社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－MPS－59 (5)
2006／5／25

WS-Notificationを基盤とするグリッド上のアプリケーション連携システム

下坂 久司[†], 廣安 知之[††], 三木 光範[††]

[†] 同志社大学大学院 [††] 同志社大学工学部

近年，グリッド技術はWebサービス技術を基盤とした標準化が進められている．特に，Webサービスにおいて状態を持つリソースを定義できるWS-Resource Frameworkと，サービス間のメッセージ通知を実現するWS-Notificationは，グリッドに適したWebサービスの実現に必要不可欠である．本論文では科学技術計算用アプリケーションの連携に注目し，WS-Notificatitonを基盤とするグリッド上のアプリケーション連携システムを提案する．またGlobus Toolkitを用いて提案システムを実装し，これをApplication Igniting Systemと呼ぶ．構築したシステムでは，科学技術計算用のアプリケーションを容易にWebサービス化でき，サービス間のメッセージ通知によるアプリケーション実行の連鎖を実現する．さらに，エンドユーザによる柔軟なアプリケーション連携の設計を支援する複数のサービスを提供する．今後の課題として，障害への対処方法の検討などが挙げられる．

Application Integration System on the Grid based on WS-Notification

Hisashi SHIMOSAKA[†], Tomoyuki HIROYASU[††] and Mitsunori MIKI[††]

[†] Graduate School of Engineering, Doshisha University
[††] Knowledge Engineering Dept., Doshisha University

Recently, Grid technologies have been standardized based on Web service technologies. Of these technologies, the WS-Resource Framework and WS-Notification are indispensable to construct Grid-enabled Web services. The WS-Resource Framework enables the definition of resource with state information and the WS-Notification realizes message notification driven by state change. This paper focuses on integration in scientific applications. We propose a new application integration system on the Grid based on WS-Notification. The proposed system, called "Application Igniting System", is implemented using the Globus Toolkit. In this system, scientific applications can be published easily as Web services and the chain of application invocation is realized using message notification. In addition, this system provides services, which supports the design of flexible application integration. We will discuss the coping strategy when an error is detected.

1 はじめに

自動車や航空機の設計，バイオインフォマティクスなどの複合領域にわたる問題解決では，異なる人や組織から提供される複数の科学技術計算用アプリケーションを，広域ネットワークを通じて統合利用できることが望まれる．広域ネットワークを利用するシステムの構築には，グリッドの利用がシステムの実用性や開発コストを考慮した場合，必要不可欠である．最近では，Global Grid ForumにおいてOpen Grid Services Architecture (OGSA)[1]が提案されており，Webサービス技術を基盤としたグリッド技術の標準化が進められている．OGSAはWS-Resource Framework (WSRF)[2]とWS-Notification (WSN)[3]の2つの仕様を基盤としている．WSRFはWebサービスに状態を有するリソースを導入するための仕様である．また，WSNは状態変化により駆動する登録型の非同期メッセージ通知の枠組みである．

本研究では，グリッド上で科学技術計算用アプリケーションを複数繋ぎ合わせることにより新しいシステムを構築でき，問題解決を行える基盤システムの開発を目標とする．アプリケーションを複数繋ぎ合わせて利用することを，本研究ではアプリケーション連携と呼ぶ．このようなグリッド上のアプリケーション連携を実現するために，本研究で構築するシステムは既存アプリケーションをWebサービスとして取り扱う．またWebサー

ビスにおいてアプリケーションの実行状態を取り扱うことで，状態変化により駆動するメッセージ通知を用いたアプリケーション実行の連鎖を実現する．さらに，エンドユーザによる柔軟なアプリケーション連携の設計を可能にする複数のサービスを提供する．

2 Application Igniting System

本研究では WSN を基盤としたグリッド上のアプリケーション連携システムを提案し，Globus Toolkit (Globus) を用いて実装する．実装したシステムでは，ある 1 つの状態変化を起点としてサービス間で次々とメッセージが通知され，複数のアプリケーションが連鎖して実行される．これらはアプリケーション実行という火が，グリッド上で燃え広がるようであるため，本研究では実装したシステムを Application Igniting System と呼ぶ．

2.1 Application Igniting System の概要

Application Igniting System の概要を図 1 に示す．本システムでは，Application Igniting Service (AI サービス) および Application Igniting Factory Service (AI Factory サービス) が広域ネットワーク上に点在する．アプリケーション所有者は AI Factory サービスにアプリケーション起動用の情報を登録する．登録されたアプリケーション情報は，Globus が提供する Index サービスに集められ，エンドユーザに提供される．エンドユーザは利用したいアプリケーションを複数選択し，ワークフローとデータフローを設計する．設計されたアプリケーション連携は，AI サービス間のメッセージ通知により実現される．ここで，アプリケーションの起動は Globus が提供する WS-GRAM (Grid Resource Allocation and Management) サービスを，アプリケーション間の入出力ファイル交換は RFT (Reliable File Transfer) サービスを介することで実現される．

2.2 アプリケーションの登録と実行

2.2.1 RSL を用いたアプリケーションの登録

アプリケーション所有者は，WS-GRAM サービスにジョブをリクエストするための RSL (Resource Specification Language) を用いてアプリケーション情報を記述し，AI Factory サービスに登録する．本システムでは，入力ファイルのパス情報などのアプリケーション起動に必要な情報の一部をシステム自身が用意するため，RSL ファイルを記述するアプリケーション所有者に対し，必要な情報を提供する必要がある．そのため，「AIS_」で始まる約 20 個程度の変数を本システムは用意している．

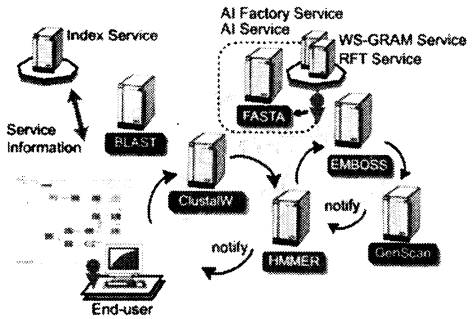

図 1: Application Igniting System の概要

アプリケーション所有者はこれらの変数を用いて，入力ファイルのパス情報などをシステムから取得する．またアプリケーションの出力ファイルを，システムが定めるパス上に配置することで，他のアプリケーションとの入出力ファイル交換を実現できる．

2.2.2 リソースの生成とポートタイプ

エンドユーザは Index サービスに収集されたアプリケーション情報を参照し，利用したいアプリケーションを複数選択する．その後，各アプリケーション情報を保持する AI Factory サービスに対し，リソースの生成を要求する．AI Factory サービスは登録されたアプリケーション情報に基づくリソースを生成した後に，リソースにアクセスするための情報を返信する．エンドユーザは返信された情報をもとに，AI サービスの提供するポートタイプを利用して，後述するワークフローおよびデータフローの設計を行う．AI サービスが提供する代表的なポートタイプとその機能は以下の通りである．

- setSubscription: 引数で指定された Producer の「CONDITIONAL BRANCH TOPIC (CBT)」に通知予約する．
- createTransferResource: 引数で指定された入出力ファイルを送受信するための，RFT サービスのリソースを生成する．
- addTransfer: 引数で指定された他のアプリケーションの出力ファイルを，アプリケーション実行前に取得するファイルのリストに追加する．

また AI サービスの各リソースは，アプリケーションの実行状態を取り扱うリソースプロパティを保持しており，「APPLICATION INVOCATION TOPIC (AIT)」として公開している．

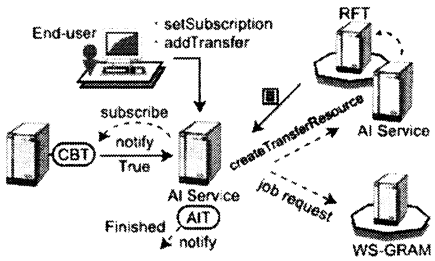

図 2: アプリケーションの起動手順

2.2.3 メッセージ通知によるアプリケーション実行

AI サービスはリソースに格納されている情報を利用し，メッセージ通知によりアプリケーションを起動する．アプリケーションを起動する際の手順を図 2 に示す．エンドユーザは，あらかじめ setSubscription ポートタイプを利用してメッセージ通知元への通知予約を指示しておく．また addTransfer ポートタイプを利用して，アプリケーションの入力ファイルとなる他のアプリケーションの出力ファイルを指定する．これにより，メッセージ通知元から「True」のメッセージを受け取ることで，AI サービスはアプリケーションを起動する．アプリケーションの起動は次の手順により構成される．まず，AI サービスはエンドユーザにより指定された他のアプリケーションの出力ファイルを取得する．これは各出力ファイルを保持する AI サービスの createTransferResource ポートタイプを利用して実現される．全てのファイル転送終了後，リソースに格納されている RSL ファイルの変数を置換し，WS-GRAM サービスにアプリケーション実行をリクエストする．アプリケーション実行が正常に終了した場合，アプリケーションの実行状態を取り扱うリソースプロパティの値を Finished に，異常終了した場合は Failed に更新する．これにより，新たなメッセージ通知が駆動する．

2.3 アプリケーション連携の設計

2.3.1 ワークフローの設計

Application Igniting System において，アプリケーションの実行順序はメッセージ通知によるアプリケーション実行の連鎖により決定される．AI サービスの各リソースはアプリケーションの実行状態を取り扱うリソースプロパティを保持しており，トピック AIT として登録している．また，後述するワークフロー設計を支援するサービス群の各リソースは，すべて条件の真偽値を保持するリソースプロパティを保持しており，トピック CBT として公開している．

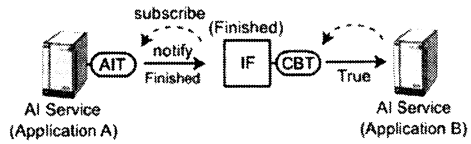

図 3: アプリケーションの逐次実行

逐次実行と条件分岐: IF サービスは，任意の Producer の AIT もしくは CBT のいずれかのトピックに通知予約を行えるポートタイプを保持する．このポートタイプの引数には条件値を指定することができ，通知されたメッセージが条件値と一致する場合に，条件の真偽値を保持するリソースプロパティの値を True に更新する．また一致しない場合には False に更新する．IF サービスを用いたアプリケーション逐次実行の例を図 3 に示す．エンドユーザはあらかじめ IF サービスのリソースを生成し，アプリケーション A に通知予約するよう指示する．その際，条件値として Finished を指定しておく．また，アプリケーション B に IF サービスへ通知予約するよう指示する．これにより，アプリケーション A の実行が正常終了し IF サービスにメッセージ Finished が通知されると，連鎖的にアプリケーション B にメッセージ True が通知される．通知によりアプリケーション B が実行される．

並列実行と同期処理: メッセージ通知は通知予約を行った Consumer 全てに通知されるため，複数のアプリケーションにメッセージ True を通知することで，アプリケーションの並列実行を実現できる．一方で，並列に実行しているアプリケーションの同期をとって，次のアプリケーションを実行できる機能が必要である．このような同期処理を実現するために AND サービスが提供される．AND サービスは 2 つの Producer に通知予約を行える特徴を持つ．メッセージ True を受け取った際には，メッセージ送信元の Producer を記録しておき，2 つの Producer 両方からメッセージ True を受け取った時点で，条件の真偽値を保持するリソースプロパティの値を True に更新する．

繰り返し: 特定のアプリケーション連携の繰り返し実行をサポートするために，LOOP サービスおよび OR サービスが提供される．OR サービスは AND サービスに似た機能を提供するが，両サービスの異なる点として，OR サービスは 2 つの Producer

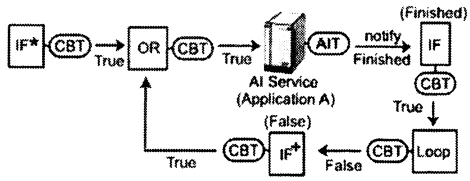

図 4: アプリケーションの繰り返し実行

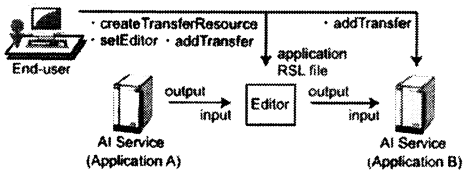

図 5: 出力ファイルの変換

のいずれかからメッセージ True を受け取ることで，リソースプロパティの値を True に更新する．LOOP サービスは，OR サービスを補助的に用いて繰り返し実行を実現する．LOOP サービスはメッセージ True を受け取った回数をカウントしており，エンドユーザがあらかじめ指示した回数に満たない場合にはリソースプロパティの値を False に，同数の場合には True に更新する．アプリケーション A を繰り返し実行する例を図 4 に示す．エンドユーザはあらかじめ 6 つのリソースを生成後，図 4 に示す矢印の逆向きに通知予約を設計する．

2.3.2 データフローの設計

入出力ファイルの交換: エンドユーザを含めたアプリケーション間の入出力ファイル交換は，AI サービスのポートタイプを利用して行われる．エンドユーザが入力ファイルを用意した場合，該当するアプリケーション情報を保持する AI サービスの createTransferResource ポートタイプを利用し，ファイル転送を実現する．一方で，あるアプリケーションの出力ファイルを別のアプリケーションの入力ファイルとする場合，後者のアプリケーション情報を保持する AI サービスに入出力ファイルの組み合わせを指示する．これには addTransfer ポートタイプを利用する．

出力ファイルの変換: 出力ファイルの変換を実現するために，Editor サービスが提供される．Editor サービスが提供するポートタイプや機能は AI サービスとほぼ同等であるが，両サービスの異なる点として，アプリケーションの取り扱いが挙げられる．Editor サービスはエンドユーザが用意した変換用のアプリケーションを，同じくエンドユーザが用意した RSL ファイルを用いて起動する．エンドユーザが用意したアプリケーションは，Editor サービスが提供する createTransferResource ポートタイプを用いて転送する．また RSL ファイルを登録するために，新たに setEditor ポートタイプを用意しており，RSL ファイルを読み込んだデータを引数にして利用する．

図 5 に，アプリケーション A の出力ファイルを変換して，アプリケーション B の入力ファイルとする手順について示す．図 5 において，エンドユーザはあらかじめ変換用のアプリケーションを作成する．また，作成したアプリケーションを起動するための RSL ファイルを用意する．これらを createTransferResource および setEditor ポートタイプを利用し，Editor サービスに転送する．その後，addTransfer ポートタイプを利用して，アプリケーション間のファイル交換を指示する．

3 まとめと今後の課題

本研究では，グリッド上でアプリケーションを複数繋ぎ合わせることにより新しいシステムを構築できる基盤システムの開発を目標として，WS-Notification を基盤とするグリッド上のアプリケーション連携システムを提案した．また提案するシステムを Globus を用いて実装し，Application Igniting System を構築した．構築したシステムは，アプリケーション所有者に対して RSL を用いたアプリケーション登録方法を提供し，メッセージ通知によりアプリケーションを起動する．アプリケーション利用者に対しては，アプリケーション連携の設計を支援する複数のサービスを提供する．設計されたアプリケーション連携は，サービス間でメッセージを連鎖して通知することにより実現される．今後の課題として，障害を検知した際の対処方法の検討などが挙げられる．

参考文献

1) Foster,I. and et.al.: The Open Grid Services Architecture, Version 1.0, Global Grid Forum OGSA-WG, GFD-I.030 (2005).

2) Czajkowski,K. and et.al.: The WS-Resource Framework, Version 1.0, (2004). http://www.oasis-open.org/committees/download.php/6796/ws-wsrf.pdf

3) Graham,S. and et.al.: Publish-Subscribe Notification for Web services, Version 1.0, (2004). http://www.oasis-open.org/committees/download.php/6661/WSNpubsub-1-0.pdf