B-008

# マルチコア環境における優先度継承条件の細分化による優先度逆転とオーバヘッドの抑制

# Priority Inversion Prevention and Overhead Reduction with Priority Inheritance Condition Segmentalization in Multi-core Environment

鴨生悠冬 † 佐藤将也 † 山内利宏 † 谷口秀夫 †
Yuuto Kamou Masaya Sato Toshihiro Yamauchi Hideo Taniguchi

## 1. はじめに

マルチコアプロセッサの普及により，スケジューラには複数のコアを考慮した優先度逆転の抑制が求められる．優先度逆転とは，低優先度のプロセスがプロセッサを解放するまで高優先度のプロセスや高優先度プロセスから処理依頼を受けたプロセスが処理を実行できず，2 つのサービスの優先度が逆転する状態である．特に，マイクロカーネル OS では，OS 処理も OS サーバとしてプロセスで動作するため，優先度逆転の抑制が重要である [1][2]．また，コア毎に独立したプロセススケジューラは，コア間の排他制御を行わないためオーバヘッドが小さいものの，優先度逆転の抑制には，コア間通信を必要とする．

したがって，コア毎に独立したプロセススケジューラをもつマイクロカーネル OS には，優先度逆転時間の抑制（要求 1）とコア間通信オーバヘッドの抑制（要求 2）が要求される．しかし，優先度逆転時間とコア間通信オーバヘッドはトレードオフの関係にある．本稿では，このトレードオフの関係を考慮し，優先度逆転時間とコア間通信オーバヘッドの両方を抑制する優先度継承方式を提案する．

## 2. 優先度継承方式

優先度継承とは，依頼先 OS サーバの優先度（$OS$）を依頼元応用プログラムのプロセス（AP プロセス）の優先度（$AP_r$）に変更することである．依頼先 OS サーバは，$OS < AP_r$ の場合に優先度継承を行うことで，登録されている依頼の内，最も高い依頼元 AP プロセスの優先度で実行でき，OS サーバより高く依頼元 AP プロセスより低い優先度をもつ AP プロセスの妨害によって生じる優先度逆転を抑制できる．

異なるコア間で走行するプロセス間（AP プロセスと OS サーバ間）における処理依頼の処理流れを図 1 に示す．依頼元コアでは依頼登録後，コア間通信要否判定処理を行う．コア間通信を行う場合，依頼先コアに IPI を送信し，結果を待つ．コア間通信を行わない場合，結果を待つ．依頼先コアでは IPI を受信後，優先度継承と依頼先 OS サーバの起床を行う．

コア間通信を行う条件を細分化すると，図 1 のコア間通信可否処理として，3 つの処理流れが考えられる．これらを図 2 に示し以下に説明する．

（A）優先度逆転を最も抑制したい場合，必ずコア間通信を行い，優先度継承を行う（常時継承方式）．この方式では，最も優先度逆転を抑制できるものの，依頼登録時に必ずコア間通信オーバヘッドが生じる．Intel Xeon E5-2630(8Core, 2.4GHz) において，このオーバヘッドは約 2 $\mu s$ である．コア間通信オーバヘッドを含んだ処理依頼の処理時間は約 2.5 $\mu s$ であり，コア間通

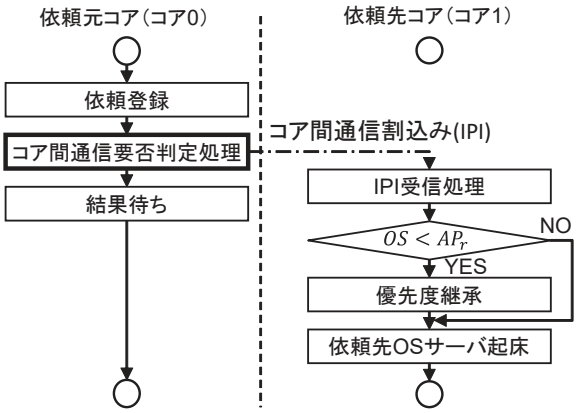


図 1 処理依頼の処理流れ

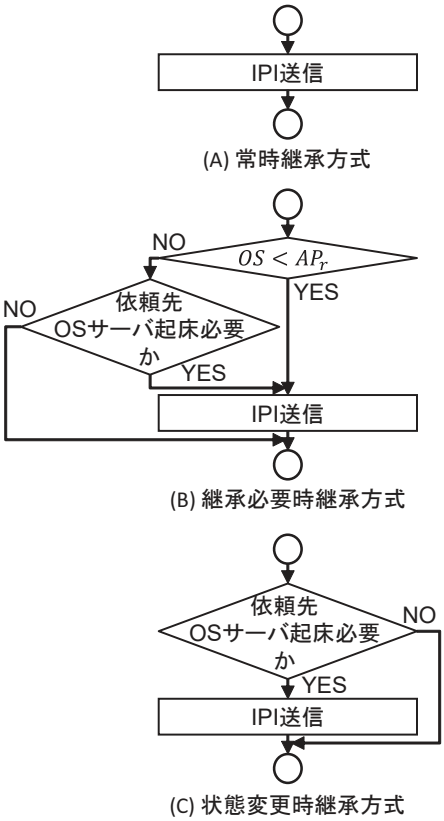


図 2 コア間通信要否判定処理

† 岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University

表 1 優先度逆転時間とコア間通信オーバヘッドの関係

| 通番 | 条件 1 | 条件 2 | 逆転条件 | 常時継承方式 | | 継承必要時継承方式 | | 状態変更時継承方式 | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | $t_{inv}$ | コア間通信 | $t_{inv}$ | コア間通信 | $t_{inv}$ | コア間通信 |
| (1) | 偽 | 偽 | 偽 | 0 | 1 回 | 0 | 0 回 | 0 | 0 回 |
| (2) | | 真 | 偽 | | | | 1 回 | | |
| (3) | | | 真 | | | | | $> 0$ | |
| (4) | 真 | 偽/真 | 偽/真 | | | | | 0 | 1 回 |

信オーバヘッドが約 80%を占める．

（B）優先度継承するか否かを判定し，コア間通信オーバヘッドを削減する (継承必要時継承方式)．具体的には，依頼先 OS サーバの起床必要（条件 1）または $OS < AP$（条件 2）を満たす場合のみ，コア間通信を行う．この方式は，依頼元コア上で依頼先コア上の OS サーバの優先度を用いて優先度継承の判定を行うため，各コア上の独立したスケジュール動作によって，優先度継承の判定後に依頼先 OS サーバの優先度が低下し，優先度逆転が生じる可能性や優先度が上昇し継承が不要になったにもかかわらず，コア間通信する可能性がある．

（C）上記の（条件 1）を満たす場合のみ，コア間通信を行う（状態変更時継承方式）．この方式は，「依頼登録時に依頼元 AP プロセスより低優先度の AP プロセスが依頼先コア上で RUN/READY 状態でないことがシステム上明らかである場合，依頼登録時に継承する必要はないこと」に着目した方式である．この方式は，コア間通信オーバヘッドを最も抑制できる．

## 3. 比較

優先度継承方式を以下の 2 つの観点から比較する．

(観点 1) 優先度逆転時間

(観点 2) コア間通信オーバヘッド

各方式における処理流れの差は依頼登録処理にのみ存在し，（条件 1）を満たさない，かつ（条件 2）を満たす場合と満たさない場合において差が生じる．なお，（条件 1）を満たす場合，差は生じない．それぞれの場合における，コア間通信回数を調べ，（観点 2）について比較する．さらに，優先度逆転が生じる場合と生じない場合についても場合分けすることで，（観点 1）を比較する．優先度逆転が生じる条件（逆転条件）は，$OS \leqq AP_o < AP_r$ である．なお，$AP_o$ は依頼先コア上の RUN/READY 状態の AP プロセスの優先度である．

表 1 に 3 つの方式の優先度逆転時間 ($t_{inv}$) とコア間通信回数の関係を示し，以下に説明する．

表 1 の（1）が成り立つ条件では，常時継承方式のみコア間通信を行い，優先度継承処理を行う．したがって，常時継承方式のオーバヘッドは他の方式に比べ，1 回のコア間通信分だけ大きい．一方，他の方式では優先度継承を行っていないため，依頼登録後から依頼先 OS サーバの依頼取得までの間に優先度逆転が生じる可能性がある．

表 1 の（2）が成り立つ条件では，継承必要時継承方式と常時継承方式はコア間通信を行い，優先度継承処理を行う．したがって，継承必要時継承方式と常時継承方式のオーバヘッドは，状態変更時継承方式に比べ，1 回のコア間通信分だけ大きい．一方，状態変更時継承方式では優先度継承を行っていないため，依頼登録後から依頼先 OS サーバの依頼取得までの間に優先度逆転が生じる可能性がある．

表 1 の（3）が成り立つ条件では，継承必要時継承方式と常時継承方式はコア間通信を行い，優先度継承処理を行う．したがって，継承必要時継承方式と常時継承方式のオーバヘッドは，状態変更時継承方式に比べ，1 回のコア間通信分だけ大きい．一方，状態変更時継承方式では優先度逆転が生じる．

表 1 の（4）が成り立つ条件では，全ての方式がコア間通信を行い，優先度継承処理を行う．したがって，各方式の優先度逆転時間とオーバヘッドの差はない．

上記に関して，以下に考察する．

（考察 1）(1) である場合が多い環境，すなわち，OS サーバへの依頼頻度が低く，異なる優先度のプロセスによる OS サーバへの依頼が少ない環境において，継承必要時継承方式は，コア間通信オーバヘッドの観点から，常時継承方式より有利である．ただし，依頼登録後から依頼取得までの間に優先度逆転が生じる可能性があることに留意する必要がある．

（考察 2）(3) より (2) である場合が多い環境，すなわち，OS サーバと同一コア上で AP プロセスがほとんど実行されない場合，状態変更時継承方式は，コア間通信オーバヘッドの観点から，継承必要時継承方式より有利である．一方，(2) より (3) である場合が多い環境において，継承必要時継承方式は，優先度逆転時間の観点から，状態変更時継承方式より有利である．

上記からどの方式が最も優れているかは，動作するシステムによって異なることが分かる．

## 4. おわりに

本稿では，コア毎に独立したスケジューラをもつマイクロカーネル OS において優先度逆転を抑制する 3 つの優先度継承方式を提案した．

また，各方式を比較し，優先度逆転時間とコア間通信オーバヘッドの関係を明らかにし，どの環境においてどの方式が優れているのかを明らかにした．

## 参考文献

[1] Kitayama, T., Nakajima, T., and Tokuda, H., “ RT-IPC: An IPC Extension for Real-Time Mach,” USENIX Microkernels and Other Kernel Architectures Symposium, Vol.39, No.9, pp.91–104 (1993).

[2] 鴨生 悠冬, 山内 利宏, 谷口 秀夫, “ マルチコア環境における優先度逆転を抑制する ***AnT*** オペレーティングシステムのスケジュール機構,” 情報処理学会研究報告, Vol.2016-OS-136, No.17, pp.1–8 (2016).