3Z-01 Partskitにおけるスケジューリング機構の部品化

原 敏弘 大鎌 広 藤原 祥隆
北見工業大学

1 はじめに

現在、GUIを使用したアプリケーションが広く利用されている。しかしながら、GUIを使用したプログラムは、そうではないプログラムと比べてプログラマの負担が大きい[1]。

筆者らは本研究室において、Partskit[2]を研究している。Partskitは並行動作するC++クラスオブジェクト(部品と呼ぶ)を通信線で接続し、組み合わせる事でプログラムを記述するというスタイルを提供している。

並行動作はプロセスやスレッドを用いる方法も考えられるが、双方とも通信インターフェースが複雑になるので、通信を通信線を介す方式に限定した。コルーチンを用いることによって競合問題等を解消し、通信インターフェースを単純化している。

さらにコルーチンの通信を適切に設定することでUNIXのフィルタコマンドの持つ再利用可能な性質とC++言語のクラスライブラリとしての再利用可能かつ拡張可能な性質を併せ持つことが可能になる。

2 疑似並列動作上の問題点

従来実装されていたPartskitのコルーチンの動作はFIFO方式である。しかし、この方式ではユーザが全く動作を要求しない時でもプログラムは動作し続け、CPU資源を無駄にしてしまう。

要求されない無駄な動作は、ユーザからのアクション等への応答速度やアプリケーションプログラム全体のパフォーマンスに少なからず影響を与え、また、アプリケーションプログラムの性質により応答速度を重視するかスループットを重視するか等最適化すべき性能指標の種類が異なるため、最適なスケジューリング方針が異なる。

そこで、本研究ではアプリケーションプログラムに最適なスケジューリング機能をアプリケーションプログラマが利用可能にするためのインターフェースをPartskitシステムに追加を行なうことを目的としている。

3 スケジューリング

プログラミング上の各部品間の制御の切り替えは、各部品内でコンテキストスイッチ(CSwitch()関数)を明示的に呼ぶことによって行われる。そこで、制御が切り替わる部品の順番を適切に変更したり、アプリケーションプログラム全体の動作が不要のときに動作を停止したりすることによってCPU資源を解放することを考える。

しかし、アプリケーションプログラムに最適なスケジューリングは前述の通りアプリケーションプログラムによって異なる。そこで、スケジューリング機能も部品(スケジューリング部品と呼ぶ)によって実現することで適切なスケジューリングが選択できるようになる。自分自身をスケジューリングに含めることによってアプリケーションプログラマが特定のアプリケーションに適したスケジューリングを実現したり、スケジューリング機能の再利用が可能になる。

スケジューリング部品には、スケジューリングのためにいくつかのヒントを与える。

まず、すべての部品はスケジュール部品に対して「期待するスケジューリング動作」をヒントとして持つことができる。このヒントを使えば、スケジューリング部品は常に部品の優先順位を他のヒントから調べて決定しなくても、アプリケーションプログラムの状態によって動作を変更しなければならない時のみ動作させ、あとはPartskitシステムに任せることも可能である。

アプリケーションプログラムの動作時に得られるヒントとして、

- 部品間の通信線上のデータ量、
- Xから送られてくるイベント、
- 時間(前回の部品の動作から指定した時間が経過したか)、
- ファイルディスクプリタ(指定したファイルディスクプリタへデータが到着したか)、

を参照する機構を用意した。これらのヒントと通信線からのデータを参考にスケジューリング部品はどの部品を優先して動作させるかを決定することになる。

さらに、すべての部品はスケジュール部品に対して「期待するスケジューリング動作」をヒントとして持つことができるようにした。このヒントを使えば、スケジューリング部品は常に部品の優先順位を他のヒントから調べて決定しなくても、アプリケーションプログラムの状態によって動作を変更しなければならない時のみ動作させ、あとはPartskitシステムに任せることも可能である。

Component implementation of scheduling mechanism in Partskit
Yoshihiro HARA, Yoshitaka FUJIWARA
Dept. of Computer Sciences Kitami Institute of Technology
165, Koen, Kitami, Hokkaido 090, Japan
Hiroshi OHKAMA
Dept. of Electrical and Electronic Engineering Muroran Institute of Technology
27-1, Mizumoto, Muroran, Hokkaido 050, Japan

スケジューリングの機構を盛り込んだ部品の動作を図1に示す。

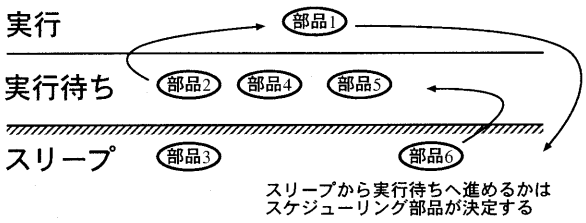

図 1: 部品のスケジューリング

3.1 部品毎のスケジューリングポリシの設定

部品が期待するスケジューリング動作は部品毎に SetPartsPriority() という部品クラスメンバ関数で登録する。

3.1.1 レスポンス重視のアプリケーションプログラム

マウスからの入力に素早く反応したい場合、X からのイベントが来ている部品に優先的に動作を割り当てた方がより素早く反応できる。このような部品は SetPartsPriority(PP_XEVENT, 1); とすることにより、X からのイベントが来たときに動作するようにとスケジューリング部品にヒントが与えられる。

3.1.2 スループット重視のアプリケーションプログラム

大規模な計算を行ったり、部品間で大量のデータを通信するような場合、計算途中の部品や通信線にデータが残っている部品に優先的に動作を割り当てた方が効率が良いと考えられる。SetPartsPriority(PP_INDATA, <通信線番号>); とすることにより、通信線にデータが来たときに動作するようにとスケジューリング部品にヒントが与えられ、SetPartsPriority(PP_ALWAYS, 1); とすると常にその部品が動作し続けるようにとスケジューリング部品にヒントが与えられる。

3.1.3 低 CPU 負荷重視のアプリケーションプログラム

時計などのデスクトップアクセサリを考えると、なるべく CPU 負荷をかけず、それでも停止しているわけにもいかないので、一定時間毎に部品を動作させることが考えられる。SetPartsPriority(PP_TIMER, <時間>); とするとこの部品は前回の動作から指定した時間が経過してから動作するようにとスケジューリング部品にヒントが与えられる。<時間>の単位は [ミリ秒] である。

3.2 スケジューリング部品による部品の動作の制御

スケジューリング部品は与えられたヒントを基に実際の部品の動作を決定する。
GetPartsPriorityResult() 関数を呼ぶと各部品に設定されたスケジューリングポリシにしたがって動作させるべき部品のリストとその理由を知ることができる。アプリケーションプログラムの動作時に得られるヒントは次の関数で得ることができる。

- getInPortBytes()
  - 通信線の入力ポートに到着しているデータ数
- getOutPortBytes()
  - 通信線の出力可能データ数
- isEventQueueEmpty()
  - X からのイベントが来ていないか
- getElapseTime()
  - 前回の動作からどれだけ時間が経過したか
- isFdEmpty()
  - ファイルディスクプリタにデータは来ていないか

これらの入手可能なヒントから動作させたい部品を SetNextParts() 関数に渡し、スリープ状態の部品を実行待ち状態に進めることによりスケジューリングを行う。

4 まとめ

アプリケーションの種類によって適切なスケジューリングポリシを実装するため、スケジューリング機構を部品として実装できる設計を提案した。

今後はこのスケジューリング部品によるスケジューリング方式の性能比較やチューニングを行っていく。

参考文献

[1] 増井俊之: “GUI ベースのプログラミング,” bit 別冊 ビジュアルインタフェース —ポスト GUI を目指して—, pp.45-64, 共立出版, 1996

[2] 畠 俊一, 大鎌 広, 藤原 祥隆: “GUI プログラミング方式: Paerskit”, 情報処理学会 全国大会, 1998.3.