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

2008－HPC－118 (4)
2008／12／16

グリッドミドルウェアの階層的管理機構による NAREGIインストーラの設計と実装

壬生亮太 †，小林泰三 †，大庭淳一 †，高見利也 †，上田将嗣 †，
山元祐介 †，増原裕之 †，天野浩文 †，青柳睦 †

† 九州大学

グリッドなどの分散した計算機をミドルウェアによって繋いだシステムでは例え最小構成であっても管理者の操作は煩雑である．ミスなく手作業で大規模なシステムの構築は事実上不可能であるため，我々は分散したソフトウェアの階層的管理を実現する機構を提案する．これは複雑な管理作業を分類し，ノード毎に提供するサービスを抽象化してシステム全体を俯瞰する階層と各ノード内で個々のソフトウェアを扱う階層に切り分けるものである．これに基づき，ノード内の管理に apt/yum-rpm を用い，NAREGI グリッドミドルウェアの管理作業の自動化と高い拡張性を達成した NAREGI インストーラの設計と実装について紹介する．

Implementation of NAREGI Installer based on a Hierarchical Design of Grid Management

Ryota MIBU † Taizo KOBAYASHI † Jun'ichi OOBA † Toshiya TAKAMI † Masatsugu UEDA †
Yusuke YAMAMOTO † Hiroyuki MASHIHARA † Hirofumi AMANO † Mutsumi AOYAGI †
† Kyushu University

Administrative operations in Grid are much complicated even in the smallest system since various functions in distributed computers are connected through middleware. Administrators must setup large distributed components manually into the system without any mistake throughout the long and winding installation/configuration process. A hierarchical management system of Grid middleware which configures grid services and functions is designed as the two layers: Inter-node layer and Intra-node layer. Based on the design, the NAREGI Installer is implemented with a carefully-designed apt/yum-rpm tree, by which automated management and high extendability are achieved.

1 グリッドの現状

グリッドの実現に向けグリッドミドルウェアが研究開発され，TeraGrid, EGEE など様々なグリッドが実現し始めている．しかしながら，管理運用面での課題はあまり解決されていない．グリッドのような分散した計算機 (ノード) をミドルウェアによって繋いだシステムでは管理者の操作は複雑である．管理者は分散配置された計算機にそれぞれの機能に合ったミドルウェアをインストール及び設定しなければならない．グリッドミドルウェアは複数のソフトウェアにより構成されており，さらにそれらのソフトウェアにはインストール又は動作のための依存関係が存在し，例え最小構成であってもその配置及び設定は複雑である．以下，NAREGI グリッドミドルウェアを例に具体的に説明する．

National Research Grid Initiative (NAREGI) とは，日本におけるサイエンスグリッドの実現を目指したグリッドミドルウェア開発プロジェクトである [1]．NAREGI グリッドミドルウェア Beta2 では，8 機能 13 ノード構成のインストールに 4,529 のコマンドと約 5ヶ月の期間を要していた．これには xml などの設定ファイルの編集が 119 回含まれており，"/C=JP/O=kyushu-u/OU=aolab/CN=host/hoge.ao.kyushu-u.ac.jp" のような LDAP の Distinguished Name (DN) の入力も繰り返し手作業で行われていた．この

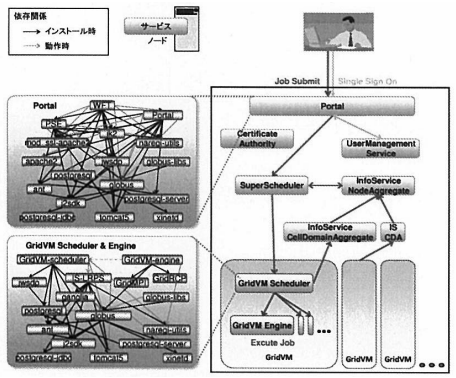

Fig. 1　NAREGI のノード構成とミドルウェアの依存関係

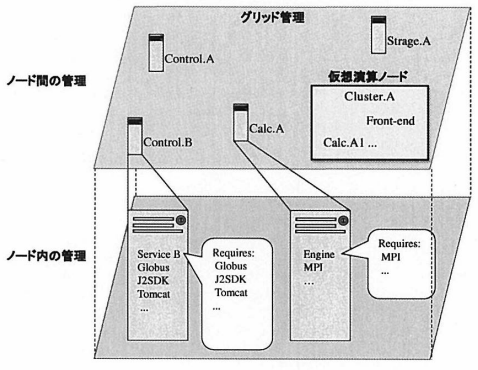

Fig. 2　グリッドミドルウェアの管理における階層.

NAREGI グリッドミドルウェアは, 2008 年 11 月現在 Ver1.1 が公開されており [2], 13 のサービスが 200 程の RedHat Package Manager (rpm) で構成されている.

規模の拡大に応じてこれらミドルウェアの管理作業は増々莫大で煩雑なものなる. そのため, インストールをはじめとするグリッドミドルウェアの管理は多大な知識と時間を必要とすることになる. さらに手作業でこれらの膨大な作業をミスせずに行うことは不可能に近い. 作業のどこかで人為的ミスが発生し, その発見と修正にさらなる労力を費やすことになる. これはグリッドのスケーラビリティを著しく低下させる本質的な問題である.

一つの計算機内でもソフトウェア間の依存関係の解決は非常に難しく, Linux では Advanced Packaging Tool (apt) / Yellow dog UpdaterModified (yum) - rpm などのパッケージ管理機構が提供されており, これにより幅広いユーザが使えるものとなっている. このようなユーザフレンドリな管理機構がグリッドの普及及び大規模システムの実現に必要である.

グリッドミドルウェアの管理が複雑である要因は, これらの複数のノードにまたがるミドルウェアとソフトウェアの依存関係を無階層に扱おうとしている点にある. そこで, 分散システムにおけるソフトウェアの階層性に着目する. グリッドの各機能は複数のノードに分散されており, 各ノードで提供される機能は様々なソフトウェアで実現されている. NAREGI では Portal, Information Service (IS), Super Scheduler (SS) などの機能を提供するノードがあり, その中の Portal ノードでは NAREGI ミドルウェア の Portal ミドルウェアの他に apach2、globus、j2sdk などのソフトウェアが必要である (Fig.1). したがって, 全体のノード構成とノード内ソフトウェアの依存関係で階層を切り分ける事ができる.

2　グリッドミドルウェアの階層的管理機構の導入

我々はグリッドミドルウェアの管理をこの階層性に基づき, グリッドシステムに分散した個々のソフトウェアに対する複雑な管理作業をノード間とノード内に分離した階層的な管理機構を提案する. これは, ノード毎に提供するサービスを抽象化しノード単位でシステム全体を俯瞰する「ノード間の管理」と各ノード内で個々のソフトウェアを扱う「ノード内の管理」に階層を分けている (Fig.2).

この管理機構ではノード間の管理はノード間の整合性を保ちノード構成を決定するものとし, 決定されたノード構成情報を基に各ノード内の管理が行われる. ノード間の管理で決定されるノード構成情報とは, ノードとサービスの関連付けがされグリッド全体を巨視的に見ることができる必要最小限の情報である. ノード間の管理では各ソフトウェアに関与しないという事が肝要である. ソフトウェアに対するすべての作業はノード内の管理として行われ, そのノードがシステムの一部になるよう設定される. 具体的には, グリッドミドルウェアのインストール, 設定, 削除, 更新, 及び起動停止

に加え, 他の必要なソフトウェアのインストールなどノード内での依存関係解決や証明書の配置などである.

この管理機構の実現のため, それぞれの階層での管理ツールは次のように設計される. ノード間の管理ツールは, グリッド全体のノード構成情報を管理し, その情報を全ノードで共有させ, 各ノード内の管理ツールを呼び出す. ノード内の管理ツールは, ノード構成情報を基にグリッドミドルウェアと必要な他のソフトウェアのダウンロード, インストール, 設定及び起動を行う. ノード間のファイルの受け渡しなどノードを跨ぐ作業が必要な場合はノード間の管理ツールが補助的に行う. このようにしてグリッド管理者はノード間の管理ツールからすべてのグリッドミドルウェアの管理作業を行う.

これにより, 次の 2 つが実現される. 一つは, 人為的ミスの削減である. 構成情報がノード間の管理ツールにより一元的に管理され, ノード内の管理ツールとの連携により全ノードで繰り返し入力されていた設定情報が機械的に各ソフトウェアの設定ファイルに書き込まれるため, 入力ミスを防ぐことができる. 残る一つは, 大規模化である. システム全体の管理を抽象化した必要最小限のノード構成情報で行うことでスケーラビリティが高くなっている。Cfengine[3] , puppet[4] といった分散システムの管理ツールや EGEE の LCFG[5] , YAIM[6] , TeraGrid の Bcfg2[7] といったグリッド管理ツールなどと同様に一つの作業ノードから複数のノードを集中管理するが, 各ノードの設定ファイルや詳細な情報を抱え込まない点で大きく異なる.

3 NAREGI インストーラの実装

NAREGI インストーラは我々の提案する管理機構に基づきノード間の管理ツールの構築作業部分を実装したもので, NAREGI グリッドミドルウェアの適切な rpm 化とノード内の管理ツール apt/yum によって構築作業を自動化している. これら rpm と NAREGI インストーラは NII にて公開されている [2] .

この NAREGI インストーラは sudo, ssh 及び scp を使った Bash スクリプトであり, 各ノード上で同名な管理ユーザを使い ssh 経由で sudo を用いて複数のノードを操作する. ノード構成情報は管理者が対話的に入力し, 共存できないサービスを一つのノードに指定していないか, 有効なホスト名であるか, OS の設定は正しいかなどの確認が行われ, 直ちに設定ファイルとして作業ノードから全ノードに配布される. この構成情報は主に VO 名, 各ノード情報 (サービスタイプ, ホスト名, IP, DN) であり, さらにいくつかのサービスについては通信用のポート番号を含んでいる.

このノード構成情報は 2 種類のファイルに分けられている. NAREGI では演算ノード群を仮想的に一つの計算機として扱うため, ノード間の階層を拡張し, 仮想計算機をノードとしてシステム全体を見るサブ階層と仮想計算機内のノード群を見るサブ階層に分ける. 2 種類のノード構成情報ファイルの内一つはコントロールノード群と仮想演算ノードとして GridVM Scheduler ノードの構成情報が書かれており, グリッドシステム全体の関連性が記述されたファイルである. 残る一つは仮想演算ノードとしてまとめられた GridVM Scheduler/Engine ノードの構成情報が書かれており, 個々の仮想演算ノード内の構成が記述されたファイルである. このように仮想ノードとして複数ノードの詳細をカプセル化し構成情報を分ける事で, さらに膨大なノードに対応することができるようになる.

ノード内の管理ツールとしては, 各 OS ディストリビューションで使われているパッケージ管理機構である rpm 及び rpm パッケージ管理ツールである apt/yum を採用している. rpm とは, Linux で広く使われているパッケージングシステムであり, ソフトウェアのインストール, 削除, 更新, 及び設定の自動化のみでなく, 様々な前後処理も自動化可能である. また, 他の rpm との依存関係 (Requires/Conflicts) が明記されている. apt/yum は rpm が持つ情報を利用して rpm のインストール、削除及び更新時に自動で依存関係を解決し, 必要な rpm を web 上の rpm tree からダウンロードしインストールなどノード内の整合性を保つ操作を行う. 信頼性のあるこれらの既存ツールを使うことで実装部分が減り開発の効率化が図れるが, 一方でこれらのパッケージングシステムの作法に従った適切なパッケージ化が必要となる. そこで我々は NAREGI グリッドミドルウェアの適切な rpm パッケージ化とそれぞれのパッケージ管理機構用に rpm tree の構築を行った. 各 rpm にはノード構成情報を基にグリッドミドルウェアを設定するスクリプトを同梱し, rpm の前後処理として実

行させている.

一方, NAREGI インストーラは証明書の管理/配布を行う機能を持ち, アーカイブからの証明書抜き出しと各ノードへの配置に加え, NAREGI CA によるテスト用証明書の発行も行うことができる. これはグリッドミドルウェアの管理ではなく, より一般的なグリッド管理である.

インストール作業はノード構成情報のみに基づき行われ, 管理者はメニューを選択するだけで作業を進めていくことができる. 自動化された各作業はインストーラ内でモジュール化されている. また, インストーラ内のコマンドはすべて抽象化されており, 各ノード上でのモジュール実行時に OS 毎に適切なコマンドとパッケージ管理ツール (apt/yum) が自動選択される. これにより OS や環境によるコマンドパスやオプションの差異を吸収し, 同じコードで NAREGI がサポートしている CentOS5, OpenSuSE10 に対応している.

実装したインストーラは単に構築作業を自動化したわけではなく, 拡張性を高めるため管理作業の適切な抽象化と適切な単位でのモジュール化がされており, メニュー構成の修正で様々な環境の構築が可能である. NAREGI は管理者の異なる複数サイトに跨がるシステムであることが想定されるため, NAREGI ミドルウェア Ver1.1.2 (予定) のインストーラではコントロールノード群と演算ノード群の分割インストール機能が拡張された.

4 効果

NAREGI グリッドミドルウェア Beta2 では, 8 機能 13 ノード構成のインストールに 4,529 のコマンドと約 5ヶ月の期間を要していた.

我々の実装した NAREGI インストーラにより, 管理者がコマンドを使用する作業は前準備として行われる次の作業となった: 全ノードについて一般的な OS インストール, ネットワーク設定, 時刻合わせ, 管理ユーザの用意, そして作業ノードへの NAREGI インストーラ rpm のインストールである. これらの準備作業後はインストーラを起動し, 管理者はノード構成情報の対話的な入力とメニュー画面に従う操作で, 次の構築作業をインストーラにより行う:

- 全ノードへの NAREGI インストーラ rpm のインストール
- OS 更新 (reboot も行う)
- ノード構成情報の共有
- 基本ソフトウェアのインストール
- 証明書の配布
- 各ミドルウェアのインストール
- rpm に同梱されたスクリプトによる設定
- サービスの起動及び NAREGI の初期設定 (IS ハンドルの作成及び配布)

このように大半の構築作業が自動化され, 8 機能 13 ノード構成のインストールはコマンドは数十に, 作業期間は 1 日に短縮された. これにより大規模なグリッドの構築が可能になったと言える. それだけでなく, ユーザは一般的な OS のインストールとネットワークの設定, そして NAREGI の適切なノード構成を決定できさえすれば, NAREGI グリッド環境の構築が可能となった事も重要である. また, rpm の更新により安定的にシステムを管理できるようになったことで NAREGI グリッドミドルウェアの開発効率も向上した.

一方で改善が必要な次のような問題点が挙げられる.

- セキュリティ: 作業ノードから全ノードにかけて同名の管理ユーザによる ssh 及び sudo をパスワード無しで実行できるよう設定する必要がある. また, ホストやユーザの認証について各サイトでのポリシーの違いに対応できない.
- エラー検出: 作業が正常に完了したかを画面またはログに出力される大量の情報から発見しなければならない.
- 逐次実行: 全作業を一ノードずつ逐次実行しており, 時間がかかる.
- 一括した操作: コントロールノード群と演算ノード群の分割操作は可能となったが, ノードの追加/切り離しなどノード毎の操作ができない.
- 動的な情報: 各ノードの状況など動的な情報を扱っておらず故障ノードへの対応ができない.

5 結論

グリッドなどの分散システムにおけるミドルウェアの管理をノード間の階層とノード内の階層に分け, システム全体に配置されるすべてのソフトウェアを階層的に管理する機構を提案した. この提案に基づき, ノード内の管理に既存のパッケージングシステムを活用したNAREGIグリッドミドルウェアの管理機構を設計した. そして, NAREGIグリッドミドルウェアの適切なrpmパッケージ化と共にこの管理機構の一部の機能としてNAREGIインストーラを実装した. このNAREGIインストーラは環境構築のための作業をほぼ自動化しており, 大規模な環境の構築及び多くの知識を必要としない操作が可能となった. また, 抽象化する事で階層による管理作業の切り分けを行い, 高い拡張性を実現した. 今後, 作業の並列化や構成情報からの必要最小限な設定情報の抽出などでさらに大規模なシステムの管理を可能にしていく予定である.

今回はグリドミドルウェアの管理機構の一部である環境構築ツールを実装したが, サービスの一括起動やrpmによる安定した更新の実現により, 運用を含めたNAREGIグリッドミドルウェアの管理ツールとしての可能性を示す事ができ, 階層概念導入が非常に有効であったと考える.

参考文献

1) S. Matsuoka and S. Shinjo and M. Aoyagi and S. Sekiguchi and H. Usami and K. Miura, *Japanese Computational Grid Research Project: NAREGI*, Proceedings of the IEEE, Vol.93, pp.522-533, 2005.

2) `http://www.naregi.org/`

3) Mark Burgess and Ricky Ralston, *Distributed resource administration using cfengine*, Software: Practice and Experience, Vol.27, pp.1083-1101, 2004.
`http://www.cfengine.org/`

4) *puppet introduction*,
`http://reductivelabs.com/trac/puppet/wiki/PuppetIntroduction`.
`http://reductivelabs.com/`

5) *The Complete Guide to LCFG*, 2005.
`http://www.lcfg.org/`

6) V. Buge and U. Felzmann and C. Jung H. Stadie and A. Vest, *Integrating the IEKP Linux cluster as a Tier-2/3 prototype centre into the LHC Computing Grid*, IEKP-KA/2006-3.
`http://yaim.info/`

7) Narayan Desai, Andrew Lusk, Rick Bradshaw, and Remy Evard *Bcfg: A Configuration Management Tool for Heterogenous Environments.*, In Proceedings of the 5th IEEE International Conference on Cluster Computing (CLUSTER03), pp.500-503, 2003.
`http://www.bcfg2.org/`