オペレーティング・システム　50－2
（１９９１．３．１５）

TPCモデルによるOLTPシステムの性能評価

小川 直樹，山永 康昌、松澤 智子，大上 貴英，野地 保
三菱電機(株) 情報電子研究所

OLTPシステムのベンチマーク方法と性能モデル化の方法と、これらを両方用いた性能評価例を紹介する。ベンチマークでは、端末エミュレータにより数百の端末の環境を疑似できる。モデルへ入力するパラメータはシステムマクロ別のCPUオーバヘッドとディスクアクセス時間である。モデルの検証のため、応答時間に着目して測定結果とモデルの結果を比較した。モデルの誤差は２０％以内であった。定量的な評価により、ベンチマーク性能を向上するには、ホストのCPU能力の向上に対応して、改善対象を、１）TPモニタのオーバヘッドの削減から、２）ディスクとデータロックの負荷の削減 に移す必要があることを確認した。

Performance Analysis Of OLTP Systems Using TPC Model

Naoki Ogawa, Yasumasa Yamanaga, Tomoko Matsuzawa, TakaHide Ohkami,Tamotsu Noji
Information Systems and Electronics Development Laboratory
Mitsubishi Electric Corp.
325 Kamimachiya, Kamakura-City, Kanagawa 247, Japan

Benchmarking and simulation modeling for OLTP systems and performance analysis of some systems are discussed. Single process emulating hundreds of terminals are introduced to benchmark systems with large scale. The simulation model is a queuing system. The response times between simulation and measured are compared as a function of number of terminals, and the differences are under twenty percent. The result of the analysis is stated that several efforts should be made to increase benchmark throughput: decreasing TP monitor's overhead, the load on the disk system, and the conflict of data locking.

1 はじめに

OLTP（オンライントランザクション処理）は、金融業の自動現金支払システムや鉄道の座席予約システムなどに幅広く用いられている。近年、これらのシステムの利用者の数は、急激に増加している。このためにホストシステムの処理性能も年々向上することが要求されている。この要求に応じるため、われわれは、OLTP システムの性能評価を行なっている。

本論文では、ベンチマークテストとシミュレーションとを用いたOLTP システムの性能評価の例を示す。評価対象システムはビジネスコンピュータであり、評価対象の処理形態は TPC [1]制定の標準ベンチマークモデル TPC-A である。第2節では、ベンチマークテストの方法について述べる。第3節では、OLTP のモデル化の手法について述べる。第4節では、ベンチマーク結果とシミュレーション結果とを報告する。第5節では、OLTP 性能向上のためのシステムの改善策とその効果について考察する。

2 ベンチマークテストの方法

2.1 ベンチマークモデル

ベンチマークモデルには TPC 制定の TPC-A を選んだ。TPC-A は、銀行業務をモデルとしており、1）軽負荷、2）更新処理主体 の OLTP 業務を代表するベンチマークである。TPC-A は業界標準であり、かつ仕様も明確であるため、今回の評価に最適と判断した。

以下のベンチマークテストは TPC-A の規定 [1] [2] に従って行なった。

2.2 システム構成

ベンチマークシステム構成を次ページ図1に示す。

このシステムのポイントは、

1. 応答時間とスループット（tps:transaction per second）の測定を厳密に行なうために、評価システムの外部にドライバシステムを接続し、
2. ドライバシステム内の端末エミュレータ（RTE: Remote Terminal Emulator）により、任意の数（数十〜数百）の実端末を疑似させる

という考えにある。

[1]Transaction Processing Performance Council

RTE により生成されたメッセージは、回線を通り評価対象システムに送られる。このメッセージは、APP により処理される。また APP の応答が再び回線を通り RTE に送られる。以下、ドライバシステム、評価対象システムの順に詳細を述べる。

2.2.1 ドライバシステム

ドライバシステムの機能は、

1. 複数の端末の動作をエミュレートすることにより、ホストシステムにトランザクションを供給することと
2. 応答時間と tps の計算に必要なデータを測定記録する

ことである。これらの機能は RTE により実現される。（次ページ図2）

RTE の特徴は、1プロセスで任意の数の端末の動作をエミュレートできることにある。このため RTE のプロセスは、エミュレートする端末数だけ独立したスタック領域を持っている。（一種のマルチスレッド構成）この方法では、RTE の使用するメモリ量は、端末ごとにエミュレータ・プロセスを設ける方法に比べてはるかに少なくて済む。これにより、始めて数百の端末のエミュレートが可能となった。

2.2.2 評価対象システム

評価対象システムの特徴は以下に述べるとおりである。（図1）

1. 端末からのメッセージは回線を通して評価システムに送られる。
2. 上記メッセージは、メッセージ・マネージャ（以後、MM と略す）により所定のアプリケーション・プログラム（以後、APP と略す）のメッセージ・キューに格納される。
3. アプリケーション・サーバ（以後、AS と略す）は、メッセージを処理するために、メッセージに対応してトランザクションを起動する。一つの APP に対して、複数のメッセージが処理を待っている場合、システム生成時に規定した制限数（多重度）までのトランザクションを並行して実行できる。AS は、APP の多重度を監視する。APP の多重度に空きが生ずると、AS は新たなトランザクションを起動する。
4. 起動されたトランザクションは、APP の実行を開始する。

5. トランザクションが完了すると、APPからの応答は、いったん応答キューに格納される。その後、応答はMMにより所定の端末に返される。

MM、ASはシステムレベルのプロセスから構成され、いわゆるTPモニタの役割を果たしている。本論文でも、これらをTPモニタと呼ぶ。

評価対象システムを構成するソフトウェアは、OS、TPモニタ、APPであり、ハードウェアはCPU、メモリ、チャネル、デバイスである。

2.3 測定項目

2.2.1節で述べたとおり応答時間とtpsはRTEにより測定する。これ以外に、評価対象システムの性能評価のために以下の詳細情報を収集する。

1. 各プログラムのCPU時間（ユーザ・モード、カーネル・モード）
2. SVCのタイプ別発行回数
3. TPモニタ内モジュールのタイプ別実行ステップ数

CPU時間は、モード変遷時にタイマ切替えロジックを入れることによりソフトウェア的に測定した。（ソフトウェアモニタ）SVC発行回数はOSのカウント機能を用いて測定した。実行ステップ数は命令実行の履歴をマイクロプログラムを用いて記録することにより測定した。

ソフトウェアモニタのオーバヘッドは、CPU時間全体の3％以下であることがわかっている。また、OSによるSVCのカウントは、通常時より行なわれているため、このオーバヘッドは問題にならない。このためCPU時間とSVC発行回数の測定は、tps、応答時間測定のベンチマークと同時に行なった。

一方、命令の履歴を収集してベンチマークをおこなうと、CPU時間、経過時間は、これを収集しない場合の約5倍程度となった。従って、tps、応答時間測定のベンチマークとは別に、実行ステップ数だけを測定するベンチマークを行なった。

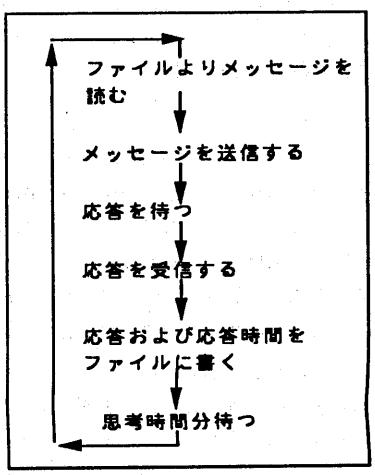


図2．RTEの動作

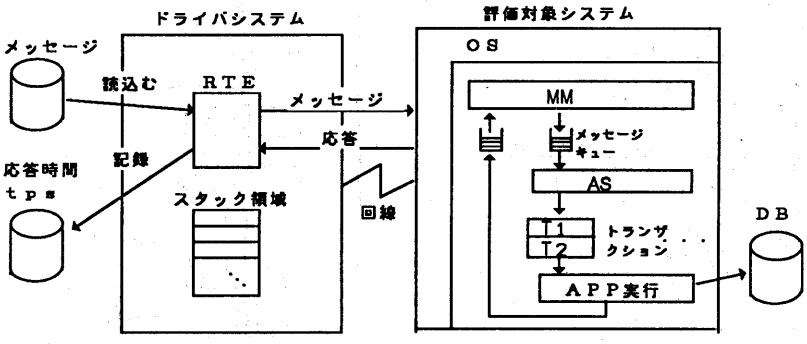


図1．システム構成

3　OLTPのモデル化

1）精度と2）単純さ（必要とするパラメータの数）とのトレードオフを考えてモデル化をおこなった。以下、モデル化の手法について述べる。

3.1　待ち行列モデル

まず本システム全体をハードウェア、ソフトウェア要素からなる待ち行列モデルとして表した。[2]（図3、図4）

モデルでは、

1. AS、MMの処理では、CPUが使用され、
2. APPの処理（図4）では、CPU、デバイス、チャネルが使用され、またレコード等の排他制御が行なわれる

と仮定した。

また、モデルでは、APP処理をTPモニタおよびファイルシステムのマクロ単位に分類した。例えば、これらのマクロには、キューからのメッセージの取りだし、キューへの応答の格納、ファイルのアクセス、ジャーナルの採取などがある。従って、図4に示すAPP処理の待ち行列は、これらのマクロ単位に繰り返し評価される。

さらに、モデルでは評価システムのバッファキャッシュの影響を統計的に推測した。（詳細は3.2節を参照）このため、図4に示す通り、APPがファイルのアクセスする際、キャッシュ・ヒットの有無の条件で処理を場合分けをした。

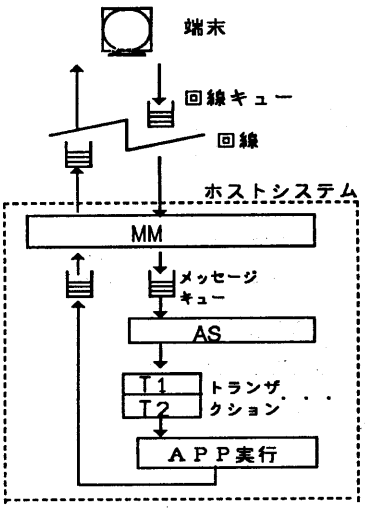


図3．システムの待ち行列モデル

3.2　キャッシュ・ヒット率の予測

バッファキャッシュのヒット率は以下の方法により予測した。

1. TPC-Aのファイル仕様（表1）をもとに、ファイルサイズ、索引サイズを計算する。（表2）
2. これらのサイズとキャッシュのサイズを比較し、ファイルおよび索引ごとのヒットの有無を統計的に予測する。

表1．TPC-Aのファイル仕様

ファイル	レコード長	レコード件数/tps	構成	アクセス方式
支店	100B	10	索引	ランダム更新
窓口	100B	100	索引	ランダム更新
口座	100B	100,000	索引	ランダム更新
履歴	50B	8時間分	順次	順次追加

表2．ファイルサイズ，索引サイズ[2]

ファイル	ファイルサイズ	索引レベル	索引サイズ（ルート，レベル1，レベル0）
支店ファイル	10KB	2	(2KB, -, 2KB)
窓口ファイル	100KB	2	(2KB, -, 12KB)
口座ファイル	100MB	3	(2KB, 62KB, 11MB)

[2]仮に，10tpsのシステムに対するデータベース容量をしめす．

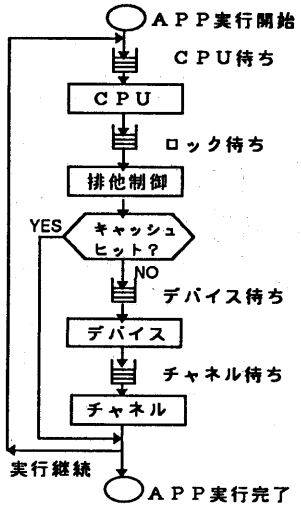


図4．APP処理の待ち行列モデル

[2]説明のため実際のモデルを簡略化したものを示す。

3.3 排他制御のモデル

本システムは、2相ロックを用いたレコードの排他制御を用いている。2相ロックでは、APPのアクセスするレコードは、トランザクション内でリード直前からコミット直後までロックされる。

また、本システムでは、上記レコード単位のロックとは別に索引ファイル、順次ファイルに対して、APPからレコード追加処理の要求が出されると、別の種類のロックが使用される。

このロックは、これらのファイルが保有する制御情報に対して適用される。APPからレコード追加処理要求が発せられると、ファイルシステムは、

1. ファイル制御情報をロックし、
2. ファイル制御情報よりファイル内の最終レコード位置へのポインタ読みだし、
3. この位置を新規レコードを追加する位置と決めて、
4. このポインタ値を更新したのち、ファイル制御情報のロックを解除し、
5. コミット時に所定の位置に新規レコードを書き込む。

評価システムでは、TPC-Aの各ファイル（表1）には上記レコードロックが適用され、履歴ファイルには上記ファイル制御情報のロックが適用される。

以上のロックの特徴をモデルに採り入れた。

3.4 入力パラメータ

モデルへ入力するパラメータは

1. CPUオーバヘッド
2. ディスクアクセス時間

である。

モデルでは、CPUオーバヘッドをTPモニタおよびファイルシステムのマクロ単位に細分化して入力する必要がある。

4節で示すシミュレーションではこれらCPUオーバヘッドにはベンチマークより得られた既知の値を用い、ディスクアクセス時間には公称値を用いた。

4 結果

本節では、いくつかの計算機（表3）を例に、そのベンチマークテスト結果とシミュレーション結果とを示す。

4.1 ベンチマーク結果

ここでは、計算機Aのベンチマークテスト結果を示す。

ベンチマークテストの結果、計算機Aは、CPUネックであることが判明した。この原因を詳しく調べるために、CPU時間の内訳、SVCタイプ別発行回数、TPモニタ内モジュールのタイプ別実行ステップ数を測定した。

(1) CPU時間の内訳（表4）

測定結果より、APPのCPU時間の比率（42％）とTPモニタ（MM、AS）のCPU時間の比率（46％）とが同程度であることが判明した。従って、APP、TPモニタ両方のオーバヘッドの削減に取り組む必要がある。

表3．評価対象の計算機

	CPU性能[1]	I/O構成
計算機A	0.3	条件は，同一である． ユーザファイル ---ディスク1台づつ（合計4台） ジャーナル---ディスク1台 チャネル，コント---1台づつ
計算機B	1	
計算機C	2	

[1] 相対性能で表す．

表4．ＣＰＵ時間内訳

	ＡＳ	ＭＭ	ＡＰＰ	その他
ＣＰＵ時間比率	14％	32%	42%	12%

しかし、われわれは、TP モニタのオーバヘッド削減に優先的に取り組むことにした。この理由は、以下のとおりである。

1. APP のオーバヘッドの大部分は、I/O 処理 (read/write)であると予測される。I/O 処理のオーバヘッド削減のためには、カーネルの改修が必要である。しかし、カーネルの改修は、システム全体に与える影響が大きいため避けたい。

2. TP モニタのオーバヘッドの比率が予想以上に大きいことから推測すると、TP モニタには改善の余地が大きいと思われる。また、TP モニタは、カーネル外部にあるので、その改修は容易である。

(2) SVC のタイプ別発行回数

一般に SVC 発行時の割り込み処理のオーバヘッドは大きいと思われる。従って、SVC をユーザ・ロジックで置換すればシステム全体のオーバヘッドが削減されると考えた。そこで、本ベンチマーク実行中に発行される SVC のタイプと発行回数を調べた。

その結果、以下の SVC の発行回数の和が、SVC の発行回数の合計の約半数を占めることが判明した。また、これらはいずれも TP モニタから発行されることも分かった。そこで、TP モニタを改修して、これらの SVC をユーザ・ロジック化することを検討した。

1. Unix の signal 相当の SVC :
MM-AS,AS-APP,MM-APPのプロセス間同期通信に使われる。

2. Unix の time 相当の SVC : TP モニタは、メッセージをキューインする時刻やトランザクションを起動した時刻をログ情報として採集している。TP モニタが時刻を得る手段としてこの SVC が使われる。

(3) TP モニタ内モジュールのタイプ別実行ステップ数

測定の結果、全 CPU 時間の 13 %が以下のモジュールの処理に費やされることが判明した。そこで、これらのモジュールを改修して、実行ステップ数を減らすことを検討した。

1. TP リソースのテーブルサーチ :
TP モニタは、端末やトランザクションごとに作業領域を持っている。TP モニタはこれらの作業領域をアクセスするために、名前 (端末名、トランザクション名) と作業領域へのポインタとの関係をテーブルの形で保持している。現在の実装では、所定の名前に対応するエントリを探すのに、このテーブルをリニアサーチしている。従って、名前の数を n とすると、サーチ時間は $O(n)$ に達する。

2. GMT (整数型) から西暦 (文字型) への変換
先に述べた Unix の time 相当の SVC は、1970 年 1 月 1 日の真夜中からの秒単位の経過時間 (GMT) を整数で返す。TP モニタではこれを西暦 (文字型) に変換した後、ログ出力している。

(1),(2),(3) に示した結果より、CPU ネックを解消するためには、TP モニタのオーバヘッドを削減することが有効であると判断した。このための改善策とその効果は、5.1 節に記述する。

4.2 シミュレーション結果

まず、仮定したモデルの検証のため、計算機 A を例に、シミュレーション結果とベンチマークテスト結果とを比較した。(図 5) その結果、応答時間において誤差 20 %以内の一致が得られた。

これにより、3 節で述べた単純なモデルでも、性能予測に十分な精度が得られることを確認した。

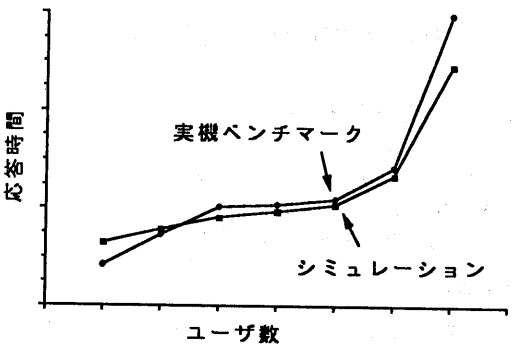


図 5 . 結果の比較

次に計算機A,B,C（表4）において、TPCベンチマーク実行時のスループット、およびボトルネック条件を予測した結果（図6、図7）を示す。

図6、図7の結果より以下のことが予測できる。

1. 計算機Aでは、CPUが有効に使われている。（CPUネック）
2. 計算機Bでは、CPU、ディスク、ロックの負荷がほぼ同等である。
3. 計算機Cでは、CPUが有効に使われていない。（ディスクネック、ロックネック）

ディスクやロックの負荷の割合が他のリソースと比べて相対的に高い原因は以下の通りである。

1. ディスク

TPC-Aの口座ファイルのレコード数は、例えば10tpsのシステムでは100万件である。本システムでは口座ファイルの索引部は3レベル、約10MB、データ部は100MBに達する。この場合、データ部、索引部いずれもキャッシュ・ヒットの確率は極めて低い。従って、口座ファイルを割り当てたディスクへのアクセスが集中する。このためディスクの負荷が高くなる。

2. ロック

TPCベンチマークでは全てのファイルをトランザクション回復処理の対象とする必要がある。本システムでは、ファイルを回復処理の対象とする場合、3.3節で述べたファイル制御情報の内容は、ディスク上で管理される。したがって、レコードの追加時には、ロックの期間中に、1. ヘッダのディスクからの入力、2. 更新結果のディスクへの出力 を行なうため、ロックの期間が長くなる。このため、このロックの負荷が高くなる。

以上のシミュレーション結果より、計算機B、Cでは、CPUオーバヘッドを削減する以前に、ディスクとロックの負荷を削減することが必要であると予測できる。改善策の詳細は5.2節で述べる。

5 考察

前節の結果より、TPCベンチマーク性能の向上のためには、

1. 計算機AではTPモニタを中心にCPUオーバヘッドを削減することが有効であり、
2. 計算機B、Cでは、CPUオーバヘッドの削減に先立ち、ディスクとロックの負荷の削減が必要である

と推測できる。

本節では、システムの改善項目をいくつか上げ、その効果を予測する。

5.1 CPUオーバヘッドの削減

CPUオーバヘッドを削減するために、TPモニタを以下のとおり改善した。

・改善1 –SVC発行回数の削減

4.2(2)で上げたSVC（signal,timeライク）を含めて、5種類のSVCをユーザ・ロジックに置換した。これによりSVCの発行回数を従来の半分以下に押えた。

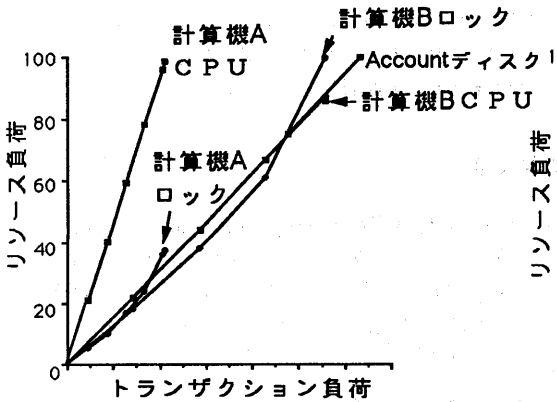


図6．計算機A，計算機Bの予測結果

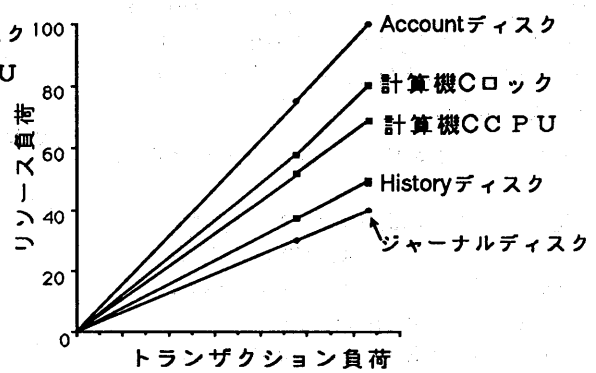


図7．計算機Cの予測結果

・改善2 –TP リソースのテーブルサーチ・アルゴリズムの改善
名前（例えば端末名）の昇順に、テーブルを整列し、バイナリ・サーチを用いてテーブルをアクセスする方式を用いた。この方式により、名前の数 n に対して、サーチ時間を $O(\log n)$ に押えた。

・改善3 – 時刻のログ方法の改善
1で述べた改善により TP モニタはユーザ・ロジックを用いて時刻を得る。しかし、この時刻も整数型の GMT である。従って、TP モニタが西暦表現（文字型）の時刻を得るためには、従来と同様の変換処理が必要である。

そこで、GMT（整数型）をそのままログ出力するように TP モニタを改造した。これにより、時刻変換処理は不必要となった。

これらの改善を施した TP モニタを使用してベンチマークを行なったところ、CPU 時間の合計が改善前の 70 %に減少した。これにより、CPU ネックの計算機ではベンチマーク性能が 1.4 倍向上することが予測できる。

5.2 ディスク負荷、ロック負荷の削減

これらの負荷を削減するために、以下の改善策が考えられる。

・改善1 – ファイルのデータ部、索引部を複数ディスク・ドライブへ分散配置する。

・改善2 – ファイル制御情報のロック期間を短縮する。このために、ファイル制御情報をメモリ上で管理できる仕組を導入し、ロック期間中のディスク入出力をなくす。（以上、ソフトウェアの改善）

・改善3 – ディスクアクセスを高速化する。（ハードウェアの改善）

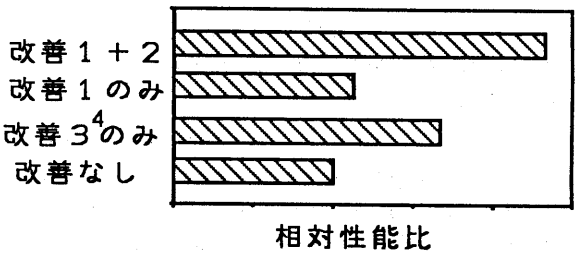
1は、ディスク負荷の削減に効果があり、2はロック負荷の削減に効果がある。3は、両方の負荷の削減に効果がある。これらの改善による計算機 C の TPC ベンチマーク性能の向上の割合をシミュレーションを用いて予測した。（図8）

シミュレーション結果より、以下のことが推測できる。

1. 製品レベルで最高速のディスク [3]を使用しても、ソフトウェアの改善がなければ、計算機 C の TPC ベンチマーク性能は改善なしの場合の 1.7 倍に押えられる。
2. 上記1、2の改善策は、単独で実現するのでは、効果が少ない。例えば、1だけの改善では、計算機 C の TPC ベンチマーク性能は改善なしの場合の 1.2 倍に押えられる。
3. 上記1、2の改善策を同時に実現すると、計算機 C の TPC ベンチマーク性能は改善なしの場合の 2.4 倍に達する。

ことが推測できる。

したがって、計算機 C では、上記1、2の両方の改善が必要と考える。シミュレーション結果では、1、2の両方の改善後、計算機 C は、CPU ネックとなる。従って、さらに性能を向上するには、CPU のオーバヘッドの削減が必要と考える。



[4] アクセス時間を通常の０．７５倍と仮定した。

図８．改善の効果

6 まとめ

当社ビジネスコンピュータおよび TPC-A で規定する処理形態を対象にベンチマークテストとシミュレーションとを用いて、性能評価をおこなった。

その結果、本論文で述べた単純なモデルでも、性能予測に十分な精度が得られることを確認した。また、ホストの CPU 能力の向上に対応して、改善の対象を、1）TP モニタのオーバヘッドの削減から、2）ディスクとデータロックの負荷の削減 に移す必要があることも確認した。

今回のモデル化では TPC-A のみを評価対象としたが、今後、これよりデータベース処理の負荷を高めた業務に対してモデルを検討していく予定である。

参考文献

[1] Transaction Processing Performance Council : *"TPC BENCHMARK A STANDARD SPECIFICATION"* ITOM INTL.CO, CODD & DATE CONSULTING GROUPE, 10 NOVENBER 1989.

[2] 田中 茂、丸山 光行：データベース処理におけるベンチマーク、情報処理、Vol.31,NO.3, pp.328-342

[3] アクセス時間を通常の 0.75 倍と仮定した。