2004－OS－95　(6)
2004／2／26

分散オペレーティングシステムSolelcにおける メモリ操作のトランザクション化による耐障害性向上手法

玉田 純子 †　水口 孝夫 †　瀧本 栄二 †
芝 公仁 ††　毛利 公一 †††　大久保 英嗣 †††

† 立命館大学大学院理工学研究科
†† 龍谷大学理工学部　††† 立命館大学理工学部

Solelc では, オペレーティングシステムを抽象化層とカーネルから構成している. 抽象化層は, 個々の計算機上で動作し, カーネルが動作する環境を提供する. カーネルは, 抽象化層の提供する機能を使用し, すべての計算機上の資源を管理する. このような構成手法によって, カーネル自体が位置透過に動作可能となり, 単一のカーネルよって複数の計算機上のすべての資源を管理することが可能となる. しかし, 単一のカーネルが複数の計算機上で動作するため, 計算機が 1 台でも停止するとシステム全体に影響が及ぶことになる. システムの信頼性と可用性を向上させるためには, 耐障害性のための機能が不可欠となる. 本稿では, メモリ操作のトランザクション化による耐障害性向上手法を提案する. 本手法は, Solelc が管理するすべての計算機によって共有される仮想空間上のメモリの冗長性を利用するものである.

A Method for Improving Fault-tolerance by Transactional Memory Operations in the Solelc Distributed Operating System

Atsuko Tamada †　Takao Mizuguchi †　Eiji Takimoto †
Masahito Shiba ††　Koichi Mouri †††　Eiji Okubo †††

† Graduate School of Science and Engineering, Ritsumeikan University
†† Faculty of Science and Technology, Ryukoku University
††† Faculty of Science and Engineering, Ritsumeikan University

In Solelc, the operating system consists of abstraction layers and a kernel. The abstraction layers work on each computer and provide the environment which the kernel works. The kernel uses functions provided by the abstraction layers and manages the resources of all computers. By this construction method, the kernel itself works in a location-transparent fashion, and it becomes possible to manage all resources on multiple computers with a single kernel. However, since a single kernel works on multiple computers, the entire system might stop when a certain computer breaks down. In order to improve the reliability and availability of the system, it is necessary to provide functions for fault-tolerance. In this paper, we propose a method for improving fault-tolerance by transactional memory operations. This method utilizes the memory redundancy on the virtual space shared by all computers which Solelc manages.

1　はじめに

分散オペレーティングシステム Solelc [1, 2] は, 位置透過な資源管理を行い, 複数の計算機が持つ資源を効率的に利用することを目的としている. これらを実現するために, Solelc では, 単一のカーネルによってシステムが持つすべての計算機を管理している.

このようなカーネルを動作させるために, Solelc では, 各計算機上で計算機資源を抽象化する機構を動作させ, 位置透過に資源管理を行うことを可能とする環境を構築している. この環境上でカーネルを動作させることによって, カーネルは位置透過に動作することが可能となり, 複数の計算機を一括して管理することができる.

すべての計算機が単一のカーネルによって管理されると, カーネルがプロセスに提供する機能は, すべての計算機で同一のものとなる. そのため, カーネルが提供するすべてのサービスを任意の計算機上で使用することができる. また, すべての計算機で同一のカーネルを使用することから, プロセスは資源に対する操作も位置透過に行うことが可能である. すなわち, 計算機資源の抽象化によって, カーネルと同様にプロセスの位置透過性も実現される.

しかし, Solelc には, 計算機が 1 台でも停止するとシステム全体に影響が及ぶという問題がある. この原因は, 単一のカーネルが複数の計算機にまたがって動作するため, カーネルの一部の機能が失われるためである. したがって, 局所的な障害によるシステム全体への影響を回避し, システムの信頼性と可用性を向上させるためには, 障害に対応する機能の実現が不可欠である.

そこで, Solelc が管理するすべての計算機によって共有される仮想アドレス空間上のメモリに冗長性があることに着目した. Solelc のメモリ管理における特性から, 1 台の計算機が停止した場合も, 該当計算機によって保持されていたメモリ内容と同じデータやコードが他の計算機に存在する. すなわち, 動作する計算機上のコードを利用して失われたカーネルの機能を復元させることや, アクセスする対象のデータを切替えることでシステムを継続して動作させることが可能である. 以上より, 冗長なメモリ領域を利用し, 耐障害性向上のための手法として, メモリ操作のトランザクション化を提案する.

以下, 本稿では, 2 章で Solelc の概要, 3 章で耐障害性向上手法, 4 章でメモリ操作のトランザクション化方式について述べ, 5 章で本研究の評価を示す.

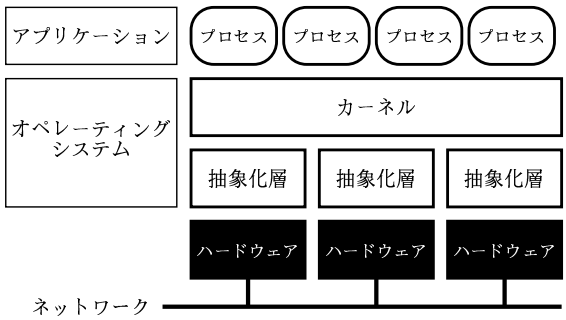


図 1　Solelc のシステム構成

2　Solelc の概要

2.1　システム構成

Solelc のシステム構成を図 1 に示す. Solelc では, オペレーティングシステムをカーネルと抽象化層の 2 層に階層化している. 抽象化層は, すべての計算機で 1 つずつ動作し, カーネルが動作するための環境を構築する. カーネルは, システム全体で 1 つであり, 抽象化層が提供する機能を用いてすべての計算機を一括して管理する. すなわち, 抽象化層が個々の計算機を管理し, カーネルがシステム全体の資源管理を行う. このような構成をとることによって, 位置透過な資源管理を実現し, 複数の計算機を効率良く管理することができる.

2.1.1　抽象化層

抽象化層の役割は, 資源の抽象化を行い, 位置透過な資源管理を行うための環境を構築することである. 抽象化層は, すべての計算機で 1 つずつ動作し, 自身の動作する計算機を管理する. 抽象化層は, 自身が動作する計算機資源を直接操作する機能を持ち, メモリ, CPU, 割込み, 周辺デバイスといった計算機資源の抽象化を行う.

さらに, 抽象化層は, 抽象化層間で協調処理を行うための通信機能を持っている. 抽象化層間の通信には専用のプロトコルが使用され, 効率の良い協調処理を実現している. 各抽象化層は, 互いに協調処理を行うことによってシステム全体の資源の位置を

管理し, カーネルからの資源操作の要求を対象となる資源を持つ計算機に転送する.

2.1.2 カーネル

Solelc のカーネルは, 管理対象の計算機の台数によらず 1 つであり, 抽象化層が提供する機能を使用してすべての資源を一括して管理する. 単一のカーネルですべての計算機を管理することによって, 計算機ごとにカーネルを動作させている従来のオペレーティングシステムとは異なり, 協調動作のための処理が不要となる. このため, カーネル自体の処理は, 単純なものとなっている.

カーネルは, 従来のオペレーティングシステムが実現しているものと同様の機能を実現する. カーネルの機能は, 複数のカーネルスレッドによって実行される. カーネルスレッドは, 抽象化層の機能を使用して動作するため, 自身の位置や資源の位置によらず同一の方法で処理を行うことができる. そのため, カーネルスレッドは, 位置透過に動作可能であり, 各々のスレッドをそれぞれ異なる計算機上で動作させたり, 動作する計算機を動的に変更したりすることができる. このような特徴を利用してカーネルスレッドを適切な計算機上で動作させることによって, 効率的な資源の使用やカーネル自体の負荷分散が可能となる.

2.1.3 プロセス

プロセスは, カーネルによって実現され, カーネルの機能を使用して動作する. また, プロセスは, システムコールによってカーネルが提供するサービスを利用することができる. Solelc では, プロセスは 1 つ以上のユーザスレッドを持ち, プロセスのコードはこれらのスレッドによって実行される.

ユーザスレッドは, 任意の計算機上で自身の属するプロセスのコードやデータを参照でき, 位置透過に動作可能である. また, Solelc では, ユーザスレッドとカーネルスレッドが互いの位置を意識することなく動作可能であることから, カーネルスレッドが動作していない計算機でもユーザスレッドを動作させることができる.

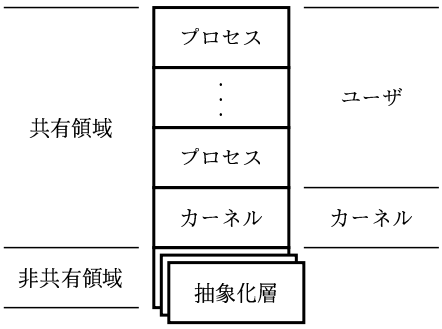


図 2 Solelc のアドレス空間

2.2 資源管理

Solelc では, システム全体の計算機資源を, 抽象化層によって抽象化し管理している. 抽象化層は, カーネルからの資源操作要求を対象となる資源を持つ計算機に適切に転送する. このため, Solelc では, システム上のどの計算機からでもすべての計算機資源を位置透過に使用可能である. Solelc におけるメモリ, CPU, 割込み, 周辺デバイスの管理について以下に述べる.

2.2.1 メモリ

Solelc では, 抽象化層とカーネルが協調動作することによってメモリ管理が行われる. すなわち, 抽象化層は, 各計算機の持つ物理メモリを, すべての計算機から共有されるシステム全体で 1 つの仮想アドレス空間として抽象化する. さらに, カーネルは, このアドレス空間を管理することによって, 結果としてすべての計算機のメモリを管理する.

アドレス空間は, 図 2 に示すように, 非共有領域と共有領域の 2 つに分けられる. 非共有領域は, 抽象化層のコードやデータが配置され, 計算機ごとに異なる内容を保持している. 共有領域にはカーネルやプロセスが配置され, すべての計算機がこの領域を共有する. このような複数の計算機によって共有されるアドレス空間を構築するために, 抽象化層は, 計算機間のメモリの一貫性制御を行っている. すなわち, 物理メモリの仮想アドレス空間への割り付けや計算機間でのページ内容の送受信を行い, 計算機間のメモリの一貫性を保っている. このため, カーネルは, すべての計算機から位置透過なメモリアクセスを行うことができる.

2.2.2 CPU

Solelc では, CPU を, 抽象化層がカーネルやプロセスのコードを実行するスレッドとして抽象化し, カーネルはスレッドを管理することによって結果として CPU 資源を管理する. スレッドは, カーネルやプロセスのコードを実行する実行実体である. すなわち, カーネルやプロセスは, スレッドを使用することによって CPU を使用する.

スレッドは, 抽象化層が実現する位置透過性によって, 任意の計算機が持つ CPU を使用して動作することが可能である. このため, 1 つの仕事を複数に分割し, それぞれ異なる CPU を使用して処理するといったことが容易に実現できる. このように, 単一のプロセスを複数の計算機で実行するのと同様に, Solelc では, カーネルの実行も複数の計算機で行うことが可能であり, 従来のシステムでは困難であったカーネル処理の負荷分散を実現することができる.

2.2.3 割込み

一般に, カーネルの処理は, プロセスや周辺デバイスが発生させる割込みを契機として行われる受動的なものである. Solelc におけるカーネルも基本的には割込みを契機として動作するが, 従来のシステムのように割込みを直接取得するのではなく, 抽象化層が生成するイベントを取得することによってカーネルの処理が行われる. イベントとは, タイマ割込みやシステムコールなど各計算機で発生した事象を抽象化したものである.

Solelc では, 抽象化層が割込みをイベントとして抽象化し, これをカーネルに通知する. イベントは, 発生した事象の種類を表すイベント番号とその事象に関する情報から構成される. 図 3 は, イベントがカーネルに通知される様子を表している. 抽象化層には, あらかじめ, カーネルによってイベントを取得するスレッドが登録される. 抽象化層は, このスレッドをすべての計算機に通知する. このため, 図 3 では, 計算機 B で発生した割込みがイベントとして抽象化された後, 計算機 A に通知されている. 抽象化層がこのような処理を行うことによって, カーネルは, 任意の計算機からすべてのイベントを取得することが可能となる.

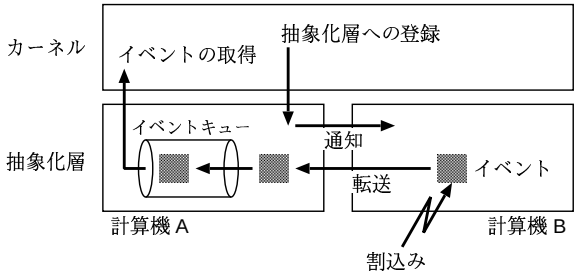


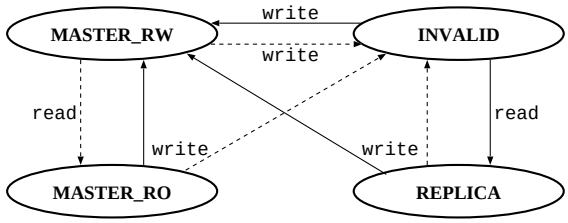
図 3 イベント処理

2.3 ページの状態

Solelc では, 抽象化層が共有領域の各ページの所有者と状態を管理することによって, 計算機間のメモリの一貫性制御を行っている. ページの所有者とは, 当該ページを管理する計算機の識別子である. ページの所有者は, 各計算機で行われるメモリ操作によって動的に変更され, ページの所有者となった計算機では当該ページの複製の管理が行われる. 抽象化層は, カーネルからの要求やスレッドが行うメモリアクセスに応じて, ページの状態を遷移させる. すべての抽象化層では各ページの所有者あるいはそれを知る計算機の識別子を保持しており, ページの内容を必要とする際, 所有者に要求を通知することによってページの内容を取得することができる.

ページの状態遷移を図 4 に示す. MASTER_RW と MASTER_RO はページを所有している状態であり, 当該ページに割り付けられた物理メモリにはそのページの内容が保持されている. MASTER_RW 状態のページは読み書き可能であるが, MASTER_RO 状態のページは読出し専用である. さらに, MASTER_RO 状態のページは複製を持ち, 複製であることを表す REPLICA 状態のページにも, 読み出し専用で物理メモリが割り付けられる. すなわち, REPLICA 状態のページは, ページの内容を持つが所有者ではなく, 他の計算機が当該ページの所有者であることを表している. また, INVALID は, 物理メモリが割り付けられておらず, ページの内容を持たない状態である. 共有領域内の有効なページは, 次の 2 つのうちのいずれかである.

- 1 台の計算機では MASTER_RW 状態で, 他のすべての計算機では INVALID 状態である.
- 1 台の計算機では MASTER_RO 状態で, 他の計算機では REPLICA 状態あるいは INVALID 状態である.



点線は, 他の計算機で行われたメモリアクセスによって, ページの状態が変更されることを示す.

図 4　ページの状態遷移

抽象化層は, 各計算機で行われるメモリ操作に応じて, これら 2 つの間を遷移させる. すなわち, 抽象化層は, 複製を持つページに対して書き込みが行われるとき, 当該ページの複製を無効化することによって, 共有領域の順序一貫性を実現している.

2.4　システムコール

カーネルが実現するサービスは, システムコールによってプロセスに提供される. Solelc では, システムコールもイベントとして処理される. すなわち, ユーザスレッドは内部割込みを発生させることによってサービス要求を発行し, カーネルはこれをイベントとして取得する. システムコールの発行によって生成されたイベントは, 抽象化層によって適切な計算機に転送されるため, ユーザスレッドとカーネルスレッドは, 互いの位置を意識することなく動作可能である.

システムコール処理は, ディスパッチャ/ワーカモデルに基づき, システムコールを取得するディスパッチャと実際に処理を行うワーカの 2 種類のスレッドによって行われる. ディスパッチャの数は 1 つであるが, ワーカは複数存在する. おのおののワーカは, ディスパッチャによって取得されたすべてのシステムコールを処理することができる. このため, 複数のシステムコールが発生した場合にも, 複数の計算機上のワーカに処理を分散させることが可能である. また, システムコールの処理状態によらず, 直ちにシステムコールを取得することができる.

3　耐障害性向上手法

Solelc では, 単一のカーネルが複数の計算機にまたがって動作するため, 計算機が 1 台でも停止した場合, その影響がシステム全体に広がる. このような場合においても, 障害が発生していない計算機が継続して動作することを目的とした耐障害性手法について検討を行った.

Solelc のカーネルは, カーネルスレッドによって実行される. Solelc におけるカーネルスレッドは, 位置透過に動作可能であり, 各々のスレッドをそれぞれ異なる計算機上で動作させたり, 動作する計算機を動的に変更したりすることができる. この特徴を利用することにより, 停止した計算機上でカーネルスレッドが動作していた場合に, 他の計算機のメモリ上にコードが保持されていれば, カーネルスレッドを再生成することが可能となる.

Solelc では, 管理するすべての計算機によって共有される仮想アドレス空間上のメモリに冗長性がある. すなわち, 1 台の計算機が停止した場合においても, 該当計算機によって保持されていたメモリ内容と同じデータやコードが他の計算機に存在する. 以上より, 動作する計算機上のコードを利用して失われたカーネルの機能を復元させたり, アクセスする対象のデータを切替えたりすることでシステムを継続して動作させることが可能となる.

分散共有メモリの障害対策として, Linda モデルのタプルスペース通信における複製管理に関する研究 [3, 4] が行われている. Linda モデルでは, 分散システムにおいて, タプルスペースと呼ばれる共有メモリ空間と, タプルスペースに対してデータのやりとりを行うための 4 種類の基本命令を提供している. タプルスペース通信では, タプルスペースを管理するサーバに障害が生じた場合, これを利用して実行されている全プロセスが停止してしまうという問題がある. 文献 [3] と [4] では, この問題を解決する方法の 1 つとして, ネットワーク上にタプルスペースの複製を配置している.

一方, Solelc では, 複数の計算機から資源の位置を意識することなくメモリ操作を可能とするため, メモリ操作の状況に応じて複製を配置している. すなわち, メモリに冗長性があり, これを一貫性制御によって管理し順序一貫性を保証している. しかし, Solelc の場合, メモリ管理を行うカーネル自体が分散配置されているため, 複製が存在していてもカーネルの一部の機能が失われ, システムの継続的な稼働が不可能となる. すなわち, 単に複製の保持だけでなく, 障害が発生する以前の有効なメモリ内容を保持する必要がある.

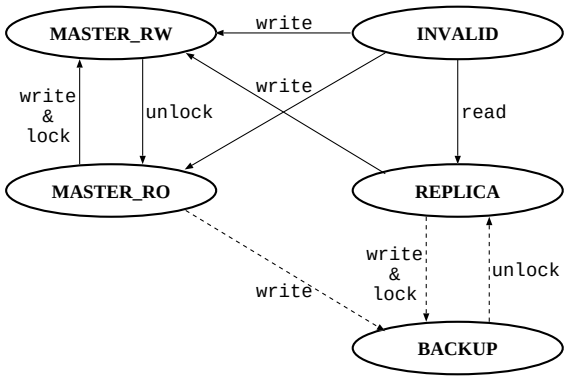


図 5　トランザクション化における
ページの状態遷移

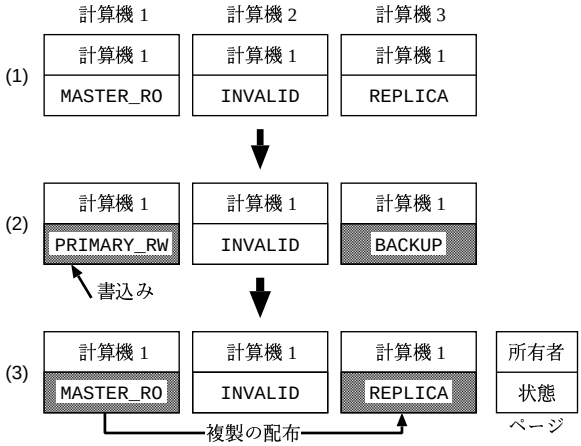


図 6　ページの状態遷移の例

これらを実現するために, メモリ操作をトランザクション化することによって, 耐障害性を向上させる手法を提案する. すなわち, カーネルの処理単位であるシステムコールを 1 つのトランザクションとし, メモリへの書込み時において他のトランザクションからの書込みを禁止する. これにより, 障害発生時に有効なメモリ内容を保持し, これを利用しカーネルスレッドの再生成を行うことによってシステムの継続的な動作を保証する.

4　メモリ操作の トランザクション化方式

4.1　トランザクション化の概要

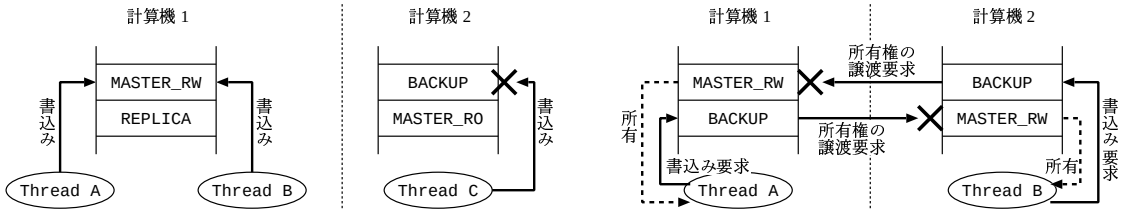
カーネルの処理は, イベントの取得を契機として行われる. 障害発生時の有効なメモリ内容とは, その直前のイベントの処理が完了した際のメモリ内容である. したがって, 本手法では, 各々のイベントの処理中に行うメモリ操作を 1 つのトランザクションとする. また, トランザクション処理の開始前にチェックポイントを設け, 操作対象となるメモリ領域の内容を他の計算機へバックアップする.

トランザクションの処理中は, その処理に必要なページへの他のトランザクションからの書込みを禁止する. 他のトランザクションからの書込みの禁止は, トランザクション処理の終了時に解除する. このため, 障害発生時に処理していたトランザクションをアボートし, バックアップを利用してチェックポイントまでロールバックすることができる. また, そのメモリから, 失われたカーネルの機能を復旧させることで, システムを停止させることなくカーネルの機能をユーザに提供することが可能となる.

4.2　ページの状態と遷移

トランザクション化を実現するためには, メモリのバックアップが必要となる. このバックアップをページの状態の 1 つとして位置づけた. その場合のページの状態遷移を図 5 に示す. 本手法では, 所有者が持つページが MASTER_RW 状態の場合, 他の計算機からの所有者権限の譲渡要求を拒否することで他のトランザクションからの書込みを禁止している. また, 状況に応じて配置されていた複製を必ず 1 つ以上配置するため, 所有者が持つページは, 書込み要求があった場合を除いて常に MASTER_RO 状態となる. 新たに追加した BACKUP 状態は, 所有者の持つページが MASTER_RW 状態の場合に REPLICA 状態から遷移する状態であり, 障害発生時に有効となるメモリ内容を持つ無効化されない保護された複製である. トランザクション化における共有領域内の有効なページは, 次の 2 つのうちのいずれかである.

- 1 台の計算機では MASTER_RW 状態で, 他の計算機では BACKUP 状態あるいは INVALID 状態である.
- 1 台の計算機では MASTER_RO 状態で, 他の計算機では REPLICA 状態あるいは INVALID 状態である.



(a) マルチスレッドのページアクセス　　(b) デッドロック

図 7　トランザクション化における問題

トランザクション処理におけるページの状態遷移の例を図 6 に示す.

(1) トランザクション処理開始時の初期状態である. 所有者が持つページが MASTER_RO 状態であるため, 他の計算機は読出しアクセス, 所有権の譲渡や複製を要求することができる.

(2) 書込み要求が発生した場合は, REPLICA 状態のページを BACKUP 状態に変更し, 複製の保護を行った後メモリ内容を書き込む. この時点からトランザクション処理の終了までの間, 他の計算機からこのページへのメモリアクセスは, すべて禁止となる.

(3) トランザクション処理終了時における所有者の持つページの状態が MASTER_RW 状態の場合, BACKUP 状態のページを REPLICA 状態とし新たなメモリ内容の複製を配布する. また, 所有者の持つページの状態を MASTER_RO 状態にする. 所有者の持つページの状態が MASTER_RO 状態の場合は, メモリ内容に変更がないため, そのままトランザクション処理を終了する.

4.3　マルチスレッドのページアクセス

4.2 節で述べたトランザクション化手法では, 各ページを計算機単位で所有しており, ページへのアクセスも計算機毎に制限していた. このため, 1 つのスレッドがあるページに書込みが可能であれば, 同じように同一計算機上のもう 1 つのスレッドも当該ページへの書込みが可能である (図 7 (a) 参照). すなわち, 同一計算機上に複数のスレッドが動作する場合, スレッドを区別してロックをかけることができない. これは, あるトランザクションの書込み処理が行われたページに, 他のトランザクションからの書込みをも許可することになる. したがって, 上述したようなマルチスレッドのページアクセスが発生する場合を考慮したトランザクション化のための手法を検討した.

この問題を解決するために, 所有者が管理するページの情報にスレッド ID を追加した. このスレッド ID は, 当該ページをロックしたスレッドの ID であり, ロックをかける際に登録する. なお, 初期値は-1 であり, その場合は任意のスレッドがページをロックすることができる.

各々のスレッドは, 書込みを行う際, 自身のスレッド ID と当該ページをロックしたスレッド ID とを比較し, 同じ ID であれば書込みを許可する. 異なる ID であった場合には, カーネルに存在するページ毎の予約キューにスレッドをつなぎ待ち時間を測定する. これにより, 同一計算機上に複数のスレッドが存在する場合においても, スレッド毎に異なるアクセス制限を設定することができる.

4.4　デッドロックの回避

メモリ操作をトランザクション化することによって, 障害発生時に有効なメモリ内容を保持することが可能となる. しかし, トランザクションの処理中は, 他のトランザクションからのアクセスが制限あるいは禁止されるため, デッドロックが発生する可能性がある (図 7 (b) 参照).

そこで, 所有者が管理しているページをロックしたスレッド ID をたどることによって, 資源割付けの循環待ち (図 8 参照) が発生しているかを探索し, デッドロックの検出を行う. 各々のスレッドは, ページアクセスによってページフォルトが発生し所有者権限の譲渡を要求した結果, ページの所有権を得ら

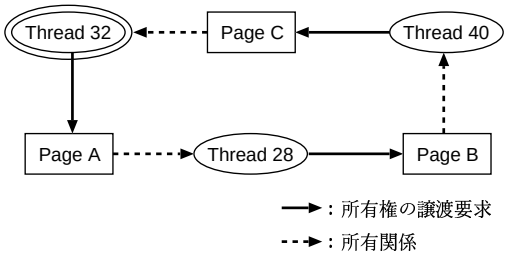


図 8　資源割付けの循環待ち

れなかった場合, 予約キューにスレッドをつなぎ待ち時間を測定する. 予約キューにつながれたスレッドは, 待ち時間が一定時間を経過してもページの所有権が得られない場合, デッドロックの可能性があるため, 当該ページをロックしたスレッド ID を参照し, そのスレッドがアクセス要求を発行しているページを検索する. さらに, 検索したページをロックしたスレッド ID を参照し, そのスレッドがアクセス要求を発行しているページを検索する.

この処理を繰り返した結果, 一定時間を満了したスレッド ID が参照された場合, 資源割付けの循環待ちが発生している. すなわち, デッドロックが発生したことにより, 循環待ちに関係するスレッドのうち最も待ち時間の短いスレッド, すなわち最後に要求を発行したスレッドから順に終了させることによって, デッドロックを回避することができる.

5　評価

システムコールの発行によりページフォルトが発生し, メモリへの書込みが行われた際に複製を配布する. このとき, 4.2 節の図 6 で示したページの状態遷移が起こる. その処理時間を計測した. 複製の配布処理は, 主に以下に示す 3 つで構成されている.

(1) 配布するページのメモリ領域のコピー
(2) MASTER_RW 状態のページを保持する計算機における送信処理
(3) BACKUP 状態のページを保持する計算機における受信処理

なお, 実験には, Celeron 500MHz を搭載した PC/AT 互換機を 100Mbps のイーサネットで接続した環境を用いている.

(1)〜(3) の各処理時間とその合計を表 1 に示す. 送受信の同期を取るため, (3) では, 受信確認のパケットを送信元計算機に返している. これらの処理時間は, fork システムコールの処理時間の約 34%を占めるオーバヘッドとなる. また, オーバヘッドは, 書き込まれたページ数に比例して増加する. しかし, このオーバヘッドは, メモリ操作をトランザクション化することによって, 障害発生時に有効なメモリ内容を保持することができることとのトレードオフであり, 今後の検討課題である.

表 1　複製配布の処理時間

処理内容	時間
(1) メモリコピー	0.036ms
(2) 送信処理	2.688ms
(3) 受信処理	1.184ms
合計	3.908ms

6　おわりに

本稿では, 分散オペレーティングシステム Solelc におけるメモリ操作のトランザクション化による耐障害性向上手法について述べた. メモリ操作をトランザクション化することによって, 障害発生前の有用なメモリ内容を保持することができる. また, このメモリを利用することによって失われたカーネルの機能を復元することやシステムを継続して動作させることが可能である.

参考文献

[1] 芝公仁, 大久保英嗣: “分散オペレーティングシステム Solelc の構成, ” 情報処理学会研究報告 2000–OS–84, Vol. 2000, No. 43, pp. 237–244 (2000).
[2] 芝公仁, 大久保英嗣: “分散オペレーティングシステム Solelc の設計と実装, ” 電子情報通信学会論文誌 D–I, Vol. J84–D–I, No. 6, pp. 617–626 (2001).
[3] 川口 昇, 中村 健二, 佐藤 文明, 水野 忠則: “分散共有メモリシステムの複製管理方式, ” 情報処理学会研究報告 96–DPS–78, Vol. 96, No. 95, pp. 37–42 (1996).
[4] 川口 昇, 佐藤 文明, 水野 忠則: “タプルスペース通信の複製管理方式, ” 情報処理学会論文誌 Vol. 39, No. 2, pp. 388–394 (1998).