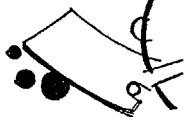


報 告**パネル討論会****シス テ ム 性 能 評 価**

昭和 59 年度前期第 28 回 全国大会†報告

パネリスト

紀 一誠¹⁾, 益田 隆司²⁾, 村岡 洋一³⁾, 村松 洋⁴⁾
吉澤 康文⁵⁾, 司会 小林 久志⁶⁾

司会(小林) いま紹介を受けました小林でございます。

本日は、この情報処理学会のパネルセッションの二つのうちの一つに、われわれの「システム性能評価」というテーマを取りあげて頂きました、情報処理学会の理事の方に厚くお礼申しあげます。

コンピュータ・サイエンスは往々にして、サイエンスというよりも、むしろアートだというコメントとか批評がときどきなされるんですが、システム性能評価の分野は、コンピュータ・サイエンスの中では、わりあい理論と実際が結びつく数少ない分野だと考えております。

システム評価の理論及びその応用ということに関しての疑問とか、問題点、今後の方向ということに関して、いろいろ疑問があると思いますけれども、ここに並んでおられる 5 人のパネラの方に、15 分から 20 分ずつそれぞれ得意の分野を、その現状及びこれからを見通しということを話して頂いて、その後会場の方も含めて、自由な討議に入りたいと思います。

まず、5 人のパネラの方を紹介させて頂きまして、一番向こうにすわっておられる益田隆司先生は、筑波大学電子・情報工学系の助教授をされておりまして、研究テーマとしましては、オペレーティングシステムの制御方式、性能評価、特に記憶管理の分野において、いろいろ論文を出されております。

益田先生は、38 年に東大工学部の応用物理学科を卒業されまして、40 年に修士卒、それから日立製作所中央研究所とシステム開発研究所で 12 年 OS、それから TSS の設計、開発評価というような仕事に従事されまして、52 年から筑波大学のほうにご勤務です。

2 番目にお話を頂く吉澤康文氏は、日立製作所シス

テム開発研究所に勤務しております。吉澤さんは、42 年に東京工業大学の応用物理を卒業されまして、その後ただちに日立の中央研究所に入社されまして、5020 TSS などの開発に従事されました。特に専門とされております分野は、会話型計算機の性能向上方式の研究、それからオペレーティングシステムの記憶管理の研究などあります。

3 番目に話して頂きます村松洋氏は、44 年に東大理学部物理学部を卒業されて、47 年に富士通に入社されています。50 年から OS の開発部門内における性能評価などを担当されておりまして、開発部門内での OS 評価、ツールの開発、ユーザサポートに従事されております。

4 番目のパネラであります紀一誠氏は、日本電気の C & C システム研究所勤務です。紀氏は待ち行列理論とその性能評価への応用ということを中心として研究されております。東大工学部の計数工学科数理工学コースを 43 年に卒業されて、ただちに日本電気に入社され、55 年まで各種コンピュータ・システムの性能評価作業に従事されております。55 年 6 月より現職におられます。

私の隣りにすわっております村岡洋一氏は、日本電信電話公社の横須賀電気通信研究所データ通信研究所に所属されておりまして、公社計算機 DIPS の開発及びその新サービスの開発などに従事されております。村岡氏は、早稲田大学の電気通信学科を 40 年に卒業されまして、45 年米国のイリノイ大学で博士号 PhD を取得されております。イリノイでは、並列処理システムの研究、それから Illiac-IV ソフトへの開発などに従事されました。

以上のそぞうたるパネラの方々を交えてのディスカッションでありますので、いろいろ活発な質問とか、ご意見が出るかと期待しておりますが、まず最初皮切りとしまして、私が 5 分間ほど「性能評価の分野

† 日時 昭和 59 年 3 月 15 日 (木), 12:30~14:45

場所 電気通信大学 [A 会場]

1) 日電, 2) 筑波大学, 3) 横須賀研 (現早稲田大学),

4) 富士通, 5) 日立, 6) 日本 IBM

における歴史的展望とこれからの課題」というようなことで問題提起を行いたいと思います。

いわゆるコンピュータ・パフォーマンス、コンピュータ・システムの評価という分野が、コンピュータ・サイエンスの中で一つのディスプリンとして認識されるようになった時期は、1960年代の後半ではないかと記憶しております。

初期は、大型のオペレーティングシステム・スループットとか、レスポンス・タイムを向上したいという立場から、多重プログラミング、それからタイムシアリングの性能解析に関心が向けられました。その後1970年前後に出てきた仮想記憶システムなど設計完了する以前から、あるいは製品を発表する以前から、その性能を予想するとか、さらには設置したあとに、それを調整するということのために、性能解析の方法や道具が真剣に取りあげられました。

1960年代がコンピュータ・オペレーティングシステムが目覚しい発展を遂げた時代であるのに対比して、70年代は、コンピュータ・ネットワークの研究開発が、米国での ARPANETなどを先駆として飛躍した時期であると思います。それに伴いシステム性能評価に関する研究分野も、ネットワークの設計、それからその性能の評価という立場から、上述の ARPANET の他に衛星通信ネットワーク、解析評価、さらには、ALOHA ネットワークやパケット、無線ネットワークなどに導入されたランダムアクセス方式やスケジューリングの解析へと発展しました。

また、最近は日本でも話題になっています、ローカル・エリア・ネットワーク (LAN) の設計、性能解析というものに、システム性能評価のコミュニティがかなり貢献しています。

1980年代に関しましては、いろいろこれからパネラの方からも話があると思うんですが、私見としましては、これからの高機能ワークステーション用のマイクロプロセッサとか、スーパーコンピュータ、さらにはデータフローマシンのような新しいアーキテクチャのコンピュータに関する性能評価というようなものに今後のテーマが移っていくのではないかと予想しております。

システム評価の方法としましては、大きく分けて三つあります。一つは待ち行列、確率過程などを中心とした数学的モデルの応用と、第2番目としては、そういう数学的モデルでは必ずしもあらわせないようなかなり複雑なシステムを対象とするときに使用される

表-1 システム性能評価の手法

(i)	待ち行列理論 確率過程論
(ii)	シミュレーション・モデル データの統計的処理
(iii)	実動システムの測定 ハードウェア及びソフトウェアモニタ

シミュレーションの手法及びその統計的な解析です。

3番目としては、実際に稼動しているシステムを測定して、その性能を評価あるいは改善するということを目的として具体的にはハードウェア・モニタ及びソフトウェア・モニタなどを使った、かなり実験的な方法、以上の三つに分けられると思います。

こういうシステム評価に関する研究者の場としましては、ここにすわっておられる益田先生を主査とする情報処理学会の「計算機システムの制御と評価」研究会というのがあります。

これは、今年度から新しく「オペレーティングシステム」研究会ということで、テーマをもっと広くカバーする予定ですが、これが日本における主たる研究者の意見の交換の場であります。米国の ACM の中の一つの専門分野グループとして SIGMETRICS があります。このグループも1年に1回、計算機性能評価に関するカンファレンスを開催しています。

そのほか、IFIPS の中にワーキング・グループ、7.3 (WG 7.3) というコンピュータ・システム・モデリングに関するワーキング・グループがありまして、現在私が昨年よりそこのチアマンをやっていますが、そのワーキング・グループの主催する国際シンポジウムが、約1年半間隔で開かれております。ことしは12月末にパリで行う予定であります。

それから専門誌としまして1980年より国際雑誌「Performance Evaluation」が年に4回(クオータリ)のジャーナルとして North-Holland 社から出版されています。私がそのチーフ・エディタ(編集主査)をつとめています。

大変表面的な展望でしたが、私のイントロダクションはこれにて終えまして、早速益田先生に最初のテーマであります「オペレーティングシステムの記憶管理」という立場から、15分ないし20分ほどお話を

表-2 性能評価システム研究者の場

情報処理学会	「計算機システムの制御と評価」研究会
A C M	SIGMETRICS
I F I P	WG 7.3 の Performance
国際誌	"Performance Evaluation"

を伺いたいと思います。

益田 筑波大学電子・情報工学系にあります益田です。

きょう私は、オペレーティングシステムの記憶管理、特に仮想記憶の管理ということで、これまでこの分野で仕事をしてまいりましたので、お話を聞いてみたいと思います。

最近主記憶装置が非常に低価格になって、大容量化してきたこと、また、この分野では、過去十数年にわたって非常に活発な研究がなされてきたことの二つの理由で、記憶管理の研究の最盛期は、過ぎつつあるところではないかと思いますが、性能評価の分野では、小林さんのスライドにもありましたように、はっきりした技術的な成果が得られている一つの重要な分野です。

今後は、小型機、超小型機の分野でも仮想記憶の技術が利用されるようになります。また、大規模なデータベースが利用される環境では、仮想記憶管理の方法もこれまでのものとは異なったものになる可能性もあります。そのような観点からみると、記憶管理の研究の重要性は今後も続きます。

そこで、ここでは、仮想記憶の制御方式に関して、これまでの技術を要約し、さらに今後の研究課題について、触れてみたいと思います。

この技術の最大のきっかけとなりましたのは、1965年にその構想が発表され、その後5年ぐらいかけて実際の開発が行われたMITのMULTICSシステムではないかと思います。

MULTICSシステムは、もう20年近くも前に発表されたシステムですが、その技術は現在、小型計算機のオペレーティングシステムとして注目されているUNIXシステム、あるいは、IBMのMVSという大型のオペレーティングシステムに対して、いろいろな意味で大きな影響を与えており、オペレーティングシステムの歴史のなかで、忘れることができないシステムです。

表-3

回顧と展望

1960年代よりOSの性能解析
多重プログラミング
タイム・シェアリング・システム
仮想記憶装置
1970年代の計算機通信ネットワークの設計及び解析
ARPANET
広域回線ネットワーク
衛星通信システム
地上無線ネットワーク
ローカル・エリア・ネットワーク
1980年代はマイクロプロセッサの性能解析？

ムです。

当時、MULTICSシステムに関して注目を集めた技術は、一つが本格的なTSSの技術であり、もう一つが、仮想記憶の技術であったわけです。

特にページングだけではなくて、セグメンテーションの機能まで含んだ本格的な仮想記憶を提案した点が特徴であったわけです。

性能評価の面からみると、TSSに関係しては、これをきっかけとしてTSSのスケジューリングの方法が活発に研究されるようになりました。日本の方では司会を勤めていらっしゃる小林さんをはじめ、やはりきょうのパネラの紀さんとか、こちらの大学にいらっしゃる鶴田先生とか、MITでMULTICSの解析で学位をとられた、現在日本電気におられる関野さんらが活躍をされてきました。

きょうお話ししますのは、MULTICSが提供したもう一つの技術である仮想記憶に関する話です。当時は主記憶が非常に高価で、小さな主記憶の上で仮想記憶を実現しようとしたのですから、仮想記憶の管理方法が重要であったわけです。当時、MULTICSの主記憶容量は384K語でした。

われわれは、HITAC 5020でMULTICS流のシステムを開発しましたが、その主記憶は最初32K語でした。最終的には64K語になりましたが、これで大体20台前後の端末を動かしていました。よほど記憶管理をうまくやらないと、たちまちシステムがスラッシング状態になってしまいます。そういう意味で、研究の必要性が非常に強くあったわけです。

古いデータで恐縮ですが5020 TSSでは、定常的な状態で、利用者のプログラムが動いている時間の割合は、5パーセント前後で、OSがその数倍も動作しているような状態になります。若干大きなプログラムが実行されたときの測定例では、経過時間400秒のうち、利用者プログラムの動作時間が、3.8秒、OSが動いている時間が240秒であるというような例もあります。システムの設計時には、仮想記憶の機能的な利点に目がいって、このような結果は予想しませんでした。MULTICSプロジェクトでも同じような状況であったと思います。このような背景から、その後、仮想記憶の管理に関して非常に活発な研究がなされたわけです。

カリфорニア大学バークレー分校のA.J. Smithが、“Bibliography on Paging and Related Topics,” ACM Operating Systems Review, Vol. 12, No. 4,

pp. 39-56 (1978) に、ページングに関する研究論文を 330 篇あまりリスト・アップしています。

また、P. J. Denning が "Working Sets Past and Present," IEEE Trans. on Software Engineering Vol. SE-6, No. 1, pp. 64-84 (1980) の参考文献に 126 篇の、特に、ワーキング・セット (working set) に関する研究論文をまとめております。

発表の年代をみると、1966 年ぐらいにきっかけがあって、1968 年から 75, 6 年までに、いい仕事が、数多くなされていることがわかります。

Denning は上述の論文のなかで、この分野で国際的に 200 人以上の第一線の研究者が活躍したと述べています。仮想記憶に関するこれらの研究を分類してみますと、

第 1 番目は、仮想記憶の管理法であり、第 2 番目は、仮想記憶の管理法を検討するために必要なプログラム動作の把握と動作モデルの作成に関するものです。

第 3 番目は、特に利用者の立場から、どういうプログラムをつければ仮想記憶を有効に利用できるかというプログラムの構成法であります。

まず、第 1 番目の仮想記憶の管理法について簡単にまとめてみます。仮想記憶の管理法としては、グローバルな方法とローカルな方法があります。グローバルな方法は、動作している利用者プログラム全体をまとめて、ある基準で、たとえば LRU スタックで管理する方法であり、ローカルな方法は、各プログラム個別の動作特性をうまくつかんで、管理をする方法です。

グローバルな方法の代表的なものとして、グローバル LRU 法があります。IBM の MVS で使っています。MULTICS, HITAC 8700/8800 の OS 7 でもこれに近い方法を使っていました。

ローカルな方法の代表的なものとしては、有名なワーキング・セット法があります。ワーキング・セット法というのは、皆様ご存じだと思いますが、あるプログラムを実行するときには、そのプログラムがウィンド・サイズと呼ばれる過去一定の時間内に参照したページ集合であるワーキング・セットを主記憶に格納して実行しようという方法です。ワーキング・セット法を使っている計算機が多いと思います。大型機の例としては、日本電気の ACOS-4/MVP もワーキング・セット法を採用しているはずです。ワーキング・セット法を修正した方法として PFF (Page Fault Frequency) 法があります。論文のレベルの話かと思っていましたら ECLIPSE の AOS/VS で採用しています。

仮想記憶管理に関して、数多くの研究がなされたにもかかわらず、グローバル方式とローカル方式のどちらが性能的に優れているかという基本的な問題に対する解答はいまだにはっきり得られていないようです。ローカル方式は、個々のプログラムの特性を把握するために、より手間がかかります。主記憶容量の大きい大型計算機ではローカル方式、グローバル方式いずれもが採用されていますが、小さな計算機では、ローカル方式が採用されることが多いようです。

最も広く利用され、その性質が理論的にも実験的にも研究されているのは、ワーキング・セット法ですが、ウィンド・サイズをどのように決めてやればよいかの根拠がないという欠点があります。

理論的にそれを決めることを試みている研究もありますが、実用性に乏しく、経験的にシステムに共通に適当なウィンド・サイズが用いられているのが通常だと思います。

Denning らは、ウィンド・サイズの許容範囲は広いということを、いくつかのプログラムに対する実験結果から報告していますが、選んだプログラムに偏りがあり、説得力のある結果ではありません。

第 2 番目のプログラムの動作とその動作モデルに関して最も重要なことは、プログラムには局所参照の性質があることが多いということです。

ワーキング・セット法は、プログラムの局所参照性を仮定した記憶管理の方法です。個々のプログラムが局所参照しているページ集合の推定量がワーキング・セットであるわけです。

いくつかのプログラムについて調べてみると、程度の差はありますが、局所参照の性質が存在するプログラムが多いことは確かです。局所参照するページ集合が時間とともに変化します。プログラム実行時、制御があるページ集合内に留まっているとき、そこにフェイズ (phase) が存在したと呼ぶことにします。

仮想記憶管理方式の解析、評価をしようとするときには、このようなプログラム動作のモデル化が必要になってきます。プログラム動作モデルとして、各ページへの参照確率を定めるような単純なモデル、利用される LRU スタックの各深さへの参照確率を定め、それにしたがってページ参照列を発生させるようなモデル、その他数多くのモデルが提案されています。LRU スタック・モデルはしばしば利用されています。プログラム動作のモデル化で重要な一つの点は、局所参照するページ集合が、先ほど述べたように、時間とともに

変化していくという性質をうまく表現することです。この視点から最も興味深い研究は、1975年頃に、A. W. Madison と A. P. Batson が提案した BLI (Bound-ed Locality Interval) モデルです。BLI モデルは、プログラム実行時のページ参照列から、局所参照するページ集合が、時間的に変化する様子、すなわち、フェイズの構成を LRU スタックに基づいて自動的に作成することを試みたすぐれたモデルです。BLI モデルでは、フェイズは階層的に存在することになります。

Madison らは、多くのプログラムの配列の参照列のデータに対して、BLI モデルを適用することにより、実際のプログラムの局所参照の特性がかなり精度よく表現されるということを確かめています。

仮想記憶管理、プログラム動作とそのモデルに関して、これまでの研究の概略を述べましたが、私たちは最近、従来とは異なった角度からこれらの問題に興味をもっておりまます。

仮想記憶管理の方法にしても、プログラム動作の把握にしても、これまですべてプログラム実行時のページ参照特性から得られる情報に基づいておりました。

これに対して、私たちは、プログラム実行時の局所参照の性質は、原始プログラムの構造に起因しているのではないかと考えたわけです。そして、局所参照の性質は、原始プログラム内の繰り返し構造に原因があることを確かめました。その考え方に基づいて、実行時のページ参照列からだけではなく、原始プログラム内の繰り返し構造から得られる情報をあわせて利用することによって、LC (Locality Contour) モデルとよぶプログラム動作モデルを提案いたしました。そして、LC モデルを先に述べた BLI モデルと実験的に比較した結果、主なフェイズの構成は、両方のモデルでほとんど一致していることが確かめられました。

LC モデルの利点は、その考え方を記憶管理に利用することができる可能性があることです。コンパイラが原始プログラム内に、実行時、フェイズの原因となりそうな繰り返し構造を発見したときには、そのループ構造に対応した目的プログラム内に、ループの入口、出口、繰り返し点に、必要に応じて、特別なスーパーバイザ呼び出し命令を埋め込んでやれば、実行時に、オペレーティングシステムは、それらのスーパーバイザ呼び出し命令によって、局所参照するページ集合を精度よく推定することができるようになる可能性があります。

特に、ワーキング・セットで局所参照するページ集合をうまく推定できないような場合には、ワーキング・セット法から、提案したような、原始プログラムの繰り返し構造を利用する制御に切り換えてやることが考えられます。この方法は、ページ参照列から得られる情報だけを利用した従来の方法とは大きく異なっていますので、本当に実現可能があるかどうかはわかりませんが、従来の方法による研究が非常に活発に行われたことを考えますと、提案したような観点からの検討もしておく価値があるのではないかと思います。

最後に、プログラムの構成法に関しても、プログラムの局所参照性を改善する試みが数多くなされています。新しいプログラムを開発するときの一般的な留意事項、すでに開発されているプログラムを再構成して局所参照性を改善する方法、大次元行列を取り扱う場合、特に、局所参照性を考慮したアルゴリズム、などに関する研究が数多くなされています。

ごく概略でしたが、仮想記憶管理に関するこれまでの技術をまとめてみました。

最後に、この分野で、私が興味を持っているこれからの研究課題について簡単に述べておきます。

これまで、大体プログラムの動作を対象にして、記憶管理技術の検討がなされてきましたが、今後はデータベースを対象にした記憶管理技術が重要ななるのではないかと思います。データベースでは、これまで述べてきたような局所参照の性質もどの程度存在しているのかが非常に疑問です。したがって、これまでの記憶管理技術が有効かどうかは改めて検討する必要があります。

たとえば、現在、データベースを動作させる場合、データベース管理システムを利用しますが、データベース管理システムはオペレーティングシステムの上で動作します。データベース管理システムは、オペレーティングシステムとは独立に、バッファ管理を行っております。すなわち、記憶管理が二重構造になっているわけです。オペレーティングシステムの仮想記憶管理との関係で、データベース管理システムのバッファ管理を考え直してやる必要があるように思います。

また、関連した課題としては、記憶階層の設計も重要な問題です。

これらの課題を検討するためには、まず、データベースの参照特性を把握することが必要です。

1981年7月の Comm. ACM に INGRES を開発し

た M. Stonebraker がこのあたりの研究課題をまとめています。また、最近では、京都大学の津田先生、大久保先生が開発されているデータベース向きのオペレーティングシステムも問題意識は上述のようなところにあるのではないかと思います。

以上で私の話を終わらせていただきます。

司会 どうもありがとうございました。

ご質問もあるうかと思いますが、質問とか討議は最後にまとめて行いたいと思いますので日立製作所の吉澤氏に、「アーキテクチャの設計に与える影響」というテーマで、お話を伺いたいと思います。

吉澤 日立製作所の吉澤です。私は性能評価というものが、オペレーティングシステムにどのように影響を与えて、そしてオペレーティングシステムがどういうふうに発展したのか、というようなことを中心に話をしたいと思います。

まず背景ですが、先ほども説明がありましたけれども、やはり性能に対するニーズが最近急激に大きくなってしまっておりまして、その元をただすと、先ほどの MUL TICS を中心に進められた TSS の開発があったと思います。

1960 年代の後半に TSS ですとか、あるいはオンラインリアルタイムシステムというものが急速に発展しまして、いわゆる対話処理計算機の利用形態がそれ以降ふえたわけです。

このように人間が計算機を直接使用することになりますと、どうしてもそのレスポンスタイムを保証することが重要な課題になってくるわけです。

最近は、さらにマンマシンで計算機を利用するという分野が大きく広がっておりまして、従来ですとバッチ処理で数日かかるようなものも、いまではリアルタイムで処理してほしいという要求が出てきているわけです。

CAD や DA、それらがデータ・ベースを利用するようなことは当たり前になってきました。そうしますと大容量の、また大量の計算機資源を利用する処理が対話型で行われしかもそれを利用する応用分野が拡大しているというのが現実です。

こういう背景の下にオペレーティングシステムが利用者の性能へのニーズを満足させるために、いろいろな発展形態をみたわけですが、まず一つ目は、計算機資源をどういうふうに利用しているのか、それから、その計算機資源を計算機の中にあるプロセスに、どのように配分しているのか、といった様子を測定し分析

処 理

する手段、そういう要求がまずあったのではないかと思います。

第2番目に、オペレーティングシステムのアーキテクチャに与えた影響としては、そういう計算機資源をどのようにプロセス間に割りふるかというスケジューリング機能に対する要求が、いろいろ出てきたということだと思います。

そして計算機を運用する上で、その運用者が性能というものを設計するといいますか、それを決定する機能をオペレーティングシステムが提供してあげるというような機能の提供まで、現在発展してきているのではないかと思います。

この分野では、まず第1に、計算機の負荷というものをどうやって測るのかということが初期の段階で非常に重要な課題でありました。この分野での技術の発展という観点からは、この段階が性能評価というものがオペレーティングシステムに与えた影響の第1期というふうに考えることができるんじゃないかなと思います。

1960 年代の後半では、計算機を購入して、そこで TSS なり、あるいはバッチ処理なり、あるいはオンラインシステムを実現した場合に、レスポンスが悪い、あるいはバッチのスループットが上がらないということが生じて、どうして計算機というのはこんなに遅いんだろうというような疑問を持ちはじめたのが、一番最初の性能評価に対する要求であったと考えられるわけです。

CPU が負荷に対して遅いんだろうかとか、主記憶容量が少ないんだろうか、チャネルやデバイスの数が少ないんだろうか、そういうふうにシステムのボトルネックがどこにあるのかということが、はじめは皆自見当がつかなかった。

そういうことで、まず最初に必要とされたのが、計算機システムの資源をどういうふうに使っているんだろうかといった、測定用モニタがまず最初に開発されたのではないかと思います。

まず、システム全体の資源をどのように使っているかということを測定するために、オペレーティングシステムの中には、CPU やチャネルやデバイスなどがどのぐらい使用されているか知るためのカウンタを内蔵させる機能がつけられてきたわけです。

もう少し進みますと、システム全体ということではなくて、アプリケーションプログラム、あるいはオペレーティングシステム自身のプログラムの動作解析と

いうものを少しづつ調べていかないと改善はできないんだということから、処理ステップ数ですか、メモリの占有量とか、I/O の発行回数といったものを分析するツールが必要になってきたわけです。

そこで、オペレーティングシステムの技術を進歩させるためには、どうしてもよい測定器をもたなければならぬということとして、いろいろな形態の測定の手段が開発されて今日に至っているんではないかと思います。

その手段はいくつかあるんですが、まず一番簡単な測定の方法といいますのは、課金情報を利用する方法です。これは、計算機を使用したときのお金を取る課金情報を各ジョブ単位にとっております。そういうものを収集しまして、システム全体でどういった資源の使われ方をしているのかを分析するツールが開発されました。

これは比較的初期の段階であります。最近ではこれがかなりいろいろな機能をもっております。

たとえば、この図に示しましたように、(図-1) スーパバイザやデータ管理、ジョブの入出力を行う入力リーダとか、出力ライタ、それからジョブのスケジューラ、そういうところにあらかじめフックを入れておきまして、事象が発生すると、課金情報収集のプログラムに制御が移り、そこでデータを蓄積する。

これらの情報を、ここに示したような課金情報としてみると、それ以外の情報としてみると、あるいはそれらをさらに分析すれば、スケジューリングの最適化を行うことができるとか、あるいは今後のチューニングをどういうふうにしていけばよいか、データセットはどういうふうに使われているか、といったことが解析プログラムを通じて得られるということになるわけです。

それから、もう少し詳しい測定を行いたいというこ

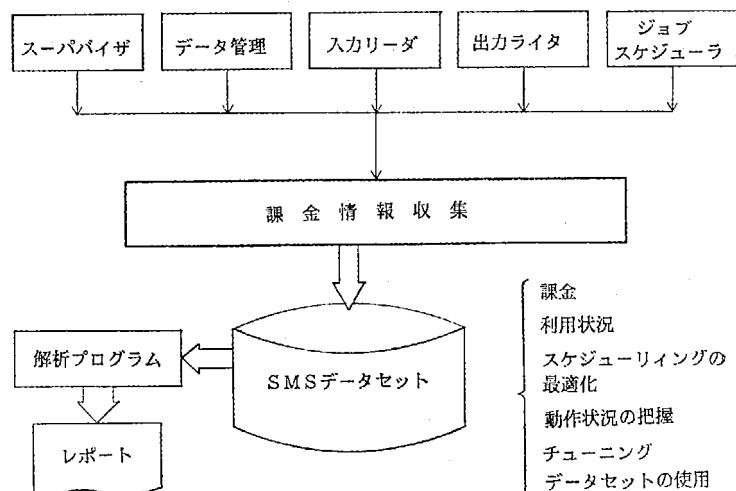


図-1 システム管理支援 (SMS) による測定方法

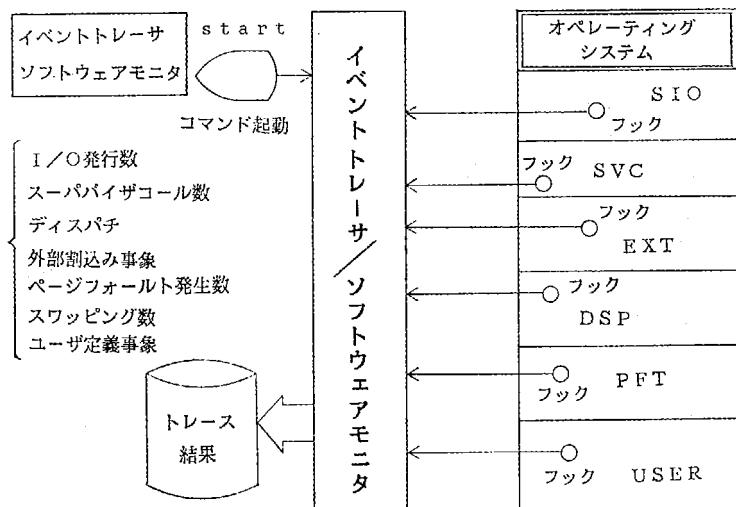


図-2 イベントトレーサ、ソフトウェアモニタによる測定方法

とになりますと(図-2)、オペレーティングシステムは、このようないろいろなコンポーネントになっておりますので、たとえば I/O の処理をする部分、スーパバイザコールの処理をするとこととか、それから CPU のスケジューリングを行う部分とかにフックを組み込んでおきデータを収集することになります。

さらにユーザプログラムにもフックを入れることにより動作情報を教えてもらいまして、これらのイベントのトレースを行います。これらを行うのをソフトウェア・モニタといっているわけですが、それでトレースした結果、あるいはサンプリングした結果を 2 次記憶に蓄積する。

こういうことを行いますと、I/O の発行数ですか、スーパーバイザコールの数、その種類、それからどういうジョブが、いつの時点でスケジュールされたか、割り込みはどういうふうに起きたのか、ページフォールトというものが、仮想記憶ですと非常に重要なポイントになるわけですが、ページフォールトがどの程度発生しているのか、会話処理などではスワッピングがどのくらい起きているのか、それからユーザプログラム、特にオンラインのプログラムですと、いろいろなアプリケーションプログラムがどのように走るか、などがわかるわけです。

そのほかツールとしては、オペレーティングシステムが管理しているいろいろな情報をある特定のプログラムでのぞきとみまして、それを統計的に編集し TSS の端末や、あるいはコンソールに表示して見せるとか、それからもっとミクロな、先ほど益田先生が話しておられましたプログラムの動作解析みたいなことを行うためには、どうしてももっとミクロなレベルの解析ツールが必要となるわけです。

その代表的なやり方というのが命令トレーサです。

これらの機能はオペレーティングシステムの中にどんどん採用されなくてはならないわけで、たとえばここに示した図(図-3)では、これはたとえば仮想計算機(VM)というものが、そのために利用されるわけですが、そこで仮想計算機の中で実行されている 1 命令ずつをこのトレーサで磁気テープの中に履歴をとりまして、あとで解析することによって、ダイナミクスステップ数ですか、あるいはメモリの参照パターンですか、I/O を発行した数、イベントの分析、こうい

うことが行えるわけです。

それによって先ほどの益田先生の説明にあったような、ああいったワーキングセットの分析が行えるようになるわけで、それが結果的にはオペレーティングシステムの、たとえばページリプレースメントアルゴリズムに反映されるという形になるわけです。

1970 年代に入りますと、TSS が非常に大規模になりました。こうなりますと、TSS ユーザの動作といいますか、たとえばシンクタイムですか、あるいはコマンド使用分布といったものがわからないと、システムの設計を行うことができないとか、また、リソースをスケジューリングする際のいろいろなパラメータ、あるいはスケジューリングの方式そのものの検討ができませんので、こういったものを分析する必要が生れたわけです。

そういう意味からすると、ユーザの端末での動きをなんとか収集しなければいけないということで、通信管理プログラムにそういう機能を入れて、モニタリングする機能をつけることが必要になってきたわけです。

次に、2 番目の重要な問題としては、いろいろな測定のツールがたくさんできたわけですが、これをオペレーティングシステムにどのような機能とし反映していくべきなのか、特にスケジューリングにどういった機能を備えればよいかということが重要になってきたわけです。まず計算機の資源を第 1 番目 CPU、そして 2 番目はメモリ、3 番目は入出力装置とわけますと、資源管理の発展のままで第 1 が CPU のスケジューリングということになると思います。

CPU の分配を行う方法としては、最も単純なものは、優先順位を与える方法ですが、TSS だけではなく、バッチがバックグラウンド・ジョブとしても動くような環境になってきますと、いろいろな性能に対する要求が出てくるわけです。

一番最初に出てきたのは、多分タイム・スライス制御によるラウンド・ロビンの方法です。これで、あるプロセスが CPU を独占してしまうことを回避することができるわけです。

それから、ダイナミック・ディスパッ칭法というのは、I/O の発行の頻度が高いプロセスに、CPU の割り付け優先度を高くしてやろうという考えです。

ダイナミック・ディスパッching のやり方というのは、計算機システム内のプロセスがなるべく

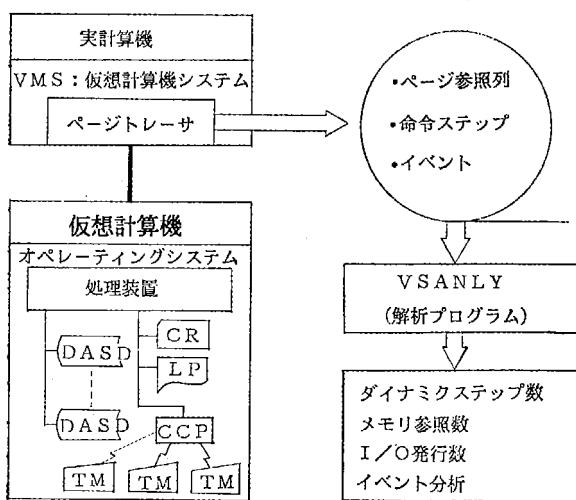


図-3 トレーサによる測定方法

こういった入出力装置のところにくついている状態になれば、スループットが高くなるわけです。

最近はまたデッドライン・スケジューリングというようなものも提案されています。このやり方は目標性能をトランザクションごとに決めまして、その締め切り時間近くなればなるほど優先度をあげてやろうというスケジューリング方式です。

最近のオペレーティングシステムでは、こういったいろいろなスケジューリング方式がなんらかの形で複合化され実現しているのが一般的になっているのではないかと思います。

次に2番目のリソースとしてのメモリ分配ですが、これは貴重なメモリを効率的に利用して多重プログラミングの多密度の向上を図ってスループットをあげよう、あるいは性能をあげようということです。

これは先ほど益田先生から説明がありましたので、省略させていただきまして、あとは3番目のリソースとしてI/Oスケジュールがあるわけですが、この分野はジョブの優先順位に従った優先順位を与えるというぐらいの機能しか、現在のところないのではないかと思います。

それから、最近のオペレーティングシステムの動向は、個々の資源、つまりメモリとかCPUとかチャネルとかといったものの個々を独立にスケジュールするのではなくて、新しいシステム資源の概念としてサービス量というような考え方を導入している点にあると思います。つまり、CPU時間とか、メモリ使用量とかI/O発行回数などの和で示したサービス量を尺度に、資源をプロセス間に分配するというような方法がとられています。

それから、3番目の問題としては、計算機を運用管理する人たちが、その性能を設計することが可能になるようなオペレーティングシステムの機能が要求されておりまして、その方面では相対的な優先度で資源を与えたり、あるいは絶対的な資源量を各プロセスに与えるという機能があります。

こうしたことを行うことによって、各トランザクションのスループット、システム全体の性能を守ることができるようにになったわけです。

最後に今後の課題ということなんですが、一つは性能評価を行う上に基本的に重要なこととして、測定の技術というものがあるわけです。

先ほどいくつかの測定の技術について示したわけですが、測定の技術としては、かなり確立されてきたの

ではないかと思われます。

しかし、確立されてきてはいるのですが、なかなか統合化といいますか、いろいろなものがバラバラにつくられてきて、必ずしもユーザーには使いやすいものにはなってないという欠点があります。また研究者にとっても使いやすいものにはなっていない。だれが使うのかということについて、よく考えられたものができるないという現状ではないかと思います。

次に、測定を行う際も、かなりのマンパワーがかかります。少し違うことをやろうとすると、そのためにはプログラムをつけ加えたり、あるいは計算機を使用したりということで、かなり経済的に高いものになってしまふ欠点があります。

こういうところが問題で、今後改善していく方法をハードウェア及びソフトウェアでなされなくてはいけないのではないかと考えられます。

2番目の課題としまして、計算機がいろいろな分野に使われ多様化しているわけです。

この多様化のために、オペレーティングシステムは汎用的なスケジューリングですとか、あるいは測定のツールとかいうものを提供してきたわけです。その結果、性能のチューニングを行おうとすると、システムはかなり複雑になっていますから、容易ではありません。これが一つのネックになっているのではないかと思います。

今後はこの辺のチューニング作業が容易にでき、性能の設計みたいなものを容易に定義できて、そしてオペレーティングシステムはそれに従ったスケジューリングを行えるような自動チューニングのようなものが、今後の技術として確立していく必要があるのではないかと思います。

以上です。

司会 大変ありがとうございました。実際のコンピュータ・システムの設計に携わっている方から、現実性をおびた有意義なお話だったと思います。

3番目に、やはりメーカーである富士通の第1ソフトウェア事業部におられます村松氏から「大型コンピュータのシステム評価」というテーマでお願いいたします。

村松 益田先生から、記憶管理の方式の話、吉澤さんから、OSを開発する上で性能に対する考慮の仕方といった話がありました。

それを受けて、実際の開発の実務レベルで一体性能をどのように考えていかなければいけないか、現実の

問題がどういうところにあるかという点をお話します。

私は富士通の大型機のオペレーティングシステムを開発する部門にいるわけですが、大型機のOSを開発する上で性能の問題を整理すると、まず第1は、方式的な問題であります。

記憶階層の管理の話、すなわちVSの使い方、マストレージの使い方、ディスクキャッシュのインプリメントの仕方、キャッシュメモリの効率に対する問題、そういったもの的方式で、システム全体の性能を大きく左右する問題があります。

方式上の問題としては、また、資源配分もOS性能を基本的に左右する問題です。

さらに、排他制御の方式とか、複数のシステムの間の同期の方式とか、方式的に性能を大きく左右して、理論的にも非常におもしろい問題がいくつかございます。

方式の問題のはかに、システムを実際に作るとなると、重要な問題がいくつかあります。

第1は、使用資源の絶対量です。資源の配分をいかに上手にやりましても、OSそのものが使う資源の絶対量が多ければ、決していい性能のOSはつくれないわけで、システムそのものが使う資源をいかに少なくしていくかが実際にOSをつくる上で非常に重要です。

第2は、使用資源のバランスであります。使用資源が少なくとも、現在使われているシステムの価格などを考慮して、I/Oとメモリのバランス、CPUの使い方とメモリの使い方のバランスなどがとれていなければいけないわけです。

第3に、使いやすさの問題があります。使いやすさは、TSSのコマンドの操作性などの問題もありますが、狭い意味での性能に限っても、動作条件が多少変わっても、予定した性能が発揮できるつくりになっていなければいけないわけです。チューニングのしやすさも、性能を考える上で非常に大きな問題であります。

こうした性能の基本的な問題を踏まえて、最近のOSについて、いくつか特徴をあげて実際に問題となることを考えてみます。一つは非常に機能が多様化してきたことがあげられます。

従来ですと、ユーザのアプリケーションに任されていた仕事を、どんどんOSの中に標準機能として取り込んでいます。それによって、OSの中に非常に

処 理

多数のコンポーネントがつくられ、それが複雑にからみ合っています。

また、他の特徴として、性能の評価という観点からは、非常に重要でめんどうな問題はインテリジェンスの分散です。

こうした特徴の一つの具体例として、富士通のJEFシステムを見てみます。すなわち、今まで英数字のデータだけを扱っているOSに、日本語のデータを扱えるようにしたならば、そのシステムの性能は一体どういうふうに変わるだろうかという例を考えてみることにします。JEFの場合日本語処理のための改造、機能追加は、ほとんどOS全般にわたっておりまして、データ管理、ジョブ管理、DB/DCサブシステム、TSSの電文処理、TSSのコマンドプロセッサ、にいたるまで日本語のための考慮が行われています。

日本語ディスプレイなどハードウェアについても、従来のディスプレイに比べて、ハードウェア内での処理の内容が複雑になっている。

システムの開発にあたって、これらの全体を見渡して方式、使用資源量、使いやすさといった性能の問題を考えていかなければいけないわけです。

もう一つ、最近のOSのつくり方の上の特徴は、機能階層を明確に分けてつくることがあげられます。

これによって、全体の制御の構造は非常に明解になりますし、拡張性や機能追加のしやすさ、機密保護などの観点からも非常に望ましいわけですが、しかし性能からは全体の流れが見えにくくなったり、部分的な改造がシステム全体の中心にどういう影響を与えるか分からなくなる傾向があります。

こうした階層化が行われますと、個々のアプリケーションや、特定の階層をつくっている人は、全体として自分のプログラムが動くのにどのぐらいのリソースを使うのか、だんだん分からなくなってしまいます。

また、機能追加が、こうした階層の一つで行われた場合、たとえば機密保護機能がファイルのアクセスの階層や端末のアクセスの階層などで行われますとシステム全体に思いがけない大きな影響を与えてしまうということも起こります。

最近のOSのもう一つの特徴としては性能面の柔軟性の追求もあげられます。

システムの構成が多少変わっても、最大限の性能が発揮できることが、実際の開発の上で大きな課題の一つとなっています。実メモリが小さくてもOSは動くようにつくられます、メモリがたくさんあれば、そ

れなりの性能を発揮することが必要です。I/O の構成に関しても、同様な配慮が必要です。

このような配慮は、逆にどのような動作条件で性能が出ればいいか、判断がしにくくなるという問題を含んでいます。

こういったことが、実際に開発の上で直面している問題の一端です。これに対して私どもでいろいろ工夫を重ねていることをまとめてみると、一つは、非常に大きなプログラムをいろいろな複数のコンポーネントに分けてつくっていますが、横断的な性能評価の努力です。また、非常に大規模になった OS に対してきわめて頻繁に機能追加だととか、改造が行われているわけですが改造時の性能検証をいかに効率的にやっていくかの努力です。小さな改造のために、知らない間に性能が少しずつ劣化している場合が、実際にはかなりありうるわけで変化を確実に追跡することにつとめています。

また、柔軟性の追求とうらはらにして、評価尺度を明確にしておくことが必要で、これも心がけていることの一つです。

こうした問題を解決する上で、裏付けとなる技術はなんであるかを列挙してみます。

第1は、性能測定の基本的な技術であります。これは、吉澤さんからも話がありました。第2は、システム設計のためには、計算機の上で動くいろいろなワーカロードを、いかに特性を整理して設計の材料にしていくか、それからそれを性能評価のモデルとして、どうやってうまく抽象化していくかの技術です。

第3は、紀さんが話してくださいますが、モデルの解法の技術があります。

こうした技術での今日、今後の課題を考えてみると、性能評価も非常に専門的な分野になっております。OS をみましても、規模が大きくなってその中の専門領域が細分化されていて、全休が分かる人間が必ずしも十分でないという状況になっています。こうした状況のため性能面でも、すべてのことが分かっている人がいれば、それほどむずかしくないはずの問題でも必ずしもタイミングで解決されていかないものがございます。今まで OS ですか、計算機のハードでいくつか大きな方式的な改良が行われているわけですが、そういったものの性能評価がタイミングで行われているかをみてみると、必ずしもそうではないわけで、OS のごく専門領域の中に閉じられた評価しか行われていないのが、やっぱり現状の大きな問題であ

り、これが第一の課題と思います。

またもう一つの課題は、実際のシステムを動かした場合のエンピリカルなデータですが、新しいシステムを設計するとき、エンピリカルなデータは必ず必要であり、大きくなるにつれて実験室的な環境でデータをとるのが、ほとんど不可能になってきています。大型計算機のユーザがどういうふうにそれを使うか、TSS の使い方がどうなっているかなどが設計のための基礎的な材料になります。しかし不十分なものしかそろっていないのが現状と思っています。

司会 どうもありがとうございました。

JEF システムなど、非常に最近の話題のシステムの性能のことなどに触れて頂きまして、大変結構なお話をでした。

いま村松氏の話の中にも、モデリングという言葉が出てきましたが、待ち行列を用いた解析的モデルの現状とその応用に関して日本電気・紀氏よりお話を伺いたいと思います。

紀 ご紹介いただきました紀です。

まず最初に、現在メーカ、ユーザ問わずにシステム・エンジニアといいますか、計算機の運用管理に携わったり、あるいは設計開発に携わる方々のおかれている状況を簡単にみてみると、その対象といたしますコンピュータ・システムそのものは、ますます全体として規模が大きくなり、制御方式も複雑化していると思われます。そこにのってくる業務も、多種多様になってきています。コスト全体も絶対値としては、どんどん増えています。要するに、対象としてはだんだん扱うのがむずかしくなる傾向にあるということです。一方、システム部門ではどういうことが行われているかといいますと、経費の節減であるとか、生産性の向上、これはほかの一般の部門と同じように要求されています。

そういう状況の中で、システム・エンジニアはどういうふうにあらねばならないかということになりますと、なるべく少ない工数で、より質の高いアウトプットをどんどん出していく訓練をしなさいということが要求されてきています。

これが現在のシステム・エンジニアの方々のおかれている一般的な状況ではないかと思います。

少ない工数で短期間に的確な性能評価をするための条件としましては、いくつか考えられますが、まず最初には明確な方法論と、それを特別なノウハウをもたなくとも、一般教養をもっている人ならばだれでもで

きる標準化が確立されていることであり、2番目としましては、そういう方法論を実際に実行いたします際に使う評価作業支援用のソフトウェア、必ずしもソフトでなくハードの場合もあるかもしれませんけど、そういうツール類が簡単に手に入る形で存在していることです。この2点が必須条件ではなかろうかと考えています。

それで、キャパシティ・プランニングということですが、この特徴をみてみると、まず一つは、先ほど来話に出てくるオペレーティングシステムであるとか、場合によってはハードウェア・コンポーネント個々のものをベースにいたしまして、それを総合的に組み上げてシステムとして構成したときに、どういう性能になるのかという総合的な評価を行うことが一つの特徴になってくると思います。

2番目は、必ずしも現在あるシステムを測定するという形だけではなくて、将来どうなるか、新しい業務がのったらどうなるか、機器構成を変えたらどうなるか、グレードアップしたらどうなるか、そういう予測という問題がつきまとするのが特徴ではないかと思います。

3番目に、これは特徴というよりも目標といいますか、バランスのとれた性能評価ということが要求されていると思います。

これはどういうことかといいますと、予測というものはコストをかけなければかかるほど、それなりに精度があると思いますけれども、寸分狂いなくきっちり隅から隅まで分かるということが最終的に必要なではなくて、マネージャであるとか、デザイナがある種の判断をするときに、その結論を間違えない程度に精度が出ていればよろしいわけです。逆にその程度に精度を押えないとコスト倒れになるという特徴があるのでないかと考えています。

それでは、性能予測技法というものにどういうものがあるかといいますと、先ほどからいろいろお話を出てきていますが、一つの技法としましては、モンテカルロ・シミュレーション、もう一つは待ち行列ネットワークを使ったモデルがあります。

それぞれ特徴がございまして、ケース・バイ・ケースで特徴をよく踏まえた上で使い分けることが必要ではないかと考えています。

モンテカルロ・シミュレーションは、金と時間を惜しまず、一つの制御方式なら制御方式をきちんと詳細に評価する場合に、現在でも方法としてはこれに頼

らざるをえないと思います。

それから、キャパシティ・プランニングの例のように、程々の精度でいいけれども、いろいろなケースを手早くやりたいというような場合には、この待ち行列ネットワーク・モデルが非常に適していると思います。

その待ち行列ネットワーク・モデルによる性能評価の特徴は3点ほどございまして、一つは、ロバストネスがあるということです。

ロバストネスというのは、モデルの頑強性とでも訳しますか、プロダクトフォームになる待ち行列ネットワークというのは、複雑怪奇な実際のシステムに比べれば、ある意味では非常に簡単で、単純な構造しかしていないわけです。

それで評価してほんとうに合うのかという心配がつきまとわけですけれども、実際にやってみると、絶対値としてその結論が合う、合わないということよりは、あとで述べます入力データの精度にからみますが、なによりも一番いいと思われるは、追随性のよさです。機器構成を変えるとか、あるいは入力データを変えてみると、そのときに当然結論も変わってくるわけですけれども、変化に対する追随性が現場のSEの方がもっている経験的な値や傾向に非常によく合致していることがあると思います。

2番目としましては、ソフトウェア・パッケージを使うことです。従来の待ち行列理論が使いにくい理由は、数学的にむき出しの表現のまま使わなければいけないことです。

これは、一般的のSEの方にとってみると、かなり苦労しなければならないことになります。それをまず理解し使いこなせる程度にトレーニングしなければなりません。

ところが、待ち行列ネットワークというものをソフトウェア・パッケージとしてインプリメントしますと別に数学的な詳細について理解していなくても、成果だけを手軽に利用できるという形になり、一種の大衆性がでてくるわけです。そういうことがもてはやされた一つの原因になっているのではないかと考えています。

3番目に経済性ですけれども、モンテカルロ・シミュレーションが、計算機の実行時間が非常にかかるということは、よく皆さんご存じだらうと思います。一方、解析型であるために、待ち行列ネットワークの計算時間はモンテカルロ型シミュレーションから比べれ

ば、桁違いに短くてすむわけです。

それから、モンテカルロ・シミュレーションでモデルをつくり評価するということは、実際にはプログラムを書く操作といいますか、モデルをコーディングするという作業が必要となります。一方、待ち行列ネットワークにおけるモデルの作成ではパラメータを入れる、つまり、どのようなシステム構成をしているか、どのような負荷がかかるか、それだけをデータとして入れればよく、モデルをつくるということがプログラムすることとは異なってきますので、その意味でモデルの作成工数が大幅に削減できます。

この3点が待ち行列ネットワークによる解析の大きな特徴ではないかと思います。

現在知られているソフトウェア・パッケージを目につくものだけ拾ってみました。忘れてはならないのは、IBMさんがおつくりになった QNET 4 です。これが多分一番最初のパッケージではないかと思います。それが RESQ という形になりました。BGS 社の BEST/1 をはじめ、たくさんのパッケージがインプレメントされております。

国内では、電電公社さんがおつくりになった QSEC それから私どもの QM-X とか PERFORMS、航技研の QNMAP というようなものが知られています。

このパッケージのベースになっております待ち行列ネットワークを簡単に紹介させていただきます。

ベースになりますのは、BCMP 型とよばれているものです。これは著者4名の頭文字をとって、現在 BCMP 型ネットワークと呼びならわされております。システムを待ち行列のネットワークとしてモデル化します。

そこには、クローズド・チェーンとオープン・チェーンというのがございまして、クローズド・チェーンは、ある決められた人数、これはたとえば、マルチ・プログラミングの多重度に相当するわけですが、ネットワーク内をめぐり歩くわけです。

オープン・チェーンというのは、外からジョブが到着し、リソースを使って外へまた退去していくものです。両者が混在してよろしいというのが、おおまかにいいますと BCMP 型のネットワークの特徴です。

バッチ・システム、リアルタイム・システム、TSS、などのサブシステムが混在するマルチ・ディメンジョンナルな処理をするシステムの性能評価に非常にじみやすい形をしているという理由で、現在のソフトウェア・パッケージはほとんど BCMP 型待ち行列網を

ベースにつくられています。

待ち行列網が実用化されるためには、計算アルゴリズムが必要ですが、現在知られているアルゴリズムは、大きく分けますと二つの系統に分かれまして、一つは Convolution Method、もう一つは MVA (Mean Value Analysis) とよばれている系統のものです。

最初に発表されましたのは、コンボルーション法で著者の Kobayashi というのは司会をされている小林先生ですが、このコンボルーション法によって初めて BCMP 型の待ち行列網が解け、実用化できるという見通しがたちまして、1975 年以降の待ち行列網・ネットワークの実用化時代の幕を開いたといつてもいいと思います。

細かい話題はいろいろありますが、時間がないので省略いたします。現在はこの大きな二つの系統の計算法がありまして、それらの改良とか、細かい手直しとかということで、たくさんペーパーが書かれております。

そのほかには、プロダクトフォームにならないネットワークが実用上非常に大事ですが、その計算法もいくつかあります。簡単にいいますと、母関数法、これはエグザクトに解く方法ですが、実際にはノード 2 ぐらいの場合しか解けなくて、実用にはあまりなっていないようです。

それから、Neuts のマトリックス・ジオメトリクス法、これは直接数値計算してしまう方法です。

3番目の拡散近似による方法、4番目のデコンポジション法、これはいろいろな呼び方がされているようで、パラメトリック・アナリシスとか、等価流量法であるとか、電気回路のノートンの定理に対応させる呼び方とか、いろいろあります。

キャパシティ・プランニングの話題にもどりますけれども、キャパシティ・プランニングというものを成功させるためには、二つの条件が必要ではないかと考えています。

一つは、予測をするための強力なソフトウェア・パッケージの存在です。これは内容的には待ち行列ネットワークとその計算メカニズムが存在していることです。

一番目に、必要かつ十分な精度をもつ入力情報をそろえること。計算メカニズムだけあってもうまくいかず、正しいデータを投入しないことには、正しい結果は当然得られないわけです。

予測精度というのは、今までの経験からいいます

と、ほとんど入力情報のもっている精度できまってしまうようです。いいかげんな情報を入れれば、もちろんいいかげんな答えしか出ませんし、データをきちんと、できる限り精度よくそろえて入れれば、それなりの結果が出てくるということだと思います。

特に利用者の方からみた問題点というのは、オペレーティングシステムの動作、あるいはそれに伴うオーバ・ヘッドを定量的に見積るというのは非常に困難なことではないかと思います。

たとえば、OS に非常に詳しい知識をもっておられる方なら問題ないかもしれませんけど、あるトランザクション・システムやバッチ・システムが動いたときに、システムディスクへどういう具合に I/O が出るのか、なん回ぐらい出るのかを正しく見積るというのは、案外むずかしいことです。

それから、IOS のカーネル部、ディスパッチャとか、I/O ハンドラとか、そういう部分でどのくらい走行ステップ数があるのか、あるいは DBMS とか、DB/DC の走行ステップ数がどのくらいになるのかというの、利用者から見た場合は、非常に把握が困難なデータです。

しかし、実際にはこれを正しく見積らざることには正しいキャパシティ・プランニングになかなかできにくいということです。

その辺を考えまして、私どもがつくりました PERFORMS というキャパシティ・プランニング支援用のシステムは、そういう OS の持つオーバ・ヘッドを自動的に見積り、モデルそのものを自動作成してしまおうという考え方で作られています。

PERFORMS の中核となるのは先ほどの BCMP 型ネットワークを計算する計算メカニズムの部分で、QM-X と呼ばれているソフトウェア・パッケージです。

そのほか実際には、既存のデータが使える場合がありますので、SMF エディタであるとか、あるいは計算結果を出力する際にいろいろ編集するためのパフォーマンス・レポート・ジェネレータや、各種のユーティリティ、こういうものを統合化いたしまして、キ

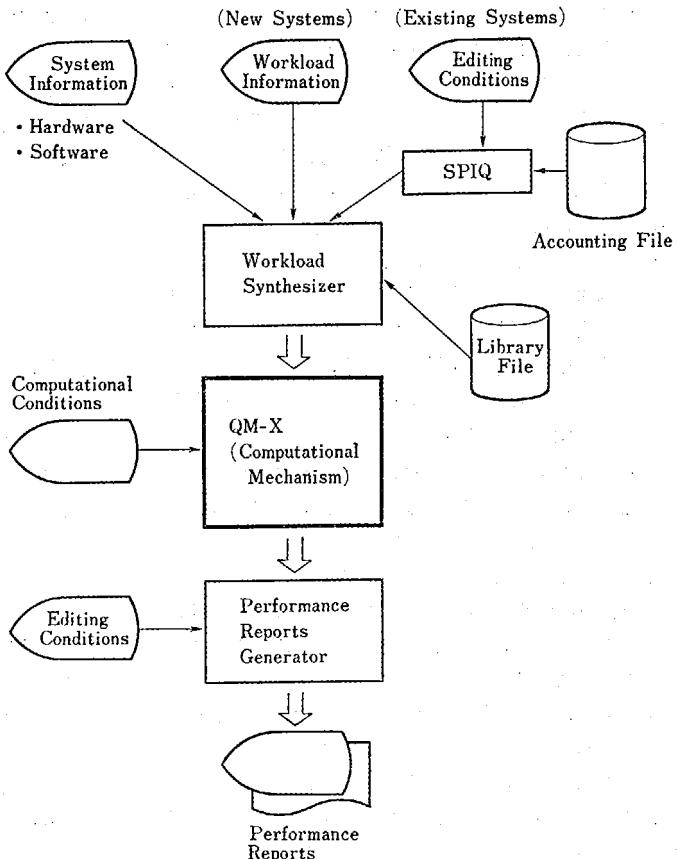


図-4 PERFORMS Configuration

キャパシティ・プランニング支援用のシステムとして PERFORMS を作成しました。

具体的な内容は、この図-4にあります。

中心になりますのは、QM-X というコンピューターショナル・メカニズムです。さらに、ワーカード・シンセサイザというモデルの自動合成機能、データ・ライブラリ・ファイルというオペレーティング・システムの動作の特性やオーバ・ヘッドの数値、あるいはハードウェア、CPU の MIPS 値であるとか、いろいろシステムにからむデータが蓄えられているファイルがあって、これと協力しながら QM-X の入力データをつくるという形になっております。

このワーカード・シンセサイザの考え方としましては、利用者が入れる情報というのは、利用者に見える情報だけですませようというのが、基本的な考え方です。

たとえばトランザクション・システムですと、 SEND/RESEND の回数や、データベースに対する DML

の回数だとか、言語タイプや、ユーザがコーディングした部分だけのダイナミック・ソースステップ数を入力情報とします。

それに対してシステム情報は、PERFORMS 側の供給する情報になります。ハードウェア・コンポーネントの性能であるとか、先ほど申しましたオーバ・ヘッドなどを供給いたしまして、ワーカロード・シンセサイザによって負荷モデルをつくりあげて、QM-Xへの入力情報とするという構造になっております。

最後に、この分野での今後の課題、問題点などについて簡単にまとめてみますと、まず将来的には、全体としての統合化、あるいは体系化が必要になってくるのではないかと考えています。

これはもう少し細かくみると、たとえば理論でいいますと、いろいろな計算法とか、近似法が考えられておりましす、そういうものがある種の試練を経た後に、使えるものは使えるものとして全体が統合化されてくるということが必要ではないかと思います。

技法としましては、測定技法、吉澤さんがお話になりましたように、測定だけでもたくさんの技法がありますけれども、そういうものとか、編集機能とか、場合によっては、シミュレーションをやるために技法であるとか、そういうものが全体の中の一つの役割りをもちらながら体系化されてくるということが必要だと思います。

モデルとしましては、益田先生がお話になりましたような、メモリ管理のモデルであるとか、あるいは通信網、その他いろいろモデルがありますけれども、そういうものの統合化が必要になると考えています。

それから、性能評価とはちょっと離れるのかもしれません、これにコストや信頼性、あるいは運用管理のいろいろな情報、こういうものを総合的にからめて、全体として計算機システムをうまくつくりあげ、それを運用していくための支援システムというものが、将来的に一番必要になるのではないかというふうに考えてています。

しかし、今まであげましたものをただ統合化すればいいということではなくて、個々の技術が成長し、ある段階まで熟していないと、統合化を考えても無意味だと思います。技法を成熟させるための息の長い、基礎的、理論的な研究がぜひとも必要であるし、応用の場での生々しい問題の発掘が必要であり、それらの技術者間の情報交流の促進が将来に向けて必要になってくるのではないかと考えています。

以上です。

司会 どうもありがとうございました。

待ち行列の理論的なものから、実際のインプリメンテーション、それから実際のキャパシティ・プランニングとか、チューニングへの応用と、幅広いお話をありがとうございました。

いま紀氏のプレゼンテーションを伺っていて、ヨークタンハイツの IBM の中央研究所で私が担当していたグループで開発した QNET とか、RESQ というものが刺激となって、日本でもヨーロッパでも、こういう研究が広がっているということを非常に感慨深く思ったのですが、一つ大事なことは、やはり、こういうものが実際の現場にインパクトを与えるためには、理論的なものと、それを実際にインプリメントする、両方の力を兼ね備えた人材として私のグループに M. Reiser という有能な同僚がいたことが、非常に幸いしていると思います。

最後になりますが、電電公社の村岡洋一氏は、システム性能評価技術、特にネットワーク関係のことを中心にお話されると思いますが、よろしくお願ひします。

村岡 ただいまご紹介のありました電電公社の横須賀通研の村岡です。

本日は、先ほど紀さん初め皆さんからご紹介がありましたようないろいろな性能評価技術を使う、ユーザの立場からお話ししたいと思います。

よく私どもの中でも、性能評価の専門の方から実際の現場ではお前は、ちゃんと性能評価の技術を使いこなしていないのではないかと、お叱りをうけますが、今日はそのお叱りに対して、多少自己弁護も含めまして、実際のシステムを開発ないし方式を設計する立場でも、性能評価の最新の技術は努力して使っておりますという例を、多少古い例で恐縮ですが、ご紹介したいと思います。

いま小林さんから、ネットワークというお話がございましたが、ネットワークの話は複雑になりすぎますので、まず情報処理を例に使わせて頂きます。

性能評価のモデルの分類その他につきまして、いま紀さんからご専門の立場からお話がありましたが、私は使っている立場から、どのように考えているかを簡単に申しあげたいと思います。

まず、M/M/1 または M/M/S のモデルですが、これはご存じの通りに、使いこなすのに一番便利なモデルです。ただ、たとえば I/O の割り込みなど、チャネ

ルの性能評価のように、呼に相関関係のある場合の近似解がうまくこの式の拡充に入ると、ありがたいと思っております。

次に、複数サーバを、直列に並べましたモデルというのも、非常に使いがってあるモデルでございまして、これもご存じのように、いろいろ便利な式ができております。

この場合、基本モデルでもそうですが、できればいろいろな呼に優先度をつけてみたり、そういったことが簡単な近似式ができるといいという気がしております。

他にもいろいろとモデルがございますが、省略しまして最後に紀さんや小林さんがおやりになったネットワーク・モデルについてふれさせて頂きます。

先ほどのこのモデルの数値解のためプログラムがいろいろできているということで、私どもの研究所でつくったプログラム（QSEC）もご紹介頂きましたが、設計の初期の段階でいろいろな現象を解析するためには、システムのふるまいをこのプログラムにマッピングしなければならないということで、簡単に使ってみるというわけにいきません。はっきり言いますと、数割のオーダぐらいで間違ってもかまわないのでありますし、精度をあげるときには、ちゃんと時間をかけてやればいいんですが、その前に簡単にやるために、なにか良い近似法がないかと願望しております。

次に、実際のシステム設計で使っている例を述べます、まず交換の例です。

交換はご存じのように、これは待ち行列モデルの発祥の場所ですから、あらためてご紹介する必要は無いかも知れませんが、念のためにいくつか例をご紹介しますと、交換機の設計段階では一般としては、入線無限出線有限のモデル、M/M/1ないしM/M/Sなどのモデルが多く使われています。

最近になりますと、交換機もご存じのように、プログラム・コントロールの交換機になってまいりましたので、ストアド・プログラミング・コントロールの交換機の内部のふるまいを解析するのに、いろいろの解析技術が使われております。

交換機の場合には、ご存じの方も多いと思いますがルックイン方式、いわゆるイベント、たとえば電話に呼が起きたかとか、I/Oがあったとかいう事象を周期的にパトロールしまして、刈りとる方法をとっておりまして、割り込み方式をとっておりません。

したがいまして、周期的なパトロール現象をうまく

解析できるような解析の方法が必要ということで、定期割り込みの優先処理のようなシステム解析ができるモデルを開発しています。

それから、入出力には、ドラムなどを使っていますから、ドラムの待ち行列解析も行います。

これも計算機のI/Oと違いまして、今申しあげましたとおり、割り込みを刈り取るというルックイン方式をやってますので、それとの関係での待ち行列の解析が、一つの課題になっておりました。

次に情報処理の例ですが、方式設計、システム開発の段階で、いわゆる解析モデルを使った例の中で古い例で恐縮ですが、機能分散システムの評価をご紹介したいと思います。

機能分散システムというのは、いまはだいぶ流行も通り過ぎたくらいですけれども、いまから10年ぐらい前は、マイクロプロセッサもそろそろ安くなってくると、専用システムをいっぱい並べてつくるほうが、いわゆる汎用システム、CPUを1台置くよりは、コストパフォーマンスがいいんじゃないかとか、使い勝手がいいんじゃないかという議論がされました。私どもも新しい計算機としては、そういう構成のほうがいいのではないかということで、だいぶ議論をしたことあります。そのとき行った検討の一部です。

もちろん機能分散システムがいいかどうかといいますのは、LSI技術、ハードウェア技術、ソフトウェア技術など、いろいろな技術の総合評価ですから、これらの要因をみて決めなければいけないわけですが、性能評価の技術を使ったのは、その中のコストパフォーマンスの評価と性能の評価です。

機能分散システム、いろいろな形が考えられると思いますが、1例としてFEP（いわゆるフロント・エンド・プロセッサまたは通信制御用のプロセッサ）、システム・コントロール・プロセッサ(SCP)、CPU（いわゆるアプリケーション用のプロセッサ）、それからデータベース・プロセッサ、そしてディスクというような周辺装置の組み合わせという構成を考えました。トランザクションが入ってきますと、そのトランザクションをSCPを見て、適当なCPUにディスパッチする。データベースに対するアクセスが必要であれば、データベース・プロセッサが処理をするという、典型的な機能分散システムだと思います。

こういう機能分散システムで、まず最初にわれわれが問題視しましたのは、プロセッサの間を一つのトランザクションが、なん回かにわたって渡り歩きますか

ら、プロセッサのところに待ちが起こることです。

单一 CPU ですと、あまり問題にならない待ちが、こういう機能分散システムでは、非常に大きなシステムの性能ダウンの原因になるのではないかと予想しました。逆にいいますと、その性能ダウンをリカバーするだけ、個々のプロセッサの性能をあげないと、单一 CPU と比べてコストパフォーマンスが、よくならないのではないかという観点から、このシステムの性能はどうなるんだろうという評価をやったわけです。プロセッサがネットワーク形に構成されたシステムですので、これはいわゆるネットワーク・モデルの使える典型的な例でございまして、実は先ほどの QSEC は、これを解析するためにプログラムを組みました。

どういう性能評価をやったかは、詳細は別としまして、一例を申しますと、单一 CPU に対して機能分散をやるプロセッサの性能は、大体約 5 割ないし 6 割、性能が高くなればいいという結果が得られました。ですから LSI, VLSI 技術なりなんなりを使って、5 割程度は高い性能の専用プロセッサが安くできるのであれば、機能分散システムのほうが有利であるということになるわけで、あとは、もし評価が正しいとすれば、VLSI 技術とノン LSI 設計のシステムとの比較と、それからハードウェアの比較ということになります。

次に、さらに評価の精度をあげて、実際のシステムのふるまいに近づけるために、いわゆるハードウェア資源であるプロセッサと、ソフトウェア資源であるタスクないし制御表というものを両方込みにして、それに対するアクセス競合が評価できるモデルをつくるということで、サーバ同時保留モデルを考えました。トランザクションはハードウェアであるプロセッサを握って、次に、ソフトウェアである制御表またはタスクのようなリソースが握れるかどうかを見るという、この二つの待ち行列を解析できるようなモデルですが、この場合には世の中をながめても適当なモデルがありませんでしたので、新しくモデルを開発しました。概略をご説明しますと次のようなモデルです。アクティブ・サーバと呼ぶプロセッサだけの待ちを、いわゆるネットワーク・モデルでまず評価して、そこでのスループットを今度パッシブ・サーバと呼ぶソフトウェア資源の解析に使うトラフィックにあてまして、それでパッシブ・サーバへの待ちを計算しまして、それをもう 1 回アクティブ・サーバの解析へフィードバックするというような解析をやっています。

その結果ですが、思った通りにソフトウェア資源に対するアクセス競合を考えない場合よりも、考えた場合のほうがスループットが落ちるという結果になりました。当然の話ですが、それが分かりましたので、先ほど申し上げました評価をこれでもう一歩キャリブレーションしまして、それで機能分散システムの実際のコストパフォーマンスがどうかということを評価しました。

大体先ほど紹介しました方法は、シミュレーションとあわせて、それほど間違ってない、いい近似解を与えると考えています。

そういうことで、多少自己弁護にはなりますけれども、システムないし方式を設計する立場でも、ちゃんとモデルないし解析評価技術を使っておりますというご紹介をしたつもりです。

最後に、これからいろいろなことを考えるにあたって、どのような問題があるだろうということで気になることがいくつかございます。

一つは、いわゆるサーバの保留時間ですが、これはよくご存じのように、ポアソン分布その他いろいろあるわけですが、現実、特にネットワークになりますと、これは皆さん電電公社に対して非常に厳しい目で使って頂いてるとみえまして、たとえば、電文などのブロック長はあるブロック長を越しますと通信料が高くなるから、あるブロック長ぎりぎりに使う場合もあり、ブロック長の分布は必ずしも、ポアソン分布にはなりません。

それから、ラインの保留時間もある時間を越すと高くなるから、ある時間ぎりぎりに使いますということで、なんとなく、特にネットワークに関しては、保留時間一定というのが案外多そうだということで、そういういた解説はこれから重要ではないかという気がしております。

次は、バースト現象です。現在これも通信網で同報処理とかいろいろございますが、そうなりますとバースト現象が起ります。

バースト現象が起ったときに、そのバースト現象がどういったような時間軸でもって変化するのか、急に起こって、それがある時間たつとおちつくわけですが、その解析技術というのもこれから必要ではないかと思っています。

ということで、私たちの反省はなんだかんだ申しましても、モデルを作成しない使いこなすための実際のシステムにおける基礎データを十分に揃えてないとい

うことです。先ほどから吉澤さん初め皆さん、トレーサとか、いろんなツールを埋め込んでおかないといけないというお話をございましたけれども、一番の反省は、なにを取りあげても、どういうふうなふるまいをしているかというのが、はっきり言いますと分からぬ場合が大部分でございます。

そういう意味では、もうちょっといろいろそういうデータが十分に、しかも簡単に取れるような仕組みをシステムに埋め込んでおいて、それをだれでもすぐ使っこなせるという形にしておくのが、大事だろうと思います。

以上でございます。

司会 どうもありがとうございました。

パネラの方、非常に盛りだくさんの内容をプレゼンテーションされてますので、予定よりちょっと時間が伸びてまして、自由ディスカッションの時間として、あと10分ちょっとしかないんですが、まずオーディエンスの方からなにかご意見、あるいはご質問ございましたら伺いたいと思います。

どうぞ。

宮崎 東北大の宮崎でございますが、いろいろお話を伺ったんですけども、性能評価をするときの性能というのはなにかということ、どうもんまりよく分からなかつたんですけれども、確かに評価尺度ということはなん人かの方がいわれましたけれども、実際に評価する尺度というのは、どういうふうに考えたらいいか伺いたいと思います。

最後の村岡さんのお話には、スループットという言葉が出てきましたが、実際に私なんかが考えますのは、たとえばCPU効率がよくなれば、性能が上がったとみるのか、レスポンス・タイムが短ければ、それでいいシステムなのか、あるいは最近は、そのシステムの使いやすさということが非常にいろいろいわれていますので、そういう使いやすさに重点をおいたシステムがいいシステムなのか、その辺のところがひとつどういうふうに考えたらいいんだろうかと感じます。

それから、先ほど村松さんのお話の中には、OSをつくるときに階層化するというようなお話がありましたが、確かに階層化しますと、OSの論理構造が明確になって、つくりやすいんだろうと思うんですけども、階層化しますと、逆にいろんな手順がふえますので、性能に影響が出てくるんじゃないかな。そのときの性能というのはなにかと、また問題だと思います。

以上が1点と、それから計測ということに関して、

測定ですか、それに関してやはりなんかの方がおっしゃいましたけども、それからそのシステムをチューナアップする、あるいは拡張していくときに計測は大切だろうと思うんですけども、その計測をするときの負荷をどういうふうに考えたらいいかという問題です。

実際に動いているシステムを、毎日稼働しているシステムを測定するのは、それはそういうモニタがあればやれると思いますけども、たとえばシステム、こういうふうに直したときに、どれぐらい性能が出るだろうかという場合です。

たとえば、私が申しました性能というのは、レスポンス・タイムはどれぐらいよくなるだろうかと、そういうことを考えたときに、負荷というのはなかなか得にくいんではないかということで、そういう測定における負荷をどういうふうに考えたらいいのかと、その2点について、特にどなたということではないですが、伺いたいと思います。

司会 かしこりました。

最初のご質問、尺度としてなにを考えているかというご質問に関して、まず私からお答えしたいと思いますが、最終的には人間、ユーザの生産性までを含めた、いわゆるコスト・エフェクティブなものというものが、最終的なターゲットになるわけですから、いまご質問の方が申されたような使いやすさというものも考えなければいかんわけですけれど、どちらかというと、これは人間工学的なもので、なかなかはっきり定義したり、計れないくらいがあります。

われわれ計算機システム、あるいはネットワーク関係の人たちが定義するパフォーマンスの尺度といいますのは、大きく分けて二つあります、一つはシステム側から見た尺度、すなわちコンピュータ・システムセンターのマネージャなどからみますと、自分のシステムが十分使われているかどうかに关心があります。

ですから先ほど言いましたように、CPUの利用率とか、ジョブをどのぐらい与えられた時間に完了できるかというスループット尺度となります。

ですから、当然、利用率とスループットはかなり関連しています。両者は互いに比例してるような関係です。それに対してユーザの立場からいいますと、オンライン・システムであれば、リアクション・タイム、レスポンス・タイムなどが短いものを、バッチ・システムであれば、ターンアラウンド・タイムが短いものを望ましいシステムとして考えるということになります。

して、ある場合にはシステム側からみた尺度と、ユーザ側からみた尺度というものが、お互いにコンフリクトする場合もあるわけですね。

それから、先ほどからいろいろ出てきました優先順位ですが、いろいろなタイプのジョブがありますと、その重要さ、あるいはチャージする値段に關係して優先順位が変わるということで、その場合にはいろいろなクラスのレスポンスとか、スループットというものを考えなければなりません。

これはほかの分野でもほとんど同じだと思います。

たとえば、通信屋さんでも、信号対雑音比や誤り率というものを数学的評価の尺度としているわけですが、究極的には音を聞いて聞きやすい、あるいは絵を見てはっきり見えるという、そういう人間の感覚まで入れたものが、最終的には品質のよさということになると思うんですけども、そういうものを数量化するのはむずかしい。現在では今申しあげましたスループット、利用率、レスポンス・タイムとか、ターンアラウンド (Turnaround)・タイムというものが、一応だれでも納得できる尺度と考えられています。

2番目の質問のOSの階層化に関しては、吉澤氏にお願いします。

吉澤 オペレーティングシステムの中で負荷をどういうふうに定義するかということなんですが、オペレーティングシステムの開発という立場では、基本的にはシステムの全体的な性能の尺度として、ベーチマークジョブというものを決めておきまして、そして同一の環境下でオペレーティングシステムのバージョン・アップのたびに測定をして、検証をしてみるというやり方をしております。

それから、もうちょっとプリミティブなといいますか、もっとオペレーティングシステムの基本的な性能の評価の場合ですが、特にオンライン・システムですと、性能が非常に厳しく要求されておりますので、頻繁に使用するスーパバイザコール、たとえばI/Oを行う、EXCPのダイナミック・ステップ数などは環境を決めておきまして、こういう環境では何ステップであるか、それがバージョン・アップのときに何ステップになったのか、減ったのか、増えたのか、そういうことを管理していくという方法が、とられて尺度になっていると考えます。

司会 それからもう一つ出ました、メジャメントのためのオーバ・ヘッドをどういうふうに取り扱うかという質問ですけども、どなたか。

紀さんは、そういうモニタなども一緒に組み込んだ仕事をされていますか。

紀 ちょっとご質問を正しく理解しているかどうかわからないんですが、測定のために当然オーバ・ヘッドがございまして、それは評価をするときには一つのジョブに、当然組み入れなければいけないもんだろうと思います。

それから先ほどのご質問で、いろいろOSにチューニングをやると、あるところをいじったことが全体にどう影響するかが、なかなかわかりにくいということではなかったかと思うんですけども、それはまったくその通りで、それをいかに把握するか、いかに正しく規定するかというところが、性能評価のかなり大事な部分になってくるんじゃないかなと思います。

私が先ほどお話をしました待ち行列ネットワークによるモデル化というのも、狙いとしてはその部分にあるわけで、モデル上で負荷を、たとえばチューニングによってある部分が少し変化したとしますと、その変化した分を計測によって把握して、入力データとして変化させてみると、そのことによってアウトプットがどう変化するか、それを見ながらチューニング結果が全体に及ぼす影響を把握していくこうという狙いで利用しております。

そんなことでよろしいでしょうか。

司会 どうもありがとうございました。

ほかになにかご質問、あるいはいまの質問に関連したことでも結構ですが、ご意見ございますか。

あと、パネラの方で、ほかの人のプレゼンテーションに対するコメントとか、補足しておくべきようなものがありましたら、あと残り時間少ないのでお願ひします。

どうぞ。

高田 リンクの高田と申します。

益田先生にお伺いしたいんですが、この特別講演の中で西尾先生が「過度の理想主義は次の時代の災いとなる」という、非常に私どもにとって感銘深いお話をされたんですが、今までいろいろとシステムの問題で皆さん積み上げてこられたと思うんですが、その中で、スループットがどうであろうとか、いろいろ測定のメジャがあると思いますけども、時代はどんどん変わりますので、むしろフレキシビリティとか、発展性とか、そういうことに重点を置くべきではないかというような気もするわけですが、その点、積み上げると同時に、そいつをゼロまで、1まで解析して、それか

らまた別に次の時代の積み上げをしなければならんというふうに考えていますけども、その点益田先生、ちょっと昨日講演伺いました、少しお考があるようになつたんで、お伺いしたいと思います。

司会 どうぞ。

益田 その通りだと思います。

きょうあえて私は、かなり古いところから話をさせて頂きましたが、性能評価の仕事が始まった当初は、非常にハードウェア資源のコストが高かったわけです。そこで、ハードウェア資源を効率よく使用することが、計算機システムのなかで、重要な課題でありました。

そこで、性能評価という一つの大きな研究分野ができて、これまで発展してきたわけですが、現在、ハードウェアの価格が、急速に低下しております。そうしますと性能ということはさておいて、使い勝手のいいシステム、変更に強いシステム、あるいは、信頼性の高いシステム、そういうシステムをどうやってつくっていけばいいかということも、これから非常に重要な問題で、特に性能との兼ね合いを考えながら、そういった新しい問題を考えていかなければならぬと思います。

現在私たちのところでもOSをどうすれば、もっと理解しやすい構造でつくることができるかという研究をしております。理解しやすいとはどういうことか、構造と性能の関係はどうかといったことも興味ある研究テーマです。

司会 どうもありがとうございました。

村松さん、ご意見ありませんか。先ほどお話をされた日本語処理の関係はいかがですか。

村松 会場から階層化の問題について話がありました。その点もあわせて感じることを言いたいと思います。

私どもがOSをつくるのは、一体なんのためにつくるかといいますと、計算機をできるだけ利用しやすくするためにつくるのです。しかし、非常に短い時間、たとえば1分間ぐらいのレベルで考へると、計算機を効率よく使うためには、OSはないほうが性能がいいわけです。機械命令を直接プログラミングしたほうが効率がいいわけです。

それに対してOSというものがあると、1時間とか2時間とかいうレンジでものを見ると、計算機の最終的な出力は、非常に大きくできます。

OSというのは、今までそういう形で、1時間と

かそういうレベルでの考え方でつくられてきたわけですが、最近のようにソフトのコストが全体の計算機システムの中で占める比重を考えると、もうちょっと長い時間のレンジで考えなければいけなくなっています。

たとえば、データベースの階層はどうして出てきたかというと、プログラムとデータとの独立性を考え、数年レンジで考えると、データベース管理という階層をつくったほうが、最終的なトータルコストが安くなるというわけです。

通信関係のネットワーク・アーキテクチャですか、プレゼンテーション・サービスみたいな階層というのも、結局ユーザのプログラム、ユーザの機能追加の要望か、ネットワークの拡張性、そういう数年レンジのプロダクティビティを考えた場合に、やっぱりそのほうが総合的にみて得だという判断で、OSをつくるわけです。

そこの判断の仕方に、あんまり安易さがあつてはいけないわけですから、やっぱり最終的な評価の尺度は、これからは1日だと、1時間のレンジでの計算機の性能ではなくて、1年だと、10年とか、そういうレンジでの計算機の性能が、やはり非常に大きな問題ではなかろうかと思います。

以上です。

司会 どうもありがとうございました。

時間も大分超過しましたので、最後に私ひとこと締めくくりの言葉を話させていただいて、終りたいと思います。

きょうプレゼンテーションして頂きましたのは、システム評価に関するいろいろな立場の実際のご経験に基づいて、幅広く討議して頂いたわけですけども、きょうここでどなたも話されなかったテーマとして、たとえば、ネットワークにおけるフロー・コントロールとか、パスコントロールとかいう、ネットワーク・アーキテクチャに直接結びついたようなパフォーマンスの問題もあります。

それから、私最初に申しましたように、スーパーコンピュータに関しては、パフォーマンスそのものが一番の関心事であります、そういうものをいろいろベクトル型のスーパーコンピュータとか、パイプライン型、アレイ型と、そういうようなものをどのように設計してチューンするかということは、パフォーマンスの技術の応用というのに、直接結びついてるわけです。

さらに、最近話題になっているデータ・フロー・マ

シンのような、従来のコンピュータとは違うアーキテクチャの仕事が、第5世代にからんでいろいろ進んでるわけですけども、そういう新しいコンピュータ・アーキテクチャの立場からのパフォーマンスに関して、まだまだやっと手がつけはじめられたという段階じゃないかと思います。

この辺でも、研究及びその応用が大いになされるべきだと思います。

それからもう一つ、最近話題になりつつありますのが、いわゆる知識工学と結びついた分野、いわゆるエキスパート・システム的な立場から、コンピュータパフォーマンスのモニタとか、チューニングとかを行うことも、これからの方針として、大いに注目すべきで

あると思います。オペレータのアシスタントとして使えるようなエキスパート・システムの開発に関する仕事も、IBM やそのほかのところで、かなり進んでおります。

それから、先ほどから申しましたように、益田先生もおっしゃっていますが、いわゆるハードウェアが非常に安くなっているということですが、VLSI のチップを駆使してマイクロ・メインフレームというような新しいコンピュータに関しましても、その性能評価ということに関して、大いにこれからやらるべきことは残っていると思います。

それではどうも長いこと、パネラの皆様、それから聴衆の皆様、ありがとうございました。(拍手)