階層型 P2P プロトコルを用いたクラスタシステム状態通報ツール

Machine information conveying system using layered-type P2P protocol

上川 純一
Junichi Uekawa

Abstract: An application to monitor machines within the Cluster was developed. To make this task feasible, an m-ary tree topology was adopted, and to defend against failures in specific key nodes of the topology, the system connection is done in the P2P style, where node connection is rearranged on the fly.

1　はじめに

近年 P2P システムが注目されていて，さまざまなシステムに P2P の手法が適応されている．

P2P の新しいトポロジを模索するために，階層化された P2P 的プロトコルを考察した．その上にクラスタ内の各ノードの情報を通報するシステムを構築した．

2　P2P システムの目標

P2P システムとして構築する場合の目標としてはさまざまであるが，ここでは以下のことを重視してみる．

- ノードシステムが簡単に設定できる
- フェイルセーフである
- 利用者がノードである事による実行負担を感じない
- 巨大なシステムによる集中管理を分散する

3　dmachinemon とは

dmachinemon とは，クラスタシステムの現在の状況というものをモニタするアプリケーションであり，　現在の利用者のノードの利用状況をさまざまな形態で提示する機能を主として，それから派生した機能を有するシステムである．

複数のプログラムからなり，基本的には，クラスタシステム内の各ノードで動く「ノード」ソフトウェア，クラスタシステムのマスターノードで動く「マスター」ソフトウェア (サーバ)，そして，ユーザが利用している端末上で動く「クライアント」ソフトウェアの三種類に大別される．

現在存在する「クライアント」ソフトウェアには，マスターソフトウェアからテキスト形式で情報を取得するものから，XML で提示するもの，HTML 形式で提示するもの，GTK+で提示するもの，そして，JAVA3D を利用して提示するものなどがある．

この「ノード」に今回提案するトポロジを導入，実装した．

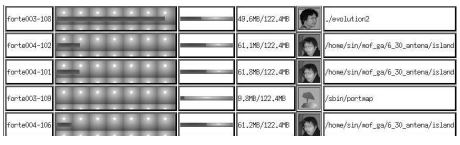

Fig. 1　HTML output application output displayed through mozilla

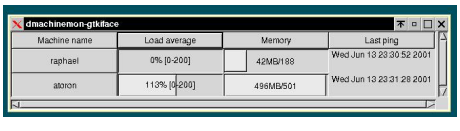

Fig. 2　A GTK+ interface to the system monitoring system

4　dmachiemon の基本的な機能

dmachinemon においては，それぞれのノードでシステム情報収集プロセスが動作し，そして，システム情報を取得し，マスターノードに通報するというメカニズムになっている．システム情報収集というのは，現在の CPU 利用率，メモリ使用率，そして最もシステムを利用しているユーザの名前，そしてプロセスの名前，等を収集することを指す．その情報を各ノード上で収集した後，マスターシステムに情報が転送される．マスターシステム上に集められた情報は，クライアントソフトウェアが取得し，情報をユーザが利用しやすいようにプレゼンテーションする．

その情報の提示用のアプリケーションは多様に存在する事が可能であり，用途に合致した情報の提示ができる点が利点であると考えられる．アプリケーションの表示画面例を Fig. 1 と Fig. 2 に提示する．また少し用途が違うが，トポロジの情報を表示するツールも JAVA3D で作成された (Fig. 3，Fig. 4)．

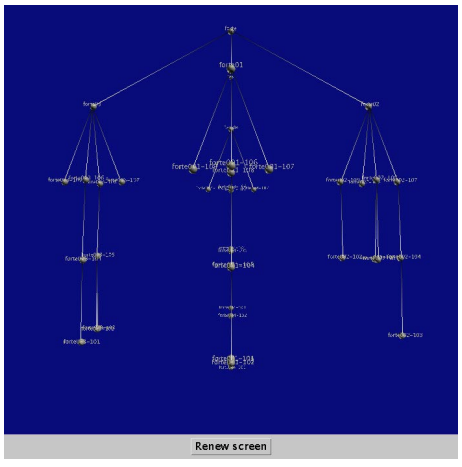

Fig. 3　Java3D interface to show the current topology

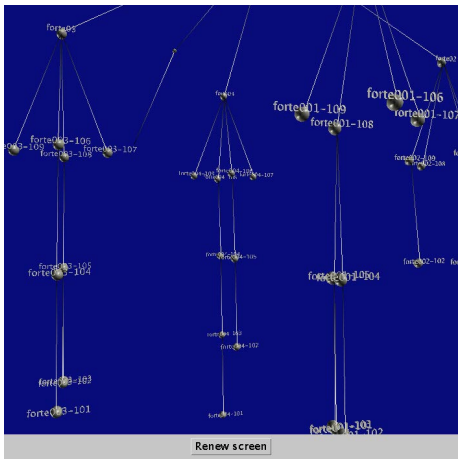

Fig. 4　Java3D interface with viewpoint moved around.

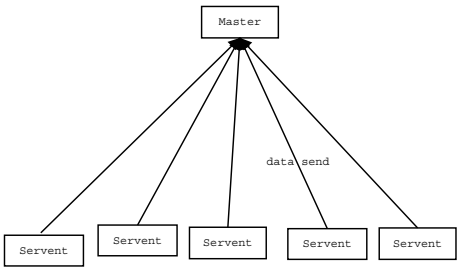

Fig. 5　Client/Server system structure for dmachinemon

5　dmachinemon の当初のトポロジ

dmachinemon は当初はクライアントサーバ型のクラスタノードモニタシステムとして設計された (Fig. 5). しかしながら，その設計では，知的システムデザイン研究室においての大規模クラスタである Cambria システム (256PE で 100BASE-TX で相互接続) のモニタとして利用しようとした場合，マスタノードに対しての負荷が大きすぎると考えられた．実際問題として，Cambria システムは，64 台単位 (以下セグメントと呼ぶ) の構成になっており，全ノードから均等に通信が行える構成にはなっていなく，セグメント間での通信は内部より高価であると考えられる．

そこで，そのような特性を有効に利用できるように考えられたのが，階層化構造 (Fig. 6) である．この形態にすることによって，クライアントソフトウェアに変更を加えることなく，マスターノードの負荷を軽減する事ができる．クライアントから見ると，アクセスすべき「マスター」ノードは一つに固定されていて，そこから全体の情報が取得できることには変わりが無いからである．

しかしながら，これには一つ問題がある．集中的なクライアントサーバシステムでは，一台のサーバシステムが「重要ノード」として存在しており，そのノードさえ動いていれば，全体としてのシステムは稼働するというシステムであったが，階層構造の場合，中間の階層に存在するノードが停止すれば，そのノードが仲介していたノード全ての機能が停止する．つまり，階層構造にした時点で複数の「重要ノード」が生まれるのである．中継をするノードが欠落すると，そのノードが中継をしていたノード全体のノードの情報が伝達されなくなるのが問題である．

6　現在の dmachinemon システムのトポロジ

今回採用したシステムは，この，階層構造を動的に生成するようになっている．つまり，どのノードがどの経路たどってマスターノードまで情報を伝達するか，とい

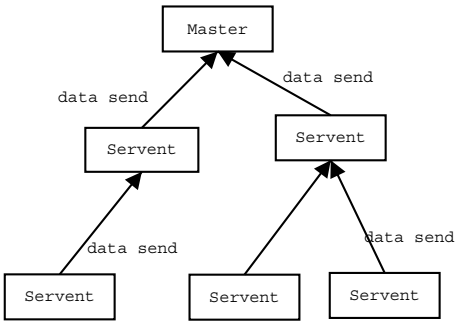

Fig. 6　Layered system structure for dmachinemon

うことが，動的に決定される．

このシステムには，マスターノードが一つ存在し，その下に複数のシステムが存在していて，お互いに階層構造になっている．ここで，初期状態は Fig. 8 に示すようになっているとする．

この構造には中間ノードの存在に依存するという脆弱性がある．この問題を解決するためには，中間ノードが欠落した場合に，下流のノードを救済する手法が必要である．ここで，下流のノードは，中流のノードを飛び越えて，より上流のノードに再接続しようとするようにした．これにより，中間ノードが一台欠落しても，情報の欠落は，一台に制限される．

これを実現するために，三つの情報が追加された．

- Seen-By: そのパケットを見た事があるノードのリストを保持す る．下位から上位にむけて送信
- Route-To: 上位から下位に与えられるもので，上位がどのよう に接続されているのかという情報を提供
- Data-Seen: データを複数回送信することを防ぐための情報で， 実際に送信したかどうかを記憶

`Route-To` は`dmachinemon` システムにおいて，唯一上流から下流にむけて伝達される例外的な情報である．Route-To 情報を利用することにより，自分より上流にどのようなシステムが存在しているのか，ということが分かる．この情報を利用すると，中間ノードが欠落した場合 (Fig. 9) でも，それによりサーブされていたノードは新しいノードを探し求める事ができる (Fig. 10)．

また，再接続を行った後に上流のにあるシステム一台に負荷がかかり状況が起きないように，自分が直接つながっているシステムが多すぎると，それを再接続させる機構も導入した．上流のシステムは，自分の下流のノードの中から適当なノードを選択し，そのノードに上流ノードとして自分の下流ノードを利用するように指定する (Fig. 11)．

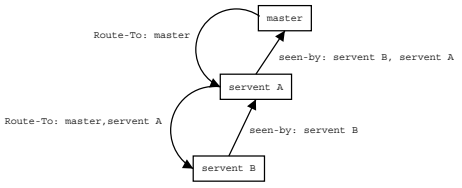

Fig. 7　Route-To: and Seen-by:

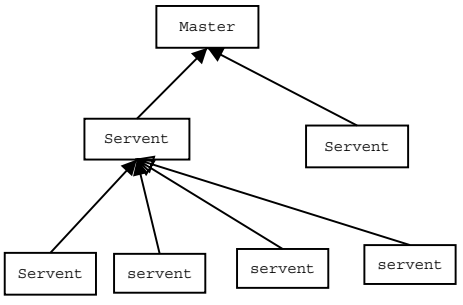

Fig. 8　Initial position

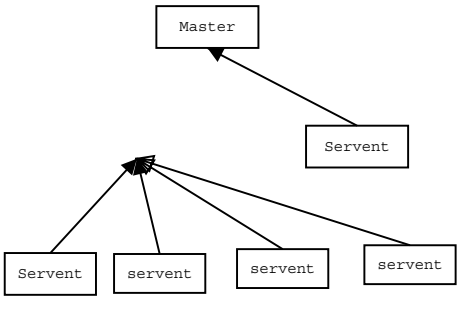

Fig. 9　Uplink is missing

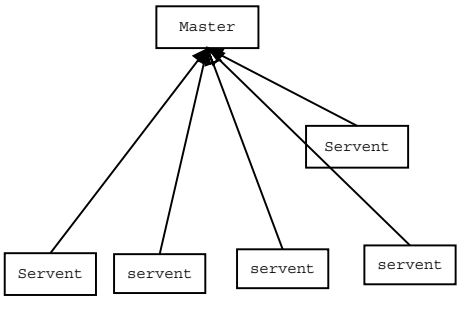

Fig. 10　Find an uplink, and relinking

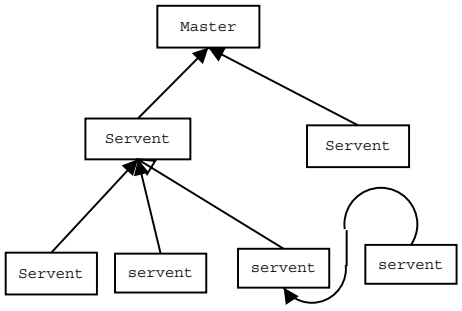

Fig. 11　Redirecting the direct downlink

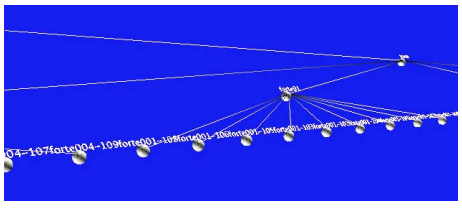

Fig. 12　Original tree structure

実際にクラスタシステム forte(40PE) で実行した時の画面例を提示しておく．forte では，初期状態として，マスターノードに 4 つの中間ノードが接続し，各中間ノードに対して 9 のノードが接続するように設定されている．そして，各ノードは 4 ノードまでしかサーブしないというように設定してある．そうすると，初期状態は Fig. 12 に示すようになっているのだが，数段の変形を経た後，Fig. 13 に示す状態に到達する．

7　複数のトポロジの情報転送量の比較

t とは単位時間あたりの通報回数，k とは通報一回の情報量，そして，n とはネットワークに存在するクライアントシステムの数であるとすると，クライアント一台は単位時間において，tk の情報を伝達しようとするので，クライアントサーバ型のシステム全体としてネットワーク上を伝達される単位時間あたりの情報の転送量は，式 1 のようになる．

$$\mathbf{I} = tkn \tag{1}$$

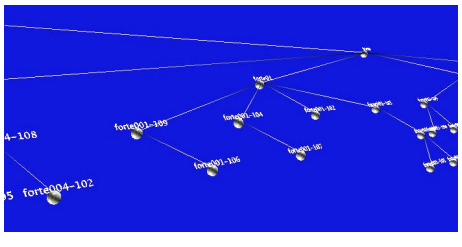

Fig. 13　After automatic rearrangement

これは必要最低限の情報転送量であり，理想的な量である．

一般的な Gnutella 型のネットワークはそれぞれのノードが他のノードに対して情報を発信する．ある単位時間において，発信される情報の量は，式 2 に示す通りである．それが各ノードにより中継される事になるので，総合の情報量は式 3 になる．

$$\mathbf{I} = tkn \tag{2}$$

$$\mathbf{I} = tkn^2 \tag{3}$$

`dmachinemon` において，m という，各ノードがいくらまでのノードをサーブするか，という値を定義する．これは，システムにおいて一定であり，ツリーは m 分木であると仮定すると，l 段の木においての情報通信量は，式 4 になる．ここで n 台のシステムがある場合のシステム全体の通信量は，式 5 になる．

$$\mathbf{I}_l = tlkm^l \tag{4}$$

$$\mathbf{I} = \log_m nkm^{\log_m n} = tkn \log_m n \tag{5}$$

以上をまとめると，C/S システムでは通信量は $O(n)$ であるが，gnutella 式 P2P システムでは本質的な通信量は $O(n^2)$ になる．これらに対して，`dmachinemon` において採用したシステムにおける通信量は，一方向に通信方向が制限されている特性も影響しているのか，$O(n \log n)$ になる．

8　展望

ここまでで，効率良くシステム全体の状況が通報できるシステムが構築された．そして，一部のクラスタ上で稼働実験を行い，実際にこのシステムが動作している．このシステムが動作していることにより，あまりシステム自体に負荷をかけることなく，システムの利用状況がわかるという利点があり，一般的に利用状況のわかりにくいクラスタシステムにおいての状況通報システムとして利用できると考えられる．

このシステムに対して，ユーザプログラムが情報を通報するためのルーチンを追加するのが容易になる API を作成し，それを利用して，さまざまなアプリケーションの上でのモニタシステムが構築されることが期待される．

また，このシステムを利用して，情報交換を行い，P2P システムで計算を行うシステムを構築することが可能なはずである．