プラットフォーム独立な教育用分散システムの性能評価ツール

山之上 卓†

プラットフォーム独立な教育用分散システムの性能評価ツールについて示す．このツールは，教育用分散システムの各端末で行われるアプリケーション操作を記録しておき，その操作を様々な分散システムで再生したときのアプリケーションの応答時間を計測するものである．多数の端末で同じ操作をいっせいに行って，ファイルサーバや Web サーバに負荷をかけたときのアプリケーションの応答時間の計測を行うこともできる．教育用分散システムの性能評価を行うため，このツールは，お絵かきソフト，テキストエディタ，簡単なプログラミング環境，Web ブラウザなど，教育現場でよく利用されるアプリケーションを備えている．このツールは Java で開発されておりプラットフォーム独立である．したがって，異なる種類の分散システム上で，まったく同じ操作をさせ，そのときの応答時間やファイルの読み書きの時間などを計測することによって，分散システムの性能の比較を行うことも可能である．

A Platform Independent Tool for Evaluating Performance of Educational Distributed Systems

Takashi Yamanoue†

A platform independent tool for evaluating performance of educational distributed systems is shown. This tool records real operations by users on computing equipment and it acquires the performance data of a distributed system by replaying the recorded operations. This tool can replay the same operations on different computing equipment. This tool also can let every computer doing the same thing simultaneously. In order to evaluate the performance of educational distributed systems, this tool equipped with applications such as a draw, a text editor, a simple programming environment and a web browser which are often used on educational distributed systems. In order to compare the performance data of various kind of computing equipment, this tool is in Java. So it is platform independent.

1. はじめに

情報処理教育用コンピュータシステムの新規導入や更新は，学校の情報処理教育用システムの管理運営を行う組織の最も重要な仕事の 1 つである．そこでは，多くの教職員や学生の要求を満足する要求仕様を作成しなければならない．その後，その要求仕様を満たす機器構成を設計しなければならない．このとき，この設計は要求を満たし，なおかつ予算の枠内に入るように考慮しなければならない．それと同時に，それが本当に動くかどうかについても考慮しなければならない．これらは多くの労力と時間を必要とする仕事である[1),8)〜10)]．

今日，情報処理教育用コンピュータシステムの多くは多数のコンピュータとネットワークで構成された分散システムである．分散システムの構成要素であるコンピュータやネットワーク機器のそれぞれについて，その性能を評価することは比較的容易であるが，分散システムの総合的な性能を評価することは難しい．

情報教育用コンピュータシステムを使って講義を行うとき，すべての学生生徒が同時に同じことを行うことは珍しいことではない．このとき，大量のデータがネットワークを流れ，大きな負荷がファイルサーバや Web サーバにかかることがある．このような場合でも講義が続けられるよう，要求仕様には，「200 台のクライアントが 0.5 秒内に同一の Web ページにいっせいにアクセスした場合，その Web サーバは，すべてのクライアントに 3 秒以内でページを表示できなければならない」というような記述を行いたい．そして，我々は，このような要求を満たすサーバがどのようなものかを知りたい．このような要求仕様を作るためにも，現存の分散システム上で，実際の授業でどのようなことが行われているかを示すデータや，このときの

† 鹿児島大学
Kagoshima University

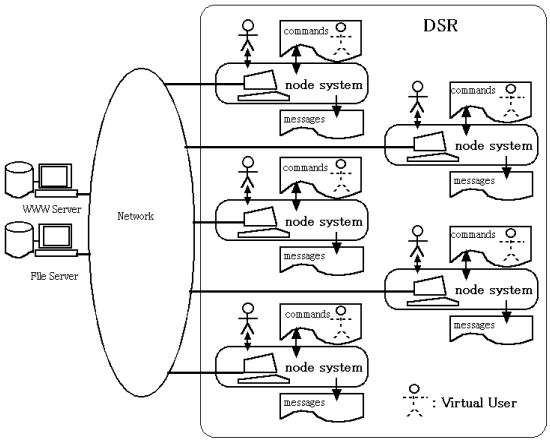

図 1 DSR と仮想的な利用者
Fig. 1 DSR and virtual users.

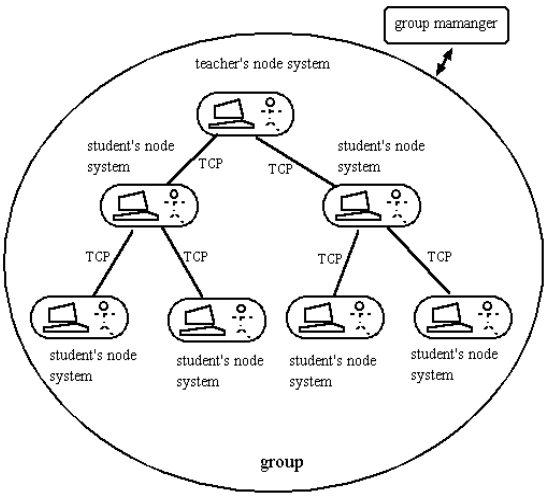

図 2 DSR の構成
Fig. 2 Structure of DSR.

分散システムの状況を示すデータが分かると便利が良い．実際の授業で行われる操作を異なる種類の分散システムで実行し，その性能データを比較できると，より有効である．

このようなデータを得るため，我々は，DSR（Distributed System Recorder）と名づけたベンチマークテストツールの開発を行っている．DSR は，分散システム上のユーザの操作を記録し，再生し，そのときの性能データを得るツールである．DSR は異なる分散システムで同じ操作を再生することができる．DSR は，数十台の端末コンピュータで，いっせいに同じ操作を行わせることもできる．

様々な種類の分散システムで動作させるため，DSR は Java を使って開発している．したがって，DSR はプラットフォーム独立である．

本論文では，DSR の構成と，DSR を試験的に利用した実験について述べる．

2. DSR

DSR は，分散システムの端末コンピュータで行われる実際の利用者によるアプリケーション操作を記録することによって，その操作を行う仮想的な利用者を作り，様々な分散システムの端末コンピュータに，その仮想的な利用者を配置し，仮想的な利用者がその分散システム上でアプリケーションの操作を行ったときの応答時間を出力することによって，様々な分散システムの性能を評価しようとするものである．DSR は利用者に対応する「ノードシステム（node system）」などで構成されており，仮想的な利用者は各ノードシステムの上で動作する（**図 1**）．

1 台の端末コンピュータで行われる現実または仮想的な利用者の操作を，多数の端末コンピュータに配信し，多数の端末コンピュータでいっせいに，その操作を行わせることもできる．このことによって，あらかじめ定められた操作を行わせるだけでなく，いっせいに端末コンピュータを操作した結果を見ながら，次のいっせい操作を行うこともできる．この章では，DSR の構成と，DSR の中で利用者に対応するノードシステムの構成について述べる．

2.1 DSR の構成

DSR は「教師ノードシステム（teacher's node system）」,「学生ノードシステム（student's node system）」,「グループマネージャ（group manager）」の 3 種類のプログラムで構成されている．**図 2** に DSR の構成を示す．

2.1.1 教師ノードシステム

教師ノードシステムは教育用分散システムの教師用端末を使用する教師に対応している．このノードシステムは教師の操作を記録し，その操作を再生し，そのときの遅延などのデータを記録する．教師ノードシステムはすべての学生ノードシステムに対して教師ノードと同じ操作を同時にさせることもできる．これは，教師の操作に対応したコマンドを教師ノードシステムからすべての学生ノードシステムへ信頼性を持って放送することにより実現している．

2.1.2 学生ノードシステム

学生ノードシステムは，教育用分散システムの学生用端末を使用する学生に対応している．学生ノードシステムは学生の操作を記録し，その操作を再生し，そのときの遅延などのデータを記録する．学生ノードシ

ステムは，教師ノードシステムから送られるコマンドを解釈実行することによって，教師ノードで行われる操作を学生ノードで再現することもできる．

2.1.3 グループマネージャ

グループマネージャは，教師ノードシステムと，学生ノードシステムのグループの結合状態を管理する．

教師ノードシステムがコマンドを放送するとき，すべての学生ノードシステムは短時間のうちに，間違いなくそのコマンドを受信しなければならない．これを実現するために，DSR は一種の P2P 技術を利用している．グループ内のすべてのノードシステムは，少なくとも 1 つの他のノードシステムと TCP で結合されている．1 つのノードシステムが他のノードシステムから TCP 結合を通じてコマンドを受け取ったとき，もしそのノードシステムがコマンドを受け取った TCP 結合以外の TCP 結合を持っていたら，その TCP 結合を通じて他のノードシステムへそのコマンドを送信する．その後，そのノードシステムは受け取ったコマンドを解釈実行する．

グループ内のノードシステムは，完全 2 分木状になるよう，TCP で結合される．教師ノードシステムは，この 2 分木の根のノードである．もしグループ内のすべての TCP 結合が，同時にデータを流すことができる場合，教師ノードシステムからコマンドが送信されてから，すべての学生ノードシステムがコマンドを受け取り終わるまで時間は，$O(logN)$ となる[3),4)]．ここで N はグループ内のノードシステム数である．

グループマネージャは，グループに参加しているノードシステムの結合状態を記憶している．新規ノードシステムがグループに参加するとき，そのノードシステムはまずグループマネージャに問合せを行う．これに対して，グループマネージャは，新規ノードシステムがグループに参加したときにグループ内のノードシステムの結合の形が完全 2 分木状になるよう，新規ノードシステムの接続先ノードシステムを探し出し，新規ノードシステムがグループに参加した状態を記憶し，新規ノードシステムに対して接続先ノードシステムの IP アドレスを教える．最後に新規ノードシステムが接続先ノードシステムに TCP 接続を行って新規ノードシステムのグループ参加手続きが終了する．

根に位置する教師ノードシステムが最初にグループに参加する．このとき教師ノードシステムは別のノードシステムに接続しない．

2.2 ノードシステムの構成

教師ノードシステムや学生ノードシステムは，「メインコントローラ」，「アプリケーション」，「イベントレコーダ/プレイヤ」，「コマンドトランシーバ」を含んでいる．**図 3** にノードシステムの概略を示す．

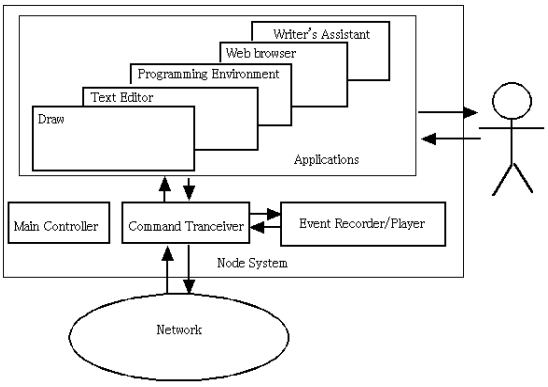

図 3 ノードシステムの構成
Fig. 3 Structure of the Node System.

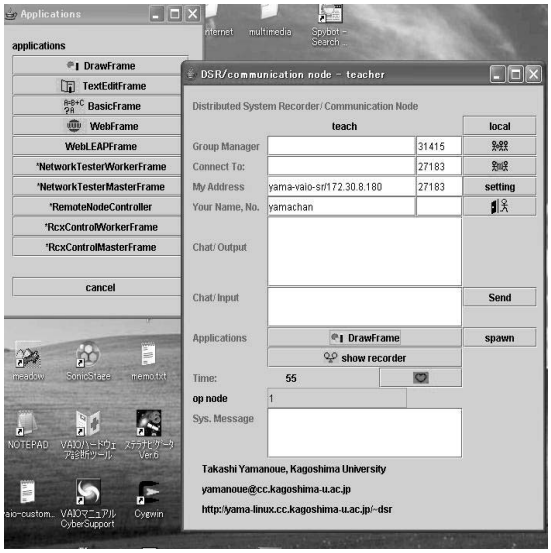

図 4 メインコントローラ
Fig. 4 Main controller.

2.2.1 メインコントローラ

メインコントローラはグループへの参加や離脱，アプリケーションやイベントレコーダ/プレイヤの起動，グループ内のチャットを行うプログラムである．**図 4** に，メインコントローラの GUI を示す．

2.2.2 アプリケーション

ノードシステムは自分自身でアプリケーションを備えている．そのため，ベンチマークテストを行うのに利用者は他のアプリケーションを用意する必要がない．また，このアプリケーションもプラットフォーム独立であるため，利用者は同じアプリケーションを使って様々な分散システムの性能比較を行うことができる．

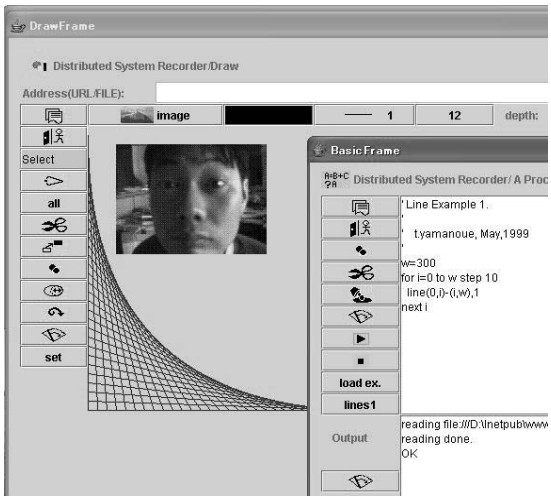

図 5 お絵かきソフトとプログラミング環境
Fig. 5 Draw and programming environment.

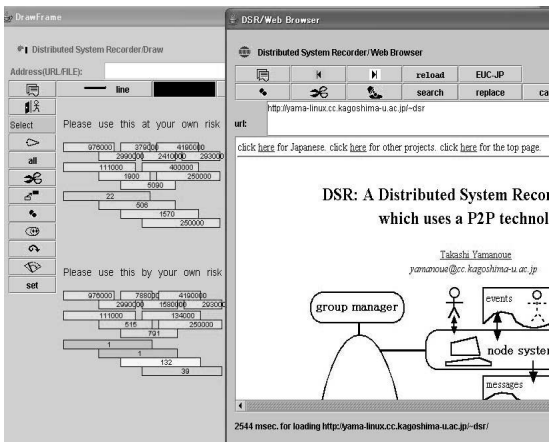

図 6 Web ブラウザと英作文支援システム
Fig. 6 Web browser and writing assistant.

実際の授業に似た状況を実現するため，教育現場でよく使われる，お絵かきソフト，テキストエディタ，プログラミング環境，Web ブラウザを単純化したアプリケーションを備えている．このほか，英作文支援システムも持っている．

図 5 にお絵かきソフトとプログラミング環境を示す．この図において，写真がお絵かきソフト上に貼り付けられており，その上に，プログラミング環境のプログラムが幾何学模様を描いている．

図 6 は Web ブラウザと英作文支援システムを示している．この英作文支援システムは，入力された文に含まれる文節の，インターネット上の出現頻度のグラフィカル表示や，KWIC（Key Word In Context）形式によるその文節の使用例の表示を行うものである[11]．文節のインターネット上の出現頻度として，その文節をサーチエンジンで検索したときに表示される件数を使用している．

教育現場では，キーボードによるコマンド入力によって OS の操作を教育する場合もある．しかしながら，利用できるコマンドは OS によって異なり，また，同じ OS を利用していても教育用分散システムによって同じコマンドが使えない場合があるため，現在の DSR では OS のコマンドを直接キーボードから入力できるようにはしていない．

2.2.3 アプリケーションとコマンド

利用者がノードシステムのアプリケーションを操作するとき，コマンドが生成される．このコマンドは，ユーザの操作によって発生するイベントに対応している．このようなコマンドの生成や解釈実行を行うた

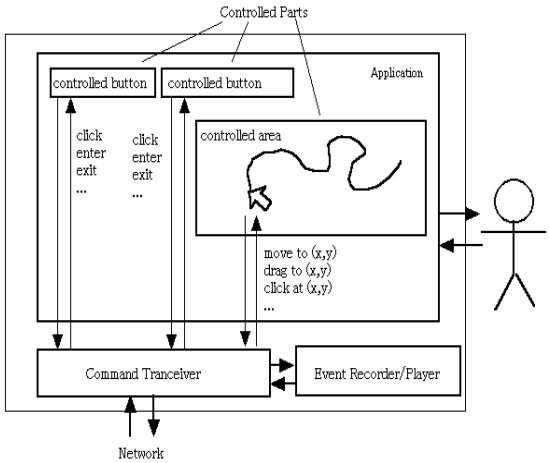

図 7 制御可能な GUI 部品とコマンド
Fig. 7 Controlled parts and command.

め，アプリケーションは，「制御可能な GUI 部品（controlled parts）」を組み合わせて作られている．制御可能な GUI 部品は，ユーザがそれを操作したとき，その操作に対応したコマンドを生成する．また，この部品は，与えられたコマンドを解釈実行することにより，そのコマンドに対応した操作を再現する（**図 7**）．

1 つのアプリケーションは，様々な制御可能な GUI 部品が階層的に組み立てられて実現されている．したがって，1 つのコマンドは，イベントが発生した GUI 部品が一意に示されるよう，住所と同じように，階層的に構成されている．たとえば，お絵かきソフトの描画画素の選択メニューで，1 番目の画素に対応するボタンをクリックしたとき，

drawing dfig.btn.click(1)

のコマンドが生成される．ここで drawing は描画

ソフトを表し，dfig は画素選択メニューを表し，btn.click(1) は，このメニューの 1 番目のボタンがクリックされたことを表す．

2.2.4 イベントレコーダ/プレイヤ

イベントレコーダ/プレイヤは教師ノードシステムなどの他のノードシステムから受け取ったコマンドの列や，そのノードシステム自身が生成したコマンドの列を，時刻とともに記録する．時刻を記録するために，イベントレコーダ/プレイヤはタイマを持っている．このタイマの時刻は，グループ内の他のノードシステムと同期がとられている．

イベントレコーダが記録されたコマンド列をそれが記録された時刻に従ってコマンドトランシーバに与えることにより，記録された操作を再生することができる．イベントレコーダ/プレイヤは，各アプリケーションにおける操作の遅延時間やファイルの読み書きの時間などを記録する機能も持っている．このメッセージは CSV 形式で記述されており，表計算ソフトウェアなどを使うことによってこれを解析することができる．

図 8 に，イベントレコーダ/プレイヤと，それに記録されたコマンドの列の例を示す．コマンドの列の各行の左端の数字が，記録開始から，そのコマンドが記録されたときまでの経過時刻をミリセカンド単位で示している．その左の “m-” は，このコマンドが，自分自身のノードシステムのアプリケーションが生成したものであることを示す．他のノードシステムのアプリケーションが生成したものである場合は，“o-” となる．この右に続く文字列が，コマンドである．

2.2.5 コマンドトランシーバ

コマンドトランシーバは，他のノード間でコマンドの送受信を行う．コマンドトランシーバは，受け取ったコマンドの解釈実行も行う．このとき，コマンドがどの制御可能な GUI 部品の操作に関するものか，解析され，対象となる制御可能な GUI 部品に操作指示が行われ，その GUI 部品が操作を再現する．

イベントレコーダ/プレイヤが記録しているコマンド列を再生してコマンドトランシーバに与えられることにより，そのノードシステムで記録された操作を再生することもできる．

2.2.6 ノードシステムの操作状態

教師ノードシステムは，学生ノードシステムの操作状態を制御するアプリケーションを備えている．操作状態には,「独立操作状態」や「コマンド受信状態」などがある．

独立操作状態はノードシステムを他と独立して操作するための状態である．グループ内のすべてのノードシステムの操作状態を独立操作状態にすることによって，各ノードシステムの利用者による個別操作が可能となる．

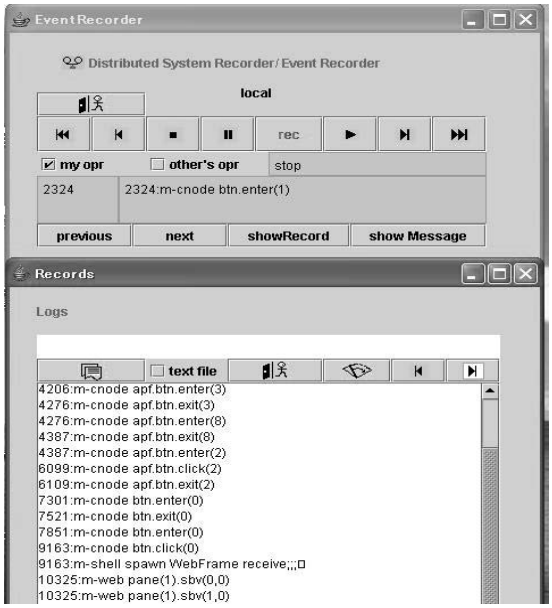

図 8 イベントレコーダ/プレイヤと記録されたコマンドの列
Fig. 8 Event recorder/player and the recorded commands.

コマンド受信状態はすべてのノードシステムで同じ操作を同時に行うための状態である．学生ノードシステムの操作状態を，コマンド受信状態にすることによって，教師ノードシステムの指示により，グループ内の全ノードシステムにいっせいに同じ操作を行わせることができる．

3. 評価実験

DSR は，異なる分散システムに，それぞれ同じ操作を行わせることができる．DSR は多数の端末コンピュータで同じ操作を同時に行わせることもできる．DSR のこれらの機能を確認するため，以下のような実験を行った．この実験は，DSR を使って 6 種類の分散システム上で，それぞれ同じ操作をさせ，ファイルの読み書きを行ったときの時間を計測し，分散システムの性能を比較するものである．この実験は以下の手順で行った．

(1) 1 台のコンピュータで，DSR を使って操作を記録する．この操作は，端末コンピュータがファイルサーバのファイルを読み書きするものである．
(2) 記録された操作を，6 種類の分散システムで再生する．このうちの 4 つでは 2 台以上の端末コンピュータ上で，同時に操作が行われる．この

とき，ファイルの読み書きに要した時間を計測する．

ここで使用した操作記録を，以下で示す．この操作は，以下の順序で行われる．

(1) 7 KB の GIF 画像を 1 つのファイルを読み込み，DSR のアプリケーションであるお絵かきソフト上で表示する．
(2) 16 KB の JPG 画像を 1 つのファイルから読み込み，お絵かきソフト上に表示する．
(3) 37 KB の JPG 画像を 1 つのファイルから読み込み，お絵かきソフト上に表示する．
(4) 絵を表す 3 KB のテキストを 1 つのファイルから読み込み，お絵かきソフト上に表示する．
(5) この 3 KB のテキストを，各端末コンピュータを利用しているユーザの個別ファイルに書き込む．
(6) この 3 KB のテキストを，各端末コンピュータを利用しているユーザの個別ファイルからそれぞれ読み込み，お絵かきソフト上に表示する．
(7) 絵を表す 10 KB のテキストを 1 つのファイルから読み込み，お絵かきソフト上に表示する．
(8) この 10 KB のテキストを，各端末コンピュータを利用しているユーザの個別ファイルに書き込む．
(9) この 10 KB のテキストを，各端末コンピュータを利用しているユーザの個別ファイルからそれぞれ読み込み，お絵かきソフト上に表示する．

ファイルの読み書きの間，マウスカーソルを動かして，マウスカーソルの反応についても計測した．

以上の操作は約 6 分間連続して行われる．採取された操作記録の大きさは約 120 K バイトであった．

この操作において画像ファイルの読み込みは，講義や演習時に教師が示した，ファイルサーバの一カ所に存在するデータを，受講者がいっせいに端末に読み込んで表示するような場合を想定している．

テキストデータの読み書きは，教師が用意した一カ所に存在するデータを受講者がいっせいに端末に読み込み，それぞれの受講者がこれを編集した後，それぞれの個別保存場所に書き込み，それぞれの個別保存場所から，いっせいにもう一度読み込むような場合を想定している．

6 種類の分散システムとして，以下を使った．

- `Windows-1`：1 台の Windows ノートパソコンである．このパソコン 1 台で，端末コンピュータと，ファイルサーバの役割を担当する．
- `Windows-2`：1 台の Windows ディスクトップコンピュータと 1 台の Windows ノートパソコンをネットワークで接続したシステムである．このディスクトップパソコンは端末コンピュータとファイルサーバの役割を担当する．ノートパソコンは端末コンピュータの役割を担当する．この 2 台のパソコンは，100 Mbps の全 2 重のスイッチによって結合されている．
- `Linux-1`：1 台の Linux thin client コンピュータ（1 ノード）と 1 台のファイルサーバ用コンピュータをネットワークで接続したシステムである．Linux thin client コンピュータは，ハードディスクを持たない端末コンピュータで，ある．OS は Linux でカーネルのバージョンは 2.2.14（Turbolinux 4.5）である．これらは，複数のスイッチングハブで構成されたネットワークに接続されている．端末コンピュータは，100 Mbps 全 2 重でこのネットワークに接続されており，ファイルサーバ用コンピュータは，2 本の 1 Gpbs 全 2 重で接続されている．
- `Linux-2`：2 台の Linux thin client コンピュータ（2 ノード）と 1 台のファイルサーバ用コンピュータをネットワークで接続したシステムである．Linux-1 と同じようにコンピュータとネットワークが接続されている．
- `Linux-10`：10 台の Linux thin client コンピュータ（10 ノード）と 1 台のファイルサーバ用コンピュータをネットワークで接続したシステムである．Linux-1 と同じようにコンピュータとネットワークが接続されている．
- `Linux-75`：75 台の Linux thin client コンピュータ（75 ノード）と 1 台のファイルサーバ用コンピュータをネットワークで接続したシステムである．

Windows-1 と Linux-1 は教師の準備や個々の学生の予習や復習など，Windows-2 と Linux-2 は教師と学生が 1 対 1 で行う面接授業など，Linux-10 は小規模のグループ学習など，Linux-75 は大規模のいっせい授業などで分散システムが利用される場合を想定している．

Linux-1 から Linux-75 までは，2000 年に導入された九州工業大学情報科学センター教育用計算機システム[7),9)] を利用し，一般ユーザがいない時間に実験を行った．

ここで，Windows-1 と Linux-1 以外では，すべての端末コンピュータで，いっせいに同じ操作を行わせた．このとき，教師ノードシステムから出される操作の指示に対し，0.3 秒以内ですべての端末コンピュー

表 1 本システムを使った分散システム性能評価実験結果
Table 1 The result of a benchmark test for distributed system using this system.

読み書きしたデータ		Windows-1	Windows-2	Linux-1	Linux-2	Linux-10	Linux-75
GIF 画像 7 KB	最大	4.1	6.2	0.0	1.0	5.3	5.9
読み込み（秒）	最小	4.1	0.2	0.0	0.3	0.0	0.0
	平均	4.1	3.3	0.0	0.7	3.7	5.1
JPEG 画像 16 KB	最大	11.6	10.2	11.8	11.4	9.1	9.5
読み込み（秒）	最小	11.6	0.7	11.8	9.2	0.7	0.1
	平均	11.6	5.5	11.8	10.3	8.1	8.0
JPEG 画像 37 KB	最大	14.6	1.1	0.1	16.1	21.0	20.5
読み込み（秒）	最小	14.6	1.2	0.1	0.0	0.0	0.0
	平均	14.6	1.2	0.1	8.1	14.3	13.5
10 KB 共通ファイル	最大	0.6	0.2	0.0	0.0	0.2	0.3
読み込み（秒）	最小	0.6	0.0	0.0	0.0	0.0	0.0
	平均	0.6	0.1	0.0	0.0	0.0	0.1
10 KB 個別ファイル	最大	1.7	1.4	2.3	1.6	2.4	2.0
書き込み（秒）	最小	1.7	0.8	2.3	1.4	1.1	0.9
	平均	1.7	1.1	2.3	1.5	1.4	1.4
10 KB 個別ファイル	最大	0.2	0.2	0.2	0.0	0.2	0.3
読み込み（秒）	最小	0.2	0.0	0.2	0.0	0.0	0.0
	平均	0.2	0.1	0.2	0.0	0.1	0.1
0.1 秒以上遅延があったマウス移動処理（平均回数/端末）		0	1.5	0.0	0.0	1.3	0.9

読み書きの時間が 0.0 となっているものは，読み書きの時間が 0.1 秒未満でであったことを表す．

図 9 75 台の Linux Think Client を使った同時操作実行
Fig. 9 Simultaneous operation using 75 Linux thin client.

タで指示された操作が開始された．

Linux-75 の環境で実験を行っている様子を **図 9** に示す．各分散システムが使用しているコンピュータの詳細を **図 10** に示す．この実験の結果を **表 1** に示す．

この実験が示すように，DSR を使うことによって，まったく同じ操作を，端末コンピュータの OS が異なる分散システム上で行わせることができた．同じ Windows や Linux でも，その種類やバージョン，様々な設定値の調整などによって実験結果は大きく変わる可能性があるが，DSR はこれらの影響を調べることも可能である．ただし，この場合，Java のバージョンについては，同じものを用いる必要がある．

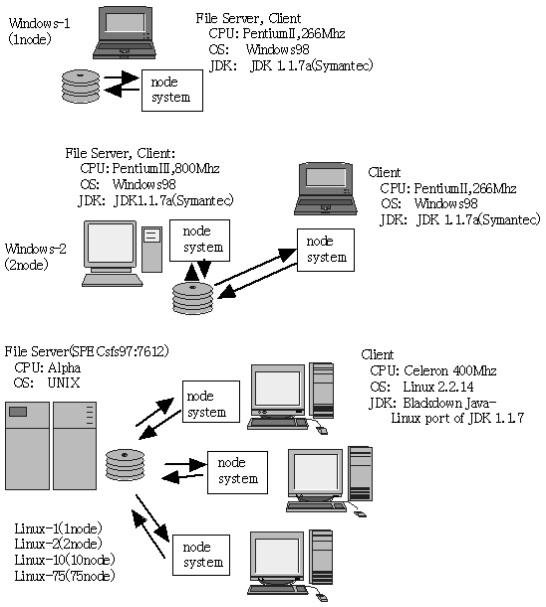

図 10 実験に使用した機器の詳細
Fig. 10 Specifications of the computers of the experiments.

Windows-1 の結果は，Windows-2 より悪かった．このことによって，ファイルの読み書きの性能は，主にハードディスクやファイルサーバの性能に依存していることが推定できる．この推定を確認するためには Windows-1 と Windows-2 で利用したファイルサーバそれぞれについて多数のクライアントを使って今回

の実験と同じ操作を行い，Linux で利用したファイルサーバの場合と比較する必要があるが，今回は，その環境を準備できなかった．これについては今後の課題とする．

Linux-75 の場合のファイル読み書きの時間は，Linux-1，Linux-2，Linux-10 のときと比べて，負荷が大きくなり，長くなるだろうと予想していたが，実験の結果は，あまり変わらなかった．この原因として，この実験は，Linux-1，Linux-2，Linux-10，Linux-75 の順番に行ったため，ファイルがファイルサーバのメモリにキャッシュされ，ハードディスクへのアクセス時間を減少させていることなどが考えられる．

本来，使用する端末コンピュータの数を変えて実験するたびにファイルサーバの再起動を行い，キャッシュの初期化を行うべきであるが，運用を行っているファイルサーバを何度も再起動することは障害発生の危険がともなうため，今回は行っていない．この問題については，内容がまったく同じで名前が異なるファイルを使用することなどによって，解決できる可能性がある．

マウス移動コマンドを受信してからその処理が終了するまで 0.1 秒以上遅延が発生した回数を数えることによって動作のなめらかさを計測をしようとしたが，今回の約 6 分間の操作での遅延はほとんど発生しなかった．

最近の教育現場では，Word などのワープロソフトや，Excel などの表計算ソフトもよく利用されているが，本システムはまだこれらのアプリケーションを備えておらず，本来はこれらのアプリケーションも必要である．DSR は，テキストエディタを備えているので，これを使って文章を編集する場合と，ワープロソフトを使って文章を編集した場合の応答時間などの関係が分かれば，異なる環境でワープロソフトを使った場合の結果を予測できる可能性がある．表計算についても，本システムが備えているプログラミング環境で計算を行った場合との関係を調べることによって，同様の予測ができる可能性がある．しかしながら，これらの予測を行うためには，より多くの比較データを収集する必要があり，これについても今後の課題とする．

DSR で分散システムをいっせい操作させてその性能の計測を行うとき，各ノードのアプリケーション操作を行うデータと，そのアプリケーションがネットワークを通じて入出力するデータが，同時に同じネットワークに流れてしまう．今回の実験の場合，アプリケーション操作のデータ量は 6 分間の操作時間で平均して，1 秒あたり約 330 バイトであった．操作データの大部分はマウスの移動データであり，これは実験中，ほぼずっと行われていたため偏りはあまりない．たとえ 1 秒間にこの 3 倍の操作データが流れたとしても，1 秒あたり約 1 K バイトである．このとき，2 分木状に接続されたノードにこのデータが流れるため，1 つのノードのネットワークインタフェースには，入力として 1 秒あたり約 1 K バイト，出力として 1 秒あたり約 2 K バイトの操作データが流れることになる．全 2 重のスイッチで通信が行われているため，1 つのネットワークインタフェースのデータ量の多い出力側で，操作に使用される 1 秒あたりのデータ量が約 2 K バイト（約 16 Kbps）であり，ネットワークの通信容量（100 Mbps）と比較すると，いっせい操作に必要な単位時間あたりのデータ量は無視できる量である．

4. 関 連 研 究

LoadRunner [5] は利用者よって発生する負荷を分散システムに与えることによって，分散システムの性能評価を行う，という意味で，DSR と類似している．LoadRunner は少ないハードウェアを使って，何千もの仮想ユーザの平行動作を模倣することができる．一方，DSR は性能評価を行うとき，現実のハードウェアを必要とするが，現実のユーザが行った操作を，実際に再現することによって，分散システムの評価を行う．DSR を使って分散システムがテストされているとき，仮想的なユーザのアプリケーション操作が，実際の分散システムの端末で再現される．また，DSR は操作時のアプリケーションの応答速度を計測することにより，端末コンピュータの性能も評価することができる．

DBS（Distributed Benchmark System）[6] は，多くの場所で分散システム（ネットワーク）の性能を計測できる，という意味で，DSR と類似している．DBS はすべての TCP 機能の性能を計測できるなど，DSR が持っていない機能を持っている．しかしながら，DBS はネットワークの性能を計測するツールであるのに対して，DSR は分散システムの上で利用者が実際にアプリケーションを使ったときの性能を計測するものである．

DSR はノードシステム間を TCP で 2 分木状に接続することによって，同一コマンドを短時間で信頼性を持ってグループ内のすべてのノードシステムに伝え，ノードシステムのいっせい操作を実現している．データのいっせい配信を行うには IP マルチキャストがよく使われるが，IP マルチキャストは信頼性がない．実際に本システムのノード間通信に IP マルチキャ

ストを使った場合の実装を行ってみたが，コマンドがすべてのノードシステムに届かない場合があるため，いっせい操作を安定して行うことは困難であった．IPマルチキャストを使った信頼性のある通信方式としてSRM[2]などもあるが，SRMでは送信されたデータが，その順番どおりに受信されることを保障していないため，コマンドが送信された順番に受信側で実行される必要のあるいっせい操作には使用できない．

5. おわりに

プラットフォーム独立な教育用分散システムの性能評価ツールDSRの構成と，それを利用した実験について述べた．

DSRを使って，あらかじめ記録したユーザの操作を他の分散システムで再生し，その分散システムの性能を計測することができた．また，分散システムの多くの端末コンピュータ上で，同じことを同時に実行させ，ファイルサーバの性能計測を行うことができた．異なる種類の分散システムの比較も行うことができた．

DSRの最も大きな欠点の1つは，DSRが備えていないアプリケーションを使った場合の，分散システムの性能計測を行えないことである．しかしながら，この欠点がDSRをプラットフォーム独立にしている．DSRは表計算ソフトや，電子メールクライアントソフトを現在備えていないが，これらは，日常的に教育現場や実生活でとてもよく使われるものなので，今後，これらをアプリケーションに加えることを計画している．DSRのアプリケーションは，実際に教育現場で使われるアプリケーションと異なるため，DSRを使って得られた結果は，現時点ではあくまでも参考値としてしか利用できない．今後実際に教育現場で使用されるアプリケーションを使った場合との比較を行い，DSRの精度を調べる必要がある．

DSRはJavaを使って作成しているため，Javaが利用できない分散システムの性能は計測できない．また，Javaが動作しても，プラットフォームによって細かい部分で動作が異なる場合がある．この違いについては，DSRのプログラムによってある程度吸収しているが，まだ不完全な部分もある．JavaはCやC++を使って開発されたシステムと比べて一般的には実行速度が遅い欠点があるが，これらの欠点については，少しずつ改善されている．

DSRは，仮想ユーザとして，単純なコマンドの列しか生成できない．WebのCGI等の相互作用を持つソフトウェアと会話するためには，もっと知的な仮想ユーザを生成する必要がある．

DSRは，大量の計測データを出力する．このデータ整理の方法も検討する必要がある．

今後，これらの改良や，デバッグを行いながら，多くの分散システムの性能を計測し，分散システムの総合的な評価方法の研究を進めていく予定である．

DSRはソースコードとともに，http://yama-linux.cc.kagoshima-u.ac.jp/~dsr/で公開している．

謝辞 本研究の実施にあたりお世話になりました，九州工業大学情報科学センターのスタッフと学生の皆さまに感謝します．本研究の一部は，文部省科学研究費（C）（2）（09680401）の補助を受けました．

参考文献

1) Brown, D.G., Burg, J.J. and Dominick, J.L.: A strategic plan for ubiquitous laptop computing, *Comm. ACM*, Vol.41, No.1, pp.25–35 (1998).
2) Floyd, S., Jacobson, V., Liu, C., McCanne, S. and L., Z.: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, *IEEE/ACM Trans. Networking*, Vol.5, No.6, pp.784–803 (1997).
3) Hirahara, T., Yamanoue, T., Anzai, H. and Arita, I.: Sending an Image to a large number of nodes in short time using TCP, *IEEE International Conference on Multimedia and Expo*, IEEE (2000).
4) 平原貴行，山之上卓，安在弘幸，有田五次郎：TCPを利用した分散ネットワーク環境のための電子黒板システム，情報処理学会論文誌，Vol.43, No.1, pp.176–184 (2002).
5) LoadRunner. http://www.mercury.com/us/products/performance-center/loadrunner/
6) Maruyama, Y.: DBS: a powerful tool for TCP performance evaluations, *Proc. SPIE, Performance and Control of Network Systems*, Vol.3231 (1997).
7) 中山 仁，大西淑雅，望月雅光，山之上卓，甲斐郷子：Linux thin clientを端末とする集合教育用計算機環境の構築，情報処理学会研究報告2000-DSM-18, pp.31–36, 情報処理学会 (2000).
8) 二宮公紀：新コンピュータシステム紹介，広報，Vol.13, pp.3–18, 鹿児島大学総合情報処理センター (2000).
9) 大西淑雅，中山 仁，甲斐郷子：ライフサイクルモデルに基づく教育用計算機システムの構築と運用，情報処理学会論文誌，Vol.45, No.1, pp.33–45 (2004).
10) 斉藤明紀，中西通雄：教育用計算機環境に対する要求と課題，情報処理，Vol.45, No.3, pp.227–232 (2004).
11) Yamanoue, T., Minami, T., Ruxton, I. and

Sakurai, W.: Learning Usage of English KWICly with WebLEAP/DSR, *Proc. 2nd International Conference on Information Technology and Applications* (*ICITA-2004*), 14-6, Harbin, China (2004).

(平成 16 年 7 月 8 日受付)
(平成 17 年 2 月 1 日採録)

山之上 卓（正会員）

昭和 34 年生．昭和 55 年呉工業高等専門学校卒業．昭和 59 年九州工業大学大学院工学研究科情報工学専攻修士課程修了．昭和 62 年九州大学大学院総合理工学研究科情報システム学専攻博士後期課程単位取得退学．平成 5 年より九州工業大学情報科学センター助教授．平成 15 年より鹿児島大学学術情報基盤センター教授．P2P，教育支援システム，分散システムの評価システムなどの研究に従事．博士（工学）．電子情報処理学会，ソフトウェア科学会，IEEE-CS，ACM 各会員