神奈川大学理学部　情報科学科　馬谷研究室

インタラクティブアプリケーション用の階層化された実行エンジン

山嶋 憂 202003805

1　はじめに

インタラクティブアプリケーションの実行エンジンの多くは，固定個の複数スケジューラ間でアクションのやり取りを行うが，あるスケジューリングキューに大量のアクションが積み重なると，一定間隔で実行すべきアクションを実行できなくなり，アプリケーション全体の動作が遅延する．本研究では，処理頻度の異なる複数のスケジューラを階層的に構成し，更にアプリケーション実行中，その階層構造を変化させることにより，処理頻度のきめ細かな調整を行える新しい実行エンジンを提案する．これにより，例えばあるスケジューラのキューが溢れた際，そのスケジューラの実行頻度を一括して下げることにより，遅延に対し柔軟に対応できるようになる．

2　背景

インタラクティブなプログラムにおけるスケジューラは，処理すべきアクションを優先順位に従い管理・制御する役割を担っている．実際，先行研究 [1][2][3] では，実行したい動作の種類や性質によってアクションを分類し，それぞれに適したスケジューリングを行うことで，アプリケーションの実行の最適化等を図っている．実行エンジン中のスケジューラが正しく機能していないと，スケジューリングキューからアクションが溢れることで処理の遅延が起こり，その状態が続けば最悪アプリケーションが停止してしまう恐れもある．また，スケジューラが複数あったとしても，一つのスケジューラのキューが溢れてしまうと，アプリケーション全体のパフォーマンスが低下することにもなる．これらを未然に防ぐためには，個々のスケジューラの処理能力を把握し，溢れる前にスケジューラ間でアクションを分散させるといった柔軟性が必要である．そこで本研究では，スケジューラを階層化し，アクションの優先順位に応じた処理頻度の変更を一括して行えるようにすることで，実行エンジンの柔軟性を向上させる．

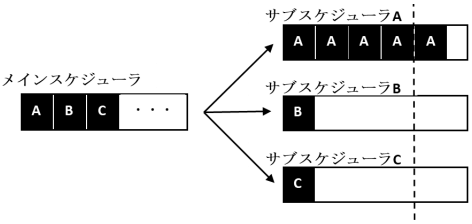

図 1　非階層的なスケジューラ

3　非階層的なスケジューラ

本研究の対象とするインタラクティブアプリケーションのスケジューラは，次の 2 種類からなるものとする．一つはユーザが指定した優先順位に従ってアクションを振り分けるメインスケジューラ，もう一つはその優先順位に対応した頻度で実際の処理を行うサブスケジューラである．例えば，図 1 は優先順位の異なる 3 種類のアクション (優先順位の高い方から順に A，B，C とする) を，メインスケジューラによって 3 つのサブスケジューラに振り分け，実行している様子を表している．今回の JavaScript 上の実装では，アクションクラスは次のように定義されている．

```
class Action {
    constructor(priority){
        this.priority = priority;
    }
    action(){
        // 実行したいコード
    }
}
```

`priority` メンバがこのアクションの優先順位を示す．スケジューラは，このアクションを実行する際，`action()` メソッドを呼び出す．図 1 中の点線は個々のスケジューラの処理能力の限界を示しており，点線を超えてアクションが積み重なると，超過したアクションの実行が著しく遅延することになる．また，限界を超えてこのスケジューラが動作しているため，アプリケーショ

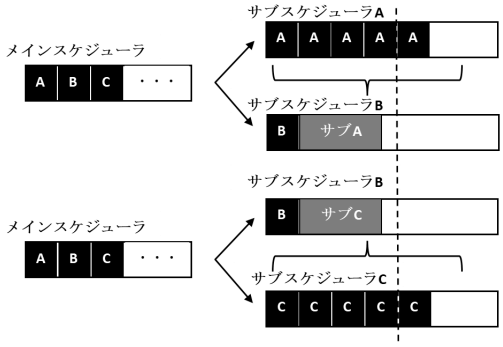

図 2　階層的なスケジューラ

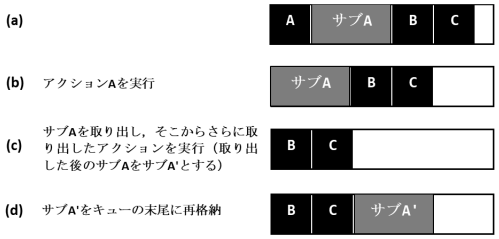

図 3　階層化されたスケジューラの実行手順

ン全体のパフォーマンスにも悪影響を及ぼしている可能性がある．

超過したアクションの実行の遅延に関しては，非階層的であるサブスケジューラでは，点線を超えてアクションが積み重なったとしても，キューの左から順にアクションを取り出すことしかできないため，右側に積まれたアクションの遅延は改善されない．それに加え，アクションを処理している間にもメインスケジューラからの振り分けが行われ続けるため，遅延は大きくなる一方である．

4　階層化されたスケジューラ

我々の提案する階層化されたスケジューラでは，点線を超えてアクションが積まれたことが検知されると，より頻度の低いスケジューラのキューに，そのスケジューラ自体を格納することができる (図 2 のサブスケジューラ B のキューに含まれているサブ A)．逆に，実行頻度の低いスケジューラのキューが溢れた場合，より頻度の高いスケジューラへの格納も可能である (図 2 のサブスケジューラ B のキューに含まれているサブ C)．両方を行えることにより，アプリケーション全体のパフォーマンスの低下と遅延の解消のどちらも回避することができる．

スケジューラのキューから取り出したものが別のスケジューラだった場合, 図 3 の手順に従い, 自身のキュー中のアクションと，格納されているスケジューラのキュー中のアクションを交互に処理する．図 3 の手順を繰り返すことにより，最終的に取り出したスケジューラが空になったら，そのスケジューラを階層構造から取り外して単独での動作を再開させ，以降，メインスケジューラからのアクションを直接受け取るようにする．

5　まとめ

本研究では，インタラクティブアプリケーションのスケジューラを階層構造にすることにより，大量のアクションの積み重なりによる動作遅延やアプリケーション全体のパフォーマンス低下への柔軟な対応を可能にした．

参考文献

[1] Jason Nieh and Monica S. Lam. *The design, implementation and evaluation of SMART: a scheduler for multimedia applications* In Proceedings of the sixteenth ACM symposium on Operating systems principles (SOSP '97), 184–197, 1997.

[2] Priscila Vriesman Arujo, Carlos Alberto Maziero, Júlio Cesar Nievola. *Automatic Classification of Processes in a General-Purpose Operating System* Brazilian Symposium on Computing System Engineering, 2011.

[3] Haoqiang Zheng and Jason Nieh. *RSIO: automatic user interaction detection and scheduling* In Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '10), 263–274, 2010.