの動作を, `ps` を例にとって説明する。一般ユーザ A, B に対しては本人の `ps` 情報のみを, 管理者権限を持つユーザ C に対してはすべての `ps` 情報を提供する場合を考える。この場合, エージェントとしては, 本人の `PS` 情報を提供するもの (`PS-each`) と, すべての `ps` 情報を提供するもの (`PS-all`) の 2 種類をあらかじめ用意し, それぞれに対して適切なアクセス制御ポリシーを記述しておく。

ユーザ A, B はそれぞれ, `InformationAggregator` に対して `PS-each` エージェントの登録を要求する。要求は各サイトの `InformationCollector` に伝えられ, そこで `PS-each` のインスタンスがそれぞれ作成される。このとき `InformationCollector` は, 各ユーザが提示するプロキシ証明書に書かれている DN (Distinguished Name: ユーザのグローバル ID) および, その DN がマップされるべきローカル UNIX ID を `grid-mapfile` などから取得し, `PS-each` エージェントに設定値として与える。エージェントは, 指定された UNIX ID を

表 1 InformationAggregator のサービスオペレーション
Table 1 Service operations of InformationAggregator.

オペレーション名	機能	入力 / 出力
getInformationName	InformationCollector の提供する情報セットを返す	サイト名 / -
configure	取得したい情報セットを指定	情報セット / -
startAggregate	指定した情報の収集を開始	- / -
stopAggregate	情報収集を停止	- / -
getInformation	指定した情報を取得	- / 指定した情報
subscribeAggregate	WS-Notification に基づく情報収集を開始	- / -
unsubscribeAggregate	WS-Notification に基づく情報収集を停止	- / -

表 2 InformationCollector のサービスオペレーション
Table 2 Service operations of InformationCollector.

オペレーション名	機能	入力 / 出力	認可判定
getInformationName	提供する情報セットを返す	- / -	
configure	取得したい情報セットを指定	情報セット / -	-
startCollect	情報収集を開始	- / -	-
stopCollect	情報収集を停止	- / -	-
getInformation	指定した情報を取得	- / 指定した情報	○
subscribe	WS-Notification に基づく情報収集を開始	- / -	○ (通知ごと)
unsubscribe	WS-Notification に基づく情報収集を停止	- / -	-

ps コマンドの-u オプションに適用することで、個々のユーザの情報を取得する。したがって、ユーザ A 用の PS-each エージェントのインスタンスとユーザ B 用の PS-each エージェントのインスタンスは、それぞれ異なるユーザ ID に対して ps を行うことになる。

ユーザ C は、PS-all エージェントの登録を要求する。PS-all エージェントは依頼したユーザに関わらず、オプションを指定せずに ps コマンドを実行し、すべての情報を取得する。

アクセス制御は、ユーザからの getInformation 要求の時点で、エージェント単位で行われる。仮に、権限を持たない一般ユーザ A が PS-all エージェントの登録を要求したとする。この場合、登録そのものは成功し、エージェントは ps コマンドですべてのユーザの情報を取得する。しかし、ユーザ A が getInformation で情報をエージェントから取得しようとした時点で、XACML を用いた認証がおこなわれ、アクセスが不許可となるため、ユーザ A に対してこの情報が提供されることはない。

3.7 InformationCollector のスケーラビリティ

IC のリソースは各ユーザに対して個別に用意され、エージェントも各ユーザに対して個別に用意される。このため、ユーザ数に比例したリソース、エージェントが必要となる。しかし、定期的に取得する情報は、前回に取得した情報を上書きしてリソース内に保持されるため、リソース、エージェントの要求するメモリ容量は小さく、数キロバイト程度であるため大きな問題にはならない。

情報収集時には、個々のユーザに対応するエージェントが個別に動作するため、CPU に関してもエージェント数に比例した負荷が発生する。たとえば前項でし

めした PS-each をナイーブに実装すると、個々のユーザに対して個別に ps コマンドが実行され、1 ユーザあたり数 ms 程度の CPU 時間が必要となる。しかし、エージェントの実装を工夫し、すべてのユーザに対する情報を 1 度の ps コマンドで取得して、内部で情報を切り分けるようにすれば、ユーザ数に比例しないコンスタントな CPU 時間で情報収集を行うことが可能である。

通信コストは、getInformation の頻度に比例する。各ユーザの getInformation の頻度が同程度であると考えると、通信コストはユーザ数に比例する。1 リクエストあたりの通信データ量は、前述の ps の場合データそのものは数十バイト程度であるが、Web Services のヘッダ、エンベロープなどを考慮に入れると数キロバイトと程度となる。各ユーザの getInformation の頻度は 1 分ないし数分に 1 回程度であると想定すると、数百ユーザ程度までは十分にスケールすることが期待できる。

3.8 提案情報サービスシステムのユースケース

ユーザにグループの概念を導入することでポリシーの記述を容易にすることができる。たとえば、あるグループに属するユーザに対して、同一グループ内に属するユーザの ps 情報のみを見せることが可能である。これを実現するためには、グループ名とユーザの DN の集合を対応づけるグループマッピングファイルを用意するとともに、このファイルを参照してポリシー認可を行うための関数を独自に用意する必要がある。また、各グループに対して個別に ps を行うエージェントを用意する。情報を取得するエージェントはグループマッピングファイルからグループに属するユーザの DN を取得し、この DN からローカルユーザ ID を取得する。

このローカルユーザ ID を ps コマンドの-u オプション^{*}に指定することで、該当するグループのユーザ情報を取得する。

また、本情報サービスの対象とする資源情報は、計算機資源由来のものだけではない。たとえば、われわれは、ネットワークと計算機を同時に確保する GridARS (Grid Advance Reservation-based System Framework)¹⁰⁾を提案し、実証実験を行っている¹¹⁾。精度よくネットワーク資源を選択するためには、ネットワーク資源に関するさまざまな情報(ネットワーク事業者内での物理的な接続関係、資源の過去の使用履歴など)の開示が必要になる。しかしこのような情報はネットワーク事業者にとってはある種の機密情報であり、一般ユーザに対して公開することはできない。提案システムを適用することにより、個別の守秘契約をむすんだユーザに対してのみ資源情報を提供することが可能になる。一般ユーザはネットワーク資源情報が得られないため、精度の良くない(かならずしも最適でない)資源選択を行うことになる。これに対して、守秘契約をむすんだユーザは、ネットワーク資源情報に基づいた精度のよい資源選択を享受することができる。

4. 情報サービスシステムの実装

WSRF の参照実装である Globus Toolkit 4 (GT4) と、サンマイクロシステムズが提供している XACML バージョン 1.x に基づく Java の参照実装¹²⁾を用い、情報サービスシステムのプロトタイプを開発した。

GT4 を用いることで、GSI (Grid Security Infrastructure) を用いたユーザの認証、grid-mapfile を用いたグローバルユーザ名とローカルサイトのユーザ名のマッピング情報の取得、各手続きと通常の情報取得時および通知ベースでの情報取得時の SSL (Secure Socket Layer) によるセキュアな通信を実現している。また、権限移譲機能によりユーザは IA とのやり取りのみで、そのユーザの証明書を IC に送ることができる。

本情報サービスシステムでは、参照実装の提供するポリシー検索モジュールを適宜呼び出し、指定したディレクトリ下のポリシーファイルを読み込んで PDP を構築する。これにより、GT4 のコンテナの再起動などをすることなく、容易にポリシーおよびポリシー評価関数を追加・修正することができる。また、XACML 認可モデルでは認可判定時に対応する情報資源を収集して判定を行うが、参照実装では提供されていない。よって、情報資源(本システムでは Obligation Service に登録されている情報)検索のためのモジュールを別途実装し、認可判定時に情報資源を利用できるようにした。

4.1 XACML によるポリシー記述例

XACML の参照実装では指定されたポリシーファ

イルと評価関数から認可判定を行う。評価関数は、string-equal など XACML の仕様であらかじめ規定されている関数の他に、独自の評価関数を用意することもできる。5 節の評価で用いたポリシーの記述例および独自評価関数の実装例を以下に示す。

図 9 にアクセス時刻を指定するためのポリシー、図 10 に証明書内の DN(Distinguished Name) とローカルユーザ名のマッピングが grid-mapfile 内にあるか判定するポリシーの記述例を示す。図 10 では<Target>タグ内など重複する部分は割愛した。なお、ここでは grid-mapfile との照合部分を<Rule>の<Condition>として実現しているが、<Target>タグ内の<Subject>部分で制約することも可能である。

2.3.2 節で述べたように、図 9 では<Policy>タグ内の PolicyId でこのポリシーの ID を記述し、RuleCombiningAlgId で複数規則を用いた場合の判定方法を指定している。ここで、permit-overrides は判定規則のうちどれか 1 つでも Permit と判定したら、それ以外の判定規則の結果にかかわらずポリシーとしての判定を Permit とすることを意味する。<Target>では、すべての Subject, Resource, Action に対してこのポリシーを適用すると指定している。<Rule>タグでは、独自に実装した指定した時刻でのみアクセス可能とする規則と、既存の DenyAllOthers 規則を列挙している。図 10 でも同様にポリシーを定義している。

図 11 に上記 2 つのポリシーをポリシーセットとした例を示す。<PolicySet>タグ内の PolicySetId でこのポリシーセットの ID を記述し、RuleCombiningAlgId で複数ポリシーを用いた場合の判定方法を指定する。deny-override は指定したポリシーのうちどれか 1 つでも Deny と判定したら、それ以外のポリシーの判定結果にかかわらずポリシーセットとしての判定を Deny とすることを意味する。すなわち、情報取得要求が送られた時刻が 9 時から 23 時の間であり、かつ grid-mapfile 内にユーザ名がある場合のみ、Permit となる。

4.2 評価関数実装例

図 12 に、独自の評価関数の実装例を示す。評価関数は、com.sun.xacml.cond.FunctionBase クラスを継承して実装する。functionName に図 9 のポリシーファイル内で指定される関数名を定義し、実装した関数のコンストラクタ内で、スーパークラスのコンストラクタに渡す。また、evaluate() メソッドによりこの評価関数を定義することができる。ここでは、evalArgs() メソッドを用いてポリシーファイル内に記述された AttributeValue 値("09:00:00", "23:00:00")を読み込み、現在の時刻と指定した時刻を比較して、指定した時刻内であれば EvaluationResult を true に、時間外であれば false にして返す。図 9 において、この規則の Effect を Permit としているため、このポリシーの判定結果は評価関数の返り値が true であれば Permit, false であれば Deny となる。

^{*} ps コマンドの-u オプションは、カンマで区切られた複数のユーザ名を処理することができる。

```

<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:1.0:policy"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  PolicyId="TimeRangePolicy1"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Description> Between 9am and 11pm local time, allow anyone to access. </Description>
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources><AnyResource/></Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Rule RuleId="PermitDuringSpecifiedTimeRange" Effect="Permit">
    <Condition
      FunctionId="http://gridars.aist.go.jp/informationService/Collector/XACML/TimeFunction">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">23:00:00</AttributeValue>
    </Condition>
  </Rule>
  <Rule RuleId="DenyAllOthers" Effect="Deny"/>
</Policy>

```

図 9 ポリシ例 1 : 指定した時刻内でのアクセスを許可するポリシ
Fig. 9 A policy example 1: A policy to allow access within a specified time.

```

<?xml version="1.0" encoding="UTF-8"?>
<Policy PolicyId="DNMatchPolicy1"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Description> permit if DN in request is registered in grid-mapfile </Description>
  <Target>.</Target>
  <Rule RuleId="DNMatch" Effect="Permit">
    <Condition
      FunctionId="http://gridars.aist.go.jp/informationService/Collector/XACML/DNMatchFunction">
    </Condition>
  </Rule>
  <Rule RuleId="DenyAllOthers" Effect="Deny"/>
</Policy>

```

図 10 ポリシ例 2 : grid-mapfile にあるユーザのみアクセスを許可するポリシ
Fig. 10 A policy example 2: A policy to allow access for users listed in grid-mapfile.

```

<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="PolicySet1"
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides">
  <Description> Example policy set. </Description>
  <Target> <Resources><AnyResource/></Resources> </Target>
  <PolicyIdReference>DNMatchPolicy1</PolicyIdReference>
  <PolicyIdReference>TimeRangePolicy1</PolicyIdReference>
</PolicySet>

```

図 11 ポリシセット例
Fig. 11 A policy set example.

5. 予備実験

情報サービスシステムおよび XACML の認可決定に要するオーバーヘッドを調査するため、2 つの予備実験を行った。

5.1 情報の収集・取得に関する所要時間

1 つめの実験として、情報サービスシステムを利用

して情報を取得する際に要するオーバーヘッドを、IA と IC が 1 対 1 の場合と 1 対 7 の場合で比較した。実験では、IA と IC は同じクラスタ内に設置した。各ホスト間の遅延は約 40us で、1 つの 1Gbps のスイッチに接続されている。全ホストの構成は Pentium 4 2.8GHz, 1GB メモリ, CentOS4.3 となっている。Java の処理系としては Sun Microsystems による JDK 1.5.0 を用いた。

```

public class GridarsInformationCollectorXACMLTimeFunction extends com.sun.xacml.cond.FunctionBase {
    private static String functionName =
        "http://gridars.aist.go.jp/informationService/Collector/XACML/TimeFunction";

    // コンストラクタ
    public GridarsInformationCollectorXACMLTimeFunction() {
        // super(functionName, functionID, paramType, paramIsBag, numParams, returnType, returnsBag)
        super(functionName, 0, TimeAttribute.identifier, false, 2, BooleanAttribute.identifier, false);
    }

    // 評価関数
    public EvaluationResult evaluate(List inputs, EvaluationCtx context) {
        // argValues にポリシーに書かれた AttributeValue("09:00:00", "23:00:00") を格納
        AttributeValue [] argValues = new AttributeValue[inputs.size()];
        EvaluationResult result = evalArgs(inputs, context, argValues);

        long startTime = argValues[0] から 09:00:00 を読み込む
        long endTime = argValues[1] から 23:00:00 を読み込む
        long currentTime = 現在の時刻を読み込む

        if((startTime <= currentTime) && (currentTime <= endTime))
            return EvaluationResult.getInstance(true);
        else
            return EvaluationResult.getInstance(false);
    }
}

```

図 12 評価関数実装例

Fig. 12 An implementation example of an evaluation function.

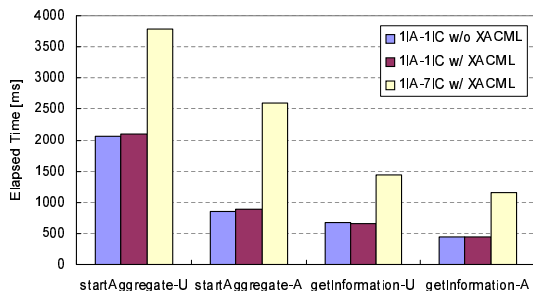


図 13 情報収集開始処理および情報取得の所要時間

Fig. 13 Elapsed time of startAggregate and getInformation.

取得する情報は UNIX の以下のコマンドの実行結果である。

```
$ ps ux --cols 80 -u <ローカルユーザ名>
```

この測定では、4.1 節で示した以下の 2 つのポリシーをポリシーセットとして認可判定を行った。

- ユーザ証明書の DN が grid-mapfile 内のローカルユーザアカウントへのマッピングがあるか
- 予め指定した時刻内の問い合わせか

図 13 に情報収集開始処理 (startAggregate) および情報取得処理 (getInformation) の所要時間を ms で示す。横軸には startAggregate と getInformation をユーザ側で測定した時間 (-U)、IA 側で測定した時間 (-A)

表 3 startAggregate および getInformation オペレーションにおける InformationCollector 内部での処理時間。7IC の場合は全 IC の平均値。

Table 3 Elapsed time of startAggregate and getInformation measured by InformationCollectors. 7IC indicates the average of seven results.

	startAggregate	getInformation
1IA-1IC w/o XACML	2.7 ms	0.8 ms
1IA-1IC w/ XACML	2.2 ms	1.9 ms
1IA-7IC w/ XACML	3.0 ms	1.4ms

を示している (図 8 参照)。認可判定は IC で行われているため、-U と -A の所要時間の差は GSI での WSRF サービスのメッセージングオーバーヘッドと一致している。また、3 つの測定結果 1IA-1IC w/o XACML, 1IA-1IC w/ XACML, 1IA-7IC w/ XACML は、IA と IC が 1 対 1 で XACML での認可判定がないものもあるもの、1 対 7 で XACML での認可判定があるものの結果を示している。グラフでは 10 回測定したうちの最短値を採用した。

図 13 より、1 対 1 の場合で認可判定の有無で比較すると、認可判定に要するオーバーヘッドがほとんどないことが分かる。実験では、表 3 で示すように認可判定を含む IC 内での処理に要する時間を同時に測定したが、いずれも 0.8-3ms 程度であった。よって、情報の収集、取得に関するオーバーヘッドは SOAP 通信のためのデータ型の変換といった WSRF/GSI に関する

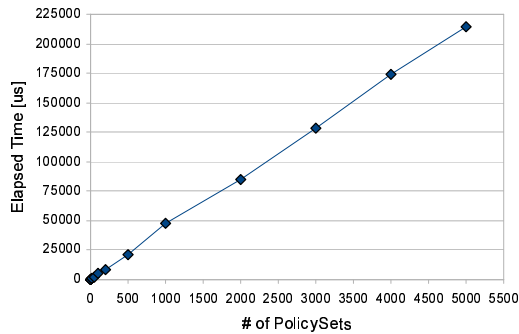


図 14 XACML 認可判定の所要時間。

Fig. 14 Elapsed time of XACML authorization decisions.

手続きでほぼ占められており、XACML の認可決定に要する時間は無視できる。

1 対 1 と 1 対 7 の場合の比較では、1 対 7 の方が WSRF での 7 つの IC への手続きのオーバーヘッドのため所要時間が長くなってしまふ。しかしながら、IA から IC への手続きは並行して行っているため、IA 側で測定した値と比較すると、IC の数に比例することなく startAggregate では 2.9 倍、getInformation では 2.6 倍程度にとどまっていた。

本システムでは 3.3.1 節で述べたような動的な情報の提供を主な目的としており、IA が同時に問い合わせる IC 数はユーザのジョブが同時に利用するサイト数を想定している。通常のグリッドシステムでは、サイト（クラスタ）内のノード数は多いものの、サイト数は参加組織の数に依存し、かつ遅延等の影響により 1 つのジョブが同時に利用するサイト数は 5 サイト以下程度である。本実験では、7 つの IC から情報を集める場合でも、その所要時間は IC 数に比例することなく許容できることが示された。

5.2 XACML での認可決定所要時間

次に、XACML での認可決定に際して、ポリシー数に対する所要時間の変化を調査した。ただし、ポリシーの読み込みは事前に行われるため、ここではポリシーの読み込みに要する時間は含まない。また、実験では、ヒープ領域を十分大きくすると同時に、評価の直前に強制的に GC(ガバージョコレクション) を実行することによって、評価時間への GC の影響は排除されている。実運用環境では認可決定中に GC が起きる可能性があるが、近代的な Java 処理系では世代別 GC や並列 GC などの技術によって、GC による影響は十分小さい。

認可決定の所要時間の測定結果を図 14 に示す。縦軸には所要時間を us で、横軸にはポリシーセット数を示している。ポリシーセットは 5.1 節で用いた 2 つのポリシーを 1 セットとして、5000 セット（ポリシーの認可決定は計 10000 回）まで変化させた。実際の運用では、一度の認可決定で評価するポリシー数は数十から多くても

数百程度と考えられるが、本実験では純粋な XACML の認可決定オーバーヘッドを調査するため、5000 セットまでの所要時間を測定した。各ポリシーセットの大きさに応じて、GC が生じない程度の回数を 1 セットとして、それぞれ 10 セットの計測を行い、その平均値を示している。

図 14 から、評価時間はポリシーセット数に対してほぼ線形であることがわかる。ポリシーセット数 1000 で 50[ms]、5000 で 230[ms] 前後となっており、いずれも 5.1 節での全体の処理時間と比較すると無視できる。従って、XACML による認可判定のオーバーヘッドはポリシー数が多い場合も全体の処理時間に対して十分短いことが示された。

6. 関連研究

Globus Toolkit 4 (GT4) では、WSRF に基づくグリッドシステムにおいて、サービスインスタンスであるリソースへのアクセスの認可のため、XACML に基づく認可フレームワークを提案している¹³⁾。このフレームワークでは、そのユーザがサービスを利用できるかどうかの制御を行っている。したがって、認可の単位はサービスであり、サービスに紐づけられたリソース内部の要素に対して、個別にアクセス制御を行うことはできない。すなわち、サービスへのアクセスを許可されたユーザは、リソース内のすべての情報に対してアクセスすることが可能になってしまう。一方、本システムはアクセス制御の単位をリソース内のエージェントとしている。これは、3.3.1 節で示したように、各ユーザに対して提供する情報を細粒度で制御するためである。

本研究が対象とする細粒度のアクセス制御を GT4 上で実現するには、各情報を個別のサービスとして一つ一つ構築し、サービス単位でアクセス制御を行わなければならない。これは理論的には不可能ではないが、大量のメモリが必要となるため現実的ではない。また、GT4 の XACML を用いた認可機能は GT4.2 以降で公開予定であり、現状では利用することはできない。

MonALISA (MONitoring Agents using a Large Integrated Services Architecture)¹⁴⁾ は Ganglia や MRTG (Multi Router Traffic Grapher)¹⁵⁾ 等のモニタリングツールで収集された情報をレポジトリに格納し、クライアントインタフェースから提供する。認証は行うものの、開示する情報の認可は行っていないため、同一レポジトリにアクセスするユーザは全ての情報にアクセス可能である。レポジトリおよびレポジトリの情報を用いた各サービスへのインタフェースは、ウェブサービスに基づいている。

Inca¹⁶⁾ は TeraGrid¹⁷⁾ におけるユーザレベルグリッドモニタリングシステムであり、エージェントによりユーザの権限で情報を収集・提供する。Inca では、複

数サイトの情報を集中管理しており、サイト外に収集した情報が流出する点、サイト毎に様々なポリシーを定義して細粒度のアクセス制御を行わない点、WSRF インタフェースではなく Web アプリケーションとして実装されている点で本研究と異なる。

7. まとめと今後の課題

本研究では、認可モデルとポリシー記述言語の標準である XACML を用い、各サイト毎に情報開示ポリシーを定義して、ユーザからの細粒度のアクセス制御を可能にする、情報サービスシステムを提案した。本システムは WSRF に基づき、各サイトで情報を収集して認可した情報を提供する InformationCollector(IC) と、複数 IC からの情報を統合してユーザに提供する InformationAggregator からなる。

また、Globus Toolkit 4 の WS Core とサンマイクロシステムズの提供する XACML の参照実装を用いて、提案する情報サービスシステムのプロトタイプを開発し、予備実験を行った。実験から、(1) 情報収集では WSRF/GSI のオーバーヘッドが大きく、XACML による認可決定の時間は無視できる、(2) 複数 IC から情報収集する場合も、手続きを並行して行うためその所要時間は許容できる、(3) XACML のポリシー数が多い場合も、判定に要する時間は全体の処理時間に対して十分小さいことが確認できた。

今後は、多様なアクセス制御ポリシーの適用、情報サービスシステムと GridARS グリッドコアロケーションフレームワークとの連携と、ユーザインタフェースの構築をすすめる。また、VOMS (Virtual Organization Membership Service)¹⁸⁾ と呼ばれる仮想組織を管理するミドルウェアと連携させることで、ユーザの属する仮想組織とロールに応じたアクセス制御を実現することも今後の課題である。また、4.1 節で記したように XACML のポリシー記述は柔軟である反面複雑であるため、一般の管理者が記述するのは困難である。よって、典型的なポリシーをあらかじめ用意して管理者が必要に応じてポリシーを取捨選択できるようにするとともに、独自のポリシーの記述を容易にするポリシー編集支援ツールを開発する。

謝辞 本研究の一部は、文部科学省科学技術振興調整費「グリッド技術による光パス網提供方式の開発」による。

参考文献

- 1) Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems, *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp.2-13 (2005).
- 2) OASIS Web Services Resource Framework (WSRF) TC: Web Services Resource 1.2 (WS-Resource) Committee Specification (2006).
- 3) Ganglia Monitoring System:
<http://ganglia.info/>.
- 4) OASIS: eXtensible Access Control Markup Language (XACML) Version 2.0 (2005).
- 5) : IEEE 1003.1e and 1003.2c: Draft Standard for Information Technology-Portable Operating System Interface (POSIX)-Part 1: System Application Program Interface (API) and Part 2: Shell and Utilities, draft 17 (withdrawn). (1997).
- 6) Kudo, M. and Hada, S.: "XML Document Security based on Provisional Authorization", *7th ACM Conference on Computer and Communication Security (CCS 2000)* (2000).
- 7) TORQUE Resource Manager: <http://www.clusterresources.com/resource-manager.php>.
- 8) Grid Engine:
<http://gridengine.sunsource.net/>.
- 9) OASIS Web Services Notification (WSN) TC: Web Services Base Notification 1.3 (WS-BaseNotification) Public Review Draft 02 (2005).
- 10) Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y. and Sekiguchi, S.: 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* (2007).
- 11) Thorpe, S. R., Battestilli, L., Karmous-Edwards, G., Hutanu, A., MacLaren, J., Mambretti, J., Moore, J. H., Sundar, K. S., Xin, Y., Takefusa, A., Hayashi, M., Hirano, A., Okamoto, S., Kudoh, T., Miyamoto, T., Tsukishima, Y., Otani, T., Nakada, H., Tanaka, H., Taniguchi, A., Sameshima, Y. and Jinno, M.: G-lambda and EnLIGHTened: Wrapped In Middleware Co-allocating Compute and Network Resources Across Japan and the US, *Submitted to GridNets2007*.
- 12) Sun's XACML Implementation:
<http://sunxacml.sourceforge.net/>.
- 13) Lang, B., Foster, I., Siebenlist, F., Ananthakrishnan, R. and Freeman, T.: A Multipolicy Authorization Framework for Grid Security, *Proceedings of NCA06* (2006).
- 14) Legrand, C., Newman, H. B., Voicu, R., Cirstoiu, C., Grigoras, C., Toarta, M. and Dobre, C.: MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications, *Proceedings of CHEP2004* (2004).
- 15) MRTG Monitoring Tool:

<http://www.mrtg.org/>.

- 16) Smallen, S., Olschanowsky, C., Ericson, K., Beckman, P. and Schopf, J.M.: The Inca Test Harness and Reporting Framework, *Proceedings of the IEEE/ACM SC2004 Conference* (2004). (online proceedings).
- 17) TeraGrid: <http://www.teragrid.org/>.
- 18) VOMS: <http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html>.

(平成?年?月?日受付)

(平成?年?月?日採録)

竹房あつ子 (正会員)

昭和48年生。平成8年お茶の水女子大学理学部情報科学科卒業。平成10年同大学大学院理学研究科情報科学専攻修士課程修了。平成12年同大学院人間文化研究科複合領域科学専攻博士課程修了。博士(理学)。同年日本学術振興会特別研究員、平成14年お茶の水女子大学理学部助手。平成17年独立行政法人産業技術総合研究所入所。現在同所情報技術研究部門研究員。並列分散処理、グリッドコンピューティング、スケジューリングに興味を持つ。ACM、電子情報通信学会各会員。

中田 秀基 (正会員)

昭和42年生。平成2年東京大学工学部精密機械工学科卒業。平成7年同大学大学院工学系研究科情報工学専攻博士課程修了。博士(工学)。同年電子技術総合研究所研究官。平成13年独立行政法人産業技術総合研究所に改組。現在同所情報技術研究部門主任研究員。平成13年より平成17年度まで東京工業大学客員助教授を兼務。グローバルコンピューティング、並列実行環境に関する研究に従事。

柳田 誠也

昭和47年生。平成7年東京大学工学部材料学科卒業。平成9年同大学大学院工学系研究科材料科学専攻修士課程修了。平成12年京都大学大学院工学研究科機械物理工学専攻博士課程修了。博士(工学)。平成15年株式会社数理技研入社。現在、同社基盤・科学技術ユニットにて、クラスタファイルシステムやグリッドコンピューティングに関する開発に従事。

工藤 知宏 (正会員)

1991年慶應義塾大学大学院理工学研究科博士課程単位取得退学。東京工科大学助手、講師、助教授を経て、1997年より新情報処理開発機構並列分散システムアーキテクチャつくば研究室長、2002年より産業技術総合研究所グリッド研究センタークラスタ技術チーム長。2008年より同所情報技術研究部門インフラウェア研究グループ長。博士(工学)。並列処理、通信アーキテクチャに関する研究に従事。情報処理学会、IEEE CS 各会員。

田中 良夫 (正会員)

昭和40年生。平成7年慶應義塾大学大学院理工学研究科後期博士課程単位取得退学。平成8年技術研究組合新情報処理開発機構入所。平成12年通産省電子技術総合研究所入所。平成13年4月より独立行政法人産業技術総合研究所。現在同所情報技術研究部門主幹研究員。博士(工学)。グリッドにおけるプログラミングミドルウェアおよびグリッドセキュリティに関する研究に従事。IC 1999, HPCS 2005, SACSIS 2006 最優秀論文賞, 2006年度情報処理学会論文賞。ACM 会員。

関口 智嗣 (正会員)

昭和34年生。昭和57年東京大学理学部情報科学科卒業。昭和59年筑波大学大学院理工学研究科修了。同年電子技術総合研究所入所。情報アーキテクチャ部主任研究官。以来、データ駆動型スーパーコンピュータSIGMA-1の開発等の研究に従事。平成13年独立行政法人産業技術総合研究所に改組。平成14年1月より同所グリッド研究センターセンター長。平成20年4月より同所情報技術研究部門長並列数値アルゴリズム、グリッドコンピューティング、ITによる地球環境の理解に興味を持つ。市村賞、情報処理学会論文賞受賞。地理情報システム学会理事、日本応用数理学会評議員、情報処理学会、SIAM、IEEE 各会員。Open Grid Forum 諮問委員。