SPUMONE利用環境でのLock Holder Preemptionの回避方法の検討と実装

三嶽仁　　巻島一雄　　杵渕雄樹　　中島達夫
早稲田大学 理工学部 コンピュータ・ネットワーク工学科

1 はじめに

主にエンジニアリングコストの削減等を目的として，仮想化技術は組込み機器にとっても重要な技術となりつつある．だが，仮想化環境は物理的なハードウェアを対象として設計，実装された既存のオペレーティングシステムの前提を常に提供可能とは限らず，Semantic Gapによる予期せぬ弊害を発生させる危険性ももたらす．例えば，リアルタイムOSと汎用OSを一つのハードウェア上で実行させた場合，汎用OS側のパフォーマンスが大きく低下することが先行研究で判明している．

本研究では，準仮想化技術SPUMONE[3]を利用した環境での，弊害の代表的な1つであるLock Holder Preemption(LHP)の回避方法を提案する．RTOSにはTOPPERS/JSPを，汎用OSにはLinuxを利用した．

LHPは，ロックを獲得しているLinuxのスレッドを実行している仮想CPUがRTOSの仮想CPUに物理CPUを奪われることで発生する．回避方法には，SPUMONEの仮想CPUマイグレーション機能を用いる．仮想CPUマイグレーション機能は，物理CPUとその上で実行される仮想CPUを，起動後も動的に変更可能にするものである．RTOSの仮想CPUが物理CPUにスケジュールされる際，その物理CPU上で実行されている仮想CPUがロックを獲得中のスレッドを実行中であれば，別の物理CPUにRTOSの仮想CPUをマイグレーションするという方式で実装を行った．

2 関連研究

Uhligらは，準仮想化と完全仮想化の両方の方式でLHPを回避する方法を提案，実装している[1]．

この論文の提案する準仮想化環境でのLHPの回避方法は，ゲストOSを修正し，ロックを獲得していることをVM側から知ることが出来るようにするというものである．VMがゲストOSのスケジューリングを行う際，その情報に基づいて，プリエンプションを遅延させることによりLHPを防ぐ．

完全仮想化環境でのLHPの回避方法は，カーネル空間ではほぼ全ての時間1つ以上のロックが獲得されているという経験則と，IA-32プロセッサではHLT命令を実行した後は，割り込みを受け取るまではアイドルループを実行しているという性質を利用する．カーネル空間とユーザ空間のスイッチやHLT命令，割り込みはVMからモニタリング可能なので，それらの情報に基づいてゲストOSがプリエンプト可能か不可能かを判断し，必要に応じてプリエンプトを遅延しスケジューリングを行う．

Wellsらは，プロセッサを拡張し，ハードウェア的にスピンロックの獲得を監視可能にし，その情報をLHPの回避に利用する方法を提案，実装している[2]．一定の時間内に実行された命令数が閾値以下であればゲストOSはスピンロックを獲得していると判断する，という機能をプロセッサに実装し，ゲストOSの修正を全く行わないでロックの獲得状況を把握している．

3 提案方式

本研究では，仮想マシンモニタSPUMONEから，Linuxのスレッドがプリエンプト可能か否かを把握し，RTOSをディスパッチするコアを動的に変更することで可能な限りLHPを回避する方法を提案する．

仮想CPUマイグレーション機能は，rtos_cpu_affinityというビットマスクとして利用される変数でRTOSの仮想CPUが実行可能な物理CPUを表現する．本研究では実験用に4コアのマシンを利用したので，下位4ビットのみが問題となる．全てのコアがRTOSを実行可能であれば，下位4ビットは1111となる．コア0，コア1のみが実行可能であれば，0011となる．Linuxの全てのコアでロックが獲得されている場合は，0000となる．

Linuxカーネルはビルド時にCONFIG_PREEMPTオプションを有効にすると，preempt_enable()とpreempt_disable()という2つのマクロが意味あるものとして定義される．これは，現在実行されているスレッドがプリエンプト可能か不可能かを識別するための情報を変更するマクロである．例えば，スピンロックを獲得しているスレッドは，プリエンプトされるとLHPのように無駄なビジーループを他のコアで発生させる原因になる．そのため，カーネル空間で実行中のスレッドのプリエンプトを許可するCONFIG_PREEMPTオプションが有効にされている場合でも，preempt_countが1以上の場合は，プリエンプトは発生しない．このpreempt_enable()とpreempt_disable()に変更を加え，preempt_countが0から1になった場合にrtos_cpu_affinityから実行中のコアのフ

ラグをクリアしてプリエンプトの回避を SPUMONE に依頼し，preempt_count が 1 から 0 になった場合にフラグをセットしてプリエンプト可能であることを SPUMONE に伝える．

Linux の全ての仮想 CPU でプリエンプト不可能な状態のスレッドが実行されている場合は，RTOS が直前にスケジュールされたコアの Linux の仮想 CPU をプリエンプトする．この場合は LHP が発生してしまうが，RTOS のリアルタイム性を保証するための措置である．

4 評価と考察

評価環境として用いたハードウェアの仕様を表 1 に示す．

表 1: 評価環境

ターゲットボード	RP1 評価ボード
CPU アーキテクチャ	SH4A
CPU	600MHz × 4
メモリ	128MB

評価用アプリケーションには，Rusty Russell 氏の作成したスケジューラ用ベンチマークプログラム hackbench を，より現実的なワークロードを表現するように修正したものを利用した．変更点は，組込み機器には過剰と思われる 1 グループ 40 個の IPC 用プロセスを 4 個に減らし，無駄なメモリコピーをそれぞれのプロセスで行わせることでユーザランドでの実行時間を増加させたという 2 点である．

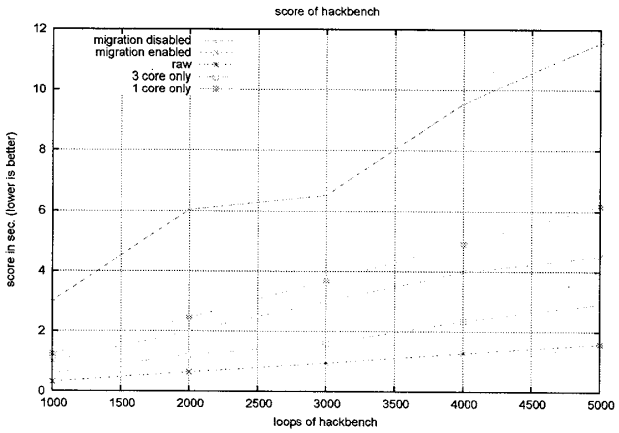

図 1: hackbench のスコア

評価結果を図 1 に示す．図中の migration disabled という折れ線は本研究による変更前のパフォーマンスのスコアである．migration enabled という折れ線は本研究による変更後のスコアである．raw という折れ線は，RTOS が時間を消費しない状況で，hackbench が 4 コア全てを利用した状況でのスコアである．3 core only という折れ線は，RTOS が時間を消費しない状況で，sched_setaffinity() を利用して hackbench が利用するコアを 3 個に制限した状況でのスコアである．1 core only という折れ線は，3 core only と同じ条件で hackbench が利用するコアを 1 個に制限した状況でのスコアである．変更前，変更後では，RTOS が 90%の CPU 時間 (500ms 周期で 450ms) を消費している．

本研究による変更を加える前では，このように RTOS が高負荷な状況になると，Linux のパフォーマンスは 1 コアのみを利用した時よりもさらに低くなる．変更を加えることで，2 倍以上にスコアは改善されるが，それでも依然として理想である 3.1 コア分のスコアには届いていない．

5 まとめ

本研究では，preempt_enable() と preempt_disable() に変更を加えることで，SPUMONE 利用環境での LHP を可能な限り回避し，Linux の性能の向上を達成した．

ベンチマークによるスコアは向上が見られたが，それでも依然として理想とするスコアとの間には差がある．これは，RTOS のリアルタイム性を保証するため，Linux の全ての仮想 CPU がプリエンプト不可能だった場合には LHP の発生を許可し，RTOS の実行を優先させているからだと考えられる．

参考文献

[1] Volkmar Uhlig, Joshua LeVasseur, Espen Skoglund, and Uwe Dannowski. Towards Scalable Multiprocessor Virtual Machines. *In Proceedings of the 3rd Virtual Machine Research & Technology Symposium (VM'04)*

[2] Philip M. Wells, Koushik Chakraborty, and Gurindar S. Sohi. Hardware Support for Spin Management in Overcommitted Virtual Machines. *15th International Conference on Parallel Architectures and Compilation Techniques (PACT ' 06)*

[3] Yuki Kinebuchi, Takushi Morita, Kazuo Makijima, Midori Sugaya and Tatsuo Nakajima. Constructing a Multi-OS Platform with Minimal Engineering Cost *In Analysis, Architectures and Modelling of Embedded Systems: Third IFIP TC 10 International Embedded Systems Symposium (IESS 2009)*