組込みOSにおける分散共有メモリの研究

松原　裕人†　　山原　亨†　　大谷　真†
湘南工科大学†

1. はじめに

組込み機器の高性能・高機能化に伴い自動車の運転制御や人工衛星内のシステムといった複数の組込みシステムが連動して動作するシステムが増加している。この様なシステムを効率的に開発・運用するためには複数の機器を透過的に扱える分散共有メモリ(distributed shared memory、DSM)が必須と言える。しかし、現時点では分散共有メモリ機能をサポートした実用的な組込みOSは殆ど無い。これを解決するため、分散共有メモリミドルウェア(Distributed Shared Memory Middleware:DSMM)及び高速なリモートメモリアクセスを行う通信デバイスドライバ(Remote Memory Access Driver:RMAD)の研究開発を行った。本論文ではこの分散共有メモリミドルウェア部分について実現方式を提案し、T-Kernel上での設計・実装を通してその評価を述べる。組込み機器における分散共有メモリ実装例は殆ど無く、特にT-Kernelにおける実装は現在の調査では初めてである。

2. 研究背景

今後、半導体の高性能化と組込みシステムの領域の拡大に伴い、複数の組込み機器が連動するシステムの増加が予想される。この課題に対応するためフリーな実装の分散共有メモリをもつミドルウェアの研究及び開発行う事とした。

2.1. 分散共有メモリの現状

現在、存在する分散共有技術の調査結果の一部と本研究の位置づけを下記の表1. に示す。

表1. 主な分散共有技術

	透過性	実装	例	適用範囲
分散共有技術	透過的	OS(ページング)	本研究	LAN(not TCP/IP)
		ハードウェア	SMP	ローカルバスI/O
		OS(ミドルウェア)	VxWorks	ローカルIPC
	非透過的	コンパイラ	LINDA	TCP/IP
			OpenMP	並列計算機、PCクラスタ
		プログラミングライブラリ	CORBA	WAN(分散オブジェクト)

現在、分散共有メモリをサポートした商用組込みOSはVxWorksのみである。本研究の目標では複雑なシステムを開発する上ではプログラミングの透過性やロジックの単純さが重要となると考え、OS内での仮想記憶アーキテクチャの延長としての実装を選択した。また、適用範囲としてはLANの様なある程度の大きさはあるが全体としては限定されるネットワークを想定する事とした。

Study on Distributed Shared Memory for embedded OS
Hiroto MATSUBARA†, Tooru YAMAHARA†, Makoto OYA†
Shonan Institute of Technology†

オープンソース・オープン仕様であり、ミドルウェアの流通を重視して設計・開発されおり、ユーザはサブシステムと呼ばれるOSモジュールを自由に追加する事が出来る。以上の2点からカーネルにメモリ管理と言う比較的複雑なサブシステムを追加する事が容易であろうと判断して選択した。

3. 設計

3.1. 分散共有メモリの実現方式

DSMの設計上、以下の課題があった。

a. メモリ実体の保証
b. メモリ一貫性の保証
c. 外部仕様の決定
d. 応答性能の確保
e. リモートメモリの参照方式

本論文では紙面の都合上a, bについて述べる。

3.2. メモリ実体の保証

組込み機器にはスワップアウトデバイスがあるとは限らず、ページリプレイスメントに失敗する可能性がある。これは図1. の様に仮想ページの有効/無効に変わらず常に1対1で物理メモリを割り当てておく方式を採用する事によって解決した。本方式は不必要なメモリを常に確保しており、リソース的観点では無駄が多い、しかし、必要最小限しか全体で確保しない方式を採用した場合に生じる作業量や複雑さとのトレードオフを考慮して本方式を選択した。

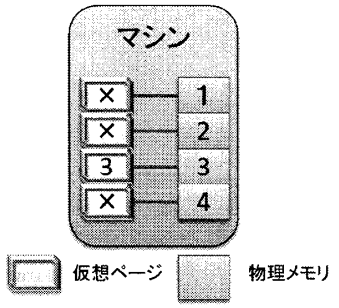

図1. メモリ実体保証

3.3. メモリ一貫性の保証

複数の機器からDSMに同時にアクセスがあった場合にもデータの一貫性が保証されなければならない。今回の場合には図.2の様に多数の機器があっても同じ仮想アドレスが同時に有効になっている数は1(データ移動時には0)とすることで一貫性を保つ方式を採用した。

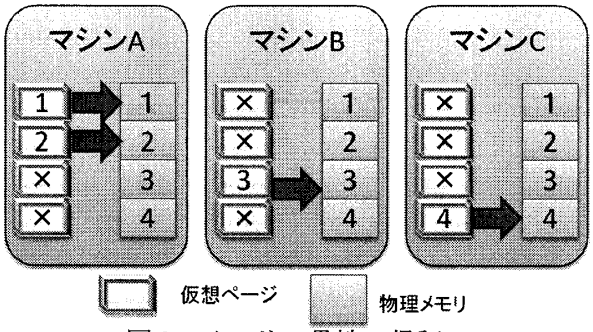

図2. メモリ一貫性の保証

4. 実装

4.1. 実装方針

本論文ではページングとMMUを用いてDSMを実現するため、MMUに対応しているT-Kernel Standard Extension(T-Kernel/SE)を使用し、T-Kernel/SEのメモリ管理拡張の一部としてDSMを実装した。これはT-Kernel/SEに追加する部分最小限に抑えるためである。

4.2. プログラム構成

プログラム構成は下記の図3.の様になった。

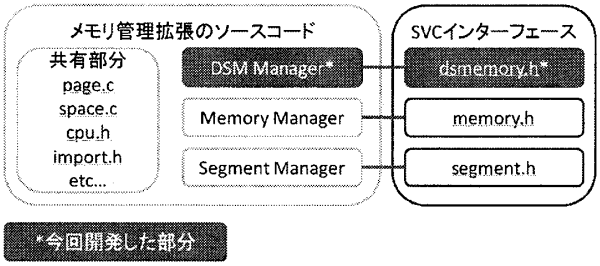

図3. プログラム構成

メモリ管理拡張の一部として実装されるためオブジェクトはメモリ管理拡張にリンクされる。しかし、T-Kernelのサブシステム管理機構を利用し、内部的には別のサブシステムとして動作し、独立したSVCとして実装した。

4.3. 実装方式

ここでは特に分散共有メモリを確保する場合と他の機器にあるメモリを参照する場合について述べる。

4.3.1. 分散共有メモリの確保

DSMの確保を行う場合には当該機器だけではなく、他の機器にも情報を送信しメモリを確保する必要がある。その為、メモリ確保動作は以下の通りの実装となった。

①自機器の物理/仮想メモリの確保
tkse_get_mbk関数を呼び出し本サブシステムインターフェースをコールしたプロセス固有空間での物理/仮想メモリを確保する。

②LSID*1、物理アドレスの取得
GetLSID_tidによりLSIDを取得、PageTableEntryとLSIDを元に仮想アドレスを物理アドレスに変換する。

③他の機器にDSM情報を送信
RMADのopen関数を呼びだし、他の機器にDSMの物理アドレス、領域サイズ、機器のユニークIDを送信する。

*1 LocalSpaceID(タスク固有メモリ空間のID)

4.3.2. 分散共有メモリの参照

無効な仮想アドレスにアクセスするとページフォールト割込みが起きる。DSMMではあえて仮想アドレスを無効にする。この時に呼び出されるページフォールトハンドラを変更し、RMADを通して他の機器からデータを取得する事によって実装した。

具体的にはPageTableEntry(PTE)構造体のPresent(p)ビットが0となっている状態を表す。この場合、DataAbort割り込みが発生し、excmgr.c内のPageFaultHdr関数が呼ばれる。

このPageFaultHdr関数を変更する事によってDSMを実現した。特定のLSIDかつアドレスでDataAbortが起きた時に、現在研究開発を行っているRMADを呼び出し、PTEのPFA(PageFrameAddress)に対応する物理ページにメモリを1ページ分コピーする。その後、PTEのpビットを1に変更しページを有効化、割り込みを終了する。これによって　他の機器からのメモリコピーを透過的に実現した。

5. 評価

DSMMとRMADは開発中であり性能測定の段階まで完成させることが出来なかった。現在、測定が終了しているRMADの通信の速度は100BASE-TXのLANボードを使用した場合27[Mbps]であり、更なる性能の向上が必要である。今回報告できなかったDSMMとDSMM+RMADの性能評価については発表で報告する。

6. 考察

今回の方式には3.2.の課題や3.3.の方式では同じ仮想アドレスに対して複数の機器からのアクセスが続くとページファイルのスラッシングと同様の現象を引き起こすと言う明らかな課題がある。しかし、今回は実装可能性の検証のためにより簡便な方式を採用した。今後はこの様な省リソースや高速化と言った問題を解決していく必要がある。方法としては機器間での積極的な空きページの交換による全体物理メモリ使用量の少ない実体保証、より緩い一貫性保証と言った方法があると考えられる。

7. まとめ

T-Kernelにおける分散共有メモリの実装可能性を確認した。

今後は本ミドルウェアの性能向上と本ミドルウェアを利用したアプリケーションの両側面から研究を進める。また、OSやハードウェアの依存部と分離する事による移植性の向上とT-Kernelミドルウェアとしてのリリースも検討していく。

参考文献

1. Andrew S. Tanenbaum, Maarten van Steen (2003)『DISTRIBUTED SYSTEM Principles and Paradigms』PEARSON Education

2. T-Engine Forum (2005)『T-Kernel 仕様書』URL:http://www.t-engine.org/japanese/text/TEF020-S001-01.00.00_ja.pdf

3. T-Engine Forum (2008)『T-Kernel Standard Extension 仕様書』URL:http://www.t-engine.org/japanese/text/TEF020-S006-01.00.02_ja.pdf

4. SH/M32R T-Engine Home Page『T-Kernel/Standard Extensionの新規システムへの実装ガイド(2) -メモリ管理とアドレス変換-』URL:http://www.superh-tkernel.org/jpn/documents/archive/TKSEportingguide_mmu_1.00.03.pdf