

Globus 上における統一的なリソース管理機能の考察

Consideration of the Unified Resource Management on Globus

金光 永煥†
Hidehiro Kanemitsu

浦野 義頼‡
Yoshiyori Urano

あらまし 分散並列処理環境や Grid 環境において、実行される各ジョブの状況、CPU、メモリなどのリソースの状態を監視することは重要な要素である。本研究では、各ノードのリソース情報だけでなく、特定のジョブ、プロセスの状態を一元的に管理する機能、及びノードの動的発見からジョブ割り当てまでの処理を Globus Toolkit4 上で新規に実装・検証することを目的とする。本研究における特定ジョブ、プロセスの一元管理機能の実装には、WSRF/OGSA 上で新規 Grid サービスを実装し、各ノードを配備することが必要である。ノードの動的発見には Jini による実装を行い、Grid サービス用プロセスとのローカルにおける通信によってオブジェクト共有を図り、負荷耐久度の面において検証を行う。

キーワード: Grid、リソース管理、WSRF、JMX

1. まえがき

複数の計算機による並列処理（または分散並列処理）は、従来から主に科学計算分野において採用されている。この場合、各計算機におけるプロセッサ間通信は高速なものでなければならない。しかし、近年のインターネット環境のブロードバンド化に伴い、Network 間または Network 内での通信速度は非常に高速なものとなり、広域分散環境においても並列処理に必要な要件を満たしつつある。そのため、広域な範囲に存在する計算機のリソースを利用して計算処理を行う形態、特に Grid コンピューティングが、さらに普及していくものと考えられる。Grid コンピューティングでは、複数の計算機に対して特定のジョブを割り当てるため、各計算機には、ジョブリクエストを受信するための統一化されたインタフェースを備えていることが望ましい。そのため、2004 年 1 月において、GGF (Global Grid Forum) によって WSRF (Web Service Resource Framework)[1]といった、Web サービス技術を基盤とした Grid 環境構築のための仕様が提案された。本論文では、各計算機（以下、単に Node と呼ぶ）のリソース状況に合わせてジョブ割り当てを行うために、WSRF 実装である Globus Toolkit4 (Globus Alliance)[2]上で、各 Node のリソース管理および Node の動的発見を行う方法について記述する。

2. 関連分野

本研究の目的は、Grid 環境を想定した各 Node の動的発見及びリソースの統一管理を通じて、特定ジョブ適正な割り当てを行うことである。これらの目的に関連したものとして、「Grid サービスの発見」及び「リソース情報の管理」という観点で調査・検討した。ここで Grid サービスとは、Web サービスインタフェースで提供された各種処理であると定義する。「Grid サービスの発見」においては、OGSI-UDDI Bridge[3]が提案されている。これは、OGSI 準拠となるインタフェースを実装した Bridge を介して Grid サービスを登録するものである。また、UDDI の機能を拡張して Grid サービスの Service Leasing や UDDI Node 間での Data Replication を実現した UDDIe[4]が

提案されている。これは、Grid サービスに対し、クライアントによるサービス利用可能な期間、利用不可である期間をあらかじめ設定することにより、より柔軟な Grid 環境を構築することを目的とするものである。

「リソース情報の管理」においては、例えば、Globus の主要コンポーネントの一つである、MDS (Monitoring and Discovery Service)[5]がある。これは、主に「Node のローカルなリソース情報の蓄積」及び「各 Node から報告されるリソース情報の蓄積」という機能に分かれている。これにより、指定されたリソース条件に合致する Node の検索が可能となる。本論文においては、これら 2 つの観点から、「Node の数が動的に変化する環境」に適した機能を提供するための手法を提案する。

3. 本研究の目的

本研究では、「2. 関連研究」において記述した従来の手法による機能に加えて、より動的な性質を持つ機能を実装する。例えば、OGSI-UDDI Bridge や UDDIe では、主に Grid サービスの検索を主目的としている。この目的のためには、サービス提供者があらかじめレジストリに登録しておかなければならない。また、クライアントは、レジストリに対してサービス検索のためのリクエストを行う必要がある。本研究では、サービス提供者、サービス利用者が特定のグループネットワークに属し、サービス提供者がネットワークに「参加」したことをサービス利用者に対して通知を行う機能を実現した。これにより、Grid サービスだけでなく、Node 自体のネットワークへの参加/切断といった状態を通知することが可能となる。また、MDS において各 Node 内で管理される情報は、CPU、メモリ、OS、ファイルシステムといった情報である。本研究では、これに加えて、任意のアプリケーションプロセスを管理する機能を実装する。これにより、Grid 環境において、Node 自身のリソース情報に加えて、Node 内のアプリケーションの状態を管理することが可能となる。

4. システム概要

この章では、本研究で提案されるシステム及びシステム内で適用される手法について説明する。

4.1 想定する環境

本研究においては、第一にジョブ実行可能 Node の数が動的に変化するネットワークを想定する。第二として、

† 早稲田大学大学院国際情報通信研究科修士 2 年

‡ 早稲田大学大学院国際情報通信研究科

各 Node には、Java VM、Grid サービスコンテナがインストールされていることを想定する。

4. 2 設計・実装

4. 2. 1 Node 内におけるリソース管理

この節では、ジョブの実行を行う各 Node 内のリソース情報を取得・管理する機構について記述する。Node 内の管理対象である主なリソース情報は図1のとおりである。

管理対象情報名	目的
物理メモリサイズ情報	メモリ空き容量の調査
仮想メモリサイズ情報	プロセス実行に利用可能な仮想メモリサイズの調査
Java VM に割り当てられている CPU 時間	ノード内における Java VM が占める CPU 使用率の調査
スワップサイズ	メモリアクセス速度の調査
スレッド情報	各スレッドの種類、情報の調査
現在の CPU 稼働総時間	CPU 使用率の調査
ヒープ/非ヒープメモリ情報	Java VM が使用するクラス、スレッドが共有するメモリサイズの調査
メモリプール情報	メモリリークの調査
GC 情報	ガーベッジコレクションと Node への負荷の関係の調査

図1. Node 内で取得・管理される主な情報

これらの情報を、情報収集用 Node へ定期的に報告することにより各 Node のリソース情報は、情報収集用 Node 内で一元的に管理することができる。また、JMX (Java Management Extensions)[6] の機能を用いることにより、各 Node 特定のリソース状況に達すると情報収集用 Node へ通知することが可能となる。図2に、Node 内でのリソース情報管理処理の概要を示す。

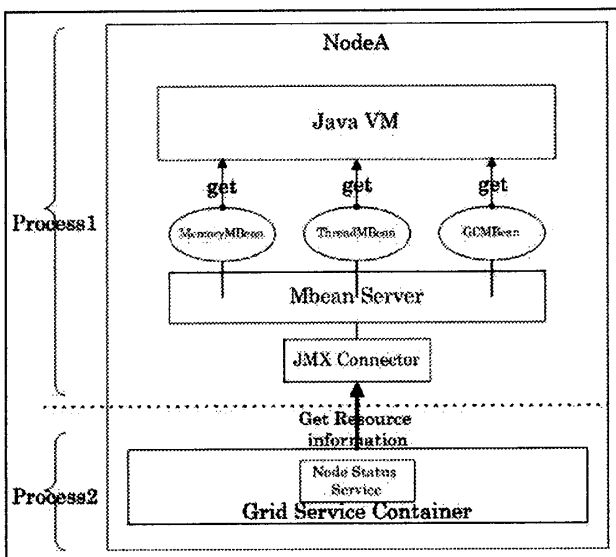


図2. Node 内での情報取得の概要図

Grid Service Container 内に、Node のリソース情報取得のためのサービスインタフェース(NodeStatusService)を配備する。NodeStatusService は、異なるプロセスで稼働している MBeanServer に対し、図1で記した各種情報を取得

する。これらの情報はそれぞれ MBean オブジェクトとして MBeanServer 内に管理される。また、特定の条件(例えば、スレッド数 20 かつ使用可能メモリサイズが 20MByte 以上)に達すると、MBeanServer から、同じ端末内のかつ MBeanServer とは異なるプロセスである Grid Service Container 内に存在する NodeStatusService に通知され、その直後に情報収集用 Node に対して通知する。この通知は WSRF による通知機能(WS-Notification)によって行われる。これにより、Node 間の通信インタフェースは WSRF で統一的行われる。

4. 2. 2 情報収集用 Node におけるリソース管理

情報収集用 Node(以下、Aggregator Node と呼ぶ)は、ジョブ実行 Node のリソース情報を管理するための Node である。Aggregator Node は、各 Node の識別情報(Host 名、IP Address など)、図1で記したリソース情報、及び各 Node で WSRF における Grid サービスクライアントとしての Daemon プロセスを管理する。図3に、Aggregator Node が行う各 Node のリソース情報管理の概要を示す。

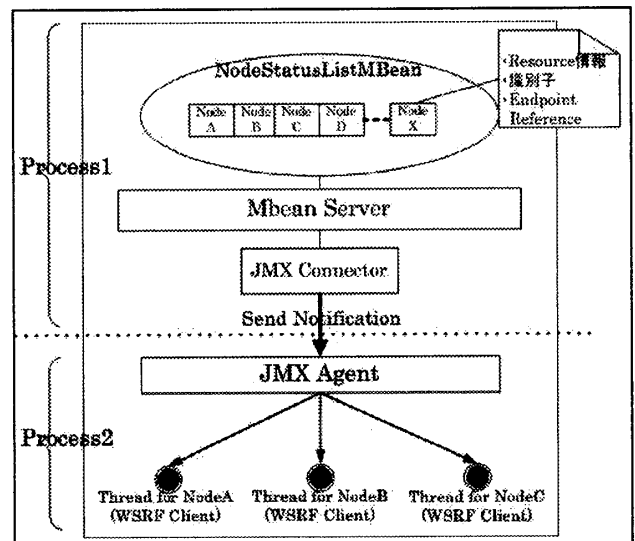


図3. Aggregator Node による情報管理処理の概要図

Aggregator Node は、各 Node のリソース情報を管理するための NodeStatusListMBean、MBean Server、JMX Agent オブジェクト、そして各 Node との WSRF による通信のための Daemon スレッドから構成される。さらに NodeStatusListMBean は、各 Node のリソース情報のリストを管理している。これにより、新規の Node のリソース情報が追加されると、JMX Agent は通知を受け取り、その Node に対して WSRF による通信を行うためのスレッド(Thread for Node X)を新規に起動させることができる。WSRF による通信においては、JMX Agent が NodeStatusService より Endpoint Reference を取得する。その後、スレッドを起動し、WS-Notification によって各 Node から通知と共にリソース情報を受信する。受信した各 Node のリソース情報は、履歴情報として NodeStatusListMBean 内に蓄積される。図4に、ジョブ実行 Node と Aggregator Node との WSRF による通信の概要図を示す。JMX Agent は、NodeStatusListMBean に新規 Node が追加されたことを通知としてローカル内で受信後、その識別子(IP Address、

ホスト名)をキーとして WSRF Client としてその Node の NodeStatusFactory サービスへアクセスし、Resource Property を生成する。その後、JMX Agent はリソース情報通知用のクライアントとしてのスレッドを起動する。

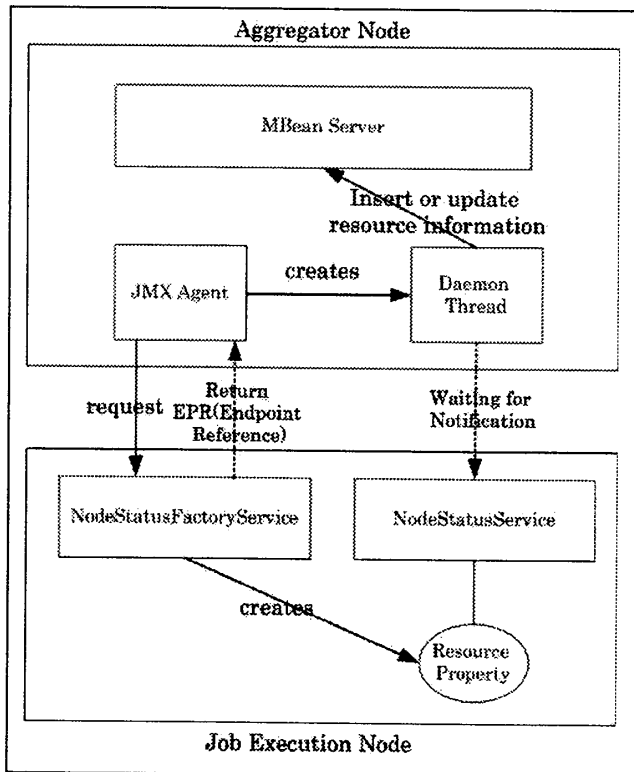


図4. WSRFによるリソース情報取得処理の概要図

4.2.3 Nodeの動的発見

これまでの処理を行うためには、Aggregator Node があらかじめジョブ実行 Node の URI を保持していなくてはならない。本研究では、Node の発見に、Jini を適用する。図5に、Jini ネットワークを利用した Node 発見の処理を示す。

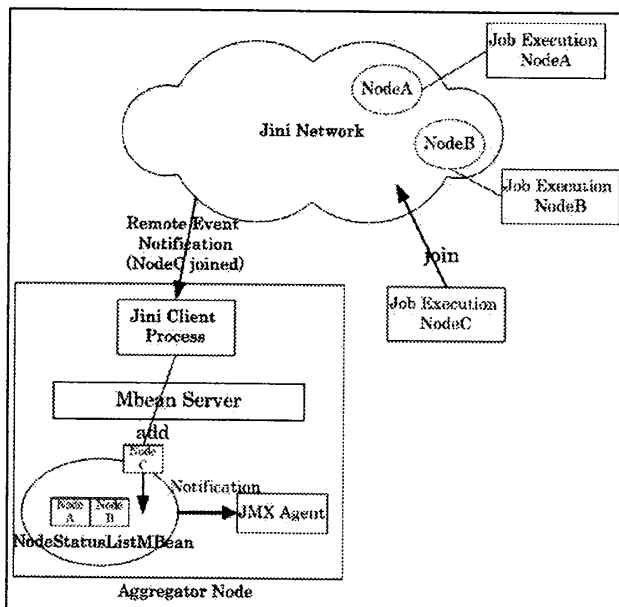


図5. Jiniを利用したNodeの発見

Aggregator Node は Jini クライアントプロセスを起動させ、ジョブ実行 Node が Jini ネットワークに参加したことを、リモートイベントとして Jini ネットワークからの通知を受信する。通知を受けた Aggregator Node は、MBean Server に新規参加 Node の識別情報(IP Address、ホスト名)を追加する。新規 Node 情報が追加された MBean Server は、JMX Agent に対して通知を送信することにより、JMX Agent は前節の処理へ移行することができる。

5. むすび

本研究の結果、Jini による Node の発見により、ジョブを動的に割り当てることが可能となった。また、Node の Jini ネットワークへの参加 / 切断を監視することにより、Node の数が変化し得る環境において適切なリソース情報をクライアントへ提供することが可能となった。今後は、実行中のジョブに対する統一的な管理手法について研究する予定である。

文 献

- [1] Karl Czajkowski, Donald F. Ferguson, Ian Foster, Jeffrey Frey, Steve Graham, Igor Sedukhin, David Snelling, Steve Tuecke and William Vambenepe, "The WS-Resource Framework", 2004.
- [2] "Globus Toolkit 4.0 Release Manuals", <http://www.globus.org/toolkit/docs/4.0/>.
- [3] John Colgrave, "UDDI and OGIS", 2003.
- [4] Ali ShaikhAli, Omer F. Rana, Rashid Al-Ali, David W. Walker, "UDDIe: An Extended Registry for Web Services".
- [5] H'el'ene N. Lim Choi Keung, Justin R.D. Dyson, Stephen A. Jarvis, Graham R. Nud, "Performance Evaluation of a Resource Monitoring and Discovery Service for the Grid", 2003.
- [6] Sun Microsystems, Inc., "Java™ Management Extensions Instrumentation and Agent Specification, v1.2", 2002.