オペーレーティング・システム 54－8
（１９９２．３．１３）

ベンチマークによる階層化システムのプラットフォーム性能評価手法

二神 新　山本 和史　大久保 利一　高橋 竜男

ＮＴＴ交換システム研究所

ソフトウェアの流通性や保守性を向上することを目的に，各種システムにおいてシステムの階層化（プラットフォーム化）が進められている．大規模ソフトを抱える交換通信分野においてもプラットフォーム化は重要な課題である．本文では，通信処理用リアルタイムシステムにおいて，各階層のインタフェース仕様に準拠した複数のプラットフォームの性能をベンチマークプログラムを使用して定量的に比較評価する技術について提案する．本評価では，実行時間をベースにシミュレーション等を組み合わせてプラットフォームの性能評価を実施するため，異なるアーキテクチャを採用したプラットフォーム間の性能を公正かつ簡易に評価可能である．

A PERFORMANCE EVALUATION METHOD ON HIERARCHICAL PLATFORM SYSTEM USING BENCHMARK PROGRAMS

Shin Futagami　Kazufumi Yamamoto　Toshikazu Okubo　Tatsuo Takahashi

NTT Communication Switching Laboratories

For the purpose of improving portability and the maintainability of software, the hierarchy structure(platform) is adopted in various systems. This idea of platform is an important also in the switching communication system which consists of a large-scale software. In this paper, the performance evaluation method using benchmark programs is proposed to compare several platforms which satisfy same hierarchical interface specification in the real time communication processing system. This method uses benchmark programs which measure performance based on processing time, and revise performance using simulation such as cache/TLB hit ratio, so it can evaluate simply and fairly the performance of platforms which employed a different architecture .

1．まえがき

近年，ソフトウェアの流通性や保守性を向上することを目的に，各種システムにおいてシステムの階層化（プラットフォーム化）が進められている．プラットフォームはシステム個別のアプリケーションソフトウェアを実行するための共通基盤となる．システム開発者は複数ベンダが提供するプラットフォームの中から性能，コスト及び信頼性等の各システムの要求条件を満足するものを選択することができる．このプラットフォームの考え方はＵＮＩＸ等の情報処理分野だけでなく，大規模ソフトを抱える交換通信分野においても重要である．

本文では，通信処理用リアルタイムシステムにおいて，各階層のインタフェース仕様に準拠した複数のプラットフォームの性能をベンチマークプログラムを使用して定量的に比較評価する技術について提案する．

2．評価モデル

通信処理用リアルタイムシステムを階層化する場合，上位ソフトウェアの流通性，拡張性等を考慮してモデル化する必要がある．本文では，アプリケーションへの共通インタフェースを提供するファイル管理等からなるＳ２インタフェースとハードウェアのアーキテクチャ差を隠蔽するＳ１インタフェースからなる階層化モデルを仮定する（図１参照）．[1] Ｓ１インタフェースはプロセッサ資源を隠蔽するカーネル部と入出力装置を隠蔽する入出力制御部からなる．また，コンパイラ等の開発環境はプラットフォーム及びアプリケーションの性能を大きく左右するため，プラットフォームの評価モデルの対象に含める必要がある．

3．プラットフォームに対する要求条件

通信処理用リアルタイムシステムの性能面から見た特徴を表１に整理する．このため，アプリケーションの走行環境を提供するプラットフォームでは以下のような特徴が要求される．[1]

（１）リアルタイム性

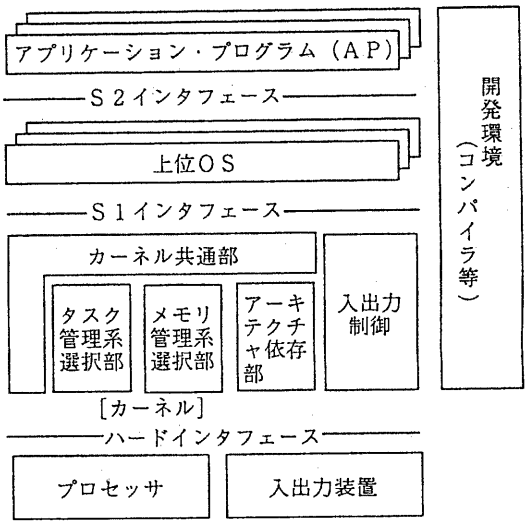

図１　通信処理用リアルタイムシステムの階層化モデル

表１　通信処理用リアルタイムシステムの特徴

機能	概要
超多重処理	一般に通信処理システムでは数千から数万のタスクを同時に処理する必要がある．このため，タスク環境，タスク間通信等の実現方法が全体の性能に大きな影響を与える．
リアルタイム処理	通信処理システムでは接続遅延等の通信品質を満足するため，規定時間内に処理を完了するリアルタイム性が必要となる．このため，タスクスケジューリング手法等が性能に影響を与える．具体的には，クロック割込，プリエンプション機能を利用したスケジューリング手法を採用する場合が多い．また，プロトコル処理上の正常性チェック等のために多種多数のタイマを使用する．
入出力処理	通信処理ノードでは多種多様な入出力装置（通信装置）をシステムに組み込み，高頻度でこれらの装置をアクセスする．一般に，入出力装置の監視オーバヘッドを削減するため，割り込み処理の代わりに周期的なルックイン処理を採用することが多い．このため，組み込む装置数によって周期処理のオーバヘッドが変化する．

走行タスク数等の実行環境に依存しないで一定時間内に処理要求を完了するリアルタイム性を保証する．

（2）高スループット性

周期処理に伴うプラットフォームの固定オーバヘッドを削減しながら，アプリケーションからの処理要求を効率良く処理する高スループット性を保証する．

4．ベンチマークによる性能評価

4．1　性能指標

プラットフォームの性能評価では，アプリケーションを動作させたときの動的特性を推定することが重要である．アプリケーション走行時のシステム性能を規定する各種要因を表2に整理する．特に，プロセッサの命令セット（ＣＩＳＣ／ＲＩＳＣ等）が異なる場合，従来のダイナミックステップ数による性能評価手法は使用できない．このため，アーキテクチャの異なる複数のプラットフォームを横並びで評価する場合には実行時間をベースにした評価手法が不可欠である．

4．2　ベンチマーク評価の特徴

ベンチマークプログラムはアーキテクチャの異なる複数のシステム間の性能を簡単に比較する手段として利用されている．現在，よく利用されている整数演算系のベンチマークプログラム及びその特徴を表3に整理する．[2]，[3] これらのベンチマークプログラムを利用した評価方法は以下のような特徴を持つ．アプリケーション走行時のプラットフォーム性能を定量的に推定するためには，これらのベンチマークプログラムの特性を生かした評価手法の確立が重要である．

（1）利点

・実行時間をベースとした性能評価が可能

・複数のプラットフォームへの移植が容易

（2）欠点

・実際のアプリケーション動作を擬似できないケースが多い．具体的には，キャッシュ／ＴＬＢヒット率や命令の使用頻度等はアプリケーションのソフトウェア特性に依存するため，ベンチマークのみではシステム性能の正確な評価ができない．特に，キャッシュ容量が異なるプラットフォームを比較する場合，ベンチマークのワーキングセットとキャッシュ容量の関係により，ベンチマーク性能とシステム性能が逆転する可能性がある．

・ベンチマークとアプリケーションの記述言語が異なる場合，コンパイラ等の開発環境の性能差を正確に評価できない．また，コンパイラの最適化性

表2　システム性能に対する性能規定要因

項目	概要
ハードウェア性能	ＭＰＵ命令実行時間，メモリアクセス時間，入出力装置等のハードウェア単体性能等
ハードウェア特性	命令セット（ＣＩＳＣ／ＲＩＳＣ等），キャッシュ／ＴＬＢの容量及び構成等
ソフトウェア特性	メモリワーキングセット，命令使用分布，ダイナミックステップ数等
コンパイラ性能	命令展開率，メモリアクセス頻度，最適化効果等

表3　各種ベンチマークプログラムの比較

項目	ＥＤＮベンチマーク	ドライストーン	ＳＰＥＣｉｎｔ
使用言語	アセンブラ，Ｃ等	Ｃ，Ｐａｓｃａｌ	Ｃ
評価対象	ハードウェア	ハードウェア，基本ＯＳ相当（ＵＮＩＸ）	基本ＯＳ相当（ＵＮＩＸ）
マルチタスク対応	×	×	△
リアルタイム対応	△	×	×
入出力動作対応	×	×	△
ワーキングセット	３００Ｂ以下	１ＫＢ以下	８ＫＢ以下

能により，実行時間が大きく変化する場合には実際のアプリケーションで使用可能な最適化レベルに合わせて評価が必要である．

5．評価手法

5．1　プラットフォームの総合性能

アプリケーション走行時のプラットフォーム性能は，以下のような性能規定要因を評価することにより推定可能である．具体的には，アプリケーションで利用可能なプロセッサリソースは，全プロセッサリソースからプラットフォームの周期処理に伴う固定分オーバヘッドとリアルタイム性を保証するためのマージンを差し引いた部分となる．アプリケーションの走行ステップはプラットフォームの提供するサービス（システムコール）部とアプリケーション固有の処理部から構成され，システムコール部はプラットフォーム固有の性能に直接左右される．また，アプリケーション固有の処理部は，アプリケーションの平均命令実行時間とダイナミックステップ数により実行時間が決定され，コンパイラ等の開発環境及びアプリケーションのソフトウェア特性に依存する．これらのパラメータはプラットフォーム毎に異なるため，特定のパラメータの評価だけではプラットフォームの優劣は判断できない（図2参照）．

（1）プラットフォームの固定分オーバヘッド

プラットフォームで実行する各種の周期処理に伴うオーバヘッド．具体的には，クロック割り込みを契機とした周期的なスケジューリング及び入出力装置に対する周期的なルックイン処理が対応する．

（2）リアルタイム特性の保証マージン

プラットフォームのリアルタイム特性を確保するため，プロセッサ使用率の上限（マージン）を保証する．

（3）プラットフォーム依存部

プラットフォームが提供する各種サービス（システムコール）の処理量．本項目は，システムコールの発行回数とシステムコールの実行時間に依存する．

（4）アプリケーション固有部

アプリケーション固有のサービス内容に依存し，アプリケーションの平均命令実行時間とダイナミックステップ数に依存する．

5．2　プラットフォームの固定分評価

プラットフォームの固定分オーバヘッドを大きく分類すると以下のようになる．

（1）リアルタイム特性を保証するため，クロック割込契機による周期的なタスクスケジューリングを実施するためのオーバヘッド

（2）入出力装置からのイベントを効率良く制御するため（群処理），割込方式の代わりに周期的なルックイン処理を使用するためのオーバヘッド

前者はカーネル部の処理に対応し，後者は入出力制御部の処理に対応する．一般に，入出力装置はシステム毎に組み込む装置及び装置数が変化するため，後者はシステム毎に処理量が変化する．特に，通信処理用リアルタイムでは，多種多数の入出力装置を使用するため，後者のオーバヘッドを無視できない．以下に，本項目に関するベンチマークを利用した評価手法の特徴を説明する．

[評価手法]

（1）クロック割込ありのケースとなしのケースのベンチマーク性能を測定し，その相対性能差を評価することにより固定分のオーバヘッド量（比率）を算出する．

（2）システム毎に組み込む入出力装置と固定分オーバヘッドの関係を明確にするため，組み込む入出力装置をユーザで選択可能とし，入出力装置とオーバヘッド量の関係を明確にする．

（3）キャッシュメモリを利用したプラットフォー

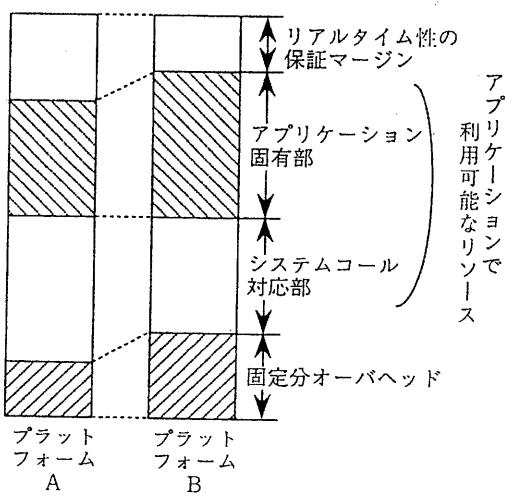

図2　プラットフォームの総合性能

ムにおける固定分のオーバヘッド量を明確にするため，キャッシュオン（全ヒット条件に近い環境）／オフ（全ミスヒット条件）の条件で上記（1），(2）項のデータを算出し，図3のようなアプリケーション走行時の固定分オーバヘッドの影響範囲を明確にする．最終的には，実システムにおけるキャッシュのヒット率の予想範囲をシミュレーション等で推定し，図3のグラフから具体的なオーバヘッド量を算出する．

（4）周期処理の起動周期は短いものは8ｍｓ程度から長いものでは数秒の単位となるため，相対性能を評価する基準ベンチマークとしては実行時間が自由に変更可能なこと及び上記（3）項のキャッシュメモリの効果を評価するためにベンチマーク全体がキャッシュメモリに格納可能なことを考慮して，本評価では基準ベンチマークプログラムとしてドライストーンを採用した．

5．3　システムコールの静的評価

プラットフォームで提供するサービス（システムコール）単位の理想状態における性能を評価する．本評価では，システムコール実行時間の下限を評価する．[1] 本評価で使用するベンチマークプログラムでは，キャッシュ／ＴＬＢが全ヒットとなる条件及び測定対象以外の負荷条件を制限することにより理想状態を擬似し，システムコールで規定される各種パラメータ条件での実行時間を評価する．以下に，本項目に関するベンチマークを利用した評価手法の特徴を説明する．

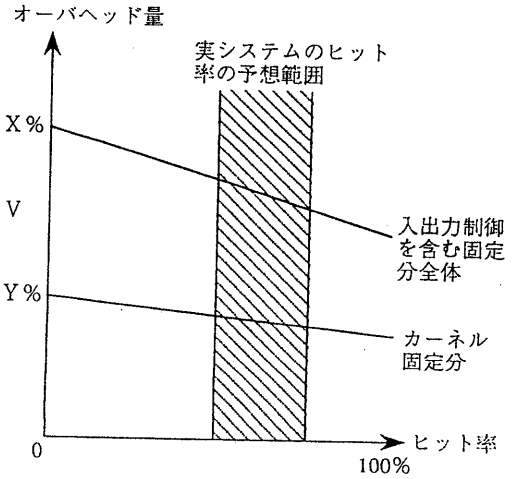

図3　キャッシュのヒット率と固定分オーバヘッドの関係

［評価手法］

（1）システムコールの実行時間を測定する前に該当システムコールを仮実行し，該当システムコールの命令列及びオペランドデータを可能な限りキャッシュメモリに事前に格納し，キャッシュ／ＴＬＢミスヒットの影響を極力小さくする．一般にシステムコール単体のワーキングセットはあまり大きくないため，最近のキャッシュメモリ／ＴＬＢの容量を考慮すればキャッシュ／ＴＬＢ全ヒットの仮定をほぼ満足できる．

（2）システムコールの対象範囲としてはカーネル部だけでなく入出力装置対応のシステムコールを含める．

（3）測定対象システムコールを規定回数繰り返し実行し，その平均値を静的な実行時間として使用することによりタイマ誤差の影響を排除する．

5．4　システムコールの動的評価

本評価では，各種の負荷条件を換えてシステムコールの動的な性能を評価する．一般に，ＯＳ内部処理等では待ちキューを使用するため，負荷条件によって実行時間が大きく変化する可能性が高い．本評価で考慮する負荷条件を以下に整理する．

（1）実行タスク数

（2）タスクの実行レベル，優先度

（3）単位時間当りのシステムコール発行回数

（4）プロセッサ使用率

本評価の目的は，システムコール実行時間の動的な特性を評価することであるため，ベンチマーク評価においてキャッシュ／ＴＬＢ全ヒット等の理想状態を擬似する必要はない．アプリケーション走行時とベンチマーク走行時の実行環境（キャッシュ／ＴＬＢのミスヒット率等）との整合については別途考慮する必要がある．以下に，本項目に関するベンチマークを利用した評価手法の特徴を説明する．

［評価手法］

ベンチマークの負荷条件は単位時間に発生するタスク数／システムコール回数を変化させることにより実現する．本ベンチマークでは，各システムコールの実行時間分布（最大／最小／平均）及びプロセッサ使用率を実測し，負荷条件と応答時間の関係を評価する．特に，タスクの実行レベル／優先度を可変化することにより，プリエンプション機能を含む

タスクスケジューリング機能のリアルタイム性能及びスループットの関係を評価する．これにより，負荷特性に応じたプラットフォームの内部アルゴリズムを含めた性能特性を評価できる（図４参照）．

５．５　平均命令実行時間

システム性能評価では，アプリケーションを走行させたときの平均命令実行時間が有効な性能指標となる．平均命令実行時間は走行ステップ数を掛けることにより実行時間に換算できる．このため，ＲＩＳＣ／ＣＩＳＣのように命令セットが異なるプラットフォームを評価する場合，平均命令実行時間と走行ステップ数を組み合わせた評価が必要である．一般に，平均命令実行時間（Ｅ）は以下のような近似式で表現できる．

Ｅ＝Ｆ０＋Ｆ１＋Ｆ２＋Ｆ３

Ｆ０：理想状態の平均命令実行時間

Ｆ１：キャッシュミスヒットに伴うロス時間

Ｆ２：ＴＬＢミスヒットに伴うロス時間

Ｆ３：各種ハザードに伴うロス時間

上記項目はプラットフォームのハードウェア構成，ソフトウェア特性等に依存するため，プラットフォーム毎に評価する必要がある．各項目の性能規定要因及び評価方法を表４に整理する．通常，ベンチマークでは全項目を含んだ形で実行時間が計測されるが，ベンチマークの走行環境とアプリケーションの走行環境は一致しないため，システム性能を正確に推定するためにはＦ１（キャッシュ関連）／Ｆ２（ＴＬＢ関連）の項目について分離して評価するのことが望ましい．具体的には，ベンチマークを利用して以下のような手順で平均命令実行時間を評価する．

［評価手法］

（１）キャッシュ／ＴＬＢ全ヒットの条件で走行可能でかつアプリケーションとソフトウェア特性が類似したベンチマークを利用してＦ０＋Ｆ１を算出する．具体的には，キャッシュメモリに格納可能なベンチマークをプラットフォーム上で実行し，その実行時間を走行ステップ数で割ることにより平均命令実行時間（Ｆ０＋Ｆ１）を得る．例えば，整数演算系のアプリケーションの場合にはドライストーンやシステムコール静的評価用のベンチマークが利用できる．但し，平均命令実行時間（Ｆ０＋Ｆ３）の値はアプリケーションを記述する言語及びそのコンパイラ性能等によって大きく左右されるため，以下のようなコンパイラ等の開発環境に注意する必要がある．

①ベンチマークの記述言語とプラットフォームの記述言語及び使用コンパイラの整合をとる．

②コンパイラの最適化機能として，実システムで使用可能な最適化レベルを使用する．

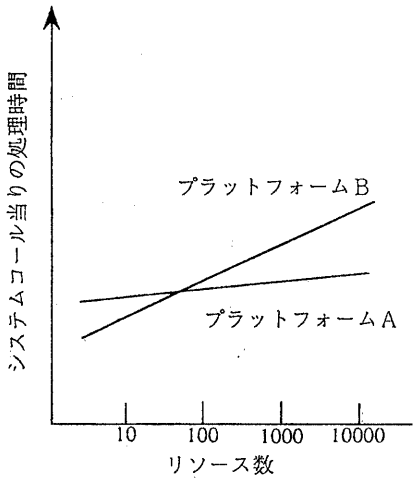

図４　システムコールの処理時間とリソース数の関係

表４　平均命令実行時間の規定要因

項目	規定要因	評価方法
Ｆ０	命令ＭＩＸ，メモリアクセス頻度，命令実行時間	ベンチマーク
Ｆ１	メモリアクセス頻度，メモリアクセスパターン，キャッシュ容量／構成，ミスヒット時のペナルティ時間	机上評価＋シミュレーション
Ｆ２	メモリアクセスパターン（命令当りのアクセス回数／データ長及び局所性等），ＴＬＢ容量／構成，ミスヒット時のペナルティ時間	机上評価＋シミュレーション
Ｆ３	ハザード（ジャンプ命令，レジスタ衝突）の発生確率，ハザード時のペナルティ時間	ベンチマーク

（2）アプリケーションのメモリアクセスパターン等のソフトウェア特性に依存するＦ１／Ｆ２の項はベンチマークのみを使用して推定することは困難である．このため，既存システムのトレースデータ等をベースにシミュレータにより各種キャッシュ／ＴＬＢ構成下でのヒット率を推定し，本推定値とミスヒット時のペナルティ時間（ハードウェア特性）をベースにＦ１／Ｆ２の値を机上評価する．Ｆ１／Ｆ２の項はヒット率の変動等により影響されるため，システム性能を推定する際にはヒット率の変動範囲を考慮した評価が必要となる．また，ハードウェア特性によってプラットフォーム毎にキャッシュ／ＴＬＢミスヒット時のペナルティ時間が異なるため，図５のようにヒット率が変化することにより，プラットフォーム性能が逆転するケースも発生する．

６．まとめ

本文では，ベンチマークを利用したプラットフォームの評価手法について提案した．ベンチマークは対象システムの性能を定量的かつ簡易に測定する手段として有効であるが，実際のアプリケーションの動作環境を完全には疑似できないため，プラットフォーム評価ではベンチマーク毎の特性を考慮した評価手法が必要である．本文で提案したベンチマークを使用したプラットフォームの評価手法では，実行時間をベースにシミュレーション等を組み合わせて性能評価するため，異なるアーキテクチャを採用したプラットフォーム間の性能を公正に評価可能である．

今後，通信処理用リアルタイムシステムのプラットフォーム性能の定量的な評価を実施するため，本文で提案した各種ベンチマークプログラム及びトレースデータ等を収集／解析する評価ツールについて検討を進める予定である．[5]

[謝辞]

本検討に際して，有益な助言を頂いた交換システム研究所第二プロジェクトチームの各位に感謝します．

[参考文献]

（1）星合他，”通信用プロセッサ向けカーネルの実現方式”，ＮＴＴ Ｒ&Ｄ，ｖｏｌ．４０，Ｎ０．１２，１９９１．

（2）T.Conte,"Benchmark Characterization", Computer,pp48-56, Jan,1991.

（3）M.Kainaga,et.al."Analysis of SPEC Benchmark Programs",TRON Project,pp208-215,1991.

（4）H.Kurosawa,et. al.,"An Evaluation Method of Kernel Products Based on CTRON",TRON Project 1990,Springer-Verlag, pp191-200.

（5）高橋他，”階層化システム構成におけるプラットフォーム評価手法に関する一検討”，Ｎｏ．Ｂ－５００，春期信学全大，１９９２．