2W－2

ネットワークトランザクションの把握のための必要データの自動抽出

田中貴子　三浦孝夫
産能大学経営情報学部

1　はじめに

情報処理環境におけるシステムの高度化や通信ネットワークの普及は著しいものがある。それに伴い、システムにおける処理すべき情報量やより高度かつ複雑な処理機能の要求が増大し、コンピュータシステムの負荷量も増大する。現状システムの性能と稼動状況の監視は、システムの効率性の維持という観点からより避けられないものである。

そのために、システムの性能を定量的に把握する必要があり、システムのモニタリングが行われる。システムの時系列情報を獲得するために、一般的に、ソフトウェアモニタが用いられる。測定対象のシステム上で、データ収集用プログラムを実行させて測定し、システム全体から見たハードウェア資源及び、ソフトウェア資源の測定が可能である。これは、中型以上のコンピュータであれば、標準のものが装備されている。

システムの性能と稼動状況におけるクラスは、以下の3つのクラスに分類できる。CPUやディスク等の物理的資源やスケジューラを含むシステムのハードウェア及び、ソフトウェアの明確な特徴や、サイズ等のデータベースの特徴、実行中のトランザクション数の制御のローカルコントロールの構造や同時実行制御アルゴリズムを追跡する**データベース・システム・モデル** (databese system model)、ユーザのプロセスを追跡する**ユーザ・モデル** (user model)、参照動作やワークロードにおけるトランザクションのプロセス要求を追跡する**トランザクション・モデル** (transaction model) がある。

本稿では、**ユーザ・モデル**と**トランザクション・モデル**の観点から、測定オーバヘッドの危険性のあるソフトウェアモニタを利用せず、システムの稼動状況を測定し、トランザクションに関する情報を収集できるような手法を考察する。ユーザがシステムの稼動状況の測定を意識せずに、トランザクション要求をした際に、同時に、測定値の収集を可能とするようなトランザクション機能を持った言語 Oberon-0/T を提案する。

2　利用環境

本稿で、Oberon-0/T の実現に利用する環境について概要を述べる。

2.1　Oberon-0 の概要

Oberon-0 は、高水準言語の1つであり、以下のような多くの環境で稼動する言語である。: Ceres、Macintosh、DECStation running ULTRIX、IBM PC、MS-DOS、IBM RISC/System 6000、Sun SPARK machine、HPPA-RISK machine. 計算機の構造を反映しない言語であるから、日常の表現に近い表現が可能となる。従って、ユーザ側がより計算機に近い表現が用意となる。また、代入、複合文、条件・繰り返し的再帰、if-文・while-文、手続きのようなサブルーチン、配列、レコード等といった、Modula-2 や Pascal の主要な特徴を受け継いだものであるが、若干簡略化されている。例えば、データ・タイプとして INTEGER や BOOLEAN のみに限定されている。

しかし、特徴として、拡張性の高さが挙げられる。オペレーティングシステムによって提供された関数とユーザが定義した関数とを区別しない。従って、JAVA や C++ のように自由にライブラリを組み込める。また、仮想機械の方式を採用しており、数多くのプラットホームでの動作が可能である。

また、他の言語に表記が似ているため、新たな言語の文法を覚える必要がなく、移植性の高さと共に、本稿でこの言語を採用した点である。

2.2　Shore の概要

Shore(Scalable Heterogeneous Object REpositry) は、データベースの特徴によって拡張されたファイルシステム、またはファイルシステムの特徴を持った DBMS といえる。トランザクション機能や回復、同時実行制御をサポートしている。複数のデータサーバが集積したものであり、Object Idendifier(OID) によって、それぞれのオブジェクトへのアクセスを許可する。また、オブジェクト自体に型宣言がされている。Unix に似た名前のスペースによって、名前の付けられたオブジェクトは、ディレクトリ、シンボルリンク、または個々のオブジェクトに置くことができる。これによって、データアクセスが Unix のようなデータアクセスが可能となっている。

言語の拘束のあるどんなプログラミング言語においてもアプリケーションをサポートしている。対称性があり、シングルサーバのオペレーションをサポートする配置のシステム構造を持つので、すべての関与するプロセッサは、Shore サーバプロセスがどこのローカルディスクにあるかに関わらず、それを動かすことが可能である。

3　必要データの自動抽出

3.1　トランザクションの把握のための必要データ

ネットワークトランザクションの把握のための必要データを自動抽出する前に、どのようなデータを抽出する必要があるかということを洗い出す。シミュレーションによって、トランザクションを設計し、シミュレーションデータを得る。シミュレーションには、GPSS や SIMSCRIPT のようなシミュレーション言語によるものと、C や FORATRAN や PL/I 等の汎用プログラミング言語によるものがある。本稿では、トランザクションの把握のための必要データを自動抽出することを目的としているため、特別な便宜を用意され、機能の細い部分が自動的に行われる前者の手法を用いる。

ここで得たデータには、数多くのパラメータがあり、それらがどのようにネットワークトランザクションの把握に対応するのかということを、データベーストランザクションと比較し、関連付けを行う。

待ち行列計算が可能であり、システムの状態が時間的に不連続なシミュレーションに向きであるという特徴を持った GPSS を用いて、Read/Write 問題についてのトランザクションのシミュレーションを行う。シミュレーションデータの **STORAGE**、**FACILITY**、**QUEUE** というデータベース

Extracting Statistical Information of Network Transactions
Takako TANAKA and Takao MIURA
Dept.of Management and Informatics, SANNO College
Kamikasuya 1573, Isehara, Kanagawa 259-11 JAPAN

トランザクションにおけるRead、Write、待ち状態に当たる項目のパラメータの中から、データベーストランザクションとの関連付けを行い、性能に関わるパラメータを洗い出した。ReadとWriteによって、その処理の優先度によって、競合が生じる。それに伴い、排他的論理和である施錠互換性を用いて、トランザクションを制御する。そのため、実行待ちのトランザクションが生じる。この実行待ちのトランザクションを計算するのに、新たに定義する必要がない。従って、その特徴から、GPSSが最適なのである。データベーストランザクションとの関連づけとは、トランザクションの並列度、スループット、トランザクションの稼働数、Write処理の実行状況、トランザクションの待ち時間と待ち状態の数、トランザクションの状態等のトランザクションにおける大きな項目と、対象項目中の詳細なパラメータがどのように関連するかということである。トランザクションの種別に関わりなく、共通するデータ操作は、トランザクションの開始と終了、データ操作としての書き込みと読み出しである。従って、それらをモニタ対象として、それに関するデータを収集すればよい。

以下の表は、トランザクションと対象項目中のパラメータとの関連を示したものである。以下の表は、トランザクションとGPSSでのパラメータとの関連を示したものである。

トランザクション	パラメータ
トランザクションの並列度	*CAPACITY*
スループット	平均使用量
トランザクションの稼働数	ストレージを使用中の数
	CONTENTS
Write 処理の実行状況	*SEIZE*の番号
トランザクションの待ち時間	平均待ち時間
待ち状態の数	待ち行列に入った*UNIT*数
	待たずに通過した*UNIT*数
トランザクションの状態	*STATUS*

CAPACIYとは、シミュレーションを設定する際に、前もって設定しておくパラメータである。また、CONTENTSは、現在Writeをしているトランザクションの番号を表しているものである。STATUSは、トランザクションの状態を表しているパラメータである。

一般的に、抽象性の高いデータのような直観的理解が困難であるようなデータをを直接モニタすることは、手間がかかる。例えば、システムのパフォーマンス・バグやホットスポット、ボトルネックに関するデータ等である。そのような抽象性の高いものを含めたデータをより手間をかけずに、モニタすることは、システムの性能評価において、大変有効なことである。従って、これを実現するような方法を提案する。

3.2 必要データの自動抽出

ソフトウェアモニタを使用せずに、システムの稼動状況を測定するために、通信ライブラリの拡張や、プログラムの拡張がまず考えられる。しかし、これらは、処理の効率性やユーザがシステムの測定を強く意識する必要があり、本稿の目的を満たさない。従って、ユーザがネットワークトランザクションを意識しなくても、トランザクションを行えるように、仮想機械上での拡張を行う。

そのために、仮想機械上での、共通コードとしての機械語を設定し、仮想機械自身がネットワーク上のユーザとなり、ネットワーク上でのトランザクションをを行うように拡張する。従って、ユーザは、単に、ローカルな範囲でのトランザクションを行っているように感じることができる。

拡張によって計測すべき値は、ユーザレベルでのトランザクションではなく、ネットワーク上でのトランザクションのユーザとなる仮想機械レベルでのトランザクションの部分で行う。

4 Oberon-0/T

仮想機械上での共通コードの設定をし、必要データの自動抽出を実現する。トランザクション機能をサポートしていないOberon-0に、その機能をサポートするよう拡張し、トランザクションの把握に関する情報を自動抽出するように拡張した言語Oberon-0/Tを提案する。

トランザクションの命令として、**read**、**write**、**begin**、**commit**、**abort**というRead処理、Write処理、処理開始、正常終了、異常終了が提供される。前節の必要データの洗い出しによる性能の量的尺度を実測において収集するために、それらの処理を以下のような命令としてOberon-0の中に組み込み、仮想機械上で、Shoreを呼び出す。

READ_OBJECT()
WRITE_OBJECT()
BEGIN_TRANSACTION
COMMMIT_TRANSACTION

READ_OBJECT()、**WRITE_OBJECT()**は、ネットワーク上のデータの読み出しと書き込みの処理を行うための命令、**BEGIN_TRANSACTION**は、トランザクションの開始をネットワーク上で宣言する命令として提供される。**COMMMIT_TRANSACTION**は、ネットワーク上でトランザクションの正常終了を宣言するものとして提供される。トランザクションの状態には、異常終了（Abort）もあるが、これは、処理上に起きる異常終了なので、ユーザがアボートを決めることはできないので、組み込まない。これらの命令をユーザがプログラムの中で宣言すれば、Oberon-0/Tが、自動的にネットワークトランザクション・データを集めてくるのである。表1は、Oberon-0/TとShoreの関数との対応を示したものである。

Oberon－0/T	*Shore*
BEGIN_TRANSACTION	*SH_BEGIN_TRANSACTION*
COMMIT_TRANSACTION	*SH_COMMIT_TRANSACTION*
READ_OBJECT()	*scan_file*
WRITE_OBJECT()	*append_file*

5 結び

トランザクションにおいて、データ処理として、データの読み出し、書き込みなどの他に、複雑な検索や更新等も行われる。しかし、それらは、ReadとWriteという2つの処理の組み合わせで、表現することが可能である。従って、本稿では、データ処理の基本動作として、Oberon-0/Tにそれらをトランザクション機能として持たせたことを示した。

これにより、測定対象のシステム上でのデータ収集プログラムを実行せずに、トランザクションの要求から直接データの収集が可能となり、シミュレーションによってあらかじめ収集したデータと、Oberon-0/Tで得た値を実行中に比較し、状況を監視することが可能である。

参考文献

[1] B.W. カーニハン, D.M. リッチ: プログラミング言語C (石田晴久訳), 共立出版

[2] H.Hellerman, T.F.Conroy: オペレーティングシステムの性能評価 (大野豊訳), 日本コンピュータ協会

[3] Niklaus Wirth.: Compiler Construction, Addison-wesley Publishing Conpany

[4] Rakesh Agrawal, Michael J. Carey and Miron Livny: Concurrency Control Performance Modeling:Alternatives and Implications, ACM Transactions on Database Systems,Vol.12,No4,pp.60 654,1987