C／Sシステムにおける印刷処理のスループット向上について

恐　神　正　博*

Improvement of throughput for printing jobs on C/S systems

Masahiro Osogami

Abstract

Client & Server Systems (I will describe 'C/S Systems' in this paper) are a very popular environment for office computing now.

Many cases of client processes are extracting data from their RDB(relational data base) to client database applications or spreadsheets to re-format the demanded forms.

When C/S systems are installed, they are designed to be extended easily, but when new hardware is installed, it can cause the old hardware to be unstable.

Due to these changes, the old hardware has to deal with lager amounts of data which can cause tasks longer to complete. (Especially printing jobs.)

This paper will try to formulate one way to improve the throughput of printing jobs on the C/S systems where unstable hardware exists.

1．はじめに

現在企業等においてクライアントサーバシステム(以降C/Sシステムと表す)は広く用いられており，またクライアントにおける処理は，多くの場合データベースソフトまたはスプレッドシート上にデータを抽出しそれらのデータベースソフトやスプレッドシートから，必要な書式にデータを加工した上で印刷を行うなどである［1］.一方，C/Sシステムは導入後にクライアントを増やしていくことも多いため，最初に導入されたクライアントほど，後から追加されたクライアントに比べ，ハードウェア的に能力が劣っていることがしばしばある．しかしながら，社会の情報化の進展に伴い，扱うデータの量は飛躍的に増えてきており，数千件あるいは数万件を超えるデータの印刷処理が行われることも多い．ここでは，世界中で圧倒的なシェアを持つマイクロソフト社のWindowsオペレーティングシステム，また，同じくスプレッドシートの圧倒的なシェアを

* 経営情報学科

持つマイクロソフト社のExcelを用いた場合の，特にC/Sシステム上で比較的能力が劣ったハードウェア上における，印刷処理のスループット向上について述べる．

2．現状の把握

現在，C/Sシステムにおいて，サーバ上においてあるデータを抽出・加工した上で印刷処理を行う場合，その方法として，

1．基幹システムに組み込む．

2．ミドルウェア等を介し必要なデータを抽出した上で，クライアントのアプリケーションソフト上で印刷処理を行う．

の２通りが考えられる．

1．の基幹システムに組み込むのが一番確実な方法であり，従来のメインフレーム等では，全ての処理がこの方法で行われていた．しかしながら，この場合システム開発の専門家により開発されるのが通常であり，頻繁に帳票の変更等を行うのは時間・費用の面からも非常に難しかった．もちろん，C/Sシステムにおいても基幹システムを組み込みその中で印刷処理を行うことは行われているが，やはり従来のメインフレームと同様，頻繁に帳票の変更等を行うのは難しい．

一方，2．の方法はC/Sシステムの特徴であり，各クライアント上におのおの必要なデータをサーバから取り出し必要な処理を行うことが非常に簡単に利用できる．この方法によれば，各クライアントのアプリケーション上で帳票の打ち出しを行うため，帳票の変更等が各クライアント側で必要に応じ行える．つまり，クライアント上で利用しているアプリケーションソフトを用いることで，システム開発の専門家に頼らず各クライアントを利用するユーザの使い慣れた環境が利用できることになる．

2．1　C/S システムにおける印刷処理の問題点

すべて良いことづくめに感じるが，C/Sシステムで各クライアントにおいて帳票等の印刷処理を行っていく場合，抽出したデータをその都度必要な形に加工して印刷するだけでなく，ある程度まとまった件数を抽出し，定型の書式に一括印刷をしていきたいという要求も当然生じてくる．

この場合，「データを１件づつそれぞれの項目ごとに書式に従って配置し，そのページを印刷する」といったような手順を指示する必要が出てくる．

しかしながら，システム開発の専門家によるプログラミングの場合システムリソースまで考慮しながら慎重に行われるプログラミングも，こういった方法による場合には，システム全体のリソースまで考慮したプログラミングにはならず，例えば１つのプリントキュー（以降キューと表す）にかかってくる前処理および後処理等を少しでも減らすため，図１のようにできるだけ１つのキューに複数のページをまとめて印刷させるような処理も行われず，扱ったデータの件数分すべてをキューとして溜めてしまうことになってしまう．さらに，こういった処理が各クライアン

トから別々に行われるため，非常に負荷の高い処理が複数のクライアントから同時に行われることもありその結果，大量のキューがサーバに溜まり，最悪の場合スプールフルによりシステムダウンに追い込まれることも考えられる．

一方，C/Sシステムにおけるもう一つの問題点として，クライアントの追加やプリンタの追加あるいはサーバの追加等が比較的容易に行われることで，最初に導入されたハードウェアと後から追加されたハードウェアとの処理能力の差が大きく開いてしまうといったことがあげられる．そして，大量の印刷処理が能力の劣ったプリンタに集まった場合，非常に時間がかかってしまう場合が生じる．

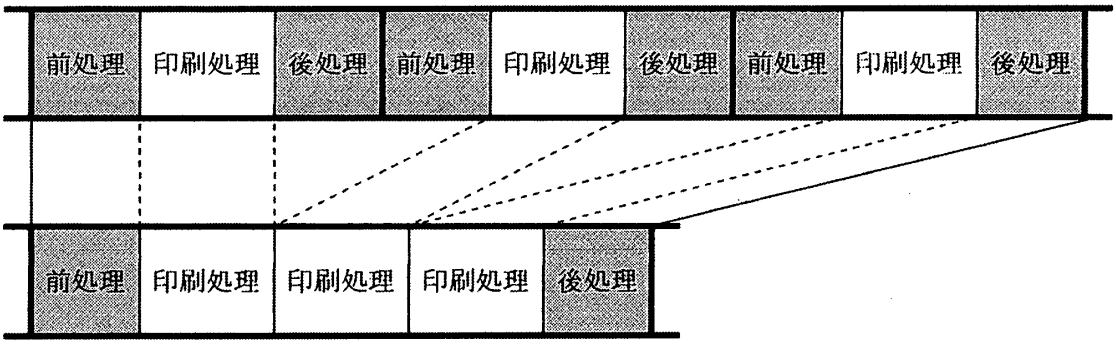

図１．印刷処理における前処理・後処理の省略イメージ図

２．２　アプリケーションの操作上の問題

C/Sシステムにおいて，サーバ上にあるデータを抽出・加工した上で印刷処理をする際よく用いられている方法として，マルチシートの扱えるスプレッドシート上に，データ用のシートおよび書式を整えた印刷用のシートを用意し，マクロと呼ばれる簡易プログラミング言語を用いてプログラミングすることがあげられる．特にマイクロソフト社のWindowsオペレーティングシステムおよび同じくマイクロソフト社のExcelは世界的にも圧倒的なシェアを持っており，ここではそれらを用いた場合について考える．

Excel上に抽出したデータを，データ用のシート上におき，そこから印刷用にフォーマットしたシートへデータを移動して印刷処理を行う場合，通常は１レコードごとに１つの印刷処理を行わせる．これは，Excelのマクロと呼ばれる簡易プログラミング言語のVBAを用いて行うことがほとんどだが，そのプログラミングが最も単純にできるからである．多くのクライアントはシステムのリソースまで考慮したプログラミングを行わないし，複数のレコードを１つの印刷処理で行うためには，フォーマットしたシートも，その処理を指示するVBAも複雑にせざるを得ないため，各クライアントにそれらを要求するのは困難だからである．

しかしこの場合，数千件のデータの印刷処理を行うと数千件の印刷ジョブを処理することとなり印刷スピードが処理スピードに追いつかないためキューが非常に多く溜まることになる．図２にNetWareのプリントジョブのコンソール画面を示すが，このようにキューがどんどん溜まって

いき，場合によっては数千以上溜まってしまうこともある．

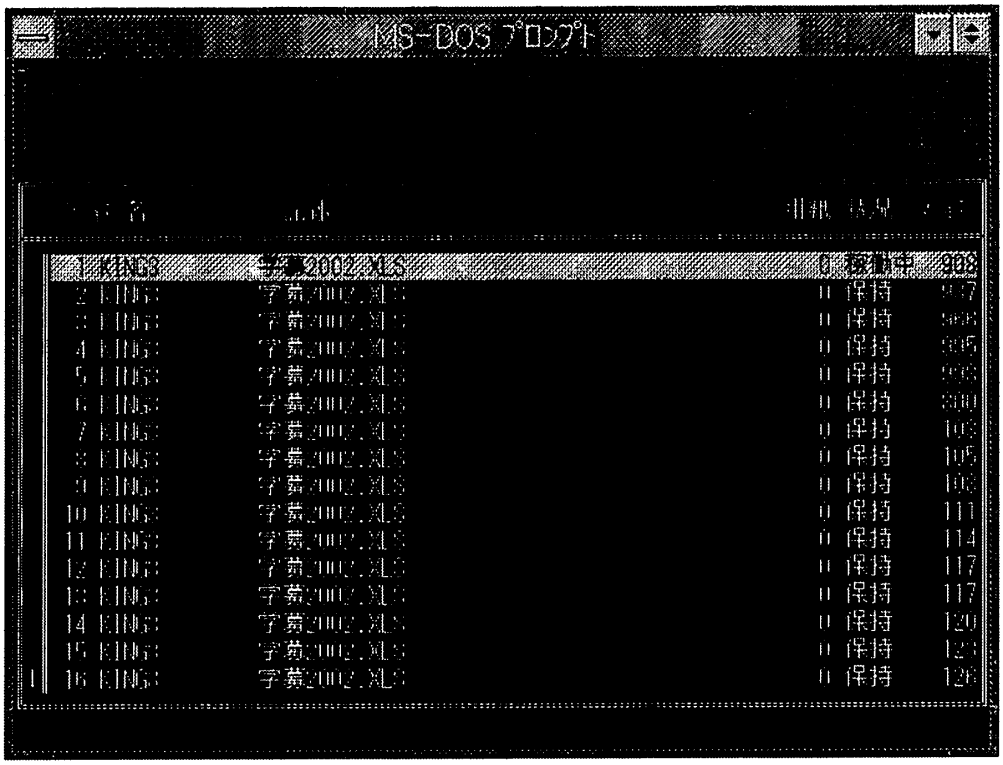

図２．キューが大量に溜まった状態のコンソール画面

また，スプールフルによりサーバがダウンするのを防ぐため，レコードを数十件ごとに分けて，キューが無くなってから次の処理を行うなどのオペレーションが必要となってくる場合もあり，処理が終わったかどうかの確認を逐一確認する等，相当の手間と時間が必要になってくる場合もある．

３．処理時間短縮の確認

ところが図１で示したように，１つのキューについて，必ず前処理と後処理がそれぞれ必要になってくるので，数千件の処理を行う場合，前処理と後処理の部分で相当な時間を要することとなる．短縮の度合いは，システムのチューニング・構成あるいはハードウェアの能力によっても左右するが，これらを省くことで印刷処理のスループットは間違いなく向上できるはずである．

そこで実際に，表１の環境を用いて，１ページごとの印刷処理を１０回行う場合（キューの数が１０個）と５ページごとの印刷処理を２回行う場合（キューの数が２個）の処理を行い，それぞれの時間を比べてみた．表２にその結果を示す．

なお表１の環境は，ハードウェアの能力があまり高くはないが，前述のようにC/Sシステムにおいて，クライアントの追加やプリンタの追加あるいはサーバの追加等が繰り返されていった結果，ハードウェアとの処理能力の差が大きく開いてしまうこともあるため，特にこの環境で実験を行った．

表１．実験を行ったシステムの構成

サーバ	富士通FM-360SV NetWare 3.12J Oracle 7.3
クライアント	富士通FMV-575D4 Windows 3.1 KeySQL 2.5 Excel 5
プリンタ	富士通F6671NT

表２．実験結果

処 理 方 法	キューとキューの間の平均待ち時間	全体の処理時間	１つのキューにおける平均印刷時間
1ページごとの印刷を10回行った場合	１分１７秒	１２分　８秒	２秒
5ページごとの印刷を2回行った場合	１分１９秒	１分５４秒	９秒

やはり結果は同じページ数の印刷を行うのに，１ページごとに分けて行った場合と連続して印刷を行った場合とでは，連続して印刷を行った場合の方が６分の１以下の時間で処理を終了できた．さらに，１つのキューにおける平均印刷時間を見ると，１ページ当たりの印刷時間にはほとんど差がなく，キューとキューとの待ち時間で全体の処理時間に差が出ることが確認できた．

特に，この環境では処理時間にかなりの差が生じたが，たとえいかなる環境でも処理時間に差は出るはずであり，また，扱うデータの件数が多いほどスループットを向上させることができるはずである．

４．複数のレコードに対する連続したページの作成

ここで，１つのキューに複数のレコードの印刷処理を行わせるためには，印刷用にフォーマットしたシートを複数ページ用にフォーマットし直し，さらにVBA等のプログラム上でそれぞれのレコードに対するおのおのの項目を挿入する位置を指定していかなければならない．図３は単ページの印刷を指示しているVBAの例だが，これを複数ページの印刷に対応させた場合，図４のようになり，Data1,Data2の部分が複数レコードの分だけ必要となってくる．これに伴い，当然印

刷用のシートも対応させていく必要が出てくる.

できるだけ連続したページを作成した方がスループットの向上につながることは実験により確認できたが，ページを増やせば増やすほど，プログラムを複雑にしていかなくてはならない.

```
Sub Insatu()
    Dim i As Integer
    Dim name0, DATA_org As String
    DATA_org = "data_base"
    name0 = "印刷"
    Sheets(name0).Select
    i = 2
    Do Until Sheets(name0_org).Cells(i, 1) = Empty
        Sheets(name0).Range("A2,B31").ClearContents
        Sheets(name0).Range("A2")  = "    " & Sheets(DATA_org).Cells(i, 2)  'Data1
        Sheets(name0).Range("B31") = Sheets(DATA_org).Cells(i, 11)          'Data2
                    .
                    .
                    .
        Sheets(name0).PrintOut From:=1, To:=1, Copies:=1
        i = i + 1
    Loop
End Sub
```

図３．単ページの印刷を指示しているVBAの例

```
Sub Insatu()
    Dim i As Integer
    Dim name0, DATA_org As String
    DATA_org = "data_base"
    name0 = "印刷"
    Sheets(name0).Select
    i = 2
    Do Until Sheets(name0_org).Cells(i, 1) = Empty
        Sheets(name0).Range("A2,B31").ClearContents
        Sheets(name0).Range("A2")  = "    " & Sheets(DATA_org).Cells(i, 2)  'Data1
        Sheets(name0).Range("B31") = Sheets(DATA_org).Cells(i, 11)          'Data2
                    .
                    .
                    .
        Sheets(name0).Range("A32,B62").ClearContents
        Sheets(name0).Range("A32")  = "    " & Sheets(DATA_org).Cells(i+1, 2) 'Data1
        Sheets(name0).Range("B62") = Sheets(DATA_org).Cells(i+1, 11)         'Data2
                    .
                    .
                    .
        Sheets(name0).PrintOut
        i = i + n
    Loop
End Sub
```

図４．図３を複数ページの印刷用に対応させた場合のVBAの例

一方，実際問題としてレイアウトの変更や追加等は結構頻繁に行われており，それらの印刷処理を行うExcelのファイルは変更や追加の数だけ増えてくることになるため，既存のExcelのファイルをすべて複数レコードの印刷に対応させるのはかなり煩雑な手間がかかることになる.

そこで，従来の単ページの印刷時に実際の印刷を行わず，そのページをそのまま別に用意した仮のシート上へ一旦コピーし，その後も単ページの印刷時の度ごとに，印刷は行わず次々と仮のシート上へコピー・挿入していく．そして，ある程度ページ数がまとまったら仮のシートを印刷するという方法により，既存のExcelファイルをほぼそのまま利用しながら，複数のレコードに対する連続したページの印刷を可能にした．図５にこのイメージを示す.

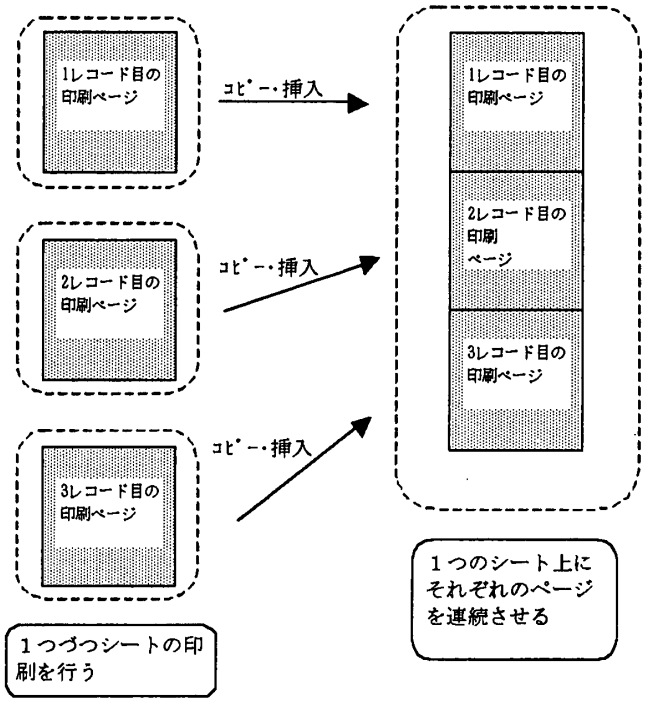

図５．複数ページへの対応イメージ図

この方法を取ると，既存のExcelファイルをそのまま利用し，印刷処理を指示しているVBA上で印刷用のシートを印刷する命令文を入れ替えるだけでよく，非常に簡単に複数のレコードに対する連続したページの印刷を可能にすることができる．

実際に従来通りの方法で行った場合の処理時間と，ここで述べた新しい方法によって行われた場合の処理時間を比べ表３に示す．但し，実際に業務で使われている全国高校宛の発送文書（件数約321件）を用いた．なお，従来の方法における時間については，サーバがスプールフルとなり全件数を一挙に印刷できなかったため，１ページごとの平均印刷時間を計測し(約1分17秒であった)それを321倍することにより求めた．

表３．従来の方法と今回の方法の処理時間

処 理 方 法	全体の処理時間
従来の単ページによる印刷	６時間５１分５７秒（２４７１７秒）
今回の複数ページによる印刷	３０分４３秒（　１８４３秒）

結果は１３倍以上の処理スピードを実現できた．従来半日以上かけて行っていた業務をほぼ半時間で完了したことになる．Excelファイルの変更もVBAで書かれた処理プログラムを２行変更し，汎用プログラムの書かれたモジュールシートを挿入しただけで終わった．

５．むすび

今回提案した手法を用いると，印刷処理におけるスループットの大幅な向上が図れ，しかも，既存のExcelファイルとそのVBAにはほとんど手を入れずに対応させることができる．今回は実際の業務における１つの処理についてのみ処理時間短縮の確認を行ったが，実際には，何十種類ものExcelファイルについて同じように改善を行うことができるため，従来何週間もかけて行われていた処理を数日で完了させることが可能になる．

また，ここで用いられた手法はExcelファイルであればそのまますべてのファイルに即対応させることができるだけでなく，マルチシートをサポートした他のスプレッドシートにも応用が可能であるため，C/S システムにおけるスプレッドシートを用いた多くの印刷処理のスループット向上を図ることができると考えられる．

参考文献

[1] 吉岡, 山本, 松尾, 須藤, "C/Sネットワーキング",
"日経BP出版センター", 1995

[2] 恐神, "C/Sシステムにおけるスプレッドシートを用いた印刷処理のスループット向上について",
"2003年電子情報通信学会基礎・境界ソサイエティー大会", 2003.9

（平成16年11月30日受理）