SPUMONE仮想環境下におけるリカバリ指向OS向けOS間通信の実装

福島拓†　石井正将†　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 分散ユビキタスコンピューティング研究室‡

1 はじめに

我々の研究室では，新たなOSアーテクチャを考案すると共に，その実装であるSPUMONEと呼ばれるシステムを開発している[1, 2].
SPUMONEは組込みシステムにおける高パフォーマンス，信頼性，可用性，保守性，リアルタイム性を得るために，ハードウェア層とOSの間に仮想化層を設けてCPU仮想レイヤーを提供する．SPUMONEは，単一のCPUを複数個のCPUがあるように見せかけるもので，仮想CPUをゲストOSに提供することで，同時に複数のOSを動作させることを可能にしている．本研究ではSPUMONE上で複数のゲストOSを協調して動作させる際に必要となるOS間での通信(IOC: InterOS Communication)について考察する．また，近年の高可用性を求める動きに合わせ我々の研究室では問題が起きたときに高速に再起動することで問題を解決するリカバリ指向OSのテストを行なっており，そのためにSimpleOSという独自のOSを開発している．本研究ではSPUMONE上でゲストOSとしてLinuxとSimpleOSを動作させ，故障したと仮定してLinux側からSimpleOSを再起動できるようにすることも目標としている．

2 関連研究

SPUMONEとは異なるハイパーバイザ型の仮想化技術のひとつにXen[3]が挙げられる．Xen上で動作するOS間のデータ転送にはTCP/IPを用いた方式と共有メモリを用いた方式が存在する．TCP/IPを用いた方式は一般的なインターフェースであるため多くのOSで利用されやすいが，TCP/IPの処理によるオーバーヘッドが大きく高パフォーマンスを得ることができない．一方共有メモリを利用した方式は高速であるが，通信に関する規約を自分達で定義し実装なくてはならない．そこでコントロール情報の転送のみにTCP/IPを採用し，データの転送には共有メモリを利用することで高速なデータ転送を実現したXenSocket[4]やXwayl[5]が考案されている．さらに，Lamia YouseffらはXenのshared pageを改良してネイティブに共有メモリへアクセスできるようにすることによってハイパーコールの呼び出しを最小限に押さえてパフォーマンスを向上させることに成功している[6].

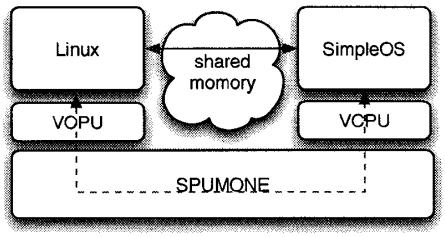

図1: SPUMONE上で動作するLinuxとSimpleOS間のIOC

3 設計と実装

本研究では組込みシステム向けにマルチOS環境を用いることを想定しており，リアルタイム性が求められているから，IOCの通信方式として動作が高速な共有メモリ方式を用いることにした．また，データの一貫性を保証する軽量なプロトコルを実装した．このように統一されたインターフェースを提供することでアプリケーション開発者はエンジニアリングコストを抑えることができる．

3.1 SimpleOS上でのIOCの設計

データの転送を通知する手段として，あるゲストOSがIOCを行なった際にSPUMONEが別のゲストOSに仮想的な割込みを発生させることを利用することにした．これによりIOCが行なわれるたびに割込みが発生し，即座にあるゲストOSから別のゲストOSへとIOCが行われたことを通知することができる．通知されたOSはあらかじめ割込みハンドラを用意しておくことによって共有メモリに書き込まれた内容を読みとることができる．

3.2 SimpleOS上でのIOCの実装

本研究ではLinuxが共有メモリに書き込みを行う際にSPUMONEのAPIを呼び出し，共有メモリへの書き込

†Taku FUKUSHIMA, Masaki ISHII, Kazuo MAKIJIMA, Yuki KINEBUCHI, Tatsuo NAKAJIMA
‡Department of Information and Computer Science, Faculty of Science and Engeneering, Waseda University

みを IPI* として SimpleOS へと伝える命令を SPUMONE に渡す．図 1 の破線がこの IPI 命令のパスに相当し，実線はデータのパスである．これを受けとった SPUMONE は SimpleOS に対し割込みを発生させる．すると SimpleOS は割込みハンドラへと処理を移し，Linux によって書き込みが行なわれた共有メモリを参照する．通常 Linux などで共有メモリの読み書きをする際には OS の標準ライブラリを利用するが，SimpleOS には標準ライブラリのようなものが存在しないのでそれらの必要な処理も実装した．

さらに，共有メモリの番地や大きさは SPUMONE に定義されており，IOC 用のディスクリプタテーブルが複数の ioc_mem_t という構造体で構成されている．それぞれの ioc_mem_t は，読み込み用と書き込み用の ioc_mem_fifo という配列で実装された円環状の FIFO 型データ構造を含んでいる．図 2 に共有メモリ構成の概略を示す．この FIFO 型のデータ構造への読み込みおよび書き込みは非同期に行なわれ，したがってそのままではデータの一貫性が保証されない．例えば，あるプロセス A があるメモリ領域を参照したところでコンテキストスイッチが起こり，別のプロセス B がその値を変更してしまった後で再びプロセス A へ制御が移った場合はそのメモリ領域の値はプロセス A が想定していないものとなってしまう．

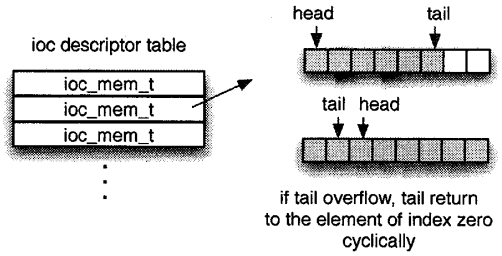

図 2: IOC に使用される共有メモリ構成の概略

4 考察

前章で掲示した共有メモリを用いた非同期通信時に発生する問題を ABA 問題という (図 3)．ABA 問題を回避し，データの一貫性を保つためにはセマフォを利用した排他制御や Lock-Free Queue [7] などを利用する方法がある．Lock-Free Queue は Fetch&Add(FAA) や Compare&Swap(CSW) などのアトミックに行なわれる Read-Modify-Write を利用して実装され，ロックを利用したものよりも高いパフォーマンスを実現することができる．アトミックな加算や減算はハードウェアによってサポートされている命令を使用し，SPUMONE が動作する SH4A アーキテクチャ上ではアセンブラの *movli* 命令および *movco* 命令で実現される．

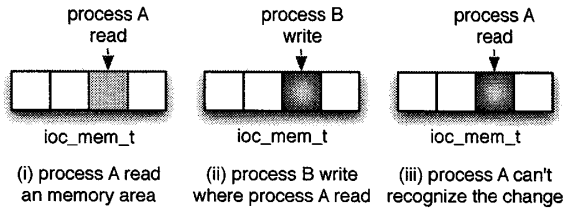

図 3: ABA 問題の例

5 まとめと今後の課題

SPUMONE 上で動作する独自 OS である SimpleOS 上に IOC を実装することに成功し，Linux と SimpleOS の間で通信をすることができるようになった．これにより Linux から SimpleOS を再起動させることが可能になり，例えば異常を検知したときに再起動させるモニタリングサービスのテストができるようになった．そういったモニタリングサービスは本研究の範囲外となるので別の機会に譲りたいと思う．また，IOC は IPC と非常に似ているため，IPC で行なわれている研究を応用しやすいと考えられる．例えばメッセージキューやメールボックス型の通信を行うことでより強固で柔軟性のある通信ができると考えられる．

参考文献

[1] Tatsuo Nakajima, Hiroo Ishikawa, Yuki Kinebuchi, Midori Sugaya, Sun Lei, Alexandre Courbot, Andrej van der Zee, Aleksi Aalto, and Know Ki Duk: An Operating System Architecture for Future Information Appliances (2008)

[2] Wataru Kanda, Yu Yumura, Yuki Kinebuchi, Kazuo Makijima: SPUMONE: Lightweight CPU Virtualization Layer for Embedded Systems

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield: Xen and the Art of Virtualization (2003)

[4] Xiaolan Zhang, Suzanne McIntosh, Pankaj Rohatgi, and John Linwood Griffin: XenSocket: A High-Throughput Interdomain Transport for Virtual Machines

[5] Xway: http://sourceforge.net/projects/xway/

[6] Lamia Youseff, Dmitrii Zagorodnov, and Rich Wolski: Inter-OS Communication on Highly Parallel Multi-Core Architecture

[7] Jhon D. Valois: Implementing Lock-Free Queues (1994)

*IPI: InterProcessor Interupption