オペレーティング・システム 43 － 5
（１９８９．６．９）

ある特定条件下のシステムオーバヘッド低減による性能向上の例

屋敷田広実
(株)東芝

概要

オンライントランザクション処理(以下OLTPと略す)は、取り扱うデータの保証をしながら、高級言語で記述されたアプリケーションプログラムの細かな要求を、多量かつ高速に処理するシステムで一般には基本ソフトウェアとアプリケーションの間に位置する。本報告は、アプリケーションや中間に位置するOLTPシステムそのものにはほとんど手を加えず、基本ソフトウェア(特にカーネル)を改造して性能向上に成功した例である。その主なポイントは、(1)I/O割り込みの抑制とtime quantamの適正化、(2)OSデバグ用システムトレースの短縮と高速化、(3)非同期出口ルーチンの即時ディスパッチ化、(4)特定プログラムの発行するSVCの特別扱い、等である。

An example of making performance up by reducing the system overhead

Hiromi YASHIKIDA
ToshibaCorporation
1-1-1、Shibaura、Minato-ku、Tokyo、105、Japan

Abstract

Online Transaction Processing (hereinafter called OLTP) exists in general between operating systems and applications,and processes speedily reliably large volumes of data at the request of application program written in high-level languages.This paper presents a successful example of enhanced performance archived by revamping the OS,specifically the kernel while the application and OLTP system itself remain virtually unchanged.The principal points are the following four items:

1) Control of I/O interruption and appropriation of time quantum.
2) System trace time reduction and high speed OS debug.
3) Impromptu dispatch for asynchronous exit routine.
4) SVC special treatment issued by specific programs,etc.

1. 調査

現行システムの処理能力と評価・改造のための詳細情報の採取

1.1 ロードジェネレータ

OLTPシステムでの実機負荷テストは容易ではない。数百～数千の端末を実務と同じようにオペレーションするには大変なコストと時間が掛かる。本システムではフロントエンドプロセッサ(以下FEPと略す)によるロードジェネレータを使って負荷テストを行った。

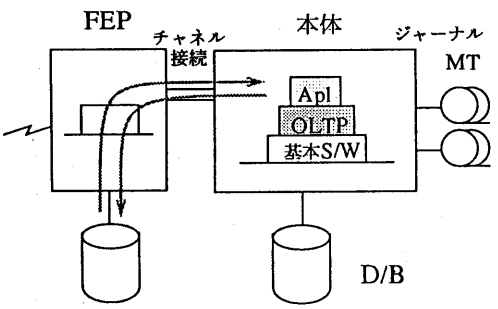


fig 1

FEP上のロードジェネレータは、あたかも回線の先にある端末からインプットされたごとくハードディスクドライブ(HDD)上のトランザクション(以下Txと略す)を読み出し、本体へ送る。本体はしかるべき処理後、FEP上のロードジェネレータへ返事を返す。FEPのCPUは本体より低速であるが、処理が少ないので本体のTx処理能力よりもはるかに高い送信能力を有する。

ロードジェネレータで発生するTxは、複数のタイプのミックスで3000件から成る。このTxタイプは本体側のアプリケーション開発の進行状況に合わせて変化する。又、本体側のデータ・ベースは実測の度に更新されてしまうので、毎回あらかじめ保存しておいた磁気テープから戻さねばならない。しかし幸い本システムのファイルシステムにはテスト更新機能が有り、負荷テスト終了時に更新前の状態に戻してくれる。

1.2 カウンタ

詳細情報の採集は、基本ソフトウェアのカウント機能と、ソフトウェアモニタによる採集機能の両方で実現している。

(1) HDDデバイスのI/Oカウント

基本ソフトウェア上のテーブルの中に幸いにもカウントされていたのでソフトウェアモニタで拾い上げるようにした。この情報でデバイスへのI/Oの集中状況が判る。

(2) カーネルからみたプログラムへのディスパッチカウント

CPUを各プログラムへ分配するディスパッチャは二つのタイプのディスパッチを行う。一つはノーマルなディスパッチで、もう一つはI/O完了時、あらかじめ予定されていたルーチン(非同期出口ルーチン)へのディスパッチである。ディスパッチ操作のために消費されるCPU時間とディスパッチ回数の積はTx処理能力を低下させる大きなオーバヘッドである。

(3) SVCのタイプ別発行回数のカウント

ユーザプログラムから発行されるSVC(スーパバイザコール)はある種の偏りがあるが、本OLTPシステムでの偏りを把握する目的でカウントした。カーネル上に新たにメモリセルを置くのは大変なのでテーブル上の使われていないメモリセルを探し、そこへカウントするようにした。

(4) OSサブモジュールの発行回数のカウント

本OSはいわゆるカーネル部と、非常駐の多くの小さなソフトウェアモジュールとで構築されているが、後者の使用頻度を調査するためカウントした。この結果使用頻度の特に高いソフトウェアモジュールは主メモリの許す範囲で主メモリ上に常駐させる。

(5) CPU時間の分類

小規模のOLTPシステムでは接続されているHDDの数が少ないためI/Oバウントになりがちだが、本システムはHDDが30台位になるためCPUバウンドになることが予想された。CPUはその動作モードによってスーパバイザモードとユーザモードに分けられる。更にスーパバイザモードではアイドル、オーバヘッド、プログラム実行に分けられる。

前者の測定はH/Wモニタがあれば最も簡単であるが準備できなかった。また、モード変遷の所にタイマーの切換ロジックを入れればソフトウェア的に測定可能であるが、そのためのオーバヘッドが大きいため本システムでは採用しなかった。

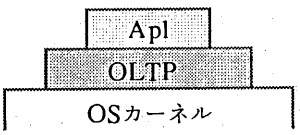


fig 2

その代りに
◇全アプリケーションプログラムのCPU時間
◇OLTPのCPU時間(スーパバイザモード+ユーザモード)
◇I/O終了処理とCPUディスパッチ時間=オーバヘッド時間
◇アイドル時間
を測定した。

2. ソースリストの追跡

1で採集した項目は(5)を除いて利用回数であり、それだけではTx処理に必要な時間は抽出できない。上記で得たカウント値からそれぞれOSモジュールのロジックについてソースリストをたどり、そのパスでのCPU時間を求めた。

(1) ディスパッチロジック

一般にCPUはプログラム実行中に、I/O終了やタイマー割込みを受けてそのプログラムをCPUのレディキューへつなぐ。CPUはI/O終了の処理を終えた後、ディスパッチャ内に入り、レディキューからプログラムを取り出して実行する。この操作のうちI/O終了処理を除いたコンテキストスイッチングと呼ばれる部分の実行時間はOSの良し悪しを決める。

科学技術計算のようにI/Oが比較的少ないプログラムではほとんど無関係であるが、OLTPシステムでは極めて重要である。

(2) I/O終了割込ロジック

本システムはOLTP専用マシンなのでI/OはFEP、D/B(HDD)、MTの三種である。

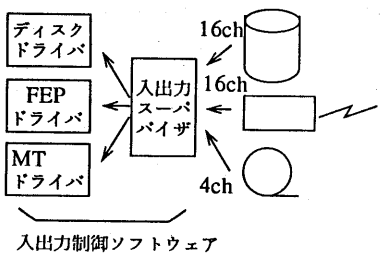


fig 3

CPUは極く限られたクリティカルゾーンの実行以外はI/O終了割込を受ける。この時、今まで実行中であったプログラムの状態保存操作が発生するが、その回数はI/O終了割込の回数と同じかそれより小さい。本システムではI/O終了割込発生時に複数のチャネルのI/Oが完了している場合、1回の割込で受けることができるからである。入出力スーパバイザ内のステップ数に比してプログラムの状態保存操作が大きいため何らかの改善が必要であった。

(3) 内部割込ロジック

ここでいう内部割込とはSVC、タイマー割込、演算エラー割込などであるが、正常動作している時はSVC割込が大半である。

しかし元々の割込処理ロジックは使用頻度を全く考慮しないジェネラルコーディングされていたので、改善による効果が期待できた。

(4) SVC毎の処理ロジック

1.2の(3)で測定されたSVCのタイプ別カウントの内、上位8種についてその処理ロジックとステップ数を調べた。代表格として一つ挙げるとすると、ユーザモードからスーパバイザモードへの遷移のためのSVCであろう。本OLTPシステムはセミシステムソフトウェアであり基本S/Wから権限を委ねられてスーパバイザモードで実行する部分が多い。

しかし常時スーパバイザモードで走行する訳ではないので頻繁にモード遷移が発生する。

(5) システムデバッグ用トレースロジック

ほとんど全てのOSはそのシステムデバッグ用に実行トレースを持っている。1の調査時、試みにシステムトレースを一時的に取らないように変更してみたところ約30%のCPU時間の短縮がみられた。そこで本システム動作中に取られるシステムトレースのタイプとその採集ロジックを追跡した。

本システムには70を越えるタイプのトレースがあるが、いくつかのものは対で使われており、必ずしも両方がなくてもトラブル発生時の解析に大きな支障がないことがわかった。

また元々本OSは複数のCPUで動作できるように設計されていたが、そのためにトレース採集ロジックが複雑かつ長大化していた。

こうして一命令ずつカウントしたステップ数は約2万ステップである。

3. 評価

事前の予想通り、負荷テスト時のCPUにアイドル時間はなく、CPUネックであることがわかった。

(1) CPU消費の分布

OLTPシステムはセミシステムソフトウェアとして、その上で動作するアプリケーションプログラムのCPU時間を保有している。また一方OSはOLTPシステムを一つのプログラムとして認識し、そのCPU時間を保持している。更にOSはI/O割込処理とディスパッチ操作のためのオーバヘッド時間を保存している。この三者とソースリストの追跡から

スーパバイザモード:ユーザモード=8:2

となり基本S/W側の改善がTx処理能力の向上に最重要であることがわかった。

(2)特定SVCへの集中

OLTPシステムが発行するSVCのタイプはほぼ8種に限られ、そのうちの3つがとび抜けて多かった。このうちの一つは先のモード遷移のSVCであり、他方は本OLTPシステムがその上で動作するアプリケーションプログラムへのCPUディスパッチのためのSVCと、アプリケーションからOLTPシステムへの要求SVCであった。

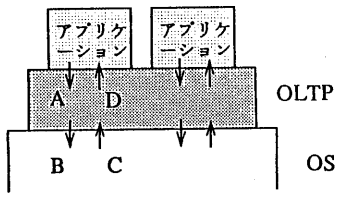


fig 4

上図

動作Aは	アプリケーションから OLTPシステムへの要求
動作Bは	OLTPシステムから OSへの要求
動作Cは	スーパバイザモードから OLTPモードへの遷移
動作Dは	OLTPモードから アプリケーションモードへの遷移

回数はB≧C>A≧Dだが、一回当りの時間を考慮に入れるとCPU消費時間はB>D>A>>Cとなった。

(3)システムトレースでCPUの30%を消費

システムトレースがこれ程の悪影響とは信じられないが、逆に言えば改善の格好の標的となる。

(4)I/O割込回数が160回/秒以上

I/Oの対象となる周辺装置によっても変化するが、プログラムがI/O要求を出すのに2.3mS位のCPU時間が必要である。またI/O終了割込の処理は0.4〜0.7mS位である。従って1秒間に160ケの入出力が行われるとすれば

$$160\times(2.3+\frac{0.7+0.4}{2})=456(mS)$$

がそのために消費される。

(5)サブディスパッチ回数が100回/秒以上

本システムのCPUの割り当てはOS内のディスパッチャからOLTPシステムへ、更にOLTPシステムは選択されたアプリケーションプログラムへとサブディスパッチされる。

このサブディスパッチはOLTPシステムから発行されるいくつかのSVCの中の一つであるが実行ステップ数が短く短縮の可能性が小さい。むしろサブディスパッチそのものの回数を減少する工夫が求められる。

4. 性能改善

100万ステップに近いアプリケーションプログラムを性能問題を意識してコーディングさせるのは不可能に近いし、また前述のように意味がない。

またOLTPシステムそのものに対してもアプリケーションよりは重みが増すが最重要なのはOS上のコーディングステップ数である。

また今回の改善の対象となるシステムはOLTP専用機になるため多少の制限を付けてもCPU走行距離を少しでも縮めることが第一である。

以下に今回の改善点を記述するがその変更量は約5千ステップである。

(1)I/O終了割込の抑制とTime quantamの適正化

I/O終了割込は余程の理由がない限り、これを抑制することはスループットの低下を招くため考慮の対象にもならない。しかしプログラム実行中には割込を受けず、OS内の状態保存操作が最も軽い所で受けるようにすれば1回当り約200μSの1秒間では

$$0.2\times160=32mS$$

のCPU時間が節約できる。

ユーザプログラム実行中に割込を受けないようにするため一般の割込禁止方法と異なり制限時間がない。

I/O終了割込を受けないためコンテキストスイッチングはタイマーレジスタにセットされた時間によって起動される。このタイマーレジスタにセットされる時間がタイムコンタムになるが今回は24mSとした。一方抑制されたI/O終了割込は幾つかがスタックされて

いる。

割込受付時にそれまでにスタックされていた処理が実行されアイドルになったチャネルにI/O要求がスタックされていればそれが発行される。つまりCPUはプログラムを実行してきて単位時間内で出せるだけのI/O要求をスタックに積み上げておき、I/O終了割込を受け付け、その処理が終了した時点で積み上げられていたI/O要求を一括処理する。(I/Oのバッチ処理)

(2) システムトレースの間引きと高速化

間引きは2つ以上の連続したトレースタイプを再合成で実現した。システムトラブル時の重要な解析材料であるので安易に間引くことは許されない。

システムトレースは各種のOSモジュールからトレースルーチンを呼ぶ集中型であった。これはマルチプロセッサを前提とした排他処理を実現するためである。今回の改造ではマルチプロセッサ方式を放棄したのでトレースも集中型と並列に各OSモジュールから直接トレースを残すことができるようにした。

こうしておいて使用頻度の高いトレースタイプのみ直接トレースを取る方法に変更した。(使用頻度の小さいものは従来通り)

この改造は最も容易かつ変更量が少なく、逆に最も高い効果が得られたと思われる。

(3) 非同期出口ルーチンの即時ディスパッチ化

OLTPシステムでは一般に多端末へのサービスを実現するため端末毎にタスクを受け持たせるマルチタスキング方式を取るが、本システムは前述のように親子方式を取っている。従ってOLTPシステムが発行する端末I/Oは全て即時復帰型である。端末I/O発行時にセットされた非同期出口ルーチンは、そのI/Oが完了した時点で入出力スーパバイザがI/Oの完了状態をプログラムへ直した後にディスパッチのレディキューへつながれ、やがてディスパッチされる。ここでOLTPシステムがセットした非同期出口ルーチンを調査したところ、ステップ数が短く単純であることがわかった。そこで入出力スーパバイザがI/O完了状態を通知後、そのまま非同期出口ルーチンを実行して再び入出力スーパバイザに戻るように改造した。(FIg5参照)

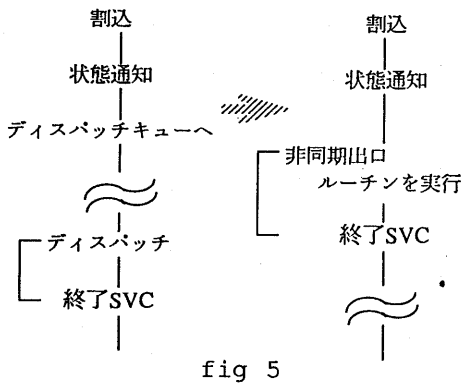


fig 5

(4) 特定プログラムが発行する特定SVCの特別扱い

OSから見るとOLTPシステムは一つのユーザプログラムである。そのユーザプログラムがスーパバイザモードで走行するためには運用開始時にシステム管理者の許可が必要であるが、この許可はタスクコントロールテーブル上にビットのON/OFFで示される。

実際にOLTPシステムが運用を開始し、スーパバイザモードへの遷移SVCが発行されると

a. SVCの割込ベクタが実行され
b. 他の割込ベクタのルートとマージされ
c. 割込んだアドレスやその他の情報を保存し
d. 再び割込のタイプ別に分岐し
e. 更にSVC割込のタイプ別に分岐し
f. タスクコントロールテーブル上のスーパバイザモード走行許可ビットをチェックし
g. スーパバイザモードのまま割込んだ次のアドレスへ戻る。

一般的なコーディングであるが、本ケースのように割込タイプ、SVCのタイプ、更にプログラムそのものも固定である場合は特別扱いすることで走行ステップ数の短縮が可能である。本ケースでは次の点で改造を加えた。

a. SVCの割込ベクタが実行され
b. 他の割込ベクタのルートとマージされる前にSVCのタイプを調べ
c. もしモード遷移のSVCなら、あらかじめセットしておいたタスクナンバーと実行中のタスクナンバーとを比較し
d. もしマッチすればスーパバイザモードのまま割込んだ次のアドレスへ戻る。

(5) 端末名のバイナリ値化

一般に端末名はわかり易い名前を付けられる。文字型にコード化されるためFEPドライバ内でのテーブル処理に時間が掛かる、実行時間が長い文字型の比較命令を端末数の1/2回数だけ繰り返すが、その処理は端

末とのメッセージ転送の時に2〜3回繰り返される。

本OLTPシスッテムでは端末側オペレータにとって文字型の端末名は不要である。そこで端末名をインデックスとして使えるようバイナリ値化すれば、FEPドライバ内でのテーブル処理は全てインデックスを使った処理が可能となる。

(6) 割込命令の先読みによる内部割込の抑制

ユーザプログラムがOSに対しての要求を割込で示すのと同じようにOLTPシステム上で動作するアプリケーションプログラムは割込でその要求をOLTPシステムへ伝える。この時CPUは

a.アプリケーションプログラム
b.OSカーネル(割込ベクタ)
c.OLTPシステム(アプリケーションの要求を処理して)

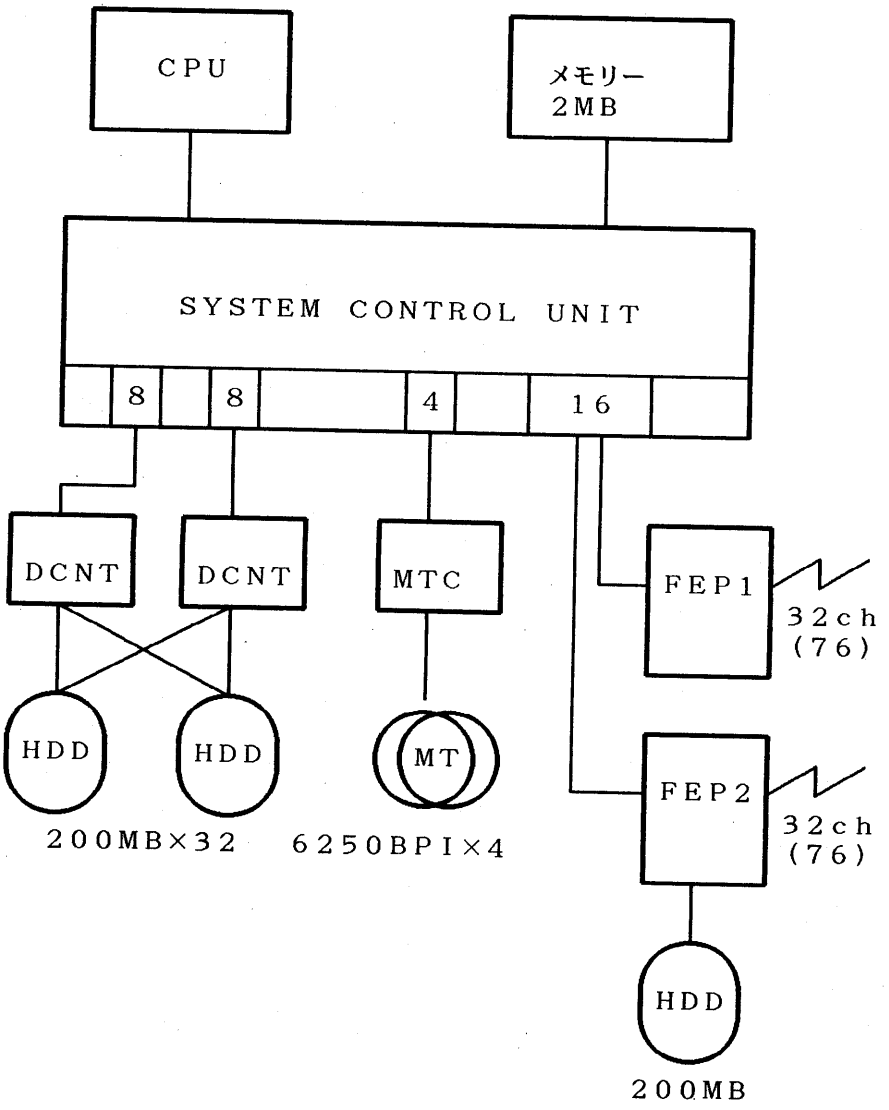
の順に動き、これを繰り返す。

アプリケーションプログラムではこの割込を発生される命令が連続することがあり、これを利用するためcからaへ戻る時、割込命令の次が再び割込命令であるかどうかを調査する。もし割込命令ならaとbのプロセスが省略でき、CPU消費時間の削減となる。

5. 結果

Txタイプのミックスによって改善率が異なるが、比較的重いミックスでは改善前のTxレートが10,500Tx/hに対し改善後は24,000Tx/h、また軽いミックスの場合、改善前が20,000Tx/hに対して改善後はその2倍の40,000Tx/hとなった。

システム構成


CPU
メモリー
2MB
SYSTEM CONTROL UNIT
8
8
4
16
DCNT
DCNT
MTC
FEP1
32ch
(76)
HDD
HDD
MT
FEP2
32ch
(76)
200MB×32
6250BPI×4
HDD
200MB