B-011

バッチジョブ実行基盤と世代管理ファイル高信頼化方法の検討と実装

細内昌明† 遠藤匠‡ 向田康裕‡ 山口俊朗‡ 佐藤一浩‡ 坂田啓一‡

メインフレームバッチジョブのオープンシステムへの移行コスト削減のため，バッチジョブ実行基盤ソフトウェアを実装した。ジョブ異常終了時の一時ファイル解放漏れを防止するファイルアロケータ等メインフレームジョブ管理の特長機能をサポートすると共に，JCL に相当するジョブ定義ファイルの仕様としてオペランド値互換の属性やプロシジャ要素をもつXML形式を採用することで，AP非依存部分の自動変換を可能とした。

また，システムダウン後も世代管理ファイルの消失を防ぎ，世代管理ファイルと世代ディレクトリとの整合性維持が可能な高信頼世代ファイル登録処理を実現し，ジョブ管理におけるファイルアロケータの信頼性を向上した。

1．背景と目的

コスト削減を主目的として，レガシーマイグレーション(メインフレームからオープンシステムへの移行)のソリューションが提供されている[1]。移行手法の1つであるリホスト（プログラム構造を変更せずに移行する手法）では，従来バッチジョブ定義ファイルをシェルスクリプトへ移行していた。しかし，メインフレームのバッチジョブファイル記述言語であるJCL(Job Control Language)とシェルとの仕様差が大きく，移行に伴う手作業が多くなるとともに，移行後のスクリプトがわかりやすいものにならなかった。

このため，メインフレームのジョブ管理と同様の機能を移行先システムで実現するバッチジョブ実行基盤ソフトウェアを実装した。

2．バッチジョブ実行基盤の概要

実装したバッチジョブ実行基盤の概要を，図1に示す。メインフレームのジョブ管理と同様に，ジョブ定義ファイルの解析処理，ジョブ実行に必要な入出力資源の割当解放を行うアロケータ，ジョブの各プログラムの起動・完了待ち制御，ジョブ実行結果であるジョブログ出力処理から構成される。

JCL に相当するジョブ定義ファイルの仕様として，JCL類似の XML 形式を採用した。JCL の文/オペランドと XML の要素/属性を対応させオペランド値と属性値に互換性をもたせたことと，JCL 特有のカタログドプロシジャをサポートしたことにより，アプリケーション非依存部分の記述の自動変換を可能とした。また，XML 実体へのシェルスクリプト埋込みが可能であるため，入力パラメータ等のアプリケーション依存部分の記述修正を実体に対して行えばよいので，自動変換後の手修正も容易である。

JCL 以外のユーザ資産であるジョブネット記述についても，JCL と同様にジョブステップ単位にプログラムを実行するためジョブネットを変更せずに移行可能とした。ソースコードについても，帳票書式サポートにより書式移行を容易にし，移行に伴う COBOL プログラムの修正量を削減した。

† (株)日立製作所　システム開発研究所
‡ (株)日立製作所　ソフトウェア事業部

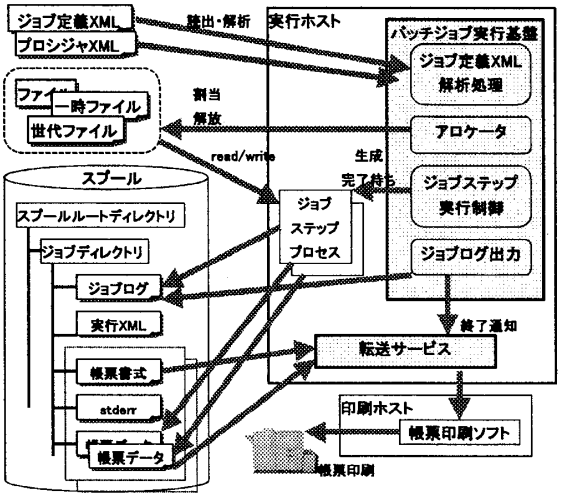

図1　バッチジョブ実行基盤の概要

3．アロケータと世代ファイルの機能概要

メインフレームジョブ管理の特長機能の1つであるアロケータが割当解放する入出力資源には，通常のファイルやスプール上のファイルのほか，一時ファイル・世代ファイルがある。一時ファイルは，ジョブ中で作成されジョブ終了時には不要となる作業用ファイルである。ジョブ定義XMLファイルに，

```
<DD NAME="ENV1" TYPE="TEMP" />
```

のようなDD要素の記述があると，プログラム開始前に作業用ディレクトリにファイルを自動生成する。プログラムでは，NAME 属性値に指定した名称の環境変数からファイル名を取得しオープンする。プログラム終了後にはバッチジョブ実行基盤が削除するので，ジョブが異常終了したときも確実に削除できる。また，スクリプトでの作業用ファイルの削除処理記述量を削減できる。

世代ファイルは，時間的関連を持ったファイルのグループ（世代グループ）内のファイルであり，バックアップ用途等に利用される。世代ファイルを生成するには，ジョブ定義ファイルに，以下のように TYPE=GDG を付加した DD 要素を記述する。

```
<DD NAME="ENV2" TYPE="GDG" DSN="gdgname(+1)" />
```

ジョブを実行すると，DSN 属性値に指定した世代グループ gdgname に，最新世代ファイルが生成・追加される。

DSN属性値には，世代グループ名の後に相対世代番号を指定する。相対世代番号は，同一世代グループ内の各世代を時間順に識別するための番号である。最新世代を0，新世代を正値，旧世代を負値（時間的に古いほど番号が増加）で表す。作成した世代ファイルを参照するには，相対世代番号に0または負値を指定する。

同一世代グループの世代ファイルは，同一のディレクトリ（世代ディレクトリ）に配置する。世代ディレクトリの構造例を以下に示す。

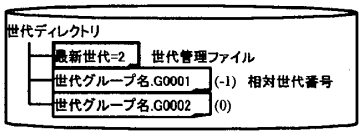

図2　世代ディレクトリの構造

世代ファイルの名称は，世代グループ名と絶対世代番号で構成される。絶対世代番号はファイルを生成するごとに単純増加する番号である。旧世代を参照するときは，最新世代の絶対世代番号に相対世代番号を加えて絶対世代番号に変換し，ファイル名を求める。最新世代の絶対世代番号を管理するため，世代管理ファイルを世代ファイルと同じディレクトリに作成する。

4．世代管理ファイルの高信頼化方法

世代管理ファイルに最新世代の絶対世代番号を記録することで，高速に世代番号変換が可能となるが，世代ファイル登録処理中にプロセス強制終了やシステムダウンが発生すると，世代管理ファイル更新中であれば，最新世代の絶対世代番号が失われる危険性が生じる。また，世代ファイル名変更と世代管理ファイル更新との間であれば，世代管理ファイル中の最新世代番号と，世代ディレクトリに存在する最新世代の世代ファイルの絶対世代番号が不一致となり，次に世代登録するときに絶対世代番号が抜けてしまうといった不都合が生じる。

このため，世代ファイル登録処理中に異常が生じても，世代管理ファイルの消失を防ぎ，世代管理ファイルと世代ディレクトリとの整合性を維持する方法を考案・実装した。

本方法の世代ファイル登録処理によるディレクトリ構造の遷移を図3に示す。まず，マスタとなる世代ファイルの複製であるバックアップ世代管理ファイルを作成する（②の状態になる）。バックアップをとったあとは，バックアップファイルは更新しないので，更新状態でない世代管理ファイルが必ず1つは存在することになり，世代管理ファイル出力中の消失を防ぐことができる。

次に，新世代として登録する登録対象ファイルを，世代ファイルのディレクトリに移動すると同時に，ファイル名を仮登録ファイル名である”世代グループ名.new”とする（③へ移行）。この処理を仮登録と呼ぶ。

次に，マスタ世代管理ファイルを更新し，最新世代を3とする（④へ移行）。

最後に，仮登録ファイルの名称を”世代グループ名.G0003”のように最新世代番号を含む名称に変更する。これを本登録処理と呼ぶ。

このように，登録対象ファイル名を，仮登録ファイル名を経て世代ファイル名に変更することで，世代ファイルの登録状態（仮登録前か，仮登録と本登録との間か，本登録後か）を，世代ファイルの名称で判別可能となる。異常終了後にどちらの世代管理ファイルの内容を有効とするかを，世代ファイルの登録状態，すなわち仮登録ファイル名が存在するか否かによって判別する。世代ディレクトリ中に仮登録ファイルが存在すればバックアップした世代管理ファイルの内容を有効とし，存在しなければマスタ世代管理ファイルの内容を有効とする。ファイル名変更がそのまま状態変更となり，世代ファイルの名称と世代管理ファイル選択が連動するので，ディレクトリ構造と世代管理ファイル内容との整合性を維持することができる。

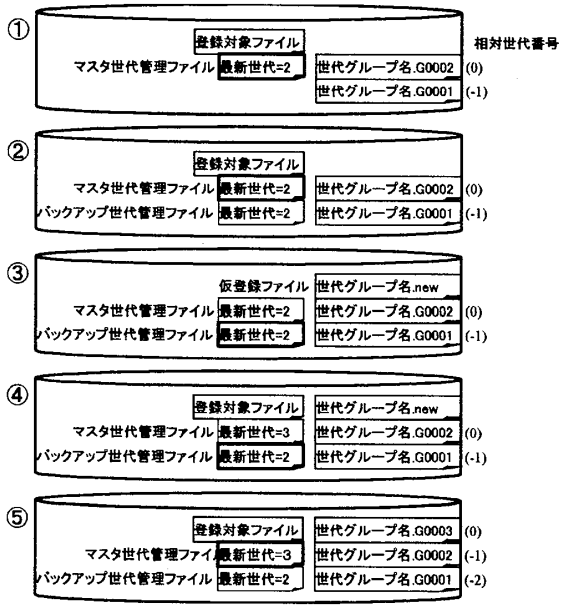

図3　世代登録によるディレクトリ構造の遷移

5．まとめ

本稿では，オープンシステムにおけるメインフレーム同等のバッチジョブ実行基盤の概要と，アロケータにおける世代管理ファイル高信頼化方法を提示した。本稿で示した方法により，メインフレームバッチジョブの移行コストを削減するとともに，システムダウン等に対するジョブ管理の信頼性を向上することができる。

参考文献

[1] 吉田他：中規模システム向けマイグレーション事業，日立システムジャーナル第7巻 p9-16