第 XXV 部

WIDE における PlanetLab を利用した研究開発

第 25 部
WIDE における PlanetLab を利用した研究開発

第 1 章　はじめに

PlanetLab ワーキンググループは、WIDE プロジェクトや日本における PlanetLab の運用及び研究利用のために 2005 年 4 月に発足した。PlanetLab の概要を説明し、本年度設置したもしくは、設置を予定している WIDE プロジェクトノード、JGN II ノードの状況を説明する。PlanetLab を用いた広域アクティブ計測基盤を構築し、実際に PlanetLab ノードを用いて動作検証を行った。

第 2 章　PlanetLab の概要

PlanetLab[191, 307] とは、2003 年に Intel や HP などアメリカの IT 企業やプリンストン大学など世界 60 以上の大学が中心として発足した、世界的なテストベッドプロジェクトである。PlanetLab の目的は、実インターネットを利用したアプリケーションやサービスに向けた、グローバルなネットワークテストベッドを構築することである。

PlanetLab へは、企業、大学など一定の条件を満たすことでそこに所属する研究者は PlanetLab の環境を利用することができる。PlanetLab は、各組織から提供される PC をインターネットに接続し、オーバーレイネットワークを構成する。従来、インターネットを利用したアプリケーションやサービスは、シミュレーションや、研究室ネットワークなど閉じられた環境で検証されてきた。PlanetLab を利用することで、分散アプリケーションの研究や検証を実インターネット環境下で行うことが可能となる。また、PlanetLab を活用した CDN（Contents Delivery Network）、分散ハッシュテーブル、オーバーレイルーティングなどの開発、検証が行われている。PlanetLab へ参加するためには、最低 2 台を PlanetLab ノードとして提供する必要がある。それぞれのノードは、PlanetLab 参加組織のメンバーが利用することができる。

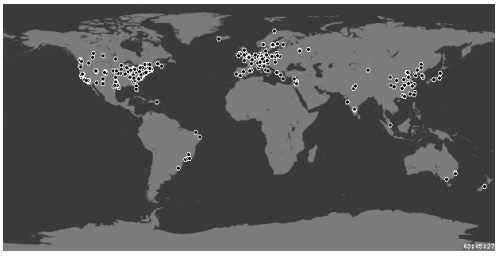

図 2.1. Current distribution of 724 nodes over 353 sites. (Extract from `http://www.planet-lab.org/`)

2.1 日本国内における PlanetLab

日本の組織は、WIDE プロジェクト、東京大学、大阪大学、情報通信機構（NICT）、慶應義塾大学、北陸先端科学技術大学院大学、奈良先端科学技術大学院大学、JGN II が参加している。2005 年に日本における PlanetLab の活用を促進する目的として、東京大学を中心に PlanetLab Japan[192] を発足し活動を行っている。

2.2 アーキテクチャ

PlanetLab のすべてのノードは、planet-lab.org によって制御されている（図 2.2）。PlanetLab のアーキテクチャは、過去 3 年間にわたって進化してきている。現在、公開されているバーションは、3.3 である。2007 年には、現在開発が進められている PlanetLab 4.0 が公開される予定である。

2.2.1 PlanetLab Architecture

PlanetLab のアーキテクチャ [185, 188] には、Service（サービス）、Slice（スライス）、Sliver（スリバー）という概念がある。Service とは、分散システム全体が提供するプログラム群である。Slice とは、すべてのノードの資源を各サービス毎に割り当てる単位である。Sliver とは、スライスをノード単位に分割した単位である（図 2.3）。

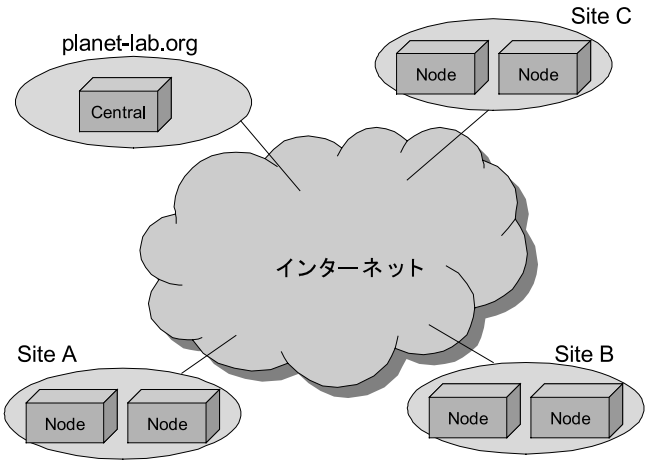

図 2.2.　PlanetLab Architecture

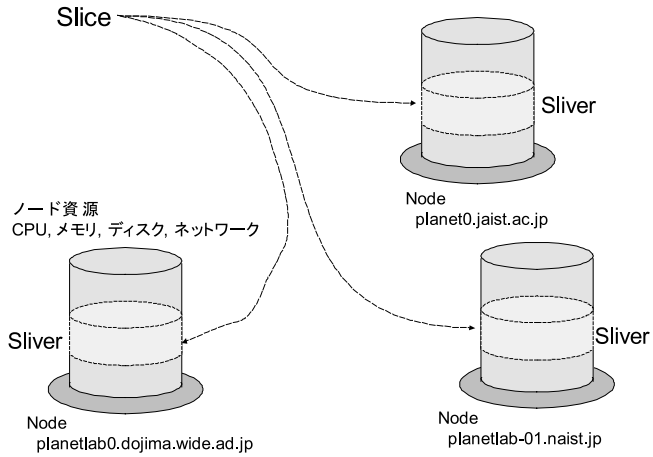

図 2.3.　PlanetLab Architecture（Slice イメージ）

2.2.2 Node Level Architecture

それぞれのノードは、Node Manager と VM（Virtual Machine）から構成される。Node Manager は、複数の VM と Network Interface を管理する。VM は、PlanetLab アーキテクチャの Sliver に相当する。それぞれの Sliver において、メモリ空間、プロセス、ファイルは、独立している。それぞれのユーザ環境はユーザごとに構築していく。Slice は、サービスごとに割り当てられている PlanetLab 上の VM の集合である。

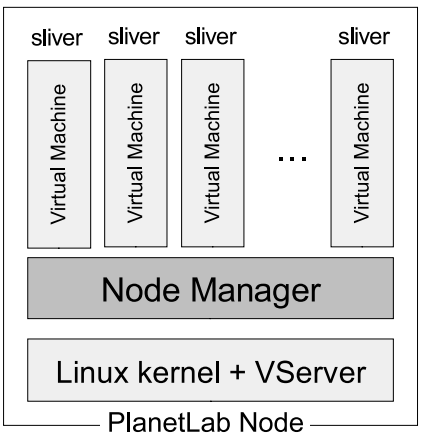

図 2.4.　Node Level Architecture

2.2.3 Linux-VServer

PlanetLab は、OS レベルでの仮想化を提供する Linux-VServer[130] を採用している（図 2.5）。FreeBSD jail や、User モード Linux と似たものである。複数の Slice をサポートするのに十分なスケーラビリティを持ち合わせている。Linux-VServer は、1 台のコンピュータ上に複数の仮想 Linux サーバを立ち上げることが可能である。VServer は、通常の Linux サーバと同様に動作するが、オーバーヘッドは少なく、相互に独立して動作することが可能であ

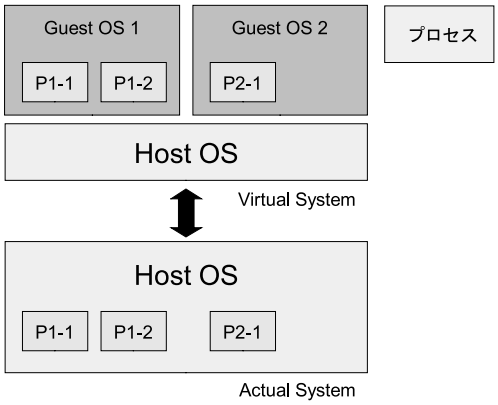

図 2.5. PlanetLab Linux-VServer

る。オペレーティングシステムレベルでの仮想化であるため、資源をグループ化し分離できる。Sliverは、ゲスト OS として実装されている。ホスト OS とゲスト OS はカーネルを共有している。ゲスト OS は、ホスト OS と同じである必要がある。

2.2.4 Network

PlanetLab は、IPv4 のみをサポートしている。ネットワーク資源は Sliver で共有している。ネットワーク資源とは、IPv4 アドレス、ポートスペースである。Node Manager は、TCP/UDP ポートを監視しており、パケットを適切な Sliver へ配送する(図 2.6)。

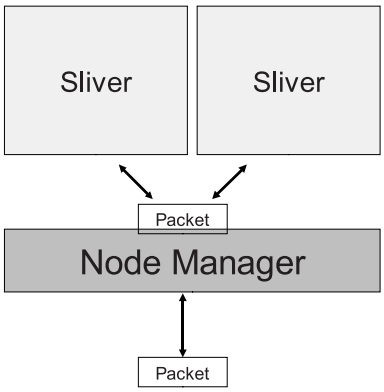

図 2.6. Network Resource Sharing

2.2.5 環境

PlanetLab で動作するプログラムは通常の UNIX プログラムと同じである。ただし、gcc や javac などのコンパイル環境は提供されていないため、ユーザごとにインストールする必要がある。Perl や Python はすべてのノードではじめから利用することが可能である。PlanetLab ノードは、RPM と yum をサポートしている。Sliver ごとに root 権限が与えられるためユーザが必要なプログラムをインストールすることが可能である。

第 3 章　現在の WIDE 及び JGNII のノードの状況

3.1 WIDE プロジェクトノード

現在、WIDE プロジェクトとして東京と大阪に 2 台ずつ、計 4 台のノードを設置している。

- planetlab0.otemachi.wide.ad.jp
- planetlab1.otemachi.wide.ad.jp
- planetlab0.dojima.wide.ad.jp
- planetlab1.dojima.wide.ad.jp

3.1.1 ノード利用率

CoMon[43] が提供しているサービスによって得られた利用率をあげる。図 3.1–図 3.22 は、2006 年 12 月 28–29 日におけるノード状態のグラフである。表 3.1 は、2006 年 12 月 29 日のある時点におけるノードの状態である。

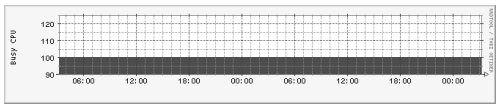

図 3.1. planetlab0.otemachi Busy CPU

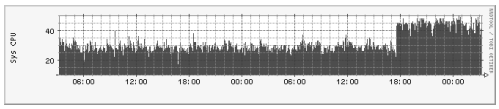

図 3.2. planetlab0.otemachi Sys CPU

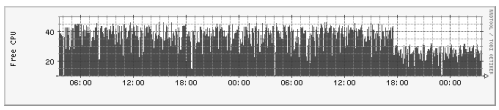

図 3.3. planetlab0.otemachi Free CPU

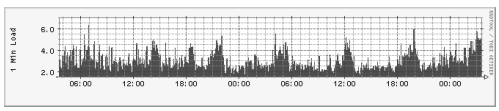

図 3.4. planetlab0.otemachi 1min Load

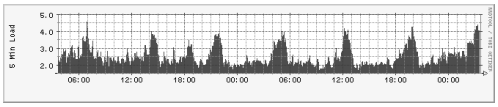

図 3.5. planetlab0.otemachi 5min Load

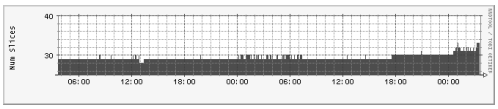

図 3.6. planetlab0.otemachi Number of Slices

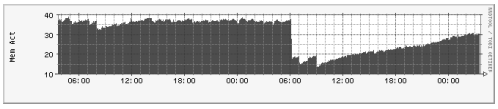

図 3.7. planetlab0.otemachi Memory Act

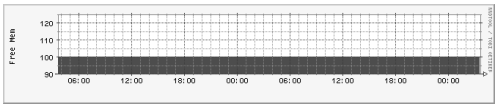

図 3.8. planetlab0.otemachi Free Memory

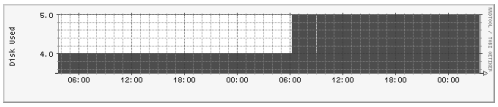

図 3.9. planetlab0.otemachi Disk Used

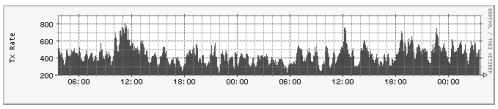

図 3.10. planetlab0.otemachi Tx Rate

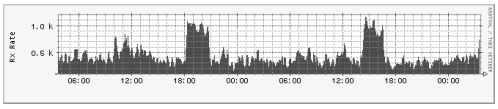

図 3.11. planetlab0.otemachi Rx Rate

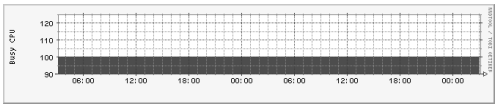

図 3.12. planetlab0.dojima Busy CPU

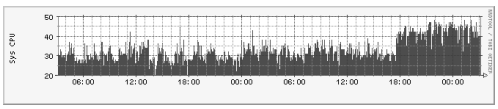

図 3.13. planetlab0.dojima Sys CPU

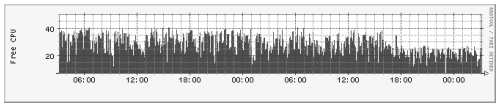

図 3.14. planetlab0.dojima Free CPU

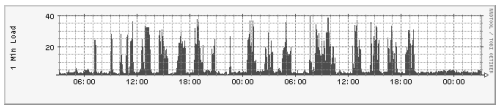

図 3.15. planetlab0.dojima 1min Load

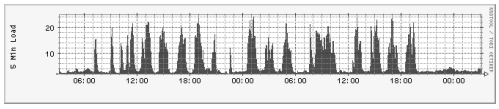

図 3.16. planetlab0.dojima 5min Load

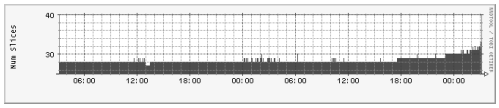

図 3.17. planetlab0.dojima Number of Slices

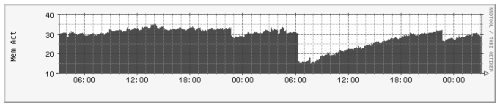

図 3.18. planetlab0.dojima Memory Act

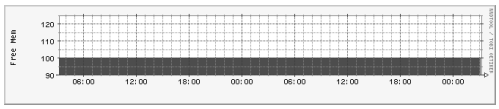

図 3.19. planetlab0.dojima Free Memory

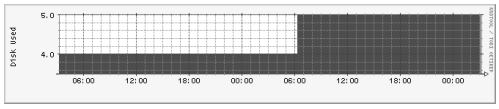

図 3.20. planetlab0.dojima Disk Used

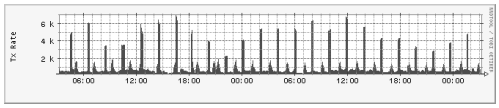

図 3.21. planetlab0.dojima Tx Rate

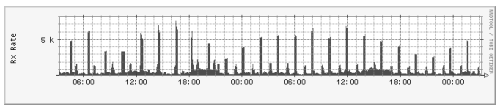

図 3.22. planetlab0.dojima Rx Rate

表 3.1. WIDE Project Node Status

Name	planetlab0.otemachi	planetlab1.otemachi	planetlab0.dojima	planetlab1.dojima
Kernel Ver.	2.6.12	2.6.12	2.6.12	2.6.12
CPU	Pentium D 3.0 GHz	Pentium D 3.0 GHz	Pentium D 3.0 GHz	Pentium D 3.0 GHz
Busy CPU[1]	100.0%	97.0%	100.0%	94.0%
Sys CPU[2]	40.0%	52.0%	37.0%	54.0%
Free CPU[3]	25.2%	47.6%	31.1%	44.8%
1min Load[4]	4.39	2.43	5.70	2.40
5min Load[5]	3.70	2.03	4.07	2.02
Num Slices[6]	33	28	31	27
Mem Size	4 GB	4 GB	4 GB	4 GB
Mem Act[7]	29%	24%	30%	32%
Free Mem[8]	100	100	100	100
Disk Size	289.4 GB	289.4 GB	289.4 GB	289.4 GB
Disk Used	5%	4%	5%	5%
Swap Used	0%	0%	0%	0%
Tx Rate[9]	685	491	565	430
Rx Rate[10]	419	388	443	246

[1] Busy CPU: the percentage of time the CPU is utilized.
[2] Sys CPU: the percentage of time the CPU is spending in the OS.
[3] Free CPU: Free CPU indicates how much of the CPU a spin-loop was able to obtain, giving some insight how much of the node's CPU a new slice would receive.
[4] 1min Load: Load averages (# of runnable processes) for the past 1 minutes.
[5] 5min Load: Load averages (# of runnable processes) for the past 5 minutes.
[6] Num Slices: Num Slices indicates the maximum number of slices reported by slicestat over the past 5 minites, measured once every 30 seconds.
[7] Mem Act: percentage of the memory is in the operating system's "active" pool.
[8] Free Mem: the output of a test program that tries to grab 100 MB of memory on the node.
[9] Tx Rate: Transmit rates (in Kbps), taken as a sum of per-slice activity.
[10] Rx Rate: Receive rates (in Kbps), taken as a sum of per-slice activity.

3.2 JGN II ノード

現在、JGN II ノードとして、仙台、名古屋、大阪、岡山、広島、高知、北九州への設置準備を行っている。マルチホーム実験を行うために、JGN II ノードは、JGN II L3 ネットワーク、SINET、地域ネットワーク網などに接続する予定である。

第 4 章 PlanetLab 上での広域計測基盤システムの構築

近年、インターネットが急速に発展し、重要な社会インフラとして活用されるようになり、接続性だけではなく通信の品質も求められるようになってきている。それにともない大規模なネットワーク環境下において通信の品質を計測することが必要となってきている。多くのネットワーク計測ツールが提案されてきた。しかしながら、これらのツールは個別の目的を達成するためのツールであり、ユーザが複数の事項を計測したいときには、それらをユーザ自身が組み合わせて利用しなければならない。ユーザの利便性を向上させ、計測の集約を行う広域アクティブ計測基盤システムの提案をする。提案するシステムを広域ネットワークテストベッドである PlanetLab 上で実装し、実験を行った。

4.1 背景

近年、インターネットが急速に発展し、重要な社会インフラとして活用されるようになり、接続性だけではなく通信の品質も求められるようになってきている。そのため、ネットワークの状態や品質を測定する計測技術の重要性が増してきている。

ネットワーク計測ツール・手法は、多くのものが提案され実装されている。個々の目的のためには各ツールを使うことで解決できる。しかしながら、ユーザが複数の事項を計測したいときには、ユーザ自身がツールを組み合わせて利用することになる。ツールごとに計測結果の出力方法などが異なり、ユーザ

は結果の解析を行うのが煩雑になりやすい。そのために、統一したインタフェースでツールを操作でき、結果を出力できるシステムが求められる。

広域ネットワークの状態を測定するためには、多数のノードの配置が必要となる。多くのノードを一括して利用できるシステムが求められる。また、ノードが何らかの事情により動作しなくなったときにでも安定して計測できるためのシステムが求められる。

ネットワーク計測 [304] には、パッシブ計測とアクティブ計測の 2 種類がある。パッシブ計測は、ルータやスイッチなどのネットワーク機器上を通過するトラフィックを収集する。アクティブ計測は、End-to-End のノード間で、試験パケットを送受信しあい、ネットワークの挙動を観測し、ネットワーク品質を推定する。本研究では、アクティブ計測を対象とする。このような要求に基づき、本論文では広域ネットワーク上で動作するアクティブ計測基盤システムの提案を行う。次章では、広域計測基盤に関連研究を述べ、考察を行い既存研究の問題点をまとめる。4.3 節では、システムの機能用件について述べる。4.4 節では、提案するシステムについて述べる。4.5 節では、提案するシステムの設計について説明する。

4.2 広域計測基盤に関する関連研究

本節では、広域計測基盤について説明し、関連研究を述べ、それらの問題点について考察する。

4.2.1 広域計測基盤

広域計測基盤とは、広域ネットワークの計測を支援するためのシステムである。通常、広域計測基盤はネットワーク上に分散する多数の計測ノードを保有しており、各計測ノード内の計測エージェントを用いて計測を行う。広域計測基盤は、計測エージェントが計測を行い、データを収集し提示する機能をユーザへ提供するシステムである。ユーザは、広域計測基盤を用いて必要なデータを収集することにより、データの解析に専念することができる。

4.2.2 Sophia

Sophia[269] は、刻一刻と状態が変化するネットワーク状態を把握するための広域計測基盤である。Sophia は、ユーザが行いたい計測内容を記述するための言語を定義している。記述言語は、Prolog のような論理型言語となっている。このような記述言語を用いることによって、システムの動作内容や計測データの取得などの操作を定義できる。また、記述言語を用いて計測スケジュールをあらかじめ記述でき必要なときに結果を得ることも可能である。Sophia において、各ノードは過去の計測データを保存しており、いつでも参照できる。計測データは、各ノードで分散管理されている。

4.2.3 協調型アクティブモニタリングシステム

蟹江 [292] らの研究は、一台の統括ホストと複数の計測ホストから構成され、複数の計測ホストが協調してアクティブ計測を行うことを目的としている。計測時に複数の計測ホスト間の同期問題とユーザが個々の計測ホストを選択し、アクティブ計測のためのジョブ割り当て作業の負担を削減することに重点をおいたシステムである。計測内容を記述するための専用言語によってシナリオを記述する。シナリオを統括ホストに投入する。記述したシナリオは、統括ホスト内のトランスレータによってシステム内部で利用できる命令に変換される。ほかの機能としては計測ホストのグループ化を行っている。計測ホストが統括ホストに新規登録する際、IP アドレスや OS の種類などにより適切なグループに登録される。計測時には、各グループから代表ホストが選出され、代表ホスト同士でアクティブ計測を行う。

4.2.4 The Network Weather Service

The Network Weather Service[274]（NWS）は、分散コンピューティング環境で動的に変化するパフォーマンス特性の正確な予報を提供し、広域に分散した計算機資源の有効活用を可能にするためのシステムである。ユーザは広域に分散する資源の中から、パフォーマンスを比較して資源を選択することが可能となる。NWS から取得できる情報としては、新しいプロセスに利用できる CPU Time の割合、TCP の接続時間、End-to-End での TCP のネットワークレイテンシ、End-to-End での TCP のネットワーク帯域である。NWS では、ホストを複数のグループに分割し、グループ間のみで計測を行う。また、グループ内ではあるホストがリーダーとなりトークンを流す。トークンを受け取ったホストのみが計測を行い、グループ内で競合が起きない工夫をしている。

4.2.5 関連研究の問題点

これらの関連研究においては、計測ノードをネットワークアドレスや AS レベルでグループ化している。また、過去の計測データを蓄積しておきユーザの要求に応じて過去のデータを返信するなどの工夫が行われている。しかしながら、いずれのシステムも複数のユーザからの要求が重複した場合の処理の集約機能は実現されていない。

4.3 広域計測基盤システム要件

広域計測基盤システムに求められる要件は以下の点があげられる。

- ユーザごとに異なる計測目的への対応
- システムのスケーラビリティ
- ユーザが多数のノードを用いて計測結果を得るための操作手段
- 計測データの再利用

まずユーザごとに異なる計測目的へ対応するというのは、システム側に各種計測ツールをモジュールとして組み込む事でユーザが自由に計測できるという事である。この機能によってユーザ自身が各計測ツールを組み合わせて利用するより計測が容易になる。

つぎにシステムのスケーラビリティであるが、広域および多数のノードで計測を行うためには、スケーラビリティが必要となる。PlanetLab で動作させることを考慮すると、600〜1000 台のノードがストレスなく動作することが求められる。

3 つめは、ユーザが多数のノードを用いて計測結果を得るための操作手段である。従来の研究では計測用の API を提供するにとどまっているものが多い。しかし、ノードを用いて計測結果を得るためには一連の動作を定義でき、ユーザが操作するためのユーザインタフェースが必要となる。

最後に、計測データの再利用である。即時性を求めない計測や過去の計測データを集めて過去の傾向としてみる場合があるので、過去に計測したデータを保存し、いつでも利用できるようにすることが求められる。

4.4 広域計測基盤システムの提案

本報告書では、シナリオ記述言語を用いて計測内容の記述を可能とし、計測ノードを様々な単位でグループ化する機能、複数のユーザの計測内容の集約機能を提供するシステムを提案する。

4.4.1 システム概要

本システムは、広域ネットワーク上で多地点に存在する計測ノードを用い、アクティブ計測によりネットワーク品質を測定する。広域計測システムの概要を図 4.1 に示す。本システムの構成要素は以下の通りである。

- ゲートウェイサーバ
  ユーザインタフェースをユーザに提供する。システムへの窓口であり、ユーザは全てゲートウェイサーバを利用して操作を行う。
- 計測ホスト
  計測ホストとは、広域ネットワーク上に設置された計測ノードを意味する。各計測ホストは、計測機能、タスクスケジューリング機能、計測データ保存機能をもつ。

本システムでは、全ての計測ホストはグループ化され、ユーザはグループ単位で計測ホストを操作す

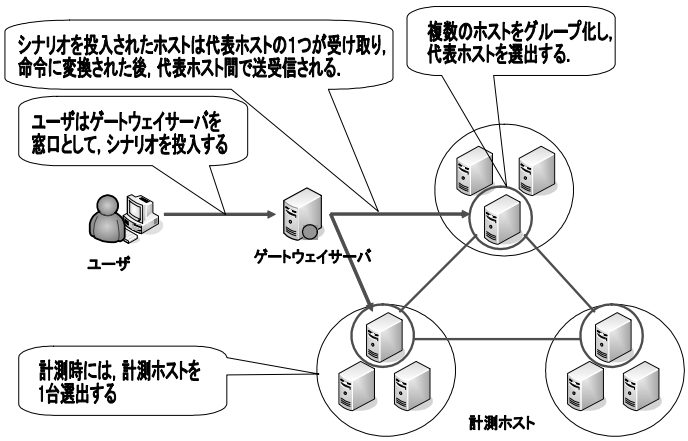

図 4.1. システム概要図

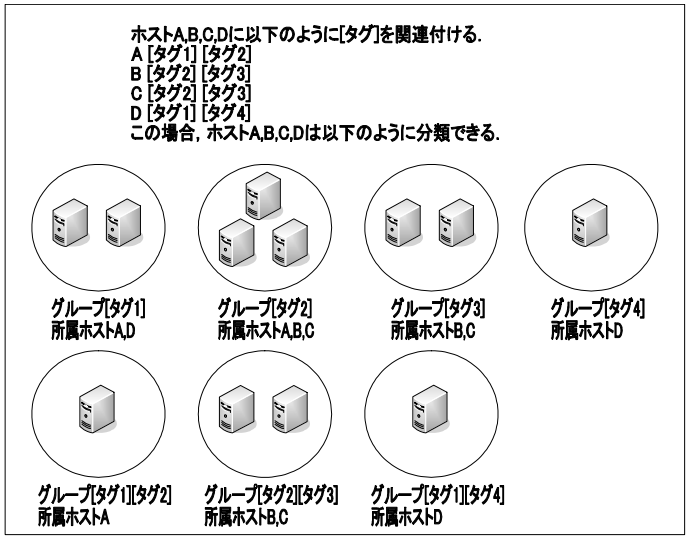

図 4.2.　タグによるグループ化の例

る。本システムでは、ユーザが行いたいひとつの計測内容をタスクと呼ぶ。各グループには 1 台以上のホストが所属し、その内の 1 台が代表ホストとなる。ユーザは、ウェブベースのインタフェースを提供するゲートウェイシステムを介して、各グループの代表ホストにタスクを投入する。本システムの操作は、専用のシナリオ記述言語によって行う。専用のシナリオ記述言語を用意することにより、ユーザの異なる要求に柔軟に対応することが可能となる。タスクを投入されたホストは、シナリオ記述言語をインタプリタで解釈し命令を計測に必要なグループに送信する。命令の送受信は、各グループの代表ホストが行う。命令を受信した代表ホストは、タスクスケジューリング表に新たなタスクを追加し、同一グループ内の計測ホストにタスクを割り当てる。計測結果は、計測を行ったグループの代表ホストに保存される。ユーザは必要に応じて、過去の計測履歴を検索、閲覧することが可能である。また、タスクスケジューリング表は、予定を格納するだけではなく、複数のユーザからタスクが投入された場合に重複するタスクを集約する。

4.4.2 計測ホストのグループ化

本システムでは、全ての計測ホストはグループ化され、ユーザはグループ単位で計測ホストを操作する。グループとは何らかの共通点をもつ計測ホストの集合である。グループ化の具体的な仕組みを以下に示す。

- グループ化の条件
  計測ホストにグループ属性を表すための識別子であるタグを付与する。全ての計測ホストはタグによりグループ化される。また、計測ホストに複数のタグを付与することが可能である。さらに、複数のタグによりグループの階層化なども可能となる。グループ化の例を図 4.2 に示す。各計測ホストは、自身が所属するグループの情報を所属グループリストで管理する。また、グループの代表ホストは、自身の管理するグループに所属する計測ホストの情報を管理グループ所属ホストリストで管理する（表 4.1）。
- 計測ホストのグループへの参加
  計測ホストを既存グループへ登録する場合を説明する。ユーザは、ゲートウェイシステムで計測ホストと付与するタグを入力する。ゲートウェイシステムは、タグが示すグループの代表ホストに新しく参加する計測ホストの情報を通知する。代表ホストは、管理リストに登録するホストを追加し、その情報を返信する。参加する計測ホストは、返信を受け取り参加が承認されたことを確認し、グループに参加する（図 4.3）。
- 新規グループの登録
  新規グループを作成する場合を説明する。グループには必ず 1 台以上の計測ホストが所属している必要がある。よって、新規グループを作成する場合は、ユーザは新規タグの定義と最低 1 台以上の計測ホストの関連付けを行う必要がある。

表 4.1. 所属グループリスト、管理グループ所属ホストリスト

リスト一覧	保有する情報
所属グループリスト	所属するグループを表す [タグ] 例）所属グループ [NAIST][WIDE] [NAIST] [WIDE]
管理グループ所属ホストリスト	グループに所属するホストの一覧 例）グループ [NAIST] の代表ホストが管理するリスト planetlab-01.naist.jp planetlab-02.naist.jp planetlab-03.naist.jp

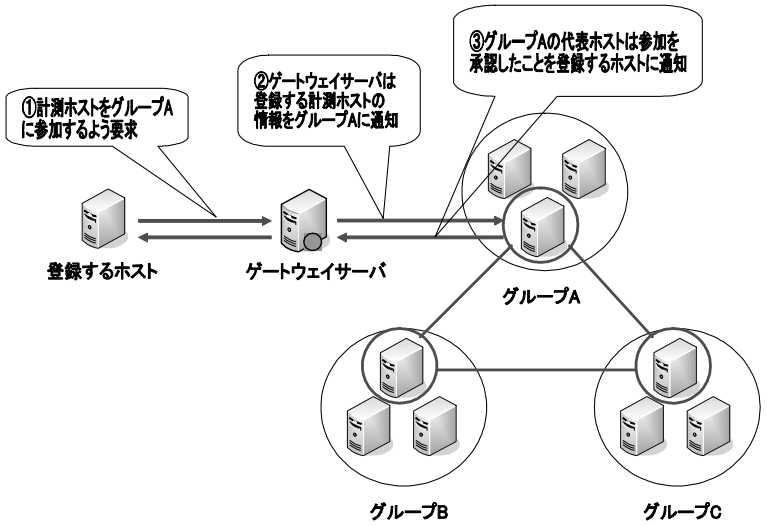

図 4.3. 計測ホストのグループへの参加

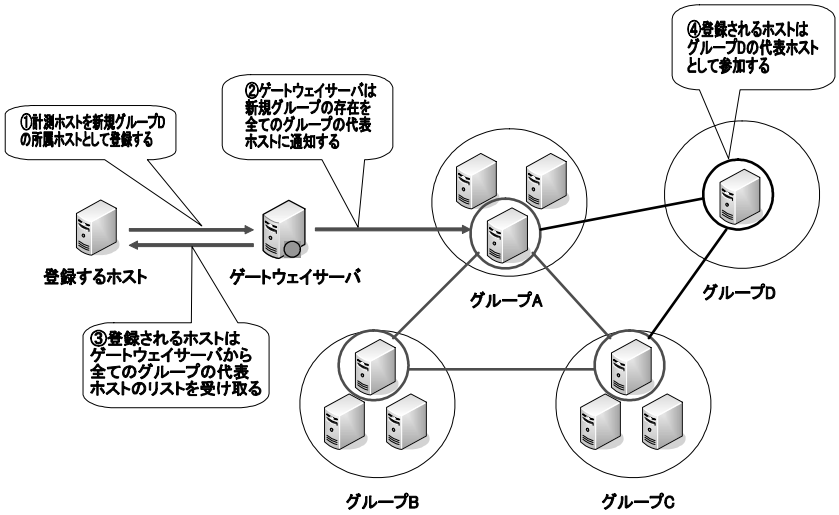

図 4.4. 新規グループの登録

ゲートウェイシステムは、新規登録したタグと計測ホストの情報を全てのグループの代表ホストに通知する。新規グループ作成時に最初に関連付けされた計測ホストは自動的に代表ホストとなり、他のグループの代表ホストのリストをゲートウェイシステムから受け取り、広域計測システムネットワークに参加する（図 4.4）。

表 4.2.　シナリオ記述言語の命令セット

命令	機能	引数
SendCmd(Src, Dst, type, Intr, opt, term) return ID	グループ間計測	Src: 送信側グループ Dst: 受信側グループ type: 計測内容 Intr: 計測間隔 opt: オプション term: 計測期間
NowReceive(Array[ID])	実行中の計測結果の取得	Array[ID]: 計測命令の ID のリスト
StopCmd(Array[ID])	実行中の計測の中止	Array[ID]: 計測命令の ID のリスト
Reply() return data	計測結果の受信	
data	計測結果取得用構造体	構造体要素 ・送信側グループ ・受信側グループ ・タイムスタンプ ・送信側選出ホスト ・受信側選出ホスト ・計測した値

- 代表ホストの交代

  グループ作成時に最初に関連付けされた計測ホストが代表ホストになるが、代表ホストがなんらかの原因で機能しなくなった場合には代替ホストを選出し代表ホストの交代を行う。代表ホストは、代替ホストをグループから複数選出する。その際、代替ホストには優先順位を通知しておく。代替ホストは、代表ホストが保有するタスクスケジューリング表などのデータの複製を持つ。

4.4.3 シナリオ記述言語とインタフェース

多数のノードを用いて広域ネットワークの計測を行うには、システム全体を操作するためのインタフェースが必要となる。本システムでは、ユーザの異なる要求やグループ単位での計測ホストの操作をシナリオ記述言語で表現する。ユーザはシナリオを記述しシステムへ投入する。インタプリタによって、シナリオは解釈されシステムのタスク管理表へ登録される。ノードやデータや計測ツールの扱いをシナリオ記述言語により一本化することで、ユーザビリティの向上を図る。一度、記述したシナリオは登録され、後から使うことも可能である。

シナリオ記述言語の命令セットとして表 4.2 を用意している。これらの命令セットを用いて、計測内容の記述を行う。シナリオ記述言語を用いた計測内容の記述の例を図 4.5、4.6 に示す。

```
[計測内容]
グループA/B間の可用帯域を計測し，可用帯域
30Mbps以上のパスのジッターと遅延を計測
（計測中に削減可能な計測内容を削減していく）

//引数が0の場合は標準の設定で計測が行われる
//term=0の場合は即座に計測
//optは計測内容ごとに指定できる計測回数など
SendCmd(GrpA,GrpB,ABW, 0,0,0)
if(ABW > 30){
        data = reply()
        SendCmd(data.src,data.dst,Jitter, 0,0)
        SendCmd(data.src,
```

図 4.5.　計測内容記述例 (1)

```
[計測内容]
グループA/B間の帯域をLevel3で2時間間隔で12/01-12/10まで計測
-オプションとして一回の計測は5分間かけて行う
その後，途中で中止し，即座に計測結果を返す

Opt = 5
Term.start = 12/01
Term.stop = 12/10
SendCmdIn(GrpA,RTT,120,Opt,Term)
―――――――――――――――――――――――――

//各計測内容はIDで管理されている
//IDのリストを渡すことでまとめて処理する
StopCmd(Array[ID])
NowReceive(Array[ID])
```

図 4.6.　計測内容記述例 (2)

4.4.4 タスクスケジューリング機能

本システムは、タスクスケジューリング機能を有しており、現在の状況を調べたい場合や、長期間に渡り計測を行いたい場合などにも対応している。また、タスクスケジューリング機能により、重複しているタスクを集約することも可能である。

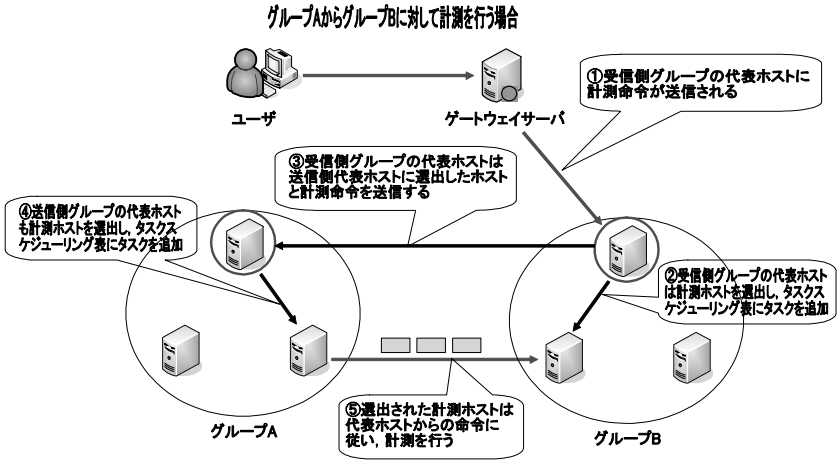

図 4.7. スケジュール管理のフロー

表 4.3. 送受信タスクスケジューリング表

送受信タスクスケジューリング表	保有する情報
送信タスクスケジューリング表	送信側グループ 受信側グループ 選出された送信側計測ホスト 選出された受信側計測ホスト 計測内容（RTT，jitter，...） 計測期間
受信タスクスケジューリング表	送信側グループ 受信側グループ 選出された受信側計測ホスト 計測内容（RTT，jitter，...） 計測期間

- グループ単位でのスケジュール管理

  タスクのスケジュール管理をグループ単位で行っている。ユーザがタスクを投入すると、計測時に受信側となるグループに計測命令が送信される。受信側グループの代表ホストは、投入されたタスクをタスクスケジューリング表に追加し、同一グループ内から計測時に受信側ホストとなる計測ホストを 1 台選出し、送信側グループに計測命令と共に通知する。送信側グループも、受信したタスクをタスクスケジューリング表へ追加し、計測ホストを 1 台選出する（図 4.7）。
- タスクスケジューリング表

  タスクスケジューリング表は受信タスクスケジューリング表と送信タスクスケジューリング表（表 4.3）の 2 つから構成される。受信タスクスケジューリング表は、グループが計測時に受信側となる場合のタスクを管理する。送信タスクスケジューリング表は、グループが計測時に送信側となる場合のタスクを管理する。ユーザからタスクが投入されると、計測時に受信側のグループの代表ホストが計測命令を受信する。タスクを受信した代表ホストは、送信タスクスケジューリング表を参照する。これは、競合する計測内容の有無を確認するためである。次に、受信タスクスケジューリング表を参照し、重複する計測内容の有無を確認する。同じ計測内容がスケジュールにあればタスクを追加せず、なければ追加する。また、計測内容が一部重複している場合は、その部分だけタスクを集約する。送信側のグループの場合、まず、受信タスクスケジューリング表を参照する。そして、競合する計測内容の有無を確認する。次に、送信タスクスケジューリング表を参照し、重複するタスクの有無を確認し、なければ、タスクを追加し、計測時の送信側ホストを 1 台選出する。計測ホストの選出は、タスクスケジューリング表を参照し、計測タスクを割当てていないホストを優先して利用する。

表 4.4.　計測ログデータベース

データベース	保有する情報
計測ログデータベース	送信側グループ 受信側グループ タイムスタンプ 選出された送信側計測ホスト 選出された受信側計測ホスト 計測内容（RTT，jitter，...）

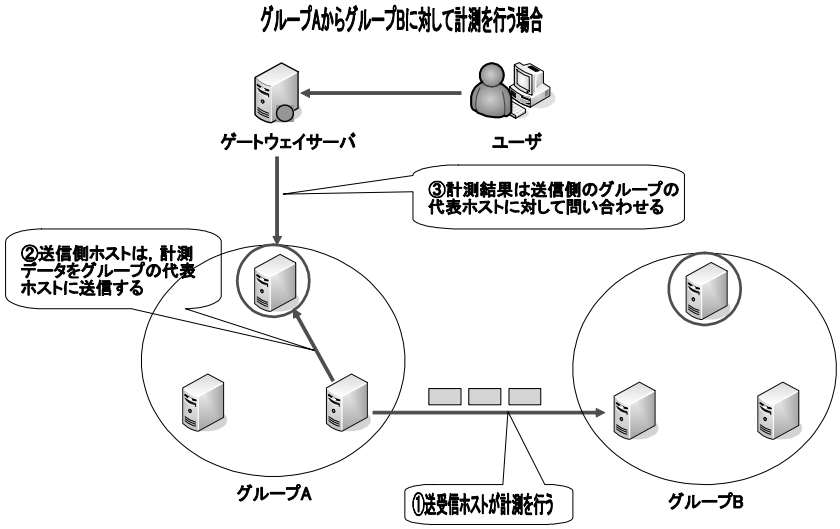

図 4.8.　計測データの保存と検索

4.4.5 データの分散保存

多数の計測ノードを用いて広域ネットワークの計測を行う場合、膨大な計測データを蓄積しなければならない。本システムでは、計測データをグループ単位で管理することにより、計測データを分散管理する。

- 計測データの保存
  計測データは、計測時の送信側ホストが取得する。送信側ホストは取得した計測データをグループの代表ホストに送信する。代表ホストは、受け取った計測データを表 4.4 に示す計測ログデータベースに保存する。計測データは、計測内容、送信側グループ、タイムスタンプ、受信側グループ、送信側ホスト、受信側ホスト、計測した値を一つのレコードとして保存する。代表ホストは、代替ホストに計測データを複製する。
- 計測データの検索
  ユーザは計測を行ったグループ名などの検索条件を指定したシナリオをシステムに投入する。計測を行ったグループの代表ホストは、データベースを検索し、持っていれば計測結果を返信する（図 4.8）。

4.5 設計と実装

システムのモジュール構成を図 4.9 に示す。本システムは計測を行う計測ホストとシステムやグループを管理する代表ホスト（以下、管理ホストと呼ぶ）から構成される。システムに参加するすべてのノードは、管理ホスト、計測ホストのいづれかに該当する。管理ホストは計測ホストの中から選出されるた

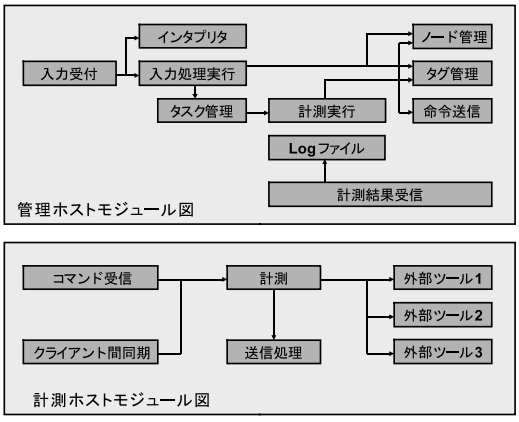

図 4.9.　広域計測システムのモジュール構成

め、ノードは管理ホスト、計測ホスト両方の機能をもつ。

- 管理ホスト　管理ホストは、ユーザからの入力受付、計測ホストの管理、グループ管理、計測データの管理を行う。
- 計測ホスト　計測ホストは、管理ホストからの命令により計測を実施し結果を管理ホストに返信する。

設計に基づき本提案のプロトタイプ実装を行った。本システムは、Fedora Core 5 上で、Ruby を用いて実装した。

4.6 最後に

本報告書では、広域ネットワーク上で多地点に設置されたノードが、ユーザの記述したシナリオによって計測を行うためのシステムの提案を行った。今後は、プロトタイプシステムの実装をすすめ、提案システムの評価を行っていく予定である。また、シナリオ記述言語はプロトタイプシステムのための必要最低限な命令セットしか定義していない。ユーザの異なる要求に対応するためには、シナリオ記述言語の拡張及び、インタプリタの実装を行う予定である。

第 5 章　まとめ

PlanetLab ワーキングループでは、WIDE プロジェクトや日本における PlanetLab の運用及び研究利用を行っている。今年度は、WIDE プロジェクトノードの設置を行った。また、今年度中に JGN II ノードの設置を予定している。広域アクティブ計測基盤を構築し、PlanetLab ノードを利用して動作検証を行った。