シミュレーションによる分散ファイルシステムの性能評価

5G−2

金澤 裕治 村上 岳生 湯原 雅信
(株) 富士通研究所

1．はじめに

経済性・耐故障性などの理由から，計算機システムにおける資源共用の必要性が認識されつつある．中でも，長期記憶装置と，そのデータを共用することの必要性は大きく，実際，ＮＦＳなどに見るように，資源共用は長期記憶の共用から実用化されてきた．しかし，今後は，ＣＰＵ資源の共有の必要性・ユーザーモビリティ・メンテナンスのしやすさなどの要請から，単にデータを共有するだけではなく，ネットワーク内の各計算機が全く同一の環境にあるように見えるような，ネットワーク透過性を有した分散ファイルシステムが必要になると考えられる．

我々は，ＵＮＩＸをベースとした分散ＯＳの研究を行なっている。分散ファイルシステム設計のために，分散ファイルシステムの性能予測が必要になるが，その性能は，アプリケーションがファイルシステムをアクセスするパターンに強く依存しているため，シミュレーション以外の方法で性能予測を行なうことは難しい．今回，そのためのシミュレーターを作成した．その１例として，分散makeのシミュレーションについて報告する．

2．シミュレータの機能

分散ファイルシステムの性能評価用シミュレーターの入力と出力として，以下のものが挙げられる．

入力：

ネットワーク・ディスク・ＣＰＵの速度，台数，ファイルシステムのブロックサイズなどのパラメータ

アプリケーションのファイルシステムアクセスパターン

出力：

評価したいファイルシステム上でそのアプリケーションを動作させたときの，実行時間・キャッシュヒット率

(可能ならば) 誤差

また，キャッシュアルゴリズムなどの変更，シミュレートするシステムの構成の変更が容易にできるものが望ましい．

3．シミュレータの構成

分散システムでは，性質の異なる複数のプロセスが，プロセス間で通信を行なったり排他制御を行なったりしながら，ＣＰＵ・ネットワークなどのハードウェア資源を順番に占有していくことにより動作が行なわれる．そのため，プロセスの論理的進行をシミュレートする論理モジュールが，物理デバイスをシミュレートする物理モジュールに作業を依頼しながらシミュレーションを進行させる手法を採用した．例えば，論理モジュールがＣＰＵ時間を消費するときは，ＣＰＵモジュールにメッセージを送り，指定したＣＰＵ時間が消費され，終了メッセージが返されてくるまで待つということを行なう．

これらのモジュールの他に，スケジューラーが１つ存在し，メッセージ・ロックなどのモジュール間の通信を，全てスケジューラーを介して渡すことにより，時間を管理している（図１）．

論理モジュールと物理モジュールに分けることには，以下のような利点が存在する．

(1) システムの構成・方式の変化に対応できる

例えば、ネットワークに接続しているホスト数を変える場合、最初に生成するＣＰＵモジュールの数を変えるだけでよい．

また，ユーザーがＣＰＵモジュールのプログラムを変更することにより，種々のスケジューリングのシミュレーションを行なうことも可能である．同様に，メモリを物理モジュールとして考えれば，ページングもシミュレートすることができる．

(2) プログラミングが容易になる

論理モジュールは，ＯＳ内のプロセス（スレッド）のイメージに近いので，自然にプログラムを行な

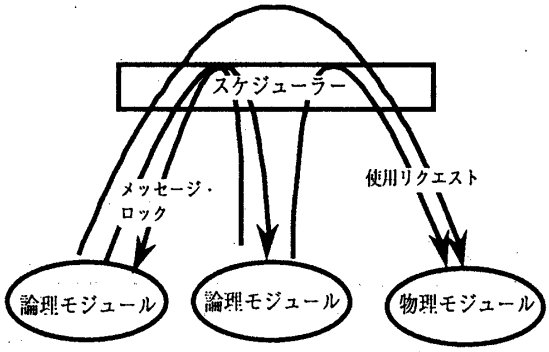


図１　シミュレーターの基本構成

Simulator for Distributed File Systems.
Yuzi Kanazawa Takeo Murakami, Masanobu Yuhara
Fujitsu Laboratries Ltd.

うことができる．

しかし，論理モジュールの動作が，実際のシステムでのプロセスの動作と完全に一致している必要はない．ハードウェアの制約により，１つの論理的流れを２つ以上のプロセスに分けている場合，実際のプロセスの流れではなく，論理的流れの方を重視することができる．

例えば、クライアントのＣＰＵモジュール，ネットワークモジュール，サーバーのＣＰＵモジュールと順に要求を送ることにより，ｒｐｃのシミュレートを１つの論理モジュールで行なうことも可能である．サーバー上にｒｐｃを処理するプロセスが１つしかない場合のシミュレートを行なう場合は，サーバー上で処理される部分でロックをかけることによって可能である．

４．シミュレータの実装

SUN LWPを使って実装を行なった．

各ホストのＣＰＵ・ディスク・ネットワークをそれぞれ１つの物理モジュールとし，各モジュールとスケジューラーを、１つのlwpスレッドとしてインプリメントしている。

各論理モジュールは，システムコールと，システムコール間のユーザー時間を記録したデータを読み込み，それに従って動作を行なうことにより，アプリケーションの動作をシミュレートする．このデータは，プローブを埋め込んだＯＳ上で実際にアプリケーションを動作させて得たものを使用している．

コード量はシミュレーターの基本部分が１５００行，ファイルシステムのシミュレーションを行なうためのコードが３０００行程度である．

５．シミュレーション結果

Ｘのライブラリをネットワーク上で分散makeをシミュレーションした結果を示す．

システムの構成として，25 MIPS程度の計算機がイーサネットで接続されている場合を想定している．ファイルのライト共有がないので，一貫性プロトコルは使用していない．シミュレーションに使用したキャッシュのヒット・ミスヒット時のＣＰＵ時間・実時間などのパラメータは，以前測定したベンチマーク［１］での実測値を使用している．

グラフ１がテンポラリファイルのためのディレクトリ(/tmp)をサーバー上に置いた場合，グラフ２は/tmpを各クライアントがローカルに持っている場合のシミュレーション結果である．

/tmpを共有した場合は６台，共有しなかった場合は10台程度から台数効果が出なくなることが示されている．

これは我々が得ているＳＵＮＯＳでの実際の測定値より多少悪い。このシミュレーションではディスク内のブロック配置のオプティマイズを行っていないために、実際のシステムに比べてディスクアクセスに時間がかかっていることが原因だろう。

６．おわりに

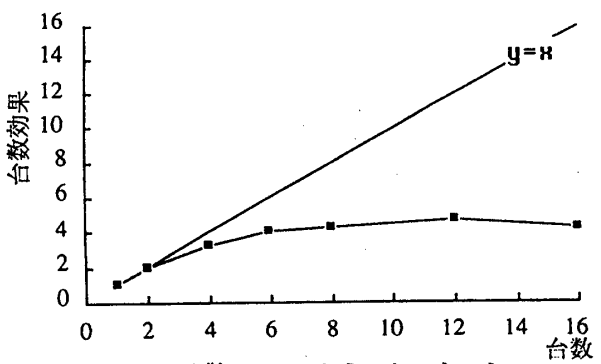
分散ファイルシステムのシミュレーションに適したシミュレーターを作成した．論理モジュールと物理モジュールを区別したことにより，ネットワーク形状やキャッシングアルゴリズムを変更してシミュレーションを行なう場合でも，柔軟に対応することが可能である．

今回報告した分散makeのシミュレーションだけではなく，ファイル内容のキャッシングのシミュレーションも現在進行中である．

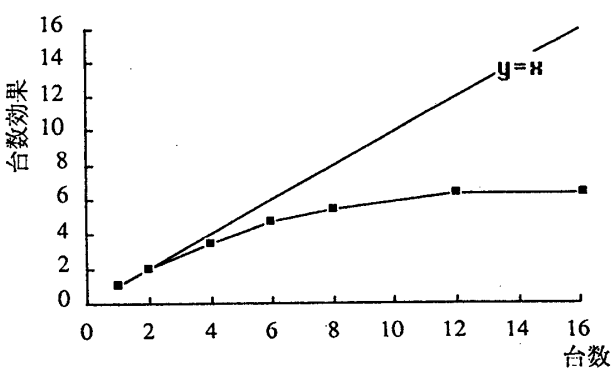
これからの課題としては，大規模な分散システムもシミュレートできるような，並列実行を行うことや，現在の版では行なっていない誤差の解析も行なうシミュレーターを開発していくことが挙げられるだろう．

参考文献

[1] 金沢,岸本,湯原："ＵＮＩＸファイルシステムの性能測定",第４３回情報処理学会全国大会.



グラフ１　分散makeのシミュレーション
(/tmp 共有)



グラフ２　分散makeのシミュレーション
(/tmp ローカル)