2004－HPC－99　(43)
2004／8／1

P2P フレームワークに対応したグリッド環境のためのモニタリングインターフェース

釘井 睦和 †，廣安 知之 ††，三木 光範 ††，谷口 義樹 †

† 同志社大学大学院　†† 同志社大学工学部

グリッド上の計算資源は広域に分散しており，通常複数のサイトをまたがってサービスが提供されることから，サイト管理者は不特定多数のユーザに対する責任を負うことになり，管理コストが増大する．本研究では多数のユーザが協調的にサイトの状況をモニタリングし，管理体制の良いモニタリングが行えるモニタリングインターフェースを提案する．提案インターフェースでは情報を一元管理するサーバを設けず，サイト間のデータのやりとりを P2P フレームワークである DNAS を用いて行う．本稿では提案システムの詳細や特徴を述べ，提案システムと既存のモニタリングソフトウェアとの比較を行い，その利点や問題点などについても述べる．

Monitoring interface for Grid environment in the P2P framework

Yoshikazu KUGII †　Tomoyuki HIROYASU ††　Mitsunori MIKI ††　Yoshiki TANIGUCHI †

† Graduate School of Engineering, Doshisha University

†† Knowledge Engineering Dept., Doshisha University

Usually, resources of computational Grid are existed widely over the Net. Administrators of each computation resource have to have responsibilities for many users who belong to the different institutions. Therefore, it is very difficult to administrate Grid resources and users. In this paper, we suggest "Monitoring interface for Grid environment in the P2P framework". In this framework, not only administrations but also login users can always check state of each resource. Therefore, this framework can reduce the cost of administrators. In the proposal system, the server that gathers information of each node is not prepared. Instead of preparing the server, the information of each node is transferred between the nodes by DNAS that is the P2P like system. In this paper, the characteristics and specifications of the proposed system are explained. At the same time, advantages and disadvantages of the system are described.

1　はじめに

近年，高速な計算機やネットワークの普及に伴い，広域ネットワークに接続されたあらゆる計算資源および情報資源を統合し，ユーザが手軽に利用できるような環境を目指すグリッド技術が注目されている [1]．グリッドは広域にまたがって存在する種々の資源を結びつけ，様々なサービスの構築を容易にする [2]．このようなグリッド技術の標準化が主に Global Grid Forum [3] を中心に検討されている．

グリッド技術によって，ユーザはどのサイトを利用するかを一切考慮せずにそれらが提供するサービスを享受することが可能になると期待されている．しかしながら，グリッド上のサイトは広域に分散しており，ユーザのタスクは通常複数のサイトをまたがって処理されることから，一部のサイトにシステムダウン等の問題があった場合にタスクの処理が中断される可能性がある．そのため，サイトの管理者は不特定多数のユーザに対する責任を負うことになり，管理コストは増大する．すでに計算資源のモニタリングツールはいくつか存在するが，これら主にサイト管理者が使用するためのものであり，管理体制の良い常時モニタリングを行うには管理者に大きな負担がかかる．本研究ではユーザがサイトの状況をチェックできるような P2P フレームワークに対応したモニタリングインターフェースを提案し，その特徴や詳細について述べる．そして実装したシステムの動作や機能について述べる．

2 グリッドコンピューティングとモニタリングシステム

2.1 グリッドコンピューティング

グリッドコンピューティングとは，広域に配置されたスーパコンピュータ，計算機クラスタ，ワークステーション，PC などあらゆる種類の計算資源や情報資源であるデータベースを接続し，高性能計算環境を提供するインフラストラクチャである．計算資源とは高性能な計算サーバや大規模なストレージを持つファイルサーバを指す．情報資源とは計算を行うために必要なデータやプログラムそのものである．グリッドは，大規模計算環境を提供するだけでなく，遠隔協調計算や多数のセンサによるリアルタイム処理，およびテラバイト級のデータベース処理なども提供できると考えられる．また今後，膨大なコンピュータシミュレーションを必要とする産業分野において，グリッドにより計算資源を得ることによって，開発コストの軽減や，開発期間の短縮を可能にすると予測される．

グリッドは，従来のコンピューティング環境に比べて次のような特徴を有する．そしてそれぞれの特徴は，様々な問題を含んでいる [4]．

- 多数の計算資源が存在する．

  今後はグリッド上で稼動する計算資源の数がますます増大すると考えられる．それに対してシステムはスケーラブルにその性能を発揮することが可能であることが望ましい．

- 計算は複数のヘテロな資源上で行われる．

  各計算拠点は等しい計算資源を持っているわけではない．中には高速な計算機，ネットワークを有している計算拠点もあれば，そうでない計算拠点もある．すなわちグリッドは多くの場合，非均質な計算資源から構成されることになるので，ジョブやタスクの計算資源への割り当て方をよく考慮する必要がある．また，アーキテクチャについても，それぞれの計算機において異なっており，アーキテクチャ的にヘテロな環境でアプリケーションを実行できるようにする必要がある．

- 計算は常にネットワークを介して行われる．

  遠隔地の計算拠点間で計算を行う場合，それをつなぐネットワークの構成により，高速にアクセスできる拠点とそうでない拠点が存在し，常に均質なネットワーク環境が得られないという問題がある．また，ネットワークを介する以上，通信データの改竄や盗聴などの問題が発生するため，認証などのセキュリティ技術などが必要となる．

- 計算資源の状態は動的に変化する．

  様々な障害により拠点までのネットワークが切断されたり，拠点自体が障害やメンテナンスのために停止したりするため，利用できる計算資源が動的に変化する．また，計算機の利用状態，CPU 負荷やメモリ利用率，通信のレイテンシなども常に変化するため，静的な計算資源の情報だけでは実行時間が予測できない．

2.2 グリッドにおけるモニタリングシステム

2.1 節で述べたように，計算資源の状態が動的に変化するような環境下ではグリッド資源を有効に活用するために，サイトに発生する障害からの迅速な復旧を行う必要があると考えられる．そのためには，物理的に分散した計算機を利用する環境において，アーキテクテャ，CPU，メモリ，ディスク，ジョブの状況，ミドルウェアやアプリケーションの動作状況を含む様々な情報を収集，管理および提供を行うシステムが必要不可欠となる．こうしたモニタリング情報は，ジョブスケジューラの判断材料となるだけでなく，グリッド環境のユーザや管理者，サポートスタッフにとってはデバッグや障害検知に有益な情報源になる．後者は，グリッド環境の耐故障性・自動管理の実現を目指す IBM の Autonomic Computing プロジェクト [5] に代表されるように，グリッド環境のモニタリングシステムが近年注目を集めている．

モニタリング情報の収集は，一般に全ての計算ノード上で行う必要があるが，その情報の管理，提供は全てのノードで行う必要はない．また，モニタリング情報を利用するアプリケーションは，基本的に情報収集を行うノード，管理・提供を行うノードとは独立に任意のノード上で実行され得る．また，グリッド環境では一つのノードが中央集権的にモニタリング情報を管理・提供することはほとんど不可能である．

一般的に，モニタリングシステムは以下の 4 つの

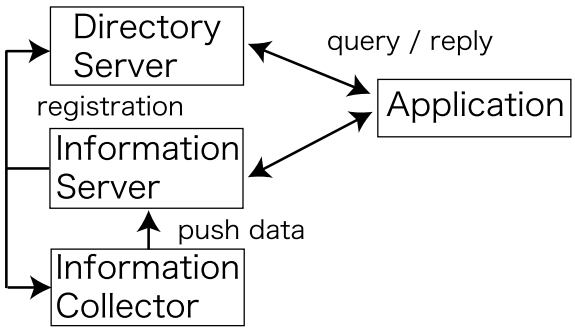

Fig. 1　モニタリングシステムの機能要素の関係

機能要素を少なくとも有する．この分類は Xuehai ら [6] によるものである．この機能要素の関係を Fig. 1 に示す．

- Information Collector
  各計算ノード上で，CPU，メモリ，ディスク，の使用率やネットワークの性能等のモニタリング情報を測定・収集する機能
- Information Server
  収集された情報を管理・提供する機能
- Directory Server
  Information Server が，どの Information Collector のどのような情報を管理・提供するかを管理・提供する機能
- Application
  Information Server の情報を利用して適切な動作をするアプリケーション機能

また，既存のグリッド環境向けのモニタリングシステムの実装としては次のようなものが挙げられる．これらは一つのノードで情報を管理・提供していることが特徴的である．

- Ganglia[7]
  Ganglia Cluster Toolkit (Ganglia) は Millennium Project の一部として California 大学の Berkeley Computer Science 部によって作成されたツールである．大規模に拡張可能なクラスタ監視，実行環境であり，履歴や傾向を Web 経由で視覚化することができる．
- NWS[8]
  UCSD で開発されたグリッド環境向けモニタリングシステム．モニタリング情報を利用した User Interface と呼ばれるモニタリング情報とそれに基づく将来の資源の利用予測を表示するアプリケーションを特徴とする．
- MDS[9]
  Globus Toolkit[10] で用いられるグリッド資源情報を管理・提供するフレームワークである．
- R-GMA[11]
  Globul Grid Forum で定義された Grid Monitoring Architecture(GMA) の EU Data Grid プロジェクトによる実装である．

3　P2P に対応したモニタリングインターフェース

3.1　設計目標

2 章で述べたようなグリッドにおけるモニタリングシステムに対して必要なインターフェースの要件を検討し，それらを設計目標とする．

- P2P フレームワークに対応
  既存のモニタリングシステムではノードの情報を一括に管理しているため，膨大な量のノードに対するスケーラビリティに限界があると考えられる．そのため，グリッドにおけるモニタリングシステムとして，今後は P2P の仕組みが利用されると考えられる．そのため，P2P に対応したインタフェースの提供が必要である．
- 主なグリッドモニタリングシステムに対応
  グリッド資源では何らかのモニタリングアプリケーションが動作していることが一般的である ganglia や MDS といった主要なモニタリングアプリケーションにも対応可能なインタフェースが必要となる．
- Web と携帯端末に対応
  ユーザビリティを考慮し，どこからでも利用可能なインターフェースを構築する．そのため，web や携帯端末の利用が必要である．
- サービスをインターフェース側で稼動
  インターフェースを利用しているユーザーが何らかのサービスを使用する場合，モニタリングシステム側においてサービスアプリケー

Table 1　得られる情報

静的な情報	動的な情報
ホスト名	負荷状況
HW 情報	メモリ使用量
OS 情報	ジョブの状況

ションを稼動させることが可能である．しかしながら，グリッド環境において，計算資源の管理者は別々であり，ユーザーが望むすべてのサービスアプリケーションを管理者が管理することは現実的ではなく．インターフェースを利用する側でアプリケーションを稼動させる必要がある．

3.2　構築したモニタリングインターフェース

構築したモニタリングインターフェースは現在，P2P フレームワークである DNAS，ganglia に対応している．

データの入力や表示，システムの操作のためのインターフェースには，アーキテクチャ非依存である Java アプレットを用いる．Java を用いることにより，PC や PDA，携帯端末などでクライアントを作成することが可能である．このことから，どこからでも情報を取得することが可能なモニタリングインターフェースを構築することが可能である．

クライアントソフトウェアを用いて，グリッド資源に接続後，たとえば，DNAS に接続する場合，近隣ノードの情報を取得し，視覚化してユーザに提供する．DNAS 上で動作するモニタリングアプリケーションによって収集される情報は，TAG を元に様々なフィールドに分類される．そのフィールドの一例を Table 1 に示す．静的な情報とは，比較的長い期間更新される可能性の少ない情報である．動的な情報とは，更新が頻繁に行われる情報である．

P2P に対応したインターフェースにより，膨大なノードの情報をすべて閲覧するのではなく，P2P 網への接続を切り替えることによってユーザが求める情報を検索するインターフェースとした．これにより資源が増大したとしても表示する範囲を移動していくことで，無限の資源を閲覧することが可能である．

作成した携帯端末クライアントソフトウェアの

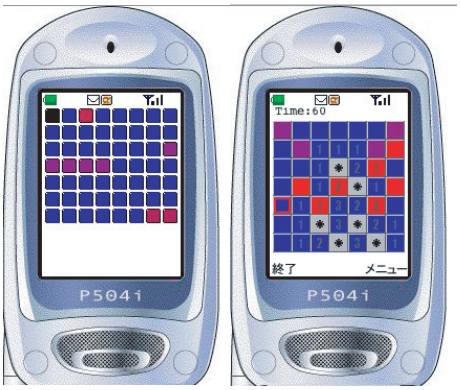

Fig. 2　携帯端末用ソフトウェア

表示画面を Fig. 2 に示す．ノードの負荷状況により色の異なるマス目が表示され，詳細な情報も閲覧することが可能となっている．

停止しているノードを発見して管理者に連絡するようなサービスを考える場合，そのアプリケーションはモニタリングシステム側に設置する．しかしながら，インターフェース側に設置することで各グリッド管理者の手を煩わすことなく，さまざまなアプリケーションを利用することができる．例えば，資源情報とゲームのデータをリンクさせたアプリケーションを作成することも可能である．ここでは，計算資源情報と連動したマインスイーパーを実装した．

4　動作確認

本節では，P2P フレームワークであるモニタリングインターフェースに提案インターフェースを接続し，動作確認を行う．

構築したシステムの動作，機能確認のための環境を以下に示す．同志社大学知的システムデザイン研究室が管理する複数の PC クラスタを用いて WAN と LAN が混在するような環境とした．

- Supernova クラスタ
  Dual Opteron 1.8GHz 256nodes
- Xenia クラスタ
  Dual Xeon 2.4GHz 64nodes
- Gregor クラスタ
  Dual PentiumIII 1GHz 64nodes

構築システムは，DNAS（Distributed Network Application System）[12] とモニタリングクライアントインターフェースから構成される．Fig. 3 にシステムの構成を示す．DNAS は，複数のノード（サイト）を再構成可能なツリー構造として扱い，

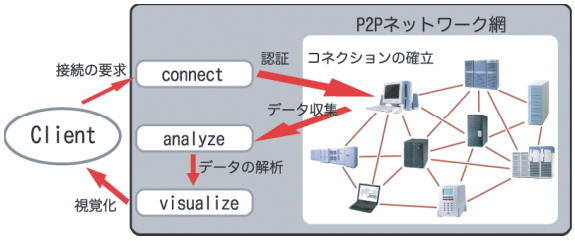

Fig. 3　提案システムの構成

ロードバランシングや効率的な情報の送受信を提供するための P2P フレームワークである．ツリー構造を用いることにより，通信の相手を階層によって明示的に制限することが可能であるため，ピュア P2P モデルの問題点であるネットワーク全体への負荷を軽減することができる．また，P2P 技術を利用することにより，近接したサイトの情報収集や異常の発見を中央サーバに依存することなく行うことができる．端末クライアントは DNAS から得られる情報を解析し，視覚化してユーザに提供する．

DNAS は，各ノードが再構成可能な論理的なツリー構造を形成し，このツリー構造を用いて通信する相手を制限することにより，従来の P2P フレームワークの問題点であるネットワーク全体への負荷を軽減することが可能となる．DNAS では，あるノードから見て上位にあるノードは uplink ノード，下位にあるノードは downlink ノードと呼ばれる．各ノードは，ユーザが自由に設定できる「レイヤー」の範囲に含まれるノードの情報を得ることができる．例として，Fig. 4（左）に，レイヤー制限を 1 にした場合に接続ノードから得られるノード情報の範囲を，Fig. 4（右）に，レイヤー制限を 2 にした場合に接続ノードから得られるノード情報の範囲を示す．この機能を利用して，ユーザはモニタリングする範囲をスケーリングすることができる．またモニタリングするユーザは，接続ノードを自由に変えることができる．

次に DNAS の情報交換の仕組みについて述べる．DNAS システムでは，まず各ノードで DNAS servent デーモンを起動しておき，DNAS servent デーモンは自分が直接接続しているノードで動作する DNAS servent デーモンと通信を行う．送受信される情報パケットは，各ノードのホスト名で始まり，コロンで区切られた TAG と DATA のペアで

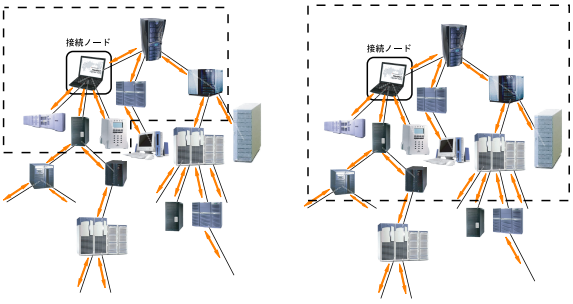

Fig. 4　レイヤー制限

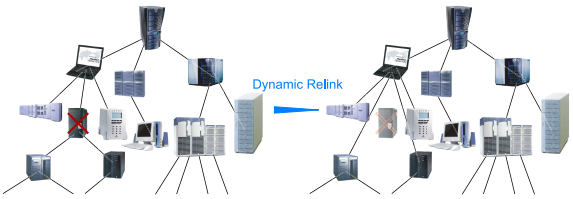

Fig. 5　uplink ノードが停止した場合の動的な再接続

構築される．この情報パケットに各ノードの負荷状況や，メモリの使用状況などの情報を入れることができる．その他に，ツリー構造を維持するために，Route-To という TAG があらかじめ定義されている．Route-To 情報は，uplink ノードからツリーの最上位ノードまでの経路にある全ノードのホスト名の一覧となっている．各ノードは，downlink ノードに対して，uplink ノードから取得した Route-To 情報に自分のホスト名を追加したものを伝達する．この Route-To 情報を利用して uplink ノードより上位のノード一覧を把握することができる．DNAS のシステムの特徴として，あるノードに障害が生じた場合に，システム全体を維持する機能を有していることが挙げられる．Fig. 5 に示すように，一定時間に一回 uplink ノードの生存状況を確認し，uplink ノードへの接続が成功しなかった場合，uplink ノードの機能が停止しているという判断を行う．そして，Route-To 情報を参考にして uplink ノードの uplink ノードがどのノードであるかという情報を得て，そのノードに再接続する．

また，1 つのノードが多数のノードと集中的に直接通信すると処理負荷が高くなるという問題点を解消するために，負荷に応じてツリー構造の再構成を行うことができる．Fig. 6 に示すように，downlink ノード数が指定した最大値より多い場合は，ネット

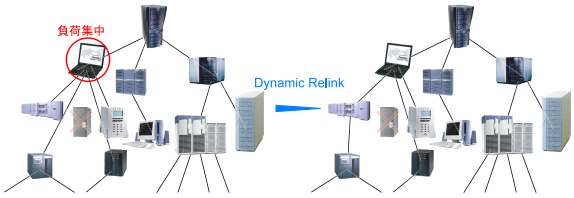

Fig. 6　downlink ノード数が多い場合の動的な再接続

ワークの通信負荷の軽減をはかるために，downlink 数を制限することができる．ランダムに 2 つのノードを選択し，一方の downlink ノードをもう一方の downlink ノードの downlink ノードとなるように再接続する．

以上のように，このシステムを利用することで単一のノードに負荷を集中することなく，効率的に情報を交換することができる．また，あるノードが停止してもその停止ノードを迂回して，システムを維持することができる．

実装したインターフェースが P2P フレームワークで正常に動作すること，管理者への通知メールの送信機能などを確認した．また，表示された値と計算ノードでの実際の値が一致していることを確認した．

5　まとめと今後の課題

本稿では．P2P フレームワークに対応したモニタリングインターフェースの必要性を述べ，提案システムと既存のモニタリングソフトウェアとの比較を行い，その利点などについても述べた．テスト環境を構築し，システムの有用性，動作も確認した．今後は提案システムを大規模なグリッドテストベッドのような広域分散環境で運用し，クライアントソフトウェアを複数の既存モニタリングアプリケーションに対応させる予定である．

参考文献

1) Foster I. and Kesseleman C. The grid: Blueprint for a new computing infrastracture. *Morgan Kaufmann*, 1998.

2) Foster I., Kesseleman C., and Tuecke S. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 2001.

3) Global Grid Forum. `http://www.gridforum.org`.

4) 中田秀基. グローバルコンピューティングにおけるセキュリティ. *Computer Today*, Vol. 17, No. 99, 2000.

5) Pratap Pattnail, Kattamuri Ekanadham, and Joefon Jann. *Autonomic Computing and Grid, in Grid Computing(Edited by Fran Berman et al.)*. WILEY, 2003.

6) Xuehai Zhang, Jeffrey L. Freschl, and Jennifer M. Schopf. A performance study of monitoring and information services for distributed systems. *High-Performance Distributed Computing*, Vol. 12, , 2003.

7) Ganglia Cluster Toolkit. `http://ganglia.sourceforge.net/`.

8) Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: a distributed resource performance for ecasting service for metacomputing. *Future Generation Computer Systems*, Vol. 15, , 1999.

9) K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Proceedings of the tenth ieee international symposium. *High-Performance Distributed Computing(HPDC-10), IEEE Press*, August 2001.

10) Globus Project. `http://www.globus.org/`.

11) ”datagrid information and monitoring services architecture: Design, requirements and evaluation criteria”. Technical report, DataGrid, 2002.

12) Junichi Uekawa, Tomoyuki Hiroyasu, Mitsunori Miki, and Yusuke Tanimura. A dynamic hierarchical system for large scale distributed applications. *Proceedings of the 14th IASTED International Conference, Parallel and Distributed Computing and Systems, pp. 422–427*, 2002.