6K−1

ＰＭアプリケーションにおけるイベント制御型構造の考察

池田潔、中野富也、阿部賢一
ＮＴＴデータ通信（株）

１．はじめに

ＭＳ　ＯＳ／２ＰＭ等のハードウェア・インタフェースが仮想化されたＯＳでは、キーボードの入力等がアプリケーションに対して非同期に通知される構造を持つ。

つまり、アプリケーションからＩ／Ｏに問合せる従来の方式だけではうまく処理できず、非同期に通知されるイベントを効率よく処理する構造が必要になる。

ここでは、その対策の１つであるイベント制御型アプリケーション構造を報告する。

２．アプリケーションの条件

端末システムにおけるオンラインのトランザクション処理やローカルのデータ入力処理等では、入力業務と通信処理、又、検索処理等とのマルチタスク処理を基本とし、同時に、処理レスポンス性能の高速化が求められている。

このために、これらの条件を満足し、ＰＭ環境下で効率よく動作するアプリケーション構造を検討した。

３．イベント制御型構造

ＰＭプログラムでは、画面表示系、キーボード入力系等のメッセージは、アプリケーションの動作に無関係にかつ非同期にキューを通じて通知される。このため、アプリケーションは、速やかにメッセージを取り出し、対応する処理への振り分けを行い、さらに、すぐ次のメッセージを処理することを要求される。

この処理が遅いと、例えば、キーボードの入力レスポンスが悪いと言う形でユーザには見える。したがって、いかに速くメッセージを処理し、次のメッセージに備えるかがＰＭ環境下のアプリケーションの課題となる。

又、マルチタスク環境をシングルプロセッサで実現するＰＭでは、特定の処理だけがプロセッサ資源を占有して走行する構造であるとその処理自体は速く終了するが、他の処理がその間停止したままになってしまうことがある。そのために、最終的なシステム全体としてのスループット低下の事態になることもあり得る。

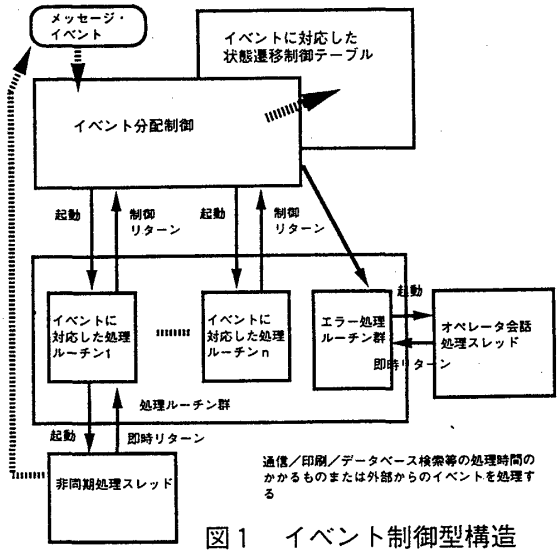

図１　イベント制御型構造

すなわち、ＰＭアーキテクチャの中でアプリケーションの処理の単位を極小化し、リソースの再配分を効率よく行う方策が必要とされる。

オンライン・トランザクション処理やデータエントリ処理の順序や表現は個々のアプリケーションで異なる。だが、実行されるプログラムはほとんどの場合相違が少なく、処理ルーチンとして小さな部品の組み合わせのみで表すことができる。

そこで、アプリケーションでイベントを振り分けるための制御テーブルを設ける。制御テーブルは、処理の状態とその状態で処理すべきイベント及び、そのイベントに対する処理ルーチンで構成する。つまり、テーブル記述を行うことで配下の処理ルーチンを組み合わせ、イベント振り分けでアプリーケション処理を実現することができる。

このことに着目し、イベントに対する処理の単純化と、それを組み合わせるためのイベント制御テーブル記述方法を検討した。

イベント制御型構造図を図１に示す。

A Programming Design using Events Driven Structure for Presentation Manager
Kiyoshi IKEDA, Tomiya NAKANO, Kenichi ABE
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION

また、通信等では通信相手との関係上、その処理を短くできない場合が存在する。このような処理をＰＭ環境下でのイベント制御型構造に取り込む場合、その処理を別スレッド化し、終了時にイベントとしてメッセージを通知することにより、処理結果をイベント分配制御に通知する形とする。

このような構造を取ることにより、処理ルーチンはイベント分配制御から常にコールされるだけとなり、直接の他の処理をコールするような方式を排除できる。処理ルーチン間のデータ交換は、主記憶の共有領域を使用することで解決できる。

通信処理等から非同期に通知されるイベントは制御テーブルに記述することで、最終的な処理の同期管理を行うことができる。

他の時間のかかるような処理においても、通信処理と同様の手法で実現できる。

端末Ｉ／Ｏ単位の状態管理が望ましい複数処理がマルチタスクで走行する場合も制御テーブルを階層化することで構造をわかりやすくすることができる。実用的には多段階層はアプリケーション構造を複雑するが再帰的な階層構造記述を可能としており、設計条件に応じて拡張は可能である。

4．性能への負荷

イベント制御構造を採用した場合、マルチタスク環境化の性能に問題がないか検証した。

図２に、前景で文字入力を実行した場合のマルチタスク処理の影響を示す。

処理の単位が短く優先的に処理される文字入力は、背景での頻度の高い環境下でもほとんど影響を受けていない。

図３に背景の印刷処理に対するマルチタスク処理の影響を示す。

ここでは、他に複数の処理が重なると印刷処理スループットは明らかに低下することを示す。

マルチタスク処理としては当然の帰結と言え、性能低下は見られるが、全体的には処理単位の極小化を行う限りイベント振り分けの停滞の結果は得られなかった。

5．イベント制御型構造の利点

本構造においてもアプリケーションの要請である次の点を満足している。

(1) マルチタスク処理においても対応できる。

(2) レスポンスタイムもサンプルベースでは許容される範囲にある。

さらに、本構造は次のような利点がある。

(3) テーブル記述により状態遷移の設計、管理が統合化され、同時に機能の拡張が容易である。

(4) 独立性の高い処理ルーチン実現し、サブルーチンとしての再利用が図れる。

(5) 制御テーブルにより機能別パッケージ化することが容易である。

(6) イベント分配制御でイベントを監視できるたマルチタスクのデバッグが容易である。

(7) システム設計／プログラム設計時における生産性向上のための設計単位の分業化と柔軟性に配慮できる。

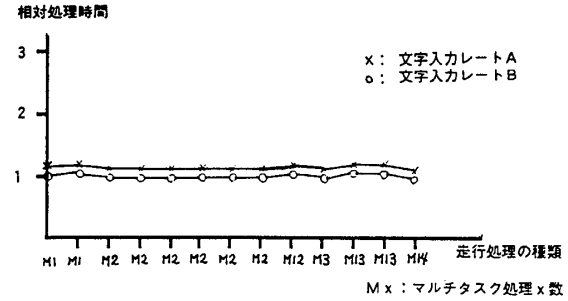

図２　前景文字入力

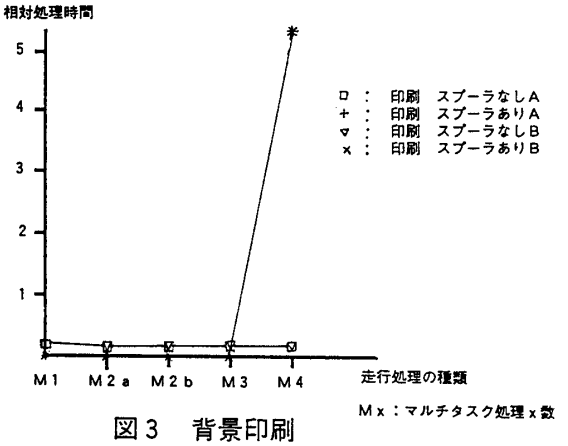

図３　背景印刷

6．おわりに

今回は、構造化の実現を中心に行ったため、現在の構造は、サンプルベースのアプリケーションでしか評価されていない。

実際のフィールドで走行するようなアプリケーションでの評価が今後の課題である。

MS OS/2 : 米国におけるﾏｲｸﾛｿﾌﾄ 社の登録商標
OS/2 : IBM Corp. の商標
PM : Presentation Manager