Vol.2021-OS-151 No.12
2021/3/2

計算機状態の高速な保存と復元を可能にする不揮発性メモリ向け*Tender*のプレート機能の実現

田中 雅大[1]　山内 利宏[1]　谷口 秀夫[1]

概要：計算機の主記憶に用いられるメモリは揮発性メモリである．***Tender*** では，揮発性の主記憶をあたかも不揮発性であるかのように見せるプレート機能を実現している．この機能は，揮発性の主記憶を搭載した計算機で仮想記憶空間上のデータを外部記憶装置上へ書き出し，計算機再起動後に仮想記憶空間上に復元する．不揮発性メモリの性能向上に伴い，不揮発性メモリを利用するソフトウェアの技術が研究されている．不揮発性の主記憶を搭載した計算機でプレート機能を利用する場合，仮想記憶空間上のデータの保存先領域を主記憶に確保できる．これにより，計算機終了時だけでなく，実行途中の計算機状態の高速な保存と復元や，計算機の緊急停止による計算機状態の消失を防ぐことができる．本稿では，不揮発性メモリ向け ***Tender*** のプレート機能の設計について述べる．評価では，計算機状態の保存と復元にかかる処理時間を測定し，揮発性の場合との比較を示す．

Masahiro Tanaka[1]　Toshihiro Yamauchi[1]　Hideo Taniguchi[1]

1. はじめに

計算機の主記憶に用いられるメモリは揮発性メモリである．揮発性メモリは，電源の供給がなければデータを保持することができないため，オペレーティングシステム（以降，OS）や応用プログラム（以降，AP）は，計算機の終了後に利用する主記憶上のデータを外部記憶装置に保存することにより，データを永続化する．

Tender オペレーティングシステム [1]（以降，***Tender***）では，揮発性の主記憶をあたかも不揮発性であるかのように見せるプレート機能 [2] を実現している．この機能では，OS が揮発性メモリ上の仮想記憶空間のデータを外部記憶装置上へ書き出し，仮想記憶空間のデータを永続化する．計算機再起動時は，外部記憶装置に保存したデータを仮想記憶空間上に復元する．このプレートの書き出しと復元により，仮想記憶空間上のデータが不揮発性メモリ上に存在するかのように計算機再起動後に復元される．

また，***Tender*** では，プレート機能を使用して，カーネル用とユーザ用の仮想記憶空間上に存在するすべてのデータを永続化することにより，OS の処理と AP の処理を永続化する揮発性メモリを利用した動作継続制御 [3] を実現している．これにより，計算機停止前の処理を計算機再起動後に継続して実行できる．

不揮発性メモリの性能向上に伴い，不揮発性メモリを利用するソフトウェアの技術が研究されている．不揮発性メモリのみを主記憶に搭載した計算機を利用する場合，***Tender*** ではプレート機能を使用せずとも，主記憶上のデータの永続化が可能となる．文献 [6] では，不揮発性メモリを利用した動作継続制御を提案し，その設計について述べた．また，不揮発性メモリの主記憶をエミュレーションする仮想計算機 [4] における実現方式について述べた．不揮発性メモリを利用した動作継続制御で保存できる計算機状態は，利用者が計算機の終了処理を要求したときのみである．このため，停電などの計算機の緊急停止が発生すると，CPU のレジスタやキャッシュのデータが消失してしまい，計算機処理を継続できない．

そこで，プレート機能を使用し，不揮発性メモリ上に任意の計算機状態を保存し，復元できるようにする．任意の計算機状態とは，計算機終了時の状態だけでなく，AP の処理中の状態など計算機の実行途中の状態を指す．これにより，計算機状態の高速な保存復元とともに計算機の緊急停止による計算機状態の消失を防ぐことができる．

本稿では，不揮発性メモリ向け ***Tender*** のプレート機能の設計について述べる．また，不揮発性メモリ向けプレー

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

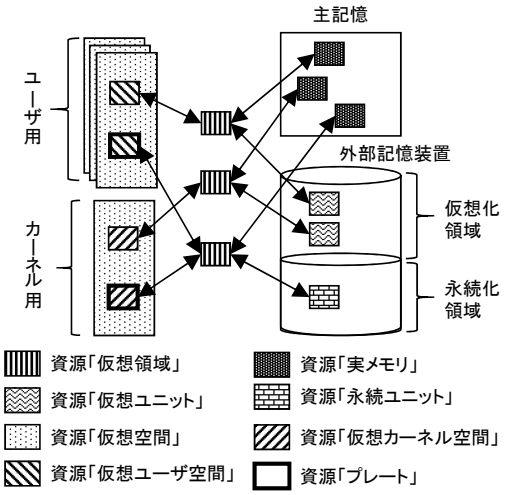

図 1　メモリ関連資源の関係

ト機能による計算機状態の保存と復元処理について，揮発性メモリの場合との比較評価を述べる．

2. *Tender* オペレーティングシステム

2.1 資源の分離と独立化

Tender では，OS が制御し管理する対象を資源と呼び，資源の分離と独立化を行っている．資源の種類ごとに，資源を管理する管理表と資源を操作するプログラムが存在する．資源は，資源名と資源識別子によって識別される．また，***Tender*** 特有の構造に資源インタフェース制御がある．資源インタフェース制御は，表プログラム構造という構造で資源を操作するプログラムを管理する．これにより，資源操作のインタフェースを統一し，各資源を操作するプログラムを資源の種類ごとに分離する．

2.2 メモリ関連資源

Tender におけるメモリ関連資源の関係について，**図 1** に示し，以下で説明する．資源「仮想領域」は，メモリイメージを仮想化した資源であり，実体は資源「実メモリ」上，もしくは外部記憶装置上の領域に存在する．資源「実メモリ」は，主記憶上の領域を表す資源である．資源「仮想ユニット」は，仮想化領域の管理単位である．仮想化領域とは，主記憶上のデータを一時的に保存するための領域であり，既存 OS のスワップ領域を利用して実現している．資源「永続ユニット」は，永続化領域の管理単位である．永続化領域とは，仮想記憶空間上のデータを永続化するために利用する領域であり，既存 OS のファイルシステム領域を利用して実現している．資源「仮想空間」は，特定のアドレス領域をもつ仮想的な空間であり，仮想アドレスから実アドレスへのアドレス変換表に相当する．資源「仮想カーネル空間」は，カーネル用の仮想空間に存在するデータを表す．資源「仮想カーネル空間」は，資源「仮想領域」をカーネル用の資源「仮想空間」に貼り付ける際に生成される．「貼り付ける」とは，仮想アドレスを実アドレスに対応付けすることである．また，仮想アドレスと実アドレスの対応付け解除を「剥がし」と呼ぶ．資源「仮想ユーザ空間」は，ユーザ用の仮想空間に存在するデータを表す．資源「仮想ユーザ空間」は，資源「仮想領域」をユーザ用の資源「仮想空間」に貼り付ける際に生成される．

2.3 プレート機能

プレート機能 [2] とは，OS が揮発性メモリ上の仮想記憶空間のデータを外部記憶装置上へ自動的に書き出すことにより，データを永続化する機能である．永続化の対象となる領域を「プレート」と呼ぶ．

プレートは，仮想記憶空間上に存在することを基本とし，仮想記憶空間と外部記憶装置の入出力契機を OS が判断する．AP は，必要に応じてプレートを自分の仮想記憶空間にマッピングして利用する．また，AP がプレートの内容の書き出しを OS に依頼することもできる．

また，プレート機能は，計算機再起動時にプレートを復元する．これにより，プレートは，計算機終了前と同じ仮想記憶空間のアドレス領域に存在し続ける性質を持つ．

Tender では，プレート機能を資源「プレート」として実現している．図 1 に示すように，1 つのプレートは実メモリ，仮想領域，永続ユニット，および仮想カーネル空間（もしくは仮想ユーザ空間）からなる．

2.4 揮発性メモリを利用した動作継続制御

2.4.1 処理の流れ

揮発性メモリを利用した動作継続制御 [3] は，計算機停止前の処理を計算機再起動後に継続して実行する機能を持つ．揮発性メモリを利用した動作継続制御の処理の流れを**図 2** に示す．揮発性メモリを利用した動作継続制御は，以下の 3 つの処理からなる．

(1) 仮想記憶空間上のデータの永続化

Tender では，OS や AP が利用する仮想記憶空間上のデータは，仮想カーネル空間か仮想ユーザ空間によって管理される．動作継続制御の有効化処理により，現存するすべての仮想カーネル空間と仮想ユーザ空間をプレート化する．「プレート化」とは，仮想カーネル空間や仮想ユーザ空間を利用してプレートを生成することを意味する．動作継続制御の有効化処理以降に，新たに生成された仮想カーネル空間と仮想ユーザ空間は，自動でプレート化される．これにより，仮想記憶空間上のすべてのデータは，プレート上に存在する．

(2) プレートの書き出し

プレート機能によって管理される仮想記憶空間上のデータを外部記憶装置上の領域に書き出す．

入出力デバイスの利用中にプレート書き出しを行うと，計算機状態の復元後に正常に入出力処理を継続で

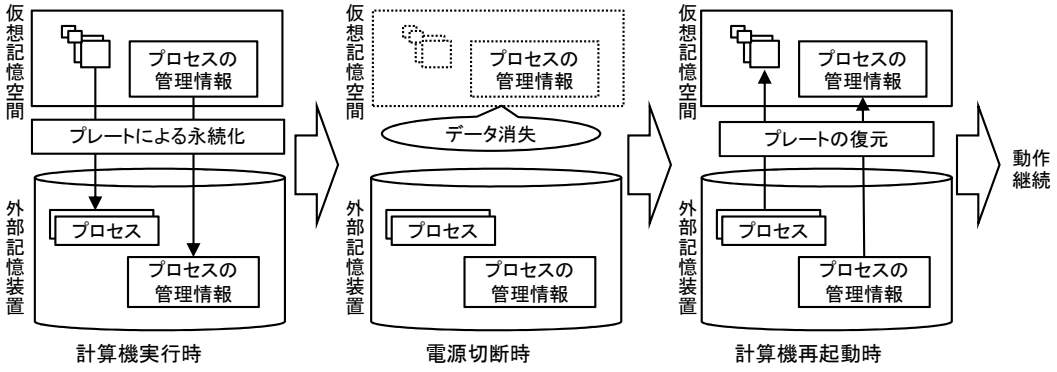

図 2　揮発性メモリを利用した動作継続制御の処理の流れ

きない．そこで，永続化データを正常に保存するために，プレート書き出し時にプロセッサやデバイスなどのハードウェア資源を占有する．***Tender*** では，ハードウェア資源を資源「システム」によって管理している [5]．資源「システム」を構成するハードウェア資源を占有（以降，システムの占有）することで，他のプロセスやデバイスの割り込みルーチンにより，システムを構成するハードウェア資源を利用されないように制御する．これにより，正常に入出力処理を継続できる計算機状態を保存する．

(3) プレートの復元

プレートの復元処理は，OS の起動処理中に行う．OS の起動処理では，OS の処理に必要な管理表の確保と初期化を行い，次にプレートの復元に必要となる資源管理部の初期化処理を行う．その後，外部記憶装置上にプレート管理表が存在する場合，プレート管理表を用いて，外部記憶装置に保存されたデータから仮想記憶空間上にプレートを復元する．最後に，復元したプロセスの動作継続を行う．具体的には，プレートの書き出し処理を依頼した AP の処理に切り替わる．

2.5　不揮発性メモリを利用した動作継続制御

2.5.1　基本方式

不揮発性メモリを利用した動作継続制御 [6] は，主記憶上に残った計算機終了前のデータを利用して，計算機再起動後に処理を継続する．不揮発性メモリを利用した動作継続制御の処理の流れを図 **3** に示す． この機能は，不揮発性メモリのみを主記憶に搭載した計算機環境を想定している．また，外部記憶装置は実行ファイルを参照する用途にのみ使用する．

不揮発性メモリを利用した動作継続制御では，終了処理時に，OS の処理の継続に必要な CPU のレジスタの値を保存する．OS の処理の継続に必要なレジスタとして，ページディレクトリの物理アドレスを格納する CR3 やグローバルディスクリプタテーブルのアドレスを格納する GDTR などがある．これらのレジスタの値を保存した後は，CPU

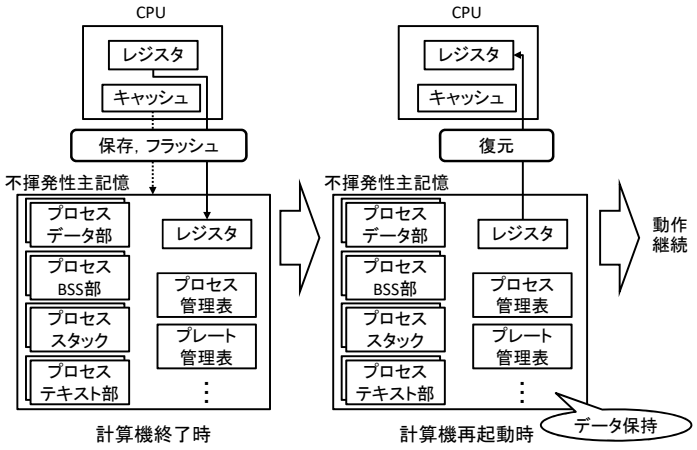

図 3　不揮発性メモリを利用した動作継続制御の処理の流れ

のキャッシュをフラッシュする．

計算機再起動後は，保存したレジスタの値を復元し，デバイスを初期化した後，終了処理前の計算機処理を再開する．

2.5.2　実現環境

不揮発性メモリを利用した動作継続制御を実現するためには，不揮発性メモリを主記憶として用いた計算機が必要である．しかし，この計算機は，簡単には入手できない．このため，不揮発性メモリの主記憶をエミュレーションする仮想計算機 [4] を利用する．計算機の緊急停止は，QEMU モニタコンソールからコマンドを使用してエミュレータを終了させることによって再現する．

3.　不揮発性メモリ向け *Tender* のプレート機能

3.1　目的

不揮発性メモリ向けプレート機能は，不揮発性メモリ上に任意の計算機状態を保存し，復元することで処理を継続することを目的とする．計算機状態の保存のためにプレート機能を利用する理由を次に示す．不揮発性メモリ上に任意の計算機状態を保存するだけであれば，使用していない主記憶上の領域に使用中の主記憶のデータを保存すればよい．しかし，この方法の場合，カーネルのテキスト部など

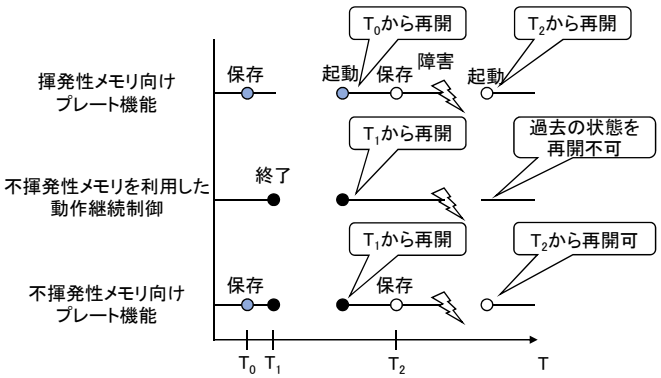

図 4　各機能による計算機状態の保存と復元

表 1　各機能における計算機状態の保存内容と保存先

機能	保存内容	保存先
揮発性メモリ向け プレート機能	任意	外部記憶装置上 （資源「永続ユニット」）
不揮発性メモリを利用した 動作継続制御	計算機 終了時	不揮発性メモリ上
不揮発性メモリ向け プレート機能	任意	不揮発性メモリ上に 確保した保存先領域

保存の必要のない領域まで保存することになり，効率が悪い．効率良く保存するためには，保存対象となる主記憶の領域に対応する仮想記憶空間の利用状況を基に，保存するかどうかを判断できることが望ましい．このため，仮想記憶空間の内容を基に永続化するかどうかを決定できるプレート機能を利用する．

不揮発性メモリを利用した動作継続制御とプレート機能による計算機状態の保存と復元を図 **4** に示す．2.5 節で述べた通り，不揮発性メモリを利用した動作継続制御は，終了時 T_1 の状態を保存し，再起動後は T_1 の状態から再開できる．しかし，CPU のレジスタやキャッシュは揮発性のため，停電などによる計算機の緊急停止後に処理を継続できない．そこで，揮発性メモリ向けプレート機能のように，任意の時点の CPU とメモリの状態を別の領域に保存し，再起動時に復元できるようにする．これにより，T_2 の時点で計算機状態を保存すると，緊急停止からの再起動後に T_2 から再開でき，計算機の緊急停止による計算機状態の消失を回避できる．

なお，揮発性メモリ向けプレート機能では，保存した計算機状態から復元するため，T_0 の状態を保存し T_1 で計算機を終了させると，再起動後は T_0 の状態を復元する．これに対して，不揮発性メモリ向けプレート機能は，正常に終了したかどうかで復元する計算機状態を決定する．したがって，T_0 の状態を保存しても T_1 の状態から復元する．

3.2　課題

3.1 節で述べた目的を達成するための課題を以下に示す．

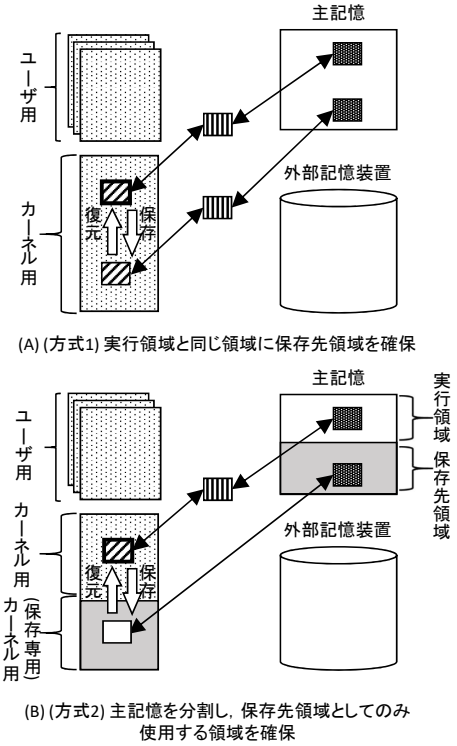

図 5　保存先領域の確保の方式

（課題 1）保存先領域の確保方法の検討

不揮発性メモリを利用した動作継続制御とプレート機能における計算機状態の保存内容と保存先を**表 1** に示す．揮発性メモリ向けプレート機能では，仮想記憶空間上のデータの保存先となる領域（以降，保存先領域）を外部記憶装置に確保する．また，不揮発性メモリを利用した動作継続制御は，計算機終了時の状態を保存するため，保存先領域を確保する必要はない．このように，従来の ***Tender*** では，主記憶上に保存先領域を確保することを考慮していない．したがって，主記憶上に保存先領域の確保方法を検討する．

（課題 2）保存先領域の管理方法の検討

主記憶は，OS や AP が利用する一時データを配置する領域（以降，実行領域）としても利用される．保存先領域を適切に管理できなければ，保存先領域のデータが上書きされたり，保存先領域を参照できず，計算機状態を復元できない可能性がある．保存先領域は，プレートの操作処理時に操作される．したがって，各プレートの操作処理時における保存先領域の管理方法について検討する．

3.3　対処

3.3.1　保存先領域の確保方法の検討

保存先領域の確保の方式を以下に 2 つ示す．また，それぞれの方式の概要を図 **5** に示す．

（方式 1）実行領域と同じ領域に保存先領域を確保

この方式は，主記憶と仮想記憶空間の領域を分割せ

ず，データを保存する方式である．仮想記憶空間のプレート化時に，仮想カーネル空間を生成し，生成した仮想カーネル空間にデータを保存する．プレート管理表ではプレート化された仮想記憶空間と保存先領域となる仮想カーネル空間の対応関係を管理する．

（方式 2）主記憶を分割し，保存先領域としてのみ使用する領域を確保

この方式は，主記憶と仮想記憶空間の領域を分割し，データを保存する方式である．実メモリ空間と仮想記憶空間の保存先領域に該当する領域をストレートマッピングする．仮想記憶空間のプレート化時には，主記憶上の保存先領域から領域を確保し，確保した実アドレスに対応する仮想アドレスに保存する．

(方式 1) は，実メモリ，仮想領域，および仮想カーネル空間を生成する必要があるのに対し，(方式 2) は実メモリのみでよい．揮発性メモリ向けプレート機能は，保存処理やプレートの生成にかかるオーバヘッドを考慮して，計算機の利用者によって任意の計算機状態の保存復元機能を使用するかどうか切り替えることができる．しかし，(方式 2) の場合，プレート機能による任意の計算機状態の保存復元機能の使用の有無に関わらず，使用できるメモリの量が制限される．計算機の使用時の性能の観点から (方式 1) を採用する．

3.3.2 保存先領域の管理方法の検討

保存先領域を操作する処理として，プレートの生成（書き出し），プレートの削除，およびプレートの復元の 3 つがある．以下で，それぞれの処理時の保存先領域の管理方法について述べる．

(1) プレートの生成（書き出し）

保存先領域が 1 つの場合，プレートの書き出し処理中に障害が発生すると，復元すべきデータが上書きされてしまい，計算機再起動後に復元できない．これを防ぐために，1 つの領域に対して，保存先領域を 2 つ確保する．プレート管理表には，どちらの保存先領域に書き出したかを管理する．これにより，プレートの書き出し処理中に障害が発生しても復元できるようにする．

また，保存先領域は仮想記憶空間のプレート化時に確保する．この理由を次に示す．プレートの書き出し時に確保する場合，仮想カーネル空間の確保により，書き出し処理中に OS の資源管理用領域が更新される．これにより，OS の資源管理用領域を最後にまとめて書き出す必要がある．前回の保存時から資源の情報に変化がない場合でも，書き出す必要があり，書き出しのサイズが増えてしまう．したがって，仮想記憶空間のプレート化時に確保する．

(2) プレートの削除

仮想カーネル空間（および仮想ユーザ空間）削除時に，削除対象の領域に対応する保存先領域を削除すると，その領域を利用され，保存内容を上書きされる可能性がある．このため，保存先領域は削除せず，保存先領域の情報を保存先領域削除予定管理表に登録する．この管理表に登録されたデータは，その保存先領域が必要なくなったとき，すなわち次のプレート書き出し処理終了時に削除する．

(3) プレートの復元

プレートの復元は，プレート書き出し時の仮想記憶空間とプレートの情報を基に行う．このため，プレート管理表と仮想カーネル空間を構成する資源の管理表が必要となる．仮想カーネル空間を構成する資源として，資源「実メモリ」，資源「仮想領域」，資源「仮想空間」，資源「仮想カーネル空間」がある．これらの管理表を復元するためには，これらの管理表が起動時に常に同じ実アドレスに保存されていなければならない．したがって，これらの管理表の保存先領域は，初期化処理時に特定の実アドレスに確保する．仮想カーネル空間を復元することによって，仮想カーネル空間を構成する資源以外の資源管理部も復元される．このため，計算機再起動時に初期化する資源管理部は，仮想カーネル空間を構成する資源の資源管理部のみでよい．

3.4 設計

3.4.1 仮想記憶空間上のデータの永続化

3.3 節を踏まえた不揮発性メモリ向けプレート機能における仮想カーネル空間のプレート化の処理流れを**図 6** に示す． 図 6 の仮想カーネル空間を仮想ユーザ空間に置き換えたものが，仮想ユーザ空間のプレート化の処理流れとなる．仮想記憶空間上のデータの永続化処理は，プレート機能による任意の計算機状態の保存復元機能を有効化するときに実行する．

仮想カーネル空間管理表のエントリを確認し，仮想カーネル空間を構成する資源の管理表かプレートの管理表であれば（図 6(5)），初期化処理時に確保した保存先領域に複写する（図 6(6)）．仮想カーネル空間を構成する資源の管理表かプレートの管理表でなければ，保存先領域を生成し複写する（図 6(7)(8)）．この生成処理によって，保存先領域も仮想カーネル空間となる．しかし，保存先領域の保存先領域を生成する必要はないため，当該仮想カーネル空間が保存先領域であればプレート化処理を省略する（図 6(4)）．

仮想カーネル空間のプレート化の後は，表プログラム構造で管理される仮想カーネル空間の削除プログラムを置換する（図 6(10)）．置換後のプログラムでは，仮想カーネル空間削除後に当該仮想カーネル空間の保存先領域を保存先領域削除予定管理表に登録する．

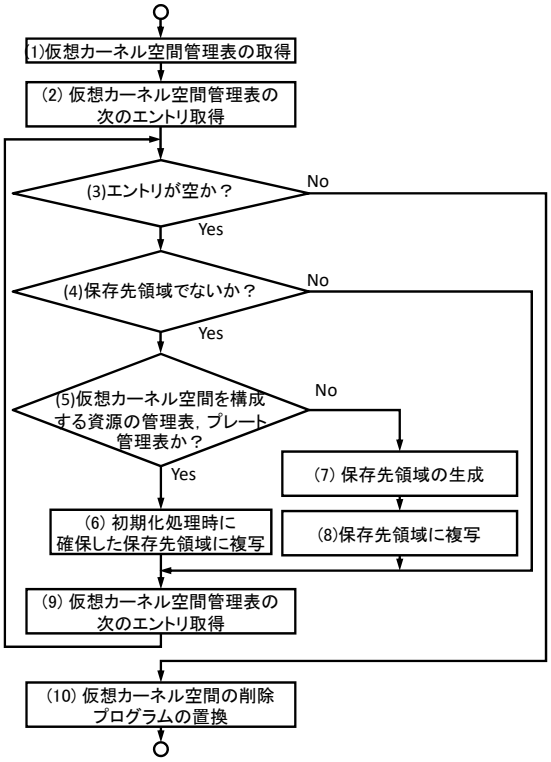

図 6 仮想カーネル空間のプレート化の処理流れ

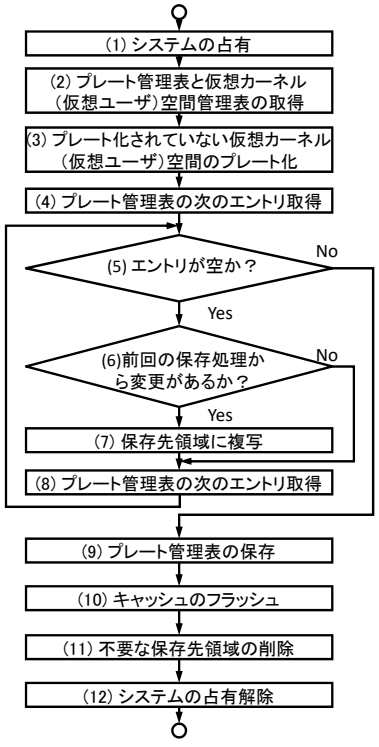

図 7 プレートの書き出し処理の流れ

3.4.2 プレートの書き出し

3.3 節を踏まえた不揮発性メモリ向けプレート機能におけるプレートの書き出し処理を**図 7** に示す．前回の計算機状態保存時から新たに仮想カーネル空間（および仮想ユーザ空間）が生成された場合，この領域の保存先領域は確保されていない．このため，保存処理のはじめに，プレート化されていない仮想カーネル空間（および仮想ユーザ空間）をプレート化し，保存先領域を確保する（図 7(3)）．

次に，前回の計算機状態保存時から更新がある場合，当該プレートを複写し，保存する（図 7(6),(7)）．すべてのプレートの複写が完了した後に，プレート管理表を保存する（図 7(9)）．プレート管理表を最後に保存する理由は，どちらの保存先領域に保存したかという情報を保存するためである．その後，キャッシュをフラッシュし（図 7(10)），保存先領域削除予定管理表に登録された不要な保存先領域を削除する（図 7(11)）．

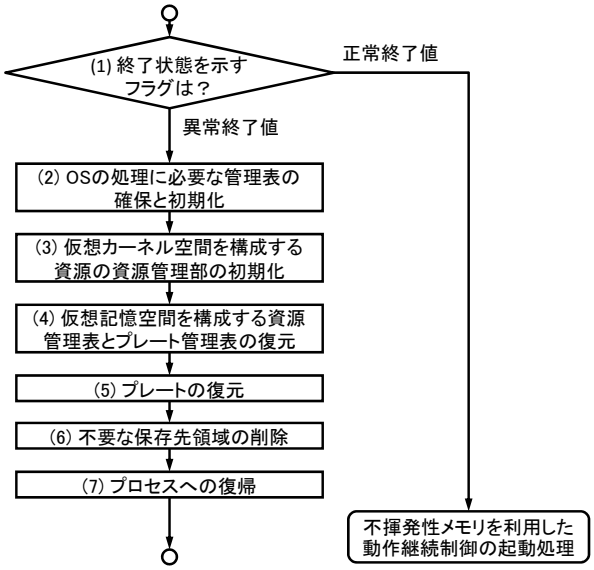

図 8 プレートの復元処理の流れ

3.4.3 プレートの復元

3.3 節を踏まえた不揮発性メモリ向けプレート機能におけるプレートの復元処理を**図 8** に示す．計算機起動時に終了状態を示すフラグの値を確認し（図 8(1)），異常終了値であれば，プレートの復元処理（図 4 の T_2 からの再開処理に相当）を行う．正常終了値であれば，不揮発性メモリを利用した動作継続制御による計算機状態の復元処理（図 4 の T_1 からの再開処理に相当）を行う．フラグの値は，起動処理時に異常終了値に設定する．計算機の終了処理を呼び出された場合，フラグの値を正常終了値に変更し，終了する．

プレートの復元処理を行う場合，不揮発性メモリ上に保存したプレートのデータを復元するために必要な初期化処理を実行する（図 8(2),(3)）．次に，特定の実アドレスに確保した保存先領域から仮想カーネル空間を構成する資源の資源管理表とプレート管理表を復元する（図 8(4)）．その後，復元したプレート管理表の情報を基に，仮想カーネル空間，仮想ユーザ空間の順にプレートを復元する（図 8(5)）．復元した計算機状態は，不要な保存先領域を削除する前の状態であるため，保存先領域を削除する（図 8(6)）．最後に，プレートの書き出しを要求したプロセスの処理を依頼

表 2　評価環境

CPU		Intel Core i3-6100T @ 3.20GHz
ハードディスク		TOSHIBA MQ01ABF050 (SATA 6Gb/s NCQ，5400rpm)
OS	ゲスト	***Tender***
	ホスト	Ubuntu 14.04 LTS (Linux 4.4.0)
メモリ	ゲスト	256MB
	ホスト	8GB
QEMU		2.1.0

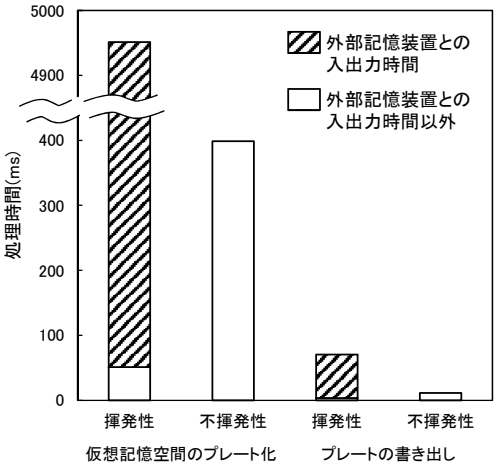

図 9　計算機状態保存処理の処理時間

した AP の処理に切り替わる（図 8(7)）．

4. 評価

4.1 内容

不揮発性メモリ向けプレート機能について，保存先領域を主記憶に確保することによって，揮発性メモリの場合とどの程度処理時間に差が出るかを示すために，以下の評価を行った．

（評価 1）計算機状態保存処理の処理時間
（評価 2）計算機状態復元処理の処理時間

評価環境を**表 2** に示す．評価には，不揮発性メモリの主記憶をエミュレーションする QEMU を利用した．計算機状態保存処理の処理時間では，仮想記憶空間上のデータの永続化と 1 回目のプレートの書き出し処理にかかった時間を測定する．評価では，起動処理終了後の計算機状態を保存し，復元するのにかかった時間を測定する．

4.2 計算機状態保存処理の処理時間

計算機状態保存処理の評価結果を**図 9** に示す．仮想記憶空間のプレート化について，外部記憶装置との入出力時間を除くと，処理時間に 347.36ms の差があることが分かる．また，揮発性の場合のプレート数と合計サイズは 170 個で 6,717KB，不揮発性の場合のプレート数と合計サイズは 456 個で 8,364KB となった．揮発性の場合，保存先領域を確保するために生成する資源は永続ユニットだけに対し，不揮発性メモリの場合，仮想カーネル空間と仮想領域を生成する．新しい資源を生成する際，OS が資源を管理するために利用する領域を新たに確保することがある．この新たに確保した領域についても保存先領域を確保する必要がある．不揮発性メモリの方が，保存先領域の生成によって資源の数が多くなるため，多くの仮想カーネル空間をプレート化する必要があり，外部記憶装置との入出力時間以外の処理時間が長くなる．

プレートの書き出し処理について，いずれも前回の保存処理からの変更点のみを保存する．揮発性の場合の変更量は 668KB，不揮発の場合の変更量は 1,400KB である．また，今回の評価では，仮想記憶空間のプレート化の直後にプレートの書き出し処理を実行したため，新たに生成する必要のあるプレートは存在しない．このため，仮想記憶空間のプレート化の処理時間よりも短くなっている．

なお，外部記憶装置との入出力時間を含めると不揮発性メモリの場合の方が，仮想記憶空間のプレート化の処理時間は 4,538.85ms（約 91.93%），1 回目のプレート書き出し処理の処理時間は 58.91ms（約 83.73%）短くなっていることが分かる．このことから，計算機状態保存処理において，外部記憶装置との入出力処理時間が大部分を占めることが分かる．

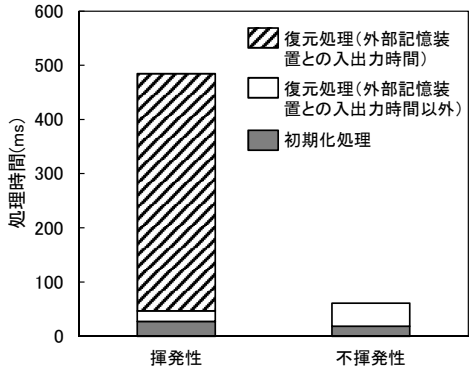

図 10　計算機状態復元処理の処理時間

4.3 計算機状態復元処理の処理時間

計算機状態復元処理の評価結果を**図 10** に示す．初期化処理について，不揮発性メモリの方が 9.01ms 短い．不揮発性メモリの場合，仮想カーネル空間を構成する資源の初期化のみでよい．これに対して，揮発性メモリの場合，永続ユニットなどのプレートを構成する資源の資源管理部の初期化が追加で必要となる．このため，初期化処理の処理時間が短い．

復元処理について，外部記憶装置との入出力時間を除くと，不揮発性メモリの方が 32.06ms 長い．計算機状態保存処理と同様，不揮発性メモリの場合の方がプレートの数が増えるため，処理時間が長くなる．

なお，外部記憶装置との入出力時間を含めると不揮発性

メモリの場合の方が，計算機状態復元処理全体で 423.75ms（約 87.48%）短くなっている．計算機状態保存処理同様に，復元処理も外部記憶装置との入出力処理時間の割合が大きいことが分かる．

5. 関連研究

不揮発性メモリを搭載した環境において，特殊なハードウェアを利用して，障害が発生しても障害発生前の処理を再開する研究に文献 [7]，[8]，[9] がある．文献 [7] は，電源監視専用のハードウェアを用意し，電源の切断を感知すると割り込みを発生させ，CPU のレジスタとキャッシュの内容を保存する．これにより，障害発生直前の処理を再開する．文献 [8] は，揮発性メモリと不揮発性メモリを主記憶に搭載する環境（以降，混載環境）において，チェックポイント生成によるオーバヘッドを削減するために，MMU を改変し，キャッシュ単位，ページ単位でデータを保存できるようにしている．文献 [9] は，コピーオンライトにかかる処理時間を削減するために，キャッシュライン単位でデータが更新できるように TLB を改変している．これらの研究に対して，不揮発性メモリ向けプレート機能は，特殊なハードウェアを利用しないため，使用する計算機に依存しない．また，文献 [8][9] について，永続化の対象は AP のみであり，不揮発性メモリ向けプレート機能のように OS の処理を永続化することはできない．

特殊なハードウェアを使用せず，障害発生前の処理を再開する研究として，文献 [10] がある．混載環境において，プロセス実行時に更新されたページの情報をログに追記し，計算機状態保存処理時にログの内容を不揮発性メモリ上にコピーする．復元処理では，不揮発性メモリ上に保存したプロセスの情報とログを基にメモリの内容を復元する．文献 [10] は，混載環境を想定し，不揮発性メモリを永続化にのみ使用するため，起動時にシステム全体の初期化が必要になる．これに対し，不揮発性メモリ向けプレート機能は，不揮発性メモリのみを利用するため，起動時には復元に必要な部分のみの初期化でよい．

6. おわりに

主記憶に不揮発性メモリを利用する環境におけるプレート機能の実現方法について述べた．この機能は，不揮発性メモリ上に AP の処理中の状態など計算機の実行途中の状態を保存する．これにより，外部記憶装置に保存するよりも高速な計算機状態の保存と復元を実現できる．また，計算機の緊急停止による計算機状態の消失を防ぐことができる．不揮発性メモリ上に確保した保存先領域から処理を継続するために，仮想記憶空間のプレート化時に保存先領域を確保する．また，仮想カーネル空間の削除に伴い，不要となった保存先領域は，保存先領域削除予定管理表に登録することで，保存先領域として利用する領域を上書きされることによって，処理が継続できなくなることを防ぐ．

計算機状態の保存処理の評価では，揮発性の場合と比較し，プレートの数が 286 個，プレートの合計サイズが 1,647KB 増えてしまうものの，仮想記憶空間のプレート化の処理時間を 4,538.85ms（約 91.93%），1 回目の保存処理の処理時間を 58.91ms（約 83.73%）短くなることを示した．また，復元処理の評価では，423.75ms（約 87.48%）短くなることを示した．

謝辞 本研究の一部は，共同研究（株式会社富士通研究所）による．

参考文献

[1] 谷口秀夫，青木義則，後藤真孝，村上大介，田端利宏：資源の独立化機構による ***Tender*** オペレーティングシステム，情報処理学会論文誌，Vol. 41, No. 12, pp. 3363–3374 (2000).
[2] Yamauchi, T., Yamamoto, Y., Nagai, K., et al.: Plate: Persistent Memory Management for Nonvolatile Main Memory, Proc. 31st ACM Symposium on Applied Computing (SAC 2016), pp.1885–1892, ACM (2016).
[3] 山本悠太，田端利宏，谷口秀夫：***Tender*** の動作継続制御機能における入出力デバイスの扱い，情報処理学会研究報告，2008-OS-109，Vol.2008，No.77，pp.61–68 (2008).
[4] 川岸 昇，渡辺 優，山内利宏，谷口秀夫：QEMU を利用した不揮発性メモリ搭載計算機の実現，平成 29 年度（第 68 回）電気・情報関連学会中国支部連合大会講演論文集，電子媒体 (2017).
[5] 長井健悟，山本悠太，山内利宏，谷口秀夫：***Tender*** の世代管理機能の実現，情報処理学会研究報告，Vol.2010-OS-115，No.2，電子媒体 (2010).
[6] 田中雅大，山内利宏，谷口秀夫：不揮発性メモリを利用した ***Tender*** における動作継続制御の実現，情報処理学会研究報告，Vol.2019-OS-147， No.16， pp.1–8，(2019).
[7] Narayanan, D., Hodson, O.: Whole-system persistence, Proc. the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2012), pp.401–410 (2012).
[8] Ren, J., Zhao, J., Khan, S., et al.: ThyNVM: enabling software-transparent crash consistency in persistent memory systems, Proc. the 48th International Symposium on Microarchitecture, pp.672–685 (2015).
[9] Ni, Y., Zhao, J., Bittman, D., et al.: Reducing NVM Writes with Optimized Shadow Paging, Proc. the 10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18), (2018).
[10] Kim, J., Moon, Y., Song, H., et al.: On Providing OS Support to Allow Transparent Use of Traditional Programming Models for Persistent Memory, ACM Journal on Emerging Technologies in Computing Systems, Vol.16, No.3, Article 33, pp.1–24, (2020).