UDC 681.3

分散システムの性能評価

Performance Evaluation for Distributed Systems

大城 卓*(1) 中村賢亮*(1) 畠山康博*(2)
Takashi OSHIRO Masaaki NAKAMURA Yasuhiro HATAKEYAMA

抄 録

ビジネスの変化に柔軟に対応できる分散システムの構築ニーズが高まっている。しかし多くの事例からも明らかなようにその構築は容易でなく，特に端末の応答時間がかかるとの性能問題も多い。性能問題は開発の最終段階で発覚することが多く，その対応にはコストがかかる，あるいは開発自体のやり直しにつながることもある。性能評価技術は，システムのライフサイクル全体に渡る性能上の問題の事前回避，及び発生した問題解決を目的とする，システムの構築を請け負うSIベンダにとり，システム開発のリスク管理として極めて重要である。分散システムの性能問題に対する新日本製鐵 エレクトロニクス・情報通信事業部システム研究開発センターの取り組みについて紹介した。

Abstract

There is an increasing need for constructing distributed systems which can freely cope with a changing of business conditions. However, as many cases show, it is evident that the construction of the system is not only easy but there are many performance problems especially such as an increasing response time at terminals. There are frequent occasions that the performance problems are revealed at the last stage of the system development, and consequently the cost to solve the problems is increased, or redoing of the development itself may be involved in. The technology of performance evaluation has the objects to avoid in advance such performance problems as extending through the life cycle of the system and to solve the problems occurred. The performance evaluation technology is very important to the SI vendors who contract for the system integrating work, since it plays the role of the risk management for developing the system. In this paper, introduced is the buckling down of the Systems Research And Development Center of Nippon Steel to the performance problems of distributed systems.

1. 緒 言

クライアント／サーバシステムに代表される分散システムは，(1)製品選択の幅が広く最先端技術を低価格で利用できる，(2)拡張性に優れ，最適なシステム構成が組める，等の利点がある。しかし，多くの事例が示すように，その構築は必ずしも容易ではない。これらの失敗事例の調査によると，端末からの応答時間が遅い，バッチジョブが規定時間内に終了しないといった性能問題が非常に多いことが分かる。性能問題は，通常，結合テスト等の開発後期に発覚し，その時点での対処が困難な場合も多く，対応に膨大なコストがかかり，最悪の場合，開発自体のやり直しにつながることもある。

分散システムの性能問題の対処が困難な理由としては以下が挙げられる。(1)オープン化及びマルチベンダ化で，標準的なハードウェア，OS，データベース，ネットワーク等を自由に選択しシステムを構築できるようになったが，その組み合わせは無数に存在し，組み合わせ後の性能の評価は非常に困難である。(2)ソフトウェア開発の多くの見積りと同様，性能に関しても定量的な手法がほとんど用いられておらず，結局は勘と経験の"出たとこ勝負"で行われている。また，詳細な情報を集め，管理することができたシングルベンダによる開発と異なり，必要な情報を集めることも困難で，従来のメインフレームの手法もあまり役には立たないようである。

このような状況の中，多くのSIベンダがこの問題に対応するべく，自社のソフトウェアの開発手法の一つとして分散システムの性能評価技術の確立に努力をしている。

本稿では，分散システムの性能問題に対する新日本製鐵エレクトロニクス・情報通信事業部システム研究開発センター(以下，当センターという)の取り組みについて紹介する。

*(1) エレクトロニクス・情報通信事業部
システム研究開発センター 主任研究員
神奈川県相模原市淵野辺5-10-1 〒229-8551 ☎(0427)68-6080

*(2) エレクトロニクス・情報通信事業部
システム研究開発センター 主任研究員 工博

2. 分散システムの性能評価

性能評価は，システムの開発から運用に至る性能上の問題を事前に回避する，及び発生した問題を特定し解決することを目的としている。システム構築を請け負うSIベンダにとり，それは，開発のリスク管理の一つであり，顧客ニーズに適合した(ライトサイジング)システムの提案そして構築という意味でも重要である。システムの性能を契約要件として定義するSLA(Service Level Agreement)[*1]を導入する事例も増えており，性能評価の重要性は増している。

性能の議論には，性能を表現する尺度(性能指標performance metrics)が重要であり，応答性(Responsiveness)と利用率(Utilization)，そしてスループット(Throughput)が良く用いられている。応答性は，入力終了から応答の返り始めまでの時間などの応答時間である。ユーザにとり重要な性能指標であり，ユーザの業務効率に直接影響する。利用率は，CPU，ディスク，ネットワークなどの計算機資源の稼動率である。システムの運用管理部門にとり，コスト管理の点からも非常に重要な尺度である。スループットは，単位時間に処理可能な量であり，例えば，ジョブスループット(単位時間当たりのジョブの処理回数)などは代表的な量である。

これらの指標は互いに関連しており，ユーザの目的，費用等を勘案して最適な値の組み合わせが得られるように性能を作り込むことが性能評価の目的である。また，開発フェーズに応じ，性能評価に要求される内容も変化する。開発初期の段階では，最適なシステム構成の検討が必要とされたり，最終テストの段階では，性能上の問題点の検出やチューニングあるいは性能検証などが重要となる。開発フェーズに応じた定量的な手法による性能評価が必要である。

2.1 性能評価の概要

性能評価は，(1)性能評価モデルの作成，(2)性能評価手法の適用，(3)統計量の解析の大きく三つの段階に分けられる。

性能評価モデルは，評価目的や評価に必要なコスト等を勘案して対象システムをモデル化したものである。システムを構成する計算機やディスク，及びネットワーク等からなるハードウェアモデル，アプリケーションプログラムやミドルウェアの構成，及びデータベースの構造等からなるソフトウェアモデル，そして対象システムに与える負荷(ワークロード)からなる。また，性能指標の測定箇所や，評価目的に応じ変化させる量(サーバマシンの処理能力や，ネットワークの伝送能力等)も同時に決めなければならない。この値を変化させ，対象システムの性能特性を分析する。システムの性能を評価する上で，“もし，・・・であった場合(What-if)”は非常に重要で，種々の条件から多角的に評価することが必要とされる。

ワークロードは，ハードウェアモデル，ソフトウェアモデルに比べ軽視されがちであるが，評価結果に大きな影響を与える。ワークロードはシステムに与える負荷であるが，複数存在するワークロード各々の負荷の大きさだけでなく発生頻度も決めなければならない。各ワークロードの単体テスト時には所望の性能が得られたが，複数のワークロードの複合テストでは予定された性能が得られない場合も多く，ワークロード間の競合による待ちが原因である可能性も高い。ワークロードのモデル化が不完全であると，得られた結果が全く意味をなさないものとなってしまう。

評価手法については後述するが，各手法を適用して得られた結果は，平均値だけでなく分散，最大値，最小値も重要である。特に，分散が大きい場合は注意が必要で，例えば，CPUの利用率の平均値が低くても，分散が大きい場合には，負荷の高いプロセスが存在しており，ある時点で極端に性能が悪化している可能性もある。また性能保証という観点からは，いわゆる90パーセンタイル値等も重要な指標となる。

2.2 性能評価の手法

性能評価手法は，(1)性能予測，(2)性能実測，(3)性能監視の三つに分けて議論されることが多い。

性能予測は，解析的手法，シミュレーション手法等で，対象システムの性能を予測する方法であり，評価対象とするシステム本体(実行環境，プログラム)を必要としないことが特徴である。

待ち行列手法を代表とする解析的手法は，平均到着率と，平均サービス時間を中心に数学的な手法でシステムの平均待ち時間を求める方法である。比較的短時間で解析できるが，平均的な値しか求めることができない，また複雑な系を対象とした場合，解析が非常に困難になる欠点もある。

シミュレーション手法は，離散型のシミュレータ(シミュレーション言語)で対象システムをモデル化し，シミュレーションすることで性能指標を得る方法である。平均値だけでなく分布を求めることができ，また負荷を動的に変化させることもできるなど，柔軟な評価が可能である。しかし，モデル作成には高度な知識を必要とし，構築コストも無視できない。要求する精度に見合ったモデルの構築が重要である。

性能実測は，システムの全体あるいは一部分に所定のワークロードを掛け，応答時間，スループットなどの性能指標を測定する手法である。ワークロードの与え方で，単一ワークロード試験，複合ワークロード試験に分けられる。性能検証の場合など，単一ワークロード試験の結果を個々に最適化した後，複合ワークロード試験でシステム全体を検証するなど，両方が実施されることも多い。複合ワークロードの生成に，一台のマシンで複数のユーザを作り出すエミュレーションも良く用いられる。

SPEC[*2]ベンチマーク,TPC(Transaction Processing Performance Council)のベンチマークなどの標準ベンチマークテストは，性能比較のために，システムの利用目的に適した一連のテストを準備し，実測で性能を評価する方法である。多くのベンダがベンチマークの結果を公開しており，対象システムと類似のベンチマークの結果が得られる場合には便利である。また，計算機単体の性能評価指標の一つとして多用されている。

一方，SIベンダが対象とする企業向け情報システムは，データベース中心に構築され，性能面でもその影響は大きい。データベースの標準ベンチマークテストの値も公表されているが，大まかな目安にはなっても，それを基にデータベース性能がシステム仕様を満たしているか判断することは難しい。アプリケーションプログラム毎に異なるデータベースの構造，データの分布，ワークロードなどの特徴を反映させたアプリケーションスペシフィックベンチマーク(ASB)[*3]の実施が不可欠となっている。

---

*1 SLA(Service Level Agreement)：システムの利用方法や利用条件に対して保証するレスポンスタイム等のシステムのサービス品質。

*2 SPECは，The Standard Performance Evaluation Corporationの商標。

*3 アプリケーションスペシフィックベンチマーク(ASB)：開発システム(アプリケーション)の特徴を反映させた試験条件に基づいて行われるベンチマークのこと。各種団体がベンチマークの方法を決定して行っている標準ベンチマークとは異なり，得られる結果に汎用性はないが，より詳細な開発システムの性能特性を把握することを目的に行われる。

性能実測は，性能予測と異なり，プロトタイプにせよ，一定の規模で動作する環境とテストデータが必要である。特にASBの実施には，高いスキルとコスト(作業量，作業時間)が求められ，このような試験環境の構築を容易にする工夫が重要である。

性能監視は，運用後のシステムの稼働状況及び性能をモニタし，想定していた性能が得られているか評価する。システムが提供する性能指標をモニタするだけでなく，運用後のシステムのワークロードとSLAで定義したそれとを比較することも重要である。特に，ワークロードが定義時よりも大きい，あるいは増大傾向にある場合，システム拡張の規模と時期を判断する基本的な情報となる。一般に稼働しているシステムは常に増大傾向にあり，特にデータベースのデータ量は直接性能に影響することが多い。性能以外にも蓄積データ量の変化にも注意しなければならず，キャパシティマネージメントにも密接に関係している。

3. シミュレーションによる性能予測

以下に，シュミレーションによる性能予測の手法と事例を紹介する。

シミュレーション手法が適用される第一の場合は，対象システムが存在しない場合である。例えば，想定されるワークロードからサーバ機の仕様や必要台数の予測をしたり，ハードウェアの利用率や応答時間の予測，現行システムの拡張時の負荷予測など，シミュレーションは開発初期の性能のリスク管理に有効な手段である。

第二の場合としては，実測ではコストがかかり過ぎ試験環境の構築が困難であったり，比較検討の条件が多い場合である。シミュレーションは，大規模かつ複雑なシステムの全体を評価でき，システムの構成を決定する場合も，作成したモデルのパラメータを変えることで繰り返し評価でき，システムの構成を最適化できる。

3.1 シミュレーションの概要

シミュレーションは，(1)モデルの作成，(2)実行と結果の解析，(3)モデルの検証の三つの段階に分けられる。

モデルは，CPU，ディスク，ネットワーク，OS，データベース等，システムのハードウェアとソフトウェアの構成要素を部品モデルとして予め作成し，これらを組み合わせて構築している。部品モデルの再利用により，対象システムが変更されても効率的にモデルが構築できる。異なる仕様のハードウェアを用いた場合，あるいは分散されるソフトウェアの配置を変更した場合など様々な条件に対し柔軟なモデル構築が可能である。また，モデルの精度は，シミュレーションを実施する時点で要求される性能指標の精度で決まり，適切な詳細さの部品モデルを選択することが重要である。

システムの提案時には，システムの全体的な応答性，利用率の見積りが求められる。この時点では利用できる情報は少なく，細かな部分にとらわれずシステム全体のモデルを作成し，サーバ機の利用状況，ユーザ数に対する性能の傾向など定性的な予測を行っている。

設計段階でシステムの実現方法に多数の選択肢がある場合には，各実現方法間のトレードオフを把握することが要求される。モデルのパラメータを変え，繰り返しシミュレーションを行い，システムの設計仕様を決定するための定量的な根拠を得ている。また選択した実現方法の特性を捉えることで，後の開発工程で発生する問題の早期発見と解決にも役立つ。

開発後期や運用段階では，実測による性能評価も可能となるが，準備と実施に時間がかかる実測の事前評価のために，シミュレーションを実施し，実測の試験仕様を作成している。実測とシミュレーションの連携は，モデル検証に加え実測試験仕様作成の意味でも重要である。

3.2 サーバ機選定のためのシミュレーション事例

提案時のサーバ機の選定を目的としたシミュレーションの事例を示す。サーバ機の選定は，想定される各業務の重さとその発生頻度とを積算し，安全率を掛けて見積る“積み上げ方式”が一般的である。しかし，CPU，ディスクなどの計算機資源の競合による待ちが発生し，この方式では正確にサーバの負荷を予測できない。

積み上げ方式で用いた同じ情報から，シミュレーションでは**図1**のような結果を得ることができる。ここでは，tpmC値[*4]で表わしたサーバ機の処理能力を変化させたときのサーバ機のCPU利用率を予測した。一般に，CPUの利用率は60～70%までが限界とされているので，利用率の平均と分散から危険率を求め，この事例では，必要な仕様は4 200(tpmC)以上であることが予測される。

更に，システムのバックグラウンド負荷を含めた予測や，ユーザ数が増えた場合の予測，サーバ機を増やした場合の効果の予測など，多くの想定(What-if)をシミュレーションで評価できる。

3.3 サーバ機の負荷予測のためのシミュレーション事例

サーバ機が選定されアプリケーション仕様がある程度明確になった段階で，システム基本設計の妥当性を評価するためのシミュレーション事例を示す。アプリケーション仕様から見積もる処理時間を用い，選定サーバ機のCPU利用率を予測し，設計の妥当性を判断する。サーバ機のCPU利用率を測定するために，ネットワーク構成全体はモデル化せず，サーバ機と直接サーバ機に接続しているネットワークセグメントのみをモデル化した。

シミュレーションの実行結果として得られたCPU利用率を**図2**に示す。実線はCPU利用率に対する頻度(その利用率で処理された合計時間の比率)を表わしている。破線はあるCPU利用率で処理される業務数の累積比率を表わしている。全体のCPU利用率の平均は21%であり，分散の片寄りも大きくないことから，選定サーバ機への負荷は安全な範囲であることが予測される。また，**図2**から90%の業務は約40%以下のCPU利用率で処理されていることが分かる。この結果からアプリケーション仕様の見直しは必要ないと判断された。

この事例では，CPU利用率のみに着目して結果の分析を行った

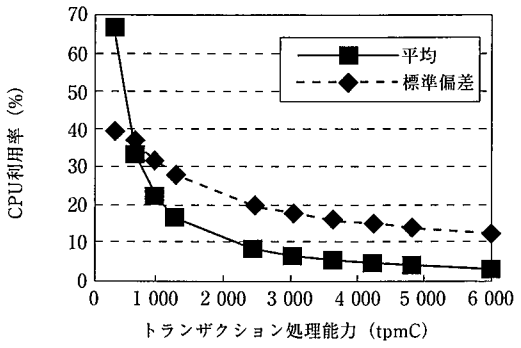


図1 サーバ機の処理能力に対するCPU利用率

[*4] tpmC値：TPC-Cのトランザクション実行比率と応答時間要件を満足した測定における，一分間当たりの特定トランザクションの処理件数。

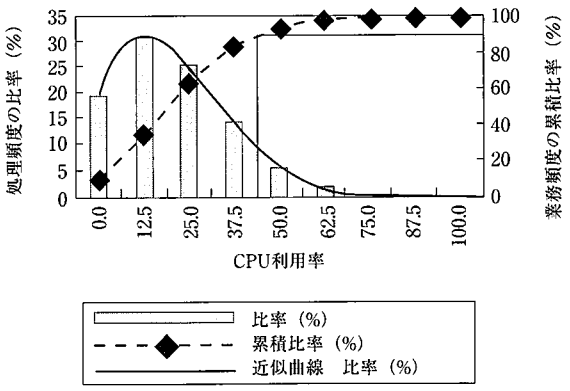


図2　CPU利用率に対する処理頻度の比率

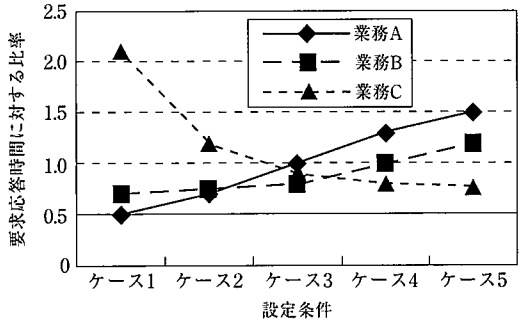


図3　設定条件に対する応答性の変化

が，シミュレーションは，業務毎のディスク，ネットワークの利用率も求めることができるので，性能問題の発生箇所の特定と原因究明も行うことができる。

3.4　ミドルウェアの構成決定のためのシミュレーション事例

この事例は，性質の異なる業務処理，比較的長い処理時間を要するオンラインバッチ処理と処理時間の短いオンライントランザクション[*5]処理とを同一システムで運用する際の，各業務処理に対する資源配分を決定するためのシミュレーションである。資源配分はデータベースアクセスミドルウェアが司るが，自由度が大きいためミドルウェアの適切な構成を決めることは困難であった。

ミドルウェアの最適な構成を決めるため構成を設定するパラメータを変え，繰り返しシミューションを行い，応答時間を予測した。**図3**は，設定パラメータが五つの場合について，業務毎に予測値と要求される応答時間の比率を表わしたものである。全ての業務が性能要件を満たす設定は，ケース3の設定であることが分かる。

この事例では，ピーク時の負荷のみでワークロードを作成したが，負荷変動を含めたワークロードでの性能評価は重要であり，シミュレーションは，負荷変動に対する応答性の変化の予測も行うことができる。

4. 実測による分散システムの性能検証

本章では，アプリケーションスペシフィックベンチマーク(ASB)の実行方法の一例としてBenchWorks™[*6]を利用した実測に基づく性能検証の実施方法を紹介する。

BenchWorks™は，データベースサーバだけでなく，データベースアクセスツール等のミドルウェアを利用して構築されるシステムのネットワークを含めた分散システムの性能評価試験支援ツールである。BenchWorks™は当センターにおける研究開発の成果物であり，新日本製鐵内外のシステム構築案件において性能検証に活用している。

4.1　BenchWorks™の概要

BenchWorks™は，ASBの実行環境構築支援ツールとして，初期から完了段階までの開発サイクル全般に渡って性能評価試験を支援する。主な機能は，(1)データベースモデリング機能，(2)ワークロードモデリング機能，(3)ワークロードの自動実行と結果収集・解析であり，段階的に精度を高めながらデータベースモデル，ワークロードモデルを構築できる。

ASBを行う際の問題は，測定のための試験環境が整わないことである。実際のシステムと同一か，同等のハードウェア(HW)環境で測定を行わなければ精度は下がり評価の曖昧さが大きくなる。HW環境だけではなく，データベース構成やアプリケーション構成も同様である。試験環境をどこまで実際のものに近づければ意味のある試験となるのかを見極め，検証の目的に応じた試験環境を構築することが必要となる。

開発の初期段階では，試験に必要なシステムや業務の情報が整理されておらず，試験仕様を作成できないことも問題となる。また，この段階では，HW環境もないであろうし，アプリケーションもできてはいないので，試験環境としてHWを準備し，その上でデータベースとワークロードをともに作成しなければならない。BenchWorks™は試験用データベースの作成機能，ワークロードの作成，実行機能を提供する。

開発の後期においては，HW環境は整い，アプリケーションもある程度完成しており，これらの環境を取り込んで試験を行うことが求められる。また，既存システムの性能評価を行う場合にも，同様に，試験環境，アプリケーションともに揃っていることになるので，これらを利用しての試験を行う必要がある。BenchWorks™を利用してこのような要件を満たした試験を行うことが可能である。

BenchWorks™は，作成したデータベースに対して定義したワークロードを自動実行し，実測結果としてシステムのスループットや応答時間のグラフ表示及び各種統計値を算出する。

4.2　データベースモデルの作成

データベースモデルの作成にあたっては，表領域，表，クラスタ，索引，制約等のデータベースオブジェクトの仕様を整理し，BenchWorks™のGUIを利用してデータベースの定義を行う。BenchWorks™では，データベースの論理構成だけではなく，ファイル配置等の物理情報も定義できる。これらのデータベース定義情報からBenchWorks™はDDL(Data Definition Language)[*7]を自動生成してDBMS(Database Management System)に送りデータベースの作成を行う。

同時に，列属性についても整理し，BenchWorks™を利用して定義していく。定義用のダイアログウィンドウで各列ごとに，データタイプ，データ分散，データ範囲とステップ幅(数値データの場合)，文字列長とバリエーション(文字データの場合)，NULLデータの発生頻度等の列属性を定義する。BenchWorks™では，これらの処理をダイアログウィンドウを利用して簡便に行うことができる(**図4**参照)。

列属性の定義は，性能評価試験の結果を大きく左右する。特に，

[*5]　トランザクション：それ以上分割できない一連の業務処理。

[*6]　BenchWorks™は，新日本製鐵(株)の登録商標。

[*7]　DDL(Data Definition Language)：リレーショナルデータベーススキーマの定義言語。

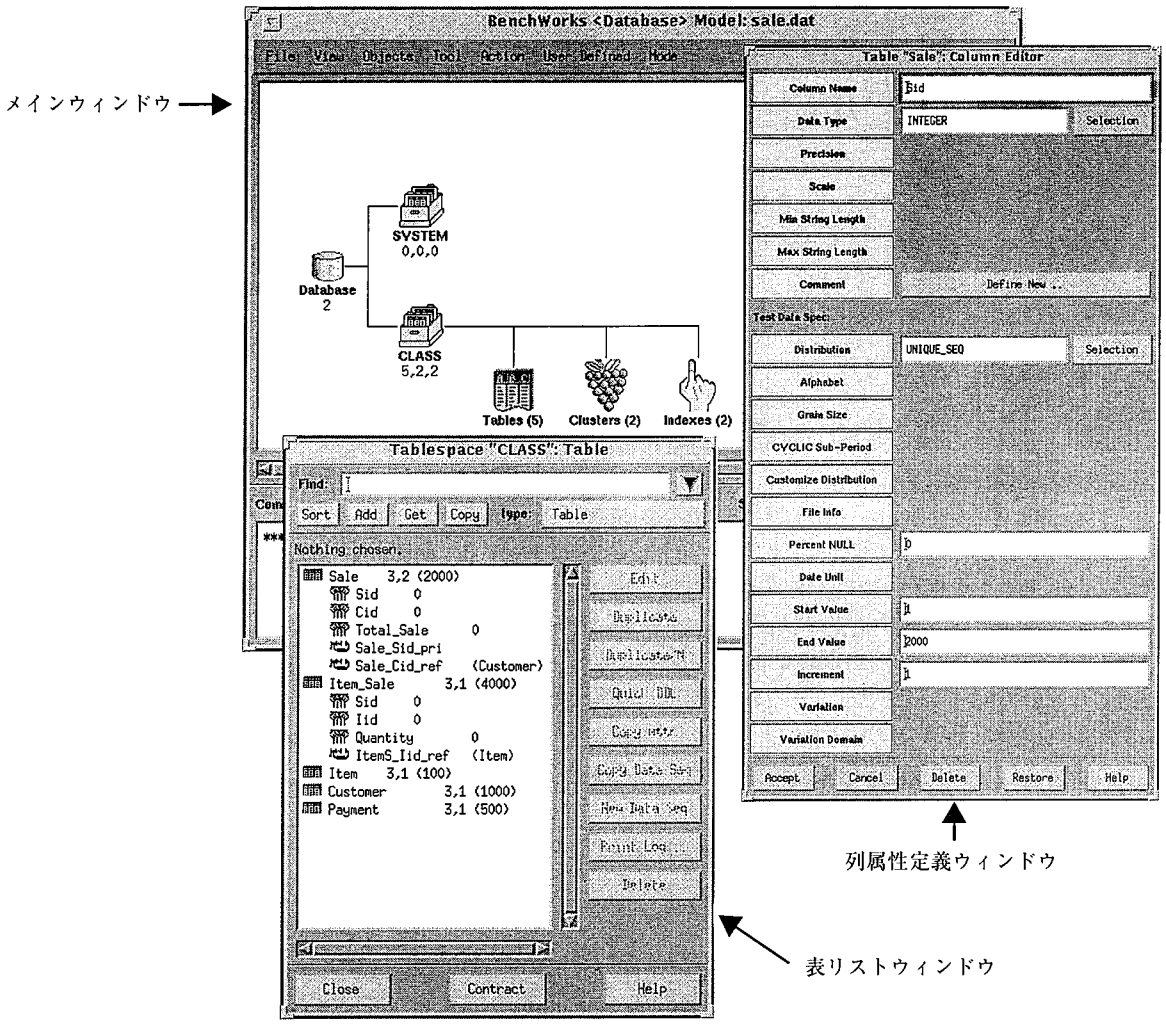


図4 データベースモデリング画面例

列のデータ分散は性能に大きく影響を与えるので，ワークロードモデルで定める各トランザクションのヒット率やヒット件数を考慮して定義することが必要となる。列属性の定義には，実際のシステムの状態をできるだけ忠実に反映させることが重要である。

BenchWorks™は，列に関する定義情報を基に表データを自動生成し，上述のDDL文を送り，定義したデータベースにデータを挿入し，試験用のデータベースを自動的に作成する。BenchWorks™は，このように繁雑な試験データの生成を自動化し，性能評価試験環境の構築を容易にする。

4.3 ワークロードモデルの作成

ワークロードモデルの作成にあたっては，トランザクションの処理内容，トランザクションの発生パターン(トランザクションの混合比率，発生頻度，処理フロー，同時アクセスユーザ数等)についての情報及び性能評価試験の実行環境について整理し，それらに基づきワークロードモデルを定義する。

トランザクションの処理内容については，作成したデータベースモデルのどの表に対しアクセスし，どのような処理を行うのか，また処理に際して従うべき制約(ビジネスルール)に関する情報が必要となる。検索条件やその条件でのヒット率，ヒット件数等の情報も有用である。

BenchWorks™は，ワークロードモデルの作成方法として二つの方法を提供している。一つは，BenchWorks™のワークロード定義のためのSQL(Structured Query Language)[*8]を拡張した専用の簡易言語によるモデリングであり，もう一つは，BenchWorks™のワークロード定義用のCライブラリを利用してのC言語によるモデリングである。簡易言語でのモデリングでは，開発の初期段階にプロトタイプシステムを作成し試験を行う場合に有効であり，簡便にワークロードモデルを記述できる。BenchWorks™のライブラリを用いたC言語のモデリングでは，より詳細にモデルを記述でき，データベースアクセスのミドルウェアや既存のアプリケーションを利用したワークロードモデルを作成できる。開発後半や既存システムの性能評価試験において有効である。

性能評価試験の実行環境については，クライアントプロセスを実行するワークステーション(WS)及び各WSでいくつのプロセスを起動し，実行するかを定義する。図5の例では，データベースサーバと2台のWSをネットワークに接続し，各WSでそれぞれm個とn個のプロセスを実行する場合を示している。この場合，m+nのクライアント数をBenchWorks™のターミナルエミュレーション機能を用いて実現している。この機能により，開発の初期段階でHWが揃っていない場合に，限られた資源で試験環境を構築し試験を実行

[*8] SQL：リレーショナルデータベースの操作言語。

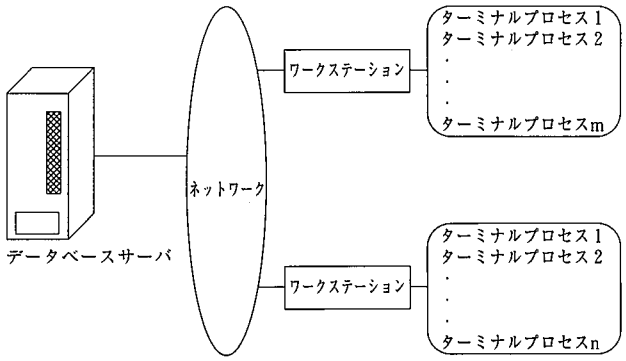


図5　性能評価試験実行環境例

することができるのもBenchWorks™の大きな特長の一つである。

4.4　評価試験の実行と結果の解析

BenchWorks™はワークロードモデルの定義に基づき自動的に試験を実行する。定義に従い各WSに所定の数のクライアントプロセスをターミナルエミュレーション機能を使用して起動し処理を実行する。試験結果として，クライアントプロセスから起動された全トランザクションの所要時間が記録される。この結果よりBenchWork™は，処理したトランザクション数とシステムのスループットや応答時間等の統計値を算出するとともに，スループットの経時変化と応答時間のヒストグラムのグラフを作成する(**図6**参照。なお，グラフ表示にはグラフ作成ソフトウェアXmgr[*9]を利用している)。

システムの性能評価のためには，データベースモデルやワークロードモデルの各種パラメータを変更して試験条件を変えながら繰り返し試験を行う。単にシステムの応答時間を測定するだけではなく，システムのブレークポイントの確認やシステムの性能特性の検証が重要となる。

BenchWorks™は，表サイズの変更や表編成といったデータベースモデルの変更や，アクセスユーザ数やトランザクションの発生パターン等のワークロードモデルの変更を容易に行うための機能を提供し，試験環境構築の作業負荷を軽減し，性能評価試験の所要時間を短縮する。

5. 結　　言

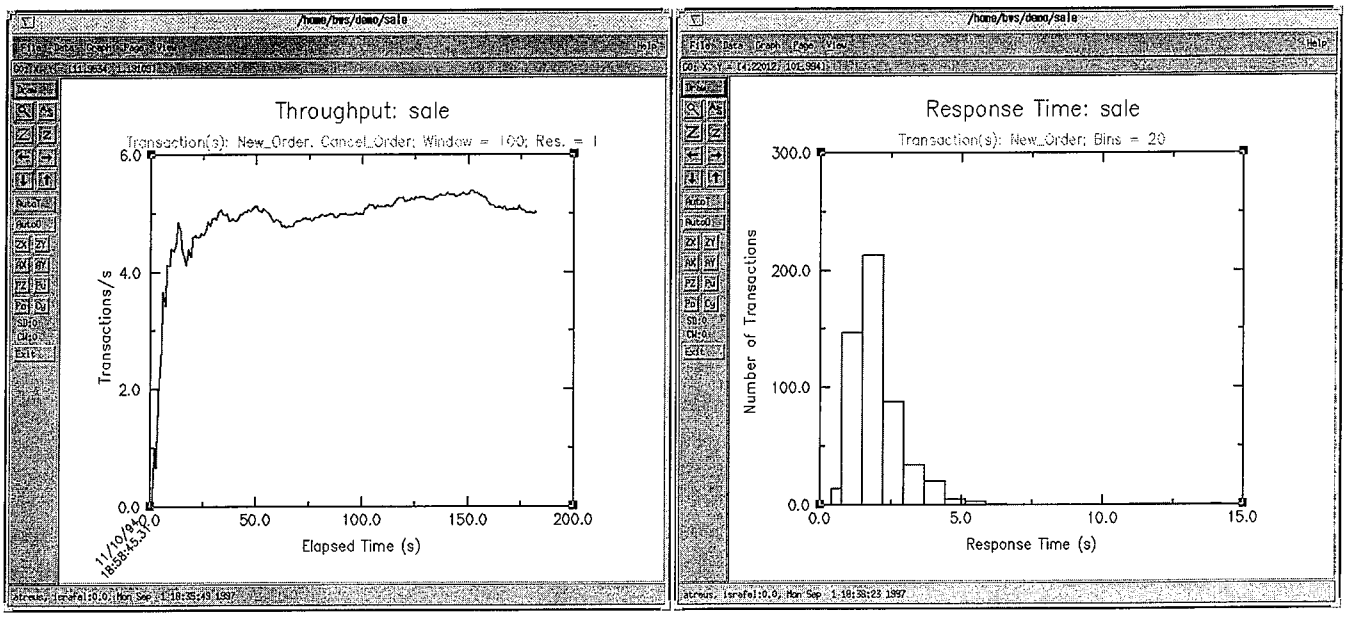
最近急速に必要性が高まっている分散システムの性能評価への当センターにおける取り組みについて紹介した。

性能評価技術それ自体は，必ずしも新しい技術ではないかもしれないが，実際の評価にあたっては，各要素技術は勿論のこと，対象システムに関する知識や，顧客との折衝技術等も要求され広範な知識そして技術力が必要である。

当センターでは，性能予測，実測，監視の三つの要素技術を核に，開発フェーズの全てに渡る性能評価のあり方について，製鉄所等の社内向けシステムも含め実プロジェクトと連携しながら研究開発を進めている。また，1997年7月に新日本製鐵エレクトロニクス・情報通信事業部ネットワークシステムソリューション部との連携で当センター内に"ベンチマーク＆コンサルテーションセンター"を設置し新たな拠点としてその運営にも当たっている。

参照文献

1) Jain, R. : The Art of Computer Systems Performance Analysis, 1st ed. John Wiley & Sons, Inc. 1991, 685p.



(a)スループットの経時変化　　(b)応答時間のヒストグラム

図6　試験結果のグラフ表示

[*9] Xmgr：Copyright 1991-1996 by Paul J. Turner, Portland, OR All Rights Reserved.