連載

今からでも遅くない！ RPGⅢユーザーに贈る

RPGⅣの魅力と可能性

第３回　静的プロシージャーによるパフォーマンス向上

中村 潤
株式会社アイ・ラーニング
IT 研修本部　IBM 製品研修部
ラーニング・アドバイザー

今回で3回目の連載となります。1回目ではフリーフォーム、2回目では日付計算について話をしました。どちらもコーディングレベルでの話ですので、コーディング、すなわち開発が差し迫っていない方にはあまり魅力的ではなかったかもしれません。今回はコーディングレベルだけでなく、パフォーマンスにも多大な影響を及ぼす話をいたします。これまでも時々登場した言葉で、ILE、統合化言語環境（Integrated Language Environment）についてです。

統合化言語環境とは、異なる言語で書かれたモジュールを組み合わせ、1つのプログラムとして実行できるものであると、前々回に説明いたしました。システム開発において複数の言語を使うことがあるのかどうかの話はさておき、プログラム開発において、モジュール化、外部サブルーチンという考え方は、どの言語においても共通した考え方です。

MAIN001は、データベース・ファイルTOKMSPを順次に読み取り、TOKMSP中のフィールド、TKBANGを外部プログラムのSUB001にパラメーターとして渡し、実行します（**図表１**、**図表２**）。TKBANGは得意先番号で、SUB001はTKBANGを受け取り、該当する得意先番号のレコードをデータベース・ファイルJUMIDPから読み取り、受注金額を集計したのち、2番目のパラメーターフィールド、GOKEIに返します。

呼び出し元のプログラム、MAIN001では集計した値を@Ｐ2で受け取り、集計した金額を元にしてこの後の処理が続いていきます。その後、次のTOKMSPのレコードが読み取られ、TOKMSPのすべてのレコードを読み終わるまで、処理は続きます。つまりSUB001は、TOKMSPのレコード件数分CALL（実行）されることになります。

この場合、MAIN001もSUB001も、プログラムのコンパイル段階では、お互いの存在を確認する情報はまったく持ち合わせていません（**図表３**）。サンプルコーディングのCALL命令ではSUB001がどこのライブラリーなのかを示す情報もありませんので、CALLされた時点で初めてSUB001をライブラリーリストから探し出し、MAIN001から渡すパラメーターと、SUB001で受け取るパラメーターの順番とデータタイプをチェックします。パラメーターの桁数、データタイプ、順番がMAIN001とSUB001で違っていると、エラーになります。

これらのチェックをすべて実行して初めてSUB001は主記憶域上にロードされ、SUB001の初期設定実行後、演算仕様書の命令を実行し始めます。このSUB001の実行プロセスは、MAIN001で読み込まれる得意先マスターファイルのレコード件数が1万件であれば、1万回行われることになります。

この1万回のオーバーヘッドの発生は、MAIN001とSUB001が完全に別プログラム、別オブジェクトであることに起因します。SUB001の初期設定プロセスは、プログラム終了時にLR標識をオフのままRETRUN命令だけ終了することによって、再度SUB001がCALLされる際には前回終了したときと同じ主記憶域上の情報を再利用できるため、ある程度のオーバーヘッドの軽減は実現できますが、MAIN001からSUB001をCALLする際の、SUB001の場所、アドレス解決だけはどうにもなりません。

図表1　プログラム：MAIN001

```
FFILENAME++IPEASF.....L.....A.DEVICE+.KEYWORDS+++++++++++++++++++++++++

FTOKMSP    IF   E           K DISK
FQPRINT    O    F  132        PRINTER

CL0N01FACTOR1+++++++OPCODE&EXTFACTOR2+++++++RESULT++++++++LEN++D+HILOEQ

C                   DOU       %EOF(TOKMSP)
C                   READ      TOKMSP

C                   IF        NOT %EOF
C                   CALL      'SUB001'
C                   PARM      TKBANG        @P1               5
C                   PARM                    @P2              11 0
C                   EXCEPT    MEISAI
C                   ENDIF
```

図表2　プログラム：SUB001

```
FILENAME++IPEASF.....L.....A.DEVICE+.KEYWORDS+++++++++++++++++++++++++

FJUMIDL01  IF   E           K DISK

CL0N01FACTOR1+++++++OPCODE&EXTFACTOR2+++++++RESULT++++++++LEN++D+HILOEQ

C     *ENTRY        PLIST
C                   PARM                    TKBANG            5
C                   PARM                    GOUKEI           11 0
C*
C                   EVAL      GOUKEI = *ZERO
C     TKBANG        SETLL     JUMIDR
C     TKBANG        READE     JUMIDR
C                   DOW       NOT %EOF(JUMIDL01)
C                   EVAL      GOUKEI+=JHKING
C     TKBANG        READE     JUMIDR
C                   ENDDO
C                   RETURN
C*
```

図表3　RPGⅢプログラムとRPGⅣプログラムのコンパイル

RPGⅢプログラムのコンパイル

```
CRTRPGPGM   PGM(PGMLIB/MAIN001)   SRCFILE(SRCLIB/QRPGSRC)   SRCMBR(MAIN001)
```

RPG Ⅳプログラムのコンパイル

```
CRTBNDRPG   PGM(PGMLIB/MAIN001)   SRCFILE(SRCLIB/QRPGLESRC)   SRCMBR(MAIN001)
```

ILE環境で実行されるプログラムは、コンパイル時にアドレスは決定済みです。プログラムがどこにあるのかがあらかじめわかっていれば、プログラムの実行が早いのは当然です。

RPG ⅢなどOPM（Original Program Model）でのプログラム開発環境は、1つのソースメンバーをコンパイルすれば、1つのプログラムができました。プログラムはオブジェクトタイプが＊PGMとなり、5250画面のコマンド入力行からCALLコマンド、RPGプログラムからもCALL命令で実行されます。実はRPG Ⅳでも、CRTBNDRPGコマンドでコンパイルを行えば、＊PGMのオブジェクトが作成され、手順そのものはOPMとさほど変わりはありません。

しかしILE環境のプログラム作成手順では、ソースメンバーをコンパイルして、モジュールというオブジェクトを作成します（オブジェクトタイプは＊MODULE）。このモジュールは通常のコンパイル同様に文法エラーをチェックし、文法エラーがなければ生成させるオブジェクトですが、＊PGMのオブジェクトのように、CALLで実行可能な形式のオブジェクトではありません。実行可能なプログラムにするには、複数のモジュールを組み合わせてプログラムを作成する作業が必要です（**図表4**）。

つまり、OPMに比べてコマンドは1回多くなってしまうわけですが、1ソースメンバーからプログラムを作成する場合は前述のようにCRTBNDRPGを使うので、これらのコマンドを使わずともプログラムは作成できます。これらの作業が必要になるのは、MAIN001とSUB001を結合させ、1つのプログラムに結合する場合です。

結合する場合、MAIN001のほうにロジックの変更が必要になります。SUB001を呼び出すロジックはCALL命令ではなく、CALLB命令を使用します。CALLBは静的プロシージャーの呼出し命令です（**図表5**）。

図表4　モジュールとプログラムの作成

モジュールの作成

```
CRTRPGMOD    MODULE(PGMLIB/MAIN001)    SRCFILE(SRCLIB/QRPGLESRC)
SRCMBR(MAIN001)
```

プログラムの作成

```
CRTPGM   PGM(PGMLIB/MAIN001)   MODULE(MAIN001)   ENTMOD(MAIN001)
```

図表5　プログラム：MAIN001から静的プロシージャーを実行する例

```
FFILENAME++IPEASF.....L.....A.DEVICE+.KEYWORDS+++++++++++++++++++++++++++

FTOKMSP    IF   E           K DISK
FQPRINT    O    F  132        PRINTER

CL0N01FACTOR1+++++++OPCODE&EXTFACTOR2+++++++RESULT++++++++LEN++D+HILOEQ

C                   DOU       %EOF(TOKMSP)
C                   READ      TOKMSP

C                   IF        NOT %EOF
C                   CALLB     'SUB001'
C                   PARM      TKBANG         @P1               5
C                   PARM                     @P2              11 0
C                   EXCEPT    MEISAI
C                   ENDIF
```

ここで初めてプロシージャーという用語が出てきました。プロシージャーとはILE環境でのプログラムの単位で、CALL命令や、5250画面でのCALLコマンドで実行することはできません。

今回のケースですと、MAIN001、SUB001、ともにCRTRPGMODコマンドでコンパイルし、モジュールを作成します。次にCRTPGMコマンドで、MAIN001、SUB001を基にした、PGM001を作成します。CRTPGMコマンドでは作成されるプログラム名と、プログラムを構成するモジュール群、さらには入口点、つまり最初に実行されるプロシージャーを含むモジュール名をENTMODパラメーター指定します（**図表6**）。

PGM001は＊PGMオブジェクトで、CALLコマンドで実行できます。PGM001がCALLされると、入口点となるプロシージャー、MAIN001が実行され、さらにMAIN001からバインドされたプロシージャー、SUB001が実行されます。SUB001がCALLされることはCRTPGMの段階で決定されているので、実行されるスピードはCALLで動的に呼び出されるプログラムのそれよりも速くなります。

しかもILEのすごいところは、CRTPGMで結合されるプログラムが、RPGと、COBOL、C、CLプログラムでも可能だということです。そういう意味では、ほとんどのお客様が、RPGとCLを結合したILEプログラムを作成できる環境にあるといえるのです。

今回のサンプルのMAIN001から呼び出されるSUB001はCLプログラムであってもよく（CLプログラムで今回の処理を実行できるかどうかは別として、単にCLプログラムが実行できるという意味です）、CLプログラムは、CRTCLMODコマンドで、モジュールを作成できます。このモジュールをCRTPGMコマンドで結合すればいいだけの話です。

また、逆にCLプログラムからプロシージャーをCALLする場合、CALLコマンドではなく、CALLPRCコマンドを使用します。ここで注意しなければいけないのは、CALLPRCコマンドが使えるのはソース仕様タイプがCLPではなく、CLLEでなくてはいけないということです。通常、CLプログラム用のソースファイルQCLSRCは、ソース仕様タイプがほとんどCLPのソースメンバーですので、それとは区別するために、QCLLESRCなどのソースファイルを作り、CLLEメンバー専用のソースファイルにするとよいでしょう（**図表7**、**図表8**）。

対話型プログラムのように、1画面で1プロシージャーを呼び出すような処理ではパフォーマンス改善はほとんど期待できませんが、膨大な件数のレコードを処理し、レコード単位で外部サブルーチンをCALLするようなプログラムであれば、このCALLBでの呼び出し、またはCLLPRCでの呼び出しをやってみる価値はあると思います。CALLBのほうが速いのですから、あえて遅いまま使い続ける必要はないでしょう。

またILEにするからといって、すべてのアプリケーションをRPG ⅢからRPG Ⅳに移行する必要はありません。適用業務では＊PGMのオブジェクトをCALLすればいいだけですから、RPGでできたプログラムであろうがCOBOLでできたプログラムであろうが関係ないとも言えます。いまRPG Ⅲをお使いの場合でも、一度にすべてRPG Ⅳに乗り換える必要はないのです。できるところから徐々に切り替えていってはいかがでしょうか。

図表6　モジュールとプログラムの作成

モジュールの作成

```
CRTRPGMOD   MODULE(PGMLIB/MAIN001)   SRCFILE(SRCLIB/QRPGLESRC)
SRCMBR(MAIN001)
CRTRPGMOD  MODULE(PGMLIB/SUB001)   SRCFILE(SRCLIB/QRPGLESRC)  SRCMBR(SUB001)
```

プログラムの作成

```
CRTPGM   PGM(PGMLIB/PGM001)   MODULE(MAIN001 SUB001)   ENTMOD(MAIN001)
```

図表7　CLプログラムがプロシージャーをCALLする場合

```
/*****************************************************************/
 /** PROGRAM ID    - CLP010                                       **/
 /**          TITLE - 業務メニュー                                 **/
 /*****************************************************************/
              PGM
              DCLF        FILE(CLP010D)
              MONMSG      MSGID(CPF0000) EXEC(GOTO CMDLBL(LOOP))
  LOOP:       SNDRCVF     RCDFMT(FMT10)
              IF          COND(&IN03 = '1') THEN(RETURN)
              IF          COND(&X1SECT = '99')  THEN(DO)
              SIGNOFF
              ENDDO
  /*                                                            */
              IF          COND(&X1SECT = '01') THEN(DO)
              CALLPRC     PRC(INT010)
              ENDDO
  /*                                                            */
              IF          COND(&X1SECT = '02') THEN(DO)
              CALLPRC     PRC(RPG010)
              ENDDO
  /*                                                            */
              IF          COND(&X1SECT = '03') THEN(DO)
              CALLPRC     PRC(CLP020)
              ENDDO
  /*                                                            */
              IF          COND(&X1SECT = '11')  THEN(DO)
              CALLPRC     PRC(RPG050)
              ENDDO
  /*                                                            */
              IF          COND(&X1SECT = '21')  THEN(DO)
              WRKOUTQ     OUTQ(GAS01XX/GAS01XX)
              ENDDO
  /*                                                            */
              GOTO        CMDLBL(LOOP)
```

図表8　モジュールとプログラムの作成

モジュールの作成

```
CRTCLMOD   MODULE(PGMLIB/CLP010)   SRCFILE(SRCLIB/QCLLESRC)     SRCMBR(CLP010)
CRTRPGMOD  MODULE(PGMLIB/INT010)   SRCFILE(SRCLIB/QRPGLESRC)    SRCMBR(INT010)
CRTRPGMOD  MODULE(PGMLIB/RPG010)   SRCFILE(SRCLIB/QRPGLESRC)    SRCMBR(RPG010)
CRTCLMOD   MODULE(PGMLIB/CLP020)   SRCFILE(SRCLIB/QCLLESRC)     SRCMBR(CLP020)
CRTRPGMOD  MODULE(PGMLIB/RPG050)   SRCFILE(SRCLIB/QRPGLESRC)    SRCMBR(RPG050)
```

プログラムの作成

```
CRTPGM  PGM(PGMLIB/MENU01)  MODULE(CLP010 INT010 RPG010 CLP020 RPG050)
                            ENTMOD(CLP010)
```