5U-1

オフィス手続き自動化システムＯＰＡの障害回復機能の設計

吉本真二，岸本一男，平松健司，翁長健治

広島大学工学部

１．概要

著者らは、複数ワークステーション（以下、ＷＳと略す）をつなぐＬＡＮ（Local Area Network）上で、処理と通信を一体化したＯＡ（Office Automation）用システムを効率よく記述することを目指し、分散並列処理言語ＯＰＡＬ（Office Procedure Automation Language）及び、その動作環境であるＯＰＡ（Office Procedure Automation）－システムを設計作成している[1][2]。ＯＰＡＬのプロトタイプは、既にＮＥＣ・ＰＣ－９８０１をつないだＬＡＮ上で稼働中であるが、障害が発生したときの対処についてはまだ考慮されていない。本研究では、障害が発生した場合に、ＯＰＡＬ処理系上でデータの一貫性を保ちながら障害回復を行う方法を設計する。

２．ＯＰＡＬの概略

ＯＰＡＬは、Ｐａｓｃａｌ風の手続き的言語であり、オフィス手続き記述のために並列処理機能及び、２次元表形式ファイルの操作能力を付加したものである。ＯＰＡＬでは、

１）共有データへのアクセスがない書類の転送や承認手続き

２）共有データのアクセスと複数人による対話的入力の両方を含むスケジュール調整手続き、物品購入手続き

３）個人で共有データのアクセスを行う文献検索

等の各事務手続きの自動化を念頭においている。

３．同時実行制御と一貫性

オフィス手続きが自動化された場合、その処理は各部署のＷＳ上での処理，ＷＳ上の担当者の判断・決定の入力，各部署間の通信の３つからなると考えられる。これら三者を全体として一体化しようとするとき、各ＷＳ上の担当者の決定の入力は、他の二つと異なり実時間に近い処理を行うことが出来ないことが予想される。従って、データ一貫性確保の方法に注意が要求される。本システムでは、ロックとアボート[3]を併用した次の方法を用いることにした。

１．データレコードは各レコードごとにタイムスタンプ欄があり、最も新しく書き込みが行われた日時が記入されている。

２．共有データファイルからの読み込みは、書き込みロックがかかっていない限りいつでも許される。このとき、読み込みのたびにトランザクション内の所定の欄に、そのレコードのタイムスタンプ欄に記入されている値をコピーして所持する。

３．共有データへの書き込みでは、先ず人間による最後の対話的入力が行われた後、システムが書き込みを行おうとするレコードに対して書き込みロックをかけ、読み込んだ全てのレコードに対して、タイムスタンプの照合を行う。もし、データの読み込みの後に同じデータへの書き込みが１つでもあったことが判明した場合、書き込みロックは失敗しプログラムはアボートされる。もし、衝突がなかったら、データのロックは成功しプログラムは書き込みを含んで進行する。プログラム終了時にこのロックは解放される。

この様なロック法の妥当性は次のように要約される。

１）対話的入力が高々一回でデータの検索，書き込みを行う場合は通常のロック方式と同じである。

２）共有ファイルへのアクセスがなく単にメールの転送となる場合（承認手続き等）、ローカルなデータに対しては一貫性が問題になることはない。

３）複数の対話的入力と共有ファイルへのアクセスの両方を含む場合、実行効率を無視すればデータの一貫性は保証される。このとき、タイムスタンプによりアボートが行われるために、実行期間が長期に及んだ場合でも１）２）のタイプの実行が妨げられることはない。

ここで、複数人による対話的入力がある場合、実行時間が長くなるためにアボートの割合が増えて、効率面で有効なプログラムが書けなくなるのではないかということが問題になる。この有効性を示す一例として、スケジュール調整手続きがある。この手続きでは、関係者のスケジュールファイルの各レコードは、５分ないし１５分を一単位とする時間を表しているとする。調整手順は次の通りである。

１）幹事はその時間を資源とみて、関係者の空き資源（時間）を検索する。

２）参照した空き時間のうちからいくつか候補を選び、関係者に出欠の可否を確認する。

３）関係者の返答を待って、最も望ましい日取りを決定し全員に通知する（書き込み）。

このとき、３）の時点で、１）の確認以後にどこかのＷＳのスケジュール表の当該時間に”書き込み”が行われているか否かが検査されるわけだが、これは人間によって現在行われているスケジュール調整と同一内容なの

A Design of Recovery Mechanism in Office Procedure Automation System OPA

Shinji YOSHIMOTO, Kazuo KISHIMOTO, Takeshi HIRAMATSU, Kenji ONAGA

Hiroshima Univ.

で、人間によるスケジュール調整が実用的に動作しているなら、このプログラムも同程度以上に効率的に実行される。

4．障害回復

本研究では、次に述べる三つの障害に対する回復をサポートするが、媒体障害については考えない。

1）プログラム自身による障害（いわゆる、トランザクション障害）：　不正な入力データ等のために発生するプログラム自身のアボート（0割，配列の添え字の範囲外の実行時の検出等）

2）システム障害：　ＯＳの誤りやハードウェア障害によって生じ、処理が無制御のまま終了する。ここでは、主記憶の内容は失われるが不揮発性の記憶装置は影響を受けないと仮定する。

3）ユーザによるアボート：　ユーザが、本人あるいは他のＷＳのユーザの入力ミス等の意味的な誤りに気付き、意図的にアボートする場合。通常の計算機でいう”ブレークキーを押す操作”に当たるが、このとき対話的入出力画面から、アボートするプログラムのＩＤ，再実行のブロック，アボート理由等を入力することにより、プログラムのあるブロックから再実行を要求できるといった機能も含めて考える。

これらを実現するための情報として、ＲＥＤＯ－Ｌｏｇファイル及び、再実行に必要なデータファイルが必要となるが、これらについては４．１で述べる。１）２）３）のうち、１）３）については４．２でその回復処理を述べる。２）については、一定時間間隔で各ＷＳが正常に動作していることをチェックし障害を発見するが、その詳細はここでは省略する。

4．1　再実行のための情報

本システムでは、１つのプログラムの実行をメインプロセスの存在するＷＳが監視しており、１）３）によるアボートが発生すると直ちに回復を行う。１つのプログラムは複数のＷＳ上で並列に実行されるので、回復に際してシステムは、プログラムの処理がどこまで進んだかを知らねばならない。この情報を格納しておくために、各ＷＳ上には　ＲＥＤＯ－Ｌｏｇ　ファイルと呼ばれるシステムファイルが１つずつ準備されている。このファイルへの書き込みアルゴリズムは、次のとおりである。

１）プログラムが起動されたとき、そのプログラムの識別子を持ったBEGIN-PROGRAMレコードを書き込む。

２）１つのプログラムを構成しているブロックが起動された時、そのプログラム及びブロックの識別子等を持ったBEGIN-BLOCKレコードを書き込み、ブロックが終了したときEND-BLOCKレコードを書き込む。但し、他のＷＳのブロックを起動する場合には、起動のための通信が行われたときにBEGIN-BLOCKレコードを、また、そのＷＳからメインプロセスのブロックを起動するための通信を受信したときにEND-BLOCKレコードを書き込む。

３）メインＷＳのデータコミット要求に対して、全てのＷＳからAcknoledgementを受信したとき、そのプログラムに対する全てのＬＯＧレコードを消去する。

また、本システムでは、プログラムのあるブロックから再実行を行うことを考慮して、各ブロックを起動したときのデータ及び、メインＷＳ上の各ブロックを起動した際に必要としたシステム変数をブロックごとにそれぞれ１つのファイルに保存しておく。このデータファイルはプログラム終了時に全て消去される。

4．2　回復処理

本節では、アボート要求を受信したメインＷＳ，アボート信号を受信したＷＳでの処理を述べる。本システムでは、ファイルへの書き込みは概念的には直接行われるが、書き込み中に障害が起こる可能性があるので、コピーに対して書き込みを行いコミット信号受信後に以前のレコードを除去する等の実装上の工夫が必要である。

＜アボート要求を受信したメインＷＳでの処理＞

プログラム自身によるアボート：

ＩＦ　対話的入力があった
　ＴＨＥＮ　最も最近実行されたＷＳに制御を渡す
　ＥＬＳＥ　ＲＥＤＯ－Ｌｏｇファイルを参照し通信を行ったＷＳに対しプログラムの無効、取り消し通知を送信する。

ユーザによるアボート：

ＲＥＤＯ－Ｌｏｇファイルとユーザによって決定されたやり直しブロックを参照し、そのブロックを起動したメインプロセスのブロックを再起動する。また、そのブロックとそのブロック以後に実行されたブロックを有するＷＳにそのプログラムの無効通知とアボート理由を送信する。その後、プログラムを再実行するかしないかはユーザの判断による。

＜アボート信号を受信したＷＳでの処理＞

ＩＦ　実行要求を格納しているキューの中にアボートするプログラムの実行要求が存在する。
　ＴＨＥＮ　実行要求をキューから消去する。
　ＥＬＳＥ　ＩＦ　現在実行中のプログラムがアボートするプログラムである
　　ＴＨＥＮ　実行中止

アボート理由を出力し、再実行が必要な場合は実行要求をキューに登録する

5．おわりに

本研究は、ＯＰＡＬ処理系上でデータの一貫性を保つ方法及び、障害回復処理について設計を行った。なお、本研究は文部省科学研究費の援助を得て行っている。

＜参考文献＞

[1] Kishimoto, K., Onaga, K., Utsunomiya, H.: OPAL: An office procedure automation language for local area network environments via active mailing and program dispatching, in "Languages for Automation", Chang, S.-K.(Ed.), Plenum Press : New York, pp.67-93,(1985).

[2] 岸本，織田，吉本，翁長：オフィス手続き自動化言語ＯＰＡＬとその実装．　準備中

[3] Date, C. J.:"An Introduction to Database Systems , Volume II", Addison-Wesley Publishing Company : Reading, 1983.