Jini を用いた Computing Portals System の開発

鈴村 豊太郎[†] 松岡 聡[†]
中田 秀基[††] 関口 智嗣[††]

Jipang(Jini-based Portals Augment Grid) システムは，分散オブジェクト技術 Jini を基盤に構築されたポータルシステムで，Ninf や NetSolve, Globus 等の各 Grid システムが提供する資源を統合し，拡張性と柔軟性を持った大規模計算基盤を構築する．また，その計算基盤を使用するためのユーザインタフェースとして，プログミングインタフェースと GUI アプリケーションを提供する．ユーザは，このユーザインタフェースを用いることによって，各 Grid システムの相違を意識することがなく，Grid を簡便かつ透過的に使用することができるようになる．また，このとき，Grid システムのクライアントパッケージをインストールする必要はなく，Java の実行時環境さえあればよい．本稿では，この Jipang システムが実現する機能とそのアーキテクチャに関して述べる．

Overview of Jini-based Computing Portals System

Toyotaro Suzumura,[†] Satoshi Matsuoka,[†] Hidemoto Nakada[††]
and Satoshi Sekiguchi[††]

Below sentences were written in the paper for JSPP 2000, so temporary. Therefore, in the near future, these will have to be absolutely changed. Recent developments of global computing systems such as Ninf, NetSolve and Globus have opened up the opportunites for providing high-performance computing services over wide-area networks. However, most research focused on the individual architectural aspects of the system, or application deployment examples, instead of the necessary charactersistics such systems should intrinsically satisfy, nor how such systems relate with each other. Our comparative study performs deployment of example applications of network-based libraries using Ninf, NetSolve, and CORBA systems.

1. はじめに

Grid[☆]は広域ネットワーク上に分散した計算／情報リソースを活用して大規模計算を実現する基盤技術であり，Globus[5)],Ninf[12)], NetSolve[4)], Legion[6)], Neos/MetaNeos 等，近年これを目的としたシステム[☆☆]が複数提案されている.

これらのシステムは独自のコンセプトで Grid を実現する基盤を構築しており，各々提供しているサービスが異なる．また，各システムは基本的にすべてのリソースにそのシステムがインストールされていることを想定して構築されており，システム間の相互運用性は考慮されていない．しかしながら，将来，すべてのホストに単一のシステムがインストールされていることを仮定するのは困難であり，ユーザがアクセス可能な計算資源には，様々な Grid system がインストールされていることが予想される．しかし, Grid のユーザにとっては, 使用可能な計算資源を最大限活用できることが望ましい. また，Grid の環境として，より広範囲の計算資源を網羅することが必要である．更に，ユーザが Grid 上でのアプリケーションを構築するときに，各システム毎のプログラミングモデルに応じてアプリケーションを構築する必要があり，ユーザの負担が非常に大きい．よって，これらのシステムのための統一したプログラミング環境を構築することが不可欠である．これによって，ユーザのプログラミングの負担が少なくなるばかりか，ユーザが単一のシステムに制限されることなく，莫大な計算資源を扱うことができるようになる．更に, ユーザが同一のアプリケーションから異なるシステムの機能を呼び出すことも可能になる．

一方，ユーザの観点から見ると，プログラミングや Grid システムをインストールすることなく，簡便かつ透過的に Grid を利用できるようになることが望ましく，近年，Web ブラウザや PDA から Grid への透過的なアクセスを可能にする "ポータルシステム" と呼ばれるシステムの開発が盛んになってきている．ポータルシステムの例として，HotPage[7)] や Gateway System[7)] が挙げられるが，HotPage は，Web 上で計算資源の動的負荷情報を表示したり，ファイル操作，ジョブの投入／削除を可能にするシステムである．また，Gateway

† 東京工業大学 Tokyo Institute of Technology
†† 電子技術総合研究所 Electrotechnical Laboratory
☆ Global Computing, Computational Grid とも呼ばれる
☆☆ これらを本稿では Grid システムと呼ぶことにする

System は，バックグランドとして Globus を稼働させ， Gateway が提供するビジュアルプログラミングツールや実行時環境を用いて，ユーザが簡便に Grid 用のアプリケーションを構築することを可能にするシステムである．また， NetSolve や Neos などもユーザが直接，ジョブを投入できる GUI を提供しており，個々の Grid システムの簡便化を計っている．しかし，いずれも特定の Grid システムに特化したインタフェースとなっており， Grid システムのための汎用のインタフェースを提供するポータルシステムを構築することが重要である．

本稿では，上記の問題を解決するポータルシステム Jipang(Jini-based Portals Augment Grid) システムを提案する．分散オブジェクト技術 Jini を基盤に構築された Jipang システムは， Ninf や NetSolve, Globus 等の各 Grid システムが提供する資源を統合し，拡張性と柔軟性を持った大規模計算基盤を構築する．また，その計算基盤を使用するためのユーザインタフェースとしては，プログミングインタフェースと GUI アプリケーションが与えられる．このユーザインタフェースを用いることによって，ユーザは各 Grid システムの相違を意識することがなく， Grid を簡便にしかも透過的に使用することができるようになる．また，このとき， Grid システムのクライアントパッケージをインストールする必要はなく， Java の実行時環境さえあればよい．

以降， 2章では Jipang の概要を説明し， 3章では， Jipang システムの基盤となる Jini のアーキテクチャについての述べる， 4章では， Jipang のアーキテクチャについて述べ，??章では Jipang が提供するユーザインタフェースを紹介する． 6章では関連研究，最後に 7章でまとめを述べる．

2. Jipang の概要

Jipang(Jini-based Portals Augment Grid) システムは， Ninf や Globus 等の様々な Grid システムが管理するリソースを統合した大規模計算基盤を構築し，各システムの相違を意識することなく，統一的かつ簡便に Grid を使用することを目指したシステムである．

Jipang システムは，分散オブジェクト技術 Jini を基盤に構築されたシステムで，統合環境を構築する Jipang インフラストラクチャとそれを使用するユーザインタフェースから構成される (図 1)．

各コンポーネントが実現する機能と特徴は以下のようである．

- Jipang インフラストラクチャ
  各 Grid システムが管理する計算リソースを統合した大規模な計算基盤を提供する．
  - 単一のシステムが提供するリソースの数よりも莫大な計算リソース / 計算サービス群を提供できる
  - 未知のシステムでも容易にこの計算基盤に統合できる
  - リソースの追加 / 削除が自動的に行われる．
  - 計算リソースの故障を検知する
  - 強固なセキュリティ機構

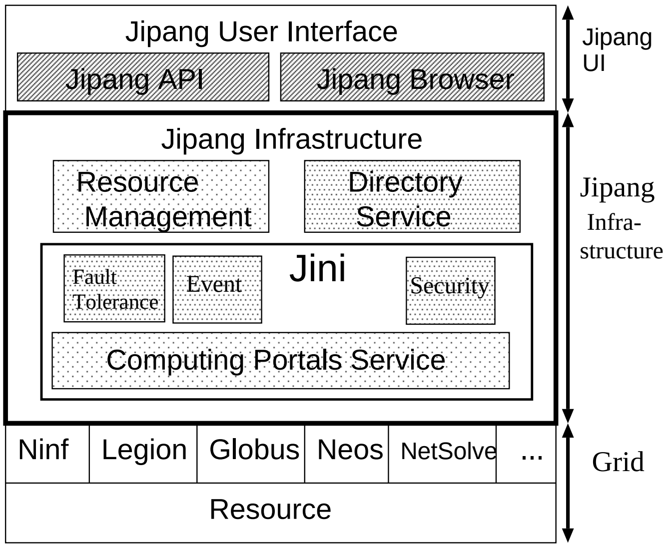

図 1 Jipang のアーキテクチャ

- ユーザインタフェース
  Jipang インフラストラクチャをユーザが使用するためのコンポーネント．
  - プログラミングインタフェース (Jipang API)
    様々な Grid システムの為の統一したプログラミング環境の提供
    * ユーザは各システム毎のプログラミングモデルに応じてアプリケーションを構築する手間が省ける.
    * 同一のアプリケーションから異なるシステムの機能を呼び出すことができる.
    * 各 Grid システム間で柔軟に他のシステムのサービスを組み込むことができ，相互運用性が容易に実現できる．
  - GUI のアプリケーション (Jipang Browser)
    Grid システムの為の汎用の GUI の提供
    * ユーザが簡便かつ透過的に Grid システムを使用することができる
    * ユーザのプログラミングの労力を省く
    * 計算資源の属性 (CPU 負荷，ネットワークバンド幅等) を表示し，ユーザが柔軟にリソース /Grid システムを選択できる

上記の Jipang インフラストラクチャは， Jipang の中心的な役割を果たすコンポーネントであり，図 1のように Jipang インフラストラクチャが各 Grid システムの上位レイヤに位置することで，各システムの相違を吸収し， ユーザインタフェースが統一的にこれらを使用することを可能にしている．また，ユーザは各 Grid システムを使用するときにクライアントパッケージをインストールする必要はなく，このユーザインタフェースを稼働させる Java の実行環境があればよい．よって， Jipang システムは，リソースの OS やアーキテクチャに左右されなく，完全ポータブルであると言える，また，従来のシステムのように，サーバのバージョンアップに伴い，クライアント側も更新する必要はなく，常に

最新のバージョンのサーバを使用することができる．

Jipang インフラストラクチャの詳しいアーキテクチャに関する説明は 4 章で行う．また，Jipang API と Jipang Browser のより詳細な機能とアーキテクチャは 5 章で行う．

3. Jini の概要

Jini システムは，Sun が提唱した分散オブジェクト技術で，シンプルで柔軟性が高く，一体化されたネットワークを目指して設計された Java ベースの分散システムである．Jini アーキテクチャでは，すべてのデバイス及びソフトウエア，そしてユーザをサービスとして扱い，それらをグループ化し連合体を形成することで，互いのリソースを必要に応じて利用し合える環境を実現する．

Jini システムの構成要素は次のとおりである．

- Jini 連合体を形成し，各自の機能を互いに利用し合うサービス
- 分散システムのサービス統合に必要なインフラストラクチャ
  - Lookup Service(LUS)
    サービスの格納場所．必要なサービスを検索し，そのサービスへアクセスする．また，各サービスの機能を表すインタフェースとそのサービスを実際に提供するオブジェクトとを関連付ける．
  - Discovery Protocol, Lookup Protocol
  - Java RMI システムの拡張バージョン
  - 分散セキュリティ・システム
- 信頼性の高い分散サービスの実現を支援するプログラミング・モデル
  - リース・インタフェース
  - イベント / 通知インタフェース
  - トランザクションインタフェース

次に Jini システムの中心となる，Discovery Protocol と Lookup Protocol について述べる．

3.1 Discovery Protocol

サービスを Jini システムへ追加する手順．サービス・プロバイダは，自身の存在通知をブロードキャストすることによって，LUS を呼び出す．次に，そのサービスのプロキシが LUS へ読み込まれる．このプロキシには，そのサービスのインタフェースと，サービス内容に関するその他の説明が含まれている．

3.2 Lookup Protocol

LUS のユーザインタフェースには，各サービスの機能や内容に関するわかりやすい説明が表示される．クライアントは，これらの説明と，型 (Java インタフェース) を手がかりとして必要なサービスを検索する．そして，LUS がサービスのプロキシ・コードをクライアントに転送する．クライアントは，このコピーのインタフェースを介して目的のサービスと通信を行うが，このインタフェースは次の 2 種類に大別される．

- プログラミング・インタフェース
  プログラミング・インタフェースには，Java 言語の型 (クラス) が指定されており，クライアントはその型を手掛かりとして，LUS から目的のサービスを探す．そして，そのサービスのメソッドを直接呼び出すことによって，そのサービスとのやり取りを行う．
- ユーザ・インタフェース
  ユーザインタフェースも LUS に格納でき，ユーザとサービスが対話するためのインタフェースを提供. ユーザ・インタフェースは Java の Applet 型[☆]なので，ブラウザやその他のユーザインタフェース・ツールに表示できる.

4. Jipang インフラストラクチャのアーキテクチャ

Jipang のインフラストラクチャは Jini を基盤に構築されており，以下のコンポーネントから構成されている (図 1)

- Jini インフラストラクチャ
- 各 Grid system 用 Jini サービス
- Computing Portals Service
  各 Grid システムの相違を吸収するための Jini サービス
- Resource Management
  リソースの統合環境の構築
- セキュリティ
- フォールトトレランス

次に 2 ~ 6 の各コンポーネントの詳細を述べる．

4.1 各 Grid システム用 Jini サービス

このコンポーネントは，各 Grid システムの管理者が実装する Jini サービスであり，これを作ることによって，Jipang システムに統合される．現在，Ninf 及び NetSolve のみ実装済みである．

この Jini サービスの実装方法と Jipang システムへのシステムの登録方法を Ninf システムを例に述べる．

Jipang から各 Grid システムを使用するためには，専用の Jini サービスを構築する必要がある．

4.2 Computing Portals Service

Computing Portals Service(CPS) は，各 Grid システムの相違を吸収し，統一的に扱うための Jini サービスである．CPS の実装は単純で，各 Grid システムの Jini サービスを包括したラッパーサービスであり，実際の処理は，特定のシステムの Jini サービスに任される．

クライアントが LAN 内の LUS から Portals Service を取得し，目的の Grid system と通信を始めるまでの過程は以下のように行われる．(図 2)

- クライアントはディスカバリプロトコルで見つかった LUS を取得し，CPS を検索する．
- LUS が CPS をクライアント側に転送する．
- クライアントはダウンロードされた CPS を用いて，目的としている Grid system と通信をはじめる．

この過程は Jipang API と Jipang Browser が吸収し，実際にはユーザから隠蔽される．

☆ Applet 型に限らず，JFrame などの GUI の枠組のクラスから成る

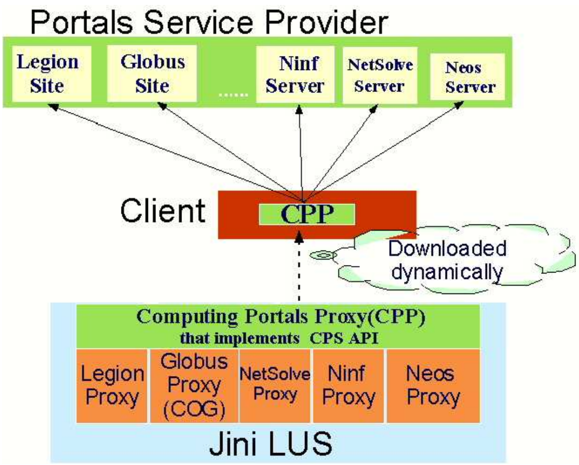

図 2 Jipang Proxy

4.3 Resource Management

Resource Management は，各 Grid システムが管理する計算資源を統合し，クライアントに適切なリソースを割り当てるコンポーネントである．この Resource Management の設計手法として，以下の 2 つの選択枝が考えられる．

（1） Jipang がすべての資源を管理する方法

この方法は，一つの Jini 連合体を複数組み合わせることにより，大規模な Jini の世界を形成できることを利用したもので，インターネット上の LUS に登録されているリソース及び Grid サービスを統合しようというものである．

インターネット上の LUS を一元管理する為に，LDAP(Light eight Directory Service) サーバを採用する． LDAP はディレクトリサービスを行う標準的なサービスで，すべての資源情報を一元的にここで管理することで情報のフィルタリングが容易に行うことができる．

この LDAP サーバには， Grid サービスだけでなく，そのサービスを提供しているリソース及びリソースの属性 (CPU 負荷情報，ネットワークバンド幅，ネットワークレイテンシ，メモリ量) などの情報も格納される．

また，クライアントからリソースの要求を受け，LDAP サーバにアクセスし，適切なリソースを返す Jini サービス"MatchMaker Service" を用意する． LDAP サーバをアクセスするライブラリは， Java 言語が用意されているので，容易に実装するできることが予想される．

この MatchMaker Service は，クライアントがリソースやサービス等の場所，名前あるいは属性を指定して要求したときに， LDAP サーバにアクセスし，候補となるリソースをクライアントに知らせる Jini サービスである．

この Resource Management の方法で，クライアントが適切なサービスを利用するまでの過程を以下のようである (図 3)．

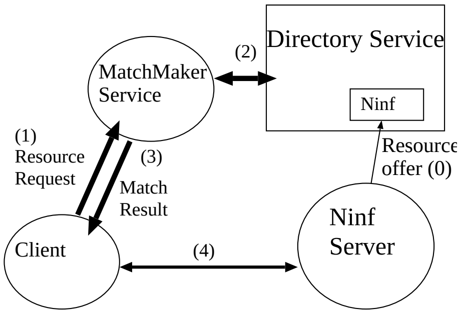

図 3 Jipang の資源管理アーキテクチャ・その (1)

- (0) サービスを近隣の LUS に登録すると，自動的にディレクトリ・サービスに反映される (何らかの障害が起きるか，明示的に LUS から削除された場合にはディレクトリサービスからも削除される)
- (1) クライアントは，近隣の LUS に登録された MatchMaker サービスを使い，サービス名もしくはリソースの属性等を指定してリソースの要求を行う
- (2)MatchMaker サービスがディレクトリ・サービスを検索
- (3)MatchMaker サービスから候補のサービスとリソースがかえってくる
- (4) クライアントは近隣の LUS に登録された Computing Portals Service を用い，(3) のリソースと通信を始める

（2） 各 Grid システムに資源管理を委ねる方法

個々の Grid システムが管理する資源については各システムが持つスケジューラ及び類似の機能を持つモジュールに委ねる．

Jini サービスである"Scheduling Service" が，Jipang システムが管理する各 Grid システムのスケジューラを管理し，ユーザがリソースの要求が来ると，この Scheduling Service がゲートウエイとなって，各システムのスケジューラに問い合わせ，結果をもらう (図 4).

各 Grid system が管理する計算資源を統合するための方法として，上の 2 つの方法を述べた．どちらの方法も長所と欠点がある．

前者の方法は，すべてのリソースを Jipang システムに統合し，一元的に管理することで，リソースの追加や削除，故障等に自動的に Jipang システムに反映され，拡張性の高い分散システムを構築することができる．しかし，リソースのスケジューリングは Jipang システムがすべて実装する必要があり，実装の負担が大きい．

一方，後者の方法は，スケジューリングは個々のシステムの責任で， Jipang が担う必要がないので， Jipang システムがリソースの情報等を管理する必要がない．よって， Jipang システムがシンプルな構造になる．また， Ninf や NetSolve などは，洗練されたスケジューリング機構を持っており，より適切な資源割り当てがなされる．しかし， Neos 等すべての Grid システムがスケジューリング機構を持っているわけではない．また，個々の Grid システムのスケジューリングモジュー

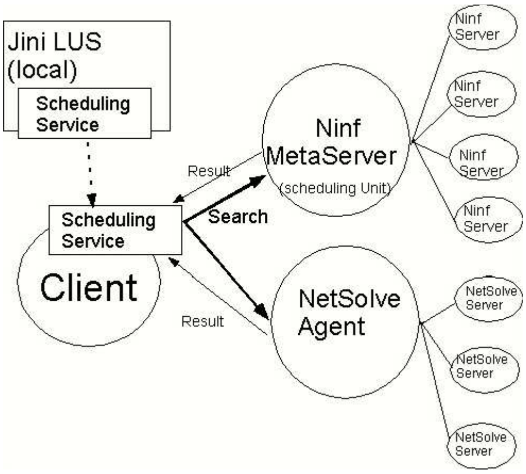

図 4 Discovery

ルは，Jipang システムの” Scheduling Service” と通信できるようにする必要があり，Jipang システムに統合する前提としてこの機構が前提となる．さらに，クライアントがリソースの要求をしたときに，“Scheduling Service” は，各 Grid システムのスケジューリングモジュールに問い合わせる．このため，1 番目の方法のようにリソースを一元的に管理していないので，通信の時間だけでなく，リソースの比較など，オーバヘッドがかかる要素ががかかってしまう．更に，次章に述べる Jipang Browser を実装するときに，この方法だとこのオーバヘッドが非常にネックとなり，相性が悪い．また，リソースを管理は個々の Grid システムの委任されてしまうので，Jipang が目的としている動的な分散システムの構築が困難になる．

現在，Jipang システムでは，動的な分散システムを構築し，かつ適切なスケジューリングが行われるように，前者の方法をベースにし，かつ，Ninf や NetSolve 等の優れたスケジューリング機能を持ったシステムの場合には，後者のように専用のスケジューリングサービスに資源管理をさせることを考えている．

4.4 セキュリティ

任意のユーザに計算資源を公開するには，強固な安全性を保証される必要があるが，Jipang システムでは，JDK1.2 の強固なセキュリティモデルによって保証されている．リソースの提供者はサービスを LUS に登録するときに，Java のクラスがロードされるコード・ソース毎に，リソースに対するアクセスの権限を細かく記述した policy ファイルを渡す．policy ファイルの例を下に示す．

```
grant codeBase
"http://www.is.titech.ac.jp/",
signedBy "suzumura"
{ permission java.io.FilePermission
                   "/tmp/*", "read";
 permission java.io.SocketPermission
                    "*", "connect";};
```

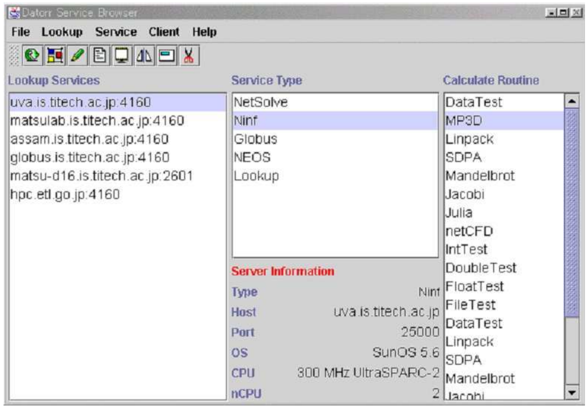

図 5 Jipang Browser のプロトタイプ

4.5 Fault Tolerance

Jini アーキテクチャには分散イベントが採り入れられおり、他のオブジェクトに対してイベント登録をしておくと、登録先のオブジェクトで特定のイベントが発生した場合、登録元のオブジェクトにその旨が通知される．Jipang では，この機能を用いることによって，リモートのサービスのサーバが fail したときなどにクライアントに通知し，他の代替のサーバを用いるようにしている．

5. Jipang のユーザインタフェース

5.1 Jipang API

Jipang API は，Ninf や NetSolve、 Globus などの様々な Grid システムを統一的に使用するためのプログラミングインタフェースを提供する．また、明示的にリソースを選択することも可能だが、 Resource Manager にリソースの割り当てを任せ、実行時に最適な性能が得られるようすることもできる．

5.2 Jipang Browser

Jipang Browser は，グラフィカルなユーザインタフェースを通して Grid を簡便かつ直感的に使用することを可能にするアプリケーションである．

Jipang Browser は以下のような機能を持つ．

- Jipang に統合されている Grid サービスの表示
- 個々の Grid サービスごとに独自の GUI を提供.
- サービスの検索機能
  ユーザはサービスの属性を指定して検索を行う
- サービスの属性とサービスを提供する計算資源の属性を表示
  提供する計算資源の属性 (CPU の動的負荷情報，ネットワークレイテンシ etc) 等をを見ながら，適切なサービスを選択できる．
- サービスのマニュアルを提供

Jipang では，未知の Grid システムでも，そのシステムが Jini 対応のシステムでかつ，GUI を提供していれば，この Jipang Browser から使用することができる．また，このときに，そのシステムは URL やマニュアルを表示し，そのシステムを全く知らないユーザにも簡便にそれらのシステムを使用できるようにする．

6. 関 連 研 究

Jipang システムのように Grid システムを統合する試みとして，以下の UCSD の Rich Wolski 等が提唱している EveryWare が挙げられる．

EveryWare[10)] は，Grid program を書くためのユーザレベルのソフトウエア・ツールキットであり，異なる Grid システムを統合し，ユーザがシームレスに個々のシステムの資源を使用できるライブラリを提供する．また，EveryWare のユーザは，Grid プログラムを書くときに特有な計算資源 (リソースのタイプ，ネットワーク負荷等) の性能の動的変化を考慮する必要がない．

SC'98 の HPCC では，Ramsey Number の探索問題を解く分散アプリケーションを実装し，その中で，Globus, Legion, Condor, NetSolve, Java, Windows NT, Unix のインフラを同時に使用している．

しかし，現在，まだ，EveryWare のアーキテクチャの詳細や API は公開されていない．

7. お わ り に

本稿では，様々な Grid システムの統合環境を提供し，統一的かつ簡便に Grid システムを使用できるツールを提供する Jipang を提案した．Jipang は，Jini の機能を利用し，様々な Grid システムが提供する資源を統合し，莫大な計算資源と計算パワーを提供する柔軟な動的分散システムの形成する．また，ユーザには，それらの資源を統一的に扱うためのプログラミングインタフェースと，GUI のアプリケーションを提供し，簡便かつ透過的にそれらを使用することを可能にしている．

現在，Ninf 及び NetSolve のみ Jipang に統合しているが，今後は，Globus，Legion，Neos 等多くの Grid システムを統合し，Jipang システムの汎用性，拡張性，柔軟性を検証する予定である．また，Jini のリースの機構を応用して遊休計算機の有効利用や、計算サーバの課金制度機構の導入なども考えている。

参 考 文 献

1) Jini connection technology, jini's specification. http://www.sun.com/Jini/main.html.
2) Sumalatha Adabala, Nirav H.Kapadia, and Jose A.B.Fortes. Interfacing wide-area network computing and cluster management software: Condor, dqs and pbs via punch. Technical report, 1999.
3) Ahmed Al-Theneyan, Piyush Mehrotra, and Mohammed Zubair. Enhancing jini for use across non-multicastable networks. Technical report.
4) Henri Casanova and Jack Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. In *Proceedings of Super Computing '96*, 1996. http://www.cs.utk.edu/netsolve/.
5) Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 1997.
6) Andrew Grimshaw, Adam Ferrari, Fritz Knabe, and Marty Humphrey. Legion: An operating system for wide-area computing. Technical report.
7) Tomasz Haupt, Erol Akarsu, and Geoffrey Fox. Webflow:a framework for web based metacomputing. Technical report.
8) M.Livny, J.Basney, R.Ramen, and T.Tannenbaum. Mechanisms for high throughput computing. Technical report.
9) Gregor von Laszewski, Ian Foster, Jarek Gawor, Warren Smith, and Steven Tuecke. Cog kits: A bridge between commodity distributed computing and high-performance grids. Technical report.
10) Rich Wolski, John Brevik, Chandra Krintz, and Graziano Obertelli. Running everyware on the computational grid. Technical report, 1999.
11) 関口智嗣, 佐藤三久, and 長島雲兵. ネットワーク数値情報ライブラリ: - ninf の設計 -. 情報処理学会研究報告 *HPC*, 94(94-HPC-52):127, 1994.
12) 中田秀基, 草野 貴之, 松岡 聡, 佐藤 三久, and 関口 智嗣. ネットワーク数値ライブラリ ninf におけるメタサーバアーキテクチャ. 情報処理学会研究報告 *HPC*, 96(22), 1996.